Foundation Models for Decision Making: Algorithms, Frameworks, and Applications

Sherry Yang

Electrical Engineering and Computer Sciences
University of California, Berkeley

Technical Report No. UCB/EECS-2024-152
http://www2.eecs.berkeley.edu/Pubs/TechRpts/2024/EECS-2024-152.html

August 2, 2024

Foundation Models for Decision Making: Algorithms, Frameworks, and Applications

by

Mengjiao Sherry Yang

A dissertation submitted in partial satisfaction of the

requirements for the degree of

Doctor of Philosophy

in

Electrical Engineering and Computer Sciences

in the

Graduate Division

of the

University of California, Berkeley

Committee in charge:

Professor Pieter Abbeel, Chair
Professor Ruslan Salakhutdinov
Professor Ken Goldberg
Professor Sergey Levine

Summer 2024

Foundation Models for Decision Making:
Algorithms, Frameworks, and Applications

Abstract

Foundation Models for Decision Making:
Algorithms, Frameworks, and Applications

by

Mengjiao Sherry Yang

Doctor of Philosophy in Electrical Engineering and Computer Sciences

University of California, Berkeley

Professor Pieter Abbeel, Chair

AlphaGo and ChatGPT are perhaps two most significant breakthroughs in artificial intelligence in the past decade. These technologies were empowered by research in sequential decision making (e.g., planning, search, and reinforcement learning) and foundation models (e.g., language and video generation model trained on internet data). This thesis proposes new techniques, algorithms, and frameworks of leveraging foundation models with broad knowledge in the context of real-world decision making tasks, impacting applications such as building dialogue agent, controlling robots, and making scientific discoveries. This thesis starts with traditional decision making in offline settings and progressively incorporating broader, internet-scale data through representation learning and generative modeling. Emphasis is placed on both theoretical foundations and practical implications. Key contributions of this thesis include algorithmic advancements of offline reinforcement learning, improved representation learning for decision making, novel generative modeling techniques as an alternative to reinforcement learning, and generative agents and generative simulators at internet scale, all aimed at equipping foundation models with enhanced decision-making capabilities and vice versa. Through extensive empirical and theoretical analysis, this thesis demonstrates that foundation models, when properly leveraged, can significantly improve decision-making tasks. The findings offer new directions for integrating machine learning models with real-world applications, paving the way for more intelligent, adaptable, and efficient systems.

Acknowledgement

There are many people who have played important roles along my research journey for which I am grateful. Firstly, I want to thank my PhD advisor, Pieter Abbeel, for challenging me to difficult but important research problems, for encouraging me to pursue wild ideas and reminding me of the bigger picture, and for providing me with all kinds of resources and support despite some of which were unconventional. I want to thank Ofir Nachum for being an important mentor, who taught me the ins and outs of doing research and inspired me to work smarter, focus more, and think deeper. I want to thank my manager and informal advisor, Dale Schuurmans, for the opportunity to be on his team while I was new to research, and for always thinking on my behalf and trying his best to help me. I want to thank my thesis committee members, Ken Goldberg, Ruslan Salakhutdinov, and Sergey Levine, for all the insightful research discussions and critical assessment of my research work. I want to thank my undergraduate and master's advisor, Patrick Winston and Julian Shun, for encouraging me to pursue scientific research while I was unclear of which path to pursue.

I am grateful for many senior researchers from whom I have learned a great deal. I want to thank Bo Dai and Hanjun Dai, for their guidance during my first set of research projects and for showing me the level of diligence and rigor involved in good research. I want to thank Lihong Li and George Tucker for their research expertise during the projects we collaborated together, which inspired me to strengthen my own knowledge of the field and understanding of the problems. I am grateful for Been Kim with whom I pursued the first research project in artificial intelligence. I also want to thank many of my collaborators, including Yilun Du, David Venuto, Haoming Jiang, Justin Fu, Charlie Snell, Siddharth Verma, Yi Su, Ilya Kostrikov, Igor Mordatch, Ekin Dogus Cubuk, KwangHwan Cho, Amil Merchant, Ruiqi Gao, Brendan McMorrow, Danilo J. Rezende, Simon Batzner, Shiang Fang, Alex Gaunt, Muratahan Aykol, Jacob Walker, Jack Parker-Holder, Jake Bruce, Andre Barreto, Doina Precup, Ankit Anand, Pete Florence, Kamyar Ghasemipour, Fei Xia, Ayzaan Wahid, Brian Ichter, Pierre Sermanet, Tianhe Yu, Andy Zeng, Jonathan Tompson, and Leslie Kaelbling for our discussions and collaborations on an broad array of projects from leveraging offline RL in dialogue systems to using generative models to solve robot learning tasks and discover novel crystal structures. I want to thank the Robot Learning Lab at UC Berkeley and members of the broader Berkeley AI Research for discussions in the hallway and late night grinds right before conference deadlines.

I am thankful for having a strong support group from friends both at work and from table tennis. I want to thank Hieu Pham for the initial encouragement for me to pursue a PhD, for the many insightful research and "political" discussions we have had throughout various "threading sessions". I want to thank Jessie Zhang for all the personal conversations and advice, and for all the adrenaline-inducing table tennis drills which gave me a sense of progress when I felt stuck at research. I want to thank William Zheng for his patience and willingness to train together for table tennis during COVID, which helped me stay sane during the lockdowns. I want to thank Iveel Tsogsuren and Stacy Ho, whom I called up

many times to sort out difficult thoughts.

Lastly, I want to express my utmost gratitude to my parents, Wenjuan Zhao and Xi Yang, for their unconditional love and support, for treating me as their equal, and for their cultivation of my problem solving skills, independence, and diligence at a young age. Without them, I would have not come this far.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

In the past decade, two significant breakthroughs in artificial intelligence (AI) include artificial Go player AlphaGo outperforming human player Lee Sedol in 2016 [21], and an artificial chatbot ChatGPT being deployed in 2022 [22]. The technologies that empowered these advances are research in sequential decision making and foundation models. In sequential decision making, the goal is to have computers (agents) automatically decide on a sequence of actions (e.g., where to place the Go pieces), and also have computers automatically improve these decisions based on feedback from the environment (e.g., outcome of a Go game). The machine learning approach to sequential decision making involves training a decision-making policy, i.e., a strategy for choosing actions based on the current observation (e.g., the Go board), through trial and error. This approach works well in game settings where the environment supports unlimited access, but struggles to scale beyond game settings in the real world where unlimited access to the environment is impractical. Even in game settings, prior work in sequential decision making has largely focused on task-specific or *tabula rasa* settings with limited prior knowledge [23]. As a result, prior work in sequential decision making generally struggles with generalization and sample efficiency, e.g., requiring 7 GPU days of interactive game-play to solve a single Atari game [24].

More recently, foundation models, defined as large machine learning models trained at scale using self-supervised learning [25], have been trained on large amounts of data from the internet. For instance, autoregressive language models [26, 27] are trained to predict the next word (token) given previous words (tokens) using text data crawled from the internet. Similarly, video generation models [28, 29] are trained to predict the next frame given language input and/or previous frames using videos from the internet. As a result, these models could generate highly realistic natural language and videos. Nevertheless, imitating content on the internet is not the final goal of these models. The final goal for these models is to solve real-world tasks such as answering people's questions and simulating real-world interactions. To achieve this goal, the generated content of these models have to be controllable by humans. How to steer these models to generate desirable content according to user feedback and how to enable these models to make a sequence of decisions that accomplish some complex task (e.g., building a website) lies at the center of sequential decision making.

It is highly beneficial to jointly consider research in foundation models and sequential decision making. On one hand, broad knowledge from foundation models can improve the sample efficiency and generalization of decision making algorithms. On the other hand, decision making algorithms can enable task-specific optimization of otherwise task-agnostic foundation models. This thesis studies techniques, frameworks, and algorithms at the intersection of *foundation models for decision making*, and shows that *broad knowledge from foundation models can be effectively transformed into task-specific decisions* to better solve a wide class of problems and applications.

This thesis approaches foundation models for decision making by starting with traditional decision making techniques in the setting with offline dataset, followed by incorporation of broader data, until eventually internet-scale vision and language data is incorporated. Significant attention will be paid to both the theoretical aspects as well as the practical implications of leveraging foundation models to solve sequential decision making problems. The work in this thesis builds on ideas from previous research on sequential decision making, but the newly proposed approaches illustrate additional comprehensiveness and scalability. The remainder of this chapter is organized as follows. Section 1.1 introduces foundation models, which is type of machine learning models trained on internet-scale data. This section discusses common techniques for training foundation models, including representation learning and generative modeling. This section then describes limitations in foundation models, including instruction following, long-horizon reasoning, multi-step planning, and multi-modality. It then gives an overview of how this thesis addresses some of these challenges by incorporating decision making techniques. Section 1.2 describes the canonical setting of sequential decision making and common decision making algorithms including imitation learning, reinforcement learning, search, and planning. The section then highlights the major bottleneck of sequential decision making, including sample efficiency and lack of good visual and textual representations. It then gives a glimpse into how this thesis address these challenges by incorporating foundation models. Section 1.3 states the contribution of this thesis and summarizes its organization.

1.1 Internet-Scale Knowledge and Foundation Models

One goal of AI can be viewed as automation, e.g., chatbots can automatically answer people's questions, self-driving cars and robots can automatically operate without human intervention. Such a goal can be broken down into two parts: knowledge and action, e.g., chatbots need to know about the particular questions being asked (knowledge) before coming up with an answer (action), self-driving cars and robots need to understand the surrounding objects (knowledge) before moving to a particular location (action). While the best actions depend on the specific tasks or user requests, knowledge about the world is general and can be shared across tasks. Prior to the development of foundation models, it was not clear how to equip AI with broad knowledge about the world.

Nevertheless, broad knowledge exist on the internet in the form of textual and visual

data. For instance, Wikipedia articles contain detailed information about specific topics, which often contain answers to people's inquiries about specific subjects. Similarly, tons of driving videos and videos of humans performing household activities exist on the internet, which could inform self-driving cars and robots of what to do in certain situations. The problem of automation can then be naturally broken down to (1) how to inject these internet-scale knowledge into machine learning models, so that models can elicit relevant knowledge when solving specific tasks just like humans do, and (2) how to decide on a set of actions that actually solve specific tasks using previously acquired knowledge. The terminology of *foundation models* was introduced to emphasize the learning objective and scale of (1). Specifically,

> *A foundation model is any model that is trained on broad data (generally using self-supervision at scale) that can be adapted (e.g., fine-tuned) to a wide range of downstream tasks [25].*

To understand this definition of foundation models further, we focus on introducing two common self-supervised objectives of foundation models: representation learning or generative modeling. We introduce these two objectives below.

1.1.1 Representation Learning

Representation learning has been studied for decades and have many different interpretations such as dimension reduction [30], sparse coding [31], manifold learning [32], and so on. Generally speaking, representation learning uses a different objective, often called the representation learning objective, from the downstream task-specific objective. The high-level idea behind representation learning objective is to learn some embedding representations of the raw input (e.g., image or language), so that such these representation can capture broad knowledge from the pretraining data. For instance, one common representation learning objective for modeling language is denoising autoencoding, as in Bidirectional Encoder Representations from Transformers (BERT) [1]. In BERT models, word tokens are encoded into high-dimensional embedding representations, and trained by predicting neighboring word tokens. In another word, good representations of words should enable the prediction of neighboring words. As a result, this denoising autoencoding objective can successfully inject knowledge into these pretrained representations by forcing them to contain information that is helpful to predicting neighboring words. In addition to denoising autoencoding, another example of representation learning objective in foundation models is contrastive learning [8]. Contrastive learning can be seen as encouraging the similarity of embeddings between similar examples and discourage similarity of embeddings between dissimilar examples. For instance, Contrastive Language-Image Pretraining (CLIP) [2] encourages the representation of the image of a dog and the word "dog" to be similar, while discourage the representation of a dog image to be similar to the word "cat". Figure 1.1 illustrates the above two representation learning objectives, denoising autoencoding and contrastive learning. Note

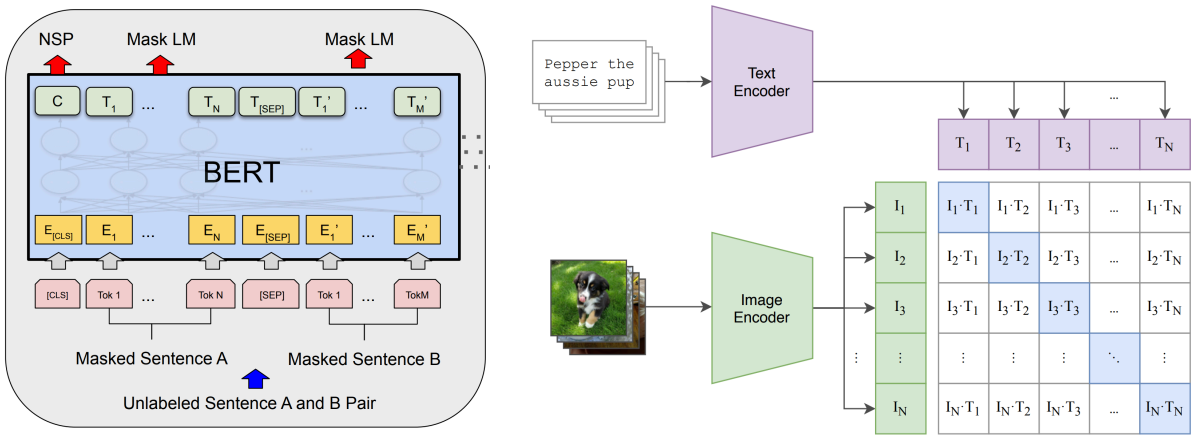


Figure 1.1: **Example objectives of representation learning** commonly used for training foundation models. In BERT [1] (left), representations of word tokens encourage predictions of neighboring words. In CLIP [2] (right), representations of the word "dog" and the an image of a dog are encouraged to be similar.

that autoencoding and contrastive learning have long been studied, but scaling such representation learning objectives to internet-scale datasets only happened in recent years, where BERT and CLIP models are trained on text and images crawled from the internet.

Note that representation learning alone is generally not sufficient for solving any specific tasks. The learned representations are usually finetuned on downstream objectives. For instance, BERT representations of word tokens can be finetuned on sentiment classification tasks, while CLIP representations of images can be finetuned on image classification tasks. The mismatch between the representation learning objective and the downstream task-specific objectives can some time limit the capabilities of representation learning approaches.

1.1.2 Generative Modeling

Different from representation learning in Section 1.1.1, generative modeling approach to pretraining foundation models does not learn embedding representations of input data directly. Instead, generative modeling focuses on learning the input distribution using approximate inference or sampling approaches. In approximate inference, a known function class is learned from data to approximate the input distribution. For instance, variational autoencoding (VAE) is the most canonical approximate inference approach to generative modeling, which uses a simpler tractable distribution (often Gaussian) to approximate the posterior distribution of the latent variables. In the case where the data distribution is highly intractable, one can restore to sampling methods such as Monte Carlo techniques. The sampling ap-

proach can be particularly effective when the data distribution it self is not of interest, but being able to sample from the learned distribution is of interest. For instance, diffusion models [33, 34, 35] rely on sampling methods. They work by defining a forward diffusion process, which progressively adds noise to the data, and a reverse diffusion process, which gradually removes this noise to generate new data samples from the model.

In addition to approximate inference and sampling based methods as in VAEs and diffusion models, self-supervised learning through next token prediction has been shown effective in modeling language. In large language models (LLMs) such as Generative Pretrained Transformers (GPTs) [26, 27], are trained using a self-supervised learning approach where the model learns to predict the next token in a sequence given the previous tokens. This process involves maximizing the likelihood of the observed data (text sequences) by adjusting the model parameters to increase the probability of the correct next token.

One of the advantages of generative modeling being the pretraining objective is that pretraining and downstream task-specific learning can use the same training objective. For instance, next token prediction is both the pretraining and the finetuning objective for LLMs. The aligned objective between pretraining and finetuning often simplifies training infrastructure and preserves knowledge from pretraining during the finetuning stage.

1.2 From Knowledge to Action via Decision Making

So far, we have discussed how foundation models can acquire broad knowledge from internet data, i.e., through self-supervised learning at scale using representation learning or generative modeling objectives. However, like we have discussed earlier in this section, acquiring knowledge is only the first step to artificial intelligence. The next step is to derive the appropriate *actions* using previously acquired knowledge, so that the derived action can achieve some desirable goal or task. The process of deriving such desirable action is often times referred to as *decision making*. In the recent research of large language models, supervised finetuning is often employed directly to finetune pretrained LLMs on task-specific supervised data, such as question and answer pairs. Using next-token prediction, LLMs can be simply finetuned to predict the answer given specific questions. However, such naïve finetuning approach often fell short in more complex problems that require extended reasoning, long-horizon planning, and more strategic decision making. These problems often require models to understand the long-term effect of actions, to be able to reason through the consequence of a combination of action sequences, and to strategically select actions that maximizes the long-term gain. As a result, the next-token-prediction objective is no longer sufficient.

The problem of optimal decision making lies at the core of sequential decision making [36], encompassing areas such as reinforcement learning, imitation learning, planning, search, and optimal control. Contrary to the paradigm of foundation models, where broad datasets with billions of images and text tokens are used during pretraining, prior work on sequential decision making has largely focused on task-specific or *tabula rasa* settings with limited prior knowledge [23]. Despite a seemingly disadvantageous setup, research in sequential decision

making has achieved significant progress in surpassing human performance on tasks such as playing board games [37] and Atari video games [38], as well as operating robots to complete navigation [39] and manipulation tasks [40, 41]. Nevertheless, since these methods learn to solve a task from scratch without broad knowledge from vision, language, or other datasets, they generally struggle with generalization and sample efficiency, e.g., requiring 7 GPU days of interactive game-play to solve a single Atari game [24]. Intuitively, broad datasets similar to those used for foundation models should also be beneficial for sequential decision making models. For example, there are countless articles and videos on the Internet about how to play Atari games. Similarly, there is a wealth of knowledge about properties of objects and scenes that would be useful to a robot, or about human wants and emotions that could improve a dialogue model.

While research on foundation models and sequential decision making has largely been disjoint due to distinct applications and foci, there is increasing activity at the intersection of these communities. On the foundation models side, with the discovery of emergent properties of large language models, target applications have graduated from simple zero or few-shot vision and language tasks to problems that now involve long-term reasoning [42, 43, 44] or multiple interactions [45]. Conversely, in the sequential decision making communities, researchers inspired by the success of large scale vision and language models have begun to curate ever-larger datasets for learning multimodel, multitask, and generalist interactive agents [46, 47, 48, 19, 49, 3]. Further blurring the lines between the two fields, some recent work has investigated the use of pretrained foundation models such as CLIP [2] to bootstrap the training of interactive agents for visual environments [50, 51], while other work has investigated foundation models as dialogue agents optimized by reinforcement learning with human feedback [22], and other work has adapted large language models to interact with external tools such as search engines [52, 53, 54, 55, 56], calculators [57, 53], translators [53], MuJoCo simulators [58], and program interpreters [59].

1.3 Thesis Statement and Organization

1.3.1 Thesis Statement

Despite early signs of success listed above, foundation models for decision making remain largely *underexplored, underutilized, and lacking solid empirical and theoretical grounding.* The challenges faced by existing research are as follows:

- Many traditional decision making benchmarks are (near-)Markovian (i.e., historyless), and this brings the value of sequence modeling into question. The true power of foundation models may require more complex tasks.

- Decision making tasks are composed of multi-modal data. At minimum, the states (observations), actions, and rewards of a task are each of different types. Moreover, across

different tasks, states and actions can be highly distinct (image vs. text observations, discrete vs. continuous actions).

- Unlike vision and language, decision making agents can further interact with the environment to collect additional experience in conjunction with learning on existing data. How such an interactive component should be integrated with foundation models is not clear.
- Decision making algorithms such as RL already exhibits a large gap between theory and practice. Hastily applying large models to decision making might create an even greater gap.

In addition to these challenges, there are a number of open questions at the intersection of foundation models and decision making:

- Can language model agents be developed to *automatically* learn to interact with humans, computers, tools, the world, and each other in a scientific and principled way?
- How can environments and tasks be structured so that vision language foundation models can benefit traditional decision making applications in control, planning, and reinforcement learning?
- Can sound, practical, and scalable algorithms be derived, similar to RLHF and MCTS for language and vision based decision making applications?
- Given that foundation models are trained on data without actions, how can this limitation be overcome from both the dataset and modeling perspectives?

This thesis seeks to address these challenges arising in the paradigm of foundation models for decision making as well as answering these open questions, in order to equip foundation models with the ability to conduct extended reasoning, planning, and search capabilities, as well as accelerating traditional decision making tasks with broad knowledge acquired from foundation models. Representation learning, generative modeling, and decision making algorithms such as RL are closely interrelated, and therefore effective solutions require attention to all three areas. Figure 1.2 illustrate the overall framework of foundation models for decision making studied by this thesis. This thesis provides evidence to support the following statement:

Thesis statement: *With appropriate techniques, frameworks, and algorithms, foundation models for decision making can be efficient, scalable, and empowering, both in theory and in practice.*

We hope that the frameworks, tools, algorithms, and concepts introduced in this thesis will stimulate further research in foundation models for decision making, leveraging extensive capabilities of foundation models to tackle more intricate tasks. The code created during

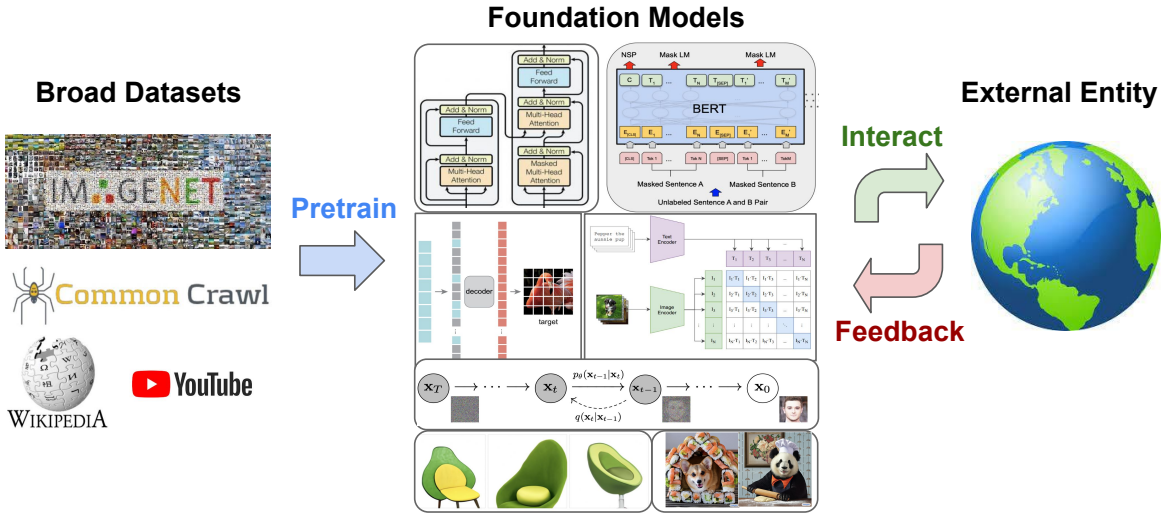


Figure 1.2: **Foundation models for decision making** illustration. Foundation models are first pretrained on internet-scale data using self-supervised learning, and are then employed to solve real-world tasks through interacting and learning from feedback of external entities such as the real world.

this thesis is openly accessible and has already been utilized by numerous researchers for benchmarking and developing their own decision-making algorithms using foundation models.

1.3.2 Thesis Organization

This thesis studies the frameworks, techniques, algorithm design, and performance analysis of leveraging foundation models in decision making problems. The results in this thesis are developed in collaboration with various co-authors: Pieter Abbeel, Ofir Nachum, Dale Schuurmans, Bo Dai, Hanjun Dai, Yilun Du, Servey Levine, George Tucker, Igor Mordatch, Charlie Snell, Justin Fu, Doina Precup, Jonathan Tompson, Yi Su, and David Venuto. The following paragraphs describe the organization and contributions of this thesis.

Chapter 2 introduces the necessary definitions and notation used throughout the thesis, as well as briefly reviews relevant concepts in decision making and generative modeling. This chapter also formulates the problem of foundation models for decision making. Then, Chapter 3 introduces algorithmic improvements to traditional RL methods to incorporate additional offline interactive data. Chapter 4 explores using offline data to learn compact representations of observations and actions, thereby accelerate downstream learning. Chapter 5 considers training conditional generative models. Finally, Chapter 6 leverages internet text and video data to learn large video and language policies and environments. The chapters are generally organized both in the chronological order of the works developed, and

also in the order of increasing model scale and data scale/diversity. For instance, Chapter 3 and Chapter 4 only considers offline interactive data following the strict form of sequential decision making, while Chapter 5 incorporates broader data such as data with auxiliary information. Finally, Chapter 6 considers internet-scale video and language datasets.

The contributions of this thesis include:

1. The first unified framework of Distribution Correction Estimation (DICE) for off-policy policy evaluation and selection (Chapter 3).
2. The first comprehensive empirical study of different representation learning objectives (e.g., contrastive, bisimulation, model-based) combined with different downstream learning objectives (imitation, online/offline RL).
3. The first proofs that contrastive learning of state representations with random Fourier features can improve the sample complexity of downstream imitation learning (Chapter 4).
4. The first proofs that contrastive learning of action representations with random Fourier features can improve the sample complexity of downstream imitation learning (Chapter 4).
5. The first to identify and solve the failure mode of Decision Transformer [60] in stochastic environments with provable guarantees (Chapter 5).
6. A novel technique called procedure cloning for incorporating auxiliary data into imitation learning to improve generalization (Chapter 5).
7. The first to formulate text-conditioned video generation as a universal planning strategy from which diverse behaviors can be synthesized (Chapter 6).
8. A novel framework for adapting pretrained video generation models to task-specific domains (Chapter 6).
9. The first to demonstrate that text-and-image conditioned video generation can serve as a universal simulator from which diverse actions and environment transitions can be simulated (Chapter 6).

Chapter 2

Preliminaries and Notation

In this chapter, we define notations and review relevant background on sequential decision making and generative modeling. Definitions and notations from this chapter will be used throughout the thesis. Individual chapters have additional definitions and notation that are specific to the chapter. Furthermore, we also outline the problem space of decision making with foundation models.

2.1 Sequential Decision Making Preliminaries

2.1.1 Markov Decision Process

Sequential decision making problems are most often formalized in terms of a Markov decision process (MDP) [61], which is defined as a tuple $\mathcal{M} := \langle S, A, \mathcal{R}, \mathcal{T}, \mu, \gamma \rangle$ consisting of a state space S, an action space A, a reward function $\mathcal{R} : S \times A \to \Delta(\mathbb{R})$,[1] a transition function $\mathcal{T} : S \times A \to \Delta(S)$, an initial state distribution $\mu \in \Delta(S)$, and a discount factor $\gamma \in [0, 1)$. A policy $\pi : S \to \Delta(A)$ interacts with the environment starting at an initial state $s_0 \sim \mu$. At each timestep $t \geq 0$, an action $a_t \sim \pi(s_t)$ is sampled and applied to the environment, after which the environment transitions into the next state $s_{t+1} \sim \mathcal{T}(s_t, a_t)$ while producing a scalar reward $r_t \sim \mathcal{R}(s_t, a_t)$.[2]

After π interacts with $\mathcal{M}$ for H timesteps (H can be infinite), an episode (trajectory) is produced $\tau := \{(s_0, a_0, r_0), (s_1, a_1, r_1), \ldots, (s_H, a_H, r_H)\}$. We use τ_t to denote the tuple (s_t, a_t, r_t), $\tau_{<t}$ to denote a sub-episode up to timestep t, $\tau_{\geq t}$ to denote a sub-episode starting from timestep t and ending at H, $\tau_{t:t+h}$ to denote a sub-episode from timestep t to $t + h$, and τ_s or τ_a to denote only the state or action portion of a trajectory. The return associated with episode τ is defined as the total discounted sum of rewards $R(\tau) := \sum_{t=0}^{H} \gamma^t r_t$. The

[1] $\Delta(\mathcal{X})$ denotes the simplex over a set $\mathcal{X}$.

[2] We will focus on fully observable MDPs in this chapter, though an MDP can be extended to a partially observable MDP (POMDP) by introducing an observation space $\mathcal{O}$, an emission function $\mathcal{E} : S \to \mathcal{O}$, and the restriction that policies can only depend on observations and previous actions. More details of POMDPs will be introduced in later chapters.

trajectory distribution of a policy $p_\pi(\tau)$ is determined by

$$p_\pi(\tau) = \mu(s_0)\Pi_{t=0}^{H}\pi(a_t|s_t)\mathcal{R}(s_t,a_t)\mathcal{T}(s_{t+1}|s_t,a_t). \tag{2.1}$$

Trajectories generated by one or multiple policies can be collected in an offline dataset $\mathcal{D}_{\text{off}} = \{\tau\}$. We can also write an offline dataset as a set of transition tuples $\mathcal{D}_{\text{off}} = \{(s_i, a_i, r_i, s_i')\}_{i=1}^{M}$, where M is the number of transitions in the dataset. Whether to treat the offline interactive dataset as a set of sequences or a set of transition tuples depends on the problem setting.

2.1.2 Imitation Learning

In standard imitation learning, $\mathcal{R}$, $\mathcal{T}$, and μ are unknown to the agent. Learning solely takes place from a fixed dataset of demonstrations $\mathcal{D}_{\text{off}}^* = \{(s,a)\}$ previously collected by an expert policy π^* interacting with $\mathcal{M}$ through $a \sim \pi^*(s)$. The goal of imitation learning is to train π on $\mathcal{D}_{\text{off}}^*$ so that π closely approximates π^* according to some metric, such as the Kullback–Leibler (KL) divergence between the trajectory distributions $D_{\text{KL}}(p_{\pi^*}(\tau)\|p_\pi(\tau))$.

Behavioral cloning (BC). Learning from expert demonstrations leads to the common framing of imitation learning as supervised learning of state to action mappings. Under this framing, behavioral cloning (BC) [62] proposes to learn π by minimizing

$$\mathcal{L}_{\text{BC}}(\pi) := \mathbb{E}_{(s,a)\sim\mathcal{D}_{\text{off}}^*}[-\log\pi(a|s)]. \tag{2.2}$$

Equation 2.2 can be viewed as the classification loss (discrete actions) or regression loss (continuous actions) of state to action mappings, connecting BC to supervised learning in vision and language.

2.1.3 Reinforcement Learning

Standard reinforcement learning [36] aims to maximize the expected returns of a policy through trial-and-error interaction with the environment:

$$J(\pi) := \mathbb{E}\left[\sum_{t=0}^{H}\gamma^t r_t \middle| \pi, \mathcal{M}\right]. \tag{2.3}$$

Policy-based methods. One conceptually straightforward way to optimize Equation 2.3 is through policy gradient, which estimates the gradient of Equation 2.3 with respect to the policy π, and maximizes $J(\pi)$ directly via gradient ascent. The most commonly used gradient estimator has the form

$$\nabla_\theta J(\pi_\theta) = \mathbb{E}_{\tau\sim p_{\pi_\theta}(\tau)}\left[\sum_{t=0}^{H}\gamma^t\nabla_\theta\log\pi_\theta(a_t|s_t)\hat{A}(s_t,a_t)\right], \tag{2.4}$$

where $\hat{A}$ is some advantage function that can be separately estimated via Monte-Carlo returns from $p_\pi(\tau)$ [63]. The biggest drawback of policy gradient is sample inefficiency: since policy

gradients are estimated from rollouts, the variance of the gradient estimate is often extreme. To mitigate high variance, various works such as PPO [64] have proposed to improve policy updates through the use of appropriate geometry [65, 66, 67] or through training a separate critic network to estimate $\hat{A}$ to futher reduce variance at the cost of introducing bias [68, 69, 70].

Value-based methods. Another family of reinforcement learning methods for optimizing Equation 2.3, such as Q-learning [71], involves learning the optimal value function $Q^*(s_t, a_t)$ by satisfying a set of Bellman *optimality* constraints:

$$Q^*(s_t, a_t) = r_t + \gamma \mathbb{E}_{s_{t+1} \sim \mathcal{T}(s_{t+1}|s_t, a_t)} \left[\max_{a_{t+1}} Q^*(s_{t+1}, a_{t+1}) \right], \tag{2.5}$$

after which an optimal policy can be extracted via $\pi^*(\cdot|s_t) = \arg_a \max Q^*(s_t, a)$. Value-based methods are typically more sample efficient than policy-based methods [72], but tend to be unstable under function approximation [36]. At the intersection of policy and value based methods, Actor-Critic methods [68] first learn $Q^\pi(s_t, a_t)$ by satisfying the set of Bellman *expectation* constraints:

$$Q^\pi(s_t, a_t) = r_t + \gamma \mathbb{E}_{s_{t+1} \sim \mathcal{T}(s_{t+1}|s_t, a_t), a_{t+1} \sim \pi(s_{t+1})} \left[Q^\pi(s_{t+1}, a_{t+1}) \right], \tag{2.6}$$

then plug $\hat{A}(s_t, a_t) = Q^\pi(s_t, a_t)$ into the policy gradient objective, Equation 2.4, to update the policy, so that learning the resulting policy will be both stable and sample efficient.

Off-policy and offline RL. To further improve the sample efficiency of on-policy methods, a set of off-policy approaches have been proposed for both policy and value based RL [73, 74, 75], where data from sources other than the current policy can be utilized for learning in conjunction with environment interaction. Offline RL [76] further considers the setting where an agent only has access to a fixed dataset of previous interactions $\mathcal{D}_{\text{off}}$, and no further environment access to $\mathcal{T}$ or $\mathcal{R}$ is available. To ensure the learned policy avoids out-of-distribution states and actions, offline RL methods often impose regularization via a divergence between the learned policy and the offline dataset [77] or on the learned value function [78]. More recently, some works have explored using additional online access as a finetuning step after offline RL to improve sample efficiency [79, 80, 81].

Using foundation models for decision making differs from traditional offline RL (with or without online finetuning) in that the latter focuses on learning RL algorithms from task-specific RL datasets $\mathcal{D}_{\text{off}}$ (i.e., datasets with task-specific states, actions, and rewards), whereas the former focuses on self-supervised learning on diverse data (e.g., data from vision and language domains) followed by task-specific adaptation.

2.1.4 Planning, Search, and Optimal Control

Unlike the model-free RL algorithms outlined above, a broader set of approaches to sequential decision making (e.g., planning, search, optimization-based control, model-based

RL) leverage explicit models of the environment. When the true environment dynamics are known (e.g., the rules of a Chess game) and simulation is cheap, planning and search algorithms, such as MCTS [82] that leverage an accurate simulator, can be highly effective [21]. When the environment can be characterized by precise dynamics, such as the constrained movements of a robot arm, approaches in optimal control—such as trajectory optimization [83], shooting [84], collocation [85], and model predictive control [86]—have long been studied prior to the recent advances in deep learning. In deterministic scenarios, given an environment governed by a known dynamics function $s_{t+1} = f(s_t, a_t)$, optimizing a sequence of actions $a_{0:T}$ to execute in the environment corresponds to

$$a_{0:T} = \arg\max_{a_{0:T}} J(s_0, a_{0:T}) = \arg\max_{a_{0:T}} \sum_{t=0}^{T} r(s_t, a_t) \text{ subject to } s_{t+1} = f(s_t, a_t). \tag{2.7}$$

Model-based RL [87] considers the setting where the environment dynamics are unknown and have to be estimated from samples, after which techniques from search, planning, and optimal control [87, 88, 89, 90, 91] can be effectively applied given the learned dynamics model.

2.2 Generative Models Preliminaries

Many foundation models can be characterized as modeling a (conditional) density $p(x)$ on a large dataset of images or texts $x \sim \mathcal{D}$. For example, x could be an image, a sequence of images, or a sequence of text tokens. Different foundation models differ in their factorizations of $p(x)$. Below, we provide a brief overview of several generative models and their factorizations of $p(x)$.

Latent Variable Models Latent variable models factorize the unknown data distribution of interest $p(x)$ into a latent variable distribution and a conditional distribution:

$$p(x) = \int p(z)p(x|z)dz, \tag{2.8}$$

where the latent variable z can be both discrete or continuous. For the special cases when z is discrete and the sum is tractable, or z is continuous and the integral is tractable, one can simply calculate $p(x)$ in closed form to support efficient maximum likelihood estimation on a given dataset. However, for the more general cases when the requisite sum or integral is intractable, techniques like VAEs [92] are applied to optimize the evidence lower-bound (ELBO) of $p(x)$ using a variational posterior $q(z|x)$:

$$\mathcal{L}_{\text{VAE}}(p, q) = \mathbb{E}_{x\sim\mathcal{D}, z\sim q(z|x)} \left[-\log p(x|z)\right] + \mathbb{E}_{x\sim\mathcal{D}} \left[D_{\text{KL}} \left(q(z|x) \| p(z)\right)\right]. \tag{2.9}$$

As an extension of VAE, VQ-VAE [93] uses a codebook to discretize the continuous latent representation to learn a more compact, discrete representation of the data.

Autoregressive Sequence Models Autoregressive sequence models have been popularized by transformer-based language models [94, 27]. At their core, autoregressive models factorize any joint distribution over a sequence $x = (x_1, ...x_L)$ in an autoregressive manner:

$$p(x) = \Pi_{\ell=1}^{L} p(x_\ell | x_{<\ell}). \tag{2.10}$$

Under this factorization, estimating the density $p(x)$ reduces to learning each conditional factor $p(x_\ell|x_{<\ell})$ which can be parametrized by a transformer.

$$\mathcal{L}_{\text{LM}}(p) = \mathbb{E}_{x\sim\mathcal{D}} \left[\sum_{\ell=1}^{L} -\log p(x_\ell|x_{<\ell}) \right]. \tag{2.11}$$

Diffusion Models Diffusion models [33, 35, 95] are a class of latent variable models that factorize the data distribution $p(x)$ as a Markov chain of Gaussian transitions from a noise distribution of the same dimension:

$$p(x) = \int p(x_K)\Pi_{k=1}^{K} p(x_{k-1}|x_k) dx_{1:K}, \tag{2.12}$$

where $p(x_K) = \mathcal{N}(\mathbf{0}, \mathbf{I})$ and $p(x_{k-1}|x_k) := \mathcal{N}(\mu(x_k, k), \sigma(x_k, k))$. The forward diffusion process corrupts x by iteratively adding Gaussian noise with a fixed variance schedule. The reverse process then achieves data generation by approximating the noise that corrupted x during the forward process.

2.2.1 Energy-Based Models

Energy-based models [96, 97] are a class of models that represent data distributions $p(x)$ by an unnormalized distribution parameterized by a learned energy function:

$$p(x) = \frac{e^{-E(x)}}{Z}, \tag{2.13}$$

where E is the energy function and $Z = \int e^{-E(x)} dx$ is the partition function. To sample from the underlying distribution $p(x)$, one typically runs an MCMC procedure such as Langevin dynamics to sample from the underlying distribution.

2.3 Foundation Models as Conditional Generative Models

We now examine the first concrete use case of foundation models in decision making: probabilistic modeling of the trajectory distribution $p(\tau)$ from an interactive dataset $\tau \sim \mathcal{D}_{\text{off}}$. Depending on what part of τ is being modeled, foundation models can serve as conditional generative models of behaviors (i.e. actions) or the underlying world models (i.e., environment dynamics). Below, we first review different generative models and then discuss and explore how they can be used to represent behaviors and models of the environment.

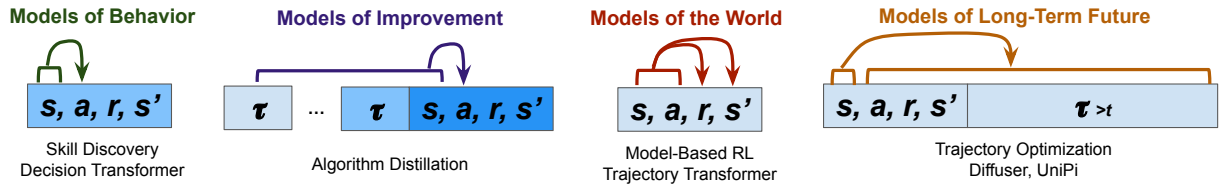


Figure 2.1: Illustrations of how conditional generative models can model behaviors, improvements, environments, and long-term futures given a trajectory $\tau \sim \mathcal{D}_{\text{off}}$. Dark blue indicates transitions with higher rewards. Models of behavior (Decision Transformers [3]) and self-improvement (Algorithm Distillation [4]) require near-expert data. Models of the world (Trajectory Transformer [5]) and long-term future (UniPi [6]) generally require data with good coverage.

2.3.1 Generative Models of Behavior

The generative models introduced above have mostly been applied to text or image data $x \sim \mathcal{D}$. Decision making, on the other hand, is concerned with task specific *interactive* data $\tau \sim \mathcal{D}_{\text{off}}$ that distinguishes state, action, and reward labels. We will see how different generative models can be adopted to model agent behaviors (this subsection) and environment dynamics (next subsection), as illustrated in Figure 2.1.

2.3.1.1 Foundation Models as Behavioral Priors

When the interactive data $\mathcal{D}_{\text{off}}$ contains diverse behaviors such as "pick up objects", "move objects horizontally", or "place objects", these behaviors can be composed to complete tasks that were not present in $\mathcal{D}_{\text{off}}$. Foundation models can be used to model such "behavioral priors" (also known as "skills" or "options"). In this approach, pretraining generally involves maximum likelihood estimation of actions conditioned on some trajectory level information. Different tractable approximations can be leveraged to optimize this underlying training objective. For instance, the VAE objective from Equation 5.7 can be directly instantiated, where the encoder q takes a trajectory τ or some future goal as input and the decoder π produces the sequence of actions as outputs [14, 98]:

$$\mathcal{L}_{\text{VAE}}(\pi, q) = \mathbb{E}_{\tau \sim \mathcal{D}_{\text{off}}, z \sim q(z|\tau)} \left[\sum_{t=0}^{H} -\log \pi(a_t | s_t, z) \right] + \mathbb{E}_{\tau \sim \mathcal{D}_{\text{off}}} \left[D_{\text{KL}}(q(z|\tau) \| p(z|s_0)) \right]. \quad (2.14)$$

The posterior distribution $q(z|\tau)$ can represent a diverse set of behavioral priors when τ is drawn from a wide set of related tasks. Since the posterior depends on future information, the prior $p(z|s_0)$ is usually constrained to only depend on the past so that behaviors can be correctly sampled at test time.

Similarly, the autoregressive sequence modeling objective from Equation 2.11 can also be instantiated to model behavioral priors [99], resulting in a policy that can depend on the

history of interaction $\pi(a_t|s_t, \tau_{<t})$. Such dependence is less common in Markovian environments, but has shown empirical benefits [19]. When the dataset consists of expert data $\mathcal{D}^*_{\text{off}}$, one can learn transformer-based BC policies by optimizing the sequence modeling objective where an autoregressive transformer encodes the history $(\tau_{<t}, s_t)$ and decodes the next action a_t as:

$$\mathcal{L}_{\text{LM}}(\pi) = \mathbb{E}_{\tau \sim \mathcal{D}^*_{\text{off}}}[\sum_{t=0}^{H} - \log \pi(a_t|\tau_{<t}, s_t)]. \tag{2.15}$$

An additional conditioning variable z that captures trajectory-level information such as the goal or return $z(\tau) = R(\tau)$ has been introduced in goal or return conditioned supervised learning [100, 101, 102, 103, 104]:

$$\mathcal{L}_{\text{LM}}(\pi) = \mathbb{E}_{\tau \sim \mathcal{D}_{\text{off}}} \left[\sum_{t=0}^{H} - \log \pi(a_t|\tau_{<t}, s_t, z(\tau)) \right]. \tag{2.16}$$

When behavior generation is conditioned on high returns, intuitively, desirable behavior is encouraged [60].

One can also utilize a diffusion model to model the conditional distribution of behaviors [105] by maximizing the likelihood in Equation 2.12:

$$\mathcal{L}_{\text{Diffusion}}(\pi) = \mathbb{E}_{\tau \sim \mathcal{D}_{\text{off}}, k \sim K} \left[\sum_{t=0}^{H} - \log \pi(a_t^{k-1}|a_t^k, s_t, z(\tau)) \right]. \tag{2.17}$$

To extract desirable behavior from a diffusion model when conditioned on high reward, one can sample trajectories with high likelihood by using reward as classifier-free guidance [106].

Other conditional generative models that use normalizing flows [107], generative adversarial networks [108], and energy-based models [17] have also been proposed for modeling behavioral priors from $\mathcal{D}_{\text{off}}$.

2.3.1.2 Generalist Agents Trained on Massive Behavior Datasets

A key advantage to generative modeling of behaviors lies in scaling up; despite different tasks possessing different observations and rewards, there are often meaningful behaviors shared across tasks (e.g., "moving left" has similar meaning in navigation, game playing, and robot manipulation tasks). Inspired by the scaling success of transformers, generalist agents modeling sequences of diverse behaviors have been developed for simulated tasks [99], over 40 Atari games [3], over 700 real-world robot tasks [19], and over 600 distinct tasks with varying modalities, observations and action specifications [49]. This has led to generalist agents that are able to play video games, caption images, chat, perform robot tasks, significantly better than specialist agents trained on single tasks. Such works have also demonstrated the benefit of scaling model parameters and the number of training tasks.

While combining multiple task-specific datasets $\mathcal{D}_{\text{off}}$ into a large multi-task dataset as described above is one way to scale up behavior modeling, exploiting Internet-scale collections

of text and video data $\mathcal{D}$ is another viable approach to scaling effectively. Internet-scale text and video data is abundant in quantity but typically has limited action annotations compared to $\mathcal{D}_{\text{off}}$. Nevertheless, previous work has still incorporated such datasets. For instance, Gato [49] approaches this issue with universal tokenization, so that data with and without actions can be jointly trained using large sequence models. UniPi [6] directly learns to predict robotic videos and trains a separate inverse model to infer actions from generated videos. Applying inverse dynamics models to label large video data (e.g., from YouTube) is also applicable to other domains such as self-driving cars [109] and video game playing [110, 111].

2.3.1.3 Large Scale Online Learning

An alternative approach to assuming access to large-scale behavior datasets, online access to massive online game simulators has enabled "large-scale" online RL models to be trained in games such as DoTA [112] and StarCraft [113] using policy gradient or actor-critic algorithms. Similarly, domain randomization [114] has been proposed to leverage online access to diverse generated environments to help bridge the sim-to-real gap in robotics. These large scale online training schemes, however, have not been able to leverage foundation models. An important direction for future work is to explore how one can utilize and learn generative models similarly in massive online settings.

2.3.1.4 Generative Models of Exploration and Self-Improvement

Generative models of behavior can also be extended to model meta-level processes, such as exploration and self-improvement, whenever the dataset itself $\mathcal{D}_{\text{off}}$ embodies exploratory and self-improving behavior (e.g., the replay buffer of a policy gradient agent trained from scratch) [4]. That is, unlike other meta-RL methods, which usually train in online settings by maximizing multi-episodic value functions [115, 116], algorithm distillation imitates the action sequence of a multi-episodic improvement process from $\mathcal{D}_{\text{off}}$ by using a transformer-based sequence model inspired by the zero-shot ability of language models, and adapts to downstream tasks purely in-context without updating any network parameters.

Similar to algorithm distillation, which prompts an agent with its prior learning experience, corrective re-prompting also treats long-horizon planning as an in-context learning problem, but uses corrective error information as prompts, essentially incorporating feedback from the environment as an auxiliary input to improve the executability of a derived plan [117].

2.3.2 Generative Models of the World

In addition to learning models of behaviors, generative models can also learn models of the world—i.e., the transition dynamics $\mathcal{T}$ and the reward function $\mathcal{R}$—from the offline dataset

$\mathcal{D}_{\text{off}}$. Conditional generation from a world model is analogous to model-based rollouts, which can be used to improve a policy.

2.3.2.1 One-Step Prediction of Reward and Dynamics for Model-based Planning

One can view learning models of $\mathcal{T}$ and $\mathcal{R}$ as a generative modeling problem given trajectories from an offline dataset $\tau \sim \mathcal{D}_{\text{off}}$. Since $\mathcal{D}_{\text{off}}$ also contains actions from a behavior policy π, then π, $\mathcal{T}$, and $\mathcal{R}$ can be jointly modeled with a single generative procedure. Specifically, the joint distribution of a trajectory $p(\tau)$ can be factored autoregressively into an environment component and a policy component,

$$p(\tau) = \Pi_{t=0}^{H} p(s_t, r_t, a_t | \tau_{<t}) = \Pi_{t=0}^{H} \mathcal{T}(s_t|\tau_{<t}) \cdot \pi(a_t|\tau_{<t}, s_t) \cdot \mathcal{R}(r_t|\tau_{<t}, s_t, a_t), \tag{2.18}$$

so that maximum likelihood estimation of $p(\tau)$ using $\mathcal{D}_{\text{off}}$ under this factorization naturally decomposes into learning the environment dynamics $\mathcal{T}, \mathcal{R}$ and the policy π that produced the dataset $\mathcal{D}_{\text{off}}$.

Unlike language models where words exist in a common discrete space, here the states, actions and rewards in τ can all be expressed in different modalities, which poses challenges to sequentially modeling τ. As a workaround, the Trajectory Transformer [5] discretizes each dimension of states, actions, and rewards in a continuous control task before applying a GPT-style autoregressive model on the discretized tokens. Discretization is more challenging in image-based domains, where learning a latent representation of an image space and latent dynamics model is more common. Here one can introduce a per-step latent variable z_t into the sequence modeling objective in Equation 2.18:

$$p(\tau) = \Pi_{t=0}^{H} \int_{z_t} \mathcal{T}_{\text{enc}}(z_t|\tau_{<t}) \cdot \mathcal{T}_{\text{dec}}(s_t|\tau_{<t}, z_t) \cdot \pi(a_t|\tau_{<t}, z_t) \cdot \mathcal{R}(r_t|\tau_{<t}, z_t, a_t) dz_t, \tag{2.19}$$

where $\mathcal{T}_{\text{enc}}(z_t|\tau_{<t})$ encodes the history into the next step's latent state, $\mathcal{T}_{\text{dec}}(s_t|\tau_{<t}, z_t)$ decodes the next step's observation, and the policy π and reward $\mathcal{R}$ can take latent state z_t as input. Along this line, both [118] and [119] apply a sequential VAE [120] to optimize the ELBO of Equation 2.19, and parametrize the latent dynamics model using an RNN or transformer based state space model respectively. Similarly, [121, 122, 123, 124] usesd VQ-VAE or masked autoencoders (MAE) to map image-based observations into discrete tokens before learning a transformer or latent state space dynamics model on the discretized observations.

The various ways a learned world model can be used to infer a high quality policy have been method and task specific. For example, heuristic decoding such as return guided beam search and MCTS have been applied to policy optimization [5, 125, 122]. Separate actor and critic pairs have also been trained using rollouts from a latent world model (also referred to as "imagination") without requiring generating image-based observations [126, 127]. A world model, when trained to predict observations and actions in the original input space, can also be used to generate additional training data for model-free RL [128, 129, 91, 130] under the Dyna framework [36] or to generate additional input context to a policy [131].

2.3.2.2 Planning with Generative Models of Long-term Future

Instead of autoregressively factoring τ by time step as in Equation 2.18, one can also directly model the joint distribution of τ across all time steps at once using a diffusion model [132, 133]:

$$p(\tau) = p(s_0, a_0, r_0, \ldots, s_H, a_H, r_H) = \int p(\tau_K) \Pi_{k=1}^{K} p(\tau_{k-1}|\tau_k) d\tau_{1:K}. \tag{2.20}$$

By learning a trajectory level generative model, planning can be more easily integrated with dynamics modelling by sampling from the composed distribution

$$\tilde{p}(\tau) \propto p(\tau) z(\tau), \tag{2.21}$$

where $z(\tau)$ specifies the trajectory-level properties that one wishes to control. For instance, [133] uses trajectory returns as $z(\tau)$ to guide a reverse diffusion process towards sampling high-return trajectories. [105] further demonstrate that $z(\tau)$ can represent different trajectory-level properties such as goals, skills, and dynamics constraints, where classifier-free guidance can be applied to conditionally sample trajectories that satisfy the desired properties. Going beyond low dimensional state action spaces, [6] also show that diffusion models of long-term futures can also be applied to high-dimensional video data τ, using $z(\tau)$ as text descriptions, effectively improving decision making with large-pretrained text-video foundation models.

In addition to the benefit of flexible conditioning (e.g., on returns, goals, constraints, skills, texts), sampling from the composed distribution in Equation 2.21 holds the promise of accurate long horizon planning, since sampling an entire trajectory does not suffer from compounding error when rolling out single-step dynamics. Beyond diffusion models, EBMs can also be used to model the joint trajectory distributions $p(\tau)$, including conditioning on latent trajectory properties $z(\tau)$, which might provide a natural approach to satisfying multiple desirable properties, such as high return and safety [134, 135].

2.4 Foundation Models as Representation Learners

In this section, we discuss foundation models for decision making that leverage representation learning for knowledge compression. On one hand, foundation models can extract representations from broad image and text data, $\mathcal{D}$, resulting in a plug-and-play style of knowledge transfer to vision and language based decision making tasks. On the other hand, foundation models can also be used to support task-specific representation learning via task-specific objectives and interactive data, $\mathcal{D}_{\text{off}}$.

2.4.1 Plug-and-Play

Off-the-shelf foundation models pretrained on Internet-scale text and image data can be used as preprocessors or initializers for various perceptual components of decision making agents.

For instance, when an agent's perception is based on images, contrastive learning [136] and masked autoencoding [137] can be directly applied to the agent's image observations, providing state representations that can be further finetuned by BC or RL objectives [138, 139, 140, 141]. When agent actions can be characterized by natural language (e.g., "move to the left then pick up the cup"), pretrained language models can be used to generate higher-level plans for longer-horizon tasks, with the hope that language based descriptions of actions generalize better than low-level motor controls [142, 143, 144, 145]. When agent observations consist of both images and text descriptions, vision-language captioning models can further enrich agent observations with language descriptions [146, 147, 145]. Vision-language models such as CLIP and PaLI [148] are further able to provide task feedback and reward information by aligning image and language modalities in the agent's observation and goal space [142, 149, 48]. Even in the case where an agent's states, actions, and rewards do not consist of images or text, pretrained language models, perhaps surprisingly, have still been found useful as policy initializers for offline RL [150], online RL [151], and structured prediction tasks [152].

Plug-and-play foundation models are generally more natural when the decision making task concerns real-world images or texts. Plug-and-play is less applicable to decision making tasks when there are idiosyncratic, domain specific state action spaces, which we will discuss in Section 2.4.3.

2.4.2 Vision and Language as Task Specifiers

An important special case of plug-and-play foundation models is to use text commands or visual inputs as task specifiers to learn more robust, general, and multi-task policies [143, 142, 19, 153]. For instance, a text description of "close the cabinet door" or a goal image with the cabinet door closed can serve as policy input to augment the current robot state. There are a few motivations behind this approach. First, using language and a goal image to specify a task provides richer information about the intended task rather than merely providing a scalar reward. Second, pretrained language models (equipped with prompting methods such as chain-of-thought) can decompose high-level tasks into lower-level instructions that are easier to execute [143, 142, 154, 155]. Furthermore, pretrained vision-language models can enable language-conditioned agents to generalize to new instructions, scenes, and objects in navigation and manipulation tasks [156, 157, 158, 159, 160, 161, 143, 142, 50, 162, 163, 164], which has been a key challenge in robotics prior to their introduction [165].

Using vision and language task specifiers to prompt for desirable agent behaviors requires additional data such as text descriptions or goal images of a given tasks. Moreover, prompting for desirable outcomes from a large language model has significant potential but is also an open problem in itself [166], whose complexity is exacerbated in decision making scenarios with external entities and world dynamics.

2.4.3 Learning Representations for Sequential Decision Making

Unlike vision-language foundation models that can learn from a broad data collection $\mathcal{D}$ but lack the notion of decision making, foundation model techniques and architectures (as opposed to the pretrained models themeselves) can be used to optimize objectives uniquely devised for sequential decision making on the basis of task-specific interactive data $\mathcal{D}_{\text{off}}$. Figure 2.2 visually illustrates these representation learning objectives.

Model-based representations. Traditionally, representation learning for sequential decision making has been framed as learning a latent state or action space of an environment by "clustering" states and actions that yield similar transition dynamics [167, 168, 169, 170, 171, 172]. Similar to how foundation models can serve as generative models of world dynamics by maximizing $p(\tau)$ in Equation 2.18, foundation models can also serve as representation learners of world dynamics under the following objective:

$$p(\tau_{s,r}) = \Pi_{t=0}^{H} p(s_{t+1}, r_t | \tau_{<t}, s_t, a_t) = \Pi_{t=0}^{H} \mathcal{T}(s_{t+1} | \tau_{<t}, \phi(s_t), a_t) \cdot \mathcal{R}(r_t | \tau_{<t}, \phi(s_t), a_t). \quad (2.22)$$

Using this factorization for maximum likelihood estimation of $p(\tau_{s,r})$ using $\mathcal{D}_{\text{off}}$ naturally leads to learning state representations $\phi(s)$ that "cluster" states with similar rewards and next state probabilities. One could also choose to maximize the likelihood of the next state *representations* as opposed to the next raw state, i.e., $\mathcal{T}(\phi(s_{t+1}) | \tau_{<t}, \phi(s_t), a_t)$ resulting in a latent dynamics model [171]. Alternative learning objectives for $\phi(s)$ can be derived depending on how $\mathcal{T}(s_{t+1} | \tau_{<t}, \phi(s_t), a_t)$ is defined. For instance, $\mathcal{T}$ may be defined as an energy-based model:

$$\mathcal{T}(s_{t+1} | \tau_{<t}, \phi(s_t), a_t) \propto \exp\{\phi(s_{t+1})^\top f(\phi(s_t), a_t, \tau_{<t})\}, \quad (2.23)$$

where f is a trainable function that maps $\phi(s_t), a_t, \tau_{<t}$ to the same embedding space as ϕ . While Equation 2.22 learns state representations by modeling the forward dynamics, one can also learn state representations based on an *inverse* dynamics model [173, 174] by predicting a_t from $\tau_{<t}, s_t, s_{t+1}$, thereby maximizing

$$p(\tau_a) = \Pi_{t=0}^{H} p(a_t | \tau_{<t}, \phi(s_t), \phi(s_{t+1})). \quad (2.24)$$

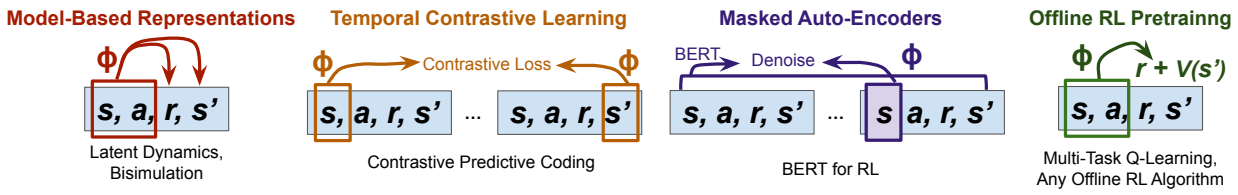


Figure 2.2: Illustrations of different representation learning objectives such as model-based representations [7], temporal contrastive learning [8], masked autoencoders [1], and offline RL [9], on a trajectory $\tau \sim \mathcal{D}_{\text{off}}$ specifically devised for sequential decision making.

In addition to forward and inverse dynamics based representations, it is also possible to learn state representations derived from predicted value functions [175], curiosity metrics [176], or other MDP-based similarity metrics such as bisimulation properties deduced from Bellman backups [177, 178, 179]. The above representation learning objectives have mostly been considered under the Markovian setting, hence the dependence on $\tau_{<t}$ is often dropped. Though the Markovian assumption makes large sequence models seem less relevant, these representation learning objectives benefit from sequence modeling architectures in image-based domains that are generally non-Markovian.

Temporal contrastive learning. The model-based representation objectives above require strictly interleaved state-action-reward tuples in the training data $\mathcal{D}_{\text{off}}$, which can preclude more flexible representation learning techniques that consider broader data sources, $\mathcal{D}$, such as YouTube videos (which can be thought of as state-only trajectories τ_s). Temporal contrastive learning such as CPC [8], on the other hand, can model more flexible sequence-level representations, and has been applied to playing games by watching YouTube videos [180]. Specifically, in temporal contrastive learning, observations that are closer temporally (e.g., observations that belong to the same trajectory) are encouraged to have similar representations. Given a sub-trajectory $\tau_{t:t+h}$, one can learn $\phi(s)$ by minimizing a contrastive loss between $\phi(s_t)$ and $\phi(s_{t+i})$:

$$-\phi(s_{t+i})^\top W_i \phi(s_t) + \log \mathbb{E}_\rho[\exp\{\phi(\tilde{s})^\top W_i \phi(s_t)\}]. \tag{2.25}$$

where $i = 1, \ldots, h$, W_i is a learnable weight matrix, and ρ is some non-trainable prior distribution. Note that the temporal contrastive learning in Equation 2.25 bears resemblance to learning an energy-based dynamics model in Equation 2.23, as established in prior work [7, 181].

Masked autoencoders. When a trajectory $\tau = (s_0, a_0, r_0, ..., s_H, a_H, r_H)$ from $\mathcal{D}_{\text{off}}$ is treated as a flattened sequence, BERT-style denoising autoencoding objectives can be applied to the sequence to learn representations of states, actions, rewards, and dynamics through specific choices of masking patterns [182, 183, 184, 124]. These methods learn representations $\phi(s)$ by first randomly masking a subset of tokens in τ to obtain $\hat{\tau}$, then pass the masked sequence $\hat{\tau}$ to a transformer, and finally reconstruct the masked portions of the original input $\bar{\tau}$ from the transformer output $F(\hat{\tau})$. The training objective, for instance, can be characterized as maximizing

$$p(\bar{\tau}|\hat{\tau}) = \Pi_{t=0}^{H} m_t p(\tau_t|\hat{\tau}) = \Pi_{t=0}^{H} m_t \frac{\exp\{F(\hat{\tau})_t^T \phi(s_t)\}}{\sum_s \exp\{F(\hat{\tau})_t^T \phi(s)\}}, \tag{2.26}$$

where for each masked input state s_t, a contrastive loss between its representation $\phi(s_t)$ and the transformer output at its sequential position $F(\hat{\tau})_t$ is applied. Unlike model-based representation learning approaches that explicitly model state transition probabilities, masked

autoencoders can learn representations from a broader dataset that potentially has missing actions and rewards, while still being able to incorporate dynamics-based information in the learned representations.

Offline RL pretraining. When the downstream decision making tasks are to be trained with RL objectives, it might seem natural to apply similar RL objectives during pretraining when acquiring value-based representations [185, 81]. At a high level, value-based pretraining encompasses any offline RL algorithms that have been pretrained on logged experience from one or more tasks relevant to the downstream interactive task of interest. Value-based pretraining has exhibited scaling capability in multi-task settings where state action spaces are similar (e.g., all of Atari games [9]).

2.4.3.1 Post Representation Learning: BC and RL Finetuning

Unlike generative foundation models that can directly produce action or next state samples, as in Section 2.3, foundation models as representation learners are only directed to extract representations of states, actions, and dynamics; hence they require additional finetuning or model-based policy optimization to achieve strong decision making performance. On the theoretical side, various works have focused on developing representation learning objectives that ensure downstream BC or policy/value-based RL finetuning using the pretrained representations are provably efficient [186, 7, 187, 188, 189]. These analyses are generally based on properties of linear MDPs. For instance, one such assumption states that the state-action value function $Q^\pi(s, a)$ can be represented as a linear combination of features $\phi(s, a)$ under the linear MDP factorization $\mathcal{T}(s'|s, a) = \langle \phi(s, a), \theta(s') \rangle$ and $\mathcal{R}(s, a) = \langle \phi(s, a), \theta_r \rangle$, which ensures that standard policy and value based RL training can take place in the more compact representation space $\phi(s, a)$ as opposed to the original state-action space. Beyond providing compact state action spaces for policy and value-based model-free RL methods, pretrained representations can also simplify model learning and policy rollouts of model-based policy optimization [69, 175, 127] as described in Section 2.3.2.

While representation learning objectives specifically devised for sequential decision making have theoretical benefits, it is less clear how these objectives can effectively incorporate broader and multi-task data when the underlying dynamics differ from that of the target task of interest.

2.5 Foundation Models as Agents and Environments

We have seen that foundation models can characterize different components of a decision making process ($\mathcal{M}$), such as agent behaviors (A), world dynamics ($\mathcal{T}$), task specifiers ($\mathcal{R}$), and state (S) and action representations. In this section, we further consider a special case where pretrained large language models can serve as agents or environments. Treating language models as agents, on one hand, enables learning from environment feedback produced

by humans, tools, or the real world, and on the other hand enables new applications such as information retrieval and web navigation to be considered under a sequential decision making framework. Language models can also be thought of as computational environments that take text as input and produce text as output, effectively supporting interactions with external prompts.

2.5.1 Interacting with Humans

Dialogue as an MDP. A piece of dialogue can be viewed as in alternating nteraction between a dialogue agent π and a human environment $\mathcal{M} = \mathcal{E}$, where a conversation $\tau_{<t} = \{e_0, a_1, e_1, ..., a_t\}$ consists of sentences a_i and e_i produced by π and $\mathcal{E}$ respectively. On the t-th turn, a state $s_t \in S$ captures the conversation history $s_t = \{\tau_{<t}, e_t\}$, an action $a_t \in A$ is an agent's response given this context, a next state $s_{t+1} \in S$ concatenates s_t with a_t and e_{t+1}, and a reward $r_t = \mathcal{R}(s_t, a_t)$ is produced. An agent π aims to maximize $\mathbb{E}_{e_0 \sim \mu, \pi, \mathcal{T}}[\sum_{t=0}^{H} \gamma^t \mathcal{R}(s_t, a_t)]$.

Optimizing dialogue agents. The application of large language models to dialogue generation is a natural one, as both the broad pretraining data $\mathcal{D}$ and the task-specific dialogue data $\mathcal{D}_{\text{off}}$ are of the same text modality, which allows for task-specific finetuning using the same self-supervised loss as pretraining [190, 191, 192, 53]. Such an approach has achieved impressive performance as assessed by humans, under metrics including safety, sensibleness, interestingness, truthfulness, and helpfulness [53, 193]. Although human feedback was initially used to evaluate dialogue systems [194], it was soon incorporated as a reward signal for optimizing dialogue agents under the *reinforcement learning with human feedback* (RLHF) framework [22, 45, 193, *inter alia*]. In practice, RLHF involves several stages: first, a pretrained language model is finetuned on dialogue data to provide an initial policy π; second, output from this model is ranked by human raters, which is then used to train a preference (reward) model $\mathcal{R}$; finally, the language model is finetuned using policy gradient in Equation 2.4 to maximize the reward given by the preference model. Other RL objectives such as Q-learning (Equation 2.5) and actor-critic (Equation 2.6) have also been used to enable dialogue agent to perform specific tasks, such as booking flights and selling items on Craigslist [195, 196, 197, 198, 199].

Limitations of dialogue agents. While using human feedback is a natural way to turn broad data $\mathcal{D}$ into task-specific data $\mathcal{D}_{\text{off}}$, solely relying on human feedback to finetune a language model agent has a number of limitations. For instance, language models have been criticized for failing to access up-to-date information [52], hallucinating facts [200, 201], and struggling to perform complex reasoning and mathematical calculations [202]. Such failure modes are unsuprising if these desired properties were never a part of the feedback the language model received. While one approach to mitigate such failure modes is to collect

human feedback on each of the desired properties, leveraging tools and external entities that can automatically provide feedback is likely to be a more scalable and reliable approach.

2.5.2 Interacting with Tools

Language model agents that generate API calls (to invoke external tools and receive responses as feedback to support subsequent interaction) can be formulated as a sequential decision making problem analogous to the dialogue formulation in the previous section. Several tools such as search engines [52, 53, 54, 55, 56], calculators [57, 53], translators [53], MuJoCo simulators [58], scratch pads [203], computer memory [204], and program interpreters [59] have been used to augment language models in a supervised finetuning or prompting setting, where response from tools are used as additional inputs to the language model.

Limitations of tool use agents. Unlike dialogue systems, where the agent and environment take turns, tool-using agents need to additionally decide when to call external tools, which tools to use, and how to use these tools (e.g., reformulating query if results are not helpful), all of which pose additional challenges. Consequently, the supervised finetuning of tool-use agents requires significant human supervision through API call annotations. While prompting-based tool-use requires fewer examples, the specific prompts typically need to be hand-crafted for each tool [205]. Moreover, language models are known to be sensitive to the prompt formats in both the zero and few-shot settings [206, 207]. As a result, the communication between language models and external tools typically needs to be cleaned-up by a rule-based parser, which further complicates the prompting setup. Recently, [208] and [205] have made progress on self-supervised learning of tool use with language models, training the language model to only an external tool if this leads to an improved response over the outcome predicted by language model alone. Nevertheless, none of the existing work considers tool use in an interactive setting where an agent can *iterate* on its behavior according to tool feedback to improve its tool-use ability.

Tools as interactive environments. It is challenging to scale supervised finetuning and prompting to a large number of tools with different uses and tools that return large amounts of feedback (e.g., hundreds of search results). One sensible way of tackling this challenge is to treat tools like web browsers as interactive environments, from which experience can be sampled by executing search queries [192, 209], and optimizing such queries via RL techniques such as policy gradient. Treating tools as interactive environments enables methods that require massive and efficient online simulator access (e.g., Monte Carlo Tree Search for AlphaGo) to be applied to a broader set of real-world problems, such as web navigation and information retrieval. Additionally, situating language models in true knowledge obtained from the environment better grounds the model, avoiding the the Dichotomy of Control problem (e.g., sequence models generating next states without respecting environment transitions) [104].

2.5.3 Language Models as Environments

Prompting as an MDP. Iterative prompting can be characterized as an MDP that captures the interaction between a prompt provider π and a language model environment $\mathcal{E}$, where a prompt history $\tau_{<t} = \{e_0, a_1, e_1, ..., a_t\}$ consists of prompts a_i and language model outputs e_i produced by π and $\mathcal{E}$ respectively. Here, e_0 is the initial context to the language model. In the t-th turn, a state $s_t \in S$ captures the prompting history and the t-th language model responses $s_t = \{\tau_{<t}, e_t\}$, an action $a_t \in A$ is given by the prompt provider, a next state $s_{t+1} \in S$ is produced by concatenating s_t with a_t and the next response of the language model e_{t+1}, and a reward $r_t = \mathcal{R}(s_t, a_t)$ is emitted. An agent π aims to maximize $\mathbb{E}_{e_0 \sim \mu, \pi, \mathcal{T}}[\sum_{t=0}^{H} \gamma^t \mathcal{R}(s_t, a_t)]$. In language model reasoning, for instance, $\mathcal{R}(s_t, a_t) = 1$ if the language model's output successfully reaches a goal answer s_t (i.e., correct reasoning), and $\mathcal{R}(s_t, a_t) = 0$ otherwise.

Under this formulation, various schemes for language model prompting can be characterized by high-level actions that map input strings to desired output strings using the language model. For instance, such high-level actions include `DECOMPOSE` [210], `RANK` [211], `DENOISE` [212], and `PARAPHRASE` [213]. These high-level actions can also be recursively composed to achieve more sophisticated iterative prompting schemes [214]. Other prompting schemes such as `SUMMARIZE`, `PRUNE`, `SEARCH` can be considered for handling challenges such as overcoming long context lengths. Given that language models with auxiliary memory have been shown to emulate universal Turing machines [204], language models could ultimately serve as "computers" that also operate on human language with prompting as a flexible new form of programming language.

Chapter 3

A Framework for Offline Policy Evaluation

Traditionally, sequential decision making, especially reinforcement learning (RL), considers the tabular rasa setting where an agent interacts with an environment through trial and error, while improving its policy (strategy for choosing actions) without any prior knowledge. As a result, sample efficiency is at odds as the agent must sequentially interacts with the environment numerous times to learn a good policy. This traditional problem setting of learning from scratch can be overly restrictive, as there is hardly ever a task for which we have absolutely zero data or prior knowledge. A more practical question naturally arises: given previously failed or imperfect interactions, how might one use the suboptimal experience to improve a policy. Furthermore, online environment access can be expensive and safety critical, whereas logged experiences of previous interactions are often accessible. It is natural to wonder if a policy can achieve good performance when only trained on previously logged experience data without further access to the environment. This is the problem setting that became known as *offline RL*, which was first introduced as an extension to off-policy RL [77, 76]. In off-policy RL, while there is a experience buffer that stores previous experiences, online interactions are still provided, whereas in offline RL, no environment access is possible.

While the problem setting of offline RL also seems restrictive (assumptions on zero environment access, step-by-step sequential interactions, etc), it opens the possibility of incorporating diverse datasets of previous interactions. Imagine that we have the data from previous plays of a video game, can we use these data to build *the best* video game agent? Similarly, given all of the driving data from human drivers, can we build *the safest* self-driving car?

As introduced in Section 2.1.3, policy improvement in RL can be broken down into policy evaluation (computing the future expected return of the current policy) and policy improvement (improving the actions to maximize future expected return). In this chapter, we will focus on policy evaluation. Specifically, we introduce a novel framework for policy evaluation, known as Distribution Correction Estimation (DICE). The major contribution of this framework is that it enables the trade off between optimization stability and unbiased estimation, so that estimators can be made more stable in function approximation settings.

3.1 A Framework for Off-Policy Evaluation

3.1.1 Introduction

One of the most fundamental problems in reinforcement learning (RL) is *policy evaluation*, where we seek to estimate the expected long-term payoff of a given *target* policy in a decision making environment. An important variant of this problem, *off-policy evaluation* (OPE) [215], is motivated by applications where deploying a policy in a live environment entails significant cost or risk [216, 217]. To circumvent these issues, OPE attempts to estimate the value of a target policy by referring only to a dataset of experience previously gathered by other policies in the environment. Often, such logging or *behavior* policies are not known explicitly (e.g., the experience may come from human actors), which necessitates the use of *behavior-agnostic* OPE methods [218].

While behavior-agnostic OPE appears to be a daunting problem, a number of estimators have recently been developed for this scenario [218, 219, 220, 221], demonstrating impressive empirical results. Such estimators, known collectively as the "DICE" family for *DIstribution Correction Estimation*, model the ratio between the propensity of the target policy to visit particular state-action pairs relative to their likelihood of appearing in the logged data. A distribution corrector of this form can then be directly used to estimate the value of the target policy.

Although there are many commonalities between the various DICE estimators, their derivations are distinct and seemingly incompatible. For example, *DualDICE* [218] is derived by a particular change-of-variables technique, whereas *GenDICE* [220] observes that the substitution strategy cannot work in the average reward setting, and proposes a distinct derivation based on distribution matching. *GradientDICE* [221] notes that GenDICE exacerbates optimization difficulties, and proposes a variant designed for limited sampling capabilities. Despite these apparent differences in these methods, the algorithms all involve a minimax optimization that has a strikingly similar form, which suggests that there is a common connection that underlies the alternative derivations.

We show that the previous DICE formulations are all in fact equivalent to regularized Lagrangians of the same linear program (LP). This LP shares an intimate relationship with the policy evaluation problem, and has a primal form we refer to as the Q-LP and a dual form we refer to as the d-LP. The primal form has been concurrently identified and studied in the context of policy optimization [222], but we focus on the d-LP formulation for off-policy evaluation here, which we find to have a more succinct and revealing form for this purpose. Using the d-LP, we identify a number of key choices in translating it into a *stable* minimax optimization problem – i.e., whether to include redundant constraints, whether to regularize the primal or dual variables – in addition to choices in how to translate an optimized solution into an *asymptotic unbiased*, "unbiased" for short, estimate of the policy value. We use this characterization to show that the known members of the DICE family are a small subset of specific choices made within a much larger, unexplored set of potential OPE methods.

To understand the consequences of the various choices, we provide a comprehensive study.

First, we theoretically investigate which configurations lead to bias in the primal or dual solutions, and when this affects the final estimates. Our analysis shows that the dual solutions offer greater flexibility in stabilizing the optimization while preserving asymptotic unbiasedness, versus primal solutions. We also perform an extensive empirical evaluation of the various choices across different domains and function approximators, and identify novel configurations that improve the observed outcomes.

3.1.2 Background

3.1.2.1 Dual Policy Evaluation

The *value* of a policy π is defined as the normalized expected per-step reward it obtains:

$$\rho(\pi) := (1-\gamma)\mathbb{E}\left[\sum_{t=0}^{\infty}\gamma^t R(s_t,a_t) \mid s_0 \sim \mu, \forall t, a_t \sim \pi(s_t), s_{t+1} \sim T(s_t,a_t)\right]. \tag{3.1}$$

In the policy evaluation setting, the policy being evaluated is referred to as the *target* policy. The value of a policy may be expressed in two equivalent ways:

$$\rho(\pi) = (1-\gamma)\cdot\mathbb{E}_{\substack{a_0\sim\pi(s_0)\\ s_0\sim\mu}}[Q^\pi(s_0,a_0)] = \mathbb{E}_{(s,a)\sim d^\pi}[R(s,a)], \tag{3.2}$$

where Q^π and d^π are the *state-action values* and *visitations* of π, respectively, which satisfy

$$Q^\pi(s,a) = R(s,a) + \mathcal{B}Q^\pi(s,a), \text{ where } \mathcal{P}^\pi Q(s,a) := \mathbb{E}_{s'\sim T(s,a),a'\sim\pi(s')}[Q(s',a')]\ , \tag{3.3}$$

$$d^\pi(s,a) = (1-\gamma)\mu(s)\pi(a|s) + \gamma\cdot\mathcal{P}_*^\pi d^\pi(s,a), \text{ where } \mathcal{P}_*^\pi d(s,a) := \pi(a|s)\textstyle\sum_{\tilde{s},\tilde{a}} T(s|\tilde{s},\tilde{a})d(\tilde{s},\tilde{a}). \tag{3.4}$$

Note that $\mathcal{P}^\pi$ and $\mathcal{P}_*^\pi$ are linear operators that are transposes (adjoints) of each other. We refer to $\mathcal{P}^\pi$ as the *policy transition operator* and $\mathcal{P}_*^\pi$ as the *transpose policy transition operator*. The function Q^π corresponds to the Q-values of the policy π; it maps state-action pairs (s,a) to the expected value of policy π when run in the environment starting at (s,a). The function d^π corresponds to the on-policy distribution of π; it is the normalized distribution over state-action pairs (s,a) measuring the likelihood π enounters the pair (s,a), averaging over time via γ-discounting. We make the following standard assumption, which is common in previous policy evaluation work [220, 222].

Assumption 1 (MDP ergodicity)**.** *There is unique fixed point solution to* (3.4)*.*

When $\gamma \in [0,1)$, (3.4) always has a unique solution, as 0 cannot belong to the spectrum of $I-\gamma\mathcal{P}_*^\pi$. For $\gamma=1$, the assumption reduces to ergodicity for discrete case under a restriction of d to a normalized distribution; the continuous case is treated by [223].

3.1.2.2 Off-policy Evaluation via the DICE Family

Off-policy evaluation (OPE) aims to estimate $\rho(\pi)$ using only a *fixed* dataset of experiences. Specifically, we assume access to a finite dataset $\mathcal{D} = \{(s_0^{(i)}, s^{(i)}, a^{(i)}, r^{(i)}, s'^{(i)})\}_{i=1}^N$, where $s_0^{(i)} \sim \mu$, $(s^{(i)}, a^{(i)}) \sim d^{\text{off}}$ are samples from some unknown distribution d^{off}, $r^{(i)} = R(s^{(i)}, a^{(i)})$, and $s'^{(i)} \sim T(s^{(i)}, a^{(i)})$. We at times abuse notation and use $(s, a, r, s') \sim d^{\text{off}}$ or $(s, a, r) \sim d^{\text{off}}$ as a shorthand for $(s, a) \sim d^{\text{off}}, r = R(s, a), s' \sim T(s, a)$, which simulates sampling from the dataset $\mathcal{D}$ when using a finite number of samples.

The recent DICE methods take advantage of the following expression for the policy value:

$$\rho(\pi) = \mathbb{E}_{(s,a,r)\sim d^{\text{off}}} \left[\zeta^*(s, a) \cdot r\right], \tag{3.5}$$

where $\zeta^*(s, a) := d^\pi(s, a)/d^{\text{off}}(s, a)$ is the *distribution correction ratio*. The existing DICE estimators seek to approximate this ratio without knowledge of d^π or d^{off}, and then apply (3.5) to derive an estimate of $\rho(\pi)$. This general paradigm is supported by the following assumption.

Assumption 2 (Boundedness)**.** *The stationary correction ratio is bounded,* $\|\zeta^*\|_\infty \leq C < \infty$.

When $\gamma < 1$, DualDICE [218] chooses a convex objective whose optimal solution corresponds to this ratio, and employs a change of variables to transform the dependence on d^π to μ. GenDICE [220], on the other hand, minimizes a divergence between successive on-policy state-action distributions, and introduces a normalization constraint to ensure the estimated ratios average to 1 over the off-policy dataset. Both DualDICE and GenDICE apply Fenchel duality to reduce an intractable convex objective to a minimax objective, which enables sampling and optimization in a stochastic or continuous action space. GradientDICE [221] extends GenDICE by using a linear parametrization so that the minimax optimization is convex-concave with convergence guarantees.

3.1.3 A Unified Framework of DICE Estimators

In this section, given a fixed target policy π, we present a linear programming representation (LP) of its state-action stationary distribution $d^\pi(s, a) \in \mathcal{P}$, referred to as the d-LP. The dual of this LP has solution Q^π, thus revealing the duality between the Q-function and the d-distribution of any policy π. We then discuss the mechanisms by which one can improve optimization stability through the application of regularization and redundant constraints. Although in general this may introduce bias into the final value estimate, there are a number of valid configurations for which the resulting estimator for $\rho(\pi)$ remains *unbiased*. We show that existing DICE algorithms cover several choices of these configurations, while there is also a sizable subset which remains unexplored.

3.1.3.1 Linear Programming for the d^π-distribution

The following theorem presents a formulation of $\rho(\pi)$ in terms of a linear program with respect to the constraints in (3.4) and (3.3).

Theorem 3. *Given a policy π, under Assumption 1, its value $\rho(\pi)$ defined in (3.1) can be expressed by the following d-LP:*

$$\max_{d:S\times A\to\mathbb{R}} \mathbb{E}_d\left[R\left(s,a\right)\right], \quad \text{s.t.,} \quad d(s,a) = \underbrace{(1-\gamma)\mu(s)\pi(a|s) + \gamma\cdot\mathcal{P}_*^\pi d(s,a)}_{\mathcal{B}_*^\pi\cdot d}. \tag{3.6}$$

*We refer to the d-LP above as the **dual** problem. Its corresponding **primal** LP is*

$$\min_{Q:S\times A\to\mathbb{R}} (1-\gamma)\,\mathbb{E}_{\mu\pi}\left[Q\left(s,a\right)\right], \quad \text{s.t.,} \quad Q(s,a) = \underbrace{R(s,a) + \mathcal{B}Q(s,a)}_{\mathcal{B}^\pi\cdot Q}. \tag{3.7}$$

Proof. Notice that under Assumption 1, the constraint in (3.6) determines a unique solution, which is the stationary distribution d^π. Therefore, the objective will be $\rho(\pi)$ by definition. On the other hand, due to the contraction of $\gamma\cdot\mathcal{P}^\pi$, the primal problem is feasible and the solution is Q^π, which shows the optimal objective value will also be $\rho(\pi)$, implying strong duality holds. □

Theorem 3 presents a succinct LP representation for policy value and reveals the duality between the Q^π-function and d^π-distribution, thus providing an answer to the question raised by [219]. Although the d-LP provides a mechanism for policy evaluation, directly solving either the primal or dual d-LPs is difficult due to the number of constraints, which will present difficulties when the state and action spaces is uncountable. These issues are exaggerated in the off-policy setting where one only has access to samples (s_0,s,a,r,s') from a stochastic process. To overcome these difficulties, one can instead approach these primal and dual LPs through the Lagrangian,

$$\max_d \min_Q L(d,Q) := (1-\gamma)\cdot\mathbb{E}_{\substack{a_0\sim\pi(s_0)\\ s_0\sim\mu}}[Q(s_0,a_0)] + \sum_{s,a} d(s,a)\cdot(R(s,a)+\gamma\mathcal{P}^\pi Q(s,a)-Q(s,a)). \tag{3.8}$$

In order to enable the use of an arbitrary off-policy distribution d^{off}, we make the change of variables $\zeta(s,a) := d(s,a)/d^{\text{off}}(s,a)$. This yields an equivalent Lagrangian in a more convenient form:

$$\max_\zeta \min_Q L_D(\zeta,Q) := (1-\gamma)\cdot\mathbb{E}_{\substack{a_0\sim\pi(s_0)\\ s_0\sim\mu}}[Q(s_0,a_0)] + \mathbb{E}_{\substack{(s,a,r,s')\sim d^{\text{off}}\\ a'\sim\pi(s')}}[\zeta(s,a)\cdot(r+\gamma Q(s',a')-Q(s,a))]. \tag{3.9}$$

The Lagrangian has primal and dual solutions $Q^* = Q^\pi$ and $\zeta^* = d^\pi/d^{\text{off}}$. Approximate solutions to one or both of $\hat{Q},\hat{\zeta}$ can be used to estimate $\hat{\rho}(\pi)$, by either using the standard

DICE paradigm in (3.5) which corresponds to the dual objective in (3.6) or, alternatively, by using the primal objective in (3.7) or the Lagrangian objective in (3.9); we further discuss these choices later in this section. Although the Lagrangian in (3.9) should in principle be able to derive the solutions Q^π, d^π and so yield accurate estimates of $\rho(\pi)$, in practice there are a number of optimization difficulties that are liable to be encountered. Specifically, even in tabular case, due to lack of curvature, the Lagrangian is not strongly-convex-strongly-concave, and so one cannot guarantee the convergence of the final solution with stochastic gradient descent-ascent (SGDA). These optimization issues can become more severe when moving to the continuous case with neural network parametrization, which is the dominant application case in practice. In order to mitigate these issues, we present a number of ways to introduce more stability into the optimization and discuss how these mechanisms may trade-off with the bias of the final estimate. We will show that the application of certain mechanisms recovers the existing members of the DICE family, while a larger set remains unexplored.

3.1.3.2 Regularizations and Redundant Constraints

The augmented Lagrangian method (ALM) [224] is proposed exactly for circumventing the optimization instability, where strong convexity is introduced by adding extra regularizations *without* changing the optimal solution. However, directly applying ALM, *i.e.*, adding $h_p(Q) := \|\mathcal{B}^\pi \cdot Q - Q\|^2_{d^\mathcal{D}}$ or $h_d(d) := D_f(d||\mathcal{B}^\pi_* \cdot d)$ where D_f denotes the f-divergence, will introduce extra difficulty, both statistically and algorithmically, due to the conditional expectation operator in $\mathcal{B}^\pi$ and $\mathcal{B}^\pi_*$ inside of the non-linear function in $h_p(Q)$ and $h_d(d)$, which is known as "double sample" in the RL literature [225]. Therefore, the vanilla stochastic gradient descent is no longer applicable [226], due to the bias in the gradient estimator.

In this section, we use the spirit of ALM but explore other choices of regularizations to introduce strong convexity to the original Lagrangian (3.9). In addition to regularizations, we also employ the use of redundant constraints, which serve to add more structure to the optimization without affecting the optimal solutions. We will later analyze for which configurations these modifications of the original problem will lead to biased estimates for $\rho(\pi)$.

We first present the unified objective in full form equipped with all choices of regularizations and redundant constraints:

$$\begin{aligned}\max_{\zeta\geq 0}\min_{Q,\lambda} L_D(\zeta, Q, \lambda) :=& (1-\gamma)\cdot \mathbb{E}_{\substack{a_0\sim\pi(s_0)\\ s_0\sim\mu}}[Q(s_0,a_0)] + \lambda \\ &+ \mathbb{E}_{\substack{(s,a,r,s')\sim d^{\text{off}}\\ a'\sim\pi(s')}}[\zeta(s,a)\cdot(\alpha_R\cdot R(s,a) + \gamma Q(s',a') - Q(s,a) - \lambda)] \\ &+ \alpha_Q\cdot\mathbb{E}_{(s,a)\sim d^{\text{off}}}[f_1(Q(s,a))] - \alpha_\zeta\cdot\mathbb{E}_{(s,a)\sim d^{\text{off}}}[f_2(\zeta(s,a))]. \end{aligned} \tag{3.10}$$

Now, let us explain each term in $(\alpha_Q, \alpha_\zeta, \alpha_R, \zeta \geq 0, \lambda)$.

- Primal and Dual regularization: To introduce better curvature into the Lagrangian, we introduce primal and dual regularization $\alpha_Q \mathbb{E}_{d^{\text{off}}}\left[f_1\left(Q\right)\right]$ or $\alpha_\zeta \mathbb{E}_{d^{\text{off}}}\left[f_2\left(\zeta\right)\right]$, respectively. Here f_1, f_2 are some convex and lower-semicontinuous functions.
- Reward: Scaling the reward may be seen as an extension of the dual regularizer, as it is a component in the dual objective in (3.6). We consider $\alpha_R \in \{0, 1\}$.
- Positivity: Recall that the solution to the original Lagrangian is $\zeta^*\left(s,a\right) = \frac{d^\pi(s,a)}{d^{\text{off}}(s,a)} \geq 0$. We thus consider adding a positivity constraint to the dual variable. This may be interpreted as modifying the original d-LP in (3.6) to add a condition $d \geq 0$ to its objective.
- Normalization: Similarly, the normalization constraint also comes from the property of the optimal solution $\zeta^*\left(s,a\right)$, *i.e.*, $\mathbb{E}_{d^{\text{off}}}\left[\zeta\left(s,a\right)\right] = 1$. If we add an extra constraint to the d-LP (3.6) as $\sum_{s,a} d(s,a) = 1$ and apply the Lagrangian, we result in the term $\lambda - \mathbb{E}_{d^{\text{off}}}\left[\lambda\zeta\left(s,a\right)\right]$ seen in (3.10).

As we can see, the latter two options come from the properties of optimal dual solution, and this suggests that their inclusion would not affect the optimal dual solution. On the other hand, the first two options (primal/dual regularization and reward scaling) will in general affect the solutions to the optimization. Whether a bias in the solution affects the final estimate depends on the estimator being used.

Remark (Robust optimization justification): Besides the motivation from ALM for strong convexity, the regularization terms in (3.10), $\alpha_Q \cdot \mathbb{E}_{(s,a)\sim d^{\text{off}}}[f_1(Q(s,a))]$ and $\alpha_\zeta \cdot \mathbb{E}_{(s,a)\sim d^{\text{off}}}[f_2(\zeta(s,a))]$, can also be interpreted as introducing robustness with some perturbations to the Bellman differences. We consider the dual regularization as an example. Particularly, the Fenchel dual form

$$\alpha_\zeta \cdot \mathbb{E}_{(s,a)\sim d^{\text{off}}}[f_2(\zeta(s,a))] = \alpha_\zeta \left\{ \max_{\delta(s,a)\in\Omega} \ \langle \zeta, \delta \rangle - \mathbb{E}_{(s,a)\sim d^{\text{off}}}\left[f_2^*\left(\delta\left(s,a\right)\right)\right] \right\},$$

where Ω denotes the domain of function f_2^*. For simplicity, we consider $f_2^*\left(\cdot\right) = \left(\cdot\right)^2$. Plug this back into (3.10), we obtain

$$\begin{aligned}
&\max_{\zeta\geq 0} \min_{Q,\lambda,\delta\in\Omega} L_D(\zeta, Q, \lambda) := (1-\gamma)\cdot \mathbb{E}_{\substack{a_0\sim\pi(s_0)\\ s_0\sim\mu}}[Q(s_0,a_0)] + \lambda \\
&+ \mathbb{E}_{\substack{(s,a,r,s')\sim d^{\text{off}}\\ a'\sim\pi(s')}}[\zeta(s,a)\cdot(\alpha_R\cdot R(s,a) + \gamma Q(s',a') - Q(s,a) - \lambda - \alpha_\zeta \delta\left(s,a\right))] \\
&+ \alpha_Q \cdot \mathbb{E}_{(s,a)\sim d^{\text{off}}}[f_1(Q(s,a))] + \alpha_\zeta \cdot \mathbb{E}_{(s,a)\sim d^{\text{off}}}[\delta^2\left(s,a\right)],
\end{aligned} \tag{3.11}$$

which can be understood as introducing slack variables or perturbations in L_2-ball to the Bellman difference $\alpha_R \cdot R(s,a) + \gamma Q(s',a') - Q(s,a)$. For different regularization, the perturbations will be in different dual spaces. From this perspective, besides the stability consideration, the dual regularization will also mitigate both statistical error, due to sampling effect in

approximating the Bellman difference, and approximation error induced by parametrization of Q. Similarly, the primal regularization can be interpreted as introducing slack variables to the stationary state-action distribution condition, please refer to Appendix A.1.1.

Given estimates $\hat{Q}, \hat{\lambda}, \hat{\zeta}$, there are three potential ways to estimate $\rho(\pi)$.

- Primal estimator: $\hat{\rho}_Q(\pi) := (1-\gamma) \cdot \mathbb{E}_{\substack{a_0 \sim \pi(s_0) \\ s_0 \sim \mu}}[\hat{Q}(s_0, a_0)] + \hat{\lambda}$.
- Dual estimator: $\hat{\rho}_\zeta(\pi) := \mathbb{E}_{(s,a,r) \sim d^{\text{off}}}[\hat{\zeta}(s,a) \cdot r]$.
- Lagrangian: $\hat{\rho}_{Q,\zeta}(\pi) := \hat{\rho}_Q(\pi) + \hat{\rho}_\zeta(\pi) + \mathbb{E}_{\substack{(s,a,r,s') \sim d^{\text{off}} \\ a' \sim \pi(s')}}\left[\hat{\zeta}(s,a)\left(\gamma \hat{Q}(s', a') - \hat{Q}(s,a) - \hat{\lambda}\right)\right]$.

The following theorem outlines when a choice of regularizations, redundant constraints, and final estimator will provably result in an unbiased estimate of policy value.

Theorem 4 (Regularization profiling)**.** *Under Assumption 1 and 2, we summarize the effects of* (α_Q, α_ζ, α_R, $\zeta \geq 0$, λ)*, which corresponds to primal and dual regularizations, w/w.o. reward, and positivity and normalization constraints. without considering function approximation.*

Regularization (with or without λ)			$\hat{\rho}_Q$	$\hat{\rho}_\zeta$	$\hat{\rho}_{Q,\zeta}$
$\alpha_\zeta = 0$ α_Q ¿ 0	$\alpha_R = 1$	ζ free	**Unbiased**	Biased	**Unbiased**
		$\zeta \geq 0$	Biased		Biased
	$\alpha_R = 0$	ζ free		**Unbiased**	**Unbiased**
		$\zeta \geq 0$			
α_ζ ¿ 0 $\alpha_Q = 0$	$\alpha_R = 1$	ζ free			
		$\zeta \geq 0$			
	$\alpha_R = 0$	ζ free			
		$\zeta \geq 0$			

Notice that the primal and dual solutions can both be unbiased under specific regularization configurations, but the dual solutions are unbiased in 6 out of 8 such configurations, whereas the primal solution is unbiased in only 1 configuration. The primal solution additionally requires the positivity constraint to be excluded (see details in Appendix A.1.2), further restricting its optimization choices.

The Lagrangian estimator is unbiased when at least one of $\hat{Q}, \hat{\lambda}$ or $\hat{\zeta}$ are unbiased. This property is referred to as *doubly robust* in the literature [227] This seems to imply that the Lagrangian estimator is optimal for behavior-agnostic off-policy evaluation. However, this is not the case as we will see in the empirical analysis. Instead, the approximate dual solutions are typically more accurate than approximate primal solutions. Since neither is exact, the Lagrangian suffers from error in both, while the dual estimator $\hat{\rho}_\zeta$ will exhibit more robust performance, as it solely relies on the approximate $\hat{\zeta}$.

3.1.3.3 Recovering Existing OPE Estimators

This organization provides a complete picture of the DICE family of estimators. Existing DICE estimators can simply be recovered by picking one of the valid regularization configurations:

- **DualDICE** [218]: $(\alpha_Q = 0, \alpha_\zeta = 1, \alpha_R = 0)$ without $\zeta \geq 0$ and without λ. DualDICE also derives an *unconstrained primal form* where optimization is exclusively over the primal variables (see Appendix A.1.4). This form results in a *biased* estimate but avoids difficults in minimax optimization, which again is a tradeoff between optimization stability and solution unbiasedness.
- **GenDICE** [220] and **GradientDICE** [221]: $(\alpha_Q = 1, \alpha_\zeta = 0, \alpha_R = 0)$ with λ. GenDICE differs from GradientDICE in that GenDICE enables $\zeta \geq 0$ whereas GradientDICE disables it.
- **DR-MWQL** and **MWL** [219]:$(\alpha_Q = 0, \alpha_\zeta = 0, \alpha_R = 1)$ and $(\alpha_Q = 0, \alpha_\zeta = 0, \alpha_R = 0)$, both without $\zeta \geq 0$ and without λ.
- **LSTDQ** [228]: With linear parametrization for $\tau(s,a) = \alpha^\top \phi(s,a)$ and $Q(s,a) = \beta^\top \phi(s,a)$, for any *unbiased* estimator without $\xi \geq 0$ and λ in Theorem 4, we can recover LSTDQ. Please refer to Appendix A.1.3 for details.
- **Algae Q-LP** [222]: $(\alpha_Q = 0, \alpha_\zeta = 1, \alpha_R = 1, \zeta \geq 0)$ without $\zeta \geq 0$ and without λ.
- **BestDICE:** $(\alpha_Q = 0, \alpha_\zeta = 1, \alpha_R = 0/1)$ with $\zeta \geq 0$ and with λ. More importantly, we discover a variant that achieves the best performance, which was not identified without this unified framework.

3.1.4 Related Work

Off-policy evaluation has long been studied in the RL literature [229, 227, 230, 231, 215, 217]. While some approaches are model-based [232], or work by estimating the value function [233], most rely on importance reweighting to transform the off-policy data distribution to the on-policy target distribution. They often require to know or estimate the behavior policy, and suffer a variance exponential in the horizon, both of which limit their applications. Recently, a series of works were proposed to address these challenges [234, 235, 236]. Among them is the DICE family [218, 220, 221], which performs some form of stationary distribution estimation. The presented work develops a convex duality framework that unifies many of these algorithms, and offers further important insights. Many OPE algorithms may be understood to correspond to the categories considered here. Naturally, the recent stationary distribution correction algorithms [218, 220, 221], are the dual methods. The FQI-style estimator [233] loosely corresponds to our primal estimator. Moreover, Lagrangian-type estimators are also considered [236, 219], although some are not for the behavior-agnostic setting [236].

Convex duality has been widely used in machine learning, and in RL in particular. In one line of literature, it was used to solve the Bellman equation, whose fixed point is the value function [237, 238, 239]. Here, duality facilitates derivation of an objective function that can be conveniently approximated by sample averages, so that solving for the fixed point is converted to that of finding a saddle point. Another line of work, more similar to the presented work, is to optimize the Lagrangian of the linear program that characterizes the value function [240, 241, 242]. In contrast to our work, these algorithms typically do not incorporate off-policy correction, but assume the availability of on-policy samples.

3.1.5 Experimental Evaluation of DICE Family

In this section, we empirically verify the theoretical findings. We evaluate different choices of estimators, regularizers, and constraints, on a set of OPE tasks ranging from tabular (Grid) to discrete-control (Cartpole) and continuous-control (Reacher), under linear and neural network parametrizations, with offline data collected from behavior policies with different noise levels (π_1 and π_2). See Appendix A.1.6 for implementation details and additional results. Our empirical conclusions are as follows:

- The dual estimator $\hat{\rho}_\zeta$ is unbiased under more configurations and yields best performance out of all estimators, and furthermore exhibits strong robustness to scaling and shifting of MDP rewards.
- Dual regularization ($\alpha_\zeta > 0$) yields better estimates than primal regularization; the choice of $\alpha_R \in \{0, 1\}$ exhibits a slight advantage to $\alpha_R = 1$.
- The inclusion of redundant constraints (λ and $\zeta \geq 0$) improves stability and estimation performance.
- As expected, optimization using the unconstrained primal form is more stable but also more biased than optimization using the minimax regularized Lagrangian.

Based on these findings, we propose a particular set of choices that generally performs well, overlooked by previously proposed DICE estimators: the dual estimator $\hat{\rho}_\zeta$ with regularized dual variable ($\alpha_\zeta > 0, \alpha_R = 1$) and redundant constraints ($\lambda, \zeta \geq 0$) optimized with the Lagrangian.

Choice of Estimator ($\hat{\rho}_Q$, $\hat{\rho}_\zeta$, or $\hat{\rho}_{Q,\zeta}$) We first consider the choice of estimator. In each case, we perform Lagrangian optimization with regularization chosen according to Theorem 4 to not bias the resulting estimator. We also use $\alpha_R = 1$ and include redundant constraints for λ and $\zeta \geq 0$ in the dual estimator. Although not shown, we also evaluated combinations of regularizations which can bias the estimator (as well as no regularizations) and found that these generally performed worse; see Section 3.1.5 for a subset of these experiments.

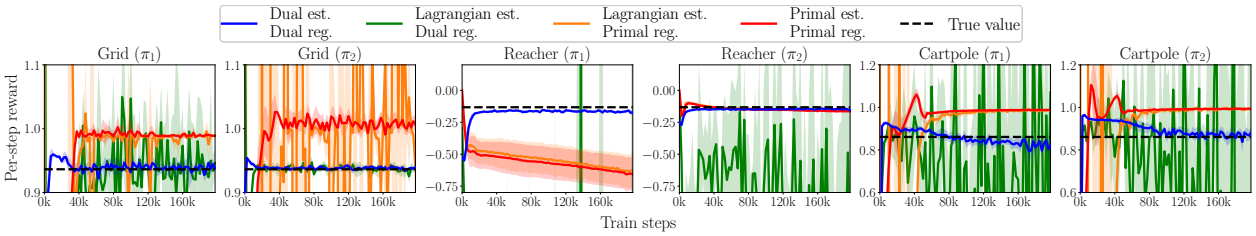


Figure 3.1: Estimation results on Grid, Reacher, and Cartpole using data collected from different behavior policies (π_2 is closer to the target policy than π_1). Biased estimator-regularizer combinations from Theorem 4 are omitted. The dual estimator with regularized dual variable outperforms all other estimators/regularizers. Lagrangian can be as good as the dual but has a larger variance.

Our evaluation of different estimators is presented in Figure 3.1. We find that the dual estimator consistently produces the best estimates across different tasks and behavior policies. In comparison, the primal estimates are significantly worse. While the Lagrangian estimator can improve on the primal, it generally exhibits higher variance than the dual estimator. Presumably, the Lagrangian does not benefit from the doubly robust property, since both solutions are biased in this practical setting.

To more extensively evaluate the dual estimator, we investigate its performance when the reward function is scaled by a constant, shifted by a constant, or exponentiated. [1] To control for difficulties in optimization, we first parametrize the primal and dual variables as linear functions, and use stochastic gradient descent to solve the convex-concave minimax objective in (3.10) with $\alpha_Q = 0$, $\alpha_\zeta = 1$, and $\alpha_R = 1$. Since a linear parametrization changes the ground truth of evaluation, we compute the upper and lower estimation bounds by only parameterizing the primal or the dual variable as a linear function. Figure 3.2 (top) shows the estimated per-step reward of the Grid task. When the original reward is used (col. 1), the primal, dual, and Lagrangian estimates eventually converge to roughly the same value (even though primal estimates converge much slower). When the reward is scaled by 10 or 100 times or shifted by 5 or 10 units (the original reward is between 0 and 1), the resulting primal estimates are severely affected and do not converge given the same number of gradient updates. When performing this same evaluation with neural network parametrization (Figure 3.2, bottom), the primal estimates continue to exhibit sensitivity to reward transformations, whereas the dual estimates stay roughly the same after being transformed back to the original scale. We further implemented target network for training stability of the primal variable, and the same concolusion holds (see Appendix). Note that while the dual solution is robust to the scale and range of rewards, the optimization objective used here still has $\alpha_R = 1$, which is different from $\alpha_R = 0$ where $\hat{\rho}_Q$ is no longer a valid estimator.

[1] Note this is separate from α_R, which only affects optimization. We use $\alpha_R = 1$ exclusively here.

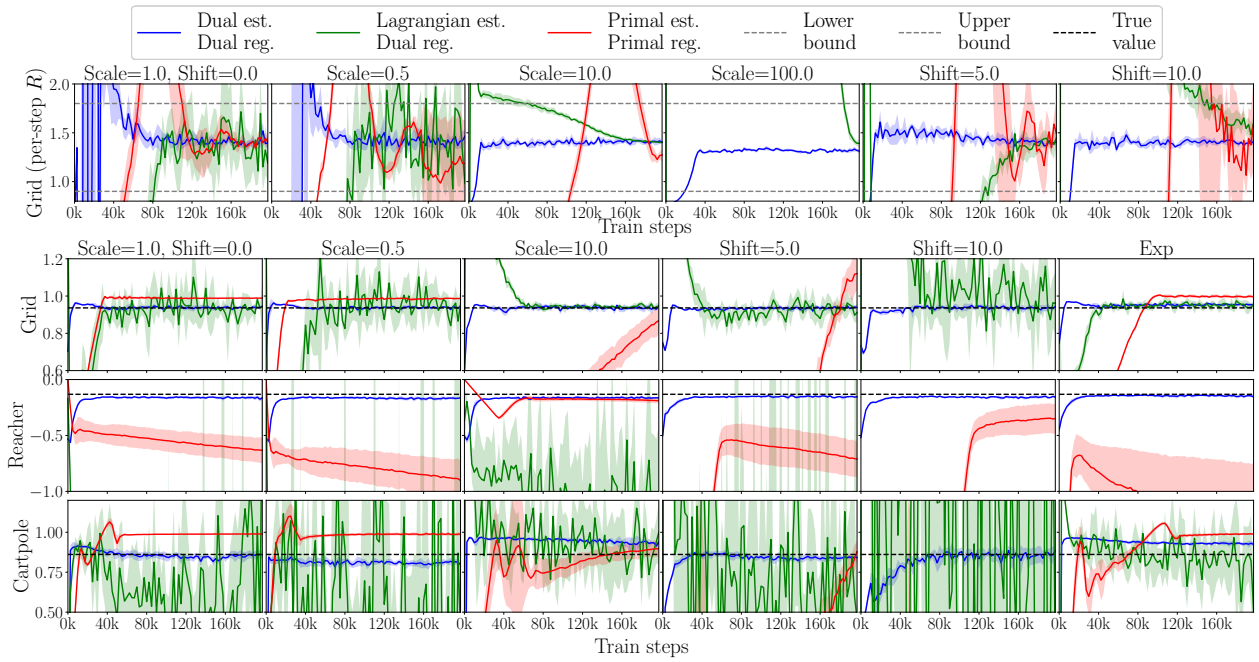


Figure 3.2: Primal (red), dual (blue), and Lagrangian (green) estimates under linear (top) and neural network (bottom) parametrization when rewards are transformed during training. Estimations are transformed back and plotted on the original scale. The dual estimates are robust to all transformations, whereas the primal and Lagrangian estimates are sensitive to the reward values.

Choice of Regularization (α_ζ, α_R, and α_Q) Next, we study the choice between regularizing the primal or dual variables. Given the results of Section 3.1.5, we focus on ablations using the dual estimator $\hat{\rho}_\zeta$ to estimate ρ_π. Results are presented in Figure 3.3. As expected, we see that regularizing the primal variables when $\alpha_R = 1$ leads to a biased estimate, especially in Grid (π_1), Reacher (π_2), and Cartpole. Regularizing the dual variable (blue lines) on the other hand does not incur such a bias. Additionally, the value of α_R has little effect on the final estimates when the dual variable is regularized (dotted versus solid blue lines). While the invariance to α_R may not generalize to other tasks, an advantage of the dual estimates with regularized dual variable is the flexibility to set $\alpha_R = 0$ or 1 depending on the reward function.

Choice of Redundant Constraints (λ and $\zeta \geq 0$) So far our experiments with the dual estimator used λ and $\zeta \geq 0$ in the optimizations, corresponding to the normalization and positive constraints in the d-LP. However, these are in principle not necessary when $\gamma < 1$, and so we evaluate the effect of removing them. Given the results of the previous sections, we focus our ablations on the use of the dual estimator $\hat{\rho}_\zeta$ with dual regularization

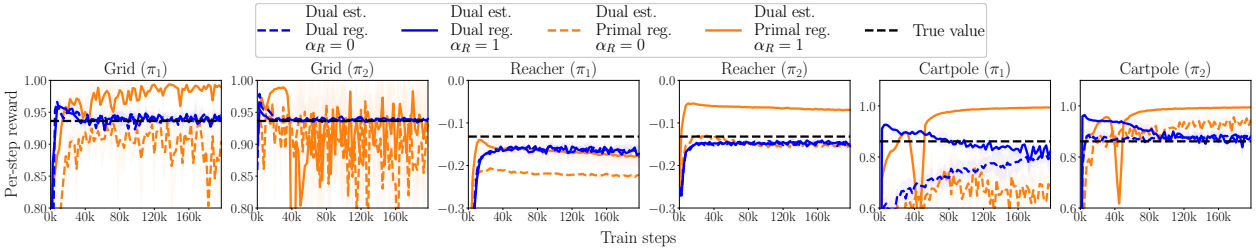


Figure 3.3: Dual estimates when $\alpha_R = 0$ (dotted line) and $\alpha_R = 1$ (solid line). Regularizing the dual variable (blue) is consistently better than regularizing the primal variable (orange). $\alpha_R \neq 0$ and $\alpha_Q \neq 0$ leads to biased estimation (solid orange). The value of α_R does not affect the final estimate when $\alpha_\zeta = 1, \alpha_Q = 0$.

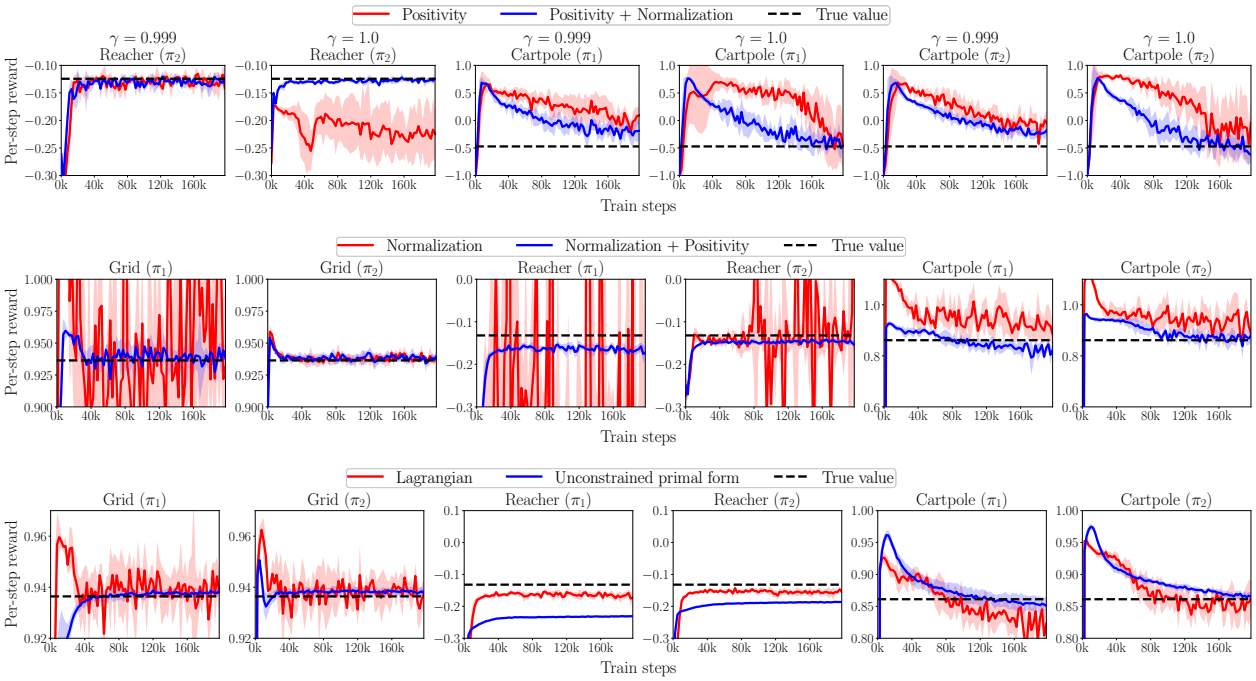


Figure 3.4: Apply positive constraint, normalization constraint, and the unconstrained primal form during optimization (blue curves). Positivity constraint (row 1) improves training stability. Normalization constraint is essential when $\gamma = 1$, and also helps when $\gamma < 1$ (row 2). Solving the unconstrained primal problem (row 3) can be useful when the action space is discrete.

$\alpha_\zeta > 0, \alpha_R = 1$.

Normalization. We consider the effect of removing the normalization constraint (λ).

Figure 3.4 (row 1) shows the effect of keeping (blue curve) or removing (red curve) this constraint during training. We see that training becomes less stable and approximation error increases, even when $\gamma < 1$.

Positivity. We continue to evaluate the effect of removing the positivity constraint $\zeta \geq 0$, which, in our previous experiments, was enforced via applying a square function to the dual variable neural network output. Results are presented in Figure 3.4 (row 2), where we again see that the removal of this constraint is detrimental to optimization stability and estimator accuracy.

Choice of Optimization (Lagrangian or Unconstrained Primal Form) So far, our experiments have used minimax optimization via the Lagrangian to learn primal and dual variables. We now consider solving the unconstrained primal form of the d-LP, which Section 3.1.3.2 suggests may lead to an easier, but biased, optimization. Figure 3.4 (row 3) indeed shows that the unconstrained primal reduces variance on Grid and produces better estimates on Cartpole. Both environments have discrete action spaces. Reacher, on the other hand, has a continuous action space, which creates difficulty when taking the expectation over next step samples, causing bias in the unconstrained primal form. Given this mixed performance, we generally advocate for the Lagrangian, unless the task is discrete-action and the stochasticity of the dynamics is known to be low.

Chapter 4

Representation Learning from Suboptimal Data

From the previous chapter, we saw that offline interactive dataset can be used to directly evaluate or improve reinforcement learning (RL) policies. Nevertheless, offline RL makes assumptions about data coverage which can be difficult to satisfy in practice. Furthermore, offline RL algorithms can be sensitive to hyperparamters. It is therefore desirable to consider alternative usage of the offline interactive dataset.

In vision and language, data from other tasks or broader dataset such as those from the internet can often be used for representation learning. For instance, CLIP [2] uses internet data to learn representations of images and text so that images and texts of the same instance have similar representations. While representation learning objectives such as contrastive learning [8] has been shown to be effective in many settings, the principled understanding of why these representation learning objectives can accelerate downstream learning has been limited. Decision making is a more complex setting compared to downstream supervised learning, hence it is non-trivial to provide principled analysis behind representation learning for RL.

In this chapter, we explore an alternative of using the offline interactive dataset. Instead of directly train RL policies on such offline data, we first employ representation learning objectives to learn compact representations of states or actions, followed by downstream learning such as imitation or online/offline RL. In Section 4.1, we first conduct an empirical study of a broad array of representation learning objectives. From this empirical study, we found that a particular class of representation learning objectives — temporal contrastive learning, has consistently shown benefit for all environments across all downstream objectives. In Section 4.2, we follow up on such empirical observation to theoretically understand the benefit of contrastive objectives in learning compact representations of actions. Overall, this chapter shows that representation learning is an effective alternative to offline RL both empirically and theoretically.

4.1 Ablation Study of Representation Learning Objectives

4.1.1 Introduction

Within the reinforcement learning (RL) research field, *offline RL* has recently gained a significant amount of interest [76, 243]. Offline RL considers the problem of performing reinforcement learning – i.e., learning a policy to solve a sequential decision-making task – exclusively from a static, offline dataset of experience. The recent interest in offline RL is partly motivated by the success of *data-driven* methods in the supervised learning literature. Indeed, the last decade has witnessed ever more impressive models learned from ever larger static datasets [244, 245, 27, 246]. Solving offline RL is therefore seen as a stepping stone towards developing scalable, data-driven methods for policy learning [10]. Accordingly, much of the recent offline RL research focuses on proposing new policy optimization algorithms amenable to learning from offline datasets (e.g., [247, 77, 46, 78, 248, 249]).

In this section, we consider a slightly different approach to incorporating offline data into sequential decision-making. We are inspired by recent successes in semi-supervised learning [250, 1, 136], in which large and potentially unlabelled offline datasets are used to learn *representations* of the data – i.e., a mapping of input to a fixed-length vector embedding – and these representations are then used to accelerate learning on a downstream supervised learning task. We therefore consider whether the same paradigm can apply to RL. Can offline experience datasets be used to learn representations of the data that accelerate learning on a downstream task?

This broad and general question has been partially answered by previous works [14, 107]. These works focus on using offline datasets to learn representations of *behaviors*, or actions. More specifically, these works learn a spectrum of behavior policies, conditioned on a latent z, through supervised action-prediction on the offline dataset. The latent z then effectively provides an abstract action space for learning a hierarchical policy on a downstream task, and this straightforward paradigm is able to accelerate learning in a variety of sequential decision-making settings. Inspired by these promising results and to differentiate our own work, we focus our efforts on the question of representation learning for *observations*, or states, as opposed to learning representations of behaviors or actions. That is, we aim to answer the question, can offline experience datasets be used to learn representations *of state observations* such that learning policies from these pretrained representations, as opposed to the raw state observations, improves performance on a downstream task?[1]

To approach this question, we devise a variety of offline datasets and corresponding downstream tasks. For offline datasets, we leverage the Gym-MuJoCo datasets from D4RL [10],

[1]Whether the two aspects of representation learning – action representations and state representations – can be combined is an intriguing question. However, to avoid an overly broad study, we focus only on state representation learning, and leave the question of combining this with action representation learning to future work.

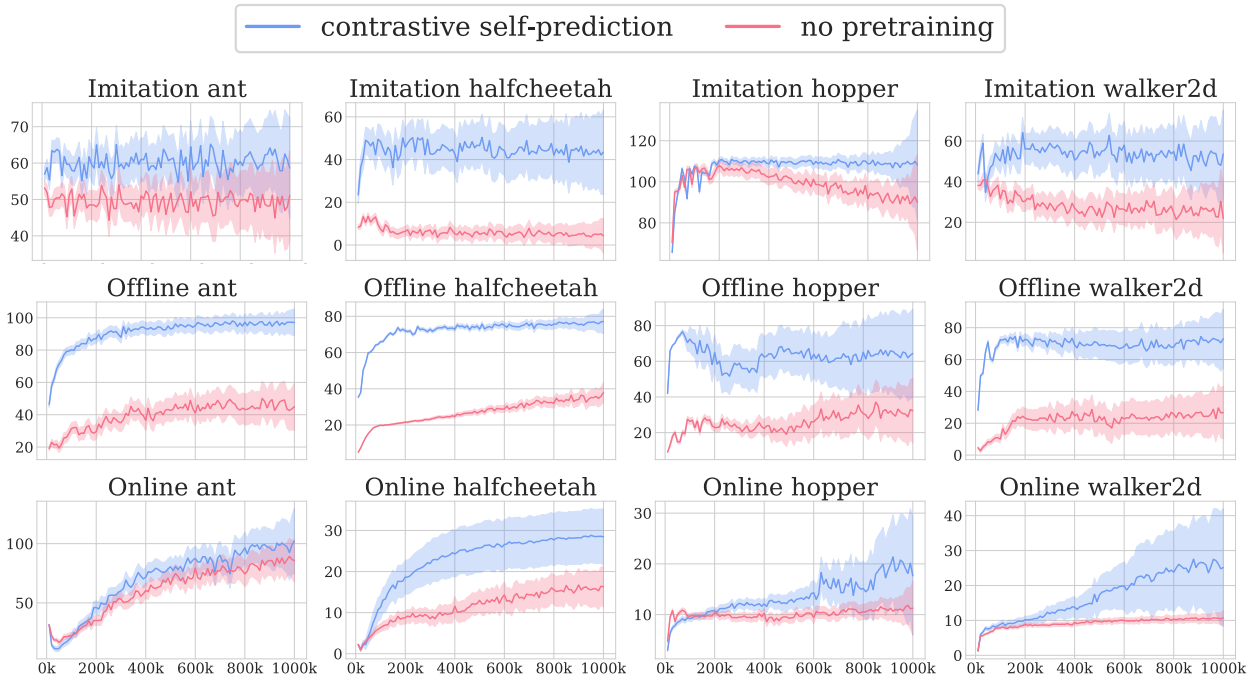


Figure 4.1: A summary of the advantages of the best-performing contrastive self-prediction variant as a pretraining representation learning objective, across a variety of settings: imitation learning, offline RL, and online RL. Each subplot shows the aggregated mean reward and standard error during training, with aggregation over offline datasets of different behavior (e.g., expert, medium, etc.), with five seeds per dataset (see Section 4.1.3). Contrastive self-prediction exhibits significant performance gains in all domains and tasks.

which provide a diverse set of datasets from continuous control simulated robotic environments. For downstream tasks, we consider three main categories: (1) low-data imitation learning, in which we aim to learn a task-solving policy from a small number of expert trajectories; (2) offline RL, in which we aim to learn a task-solving policy from the same offline dataset used for representation learning; and (3) online RL, in which we aim to learn a task-solving policy using online access to the environment.

Once these settings are established, we then continue to evaluate the ability of state representation learning on the offline dataset to accelerate learning on the downstream task. Our experiments are separated into two parts, *breadth* and *depth*. First for breadth, we consider a diverse variety of representation learning objectives taken from the RL and supervised learning literature. The results of these experiments show that, while several of these objectives perform poorly, a few yield promising results. This promising set essentially comprises of objectives which we call *contrastive self-prediction*; these objectives take sub-trajectories of experience and then use some components of the sub-trajectory to predict

other components, with a contrastive loss when predicting states – e.g., using a contrastive loss on the affinity between a sequence of states and actions and the next state, akin to popular methods in the supervised learning literature [250, 1].

These initial findings guide our second set of experiments. Aiming for depth, we devise an extensive ablation based on contrastive self-prediction to investigate what components of the objective are most important and in which settings. For example, whether it is important to include reward as part of the sub-trajectory, or whether discrete representations are better than continuous, whether pre-training and fixing the representations is better than finetuning, etc. In short, we find that state representation learning can yield a dramatic improvement in downstream learning. Compared to performing policy learning from raw observations, we show that relatively simple representation learning objectives on offline datasets can enable better and faster learning on imitation learning, offline RL, and and partially observable[2] online RL (see Figure 4.1). We believe these results are especially compelling for the imitation learning setting – where even a pretraining dataset that is far from expert behavior yields dramatic improvement in downstream learning – and in the offline RL setting – where we show the benefits of representation learning are significant even when the pretraining dataset *is the same as* the downstream task dataset. We hope that these impressive results guide and encourage future researchers to develop even better ways to incorporate representation learning into sequential decision-making.

4.1.2 Background and Related Work

Representation learning for RL has a rich and diverse existing literature, and we briefly review these relevant works.

Abstraction and Bisimulation Traditionally, representation learning has been framed as learning or identifying *abstractions* of the state or action space of an environment [168, 169, 167, 170]. These methods aim to reduce the original environment state and action spaces to more compact spaces by clustering those states and actions which yield similar rewards and dynamics. Motivated by similar intuitions, research into *bisimulation* metrics has aimed to devise or learn similarity functions between states [177, 178]. While these methods originally required explicit knowledge of the reward and dynamics functions of the environment, a number of recent works have translated these ideas to stochastic representation learning objectives using deep neural networks [171, 179, 172]. Many of these modern approaches effectively learn reward and transition functions in the learned embedding space, and training of these models is used to inform the learned state representations.

Representations in Model-Based Learning The idea of learning latent state representations via learning reward and dynamics models leads us to related work in the model-based RL literature. Several recent model-based RL methods use latent state representa-

[2]Results on fully observable online RL are in Appendix A.2.2.

tions as a way to simplify the model learning and policy rollout elements of model-based policy optimization [175, 251, 118], with the rollout in latent space sometimes referred to as 'imagination' [126, 127]. Similar ideas have also appeared under the label of 'embed to control' [252, 253]. Other than learning representations through forward models, there are also works which propose to learn *inverse* models, in which an action is predicted based on the representations of its preceding state and subsequent state [173, 254].

Contrastive Objectives Beyond model-based representations, many previous works propose the use of contrastive losses as a way of learning useful state representations [255, 256, 257, 258]. These works effectively define some notion of similarity between states and use a contrastive loss to encourage similar states to have similar representations. The similarity is usually based on either temporal vicinity (pairs of states which appear in the same sub-trajectory) or user-specified augmentations, such as random shifts of image observations [257]. Previous work has established connections between the use of contrastive loss and mutual information maximization [259] and energy-based models [260].

State Representation Learning in Offline RL The existing works mentioned above almost exclusively focus on online settings, often learning the representations on a continuously evolving dataset and in tandem with online policy learning. In contrast, our work focuses on representation learning on offline datasets and separated from downstream task learning. This serves two purposes: First, using static offline datasets makes comparisons between different methods easier, avoiding confounding factors associated with issues of exploration or nonstationary datasets. Second, the offline setting is arguably more practical; in practice, static offline datasets are more common than cheap online access to an environment [76]. Previous work in a similar vein to ours includes [258] and [254], which propose to use unsupervised pretraining, typically only on expert demonstrations, as a way of initializing an image encoder for downstream online RL. Our own work complements these existing studies, by presenting extensive comparisons of a variety of representation learning objectives in several distinct settings. Moreover, our work is unique for showing benefits of representation learning on non-image tasks, thus avoiding the use of any explicit or implicit prior knowledge that is typically exploited for images (e.g., using image-based augmentations or using a convolutional network architecture).

4.1.3 Task Setups

We now continue to our own contributions, starting by elaborating on the experimental protocol we design to evaluate representation learning in the context of low-data imitation learning, offline RL (specifically, offline policy optimization), and online RL in partially observable environments. This protocol is summarized in Table 4.1.

Table 4.1: A summary of our experimental setups. In total, there are 16 choices of offline data and downstream task combinations each for imitation learning, offline RL, and online RL. Given that we run each setting with five random seeds, this leads to a total of 240 training runs for every representation learning objective we consider.

Imitation		
domain $\in$ {halfcheetah, hopper, walker2d, ant}		data: {domain}-{data}-v0
data $\in$ {medium, medium-replay}	$\rightarrow$	task: BC on first N
$N \in \{10000, 25000\}$		from {domain}-expert-v0
Offline RL		
domain $\in$ {halfcheetah, hopper, walker2d, ant}		data: {domain}-{data}-v0
data $\in$ {expert, medium-expert, medium, medium-replay}	$\rightarrow$	task: BRAC on {domain}-{data}-v0
Online RL		
domain $\in$ {halfcheetah, hopper, walker2d, ant}		data: {domain}-{data}-v0 masking
data $\in$ {expert, medium-expert, medium, medium-replay}	$\rightarrow$	task: SAC on randomly masked version of {domain}

Datasets We leverage the Gym-MuJoCo datasets from D4RL [10]. These datasets are generated from running policies on the well-known MuJoCo benchmarks of simulated locomotive agents: halfcheetah, hopper, walker2d, and ant. Each of these four domains is associated with four datasets – expert, medium-expert, medium, and medium-replay – corresponding to the quality of the policies used to collect that data. Each dataset is composed of a number of trajectories $\tau := (s_0, a_0, r_0, s_1, a_1, r_1, \ldots, s_T)$. For example, the dataset ant-expert-v0 is a dataset of trajectories generated by expert task-solving policies on the ant domain, while the dataset halfcheetah-medium-v0 is generated by mediocre, far from task-solving, policies.

Notably, although the underyling MuJoCo environments are Markovian, the datasets are not necessarily Markovian, as they may be generated by multiple distinct policies.

Imitation Learning in Low-Data Regime Imitation learning [261] seeks to match the behavior of an agent with that of an expert. While expert demonstrations are often limited and expensive to obtain in practice, non-expert experience data (e.g., generated from a mediocre agent randomly interacting with an environment) can be much more easily accessible.

To mimic this practical scenario, we consider an experimental protocol in which the downstream task is behavioral cloning [262] on a small set of expert trajectories – selected by taking either the first 10k or 25k transitions from an expert dataset in D4RL, corresponding to about 10 and 25 expert trajectories, respectively. We then consider either the medium

or medium-replay datasets from the same domain for representation learning.[3] Thus, this set of experiments aims to determine whether representations learned from large datasets of mediocre behavior can help elevate the performance of behavioral cloning on a much smaller expert dataset.

Offline RL with Behavior Regularization One of the main motivations for the introduction of the D4RL datasets was to encourage research into fully offline reinforcement learning; i.e., whether it is possible to learn return-maximizing policies exclusively from a static offline dataset. Many algorithms for this setting have recently been proposed, commonly employing some sort of *behavior regularization* [263, 264, 77]. In its simplest form, behavior regularization augments a vanilla actor-critic algorithm with a divergence penalty measuring the divergence of the learned policy from the offline data, thus compelling the learned policy to stay close to the actions appearing in the dataset.

While the actor and critic are typically trained with the raw observations as input, with this next set of experiments, we aim to determine whether representation learning can help in this regime as well. In this setting, the pretraining and downstream datasets are the same, determined by a single choice of domain (halfcheetah, hopper, walker2d, or ant) and data (expert, medium-expert, medium, or medium-replay). For the downstream algorithm, we use behavior regularized actor-critic (BRAC) [77], which is a simple behavior regularized method employing a KL divergence penalty. Notably, although the original BRAC work uses different regularization strengths and policy learning rates for different domains, we fix these to values which we found to generally perform best (regularization strength of 1.0 and policy learning rate of 0.00003).

Thus, this set of experiments aims to determine whether learning BRAC from learned state representations is better (in terms of performance and less dependence on hyperparameters) than learning BRAC from the raw states, even when the state representations are learned using the same offline dataset.

Online RL in Partially Observable Environments In this set of experiments, we aim to determine whether representations learned from offline datasets can improve or accelerate learning in an online domain. One of the most popular online RL algorithms is soft actor critic (SAC) [265]. SAC is a well-performing algorithm on its own, and so to increase the difficulty of the downstream task, we consider a simple modification to make our domains partially observable: zero-masking out a random dimension of the state observation. This modification also brings our domains closer to practice, where partial observability due to flaky sensor readings is common [266]. For those interested, we include results of representation learning on the standard, fully-observable MuJoCo environments in Appendix A.2.2.

Accordingly, the offline dataset is determined by a choice of domain (halfcheetah, hopper, walker2d, or ant) and data (expert, medium-expert, medium, or medium-replay), with the

[3]To avoid issues of extrapolation when transferring learned representations to the expert dataset, we include the small number of expert demonstrations in the offline dataset during pretraining.

same masking applied to this dataset. Representations learned on this dataset are then applied downstream, where SAC is trained on the online domain, with the representation module providing an embedding of the masked observations of the environment within a learned embedding space.

Evaluation Each representation learning variant we evaluate is run with five seeds on each of the experimental setups described above. Unless otherwise noted, a single seed corresponds to an initial pretraining phase of 200k steps, in which a representation learning objective is optimized using batches of 256 sub-trajectories randomly sampled from the offline dataset. After pretraining, the learned representation is fixed and applied to the downstream task, which performs the appropriate training (BC, BRAC, or SAC) for 1M steps. In this downstream phase, every 10k steps, we evaluate the learned policy on the downstream domain environment by running it for 10 episodes and computing the average total return. We normalize this total return according to the normalization proposed in [10], such that a score of 0 roughly corresponds to a random agent and a score of 100 to an expert agent. We average the last 10 evaluations within the 1M downstream training, and this determines the final score for the run. To aggregate over multiple seeds and task setups, we simply compute the average and standard error of this final score.

4.1.4 Experiments: Breadth Study

We begin our empirical study with an initial assessment into the performance of a broad set of representation learning ideas from the existing literature.

4.1.5 Representation Learning Objectives

We describe the algorithms we consider below. While it is infeasible for us to extensively evaluate all previously proposed representation learning objectives, our choice of objectives here aims to cover a diverse set of recurring themes and ideas from previous work (see Section 4.1.2).

We use the notation

$$\tau_{t:t+k} := (s_t, a_t, r_t, \ldots, s_{t+k-1}, a_{t+k-1}, r_{t+k-1}, s_{t+k})$$

to denote a length-$(k+1)$ sub-trajectory of state observations, actions, and rewards; we use $s_{t:t+k}, a_{t:t+k}, r_{t:t+k}$ to denote a subselection of this trajectory based on states, actions, and rewards, respectively. We use ϕ to denote the representation function; i.e., $\phi(s)$ is the representation associated with state observation s, and $\phi(s_{t:t+k}) := (\phi(s_t), \ldots, \phi(s_{t+k}))$. All learned functions, including ϕ, are parameterized by neural networks. Unless otherwise noted, ϕ is parameterized as a two-hidden layer fully-connected network with 256 units per layer and output of dimension 256 (see further details in Appendix A.2.1).

Inverse model Given a sub-trajectory $\tau_{t:t+1}$, use $\phi(s_{t:t+1})$ to predict a_t. That is, we train an auxiliary f such that $f(\phi(s_{t:t+1}))$ is a distribution over actions, and the learning objective is $-\log P(a_t|f(\phi(s_{t:t+1})))$. This objective may be generalized to sequences longer than $k+1=2$ as $-\log P(a_{t+k-1}|f(\phi(s_{t:t+k}), a_{t:t+k-1}))$.

Forward raw model Given a sub-trajectory $\tau_{t:t+1}$, use $\phi(s_t), a_t$ to predict r_t, s_{t+1}. That is, we train an auxiliary f, g such that $f(\phi(s_t), a_t)$ is a distribution over next states and $g(\phi(s_t), a_t)$ is a scalar reward prediction. The learning objective is $||r_t - g(\phi(s_t), a_t)||^2 - \log P(s_{t+1}|f(\phi(s_t), a_t))$. This objective may be generalized to sequences longer than $k+1=2$ as $||r_t - g(\phi(s_{t:t+k-1}), a_{t:t+k-1})||^2 - \log P(s_{t+1}|f(\phi(s_{t:t+k-1}), a_{t:t+k}))$.

Forward latent model; a.k.a., DeepMDP [171] This is the same as the forward raw model, only that f now describes a distribution over next state representations. Thus, the log-probability with respect to f becomes $-\log P(\phi(s_{t+1})|f(\phi(s_t), a_t))$.

Forward energy model This is the same as the forward raw model, only that f is no longer a distribution over raw states. Rather, f maps $\phi(s_t), a_t$ to the same embedding space as ϕ and the probability $P(s_{t+1}|f(\phi(s_t), a_t))$ is defined in an energy-based way:

$$\frac{\rho(s_{t+1}) \exp\{\phi(s_{t+1})^\top W f(\phi(s_t), a_t)\}}{\mathbb{E}_\rho[\exp\{\phi(\tilde{s})^\top W f(\phi(s_t), a_t)\}]}, \tag{4.1}$$

where W is a trainable matrix and ρ is a non-trainable prior distribution (we set ρ to be the distribution of states in the offline dataset).

(Momentum) temporal contrastive learning (TCL) Given a sub-trajectory $\tau_{t:t+1}$, we apply a contrastive loss between $\phi(s_t), \phi(s_{t+1})$. The objective is

$$-\phi(s_{t+1})^\top W \phi(s_t) + \log \mathbb{E}_\rho[\exp\{\phi(\tilde{s})^\top W \phi(s_t)\}], \tag{4.2}$$

where W and ρ are as in the forward energy model above. This objective may be generalized to sequences longer than $k+1=2$ by having multiple terms in the loss for $i=1,\ldots,k$:

$$-\phi(s_{t+i})^\top W_i \phi(s_t) + \log \mathbb{E}_\rho[\exp\{\phi(\tilde{s})^\top W_i \phi(s_t)\}]. \tag{4.3}$$

If momentum is used, we apply the contrastive loss between $f(\phi(s_t))$ and $\phi_{target}(s_{t+i})$, where f is a learned function and ϕ_{target} denotes a non-trainable version of ϕ, with weights corresponding to a slowly moving average of the weights of ϕ, as in [258, 267].

Attentive Contrastive Learning (ACL) Following the theme of contrastive losses and inspired by a number of works in the RL [259] and NLP [250] literature which apply such losses between tokens and contexts using an attention mechanism, we devise a similar objective for our settings. Implementation-wise, we borrow ideas from BERT [1], namely we

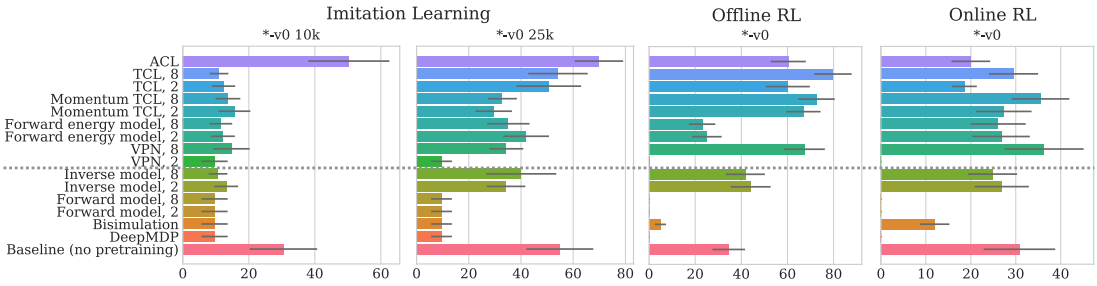


Figure 4.2: Performance of downstream imitation learning, offline RL, and online RL tasks under a variety of representation learning objectives. x-axis shows aggregated average rewards (over five seeds) across the domains and datasets described in Section 4.1.3. Methods that failed to converge are eliminated from the results (see Appendix A.2.1). ACL is set to the default configuration that favors imitation learning (see Section 4.1.6). When applicable, we also label variants with $k+1 \in \{2, 8\}$. Methods above the dotted line are variants of contrastive self-prediction. ACL performs well on imitation learning. VPN and (momentum) TCL perform well on offline and online RL.

(1) take a sub-trajectory $s_{t:t+k}, a_{t:t+k}, r_{t:t+k}$, (2) randomly mask a subset of these, (3) pass the masked sequence into a transformer, and then (4) for each masked input state, apply a contrastive loss between its representation $\phi(s)$ and the transformer output at its sequential position. We use $k+1=8$ in our implementation. Figure 4.5 provides a diagram of ACL.

Value prediction network (VPN) Taken from [175], this objective uses an RNN starting at $\phi(s_t)$ and inputting $a_{t:t+k}$ for k steps to predict the k-step future rewards and value functions. While the original VPN paper defines the $(k+1)$-th value function in terms of a max over actions, we avoid this potential extrapolation issue and simply use the $(k+1)$-th action provided in the offline data. As we will elaborate on later, VPN bears similarities to ACL in that it uses certain components of the input sequence (states and actions) to predict other components (values).

Deep bisimulation for control This objective is taken from [179], where the representation function ϕ is learned to respect an L1 distance based on a bisimulation similarity deduced from Bellman backups.

Results The results of these representation learning objectives are presented in Figure 4.2. Representation learning, even before the extensive ablations we will embark on in Section 4.1.6, on average improves downstream imitation learning, offline RL, and online RL tasks by 1.5x, 2.5x, and 15% respectively. The objectives that appear to work best – ACL, (Momentum) TCL, VPN – fall under a class of objectives we term *contrastive self-prediction*, where *self-prediction* refers to the idea that certain components of a sub-trajectory are pre-

dicted based on other components of the same sub-trajectory, while *contrastive* refers to the fact that this prediction should be performed via a contrastive energy-based loss when the predicted component is a state observation.

We also find that a longer sub-trajectory $k + 1 = 8$ is generally better than a short one $k + 1 = 2$. The advantage here is presumably due to the non-Markovian nature of the dataset. Even if the environment is Markovian, the use of potentially distinct policies for data collection can lead to non-Markovian data.

Despite these promising successes, there are a number of objectives which perform poorly. Raw predictions of states (forward model) yields disappointing results in these settings. Forward models of future representations – DeepMDP, Bisimulation – also exhibit poor performance. This latter finding was initially surprising to us, as many theoretical notions of state abstractions are based on the principle of predictability of future state representations. Nevertheless, even after extensive tuning of these objectives and attempts at similar objectives (e.g., we briefly investigated incorporating ideas from [118]), we were not able to achieve any better results. Even if it is possible to find better architectures or hyperparameters, we believe the difficulty in tuning these baselines still makes them unattractive in comparison to the simpler and better performing alternatives.

4.1.6 Experiments: Depth Study

The favorable results of objectives based on the idea of contrastive self-prediction is compelling, but the small number of objectives evaluated leaves many questions unanswered. For example, when generating the context embedding for a specific prediction, should one use past states (as in TCL and Momentum TCL) or also include actions and/or rewards (as in ACL and VPN)? Should this context use the same representation network ϕ (as in TCL and VPN), a momentum version of it (as in Momentum TCL), or a completely separate network (as in ACL)?

We use this section to study these and other important questions by conducting a series of ablations on the factors which compose a specific contrastive self-prediction objective and how it is applied to downstream learning. We describe all these factors in Table 4.2, as well as a high-level summary of their effects. Further anecdotal observations found during our research are summarized in Appendix A.2.3.

We choose the transformer-based implementation of ACL to serve as the skeleton for all these ablations (see Figure 4.5), due to its general favorable empirical performance in the previous section, as well as its ease of modification. For each downstream task below, we present the ablations with respect to the *default* configuration of the factors in Table 4.2 that corresponds to the original ACL introduced in Section 4.1.4, and change one factor at a time to observe its effect on downstream task performance.

Results The results of our ablation studies are presented in Figure 4.4, and we highlight some of the main findings below. We also take the best performing ablation from each row (imitation, offline RL, and online RL) and plot the performance during training in Figure 4.1.

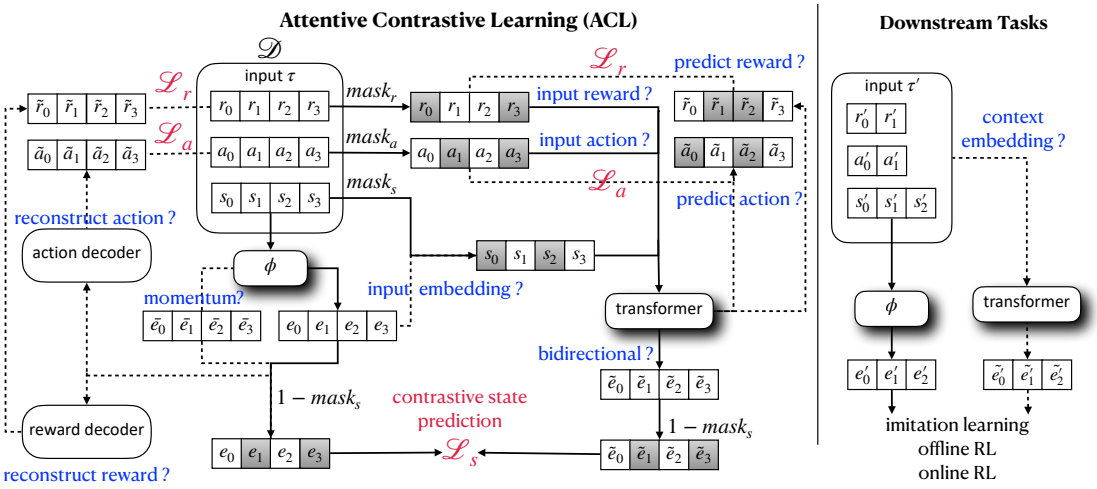


Figure 4.3: A pictoral representation of our depth study based on contrastive self-prediction. We use the transformer-based architecture of attentive contrastive learning (ACL) as a skeleton for ablations with respect to various representation learning details. Solid arrows correspond to the configuration of ACL. Dotted arrows and blue text are factors considered in the ablation study. Gray blocks are masked state/action/reward entries. After the pretraining phase, the representation network ϕ is reused for downstream tasks, unless 'context embedding' is true, in which case the transformer is used.

Further results are available in Appendix A.2.2, and more of our interpretations of these results are included in Appendix A.2.4.

Let us first consider the effects of inclusion or prediction of actions and rewards. We notice some interesting behavior across the different downstream modes. Namely, it appears that imitation learning is best served by focusing only on state contrastive learning and not including or predicting actions and rewards, whereas the offline and online RL settings appear to benefit from these. Due to the mixed results we initially observed from including or predicting actions and rewards, we also introduce the idea of *reconstructing* actions and rewards based on $\phi(s)$, and we found this to have much more consistent benefit in the RL settings, although it still degrades imitation learning performance. This disconnect between objectives which are good for imitation learning vs. RL, first seen in Section 4.1.4, thus continues to be present in these ablations as well, and we find that no single objective dominates in all settings.

We also evaluate a number of representation learning paradigms popular in the NLP literature [1], namely using bidirectional transformers, finetuning, and context embedding. Although these techniques are ubiquitous in the NLP literature, we find mixed results in RL settings. Context embedding consistently hurts performance. Bidirectional transformer hurts imitation learning but helps online RL. Finetuning leads to a modest degredation in performace in imitation and offline RL but can improve online RL depending on the domain

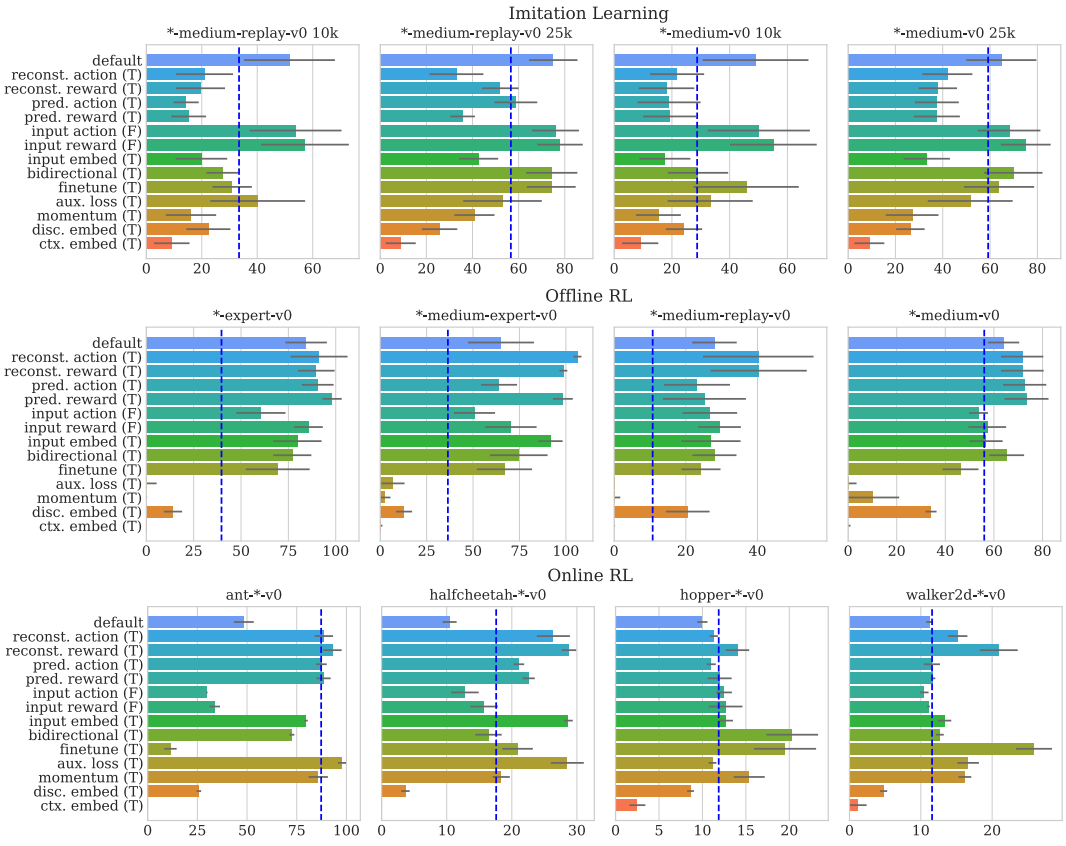


Figure 4.4: Ablation results on imitation learning, offline RL, and online RL. x-axis shows average rewards and standard error aggregated over either different Gym-MuJoCo datasets (imitation and offline RL) or domains (online RL). Blue dotted lines show average rewards without pretraining. (T) and (F) mean setting each factor to true or false (opposite from the default configuration). Reconstructing, predicting, or inputting action or reward (row 2-7) impairs imitation performance but are important for offline and online RL. Bidirectional transformer hurts imitation learning when downstream sample size is small. Finetuning and auxiliary loss can help online RL. Additional results are presented in Appendix A.2.2.

being evaluated.

We additionally considered using the representation learning objective as an auxiliary training loss, which is popular in the online RL literature [254, 258]. And indeed, we find that it can dramatically improve representation learning in online RL, but at the same time, dramatically degrade performance in the offline settings (imitation learning or offline RL).

4.2 Representation Learning with Contrastive Fourier Features

4.2.1 Introduction

Imitation learning uses expert demonstration data to learn sequential decision making policies [268]. Such demonstrations, often produced by human experts, can be costly to obtain in large number. On the other hand, practical application domains, such as recommendation [269] and dialogue [194] systems, provide large quantities of offline data generated by suboptimal agents. Since the offline data is suboptimal in performance, using it directly for imitation learning is infeasible. While some prior works have proposed using suboptimal offline data for offline reinforcement learning (RL) [270, 77, 76], this would require reward information, which may be unavailable or infeasible to compute from suboptimal data [271]. Nevertheless, conceptually, suboptimal offline datasets should contain useful information about the environment, if only we could distill that information into a useful form that can aid downstream imitation learning.

One approach to leveraging suboptimal offline datasets is to use the offline data to extract a lower-dimensional *latent action space*, and then perform imitation learning on an expert dataset using this latent action space. If the latent action space is learned properly, one may hope that performing imitation learning in the latent space can reduce the need for large quantities of expert data. While a number of prior works have studied similar approaches in the context of hierarchical imitation and RL setting [272, 273, 274, 275, 276, 277, 14, 13, 278], such methods typically focus on the theoretical and practical benefits of *temporal abstraction* by extracting temporally extended skills from data or experience. That is, the main benefit of these approaches is that the latent action space operates at a lower temporal frequency than the original environment action space. We instead focus directly on the question of *action representation*: instead of learning skills that provide for temporal abstraction, we aim to directly reparameterize the action space in a way that provides for more sample-efficient downstream imitation without the need to reduce control frequency. Unlike learning temporal abstractions, action reparamtrization does not have to rely on any hierarchical structures in the offline data, and can therefore utilize highly suboptimal datasets (e.g., with random actions).

Aiming for a provably-efficient approach to utilizing highly suboptimal offline datasets, we use first principles to derive an upper bound on the quality of an imitation learned policy involving three terms corresponding to (4.4) action representation and (4.5) action decoder learning on a suboptimal offline dataset, and finally, (4.6) behavioral cloning (i.e., max-likelihood learning of latent actions) on an expert demonstration dataset. The first term in our bound immediately suggests a practical offline training objective based on a transition dynamics loss using an *factored* transition model. We show that under specific factorizations (e.g., low-dimensional or linear), one can guarantee improved sample efficiency on the expert dataset. Crucially, our mathematical results avoid the potential shortcomings

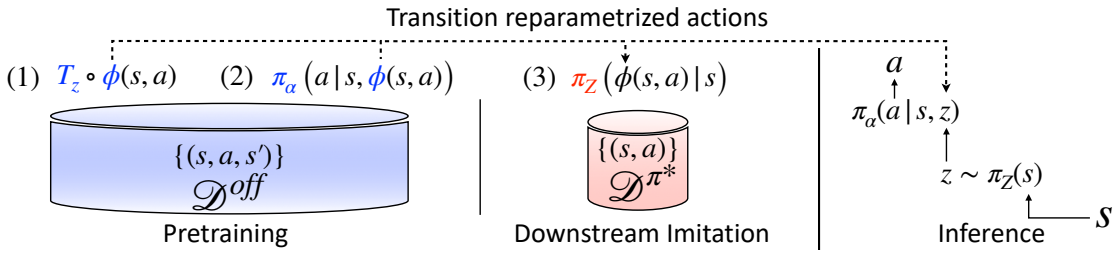


Figure 4.5: The TRAIL framework. Pretraining learns a factored transition model $\mathcal{T}_Z \circ \phi$ and an action decoder q on $\mathcal{D}_{\text{off}}$. Downstream imitation learns a latent policy π_Z on $\mathcal{D}^*_{\text{off}}$ with expert actions reparametrized by ϕ. During inference, π_Z and q are combined to sample an action.

of temporal skill extraction, as our bound is guaranteed to hold even when there is no temporal abstraction in the latent action space.

We translate these mathematical results into an algorithm that we call *Transition-Reparametrized Actions for Imitation Learning* (TRAIL). As shown in Figure 4.5, TRAIL consists of a pretraining stage (corresponding to the first two terms in our bound) and a downstream imitation learning stage (corresponding to the last term in our bound). During the pretraining stage, TRAIL uses an offline dataset to learn a factored transition model and a paired action decoder. During the downstream imitation learning stage, TRAIL first reparametrizes expert actions into the latent action space according to the learned transition model, and then learns a latent policy via behavioral cloning in the latent action space. During inference, TRAIL uses the imitation learned latent policy and action decoder in conjunction to act in the environment. In practice, TRAIL parametrizes the transition model as an energy-based model (EBM) for flexibility and trains the EBM with a contrastive loss. The EBM enables the low-dimensional factored transition model referenced by our theory, and we also show that one can recover the *linear* transition model in our theory by approximating the EBM with random Fourier features [279].

To summarize, our contributions include (i) a provably beneficial objective for learning action representations without temporal abstraction and (ii) a practical algorithm for optimizing the proposed objective by learning an EBM or linear transition model. An extensive evaluation on a set of navigation and locomotion tasks demonstrates the effectiveness of the proposed objective. TRAIL's empirical success compared to a variety of existing methods suggests that the benefit of learning *single-step* action representations has been overlooked by previous temporal skill extraction methods. Additionally, TRAIL significantly improves behavioral cloning even when the offline dataset is unimodal or highly suboptimal (e.g., obtained from a random policy), whereas temporal skill extraction methods lead to *degraded* performance in these scenarios. Lastly, we show that TRAIL, without using reward labels, can perform similarly or better than offline reinforcement learning (RL) with orders of magnitude less expert data, suggesting new ways for offline learning of squential decision making policies.

4.2.2 Related Work

Learning action abstractions is a long standing topic in the hierarchical RL literature [272, 273, 274, 275, 277]. A large body of work focusing on *online skill discovery* have been proposed as a means to improve exploration and sample complexity in online RL. For instance, [280, 281, 282, 283, 284] propose to learn a diverse set of skills by maximizing an information theoretic objective. Online skill discovery is also commonly seen in a hierarchical framework that learns a continuous space [276, 285, 277, 286] or a discrete set of lower-level policies [287, 288, 289], upon which higher-level policies are trained to solve specific tasks. Different from these works, we focus on learning action representations *offline* from a fixed suboptimal dataset to accelerate imitation learning.

Aside from online skill discovery, *offline skill extraction* focuses on learning temporally extended action abstractions from a fixed offline dataset. Methods for offline skill extraction generally involve maximum likelihood training of some latent variable models on the offline data, followed by downstream planning [98], imitation learning [290, 14, 278], offline RL [14, 291], or online RL [292, 293, 294, 295, 107, 13, 12, 296] in the induced latent action space. Among these works, those that provide a theoretical analysis attribute the benefit of skill extraction predominantly to increased temporal abstraction as opposed to the learned action space being any "easier" to learn from than the raw action space [14, 256]. Unlike these methods, our analysis focuses on the advantage of a lower-dimensional reparametrized action space agnostic to temporal abstraction. Our method also applies to offline data that is highly suboptimal (e.g., contains random actions) and potentially unimodal (e.g., without diverse skills to be extracted), which have been considered challenging by previous work [14].

While we focus on reducing the complexity of the action space through the lens of action representation learning, there exists a disjoint set of work that focuses on accelerating RL with *state* representation learning [297, 298, 178, 171, 179, 299, 7], some of which have proposed to extract a latent state space from a learned dynamics model. Analogous to our own derivations, these works attribute the benefit of representation learning to a smaller latent state space reduced from a high-dimensional input state space (e.g., images). Lastly, there exist model-based approaches that utilizes offline data to learn model dynamics which in tern accelerates imitation [300, 301]. These work differ from our focus of using the offline data to learn latent action space.

4.2.3 Preliminaries

In this section, we introduce the problem statements for imitation learning and learning-based control, and define relevant notations.

Learning goal. Imitation learning aims to recover an *expert policy* π^* with access to only a fixed set of samples from the expert: $\mathcal{D}^*_{\text{off}} = \{(s_i, a_i)\}_{i=1}^n$ with $s_i \sim d^{\pi^*}$ and $a_i \sim \pi^*(s_i)$. One approach to imitation learning is to learn a policy π that minimizes some discrepancy between π and π^*. In our analysis, we will use the total variation (TV) divergence in state

visitation distributions,

$$\text{Diff}(\pi, \pi^*) = D_{\text{TV}}(d^\pi \| d^{\pi^*}),$$

as the way to measure the discrepancy between π and π^*. Our bounds can be easily modified to apply to other divergence measures such as the Kullback–Leibler (KL) divergence or difference in expected future returns. *Behavioral cloning* (BC) [62] solves the imitation learning problem by learning π from $\mathcal{D}^*_{\text{off}}$ via a maximum likelihood objective

$$J_{\text{BC}}(\pi) := \mathbb{E}_{(s,a)\sim(d^{\pi^*},\pi^*)}[-\log \pi(a|s)],$$

which optimizes an upper bound of $\text{Diff}(\pi, \pi^*)$ defined above [302, 7]:

$$\text{Diff}(\pi, \pi^*) \leq \frac{\gamma}{1-\gamma}\sqrt{\frac{1}{2}\mathbb{E}_{d^{\pi^*}}[D_{\text{KL}}(\pi^*(s)\|\pi(s))]} = \frac{\gamma}{1-\gamma}\sqrt{\text{const}(\pi^*) + \frac{1}{2}J_{\text{BC}}(\pi)}.$$

BC with suboptimal offline data. The standard BC objective (i.e., direct max-likelihood on $\mathcal{D}^*_{\text{off}}$) can struggle to attain good performance when the amount of expert demonstrations is limited [303, 304]. We assume access to an additional *suboptimal* offline dataset $\mathcal{D}_{\text{off}} = \{(s_i, a_i, s'_i)\}_{i=1}^m$, where the suboptimality is a result of (i) suboptimal action samples $a_i \sim \text{Unif}_A$ and (ii) lack of reward labels. We use $(s, a, s') \sim d^{\text{off}}$ as a shorthand for simulating finite sampling from $\mathcal{D}_{\text{off}}$ via $s_i \sim d^{\text{off}}, a_i \sim \text{Unif}_A, s'_i \sim \mathcal{T}(s_i, a_i)$, where d^{off} is an *unknown* offline state distribution. We assume d^{off} sufficiently covers the expert distribution; i.e., $d^{\pi^*}(s) > 0 \Rightarrow d^{\text{off}}(s) > 0$ for all $s \in S$. The uniform sampling of actions in $\mathcal{D}_{\text{off}}$ is largely for mathematical convenience, and in theory can be replaced with any distribution uniformly bounded from below by $\eta > 0$, and our derived bounds will be scaled by $\frac{1}{|A|\eta}$ as a result. This works focuses on how to utilize such a suboptimal $\mathcal{D}_{\text{off}}$ to provably accelerate BC.

4.2.4 Near-Optimal Imitation Learning with Reparametrized Actions

In this section, we provide a provably-efficient objective for learning action representations from suboptimal data. Our initial derivations (Theorem 5) apply to general policies and latent action spaces, while our subsequent result (Theorem 7) provides improved bounds for specialized settings with continuous latent action spaces. Finally, we present our practical method TRAIL for action representation learning and downstream imitation learning.

Performance Bound with Reparametrized Actions Despite $\mathcal{D}_{\text{off}}$ being highly suboptimal (e.g., with random actions), the large set of (s, a, s') tuples from $\mathcal{D}_{\text{off}}$ reveals the transition dynamics of the environment, which a latent action space should support. Under this motivation, we propose to learn a *factored* transition model $\overline{\mathcal{T}} := \mathcal{T}_Z \circ \phi$ from the offline dataset $\mathcal{D}_{\text{off}}$, where $\phi : S \times A \to Z$ is an action representaiton function and

$\mathcal{T}_Z : S \times Z \to \Delta(S)$ is a latent transition model. Intuitively, good action representations should enable good imitation learning.

We formalize this intuition in the theorem below by establishing a bound on the quality of a learned policy based on (4.4) an offline pretraining objective for learning ϕ and $\mathcal{T}_Z$, (4.5) an offline decoding objective for learning an action decoder q, and (4.6) a downstream imitation learning objective for learning a latent policy π_Z with respect to latent actions determined by ϕ.

Theorem 5. *Consider an action representation function $\phi : S \times A \to Z$, a factored transition model $\mathcal{T}_Z : S \times Z \to \Delta(S)$, an action decoder $q : S \times Z \to \Delta(A)$, and a tabular latent policy $\pi_Z : S \to \Delta(Z)$. Define the transition representation error as*

$$J_{\mathrm{T}}(\mathcal{T}_Z, \phi) := \mathbb{E}_{(s,a)\sim d^{\mathrm{off}}} \left[D_{\mathrm{KL}}(\mathcal{T}(s,a) \| \mathcal{T}_Z(s, \phi(s,a))) \right],$$

the action decoding error as

$$J_{\mathrm{DE}}(q, \phi) := \mathbb{E}_{(s,a)\sim d^{\mathrm{off}}}[-\log q(a|s, \phi(s,a))],$$

and the latent behavioral cloning error as

$$J_{\mathrm{BC},\phi}(\pi_Z) := \mathbb{E}_{(s,a)\sim (d^{\pi^*}, \pi^*)}[-\log \pi_Z(\phi(s,a)|s)].$$

Then the TV divergence between the state visitation distributions of $q \circ \pi_Z : S \to \Delta(A)$ and π^*

$$\mathrm{Diff}(q \circ \pi_Z, \pi^*) \leq$$

can be bounded as

$$\begin{cases} \textit{Pretraining} & C_1 \cdot \sqrt{\frac{1}{2} \underbrace{\mathbb{E}_{(s,a)\sim d^{\mathrm{off}}} \left[D_{\mathrm{KL}}(\mathcal{T}(s,a) \| \mathcal{T}_Z(s, \phi(s,a))) \right]}_{= J_{\mathrm{T}}(\mathcal{T}_Z, \phi)}} \\ & + C_2 \cdot \sqrt{\frac{1}{2} \underbrace{\mathbb{E}_{s\sim d^{\mathrm{off}}}[\max_{z \in Z} D_{\mathrm{KL}}(\pi_{\alpha^*}(s,z) \| q(s,z))]}_{\approx \ \mathrm{const}(d^{\mathrm{off}}, \phi) + J_{\mathrm{DE}}(q, \phi)}} \\ \textit{Downstream Imitation} & + C_3 \cdot \sqrt{\frac{1}{2} \underbrace{\mathbb{E}_{s\sim d^{\pi^*}}[D_{\mathrm{KL}}(\pi_{*,Z}(s) \| \pi_Z(s))]}_{= \ \mathrm{const}(\pi^*, \phi) + J_{\mathrm{BC},\phi}(\pi_Z)}}, \end{cases}$$

where $C_1 = \gamma |A| (1-\gamma)^{-1} (1 + D_{\chi^2}(d^{\pi^*} \| d^{\mathrm{off}})^{\frac{1}{2}})$, $C_2 = \gamma (1-\gamma)^{-1} (1 + D_{\chi^2}(d^{\pi^*} \| d^{\mathrm{off}})^{\frac{1}{2}})$, $C_3 = \gamma (1-\gamma)^{-1}$, π_{α^*} is the optimal action decoder for a specific data distribution d^{off} and a specific ϕ:

$$\pi_{\alpha^*}(a|s,z) = \frac{d^{\mathrm{off}}(s,a) \cdot \mathbb{1}[z = \phi(s,a)]}{\sum_{a' \in A} d^{\mathrm{off}}(s,a') \cdot \mathbb{1}[z = \phi(s,a')]},$$

and $\pi_{*,Z}$ is the marginalization of π^* onto Z according to ϕ:

$$\pi_{*,Z}(z|s) := \sum_{a \in A, z=\phi(s,a)} \pi^*(a|s).$$

Theorem 5 essentially decomposes the imitation learning error into (4.4) a transition-based representation error J_{T}, (4.5) an action decoding error J_{DE}, and (4.6) a latent behavioral cloning error $J_{\mathrm{BC},\phi}$. Notice that only (4.6) requires expert data $\mathcal{D}^*_{\mathrm{off}}$; (4.4) and (4.5) are trained on the large offline data $\mathcal{D}_{\mathrm{off}}$. By choosing $|Z|$ that is smaller than $|A|$, fewer demonstrations are needed to achieve small error in $J_{\mathrm{BC},\phi}$ compared to vanilla BC with J_{BC}. The Pearson χ^2 divergence term $D_{\chi^2}(d^{\pi^*} \| d^{\mathrm{off}})$ in C_1 and C_2 accounts for the difference in state visitation between the expert and offline data. In the case where d^{π^*} differs too much from d^{off}, known as the distribution shift problem in offline RL [76], the errors from J_{T} and J_{DE} are amplified and the terms (4.4) and (4.5) in Theorem 5 dominate. Otherwise, as $J_{\mathrm{T}} \to 0$ and $q, \phi \to \operatorname{argmin} J_{\mathrm{DE}}$, optimizing π_Z in the latent action space is guaranteed to optimize π in the original action space.

Sample Complexity To formalize the intuition that a smaller latent action space $|Z| < |A|$ leads to more sample efficient downstream behavioral cloning, we provide the following theorem in the tabular action setting. First, assume access to an oracle latent action representation function $\phi_{orcl} := \mathcal{OPT}_\phi(\mathcal{D}_{\mathrm{off}})$ which yields pretraining errors $(4.4)(\phi_{orcl})$ and $(4.5)(\phi_{orcl})$ in Theorem 5. For downstream behavioral cloning, we consider learning a tabular π_Z on $\mathcal{D}^*_{\mathrm{off}}$ with n expert samples. We can bound the expected difference between a latent policy $\pi_{\phi_{orcl},Z}$ with respect to ϕ_{orcl} and π^* as follows.

Theorem 6. *Let $\phi_{orcl} := \mathcal{OPT}_\phi(\mathcal{D}_{off})$ and $\pi_{orcl,Z}$ be the latent BC policy with respect to ϕ_{orcl}. We have,*

$$\mathbb{E}_{\mathcal{D}^*_{off}}[\mathrm{Diff}(\pi_{\phi_{orcl},Z}, \pi^*)] \leq (4.4)(\phi_{orcl}) + (4.5)(\phi_{orcl}) + C_3 \cdot \sqrt{\frac{|Z||S|}{n}},$$

where C_3 is the same as in Theorem 5.

We can contrast this bound to its form in the vanilla BC setting, for which $|Z| = |A|$ and both $(4.4)(\phi_{orcl})$ and $(4.5)(\phi_{orcl})$ are zero. We can expect an improvement in sample complexity from reparametrized actions when the errors in (4.4) and (4.5) are small and $|Z| < |A|$.

Linear Transition Models with Deterministic Latent Policy Theorem 5 has introduced the notion of a latent expert policy $\pi_{*,Z}$, and minimizes the KL divergence between $\pi_{*,Z}$ and a *tabular* latent policy π_Z. However, it is not immediately clear, in the case of continuous latent actions, how to ensure that the latent policy π_Z is expressive enough to capture any $\pi_{*,Z}$. In this section, we provide guarantees for recovering stochastic expert policies with continuous latent action space under a linear transition model.

Consider a *continuous* latent space $Z \subset \mathbb{R}^d$ and a *deterministic* latent policy $\pi_\theta(s) = \theta_s$ for some $\theta \in \mathbb{R}^{d \times |S|}$. While a deterministic θ in general cannot capture a stochastic π^*, we show that under a linear transition model $\mathcal{T}_Z(s'|s, \phi(s,a)) = w(s')^\top \phi(s,a)$, there always exists a deterministic policy $\pi_\theta : S \to \mathbb{R}^d$, such that $\theta_s = \pi_{*,Z}(s)$, $\forall s \in S$. This means that our scheme for offline pretraining paired with downstream imitation learning can *provably* recover any expert policy π^* from a deterministic π_θ, regardless of whether π^* is stochastic.

Theorem 7. *Let $\phi : S \times A \to Z$ for some $Z \subset \mathbb{R}^d$ and suppose there exist $w : S \to \mathbb{R}^d$ such that $\mathcal{T}_Z(s'|s, \phi(s,a)) = w(s')^\top \phi(s,a)$ for all $s, s' \in S, a \in A$. Let $q : S \times Z \to \Delta(A)$ be an action decoder, $\pi : S \to \Delta(A)$ be any policy in $\mathcal{M}$ and $\pi_\theta : S \to \mathbb{R}^d$ be a deterministic latent policy for some $\theta \in \mathbb{R}^{d \times |S|}$. Then,*

$$\text{Diff}(q \circ \pi_\theta, \pi^*) \leq (4.4)(\mathcal{T}_Z, \phi) + (4.5)(q, \phi)$$

$$\left.\begin{matrix} \textit{Downstream} \\ \textit{Imitation} \end{matrix}\right\} + C_4 \cdot \left\| \frac{\partial}{\partial \theta} \mathbb{E}_{s \sim d^{\pi^*}, a \sim \pi^*(s)} [(\theta_s - \phi(s,a))^2] \right\|_1, \tag{4.7}$$

where $C_4 = \frac{1}{4}|S|\|w\|_\infty$, (4.4) and (4.5) corresponds to the first and second terms in the bound in Theorem 5.

By replacing term (4.6) in Theorem 5 that corresponds to behavioral cloning in the latent action space by term (4.7) in Theorem 7 that is a convex function unbounded in all directions, we are guaranteed that π_θ is provably optimal regardless of the form of π^* and $\pi_{*,Z}$. Note that the downstream imitation learning objective implied by term (4.7) is simply the mean squared error between actions θ_s chosen by π_θ and reparameterized actions $\phi(s,a)$ appearing in the expert dataset.

4.2.5 TRAIL: Reparametrized Actions and Imitation Learning in Practice

In this section, we describe our learning framework, Transition-Reparametrized Actions for Imitation Learning (TRAIL). TRAIL consists of two training stages: pretraining and downstream behavioral cloning. During pretraining, TRAIL learns $\mathcal{T}_Z$ and ϕ by minimizing $J_{\text{T}}(\mathcal{T}_Z, \phi) = \mathbb{E}_{(s,a) \sim d^{\text{off}}} [D_{\text{KL}}(\mathcal{T}(s,a) \| \mathcal{T}_Z(s, \phi(s,a)))]$. Also during pretraining, TRAIL learns q and ϕ by minimizing $J_{\text{DE}}(q, \phi) := \mathbb{E}_{(s,a) \sim d^{\text{off}}}[-\log q(a|s, \phi(s,a))]$. TRAIL parametrizes q as a multivariate Gaussian distribution. Depending on whether $\mathcal{T}_Z$ is defined according to Theorem 5 or Theorem 7, we have either TRAIL EBM or TRAIL linear.

TRAIL EBM for Theorem 5. In the tabular action setting that corresponds to Theorem 5, to ensure that the factored transition model $\mathcal{T}_Z$ is flexible to capture any complex (e.g., multi-modal) transitions in the offline dataset, we propose to use an energy-based model (EBM) to parametrize $\mathcal{T}_Z(s'|s, \phi(s,a))$,

$$\mathcal{T}_Z(s'|s, \phi(s,a)) \propto \rho(s') \exp(-\|\phi(s,a) - \psi(s')\|^2), \tag{4.8}$$

where ρ is a fixed distribution over S and $\psi : S \to Z$ is a function of s'. In our implementation we set ρ to be the distribution of s' in d^{off}, which enables a practical learning objective for $\mathcal{T}_Z$ by minimizing $\mathbb{E}_{(s,a)\sim d^{\text{off}}}[D_{\text{KL}}(\mathcal{T}(s,a)\|\mathcal{T}_Z(s,\phi(s,a)))]$ in Theorem 5 using a contrastive loss:

$$\mathbb{E}_{d^{\text{off}}}[-\log \mathcal{T}_Z(s'|s,\phi(s,a)))] = \text{const}(d^{\text{off}}) + \frac{1}{2}\mathbb{E}_{d^{\text{off}}}[||\phi(s,a) - \psi(s')||^2] + \log \mathbb{E}_{\tilde{s}'\sim\rho}[\exp\{-\frac{1}{2}||\phi(s,a) - \psi(\tilde{s}')||^2\}].$$

During downstream behavioral cloning, TRAIL EBM learns a latent Gaussian policy π_Z by minimizing $J_{\text{BC},\phi}(\pi_Z) = \mathbb{E}_{(s,a)\sim(d^{\pi^*},\pi^*)}[-\log \pi_Z(\phi(s,a)|s)]$ with ϕ fixed. During inference, TRAIL EBM first samples a latent action according to $z \sim \pi_Z(s)$, and decodes the latent action using $a \sim q(s,z)$ to act in an environment. Figure 4.5 describes this process pictorially.

TRAIL Linear for Theorem 7. In the continuous latent action setting that corresponds to Theorem 7, we propose TRAIL linear, an approximation of TRAIL EBM, to enable learning *linear* transition models required by Theorem 7. Specifically, we first learn f, g that parameterize an energy-based transition model $\overline{\mathcal{T}}(s'|s,a) \propto \rho(s')\exp\{-||f(s,a) - g(s')||^2/2\}$ using the same contrastive loss as above (replacing ϕ and ψ by f and g), and then apply random Fourier features [279] to recover $\bar{\phi}(s,a) = \cos(Wf(s,a) + b)$, where W is a $d \times k$ matrix with entries sampled from a unit Gaussian and b a vector with entries sampled uniformly from $[0, 2\pi]$. W and b are implemented as an untrainable neural network layer on top of f. This results in an approximate linear transition model,

$$\overline{\mathcal{T}}(s'|s,a) \propto \rho(s')\exp\{-||f(s,a) - g(s')||^2/2\} \propto \bar{\psi}(s')^\top \bar{\phi}(s,a).$$

During downstream behavioral cloning, TRAIL linear learns a deterministic policy π_θ in the continuous latent action space determined by $\bar{\phi}$ via minimizing $\left\|\frac{\partial}{\partial\theta}\mathbb{E}_{s\sim d^{\pi^*},a\sim\pi^*(s)}[(\theta_s - \bar{\phi}(s,a))^2]\right\|_1$ with $\bar{\phi}$ fixed. During inference, TRAIL linear first determines the latent action according to $z = \pi_\theta(s)$, and decodes the latent action using $a \sim q(s,z)$ to act in an environment.

4.2.6 Experimental Evaluation

We now evaluate TRAIL on a set of navigation and locomotion tasks (Figure 4.6). Our evaluation is designed to study how well TRAIL can improve imitation learning with limited expert data by leveraging available suboptimal offline data. We evaluate the improvement attained by TRAIL over vanilla BC, and additionally compare TRAIL to previously proposed temporal skill extraction methods. Since there is no existing benchmark for imitation learning with suboptimal offline data, we adapt existing datasets for offline RL, which contain suboptimal data, and augment them with a small amount of expert data for downstream imitation learning.

Evaluating Navigation without Temporal Abstraction

Description and Baselines. We start our evaluation on the AntMaze task from D4RL [10], which has been used as a testbed by recent works on temporal skill extraction for few-shot imitation [14] and RL [14, 13, 12]. We compare TRAIL to OPAL [14], SkilD [12], and SPiRL [13], all of which use an offline dataset to extract temporally extended (length $t = 10$) skills to form a latent action space for downstream learning. SkiLD and SPiRL are originally designed only for downstream RL, so we modify them to support downstream imitation learning as described in Appendix A.2.7. While a number of other works have also proposed to learn primitives for hierarchical imitation [290, 278] and RL [292, 293, 295, 294, 107], we chose OPAL, SkiLD, and SPiRL for comparison because they are the most recent works in this area with reported results that suggest these methods are state-of-the-art, especially in learning from *suboptimal* offline data based on D4RL. To construct the suboptimal and expert datasets, we follow the protocol in [14], which uses the full `diverse` or `play` D4RL AntMaze datasets as the suboptimal offline data, while using a set of $n = 10$ expert trajectories (navigating from one corner of the maze to the opposite corner) as the expert data. The `diverse` and `play` datasets are suboptimal in the corner-to-corner navigation task, as they only contain data that navigates to random or fixed locations different from task evaluation.

Implementation Details. For TRAIL, we parameterize $\phi(s, a)$ and $\psi(s')$ using separate feed-forward neural networks (see details in Appendix A.2.7) and train the transition EBM via the contrastive objective described in Section 4.2.5. We parametrize both the action decoder q and the latent π_Z using multivariate Gaussian distributions with neural-network approximated mean and variance. For the temporal skill extraction methods, we implement the trajectory encoder using a bidirectional RNN and parametrize skill prior, latent policy, and action decoder as Gaussians following [14]. We adapt SPiRL and SkiLD for imitation learning by including the KL Divergence term between the latent policy and the skill prior during downstream behavioral cloning (see details in Appendix A.2.7). We do a search on the extend of temporal abstraction, and found $t = 10$ to work the best as reported in these papers' maze experiments. We also experimented with a version of vanilla BC pretrained on the suboptimal data and fine-tuned on expert data for fair comparison, which did not show a significant difference from directly training vanilla BC on expert data.

Figure 4.6: Tasks for our empirical evaluation. We include the challenging AntMaze navigation tasks from D4RL [10] and low (1-DoF) to high (21-DoF) dimensional locomotaion tasks from DeepMind Control Suite [11].

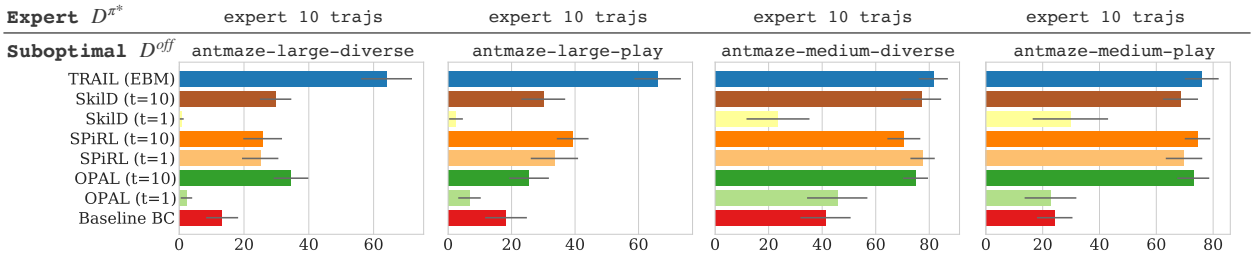


Figure 4.7: Average success rate (%) over 4 seeds of TRAIL EBM (Theorem 5) and temporal skill extraction methods – SkiLD [12], SPiRL [13], and OPAL [14] – pretrained on suboptimal $\mathcal{D}_{\text{off}}$. Baseline BC corresponds to direct behavioral cloning of expert $\mathcal{D}^*_{\text{off}}$ without latent actions.

Results. Figure 4.7 shows the average performance of TRAIL in terms of task success rate (out of 100%) compared to the prior methods. Since all of the prior methods are proposed in terms of temporal abstraction, we evaluate them both with the default temporal abstract, $t = 10$, as well as without temporal abstraction, corresponding to $t = 1$. Note that TRAIL uses *no* temporal abstraction. We find that on the simpler `antmaze-medium` task, TRAIL trained on a single-step transition model performs similarly to the set of temporal skill extraction methods with $t = 10$. However, these skill extraction methods experience a degradation in performance when temporal abstraction is removed ($t = 1$). This corroborates the existing theory in these works [14], which attributes their benefits predominantly to temporal abstraction rather than producing a latent action space that is "easier" to learn. Meanwhile, TRAIL is able to excel without any temporal abstraction.

These differences become even more pronounced on the harder `antmaze-large` tasks. We see that TRAIL maintains significant improvements over vanilla BC, whereas temporal skill extraction fails to achieve good performance even with $t = 10$. These results suggest that TRAIL attains significant improvement specifically from utilizing the suboptimal data for learning suitable action representations, rather than simply from providing temporal abstraction. Of course, this does not mean that temporal abstraction is never helpful. Rather, our results serve as evidence that suboptimal data can be useful for imitation learning not just by providing temporally extended skills, but by actually reformulating the action space to make imitation learning easier and more efficient.

Evaluating Locomotion with Highly Suboptimal Offline Data

Description. The performance of TRAIL trained on a *single-step* transition model in the previous section suggests that learning single-step latent action representations can benefit a broader set of tasks for which temporal abstraction may not be helpful, e.g., when the offline data is highly suboptimal (with near-random actions) or unimodal (collected by a single

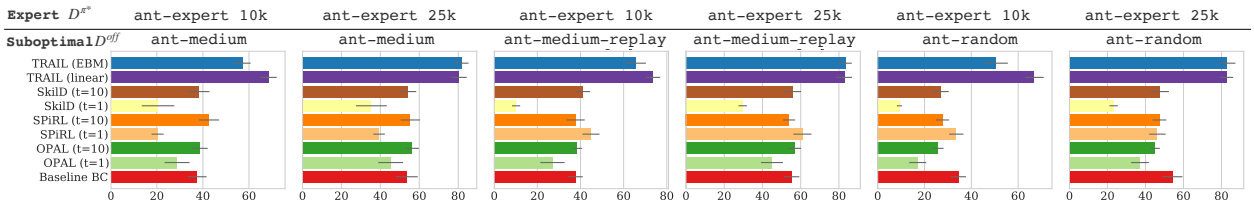


Figure 4.8: Average rewards (over 4 seeds) of TRAIL EBM (Theorem 5), TRAIL linear (Theorem 7), and baseline methods when using a variety of unimodal (`ant-medium`), low-quality (`ant-medium-replay`), and random (`ant-random`) offline datasets $\mathcal{D}_{\text{off}}$ paired with a smaller expert dataset $\mathcal{D}^*_{\text{off}}$ (either 10k or 25k expert transitions).

stationary policy). In this section, we consider a Gym-MuJoCo task from D4RL using the same 8-DoF quadruped ant robot as the previously evaluated navigation task. We first learn action representations from the `medium`, `medium-replay`, or `random` datasets, and imitate from 1% or 2.5% of the `expert` datasets from D4RL. The `medium` dataset represents data collected from a mediocre stationary policy (exhibiting unimodal behavior), and the `random` dataset is collected by a randomly initialized policy and is hence highly suboptimal.

Implementation Details. For this task, we additionally train a linear version of TRAIL by approximating the transition EBM using random Fourier features [279] and learn a *deterministic* latent policy following Theorem 7. Specifically, we use separate feed-forward networks to parameterize $f(s, a)$ and $g(s')$, and extract action representations using $\phi(s, a) = \cos(Wf(s, a) + b)$, where W, b are untrainable randomly initialized variables as described in Section 4.2.5. Different from TRAIL EBM which parametrizes π_Z as a Gaussian, TRAIL linear parametrizes the *deterministic* π_θ using a feed-forward neural network.

Results. Our results are shown in Figure 4.8. Both the EBM and linear versions of TRAIL consistently improve over baseline BC, whereas temporal skill extraction methods generally lead to worse performance regardless of the extent of abstraction, likely due to the degenerate effect (i.e., latent skills being ignored by a flexible action decoder) resulted from unimodal offline datasets as discussed in [14]. Surprisingly, TRAIL achieves a significant performance boost even when latent actions are learned from the `random` dataset, suggesting the benefit of learning action representations from transition models when the offline data is highly suboptimal. Additionally, the linear variant of TRAIL performs slightly better than the EBM variant when the expert sample size is small (i.e., 10k), suggesting the benefit of learning deterministic latent policies from Theorem 7 when the environment is effectively approximated by a linear transition model.

Evaluation on DeepMind Control Suite

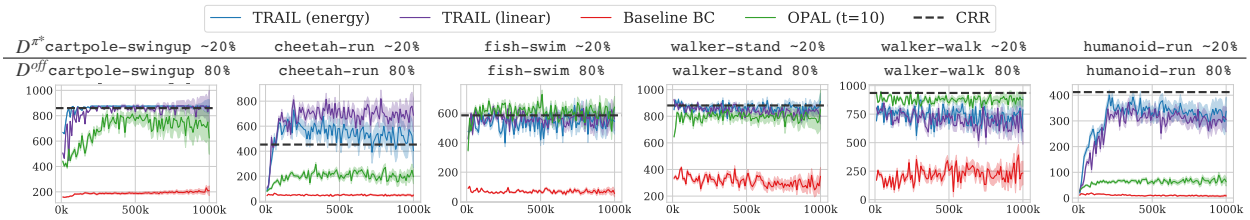


Figure 4.9: Average task rewards (over 4 seeds) of TRAIL EBM (Theorem 5), TRAIL linear (Theorem 7), and OPAL (other temporal methods are included in Appendix A.2.8) pretrained on the bottom 80% of the RL Unplugged datasets followed by behavioral cloning in the latent action space on $\frac{1}{10}$ of the top 20% of the RL Unplugged datasets following the setup in [15]. Baseline BC achieves low rewards due to the small expert sample size. Dotted lines denote the performance of CRR [16], an offline RL method trained on the full RL Unplugged datasets with reward labels.

Description. Having witnessed the improvement TRAIL brings to behavioral cloning on AntMaze and MuJoCo Ant, we wonder how TRAIL perform on a wider spectrum of locomotion tasks with various degrees of freedom. We consider 6 locomotion tasks from the DeepMind Control Suite [11] ranging from simple (e.g., 1-DoF `cartople-swingup`) to complex (e.g., 21-DoF `humanoid-run`) tasks. Following the setup in [15], we take $\frac{1}{10}$ of the trajectories whose episodic reward is among the top 20% of the open source RL Unplugged datasets [20] as expert demonstrations, and the bottom 80% of RL Unplugged as the suboptimal offline data. For completeness, we additionally include comparison to Critic Regularized Regression (CRR) [16], an offline RL method with competitive performance on these tasks. CRR is trained on the full RL Unplugged datasets (i.e., combined suboptimal and expert datasets) with reward labels.

Results. Figure 4.9 shows the comparison results. TRAIL outperforms temporal extraction methods on both low-dimensional (e.g., `cartpole-swingup`) and high-dimensional (`humanoid-run`) tasks. Additionally, TRAIL performs similarly to or better than CRR on 4 out of the 6 tasks despite not using any reward labels, and only slightly worse on `humanoid-run` and `walker-walk`. To test the robustness of TRAIL when the offline data is highly suboptimal, we further reduce the size and quality of the offline data to the bottom 5% of the original RL Unplugged datasets. As shown in Figure A.13 in Appndix A.2.8, the performance of temporal skill extraction declines in `fish-swim`, `walker-stand`, and `walker-walk` due to this change in offline data quality, whereas TRAIL maintains the same performance as when the bottom 80% data was used, suggesting that TRAIL is more robust to low-quality offline data.

This set of results suggests a promising direction for offline learning of sequential decision making policies, namely to learn latent actions from abundant low-quality data and behavioral cloning in the latent action space on scarce high-quality data. Notably, compared

to offline RL, this approach is applicable to settings where data quality cannot be easily expressed through a scalar reward.

Table 4.2: Factors of contrastive self-prediction considered in our ablation study and summaries of their effects. For each effect entry, ↓ means decreased performance, ↑ means improved performance, and = means no significant effect.

Factor	**Description**	**Imitation**	**Offline**	**Online**
reconstruct action	Add action prediction loss based on $\phi(s)$.	↓	↑	↑
reconstruct reward	Add a reward prediction loss based on $\phi(s)$.	↓	↑	↑
predict action	Add an action prediction loss based on transformer outputs. Whenever this is true, we also set 'input embed' to true.	↓	↑	↑
predict reward	Add a reward prediction loss based on transformer outputs. Whenever this is true, we also set 'input embed' to true.	↓	↑	↑
input action	Include actions in the input sequence to transformer.	↓	↑	↑
input reward	Include rewards in the input sequence to transformer.	↓	↑	↑
input embed	Use representations $\phi(s)$ as input to transformer, as opposed to raw observations.	↓	=	↑
bidirectional	To generate sequence output at position i, use full input sequence as opposed to only inputs at position $> i$.	↓	=	↑
finetune	Pass gradients into ϕ during learning on downstream tasks.	↓	↓	↑
auxiliary loss	Use representation learning objective as an auxiliary loss during downstream learning, as opposed to pretraining.	↓	↓	↑
momentum	Adopt an additional momentum representation network. Whenever this is true, we also set 'input embed' to true.	↓	↓	↑
discrete embedding	Learn discrete representations. Following [118], we treat the 256-dim output of ϕ as logits to sample 16 categorical distributions of dimension 16 each and use straight-through gradients.	↓	↓	↓
context embedding	Following [1], use transformer output as representations for downstream tasks. Whenever this is true, we also set 'input embed' to true.	↓	↓	↓

Chapter 5

Generative Modeling as an Alternative to Offline RL

In the previous chapters, we have seen that offline interactive data can be used to train RL policies (Chapter 3) or learn representations of states and actions (Chapter 4). Both approaches make assumptions about data coverage, and have been difficult to scale up to multiple environments/tasks. Therefore, it is desirable to further explore alternatives of leverage offline interactive data.

In language modeling, generative pretrained transformers (GPT) has shown that next-token prediction can be a powerful pretraining objective that is aligned with downstream task specific finetuning. Generative pretraining in language has an advantage — language is a unified representation of information while language generation is a unified task interface. For instance, broad data with diverse information such as code, Wikipedia articles, and math equations can all be represented as text. Meanwhile, many tasks such as translation, summarization, and sentiment classification can all be cast into a language generation task. As a result, we can pour internet-scale language data into a single model trained using a single objective. This approach scales favourably with dataset and model size. It is therefore interesting to explore whether generative pretraining objectives can apply to sequential decision making settings.

In this chapter, we explore next-token prediction in sequential decision making settings. Specifically, Section 5.1 proposes incorporating problem solving as a part of the next-token prediction process during imitation learning. In another word, instead of directly predicting actions from states, this section explores predicting intermediate thoughts and auxiliary information that leads to the optimal action before predicting the action itself. Procedure cloning generalizes significantly better than traditional imitation learning. In Section 5.2, we focus on studying a family of methods named Return-Conditioned Supervised Learning (RCSL), which treats the mapping from states to action as a next-token prediction problem, and allows flexible incorporation of additional conditioning information such as the desired returns to be achieved. Overall, this chapter shows, both empirically and theoretically, that conditional generation can be an effective alternative to offline RL and using offline data for

representation learning.

5.1 Chain-of-Thought Imitation with Procedure Cloning

5.1.1 Introduction

The idea of learning by imitation in autonomous agents closely resembles how humans (especially children) learn in real life — by watching and mimicking how someone else performs a certain task [268]. While humans are able to generalize exceedingly well from a small number of demonstrations, today's imitation-learned autonomous agents often struggle in situations that only slightly differ from the demonstrations, e.g., opening a door in a different shape or color [165]. One explanation for such a difference is that while humans imitate, we *understand* the task at a high level as opposed to only remembering a mapping from images to actions [305], and indeed, generalization failures can also occur in humans when there is a lack of understanding of the underlying reasons for the behavior, such as solving a complex math problem. As a result, students are told to "learn principles, not formulas" and "understand, do not memorize" to encourage better generalization — to be able to solve problems that are similar but different from the ones taught in lectures [306, 307, 308]. Just as students are taught the step-by-step derivations of a math problem during a lecture, is it possible to have an equivalent "chain of thought" supervision for training autonomous agents?

While solving complex math derivations may not be the predominant application of today's imitation-learned autonomous agents, the tasks they are commonly applied to — e.g., path navigation, robot manipulation, and strategy games — often *do* employ chain-of-thought reasoning procedures in the form of planning, search, or other multi-step algorithms when collecting expert data [309, 310, 21]. Since such multi-step algorithms are application-specific and therefore lack a common representation which can be systematically characterized, it is typical for imitation learning to assume access to only logged demonstrations of state-action pairs, leaving out the much richer insight into expert behavior provided by the algorithm's intermediate computations. In addition, the planning or search procedures used by the expert may rely on tools not available to the agent during inference (e.g., environment simulators), so how these intermediate computations should be used to facilitate imitation learning is not clear.

In this work, we formulate *chain of thought imitation* as an extension of traditional imitation learning. Different from the traditional imitation learning setup where an agent only has access to expert state-action pairs as demonstrations, chain of thought imitation also has access to the intermediate computations that generated the expert state-action pairs in the training data (Figure 5.1). We then propose *procedure cloning* (PC), an alternative to behavioral cloning (BC) [62], which applies supervised sequence prediction to imitate the complete series of of expert computations before outputting an expert action (Figure 5.1). Procedure cloning learns a policy by maximizing the likelihood of the joint distribution of procedure observations and expert actions, which can be modeled autoregressively using a transformer-

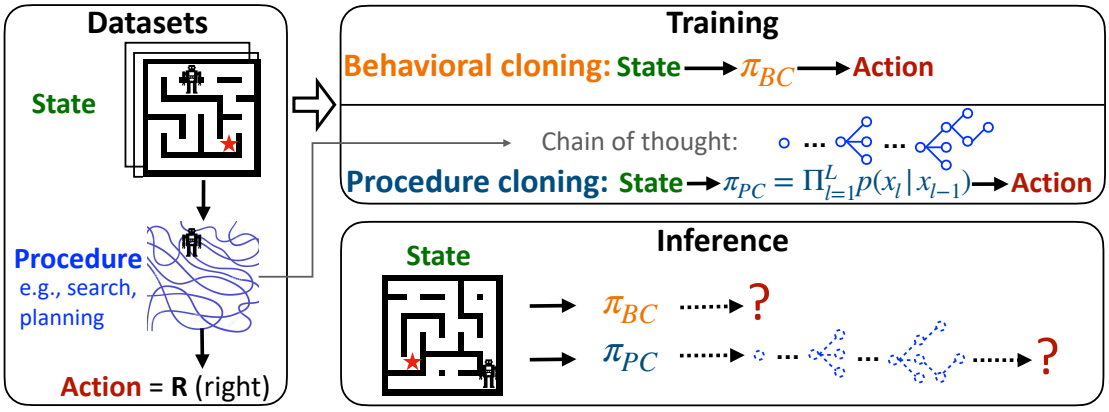


Figure 5.1: Visualization of the dataset collection, training, and inference of BC and PC on a maze navigation task. During dataset collection, the expert uses a search procedure to determine the optimal action to generate a path to the goal location (red star). During training, BC discards these intermediate search outputs and learns to map states to actions directly. In contrast, PC learns the complete sequence of intermediate computations (i.e., branches and backtracks) associated with the search procedure. During inference, PC generates a sequence of intermediate search outcomes emulating the search procedure on a new test map before outputting the final action.

like architecture [94]. During inference, a procedure cloned policy autoregressively generates procedure observations from a given input state, mimicking the computations of a search or planning algorithm before outputting the final action, thus avoiding any reliance on privileged tools or information used by the expert's procedure. From a modeling perspective, procedure cloning can employ a more expressive model (e.g., transformer) trained on more data (i.e., procedure observations), which leads to better generalization according to the new "scaling law" of large language models [311]. Intuitively, procedure cloning learns not only what to do (i.e., the output action), but how and why to do it (i.e., the procedure), which further resembles how humans learn complex tasks.

We demonstrate how to conduct procedure cloning and leverage the generalization ability of procedure cloned agents on a variety of path navigation, robotic manipulation, and strategy game tasks. For example, in path navigation an expert trajectory may be determined by using a BFS search algorithm on an annotated map (e.g., x, y coordinates of obstacles), which are expensive to obtain [312]. We show that a procedure cloning agent can successfully learn to imitate BFS on a previously unseen test map without requiring additional annotations, achieving 100% test accuracy while a BC agent completely fails to navigate (0% accuracy) when the maze layout changes. Similarly, in an image-based robotic manipulation task, we observe that BC quickly overfits to the set of training images, whereas procedure cloning learns to predict the intermediate computation outcomes of a scripting policy, thus generalizing much better and achieving a success metric of 83.9% compared to 78.2% from

the previous state-of-the-art [17]. Finally, in strategy games expert trajectories are collected by running MCTS [23], which requires access to a simulator and can be extremely slow [313]. We show that procedure cloning can effectively learn from the path traversed by MCTS collected in a deterministic environment to then successfully generalize in a zero-shot manner to stochastic environments and more difficult game settings, where running MCTS, even if given access to a simulator, performs poorly.

Related Work

Generalization in sequential decision making. Learning decision making policies with good generalization properties has a long-standing history in bandit [314, 315, 316], imitation learning [302], and reinforcement learning (RL) [317, 318, 319, 320] settings in the form of regret or Probably Approximately Correct (PAC) bound analysis. These studies are concerned with generalization in a continual learning setup without explicit separation of training and testing stages. We are more interested in an agent's ability to generalize to a separate "test" environment whose configuration is different from the training environment, which is often what happens when an agent is deployed to the real world after being trained in a simulator. With high-capacity neural network parametrized policies being the norm in training image-based agents, the risk of overfitting to the training environment is nontrivial [321, 322]. Various works have injected stochasticity into the training process including using stochastic policies [323], random starts [324, 325], sticky actions [326], and frame skipping [327] as a means to prevent overfitting to the specific environment dynamics experienced by the agent during training. A variety of image-based regularization techniques have also been applied to deep RL including dropout and l-2 regularization [328], data augmentation and batch normalization [329], domain randomization [114], and network randomization [330]. Instead of improving generalization through regularization or data augmentation like in existing work, we instead ask the question of whether learning the sequence of computations as opposed to only the final expert action during training can help an agent generalize better.

Access to additional task information Many previous works in imitation learning and RL have observed that access to additional task information can help an agent learn better policies. For instance, [303] relies on access to the simulator for collecting more expert trajectories to reduce the quadratic error in task horizon to linear. [331] assumes that expert demonstrations contain both high-level and low-level trajectories, and learns a hierarchical policy explicitly. While our procedure observations can appear to be high-level labels or subgoals similar to [331, 332], we do not make assumptions about the structure of procedure observations, which can simply be scalar variable values computed during program execution. There is also a large body of literature that assumes access to a suboptimal offline dataset in addition to the expert demonstration data collected from the same environment, and conduct representation learning [299, 179, 182, 7, 333], hierarchical skill extraction [290, 14, 278], or dynamics model learning [334, 300, 301] on the suboptimal offline data followed by imitation

learning from an expert. These works commonly assume good coverage in the offline data, which requires large amounts of state-action pairs being collected from running additional policies in the environment. We instead propose observing additional information from only running the expert policy, which requires far fewer state-action pairs being collected for training. Another line of existing work in the direction of multi-task learning suggests that learning an auxiliary task enhances imitation learning or RL [335, 336, 165, 337, 165, 338, 339]. For instance, [340] showed that predicting internal features of the emulator (e.g., enemy on the screen) is beneficial, [341] found predicting scene depth helps navigation, and [342] found predicting explanations helps relational tasks with causal structure. Chain of thought imitation differs from multi-task learning in that procedures directly influence the action outcomes through computations, in which case learning the procedure directly (as opposed to treating it as an auxiliary objective similar to existing work) is more beneficial. The graphical model in Figure 5.2 visualizes this distinction.

Chain of thought sequence modeling. The idea of decomposing multi-step problems into intermediate steps (the so-called chain of thought [43]) and learning the intermediate steps using a sequence model has been applied to *domain specific* problems such as program induction [343], learning to solve math problems [57], learning to execute [203], learning to reason [344, 345, 346, 347, 348, 349], and language model prompting [43]. The chain of thought imitation learning problem we formulate is *domain agnostic* and applicable to many sequential decision making task traditionally solved by imitation learning in a Markovian setting such as robot locomotion, navigation, manipulation, and strategy games. Unlike language-based tasks, problems in decision making have only recently started being explored by language models [350, 60, 5, 351, 352, 150], as the Markovian nature of these problems brings the value of sequence modeling into question. Our work contributes to bridging the gap between learning memoryless policies in Markovian environments and the intuition that large sequence models should help in reasoning-based decision making.

5.1.2 Procedure Cloning

In this section, we first observe that in many imitation learning situations, expert demonstrations can provide much richer insight into the desirable behavior than just the final optimal action. We formulate learning under these situations as *chain of thought imitation*, where an agent can learn from not just the optimal action but the "thought process" an expert goes through before arriving at the final decision. We then formalize such thought process as *procedures*, and propose *procedure cloning* for learning such thought process through supervised sequence prediction.

Chain of thought imitation Depending on the form of the expert policy π^* used to generate the training data $\mathcal{D}^*_{\text{RL}}$, an agent potentially has access to a rich set of information about a task which can facilitate learning. For instance, when π^* is some scripting policy

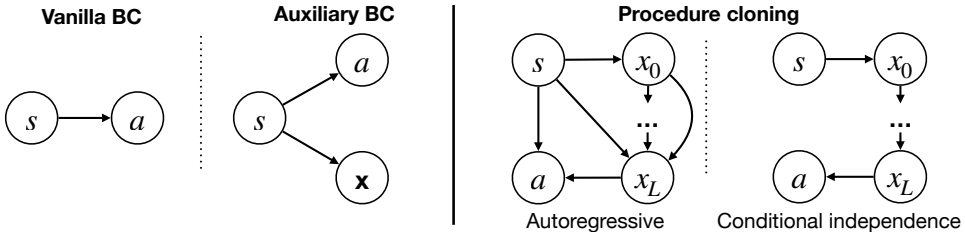


Figure 5.2: Graphical models of vanilla BC, auxiliary BC, and procedure cloning with autoregressive and conditionally independent factorization. Node s represents an input MDP state, a represents an expert action, and $\mathbf{x}$ represents the sequence of procedure observations $(x_0, ..., x_L)$.

following a fixed set of rules [353], the rules (e.g., "if close to enemy then fire") reveal causal information between firing and the enemy disappearing. In other settings when π^* is a search algorithm, the induced search tree exposes the path for finding the optimal action. When π^* is a multi-step hand-coded algorithm, the breakdown of the steps (e.g., "first move to objects then sweep") reveals important ordering information about the task. Even in the case where π^* is a human demonstrator, we can ask the human to explain the thought process that led to their decision. We refer to learning from the procedures (e.g., planning, search, multi-step algorithm) that generated the final action as *chain of thought imitation*.

Procedures and procedure observations To formulate chain of thought imitation as a learning problem, we define a procedure $\mathbf{\Pi} : S \to \Delta(A)$ as a sequence of computations $(\Pi_0, \Pi_1, ..., \Pi_L, \Pi_{L+1})$ that first transform an input state $s \in S$ into some computation state (e.g., a variable value) using $\Pi_0 : S \to \mathbb{R}^d$, followed by repeatedly applying each subprocedure, $\Pi_\ell : \mathbb{R}^d \to \mathbb{R}^d, \forall \ell \in [1, L]$, to the computation state before mapping the last computation state back to the MDP action space using $\Pi_{L+1} : \mathbb{R}^d \to \Delta(A)$, in a sequential order. A procedure can be broken down to subprocedures at any user-defined granularity (e.g., a function or a for loop), depending on how frequently the computation state is retrieved for procedure learning. Each pair of training data, $(s, a) \sim \mathcal{D}^*_{\text{RL}}$, is acquired through executing such a procedure, i.e., $\pi^* = \mathbf{\Pi}$.

We define *procedure observations* as the sequence of computation states captured from a procedure: $\mathbf{x} = (x_0, ...x_L) \in \mathbb{R}^{(L+1)\times d}$ where $x_0 = \Pi_0(s)$ and $x_\ell = \Pi_\ell \circ \Pi_{l-1} \circ ... \circ \Pi_1 \circ \Pi_0(s), \forall \ell \in [1, L]$. The training data with procedure observations is denoted as $\mathcal{D}_\Pi = \{(s^{(i)}, \mathbf{x}^{(i)}, a^{(i)})\}_{i=1}^n$. Note that we do not assume any procedure or procedure observation is available at test time, hence we still need to learn a policy π that only takes s as input.

Procedure cloning With the procedure observations defined above, we are now ready for learning procedures through *procedure cloning* (PC). We model a PC policy by estimating the joint distribution of the procedure observations and the final action conditioned on the

input state $p(a, \mathbf{x}|s)$, which we can factorize autoregressively as:

$$p(a, \mathbf{x}|s) = p(a|\mathbf{x}, s) \cdot \Pi_{l=1}^{L} p(x_\ell|\mathbf{x}_{<\ell}, s) \cdot p(x_0|s). \tag{5.1}$$

Under this factorization, estimating $p(a, \mathbf{x}|s)$ reduces to estimating each conditional factor, which can be parametrized using a transformer model [94]. The autoregressive factorization is highly flexible, but if the amount of expert demonstrations is small, and each procedure observation x_ℓ only depends on the previous procedure observation $x_{\ell-1}$ (i.e., the computation states are fully observed), and the final action only depends on the last computation state, a conditionally independent factorization can be more desirable. In other words, autoregressive models need more data to train, so the conditional independence factorization below is preferable if the procedure information available fully captures the computation state:

$$p(a, \mathbf{x}|s) = p(a|x_L) \cdot \Pi_{l=1}^{L} p(x_\ell|x_{\ell-1}) \cdot p(x_0|s). \tag{5.2}$$

The graphical models of the two factorizations of PC policies are shown in Figure 5.2. To learn a PC policy, we maximize the empirical likelihood of the joint distribution in Equation 5.1 on given samples $\mathcal{D}_\Pi = \{(s^{(i)}, \mathbf{x}^{(i)}, a^{(i)})\}_{i=1}^{n}$:

$$\min_{\phi,\theta,\psi} J_{\text{PC}}(\phi, \theta, \psi) = \hat{\mathbb{E}}_{(s,\mathbf{x},a)\sim\mathcal{D}_\Pi}[-\log p(a, \mathbf{x}|s)] \tag{5.3}$$

$$= \hat{\mathbb{E}}_{(s,\mathbf{x},a)\sim\mathcal{D}_\Pi}\left[-\log q_\psi(a|\mathbf{x}, s) - \sum_{\ell=1}^{L} \log p_\theta(x_\ell|\mathbf{x}_{<\ell}, s) - \log p_\phi(x_0|s)\right]. \tag{5.4}$$

Connection to BC with auxiliary tasks. The PC objective in Equation 5.4 can be reduced to the vanilla BC objective in Equation (2.2) by discarding the second and third term inside the expectation of Equation 5.4 and setting $q_\psi(a|\mathbf{x}, s) = \pi(a|s)$. We note that several previous works [336, 335, 165, 337, 340, 341] have used procedure information as auxiliary tasks to BC, which may be interpreted as learning $p(a, \mathbf{x}|s)$ under the assumption that a and $\mathbf{x}$ are independent conditioned on s: $p(a, \mathbf{x}|s) = \pi(a|s) \cdot p(\mathbf{x}|s)$. Procedure cloning instead focuses on situations where such a conditional independence assumption does not hold (i.e., a is directly computed by the procedure represented by $\mathbf{x}$), and in these situations, as we will show in our experiments, treating the procedure information as a precursor to a can perform better than using it as an auxiliary task. The graphical models of vanilla BC and BC with auxiliary task objective are shown in Figure 5.2.

5.1.3 Proof of concept: Synthetic maze navigation

In this section, we study a tabular maze navigation task with synthetically generated maze layouts (see Figure 5.3). We describe the task setup, followed by how to extract procedure data from a breadth-first search (BFS) path planning algorithm to train a procedure cloning agent, and empirically show that procedure cloning indeed generalizes much better to unseen maze layouts than other BC baselines. While this proof of concept illustration might seem

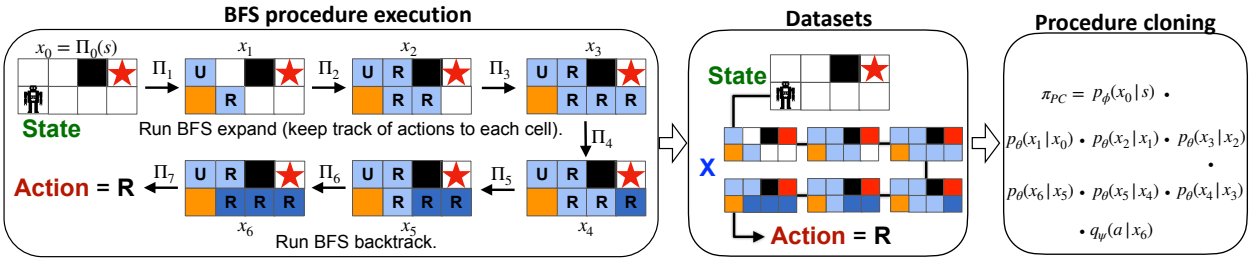


Figure 5.3: In a discrete maze, the expert employs BFS by first expanding a search perimeter until it encounters the goal cell, at which point it backtracks to find the optimal action at the starting state (cells in light blue are visited and dark blue are backtracked). We encode this algorithm as a sequence of procedure observations $(x_0, ..., x_6)$ of the intermediate computation states, with each x_i represented by a 2D array and each cell of the array containing BFS-relevant information (i.e., whether this cell is being expanded or backtracked and the action recorded when expanding to this cell). Procedure cloning is trained to predict the entire sequence of computations from input state to output action using a sequential model $p(a|x_L) \cdot \Pi_{l=1}^{L} p(x_\ell|x_{\ell-1}) \cdot p(x_0|s)$.

domain-specific to BFS in mazes, we will see in Section 5.1.4.3 that procedure cloning can be applied to many types of search or multi-step algorithms.

Task description and evaluation protocol. We use a gridworld maze environment in which an agent seeks to navigate to a goal location in a maze from a random starting location using 4 discrete actions including up (`U`), down (`D`), left (`L`), and right (`R`). The input to the agent is a multi-channel "image" with the maze wall, goal location, and agent location encoded in separate channels. The maze layout is algorithmically generated with random internal walls that form a tunnel-shaped map (see example maze in Figure 5.4. We generate a set of mazes $S_0 \subset S$ and split S_0 into disjoint training S_0^{train} and testing S_0^{test} sets. We then generate expert trajectories by running BFS on only the training set of mazes S_0^{train}. At test time, the agent is evaluated on S_0^{test} by their success rate of navigating to the goal. Figure 5.3 visualizes the data and training pipeline.

Procedure data collection. BFS is a common path planning algorithm for navigation [354, 355, 356, 357], which we use to generate expert trajectories for training an imitation learning agent. To compute the optimal action at each time step, BFS keeps track of a `visited` 2D array (colored cells in Figure 5.3) that marks whether each position (1) has been visited by the search, (2) if so, which action visited it, and (3) has a position been backtracked. We simply take a snapshot of the entire `visited` array as procedure observations x_ℓ every time BFS expands the search perimeter, resulting in a series of procedure data $\mathbf{x} = (x_1, ..., x_L)$ as shown in Figure 5.3. Π_0 is the identity map and $x_0 = s$.

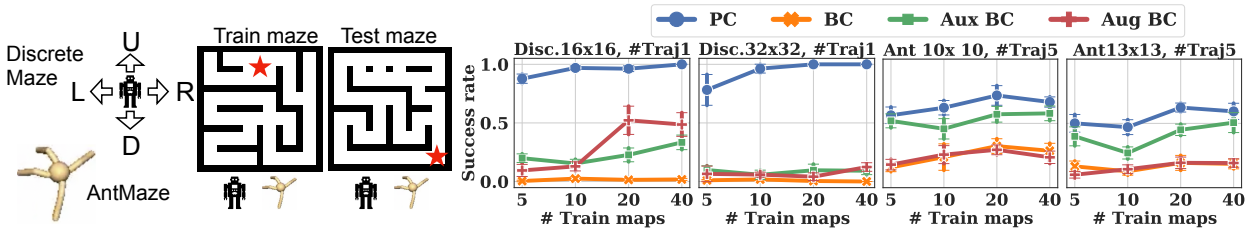


Figure 5.4: [Left] Visualization of the discrete maze (4 discrete actions) and AntMaze (8 continuous actions). [Right] Average success rate of PC and BC agents navigating to the goal from random start locations over 10 test mazes. Agents are trained on 5, 10, 20, 40 mazes of 1 and 5 expert trajectories on discrete maze and AntMaze, respectively. We find that procedure cloning leads to much better test maze generalization compared to alternative approaches.

Procedure learning. Since each of the `visited` 2D arrays x_ℓ only depends on the previous `visited` array $x_{\ell-1}$, and the final `visited` array after backtracking uniquely identifies the expert action on its own, we choose the conditionally independent factorization of $p(a, \mathbf{x}|s)$ (Equation 5.2) described in Section 5.1.2. Specifically, we parametrize $p_\theta(x_\ell|x_{\ell-1}), \forall \ell \in [1, L]$ using a deep convolutional neural network that takes in the current `visited` array $x_{\ell-1}$ as input and produces the next `visited` array x_ℓ as output. We optimize θ using the cross-entropy loss between the predicted and true next `visited` array. Since the procedure observations x_ℓ and the original input s are in the same image space, $\phi(x)$ shares the same parameters as $p_\theta(x_\ell|x_{\ell-1})$. During inference when a new test maze layout s is given, we apply $\hat{x}_0 = \phi(s)$ and $\hat{x}_\ell \sim p_\theta(\cdot|\hat{x}_{\ell-1})$ repeatedly until an array $\hat{x}_L$ is predicted for which the entry in $\hat{x}_L$ corresponding to the agent's current location is labelled as "backtracked", and we return the backtracked action as the final output action.

Generalization results. We compare procedure cloning to applying data augmentation with random crop, translation, and zoom (Aug BC) or auxiliary objective of predicting the `visited` array (Aux BC) to vanilla BC. BC policies are parametrized with convolutional neural networks (CNN) and multi-layer perceptrons (MLPs). Aux BC receives the same information as PC when computing the auxiliary loss, i.e., the visitation maps are given to both PC and Aux BC. Figure 5.4 (Disc.16 $\times$ 16 and Disc.32 $\times$ 32) shows the average success rate (over 5 trajectories) of reaching the goal from random start locations on 10 test maze layouts unseen during training. Procedure Cloning successfully generalizes to test mazes, whereas vanilla BC completely fails to learn (0% sucess rate) in bigger maze 32 $\times$ 32. Aux BC and Aug BC help in the smaller maze but not in the bigger maze. The poor test performance of BC is due to generalization failure as opposed to insufficient model capacity as BC's success rate on the training mazes are close to 100%.

5.1.4 Experiments

We now evaluate procedure cloning in larger scale settings on tasks including simulated robotic navigation [10] and manipulation [358, 17], and learning to play MinAtar [359] (a miniature version of Atari [360]). Procedure Cloning exhibits significant generalization to previously unseen maze layouts, positions of objects being manipulated, and environment configurations such as transition stochasticity and game difficulty in each of the tasks, respectively. See Appendix A.3.2 for more results.

5.1.4.1 Evaluating continuous robot navigation in AntMaze

Task description. Following the discrete maze navigation task in Section 5.1.3, we now consider a more realistic setting of mimicking real-world robotic navigation. We adopt the AntMaze environment from D4RL [10], where an 8-DoF "Ant" quadruped robot with continuous state and action spaces is placed in a 2D maze environment. The original task designed for offline RL only has one maze layout which is not revealed to the agent (the agent only sees its current position and joint-based state measurements). To adopt the task for evaluating generalization across maze layouts, we also pass the algorithmically generated maze layouts from Section 5.1.3 to the agent as inputs. During evaluation, the agent needs to navigate to the goal in previously unseen maze layouts.

PC implementation. The procedure that generated the expert training trajectories for goal-reaching in AntMaze involves a low-level PID controller that is good for navigating to local locations close to the robot and a high-level waypoint generator which uses BFS search to find the next local waypoint [10]. We apply procedure cloning to the high-level waypoint generator following the same steps as in discrete maze described in Section 5.1.3. The predicted waypoint is then passed to a Gaussian parametrized policy (optimized with max-likelihood) together with the agent's joint measurements. For BC and variants, we use a convolutional neural network to embed the maze layout into a fixed dimensional vector and concatenate the robot joint measurements together with the agent and goal locations to a Gaussian policy.

Generalization results. Figure 5.4 (Ant 10×10 and Ant 13×13) shows the average (over 5 trajectories) success rate of the robot reaching the goal from random starting locations on 10 test mazes. Procedure Cloning consistently provides significant benefits over vanilla BC, whereas data augmentation on the maze layout and predicting the `visited` array as an auxiliary loss are less helpful. We know the poor test performance of BC (and other baseline methods) is more due to generalization failure as opposed to insufficient model capacity because the success rate on the training mazes between procedure cloning and auxiliary BC are similar (Figure A.23 in Appendix A.3.2).

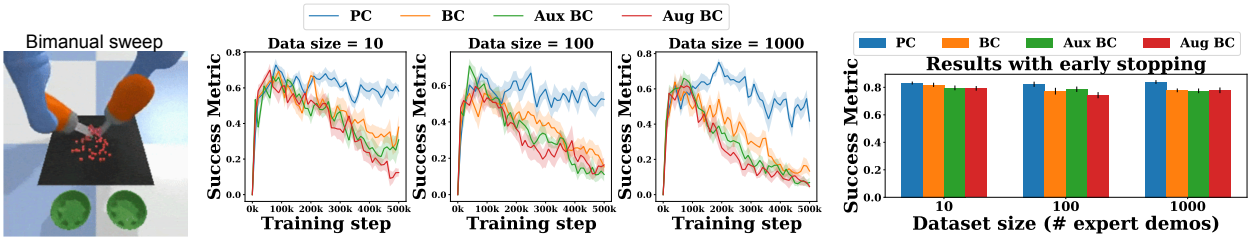


Figure 5.5: [Left] Visualization of the bimanual sweep task. [Middle] Average success metric (proportion of particles in bowls at the end of the episode) of PC and BC agents completing the bimanual sweeping task after learning on 10, 100, 1000 expert trajectories; each variant is an aggregate of 10 runs. All of our algorithm implementations use the implicit loss function described in [17] for this task. [Right] When using 1000 expert demonstrations with early stopping, PC achieves 83.9% compared to 78.2% success of the existing state-of-the-art achieved by implicit BC.

5.1.4.2 Evaluating image-based robot manipulation

Task description. The bimanual sweeping task [358, 17] requires two 7-DoF robot arms equipped with spatula-like end-effectors to sweep a pile of particles evenly into two bowls while avoiding dropping particles between the tips of the spatulas. The scripted oracle for collecting expert trajectories uses access to privileged information including object poses and contact points, which are not accessible at test time. Rather, only high-resolution (96×96) images in conjunction with end-effector positions and orientations are given to the imitation-learned agent during inference. Actions are end-effector positions for each robot arm (6 dimensions for each arm for a total of 12 dimensions for the action); for how to incorporate kinematic actions into this environment, we refer the reader to [358]. Random seeds determining the initial position of the particles are partitioned so that test-time configurations are not seen in training.

PC implementation and results. The scripted policy first scoops the particles by moving the spatulas to their computed geometric center and then moves the spatulas to a bowl before releasing the particles. At each state of a trajectory, we collect the Cartesian coordinates of one of these two goal points – geometric center of the particles or position of the bowl – depending on the expert's behavior mode at that state, and use these as procedure observations. We supervise the PC agent to first predict these coordinates, then predict the actions conditioned on the predicted coordinates and end-effector positions and orientations. For both BC and PC we parameterize log-likelihood using energy-based models, also known as an *implicit* loss, which is state-of-the-art for this task [17]. We compare PC to BC trained directly on image inputs and auxiliary BC which learns to predict oracle coordinates as an auxiliary objective. We see that the BC variants quickly overfit to the training set of observations, whereas PC generalizes much better (Figure 5.5, left). Despite early stopping

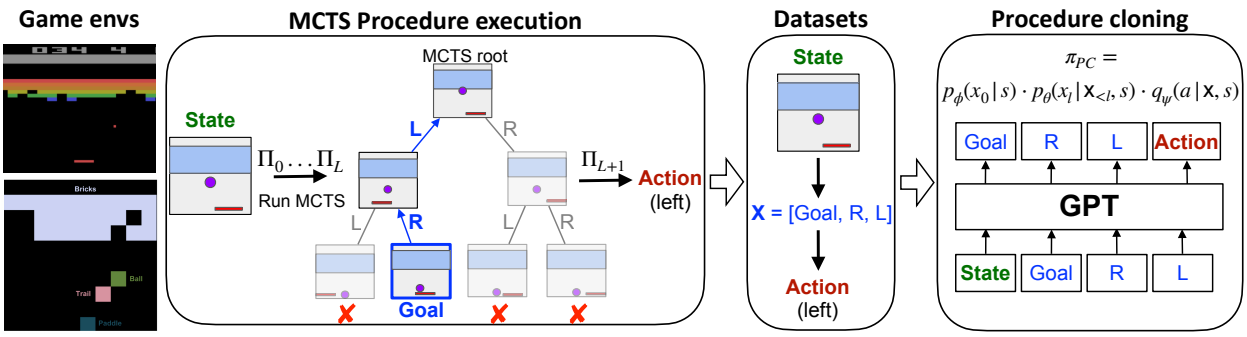


Figure 5.6: In the MinAtar game-playing environment, the expert uses MCTS ($\Pi_0, ..., \Pi_L$) to find an optimal future trajectory [`L`, `R`, `Goal`]. We treat this future trajectory in reverse order [`Goal`, `R`, `L`] as procedure observations, so that procedure cloning is trained to first predict the goal image (MCTS leaf node) and then predict the optimal action sequence backwards from the goal using a GPT-like autoregressive model, ultimately predicting the expert's output action as its last prediction.

(taking the maximum evaluation success rate over all training steps), PC still outperforms all of the BC variants (Figure 5.5, right), improving significantly over the state-of-the-art.

5.1.4.3 Evaluating strategy games in MinAtar

Task description. MinAtar is a minature version of the Atari Arcade Learning Environment [360] consisting of 5 games with simplified 10×10 multi-channel images as inputs. To generate the expert trajectories for training, we run a AlphaZero-style [23] Monte-Carlo tree search (MCTS) algorithm on the deterministic version of the environments to collect expert trajectories (see Appendix A.3.1 for details). During evaluation, we test the imitation-learned agents on a set of test environments with different seeds from the training environments where expert trajectories are collected. To further evaluate generalization, we apply sticky actions with probability 0.1 and game difficulty ramping to the test environments where such an option is available.

PC implementation. In contrast to running BFS in maze navigation where the `visited` array cleanly captures the entire state of the search, running MCTS in MinAtar is more convoluted, involving a number of MCTS simulation runs each with selection, expansion, rollouts, and backtrack steps and different tree structures, making capturing the full search state difficult. Fortunately, procedure cloning with autoregressive factorization (Equation 5.1) allows procedure data to be partial observations of the computation state, and so we elect to use only a subset of the MCTS computation states as supervision for PC. Namely, we record the optimal action sequence *after* the last MCTS simulation run (from the final search tree) and the goal image at the tree leaf as procedure observations (highlighted in blue in

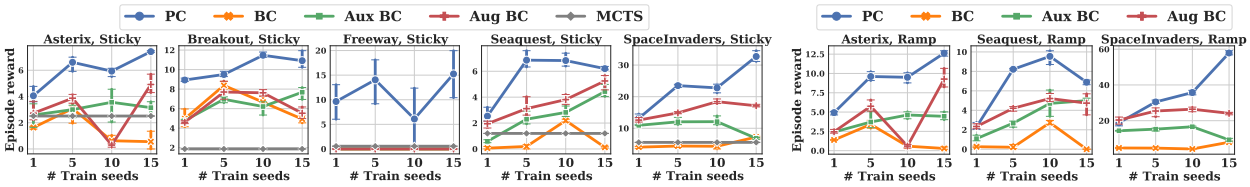


Figure 5.7: Average episode reward (over 50 episodes) of PC and BC agents playing MinAtar games over 3 test environments using sticky actions (left) and game difficulty ramping (right) not see in the training environments.

Figure 5.6); this is effectively the optimal future trajectory determined by MCTS. A PC policy is trained to use the input state to first predict the goal image using a CNN and then use the goal as input to an autoregressive action sequence model $p(x_\ell|\mathbf{x}_{<\ell}, s)$ where x_ℓ is the optimal action that is ℓ-steps away from the goal (i.e., predicting the optimal action sequence *backwards* from the goal). Figure 5.6 illustrates the data and training pipeline of procedure cloning.

Generalization results. Figure 5.7 shows the average reward (over 50 episodes) collected by running PC and BC policies in 3 test environments with different environment seeds than the environments used to collect training trajectories. Figure 5.7 (left) evaluates generalization to stochastic environments, where we found sticky actions caused MCTS (gray) to struggle to search for good actions without extensive tuning. Figure 5.7 (right) evaluates generalization to more difficult game settings (available in 3 out of 5 MinAtar games). Autoregressive PC policy generalizes the best across all games and all settings.

5.2 Overcome Failures of RCSL in Stochastic Environments

5.2.1 Introduction

Offline reinforcement learning (RL) aims to extract an optimal policy solely from an existing dataset of previous interactions [247, 77, 78]. As researchers begin to scale offline RL to large image, text, and video datasets [46, 48, 110, 49, 150], a family of methods known as *return-conditioned supervised learning* (RCSL), including Decision Transformer (DT) [60, 3] and RL via Supervised Learning (RvS) [361], have gained popularity due to their algorithmic simplicity and ease of scaling. At the heart of RCSL is the idea of conditioning a policy on a specific future outcome, often a return [362, 101, 60] but also sometimes a goal state or generic future event [363, 364, 98]. RCSL trains a policy to imitate actions associated with a conditioning input via supervised learning. During inference (i.e., at evaluation), the

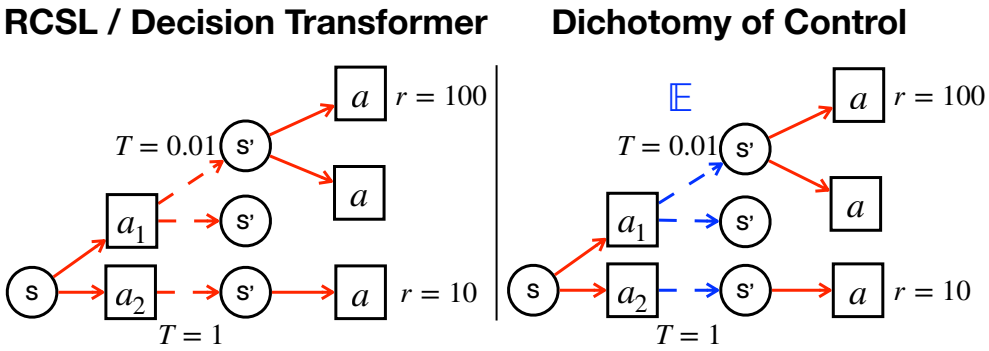


Figure 5.8: Illustration of DT (RCSL) and DoC. Circles and squares denote states and actions. Solid arrows denote policy decisions. Dotted arrows denote (stochastic) environment transitions. All arrows and nodes are present in the dataset, i.e., there are 4 trajectories, 2 of which achieve 0 reward. DT maximizes returns across an entire trajectory, leading to suboptimal policies when a large return ($r = 100$) is achieved only due to very low-probability environment transitions ($T = 0.01$). DoC separates policy stochasticity from that of the environment and only tries to control action decisions (solid arrows), achieving optimal control through maximizing expected returns at each timestep.

policy is conditioned on a desirable high-return or future outcome, with the hope of inducing behavior that can achieve this desirable outcome.

Despite the empirical advantages that come with supervised training [361, 365], RCSL can be highly suboptimal in stochastic environments [103, 102], where the future an RCSL policy conditions on (e.g., return) can be primarily determined by randomness in the environment rather than the data collecting policy itself. Figure 5.8 (left) illustrates an example, where conditioning an RCSL policy on the highest return observed in the dataset ($r = 100$) leads to a policy (a_1) that relies on a stochastic transition of very low probability ($T = 0.01$) to achieve the desired return of $r = 100$; by comparison the choice of a_2 is much better in terms of average return, as it surely achieves $r = 10$. The crux of the issue is that the RCSL policy is *inconsistent* with its conditioning input. Conditioning the policy on a desired return (i.e., 100) to act in the environment leads to a distribution of real returns (i.e., $0.01 * 100$) that is wildly different from the return value being conditioned on. This issue would not have occurred if the policy could also maximize the transition probability that led to the high-return state, but this is not possible as transition probabilities are a part of the environment and not subject to the policy's control.

A number of works propose a generalization of RCSL, known as future-conditioned supervised learning methods. These techniques have been shown to be effective in imitation learning [107, 13], offline Q-learning [14], and online policy gradient [366]. It is common in future-conditioned supervised learning to apply a KL divergence regularizer on the latent variable – inspired by variational auto-encoders (VAE) [92] and measured with respect

to a learned prior conditioned only on past information – to limit the amount of future information captured in the latent variable. It is natural to ask whether this regularizer could remedy the insconsistency of RCSL. Unfortunately, as the KL regularizer makes no distinction between future information that is controllable versus that which is not, such an approach will still exhibit inconsistency, in the sense that the latent variable representation may contain information about the future that is due only to environment stochasticity.

It is clear that the major issue with both RCSL and naïve variational methods is that they make no distinction between stochasticity of the policy (controllable) and stochasticity of the environment (uncontrollable) [367, 368]. An optimal policy should maximize over the controllable (actions) and take expectations over uncontrollable (e.g., transitions) as shown in Figure 5.8 (right). This implies that, under a variational approach, the latent variable representation that a policy conditions on should not incorporate any information that is solely due to randomness in the environment. In other words, while the latent representation can and should include information about future behavior (i.e., actions), it should not reveal any information about the rewards or transitions associated with this behavior.

To this end, we propose a future-conditioned supervised learning framework termed *dichotomy of control* (DoC), which, in Stoic terms [369], has *"the serenity to accept the things it cannot change, courage to change the things it can, and wisdom to know the difference."* DoC separates mechanisms within a policy's control (actions) from those beyond a policy's control (environment stochasticity). To achieve this separation, we condition the policy on a latent variable representation of the future while minimizing the mutual information between the latent variable and future stochastic rewards and transitions in the environment. Theoretically, we show that DoC policies are *consistent* with their conditioning inputs, ensuring that conditioning on a high-return future will correctly induce high-return behavior. Empirically, we show that DoC can outperform both RCSL and naïve variational methods on highly stochastic environments.

5.2.2 Related Work

Return-Conditioned Supervised Learning. Since offline RL algorithms [247, 77, 78] can be sensitive to hyper-parameters and difficult to apply in practice [361, 365], return-conditioned supervised learning (RCSL) has become a popular alternative, particularly when the environment is deterministic and near-expert demonstrations are available [102]. RCSL learns to predict behaviors (actions) by conditioning on desired returns [100, 101] using an MLP policy [361] or a transformer-based policy that encapsulates history [60]. Richer information other than returns, such as goals [363, 364] or trajectory-level aggregates [351], have also been used as inputs to a conditional policy in practice. Our work also conditions policies on richer trajectory-level information in the form of a latent variable representation of the future, with additional theoretical justifications of such conditioning in stochastic environments.

RCSL Failures in Stochastic Environments. Despite the empirical success of RCSL achieved by DT and RvS, recent work has noted the failure modes in stochastic environments. [367] and [368] presented counter-examples where online RvS can diverge in stochastic environments. [103] first identified the failure of return-conditioned supervised learning with stochastic transitions and proposed to cluster offline trajectories and condition the policy on the average cluster returns. While conditioning on expected as opposed to maximum returns is more reasonable, this approach has technical limitations and can lead to undesirable policy-averaging, i.e., a single policy covering two very different behaviors (clusters) that happen to have the same return. [102] also identified near-determinism as a necessary condition for RCSL to achieve optimality guarantees similar to other offline RL algorithms but did not propose a solution for RCSL in stochastic settings. [370] also identifies overly optimistic behavior of DT and proposes to use discrete β-VAE to induce diverse future predictions a policy can condition on. This approach only differs the issue with stochastic environments to stochastic latent variables, i.e., the latent variables will still contain stochastic environment information that the policy cannot reliably reproduce.

Learning Latent Variables from Offline Data. Various works have explored learning a latent variable representation of the future (or past) transitions in offline data via maximum likelihood and use the latent variable to assist planning [98], imitation learning [290, 14, 278], offline RL [14, 291], or online RL [292, 293, 371, 294, 107, 296, 366]. These works generally focus on the benefit of increased temporal abstraction afforded by using latent variables as higher-level actions in a hierarchical policy. [370] has introduced latent variable models into RCSL, which is one of the essential tools that enables our method, but they did not incoporate the appropriate constraints which can allow RCSL to effectively combat environment stochasticity, as we will see in our work. Lastly, [372] used mutual information constraints to separate controllable from uncontrollable aspects of an MDP with the goal of accelerating reinforcement learning, whereas we study the dichotomy of control under the context of return-and-future conditioned supervised learning.

5.2.3 Preliminaries

Learning a Policy in RCSL In future- or return-conditioned supervised learning, one uses a fixed training data distribution $\mathcal{D}$ of episodes τ (collected by unknown and potentially multiple agents) to learn a policy π, where π is trained to predict a_t conditioned on the history $\tau_{0:t-1}$, the observation s_t, and an additional conditioning variable z that may depend on both the past and future of the episode. For example, in return-conditioned supervised learning, policy training minimizes the following objective over π:

$$\mathcal{L}_{\text{RCSL}}(\pi) := \mathbb{E}_{\tau\sim\mathcal{D}}\left[\sum_{t=0}^{H} -\log \pi(a_t|\tau_{0:t-1}, s_t, z(\tau))\right], \tag{5.5}$$

where $z(\tau)$ is the return $R(\tau)$.

Inconsistency of RCSL To apply an RCSL-trained policy π during inference — i.e., interacting online with the environment — one must first choose a specific z.[1] For example, one might set z to be the maximal return observed in the dataset, in the hopes of inducing a behavior policy which achieves this high return. Using π_z as a shorthand to denote the policy π conditioned on a specific z, we define the expected return $V_{\mathcal{M}}(\pi_z)$ of π_z in $\mathcal{M}$ as,

$$V_{\mathcal{M}}(\pi_z) := \mathbb{E}_{\tau \sim \Pr[\cdot|\pi_z, \mathcal{M}]}\left[R(\tau)\right]. \tag{5.6}$$

Ideally the expected return induced by π_z is close to z, i.e., $z \approx V_{\mathcal{M}}(\pi_z)$, so that acting according to π conditioned on a high return induces behavior which actually achieves a high return. However, RCSL training according to Equation (5.5) will generally yield policies that are highly *inconsistent* in stochastic environments, meaning that the achieved returns may be significantly different than z (i.e., $V_{\mathcal{M}}(\pi_z) \neq z$). This has been highlighted in various previous works [102, 103, 368, 373, 370], and we provided our own example in Figure 5.8.

Approaches to Mitigating Inconsistency A number of future-conditioned supervised learning approaches propose to learn a stochastic latent variable embedding of the future, $q(z|\tau)$, while regularizing q with a KL-divergence from a learnable *prior* conditioned only on the past $p(z|s_0)$ [14, 366, 98], thereby minimizing:

$$\mathcal{L}_{\text{VAE}}(\pi, q, p) := \mathbb{E}_{\tau \sim \mathcal{D}, z \sim q(z|\tau)}\left[\sum_{t=0}^{H} -\log \pi(a_t|\tau_{0:t-1}, s_t, z)\right] + \beta \cdot \mathbb{E}_{\tau \sim \mathcal{D}}\left[D_{\text{KL}}(q(z|\tau) \| p(z|s_0))\right]. \tag{5.7}$$

One could consider adopting such a future-conditioned objective in RCSL. However, since the KL regularizer makes no distinction between observations the agent can control (actions) from those it cannot (environment stochasticity), the choice of coefficient β applied to the regularizer introduces a 'lose-lose' trade-off. Namely, as noted in [14], if the regularization coefficient is too large ($\beta \geq 1$), the policy will not learn diverse behavior (since the KL limits how much information of the future actions is contained in z); while if the coefficient is too small ($\beta < 1$), the policy's learned behavior will be inconsistent (in the sense that z will contain information of environment stochasticity that the policy cannot reliably reproduce). The discrete β-VAE incoporated by [370] with $\beta < 1$ corresponds to this second failure mode.

5.2.4 Dichotomy of Control

In this section, we first propose the DoC objective for learning future-conditioned policies that are guaranteed to be consistent. We then present a practical framework for optimizing DoC's constrained objective in practice and an inference scheme to enable better-than-dataset behavior via a learned value function and prior.

[1] For simplicitly, we assume z is chosen at timestep $t = 0$ and held constant throughout an entire episode. As noted in [102], this protocol also encompasses instances like DT [60] in which z at timestep t is the (desired) return summed starting at t.

5.2.4.1 Dichotomy of Control via Mutual Information Minimization

As elaborated in the prevous section, whether $z(\tau)$ is the return $R(\tau)$ or more generally a stochastic latent variable with distribution $q(z|\tau)$, existing RCSL methods fail to satisfy consistency because they insufficiently enforce the type of future information z can contain. A key observation is that z should not include any information due to environment stochasticity, i.e., any information about a future r_t, s_{t+1} that is not already known given the previous history up to that point $\tau_{0:t-1}, s_t, a_t$. (A similar observation was made in [103] under the more restrictive assumption that z is a cluster index, which we do not require here.) To address the independence requirement in a general and sound way, we modify the RCSL objective from Equation 5.5 to incorporate a conditional mutual information constraint between z and each pair r_t, s_{t+1} in the future:

$$\mathcal{L}_{\text{DoC}}(\pi, q) := \mathbb{E}_{\tau\sim\mathcal{D}, z\sim q(z|\tau)}\left[\sum_{t=0}^{H} -\log \pi(a_t|\tau_{0:t-1}, s_t, z)\right]$$

$$\text{s.t.} \quad \text{MI}(r_t; z \mid \tau_{0:t-1}, s_t, a_t) = 0, \text{MI}(s_{t+1}; z \mid \tau_{0:t-1}, s_t, a_t) = 0, \tag{5.8}$$

$$\forall\ \tau_{0:t-1}, s_t, a_t \text{ and } 0 \le t \le H, \tag{5.9}$$

where $\text{MI}(r_t; z|\tau_{0:t-1}, s_t, a_t)$ denotes the mutual information between r_t and z given $\tau_{0:t-1}, s_t, a_t$ when measured under samples of r_t, z from $\mathcal{D}, q$; and analogously for $\text{MI}(s_{t+1}; z|\tau_{0:t-1}, s_t, a_t)$.

The first part of the DoC objective conditions the policy on a latent variable representation of the future, similar to the first part of the future-conditioned VAE objective in Equation (5.7). However, unlike Equation (5.7), the DoC objective enforces a much more precise constraint on q, given by the MI constraints in Equation (5.8).

5.2.4.2 Dichotomy of Control in Practice

Contrastive Learning of DoC Constraints. To satisfy the mutual information constraints in Equation (5.8) we transform the MI to a contrastive learning objective. Specifically, for the constraint on r and z (and similarly on s_{t+1} and z) one can derive,

$$\begin{aligned}
&\text{MI}(r_t; z|\tau_{0:t-1}, s_t, a_t) \\
&\quad = D_{\text{KL}}\left(\Pr[r_t, z|\tau_{0:t-1}, s_t, a_t] \| \Pr[r_t|\tau_{0:t-1}, s_t, a_t]\Pr[z|\tau_{0:t-1}, s_t, a_t]\right) \\
&\quad = \mathbb{E}_{\Pr[r_t, z|\tau_{0:t-1}, s_t, a_t]}\left[\log\left(\frac{\Pr[r_t|z, \tau_{0:t-1}, s_t, a_t]}{\Pr[r_t|\tau_{0:t-1}, s_t, a_t]}\right)\right] \\
&\quad = \mathbb{E}_{\Pr[r_t, z|\tau_{0:t-1}, s_t, a_t]} \log \Pr[r_t|z, \tau_{0:t-1}, s_t, a_t] - \mathbb{E}_{\Pr[r_t|\tau_{0:t-1}, s_t, a_t]} \log \Pr[r_t|\tau_{0:t-1}, s_t, a_t].
\end{aligned} \tag{5.10}$$

The second expectation above is a constant with respect to z and so can be ignored during learning. We further introduce a conditional distribution $\omega(r_t|z, \tau_{0:t-1}, s_t, a_t)$ parametrized by an energy-based function $f : \Omega \mapsto \mathbb{R}$:

$$\omega(r_t|z, \tau_{0:t-1}, s_t, a_t) \propto \rho(r_t) \exp\left\{f(r_t, z, \tau_{0:t-1}, s_t, a_t)\right\}, \tag{5.11}$$

where ρ is some fixed sampling distribution of rewards. In practice, we set ρ to be the marginal distribution of rewards in the dataset. Hence we express the first term of Equation 5.10 via an optimization over ω, i.e.,

$$\begin{aligned}&\max_{\omega} \ \mathbb{E}_{\Pr[r_t, z|\tau_{0:t-1}, s_t, a_t]} \left[\log \omega(r_t|z, \tau_{0:t-1}, s_t, a_t)\right] \\ =&\max_{f} \mathbb{E}_{\Pr[r_t, z|\tau_{0:t-1}, s_t, a_t]} \left[f(r_t, z, \tau_{0:t-1}, s_t, a_t) - \log \mathbb{E}_{\rho(\tilde{r})} \left[\exp\{f(\tilde{r}, z, \tau_{0:t-1}, s_t, a_t)\}\right]\right].\end{aligned}$$

Combining this (together with the analogous derivation for $\mathrm{MI}(s_{t+1}; z|\tau_{0:t-1}, s_t, a_t)$) with Equation 5.8 via the Lagrangian, we can learn π and $q(z|\tau)$ by minimizing the final DoC objective:

$$\begin{aligned}\mathcal{L}_{\mathrm{DoC}}(\pi, q) = \max_{f} \mathbb{E}_{\tau\sim\mathcal{D}, z\sim q(z|\tau)} \left[\sum_{t=0}^{H} -\log \pi(a_t|\tau_{0:t-1}, s_t, z)\right] \\ +\beta \cdot \sum_{t=0}^{H} \mathbb{E}_{\tau\sim\mathcal{D}, z\sim q(z|\tau)} \left[f(r_t, s_{t+1}, z, \tau_{0:t-1}, s_t, a_t) - \log \mathbb{E}_{\rho(\tilde{r}, \tilde{s}')} \left[\exp\{f(\tilde{r}, \tilde{s}', z, \tau_{0:t-1}, s_t, a_t)\}\right]\right].\end{aligned} \tag{5.12}$$

Algorithm 1 Inference with Dichotomy of Control

Inputs Policy $\pi(\cdot|\cdot, \cdot, \cdot)$, prior $p(\cdot)$, value function $V(\cdot)$, initial state s_0, number of samples hyperparameter K.
Initialize $z^*; V^*$ ▷ `Track the best latent and its value.`
for $k = 1$ to K **do**
  Sample $z_k \sim p(z|s_0)$ ▷ `Sample a latent from the learned prior.`
  if $V(z_k) > V^*$ **then**
    $z^* = z_k; V^* = V$ ▷ `Set best latent to the one with the highest value.`
return $\pi(\cdot|\cdot, \cdot, z^*)$ ▷ `Policy conditioned on the best` z^*.

DoC Inference. As is standard in RCSL approaches, the policy learned by DoC requires an appropriate conditioning input z to be chosen during inference. To choose a desirable z associated with high return, we propose to (1) enumerate or sample a large number of potential values of z, (2) estimate the expected return for each of these values of z, (3) choose the z with the highest associated expected return to feed into the policy. To enable such an inference-time procedure, we need to add two more components to the method formulation: First, a prior distribution $p(z|s_0)$ from which we will sample a large number of values of z; second, a value function $V(z)$ with which we will rank the potential values of z. These components are learned by minimizing the following objective:

$$\mathcal{L}_{\mathrm{aux}}(V, p) = \mathbb{E}_{\tau\sim\mathcal{D}, z\sim q(z|\tau)} \Big[\left(V(z) - R(\tau)\right)^2 + D_{\mathrm{KL}}(\mathrm{stopgrad}(q(z|\tau)) \| p(z|s_0))\Big]. \tag{5.13}$$

Note that we apply a stop-gradient to $q(z|\tau)$ when learning p so as to avoid regularizing q with the prior. This is unlike the VAE approach, which by contrast advocates *for* regularizing q via the prior. See Algorithm 1 for inference pseudocode (and Appendix A.4.4 for training pseudocode.

5.2.5 Consistency Guarantees for Dichotomy of Control

We provide a theoretical justification of the proposed learning objectives $\mathcal{L}_{\text{DoC}}$ and $\mathcal{L}_{\text{aux}}$, showing that, if they are minimized, the resulting inference-time procedure will be sound, in the sense that DoC will learn a V and π such that the true value of π_z in the environment $\mathcal{M}$ is equal to $V(z)$. More specifically we define the following notion of *consistency*:

Definition 8 (Consistency)**.** *A future-conditioned policy π and value function V are **consistent** for a specific conditioning input z if the expected return of z predicted by V is equal to the true expected return of π_z in the environment: $V(z) = V_{\mathcal{M}}(\pi_z)$.*

To guarantee consistency of π, V, we will make the following two assumptions:

Assumption 9 (Data and environment agreement)**.** *The per-step reward and next-state transitions observed in the data distribution are the same as those of the environment. In other words, for any $\tau_{0:t-1}, s_t, a_t$ with $\Pr[\tau_{0:t-1}, s_t, a_t|\mathcal{D}] > 0$, we have $\Pr[\hat{r}_t = r_t|\tau_{0:t-1}, s_t, a_t, \mathcal{D}] = \mathcal{R}(\hat{r}_t|\tau_{0:t-1}, s_t, a_t)$ and $\Pr[\hat{s}_{t+1} = s_{t+1}|\tau_{0:t-1}, s_t, a_t, \mathcal{D}] = \mathcal{T}(\hat{s}_{t+1}|\tau_{0:t-1}, s_t, a_t)$ for all $\hat{r}_t, \hat{s}_{t+1}$.*

Assumption 10 (No optimization or approximation errors)**.** *DoC yields policy π and value function V that are Bayes-optimal with respect to the training data distribution and q. In other words, $V(z) = \mathbb{E}_{\tau\sim\Pr[\cdot|z,\mathcal{D}]}[R(\tau)]$ and $\pi(\hat{a}|\tau_{0:t-1}, s_t, z) = \Pr[\hat{a} = a_t|\tau_{0:t-1}, s_t, z, \mathcal{D}]$.*

Given these two assumptions, we can then establish the following consistency guarantee for DoC.

Theorem 11. *Suppose DoC yields π, V, q with q satisfying the MI constraints:*

$$\text{MI}(r_t; z|\tau_{0:t-1}, s_t, a_t) = \text{MI}(s_{t+1}; z|\tau_{0:t-1}, s_t, a_t) = 0, \tag{5.14}$$

for all $\tau_{0:t-1}, s_t, a_t$ with $\Pr[\tau_{0:t-1}, s_t, a_t|\mathcal{D}] > 0$. Then under Assumptions 9 and 10, V and π are consistent for any z with $\Pr[z|q, \mathcal{D}] > 0$.

Consistency in Markovian environments. While the results above are focused on environments and policies that are non-Markovian, one can extend Theorem 11 to Markovian environments and policies. This result is somewhat surprising, as the assignments of z to episodes τ induced by q are necessarily history-dependent, and projecting the actions appearing in these clusters to a non-history-dependent policy would seemingly lose important information. However, a Markovian assumption on the rewards and transitions of the environment is sufficient to ensure that no 'important' information will be lost, at least in

terms of the satisfying requirements for consistency in Definition 8. Alternative notions of consistency are not as generally applicable.

We begin by stating our assumptions.

Assumption 12 (Markov environment)**.** *The rewards and transitions of $\mathcal{M}$ are Markovian; i.e., $\mathcal{R}(\tau_{0:t-1}, s_t, a_t) = \mathcal{R}(\tilde{\tau}_{0:t-1}, s_t, a_t)$ and $\mathcal{T}(\tau_{0:t-1}, s_t, a_t) = \mathcal{T}(\tilde{\tau}_{0:t-1}, s_t, a_t)$ for all $\tau, \tilde{\tau}, s_t, a_t$. We use the shorthand $\mathcal{R}(s_t, a_t), \mathcal{T}(s_t, a_t)$ for these history-independent functions.*

Assumption 13 (Markov policy, without optimization or approximation errors)**.** *The policy learned by DoC is Markov. This policy π as well as its corresponding learned value function V are Bayes-optimal with respect to the training data distribution and q. In other words, $V(z) = \mathbb{E}_{\tau \sim \Pr[\cdot|z,\mathcal{D}]}[R(\tau)]$ and $\pi(\hat{a}|s_t, z) = \Pr[\hat{a} = a_t|s_t, z, \mathcal{D}]$.*

With these two assumptions, we can then establish the analogue to Theorem 11, which relaxes the dependency on history for both the policy π and the MI constraints:

Theorem 14. *Suppose DoC yields π, V, q with q satisfying the MI constraints:*

$$\mathrm{MI}(r_t; z|s_t, a_t) = \mathrm{MI}(s_{t+1}; z|s_t, a_t) = 0, \tag{5.15}$$

for all s_t, a_t with $\Pr[s_t, a_t|\mathcal{D}] > 0$. Then under Assumptions 9, 12, and 13, V and π are consistent for any z with $\Pr[z|q, \mathcal{D}] > 0$.

5.2.6 Experiments

We conducted an empirical evaluation to ascertain the effectiveness of DoC. For this evaluation, we considered three settings: (1) a Bernoulli bandit problem with stochastic rewards, based on a canonical 'worst-case scenario' for RCSL [102]; (2) the FrozenLake domain from [327], where the future VAE approach proves ineffective; and finally (3) a modified set of OpenAI Gym [327] environments where we introduced environment stochasticity. In these studies, we found that DoC exhibits a significant advantage over RCSL/DT, and outperforms future VAE when the analogous to "one-step" RL is insufficient. For DT, we use the same implementation and hyperparameters as [60]. Both VAE and DoC are built upon the DT implementation and additionally learn a Gaussian latent variable over succeeding 20 future steps. See experiment details in Appendix A.4.5 and additional results in Appendix A.4.6.

5.2.6.1 Evaluating Stochastic Rewards in Bernoulli Bandit

Bernoulli Bandit. Consider a two-armed bandit as shown in Figure 5.9 (left). The two arms, a_1, a_2, have stochastic rewards drawn from Bernoulli distributions of $\texttt{Bern}(1-p)$ and $\texttt{Bern}(p)$, respectively. In the offline dataset, the a_1 arm with reward $\texttt{Bern}(1-p)$ is pulled with probability $\pi_D(a_1) = p$. When p is small, this corresponds to the better arm only being pulled occasionally. Under this setup, $\pi_{\mathrm{RCSL}}(a_1|r=1) = \pi_{\mathrm{RCSL}}(a_2|r=1) = 0.5$, which is highly suboptimal compared to always pulling the optimal arm a_1 with reward $\texttt{Bern}(1-p)$ for $p < 0.5$.

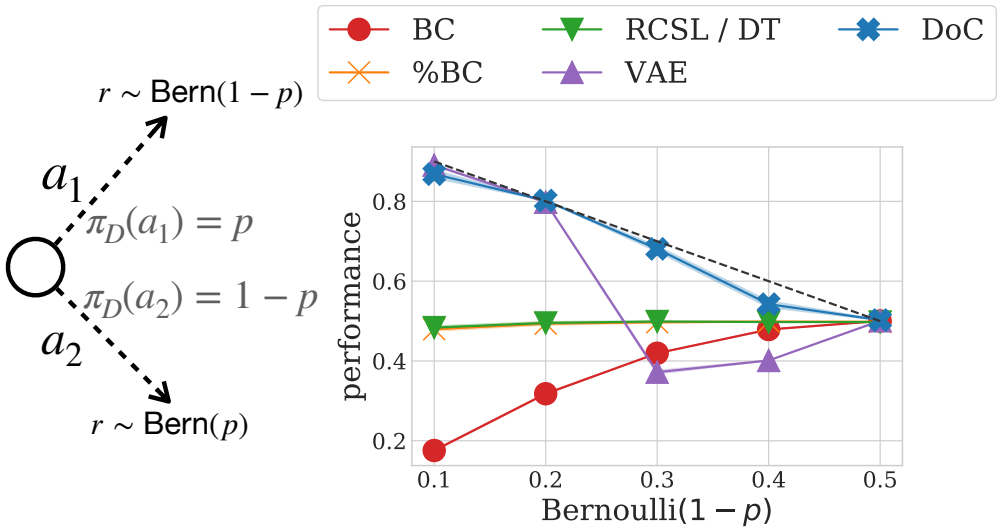


Figure 5.9: [Left] Bernoulli bandit where the better arm a_1 with reward $\texttt{Bern}(1-p)$ for $p < 0.5$ is pulled with probability $\pi_D(a_1) = p$ in the offline data. [Right] Average rewards achieved by DoC and baselines across 5 environment seeds. RCSL is highly suboptimal when p is small, whereas DoC achieves close to Bayes-optimal performance (dotted line) for all values of p.

Results. We train tabular DoC and baselines on 1000 samples where the superior arm with $r \sim \texttt{Bern}(1-p)$ is pulled with probability p for $p \in \{0.1, ..., 0.5\}$. Figure 5.9 (right) shows that RCSL and percentage BC (filtered by $r = 1$) always result in policies that are indifferent in the arms, whereas DoC is able to recover the Bayes-optimal performance (dotted line) for all p values considered. Future VAE performs similarly to DoC for small p values, but is sensitive to the KL regularization coefficient when p is close to 0.5.

5.2.6.2 Evaluating Stochastic Transitions in FrozenLake

FrozenLake. Next, we consider the FrozenLake environment with stochastic transitions where the agent taking an action has probability p of moving in the intended direction, and probability $0.5 \cdot (1-p)$ of slipping to either of the two sides of the intended direction. We collect 100 trajectories of length 100 using a DQN policy trained in the original environment ($p = \frac{1}{3}$) which achieves an average return of 0.7, and vary p during data collection and evaluation to test different levels of stochasticity. We also include uniform actions with probability ϵ to lower the performance of the offline data so that BC is highly suboptimal.

Results. Figure 5.10 presents the visualization (left) and results (right) for this task. When the offline data is closer to being expert ($\epsilon = 0.3$), DT, future VAE, and DoC perform similarly with better performance in more deterministic environments. As the offline dataset

becomes more suboptimal ($\epsilon = 0.5$), DoC starts to dominate across all levels of transition stochasticity. When the offline data is highly suboptimal ($\epsilon = 0.7$), DT and future VAE has little advantage over BC, whereas DoC continues to learn policies with reasonable performance.

5.2.6.3 Evaluating Stochastic Gym MuJoCo

Environments. We now consider a set of Gym MuJoCo environments including Reacher, Hopper, HalfCheetah, and Humanoid. We additionally consider AntMaze from D4RL [10]. These environments are deterministic by default, which we modify by introducing time-correlated Gaussian noise to the actions before inputing the action into the physics simulator during data collection and evaluation for all but AntMaze environments. Specifically, the Gaussian noise we introduce to the actions has 0 mean and standard deviation of the form $(1 - e^{-0.01 \cdot t}) \cdot \sin(t) \cdot \sigma$ where t is the step number and $\sigma \in [0, 1]$. For AntMaze where the dataset has already been collected in the deterministic environment by D4RL, we add gaussian noise with 0.1 standard deviation to the reward uniformly with probability 0.1 (both to the dataset and during evaluation).

Results. Figure 5.11 shows the average performance (across 5 seeds) of DT, future VAE, and DoC on these stochastic environments. Both future VAE and DoC generally provide benefits over DT, where the benefit of DoC is more salient in harder environments such as HalfCheetah and Humanoid. We found future VAE to be sensitive to the β hyperparameter, and simply using $\beta = 1$ can result in the falure case as shown in Reacher-v2.

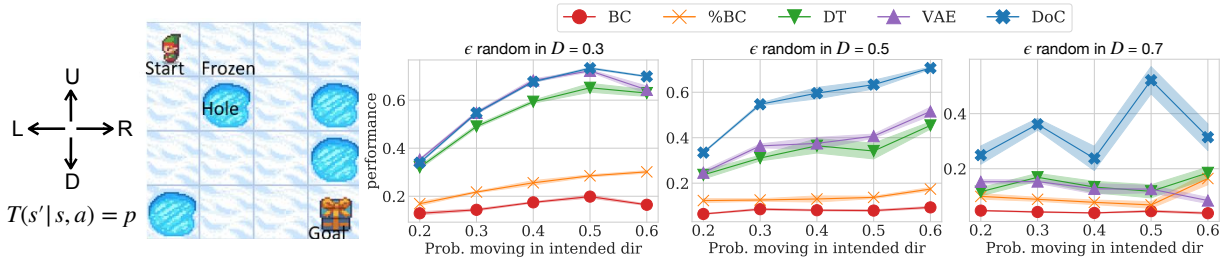


Figure 5.10: [Left] Visualization of the stochastic FrozenLake task. The agent has a probability p of moving in the intended direction and $1 - p$ of slipping to either sides. [Right] Average performance (across 5 seeds) of DoC and baselines on FrozenLake with different levels of stochasticity (p) and offline dataset quality (ϵ). DoC outperforms DT and future VAE, where the gain is more salient when the offline data is less optimal ($\epsilon = 0.5$ and $\epsilon = 0.7$).

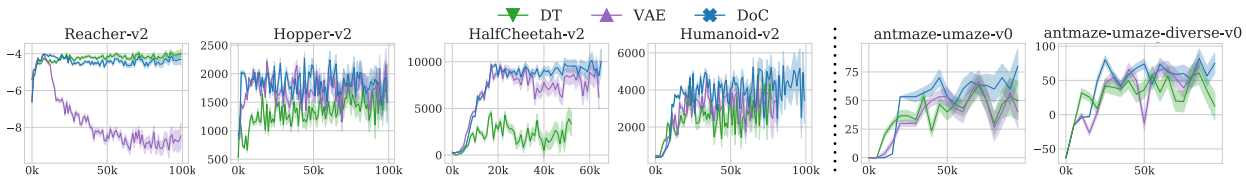


Figure 5.11: Average performance (across 5 seeds) of DoC and baselines on modified stochastic Gym MuJoCo and AntMaze tasks. DoC and future VAE generally provide benefits over DT, where DoC provide more benefits on harder tasks such as Humanoid. Future VAE can be sensitive to the KL coefficient β, which can result in the failure mode shown in Reacher-v2 if not tuned properly.

Chapter 6

RL, Search, and Planning with Internet-Scale Videos

In the previous chapter, we have seen that conditional generation can be used to flexibly parametrize policies that can incorporate auxiliary information such as chain-of-thought reasoning steps or desirable future returns. So far, we have only considered datasets that are largely task specific and are in the form of interactive data with interleaved state, action, reward, and some auxiliary information. It is worth considering much broader dataset such as text and videos from the internet, and investigate how information contained in these broad data may accelerate decision making problems.

There has been tremendous progress in training large language models (LLMs) from internet text datasets in the past few years. The impressive performance of LLMs on a wide variety of tasks makes it tempting to reduce the artificial intelligence agenda to scaling up these systems. However, this is not sufficient. Firstly, the quantity of publicly available text data is becoming a bottleneck to further scaling. Secondly, and perhaps more importantly, natural language alone might not be enough to describe all intelligent behavior or capture all information about the physical world we live in (e.g., imagine teaching someone how to tie a knot using words only). While language is a powerful tool to describe higher-level abstractions, it is not always sufficient to capture the physical world in all its wealth of detail.

Thankfully, there are abundant video data on the internet (e.g., over ten thousand years of consecutive video watching from YouTube alone) encapsulating a wealth of information imbued with knowledge of the world. Nevertheless, today's machine learning models trained on internet text or video data have demonstrated remarkably different capabilities. LLMs have advanced to tackling intricate tasks that require sophisticated reasoning, tool use, and decision making. In contrast, video generation models have been less explored, primarily focusing on creating entertainment videos for human consumption [28, 374]. Given the paradigm shift unfolding in language modeling, it is important to ask whether we can elevate video generation models to the level of autonomous agents, simulation environments, and computational engines similar to language models so that applications requiring visual modalities such as robotics, self-driving, and science can more directly benefit from internet visual knowledge and pretrained video models.

In this chapter, we take the position that *video generation will be to the physical world as language modeling is to the digital world.* To arrive at this conclusion, we first show how video generation is a universal policy that can characterize broad robotics tasks and even learn from human videos (Section 6.1). Next, we further illustrate that video generation is not only a universal policy, but also a universal environment simulator (Section 6.2). With such a universal simulator, diverse activities from those of humans, to robot manipulation, and to various dynamical systems, can all be simulated using a unified interface. Lastly, we have noted that video generation can be expensive to train and evaluate. Therefore, we propose efficient adaptation of video generation models to task specific domains (Section 6.3). Overall, this chapter shows that video generation can be treated as a unified interface to solve real-world decision making tasks such as robotics and self-driving. This last chapter opens many research opportunities in leveraging internet data to learn generalist agent and environment models.

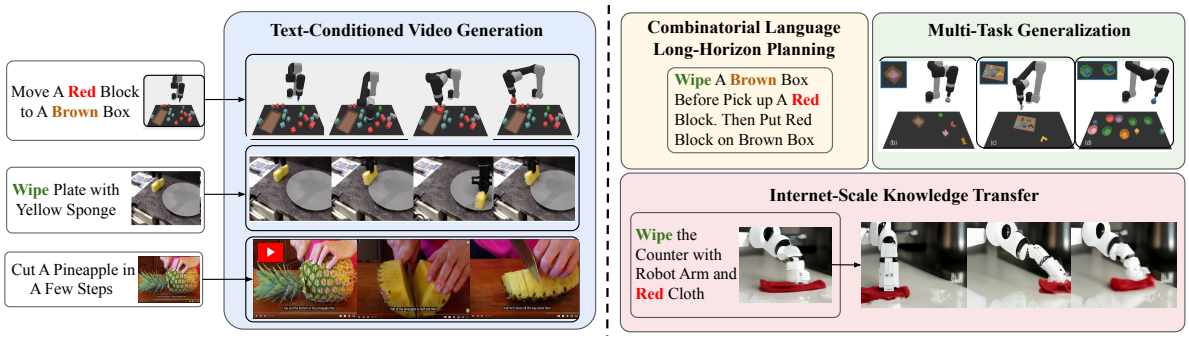


Figure 6.1: **Text-Conditional Video Generation as Universal Policies.** Text-conditional video generations enables us to train general purpose policies on wide sources of data (simulated, real robots and YouTube) which may be applied to downstream multi-task settings requiring combinatorical language generalization, long-horizon planning, or internet-scale knowledge.

6.1 Text-to-Video as Universal Policies

6.1.1 Introduction

Building models that solve a diverse set of tasks has become a dominant paradigm in the domains of vision and language. In natural language processing, large pretrained models have demonstrated remarkable zero-shot learning of new language tasks [27, 375, 376]. Similarly, in computer vision, models such as those proposed in [2, 377] have shown remarkable zero-shot classification and object recognition capabilities. A natural next step is to use such tools to construct agents that can complete different decision making tasks across many environments.

However, training such agents faces the inherent challenge of environmental diversity, since different environments operate with distinct state action spaces (e.g., the joint space and continuous controls in MuJoCo are fundamentally different from the image space and discrete actions in Atari). Such diversity hampers knowledge sharing, learning, and generalization across tasks and environments. Although substantial effort has been devoted to encoding different environments with universal tokens in a sequence modeling framework [49], it is unclear whether such an approach can preserve the rich knowledge embedded in pre-trained vision and language models and leverage this knowledge to transfer to downstream reinforcement learning (RL) tasks. Furthermore, it is difficult to construct reward functions which specify different tasks across environments.

In this work, we address the challenges in environment diversity and reward specification by leveraging *video* (i.e., image sequences) as a universal interface for conveying action and observation behavior in different environments, and *text* as a universal interface for expressing task descriptions. In particular, we design a video generator as a planner that

sequentially conditions on a current image frame and a text snippet describing a current goal (i.e., the next high-level step) to generate a trajectory in the form of an image sequence, after which an inverse dynamics model is used to extract the underlying actions from the generated video. Such an approach allows the universal nature of language and video to be leveraged in generalizing to novel goals and tasks across diverse environments. Specifically, we instantiate the text-conditioned video generation model using video diffusion. A set of underlying actions are then regressed from the synthesized frames and used to construct a policy to implement the planned trajectory. The proposed model, *UniPi*, is visualized in Figure 6.1.

We have found that formulating policy generation via text-conditioned video synthesis yields the following advantages:

Combinatorial Generalization. The rich combinatorial nature of language can be leveraged to synthesize novel combinatorial behaviors in the environment. This enables the proposed approach to rearrange objects to new unseen combinations of geometric relations, as shown in Section 6.1.4.1.

Multi-task Learning. Formulating action prediction as a video prediction problem readily enables learning across many different tasks. We illustrate in Section 6.1.4.2 how this enables learning across language-conditioned tasks and generalizing to new ones at test time without finetuning.

Action Planning. The video generation procedure corresponds to a planning procedure where a sequence of frames representing actions is generated to reach the target goal. Such a planning procedure is naturally hierarchical: a temporally sparse sequence of images toward a goal can first be generated, before being refined with a more specific plan. Moreover, the planning procedure is steerable, in the sense that the plan generation can be biased by new constraints introduced at test-time through test-time sampling. Finally, plans are produced in a video space that is naturally interpretable by humans, making action verification and plan diagnosis easy. We illustrate the efficacy of hierarchical sampling in Table 6.2 and steerability in Figure 6.5.

Internet-Scale Knowledge Transfer. By pretraining a video generation model on a large-scale text-video dataset recovered from the internet, one can recover a vast repository of "demonstrations" that aid the construction of a text-conditioned policy in novel environments. We illustrate how this enables the realistic synthesis of robot motion videos from given natural language instructions in Section 6.1.4.3.

The main contribution of this work is to formulate text-conditioned video generation as a universal planning strategy from which diverse behaviors can be synthesized. While such an approach departs from typical policy generation in RL, where subsequent actions to execute are directly predicted from a current state, we illustrate that UniPi exhibits notable

generalization advantages over traditional policy generation methods across a variety of domains.

6.1.2 Background

We first motivate a new abstraction, the *Unified Predictive Decision Process (UPDP)*, as an alternative to the Markov Decision Process (MDP) commonly used in RL, and then show an instantiation of a UPDP with diffusion models.

Limitations of Markov Decision Process The Markov Decision Process [61] is a broad abstraction used to formulate many sequential decision making problems. Many RL algorithms have been derived from MDPs with empirical successes [378, 118, 187], but existing algorithms are typically unable to combinatorially generalize across different environments. Such difficulty can be traced back to certain aspects of the underlying MDP abstraction:

i) The lack of a universal state interface across different control environments. In fact, since different environments typically have separate underlying state spaces, one would need to a construct a complex state representation to represent all environments, making learning difficult.

ii) The explicit requirement of a real-valued reward function in an MDP. The RL problem is usually defined as maximizing the accumulated reward in an MDP. However, in many practical applications, how to design and transfer rewards is unclear and different across environments.

iii) The dynamics model in an MDP is environment and agent dependent. Specifically, the dynamics model $T(s'|s,a)$ characterizing transition between states $(s.s')$ under action a, is explicitly dependent to the environment and action space of the agent, which can be significantly different between different agents and tasks.

Unified Predictive Decision Process These difficulties inspire us to construct an alternative abstraction for unified sequential decision making across many environments. Our abstraction, termed *Unified Predictive Decision Process (UPDP)*, exploits images as a universal interface across environments, texts as task specifiers to avoid reward design, and a task-agnostic planning module separated from environment-dependent control to enable knowledge sharing and generalization.

Formally, we define a UPDP to be a tuple $\mathcal{G} = \langle \mathcal{X}, \mathcal{C}, H, \rho \rangle$, where $\mathcal{X}$ denotes the observation space of images, $\mathcal{C}$ denotes the space of textual task descriptions, $H \in \mathcal{N}$ is a finite horizon length, and $\rho(\cdot|x_0, c) : \mathcal{X} \times \mathcal{C} \to \Delta(\mathcal{X}^H)$ is a conditional video generator. That is, $\rho(\cdot|x_o, c) \in \Delta(\mathcal{X}^H)$ is a conditional distribution over H-step image sequences determined by the first frame x_0 and the task description c. Intuitively, ρ synthesizes H-step image trajectories that illustrate possible paths for completing a target task c. For simplicity, we focus on finite horizon, episodic tasks.

Given a UPDP $\mathcal{G}$, we define a trajectory-task conditioned policy $\pi(\cdot|\{x_h\}_{h=0}^H, c) : \mathcal{X}^{H+1} \times \mathcal{C} \to \Delta(\mathcal{A}^H)$ to be a conditional distribution over H-step action sequences $\mathcal{A}^H$. Ideally, $\pi(\cdot|\{x_h\}_{h=0}^H, c)$ specifies a conditional distribution of action sequences that achieves the given trajectory $\{x_h\}_{h=0}^H$ in the UPDP $\mathcal{G}$ for the given task c. To achieve such an alignment, we will consider an offline RL scenario where we have access to a dataset of existing experience $\mathcal{D} = \{(x_i, a_i)_{i=0}^{H-1}, x_H, c\}_{j=1}^n$ from which both $\rho(\cdot|x_0, c)$ and $\pi(\cdot|\{x_h\}_{h=0}^H, c)$ can be estimated.

In contrast to an MDP, a UPDP directly models video-based trajectories and bypasses the need to specify a reward function beyond the textual task description. Since the space of video observations $\mathcal{X}^H$ and task descriptions $\mathcal{C}$ are both naturally shared across environments and easily interpretable by humans, any video-based planner $\rho(\cdot|x_0, c)$ can be conveniently *reused*, *transferred* and *debugged*. Another benefit of a UPDP over an MDP is that UPDP isolates video-based planner using $\rho(\cdot|x_0, c)$ from deferred action selection using $\pi(\cdot|\{x_h\}_{h=0}^H, c)$. This design choice isolates planning decisions from action-specific mechanisms, allowing the planner to be environment and agent agnostic.

UPDP can be understood as implicitly planning over an MDP and directly outputting the optimal trajectory under instructions. The abstraction of UPDP bypasses reward design, state extraction and explicit planning, and allows for *non-Markovian* modeling of image-based state space. However, learning a planner in UPDP requires videos and task descriptions, whereas traditional MDPs do not require such data, so whether MDP or UPDP is more suitable for a given task depends on what types of training data is available. Although the non-Markovian model and the requirement of video and text data induce additional difficulties in UPDP comparing to MDP, it is possible to leverage existing large text-video models that have been pretrained on massive, web-scale datasets to alleviate these complexities.

Diffusion Models for UPDP Let $\tau = [x_1, \ldots, x_H] \in \mathcal{X}^H$ denote a sequence of images. We leverage the significant recent advances in diffusion models for capturing the conditional distribution $\rho(\tau|x_0, c)$, which we will leverage as a text and initial-frame conditioned video generator in a UPDP. We emphasize that the UPDP formulation is also compatible with other probabilistic models, such as a variational autoencoder [379], energy-based model [97, 380], or generative adversarial network [381]. For completeness we briefly cover the core formulation at a high-level, but defer details to background references [382].

We start with an unconditional model. A continuous-time diffusion model defines a forward process $q_k(\tau_k|\tau) = \mathcal{N}(\cdot; \alpha_k\tau, \sigma_k^2 I)$, where $k \in [0, 1]$ and α_k, σ_k^2 are scalars with predefined schedules. A generative process $p(\tau)$ is also defined, which reverses the forward process by learning a denoising model $s(\tau_k, k)$. Correspondingly τ can be generated by simulating this reverse process with an ancestral sampler [35] or numerical integration [383]. In our case, the unconditional model needs to be further adapted to condition on both the text instruction c and the initial image x_0. Denote the conditional denoiser as $s(\tau_k, k|c, x_0)$. We leverage classifier-free guidance [106] and use $\hat{s}(\tau_k, k|c, x_0) = (1 + \omega)s(\tau_k, k|c, x_0) - \omega s(\tau_k, k)$ as the denoiser in the reverse process for sampling, where ω controls the strength of the text and first-frame conditioning.

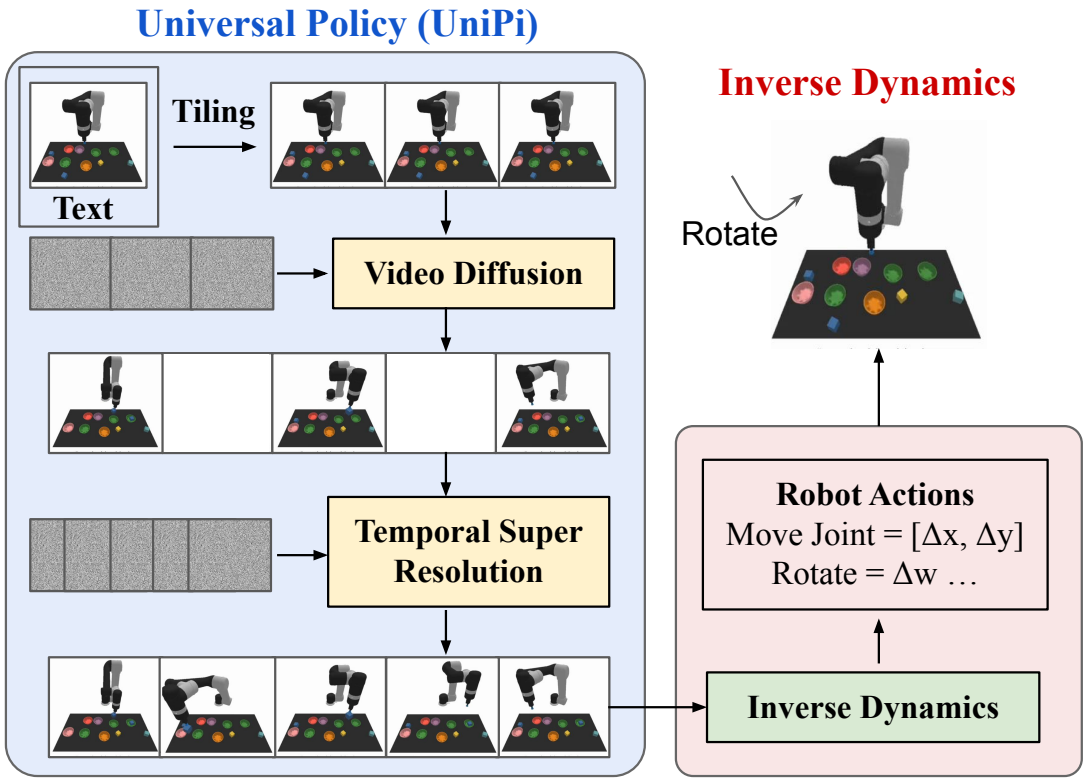


Figure 6.2: Given an input observation and text instruction, we plan a set of images representing agent behavior. Images are converted to actions using an inverse dynamics model.

6.1.3 Decision Making with Videos

Next we describe our proposed approach UniPi in detail, which is a concrete instantiation of the diffusion UPDP. UniPi incorporates each of the two main components discussed in Section 6.1.2 and shown in Figure 6.2: **(i)** a diffusion model for the universal video-based planner $\rho(\cdot|x_0, c)$, which synthesizes videos conditioned on the first frame and task descriptions; and **(ii)** a task-specific action generator $\pi(\cdot|\{x_h\}_{h=0}^{H}, c)$, which infers actions sequences from generated videos through inverse dynamics modeling.

Universal Video-Based Planner Encouraged by the recent success of text-to-video models [28], we seek to construct a video diffusion module as the trajectory planner, which can faithfully synthesize future image frames given an initial frame and textual task description. However, the desired planner departs from the typical setting in text-to-video models [384, 28] which normally generate unconstrained videos given a text description. Planning through video generation is more challenging as it requires models to both be able to generate constrained videos that start at a specified image, and then complete the target task. Moreover, to ensure valid action inference across synthesized frames in a video, the

	Seen		Novel	
Model	**Place**	**Relation**	**Place**	**Relation**
State + Transformer BC [19]	19.4 ± 3.7	8.2 ± 2.0	11.9 ± 4.9	3.7 ± 2.1
Image + Transformer BC [19]	9.4 ± 2.2	11.9 ± 1.8	9.7 ± 4.5	7.3 ± 2.6
Image + TT [5]	17.4 ± 2.9	12.8 ± 1.8	13.2 ± 4.1	9.1 ± 2.5
Diffuser [133]	9.0 ± 1.2	11.2 ± 1.0	12.5 ± 2.4	9.6 ± 1.7
UniPi (Ours)	**59.1** ± 2.5	**53.2** ± 2.0	**60.1** ± 3.9	**46.1** ± 3.0

Table 6.1: **Task Completion Accuracy on Combinatorial Environment.** UniPi generalizes well to both seen and novel combinations of language prompts in Place (e.g., place X in Y) and Relation (e.g., place X to the left of Y) tasks.

video prediction module needs to be able to track the underlying environment state across synthesized video frames.

Conditional Video Synthesis. To generate a valid and executable plan, a text-to-video model must synthesize a constrained video plan starting at the current observed image. One approach to solve this problem is to modify the underlying *test-time* sampling procedure of an unconditional model, by fixing the first frame of the generated video plan to always begin at the observed image, as done in [133]. However, we found that this performed poorly and led to subsequent frames in the video plan to deviate significantly from the original observed image. Instead, we found it more effective to explicitly train a constrained video synthesis model by providing the first frame of each video as explicit conditioning context during training.

Trajectory Consistency through Tiling. Existing text-to-video models typically generate videos where the underlying environment state changes significantly during the temporal duration [28]. To construct an accurate trajectory planner, it is important that the environment remain consistent across all time points. To enforce environment consistency in conditional video synthesis, we provide, as additional context, the observed image when denoising each frame in the synthesized video. In particular, we re-purposed a temporal super-resolution video diffusion architecture, and provided as context the conditioned visual observation tiled across time, as the opposed to a low temporal-resolution video for denoising at each timestep. In this model, we directly concatenate each intermediate noisy frame with the conditioned observed image across sampling steps, which serves as a strong signal to maintain the underlying environment state across time.

Hierarchical Planning. When constructing plans in high dimensional environments with long time horizons, directly generating a set of actions to reach a goal state quickly becomes intractable due to the exponential blow-up of the underlying search space. Planning methods

often circumvent this issue by leveraging a natural hierarchy in planning. Specifically, planning methods first construct coarse plans operating on low dimensional states and actions, which may then be refined into plans in the underlying state and action spaces. Similar to planning, our conditional video generation procedure likewise exhibits a natural temporal hierarchy. We first generate videos at a coarse level by sparsely sampled videos ("abstractions") of our desired behavior along the time axis. Then we refine the videos to represent valid behavior in the environment by super-resolving videos across time. Meanwhile, coarse-to-fine super-resolution further improves consistency via interpolation between frames.

Flexible Behavioral Modulation. When planning a sequence of actions to a given subgoal, one can readily incorporate external constraints to modulate the generated plan. Such test-time adaptability can be implemented by composing a prior $h(\tau)$ during plan generation to specify desired constraints across the synthesized action trajectory [133], which is also compatible with UniPi. In particular, the prior $h(\tau)$ can be specified using a learned classifier on images to optimize a particular task, or as a Dirac delta on a particular image to guide a plan towards a particular set of states. To train the text-conditioned video generation model, we utilize the video diffusion algorithm in [28], where pretrained language features from T5 [385] are encoded. Please see Appendix A.5.1 for the underlying architecture and training details.

Task Specific Action Adaptation Given a set of synthesized videos, we may train a small task-specific inverse-dynamics model to translate frames into a set of actions as described below.

- **Inverse Dynamics.** We train a small model to estimate actions given input images. The training of the inverse dynamics is independent from the planner and can be done on a separate, smaller and potentially suboptimal dataset generated by a simulator. **Action Execution.** Finally, we generate an action sequence given x_0 and c by synthesizing H image frames and applying the learned inverse-dynamics model to predict the corresponding H actions. Inferred actions can then be executed via *closed-loop* control, where we generate H new actions after each step of action execution (i.e., model predictive control), or via *open-loop* control, where we sequentially execute each action from the intially inferred action sequence. For computational efficiency, we use an open-loop controller in all our experiments in this work.

6.1.4 Experimental Evaluation

The focus of these experiments is to evaluate UniPi in terms of its ability to enable effective, generalizable decision making. In particular, we evaluate

(1) the ability to combinatorially generalize across different subgoals in Section 6.1.4.1,

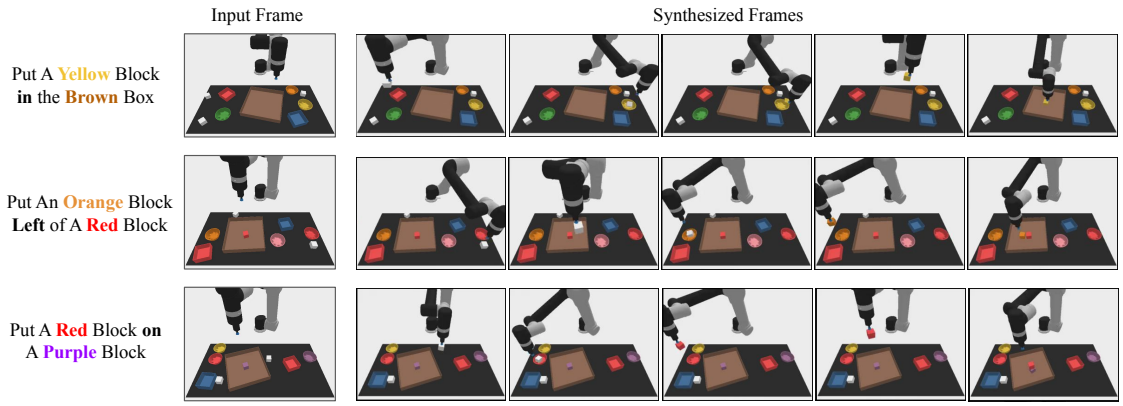


Figure 6.3: **Combinatorial Video Generation.** Generated videos for unseen language goals at test time.

(2) the ability to effectively learn and generalize across many tasks in Section 6.1.4.2,

(3) the ability to leverage existing videos on the internet to generalize to complex tasks in Section 6.1.4.3.

See experimental details in Appendix A.5.1. Additional results are given in Appendix A.5.2 and videos in the supplement.

6.1.4.1 Combinatorial Policy Synthesis

First, we measure the ability of UniPi to combinatorially generalize to different language tasks.

Setup. To measure combinatorial generalization, we use the combinatorial robot planning tasks in [386]. In this task, a robot must manipulate blocks in an environment to satisfy language instructions, i.e., put a red block right of a cyan block. To accomplish this task, the robot must first pick up a white block, place it in the appropriate bowl to paint it a particular color, and then pick up and place the block in a plate so that it satisfies the specified relation. In contrast to [386] which uses pre-programmed pick and place primitives for action prediction, we predict actions in the continuous robotic joint space for both the baselines and our approach.

We split the language instructions in this environment into two sets: one set of instructions (70%) that is seen during training, and another set (30%) that is only seen during testing. The precise locations of individual blocks, bowls, and plates in the environment are fully randomized in each environment iteration. We train the video model on 200k example videos of generated language instructions in the train set. Details of this environment can

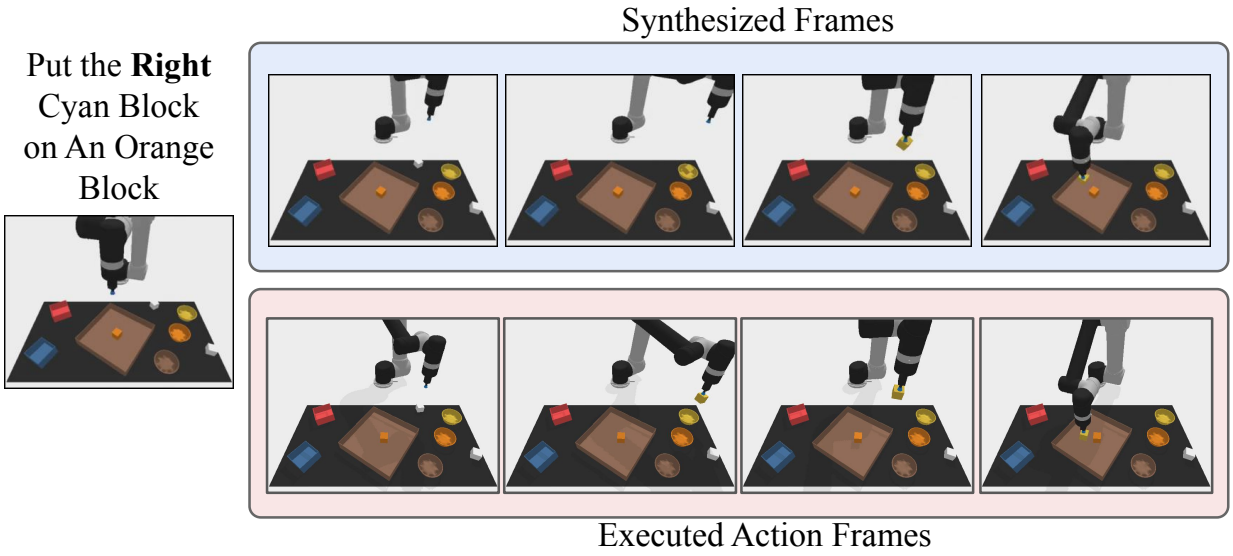


Figure 6.4: **Action Execution.** Synthesized video plans and executed actions in the simulated environment. The two video plans roughly align with each other.

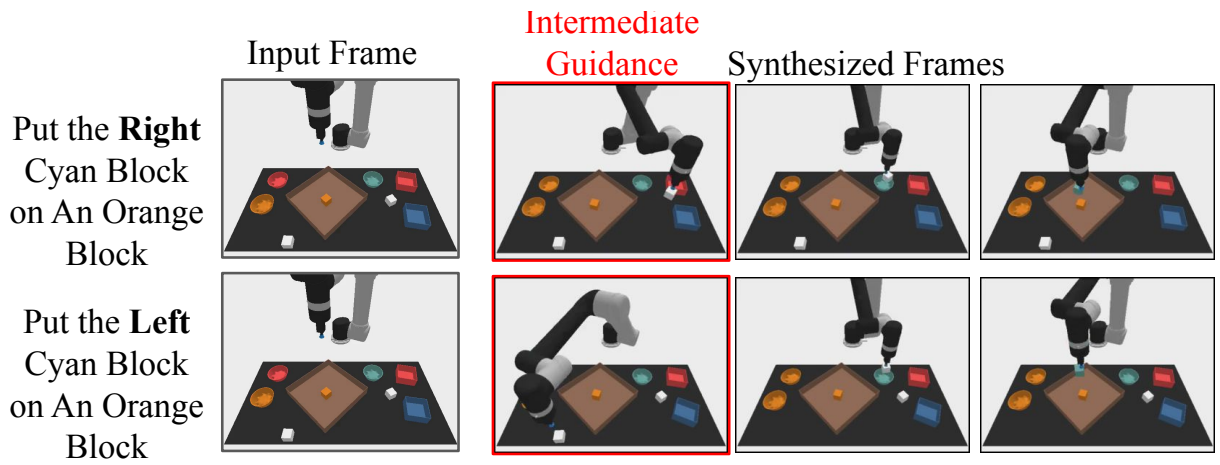


Figure 6.5: **Adaptable Planning.** By guiding test-time sampling towards a an intermediate image through fixing that intermediate frame during sampling, we can adapt our planning procedure to move a particular block.

be found in Appendix A.5.1. We constructed demonstrations of videos in this task by using a scripted agent.

Baselines. We compare the proposed approach with three separate representative approaches. First, we compare to existing work that uses goal-conditioned transformers to

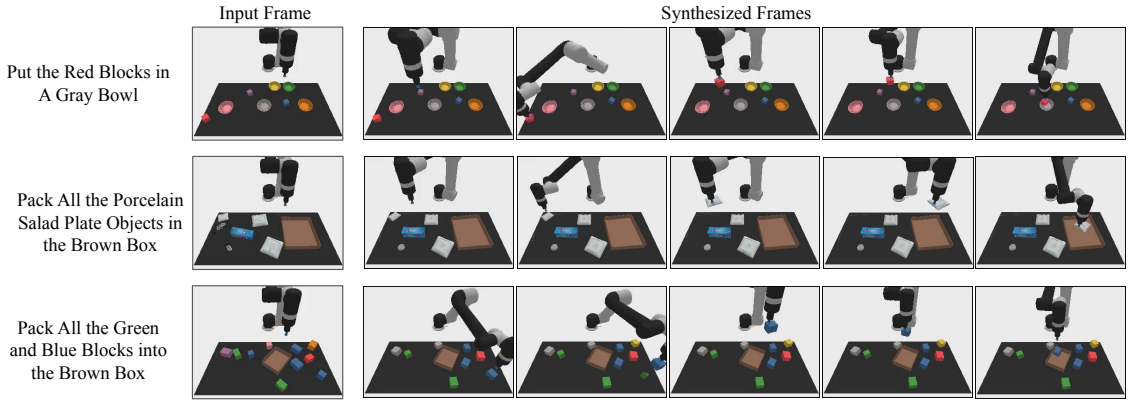


Figure 6.6: **Multitask Video Generation.** Generated video plans on different new test tasks in the multitask setting.

Frame Condition	Frame Consistency	Temporal Heirarchy	Place	Relation
No	No	No	13.2 ± 3.2	12.4 ± 2.4
Yes	No	No	52.4 ± 2.9	34.7 ± 2.6
Yes	Yes	No	53.2 ± 3.0	39.4 ± 2.8
Yes	Yes	Yes	**59.1** ± 2.5	**53.2** ± 2.0

Table 6.2: **Task Completion Accuracy Ablations.** Each component of UniPi improves its performance.

learn across multiple environments, where goals can be specified as episode returns [3], expert demonstrations [49], or text and images [19]. To represent these baselines, we construct a transformer behavior cloning (BC) agent to predict the subsequent action to execute given the task description and either the visual observation (Image + Transformer BC) or the underlying robot joint state (State + Transformer BC). Second, given that our approach regresses a sequence of actions to execute, we further compare with transformer models that regress a sequence of future actions to execute, similar to the goal-conditioned behavioral cloning of the Trajectory Transformer [5] (Image + TT). Finally, to highlight the importance of the video-as-policy approach, we compare UniPi with learning a diffusion process that, conditioned on an image observation, directly infers future robot actions in the joint space (as opposed to diffusing future image frames), corresponding to [133, 105]. For both our method and each baseline, we condition the policy on encoded language instructions using pretrained T5 embeddings. Note that in this setting, existing offline reinforcement learning baselines are not directly applicable as we do not have access to the reward functions in the environment.

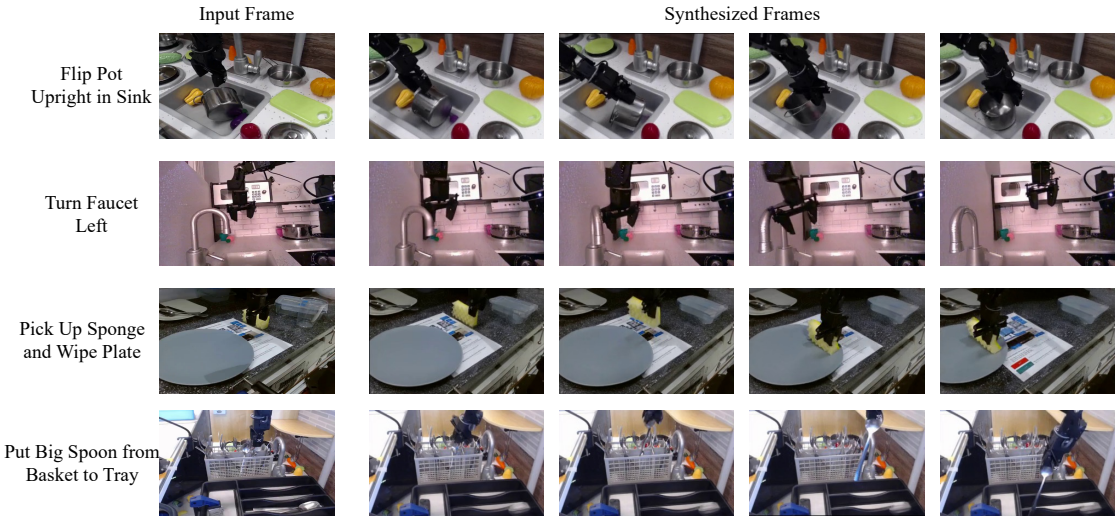


Figure 6.7: **High Fidelity Plan Generation.** UniPi can generate high resolution video plans across different language prompts.

Metrics. To compare UniPi with baselines, we measure final task completion accuracy across new instances of the environment and associated language prompts. We subdivide the evaluation along two axes: **(1)** whether the language instruction has been seen during training and **(2)** whether the language instruction specifies placing a block in relation to some other block as opposed to direct pick-and-place.

Combinatorial Generalization. In Table 6.1, we find that UniPi generalizes well to both seen and novel combinations of language prompts . We illustrate our action generation pipeline in Figure 6.4 and different generated video plans using our approach in Figure 6.3.

Ablations. In Table 6.2, we ablate UniPi on seen language instructions and in-relation-to tasks. Specifically, we study the effect of conditioning the video generative model on the first observation frame (frame condition), tiling the observed frame across timesteps (frame consistency) and super-resolving video generation across time (temporal hierarchy). All components of UniPi are crucial for good performance. In settings where frame consistency is not enforced, we provide a zeroed out image as context to the non-start frames in a video.

Adaptability. We next assess the ability of UniPi to adapt at test time to new constraints. In Figure 6.5, we illustrate the ability to construct plans which color and move one particular block to a specified geometric relation.

Model	Place Bowl	Pack Object	Pack Pair
State + Transformer BC	9.8 ± 2.6	21.7 ± 3.5	1.3 ± 0.9
Image + Transformer BC	5.3 ± 1.9	5.7 ± 2.1	7.8 ± 2.6
Image + TT	4.9 ± 2.1	19.8 ± 0.4	2.3 ± 1.6
Diffuser	14.8 ± 2.9	15.9 ± 2.7	10.5 ± 2.4
UniPi (Ours)	**51.6** ± 3.6	**75.5** ± 3.1	**45.7** ± 3.7

Table 6.3: **Task Completion Accuracy on Multitask Environment.** UniPi generalizes well to new environments when trained on a set of different multi-task environments.

6.1.4.2 Multi-Environment Transfer

We next evaluate the ability of UniPi to effectively learn across a set of different tasks and generalize, at test time, to a new set of unseen environments.

Setup. To measure multi-task learning and transfer, we use the suite of language guided manipulation tasks from [162]. We train our method using demonstrations across a set of 10 separate tasks from [162], and evaluate the ability of our approach to transfer to 3 different test tasks. Using a scripted oracle agent, we generate a set of 200k videos of language execution in the environment. We report the underlying accuracy in which each language instruction is completed.

Baselines. We use the same baseline methods as in Section 6.1.4.1. While our environment setting is similar to that of [162], this method is not directly comparable to our approach, as CLIPort abstracts actions to the existing primitives of pick and place as opposed to using joint space of a robot. CLIPort is also designed to solve the significantly simpler problem of inferring only the poses upon which to pick and place objects (with no easy manner to adapt to our setting).

Multitask Generalization. In Table 6.3 we present results of our approach and baselines across new tasks. Our approach is able to generalize and synthesize new videos and decisions of different language tasks, and can generate videos consisting of picking different kinds of objects and different colored objects. We further present video visualizations of our approach in Figure 6.6.

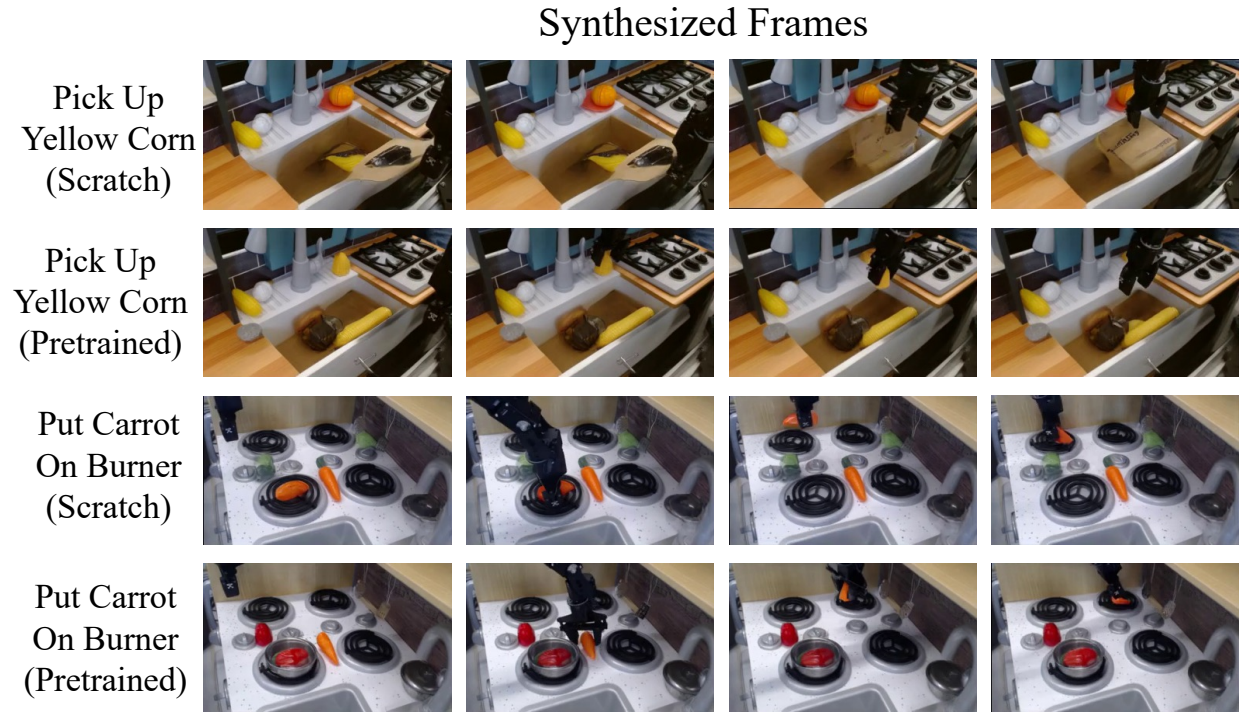


Figure 6.8: **Pretraining Enables Combinatorical Generalization.** Using internet pretraining enables UniPi to synthesize videos of tasks not seen during training. In contrast, a model trained from scratch incorrectly generates plans of different tasks.

6.1.4.3 Real World Transfer

Finally we evaluate the extent to which UniPi can generalize to real world scenarios and construct complex behaviors by leveraging widely available videos on the internet.

Setup. Our training data consists of an internet-scale pretraining dataset and a smaller real-world robotic dataset. The pretraining dataset uses the same data as [28], which consists of 14 million video-text pairs, 60 million image-text pairs, and the publicly available LAION-400M image-text dataset. The robotic dataset is adopted from the Bridge dataset [387] with 7.2k video-text pairs, where we use the task IDs as texts. We partition the 7.2k video-text pairs into train (80%) and test (20%) splits. We pretrain UniPi on the pretraining dataset followed by finetuning on the train split of the Bridge data.

Video Synthesis. We are particularly interested in the effect of pretraining on internet-scale video data that is not robotic specific. We report the CLIP scores, FIDs, and VIDs (averaged across frames and computed on 32 samples) of UniPi trained on Bridge data, with and without pretraining. As shown in Table 6.4, UniPi with pretraining achieves

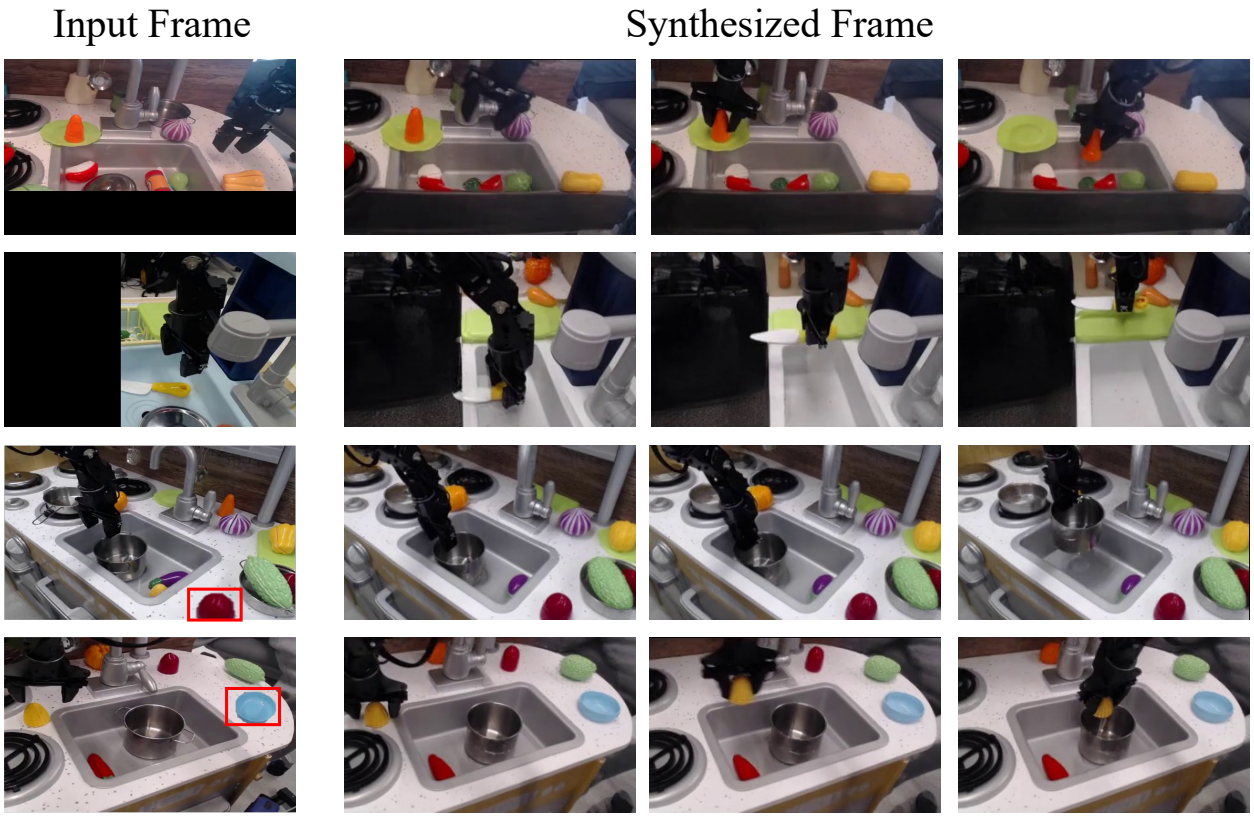


Figure 6.9: **Robustness to Background Change.** UniPi learns to be robust to changes of underlying background, such as black cropping or the addition of photo-shopped objects.

Model (24x40)	**CLIP Score ↑**	**FID ↓**	**FVD ↓**
No Pretrain	24.43 ± 0.04	17.75 ± 0.56	288.02 ± 10.45
Pretrain	**24.54** ± 0.03	**14.54** ± 0.57	**264.66** ± 13.64

Table 6.4: **Video Generation Quality of UniPi on Real Environment.** The use of existing data on the internet improves video plan predictions under all metrics considered.

significantly better FID and FVD and a marginally better CLIP score than UniPi without pretraining, suggesting that pretraining on non-robot data helps with generating plans for robots. Interestingly, UniPi without pretraining often synthesizes plans that fail to complete the task (Figure 6.7), which is not well reflected in the CLIP score, suggesting the need for better generation metrics for control-specific tasks.

Generalization. We find that internet-scale pretraining enables UniPi to generalize to novel task commands and scenes in the test split not seen during training, whereas UniPi trained only on task-specific robot data fails to generalize. Specifically, Figure 6.8 shows the generalization results of novel task commands that do not exist in the Bridge dataset. Additionally, UniPi is relatively robust to background changes such as black cropping or the addition of photo-shopped objects as shown in Figure 6.9.

6.1.5 Related Work

Learning Generative Models of the World. Models trained to generate environment rewards and dynamics that can serve as "world models" for model-based reinforcement learning and planning have been recently scaled to large-scale architectures developed for vision and language [118, 5, 388, 121]. These works separate learning the world model from planning and policy learning, and arguably present a mismatch between the generative modeling objective of the world and learning optimal policies. Additionally, learning a world model requires the training data to be in a strict state-action-reward format, which is incompatible with the largely available datasets on the internet, such as YouTube videos. While methods such as VPT [110] can utilize internet-scale data through learning an inverse dynamics model to label unlabled videos, an inverse dynamics model itself does not support model-based planning or reinforcement learning to further improve learned policies beyond imitation learning. Our text-conditioned video policies can be seen as jointly learning the world model and conducting hierarchical planning simultaneously, and is able to leverage widely available datasets that are not specifically designed for sequential decision making.

Diffusion Models for Decision Making. Diffusion models have recently been applied to different decision making problems [133, 105, 389, 390, 391, 392]. Most similar to this work, [133] trained an unconditional diffusion model to generate trajectories consisting of joint-based states and actions, and used a separately trained reward model to select generated plans. [105] on the other hand, trained a conditional diffusion model to guide behavior synthesis from desired rewards, constraints or agent skills. Unlike both works, which learn task-specific policies from scratch, our approach of text-condition video generation as a universal policy can leverage internet-scale knowledge to learn generalist agents that can be deployed to a variety of novel tasks and environments. Additionally, [393] applied web-scale text-conditioned image diffusion to generate a goal image to condition a policy on, whereas our work uses video diffusion to learning universal policies directly.

Learning Generalist Agents. Inspired by the large-scale pretraining success of vision and language domains, large-scale sequence and image models have recently been applied to learning generalist decision making agents [49, 3, 9]. However, these generalist agents can only operate under environments with the same state and action spaces (e.g., Atari games) [3, 9], or require studious tokenization [49] that might seem unnatural in reinforce-

ment learning settings where different environments have distinct state and actions spaces. Another downside of using customized tokens for control is the inability to directly utilize knowledge from pretrained vision and language models. Our approach, on the other hand, uses text and images as universal interfaces for policy learning so that the knowledge from pretrained vision and language models can be preserved. Our choice of diffusion as opposed to autoregressive sequence modeling also enables long-term and hierarchical planning.

Learning Text-Conditioned Policies. There has been a growing amount of work using text commands as a way to learn multi-task and generalist control policies [394, 143, 19, 156, 149, 153]. Different from our framing of video-as-policies, existing work directly trains a language-conditioned control policy in the action space of some specific robot, leaving cross-morphology multi-environment learning of generalist agents as an unsolved problem. We believe this work is the first to propose images as a universal state and action space to enable broad knowledge transfer across environments, tasks, and even between humans and robots.

6.2 Video Generation as Real-World Simulators

6.2.1 Introduction

Generative models trained on internet data can now produce highly realistic text [395],image [396], and video [28]. Perhaps the ultimate goal of generative models is to be able to simulate the visual effects of a wide variety of actions, from how cars are driven on a street to how furniture and meals are prepared. With a real-world simulator, humans can "interact" with diverse scenes and objects, robots can learn from simulated experience without risking physical damage, and a vast amount of "real-world" like data can be simulated to train other types of machine intelligence.

One roadblock to building this simulator lies in the datasets — different datasets cover different information that have to be brought together to simulate realistic experience. For instance, paired text-image data from the internet contains rich scenes and objects but little movement [397, 398], video captioning and question answering data contain rich high-level descriptions but little low-level movement detail [399, 400], human activity data contains rich human action but little mechanical motion [401, 402], and robotics data contains rich robot action but are limited in quantity [403, 404]. Since different datasets are curated by different industrial or research communities for different purposes, divergence in information is natural and hard to overcome, posing difficulties to a real-world simulator that seeks to capture all visual aspects of the world.

In this work, we propose to combine a wealth of data in a conditional video generation framework to instantiate a universal simulator (UniSim)[1]. Under a unified action-in-video-

[1]Note that by "universal", we mean the model can simulate through the unified interface of actions and videos, as opposed to being able to simulate everything. Sound, for instance, is not being simulated.

out interface, the simulator enables rich interaction through fine-grained motion control of otherwise static scenes and objects. To support long-horizon repeated interactions, we formulate the simulator as an *observation prediction model* that can be rolled out autoregressively to support consistent simulation across video generation boundaries.

While the potential applications of the simulator are broad, we demonstrate three specific use cases. We first show how the simulator enables a vision-language policy to perform long-horizon goal-conditioned tasks through hindsight relabeling of simulated experience [405]. In addition to learning high-level vision-language policies, we illustrate how the simulator can enable learning low-level control policies by leveraging model-based reinforcement learning (RL) [406]. Both the high-level vision-language policy and the low-level control policy, while trained purely in simulation, can generalize to real robot settings. This is enabled by using the simulator that is nearly visually indistinguishable from the real world, achieving one step towards bridging the sim-to-real gap in embodied learning [407]. Furthermore, we can simulate rare events where data collection is expensive or dangerous (e.g., crashes in self-driving cars). Such simulated videos can then be used to improve other machine intelligence such as rare event detectors, suggesting broad applications of UniSim beyond embodied learning. The main contributions can be summarized as follows:

- We take the first step toward building a universal simulator of real-world interaction by combining diverse datasets rich in along different dimensions — e.g., objects, scenes, actions, motions, language, and motor controls — in a unified action-in-video-out generative framework.
- We formulate the action-in-video-out framework as an *observation prediction model* conditioned on fini
- We illustrate how the simulator can enable both high-level language policies, low-level control policies, and video captioning models to generalize to the real world when trained purely in simulation, thereby bridging the sim-to-real gap.

6.2.2 Learning an Interactive Real-World Simulator

We define a simulator of the real world as a model that, given some state of the world (e.g., an image frame), can take in some action as input, and produce the visual consequence of the action (in the form of a video) as output. Learning such a simulator is hard, since different actions have different formats (e.g.,language instructions, robot controls, camera movements) and videos have different frame rates. Nevertheless, we propose specific strategies for processing each type of data to unify the action space and align videos of variable lengths to actions in Section 6.2.2.1. With a unified action space, we then train an action-conditioned video generation model to fuse information across datasets through a universal interface relating actions to videos in Section 6.2.2.2.

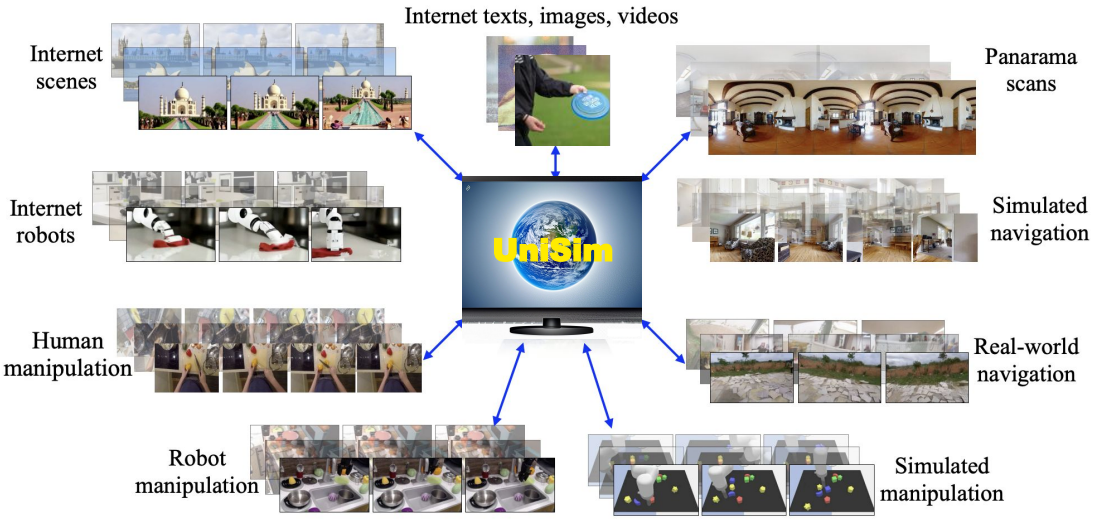


Figure 6.10: **A universal simulator (UniSim).** The simulator of the real-world learns from broad data with diverse information including objects, scenes, human activities, motions in navigation and manipulation, panorama scans, and simulations and renderings.

6.2.2.1 Orchestrating Diverse Datasets

Below, we highlight diverse information in different datasets and propose ways to process actions into a common format (see all datasets used to train UniSim in Appendix A.6.2):

- **Simulated execution and renderings.** While annotating actions for real-world videos is expensive, simulation engines such as Habitat [408] can render a wide variety of actions. We use datasets previously collected from these simulators, i.e., Habitat object navigation with HM3D [18] and Language Table Data from [409] to train UniSim. We extract text descriptions as actions when available. For simulated continuous control actions, we encode them via language embeddings and concatenate the text embeddings with discretized control values.

- **Real robot data.** An increasing amount of video data of real-robot executions paired with task descriptions such as the Bridge Data [387] and data that enabled RT-1 and RT-2 [19] are becoming increasingly available. Despite low-level control actions often being different across robots, the task descriptions can serve as high-level actions in UniSim. We further include discretize continuous controls actions when available similar to simulated robotics data.

- **Human activity videos.** Rich human activity data such as Ego4D [402], EPIC-KITCHENS [410], and Something-Something V2 [411] have been curated. Different from low-level robot controls, these activities are high-level actions that humans take to interact with the world.

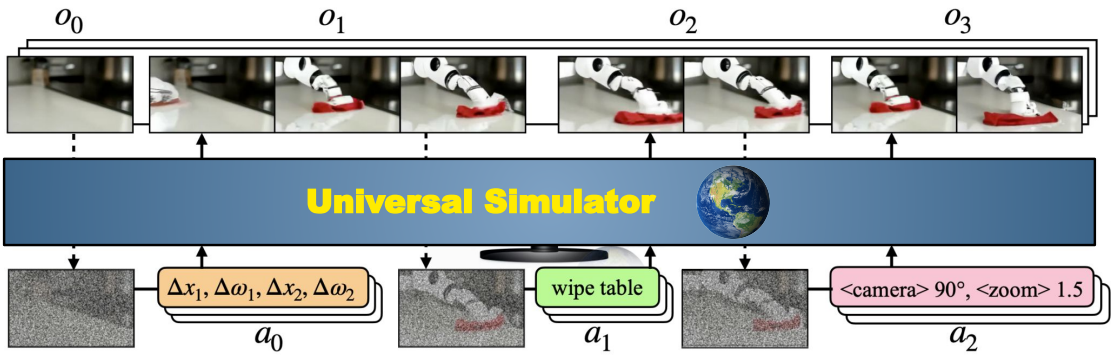


Figure 6.11: **Training and inference of UniSim.** UniSim is a video diffusion model trained to predict the next (variable length) set of observation frames (o_t) given observations from the past (e.g., o_{t-1}) and action input a_{t-1}. UniSim can handle temporally extended actions in various modalities such as motor controls ($\Delta x_1, \Delta\omega_1, \Delta x_2, ...$), language descriptions ("wipe table"), and actions extracted from camera motions and other sources. Each dotted arrow indicates concatenating the initial noise sample for the next video segment with the previous frame.

But these actions are often provided as labels for video classification or activity recognition tasks [411]. In this case, we convert the video labels into text actions. In addition, we subsample the videos to construct chunks of observations at a frame rate that captures meaningful actions.

- **Panorama scans.** There exists a wealth of 3D scans such as Matterport3D [412]. These static scans do not contain actions. We construct actions (e.g., turn left) by truncating panorama scans and utilize the information of camera poses between two images.
- **Internet text-image data.** Paired text-image datasets such as LAION [413] contain static images of a variety of objects without actions. However, the captions often contain motion information such as "a person *walking*". To use image data in UniSim, we treat individual images as single-frame videos and image captions as actions.

For each of these datasets, we process text tokens into continuous representations using T5 language model embeddings [385] concatenated with low-level actions such as robot controls. This serves as the final unified action space of our simulator.

6.2.2.2 Simulating Long-Horizon Interactions through Observation Prediction

With observations from different environments that have been converted to videos, and actions of different formats that have been converted to continuous embeddings, we can formulate interactions with many real-world environments as interacting with a universal simulator. We then formulate the universal simulator as an *observation prediction model*

that predicts observations conditioned on actions and previous observations as shown in Figure 6.11. We finally show that this observation prediction model can be parametrized using video diffusion.

Simulating Real-World Interactions. We define an observation space $\mathcal{O}$ and an action space A which capture the videos and actions described in Section 6.2.2.1. At a specific interactive step t, an agent, having observed a set of history frames $h_{t-1} \in O$, decides on some temporally extended action $a_{t-1} \in A$, which can be resolved into a sequence of low-level robot commands to be executed in the real world. During the execution, the next set of video frames $o_t \in O$ are captured from the real world. The goal of a simulator is to predict o_t from h_{t-1} and a_{t-1}. We can formulate this prediction problem as learning an *observation prediction* model $p(o_t|h_{t-1}, a_{t-1})$. While an ideal predictive model should condition on all information of the past, i.e., $(o_0, a_0 \ldots, a_{t-2}, o_{t-1})$, through some recurrent state, we found conditioning on a finite set of frames (e.g., frames from the most recent interaction, o_{t-1}) greatly simplifies the modeling problem. To simulate long interactions, we can sample from the observation prediction model $p(o_t|h_{t-1}, a_{t-1})$ autoregressively conditioned on the previously sampled observations. One advantage of this observation prediction model is that the simulator stays the same across all tasks and can be used in combination with any reward function, which can be separately learned. The learned reward function can then be used to optimize policies $\pi(a_t|h_t)$ using existing decision making algorithms such as planning and RL, as we will illustrate in Section 6.2.4 and Section 6.2.5.

Parametrizing and Training the Simulator. We parametrize $p(o_t|h_{t-1}, a_{t-1})$ using diffusion models [33, 35] as an instantiation of UniSim outlined in Figure 6.11. Specifically, the reverse process learns a denoising model $\epsilon_\theta(o_t^{(k)}, k|h_{t-1}, a_{t-1})$ that, conditioned on the history, generates the next observationfrom initial noise samples using K denoising steps. In practice, we only use previous video frames and omit previous actions as history, and concatenate previous video frames with initial noise samples $o_t^{(K)} \sim \mathcal{N}(0, I)$ channelwise to serve as conditional inputs to the denoising model. To condition on an action a_{t-1}, we leverage classifier-free guidance [106]. The final $\overline{\mathcal{T}}(o_t|h_{t-1}, a_{t-1})$ is parametrized by the variance schedule:

$$\epsilon_\theta(o_t^{(k)}, k|h_{t-1}, a_{t-1}) = (1+\eta)\epsilon_\theta(o_t^{(k)}, k|h_{t-1}, a_{t-1}) - \eta\epsilon_\theta(o_t, k|h_{t-1}), \tag{6.1}$$

where η controls action conditioning strength. With this parametrization, we train ϵ_θ by minimizing

$$\mathcal{L}_{\text{MSE}} = \left\| \epsilon - \epsilon_\theta\left(\sqrt{1-\beta^{(k)}}o_t + \sqrt{\beta^{(k)}}\epsilon,\ k \middle| h_{t-1}, a_{t-1}\right)\right\|^2,$$

where $\epsilon \sim \mathcal{N}(0, I)$, and $\beta^{(k)} \in \mathbb{R}$ are a set of K different noise levels for each $k \in [1, K]$. Given the learned ϵ_θ, an observation o_t can be generated by sampling from the initial distribution $o_t^{(K)} \sim \mathcal{N}(0, I)$ and iteratively denoising according to the following process for k from K to 0

$$o_t^{(k-1)} = \alpha^{(k)}(o_t^{(k)} - \gamma^{(k)}\epsilon_\theta(o_t^{(k)}, k|h_{t-1}, a_{t-1})) + \xi, \quad \xi \sim \mathcal{N}\big(0, \sigma_k^2 I\big), \tag{6.2}$$

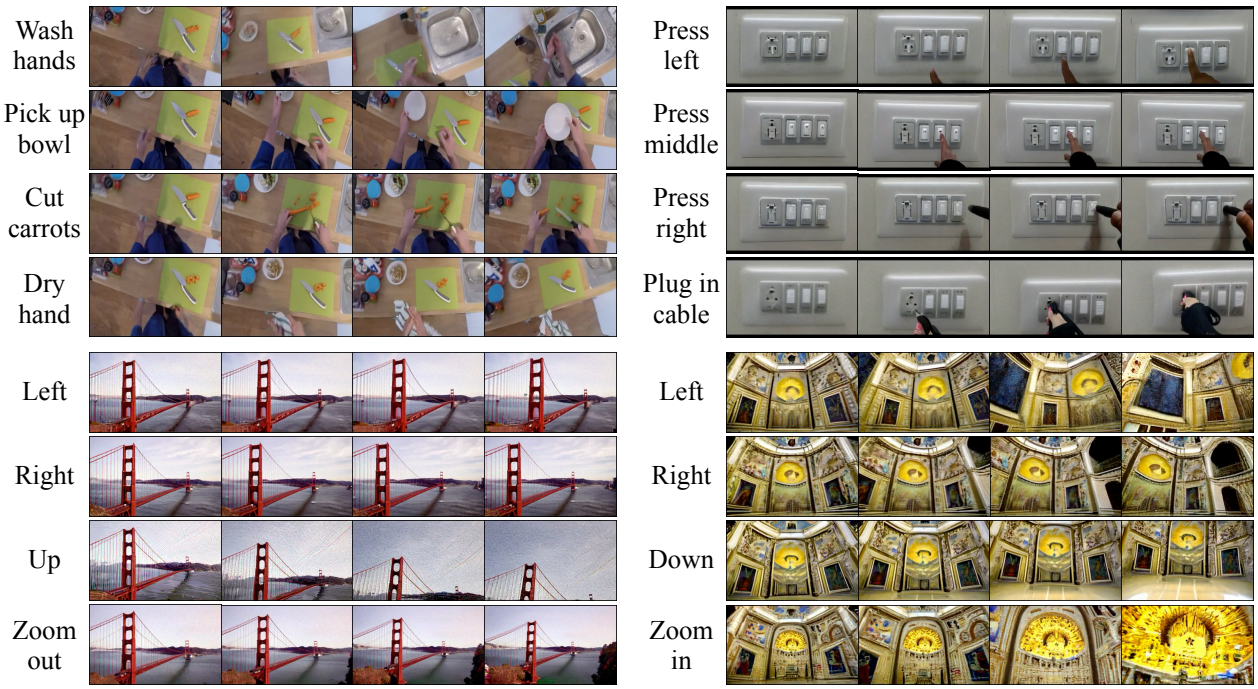


Figure 6.12: **Action-rich simulations.** UniSim can support manipulation actions such as "cut carrots", "wash hands", and "pickup bowl" from the same initial frame (top left) and other navigation actions.

where $\gamma^{(k)}$ is the denoising step size, $\alpha^{(k)}$ is a linear decay on the current denoised sample, and σ_k is a time varying noise level that depends on $\alpha^{(k)}$ and $\beta^{(k)}$.

Architecture and Training. We use the video U-Net architecture to implement UniSim by employing interleaved temporal and spatial attention and convolution layers in both the downsampling and upsampling passes. For history conditioning, we replicate the conditioning frames at all future frame indices, and concatenate the conditioning frames with the noise sample for each of the future frame to serve as input to the U-Net. UniSim model has 5.6B parameters and requires 512 TPU-v3 and 20 days to train on all data. See more details in Appendix A.6.3

6.2.3 Simulating Real-World Interactions

We now demonstrate emulating real-world manipulation and navigation environments by simulating both action-rich and long-horizon interactions for both humans and robots.

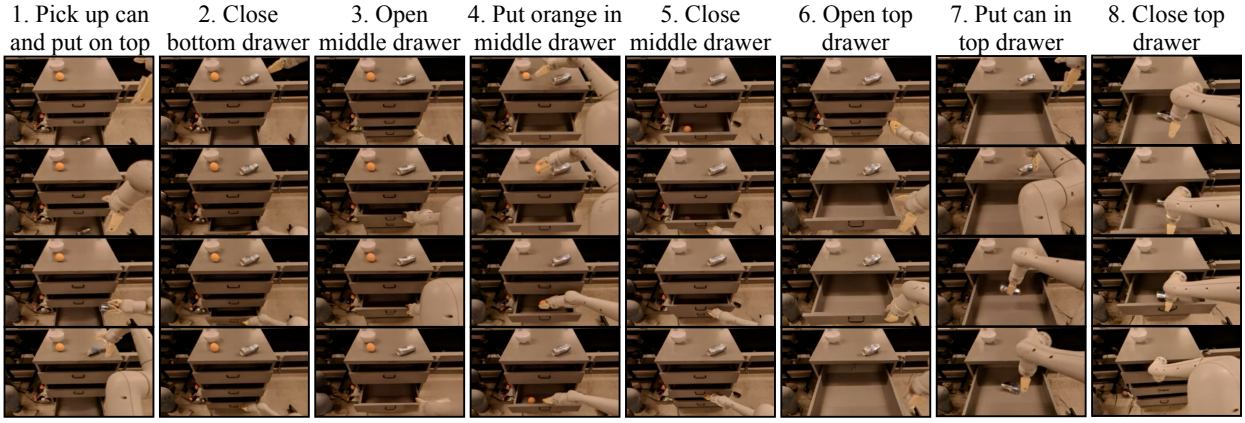


Figure 6.13: **Long-horizon simulations.** UniSim sequentially simulates 8 interactions autoregressively. The simulated interactions maintain temporal consistency across long-horizon interactions, correctly preserving objects and locations (can on counter in column 2-7, orange in drawer in column 4-5).

6.2.3.1 Action-Rich, Long-Horizon, and Diverse Interactions

Action-Rich Simulation. We first demonstrate action-rich interactions through natural language actions. Figure 6.12 shows simulation of human manipulation and navigation starting from the same initial observation (left-most column). We can instruct a person in the initial frame to perform various kitchen tasks (top left), press different switches (top right), or navigate scenes (bottom). The model only trained on generic internet data, without action-rich manipulation data such as EPIC-KITCHENS [410], fails to simulate action-rich manipulations (Appendix A.6.6).

Long-Horizon Simulation. Next, we illustrate 8 *sequential* interactions in Figure 6.13. We condition the simulation of each interaction on previous observations and new language action as described in Section 6.2.2.2. UniSim successfully preserves objects manipulated by previous instructions (e.g., the orange and can are preserved in the drawers in Columns 4, 5, 7, 8 after being put in the drawers). See additional long-horizon interactions in Appendix A.6.1.

Diversity and Stochasticity in the Simulator. UniSim can also support highly diverse and stochastic environment transitions, e.g., diverse objects being revealed after removing the towel on top (Figure 6.14 left), diverse object colors and locations (cups and pens in Figure 6.14 right), and real-world variabilities such as change in camera angles. Flexibility in diffusion models promotes simulation of highly stochastic environments that cannot be controlled by actions, so that a policy can learn to only control the controllable part [104].

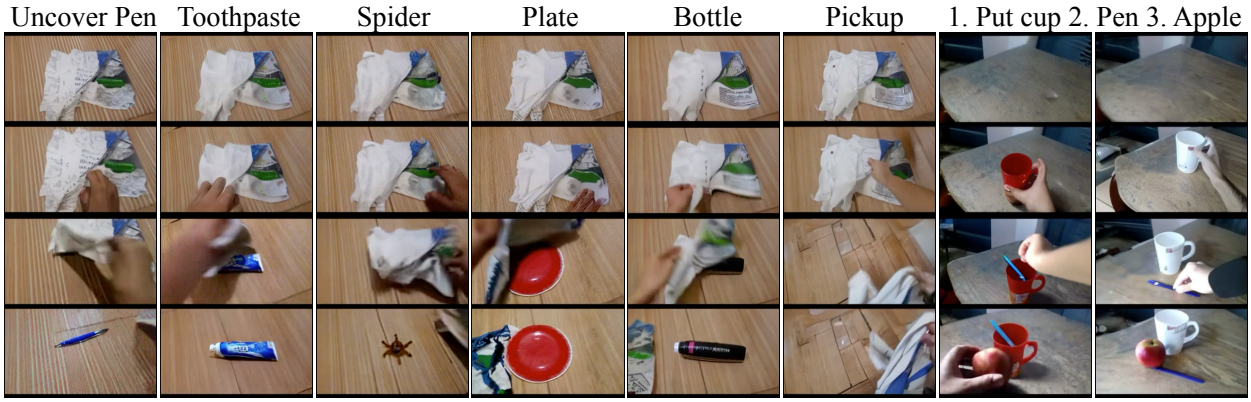


Figure 6.14: **Diverse and stochastic simulations.** On the left, we use text to specify the object being revealed by suffixing "uncovering" with the object name. On the right, we only specify "put cup" or "put pen", and cups and pens of different colors are sampled as a result of the stochastic sampling process during video generation.

Condition	FID ↓	FVD ↓	IS ↑	CLIP ↑
1 frame	59.47	315.69	3.03	22.55
4 distant	34.89	237	3.43	22.62
4 recent	**34.63**	**211.3**	**3.52**	**22.63**

Table 6.5: **Ablations of history conditioning** using FVD, FID, and Inception score, and CLIP score on Ego4D. Conditioning on multiple frames is better than on a single frame, and recent history has an edge over distant history.

Figure 6.15: **Simulations of low-data domains** using the Habitat object navigation using HM3D dataset [18] with only 700 training examples. Prefixing language actions with dataset identifier leads to video samples that complete the action (top).

6.2.3.2 Ablation and Analysis

Frame Conditioning Ablations. We ablate over choices of past frames to condition on using a validation split of the Ego4D dataset [402], which contains egocentric movement requiring proper handling of observation history. We compare UniSim conditioned on different numbers of past frames in Table 6.5. Conditioning on 4 frames is better than conditioning on a single frame, but conditioning on history that is too far in the past (4 frames with exponentially increasing distances) can hurt performance. Increasing the number of conditioning frames beyond 4 did not further improve performance on Ego4D, but it could be helpful for applications that require memory from distant past (e.g., navigation for retrieval).

Simulating Low-Data Domains. During joint training of UniSim on diverse data, we found that naïvely combining datasets of highly varying size can result in low generation

quality in low-data domains. While we can increase the weight of these domains in the data mixture during training, we found that attaching a domain identifier such as the name of the dataset to the actions being conditioned on improves generation quality in low-data domains, as shown in Figure 6.15. While such domain identifier improves in-distribution generation quality, we found domain-specific identifiers to hurt generalization to other domains, and should only be applied with the test domain is in distribution of the training domain.

6.2.4 UniSim for Long-Horizon Planning

We now demonstrate how UniSim can be used to train other types of machine intelligence such as vision-language policies, RL agents, and vision-language models through simulating highly realistic experiences.

Language models and vision language models (VLM) have recently been used as policies that can operate in image or text based observation and action spaces [147, 414, 415]. One major challenge in learning such agents lies in the need for large amounts of language action labels. The labor intensity in data collection only increases as tasks increase in horizon and complexity. UniSim can generate large amounts of training data for VLM policies through hindsight relabeling.

Setup and Baseline. We use data from the Language Table environment [156] for learning geometric rearrangements of blocks on a table. We train an image-goal conditioned VLM policy to predict language instructions and the motor controls from the start and goal images using the PALM-E architecture [414] (See data and model details in Appendix A.6.4). For the baseline, the goal is set to the last frame of the original short-horizon trajectories. During each evaluation run, we set the long-horizon goal by modifying the location of 3-4 blocks, and measure the blocks' distance to their goal states after executing 5 instructions using the VLM policy. We define the reduction in distance to goal (RDG) metric as

$$\text{RDG} = \frac{\|s_0 - s_{\text{goal}}\|_2 - \|s_T - s_{\text{goal}}\|_2}{\|s_0 - s_{\text{goal}}\|_2}, \tag{6.3}$$

where s_T represents the underlying block locations after executing the policy, s_0 and s_{goal} represents the initial and goal block locations.

Generating Hindsight Data with the Simulator. To use the simulator for long-horizon tasks, we draw inspiration from hindsight relabeling [416]. Specifically, we create a total of 10k long-horizon trajectories from the simulator by doing rollouts in the simulator 3-5 times per trajectory, where each rollout corresponds to one scripted language instruction. We then use the final frame from each long-horizon rollout as a goal input and the scripted language instructions as supervision for training the VLM policy.

Results on Real-Robot Evaluation. Despite the VLM policy only being trained on simulated data, it is able to produce effective high-level language actions given an initial and goal image from the real Language Table domain where the data for training the simulator was collected. The simulator can simulate video trajectories from the initial real observation,

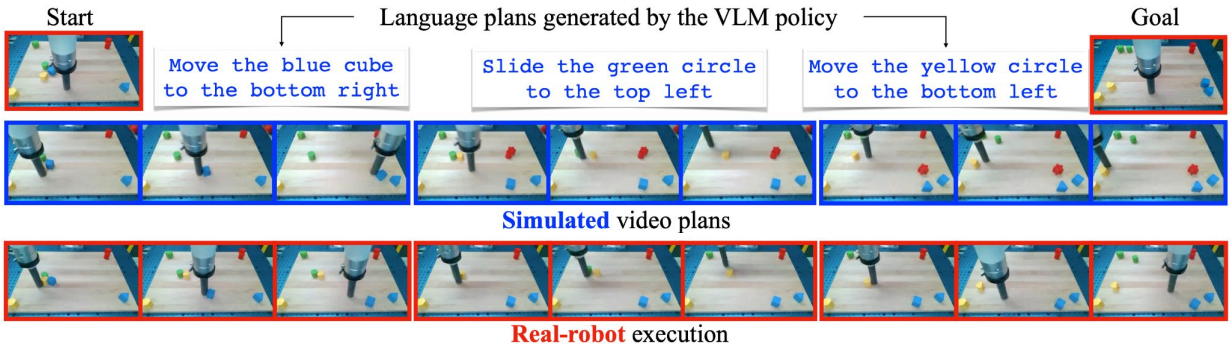


Figure 6.16: **Long-horizon simulation.** A VLM poliy generates high-level language actions (first row) which are executed in the simulator (middle row) similar to how they are executed in the real world (bottom row) using the Language Table robot. The VLM trained on data from the simulator complete long-horizon tasks by successfully moving three blocks (blue, green, yellow) to match their target location in the goal image.

from which robot actions are recovered using an inverse dynamics model and executed on the real robot. Figure 6.16 shows that the language actions produced by the VLM, the generated videos from the simulator according to the language actions, and the executions on the real robot. We see that the simulated video trajectory is successfully translated to robot actions in the real world. See additional results from the long-horizon VLM policy in Appendix A.6.1.

Results on Simulated Evaluation. In addition to testing the language instructions and simulated video by converting video trajectory into robot actions executed on the real robot, we also conduct simulator based evaluation to compare the reduction in distance to goal (RDG) of the VLM policy using generated long-horizon data to using the original short-horizon data in Table 6.6. The VLM trained using long-horizon generated data performs 3-4 times better than using the original data in completing long-horizon goal-conditioned tasks.

6.2.5 Real-World Simulator for Reinforcement Learning

Reinforcement learning (RL) has achieved superhuman performance on difficult tasks such as playing Go and Atari games [417, 324], but has limited real world applications due, among other reasons, to the lack of a realistic environment simulator [418]. We investigate whether the simulator can enable effective training of RL agents by providing the agent with a realistic simulator that can be accessed in parallel.

Setup. We finetune the PaLI 3B vision-language model [148] to predict low-level control actions (joint movements in $\Delta x, \Delta y$) from an image observation and a task description (e.g., "move the blue cube to the right") using behavioral cloning (BC) to serve as the low-level control policy and the baseline, which we call the vision-language-action (VLA) policy similar

	RDG (moved)	RDG (all)		Succ. rate (all)	Succ. rate (pointing)
VLM-BC	0.11 ± 0.13	0.07 ± 0.11	VLA-BC	0.58	0.12
Simulator-Hindsight	**0.34** ±0.13	**0.34** ± 0.13	Simulator-RL	**0.81**	**0.71**

Table 6.6: **Evaluation of long-horizon actions.** Reduction in distance to goal (RDG) defined in Equation 6.3 across 5 evaluation runs of VLM trained using simulated long-horizon data (bottom row) compared to VLM trained on original short-horizon data (top row). Using the simulator performs much better both in RGD of moved blocks (left) and RGD in all blocks (right).

Table 6.7: **Evaluation of RL policy.** Percentage of successful simulated rollouts (out of 48 tasks) using the VLA policy with and without RL finetuning on Language Table (assessed qualitatively using video rollouts in the simulator). Simulator-RL improves the overall performance, especially in pointing-based tasks which contain limited expert demonstrations.

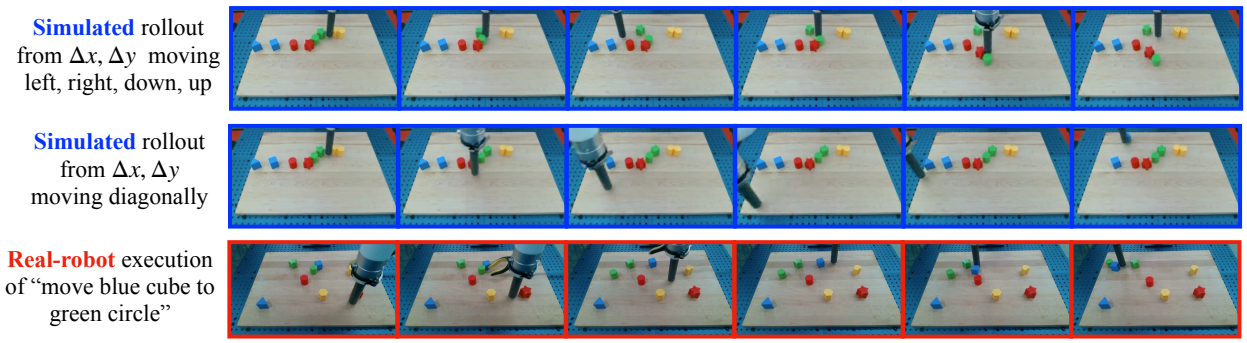


Figure 6.17: **[Top] Simulation from low-level controls**. UniSim supports low-level control actions as inputs to move endpoint horizontally, vertically, and diagonally. **[Bottom] Real-robot execution of an RL policy** trained in simulation and zero-shot onto the real Language Table task. The RL policy can successfully complete the task of "moving blue cube to green circle".

to [415]. Because UniSim can take low-level control actions as input, we can directly conduct model-based rollouts in the simulator using control actions generated by VLA policy. To acquire reward information, we use the number of steps-to-completion from the training data as a proxy reward to train a model that maps the current observation to learned reward. We then use the REINFORCE algorithm [63] to optimize the VLA policy, treating the rollouts from the simulator as the on-policy rollouts from the real environment and use the learned reward model to predict rewards from simulated rollouts. See details of RL training in Appendix A.6.4.

Results. We first do a sanity check on simulating real-robot executions by applying low-level control actions (e.g., $\Delta x = 0.05, \delta y = 0.05$) repeatedly for 20-30 environment

	Activity	MSR-VTT	VATEX	SMIT
No finetune	15.2	21.91	13.31	9.22
Activity	54.90	24.88	36.01	16.91
Simulator	46.23	**27.63**	**40.03**	**20.58**

Table 6.8: **VLM trained in UniSim** to perform video captioning tasks. CIDEr scores for PaLI-X finetuned only on simulated data from UniSim compared to no finetuning and finetuning on true video data from ActivityNet Captions. Finetuning only on simulated data has a large advantage over no finetuning and transfers better to other tasks than finetuning on true data.

steps to move the endpoint left, right, down, up, and diagonally in Figure 6.17 (top two rows). We see that the simulated rollouts capture both the endpoint movements and the physics of collision. To compare the RL policy trained in simulation to the BC policy, we qualitatively assessed the simulated rollouts in the simulator. Table 6.7 shows that RL training significantly improves the performance of the VLA policy across a wide set of tasks, especially in tasks such as "point to blue block". We then directly deploy the RL policy trained in the simulator onto the real robot in zero-shot, and observe successful task executions as shown in Figure 6.17 (bottom row). Additional results on real robot can be found in Appendix A.6.1.

6.2.6 Realistic Simulator for Broader Vision-Language Tasks

UniSim can generate training data for other machine-learning subproblems. This is especially useful when natural data is rare or difficult to collect (e.g., footage of crimes or accidents). We provide such a proof-of-concept by training vision-language models on purely generated data from UniSim, and observe significant performance benefits in video captioning.

Setup. We finetune PaLI-X [419], a VLM with 55B parameters pretrained on a broad set of image, video, and language tasks, to caption a set of videos generated by UniSim using texts from the training split of ActivityNet Captions [400]. We measure the CIDEr score of the finetuned model on the test split of ActivityNet Captions as well as other captioning tasks following the same setup as [419]. See finetuning details of PaLI-X in Appendix A.6.4.

Results. We compare PaLI-X finetuned on purely generated videos to pretrained PaLI-X without finetuning and PaLI-X finetuned on original ActivityNet Captions in Table 6.8. Purely finetuning on generated data drastically improves the captioning performance from no finetuning at all on ActivityNet (15.2 to 46.23), while achieving 84% performance of finetuning on true data. Furthermore, PaLI-X finetuned on generated data transfers better to other captioning tasks such as MSR-VTT [399], VATEX [420], and SMIT [421] than PaLI-X finetuned on true data, which tends to overfit to ActivityNet. These results suggest that UniSim can serve as an effective data generator for improving broader vision-language models.

6.2.7 Related Work

Internet-Scale Generative Models. Language models trained on internet text succeed at text-based tasks [395, 422] but not physical tasks, which requires perception and control. Internet-scale generative models can synthesize realistic images and videos [423, 28, 374, 424, 425], but have mostly been applied to generative media [426] as opposed to empowering sophisticated agents capable of multi-turn interactions. [427] shows video generation can serve as policies, but the major bottleneck for policy learning often lies in limited access to real-world environments [418]. We focus on this exact bottleneck by learning universal simulators of the real world, enabling realistic and unlimited "environment" access for training sophisticated agents interactively.

Learning World Models. Learning an accurate dynamics model in reaction to control inputs has been a long-standing challenge in system identification [428], model-based reinforcement learning [429], and optimal control [430, 431]. Most systems choose to learn one dynamics model per system in the lower dimensional state space as opposed to in the pixel space [432, 433, 434], which, despite being a simpler modeling problem, limits knowledge sharing across systems. With large transformer architectures, learning image-based world models has become plausible [118, 119, 123, 121, 435, 436], but mostly in games or simulated domains with visually simplistic and abundant data. In generative modeling of videos, previous works have leveraged text prompts [437, 438], driving motions [439, 440], 3D geometries [441, 442], physical simulations [443], frequency information [444], and user annotations [445] to introduce movements into videos. However, they focus on generating domain specific videos (e.g., for self-driving) as opposed to building a universal simulator that can be used to further improve other agents. The amount of control over generated videos in these existing work is also limited, as they do not treat video generation as a dynamics modeling problem like in our work.

6.3 Efficient Adaptation of Video Generation Models

6.3.1 Introduction

Large text-to-video models with billions of parameters trained on internet-scale data have become capable of generating highly realistic videos from general text descriptions [28, 446, 374]. When models are used for specific domains such as generating video plans for robotics [427] and self-driving cars [447], videos for animation [448], or videos with customizable styles similar to those common in text-to-image [448, 396, 135, 449, 450, 451], a pretrained text-to-video model requires task-specific adaptation. Efficient and effective adaptation of text-to-video models is what stands in the way from expanding their current application of these models in media and entertainment to their potential to solve real-world problems by modeling real-world physics and dynamics in problem-specific settings.

Unfortunately, similar to state-of-the-art language models [395, 422], pretrained text-to-video models are black boxes to the general public; one can use them to generate videos,

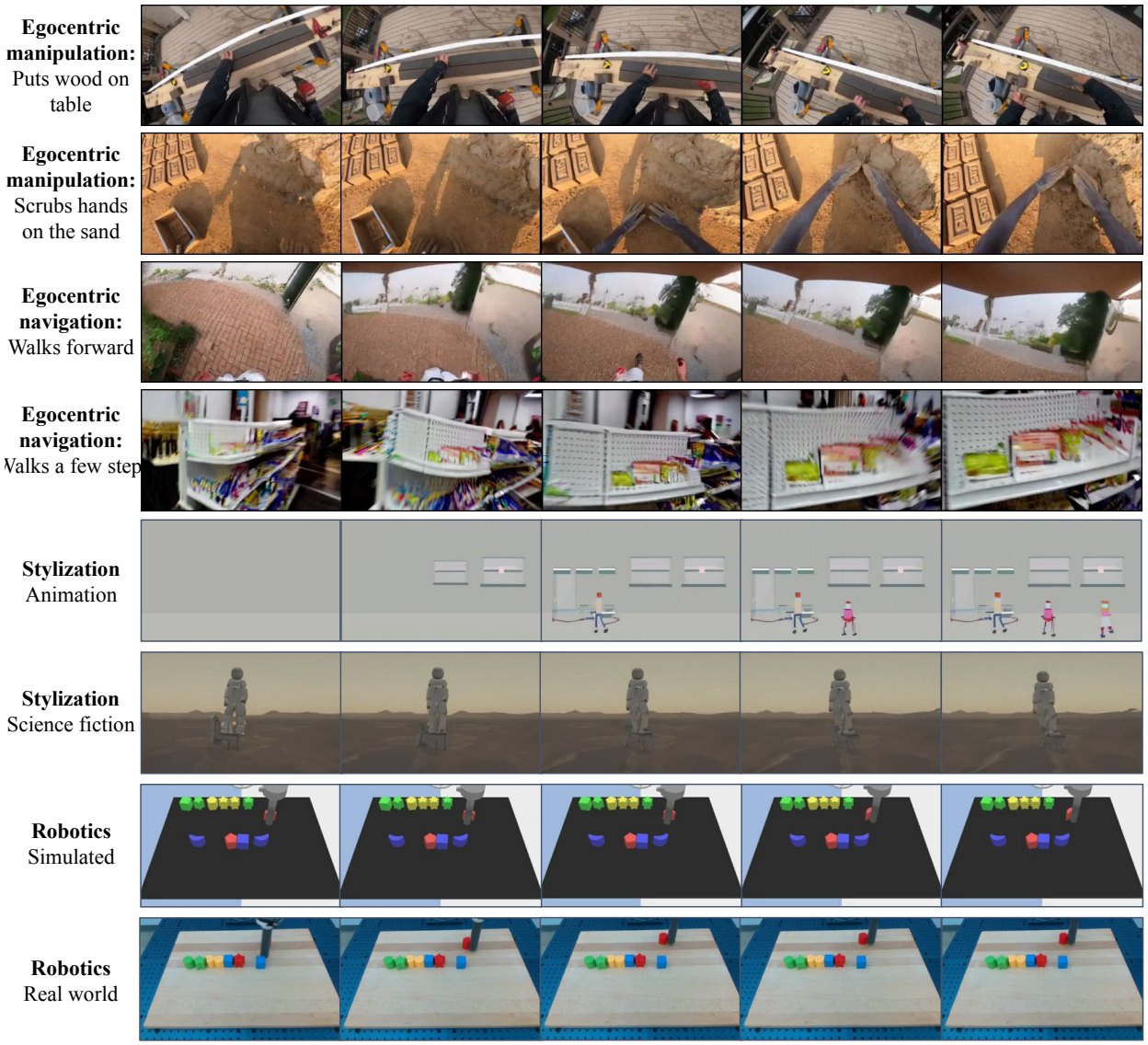


Figure 6.18: **Video Adapter Generated Videos.** Video Adapter is capable of flexible generation of diverse videos with distinct styles including videos with manipulation and navigation based egocentric motions, videos with personalized styles such as animation and science fictions, and simulated and real robotic videos.

but not finetune them to solve domain-specific tasks, as the parameters of pretrained text-to-video models are not publicly available [28, 384, 425]. This rules out direct applications of efficient finetuning from text-to-image, such as LoRA in stable diffusion [452], DreamBooth [453], and ControlNet [451], which require access to the pretrained model weights. While some finetuning-free techniques can control image generation by manipulating visual

or textual features [396, 449], it is not clear how to manipulate these features for video generation, as these features would then have to capture complex temporal information extracted from networks with orders of magnitude larger sizes than text-to-image [425].

Inspired by finetuning-free adaptation of language models through in-context learning [27] and sophisticated prompting [43], which essentially modify the prior distribution of pretrained langauge models to perform specific tasks, we ask the natural question of whether it is possible to modify the prior distribution of pretrained text-to-video models to achieve downstream tasks without finetuning the pretrained model. Intuitively, even though the exact video statistics in a downstream task differ from the pretraining videos, certain video properties such as dynamics of the world and semantics of objects from the large pretrained model are still tremendously helpful to the generation of downstream videos. This suggests that a large pretrained video model could be used as a knowledge prior to guide the generation of task-specific videos while maintaining broad properties such as temporal consistency and object permanence.

To this end, we propose Video Adapter, a probabilistic approach for exploiting a black-box video diffusion model to guide the generation of task and domain specific videos. By factoring the domain-specific video distribution into a pretrained prior and a small trainable component, we can preserve desirable characteristics of the pretrained model (i.e., temporal consistency and object permanence) in generating specialized videos, effectively adapting the black-box pretrained model without requiring access to the pretrained model weights. One limitation of Video Adapter is that it requires scores of the black-box video diffusion model as outputs, but we note that this is hard to avoid if one wants to effectively use the broad knowledge of pretrained models. Therefore, we advocate for proprietary text-to-video APIs to expose diffusion scores as additional outputs to broaden the applications of large video diffusion models.

We evaluate Video Adapter on a variety of tasks and domains as illustrated in Figure 6.18. Quantitatively, Video Adapter achieves better FVD and Inception Scores than the pretrained video model or the task-specific small models in generating domain specific videos for robotics [387] and egocentric movements [402]. Qualitatively, we show that Video Adapter can generate stylized videos such as sci-fi and animation, and further enable domain randomization in robotics [114] for bridging sim-to-real [454] through randomized stylisation of lighting and distractors.

6.3.2 Preliminaries

We first introduce relevant background information on denoising diffusion probabilistic models (DDPMs) and discuss their connection to Energy-Based Models (EBMs). We will then use this connection to EBMs to convert black-box video diffusion models to probabilistic priors.

Denoising Diffusion Probabilistic Models. Denoising diffusion probablistic models [33, 35] are a class of probabilistic generative models where the generation of a video $\tau =$

$[x_1, \ldots, x_H] \in X^H$ is formed by iterative denoising. Given a video τ sampled from a video distribution $p(\tau)$, a randomly sampled Gaussian noise variable $\epsilon \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$, and a set of T different noise levels β_t, a denoising model ϵ_θ is trained to denoise the noise corrupted video τ at each specified noise level $t \in [1, T]$:

$$\mathcal{L}_{\text{MSE}} = \left\| \epsilon - \epsilon_\theta\left(\sqrt{1-\beta_t}\tau + \sqrt{\beta_t}\epsilon,\, t\right) \right\|^2$$

Given this learned denoising function, new videos may be generated from the diffusion model by initializing a video sample τ_T at noise level T from a Gaussian $\mathcal{N}(\mathbf{0}, \mathbf{I})$. This sample τ_T is then iteratively denoised following the expression:

$$\tau_{t-1} = \alpha_t(\tau_t - \gamma_t \epsilon_\theta(\tau_t, t) + \xi), \quad \xi \sim \mathcal{N}\left(\mathbf{0}, \sigma_t^2 \mathbf{I}\right), \tag{6.4}$$

where γ_t is the step size of denoising, α_t is a linear decay on the currently denoised sample, and σ_t is a time varying noise that depends on α_t and γ_t. The final sample τ_0 after T rounds of denoising corresponds to the final generated video.

Energy-Based Models View of DDPMs. The denoising function ϵ_θ estimates the score [455, 456, 135] of an underlying (unnormalized) EBM probability distribution [96, 97] characterizing the noise perturbed data. Therefore, a diffusion model corresponds to an EBM, $p_\theta(\tau) \propto e^{-E_\theta(\tau)}$, where the denoising function is given by $\epsilon(\tau_t, t) = \nabla_\tau E_\theta(\tau_t)$. The sampling procedure in a diffusion model corresponds to the Langevin sampling procedure on an EBM (see derivation in Appendix A.7.1):

$$\tau_{t-1} = \alpha_t(\tau_t - \gamma \nabla_\tau E_\theta(\tau_t) + \xi), \quad \xi \sim \mathcal{N}\left(\mathbf{0}, \sigma_t^2 \mathbf{I}\right). \tag{6.5}$$

This equivalence of diffusion models and EBMs allows us to consider sampling from the product of two different diffusion models $p_1(\tau)p_2(\tau)$, such that each diffusion model corresponds to an EBM, $e^{-E_1(\tau)}$ and $e^{-E_2(\tau)}$, and the product is given by $e^{-E'(\tau)} = e^{-(E_1(\tau)+E_2(\tau))}$. In particular, we can sample from this new distribution also by using Langevin sampling:

$$\tau_{t-1} = \alpha_t(\tau_t - \gamma \nabla_\tau E'_\theta(\tau_t) + \xi), \quad \xi \sim \mathcal{N}\left(\mathbf{0}, \sigma_t^2 \mathbf{I}\right), \tag{6.6}$$

which corresponds to the sampling procedure using denoising functions

$$\tau_{t-1} = \alpha_t(\tau_t - \gamma(\epsilon_{\theta_1}(\tau_t, t) + \epsilon_{\theta_2}(\tau_t, t)) + \xi), \quad \xi \sim \mathcal{N}\left(\mathbf{0}, \sigma_t^2 \mathbf{I}\right). \tag{6.7}$$

Below we will illustrate how this factored EBM parameterization of a diffusion model can allow a black-box pretrained model to be leveraged as a probabilistic prior.

6.3.3 Probabilistic Adaptation of Black-Box Text-to-Video Models

To explain how a black-box text-conditioned video diffusion model can be effectively used as a probabilistic prior for video generation, we will first introduce the functional form of

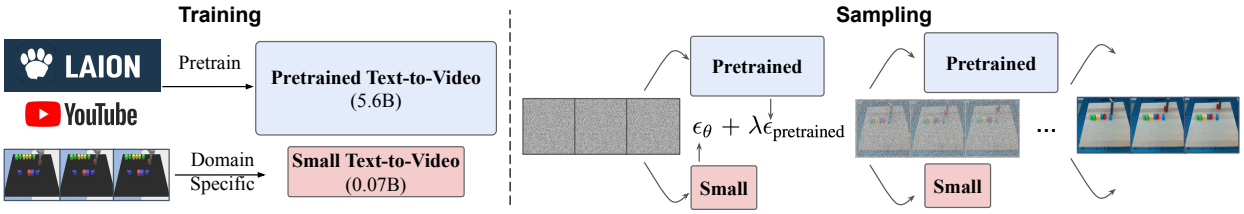


Figure 6.19: **Video Adapter Framework.** Video Adapter only requires training a small domain-specific text-to-video model with orders of magnitude fewer parameters than a large video model pretrained from internet data. During sampling, Video Adapter composes the scores of the pretrained and the domain specific video models, achieving high-quality and flexible video synthesis.

probabolistic adaptation in Section 6.3.3.1, and then discuss how the probabilistic composition can be implemented with diffusion models in Section 6.3.3.2. To generate high-quality videos, we also explain in Section 6.3.3.3 how the underlying probabilistic composition can be sharpened to generate low temperature samples.

6.3.3.1 Black-Box Text-to-Video Models as Probabilistic Priors

Black-box text-to-video models were pretrained on massive datasets consisting of millions of videos, and are therefore able to capture a powerful prior $p_{\text{pretrained}}(\tau|\text{text})$ on the natural distribution of videos τ. Given a smaller task-specific dataset of video-text pairs, $D_{\text{Adapt}} = \{(\tau_0, \text{text}_0), \ldots, (\tau_n, \text{text}_n)\}$, how can one leverage the powerful prior captured by a pretrained video diffusion model to synthesize videos similar to those in D_{Adapt}? One approach is to directly finetune the weights of $p_{\text{pretrained}}(\tau|\text{text})$ using D_{Adapt}, but $p_{\text{pretrained}}(\tau|\text{text})$ has billions of parameters whose weights are often proprietary to private enterprises. Similar challenges with large language models have led to prompting and in-context learning. Analogously, we propose Video Adapter as a finetuning-free method to adapt pretrained video diffusion to a new dataset of videos D_{Adapt} through probabilistic composition. Specifically, given D_{Adapt}, we learn a separate small video diffusion model $p_\theta(\tau|\text{text})$ to represent the distribution of videos in D_{Adapt}. We then adapt $p_{\text{pretrained}}(\tau|\text{text})$ to D_{Adapt} by constructing a product distribution $p_{\text{produce}}(\tau|\text{text})$ in the form (see adaptation to multiple domains in Appendix A.7.4):

$$\underbrace{p_{\text{product}}(\tau|\text{text})}_{\text{Product Distribution}} \propto \underbrace{p_{\text{pretrained}}(\tau|\text{text})}_{\text{Pretrained Prior}} \underbrace{p_\theta(\tau|\text{text})}_{\text{Small Video Model}}, \tag{6.8}$$

By fixing the pretrained model $p_{\text{pretrained}}(\tau|\text{text})$, we train the small video model $p_\theta(\tau|\text{text})$ via maximum likelihood estimation on D_{Adapt}. This allows $p_\theta(\tau|\text{text})$ to exhibit high likelihood across videos in D_{Adapt}, but because $p_\theta(\tau|\text{text})$ is a small model trained on less diverse data, it can exhibit erroneously high likelihood across many unrealistic videos. The product distribution $p_{\text{product}}(\tau|\text{text})$ removes unrealistic videos by downweighting any videos τ that

are not likely under the pretrained prior, enabling one to generate videos in the style of D_{Adapt} that are realistic under $p_{\text{pretrained}}(\tau|\text{text})$.

6.3.3.2 Implementing Probabilistic Adaptation

To adapt the black-box model $p_{\text{product}}(\tau|\text{text})$ from Equation 6.8, as well as to sample from it, we exploit the EBM interpretation of diffusion models discussed in Section 6.3.2. Based on the EBM interpretation, the pretrained diffusion model $p_{\text{pretrained}}(\tau|\text{text})$ corresponds to an EBM $e^{-E_{\text{pretrained}}(\tau|\text{text})}$ while the smaller video model $p_\theta(\tau|\text{text})$ parameterizes an EBM $e^{-E_\theta(\tau|\text{text})}$. The product distribution then corresponds to:

$$p_{\text{product}}(\tau|\text{text}) \propto p_{\text{pretrained}}(\tau|\text{text})p_\theta(\tau|\text{text}) \propto e^{-(E_{\text{pretrained}}(\tau|\text{text})+E_\theta(\tau|\text{text}))} = e^{-E'(\tau|\text{text})},$$

which specifies a new EBM $E'(\tau)$ from the sum of energy functions of the component models.

Substituting this EBM into Equation 6.6, we can sample from the product distribution $p_{\text{product}}(\tau|\text{text})$ through the diffusion sampling procedure:

$$\tau_{t-1} = \alpha_t(\tau_t - \gamma\nabla_\tau(E_{\text{pretrained}}(\tau_t|\text{text}) + E_\theta(\tau_t|\text{text})) + \xi), \quad \xi \sim \mathcal{N}\big(\mathbf{0}, \sigma_t^2\mathbf{I}\big)$$

which corresponds to sampling from Equation 6.4 according to

$$\tau_{t-1} = \alpha_t(\tau_t - \gamma(\epsilon_{\text{pretrained}}(\tau_t, t|\text{text}) + \epsilon_\theta(\tau_t, t|\text{text})) + \xi), \quad \xi \sim \mathcal{N}\big(\mathbf{0}, \sigma_t^2\mathbf{I}\big).$$

Thus, to probabilistically adapt the pretrained black-box model to a new dataset D_{Adapt}, we can use the standard diffusion sampling procedure, but change the denoising prediction to the sum of predictions from both the black-box pretrained model and the task-specific small model. To control the strength of the pretrained prior in final video generation, we can introduce a weight term λ to scale the pretrained distribution $p^\lambda_{\text{pretrained}}(\tau|\text{text})$, which corresponds to scaling the denoised prediction from $\epsilon_{\text{pretrained}}(\tau_t, t|\text{text})$ by a scalar λ

$$\epsilon(\tau_t, t|\text{text}) = \epsilon_\theta(\tau_t, t|\text{text}) + \lambda\epsilon_{\text{pretrained}}(\tau_t, t|\text{text}). \tag{6.9}$$

The combined model can be further refined by integrating multiple steps MCMC sampling between each diffusion noise distribution similar to [457, 458]. Note that the prior strength λ can be tunab

6.3.3.3 Adapting Low Temperature Sampling

In practice, text conditioning in the denoising model $\epsilon(\tau_t, t|\text{text})$ from Equation 6.9 are often parametrized using classifier-free guidance [106] to generate sharp images or videos conditioned on text while avoiding distractions from spurious likelihood modes of diffusion models. This corresponds to sampling from the modified probability distribution:

$$p^{\text{cfg}}(\tau|\text{text}) \;\propto\; p(\tau)\left(\frac{p(\tau|\text{text})}{p(\tau)}\right)^\omega \;\propto\; p(\tau)p(\text{text}|\tau)^\omega,$$

where ω corresponds to the classifier free guidance strength, typically chosen to be significantly larger than 1. By upweighting the expression $p(\text{text}|\tau)$ via the inverse temperature ω, the modified distribution $p^{\text{cfg}}(\tau|\text{text})$ above can generate lower temperature video samples conditioned on the text.

It appears straightforward to similarly construct low temperature samples when adapting the black-box model by sampling from the distribution

$$p^{\text{cfg}}_{\text{product}}(\tau|\text{text}) \propto p^{\text{cfg}}_{\text{pretrained}}(\tau|\text{text})p^{\text{cfg}}_{\theta}(\tau|\text{text}), \tag{6.10}$$

but using the classifier-free distribution $p^{\text{cfg}}_{\text{pretrained}}(\tau|\text{text})$ as the probabilistic prior is now problematic, since classifier-free guidance has restricted $p^{\text{cfg}}_{\text{pretrained}}(\tau|\text{text})$ to very few high probability modes which might be incompatible with D_{Adapt}. To effectively leverage a broad probabilistic prior while simultaneously generating low temperature samples emulating D_{Adapt}, we propose to first construct a new text-conditioned video distribution following Section 6.3.3.1:

$$p_{\text{product}}(\tau|\text{text}) \propto p_{\text{pretrained}}(\tau|\text{text})p_{\theta}(\tau|\text{text}).$$

We can then use the density ratio of this composed text-conditioned distribution with the unconditional video density $p_\theta(\tau)$ learned on D_{Adapt} to construct a new implicit classifier $p_{\text{product}}(\tau|\text{text})$. By increasing the inverse temperature ω on this implicit classifier, we can generate low temperature and high quality video samples conditioned on a given text by sampling from the modified distribution:

$$p^{*}_{\text{product}}(\tau|\text{text}) = p_{\theta}(\tau)\left(\frac{p_{\text{product}}(\tau|\text{text})}{p_{\theta}(\tau)}\right)^{\omega},$$

which corresponds to sampling from a modified denoising function:

$$\tilde{\epsilon}_{\theta}(\tau, t|\text{text}) = \epsilon_{\theta}(\tau, t) + \omega(\epsilon_{\theta}(\tau, t|\text{text}) + \lambda\epsilon_{\text{pretrained}}(\tau, t|\text{text}) - \epsilon_{\theta}(\tau, t))$$

We quantitatively and qualitatively ablate the effect of this denoising function in Figure 6.23 and Table 6.10, showing that this variant leads to better blending of styles between models. The overall pseudocode for the proposed approach with classifier-free guidance is given in Algorithm 2.

6.3.4 Experiments

In this section, we illustrate how a black-box pretrained text-to-video model can deliver a rich set of downstream capabilities when combined with a task-specific video model. In particular, leveraging a high quality and broad probabilistic prior enables (1) controllable video synthesis from edge-only inputs, (2) high-quality video modeling that outperforms both the pretrained model and the task-specific video model, and (3) domain randomization and data augmentation for robotics. See experiment details and additional experimental results in Appendix A.7.2 and in supplementary material.

Algorithm 2 Sampling algorithm of Video Adapter.

Input: Pretrained black-box model $\epsilon_{\text{pretrained}}(\tau, t|\text{text})$, inverse temperature ω, prior strength λ.

Initialize sample $\tau_T \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I})$
for $t = T, \ldots, 1$ **do**
  $\tilde{\epsilon}_{\text{text}} \leftarrow \epsilon_\theta(\tau_t, t|\text{text}) + \lambda\epsilon_{\text{pretrained}}(\tau_t, t|\text{text})$ // Compute score using text-conditioned prior.
  $\epsilon \leftarrow \epsilon_\theta(\tau_t, t)$ // Compute unconditional score.
  $\tilde{\epsilon}_{\text{cfg}} \leftarrow \epsilon + \omega(\tilde{\epsilon}_{\text{text}} - \epsilon)$ // Compute weight for low temperature sampling.
  $\tau_{t-1} = \texttt{ddpm\_sample}(\tau_t, \tilde{\epsilon}_{\text{cfg}})$ // Run diffusion sampling (can use other samplers).

6.3.4.1 Adapting to Specific Video Domains

Setup. We first demonstrate that the probabilistic prior in Video Adapter can be used to adapt and modify the styles of videos. We curate two adaptation datasets D_{Adapt}, one with an "animation" style and the other with a "scifi" style, where videos containing relevant keywords in their descriptions are grouped together to form D_{Adapt}. A black-box large video diffusion model with 5.6B parameters was pretrained on mapping Sobel edges to all videos, and two task-specific small models with 330M parameters were trained to map Sobel edges to D_{Adapt} videos.

Stylizing Video Generation. In Figure 6.21 and Figure 6.22, we demonstrate how the pretrained prior can adapt the animation and scifi models to alternative styles while maintaining the original animation and scifi contents. These results show that Video Adapter can effectively combine rich knowledge of styles from the black-box model, such as "digital art", "outdoor video", "storybook illustration", with the animation content of the small model, thereby achieving flexible stylization.

Specific Animation Style. We further trained a small video model on an "animation" style of a particular artist. In Figure 6.20, we illustrate how the pretrained prior can maintain the anime content while changing the styles such as background color.

Figure 6.20: **Instance Specific Stylization.** Video Adapter enables the stylization of video model trained on a single animation style

Analysis. In Figure 6.23, we change the magnitude of the weight on the pretrained prior, and compare Video Adapter with directly interpolating the classifier-free scores between the pretrained and adapter models (as in Equation 6.10). We find that

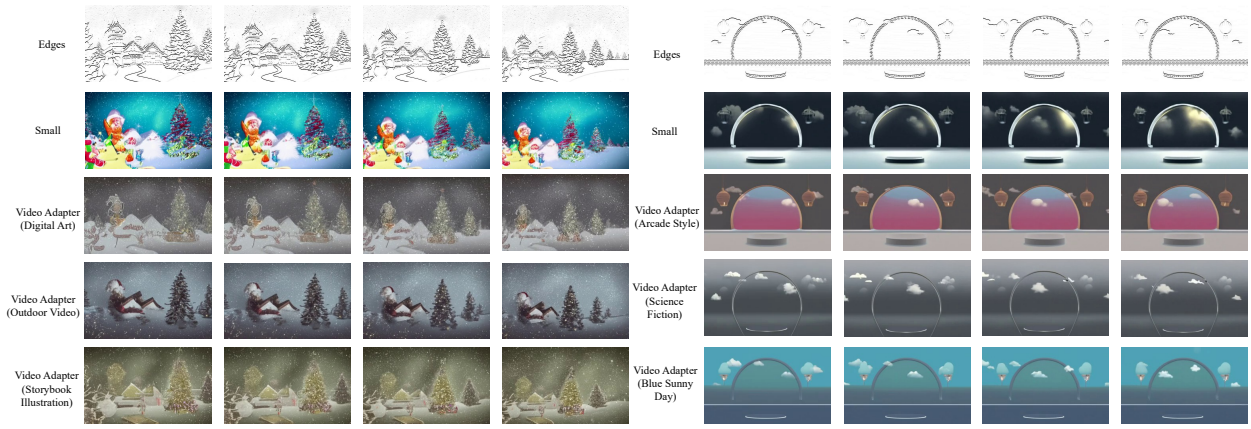


Figure 6.21: **Video Adapter enables stylization of a Animation Specific Model.** Video Adapter enables a large pretrained model to adapt and change the style a small animation style model.

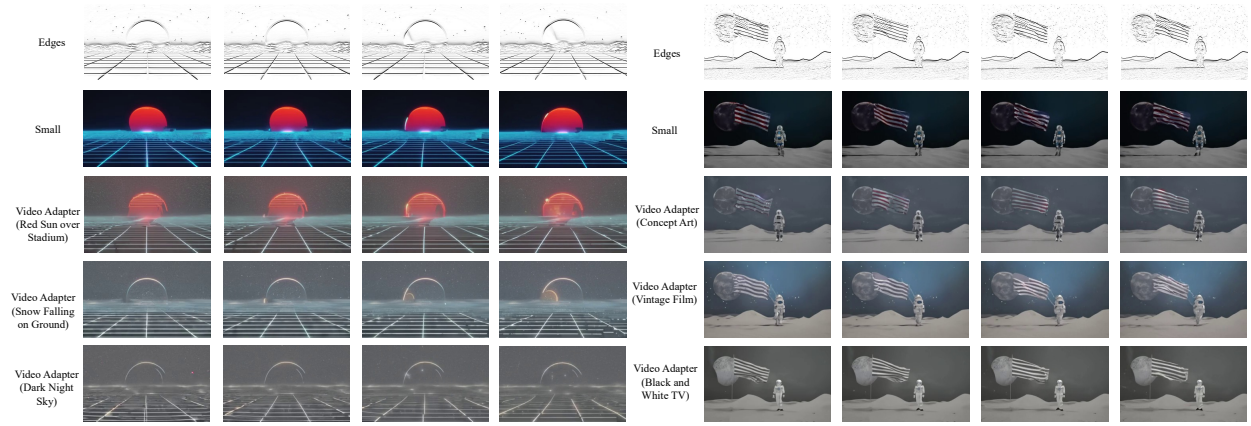


Figure 6.22: **Video Adapter enables stylization of a SciFi Specific Model.** Video Adapter enables a large pretrained model to adapt and change the style a small Scifi animation style model.

Video Adapter maintains the adapter style more accurately, whereas classifier-free score interpolation collapses to the teacher style with intermediate interpolation, leading to erratic artifacts.

6.3.4.2 High-Quality Efficient Video Modeling

Setup. To demonstrate Video Adapter's ability in adapting the black-box pretrained model to domains that are not a part of pretraining, we consider adapting to Ego4D [402] and Bridge

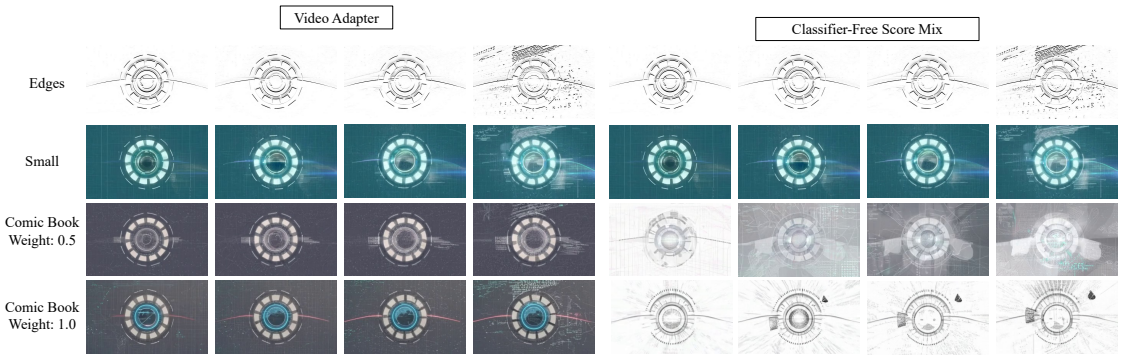


Figure 6.23: **Analysis of Video Adapter.** As adaptation weight increases, Video Adapter modifies the style as instructed (left), whereas directly mixing two classifier-free guidance scores fails to adapt the video (right).

Data [387]. These adaptations are nontrivial, as Ego4D consists of mostly egocentric videos that are not commonly found on the internet. Similarly, the Bridge Data consists of task-specific videos of a WidowX250 robot that is out of the distribution of the pretraining data. For Ego4D, we take a subset of the original dataset consisting of 97k text-video pairs and split them into train (90%) and test (10%) to form D_{Adapt}. For the Bridge Data, we take the entire dataset consisting of 7.2k text-video pairs and use the same train-test split to form D_{Adapt}.

For the pretrained model, we use the 5.6B base model pretrained on generic internet videos from [28]. For the task-specific small model, we downscale the video diffusion model from [28] by a factor of 80, 40, and 2 to create a diverse set of small models to be trained on task-specific D_{Adapt}. Table 6.9 shows the number of parameters of pretrained and small video models. Both the pretrained model and the small models are trained to generate subsequent frames conditioned on the first frame.

Quantitative Results. Table 6.9 shows the quantitative performance of Video Adapter under different video modeling metrics. On the Bridge Data, training a small model with parameters equivalent to 1.25% of the pretrained video model (first row) already achieves better metrics than the pretrained model. However, Video Adapter incorporating the pretrained model as a probablistic prior is able to further improve the metrics of the small model (second row). On Ego4D, due to the complexity of the egocentric videos, the smallest model with 1.25% of the pretrained video model can no longer achieve performance better than the pretrained model (first row), but incorporating the pretrained model during sampling still improves performance (second row). After increasing the size of the small model, Video Adapter is able to achive better metrics than both the pretrained and task-specific model. We further compare Video Adapter to finetuning the pretrained model for an equivalent number of TPU hours (see Appendix A.7.3), and show that Video Adapter achieves better performance than full tuning. Note that we only compare to full tuning out of curiosity as

	Bridge			Ego4D		
Model	FVD ↓	FID ↓	Param (B)↓	FVD ↓	IS ↑	Param (B) ↓
Small (S)	186.8	38.8	0.07	228.3	2.28	0.07
Small (S) + Pretrained	**177.4**	**37.6**	0.07	156.3	2.82	0.07
Small (L)	152.5	30.1	0.14	65.1	3.31	2.8
Small (L) + Pretrained	**148.1**	**29.5**	0.14	**52.5**	**3.53**	2.8
Pretrained	350.1	42.6	5.6	91.7	3.12	5.6
Pretrained Finetune	321.0	39.4	5.6	75.5	3.33	5.6

Table 6.9: **Video Modeling Quantitative Performance** Video Adapter (Small + Pretrained) achieves better FVD, FID, and Inception Scores than both the pretrained model, pretrained model finetuned for equivalent number of TPU hours, and the task-specific small model with parameters as fewer as 1% of the pretrained model.

opposed to benchmarking, as the motivation of this work is the lack of weight access to the black-box pretrained models.

Qualitative Results. Figure 6.24 and Figure 6.25 show the generated videos on Bridge Data and Ego4D. On the Bridge Data in Figure 6.24, the pretrained model produces videos that do not correspond to the task described by the text (there is no robot arm movements in the generated video). The task-specific small model produces videos with unrealistic movements that teleport the robot arm. Video Adapter, on the other hand, produces videos with realistic movements that complete the task.

On Ego4D in Figure 6.25, the pretrained model produces high quality videos that contain little egocentric movement (first row), as the pretraining data mostly consists of generic videos from the internet that are not egocentric. The task-specific small model trained on Ego4D, on the other hand, produces videos with egocentric movement but of low quality (second row) due to limited model capacity. Video Adapter combines the best of both and generates high-quality egocentric videos (third row).

Model	FVD ↓	FID ↓
CFG Mix	167.4	33.1
Small (L)	152.5	30.1
Video Adapter	**148.1**	**29.5**

Table 6.10: **Ablations.** Video Adapter improves the underlying video modeling performance of models on while directly mixing classifier-free scores (CFG Mix) hurts performance.

Ablations. In Table 6.10, we report generative modeling performance of the small model on Bridge either using Video Adapter, or a interpolation between the classifier-free scores of pretrained and small models. We find that Video Adapter improves performance, while interpolation between classifier-free scores hurts performance.

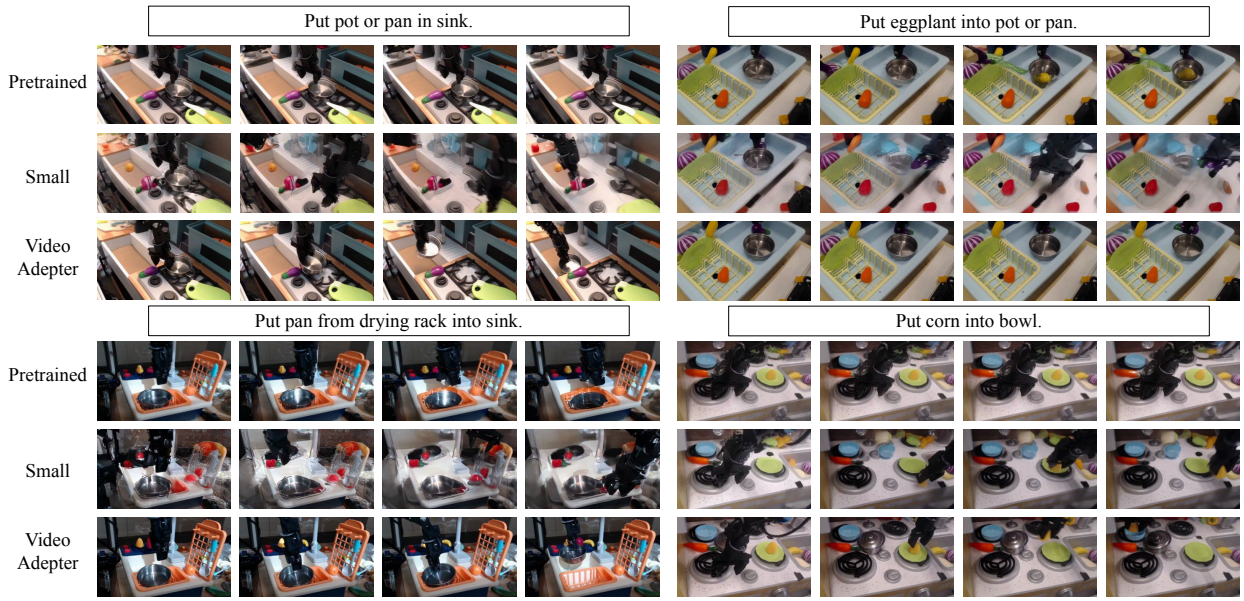


Figure 6.24: **Video Adapter on Bridge Data.** The pretrained model (first row) produces videos that are high-quality but are generally static and fail to complete the task. The small (L) model (second row) produces low-quality videos with unrealistic arm movements. Video Adapter (third row) produces high-quality videos and successfully completes the task.

6.3.4.3 Sim-to-Real Video Augmentation

Setup. One important application of controllable video synthesis is to render realistic robotic videos from simulation with a variety of data augmentations so that policies trained on the augmented observations are more likely to generalize well to real-world settings [454]. To demonstrate Video Adapter's capability in supporting sim-to-real transfer, we train a task-specific small edge-to-real model on 160k real robot trajectories of the LanguageTable dataset [459], generating videos of execution conditioned on the Sobel edges of the real videos. Similarly, we train another small edge-to-sim model on 160k simulated robot videos. Note that the simulated and real robotics data are not paired (paired sim-to-real data are hard to find) but are connected through edge-conditioning. We again leverage the edge-conditioned large model pretrained on internet data for style specification.

Adapted Videos. Figure 6.26 shows the generated robotic videos from Video Adapter. Video Adapter can effectively generate paired simulated and real robotic videos that complete a task described by a language prompt, and further generate videos with various data augmentation styles that can be utilized to train policies with better sim-to-real transfer abilities through techniques similar to domain randomization [114].

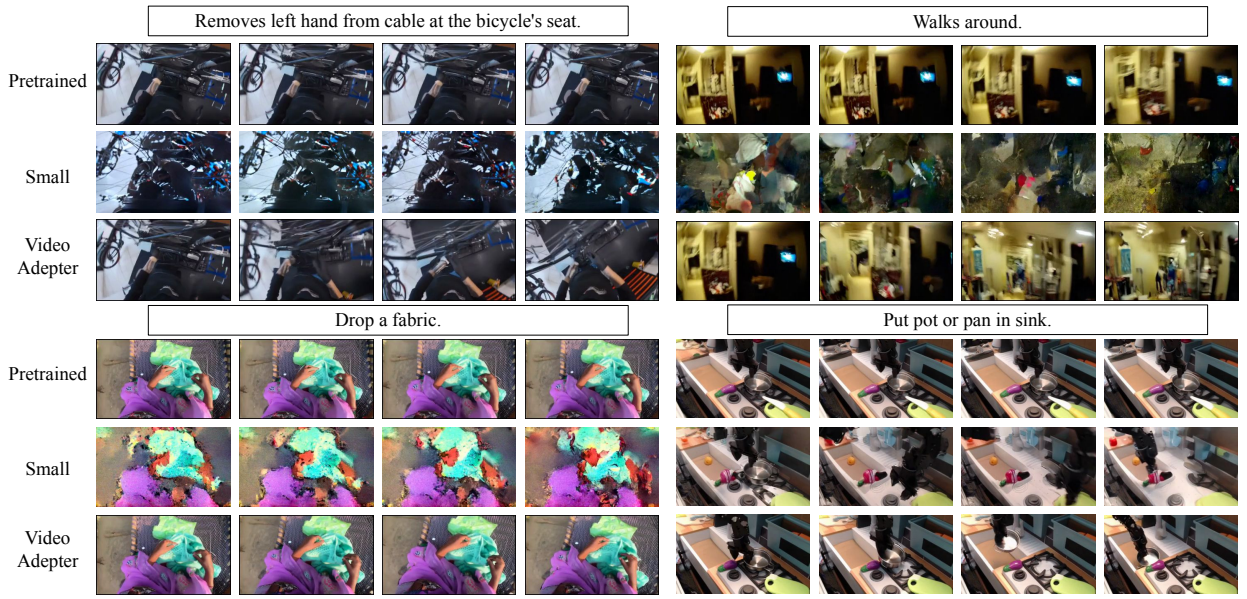


Figure 6.25: **Video Adapter on Ego4D.** The pretrained model (first row) produces high-quality but nearly static videos that do not reflect the egocentric nature.The small (L) model (second row) produces low-quality videos but with more egocentric movements. Video Adapter (third row) produces high-quality and egocentric videos.

6.3.5 Related Work

Text-to-Video Synthesis. Following the recent success of text-to-image models [460, 461, 396, 462, 463, 464, 465], large text-to-video models with autoregressive [446, 384, 466, 423] and diffusion [28, 374, 425, 438, 467] architectures have been developed, often by extending existing text-to-image models. Unfortunately, the model weights of large text-to-video models are generally not publically available, preventing downstream adaptations of these models.

Adapting Pretrained Models Adapting pretrained models for customized editing, inpainting, and stylization has been extensively studied in text-to-image and image-to-image translation models [449, 468, 469, 470, 471, 472, 450, 473, 474, 475]. In text-to-video, most existing work either leverages text prompts [476, 467], finetunes a pretrained model on stylized data [477], or performing light training on a copy of the pretrained video model similar to ControlNet [478]. Text-only adaption can be unreliable, whereas finetuning, prefix-tuning [479], low-rank adaptation [480], and ControlNet all require access to the pretrained model weights, which are often not available for text-to-video models. Black-box adaptation has been applied extensively in language models [27, 43, 166], and large video models will soon face the same problem.

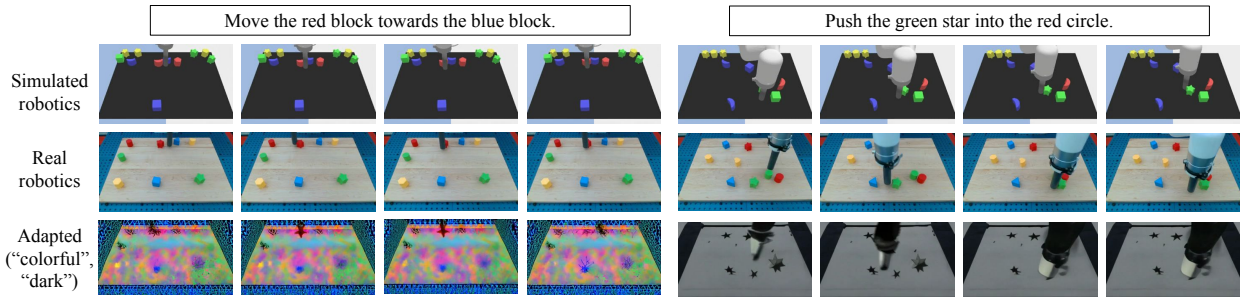


Figure 6.26: **Video Adapter on sim-to-real transfer.** First row: simulated videos of execution plans generated by Video Adapter. Second row: real videos of execution plans generated by Video Adapter. Third row: real videos of execution plans generated by Video Adapter with data augmentation.

Compositional Generative Models. The techniques in this work are further related to existing work on compositional generative modeling [135, 481, 134, 457, 482, 483, 484, 485, 486, 487, 488, 489, 490], where different generative models are probabilistically combined to jointly generate outputs. In [134], an approach to combine different probability distributions using EBMs is introduced. Most similar in spirit to this work, [486] composes a pretrained language model with a small EBM to improve language generation. However, different from this work, the small EBM is used to improve to global consistency of the language model, whereas we aim to use a small model to probabilistically adapt to a large pretrained video model to separate domains.

Chapter 7

Conclusion

Over the course of the various chapters of this thesis, we have developed techniques, frameworks, and algorithms for decision making with foundation models, and have illustrated that decision making with foundation models can be efficient, scalable, and empowering, both in theory and in practice. The techniques, frameworks, and algorithms developed have been organized into three categories. Chapter 3 developed a framework for off-policy evaluation with offline dataset. The ability to incorporate offline data has been a major milestone to efficient learning compared to traditional approaches to reinforcement learning. Chatper 4 focused on an empirical study followed by developing representation learning objectives and algorithms for learning state and action representations that provably accelerates downstream imitation learning. Chapter 5 developed conditional generation techniques to incorporate broad information such as chain-of-thought and future desired reward into a policy. These three approaches, i.e., offline RL, representation learning, and generative modeling, have been broadly adopted in many other works that leverage offline dataset for decision making. Lastly, Chapter 6 scales existing methods such as model-based search, planning, and reinforcement learning to internet-scale text and videos, empowering real-world RL, planning, and search in learned simulators. Below, we draw specific conclusions from each chapter.

Off-policy evaluation. We have proposed a unified view of off-policy evaluation via the regularized Lagrangian of the d-LP. Under this unification, existing DICE algorithms are recovered by specific (suboptimal) choices of regularizers, (redundant) constraints, and ways to convert optimized solutions to policy values. By systematically studying the mathematical properties and empirical effects of these choices, we have found that the dual estimates (i.e., policy value in terms of the state-action distribution) offer greater flexibility in incorporating optimization stablizers while preserving asymptotic unibasedness, in comparison to the primal estimates (i.e., estimated Q-values). Our study also reveals alternative estimators not previously identified in the literature that exhibit improved performance. Overall, these findings suggest promising new directions of focus for OPE research in the offline setting.

Representation learning for RL. We have derived a near-optimal objective for learning a latent action space from suboptimal offline data that provably accelerates downstream imitation learning. To learn this objective in practice, we propose transition-reparametrized actions for imitation learning (TRAIL), a two-stage framework that first pretrains a factored transition model from offline data, and then uses the transition model to reparametrize the action space prior to behavioral cloning. Our empirical results suggest that TRAIL can improve imitation learning drastically, even when pretrained on highly suboptimal data (e.g., data from a random policy), providing a new approach to imitation learning through a combination of pretraining on task-agnostic or suboptimal data and behavioral cloning on limited expert datasets. That said, our approach to action representation learning is not necessarily specific to imitation learning, and insofar as the reparameterized action space simplifies downstream control problems, it could also be combined with reinforcement learning in future work. More broadly, studying how learned action reparameterization can accelerate various facets of learning-based control represents an exciting future direction, and we hope that our results provide initial evidence of such a potential.

Return-conditioned supervised learning. Despite the empirical promise of return- or future-conditioned supervised learning (RCSL) with large transformer architectures, environment stochasticity hampers the application of supervised learning to sequential decision making. To address this issue, we proposed to augment supervised learning with the dichotomy of control principle (DoC), guiding a supervised policy to only control the controllable (actions). Theoretically, DoC learns *consistent* policies, guaranteeing that they achieve the future or return they are conditioned on. Empirically, DoC outperforms RCSL in highly stochastic environments. While DoC still falls short in addressing other RL challenges such as 'stitching' (i.e., composing sub-optimal trajectories), we hope that dichotomy of control serves as a stepping stone in solving sequential decision making with large-scale supervised learning.

UniPi. We have demonstrated the utility of representing policies using text-conditioned video generation, showing that this enables effective combinatorial generalization, multi-task learning, and real world transfer. These positive results point to the broader direction of using generative models and the wealth of data on the internet as powerful tools to generate general-purpose decision making systems.

We further discuss limitations and future directions for specific work presented in this thesis below.

Limitation and Future Work

Limitations of representation learning for RL. Even with this multitude of fresh insight into the question of representation learning in RL, our study is limited in a number of aspects, and these aspects can serve as a starting point for future work. For example,

one may consider additional downstream tasks such as multi-task, transfer, or exploration. Alternatively, one can extend our ablations to real-world domains like robot learning. Or, one may consider ablating over different network architectures. Despite these limitations, we hope our current work proves useful to RL researchers, and serves as a guide for developing even better and more general representation learning objectives.

Limitations of Procedure Cloning. One major limitation of procedure cloning compared to traditional BC is in the computational overhead, since PC needs to predict intermediate procedures. Furthermore, the choice of how to encode the expert's algorithm into a form amenable to PC is up to the practitioner. While we have presented ways to encode a variety of policies here (BFS, MCTS, scripted robotic policies), applying PC to other domains may require some amount of trial-and-error in designing the ideal computation sequence for PC.

Limitation and future work of video adaptaion. As video foundation models become more powerful but remain proprietary, black-box adaptation of these models is inevitible. We have proposed Video Adapter for leveraging black-box text-to-video models as probabilistic priors for guiding generation of specific videos. One limitation of Video Adapter is it still requires training a small domain-specific model, so adaptation is not completely training free. Another limitation is Video Adapter requires diffusion scores from pretrained black-box models. We advocate future video diffusion models to make scores as a part of the output to improve accessibility of these models.

Limitations of UniPi. Our current approach has several limitations. First, the underlying video diffusion process can be slow, it can take a minute to generate highly photorealistic videos. This slowness can be overcome by distilling the diffusion process into a faster sampling network, which in our initial experimentation resulted in a 16x speed-up. UniPi may further be sped up with by faster speed samplers in diffusion models. Second, the environments considered in this work are generally fully observed. In partially observable environments, video diffusion models might make hallucination of objects or movements that are unfaithful or not in the physical world. Integrating video models with semantic knowledge about the world may help resolve this issue, and the integration of UniPi with LLMs would be an interesting direction of future work.

Limitation and future work of UniSim While we have shown it is possible to learn a simulator of the real world in response to various action inputs ranging from texts to robot controls. UniSim can simulate visually realistic experiences for interacting with humans and training autonomous agents. We hope UniSim will instigate broad interest in learning and applying real-world simulators to improve machine intelligence. UniSim has a few limitations that call for future work:

- **Hallucination.** When an action is unrealistic given the scene (e.g., "wash hands" is given to a tabletop robot), we observe hallucinations (e.g., the table turns into a sink or the view turns away from the tabletop robot and a sink shows up). Ideally, we want UniSim to detect actions that are not possible to simulate as opposed to hallucinating unrealistic outcomes.
- **Limited memory.** The simulator conditioned on a few frames of the recent history cannot capture long-term memory (e.g., an apple in a drawer could disappear when the drawer is opened if putting the apple in the drawer is not a part of the history for conditioning). How much history to condition on depends on the application of the (e.g., whether the simulator will be used for policy learning in a near-Markov setting or question answering that requires long-term memory).
- **Limited out-of-domain generalization.** This is especially true for domains that are not represented in the training data. For instance, the simulator is mostly trained on 4 robot morphologies, and its ability to generalize to an unseen robot is limited. Further scaling up training data could help, as the training data is nowhere near all the video data available on the internet.
- **Visual simulation only.** Our simulator is not suitable for environments where actions do not cause visual observation change (e.g., different forces in grasping a static cup). A true universal simulator should capture all aspects of the world beyond visual experience (e.g., sound, sensory, etc).

Bibliography

[1] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[2] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *International Conference on Machine Learning*, pages 8748–8763. PMLR, 2021.

[3] Kuang-Huei Lee, Ofir Nachum, Mengjiao Yang, Lisa Lee, Daniel Freeman, Winnie Xu, Sergio Guadarrama, Ian Fischer, Eric Jang, Henryk Michalewski, et al. Multi-game decision transformers. *arXiv preprint arXiv:2205.15241*, 2022.

[4] Michael Laskin, Luyu Wang, Junhyuk Oh, Emilio Parisotto, Stephen Spencer, Richie Steigerwald, DJ Strouse, Steven Hansen, Angelos Filos, Ethan Brooks, et al. In-context reinforcement learning with algorithm distillation. *arXiv preprint arXiv:2210.14215*, 2022.

[5] Michael Janner, Qiyang Li, and Sergey Levine. Offline reinforcement learning as one big sequence modeling problem. *Advances in neural information processing systems*, 34:1273–1286, 2021.

[6] Yilun Du, Mengjiao Yang, Bo Dai, Hanjun Dai, Ofir Nachum, Joshua B Tenenbaum, Dale Schuurmans, and Pieter Abbeel. Learning universal policies via text-guided video generation. *arXiv e-prints*, pages arXiv–2302, 2023.

[7] Ofir Nachum and Mengjiao Yang. Provable representation learning for imitation with contrastive fourier features. *Advances in Neural Information Processing Systems*, 34:30100–30112, 2021.

[8] Aaron van den Oord, Yazhe Li, and Oriol Vinyals. Representation learning with contrastive predictive coding. *arXiv preprint arXiv:1807.03748*, 2018.

[9] Aviral Kumar, Rishabh Agarwal, Xinyang Geng, George Tucker, and Sergey Levine. Offline q-learning on diverse multi-task data both scales and generalizes. *arXiv preprint arXiv:2211.15144*, 2022.

[10] Justin Fu, Aviral Kumar, Ofir Nachum, George Tucker, and Sergey Levine. D4rl: Datasets for deep data-driven reinforcement learning. *arXiv preprint arXiv:2004.07219*, 2020.

[11] Yuval Tassa, Yotam Doron, Alistair Muldal, Tom Erez, Yazhe Li, Diego de Las Casas, David Budden, Abbas Abdolmaleki, Josh Merel, Andrew Lefrancq, et al. Deepmind control suite. *arXiv preprint arXiv:1801.00690*, 2018.

[12] Karl Pertsch, Youngwoon Lee, Yue Wu, and Joseph J Lim. Guided reinforcement learning with learned skills. In *Self-Supervision for Reinforcement Learning Workshop-ICLR 2021*, 2021.

[13] Karl Pertsch, Youngwoon Lee, and Joseph J Lim. Accelerating reinforcement learning with learned skill priors. *arXiv preprint arXiv:2010.11944*, 2020.

[14] Anurag Ajay, Aviral Kumar, Pulkit Agrawal, Sergey Levine, and Ofir Nachum. Opal: Offline primitive discovery for accelerating offline reinforcement learning. *arXiv preprint arXiv:2010.13611*, 2020.

[15] Konrad Zolna, Alexander Novikov, Ksenia Konyushkova, Caglar Gulcehre, Ziyu Wang, Yusuf Aytar, Misha Denil, Nando de Freitas, and Scott Reed. Offline learning from demonstrations and unlabeled experience. *arXiv preprint arXiv:2011.13885*, 2020.

[16] Ziyu Wang, Alexander Novikov, Konrad Zolna, Jost Tobias Springenberg, Scott Reed, Bobak Shahriari, Noah Siegel, Josh Merel, Caglar Gulcehre, Nicolas Heess, et al. Critic regularized regression. *arXiv preprint arXiv:2006.15134*, 2020.

[17] Pete Florence, Corey Lynch, Andy Zeng, Oscar A Ramirez, Ayzaan Wahid, Laura Downs, Adrian Wong, Johnny Lee, Igor Mordatch, and Jonathan Tompson. Implicit behavioral cloning. In *Conference on Robot Learning*, pages 158–168. PMLR, 2022.

[18] Santhosh Kumar Ramakrishnan, Aaron Gokaslan, Erik Wijmans, Oleksandr Maksymets, Alexander Clegg, John M Turner, Eric Undersander, Wojciech Galuba, Andrew Westbury, Angel X Chang, Manolis Savva, Yili Zhao, and Dhruv Batra. Habitat-matterport 3d dataset (HM3d): 1000 large-scale 3d environments for embodied AI. In *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2021.

[19] Anthony Brohan, Noah Brown, Justice Carbajal, Yevgen Chebotar, Joseph Dabis, Chelsea Finn, Keerthana Gopalakrishnan, Karol Hausman, Alex Herzog, Jasmine Hsu, et al. Rt-1: Robotics transformer for real-world control at scale. *arXiv preprint arXiv:2212.06817*, 2022.

[20] Caglar Gulcehre, Ziyu Wang, Alexander Novikov, Thomas Paine, Sergio Gómez, Konrad Zolna, Rishabh Agarwal, Josh S Merel, Daniel J Mankowitz, Cosmin Paduraru,

et al. Rl unplugged: A suite of benchmarks for offline reinforcement learning. *Advances in Neural Information Processing Systems*, 33:7248–7259, 2020.

[21] David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. Mastering the game of go with deep neural networks and tree search. *nature*, 529(7587):484–489, 2016.

[22] Long Ouyang, Jeff Wu, Xu Jiang, Diogo Almeida, Carroll L Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. Training language models to follow instructions with human feedback. *arXiv preprint arXiv:2203.02155*, 2022.

[23] David Silver, Thomas Hubert, Julian Schrittwieser, Ioannis Antonoglou, Matthew Lai, Arthur Guez, Marc Lanctot, Laurent Sifre, Dharshan Kumaran, Thore Graepel, et al. Mastering chess and shogi by self-play with a general reinforcement learning algorithm. *arXiv preprint arXiv:1712.01815*, 2017.

[24] Rishabh Agarwal, Max Schwarzer, Pablo Samuel Castro, Aaron Courville, and Marc G Bellemare. Beyond tabula rasa: Reincarnating reinforcement learning. *arXiv preprint arXiv:2206.01626*, 2022.

[25] Rishi Bommasani, Drew A Hudson, Ehsan Adeli, Russ Altman, Simran Arora, Sydney von Arx, Michael S Bernstein, Jeannette Bohg, Antoine Bosselut, Emma Brunskill, et al. On the opportunities and risks of foundation models. *arXiv preprint arXiv:2108.07258*, 2021.

[26] Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners. 2019.

[27] Tom B Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *arXiv preprint arXiv:2005.14165*, 2020.

[28] Jonathan Ho, William Chan, Chitwan Saharia, Jay Whang, Ruiqi Gao, Alexey Gritsenko, Diederik P Kingma, Ben Poole, Mohammad Norouzi, David J Fleet, et al. Imagen video: High definition video generation with diffusion models. *arXiv preprint arXiv:2210.02303*, 2022.

[29] Tim Brooks, Bill Peebles, Connor Holmes, Will DePue, Yufei Guo, Li Jing, David Schnurr, Joe Taylor, Troy Luhman, Eric Luhman, Clarence Ng, Ricky Wang, and Aditya Ramesh. Video generation models as world simulators. 2024.

[30] Imola K Fodor. A survey of dimension reduction techniques. Technical report, Lawrence Livermore National Lab.(LLNL), Livermore, CA (United States), 2002.

[31] Honglak Lee, Alexis Battle, Rajat Raina, and Andrew Ng. Efficient sparse coding algorithms. *Advances in neural information processing systems*, 19, 2006.

[32] Lawrence Cayton et al. *Algorithms for manifold learning.* eScholarship, University of California, 2008.

[33] Jascha Sohl-Dickstein, Eric Weiss, Niru Maheswaranathan, and Surya Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics. In *International Conference on Machine Learning*, pages 2256–2265. PMLR, 2015.

[34] Yang Song, Jascha Sohl-Dickstein, Diederik P Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-based generative modeling through stochastic differential equations. *arXiv preprint arXiv:2011.13456*, 2020.

[35] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in Neural Information Processing Systems*, 33:6840–6851, 2020.

[36] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction.* MIT press, 2018.

[37] Gerald Tesauro. Td-gammon, a self-teaching backgammon program, achieves master-level play. *Neural computation*, 6(2):215–219, 1994.

[38] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*, 2013.

[39] Dean A Pomerleau. Alvinn: An autonomous land vehicle in a neural network. *Advances in neural information processing systems*, 1, 1988.

[40] Dmitry Kalashnikov, Alex Irpan, Peter Pastor, Julian Ibarz, Alexander Herzog, Eric Jang, Deirdre Quillen, Ethan Holly, Mrinal Kalakrishnan, Vincent Vanhoucke, et al. Scalable deep reinforcement learning for vision-based robotic manipulation. In *Conference on Robot Learning*, pages 651–673. PMLR, 2018.

[41] Ilge Akkaya, Marcin Andrychowicz, Maciek Chociej, Mateusz Litwin, Bob McGrew, Arthur Petron, Alex Paino, Matthias Plappert, Glenn Powell, Raphael Ribas, et al. Solving rubik's cube with a robot hand. *arXiv preprint arXiv:1910.07113*, 2019.

[42] Aarohi Srivastava, Abhinav Rastogi, Abhishek Rao, Abu Awal Md Shoeb, Abubakar Abid, Adam Fisch, Adam R Brown, Adam Santoro, Aditya Gupta, Adrià Garriga-Alonso, et al. Beyond the imitation game: Quantifying and extrapolating the capabilities of language models. *arXiv preprint arXiv:2206.04615*, 2022.

[43] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Ed Chi, Quoc Le, and Denny Zhou. Chain of thought prompting elicits reasoning in large language models. *arXiv preprint arXiv:2201.11903*, 2022.

[44] Aitor Lewkowycz, Anders Andreassen, David Dohan, Ethan Dyer, Henryk Michalewski, Vinay Ramasesh, Ambrose Slone, Cem Anil, Imanol Schlag, Theo Gutman-Solo, et al. Solving quantitative reasoning problems with language models. *arXiv preprint arXiv:2206.14858*, 2022.

[45] OpenAI. Chatgpt: Optimizing language models for dialogue, Dec 2022.

[46] Rishabh Agarwal, Dale Schuurmans, and Mohammad Norouzi. An optimistic perspective on offline reinforcement learning. In *International Conference on Machine Learning*, pages 104–114. PMLR, 2020.

[47] Andrew Szot, Alexander Clegg, Eric Undersander, Erik Wijmans, Yili Zhao, John Turner, Noah Maestre, Mustafa Mukadam, Devendra Singh Chaplot, Oleksandr Maksymets, et al. Habitat 2.0: Training home assistants to rearrange their habitat. *Advances in Neural Information Processing Systems*, 34:251–266, 2021.

[48] Linxi Fan, Guanzhi Wang, Yunfan Jiang, Ajay Mandlekar, Yuncong Yang, Haoyi Zhu, Andrew Tang, De-An Huang, Yuke Zhu, and Anima Anandkumar. Minedojo: Building open-ended embodied agents with internet-scale knowledge. *arXiv preprint arXiv:2206.08853*, 2022.

[49] Scott Reed, Konrad Zolna, Emilio Parisotto, Sergio Gomez Colmenarejo, Alexander Novikov, Gabriel Barth-Maron, Mai Gimenez, Yury Sulsky, Jackie Kay, Jost Tobias Springenberg, et al. A generalist agent. *arXiv preprint arXiv:2205.06175*, 2022.

[50] Apoorv Khandelwal, Luca Weihs, Roozbeh Mottaghi, and Aniruddha Kembhavi. Simple but effective: Clip embeddings for embodied ai. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 14829–14838, 2022.

[51] Tianxin Tao, Daniele Reda, and Michiel van de Panne. Evaluating vision transformer methods for deep reinforcement learning from pixels. *arXiv preprint arXiv:2204.04905*, 2022.

[52] Mojtaba Komeili, Kurt Shuster, and Jason Weston. Internet-augmented dialogue generation. *arXiv preprint arXiv:2107.07566*, 2021.

[53] Romal Thoppilan, Daniel De Freitas, Jamie Hall, Noam Shazeer, Apoorv Kulshreshtha, Heng-Tze Cheng, Alicia Jin, Taylor Bos, Leslie Baker, Yu Du, et al. Lamda: Language models for dialog applications. *arXiv preprint arXiv:2201.08239*, 2022.

[54] Angeliki Lazaridou, Elena Gribovskaya, Wojciech Stokowiec, and Nikolai Grigorev. Internet-augmented language models through few-shot prompting for open-domain question answering. *arXiv preprint arXiv:2203.05115*, 2022.

[55] Kurt Shuster, Jing Xu, Mojtaba Komeili, Da Ju, Eric Michael Smith, Stephen Roller, Megan Ung, Moỹa Chen, Kushal Arora, Joshua Lane, Morteza Behrooz, William Ngan, Spencer Poff, Naman Goyal, Arthur Szlam, Ỹ-Lan Boureau, Melanie Kambadur, and Jason Weston. Blenderbot 3: a deployed conversational agent that continually learns to responsibly engage, 2022.

[56] Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. React: Synergizing reasoning and acting in language models, 2022.

[57] Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Jerry Plappert, Matthias andT̃worek, Jacob Hilton, Reiichiro Nakano, et al. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021.

[58] Ruibo Liu, Jason Wei, Shixiang Shane Gu, Te-Yen Wu, Soroush Vosoughi, Claire Cui, Denny Zhou, and Andrew M Dai. Mind's eye: Grounded language model reasoning through simulation. *arXiv preprint arXiv:2210.05359*, 2022.

[59] Luyu Gao, Aman Madaan, Shuyan Zhou, Uri Alon, Pengfei Liu, Yiming Yang, Jamie Callan, and Graham Neubig. Pal: Program-aided language models, 2022.

[60] Lili Chen, Kevin Lu, Aravind Rajeswaran, Kimin Lee, Aditya Grover, Misha Laskin, Pieter Abbeel, Aravind Srinivas, and Igor Mordatch. Decision transformer: Reinforcement learning via sequence modeling. *Advances in neural information processing systems*, 34:15084–15097, 2021.

[61] Martin L Puterman. *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. John Wiley & Sons, Inc., 1994.

[62] Dean A Pomerleau. Alvinn: An autonomous land vehicle in a neural network. Technical report, CARNEGIE-MELLON UNIV PITTSBURGH PA ARTIFICIAL INTELLIGENCE AND PSYCHOLOGY ..., 1989.

[63] Ronald J Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3):229–256, 1992.

[64] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

[65] Sham M Kakade. A natural policy gradient. *Advances in neural information processing systems*, 14, 2001.

[66] Jan Peters, Katharina Mulling, and Yasemin Altun. Relative entropy policy search. In *Twenty-Fourth AAAI Conference on Artificial Intelligence*, 2010.

[67] John Schulman, Sergey Levine, Pieter Abbeel, Michael Jordan, and Philipp Moritz. Trust region policy optimization. In *International conference on machine learning*, pages 1889–1897. PMLR, 2015.

[68] Richard S Sutton, David McAllester, Satinder Singh, and Yishay Mansour. Policy gradient methods for reinforcement learning with function approximation. *Advances in neural information processing systems*, 12, 1999.

[69] David Silver, Guy Lever, Nicolas Heess, Thomas Degris, Daan Wierstra, and Martin Riedmiller. Deterministic policy gradient algorithms. In *International conference on machine learning*, pages 387–395. PMLR, 2014.

[70] John Schulman, Philipp Moritz, Sergey Levine, Michael Jordan, and Pieter Abbeel. High-dimensional continuous control using generalized advantage estimation. *arXiv preprint arXiv:1506.02438*, 2015.

[71] Christopher JCH Watkins and Peter Dayan. Q-learning. *Machine learning*, 8(3):279–292, 1992.

[72] Shixiang Gu, Timothy Lillicrap, Zoubin Ghahramani, Richard E Turner, and Sergey Levine. Q-prop: Sample-efficient policy gradient with an off-policy critic. *arXiv preprint arXiv:1611.02247*, 2016.

[73] Timothy P Lillicrap, Jonathan J Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*, 2015.

[74] Volodymyr Mnih, Adria Puigdomenech Badia, Mehdi Mirza, Alex Graves, Timothy Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. Asynchronous methods for deep reinforcement learning. In *International conference on machine learning*, pages 1928–1937. PMLR, 2016.

[75] Ofir Nachum, Mohammad Norouzi, Kelvin Xu, and Dale Schuurmans. Bridging the gap between value and policy based reinforcement learning. *Advances in neural information processing systems*, 30, 2017.

[76] Sergey Levine, Aviral Kumar, George Tucker, and Justin Fu. Offline reinforcement learning: Tutorial, review, and perspectives on open problems. *arXiv preprint arXiv:2005.01643*, 2020.

[77] Yifan Wu, George Tucker, and Ofir Nachum. Behavior regularized offline reinforcement learning. *arXiv preprint arXiv:1911.11361*, 2019.

[78] Aviral Kumar, Aurick Zhou, George Tucker, and Sergey Levine. Conservative q-learning for offline reinforcement learning. *Advances in Neural Information Processing Systems*, 33:1179–1191, 2020.

[79] Ashvin Nair, Murtaza Dalal, Abhishek Gupta, and Sergey Levine. Accelerating online reinforcement learning with offline datasets. *arXiv preprint arXiv:2006.09359*, 2020.

[80] Tengyang Xie, Nan Jiang, Huan Wang, Caiming Xiong, and Yu Bai. Policy finetuning: Bridging sample-efficient offline and online reinforcement learning. *Advances in neural information processing systems*, 34:27395–27407, 2021.

[81] Philip J Ball, Laura Smith, Ilya Kostrikov, and Sergey Levine. Efficient online reinforcement learning with offline data. *arXiv preprint arXiv:2302.02948*, 2023.

[82] Levente Kocsis, Csaba Szepesvári, and Jan Willemson. Improved monte-carlo search. *Univ. Tartu, Estonia, Tech. Rep*, 1, 2006.

[83] Oskar Von Stryk and Roland Bulirsch. Direct and indirect methods for trajectory optimization. *Annals of operations research*, 37(1):357–373, 1992.

[84] Hans Georg Bock and Karl-Josef Plitt. A multiple shooting algorithm for direct solution of optimal control problems. *IFAC Proceedings Volumes*, 17(2):1603–1608, 1984.

[85] Oskar Von Stryk. Numerical solution of optimal control problems by direct collocation. In *Optimal control*, pages 129–143. Springer, 1993.

[86] Eduardo F Camacho and Carlos Bordons Alba. *Model predictive control*. Springer science & business media, 2013.

[87] Kenji Doya, Kazuyuki Samejima, Ken-ichi Katagiri, and Mitsuo Kawato. Multiple model-based reinforcement learning. *Neural computation*, 14(6):1347–1369, 2002.

[88] Marc Deisenroth and Carl E Rasmussen. Pilco: A model-based and data-efficient approach to policy search. In *Proceedings of the 28th International Conference on machine learning (ICML-11)*, pages 465–472, 2011.

[89] Yuval Tassa, Tom Erez, and Emanuel Todorov. Synthesis and stabilization of complex behaviors through online trajectory optimization. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 4906–4913. IEEE, 2012.

[90] Anusha Nagabandi, Gregory Kahn, Ronald S Fearing, and Sergey Levine. Neural network dynamics for model-based deep reinforcement learning with model-free fine-tuning. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 7559–7566. IEEE, 2018.

[91] Lukasz Kaiser, Mohammad Babaeizadeh, Piotr Milos, Blazej Osinski, Roy H Campbell, Konrad Czechowski, Dumitru Erhan, Chelsea Finn, Piotr Kozakowski, Sergey Levine, et al. Model-based reinforcement learning for atari. *arXiv preprint arXiv:1903.00374*, 2019.

[92] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

[93] Aaron Van Den Oord, Oriol Vinyals, et al. Neural discrete representation learning. *Advances in neural information processing systems*, 30, 2017.

[94] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[95] Diederik Kingma, Tim Salimans, Ben Poole, and Jonathan Ho. Variational diffusion models. *Advances in neural information processing systems*, 34:21696–21707, 2021.

[96] Yann LeCun, Sumit Chopra, Raia Hadsell, M Ranzato, and Fujie Huang. A tutorial on energy-based learning. *Predicting structured data*, 1(0), 2006.

[97] Yilun Du and Igor Mordatch. Implicit generation and generalization in energy-based models. *arXiv preprint arXiv:1903.08689*, 2019.

[98] Corey Lynch, Mohi Khansari, Ted Xiao, Vikash Kumar, Jonathan Tompson, Sergey Levine, and Pierre Sermanet. Learning latent plans from play. In *Conference on Robot Learning*, pages 1113–1132. PMLR, 2020.

[99] Nur Muhammad Mahi Shafiullah, Zichen Jeff Cui, Ariuntuya Altanzaya, and Lerrel Pinto. Behavior transformers: Cloning k modes with one stone. *arXiv preprint arXiv:2206.11251*, 2022.

[100] Juergen Schmidhuber. Reinforcement learning upside down: Don't predict rewards–just map them to actions. *arXiv preprint arXiv:1912.02875*, 2019.

[101] Aviral Kumar, Xue Bin Peng, and Sergey Levine. Reward-conditioned policies. *arXiv preprint arXiv:1912.13465*, 2019.

[102] David Brandfonbrener, Alberto Bietti, Jacob Buckman, Romain Laroche, and Joan Bruna. When does return-conditioned supervised learning work for offline reinforcement learning? *arXiv preprint arXiv:2206.01079*, 2022.

[103] Keiran Paster, Sheila McIlraith, and Jimmy Ba. You can't count on luck: Why decision transformers fail in stochastic environments. *arXiv preprint arXiv:2205.15967*, 2022.

[104] Mengjiao Yang, Dale Schuurmans, Pieter Abbeel, and Ofir Nachum. Dichotomy of control: Separating what you can control from what you cannot. *arXiv preprint arXiv:2210.13435*, 2022.

[105] Anurag Ajay, Yilun Du, Abhi Gupta, Joshua Tenenbaum, Tommi Jaakkola, and Pulkit Agrawal. Is conditional generative modeling all you need for decision-making? *arXiv preprint arXiv:2211.15657*, 2022.

[106] Jonathan Ho and Tim Salimans. Classifier-free diffusion guidance. *arXiv preprint arXiv:2207.12598*, 2022.

[107] Avi Singh, Huihan Liu, Gaoyue Zhou, Albert Yu, Nicholas Rhinehart, and Sergey Levine. Parrot: Data-driven behavioral priors for reinforcement learning. *arXiv preprint arXiv:2011.10024*, 2020.

[108] Jonathan Ho and Stefano Ermon. Generative adversarial imitation learning. *Advances in neural information processing systems*, 29, 2016.

[109] Qihang Zhang, Zhenghao Peng, and Bolei Zhou. Learning to drive by watching youtube videos: Action-conditioned contrastive policy pretraining. In *European Conference on Computer Vision*, pages 111–128. Springer, 2022.

[110] Bowen Baker, Ilge Akkaya, Peter Zhokhov, Joost Huizinga, Jie Tang, Adrien Ecoffet, Brandon Houghton, Raul Sampedro, and Jeff Clune. Video pretraining (vpt): Learning to act by watching unlabeled online videos. *arXiv preprint arXiv:2206.11795*, 2022.

[111] David Venuto, Sherry Yang, Pieter Abbeel, Doina Precup, Igor Mordatch, and Ofir Nachum. Multi-environment pretraining enables transfer to action limited datasets. *arXiv preprint arXiv:2211.13337*, 2022.

[112] Christopher Berner, Greg Brockman, Brooke Chan, Vicki Cheung, Przemyslaw Dkebiak, Christy Dennison, David Farhi, Quirin Fischer, Shariq Hashme, Chris Hesse, et al. Dota 2 with large scale deep reinforcement learning. *arXiv preprint arXiv:1912.06680*, 2019.

[113] Oriol Vinyals, Igor Babuschkin, Wojciech M Czarnecki, Michaël Mathieu, Andrew Dudzik, Junyoung Chung, David H Choi, Richard Powell, Timo Ewalds, Petko Georgiev, et al. Grandmaster level in starcraft ii using multi-agent reinforcement learning. *Nature*, 575(7782):350–354, 2019.

[114] Josh Tobin, Rachel Fong, Alex Ray, Jonas Schneider, Wojciech Zaremba, and Pieter Abbeel. Domain randomization for transferring deep neural networks from simulation to the real world. In *2017 IEEE/RSJ international conference on intelligent robots and systems (IROS)*, pages 23–30. IEEE, 2017.

[115] Jane X Wang, Zeb Kurth-Nelson, Dhruva Tirumala, Hubert Soyer, Joel Z Leibo, Remi Munos, Charles Blundell, Dharshan Kumaran, and Matt Botvinick. Learning to reinforcement learn. *arXiv preprint arXiv:1611.05763*, 2016.

[116] Yan Duan, John Schulman, Xi Chen, Peter L Bartlett, Ilya Sutskever, and Pieter Abbeel. Rl^2: Fast reinforcement learning via slow reinforcement learning. *arXiv preprint arXiv:1611.02779*, 2016.

[117] Shreyas Sundara Raman, Vanya Cohen, Eric Rosen, Ifrah Idrees, David Paulius, and Stefanie Tellex. Planning with large language models via corrective re-prompting. *arXiv preprint arXiv:2211.09935*, 2022.

[118] Danijar Hafner, Timothy Lillicrap, Mohammad Norouzi, and Jimmy Ba. Mastering atari with discrete world models. *arXiv preprint arXiv:2010.02193*, 2020.

[119] Chang Chen, Yi-Fu Wu, Jaesik Yoon, and Sungjin Ahn. Transdreamer: Reinforcement learning with transformer world models. *arXiv preprint arXiv:2202.09481*, 2022.

[120] Yizhe Zhu, Martin Renqiang Min, Asim Kadav, and Hans Peter Graf. S3vae: Self-supervised sequential vae for representation disentanglement and data generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6538–6547, 2020.

[121] Vincent Micheli, Eloi Alonso, and François Fleuret. Transformers are sample efficient world models. *arXiv preprint arXiv:2209.00588*, 2022.

[122] Sherjil Ozair, Yazhe Li, Ali Razavi, Ioannis Antonoglou, Aaron Van Den Oord, and Oriol Vinyals. Vector quantized models for planning. In *International Conference on Machine Learning*, pages 8302–8313. PMLR, 2021.

[123] Younggyo Seo, Kimin Lee, Stephen L James, and Pieter Abbeel. Reinforcement learning with action-free pre-training from videos. In *International Conference on Machine Learning*, pages 19561–19579. PMLR, 2022.

[124] Younggyo Seo, Danijar Hafner, Hao Liu, Fangchen Liu, Stephen James, Kimin Lee, and Pieter Abbeel. Masked world models for visual control. *arXiv preprint arXiv:2206.14244*, 2022.

[125] Jiankai Sun, De-An Huang, Bo Lu, Yun-Hui Liu, Bolei Zhou, and Animesh Garg. Plate: Visually-grounded planning with transformers in procedural tasks. *IEEE Robotics and Automation Letters*, 7(2):4924–4930, 2022.

[126] Sébastien Racanière, Théophane Weber, David Reichert, Lars Buesing, Arthur Guez, Danilo Jimenez Rezende, Adrià Puigdomènech Badia, Oriol Vinyals, Nicolas Heess, Yujia Li, et al. Imagination-augmented agents for deep reinforcement learning. *Advances in neural information processing systems*, 30, 2017.

[127] Danijar Hafner, Timothy Lillicrap, Jimmy Ba, and Mohammad Norouzi. Dream to control: Learning behaviors by latent imagination. *arXiv preprint arXiv:1912.01603*, 2019.

[128] Richard S Sutton. Integrated architectures for learning, planning, and reacting based on approximating dynamic programming. In *Machine learning proceedings 1990*, pages 216–224. Elsevier, 1990.

[129] Vladimir Feinberg, Alvin Wan, Ion Stoica, Michael I Jordan, Joseph E Gonzalez, and Sergey Levine. Model-based value estimation for efficient model-free reinforcement learning. *arXiv preprint arXiv:1803.00101*, 2018.

[130] Alekh Agarwal, Sham Kakade, and Lin F Yang. Model-based reinforcement learning with a generative model is minimax optimal. In *Conference on Learning Theory*, pages 67–83. PMLR, 2020.

[131] Yilun Du and Karthik Narasimhan. Task-agnostic dynamics priors for deep reinforcement learning. In *International Conference on Machine Learning*, 2019.

[132] Yilun Du, Toru Lin, and Igor Mordatch. Model based planning with energy based models. *CORL*, 2019.

[133] Michael Janner, Yilun Du, Joshua B Tenenbaum, and Sergey Levine. Planning with diffusion for flexible behavior synthesis. *arXiv preprint arXiv:2205.09991*, 2022.

[134] Yilun Du, Shuang Li, and Igor Mordatch. Compositional visual generation with energy based models. In *Advances in Neural Information Processing Systems*, 2020.

[135] Nan Liu, Shuang Li, Yilun Du, Antonio Torralba, and Joshua B Tenenbaum. Compositional visual generation with composable diffusion models. *arXiv preprint arXiv:2206.01714*, 2022.

[136] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. In *International conference on machine learning*, pages 1597–1607. PMLR, 2020.

[137] Kaiming He, Xinlei Chen, Saining Xie, Yanghao Li, Piotr Dollár, and Ross Girshick. Masked autoencoders are scalable vision learners. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 16000–16009, 2022.

[138] Pierre Sermanet, Corey Lynch, Yevgen Chebotar, Jasmine Hsu, Eric Jang, Stefan Schaal, Sergey Levine, and Google Brain. Time-contrastive networks: Self-supervised learning from video. In *2018 IEEE international conference on robotics and automation (ICRA)*, pages 1134–1141. IEEE, 2018.

[139] Ilya Kostrikov, Denis Yarats, and Rob Fergus. Image augmentation is all you need: Regularizing deep reinforcement learning from pixels. *arXiv preprint arXiv:2004.13649*, 2020.

[140] Michael Laskin, Aravind Srinivas, and Pieter Abbeel. Curl: Contrastive unsupervised representations for reinforcement learning. In *International Conference on Machine Learning*, pages 5639–5650. PMLR, 2020.

[141] Tete Xiao, Ilija Radosavovic, Trevor Darrell, and Jitendra Malik. Masked visual pre-training for motor control. *arXiv preprint arXiv:2203.06173*, 2022.

[142] Wenlong Huang, Pieter Abbeel, Deepak Pathak, and Igor Mordatch. Language models as zero-shot planners: Extracting actionable knowledge for embodied agents. *arXiv preprint arXiv:2201.07207*, 2022.

[143] Michael Ahn, Anthony Brohan, Noah Brown, Yevgen Chebotar, Omar Cortes, Byron David, Chelsea Finn, Keerthana Gopalakrishnan, Karol Hausman, Alex Herzog, et al. Do as I can, not as I say: Grounding language in robotic affordances. *arXiv preprint arXiv:2204.01691*, 2022.

[144] Zihao Wang, Shaofei Cai, Anji Liu, Xiaojian Ma, and Yitao Liang. Describe, explain, plan and select: Interactive planning with large language models enables open-world multi-task agents. *arXiv preprint arXiv:2302.01560*, 2023.

[145] Danny Driess, Fei Xia, Mehdi S. M. Sajjadi, Corey Lynch, Aakanksha Chowdhery, Brian Ichter, Ayzaan Wahid, Jonathan Tompson, Quan Vuong, Tianhe Yu, Wenlong Huang, Yevgen Chebotar, Pierre Sermanet, Daniel Duckworth, Sergey Levine, Vincent Vanhoucke, Karol Hausman, Marc Toussaint, Klaus Greff, Andy Zeng, Igor Mordatch, and Pete Florence. Palm-e: An embodied multimodal language model. In *arXiv preprint arXiv:2302.11111*, 2023.

[146] Allison C Tam, Neil C Rabinowitz, Andrew K Lampinen, Nicholas A Roy, Stephanie CY Chan, DJ Strouse, Jane X Wang, Andrea Banino, and Felix Hill. Semantic exploration from language abstractions and pretrained representations. *arXiv preprint arXiv:2204.05080*, 2022.

[147] Yuqing Du, Olivia Watkins, Zihan Wang, Cédric Colas, Trevor Darrell, Pieter Abbeel, Abhishek Gupta, and Jacob Andreas. Guiding pretraining in reinforcement learning with large language models. *arXiv preprint arXiv:2302.06692*, 2023.

[148] Xi Chen, Xiao Wang, Soravit Changpinyo, AJ Piergiovanni, Piotr Padlewski, Daniel Salz, Sebastian Goodman, Adam Grycner, Basil Mustafa, Lucas Beyer, et al. Pali: A jointly-scaled multilingual language-image model. *arXiv preprint arXiv:2209.06794*, 2022.

[149] Parsa Mahmoudieh, Deepak Pathak, and Trevor Darrell. Zero-shot reward specification via grounded natural language. In *ICLR 2022 Workshop on Generalizable Policy Learning in Physical World*, 2022.

[150] Machel Reid, Yutaro Yamada, and Shixiang Shane Gu. Can wikipedia help offline reinforcement learning? *arXiv preprint arXiv:2201.12122*, 2022.

[151] Shuang Li, Xavier Puig, Yilun Du, Clinton Wang, Ekin Akyurek, Antonio Torralba, Jacob Andreas, and Igor Mordatch. Pre-trained language models for interactive decision-making. *arXiv preprint arXiv:2202.01771*, 2022.

[152] Kevin Lu, Aditya Grover, Pieter Abbeel, and Igor Mordatch. Pretrained transformers as universal computation engines. *arXiv preprint arXiv:2103.05247*, 2021.

[153] Hao Liu, Lisa Lee, Kimin Lee, and Pieter Abbeel. Instruction-following agents with jointly pre-trained vision-language models. *arXiv preprint arXiv:2210.13431*, 2022.

[154] Yunfan Jiang, Agrim Gupta, Zichen Zhang, Guanzhi Wang, Yongqiang Dou, Yanjun Chen, Li Fei-Fei, Anima Anandkumar, Yuke Zhu, and Linxi Fan. Vima: General robot manipulation with multimodal prompts. *arXiv preprint arXiv:2210.03094*, 2022.

[155] DeepMind Interactive Agents Team, Josh Abramson, Arun Ahuja, Arthur Brussee, Federico Carnevale, Mary Cassin, Felix Fischer, Petko Georgiev, Alex Goldin, Tim Harley, et al. Creating multimodal interactive agents with imitation and self-supervised learning. *arXiv preprint arXiv:2112.03763*, 2021.

[156] Corey Lynch and Pierre Sermanet. Language conditioned imitation learning over unstructured data. *arXiv preprint arXiv:2005.07648*, 2020.

[157] Felix Hill, Sona Mokra, Nathaniel Wong, and Tim Harley. Human instruction-following with deep reinforcement learning via transfer-learning from text. *arXiv preprint arXiv:2005.09382*, 2020.

[158] Weituo Hao, Chunyuan Li, Xiujun Li, Lawrence Carin, and Jianfeng Gao. Towards learning a generic agent for vision-and-language navigation via pre-training. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 13137–13146, 2020.

[159] Arjun Majumdar, Ayush Shrivastava, Stefan Lee, Peter Anderson, Devi Parikh, and Dhruv Batra. Improving vision-and-language navigation with image-text pairs from the web. In *European Conference on Computer Vision*, pages 259–274. Springer, 2020.

[160] Suraj Nair, Eric Mitchell, Kevin Chen, Silvio Savarese, Chelsea Finn, et al. Learning language-conditioned robot behavior from offline data and crowd-sourced annotation. In *Conference on Robot Learning*, pages 1303–1315. PMLR, 2022.

[161] Eric Jang, Alex Irpan, Mohi Khansari, Daniel Kappler, Frederik Ebert, Corey Lynch, Sergey Levine, and Chelsea Finn. Bc-z: Zero-shot task generalization with robotic imitation learning. In *Conference on Robot Learning*, pages 991–1002. PMLR, 2022.

[162] Mohit Shridhar, Lucas Manuelli, and Dieter Fox. Cliport: What and where pathways for robotic manipulation. In *Conference on Robot Learning*, pages 894–906. PMLR, 2022.

[163] Pierre-Louis Guhur, Shizhe Chen, Ricardo Garcia, Makarand Tapaswi, Ivan Laptev, and Cordelia Schmid. Instruction-driven history-aware policies for robotic manipulations. *arXiv preprint arXiv:2209.04899*, 2022.

[164] Dhruv Shah, Blazej Osinski, Brian Ichter, and Sergey Levine. Lm-nav: Robotic navigation with large pre-trained models of language, vision, and action. *arXiv preprint arXiv:2207.04429*, 2022.

[165] Yuke Zhu, Ziyu Wang, Josh Merel, Andrei Rusu, Tom Erez, Serkan Cabi, Saran Tunyasuvunakool, János Kramár, Raia Hadsell, Nando de Freitas, et al. Reinforcement and imitation learning for diverse visuomotor skills. *arXiv preprint arXiv:1802.09564*, 2018.

[166] Pengfei Liu, Weizhe Yuan, Jinlan Fu, Zhengbao Jiang, Hiroaki Hayashi, and Graham Neubig. Pre-train, prompt, and predict: A systematic survey of prompting methods in natural language processing. *ACM Computing Surveys*, 55(9):1–35, 2023.

[167] Richard Dearden and Craig Boutilier. Abstraction and approximate decision-theoretic planning. *Artificial Intelligence*, 89(1-2):219–283, 1997.

[168] David Andre and Stuart J Russell. State abstraction for programmable reinforcement learning agents. In *Aaai/iaai*, pages 119–125, 2002.

[169] Shie Mannor, Ishai Menache, Amit Hoze, and Uri Klein. Dynamic abstraction in reinforcement learning via clustering. In *Proceedings of the twenty-first international conference on Machine learning*, page 71, 2004.

[170] David Abel, Dilip Arumugam, Lucas Lehnert, and Michael Littman. State abstractions for lifelong reinforcement learning. In *International Conference on Machine Learning*, pages 10–19. PMLR, 2018.

[171] Carles Gelada, Saurabh Kumar, Jacob Buckman, Ofir Nachum, and Marc G Bellemare. Deepmdp: Learning continuous latent space models for representation learning. In *International Conference on Machine Learning*, pages 2170–2179. PMLR, 2019.

[172] Rishabh Agarwal, Marlos C Machado, Pablo Samuel Castro, and Marc G Bellemare. Contrastive behavioral similarity embeddings for generalization in reinforcement learning. *arXiv preprint arXiv:2101.05265*, 2021.

[173] Deepak Pathak, Pulkit Agrawal, Alexei A Efros, and Trevor Darrell. Curiosity-driven exploration by self-supervised prediction. In *International Conference on Machine Learning*, pages 2778–2787. PMLR, 2017.

[174] Evan Shelhamer, Parsa Mahmoudieh, Max Argus, and Trevor Darrell. Loss is its own reward: Self-supervision for reinforcement learning. *arXiv preprint arXiv:1612.07307*, 2016.

[175] Junhyuk Oh, Satinder Singh, and Honglak Lee. Value prediction network. *arXiv preprint arXiv:1707.03497*, 2017.

[176] Yilun Du, Chuang Gan, and Phillip Isola. Curious representation learning for embodied intelligence. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 10408–10417, 2021.

[177] Norm Ferns, Prakash Panangaden, and Doina Precup. Metrics for finite markov decision processes. In *UAI*, volume 4, pages 162–169, 2004.

[178] Pablo Castro and Doina Precup. Using bisimulation for policy transfer in mdps. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 24, 2010.

[179] Amy Zhang, Rowan McAllister, Roberto Calandra, Yarin Gal, and Sergey Levine. Learning invariant representations for reinforcement learning without reconstruction. *arXiv preprint arXiv:2006.10742*, 2020.

[180] Yusuf Aytar, Tobias Pfaff, David Budden, Thomas Paine, Ziyu Wang, and Nando De Freitas. Playing hard exploration games by watching youtube. *Advances in neural information processing systems*, 31, 2018.

[181] Tung D Nguyen, Rui Shu, Tuan Pham, Hung Bui, and Stefano Ermon. Temporal predictive coding for model-based planning in latent space. In *International Conference on Machine Learning*, pages 8130–8139. PMLR, 2021.

[182] Mengjiao Yang and Ofir Nachum. Representation matters: Offline pretraining for sequential decision making. *arXiv preprint arXiv:2102.05815*, 2021.

[183] Fangchen Liu, Hao Liu, Aditya Grover, and Pieter Abbeel. Masked autoencoding for scalable and generalizable decision making. *arXiv preprint arXiv:2211.12740*, 2022.

[184] Micah Carroll, Orr Paradise, Jessy Lin, Raluca Georgescu, Mingfei Sun, David Bignell, Stephanie Milani, Katja Hofmann, Matthew Hausknecht, Anca Dragan, et al. Unimask: Unified inference in sequential decision problems. *arXiv preprint arXiv:2211.10869*, 2022.

[185] Bogdan Mazoure, Benjamin Eysenbach, Ofir Nachum, and Jonathan Tompson. Contrastive value learning: Implicit models for simple offline rl. *arXiv preprint arXiv:2211.02100*, 2022.

[186] Chi Jin, Zhuoran Yang, Zhaoran Wang, and Michael I Jordan. Provably efficient reinforcement learning with linear function approximation. In *Conference on Learning Theory*, pages 2137–2143. PMLR, 2020.

[187] Tianjun Zhang, Tongzheng Ren, Mengjiao Yang, Joseph Gonzalez, Dale Schuurmans, and Bo Dai. Making linear mdps practical via contrastive representation learning. In *International Conference on Machine Learning*, pages 26447–26466. PMLR, 2022.

[188] Aldo Pacchiano, Ofir Nachum, Nilseh Tripuraneni, and Peter Bartlett. Joint representation training in sequential tasks with shared structure. *arXiv preprint arXiv:2206.12441*, 2022.

[189] Tongzheng Ren, Chenjun Xiao, Tianjun Zhang, Na Li, Zhaoran Wang, Sujay Sanghavi, Dale Schuurmans, and Bo Dai. Latent variable representation for reinforcement learning. *arXiv preprint arXiv:2212.08765*, 2022.

[190] Daniel Adiwardana, Minh-Thang Luong, David R So, Jamie Hall, Noah Fiedel, Romal Thoppilan, Zi Yang, Apoorv Kulshreshtha, Gaurav Nemade, Yifeng Lu, et al. Towards a human-like open-domain chatbot. *arXiv preprint arXiv:2001.09977*, 2020.

[191] Stephen Roller, Emily Dinan, Naman Goyal, Da Ju, Mary Williamson, Yinhan Liu, Jing Xu, Myle Ott, Eric Michael Smith, Y-Lan Boureau, and Jason Weston. Recipes for building an open-domain chatbot. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 300–325, Online, April 2021. Association for Computational Linguistics.

[192] Reiichiro Nakano, Jacob Hilton, Suchir Balaji, Jeff Wu, Long Ouyang, Christina Kim, Christopher Hesse, Shantanu Jain, Vineet Kosaraju, William Saunders, et al. Webgpt: Browser-assisted question-answering with human feedback. *arXiv preprint arXiv:2112.09332*, 2021.

[193] Yuntao Bai, Andy Jones, Kamal Ndousse, Amanda Askell, Anna Chen, Nova DasSarma, Dawn Drain, Stanislav Fort, Deep Ganguli, Tom Henighan, et al. Training a helpful and harmless assistant with reinforcement learning from human feedback. *arXiv preprint arXiv:2204.05862*, 2022.

[194] Haoming Jiang, Bo Dai, Mengjiao Yang, Tuo Zhao, and Wei Wei. Towards automatic evaluation of dialog systems: A model-free off-policy evaluation approach. *arXiv preprint arXiv:2102.10242*, 2021.

[195] N. Jaques, S. Gu, D. Bahdanau, J. M. Hernandez-Lobato, R. E. Turner, and D. Eck. Sequence tutor: Conservative fine-tuning of sequence generation models with kl-control. *International Conference on Machine Learning (ICML)*, 2017.

[196] Siddharth Verma, Justin Fu, Mengjiao Yang, and Sergey Levine. Chai: A chatbot ai for task-oriented dialogue with offline reinforcement learning. *arXiv preprint arXiv:2204.08426*, 2022.

[197] Charlie Snell, Sherry Yang, Justin Fu, Yi Su, and Sergey Levine. Context-aware language modeling for goal-oriented dialogue systems. *arXiv preprint arXiv:2204.10198*, 2022.

[198] Youngsoo Jang, Jongmin Lee, and Kee-Eung Kim. GPT-critic: Offline reinforcement learning for end-to-end task-oriented dialogue systems. In *International Conference on Learning Representations*, 2022.

[199] Charlie Snell, Ilya Kostrikov, Yi Su, Mengjiao Yang, and Sergey Levine. Offline rl for natural language generation with implicit language q learning. *arXiv preprint arXiv:2206.11871*, 2022.

[200] Joshua Maynez, Shashi Narayan, Bernd Bohnet, and Ryan McDonald. On faithfulness and factuality in abstractive summarization. *arXiv preprint arXiv:2005.00661*, 2020.

[201] Ziwei Ji, Nayeon Lee, Rita Frieske, Tiezheng Yu, Dan Su, Yan Xu, Etsuko Ishii, Yejin Bang, Andrea Madotto, and Pascale Fung. Survey of hallucination in natural language generation. *ACM Computing Surveys*, 2022.

[202] Arkil Patel, Satwik Bhattamishra, and Navin Goyal. Are nlp models really able to solve simple math word problems? *arXiv preprint arXiv:2103.07191*, 2021.

[203] Maxwell Nye, Anders Johan Andreassen, Guy Gur-Ari, Henryk Michalewski, Jacob Austin, David Bieber, David Dohan, Aitor Lewkowycz, Maarten Bosma, David Luan, et al. Show your work: Scratchpads for intermediate computation with language models. *arXiv preprint arXiv:2112.00114*, 2021.

[204] Dale Schuurmans. Memory augmented large language models are computationally universal. *arXiv preprint arXiv:2301.04589*, 2023.

[205] Timo Schick, Jane Dwivedi-Yu, Roberto Dessì, Roberta Raileanu, Maria Lomeli, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. Toolformer: Language models can teach themselves to use tools. *arXiv preprint arXiv:2302.04761*, 2023.

[206] Zhengbao Jiang, Frank F. Xu, Jun Araki, and Graham Neubig. How can we know what language models know? *Transactions of the Association for Computational Linguistics*, 8:423–438, 2020.

[207] Timo Schick and Hinrich Schütze. Exploiting cloze-questions for few-shot text classification and natural language inference. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 255–269, Online, April 2021. Association for Computational Linguistics.

[208] Aaron Parisi, Yao Zhao, and Noah Fiedel. Talm: Tool augmented language models. *arXiv preprint arXiv:2205.12255*, 2022.

[209] Izzeddin Gur, Ofir Nachum, Yingjie Miao, Mustafa Safdari, Austin Huang, Aakanksha Chowdhery, Sharan Narang, Noah Fiedel, and Aleksandra Faust. Understanding html with large language models. *arXiv preprint arXiv:2210.03945*, 2022.

[210] Ofir Press, Muru Zhang, Sewon Min, Ludwig Schmidt, Noah A Smith, and Mike Lewis. Measuring and narrowing the compositionality gap in language models. *arXiv preprint arXiv:2210.03350*, 2022.

[211] Sawan Kumar and Partha Talukdar. Reordering examples helps during priming-based few-shot learning. *arXiv preprint arXiv:2106.01751*, 2021.

[212] Freda Shi, Xinyun Chen, Kanishka Misra, Nathan Scales, David Dohan, Ed Chi, Nathanael Schärli, and Denny Zhou. Large language models can be easily distracted by irrelevant context. *arXiv preprint arXiv:2302.00093*, 2023.

[213] Zhengbao Jiang, Jun Araki, Haibo Ding, and Graham Neubig. How can we know when language models know? on the calibration of language models for question answering. *Transactions of the Association for Computational Linguistics*, 9:962–977, 2021.

[214] Denny Zhou, Nathanael Schärli, Le Hou, Jason Wei, Nathan Scales, Xuezhi Wang, Dale Schuurmans, Olivier Bousquet, Quoc Le, and Ed Chi. Least-to-most prompting enables complex reasoning in large language models. *arXiv preprint arXiv:2205.10625*, 2022.

[215] Doina Precup, Richard S. Sutton, and Satinder P. Singh. Eligibility traces for off-policy policy evaluation. In *Proceedings of the 17th International Conference on Machine Learning*, pages 759–766, 2000.

[216] S. Murphy, M. van der Laan, and J. Robins. Marginal mean models for dynamic regimes. *Journal of American Statistical Association*, 96(456):1410–1423, 2001.

[217] P. Thomas, G. Theocharous, and M. Ghavamzadeh. High confidence off-policy evaluation. In *Proceedings of the 29th Conference on Artificial Intelligence*, 2015.

[218] Ofir Nachum, Yinlam Chow, Bo Dai, and Lihong Li. DualDICE: Behavior-agnostic estimation of discounted stationary distribution corrections. In *Advances in Neural Information Processing Systems*, pages 2315–2325, 2019.

[219] Masatoshi Uehara and Nan Jiang. Minimax weight and Q-function learning for off-policy evaluation. *arXiv preprint arXiv:1910.12809*, 2019.

[220] Ruiyi Zhang, Bo Dai, Lihong Li, and Dale Schuurmans. GenDICE: Generalized offline estimation of stationary values. In *International Conference on Learning Representations*, 2020.

[221] Shangtong Zhang, Bo Liu, and Shimon Whiteson. GradientDICE: Rethinking generalized offline estimation of stationary values. *arXiv preprint arXiv:2001.11113*, 2020.

[222] Ofir Nachum, Bo Dai, Ilya Kostrikov, Yinlam Chow, Lihong Li, and Dale Schuurmans. AlgaeDICE: Policy gradient from arbitrary experience, 2019.

[223] Sean P Meyn and Richard L Tweedie. *Markov Chains and Stochastic Stability*. Springer Science & Business Media, 2012.

[224] R Tyrrell Rockafellar. Augmented Lagrange multiplier functions and duality in non-convex programming. *SIAM Journal on Control*, 12(2):268–285, 1974.

[225] Leemon Baird. Residual algorithms: Reinforcement learning with function approximation. In *In Proceedings of the Twelfth International Conference on Machine Learning*, pages 30–37. Morgan Kaufmann, 1995.

[226] Bo Dai, Niao He, Yunpeng Pan, Byron Boots, and Le Song. Learning from conditional distributions via dual embeddings. *arXiv preprint arXiv:1607.04579*, 2016.

[227] Nan Jiang and Lihong Li. Doubly robust off-policy value evaluation for reinforcement learning. *arXiv preprint arXiv:1511.03722*, 2015.

[228] Michail G Lagoudakis and Ronald Parr. Least-squares policy iteration. *The Journal of Machine Learning Research*, 4:1107–1149, 2003.

[229] Mehrdad Farajtabar, Yinlam Chow, and Mohammad Ghavamzadeh. More robust doubly robust off-policy evaluation. *arXiv preprint arXiv:1802.03493*, 2018.

[230] Nathan Kallus and Masatoshi Uehara. Double reinforcement learning for efficient off-policy evaluation in Markov decision processes. *arXiv preprint arXiv:1908.08526*, 2019.

[231] R. Munos, T. Stepleton, A. Harutyunyan, and M. Bellemare. Safe and efficient off-policy reinforcement learning. In *Advances in Neural Information Processing Systems*, pages 1054–1062, 2016.

[232] Raphael Fonteneau, Susan A. Murphy, Louis Wehenkel, and Damien Ernst. Batch mode reinforcement learning based on the synthesis of artificial trajectories. *Annals of Operations Research*, 208(1):383–416, 2013.

[233] Yaqi Duan and Mengdi Wang. Minimax-optimal off-policy evaluation with linear function approximation, 2020. arXiv:2002.09516.

[234] Nathan Kallus and Masatoshi Uehara. Efficiently breaking the curse of horizon: Double reinforcement learning in infinite-horizon processes. *arXiv preprint arXiv:1909.05850*, 2019.

[235] Qiang Liu, Lihong Li, Ziyang Tang, and Dengyong Zhou. Breaking the curse of horizon: Infinite-horizon off-policy estimation. In *Advances in Neural Information Processing Systems*, pages 5356–5366, 2018.

[236] Ziyang Tang, Yihao Feng, Lihong Li, Dengyong Zhou, and Qiang Liu. Doubly robust bias reduction in infinite horizon off-policy estimation. In *Proceedings of the 8th International Conference on Learning Representations*, 2020.

[237] Bo Dai, Albert Shaw, Lihong Li, Lin Xiao, Niao He, Zhen Liu, Jianshu Chen, and Le Song. SBEED: Convergent reinforcement learning with nonlinear function approximation. In *Proceedings of the 35th International Conference on Machine Learning*, pages 1133–1142, 2018.

[238] Simon S Du, Jianshu Chen, Lihong Li, Lin Xiao, and Dengyong Zhou. Stochastic variance reduction methods for policy evaluation. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 1049–1058. JMLR. org, 2017.

[239] Bo Liu, Ji Liu, Mohammad Ghavamzadeh, Sridhar Mahadevan, and Marek Petrik. Finite-sample analysis of proximal gradient TD algorithms. In *Proc. The 31st Conf. Uncertainty in Artificial Intelligence, Amsterdam, Netherlands*, 2015.

[240] Joan Bas-Serrano and Gergely Neu. Faster saddle-point optimization for solving large-scale Markov decision processes, 2019. arXiv:1909.10904.

[241] Yichen Chen, Lihong Li, and Mengdi Wang. Scalable bilinear π learning using state and action features. *arXiv preprint arXiv:1804.10328*, 2018.

[242] Mengdi Wang. Randomized linear programming solves the discounted Markov decision problem in nearly-linear (sometimes sublinear) running time. *arXiv preprint arXiv:1704.01869*, 2017.

[243] Sascha Lange, Thomas Gabel, and Martin Riedmiller. Batch reinforcement learning. In *Reinforcement learning*, pages 45–73. Springer, 2012.

[244] Alon Halevy, Peter Norvig, and Fernando Pereira. The unreasonable effectiveness of data. *IEEE Intelligent Systems*, 24(2):8–12, 2009.

[245] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 25:1097–1105, 2012.

[246] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.

[247] Scott Fujimoto, David Meger, and Doina Precup. Off-policy deep reinforcement learning without exploration. In *International Conference on Machine Learning*, pages 2052–2062, 2019.

[248] Tianhe Yu, Garrett Thomas, Lantao Yu, Stefano Ermon, James Zou, Sergey Levine, Chelsea Finn, and Tengyu Ma. Mopo: Model-based offline policy optimization. *arXiv preprint arXiv:2005.13239*, 2020.

[249] Tatsuya Matsushima, Hiroki Furuta, Yutaka Matsuo, Ofir Nachum, and Shixiang Gu. Deployment-efficient reinforcement learning via model-based offline optimization. *arXiv preprint arXiv:2006.03647*, 2020.

[250] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013.

[251] David Silver, Thomas Hubert, Julian Schrittwieser, Ioannis Antonoglou, Matthew Lai, Arthur Guez, Marc Lanctot, Laurent Sifre, Dharshan Kumaran, Thore Graepel, et al. A general reinforcement learning algorithm that masters chess, shogi, and go through self-play. *Science*, 362(6419):1140–1144, 2018.

[252] Manuel Watter, Jost Tobias Springenberg, Joschka Boedecker, and Martin Riedmiller. Embed to control: A locally linear latent dynamics model for control from raw images. *arXiv preprint arXiv:1506.07365*, 2015.

[253] Nir Levine, Yinlam Chow, Rui Shu, Ang Li, Mohammad Ghavamzadeh, and Hung Bui. Prediction, consistency, curvature: Representation learning for locally-linear control. *arXiv preprint arXiv:1909.01506*, 2019.

[254] Evan Shelhamer, Parsa Mahmoudieh, Max Argus, and Trevor Darrell. Loss is its own reward: Self-supervision for reinforcement learning. *CoRR*, abs/1612.07307, 2016.

[255] Yifan Wu, George Tucker, and Ofir Nachum. The laplacian in rl: Learning representations with efficient approximations. *arXiv preprint arXiv:1810.04586*, 2018.

[256] Ofir Nachum, Shixiang Gu, Honglak Lee, and Sergey Levine. Near-optimal representation learning for hierarchical reinforcement learning. *arXiv preprint arXiv:1810.01257*, 2018.

[257] Aravind Srinivas, Michael Laskin, and Pieter Abbeel. Curl: Contrastive unsupervised representations for reinforcement learning, 2020.

[258] Adam Stooke, Kimin Lee, Pieter Abbeel, and Michael Laskin. Decoupling representation learning from reinforcement learning, 2020.

[259] Aaron van den Oord, Yazhe Li, and Oriol Vinyals. Representation learning with contrastive predictive coding, 2019.

[260] Yann LeCun and Fu Jie Huang. Loss functions for discriminative training of energy-based models. In *AIStats*, volume 6, page 34. Citeseer, 2005.

[261] Ahmed Hussein, Mohamed Medhat Gaber, Eyad Elyan, and Chrisina Jayne. Imitation learning: A survey of learning methods. *ACM Computing Surveys (CSUR)*, 50(2):1–35, 2017.

[262] Dean A Pomerleau. Efficient training of artificial neural networks for autonomous navigation. *Neural computation*, 3(1):88–97, 1991.

[263] Aviral Kumar, Justin Fu, Matthew Soh, George Tucker, and Sergey Levine. Stabilizing off-policy q-learning via bootstrapping error reduction. In *Advances in Neural Information Processing Systems*, 2019.

[264] Natasha Jaques, Asma Ghandeharioun, Judy Hanwen Shen, Craig Ferguson, Agata Lapedriza, Noah Jones, Shixiang Gu, and Rosalind Picard. Way off-policy batch deep reinforcement learning of implicit human preferences in dialog. *arXiv preprint arXiv:1907.00456*, 2019.

[265] Tuomas Haarnoja, Aurick Zhou, Kristian Hartikainen, George Tucker, Sehoon Ha, Jie Tan, Vikash Kumar, Henry Zhu, Abhishek Gupta, Pieter Abbeel, and Sergey Levine. Soft actor-critic algorithms and applications, 2019.

[266] Gabriel Dulac-Arnold, Daniel Mankowitz, and Todd Hester. Challenges of real-world reinforcement learning, 2019.

[267] Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. Momentum contrast for unsupervised visual representation learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9729–9738, 2020.

[268] Stefan Schaal. Is imitation learning the route to humanoid robots? *Trends in cognitive sciences*, 3(6):233–242, 1999.

[269] M Mehdi Afsar, Trafford Crump, and Behrouz Far. Reinforcement learning based recommender systems: A survey. *arXiv preprint arXiv:2101.06286*, 2021.

[270] Aviral Kumar, Justin Fu, George Tucker, and Sergey Levine. Stabilizing off-policy q-learning via bootstrapping error reduction. *arXiv preprint arXiv:1906.00949*, 2019.

[271] Pieter Abbeel and Andrew Y Ng. Apprenticeship learning via inverse reinforcement learning. In *Proceedings of the twenty-first international conference on Machine learning*, page 1, 2004.

[272] Ronald Parr and Stuart Russell. Reinforcement learning with hierarchies of machines. *Advances in neural information processing systems*, pages 1043–1049, 1998.

[273] Thomas G Dietterich et al. The maxq method for hierarchical reinforcement learning. In *ICML*, volume 98, pages 118–126. Citeseer, 1998.

[274] Richard S Sutton, Doina Precup, and Satinder Singh. Between mdps and semi-mdps: A framework for temporal abstraction in reinforcement learning. *Artificial intelligence*, 112(1-2):181–211, 1999.

[275] Tejas D Kulkarni, Karthik Narasimhan, Ardavan Saeedi, and Josh Tenenbaum. Hierarchical deep reinforcement learning: Integrating temporal abstraction and intrinsic motivation. *Advances in neural information processing systems*, 29:3675–3683, 2016.

[276] Alexander Sasha Vezhnevets, Simon Osindero, Tom Schaul, Nicolas Heess, Max Jaderberg, David Silver, and Koray Kavukcuoglu. Feudal networks for hierarchical reinforcement learning. In *International Conference on Machine Learning*, pages 3540–3549. PMLR, 2017.

[277] Ofir Nachum, Shixiang Gu, Honglak Lee, and Sergey Levine. Data-efficient hierarchical reinforcement learning. *arXiv preprint arXiv:1805.08296*, 2018.

[278] Kourosh Hakhamaneshi, Ruihan Zhao, Albert Zhan, Pieter Abbeel, and Michael Laskin. Hierarchical few-shot imitation with skill transition models. *arXiv preprint arXiv:2107.08981*, 2021.

[279] Ali Rahimi, Benjamin Recht, et al. Random features for large-scale kernel machines. In *NIPS*, volume 3, page 5. Citeseer, 2007.

[280] Benjamin Eysenbach, Abhishek Gupta, Julian Ibarz, and Sergey Levine. Diversity is all you need: Learning skills without a reward function. *arXiv preprint arXiv:1802.06070*, 2018.

[281] Archit Sharma, Shixiang Gu, Sergey Levine, Vikash Kumar, and Karol Hausman. Dynamics-aware unsupervised discovery of skills. *arXiv preprint arXiv:1907.01657*, 2019.

[282] Karol Gregor, Danilo Jimenez Rezende, and Daan Wierstra. Variational intrinsic control. *arXiv preprint arXiv:1611.07507*, 2016.

[283] David Warde-Farley, Tom Van de Wiele, Tejas Kulkarni, Catalin Ionescu, Steven Hansen, and Volodymyr Mnih. Unsupervised control through non-parametric discriminative rewards. *arXiv preprint arXiv:1811.11359*, 2018.

[284] Jinxin Liu, Donglin Wang, Qiangxing Tian, and Zhengyu Chen. Learn goal-conditioned policy with intrinsic motivation for deep reinforcement learning. *arXiv preprint arXiv:2104.05043*, 2021.

[285] Karol Hausman, Jost Tobias Springenberg, Ziyu Wang, Nicolas Heess, and Martin Riedmiller. Learning an embedding space for transferable robot skills. In *International Conference on Learning Representations*, 2018.

[286] Ofir Nachum, Michael Ahn, Hugo Ponte, Shixiang Gu, and Vikash Kumar. Multi-agent manipulation via locomotion using hierarchical sim2real. *arXiv preprint arXiv:1908.05224*, 2019.

[287] Pierre-Luc Bacon, Jean Harb, and Doina Precup. The option-critic architecture. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 31, 2017.

[288] Martin Stolle and Doina Precup. Learning options in reinforcement learning. In *International Symposium on abstraction, reformulation, and approximation*, pages 212–223. Springer, 2002.

[289] Xue Bin Peng, Michael Chang, Grace Zhang, Pieter Abbeel, and Sergey Levine. Mcp: Learning composable hierarchical control with multiplicative compositional policies. *arXiv preprint arXiv:1905.09808*, 2019.

[290] Thomas Kipf, Yujia Li, Hanjun Dai, Vinicius Zambaldi, Alvaro Sanchez-Gonzalez, Edward Grefenstette, Pushmeet Kohli, and Peter Battaglia. Compile: Compositional imitation learning and execution. In *International Conference on Machine Learning*, pages 3418–3428. PMLR, 2019.

[291] Wenxuan Zhou, Sujay Bajracharya, and David Held. Plas: Latent action space for offline reinforcement learning. *arXiv preprint arXiv:2011.07213*, 2020.

[292] Roy Fox, Sanjay Krishnan, Ion Stoica, and Ken Goldberg. Multi-level discovery of deep options. *arXiv preprint arXiv:1703.08294*, 2017.

[293] Sanjay Krishnan, Roy Fox, Ion Stoica, and Ken Goldberg. Ddco: Discovery of deep continuous options for robot learning from demonstrations. In *Conference on robot learning*, pages 418–437. PMLR, 2017.

[294] Tanmay Shankar and Abhinav Gupta. Learning robot skills with temporal variational inference. In *International Conference on Machine Learning*, pages 8624–8633. PMLR, 2020.

[295] Tanmay Shankar, Shubham Tulsiani, Lerrel Pinto, and Abhinav Gupta. Discovering motor programs by recomposing demonstrations. In *International Conference on Learning Representations*, 2019.

[296] Xiaofei Wang, Kimin Lee, Kourosh Hakhamaneshi, Pieter Abbeel, and Michael Laskin. Skill preferences: Learning to extract and execute robotic skills from human feedback. *arXiv preprint arXiv:2108.05382*, 2021.

[297] Satinder P Singh, Tommi Jaakkola, and Michael I Jordan. Reinforcement learning with soft state aggregation. *Advances in neural information processing systems*, pages 361–368, 1995.

[298] Zhiyuan Ren and Bruce H Krogh. State aggregation in markov decision processes. In *Proceedings of the 41st IEEE Conference on Decision and Control, 2002.*, volume 4, pages 3819–3824. IEEE, 2002.

[299] Sanjeev Arora, Simon Du, Sham Kakade, Yuping Luo, and Nikunj Saunshi. Provable representation learning for imitation learning via bi-level optimization. In *International Conference on Machine Learning*, pages 367–376. PMLR, 2020.

[300] Jonathan Chang, Masatoshi Uehara, Dhruv Sreenivas, Rahul Kidambi, and Wen Sun. Mitigating covariate shift in imitation learning via offline data with partial coverage. *Advances in Neural Information Processing Systems*, 34, 2021.

[301] Rafael Rafailov, Tianhe Yu, Aravind Rajeswaran, and Chelsea Finn. Visual adversarial imitation learning using variational models. *Advances in Neural Information Processing Systems*, 34, 2021.

[302] Stéphane Ross and Drew Bagnell. Efficient reductions for imitation learning. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pages 661–668. JMLR Workshop and Conference Proceedings, 2010.

[303] Stéphane Ross, Geoffrey Gordon, and Drew Bagnell. A reduction of imitation learning and structured prediction to no-regret online learning. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, pages 627–635. JMLR Workshop and Conference Proceedings, 2011.

[304] Stephen Tu, Alexander Robey, and Nikolai Matni. Closing the closed-loop distribution shift in safe imitation learning. *arXiv preprint arXiv:2102.09161*, 2021.

[305] Peter Jarvis. *Towards a comprehensive theory of human learning*. Routledge, 2012.

[306] George Katona. Organizing and memorizing: studies in the psychology of learning and teaching. 1940.

[307] Michael L Crawford. Teaching contextually. *Research, rationale, and techniques for improving student motivation and achievement in mathematics and science. Texas: Cord*, 2001.

[308] Heidi L Lujan and Stephen E DiCarlo. Too much teaching, not enough learning: what is the solution? *Advances in physiology education*, 30(1):17–22, 2006.

[309] James Bruce and Manuela M Veloso. Real-time randomized path planning for robot navigation. In *Robot soccer world cup*, pages 288–295. Springer, 2002.

[310] Richard M Murray, Zexiang Li, and S Shankar Sastry. *A mathematical introduction to robotic manipulation*. CRC press, 2017.

[311] Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361*, 2020.

[312] Charles Thorpe and Jay Gowdy. Annotated maps for autonomous land vehicles. In *1990 IEEE International Conference on Systems, Man, and Cybernetics Conference Proceedings*, pages 282–288. IEEE, 1990.

[313] Tristan Cazenave and Nicolas Jouandeau. A parallel monte-carlo tree search algorithm. In *International Conference on Computers and Games*, pages 72–80. Springer, 2008.

[314] Peter Auer, Nicolo Cesa-Bianchi, and Paul Fischer. Finite-time analysis of the multi-armed bandit problem. *Machine learning*, 47(2):235–256, 2002.

[315] Sébastien Bubeck and Nicolo Cesa-Bianchi. Regret analysis of stochastic and non-stochastic multi-armed bandit problems. *arXiv preprint arXiv:1204.5721*, 2012.

[316] Alekh Agarwal, Daniel Hsu, Satyen Kale, John Langford, Lihong Li, and Robert Schapire. Taming the monster: A fast and simple algorithm for contextual bandits. In *International Conference on Machine Learning*, pages 1638–1646. PMLR, 2014.

[317] Peter Auer, Thomas Jaksch, and Ronald Ortner. Near-optimal regret bounds for reinforcement learning. *Advances in neural information processing systems*, 21, 2008.

[318] Alexander L Strehl, Lihong Li, and Michael L Littman. Reinforcement learning in finite mdps: Pac analysis. *Journal of Machine Learning Research*, 10(11), 2009.

[319] Christoph Dann, Tor Lattimore, and Emma Brunskill. Unifying pac and regret: Uniform pac bounds for episodic reinforcement learning. *Advances in Neural Information Processing Systems*, 30, 2017.

[320] Mohammad Gheshlaghi Azar, Ian Osband, and Rémi Munos. Minimax regret bounds for reinforcement learning. In *International Conference on Machine Learning*, pages 263–272. PMLR, 2017.

[321] Chiyuan Zhang, Oriol Vinyals, Remi Munos, and Samy Bengio. A study on overfitting in deep reinforcement learning. *arXiv preprint arXiv:1804.06893*, 2018.

[322] Karl Cobbe, Chris Hesse, Jacob Hilton, and John Schulman. Leveraging procedural generation to benchmark reinforcement learning. In *International conference on machine learning*, pages 2048–2056. PMLR, 2020.

[323] Matthew Hausknecht and Peter Stone. The impact of determinism on learning atari 2600 games. In *Workshops at the Twenty-Ninth AAAI Conference on Artificial Intelligence*, 2015.

[324] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *nature*, 518(7540):529–533, 2015.

[325] Arun Nair, Praveen Srinivasan, Sam Blackwell, Cagdas Alcicek, Rory Fearon, Alessandro De Maria, Vedavyas Panneershelvam, Mustafa Suleyman, Charles Beattie, Stig Petersen, et al. Massively parallel methods for deep reinforcement learning. *arXiv preprint arXiv:1507.04296*, 2015.

[326] Marlos C Machado, Marc G Bellemare, Erik Talvitie, Joel Veness, Matthew Hausknecht, and Michael Bowling. Revisiting the arcade learning environment: Evaluation protocols and open problems for general agents. *Journal of Artificial Intelligence Research*, 61:523–562, 2018.

[327] Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. Openai gym. *arXiv preprint arXiv:1606.01540*, 2016.

[328] Jesse Farebrother, Marlos C Machado, and Michael Bowling. Generalization and regularization in dqn. *arXiv preprint arXiv:1810.00123*, 2018.

[329] Karl Cobbe, Oleg Klimov, Chris Hesse, Taehoon Kim, and John Schulman. Quantifying generalization in reinforcement learning. In *International Conference on Machine Learning*, pages 1282–1289. PMLR, 2019.

[330] Kimin Lee, Kibok Lee, Jinwoo Shin, and Honglak Lee. Network randomization: A simple technique for generalization in deep reinforcement learning. *arXiv preprint arXiv:1910.05396*, 2019.

[331] Hoang Le, Nan Jiang, Alekh Agarwal, Miroslav Dudik, Yisong Yue, and Hal Daumé III. Hierarchical imitation and reinforcement learning. In *International conference on machine learning*, pages 2917–2926. PMLR, 2018.

[332] Michael H Lim, Andy Zeng, Brian Ichter, Maryam Bandari, Erwin Coumans, Claire Tomlin, Stefan Schaal, and Aleksandra Faust. Multi-task learning with sequence-conditioned transporter networks. *arXiv preprint arXiv:2109.07578*, 2021.

[333] Mengjiao Yang, Sergey Levine, and Ofir Nachum. Trail: Near-optimal imitation learning with suboptimal data. *arXiv preprint arXiv:2110.14770*, 2021.

[334] Arun Venkatraman, Martial Hebert, and J Andrew Bagnell. Improving multi-step prediction of learned time series models. In *Twenty-Ninth AAAI Conference on Artificial Intelligence*, 2015.

[335] Long-Ji Lin and Tom M Mitchell. *Memory approaches to reinforcement learning in non-Markovian domains*. Citeseer, 1992.

[336] Xiujun Li, Lihong Li, Jianfeng Gao, Xiaodong He, Jianshu Chen, Li Deng, and Ji He. Recurrent reinforcement learning: a hybrid approach. *arXiv preprint arXiv:1509.03044*, 2015.

[337] Max Jaderberg, Volodymyr Mnih, Wojciech Marian Czarnecki, Tom Schaul, Joel Z Leibo, David Silver, and Koray Kavukcuoglu. Reinforcement learning with unsupervised auxiliary tasks. *arXiv preprint arXiv:1611.05397*, 2016.

[338] Joonho Lee, Jemin Hwangbo, Lorenz Wellhausen, Vladlen Koltun, and Marco Hutter. Learning quadrupedal locomotion over challenging terrain. *Science robotics*, 5(47):eabc5986, 2020.

[339] Dian Chen, Brady Zhou, Vladlen Koltun, and Philipp Krähenbühl. Learning by cheating. In *Conference on Robot Learning*, pages 66–75. PMLR, 2020.

[340] Guillaume Lample and Devendra Singh Chaplot. Playing fps games with deep reinforcement learning. In *Thirty-First AAAI Conference on Artificial Intelligence*, 2017.

[341] Piotr Mirowski, Razvan Pascanu, Fabio Viola, Hubert Soyer, Andrew J Ballard, Andrea Banino, Misha Denil, Ross Goroshin, Laurent Sifre, Koray Kavukcuoglu, et al. Learning to navigate in complex environments. *arXiv preprint arXiv:1611.03673*, 2016.

[342] Andrew K Lampinen, Nicholas A Roy, Ishita Dasgupta, Stephanie CY Chan, Allison C Tam, James L McClelland, Chen Yan, Adam Santoro, Neil C Rabinowitz, Jane X Wang, et al. Tell me why!–explanations support learning of relational and causal structure. *arXiv preprint arXiv:2112.03753*, 2021.

[343] Wang Ling, Dani Yogatama, Chris Dyer, and Phil Blunsom. Program induction by rationale generation: Learning to solve and explain algebraic word problems. *arXiv preprint arXiv:1705.04146*, 2017.

[344] Peter Clark, Oyvind Tafjord, and Kyle Richardson. Transformers as soft reasoners over language. *arXiv preprint arXiv:2002.05867*, 2020.

[345] Mohammed Saeed, Naser Ahmadi, Preslav Nakov, and Paolo Papotti. Rulebert: Teaching soft rules to pre-trained language models. *arXiv preprint arXiv:2109.13006*, 2021.

[346] Nazneen Fatema Rajani, Bryan McCann, Caiming Xiong, and Richard Socher. Explain yourself! leveraging language models for commonsense reasoning. *arXiv preprint arXiv:1906.02361*, 2019.

[347] Alon Talmor, Oyvind Tafjord, Peter Clark, Yoav Goldberg, and Jonathan Berant. Teaching pre-trained models to systematically reason over implicit knowledge. 2020.

[348] Zhengzhong Liang, Steven Bethard, and Mihai Surdeanu. Explainable multi-hop verbal reasoning through internal monologue. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1225–1250, 2021.

[349] Eric Zelikman, Yuhuai Wu, and Noah D Goodman. Star: Bootstrapping reasoning with reasoning. *arXiv preprint arXiv:2203.14465*, 2022.

[350] Emilio Parisotto, Francis Song, Jack Rae, Razvan Pascanu, Caglar Gulcehre, Siddhant Jayakumar, Max Jaderberg, Raphael Lopez Kaufman, Aidan Clark, Seb Noury, et al. Stabilizing transformers for reinforcement learning. In *International Conference on Machine Learning*, pages 7487–7498. PMLR, 2020.

[351] Hiroki Furuta, Yutaka Matsuo, and Shixiang Shane Gu. Generalized decision transformer for offline hindsight information matching. *arXiv preprint arXiv:2111.10364*, 2021.

[352] Qinqing Zheng, Amy Zhang, and Aditya Grover. Online decision transformer. *arXiv preprint arXiv:2202.05607*, 2022.

[353] Steven Rabin. *Game AI pro: collected wisdom of game AI professionals*. CRC Press, 2013.

[354] Don Murray and James J Little. Using real-time stereo vision for mobile robot navigation. *autonomous robots*, 8(2):161–171, 2000.

[355] Wael Sabra, Matthew Khouzam, Arnaud Chanu, and Sylvain Martel. Use of 3d potential field and an enhanced breadth-first search algorithms for the path planning of microdevices propelled in the cardiovascular system. In *2005 IEEE Engineering in Medicine and Biology 27th Annual Conference*, pages 3916–3920. IEEE, 2006.

[356] Soumabha Bhowmick, Abhishek Pant, Jayanta Mukherjee, and Alok Kanti Deb. A novel floor segmentation algorithm for mobile robot navigation. In *2015 Fifth National Conference on Computer Vision, Pattern Recognition, Image Processing and Graphics (NCVPRIPG)*, pages 1–4. IEEE, 2015.

[357] Hrudaya Kumar Tripathy, Sushruta Mishra, Hiren Kumar Thakkar, and Deepak Rai. Care: a collision-aware mobile robot navigation in grid environment using improved breadth first search. *Computers & Electrical Engineering*, 94:107327, 2021.

[358] Aditya Ganapathi, Pete Florence, Jake Varley, Kaylee Burns, Ken Goldberg, and Andy Zeng. Implicit kinematic policies: Unifying joint and cartesian action spaces in end-to-end robot learning. *arXiv preprint arXiv:2203.01983*, 2022.

[359] Kenny Young and Tian Tian. Minatar: An atari-inspired testbed for thorough and reproducible reinforcement learning experiments. *arXiv preprint arXiv:1903.03176*, 2019.

[360] Marc G Bellemare, Yavar Naddaf, Joel Veness, and Michael Bowling. The arcade learning environment: An evaluation platform for general agents. *Journal of Artificial Intelligence Research*, 47:253–279, 2013.

[361] Scott Emmons, Benjamin Eysenbach, Ilya Kostrikov, and Sergey Levine. Rvs: What is essential for offline rl via supervised learning? *arXiv preprint arXiv:2112.10751*, 2021.

[362] Rupesh Kumar Srivastava, Pranav Shyam, Filipe Mutz, Wojciech Jaśkowski, and Jürgen Schmidhuber. Training agents using upside-down reinforcement learning. *arXiv preprint arXiv:1912.02877*, 2019.

[363] Felipe Codevilla, Matthias Müller, Antonio López, Vladlen Koltun, and Alexey Dosovitskiy. End-to-end driving via conditional imitation learning. In *2018 IEEE international conference on robotics and automation (ICRA)*, pages 4693–4700. IEEE, 2018.

[364] Dibya Ghosh, Abhishek Gupta, Ashwin Reddy, Justin Fu, Coline Devin, Benjamin Eysenbach, and Sergey Levine. Learning to reach goals via iterated supervised learning. *arXiv preprint arXiv:1912.06088*, 2019.

[365] Aviral Kumar, Joey Hong, Anikait Singh, and Sergey Levine. Should i run offline reinforcement learning or behavioral cloning? In *International Conference on Learning Representations*, 2021.

[366] David Venuto, Elaine Lau, Doina Precup, and Ofir Nachum. Policy gradients incorporating the future. *arXiv preprint arXiv:2108.02096*, 2021.

[367] Keiran Paster, Sheila A McIlraith, and Jimmy Ba. Planning from pixels using inverse dynamics models. *arXiv preprint arXiv:2012.02419*, 2020.

[368] Miroslav Štrupl, Francesco Faccio, Dylan R Ashley, Jürgen Schmidhuber, and Rupesh Kumar Srivastava. Upside-down reinforcement learning can diverge in stochastic environments with episodic resets. *arXiv preprint arXiv:2205.06595*, 2022.

[369] Fred R Shapiro. Who wrote the serenity prayer? *The Chronicle Review*, 28, 2014.

[370] Adam R Villaflor, Zhe Huang, Swapnil Pande, John M Dolan, and Jeff Schneider. Addressing optimism bias in sequence modeling for reinforcement learning. In *International Conference on Machine Learning*, pages 22270–22283. PMLR, 2022.

[371] Anirudh Goyal, Alex Lamb, Jordan Hoffmann, Shagun Sodhani, Sergey Levine, Yoshua Bengio, and Bernhard Schölkopf. Recurrent independent mechanisms. *arXiv preprint arXiv:1909.10893*, 2019.

[372] Thomas Dietterich, George Trimponias, and Zhitang Chen. Discovering and removing exogenous state variables and rewards for reinforcement learning. In *International Conference on Machine Learning*, pages 1262–1270. PMLR, 2018.

[373] Benjamin Eysenbach, Soumith Udatha, Sergey Levine, and Ruslan Salakhutdinov. Imitating past successes can be very suboptimal. *arXiv preprint arXiv:2206.03378*, 2022.

[374] Uriel Singer, Adam Polyak, Thomas Hayes, Xi Yin, Jie An, Songyang Zhang, Qiyuan Hu, Harry Yang, Oron Ashual, Oran Gafni, et al. Make-a-video: Text-to-video generation without text-video data. *arXiv preprint arXiv:2209.14792*, 2022.

[375] Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, et al. Palm: Scaling language modeling with pathways. *arXiv preprint arXiv:2204.02311*, 2022.

[376] Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, Diego de Las Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, et al. Training compute-optimal large language models. *arXiv preprint arXiv:2203.15556*, 2022.

[377] Jean-Baptiste Alayrac, Jeff Donahue, Pauline Luc, Antoine Miech, Iain Barr, Yana Hasson, Karel Lenc, Arthur Mensch, Katie Millican, Malcolm Reynolds, et al. Flamingo: A visual language model for few-shot learning. *NeurIPS*, 2022.

[378] Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International conference on machine learning*, pages 1861–1870. PMLR, 2018.

[379] Diederik P. Kingma and Max Welling. Auto-encoding variational bayes. *CoRR*, abs/1312.6114, 2013.

[380] Bo Dai, Zhen Liu, Hanjun Dai, Niao He, Arthur Gretton, Le Song, and Dale Schuurmans. Exponential family estimation via adversarial dynamics embedding. *Advances in Neural Information Processing Systems*, 32, 2019.

[381] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in Neural Information Processing Systems*. 2014.

[382] Jascha Sohl-Dickstein, Eric Weiss, Niru Maheswaranathan, and Surya Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics. In *International Conference on Machine Learning*, 2015.

[383] Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising diffusion implicit models. *arXiv preprint arXiv:2010.02502*, 2020.

[384] Ruben Villegas, Mohammad Babaeizadeh, Pieter-Jan Kindermans, Hernan Moraldo, Han Zhang, Mohammad Taghi Saffar, Santiago Castro, Julius Kunze, and Dumitru Erhan. Phenaki: Variable length video generation from open domain textual description. *arXiv preprint arXiv:2210.02399*, 2022.

[385] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *The Journal of Machine Learning Research*, 21(1):5485–5551, 2020.

[386] Jiayuan Mao, Tomas Lozano-Perez, Joshua B. Tenenbaum, and Leslie Pack Kaelbing. PDSketch: Integrated Domain Programming, Learning, and Planning. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2022.

[387] Frederik Ebert, Yanlai Yang, Karl Schmeckpeper, Bernadette Bucher, Georgios Georgakis, Kostas Daniilidis, Chelsea Finn, and Sergey Levine. Bridge data: Boosting generalization of robotic skills with cross-domain datasets. *arXiv preprint arXiv:2109.13396*, 2021.

[388] Michael R Zhang, Thomas Paine, Ofir Nachum, Cosmin Paduraru, George Tucker, ziyu wang, and Mohammad Norouzi. Autoregressive dynamics models for offline policy evaluation and optimization. In *International Conference on Learning Representations*, 2021.

[389] Zhendong Wang, Jonathan J Hunt, and Mingyuan Zhou. Diffusion policies as an expressive policy class for offline reinforcement learning. *arXiv preprint arXiv:2208.06193*, 2022.

[390] Edwin Zhang, Yujie Lu, William Wang, and Amy Zhang. Lad: Language augmented diffusion for reinforcement learning. *arXiv preprint arXiv:2210.15629*, 2022.

[391] Julen Urain, Niklas Funk, Georgia Chalvatzaki, and Jan Peters. Se (3)-diffusionfields: Learning cost functions for joint grasp and motion optimization through diffusion. *arXiv preprint arXiv:2209.03855*, 2022.

[392] Ziyuan Zhong, Davis Rempe, Danfei Xu, Yuxiao Chen, Sushant Veer, Tong Che, Baishakhi Ray, and Marco Pavone. Guided conditional diffusion for controllable traffic simulation. *arXiv preprint arXiv:2210.17366*, 2022.

[393] Ivan Kapelyukh, Vitalis Vosylius, and Edward Johns. Dall-e-bot: Introducing web-scale diffusion models to robotics. *arXiv preprint arXiv:2210.02438*, 2022.

[394] Wenlong Huang, Fei Xia, Ted Xiao, Harris Chan, Jacky Liang, Pete Florence, Andy Zeng, Jonathan Tompson, Igor Mordatch, Yevgen Chebotar, et al. Inner monologue: Embodied reasoning through planning with language models. *arXiv preprint arXiv:2207.05608*, 2022.

[395] OpenAI. Gpt-4 technical report, 2023.

[396] Aditya Ramesh, Prafulla Dhariwal, Alex Nichol, Casey Chu, and Mark Chen. Hierarchical text-conditional image generation with clip latents. *arXiv preprint arXiv:2204.06125*, 2022.

[397] Christoph Schuhmann, Romain Beaumont, Richard Vencu, Cade Gordon, Ross Wightman, Mehdi Cherti, Theo Coombes, Aarush Katta, Clayton Mullis, Mitchell Wortsman, et al. Laion-5b: An open large-scale dataset for training next generation image-text models. *arXiv preprint arXiv:2210.08402*, 2022.

[398] Xiaohua Zhai, Alexander Kolesnikov, Neil Houlsby, and Lucas Beyer. Scaling vision transformers. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12104–12113, 2022.

[399] Jun Xu, Tao Mei, Ting Yao, and Yong Rui. Msr-vtt: A large video description dataset for bridging video and language. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5288–5296, 2016.

[400] Ranjay Krishna, Kenji Hata, Frederic Ren, Li Fei-Fei, and Juan Carlos Niebles. Dense-captioning events in videos. In *Proceedings of the IEEE international conference on computer vision*, pages 706–715, 2017.

[401] Antoine Miech, Dimitri Zhukov, Jean-Baptiste Alayrac, Makarand Tapaswi, Ivan Laptev, and Josef Sivic. Howto100m: Learning a text-video embedding by watching hundred million narrated video clips. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 2630–2640, 2019.

[402] Kristen Grauman, Andrew Westbury, Eugene Byrne, Zachary Chavis, Antonino Furnari, Rohit Girdhar, Jackson Hamburger, Hao Jiang, Miao Liu, Xingyu Liu, et al. Ego4d: Around the world in 3,000 hours of egocentric video. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 18995–19012, 2022.

[403] Sudeep Dasari, Frederik Ebert, Stephen Tian, Suraj Nair, Bernadette Bucher, Karl Schmeckpeper, Siddharth Singh, Sergey Levine, and Chelsea Finn. Robonet: Large-scale multi-robot learning. *arXiv preprint arXiv:1910.11215*, 2019.

[404] Ajay Mandlekar, Yuke Zhu, Animesh Garg, Jonathan Booher, Max Spero, Albert Tung, Julian Gao, John Emmons, Anchit Gupta, Emre Orbay, et al. Roboturk: A crowdsourcing platform for robotic skill learning through imitation. In *Conference on Robot Learning*, pages 879–893. PMLR, 2018.

[405] Marcin Andrychowicz, Filip Wolski, Alex Ray, Jonas Schneider, Rachel Fong, Peter Welinder, Bob McGrew, Josh Tobin, OpenAI Pieter Abbeel, and Wojciech Zaremba. Hindsight experience replay. *Advances in neural information processing systems*, 30, 2017.

[406] Richard S. Sutton. Learning to predict by the methods of temporal differences. *Machine Learning*, 1988.

[407] Andrei A Rusu, Matej Večerík, Thomas Rothörl, Nicolas Heess, Razvan Pascanu, and Raia Hadsell. Sim-to-real robot learning from pixels with progressive nets. In *Conference on robot learning*, pages 262–270. PMLR, 2017.

[408] Manolis Savva, Abhishek Kadian, Oleksandr Maksymets, Yili Zhao, Erik Wijmans, Bhavana Jain, Julian Straub, Jia Liu, Vladlen Koltun, Jitendra Malik, et al. Habitat: A platform for embodied ai research. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 9339–9347, 2019.

[409] Corey Lynch, Ayzaan Wahid, Jonathan Tompson, Tianli Ding, James Betker, Robert Baruch, Travis Armstrong, and Pete Florence. Interactive language: Talking to robots in real time. *IEEE Robotics and Automation Letters*, 2023.

[410] Dima Damen, Hazel Doughty, Giovanni Maria Farinella, Sanja Fidler, Antonino Furnari, Evangelos Kazakos, Davide Moltisanti, Jonathan Munro, Toby Perrett, Will Price, et al. Scaling egocentric vision: The epic-kitchens dataset. In *Proceedings of the European conference on computer vision (ECCV)*, pages 720–736, 2018.

[411] Raghav Goyal, Samira Ebrahimi Kahou, Vincent Michalski, Joanna Materzynska, Susanne Westphal, Heuna Kim, Valentin Haenel, Ingo Fruend, Peter Yianilos, Moritz Mueller-Freitag, et al. The" something something" video database for learning and evaluating visual common sense. In *Proceedings of the IEEE international conference on computer vision*, pages 5842–5850, 2017.

[412] Angel Chang, Angela Dai, Thomas Funkhouser, Maciej Halber, Matthias Niessner, Manolis Savva, Shuran Song, Andy Zeng, and Yinda Zhang. Matterport3d: Learning from rgb-d data in indoor environments. *arXiv preprint arXiv:1709.06158*, 2017.

[413] Christoph Schuhmann, Richard Vencu, Romain Beaumont, Robert Kaczmarczyk, Clayton Mullis, Aarush Katta, Theo Coombes, Jenia Jitsev, and Aran Komatsuzaki. Laion-400m: Open dataset of clip-filtered 400 million image-text pairs. *arXiv preprint arXiv:2111.02114*, 2021.

[414] Danny Driess, Fei Xia, Mehdi SM Sajjadi, Corey Lynch, Aakanksha Chowdhery, Brian Ichter, Ayzaan Wahid, Jonathan Tompson, Quan Vuong, Tianhe Yu, et al. Palm-e: An embodied multimodal language model. *arXiv preprint arXiv:2303.03378*, 2023.

[415] Anthony Brohan, Noah Brown, Justice Carbajal, Yevgen Chebotar, Xi Chen, Krzysztof Choromanski, Tianli Ding, Danny Driess, Avinava Dubey, Chelsea Finn, et al. Rt-2: Vision-language-action models transfer web knowledge to robotic control. *arXiv preprint arXiv:2307.15818*, 2023.

[416] Paulo Rauber, Avinash Ummadisingu, Filipe Mutz, and Jürgen Schmidhuber. Hindsight policy gradients. In *International Conference on Learning Representations*, 2019.

[417] David Silver, Thomas Hubert, Julian Schrittwieser, Ioannis Antonoglou, Matthew Lai, Arthur Guez, Marc Lanctot, Laurent Sifre, Dharshan Kumaran, Thore Graepel, Timothy P. Lillicrap, Karen Simonyan, and Demis Hassabis. Mastering chess and shogi by self-play with a general reinforcement learning algorithm. *arXiv preprint arXiv:1712.01815*.

[418] Gabriel Dulac-Arnold, Daniel Mankowitz, and Todd Hester. Challenges of real-world reinforcement learning. *arXiv preprint arXiv:1904.12901*, 2019.

[419] Xi Chen, Josip Djolonga, Piotr Padlewski, Basil Mustafa, Soravit Changpinyo, Jialin Wu, Carlos Riquelme Ruiz, Sebastian Goodman, Xiao Wang, Yi Tay, et al. Pali-x: On scaling up a multilingual vision and language model. *arXiv preprint arXiv:2305.18565*, 2023.

[420] Xin Wang, Jiawei Wu, Junkun Chen, Lei Li, Yuan-Fang Wang, and William Yang Wang. Vatex: A large-scale, high-quality multilingual dataset for video-and-language research. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 4581–4591, 2019.

[421] Mathew Monfort, SouYoung Jin, Alexander Liu, David Harwath, Rogerio Feris, James Glass, and Aude Oliva. Spoken moments: Learning joint audio-visual representations from video descriptions. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 14871–14881, 2021.

[422] Rohan Anil, Andrew M Dai, Orhan Firat, Melvin Johnson, Dmitry Lepikhin, Alexandre Passos, Siamak Shakeri, Emanuel Taropa, Paige Bailey, Zhifeng Chen, et al. Palm 2 technical report. *arXiv preprint arXiv:2305.10403*, 2023.

[423] Chenfei Wu, Lun Huang, Qianxi Zhang, Binyang Li, Lei Ji, Fan Yang, Guillermo Sapiro, and Nan Duan. Godiva: Generating open-domain videos from natural descriptions. *arXiv preprint arXiv:2104.14806*, 2021.

[424] Mengjiao Yang, Yilun Du, Bo Dai, Dale Schuurmans, Joshua B Tenenbaum, and Pieter Abbeel. Probabilistic adaptation of text-to-video models. *arXiv preprint arXiv:2306.01872*, 2023.

[425] Andreas Blattmann, Robin Rombach, Huan Ling, Tim Dockhorn, Seung Wook Kim, Sanja Fidler, and Karsten Kreis. Align your latents: High-resolution video synthesis with latent diffusion models. *arXiv preprint arXiv:2304.08818*, 2023.

[426] Chenshuang Zhang, Chaoning Zhang, Mengchun Zhang, and In So Kweon. Text-to-image diffusion model in generative ai: A survey. *arXiv preprint arXiv:2303.07909*, 2023.

[427] Yilun Du, Mengjiao Yang, Bo Dai, Hanjun Dai, Ofir Nachum, Joshua B Tenenbaum, Dale Schuurmans, and Pieter Abbeel. Learning universal policies via text-guided video generation, 2023. *URL https://arxiv. org/abs/2302.00111*, 2023.

[428] Lennart Ljung and Torkel Glad. *Modeling of dynamic systems*. Prentice-Hall, Inc., 1994.

[429] Richard S Sutton. Dyna, an integrated architecture for learning, planning, and reacting. *ACM Sigart Bulletin*, 2(4):160–163, 1991.

[430] Karl J. Åström and Björn Wittenmark. Adaptive control of linear time-invariant systems. *Automatica*, 9(6):551–564, 1973.

[431] Dimitri P. Bertsekas. *Dynamic Programming and Optimal Control*. Athena Scientific, 1995.

[432] Alessandro Achille and Stefano Soatto. A separation principle for control in the age of deep learning. *Annual Review of Control, Robotics, and Autonomous Systems*, 1:287–307, 2018.

[433] Timothée Lesort, Natalia Díaz-Rodríguez, Jean-Franois Goudou, and David Filliat. State representation learning for control: An overview. *Neural Networks*, 108:379–392, 2018.

[434] Pablo Samuel Castro. Scalable methods for computing state similarity in deterministic markov decision processes. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 10069–10076, 2020.

[435] Ziyi Wu, Nikita Dvornik, Klaus Greff, Thomas Kipf, and Animesh Garg. Slotformer: Unsupervised visual dynamics simulation with object-centric models. *arXiv preprint arXiv:2210.05861*, 2022.

[436] Danijar Hafner, Jurgis Pasukonis, Jimmy Ba, and Timothy Lillicrap. Mastering diverse domains through world models. *arXiv preprint arXiv:2301.04104*, 2023.

[437] Sihyun Yu, Kihyuk Sohn, Subin Kim, and Jinwoo Shin. Video probabilistic diffusion models in projected latent space. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 18456–18466, 2023.

[438] Daquan Zhou, Weimin Wang, Hanshu Yan, Weiwei Lv, Yizhe Zhu, and Jiashi Feng. Magicvideo: Efficient video generation with latent diffusion models. *arXiv preprint arXiv:2211.11018*, 2022.

[439] Aliaksandr Siarohin, Stéphane Lathuilière, Sergey Tulyakov, Elisa Ricci, and Nicu Sebe. Animating arbitrary objects via deep motion transfer. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2377–2386, 2019.

[440] Yaohui Wang, Di Yang, Francois Bremond, and Antitza Dantcheva. Latent image animator: Learning to animate images via latent space navigation. *arXiv preprint arXiv:2203.09043*, 2022.

[441] Chung-Yi Weng, Brian Curless, and Ira Kemelmacher-Shlizerman. Photo wake-up: 3d character animation from a single photo. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 5908–5917, 2019.

[442] Tianfan Xue, Jiajun Wu, Katherine L Bouman, and William T Freeman. Visual dynamics: Stochastic future generation via layered cross convolutional networks. *IEEE transactions on pattern analysis and machine intelligence*, 41(9):2236–2250, 2018.

[443] Yung-Yu Chuang, Dan B Goldman, Ke Colin Zheng, Brian Curless, David H Salesin, and Richard Szeliski. Animating pictures with stochastic motion textures. In *ACM SIGGRAPH 2005 Papers*, pages 853–860. 2005.

[444] Zhengqi Li, Richard Tucker, Noah Snavely, and Aleksander Holynski. Generative image dynamics, 2023.

[445] Zekun Hao, Xun Huang, and Serge Belongie. Controllable video generation with sparse trajectories. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 7854–7863, 2018.

[446] Wenyi Hong, Ming Ding, Wendi Zheng, Xinghan Liu, and Jie Tang. Cogvideo: Large-scale pretraining for text-to-video generation via transformers. *arXiv preprint arXiv:2205.15868*, 2022.

[447] Eder Santana and George Hotz. Learning a driving simulator. *arXiv preprint arXiv:1608.01230*, 2016.

[448] He Wang, Sören Pirk, Ersin Yumer, Vladimir G Kim, Ozan Sener, Srinath Sridhar, and Leonidas J Guibas. Learning a generative model for multi-step human-object interactions from videos. In *Computer Graphics Forum*, volume 38, pages 367–378. Wiley Online Library, 2019.

[449] Rinon Gal, Yuval Alaluf, Yuval Atzmon, Or Patashnik, Amit H Bermano, Gal Chechik, and Daniel Cohen-Or. An image is worth one word: Personalizing text-to-image generation using textual inversion. *arXiv preprint arXiv:2208.01618*, 2022.

[450] Nataniel Ruiz, Yuanzhen Li, Varun Jampani, Yael Pritch, Michael Rubinstein, and Kfir Aberman. Dreambooth: Fine tuning text-to-image diffusion models for subject-driven generation. *arXiv preprint arXiv:2208.12242*, 2022.

[451] Lvmin Zhang and Maneesh Agrawala. Adding conditional control to text-to-image diffusion models, 2023.

[452] James Seale Smith, Yen-Chang Hsu, Lingyu Zhang, Ting Hua, Zsolt Kira, Yilin Shen, and Hongxia Jin. Continual diffusion: Continual customization of text-to-image diffusion with c-lora. *arXiv preprint arXiv:2304.06027*, 2023.

[453] Nataniel Ruiz, Yuanzhen Li, Varun Jampani, Yael Pritch, Michael Rubinstein, and Kfir Aberman. Dreambooth: Fine tuning text-to-image diffusion models for subject-driven generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 22500–22510, 2023.

[454] Wenshuai Zhao, Jorge Peña Queralta, and Tomi Westerlund. Sim-to-real transfer in deep reinforcement learning for robotics: a survey. In *2020 IEEE symposium series on computational intelligence (SSCI)*, pages 737–744. IEEE, 2020.

[455] Pascal Vincent. A connection between score matching and denoising autoencoders. *Neural computation*, 23(7):1661–1674, 2011.

[456] Yang Song and Stefano Ermon. Generative modeling by estimating gradients of the data distribution. *Advances in neural information processing systems*, 32, 2019.

[457] Yilun Du, Conor Durkan, Robin Strudel, Joshua B Tenenbaum, Sander Dieleman, Rob Fergus, Jascha Sohl-Dickstein, Arnaud Doucet, and Will Grathwohl. Reduce, reuse, recycle: Compositional generation with energy-based diffusion models and mcmc. *arXiv preprint arXiv:2302.11552*, 2023.

[458] Anders Sjöberg, Jakob Lindqvist, Magnus Önnheim, Mats Jirstrand, and Lennart Svensson. Mcmc-correction of score-based diffusion models for model composition. *arXiv preprint arXiv:2307.14012*, 2023.

[459] Corey Lynch, Ayzaan Wahid, Jonathan Tompson, Tianli Ding, James Betker, Robert Baruch, Travis Armstrong, and Pete Florence. Interactive language: Talking to robots in real time. *arXiv preprint arXiv:2210.06407*, 2022.

[460] Alex Nichol, Prafulla Dhariwal, Aditya Ramesh, Pranav Shyam, Pamela Mishkin, Bob McGrew, Ilya Sutskever, and Mark Chen. Glide: Towards photorealistic image generation and editing with text-guided diffusion models. *arXiv preprint arXiv:2112.10741*, 2021.

[461] Yogesh Balaji, Seungjun Nah, Xun Huang, Arash Vahdat, Jiaming Song, Karsten Kreis, Miika Aittala, Timo Aila, Samuli Laine, Bryan Catanzaro, et al. ediffi: Text-to-image diffusion models with an ensemble of expert denoisers. *arXiv preprint arXiv:2211.01324*, 2022.

[462] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10684–10695, 2022.

[463] Chitwan Saharia, William Chan, Saurabh Saxena, Lala Li, Jay Whang, Emily Denton, Seyed Kamyar Seyed Ghasemipour, Burcu Karagol Ayan, S Sara Mahdavi, Rapha Gontijo Lopes, et al. Photorealistic text-to-image diffusion models with deep language understanding. *arXiv preprint arXiv:2205.11487*, 2022.

[464] Jiahui Yu, Yuanzhong Xu, Jing Yu Koh, Thang Luong, Gunjan Baid, Zirui Wang, Vijay Vasudevan, Alexander Ku, Yinfei Yang, Burcu Karagol Ayan, et al. Scaling autoregressive models for content-rich text-to-image generation. *arXiv preprint arXiv:2206.10789*, 2022.

[465] Huiwen Chang, Han Zhang, Jarred Barber, AJ Maschinot, Jose Lezama, Lu Jiang, Ming-Hsuan Yang, Kevin Murphy, William T Freeman, Michael Rubinstein, et al. Muse: Text-to-image generation via masked generative transformers. *arXiv preprint arXiv:2301.00704*, 2023.

[466] Chenfei Wu, Jian Liang, Lei Ji, Fan Yang, Yuejian Fang, Daxin Jiang, and Nan Duan. Nüwa: Visual synthesis pre-training for neural visual world creation. In *Computer Vision–ECCV 2022: 17th European Conference, Tel Aviv, Israel, October 23–27, 2022, Proceedings, Part XVI*, pages 720–736. Springer, 2022.

[467] Patrick Esser, Johnathan Chiu, Parmida Atighehchian, Jonathan Granskog, and Anastasis Germanidis. Structure and content-guided video synthesis with diffusion models. *arXiv preprint arXiv:2302.03011*, 2023.

[468] Amir Hertz, Ron Mokady, Jay Tenenbaum, Kfir Aberman, Yael Pritch, and Daniel Cohen-Or. Prompt-to-prompt image editing with cross attention control. *arXiv preprint arXiv:2208.01626*, 2022.

[469] Bahjat Kawar, Michael Elad, Stefano Ermon, and Jiaming Song. Denoising diffusion restoration models. *arXiv preprint arXiv:2201.11793*, 2022.

[470] Haoying Li, Yifan Yang, Meng Chang, Shiqi Chen, Huajun Feng, Zhihai Xu, Qi Li, and Yueting Chen. Srdiff: Single image super-resolution with diffusion probabilistic models. *Neurocomputing*, 479:47–59, 2022.

[471] Andreas Lugmayr, Martin Danelljan, Andres Romero, Fisher Yu, Radu Timofte, and Luc Van Gool. Repaint: Inpainting using denoising diffusion probabilistic models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11461–11471, 2022.

[472] Chenlin Meng, Yutong He, Yang Song, Jiaming Song, Jiajun Wu, Jun-Yan Zhu, and Stefano Ermon. Sdedit: Guided image synthesis and editing with stochastic differential equations. In *International Conference on Learning Representations*, 2021.

[473] Chitwan Saharia, William Chan, Huiwen Chang, Chris Lee, Jonathan Ho, Tim Salimans, David Fleet, and Mohammad Norouzi. Palette: Image-to-image diffusion models. In *ACM SIGGRAPH 2022 Conference Proceedings*, pages 1–10, 2022.

[474] Hiroshi Sasaki, Chris G Willcocks, and Toby P Breckon. Unit-ddpm: Unpaired image translation with denoising diffusion probabilistic models. *arXiv preprint arXiv:2104.05358*, 2021.

[475] Xuan Su, Jiaming Song, Chenlin Meng, and Stefano Ermon. Dual diffusion implicit bridges for image-to-image translation. In *The Eleventh International Conference on Learning Representations*, 2022.

[476] Eyal Molad, Eliahu Horwitz, Dani Valevski, Alex Rav Acha, Yossi Matias, Yael Pritch, Yaniv Leviathan, and Yedid Hoshen. Dreamix: Video diffusion models are general video editors. *arXiv preprint arXiv:2302.01329*, 2023.

[477] Jay Zhangjie Wu, Yixiao Ge, Xintao Wang, Weixian Lei, Yuchao Gu, Wynne Hsu, Ying Shan, Xiaohu Qie, and Mike Zheng Shou. Tune-a-video: One-shot tuning of image diffusion models for text-to-video generation. *arXiv preprint arXiv:2212.11565*, 2022.

[478] Rohan Dhesikan and Vignesh Rajmohan. Sketching the future (stf): Applying conditional control techniques to text-to-video models. *arXiv preprint arXiv:2305.05845*, 2023.

[479] Xiang Lisa Li and Percy Liang. Prefix-tuning: Optimizing continuous prompts for generation. *arXiv preprint arXiv:2101.00190*, 2021.

[480] Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. Lora: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685*, 2021.

[481] Weili Nie, Arash Vahdat, and Anima Anandkumar. Controllable and compositional generation with latent-space energy-based models. *Advances in Neural Information Processing Systems*, 34, 2021.

[482] Yilun Du, Shuang Li, Yash Sharma, B. Joshua Tenenbaum, and Igor Mordatch. Unsupervised learning of compositional energy concepts. In *Advances in Neural Information Processing Systems*, 2021.

[483] Nan Liu, Shuang Li, Yilun Du, Josh Tenenbaum, and Antonio Torralba. Learning to compose visual relations. *Advances in Neural Information Processing Systems*, 34:23166–23178, 2021.

[484] Zihao Wang, Lin Gui, Jeffrey Negrea, and Victor Veitch. Concept algebra for text-controlled vision models. *arXiv preprint arXiv:2302.03693*, 2023.

[485] Tailin Wu, Megan Tjandrasuwita, Zhengxuan Wu, Xuelin Yang, Kevin Liu, Rok Sosic, and Jure Leskovec. Zeroc: A neuro-symbolic model for zero-shot concept recognition and acquisition at inference time. *Advances in Neural Information Processing Systems*, 35:9828–9840, 2022.

[486] Yuntian Deng, Anton Bakhtin, Myle Ott, Arthur Szlam, and Marc'Aurelio Ranzato. Residual energy-based models for text generation. *arXiv preprint arXiv:2004.11714*, 2020.

[487] Julen Urain, Anqi Li, Puze Liu, Carlo D'Eramo, and Jan Peters. Composable energy policies for reactive motion generation and reinforcement learning. *arXiv preprint arXiv:2105.04962*, 2021.

[488] Nikolaos Gkanatsios, Ayush Jain, Zhou Xian, Yunchu Zhang, Christopher Atkeson, and Katerina Fragkiadaki. Energy-based models as zero-shot planners for compositional scene rearrangement. *arXiv preprint arXiv:2304.14391*, 2023.

[489] Rohit Gandikota, Joanna Materzynska, Jaden Fiotto-Kaufman, and David Bau. Erasing concepts from diffusion models. *arXiv preprint arXiv:2303.07345*, 2023.

[490] Ryan Po and Gordon Wetzstein. Compositional 3d scene generation using locally conditioned diffusion. *arXiv preprint arXiv:2303.12218*, 2023.

[491] H. Van Hasselt, A. Guez, and D. Silver. Deep reinforcement learning with double Q-learning. In *AAAI*, volume 16, pages 2094–2100, 2016.

[492] Prajit Ramachandran, Barret Zoph, and Quoc V. Le. Searching for activation functions, 2017.

[493] Joshua Achiam, David Held, Aviv Tamar, and Pieter Abbeel. Constrained policy optimization. In *International Conference on Machine Learning*, pages 22–31. PMLR, 2017.

[494] Daniel Berend and Aryeh Kontorovich. On the convergence of the empirical distribution. *arXiv preprint arXiv:1205.6711*, 2012.

[495] Matt Hoffman, Bobak Shahriari, John Aslanides, Gabriel Barth-Maron, Feryal Behbahani, Tamara Norman, Abbas Abdolmaleki, Albin Cassirer, Fan Yang, Kate Baumli, et al. Acme: A research framework for distributed reinforcement learning. *arXiv preprint arXiv:2006.00979*, 2020.

[496] Levente Kocsis and Csaba Szepesvári. Bandit based monte-carlo planning. In *European conference on machine learning*, pages 282–293. Springer, 2006.

[497] Aviv Tamar, Yi Wu, Garrett Thomas, Sergey Levine, and Pieter Abbeel. Value iteration networks. *Advances in neural information processing systems*, 29, 2016.

[498] Scott Reed and Nando De Freitas. Neural programmer-interpreters. *arXiv preprint arXiv:1511.06279*, 2015.

[499] Danfei Xu, Suraj Nair, Yuke Zhu, Julian Gao, Animesh Garg, Li Fei-Fei, and Silvio Savarese. Neural task programming: Learning to generalize across hierarchical tasks. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 3795–3802. IEEE, 2018.

[500] Yiming Ding, Carlos Florensa, Pieter Abbeel, and Mariano Phielipp. Goal-conditioned imitation learning. *Advances in neural information processing systems*, 32, 2019.

[501] Ajay Mandlekar, Fabio Ramos, Byron Boots, Silvio Savarese, Li Fei-Fei, Animesh Garg, and Dieter Fox. Iris: Implicit reinforcement without interaction at scale for learning control from offline robot manipulation data. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 4414–4420. IEEE, 2020.

[502] Aravind Srinivas, Allan Jabri, Pieter Abbeel, Sergey Levine, and Chelsea Finn. Universal planning networks: Learning generalizable representations for visuomotor control. In *International Conference on Machine Learning*, pages 4732–4741. PMLR, 2018.

[503] Martin L Puterman. *Markov decision processes: discrete stochastic dynamic programming*. John Wiley & Sons, 2014.

[504] Peter Anderson, Qi Wu, Damien Teney, Jake Bruce, Mark Johnson, Niko Sünderhauf, Ian Reid, Stephen Gould, and Anton Van Den Hengel. Vision-and-language navigation: Interpreting visually-grounded navigation instructions in real environments. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3674–3683, 2018.

[505] Chao Jia, Yinfei Yang, Ye Xia, Yi-Ting Chen, Zarana Parekh, Hieu Pham, Quoc Le, Yun-Hsuan Sung, Zhen Li, and Tom Duerig. Scaling up visual and vision-language representation learning with noisy text supervision. In *International conference on machine learning*, pages 4904–4916. PMLR, 2021.

[506] Saeed Saremi, Arash Mehrjou, Bernhard Schölkopf, and Aapo Hyvärinen. Deep energy estimator networks. *arXiv preprint arXiv:1805.08306*, 2018.

[507] Saeed Saremi and Aapo Hyvarinen. Neural empirical bayes. *arXiv preprint arXiv:1903.02334*, 2019.

[508] Ahmed El Alaoui, Andrea Montanari, and Mark Sellke. Sampling from the sherrington-kirkpatrick gibbs measure via algorithmic stochastic localization. In *2022 IEEE 63rd Annual Symposium on Foundations of Computer Science (FOCS)*, pages 323–334. IEEE, 2022.

Appendix A

Appendix

A.1 Appendix for Distribution Correction Estimation

A.1.1 Robustness Justification

We explain the robustness interpretation of the dual regularization as the perturbation of Bellman differences. In this section, we elaborate the robustness interpretation of the primal regularization. For simplicity, we also consider $f_1(\cdot) = (\cdot)^2$. Therefore, we have $\alpha_Q \cdot \mathbb{E}_{(s,a)\sim d^{\text{off}}}[f_1(Q(s,a))] = \alpha_Q \cdot \left\{\max_{\delta(s,a)} \langle Q, \delta\rangle - \mathbb{E}_{(s,a)\sim d^{\text{off}}}\left[\delta^2(s,a)\right]\right\}$. Plug the dual form into (3.10) and with strong duality, we have

$$\begin{aligned}\max_{\zeta\geq 0,\delta}\min_{Q,\lambda} L_D(\zeta, Q, \lambda, \delta) :=& (1-\gamma)\cdot \mathbb{E}_{\substack{a_0\sim\pi(s_0)\\ s_0\sim\mu}}[Q(s_0,a_0)] + \alpha_Q \mathbb{E}_{(s,a)\sim d^{\text{off}}}\left[\delta(s,a)\cdot Q(s,a)\right] + \lambda \\ &+ \mathbb{E}_{\substack{(s,a,r,s')\sim d^{\text{off}}\\ a'\sim\pi(s')}}[\zeta(s,a)\cdot(\alpha_R\cdot R(s,a) + \gamma Q(s',a') - Q(s,a) - \lambda)] \\ &- \alpha_Q\cdot\mathbb{E}_{(s,a)\sim d^{\text{off}}}[\delta^2(s,a)] - \alpha_\zeta\cdot\mathbb{E}_{(s,a)\sim d^{\text{off}}}[f_2(\zeta(s,a))], \end{aligned} \tag{A.1}$$

which can be understood as the Lagrangian of

$$\begin{aligned}\max_{\zeta\geq 0,\delta}\quad & \alpha_R\mathbb{E}_{(s,a)\sim d^{\text{off}}}\left[\zeta(s,a)\cdot R(s,a)\right] - \alpha_Q\cdot\mathbb{E}_{(s,a)\sim d^{\text{off}}}[\delta^2(s,a)] - \alpha_\zeta\cdot\mathbb{E}_{(s,a)\sim d^{\text{off}}}[f_2(\zeta(s,a))] \\ \text{s.t.}\quad & (1-\gamma)\,\mu_0\pi + \alpha_Q d^{\text{off}}\cdot\delta + \gamma\cdot\mathcal{P}_*^\pi\cdot\left(d^{\text{off}}\cdot\zeta\right) = \left(d^{\text{off}}\cdot\zeta\right) \\ & \mathbb{E}_{(s,a)\sim d^{\text{off}}}[\zeta] = 1. \end{aligned} \tag{A.2}$$

As we can see, the primal regularization actually introduces L_2-ball perturbations to the stationary state-action distribution condition (A.2). For different regularization, the perturbations will be in different dual spaces. For examples, with entropy-regularization, the perturbation lies in the simplex. The corresponding optimization of (3.11) is

$$\min_{Q}\quad (1-\gamma)\,\mathbb{E}_{\mu_0\pi}\left[Q(s,a)\right] + \alpha_Q\cdot\mathbb{E}_{(s,a)\sim d^{\text{off}}}\left[f_1(Q)\right] + \alpha_\zeta\cdot\mathbb{E}_{(s,a)\sim d^{\text{off}}}\left[\delta^2(s,a)\right] \tag{A.3}$$

$$\text{s.t.}\quad Q(s,a) \geq R(s,a) + \mathcal{B}Q(s,a) - \alpha_\zeta\delta(s,a). \tag{A.4}$$

In both (A.3) and (A.2), the relaxation of dual ζ in (A.2) does not affect the optimality of dual solution: the stationary state-action distribution is still the only solution to (A.2); while in (A.3), the relaxation of primal Q will lead to different optimal primal solution. From this view, one can also justify the advantages of the dual OPE estimation.

A.1.2 Proof for Theorem 4

The full enumeration of $\alpha_Q, \alpha_\zeta, \alpha_R, \lambda$, and $\zeta \gtrless 0$ results in $2^5 = 32$ configurations. We note that it is enough to characterize the solutions Q^*, ζ^* under these different configurations. Clearly, the primal estimator $\hat{\rho}_Q$ is unbiased when $Q^* = Q^\pi$, and the dual estimator $\hat{\rho}_\zeta$ is unbiased when $\zeta^* = d^\pi / d^{\text{off}}$. For the Lagrangian estimator $\hat{\rho}_{Q,\zeta}$, we may write it in two ways:

$$\hat{\rho}_{Q,\zeta}(\pi) = \hat{\rho}_Q(\pi) + \sum_{s,a} d^{\text{off}}(s,a)\zeta(s,a)(R(s,a) + \gamma \mathcal{P}^\pi Q(s,a) - Q(s,a)) \tag{A.5}$$

$$= \hat{\rho}_\zeta(\pi) + \sum_{s,a} Q(s,a)((1-\gamma)\mu(s)\pi(a|s) + \gamma \mathcal{P}^\pi_* d^{\text{off}} \times \zeta(s,a) - d^{\text{off}} \times \zeta(s,a)). \tag{A.6}$$

It is clear that when $Q^* = Q^\pi$, the second term of (A.5) is 0 and $\hat{\rho}_{Q,\zeta}(\pi) = \rho(\pi)$. When $\zeta^* = d^\pi / d^{\text{off}}$, the second term of (A.6) is 0 and $\hat{\rho}_{Q,\zeta}(\pi) = \rho(\pi)$. Therefore, the Lagrangian estimator is unbiased when either $Q^* = Q^\pi$ or $\zeta^* = d^\pi / d^{\text{off}}$.

Now we continue to characterizing Q^*, ζ^* under different configurations. First, when $\alpha_Q = 0, \alpha_\zeta = 0$, it is clear that the solutions are always unbiased by virtue of Theorem 3 (see also [222]). When $\alpha_Q > 0, \alpha_\zeta > 0$, the solutions are in general biased. We summarize the remaining configurations (in the discounted case) of $\alpha_Q > 0, \alpha_\zeta = 0$ and $\alpha_Q = 0, \alpha_\zeta > 0$ in the table below. We provide proofs for the configurations of the shaded cells. Proofs for the rest configurations can be found in [218, 222].

Proof. Under our Assumptions 1 and 2, the strong duality holds for (3.10). We provide the proofs by checking the configurations case-by-case.

- **iii)-iv)** In this configuration, the regularized Lagrangian (3.10) becomes

$$\max_{\zeta \geq 0} \min_{Q,\lambda} L_D(\zeta, Q, \lambda) := (1-\gamma) \cdot \mathbb{E}_{\substack{a_0 \sim \pi(s_0) \\ s_0 \sim \mu}}[Q(s_0, a_0)] + \alpha_Q \cdot \mathbb{E}_{(s,a) \sim d^{\text{off}}}[f_1(Q(s,a))] + \lambda$$
$$+ \mathbb{E}_{\substack{(s,a,r,s') \sim d^{\text{off}} \\ a' \sim \pi(s')}}[\zeta(s,a) \cdot (\gamma Q(s', a') - Q(s,a) - \lambda)],$$

  which is equivalent to

$$\max_{\zeta \geq 0} \min_{Q} L_D(\zeta, Q) = \left\langle (1-\gamma)\, \mu_0 \pi + \gamma \cdot \mathcal{P}^\pi_* \cdot \left(d^{\text{off}} \cdot \zeta\right) - d^{\text{off}} \cdot \zeta, Q \right\rangle + \alpha_Q \mathbb{E}_{d^{\text{off}}}\left[f_1\left(Q\right)\right]$$
$$\text{s.t.} \quad \mathbb{E}_{d^{\text{off}}}\left[\zeta\right] = 1. \tag{A.7}$$

Apply the Fenchel duality w.r.t. Q, we have

$$\max_{\zeta} \quad L_D\left(\zeta, Q^*\right) = -\alpha_Q \mathbb{E}_{d^{\text{off}}}\left[f_1^*\left(\frac{(1-\gamma)\mu_0\pi + \gamma\cdot\mathcal{P}_*^\pi\cdot\left(d^{\text{off}}\cdot\zeta\right) - d^{\text{off}}\cdot\zeta}{\alpha_Q d^{\text{off}}}\right)\right] \tag{A.8}$$

$$\text{s.t.} \qquad \mathbb{E}_{d^{\text{off}}}\left[\zeta\right] = 1. \tag{A.9}$$

If $f_1^*\left(\cdot\right)$ achieves the minimum at zero, it is obvious that

$$d^{\text{off}}\cdot\zeta^* = \left(1-\gamma\right)\mu_0\pi + \gamma\cdot\mathcal{P}_*^\pi\cdot\left(d^{\text{off}}\cdot\zeta^*\right) \Rightarrow d^{\text{off}}\cdot\zeta^* = d^\pi.$$

Therefore, we have

$$L\left(\zeta^*, Q^*\right) = -\alpha_Q f_1^*\left(0\right),$$

and

$$\begin{aligned} Q^* &= \operatorname*{argmax}_Q \left\langle \left(1-\gamma\right)\mu_0\pi + \gamma\cdot\mathcal{P}_*^\pi\cdot\left(d^{\text{off}}\cdot\zeta^*\right) - d^{\text{off}}\cdot\zeta^*, Q\right\rangle + \alpha_Q\mathbb{E}_{d^{\text{off}}}\left[f_1\left(Q\right)\right] \\ &= f_1^{*\prime}\left(0\right) \end{aligned}$$

- **i)-ii)** Following the derivation in case **iii)-iv)**, we have the regularized Lagrangian as almost the same as (A.7) but has an extra term $\alpha_R\mathbb{E}_{d^{\text{off}}}\left[\zeta\cdot R\right]$, *i.e.*

$$\begin{aligned} \max_{\zeta}\min_{Q} L_D(\zeta, Q) := \quad & \left(1-\gamma\right)\cdot\mathbb{E}_{\substack{a_0\sim\pi(s_0)\\ s_0\sim\mu}}[Q(s_0,a_0)] + \alpha_Q\cdot\mathbb{E}_{(s,a)\sim d^{\text{off}}}[f_1(Q(s,a))] \\ & + \mathbb{E}_{\substack{(s,a,r,s')\sim d^{\text{off}}\\ a'\sim\pi(s')}}[\zeta(s,a)\cdot(\alpha_R\cdot R\left(s,a\right) + \gamma Q(s',a') - Q(s,a))]. \end{aligned}$$

We first consider the case where the ζ is free and the normalization constraint is not enforced.

After applying the Fenchel duality w.r.t. Q, we have

$$\max_{\zeta} \quad L_D\left(\zeta, Q^*\right) = \alpha_R\left\langle d^{\text{off}}\cdot\zeta, R\right\rangle - \alpha_Q\mathbb{E}_{d^{\text{off}}}\left[f_1^*\left(\frac{d^{\text{off}}\cdot\zeta - (1-\gamma)\mu_0\pi - \gamma\cdot\mathcal{P}_*^\pi\cdot\left(d^{\text{off}}\cdot\zeta\right)}{\alpha_Q d^{\text{off}}}\right)\right] \tag{A.10}$$

We denote

$$\begin{aligned} \nu &= \frac{d^{\text{off}}\cdot\zeta - \left(1-\gamma\right)\mu_0\pi - \gamma\cdot\mathcal{P}_*^\pi\cdot\left(d^{\text{off}}\cdot\zeta\right)}{d^{\text{off}}} \\ \Rightarrow\ d^{\text{off}}\cdot\zeta &= \left(\mathcal{I} - \gamma\cdot\mathcal{P}_*^\pi\right)^{-1}\left(\left(1-\gamma\right)\mu_0\pi + d^{\text{off}}\cdot\nu\right), \end{aligned}$$

and thus,

$$\begin{aligned} L_D\left(\zeta^*, Q^*\right) &= \max_{\nu}\left\langle\left(\mathcal{I}-\gamma\cdot\mathcal{P}_*^\pi\right)^{-1}\left(\left(1-\gamma\right)\mu_0\pi + d^{\text{off}}\cdot\nu\right), \alpha_R R\right\rangle - \alpha_Q\mathbb{E}_{d^{\text{off}}}\left[f_1^*\left(\frac{\nu}{\alpha_Q}\right)\right] \\ &= \alpha_R\left(1-\gamma\right)\mathbb{E}_{\substack{a_0\sim\pi(s_0)\\ s_0\sim\mu}}\left[Q^\pi\left(s_0,a_0\right)\right] + \max_{\nu}\mathbb{E}_{d^{\text{off}}}\left[\nu\cdot\left(Q^\pi\right)\right] - \alpha_Q\mathbb{E}_{d^{\text{off}}}\left[f_1^*\left(\frac{\nu}{\alpha_Q}\right)\right], \\ &= \alpha_R\left(1-\gamma\right)\mathbb{E}_{\substack{a_0\sim\pi(s_0)\\ s_0\sim\mu}}\left[Q^\pi\left(s_0,a_0\right)\right] + \alpha_Q\mathbb{E}_{d^{\text{off}}}\left[f_1\left(Q^\pi\right)\right] \end{aligned}$$

where the second equation comes from the fact $Q^\pi = (\mathcal{I} - \mathcal{B})^{-1} R$ and last equation comes from Fenchel duality with $\nu^* = \alpha_Q f_1' (Q^\pi)$.

Then, we can characterize

$$\begin{aligned} \zeta^* &= \frac{(\mathcal{I} - \gamma \cdot \mathcal{P}_*^\pi)^{-1} ((1-\gamma)\mu_0\pi)}{d^{\text{off}}} + \alpha_Q \frac{(\mathcal{I} - \gamma \cdot \mathcal{P}_*^\pi)^{-1} \left(d^{\text{off}} \cdot f_1' (Q^\pi)\right)}{d^{\text{off}}} \\ &= \frac{d^\pi}{d^{\text{off}}} + \alpha_Q \frac{(\mathcal{I} - \gamma \cdot \mathcal{P}_*^\pi)^{-1} \left(d^{\text{off}} \cdot f_1' (Q^\pi)\right)}{d^{\text{off}}}, \end{aligned}$$

and

$$Q^* = (f_1')^{-1} \left(\frac{d^{\text{off}} \cdot \zeta^* - (1-\gamma)\mu_0\pi - \gamma \cdot \mathcal{P}_*^\pi \cdot \left(d^{\text{off}} \cdot \zeta^*\right)}{\alpha_Q d^{\text{off}}} \right) = Q^\pi.$$

If we have the positive constraint, *i.e.*, $\zeta \geq 0$, we denote

$$\exp(\nu) = \frac{(\mathcal{I} - \gamma \cdot \mathcal{P}_*^\pi) \left(d^{\text{off}} \cdot \zeta\right)}{d^{\text{off}}} \Rightarrow d^{\text{off}} \cdot \zeta = (\mathcal{I} - \gamma \cdot \mathcal{P}_*^\pi)^{-1} d^{\text{off}} \cdot \exp(\nu),$$

then,

$$L_D(\zeta^*, Q^*) = \max_\nu \mathbb{E}_{d^{\text{off}}} [\exp(\nu) \cdot Q^\pi] - \alpha_Q \mathbb{E}_{d^{\text{off}}} \left[f_1^* \left(\frac{1}{\alpha_Q} \left(\exp(\nu) - \frac{(1-\gamma)\mu_0\pi}{d^{\text{off}}} \right) \right) \right].$$

By first-order optimality condition, we have

$$\begin{aligned} & \exp(\nu^*) \left(Q^\pi - f_1^{*\prime} \left(\frac{1}{\alpha_Q} \left(\exp(\nu) - \frac{(1-\gamma)\mu_0\pi}{d^{\text{off}}} \right) \right) \right) = 0 \\ = \quad & \exp(\nu^*) = \left(\alpha_Q f_1' (Q^\pi) + \frac{(1-\gamma)\mu_0\pi}{d^{\text{off}}} \right)_+ \\ \Rightarrow \quad & d^{\text{off}} \cdot \zeta^* = (\mathcal{I} - \mathcal{B})^{-1} \cdot d^{\text{off}} \left(\alpha_Q f_1' (Q^\pi) + \frac{(1-\gamma)\mu_0\pi}{d^{\text{off}}} \right)_+ \\ \Rightarrow \quad & \zeta^* = \frac{1}{d^{\text{off}}} (\mathcal{I} - \mathcal{B})^{-1} \cdot d^{\text{off}} \left(\alpha_Q f_1' (Q^\pi) + \frac{(1-\gamma)\mu_0\pi}{d^{\text{off}}} \right)_+. \end{aligned} \tag{A.11}$$

For Q^*, we obtain from the Fenchel duality relationship,

$$\begin{aligned} Q^* &= f_1^{*\prime} \left(\frac{1}{\alpha_Q} \left(\exp(\nu^*) - \frac{(1-\gamma)\mu_0\pi}{d^{\text{off}}} \right) \right) \\ &= f_1^{*\prime} \left(\frac{1}{\alpha_Q} \left(\left(\alpha_Q f_1' (Q^\pi) + \frac{(1-\gamma)\mu_0\pi}{d^{\text{off}}} \right)_+ - \frac{(1-\gamma)\mu_0\pi}{d^{\text{off}}} \right) \right). \end{aligned} \tag{A.12}$$

Then, the $L_D(\zeta^*, Q^*)$ can be obtained by plugging (ζ^*, Q^*) in (A.11) and (A.12). Obviously, in this case, the estimators are all biased.

As we can see, in both **i)** and **ii)**, none of the optimal dual solution ζ^* satisfies the normalization condition. Therefore, with the extra normalization constraint, the optimization will be obviously biased.

- **v)-viii)** These cases are also proved in [222] and we provide a more succinct proof here. In these configurations, whether α_R is involved or not does not affect the proof. We will keep this component for generality. We ignore the $\zeta \geq 0$ and λ for simplicity, the conclusion does not affected, since the optimal solution ζ^* automatically satisfies these constraints.

  Consider the regularized Lagrangian (3.10) with such configuration, we have

$$\begin{aligned}\min_{Q}\max_{\zeta}\ L_D(\zeta, Q) := \quad & (1-\gamma)\cdot \mathbb{E}_{\substack{a_0\sim\pi(s_0)\\ s_0\sim\mu}}[Q(s_0,a_0)] - \alpha_\zeta \cdot \mathbb{E}_{(s,a)\sim d^{\text{off}}}[f_2(\zeta(s,a))] \\ & + \mathbb{E}_{\substack{(s,a,r,s')\sim d^{\text{off}}\\ a'\sim\pi(s')}}[\zeta(s,a)\cdot(\alpha_R\cdot R(s,a)+\gamma Q(s',a')-Q(s,a))] \end{aligned} \tag{A.13}$$

  Apply the Fenchal duality to ζ, we obtain

$$\min_{Q}\ L_D\left(\zeta^*, Q\right) := (1-\gamma)\cdot\mathbb{E}_{\substack{a_0\sim\pi(s_0)\\ s_0\sim\mu}}[Q(s_0,a_0)] + \alpha_\zeta \mathbb{E}_{d^{\text{off}}}\left[f_2^*\left(\frac{1}{\alpha_\zeta}\left(\mathcal{B}^\pi\cdot Q\left(s,a\right) - Q\left(s,a\right)\right)\right)\right], \tag{A.14}$$

  with $\mathcal{B}^\pi\cdot Q\left(s,a\right) := \alpha_R\cdot R\left(s,a\right)+\gamma\mathcal{P}^\pi Q\left(s,a\right)$. We denote $\nu\left(s,a\right) = \mathcal{B}\cdot Q\left(s,a\right) - Q\left(s,a\right)$, then, we have

$$Q\left(s,a\right) = \left(\mathcal{I}-\mathcal{B}\right)^{-1}\left(\alpha_R\cdot R - \nu\right).$$

  Plug this into (A.14), we have

$$\begin{aligned} L_D\left(\zeta^*, Q^*\right) &= \min_{\nu}\ (1-\gamma)\cdot\mathbb{E}_{\substack{a_0\sim\pi(s_0)\\ s_0\sim\mu}}\left[\left(\left(\mathcal{I}-\mathcal{B}\right)^{-1}\left(\alpha_R\cdot R-\nu\right)\right)(s_0,a_0)\right] \\ &\quad + \alpha_\zeta\mathbb{E}_{d^{\text{off}}}\left[f_2^*\left(\frac{1}{\alpha_\zeta}\nu\left(s,a\right)\right)\right], \\ &= \alpha_R\mathbb{E}_{d^\pi}\left[R\left(s,a\right)\right] - \alpha_\zeta\max_{\nu}\left(\mathbb{E}_{d^\pi}\left[\frac{\nu(s_0,a_0)}{\alpha_\zeta}\right] + \mathbb{E}_{d^{\text{off}}}\left[f_2^*\left(\frac{1}{\alpha_\zeta}\nu\left(s,a\right)\right)\right]\right), \\ &= \alpha_R\mathbb{E}_{d^\pi}\left[R\left(s,a\right)\right] - \alpha_\zeta D_f\left(d^\pi||d^{\text{off}}\right) \end{aligned} \tag{A.15}$$

  The second equation comes from the fact $d^\pi = \left(\mathcal{I}-\gamma\cdot\mathcal{P}_*^\pi\right)^{-1}(\mu\pi)$. The last equation is by the definition of the Fenchel duality of f-divergence. Meanwhile, the optimal $\frac{1}{\alpha_\zeta}\nu^* = f_2'\left(\frac{d^\pi}{d^{\text{off}}}\right)$. Then, we have

$$\begin{aligned} Q^* &= -\left(\mathcal{I}-\mathcal{B}\right)^{-1}\nu^* + \left(\mathcal{I}-\mathcal{B}\right)^{-1}\left(\alpha_R\cdot R\right) \\ &= -\alpha_\zeta\left(\mathcal{I}-\mathcal{B}\right)^{-1} f_2'\left(\frac{d^\pi}{d^{\text{off}}}\right) + \alpha_R Q^\pi, \end{aligned}$$

and

$$\begin{aligned}\zeta^*(s,a) &= \operatorname*{argmax}_{\zeta} \zeta \cdot \nu^*(s,a) - \alpha_\zeta f_2\left(\zeta\left(s,a\right)\right) \\ &= f_2^{*\prime}\left(\frac{1}{\alpha_\zeta}\nu^*\left(s,a\right)\right) = \frac{d^\pi\left(s,a\right)}{d^{\text{off}}\left(s,a\right)}.\end{aligned}$$

□

A.1.3 Recovering Existing OPE estimators

We verify the LSTDQ as a special case of the unified framework if the primal and dual are linearly parametrized, *i.e.*, $Q\left(s,a\right) = w^\top \phi\left(s,a\right)$ and $\tau\left(s,a\right) = v^\top \phi\left(s,a\right)$, from any unbiased estimator without $\xi \geq 0$ and λ. For simplicity, we assume the solution exists.

- When $(\alpha_Q = 1, \alpha_\zeta = 0, \alpha_R = 1)$, we have the estimator as

$$\begin{aligned}\max_v \min_w L_D(v,w) :=& (1-\gamma)\cdot w^\top \mathbb{E}_{\substack{a_0\sim\pi(s_0)\\ s_0\sim\mu}}[\phi(s_0,a_0)] + \alpha_Q \cdot \mathbb{E}_{(s,a)\sim d^{\text{off}}}[f_1(w^\top\phi(s,a))] \\ &+ v^\top \mathbb{E}_{\substack{(s,a,r,s')\sim d^{\text{off}}\\ a'\sim\pi(s')}}[\phi(s,a)\cdot(\alpha_R\cdot R(s,a) + \gamma w^\top\phi(s',a') - w^\top\phi(s,a))].\end{aligned}$$

  Then, we have the first-order optimality condition for v as

$$\begin{aligned}&\mathbb{E}_{\substack{(s,a,r,s')\sim d^{\text{off}}\\ a'\sim\pi(s')}}[\phi(s,a)\cdot(\alpha_R\cdot R(s,a) + \gamma w^\top\phi(s',a') - w^\top\phi(s,a))] = 0, \\ \Rightarrow\quad & w = \underbrace{\mathbb{E}_{\substack{(s,a,r,s')\sim d^{\text{off}}\\ a'\sim\pi(s')}}[\phi(s,a)\cdot(\phi(s,a) - \gamma\phi(s',a'))]^{-1}}_{\Xi}\mathbb{E}_{(s,a)\sim d^{\text{off}}}\left[\alpha_R\cdot R\left(s,a\right)\phi\left(s,a\right)\right], \\ \Rightarrow\quad & Q^*\left(s,a\right) = w^\top\phi\left(s,a\right),\end{aligned}$$

  which leads to

$$\begin{aligned}\hat{\rho}_Q(\pi) &= (1-\gamma)\cdot\mathbb{E}_{\substack{a_0\sim\pi(s_0)\\ s_0\sim\mu}}[\hat{Q}(s_0,a_0)] \\ &= (1-\gamma)\,\mathbb{E}_{\substack{a_0\sim\pi(s_0)\\ s_0\sim\mu}}[\phi\left(s,a\right)]^\top \Xi^{-1}\mathbb{E}_{(s,a)\sim d^{\text{off}}}\left[R\left(s,a\right)\phi\left(s,a\right)\right].\end{aligned}$$

- When $(\alpha_Q = 0, \alpha_\zeta = 1, \alpha_R = \{0/1\})$, we have the estimator as

$$\begin{aligned}\max_v \min_w L_D(v,w) :=& (1-\gamma)\cdot w^\top \mathbb{E}_{\substack{a_0\sim\pi(s_0)\\ s_0\sim\mu}}[\phi(s_0,a_0)] - \alpha_\zeta \cdot \mathbb{E}_{(s,a)\sim d^{\text{off}}}[f_2(v^\top\phi(s,a))] \\ &+ v^\top \mathbb{E}_{\substack{(s,a,r,s')\sim d^{\text{off}}\\ a'\sim\pi(s')}}[\phi(s,a)\cdot(\alpha_R\cdot R(s,a) + \gamma w^\top\phi(s',a') - w^\top\phi(s,a))].\end{aligned}$$

Then, we have the first-order optimality condition as

$$v^\top \mathbb{E}_{\substack{(s,a,r,s')\sim d^{\text{off}} \\ a'\sim\pi(s')}}[\phi(s,a)\cdot(\gamma\phi(s',a')-\phi(s,a))]+(1-\gamma)\cdot\mathbb{E}_{\substack{a_0\sim\pi(s_0) \\ s_0\sim\mu}}[\phi(s_0,a_0)]=0,$$

which leads to

$$v=(1-\gamma)\cdot\Xi^{-1}\mathbb{E}_{\substack{a_0\sim\pi(s_0) \\ s_0\sim\mu}}[\phi(s_0,a_0)].$$

Therefore, the dual estimator is

$$\begin{aligned}\hat{\rho}_\zeta(\pi) &= \mathbb{E}_{(s,a,r)\sim d^{\text{off}}}\left[R\cdot\phi(s,a)\right]^\top v \\ &= (1-\gamma)\,\mathbb{E}_{\substack{a_0\sim\pi(s_0) \\ s_0\sim\mu}}[\phi(s,a)]^\top\Xi^{-1}\mathbb{E}_{(s,a)\sim d^{\text{off}}}\left[R(s,a)\,\phi(s,a)\right].\end{aligned}$$

- When $(\alpha_Q=1,\alpha_\zeta=0,\alpha_R=0)$, by the conclusion for (A.7), we have

$$v^\top \mathbb{E}_{\substack{(s,a,r,s')\sim d^{\text{off}} \\ a'\sim\pi(s')}}[\phi(s,a)\cdot(\gamma\phi(s',a')-\phi(s,a))]+(1-\gamma)\cdot\mathbb{E}_{\substack{a_0\sim\pi(s_0) \\ s_0\sim\mu}}[\phi(s_0,a_0)]=0,$$

which leads to similar result as above case.

A.1.4 Alternative Biased Form

Unconstrained Primal Forms

When $\alpha_\zeta > 0$ and $\alpha_Q = 0$, the form of the Lagranian can be simplified to yield an optimization over only Q. Then, we may simplify,

$$\begin{aligned}&\max_{\zeta(s,a)}\zeta(s,a)\cdot(\alpha_R\cdot R(s,a)+\gamma\mathcal{P}^\pi Q(s,a)-Q(s,a))-\alpha_\zeta\cdot f_2(\zeta(s,a)) \\ &\qquad=\alpha_\zeta\cdot f_2^*\left(\frac{1}{\alpha_\zeta}(\alpha_R\cdot R(s,a)+\gamma\mathcal{P}^\pi Q(s,a)-Q(s,a))\right). \quad \text{(A.16)}\end{aligned}$$

So, the Lagrangian may be equivalently expressed as an optimization over only Q:

$$\begin{aligned}&\min_Q(1-\gamma)\cdot\mathbb{E}_{\substack{a_0\sim\pi(s_0) \\ s_0\sim\mu}}[Q(s_0,a_0)]+\alpha_Q\cdot\mathbb{E}_{(s,a)\sim d^{\text{off}}}[f_1(Q(s,a))] \\ &\qquad+\alpha_\zeta\cdot\mathbb{E}_{(s,a)\sim d^{\text{off}}}\left[f_2^*\left(\frac{1}{\alpha_\zeta}(\alpha_R\cdot R(s,a)+\gamma\mathcal{P}^\pi Q(s,a)-Q(s,a))\right)\right]. \quad \text{(A.17)}\end{aligned}$$

We call this the *unconstrained primal form*, since optimization is now exclusively over primal variables. Still, given a solution Q^*, the optimal ζ^* to the original Lagrangian may be derived as,

$$\zeta^*(s,a)=f_2^{*\prime}((\alpha_R\cdot R(s,a)+\gamma\mathcal{P}^\pi Q^*(s,a)-Q^*(s,a))/\alpha_\zeta). \quad \text{(A.18)}$$

Although the unconstrained primal form is simpler, in practice it presents a disadvantage, due to inaccessibility of the transition operator $\mathcal{P}^\pi$. That is, in practice, one must resort to optimizing the primal form as

$$\min_Q (1-\gamma)\cdot \mathbb{E}_{\substack{a_0\sim\pi(s_0)\\ s_0\sim\mu}}[Q(s_0,a_0)] + \alpha_Q \cdot \mathbb{E}_{(s,a)\sim d^{\text{off}}}[f_1(Q(s,a))] \\ + \alpha_\zeta \cdot \mathbb{E}_{\substack{(s,a,r,s')\sim d^{\text{off}}\\ a'\sim\pi(s')}}\left[f_2^*\left(\frac{1}{\alpha_\zeta}(\alpha_R\cdot R(s,a)+\gamma Q(s',a')-Q(s,a))\right)\right]. \quad \text{(A.19)}$$

This is in general a *biased* estimate of the true objective and thus leads to biased solutions, as the expectation over the next step samples are taken inside a square function (we choose f_2 to be the square function). Still, in some cases (e.g., in simple and discrete environments), the bias may be desirable as a trade-off in return for a simpler optimization.

Unconstrained Dual Form We have presented an unconstrained primal form. Similarly, we can derive the unconstrianed dual form by removing the primal variable with a particular primal regularization $\alpha_Q \mathbb{E}_{d^{\text{off}}}[f_1(Q)]$. Then, we can simplify

$$\min_{Q(s',a')} \frac{1}{d^{\text{off}}(s',a')}(1-\gamma)\mu(s')\pi(a'|s')\cdot Q(s',a') + \alpha_Q f_1(Q) \\ + \frac{1}{d^{\text{off}}(s',a')}\left(\gamma\int P^\pi(s',a'|s,a)\, d^{\text{off}}\cdot\zeta(s,a)\, dsda - d^{\text{off}}(s',a')\,\zeta(s',a')\right)\cdot Q(s',a') \\ = -\alpha_Q\cdot f_1^*\left(\frac{d^{\text{off}}\cdot\zeta-(1-\gamma)\mu\pi-\gamma\left(\mathcal{P}_*^\pi\cdot d^{\text{off}}\right)\zeta}{\alpha_Q d^{\text{off}}}\right), \quad \text{(A.20)}$$

with $Q^* = f_1^{*\prime}\left(\frac{d^{\text{off}}\cdot\zeta-(1-\gamma)\mu\pi-\gamma\left(\mathcal{P}_*^\pi\cdot d^{\text{off}}\right)\zeta}{\alpha_Q d^{\text{off}}}\right)$.

So, the regularized Lagrangian can be represented as

$$\max_d \alpha_R\mathbb{E}_{d^{\text{off}}}[\zeta\cdot R] \\ -\alpha_Q\mathbb{E}_{d^{\text{off}}}\left[f_1^*\left(\frac{d^{\text{off}}\cdot\zeta-(1-\gamma)\mu\pi-\gamma\left(\mathcal{P}_*^\pi\cdot d^{\text{off}}\right)\zeta}{\alpha_Q d^{\text{off}}}\right)\right] - \alpha_\zeta\mathbb{E}_{d^{\text{off}}}[f_2(\zeta)]. \quad \text{(A.21)}$$

Similarly, to approximate the intractable second term, we must use

$$\max_d \alpha_R\mathbb{E}_{d^{\text{off}}}[\zeta\cdot R] \\ -\alpha_Q\mathbb{E}_{\substack{(s,a,r,s')\sim d^{\text{off}}\\ a'\sim\pi(s')}}\left[f_1^*\left(\frac{\zeta(s',a')-(1-\gamma)\mu(s')\pi(a'|s')-\gamma\zeta(s,a)}{\alpha_Q d^{\text{off}}}\right)\right] - \alpha_\zeta\mathbb{E}_{d^{\text{off}}}[f_2(\zeta)],$$

which will introduce bias.

A.1.5 Undiscounted MDP

When $\gamma = 1$, the value of a policy is defined as the average per-step reward:

$$\rho(\pi) := \lim_{t_{\text{stop}} \to \infty} \mathbb{E}\left[\frac{1}{t_{\text{stop}}} \sum_{t=0}^{t_{\text{stop}}} R(s_t, a_t) \,\middle|\, s_0 \sim \mu, \forall t, a_t \sim \pi(s_t), s_{t+1} \sim T(s_t, a_t) \right]. \tag{A.22}$$

The following theorem presents a formulation of $\rho(\pi)$ in the undiscounted case:

Theorem 15. *Given a policy π and a discounting factor $\gamma = 1$, the value $\rho(\pi)$ defined in (A.22) can be expressed by the following d-LP:*

$$\max_{d: S \times A \to \mathbb{R}} \mathbb{E}_d [R(s,a)], \quad \text{s.t.,} \quad d(s,a) = \mathcal{P}_*^\pi d(s,a) \text{ and } \sum_{s,a} d(s,a) = 1. \tag{A.23}$$

The corresponding primal LP under the undiscounted case is

$$\min_{Q: S \times A \to \mathbb{R}} \lambda, \quad \text{s.t.,} \quad Q(s,a) = R(s,a) + \mathcal{P}^\pi Q(s,a) - \lambda. \tag{A.24}$$

Proof. With the additional constraint $\sum_{s,a} d(s,a) = 1$ in (A.23), the Markov chain induced by π is ergodic with a unique stationary distribution $d^* = d^\pi$, so the dual objective is still $\rho(\pi)$ by definition. Unlike in the discounted case, any optimal Q^* with a constant offset would satisfy (A.24), so the optimal solution to (A.24) is independent of Q. □

A.1.6 Experiment Details

OPE tasks For all tasks, We use $\gamma = 0.99$ in all experiments except for the ablation study of normalization constraint where $\gamma = 0.995$ and $\gamma = 1$ are also evaluated. We collect 400 trajectories for each of the tasks, and the trajectory length for Grid, Reacher, and Cartpole are 100, 200, and 250 respectively for $\gamma < 1$, or 1000 for $\gamma = 1$.

Grid. We use a 10×10 grid environment where an agent can move left/right/up/down. The observations are the x, y coordinates of this agent's location. The reward of each step is defined as $\exp(-0.2|x - 9| - 0.2|y - 9|)$. The target policy is taken to be the optimal policy for this task (i.e., moving all the way right then all the way down) plus 0.1 weight on uniform exploration. The behavior policies π_1 and π_2 are taken to be the optimal policy plus 0.7 and 0.3 weights on uniform exploration respectively.

Reacher. We train a deterministic policy on the Reacher task from OpenAI Gym [327] until convergence, and define the target policy to be a Gaussian with the pre-trained policy as the mean and 0.1 as the standard deviation. The behavior policies π_1 and π_2 have the same mean as the target policy but with 0.4 and 0.2 standard deviation respectively.

Cartpole. We modify the Cartpole task from OpenAI Gym [327] to infinite horizon by changing the reward to -1 if the original task returns termination and 1 otherwise. We train a deterministic policy on this task until convergence, and define the target policy to be the pre-trained policy (weight 0.7) plus uniform random exploration (weight 0.3). The behavior policies π_1 and π_2 are taken to be the pre-trained policy (weight 0.55 and 0.65) plus uniform random exploration (weight 0.45 and 0.35) respectively.

Linear Parametrization Details To test estimation robustness to scaling and shifting of MDP rewards under linear parametrization, we first determine the estimation upper bound by parametrizing the primal variable as a linear function of the one-hot encoding of the state-action input. Similarly, to determine the lower bound, we parametrize the dual variable as a linear function of the input. These linear parametrizations are implemented using feed-forward networks with two hidden-layers of 64 neurons each and without non-linear activations. Only the output layer is trained using gradient descent; the rest layers are randomly initialized and fixed. The true estimates where both primal and dual variables are linear functions are verified to be between the lower and upper bounds.

Neural Network Details For the neural network parametrization, we use feed-forward networks with two hidden-layers of 64 neurons each and ReLU as the activation function. The networks are trained using the Adam optimizer ($\beta_1 = 0.99$, $\beta_2 = 0.999$) with batch size 2048. The learning rate of each task and configuration is found via hyperparameter search, and is determined to be 0.00003 for all configurations on Grid, 0.0001 for all configurations on Reacher, and 0.0001 and 0.00003 for dual and primal regularization on Cartpole respectively.

A.1.7 Additional Results

Comparison to unregularized Lagrangian We compare the best performing DICE estimator discovered in our unified framework to directly solving the Lagrangian without any regularization or redundant constraints, *i.e.*, DR-MWQL as primal, MWL as dual, and their combination [219]. Results are shown in Figure A.1. We see that the BestDICE estimator outperforms the original primal, dual and Lagrangian both in terms of training stability and final estimation. This demonstrates that regularization and redundant constraints are crucial for optimization, justifying our motivation.

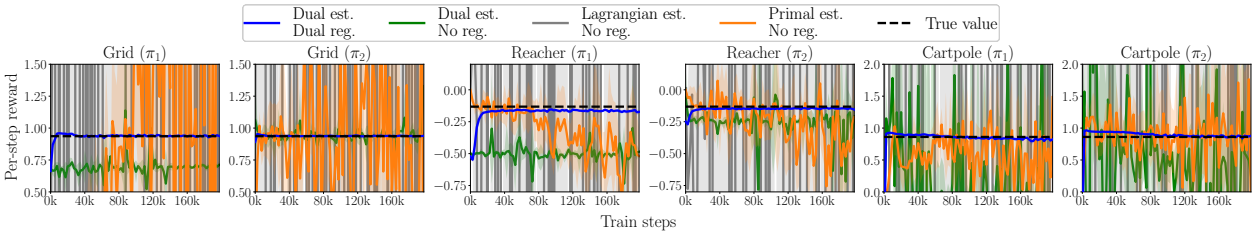


Figure A.1: Primal (orange), dual (green), and Lagrangian (gray) estimates by solving the original Lagrangian without any regularization or redundant constraints, in comparison with the best DICE estimates (blue).

Primal Estimates with Target Networks We use target networks with double Q-learning [491] to improve the training stability of primal variables, and notice performance

improvements in primal estimates on the Reacher task in particular. However, the primal estimates are still sensitive to scaling and shifting of MDP rewards, as shown in Figure A.2.

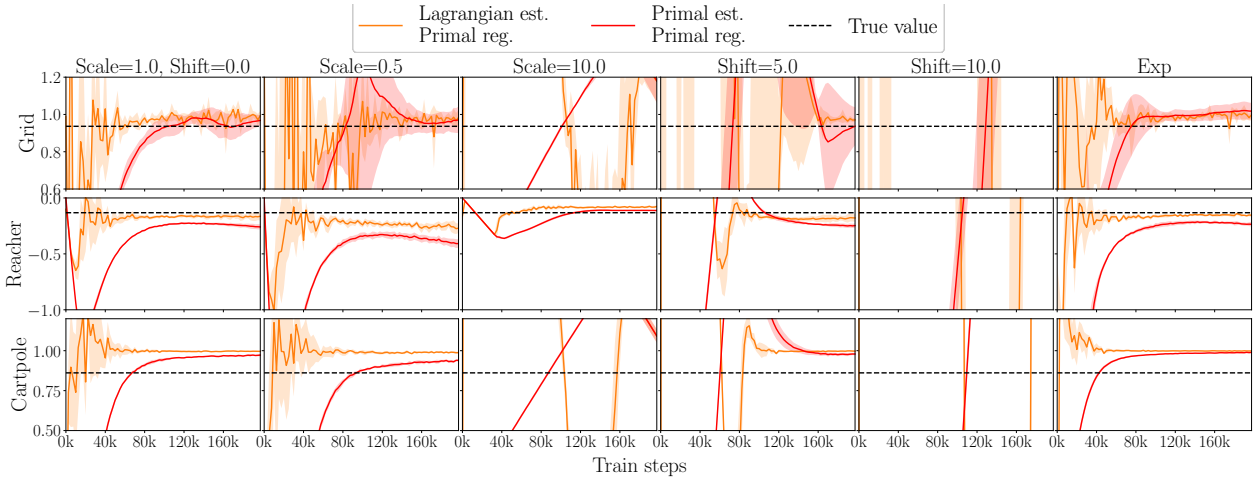


Figure A.2: Primal (red) and Lagrangian (orange) estimates under the neural network parametrization with target networks to stabilize training when rewards are transformed during training. Estimations are transformed back and plotted on the original scale. Despite the performance improvements on Reacher compared to Figure 3.2, the primal and Lagrangian estimates are still sensitive to the reward values.

Additional Regularization Comparison In addition to the two behavior policies in the main text (i.e., π_1 and π_2), we show the effect of regularization using data collected from a third behavior policy (π_3). Similar conclusions from the main text still hold (i.e., dual regularizer is generally better; primal regularizer with reward results in biased estimates) as shown in Figure A.3.

Additional Ablation Study We also conduct additional ablation study on data collected from a third behavior policy (π_3). Results are shown in Figure A.4. Again we see that the positivity constraint improves training stability as well as final estimates, and unconstrained primal form is more stable but can lead to biased estimates.

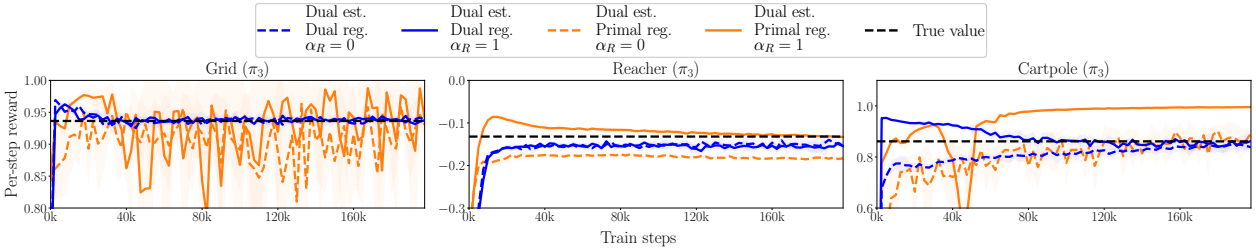


Figure A.3: Dual estimates when $\alpha_R = 0$ (dotted line) and $\alpha_R = 1$ (solid line) on data collected from a third behavior policy (π_3). Regularizing the dual variable (blue) is better than or similar to regularizing the primal variable (orange).

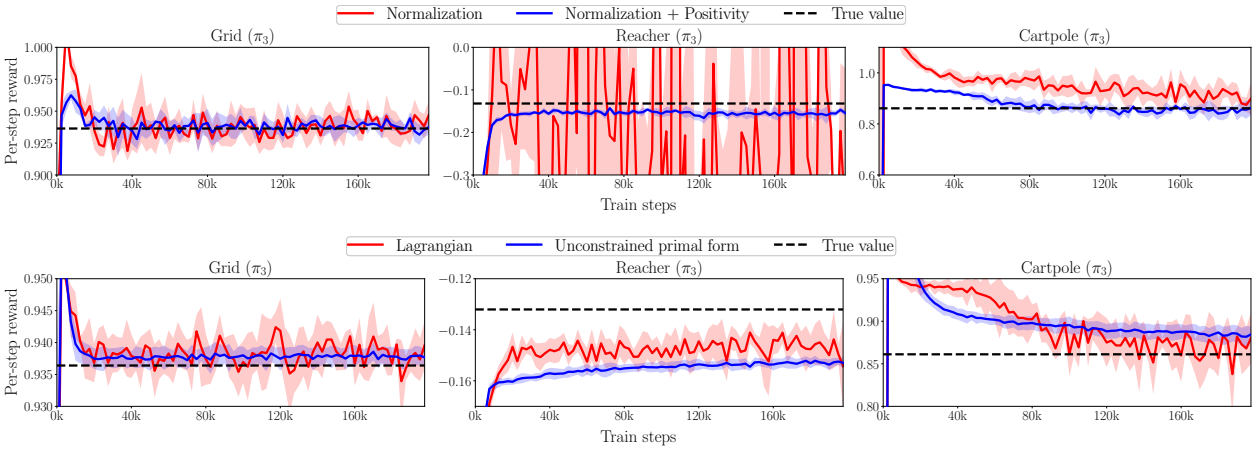


Figure A.4: Apply positive constraint and unconstrained primal form on data collected from a third behavior policy (π_3). Positivity constraint (row 1) improves training stability. The unconstrained primal problem (row 2) is more stable but leads to biased estimates.

A.2 Appendix for Representation Learning for Decision Making

A.2.1 Experimental Details for Representation Learning Ablation

Representation Network We parametrize the representation function ϕ as a two-hidden layer fully connected neural network with 256 units per layer and output dimension 256. A Swish [492] activation function is applied to the output of each hidden layer. We experimented with representation dimension sizes $16, 64, 256$, and 512, and found 256 and 512 to generally work the best (see Figure A.7 in Appendix A.2.2).

Transformer Network The BERT-style transformer used in attentive contrastive learning (ACL) consists of one preprocessing layer of 256 units and ReLU activaiton, followed by a multi-headed attention layer (4 heads with 128 units each), followed by a fully connected feed forward layer with hidden dimension 256 and ReLU activation, finally followed by an output layer of 256 units (the same as ϕ's output). We experimented with different number of attention blocks and number of heads in each block, but did not observe significant difference in performance.

When masking input (sequences of state, actions, or rewards), we randomly choose to 'drop' each item with probability 0.3, 'switch' with probability 0.15, and 'keep' with probability 0.15. 'Drop' refers to replacing the item with a trainable variable of the same dimension. 'Switch' refers to replacing the item with a randomly sampled item from the same batch. 'Keep' refers to leaving the item unchanged. These probability rates where chosen arbitrarily and not tuned.

Action Prediction and Reconstruction Whenever a loss includes action prediction or reconstruction, we follow [265], and (1) utilize an output distribution given by a tanh-squashed Gaussian and (2) apply an additive adaptive entropy regularizer to the action prediction loss.

Other Networks With few exceptions, all other functions f, g mentioned in Section 4.1.4 are two-hidden layer fully connected neural networks with 256 units per layer and using a Swish [492] activation.

The only exception is Momentum TCL, where f is the same structure but using a residual connection on the output.

Training During pretraining, we use the Adam optimizer with learning rate 0.0001, except for the TCL variants, for which we found 0.0003 to work better. For Momentum TCL, we use a moving average with rate 0.05.

Convergence Failures Representations learned under objectives including forward-raw model, VPN (with $k+1 = 2$), and DeepMDP consistently diverge and output NaNs on offline and online RL, and are therefore removed from the results in Figure 4.2. The bisimulation objective on offline and online RL fails to converge in some runs but occasionally succeeds, therefore the means of succeeded runs are computed and shown in Figure 4.2.

A.2.2 Additional Experimental Results for Representation Ablation

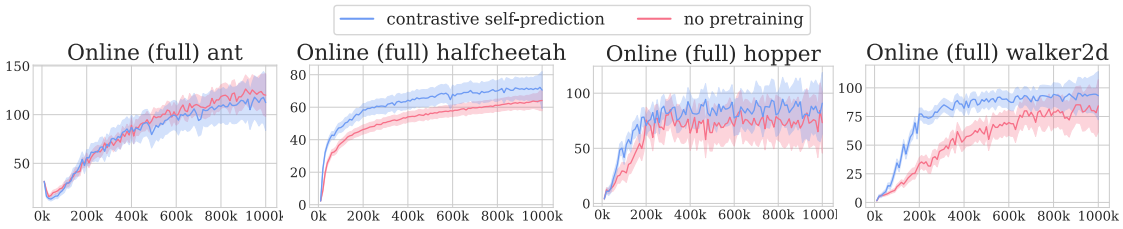


Figure A.5: Average reward of best ACL ablation on fully-observable online RL compared to the baseline without pretraining.

Fully-Observable Online Environments

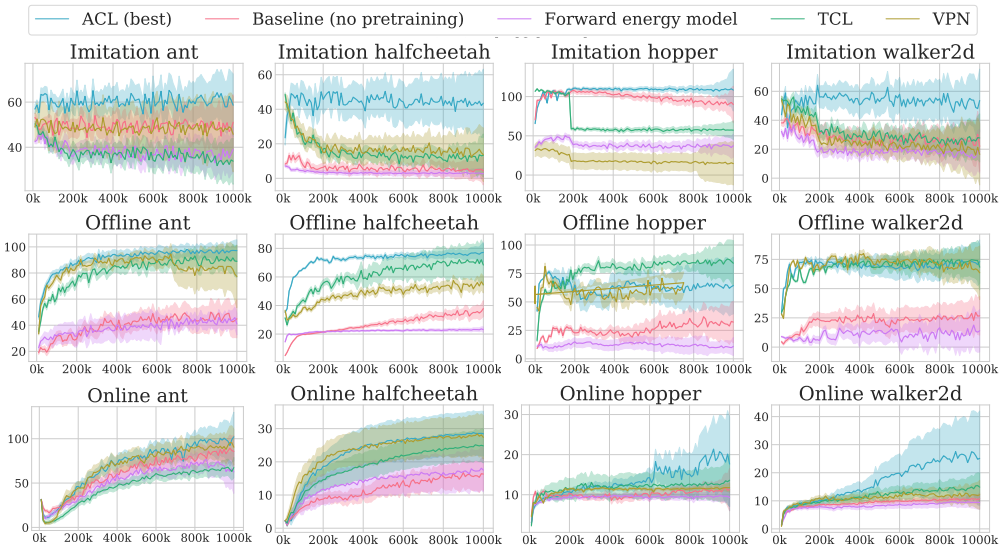


Figure A.6: Additional training curves of contrastive learning objectives aggregated over different offline datasets in the same domain. Both in this figure and in Figure 4.1, we plot the best variant of ACL according to the ablation study, namely we set "input reward" to false in imitation learning, "reconstruct action" to true in offline RL, and "auxiliary loss" (in ant and halfcheetah) or "finetuning" (in hopper and walker2d) to true in online RL. The best variant of ACL generally performs the best compared to other contrastive learning objectives, although TCL's performance is competitive in offline RL.

Additional Contrastive Learning Results

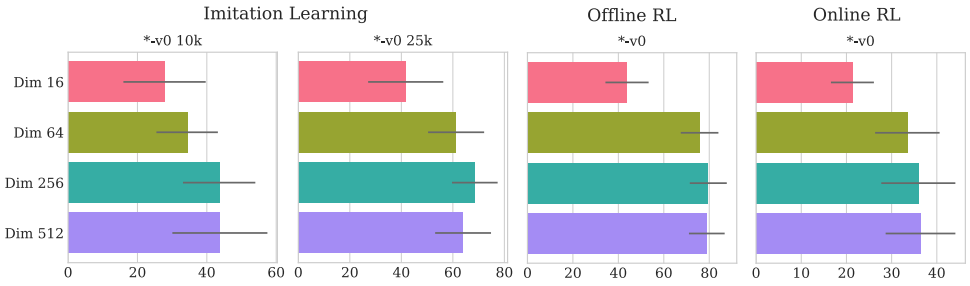


Figure A.7: Average reward across domains and datasets with different representation dimensions. 256 and 512 work the best (this ablation is conducted with "reconstruct action" and "reconstruct reward" set to true).

Ablation over Representation Size

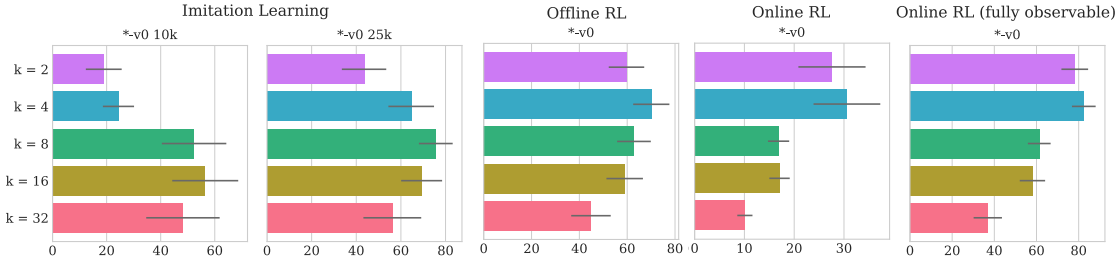


Figure A.8: Average reward across domains and datasets with different pretraining window k in imitation learning, offline RL, and partially/fully observable online RL.

Ablation over Pretraining Window Size

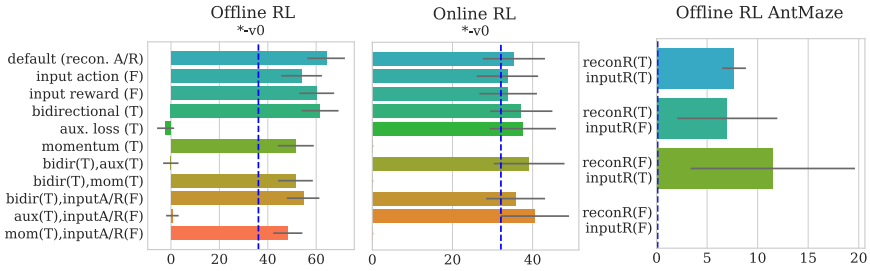


Figure A.9: Left: Ablation (with compounding factors) with reconstructing action/reward as default. Right: reward ablation on antmaze-umaze with sparse reward.

Ablation over Compounding Factors and on Sparse Reward

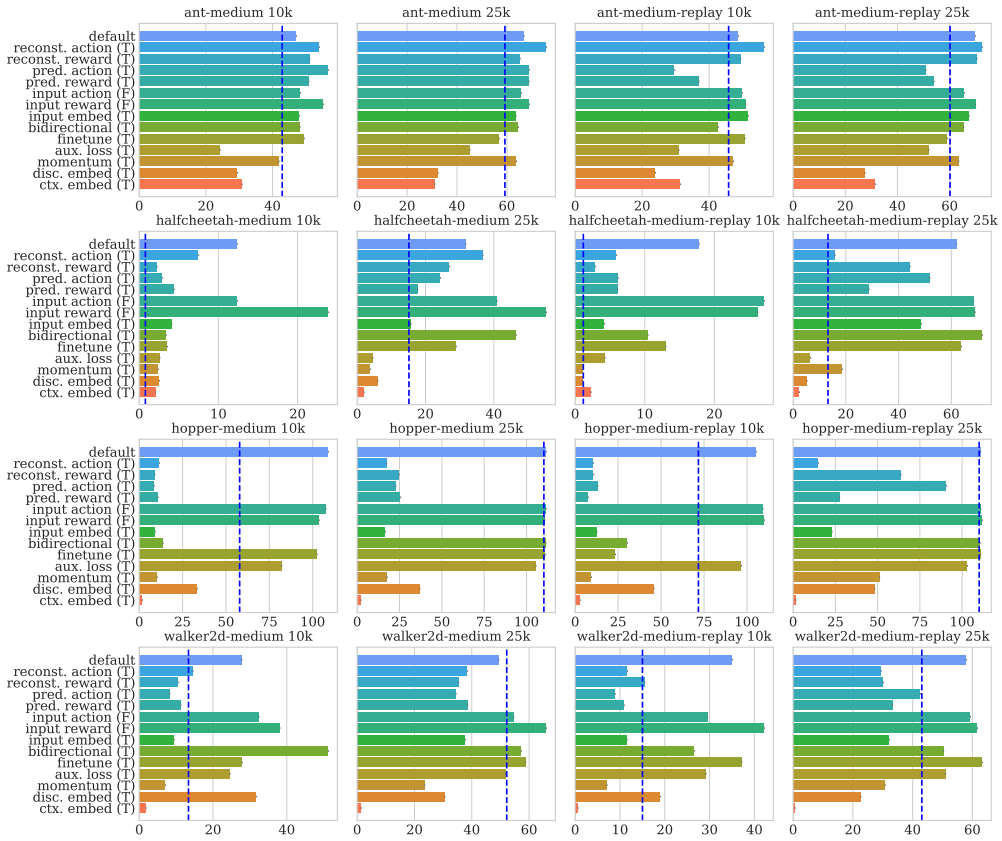


Figure A.10: Imitation learning ablation on individual domains and datasets. The negative impact of inputting action and reward to pretraining is more evident in halfcheetah and walker2d. Reconstructing/predicting action/reward is especially harmful in halfcheetah, hopper, and walker2d. There always exists some variant of ACL that is better than without representation learning (blue lines) in all domain-dataset combinations.

Ablation Results for Individual Domains and Datasets

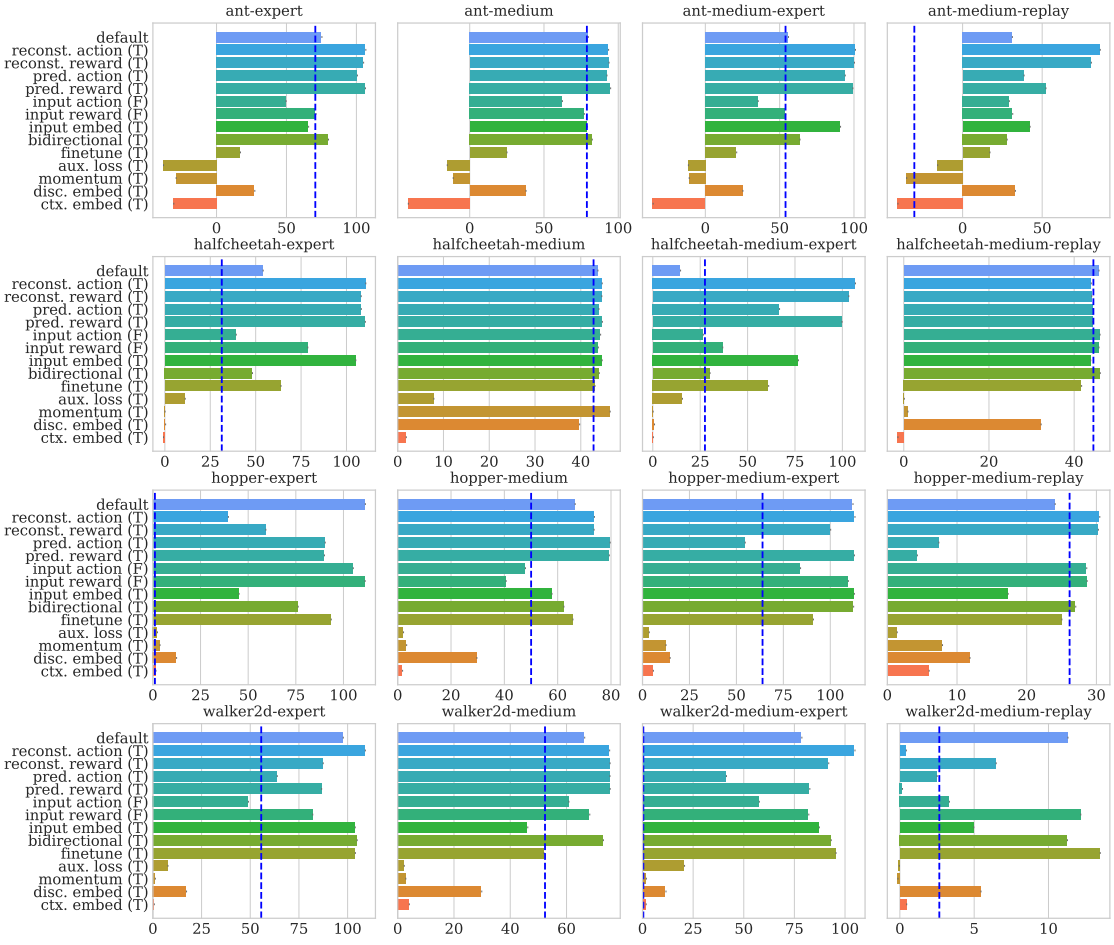


Figure A.11: Offline RL ablation on individual domains and datasets. The benefit of representation learning is more evident when expert trajectories are present (e.g., expert and medium-expert) than when they are absent (medium and medum-replay). Reconstructing action and reward is more important in ant and halfcheetah than in hopper and walker2d.

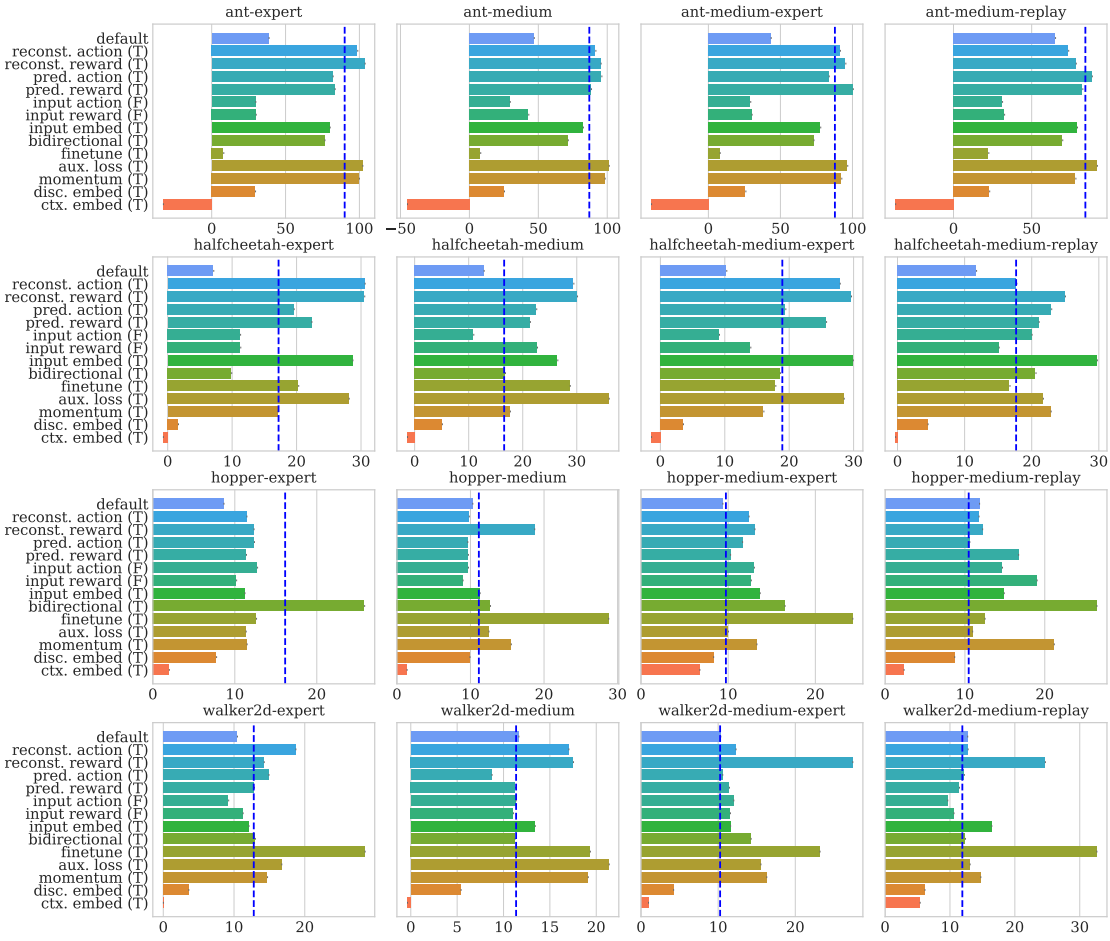


Figure A.12: Online RL ablation on individual domains and datasets. Auxiliary loss generally improves performance in all domains and datasets. Finetuning improves halfcheetah, hopper, and walker2d but significantly impairs ant.

A.2.3 Additional Anecdotal Conclusions from Representation Ablation

1. **More ablations.** Although we present our ablations as only changing one factor at a time, we also experimented with changing multiple factors at a time. We did not find any of these additional ablations to change the overall conclusions.

2. **Reconstruct action.** One ablation that did work surprisingly well was to only reconstruct the action (with no other loss). This appeared to perform poorly on imitation learning, but well on other settings.

3. **More transformers.** We experimented with a different application of transformers than ACL. Namely, we attempted to treat each dimension of the state as a token in a sequence (as opposed to using the whole state observation as the token). We found this to provide promising results, although it did not convincingly improve upon the configuration of ACL. Still, it may merit further investigation by future work.

4. **Transformer architecture.** We experimented with a different number of attention blocks or number of heads in each block, but did not observe significant differences in performance.

5. **Normalized or regularized representations.** We experimented with applying an explicit normalization layer on the output of ϕ and found no benefits. We also experimented with a stochastic representation along with a KL-divergence regularizer to the standard normal distribution, and again found no benefits.

A.2.4 Additional Interpretations of Results for Representation Ablation

While we wanted to avoid making claims without exhaustive proof regarding why certain design choices are better than others, we believe many of our findings are interpretable, and here is a selection of our hypotheses:

1. Reward prediction helps in offline and online RL because rewards are critical to the downstream task.

2. Bisimulation-style approaches perform poorly because they are too eager to reduce the latent space (e.g., in the absence of reward, ϕ would be constant).

3. The poor performance of auxiliary training in offline RL may reflect the fact that offline RL is generally more liable to divergence in training, which would dominate gradients from any auxiliary objective and render the representation learning objective useless.

4. The poor performance of context embeddings in all setups may be explained as a consequence of an overly-rich representation – i.e., using context embedding means that in the downstream task the same state may appear multiple times as different representations (since it is in a different context), and this can complicate learning in near-Markovian environments, unlike in NLP.

A.2.5 Proofs for Foundational Lemmas for TRAIL

Lemma 16. *If π_1 and π_2 are two policies in $\mathcal{M}$ and $d^{\pi_1}(s)$ and $d^{\pi_2}(s)$ are the state visitation distributions induced by policy π_1 and π_2 where $d^{\pi}(s) := (1-\gamma)\sum_{t=0}\gamma^t \cdot \Pr[s_t = s|\pi, \mathcal{M}]$. Define $\text{Diff}(\pi_2, \pi_1) = D_{\text{TV}}(d^{\pi_2}\|d^{\pi_1})$ then*

$$\text{Diff}(\pi_2, \pi_1) \leq \frac{\gamma}{1-\gamma}\text{Err}_{d^{\pi_1}}(\pi_1, \pi_2, \mathcal{T}), \tag{A.25}$$

where

$$\text{Err}_{d^{\pi_1}}(\pi_1, \pi_2, \mathcal{T}) := \frac{1}{2}\sum_{s' \in S}\left|\mathbb{E}_{s\sim d^{\pi_1}, a_1 \sim \pi_1(s), a_2 \sim \pi_2(s)}[\mathcal{T}(s'|s, a_1) - \mathcal{T}(s'|s, a_2)]\right|. \tag{A.26}$$

is the TV-divergence between $\mathcal{T} \circ \pi_1 \circ d^{\pi_1}$ and $\mathcal{T} \circ \pi_2 \circ d^{\pi_1}$.

Proof. Following similar derivations in [493, 256], we express $D_{\text{TV}}(d^{\pi_2}\|d^{\pi_1})$ in linear operator notation:

$$\text{Diff}(\pi_2, \pi_1) = D_{\text{TV}}(d^{\pi_2}\|d^{\pi_1}) = \frac{1}{2}\mathbf{1}|(1-\gamma)(I - \gamma\mathcal{T}\Pi_2)^{-1}\mu - (1-\gamma)(I - \gamma\mathcal{T}\Pi_1)^{-1}\mu|, \tag{A.27}$$

where Π_1, Π_2 are linear operators $S \to S \times A$ such that $\Pi_i\nu(s, a) = \pi_i(a|s)\nu(s)$ and $\mathbf{1}$ is an all ones row vector of size $|S|$. Notice that d^{π_1} may be expressed in this notation as $(1-\gamma)(I - \gamma\mathcal{T}\Pi_1)^{-1}\mu$. We may re-write the above term as

$$\begin{aligned}&\frac{1}{2}\mathbf{1}|(1-\gamma)(I - \gamma\mathcal{T}\Pi_2)^{-1}((I - \gamma\mathcal{T}\Pi_1) - (I - \gamma\mathcal{T}\Pi_2))(I - \gamma\mathcal{T}\Pi_1)^{-1}\mu| \\ =&\gamma \cdot \frac{1}{2}\mathbf{1}|(I - \gamma\mathcal{T}\Pi_2)^{-1}(\mathcal{T}\Pi_2 - \mathcal{T}\Pi_1)d^{\pi_1}|.\end{aligned} \tag{A.28}$$

Using matrix norm inequalities, we bound the above by

$$\gamma \cdot \frac{1}{2}\|(I - \gamma\mathcal{T}\Pi_2)^{-1}\|_{1,\infty} \cdot \mathbf{1}|(\mathcal{T}\Pi_2 - \mathcal{T}\Pi_1)d^{\pi_1}|. \tag{A.29}$$

Since $\mathcal{T}\Pi_2$ is a stochastic matrix, $\|(I-\gamma\mathcal{T}\Pi_2)^{-1}\|_{1,\infty} \leq \sum_{t=0}^{\infty}\gamma^t\|\mathcal{T}\Pi_2\|_{1,\infty} = (1-\gamma)^{-1}$. Thus, we bound the above by

$$\frac{\gamma}{2(1-\gamma)}\mathbf{1}|(\mathcal{T}\Pi_2 - \mathcal{T}\Pi_1)d^{\pi_1}| = \frac{\gamma}{1-\gamma}\text{Err}_{d^{\pi_1}}(\pi_1, \pi_2, \mathcal{T}), \tag{A.30}$$

and so we immediately achieve the desired bound in (A.25). □

The divergence bound above relies on the true transition model $\mathcal{T}$ which is not available to us. We now introduce an approximate transition model $\overline{\mathcal{T}}$ to proxy $\mathrm{Err}_{d^{\pi_1}}(\pi_1, \pi_2, \mathcal{T})$.

Lemma 17. *For π_1 and π_2 two policies in $\mathcal{M}$ and any transition model $\overline{\mathcal{T}}(\cdot|s,a)$ we have,*

$$\mathrm{Err}_{d^{\pi_1}}(\pi_1, \pi_2, \mathcal{T}) \leq |A| \mathbb{E}_{(s,a)\sim(d^{\pi_1}, \mathrm{Unif}_A)}[D_{\mathrm{TV}}(\mathcal{T}(s,a) \| \overline{\mathcal{T}}(s,a))] + \mathrm{Err}_{d^{\pi_1}}(\pi_1, \pi_2, \overline{\mathcal{T}}). \quad \text{(A.31)}$$

Proof.

$$\mathrm{Err}_{d^{\pi_1}}(\pi_1, \pi_2, \mathcal{T}) = \frac{1}{2} \sum_{s' \in S} \left| \mathbb{E}_{s\sim d^{\pi_1}, a_1 \sim \pi_1(s), a_2 \sim \pi_2(s)} [\mathcal{T}(s'|s, a_1) - \mathcal{T}(s'|s, a_2)] \right| \quad \text{(A.32)}$$

$$= \frac{1}{2} \sum_{s' \in S} \left| \sum_{a \in A} \mathbb{E}_{s\sim d^{\pi_1}} [\mathcal{T}(s'|s,a)\pi_1(a|s) - \mathcal{T}(s,a)\pi_2(a|s)] \right| \quad \text{(A.33)}$$

$$= \frac{1}{2} \sum_{s' \in S} \left| \sum_{a \in A} \mathbb{E}_{s\sim d^{\pi_1}} [(\mathcal{T}(s'|s,a) - \overline{\mathcal{T}}(s'|s,a))(\pi_1(a|s) - \pi_2(a|s)) + \overline{\mathcal{T}}(s'|s,a)(\pi_1(a|s) - \pi_2(a|s))] \right| \quad \text{(A.34)}$$

$$\leq \frac{1}{2} \sum_{s' \in S} \left| \sum_{a \in A} \mathbb{E}_{s\sim d^{\pi_1}} [(\mathcal{T}(s'|s,a) - \overline{\mathcal{T}}(s'|s,a))(\pi_1(a|s) - \pi_2(a|s))] \right| + \mathrm{Err}_{d^{\pi_1}}(\pi_1, \pi_2, \overline{\mathcal{T}}) \quad \text{(A.35)}$$

$$\leq \frac{1}{2} \sum_{s' \in S} \sum_{a \in A} \mathbb{E}_{s\sim d^{\pi_1}} [\left| (\mathcal{T}(s'|s,a) - \overline{\mathcal{T}}(s'|s,a))(\pi_1(a|s) - \pi_2(a|s)) \right|] + \mathrm{Err}_{d^{\pi_1}}(\pi_1, \pi_2, \overline{\mathcal{T}}) \quad \text{(A.36)}$$

$$\leq |A| \mathbb{E}_{(s,a)\sim(d^{\pi_1}, \mathrm{Unif}_A)} [D_{\mathrm{TV}}(\mathcal{T}(s'|s,a) \| \overline{\mathcal{T}}(s'|s,a)|] + \mathrm{Err}_{d^{\pi_1}}(\pi_1, \pi_2, \overline{\mathcal{T}}), \quad \text{(A.37)}$$

and we arrive at the inequality as desired where the last step comes from $D_{\mathrm{TV}}(\mathcal{T}(s,a) \| \overline{\mathcal{T}}(s,a)) = \frac{1}{2} \sum_{s' \in S} |\mathcal{T}(s'|s,a) - \overline{\mathcal{T}}(s'|s,a)|$. □

Now we introduce a representation function $\phi : S \times A \to Z$ and show how the error above may be reduced when $\overline{\mathcal{T}}(s,a) = \mathcal{T}_Z(s, \phi(s,a))$:

Lemma 18. *Let $\phi : S \times A \to Z$ for some space Z and suppose there exists $\mathcal{T}_Z : S \times Z \to \Delta(S)$ such that $\overline{\mathcal{T}}(s,a) = \mathcal{T}_Z(s, \phi(s,a))$ for all $s \in S, a \in A$. Then for any policies π_1, π_2,*

$$\mathrm{Err}_{d^{\pi_1}}(\pi_1, \pi_2, \overline{\mathcal{T}})] \leq \mathbb{E}_{s\sim d^{\pi_1}} [D_{\mathrm{TV}}(\pi_{1,Z} \| \pi_{2,Z})], \quad \text{(A.38)}$$

where $\pi_{k,Z}(z|s)$ is the marginalization of π_k onto Z:

$$\pi_{k,Z}(z|s) := \sum_{a \in A, z = \phi(s,a)} \pi_k(a|s) \quad \text{(A.39)}$$

for all $z \in Z, k \in \{1, 2\}$.

Proof.

$$\frac{1}{2}\sum_{s'\in S}\left|\mathbb{E}_{s\sim d^{\pi_1},a_1\sim\pi_1(s),a_2\sim\pi_2(s)}[\overline{\mathcal{T}}(s'|s,a_1)-\overline{\mathcal{T}}(s'|s,a_2)]\right| \tag{A.40}$$

$$=\frac{1}{2}\sum_{s'\in S}\left|\sum_{s\in S,a\in A}\mathcal{T}_Z(s'|s,\phi(s,a))\pi_1(a|s)d^{\pi_1}(s)-\sum_{s\in S,a\in A}\mathcal{T}_Z(s'|s,\phi(s,a))\pi_2(a|s)d^{\pi_1}(s)\right|$$

$$=\frac{1}{2}\sum_{s'\in S}\left|\sum_{s\in S,z\in Z}\mathcal{T}_Z(s'|s,z)\sum_{\substack{a\in A,\\ \phi(s,a)=z}}\pi_1(a|s)d^{\pi_1}(s)-\sum_{s\in S,z\in Z}\mathcal{T}_Z(s'|s,z)\sum_{\substack{a\in A,\\ \phi(s,a)=z}}\pi_2(a|s)d^{\pi_1}(s)\right|$$

$$=\frac{1}{2}\sum_{s'\in S}\left|\sum_{s\in S,z\in Z}\mathcal{T}_Z(s'|s,z)\pi_{1,Z}(z|s)d^{\pi_1}(s)-\sum_{s\in S,z\in Z}\mathcal{T}_Z(s'|s,z)\pi_{2,Z}(z|s)d^{\pi_1}(s)\right|$$

$$=\frac{1}{2}\sum_{s'\in S}\left|\mathbb{E}_{s\sim d^{\pi_1}}\left[\sum_{z\in Z}\mathcal{T}_Z(s'|s,z)(\pi_{1,Z}(z|s)-\pi_{2,Z}(z|s))\right]\right| \tag{A.41}$$

$$\leq\frac{1}{2}\mathbb{E}_{s\sim d^{\pi_1}}\left[\sum_{z\in Z}\sum_{s'\in S}\mathcal{T}_Z(s'|s,z)\left|\pi_{1,Z}(z|s)-\pi_{2,Z}(z|s)\right|\right] \tag{A.42}$$

$$=\frac{1}{2}\mathbb{E}_{s\sim d^{\pi_1}}\left[\sum_{z\in Z}\left|\pi_{1,Z}(z|s)-\pi_{2,Z}(z|s)\right|\right] \tag{A.43}$$

$$=\mathbb{E}_{s\sim d^{\pi_1}}\left[D_{\mathrm{TV}}(\pi_{1,Z}\|\pi_{2,Z})\right], \tag{A.44}$$

and we arrive at the inequality as desired. □

Lemma 19. *Let $d\in\Delta(S,A)$ be some state-action distribution, $\phi:S\times A\to Z$, and $\pi_Z:S\to\Delta(Z)$. Denote π_{α^*} as the optimal action decoder for d,ϕ:*

$$\pi_{\alpha^*}(a|s,z)=\frac{d(s,a)\cdot\mathbb{1}[z=\phi(s,a)]}{\sum_{a'\in A}d(s,a')\cdot\mathbb{1}[z=\phi(s,a')]},$$

and $\pi_{\alpha^*,Z}$ as the marginalization of $\pi_{\alpha^*}\circ\pi_Z$ onto Z:

$$\pi_{\alpha^*,Z}(z|s):=\sum_{a\in A,z=\phi(s,a)}(\pi_{\alpha^*}\circ\pi_Z)(a|s)=\sum_{a\in A,z=\phi(s,a)}\sum_{\tilde{z}\in Z}\pi_{\alpha^*}(a|s,\tilde{z})\pi_Z(\tilde{z}|s).$$

Then we have

$$\pi_{\alpha^*,Z}(z|s)=\pi_Z(z|s) \tag{A.45}$$

for all $z\in Z$ and $s\in S$.

Proof.

$$\pi_{\alpha^*,Z}(z|s) = \sum_{a\in A, z=\phi(s,a)} \sum_{\tilde{z}\in Z} \pi_{\alpha^*}(a|s,\tilde{z})\pi_Z(\tilde{z}|s) \tag{A.46}$$

$$= \sum_{a\in A, z=\phi(s,a)} \sum_{\tilde{z}\in Z} \frac{d(s,a)\cdot \mathbb{1}[\tilde{z}=\phi(s,a)]}{\sum_{a'\in A} d(s,a')\cdot \mathbb{1}[\tilde{z}=\phi(s,a')]}\pi_Z(\tilde{z}|s) \tag{A.47}$$

$$= \sum_{a\in A, z=\phi(s,a)} \frac{d(s,a)\cdot \mathbb{1}[z=\phi(s,a)]}{\sum_{a'\in A} d(s,a')\cdot \mathbb{1}[z=\phi(s,a')]}\pi_Z(z|s) \tag{A.48}$$

$$= \pi_Z(z|s) \sum_{a\in A, z=\phi(s,a)} \frac{d(s,a)\cdot \mathbb{1}[z=\phi(s,a)]}{\sum_{a'\in A} d(s,a')\cdot \mathbb{1}[z=\phi(s,a')]} \tag{A.49}$$

$$= \pi_Z(z|s), \tag{A.50}$$

and we have the desired equality. □

Lemma 20. *Let $\pi_Z : S \to \Delta(Z)$ be a latent policy in Z and $q : S \times Z \to A$ be an action decoder, $\pi_{\alpha,Z}$ be the marginalization of $q \circ \pi_Z$ onto Z:*

$$\pi_{\alpha,Z}(z|s) := \sum_{a\in A, z=\phi(s,a)} (q\circ\pi_Z)(a|s) = \sum_{a\in A, z=\phi(s,a)} \sum_{\tilde{z}\in Z} \pi_\alpha(a|s,\tilde{z})\pi_Z(\tilde{z}|s).$$

Then for any $s \in S$ we have

$$D_{\mathrm{TV}}(\pi_Z(s)\|\pi_{\alpha,Z}(s)) \le \max_{z\in Z} D_{\mathrm{TV}}(\pi_{\alpha^*}(s,z)\|q(s,z)), \tag{A.51}$$

where π_{α^*} is the optimal action decoder defined in Lemma 19 (and this holds for any choice of d from Lemma 19).

Proof.

$$
\begin{aligned}
& D_{\mathrm{TV}}(\pi_Z(s) \| \pi_{\alpha,Z}(s)) && \text{(A.52)}\\
=& \frac{1}{2} \sum_{z \in Z} |\pi_Z(z|s) - \pi_{\alpha,Z}(z|s)| && \text{(A.53)}\\
=& \frac{1}{2} \sum_{z \in Z} \left| \pi_Z(z|s) - \sum_{a \in A, z=\phi(s,a)} \sum_{\tilde{z} \in Z} \pi_\alpha(a|s,\tilde{z}) \pi_Z(\tilde{z}|s) \right| && \text{(A.54)}\\
=& \frac{1}{2} \sum_{z \in Z} \left| \pi_Z(z|s) - \sum_{a \in A, z=\phi(s,a)} \sum_{\tilde{z} \in Z} \left(\pi_\alpha(a|s,\tilde{z}) - \pi_{\alpha^*}(a|s,\tilde{z}) + \pi_{\alpha^*}(a|s,\tilde{z})\right) \pi_Z(\tilde{z}|s) \right| && \text{(A.55)}\\
=& \frac{1}{2} \sum_{z \in Z} \left| \sum_{a \in A, z=\phi(s,a)} \sum_{\tilde{z} \in Z} \left(\pi_\alpha(a|s,\tilde{z}) - \pi_{\alpha^*}(a|s,\tilde{z})\right) \pi_Z(\tilde{z}|s) \right| \quad \text{(by Lemma 19)} && \text{(A.56)}\\
\leq& \frac{1}{2} \mathbb{E}_{\tilde{z} \sim \pi_Z(s)} \left[\sum_{z \in Z} \sum_{a \in A, z=\phi(s,a)} |\pi_\alpha(a|s,\tilde{z}) - \pi_{\alpha^*}(a|s,\tilde{z})| \right] && \text{(A.57)}\\
=& \frac{1}{2} \mathbb{E}_{\tilde{z} \sim \pi_Z(s)} \left[\sum_{a \in A} |\pi_\alpha(a|s,\tilde{z}) - \pi_{\alpha^*}(a|s,\tilde{z})| \right] && \text{(A.58)}\\
=& \mathbb{E}_{\tilde{z} \sim \pi_Z(s)} \left[D_{\mathrm{TV}}(q(s,\tilde{z}) \|, \pi_{\alpha^*}(s,\tilde{z})) \right] && \text{(A.59)}\\
\leq& \max_{z \in Z} D_{\mathrm{TV}}(q(s,z) \| \pi_{\alpha^*}(s,z)), && \text{(A.60)}\\
& && \text{(A.61)}
\end{aligned}
$$

and we have the desired inequality. □

Lemma 21. *Let $\pi_{1,Z}$ be the marginalization of π_1 onto Z as defined in Lemma 18, and let π_Z, q, $\pi_{\alpha,Z}$ be as defined in Lemma 20, and let $\pi_{\alpha^*,Z}$ be as defined in Lemma 19. For any $s \in S$ we have*

$$
D_{\mathrm{TV}}(\pi_{1,Z}(s) \| \pi_{\alpha,Z}(s)) \leq \max_{z \in Z} D_{\mathrm{TV}}(q(s,z) \| \pi_{\alpha^*}(s,z)) + D_{\mathrm{TV}}(\pi_{1,Z}(s) \| \pi_Z(s)). \tag{A.62}
$$

Proof. The desired inequality is achieved by plugging the inequality from Lemma 20 into the following triangle inequality:

$$
D_{\mathrm{TV}}(\pi_{1,Z}(s) \| \pi_{\alpha,Z}(s)) \leq D_{\mathrm{TV}}(\pi_Z(s) \| \pi_{\alpha,Z}(s)) + D_{\mathrm{TV}}(\pi_{1,Z}(s) \| \pi_Z(s)). \tag{A.63}
$$

□

Our final lemma will be used to translate on-policy bounds to off-policy.

Lemma 22. *For two distributions $\rho_1, \rho_2 \in \Delta(S)$ with $\rho_1(s) > 0 \Rightarrow \rho_2(s) > 0$, we have,*

$$\mathbb{E}_{\rho_1}[h(s)] \leq (1 + D_{\chi^2}(\rho_1 \| \rho_2)^{\frac{1}{2}}) \sqrt{\mathbb{E}_{\rho_2}[h(s)^2]}. \tag{A.64}$$

Proof. The lemma is a straightforward consequence of Cauchy-Schwartz:

$$\begin{aligned}
\mathbb{E}_{\rho_1}[h(s)] &= \mathbb{E}_{\rho_2}[h(s)] + (\mathbb{E}_{\rho_1}[h(s)] - \mathbb{E}_{\rho_2}[h(s)]) && \text{(A.65)}\\
&= \mathbb{E}_{\rho_2}[h(s)] + \sum_{s \in S} \frac{\rho_1(s) - \rho_2(s)}{\rho_2(s)^{\frac{1}{2}}} \cdot \rho_2(s)^{\frac{1}{2}} h(s) && \text{(A.66)}\\
&\leq \mathbb{E}_{\rho_2}[h(s)] + \left(\sum_{s \in S} \frac{(\rho_1(s) - \rho_2(s))^2}{\rho_2(s)} \right)^{\frac{1}{2}} \cdot \left(\sum_{s \in S} \rho_2(s) h(s)^2 \right)^{\frac{1}{2}} && \text{(A.67)}\\
&= \mathbb{E}_{\rho_2}[h(s)] + D_{\chi^2}(\rho_1 \| \rho_2)^{\frac{1}{2}} \cdot \sqrt{\mathbb{E}_{\rho_2}[h(s)^2]}. && \text{(A.68)}
\end{aligned}$$

Finally, to get the desired bound, we simply note that the concavity of the square-root function implies $\mathbb{E}_{\rho_2}[h(s)] \leq \mathbb{E}_{\rho_2}[\sqrt{h(s)^2}] \leq \sqrt{\mathbb{E}_{\rho_2}[h(s)^2]}$. □

A.2.6 Proofs for Major Theorems for TRAIL

Proof of Theorem 5

Proof. Let $\pi_2 := q \circ \pi_Z$, we have $\pi_{2,Z}(z|s) = \pi_{\alpha,Z}(z|s) = \sum_{a \in A, \phi(s,a)=z} (q \circ \pi_Z)(z|s)$. By plugging the result of Lemma 21 into Lemma 18, we have

$$\mathrm{Err}_{d^{\pi_1}}(\pi_1, \pi_2, \overline{\mathcal{T}})] \leq \mathbb{E}_{s \sim d^{\pi_1}} \left[\max_{z \in Z} D_{\mathrm{TV}}(\pi_{\alpha^*}(s,z) \| q(s,z)) + D_{\mathrm{TV}}(\pi_{1,Z}(s) \| \pi_Z(s)) \right]. \tag{A.69}$$

By plugging this result into Lemma 17, we have

$$\begin{aligned}
\mathrm{Err}_{d^{\pi_1}}(\pi_1, \pi_2, \mathcal{T}) \leq |A| \mathbb{E}_{(s,a) \sim (d^{\pi_1}, \mathrm{Unif}_A)}[D_{\mathrm{TV}}(\mathcal{T}(s,a) \| \overline{\mathcal{T}}(s,a))] && \text{(A.70)}\\
+ \mathbb{E}_{s \sim d^{\pi_1}} \left[\max_{z \in Z} D_{\mathrm{TV}}(\pi_{\alpha^*}(s,z) \| q(s,z)) \right] && \text{(A.71)}\\
+ \mathbb{E}_{s \sim d^{\pi_1}} \left[D_{\mathrm{TV}}(\pi_{1,Z}(s) \| \pi_Z(s)) \right]. && \text{(A.72)}
\end{aligned}$$

By further plugging this result into Lemma 16 and let $\pi_1 = \pi^*$, we have:

$$\begin{aligned}
\mathrm{Diff}(q \circ \pi_Z, \pi^*) \leq \frac{\gamma |A|}{1-\gamma} \cdot \mathbb{E}_{(s,a) \sim (d^{\pi_1}, \mathrm{Unif}_A)}[D_{\mathrm{TV}}(\mathcal{T}(s,a) \| \mathcal{T}_Z(s, \phi(s,a))]\\
+ \frac{\gamma}{1-\gamma} \cdot \mathbb{E}_{s \sim d^{\pi^*}}[\max_{z \in Z} D_{\mathrm{TV}}(\pi_{\alpha^*}(s,z) \| q(s,z))]\\
+ \frac{\gamma}{1-\gamma} \cdot \mathbb{E}_{s \sim d^{\pi^*}}[D_{\mathrm{TV}}(\pi_{*,Z}(s) \| \pi_Z(s))]. && \text{(A.73)}
\end{aligned}$$

Finally, by plugging in the off-policy results of Lemma 22 to the bound in Equation (A.73) and by applying Pinsker's inequality $D_{\mathrm{TV}}(\mathcal{T}(s,a)\|\mathcal{T}_Z(s,\phi(s,a)))^2 \leq \frac{1}{2}D_{\mathrm{KL}}(\mathcal{T}(s,a)\|\mathcal{T}_Z(s,\phi(s,a)))$, we have

$$\begin{aligned}\mathrm{Diff}(q\circ\pi_Z,\pi^*) \leq C_1\cdot\sqrt{\frac{1}{2}\underbrace{\mathbb{E}_{(s,a)\sim d^{\mathrm{off}}}\left[D_{\mathrm{KL}}(\mathcal{T}(s,a)\|\mathcal{T}_Z(s,\phi(s,a)))\right]}_{=\ J_{\mathrm{T}}(\mathcal{T}_Z,\phi)}} \\ + C_2\cdot\sqrt{\frac{1}{2}\underbrace{\mathbb{E}_{s\sim d^{\mathrm{off}}}[\max_{z\in Z} D_{\mathrm{KL}}(\pi_{\alpha^*}(s,z)\|q(s,z))]}_{\approx\ \mathrm{const}(d^{\mathrm{off}},\phi)+J_{\mathrm{DE}}(q,\phi)}} \\ + C_3\cdot\sqrt{\frac{1}{2}\underbrace{\mathbb{E}_{s\sim d^{\pi^*}}[D_{\mathrm{KL}}(\pi_{*,Z}(s)\|\pi_Z(s))]}_{=\ \mathrm{const}(\pi^*,\phi)+J_{\mathrm{BC},\phi}(\pi_Z)}},\end{aligned} \tag{A.74}$$

where $C_1 = \gamma|A|(1-\gamma)^{-1}(1+D_{\chi^2}(d^{\pi^*}\|d^{\mathrm{off}})^{\frac{1}{2}})$, $C_2 = \gamma(1-\gamma)^{-1}(1+D_{\chi^2}(d^{\pi^*}\|d^{\mathrm{off}})^{\frac{1}{2}})$, and $C_3 = \gamma(1-\gamma)^{-1}$. Since the $\max_{z\in Z}$ is not tractable in practice, we approximate $\mathbb{E}_{s\sim d^{\mathrm{off}}}[\max_{z\in Z} D_{\mathrm{KL}}(\pi_{\alpha^*}(s,z)\|q(s,z))]$ using $\mathbb{E}_{(s,a)\sim d^{\mathrm{off}}}[D_{\mathrm{KL}}(\pi_{\alpha^*}(s,\phi(s,a))\|q(s,\phi(s,a)))]$, which reduces to $J_{\mathrm{DE}}(q,\phi)$ with additional constants. We now arrive at the desired off-policy bound in Theorem 5. □

Proof of Theorem 6

Lemma 23. *Let $\rho\in\Delta(\{1,\ldots,k\})$ be a distribution with finite support. Let $\hat{\rho}_n$ denote the empirical estimate of ρ from n i.i.d. samples $X\sim\rho$. Then,*

$$\mathbb{E}_n[D_{\mathrm{TV}}(\rho\|\hat{\rho}_n)] \leq \frac{1}{2}\cdot\frac{1}{\sqrt{n}}\sum_{i=1}^{k}\sqrt{\rho(i)} \leq \frac{1}{2}\cdot\sqrt{\frac{k}{n}}. \tag{A.75}$$

Proof. The first inequality is Lemma 8 in [494] while the second inequality is due to the concavity of the square root function. □

Lemma 24. *Let $\mathcal{D} := \{(s_i,a_i)\}_{i=1}^n$ be i.i.d. samples from a factored distribution $x(s,a) := \rho(s)\pi(a|s)$ for $\rho\in\Delta(S), \pi: S\to\Delta(A)$. Let $\hat{\rho}$ be the empirical estimate of ρ in $\mathcal{D}$ and $\hat{\pi}$ be the empirical estimate of π in $\mathcal{D}$. Then,*

$$\mathbb{E}_{\mathcal{D}}[\mathbb{E}_{s\sim\rho}[D_{\mathrm{TV}}(\pi(s)\|\hat{\pi}(s))]] \leq \sqrt{\frac{|S||A|}{n}}. \tag{A.76}$$

Proof. Let $\hat{x}$ be the empirical estimate of x in $\mathcal{D}$. We have,

$$\mathbb{E}_{s\sim\rho}[D_{\text{TV}}(\pi(s)\|\hat{\pi}(s))] = \frac{1}{2}\sum_{s,a}\rho(s)\cdot|\pi(a|s)-\hat{\pi}(a|s)| \tag{A.77}$$

$$= \frac{1}{2}\sum_{s,a}\rho(s)\cdot\left|\frac{x(s,a)}{\rho(s)}-\frac{\hat{x}(s,a)}{\hat{\rho}(s)}\right| \tag{A.78}$$

$$\leq \frac{1}{2}\sum_{s,a}\rho(s)\cdot\left|\frac{\hat{x}(s,a)}{\rho(s)}-\frac{\hat{x}(s,a)}{\hat{\rho}(s)}\right| + \frac{1}{2}\sum_{s,a}\rho(s)\cdot\left|\frac{\hat{x}(s,a)}{\rho(s)}-\frac{x(s,a)}{\rho(s)}\right| \tag{A.79}$$

$$= \frac{1}{2}\sum_{s,a}\rho(s)\cdot\left|\frac{\hat{x}(s,a)}{\rho(s)}-\frac{\hat{x}(s,a)}{\hat{\rho}(s)}\right| + D_{\text{TV}}(x\|\hat{x}) \tag{A.80}$$

$$= \frac{1}{2}\sum_{s}\rho(s)\cdot\left|\frac{1}{\rho(s)}-\frac{1}{\hat{\rho}(s)}\right|\left(\sum_{a}\hat{x}(s,a)\right) + D_{\text{TV}}(x\|\hat{x}) \tag{A.81}$$

$$= \frac{1}{2}\sum_{s}\rho(s)\cdot\left|\frac{1}{\rho(s)}-\frac{1}{\hat{\rho}(s)}\right|\cdot\hat{\rho}(s) + D_{\text{TV}}(x\|\hat{x}) \tag{A.82}$$

$$= D_{\text{TV}}(\rho\|\hat{\rho}) + D_{\text{TV}}(x\|\hat{x}). \tag{A.83}$$

Finally, the bound in the lemma is achieved by application of Lemma 23 to each of the TV divergences. □

To prove Theorem 6, we first rewrite Theorem 5 as

$$\text{Diff}(\pi_Z,\pi^*) \leq (4.4)(\phi) + (4.5)(\phi) + C_3\cdot\mathbb{E}_{s\sim d^{\pi^*}}[D_{\text{TV}}(\pi_{*,Z}(s)\|\pi_Z(s))], \tag{A.84}$$

where (4.4) and (4.5) are the first two terms in the bound of Theorem 5, and $C_3 = \frac{\gamma}{1-\gamma}$.

The result in Theorem 6 is then derived by setting $\phi = \phi_{\pi_{orcl}}$ and $\pi_Z := \pi_{\phi_{orcl},Z}$ and using the result of Lemma 24.

Note that the above sample analysis can be extended to the continuous latent action space characterized by Theorem 7 as follows.

Theorem 25. *Let $\phi_{orcl} := \mathcal{OPT}_\phi(\mathcal{D}_{\text{off}})$ and $\pi_{orcl,\theta}$ be the latent BC policy with respect to ϕ_{orcl}. Let d be the dimension of the continuous latent actions and $\|\phi\|_\infty$ be the l_∞ norm of ϕ_{orcl} for any s,a. We have*

$$\mathbb{E}_{\mathcal{D}^*_{\text{off}}}[\text{Diff}(\pi_{\phi_{orcl},\theta},\pi^*)] \leq (4.4)(\phi_{orcl}) + (4.5)(\phi_{orcl}) + C_4\cdot d\|\phi\|_\infty\sqrt{\frac{2|S|}{n+1}},$$

where (4.4), (4.5), and C_4 are the same as in Theorem 7.

Proof. We use $\mu \in \mathbb{R}^{d\times|S|}$ to denote the optimal setting of θ which yields a zero l_1-norm of $\frac{\partial}{\partial\theta}\mathbb{E}_{s\sim d^\pi, a\sim\pi^*}[(\theta_s - \phi(s,a))^2]$; i.e.,

$$\mu_s = \mathbb{E}_{a\sim\pi^*(s)}[\phi(s,a)]. \tag{A.85}$$

According to Theorem 7, we want to bound the l_1-norm of $\frac{\partial}{\partial\theta}\mathbb{E}_{s\sim d^\pi, a\sim\pi^*}[(\theta_s - \phi(s,a))^2]$ evaluated at the approximate solution $\hat{\mu} \in \mathbb{R}^{d\times|S|}$ with respect to finite dataset $\mathcal{D}^*_{\text{off}}$; i.e.,

$$\hat{\mu}_s = \mathbb{E}_{a\sim\mathcal{D}^*_{\text{off}}(\cdot|s)}\left[\phi(s,a)\right], \tag{A.86}$$

with the convention that $\hat{\mu}_s = 0$ if s does not appear in $\mathcal{D}^*_{\text{off}}$. To this end, we have the following derivation, which uses $\mathbb{E}_n$ to denote the expectation over realizations of $\hat{\mu}$ due to n-size draws of the target dataset $\mathcal{D}^*_{\text{off}}$:

$$\begin{aligned}
\mathbb{E}_n\left[\left\|\frac{\partial}{\partial\theta}\Big|_{\theta=\hat{\mu}}\mathbb{E}_{s\sim d^\pi, a\sim\pi^*}\left[(\theta_s - \phi(s,a))^2\right]\right\|_1\right] &= \mathbb{E}_n\left[\mathbb{E}_{s\sim d^\pi}[\|\hat{\mu}_s - \mathbb{E}_{a\sim\pi^*}[\phi(s,a)]\|_1]\right] && \text{(A.87)}\\
&= \mathbb{E}_n\left[\mathbb{E}_{s\sim d^\pi}[\|\hat{\mu}_s - \mu_s\|_1]\right] && \text{(A.88)}\\
&= \mathbb{E}_{s\sim d^\pi}[\mathbb{E}_n\left[\|\hat{\mu}_s - \mu_s\|_1\right]]. && \text{(A.89)}
\end{aligned}$$

We now split up the inner expectation based on the number of times k that s appears in $\mathcal{D}^*_{\text{off}}$:

$$\begin{aligned}
\mathbb{E}_{s\sim d^\pi}[\mathbb{E}_n\left[\|\hat{\mu}_s - \mu_s\|_1\right]] &= \mathbb{E}_{s\sim d^\pi}\left[\sum_{k=0}^{n}\Pr[\text{count}(s)=k]\cdot\mathbb{E}_k\left[\|\hat{\mu}_s - \mu_s\|_1\right]\right] && \text{(A.90)}\\
&\leq \sqrt{\mathbb{E}_{s\sim d^\pi}\left[\sum_{k=0}^{n}\Pr[\text{count}(s)=k]\cdot\mathbb{E}_k\left[\|\hat{\mu}_s - \mu_s\|_1\right]^2\right]} && \text{(A.91)}\\
& && \text{(A.92)}
\end{aligned}$$

where $\mathbb{E}_k$ denotes the expectation over realizations of $\hat{\mu}_s$ over k-size draws of $a \sim \pi^*(s)$. By standard combinatorics, we know

$$\Pr[\text{count}(s)=k] = \binom{n}{k} d^\pi(s)^k (1-d^\pi(s))^{n-k}. \tag{A.93}$$

Furthermore, for $k=0$, we have

$$\mathbb{E}_k\left[\|\hat{\mu}_s - \mu_s\|_1\right]^2 = \|\mu_s\|_1^2 \leq d^2\|\phi\|_\infty^2, \tag{A.94}$$

while for $k>0$, since $\mathbb{E}_k[\hat{\mu}_s] = \mu_s$, we have

$$\mathbb{E}_k\left[\|\hat{\mu}_s - \mu_s\|_1\right]^2 \leq d\cdot\mathbb{E}_k\left[\|\hat{\mu}_s - \mu_s\|_2^2\right] = d\cdot\text{Var}_k\left[\hat{\mu}_s\right] \leq \frac{d^2\|\phi\|_\infty^2}{k} \leq \frac{2d^2\|\phi\|_\infty^2}{k+1}. \tag{A.95}$$

Combining equations A.93, A.94, and A.95 we have for any $k \geq 0$

$$d^\pi(s) \cdot \Pr[\text{count}(s) = k] \cdot \mathbb{E}_k \left[\|\hat{\mu}_s - \mu_s\|_1\right]^2 \leq \frac{2d^2\|\phi\|_\infty^2}{k+1}\binom{n}{k} d^\pi(s)^{k+1}(1-d^\pi(s))^{n-k}$$
$$= \frac{2d^2\|\phi\|_\infty^2}{n+1}\binom{n+1}{k+1} d^\pi(s)^{k+1}(1-d^\pi(s))^{n-k}, \tag{A.96}$$

and so by the binomial theorem,

$$\sum_{k=0}^{n} d^\pi(s) \cdot \Pr[\text{count}(s) = k] \cdot \mathbb{E}_k \left[\|\hat{\mu}_s - \mu_s\|_1\right]^2 \leq \frac{2d^2\|\phi\|_\infty^2}{n+1}. \tag{A.97}$$

Plugging the above into equation A.91 we deduce

$$\mathbb{E}_{s\sim d^\pi}\left[\mathbb{E}_n \left[\|\hat{\mu}_s - \mu_s\|_1\right]\right] \leq d\|\phi\|_\infty \sqrt{\frac{2|S|}{n+1}}, \tag{A.98}$$

and we have the convergence rate as desired. □

Proof of Theorem 7

Proof. The gradient term in Theorem 7 with respect to a specific column θ_s of θ may be expressed as

$$\frac{\partial}{\partial \theta_s}\mathbb{E}_{\tilde{s}\sim d^\pi, a\sim\pi(\tilde{s})}[(\theta_{\tilde{s}} - \phi(\tilde{s}, a))^2]$$
$$= -2\mathbb{E}_{a\sim\pi(s)}[d^\pi(s)\phi(s,a)] + 2d^\pi(s)\theta_s$$
$$= -2\mathbb{E}_{a\sim\pi(s)}[d^\pi(s)\phi(s,a)] + 2\mathbb{E}_{z=\theta_s}[d^\pi(s)\cdot z], \tag{A.99}$$

and so,

$$w(s')^\top \frac{\partial}{\partial \theta_s}\mathbb{E}_{\tilde{s}\sim d^\pi, a\sim\pi(\tilde{s})}[(\theta_{\tilde{s}} - \phi(\tilde{s}, a))^2]$$
$$= -2\mathbb{E}_{a\sim\pi(s)}[d^\pi(s)\overline{\mathcal{T}}(s'|s,a)] + 2\mathbb{E}_{z=\theta_s}[d^\pi(s)w(s')^\top z]. \tag{A.100}$$

Summing over $s \in S$, we have:

$$\sum_{s\in S} w(s')^\top \frac{\partial}{\partial \theta_s}\mathbb{E}_{\tilde{s}\sim d^\pi, a\sim\pi(\tilde{s})}[(\theta_{\tilde{s}} - \phi(\tilde{s}, a))^2]$$
$$= 2\mathbb{E}_{s\sim d^\pi, a\sim\pi(s), z=\theta_s}[-\overline{\mathcal{T}}(s'|s,a) + \mathcal{T}_Z(s'|s,z)] \tag{A.101}$$

Thus, we have:

$$\begin{aligned}\mathrm{Err}_{d^\pi}(\pi, \pi_\theta, \overline{\mathcal{T}}) &= \frac{1}{2}\sum_{s'\in S}\left|\mathbb{E}_{s\sim d^\pi, a\sim\pi(s), z=\theta_s}[-\overline{\mathcal{T}}(s'|s,a) + \mathcal{T}_Z(s'|s,z)]\right| \\ &= \frac{1}{4}\sum_{s'\in S}\left|\sum_{s\in S} w(s')^\top \frac{\partial}{\partial\theta_s}\mathbb{E}_{\tilde{s}\sim d^\pi, a\sim\pi(\tilde{s})}[(\theta_{\tilde{s}} - \phi(\tilde{s},a))^2]\right| \\ &\leq \frac{1}{4}|S|\|w\|_\infty \cdot \left\|\frac{\partial}{\partial\theta}\mathbb{E}_{s\sim d^\pi, a\sim\pi(s)}[(\theta_s - \phi(s,a))^2]\right\|_1 . \end{aligned} \tag{A.102}$$

Then by combining Lemmas 16, 17, 22, and apply Equation (A.102) (as opposed to Lemma 18 as in the tabular case), we arrive at the desired bound in Theorem 7. □

A.2.7 Experiment Details for TRAIL

Architecture We parametrize ϕ as a two-hidden layer fully connected neural network with 256 units per layer. A Swish [492] activation function is applied to the output of each hidden layer. We use embedding size 64 for AntMaze and 256 for Ant and all DeepMind Control Suite (DMC) tasks after sweeping values of 64, 256, and 512, though we found TRAIL to be relatively robust to the latent dimension size as long as it is not too small (i.e., ≥ 64). The latent skills in temporal skill extraction require a much smaller dimension size, e.g., 8 or 10 as reported by [14, 12]. We tried increasing the latent skill size for these work during evaluation, but found the reported value 8 to work the best. We additionally experimented with different extend of skill extraction, but found the previously reported $t = 10$ to also work the best. We implement the trajectory encoder in OPAL, SkiLD, and SPiRL using a bidirectional LSTM with hidden dimension 256. We use $\beta = 0.1$ for the KL regularization term in the β VAE of OPAL (as reported). We also use 0.1 as the weight for SPiRL and SkiLD's KL divergence terms.

Training and Evaluation During pretraining, we use the Adam optimizer with learning rate 0.0003 for 200k iterations with batch size 256 for all methods that require pretraining. During downstream behavioral cloning, learned action representations are fixed, but the action decoder is fine-tuned on the expert data as suggested by [14]. Behavioral cloning for all methods including vanilla BC is trained with learning rate 0.0001 for 1M iterations. We experimented with learning rate decay of downstream BC by a factor of 3 at the 200k boundary for all methods. We found that when the expert sample size is small, decaying learning rate can prevent overfitting for all methods. The reported results are with learning rate decay on AntMaze and without learning rate decay on other environments for all methods. During the downstream behavioral cloning stage, we evaluate the latent policy combined with the action decoder every 10k steps by executing $q \circ \pi_Z$ in the environment for 10 episodes and compute the average total return. Each method is run with 4 seeds where each seed corresponds to one set of action representations and downstream imitation

learning result on that set of representations. We report the mean and standard error for all methods in the bar and line figures.

Modification to SkiLD and SPiRL Since SkiLD [12] and SPiRL [13] are originally designed for RL as opposed to imitation learning, we replace the downstream RL algorithms of SkiLD and SPiRL by behavioral cloning with regularization (but keep skill extraction the same as the original methods). Specifically, for SkILD, we apply a KL regularization term between the latent policy and the learned skill prior in the suboptimal offline dataset during pretraining, and another KL regularization term between the latent policy and a learn "skill posterior" on the expert data as done in the original paper during downstream behavioral cloning. We do not need to train the binary classifier that SkiLD trains to decide which regularizer to apply because we know which set of actions are expert versus suboptimal in the imitation learning setting. For SPiRL, we apply the KL divergence between latent policy and skill prior extracted from offline data (i.e., using the red term in Algorithm 1 of [13]) as an additional term to latent behavioral cloning.

Dataset Details

AntMaze. For the expert data in AntMaze, we use the goal-reaching expert policies trained by [14] (expert means that the agent is trained to navigate from the one corner of the maze to the opposite corner) to collect $n = 10$ trajectories. For the suboptimal data in AntMaze, we use the full D4RL datasets `antmaze-large-diverse-v0`, `antmaze-medium-play-v0`, `antmaze-medium-diverse-v0`, and `antmaze-medium-play-v0`.

Ant. For the expert data in Ant, we use a small set of expert trajectories selected by taking either the first 10k or 25k transitions from `ant-expert-v0` in D4RL, corresponding to about 10 and 25 expert trajectories, respectively. For the suboptimal data in Ant, we use the full D4RL datasets `ant-medium-v0`, `ant-medium-replay-v0`, and `ant-random-v0`.

RL Unplugged. For DeepMind Control Suite [11] set of tasks, we use the RL Unplugged [20] dataset. For the expert data, we take $\frac{1}{10}$ of the trajectories whose episodic reward is among the top 20% of the open source RL Unplugged datasets following the setup in [15]. For the suboptimal data, we use the bottom 80% of the RL Unplugged dataset. Table A.1 records the total number of trajectories available in RL Unplugged for each task (80% of which are used as suboptimal data), and the number of expert trajectories used in our evaluation.

Task	# Total	# $\mathcal{D}^*_{\text{off}}$
cartpole-swingup	40	2
cheetah-run	300	3
fish-swim	200	1
humanoid-run	3000	53
walker-stand	200	4
walker-walk	200	6

Table A.1: Total number of trajectories from RL Unplugged [20] locomotion tasks used to train CRR [16] and the number of expert trajectories used to train TRAIL. The bottom 80% of # Total is used to learn action representations by TRAIL.

A.2.8 Additional Empirical Results for TRAIL

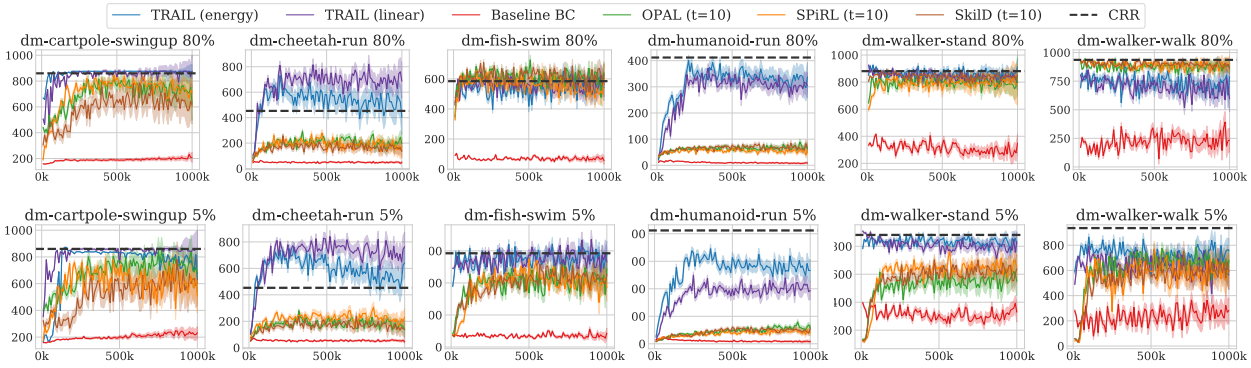


Figure A.13: Average task rewards (over 4 seeds) of TRAIL EBM (Theorem 5), TRAIL linear (Theorem 7), and OPAL, SkiLD, SPiRL trained on the bottom 80% (top) and bottom 5% (bottom) of the RL Unplugged datasets followed by behavioral cloning in the latent action space. Baseline BC achieves low rewards due to the small expert sample size. Dotted lines denote the performance of CRR [16] trained on the full dataset with reward labels.

Additional baselines for RL Unplugged

FrankaKitchen Results

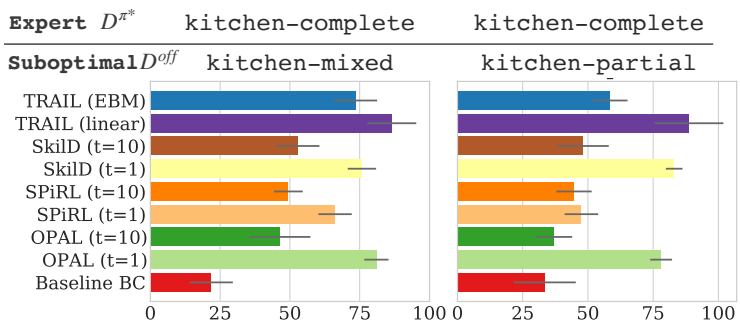


Figure A.14: Average rewards (over 4 seeds) of TRAIL EBM (Theorem 5), TRAIL linear (Theorem 7), and baseline methods pretrained on `kitchen-mixed` and `kitchen-partial` from D4RL to imitate `kitchen-complete`. TRAIL linear without temporal abstraction performs slightly better than SKiLD and OPAL with temporal abstraction over 10 steps.

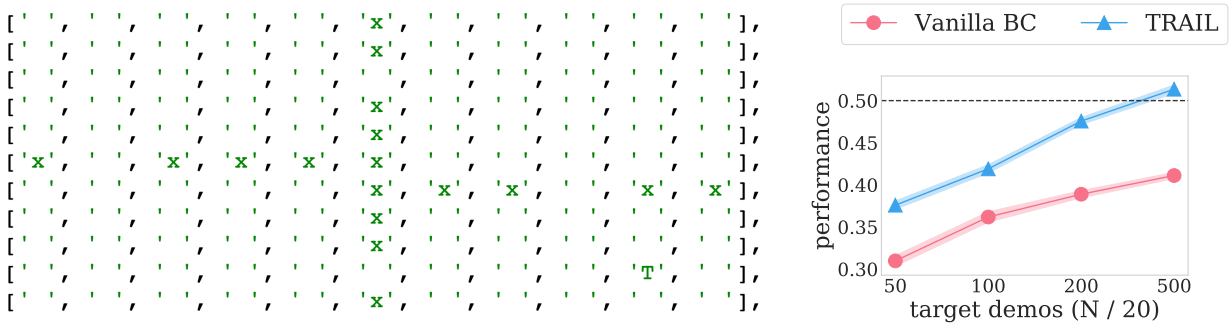


Figure A.15: Average task rewards (over 4 seeds) of TRAIL EBM (Theorem 5) and vanilla BC (right) in a discrete four-room maze environment (left) where an agent is randomly placed in the maze and tries to reach the target 'T'. TRAIL learns a discrete latent action space of size 4 from the discrete original action space of size 12 on 500 uniform random trajectories of length 20 shows clear benefit over vanilla BC on expert data.

Discrete Maze Results We conduct additional evaluation on an environment with tabular state and action spaces. As shown in Figure A.15, an agent is randomly placed into a four-room environment, and the task is to navigate to the target 'T'. The task reward is 1 at 'T' and 0 elsewhere. There are 12 discrete actions corresponding to rotating clockwise by $90, 180, 270, 360$ degrees, rotating counterclockwise by $90, 180, 270, 360$ degrees, moving forward by 1 or 2 grids, and moving backward by 1 or 2 grids (the action space is artificially blown up as suggested by the reviewer). TRAIL is pretrained on 500 trajectories of length 20 with uniform action selection. The expert demonstration always navigates to the target 'T' from any random starting location. TRAIL's latent action dimension is set to 4. We see

that TRAIL with a smaller latent action space offers benefits over vanilla BC.

A.2.9 Ablation Study for TRAIL

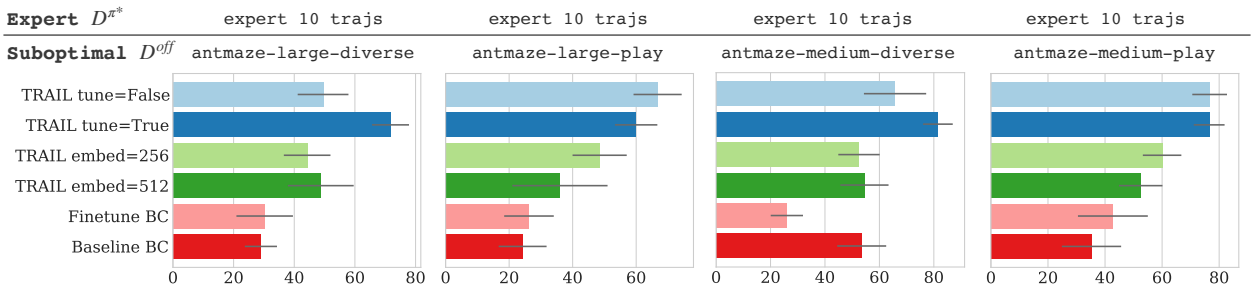


Figure A.16: Ablation study on action decoder finetuning, latent dimension size, and pretraining baseline BC on suboptimal data in the AntMaze environment. TRAIL with default embedding dimension 64 and finetuning the action decoder corresponds to the second row. Other dimension size (256 and 512) lead to worse performance. Finetuning the action decoder on the expert data has some small benefits. Pretraining BC on suboptimal data before finetuning on expert does not lead to significantly better performance.

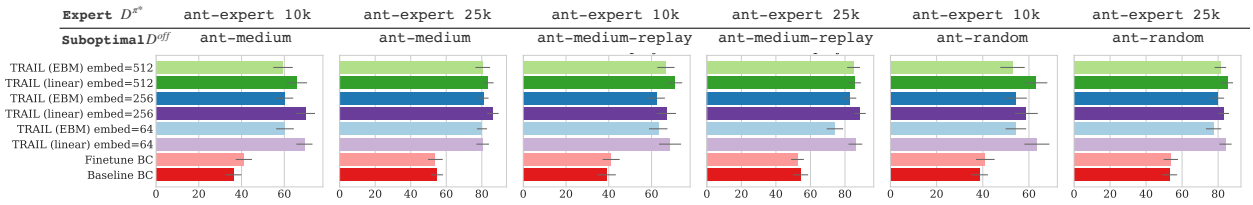


Figure A.17: Ablation study on latent dimension size in the Ant environment. TRAIL is generally robust to the choices of the latent action dimension (64, 256, 512) for the Ant task.

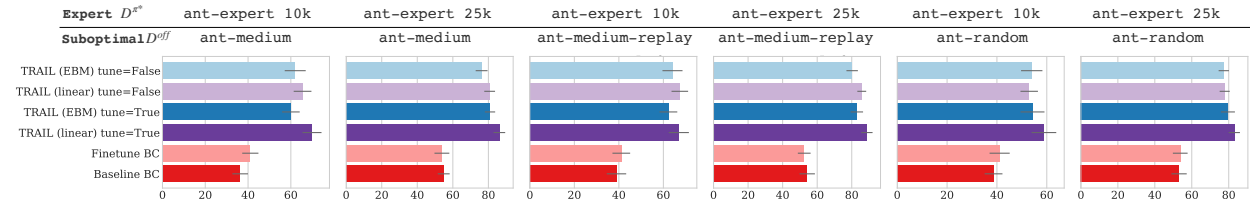


Figure A.18: Ablation study on finetuning the action decoder in the Ant environment. Finetuning the action decoder leads to a slight benefit.

A.2.10 Visualization of Latent Actions in TRAIL

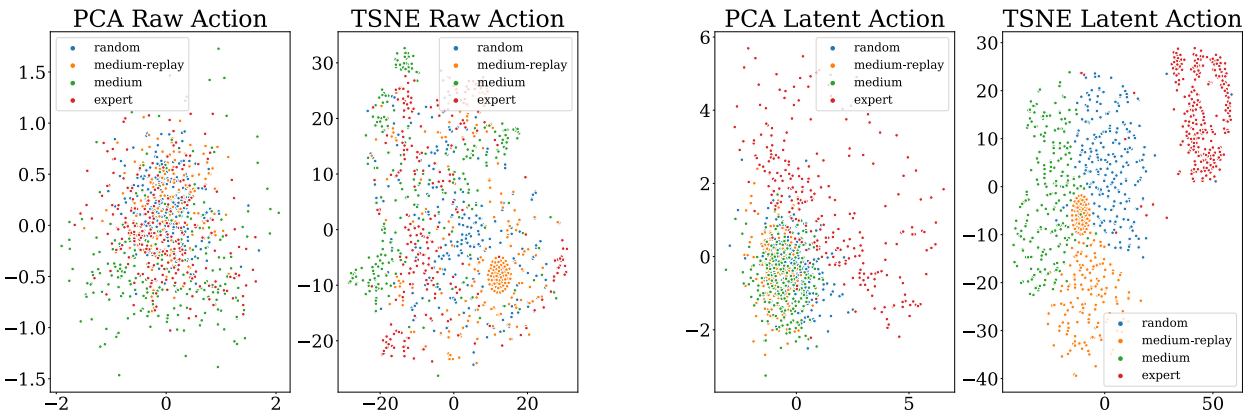


Figure A.19: PCA and t-SNE visualizations of the `random`, `medium-replay`, `medium`, and `expert` D4RL Ant datasets. Without action representation learning (left), the distinction between expert and suboptimal actions is not obvious. The latent actions of TRAIL (right), on the other hand, results in the expert latent actions being more visually separable from suboptimal actions.

A.3 Appendix for Procedure Cloning

A.3.1 Experimental details

Maze generation details We algorithmically generate maze layouts in a 2D gridworld environment. Each maze contains a randomly generated goal location and randomly generated internal walls that form a tunnel-like map as shown in Figure A.20. The long corridors of the maze layouts make it difficult to learn a good navigation policy with little data, especially when the maze is large. The maze layout generator ensures that there is at least one valid path from any empty location to the goal.

Details of training the AlphaZero-style MCTS expert For the expert MCTS policy on MinAtar, we modify the AlphaZero-style MCTS agent implementation from the Acme [495] library[1]. We use the MinAtar environment simulator for rollouts (as opposed to learning an environment model). During each MCTS simulation run, an action is selected

[1]Code of Acme: https://github.com/deepmind/acme

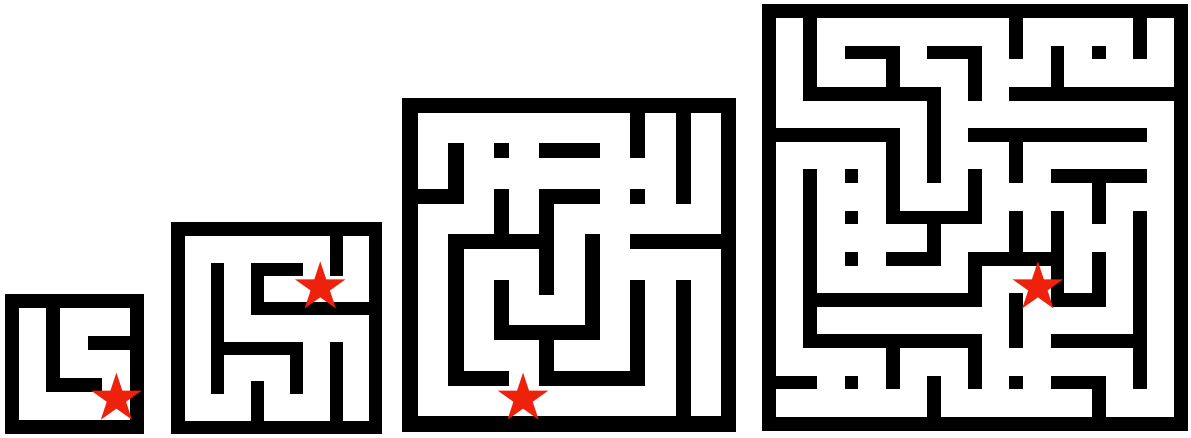


Figure A.20: Example maze layouts with increasing maze size.

similar to UCT [496] according to:

$$a_{\text{search}} = \underset{a}{\text{argmax}} \left[Q(s,a) + \beta \frac{\sqrt{N(s)}}{N(s,a)+1} \pi_{\text{prior}}(a \mid s) \right], \tag{A.103}$$

where $Q(s,a)$ is a learned value function using TD-learning, N is the visit-count of the tree node corresponding to s, β is a hyperparameter for controlling how greedily to search, and π_{prior} is a prior for guiding the MCTS search and is updated by minimizing $D_{\text{KL}}\left(\pi_{\text{prior}}(\cdot|s) \| \pi_{\text{MCTS}}(\cdot|s)\right)$, where $\pi_{\text{MCTS}}(\cdot|s)$ is the softmax of the node s's children visit-counts. The hyperparameters for training the MCTS expert is shown in Table A.2.

Table A.2: MCTS hyperparameters on MinAtar.

Hyperparameter	Value
Number of simulation runs	500
UCT β	1.0
TD discount γ	0.99
TD buffer size	100,000
Learning rate	0.0003
Batch size	64

Details for training PC policies

Predicting multiple goals on MinAtar. Since we do not limit the search depth of the MCTS agent explicitly, the resulting search tree after 500 simulation runs can have

arbitrarily long branches. Therefore, for collecting procedure data, we set a fixed depth (i.e., 15) as the goal which represents the state 15 steps into the future, and the GPT-like [27] autoregressive prediction model is trained on sequence of length 15 actions. We further found that predicting more than a single goal along the optimal path and more than a single backward action sequence during each time step improve PC's performance.

Number of expert samples in $\mathcal{D}_\Pi$. The timeout for collecting expert trajectories and for evaluating imitation-learned agents are set to 100, 1000, 2500 for discrete maze, AntMaze, and MinAtar respectively. Because some MinAtar games are easier to learn than others, i.e., Freeway only requires a single expert trajectory to learn a good BC policy, we take only the first 50 samples of a single trajectory as expert data for Freeway to increase the challenge of imitation learning. For the rest of the games, we take the full trajectories. Results in the main text use 10 expert trajectories for each game. Results for other number of expert trajectories can be found in Appendix A.3.2.

PC timeout. During evaluation, PC needs to predict the stopping condition to output a final action after predicting a series of intermediate procedures. This can result in the intermidiate procedure being arbitrarily long. We therefore use the environment timeout steps as a hard stopping condition for PC, and output an action sampled uniformly at random if PC times out.

Architecture and hyperparameters

Architecture. The CNNs used in discrete maze, AntMaze, and MinAtar of our experiments have 5 layers of convolution with kernel size (3, 3, 3, 3, 3), stride (1, 1, 1, 1, 1), channel (128, 128, 256, 256, 256), `SAME` padding, and without any pooling. The MLPs in discrete maze, AntMaze, and MinAtar have 2 hidden layers of 256 units each interleaved with ReLU activation. We have experimented with larger CNN and MLP model capacity, but they do not improve the generalization performance of the agents. The transformer autoregressive model in the MinAtar experiment has 4 self-attention layers of 8 heads with 128 hidden units each, followed by 2 feedforward layers of 256 units interleaved with ReLU activation. The bimanual sweep experiment follows the same architecture as [17] – this original architecture already performs autoregressive modelling on actions, so it is straightforward to incorporate procedure observations into this model, treating them as additional predictions that appear before the output actions. For all data augmentation baseline, we apply random crop of the size of the image, random horizontal and vertical translation of [-10%, +10%], and random horizontal and vertical zoom of [-10%, +10%] sequentially.

Training. For discrete maze, AntMaze, and MinAtar, we use the Adam optimizer with learning rate 3e-4, train the models for 500k steps with batch size 32, and evaluate every 10k steps. The line plots we present show the average results of the the last 3 evaluations.

Each model is trained on a single NVIDIA P100 GPU. For bimanual sweep, we follow the same training procedure as [17] using TPU v2.

A.3.2 Additional experimental results

Results on value iteration network We were curious about whether having a planning-like (procedure-like) module in the policy parametrization similar to Value Iteration Network (VIN) [497] can solve the generalization task in our maze setup. We therefore evaluate VIN on 5, 10, 20, 40 training mazes of size 8×8, 16×16, and 32×32 with 1 and 16 trajectories each and test if VIN can generalize to the 10 unseen test mazes. The original VIN paper used $k = 10$ (the number of VI blocks) and $k = 20$ for maze size 8×8, 16×16, which we follow, and use $k = 40$ for maze 32×32. Figure A.21 shows that VIN does not generalize in our setting, likely due to VIN heavily relying on heavily on the grid structure of the task, where as our tunnel-shaped map is highly convoluted and requires a lot more training data (VIN was originally trained on 5000 training mazes with 7 trajectories each, which is way more data than our evaluation setting).

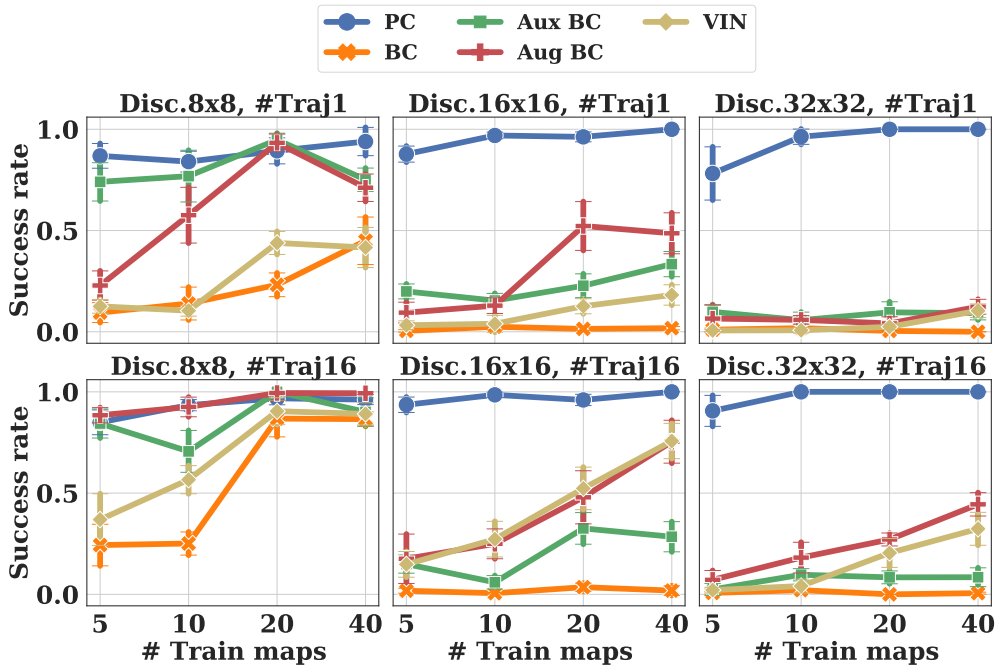


Figure A.21: Average success rate of PC, BC (variants), and VIN navigating to the goal from random start locations over 10 test maze layouts in discrete maze.

Additional results on discrete maze

Additional results on AntMaze

Additional results on MinAtar

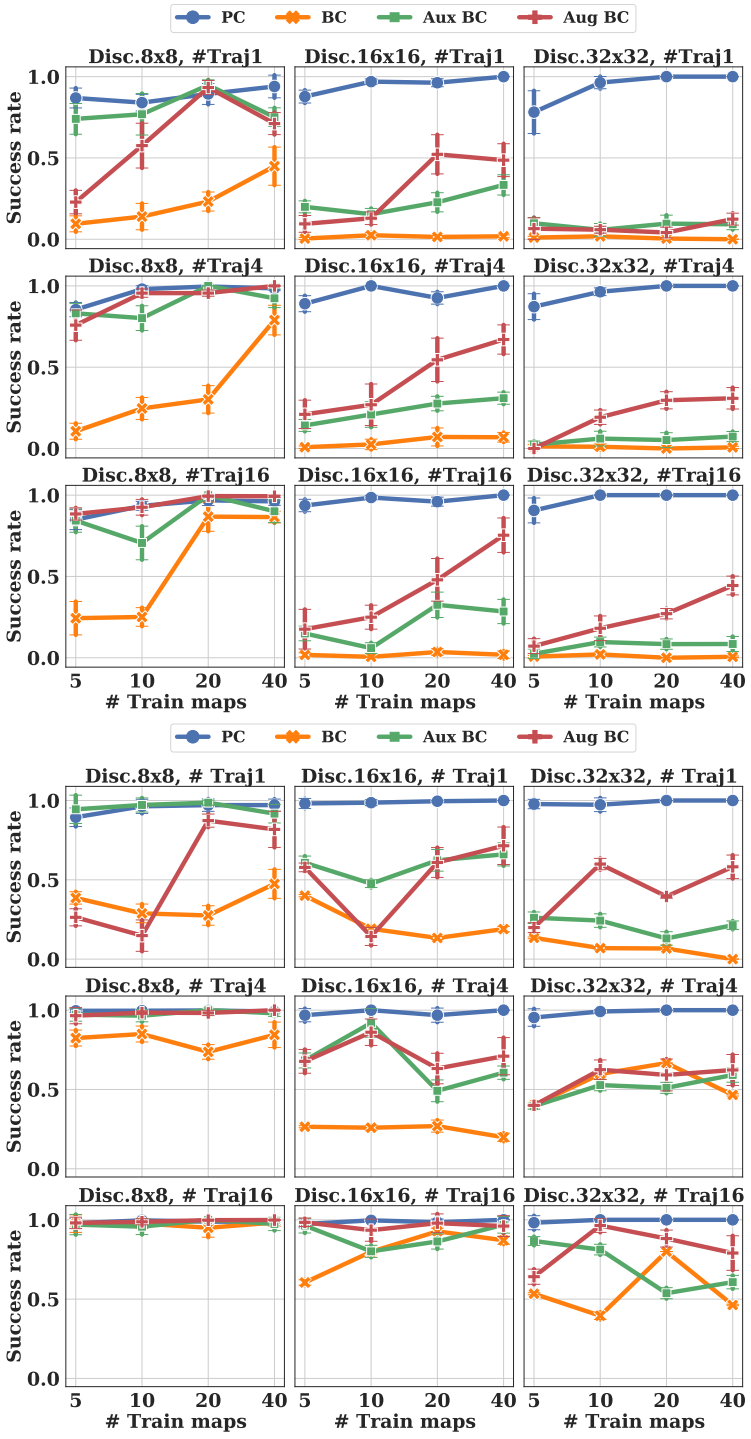


Figure A.22: Average success rate of PC and BC agents navigating to the goal from random start locations over testing mazes (top) and training mazes (bottom) in discrete maze‘. While BC variants can achieve high training performance when the number of expert trajectories is large, they still exhibit poor generalization performance on test mazes.

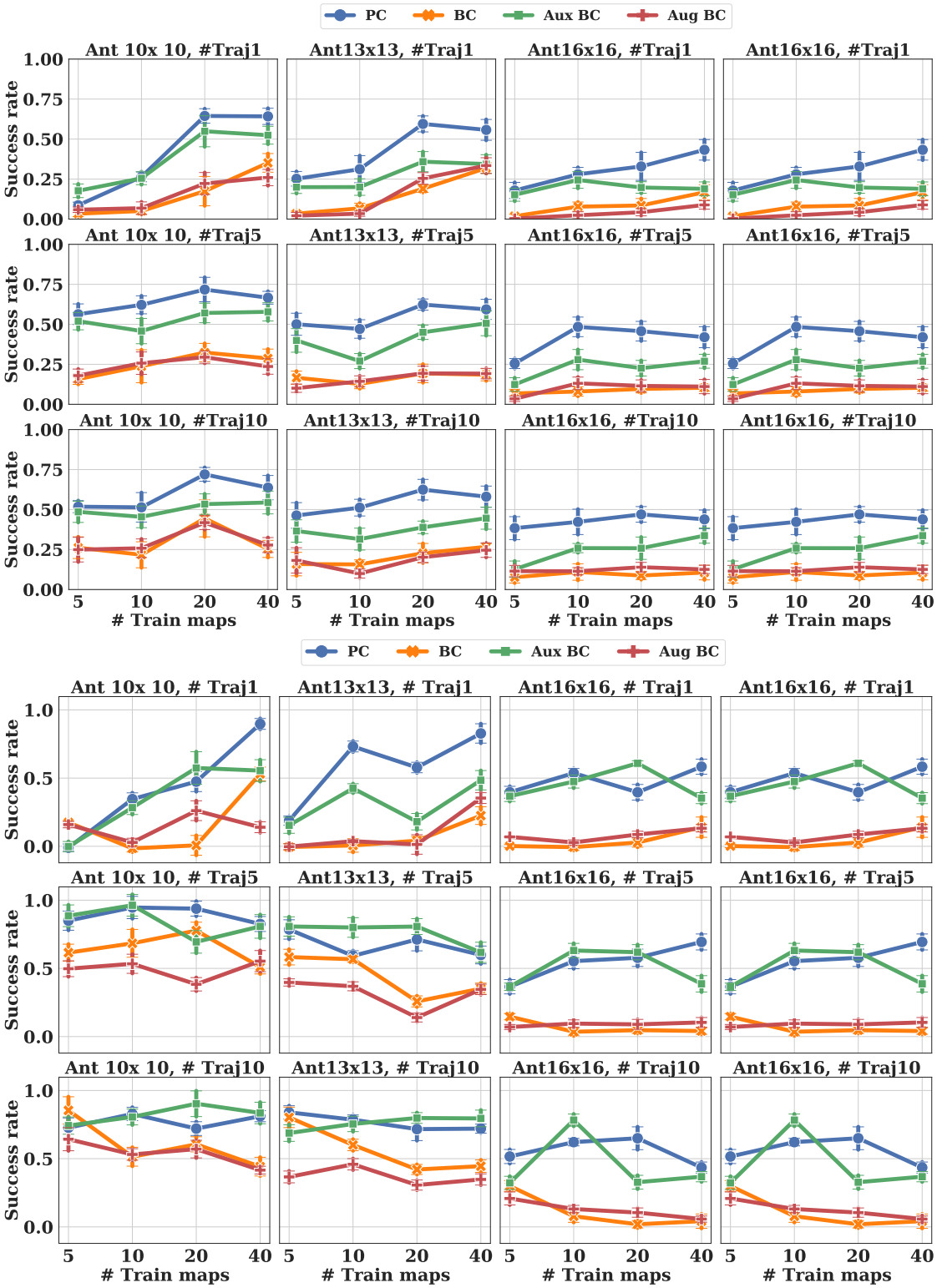


Figure A.23: Average success rate of PC and BC agents navigating to the goal from random start locations over testing mazes (top) and training mazes (bottom) in AntMaze. While Aux BC can achieve similar training performance as PC, Aux BC still exhibits poorer generalization performance on test mazes.

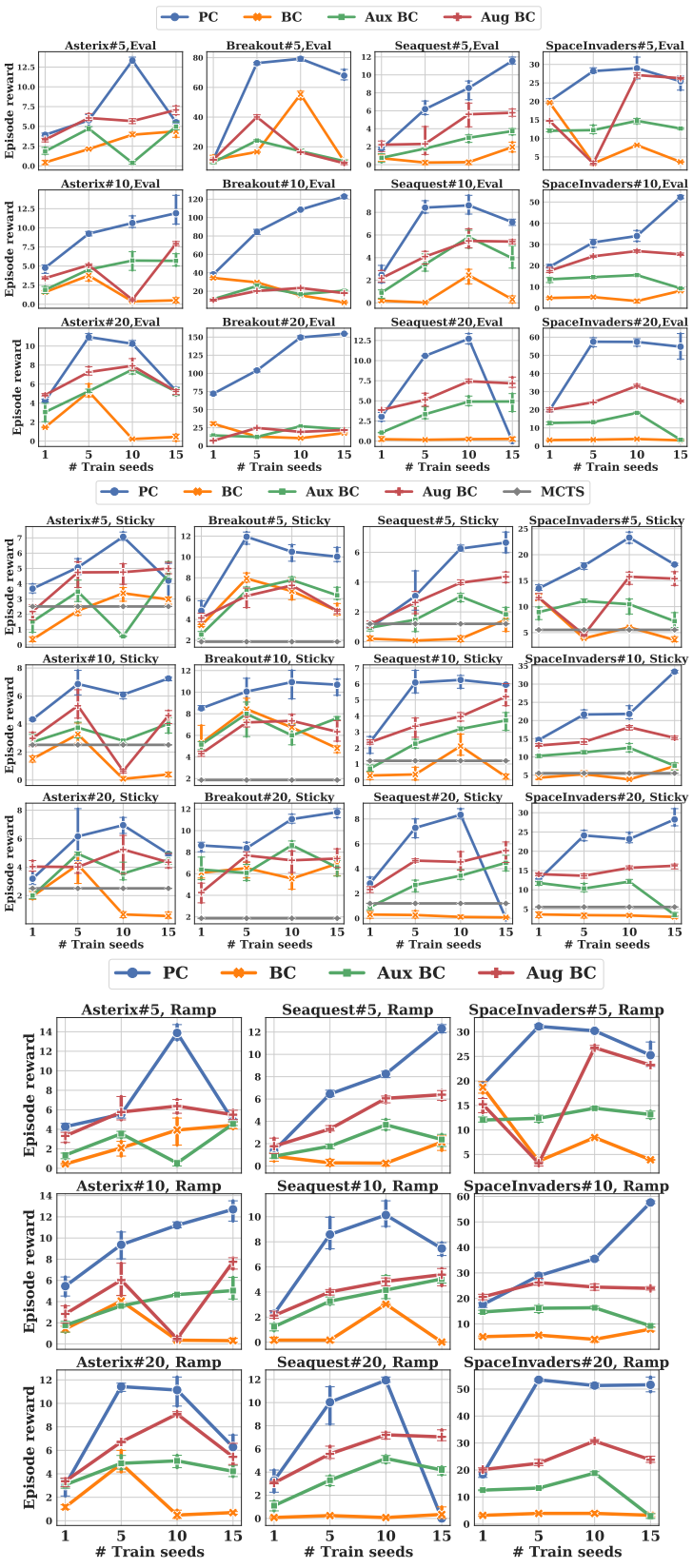


Figure A.24: Average episode reward of PC and BC on MinAtar over 3 test environments with the same configuration as training (top), with sticky actions (middle), and with difficulty ramping (bottom). Rows have different number of expert trajectories (50, 10, 20). PC

A.3.3 Further analysis of procedure cloning

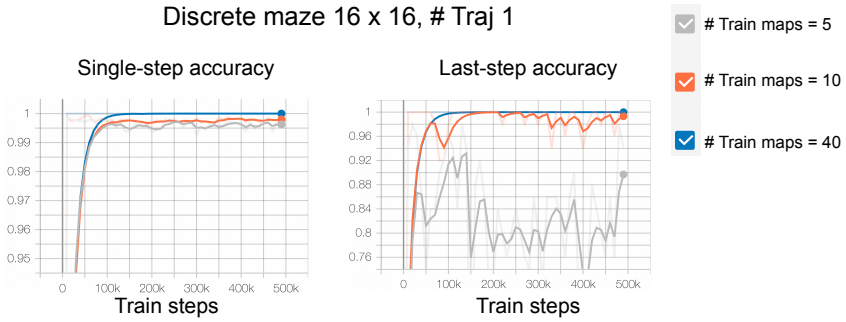


Figure A.25: Per-step accuracy of predicting visitation map and final expert action prediction accuracy with compounding error evaluated on discrete maze of size 16x16 where a single trajectory from each maze layout is used for training. The compounding error quickly becomes negligible as the number of training mazes increases (i.e., 10 training mazes).

Compounding error of PC during evaluation

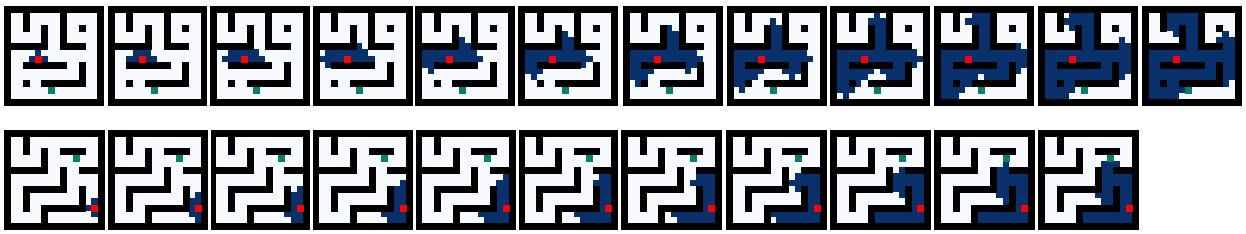


Figure A.26: Example visualizations of PC's learned BFS procedure on discrete maze environment. PC learned to conduct BFS from the goal (red) backwards to the agent location (green). For the ease of visualization, the exact actions taken by the learned BFS traversal are eliminated from the blue cells.

Visualization of learned PC procedure

Zero-shot generalization to greater distribution shift

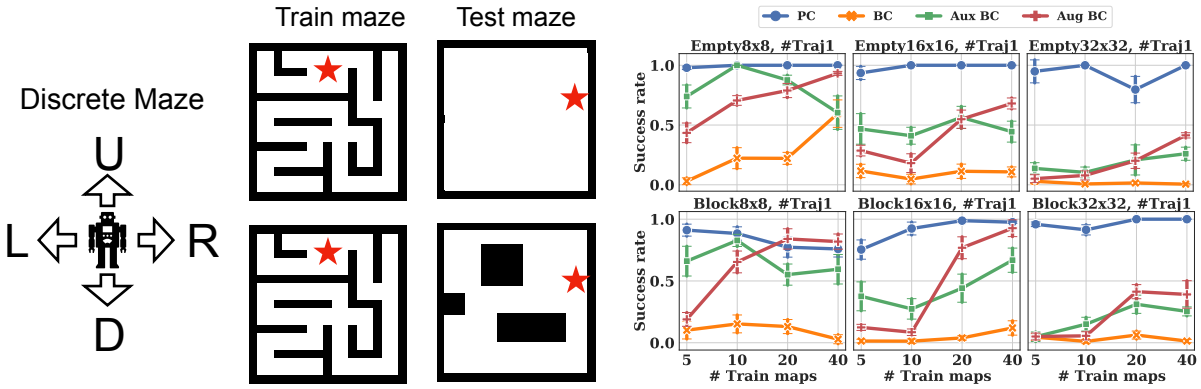


Figure A.27: [Left] Visualization of training and test mazes. Training mazes have tunnel-shaped inner walls whereas test mazes have no inner walls or block-shaped inner walls. [Right] Average success rate of PC and BC agents navigating to the goal from random start locations over 10 test mazes. Agents are trained on 5, 10, 20, 40 mazes of 1 trajectory each. We find that procedure cloning leads to much better test maze generalization compared to alternative approaches when the test mazes exhibit drastic distribution shift from the training mazes.

A.3.4 Additional related work

Neural program induction While neural program induction (NPI) may be interpreted to apply to any "program" similar to PC, its demonstrated applications (both in [498] and [499]) are based on problems that exhibit hierarchical and modular solutions with shared and repeated subprograms (e.g., a "pick" primitive within a block-stacking task). Thus, NPI not only advocates to imitate the full program but also parameterize the agent in a modular way — like a program with function calls and return values — to take advantage of this modular and repeated structure. The main argument in these existing papers hinges on the modular decomposition of both task and agent, which allows for more efficient data sharing, especially in multiple-task settings. In contrast, our PC evaluations are on environments with much less modular structures, and we avoid using specialized agent parameterizations in favor of more generic architectures (e.g., transformers). Thus, we believe our work is complementary to NPI, showing that the paradigm of using procedural information – while first proposed by the NPI work to some extent – applies to much more general settings than initially suggested by NPI.

Goal-conditional imitation learning. While some of our experiments may be interpreted as utilizing "goals" as intermediate computation, there is a key difference from goal-conditioned imitation learning [500, 501]. Namely, goal-conditioned imitation learning advocates for using portions of the observation (or learned functions of the observation) as goals, whereas PC uses information beyond what is available in the observation (e.g., coordinate positions of objects). Thus, the common argument in goal-conditioned imitation centers around getting more learning signal (goal-reaching) from the same data (s,a,s' tuples). In contrast, PC shows that having richer data in the demonstrations (procedure information) is useful, *even if that data is not available to the agent during inference*. We also note that another common argument made in goal-conditioned imitation learning is attributing their sample efficiency benefits to reduced temporal frequency induced by the hierarchical design, whereas PC shows benefits without any change in temporal frequency. Thus, we believe that PC presents a novel approach.

Learning-to-plan methods Learning-to-plan methods (e.g., Universal Planning Networks [502], Value Iteration Networks [497]) are distinct from PC in that they train policies end-to-end using state-action tuples. No additional supervision of the intermediate computations of the expert is used; rather, these methods effectively propose a different policy parameterization leveraging inductive biases (i.e., with an embedded end-to-end differentiable planner). As shown in our evaluations of Implicit BC and Value Iteration Networks (Appendix A.3.2), flexible policy parametrizations and architectures with desired inductive bias still fail to generalize without proper integration of procedure information during training, showing that PC provides benefits orthogonal to these existing works.

A.4 Appendix for Dichotomy of Control

A.4.1 Proof of Theorem 11

The proof relies on the following lemma, showing that the MI constraints ensure that the observed rewards and dynamics conditioned on z in the training data are equal to the rewards and dynamics of the environment.

Lemma 26. *Suppose DoC yields q satisfying the MI constraints:*

$$\mathrm{MI}(r_t; z|\tau_{0:t-1}, s_t, a_t) = \mathrm{MI}(s_{t+1}; z|\tau_{0:t-1}, s_t, a_t) = 0, \tag{A.104}$$

for all $\tau_{0:t-1}, s_t, a_t$ with $\Pr[\tau_{0:t-1}, s_t, a_t|\mathcal{D}] > 0$. Then under Assumption 9,

$$\Pr\left[\hat{r} = r_t \mid \tau_{0:t-1}, s_t, a_t, z, \mathcal{D}\right] = \mathcal{R}(\hat{r}_t \mid \tau_{0:t-1}, s_t, a_t), \tag{A.105}$$

$$\Pr\left[\hat{s}_{t+1} = s_{t+1} \mid \tau_{0:t-1}, s_t, a_t, z, \mathcal{D}\right] = \mathcal{T}(\hat{s}_{t+1} \mid \tau_{0:t-1}, s_t, a_t), \tag{A.106}$$

for all $\tau_{0:t-1}, s_t, a_t, z$ and $\hat{r}, \hat{s}_{t+1}$, as long as $\Pr[\tau_{0:t-1}, s_t, a_t, z|\mathcal{D}] > 0$.

Proof. We show the derivations relevant to reward, with those for next-state being analogous. We start with the definition of mutual information:

$$\mathrm{MI}(r_t; z|\tau_{0:t-1}, s_t, a_t) = \mathbb{E}_{(r_t,z)\sim\Pr[\cdot|\tau_{0:t-1},s_t,a_t,\mathcal{D}]}\left[\log \frac{\Pr\left[r_t|\tau_{0:t-1}, s_t, a_t, z, \mathcal{D}\right]}{\Pr\left[r_t|\tau_{0:t-1}, s_t, a_t, \mathcal{D}\right]}\right] \tag{A.107}$$

$$= \mathbb{E}_{z\sim\Pr[\cdot|\tau_{0:t-1},s_t,a_t,\mathcal{D}]}\left[D_{\mathrm{KL}}(\Pr[r|\tau_{0:t-1}, s_t, a_t, z, \mathcal{D}]\|\Pr[r|\tau_{0:t-1}, s_t, a_t, \mathcal{D}])\right]. \tag{A.108}$$

The KL divergence is a nonnegative quantity, and it is zero only when the two input distributions are equal. Thus, the constraint $\mathrm{MI}(r_t; z|\tau_{0:t-1}, s_t, a_t) = 0$ implies,

$$\Pr\left[r|\tau_{0:t-1}, s_t, a_t, z, \mathcal{D}\right] = \Pr[r|\tau_{0:t-1}, s_t, a_t, \mathcal{D}], \tag{A.109}$$

for all $\tau_{0:t-1}, s_t, a_t, z$ with $\Pr[z|\tau_{0:t-1}, s_t, a_t, \mathcal{D}] > 0$. From Assumption 9 we know

$$\Pr[r|\tau_{0:t-1}, s_t, a_t, \mathcal{D}] = \mathcal{R}(r|\tau_{0:t-1}, s_t, a_t), \tag{A.110}$$

and so we immediately have the desired result.

We will further employ the following lemma, which takes us most of the way to proving Theorem 11:

Lemma 27. *Suppose DoC yields π, q with q satisfying the MI constraints:*

$$\mathrm{MI}(r_t; z|\tau_{0:t-1}, s_t, a_t) = \mathrm{MI}(s_{t+1}; z|\tau_{0:t-1}, s_t, a_t) = 0, \tag{A.111}$$

for all $\tau_{0:t-1}, s_t, a_t$ with $\Pr[\tau_{0:t-1}, s_t, a_t|\mathcal{D}] > 0$. Then under Assumptions 9 and 10, we have

$$\Pr\left[\tau \mid z, \mathcal{D}\right] = \Pr\left[\tau \mid \pi_z, \mathcal{M}\right], \tag{A.112}$$

for all τ and all z with $\Pr[z|q, \mathcal{D}] > 0$.

Proof. We may write the probability $\Pr[\tau \mid z, \mathcal{D}]$ as,

$$\Pr[\tau \mid z, \mathcal{D}] = \prod_{t=0}^{H} \Pr[a_t \mid \tau_{0:t-1}, s_t, z, \mathcal{D}] \cdot \prod_{t=0}^{H} \Pr[r_t \mid \tau_{0:t-1}, s_t, a_t, z, \mathcal{D}] \cdot \prod_{t=0}^{H-1} \Pr[s_{t+1} \mid \tau_{0:t-1}, s_t, a_t, z, \mathcal{D}]. \quad \text{(A.113)}$$

Case 1: We begin by considering the case of τ satisfying $\Pr[\tau \mid z, \mathcal{D}] > 0$. For such a τ, by Assumption 10 we may write the first probability above as

$$\Pr[a_t \mid \tau_{0:t-1}, s_t, z, \mathcal{D}] = \pi_z(a_t|\tau_{0:t-1}, s_t). \quad \text{(A.114)}$$

Moreover, by Lemma 26 we may write the second and third probabilities as

$$\Pr[r_t \mid \tau_{0:t-1}, s_t, a_t, z, \mathcal{D}] = \mathcal{R}(r_t|\tau_{0:t-1}, s_t, a_t) \quad \text{(A.115)}$$
$$\Pr[s_{t+1} \mid \tau_{0:t-1}, s_t, a_t, z, \mathcal{D}] = \mathcal{T}(s_{t+1}|\tau_{0:t-1}, s_t, a_t). \quad \text{(A.116)}$$

Therefore, for any τ with $\Pr[\tau \mid z, \mathcal{D}] > 0$ we have,

$$\begin{aligned}\Pr[\tau \mid z, \mathcal{D}] &= \prod_{t=0}^{H} \pi_z(a_t|\tau_{0:t-1}, s_t) \cdot \prod_{t=0}^{H} \mathcal{R}(r_t|\tau_{0:t-1}, s_t, a_t) \cdot \prod_{t=0}^{H-1} \mathcal{T}(s_{t+1}|\tau_{0:t-1}, s_t, a_t) \\ &= \Pr[\tau \mid \pi_z, \mathcal{M}]. \end{aligned} \quad \text{(A.117)}$$

Case 2: To handle the case of $\Pr[\tau \mid z, \mathcal{D}] = 0$ we will show that $\Pr[\tau_{0:t} \mid z, \mathcal{D}] = 0$ implies $\Pr[\tau_{0:t} \mid \pi_z, \mathcal{M}] = 0$ by induction on t. The base case of $t = -1$ is trivial. For $t > -1$, we may write,

$$\Pr[\tau_{0:t} \mid z, \mathcal{D}] = \Pr[\tau_{0:t-1} \mid z, \mathcal{D}] \cdot \Pr[s_t \mid \tau_{0:t-2}, s_{t-1}, a_{t-1}, z, \mathcal{D}] \cdot \Pr[a_t \mid \tau_{0:t-1}, s_t, z, \mathcal{D}] \cdot \Pr[r_t \mid \tau_{0:t-1}, s_t, a_t, z, \mathcal{D}], \quad \text{(A.118)}$$

$$\Pr[\tau_{0:t} \mid \pi_z, \mathcal{M}] = \Pr[\tau_{0:t-1} \mid \pi_z, \mathcal{M}] \cdot \mathcal{T}(s_t|\tau_{0:t-2}, s_{t-1}, a_{t-1}) \cdot \pi_z(a_t|\tau_{0:t-1}, s_t) \cdot \mathcal{R}(r_t|\tau_{0:t-1}, s_t, a_t). \quad \text{(A.119)}$$

Suppose, for the sake of contradiction, that $\Pr[\tau_{0:t} \mid z, \mathcal{D}] = 0$ while $\Pr[\tau_{0:t} \mid \pi_z, \mathcal{M}] > 0$. By the inductive hypothesis, $\Pr[\tau_{0:t-1} \mid z, \mathcal{D}] > 0$. Thus, by Lemma 26 we must have

$$\Pr[s_t \mid \tau_{0:t-2}, s_{t-1}, a_{t-1}, z, \mathcal{D}] = \mathcal{T}(s_t|\tau_{0:t-2}, s_{t-1}, a_{t-1}), \quad \text{(A.120)}$$

and so $\mathcal{T}(s_t|\tau_{0:t-2}, s_{t-1}, a_{t-1}) > 0$ implies that $\Pr[s_t \mid \tau_{0:t-2}, s_{t-1}, a_{t-1}, z, \mathcal{D}] > 0$. Thus, by Assumption 10 we must have

$$\Pr[a_t \mid \tau_{0:t-1}, s_t, z, \mathcal{D}] = \pi_z(a_t|\tau_{0:t-1}, s_t), \tag{A.121}$$

and so $\pi_z(a_t|\tau_{0:t-1}, s_t) > 0$ implies that $\Pr[a_t \mid \tau_{0:t-1}, s_t, z, \mathcal{D}] > 0$. Lastly, by Lemma 26 we must have

$$\Pr[r_t \mid \tau_{0:t-1}, s_t, a_t, z, \mathcal{D}] = \mathcal{R}(r_t|\tau_{0:t-1}, s_t, a_t), \tag{A.122}$$

and so $\mathcal{R}(r_t|\tau_{0:t-1}, s_t, a_t) > 0$ implies that $\Pr[r_t \mid \tau_{0:t-1}, s_t, a_t, z, \mathcal{D}] > 0$. Altogether, we find that each of the three terms on the RHS of Equation (A.118) is strictly positive and so $\Pr[\tau_{0:t} \mid z, \mathcal{D}] > 0$; contradiction.

Theorem Proof We are now prepared to prove Theorem 11.

Using Assumption 10, we can express $V(z)$ as,

$$V(z) = \int \Pr[\hat{\tau} = \tau \mid z, \mathcal{D}] \cdot R(\hat{\tau})\, d\hat{\tau}. \tag{A.123}$$

By Lemma 27 we have,

$$V(z) = \int \Pr[\hat{\tau} = \tau \mid z, \mathcal{D}] \cdot R(\hat{\tau})\, d\hat{\tau} \tag{A.124}$$

$$= \int \Pr[\hat{\tau} = \tau \mid \pi_z, \mathcal{M}] \cdot R(\hat{\tau})\, d\hat{\tau} \tag{A.125}$$

$$= V_{\mathcal{M}}(\pi_z), \tag{A.126}$$

as desired.

A.4.2 Proof of Theorem 14

We begin by proving a result under stricter conditions, namely, when the MI constraints retain the conditioning on history.

Lemma 28. *Suppose DoC yields π, V, q with q satisfying the MI constraints:*

$$\mathrm{MI}(r_t; z|\tau_{0:t-1}, s_t, a_t) = \mathrm{MI}(s_{t+1}; z|\tau_{0:t-1}, s_t, a_t) = 0, \tag{A.127}$$

for all $\tau_{0:t-1}, s_t, a_t$ with $\Pr[\tau_{0:t-1}, s_t, a_t|\mathcal{D}] > 0$. Then under Assumptions 9, 12, and 13, V and π are consistent for any z with $\Pr[z|q, \mathcal{D}] > 0$.

Proof. Let

$$\pi_z^{\text{hist}}(\hat{a} \mid \tau_{0:t-1}, s_t) = \Pr[\hat{a} = a_t \mid \tau_{0:t-1}, s_t, z, \mathcal{D}]. \tag{A.128}$$

By Lemma 27 and Theorem 11 we have

$$\Pr[\tau \mid z, \mathcal{D}] = \Pr\left[\tau \mid \pi_z^{\text{hist}}, \mathcal{M}\right], \tag{A.129}$$

for all τ and

$$V(z) = V_{\mathcal{M}}(\pi_z^{\text{hist}}), \tag{A.130}$$

for all z with $\Pr[z \mid q, \mathcal{D}] > 0$.

It is left to show that $V_{\mathcal{M}}(\pi_z^{\text{hist}}) = V_{\mathcal{M}}(\pi_z)$. To do so, we invoke Theorem 5.5.1 in [503], which states that, for any history-dependent policy, there exists a Markov policy such that the state-action visitation occupancies of the two policies are equal (and, accordingly, their values are equal). In other words, there exists a Markov policy $\tilde{\pi}_z$ such that

$$\Pr\left[\hat{s} = s_t, \hat{a} = a_t \mid \pi_z^{\text{hist}}, \mathcal{M}\right] = \Pr\left[\hat{s} = s_t, \hat{a} = a_t \mid \tilde{\pi}_z, \mathcal{M}\right], \tag{A.131}$$

for all $t, \hat{s}, \hat{a}$, and

$$V_{\mathcal{M}}(\pi_z^{\text{hist}}) = V_{\mathcal{M}}(\tilde{\pi}_z). \tag{A.132}$$

To complete the proof, we show that $\tilde{\pi}_z = \pi_z$. By Equation (A.129) we have

$$\Pr\left[\hat{s} = s_t, \hat{a} = a_t \mid \pi_z^{\text{hist}}, \mathcal{M}\right] = \Pr\left[\hat{s} = s_t, \hat{a} = a_t \mid z, \mathcal{D}\right]. \tag{A.133}$$

Thus, for any $t, \hat{s}, \hat{a}$ we have

$$\tilde{\pi}_z(\hat{a} = a_t | \hat{s} = s_t) = \frac{\Pr\left[\hat{s} = s_t, \hat{a} = a_t \mid \tilde{\pi}_z, \mathcal{M}\right]}{\Pr\left[\hat{s} = s_t \mid \tilde{\pi}_z, \mathcal{M}\right]} \tag{A.134}$$

$$= \frac{\Pr\left[\hat{s} = s_t, \hat{a} = a_t \mid \pi_z^{\text{hist}}, \mathcal{M}\right]}{\Pr\left[\hat{s} = s_t \mid \pi_z^{\text{hist}}, \mathcal{M}\right]} \tag{A.135}$$

$$= \frac{\Pr\left[\hat{s} = s_t, \hat{a} = a_t \mid z, \mathcal{D}\right]}{\Pr\left[\hat{s} = s_t \mid z, \mathcal{D}\right]} \tag{A.136}$$

$$= \pi_z(\hat{a} = a_t | \hat{s} = s_t), \tag{A.137}$$

where the first equality is Bayes' rule, the second equality is due to Equation (A.131), the third equality is due to Equation (A.133), and last equality is by definition of π_z (Assumption 13).

Before continuing to the main proof, we present the following analogue to Lemma 26:

Lemma 29. *Suppose DoC yields q satisfying the MI constraints:*

$$\text{MI}(r_t; z | s_t, a_t) = \text{MI}(s_{t+1}; z | s_t, a_t) = 0, \tag{A.138}$$

for all s_t, a_t with $\Pr[s_t, a_t | \mathcal{D}] > 0$. Then under Assumptions 9 and 12,

$$\Pr\left[\hat{r} = r_t \mid s_t, a_t, z, \mathcal{D}\right] = \mathcal{R}(\hat{r}_t \mid s_t, a_t), \tag{A.139}$$

$$\Pr\left[\hat{s}_{t+1} = s_{t+1} \mid s_t, a_t, z, \mathcal{D}\right] = \mathcal{T}(\hat{s}_{t+1} \mid s_t, a_t), \tag{A.140}$$

for all s_t, a_t, z and $\hat{r}, \hat{s}_{t+1}$, as long as $\Pr[s_t, a_t, z | \mathcal{D}] > 0$.

Proof. The proof is analogous to the proof of Lemma 26.

Theorem Proof We can now tackle the proof of Theorem 14. To do so, we start by interpreting the episodes τ in the training data $\mathcal{D}$ as coming from a modified Markovian environment $\mathcal{M}^\dagger$. Specifically, we define $\mathcal{M}^\dagger$ as an environment with the same state space as $\mathcal{M}$ but with an action space consisting of tuples (a, r, s'), where a is an action from the action space of $\mathcal{M}$, r is a scalar, and s' is a state from the state space of $\mathcal{M}$. We define the reward and transition functions of $\mathcal{M}^\dagger$ to be deterministic, so that the reward and next state associated with (a, r, s') is r and s', respectively. This way, we may interpret any episode $\tau = (s_t, a_t, r_t)_{t=0}^H$ in $\mathcal{M}$ as an episode

$$\tau^\dagger = (s_t, (a_t, r_t, s_{t+1}), r_t)_{t=0}^H \tag{A.141}$$

in the modified environment $\mathcal{M}^\dagger$. Denoting $\mathcal{D}^\dagger$ as the training data distribution when interpreted in this way, we note that the MI constraints of Lemma 28 hold, since rewards and transitions are deterministic. Thus, the policy $\pi^\dagger$ defined as

$$\pi^\dagger((\hat{a}, \hat{r}, \hat{s}')|s_t, z) = \Pr[(\hat{a}, \hat{r}, \hat{s}') = (a_t, r_t, s_{t+1})|s_t, z, \mathcal{D}^\dagger] \tag{A.142}$$

satisfies

$$V(z) = V_{\mathcal{M}^\dagger}(\pi_z^\dagger). \tag{A.143}$$

It is left to show that $V_{\mathcal{M}^\dagger}(\pi_z^\dagger) = V_{\mathcal{M}}(\pi_z)$. To do so, consider an episode $\tau^\dagger \sim \Pr[\cdot|\pi_z^\dagger, \mathcal{M}^\dagger]$. For any single-step transition in this episode,

$$(s_t, (a_t, r_t, s_{t+1}), r_t, s_{t+1}), \tag{A.144}$$

we have, by definition of $\pi_z^\dagger$,

$$\Pr[\hat{a} = a_t|s_t, \pi_z^\dagger] = \Pr[\hat{a} = a_t|s_t, z, \mathcal{D}^\dagger] = \pi_z(\hat{a}|s_t). \tag{A.145}$$

In a similar vein, by definition of $\pi_z^\dagger$ and Lemma 29 we have,

$$\Pr[\hat{r} = r_t|s_t, a_t, \pi_z^\dagger] = \Pr[\hat{r} = r_t|s_t, a_t, z, \mathcal{D}^\dagger] = \mathcal{R}(\hat{r}|s_t, a_t), \tag{A.146}$$

$$\Pr[\hat{s}_{t+1} = s_{t+1}|s_t, a_t, \pi_z^\dagger] = \Pr[\hat{s}_{t+1} = s_{t+1}|s_t, a_t, z, \mathcal{D}^\dagger] = \mathcal{T}(\hat{s}_{t+1}|s_t, a_t). \tag{A.147}$$

Thus, any $\tau^\dagger = (s_t, (a_t, r_t, s_{t+1}), r_t)_{t=0}^H$ sampled from $\pi_z^\dagger, \mathcal{M}^\dagger$ can be mapped back to a $\tau = (s_t, a_t, r_t)_{t=0}^H$ in the original environment $\mathcal{M}$, where $\Pr[\tau^\dagger|\pi_z^\dagger, \mathcal{M}^\dagger] = \Pr[\tau|\pi_z, \mathcal{M}]$. It is clear that $R(\tau^\dagger) = R(\tau)$, and so we immediately have

$$V_{\mathcal{M}^\dagger}(\pi_z^\dagger) = V_{\mathcal{M}}(\pi_z), \tag{A.148}$$

as desired.

A.4.3 Invalidity of Alternative Consistency Frameworks

[103] propose a similar but distinct notion of consistency compared to ours (i.e., Definition 8), and claim that it can be achieved with stationary policies in Markovian environments. In this section, we show that this is, in fact, false, supporting the benefits of our framework. We begin by rephrasing Theorem 2.1 of [103] using our own notation:

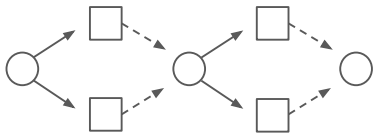


Figure A.28: Deterministic environment used in the counter-example described in Appendix A.4.3. Circles represent states and squares represent actions; solid arrows represent choice of actions and dashed arrows represent environment dynamics.

(Incorrect) Theorem 2.1 of [103]. *Suppose $\mathcal{M}$ is Markovian and $\mathcal{D}, q$ are given such that*

$$\Pr[\hat{s}_{t+1} = s_t \mid s_t, a_t, z, \mathcal{D}] = \Pr[\hat{s}_{t+1} = s_t \mid s_t, a_t, \mathcal{D}], \tag{A.149}$$

for all $s_t, a_t, z, \hat{s}_{t+1}$ with $\Pr[s_t, a_t, z|q, \mathcal{D}] > 0$ and define a Markov policy π as

$$\pi(\hat{a}|s_t, z) = \Pr[\hat{a} = a_t|s_t, z, \mathcal{D}]. \tag{A.150}$$

Then for any z with $\Pr[z|q, \mathcal{D}] > 0$ and any τ,

$$\Pr[\tau \mid \pi_z, \mathcal{M}] > 0 \text{ if and only if } \Pr[\tau \mid z, \mathcal{D}] > 0. \tag{A.151}$$

Counter-example. A simple counter-example may be constructed by considering the Markovian environment displayed in Figure A.28. The environment has three states. The first state gives a choice of two actions ($a_0 \in \{0, 1\}$), and each action deterministically transitions to the same second state. The second state again provides a choice of two actions ($a_1 \in \{0, 1\}$), and each of these again deterministically transitions to the same terminal state. Thus, episodes in this environment are uniquely determined by choice of a_0, a_1. There are four unique episodes:

$$\tau_0 = \langle a_0 = 0, a_1 = 0 \rangle, \tag{A.152}$$
$$\tau_1 = \langle a_0 = 1, a_1 = 1 \rangle, \tag{A.153}$$
$$\tau_2 = \langle a_0 = 0, a_1 = 1 \rangle, \tag{A.154}$$
$$\tau_3 = \langle a_0 = 1, a_1 = 0 \rangle. \tag{A.155}$$

We now construct q as a deterministic function, clustering these four trajectories into two distinct z:

$$z_0 = q(\tau_0) = q(\tau_1), \tag{A.156}$$
$$z_1 = q(\tau_2) = q(\tau_3). \tag{A.157}$$

Suppose $\mathcal{D}$ includes $\tau_0, \tau_1, \tau_2, \tau_3$ with equal probability. Since the environment is deterministic, the conditions of Theorem 2.1 in [103] are trivially satisfied. Learning a policy π with

respect to z_0 yields

$$\pi(\cdot|s_0, z_0) = [0.5, 0.5], \tag{A.158}$$
$$\pi(\cdot|s_1, z_0) = [0.5, 0.5]. \tag{A.159}$$

However, it is clear that interacting with $\pi(\cdot|\cdot, z_0)$ in the environment will lead to τ_2, τ_3 with non-zero probability, while τ_2, τ_3 are never associated with z_0 in the data $\mathcal{D}$. Contradiction.

A.4.4 Pseudocode for DoC training

Algorithm 3 Training with Dichotomy of Control

Inputs Offline dataset $\mathcal{D} = \{\tau^{(m)}\}_{m=1}^{M}$ where $\tau^{(m)} = (s_t^{(m)}, a_t^{(m)}, r_t^{(m)})_{t=0}^{H}$ with initial states $\{s_0^{(m)}\}_{m=1}^{M}$ and initial return-to-go values $\{R^{(m)}\}_{m=1}^{M}$, a parametrized distribution $q_\phi(\cdot)$, a policy $\pi_{\theta_1}(\cdot, \cdot)$, a value function $V_{\theta_2}(\cdot)$, a prior $p_\psi(\cdot)$, an energy function $f_w(\cdot)$, a fixed distribution $\rho(r, s')$, learning rates η, and training batch size B.
while training has not converged **do**
  Sample batch $\{(\tau = (s_t, a_t, r_t)_{t=0}^{H})^{(m)}\}_{m=1}^{B}$ from $\mathcal{D}$, for $m = 1, \ldots, B$.
  Sample z from $q_\phi(\tau)$ with reparametrization.
  Compute $\mathcal{L}_{\text{DoC}} + \mathcal{L}_{aux}$ according to Equation 5.12 and Equation 5.13.
  Update $\phi \leftarrow \phi - \eta\nabla_\phi\hat{\mathcal{L}}$, $\psi \leftarrow \psi - \eta\nabla_\psi \text{stopgrad}(\hat{\mathcal{L}}, \phi)$, $w \leftarrow w + \eta\nabla_w\hat{\mathcal{L}}$, $\theta_1 \leftarrow \theta_1 - \eta\nabla_{\theta_1}\hat{\mathcal{L}}$, $\theta_2 \leftarrow \theta_2 - \eta\nabla_{\theta_1}\hat{\mathcal{L}}$.
return $\pi_{\theta_1}(\cdot, \cdot), V_{\theta_2}(\cdot), p_\psi(\cdot)$

A.4.5 Experiment Details

Hyperparameters. We use the same hyperparameters as the publically available Decision Transformer [60] implementation. For VAE, we additionally learn a future and a prior both parametrized the same as the policy using transformers with context length 20. All models are trained on NVIDIA GPU P100.

Table A.3: Hyperparameters of Decision Transformer, future-conditioned VAE, and Dichotomy of Control.

Hyperparameter	**Value**
Number of layers	3
Number of attention heads	1
Embedding dimension	128
Latent future dimension	128
Nonlinearity function	ReLU
Batch size	64
Context length K	20 FrozenLake, HalfCheetah, Hopper, Humanoid, AntMaze 5 Reacher
Future length K_f	Same as context length K
Return-to-go conditioning for DT	1 FrozenLake 6000 HalfCheetah 3600 Hopper 5000 Humanoid 50 Reacher 1 AntMaze
Dropout	0.1
Learning rate	10^{-4}
Grad norm clip	0.25
Weight decay	10^{-4}
Learning rate decay	Linear warmup for first 10^5 training steps
β coefficient	1.0 for DoC, Best of $0.1, 1.0, 10$ for VAE

Details of the offline datasets

FrozenLake. We train a DQN [38] policy for 100k steps in the original 4x4 FrozenLake Gym environment with stochasticity level $p = \frac{1}{3}$. We then modify p to simulate environments of different stochasticity levels, while collecting 100 trajectories of maximum length 100 at each level using the trained DQN agent with probability ϵ of selecting a random action as opposed to the action output by the DQN agent to emulate offline data with different quality.

Gym MuJoCo. We train SAC [378] policies on the original set of Gym MuJoCo environments for 100M steps. To simulate stochasticity in these environments, we modify the original Gym MuJoCo environments by introducing noise to the actions before inputting the action to the physics simulator to compute rewards and next states. The noise has 0 mean and standard deviation of the form $(1 - e^{-0.01 \cdot t}) \cdot \sin(t) \cdot \sigma$ where t is the step number and $\sigma \in [0, 1]$. We then collect 1000 trajectories of 1000 steps each for all environments except for Reacher (which has 50 steps in each trajectory) in the stochastic version of the environment using the SAC policy to acquire the offline dataset for training.

AntMaze. For the AntMaze task, we use the AntMaze dataset from D4RL [10], which contains 1000 trajectories of 1000 steps each. We add gaussian noise with standard deviation 0.1 to the rewards in the dataset uniformly with probability 0.1 to both the offline dataset and during environment evaluation to simulate stochastic rewards from the environment.

A.4.6 Additional Results

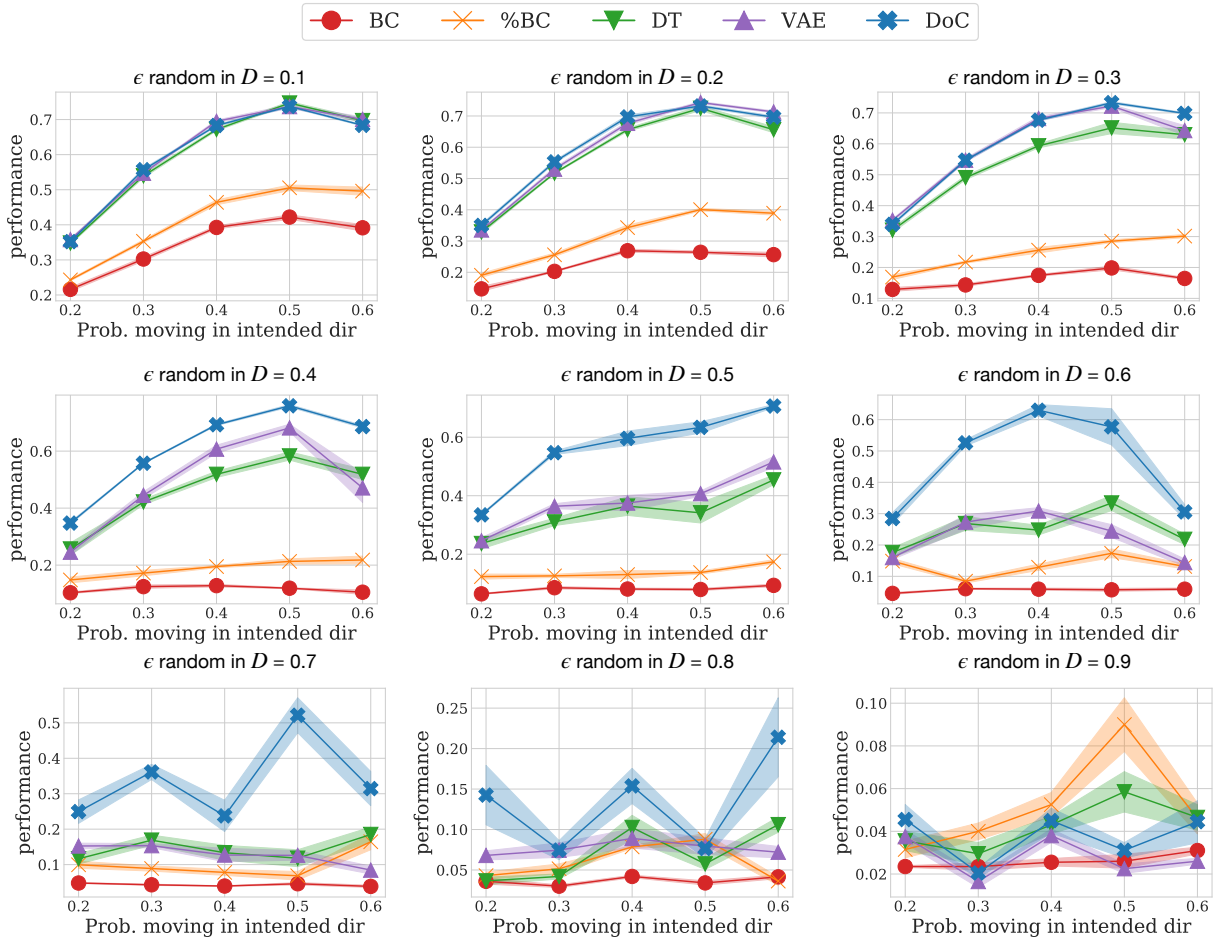


Figure A.29: Average performance (across 5 seeds) of DoC and baselines on FrozenLake with different levels of stochasticity (p) and offline dataset quality (ϵ). DoC outperforms DT and future VAE with bigger gains the offline data is less optimal.

FrozenLake with different offline dataset quality.

Improvement of DoC over RvS To test the effect of applying the MI constraint to other future-conditioned supervised learning baselines, we evaluate RvS parametrized by MLP policies [361] with VAE and DoC modifications. In general, MLP parametrization performs worse than transformer parametrization, but DoC is still able to provide significant benefit over vanilla RvS.

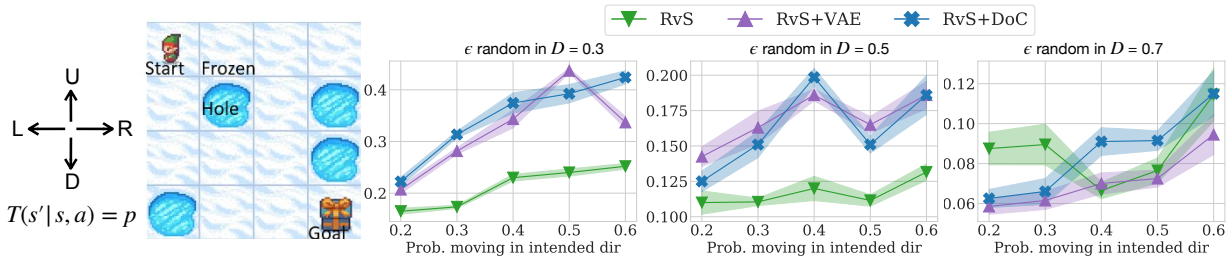


Figure A.30: Average performance (across 5 seeds) of DoC and baselines on FrozenLake with different levels of stochasticity (p) and offline dataset quality (ϵ). DoC outperforms RvS and future VAE with bigger gains the offline data is less optimal.

A.4.7 Additional Ablations

VAE with Stop Gradient One difference between DoC and VAE is whether there is a stop gradient operation on the posterior $q(z|\tau)$ when minimizing the KL-divergence between $q(z|\tau)$ and the prior $p(z|s_0)$. We conduct the ablation below in Figure A.31 where we also apply stop gradient to VAE, and observe that VAE's performance drops significantly.

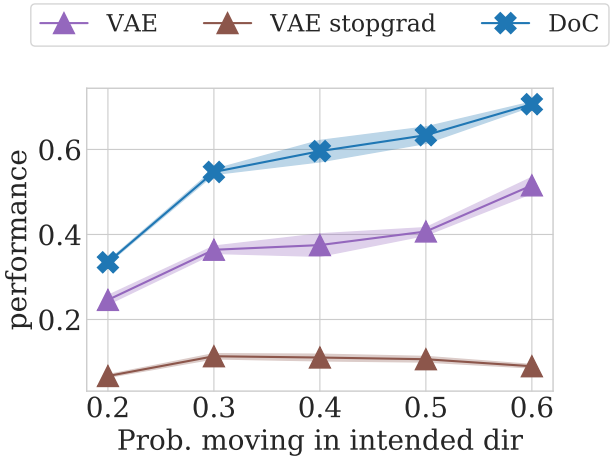


Figure A.31: Average performance (across 5 seeds) of DoC and baselines on FrozenLake with different levels of stochasticity (p) and offline dataset quality (ϵ). DoC outperforms RvS and future VAE with bigger gains the offline data is less optimal.

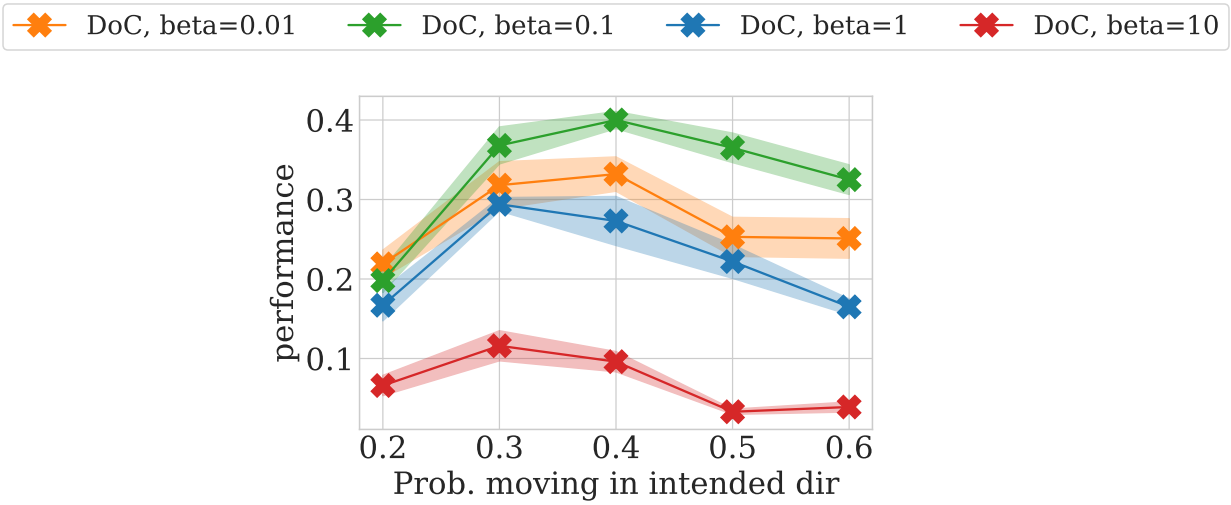


Figure A.32: Average performance (across 5 seeds) of DoC with different regularization strength (β). The effect of β is more pronounced when the dataset is highly optimal (e.g., ϵ random in $D = 0.7$), for which we found a smaller β (e.g., 0.1) to generally perform better.

DoC with different regularization strength (β)

DoC with different number of future samples (K)

A.5 Appendix for Learning Universal Policies

A.5.1 Architecture, Training, and Evaluation Details

Video Diffusion Training Details We use the same base architecture and training setup as [28] which utilizes Video U-Net architecture with 3 residual blocks of 512 base channels and channel multiplier [1, 2, 4], attention resolutions [6, 12, 24], attention head dimension 64, and conditioning embedding dimension 1024. We use noise schedule log SNR with range [-20, 20]. We make modifications Video U-Net to support first-frame conditioning during training. Specifically, we replicate the first frame to be conditioned on at all future frame indices, and apply temporal super resolution model condition on the replicated first frame by concatenating the first frame channel-wise to the noisy data similar to [463]. We use temporal convolutions as opposed to temporal attention to mix frames across time, to maintain local temporal consistency across time, which has also been previously noted in [28]. We train each of our video diffusion models for 2M steps using batch size 2048 with learning rate 1e-4 and 10k linear warmup steps. We use 256 TPU-v4 chips for our first-frame conditioned generation model and temporal super resolution model.

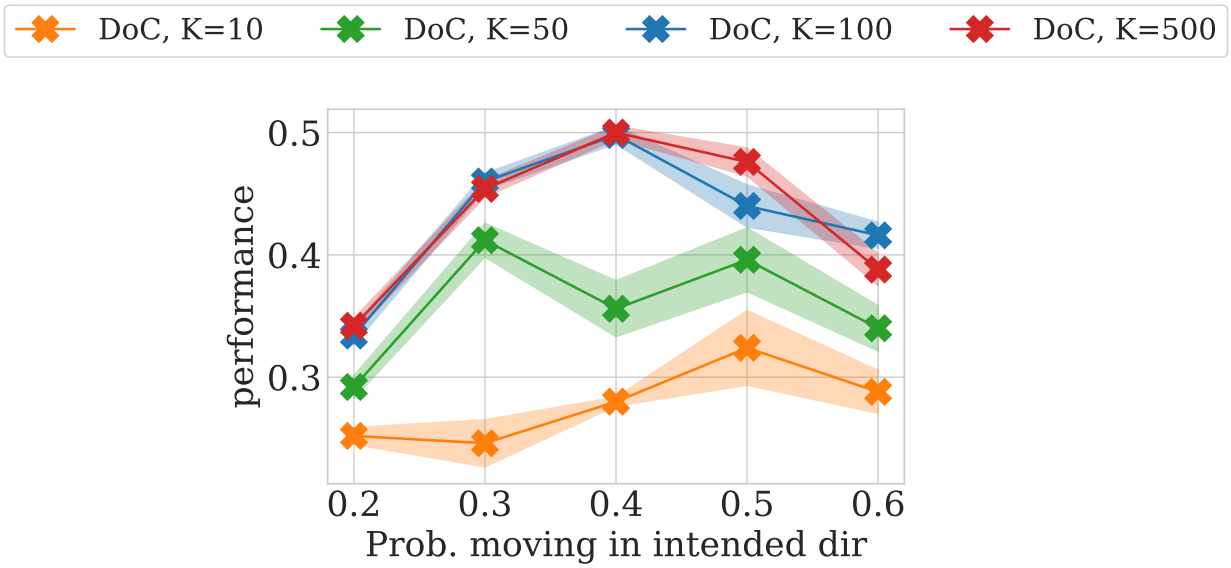


Figure A.33: Average performance (across 5 seeds) of DoC with different number of samples during inference (K). We found that higher number of samples leads to better performance as we expect, and the gain beyound 100 samples is negligible.

We use T5-XXL [385] to process input prompts which consists of 4.6 billion parameters. For combinatorial and multi-task generalization experiments on simulated robotic manipulation, we train a first-frame conditioned video diffusion models on 10x48x64 videos (skipping every 8 frames) with 1.7B parameters and a temporal super resolution of 20x48x64 (skipping every 4 frames) with 1.7B parameters. The resolution of the videos are chosen so that the objects being manipulated (e.g., blocks being moved around) are clearly visible in the video. For the real world video results, we finetune the 16x40x24 (1.7B), 32x40x24 (1.7B), 32x80x48 (1.4B), and 32x320x192 (1.2B) temporal super resolution models pretrained on the data used by [28].

Inverse Dynamics Training Details UniPi's inverse dynamics model is trained to directly predict the 7-dimensional controls of the simulated robot arm from an image observation mean squared error. The inverse dynamics model consists of a 3x3 convolutional layer, 3 layers of 3x3 convolutions with residual connection, a mean-pooling layer across all pixel locations, and an MLP layer of (128, 7) channels to predict the final controls. The inverse dynamics model is trained using the Adam optimizer with gradient norm clipped at 1 and learning rate 1e-4 for a total of 2M steps where linear warmup is applied to the first 10k steps.

Baselines Training Details We describe the architecture details of various baselines below. The training details (e.g., learning rate, warm up, gradient clip) of each baseline follow those of the inverse dynamics model detailed above.

Transformer BC [49, 3]. We employ the same transformer architecture as the 10M model of [3] with 4 attention layers of 8 heads each and hidden size 512. We apply 4 layers of 3x3 convolution with residual connection to extract image features, which, together with T5 text embeddings, are used as inputs to the transformer. We additionally experimented with vision transformer style linearization of the image patches similar to [3], but found the performance to be similar. We use a context length of 4 and skip every 4 frames similar to UniPi's inverse dynamics. We tried increasing the context length of the transformer to 8 but it did not help improve performance.

Transformer TT [5]. We use a similar transformer architecture as the Transformer BC baseline detailed above. Instead of predicting the immediate next control in the sequence as in Transformer BC, we predict the next 8 controls (skipping every 4 controls similar to other baselines) at the output layer. We have also tried autoregressively predicting the next 8 controls, but found the errors to accumulate quickly without additional discretization.

State-Based Diffusion [133]. For the state-based diffusion baseline, we use a similar architecture as UniPi's first-frame conditioned video diffusion, where instead of diffusing and generating future image frames, we replicate future controls across different pixel locations and apply the same U-Net structure as UniPi to learn state-based diffusion models.

Details of the Combinatorial Planning Task In the combinatorial planning tasks, we sample random 6 DOF poses for blocks, colored bowls, the final placement box. Blocks start off uncolored (white) and must be placed in a bowl to obtain a color. The robot then must manipulate and move the colored block to have the desired geometric relation in the placement box. The underlying action space of the agent corresponds to 6 joint values of robot plus a discrete contact action. When the contact action is active, the nearest block on the table is attached to the robot gripper (where for methods that predict continuous actions, we thresholded action prediction > 0.5 to correspond to contact). Given individual action predictions for different models, we simulate the next state of the environment by running the joint controller in Pybullet to try reach the predicted joint state (with a timeout of 2 seconds due to certain actions being physically infeasible). As only a subset of the video dataset contained action annotations, we trained the inverse-dynamics model on action annotations from 20k generated videos.

Details of the CLIPort Multi-Environment Task In the CLIPort environment, we use the same action space as the combinatorial planning tasks and execute actions similarly using the built in joint controller in Pybullet. As our training data, we use a scripted agent on

`put-block-in-bowl-unseen-colors`, `packing-unseen-google-objects-seq`, `assembling-kits-seq-un`
`stack-block-pyramid-seq-seen-colors`, `tower-of-hanoi-seq-seen-colors`, `assembling-kits-seq-s`
`tower-of-hanoi-seq-unseen-colors`, `stack-block-pyramid-seq-unseen-colors`, `packing-seen-goog`
`packing-boxes-pairs-seen-colors`, `packing-seen-google-objects-group`. As our test data, we used the environments `put-block-in-bowl-seen-colors`, `packing-unseen-google-objects-gr`
`packing-boxes-pairs-unseen-colors`. We trained the inverse dynamics on action annotation across the 200k generated videos.

A.5.2 Additional Results

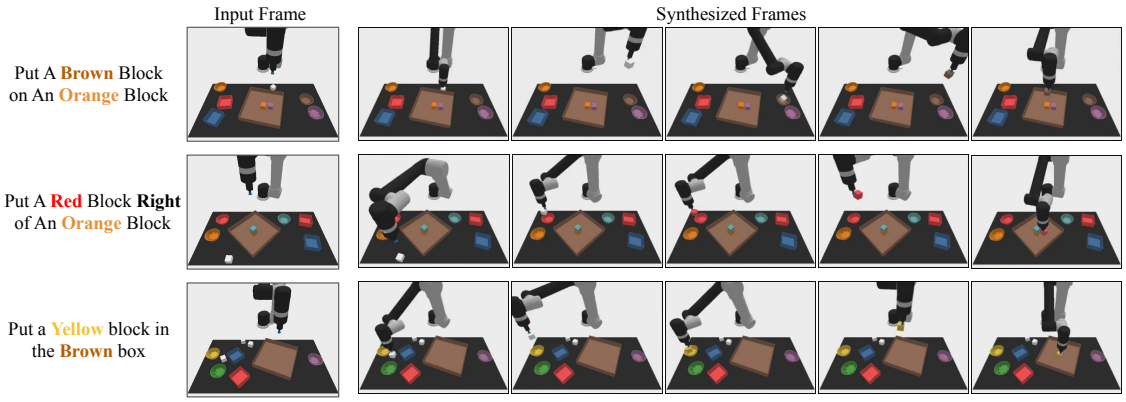


Figure A.34: **Combinatorial Video Generation.** Additional results on UniPi's generated videos for unseen language goals at test time.

Additional Results on Combinatorial Generalization

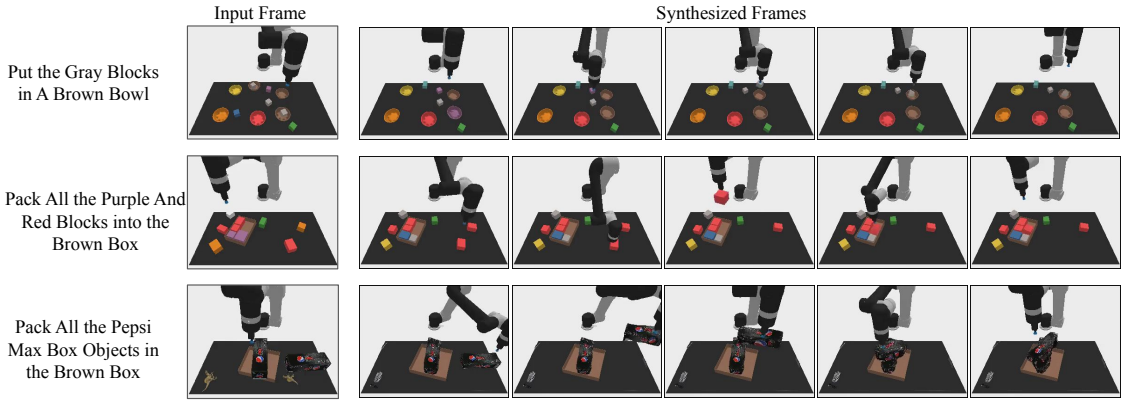


Figure A.35: **Multitask Video Generation.** Additional results on UniPi's generated video plans on different new tasks in the multitask setting.

Additional Results on Multi-Environment Transfer

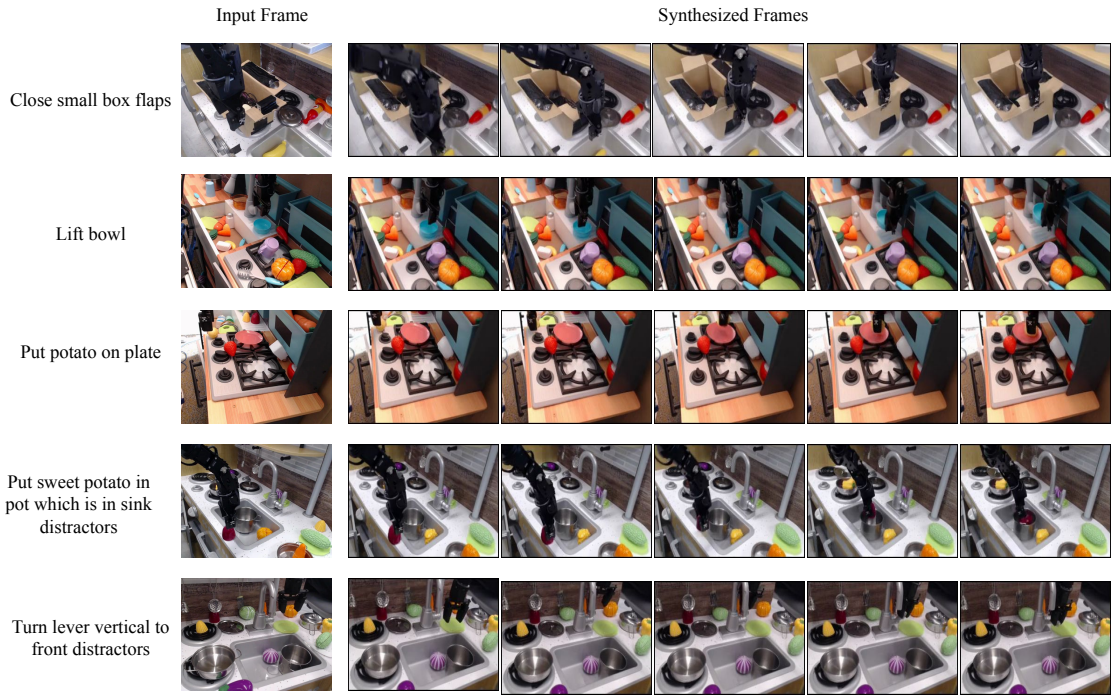


Figure A.36: **High Fidelity Plan Generation.** Additional results on UniPi's high resolution video plans across different language prompts.

Additional Results on Real-World Transfer

A.6 Appendix for Learning Universal Simulators

In this Appendix we provide additional qualitative results on long-horizon simulation of human and robot interactions (Section A.6.1), long-horizon VLM policies (Section A.6.1), and low-level RL policies (Section A.6.1) that work on real robot. We also provided details on the dataset used to train UniSim in Section A.6.2, the model architecture and training details of UniSim in Section A.6.3, and the details of the three experimental setups for applications of UniSim in Section A.6.4. Finally, we provide failed examples when UniSim is not jointly trained on broad datasets (Section A.6.6). Video demos can be found at anonymous-papers-submissions.github.io

A.6.1 Additional Results

Additional Long-Horizon Interaction

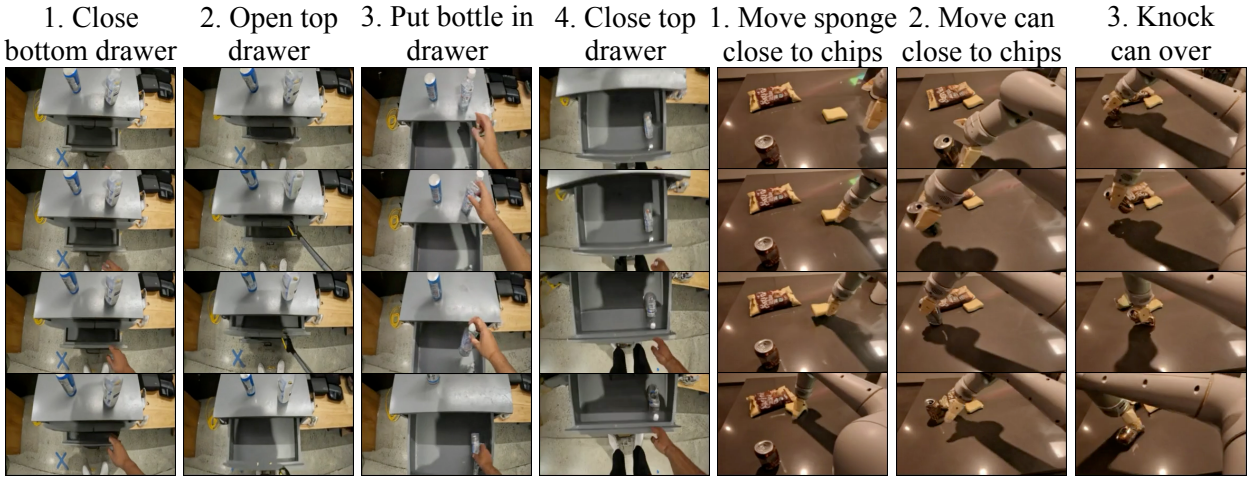


Figure A.37: Additional results on long-horizon interaction with humans and robots. UniSim can generate consistent video rollouts across 3-4 high-level language actions.

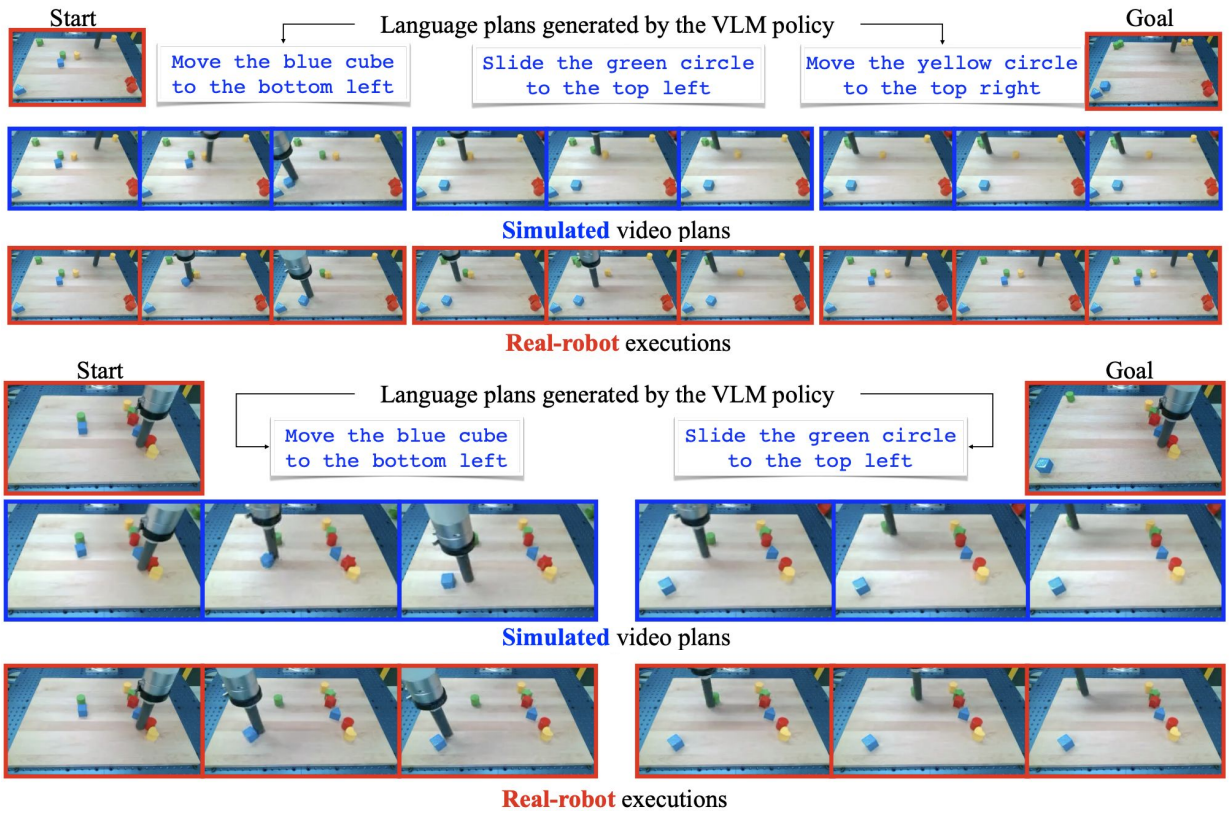
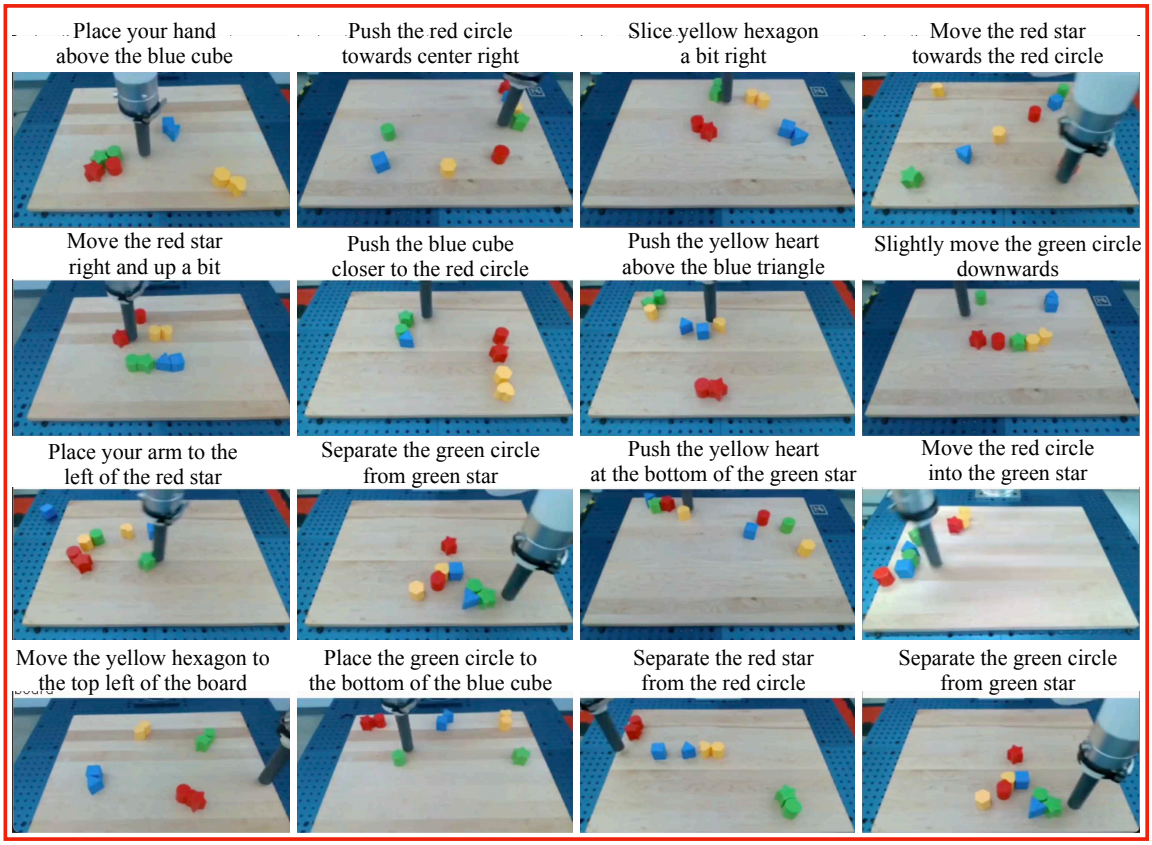


Figure A.38: Additional results on applying UniSim to train vision-language policies to complete long-horizon tasks. VLM finetuned with hindsight labeled data is able to generate long-horizon instructions that moves two or three blocks successfully to match their location in the goal image.

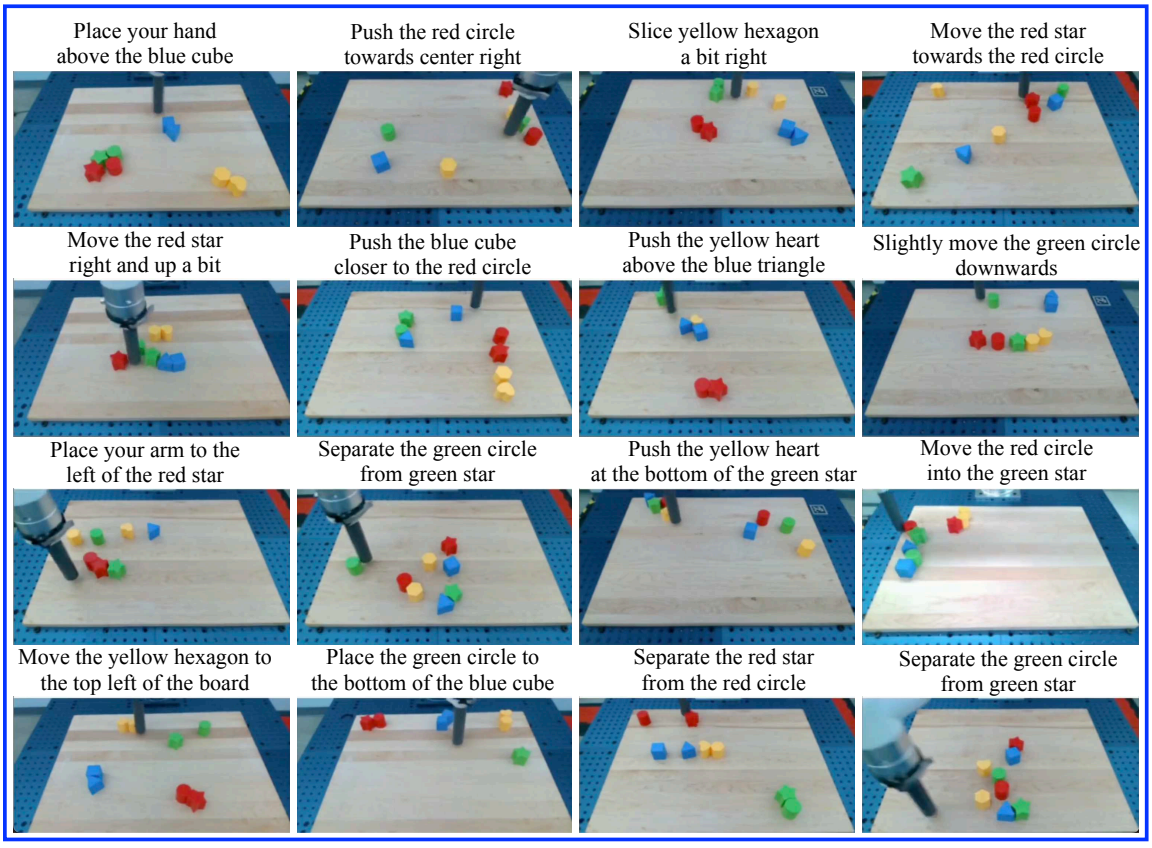
Additional Real-Robot Results for Long-Horizon Language Policy

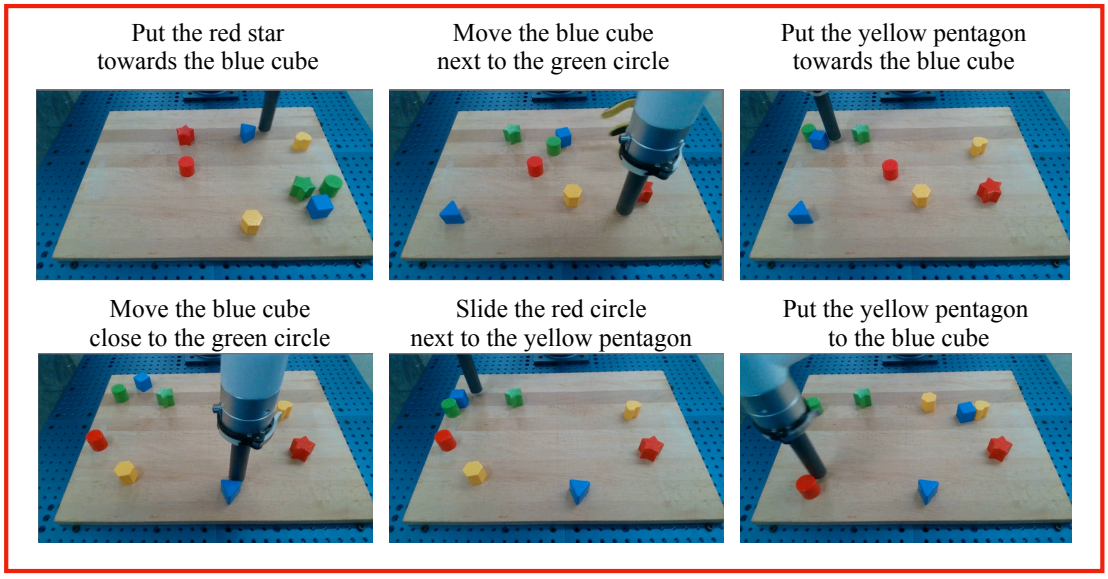
Additional Results on Learning RL Policy in UniSim



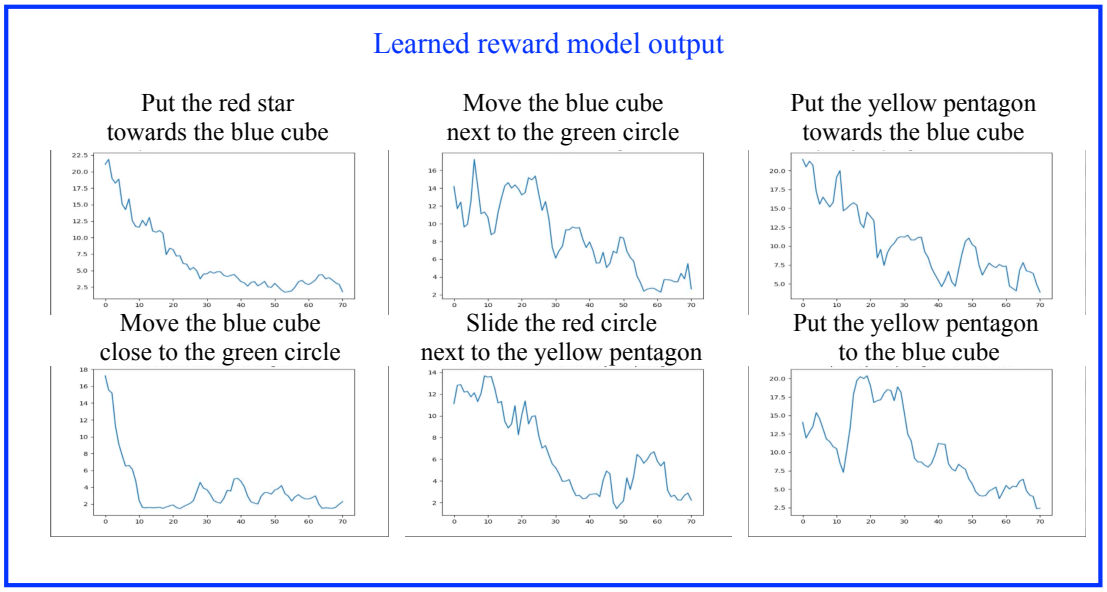
Real first observation of each trajectory

Simulated last observation of each trajectory

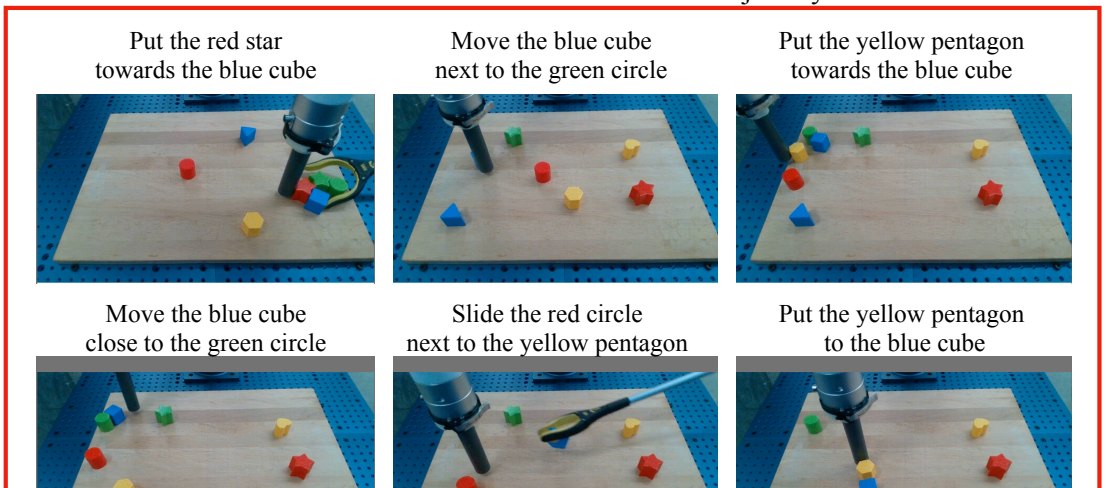




Real first observation of each trajectory



Real-robot last observation of each trajectory



A.6.2 Datasets

We provide the datasets used to train UniSim below, including dataset name, number of training examples (approximate), and weight in the data mixture. Miscellaneous data are collections of datasets that have not been published. Some of these datasets have been processed into train and validation split, hence the number of training examples may differ from the original data size. When text are available in the original dataset, we use T5 language model embeddings [385] to preprocess the text into continuous representations. When low-level controls are available in the original dataset, we encode them both as text and normalize then discretize them into 4096 bins contatenated with language embeddings (if present). The choice of mixture weights are either 0.1 or 0.05 without careful tuning. How data mixture weights affect simulation performance is an interesting line of future work.

	Dataset	**# Examples**	**Weight**
Simulation	Habitat HM3D [18]	710	0.1
	Language Table sim [156]	160k	0.05
Real Robot	Bridge Data [387]	2k	0.05
	RT-1 data [19]	70k	0.1
	Language Table real [156]	440k	0.05
	Miscellaneous robot videos	133k	0.05
Human activities	Ego4D [402]	3.5M	0.1
	Something-Something V2 [411]	160k	0.1
	EPIC-KITCHENS [410]	25k	0.1
	Miscellaneous human videos	50k	0.05
Panorama scan	Matterport Room-to-Room scans [504]	3.5M	0.1
Internet text-image	LAION-400M [413]	400M	0.05
	ALIGN [505]	400M	0.05
Internet video	Miscellaneous videos	13M	0.05

Table A.4: Dataset name, number of training examples, and mixture weights used for training UniSim.

A.6.3 Architecture and Training

We the 3D U-Net architecture to parametrize UniSim video model. We apply the spatial downsampling pass followed by the spatial upsampling pass with skip connections to the downsampling pass activations with interleaved 3D convolution and attention layers as in the standard 3D U-Net. The video models in UniSim consist of one history conditioned video prediction model as the base and two additional spatial super-resolution models similar to [28]. The history conditioned base model operates at temporal and spatial resolution $[16, 24, 40]$, and the two spatial super-resolution models operate at spatial resolution $[24, 40] \rightarrow [48, 80]$ and $[48, 80] \rightarrow [192, 320]$, respectively. To condition the base video model on the history, we take 4 frames from the previous video segment and concatenate them channelwise to the noise samples inputted to the U-Net. We employ temporal attention for the forward model to allow maximum modeling flexibility but temporal convolution to the super-resolution models for efficiency reasons similar to [28]. The model and training hyperparamters of UniSim are summarized in Table A.5.

Hyperparameter	**Value**
Base channels	1024
Optimizer	Adam ($\beta_1 = 0.9, \beta_2 = 0.99$)
Channel multipliers	1, 2, 4
Learning rate	0.0001
Blocks per resolution	3
Batch size	256
Attention resolutions	6, 12, 24
Num attention heads	16, 16, 8
Conditioning embedding dimension	4096
Conditioning embedding MLP layers: 4	
Conditioning token length	64
EMA	0.9999
Dropout	0.1
Training hardware	512 TPU-v3 chips
Training steps	1000000
Diffusion noise schedule	cosine
Noise schedule log SNR range	[-20, 20]
Sampling timesteps	256
Sampling log-variance interpolation	$\gamma = 0.1$
Weight decay	0.0
Prediction target	ϵ

Table A.5: Hyperparameters for training UniSim diffusion model.

A.6.4 Details of Experimental Setups

Details of Learning Long-Horizon Policy **Language Table Dataset and environment.** The Language Table [156] dataset consists of 160k simulated trajectories and 440k real trajectories where each trajectory contains a language instruction (e.g., "move blue cube to the right"), a sequence of visuomotor controls, and a sequence of image frames corresponding to the execution of the task. The original trajectories have short horizons (e.g., only moving one block).

PALM-E VLM Policy. We modify the original PALM-E 12B model [414] to condition on a goal image as additional input before decoding the text actions. The VLM is finetuned on either the original short horizon data or the long horizon simulated data using 64 TPUv3 chips for 1 day. The supervision for short-horizon baseline is the single step language instruction in the original data, whereas the supervision for long-horizon UniSim data is the scripted long-horizon language instructions chained together that generated the video data. Other model architecture and training details follow [414].

Simulated evaluation. In setting up goal in the simulated environments, a subset of 3-4 blocks (randomly selected) are moved by 0.05, 0.1, or 0.2 along the x,y axes (randomly selected). The original observation space has $x \in [0.15, 0.6]$ and $y \in [-0.3048, 0.3048]$. So the modification of goal location corresponds to meaningful block movements. For executing the long-horizon VLM policy trained on UniSim data, we first sample one language instruction from the VLM, predict a video of 16 frames, and use a separately trained inverse dynamics model similar to [427] to recover the low-level control actions, which we found to slightly outperform directly regressing on control actions from language outputs of the VLM. We execute 5 instructions in total, and measure the final distance to goal according to the ground truth simulator state. We 5 evaluations each with a different random seed for sampling the initial state and resetting the goal, and report the mean and standard error.

Details of RL Policy Training **Stage 1 (Supervised Learning) Model Architecture** The PaLI 3B model trained on Language-Table uses a Vision Transformer architecture G/14 [398] to process images, and the encoder-decoder architecture of UL2 language model for encoding task descriptions and decoding tokens which can represent language, control actions, or other values of interest (described below). **Objectives** In the first stage of training, using a dataset of demonstrations, we finetune the pretrained PaLI 3B vision language model checkpoint [148] with the following tasks:

- **Behavioral Cloning:** Given observations and task instruction, predict the demonstration action. The continuous actions of the Language-Table domain are discretized into the form "+1 -5", and represented using extra tokens from the PaLI model's token vocabulary. As an example, "+1 -5" is represented by the token sequence `(<extra_id_65>, <extra_id_1>, <extra_id_66>, <extra_id_5>)`.

- **Timestep to Success Prediction:** Given observations and task instruction, predict how many timesteps are left until the end of episode (i.e. success). Similar to actions,

the number of steps remaining is represented via extra tokens from the PaLI model's token vocabulary.

- **Instruction Prediction:** Given the first and last frame of an episode, predict the task instruction associated with that episode.

We use learning rate 0.001, dropout rate 0.1, and batch size 128 to finetune the PaLI 3B model for 300k gradient steps with 1k warmup steps on both the simulated and real Language Table dataset similar to RT-2 [415].

Stage 2 (RL Training) Reward Definition As mentioned above, during Stage 1, given an observation and goal, the PaLI model is finetuned to predict how many timesteps are left until the demonstration episode reaches a success state. Let us denote this function by $d(o, g)$. The reward we use during RL training is defined as $r(o_t, a_t, o_{t+1}, g) = -[d(o_{t+1}, g) - d(o_t, g)] \cdot \mathcal{C}$, where $\mathcal{C} > 0$ is a small constant used to stabilize training ($\mathcal{C} = 5e - 2$ in this work). Intuitively, this reward tracks if from timestep t to $t+1$ the policy arrived closer to accomplishing the desired goal. Before starting Stage 2, we make a copy of the Stage 1 model checkpoint and keep it frozen to use as the reward model for RL training. **Environment Definition** To implement video generation as environment transitions, we expose the inference interface of the video generation model through remote procedure call, and use the DeepMind RL Environment API (also known as DM Env API) [11] to wrap the remote procedure call in the step function of the environment. When the environment is reset to start a new episode, a goal instruction is randomly sampled from the ones available in the dataset of demonstrations used in Stage 1. **RL Method** We initialize the RL trained policy using the Stage 1 checkpoint, which as mentioned was also trained with a Behavioral Cloning objective. A collection of actor processes perform policy rollouts in the video generation environment, and add rewards to the trajectories using the reward model defined above. The policy is updated using the REINFORCE [63] objective, i.e. $\nabla_\pi \mathcal{L}(o_t, a_t, g) = \nabla_\pi \log \pi(a_t | o_t, g) \cdot \left[\sum_{i=t}^{T} \gamma^{i-t} \cdot r(o_i, a_i, o_{i+1}, g) \right]$, where $\mathcal{L}(o_t, a_t, g)$ represents the loss associated with the observation-action pair (o_t, a_t) in an episode with the goal g. The actors are rate limited to prevent generated trajectories from being very off-policy. We report the hyperparameters associated with RL training in Table A.6.

Hyperparameter	**Value**
Max steps per episode	100
Number of actor processes	64
Number of image history stack	2
Learner batch size	64
Discounting factor γ	0.9

Table A.6: Hyperparameters for training the VLA RL policy using the ACME framework.

Details of Video Captioning Note that even though UniSim is a video based simulator trained to condition on past history, we can achieve text-only conditioning by inputting placeholder frames such as white images while increasing the classifier-free guidance strength on text. We found this to work well in generating videos purely from captions of ActivityNet Captions. For generating data to train VLMs, we take the training split of ActivityNet Captions which consists of 30,740 text-video examples after the 50/25/25% train/val1/val2 split as in [419]. For each of the 30,740 text, we generate 4 videos from UniSim, and use the text labels as supervision in finetuning PaLI-X. As a result, we have 4X amount of the original training data (in terms the number of videos). In addition, we found the generated videos to generally align better semantically than the original ActivityNet Captions videos, which could contain noise and ambiguous videos that could be labeled differently. We use ground truth temporal proposals at evaluation following [419] and [400]. Following [419], we use the val1 split for validation and val2 split for testing.

A.6.5 Additional Ablations

Ablations of Datasets We conduct ablations on dataset used in UniSim by computing the FVD and CLIP scores over 1024 samples from the test split. We observe that including internet data and various activity and robot data performs the best. Removing the internet data led to significantly worse FVD, highlighting the importance of using internet data in UniSim.

Dataset	FVD ↓	CLIP ↑
Internet only	219.62	22.27
Without internet	307.80	21.99
Universal simulator	**211.30**	**22.63**

Table A.7: **Ablations of datasets** using FVD and CLIP score on the held-out test split. Including internet data and diverse human activity and robot data in UniSim achieves the best FVD and CLIP scores.

Ablations of Model Size We conduct ablations on model size by computing the FVD and CLIP scores over 1024 samples from the test split. We found that while increasing the model size improves the video modeling performance, the amount of improvement measured by FVD plateaus as the model gets bigger, which is slightly disappointing from a scaling point of view.

Model size	FVD ↓	CLIP ↑
500M	277.85	22.08
1.6B	224.61	22.27
5.6B	**211.30**	**22.63**

Table A.8: **Ablations of model size** using FVD and CLIP score on the held-out test split. The largest model achieves the best FVD and CLIP scores.

A.6.6 Failed Simulations without Joint Training

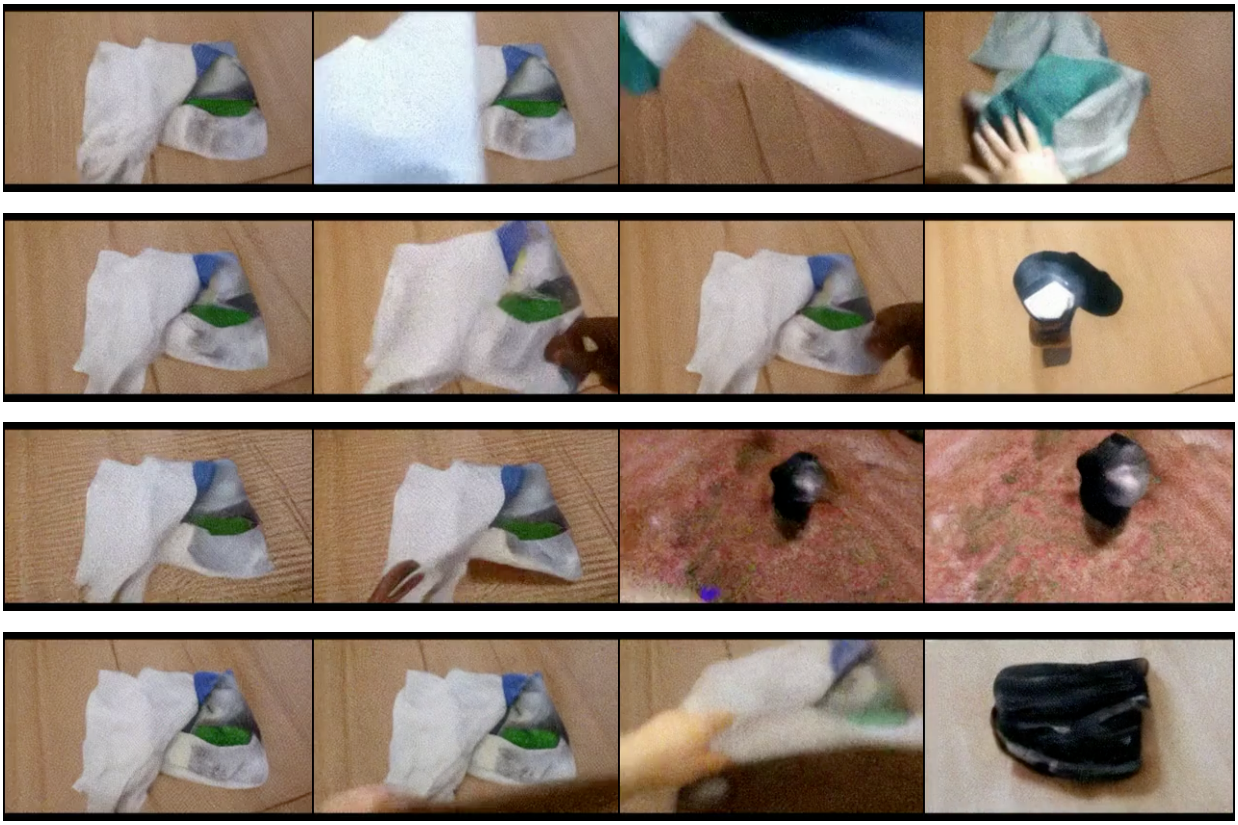


Figure A.41: Failed environment simulation from the action "uncover bottle" without training on broad data as in UniSim. Top two videos are generated from only training on SSV2. Bottom two videos are generated from only training on generic internet data (without SSV2, EpicKitchen, Ego4D, and various robotics dataset).

A.7 Appendix for Efficient Adaptation of Video Generation

In the section of the appendix Appendix we provided a detail derivation of connection between diffusion models and EBMs in Section A.7.1. We further provide additional experimental details in Section A.7.2. Finally, we provide a comparison with using the same computational budget to finetune the existing large pretrained model in Section A.7.3.

A.7.1 Connection between Diffusion and EBM

The sampling procedure in a diffusion model corresponds to the Langevin sampling procedure on an EBM. To see this, we consider perturbing a sample $\tau^{t-1} \sim p\left(\tau^{t-1}\right)$ from target distribution $p(\tau^{t-1})$ with a Gaussian noise, *i.e.*,

$$\tau^t = \tau^{t-1} + \xi, \quad \xi \sim \mathcal{N}\left(\mathbf{0}, \sigma_t^2 \mathbf{I}\right)$$

which corresponds to the transition operator

$$\mathcal{T}(\tau^t|\tau^{t-1}) \propto \exp\left(\frac{-\left\|\tau^t - \tau^{t-1}\right\|^2}{2\sigma_t^2}\right)$$

where the joint distribution of τ^t and τ^{t-1} is

$$p(\tau^t, \tau^{t-1}) \propto \exp\left(\psi\left(\tau^{t-1}\right) - \frac{\left\|\tau^t - \tau^{t-1}\right\|^2}{2\sigma_t^2}\right).$$

We can express the Bayes estimator of τ^{t-1} given the perturbed observation τ^t as

$$m(\tau^t) = \int \tau^{t-1} p_\theta(\tau^{t-1}|\tau^t) d\tau^{t-1} = \tau^t + \sigma_t^2 \nabla \log p\left(\tau^t\right) \tag{A.160}$$

Proof. By the property of Gaussian distribution, we have

$$\sigma^2 \nabla_{x'} p\left(x'|x\right) = p\left(x'|x\right)\left(x - x'\right). \tag{A.161}$$

Therefore, we have

$$\sigma \nabla_{x'} \int p\left(x'|x\right) p(x) dx = \int \left(x - x'\right) p\left(x', x\right) dx = \int x p\left(x', x\right) dx - x' p(x') \tag{A.162}$$

$$\Rightarrow \quad \sigma \nabla_{x'} \log p\left(x'\right) = \int x \frac{p\left(x', x\right)}{p\left(x'\right)} dx - x' = \mathbb{E}\left[X|x'\right] - x' \tag{A.163}$$

□

Thus, we can represent the perturbed data with an EBM $p(\tau^t) \propto \exp\left(E_\theta\left(\tau^t, \sigma_t\right)\right)$, and learn the parameters through regression [455, 506, 507, 456], which leads to the optimal solution

$$\begin{aligned} &\min_\theta \ \mathbb{E}_{\tau^{t-1} \sim \mathcal{D}, \xi \sim \mathcal{N}\left(0, \sigma_t^2 \mathbf{I}\right)}\left[\left\|\tau^{t-1} - m(\tau^t)\right\|^2\right] \\ &= \mathbb{E}_{\tau^{t-1} \sim \mathcal{D}, \xi \sim \mathcal{N}\left(0, \sigma^2 \mathbf{I}\right)}\left[\left\|-\xi - \nabla E_\theta\left(\tau^{t-1} + \xi, \sigma_t\right)\right)\right\|^2\right], \end{aligned} \tag{A.164}$$

whch also corresponds to the denoising diffusion training objective.

Once we have the trained $E_\theta\left(\tau^t\right)$, we can then recover the sample τ^{t-1} according the denoising sampling procedure

$$\tau^{t-1} = \alpha^t m(\tau^t) + \alpha^t \xi = \alpha^t(\tau^t - \gamma \nabla_{\tau^t} E_\theta\left(\tau^t, \sigma_t\right)) + \alpha^t \xi, \quad \xi \sim \mathcal{N}\left(\mathbf{0}, \sigma_t^2 \mathbf{I}\right) \tag{A.165}$$

which corresponds to the sampling via stochastic localization [508] and Equation 6.5 in the main paper.

A.7.2 Experimental Details

Experiment Details

Dataset The large pretrained model is trained on 14 million video-text pairs plus 60 million image-text pairs, and with the LAION-400M image-text dataset. The images are spatially resized to 24x40 and videos using anti-aliased bi-linear resizing. We use different frame rate for different types of videos for best visualization results. For the Bridge [387] we directly use the released opensource dataset. For Ego4D [402] data, we take a small portion of the released dataset. For Anime and Sci-Fi style, we curate two separates datasets with their respective keywords. The keywords used for filtering data for Anime style are (in small letter) "disney", "cartoon", "anime", "animation", "comic", "pixar", "animated", "fantasy". The keywords used for filtering data for Sci-Fi style are "science fiction", "sci-fi", "scifi", "astronaut", "alien", "NASA", "interstellar". For the animation with a particular artist style, we use the Case Closed animation (also named Detective Conan). For the Language Table dataset, we used the data from [459].

Dataset	Pretrain	Bridge	Ego4D	Anime	Sci-Fi	Case Closed	LangTable Sim	LangTable Real
# Train	474M	2.3k	97k	0.6M	21k	5k	0.16M	0.16M

Table A.9: **Training data size.** Number of text-video or text-image pairs used for training the pretrained large model and each of the small model. Training data for particular styles can be magnitude smaller than the pretraining dataset.

Architecture. To pretrain the large model, we use the same pretraining dataset, base architecture, and training setup as [28], with modifications of first-frame conditioning for Bridge and Ego4D, and edge conditioning for stylisation and sim-to-real. Specifically, the large model architecture consists of video U-Net with 3 residual blocks of 1024 base channels and channel multiplier [1, 2, 4], attention resolutions [6, 12, 24], attention head dimension 64, and conditioning embedding dimension 1024. To support first frame conditioning, we replicate the first frame across all future frame indices, and concatenate the replicated first frame channel-wise to the noisy data following [427]. To support edge conditioning, we run a sobel edge detector and use gradient approximations in the x-direction as the conditional video, and concatenates these edge frames with noisy data similar to first-frame conditioning. The large model consists of 5.6 billion parameters in total. For the set of small models for adaptation, Ego4D Small (L) has 512 base channels in each of the residual blocks. Ego4D Small (S) and Bridge Small (S) have a single residual block with 32 base channels. Bridge Small (L) has a single residual block with 64 base channels. The set of stylisation models (animation, sci-fi, and particular anime style) have 3 residual blocks and 256 base channels. For illustrating the generated videos at a higher resolution, we train two additional

spatial super resolution models 24x40 → 48x80 (1.4B) and 48x80 → 192x320 (1.2B). We additionally use T5-XXL [385] to process input text prompts which consists of 4.6 billion parameters, which we omit from the parameter count as all large and small models require text embeddings.

Training and Evaluation. We train each of our video diffusion models for 2M steps using batch size 2048 with learning rate 1e-4 and 10k linear warmup steps. The large 5.6B pretrained model requires 512 TPU-v4 chips, whereas various small models require anywhere between 8 and 256 TPU-v4 chips depending on the size. We use noise schedule log SNR with range [-20, 20]. We use 128 samples and 1024 samples to compute the FVD, FID, and Inception Scores metric on Bridge and Ego4dD, respectively.

Sampling. All diffusion models are trained with 1000 timesteps of sampling. To generate videos, we combined scores from both pretrained models and adapter models for all timesteps except the last 100 timesteps. The last 100 timesteps capture high frequency information in an image, and we found better image quality if we did not combine scores in these timesteps. For simplicity, we use a pretrained neural strength of 0.2 for Ego4D and 0.1 for Bridge, and 0.4 for all animation datasets, but found additional gains when using large neural strength at earlier timesteps and smaller ones later.

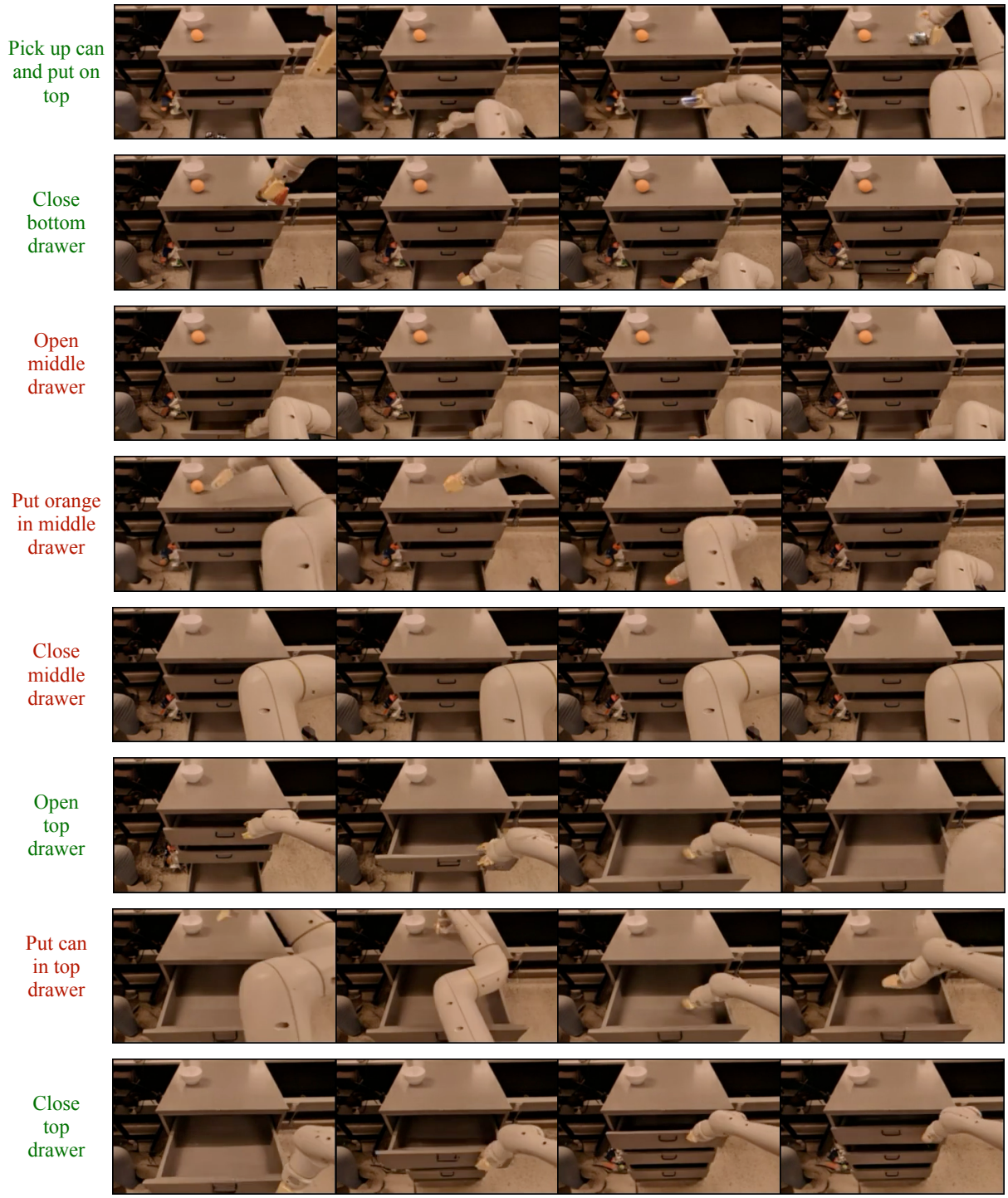


Figure A.42: When the text-to-video model behind UniSim is only trained on data from [19] as opposed incorporating broad data from the internet and other manipulation datasets, long-horizon interaction simulations fail half of the time (red text).

A.7.3 Comparison to Finetuning

To illustrate the computational efficiency of Video Adapter, we further compare video modeling metrics of Video Adapter to finetuning the pretrained model for an equivalent number of TPU time. Specifically, the pretrained model requires 512 TPU-v4 chips whereas the small model on Bridge data requires 8 TPU-v4 chips. The small Bridge model requires 100k steps to reach convergence, and hence we finetune the pretrained model for 100,000 / 64 = 1,560 steps. Video Adapter achieves better FVD and FID than finetuning the pretrained model for an equal number of TPU steps as shown in Table 6.9.

A.7.4 Composing Multiple Factors

Given a set of N separate datasets $\{D_i\}_{i=1:N}$ specifying a set of N different styles of videos, we can also apply our probabilistic adaptation framework across these N models by learning N separate distributions $\{p_i(\tau|\text{text})\}_{i=1:N}$. We can directly sample from the product distribution

$$\underbrace{p_{\text{product}}(\tau|\text{text})}_{\text{Product Distribution}} \propto \underbrace{p_{\text{pretrained}}(\tau|\text{text})}_{\text{Pretrained Prior}} \underbrace{\prod_{i=1:N} p_i(\tau|\text{text})}_{\text{N Different Video Models}}, \tag{A.166}$$

which now assigns high likelihood to videos that exhibit each of the composed styles.

We can directly sample from this composed model using the modified composite denoising function

$$\tilde{\epsilon}_\theta(\tau, t|\text{text}) = \epsilon_\theta(\tau, t) + \omega \sum_{i=1}^{N} (\epsilon_i(\tau, t|\text{text}) + \lambda \epsilon_{\text{pretrained}}(\tau, t|\text{text}) - \epsilon_i(\tau, t)),$$

which simply corresponds to using a weighted average over the predictions of each of the N models.

A.7.5 Comparison to Parameter Efficient Finetuning

The problem setting of Video Adapter is that pretrained model weights are *not* accessible. This scenario is common in large language models (e.g., GPT-4, Bard), and large text-to-video models are heading in the same direction. Parameter efficient finetuning (e.g., LoRA, null-text inversion, prefix-tuning) requires access to pretrained model weights, whereas Video Adapter does not. Therefore Video Adapter is not comparable to parameter efficient finetuning.

Nevertheless, we conducted comparisons to LoRA and null-text inversion out of curiosity (prefix-tuning is omitted since it has only been applied to language models). For LoRA, we use rank 1 and rank 64 to compare to the smaller and larger task-specific VideoAdapter model. For null-text inversion, we use an unconditional null embedding of size [64, 4096] (the same dimension as the original text embeddings). We report the video modeling metrics in Table A.10.

			Bridge	Ego4D
Model	FVD ↓	FID ↓	FVD ↓	IS ↑
Small (S)	186.8	38.8	228.3	2.28
Small (S) + Pretrained	177.4	37.6	156.3	2.82
Small (L)	152.5	30.1	65.1	3.31
Small (L) + Pretrained	**148.1**	**29.5**	52.5	**3.53**
LoRA-Rank1	170.2	32.2	74.5	3.4
LoRA-Rank64	165.5	31.6	**50.3**	**3.5**
Null-text inversion	288.8	40.2	90.2	3.1

Table A.10: **Comparison of Video Adapter to parameter efficient finetuning** under fixed compute budget. Video Adapter performs close to LoRA-Rank64 on Ego4D and better than parameter efficient finetuning on Bridge.

We observe that LoRA-Rank1 performs slightly better than Video Adapter (small). However, In the LoRA-Rank1 case, LoRA still performs worse than training a small domain specific model. In this case, Video Adapter can simply use the small model without the pretrained prior. In comparison, we found LoRA-Rank64 leads to mixed results when compared to Video Adapter (large), i.e., LoRA outperforms Video Adapter on Ego4D but not on Bridge data. We found that null-text inversion performs the worst, potentially due to limited flexibility of null-embeddings during finetuning.

Our results illustrate that Video Adapter, despite requiring only black-bo adaptation without access to pretrained model weights performs better than Null-text inversion and very comparably to LoRA finetuning (with pretrained model weights).