VidTwin: Video VAE with Decoupled Structure and Dynamics

Supplementary Material

A. Additional Experimental Results

To enhance the visual experience, we strongly encourage viewing the videos on the website.

A.1. Additional Reconstruction Examples

Fig. 3 presents additional reconstruction examples. By zooming in, one can observe that our VidTwin effectively captures intricate details, such as raindrops in the first and second cases. Moreover, by decoupling structural and dynamic motion features, our model excels at preserving rapid motion dynamics. For example, in the third case, VidTwin accurately reproduces the light trails of a fast-moving car, where other baselines fail to do so.

A.2. Additional Decoupling Examples

In Sec. 4.4.1 in the main text, we demonstrated the ability to separately recover the *Structure Latent* and *Dynamics Latent* components. Additional examples are shown in Fig. 2. Videos generated using *Structure Latent* predominantly capture primary structures and main objects, while those generated with *Dynamics Latent* focus on colors and rapid movements.

A notable example is observed in the bottom-right case, where fireworks visible in the first frame disappear in the second. However, the *Structure Latent*-generated video retains the fireworks from the first frame, demonstrating that *Structure Latent* effectively encodes low-frequency, gradually evolving information.

We would like to emphasize that our primary objective is not to completely decouple structure and dynamics, as this is a challenging problem even for humans. Instead, we observe potential in reducing temporal redundancy in video representation. Based on this observation, we designed an algorithm that strives to decouple video content into these two spaces. Therefore, the cross-reenactment experiment was only designed to intuitively demonstrate the roles of the two latents rather than being specifically optimized for cross-reenactment videos.

A.3. Additional Cross-Reenactment Examples

Fig. 4 provides further examples of the cross-reenactment experiments described in Sec. 4.4.1. In these examples, the generated videos inherit the basic structure from Video A while incorporating local details and motions from Video B. Notably, motion patterns such as horizontal movements and wave-like motions, as seen in the two bottom cases, are effectively transferred.

A.4. Comparison with Concurrent Baselines

Recent works have explored the field of video autoencoders [2, 9, 10, 18, 22]. We observe that most of these baselines still fall into the category of methods that represent frames as latent vectors of uniform size, as discussed in Sec. 1. A comparison between our model and these baselines is presented in Tab. 2. Notably, our model achieves performance comparable to state-of-the-art methods. CogVideoX [18] demonstrates impressive results, likely due to its large-scale training data. Additionally, even our highest compression rate model achieves a lower compression rate than other models (typically 0.6% with $8, 8, 4$).

B. Additional Analysis for VidTwin

B.1. Definition of Compression Rate and the Trade-off with Reconstruction Quality.

Differs from the typical representation that uses the downsampling factors for height, width, and number of frames for compression rate, we define it as the ratio between the dimension of the latent used in the downstream model and the input video's dimension. For example, the typical downsampling factor $(8, 8, 4)$ with channels 4 corresponds to a compression rate of 0.65% in our definition. Additionally, we present the trade-off between compression rate and reconstruction quality in Tab. 3. As shown, lower compression rates generally result in better reconstruction quality.

B.2. Initial Scalability Exploration

In Sec. 4.3, we described training our architecture at varying parameter scales and observed consistent performance improvements with larger models. Tab. 1 summarizes the configurations of each model, evaluated at the same training step. The results demonstrate a steady enhancement in reconstruction quality with increasing model size. In future work, we plan to explore additional model scales and investigate potential scaling laws, including exponential trends and other patterns.

Table 1. Settings and performance of VidTwin at different scales.

Models	Depth	Num. Heads	Dim. Hidden	Num. Params.	PSNR	SSIM
VidTwin $_{\text{small}}$	12	8	512	126M	24.83	0.683
VidTwin $_{\text{base}}$	16	12	768	335M	26.13	0.732
VidTwin $_{\text{large}}$	16	12	1536	1.3B	27.16	0.751

Table 2. Comparison with other concurrent works.

Models (Comp. Rate)	PSNR↑	LPIPS↓
CV-VAE (0.53%)	28.06	0.24
OD-VAE (0.53%)	29.18	0.19
Open-Sora (0.53%)	29.89	0.15
CogVideoX (0.53%)	**31.92**	**0.09**
VidTwin (0.20%)	28.14	0.24
VidTwin (0.48%)	**30.04**	**0.15**

Table 3. The reconstruction quality of different compression rates.

Compression Rate	PSNR↑	LPIPS↓
0.11%	24.41	0.35
0.16%	27.03	0.28
0.20%	28.14	0.24
0.48%	**30.04**	**0.15**

Table 4. Subjective evaluation of VidTwin with increased frames and higher FPS.

Model	Sem. ↑	Tempo. ↑	Deta. ↑
VidTwin	4.71	4.62	4.73
w/ 32 frames	4.70	4.53	4.69
w/ 40 fps	4.73	4.64	4.71

B.3. Performance of VidTwin with Increased Frames and Higher FPS

We selected the same subjective evaluation subset as in Sec. 4.2 and sampled videos with 32 frames and 40 FPS. A new user study was conducted, and the results are presented in Tab. 4. The findings indicate that VidTwin maintains strong performance with an increased number of frames and a higher frame rate.

B.4. Failure modes of VidTwin

We provide a failure case in Fig. 1, depicting a basketball scene with fast player movements and camera motion. While the background remains well-preserved, the fast-moving individuals appear blurred. In terms of components, the S. Latent captures the background but becomes blurred for the players, which is expected as it encodes slowly changing semantic information. The D. Latent captures the fast-changing players but struggles to accurately integrate them into the reconstructed video due to their extremely rapid movement. We plan to address this issue by pretraining on low-fps videos and fine-tuning on high-fps videos in future work.

Figure 1. Failure modes of VidTwin.

C. Additional Information on Experimental Settings

C.1. Baselines and Compression Rates

This section provides details on the baselines used in our evaluation and discusses their compression rates, as outlined in Sec. 4.2. Notably, MAGVIT-v2 [19], iVideoGPT [17], and CMD [20] do not offer official code or pretrained checkpoints. Therefore, we reimplement these methods based on the descriptions provided in their respective papers.

MAGVIT-v2 [19]: MAGVIT-v2 employs 3D causal CNN layers to downsample videos into latents, with a temporal downsampling factor of 4 and spatial downsampling factor of 8. The latent dimension is set to 5, as reported in the paper, resulting in a compression rate of:

$$\frac{5}{3 \times 4 \times 8 \times 8} \approx 0.65\%.$$

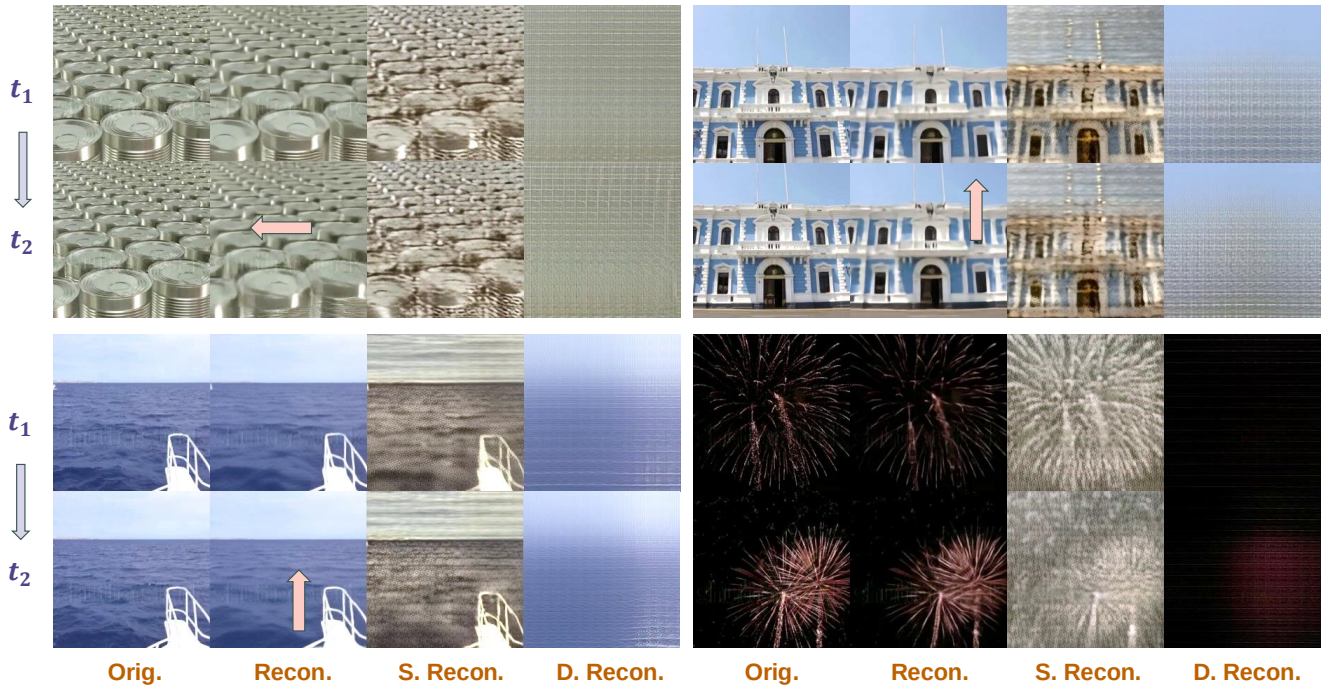

Figure 2. Additional examples of decoupling *Structure Latent* and *Dynamics Latent*.

EMU-3 [16]: EMU-3 is a generative model proposed by BAAI[1]. For our evaluation, we primarily utilize its video tokenizer, which is based on SBER-MoVQGAN[2]. This tokenizer incorporates two temporal residual layers with 3D convolutional kernels in both the encoder and decoder modules, enhancing video tokenization. Similar to MAGVIT-v2, it achieves a $4\times$ temporal compression and 8×8 spatial compression. The compression rate, with a latent size of 4, is calculated in the same manner.

CV-VAE [21] CV-VAE is a video VAE of latent video models, designed to have a latent space compatible with that of a given image VAE, such as the image VAE in Stable Diffusion (SD). In terms of compression rate, it matches that of EMU-3, achieving $4\times$ temporal compression and 8×8 spatial compression.

CMD [20]: CMD decouples video representations into content frames and motion latents. For a video of size (c, f, h, w), the content frame has dimensions (c, h, w), and the motion latent is $(d, h + w, f)$, where d is the dimension of the motion vector. Based on the settings described in the paper, the compression rate is:

$$\frac{1}{f} + \frac{d(h+w)}{chw} = \frac{1}{16} + \frac{2 \times 224 \times 32}{3 \times 224 \times 224} \approx 6.9\%.$$

[1] https://www.baai.ac.cn/

[2] https://github.com/ai-forever/MoVQGAN

The primary bottleneck lies in the content frame, and we hypothesize that longer video clips could reduce the compression rate (though at the potential cost of performance).

iVideoGPT [17]: iVideoGPT employs a conditional VQ-GAN [3] with dual encoders and decoders. The context frames $1 : T_0$ are encoded using N_0 tokens, while subsequent frames are encoded with fewer tokens (n), conditioned on the context tokens to capture the essential dynamics. The compression rate is given by:

$$\frac{N_0 d + n(T - T_0)d}{C \times T \times H \times W},$$

and, based on the information in the paper, we calculate it as:

$$\frac{2 \times 16^2 \times 64 + 14 \times 4^2 \times 64}{3 \times 16 \times 256^2} \approx 1.5\%.$$

C.2. Pseudo DiT for Resource Consumption Evaluation

Our VidTwin model offers a highly compressed latent space, significantly reducing the resource requirements of downstream generative models. To validate this, in Sec. 4.4.2, we compare the performance of a generative model applied to the latent spaces produced by VidTwin and the baselines.

For a fair comparison, we utilize the same DiT [11] architecture in all experiments. The configuration includes 6

Figure 3. Additional reconstruction cases comparing our VidTwin model with baselines. Zoom in to observe finer details.

layers, 8 attention heads, a hidden dimension size of 512, and a feed-forward network (FFN) dimension of 2048, resulting in a total of 12,610,560 parameters. Additionally, a unified patch size of 2 is used for all dimensions.

We calculate the FLOPs using a single sample (batch size $= 1$). For memory consumption, we employ the Adam [6] optimizer and record the maximum GPU memory usage during training.

D. Implementation Details

D.1. Model Details

As described in Sec. 3.1, our VidTwin adopts an Encoder-Decoder architecture. Specifically, we utilize a Spatial-Temporal Transformer [1] backbone. In each block, spatial attention is first applied to the height and width dimensions, followed by temporal attention along the temporal dimension. Temporal attention uses causal masking, ensuring that earlier frames do not attend to later ones, similar to the configuration in MAGVIT-v2 [19]. We evaluate three different scales (outlined in Tab. 1) by adjusting the depth, hidden state dimensions, and other parameters. For spatial dimensions, a patch size of 16 is used for both height and width, while for the temporal dimension, the patch size is set to 1.

The Q-Former [8], employed for extracting *Structure Latent* components, consists of 6 layers with a hidden dimension of 64 and 8 attention heads. For downsampling, we primarily use convolutional layers with a stride of 2, while

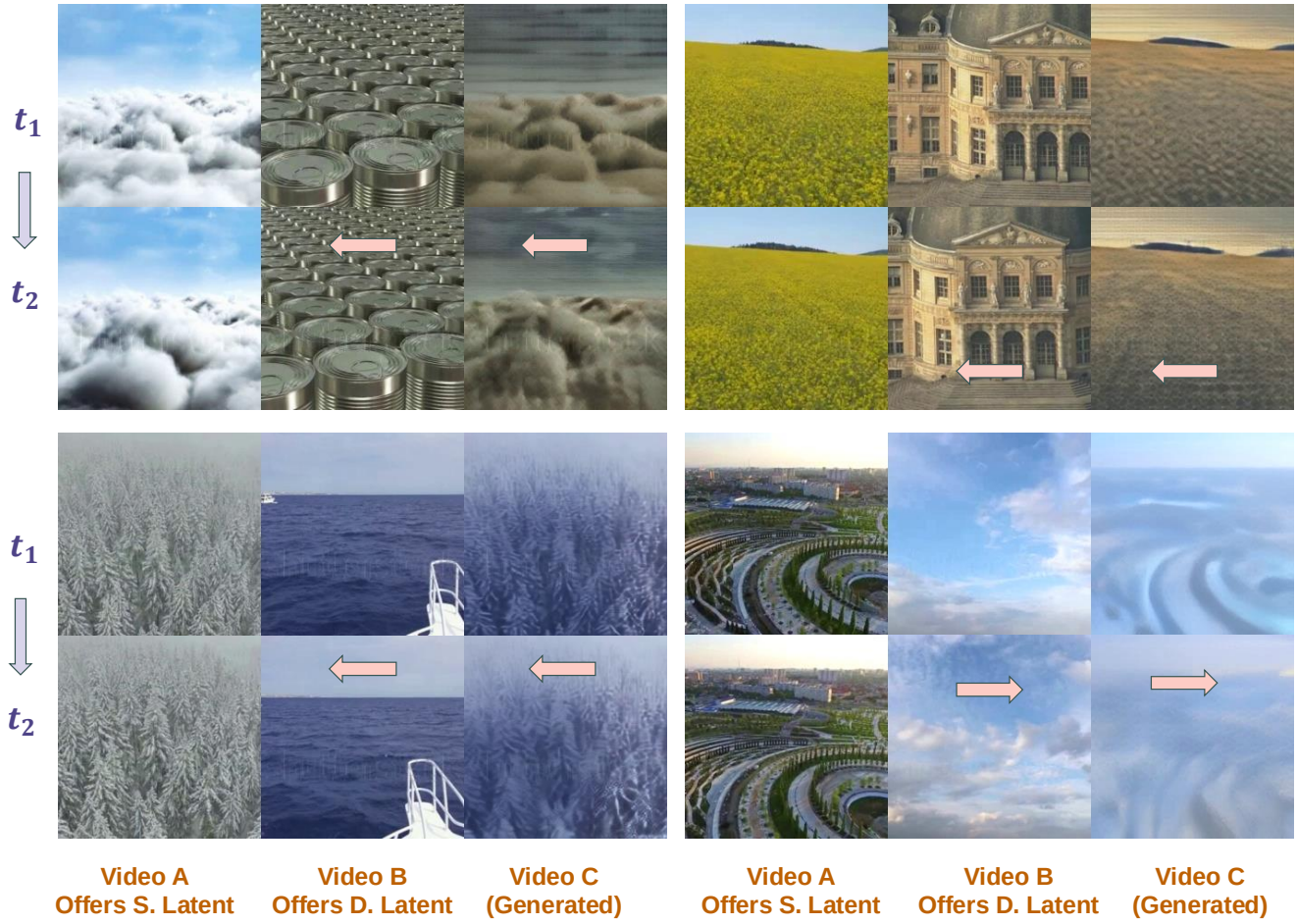

Figure 4. Additional examples of cross-reenactment.

upsampling is performed using Upsample layers with a factor of 2. By varying the number of convolutional layers, latents of different sizes can be generated.

Recommended latent size settings are as Tab. 5. From our experiments, we see that these configurations exhibit minimal performance differences, allowing users to select a setting based on specific requirements.

D.2. Data and Training Details

The key hyperparameters for training data and optimization are summarized as Tab. 6.

D.3. Diffusion Model Details

In Sec. 4.4.3, we describe the design of a diffusion model tailored to the latent space of our VidTwin model. This model adopts the DiT [11] architecture with 18 layers and a hidden state size of 1152. Conditioning is introduced via cross-attention, and for the UCF-101 dataset [14], we use a 256-dimensional vector to encode the class information.

The diffusion process consists of 1000 steps, with DDIM [13] used as the sampling strategy and 50 steps for inference. Classifier-free guidance [4] is applied, where conditioning is randomly dropped in 20% of the samples during training. The classifier-free guidance weight is set to 5 during sampling.

For training, we use the Adam optimizer [6] with $\beta_1 = 0.9, \beta_2 = 0.999$. The learning rate is managed with a Lambda scheduler and includes 10,000 warmup steps. Training is conducted on 8 $\times$ 40G A100 GPUs, with an input configuration of 16 video frames at a resolution of 224.

E. Basics for Diffusion Models and VAE

E.1. Basics for Diffusion Models

Diffusion models are a class of emerging generative models designed to approximate data distributions. The training process consists of two phases: the forward diffusion process and the backward denoising process. Given a data

Table 5. Recommended settings for latent sizes.

Setting	Structure Latent	Dynamics Latent
1	$h_{\boldsymbol{S}} = w_{\boldsymbol{S}} = 7, n_q = 16, d_{\boldsymbol{S}} = 4$	$h_{\boldsymbol{D}} = w_{\boldsymbol{D}} = 7, d_{\boldsymbol{D}} = 8$
2	$h_{\boldsymbol{S}} = w_{\boldsymbol{S}} = 7, n_q = 16, d_{\boldsymbol{S}} = 4$	$h_{\boldsymbol{D}} = w_{\boldsymbol{D}} = 4, d_{\boldsymbol{D}} = 16$
3	$h_{\boldsymbol{S}} = w_{\boldsymbol{S}} = 7, n_q = 12, d_{\boldsymbol{S}} = 4$	$h_{\boldsymbol{D}} = w_{\boldsymbol{D}} = 7, d_{\boldsymbol{D}} = 8$

Table 6. Training Configuration

Parameter	**Value**
Input Video Resolution	224
Input Video Frames	16
Input Video FPS	8
Optimizer	Adam; $\beta_1 = 0.9, \beta_2 = 0.99$
Learning Rate	1.6×10^{-4}
Warmup Steps	5000
Learning Rate Scheduler	Cosine Annealing
$\mathcal{L}_p$	0.05
Weight Decay	0.0001
$\mathcal{L}_{GAN}$	0.05
$\mathcal{L}_{KL}$	0.001
Training Batch Size	6
Training Device	4 × 80G A100 GPUs

point sampled from the real data distribution, $x_0 \sim q(x)$[3], the forward diffusion process gradually adds Gaussian noise to the sample, generating a sequence of noisy samples $x_1, \ldots, x_T$. The noise scales are controlled by a variance schedule $\beta_t \in (0, 1)$, and the density can be expressed as:

$$q(x_t|x_{t-1}) = \mathcal{N}(x_t; \sqrt{1-\beta_t}x_{t-1}, \beta_t \mathbf{I}).$$

Using the reparameterization trick [5], this process allows for sampling at any arbitrary time step in closed form:

$$q(x_t|x_0) = \mathcal{N}(x_t; \sqrt{\overline{\alpha}_t}x_0, \sqrt{1-\overline{\alpha}_t}\mathbf{I}),$$

where $\alpha_t = 1 - \beta_t$ and $\overline{\alpha}_t = \prod_{i=1}^{t} \alpha_i$. From this, it is evident that as $T \to \infty$, x_T converges to an isotropic Gaussian distribution, aligning with the initial condition used during inference.

However, obtaining a closed form for the reverse process $q(x_{t-1}|x_t)$ is challenging. When β_t is sufficiently small, the posterior also approximates a Gaussian distribution. In this case, a model $p_\theta(x_{t-1}|x_t)$ can be trained to approximate these conditional probabilities:

$$p_\theta(x_{t-1}|x_t) = \mathcal{N}(x_{t-1}; \mu_\theta(x_t, t), \Sigma_\theta(x_t, t)),$$

where $\mu_\theta(x_t, t)$ and $\Sigma_\theta(x_t, t)$ are parameterized by a denoising network f_θ, such as a U-Net [12] or a Transformer [15]. By deriving the variational lower bound to optimize the negative log-likelihood of x_0, Ho et al. [5] introduces a simplified DDPM learning objective:

$$\mathcal{L}_{\text{simple}} = \sum_{t=1}^{T} \mathbb{E}_q \left[\|\epsilon_t(x_t, x_0) - \epsilon_\theta(x_t, t)\|^2 \right],$$

where ϵ_t represents the noise added to the original data x_0. In our work, we adopt a simpler architecture that directly predicts x_0, with the loss function defined as:

$$\mathcal{L} = \|x_0 - f_\theta(x_t, t)\|.$$

During inference, the reverse process begins by sampling noise from a Gaussian distribution, $p(x_T) = \mathcal{N}(x_T; \mathbf{0}, \mathbf{I})$, and iteratively denoising it using $p_\theta(x_{t-1}|x_t)$ until x_0 is obtained. DDIM [13] refines this process by ensuring its marginal distribution matches that of DDPM. Consequently, during generation, only a subset of diffusion steps $\{\tau_1, \ldots, \tau_S\}$ is sampled, significantly reducing inference latency.

E.2. Basics for VAE

Variational Autoencoders (VAEs) [7] are a class of generative models that combine probabilistic reasoning with neural networks to learn the underlying distribution of high-dimensional data. A VAE consists of two components: an encoder and a decoder. The encoder maps input data x to a latent variable z characterized by a probabilistic distribution $q(z|x)$, typically parameterized as a Gaussian. The decoder reconstructs the input by sampling from the latent space and generating data through $p(x|z)$.

To ensure that the latent space conforms to a structured prior distribution, typically a standard Gaussian $p(z) = \mathcal{N}(0, I)$, VAEs optimize the Evidence Lower Bound (ELBO):

$$\mathcal{L} = \mathbb{E}_{q(z|x)}[\log p(x|z)] - D_{\text{KL}}(q(z|x)\|p(z)),$$

[3] We follow the notation and derivation process of https://lilianweng.github.io/posts/2021-07-11-diffusion-models.

where the first term represents the reconstruction loss, ensuring that the generated data resembles the input, and the second term is the Kullback-Leibler divergence, which regularizes the latent space.

A key point of VAEs is the reparameterization trick, which facilitates gradient-based optimization by expressing the latent variable z as:

$$z = \mu + \sigma \cdot \epsilon, \quad \epsilon \sim \mathcal{N}(0, I),$$

where μ and σ are outputs of the encoder network.

VAEs have found applications in areas such as image synthesis, data compression, and representation learning due to their ability to generate diverse, high-quality samples while maintaining interpretability of the latent space. In our work, we employ a VAE as the backbone model and introduce two submodules to decouple the video latent representation effectively.

References

[1] Gedas Bertasius, Heng Wang, and Lorenzo Torresani. Is space-time attention all you need for video understanding?, 2021. 4

[2] Liuhan Chen, Zongjian Li, Bin Lin, Bin Zhu, Qian Wang, Shenghai Yuan, Xing Zhou, Xinhua Cheng, and Li Yuan. Od-vae: An omni-dimensional video compressor for improving latent video diffusion model, 2024. 1

[3] Patrick Esser, Robin Rombach, and Björn Ommer. Taming transformers for high-resolution image synthesis, 2021. 3

[4] Jonathan Ho and Tim Salimans. Classifier-free diffusion guidance, 2022. 5

[5] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models, 2020. 6

[6] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization, 2017. 4, 5

[7] Diederik P Kingma and Max Welling. Auto-encoding variational bayes, 2022. 6

[8] Junnan Li, Dongxu Li, Silvio Savarese, and Steven Hoi. Blip-2: Bootstrapping language-image pre-training with frozen image encoders and large language models, 2023. 4

[9] Zongjian Li, Bin Lin, Yang Ye, Liuhan Chen, Xinhua Cheng, Shenghai Yuan, and Li Yuan. Wf-vae: Enhancing video vae by wavelet-driven energy flow for latent video diffusion model, 2024. 1

[10] Bin Lin, Yunyang Ge, Xinhua Cheng, Zongjian Li, Bin Zhu, Shaodong Wang, Xianyi He, Yang Ye, Shenghai Yuan, Liuhan Chen, Tanghui Jia, Junwu Zhang, Zhenyu Tang, Yatian Pang, Bin She, Cen Yan, Zhiheng Hu, Xiaoyi Dong, Lin Chen, Zhang Pan, Xing Zhou, Shaoling Dong, Yonghong Tian, and Li Yuan. Open-sora plan: Open-source large video generation model, 2024. 1

[11] William Peebles and Saining Xie. Scalable diffusion models with transformers, 2023. 3, 5

[12] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation, 2015. 6

[13] Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising diffusion implicit models, 2022. 5, 6

[14] Khurram Soomro, Amir Roshan Zamir, and Mubarak Shah. Ucf101: A dataset of 101 human actions classes from videos in the wild, 2012. 5

[15] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need, 2023. 6

[16] Xinlong Wang, Xiaosong Zhang, Zhengxiong Luo, Quan Sun, Yufeng Cui, Jinsheng Wang, Fan Zhang, Yueze Wang, Zhen Li, Qiying Yu, Yingli Zhao, Yulong Ao, Xuebin Min, Tao Li, Boya Wu, Bo Zhao, Bowen Zhang, Liangdong Wang, Guang Liu, Zheqi He, Xi Yang, Jingjing Liu, Yonghua Lin, Tiejun Huang, and Zhongyuan Wang. Emu3: Next-token prediction is all you need, 2024. 3

[17] Jialong Wu, Shaofeng Yin, Ningya Feng, Xu He, Dong Li, Jianye Hao, and Mingsheng Long. ivideogpt: Interactive videogpts are scalable world models, 2024. 2, 3

[18] Zhuoyi Yang, Jiayan Teng, Wendi Zheng, Ming Ding, Shiyu Huang, Jiazheng Xu, Yuanming Yang, Wenyi Hong, Xiaohan Zhang, Guanyu Feng, Da Yin, Xiaotao Gu, Yuxuan Zhang, Weihan Wang, Yean Cheng, Ting Liu, Bin Xu, Yuxiao Dong, and Jie Tang. Cogvideox: Text-to-video diffusion models with an expert transformer, 2024. 1

[19] Lijun Yu, José Lezama, Nitesh B Gundavarapu, Luca Versari, Kihyuk Sohn, David Minnen, Yong Cheng, Vighnesh Birodkar, Agrim Gupta, Xiuye Gu, et al. Language model beats diffusion–tokenizer is key to visual generation. *arXiv preprint arXiv:2310.05737*, 2023. 2, 4

[20] Sihyun Yu, Weili Nie, De-An Huang, Boyi Li, Jinwoo Shin, and Anima Anandkumar. Efficient video diffusion models via content-frame motion-latent decomposition, 2024. 2, 3

[21] Sijie Zhao, Yong Zhang, Xiaodong Cun, Shaoshu Yang, Muyao Niu, Xiaoyu Li, Wenbo Hu, and Ying Shan. Cv-vae: A compatible video vae for latent generative video models, 2024. 3

[22] Zangwei Zheng, Xiangyu Peng, Tianji Yang, Chenhui Shen, Shenggui Li, Hongxin Liu, Yukun Zhou, Tianyi Li, and Yang You. Open-sora: Democratizing efficient video production for all, 2024. 1