Appendix: Video Probabilistic Diffusion Models in Projected Latent Space

Project page: https://sihyun.me/PVDM

A. Detailed description of training objective

In what follows, we describe the training objective that we used for our autoencoder consists of the encoder f_ϕ and the decoder g_ψ. We use the similar objective $\mathcal{L}(\phi, \psi)$ used in VQGAN [41] and TATS [12], except using vector-quantization regularization:

$$\mathcal{L}(\phi, \psi) := \mathcal{L}_{\texttt{pixel}}(\phi, \psi) + \lambda_1 \mathcal{L}_{\texttt{LPIPS}}(\phi, \psi) + \lambda_2 \max_h \mathcal{L}_{\texttt{GAN}}(\phi, \psi)$$

where $\mathcal{L}_{\texttt{pixel}}$, $\mathcal{L}_{\texttt{LPIPS}}$, $\mathcal{L}_{\texttt{GAN}}$, $\mathcal{L}_{\texttt{VQ}}(\phi, \psi)$ denote pixel-level ℓ_1-reconstruction loss, negative perceptual similarity [70], and adversarial objective [13] with a discriminator network h, respectively. Unlike prior works that uses KL-regularization [41] or VQ-regularization [12] to control the latent, we find simply adding $\tanh$ activation after projection network works well in training latent diffusion without without degradation autoencoder training.

For $\mathcal{L}_{\texttt{GAN}}$, we use the sum of hinge-loss and feature similarity between real samples and their reconstructions, following the objective used in TATS to train the autoencoder. Following the training setup in VQGAN, we choose $\lambda_1 = 1$ and $\lambda_3 = 0.25$ as a constant and set $\lambda_2 = 0$ before the other terms converge and $\lambda_2 = 0.25$ after the convergence. We find early stopping is crucial in training our autoencoder with $\mathcal{L}_{\texttt{GAN}}$; we stop the training by tracking the R-FVD values during training.

B. Detailed description of experiment setup

B.1. Datasets

UCF-101 [54] is a video dataset composed of diverse human actions. Specifically, the dataset contains 101 classes, and each video has a 320×240 resolution. There are 13,320 videos in total, where we used only the train split (9,357 videos) for training and the rest of the test split for the evaluation, following the common practice in evaluating unconditional video generation on UCF-101 [67]. All videos are center-cropped and resized to 256×256 resolution.

SkyTimelapse [64] is a collection of sky time-lapse total of 5,000 videos. We use the same data preprocessing following the official link.[3] Moreover, similar to UCF-101, all videos are center-cropped and resized to 256×256 resolution. We use the train split for both training the model and the evaluation, following the evaluation setup used in StyleGAN-V [47].

B.2. Metrics

For a fair comparison with prior works, we carefully choose the setups for quantitative evaluation. For measuring the Inception score (IS; [44]) on UCF-101 [54], we use the C3D network [56] trained on the Sports-1M dataset [25] and fine-tuned on UCF-101. Following the representative setups for evaluating IS for videos, also used in recent state-of-the-art methods [12, 67], we generate 10,000 samples to measure the score.

In the case of Fréchet video distance (FVD; [58]), we used the fixed protocol proposed by StyleGAN-V [47]. Unlike predominant setups that first preprocess the given video dataset as a fixed-length video clip and then sample real samples for calculating real statistics, the paper proposes first to sample the video data and randomly choose the fixed-length video clip. Such a different protocol is inspired by the observation from the previous evaluation procedure—if the dataset contains extremely long videos, the statistics can be easily biased due to the large number of clips from this video. We sample 2,048 samples (or the size of the real data if it is smaller) for calculating real statistics and 2,048 samples for evaluating fake statistics.

[3] https://github.com/weixiong-ur/mdgan

C. Detailed description of the baselines

In this section, we describe the main idea of video generation methods that we used for the evaluation.

- **VGAN** [62] replaces 2D convolutional networks to 3D convolutional networks to extend image generative adversarial network (GAN; [13]) architecture for video generation.
- **TGAN** [42] extends Wasserstein GAN [3] for images based on separating the motion and the content generator.
- **MoCoGAN** [57] decomposes motion and the content of the video for video generation via having a separate content generator and an autoregressive motion generator.
- **ProgressiveVGAN** [1] Inspired by GAN-based high-resolution image generation via progressive architecture design [26], ProgressiveGAN also generates the video progressively in both spatial and temporal directions.
- **LDVD-GAN** [23] mitigates computation inefficiency of video GANs with a discriminator using low-dimensional kernels.
- **VideoGPT** [65] is a two-stage model: it encodes videos as a sequence of discrete latent vectors using VQ-VAE [60] and learns the autoregressive model with these sequences via Transformer [61].
- **TGANv2** [43] proposes a computation-efficient video GAN based on designing submodules for a generator and a discriminator.
- **DVD-GAN** [7] uses two discriminators for identifying real/fake of each spatial and temporal dimension, where the input of the temporal discriminator is low-resolution for efficiency.
- **MoCoGAN-HD** [55] proposes to use a strong image generator for high-resolution image synthesis: they generate videos via modeling trajectories in the latent space of the generator.
- **DIGAN** [67] proposes a video GAN based on exploiting the concept of implicit neural representations and computation-efficient discriminators.
- **StyleGAN-V** [47] also introduces neural-representation-based video GAN to learn a given video distribution with a computation-efficient discriminator.
- **TATS** [12] proposes a new VQGAN [10] for videos and trains an autoregressive model to learn the latent distribution.
- **VDM** [21] extends image diffusion models by proposing a new U-Net architecture based on 3D convolutional layers.

D. More details on training setup

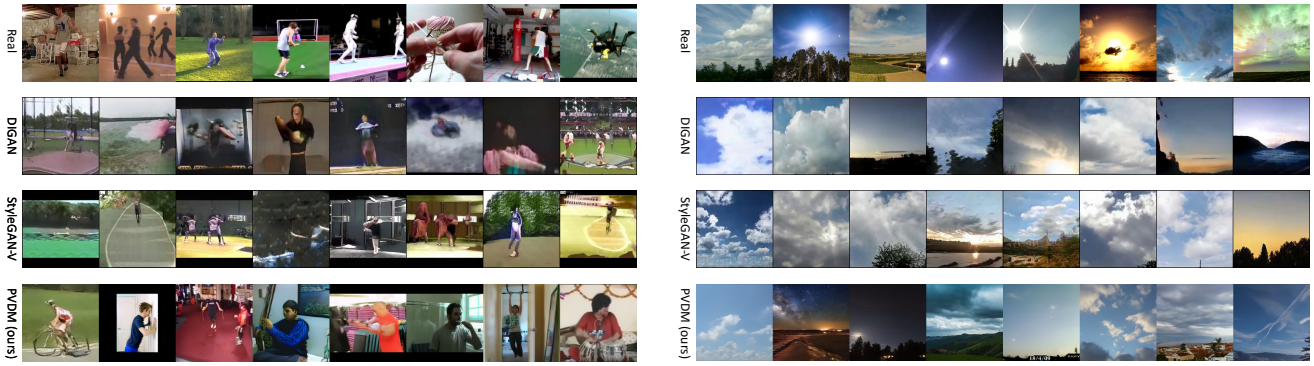
For all experiments, we use a batch size of 24 and a learning rate $1e$-4 for training autoencoders. We train the model until both FVD and PSNR converge. We use 4-layer Transformer for 3D-to-2D projections, where the number of heads is set to 4, the hidden dimension is set to 384, and the hidden dimension of the multilayer perceptron (MLP) in Transformer is to be 512. We set the dimension of the latent codebook to 4. For training diffusion models, we set a learning rate as $1e$-4 with a batch size of 64. For more hyperparameters, such as the base channel and the depth of U-Net architecture, we adopt a similar setup to LDMs [41]. In particular, we set the codebook channel C to 4 and the patch size to $4 \times 4 \times 1$, so that a $256 \times 256 \times 16 \times 3$ dimension video is encoded to a $(32 \times 32 + 32 \times 16 + 32 \times 16) \times 4 = 8{,}192$ dimension vector. Throughout the experiments, we consider two model configurations: PVDM-S (small model) and PVDM-L (large model), where we summarize the detailed configurations and hyperparameters in Table 6.

	Base channel	Attn. res.	# ResBlock	Channel mul.	# heads	Linear start	Linear end	Timesteps	# Iter.
PVDM-S	128	[4,2,1]	2	[1,2,4]	8	0.0015	0.0195	1000	400k
PVDM-L	256	[4,2,1]	2	[1,2,4]	8	0.0015	0.0195	1000	850k

Table 6. Model configurations for PVDM-S and PVDM-L.

E. Qualitative comparison of generated results

We provide the qualitative comparison of the first frame generated from baselines in Figure 6; note that we provide the comparison of videos on our anonymized website. Compared with other recent video synthesis methods, our PVDM shows overall high-quality synthesis results, especially on the complex dataset, UCF-101.



(a) UCF-101 (b) SkyTimelapse

Figure 6. Illustrations of the synthesized first frame results of PVDM and baselines trained on UCF-101 and SkyTimelapse datasets. Baseline results are from the StyleGAN-V website: https://universome.github.io/stylegan-v.

Table 7. Ablation study of our projected autoencoder.

Backbone	Proj.	VQ.	dim($\mathbf{z}$)	$\mathbf{z}$ shape	PSNR ↑	R-FVD ↓
2D CNN	-	✓	32,768	3D	25.23	147.5
2D CNN	-	✓	8,192	3D	21.89	559.9
Timesformer	-	✓	8,192	3D	24.65	134.9
Timesformer	✓	✓	8,192	2D	24.73	63.34
Timesformer	✓	-	8,192	2D	26.99	32.26

Table 8. Autoencoder performance (R-FVD) with different model sizes. # Enc. and # Dec. denote the number of parameters for encoder and decoder, respectively.

# Enc.+# Dec.	R-FVD ↓
11M+10M	154.7
23M+17M	67.76
35M+27M	63.34

F. More analysis in our autoencoder architecture

Ablation study on our autoencoder. We also conduct an ablation study of autoencoder on UCF-101 and report it in Table 7 (we adjust the model sizes to be equal): Our autoencoder components are particularly effective in capturing perceptual details with "2D-shaped" latents $\mathbf{z}$, as shown by better R-FVD than other variants. We also find the performance of 2D autoencoder degrades a lot for small $\dim(\mathbf{z})$ (8,192) as 2D AE has no temporal layers and thus is hard to capture the temporal coherence with compact latents. We also find removing popular vector quantization used for training autoencoders is not necessary in improving our autoencoder quality and does not prevents diffusion model training; it was enough to clip the range of latents by adding activation functions after the encoder ouputs (e.g., `tanh`).

Autoencoder performance with various model sizes. Table 8 summarizes the autoencoder performance w.r.t. varying model sizes. It shows our autoencoder is scalable and it works even fairly well using $\approx 2/3$ number of parameters.

G. Discussion with concurrent work

The concurrent work, MagicVideo [71], is also a latent video diffusion model. However, unlike our PVDM, they encode videos in a frame-wise manner, resulting in an unfavorable increase of latent dimension with length and thus limiting the scalability. Another concurrent work, LVDM [17] uses 3D CNN to encode video clip and design a latent diffusion model, but they still deal with cubic-shaped latent tensors in designing diffusion models. Make-a-video [46] extends a popular image diffusion model for video synthesis. They also avoid using 3D convolutions, yet they deal with 3D RGB arrays directly and are still difficult to be trained on high-resolution, long videos. As a result, it leads the framework to heavily rely on complex spatiotemporal interpolation modules to generate high-dimensional videos.

H. Limitations and future work

While PVDM shows promise in efficient and effective video synthesis, there are many possible future directions for further improvement as there still exists a gap between the real and the generated videos. For instance, designing diffusion model architectures specialized for our triplane latents to model latent distribution better or investigating better latent structures than our triplane idea to encode video more efficiently would be interesting directions. While we have not conducted experiments in large-scale video datasets to perform challenging text-to-video generation due to the limited resources, we strongly believe our framework also works well in this task and leave it for the future work.