Generalization via Information Bottleneck in Deep Reinforcement Learning

Xingyu Lu
Stas Tiomkin
Pieter Abbeel

Electrical Engineering and Computer Sciences
University of California at Berkeley

Technical Report No. UCB/EECS-2020-56
http://www2.eecs.berkeley.edu/Pubs/TechRpts/2020/EECS-2020-56.html

May 23, 2020

Acknowledgement

I would like to sincerely thank Professor Pieter Abbeel for the opportunity to work in the Robotic Learning Lab and his guidance during my time as a member of the group. He provided invaluable insights into my research projects, helping me navigate through different ideas and project stages. His advice and guidance enabled me to grow both as a student and as a researcher, for which I am forever grateful.

I would also like to especially thank my research mentor Stas Tiomkin for supporting me during my candidacy. He introduced to me many exciting and interesting research directions, from which I found a passion for information theoretic approaches in reinforcement learning. My research experience here would not have been as warm, exciting, and inspiring without him.

Generalization via Information Bottleneck in Deep Reinforcement Learning

by Xingyu Lu

Research Project

Submitted to the Department of Electrical Engineering and Computer Sciences, University of California at Berkeley, in partial satisfaction of the requirements for the degree of **Master of Science, Plan II**.

Approval for the Report and Comprehensive Examination:

Committee:

[illegible]

Professor Pieter Abbeel
Research Advisor

23 - MAY - 2020

(Date)

* * * * * * *

[illegible]

Professor Sergey Levine
Second Reader

May 23, 2020

(Date)

Generalization via Information Bottleneck in Deep Reinforcement Learning

Abstract

Generalization via Information Bottleneck in Deep Reinforcement Learning

by

Xingyu Lu

Master of Science in Electrical Engineering and Computer Science

University of California, Berkeley

Professor Pieter Abbeel, Chair

Despite the significant progress of deep reinforcement learning (RL) in solving sequential decision making problems, agents trained through RL often over-specialize to training environments and struggle to adapt to new, unseen circumstances. In this work, our primary contribution is to propose an information theoretic regularization objective and an annealing-based optimization method to promote better generalization ability in RL agents.

Training an agent that is resilient to unseen changes in environments helps combat "diversity": a robot trained in an ideal setting may be required to perform in more adversarial circumstances, such as increased obstacles, darker lighting and smoother/rougher surfaces; to prevail in these unseen environments, agent must be capable of adapting to unseen dynamics.

Our work tackles the generalization problem from an information theoretic perspective. We hypothesize that the expressiveness of deep neural networks (DNN) may cause memorization of inputs from training environments, preventing the agent from learning the general dynamics. We address this issue by adding communication constraints between observations and internal representations in the form of an information bottleneck (IB). We suggest an optimization scheme, based on annealing, to obtain a family of solutions parameterized by the regularization weight.

In the first part of the thesis we focus our attention on various maze environments, which have simple dynamics and can be naturally randomized by altering their wall and goal placements. Through experiments in these environments, we study 1) the learning behavior of an agent directly through IB; 2) the benefits of the proposed annealing scheme; 3) the characteristics of the output of the IB; 4) the generalization benefits of a tight IB. In the second part of the thesis we direct our attention to control environments with simulated robotic agents. We demonstrate the generalization benefits of the IB over unseen goals and dynamics, comparing it with other regularization methods and state-of-the-art DRL baselines.

To the best of our knowledge, our work is the first to study the benefits of information bot-

tleneck in DRL for general domain randomization. We tackle the joint-optimization problem by proposing an annealing scheme, and study a variety of domain randomization settings including varying maze layouts, introducing unseen goals, and changing robot dynamics.

Furthermore, this work opens doors for the systematic study of generalization from training conditions to extremely different testing settings, focusing on the established connections between information theory and machine learning, in particular through the lens of state compression and estimation.

Contents

List of Figures

List of Tables

Acknowledgments

I would like to sincerely thank Professor Pieter Abbeel for the opportunity to work in the Robotic Learning Lab and his guidance during my time as a member of the group. He provided invaluable insights into my research projects, helping me navigate through different ideas and project stages. His advice and guidance enabled me to grow both as a student and as a researcher, for which I am forever grateful.

I would also like to especially thank my research mentor Stas Tiomkin for supporting me during my candidacy. He introduced to me many exciting and interesting research directions, from which I found a passion for information theoretic approaches in reinforcement learning. He patiently taught me knowledge in the field and provided me emotional support throughout my candidacy. My research experience here would not have been as warm, exciting, and inspiring without him.

Chapter 1

Introduction

Despite the significant progress of deep reinforcement learning (RL) in solving sequential decision making problems, agents trained through RL often over-specialize to training environments and struggle to adapt to new, unseen circumstances ([16]).

The standard state-of-the-art DRL approaches often use the combination of an encoder, representing either observations or raw states, and a policy, mapping representations to actions ([11], [12], [13]). An encoder is supposed to "understand" the environment by extracting relevant features, while a policy assigns optimal actions based on the state representation produced by the encoder. This DRL cascade is usually optimised end-to-end on training tasks, aiming to provide an optimal encoder and an optimal policy. In a recent study of generalization ([16]), however, it was found that policies trained through end-to-end RL often perform poorly on tasks unseen during training.

We hypothesize that the poor generalization in unseen tasks is due to the DNNs memorizing environment observations, rather than extracting the relevant information for a task. In this work, we suggest adding communication constraints as an information bottleneck on the encoder. Such bottleneck would limit the information flow between observations and representations, thus encouraging the encoder to only extract relevant information from the environment and preventing memorization.

A joint optimisation of encoder and policy with an information bottleneck is a challenging problem, because, in general, the separation principle ([25]) is not applicable. The separation principle allows to estimate state from observation (and under certain conditions to compress observation [21]), and then to derive an policy. In the cases where the separation principle is not applicable, a joint optimisation of encoder and policy can be seen as 'chicken and egg' problem: to derive an optimal policy one needs a meaningful state representation, which in turn depends on the performance of the policy.

Our main contributions are as follows. Firstly, we tackle the problem of poor generalization of DRL to unseen tasks by applying an information bottleneck between observations and state representations (figure 1.1). Specifically, we find a stochastic mapping from observations to internal representations, and regularize such mapping to limit the amount of information flow. Secondly, we propose an annealing scheme for a stable join-optimization

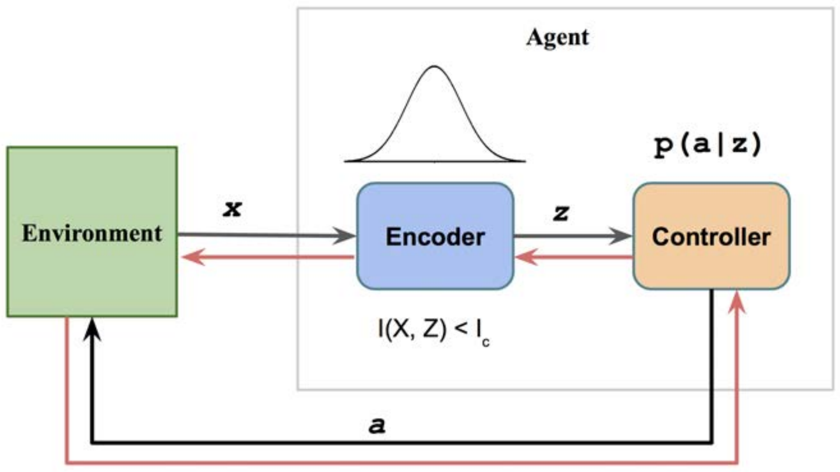


Figure 1.1: Illustration of the proposed scheme. We add stochasticity to the perception component (encoder) of the network, and constrain the information flow through it. The arrows in black indicate the input/output flow of each component, while the arrows in red indicate end-to-end gradients.

of the encoder and policy components of the network, finding a family of solutions parameterized by the weight of the information constraint. Thirdly, we demonstrate that policies trained with an information bottleneck achieve significantly better performance on tasks with unseen layouts, goals and dynamics, as compared to the standard DRL methods. Finally, we demonstrate that our method produces state representations which admit a semantic interpretation, which is in general not guaranteed for end-to-end DRL. Specifically, we demonstrate that the encoder in our approach maps stochastic observations to a space where distances between points are consistent with their values from the optimal critic.

Chapter 2

Related Work

There is a series of previous works that address the problem of control with information bottlenecks. [2] is one of the first works in this direction, where the effects of state compression were studied in the case of linear and known dynamics. Specifically, they showed that in the case of Linear Quadratic Regulator, there exists an optimal compression scheme of state observations. The following works, [22, 23, 21, 24], studied the optimality of compression schemes under different assumptions, although all of them assumed known dynamics, and did not consider information bottleneck for its generalization benefits.

Recently, it was shown that information bottleneck improves generalization in adversarial inverse reinforcement learning ([17]). By placing a bottleneck on the discriminator of a GAN, the author effectively balances the performance of the discriminator and the generator to provide more meaningful gradients. This work, however, focuses strictly on imitation learning, and does not consider any online learning setting involving long-horizon planning.

Another relevant work is the work by Pacelli and Majumdar [15], where information bottleneck is estimated and optimized through separate MINE estimators ([1]) at each time step. While this work also tackles the problem of generalization, it only focuses on image-based environments with changing textures, without considering changing environment goals or dynamics. Additionally, the use of separate MINE estimators at each time step may limit the scalability of the method for long horizon problems. Our work, in contrast, trains a single encoder whose information is regularized without any explicit estimators, and we focus on domain randomization problems with changing environment layouts, goals, and dynamics.

Finally, in the work from Goyal et al. [7], the information bottleneck between actions and goals is studied with an aim to create goal independent policies. While both [7] and our work utilize the variational approximation of the upper bound on the mutual information, their work focuses on finding high information states for more efficient exploration, which is a different objective from our work.

To the best of our knowledge, our work is the first to study the benefits of information bottleneck in DRL for general domain randomization. We tackle the joint-optimization problem by proposing an annealing scheme, and study a variety of domain randomization settings including varying maze layouts, introducing unseen goals, and changing robot dynamics.

Chapter 3

Preliminary

3.1 Markov Decision Process and Reinforcement Learning

This paper assumes a finite-horizon Markov Decision Process (MDP) [18], defined by a tuple $(\mathcal{S}, \mathcal{A}, \mathcal{P}, r, \gamma, T)$. Here, $\mathcal{S} \in \mathbb{R}^d$ denotes the state space (which could either be noisy observations or raw internal states), $\mathcal{A} \in \mathbb{R}^m$ denotes the action space, $\mathcal{P} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to \mathbb{R}^+$ denotes the state transition distribution, $r : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ denotes the reward function, $\gamma \in [0, 1]$ is the discount factor, and finally T is the horizon. At each step t, the action $a_t \in \mathcal{A}$ is sampled from a policy distribution $\pi_\theta(a_t|s_t)$ where $s \in \mathcal{S}$ and θ is the policy parameter. After transiting into the next state by sampling from $p(s_{t+1}|a_t, s_t)$, where $p \in \mathcal{P}$, the agent receives a scalar reward $r(s_t, a_t)$. The agent continues performing actions until it enters a terminal state or t reaches the horizon, by when the agent has completed one episode. We let τ denote the sequence of states that the agent enters in one episode.

With such definition, the goal of RL is to learn a policy $\pi_{\theta^*}(a_t|s_t)$ that maximizes the expected discounted reward $\mathbb{E}_{\pi,P}[R(\tau_{0:T-1})] = \mathbb{E}_{\pi,P}[\sum_0^{T-1} \gamma^t r(s_t, a_t)]$, where expectation is taken on the possible trajectories τ and the starting states x_0. In this paper, we assume model-free learning, meaning the agent does not have access to the environment dynamics $\mathcal{P}$.

3.2 Mutual Information

Mutual information measures the amount of information obtained about one random variable after observing another random variable [5]. Formally, given two random variables X and Y with joint distribution $p(x, y)$ and marginal densities $p(x)$ and $p(y)$, their MI is defined as

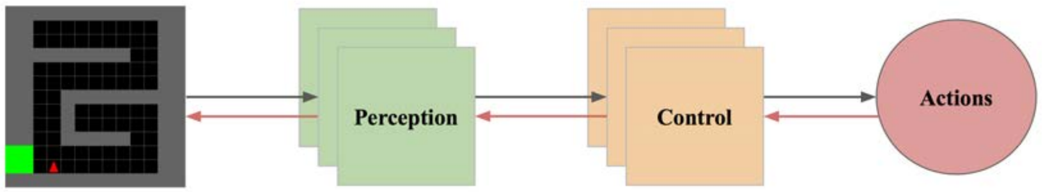


Figure 3.1: Illustration of a typical end-to-end deep reinforcement learning scheme. The arrows in black indicate the input-output flow, while the red arrows indicate the gradient flow.

the KL-divergence between joint density and product of marginal densities:

$$MI(X;Y) = D_{KL}(p(x,y)\|p(x)p(y)) \tag{3.1}$$

$$= \mathbb{E}_{p(x,y)}[\log \frac{p(x,y)}{p(x)p(y)}]. \tag{3.2}$$

Mutual information quantifies reduction in uncertainty of one random variable given another random variable distributed jointly. As a constraint, it limits the dependencies between two random variables in an objective function. In general, this setting is formulated as a constrained optimization problem: finding an optimum of an objective under the constraint of mutual information between relevant random variables.

The rate-distortion function [5] and control under information constraints [22] are the specific cases of this setting, which we study in this work with the aim to improve the generalization of DRL to unseen tasks.

3.3 End-to-End Training

For all baseline experiments, we consider a typical end-to-end setting with deep neural networks, where the agent learns the perception system and planning jointly [14, 8]. This is in contract to the classic separation principle [25], where a state observer and a controller are designed and optimized separately for control. A simple block diagram illustrating end-to-end scheme is presented in Figure 3.1.

The main advantage of end-to-end training is its versatility to understand and solve complex systems, as back-propagating gradients all the way from control policy to perception layers often automatically results in near-optimal state representations. The disadvantage, on the other hand, is often poor generalization: as the network's parameters are optimized for specific tasks, without additional constraints they often end up overfitting onto the training tasks, in forms such as memorization of the environment observation. In this work we study this issues, and suggest a regularization objective as well as an optimization method while maintaining the end-to-end nature of training.

Chapter 4

Method

4.1 Overview

We consider an architecture in which the agent learns with limited information from the environment: instead of learning directly from the environment states $s \in S$, the agent needs to estimate noisy encoding $z \in Z$ of the state, whose information is limited by a bottleneck.

Formally, we decompose the agent policy π_θ into an encoder f_{θ_1} and a decoder g_{θ_2} (action policy), where $\theta = \{\theta_1, \theta_2\}$. The encoder maps environment states into stochastic embedding, and the decoder outputs agent actions $a \in A$:

$$p_{\pi_\theta}(a|s) = \int_z p_{g_{\theta_2}}(a|z) p_{f_{\theta_1}}(z|s) dz \tag{4.1}$$

With such setup, we maximize the RL objective with a constraint on the mutual information between the environment states and the embedding:

$$J(\theta) = \max_\theta \quad \mathbb{E}_{\pi_\theta,\tau}[R(\tau)] \tag{4.2}$$
$$s.t. \quad I(Z,S) \leq I_c \tag{4.3}$$

This objective forces the agent to perform meaningful estimation of the states. As it learns to "forget" parts of the environment states that are not the most crucial for learning, the agent picks up more general skills that are useful for transfer across similar tasks.

4.2 Variational approximation of Information

To estimate mutual information between S, and Z, We makes use of the following identity:

$$I(Z,S) = D_{\mathrm{KL}}\left[p(Z,S) \,|\, p(Z)p(s)\right] \tag{4.4}$$
$$= \mathbb{E}_S\left[D_{\mathrm{KL}}\left[p(Z|S) \,|\, p(Z)\right]\right] \tag{4.5}$$

In practice, we take samples of $D_{\mathrm{KL}}\left[p(Z|S)\,|\,p(Z)\right]$ to estimate the mutual information. While $p(Z|S)$ is straightforward to compute, calculating $p(Z)$ requires marginalization across the entire state space S, which in most non-trivial environments are intractable. Instead, we follow the method adopted in many recent works and introduce an approximator, $q(Z) \sim \mathcal{N}(\vec{0}, \mathbb{I})$, to replace $p(Z)$.

This achieves an upper bound on $I(Z, S)$:

$$\begin{aligned}
&\mathbb{E}_S[D_{\mathrm{KL}}\left[p(Z|S)\,|\,q(Z)\right]] \\
&= \int_s dx\, p(s) \int_z dz\, p(z|s)\, \log \frac{p(z|s)}{q(z)} \\
&= \int_{z,s} dx\, dz\, p(z|s)\, \log p(z|s) - \int_z dz\, p(z) \log q(z) \\
&\geq \int_{z,s} dx\, dz\, p(z|s)\, \log p(z|s) - \int_z dz\, p(z) \log p(z) \\
&= \int_s dx\, p(s) \int_z dz\, p(z|s)\, \log \frac{p(z|s)}{p(z)} \\
&= I(Z, S)
\end{aligned}$$

where the inequality arises because of the non-negativeness KL-divergence:

$$D_{\mathrm{KL}}[p(z)\,|q(z)] \geq 0 \tag{4.6}$$

$$\int_z dz\, p(z)\, \log p(z) \geq \int_z dz\, p(z)\, \log q(z) \tag{4.7}$$

4.3 Unconstrained Lagrangian

Finally, we introduce a Lagrangian multiplier β and optimize on the upper bound of $I(Z, S)$ given by the approximator $q(Z)$:

$$\mathcal{L}(\theta) = \max_{\theta}\ \mathbb{E}_{\pi_\theta,\tau}[R(\tau)] - \beta \mathbb{E}_S[D_{\mathrm{KL}}\left[p(Z|S)\,|\,q(Z)\right]] \tag{4.8}$$

As discussed in [20], the gradient update at time t is the policy gradient update with the modified reward, minus a scaled penalty by KL-divergence between state and embedding:

$$\begin{aligned}
\nabla_{\theta,t}\mathcal{L}(\theta) = &R'(t)\nabla_\theta \log(\pi_\theta(a_t, s_t)) \\
&-\beta \nabla_\theta D_{\mathrm{KL}}[p(Z|s)\,|\,q(Z)]
\end{aligned} \tag{4.9}$$

where $R'(t) = \sum_{i=1}^{t} \gamma^i r'(i)$ is the discounted reward until step t, and $r'(i)$ is the environment reward $r(i)$ modified by the KL penalty: $r'(i) = r(i) + \beta D_{\mathrm{KL}}[p(Z|s)\,|\,q(Z)]$.

Algorithm 1: Annealing scheme

```
Input: An initial regularization weight $\beta_{init}$
Input: A multiplicative factor $\alpha$
Output: A set of policies $\{\pi_{\theta_t}\}$
pretrain policy $\pi_{\theta_0}$ with deterministic encoder;
$\beta_0 \leftarrow \beta_{init}$;
for $t$ = 1, ..., $T$ do
    $\pi_{\theta_t} \leftarrow \pi_{\theta_{t-1}}$;
    $\beta_t \leftarrow \beta_{t-1} * \alpha$;
    for episode $= 1, \ldots, N$ do
        Collect trajectory $\tau$ using $\pi_{\theta_t}$;
        Update $\theta_t$ using Equation 4.9;
    end
end
```

4.4 Information-Value trade-off by annealing

In analogy to the rate-distortion function mentioned in the preliminaries, we generate a family of solutions (optimal pairs of encoder and policy) parametrized by the information bottleneck constraint weight β. In our case, each solution is characterized by a correspondingly constrained amount of information required to maximize the environment rewards.

The rationale is as follows: to encourage the agent to extract relevant information from the environment, we want to impose high penalty for passing too much information through the encoder. At the beginning of training, such penalty produces gradients that offsets the agent's learning gradients, making it difficult for the agent to form good policies.

To tackle this problem, we create the entire family of solutions through *annealing*, starting from a deterministic (unconstrained) encoder, and gradually injecting noise by increasing the penalty coefficient (*temperature parameter*), β. The details of this annealing scheme are summarized in Algorithm 1.

This approach allows training of well-formed policies for much larger β values, as the encoder has already learned to extract useful information from the environment, and only needs to learn to "forget" more information as β increases. In the experiment section, we will demonstrate that training the model using annealing enables the agent to learn with much larger β coefficients compared to from scratch.

4.5 Domain Randomization and Transfer

Without any additional constraint on the encoder, it is unclear what information the embedding would retain after shrinking the information bottleneck. For a simple pendulum task with a white background and a red arm, for instance, the encoder may learn to only look at the arm or only look at the background; either way, it would allow the agent to solve the task while removing information from the states.

To tackle this issue, we introduce additional domain randomization during training. While domain randomization has been shown to greatly encourage the agent to generalize across domains ([19]), we found that with an appropriate information bottleneck the agent learns to generalize with very limited domain randomization.

Intuitive, if randomization is limited, the agent may learn to solve the task by memorizing all the randomized configurations. For instance, for a maze with 3 similar layouts, it may remember all three layouts and all possible agent positions to solve the task. With a tight information bottleneck, however, the agent is forced to extract limited information from the mazes, so it would have to learn general maze-solving skills.

Our full scheme for learning and transferring domain-independent skills through information bottleneck is detailed in Algorithm 2. In the experiment section, we show that with an information bottleneck, the agent learns to quickly transfer to new environments with extremely limited domain randomization.

Algorithm 2: Domain Randomization with Information Bottleneck

```
Input: A set of training environments $T_{train}$
Input: A set of test environments $T_{test}$
Input: An initial regularization weight $\beta_{init}$
Input: A multiplicative factor $\alpha$
Input: A desired regularization weight $\beta^*$
Input: (Optional) A desired regularization weight for transfer $\beta'$
Output: A policy $\pi^*$ for test environments
train policies $\{\pi_{\theta_t}\}$ using Annealing($\beta_{init}$, $\alpha$), sampling tasks $T \in T_{train}$;
take policy $\pi_{\theta_{t^*}}$ corresponding to regularization weight $\beta^*$;
(Optional) $\beta^* \leftarrow \beta'$;
for episode $= 1, \ldots, N$ do
    Collect trajectory $\tau$ in $T \in T_{test}$ using $\pi_{\theta_t^*}$;
    Update $\theta_{t^*}$ using Equation 4.9;
end
```

Chapter 5

Experiments - Maze Environments

In this chapter, we apply the approaches described in Chapter 4 to discrete maze environments. In doing so, we aim to answer the following questions:

1. How does an agent learn from scratch with an information bottleneck?
2. How effective is learning through annealing compared to learning from scratch?
3. How well can a policy trained end-to-end with an information bottleneck transfer to new, unseen maze layouts?

5.1 Environment Overview

MiniGrid Environments are used as the primary discrete experiments ([4]). The maze environments have relatively simple dynamics, but the reward signals are strictly sparse: the agent does not receive any rewards until it reaches the goal.

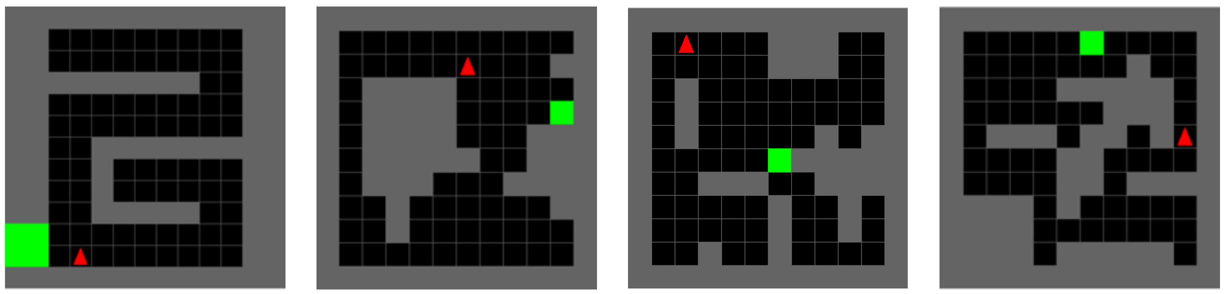


Figure 5.1: Visualization of examples of grid layouts used in this paper. The first layout is fixed for Section 5.2, while the other 3 are sampled from randomly generated maze layouts.

To enforce fairness of comparison and to validate the results statistically, we randomly generate and sample maze environments of the same size to test the agent's ability to transfer to new layouts. Additionally, while the algorithm's behavior is consistent for all mazes, we fix one maze layout for illustration. The fixed layout and examples of the randomly generated layouts are listed in figure 5.1. For these environments, the gray blocks indicate walls, the dark blocks indicate free space, the green blocks the goal, and finally the red triangle the agent. These mazes share the same size of 12-by-12, and their difficulties vary depending on the random seeds. These environments are fully observable: the agent receives full observation information about its position, the goal position, and the placement of the walls as 2D compact grid encodings.

For each transfer experiment, we randomly sample 4 mazes, 3 of which are used for the training set and 1 for testing. For the fixed maze, we fix the goal to be the bottom left corner of the maze, as illustrated in figure 5.1; for all randomly generated mazes, we randomize the goal position at the beginning of the episode. Randomization of the goal is important for generalization, as it dramatically increases the dimension of the underlying task.

This particular experiment setup allows us to focus the agent on extracting the relevant information from the mazes: to solve the tasks in 3 training mazes, the agent may either memorize all three layouts and act accordingly, or it could learn general navigational skills to reach the goal. In the following sections, we show that the agent learns to generalize only with a tight information bottleneck given limited randomization.

5.2 Direct Learning with Information Bottleneck

We first study the behavior of an agent learning with an information bottleneck from scratch, which means the agent tries to directly find an optimal encoder and an optimal policy for a particular β. Specifically, we show that for small values of β, when there is enough information about the state, the agent succeeds to learn.

In the following section we show that, in general, the agent does not succeed to learn from scratch for high values of β (meaning a high weight of the information constraint in the optimization). In this case, the annealing scheme allows to solve the task even for high values of β.

The environment used has a simple layout, and the agent is able to achieve around 0.4 mean reward with a random policy (reward range for this environment is 0 to 0.9). With a relatively small β value of 10^{-4}, the agent quickly learns the optimal policy, achieving mean episode rewards around 0.75 (figure 5.3).

In contrary to general end-to-end training, learning with an information bottleneck can more be interpretable as we reference the mutual information (MI) curve[1] of the encoder's input (raw observation) and output (embedding). In particular, figure 5.3 shows that the MI curve initially drops, before an abrupt jump at around 300,000 frames that directly

[1]This curve is in reality an upper bound on mutual information. For ease of notation, the rest of the paper will still refer to this curve as the MI curve, as long as such ambiguity does not invalidate reasoning.

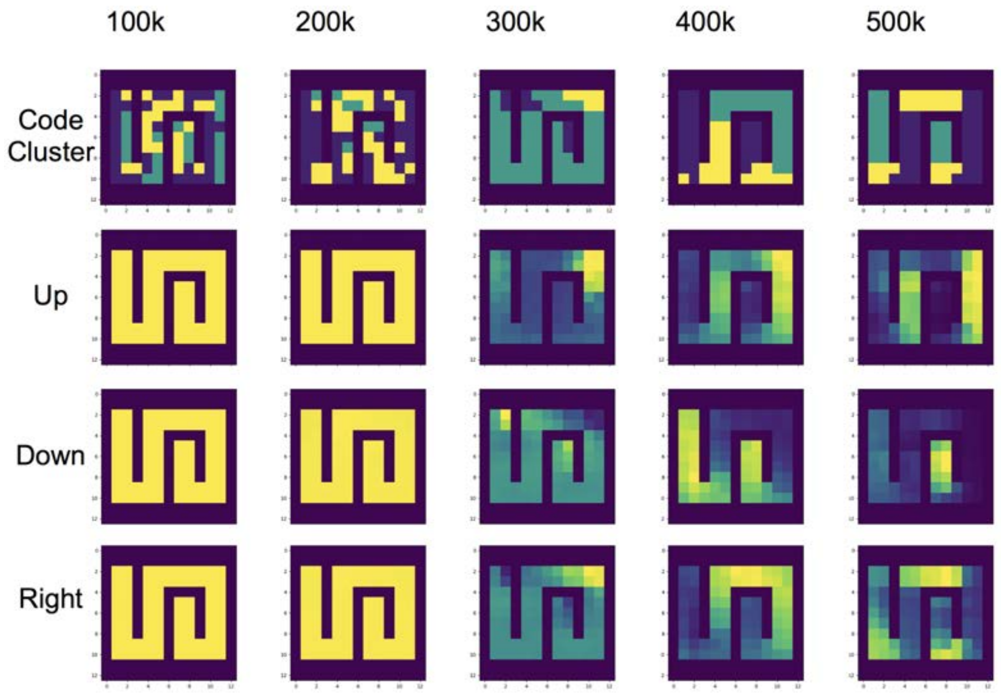


Figure 5.2: Visualization of the encoder output and policy by iterations. For top row, coloring is done by clustering the encoder output; for all subsequent rows, coloring indicates action probability. Note that a solution quickly emerges between 400,000 and 500,000 steps.

corresponds to the increase in the agent's mean reward. To understand such change in the MI curve, we visualize the change in the policy and the encoder throughout training: Figure 5.2 shows the formation of local solutions before a global solution emerges, and the convergence from local solutions to the global solution exactly corresponds to the jump in MI and the mean reward.

This behaviour of information drop before the fast reward accumulation admits a general interpretation: the information bottleneck "squeezes" the information from the state to only allow relevant information required to solve the task. As illustrated in the next section, this behavior causes learning to fail if the initial β value is too high.

Furthermore, we demonstrate that the code learned through information bottleneck learns structured information about the maze. Figure 5.4 illustrates the projection of every state's embedding (after convergence) onto 2D space through T-SNE, with each point colored by its critic value. From the projection plot, we observe the emergence of consistent value gradients as well as local clustering by actions.

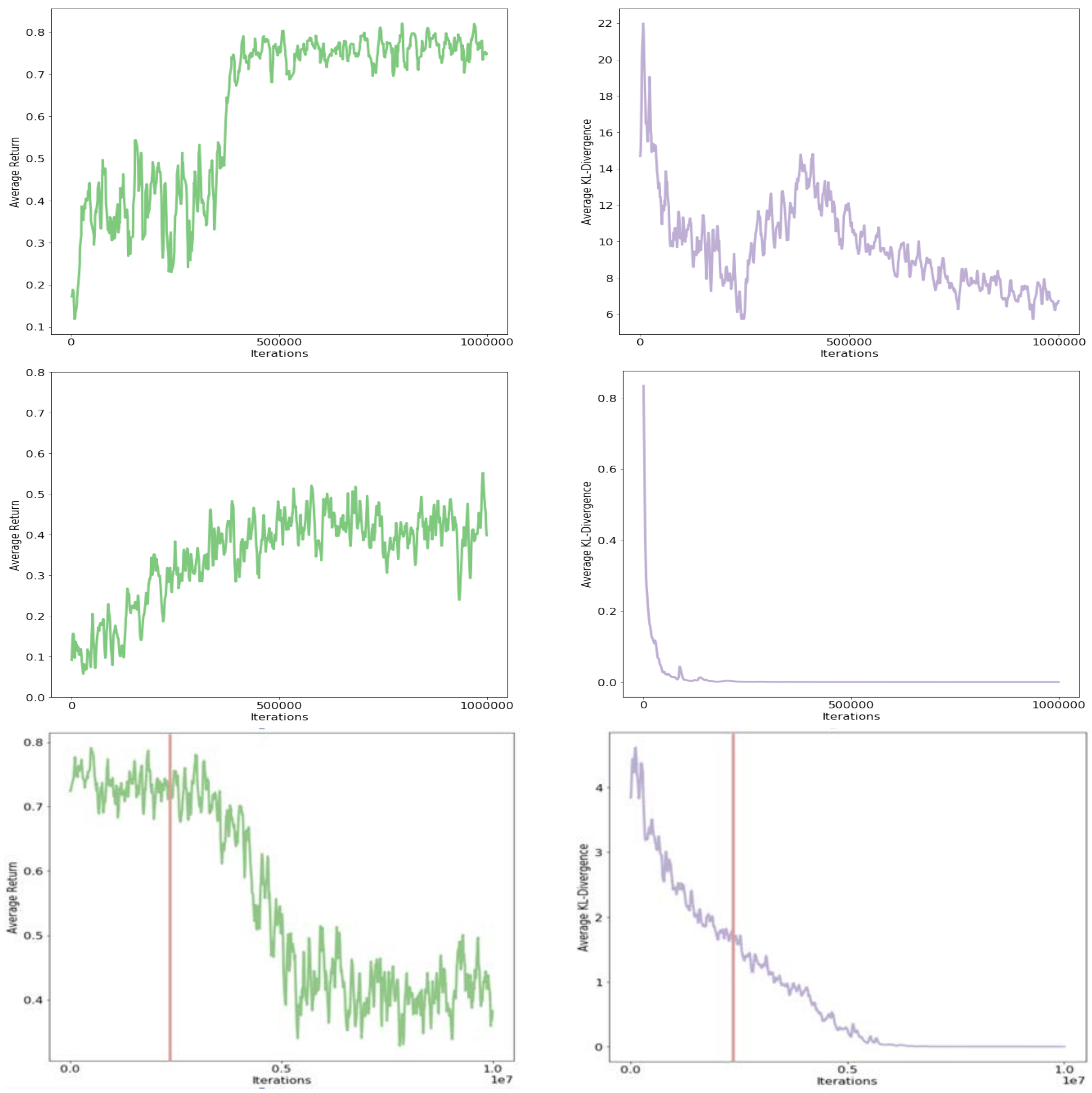


Figure 5.3: Learning and MI curves of the fixed maze (first in figure 5.1). The three sets of curves correspond to $\beta = 0.0001$ from scratch (top) , $\beta = 0.005$ from scratch (middle), and annealed β from 0.0001 to 1 (bottom). For a small β value, the agent is able to learn the optimal policy in 500,000 steps; for larger β values, however, we can only achieve learning through annealing. To better illustrate the learning behavior, a single run is used for each plot.

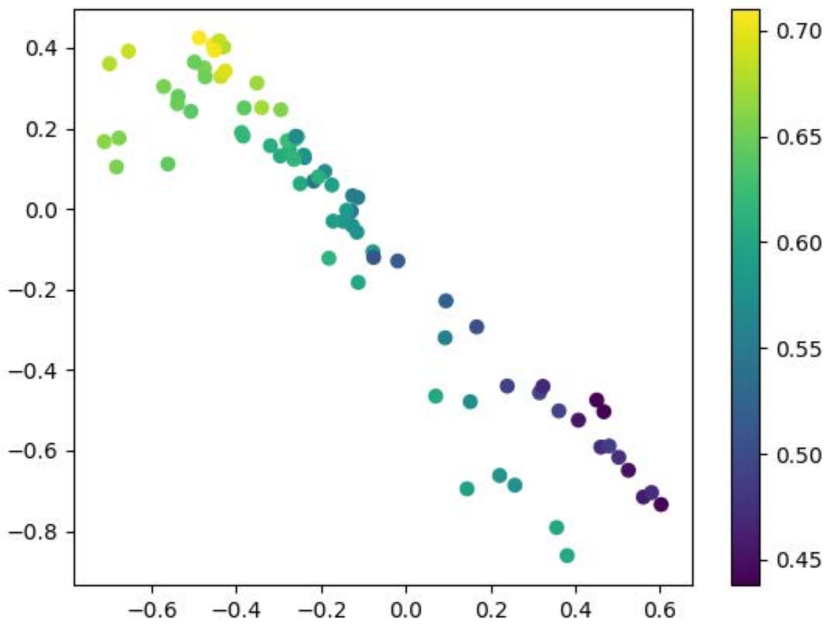


Figure 5.4: T-SNE projection of the encoder output for every state on 2D plane. There exist 1) a consistent color gradient along the diagonal by critic values 2) branching by optimal actions.

5.3 Advantages of Learning Through Annealing

In this section, we show that even with a simple maze layout, the agent still struggles to learn with relatively large β values from scratch; learning through annealing, on the other hand, enables the agent to smoothly compress information while maintaining high quality policies.

Figure 5.3 also illustrates the MI curve of an agent learning the fixed maze from scratch, with a large β value of 0.005. As shown in the plot, the relatively high KL penalty causes a rapid drop in the MI curve, resulting a high stochastic encoder that does not contain any information about the states. Learning through annealing, on the other hand, results in a smooth and gradual drop in MI curve as well as the mean reward. For the same β value of 0.005 (indicated by the red vertical bar), the agent has maintained the optimal policy while achieving higher compression of the states.

As we show in the next section, being able to learn with tight information bottlenecks is crucial for better generalization. This highlights the benefits of annealing, as it allows us to have well-formed solutions for every choices of β.

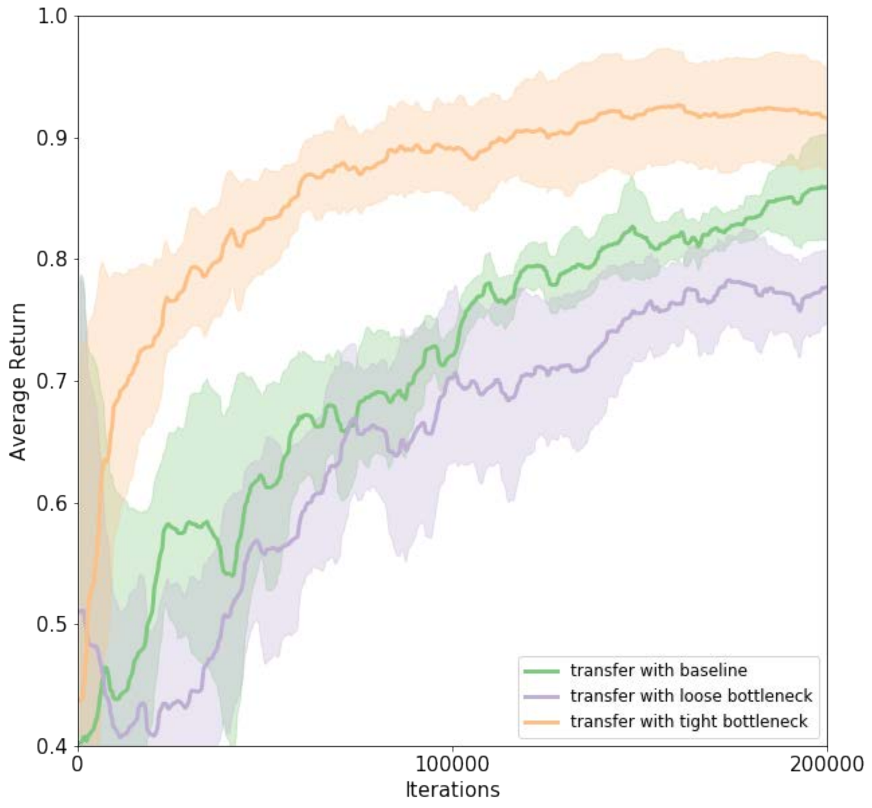


Figure 5.5: Learning curves of randomly generated mazes for baseline and different information bottlenecks. The orange curve reaches near-optimal values the fastest, while the green and purple curves are very similar.

5.4 Generalization through information bottleneck

We now evaluate the transferrability of the agent's policy with an information bottleneck. Specifically, we apply Algorithm 2 to the random mazes, and compare the transfer results to learning with full information.

Figure 5.5 shows the learning curves of three different setups: learning with a tight information bottleneck ($\beta = 0.05$); learning with a loose information bottleneck ($\beta = 0.0001$ as *ablation*); learning with full information ($\beta = 0$ and deterministic encoder as *baseline*).

As the plot shows, learning with a tight information bottleneck achieves the best transfer learning result, reaching near-optimal solution of 0.9 mean reward around 2 times faster compared to the baseline. The close performance between the baseline and the ablation suggests the benefit of generalization only emerges as we tighten the information bottleneck.

Chapter 6

Experiments - Control Environments

We have studied in Chapter 5 the learning behavior of information bottleneck, the benefits of annealing, and the generalization advantages of policies trained through a tight information bottleneck.

We now turn our attention to control environments to fully explore the generalization benefits of information bottlenecks. In particular, we apply our proposed approach (Chapter 4) to control environments, including Reacher, CartPole, and HalfCheetah from OpenAI Gym ([3]). We demonstrate that by finding the optimal information bottleneck size through annealing, we train policies in these environment that efficiently generalize to unseen goal and dynamics, which may require extreme policy extrapolation.

6.1 Reacher Setup and Results

In the Reacher environment, a robotic arm has two sections with a joint in the middle. The arm's one end is fixed at the center of the plane, and its goal is typically to reach a certain point on the plane with the other end of the arm. While this task can be naturally formulated as a sparse reward problem ([10]), we adapt the environment's default dense reward signals for ease of training.

As shown in figure 6.1, we fix 3 goal positions for training and test the agent's ability to generalize to the 4th position. With this setup, we evaluate how well a policy generalizes to a new, unseen goal. As during training we only provide limited goal randomization, it is possible for a policy to simply memorize a separate policy for each goal, causing it to overfit onto the training setup.

For this environment, we study information bottleneck as well as other regularization techniques for generalization ([6]), including L-2 regularization and dropout. With a high β value of 0.9 learned through annealing, we achieve significantly better transfer results compared to the baseline. Other regularization techniques such as $L-2$ penalty and dropout also provide generalization benefits over the baseline, however their performances are below that of a well-tuned information bottleneck. Moreover, combining information bottleneck

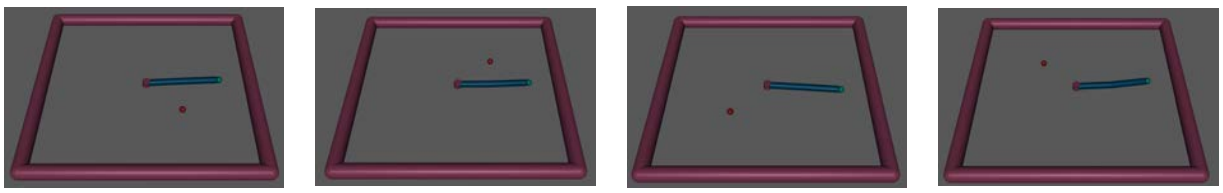


Figure 6.1: Visualization of all 4 goal positions in Reacher environment. The first 3 goals are used for training, while the 4th goal is used for testing.

with dropout, we are able to converge to the optimal policy within 700,000 steps (figure 6.2) [1].

6.2 CartPole Setup and Results

The CartPole environment consists of a pole attached to a cart sliding on a frictionless surface. The pole is free to swing around the connection point to the cart, and the environment goal is to move the cart either left or right to keep the pole upright. The agent obtains a reward of 1 for keeping the pole upright at each time step, and can achieve a maximum of 200 reward over the entire episode. Should the pole fail to maintain an angle of 12 degrees from the vertical line, the episode will terminate early.

Unlike other environments studied in this chapter, the CartPole environment is configured to have 2 discrete actions: moving left and right at each time step. While this results in a simple system, we study a harder generalization problem: generalizing to unseen environment dynamics. Specifically, we vary two environment parameters: the magnitude of the cart's push force, and the length of the pole. The push force affects the cart's movement at each time step, while the length of the pole affects its torque.

We provide limited randomization during training compared the configuration in [16]: we range push forces from 7 to 13, and the pole length from 0.45 to 0.55. For evaluation, we consider a much wider range as well as extreme values: we first test the policy's performance on push forces ranging from 1 to 40 and pole lengths from 0.1 to 1.7; then, we test on extremely large values of push forces (80, 160) and pole lengths (1.7, 3.4, 6.8) to assess the policy's stability. While push force is difficult to visualize, Figure 6.3 illustrates the different pole lengths used for training and evaluation. We consider an oracle-based approach, providing to the agent parameter values, to focus our attention on the policy's generalizability to unseen dynamics.

[1] Combination of information bottleneck and L-2 regularization is trivial and not included, as optimizing mutual information using a variational approximator already uses a L-2 penalty on the encoder.

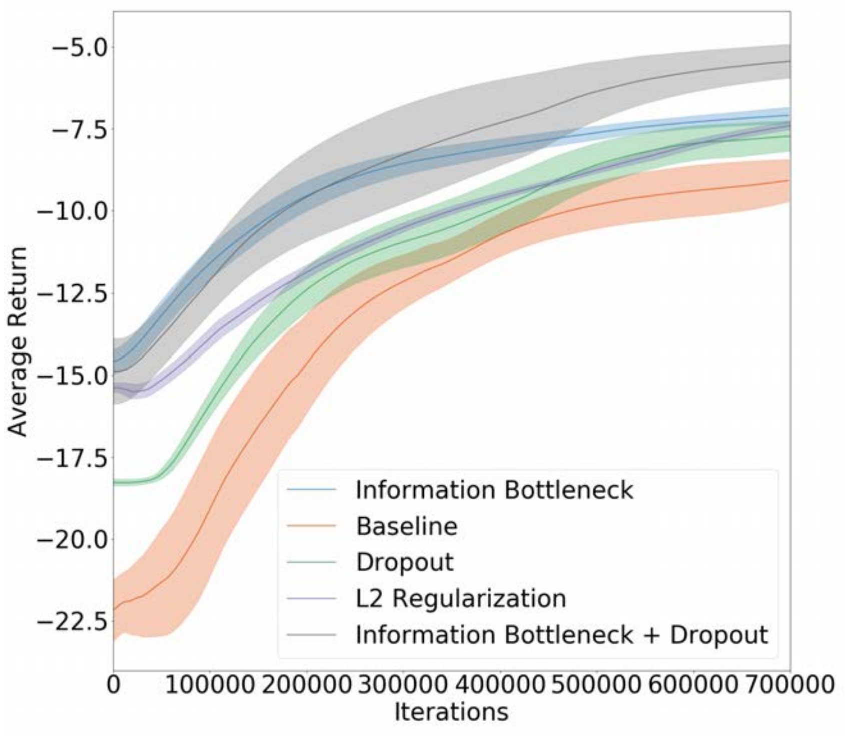


Figure 6.2: Learning curves of Reacher on the test goal. Overall, the combination of information bottleneck and dropout (gray) gives the best performance, starting with a higher reward at zero-shot and converging to the optimal solution much faster than other settings.

Evaluating Generalization Benefits

Given extremely limited randomization during training, the agent may easily overcome the changing dynamics across episodes through memorization. This would lead to very good training performance, but poor generalization over unseen parameters that change the environment dynamics. As illustrated in Figure 6.4, both the baseline and our approach achieve good training performance; the baseline, however, fails to generalize beyond unseen pole lengths, while our method produces a policy that adapts to almost all test configurations. The difference in generalization to unseen dynamics between the baseline and our approach showcases the power of information bottleneck: by limiting the amount of information flow between observation and representation, we force the DNN to learn a general representation of the environment dynamics that can be readily adapted to unseen values.

Perhaps surprisingly, a policy trained with a well-tuned bottleneck performs well even

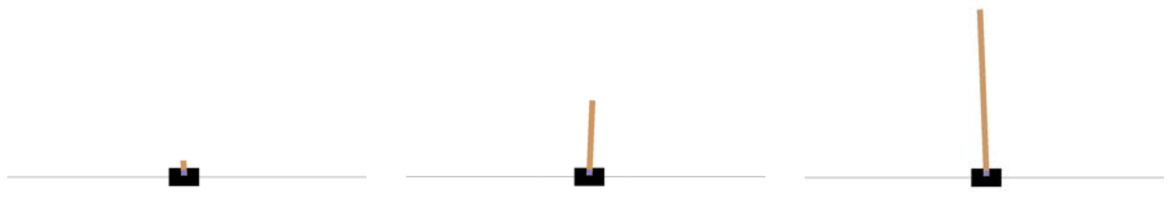


Figure 6.3: Visualization of 3 different pole lengths in the CartPole environment. The lengths are: 0.1 (left), 0.5 (middle), and 1.3 (right). The middle configuration is included in training, while the configurations on two sides are seen only during testing.

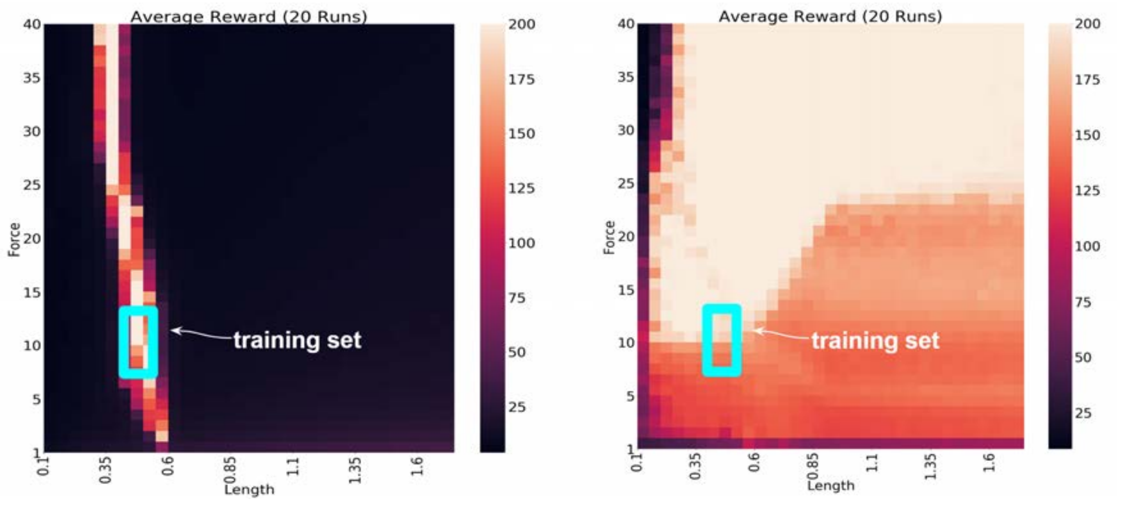


Figure 6.4: Evaluation performance of policies trained through baseline (left) and information bottleneck (right) on CartPole. The x-axis indicates the length of the pole, while the y-axis indicates the push force of the cart. Each evaluation result is averaged over 20 episodes, and the parameter set used in training is marked with a blue rectangle.

in extreme configurations, as seen in Figure 6.5. Even with parameter values that are more than 10 times larger than the average training values, the policy trained using a information bottleneck still achieves optimal performance.

6.3 HalfCheetah Setup and Results

Finally, we demonstrate the generalization benefits of our method in the HalfCheetah environment. In this environment, a bipedal robot with 6 joints and 8 links imitates a 2D cheetah, and its goal is to learn to move in the positive direction without falling over. The

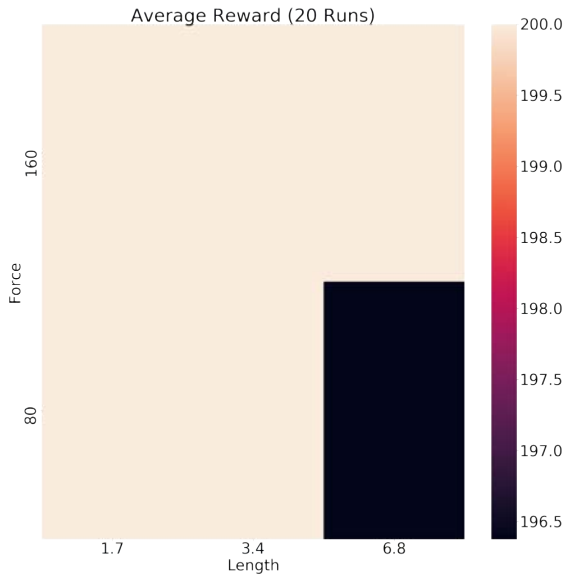


Figure 6.5: Evaluation performance of policy with information bottleneck on extreme configurations in CartPole. The x-axis indicates the length of the pole, while the y-axis indicates the push force of the cart. Each evaluation result is averaged over 20 episodes. Note that the agent achieves near-optimal reward in all 6 configurations.

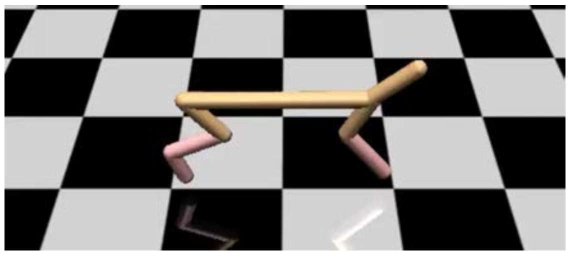
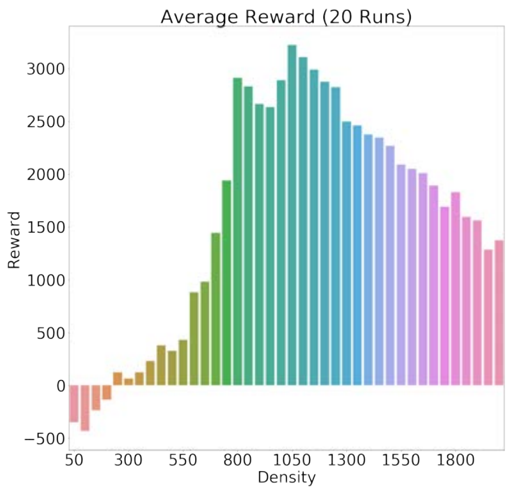


Figure 6.6: Visualization of the HalfCheetah environment. Although not visually different, different configurations of the robot have different torso densities, altering its movement dynamics.



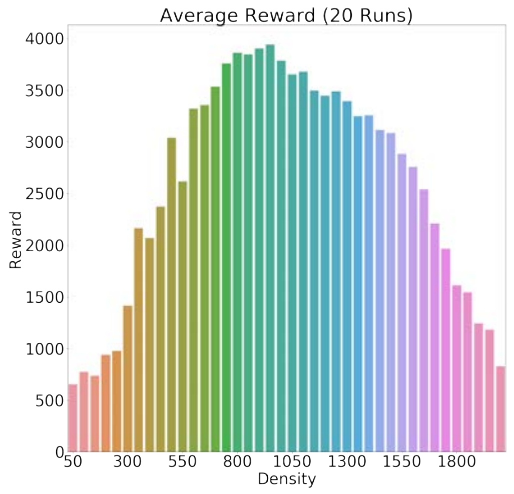


Figure 6.7: Visualization of 3 different pole lengths in the CartPole environment. The lengths are: 0.1 (left), 0.5 (middle), and 1.3 (right). The middle configuration is included in training, while the configurations on two sides are seen only during testing. All evaluations use rewards averaged over 20 episodes.

environment reward is a combination of its velocity in the positive direction and the cost of its movement (in the form of a L-2 cost on action). A illustration of the environment is provided in Figure 6.6.

The environment has continuous actions corresponding to the force values applied to its joints. Its dynamics is more complex in nature compared to CartPole, making generalization a challenging task. Similar to [16], we vary the torso density of the robot to change its movement dynamics. In particular, we vary the training density from 750 to 1000, and test the policy's performance on density values ranging from 50 to 2000. As the robot's actions corresponding to forces, whose effects are linearly affected by density, policy extrapolation from the training parameters to the test parameters is extremely challenging.

As shown in Figure 6.7, while both the baseline's and our method's performances suffer outside of the training range, our method achieves significantly better reward when the density is low. Figure 6.8 better illustrates the performance difference between the baseline and our method: for most test configurations our method performs significantly better than the baseline, especially for density values that are lower than those seen in testing. This again indicates better stability and generalization in the policy trained with an information bottleneck.

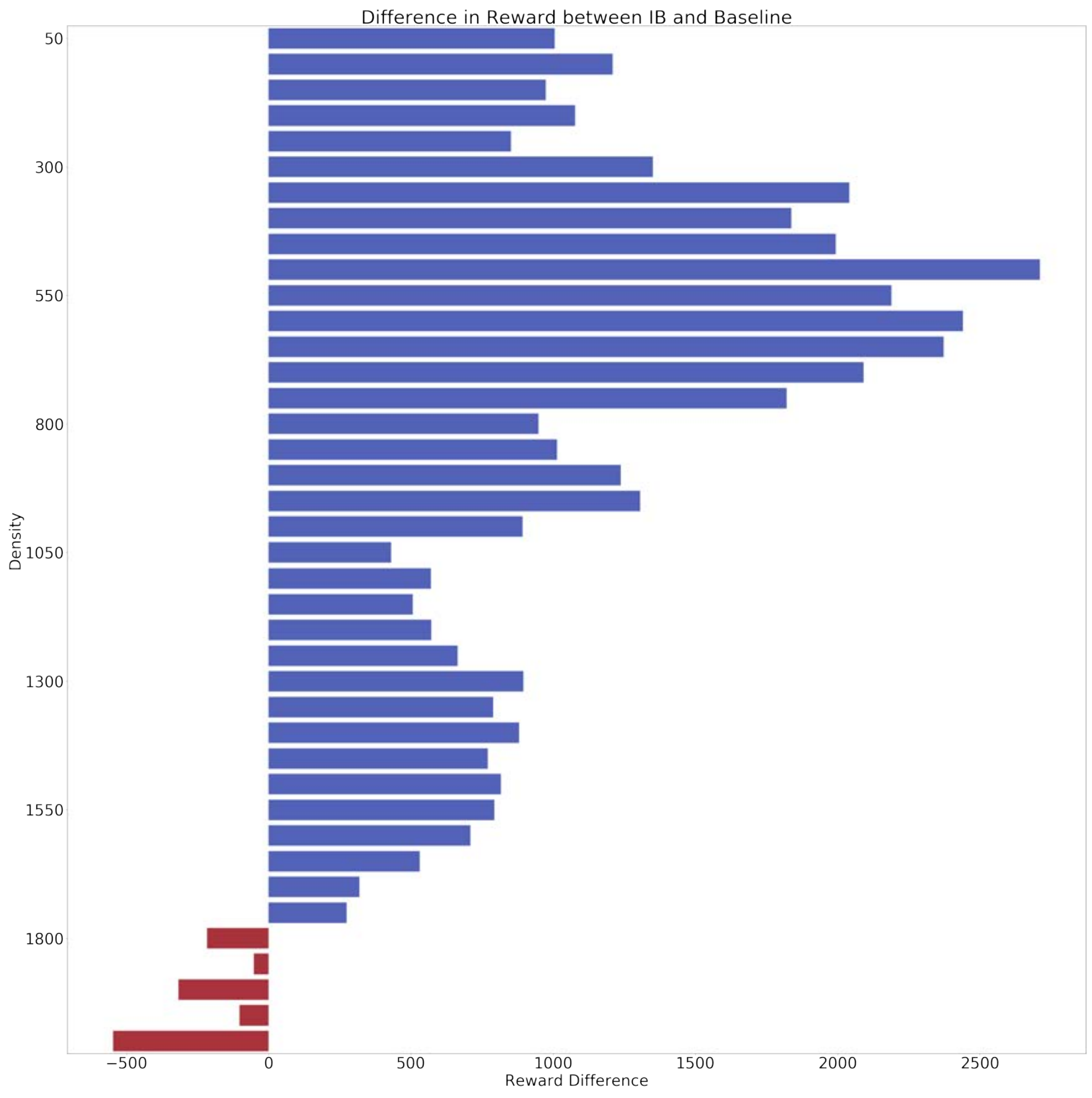


Figure 6.8: Visualization of the evaluation reward difference between the baseline and our method. The y-axis indicates the torso density, while the x-axis indicates the reward difference averaged over 20 episodes. The red bars indicate configurations where the baseline achieves higher evaluation reward than our method, and the blue bars indicate where our method outperforms the baseline.

Chapter 7

Conclusion and Future Work

In this work we proposed a principled way to improve generalization to unseen tasks in deep reinforcement learning, by introducing a stochastic encoder with an information bottleneck optimized through annealing.

We have proved our hypothesis that generalization in DRL can be improved by preventing explicit memorization of training environment observations. We showed that an explicit information bottleneck in the DRL cascade forces the agent to learn to squeeze the minimum amount of information from the observation before the optimal solution is found, preventing it from overfitting onto the training tasks. This led to much better generalization performances (for unseen maze layouts, unseen goals, and unseen dynamics) than baselines and other regularization techniques such as L-2 penalty and dropout.

Practically, we showed that the suggested annealing scheme allowed the agent to find optimal encoder-policy pairs under different information constraints, even for significant information compression that corresponds to very large β values. This annealing scheme was designed to gradually inject noise to the encoder to reduce information (by gradually increasing β), while keeping a well-formed policy that received meaningful RL gradients. This slow change in the values of β is critical, when it is not guarantied to have an optimal simultaneous solutions for the encoder and policy, as in cases where the separation principle is not satisfied.

Overall, we found significant generalization advantages of our approach over the baseline in the maze environment as well as control environment such as CartPole, Reacher, and HalfCheetah. A CartPole policy trained using an information bottleneck, for instance, was able to generalize to test parameters more than 10 times larger than the training parameters, completely beating the baseline's generalization performance.

A promising future direction for research is to rigorously study the properties of the representation space, which may contribute to improving the interpretability of representations in deep neural networks in general. One of the insights of this work was that the produced representation in the maze environments preserved critic value distances of the original states; the representation space was thus consistent with the planning space, allowing generalization over unseen layouts.

Bibliography

[1] Mohamed Ishmael Belghazi et al. “Mutual Information Neural Estimation”. In: *International Conference on Machine Learning*. 2018, pp. 530–539.

[2] Vivek S Borkar and Sanjoy K Mitter. “LQG control with communication constraints”. In: *Communications, Computation, Control, and Signal Processing*. Springer, 1997, pp. 365–373.

[3] Greg Brockman et al. *OpenAI Gym*. 2016. eprint: `arXiv:1606.01540`.

[4] Maxime Chevalier-Boisvert, Lucas Willems, and Suman Pal. *Minimalistic Gridworld Environment for OpenAI Gym*. `https://github.com/maximecb/gym-minigrid`. 2018.

[5] Thomas M Cover and Joy A Thomas. *Elements of information theory*. John Wiley & Sons, 2012.

[6] Jesse Farebrother, Marlos C Machado, and Michael Bowling. “Generalization and regularization in DQN”. In: *arXiv preprint arXiv:1810.00123* (2018).

[7] Anirudh Goyal et al. “Infobot: Transfer and exploration via the information bottleneck”. In: *ICLR2019* (2019).

[8] Tuomas Haarnoja et al. “Learning to walk via deep reinforcement learning”. In: *arXiv preprint arXiv:1812.11103* (2018).

[9] Ilya Kostrikov. *Pytorch implementations of reinforcement learning algorithms*. `https://github.com/ikostrikov/pytorch-a2c-ppo-acktr,2018.`. 2018.

[10] Sameera Lanka and Tianfu Wu. “ARCHER: Aggressive Rewards to Counter bias in Hindsight Experience Replay”. In: *arXiv preprint arXiv:1809.02070* (2018).

[11] Sergey Levine et al. “End-to-end training of deep visuomotor policies”. In: *The Journal of Machine Learning Research* 17.1 (2016), pp. 1334–1373.

[12] Timothy P Lillicrap et al. “Continuous control with deep reinforcement learning”. In: *arXiv preprint arXiv:1509.02971* (2015).

[13] Volodymyr Mnih et al. “Asynchronous methods for deep reinforcement learning”. In: *International conference on machine learning*. 2016, pp. 1928–1937.

[14] Volodymyr Mnih et al. “Asynchronous methods for deep reinforcement learning”. In: *International conference on machine learning*. 2016, pp. 1928–1937.

[15] Vincent Pacelli and Anirudha Majumdar. "Learning Task-Driven Control Policies via Information Bottlenecks". In: *arXiv preprint arXiv:2002.01428* (2020).

[16] Charles Packer et al. "Assessing generalization in deep reinforcement learning". In: *arXiv preprint arXiv:1810.12282* (2018).

[17] Xue Bin Peng et al. "Variational discriminator bottleneck: Improving imitation learning, inverse rl, and gans by constraining information flow". In: *ICLR 2019* (2019).

[18] Martin L Puterman. "Markov Decision Processes: Discrete Stochastic Dynamic Programming". In: (1994).

[19] Xinyi Ren et al. "Domain randomization for active pose estimation". In: *2019 International Conference on Robotics and Automation (ICRA)*. IEEE. 2019, pp. 7228–7234.

[20] DJ Strouse et al. "Learning to share and hide intentions using information regularization". In: *Advances in Neural Information Processing Systems*. 2018, pp. 10249–10259.

[21] Takashi Tanaka, Peyman Mohajerin Esfahani, and Sanjoy K Mitter. "LQG control with minimum directed information: Semidefinite programming approach". In: *IEEE Transactions on Automatic Control* 63.1 (2017), pp. 37–52.

[22] Sekhar Tatikonda and Sanjoy Mitter. "Control under communication constraints". In: *IEEE Transactions on automatic control* 49.7 (2004), pp. 1056–1068.

[23] Sekhar Tatikonda, Anant Sahai, and Sanjoy Mitter. "Stochastic linear control over a communication channel". In: *IEEE transactions on Automatic Control* 49.9 (2004), pp. 1549–1561.

[24] Stas Tiomkin and Naftali Tishby. "A unified bellman equation for causal information and value in markov decision processes". In: *arXiv preprint arXiv:1703.01585* (2017).

[25] Hans S Witsenhausen. "Separation of estimation and control for discrete time systems". In: *Proceedings of the IEEE* 59.11 (1971), pp. 1557–1566.

Chapter 8

Appendix

8.1 Environment Descriptions

GridWorld

The agent is a point that can move horizontally or vertically in a 2-D maze structure. Each state observation is a compact encoding of the maze, with each layer containing information about the placement of the walls, the goal position, and the agent position respectively. The goal state is one in which the goal position and the agent position are the same. The agent obtains a positive reward of 1 when it reaches the goal, and no reward otherwise.

Reacher

The agent is a robotic arm with two rigid sections connected by a joint. The agent moves about a fixed center on the plane by applying a force to each rigid section. Each state observation encodes the angles of two sections, the position of the tip of the arm, and the direction to goal. The reward r_t received at time t is $-(\|p_t - p_g\| + \alpha \|a_t\|)$, where p_t is the position of the tip of the arm at time t, p_g is the position of the goal, and a_t is the action input at time t.

CartPole

The agent is a cart sliding on a frictionless horizontal surface with a pole attached to its top. The pole is free to swing about the cart, and at each time step the cart moves to the left or to the right to keep the pole in upright position. Each sate observation consists of four variables: the cart position, the cart velocity, the pole angle, and the pole velocity at tip. The reward at every time t is 1, and the episode terminates when it reaches 200 in length or when the pole fails to maintain an upright angle of at most 12 degrees.

Environment	Gridworld	Reacher	CartPole	HalfCheetah
State Dimensions	(12, 12, 3)	(11,)	(4,)	(18,)
Action Dimensions	(4,)	(2,)	(2,)	(6,)
Maximum Steps	100	50	200	1000

Table 8.1: Environment dimensions and horizons

HalfCheetah

The agent is a bipedal robot with 6 joints and 8 links imitating a 2D cheetah. The agent moves horizontally on a smooth surface, and its goal is to learn to move in the positive direction without falling over, by applying continuous forces to each individual joint. The state observations encode the robot's position, velocity, joint angles, and joint angular velocities. The reward r_t at each time t is the robot's velocity in the positive direction, $v_t = x_t - x_{t-1}$, minus the action costs $\alpha \|a_t\|$. Here, x_t indicates the position of the robot at time t, and a_t is the robot's action input.

8.2 Network Parameters and Hyperparameters for Learning

For all maze experiments we use standard A2C, and for all control experiments we use PPO. Our baseline is adopted from [9], and we modify the code to add a stochastic encoder.

GridWorld

For baseline, we use 3 layers of convolutional layers with 2-by-2 kernels, and channel size 16, 32, 64 respectively. The convolutional layers are followed by a linear layer ("deterministic encoder") of hidden size 64. Finally, the actor and critic each uses 1 linear layers of hidden size 64. We use Tanh activations between layers. For our approach, we add an additional linear layer after the convolution to output the diagonal variance of the encoder to provide stochasticity.

Reacher

For baseline, we use 1 linear layer of hidden size 64, followed by an additional linear layer of hidden size 32 ("deterministic encoder"). Actor and critic each uses 2 linear layers of hidden size 128. For our approach, we again add an additional linear layer of hidden size 32 after the first linear layer to output the diagonal variance for the stochastic encoder.

Parameter	Value
gamma	0.99
entropy coefficient	0.01
leaning rate	7×10^{-4}
gae-lambda coef	0.95
value loss coef	0.5
encoder dimension	64
β	0.005

Table 8.2: Hyperparameters for GridWorld

CartPole

We follow the same architecture as for Reacher.

HalfCheetah

We follow mostly the same architecture as for Reacher, except the encoder layer has a hidden size of 128 instead of 32.

8.3 Hyperparameter Selection

The most crucial hyperparameter value is β, which determines the size of the information bottleneck. We performed the annealing scheme introduced in Algorithm 1 to find optimal representations and control policies for each β to determine the optimal β value. For all other hyperparameters, we mostly followed the hyperparameters used in each environment's respective baselines, with the exception of tuning the learning rates, batch size, and encoder dimension. Learning rate was tuned through random initialization and short training; batch size and encoder dimension were turned through a binary sweep.

We provide hyperparameter choices in Table 8.2 and Table 8.3 respectively.

8.4 Full Evaluation Results Along Annealing Curve for Cartpole and HalfCheetah

We provide the full evaluation results for both control environments along their respective annealing curves in Figure 8.1 and Figure 8.2. For each set of plots, we demonstrate the gradual increase in generalization performance due to tightening of the information bottleneck, followed by a sudden deterioration of the policy as the encoder loses too much information.

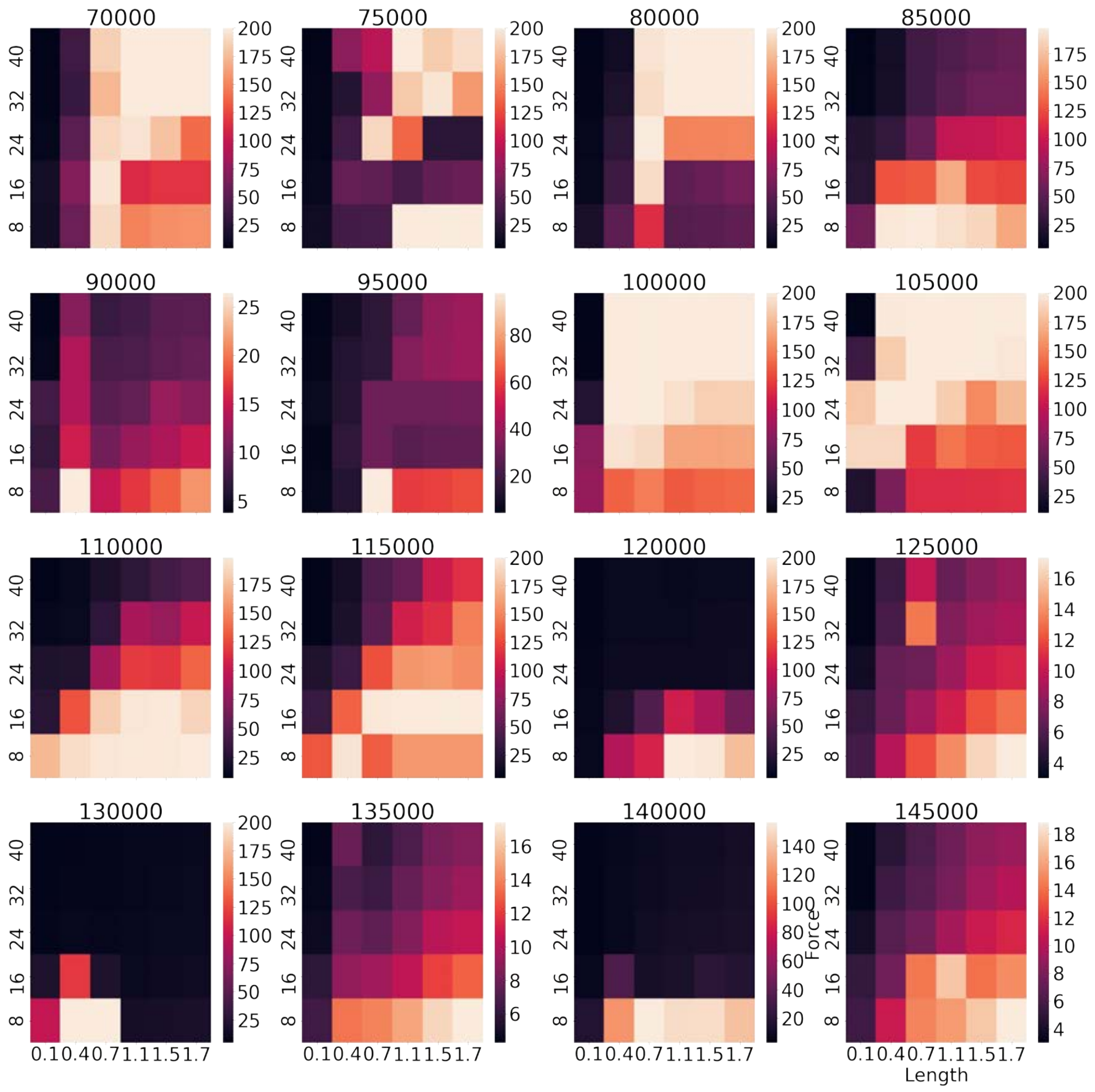


Figure 8.1: Full evaluation results for CartPole policies trained with an information bottleneck through annealing. Each subplot's title indicates the iteration number, the x-axis the pole length, the y-axis the push force, and the value of each cell the evaluation reward averaged over 20 episodes. The best policy was found at iteration 10,000, which corresponds to a β value of 5e-5

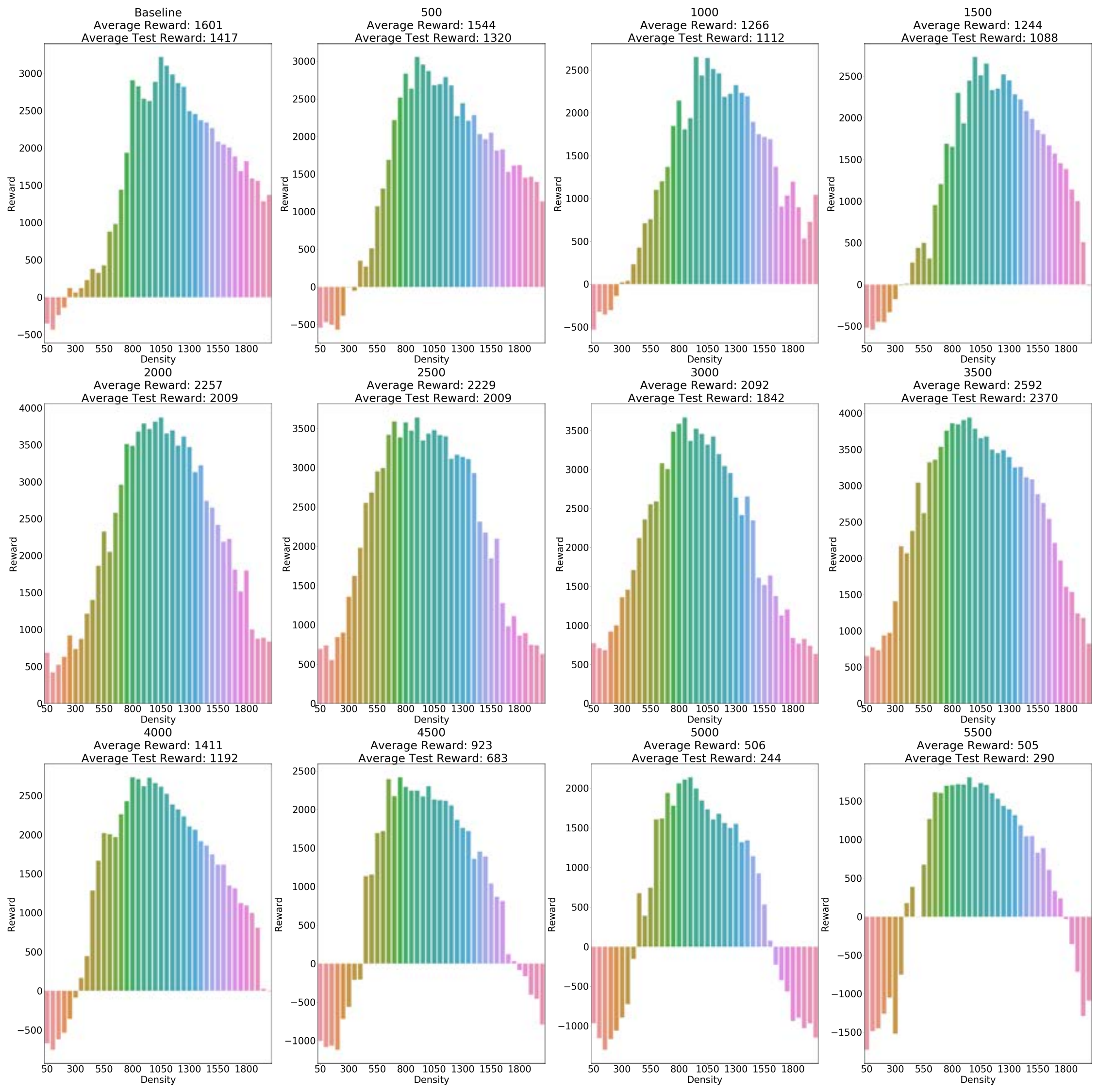


Figure 8.2: Full evaluation results for HalfCheetah policies trained with an information bottleneck through annealing. Each subplot's title indicates the iteration number, overall average reward across all configurations and average test reward across all test configurations. The x-axis indicates the robot's torso length, and the y-axis indicates the evaluation reward averaged over 20 episodes. The best policy was found at iteration 3500, which corresponds to a β value of 5e-4.

Parameter	Value
gamma	0.99
entropy coefficient	0.0
leaning rate	3×10^{-4}
clip range	$[-0.2, 0.2]$
max gradient norm	0.5
batch size	128
gae-lambda coef	0.95
entropy coef	0.01
value loss coef	0.5
encoder dimension	64
β for Reacher	0.9
β for CartPole	5e-5
β for HalfCheetah	5e-4

Table 8.3: Hyperparameters for Reacher, CartPole, and HalfCheetah