Efficiency-optimized Video Diffusion Models

Zijun Deng
Wangxuan Institute of Computer Technology & National Key Laboratory for Multimedia Information Processing, Peking University
Beijing, China
dengzijun@stu.pku.edu.cn

Xiangteng He
Wangxuan Institute of Computer Technology & National Key Laboratory for Multimedia Information Processing, Peking University
Beijing, China
hexiangteng@pku.edu.cn

Yuxin Peng*
Wangxuan Institute of Computer Technology & National Key Laboratory for Multimedia Information Processing, Peking University
Beijing, China
pengyuxin@pku.edu.cn

ABSTRACT

Video diffusion models have recently shown strong capability in synthesizing high-fidelity videos in various ways, including prediction, interpolation, and unconditional generation. However, their synthesis ability credits a lot to leveraging large denoising models to reverse the long noise-adding process, which also brings extremely expansive sampling and training costs. After examing the source of the computation cost, we confirm that the main calculation comes from the redundancy of the convolution. To address this issue, we propose Efficiency-optimized Video Diffusion Models to reduce the network's computation cost by minimizing the input and output channels of the convolution. First, a **bottleneck residual pathway** is proposed to conduct a channel-wise downsample to the convolution pathways, which extracts crucial information from the input and reduces computation cost. Second, a **three-path channel split strategy** is proposed to reduce channel redundancy by handling part of the input channels with more efficient pointwise convolution and skip-connection pathways. Furthermore, a **mixed self-attention mechanism** is proposed to optimize the computation cost of the self-attention in the network by adaptively choosing the algorithm with lower time complexity according to the input token lengths and hidden dimensions. Extensive experiments on three downstream tasks show that our Efficiency-optimized Video Diffusion Models can achieve an **10**× speed-up while achieving comparable or even better results in the performance of fidelity compared with the state-of-the-art methods. The code is available at https://github.com/PKU-ICST-MIPL/EVDM_ACMMM2023.

CCS CONCEPTS

• **Computing methodologies** → *Computer vision*; *Image and video acquisition.*

KEYWORDS

Diffusion model; Video generation; Model acceleration

*Corresponding author.

MM '23, October 29-November 3, 2023, Ottawa, ON, Canada

ACM ISBN 979-8-4007-0108-5/23/10…$15.00
https://doi.org/10.1145/3581783.3612406

ACM Reference Format:
Zijun Deng, Xiangteng He, and Yuxin Peng. 2023. Efficiency-optimized Video Diffusion Models. In *Proceedings of the 31st ACM International Conference on Multimedia (MM '23), October 29-November 3, 2023, Ottawa, ON, Canada.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3581783.3612406

1 INTRODUCTION

Today, the widespread of electronic devices has significantly advanced the development of news media technology. Different types of videos can be readily accessed through various smart devices such as laptops, smartphones, and so on. Therefore, there is a great demand for generating high-quality videos but it is time-consuming to generate videos manually. As a result, automatically generating high-fidelity coherent videos has recently attracted more and more attention. Some research works have demonstrated its promising performance in promoting the development of short video and movie industries.

Video Diffusion Models have recently shown promising results in synthesizing high-fidelity videos. Based on the Denoising Diffusion Possibility Models, they gradually add Gaussian noise to the videos in the forward process, then use a denoising network to predict the added noise in the reverse process to reconstruct the videos. The forward process usually takes thousands of steps to ensure the corrupted videos are approximately Gaussian noise. In addition, to simplify the calculation, each added noise should be small enough to make the posterior distribution approximate to the Gaussian distribution. As a result, they need large networks to predict the subtle added noise in each step. To acquire satisfactory video samples, the reverse process also requires a lot of network evaluations to reverse the long noise-adding process, which results in the remarkable growth of computation cost. Therefore, the high-fidelity results of the Video Diffusion Models owe much to trading fidelity with speed, which leads to expansive sampling and training costs.

Since the long inference time is an obstacle to the application of diffusion models, existing works [13, 23, 26] consider using fewer network evaluations to approximate the reverse denoising process. However, such solutions do nothing to reduce the training time because the computation cost of the large denoising network remains high, demanding vast amounts of computing resources to train diffusion models. Moreover, as video data consume more calculations in the network than 2D data like images, training a video diffusion model is rather time-consuming compared with other diffusion models like well-known text-to-image diffusion models.

To reduce the computation cost of the video diffusion models, we propose Efficiency-optimized Video Diffusion Models. To start with, we choose to use 2D U-net networks for our optimization since 3D convolution and self-attention operation take significantly more time than the 2D version. We then examine the source of the computation cost and confirm that the majority comes from the convolution and self-attention layers. Moreover, we observe that the convolution takes much more calculations than the self-attention because of the channel redundancy. Therefore, we reduce the network's computation cost mainly by minimizing the input and output channels of the convolution. As part of the computation cost also comes from the self-attention layers, we further develop a mixed self-attention algorithm to reduce their computation cost.

To reduce the channel redundancy of the convolution, we first consider using computationally more efficient operations to replace the original 3x3 convolution and handle part of the input channels. Therefore, we propose a three-path channel split strategy, which adds pointwise convolution and direct input-output link pathways to deal with part of the input channels. To further reduce the computation cost, we use a bottleneck pointwise convolution to downsample the channel of the input feature maps. However, the temporal information can easily get lost in channel-wise downsampling because the 2D architecture integrates the temporal axis into the channel axis. To address this issue, we propose to enhance the downsampled feature with temporal information, thus ensuring both efficiency and fidelity.

For self-attention acceleration, previous methods [3, 14, 33] mainly base their algorithm on the assumption that the token length is bigger than the hidden dimension. This assumption holds in most cases but fails in the denoising network of diffusion models because the size of the input token and its hidden dimension is not fixed in different network layers. Therefore, we devise a mixed self-attention mechanism, which uses a threshold gate to analyze the token length and hidden dimension and adaptively chooses the self-attention algorithm with a lower computation cost.

Our main contribution can be summarized as follows:

- A **bottleneck residual pathway** is proposed to downsample the channel of the features that flow through the network by a bottleneck pointwise convolution. Noticing that the temporal features lie in the channel axis in the 2D architecture, we conduct a temporal enhancement to bring back the lost temporal information in the channel-wise downsampling.
- A **three-path channel split strategy** is proposed to reduce the channel redundancy of convolution via incorporating pointwise convolution pathways and direct link pathways to handle part of the input channels.
- A **mixed self-attention mechanism** is proposed to analyze the input's token length and hidden dimension and adaptively choose the self-attention algorithm with a lower computation cost.

To verify the effectiveness of our model, we conduct extensive experiments on three downstream tasks: video prediction, video interpolation, and video generation. Experimental results of these three tasks on SM-MNIST, KTH, BAIR, and UCF datasets show that our Efficiency-optimized Video Diffusion models can achieve a 10× reduction of the computation cost while achieving comparable or even better results in the performance of fidelity compared with the state-of-the-art methods.

2 RELATED WORK

Video Prediction and Generation. Early video Synthesis studies mainly focus on the video prediction task [16, 32], which is, predicting a sequence of future frames according to the past frames. Early works [6, 20, 27] mainly adopted a combination of convolutional and recurrent modules for passive video prediction. However, this framework is limited by its partial observation of the past frames [40], therefore is poor in predicting the long-term future. Later, GAN-based methods have shown promising results on many tasks. DMVP [18] was the first work to employ adversarial learning in the network to improve the synthesis quality. After that, a lot of methods [22, 29, 34, 35] were proposed to further improve the generation quality. In this period, the more challenging task, unconditional video generation, began to be studied. StyleGAN-v [25] employs the powerful StyleGAN2 architecture for video generation, realizing high-resolution video generation. GAN-based methods can generate high-quality videos on single-domain videos. However, due to the well-known mode-coverage [22, 22] shortage of the GAN, these methods lack the ability to generate open-domain videos. In addition, limited by inductive bias on the locality of convolutional neural networks, the GAN-based methods struggle with complex scenes with multiple objects.

Recently, Autoregression Transformer has emerged as a powerful generative model, showing promising results in image generation. Some works [11, 19, 37, 38] employ Autoregression models in the video generation field. Among them, Latent Video Transformer [19] proposed an architecture to model the video frames in the latent space and predict latent representation for the next frames in an autoregressive manner. Nuwa [38] and Cogvideo [11] focus on open-domain text-to-video generation. Although these methods can generate open-domain videos, they suffer from unidirectional bias and accumulated prediction errors [8], which limits their sample quality greatly.

Denoising Diffusion Probabilistic Models. Denoising Diffusion Probabilistic Models were first proposed in [9]. [5] first applies it to image generation and achieves great success. To further applies diffusion models to video generation, VDM [10] extends the previous 2D architecture to 3D and employs the temporal attention component to learn the temporal feature of the videos. Make-a-video [24] further introduces temporal 1D convolution to capture the temporal feature of the videos. It also proposes the frame interpolation network to generate high-fps videos. Phenaki [30] further studies the challenging long video generation and realizes generating video with 2 minutes length.

Although video diffusion models can generate open-domain high-fidelity videos, their success owes much to the trade between quality and speed (section 1), which leads to expansive sampling and training costs. In our work, we focus on reducing training time.

Acceleration for the Diffusion Model. DDIM [26] was the first work to accelerate the diffusion model. Denoising Diffusion Probabilistic Models require simulating a Markov chain for many steps to produce a sample, in which every step requires a network evaluation. To remedy this, DDIM generalizes DDPMs via a class

of non-Markovian diffusion processes, which use fewer steps to produce the sample. GGF [13] devises a Stochastic Differential Equation (SDE) solver with adaptive step sizes tailored to the diffusion models, which can use 10x fewer steps than the DDPM. PD diffusion [23] employs a progressive distillation mechanism to distill a trained deterministic diffusion sampler, which realizes using half as many sampling steps as the DDPM while maintaining the sample quality.

The above methods achieve strong results in reducing sampling costs by approximately using fewer steps to produce a sample. However, they can not reduce the computation cost of the network, thus the training cost remains expensive. Our approach can accelerate both the training and the sampling by reducing the computation cost of the network.

3 EFFICIENCY-OPTIMIZED VIDEO DIFFUSION MODELS

3.1 Video Diffusion Model

The video Diffusion Model is based on the Denoising Diffusion Probabilistic Model(DDPM), which learns the distribution of the data by gradually reconstructing the data. Specifically, the Denoising Diffusion Probabilistic Model first conducts a forward process that corrupts the data by gradually adding Gaussian noise to the data in a Markov chain. Then, it learns a reverse process to restore the structure of the data.

Let x_0 be a sample from the distribution $x_0 \sim q(x_0)$. The forward process adds Gaussian noise to the x from $t = 0$ to $t = T$:

$$q_t\left(x_t \mid x_{t-1}\right) \coloneqq \mathcal{N}\left(x_t; \sqrt{1-\alpha_t}x_{t-1}, \alpha_t \mathbf{I}\right), \tag{1}$$

where x_t denotes the sample in the t step. Moreover, the x_t can be directly derived by x_0:

$$q_t\left(x_t \mid x_0\right) \coloneqq \mathcal{N}\left(x_t; \sqrt{\bar{\alpha}_t}x_0, \left(1-\bar{\alpha}_t\right)\mathbf{I}\right), \tag{2}$$

where $\bar{\alpha}_t = \prod_{s=1}^{t}\left(1-\alpha_s\right)$.

Studies show that if the noise we added in the forward process is small enough, then the posterior distribution $q\left(x_{t-1} \mid x_t\right)$ can approximately be considered as a diagonal Gaussian distribution. Therefore, we use a network θ to learn the mean $\mu_\theta\left(x_t\right)$ and variance $\Sigma_\theta\left(x_t\right)$ of the approximate distribution p_θ :

$$p_\theta\left(x_{t-1} \mid x_t\right) \coloneqq \mathcal{N}\left(\mu_\theta\left(x_t\right), \Sigma_\theta\left(x_t\right)\right) \tag{3}$$

However, it's inefficient to use $q_t\left(x_{0:T}\right)$ to supervise the $p_\theta\left(x_{1:T} \mid x_0\right)$, as each of $x_{0:T}$ need to be calculate. To address this issue, the diffusion model uses $q\left(x_{t-1} \mid x_t, x_0\right)$ to conduct the reverse process:

$$q\left(x_{t-1} \mid x_t, x_0\right) \coloneqq \mathcal{N}\left(x_{t-1}; \tilde{\boldsymbol{\mu}}_t\left(x_t, x_0\right), \tilde{\beta}_t \mathbf{I}\right), \tag{4}$$

where $\tilde{\mu}_t\left(x_t, x_0\right) = \frac{\sqrt{\bar{\alpha}_{t-1}}\alpha_t}{1-\bar{\alpha}_t}x_0 + \frac{\sqrt{\alpha_t}\left(1-\bar{\alpha}_{t-1}\right)}{1-\bar{\alpha}_t}x_t$, and $\tilde{\beta}_t = \frac{1-\bar{\alpha}_{t-1}}{1-\bar{\alpha}_t}\alpha_t$. According to eq. 2, x_0 can be estimated by x_t:

$$\hat{x}_0 = \left(\mathrm{x}_t - \sqrt{1-\bar{\alpha}_t}\epsilon\right)/\sqrt{\bar{\alpha}_t}, \tag{5}$$

where $\epsilon \sim \mathcal{N}(0, \mathbf{I})$. As $p_\theta\left(x_0 \mid x_t\right)$ can be derived by eq.3, a $\epsilon_\theta\left(x_t \mid t\right)$ can also be deducted using a neural network θ conditioned on timestep embedding. Therefore, we can use the ϵ to supervise the $\epsilon_\theta\left(x_t \mid t\right)$ to efficiently train the network θ.

$$L(\theta) = E_{t\sim[1,T], x_0\sim q(x), \epsilon\sim\mathcal{N}(0,\mathbf{I})}\left[\|\epsilon - \epsilon_\theta\left(x_t \mid t\right)\|^2\right] \tag{6}$$

Given the model θ, following eq. 3, we can acquire data from noise. Directly adapting the above method can fulfill the goal of unconditional video generation. Further modifications to the architecture can adapt the network to video prediction and video interpolation tasks.

To adapt the diffusion model to the video prediction and video interpolation tasks, a straightforward idea is to condition the denoising process on the given past frames and future frames. To further unify these tasks in a single framework, binary masks m_p, m_f are applied to the past frames and future frames with a probability $p_{mask} = 0.5$:

$$\begin{aligned} L_{unify}(\theta) = E_{t\sim[1,T],[\mathbf{p},x_0,\mathbf{f}]\sim q(x),\epsilon\sim\mathcal{N}(0,\mathbf{I})} \\ [\|\epsilon - \epsilon_\theta(x_t \mid t, m_p\mathbf{p}, m_f\mathbf{f})\|^2] \end{aligned} \tag{7}$$

The binary mask design allows the network to denoise without any condition frames, which means the network trained on L_{unify} can do three tasks: video generation, video prediction, and video interpolation.

In the reverse process of the diffusion model, a network θ is needed to predict the Gaussian noise added in each forward step. Only a huge network is capable of such an incredibly difficult task, as each of the added noises should be small enough to make the posterior distribution approximately a diagonal Gaussian distribution (see section 3.1). However, the training cost of such a huge network is extremely expensive. To address this issue, we focus on reducing the diffusion model's training cost by optimizing the design of convolution and self-attention in the denoising network.

3.2 Bottleneck Residual Pathway

As can be observed in Table 1, the most frequent operation in the network is convolution. Therefore, We first consider reducing the computational cost of the convolution in the network. Observed that the huge computation costs mainly owing to the large input channel and output channel of the convolution kernels, we propose to downsample the channel of the feature maps that flow through the convolution.

Inspired by the inception module of the GoogleNet [28], we design a Bottleneck Residual Pathway that first uses a bottleneck pointwise convolution to downsample the channels of the feature maps, then sends the downsampled feature to two convolution pathways and concatenates their output. We also consider that the channel-wise downsampling may cause the loss of temporal information. To remedy this, we enhance downsampled features with the embedding of timestep and past frames. We also conduct this enhancement to the features before the bottleneck convolution, to encourage the network further aware of temporal information.

Specifically, given the input feature $\mathrm{X} \in R^{B\times C\times H\times W}$, where B, C, H, W denotes the batch size, channels, and height and width of the feature maps respectively, we first normalize and enhance it with the embedding of the timestep and past frames:

$$\begin{aligned} \mathrm{Y} &= GN(\mathrm{X}) \cdot (1 + \mathrm{e}_1) + \mathrm{e}_2, \\ \mathrm{e} &= ext(G_{emb}(\mathbf{p}, t)W_l + b_l), \end{aligned} \tag{8}$$

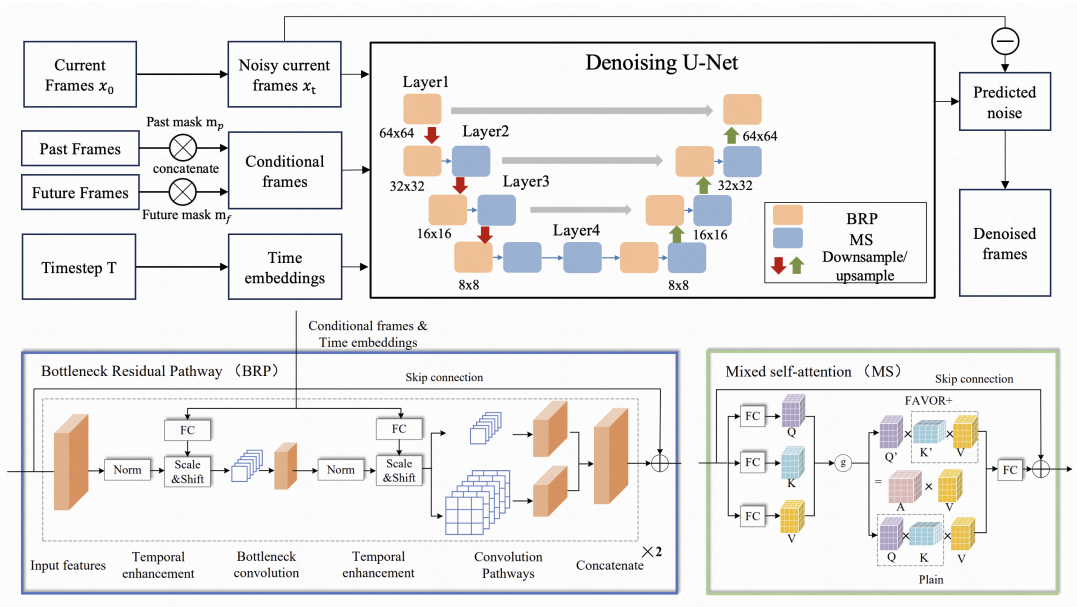

Figure 1: The framework of the Efficiency-optimized Video Diffusion Models. Given noisy current frames, the network predicts the noise added to them and generates denoised frames, receiving past and future frames as conditional information.

where $\mathrm{e} \in R^{B\times E\times 1\times 1}$ is the concatenation of $\mathrm{e}_1, \mathrm{e}_2 \in R^{B\times E/2\times 1\times 1}$ and E denotes the hidden dimension of embedding. $GN(.)$ denotes GroupNorm, G_{emb} refers to the embedding network for the past frames $\mathbf{p}$ and timestep t. $W_l \in R^{E\times C}$ and $b_l \in R^C$ are the parameters of a linear layer, $ext(.)$ extends the output of the linear layer to 4 dimensions. Note that the frame dimension of the video is integrated into the channel dimension to reduce the computational cost.

Then we pass $\mathrm{Y} \in R^{B\times C\times H\times W}$ to the bottleneck pointwise convolution, downsampling the channels of the feature maps, and enhance it with the embedding of the past frames again:

$$\mathrm{Y_b} = GN(ReLU(Y) * \mathrm{K}_1) \cdot (1 + \mathrm{e}_1) + \mathrm{e}_2, \tag{9}$$

where $\mathrm{K}_1 \in R^{1\times 1\times C\times \frac{C_o}{\delta}}$ is the kernel of the pointwise convolution, C_o is the ouput channel of Bottleneck Residual Pathway.

Finally, we send the downsampled features $\mathrm{Y_b} \in R^{B\times \frac{C_o}{\delta}\times H\times W}$ to two convolution pathways and concatenate their output. One pathway is a 3x3 convolution, the other is a pointwise convolution:

$$\mathrm{Y_{out}} = [\mathrm{Y_b} * \mathrm{K}_2, \mathrm{Y_b} * \mathrm{K}_3], \tag{10}$$

where $[\cdot,\cdot]$ denotes the concatenation operation, $\mathrm{K}_2 \in R^{1\times 1\times \frac{C_o}{\delta}\times \frac{C_o}{2\delta}}$ and $\mathrm{K}_3 \in R^{3\times 3\times \frac{C_o}{\delta}\times \frac{C_o}{2\delta}}$ are the kernels of the 3x3 convolution and pointwise convolution respectively. Our Bottleneck Residual Pathway repeats the above procedure and aggregates the output with a skip connection to avoid degradation.

3.3 Mixed Self-attention Mechanism

Self-attention layers in the network are also highly time-consuming. Therefore, we consider reducing the computation cost of self-attention in the network. Given a feature map $\mathrm{X} \in R^{B\times C\times H\times W}$, we first reshape it into a 3-dimension feature and use 3 linear layers T to calculate the query, key, and value metrics $Q, K, V \in R^{B'\times L\times d}$, $(B' = Bh, L = HW, d = C/h_n)$:

$$Q = TW_1 + b_1, K = TW_2 + b, V = TW_3 + b_3 \tag{11}$$

where h_n denotes the number of heads in layer $n (1 \le n \le 4)$, $W_1, W_2, W_3 \in R^{d\times d}, b_1, b_2, b_3 \in R^d$ are parameters of these linear layers. The plain self-attention algorithm then calculates the attention metrics and applies it to the V:

$$T_{out} = (Softmax(A)V)W_4 + b_4, A = QK^\top/\sqrt{d} \tag{12}$$

where $A \in R^{B'\times L\times L}$ is the attention metric and $W_4 \in R^{d\times d}, b_4 \in R^d$ are the parameters of the last linear layer of self-attention. The time complexity of eq. 12 is $O(L^2 d)$.

The FAVOR+ algorithm [3] approximates $Q', K' \in R^{B'\times L\times d}$ that satisfies:

$$AV = Q'(K'^\top V) \tag{13}$$

This allows them to reduce the time complexity to $O(Ld^2)$.

However, just like most self-attention acceleration algorithms, the FAVOR+ algorithm assumes that $L \gg d$. This assumption holds in most cases but fails in the denoising network of the diffusion model. To minimize the time complexity of the self-attention in all network layers, we devise a threshold gate that when the condition of the gate is satisfied, we use FAVOR+, otherwise the plain algorithm:

$$L > \lambda d \Rightarrow \frac{HW}{4^n} > \lambda \frac{C_0}{h_n} M_n \tag{14}$$

where C_0 is the channel of the first network layer, n is the number of layer, $\lambda > 1$ is the hyperparameter that defines how many times should the L bigger than d. The eq.14 is based on the network architecture that in Layer n, the width and height are reduced by 2^n while the channels are multiplied by M_n. Our mixed attention mechanism guarantees that in the self-attention layer that uses

Table 1: Parameters and FLOPs comparison in each layer of MCVD[31] and our models when base channel $c_0 = 96$. We only analyze the convolution and the attention blocks, as they constitute the majority of parameters and FLOPs.

Layer	Convolution #Params (M)			Convolution #GFLOPs			Attention #GFLOPs		
	MCVD	EVDM	EVDM-L	MCVD	EVDM	EVDM-L	MCVD	EVDM	EVDM-L
Layer1	2.59	1.66	1.36	334.33	59.78	16.77	0	0	0
Layer2	8.23	4.07	2.99	483.53	95.7	25.22	106.49	36.5	36.5
Layer3	16.86	6.99	4.72	274.08	53.9	13.93	23.63	19.12	19.12
Layer4	31.14	11.4	7.26	139.67	26.82	6.86	10.5	10.92	10.92
total	58.82	24.12	16.33	1231.61	236.2	66.6	140.62	66.53	66.53
reduction	-	× 2.43	× 3.60	-	× 5.21	× 18.49	-	× 2.11	× 2.11

FAVOR+, the time complexity is less than $O(L^2 d/\lambda)$, thus at least λ time faster than the plain algorithm.

3.4 Further Architecture Optimizations

As Table 1 shows, although the above method reduces the computation cost significantly, the computation cost of the convolution remains high and takes a large part of the overall complexity. Therefore, taking the architecture in the above section as a basic version of our approach, in this section, we further optimize the BRP to reduce the computation cost and develop a lighter version (EVDM-L).

We first observe that the skip connection uses a pointwise convolution to map the large input feature maps into the output. The input channel and the output channel of this pointwise convolution are large, resulting in a big computation cost. Based on the observation, we propose a shortcut optimization method to reduce the computation cost. We first consider conducting the convolution by groups, which divides the channels of input feature maps into N_g groups and use N_g small convolution kernels to calculate the results of each group. However, conducting convolution by groups also prevents the channel groups from communicating, and weakens the feature extraction. To remedy this, we adopt the channel shuffle strategy [42] to encourage channel-wise communication.

In addition, we design a new channel split way to further strengthen channel-wise communication and reduce the possible information loss of the bottleneck design. Given the input feature maps with C_{in} channels, the output feature maps have C_{out} channels, we consider two circumstances: $C_{in} < C_{out}$ and $C_{in} \geq C_{out}$. These two cases

Table 2: Comparison between the MCVD and our EVDM models. A more specific comparison of the model used in the KTH dataset is given in Table 1.

Dataset	Model	C_0	Params	GFLOPs	FVD
SM-MNIST	MCVD	64	27.9M	631.91	23.9
SM-MNIST	EVDM	64	12.5M	189.66	**12.9**
SM-MNIST	EVDM-L	64	**5.92M**	**60.64**	19.2
KTH	MCVD	96	62.8M	1376.87	323.0
KTH	EVDM	96	28.1M	307.41	**290.0**
KTH	EVDM-L	96	**20.4M**	**133.99**	298.0
BAIR	MCVD	192	251.2M	5328.15	89.5
BAIR	EVDM	192	112.4M	1219.60	**59.0**
BAIR	EVDM-L	192	**53.0M**	**526.48**	64.4
UCF	MCVD	288	739.4M	9946.38	1143.0
UCF	EVDM	288	252.7M	1362.88	**1103.6**
UCF	EVDM-L	288	**182.7M**	**583.38**	1118.4

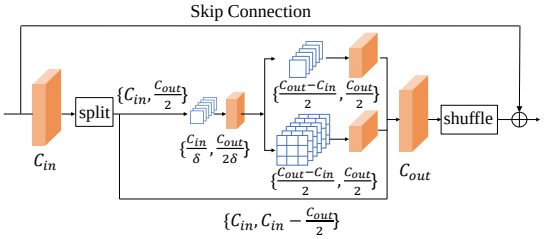

Figure 2: An illustration of our channel split strategy. For $\{C_x, C_y\}$ in the figure, C_x denotes the channels in the $C_{in} < C_{out}$ case, C_y denotes the $C_{in} \geq C_{out}$ case.

mostly occur in the downsample layers and the upsample layers, respectively. For the first case, we send the input feature maps directly into the bottleneck convolution. In the other path, the input feature maps are concatenated with the output of the convolution pathways, which has $\frac{C_{out}-c_{in}}{2}$ channels. For the second case, we split the channels of the input feature maps into two groups with $\frac{C_{out}}{2}, C_{in} - \frac{C_{out}}{2}$ channels respectively. The former is sent into the bottleneck convolution, and the latter is concatenated with the output of the convolution pathways that have $\frac{C_{out}}{2}$ channels. We give an illustration of our channel split strategy in Figure 2.

With the channel split strategy, part of the input will be directly integrated into the output feature maps. The channel shuffle operation also avoids the channel split choosing a fixed part of channels. These designs not only reduce the computation cost by cutting down the input channels and output channels of the convolution but also allow the input features to strengthen the convolution results and mitigate the information loss of the bottleneck convolution. We also increase the value of δ to downsample the channels more radically in the bottleneck convolution for the faster version of our approach.

4 EXPERIMENTS

4.1 Datasets

To evaluate the performance of our Efficiency-optimized Video Diffusion Models, we conduct experiments on SM-MNIST, KTH, BAIR, and UCF101 datasets. (1) **SM-MNIST** is a video dataset of moving handwritten digits. It contains 60,000 black-and-white videos in 64x64 resolution. (2) **KTH** is a human action dataset of a single person acting in a simple background. There are 600 greyscale videos with 64x64 resolution in this dataset. (3) **BAIR** is a dataset of 64x64 videos of a robot pushing objects on the top of a table. It contains 44,000 RGB videos in total. (4) **UCF101** is a dataset of

Table 3: Video prediction results on BAIR (64×64) trained on k frames to predict 15 frames based on one frame.

Method	k	Params	GFLOPs	FVD ↓	PSNR ↑	SSIM ↑
MCVD-s [NeurIPS 2022] [31]	5	328.6M	8880.08	103.8	**18.8**	0.826
CCVS [NeurIPS 2021] [15]	15	-	-	99.0	-	-
MCVD-c [NeurIPS 2022] [31]	5	251.2M	5326.79	98.8	**18.8**	0.829
Video Transformer [ICLR 2020] [36]	15	373.0M	-	96.0	-	-
MCVD concat pf-mask [NeurIPS 2022] [31]	5	251.2M	5328.48	89.5	16.9	0.780
EVDM-L (Ours)	5	**53.0M**	**526.48**	64.4	18.7	0.822
EVDM (Ours)	5	112.4M	1219.6	**59.0**	**18.8**	**0.830**

realistic action videos collected from YouTube. It has 101 action categories of 13,320 RGB videos. We follow [31] to preprocess the data and split the train-test set.

4.2 Evaluation Metrics

We use FVD, SSIM, and PSNR to quantify the performance of our model. We also compute the parameters and FLOPs for speed and model size comparison. FVD calculates the distance of the I3D feature between the synthesis videos and real videos. A lower FVD means the generated videos of the network are closer to the real videos. SSIM measures the structure similarity of two images, considering the similarity of the luminance, contrast, and structure. We follow the previous works to calculate the maximum SSIM across the generated frames. A higher SSIM also indicates that the generated videos are closer to the real videos. PSNR measures the ratio between the power of a signal and the power of noise that affects the fidelity of its representation. A higher PSNR refers to the higher quality of the video, which cannot easily be corrupted by the noise. FLOPs refer to floating point operations, a lower FLOPs mean the network has less computation cost and can run faster. Our FLOPs results are calculated under the batch size of 64.

4.3 Implement Details

Following [31], we use different sizes of models in the experiments to leverage the computation resources more efficiently. All of our models use self-attention in Layer 2, Layer 3, and Layer 4. The M_n of all our models are set to $[1, 2, 3, 4]$, $1 \leq n \leq 4$. h_n is calculated according to $d = C/h_n, C = C_0 M_n$. λ is set to 2 in our network. δ is set to 4 in the basic version of our approach, and 8 in the faster version of our approach (EVDM-L). Our models are trained on ≤ 4 Nvidia A40 GPUs in 1-6 days.

4.4 Comparison with the State-of-the-art

For comparison experiments, we evaluate the performance of our approach on 3 different downstream tasks: video prediction, video generation, and video interpolation. We show the results of video prediction in Tables 3 - 4 on BAIR and KTH. We present the unconditional video generation results on UCF and BAIR in Tables 5 & 6. We present the video interpolation results of SM-MNIST, KTH, and BAIR in Table 7.

First, we evaluate our performance in video prediction tasks on KTH, and BAIR datasets. Table 3 shows the BAIR dataset results. Our approach uses the least parameters and achieves the best result in all evaluation metrics. Whatś especially exciting is that our EVDM-L model significantly outperforms the previous SOTA with only 1/10 of its computation cost. Regarding our basic approach first (EVDM), we achieve a 2.23x reduction in model size and a 4.37x reduction in computation cost in terms of efficiency. In terms of fidelity, the significantly lower FVD shows that our approach can generate videos with higher quality and is also closer to real videos. These promising results are mainly credited to our careful optimization of the convolution and self-attention layers. Our bottleneck residual pathway conducts a channel-wise downsample, enhancing the downsampled features with temporal information. Our mixed self-attention adaptively chooses the self-attention algorithm with a lower computation cost. The faster version of our approach also significantly outperforms the previous SOTA in terms of efficiency and fidelity. This credits to our further optimization of BRP in our basic approach. The shortcut optimization employs group convolution to reduce the computation cost of the shortcut connection. The channel split strategy reduces the input channels and output channels of the convolution pathways, while also boosting channel-wise communication. These careful designs enable the faster version of our approach to be more efficient than the basic version.

Table 4: Video prediction results on KTH (64 x 64), predicting 30 frames based on 10 frames.

Method	FVD ↓	PSNR ↑	SSIM ↑
SAVP [arxiv 2018] [4]	374	26.5	0.756
MCVD [NeurIPS 2022] [2]	323	27.5	0.835
SLAMP [ICCV 2021] [1]	228	29.4	0.865
SRVP [ICML 2020] [7]	222	**29.7**	**0.870**
EVDM-L 100 steps (Ours)	298	26.8	0.816
EVDM 100 steps (Ours)	290	27.2	0.833
EVDM 1000 steps (Ours)	**206**	28.1	0.828

Our faster approach (EVDM-L) achieves a **10.4**× speed up compared with MCVD while performing comparably in FVD and SSIM metrics. These results indicate that EVDM-L is much more efficient than the MCVD.

Table 4 shows the video prediction results of the KTH dataset. The MCVD in the table is the MCVD concat model. We can observe from Table 2 and Table 4 that both EVDM and EVDM-L outperform the MCVD by a large margin while achieving **4.47**× and **10.3**× speed-up. Our EVDM and EVDM-L reduce 11% and 8% the FVD score of the MCVD using 100 DDPM sampling steps.

Table 5: Unconditional generation of UCF101, generating 16 video frames.

Method	FVD ↓
MoCoGAN-MDP [ICCV 2019] [41]	1277.0
MCVD concat p-mask [NeurIPS 2022] [31]	1228.3
TGANv2 [IJCV 2020] [21]	1209.0
MCVD (spatin p-mask) [NeurIPS 2022] [31]	1143.0
EVDM-L (Ours)	1118.4
EVDM (Ours)	**1103.6**

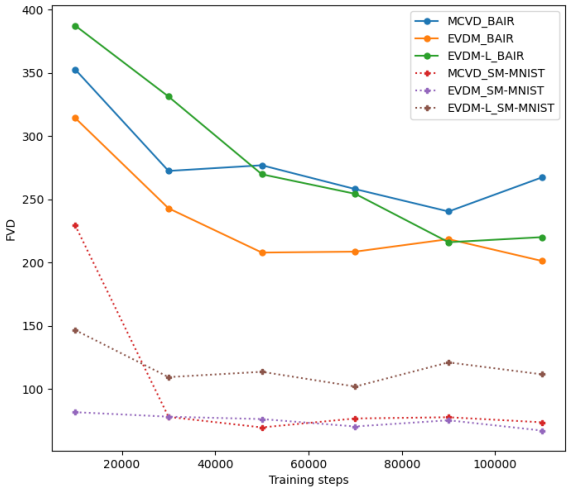

Figure 3: The FVD score of MCVD, EVDM, and EVDM-L during training.

Second, we evaluate our performance in the video generation task on UCF101 and BAIR datasets. Table 6 shows the video generation results of the BAIR dataset. The basic version of our approach outperforms the previous SOTA MCVD while achieving a 4x speed-up. EVDM-L achieves comparable results compared with MCVD with 21% its model size and **9.9%** its computation cost. Table 5 shows the video generation results of the UCF dataset. Combine the FLOPs data of Table 2 and the fidelity results of Table 5, we can observe that our basic version and faster version achieve comparable results while reducing the computation cost **7.3** times and **17** times compared with MCVD.

Third, we evaluate our performance in video interpolation tasks on SM-MNIST, KTH, and BAIR datasets. Table 7 reports the results. In all three datasets, our basic approach (EVDM) significantly outperforms the previous SOTA MCVD with much fewer parameters and calculations. Our EVDM approach improves the PSNR and SSIM results in SM-MNIST, KTH datasets, and BAIR datasets respectively, which indicates that our EVDM can generate videos with better quality and less noise.

To provide additional insights into why our approach maintains generative fidelity, we have conducted a new experiment that records the FVD scores of our approach and MCVD during training on the BAIR and SM-MNIST datasets. The results of this experiment are presented in Figure 3.

Upon observation, it is evident that our approach exhibits similar or even faster convergence rates compared to MCVD. This observation indicates that our acceleration approach does not compromise the network's capability, thereby validating that our approach maintains generative fidelity by reducing the redundancy in the network.

Table 6: Unconditional generation of BAIR, generating 16 video frames.

Method	FVD ↓
MCVD spatin p-mask [NeurIPS 2022] [31]	267.8
MCVD concat p-mask [NeurIPS 2022] [31]	228.5
EVDM-L (Ours)	231.7
EVDM (Ours)	**219.9**

4.5 Ablation Studies

To further verify the effectiveness of each component of our EVDM approach, we conduct ablation studies on the KTH dataset. Before analyzing the results, let's take a brief review of our approach. Our EVDM approach uses bottleneck residual pathway (BRP) and mixed self-attention mechanism (MSM) to reduce the computation cost of convolution and self-attention respectively. As the temporal enhancement in the bottleneck residual pathway plays an important role in our architecture, we also study it in the ablation experiments. To further reduce the computation cost, we design a shortcut optimization and a channel split strategy. And finally, we increase the value of δ to further accelerate. In the ablation experiment, we study the effectiveness of these components and present the results in Table 8.

4.5.1 Effectiveness of bottleneck residual pathway. Table 8 shows that the main acceleration is contributed by BRP. The results of the first 2 lines show that the BRP cuts down 72% of calculation operations of the origin network, which indicates the channel-wise downsample indeed greatly reduces the computation cost of the convolution. Except for speed, fidelity is also a critical issue. The results of the first 2 lines indicate that pure downsampling to the feature indeed causes information loss and damages the performance of the model. The results of the third line show that temporal enhancement helps remedy this issue by adding temporal information to the feature maps, helping FVD goes back to 359. In addition, by comparing the last 2 lines, we can see that the (BRP-eh) notably improves the FVD and PSNR metrics while keeping the computation cost unchanged. This indicates that temporal enhancement can help the models capture the temporal feature of the videos and generate videos that have higher fidelity and a closer distance to the real videos.

4.5.2 Effectiveness of mixed self-attention mechanism. Table 8 shows that the MSM also contributes to the reduction of the computation cost. In the fourth line, we add the mixed self-attention mechanism to the architecture of the second line. We can see that the GFLOPs of the fourth line are 21% fewer that the second line by adding the MSM. This shows the computation cost of self-attention is reduced by performing our mixed self-attention mechanism. Moreover, we can also observe that FVD and PSNR results of the fourth line are much lower than the second line, which indicates that the mixed self-attention mechanism also helps fix the information loss problem. This is because the mixed self-attention can encourage the model to focus on the crucial information and prevent them from getting lost in the channel-wise downsampling of BRP.

4.5.3 Effectiveness of shortcut optimization. In the second part of the ablation experiment, we take the EVDM as the baseline and study the effectiveness of the modules in the faster version of our approach. In the first line of this part, we add shortcut optimization (SO) to the basic version of our approach. In the third and the fourth line, we add the channel split strategy (Split) and both the SO and Split. The results of the first two lines show that the computation cost of the EVDM is 15.5% lower by adding the shortcut optimization to the architecture. In terms of fidelity, the FVD and PSNR results of the EVDM+SO decrease slightly, which implies that the group convolution of the shortcut optimization indeed weakens the

Table 7: Video interpolation results on SM-MNIST, KTH, and BAIR datasets. Given p past frames and f future frames, the model interpolates k frames. We report the average of the best metrics out of 10 trajectories per test sample in SM-MNIST and KTH datasets, following MCVD. The rest results in the table are the best metrics out of 100 trajectories.

Method	SM-MNIST				KTH				BAIR			
	p+f	k	PSNR ↑	SSIM ↑	p+f	k	PSNR ↑	SSIM ↑	p+f	k	PSNR ↑	SSIM ↑
FSTN [ICCV 2017] [17]	18	7	14.730	0.765	18	7	29.431	0.899	18	7	19.908	0.850
superSloMo [CVPR 2018] [12]	18	7	13.387	0.749	18	7	28.756	0.893	-	-	-	-
SDVI full [WACV 2020] [39]	18	7	16.025	0.842	18	7	29.190	0.901	18	7	21.432	0.880
MCVD [NeurIPS 2022] [31]	10	5	27.693	0.941	10	5	35.611	0.963	4	5	25.162	0.932
EVDM-L (Ours)	10	5	26.536	0.933	10	5	**36.642**	0.962	4	5	24.668	0.926
EVDM (Ours)	10	5	**28.131**	**0.948**	10	5	36.616	**0.966**	4	5	**25.567**	**0.936**

Table 8: Ablation studies on the KTH dataset. The BRP-eh refers to the temporal enhancement of the bottleneck residual pathway. SO refers to the shortcut optimization in section 3.4. Split indicates the channel split strategy.

Architecture	BRP	BRP-eh	MSM	SO	Split	δ	Params	GFLOPs	FVD ↓	PSNR ↑
baseline	×	×	×	×	×	-	62.8M	1376.87	317	26.6
baseline+BRP	√	×	×	×	×	4	25.4M	381.64	376	26.4
baseline+BRP+BRP-eh	√	√	×	×	×	4	28.1M	381.49	359	26.6
baseline+BRP+MSM	√	×	√	×	×	4	25.4M	307.53	349	26.6
EVDM	√	√	√	×	×	4	28.1M	307.41	**290**	**27.2**
EVDM+SO	√	√	√	√	×	4	32.5M	259.86	307	27.0
EVDM+Split	√	√	√	×	√	4	25.0M	226.12	300	26.9
EVDM+SO+Split	√	√	√	√	√	4	22.2M	147.32	300	27.2
EVDM-L	√	√	√	√	√	8	**20.4M**	**133.99**	298	26.8

communication between channel groups. The results of the fourth line show that our channel split strategy remedies this issue by connecting the channels of the input and output feature maps to boost channel-wise communication. We can also see from the third and fourth lines that the SO helps the EVDM+SO+Split outperforms the EVDM+Split, which verifies the effectiveness of the shortcut optimization.

4.5.4 Effectiveness of channel split strategy. Table 8 shows that the major computation reduction of the second version of our approach is contributed by the channel split strategy. Comparing the results of the first line and the third line, we can observe that the channel split strategy reduces 27% of the computation cost of the EVDM. The channel split strategy also helps the model maintain its accuracy while becoming faster and smaller. By comparing the results of the fourth line and the second line, we can see that the results of FVD and PSNR are improved, which verifies our claim that the channel split strategy can boost channel-wise communication and remedy the issues caused by group convolution. Moreover, the results of the last two lines indicate that although we increase the value of δ to downsample the channels more radically, the results of the FVD and PSNR are nearly unchanged. This verifies our claim that channel-wise splitting can weaken information loss.

The results of the ablation study show that all of our components contribute to the reduction of computation cost or performance improvement. Combining BRP and MSM can achieve 4x speed-up while achieving the best fidelity. With our further optimization of the architecture, our model can achieve 10x speed-up while maintaining fidelity.

5 CONCLUSION

In this paper, we present the Efficiency-optimized Video Diffusion Model, which reduces the computation cost of the video diffusion model by optimizing the design of convolution and self-attention blocks. Specifically, BRP performs depth-wise splitting to the convolution kernels and enhances the feature maps with temporal information, ensuring both efficiency and fidelity. A three-path channel split strategy is proposed to reduce the channel redundancy of convolution via incorporating pointwise convolution pathways and direct link pathways to handle part of the input channels. MSM adapts the fast self-attention algorithm to the variation of the token length and hidden dimension in the network, which minimizes the time complexity of self-attention. Our model achieves a 10 × speed up while maintaining the generative fidelity.

Although notably optimizing the efficiency of the video diffusion model, our work still has some limitations. Limited by the training data, our model currently can only generate a limited range of actions and scenes. Also, the videos generated by our model are still short and low-resolution compared with real movies. To overcome these limitations, our future work will focus on leveraging the superior efficiency of our proposed backbone to conduct large-scale pretraining. This endeavor aims to enhance our model's capability to generate open-domain, high-fidelity videos.

Table 9: Comparison of the training and inference time between the MCVD and our EVDM models.

Model	GPU	GPU hours	Latency (ms)
MCVD	Tesla A40	75.2	16.5
EVDM	Tesla A40	38.7	10.9
EVDM-L	Tesla A40	**25.4**	**6.1**
MCVD	Tesla A40	181.2	26.4
EVDM	Tesla A40	115.7	19.3
EVDM-L	Tesla A40	**73.1**	**7.5**

6 ACKNOWLEDGEMENT

This work was supported by the grants from the National Natural Science Foundation of China (62132001, 61925201).

REFERENCES

[1] Adil Kaan Akan, Erkut Erdem, Aykut Erdem, and Fatma Güney. 2021. Slamp: Stochastic latent appearance and motion prediction. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 14728–14737.

[2] Lluis Castrejon, Nicolas Ballas, and Aaron Courville. 2019. Improved conditional vrnns for video prediction. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 7608–7617.

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

[4] Emily Denton and Rob Fergus. 2018. Stochastic video generation with a learned prior. In *International conference on machine learning*. PMLR, 1174–1183.

[5] Prafulla Dhariwal and Alexander Nichol. 2021. Diffusion models beat gans on image synthesis. *Advances in Neural Information Processing Systems* 34 (2021), 8780–8794.

[6] Chelsea Finn, Ian Goodfellow, and Sergey Levine. 2016. Unsupervised learning for physical interaction through video prediction. *Advances in neural information processing systems* 29 (2016).

[7] Jean-Yves Franceschi, Edouard Delasalles, Mickaël Chen, Sylvain Lamprier, and Patrick Gallinari. 2020. Stochastic latent residual video prediction. In *International Conference on Machine Learning*. PMLR, 3233–3246.

[8] Shuyang Gu, Dong Chen, Jianmin Bao, Fang Wen, Bo Zhang, Dongdong Chen, Lu Yuan, and Baining Guo. 2022. Vector quantized diffusion model for text-to-image synthesis. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 10696–10706.

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

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

[11] Wenyi Hong, Ming Ding, Wendi Zheng, Xinghan Liu, and Jie Tang. 2022. CogVideo: Large-scale Pretraining for Text-to-Video Generation via Transformers. *arXiv preprint arXiv:2205.15868* (2022).

[12] Huaizu Jiang, Deqing Sun, Varun Jampani, Ming-Hsuan Yang, Erik Learned-Miller, and Jan Kautz. 2018. Super slomo: High quality estimation of multiple intermediate frames for video interpolation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 9000–9008.

[13] Alexia Jolicoeur-Martineau, Ke Li, Rémi Piché-Taillefer, Tal Kachman, and Ioannis Mitliagkas. 2021. Gotta go fast when generating data with score-based models. *arXiv preprint arXiv:2105.14080* (2021).

[14] Nikita Kitaev, Łukasz Kaiser, and Anselm Levskaya. 2020. Reformer: The efficient transformer. *arXiv preprint arXiv:2001.04451* (2020).

[15] Guillaume Le Moing, Jean Ponce, and Cordelia Schmid. 2021. Ccvs: Context-aware controllable video synthesis. *Advances in Neural Information Processing Systems* 34 (2021), 14042–14055.

[16] Ce Liu, Jenny Yuen, and Antonio Torralba. 2010. Sift flow: Dense correspondence across scenes and its applications. *IEEE transactions on pattern analysis and machine intelligence* 33, 5 (2010), 978–994.

[17] Chaochao Lu, Michael Hirsch, and Bernhard Scholkopf. 2017. Flexible spatio-temporal networks for video prediction. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 6523–6531.

[18] Michael Mathieu, Camille Couprie, and Yann LeCun. 2015. Deep multi-scale video prediction beyond mean square error. *arXiv preprint arXiv:1511.05440* (2015).

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

[20] MarcAurelio Ranzato, Arthur Szlam, Joan Bruna, Michael Mathieu, Ronan Collobert, and Sumit Chopra. 2014. Video (language) modeling: a baseline for generative models of natural videos. *arXiv preprint arXiv:1412.6604* (2014).

[21] Masaki Saito, Shunta Saito, Masanori Koyama, and Sosuke Kobayashi. 2020. Train sparsely, generate densely: Memory-efficient unsupervised training of high-resolution temporal gan. *International Journal of Computer Vision* 128, 10 (2020), 2586–2606.

[22] Tim Salimans, Ian Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, and Xi Chen. 2016. Improved techniques for training gans. *Advances in neural information processing systems* 29 (2016).

[23] Tim Salimans and Jonathan Ho. 2022. Progressive distillation for fast sampling of diffusion models. *arXiv preprint arXiv:2202.00512* (2022).

[24] Uriel Singer, Adam Polyak, Thomas Hayes, Xi Yin, Jie An, Songyang Zhang, Qiyuan Hu, Harry Yang, Oron Ashual, Oran Gafni, et al. 2022. Make-A-Video: Text-to-Video Generation without Text-Video Data. *arXiv preprint arXiv:2209.14792* (2022).

[25] Ivan Skorokhodov, Sergey Tulyakov, and Mohamed Elhoseiny. 2022. Stylegan-v: A continuous video generator with the price, image quality and perks of stylegan2. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 3626–3636.

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

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

[28] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. 2015. Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 1–9.

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

[30] Ruben Villegas, Mohammad Babaeizadeh, Pieter-Jan Kindermans, Hernan Moraldo, Han Zhang, Mohammad Taghi Saffar, Santiago Castro, Julius Kunze, and Dumitru Erhan. 2022. Phenaki: Variable length video generation from open domain textual description. *arXiv preprint arXiv:2210.02399* (2022).

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

[32] Jacob Walker, Abhinav Gupta, and Martial Hebert. 2014. Patch to the future: Unsupervised visual prediction. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*. 3302–3309.

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

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

[35] Yaohui Wang, Piotr Bilinski, Francois Bremond, and Antitza Dantcheva. 2020. G3AN: Disentangling appearance and motion for video generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 5264–5273.

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

[37] Chenfei Wu, Lun Huang, Qianxi Zhang, Binyang Li, Lei Ji, Fan Yang, Guillermo Sapiro, and Nan Duan. 2021. Godiva: Generating open-domain videos from natural descriptions. *arXiv preprint arXiv:2104.14806* (2021).

[38] Chenfei Wu, Jian Liang, Lei Ji, Fan Yang, Yuejian Fang, Daxin Jiang, and Nan Duan. 2022. Nüwa: Visual synthesis pre-training for neural visual world creation. In *European Conference on Computer Vision*. Springer, 720–736.

[39] Qiangeng Xu, Hanwang Zhang, Weiyue Wang, Peter Belhumeur, and Ulrich Neumann. 2020. Stochastic dynamics for video infilling. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*. 2714–2723.

[40] Wei Yu, Wenxin Chen, Songheng Yin, Steve Easterbrook, and Animesh Garg. 2022. Modular action concept grounding in semantic video prediction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 3605–3614.

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

[42] Xiangyu Zhang, Xinyu Zhou, Mengxiao Lin, and Jian Sun. 2018. Shufflenet: An extremely efficient convolutional neural network for mobile devices. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 6848–6856.