Learning About The World Through Video Generation

Wilson Yan

Learning About The World Through Video Generation

By

Wilson Yan

A dissertation submitted in partial satisfaction of the

requirements for the degree of

Doctor of Philosophy

in

Computer Science

in the

Graduate Division

of the

University of California, Berkeley

Fall 2024

Learning About The World Through Video Generation

Abstract

Learning About The World Through Video Generation

by

Wilson Yan

Doctor of Philosophy in Computer Science

University of California, Berkeley

Professor Pieter Abbeel, Chair

Learning large-scale video generation models provides a key avenue to learn about the visual world through internet-scale video data. Learning to generate accurate video requires a model to have a deep understanding of real world concepts, such as motion, physics, object interactions, and 3D consistency. In this dissertation, I will present my research that aims to address core bottlenecks in fundamental architectures and scaling of video generation models, and well us applications of such video models in downstream tasks.

In the first part of my dissertation, I will address computational bottlenecks in video generation models through developing various methods in learning well-compressed, spatio-temporal hierarchical representations of video data. Specifically, I first present VideoGPT, where we learn a compressed latent space with a simple 3D CNN autoencoder that downsamples pixel representations of video in both space and time – resulting in orders of magnitudes of of savings in computation when learning a video generation model in this latent space. Next, I investigate more efficient video generation architectures in TECO that is able to scale to long sequences of videos. I then present ElasticTok, a method that is able to more efficiently encode video data by leveraging adaptive representations with variable-length encodings.

Next, I will focus on algorithmic approaches to scaling to longer context. In Large World Model, we demonstrate core training methodologies to stably train long-context models on a mixtures of language, video, and image data of up to millions of tokens.

Finally, I will present two studies on exploring the use of pre-trained video generation models for downstream tasks. In the first paper, I present VIPER, where we use video prediction model likehoods as reward signal to learn a reinforcement learning agent. I then present MoCA, where we show that video generation models can be used to perform complex video editing tasks.

Contents

List of Figures

List of Tables

Acknowledgments

I would like to deeply thank my advisor Pieter Abbeel for supporting me on my research journey, and general flexibility in terms of accommodating my wide array of research interests over the years. I have worked with Pieter for a long time – starting from the Sophomore year of my undergraduate studies up to the end of my PhD for a total of 7 years. I have learned a lot from him in terms of learning how to do research, think about ideas, and iterate quickly. He provided valuable insights into complex research questions, as well as what problems to focus on that are maximally impactful.

Thank you to all of my many collaborators: Danijar Hafner, Stephen James, Lerrel Pinto, Pieter Abbeel, Hao Liu, Yunzhi Zhang, Aravind Srinivas, Ryo Okumura, Alejandro Escontrela, Ademi Adeniji, Ajay Jain, Xue Bin Peng, Ken Goldberg, Youngwoon Lee, Samaneh Azadi, Andrew Brown, Rohit Girdhar, Matei Zaharia, Yilin Wu, Ashwin Vangipuram, and Amy Lu. It was great working with all of you, and I've learned a lot and grown as a researcher in the process. I would also like to thank all of my other fellow labmates: Younggyo Seo, Kuba Grudzien, Xinyue Chen, Kevin Frans, Fred Shentu, Philipp Wu, Himanshu Singh, Alex Yin, Kevin Zakka, Carlo Sferrazza, Oleh Rybkin, Jongmin Lee, Olivia Watkins, Yuqing Du, and Sherry Yang. I had a great time hanging out with you all and discussing research ideas.

I would additionally like to thank Samaneh Azadi, whom I was fortunate to be my mentor at Meta. Thank you for giving me a chance to work with you also also the fantastic video generation team at Meta, including Rohit Girdhar and Andrew Brown.

Lastly, I thank my mom, dad, and brother – all of whom have supported me since the beginning and over the course of my PhD. I could not have done it without all of you.

Chapter 1

Introduction

Learning generally intelligent systems requires a deep understanding of the underlying physical world, including concepts such as motion, physics, object interactions, and 3D understanding. Recent developments in scaling language models on simple generative modeling objectives have demonstrated astonishing capabilities in difficult tasks such as math, science, and coding. Analogously, can we leverage large-scale generative models on video data for artificial intelligence systems to acquire knowledge about physical concepts? Intuitively, accurately generating short videos requires understanding of basic motion, and simple 3D consistency such as when a camera pans a little to the left or right. Generating even longer videos requires learning long-term dependencies in data, such as crafting consistent narratives in a movie, or retaining global 3D consistency when generating a virtual tour of a house.

Prior research in generative models already provide most of the tools need for scaling. Transformers [267] are powerful, general-purpose machine learning architectures that have shown to work on wide array of modalities such as language, audio, video, and proteins. Models are also easy to scale to train on thousands of chips, with common model or activation sharding techniques such as FSDP [68], tensor parallelism [228], and pipeline parallelism [110]. Lastly, recent progress in generative modeling training objectives such as autoregressive over discrete tokens or diffusion have demonstrated capabilities in modeling high complex distributions.

However, the extremely high dimensionality of video data remains a key bottleneck in video generation models – a single 1080p HD 24FPS video that is one minute long is represented as approximately 9 billion floats, or 36GB to even load onto a compute device. This results in high computationally expensive models that are more costly to scale. Tokenization approaches in language models have shown to be effective in reducing data dimensionality, enabling 4K tokens to roughly encode 6 pages of text content, but a similar amount of encoded video tokens only represents at most a few seconds video data. This becomes more prohibitively expensive when scaling to larger resolution or longer video, even on existing large-scale compute clusters. As such, it becomes imperative to develop more efficient video generation architectures that can scale to both high resolution and long video, while still being able to model these complex visual distributions.

In addition, it is less clear on how to leverage the knowledge learned about the world through video generation models, as it is more difficult to similarly prompt these models for more general tasks as in language models.

1.1 Contribution

In this dissertation, I present core contributions in designing scalable methods for video generation, as well as applications of video generation models in downstream tasks.

Chapter 2 focuses on developing a simple, yet scalable method for video generation. In Chapter 3, I study the problem of long video generation, and associated architectural design choices in better modeling global dependencies in longer video. Chapter 4 seeks to advance core fundamentals in prior video autoencoder architectures, and enable models to scale to longer sequences through adaptive tokenization for image and video data.

In Chapter 5, I focus on scaling to even longer videos and context lengths, as well as make algorithmic and training improvements to stably and efficiently jointly train on a combination of language and video.

The final part of my thesis investigates leveraging pretrained video generation models in downstream applications. Chapter 6 uses a video generation as a reward model in learning reinforcement learning agents, and Chapter 7 finetunes an existing video generation model to perform complex video editing tasks.

Chapter 2

VideoGPT: Video Generation Using VQ-VAE and Transformers

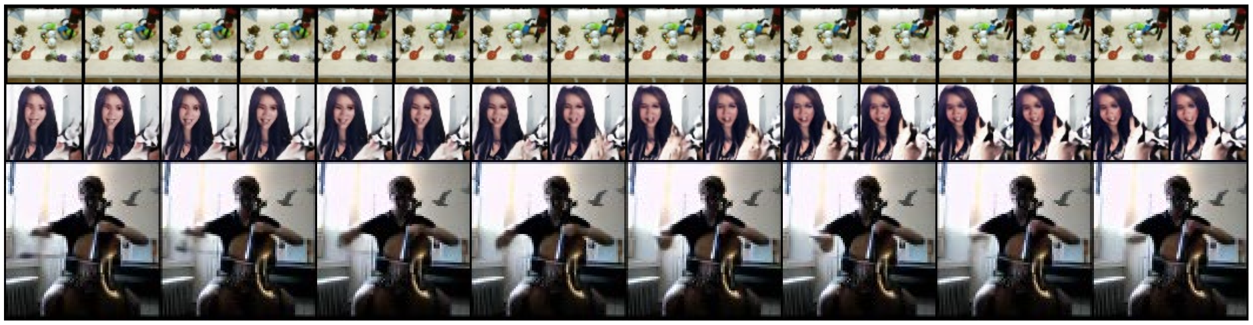


Figure 2.1: 64×64 and 128×128 video samples generated by VideoGPT

2.1 Introduction

Deep generative models of multiple types [130, 81, 264, 60] have seen incredible progress in the last few years on multiple modalities including natural images [265, 306, 23, 129, 99, 124, 125, 266, 205, 262, 101, 39, 203], audio waveforms conditioned on language features [263, 185, 193, 19], natural language in the form of text [196, 28], and music generation [56]. These results have been made possible thanks to fundamental advances in deep learning architectures [95, 264, 265, 267, 306, 169] as well as the availability of compute resources [120, 6] that are more powerful and plentiful than a few years ago.

While there have certainly been impressive efforts to model videos [272, 122, 257, 49], high-fidelity natural videos is one notable modality that has not seen the same level of progress in generative modeling as compared to images, audio, and text. This is reasonable since the complexity of natural videos requires modeling correlations across both space and time with much higher input dimensions. Video modeling is therefore a natural next challenge for current deep generative models. The complexity of the problem also demands more compute resources which can also be deemed as one important reason for the *relatively* slow progress in generative modeling of videos.

Why is it useful to build generative models of videos? Conditional and unconditional video generation implicitly addresses the problem of video prediction and forecasting. Video prediction [240, 70, 123, 234] can be seen as learning a generative model of future frames conditioned on the past frames. Architectures developed for video generation can be useful in forecasting applications for weather prediction [234], autonomous driving (for e.g., such as predicting the future in more semantic and dense abstractions like segmentation masks [163]). Finally, building generative models of the world around us is considered as one way to measure our understanding of physical common sense and predictive intelligence [136].

Multiple classes of generative models have been shown to produce strikingly good samples such as autoregressive models [264, 265, 188, 169, 196, 39], generative adversarial networks (GANs) [81, 195], variational autoencoders (VAEs) [130, 131, 174, 167, 262, 44], Flows [59, 60, 129, 99], vector quantized VAE (VQ-VAE) [266, 205, 203], and lately diffusion and score matching models [233, 236, 101]. These different generative model families have their tradeoffs across various dimensions: sampling speed, sample diversity, sample quality, optimization stability, compute requirements, ease of evaluation, and so forth. Excluding score-matching models, at a broad level, one can group these models into likelihood-based (PixelCNNs, iGPT, NVAE, VQ-VAE, Glow), and adversarial generative models (GANs). The natural question is: What is a good model class to pick for studying and scaling video generation?

First, we make a choice between likelihood-based and adversarial models. Likelihood-based models are convenient to train since the objective is well understood, easy to optimize across a range of batch sizes, and easy to evaluate. Given that videos already present a hard modeling challenge due to the nature of the data, we believe likelihood-based models present fewer difficulties in the optimization and evaluation, hence allowing us to focus on the architecture modeling[1]. Next, among likelihood-based models, we pick autoregressive models simply because they have worked well on discrete data in particular, have shown greater success in terms of sample quality [203], and have well established training recipes and modeling architectures that take advantage of latest innovations in Transformer architectures [267, 45, 100, 111].

[1]It is not the focus of this paper to say likelihood models are better than GANs for video modeling. This is purely a design choice guided by our inclination to explore likelihood based generative models and non-empirically established beliefs with respect to stability of training.

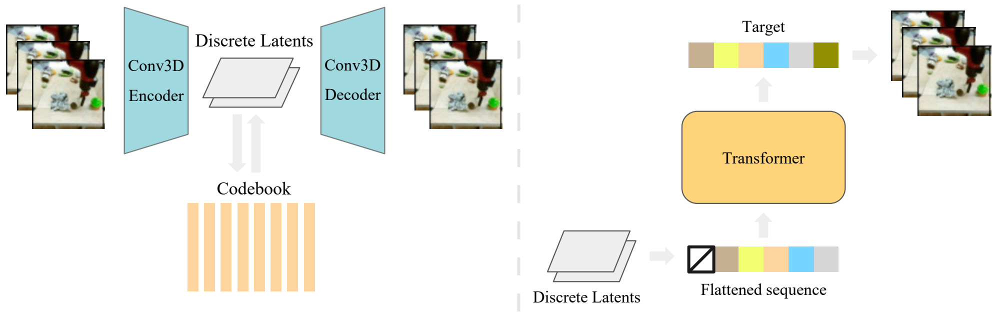


Figure 2.2: We break down the training pipeline into two sequential stages: training VQ-VAE (Left) and training an autoregressive transformer in the latent space (Right). The first stage is similar to the original VQ-VAE training procedure. During the second stage, VQ-VAE encodes video data to latent sequences as training data for the prior model. For inference, we first sample a latent sequence from the prior, and then use VQ-VAE to decode the latent sequence to a video sample.

Finally, among autoregressive models, we consider the following question: Is it better to perform autoregressive modeling in a downsampled latent space without spatio-temporal redundancies compared to modeling at the atomic level of all pixels across space and time? Below, we present our reasons for choosing the former: Natural images and videos contain a lot of spatial and temporal redundancies and hence the reason we use image compression tools such as JPEG [274] and video codecs such as MPEG [137] everyday. These redundancies can be removed by learning a denoised downsampled encoding of the high resolution inputs. For example, 4x downsampling across spatial and temporal dimensions results in 64x downsampled resolution so that the computation of powerful deep generative models is spent on these more fewer and useful bits. As shown in VQ-VAE [266], even a lossy decoder can transform the latents to generate sufficiently realistic samples. This framework has in recent times produce high quality text-to-image generation models such as DALL-E [203]. Furthermore, modeling in the latent space downsampled across space and time instead of the pixel space improves sampling speed and compute requirements due to reduced dimensionality.

The above line of reasoning leads us to our proposed model: VideoGPT[2], a simple video generation architecture that is a minimal adaptation of VQ-VAE and GPT architectures for videos. VideoGPT employs 3D convolutions and transposed convolutions [255] along with axial attention [111, 100] for the autoencoder in VQ-VAE, learning a downsampled set of discrete latents from raw pixels of the video frames. These latents are then modeled using a strong autoregressive prior using a GPT-like [196, 45, 39] architecture. The generated latents from the autoregressive prior are then decoded to videos of the original resolution using the decoder of the VQ-VAE.

[2]We note that Video Transformers [284] also employ generative pre-training for videos using the Subscale Pixel Networks (SPN) [169] architecture. Despite this, it is fair to use the GPT terminology for our model because our architecture more closely resembles the vanilla Transformer in a manner similar to iGPT [39].

Our results are highlighted below:

1. On the widely benchmarked BAIR Robot Pushing dataset [63], VideoGPT can generate realistic samples that are competitive with existing methods such as TrIVD-GAN [164], achieving an FVD of 103 when benchmarked with real samples, and an FVD* [205] of 94 when benchmarked with reconstructions.

2. In addition, VideoGPT is able to generate realistic samples from complex natural video datasets, such as UCF-101 and the Tumblr GIF dataset

3. We present careful ablation studies for the several architecture design choices in VideoGPT including the benefit of axial attention blocks, the size of the VQ-VAE latent space, number of codebooks, and the capacity (model size) of the autoregressive prior.

4. VideoGPT can easily be adapted for action conditional video generation. We present qualitative results on the BAIR Robot Pushing dataset and Vizdoom simulator [127].

2.2 Background

VQ-VAE

The Vector Quantized Variational Autoencoder (VQ-VAE) [266] is a model that learns to compress high dimensional data points into a discretized latent space and reconstruct them. The encoder $E(x) \to h$ first encodes x into a series of latent vectors h which is then discretized by performing a nearest neighbors lookup in a codebook of embeddings $C = \{e_i\}_{i=1}^{K}$ of size K. The decoder $D(e) \to \hat{x}$ then learns to reconstruct x from the quantized encodings. The VQ-VAE is trained using the following objective:

$$\mathcal{L} = \underbrace{\|x - D(e)\|_2^2}_{\mathcal{L}_{\text{recon}}} + \underbrace{\|sg[E(x)] - e\|_2^2}_{\mathcal{L}_{\text{codebook}}} + \underbrace{\beta \, \|sg[e] - E(x)\|_2^2}_{\mathcal{L}_{\text{commit}}}$$

where sg refers to a stop-gradient. The objective consists of a reconstruction loss $\mathcal{L}_{\text{recon}}$, a codebook loss $\mathcal{L}_{\text{codebook}}$, and a commitment loss $\mathcal{L}_{\text{commit}}$. The reconstruction loss encourages the VQ-VAE to learn good representations to accurately reconstruct data samples. The codebook loss brings codebook embeddings closer to their corresponding encoder outputs, and the commitment loss is weighted by a hyperparameter β and prevents the encoder outputs from fluctuating between different code vectors.

An alternative replacement for the codebook loss described in [266] is to use an EMA update which empirically shows faster training and convergence speed. In this paper, we use the EMA update when training the VQ-VAE.

GPT

GPT and Image-GPT [39] are a class of autoregressive transformers that have shown tremendous success in modelling discrete data such as natural language and high dimensional images. These models factorize the data distribution $p(x)$ according to $p(x) = \prod_{i=1}^{d} p(x_i|x_{<i})$ through masked self-attention mechanisms and are optimized through maximum likelihood. The architectures employ multi-head self-attention blocks followed by pointwise MLP feedforward blocks following the standard design from [267].

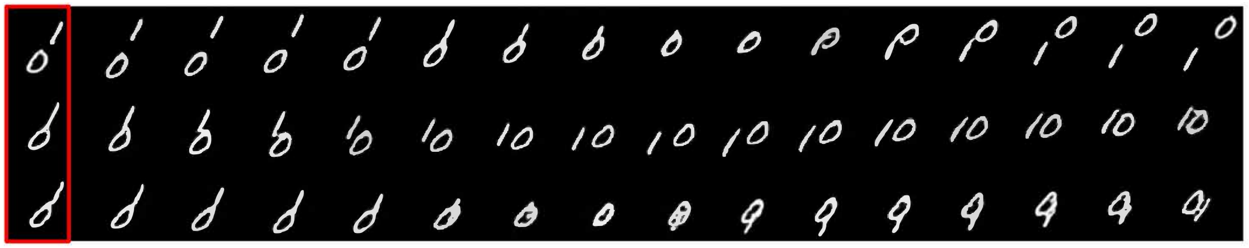


Figure 2.3: Moving MNIST samples conditioned on a single given frame (red).

2.3 VideoGPT

Our primary contribution is VideoGPT, a new method to model complex video data in a computationally efficient manner. An overview of our method is shown in Fig 2.2.

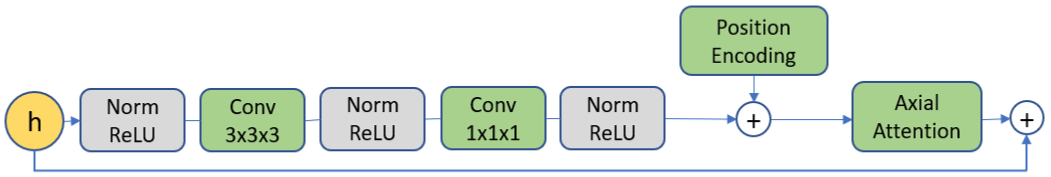


Figure 2.4: Architecture of the attention residual block in the VQ-VAE as a replacement for standard residual blocks.

Learning Latent Codes In order to learn a set of discrete latent codes, we first train a VQ-VAE on the video data. The encoder architecture consists of a series of 3D convolutions that downsample over space-time, followed by attention residual blocks. Each attention residual block is designed as shown in Fig 2.4, where we use LayerNorm [9], and axial attention layers following [111, 100].

The architecture for the decoder is the reverse of the encoder, with attention residual blocks followed by a series of 3D transposed convolution that upsample over space-time. The position encodings are learned spatio-temporal embeddings that are shared between all axial attention layers in the encoder and decoder.

Learning a Prior The second stage of our method is to learn a prior over the VQ-VAE latent codes from the first stage. We follow the Image-GPT architecture for prior network, except that we add dropout layers after the feedforward and attention block layers for regularization.

Although the VQ-VAE is trained unconditionally, we can generate conditional samples by training a conditional prior. We use two types of conditioning:

- **Cross Attention**: For video frame conditioning, we first feed the conditioned frames into a 3D ResNet, and then perform cross-attention on the ResNet output representation during prior network training.
- **Conditional Norms**: Similar to conditioning methods used in GANs, we parameterize the gain and bias in the transformer Layer Normalization [9] layers as affine functions of the conditional vector. This conditioning method is used for action and class-conditioning models.

2.4 Experiments

In the following section, we evaluate our method and design experiments to answer the following questions:

- Can we generate high-fidelity samples from complex video datasets?
- How do different architecture design choices for VQ-VAE and prior network affect performance?

Training Details

All image data is scaled to $[-0.5, 0.5]$ before training. For VQ-VAE training, we use random restarts for embeddings, and codebook initialization by copying encoder latents as described in [56]. In addition, we found VQ-VAE training to be more stable (less codebook collapse) when using Normalized MSE for the reconstruction loss, where MSE loss is divided by the variance of the dataset. For all datasets except UCF-101, we train on 64×64 videos of sequence length 16. For the transformer, we train Sparse Transformers [45] with local and strided attention across space-time. Exact architecture details and hyperparameters can be found in Appendix A.1. We achieve all results with a maximum of 8 Quadro RTX 6000 GPUs (24 GB memory).

Moving MNIST

For Moving MNIST, VQ-VAE downsamples input videos by a factor of 4 over space-time (64x total reduction), and contains two residual layers with no axial-attention. We use a codebook of 512 codes, each 64-dim embeddings. To learn the single-frame conditional prior, we train a conditional transformer with 384 hidden features, 4 heads, 8 layers, and a ResNet-18 single frame encoder. Fig 2.3 shows several different generated trajectories conditioned on a single frame.

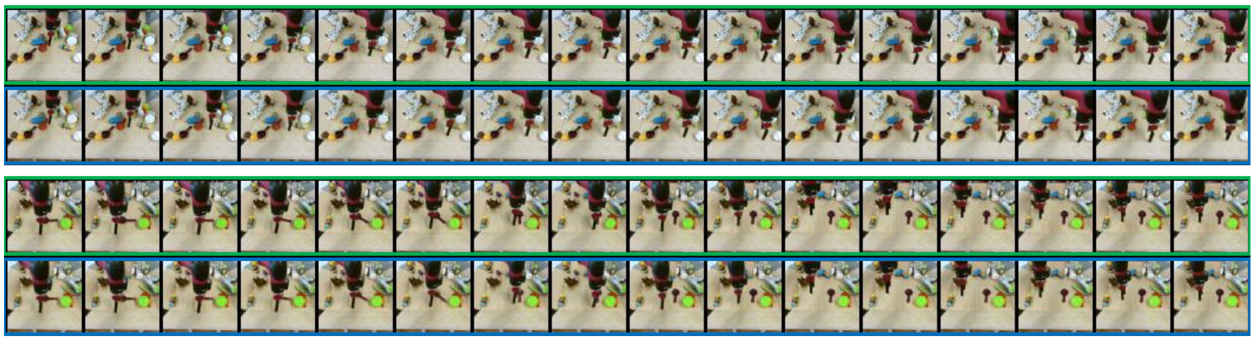


Figure 2.5: VQ-VAE reconstructions for BAIR Robot Pushing. The original videos are contained in green boxes and reconstructions in blue.

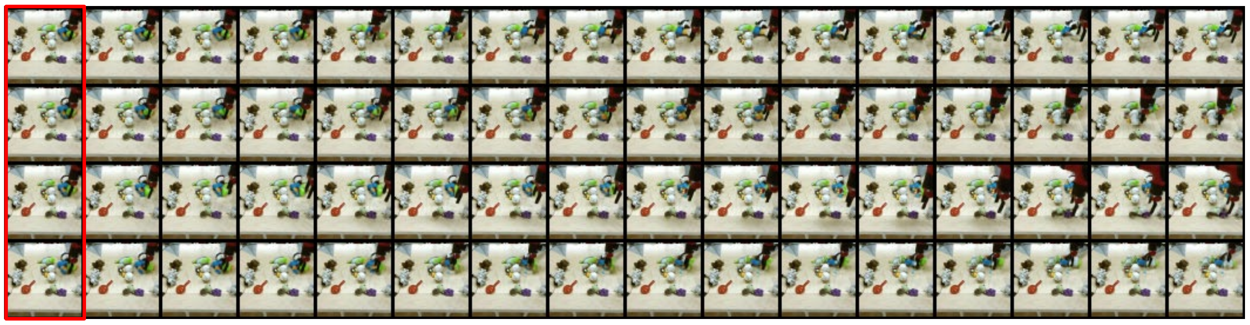


Figure 2.6: BAIR Robot Pushing samples from a single-frame conditioned VideoGPT model. Frames highlighting in red are conditioning frames. Although all videos follow the same starting frame, the samples eventually diverge to varied trajectories.

BAIR Robot Pushing

For BAIR, VQ-VAE downsamples the inputs by a factor of 2x over each of height, width and time dimensions. The embedding in the latent space is a 256-dimensional vector, which is discretized through a codebook with 1024 codes. We use 4 axial-attention residual blocks for the VQ-VAE encoder and a prior network with a hidden size of 512 and 16 layers.

Quantitatively, Table 2.1[3] shows FVD results on BAIR, comparing our method with prior work. Although our method does not achieve state of the art, it is able to produce very realistic samples competitive with the best performing GANs.

Qualitatively, Fig 2.5 shows VQ-VAE reconstructions on BAIR. Fig 2.6 shows samples primed with a single frames. We can see that our method is able to generate realistically looking samples. In addition, we see that VideoGPT is able to sample different trajectories from the same initial frame, showing that it is not simply copying the dataset.

Table 2.1: FVD on BAIR

Method[3]	FVD (↓)
SV2P	262.5
LVT	125.8
SAVP	116.4
DVD-GAN-FP	109.8
VideoGPT (ours)	103.3
TrIVD-GAN-FP	103.3
Video Transformer	94 ± 2
FitVid	**93.6**

Table 2.2: IS on UCF-101

Method[4]	IS (↑)
VGAN	8.31 ± 0.09
TGAN	11.85 ± 0.07
MoCoGAN	12.42 ± 0.03
Progressive VGAN	14.56 ± 0.05
TGAN-F	22.91 ± 0.19
VideoGPT (ours)	24.69 ± 0.30
TGANv2	28.87 ± 0.67
DVD-GAN	$\mathbf{32.97 \pm 1.7}$

ViZDoom

For ViZDoom, we use the same VQ-VAE and transformer architectures as for the BAIR dataset, with the exception that the transformer is trained without single-frame conditioning. We collect the training data by training a policy in each ViZDoom environment, and collecting rollouts of the final trained policies. The total dataset size consists of 1000 episodes of length 100 trajectories, split into an 8:1:1 train / validataion / test ratio. We experiment on the Health Gathering Supreme and Battle2 ViZDoom environments, training both unconditional and action-conditioned priors. VideoGPT is able to capture complex 3D camera movements and environment interactions. In addition, action-conditioned samples are visually consistent with the input action sequence and show a diverse range of backgrounds and scenarios under different random generations for the same set of actions. Samples can be found in Appendix A.2.[4]

[3]SV2P [10], SAVP [139], DVD-GAN-FP [49], Video Transformer [284], Latent Video Transformer (LVT) [200], and TrIVD-GAN [164] are our baselines

[4]VGAN [272], TGAN [216], MoCoGAN [257], Progressive VGAN [3], TGAN-F [121], TGANv2 [215], DVD-GAN [49] are our baselines for IS on UCF-101.

UCF-101

UCF-101 [237] is an action classification dataset with 13,320 videos from 101 different classes. We train unconditional VideoGPT models on 16 frame 64×64 and 128×128 videos, where the original videos have their shorter side scaled to 128 pixels, and then center cropped.

Table 2.2 shows results comparing Inception Score[5] (IS) [217] calculations against various baselines. Unconditionally generated samples are shown in Figure 2.7. Similarly observed in [49], we notice that that VideoGPT easily overfits UCF-101 with a train loss of 3.40 and test loss of 3.12, suggesting that UCF-101 may be too small a dataset of the relative complexity of the data itself, and more exploration would be needed on larger datasets.

Tumblr GIF (TGIF)

TGIF [152] is a dataset of 103,068 selected GIFs from Tumblr, totalling around 100,000 hours of video. Figure 2.8 shows samples from a trained unconditional VideoGPT model. We see that the video sample generations are able to capture complex interactions, such as camera movement, scene changes, and human and object dynamics. Unlike UCF-101, VideoGPT did not overfit on TGIF with a train loss of 2.87 and test loss 2.86.

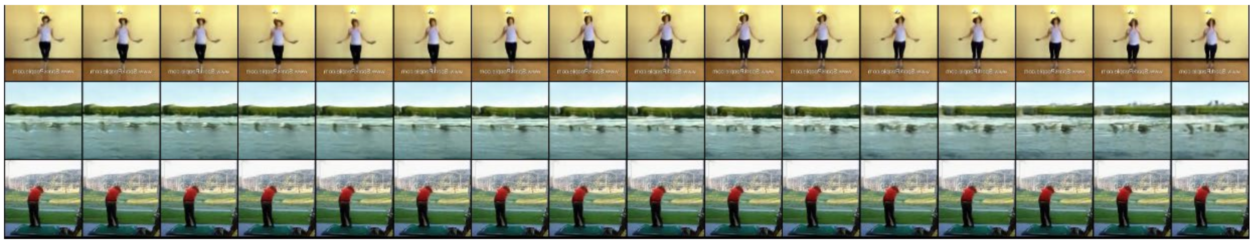


Figure 2.7: 128×128 UCF-101 unconditional samples

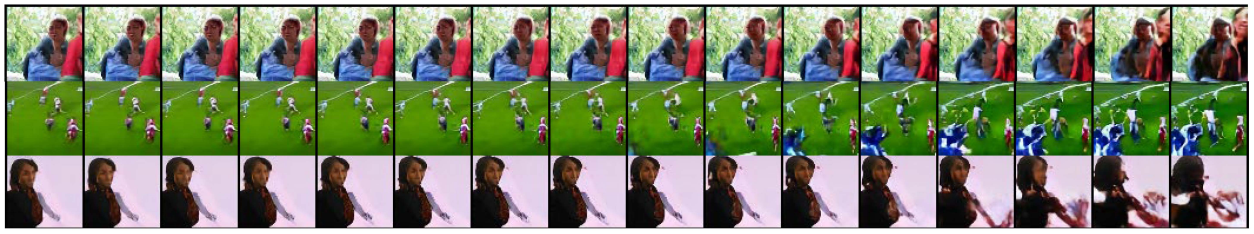


Figure 2.8: 64×64 TGIF unconditional samples

[5]Inception Score is calculated using the code at https://github.com/pfnet-research/tgan2

Ablations

In this section, we perform ablations on various architectural design choices for VideoGPT.

Axial-attention in VQ-VAE increases reconstruction and generation quality.

Table 2.3: Ablation on attention in VQ-VAE. R-FVD is with reconstructed examples

VQ-VAE Architecture	NMSE ($\downarrow$)	R-FVD ($\downarrow$)
No Attention	0.0041	15.3
With Attention	$\mathbf{0.0033}$	$\mathbf{14.9}$

We compare VQ-VAE with and without axial attention blocks as shown in Table 2.3. Empirically, incorporating axial attention into the VQ-VAE architecture improves reconstruction (NMSE) performance, and has better reconstruction FVD. Note that in order to take into account the added parameter count from attention layers, we increase the number of convolutional residual blocks in the ”No Attention” version for better comparability. Fig 2.5 shows samples of videos reconstructed by VQ-VAE with axial attention module.

Larger prior network capacity increases performance.

Table 2.4: Ablations comparing the number of transformer layers

Transformer Layers	bits/dim	FVD ($\downarrow$)
2	2.84	120.4 ± 6.0
4	2.52	110.0 ± 2.4
8	2.39	103.3 ± 2.2
16	2.05	103.6 ± 2.0

Computational efficiency is a primary advantage of our method, where we can first use the VQ-VAE to downsample by space-time before learning an autoregressive prior. Lower resolution latents allow us to train a larger and more expressive prior network to learn complex data distributions under memory constraints. We run an ablation on the prior network size which shows that a larger transformer network produces better results. Table 2.4 shows the results of training transformers of varied number of layers on BAIR. We can see that for BAIR, our method benefits from training larger models, where the bits per dim shows substantial improvement in increasing layers, and FVD and sample quality show increments in performance up until around 8 layers.

A balanced temporal-spatial downsampling in VQ-VAE latent space increase performance.

Table 2.5: Ablations comparing different VideoGPT latent sizes on BAIR. R-FVD is the FVD of VQ-VAE reconstructions, and FVD* is the FVD between samples generated by VideoGPT and samples encoded-decoded from VQ-VAE. For each partition below, the total number of latents is the same with varying amounts of spatio-temporal downsampling

Latent Size	R-FVD (↓)	FVD (↓)	FVD* (↓)
$4 \times 16 \times 16$	82.1	135.4 ± 3.7	81.8 ± 2.3
$16 \times 8 \times 8$	108.1	166.9 ± 3.1	81.6 ± 2.2
$8 \times 16 \times 16$	49.9	124.7 ± 2.7	90.2 ± 2.4
$1 \times 64 \times 64$	41.6	126.7 ± 3.1	98.1 ± 3.6
$4 \times 32 \times 32$	28.3	104.6 ± 2.7	90.6 ± 2.7
$16 \times 16 \times 16$	32.8	113.4 ± 2.5	94.9 ± 1.7
$2 \times 64 \times 64$	22.4	124.3 ± 1.4	104.4 ± 2.5
$\mathbf{8 \times 32 \times 32}$	$\mathbf{14.9}$	$\mathbf{103.6 \pm 2.0}$	$\mathbf{94.6 \pm 1.5}$
$4 \times 64 \times 64$	15.7	109.4 ± 2.1	102.3 ± 2.8
$16 \times 32 \times 32$	10.1	118.4 ± 3.2	113.8 ± 3.3

A larger downsampling ratio results in a smaller latent code size, which allows us to train larger and more expressive prior models. However, limiting the expressivity of the discrete latent codes may introduce a bottleneck that results in poor VQ-VAE reconstruction and sample quality. Thus, VideoGPT presents an inherent trade-off between the size of the latents, and the allowed capacity of prior network. Table 2.5 shows FVD results from training VideoGPT on varying latent sizes for BAIR. We can see that larger latents sizes have better reconstruction quality (lower R-FVD), however, the largest latents $16 \times 32 \times 32$ does not perform the best sample-quality-wise due to limited compute constraints on prior model size. On the other hand, the smallest set of latents $4 \times 16 \times 16$ and $16 \times 8 \times 8$ have poor reconstruction quality and poor samples. There is a sweet-spot in the trade-off at around $8 \times 32 \times 32$ where we observe the best sample quality.

In addition to looking at the total number of latents, we also investigate the appropriate downsampling for each latent resolution. Each partition in Table 2.5 shows latent sizes with the same number of total latents, each with different spatio-temporal downsampling allocations. Unsurprisingly, we find that a balance of downsampling ratio ($2 \times 2 \times 2$, corresponding to latent size $8 \times 32 \times 32$) between space and time is the best, as opposed to downsampling over only space or only time.

Further increasing the number of latent codes does not affect performance.

Table 2.6: Ablations comparing the number of codebook codes

# of Codes	R-FVD (↓)	FVD (↓)	bits/dim
256	18.2	103.8 ± 3.7	1.55
1024	14.9	103.6 ± 2.0	2.05
4096	11.3	103.9 ± 5.1	2.60

In Table 2.6, we show experimental results for running VideoGPT with different number of codes in the codebooks. For all three runs, the VQ-VAE latent code vector has size $8 \times 32 \times 32$. In the case of BAIR, we find that reconstruction quality improves with increasing the number of codes due to better expressivity in the discrete bottleneck. However, they ultimately do not affect sample quality. This may be due to the fact that in the case of BAIR, using 256 codes surpasses a base threshold for generation quality.

Using one VQ-VAE codebook instead of multiple improves performance.

Table 2.7: Ablations comparing the number of codebooks

Latent Size	R-FVD (↓)	FVD (↓)	bits/dim
$\mathbf{8 \times 32 \times 32 \times 1}$	$\mathbf{14.9}$	$\mathbf{103.6 \pm 2.0}$	$\mathbf{2.05}$
$16 \times 16 \times 16 \times 2$	17.2	106.3 ± 1.7	2.41
$8 \times 16 \times 16 \times 4$	17.7	131.4 ± 2.9	2.68
$4 \times 16 \times 16 \times 8$	23.1	135.7 ± 3.3	2.97

In our main results, we use one codebook for VQ-VAE. In Table 2.7, we compare VideoGPT with different number of codebooks. Specifically, multiple codebooks is implemented by multiplying VQ-VAE's encode output channel dimension by C times, where C is the number of codebooks. The encoder output is then sliced along channel dimension, and each slice is quantized through a separate codebook. As a result, the size of the discrete latents are of dimension $T \times H \times W \times C$, as opposed to $T \times H \times W$ when using a single codebook. Generally, multiple codebooks may be more favorable over increasing the downsampling resolution as multiple codebooks allows a combinatorially better scaling in bottleneck complexity. In our experiments, we increase the number of codebooks, and reduce spatio-temporal resolutions on latent sizes to keep the size of the latent space constant. We see that increasing the number of codebooks worsens sample quality performance, and the best results are attained at the highest resolution with one codebook. Nevertheless, incorporating multiple codebooks might shows its advantage when trained with a larger dataset or a different VQ-VAE architecture design.

2.5 Related Work

Video Predictiont The problem of video prediction [240] is quite related to video generation in that the latter is one way to solve the former. Plenty of methods have been proposed for video prediction on the BAIR Robot dataset [70, 63, 10, 54, 53, 139] where the future frames are predicted given the past frame(s) and (or) action(s) of a robot arm moving across multiple objects thereby benchmarking the ability of video models to capture object-robot interaction, object permanance, robot arm motion, etc. Translating videos to videos is another paradigm to think about video prediction with a prominent example being `vid2vid` [276]. The `vid2vid` framework uses automatically generated supervision from more abstract information such as semantic segmentation [163] masks, keypoints, poses, edge detectors, etc to further condition the GAN based video translation setup.

Video Generation Most modern generative modeling architectures allow for easy adaptation of unconditional video generation to conditional versions through conditional batch-norm [23], concatenation [218, 265], etc. Video Pixel Networks [123] propose a convolutional LSTM based encoding of the past frames to be able to generate the next frame pixel by pixel autoregressively with a PixelCNN [265] decoder. The architecture serves both as a video generative as well as predictive model, optimized through log-likelihood loss at the pixel level. Subscale Video Transformers [284] extend the idea of Subscale Pixel Networks [169] for video generation at the pixel level using the subscale ordering across space and time. However, the sampling time and compute requirements are large for these models. In the past, video specific architectures have been proposed for GAN based video generation with primitive results by [272]. Recently, DVD-GAN proposed by [49] adopts a BigGAN like architecture for videos with disentangled (axial) non-local [278] blocks across space and time. They present a wide range of results, unconditional, past frame(s) conditional, and class conditional video generation.

Other examples of prior work with video generation of GANs include [216], [257], [3], [303]. In addition, [215] and [121] propose more scalable and efficient GAN models for training on less compute. Our approach builds on top of VQ-VAE [266] by adapting it for video generation. A clean architecture with VQ-VAE for video generation has not been presented yet and we hope VideoGPT is useful from that standpoint. While VQ-VAE-2 [205] proposes using multi-scale hierarchical latents and SNAIL blocks [42] (and this setup has been applied to videos in [273]), the pipeline is inherently complicated and hard to reproduce. For simplicity, ease of reproduction and presenting the first VQ-VAE based video generation model with minimal complexity, we stick with a single scale of discrete latents and transformers for the autoregressive priors, a design choice also adopted in DALL-E [203].

2.6 Conclusion

We have presented VideoGPT, a new video generation architecture adapting VQ-VAE and Transformer models typically used for image generation to the domain of videos with minimal modifications. We have shown that VideoGPT is able to synthesize videos that are competitive with state-of-the-art GAN based video generation models. We have also presented ablations on key design choices used in VideoGPT which we hope is useful for future design of architectures in video generation.

Acknowledgement

The work was in part supported by NSF NRI Grant #2024675 and by Berkeley Deep Drive.

Chapter 3

Temporally Consistent Transformers for Video Generation

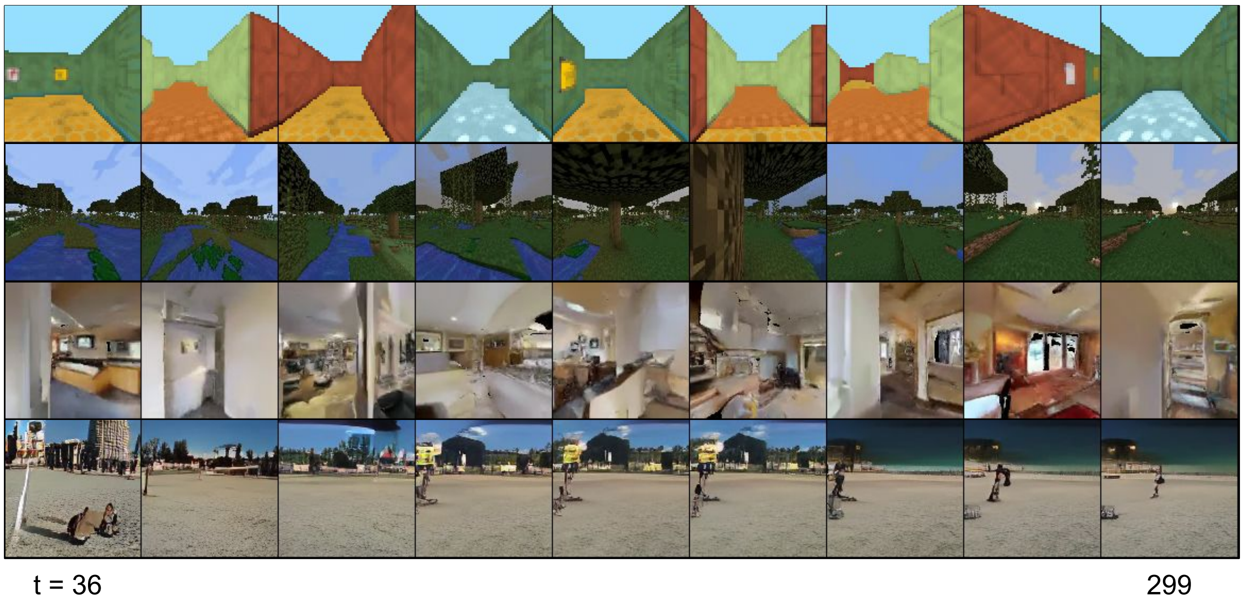


Figure 3.1: TECO generates sharp and consistent video predictions for hundreds of frames on challenging datasets. The figure shows evenly spaced frames of the 264 frame predictions, after being conditioned on 36 context frames. From top to bottom, the datasets are are DMLab, Minecraft, Habitat, and Kinetics-600.

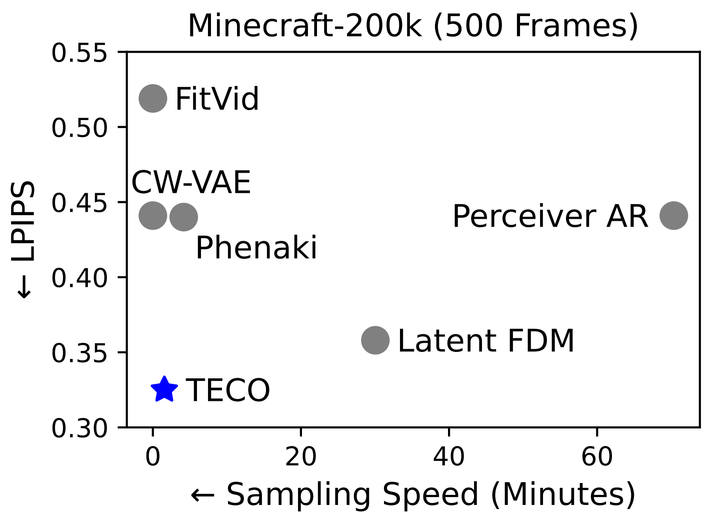


Figure 3.2: TECO generates temporally consistent videos of high fidelity (low LPIPS) over hundreds of frames while offering orders of magnitude faster sampling speed compared to previous video generation models.

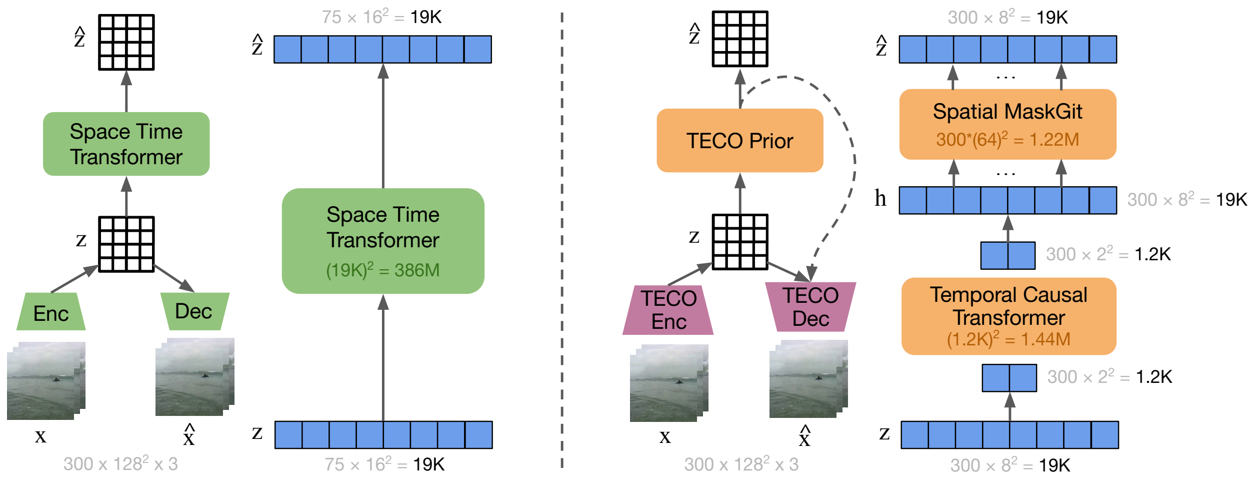
3.1 Introduction

Recent work in video generation has seen tremendous progress [104, 49, 293, 138, 75, 249, 164] in producing high-fidelity and diverse samples on complex video data, which can largely be attributed to a combination of increased computational resources and more compute efficient high-capacity neural architectures. However, much of this progress has focused on generating short videos, where models perform well by basing their predictions on only a handful of previous frames.

Video prediction models with short context windows can generate long videos in a sliding window fashion. While the resulting videos can look impressive at first sight, they lack temporal consistency. We would like models to predict temporally consistent videos — where the same content is generated if a camera pans back to a previously observed location. On the other hand, the model should imagine a new part of the scene for locations that have not yet been observed, and future predictions should remain consistent to this newly imagined part of the scene.

Prior work has investigated techniques for modeling long-term dependencies, such as temporal hierarchies [220] and strided sampling with frame-wise interpolation [75, 106]. Other methods train on sparse sets of frames selected out of long videos [93, 231, 49, 215, 302], or model videos via compressed representations [293, 200, 138, 224, 85, 273]. Refer to B.12 for more detailed discussion on related work.

Despite this progress, many methods still have difficulty scaling to datasets with many long-range dependencies. While Clockwork-VAE [220] trains on long sequences, it is limited by training time



(a) Space Time Transformer (b) Temporally Consistent Transformer (Ours)

Figure 3.3: The architectural design of TECO. (a) Prior work on video generation models over VQ codes adopt a single spatio-temporal transformer over all codes. This is prohibitive when scaling to long sequences due to the quadratic complexity of attention. (b) We propose a novel and efficient architecture that aggressively downsamples in space before feeding into a temporal transformer, and then expands back out with a spatial MaskGit that is separately applied per frame. In the figure, the transformer blocks show the number of attention links. On training sequences of 300 frames, TECO sees orders of magnitude more efficiency over existing models, allowing the use of larger models for a given compute budget.

(due to recurrence) and difficult to scale to complex data. On the other hand, transformer-based methods over latent spaces [293] scale poorly to long videos due to quadratic complexity in attention, with long videos containing tens of thousands of tokens. Methods that train on subsets of tokens are limited by truncated backpropagation through time [112, 199, 51] or naive temporal operations [94].

In addition, there generally do not exist benchmarks for properly evaluating temporal consistency in video generation methods, where prior works either focus on generating long videos where short-term dependencies are sufficient for accurate prediction [75, 231] and/or rely on metrics such as FVD [261] which are more sensitive to image fidelity rather than capture long-range temporal dependencies.

In this paper, we introduce a set of novel long-horizon video generation benchmarks, as well as corresponding evaluation metrics to better capture temporal consistency. In addition, we propose **Te**mporally **Co**nsistent Video Transformer (TECO), a vector-quantized latent dynamics model that effectively models long-term dependencies in a compact representation space using efficient transformers. The key contributions are summarized as follows:

- To better evaluate temporal consistency in video predictions, we propose 3 video datasets with long-range dependencies including metrics, generated from 3D scenes in DMLab [15], Minecraft [87], and Habitat [243, 219]
- We benchmark SOTA video generation models on the datasets and analyze capabilities of each in learning long-horizon dependencies.
- We introduce TECO, an efficient and scalable video generation model that learns compressed representations to allow for efficient training and generation. We show that TECO has strong performance on a variety of difficult video prediction tasks, and is able to leverage long-term temporal context to generate high quality videos with consistency while maintaining fast sampling speed.

3.2 Preliminaries

VQ-GAN

VQ-GAN [65, 266] is an autoencoder that learns to compress data into discrete latents, consisting of an encoder E, decoder G, codebook C, and discriminator D. Given an image $x \in \mathbb{R}^{H\times W\times 3}$, the encoder E maps x to its latent representation $h \in \mathbb{R}^{H'\times W'\times D}$, which is quantized by nearest neighbors lookup in a codebook of embeddings $C = \{e_i\}_{i=1}^K$ to produce $z \in \mathbb{R}^{H'\times W'\times D}$. z is fed through decoder G to reconstruct x. A straight-through estimator [18] is used to maintain gradient flow through the quantization step. The codebook optimizes the following loss:

$$\mathcal{L}_{\text{VQ}} = \|\text{sg}(h) - e\|_2^2 + \beta \, \|h - \text{sg}(e)\|_2^2 \tag{3.1}$$

where $\beta = 0.25$ is a hyperparameter, and e is the nearest-neighbors embedding from C. For reconstruction, VQ-GAN replaces the original ℓ_2 loss with a perceptual loss [309], $\mathcal{L}_{\text{LPIPS}}$. Finally, in order to encourage higher-fidelity samples, patch-level discriminator D is trained to classify between real and reconstructed images, with:

$$\mathcal{L}_{\text{GAN}} = \log D(x) + \log(1 - D(\hat{x})) \tag{3.2}$$

Overall, VQ-GAN optimizes the following loss:

$$\min_{E,G,C} \max_{D} \; \mathcal{L}_{\text{LPIPS}} + \mathcal{L}_{\text{VQ}} + \lambda \mathcal{L}_{\text{GAN}} \tag{3.3}$$

where $\lambda = \frac{\|\nabla_{G_L}\mathcal{L}_{\text{LPIPS}}\|_2}{\|\nabla_{G_L}\mathcal{L}_{\text{GAN}}\|_2 + \delta}$ is an adaptive weight, G_L is the last decoder layer, $\delta = 10^{-6}$, and $\mathcal{L}_{LPIPS}$ is the same distance metric described in Zhang et al. [309].

MaskGit

MaskGit [35] models distributions over discrete tokens, such as produced by a VQ-GAN. It generates images with competitive sample quality to autoregressive models at a fraction of the sampling cost by using a masked token prediction objective during training. Formally, we denote $z \in \mathbb{Z}^{H \times W}$ as the discrete latent tokens representing an image. For each training step, we uniformly sample $t \in [0, 1)$ and randomly generate a mask $m \in \{0, 1\}^{H \times W}$ with $N = \lceil \gamma HW \rceil$ masked values, where $\gamma = \cos\left(\frac{\pi}{2}t\right)$. Then, MaskGit learns to predict the masked tokens with the following objective

$$\mathcal{L}_{\text{mask}} = -\,\mathrm{E}_{z \in \mathcal{D}}\left[\log p(z \mid z \odot m)\right]. \tag{3.4}$$

During inference, because MaskGit has been trained to model any set of unconditional and conditional probabilities, we can sample any subset of tokens per sampling iteration. [35] introduces a confidence-based sampling mechanism whereas other work [140] proposes an iterative sample-and-revise approach.

3.3 TECO

We present **Te**mporally **Co**nsistent Video Transformer (TECO), a video generation model that more efficiently scales to training on longer horizon videos.

Architectural Overview

Our proposed framework is shown in Figure 3.3, where $x_{1:T}$ consists of a sequence of video frames. Our primary innovation centers around designing a more efficient architecture that can scale to long sequences. Prior SOTA methods [293, 75, 270] over VQ-codes all train a single spatio-temporal transformer to model every code, however, this becomes prohibitively expensive with sequences containing tens of thousands of tokens. On the other hand, these models have shown to be able to learn highly multi-modal distributions and scale well on complex video. As such, we design the TECO architecture with the intention to retain its high-capacity scaling properties, while ensuring orders of magnitude more efficient training and inference. In the following sections, we motivate each component for our model, with several specific design choices to ensure efficiency and scalability. TECO consists of four components:

$$\begin{aligned}
&\text{Encoder:} && z_t = E(x_t, x_{t-1}) \\
&\text{Temporal Transformer:} && h_t = H(z_{\leq t}) \\
&\text{Spatial MaskGit:} && p(z_t \mid h_{t-1}) \\
&\text{Decoder:} && p(x_t \mid z_t, h_{t-1})
\end{aligned} \tag{3.5}$$

Encoder We can achieve compressed representations by leveraging spatio-temporal redundancy in video data. To do this, we learn a CNN encoder $z_t = E(x_t, x_{t-1})$ which encodes the current frame x_t conditioned on the previous frame by channel-wise concatenating x_{t-1}, and then quantizes the output using codebook C to produce z_t. We apply the VQ loss defined in Equation (3.1) per timestep. In addition, we ℓ_2-normalize the codebook and embeddings to encourage higher codebook usage [297]. The first frame is concatenated with zeros and does not quantize z_1 to prevent information loss.

Temporal Transformer Compressed, discrete latents are more lossy and tend to require higher spatial resolutions compared to continuous latents. Therefore, before modeling temporal information, we apply a single strided convolution to downsample each discrete latent z_t, where visually simpler datasets allow for more downsampling and visually complex datasets require less downsampling. Afterwards, we learn a large transformer to model temporal dependencies, and then apply a transposed convolution to upsample our representation back to the original resolution of z_t. In summary, we use the following architecture:

$$h_t = H(z_{<t}) = \text{ConvT}(\text{Transformer}(\text{Conv}(z_{<t}))) \tag{3.6}$$

Decoder The decoder is an upsampling CNN that reconstructs $\hat{x}_t = D(z_t, h_t)$, where z_t can be interpreted as the posterior of timestep t, and h_t the output of the temporal transformer which summarizes information from previous timesteps. z_t and h_t are concatenated channel-wise and into the decoder. Together with the encoder, the decoder optimizes the following cross entropy reconstruction loss

$$\mathcal{L}_{\text{recon}} = -\tfrac{1}{T}\textstyle\sum_{t=1}^{T} \log p(x_t \mid z_t, h_t). \tag{3.7}$$

which encourages z_t features to encode relative information between frames since the temporal transformer output h_t aggregates information over time, learning more compressed codes for efficient modeling over longer sequences.

Spatial MaskGit Lastly, we use a MaskGit [35] to model the prior, $p(z_t \mid h_t)$. We show that using a MaskGit prior allows for not just faster but also higher quality sampling compared to an autoregressive prior. During every training iteration, we follow prior work to sample a random mask m_t and optimize

$$\mathcal{L}_{\text{prior}} = -\tfrac{1}{T}\textstyle\sum_{t=1}^{T} \log p(z_t \mid z_t \odot m_t). \tag{3.8}$$

where h_t is concatenated channel-wise with masked z_t to predict the masked tokens. During generation, we follow Lee et al. [140], where we initially generate each frame in chunks of 8 at a time and then go through 2 revise rounds of re-generating half the tokens each time.

Training Objective The final objective is the following:

$$\mathcal{L}_{\text{TECO}} = \mathcal{L}_{\text{VQ}} + \mathcal{L}_{\text{recon}} + \mathcal{L}_{\text{prior}} \tag{3.9}$$

DropLoss

We propose DropLoss, a simple trick to allow for more scalable and efficient training (Figure 3.4). Due to its architecture design, TECO can be separated into two components: (1) learning temporal representations, consisting of the encoder and the temporal transformer, and (2) predicting future frames, consisting of the dynamics prior and decoder. We can increase training efficiency by dropping out random timesteps that are not decoded and thus omitted from the reconstruction loss. For example, given a video of T frames, we compute h_t for all $t \in \{1, \dots, T\}$, and then compute the losses $\mathcal{L}_{\text{prior}}$ and $\mathcal{L}_{\text{recon}}$ for only 10% of the indices. Because random indices are selected each iteration, the model still needs to learn to accurately predict all timesteps. This reduces training costs significantly because the decoder and dynamics prior require non-trivial computations. DropLoss is applicable to both a wide class of architectures and to tasks beyond video prediction.

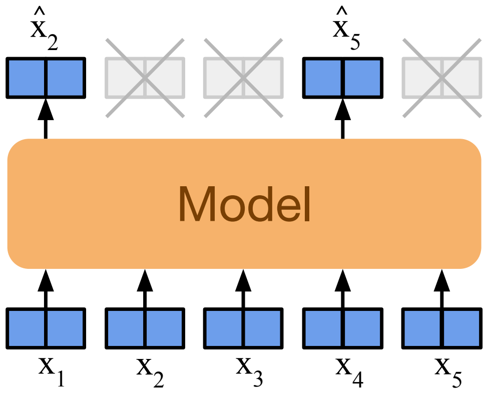


Figure 3.4: DropLoss improves training scalability on longer sequences by only computing the loss on a random subset of time indices for each training iteration. For TECO, we do not need to compute the decoder and MaskGit for dropped out timesteps.

3.4 Experiments

Datasets

We introduce three challenging video datasets to better measure long-range consistency in video prediction, centered around 3D environments in DMLab [15], Minecraft [87], and Habitat [219], with videos of agents randomly traversing scenes of varying difficulty. These datasets require video prediction models to re-produce observed parts of scenes, and newly generate unobserved parts. In contrast, many existing video benchmarks do not have strong long-range dependencies, where a model with limited context is sufficient. Refer to Appendix B.13 for further details on each dataset.

DMLab-40k DeepMind Lab is a simulator that procedurally generates random 3D mazes with random floor and wall textures. We generate 40k action-conditioned 64×64 videos of 300 frames of an agent randomly traversing 7×7 mazes by choosing random points in the maze and navigating to them via the shortest path. We train all models for both action-conditioned and unconditional prediction (by periodically masking out actions) to enable both types of generations. We further discuss the use cases of both action and unconditional models in Section 3.4.

Minecraft-200k This popular game features procedurally generated 3D worlds that contain complex terrain such as hills, forests, rivers, and lakes. We collect 200k action-conditioned videos of length 300 and resolution 128×128 in Minecraft's marsh biome. The player iterates between walking forward for a random number of steps and randomly rotating left or right, resulting in parts of the scene going out of view and coming back into view later. We train action-conditioned for all models for ease of interpreting and evaluating, though it is generally easy for video models to unconditionally learn these discrete actions.

Habitat-200k Habitat is a simulator for rendering trajectories through scans of real 3D scenes. We compile $\sim$1400 indoor scans from HM3D [201], MatterPort3D [34], and Gibson [289] to generate 200k action-conditioned videos of 300 frames at a resolution of 128×128 pixels. We use Habitat's in-built path traversal algorithm to construct action trajectories that move our agent between randomly sampled locations. Similar to DMLab, we train all video models to perform both unconditional and action-conditioned prediction.

Kinetics-600 Kinetics-600 [32] is a highly complex real-world video dataset, originally proposed for action recognition. The dataset contains $\sim$400k videos of varying length of up to 300 frames. We evaluate our method in the video prediction without actions (as they do not exist), generating 80 future frames conditioned on 20. In addition, we filter out videos shorter than 100 frames, leaving 392k videos that are split for training and evaluation. We use a resolution of 128×128 pixels. Although Kinetics-600 does not have many long-range dependencies, we evaluate our method on this dataset to show that it can scale to complex, natural video.

Baselines

We compare against SOTA baselines selected from several different families of models: latent-variable-based variational models, autoregressive likelihood models, and diffusion models. In addition, for efficiency, we train all models on VQ codes using a pretrained VQ-GAN for each dataset. For our diffusion baseline, we follow [211] and use a VAE instead of a VQ-GAN. Note that we do not have any GANs for our baselines, since to the best of our knowledge, there does not exist a GAN that trains on latent space instead of raw pixels, an important aspect for properly scaling to long video sequences.

Space-time Transformers We compare TECO to several variants of space-time transformers as depicted in Figure 3.3: VideoGPT [293] (autoregressive over space-time), Phenaki [270] (MaskGit over space-time full attention), MaskViT [85] (MaskGit over space-time axial attention), and Hourglass transformers [182] (hierarchical autoregressive over space-time). Note that we do not

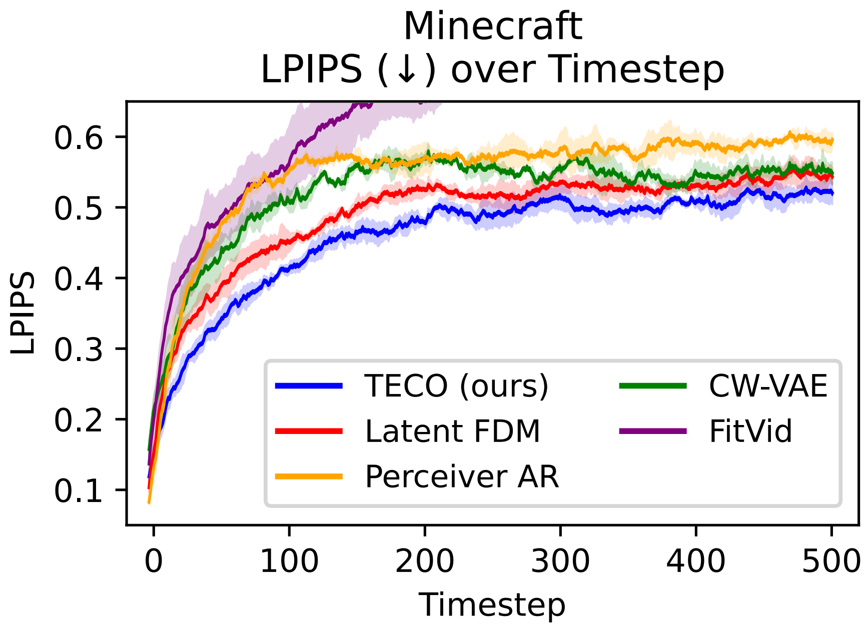


Figure 3.5: Quantitative comparisons between TECO and baseline methods in long-horizon temporal consistency, showing LPIPS between generated and ground-truth frames for each timestep. Timestep 0 corresponds to the first predicted frame (conditioning frames are not included in the plot). Our method is able to remain more temporally consistent over hundreds of timesteps of prediction compared to SOTA models.

Table 3.1: Quantitative evaluation on all four datasets. Detailed results in Appendix B.10.

	DMLab				Minecraft			
Method	FVD ↓	PSNR ↑	SSIM ↑	LPIPS ↓	FVD ↓	PSNR ↑	SSIM ↑	LPIPS ↓
FitVid	176	12.0	0.356	0.491	956	13.0	0.343	0.519
CW-VAE	125	12.6	0.372	0.465	397	13.4	0.338	0.441
Perceiver AR	96	11.2	0.304	0.487	**76**	13.2	0.323	0.441
Latent FDM	181	17.8	0.588	0.222	167	13.4	0.349	0.429
TECO (ours)	**48**	**21.9**	**0.703**	**0.157**	116	**15.4**	**0.381**	**0.340**

	Habitat				Kinetics-600			
Method	FVD ↓	PSNR ↑	SSIM ↑	LPIPS ↓	FVD ↓	PSNR ↑	SSIM ↑	LPIPS ↓
Perceiver AR	164	**12.8**	**0.405**	0.676	1022	13.4	0.310	0.404
Latent FDM	433	12.5	0.311	**0.582**	960	13.2	0.334	0.413
TECO (ours)	**73**	**12.8**	0.363	0.604	**799**	**13.8**	**0.341**	**0.381**

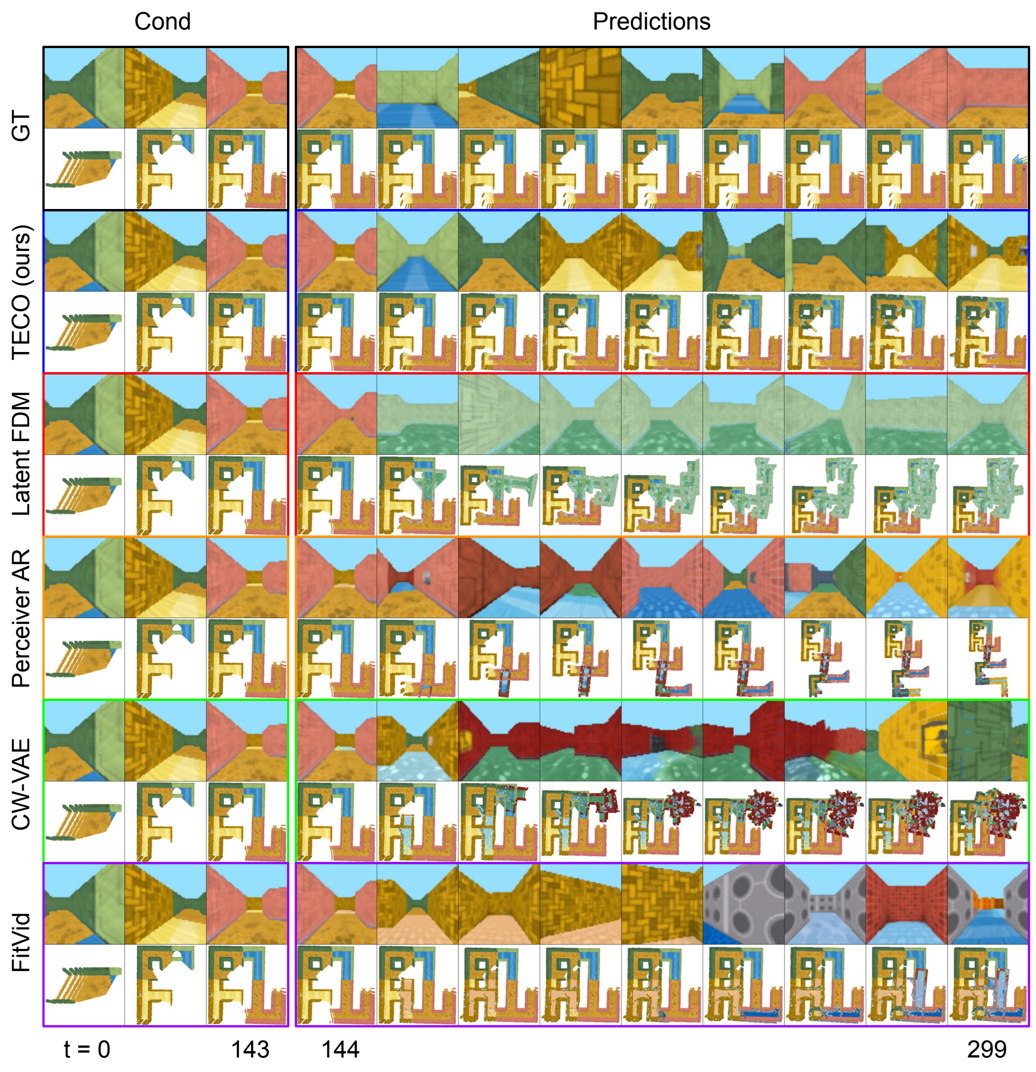


Figure 3.6: 3D visualization of predicted trajectories in DMLab for each model, generating 156 frames conditioned on 144. TECO is the only model that retain maze consistency with ground-truth, whereas baselines tend to extend out of the maze or create fictitious corridors that did not exist. **Video predictions use only the first-person RGB frames**. Refer to Appendix B.13 for more details on 3D evaluation. A video corresponding to this figure is available at: `https://wilson1yan.github.io/teco`.

include the text-conditioning for Phenaki as it is irrelevant in our case. We only evaluate these models on DMLab, as Table 3.2 and Table 3.1 show that Perceiver-AR (a space-time transformer with improvements specifically for learning long dependencies) is a stronger baseline.

FitVid FitVid [11] is a state-of-the-art variational video model based on CNNs and LSTMs that

Table 3.2: TECO substantially outperforms similar video generation models that use space-time transformers.

Model	FVD↓	PSNR↑	SSIM↑	LPIPS↓
TATS	156	11.1	0.296	0.468
Phenaki	725	11.0	0.202	0.474
MaskViT	76	12.4	0.360	0.435
Hourglass	110	11.7	0.335	0.458
TECO (ours)	**48**	**21.9**	**0.703**	**0.157**

scales to complex video by leveraging efficient architectural design choices in its encoder and decoder.

Clockwork VAE CW-VAE [220] is a variational video model that is designed to learn long-range dependencies through a hierarchy of latent variables with exponentially slower tick speeds for each new level.

Perceiver AR We use Perceiver AR [94] as our AR baseline over VQ-GAN discrete latents, which has been show to be an effective generative model that can efficiently incorporate long-range sequential dependencies. Conceptually, this baseline is similar to HARP [224] with a Perceiver AR as the prior instead of a sparse transformer [45]. We choose Perceiver AR over other autoregressive baselines such as VideoGPT [293] or TATS [75] primarily due to the prohibitive costs of transformers when applied to tens of thousands of tokens.

Latent FDM We train a Latent FDM model for our diffusion baseline. Although FDM [93] is originally trained on pixel observations, we also train in latent space for a more fair comparison with our method and other baselines, as training on long sequences in pixel space is too expensive. We follow LDM [211] to separately train an autoencoder to encode each frame into a set of continuous latents.

Experimental Setup

Training All models are trained for 1 million iterations under fixed compute budgets allocated for each dataset (measured in TPU-v3 days) on TPU-v3 instances ranging from v3-8 to v3-128 TPU pods (similar to 4 V100s to 64 V100s) with training times of roughly 3-5 days. For DMLab, Minecraft, and Habitat we train all models on full 300 frames videos, and 100 frames for Kinetics-600. Our VQ-GANs are trained on 8 A5000 GPUs, taking about 2-4 days for each dataset, and downsample all videos to 16×16 grids of discrete latents per frame regardless of original video resolution. More details on exact hyperparameters and compute budgets for each dataset can be found in Appendix B.14.

Evaluation

Standard methods for evaluating video prediction quality (FVD [261] or per-frame metrics PSNR [113], SSIM [282], and LPIPS [309]) do not measure long-consistency well. FVD is more sensitive to image fidelity, and relies on an I3D network trained on short Kinetics-600 clips. Evaluations using PSNR, SSIM, and LPIPS generally require sampling hundreds of futures and compare the sample that most accurately matches ground-truth. However, this does not align well with the goal of temporal consistency, as we would like the model to deterministically re-generate observed parts of the environment, and not accidentally generate the correct future after many samples.

Therefore, we propose a modified evaluation metric using PSNR, SSIM, and LPIPS that better measures temporal consistency in video generation by leveraging sufficient conditioning. Intuitively, if a video model is conditioned with enough information, future predictions should be approximately deterministic, meaning that only one sample should be needed to expect an accurate match with ground-truth. In the case of 3D environments, we can approximately make generation deterministic by conditioning on past frames (after the model has already seen most of the 3D environment) and actions (to remove stochasticity of movement). As such, for DMLab, Minecraft, and Habitat, we condition on 144 past frames as well as actions, and measure PSNR, SSIM, and LPIPS with 156 future ground-truth frames. However, note that per-frame metrics only capture temporal consistency, and do not capture a video model's ability to model the stochasticity of video data. Therefore, we also compute FVD on 300 frame videos, conditioned on 36 frames (264 predicted frames). For Kinetics-600, we evaluate FVD on 100 frame videos, conditioned on 20 frames (80 predicted frames). We compute all metrics over batches of 256 examples, averaged over 4 runs to make 1024 total samples.

Benchmark Results

DMLab & Minecraft Table 3.1 shows quantitative results on the DMLab and Minecraft datasets. TECO performs the best across all metrics for both datasets when training on the full 300 frame videos. Figure 3.6 shows sample trajectories and 3D visualizations of the generated DMLab mazes, where TECO is able to generate more consistent 3D mazes. For both datasets, CW-VAE, FitVid, and Perceiver AR can produce sharp predictions, but do not model long-horizon context well, with per-frame metrics sharply dropping as the future prediction horizon increases as seen in Figure B.10. Latent FDM has consistent predictions, but high FVD most likely due to FVD being sensitive to high frequency errors.

Habitat Table 3.1 shows results for our Habitat dataset. We only evaluate our strongest baselines, Perceiver AR and Latent FDM due to the need to implement model parallelism. Because of high complexity of Habitat videos, all models generally perform equally as bad in per-frame metrics. However, TECO has significantly better FVD. Qualitatively, Latent FDM quickly collapses to blurred predictions with poor sample quality, and Perceiver AR generates high quality frames, though less temporally consistent than TECO: agents in Habitat videos navigate to far points in the scene and back whereas Perceiver AR tends to generate samples where the agent constantly turns. TECO generates traversals of a scene that match the data distribution more closely.

Kinetics-600 Table 3.1 shows FVD for predicting 80 128×128 frames conditioned on 20 for Kinetics-600. Although Kinetics-600 does not have many long-range dependencies, we found that TECO is able to produce more stable generations that degrade slower by incorporating longer contexts. In contrast, Perceiver AR tends to degrade quickly, with Latent FDM performing in between.

Sampling Speed Figure 3.5 reports sampling speed for all models on Minecraft. We observed similar results for the different model sizes used on other datasets. FitVid and CW-VAE are both significantly faster that other methods, but have poor sample quality. On the other end, Perceiver AR and Latent FDM can produce high quality samples, but are 20-60x slower than TECO, which has comparably fast sampling speed while retaining high sample quality.

Ablations

In this section, we perform ablations on various architectural decisions of our model. For simplicity, we evaluate our methods on short sequences of 16 frames from Something-Something-v2 (SSv2), as it provides insight into scaling our method on complex real-world data more similar to Kinetics-600 while being computationally cheaper to run.

Details can be found in the Appendix, Table B.1. We demonstrate that using VQ-latent dynamics with a MaskGit prior outperforms other formulations for latent dynamics models such as variational methods. In addition, we show that conditional encodings learn better representations for video predictions. We also ablate the codebook size, showing that although there exists an optimal codebook size, it does not matter too much as along as there are not too many codes, which may the prior more difficult to learn. Lastly, we show the benefits of DropLoss, with up to 60% faster training and a minimal increase in FVD. The benefits are greater for longer sequences, and allow video models to better account for long horizon context with little cost in performance.

Further Insights

We highlight a few key experimental insights for designing long-horizon video generation models. Further details can be found in Appendix B.9 and Appendix B.7.

Trade-off between fidelity and learning long-range dependencies Given a network with fixed capacity, there exists an inherent trade-off between generating high fidelity and temporally consistent videos. We find that long-horizon information can be prioritized through bottlenecking representations, whereas allocating more computation towards higher resolution representations encourages higher fidelity. Due to TECO learning more compact representations, it achieves a better trade-off between fidelity and temporal consistency compared to our baseline models, demonstrated by better PSNR / SSIM / LPIPS, in addition to FVD.

Although frame quality saturates early-on, long-term consistency improves when training longer During training, we observe an interesting phenomenon where short-horizon metrics tend to saturate earlier on during training, while long-horizon metrics continue to improve until end of training. We hypothesize that this may be due to the likelihood objective, where modeling bits from neighboring frames is easier than learning long-horizon bits scattered throughout the video. This finding motivates the use of an efficient video architecture for TECO, which can be trained for more gradient steps given a fixed computational budget.

3.5 Discussion

We introduced TECO, an efficient video prediction model that leverages hundreds of frames of temporal context, as well as a comprehensive benchmark to evaluate long-horizon consistency. Our evaluation demonstrated that TECO accurately incorporates long-range context, outperforming SOTA baselines across a wide range of datasets. In addition, we introduce several difficult video datasets, which we hope make it easier to evaluate temporal consistency in future video prediction models. We identify several limitations as directions for future work:

- Although we show that PSNR, SSIM, and LPIPS can be reliable metrics to measure consistency when video models are properly conditioned, there remains room for better evaluation metrics that provide a reliable signal as the prediction horizon grows, since new parts of a scene that are generated are unlikely to correlate with ground truth.
- Our focus was on learning a compressed tokens and an expressive prior, which we combined with a simple full attention transformer as the sequence model. Leveraging prior work on efficient sequence models [47, 275, 305, 83, 94] would likely allow for further scaling.
- We trained all models on top of pretrained VQ-GAN codes to reduce the data dimensionality. This compression step lets us train on longer sequences at a cost of reconstruction error, which causes noticeable artifacts in Kinetics-600, such as corrupted text and incoherent faces. Although TECO can train directly on pixels, a ℓ_2 loss results in slightly blurry predictions. Training directly on pixels with diffusion or GAN losses would be promising.

3.6 Acknowledgements

This work was in part supported by Panasonic through BAIR Commons, Darpa RACER, the Hong Kong Centre for Logistics Robotics, and BMW. In addition, we thank teh TRC program (https://sites.research.google/trc/about/) and Cirrascale Cloud Services (https://cirrascale.com/) for providing compute resources.

Chapter 4

ElasticTok: Adaptive Tokenization for Image and Video

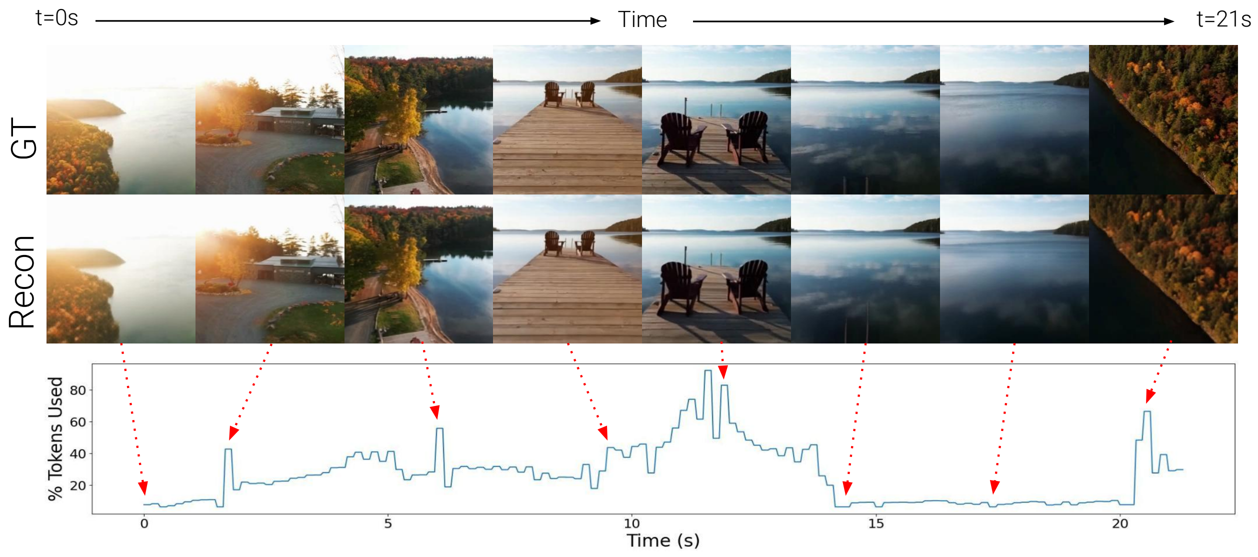


Figure 4.1: **ElasticTok adaptively represent video based on information available**. (Top) Ground-truth video frames. (Middle) Reconstructed frames with varying token usage. (Bottom) The bottom section depicts how ElasticTok dynamically adjusts token allocation over time, with the percentage of tokens used correlating to different content complexities in the video.

4.1 Introduction

Efficient video tokenization remains a key bottleneck in learning general purpose vision models that are capable of processing long video sequences – a crucial aspect towards developing intelligent agents for the visual world. Prevailing approaches [266, 65, 300, 293] are restricted to encoding videos to a fixed number of tokens irrespective of the visual content of the original video. As a result, being able to reliably encode with little information loss requires increasing the number of tokens to account for the worst case, highly complex visual inputs. However, this in turn causes many wasted tokens on simpler data, and leads to significant computational challenges for downstream training, where encoding a video or trajectory can require tens or even hundreds of millions of tokens, resulting in substantial computational costs and inefficiency [155, 27, 156]. On the other hand, using a too few number of tokens will result in lossy encodings, and fundamentally limit the capabilities of vision models when processing more complex visual data.

Intuitively, we would want do learn a model that adaptively encodes visual input in a data-dependent manner to variable length encodings, similar to how existing works in image and video compression [207, 148, 48, 41] compress data to a varying number of bytes. Taking inspiration from this, we introduce ElasticTok, a method that conditions on prior frames to adaptively encode a frame into a variable number of tokens. To enable this in a computationally scalable way, we propose a masking technique that drops a random number of tokens at the end of each frame's token encoding. During inference, ElasticTok can dynamically allocate tokens when needed – more complex data can leverage more tokens, while simpler data only needs a few tokens.

Our empirical evaluations demonstrate the effectiveness of ElasticTok, highlighted below.

- We show that ElasticTok can leverage adaptive tokenization to efficiently represent long videos with up to 2-5x fewer tokens.
- We show that ElasticTok enables flexibility in downstream vision-language tasks, allowing users to allocate tokens based on their compute budget.
- We show that ElasticTok allows leveraging different objectives during inference to adaptively trade off various semantic aspects of images.

4.2 Method

In this section, we present ElasticTok, a scalable adaptive visual tokenizer than can efficiently encode image and video. At a high level, we leverage a specialized random masking scheme when training any standard autoencoder to learn our elastic encodings. In the following sections, we provide a more detailed description of our model – first covering the simpler single block case (image / short video), and then moving on to the multi block case (longer video). An overview of our method is shown in Figure 4.2.

Training

First, we consider a single-block case in which we want to learn elastic codes over an image or a short video chunk (*e.g.*, 4 frames). Let x represent a sequence of input frames, where images are considered as 1 frame videos. Our proposed method extends upon any existing variants of autoencoders (VQ-VAE [266], FSQ [172], VAE [130]) by incorporating additional masking of the encoder tokens.

For each training input x, we first uniformly sample the number of tokens to keep $\ell \sim U(\{M_{min}, \dots, M_{max}\})$, where M_{min} and M_{max} define the supported range of encoding lengths. M_{max} is generally set as N, the maximum number of tokens the encoder can output (*e.g.*, 2048 or 4096), and M_{min} a lower bound such as 128 or 256. We found this lower bound necessary as we found that sampling too few tokens could destabilize training. After sampling ℓ, we then compute its mask m, defined as a binary vector of length N with the first ℓ values set to 1. Input x and mask m are then fed into the encoder $E(x, m)$ to compute z. z is masked as $z_m = z \odot m$, and fed through the decoder $D(z_m)$ to produce $\hat{x}$. Finally, our model optimizes a reconstruction loss, with potentially auxiliary losses depending on the exact autoencoder being used (*e.g.*, KL for VAE or VQ loss for VQ-VAE). Although our method is generally architecture invariant, we opt to use ViTs [62] as our encoder and decoders for simplicity.

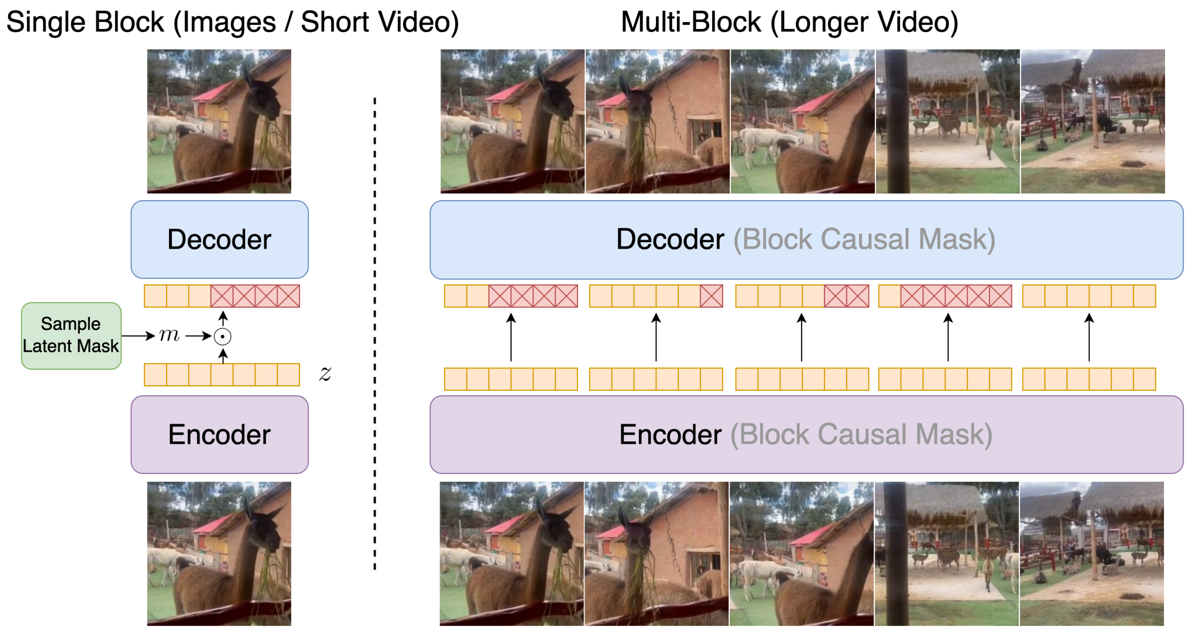

Single Block (Images / Short Video)
Multi-Block (Longer Video)
Decoder
Sample Latent Mask
m
$\odot$
z
Encoder
Decoder (Block Causal Mask)
Encoder (Block Causal Mask)

Figure 4.2: **ElasticTok adaptively encodes image and video to variable length outputs based on the complexity of the input data**. Single block uses an Encoder-Decoder pipeline with a sampled latent mask. Multi-block extends this with a Block Causal Mask to handle longer video sequences.

We can extend our method to process longer video by breaking up video into blocks, with sizes defined by the number of frames used for the single block model (e.g 4 frames per block). The training process remains the same as the single block case, with two key differences: (1) masks for each multi block video sequence are sampled independently for each block, and (2) a block causal mask (block size N) is used in the encoder and decoder. The number of blocks is constrained only by the transformer's context size. We utilize a prior long-context method [156] to train ElasticTok on longer videos. Architecturally, there are no added parameters which we found useful when progressively training our model from single block to multi block.

Exact details of our model's forward pass are described in Algorithm 1.

Inference

There are two primary ways to use ElasticTok for inference – by specify a target encoding length, or a target reconstruction threshold.

Target encoding lengths. This method is simple to implement, as it only requires generating the correct mask and running the input through the encoder. However, although such inference is simple and efficient, it is difficult from a user's perspective due to lack of knowledge of how many tokens to specify.

Target reconstruction threshold. This method presents a more intuitive inference approach that will automatically adaptively allocate tokens between different inputs based on a specified reconstruction threshold (*e.g.*, target pixel-wise MSE loss). Visual content that is easy to reconstruct may require fewer tokens to meet the threshold whereas more complex inputs may require more tokens. Doing so requires a search process over different encoding lengths to determine lowest encoding length that still satisfies the given threshold.

We consider a few different search algorithms, detailed below:

- **Full Search**: Exhaustive search over every possible number of tokens lengths, and treat the result as ground truth. We use our discrete latent with max 4096 tokens.
- **Binned Search (100)**: Similar to Full Search, but only evaluate on 100 uniformly spaced token lengths.
- **Binary Search**: We perform search using binary search, where the token length is the ”array index”, and the evaluated reconstruction loss is the ”array value.” Note that this assumes that reconstruction loss is monotonic with respect to token length, which is generally close to true, but not fully accurate, hence error in the result produced.
- **Neural Regression**: We collect 100k examples of paired (image / video, token length) data, and finetune our autoencoder to directly regress the number of tokens given an image or video.

Each search algorithm has its own trade-off between accuracy and efficiency – we further examine this relationship in Section 4.3.

In the multi block case, we iteratively perform the search process for each block in a block autoregressive manner, using caching similar to in language models. Algorithm 2 provides more details on the exact inference process.

Algorithm 1 Training

```
Required: Input video $x$. Tokens per block $B$.
Required: Encoder $E$. Decoder $D$
$x \leftarrow$ PatchifyRearrange($x$) // $B \times L \times D$
$N_b \leftarrow L/B$
for $i$ in $\{0, \ldots, N_b - 1\}$ do
    Sample $\ell_i \sim U(\{M_{min}, \ldots, M_{max}\})$
    Initialize masks $m_i \leftarrow \mathbf{0}_B$
    Fill masks $m_i[:\ell_i] = 1$
end for
Concatenate masks $m \leftarrow (m_0, \ldots, m_{N_b-1})$
Encode $z \leftarrow E(x, m)$ // $B \times L \times D_z$
Mask out along sequence length $z_m \leftarrow z \odot m$
Decode $\hat{x} \leftarrow D(z_m)$
Compute loss $\mathcal{L}(\hat{x}, x)$ // e.g., MSE + LPIPS
```

Algorithm 2 Inference

```
Required: Input sequence $x$. Tokens per block $B$.

Required: Encoder $E$. Decoder $D$
Required: Target reconstruction threshold $t$
$x \leftarrow$ PatchifyRearrange($x$) // $B \times L \times D$
$N_b \leftarrow L/B$
Initialize KV cache $c$ of length $L$
for $i$ in $\{0, \ldots, N_b - 1\}$ do
    // See Section 2.2 for search algorithms
    $\ell_i, m_i, c \leftarrow$ SearchAlgo($x_{iB:(i+1)B}, c, t, D, E$)

    Encode $z_i \leftarrow E(x_{iB:(i+1)B}, m_i, c) \odot m_i$
end for
Return $z \leftarrow (z_0, \ldots, z_{N_b-1})$
```

4.3 Experiments

In this section, we introduce our evaluation setup and present the results of pretraining ElasticTok to adaptively represent images and videos, as well as its performance on downstream tasks.

Experimental Setup

Model details. We train 200M parameter discrete and continuous autoencoders on images and video sequences. For learning discrete representations, we use FSQ [171], while for continuous representations, we use a VAE [130]. Both models employ ViT encoders and decoders. Similar to prior works [65], we optimize both a pixel-wise reconstruction loss (MSE) and a perceptual loss (LPIPS) [119]. Notably, we do not use a GAN [81], as we find it more difficult to stably train when progressively extending to much longer sequences (e.g. 1000+ frames), which we leave to future work, as it is not the main focus of this paper.

Training details. All models are jointly trained on 256×256 images and 24 FPS videos. Each video block consists of 4 frames, and joint training is conducted by treating images as single-frame videos, replicating each image 4 times to match the block size. Each block is encoded into 2048 or 4096 tokens by the continuous and discrete autoencoders, respectively. Following prior research in LWM [155] on scaling sequence length, we begin by training our models on single-block cases (images and short videos) and progressively extend the context length to cover up to 40-second videos (512K to 1M tokens). We train our long video models using v4-512 TPUs from Google Cloud on the COYO-700M image dataset and a custom dataset consisting of 6M videos scraped from the web. We additionally train an image-only model on ImageNet using v4-256 TPUs. Details for further training details can be find in Appendix C.2, and data details in Appendix C.3

Baselines. We compare our proposed method against fixed token baselines trained at a varying token capacities. The architecture, training processes, and total FLOPs are exactly the same as our method with the only difference being restricted to one fixed mask, instead of sampling variable masks.

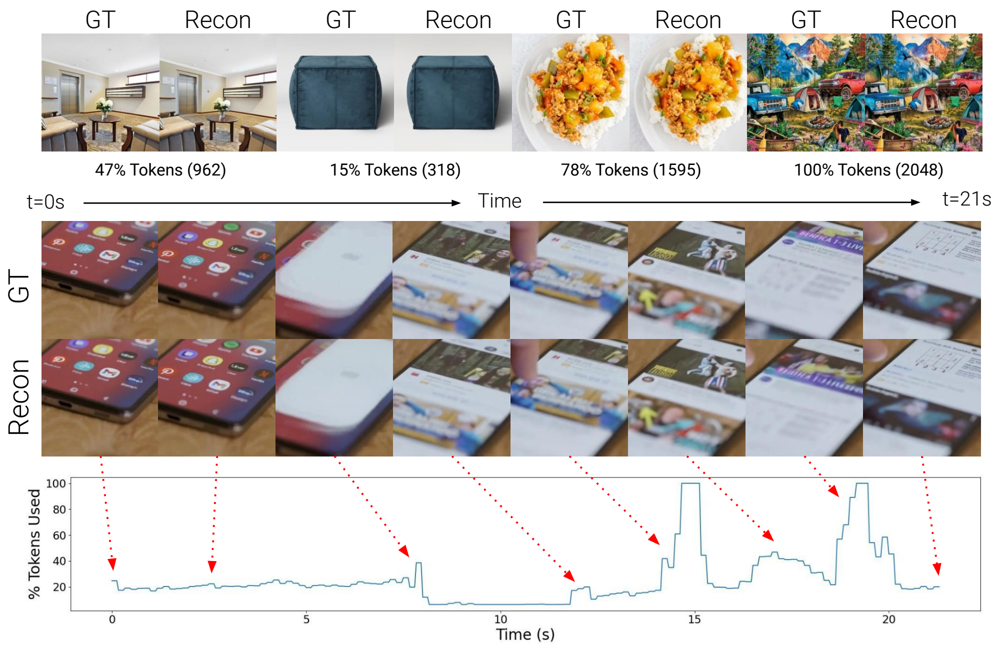


Figure 4.3: **ElasticTok adaptively encodes image and video to variable length outputs based on the complexity of the input data**. (Top) Ground-truth video frames. (Middle) Reconstructed frames with varying token usage. (Bottom) The bottom section depicts how ElasticTok dynamically adjusts token allocation over time, with the percentage of tokens used correlating to different content complexities in the video.

Main Results

Intuitively, the advantages of our elastic tokenization allow variable length codes that can depend on the visual content of a given image or video. Figure 4.3 shows some qualitative examples that demonstrate this. All images and videos are provided the same MSE reconstruction quality (0.003) threshold to satisfy, and require different number of tokens depending on the complexity the images. Simpler images such as the blue cushion require fewer tokens, while more complex images such as the painting or recipe have details that require more tokens to properly encode. For the provided video example, we generally see larger token usage spikes in the event of a scene change or faster motion, such as when the phone animates a transition screen, or when the finger scrolls up in the newsfeed. More qualitative examples can be found at: largeworldmodel.github.io/elastictok.

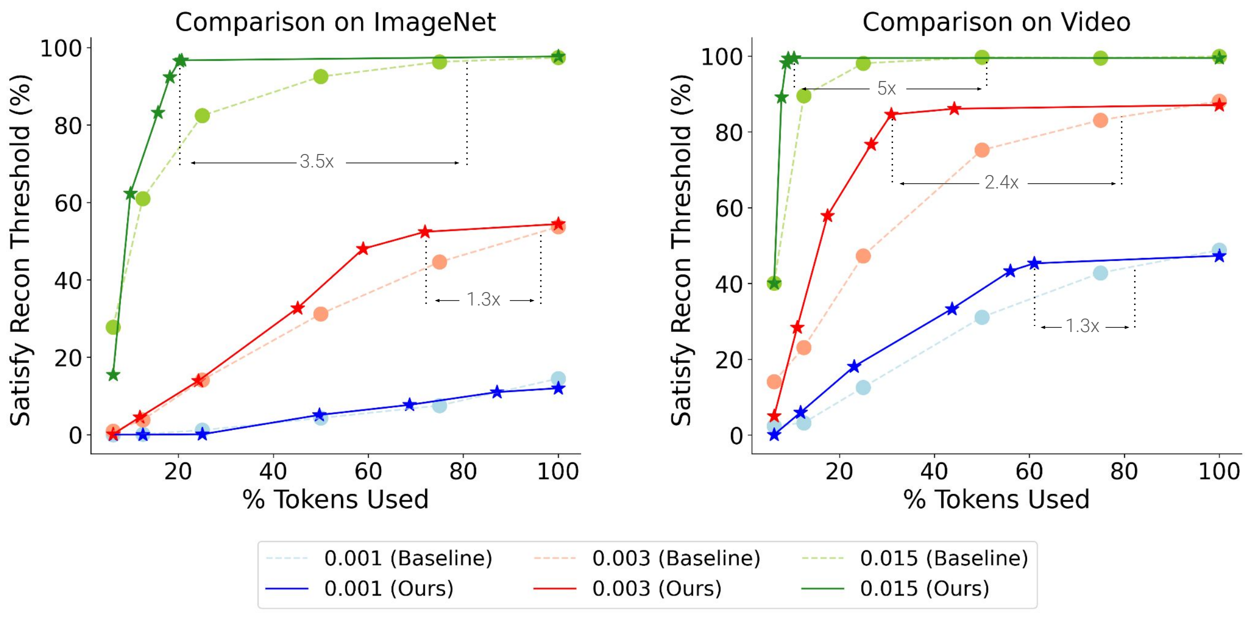


Figure 4.4: **Performance comparison between baseline and ElasticTok on ImageNet and Video**. The y-axis shows the percentage of samples that satisfy the reconstruction threshold, while the x-axis represents the percentage of tokens used. (Left) On image, ElasticTok achieves a 3.5x and 1.3x efficiency boost at different reconstruction thresholds. (Right) On video, ElasticTok shows a 5x and 2.4x improvement over the baseline, maintaining superior performance while using fewer tokens. Figure C.1 in Appendix C.4 shows reference examples of reconstruction quality for an image at different thresholds.

Figure 4.4 shows quantitative comparisons between our method and different fixed token baselines trained at each token length. In order to show the benefits of elastic token representations, we compute a quantitative metric that measures the percentage of images or videos blocks in which a model satisfies a specified reconstruction (MSE) threshold. As shown, our method can leverage its elastic representations to achieve similar reconstruction satisfactory percentages compared to baselines while using 1.3x - 5x fewer tokens, depending on the given threshold. Generally, more lax (0.015) reconstruction thresholds present larger performance improvements due to more room to use fewer than max tokens. In contrast, more strict (0.001) thresholds usually almost always require all tokens. Different thresholds may be useful for different cases, where more lossy encodings can be used for simple VQA tasks, while more accurate reconstructions are useful for tasks such as image / video generation. Table 4.1 shows similar results for the continuous variant of our model, compared against a single baseline model fixed to 50% token usage.

Table 4.2 shows results comparing ElasticTok against a quality baseline. In which we first run inference using our model with a given threshold, and then compute which token length baseline is needed to match the quality (worst-case MSE over video blocks) of our model. Note that our model can use a different number of tokens each video block, where as the baseline uses the same number of tokens for all blocks. The adaptivity of our model helps match baseline model quality with lower average token usage.

Figure 4.5 shows qualitative examples of progressive reconstruction quality as we increase the number of tokens. Different visual inputs will saturate reconstruction quality at a different number of tokens. This flexibility allows us to use a large number of tokens for very hard images (*e.g.*inputs with detailed fine-grained text), while saving tokens on inputs that are easier to reconstruct. Figure 4.6 shows the performance of our method as we increase the sequence length of the model – performing better as sequence length increases due to being able to leverage long context for more accurate reconstruction. However, performance degrades at the max context length (1M tokens), which we hypothesize is due to not enough training due to our relatively limited compute budget for that context size.

Takeaway: ElasticTok offers significant advantages in image and video reconstruction by using variable-length codes that depend on content complexity. Qualitatively, simpler images require fewer tokens, while more complex ones need more tokens for accurate encoding. Quantitatively, elastic tokenization achieves similar reconstruction quality compared to fixed token baselines while using up to 5x fewer tokens

Downstream Tasks

In order to evaluate the quality of our learned representations, we finetune a pretrained language model with our visual tokens, and evaluate on text-image and text-video VQA tasks. We use OpenLLaMA-3B [76], and finetune with visual tokens from the continuous variant of our model (max 2048 tokens). We finetune the entire model for 80M text-image pairs from COYO-700m [30], and chat finetune with data from [38]. For video, we continue to train on WebVid10M [12] and finetune on Video-ChatGPT [166] instruct data. Figure 4.7 shows evaluation results on GQA [5], POPE [151], MSVD-QA [291], and MSRVTT-QA [291] at a varying number of input inference

Table 4.1: Performance comparison between baseline and ElasticTok across different reconstruction thresholds with the continuous token model. Percentage shows the proportion of video blocks that satisfy the given reconstruction threshold. Our model can satisfy **more** video blocks.

	Threshold		
	0.001	0.003	0.015
Baseline	40%	78%	99%
Ours	**50%**	**88%**	**99%**

Table 4.2: Comparing performance of our Quality Baseline and ElasticTok at different reconstruction thresholds. Our model can use **fewer** tokens to reach the same quality (match worse-case MSE) as the baseline model.

	Threshold		
	0.001	0.003	0.015
Baseline	78%	57%	15%
Ours	**75%**	**37%**	**9.4%**

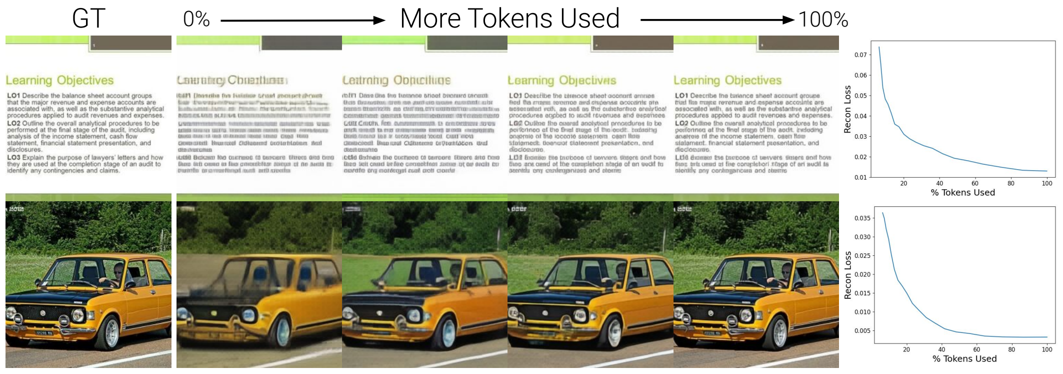


Figure 4.5: **Loss progressively declines as more tokens are used**. The top row illustrates the impact on text clarity, while the bottom row shows the effect on image sharpness. The graphs on the right quantify the reconstruction loss relative to token usage percentage, showing a rapid decline as more tokens are consumed.

tokens. Benchmark performance generally increases at the number inference tokens increases, suggesting the usefulness of our model as a means for users to potentially be able to flexibly choose how many tokens use, as a compute / accuracy trade-off, especially useful for users with more limited compute / API call budgets. Lastly, Table 4.3 shows that our VLM finetuned on our adaptive tokens can match the performance of a VLM finetuned on a fixed token (50% tokens) baseline tokenizer, suggesting that there is little to no loss in accuracy when switching to ElasticTok as the tokenizer, with additional added adaptivity benefits.

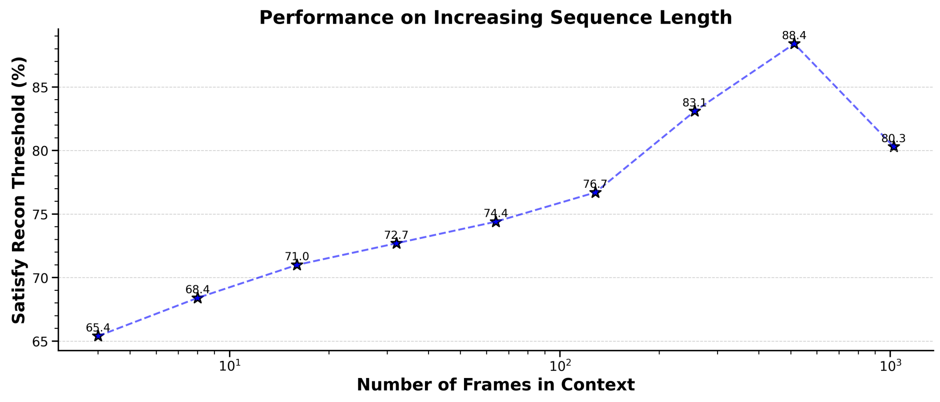


Figure 4.6: **Progressive performance increase with more frames**. Performance improves with increasing sequence length, peaking around 100 frames before a slight decline. (Note the log scale for the x-axis)

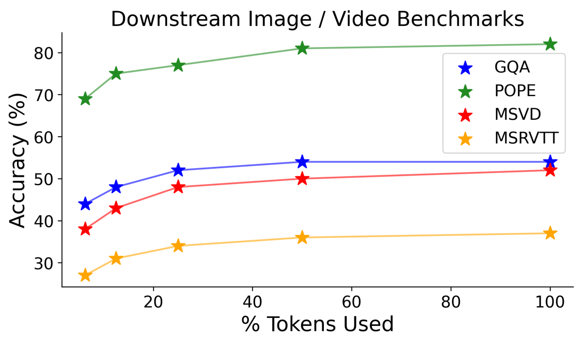


Figure 4.7: **The accuracy and compute trade-off with varying percentages of tokens used**. This allows users to adjust the accuracy based on computational budget.

	GQA	POPE	MSVD	MSRVTT
Ours	54%	82%	52%	37%
Baseline	54%	82%	53%	37%

Table 4.3: **Comparison of our method with baseline on image and video benchmarks**. Our method can match the performance of the baseline trained on a fixed number (100%) of tokens. However, baseline models are restricted to a fixed token output, and require full pretraining a new model for every possible token length, whereas ElasticTok only requires a single model to generalize to all token lengths.

Inference

Speed. Table 4.4 shows comparisons of the accuracy and inference speed (NFEs) for each of these methods. Error rate is computed as the relative error between the ground truth number of tokens used, and the number of tokens returned by each method. NFE (Number of Function Evaluations) is the number of forward passes that are needed. In generally, methods closer to exhaustive search (Full / Binned Search) are more accurate, while methods that orders of magnitudes faster generally have much higher error around $5-10\%$. Users with less compute available for inference may want to use the faster inference methods at a slight cost to token encoding accuracy. Users with a lot of compute can leverage more exhaustive approaches while retaining fast inference speed by computing search for token lengths as parallel batches.

Target Objective. One benefit of computing elastic tokens is that we can switch to any target objective during inference, and can adaptively tokenize visual content based on certain visual preferences, or aspects that users want to prioritize by allocating more tokens to. For example, Figure 4.8 shows qualitative examples comparing running inference using an MSE objective versus using a CLIP loss (image-image cosine distance) On average, thresholds are set so that both models as the same average number of tokens over the dataset, but OCR capabilities of CLIP allow it to show preference towards allocating more tokens in reconstructing the text (bottom images), while deprioritizing other images such as the fine-grained details in the bottom image. In general, any scalar function is usable, and does not need to be differentiable (*e.g.*, running OCR detection, and computing a more direct text reconstruction metric, or running segmentation and priortizing object clarity).

Takeaway: ElasticTok allows multiple adaptive inference methods: full search is the most accurate but slow, while faster methods like neural regression slightly sacrifice accuracy (5-10% error) for speed. Users with less compute can opt for faster methods with a slight accuracy loss. Additionally, inference objectives can be adapted to prioritize specific content, such as focusing on text over other image features, allowing for flexible token allocation based on user preferences.

Inference Method	**Error Rate**	**NFEs**
Full Search	0%	4096
Binned Search (100)	0.5%	100
Binary Search	7%	12
Neural Regression	9%	1

Table 4.4: Comparison of inference methods showing their respective error rates, number of function evaluations (NFEs). Note that while Full and Binned Search are more computationally expensive, they could also benefit more from parallel function evaluations if compute is available.

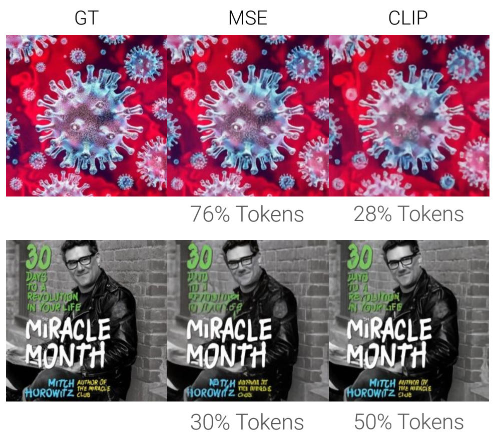


Figure 4.8: **Comparison of different loss functions for inference**. Inference with a CLIP model (cosine distance) prioritizes textual reconstruction (bottom) while deprioritizing other detailed visual features (top).

Frequency Analysis

We hypothesize that visual content that tends to have more fine-grained details, and high-frequncy features tends to use a higher number of tokens. To investigate this, we run inference on 2k videos (5s, 32 blocks), and also compute an approximate frequency metric for each video block. For each frame in a video block, we convert it to grayscale, run a Sobel edge detection filter (horizontal / vertical), and compute average gradient magnitudes. Figure 4.9 shows scatter plots and correlation for the single and multi block cases. For the single block case, the correlation between encoding length and amount of high-frequency detail is quite high (0.77). The multi block case is lower (0.67), which is most likely due to slight decorrelation from being able to leverage past frames (conditional encoding) in videos.

Takeaway: Pearson correlation shows that ElasticTok adaptively allocate number of tokens for videos with more detailed visual content and high-frequency features require more tokens for encoding.

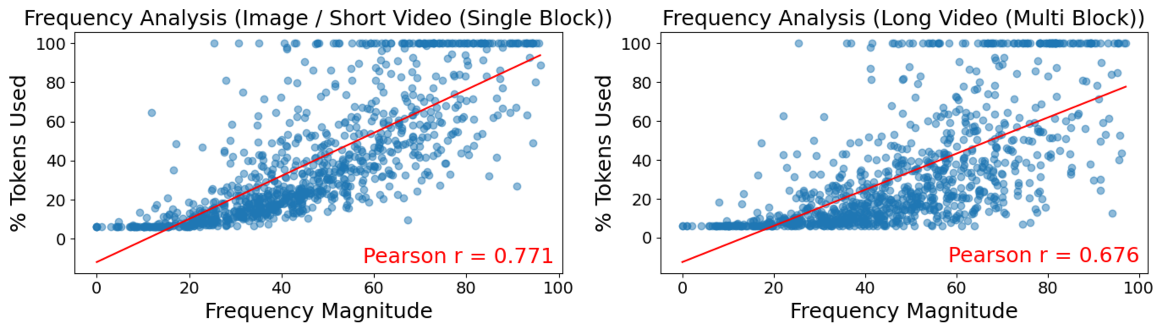


Figure 4.9: **Comparison of token usage versus frequency magnitude in single-block and multi-block frequency analysis**. Both scatter plots show a strong positive correlation between frequency magnitude and token usage in a single-block setting a multi-block setting. The red lines represent the linear regression fits for each case.

4.4 Related Work

Adaptive Representation. There have been a lot of research studies on learning adaptive or ordered representations. Early work on nested dropout [209] proposes using dropout [239] in the context of autoencoders to learn ordered representations. In this approach, an index is sampled from a prior distribution, and all units with an index larger than the sampled one are dropped. Similar to our ElasticTok , this method imposes an inherent ordering on the representation dimensions. Units that are dropped less frequently encode more important information, while those that are dropped more often encode less critical details. Other works study adaptive architectures based on context size [128], slimmable neural networks that train a network by sampling multiple sub-networks of different channel numbers simultaneously, where the weights are shared among different widths [141, 296], adaptive width and depth in BERT [108], or dropping random layers during training to increase robustness to pruning [69, 109]. Dieleman et al. [57] learns a variable-rate representation on audio applied to speech. Transframer [181] uses variable-length DCT representations with transformers for general image and frame-level video prediction tasks. Matryoshka representation learning [135] explores adding nested substructures inside the standard Transformer block. However, none of the previous works consider learning elastic representations in autoencoders for long sequences like videos. Our work provides a scalable solution for representing videos with elastic representations.

Tokenization. Representing visual images with a set of discrete tokens has been extensively studied, such as in VQVAE [266, 205] and VQGAN [297]. Recent research has highlighted several shortcomings of traditional tokenization methods, including low vocabulary utilization. In response, alternative approaches like FSQ have been explored [172, 301]. These discrete visual tokens facilitate integration with next-token prediction in language models and multimodal models [301, 155, 290, 279]. Parallel to this, other studies have investigated the use of continuous embeddings for representing images and videos [153, 158, 312]. Other research proposes next-scale prediction [248], which employs hierarchical multi-scale modeling to first capture coarse details, followed by finer ones. However, this approach requires a manually defined hierarchy and uses a fixed number of tokens. Our work, while complementary to these efforts, focuses on adaptive representation and can be directly applied to both discrete and continuous approaches.

Compression. Adaptive learning for images and video have also been well-studied in the context of variable-rate compression. JPEG [48] remains one of the most popular lossy image compression algorithms in the world, using a combination of DCT with quantization followed by entropy coding. For video, codecs such as H264 [285], H265 [242], VP9 [179], and AV1 [92] are popularly used compression algorithms. Prior works have additionally explored extending neural networks to learn more effective compressors. Theis et al. [247] and Minnen et al. [173] introduce methods that leverage CNNs to learn effective variable bit-rate compression algorithms over images competitive with JPEG. Li et al. [148], Mentzer et al. [170], and Li et al. [147] extend neural networks to learn how to effectively compress videos. While similar to our work in that these methods also learn adaptive encodings, these compression methods are more difficult to utilize for downstream training and generation tasks due to lack of direct compatibility. In contrast, our work builds upon existing tokenization strategies (VAE, FSQ) that have shown to work well for such downstream tasks, as well as take advantage of the adaptive representations to better utilize compute.

4.5 Discussion and Conclusion

In this work, we propose an elastic representation approach address the inefficiencies of traditional video encoding approaches by introducing an adaptive encoding method that selectively encodes new information based on the context of previous frames. By dynamically allocating resources, it reduces computational costs while maintaining high-quality video representation. The proposed technique of dropping tokens at the end of each sequence allows the model to prioritize essential information, ensuring scalability and efficiency during inference. Our ElasticTok method demonstrates strong performance in both video representation and downstream tasks.

Lastly, we identify some limitations of our method, as well as possible directions for future work.

- **Masking schemes**: Our model currently slightly underperforms baselines at the tail ends of encoding lengths, which we hypothesize may be partially due to conflicting representations that the model needs to learn in each case (low frequency, global encodings vs high frequency, local encodings). Preliminary investigations showed that changing the way tokens are distributed (as opposed to our method of left-aligned tokens) improved reconstruction accuracy. Future work may also explore learnable encoding masks to dynamically select which tokens to keep for each input.
- **Other temporal modalities**: Although our primary focus was on image and video, our method is generally modality-agnostic and can be extended to any temporal data, such as audio and decision-making trajectories (e.g., state, action, reward).
- **Conditional encoding and different training objectives**: Our focus was on leveraging temporal encoding to achieve more efficient tokenization while retaining high reconstructed visual quality. However, our method can be explored more broadly as a means of encouraging meaningful lossy encodings. For example, robotics models can learn text-conditioned visual encoders that capture only task-relevant information from an image. In other scenarios, different reconstruction objectives could prioritize facial or textual reconstruction while ignoring background reconstruction accuracy.

Acknowledgments

We gratefully acknowledge the Google TPU Research Cloud (TRC) program for providing access to TPUs. We extend our thanks to Xinyang Geng, Jimmy Shi, Tim Rocktäschel, Joao Carreira, and Satinder Singh for their valuable discussions and insightful feedback on an earlier version of this paper, and to Doina Precup for valuable support of this work. Pieter Abbeel holds concurrent appointments as a Professor at UC Berkeley and as an Amazon Scholar. This paper describes work performed at UC Berkeley and is not associated with Amazon.

Chapter 5

World Model on Million-Length Video and Language With Blockwise RingAttention

5.1 Introduction

Enabling long-context understanding remains a key challenge in scaling existing sequence models—a crucial step toward developing generally intelligent models that can process and operate over extended temporal horizons, potentially involving millions of tokens. Current modeling approaches are predominantly limited to processing short sequences, whether in the form of language, images, or video clips [28, 253, 254, 186, 27, 245]. As a result, these models fall short when tasked with understanding complex, long-form language and visual contexts.

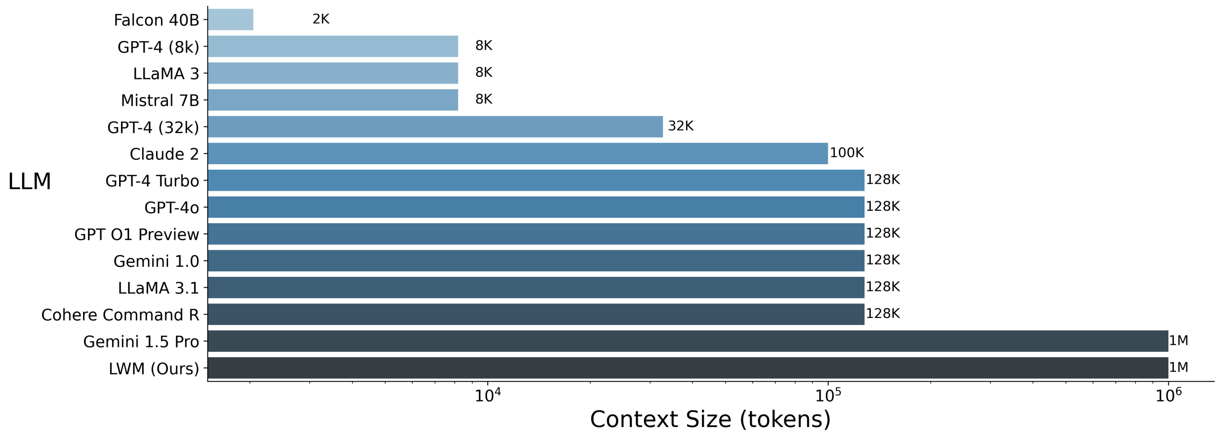


Figure 5.1: **Comparison of context size in state-of-the-art LLMs.** Our model and concurrent work Gemini 1.5 both achieve a 1M context size, significantly outperforming other LLMs.

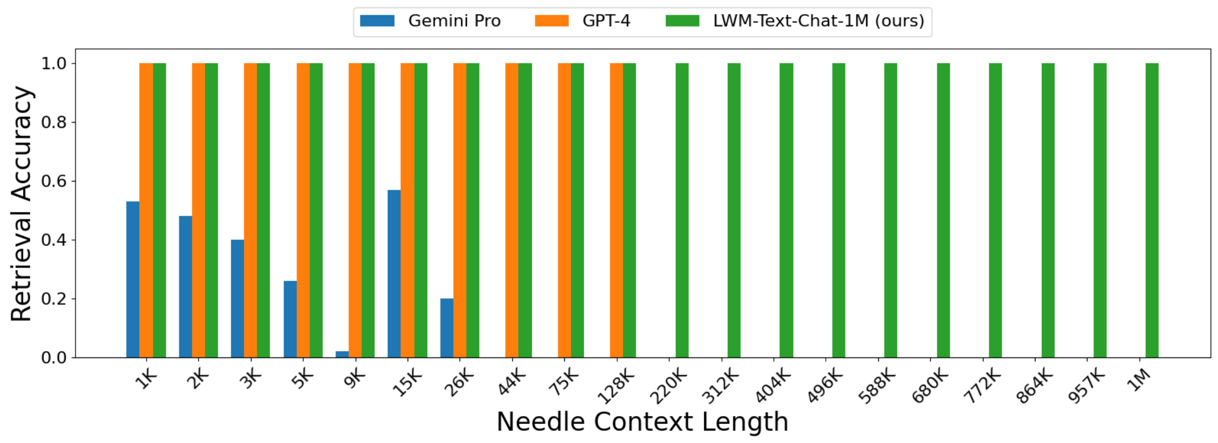


Figure 5.2: **Retrieval comparisons against Gemini Pro and GPT-4.** Needle retrieval comparisons against Gemini Pro and GPT-4 for each respective max context length – 32K and 128K. Our model performs competitively while being able to extend to 8x longer context length. Note that in order to show fine-grained results, the x-axis is log-scale from 0-128K, and linear-scale from 128K-1M.

However, training models to process sequences that exceed millions of tokens is a significant challenge due to the high memory and computational costs, as well as the lack of long-context data. In this work, we address these challenges by leveraging Blockwise RingAttention [156, 154], a technique that scales context size without approximations or overheads, enabling efficient training on long sequences. We curate an extensive dataset of long-form videos and books from public sources, covering a wide variety of activities and narrative structures. To address the scarcity of long-form conversational datasets, we developed a model-based question-answering technique, where a short-context model generates training data from books, significantly enhancing the model's chat capabilities over long sequences. To mitigate computational costs, we gradually extended context size from an initial 4K tokens to 1M tokens, achieving a cost-effective and scalable approach for long-context modeling.

Following this, we further train our long-context language model to incorporate visual modalities, such as image and video. Contrary to existing popular vision-language models [157, 186, 38], we opt to additionally optimize next-token prediction losses for image and video (generation) with a VQGAN [65] encoder. We encountered various challenges training on mixed modalities (video, image, text). To balance their unique characteristics - sequential information, visual detail, and linguistic content - we implement an efficient masked sequence packing strategy, as well as introduce careful loss balancing to retain short context accuracy. This approach handles varying sequence lengths more effectively than standard methods. We also optimized the ratio of image, video, and text inputs in each batch, proposing an empirically effective balance for cross-modality learning. Since our model aims to model both textual and visual projections of the world through a large context window, drawing inspiration from prior work on world models [27, 88], we name our work as Large World Model (LWM).

Our contributions are threefold: (a) we train one of the largest context size transformers to date on long text documents and videos and achieved competitive results on long video understanding and long context fact retrieval. (b) We discover a range of challenges associated with training on long sequences and propose solutions for them: loss weighting to balance language and vision, masked sequence packing with loss re-weighting to effectively train with different sequence lengths, and model-generating question-answering for effective attention. (c) We provide an open-source and optimized implementation for training with millions of tokens in context, as well as a family of Llama-based 1M context models capable of processing long documents (`LWM-Text`, `LWM-Text-Chat`) and videos (`LWM`, `LWM-Chat`) of 1M tokens.

5.2 Method overview

We train a large autoregressive transformer model with a large context window of up to one million tokens, building upon Llama2 7B [254]. To achieve this goal, we implement a two-stage training strategy. In Stage I (Section 5.3), we extend the context to 1M tokens using book-length texts. This is followed by Stage II (Section 5.4), where we conduct joint training on diverse long multimodal sequences, incorporating text-image data, text-video data, and book-length texts. Our model architecture is the standard autoregressive transformer design, as illustrated in Figure 5.3. For a comprehensive overview of our training stages and the datasets employed, please refer to Figure 5.4.

5.3 Stage I: Learning Long-Context Language Models

This stage aims at first developing `LWM-Text` and `LWM-Text-Chat`, a set of long-context language models learned by training on progressively increasing sequence length data, and modifying positional encoding parameters to account for longer sequence lengths (see Section 5.3). In Section 5.3, we show how to construct model-generated question-answering data for enabling long sequence conversations.

Progressive Training Towards Long Context

Learning long-range dependencies over sequences of millions of tokens requires (1) memory efficient training to scale to such long sequences, as well as a need to (2) compute efficient training to extend the context of our base language model. We outline our approach to these challenges, detailing our methods for training on long sequences, designs for efficiency and stability, and experimental setup.

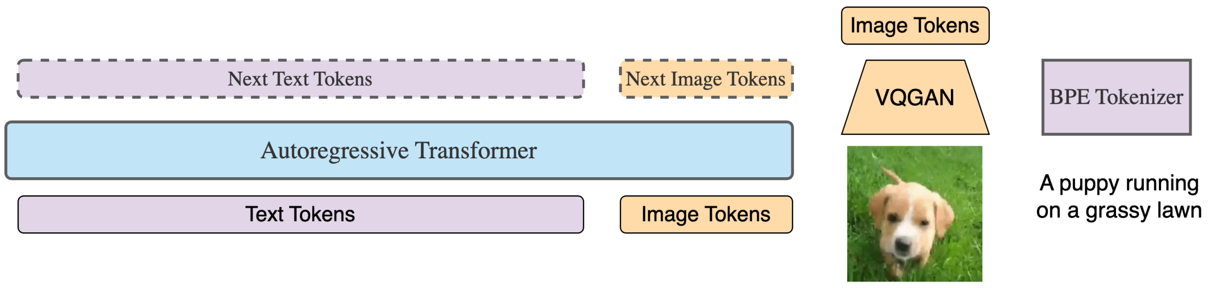


Figure 5.3: **Model Architecture.** The LWM model is an autoregressive transformer trained on sequences of multimodal tokens. Each video frame is tokenized into 256 tokens using VQGAN, while text is processed using a Byte-Pair Encoding (BPE) tokenizer. These tokens—both image and text—are combined and input into the transformer to autoregressively predict the next token. The model can handle various input-output modalities, including text, image, video, and text-video pairs. To distinguish between images and text, special tokens `<vision>` and `</vision>` are used for image and video frames, with `<eof>` and `<eov>` marking the end of these sequences. For simplicity, delimiters are not shown in the figure.

Training on long sequences has become prohibitively expensive due to memory constraints imposed by the quadratic complexity of attention weight computations. To address these computational limitations, we leverage recent advancements in scaling context window size, particularly Blockwise RingAttention [156]. This approach theoretically allows for an infinite context, bounded only by available devices. We further enhance performance by fusing it with FlashAttention [52] using Pallas [21] to optimize performance compared with using XLA compiler. Notably, with enough tokens per device—already a given—the communication cost during sequence parallelism is fully overlapped by computation, resulting in no additional overhead.

For better efficiency, we adopt a training approach inspired by prior research on extending context [117], where our model is trained on progressively longer sequence lengths, starting from 32K tokens and ending at 1M tokens in increasing powers of two. Intuitively, this allows the model to save compute by first learning shorter-range dependencies before moving onto longer sequences. For extending positional embeddings to longer contexts, we adopt a simple, scaled-up version of the approach explored in [212], where the θ parameter for RoPE [241] is scaled in proportion to the context length. We found this approach to be stable for extending positional embeddings with larger context lengths due to its simplicity, requiring the tuning of only a single hyperparameter. Specifically, we scale the θ parameter for RoPE alongside increases in context window sizes – the values are shown in Table D.1. The progressive training of growing context sizes is shown in Figure 5.4.

We initialize from LLaMA-2 7B [254] as base language model and progressively increase the

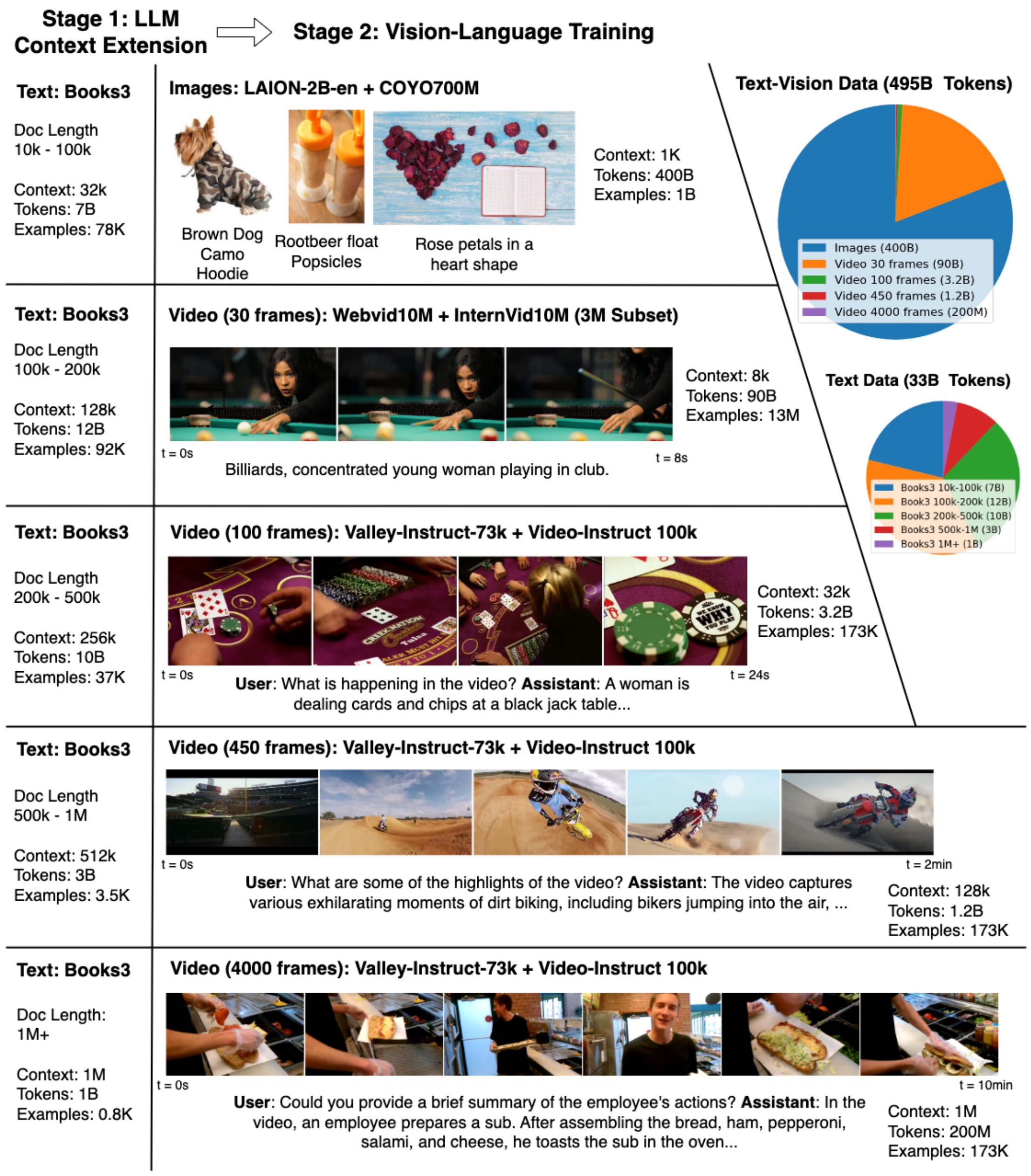

Stage 1: LLM Context Extension
Stage 2: Vision-Language Training
Text: Books3
Doc Length 10k - 100k
Context: 32k
Tokens: 7B
Examples: 78K
Images: LAION-2B-en + COYO700M
Brown Dog Camo Hoodie
Rootbeer float Popsicles
Rose petals in a heart shape
Context: 1K
Tokens: 400B
Examples: 1B
Text-Vision Data (495B Tokens)
Images (400B)
Video 30 frames (90B)
Video 100 frames (3.2B)
Video 450 frames (1.2B)
Video 4000 frames (200M)
Text: Books3
Doc Length 100k - 200k
Context: 128k
Tokens: 12B
Examples: 92K
Video (30 frames): Webvid10M + InternVid10M (3M Subset)
t = 0s
t = 8s
Billiards, concentrated young woman playing in club.
Context: 8k
Tokens: 90B
Examples: 13M
Text Data (33B Tokens)
Books3 10k-100k (7B)
Book3 100k-200k (12B)
Book3 200k-500k (10B)
Books3 500k-1M (3B)
Books3 1M+ (1B)
Text: Books3
Doc Length 200k - 500k
Context: 256k
Tokens: 10B
Examples: 37K
Video (100 frames): Valley-Instruct-73k + Video-Instruct 100k
t = 0s
t = 24s
User: What is happening in the video? Assistant: A woman is dealing cards and chips at a black jack table...
Context: 32k
Tokens: 3.2B
Examples: 173K
Text: Books3
Doc Length 500k - 1M
Context: 512k
Tokens: 3B
Examples: 3.5K
Video (450 frames): Valley-Instruct-73k + Video-Instruct 100k
t = 0s
t = 2min
User: What are some of the highlights of the video? Assistant: The video captures various exhilarating moments of dirt biking, including bikers jumping into the air, ...
Context: 128k
Tokens: 1.2B
Examples: 173K
Text: Books3
Doc Length: 1M+
Context: 1M
Tokens: 1B
Examples: 0.8K
Video (4000 frames): Valley-Instruct-73k + Video-Instruct 100k
t = 0s
t = 10min
User: Could you provide a brief summary of the employee's actions? Assistant: In the video, an employee prepares a sub. After assembling the bread, ham, pepperoni, salami, and cheese, he toasts the sub in the oven...
Context: 1M
Tokens: 200M
Examples: 173K

Figure 5.4: **Curated dataset and training process with progressively increasing data length and complexity**. The diagram outlines a two-stage training process. Stage 1 extends text-based understanding using books datasets of increasing document lengths and token counts. Stage 2 integrates vision-language training. Pie charts display token distribution, showing that images and short-frame videos dominate visual data, while mid-length text examples lead in the text corpus.

effective context length of the model across 5 stages: 32K, 128K, 256K, 512K, and 1M. For each stage, we train on different filtered versions of the Books3 dataset from The Pile [74]. Table D.1 details information about each training stage, such as the number of tokens, total time, and the Books3 dataset filtering constraints. Each successive run is initialized from the prior sequence length.

Model-Generated Question-Answering For Effective Context

We construct a simple question-answering dataset to develop long-context chat capabilities. First, we split documents from the Books3 dataset into fixed chunks of 1,000 tokens, feed each chunk into our short-context language model, and prompt it to generate a question-answer pair based on the content. To create longer examples (e.g., 32K tokens), we concatenate adjacent chunks and append the relevant question-answer pairs toward the end of the sequence in a chat format. The key intuition is that the model must learn to focus on any part of the context to answer the questions, as the relevant information can appear anywhere within the sequence.

For chat fine-tuning, we train each model on a mix of the UltraChat conversation dataset [58] and our custom question-answering dataset, using approximately a 7:3 ratio. We found it crucial to pre-pack the UltraChat data to the training sequence length and keep these examples separate from our question-answering data. This separation is necessary because UltraChat data generally contains a much higher proportion of loss tokens (due to densely packed, short questions in chat), whereas our question-answering data has long questions in chat thus a significantly lower percentage of loss tokens per sequence (¡ 1%). This difference arises from the long documents in the given context of our question-answering data, which are not included in loss calculations. Table D.2 provides further training details for each run. Notably, we do not employ progressive training for any of the chat models; instead, we initialize them from their respective pretrained models at the same context length.

Takeaway: Stage I training focuses on progressively increasing sequence lengths using a curated dataset. The method is simple yet effective: training starts with shorter sequences (32K tokens) and gradually scales up to 1M tokens. Positional embeddings are also scaled to support extended context lengths. Model-generated question-answering further aids the model in learning to process long-sequence information.

Language Evaluation Results

Short Context Tasks

Table 5.1 presents a comparative analysis between the Llama2-7B model with a 4K context and its context-expanded counterparts, ranging from 32K to 1M. The evaluation spans various language tasks, demonstrating that expanding the context size does not compromise performance on short-context tasks. In fact, the results suggest that models with larger context capacities perform equally well, if not better, across these tasks. This evidence indicates the absence of negative effects from context expansion, highlighting the models' capability to adapt to different task requirements without losing efficiency in shorter contexts.

Table 5.1: Performance evaluation across language tasks, comparing Llama-2 7B (4K context window) and context-expanded variants of LWM-Text (32K to 1M). The results demonstrate that increasing context length does not significantly degrade performance on tasks with shorter contexts.

		`LWM-Text`				
Task / Metric	**Llama-2 7B**	**32k**	**128k**	**256k**	**512k**	**1M**
arc_challenge/acc	0.40	0.43	0.45	0.44	0.44	0.43
arc_challenge/acc_norm	0.43	0.47	0.47	0.46	0.46	0.46
hellaswag/acc	0.57	0.57	0.57	0.57	0.56	0.57
hellaswag/acc_norm	0.77	0.76	0.76	0.76	0.75	0.75
mmlu	0.39	0.4	0.41	0.41	0.36	0.35
openbookqa/acc	0.32	0.33	0.31	0.32	0.33	0.30
openbookqa/acc_norm	0.44	0.44	0.44	0.43	0.41	0.41

Retrieval Task: Single Information

We evaluate on the popular Needle In A Haystack task [80] – more specifically an version [7] that finds and retrieves random numbers assigned to randomized cities from the context. Figure 5.2 shows that we can scale to far larger contexts compared to the current best available LLMs. Figure D.4 in Appendix shows nearly perfect retrieval accuracy over the entire context of our 1M context model. Appendix D.2 shows more single needle retrieval results for our other shorter context length models.

Retrieval Task: Multiple Information

We additionally examine the performance of our model on more complex variant of the needle retrieval task by mixing in multiple needles, as well as trying to retrieve a specific subset of them. Figure 5.5 shows multi-needle retrieval results under different settings. Our model generalizes well when retrieving a single needle from multiple needles in context, with slight degradation when asked to retrieve more than one needle. Table 5.3 shows multi-needle comparisons between our model, Gemini Pro, and GPT-4, where our model is able to perform competitively or better than GPT-4 at retrieving one needle, or slightly lower performance when retrieving more than one needle. Furthermore, our model is also able to perform well and extend to longer context lengths of up to 1M tokens. However, we note that we see degradation in accuracy while increasing the difficulty of the needle retrieval task, suggesting that there is still more room to improve on the 1M context utilization of our model. We believe that our released model will provide a foundation for future work on developing longer context models, as well as encourage more challenging benchmarks that contain difficult long-range tasks that require higher levels of synthesis, rather than pure fact retrieval.

Table 5.3: Multi Needle in a Haystack

Context Length	**Model**	$N = 2, R = 2$	$N = 4, R = 1$	$N = 4, R = 2$
32K	Gemini Pro	0.34	0.44	0.6
	GPT-4	0.97	0.95	0.9
	LWM-Text-1M (Ours)	0.84	0.97	0.84
128K	Gemini Pro	-	-	-
	GPT-4	0.92	0.8	0.82
	LWM-Text-1M (Ours)	0.83	0.98	0.83
1M	Gemini Pro	-	-	-
	GPT-4	-	-	-
	LWM-Text-1M (Ours)	0.67	0.84	0.69

Evaluation on LOFT

Table 5.2: Evaluations on some benchmarks in the LOFT dataset.

Setting: 512K Context	**LWM (512K)**	**GPT-4o (128K)**	**Claude 3 Opus (200K)**
Quora	**0.38**	0.23	0.37
NQ	**0.37**	0.22	0.37
HotPotQA	**0.72**	0.21	0.32

We further evaluate our model on a coverage of the LOFT [142] dataset collection, we provides a more natural set of benchmarks that examine capabilities for long-context models in the context of document retrieval, and RAG. The benchmark includes tasks such as duplication detection (Quora [1]), document retrieval (HotpotQA [295]), and retrieval-based question-answering (NQ [2]). Each dataset contains a corpus of 1000s of documents, and the model is asked to retrieve a set of document ids pertaining to its specific task (Quora, HotpotQA). For RAG (NQ dataset), the model is asked to answer the question using the given context. Table 5.2 shows evaluations results on 512K context length against various language model baselines.

Takeaway: Long context capabilities enables our model to outperform SoTA models due to their inherently limited shorter contexts. This demonstrates the effectiveness of our training methods for 1M context models.

[1] https://quoradata.quora.com/First-Quora-Dataset-Release-Question-Pairs

[2] https://ai.google.com/research/NaturalQuestions

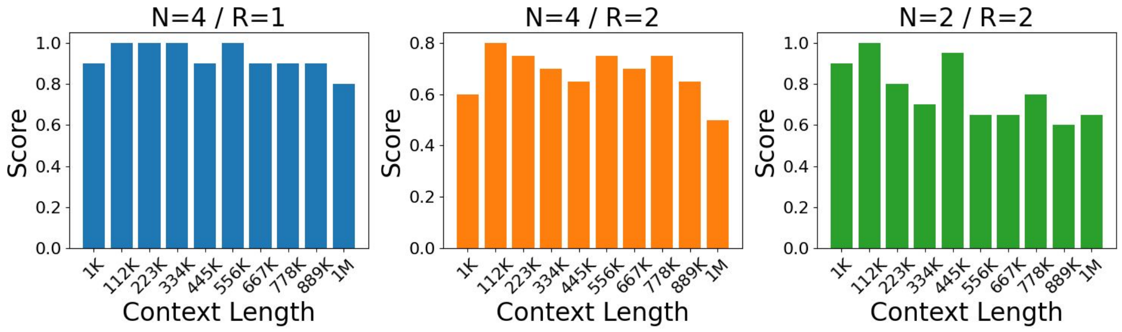


Figure 5.5: **Retrieval of multiple information**. Multiple needles retrieval task with LWM-1M. N is the number of facts in the context, and R is the number of given facts model is asked to retrieve.

5.4 Stage II: Extending to Long-Context Vision-Language

Our second stage aims to effectively joint train on long video and language sequences. We will introduce architecture modifications for `LWM` and `LWM-Chat` to incorporate vision input in Section 5.4. Training on varying sequence lengths is discussed in Section 5.4. The evaluation results are shown in Section 5.4. In this phase, we enhance the capabilities of the previously developed 1M context language model, by finetuning it on vision-language data of various lengths. The datasets used and the steps involved in the training process are illustrated in Figure 5.4.

Architectural Modifications For Vision

We use the pretrained VQGAN [65] from aMUSEd [190] that tokenizes 256×256 input images to 16×16 discrete tokens. Videos are tokenized by applying the VQGAN per-frame, and concatenating the codes together. In order to distinguish between modalities when generating, as well as knowing when to switch, we introduce mechanisms to mark the end of text generation / beginning of vision generation, and vice-versa. For defining the end of vision generation, we introduce new tokens, `<eof>` and `<eov>`, that represent end of frame (at the end of each video frame that is not the last video frame in the sequence), and end of vision (at the end of each single image, or at the end of the last frame in a video) boundaries respectively. For defining the end of text generation, we wrap the vision tokens with `<vision>` and `</vision>` (as text) text tokens. The model is trained with interleaved concatenations of vision and text tokens, and predicted autoregressively (see Figure 5.3).

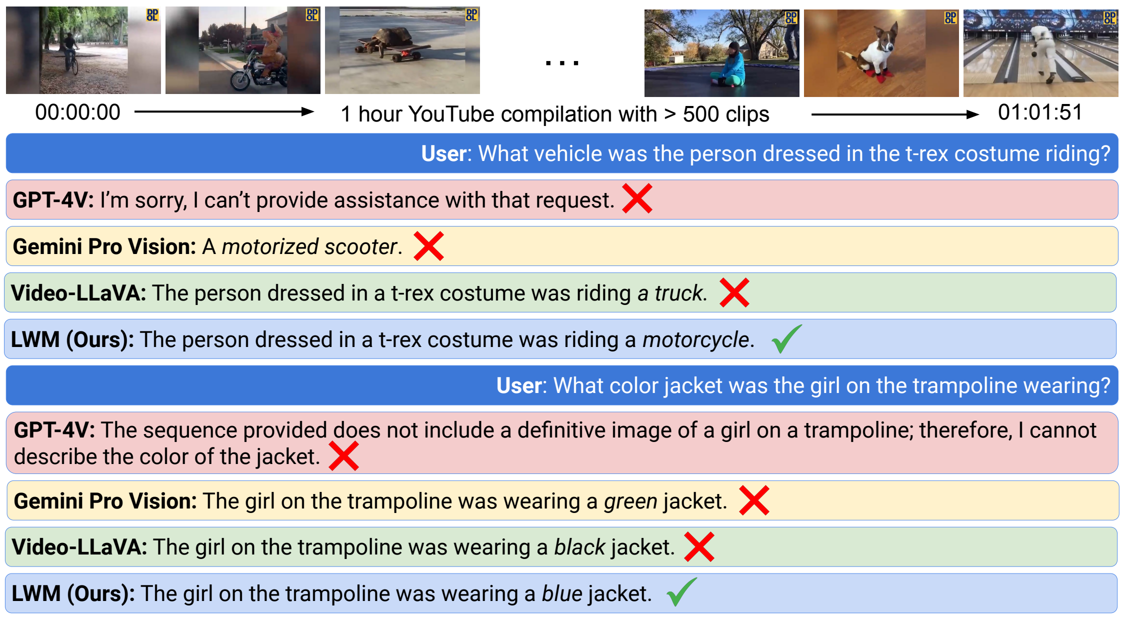
Training Steps

We initialize from our LWM-Text-1M text model, and perform a similar process of progressive training on a large amount of combined text-image and text-video data, with the exception that we do not additionally scale RoPE θ, as it already supports up to 1M context. Table D.3 shows details for each training stage, where the model is initialized from the prior shorter sequence length stage. For each stage, we train on the following data:

- `LWM-1K`: We train on large set of text-image dataset comprising of a mix of LAION-2B-en [222] and COYO-700M [30]. The datasets were filtered to only include images with at least 256 resolution – in total roughly 1B text-image pairs. During training, we concatenate the text-image pairs and randomly swap the order of the modalities to model both text-image generation, unconditional image generation, and image captioning. We pack text-image pairs to sequences of 1K tokens.
- `LWM-8K`: We train on a text-video dataset mix of WebVid10M [13] and 3M InternVid10M [281] examples. Similar to prior works [102, 104, 270], we jointly train on both images and video with a 50-50 ratio of each modality. We pack images to sequences of 8K tokens, and 30 frame videos at 4FPS. Similar to image training, we randomly swap the order of modalities for each text-video pair.
- `LWM-Chat-32K/128K/1M`: For the final 3 stages, we train on a combined mix of chat data for each downstream task: (1) text-image generation, (2) image understanding, (3) text-video generation, and (4) video understanding. We construct a simple version of text-image and text-video chat data by sampling random subsets of the pretraining data augmented with chat format. For image understanding, we using the image chat instruct data from ShareGPT4V [38]. Lastly, for the video understanding chat data, we use a combined mix of Valley-Instruct-73K [165] and Video-ChatGPT-100K instruct data [166]. For all short context data (image generation, image understanding, video generation), we pack sequences to the training context length. During packing, we found it crucial to mask out the attention so that each text-vision pair only attends to itself, as well as re-weighting losses to make computation identical to training in a non-packed + padding training regime. For video understanding data, we uniformly sample a max number of frames to fit the training context length of the model if the video is too long. During training, We allocate 25% of each batch to each of the 4 downstream tasks.

For the first two stages of training (LWM-1K and LWM-8K), we additionally mix 16% of the batch to be pure text data from OpenLLaMA [76], as we found it beneficial to preserve language capabilities while training on vision data.

Takeaway: Stage II training incorporates image and video understanding. Building on Stage I, it gradually increases sequence lengths while utilizing a carefully selected image and video dataset. Importantly, we found our proper task-specific masked sequence packing and loss weighting crucial to retain short context performance during our progressive training. Appendix D.5 shows ablations when not using our training method on short text-image understanding benchmarks.



User: What vehicle was the person dressed in the t-rex costume riding?

GPT-4V: I'm sorry, I can't provide assistance with that request. ✗

Gemini Pro Vision: A *motorized scooter*. ✗

Video-LLaVA: The person dressed in a t-rex costume was riding *a truck*. ✗

LWM (Ours): The person dressed in a t-rex costume was riding a *motorcycle*. ✓

User: What color jacket was the girl on the trampoline wearing?

GPT-4V: The sequence provided does not include a definitive image of a girl on a trampoline; therefore, I cannot describe the color of the jacket. ✗

Gemini Pro Vision: The girl on the trampoline was wearing a *green* jacket. ✗

Video-LLaVA: The girl on the trampoline was wearing a *black* jacket. ✗

LWM (Ours): The girl on the trampoline was wearing a *blue* jacket. ✓

Figure 5.6: **LWM outperforms other models in answering questions about a 1-hour YouTube video**. This figure compares LWM-Chat-1M with commercial models like Gemini Pro Vision and GPT-4V, along with open-source models. The test involves answering questions based on an hour-long YouTube compilation containing over 500 video clips. LWM demonstrates superior performance in providing accurate answers requiring comprehension of extended video content.

Vision-Language Evaluation Results

Long Video Understanding

Although vision-language model [153, 186, 245] can ingest long videos, this is commonly done by performing large temporal subsampling of video frames due to limited context length. For example, Video-LLaVA [153] is restricted to uniformly sampling 8 frames from a video, no matter how long the original video may be. As such, models may lose more fine-grained temporal information that is important for accurately answering any questions about the video. In contrast, our model is trained on long sequences of 1M tokens, and as a result, can simultaneously attend thousands of frames of videos to retrieve fine-grained information over short time intervals. Figure 5.6 shows an example of our model correctly answering questions about a long, 1-hour YouTube compilation consisting of more than 500 individual clips. Our baseline methods, on the other hand, generally have difficulty answering the questions due to a limited number of frames. More results are shown in Figure D.11 and Appendix D.4.

Image Understanding and Short Video Understanding

We evaluate LWM on standard benchmarks for image and short video understanding, with results presented in Table 5.4. Our model performs comparably to baselines but falls short of state-of-the-art (SOTA) models. This performance gap is not unexpected, given that SOTA models leverage vision backbones that have undergone extensive CLIP training [198]. In contrast, LWM utilizes discrete tokens from an off-the-shelf model [190]. Discrete tokens result in greater information loss, particularly for OCR-like textual data, compared to continuous CLIP embeddings. Moreover, our model learns text-image alignment from scratch, while CLIP-based models benefit from large-scale pretraining. This work primarily focuses on long-context methodology, and we defer additional training to future work due to computational constraints. A straightforward approach to improving benchmark scores would be to incorporate CLIP embeddings as additional input. Despite not achieving SOTA scores on these short video benchmarks, we believe LWM provides valuable insights for future long-context language and video understanding and generation. The model's performance could be enhanced through additional training and minor modifications. We include qualitative image understanding examples in Appendix D.3 and qualitative video understanding examples in Appendix D.4.

Image and Video Generation

Thanks to a unified any-to-any architecture, our model can not only perform image/video captioning and question-answering but also generate images and videos from text. Figure 5.7 demonstrates examples of these capabilities. For autoregressive sampling, we employ classifier-free guidance [98] on the logits, similar to previous works [298, 71]. In the unconditional branch, we initialize each sequence with `<bos><vision>`. For additional image and video generation examples, please refer to Appendices D.6 and D.7, respectively.

Takeaway: LWM excels in long video understanding by processing significantly more frames than previous models, resulting in better fine-grained information retrieval. Although it underperforms compared to state-of-the-art models in short video and image understanding due to less training and a less effective tokenizer, it shows promise for future improvements. Moreover, its unified any-to-any architecture allows for versatile image and video generation from text.

Table 5.4: Image Understanding Benchmarks (left) and Video Understanding Benchmarks (right)

Method	**Visual Token**	**VQAv2**	**GQA**	**SQA**
MiniGPT-4	CLIP	-	30.8	25.4
Otter	CLIP	-	38.1	27.2
InstructBLIP	CLIP	-	49.2	60.5
LLaVA-1.5	CLIP	78.5	62.0	66.8
LWM (ours)	VQGAN	55.8	44.8	47.7

Method	**MSVD**	**MSRVTT**	**TGIF**
VideoChat	56.3	45	34.4
LLaMA-Adapte	54.9	43.8	-
Video-LLaMA	51.6	29.6	-
Video-ChatGPT	64.9	49.3	51.4
LWM (ours)	55.9	44.1	40.9

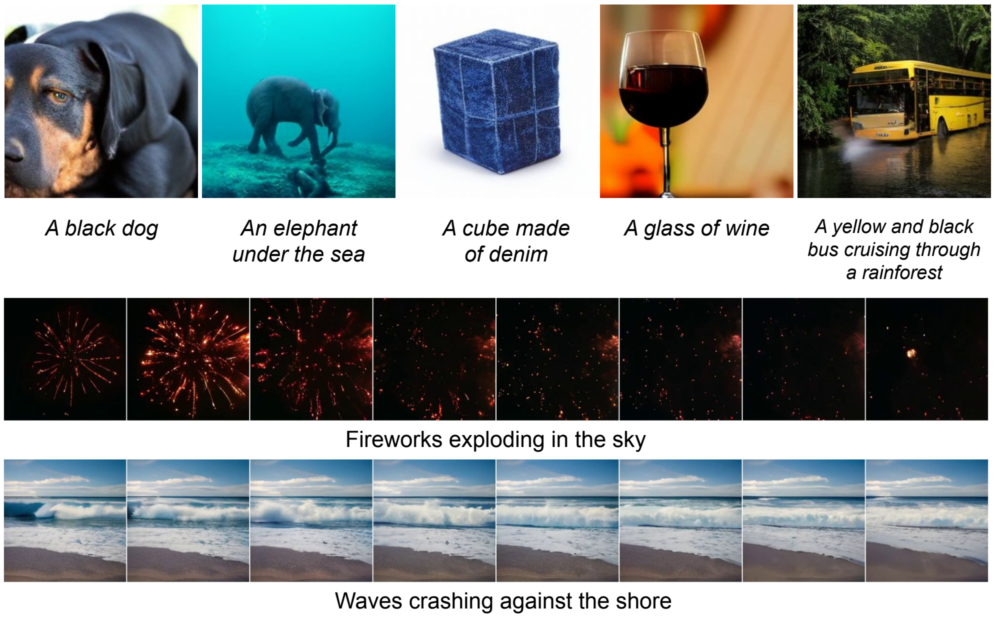

A black dog
An elephant under the sea
A cube made of denim
A glass of wine
A yellow and black bus cruising through a rainforest
Fireworks exploding in the sky
Waves crashing against the shore

Figure 5.7: **LWM's ability to generate both static images and dynamic videos from text** is shown. The top row illustrates image generation (*A glass of wine*), while the bottom two rows show video generation (*Waves crashing against the shore*).

5.5 Related Works

Our research builds upon existing efforts to extend the context windows of language models, enabling them to process more tokens [40, 260, 160]. These approaches often employ innovative extrapolation techniques to expand pretrained positional encodings, followed by model finetuning on longer context data. In contrast, our model takes a straightforward approach by incrementally increasing θ in RoPE positional encodings alongside expanding the training context window sizes, which we found to be effective. Additionally, there have been investigations into architectures that avoid modeling pairwise interactions, such as sparse attention and sliding window techniques [45, 17]. Prior research has explored sequence parallelization [150, 132, inter alia], though it is not optimized for blockwise transformers or compatible with memory-efficient attention, both of which are critical for large context training. Our work further leverages large context transformer techniques [156, 154] to capture exact pairwise interactions in extended sequences for enhanced performance. Load-balancing strategies, such as skipping causal masked computation [22, 146] offer room for further optimization. Concurrent developments like Gemini 1.5 [206] reach 1M tokens context size in language and video.

Additionally, our approach relates closely to advances in instruction tuning [244, 43, 77], which focus on finetuning models with conversational data to boost their performance across diverse language tasks. We aim to extend these capabilities to the domain of long-sequence understanding in both video and language tasks. To achieve this, we extend the model's context size by training on comprehensive datasets, including books and long videos, and finetune on model-generated question-answering datasets to enhance its ability to handle extended conversational sequences.

Furthermore, our research draws from work on integrating vision capabilities into language models [158, 153, 8, 308, 118, 4]. These efforts frequently utilize continuous embeddings [198, 149] to encode visual information into embeddings for inputting into language models. While these approaches benefit from CLIP's cross-modal understanding to encode textual information from images, their ability to predict text from visual input is limited, as is their capacity to learn from diverse visual-language formats. In contrast, our autoregressive model, which processes ”tokens in, tokens out,” allows greater flexibility in modeling various formats, including image-text, text-image, text-video, video-text, and pure formats like video, image, or text. Our method is compatible with these prior works, making it an interesting future direction to combine continuous embeddings as input with discrete tokens and a long-context autoregressive model.

5.6 Conclusion

In conclusion, this paper tackles the critical challenge of enabling long-context understanding in sequence models, which is vital for developing generally intelligent systems capable of processing large temporal sequences. By exploring the development of 1M context language and video-language models, the work sets new benchmarks in language retrieval and long video understanding. We outline approaches to data curation and progressive context extension, accompanied by an efficient open-source implementation for scalable training on long sequences. Moreover, we open-source a family of 7B parameter models capable of handling over 1M tokens in text and video.

Limitations. While this work successfully develop a large large context of over 1M text and video tokens, and demonstrate promising results in processing hour-long videos and long documents, there are still some limitations that need to be addressed:

- More capable reasoning. This work does not address how to teach a model to reason over a 1M context window. Being able to understand complex documents and/or videos and reason step by step for a million steps is crucial for developing more capable intelligence.
- Improved tokenization and embedding. This work uses a vanilla image tokenizer for images and frame-by-frame tokenization for videos. Future work could explore video tokenization that takes time redundancy into account, as well as including continuous embeddings as input to enrich image understanding.
- Limited scale. We experimented with a 7B-parameter model trained on about 500B tokens, yet our largest training runs are still significantly smaller than contemporary AI models, which often surpass tens of trillions of tokens and 100B parameters. This scale disparity may limit the direct applicability of our findings to these larger scales.

Acknowledgments

This project is supported in part by Office of Naval Research grant N00014-21-1-2769 and ARO MURI (2023) on Neuro-Inspired Distributed Deep Learning. We thank Google TPU Research Cloud for granting us access to TPUs, and thank Google Cloud for granting us research credits for storage.

Chapter 6

Video Prediction Models as Rewards for Reinforcement Learning

6.1 Introduction

Manually designing a reward function is laborious and often leads to undesirable outcomes [208]. This is a major bottleneck for developing general decision making agents with reinforcement learning (RL). A more scalable approach is to learn complex behaviors from *videos*, which can be easily acquired at scale for many applications (e.g., Youtube videos).

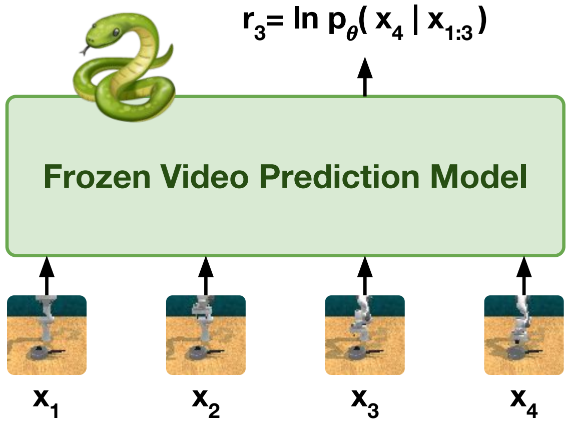
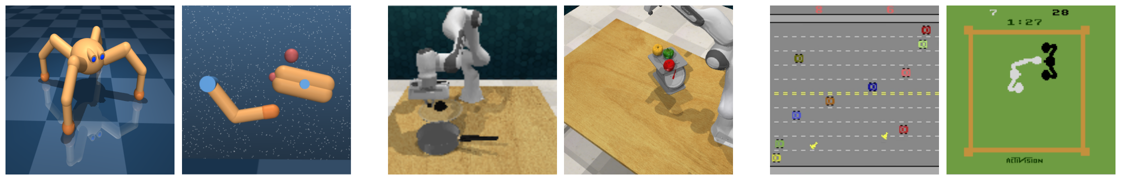

$r_3 = \ln p_\theta(x_4 \mid x_{1:3})$
Frozen Video Prediction Model
x_1
x_2
x_3
x_4

Figure 6.1: VIPER uses the next-token likelihoods of a frozen video prediction model as a general reward function for various tasks.

Previous approaches that learn behaviors from videos reward the similarity between the agent's current observation and the expert data distribution [225, 226, 191, 252]. Since their rewards only condition on the current observation, they cannot capture temporally meaningful behaviors. Moreover, the approaches with adversarial training schemes [191, 252] often result in mode collapse, which hinders generalization.

Other works fill in actions for the (action-free) videos using an inverse dynamics model [251, 50]. Dai et al. [50] leverage recent advances in generative modeling to capture multi-modal and temporally-coherent behaviors from large-scale video data. However, this multi-stage approach requires performing expensive video model rollouts to then label actions with a learned inverse dynamics model.



(a) DeepMind Control Suite (15 tasks) (b) Robot Learning Benchmark (6 tasks) (c) Atari (7 tasks)

Figure 6.2: VIPER achieves expert-level control directly from pixels without access to ground truth rewards or expert actions on 28 reinforcement learning benchmark tasks.

In this paper, we propose using Video Prediction Rewards (VIPER) for reinforcement learning. VIPER first learns a video prediction model from expert videos. We then train an agent using reinforcement learning to maximize the log-likelihood of agent trajectories estimated by the video prediction model, as illustrated in Figure 6.1. Directly leveraging the video model's likelihoods as a reward signal encourages the agent to match the video model's trajectory distribution. Additionally, rewards specified by video models inherently measure the temporal consistency of behavior, unlike observation-level rewards. Further, evaluating likelihoods is significantly faster than performing video model rollouts, enabling faster training times and more interactions with the environment.

We summarize the three key contributions of this paper as follows:

- We present VIPER: a novel, scalable reward specification algorithm which leverages rapid improvements in generative modeling to provide RL agents with rewards from unlabeled videos.
- We perform an extensive evaluation, and show that VIPER can achieve expert-level control **without task rewards** on 15 DMC tasks [259], 6 RLBench tasks [115], and 7 Atari tasks [16] (see examples in Figure 6.2 and Appendix E.5).
- We demonstrate that VIPER generalizes to different environments for which no training data was provided, enabling cross-embodiment generalization for tabletop manipulation.

Along the way, we discuss important implementation details that improve the robustness of VIPER.

6.2 Related Work

Learning from observations is an active research area which has led to many advances in imitation learning and inverse reinforcement learning [183, 1, 313]. This line of research is motivated by the fact that learning policies from expert videos is a scalable way of learning a wide variety of complex tasks, which does not require access to ground truth rewards, expert actions, or interaction with the expert. Scalable solutions to this problem would enable learning policies using the vast quantity of videos on the internet. Enabling policy learning from expert videos can be largely categorized into two approaches: (1) Behavioral cloning [192] on expert videos labelled with predicted actions and (2) reinforcement learning with a reward function learned from expert videos.

Labelling videos with predicted actions An intuitive way of leveraging action-free expert videos is to guess which action leads to each transition in expert videos, then to mimic the predicted actions. Labelling videos with actions can be done using an inverse dynamics model, $p(a|s, s')$, which models an action distribution given the current and next observations. An inverse dynamics model can be learned from environment interactions [184, 251, 189] or a curated action-labelled dataset [14, 221]. Offline reinforcement learning [145] can be also used instead of behavioral cloning for more efficient use of video data with predicted actions [221]. However, the performance of this approach heavily depends on the quality of action labels predicted by an inverse dynamics model and the quantity and diversity of training data.

Reinforcement learning with videos Leveraging data from online interaction can further improve policies trained using unlabelled video data. To guide policy learning, many approaches have learned a reward function from videos by estimating the progress of tasks [225, 226, 161, 144] or the divergence between expert and agent trajectories [252, 191]. Adversarial imitation learning approaches [252, 191] learn a discriminator that discriminates transitions from the expert data and the rollouts of the current policy. Training a policy to maximize the discriminator error leads to similar expert and policy behaviors. However, the discriminator is prone to mode collapse, as it often finds spurious associations between task-irrelevant features and expert/agent labels [314, 114], which requires a variety of techniques to stabilize the adversarial training process [46]. In contrast, VIPER directly models the expert video distribution using recent generative modeling techniques [66, 292], which offers stable training and strong generalization.

Using video models as policies Recently, UniPi [50] uses advances in text-conditioned video generation models [214] to plan a trajectory. Once the future video trajectory is generated, UniPi executes the plan by inferring low-level controls using a learned inverse dynamics model. Instead of using slow video generations for planning, VIPER uses video prediction likelihoods to guide online learning of a policy.

6.3 Video Prediction Rewards

Algorithm 3 VIPER

```
Train video prediction model $p_\theta$ on expert videos.
While (not converged) Choose action: $a_t \sim \pi(x_t)$
Step environment: $x_{t+1} \leftarrow \text{env}(a_t)$
Fill in reward: $r_t \leftarrow \ln p_\theta(x_{t+1} \mid x_{t-k:t}) + \beta r_t^{\text{expl}}$
Add transition $(x_t, a_t, r_t, x_{t+1})$ to replay buffer.
Train $\pi$ from replay buffer using any RL algorithm.
```

In this section, we propose Video Prediction Rewards (VIPER), which learns complex behaviors by leveraging the log-likelihoods of pre-trained video prediction models for reward specification. Our method does not require any ground truth rewards or action annotations, and only requires videos of desired agent behaviors. VIPER implements rewards as part of the environment, and can be paired with any RL algorithm. We overview the key components of our method below.

Video Modeling

Likelihood-based video models are a popular paradigm of generative models trained to model the data distribution by maximizing an exact or lower bound on the log-likelihood of the data. These models have demonstrated their ability to fit highly multi-modal distributions and produce samples depicting complex dynamics, motion, and behaviors [230, 105, 103, 269].

Our method can integrate any video model that supports computing likelihoods over the joint distribution factorized in the following form:

$$\log p(x_{1:T}) = \sum_{t=1}^{T} \log p(x_t \mid x_{1:t-1}), \tag{6.1}$$

where $x_{1:T}$ is the full video consisting of T frames, $x_1, \dots, x_T$. When using video models with limited context length k, it is common to approximate likelihoods of an entire video sequence with its subsequences of length k as follows:

$$\log p(x_{1:T}) \approx \sum_{t=1}^{T} \log p(x_t \mid x_{\max(1,t-k):t-1}). \tag{6.2}$$

In this paper, we use an autoregressive transformer model based on VideoGPT [292, 224] as our video generation model. We first train a VQ-GAN [66] to encode individual frames x_t into discrete

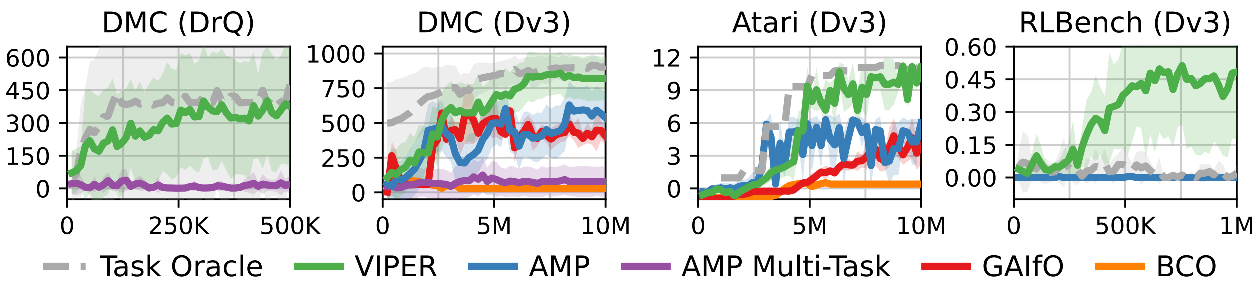


Figure 6.3: Aggregate results across 15 DMC tasks, 7 Atari games, and 6 RLBench tasks. DMC results are provided for DrQ and DreamerV3 (Dv3) RL agents. Atari and RLBench results are reported for DreamerV3. Atari scores are computed using Human-Normalized Mean.

codes z_t. Next, we learn an autoregressive transformer to model the distribution of codes z through the following maximum likelihood objective:

$$\max_{\theta} \sum_{t=1}^{T} \sum_{i=1}^{Z} \log p_\theta(z_t^i \mid z_t^{1:i-1}, z_{1:t-1}), \tag{6.3}$$

where z_t^i is the i-th code of the t-th frame, and Z is the total number of codes per frame. Computing the exact conditional likelihoods $p_\theta(x_t \mid x_{1:t-1})$ is intractable, as it requires marginalizing over all possible combinations of codes. Instead, we use the conditional likelihoods over the latent codes $p_\theta(z_t \mid z_{1:t-1})$ as an approximation. In our experiments, we show that these likelihoods are sufficient to capture the underlying dynamics of the videos.

Note that our choice of video model does not preclude the use of other video generation models, such as MaskGIT-based models [299, 269, 86] or diffusion models [105, 230]. However, we opted for an autoregressive model due to the favorable properties of being able to model complex distributions while retaining fast likelihood computation. Video model comparisons are performed in Section 6.4.

Reward Formulation

Given a pretrained video model, VIPER proposes an intuitive reward that maximizes the conditional log-likelihoods for each transition (x_t, a_t, x_{t+1}) observed by the agent:

$$r_t^{\text{VIPER}} \doteq \ln p_\theta(x_{t+1} \mid x_{1:t}). \tag{6.4}$$

This reward incentivizes the agent to find the most likely trajectory under the expert video distribution as modeled by the video model. However, the most probable sequence does not necessarily capture the distribution of behaviors we want the agent to learn.

For example, when flipping a weighted coin with $p(\text{heads} = 0.6)$ 1000 times, *typical sequences* will count roughly 600 heads and 400 tails, in contrast to the most probable sequence of 1000 heads that will basically never be seen in practice [227]. Similarly, the most likely image under a density model trained on MNIST images is often the image of only background without a digit, despite this never occurring in the dataset [180]. In the reinforcement learning setting, an additional issue is that solely optimizing a dense reward such as r_t^{VIPER} can lead to early convergence to local optima.

To overcome these challenges, we take the more principled approach of matching the agent's trajectory distribution $q(x_{1:T})$ to the sequence distribution $p_\theta(x_{1:T})$ of the video model by minimizing the KL-divergence between the two distributions [250, 90]:

$$\begin{aligned}\mathrm{KL}\big[q(x_{1:T})\;\big\|\;p(x_{1:T})\big] &= \underbrace{\mathrm{E}_q\big[-\ln p_\theta(x_{1:T})\big]}_{\text{cross-entropy}} - \underbrace{\mathrm{H}\big[q(x_{1:T})\big]}_{\text{entropy}}\\ &= -\sum_{t=1}^{T}\Big[\mathrm{E}_q\big[\underbrace{\ln p_\theta(x_{t+1}\mid x_{1:t})}_{r_t^{\text{VIPER}}}\big] + \underbrace{\mathrm{H}\big[q(x_{t+1}\mid x_{1:t})\big]}_{\text{exploration term}}\Big],\end{aligned} \tag{6.5}$$

The KL objective shows that for the agent to match the video distribution under the video prediction model, it has to not only maximize the VIPER reward but also balance this reward while maintaining high entropy over its input sequences [116]. In the reinforcement learning literature, the entropy bonus over input trajectories corresponds to an exploration term that encourages the agent to explore and inhabit a diverse distribution of sequences within the regions of high probability under the video prediction model. This results in the final reward function that the agent maximizes:

$$r_t^{\text{KL}} \doteq r_t^{\text{VIPER}} + \beta\, r_t^{\text{expl}}, \tag{6.6}$$

where β determines the amount of exploration. To efficiently compute the likelihood reward, we approximate the context frames with a sliding window as discussed in Equation (6.2):

$$r_t^{\text{VIPER}} \approx \ln p_\theta(x_{t+1}\mid x_{\max(1,t-k):t}). \tag{6.7}$$

Figure 6.1 shows the process of computing rewards using log probabilities under the video prediction model. VIPER is agnostic to the choice of exploration reward and in this paper we opt for Plan2Explore [223] and RND [29].

Data Curation

In this work, we explore whether VIPER provides adequate reward signal for learning low-level control. We utilize the video model likelihoods provided by an autoregressive video prediction model pre-trained on data from a wide variety of environments. We curate data by collecting expert video trajectories from task oracles and motion planning algorithms with access to state information. Fine-tuning large text-conditioned video models [214, 269, 230] on expert videos would likely lead to improved generalization performance beyond the curated dataset, and would make for an interesting future research direction. We explore the favorable generalization capabilities of video models trained on small datasets, and explore how this leads to more general reward functions in Section 6.4.

6.4 Experiments

We evaluate VIPER on 28 different tasks across the three domains shown in Figure 6.2. We utilize 15 tasks from the DeepMind Control (DMC) suite [259], 7 tasks from the Atari Gym suite [24], and 6 tasks from the Robot Learning Benchmark (RLBench) [115]. We compare VIPER agents to variants of Adversarial Motion Priors (AMP) [191], which uses adversarial training to learn behaviors from reward-free and action-free expert data. All agents are trained using raw pixel observations from the environment, with no access to state information or task rewards. In our experiments, we aim to answer the following questions:

1. Does VIPER provide an adequate learning signal for solving a variety of tasks? (Section 6.4)
2. Do video models trained on many different tasks still provide useful rewards? (Section 6.4)
3. Can the rewards generalize to novel scenarios where no expert data is available? (Section 6.4)
4. How does the video model data quality and quantity affect the learned rewards? (Section 6.4)
5. What implementation details matter when using video model likelihoods as rewards? (Section 6.4)

Video Model Training Details

Training data To collect expert videos in DMC and Atari tasks, **Task Oracle** RL agents are trained until convergence with access to ground truth state information and task rewards. After training, videos of the top k episodes out of 1000 episode rollouts for each task are sampled as expert videos, where $k = 50$ and 100 for DMC and Atari, respectively. For RLBench, a sampling-based motion planning algorithm with access to full state information is used to gather 100 demos for each task, where selected tasks range from “easy” to “medium”.

Video model training We train a *single* autoregressive video model for each suite of tasks. Example images for each suite are shown in Appendix E.5. For DMC, we train a single VQ-GAN across all tasks that encodes 64×64 images to 8×8 discrete codes. Similarly for RLBench and Atari, we train VQ-GANs across all tasks within each domain, but encode 64×64 images to 16×16 discrete codes. In practice, the level of VQ compression depends on the visual complexity of the environment – more texturally simple environments (e.g., DMC) allow for higher levels of spatial compression. We follow the original VQ-GAN architecture [66], consisting of a CNN encoder and decoder. We train VQ-GAN with a batch size of 128 and learning rate 10^{-4} for 200k iterations.

To train our video models, we encode each frame of the trajectory and stack the codes temporally to form an encoded 3D video tensor. Similar to VideoGPT [292], the encoded VQ codes for all video frames are jointly modeled using an autoregressive transformer in raster scan order. For DMC and

Atari, we train on 16 frames at a time with a one-hot label for task conditioning. For RLBench, we train both single-task and multi-task video models on 4 frames with frame skip 4. We add the one-hot task label for the multi-task model used in our cross-embodiment generalization experiments in Section 6.4. We perform task conditioning identical to the class conditioning mechanism in VideoGPT, with learned gains and biases specific to each task ID for each normalization layer.

Figure 6.4 shows example video model rollouts for each domain. In general, our video models are able to accurately capture the dynamics of each environment to produce probable futures. Further details on model architecture and hyperparameters can be found in Appendix E.2. All models are trained on TPUv3-8 instances which are approximately similar to 4 Nvidia V100 GPUs.

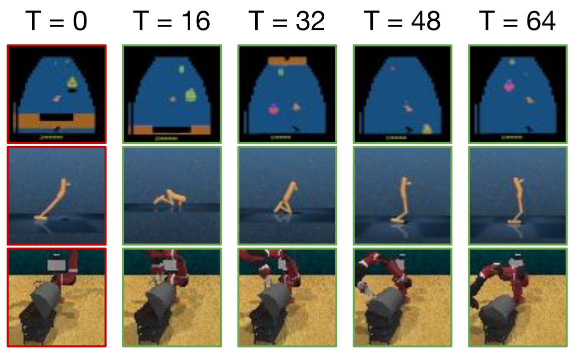


Figure 6.4: Video model rollouts for 3 different evaluation environments.

Video Model Likelihoods as Rewards for Reinforcement Learning

To learn behaviors from expert videos, we provide reward signals using Equation (6.6) in VIPER and the discriminator error in AMP. Both VIPER and AMP are agnostic to the choice of RL algorithm; but, in this paper, we evaluate our approach and baselines with two popular RL algorithms: DrQ [133] and DreamerV3 [91]. We use Random Network Distillation [29] (model-free) as the exploration objective for DrQ, and Plan2Explore [223] (model-based) for DreamerV3. Hyperparameters for each choice of RL algorithm are shown in Appendix E.3. We compare VIPER to two variants of the AMP algorithm: the single-task variant where only expert videos for the specific task are provided to the agent, and the multi-task case where expert videos for all tasks are provided to the agent.

As outlined in Algorithm 3, we compute Equation (6.6) for every environment step to label the transition with a reward before adding it to the replay buffer for policy optimization. In practice, batching Equation (6.6) across multiple environments and leveraging parallel likelihood evaluation leads to only a small decrease in training speed. DMC agents were trained using 1 Nvidia V100 GPU, while Atari and RLBench agents were trained using 1 Nvidia A100 GPU.

We first verify whether VIPER provides meaningful rewards aligned with the ground truth task rewards. Figure 6.5 visualizes the ground truth task rewards and our log-likelihood rewards (Equation (6.7)) for a reference (expert) trajectory and a random out-of-distribution trajectory. In the reward curves on the left, VIPER starts to predict high rewards for the expert transitions once the transitions becomes distinguishable from a random trajectory, while it consistently outputs low rewards for out-of-distribution transitions. The return plot on the right clearly shows positive correlation between returns of the ground truth reward and our proposed reward.

Then, we measure the task rewards when training RL agents with predicted rewards from VIPER and baselines and report the aggregated results for each suite and algorithm in Figure 6.3. The full results can be found in Appendix E.4.

In DMC, VIPER achieves near expert-level performance from pixels with our video prediction rewards alone. Although VIPER slightly underperforms Task Oracle, this is surprising as the Task Oracle uses *full state information* along with *dense task rewards*. VIPER outperforms both variants of AMP. Worth noting is the drastic performance difference between the single-task and multi-task AMP algorithms. This performance gap can possibly be attributed to mode collapse, whereby the discriminator classifies all frames for the current task as fake samples, leading to an uninformative reward signal for the agent. Likelihood-based video models, such as VIPER, are less susceptible to mode collapse than adversarial methods.

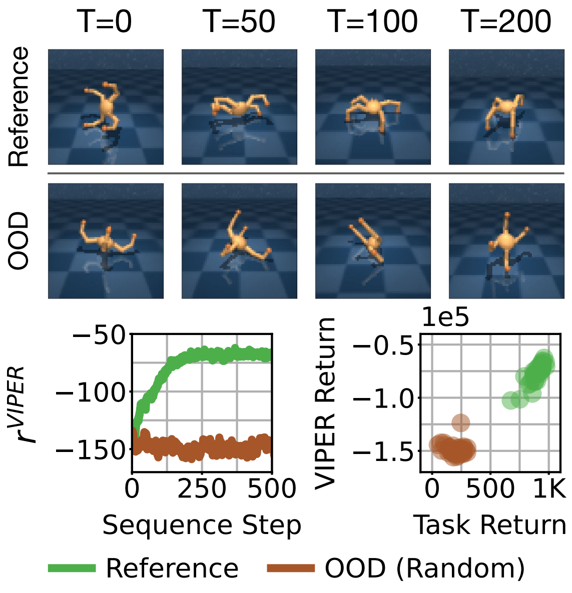


Figure 6.5: VIPER incentivizes the agent to maximize trajectory likelihood under the video model. As such, it provides high rewards for reference (expert) sequences, and low rewards for unlikely behaviors. Mean rewards r^{VIPER} and returns are computed across 40 trajectories.

In Atari, VIPER approaches the performance of the Task Oracle trained with the original sparse task reward, and outperforms the AMP baseline. Shown in Figure 6.6, we found that masking the scoreboard in each Atari environment when training the video model improved downstream RL performance.

Since the video model learns to predict all aspects of the data distribution, including the scoreboard, it tends to provide noisier reward signal during policy training when the agent encounters out-of-distributions scores not seen by the video model. For example, expert demos for Pong only contain videos where the player scores and the opponent's score is always zero. During policy training, we observe that the Pong agent tends to exhibit more erratic behaviors as soon as the opponent scores at least one point, whereas when masking the scoreboard, learned policies are generally more stable. These results suggest the potential benefits of finetuning large video models on expert demos to learn more generalizable priors, as opposed to training from scratch.

For RLBench, VIPER outperforms the Task Oracle because RLBench tasks provide very sparse rewards after long sequences of actions, which pose a challenging objective for RL agents. VIPER instead provides a dense reward extracted from the expert videos, which helps learn these challenging tasks. When training the video model, we found it beneficial to train at a reduced frame rate, accomplished by subsampling video sequences by a factor of 4. Otherwise, we observed the video model would assign high likelihoods to stationary trajectories, resulting in learned policies rarely moving and interacting with the scene. We hypothesize that this may be partially due to the high control frequency of the environment, along with the initial slow acceleration of the robot arm in demonstrations, resulting in very little movement between adjacent frames. When calculating likelihoods for reward computation, we similarly input observations strided by time, e.g., $p(x_t \mid x_{t-4}, x_{t-8}, \dots)$.

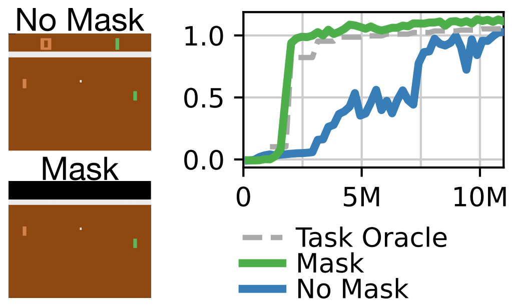


Figure 6.6: RL training curves on Atari Pong when using VIPER trained with or without masking the scoreboard.

Generalization of Video Prediction Rewards

Prior works in the space of large, pretrained generative models have shown a powerful ability to generalize beyond the original training data, demonstrated by their ability to produce novel generations (e.g., unseen text captions for image or video generation [214, 269, 230]) as well as learn more generalizable models from limited finetuning data [50, 283]. Both capabilities provide promising directions for extending large video generation models to VIPER, where we can leverage text-video models to capture priors specified by text commands, or finetune on a small set of demonstrations to learn a

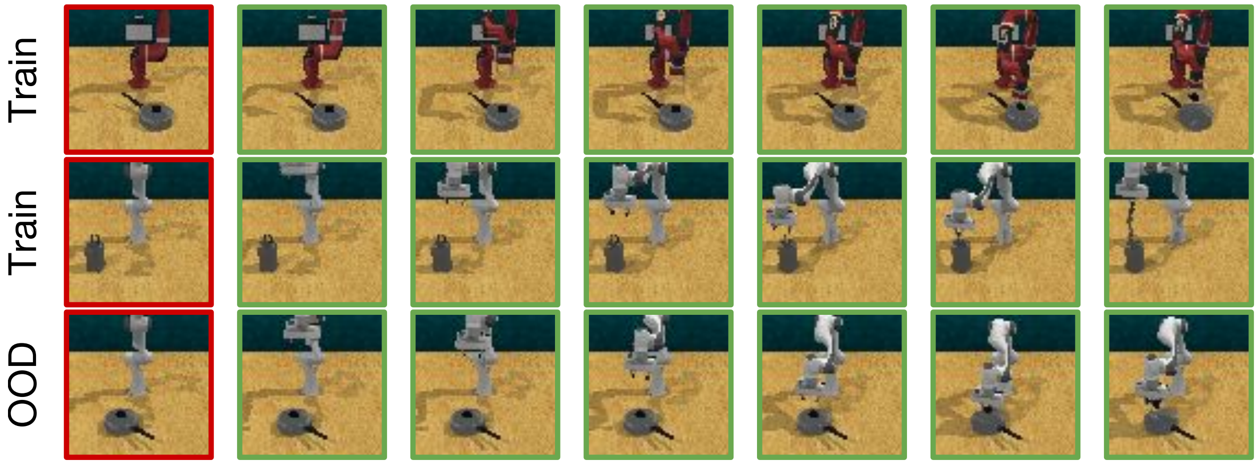


Figure 6.7: Sampled video predictions for in distribution reference videos (Train) and an OOD arm/task combination (OOD). The video model displays cross-embodiment generalization to arm/task combination not observed in the training data. Video model generalization can enable specifying new tasks where no reference data is available.

task-specific prior that better generalizes to novel states.

In this section, we seek to understand how this generalization can be used to learn more general reward functions. We train a model on two datasets of different robot arms, and evaluate the *cross-embodiment generalization* capabilities of the model. Specifically, we gather demonstrations for 23 tasks on the Rethink Robotics Sawyer Arm, and demonstrations for 30 tasks on the Franka Panda robotic arm, where only 20 tasks are overlapping between arms. We then train a task-conditioned autoregressive video model on these demonstration videos and evaluate the video model by querying unseen arm/task combinations, where a single initial frame is used for open loop predictions.

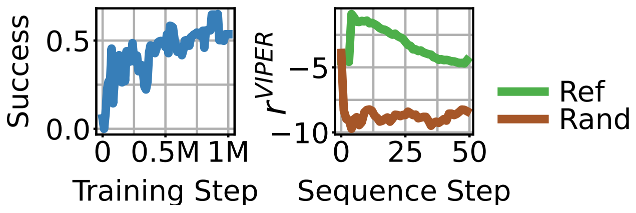


Figure 6.8: (Left) Training curve for RL agent trained with VIPER on OOD task. (Right) Task-conditional likelihood for reference and random trajectory for an OOD task.

Sample video model rollouts for in distribution training tasks and an OOD arm/task combination are shown in Figure 6.7. Even though the video model was not directly trained on demonstrations of the Franka Panda arm to solve the saucepan task in RLBench, it is able to generate reasonable trajectories for the arm and task combination. Figure 6.8 further validates this observation by assigning higher likelihood to expert videos (Ref) compared to random robot trajectories (Rand). We observe that these generalization capabilities also extend to downstream RL, where we use our trained video model with VIPER to learn a policy for the Franka Robot arm to solve an OOD task without requiring demos for that specific task and arm combination. Figure E.10 further extends this analysis to include more in distribution and out of distribution tasks. These results demonstrate a promising direction for future work in applying VIPER to larger scale video models that will be able to better generalize and learn desired behaviors only through a few demonstrations.

Impact of Data Quality and Quantity

Learning from sub-optimal data is an important feature of learned reward models, as large-scale video data may often contain suboptimal solutions to a given task. As such, we evaluate VIPER's ability to learn rewards from sub-optimal data.

We train video models with suboptimal (good) data, which has 50-75% returns of the expert data. In Figure 6.9, VIPER learns the suboptimal behaviors provided in the suboptimal video data. This suboptimal VIPER can be still useful if combined with a sparse task reward, comparable to an agent learned from dense task rewards.

Additionally, we evaluate how VIPER performs under different video dataset sizes. As shown in Figure 6.10, VIPER can learn a meaningful reward function only with one expert trajectory, although adding more videos quickly improves the performance of VIPER, with diminishing returns after 50 trajectories.

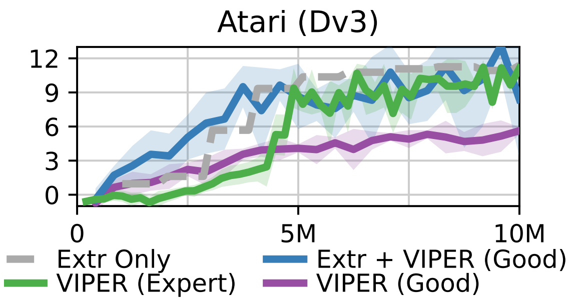


Figure 6.9: Atari performance with VIPER models trained on suboptimal data.

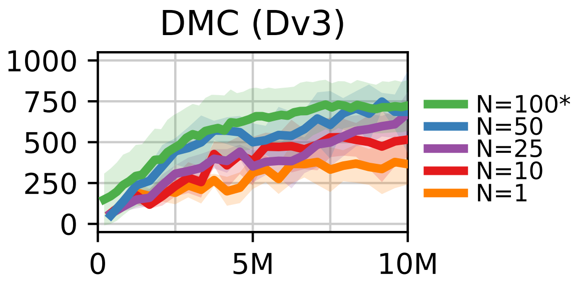


Figure 6.10: DMC performance with VIPER trained on different dataset sizes, where N is the number of expert trajectories. *Original dataset size.

Ablation Studies

In this section, we study the contributions of various design decisions when using video prediction models as rewards: (1) how to weight the exploration objective, (2) which video model to use, and (3) what context length to use. Ablation studies are performed across DMC tasks.

Exploration objective As discussed in section 6.3, an exploration objective may help the RL agent learn a distribution of behaviors that closely matches the distribution of the original data and generalizes, while preventing locally optimal solutions. In Figure 6.11, we ablate the β parameter introduced in Equation 6.6 using a VideoGPT-like model as the VIPER backbone. $\beta = 0$ corresponds to no exploration objective, whereas $\beta = 1$ signifies equal weighting between r^{VIPER} and r^{expl}. We ablate the exploration objective using the Plan2Explore [223] reward, which provides the agent with a reward proportional to the disagreement between an ensemble of one-step dynamics models. Using no exploration objective causes the policy's behavior to collapse, while increasing the weight of the exploration objective leads to improved performance.

Video model Although our experiments focus on using an autoregressive VideoGPT-like model to compute likelihoods, VIPER generally allows for any video model that supports computing conditional likelihoods or implicit densities.

Figure 6.11 shows additional ablations replacing our VideoGPT model with a similarly sized MaskGIT [36] model, where frames are modeled with MaskGIT over space, and autoregressive over time. MaskGIT performs substantially worse than the VideoGPT model, which is possibly due to noisier likelihoods from parallel likelihood computation. In addition, while VideoGPT only

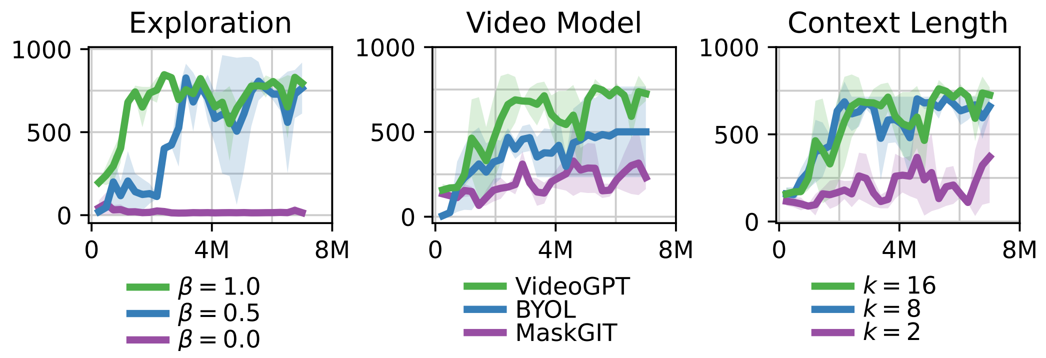


Figure 6.11: Effect of exploration reward term, video model choice, and context length on downstream RL performance. An equally weighted exploration reward term and longer video model context leads to improved performance. MaskGIT substantially underperforms VideoGPT as a choice of video model. The BYOL model performs moderately due to the deterministic architecture not properly handling multi-modality.

requires 1 forward pass to compute likelihood, MaskGIT requires as many as 8 forward passes on the sequence of tokens for a frame, resulting in an approximately $8\times$ slowdown in reward computation, and $2.5\times$ in overall RL training.

Finally, we evaluate the performance of a video model that computes implicit densities using a negative distance metric between online predictions and target encodings with a recurrent Bootstrap Your Own Latent (BYOL) architecture [84, 82]. We refer the reader to Appendix E.1 for more details about the implementation. While BYOL outperforms MaskGIT in Figure 6.11, its deterministic recurrent architecture is unable to predict accurate embeddings more than a few steps into the future. This limits the ability of the learned reward function to capture temporal dynamics. We observe that agents trained with BYOL often end up in a stationary pose, which achieves high reward from the BYOL model.

Context length The context length k of the video model is an important choice in determining how much history to incorporate into the conditional likelihood. At the limit where $k = 0$, the reward is the unconditional likelihood over each frame. Such a choice would lead to stationary behavior on many tasks. Figure 6.11 shows that increasing the context length can help improve performance when leveraging a VideoGPT-based model for downstream RL tasks, subject to diminishing returns. We hypothesize that a longer context length may help for long-horizon tasks.

6.5 Conclusion

This paper presents Video Prediction Rewards (VIPER), a general algorithm that enables agents to learn behaviors from videos of experts. To achieve this, we leverage a simple pre-trained autoregressive video model to provide rewards for a reinforcement learning agent. These rewards are parameterized as the conditional probabilities of a particular frame given a context past of frames. In addition, we include an entropy maximization objective to ensure that the agent learns diverse behaviors which match the video model's trajectory distribution. VIPER succeeds across 3 benchmarks and 28 tasks, including the DeepMind Control Suite, Atari, and the Reinforcement Learning Benchmark. We find that simple data augmentation techniques can dramatically improve the effectiveness of VIPER. We also show that VIPER generalizes to out-of-distribution tasks for which no demonstrations were provided. Moreover, VIPER can learn reward functions for a wide variety of tasks using a single video model, outperforming AMP [191], which suffers from mode collapse as the diversity of the expert videos increases.

Limitations of our work include that VIPER is trained using in-domain data of expert agents, which is not a readily-available data source in the real world. Additionally, video prediction rewards trained on stochastic data may lead the agent to prefer states that minimize the video model's uncertainty, which may lead to sub-optimal behaviors. This challenge may be present for cases where either the environment is stochastic or the demonstrator provides noisy demonstrations. Moreover, selecting the right trade-off between VQCode size and context length has a significant impact on the quality of the learned rewards, with small VQCodes failing to model important components of the environment (e.g., the ball in Atari Breakout) and short context lengths leading to myopic behaviors which results in poor performance. Exploring alternative video model architectures for VIPER would likely be a fruitful research direction.

To improve the generalization capabilities of VIPER, larger pre-trained video models are necessary. Future work will explore how fine-tuning or human preferences can be used to improve video prediction rewards. Using text-to-video models remain another interesting direction for future work in extracting text-conditioned task-specific priors from a pretrained video model. In addition, extending this line of work to leverage Video Diffusion Models [105] as video prediction rewards may lead to interesting outcomes.

6.6 Acknowledgments

This work was supported in part by an NSF Fellowship, NSF NRI #2024675, ONR MURI N00014-22-1-2773, Komatsu, and the Vanier Canada Graduate Scholarship. We also thank Google TPU Research Cloud and Cirrascale (https://cirrascale.com) for providing compute resources.

Chapter 7

Motion-Conditioned Image Animation for Video Editing

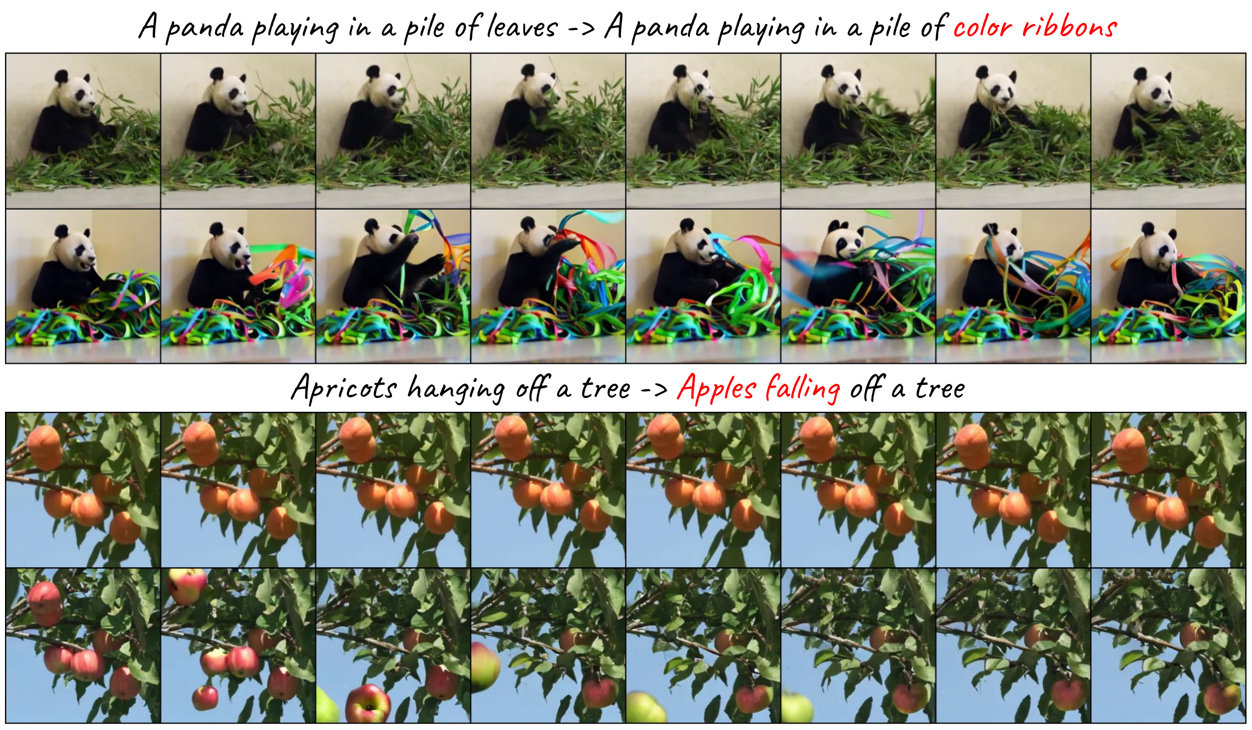

A panda playing in a pile of leaves -> A panda playing in a pile of color ribbons
Apricots hanging off a tree -> Apples falling off a tree

Figure 7.1: **MoCA is able to generate a diverse range of edits, such as object replacement, style changes, and motion edits.** The frames in the top row in each example represent the source video while the bottom ones show the edited frames by MoCA. The source and editing prompts are shown above each example.

7.1 Introduction

Recent advancements in image and video generation models have seen tremendous progress, with existing models able to synthesize highly complex images [202, 204, 210, 214, 37] or videos [269, 229, 20, 103, 79] given textual descriptions. Outside of generating purely novel content, these models have shown to be powerful tools in achieving advanced image and video editing capabilities for downstream content creation.

Given a source video, a caption of the source video, and an editing textual prompt, a video editing method should produce a new video that is aligned with the provided editing prompt while retaining faithfulness to all other non-edited characteristics of the original source video. Video edit types can be broadly split into two main categories of spatial and temporal edits. Spatial edits generally consist of image-based edits extended to video, such as editing a video in the style of Van Gogh, inserting an object into the scene, or changing the background. Due to the added temporal dimension in video, we can also change the underlying motion of the object, such as making a panda play in a pile of ribbons, or replacing apricots in a video with apples and making them fall off a tree (see Figure 7.1).

Current methods in video editing focus more on spatial editing problems while ignoring the motion editing problem. Proposed methods leverage pre-trained text-to-image or video models for editing by further fine-tuning with conditioning on auxiliary information such as depth maps or edge maps [67, 307], fine-tuning for each edit example [287, 177, 213, 126, 73], or exploiting the diffusion process to restrict the generated edits to share similar features and structures with the source content [96, 159, 31, 258, 168, 294, 78]. Notably, most proposed methods either in image or video editing are generally specialized only to a subset of editing tasks and do not perform well on others. For example, methods to utilize depth or edge maps of the video [67, 307] find it more difficult to perform motion edits due to adherence to the original video structure. As such, it becomes important to assess video editing capabilities across a wide range of different edits in order to better understand their advantages and disadvantages.

In this paper, we present two key contributions for video editing. First, we introduce a simple yet strong approach for video editing, **Mo**tion-**C**onditioned Image **A**nimation (MoCA), that decomposes the problem into image editing and image animation. We first use the existing image editing methods to edit the first video frame, then produce an edited video using a motion-conditioned image animation model. We use an optical flow representation of the source video using a pretrained RAFT [246] model as the motion conditioning to retain the original motion characteristics of the source video. In the video edits consisting of a motion edit, we drop out this motion conditioning. Through our extensive experiments and human evaluations, we show that this simple baseline outperforms the state-of-the-art video editing models across a wide range of edit types.

Secondly, we introduce a dataset of 250+ video edits that comprehensively covers a wide range of video editing types. We combine existing datasets for video editing, and introduce our own subset of curated videos from YouTube-8M [2] with a stronger emphasis in including motion-based edits due to a general lacking of such examples in current public video editing datasets [288]. Using our combined dataset, we comprehensively benchmark prior video editing methods along a range of pre-categorized edits types, such as style, background, object, and motion-based edits via human evaluation and automatic metrics. Additionally, we perform an analysis of the alignment between the automatic metrics for measuring video editing quality and human judgement.

7.2 Related Work

By the remarkable progress in text conditional image and video generation models, text-guided image and video editing have emerged as key editing tools that enable average users and artists to create new content easily from existing photos or videos. In this section, we will discuss the existing works on diffusion-based text-driven image and video editing and their applicability to various manipulation tasks.

Text-driven Image Editing

Prior works have proposed a variety of methods for text-conditioned image editing. One family of such image editing methods focus on using diffusion models, and produce image edits by altering the backward diffusion process. SDEdit [168] is a simple image editing approach that applies various diffusion noise levels to an input source image, and produces image edits by sampling back out through the diffusion process conditioned on an edit prompt. Plug-and-Play [258] samples edited videos initialized from the DDIM [235] inversion, with selected visual features copied between the source and generated images during diffusion sampling. Prompt-to-Prompt [96] (P2P) enables general edit types (style changes, object replacement, changing texture) through replacing self and cross attention of the generated image with the attention maps of the source image during diffusion sampling. Null-Text Inversion [176] extends P2P to enable better editing performance when editing real images through optimizing null text embeddings to allow for more faithful reconstruction of the source image during diffusion sample.

Another class of image editing techniques that allow for more global changes in visual features are built on ControlNet [307] or T2I-Adapters [178], where pretrained text-to-image models are augmented and finetuned to incorporate conditioning information, such as depth maps or contour maps computed from edge detection algorithms.

Most prior image editing methods are generally constrained structurally, and have a more difficult time producing image edits with large pose changes, such as editing an image of a bird to spread its wings. MasaCtrl [31] achieves this through mutual self-attention, where select self-attention layers in the diffusion networks attend to the keys and values of the corresponding layers of the source image during the diffusion process. Larger pose changes are enabled by only enabling mutual self-attention replacement during later diffusion timesteps. Imagic [126] similarly achieves image edits with larger pose changes through a text embedding optimization and model fine-tuning process.

Lastly, the Instruct-Pix2Pix [26] family of models propose to treat the image editing problem as a supervised learning problem. Core work around these model requires collecting supervised data as pairs of (text editing instructions and images before/after the edit), and fine-tuning a pre-trained text-to-image model on the collected data.

In this paper, we focus on video editing, as it provides a more challenging task in accomplishing complex edits over both space and time.

Text-driven Video Editing

Video editing similarly can be deployed for various manipulation tasks including style transfer, object or scene manipulations, and motion editing. However, these manipulation tasks are more challenging in videos since the generated content should be consistent across frames. Most of the existing works in video editing focus more on the first two manipulation types while ignoring the motion editing problem, as they generally propose video editing methods that leverage pretrained text-to-image models. Pix2Video [33] and FateZero [194] both propose different variants of extending self-attention with cross-frame attention. TokenFlow [78] Rerender a Video [294], and CoDef [187] perform video edits through image editing techniques and propagating edited features temporally using estimated motions by computing temporal inter-frame correspondences [78], optical flow estimation and warping [294], or estimating canonical images and temporal deformation fields using optical flow of the source video [187], respectively.

Methods such as Gen-1 [67], VideoComposer [277], and ControlVideo [310] train text-to-video models with additional conditioning inputs such as depth maps or motion vectors to allow for controllability of general structure in the resulting video edits.

Tune-a-Video [287] and Dreamix [177] both propose fine-tuning a pre-trained diffusion model for each source video, where Tune-a-Video finetunes a pretrained text-to-image model and Dreamix fine-tunes a pre-trained text–to-video model. For both methods, edits are produced using the fine-tuned video model to sample back out conditioned on given the edit prompts.

Most prior works tend to target specific types of edits and evaluate on their own constructed sets of edit prompts. As such, the benefits of each method across different kinds of edits are less clear, and motivates us to propose a benchmark centered around a more rigorous analysis of the pros and cons of each method. In addition, we propose our own video editing method that leverages existing text conditional image editing models to edit the first frame of a source video and extrapolate its future frames via a motion conditional video generation diffusion model to enable alignment of the edit with the source video and the editing prompt.

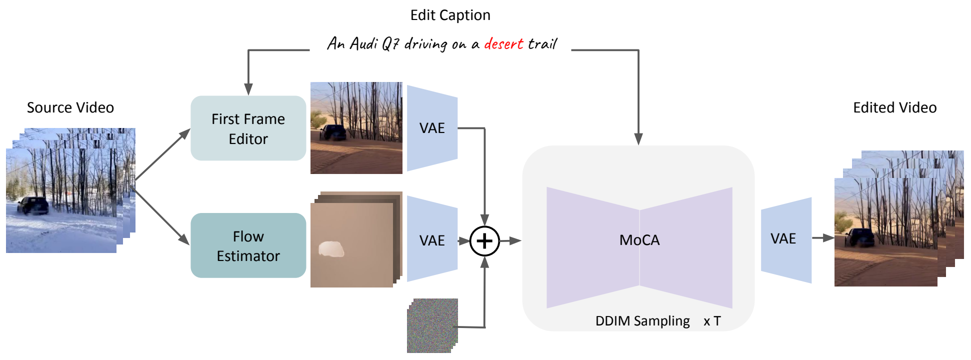


Figure 7.2: **An overview of MoCA.** Given a source video, we compute its optical flow, and apply image editing techniques on the first frame. To produce the resulting video edit, we sample our model conditioned on motion, the edited first frame, and the edit caption. For motion-based edits, we dropout the optical flow conditioning.

7.3 Background

Conditional latent diffusion models. Diffusion models learn to generate samples from a training distribution by reversing a gradual noising process. At the sampling time starting from a Gaussian noise, the model generates less noisy samples in T time-steps where each time-step, t, corresponds with a specific noise level [55]. In latent diffusion models for each input image x, this noising and denoising process is applied on the latent space, $z = \mathcal{E}(x)$, of a pretrained variational autoencoder with encoder $\mathcal{E}$, resulting in more efficient training and sampling steps. In a text-conditional latent diffusion model, text features are extracted from a pre-trained language model, and then fed into the latent diffusion U-Net blocks via cross-attention modules. In addition to the text conditioning, the image or video generation models can be also conditioned on an additional input image by concatenating its features with the noisy latent features at each time-step, z_t, and adding extra input channels to the first convolutional layer of the U-Net [26, 79, 304]. The network will be trained to predict the noise added to the noisy latent features given image and text conditioning inputs, respectively.

Classifier free guidance. Classifier-free guidance was proposed in [97] and is widely used to improve the fidelity and diversity of the generated samples and their correspondence with the conditioning input in a diffusion model. During training, the diffusion model is trained jointly in a conditional and unconditional setting where the conditioning input is set to NULL with a specific frequency. At inference, the generated samples are guided to be more faithful to the conditioning input while being further away from the NULL input with a guidance scale $s >= 1$.

7.4 MoCA

Inspired by the success of image conditioning for video generation [79, 304] and text-driven image editing methods [96, 256, 26, 213, 307], we introduce a simple yet strong baseline for text-driven video editing that can be deployed in a wide range of video editing applications. We decompose the video editing problem into image editing, and motion-conditioned image animation. We choose to adopt this decomposition, as (1) image editing has shown large success, with the growing availability of more capable image editors that are able to edit a variety of complex editing prompts, and (2) the recent success of image animation methods for video generation to model temporal dynamics in videos [79, 304]. As a simple approach, we could first leverage image editing techniques to edit the first frame given the editing prompt, and then use an image animation model to predict the rest of the frames. However, this typically results in videos that diverge from the motion of the original source video, which is important to retain especially for edits that only target to change the style, or background of the video. Therefore, for these cases, we propose to additionally condition on a motion representation of the video. When editing video motion, we dropout conditioning on the motion and predict future frames conditioned only on the edited image. An overview of MoCA is presented in Figure 7.2.

To achieve this, we train a latent diffusion video generation model conditioned on (1) a text prompt, (2) an image as the first frame of each video, and (3) an optical flow representing the motion in the video. We use a variational autoencoder (VAE) pre-trained on an internal image database to encode an input video with shape $T \times 3 \times H \times W$ frame-wise to a tensor of shape $T \times C \times H' \times W'$. We learn the diffusion process on this latent space and finally decode the latent to the video pixel space via its decoder. We use a pre-trained Flan-T5-XXL [162] text encoder to extract text features, c_T, and feed them into the model via cross-attention modules. At inference time, we edit the first frame of the source video via off-the-shelf text-based image editing models [96, 168] and use that as the image conditioning input while using the optical flow estimation of the source video as the motion conditioning. In the case where motion edits are desired, we dropout the motion-conditioning.

Image Editing We leverage a diverse range of existing image editing methods, and find certain editing methods to be useful for specific editing types. We have found Prompt-to-Prompt [96] effective for style, background, and object replacement edits, and SDEdit [307] for multi-spatial edits that contain larger feature and pose changes. For motion-only edits, we keep the original source frame.

Image conditioning. Inspired by Emu-video [79], to condition the video generation model on the first frame as an input, c_I, we encode the first frame using the same auto-encoder, $\mathcal{E}(c_I)$, repeat it T times to have the same shape as the video latent features, and then concatenate it channel-wise with the noisy latent features at each time-step.

Motion conditioning. To condition the video generation model on the optical flow motion representation, c_M, we convert the optical flow into an RGB video, encode each frame using the same auto-encoder, $\mathcal{E}(\text{toRGB}(c_M))$ and concatenate it channel-wise with the noisy latent and image conditioning features at each time-step. Since conversion to RGB re-normalizes the frames, we additionally compute an average flow magnitude term, and condition the video model on it. This conditioning is performed similar to the diffusion time-step conditioning.

Classifier free guidance for three conditionings. Similar to [26, 79], we leverage classifier-free guidance with respect to all three conditioning inputs to control faithfulness to each of the inputs at inference time. During training, we randomly set each of the individual conditioning inputs and their pairwise combination to NULL for 10% of the examples, respectively. To train for both motion-conditioned image animation, and image animation only, we train with 50% dropout on motion conditioning. We have therefore, three guidance scales for the text, image, and motion conditioning inputs that we adjust based on the editing application. During inference, we use the following conditioning order to compute the classifier guidance, where v_θ is the output from the U-Net, and $\tilde{v}_\theta$ is used to denoise the input image:

$$\begin{aligned}
\tilde{v}_\theta(z_t, c_M, c_T, c_I) = {} & v_\theta(z_t, \varnothing, \varnothing, \varnothing) \\
& + s_I \cdot (v_\theta(z_t, \varnothing, \varnothing, c_I) - v_\theta(z_t, \varnothing, \varnothing, \varnothing)) \\
& + s_T \cdot (v_\theta(z_t, \varnothing, c_T, c_I) - v_\theta(z_t, \varnothing, \varnothing, c_I)) \\
& + s_M \cdot (v_\theta(z_t, c_M, c_T, c_I) - v_\theta(z_t, \varnothing, c_T, c_I))
\end{aligned}$$

7.5 Experiments

Implementation Details

Our video generation model is built from a text-to-image U-Net based latent diffusion model pre-trained on our internal database of 400M (image, text) pairs. Similar to earlier works in video generation [229, 20, 79], we expand this 2D U-Net to video generation by adding temporal modules consisting of 1D temporal convolution layers and 1D temporal attention blocks after each spatial convolution and attention block, respectively.

We initialize all the spatial parameters from the pre-trained text-to-image model and fine-tune all the temporal and spatial layers of our video prediction model, with 1.4B trainable parameters, on an internal licensed dataset consisting of 34M pairs of video-text samples. We sample random 256×256 2-second clips from each video using a frame rate of 4 frames per second with the first frame as the conditioning image. Videos are encoded via the pre-trained VAE to the $4 \times 8 \times 32 \times 32$ resolution. We additionally use RAFT [246] to extract an estimated optical flow representation for each video during training. Similar to [79], we train our model using zero terminal-SNR and v-prediction, on a batch size of 512 split across 32 A100 GPUs. During inference we use 64 DDIM steps for sampling.

Evaluation Dataset

	LOVEU TGVE	**Dreamix Dataset**	**Our Dataset**
Style	35	1	11
Background	35	1	7
Object	35	7	14
Motion	0	2	68
Multi-Spatial	35	0	0
Multi-Motion	0	0	17
Total	140	14	117
# Unique Videos	35	9	37
Avg Edits Per Video	4	1.56	3.16

Table 7.1: **Details for each individual dataset in the VideoEdit benchmark, as well the total distribution of edits types.** Our custom curated dataset focuses more heavily on motion editing due to the general lack of motion edits available in the combined LOVU-TGVE and Dreamix datasets.

We introduce a dataset of 271 edit tasks, defined as a set of (source video, edit prompt) pairs designed to comprehensively evaluate and benchmark video editing capabilities of current methods. Our dataset consists of a combination of existing video editing datasets, as well as our own curated subset:

- **LOVEU-TGVE Dataset** [288]: comprises of 35 source videos, with 4 different manipulation tasks proposed for each video (140 edits total). We filtered out videos with human faces and hands.
- **Dreamix Dataset** [177]: consists of 14 videos downloaded from the dreamix paper website, with edits primarily focusing on scene changes with motion.
- **Our Custom Dataset**: we curate an additional 37 videos from YouTube-8m [2], focused on including a diverse range of motion edits as well as a composition of scene and motion edits (117 edits total).

We group each edit task into one of following edit types, some of which are explored in [288, 177]:

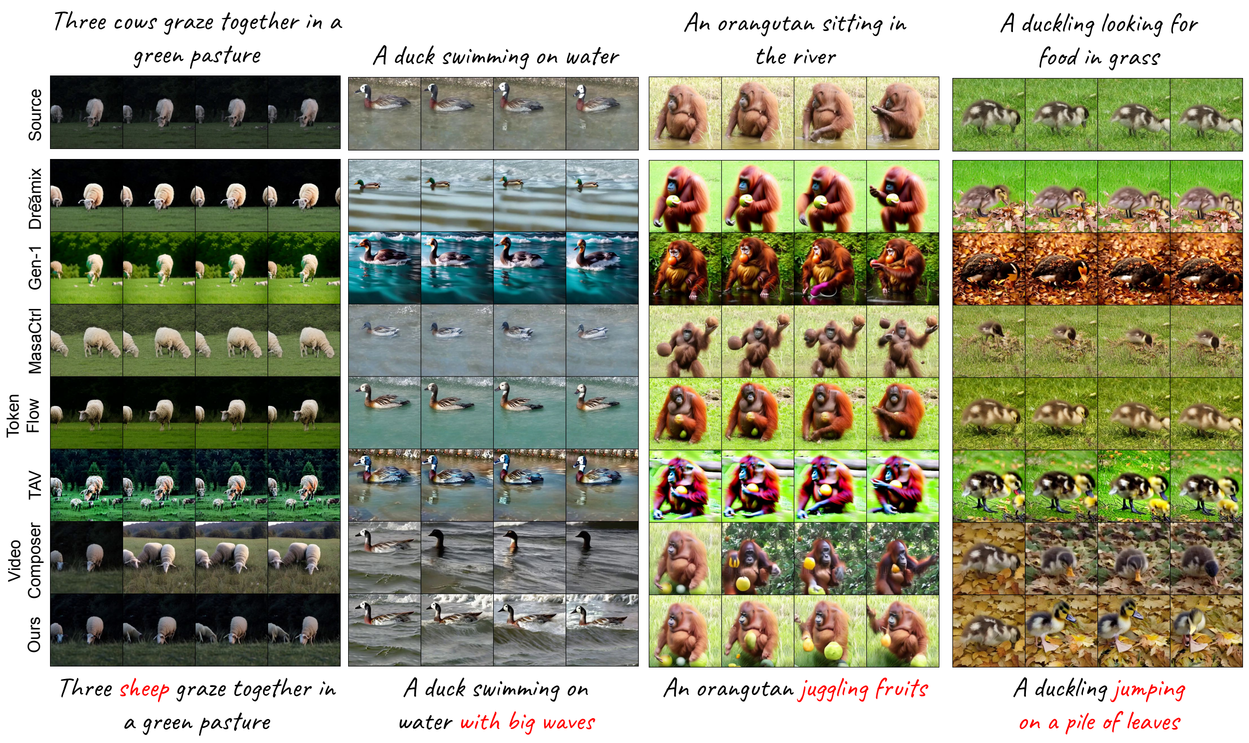

Three cows graze together in a green pasture
A duck swimming on water
An orangutan sitting in the river
A duckling looking for food in grass
Source
Dreamix
Gen-1
MasaCtrl
Token Flow
TAV
Video Composer
Ours
Three sheep graze together in a green pasture
A duck swimming on water with big waves
An orangutan juggling fruits
A duckling jumping on a pile of leaves

Figure 7.3: **Comparison of our method against baselines for a given video editing task.** Our method is able to accurately edit both the spatial and temporal properties of the source video.

- *Style*: changes in the composition of the video, such as making the video reflect a specific artistic style (crayon drawing, oil painting, impressionism),
- *Object*: adding or replacing objects in the scene, such as replacing a lion with a zebra, or placing a hat on a person's head,
- *Background*: changes in the background scene of the video, such as replacing a snowy mountain background with a desert,
- *Motion*: changes in the motion of an entity compared to the source video, such as making a monkey jump, or a car turn in a different direction,
- *Multi-Spatial*: a combination of style, object, and background changes in the video,
- *Multi-Motion*: a combination of style, object, and background changes in addition to a motion change.

Table 7.1 shows a break-down of the number of videos and edits of each type for each dataset.

Baselines

We compare against a set of SOTA baselines to comprehensively target different families of video editing models.

- **TokenFlow** is a tuning-free text-to-image based editing model [78]. It leverages a pre-trained text-to-image diffusion model to edit videos by computing and propagating spatial edits across temporal correspondences found in the original source videos. We use the public repo [1] to run this baseline.

- **Tune-a-Video** is a fine-tuning text-to-image based editing model [287]. For each edit task, Tune-a-Video extends a pre-trained text-to-image diffusion model to the temporal domain and fine-tunes the weights on a given source video. During inference, the edit result is generated through a DDIM initialized sampling using the fine-tuned model conditioned on the edit prompt. We use the public repo [2] to run this baseline.

- **Dre͠amix** is a fine-tuning text-to-video based editing model [177]. It fine-tunes a text-to-video model for *each source video*, and edits are generated by sampling at different levels of noise strengths, conditioned on the edit prompt. Due to the lack of public code and models, we perform Dreamix fine-tuning using our own internal text-to-video model (as indicated by the tilde). Our text-to-video model follows the same training parameters and model architecture (LDM) as MoCA, only without the initial concatenating for image and motion conditioning.

- **MasaCtrl** is a tuning-free text-to-video based editing model [31]. Originally presented in the text-to-image domain, MasaCtrl enables more robust structural (e.g. pose) changes in image edits compared to the prior methods. It replaces self-attention layers with mutual self-attention to query correlated local structures and textures from a source image. We extend MasaCtrl to the video domain using our text-to-video model (same as that of the Dre͠amix baseline). Importantly here, we found it crucial to only apply mutual self-attention layers in the spatial, and not temporal transformer layers of our network.

- **Gen-1** is a tuning-free video editing model [67]. Gen-1 video-to-video model generates a video conditioned on a given edit prompt and a depth map of the source video. We use the public web interface in generating edit results.

- **VideoComposer** is a tuning-free text-to-video generation model [277]. VideoComposer is a motion-conditioned video model that can generate videos conditioned on a single image, and desired motion extracted from the source video. We use the public repo [3] to run this baseline. When generating edits, we condition VideoComposer on the same edited image as given to our method.

[1]`https://github.com/omerbt/TokenFlow`
[2]`https://github.com/showlab/Tune-A-Video`
[3]`https://github.com/damo-vilab/videocomposer`

	Style	BG	Object	Motion	M-S	M-M	Total
Dre͠amix [177]	53%	53%	63%	81%	49%	65%	63%
Gen-1 [67]	66%	40%	80%	99%	57%	90%	74%
MasaCtrl [31]	74%	72%	80%	76%	71%	65%	75%
Tune-a-Video [287]	53%	67%	70%	86%	74%	85%	72%
TokenFlow [78]	70%	77%	63%	83%	83%	85%	76%
VideoComposer [277]	78%	76%	92%	84%	74%	85%	82%

Table 7.2: **Human evaluation results for preference of our method over each of the baselines.** User ratings generally show greater preference for our method, with the exception of Gen-1 for background edits, and Dre͠amix for multi-spatial edits. BG is Background, M-S is Multi-Spatial, and M-M is Multi-Motion.

All baselines are run in their native resolutions and frame rates, and spatio-temporally down-sampled to 256 resolution, 4 frames per second for fair comparisons to our method. For each method and (video, edit prompt) pair, we perform a hyper-parameter sweep to generate 10 candidate edits, and use human evaluators to select the best edit. The hyper-parameter sweeps vary for each method, with some over one to three hyper-parameters while still restricted to the same max 10 candidate generations.

Human Evaluation

Methodology We use crowd-sourced workers from Amazon Mechanical Turk (AMT) for our human evaluations. For each task given the source video and the editing prompt, evaluators are asked to perform a binary selection on their preferred video edit out of two given edits, one from our proposed method. Inspired by the JUICE metric introduced in [79], they are also required to choose the reasoning for their selection as either a better consistency with the source video or a higher alignment with the editing prompt or both. The same task is given to five different evaluators, and the overall preferred video edit is selected through a majority vote. We evaluate paired comparisons between our method and all given baselines, and report the final metrics as the percentage of video edit examples for which our method is preferred.

Results Table 7.2 shows human evaluation results comparing our method against each of the baselines, partitioned by edit type. A value of 50% means that both methods perform equally, with values greater than 50% showing a stronger human preference towards the edits produced by our method. Evaluators significantly preferred our method over all other baselines. When examining results split by edit type, human raters showed a larger gap in preference for our method's motion edits, with a more narrow gap on spatial edits. Gen-1 notably shows capabilities in background edits, as seen by the 40% preference for our method, and 60% for Gen-1. We hypothesize that

		Style	**BG**	**Object**	**Motion**	**M-S**	**M-M**	**Total**
ImageCLIP	M_{sim}	45%	43%	47%	63%	44%	50%	51%
	M_{dir}	80%	72%	74%	53%	81%	66%	68%
	M_{geo}	78%	71%	70%	50%	82%	67%	67%
VideoCLIP	M_{sim}	42%	44%	53%	74%	40%	51%	55%
	M_{dir}	78%	74%	77%	59%	76%	73%	72%
	M_{geo}	79%	72%	77%	56%	71%	78%	69%

Table 7.3: **Classification accuracy of each CLIP-based automatic metric**, considering binary human decisions comparing MoCA edits against different baselines as the ground truth labels. Note that random guessing achieves roughly 50% accuracy. M_{dir} and M_{geo}, standing for CLIP text-video directional and geometric similarity scores, show relatively high accuracy (up to 80%) on spatial-based edits, such as style, background, object, and multi-spatial. However, both methods have a much harder time selecting the correct motion-based edits.

since background edits require larger visual feature deviations from the original source video, Gen-1 performs well on this task by generating high quality videos. However, it is less preferred on other video edit types since it does not preserve the visual features and style of the source video due to only conditioning on depth maps.

Dre͡amix shows similarly competitive results in the spatial edits, but struggles more on the motion edits. We found that Dre͡amix has a tendency to overfit to the motion of the source video, even when adjusting the number of fine-tuning steps. MasaCtrl shows strong motion-editing abilities, but struggles more with spatial edits, especially those with larger feature changes, due to its reliance on mutual self-attention on visual features of the source content. Tune-a-Video has strong spatial editing capabilities, but generates less temporally coherent motion, and performs poorly on the motion edits. Similarly, TokenFlow has a reasonable performance in spatial edits, but struggles with motion edits due to its reliance on a pre-trained text-to-image model. Lastly, VideoComposer generally struggles to remain faithful to the image conditioning input, or produces less temporally coherent motions.

Lastly, Figure 7.4 shows the distribution of factors selected in which human raters preferred our method over baselines. In general human raters preferred MoCA due to its stronger alignment with the edit prompt. VideoComposer shows a slightly different distribution, of which we hypothesize may be due to the fact that it would generally produce videos with high text-video alignment, but may deviate far from the source or edited image, thus the higher distribution in selecting consistency with the source video as a deciding factor. An example video edit by all models is shown in Figure 7.3.

Automatic Evaluation

In addition to presenting human evaluation results, we also investigate automatic evaluation metrics using pre-trained Video and Image CLIP models [197, 280]. We perform an analysis over several CLIP-based metrics to measure video editing quality.

CLIP video similarity score. Given a source video V_{source} and an edit result V_{edit}, we first measure faithfulness between the source video and the resulting edited video:

$$\mathrm{M}_{\text{sim}} = \mathcal{E}_V(V_{\text{source}}) \cdot \mathcal{E}_V(V_{\text{edit}})$$

where $\mathcal{E}_V$ is the VideoCLIP encoder.

CLIP text-video directional similarity score. Given a source prompt $\mathrm{T}_{\text{source}}$ and an edit prompt T_{edit}, we measure the edit quality using a CLIP text-video directional similarity metric [72, 26], defined as:

$$\begin{aligned}\Delta T &= \mathcal{E}_T(\mathrm{T}_{\text{edit}}) - \mathcal{E}_T(\mathrm{T}_{\text{source}}) \\ \Delta V &= \mathcal{E}_V(\mathrm{V}_{\text{edit}}) - \mathcal{E}_V(\mathrm{V}_{\text{source}}) \\ \mathrm{M}_{\text{dir}} &= \frac{\Delta V \cdot \Delta T}{\|\Delta V\|_2 \, \|\Delta T\|_2}\end{aligned}$$

where $\mathcal{E}_T$ is the VideoCLIP text encoder. This metric measures the consistency of the change between the two videos (in CLIP space) with the change between the two prompts.

CLIP text-video geometric similarity score. Lastly, since both metrics are important in measuring the overall editing quality [26], we consider an additional metric consisting of the geometric average of both metrics which would be penalized if one of the metrics is too low.

$$\mathrm{M}_{\text{geo}} = \sqrt{\mathrm{M}_{\text{sim}} * \mathrm{M}_{\text{dir}}}$$

We similarly compute these scores using a pre-trained image-CLIP encoder as the average of per-frame similarity scores.

Correlation between automatic and human scores. In order to measure the alignment of each evaluation metric to the human judgements, we treat the paired edit selection task as a binary classification problem, where for each pair of given video edits, we compute the ground-truth label as the majority vote among human raters. Table 7.3 shows the classification results for each of the automatic evaluation metrics using both Image and Video-based CLIP models.We use the original L/14 Image CLIP model, and a VideoCLIP model introduced in [280]. Note that random guessing would achieve roughly 50% accuracy.

Both encoder models show similar trends across all metrics, where even the highest measured overall accuracy (72%) for M_{dir} using VideoCLIP is still far off from perfectly aligning with human judgement. When split by edit type, metrics computed for spatial-type edits (style, background, object, multi-spatial) are most aligned with human raters (up to 80% accurate), whereas motion-based edits are more difficult to automatically evaluate (56% for motion-only edits). We hypothesize that this may be due to both models having only elementary understanding of motion, and being more biased towards spatial features of videos.

For further comparisons of our method and baselines, we use the two metrics with highest observed correlations, $\mathrm{M}_{\mathrm{dir}}$ and $\mathrm{M}_{\mathrm{geo}}$. Results are shown in Table 7.4, where our method outperforms all baseline methods. Table F.2 in the Supplementary shows a more detailed breakdown of evaluation results by edit type. However, in general, we note that due to the relatively low correlation between automatic metrics and human ratings, human judgements are more reliable in these evaluations.

Effect of Motion Conditioning

Lastly, we perform an ablation study on the motion conditioning introduced in our method. Table 7.5 shows a comparison between our method with and without the motion conditioning support. Both models are trained on the same data for the same amount of iterations. We only perform evaluations on a subset of edit types (Style, Object, Background, Multi-Spatial) as our method does not use motion conditioning for the other motion-based edits ($s_M = 0$). For spatial edits, we find we are able to better preserve the original motion of the entities through motion conditioning. Figure 7.5 Shows an example of when motion conditioning is beneficial in our model.

7.6 Discussion

We introduced MoCA, a method that decomposes the video editing problem into spatial and temporal components. Spatial edits are applied to the first frame of a source video, and then extrapolated using a motion-conditioned image animation model to preserve the motion of the original video. In addition, we allow motion editing by removing the motion conditioning and letting the animation model generate new frames according to the motion described in the edit prompt. We demonstrate that this simple method is a strong baseline outperforming existing methods on video editing. In addition, we introduce a new curated subset of video edits focused on motion editing, as well as a comprehensive analysis and benchmarking across a wide range of other video edits. By providing this comprehensive framework, we aim to facilitate the assessment of advancements and abilities of video editing techniques in subsequent research. We identify several limitations as directions for future work.

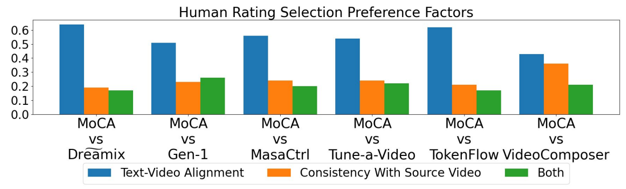


Figure 7.4: **Percentage of each reason selected when human evaluators prefer MoCA edits to each of the baselines.** The reasons for picking one model over another on each video edit could be either its better alignment with the edit prompt, higher consistency with the source video, or both. Generally, human raters preferred our method in terms of better alignment with the desired edit prompt.

Method	$\mathrm{M_{dir}}(\uparrow)$	$\mathrm{M_{geo}}(\uparrow)$
MoCA	**0.145**	**0.301**
Dreamix [177]	0.107	0.252
Gen-1 [67]	0.111	0.254
MasaCtrl [31]	0.090	0.231
Tune-a-Video [287]	0.116	0.265
TokenFlow [78]	0.098	0.235
VideoComposer [277]	0.128	0.278

Table 7.4: **Automatic scores evaluating the editing quality of each model.** We compute the VideoCILP-based $\mathrm{M_{dir}}$ and $\mathrm{M_{geo}}$ scores as the CLIP text-video directional and geometric similarity scores, respectively, averaged across all edit tasks for each baseline. Our method shows higher editing capabilities compared to the baseline methods.

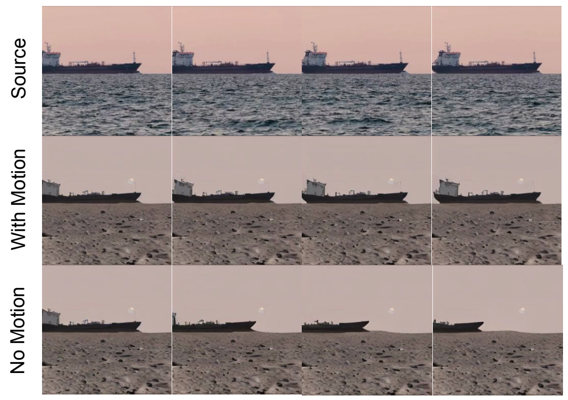


Figure 7.5: **MoCA edits for "A boat sailing on the moon" with and without motion conditioning.** Using motion conditioning allows the model to more faithfully follow the boat's movement in the original source video. Without motion conditioning, the model tends to generate more random movement directions, such as moving backwards.

	Style	**Obj**	**BG**	**M-S**	**Tot**
$s_M = 0$	57%	60%	57%	57%	58%

Table 7.5: **Ablation study on the motion conditioning in MoCA** comparing the video edits conditioned on the motion of the source video against those without any motion conditioning. Human raters show a preference to our model with motion conditioning.

- Analysis in Section 7.5 showed that all existing evaluation metrics for video editing are rather lacking in their alignment with human judgement. As such, there remains room for developing more accurate evaluation metrics for video editing, as human evaluations can be time consuming or expensive when using crowd sourced workers. In addition, a strong automatic metric may be useful for automatic selection of desired edits for hyperparameter searching or when comparing results from different random seeds.

- Due to our reliance on video extrapolation as means for video editing, our method has less fidelity when preserving any aspects of source videos that is introduced after the first frame, such as longer videos, or videos with more camera motion. Further work may involve incorporating other conditioning schemes that aim to preserve these parts of source videos, similar to our proposal of augmenting a video extrapolation model with motion conditioning to preserve motion changes.

Chapter 8

Conclusion

In this dissertation, I focused on developing techniques that better enable scaling video generation models. Importantly, this includes scaling on two axes - model size and sequence length. Scaling model sizes allows the model more complex video distributions, and scaling context lengths enables better learning important bits of information over longer videos, such as long-term 3D consistency or object permanence – all of which are crucial for eventually developing intelligent systems that can navigate with and interact the real world.

Chapters 2, 3, and 4 demonstrated a series of research contributions in efficient scaling of video models through first learning spatio-temporally compressed representations through architectural modifications in both the autoencoder and video generation models, as well as incorporating adaptivity during the learning process to further boost downstream efficiency gains.

Chapter 5 focused on extending video models to even longer context, spanning potentially hours of video, and millions of tokens. We introduced various training methodologies and training tricks necessary for stable, and efficient learning.

Chapters 6 and 7 investigated leveraging the underlying understanding of the world learned through large-scale video pre-training, and using these models for downstream tasks. We showed that such models can be used to efficiently learn reinforcement learning agents by using a video prediction model likelihood as a reward functions. In addition, we showed that such a model can also be used for effective complex spatio-temporal video editing tasks.

Although substantial progress has been made towards scaling video generation models, there is still much future work to be done. Existing video generation models are still severely underfitting its data – video models currently benefit from heavy data filtering, only keeping very high quality data for the model to focus learning on. In addition, dense text captions and the usage of classifier-free guidance techniques during inference massively improve video generation quality, suggesting that less conditional distributions are still too difficult for existing video models to learn, even for shorter video. Longer video presents an even more difficult problem due to the nature of exponentially branching futures with time. As such, there is still a large amount of potential in developing more efficient video architectures that can not only scale to more complex distributions well, as well as better scale to longer contexts – problems that could be potentially addressed with architecture design and further compressed tokenization.

Lastly, although we are currently knowledgeable in terms of scaling video generation models, it is less clear of how to exactly leverage the learned understanding of the world. I am additionally interested in exploring the applications of video generation models to algorithms that involve visual planning, such as model-based reinforcement learning algorithms. I am also interested in eventually developing omni-modal models that jointly model many modalities such as video, language, and audio, which in turn provides a clearer interface to use any acquired world knowledge.

Bibliography

[1] Pieter Abbeel and Andrew Y Ng. Apprenticeship learning via inverse reinforcement learning. In *International Conference on Machine Learning*, 2004.

[2] Sami Abu-El-Haija, Nisarg Kothari, Joonseok Lee, Paul Natsev, George Toderici, Balakrishnan Varadarajan, and Sudheendra Vijayanarasimhan. Youtube-8m: A large-scale video classification benchmark. *arXiv preprint arXiv:1609.08675*, 2016.

[3] Dinesh Acharya, Zhiwu Huang, Danda Pani Paudel, and Luc Van Gool. Towards high resolution video generation with progressive growing of sliced wasserstein gans. *arXiv preprint arXiv:1810.02419*, 2018.

[4] Emanuele Aiello, Lili Yu, Yixin Nie, Armen Aghajanyan, and Barlas Oguz. Jointly training large autoregressive multimodal models. *arXiv preprint arXiv:2309.15564*, 2023.

[5] Joshua Ainslie, James Lee-Thorp, Michiel de Jong, Yury Zemlyanskiy, Federico Lebrón, and Sumit Sanghai. Gqa: Training generalized multi-query transformer models from multi-head checkpoints. *arXiv preprint arXiv:2305.13245*, 2023.

[6] Dario Amodei and Danny Hernandez. Ai and compute. *Heruntergeladen von https://blog. openai. com/aiand-compute*, 2018.

[7] ArizeAI. Needle in a haystack - pressure testing llms. `https://github.com/Arize-ai/LLMTest_NeedleInAHaystack`, 2023.

[8] Anas Awadalla, Irena Gao, Josh Gardner, Jack Hessel, Yusuf Hanafy, Wanrong Zhu, Kalyani Marathe, Yonatan Bitton, Samir Gadre, Shiori Sagawa, et al. Openflamingo: An open-source framework for training large autoregressive vision-language models. *arXiv preprint arXiv:2308.01390*, 2023.

[9] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016.

[10] Mohammad Babaeizadeh, Chelsea Finn, Dumitru Erhan, Roy H Campbell, and Sergey Levine. Stochastic variational video prediction. *arXiv preprint arXiv:1710.11252*, 2017.

[11] Mohammad Babaeizadeh, Mohammad Taghi Saffar, Suraj Nair, Sergey Levine, Chelsea Finn, and Dumitru Erhan. Fitvid: Overfitting in pixel-level video prediction. *arXiv preprint arXiv:2106.13195*, 2021.

[12] Max Bain, Arsha Nagrani, Gül Varol, and Andrew Zisserman. Frozen in time: A joint video and image encoder for end-to-end retrieval. In *IEEE International Conference on Computer Vision*, 2021.

[13] Max Bain, Arsha Nagrani, Gül Varol, and Andrew Zisserman. Frozen in time: A joint video and image encoder for end-to-end retrieval. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 1728–1738, 2021.

[14] 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. In *Advances in Neural Information Processing Systems*, 2022.

[15] Charles Beattie, Joel Z Leibo, Denis Teplyashin, Tom Ward, Marcus Wainwright, Heinrich Küttler, Andrew Lefrancq, Simon Green, Víctor Valdés, Amir Sadik, et al. Deepmind lab. *arXiv preprint arXiv:1612.03801*, 2016.

[16] 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.

[17] Iz Beltagy, Matthew E Peters, and Arman Cohan. Longformer: The long-document transformer. *arXiv preprint arXiv:2004.05150*, 2020.

[18] Yoshua Bengio. Estimating or propagating gradients through stochastic neurons. *arXiv preprint arXiv:1305.2982*, 2013.

[19] Mikołaj Bińkowski, Jeff Donahue, Sander Dieleman, Aidan Clark, Erich Elsen, Norman Casagrande, Luis C Cobo, and Karen Simonyan. High fidelity speech synthesis with adversarial networks. *arXiv preprint arXiv:1909.11646*, 2019.

[20] 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. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 22563–22575, 2023.

[21] James Bradbury, Roy Frostig, Peter Hawkins, Matthew James Johnson, Chris Leary, Dougal Maclaurin, George Necula, Adam Paszke, Jake VanderPlas, Skye Wanderman-Milne, and Qiao Zhang. JAX: composable transformations of Python+NumPy programs, 2018. URL http://github.com/google/jax.

[22] William Brandon, Aniruddha Nrusimha, Kevin Qian, Zachary Ankner, Tian Jin, Zhiye Song, and Jonathan Ragan-Kelley. Striped attention: Faster ring attention for causal transformers. *arXiv preprint arXiv:2311.09431*, 2023.

[23] Andrew Brock, Jeff Donahue, and Karen Simonyan. Large scale gan training for high fidelity natural image synthesis. *arXiv preprint arXiv:1809.11096*, 2018.

[24] Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. Openai gym. *arXiv preprint arXiv:1606.01540*, 2016.

[25] Tim Brooks, Janne Hellsten, Miika Aittala, Ting-Chun Wang, Timo Aila, Jaakko Lehtinen, Ming-Yu Liu, Alexei A Efros, and Tero Karras. Generating long videos of dynamic scenes. *arXiv preprint arXiv:2206.03429*, 2022.

[26] Tim Brooks, Aleksander Holynski, and Alexei A Efros. Instructpix2pix: Learning to follow image editing instructions. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 18392–18402, 2023.

[27] 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. URL `https://openai.com/research/video-generation-models-as-world-simulators`.

[28] 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.

[29] Yuri Burda, Harrison Edwards, Amos Storkey, and Oleg Klimov. Exploration by random network distillation. In *International Conference on Learning Representations*, 2019.

[30] Minwoo Byeon, Beomhee Park, Haecheon Kim, Sungjun Lee, Woonhyuk Baek, and Saehoon Kim. Coyo-700m: Image-text pair dataset. `https://github.com/kakaobrain/coyo-dataset`, 2022.

[31] Mingdeng Cao, Xintao Wang, Zhongang Qi, Ying Shan, Xiaohu Qie, and Yinqiang Zheng. Masactrl: Tuning-free mutual self-attention control for consistent image synthesis and editing. *arXiv preprint arXiv:2304.08465*, 2023.

[32] Joao Carreira and Andrew Zisserman. Quo vadis, action recognition? a new model and the kinetics dataset. In *proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6299–6308, 2017.

[33] Duygu Ceylan, Chun-Hao P Huang, and Niloy J Mitra. Pix2video: Video editing using image diffusion. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 23206–23217, 2023.

[34] 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.

[35] Huiwen Chang, Han Zhang, Lu Jiang, Ce Liu, and William T Freeman. Maskgit: Masked generative image transformer. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11315–11325, 2022.

[36] Huiwen Chang, Han Zhang, Lu Jiang, Ce Liu, and William T Freeman. Maskgit: Masked generative image transformer. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11315–11325, 2022.

[37] 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.

[38] Lin Chen, Jisong Li, Xiaoyi Dong, Pan Zhang, Conghui He, Jiaqi Wang, Feng Zhao, and Dahua Lin. Sharegpt4v: Improving large multi-modal models with better captions. *arXiv preprint arXiv:2311.12793*, 2023.

[39] Mark Chen, Alec Radford, Rewon Child, Jeff Wu, Heewoo Jun, Prafulla Dhariwal, David Luan, and Ilya Sutskever. Generative pretraining from pixels. In *Proceedings of the 37th International Conference on Machine Learning*, 2020.

[40] Shouyuan Chen, Sherman Wong, Liangjian Chen, and Yuandong Tian. Extending context window of large language models via positional interpolation. *arXiv preprint arXiv:2306.15595*, 2023.

[41] Tong Chen, Haojie Liu, Qiu Shen, Tao Yue, Xun Cao, and Zhan Ma. Deepcoder: A deep neural network based video compression. In *2017 IEEE Visual Communications and Image Processing (VCIP)*, pages 1–4. IEEE, 2017.

[42] Xi Chen, Nikhil Mishra, Mostafa Rohaninejad, and Pieter Abbeel. Pixelsnail: An improved autoregressive generative model. *arXiv preprint arXiv:1712.09763*, 2017.

[43] Wei-Lin Chiang, Zhuohan Li, Zi Lin, Ying Sheng, Zhanghao Wu, Hao Zhang, Lianmin Zheng, Siyuan Zhuang, Yonghao Zhuang, Joseph E. Gonzalez, Ion Stoica, and Eric P. Xing. Vicuna: An open-source chatbot impressing gpt-4 with 90%* chatgpt quality, March 2023. URL https://lmsys.org/blog/2023-03-30-vicuna/.

[44] Rewon Child. Very deep vaes generalize autoregressive models and can outperform them on images. *arXiv preprint arXiv:2011.10650*, 2020.

[45] Rewon Child, Scott Gray, Alec Radford, and Ilya Sutskever. Generating long sequences with sparse transformers. *arXiv preprint arXiv:1904.10509*, 2019.

[46] Soumith Chintala, Emily Denton, Martin Arjovsky, and Michael Mathieu. How to train a gan? tips and tricks to make gans work. https://github.com/soumith/ganhacks, 2020.

[47] Krzysztof Choromanski, Valerii Likhosherstov, David Dohan, Xingyou Song, Andreea Gane, Tamas Sarlos, Peter Hawkins, Jared Davis, Afroz Mohiuddin, Lukasz Kaiser, et al. Rethinking attention with performers. *arXiv preprint arXiv:2009.14794*, 2020.

[48] Charilaos Christopoulos, Athanassios Skodras, and Touradj Ebrahimi. The jpeg2000 still image coding system: an overview. *IEEE transactions on consumer electronics*, 46(4): 1103–1127, 2000.

[49] Aidan Clark, Jeff Donahue, and Karen Simonyan. Adversarial video generation on complex datasets, 2019.

[50] Yilun Dai, Mengjiao Yang, Bo Dai, Hanjun Dai, Ofir Nachum, Josh Tenenbaum, Dale Schuurmans, and Pieter Abbeel. Learning universal policies via text-guided video generation. *arXiv preprint arXiv:2302.00111*, 2023.

[51] Zihang Dai, Zhilin Yang, Yiming Yang, Jaime Carbonell, Quoc V Le, and Ruslan Salakhutdinov. Transformer-xl: Attentive language models beyond a fixed-length context. *arXiv preprint arXiv:1901.02860*, 2019.

[52] Tri Dao, Dan Fu, Stefano Ermon, Atri Rudra, and Christopher Ré. Flashattention: Fast and memory-efficient exact attention with io-awareness. *Advances in Neural Information Processing Systems*, 35:16344–16359, 2022.

[53] Emily Denton and Rob Fergus. Stochastic video generation with a learned prior. *arXiv preprint arXiv:1802.07687*, 2018.

[54] Emily L Denton et al. Unsupervised learning of disentangled representations from video. In *Advances in neural information processing systems*, pages 4414–4423, 2017.

[55] Prafulla Dhariwal and Alexander Nichol. Diffusion models beat gans on image synthesis. *Advances in neural information processing systems*, 34:8780–8794, 2021.

[56] Prafulla Dhariwal, Heewoo Jun, Christine Payne, Jong Wook Kim, Alec Radford, and Ilya Sutskever. Jukebox: A generative model for music. *arXiv preprint arXiv:2005.00341*, 2020.

[57] Sander Dieleman, Charlie Nash, Jesse Engel, and Karen Simonyan. Variable-rate discrete representation learning. *arXiv preprint arXiv:2103.06089*, 2021.

[58] Ning Ding, Yulin Chen, Bokai Xu, Yujia Qin, Zhi Zheng, Shengding Hu, Zhiyuan Liu, Maosong Sun, and Bowen Zhou. Enhancing chat language models by scaling high-quality instructional conversations. *arXiv preprint arXiv:2305.14233*, 2023.

[59] Laurent Dinh, David Krueger, and Yoshua Bengio. Nice: Non-linear independent components estimation. *arXiv preprint arXiv:1410.8516*, 2014.

[60] Laurent Dinh, Jascha Sohl-Dickstein, and Samy Bengio. Density estimation using Real NVP. *arXiv preprint arXiv:1605.08803*, 2016.

[61] Andreas Doerr, Christian Daniel, Martin Schiegg, Nguyen-Tuong Duy, Stefan Schaal, Marc Toussaint, and Trimpe Sebastian. Probabilistic recurrent state-space models. In *International conference on machine learning*, pages 1280–1289. PMLR, 2018.

[62] Alexey Dosovitskiy. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.

[63] Frederik Ebert, Chelsea Finn, Alex X Lee, and Sergey Levine. Self-supervised visual planning with temporal skip connections. *arXiv preprint arXiv:1710.05268*, 2017.

[64] SM Ali Eslami, Danilo Jimenez Rezende, Frederic Besse, Fabio Viola, Ari S Morcos, Marta Garnelo, Avraham Ruderman, Andrei A Rusu, Ivo Danihelka, Karol Gregor, et al. Neural scene representation and rendering. *Science*, 360(6394):1204–1210, 2018.

[65] Patrick Esser, Robin Rombach, and Bjorn Ommer. Taming transformers for high-resolution image synthesis. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 12873–12883, 2021.

[66] Patrick Esser, Robin Rombach, and Bjorn Ommer. Taming transformers for high-resolution image synthesis. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 12873–12883, 2021.

[67] Patrick Esser, Johnathan Chiu, Parmida Atighehchian, Jonathan Granskog, and Anastasis Germanidis. Structure and content-guided video synthesis with diffusion models. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 7346–7356, 2023.

[68] Facebook. Fully Sharded Data Parallel: faster AI training with fewer GPUs — engineering.fb.com. `https://engineering.fb.com/2021/07/15/open-source/fsdp/`, 2023. [Accessed 16-May-2023].

[69] Angela Fan, Edouard Grave, and Armand Joulin. Reducing transformer depth on demand with structured dropout. *arXiv preprint arXiv:1909.11556*, 2019.

[70] Chelsea Finn, Ian Goodfellow, and Sergey Levine. Unsupervised learning for physical interaction through video prediction. In *Advances in neural information processing systems*, pages 64–72, 2016.

[71] Oran Gafni, Adam Polyak, Oron Ashual, Shelly Sheynin, Devi Parikh, and Yaniv Taigman. Make-a-scene: Scene-based text-to-image generation with human priors. In *European Conference on Computer Vision*, pages 89–106. Springer, 2022.

[72] Rinon Gal, Or Patashnik, Haggai Maron, Gal Chechik, and Daniel Cohen-Or. Stylegan-nada: Clip-guided domain adaptation of image generators. *arXiv preprint arXiv:2108.00946*, 2021.

[73] 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.

[74] Leo Gao, Stella Biderman, Sid Black, Laurence Golding, Travis Hoppe, Charles Foster, Jason Phang, Horace He, Anish Thite, Noa Nabeshima, et al. The pile: An 800gb dataset of diverse text for language modeling. *arXiv preprint arXiv:2101.00027*, 2020.

[75] Songwei Ge, Thomas Hayes, Harry Yang, Xi Yin, Guan Pang, David Jacobs, Jia-Bin Huang, and Devi Parikh. Long video generation with time-agnostic vqgan and time-sensitive transformer. *arXiv preprint arXiv:2204.03638*, 2022.

[76] Xinyang Geng and Hao Liu. Openllama: An open reproduction of llama. *URL: https://github.com/openlm-research/open_llama*, 2023.

[77] Xinyang Geng, Arnav Gudibande, Hao Liu, Eric Wallace, Pieter Abbeel, Sergey Levine, and Dawn Song. Koala: A dialogue model for academic research. *Blog post, April*, 1, 2023.

[78] Michal Geyer, Omer Bar-Tal, Shai Bagon, and Tali Dekel. Tokenflow: Consistent diffusion features for consistent video editing. *arXiv preprint arXiv:2307.10373*, 2023.

[79] Rohit Girdhar, Mannat Singh, Andrew Brown, Quentin Duval, Samaneh Azadi, Sai Saketh Rambhatla, Akbar Shah, Xi Yin, Devi Parikh, and Ishan Misra. Emu video: Factorizing text-to-video generation by explicit image conditioning. *arXiv preprint arXiv:2311.10709*, 2023.

[80] gkamradt. Needle in a haystack - pressure testing llms. `https://github.com/gkamradt/LLMTest_NeedleInAHaystack/tree/main`, 2023. [Online; accessed 7-Feb-2024].

[81] 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*, pages 2672–2680, 2014.

[82] Jean-Bastien Grill, Florian Strub, Florent Altché, Corentin Tallec, Pierre Richemond, Elena Buchatskaya, Carl Doersch, Bernardo Avila Pires, Zhaohan Guo, Mohammad Gheshlaghi Azar, et al. Bootstrap your own latent-a new approach to self-supervised learning. In *Advances in neural information processing systems*, volume 33, pages 21271–21284, 2020.

[83] Albert Gu, Karan Goel, and Christopher Ré. Efficiently modeling long sequences with structured state spaces. *arXiv preprint arXiv:2111.00396*, 2021.

[84] Zhaohan Guo, Shantanu Thakoor, Miruna Pîslar, Bernardo Avila Pires, Florent Altché, Corentin Tallec, Alaa Saade, Daniele Calandriello, Jean-Bastien Grill, Yunhao Tang, et al. Byol-explore: Exploration by bootstrapped prediction. In *Advances in neural information processing systems*, volume 35, pages 31855–31870, 2022.

[85] Agrim Gupta, Stephen Tian, Yunzhi Zhang, Jiajun Wu, Roberto Martín-Martín, and Li Fei-Fei. Maskvit: Masked visual pre-training for video prediction. *arXiv preprint arXiv:2206.11894*, 2022.

[86] Agrim Gupta, Stephen Tian, Yunzhi Zhang, Jiajun Wu, Roberto Martín-Martín, and Li Fei-Fei. Maskvit: Masked visual pre-training for video prediction. *arXiv preprint arXiv:2206.11894*, 2022.

[87] William H Guss, Brandon Houghton, Nicholay Topin, Phillip Wang, Cayden Codel, Manuela Veloso, and Ruslan Salakhutdinov. Minerl: A large-scale dataset of minecraft demonstrations. *arXiv preprint arXiv:1907.13440*, 2019.

[88] David Ha and Jürgen Schmidhuber. World models. *arXiv preprint arXiv:1803.10122*, 2018.

[89] Danijar Hafner, Timothy Lillicrap, Ian Fischer, Ruben Villegas, David Ha, Honglak Lee, and James Davidson. Learning latent dynamics for planning from pixels. In *International conference on machine learning*, pages 2555–2565. PMLR, 2019.

[90] Danijar Hafner, Pedro A Ortega, Jimmy Ba, Thomas Parr, Karl Friston, and Nicolas Heess. Action and perception as divergence minimization. *arXiv preprint arXiv:2009.01791*, 2020.

[91] Danijar Hafner, Jurgis Pasukonis, Jimmy Ba, and Timothy Lillicrap. Mastering diverse domains through world models. *arXiv preprint arXiv:2301.04104*, 2023.

[92] Jingning Han, Bohan Li, Debargha Mukherjee, Ching-Han Chiang, Adrian Grange, Cheng Chen, Hui Su, Sarah Parker, Sai Deng, Urvang Joshi, et al. A technical overview of av1. *Proceedings of the IEEE*, 109(9):1435–1462, 2021.

[93] William Harvey, Saeid Naderiparizi, Vaden Masrani, Christian Weilbach, and Frank Wood. Flexible diffusion modeling of long videos. *arXiv preprint arXiv:2205.11495*, 2022.

[94] Curtis Hawthorne, Andrew Jaegle, Cătălina Cangea, Sebastian Borgeaud, Charlie Nash, Mateusz Malinowski, Sander Dieleman, Oriol Vinyals, Matthew Botvinick, Ian Simon, et al. General-purpose, long-context autoregressive modeling with perceiver ar. *arXiv preprint arXiv:2202.07765*, 2022.

[95] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *arXiv preprint arXiv:1512.03385*, 2015.

[96] 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.

[97] Jonathan Ho and Tim Salimans. Classifier-free diffusion guidance. *arXiv preprint arXiv:2207.12598*, 2022.

[98] Jonathan Ho and Tim Salimans. Classifier-free diffusion guidance. *arXiv preprint arXiv:2207.12598*, 2022.

[99] Jonathan Ho, Xi Chen, Aravind Srinivas, Yan Duan, and Pieter Abbeel. Flow++: Improving flow-based generative models with variational dequantization and architecture design. *arXiv preprint arXiv:1902.00275*, 2019.

[100] Jonathan Ho, Nal Kalchbrenner, Dirk Weissenborn, and Tim Salimans. Axial attention in multidimensional transformers. *arXiv preprint arXiv:1912.12180*, 2019.

[101] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *arXiv preprint arXiv:2006.11239*, 2020.

[102] 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.

[103] 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.

[104] Jonathan Ho, Tim Salimans, Alexey Gritsenko, William Chan, Mohammad Norouzi, and David J Fleet. Video diffusion models. *arXiv preprint arXiv:2204.03458*, 2022.

[105] Jonathan Ho, Tim Salimans, Alexey Gritsenko, William Chan, Mohammad Norouzi, and David J Fleet. Video diffusion models. In *Advances in Neural Information Processing Systems*, 2022.

[106] 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.

[107] Tobias Höppe, Arash Mehrjou, Stefan Bauer, Didrik Nielsen, and Andrea Dittadi. Diffusion models for video prediction and infilling. *arXiv preprint arXiv:2206.07696*, 2022.

[108] Lu Hou, Zhiqi Huang, Lifeng Shang, Xin Jiang, Xiao Chen, and Qun Liu. Dynabert: Dynamic bert with adaptive width and depth. *Advances in Neural Information Processing Systems*, 33:9782–9793, 2020.

[109] Gao Huang, Yu Sun, Zhuang Liu, Daniel Sedra, and Kilian Q Weinberger. Deep networks with stochastic depth. In *Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11–14, 2016, Proceedings, Part IV 14*, pages 646–661. Springer, 2016.

[110] Yanping Huang, Youlong Cheng, Ankur Bapna, Orhan Firat, Dehao Chen, Mia Chen, HyoukJoong Lee, Jiquan Ngiam, Quoc V Le, Yonghui Wu, et al. Gpipe: Efficient training of giant neural networks using pipeline parallelism. *Advances in neural information processing systems*, 32, 2019.

[111] Zilong Huang, Xinggang Wang, Lichao Huang, Chang Huang, Yunchao Wei, and Wenyu Liu. Ccnet: Criss-cross attention for semantic segmentation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 603–612, 2019.

[112] DeLesley Hutchins, Imanol Schlag, Yuhuai Wu, Ethan Dyer, and Behnam Neyshabur. Block-recurrent transformers. *arXiv preprint arXiv:2203.07852*, 2022.

[113] Quan Huynh-Thu and Mohammed Ghanbari. Scope of validity of psnr in image/video quality assessment. *Electronics letters*, 44(13):800–801, 2008.

[114] Andrew Jaegle, Yury Sulsky, Arun Ahuja, Jake Bruce, Rob Fergus, and Greg Wayne. Imitation by predicting observations. In *International Conference on Machine Learning*, pages 4665–4676. PMLR, 2021.

[115] Stephen James, Zicong Ma, David Rovick Arrojo, and Andrew J. Davison. Rlbench: The robot learning benchmark & learning environment. *IEEE Robotics and Automation Letters*, 2020.

[116] Edwin T Jaynes. Information theory and statistical mechanics. *Physical review*, 106(4):620, 1957.

[117] Hongye Jin, Xiaotian Han, Jingfeng Yang, Zhimeng Jiang, Chia-Yuan Chang, and Xia Hu. Growlength: Accelerating llms pretraining by progressively growing training length. *arXiv preprint arXiv:2310.00576*, 2023.

[118] Yang Jin, Kun Xu, Liwei Chen, Chao Liao, Jianchao Tan, Bin Chen, Chenyi Lei, An Liu, Chengru Song, Xiaoqiang Lei, et al. Unified language-vision pretraining with dynamic discrete visual tokenization. *arXiv preprint arXiv:2309.04669*, 2023.

[119] Justin Johnson, Alexandre Alahi, and Li Fei-Fei. Perceptual losses for real-time style transfer and super-resolution. In *Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11-14, 2016, Proceedings, Part II 14*, pages 694–711. Springer, 2016.

[120] Norman P Jouppi, Cliff Young, Nishant Patil, David Patterson, Gaurav Agrawal, Raminder Bajwa, Sarah Bates, Suresh Bhatia, Nan Boden, Al Borchers, et al. In-datacenter performance analysis of a tensor processing unit. In *Proceedings of the 44th Annual International Symposium on Computer Architecture*, pages 1–12, 2017.

[121] Emmanuel Kahembwe and Subramanian Ramamoorthy. Lower dimensional kernels for video discriminators. *Neural Networks*, 132:506–520, 2020.

[122] Nal Kalchbrenner, Aaron van den Oord, Karen Simonyan, Ivo Danihelka, Oriol Vinyals, Alex Graves, and Koray Kavukcuoglu. Video pixel networks. *arXiv preprint arXiv:1610.00527*, 2016.

[123] Nal Kalchbrenner, Aäron Oord, Karen Simonyan, Ivo Danihelka, Oriol Vinyals, Alex Graves, and Koray Kavukcuoglu. Video pixel networks. In *International Conference on Machine Learning*, pages 1771–1779. PMLR, 2017.

[124] Tero Karras, Timo Aila, Samuli Laine, and Jaakko Lehtinen. Progressive growing of gans for improved quality, stability, and variation. *arXiv preprint arXiv:1710.10196*, 2017.

[125] Tero Karras, Samuli Laine, and Timo Aila. A style-based generator architecture for generative adversarial networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4401–4410, 2019.

[126] Bahjat Kawar, Shiran Zada, Oran Lang, Omer Tov, Huiwen Chang, Tali Dekel, Inbar Mosseri, and Michal Irani. Imagic: Text-based real image editing with diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6007–6017, 2023.

[127] Michał Kempka, Marek Wydmuch, Grzegorz Runc, Jakub Toczek, and Wojciech Jaśkowski. Vizdoom: A doom-based ai research platform for visual reinforcement learning. In *2016 IEEE Conference on Computational Intelligence and Games (CIG)*, pages 1–8. IEEE, 2016.

[128] Gyuwan Kim and Kyunghyun Cho. Length-adaptive transformer: Train once with length drop, use anytime with search. *arXiv preprint arXiv:2010.07003*, 2020.

[129] Diederik P Kingma and Prafulla Dhariwal. Glow: Generative flow with invertible 1x1 convolutions. *arXiv preprint arXiv:1807.03039*, 2018.

[130] Diederik P Kingma and Max Welling. Auto-encoding variational Bayes. *Proceedings of the 2nd International Conference on Learning Representations*, 2013.

[131] Diederik P. Kingma, Tim Salimans, Rafal Jozefowicz, Xi Chen, Ilya Sutskever, and Max Welling. Improving variational inference with inverse autoregressive flow. In *Advances in Neural Information Processing Systems*, 2016.

[132] Vijay Korthikanti, Jared Casper, Sangkug Lym, Lawrence McAfee, Michael Andersch, Mohammad Shoeybi, and Bryan Catanzaro. Reducing activation recomputation in large transformer models. *arXiv preprint arXiv:2205.05198*, 2022.

[133] Ilya Kostrikov, Denis Yarats, and Rob Fergus. Image augmentation is all you need: Regularizing deep reinforcement learning from pixels. In *International Conference on Learning Representations*, 2021.

[134] Manoj Kumar, Mohammad Babaeizadeh, Dumitru Erhan, Chelsea Finn, Sergey Levine, Laurent Dinh, and Durk Kingma. Videoflow: A flow-based generative model for video. *arXiv preprint arXiv:1903.01434*, 2(5):3, 2019.

[135] Aditya Kusupati, Gantavya Bhatt, Aniket Rege, Matthew Wallingford, Aditya Sinha, Vivek Ramanujan, William Howard-Snyder, Kaifeng Chen, Sham Kakade, Prateek Jain, et al. Matryoshka representation learning. *Advances in Neural Information Processing Systems*, 35:30233–30249, 2022.

[136] Brenden M Lake, Ruslan Salakhutdinov, and Joshua B Tenenbaum. Human-level concept learning through probabilistic program induction. *Science*, 350(6266):1332–1338, 2015.

[137] Didier Le Gall. Mpeg: A video compression standard for multimedia applications. *Communications of the ACM*, 34(4):46–58, 1991.

[138] Guillaume Le Moing, Jean Ponce, and Cordelia Schmid. Ccvs: Context-aware controllable video synthesis. *Advances in Neural Information Processing Systems*, 34, 2021.

[139] Alex X Lee, Richard Zhang, Frederik Ebert, Pieter Abbeel, Chelsea Finn, and Sergey Levine. Stochastic adversarial video prediction. *arXiv preprint arXiv:1804.01523*, 2018.

[140] Doyup Lee, Chiheon Kim, Saehoon Kim, Minsu Cho, and Wook-Shin Han. Draft-and-revise: Effective image generation with contextual rq-transformer. *arXiv preprint arXiv:2206.04452*, 2022.

[141] Hankook Lee and Jinwoo Shin. Anytime neural prediction via slicing networks vertically. *arXiv preprint arXiv:1807.02609*, 2018.

[142] Jinhyuk Lee, Anthony Chen, Zhuyun Dai, Dheeru Dua, Devendra Singh Sachan, Michael Boratko, Yi Luan, Sébastien MR Arnold, Vincent Perot, Siddharth Dalmia, et al. Can long-context language models subsume retrieval, rag, sql, and more? *arXiv preprint arXiv:2406.13121*, 2024.

[143] Wonkwang Lee, Whie Jung, Han Zhang, Ting Chen, Jing Yu Koh, Thomas Huang, Hyungsuk Yoon, Honglak Lee, and Seunghoon Hong. Revisiting hierarchical approach for persistent long-term video prediction. In *International Conference on Learning Representations*, 2021. URL `https://openreview.net/forum?id=3RLN4EPMdYd`.

[144] Youngwoon Lee, Andrew Szot, Shao-Hua Sun, and Joseph J. Lim. Generalizable imitation learning from observation via inferring goal proximity. In *Neural Information Processing Systems*, 2021.

[145] 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.

[146] Dacheng Li, Rulin Shao, Anze Xie, Eric P Xing, Joseph E Gonzalez, Ion Stoica, Xuezhe Ma, and Hao Zhang. Lightseq: Sequence level parallelism for distributed training of long context transformers. *arXiv preprint arXiv:2310.03294*, 2023.

[147] Jiahao Li, Bin Li, and Yan Lu. Deep contextual video compression. *Advances in Neural Information Processing Systems*, 34:18114–18125, 2021.

[148] Jiahao Li, Bin Li, and Yan Lu. Neural video compression with diverse contexts. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 22616–22626, 2023.

[149] Junnan Li, Dongxu Li, Caiming Xiong, and Steven Hoi. Blip: Bootstrapping language-image pre-training for unified vision-language understanding and generation. In *International Conference on Machine Learning*, pages 12888–12900. PMLR, 2022.

[150] Shenggui Li, Fuzhao Xue, Yongbin Li, and Yang You. Sequence parallelism: Making 4d parallelism possible. *arXiv preprint arXiv:2105.13120*, 2021.

[151] Yifan Li, Yifan Du, Kun Zhou, Jinpeng Wang, Wayne Xin Zhao, and Ji-Rong Wen. Evaluating object hallucination in large vision-language models. *arXiv preprint arXiv:2305.10355*, 2023.

[152] Yuncheng Li, Yale Song, Liangliang Cao, Joel Tetreault, Larry Goldberg, Alejandro Jaimes, and Jiebo Luo. Tgif: A new dataset and benchmark on animated gif description. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4641–4650, 2016.

[153] Bin Lin, Bin Zhu, Yang Ye, Munan Ning, Peng Jin, and Li Yuan. Video-llava: Learning united visual representation by alignment before projection. *arXiv preprint arXiv:2311.10122*, 2023.

[154] Hao Liu and Pieter Abbeel. Blockwise parallel transformer for large context models. *Advances in neural information processing systems*, 2023.

[155] Hao Liu, Wilson Yan, Matei Zaharia, and Pieter Abbeel. World model on million-length video and language with ringattention. *arXiv preprint arXiv:2402.08268*, 2024.

[156] Hao Liu, Matei Zaharia, and Pieter Abbeel. Ring attention with blockwise transformers for near-infinite context. *International Conference on Learning Representations(ICLR)*, 2024.

[157] Haotian Liu, Chunyuan Li, Yuheng Li, and Yong Jae Lee. Improved baselines with visual instruction tuning. *arXiv preprint arXiv:2310.03744*, 2023.

[158] Haotian Liu, Chunyuan Li, Qingyang Wu, and Yong Jae Lee. Visual instruction tuning. *arXiv preprint arXiv:2304.08485*, 2023.

[159] Shaoteng Liu, Yuechen Zhang, Wenbo Li, Zhe Lin, and Jiaya Jia. Video-p2p: Video editing with cross-attention control. *arXiv preprint arXiv:2303.04761*, 2023.

[160] Xiaoran Liu, Hang Yan, Shuo Zhang, Chenxin An, Xipeng Qiu, and Dahua Lin. Scaling laws of rope-based extrapolation. *arXiv preprint arXiv:2310.05209*, 2023.

[161] YuXuan Liu, Abhishek Gupta, Pieter Abbeel, and Sergey Levine. Imitation from observation: Learning to imitate behaviors from raw video via context translation. In *IEEE International Conference on Robotics and Automation*, 2018.

[162] Shayne Longpre, Le Hou, Tu Vu, Albert Webson, Hyung Won Chung, Yi Tay, Denny Zhou, Quoc V Le, Barret Zoph, Jason Wei, et al. The flan collection: Designing data and methods for effective instruction tuning. *arXiv preprint arXiv:2301.13688*, 2023.

[163] Pauline Luc, Natalia Neverova, Camille Couprie, Jakob Verbeek, and Yann LeCun. Predicting deeper into the future of semantic segmentation. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, Oct 2017.

[164] Pauline Luc, Aidan Clark, Sander Dieleman, Diego de Las Casas, Yotam Doron, Albin Cassirer, and Karen Simonyan. Transformation-based adversarial video prediction on large-scale data. *arXiv preprint arXiv:2003.04035*, 2020.

[165] Ruipu Luo, Ziwang Zhao, Min Yang, Junwei Dong, Minghui Qiu, Pengcheng Lu, Tao Wang, and Zhongyu Wei. Valley: Video assistant with large language model enhanced ability. *arXiv preprint arXiv:2306.07207*, 2023.

[166] Muhammad Maaz, Hanoona Rasheed, Salman Khan, and Fahad Shahbaz Khan. Video-chatgpt: Towards detailed video understanding via large vision and language models. *arXiv preprint arXiv:2306.05424*, 2023.

[167] Tanya Marwah, Gaurav Mittal, and Vineeth N Balasubramanian. Attentive semantic video generation using captions. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1426–1434, 2017.

[168] 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. *arXiv preprint arXiv:2108.01073*, 2021.

[169] Jacob Menick and Nal Kalchbrenner. Generating high fidelity images with subscale pixel networks and multidimensional upscaling. *arXiv preprint arXiv:1812.01608*, 2018.

[170] Fabian Mentzer, George Toderici, David Minnen, Sung-Jin Hwang, Sergi Caelles, Mario Lucic, and Eirikur Agustsson. Vct: A video compression transformer. *arXiv preprint arXiv:2206.07307*, 2022.

[171] Fabian Mentzer, David Minnen, Eirikur Agustsson, and Michael Tschannen. Finite scalar quantization: Vq-vae made simple. *arXiv preprint arXiv:2309.15505*, 2023.

[172] Fabian Mentzer, David Minnen, Eirikur Agustsson, and Michael Tschannen. Finite scalar quantization: VQ-VAE made simple. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=8ishA3LxN8.

[173] David Minnen, Johannes Ballé, and George D Toderici. Joint autoregressive and hierarchical priors for learned image compression. *Advances in neural information processing systems*, 31, 2018.

[174] Gaurav Mittal, Tanya Marwah, and Vineeth N Balasubramanian. Sync-draw: Automatic video generation using deep recurrent attentive architectures. In *Proceedings of the 25th ACM international conference on Multimedia*, pages 1096–1104, 2017.

[175] Md Mohaiminul Islam and Gedas Bertasius. Long movie clip classification with state-space video models. *arXiv e-prints*, pages arXiv–2204, 2022.

[176] Ron Mokady, Amir Hertz, Kfir Aberman, Yael Pritch, and Daniel Cohen-Or. Null-text inversion for editing real images using guided diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6038–6047, 2023.

[177] 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.

[178] Chong Mou, Xintao Wang, Liangbin Xie, Jian Zhang, Zhongang Qi, Ying Shan, and Xiaohu Qie. T2i-adapter: Learning adapters to dig out more controllable ability for text-to-image diffusion models. *arXiv preprint arXiv:2302.08453*, 2023.

[179] Debargha Mukherjee, Jingning Han, Jim Bankoski, Ronald Bultje, Adrian Grange, John Koleszar, Paul Wilkins, and Yaowu Xu. A technical overview of vp9—the latest open-source video codec. *SMPTE Motion Imaging Journal*, 124(1):44–54, 2015.

[180] Eric Nalisnick, Akihiro Matsukawa, Yee Whye Teh, Dilan Gorur, and Balaji Lakshminarayanan. Do deep generative models know what they don't know? In *International Conference on Learning Representations*, 2019.

[181] Charlie Nash, Joao Carreira, Jacob Walker, Iain Barr, Andrew Jaegle, Mateusz Malinowski, and Peter Battaglia. Transframer: Arbitrary frame prediction with generative models. *arXiv preprint arXiv:2203.09494*, 2022.

[182] Piotr Nawrot, Szymon Tworkowski, Michał Tyrolski, Łukasz Kaiser, Yuhuai Wu, Christian Szegedy, and Henryk Michalewski. Hierarchical transformers are more efficient language models. *arXiv preprint arXiv:2110.13711*, 2021.

[183] Andrew Y. Ng and Stuart J. Russell. Algorithms for inverse reinforcement learning. In *International Conference on Machine Learning*, 2000.

[184] Scott Niekum, Sarah Osentoski, George Konidaris, Sachin Chitta, Bhaskara Marthi, and Andrew G Barto. Learning grounded finite-state representations from unstructured demonstrations. *The International Journal of Robotics Research*, 34(2):131–157, 2015.

[185] Aaron van den Oord, Yazhe Li, Igor Babuschkin, Karen Simonyan, Oriol Vinyals, Koray Kavukcuoglu, George van den Driessche, Edward Lockhart, Luis C Cobo, Florian Stimberg, et al. Parallel wavenet: Fast high-fidelity speech synthesis. *arXiv preprint arXiv:1711.10433*, 2017.

[186] OpenAI. Gpt-4 technical report, 2023.

[187] Hao Ouyang, Qiuyu Wang, Yuxi Xiao, Qingyan Bai, Juntao Zhang, Kecheng Zheng, Xiaowei Zhou, Qifeng Chen, and Yujun Shen. Codef: Content deformation fields for temporally consistent video processing. *arXiv preprint arXiv:2308.07926*, 2023.

[188] Niki Parmar, Ashish Vaswani, Jakob Uszkoreit, Łukasz Kaiser, Noam Shazeer, and Alexander Ku. Image transformer. *arXiv preprint arXiv:1802.05751*, 2018.

[189] Deepak Pathak, Parsa Mahmoudieh, Michael Luo, Pulkit Agrawal, Dian Chen, Fred Shentu, Evan Shelhamer, Jitendra Malik, Alexei A. Efros, and Trevor Darrell. Zero-shot visual imitation. In *International Conference on Learning Representations*, 2018.

[190] Suraj Patil, William Berman, Robin Rombach, and Patrick von Platen. amused: An open muse reproduction. *arXiv preprint arXiv:2401.01808*, 2024.

[191] Xue Bin Peng, Ze Ma, Pieter Abbeel, Sergey Levine, and Angjoo Kanazawa. AMP: adversarial motion priors for stylized physics-based character control. *ACM Transactions on Graphics (TOG)*, 40(4):1–20, 2021.

[192] Dean A Pomerleau. Alvinn: An autonomous land vehicle in a neural network. In *Advances in Neural Information Processing Systems*, pages 305–313, 1989.

[193] Ryan Prenger, Rafael Valle, and Bryan Catanzaro. Waveglow: A flow-based generative network for speech synthesis. In *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 3617–3621. IEEE, 2019.

[194] Chenyang Qi, Xiaodong Cun, Yong Zhang, Chenyang Lei, Xintao Wang, Ying Shan, and Qifeng Chen. Fatezero: Fusing attentions for zero-shot text-based video editing. *arXiv preprint arXiv:2303.09535*, 2023.

[195] Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv preprint arXiv:1511.06434*, 2015.

[196] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners. *OpenAI Blog*, 1(8):9, 2019.

[197] 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.

[198] 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.

[199] Jack W Rae, Anna Potapenko, Siddhant M Jayakumar, and Timothy P Lillicrap. Compressive transformers for long-range sequence modelling. *arXiv preprint arXiv:1911.05507*, 2019.

[200] Ruslan Rakhimov, Denis Volkhonskiy, Alexey Artemov, Denis Zorin, and Evgeny Burnaev. Latent video transformer. *arXiv preprint arXiv:2006.10704*, 2020.

[201] Santhosh K Ramakrishnan, Aaron Gokaslan, Erik Wijmans, Oleksandr Maksymets, Alex Clegg, John Turner, Eric Undersander, Wojciech Galuba, Andrew Westbury, Angel X Chang, et al. Habitat-matterport 3d dataset (hm3d): 1000 large-scale 3d environments for embodied ai. *arXiv preprint arXiv:2109.08238*, 2021.

[202] Aditya Ramesh, Mikhail Pavlov, Gabriel Goh, Scott Gray, Chelsea Voss, Alec Radford, Mark Chen, and Ilya Sutskever. Zero-shot text-to-image generation. In *International Conference on Machine Learning*, pages 8821–8831. PMLR, 2021.

[203] Aditya Ramesh, Mikhail Pavlov, Gabriel Goh, Scott Gray, Chelsea Voss, Alec Radford, Mark Chen, and Ilya Sutskever. Zero-shot text-to-image generation. *arXiv preprint arXiv:2102.12092*, 2021.

[204] 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.

[205] Ali Razavi, Aaron van den Oord, and Oriol Vinyals. Generating diverse high-fidelity images with vq-vae-2. In *Advances in Neural Information Processing Systems*, pages 14866–14876, 2019.

[206] Machel Reid, Nikolay Savinov, Denis Teplyashin, Dmitry Lepikhin, Timothy Lillicrap, Jean-baptiste Alayrac, Radu Soricut, Angeliki Lazaridou, Orhan Firat, Julian Schrittwieser, et al. Gemini 1.5: Unlocking multimodal understanding across millions of tokens of context. *arXiv preprint arXiv:2403.05530*, 2024.

[207] Iain E Richardson. *H. 264 and MPEG-4 video compression: video coding for next-generation multimedia*. John Wiley & Sons, 2004.

[208] Martin Riedmiller, Roland Hafner, Thomas Lampe, Michael Neunert, Jonas Degrave, Tom Wiele, Vlad Mnih, Nicolas Heess, and Jost Tobias Springenberg. Learning by playing solving sparse reward tasks from scratch. In *International conference on machine learning*, pages 4344–4353. PMLR, 2018.

[209] Oren Rippel, Michael Gelbart, and Ryan Adams. Learning ordered representations with nested dropout. In *International Conference on Machine Learning*, pages 1746–1754. PMLR, 2014.

[210] 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.

[211] 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.

[212] Baptiste Rozière, Jonas Gehring, Fabian Gloeckle, Sten Sootla, Itai Gat, Xiaoqing Ellen Tan, Yossi Adi, Jingyu Liu, Tal Remez, Jérémy Rapin, et al. Code llama: Open foundation models for code. *arXiv preprint arXiv:2308.12950*, 2023.

[213] 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.

[214] 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, Tim Salimans, Jonathan Ho, David J Fleet, and Mohammad Norouzi. Photorealistic text-to-image diffusion models with deep language understanding. In *Advances in Neural Information Processing Systems*, 2022.

[215] Masaki Saito and Shunta Saito. Tganv2: Efficient training of large models for video generation with multiple subsampling layers. *arXiv preprint arXiv:1811.09245*, 2018.

[216] Masaki Saito, Eiichi Matsumoto, and Shunta Saito. Temporal generative adversarial nets with singular value clipping. In *Proceedings of the IEEE international conference on computer vision*, pages 2830–2839, 2017.

[217] Tim Salimans, Ian Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, and Xi Chen. Improved techniques for training gans. *arXiv preprint arXiv:1606.03498*, 2016.

[218] Tim Salimans, Andrej Karpathy, Xi Chen, and Diederik P Kingma. Pixelcnn++: Improving the pixelcnn with discretized logistic mixture likelihood and other modifications. *arXiv preprint arXiv:1701.05517*, 2017.

[219] Manolis Savva, Abhishek Kadian, Oleksandr Maksymets, Yili Zhao, Erik Wijmans, Bhavana Jain, Julian Straub, Jia Liu, Vladlen Koltun, Jitendra Malik, Devi Parikh, and Dhruv Batra. Habitat: A Platform for Embodied AI Research. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2019.

[220] Vaibhav Saxena, Jimmy Ba, and Danijar Hafner. Clockwork variational autoencoders. *Advances in Neural Information Processing Systems*, 34:29246–29257, 2021.

[221] Karl Schmeckpeper, Oleh Rybkin, Kostas Daniilidis, Sergey Levine, and Chelsea Finn. Reinforcement learning with videos: Combining offline observations with interaction. In *Conference on Robot Learning*, 2021.

[222] 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. *Advances in Neural Information Processing Systems*, 35:25278–25294, 2022.

[223] Ramanan Sekar, Oleh Rybkin, Kostas Daniilidis, Pieter Abbeel, Danijar Hafner, and Deepak Pathak. Planning to explore via self-supervised world models. In *International Conference on Machine Learning*, 2020.

[224] Younggyo Seo, Kimin Lee, Fangchen Liu, Stephen James, and Pieter Abbeel. Harp: Autoregressive latent video prediction with high-fidelity image generator. *International Conference on Image Processing*, 2022.

[225] Pierre Sermanet, Kelvin Xu, and Sergey Levine. Unsupervised perceptual rewards for imitation learning. *Robotics: Science and Systems*, 2017.

[226] Pierre Sermanet, Corey Lynch, Yevgen Chebotar, Jasmine Hsu, Eric Jang, Stefan Schaal, and Sergey Levine. Time-contrastive networks: Self-supervised learning from video. In *IEEE International Conference on Robotics and Automation*, pages 1134–1141, 2018.

[227] Claude Elwood Shannon. A mathematical theory of communication. *ACM SIGMOBILE mobile computing and communications review*, 5(1):3–55, 2001.

[228] Mohammad Shoeybi, Mostofa Patwary, Raul Puri, Patrick LeGresley, Jared Casper, and Bryan Catanzaro. Megatron-lm: Training multi-billion parameter language models using model parallelism. *arXiv preprint arXiv:1909.08053*, 2019.

[229] 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.

[230] Uriel Singer, Adam Polyak, Thomas Hayes, Xi Yin, Jie An, Songyang Zhang, Qiyuan Hu, Harry Yang, Oron Ashual, Oran Gafni, Devi Parikh, Sonal Gupta, and Yaniv Taigman. Make-a-video: Text-to-video generation without text-video data. In *International Conference on Learning Representations*, 2023.

[231] Ivan Skorokhodov, Sergey Tulyakov, and Mohamed Elhoseiny. Stylegan-v: A continuous video generator with the price, image quality and perks of stylegan2. *arXiv preprint arXiv:2112.14683*, 2021.

[232] Jascha Sohl-Dickstein, Eric Weiss, Niru Maheswaranathan, and Surya Ganguli. Modeling arbitrary probability distributions using nonequilibrium thermodynamics. *Unpublished Manuscript*, 2014.

[233] Jascha Sohl-Dickstein, Eric A Weiss, Niru Maheswaranathan, and Surya Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics. *arXiv preprint arXiv:1503.03585*, 2015.

[234] Casper Kaae S nderby, Lasse Espeholt, Jonathan Heek, Mostafa Dehghani, Avital Oliver, Tim Salimans, Shreya Agrawal, Jason Hickey, and Nal Kalchbrenner. Metnet: A neural weather model for precipitation forecasting. *arXiv preprint arXiv:2003.12140*, 2020.

[235] Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising diffusion implicit models. *arXiv preprint arXiv:2010.02502*, 2020.

[236] Yang Song and Stefano Ermon. Generative modeling by estimating gradients of the data distribution. In *Advances in Neural Information Processing Systems*, pages 11918–11930, 2019.

[237] Khurram Soomro, Amir Roshan Zamir, and Mubarak Shah. Ucf101: A dataset of 101 human actions classes from videos in the wild. *arXiv preprint arXiv:1212.0402*, 2012.

[238] Charles Spearman. The proof and measurement of association between two things. 1961.

[239] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958, 2014.

[240] Nitish Srivastava, Elman Mansimov, and Ruslan Salakhudinov. Unsupervised learning of video representations using lstms. In *International conference on machine learning*, pages 843–852, 2015.

[241] Jianlin Su, Murtadha Ahmed, Yu Lu, Shengfeng Pan, Wen Bo, and Yunfeng Liu. Roformer: Enhanced transformer with rotary position embedding. *Neurocomputing*, 568:127063, 2024.

[242] Vivienne Sze, Madhukar Budagavi, and Gary J Sullivan. High efficiency video coding (hevc). In *Integrated circuit and systems, algorithms and architectures*, volume 39, page 40. Springer, 2014.

[243] Andrew Szot, Alex Clegg, Eric Undersander, Erik Wijmans, Yili Zhao, John Turner, Noah Maestre, Mustafa Mukadam, Devendra Chaplot, Oleksandr Maksymets, Aaron Gokaslan, Vladimir Vondrus, Sameer Dharur, Franziska Meier, Wojciech Galuba, Angel Chang, Zsolt Kira, Vladlen Koltun, Jitendra Malik, Manolis Savva, and Dhruv Batra. Habitat 2.0: Training home assistants to rearrange their habitat. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2021.

[244] Rohan Taori, Ishaan Gulrajani, Tianyi Zhang, Yann Dubois, Xuechen Li, Carlos Guestrin, Percy Liang, and Tatsunori B Hashimoto. Alpaca: A strong, replicable instruction-following model. *Stanford Center for Research on Foundation Models. https://crfm. stanford. edu/2023/03/13/alpaca. html*, 3(6):7, 2023.

[245] Gemini Team, Rohan Anil, Sebastian Borgeaud, Yonghui Wu, Jean-Baptiste Alayrac, Jiahui Yu, Radu Soricut, Johan Schalkwyk, Andrew M Dai, Anja Hauth, et al. Gemini: a family of highly capable multimodal models. *arXiv preprint arXiv:2312.11805*, 2023.

[246] Zachary Teed and Jia Deng. Raft: Recurrent all-pairs field transforms for optical flow. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part II 16*, pages 402–419. Springer, 2020.

[247] Lucas Theis, Wenzhe Shi, Andrew Cunningham, and Ferenc Huszár. Lossy image compression with compressive autoencoders. In *International conference on learning representations*, 2022.

[248] Keyu Tian, Yi Jiang, Zehuan Yuan, Bingyue Peng, and Liwei Wang. Visual autoregressive modeling: Scalable image generation via next-scale prediction. *arXiv preprint arXiv:2404.02905*, 2024.

[249] Yu Tian, Jian Ren, Menglei Chai, Kyle Olszewski, Xi Peng, Dimitris N Metaxas, and Sergey Tulyakov. A good image generator is what you need for high-resolution video synthesis. *arXiv preprint arXiv:2104.15069*, 2021.

[250] Emanuel Todorov. Efficient computation of optimal actions. *Proceedings of the national academy of sciences*, 106(28):11478–11483, 2009.

[251] Faraz Torabi, Garrett Warnell, and Peter Stone. Behavioral cloning from observation. In *International Joint Conferences on Artificial Intelligence*, 2018.

[252] Faraz Torabi, Garrett Warnell, and Peter Stone. Generative adversarial imitation from observation. *arXiv preprint arXiv:1807.06158*, 2018.

[253] Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*, 2023.

[254] Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*, 2023.

[255] Du Tran, Lubomir Bourdev, Rob Fergus, Lorenzo Torresani, and Manohar Paluri. Learning spatiotemporal features with 3d convolutional networks. In *Proceedings of the IEEE international conference on computer vision*, pages 4489–4497, 2015.

[256] Linoy Tsaban and Apolinário Passos. Ledits: Real image editing with ddpm inversion and semantic guidance. *arXiv preprint arXiv:2307.00522*, 2023.

[257] Sergey Tulyakov, Ming-Yu Liu, Xiaodong Yang, and Jan Kautz. Mocogan: Decomposing motion and content for video generation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1526–1535, 2018.

[258] Narek Tumanyan, Michal Geyer, Shai Bagon, and Tali Dekel. Plug-and-play diffusion features for text-driven image-to-image translation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1921–1930, 2023.

[259] Saran Tunyasuvunakool, Alistair Muldal, Yotam Doron, Siqi Liu, Steven Bohez, Josh Merel, Tom Erez, Timothy Lillicrap, Nicolas Heess, and Yuval Tassa. dm_control: Software and tasks for continuous control. *Software Impacts*, 6:100022, 2020. ISSN 2665-9638.

[260] Szymon Tworkowski, Konrad Staniszewski, Mikołaj Pacek, Yuhuai Wu, Henryk Michalewski, and Piotr Miłoś. Focused transformer: Contrastive training for context scaling. *arXiv preprint arXiv:2307.03170*, 2023.

[261] Thomas Unterthiner, Sjoerd van Steenkiste, Karol Kurach, Raphaël Marinier, Marcin Michalski, and Sylvain Gelly. Fvd: A new metric for video generation. 2019.

[262] Arash Vahdat and Jan Kautz. Nvae: A deep hierarchical variational autoencoder. *arXiv preprint arXiv:2007.03898*, 2020.

[263] Aaron van den Oord, Sander Dieleman, Heiga Zen, Karen Simonyan, Oriol Vinyals, Alex Graves, Nal Kalchbrenner, Andrew Senior, and Koray Kavukcuoglu. Wavenet: A generative model for raw audio. *arXiv preprint arXiv:1609.03499*, 2016.

[264] Aaron van den Oord, Nal Kalchbrenner, and Koray Kavukcuoglu. Pixel recurrent neural networks. *International Conference on Machine Learning (ICML)*, 2016.

[265] Aaron van den Oord, Nal Kalchbrenner, Oriol Vinyals, Lasse Espeholt, Alex Graves, and Koray Kavukcuoglu. Conditional image generation with pixelcnn decoders. *arXiv preprint arXiv:1606.05328*, 2016.

[266] Aaron Van Den Oord, Oriol Vinyals, et al. Neural discrete representation learning. In *Advances in Neural Information Processing Systems*, pages 6306–6315, 2017.

[267] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. *arXiv preprint arXiv:1706.03762*, 2017.

[268] Ruben Villegas, Arkanath Pathak, Harini Kannan, Dumitru Erhan, Quoc V Le, and Honglak Lee. High fidelity video prediction with large stochastic recurrent neural networks. *Advances in Neural Information Processing Systems*, 32:81–91, 2019.

[269] 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.

[270] 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.

[271] Vikram Voleti, Alexia Jolicoeur-Martineau, and Christopher Pal. Masked conditional video diffusion for prediction, generation, and interpolation. *arXiv preprint arXiv:2205.09853*, 2022.

[272] Carl Vondrick, Hamed Pirsiavash, and Antonio Torralba. Generating videos with scene dynamics. In *Advances in neural information processing systems*, pages 613–621, 2016.

[273] Jacob Walker, Ali Razavi, and Aäron van den Oord. Predicting video with vqvae. *arXiv preprint arXiv:2103.01950*, 2021.

[274] Gregory K Wallace. The jpeg still picture compression standard. *IEEE transactions on consumer electronics*, 38(1):xviii–xxxiv, 1992.

[275] Sinong Wang, Belinda Z Li, Madian Khabsa, Han Fang, and Hao Ma. Linformer: Self-attention with linear complexity. *arXiv preprint arXiv:2006.04768*, 2020.

[276] Ting-Chun Wang, Ming-Yu Liu, Jun-Yan Zhu, Guilin Liu, Andrew Tao, Jan Kautz, and Bryan Catanzaro. Video-to-video synthesis. *arXiv preprint arXiv:1808.06601*, 2018.

[277] Xiang Wang, Hangjie Yuan, Shiwei Zhang, Dayou Chen, Jiuniu Wang, Yingya Zhang, Yujun Shen, Deli Zhao, and Jingren Zhou. Videocomposer: Compositional video synthesis with motion controllability. *arXiv preprint arXiv:2306.02018*, 2023.

[278] Xiaolong Wang, Ross Girshick, Abhinav Gupta, and Kaiming He. Non-local neural networks. *arXiv preprint arXiv:1711.07971*, 2017.

[279] Xinlong Wang, Xiaosong Zhang, Zhengxiong Luo, Quan Sun, Yufeng Cui, Jinsheng Wang, Fan Zhang, Yueze Wang, Zhen Li, Qiying Yu, et al. Emu3: Next-token prediction is all you need. *arXiv preprint arXiv:2409.18869*, 2024.

[280] Yi Wang, Kunchang Li, Yizhuo Li, Yinan He, Bingkun Huang, Zhiyu Zhao, Hongjie Zhang, Jilan Xu, Yi Liu, Zun Wang, et al. Internvideo: General video foundation models via generative and discriminative learning. *arXiv preprint arXiv:2212.03191*, 2022.

[281] Yi Wang, Yinan He, Yizhuo Li, Kunchang Li, Jiashuo Yu, Xin Ma, Xinhao Li, Guo Chen, Xinyuan Chen, Yaohui Wang, et al. Internvid: A large-scale video-text dataset for multimodal understanding and generation. *arXiv preprint arXiv:2307.06942*, 2023.

[282] Zhou Wang, Alan C Bovik, Hamid R Sheikh, and Eero P Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE transactions on image processing*, 13(4):600–612, 2004.

[283] Jason Wei, Maarten Bosma, Vincent Zhao, Kelvin Guu, Adams Wei Yu, Brian Lester, Nan Du, Andrew M. Dai, and Quoc V Le. Finetuned language models are zero-shot learners. In *International Conference on Learning Representations*, 2022.

[284] Dirk Weissenborn, Oscar Täckström, and Jakob Uszkoreit. Scaling autoregressive video models. *arXiv preprint arXiv:1906.02634*, 2019.

[285] Thomas Wiegand, Gary J Sullivan, Gisle Bjontegaard, and Ajay Luthra. Overview of the h. 264/avc video coding standard. *IEEE Transactions on circuits and systems for video technology*, 13(7):560–576, 2003.

[286] Chao-Yuan Wu, Yanghao Li, Karttikeya Mangalam, Haoqi Fan, Bo Xiong, Jitendra Malik, and Christoph Feichtenhofer. Memvit: Memory-augmented multiscale vision transformer for efficient long-term video recognition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 13587–13597, 2022.

[287] Jay Zhangjie Wu, Yixiao Ge, Xintao Wang, Stan Weixian Lei, Yuchao Gu, Yufei Shi, 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. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 7623–7633, 2023.

[288] Jay Zhangjie Wu, Xiuyu Li, Difei Gao, Zhen Dong, Jinbin Bai, Aishani Singh, Xiaoyu Xiang, Youzeng Li, Zuwei Huang, Yuanxi Sun, Rui He, Feng Hu, Junhua Hu, Hai Huang, Hanyu Zhu, Xu Cheng, Jie Tang, Mike Zheng Shou, Kurt Keutzer, and Forrest Iandola. Cvpr 2023 text guided video editing competition, 2023.

[289] Fei Xia, Amir R Zamir, Zhiyang He, Alexander Sax, Jitendra Malik, and Silvio Savarese. Gibson env: Real-world perception for embodied agents. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 9068–9079, 2018.

[290] Jinheng Xie, Weijia Mao, Zechen Bai, David Junhao Zhang, Weihao Wang, Kevin Qinghong Lin, Yuchao Gu, Zhijie Chen, Zhenheng Yang, and Mike Zheng Shou. Show-o: One single transformer to unify multimodal understanding and generation. *arXiv preprint arXiv:2408.12528*, 2024.

[291] Dejing Xu, Zhou Zhao, Jun Xiao, Fei Wu, Hanwang Zhang, Xiangnan He, and Yueting Zhuang. Video question answering via gradually refined attention over appearance and motion. In *Proceedings of the 25th ACM international conference on Multimedia*, pages 1645–1653, 2017.

[292] Wilson Yan, Yunzhi Zhang, Pieter Abbeel, and Aravind Srinivas. Videogpt: Video generation using vq-vae and transformers. *arXiv preprint arXiv:2104.10157*, 2021.

[293] Wilson Yan, Yunzhi Zhang, Pieter Abbeel, and Aravind Srinivas. Videogpt: Video generation using vq-vae and transformers. *arXiv preprint arXiv:2104.10157*, 2021.

[294] Shuai Yang, Yifan Zhou, Ziwei Liu, and Chen Change Loy. Rerender a video: Zero-shot text-guided video-to-video translation. *arXiv preprint arXiv:2306.07954*, 2023.

[295] Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William W Cohen, Ruslan Salakhutdinov, and Christopher D Manning. Hotpotqa: A dataset for diverse, explainable multi-hop question answering. *arXiv preprint arXiv:1809.09600*, 2018.

[296] Jiahui Yu, Linjie Yang, Ning Xu, Jianchao Yang, and Thomas Huang. Slimmable neural networks. *arXiv preprint arXiv:1812.08928*, 2018.

[297] Jiahui Yu, Xin Li, Jing Yu Koh, Han Zhang, Ruoming Pang, James Qin, Alexander Ku, Yuanzhong Xu, Jason Baldridge, and Yonghui Wu. Vector-quantized image modeling with improved vqgan. *arXiv preprint arXiv:2110.04627*, 2021.

[298] 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*, 2(3):5, 2022.

[299] Lijun Yu, Yong Cheng, Kihyuk Sohn, José Lezama, Han Zhang, Huiwen Chang, Alexander G Hauptmann, Ming-Hsuan Yang, Yuan Hao, Irfan Essa, et al. Magvit: Masked generative video transformer. *arXiv preprint arXiv:2212.05199*, 2022.

[300] Lijun Yu, Yong Cheng, Kihyuk Sohn, José Lezama, Han Zhang, Huiwen Chang, Alexander G Hauptmann, Ming-Hsuan Yang, Yuan Hao, Irfan Essa, et al. Magvit: Masked generative video transformer. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10459–10469, 2023.

[301] Lijun Yu, José Lezama, Nitesh B Gundavarapu, Luca Versari, Kihyuk Sohn, David Minnen, Yong Cheng, Agrim Gupta, Xiuye Gu, Alexander G Hauptmann, et al. Language model beats diffusion–tokenizer is key to visual generation. *arXiv preprint arXiv:2310.05737*, 2023.

[302] Sihyun Yu, Jihoon Tack, Sangwoo Mo, Hyunsu Kim, Junho Kim, Jung-Woo Ha, and Jinwoo Shin. Generating videos with dynamics-aware implicit generative adversarial networks. *arXiv preprint arXiv:2202.10571*, 2022.

[303] Vladyslav Yushchenko, Nikita Araslanov, and Stefan Roth. Markov decision process for video generation. In *Proceedings of the IEEE International Conference on Computer Vision Workshops*, pages 0–0, 2019.

[304] Yan Zeng, Guoqiang Wei, Jiani Zheng, Jiaxin Zou, Yang Wei, Yuchen Zhang, and Hang Li. Make pixels dance: High-dynamic video generation. *arXiv preprint arXiv:2311.10982*, 2023.

[305] Shuangfei Zhai, Walter Talbott, Nitish Srivastava, Chen Huang, Hanlin Goh, Ruixiang Zhang, and Josh Susskind. An attention free transformer. *arXiv preprint arXiv:2105.14103*, 2021.

[306] Han Zhang, Ian Goodfellow, Dimitris Metaxas, and Augustus Odena. Self-attention generative adversarial networks. In *International Conference on Machine Learning*, pages 7354–7363. PMLR, 2019.

[307] Lvmin Zhang, Anyi Rao, and Maneesh Agrawala. Adding conditional control to text-to-image diffusion models. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 3836–3847, 2023.

[308] Renrui Zhang, Jiaming Han, Aojun Zhou, Xiangfei Hu, Shilin Yan, Pan Lu, Hongsheng Li, Peng Gao, and Yu Qiao. Llama-adapter: Efficient fine-tuning of language models with zero-init attention. *arXiv preprint arXiv:2303.16199*, 2023.

[309] Richard Zhang, Phillip Isola, Alexei A Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 586–595, 2018.

[310] Yabo Zhang, Yuxiang Wei, Dongsheng Jiang, Xiaopeng Zhang, Wangmeng Zuo, and Qi Tian. Controlvideo: Training-free controllable text-to-video generation. *arXiv preprint arXiv:2305.13077*, 2023.

[311] Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric Xing, et al. Judging llm-as-a-judge with mt-bench and chatbot arena. *arXiv preprint arXiv:2306.05685*, 2023.

[312] Deyao Zhu, Jun Chen, Xiaoqian Shen, Xiang Li, and Mohamed Elhoseiny. Minigpt-4: Enhancing vision-language understanding with advanced large language models. *arXiv preprint arXiv:2304.10592*, 2023.

[313] Brian D Ziebart, Andrew L Maas, J Andrew Bagnell, and Anind K Dey. Maximum entropy inverse reinforcement learning. In *Association for the Advancement of Artificial Intelligence*, 2008.

[314] Konrad Zolna, Scott Reed, Alexander Novikov, Sergio Gomez Colmenarej, David Budden, Serkan Cabi, Misha Denil, Nando de Freitas, and Ziyu Wang. Task-relevant adversarial imitation learning. In *Conference on Robot Learning*, 2020.

Appendix A

Appendix For Chapter 2

A.1 Architecture Details and Hyperparameters

VQ-VAE Encoder and Decoder

Table A.1: Hyperparameters of VQ-VAE encoder and decoder models for each dataset

	MMNIST	BAIR / RoboNet / ViZDoom	UCF-101 / TGIF
Input size	$16 \times 64 \times 64$	$16 \times 64 \times 64$	$16 \times 64 \times 64$
Latent size	$4 \times 16 \times 16$	$8 \times 32 \times 32$	$4 \times 32 \times 32$
β	0.25	0.25	0.25
Batch size	32	32	32
Learning rate	7×10^{-4}	7×10^{-4}	7×10^{-4}
Hidden units	240	240	240
Residual units	128	128	128
Residual layers	2	4	4
Uses attention	No	Yes	Yes
Codebook size	512	1024	1024
Codebook dim	64	256	256
Encoder filter	3	3	3
Upsample filter	4	4	4
Training steps	20k	100K	100K

Prior Networks

Table A.2: Hyperparameters of prior networks for each dataset

	MMNIST	BAIR / RoboNet	ViZDoom	UCF-101 / TGIF
Input size	$4 \times 16 \times 16$	$8 \times 32 \times 32$	$8 \times 32 \times 32$	$4 \times 32 \times 32$
Cond sizes	$1 \times 64 \times 64$	$3 \times 64 \times 64$, 64	$60/315$	n/a
Batch size	32	32	32	32
Learning rate	3×10^{-4}	3×10^{-4}	3×10^{-4}	3×10^{-4}
Vocabulary size	512	1024	1024	1024
Attention heads	4	4	4	8
Attention layers	8	16	16	20
Embedding size	192	512	512	1024
FFN dim	384	2048	2048	4096
Resnet depth	18	34	n/a	n/a
Resnet units	512	512	n/a	n/a
Dropout	0.1	0.2	0.2	0.2
Training steps	80k	150K	150K	200K / 600K

A.2 ViZDoom Samples

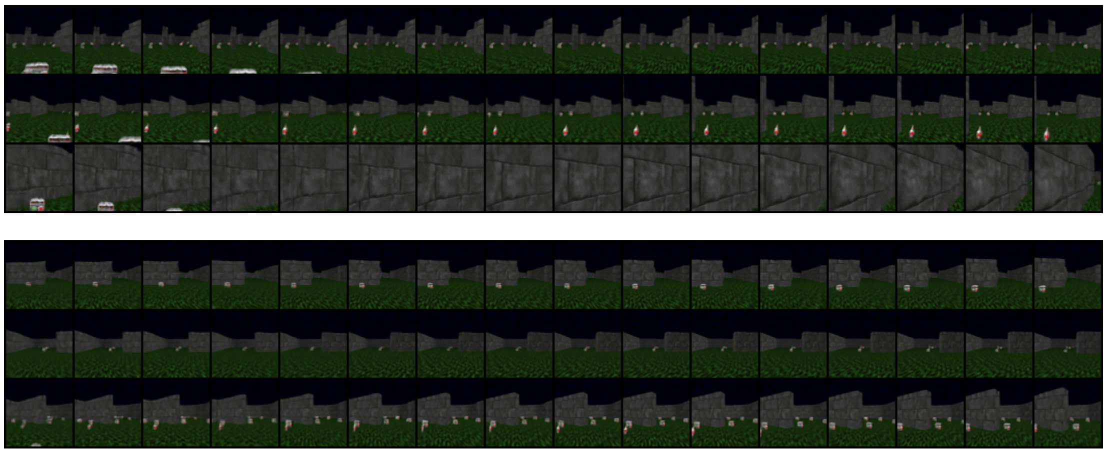


Figure A.1: Samples for ViZDoom health gathering supreme environment. (Top) shows unconditionally generated samples. (Bottom) shows samples conditioned on the same action sequence (turn right and go straight).

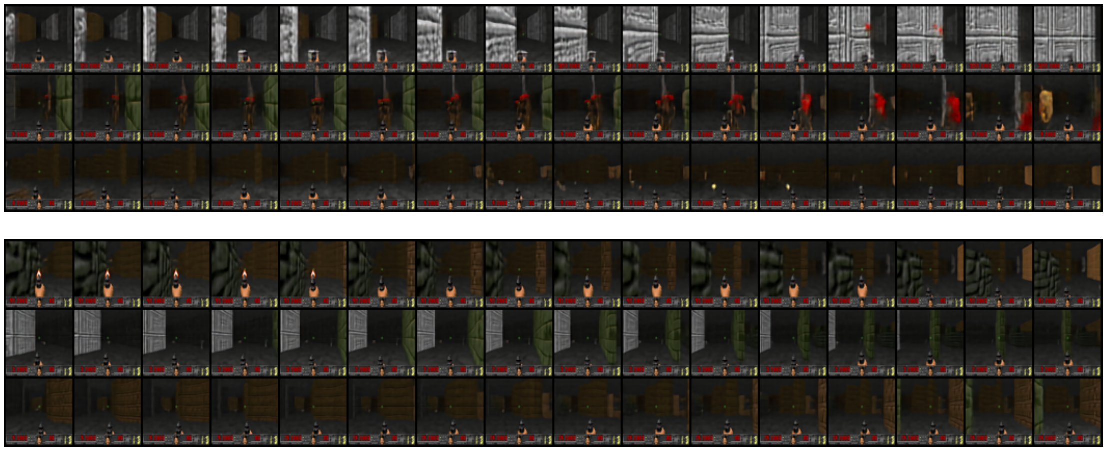


Figure A.2: Samples for ViZDoom battle2 environment. (Top) shows unconditionally generated samples. (Bottom) shows three samples conditioned on the same action sequence (moving forward and right).

Appendix B

Appendix For Chapter 3

B.1 Sampling Process

Given a sequence of conditioning frames, $o_1, \ldots, o_t$, we encode each frame using the pretrained VQ-GAN to produce $x_1, \ldots, x_t$, and then use the conditional encoder to compute $z_1, \ldots, z_t$. In order to generate the next frame, we use the temporal transformer to compute h_t, and feed it into the MaskGit dynamics prior to predict $\hat{z}_{t+1}$. Let $z_{t+1} = \hat{z}_{t+1}$ and feed it through the temporal transformer and MaskGit to predict $\hat{z}_{t+2}$. We repeat this process until the entire trajectory is predicted, $\hat{z}_{t+1}, \ldots, \hat{z}_T$. In order to decode back into frames, we first decode into the VQ-GAN latents, and then decode to RGB using the VQ-GAN decoder. Note that generation can be completely done in latent space, and rendering back to RGB can be done in parallel over time once the latents for all timesteps are computed.

B.2 Samples

DMLab

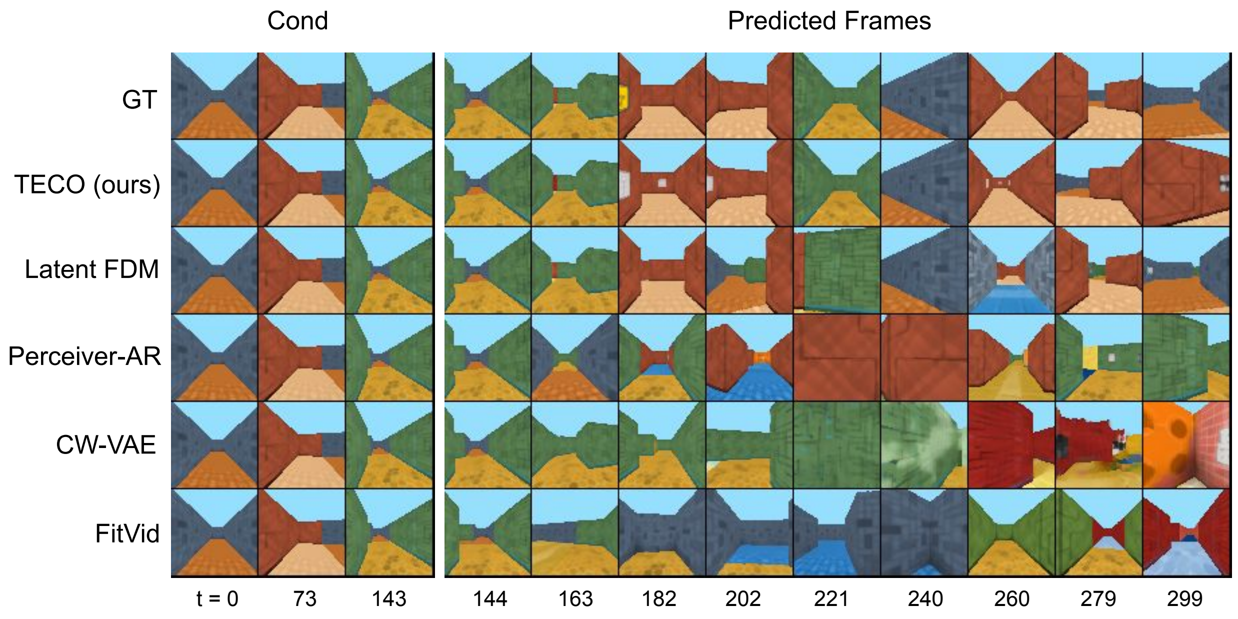


Figure B.1: 156 frames generated conditioned on 144 (action-conditioned)

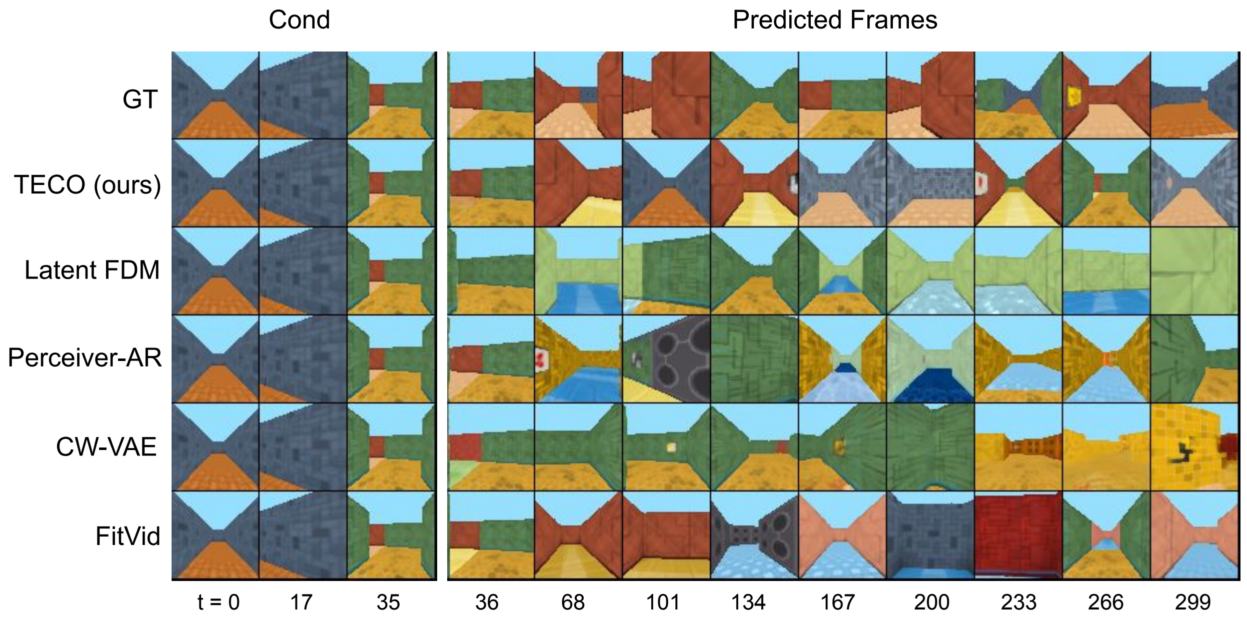


Figure B.2: 264 frames generated conditioned on 36 (**no** action-conditioning)

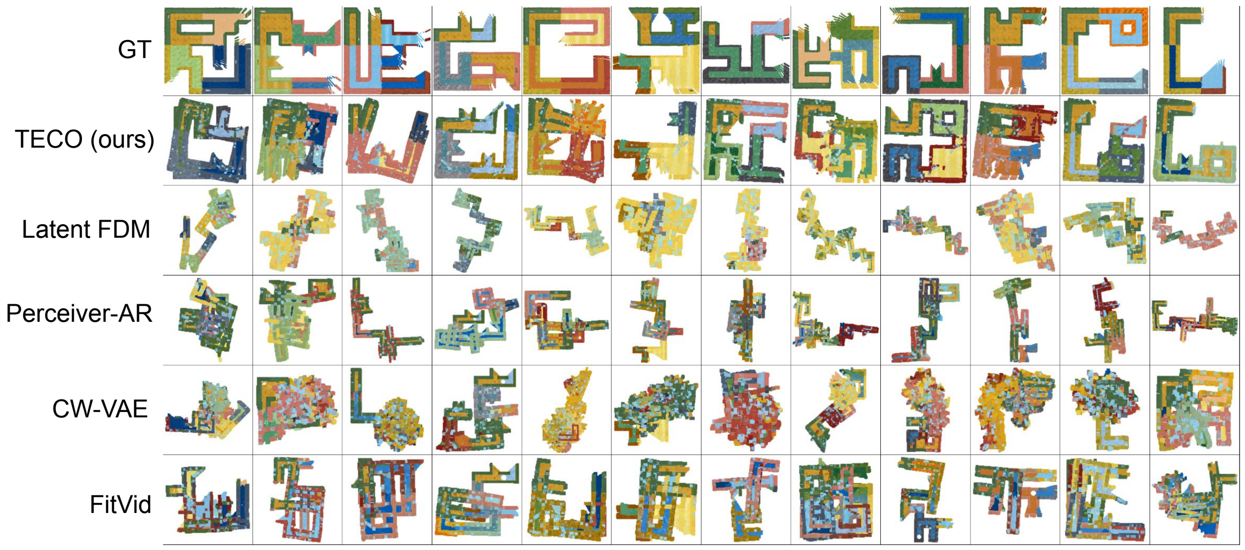


Figure B.3: 3D visualizations of the resulting generated DMLab mazes

Minecraft

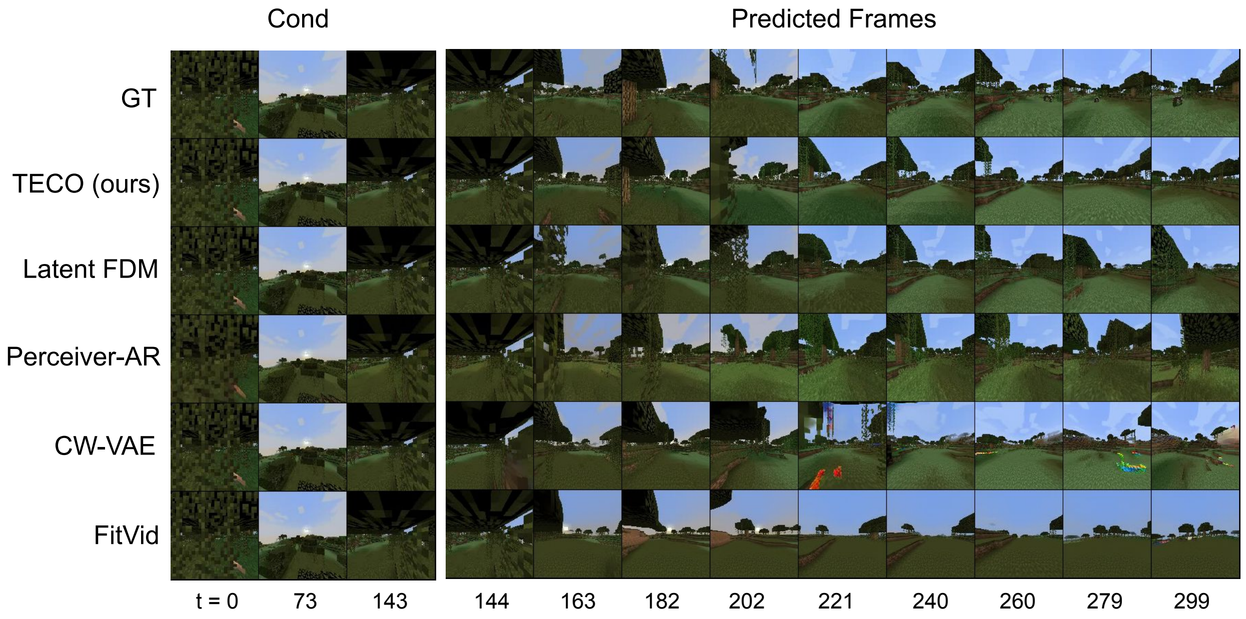


Figure B.4: 156 frames generated conditioned on 144 (action-conditioned)

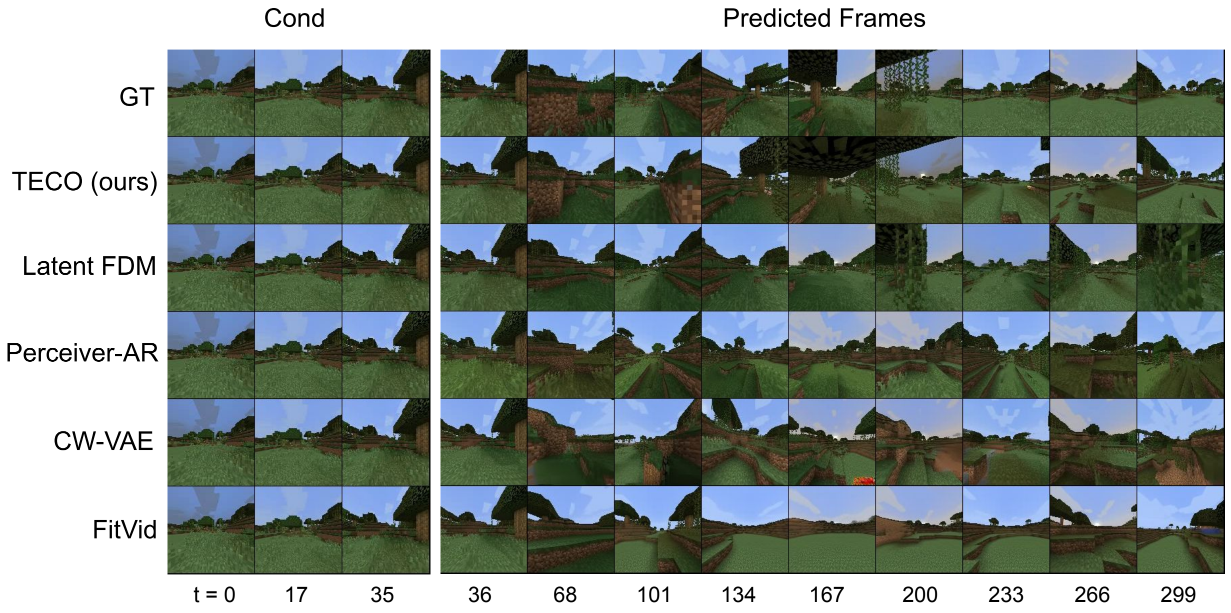


Figure B.5: 264 frames generated conditioned on 36 (action-conditioned)

Habitat

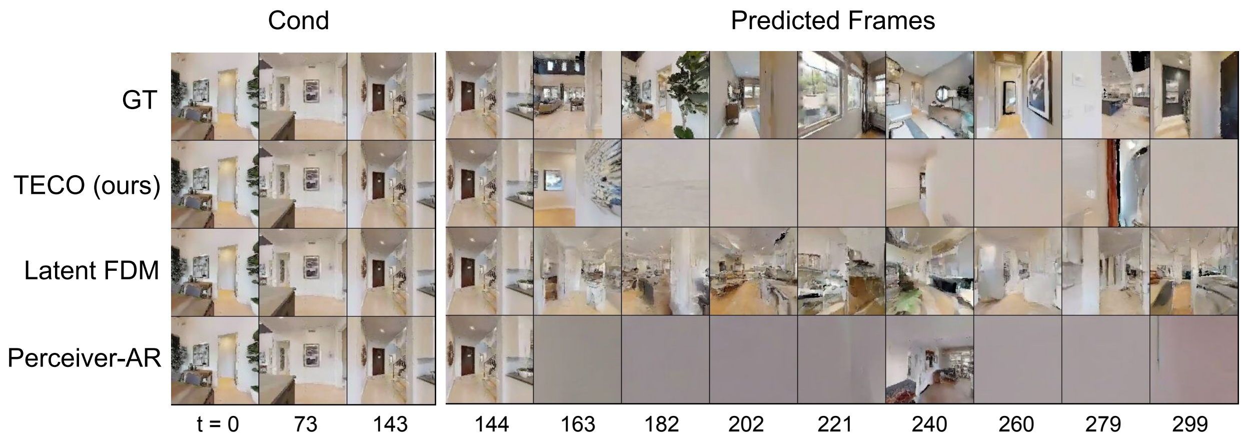


Figure B.6: 156 frames generated conditioned on 144 (action-conditioned)

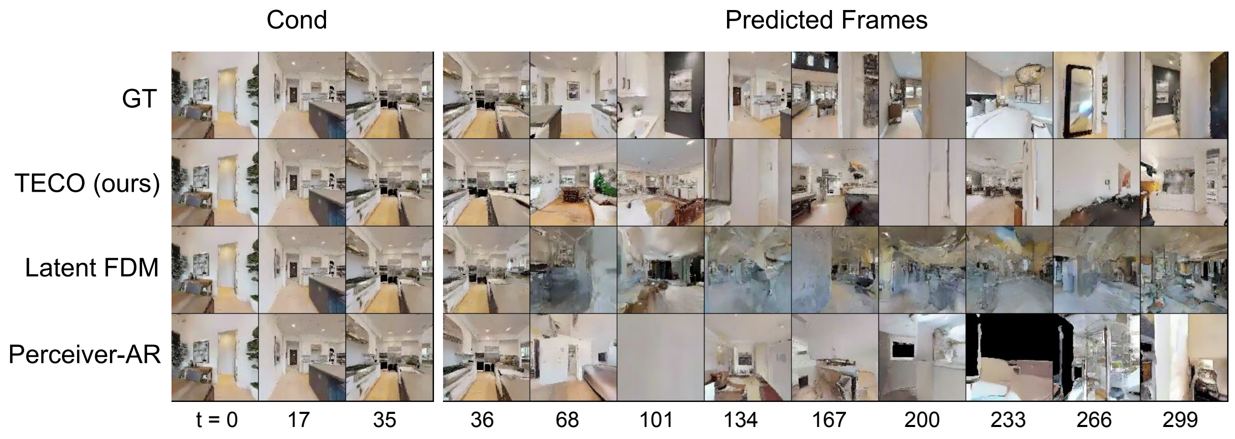


Figure B.7: 264 frames generated conditioned on 36 (**no** action-conditioning)

Kinetics-600

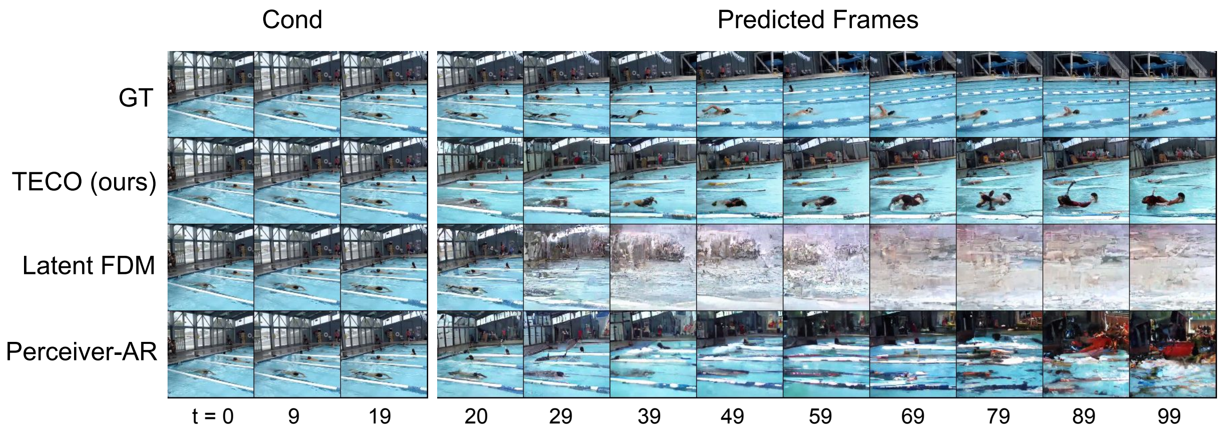


Figure B.8: 80 frames generated conditioned on 20 (no top-k sampling)

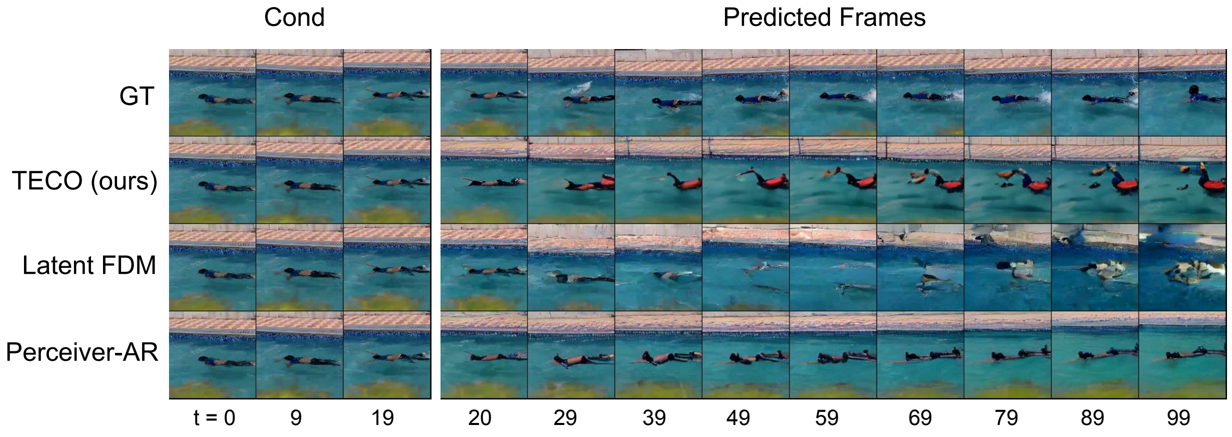
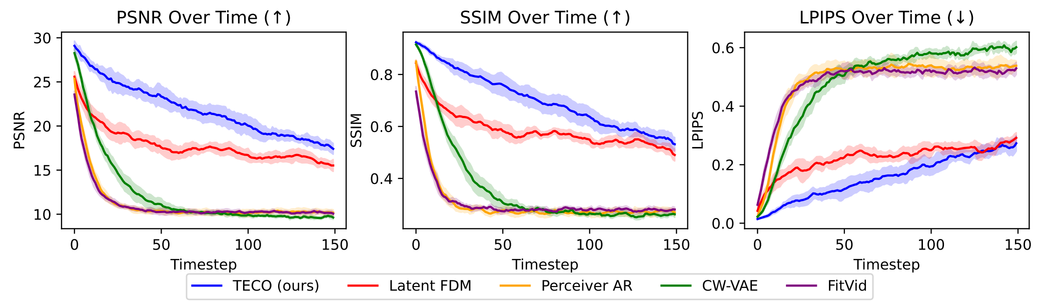
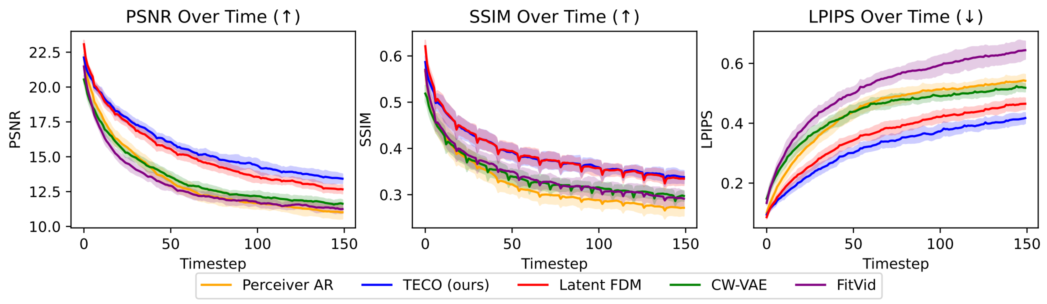


Figure B.9: 80 frames generated conditioned on 20 (with top-k sampling)

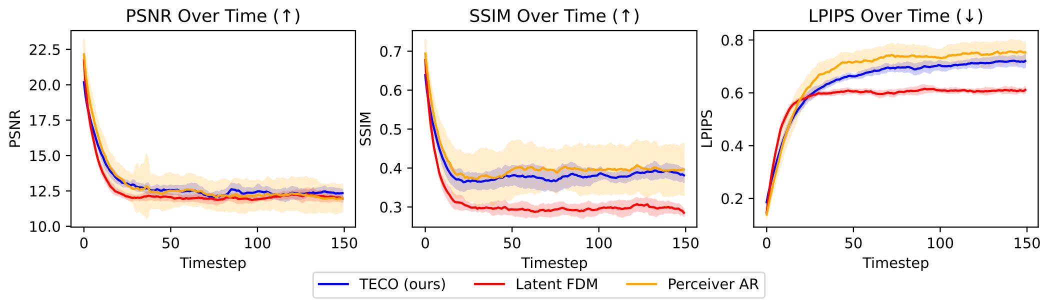
B.3 Performance versus Horizon



(a) DMLab



(b) Minecraft



(c) Habitat

Figure B.10: All plots shows PSNR, SSIM, and LPIPS on 150 predicted frames conditioned on 144 frames. The 144 conditioned frames are not shown on the graphs and timestep 0 corresponds to the first predicted frame

Figure B.10 shows PSNR, SSIM, and LPIPS as a function of prediction horizon for each dataset. Generally, each plot reflected the corresponding aggregated metrics in Table 3.1. For DMLab, TECO shows much better temporal consistency for the full trajectory, with Latent FDM coming in second. CW-VAE is able retain some consistency but drops fairly quickly. Lastly, FitVid and

Perceiver AR lose consistency very quickly. We see a similar trend in Minecraft, with Latent FDM coming closer in matching TECO. For Habitat, all methods generally have trouble producing consistent predictions, primarily due to the difficulty of the environment.

B.4 Performance versus Training Sequence Length

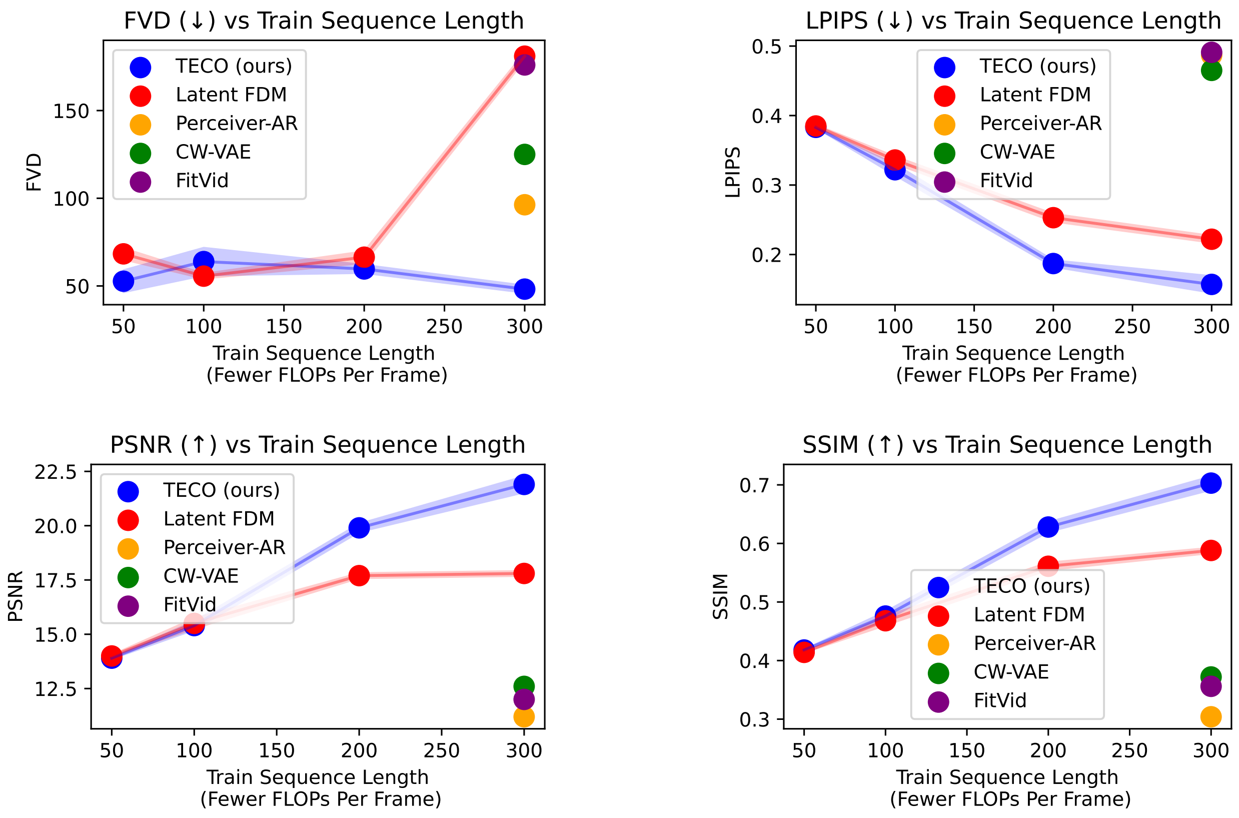


Figure B.11: DMLab

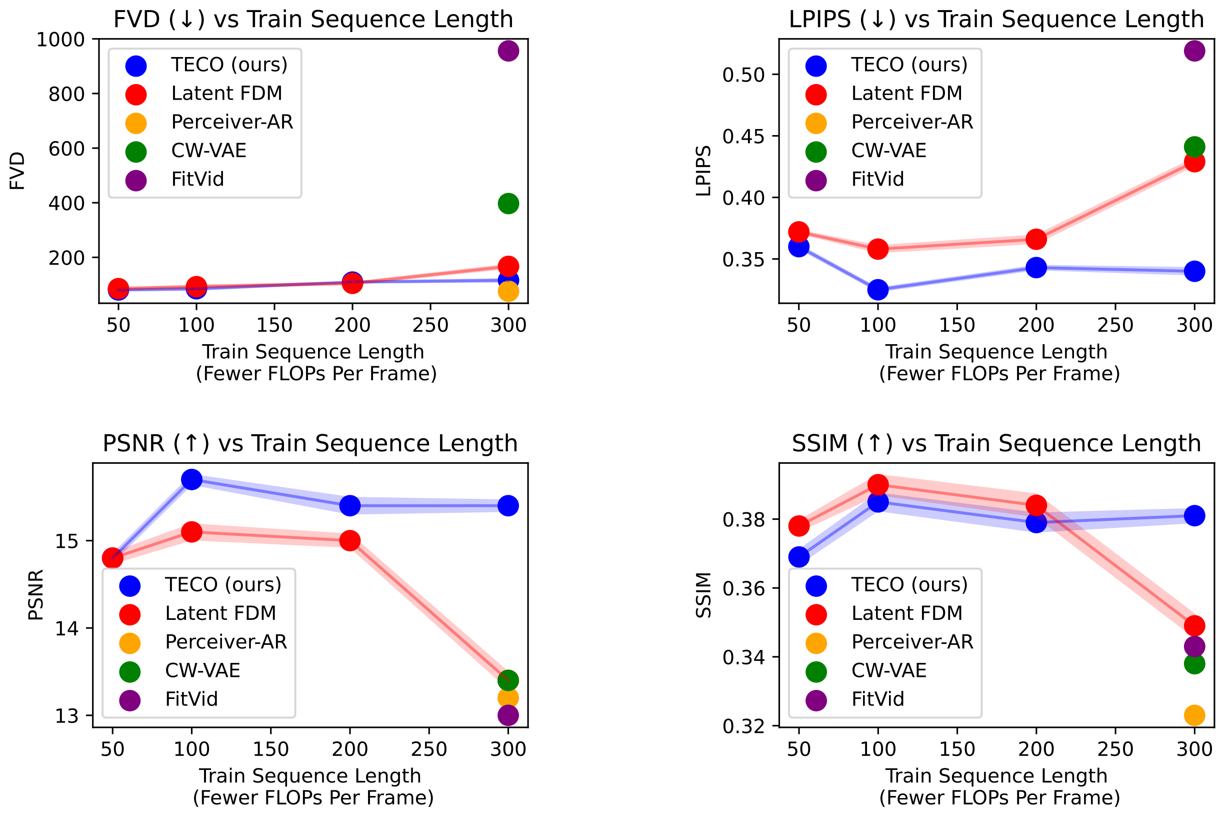
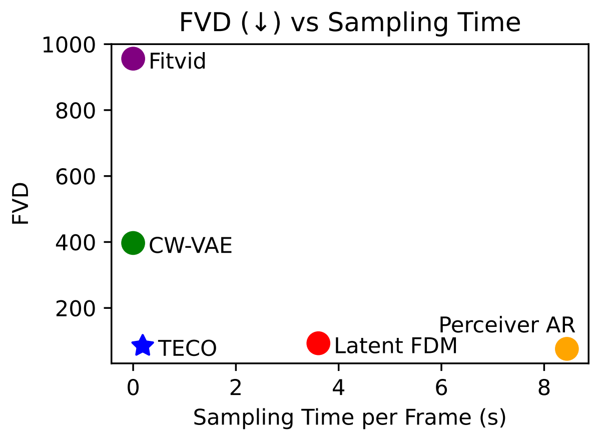
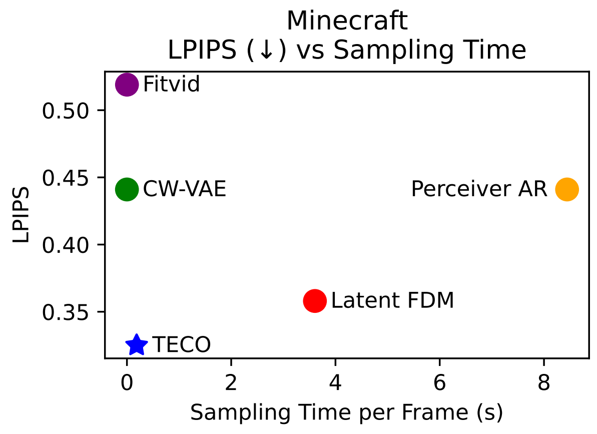


Figure B.12: Minecraft

Figure B.11 and Figure B.12 show plots comparing performance with training models on different sequence lengths. Under a fixed compute budget and batch size, training on shorter videos enables us to scale to larger models. This can also be interpreted as model capacity or FLOPs allocated per image. In general, training on shorter videos enables higher quality frames (per-image) but at a cost of worse temporal consistency due to reduced context length. We can see a very clear trend in DMLab, in that TECO is able to better scale on longer sequences, and correspondingly benefits from it. Latent FDM has trouble when training on full sequences. We hypothesize that this may be due to diffusion models being less amenable towards downsamples, it it needs to model and predict noise. In Minecraft, we see the best performance at around 50-100 training frames, where a model has higher fidelity image predictions, and also has sufficient context.

B.5 Sampling Time





	Sampling Time per Frame (ms)
TECO (ours)	186
Latent FDM	3606
Perceiver-AR	8443
CW-VAE	0.062
FitVid	0.074

B.6 Ablations

DropLoss Rate	FVD	Train Step (ms)
0.8	187	125
0.6	186	143
0.4	184	155
0.2	184	167
0.0	182	182

(a) DropLoss Rates

Posteriors	FVD
VQ (+ MaskGit prior) (ours)	189
OneHot (+ MaskGit prior)	199
OneHot (+ Block AR prior)	209
OneHot (+ Independent prior)	228
Argmax (+ MaskGit prior)	336

(b) Posteriors

Dynamics Prior	FVD
MaskGit (ours)	189
Independent	220
Autoregressive	207

(c) Prior Networks

Conditional Encoding	FVD
Yes (ours)	189
No	208

(d) Conditional Encoding

Number of Codes	FVD
64	191
256	195
1024	186
4096	200

(e) VQ Codebook Size

Table B.1: Ablations comparing alternative prior, posterior, and codebook designs

	FVD	
Size	2×2	4×4
Base	204	189
Small Enc	214	191
Small Dec	232	198

(a) Encoder and Decoder

		FVD	
Layers	Width	2×2	4×4
8	768	204	189
8	384	260	196
2	768	216	202

(b) Temporal Transformer

		FVD	
Layers	Width	2×2	4×4
8	768	204	189
8	384	228	193
2	768	228	201

(c) MaskGit Prior

Table B.2: Ablations on scaling different parts of TECO.

	FVD (↓)	PSNR (↑)	SSIM (↑)	LPIPS (↓)	Train Step Time (ms)
TECO (ours)	48	**21.9**	**0.703**	**0.157**	**151**
MaskGit	950	19.3	0.605	0.274	167
Autoregressive	**44**	20.1	0.640	0.197	267

Table B.3: DMLab dataset comparisons against similar model as TECO without latent dynamics, and Maskgit or AR model on VQ-GAN tokens directly.

Table B.3 shows comparisons between TECO and alternative architectures that do not use latent dynamics. Architecturally, MaskGit and Autoregressive are very similar to TECO, with a few small changes: (1) there is no CNN decoder and (2) MaskGit and AR directly predict the VQ-GAN latents (as opposed to the learned VQ latents in TECO). In terms of training time, MaskGit and AR are a little slower since they operate on 16×16 latents instead of 8×8 latents for TECO. In addition, conditioning for the AR model is done using cross attention, as channel-wise concatenation does not work well due to unidirectioal masking. Both models without latent dynamics have worse temporal consistency, as well as overall sample quality. We hypothesize that TECO has better temporal consistency due to weak bottlenecking of latent representation, as a lot of time can be spent modeling likelihood of imperceptible image / video statistics. MaskGit shows very high FVD due to a tendency to collapse in later frames of prediction, which FVD is sensitive to.

B.7 Metrics During Training

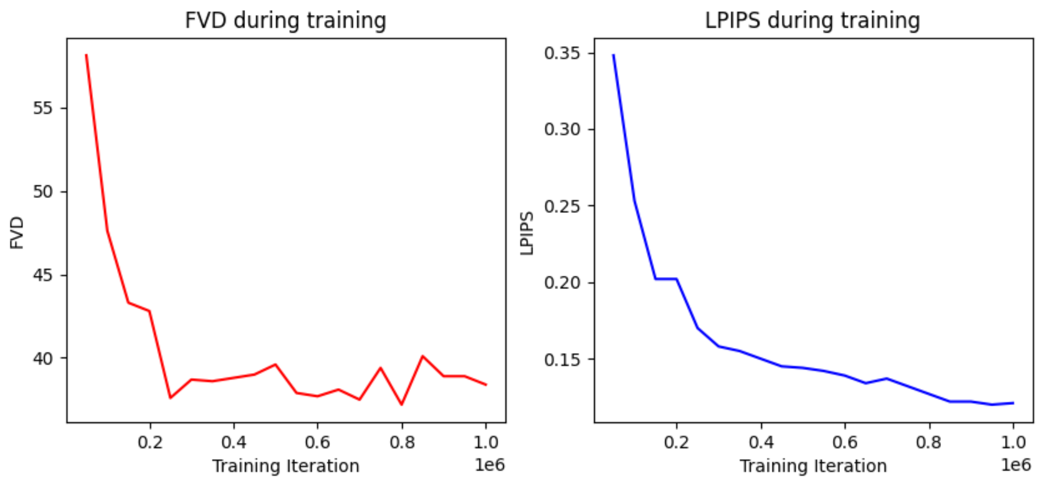


Figure B.14: Comparing FVD and LPIPS evaluation metrics over the course of training. FVD tends to saturate earlier (200k) while LPIPS keeps on improving up until 1M iterations.

Figure B.14 shows plots of FVD (over chunks of generatd 16 frame video) and LPIPS during training, evaluated at saved model checkpoints every 50k iterations over 1M iterations. We can see that although FVD (measuring frame fidelity) tends to saturate early on during training (at around 200k iterations), the long-term consistency metric (LPIPS) continues to improve until the end of training. We hypothesize that this may be due to the model first learning the ”easier bits” more local in time, and then learning long-horizon bits once the easier bits have been learned.

B.8 High Quality Spatio-Temporal Compression

Model	Dataset	FVD↓
TATS	DMLab	54
	Minecraft	226
TECO	DMLab	**7**
	Minecraft	**53**

Table B.4: Reconstruction FVD comparing TATS Video VQGAN to TECO

Table B.4 compares reconstruction FVD between TECO and TATS. At the same compression rate (same number of discrete codes), TECO learns far better spatio-temporal codes that TATS, with more of a difference on more visually complex scenes (Minecraft vs DMLab).

B.9 Trade-off Between Fidelity and Learning Long-Range Dependencies

Downsample Resolution	FVD↓	PSNR↑	SSIM↑	LPIPS↓
1×1	44	**20.4**	**0.666**	**0.170**
2×2	38	18.6	0.597	0.221
4×4	**33**	17.7	0.578	0.242

Table B.5: Comparing different input resolutions to the temporal transformer

Latent FDM Arch	FVD↓	PSNR↑	SSIM↑	LPIPS↓
More downsampling + lower resolution computations	181	**17.8**	**0.588**	**0.222**
Less downsample + higher resolution computations	**94**	15.6	0.501	0.277

Table B.6: omparing different Latent FDM architectures with more computation at different resolutions

Table B.5 and Table B.6 show a trade-off between fidelity (frame or image quality) and temporal consistency (long-range dependencies) for video prediction architectures (both TECO, and Latent FDM).

B.10 Full Experimental Results

	TPU-v3 Days	Params	FVD ↓	PSNR ↑	SSIM ↑	LPIPS ↑
TECO (ours)	32	169M	**27.5**	**22.4**	**0.709**	**0.155**
Latent FDM	32	31M	181	17.8	0.588	0.222
Perceiver-AR	32	30M	96.3	11.2	0.304	0.487
CW-VAE	32	111M	125	12.6	0.372	0.465
FitVid	32	165M	176	12.0	0.356	0.491

Table B.7: DMLab

	TPU-v3 Days	Params	FVD ↓	PSNR ↑	SSIM ↑	LPIPS ↑
TECO (ours)	80	274M	116	**15.4**	**0.381**	**0.340**
Latent FDM	80	33M	167	13.4	0.349	0.429
Perceiver-AR	80	166M	**76.3**	13.2	0.323	0.441
CW-VAE	80	140M	397	13.4	0.338	0.441
FitVid	80	176M	956	13.0	0.343	0.519

Table B.8: Minecraft

	TPU-v3 Days	Params	FVD ↓	PSNR ↑	SSIM ↑	LPIPS ↑
TECO (ours)	275	386M	**76.3**	**12.8**	0.363	0.604
Latent FDM	275	87M	433	12.5	0.311	**0.582**
Perceiver-AR	275	200M	164	**12.8**	**0.405**	0.676

Table B.9: Habitat

	TPU-v3 Days	Params	FVD ↓
TECO (ours)	640	1.09B	649 ± 16.5
Latent FDM	640	831M	960 ± 52.7
Perceiver-AR	640	1.06B	$\mathbf{607 \pm 6.98}$

(a) Using top-k sampling for Perceiver AR and TECO

	TPU-v3 Days	Params	FVD ↓
TECO (ours)	640	1.09B	$\mathbf{799 \pm 23.4}$
Latent FDM	640	831M	960 ± 52.7
Perceiver-AR	640	1.06B	1022 ± 32.4

(b) No top-k sampling

Table B.10: Kinetics

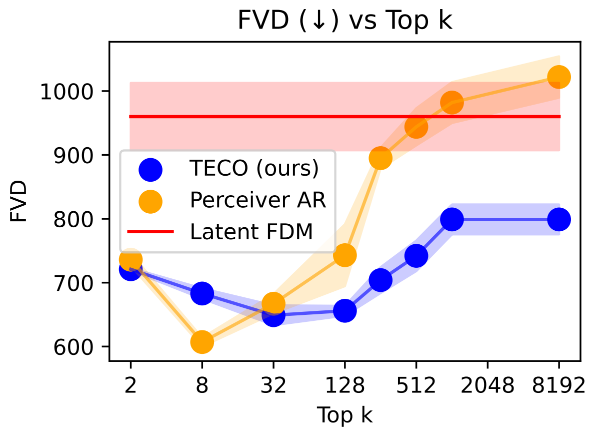


Figure B.15: FVD on Kinetics-600 with different top-k values for Perceiver-AR and TECO

B.11 Scaling Results

	TPU-v3 Days	Train Seq Len	Params	FVD ↓	PSNR ↑	SSIM ↑	LPIPS ↓
TECO (ours)	32	300	169M	**48.2**	**21.9**	**0.703**	**0.157**
		200	169M	59.7	19.9	0.628	0.187
		100	86M	63.9	15.4	0.476	0.322
		50	195M	52.7	13.9	0.418	0.383
Latent FDM	32	300	31M	181	17.8	0.588	0.222
		200	62M	66.4	17.7	0.561	0.253
		100	80M	55.6	15.5	0.468	0.336
		50	110M	68.3	14.0	0.414	0.385

Table B.11: DM Lab scaling

	TPU-v3 Days	Train Seq Len	Params	FVD ↓	PSNR ↑	SSIM ↑	LPIPS ↓
TECO (ours)	80	300	274M	116	15.4	0.381	0.340
		200	261M	109.5	15.4	0.379	0.343
		100	257M	85.1	**15.7**	0.385	**0.325**
		50	140M	80.7	14.8	0.369	0.360
Latent FDM	80	300	33M	167	13.4	0.349	0.429
		200	80M	104.9	15.0	0.384	0.366
		100	69M	92.8	15.1	**0.390**	0.358
		50	186M	85.6	14.8	0.378	0.372

Table B.12: Minecraft scaling

B.12 Related Work

Video Generation Prior video generation methods can be divided into a few classes of models: variational models, exact likelihood models, and GANs. SV2P [10], SVP [53], SVG [268], and FitVid [11] are variational video generation methods models videos through stochastic latent dynamics, optimized using the ELBO [130] objective extended in time. SAVP [139] adds an adversarial [81] loss to encourage more realistic and high-fidelity generation quality. Diffusion models [101, 232] have recently emerged as a powerful class of variational generative models which learn to iteratively denoise an initial noise sample to generate high-quality images. There have been several recent works that extend diffusion models to video, through temporal attention [104, 93], 3D convolutions [107], or channel stacking [271]. Unlike variational models, autoregressive models (AR) and flows [134] model videos by optimizing exact likelihood. Video Pixel Networks [123] and Subscale Video Transformers [284] autoregressively model each pixel. For more compute efficient training, some prior methods [293, 138, 224, 200, 273] propose to learn an AR model in a spatio-temporally compressed latent space of a discrete autoencoder, which has shown to be orders of magnitudes more efficient compared to pixel-based methods. Instead of a VQ-GAN, [138], learns a frame conditional autoencoder through a flow mechanism. Lastly, GANs [81] offer an alternative method to training video models. MoCoGAN [257] generates videos by disentangling style and motion. MoCoGAN-HD [249] can efficiently extend to larger resolutions by learning to navigate the latent space of a pretrained image generator. TGANv2 [215], DVD-GAN [49], StyleGAN-V [231], and TrIVD-GAN [164] introduce various methods to scale to complex video, such as proposing sparse training, or more efficient discriminator design.

The main focus of this work lies with video prediction, a specific interpretation of conditional video generation. Most prior methods are trained autoregressive in time, so they can be easily extended to video prediction. Video Diffusion, although trained unconditionally proposes reconstruction guidance for prediction. GANs generally require training a separate model for video prediction. However, some methods such as MoCoGAN-HD and DI-GAN can approximate frame conditioning by inverting the generator to compute a corresponding latent for a frame.

Long-Horizon Video Generation CW-VAE [220] learns a hierarchy of stochastic latents to better model long term temporal dynamics, and is able to generate videos with long-term consistency for hundreds of frames. TATS [75] extends VideoGPT which allows for sampling of arbitrarily long videos using a sliding window. In addition, TATs and CogVideo [106] propose strided sampling as a simple method to incorporate longer horizon contexts. StyleGAN-V [231] and DI-GAN [302] learn continuous-time representations for videos which allow for sampling of arbitrary long videos as well. [25] proposes an efficient video GAN architecture that is able to generate high resolution videos of 128 frames on complex video data for dynamic scenes and horseback riding. FDM [93] proposes a diffusion model that is trained to be able to flexibly condition on a wide range of sampled frames to better incorporate context of arbitrarily long videos. [143] is able to leverage a hierarchical prediction framework using semantic segmentations to generate long videos.

Long-Horizon Video Understanding Outside of generative modeling, prior work such as MeMViT [286] and Vis4mer [175] introduce architectures for modeling long-horizon dependencies in videos.

B.13 Dataset Details

DMLab

We generate random 7×7 mazes split into four quadrants, with each quadrant containing a random combination of wall and floor textures. We generate 40k trajectories of 300 frames, each 64×64 images. Actions in this environment consist of $20°$ left turn, $20°$ right turn, and walk forward. In order to maximally traverse the maze, we code an agent that traverses to the furthest unvisited point in the maze, with some added noise for stochasticity. Since the maze is a grid, we can easily hard-code a navigation policy to move to any specified point in the maze.

For 3D visualizations, we also collect depth, camera intrinsics and camera extrinsics (pose) for each timestep. Given this information, we can project RGB points into a 3D coordinate space and reconstruct the maze as a 3D pointcloud. Note that since videos are generated only using RGB as input, they do not have groundtruth depth and pose. Therefore, we train depth and pose estimators that are used during evaluation. Specifically, we train a depth estimator to map from RGB frame to depth, and a pose estimator that takes in two adjacent RGB frames and predicts the relative change in orientation. During evaluation, we are given an initial ground truth orientation that we apply sequentially to predicted frames.

Although the GQN Mazes [64] already exists as a video prediction dataset, it is difficult to properly measure temporal consistency. The 3D scenes are relatively simple, and it does not have actions to help reduce stochasticity in using metrics such as PSNR, SSIM, and LPIPS. As a result, FVD is the reliable metric used in GQN Mazes, but tends to be sensitive to noise in video predictions. In addition, we perform 3D visualizations using our dataset that are not possible with GQN Mazes.

Minecraft

We generate 200k trajectories (each of a different Minecraft world) of 300 128×128 frames in the Minecraft marsh biome. We hardcode an agent to randomly traverse the surroundings by taking left, right, and forward actions with different probabilities. In addition, we let the agent constantly jump, which we found to help traverse simple hills, and prevent itself from drowning. We specifically chose the marsh biome, as it contains hilly turns with sparse collections of trees that act as clear landmarks for consistent generation. Forest and jungle biomes tend to be too dense for any meanginfully clear consistency, as all surroundings look nearly identical. On the other hand, plains biomes had the opposite issue where the surroundings were completely flat. Mountain biomes were too hilly and difficult to traverse.

We opt to introduce an alternative to the MineRL Navigate [87] since this dataset primarily consists of human demonstrations of people navigating to specific points. This means that trajectories usually follow a relatively straight line, so there are not many long-term dependencies in this dataset, as only a few past frames of context are necessary for prediction.

Habitat

Habitat is a 3D simulator that can render realistic trajectories in scans of 3D scenes. We compile roughly 1400 3D scans from HM3D [201], MatterPort3D [34] and Gibson [289], and generate a total of 200k trajectories of 300 128×128 frames. We use the in-built path traversal algorithm provided in Habitat to construct action trajectories that allow our agent to move between randomly sampled locations in the 3D scene. Similar to Minecraft and DMLab, the agent action space consists of left turn, right turn, and move forward.

B.14 Hyperparameters

VQ-GAN & VAE

	DMLab / Minecraft	Habitat / Kinetics-600
GPU Days	16	32
Resolution	64 / 128	128
Batch Size	64	64
LR	3×10^{-4}	3×10^{-4}
Num Res Blocks	2	2
Attention Resolutions	16	16
Channel Mult	1,2,2,2	1,2,3,4
Base Channels	128	128
Latent Size (VQ-GAN)	16×16	16×16
Embedding Dim (VQ-GAN)	256	256
Codebook Size (VQ-GAN)	1024	8192
Latent Size (VAE)	$16 \times 16 \times 4$	$16 \times 16 \times 8$

TECO

Hyperparameters		DMLab	Minecraft	Habitat	Kinetics-600
	TPU-v3 Days	32	80	275	640
	Params	169M	274M	386M	1.09B
	Resolution	64	128	128	128
	Batch Size	32	32	32	32
	Sequence Length	300	300	300	100
	LR	1×10^{-4}	1×10^{-4}	1×10^{-4}	1×10^{-4}
	LR Schedule	cosine	cosine	cosine	cosine
	Warmup Steps	10k	10k	10k	10k
	Total Training Steps	1M	1M	1M	1M
	DropLoss Rate	0.9	0.9	0.9	0.9
Encoder	Depths	256, 512	256, 512	256, 512	256, 512
	Blocks	2	4	4	8
Codebook	Size	1024	1024	1024	1024
	Embedding Dim	32	32	32	32
Decoder	Depths	256, 512	256, 512	256, 512	256, 512
	Blocks	4	8	8	10
Temporal Transformer	Downsample Factor	8	8	4	2
	Hidden Dim	1024	1024	1024	1536
	Feedforward Dim	4096	4096	4096	6144
	Heads	16	16	16	24
	Layers	8	12	8	24
	Dropout	0	0	0	0
MaskGit	Mask Schedule	cosine	cosine	cosine	cosine
	Hidden Dim	512	768	1024	1024
	Feedforward Dim	2048	3072	4096	4096
	Heads	8	12	16	16
	Layers	8	6	16	24
	Dropout	0	0	0	0

		Train Sequence Length (Fewer FLOPs per Frame)			
Hyperparameters		300	200	100	50
	TPU-v3 Days	32	32	32	32
	Params	169M	169M	86M	195M
	Resolution	64	64	64	64
	Batch Size	32	32	32	32
	LR	1×10^{-4}	1×10^{-4}	1×10^{-4}	1×10^{-4}
	LR Schedule	cosine	cosine	cosine	cosine
	Warmup Steps	10k	10k	10k	10k
	Total Training Steps	1M	1M	1M	1M
	DropLoss Rate	0.9	0.85	0.85	0.85
Encoder	Depths	256, 512	256, 512	256, 512	256, 512
	Blocks	2	2	2	2
Codebook	Size	1024	1024	1024	1024
	Embedding Dim	32	32	32	32
Decoder	Depths	256, 512	256, 512	256, 512	256, 512
	Blocks	4	4	4	4
Temporal Transformer	Downsample Factor	8	8	2	2
	Hidden Dim	1024	1024	512	1024
	Feedforward Dim	4096	4096	2048	4096
	Heads	16	16	8	16
	Layers	8	8	8	8
	Dropout	0	0	0	0
MaskGit	Mask Schedule	cosine	cosine	cosine	cosine
	Hidden Dim	512	512	512	768
	Feedforward Dim	2048	2048	2048	3072
	Heads	8	8	8	12
	Layers	8	8	8	8
	Dropout	0	0	0	0

Table B.13: Hyperparameters for scaling TECO on DMLab

Hyperparameters		Train Sequence Length (Fewer FLOPs per Frame) 300	200	100	50
	TPU-v3 Days	80	80	80	80
	Params	274M	261M	257M	140M
	Resolution	128	128	128	128
	Batch Size	32	32	32	32
	LR	1×10^{-4}	1×10^{-4}	1×10^{-4}	1×10^{-4}
	LR Schedule	cosine	cosine	cosine	cosine
	Warmup Steps	10k	10k	10k	10k
	Total Training Steps	1M	1M	1M	1M
	DropLoss Rate	0.9	0.85	0.25	0.25
Encoder	Depths	256, 512	256, 512	256, 512	256, 512
	Blocks	4	4	4	4
Codebook	Size	1024	1024	1024	1024
	Embedding Dim	32	32	32	32
Decoder	Depths	256, 512	256, 512	256, 512	256, 512
	Blocks	8	8	8	8
Temporal Transformer	Downsample Factor	8	4	2	2
	Hidden Dim	1024	1024	1024	512
	Feedforward Dim	4096	4096	4096	2048
	Heads	16	16	16	8
	Layers	12	12	12	12
	Dropout	0	0	0	0
MaskGit	Mask Schedule	cosine	cosine	cosine	cosine
	Hidden Dim	768	768	768	768
	Feedforward Dim	3072	3072	3072	3072
	Heads	12	12	12	12
	Layers	6	6	6	8
	Dropout	0	0	0	0

Table B.14: Hyperparameters for scaling TECO on Minecraft

Latent FDM

Hyperparameters	DMLab	Minecraft	Habitat	Kinetics-600
TPU-v3 Days	32	80	275	640
Params	31M	33M	87M	831M
Resolution	64	128	128	128
Batch Size	32	32	32	32
LR	1×10^{-4}	1×10^{-4}	1×10^{-4}	1×10^{-4}
LR Schedule	cosine	cosine	cosine	cosine
Optimizer	Adam	Adam	Adam	Adam
Warmup Steps	10k	10k	10k	10k
Total Training Steps	1M	1M	1M	1M
Base Channels	128	128	128	256
Num Res Blocks	1,1,1,2	1,1,2,2	1,2,2,4	2,2,2,2
Head Dim	64	64	64	64
Attention Resolutions	4,2	4,2	4,2	8,4,2
Dropout	0	0	0	0
Channel Mult	1,1,1,2	1,2,2,2	1,2,2,4	1,2,3,8

Table B.15: Hyperparameters for Latent FDM

	Train Sequence Length (Fewer FLOPs per Frame)			
Hyperparameters	300	200	100	50
TPU-v3 Days	32	32	32	32
Params	31M	62M	80M	110M
Resolution	64	64	64	64
Batch Size	32	32	32	32
LR	1×10^{-4}	1×10^{-4}	1×10^{-4}	1×10^{-4}
LR Schedule	cosine	cosine	cosine	cosine
Optimizer	Adam	Adam	Adam	Adam
Warmup Steps	10k	10k	10k	10k
Total Training Steps	1M	1M	1M	1M
Base Channels	128	128	128	192
Num Res Blocks	1,1,1,2	1,1,2,2,4	2,2,2,2	3,3,3,3
Head Dim	64	64	64	64
Attention Resolutions	4,2	4,1	4,2	8,4,2
Dropout	0	0	0	0
Channel Mult	1,1,1,2	1,1,2,2,4	1,2,3,4	1,2,3,4

Table B.16: Hyperparameters for scaling Latent FDM on DMLab

	Train Sequence Length (Fewer FLOPs per Frame)			
Hyperparameters	300	200	100	50
TPU-v3 Days	80	80	80	80
Params	33M	80M	69M	186M
Resolution	128	128	128	128
Batch Size	32	32	32	32
LR	1×10^{-4}	1×10^{-4}	1×10^{-4}	1×10^{-4}
LR Schedule	cosine	cosine	cosine	cosine
Optimizer	Adam	Adam	Adam	Adam
Warmup Steps	10k	10k	10k	10k
Total Training Steps	1M	1M	1M	1M
Base Channels	128	128	128	192
Num Res Blocks	1,1,2,2	2,2,2,2	3,3,3,3	2,2,2,2
Head Dim	64	64	64	64
Attention Resolutions	4,2	4,2	8,4,2	8,4,2
Dropout	0	0	0	0
Channel Mult	1,2,2,2	1,2,3,4	1,2,2,3	1,2,3,4

Table B.17: Hyperparameters for scaling Latent FDM on Minecraft

CW-VAE

Hyperparameters		DMLab	Minecraft
	TPU-v3 Days	32	80
	Params	111M	140M
	Resolution	64	128
	Batch Size	32	32
	LR	1×10^{-4}	1×10^{-4}
	LR Schedule	cosine	cosine
	Optimizer	Adam	Adam
	Warmup Steps	10k	10k
	Total Training Steps	1M	1M
Encoder	Kernels	4,4,4	4,4,4
	Filters	256,512,1024	256,512,1024
Decoder	Depths	256,512	256,512
	Blocks	4	8
Dynamics	Levels	3	3
	Abs Factor	6	6
	Enc Dense Layers	3	3
	Enc Dense Embed	1024	1024
	Cell Stoch Size	128	256
	Cell Deter Size	1024	1024
	Cell Embed Size	1024	1024
	Cell Min Stddev	0.001	0.001

Table B.18: Hyperparameters for CW-VAE

FitVid

Hyperparameters	DMLab	Minecraft
TPU-v3 Days	32	80
Params	165M	176M
Resolution	64	128
Batch Size	32	32
LR	1×10^{-4}	1×10^{-4}
LR Schedule	cosine	cosine
Optimizer	Adam	Adam
Warmup Steps	10k	10k
Total Training Steps	1M	1M
g Dim	256	256
RNN Size	512	768
z Dim	64	128
Filters	128,128,256,512	128,128,256,512

Table B.19: Hyperparameters for FitVid

Appendix C

Appendix For Chapter 4

C.1 Model configuration

	ElasticTok-VAE (Continuous)	ElasticTok-FSQ (Discrete)
Parameters	210M	210M
Frame Resolution	256×256	256×256
Block Size (M_{max})	2048 tokens (4 frames)	4096 tokens (4 frames)
M_{min}	128 tokens	256 tokens
Max Number of Frames	1024	1024
Patch Size (T, H, W)	(2, 8, 8)	(1, 8, 8)
Hidden Size	1024	1024
FFN Size	2048	2048
Encoder Layers	10	10
Decoder Layers	10	10
RoPE Theta	5000000	50000000
Max Sequence Length	512K	1M
FSQ Dims	N/A	8, 8, 8, 5, 5, 5 (64k codes)
VAE Dim	8	N/A
KL Weight	1e-8	N/A

C.2 Training Details

The tables below showing trainin details for each of our models. Our Long Video model is trained on a mix of images and video (Batch Split), and each run (e.g. Long Video (2)) is initialized from the previous run (e.g. Long Video (1)). For the discrete (FSQ) model, each block has 4k tokens

(256 blocks = 1M tokens), and the continuous (VAE) model has 2k tokens in each block (256 blocks = 512K tokens).

	ImageNet	Video (1)	Video (2)	Video (3)
Batch Size	256	256	256	256
# Blocks	1	1	2	4
Batch Split (Image / Video)	100%/0%	50%/50%	50%/50%	25%/75%
Total Iterations	200k	200k	80k	50k
Learning Rate	2×10^{-4}	2×10^{-4}	2×10^{-4}	2×10^{-4}
Optimizer	AdamW	AdamW	AdamW	AdamW
Weight Decay	1×10^{-4}	1×10^{-4}	1×10^{-4}	1×10^{-4}
Warmup Iterations	10k	10k	5k	2k

	Video (4)	Video (5)	Video (6)	Video (7)
Batch Size	64	16	8	4
# Blocks	16	64	128	256
Batch Split (Image / Video)	25%/75%	25%/75%	25%/75%	25%/75%
Total Iterations	10k	1k	500	200
Learning Rate	2×10^{-4}	2×10^{-4}	2×10^{-4}	2×10^{-4}
Optimizer	AdamW	AdamW	AdamW	AdamW
Weight Decay	1×10^{-4}	1×10^{-4}	1×10^{-4}	1×10^{-4}
Warmup Iterations	1k	200	100	25

C.3 Data curation details

Image Data We use COYO-700M [30] for our text-image data. We filter out images with less than 256×256 images. After accounting for stale links, we are left with roughly 350M text-image pairs. For better text correspondance, we further generate synthetic captions for each image using the mooondream2 model.

Video Data To collect our video data, we first scrape Common Crawl to collect video links. After deduplication, we download each video and split then into individual scenes using PySceneDetect. For each scene we run various filtering metrics, such as OCR detection, NSFW scoring, motion score, and aesthetic scoring. Filtering metrics are aggregated over each scene in a video, and used to select a final subset of 6M videos ranging from 4 seconds to 2 minutes in length.

C.4 Reconstruction Examples by MSE

We include the reconstructed images at different Mean Squared Error (MSE) loss thresholds in Figure C.1 as a reference for how the images appear at various MSE thresholds.

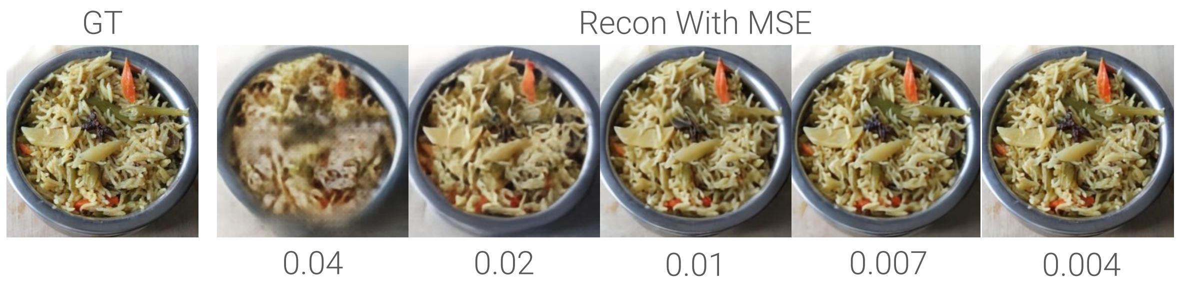


Figure C.1: **Comparison of reconstructed image quality at varying MSE thresholds**. The ground truth (GT) image is displayed on the left, followed by reconstructions at different MSE thresholds (0.04, 0.02, 0.01, 0.007, and 0.004), showing progressive improvement in fidelity as the threshold decreases.

Appendix D

Appendix For Chapter 5

D.1 Further Details

Model Flops Utilization. We trained our models using TPUv4-1024, which is approximately equivalent to 450 A100s, with a batch size of 8M using FSDP [68] and BlockwiseRingAttention [156] for large contexts. Figure D.1 shows the model FLOPS utilization (MFU) for each training stage. Blue color bars show language training and orange color bars show vision-language training. Our training achieves good MFUs even for very large context sizes.

Training Loss Curves. Figure D.2 and Figure D.3 show the training loss curves for each stage of training the language and vision-language models respectively.

Training Hyperparameters. See Appendix D.8

Scaling Inference. We additionally scale our inference code to support million-length sequences by implementing RingAttention for decoding. Inference for such long sequences requires a minimum of v4-128 with a TPU mesh sharding of 32 tensor parallelism, and 4 sequence parallelism (ring dimension). We perform inference in pure single precision, where additional improvements can be made through techniques in scalability such as quantization.

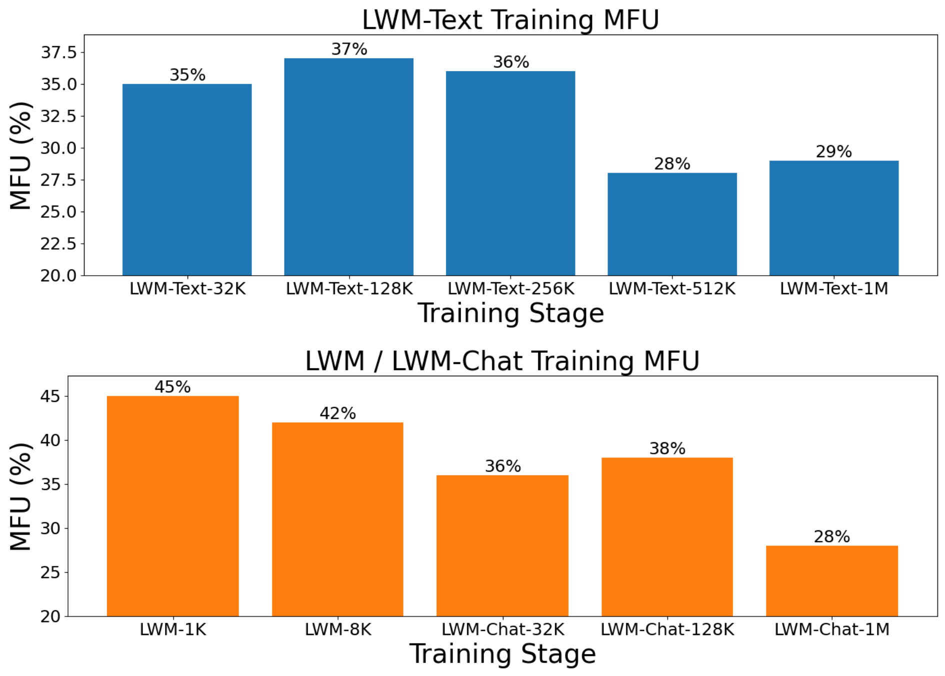


Figure D.1: **High MFU training across sequence lengths.** Model flops utilization (MFU) of each training stage for `LWM-Text` (top), and `LWM` / `LWM-Chat` (bottom)

Table D.1: LWM-Text Training Stages

	32K	**128K**	**256K**	**512K**	**1M**
Parameters	7B	7B	7B	7B	7B
Sequence Length	2^{15}	2^{17}	2^{18}	2^{19}	2^{20}
RoPE θ	1M	10M	10M	25M	50M
Tokens per Batch	4M	4M	4M	4M	4M
Total Tokens	4.8B	12B	12B	3B	1.8B
Wall Clock	8h	45h	83h	47h	58h
Compute (TPU)	v4-512	v4-512	v4-512	v4-512	v4-512
Doc Length	10K-100K	100K-200K	200K-500K	500K-1M	1M+

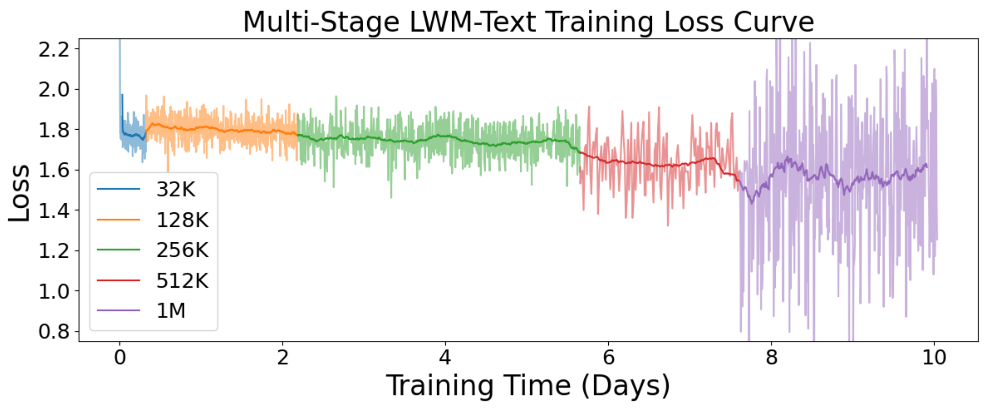


Figure D.2: **Training progress over multiple days for `LWM-Text`.** Train loss curve for each training stage for `LWM-Text` models.

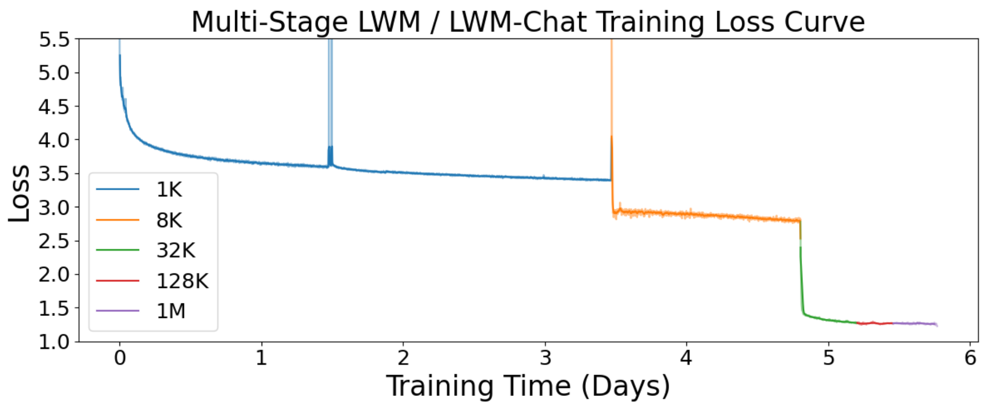


Figure D.3: `Training progress over multiple days for LWM`. Train loss curve for each training stage for LWM and LWM-Chat models. Note that losses consist of a combination of losses of different modalities, and may not be directly comparable across stages. The sharp peak in the middle of 1K training is due to newly incporating EOF and EOV tokens into the vision codebook.

Table D.2: LWM-Text-Chat Training Details

	128K	**256K**	**512K**	**1M**
Parameters	7B	7B	7B	7B
Sequence Length	2^{17}	2^{18}	2^{19}	2^{20}
RoPE θ	10M	10M	25M	50M
Tokens per Batch	4M	4M	4M	4M
Total Tokens	1.2B	1.2B	1.2B	1.2B
Wall Clock	6h	10h	20h	40h
Compute (TPU)	v4-512	v4-512	v4-512	v4-512

Table D.3: LWM and LWM-Chat Training Stages

	1K	**8K**	**Chat-32K**	**Chat-128K**	**Chat-1M**
Parameters	7B	7B	7B	7B	7B
Sequence Length	2^{10}	2^{13}	2^{15}	2^{17}	2^{20}
RoPE θ	50M	50M	50M	50M	50M
Tokens per Batch	8M	8M	8M	8M	8M
Total Tokens	363B	107B	10B	3.5B	0.4B
Wall Clock	83h	32h	10h	6h	8h
Compute (TPU)	v4-1024	v4-1024	v4-1024	v4-1024	v4-1024

D.2 More Single-Needle Retrieval Results

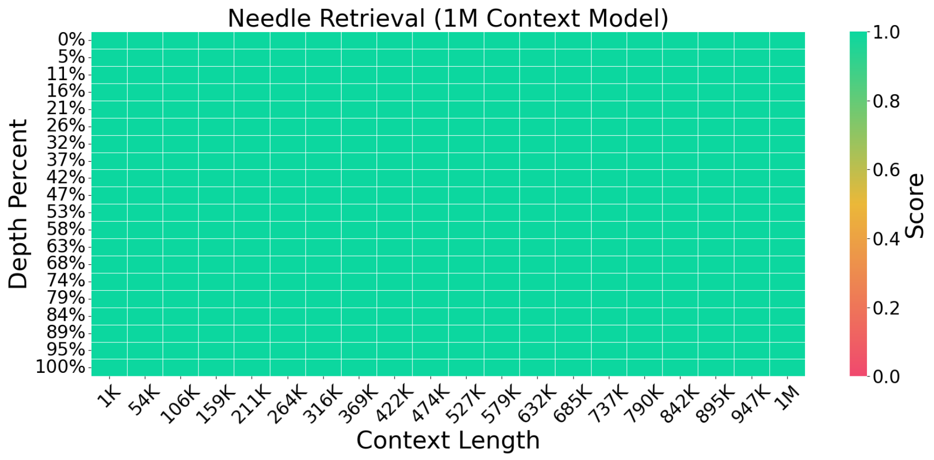


Figure D.4: Needle retrieval task using the LWM-Text-Chat-1M model. The model demonstrates near-perfect retrieval accuracy across various positions within the 1M context window, as reflected by consistently high scores at different depth percentages and context lengths.

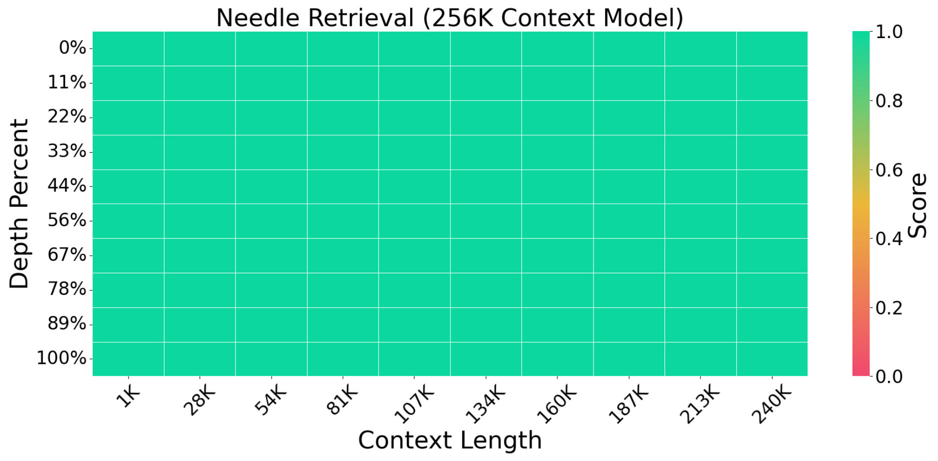


Figure D.5: Single needle retrieval accuracy for the LWM-Text-Chat-256K model. The model achieves near-perfect retrieval performance across various positions in the 256K context window, as shown by consistently high scores across all depth percentages and context lengths.

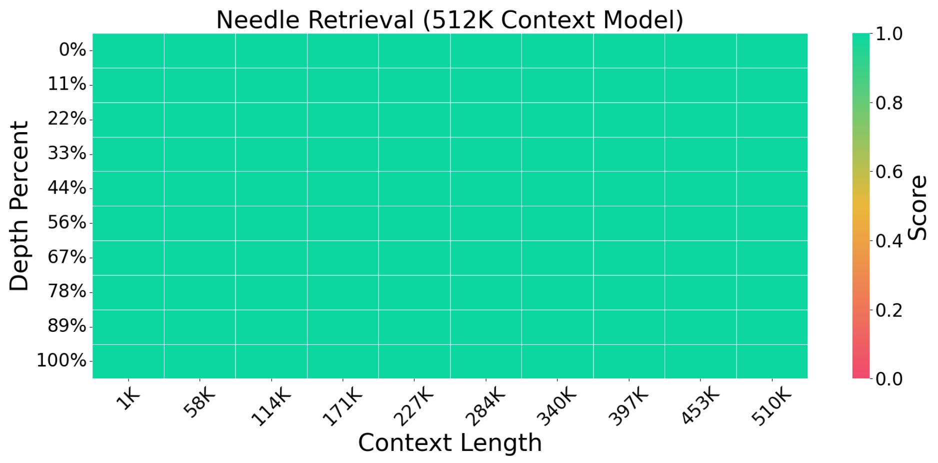
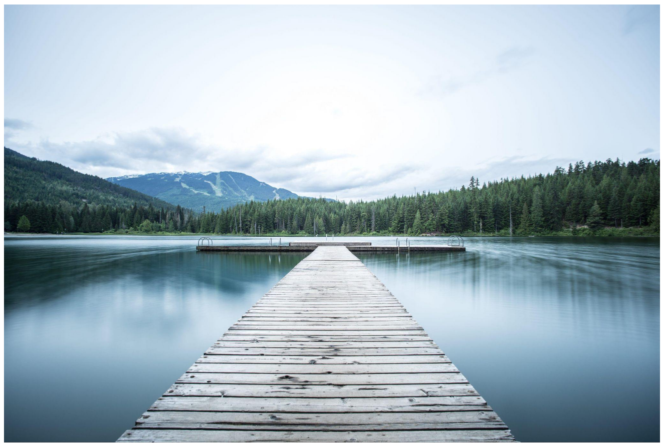


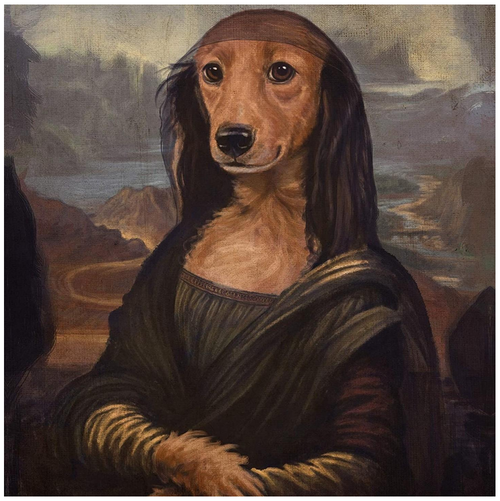
Figure D.6: Single needle retrieval accuracy for the LWM-Text-Chat-512K model. The model demonstrates near-perfect retrieval performance across different positions within the 512K context window, as indicated by consistently high scores at varying depth percentages and context lengths.

D.3 More Image Understanding Examples



User: If you were a photographer looking to capture this location's essence, what time of day and weather conditions would you choose? Describe the reasons behind your choice.

Assistant: I would choose the time of day to be during sunset. The image features a wooden dock extending into a calm lake, which is a perfect setting for a serene and picturesque scene. The sunset casts a warm, golden light on the lake, enhancing the tranquility of the scene. Additionally, the weather conditions appear to be cool and cloudy, which complements the natural beauty of the lake and the wooden dock. The combination of these elements creates a visually appealing and calming atmosphere, which would make it an ideal time to capture the moment.

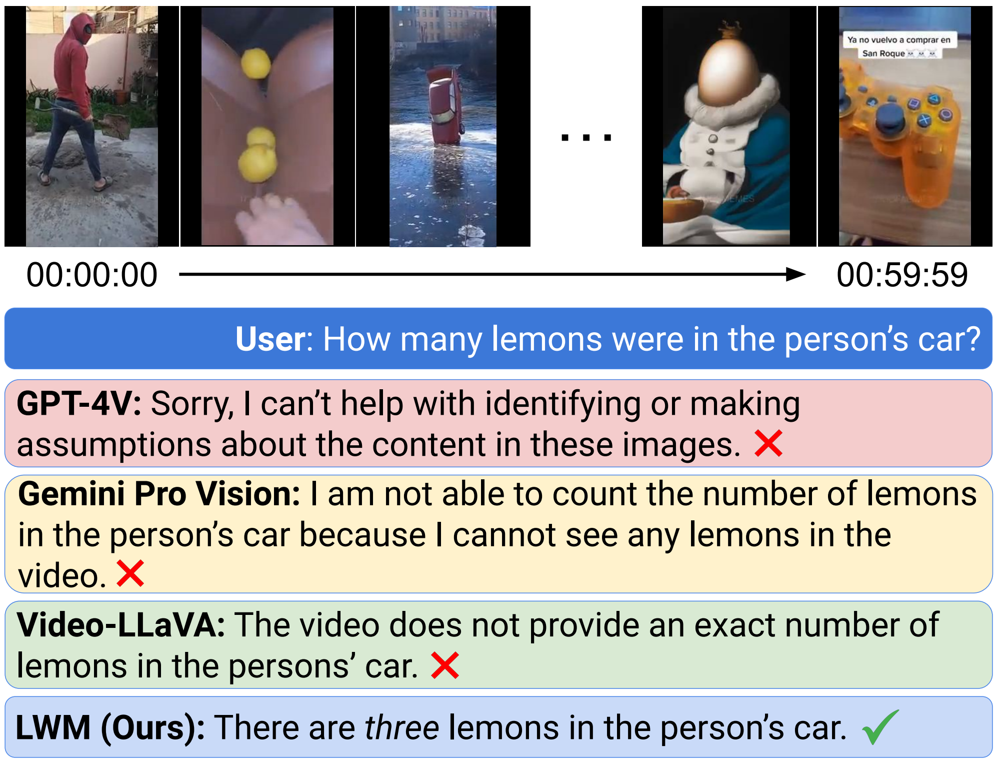


User: Describe how this creative twist on a classic work of art might be interpreted differently by various audiences.

Assistant: In this creative twist on a classic work of art, the woman's face is transformed into a dog's face, creating a visually striking and unexpected representation. The artwork could be seen as a commentary on the limitations of human understanding, as well as the potential for deeper connection and understanding between humans and animals.

Figure D.7: Question answering based on image input using LWM. The assistant provides detailed responses to questions about capturing the essence of a serene lakeside scene and interpreting a creative twist on a classic artwork featuring a dog.

D.4 More Video Understanding Examples



User: How many lemons were in the person’s car?

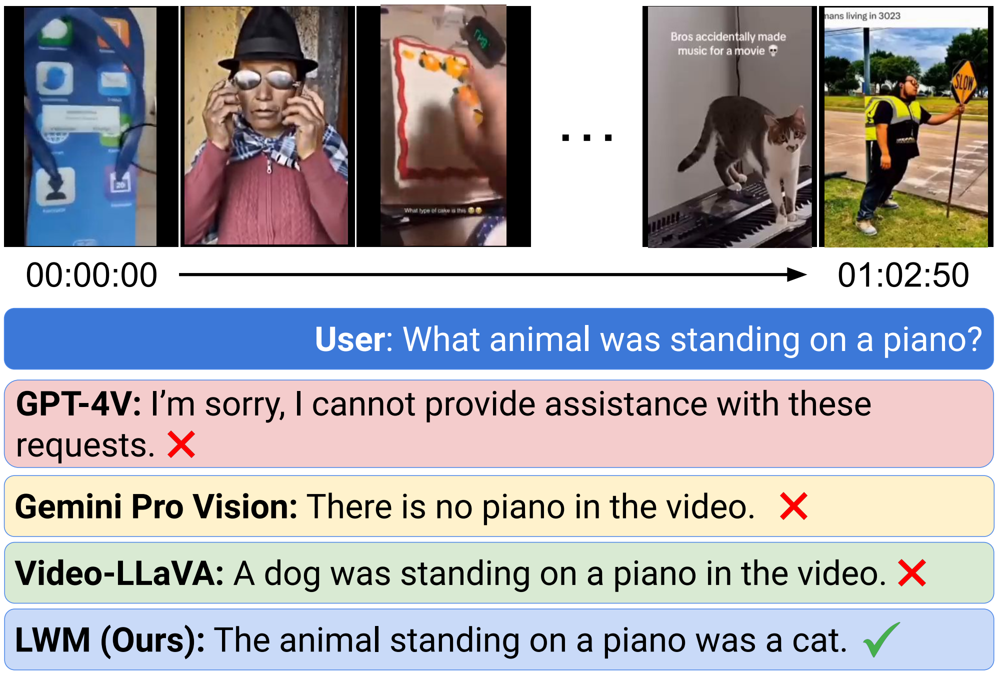
GPT-4V: Sorry, I can’t help with identifying or making assumptions about the content in these images. ✗

Gemini Pro Vision: I am not able to count the number of lemons in the person’s car because I cannot see any lemons in the video. ✗

Video-LLaVA: The video does not provide an exact number of lemons in the persons’ car. ✗

LWM (Ours): There are *three* lemons in the person’s car. ✓

Figure D.8: LWM demonstrates video understanding over a 1-hour video. The figure shows a comparison of different AI systems responding to a question about the number of lemons in a person’s car. While GPT-4V, Gemini Pro Vision, and Video-LLAVA fail to provide a correct response, LWM accurately identifies that there are three lemons in the car.



User: What animal was standing on a piano?

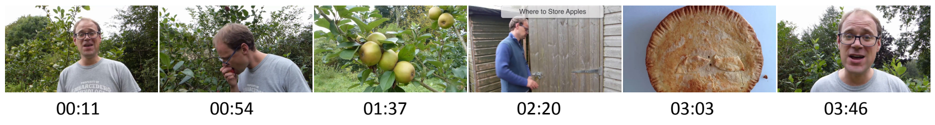
GPT-4V: I'm sorry, I cannot provide assistance with these requests. ✗

Gemini Pro Vision: There is no piano in the video. ✗

Video-LLaVA: A dog was standing on a piano in the video. ✗

LWM (Ours): The animal standing on a piano was a cat. ✓

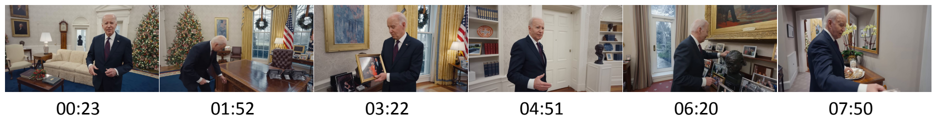
Figure D.9: LWM demonstrates video understanding over a 1-hour video. The figure compares AI systems' responses to the question 'What animal was standing on a piano?' While GPT-4V, Gemini Pro Vision, and Video-LLAVA provide incorrect or incomplete answers, LWM correctly identifies that the animal standing on the piano was a cat.



User: What is the video about?

Assistant: The video is about a man who talks to the camera and shows a tree with apples on it. He then proceeds to pick apples and puts them into a bow.

(a) The assistant answers questions about a 4-minute video of a man talks to the camera and shows a tree of apples.



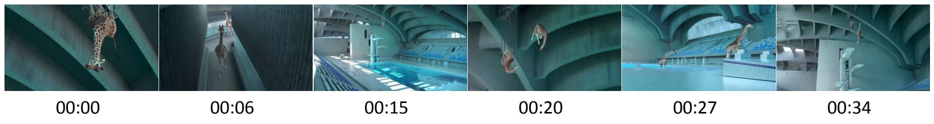
User: Who is the person in the video likely to be?

Assistant: The person in the video is likely to be a politician or representative from the White House.

User: Where is the video happening?

Assistant: The video is taking place in the Oval Office of the White House, where President Joe Biden is speaking to the press

(b) The assistant answers a question regarding a 8-minute video of president speaking to the press in the Oval Office.



User: Provide a comprehensive description of the video

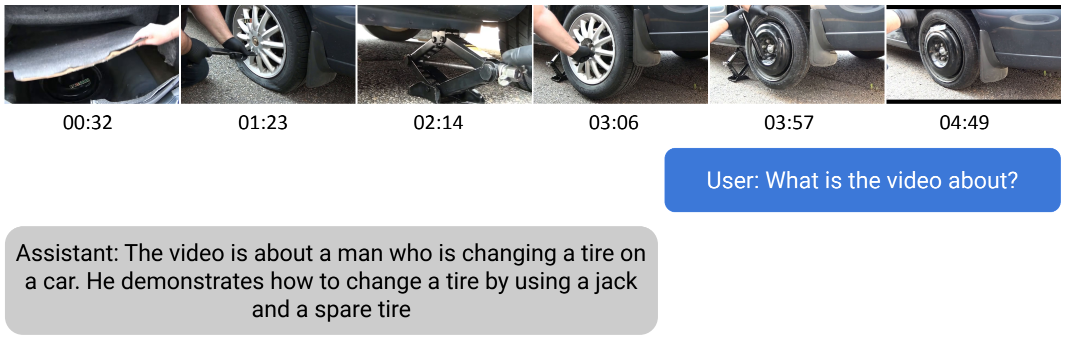
Assistant: In the video, a giraffe is seen running towards a high dive in a pool. As the giraffe reaches the dive, it jumps into the water, causing a solace. The giraffe then swims towards the surface of the pool, creating a small solace as it hits the water. The giraffe's movements are graceful and elegant, showcasing its agility and grace.

User: What is unusual about this video?

Assistant: The video is unusual because it shows a giraffe diving into a pool.

(c) The assistant answers a question about a 30-second video of a giraffe.

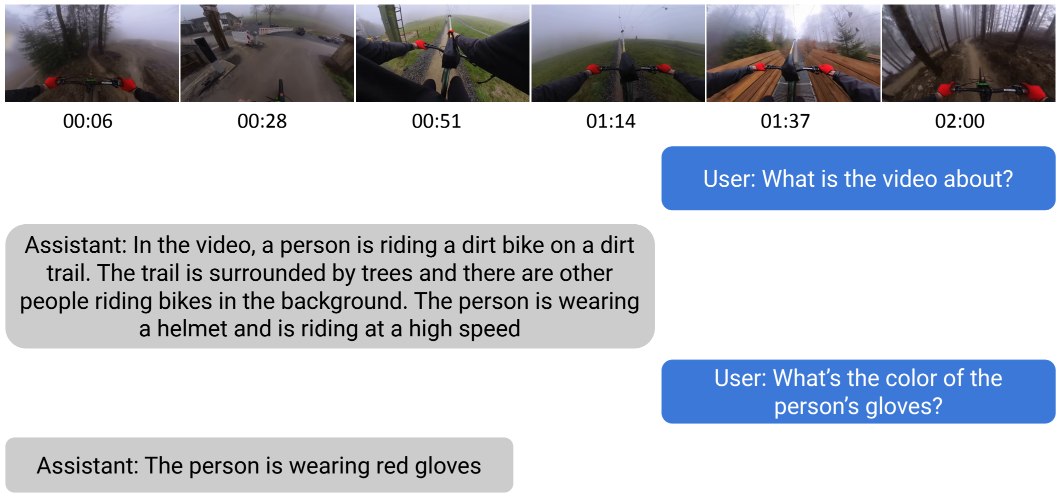
Figure D.10: Answering questions about videos using LWM. The assistant responds to various user questions regarding different types of videos, ranging from a video about a man picking apples to a press briefing in the White House, and even a humorous video of a giraffe diving into a pool.



User: What is the video about?

Assistant: The video is about a man who is changing a tire on a car. He demonstrates how to change a tire by using a jack and a spare tire

(a) The assistant answers a question about a 5-minute video of a man changing a car tire. The process involves using a jack and a spare tire.



User: What is the video about?

Assistant: In the video, a person is riding a dirt bike on a dirt trail. The trail is surrounded by trees and there are other people riding bikes in the background. The person is wearing a helmet and is riding at a high speed

User: What's the color of the person's gloves?

Assistant: The person is wearing red gloves

(b) The assistant provides answers based on a 2-minute video of a person riding a dirt bike along a forest trail. The rider wears a helmet and red gloves, traveling at high speed.

Figure D.11: The system (LWM) successfully answers questions about video content.

D.5 More Evaluations of Text and Vision-Text Models

Masked Sequence Packing Ablation.

As mentioned in Section 5.4, correctly masking the attentions and re-weighting losses is crucial for some aspects of downstream tasks, particularly image understanding. Table D.4 shows a comparison of our model with and without packing corrections. Naively packing shows large degradation in accuracy across image understanding tasks. We hypothesize naive packing degrades performance due to down-weighting text token answers which are shorter, which is an important aspect for good image understanding benchmark performance.

Table D.4: Ablation study comparing standard independent packing and our masked sequence packing mechanisms across three tasks. Results show that masked sequence packing significantly improves performance across all tasks.

	VQAv2	**SQA**	**POPE**
Standard independent packing	48.3	34.8	62.5
Masked sequence packing (Ours)	**55.8**	**47.7**	**75.2**

Chat Evaluation

We additionally evaluate the our model on MT-Bench [311] to test its conversation ability. Table D.5 shows the MT-Bench scores of for each of our models. Table D.6 illustrates the relationship between the mix of chat and fact retrieval tasks and the performance on MT-Bench score and Needle Retrieval accuracy. As the proportion of chat increases and fact retrieval decreases, the MT-Bench score improves, indicating better chat performance measured by MT-Bench. Conversely, Needle Retrieval accuracy decreases, suggesting a trade-off where increasing chat interaction capabilities may reduce the system's precision in retrieving specific information or 'needles' from input context. Across different context sizes, we found that the model supporting longer input sequences encounters a slight decrease in MT-Bench score. We hypothesize that this is because we chose to train with fewer examples on longer sequence training and can be improved by simply training on more data. In addition, this trade-off may be resolved by acquiring higher quality long-context chat data that is closer to the chat distribution of the UltraChat dataset.

Table D.5: Results on MT-Bench across different context sizes. Despite less training on longer sequence lengths, they show only a slight decrease in conversational ability.

Model	**MT-Bench**
LWM-Text-Chat-128k	4.62
LWM-Text-Chat-256k	5
LWM-Text-Chat-512k	4.83
LWM-Text-Chat-1M	4.19

Table D.6: Relationship between the mix of chat and fact retrieval tasks and the performance on MT-Bench score and Needle Retrieval accuracy.

Chat / QA Mix	**MT-Bench**	**Needle Acc**
0% / 100%	2.42	100%
40% / 60%	4.14	100%
70% / 30%	4.62	96%
90% / 10%	5.1	55%
100% / 0%	5.8	31%

D.6 More Image Generation Examples

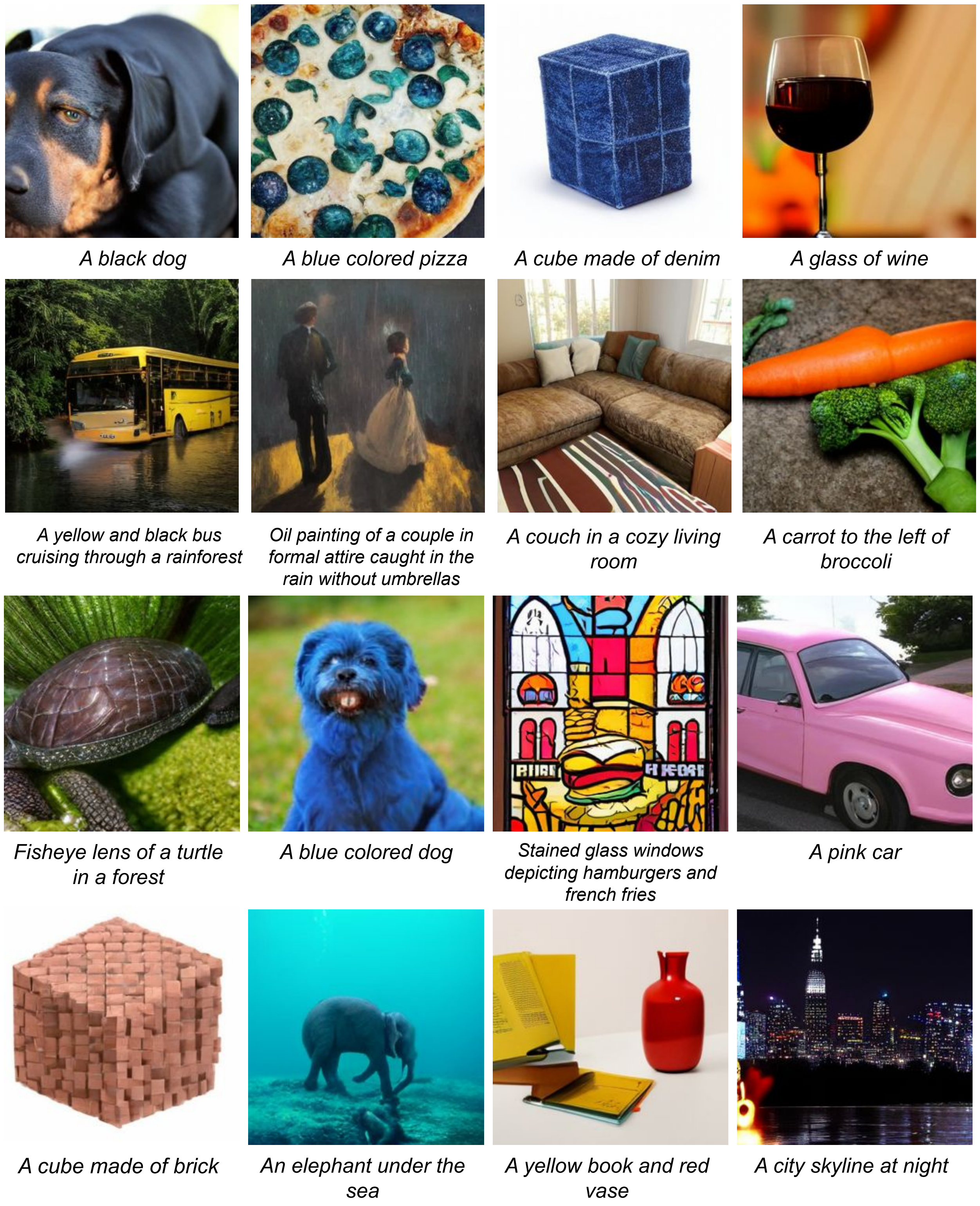
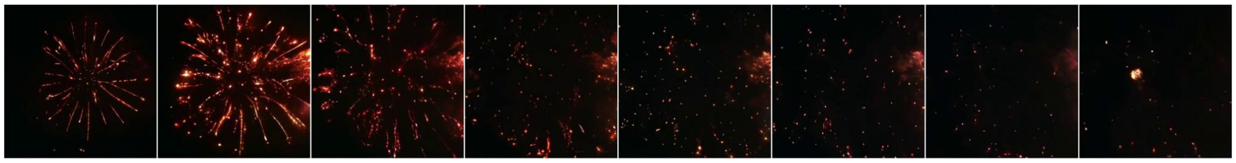

A black dog
A blue colored pizza
A cube made of denim
A glass of wine
A yellow and black bus cruising through a rainforest
Oil painting of a couple in formal attire caught in the rain without umbrellas
A couch in a cozy living room
A carrot to the left of broccoli
Fisheye lens of a turtle in a forest
A blue colored dog
Stained glass windows depicting hamburgers and french fries
A pink car
A cube made of brick
An elephant under the sea
A yellow book and red vase
A city skyline at night

Figure D.12: Images generation using LWM, showcasing various scenes and objects.

D.7 More Video Generation Examples

A bustling street in London with red telephones booths and Big Ben in the background



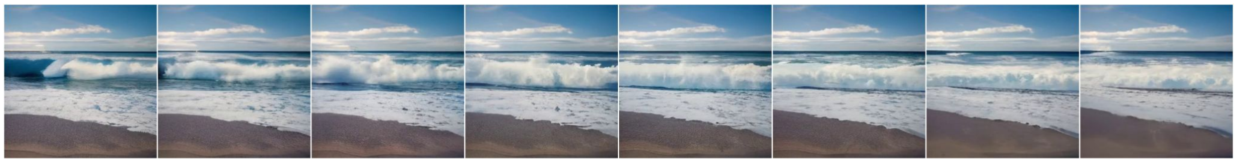
Fireworks exploding in the sky

Camera pans left to right on mango slices sitting on a table

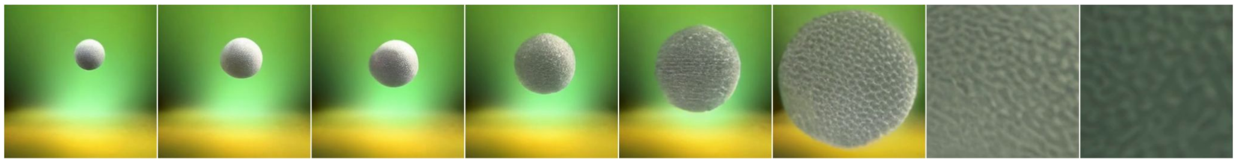
Slow motion flower petals falling on the ground

A boat sailing on a stormy ocean

A burning campfire in a forest



Waves crashing against the shore



A ball thrown in the air

Figure D.13: Video sequences generated using LWM, showing various scenes.

D.8 Training Hyperparameters

Table D.7: LWM-Text Training Stages

	32K	**128K**	**256K**	**512K**	**1M**
Parameters	7B	7B	7B	7B	7B
Initialize From	LLaMA-2 7B	Text-32K	Text-128K	Text-256K	Text-512K
Precision	float32	float32	float32	float32	float32
Sequence Length	2^{15}	2^{17}	2^{18}	2^{19}	2^{20}
RoPE θ	1M	10M	10M	25M	50M
Tokens per Batch	4M	4M	4M	4M	4M
Total Tokens	4.8B	12B	12B	3B	1.8B
Total Steps	1200	3000	3000	720	450
LR Schedule	Constant	Constant	Constant	Constant	Constant
LR Warmup Steps	100	200	200	50	25
LR	4×10^{-5}	4×10^{-5}	4×10^{-5}	4×10^{-5}	4×10^{-5}
Compute (TPU)	v4-512	v4-512	v4-512	v4-512	v4-512
Mesh Sharding	1,-1,4,1	1,-1,8,1	1,-1,16,1	1,-1,16,2	1,-1,16,4

Table D.8: LWM-Text-Chat Training Details

	128K	**256K**	**512K**	**1M**
Parameters	7B	7B	7B	7B
Initialize From	Text-128K	Text-256K	Text-512K	Text-1M
Precision	float32	float32	float32	float32
Sequence Length	2^{17}	2^{18}	2^{19}	2^{20}
RoPE θ	10M	10M	25M	50M
Tokens per Batch	4M	4M	4M	4M
Total Tokens	1.2B	1.2B	1.2B	1.2B
Total Steps	300	300	300	300
LR Schedule	Constant	Constant	Constant	Constant
LR Warmup Steps	25	25	25	25
LR	4×10^{-5}	4×10^{-5}	4×10^{-5}	4×10^{-5}
Compute (TPU)	v4-512	v4-512	v4-512	v4-512
Mesh Sharding	1,-1,4,1	1,-1,8,1	1,-1,16,1	1,-1,16,2

Table D.9: LWM / LWM-Chat Training Stages

	1K	**8K**	**32K**	**128K**	**1M**
Parameters	7B	7B	7B	7B	7B
Initialize From	Text-1M	1K	8K	32K	128K
Precision	float32	float32	float32	float32	float32
Sequence Length	2^{10}	2^{13}	2^{15}	2^{17}	2^{20}
RoPE θ	50M	50M	50M	50M	50M
Tokens per Batch	8M	8M	8M	8M	8M
Total Tokens	363B	107B	10B	3.5B	0.4B
Total Steps	45000	14000	1200	450	50
LR Schedule	Cosine	Cosine	Cosine	Cosine	Cosine
LR Warmup Steps	1000	500	100	50	5
Max LR	6×10^{-4}	6×10^{-4}	8×10^{-5}	8×10^{-5}	8×10^{-5}
Min LR	6×10^{-5}	6×10^{-5}	8×10^{-5}	8×10^{-5}	8×10^{-5}
Compute (TPU)	v4-1024	v4-1024	v4-1024	v4-1024	v4-1024
Mesh Sharding	1,-1,1,1	1,-1,1,1	1.-1.4,1	1.-1.8,1	1,-1,16,4

Appendix E

Appendix For Chapter 6

E.1 Bootstrap Your Own Latent Details

Model Architecture. The BYOL model used in this work derives from the recurrent BYOL-Explore architecture proposed in [84], with a few modifications. Namely Guo et al. propose BYOL-Explore, a multi-step predictive latent world model. The BYOL-Explore architecture trains an online network using targets generated by an exponential moving average target network on future observations. Observations o_t are first encoded into an observation representation $f_\theta(o_t)$. Online predictions are computed by then encoding the observation representation using a closed-loop RNN $h^c_\theta(f_\theta(o_t))$. Guo et al. also pass actions into the closed loop RNN, but we wish to learn an action-free reward mechanism that can be trained solely from videos. At each timestep, the carry b_t from the closed loop RNN is then fed into an open-loop RNN h^o_θ which predicts open-loop representations K steps into the future: $(b_{t,k} \in \mathcal{R}^{\mathcal{M}})_{k=1}^{K-1}$, where $b_{t,k} = h^o_\theta(b_{t,k-1})$. The original BYOL-Explore architecture feeds actions a_{t+k+1} as inputs to the open loop cell h^o_θ as well, but we opt to omit these inputs for the same reason outlined above.

Given the deterministic RNN architecture used to predict online targets, omitting the action conditioning may lead to poor performance in multi-modal or stochastic environments. A more principled approach would utilize a probabilistic recurrent state space model [61, 89] to account for the multi-modality of futue states. We leave this approach to future work.

Finally, a predictor head $g_\theta(b_{t,k})$ outputs online predictions. We refer the reader to [84] for a figure of the outlined architecture.

The target network is an observation encoder f_ϕ whose parameters are an exponential moving average of f_θ. The loss function for the online network is then defined as:

$$\mathcal{L}_{\text{BYOL-Explore}}(\theta, t, k) = \left\| \frac{g_\theta(b_{t,k})}{\|g_\theta(b_{t,k})\|_2} - \texttt{sg}\left(\frac{f_\phi(o_{t+k})}{\|f_\phi(o_{t+k})\|_2}\right) \right\|_2^2,$$
$$\mathcal{L}_{\text{BYOL-Explore}}(\theta) = \frac{1}{B(T-1)} \sum_{t=0}^{T-2} \frac{1}{K(t)} \sum_{k=1}^{K(t)} \mathcal{L}_{\text{BYOL-Explore}}(\theta, t, k),$$

where $K(t) = \min(K, T-1-t)$ is the valid open-loop horizon for a trajectory of length T and $\texttt{sg}$ is the stop-gradient operator.

Computing Rewards. The uncertainty associated with the transition (o_t, a_t, o_{t+1}) is the sum of the corresponding prediction losses:

$$\ell_t = \sum_{p+q=t+1} \mathcal{L}_{\text{BYOL-Explore}}(\theta, j, p, q),$$

where $0 \leq p \leq T-2$, $1 \leq q \leq K$ and $0 \leq t \leq T-2$. This accumulates all the losses corresponding to the latent dynamics model uncertainties relative to the observation o_{t+1}. For our purposes, we calculate rewards as $r_t = -\ell_t$, which is equivalent to negating the reward used in the original BYOL-Explore formulation, and can be interpreted as negating the exploration objective to ensure that the agent stays close to the trajectory distribution found in the original dataset.

E.2 Video Model Hyperparameters

Table E.1: Hyperparameters and training details for all VQ-GAN models

	DMC	Atari	RLBench (single + multi task)
Input size	$64 \times 64 \times 3$	$64 \times 64 \times 3$	$64 \times 64 \times 3$
Latent size	8×8	16×16	16×16
β (commitment loss coefficient)	0.25	0.25	0.25
Batch size	128	128	128
Learning rate	10^{-4}	10^{-4}	10^{-4}
Learning rate schedule	constant	constant	constant
Training steps	200k	200k	200k
Base channels	128	128	128
Ch mult	[1, 1, 2, 2]	[1, 2, 2]	[1, 2, 2]
Num res blocks	1	1	2
Codebook size	1024	1024	1024
Codebook dimension	64	64	64
Perceptual loss weight	0.1	0.1	0.1
Disc base features	32	32	32
Disc gradient penalty weight	10^{8}	10^{8}	10^{8}
Disc max hidden feature size	512	512	512
Disc mbstd group size	4	4	4
Disc mbstd num features	1	1	1
Disc loss weight	0.1	0.1	0.1

Table E.2: Hyperparameters and training details for all VideoGPT and MaskGit models

	DMC	Atari	RLBench (single task)	RLBench (multi-task)
Sequence length	16	16	4	4
Frame skip	1	1	4	4
Latent size	8×8	16×16	16×16	16×16
Number of classes	16	8	n/a	34
Batch size	32	32	32	32
Learning rate	10^{-4}	10^{-4}	10^{-4}	10^{-4}
Learning rate schedule	constant	constant	constant	constant
Training steps	500k	500k	500k	500k
Adam (β_1, β_2)	(0.9, 0.999)	(0.9, 0.999)	(0.9, 0.999)	(0.9, 0.999)
Hidden dim	256	512	256	512
Feedforward dim	1024	2048	1024	2048
Number of heads	8	8	8	8
Number of layers	8	8	8	8
Dropout	0	0	0	0

E.3 Reinforcement Learning Hyperparameters

Table E.3: Hyperparameters and training details for DreamerV3

	DMC	Atari	RLBench (single task)	RLBench (multi-task)
General				
Replay Capacity (FIFO)	10^6	10^6	10^6	10^6
Start learning (prefill)	0	0	0	0
Batch Size	16	16	16	16
Batch length	64	64	64	64
MLP Size	2×512	4×1024	4×1024	4×1024
Activation	LayerNorm + swish			
World Model				
RSSM Size	512	4096	4096	4096
Number of Latents	32	32	32	32
Classes per Latent	32	32	32	32
KL Balancing	0.667	0.667	0.667	0.667
Actor Critic				
Imagination Horizon	15	15	15	15
Discount	0.995	0.995	0.995	0.995
Return Lambda	0.95	0.95	0.95	0.95
Target Update Interval	50	50	50	50
All Optimizers				
Gradient Clipping	100	100	100	100
Learning Rate	10^{-4}	10^{-4}	10^{-4}	10^{-4}
Adam epsilon	10^{-6}	10^{-6}	10^{-6}	10^{-6}
Plan2Explore				
Ensemble size	10	10	10	10

Table E.4: Hyperparameters and training details for DrQ

	DMC
Actor Critic	
MLP Size	256×256
CNN Features	$32 \times 64 \times 128 \times 256$
CNN Filters	$3 \times 3 \times 3 \times 3$
CNN Strides	$2 \times 2 \times 2 \times 2$
Learning Rate	$3e-4$
Discount	0.99
Number of Critics	2
τ	0.005
Init Temperature	0.1
Augmentations	Random crop
Random Network Distillation	
RND CNN Features	$32 \times 64 \times 64$
RND MLP Size	512×512
RND learning rate	$3e-4$
RND Exploration Weight	1

E.4 Environment Training Curves

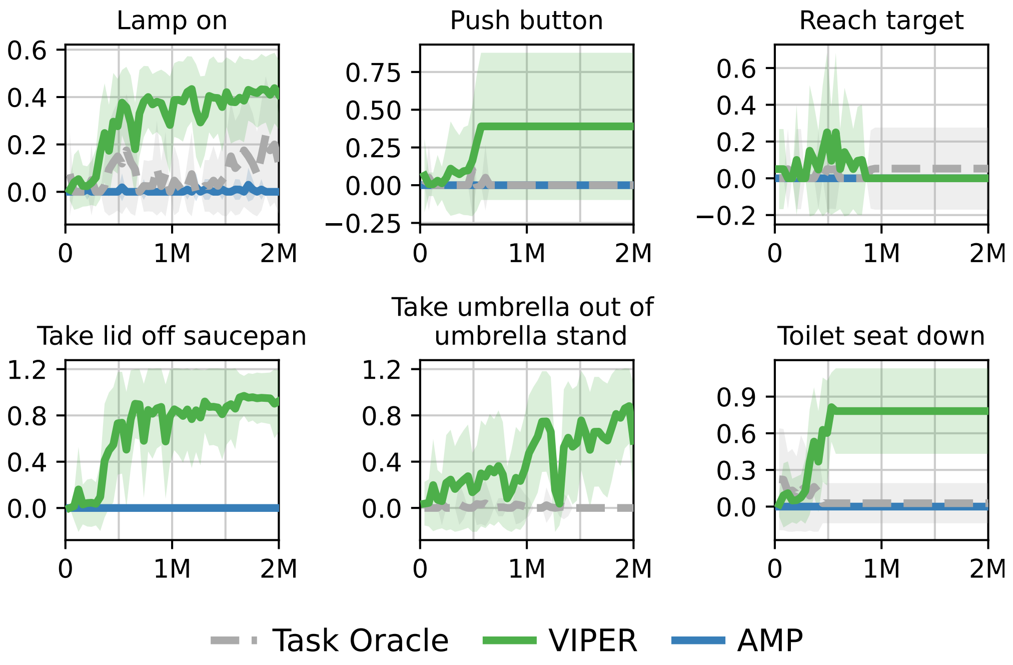


Figure E.1: Results across 6 Reinforcement Learning Benchmark tasks, with mean and standard deviation computed across 3 seeds.

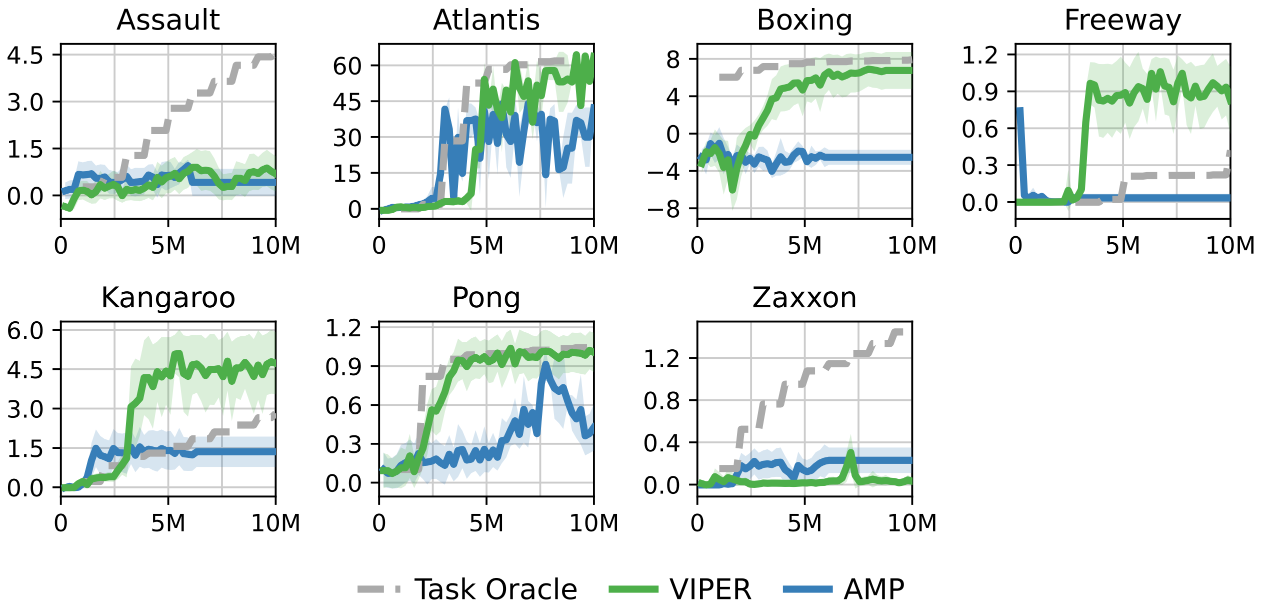


Figure E.2: Results across 7 Atari tasks. Scores computed using Human-Normalized Mean across 3 seeds.

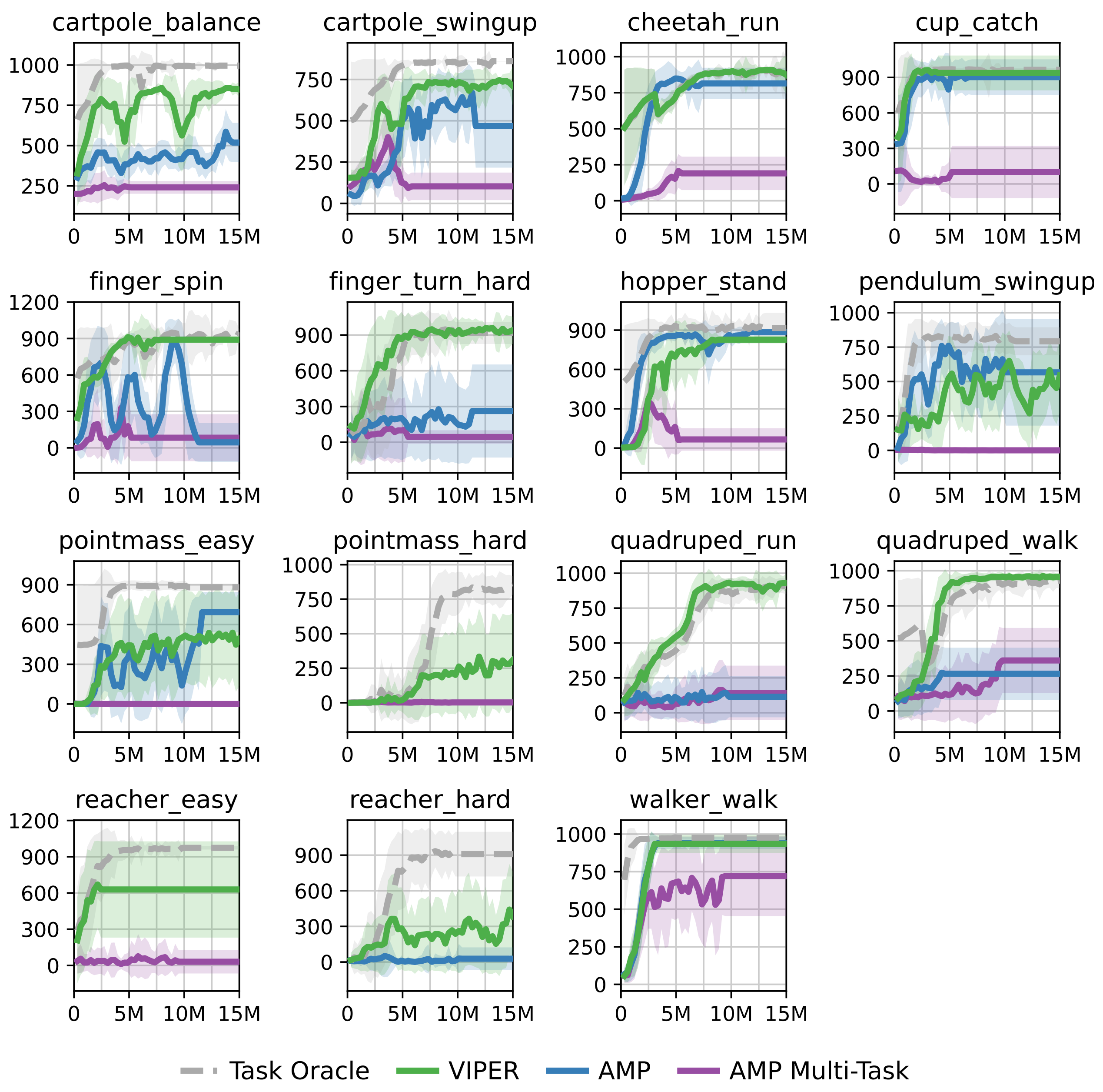


Figure E.3: Results across 15 DeepMind Control Suite tasks, with mean and standard deviation computed across 3 seeds. AMP runs were stopped early due to poor performance.

E.5 Training Environments

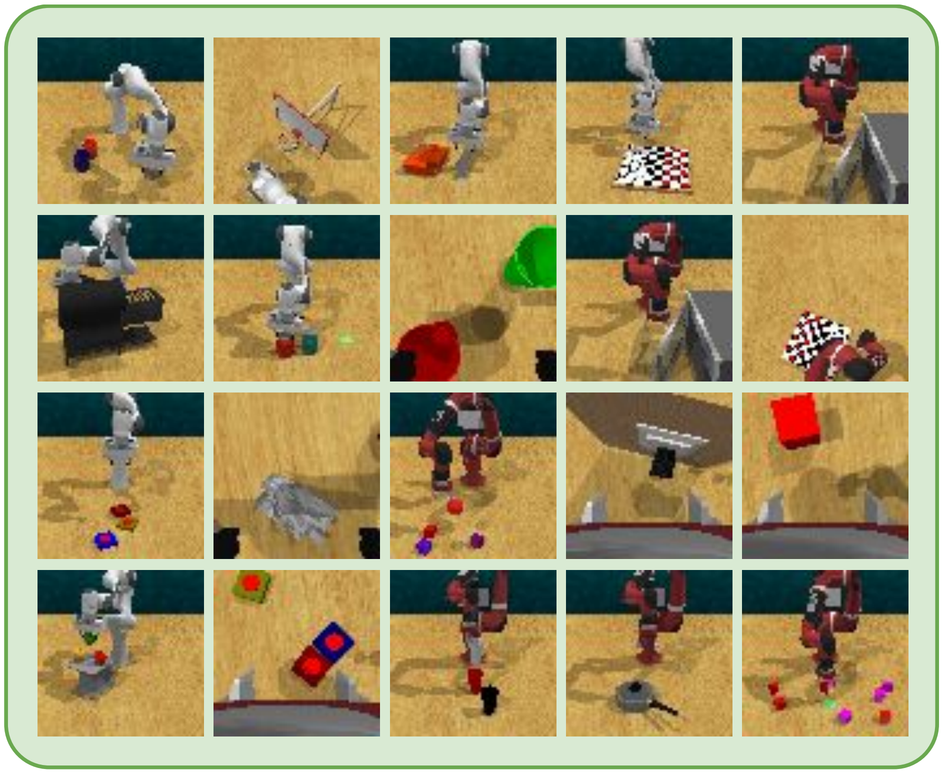


Figure E.4: A single autoregressive video model is trained on 30 tasks for the Franka Panda and 23 tasks for the ReThink Robotics Sawyer, using 16×16 VQCodes and a context length of 4, with frame skip 4.

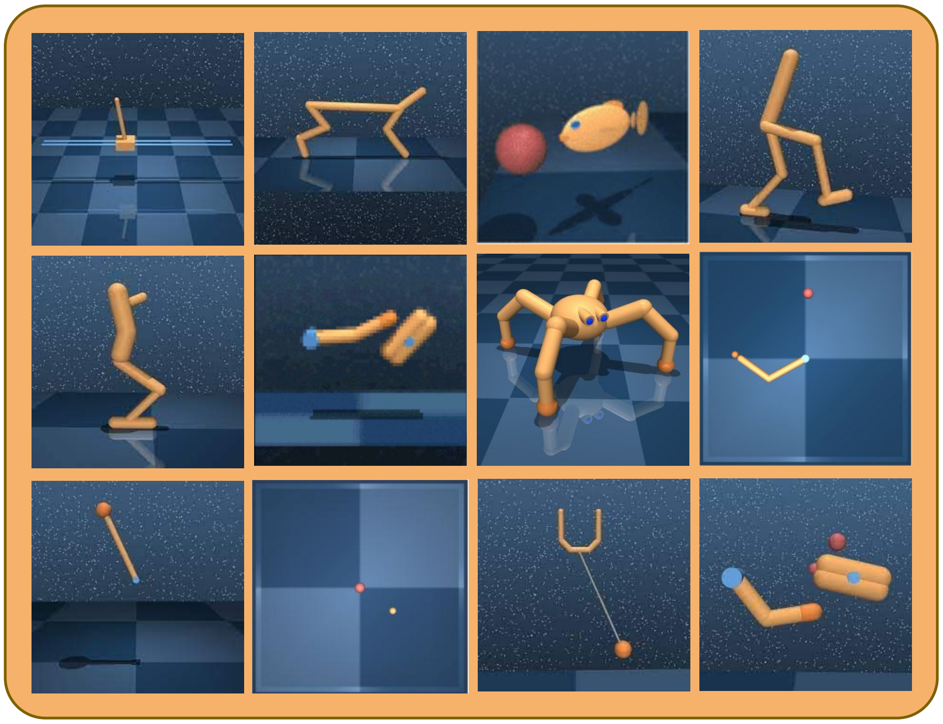


Figure E.5: A single task-conditioned autoregressive video model is trained on 15 DeepMind Control Tasks, using 8×8 VQCodes and a context length of 16.

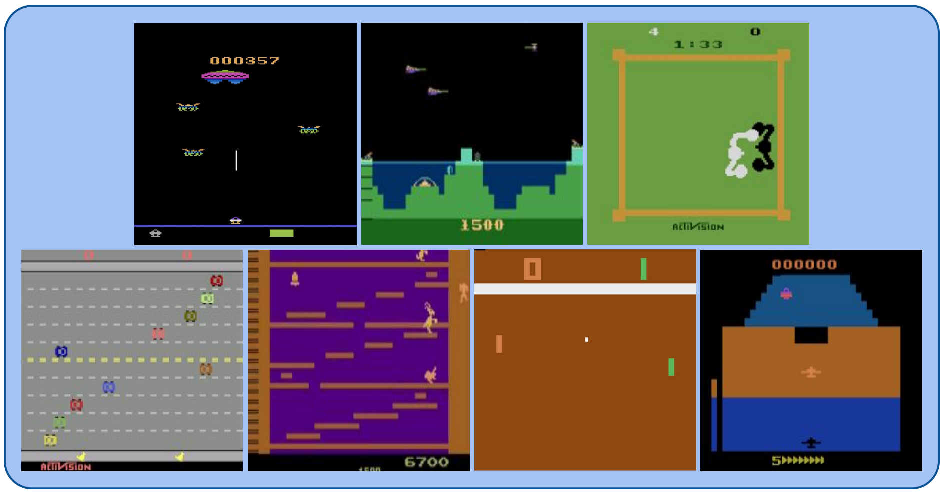
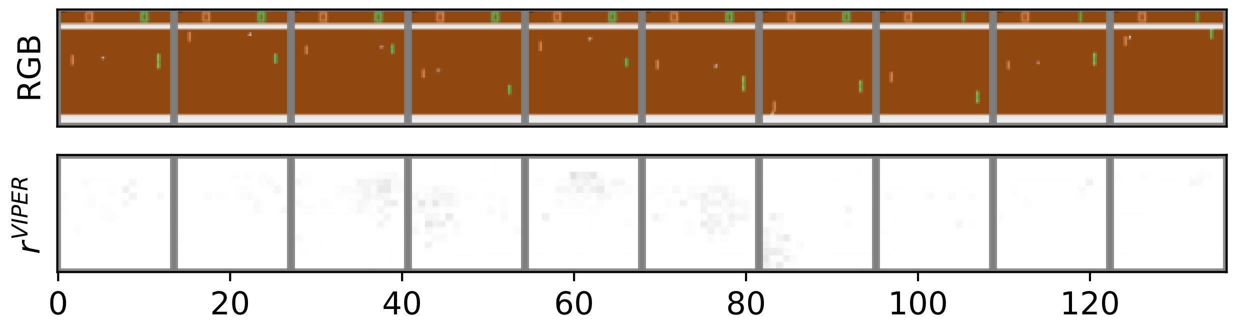


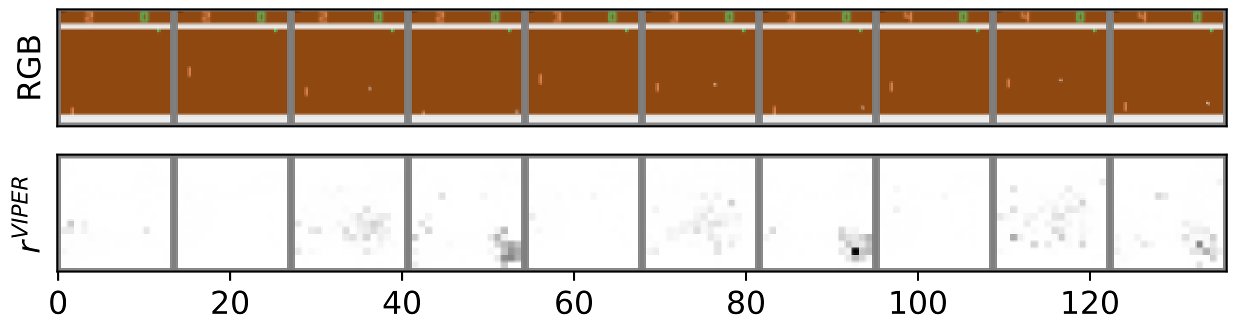
Figure E.6: A single autoregressive video model is trained on 7 Atari tasks, using 16×16 VQCodes and a context length of 16.

E.6 Visualizing Video Prediction Uncertainty

Video model uncertainty can be visualized by upsampling the conditional likelihoods over VQCodes. Since VQCodes tend to model local features, this provides a useful tool for analyzing which regions of the image the video model is uncertain about. We visualize video prediction uncertainty for three environments below:



(a) Reference Trajectory



(b) Random Trajectory

Figure E.7: Uncertainty visualized for a reference trajectory (top) and a random trajectory (bottom). Brighter values correspond to higher likelihoods. For the random trajectory, the video model assigns lower likelihoods to regions of the image containing the ball. This is especially true for random trajectories, where the video model, trained solely on expert trajectories, cannot accurately predict what happens when the agent plays sub-optimally.

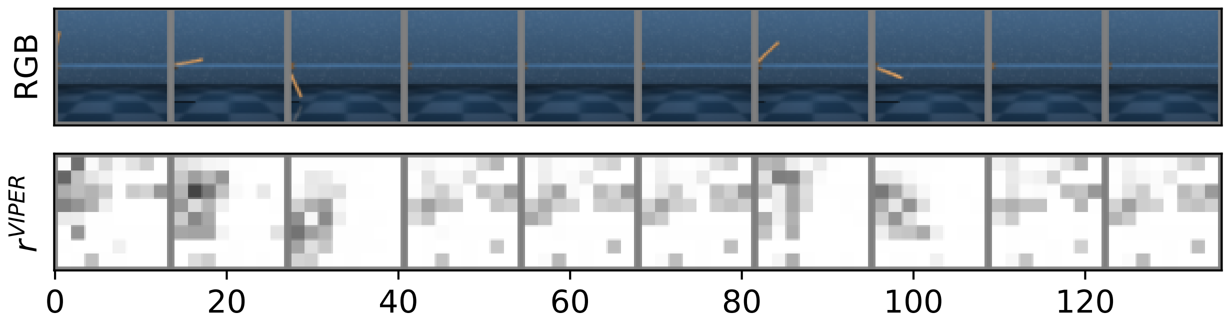
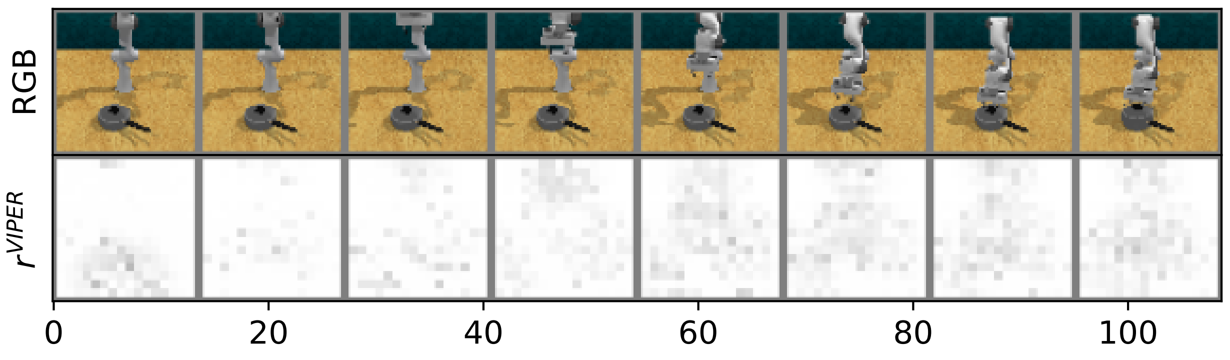
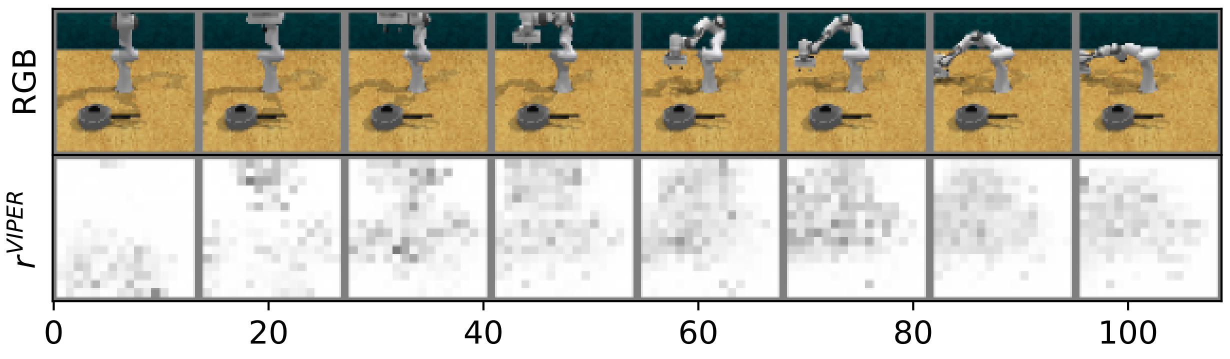


Figure E.8: Uncertainty visualized for a random trajectory from the dmc cartpole balance task. Brighter values correspond to higher likelihoods.



(a) Reference Trajectory



(b) Random Trajectory

Figure E.9: Uncertainty visualized for a reference trajectory (top) and a random trajectory (bottom). Brighter values correspond to higher likelihoods. Notice the high uncertainty over the position of objects on the table for the first frame. This corresponds to the unconditional likelihood (with no context). Since the position of the saucepan is randomized for every episode, the video model assigns some uncertainty to this initial configuration. For the random trajectory, there is high uncertainty assigned to the position of the arm since the video model has not seen trajectories that display poor behavior.

E.7 VIPER Out-of-distribution Analysis

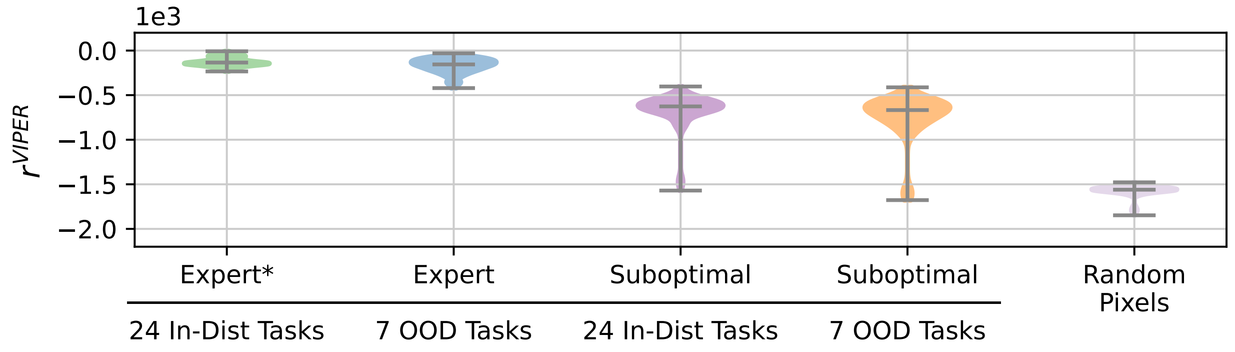


Figure E.10: Predicted r^{VIPER} on various RLBench tasks. We evalute 10 trajectories for each task. *Training data for the VIPER model.

Appendix F

Appendix For Chapter 7

F.1 Qualitative Examples

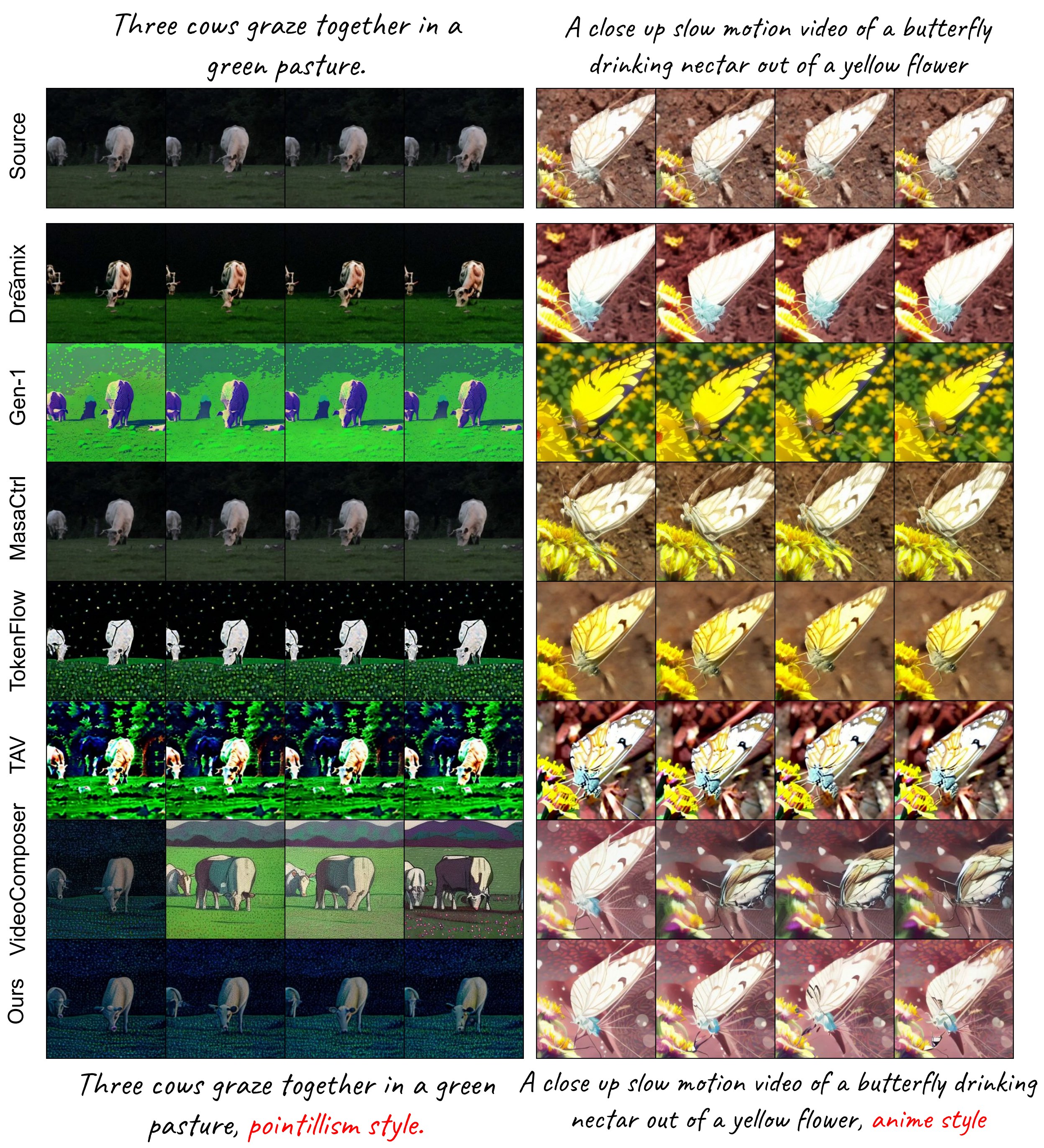

Three cows graze together in a green pasture.
A close up slow motion video of a butterfly drinking nectar out of a yellow flower
Source
Dreamix
Gen-1
MasaCtrl
TokenFlow
TAV
VideoComposer
Ours
Three cows graze together in a green pasture, pointillism style.
A close up slow motion video of a butterfly drinking nectar out of a yellow flower, anime style

Figure F.1: Comparisons for style video edit prompts

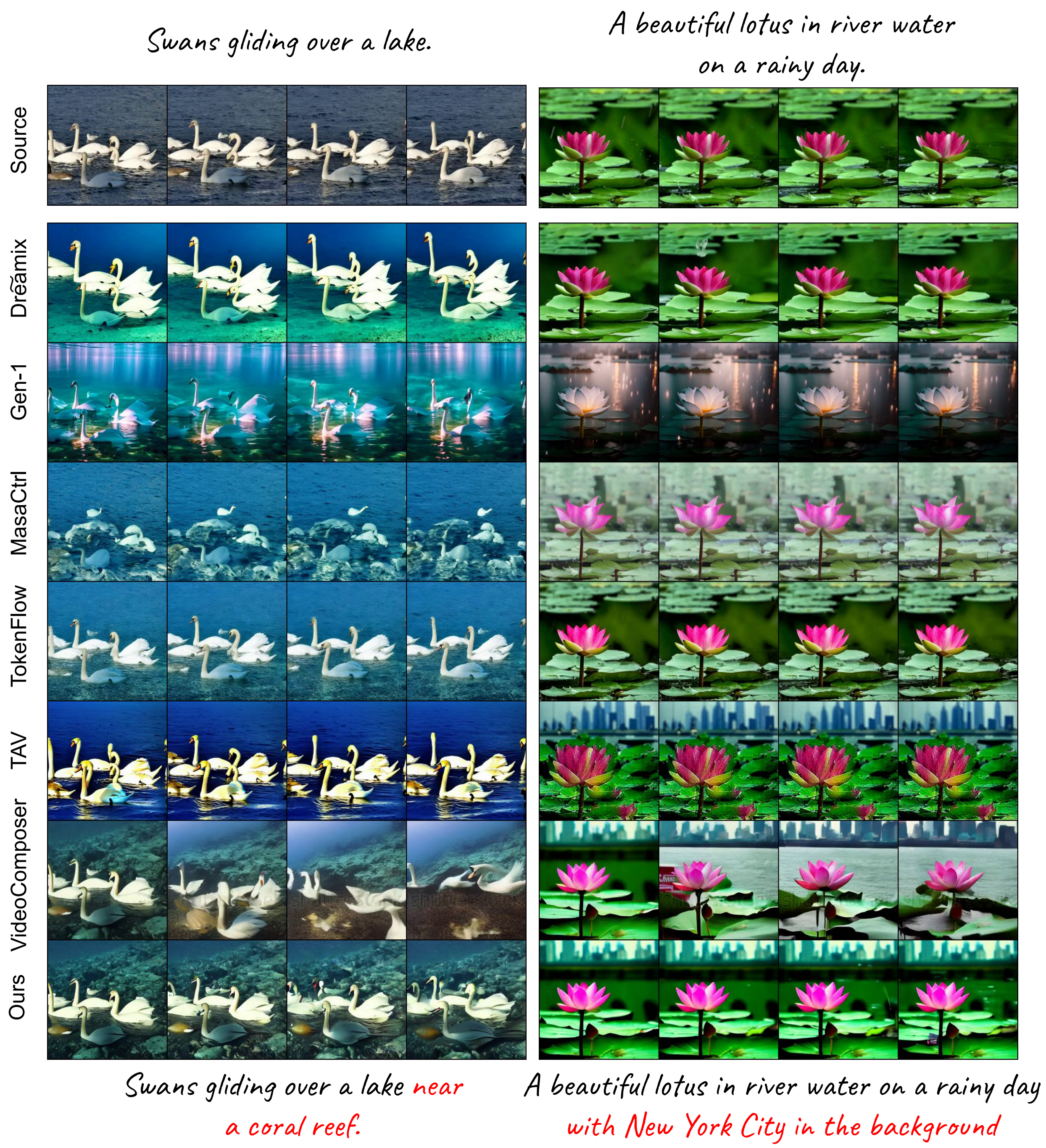

Swans gliding over a lake.
A beautiful lotus in river water on a rainy day.
Source
Dreamix
Gen-1
MasaCtrl
TokenFlow
TAV
VideoComposer
Ours
Swans gliding over a lake near a coral reef.
A beautiful lotus in river water on a rainy day with New York City in the background

Figure F.2: Comparisons for background video edit prompts

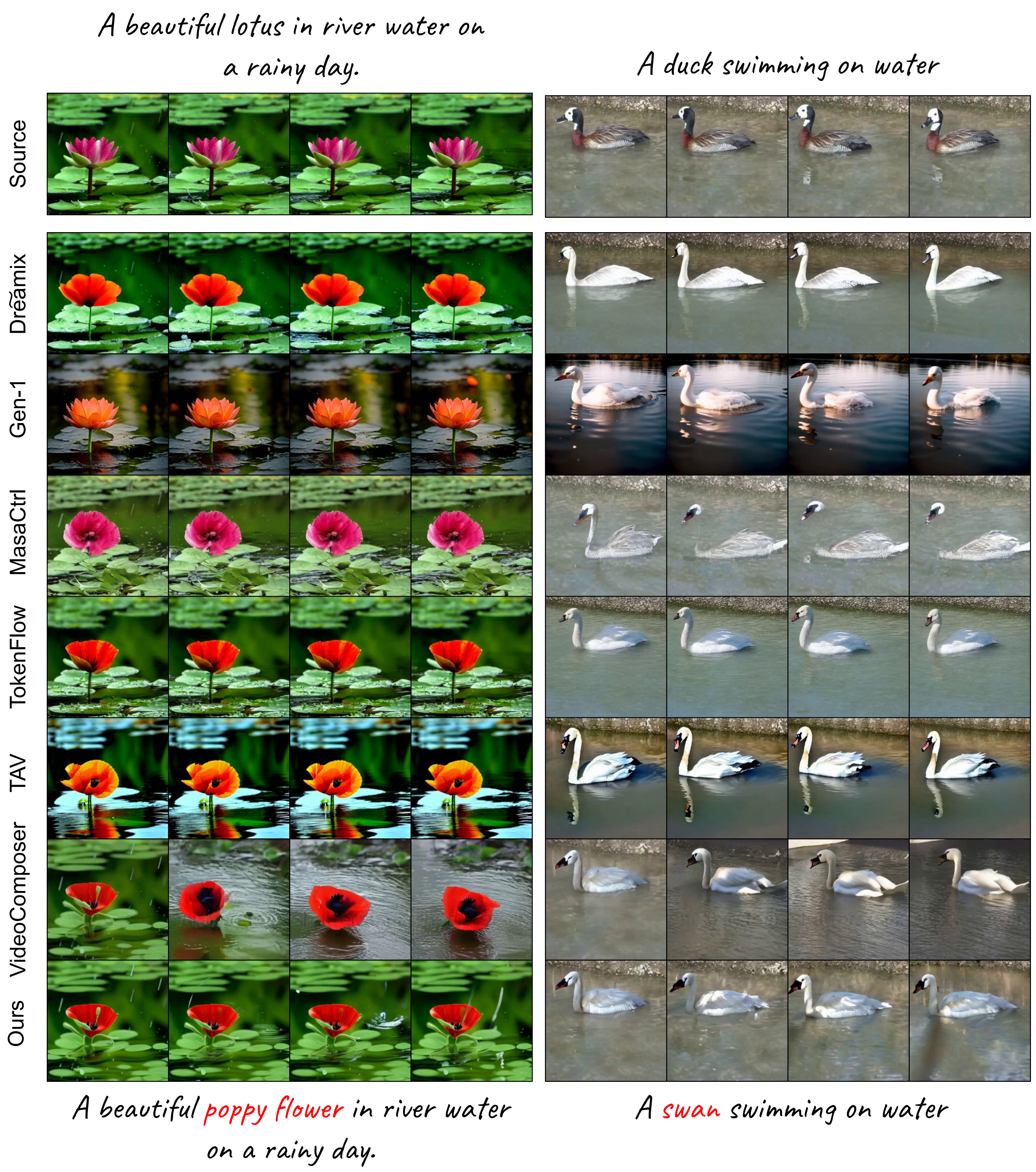

A beautiful lotus in river water on a rainy day.
A duck swimming on water
Source
Dreamix
Gen-1
MasaCtrl
TokenFlow
TAV
VideoComposer
Ours
A beautiful poppy flower in river water on a rainy day.
A swan swimming on water

Figure F.3: Comparisons for object video edit prompts

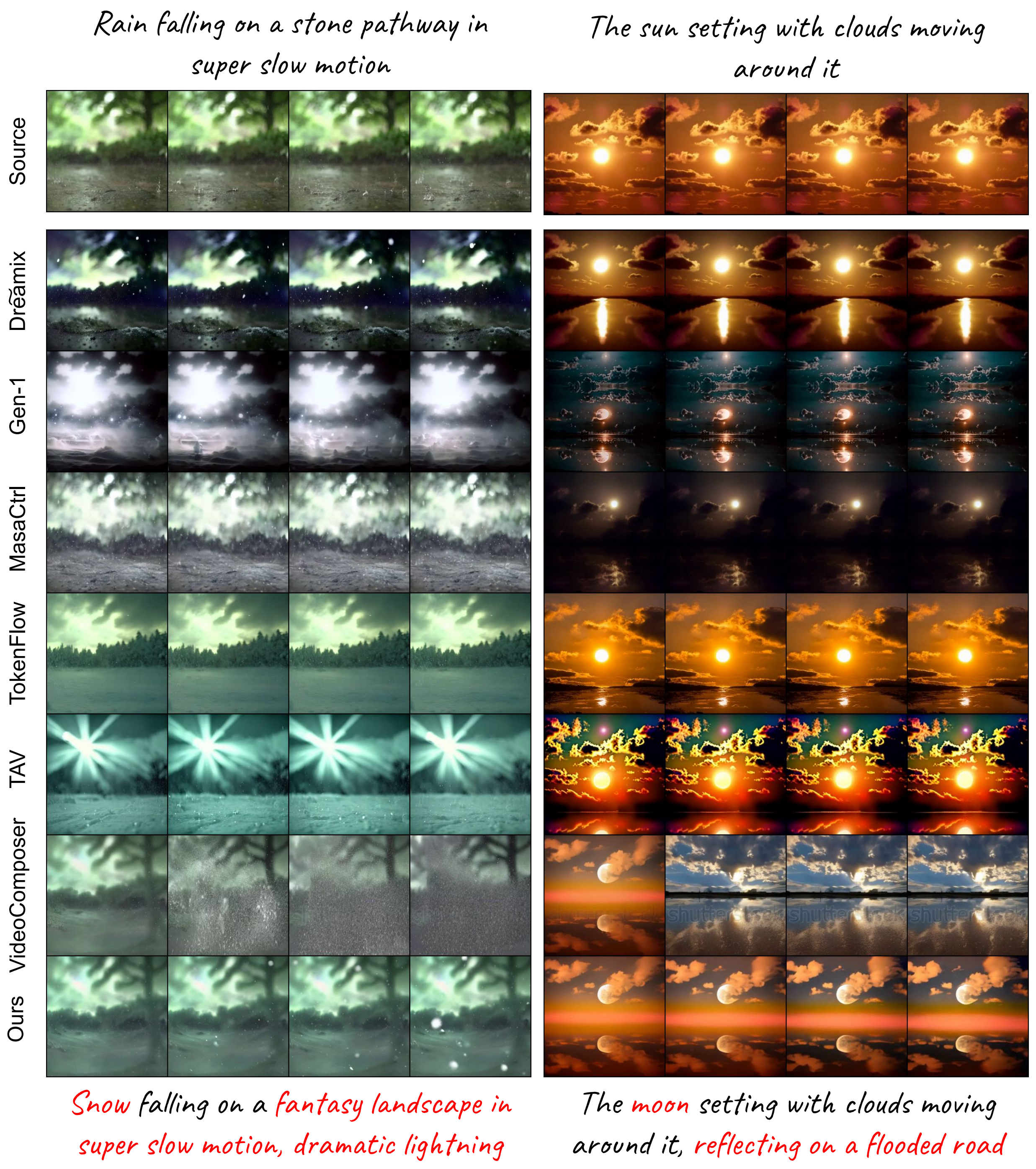

Rain falling on a stone pathway in super slow motion
The sun setting with clouds moving around it
Source
Dreamix
Gen-1
MasaCtrl
TokenFlow
TAV
VideoComposer
Ours
Snow falling on a fantasy landscape in super slow motion, dramatic lightning
The moon setting with clouds moving around it, reflecting on a flooded road

Figure F.4: Comparisons for multi-spatial video edit prompts

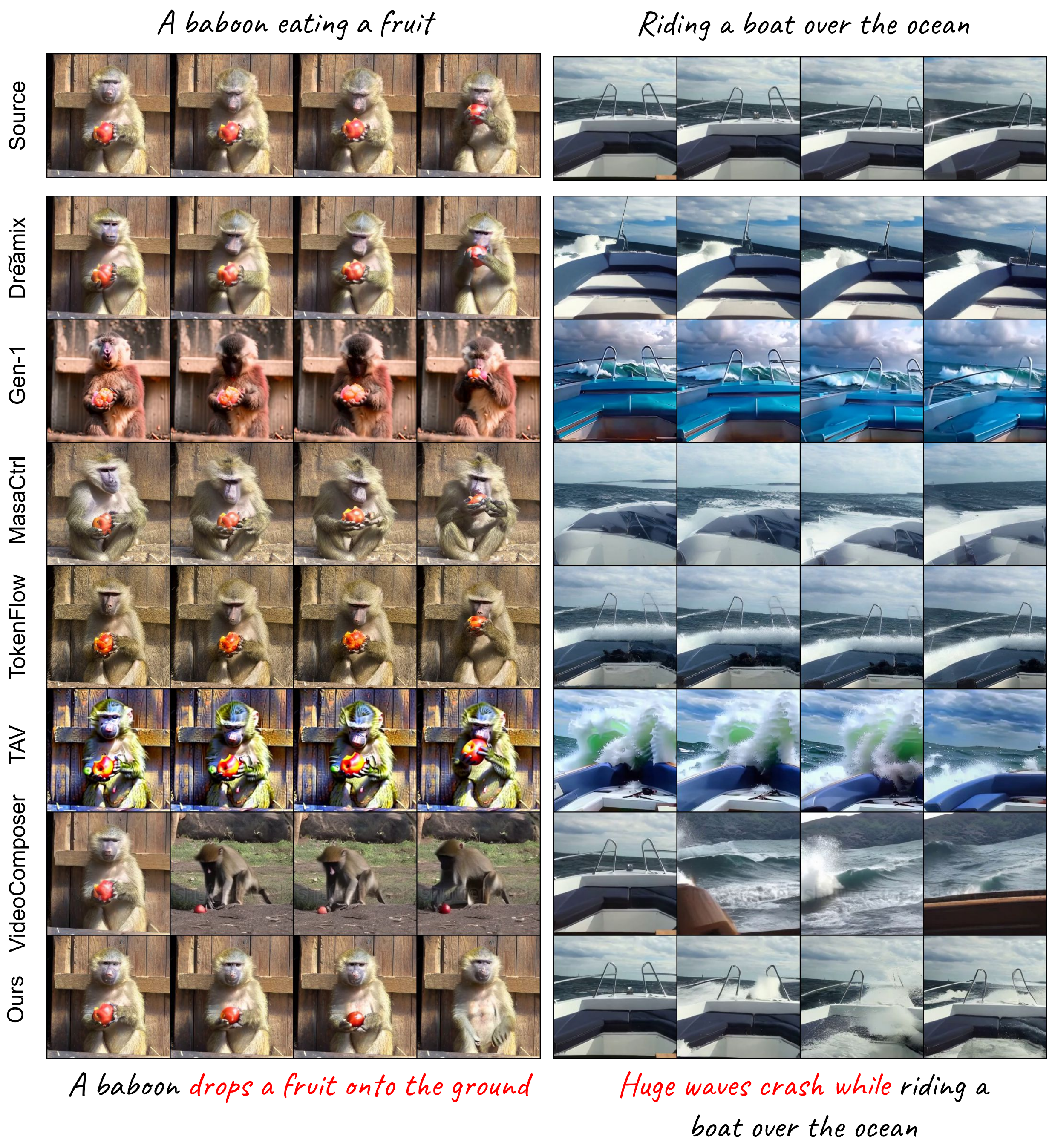

A baboon eating a fruit
Riding a boat over the ocean
Source
Dreamix
Gen-1
MasaCtrl
TokenFlow
TAV
VideoComposer
Ours
A baboon drops a fruit onto the ground
Huge waves crash while riding a boat over the ocean

Figure F.5: Comparisons for motion video edit prompts

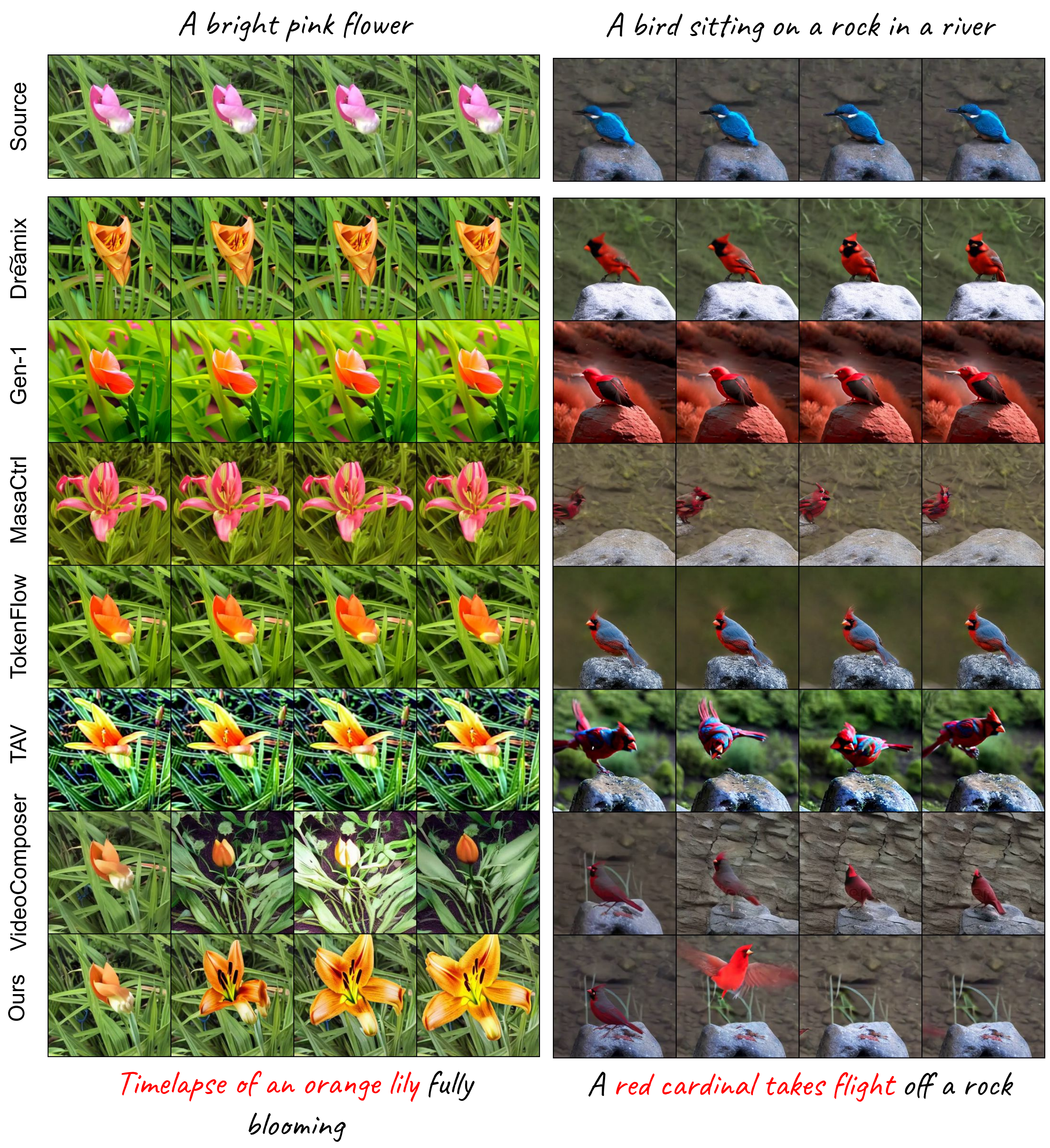

A bright pink flower
A bird sitting on a rock in a river
Source
Dreamix
Gen-1
MasaCtrl
TokenFlow
TAV
VideoComposer
Ours
Timelapse of an orange lily fully blooming
A red cardinal takes flight off a rock

Figure F.6: Comparisons for multi-motion video edit prompts

F.2 Automatic Evaluation Results

We provide a more detailed comparison on different video editing methods using the VideoCLIP-based M_{geo} metric in Table F.2. The results confirm the superiority of MoCA to other methods for all edit tasks. Additionally, we analyze the Spearman correlation [238] between automatic metrics introduced in Section 7.5 and human judgements. The results in Table F.1 suggest the VideoCLIP based M_{geo} and M_{dir} metrics as the most reliable automatic metric in evaluating the performance of video editing models.

		Style	**BG**	**Object**	**Motion**	**M-S**	**M-M**	**Total**
ImageCLIP	M_{sim}	-0.095	-0.102	-0.042	0.174	-0.076	-0.025	-0.006
	M_{dir}	0.238	0.290	0.161	0.035	0.219	0.141	0.150
	M_{geo}	0.240	0.288	0.156	0.047	0.209	0.137	0.152
VideoCLIP	M_{sim}	-0.080	-0.128	0.010	0.323	-0.080	0.006	0.036
	M_{dir}	0.290	0.328	0.183	0.049	0.202	0.189	0.189
	M_{geo}	0.300	0.304	0.189	0.140	0.201	0.205	0.203

Table F.1: **[238] correlation measuring the alignment between human judgements for editing quality and various CLIP-based metrics.** Among all six metrics in the table, M_{geo} and M_{dir} scores computed with a VideoCLIP model show the highest correlation with human ratings. Trends are similar to classification results shown in Table 7.3, with higher correlation in spatial edits, and lower for motion-based edits.

Method	Style	B-G	Object	Motion	M-S	M-M
MoCA	**0.331**	**0.375**	**0.370**	0.185	**0.349**	**0.334**
Dre͠amix	0.2223	0.304	0.356	0.141	0.290	0.321
Gen-1	0.254	0.317	0.295	0.146	0.309	0.209
MasaCtrl	0.225	0.253	0.295	0.154	0.270	0.283
Tune-a-Video	0.223	0.261	0.346	0.164	0.303	0.273
TokenFlow	0.206	0.239	0.314	0.0963	0.301	0.226
VideoComposer	0.259	0.328	0.326	**0.187**	0.301	0.202

Table F.2: **M_{geo} metric computed for each model based on VideoCLIP features, averaged over all edit examples per manipulation type.** This table presents a more detailed split of results shown in Table 7.4 by edit type.