Latent Video Transformer

Ruslan Rakhimov[1,*], Denis Volkhonskiy[1,*], Alexey Artemov[1], Denis Zorin[1,2] and Evgeny Burnaev[1]
[1]*Skolkovo Institute of Science and Technology, Moscow, Russia*
[2]*New York University, New York, U.S.A.*

Keywords: Video Generation, Deep Learning, Generative Adversarial Networks.

Abstract: The video generation task can be formulated as a prediction of future video frames given some past frames. Recent generative models for videos face the problem of high computational requirements. Some models require up to 512 Tensor Processing Units for parallel training. In this work, we address this problem via modeling the dynamics in a latent space. After the transformation of frames into the latent space, our model predicts latent representation for the next frames in an autoregressive manner. We demonstrate the performance of our approach on BAIR Robot Pushing and Kinetics-600 datasets. The approach tends to reduce requirements to 8 Graphical Processing Units for training the models while maintaining comparable generation quality.

1 INTRODUCTION

Video prediction and generation is an important problem with a lot of down-stream applications: self-driving, anomaly detection, timelapse generation (Nam et al., 2019), animating landscape (Endo et al., 2019) etc. The task is to generate the most probable future frames given several initial ones.

Recent advances in generative learning allow generation of realistic objects with high quality: images, text, and speech. However, video generation is still a very challenging task. Even for short videos (16 frames) of low resolution, neural networks require up to 512 Tensor Processing Units (TPUs) (Luc et al., 2020) for parallel training. Despite this, the quality of the generated video remains low.

In this work, we introduce a Latent Video Transformer. We combine the idea of representation learning and recurrent video generation. Instead of working in pixel space, we conduct the generation process in the latent space. Our model tends to significantly reduce computational requirements without significant deterioration in quality.

The key novelty in our model is the usage of a discrete latent space (van den Oord et al., 2017). It allows us to represent each frame as a set of indices. Thanks to discrete representation we can use autoregressive generative models and other approaches from natural language processing.

[*]Equal contribution

We analyzed the results of our model on two datasets: BAIR Robot Pushing (Ebert et al., 2017) and Kinetics 600 (Carreira et al., 2018). On both datasets, we obtained quality comparable to state-of-the-art methods.

To summarize, our contributions are as follows:

- We proposed a new autoregressive model for video generation, that works in the latent space rather than pixel space;
- We reduced computational requirements comparing to previously proposed methods.

2 RELATED WORK

2.1 Video Generation

Video generation is a long-standing problem. One can formulate it in different ways. Future video prediction, unconditional video synthesis or video to video translation (Wang et al., 2018a; Pan et al., 2019; Wang et al., 2019). There exist other rare setups like generating video from one image (Shaham et al., 2019). Video prediction and unconditional video synthesis have been addressed for a long time and the solutions include recurrent models (Finn et al., 2016a; Wang et al., 2018b; Wang et al., 2018c; Byeon et al., 2018), VAE-based models (Denton and Fergus, 2018a; Denton et al., 2017; Lee et al., 2018; Hsieh et al., 2018; Denton and Fergus, 2018b), autoregressive models

Rakhimov, R., Volkhonskiy, D., Artemov, A., Zorin, D. and Burnaev, E.
Latent Video Transformer.
DOI: 10.5220/0010241801010112
In *Proceedings of the 16th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications (VISIGRAPP 2021) - Volume 5: VISAPP*, pages 101-112
ISBN: 978-989-758-488-6

(Ranzato et al., 2014; Srivastava et al., 2015; Xingjian et al., 2015; Kalchbrenner et al., 2017; Weissenborn et al., 2019; Ho et al., 2019), normalizing flows (Kumar et al., 2019), GANs (Mathieu et al., 2015; Vondrick et al., 2016; Saito et al., 2017; Tulyakov et al., 2018; Acharya et al., 2018; Saito and Saito, 2018; Clark et al., 2019; Luc et al., 2020) and optical flow (Patraucean et al., 2015; Ohnishi et al., 2018).

In the first attempts, fully deterministic models have been used. Later generative models were applied. Similar to image generation, video generative models inherited similar benefits and drawbacks. Variational autoencoder (VAE)-based models try to model videos in latent space but produce blurry results. Later GANs were applied to address those issues, but they suffer from mode-dropping behavior. Some models (Lee et al., 2018) try to combine VAE and GANs.

The recent state-of-the-art approaches DVD-GAN-FP (Clark et al., 2019) and its modification TRIVD-GAN-FP (Luc et al., 2020) follow the success of BigGAN (Brock et al., 2018). They use 2D residuals blocks for independent frames prediction with Convolutional Gated Recurrent units between frames.

Another branch of generative models is autoregressive (AR) models. PixelCNN (Van den Oord et al., 2016; Salimans et al., 2017) generates new images by producing a new pixel value conditioning on previous (seen, already generated) ones in the raster-scan order. Later, PixelSnail (Chen et al., 2017) increased the quality of generated samples by utilizing an attention mechanism. Recently, such an approach was applied to video generation (Kalchbrenner et al., 2017; Weissenborn et al., 2019). The latest work, Video Transformer (Weissenborn et al., 2019), utilizes autoregressive video generation along with subscaling (Menick and Kalchbrenner, 2018) and attention mechanism (Vaswani et al., 2017).

The main challenge of AR models is a generation speed. Even though the latest AR model (VideoTransformer (Weissenborn et al., 2019)) applied the subscaling mechanism (Menick and Kalchbrenner, 2018), introduced block-local attention, the generation speed is still quite slow.

Also, DVD-GAN-FP, TRIVD-GAN-FP, Video Transformer (VT) — they are all suffering from significant resource requirements, even for generating low-resolution video with frames of size 64x64. For instance, VT needs 128 TPUs and 1M steps for training.

Our work is in the field of autoregressive models and follows the setup of VideoTransformer (Weissenborn et al., 2019). The key novelty is that we mitigate GPU memory consumption and accelerate inference speed by working in a discrete latent space.

2.2 Discrete Latent Space

Autoencoder is a neural network trained in a self-supervised manner. Autoencoder takes an input (image, text, audio, etc.) and transfers (encodes) it into a more compact latent representation. The learning consists of finding such an encoder and decoder so that we can encode and decode the input as closely as possible.

Usually, latent space is continuous. However, some works like VQ-VAE (van den Oord et al., 2017), VQ-VAE2 (Razavi et al., 2019) model it as discrete with a categorical distribution inside. They demonstrated good reconstruction and generation quality. As generating from uniform distribution directly produced inferior results, autoregressive models were applied to learn the prior inside the latent space.

We follow this pipeline but for video modeling. First, encoding conditioning frames to discrete latent space, generate new (latent) frames using an autoregressive model, and decode the generated frames back to pixel space. Parallel to this work, a similar pipeline was applied to audio generation (Jukebox (Dhariwal et al., 2020)).

Discrete latent space also occurred to be useful in other works. Discrete quantization was added to a discriminator in GAN (Zhao et al., 2020). (Kaiser et al., 2018) uses discrete variables to increase the speed of the autoregressive model for neural machine translation.

3 LATENT VIDEO TRANSFORMER

Consider a video X to be a sequence of T frames $\{x_t\}_{t=1}^{T}$. Each frame $x_t \in \mathbb{R}^{H \times W \times 3}$ has height H, width W and 3 RGB channels. Given the first T_0 frames, the goal is to generate the remaining $T - T_0$ frames. For this purpose, we propose a model: Latent Video Transformer (LVT). In general, it consists of two parts: a frame autoencoder and an autoregressive generative model.

We use the frame autoencoder to learn a compact latent representation so that we can transfer the task of video modeling from the pixel space to the latent space. The recurrent model is then used for generating new frames. The key novelty compared to existing models that operate in the latent space is a discrete structure of the latent space. Discrete representation helps us to use autoregressive generative models and other approaches, tailored for working for discrete data, e.g. those used for natural language processing.

3.1 Frame Autoencoder

We train a frame autoencoder to transfer individual images (frames) to latent space. The particular choice of the autoencoder is VQ-VAE (van den Oord et al., 2017) — variational autoencoder with discrete latent space.

VQ-VAE (see Fig. 1) learns to encode an input image $x \in \mathbb{R}^{H \times W \times 3}$ using a codebook $e \in \mathbb{R}^{K \times D}$, where K denotes the codebook size (i.e., latent space is K-way categorical) and D represents the size of an embedding in the codebook.

In general VQ-VAE consists of an *encoder* which encodes the image into more compact representation $E(x) = z_e(x) \in \mathbb{R}^{h \times w \times D}$; *a bottleneck*, that discretizes each pixel by mapping it to its nearest embedding e_i from the codebook and produces $z(x) \in [K]^{h \times w \times 1}$; *a decoder D* takes as input discrete latent codes $z(x)$, maps indexes to corresponding embeddings, and decodes the result of mapping $z_q(x) \in \mathbb{R}^{h \times w \times D}$ back to input pixel space.

VQ-VAE is trained with the following objective:

$$L = \|x - D(z_q(x))\|^2 + \|z_e(x) - \mathrm{sg}[e]\|^2, \quad (1)$$

where sg[] is the stop gradient operator, which returns its argument during the forward pass and zero gradients during backward pass. The first term is a reconstruction loss, and the second term is regularization term to make the encodings less volatile. We use EMA updates over the codebook variables.

Decomposed Vector Quantization. If the size K of the codebook e is large, then the model tends to index collapse. It means that some embedding vector e_i is close to a lot of encoders outputs. In this case, it receives a strong update signal (Kaiser et al., 2018). As a result, the model would use only a limited number of vectors from e.

In order to overcome this issue, we exploit Sliced Vector Quantization (Kaiser et al., 2018). We introduce several codebooks $\{e^j \in \mathbb{R}^{K \times D/n_c}\}_{j=1}^{n_c}$ and split the output of encoder $z_e(x)$ along the channel dimension into n_c parts with individual codebook per each part (see Figure 1). The output from discretization bottleneck in this case is $z \in [K]^{h \times w \times n_c}$.

3.2 Latent Video Generator

Frame encoder transforms the first T_0 frames to a discrete representation $Z_0 \in [K]^{T_0 \times h \times w \times n_c}$.

The autoregressive model is used to generate new $T - T_0$ frames conditioned on Z_0. As such model, we use the Video Transformer (Weissenborn et al., 2019), autoregressive video generative model, but apply it in the latent space in contrast to the pixel space in the original paper. Next, we describe the architecture of a video transformer. We refer to a latent representation of a video as *latent video* and individual elements of it as *latent frames* and *pixels*. For exhaustive architecture details, we refer the reader to the original paper (Weissenborn et al., 2019).

The model takes as input a tensor $Z \in [K]^{T \times h \times w \times n_c}$ and primes the generation process on first T_0 given latent frames, i.e. $Z_{:T_0,:,:,:} = Z_0$. The other latent frames could be randomly filled as the generation process is conditioned only on already generated or priming pixels.

First, the model utilizes the idea of subscaling (Menick and Kalchbrenner, 2018): let's generate a latent video as a sequence of *non-overlapping* slices. After defining a subscale factor $\mathbf{s} = (s_t, s_h, s_w)$, it divides latent video into $s = s_t s_h s_w$ slices of size $T/s_t \times h/s_h \times w/s_w$. The generation process happens slice by slice, pixel by pixel inside one slice, channel by channel for one pixel:

$$p(Z) = \prod_{i=0}^{Thw-1} \prod_{k=0}^{n_c-1} p\left(Z^k_{\pi(i)} | Z_{\pi(<i)}, Z^{<k}_{\pi(i)}\right) \quad (2)$$

Pixels in each slice $Z_{(a,b,c)}$ are generated in raster-scan order and slices are generated in the subscale order: $Z_{(0,0,0)}, Z_{(0,0,1)}, \ldots, Z_{(s_t-1, s_h-1, s_w-1)}$.

The model follows the original Transformer (Vaswani et al., 2017) and consists of an encoder and a decoder. To generate a new pixel value inside slice $Z_{(a,b,c)}$, firstly, the encoder outputs the representation of already generated slices $Z_{<(a,b,c)}$. This representation goes to the decoder, which mixes it with a representation of already generated pixels inside a current slice $Z_{(a,b,c)}$. This autoregressive order is preserved by padding input latent video inside the encoder, and masking used in convolutions and attention inside the decoder. After generating a new pixel value, we replace the respective padding with the generated output and repeat the generation process recursively. The generation process in case of spatiotemporal ($s_t > 0, s_h > 0, s_w > 0$) subscaling can be seen at Fig. 2.

Finally, when the generation process is done, the latent frame decoder takes as input $Z \in [K]^{T \times h \times w \times n_c}$ (now all values are valid), maps it to already learned embeddings $Z_q \in \mathbb{R}^{T \times h \times w \times D}$ and decodes it back frame by frame to an original pixel space $X \in \mathbb{R}^{T \times H \times W \times 3}$.

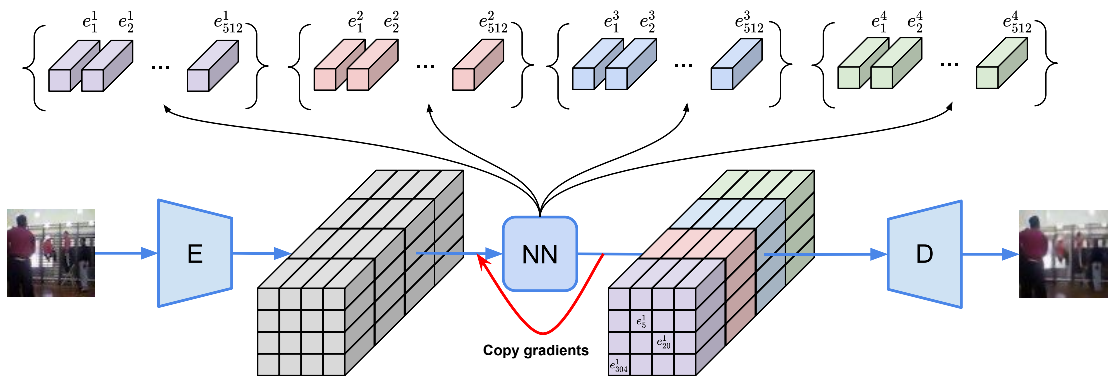
e^1_1 e^1_2 ... e^1_{512}
e^2_1 e^2_2 ... e^2_{512}
e^3_1 e^3_2 ... e^3_{512}
e^4_1 e^4_2 ... e^4_{512}
E
NN
D
Copy gradients
e^1_5
e^1_{20}
e^1_{304}

Figure 1: Frame autoencoder architecture. An input image is passed through the encoder and split along the channel dimension into $n_c = 4$ parts. Then we map pixels in each part to the corresponding nearest embeddings in the codebook. These nearest embeddings are then passed as an input to the decoder.

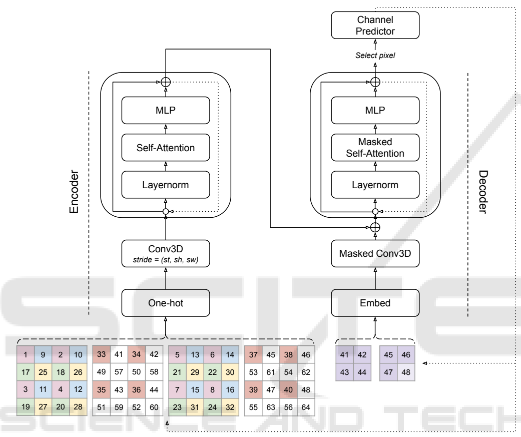

Figure 2: Video Transformer adapted to latent codes. Numbers represent generation order. Pixels are colored if they are already generated. White-colored pixels are zero-padded. Pixels with the same color belong to the same slice. The example represents the generation of the last pixel of slice $Z_{(1,0,1)}$ for a latent video of size $(t,h,w) = (4,4,4)$ and $(s_t,s_h,s_w) = (2,2,2)$.

4 EXPERIMENTS

4.1 Experimental Setup

We model the videos of length $T = 16$ and spatial size 64×64 similar to the setup of prior works in this field (Clark et al., 2019; Weissenborn et al., 2019).

Measures of Quality. Video prediction is a challenging problem, as there are many possible future outcomes for given conditioning frames. Therefore conventional metrics as Peak Signal-to-Noise Ratio (PSNR) and Structural Similarity Index Measure (SSIM) that require ground truth correspondence were later displaced by the better-suited metric — Fréchet Video Distance (FVD) (Heusel et al., 2017). FVD applies idea from Fréchet Inception Distance (Heusel et al., 2017) for videos and computes Fréchet distance between real and generated samples based on statistics calculated on logits from action-recognition Inception3D network trained on Kinetics-400 (Kay et al., 2017) dataset. The metric was shown to better correlate with human perception, than previously used ones.

We also report bits per dimension[1] (bits/dim) — negative $\log_2$-probability averaged across all generated (latent) pixels and channels.

Frame Autoencoder. The encoder contains two strided convolutional layers with ReLU activation function, stride 2 and kernel size 4×4, followed by a convolution layer with kernel size 3×3, the same padding, followed by two residual blocks (ReLU, 3×3 conv, ReLU, 1×1 conv). The decoder has a symmetrical structure containing two residual blocks, followed by two transposed convolutions with stride 2 and window size 4×4. For Kinetics-600 dataset, we use four residual blocks instead of two both in encoder and decoder.

In our experiments we explore two setups for the codebook structure: (the default one) $n_c = 1$, $K = 512$ and $n_c = 4$, $K = 2048$. The embedding dimension inside codebook is $D = 256$. The encoder and discretization bottleneck converts $64 \times 64 \times 3$ RGB image into $16 \times 16 \times n_c$ discrete latent codes indices.

We trained VQ-VAE using Adam optimizer with learning rate 0.0003 for 500K steps in case of BAIR Robot pushing dataset and 1M steps in case of

[1]Due to the nature of frame autoencoder, it is not possible to compute bits/dim directly. We provide bits/dim in the latent space. Other works provide this metric for the pixel space. One can consider bits/dim in the latent space as the lower bound to real bits/dim on images.

Kinetics-600 dataset using a batch of 32 images.

Latent Video Generator. As no public code was available, Video Transformer implementation was written from scratch using the setup of a medium size model and following the implementation and training details from the original paper (Weissenborn et al., 2019).

Different from the original transformer, an attention block in the encoder and the decoder is not block-local and spans across the whole input. It is possible due to the reduction of the input size via VQ-VAE. In almost all our experiments we also compare different subscaling types: (i) spatiotemporal ($\mathbf{s} = (4,2,2)$), (ii) spatial ($\mathbf{s} = (1,2,2)$), and (iii) single frame ($\mathbf{s} = (T,1,1)$).

Each model was trained on 8 Nvidia V100 GPUs for two days for 600K steps. Sampling one video of 16 frames with 5 priming frames takes about 30 seconds on 1 Nvidia V100 GPU, compared to one minute used by the original Video Transformer (Weissenborn et al., 2019). The approximate size of a latent video generator is 50M parameters.

We provide quantitative and qualitative results on two datasets: BAIR Robot Pushing (Ebert et al., 2017) and Kinetics 600 (Carreira et al., 2018).

4.2 BAIR Robot Pushing

BAIR Robot Pushing (Ebert et al., 2017) dataset consists of 40K training and 256 test videos of robotic arm motion with a fixed camera position. First, we evaluate the VQ-VAE's reconstruction error with varying number of codebooks used inside the discretization bottleneck. We provide mean squared error (MSE) and FVD for reconstructed videos of 16 frames length (see Table 1).

Table 1: VQ-VAE performance on BAIR Robot Pushing dataset.

$\mathbf{n}_c$	**MSE**($\downarrow$)	**FVD**($\downarrow$)
1	0.0016	222.71
4	0.0004	47.41

In terms of video prediction, following the setup of previous approaches (Weissenborn et al., 2019; Clark et al., 2019; Luc et al., 2020), we train video generator conditioning on one frame and report metrics for videos of 16 frames. FVD and bits/dim are computed on videos with five priming frames and one priming frame accordingly (see Table 2). We report the mean and standard deviations of 10 runs.

It can bee seen that both VQ-VAE and Video Transformer demonstrate better accuracy when using four codebooks inside the discretization bottleneck. Preliminary experiments showed that a further increase of the number of codebooks would lead to overfitting. Finally, we compare our approach to others (see Table 3). We also provide the *baseline* solution: what if we take the last ground truth frame and use it as a prediction for all future frames.

We achieve comparable performance in comparison to other methods. We also provide samples for qualitative assessment (see Fig 3).

4.3 Kinetics-600

Kinetics-600 (Carreira et al., 2018) dataset consists of 350k train and 50k test videos. There are 600 classes presented. Following the setup of previous approaches (Weissenborn et al., 2019; Clark et al., 2019; Luc et al., 2020), we cropped each video to the size of the smallest side. Then we resized them to 64×64 using Lanczos filter. First, we evaluate the VQ-VAE's reconstruction error with varying number of codebooks used inside the discretization bottleneck. For that, we provide mean squared error (MSE) and FVD for reconstructed videos of 16 frames (see Table 4).

VQ-VAE with four codebooks outperforms in terms of both metrics, and therefore later, we conduct the experiments under only this setup as the evaluation time for Kinetics-600 is particularly significant due to the large size of test data.

In terms of video prediction, following the setup of previous approaches (Weissenborn et al., 2019; Clark et al., 2019; Luc et al., 2020), we train the video generator conditioning on one frame and report metrics for videos of 16 frames. FVD and bits/dim are computed on videos with five priming frames and one priming frame accordingly (see Table 5).

We compare our approach to others (see Table 6). Here the baseline is the prediction of the next frame by the previous known frame.

Our results are inferior to others on this dataset. We conclude that it is caused by error accumulation inside the Transformer model. We link it to the high complexity and diversity of the Kinetics-600 dataset. We want to emphasize that only four other approaches tried to model videos from this dataset, and all of them use six times bigger generative models (up to 350M parameters) than ours. In the meantime, increasing the size of our model led to a very slow convergence.

We also provide samples from our model for qualitative assessment (see Fig. 4). One can notice artifacts in the second video. We found approximately half of the videos to be good and half of the videos to have artifacts. We provide more visualization results in the Appendix.

Table 2: Video prediction performance on BAIR Robot Pushing dataset. Best results in bold.

Subscaling type	$\mathbf{n_c}$	**Bits/dim(↓)**	**FVD(↓)**
Single Frame	1	1.28	258.89 ± 2.85
Spatial	1	1.79	524.43 ± 9.41
Spatiotemporal	1	**1.25**	275.71 ± 5.41
Single Frame	4	1.53	$\mathbf{125.76 \pm 2.90}$
Spatial	4	2.99	920.37 ± 7.71
Spatiotemporal	4	1.62	145.85 ± 1.68

Table 3: Comparison of different methods for video prediction on BAIR Robot Pushing dataset.

Method	**bits/dim(↓)**	**FVD(↓)**
Baseline	-	320.90
VideoFlow (Kumar et al., 2019)	1.87	-
SVP-FP (Denton and Fergus, 2018c)	-	315.5
CDNA (Finn et al., 2016b)	-	296.5
LVT (ours, $n_c = 1$)	1.25	275.71 ± 5.41
SV2P (Denton and Fergus, 2018a)	-	262.5
LVT (ours, $n_c = 4$)	1.53	125.8 ± 2.9
SAVP (Lee et al., 2018)	-	116.4
DVD-GAN-FP (Clark et al., 2019)	-	109.8
TriVD-GAN-FP (Luc et al., 2020)	-	103.3
Axial Transformer (Ho et al., 2019)	1.29	-
Video Transformer (Weissenborn et al., 2019)	1.35	$\mathbf{94 \pm 2}$

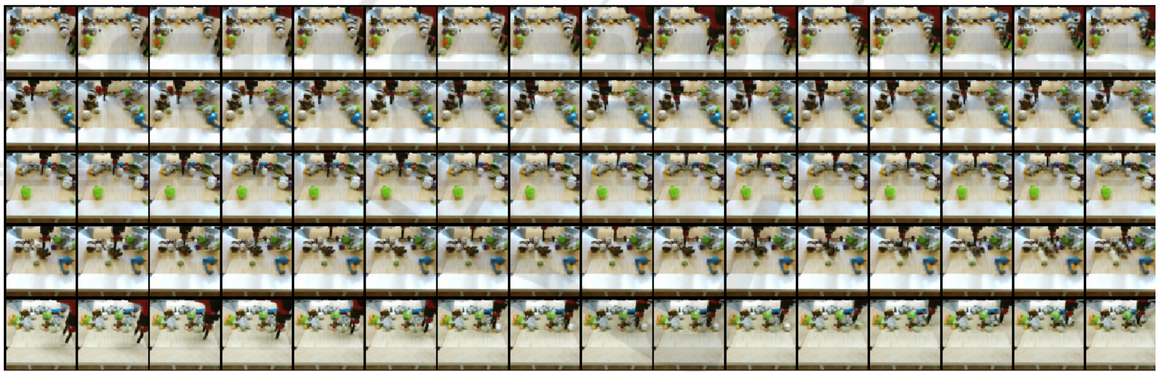
Figure 3: Samples from BAIR Robot Pushing dataset. Each row represents a single video with first 5 frames being real and others generated.

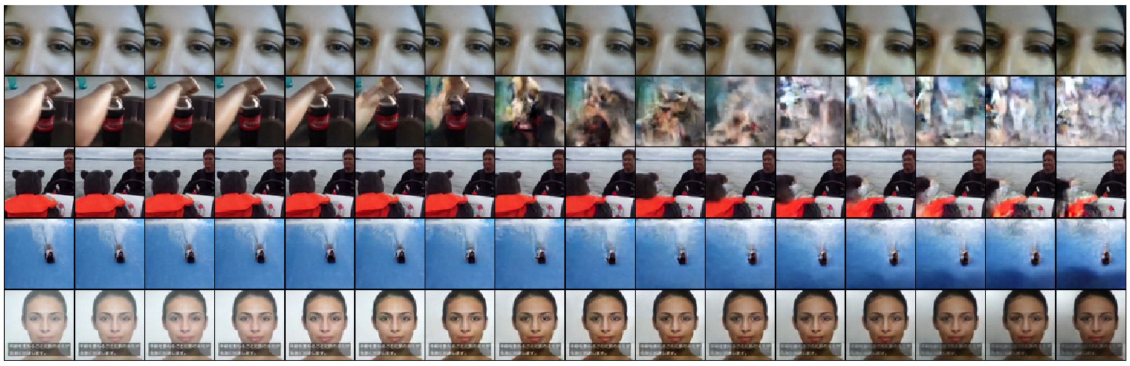
Figure 4: Samples from Kinetics-600 dataset. Each row represents a single video with first five frames being real and others generated.

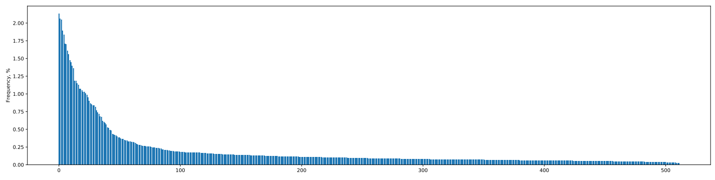

Figure 5: Sorted codes frequencies for BAIR Robot Pushing dataset.

Table 4: VQ-VAE performance on Kinetics-600 dataset.

$\mathbf{n_c}$	MSE(↓)	FVD(↓)
1	0.002	396.58
4	0.0004	25.95

Table 5: Video prediction performance on Kinetics-600 dataset. Best results in bold.

Subscaling type	$\mathbf{n_c}$	Bits/dim(↓)	FVD(↓)
Single Frame	4	**2.14**	**224.73**
Spatial	4	4.22	2845.06 ± 612.07
Spatiotemporal	4	2.47	338.39 ± 0.21

4.4 Ablation Study of Attention

The Video Transformer contains memory and time costly operation of multi-head attention. In general case multi-head attention computes feature y_q as:

$$y_q = \sum_{m=1}^{M} W_m \left[\sum_{k \in \Omega_q} A_m \left(x_q, x_k\right) \odot V_m x_k \right], \qquad (3)$$

where m, q, k are the indexes of attention head, query and key elements respectively. W_m and V_m are learnable matrices. A_m computes the attention weight for an each key element. We also normalize attention weights, s.t. $\sum_{k \in \Omega_q} A_m\left(q, k, z_q, x_k\right) = 1$.

In our default setup, the attention weights are computed as:

$$A_m\left(x_q, x_k\right) \propto \exp\left(x_q^\top Q_m^\top K_m x_k + b_{kq}\right), \qquad (4)$$

where Q_m, K_m are learnable matrices for retrieving key and content embeddings, and b_{kq} is computed as the sum of per-dimension relative distance biases between pixels k and q. We would refer to this type of attention as *"query-key + relative distance"*.

We also explore two other variants:

- *"key + relative distance":* $A_m\left(x_q, x_k\right) \propto \exp\left(u_m^\top K_m x_k + b_{kq}\right)$, where u_m is a lernable vector,
- *"relative distance only":* $A_m\left(x_q, x_k\right) \propto \exp\left(b_{kq}\right)$.

The empirical comparison of these different types of attention can be seen at Table 7.

Similar to (Zhu et al., 2019) we find that query-key term inside self-attention module does not play a crucial role in the success of Latent Video Generator and can be replaced with cheaper variants with a cost of slight quality reduction.

4.5 Adaptive Input and Adaptive Softmax

We analyzed how often each latent code from the codebook was used for encoding images from train videos in BAIR Robot Pushing dataset. We found that 218 latent codes out of 512 constitute the 80% of probability mass (see Fig. 5). Based on this fact, we tried to improve metrics using Adaptive Input (Baevski and Auli, 2018) and Adaptive Softmax (Grave et al., 2017). Neither of them brings an improvement to quality (see Table 8), despite their successful applications in natural language processing. For an interested reader, we refer to the original papers for particular details.

4.6 BAIR Visualizations

We present visualizations of BAIR robot pushing dataset in Fig. 6. At the first row of each figure there is a video: 5 real and 11 generated frames. Rows 2-5 contain codes visualizations. Each code row corresponds to one codebook. Since a single code is just a matrix of indexes, we decode it with a technique called *indexed color*. In other words, we assigned each index in a code to a specific color. Rows 6-9 represent binary mask denoting whether the latent code between consecutive frames changes or not (yellow means a change).

One can see that results on the BAIR dataset are quite realistic. Also, it is important to note that some of the codes stay the same from frame to frame. The static background causes it. Codes that are responsible for non-static objects change from frame to frame.

4.7 Kinetics-600 Visualizations

We present good samples along with codes on the Kinetics-600 dataset in Fig. 7. Bad samples with codes are presented in Fig. 8. One can see that bad samples and good samples are different in their codes and code differences. For latent transformer, it is easier to predict the latent code of the next frame if it is similar to the latent code of the current frame.

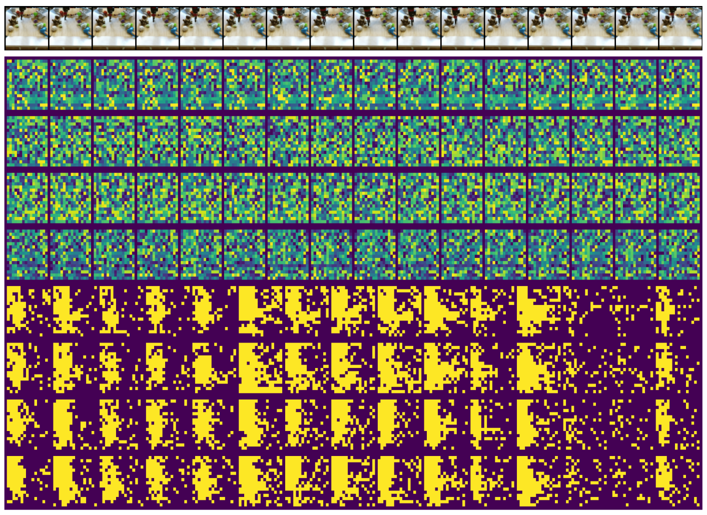

Figure 6: Sample from BAIR robot pushing dataset. The first row represents a single video with the first five frames being real and others generated. Rows 2-5 represent four latent codes, one row for each codebook. Rows 6-9 represent binary mask denoting whether the latent code between consecutive frames changes or not (yellow means a change).

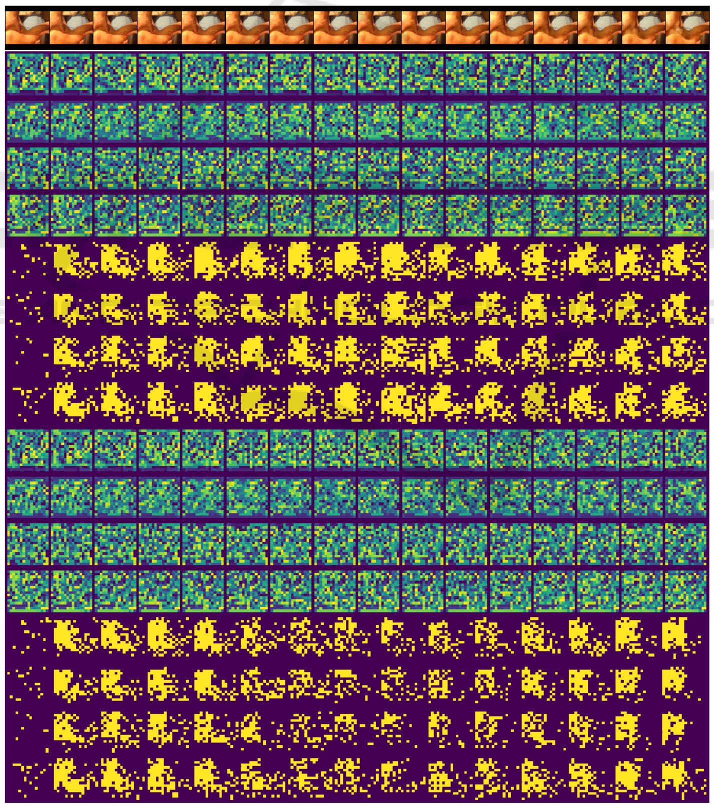

Figure 7: Good sample from Kinetics-600 dataset. The first row represents a single video with the first five frames being real and others generated. Rows 2-5 represent four latent codes for real video, one row for each codebook. Rows 6-9 represent binary mask denoting whether the latent code between consecutive frames changes or not (yellow means a change). Rows 10-13 represent four latent codes for generated video, one row for each codebook. Rows 14-17 represent binary mask denoting whether the latent code between consecutive frames changes or not (yellow means a change).

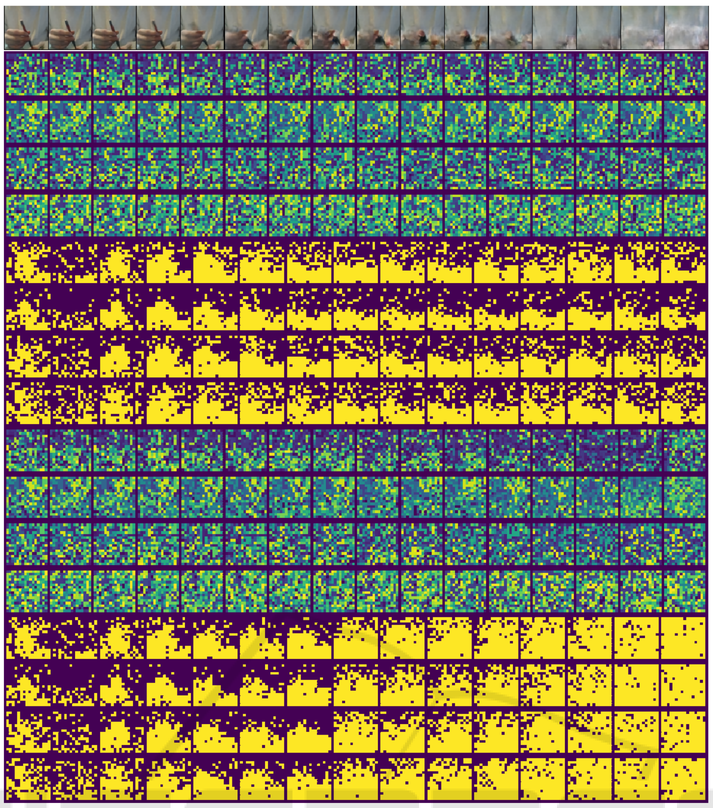

Figure 8: Bad sample from Kinetics-600 dataset. The first row represents a single video with the first five frames being real and others generated. Rows 2-5 represent four latent codes for real video, one row for each codebook. Rows 6-9 represent binary mask denoting whether the latent code between consecutive frames changes or not (yellow means a change). Rows 10-13 represent four latent codes for generated video, one row for each codebook. Rows 14-17 represent binary mask denoting whether the latent code between consecutive frames changes or not (yellow means a change).

Table 6: Comparison of different methods for video prediction on Kinetics-600 dataset.

Method	**Bits/dim**(↓)	**FVD**(↓)
Baseline	-	271.00
LVT (ours)	2.14	224.73
Video Transformer (Weissenborn et al., 2019)	1.19	170 ± 5
DVD-GAN-FP (Clark et al., 2019)	-	69.15 ± 1.16
TriVD-GAN-FP (Luc et al., 2020)	-	25.74 ± 0.66

Table 7: Attention comparison on BAIR Robot Pushing dataset. The results are obtained using the model with a single frame subscaling type modeling the latent space with $n_c = 4$ codebooks.

attention type	**Bits/dim**(↓)	**FVD**(↓)	**params (M)**	**inference time (sec)**
query-key + relative distance	1.53	125.76 ± 2.90	49.87	35
key-only + relative distance	1.57	130.27 ± 4.26	41.50	32
relative distance only	1.58	141.62 ± 4.34	33.09	30

5 CONCLUSION

In this work, we tackled the video generation problem. Given several first frames, the goal was to predict the continuation of a video. Modern methods for video generation requires up to 512 Tensor Processing Units for parallel training. We were focused on the reduction of the computational requirements of the model. We showed that one could achieve comparable results on video prediction by training

Table 8: Effects of applying adaptive input / softmax in decoder architecture. The results are obtained using the model with a single frame subscaling. Latent space is modelled with $n_c = 1$.

Subscaling type	Decoder Input	Decoder Output	Bits/dim(↓)	FVD(↓)
Single Frame	128d Embedding	512d Softmax	1.28	$\mathbf{258.89 \pm 2.85}$
Spatial	128d Embedding	512d Softmax	1.79	524.43 ± 9.41
Spatiotemporal	128d Embedding	512d Softmax	**1.25**	275.71 ± 5.41
Single Frame	128d Embedding	512d Softmax (tied emb)	1.27	265.10 ± 3.85
Single Frame	Adaptive Input	512d Softmax	1.26	259.73 ± 6.35
Single Frame	128d Embedding	Adaptive Softmax	1.37	265.99 ± 4.16
Single Frame	Adaptive Input	Adaptive Softmax	1.36	259.88 ± 5.25
Single Frame	Adaptive Input	Adaptive Softmax (tied emb)	1.35	259.55 ± 8.50
Single Frame	Adaptive Input	Adaptive Softmax (tied emb/proj)	1.34	264.79 ± 4.27

a model using the usual research setup — 8 V100 GPUs. To achieve such a result, we moved the video generation process from pixel space to a latent space. We demonstrated decent results on the dataset BAIR Robot Pushing. In the meantime, in some cases, we observe visual artifacts on the Kinetics-600 dataset. For future work, one can consider joint training of Frame Autoencoder and Latent Video Generator.

ACKNOWLEDGEMENT

The authors acknowledge the usage of the Skoltech CDISE HPC cluster Zhores for obtaining the results presented in this paper. The authors were supported by the Russian Science Foundation under Grant 19-41-04109. They also acknowledge Vage Egiazarian for thoughtful discussions of the model and the experiments.

REFERENCES

Acharya, D., Huang, Z., Paudel, D. P., and Van Gool, L. (2018). Towards high resolution video generation with progressive growing of sliced wasserstein gans. *arXiv preprint arXiv:1810.02419*.

Baevski, A. and Auli, M. (2018). Adaptive input representations for neural language modeling. *arXiv preprint arXiv:1809.10853*.

Brock, A., Donahue, J., and Simonyan, K. (2018). Large scale gan training for high fidelity natural image synthesis. *arXiv preprint arXiv:1809.11096*.

Byeon, W., Wang, Q., Kumar Srivastava, R., and Koumoutsakos, P. (2018). Contextvp: Fully context-aware video prediction. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 753–769.

Carreira, J., Noland, E., Banki-Horvath, A., Hillier, C., and Zisserman, A. (2018). A short note about kinetics-600. *arXiv preprint arXiv:1808.01340*.

Chen, X., Mishra, N., Rohaninejad, M., and Abbeel, P. (2017). Pixelsnail: An improved autoregressive generative model. *arXiv preprint arXiv:1712.09763*.

Clark, A., Donahue, J., and Simonyan, K. (2019). Adversarial video generation on complex datasets. *arXiv preprint arXiv:1907.06571*.

Denton, E. and Fergus, R. (2018a). Stochastic video generation with a learned prior. *arXiv preprint arXiv:1802.07687*.

Denton, E. and Fergus, R. (2018b). Stochastic video generation with a learned prior. In *International Conference on Machine Learning*, pages 1174–1183.

Denton, E. and Fergus, R. (2018c). Stochastic video generation with a learned prior. *arXiv preprint arXiv:1802.07687*.

Denton, E. L. et al. (2017). Unsupervised learning of disentangled representations from video. In *Advances in neural information processing systems*, pages 4414–4423.

Dhariwal, P., Jun, H., Payne, C., Kim, J. W., Radford, A., and Sutskever, I. (2020). Jukebox: A generative model for music. *arXiv preprint arXiv:2005.00341*.

Ebert, F., Finn, C., Lee, A. X., and Levine, S. (2017). Self-supervised visual planning with temporal skip connections. *arXiv preprint arXiv:1710.05268*.

Endo, Y., Kanamori, Y., and Kuriyama, S. (2019). Animating landscape: self-supervised learning of decoupled motion and appearance for single-image video synthesis. *arXiv preprint arXiv:1910.07192*.

Finn, C., Goodfellow, I., and Levine, S. (2016a). Unsupervised learning for physical interaction through video prediction. In *Advances in neural information processing systems*, pages 64–72.

Finn, C., Goodfellow, I., and Levine, S. (2016b). Unsupervised learning for physical interaction through video prediction. In *Advances in neural information processing systems*, pages 64–72.

Grave, E., Joulin, A., Cissé, M., Jégou, H., et al. (2017). Efficient softmax approximation for gpus. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 1302–1310. JMLR. org.

Heusel, M., Ramsauer, H., Unterthiner, T., Nessler, B., and Hochreiter, S. (2017). Gans trained by a two time-scale update rule converge to a local nash equilibrium. In *Advances in neural information processing systems*, pages 6626–6637.

Ho, J., Kalchbrenner, N., Weissenborn, D., and Salimans, T. (2019). Axial attention in multidimensional transformers. *arXiv preprint arXiv:1912.12180*.

Hsieh, J.-T., Liu, B., Huang, D.-A., Fei-Fei, L. F., and Niebles, J. C. (2018). Learning to decompose and disentangle representations for video prediction. In *Advances in Neural Information Processing Systems*, pages 517–526.

Kaiser, Ł., Roy, A., Vaswani, A., Parmar, N., Bengio, S., Uszkoreit, J., and Shazeer, N. (2018). Fast decoding in sequence models using discrete latent variables. *arXiv preprint arXiv:1803.03382*.

Kalchbrenner, N., van den Oord, A., Simonyan, K., Danihelka, I., Vinyals, O., Graves, A., and Kavukcuoglu, K. (2017). Video pixel networks. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 1771–1779. JMLR. org.

Kay, W., Carreira, J., Simonyan, K., Zhang, B., Hillier, C., Vijayanarasimhan, S., Viola, F., Green, T., Back, T., Natsev, P., et al. (2017). The kinetics human action video dataset. *arXiv preprint arXiv:1705.06950*.

Kumar, M., Babaeizadeh, M., Erhan, D., Finn, C., Levine, S., Dinh, L., and Kingma, D. (2019). Videoflow: A flow-based generative model for video. *arXiv preprint arXiv:1903.01434*, 2(5).

Lee, A. X., Zhang, R., Ebert, F., Abbeel, P., Finn, C., and Levine, S. (2018). Stochastic adversarial video prediction. *arXiv preprint arXiv:1804.01523*.

Luc, P., Clark, A., Dieleman, S., Casas, D. d. L., Doron, Y., Cassirer, A., and Simonyan, K. (2020). Transformation-based adversarial video prediction on large-scale data. *arXiv preprint arXiv:2003.04035*.

Mathieu, M., Couprie, C., and LeCun, Y. (2015). Deep multi-scale video prediction beyond mean square error. *arXiv preprint arXiv:1511.05440*.

Menick, J. and Kalchbrenner, N. (2018). Generating high fidelity images with subscale pixel networks and multidimensional upscaling. *arXiv preprint arXiv:1812.01608*.

Nam, S., Ma, C., Chai, M., Brendel, W., Xu, N., and Kim, S. J. (2019). End-to-end time-lapse video synthesis from a single outdoor image. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1409–1418.

Ohnishi, K., Yamamoto, S., Ushiku, Y., and Harada, T. (2018). Hierarchical video generation from orthogonal information: Optical flow and texture. In *Thirty-Second AAAI Conference on Artificial Intelligence*.

Pan, J., Wang, C., Jia, X., Shao, J., Sheng, L., Yan, J., and Wang, X. (2019). Video generation from single semantic label map. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3733–3742.

Patraucean, V., Handa, A., and Cipolla, R. (2015). Spatio-temporal video autoencoder with differentiable memory. *arXiv preprint arXiv:1511.06309*.

Ranzato, M., Szlam, A., Bruna, J., Mathieu, M., Collobert, R., and Chopra, S. (2014). Video (language) modeling: a baseline for generative models of natural videos. *arXiv preprint arXiv:1412.6604*.

Razavi, A., van den Oord, A., and Vinyals, O. (2019). Generating diverse high-fidelity images with vq-vae-2. In *Advances in Neural Information Processing Systems*, pages 14837–14847.

Saito, M., Matsumoto, E., and Saito, S. (2017). Temporal generative adversarial nets with singular value clipping. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2830–2839.

Saito, M. and Saito, S. (2018). Tganv2: Efficient training of large models for video generation with multiple subsampling layers. *arXiv preprint arXiv:1811.09245*.

Salimans, T., Karpathy, A., Chen, X., and Kingma, D. P. (2017). Pixelcnn++: Improving the pixelcnn with discretized logistic mixture likelihood and other modifications. *arXiv preprint arXiv:1701.05517*.

Shaham, T. R., Dekel, T., and Michaeli, T. (2019). Singan: Learning a generative model from a single natural image. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 4570–4580.

Srivastava, N., Mansimov, E., and Salakhudinov, R. (2015). Unsupervised learning of video representations using lstms. In *International conference on machine learning*, pages 843–852.

Tulyakov, S., Liu, M.-Y., Yang, X., and Kautz, J. (2018). Mocogan: Decomposing motion and content for video generation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1526–1535.

Van den Oord, A., Kalchbrenner, N., Espeholt, L., Vinyals, O., Graves, A., et al. (2016). Conditional image generation with pixelcnn decoders. In *Advances in neural information processing systems*, pages 4790–4798.

van den Oord, A., Vinyals, O., et al. (2017). Neural discrete representation learning. In *Advances in Neural Information Processing Systems*, pages 6306–6315.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. (2017). Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008.

Vondrick, C., Pirsiavash, H., and Torralba, A. (2016). Generating videos with scene dynamics. In *Advances in neural information processing systems*, pages 613–621.

Wang, T.-C., Liu, M.-Y., Tao, A., Liu, G., Kautz, J., and Catanzaro, B. (2019). Few-shot video-to-video synthesis. *arXiv preprint arXiv:1910.12713*.

Wang, T.-C., Liu, M.-Y., Zhu, J.-Y., Liu, G., Tao, A., Kautz, J., and Catanzaro, B. (2018a). Video-to-video synthesis. *arXiv preprint arXiv:1808.06601*.

Wang, Y., Gao, Z., Long, M., Wang, J., and Yu, P. S. (2018b). Predrnn++: Towards a resolution of the deep-in-time dilemma in spatiotemporal predictive learning. *arXiv preprint arXiv:1804.06300*.

Wang, Y., Jiang, L., Yang, M.-H., Li, L.-J., Long, M., and Fei-Fei, L. (2018c). Eidetic 3d lstm: A model for video prediction and beyond.

Weissenborn, D., Täckström, O., and Uszkoreit, J. (2019). Scaling autoregressive video models. *arXiv preprint arXiv:1906.02634*.

Xingjian, S., Chen, Z., Wang, H., Yeung, D.-Y., Wong, W.-K., and Woo, W.-c. (2015). Convolutional lstm network: A machine learning approach for precipitation nowcasting. In *Advances in neural information processing systems*, pages 802–810.

Zhao, Y., Li, C., Yu, P., Gao, J., and Chen, C. (2020). Feature quantization improves gan training. *arXiv preprint arXiv:2004.02088*.

Zhu, X., Cheng, D., Zhang, Z., Lin, S., and Dai, J. (2019). An empirical study of spatial attention mechanisms in deep networks. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 6688–6697.