Supplementary Material of Long Video Generation with Time-Agnostic VQGAN and Time-Sensitive Transformer

A Implementation Details

In this section, we provide additional details on our proposed TATS model, including designs to stabilize training 3D-VQGAN with GAN losses, discussions on the undesired temporal dependence induced by the zero padding, ablations on different potential solutions, and description of our interpolation attention.

A.1 Training video VQGAN with GAN losses

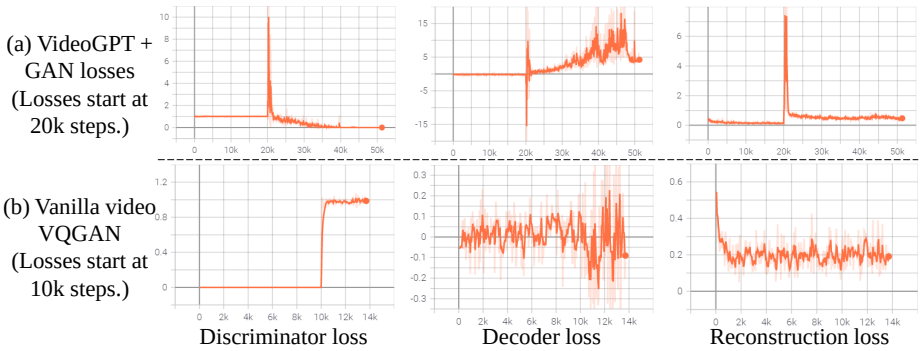


Fig. 10: Training VideoGPT with GAN losses leads to discriminator collapse.

In this section, we describe how we train 3D-VQGAN with GAN losses and clarify several major architecture choices of our proposed vanilla video VQGAN compared with VQGAN [11] and VideoGPT [52]. We find that directly applying GAN losses to VideoGPT leads to severe discriminator collapse. As shown in Figure 10 (a), after adding GAN losses at the 20k step, the discriminator losses saturate quickly and thus provide nonsensical gradient to the decoder. As a result, the decoder (generator) loss and reconstruction loss entirely fall apart. We found that the tricks proposed in VQGAN [11] such as starting GAN losses at a later step and using adaptive weight, do not help the situation. We present several empirical methods we found effective in stabilizing the GAN losses.

First, we find that the axial-attention layers introduced in VideoGPT interact poorly with the GAN losses and exacerbate the collapse, which is also noticed in

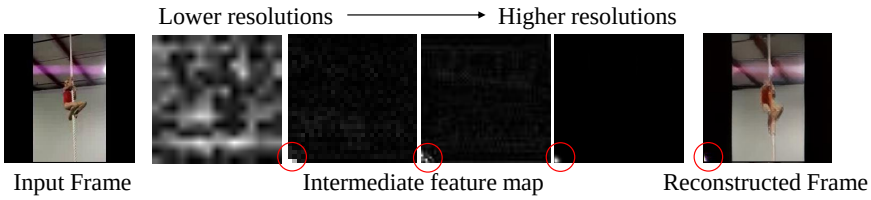
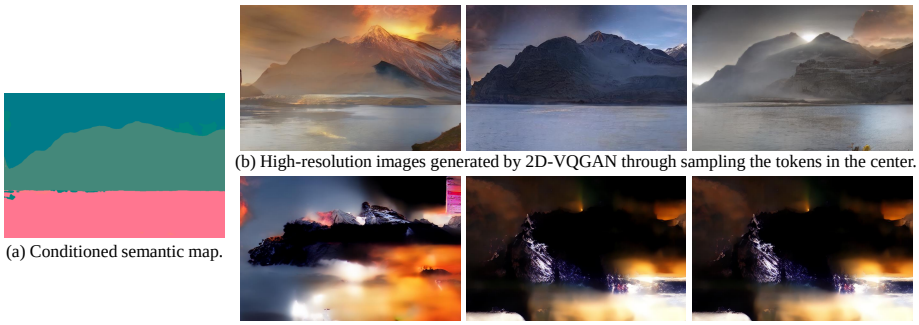


Fig. 11: Blob-shaped artifacts due the the normalization layers in VQGAN.

recent work on training ViT with GAN losses [27]. Therefore, we utilize a pure convolution architecture similar to VQGAN [11] for video compression. Second, a more powerful decoder helps the reconstruction follow the discriminator closely. We doubled the number of feature maps whenever the resolutions are halved, in contrast to VideoGPT where all the layers have a constant number of channels. As a consequence, similar to the previous observation [6], we find that training large VAE models would cause exploded gradients. Furthermore, similar to the proposed solution of gradient skipping [6], we always clip the gradient to have the Euclidean norm equal to 1 during the training. Last, we notice that the blob-shaped artifacts often appear in the reconstruction and exaggerate along with the intermediate feature maps as shown in Figure 11, which was also observed in StyleGANs [21,22] and can be attributed to the normalization layers. This is especially pronounced in the training video VQGAN due to the small batch sizes. We use Synced Batch Normalization as a replace of Group Normalization [49] used in original VQGAN [11] and accumulate gradients across multiple steps and successfully mitigate this issue. Our vanilla video VQGAN can be steadily trained with the GAN losses with the above choices of architecture designs.

A.2 Zero paddings inhibit sliding attention window



(a) Conditioned semantic map.

(b) High-resolution images generated by 2D-VQGAN through sampling the tokens in the center.

(c) High-resolution images generated by 2D-VQGAN through sampling the tokens near the border.

Fig. 12: 2D-VQGAN generates larger images by generating tokens in the center.

Following the intuition in method section, we provide a more detailed discussion on why zero paddings in VQGAN inhibit the usage of the sliding attention window. We aim to show that when zero padding is used, there are barely tokens starting with $\mathbf{z}^{(1:t-1)}$ in the training set. However, this is necessary for the transformer to generate $\mathbf{z}^{(t)}$ using a sliding window.

For simplicity, we absorb the quantization step $\mathbf{q}$ into $f_{\mathcal{E}}(\mathbf{x})$. In order for there to be tokens starting with $\mathbf{z}^{(1:t-1)}$ in the transformer training data, there need to be real video clips that can be encoded in $\mathbf{z}^{(1:t)}$. It is desired that $f_{\mathcal{E}}$ is temporally shift-equivariant given 3D convolutions so that $\mathbf{z}^{(1:t)}$ is the output of the clips slightly shifted from the original position, i.e. $\mathbf{z}^{(1:t)} = f_{\mathcal{E}}(\mathbf{x}^{(d:T+d-1)})$, where d is the compression rate of $f_{\mathcal{E}}$ in the temporal dimension. However, we find that the encoder is *not* temporally shift-equivariant and encodes $\mathbf{x}^{(d:T-1)}$ differently when these frames are positioned at different places, i.e.

$$[f_{\mathcal{E}}(\mathbf{x}^{(0:T-1)})]^{(1:t-1)} \neq [f_{\mathcal{E}}(\mathbf{x}^{(d:T+d-1)})]^{(0:t-2)} \tag{1}$$

To see this theoretically, we consider a shift-equivariant version of $f_{\mathcal{E}}$ by moving all the internal zero paddings to the input. We denote this encoder as $\hat{f}_{\mathcal{E}}$ such that

$$f_{\mathcal{E}}(\mathbf{x}) = \hat{f}_{\mathcal{E}}([\mathbf{0}^N\ \mathbf{x}\ \mathbf{0}^N]), \tag{2}$$

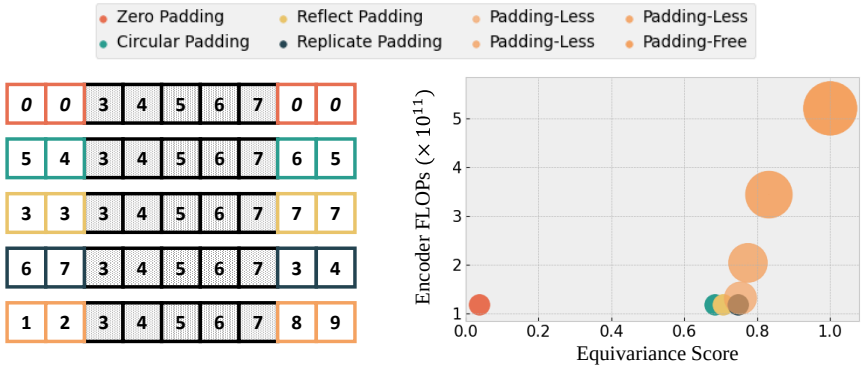
where $[\mathbf{0}^N\ \mathbf{x}\ \mathbf{0}^N]$ is the concatenation of $\mathbf{x}$ with N zero paddings $\mathbf{0} \in \mathbb{R}^{h \times w}$. One can show that the number of paddings needed is $N = \mathcal{O}(Ld)$, where L is the number of convolutional layers in the encoder. Given the shift equivariance of $\hat{f}_{\mathcal{E}}$, we can derive the desired latent tokens for transformer training as

$$z^{(1:t)} = \hat{f}_{\mathcal{E}}([\mathbf{0}^{N-d}\ \mathbf{x}^{(d:T-1)}\ \mathbf{0}^{N+d}]).$$

However, we cannot find such real videos corresponding to $\hat{\mathbf{x}} = [\mathbf{0}^{N-d}\ \mathbf{x}^{(d:T-1)}\ \mathbf{0}^{N+d}]$ as the input to $f_{\mathcal{E}}$ based on the Equation 2 for two reasons. First, according to Equation 2 the input to $f_{\mathcal{E}}$ should be $\hat{\mathbf{x}}^{(N:N+T)} = [\mathbf{x}^{(T+d:T)}\ \mathbf{0}^d]$. There are rare videos whose last d frames are blank in the real datasets. In addition, $\hat{\mathbf{x}}^{(N-d:N)} = \mathbf{x}^{(d:T+d)}$ indicates that real frames are used to pad the input, while $f_{\mathcal{E}}$ only uses zeros. Therefore, we show that no such tokens are starting with $\mathbf{z}^{(1:t-1)}$ in the training set. As a result, the transformer cannot generalize to the sequence starting with $z^{(1:t-1)}$ using a sliding attention window.

This problem occurs in all the generative models that utilize VQVAE and transformers. However, it is often disguised and ignored in previous studies. In practice, the severity of this issue depends on the receptive field, and the relative position of the generated token to the border. In the case of the original VQGAN [11] that uses a sliding window to generate high-resolution images, this issue is disguised as the spatially centered token is always chosen to generate, which is far away from the border and much less affected by the zero-padding given the large spatial size (256). We show in Figure 12 that, when generating the tokens near the border using a sliding window using the high-resolution VQGAN, the quality of images degrades quickly, similar to our observation in video generation. Furthermore, tokens at any position are close to the borders for synthesizing long videos due to the small temporal length (16).

A.3 Quantifying the time agnostics of different padding types



(a) Demonstration of different padding types. The numbers represent the index of frames except that *0* represents a frame of 0s.

(b) Trade-off between equivariance scores and FLOPs. The relative GPU memory utilization is indicated by the volume, and the level of padding-free is indicated by the transparency.

Fig. 13: Demonstration of different padding types and their computational costs as well as effects on the consistency score. Note that when using less or no paddings, extra real paddings are added to the input videos.

Adding real frames makes the VQGAN fully time-agnostic but increases the computational cost. Using other paddings is less effective but brings no overheads. Therefore, it is essential to quantify the temporal dependence to understand the trade-off between the desired property and the cost to achieve it. To that end, we propose an equivariance score as a measure of time agnostic shown in Equation 1, which calculates the percentage of tokens that are identical when the same frames are positioned at the beginning of the end of the clips, which are the two extreme cases that are mostly affected by the paddings from either side:

$$\sum_{i=1}^{t}\sum_{j=0}^{h-1}\sum_{k=0}^{w-1}\frac{\mathbf{1}\left([f_{\mathcal{E}}(\mathbf{x}_{0:T})]_{i,j,k},[f_{\mathcal{E}}(\mathbf{x}_{d:T+d})]_{i-1,j,k}\right)}{t\times h\times w}, \tag{3}$$

where $\mathbf{1}$ is an identity function. We report the mean and standard deviation across 1024 clips using a video VQGAN trained on the UCF-101 dataset across different padding strategies. We visualize the trade-off between the costs and effects on the consistency score in Figure 13. We also partially remove the zero paddings from different numbers of layers to picture the trade-off varies more accurately, where the padding type is called "Padding-Less".

As shown in Figure 13, we find that the more paddings are removed, the more time agnostic the encoder becomes. Note that when removing all the paddings and concatenating enough real frames to the input, we obtain a perfectly time-agnostic encoder that achieves the equivariance score = 1. However, it also significantly increases the memory and computational costs of the training. As for

the other padding types, although the reflect and circular paddings provide more realistic video changes, they could drift further from the real frames and yield a smaller equivariance score than the replicate padding. For example, a walking person is more likely to stop than walk backward in the following frames. We find that the replicate padding, which gives 0.75 consistency score, already resolves the time-dependence issue well in practice. Given the little extra cost it brings, we use replicate paddings and no real frames in our experiments.

A.4 Details of the interpolation attention

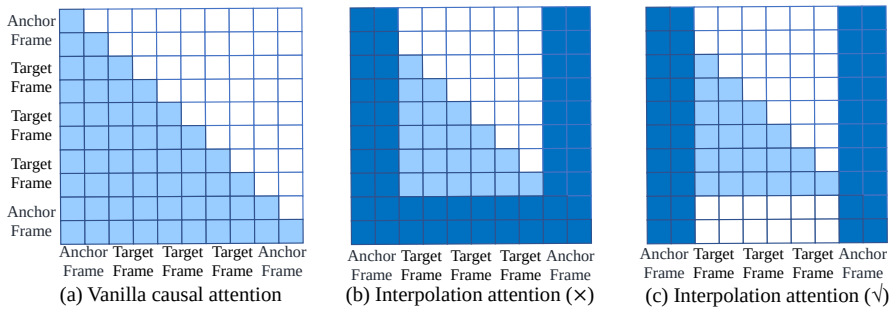


Fig. 14: Illustration of the vanilla causal attention and the interpolation causal attention. For simplification, every frame is assumed to have 2 tokens.

We implement the interpolation transformer based on designed attention called interpolation causal attention, as shown in the right scheme of Figure 14(c). Specifically, the anchor frame (dark) is given during the inference, and the target frame (light) needs to be generated. In the vanilla causal attention shown in Figure 14(a), tokens attend to the tokens *in front of* it. In the interpolation causal attention, tokens attend to *both* the tokens before it and the anchor tokens, which allows acquiring information from the anchor frames at both ends generated by the autoregressive transformer. We want to stress that it is important to not attend the last anchor frame on the frames to be generated like the one in the Figure 14(b), since a multi-layer self-attention will form a shortcut and leak the information of the frames in the middle to themselves and cause the training to collapse.

B Experiment setups

In this section, we provide additional details of our experiments.

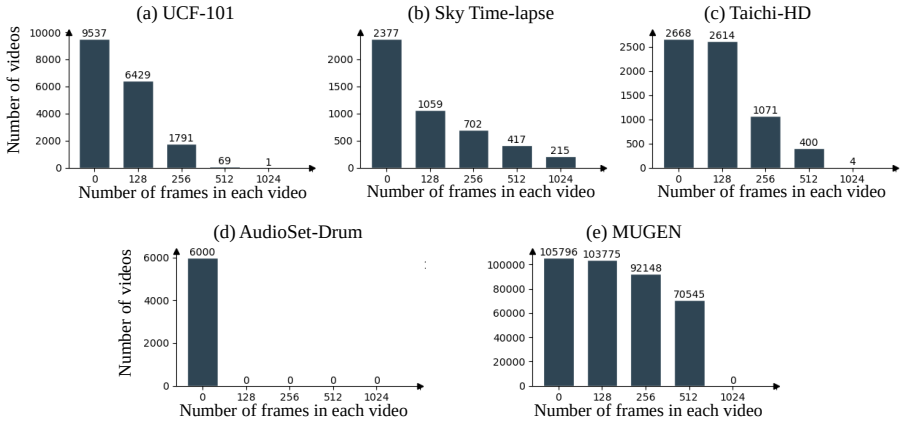


Fig. 15: **Dataset statistics.** Distribution of the number of videos in each dataset that have at least a certain number of frames.

B.1 Dataset and evaluation details

We validate our approach on the UCF-101 [41], Sky Time-lapse [50], Taichi-HD [39], AudioSet-Drum [13], and MUGEN [15] datasets. Below we provide basic descriptions of these datasets and our data processing steps on each of them. We also report the relevant dataset statistics such as the number of long videos and the number of frames per video in Figure 15.

- **UCF-101** is a dataset designed for action recognition which contains 101 classes and 13,320 videos in total. We train our model on its *train split* which contains 9,537 videos following the official splits.[1] We also report number of frames in videos of every class in Figure 16.
- **Sky Time-lapse** contains time-lapse videos that depict the sky under different time and weather conditions. The paper [50] claims that 5,000 videos are collected but only 2,647 are actually released in the official dataset.[2] We follow [53] to train our model on the train split and test using videos from the test split.
- **Taichi-HD** has 2,668 videos in total recording a single person performing Taichi.[3] We follow [53] to sample frames from every 4 frames when training our TATS-base model for a fair comparison. However, training TATS-Hierarchical with this setting would drop 60% of videos for not having enough frames as shown in Figure 15. Therefore we do not skip frames for TATS-Hierarchical.
- **AudioSet-Drum** is a collection of drum kit videos with audio available in the dataset. The train split contains 6,000 videos, and the test split contains

[1] https://www.crcv.ucf.edu/data/UCF101.php
[2] https://github.com/weixiong-ur/mdgan
[3] https://github.com/AliaksandrSiarohin/first-order-model/blob/master/data/taichi-loading/README.md

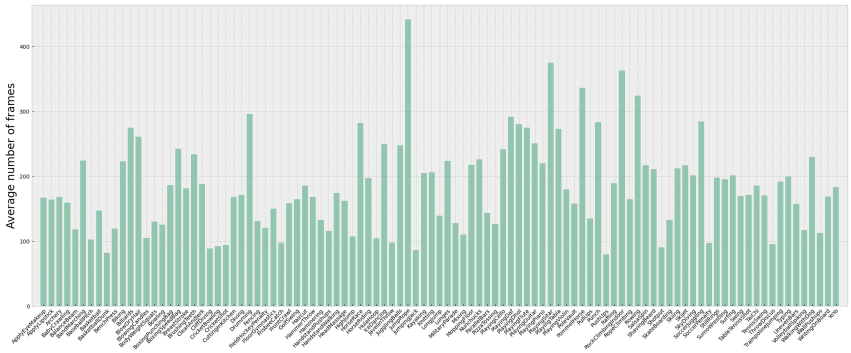


Fig. 16: Average number of frames for each of class in the UCF101 dataset.

1,000 videos. All the video clips have 90 frames. We use the STFT features extracted by [5] as the audio data[4], and follow its evaluation setting to measure the image quality of the 45^{th} frames on the test set.

– **MUGEN** is a dataset containing videos collected from the open-sourced platform game CoinRun [8] by recording the gameplay of trained RL agents. We use their template-based algorithm auto-text, which generates textual descriptions for videos with arbitrary lengths. The train and test splits contain 104,796 and 11,802 videos, respectively, each at 3.2s to 21s (96 to 602 frames) long.

To calculate the VFDs and DVDs, we generate 2,048 and 512 videos for short and long video evaluation, respectively, considering time cost. To calculate the IS, we generate 10,000 videos. To calculate the CCS and ICS for long video evaluation, we also generate 512 videos. We run the evaluations for 10 times and report the standard deviations.

B.2 Training and inference details

VQGAN. Suggested by the VQGAN training recipe [11], we start GAN losses after the reconstruction loss generally converges after $10K$ steps. We adopt a codebook with vocabulary size $K = 16,384$ and embedding size $c = 256$. We do not use the random start trick for codebook embeddings [10,52]. Following [47], we set $\lambda_{\text{rec}} = \lambda_{\text{match}} = 4.0$ and $\lambda_{\text{disc}} = 1.0$. We use the ADAM optimizer [24] with $lr = 3e^{-5}$ and $(\beta_1, \beta_2) = (0.5, 0.9)$. As discussed in Section A.1, we always clip the gradients to have euclidean norm = 1. We train the VQGAN on 8 NVIDIA V100 32GB GPUs with batch size = 2 on each gpu and accumulated batches = 6 for $30K$ steps. Each model usually takes around 57 hours to train.

[4] https://sites.google.com/site/metrosmiles/research/research-projects/sound2sight

Transformers. Both autoregressive and interpolation transformers contain 24 layers, 16 heads, and embedding size = 1024. We use the AdamW optimizer [29] with a base learning rate of $4.5e^{-6}$, where we linearly scale up by the total batch size. We train the transformers on 8 NVIDIA V100 32GB GPUs with batch size = 3 on each GPU until the training loss saturates. Specifically, we train the autoregressive transformers for $500K$ steps as a general setting except that we train TATS-base on the UCF-101 dataset for $1.35M$ as it keeps improving the results. It takes around 10 days to train the transformer models. In practice, we find that the interpolation setting simplifies the problem dramatically, so we only train the interpolation transformers for $30K$ for the best generalization performance. At the inference time, we adopt sampling strategy where temperature $t = 1$, top-k with $k = 2048$, and top-p with $p = 0.80$ as our general setting (for interpolation transformers, a smaller q and k usually produces better results). Sampling 1 video with 1024 frames takes around 30 minutes using TATS-base on a single Quadro P6000 GPU, while TATS-hierarchical reduces this time to 7.5 minutes for autoregressive transformer and 23 seconds for interpolation transformer.

Baselines Apart from those that have been discussed in the related work section, we provide additional descriptions on the baselines we compared with and other GAN-based video generation models in this section. TGAN [37] proposes to generate a fixed number of latent vectors as input to an image generator to synthesize the corresponding frames. MoCoGAN [44] utilizes a RNN to sample motion vectors to synthesize different frames. MoCoGAN-HD [43] leverages a LSTM to predict a trajectory in the latent space of a trained image generator. DVD-GAN [7] adopts a similar architecture as MoCoGAN with a focus on scaling up training. TSB [34] further improves MoCoGAN by mixing information of adjacent frames using an operation called temporal shift. TGAN2 [38] proposes to divide the generator into multiple small sub-generators and introduces a subsampling layer that reduces the frame rate between each pair of consecutive sub-generators. HVG [4] further introduces a hierarchical pipeline to interpolate and upsample low-resolution and low-frame-rate videos gradually. ProgressiveVGAN [1] extends ProgressiveGAN [20] to video generation by simultaneously generating in the temporal direction progressively. LDVD-GAN [19] analyzes the discriminators used in video GANs and designs improved discriminators considering the convolution kernel dimensionality. CCVS [26] utilizes VQVAE to compress frames for training transformer and a flow module to improve temporal consistency. VideoGPT [52] further leverages 3D convolution and axial self-attention in the VQVAE to encode the temporal dimension as well. DIGAN [53] first generalized the idea of implicit neural representations to video generation by decomposing the network weights for the spatial and temporal coordinates. StyleGAN-v [40] maps a continuous temporal positional embedding to the input feature map of the StyleGAN.

B.3 Comparison of the computational costs

We compute the time to generate a single video of 1024 frames using different methods in Table 1. The slow inference speed of autoregressive models is often criticized. However, among all the methods that build on VQVAE and transformer framework, our TATS-hierarchical is the fastest – 1/3 the time of CCVS and 1/5 the time of VideoGPT. Accelerating autoregressive transformers is an active research area. Further improvements can be achieved by methods such as sparse attention, which we leave for future exploration.

Table 1: Time for generating a 1024-frame video.

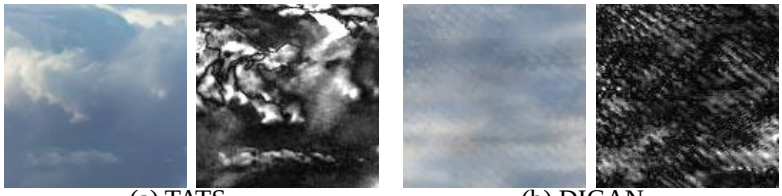
DIGAN	MoCoGAN-HD	CCVS	VideoGPT	TATS-base	TATS-hierarchical
4.2 sec	28.5 sec	22 min	42 min	30 min	7.5 min

C Additional results on long video generation

More videos with repeated actions and smooth transitions. Video generation results with repeated actions can be widely seen in other UCF-101 classes as well as shown in Figure 18. In addition to the generated sky videos with smooth transitions, we also show that there are such cases in UCF-101 and Taichi-HD datasets in Figure 19. We can see in the example of UCF-101 *Sky Diving* video that the transition proceeds clearly with unrealistic content as only limited data are available per class. In the example of Taichi video, we can see that the color of the pants transforms smoothly. However, there is still unrealistic content in such videos. Therefore, we argue that a split of content and motion generation could be helpful in these cases [44,53,43].

Failure cases. Errors could occur and accumulate during the application of sliding window attention. We show several typical failure examples in Figure 20. In the example of *Boxing Punching Bag*, an error occurs at around 300 frames, and the general quality of the video deteriorates after that. Repeated tokens are generated in the deteriorated area, which is a commonly observed issue in sequence generation [16,18]. We also find that this kind of issue happens more often in the areas which contain large motions. The video quality could also degrade quickly if the thematic event of the video has a clear end. For example, the generated *Long Jump* video quickly transits to the other scene and degenerates when the person finishes the action.

Additional comparison with DIGAN. We find that DIGAN presents clear quality degradation on the Sky dataset. To show this, we calculate the average difference between the 1000^{th} and 1024^{th} frames generated by DIGAN and our method in Figure 17. DIGAN results show periodic artifacts induced by



(a) TATS (b) DIGAN

Fig. 17: The 1000^{th} generated frame and the average pixel difference w.r.t. the 1024^{th} frame. DIGAN generations contain clear periodic artifacts.

the sinusoidal positional encodings (repeated changes in the diagonal direction). However, this is not reflected in the FVD metric, which we conjecture is due to the domain gap with respect to the Kinetics dataset which the I3D model is trained on. So to quantify this, we conduct human evaluation to compare TATS with DIGAN. We randomly sampled 100 generated videos for each method and cropped the videos to the last 128 frames. We showed raters a pair of videos (one from each method), and asked them to select the video with fewer artifacts. Each video pair was evaluated by around 5 raters. TATS was chosen over DIGAN 67% of the time (355 vs. 175). Under a binomial test, TATS is statistically significantly better than DIGAN with confidence > 0.95.

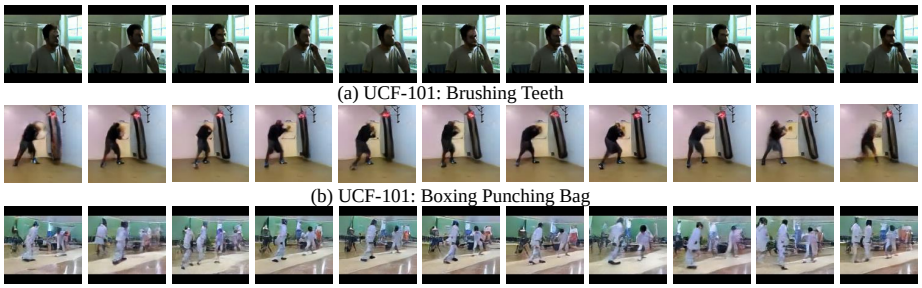
D Additional related works

The implicit bias of zero padding. The positional inductive bias introduced by zero padding has drawn increasing attention recently [17,23,51,2]. In most cases, such inductive bias is helpful in classification [17], generation [51], or object detection [2]. Our paper observes a case where this inductive bias is harmful.

Video prediction. Video prediction aims at modeling the transformation between frames and predicting future frames given real frames [36,42,12,45,25,35]. For instance, [45] divides the frames into patches, calculates the affine transformation between temporally adjacent patches, and predicts such affine transformation to be applied to the most recent frame. [30] predicts videos of semantic maps and argues that autoregressive models lead to error propagation when more frames are predicted while being more accurate in the semantic segmentation space. Flow-based models [25] and ODE-based models [35] have also been used for video prediction. Different from video prediction, video generation focus on producing videos from noise. When applying to long videos, one substantial difference is that video prediction starts from a real frame while video generation starts from a generated frame. Some video generation models fall into the middle part [26] that predicts future frames given frames generated by an image generator. We have shown that these models also suffer from quality degradation.

Conditional video generation. Text-conditioned video generation has been studied in multiple papers. SyncDraw [33] first proposes to combine VAE and

RNN for video generation while conditioning on texts. T2V [28] uses CVAE to generate the gist then GANs with 3D convolutions to generate fixed-length low-resolution videos. The text is encoded in a convolutions filter to process the gist. TFGAN [3] proposes a multi-scale text-conditioning discriminator and follows MoCoGAN [44] to use an RNN to model the temporal information. IRC-GAN [9] proposes to use a recurrent transconvolutional generator and mutual-information introspection to generate videos based text. Craft [14] sequentially composes a scene layout and retrieves entities from a video database to create complex scene videos. GODIVA [48] relies on a pretrained VQVAE model to produce video frame representations and autoregressively predicts the video representations based on the input text representations and former frames. Each element's prediction in the current frame's representation attends to the previous row, column, and time. As for audio-conditioned video generation, Sound2Sight [5] proposes to model the previous frames and audios with a multi-head audio-visual transformer and predicts future frames with a prediction network based on sequence-to-sequence architecture. Vougioukas et al. [46] studies speech-driven facial animation, which aims at generating face videos given the speech audios. They propose a framework based on RNNs and a frame generator. Some conditional generation frameworks are also able to generate long videos either when minimal changes occur in the video scenes [32] or strong conditional information is given such as segmentation maps [47,31]. Our paper focuses on more realistic and complex videos with weak or no conditional information available.

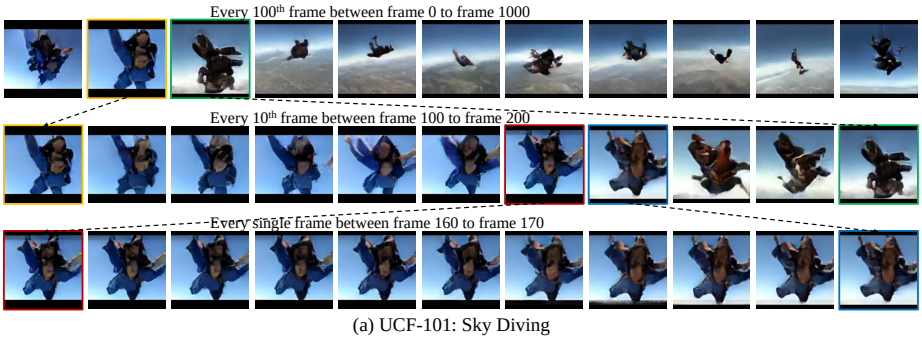


(a) UCF-101: Brushing Teeth

(b) UCF-101: Boxing Punching Bag

(c) UCF-101: Fencing

Fig. 18: More class-conditional generation results of UCF-101 videos with $1,024$ frames that contains repeated action.



(a) UCF-101: Sky Diving

(b) Taichi-HD

Fig. 19: Unconditional and class-conditional generation results of *Sky Diving* and Taichi-HD videos with $1,024$ frames that contain smooth transitions.

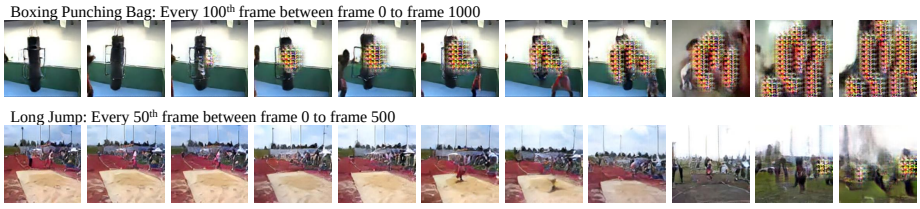


Fig. 20: Failure cases of class-conditional generation results of long videos on the UCF-101 dataset.

References

1. Acharya, D., Huang, Z., Paudel, D.P., Van Gool, L.: Towards high resolution video generation with progressive growing of sliced wasserstein gans. arXiv preprint arXiv:1810.02419 (2018) 8
2. Alsallakh, B., Kokhlikyan, N., Miglani, V., Yuan, J., Reblitz-Richardson, O.: Mind the pad – CNNs can develop blind spots. In: ICLR (2021) 10
3. Balaji, Y., Min, M.R., Bai, B., Chellappa, R., Graf, H.P.: Conditional gan with discriminative filter generation for text-to-video synthesis. In: IJCAI (2019) 11
4. Castrejon, L., Ballas, N., Courville, A.: Hierarchical video generation for complex data. arXiv preprint arXiv:2106.02719 (2021) 8
5. Chatterjee, M., Cherian, A.: Sound2sight: Generating visual dynamics from sound and context. In: ECCV (2020) 7, 11
6. Child, R.: Very deep vaes generalize autoregressive models and can outperform them on images. In: ICLR (2020) 2
7. Clark, A., Donahue, J., Simonyan, K.: Adversarial video generation on complex datasets. arXiv preprint arXiv:1907.06571 (2019) 8
8. Cobbe, K., Klimov, O., Hesse, C., Kim, T., Schulman, J.: Quantifying generalization in reinforcement learning. In: ICML (2019) 7
9. Deng, K., Fei, T., Huang, X., Peng, Y.: Irc-gan: Introspective recurrent convolutional gan for text-to-video generation. In: IJCAI (2019) 11
10. Dhariwal, P., Jun, H., Payne, C., Kim, J.W., Radford, A., Sutskever, I.: Jukebox: A generative model for music. arXiv preprint arXiv:2005.00341 (2020) 7
11. Esser, P., Rombach, R., Ommer, B.: Taming transformers for high-resolution image synthesis. In: CVPR (2021) 1, 2, 3, 7
12. Finn, C., Goodfellow, I., Levine, S.: Unsupervised learning for physical interaction through video prediction. NeurIPS (2016) 10
13. Gemmeke, J.F., Ellis, D.P., Freedman, D., Jansen, A., Lawrence, W., Moore, R.C., Plakal, M., Ritter, M.: Audio set: An ontology and human-labeled dataset for audio events. In: ICASSP (2017) 6
14. Gupta, T., Schwenk, D., Farhadi, A., Hoiem, D., Kembhavi, A.: Imagine this! scripts to compositions to videos. In: ECCV (2018) 11
15. Hayes, T., Zhang, S., Yin, X., Pang, G., Sheng, S., Yang, H., Ge, S., Hu, Q., Parikh, D.: Mugen: A playground for video-audio-text multimodal understanding and generation. arXiv preprint arXiv:2204.08058 (2022) 6
16. Holtzman, A., Buys, J., Du, L., Forbes, M., Choi, Y.: The curious case of neural text degeneration. In: ICLR (2020) 9
17. Islam, M.A., Jia, S., Bruce, N.D.: How much position information do convolutional neural networks encode? In: ICLR (2019) 10
18. Jiang, S., de Rijke, M.: Why are sequence-to-sequence models so dull? understanding the low-diversity problem of chatbots. In: EMNLP Workshop (2018) 9
19. Kahembwe, E., Ramamoorthy, S.: Lower dimensional kernels for video discriminators. Neural Networks (2020) 8
20. Karras, T., Aila, T., Laine, S., Lehtinen, J.: Progressive growing of gans for improved quality, stability, and variation. In: ICLR (2018) 8
21. Karras, T., Laine, S., Aila, T.: A style-based generator architecture for generative adversarial networks. In: CVPR (2019) 2
22. Karras, T., Laine, S., Aittala, M., Hellsten, J., Lehtinen, J., Aila, T.: Analyzing and improving the image quality of stylegan. In: CVPR (2020) 2

23. Kayhan, O.S., Gemert, J.C.v.: On translation invariance in cnns: Convolutional layers can exploit absolute spatial location. In: CVPR (2020) 10
24. Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization. ICLR (2015) 7
25. Kumar, M., Babaeizadeh, M., Erhan, D., Finn, C., Levine, S., Dinh, L., Kingma, D.: Videoflow: A conditional flow-based model for stochastic video generation. In: ICLR (2019) 10
26. Le Moing, G., Ponce, J., Schmid, C.: Ccvs: Context-aware controllable video synthesis. NeurIPS (2021) 8, 10
27. Lee, K., Chang, H., Jiang, L., Zhang, H., Tu, Z., Liu, C.: Vitgan: Training gans with vision transformers. arXiv preprint arXiv:2107.04589 (2021) 2
28. Li, Y., Min, M., Shen, D., Carlson, D., Carin, L.: Video generation from text. In: AAAI (2018) 11
29. Loshchilov, I., Hutter, F.: Decoupled weight decay regularization. In: International Conference on Learning Representations (2018) 8
30. Luc, P., Neverova, N., Couprie, C., Verbeek, J., LeCun, Y.: Predicting deeper into the future of semantic segmentation. In: ICCV (2017) 10
31. Mallya, A., Wang, T.C., Sapra, K., Liu, M.Y.: World-consistent video-to-video synthesis. In: ECCV (2020) 11
32. Menapace, W., Lathuilière, S., Tulyakov, S., Siarohin, A., Ricci, E.: Playable video generation. In: CVPR (2021) 11
33. Mittal, G., Marwah, T., Balasubramanian, V.N.: Sync-draw: Automatic video generation using deep recurrent attentive architectures. In: MM (2017) 10
34. Munoz, A., Zolfaghari, M., Argus, M., Brox, T.: Temporal shift gan for large scale video generation. In: WACV (2021) 8
35. Park, S., Kim, K., Lee, J., Choo, J., Lee, J., Kim, S., Choi, Y.: Vid-ode: Continuous-time video generation with neural ordinary differential equation. In: AAAI. AAAI (2021) 10
36. Ranzato, M., Szlam, A., Bruna, J., Mathieu, M., Collobert, R., Chopra, S.: Video (language) modeling: a baseline for generative models of natural videos. arXiv preprint arXiv:1412.6604 (2014) 10
37. Saito, M., Matsumoto, E., Saito, S.: Temporal generative adversarial nets with singular value clipping. In: ICCV (2017) 8
38. Saito, M., Saito, S., Koyama, M., Kobayashi, S.: Train sparsely, generate densely: Memory-efficient unsupervised training of high-resolution temporal gan. IJCV (2020) 8
39. Siarohin, A., Lathuilière, S., Tulyakov, S., Ricci, E., Sebe, N.: First order motion model for image animation. NeurIPS (2019) 6
40. Skorokhodov, I., Tulyakov, S., Elhoseiny, M.: Stylegan-v: A continuous video generator with the price, image quality and perks of stylegan2. arXiv preprint arXiv:2112.14683 (2021) 8
41. Soomro, K., Zamir, A.R., Shah, M.: Ucf101: A dataset of 101 human actions classes from videos in the wild. arXiv preprint arXiv:1212.0402 (2012) 6
42. Srivastava, N., Mansimov, E., Salakhudinov, R.: Unsupervised learning of video representations using lstms. In: ICML (2015) 10
43. Tian, Y., Ren, J., Chai, M., Olszewski, K., Peng, X., Metaxas, D.N., Tulyakov, S.: A good image generator is what you need for high-resolution video synthesis. In: ICLR (2021) 8, 9
44. Tulyakov, S., Liu, M.Y., Yang, X., Kautz, J.: Mocogan: Decomposing motion and content for video generation. In: CVPR (June 2018) 8, 9, 11

45. Van Amersfoort, J., Kannan, A., Ranzato, M., Szlam, A., Tran, D., Chintala, S.: Transformation-based models of video sequences. arXiv preprint arXiv:1701.08435 (2017) 10
46. Vougioukas, K., Petridis, S., Pantic, M.: End-to-end speech-driven facial animation with temporal gans. BMVC (2018) 11
47. Wang, T.C., Liu, M.Y., Zhu, J.Y., Liu, G., Tao, A., Kautz, J., Catanzaro, B.: Video-to-video synthesis. In: NeurIPS (2018) 7, 11
48. Wu, C., Huang, L., Zhang, Q., Li, B., Ji, L., Yang, F., Sapiro, G., Duan, N.: Godiva: Generating open-domain videos from natural descriptions. arXiv preprint arXiv:2104.14806 (2021) 11
49. Wu, Y., He, K.: Group normalization. In: ECCV (2018) 2
50. Xiong, W., Luo, W., Ma, L., Liu, W., Luo, J.: Learning to generate time-lapse videos using multi-stage dynamic generative adversarial networks. In: CVPR (2018) 6
51. Xu, R., Wang, X., Chen, K., Zhou, B., Loy, C.C.: Positional encoding as spatial inductive bias in gans. In: CVPR (2021) 10
52. Yan, W., Zhang, Y., Abbeel, P., Srinivas, A.: Videogpt: Video generation using vq-vae and transformers. arXiv preprint arXiv:2104.10157 (2021) 1, 7, 8
53. Yu, S., Tack, J., Mo, S., Kim, H., Kim, J., Ha, J.W., Shin, J.: Generating videos with dynamics-aware implicit generative adversarial networks. In: ICLR (2021) 6, 8, 9