Video Probabilistic Diffusion Models in Projected Latent Space

Sihyun Yu[1], Kihyuk Sohn[2], Subin Kim[1], Jinwoo Shin[1]
[1]Korea Advanced Institute of Science and Technology (KAIST)
[2]Google Research

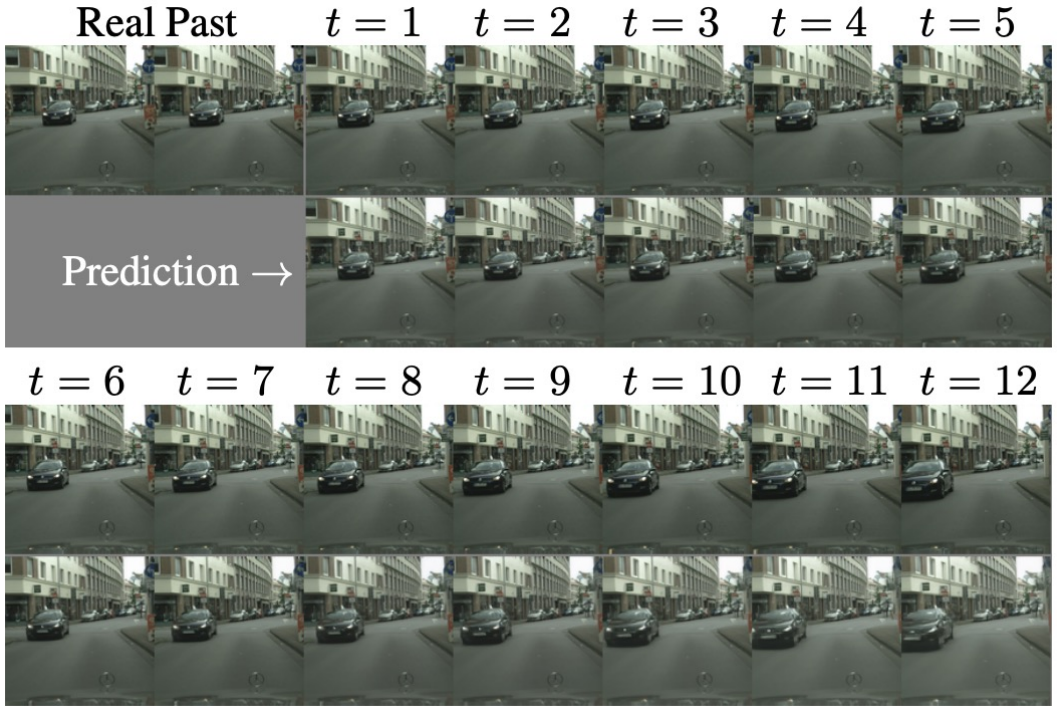
KAIST Google Research

JUNE 18-22, 2023
CVPR
VANCOUVER, CANADA

Introduction: Video Diffusion Models

Video diffusion models have recently shown great potential, yet they are difficult to be scaled-up

- **Challenge 1.** It requires significant memory and computation [Ho et al., 2022]
- **Challenge 2.** It often requires tremendous computation costs for generating longer videos [Voleti et al., 2022]



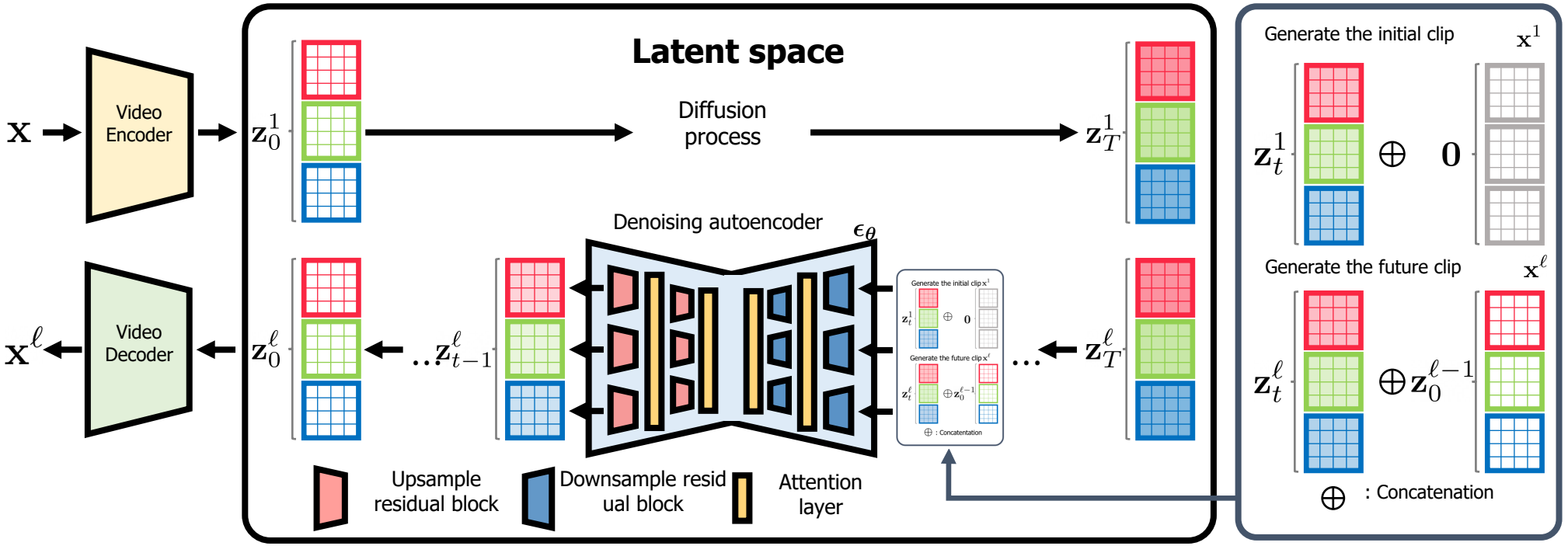
[Ho et al., 2022] Video Diffusion Models, NeurIPS 2022.
[Voleti et al., 2022] MCVD- Masked Condifional Video Diffusion Model for Prediction, Generation, and Interpolation, NeurIPS 2022.

Motivation: Efficient Video Diffusion Models?

🤔 Can we enjoy the power of diffusion models in an efficient manner for video generation?

💡 **Idea:** Encode videos as succinct, non-3D latents and train the diffusion model in this latent space!

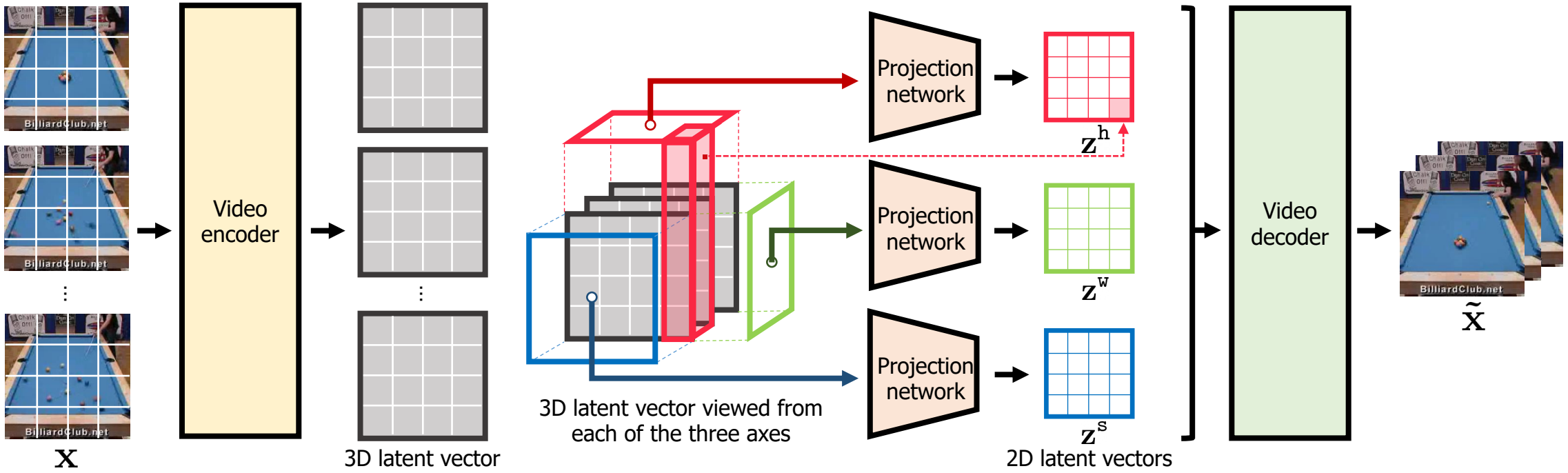
- (Autoencoder) We propose a new autoencoder of a mapping of **video → triplane latents**
- (Diffusion models) We propose a video diffusion model architecture without using 3D convolutions



Method: Autoencoder

Autoencoder: Factorizes a given video (cubic tensor) into three 2D "image-like" latent vectors

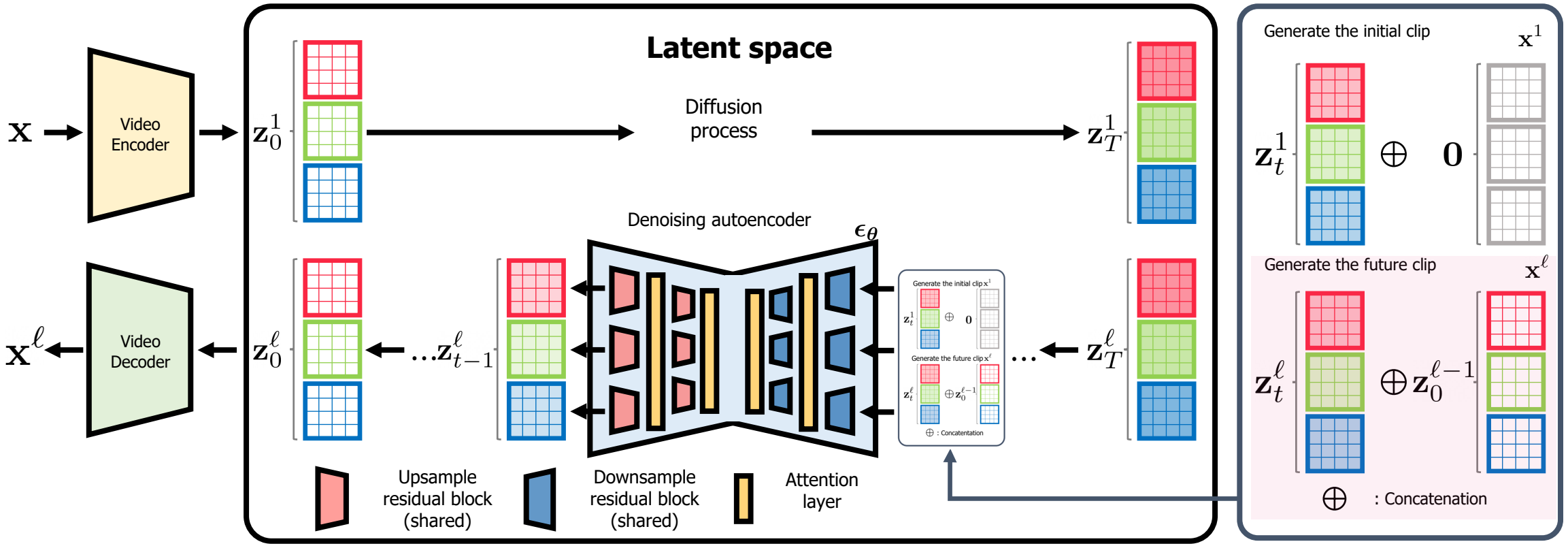
- $\mathbf{z}^s$ encodes common contents in videos (e.g., background), and the other two vectors encode the motion of videos
- Since we avoid using cubic latent vectors, it enables compute-efficient diffusion model design architecture



Method: Diffusion Model Architecture

Diffusion model: Based on popular 2D convolutional U-Net architectures used for images

- It does not require any 3D convolutional layer, hence much more compute-efficient
- Conditioned on the last video "clip", our model generates the next "clip" for generating longer videos



Method: Diffusion Model Training Objective

Objective: Joint training of unconditional (initial clip) and conditional (future clips) generation

- For unconditional modeling, the condition is replaced with 0.
- Trained with the sum of below two denoising objectives:

$$\mathbb{E}_{(\mathbf{x}_0^1,\mathbf{x}_0^2),\boldsymbol{\epsilon},t}\Big[\lambda||\boldsymbol{\epsilon}-\boldsymbol{\epsilon}_{\boldsymbol{\theta}}(\mathbf{z}_t^2,\mathbf{z}_0^1,t)||_2^2+(1-\lambda)||\boldsymbol{\epsilon}-\boldsymbol{\epsilon}_{\boldsymbol{\theta}}(\mathbf{z}_t^2,\mathbf{0},t)||_2^2\Big],$$

Conditional term

Unconditional term

Method: Sampling Procedure

Sampling: Autoregressively generate video "clips" (rather than the video frame one-by-one)

- Initial clip: Sampled from a prior Gaussian distribution only
- Future clips: Conditioned with a given condition (prior clip)

Algorithm 1 projected latent video diffusion model (PVDM)

for $\ell = 1$ to L **do** ▷ *Iteratively generate the video clips* $\mathbf{x}^\ell$.
  Sample the random noise $\mathbf{z}_T^\ell \sim p(\mathbf{z}_T)$.
  for $t = T$ to 1 **do**
    if $\ell = 1$ **then**
      Compute the unconditional score $\boldsymbol{\epsilon}_t = \boldsymbol{\epsilon}_{\boldsymbol{\theta}}(\mathbf{z}_t^\ell, \mathbf{0}, t)$.
    else
      Compute the conditional score $\boldsymbol{\epsilon}_t = \boldsymbol{\epsilon}_{\boldsymbol{\theta}}(\mathbf{z}_t^\ell, \mathbf{z}_0^{\ell-1}, t)$.
    end if
    Sample $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}_\mathbf{z}, \mathbf{I}_\mathbf{z})$ and compute $\mathbf{z}_{t-1}^\ell = \frac{1}{\sqrt{1-\beta_t}}\left(\mathbf{z}_t^\ell - \frac{\beta_t}{\sqrt{1-\bar{\alpha}_t}}\boldsymbol{\epsilon}_t\right) + \sigma_t\boldsymbol{\epsilon}$.
  end for
  Decode the ℓ-th clip $\mathbf{x}^\ell = g_\psi(\mathbf{z}_0^\ell)$.
end for
Output the generated video $[\mathbf{x}^1, \ldots, \mathbf{x}^L]$.

Experiment: Quantitative Results

Quantitative result: Outperforms prior video generation methods in popular benchmarks

- Both on FVD (short/long video clips) and IS

Table 1. FVD_{16} and FVD_{128} values (lower values are better) of video generation models on UCF-101 and SkyTimelapse. Bolds indicate the best results, and we mark our method as blue. We report FVD values of other baselines obtained by the reference (StyleGAN-V [47]). N/M-s denotes the model is evaluated with the DDIM sampler [51] with N steps (for the initial clip) and M steps (for future clips).

Method	UCF-101		SkyTimelapse	
	$FVD_{16}\downarrow$	$FVD_{128}\downarrow$	$FVD_{16}\downarrow$	$FVD_{128}\downarrow$
VideoGPT [65]	2880.6	N/A	222.7	N/A
MoCoGAN [57]	2886.8	3679.0	206.6	575.9
+ StyleGAN2 [28]	1821.4	2311.3	85.88	272.8
MoCoGAN-HD [55]	1729.6	2606.5	164.1	878.1
DIGAN [67]	1630.2	2293.7	83.11	196.7
StyleGAN-V [47]	1431.0	1773.4	79.52	197.0
PVDM-S (ours); 100/20-s	457.4	902.2	71.46	159.9
PVDM-L (ours); 200/200-s	398.9	**639.7**	61.70	137.2
PVDM-L (ours); 400/400-s	**343.6**	648.4	**55.41**	**125.2**

Table 2. IS values (higher values are better) of video generation models on UCF-101. Bolds indicate the best results and subscripts denote the standard deviations. * denotes the model is trained on train+test split, otherwise the method uses only the train split for training.

Method	IS ↑
MoCoGAN [57]	12.42±0.07
ProgressiveVGAN [1]	14.56±0.05
LDVD-GAN [23]	22.91±0.19
VideoGPT [65]	24.69±0.30
TGANv2 [43]	28.87±0.67
StyleGAN-V* [47]	23.94±0.73
DIGAN [67]	29.71±0.53
VDM* [21]	57.00±0.62
TATS [12]	57.63±0.24
PVDM-L (ours)	**74.40**±**1.25**

Experiment: Qualitative Comparison with Baselines

Qualitative result: Shows more realistic synthesis results

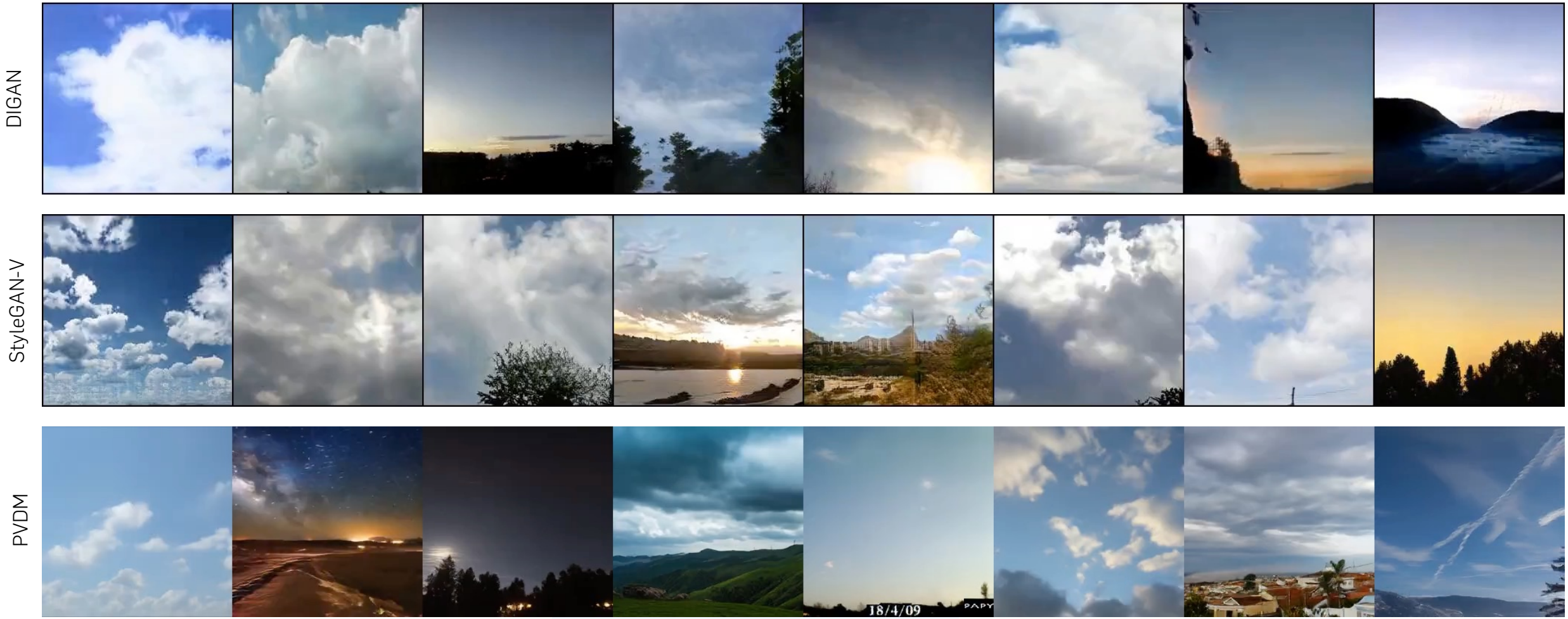
DIGAN

StyleGAN-V

PVDM

Experiment: Qualitative Comparison with Baselines

Qualitative result: **Shows more realistic synthesis results**



Analysis: Efficiency

Compared with VDM, our method shows strong memory-/compute- efficiency

- e.g.) PVDM can achieve 17.6x better compute-efficiency on inference than VDM (with the same sampling setup)

Table 5. Maximum batch size for training and time (s), memory (GB) for synthesizing a 256×256 resolution video measured with a single NVIDIA 3090Ti 24GB GPU. N/A denotes the values cannot be measured due to the out-of-memory problem. N/M-s denotes the model is evaluated with the DDIM sampler [51] with N steps (for the initial clip) and M steps (for future clips).

	Train	Inference (time/memory)	
Length →	16	16	128
TATS [12]	0	84.8/18.7	434/19.2
VideoGPT [65]	0	139/15.2	N/A
VDM [21]; 100/20-s	0	113/11.1	N/A
PVDM-L (ours); 200/200-s	2	20.4/5.22	166/5.22
PVDM-L (ours); 400/400-s	2	40.9/5.22	328/5.22
PVDM-S (ours); 100/20-s	**7**	**7.88/4.33**	**31.3/4.33**

Analysis: Encoding Quality

Our autoencoder architecture is beneficial in encoding videos as compact latents

- Compared with popular 2N CNN or Video Transformers, our model shows much better quality

Table 4. Quantitative evaluation results between reconstruction from the autoencoder of PVDM and the real videos.

	UCF-101		SkyTimelapse	
	Train	Test	Train	Test
R-FVD ↓	25.87	32.26	7.37	36.52*
PSNR ↑	27.34	26.99	34.33	32.68*

∗ Evaluated with 196 samples due to the smaller test set size (196) than 2,048.

Table 7. Ablation study of our projected autoencoder.

Backbone	Proj.	VQ.	dim(**z**)	**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

Summary: PVDM

Summary: We propose latent video diffusion models for scalable video synthesis

We propose PVDM = Projected-latent Video Diffusion Models

1. Achieves state-of-the-art performance on various video generation benchmarks
2. Can generate long videos of high-resolution frames without demanding recourses
3. Shows strong memory-/compute-efficiency on both training/inference

Future possible directions (but not limited to):

1. Diffusion model architecture more specialized for our triplane latent space
2. Better structural latent space for representing videos
3. Applications – e.g., video inpainting
4. Method to be combined with “image” foundation models (e.g., stable-diffusion)