StyleGAN-V: A Continuous Video Generator with the Price, Image Quality and Perks of StyleGAN2

Ivan Skorokhodov
KAUST

Sergey Tulyakov
Snap Inc.

Mohamed Elhoseiny
KAUST

Abstract

Videos show continuous events, yet most — if not all — video synthesis frameworks treat them discretely in time. In this work, we think of videos of what they should be — time-continuous signals, and extend the paradigm of neural representations to build a continuous-time video generator. For this, we first design continuous motion representations through the lens of positional embeddings. Then, we explore the question of training on very sparse videos and demonstrate that a good generator can be learned by using as few as 2 frames per clip. After that, we rethink the traditional image + video discriminators pair and design a holistic discriminator that aggregates temporal information by simply concatenating frames' features. This decreases the training cost and provides richer learning signal to the generator, making it possible to train directly *on 1024^2 videos for the first time. We build our model on top of StyleGAN2 and it is just ≈5% more expensive to train at the same resolution while achieving almost the same image quality. Moreover, our latent space features similar properties, enabling spatial manipulations that our method can propagate in time. We can generate arbitrarily long videos at arbitrary high frame rate, while prior work struggles to generate even 64 frames at a fixed rate. Our model is tested on four modern 256^2 and one 1024^2-resolution video synthesis benchmarks. In terms of sheer metrics, it performs on average ≈30% better than the closest runner-up. Project website: https://universome.github.io/stylegan-v.*

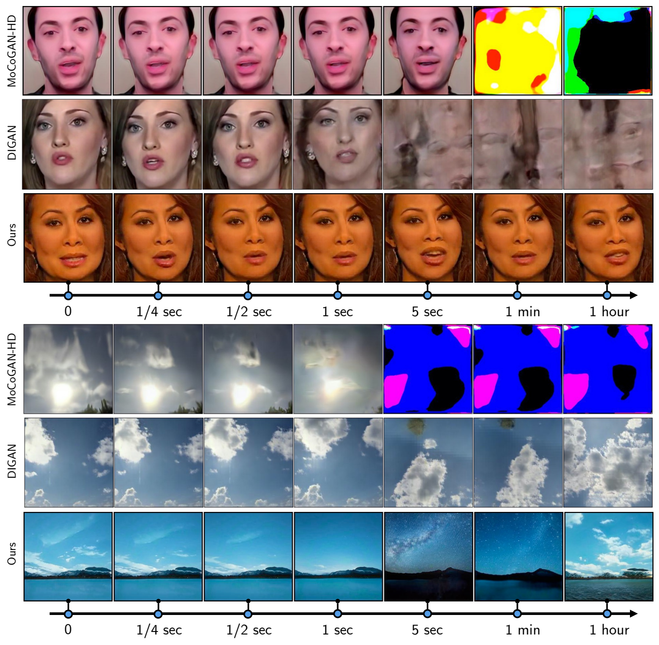

Figure 1. Examples of 1-hour long videos, generated with different methods. MoCoGAN-HD [65] fails to generate long videos due to the instability of the underlying LSTM model when unrolled to large lengths. DIGAN [80] struggles to generate long videos due to the entanglement of spatial and temporal positional embeddings. StyleGAN-V (our method) generates plausible videos of arbitrary length and frame-rate. Also, unlike DIGAN, it learns temporal patterns not only in terms of motion, but also appearance transformations, like time of day and weather changes.

1. Introduction

Recent advances in deep learning pushed image generation to the unprecedented photo-realistic quality [8, 28] and spawned a lot of its industry applications. Video generation, however, does not enjoy a similar success and struggles to fit complex real-world datasets. The difficulties are caused not only by the more complex nature of the underlying data distribution, but also due to the computationally intensive video representations employed by modern generators. They treat videos as discrete sequences of images, which is very demanding for representing long high-resolution videos and induces the use of expensive `conv3d`-based architectures to model them [13, 54, 55, 67]. [1]

In this work, we argue that this design choice is not optimal and propose to treat videos in their natural form: as continuous signals $\boldsymbol{x}(t)$, that map *any* time coordinate

[1] E.g., DVD-GAN [13] requires ≈\$30k to train on 256^2 resolution [65]

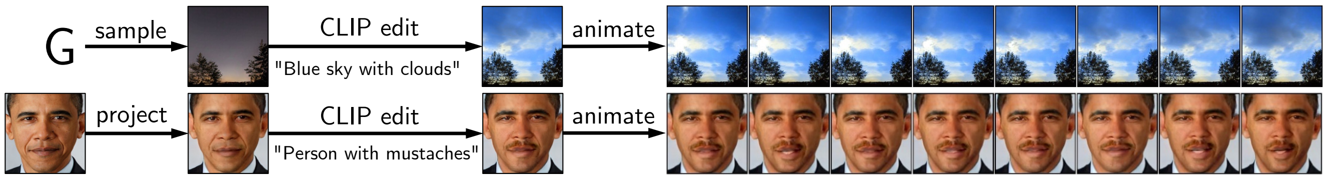

Figure 2. Our model enjoys all the perks of StyleGAN2 [30], including the ability of semantic manipulation. In this example, we edited a generated frame (top row) or projected off-the-shelf image (bottom row) with CLIP and animated it with our model. To the best of our knowledge, our work is the first one which demonstrates such capabilities for video generators.

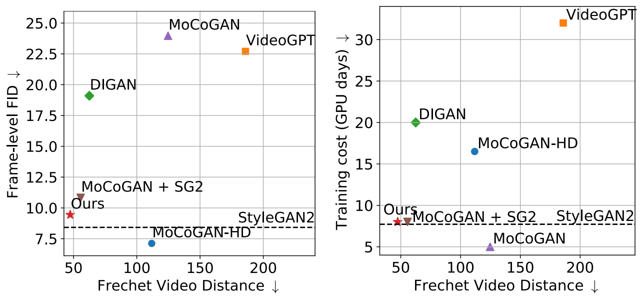

Figure 3. FID scores and training cost by FVD_{16} for modern video generators on FaceForensics 256^2 [53]. Our method (denoted by $\star$) shows that video generators can be as efficient and as good in terms of image quality as traditional image based generators (like, StyleGAN2 [30], denoted with the dashed line).

$t \in \mathbb{R}_+$ into an image frame $\boldsymbol{x}(t) = \boldsymbol{x}_t \in \mathbb{R}^{3\times h\times w}$. Consequently, we develop a GAN-based continuous video synthesis framework by extending the recent paradigm of neural representations [38, 58, 64] to the video generation domain.

Developing such a framework comes with three challenges. First, sine/cosine positional embeddings are periodic by design and depend only on the input coordinates. This does not suit video generation, where temporal information should be aperiodic (otherwise, videos will be cycled) and different for different samples. Next, since videos are perceived as infinite continuous signals, one needs to develop an appropriate sampling scheme to use them in a practical framework. Finally, one needs to accordingly redesign the discriminator to work with the new sampling scheme.

To solve the first issue, we develop positional embeddings with time-varying wave parameters which depend on motion information, sampled uniquely for different videos. This motion information is represented as a sequence of motion codes produced by a *padding-less* conv1d-based model. We prefer it over the usual LSTM network [3, 55, 65, 67] to alleviate the RNN's instability when unrolled to large depths and to produce frames non-autoregressively.

Next, we investigate the question of how many samples are needed to learn a meaningful video generator. We argue that it can be learned from *extremely* sparse videos (as few as 2 frames per clip), and justify it with a simple theoretical exposition (§3.3) and practical experiments (see Table 2).

Finally, since our model sees only 2-4 randomly sampled frames per video, it is highly redundant to use expensive conv3d-blocks in the discriminator, which are designed to operate on long sequences of equidistant frames. That's why we replace it with a conv2d-based model, which aggregates information temporally via simple concatenation and is conditioned on the time distances between its input frames. Such redesign improves training efficiency (see Table 1), provides more informative gradient signal to the generator (see Fig 4) and simplifies the overall pipeline (see §3.2), since we no longer need two different discriminators to operate on image and video levels separately, as modern video synthesis models do (e.g., [13, 55, 67]).

We build our model, named StyleGAN-V, on top of the image-based StyleGAN2 [30]. It is able to produce arbitrarily long videos at arbitrarily high frame-rate in a non-autoregressive manner and enjoys great training efficiency — it is only $\approx$5% costlier than the classical *image-based* StyleGAN2 model [30], while having only $\approx$10% worse *plain* image quality in terms of FID [23] (see Fig 3). This allows us to easily scale it to HQ datasets and we demonstrate that it is *directly* trainable on 1024^2 resolution.

For empirical evaluation, we use 5 benchmarks: FaceForensics 256^2 [53], SkyTimelapse 256^2 [78], UCF101 256^2 [62], RainbowJelly 256^2 (introduced in our work) and MEAD 1024^2 [72]. Apart from our model, we train from scratch 5 different methods and measure their performance using the same evaluation protocol. Frechet Video Distance (FVD) [68] serves as the main metric for video synthesis, but there is no *complete* official implementation for it (see §4 and Appx C). This leads to discrepancies in the evaluation procedures used by different works because FVD, similarly to FID [23], is *very* sensitive to data format and sampling strategy [46]. That's why we implement, document and release our complete FVD evaluation protocol. In terms of sheer metrics, our method performs on average $\approx$30% better than the closest runner-up.

2. Related work

Video synthesis. Early works on video synthesis mainly focused on *video prediction* [34, 70], i.e. generating future frames given a sequence of the previously seen ones. Early approaches for this problem typically employed recurrent convolutional models trained with reconstruction objective [16,52,63], but later adversarial losses were introduced to improve the synthesis quality [35,69,73]. Some recent works explore autoregressive video prediction with recurrent or attention-based models (e.g., [26,51,71,76,79]). Another close line of research is *video interpolation*, i.e. increasing the frame rate of a given video (e.g., [6, 24, 42]). In our work, we study *video generation*, which is a more challenging problem than video prediction since it seeks to synthesize videos from scratch, i.e. without using the expressive conditioning on previous frames. Classical methods in this direction are typically based on GANs [19]. MoCoGAN [67] and TGAN [54] decompose generator's input noise into a content code and motion codes, which became a standard strategy for many subsequent works (e.g., [3, 40, 55, 65]). Several approaches consider video generation from a single clip (e.g., [5, 20, 21]).

Some recent works also consider high-resolution video synthesis [17, 65], but only with training in the latent space of a pretrained image generator. StyleGAN-V is trained on *extremely* sparse videos. This makes it related to [10,55,75], which use a pyramid of discriminators operating on different temporal resolutions (with a subsampling factor of up to $\times 8$). Our model builds on the time continuity, which in the context of video synthesis was also explored by [45].

To the best of our knowledge, all modern video synthesis approaches utilize expensive `conv3d` blocks either in their decoder and/or encoder components (e.g., [2, 13, 25, 40, 55, 65,67]). Often, GAN-based approaches utilize two discriminators, operating on image and video levels independently, where the video discriminator operates at a low resolution to save computation (e.g., [13, 65, 67, 74]). In our work, we aggregate the temporal information via a simple concatenation of feature vectors extracted from the frames and this strategy suffices to build a state-of-the-art video generator.

Neural Representations. Neural representations is a recent paradigm that uses neural networks to represent continuous signals, such as images, videos, audios, 3D objects and scenes (e.g., [18, 38, 58, 59, 64]). It is mostly popular for 3D reconstruction and geometry processing tasks (e.g., [33,37,41,43,48]), including video-based reconstruction [32, 44, 49, 77]. Several recent projects explored the task of building generative models over such representations to synthesize images (e.g., [4, 60, 61]), 3D objects (e.g., [11,31,56]) or multi-modal signals (e.g., [14,15]), and our work extends this line of research to video generation.

Concurrent works. The development of neural representations-based approaches moves extremely fast and there are two concurrent works which propose ideas similar to our ones. DIGAN [80] is a concurrent project that explores the same direction of using neural-based representations for continuous video synthesis and shares a *lot* of ideas with our work. The authors also consider a continuous-time generator, trained by a discriminator without `conv3d` layers. The core difference with our work is that they use a different parametrization of motions and use a dual discriminator D: one operates on $(\boldsymbol{x}_1, \boldsymbol{x}_2, \Delta t)$ and the second one on individual images. We enumerate the differences and similarities in Appx H. NeRV [12] uses convolutional neural representations of videos for compression and denoising tasks. GEM [14] utilizes generative latent optimization [7] to build a multi-modal generative model.

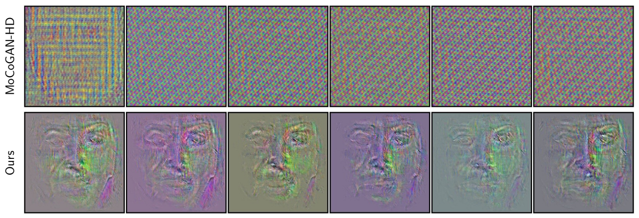

Figure 4. Visualizing the gradient signal to G at $\approx$50% of training from `conv3d`-based discriminator of MoCoGAN-HD (upper row) and our one (lower row) at $t = 0, 2, 4, 6, 8, 12$ timesteps.

3. Model

Our model is based on the paradigm of *neural representations* [38, 58, 64], i.e. representing signals as neural networks. We treat each video as a function $\boldsymbol{x}_t = \boldsymbol{x}(t)$ which is continuous in time $t \in \mathbb{R}_+$. In this manner, the training dataset $\mathcal{D}$ is a set of subsampled signals $\mathcal{D} = \{\boldsymbol{x}^{(i)}\}_{i=1}^N = \{(\boldsymbol{x}^{(i)}_{t_0}, ..., \boldsymbol{x}^{(i)}_{t_{\ell_i}})\}_{i=1}^N$, where N denotes the total number of videos, t_j denotes the time position of the j-th frame and ℓ_i is the amount of frames in the i-th video.[2] Note that each video might have a different length ℓ_i and in practice these lengths vary a lot (see Appx E for datasets statistics). Our goal is to train a generative model over video signals, having only their subsampled versions. To achieve this, we develop the following framework.

We build the model on top of StyleGAN2 [28] and redesign its generator and discriminator networks for video synthesis with minimal modifications. Our generator is conceptually similar to MoCoGAN [67], i.e., we separate latent information into content code $\boldsymbol{z}^{\mathsf{c}}$ and motion trajectory $\boldsymbol{v}_t = \boldsymbol{v}(t)$. In contrast to MoCoGAN, our motion codes $\boldsymbol{v}_t$ are continuous in time $t \in \mathbb{R}_+$ and we describe their design in §3.1. The *only* modification we do on top of StyleGAN2's generator is the concatenation of our continuous motion codes $\boldsymbol{v}_t$ to its constant input tensor. The discriminator model D takes k frames $\boldsymbol{x}_{t_1}, ..., \boldsymbol{x}_{t_k}$ of a sparsely sampled video, independently extracts features

[2]To simplify the notation, we assume that all videos have the same frame-rate and that all the videos were sampled starting at $t = t_0$.

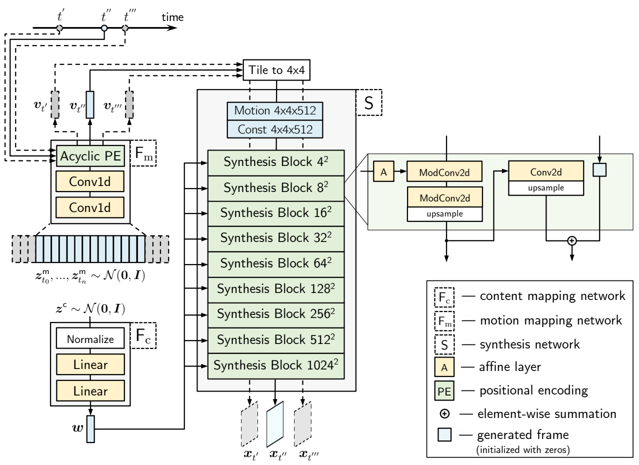

Figure 5. Generator architecture: the only change we do on top of StyleGAN2 generator's synthesis network S is the concatenation of our motion codes to the constant input tensor. S produces frames $\boldsymbol{x}_t$ non-autoregressively using the content code $\boldsymbol{w}$ and motion code $\boldsymbol{v}_t$.

$\boldsymbol{h}_{t_1}, ..., \boldsymbol{h}_{t_k}$ from them, concatenates those features together channel-wise into a global video descriptor $\boldsymbol{h}$ and predicts the real/fake class from it. We condition D on the time distances $\delta^x_i = t_{i+1} - t_i$ between frames to make it easier for it to operate on different frame rates.

3.1. Generator structure

Overview. Generator consists of three components: content mapping network F_c, motion mapping network F_m and synthesis network S. F_c and S are borrowed from StyleGAN2 and we only modify S by tiling and concatenating motion codes $\boldsymbol{v}_t$ to its constant input tensor.

A video is generated the following way. First, we sample the content noise $\boldsymbol{z}^\mathrm{c} \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I})$ and, following StyleGAN2, transform it into latent code $\boldsymbol{w} = \mathsf{F}_\mathrm{c}(\boldsymbol{z}^\mathrm{c}) \in \mathbb{R}^{512}$. It is shared for all timesteps $t \in \mathbb{R}_+$ of a video. Then, to generate a frame $\boldsymbol{x}_t$ in the specified time location t, we first compute its motion code $\boldsymbol{v}_t$, which is done in three steps. First, we sample a discrete sequence of equidistant trajectory noise $\boldsymbol{z}^\mathrm{m}_{t_0}, ..., \boldsymbol{z}^\mathrm{m}_{t_n} \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I})$ (we assume $t_0 = 0$ everywhere), positioned at distance $\delta^z = t_{i+1} - t_i$ from one another. The number of tokens n is determined by the condition $t < t_n$, i.e. it should be long enough to cover the desired timestep t.[3] Then, we process it with `conv1d`-based motion mapping network F_m with a large kernel size into the sequence $\boldsymbol{u}_{t_0}, ..., \boldsymbol{u}_{t_n}$. After that, we take a pair of tokens $\boldsymbol{u}_\ell, \boldsymbol{u}_r$ which t lies between (i.e. $\ell = t_i$ for some $i \in \{0, 1, ..., n\}$ and $r = t_{i+1}$) and compute an acyclic positional embedding $\boldsymbol{v}_t$ from them, described next. This positional embedding serves as the motion code for our generator. In fact, we do not need to sample all the motion noise vectors $\boldsymbol{z}^\mathrm{m}_{t_0}, ..., \boldsymbol{z}^\mathrm{m}_{t_n}$ to produce $\boldsymbol{v}_t$, but only those ones which $\boldsymbol{v}_t$ depends on. In this way, our generator can produce frames non-autoregressively.

[3]In practice, since F_m uses padding-less convolutions, this sequence is slightly larger. We elaborate on this in Appx B.

Acyclic positional encoding. Traditional positional embeddings [58, 64] are cyclic by default. This does not create problems in traditional applications (like image or scene representations) because utilized spatial domain there never exceeds the period length [38, 60]. But for video generation, cyclicity is not desirable, because it makes a video getting looped at some point. To solve this issue, we develop *acyclic* positional encoding.

A sine-based positional embedding vector $\boldsymbol{p} \in \mathbb{R}^d$ can be expressed in the following form:

$$\boldsymbol{p}(\boldsymbol{\alpha}, \boldsymbol{\omega}, \boldsymbol{\rho}, t) = \boldsymbol{\alpha} \odot \sin(\boldsymbol{\omega} \cdot t + \boldsymbol{\rho}), \tag{1}$$

where $\odot$ denotes element-wise vector multiplication, $\boldsymbol{\alpha}, \boldsymbol{\omega}, \boldsymbol{\rho} \in \mathbb{R}^d$ are amplitudes, periods and phases of the corresponding waves, and the sine function is applied element-wise. By default, these embeddings are periodic and always the same for any input [38, 58, 64], which is not desirable for video synthesis, where natural videos contain different motions and are typically aperiodic. To solve this issue, we compute the wave parameters from motion noise $\boldsymbol{z}^\mathrm{m}_{t_0}, ..., \boldsymbol{z}^\mathrm{m}_{t_n}, ...$ the following way. First, "raw" motion codes $\tilde{\boldsymbol{v}}_t$ are computed using wave parameters $\boldsymbol{\alpha}_\ell, \boldsymbol{\omega}_\ell, \boldsymbol{\rho}_\ell$ predicted from $\boldsymbol{u}_\ell$:

$$\tilde{\boldsymbol{v}}_t = \boldsymbol{\alpha}_\ell \odot \sin(\boldsymbol{\omega}_\ell \cdot t + \boldsymbol{\rho}_\ell), \tag{2}$$

where

$$\boldsymbol{\alpha}_\ell = W_\alpha \boldsymbol{u}_\ell, \quad \boldsymbol{\omega}_\ell = W_\omega \boldsymbol{u}_\ell, \quad \boldsymbol{\rho}_\ell = W_\rho \boldsymbol{u}_\ell, \tag{3}$$

and $W_\alpha, W_\omega, W_\rho \in \mathbb{R}^{d \times d}$ are learnable weight matrices. Using $\tilde{\boldsymbol{v}}_t$ directly as motion codes does not lead to good results since it contains discontinuities (see Fig 9d). That's why we "stitch" their start and end values via:

$$\boldsymbol{v}_t = \tilde{\boldsymbol{v}}_t - \mathrm{lerp}(\tilde{\boldsymbol{v}}_\ell, \tilde{\boldsymbol{v}}_r, t) + \mathrm{lerp}(W_a \boldsymbol{u}_\ell, W_a \boldsymbol{u}_r, t), \tag{4}$$

where $W_a \in \mathbb{R}^{d \times d}$ is a learnable weight matrix and $\texttt{lerp}(\boldsymbol{x}, \boldsymbol{y}, t)$ is the element-wise linear interpolation between $\boldsymbol{x}$ and $\boldsymbol{y}$ using the time position t. The first subtraction in Eq (4) alters the positional embeddings to make them converge to zero values at locations $\{t_0, t_1, ..., t_n, ...\}$. This limits the expressive power of the positional embeddings and that's why we add the "alignment" vectors $\boldsymbol{a} = W_a \boldsymbol{u}$ to restore it. See Fig 9e in Appx B for the visualization.

In practice, we found it useful to compute periods as:

$$\boldsymbol{\omega}_t = (\tanh(W_\omega \boldsymbol{u}_t) + \mathbb{1}) \odot \boldsymbol{\sigma}, \tag{5}$$

where $\mathbb{1}$ is a vector of ones and $\boldsymbol{\sigma}$ are linearly-spaced scaling coefficients. See Appx B and the source code for details.

One could try using continuous codes $\boldsymbol{u}_t = \texttt{lerp}(\boldsymbol{u}_\ell, \boldsymbol{u}_r, t)$ directly as motion codes instead of $\boldsymbol{v}_t$. This also eliminates cyclicity (in theory), but leads to poor results in practice: if the distance δ^z is small, then the motion trajectory will contain unnatural sharp transitions; and when δ^z is increased, G loses its ability to properly model high-frequency motions (like blinking) since the codes change too slowly. We empirically validate this in Tab 2 (also see samples on the project webpage).

3.2. Discriminator structure

Modern video generators typically utilize two separate discriminators which operate on image and video levels separately [13, 65, 67]. But since we train on extremely sparse videos and aim to have a computationally efficient model, we propose to use a *holistic* discriminator $\mathsf{D}(\boldsymbol{x}_{t_1}, ..., \boldsymbol{x}_{t_k})$, which is conditioned on the time distances between frames $\delta_i^x = t_{i+1} - t_i$. It consists of two parts: 1) feature extractor backbone D_b, which independently embeds an image frame $\boldsymbol{x}_t$ into a 3D feature vector $\boldsymbol{h}_{t_i} \in \mathbb{R}^{512\times16\times16}$; and the convolutional head D_h, which takes the concatenation of all the features $\boldsymbol{h} = \texttt{concat}[\boldsymbol{h}_{t_1}, ..., \boldsymbol{h}_{t_k}] \in \mathbb{R}^{512k\times16\times16}$ and outputs the real/fake logit $y \in \mathbb{R}$.

We input the time distances information $\delta_1^x, ..., \delta_{k-1}^x$ between k frames $\boldsymbol{x}_{t_1}, ..., \boldsymbol{x}_{t_k}$ into D the following way. First, we encode them with positional encoding, preprocess with a 2-layer MLP into $\boldsymbol{p}(\delta_1^x), ..., \boldsymbol{p}(\delta_{k-1}^x) \in R^d$ and concatenate into a single vector $\boldsymbol{p}_\delta \in \mathbb{R}^{(k-1)\cdot d}$. After that, we use the projection discriminator strategy [39] and compute the output logit as a simple dot product between $\boldsymbol{p}_\delta$ and the corresponding video feature vector. The overall architecture is visualized in Fig 6.

Such a design is *greatly* more efficient than using both image and video discriminators and provides a more informative learning signal to the generator (see Fig 4).

3.3. Implicit assumptions of sparse training

Consider the problem of learning a probability distribution $p(\boldsymbol{x}) = p(x_1, ..., x_n)$ and consider that we utilize sparse training, i.e. select k coordinates of vector $\boldsymbol{x}$ randomly on each iteration of the optimization process. Then the optimization objective is equivalent to learning all possible marginal distributions $p(x_{i_1}, ..., x_{x_k})$ instead of learning joint $p(\boldsymbol{x})$. When does learning marginals allow to obtain the full joint distribution at the end? The following simple statement adds some clarity to this question.

A trivial but serviceable statement. *Let's denote by $\mathcal{J}_{<i}^k$ a collection of sets J_i of up to k indices j s.t. $\forall J_i \in \mathcal{J}_{<i}^k$ we have $j < i$ for all $j \in J_i$. In other words, J_i is a set of up to k indices $j \in [1, i)$. Then, $p(\boldsymbol{x})$ can be represented as a product of n marginals $p(x_i, \boldsymbol{x}_{J_i})$ for $i \in [1, n]$ if and only if $\forall i$ there exists $J_i \in \mathcal{J}_{<i}^{k-1}$ s.t. $p(x_i|\boldsymbol{x}_{<i}) \equiv p(x_i|\boldsymbol{x}_{J_i})$.*

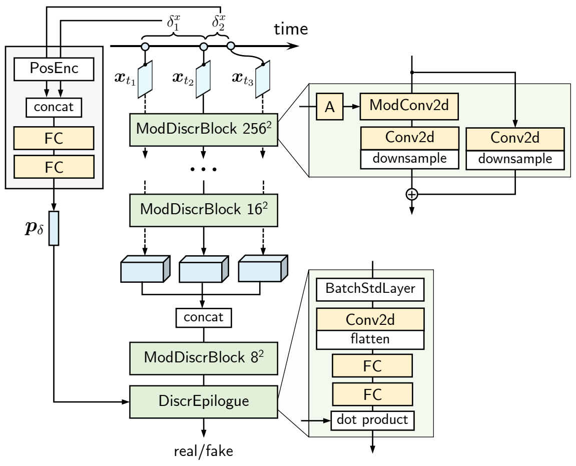

Figure 6. Discriminator architecture for $k = 3$ frames per video. The only changes we do on top of the StyleGAN2 [30] discriminator are concatenating activations channel-wise at the 16^2 resolution and conditioning the model on the positional embeddings of the time distances between frames.

The above statement is primitive (see the proof in Appx F) but can provide useful practical intuition. For video synthesis, it implies that one can learn a video generator by using only k frames per video only if for any frame $\boldsymbol{x}_i$, there exists *at most* $k-1$ previous frames sufficient to properly predict it (see Appx F). And we argue that very few frames suffice to make such a prediction for the modern video synthesis benchmarks. For example, in SkyTimelapse [78], the motions are typically unidirectional and thus easily predictable from only 2 previous frames, which corresponds to training with $k = 3$ frames per video.

We treat videos as infinite continuous signals, but in practice one has to set a limit on the maximum time location T which can be seen during training. To the best of our knowledge, previous methods use at most $T = 64$ [10, 55], but in our case we easily train the model with $T = 1024$ since our generator is non-autoregressive and our discriminator uses only the relative temporal information. We set the maximum distance between t_1 and t_k to 32 to cover short and medium-term movements: otherwise, we observed unstable training and abrupt motions. To sample frames, we first sample the distance $(t_k - t_1) \sim U[k-1, 32]$ between them, and then sample the offset $t_1 \sim U[0, T - t_k]$. After that, frames locations t_i for $i \in \{2, ..., k-1\}$ are selected at random without repetitions.

4. Experiments

Datasets. We test our model on 5 benchmarks: FaceForensics 256^2 [53], SkyTimelapse 256^2 [78], UCF101 256^2 [62], RainbowJelly 256^2 (introduced by us and described in Appx E) and MEAD 1024^2 [72]. We used the

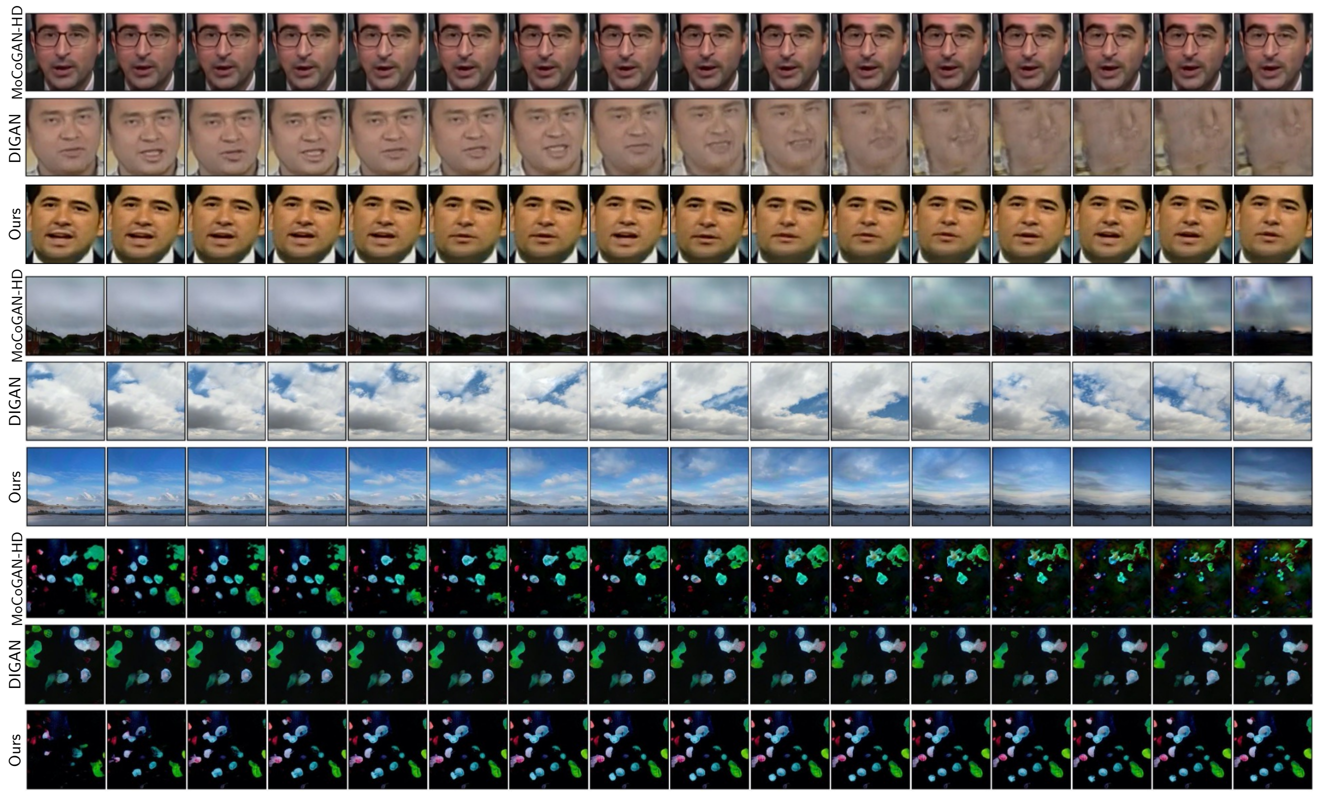

Figure 7. *Random* samples from the existing methods on FaceForensics 256^2, SkyTimelapse 256^2 and RainbowJelly 256^2, respectively. We sample a 64-frames video and display each 4-th frame, starting from $t = 0$.

`train` splits (when available) for all the datasets except for UCF101, where we used `train+test` splits. We provide the datasets details in Appx E.

Evaluation. Following prior work, we use Frechet Video Distance (FVD) [68] and Inception Score (IS) as our evaluation metrics with FVD being the main one since FID (its image-based counterpart) better aligns with human-perceived quality [23]. We use two versions of FVD: FVD_{16} and FVD_{128}, which use 16 and 128-frames-long videos to compute the statistics. Inception Score is used only to evaluate the generation quality on UCF-101 since it uses a UCF-101-finetuned C3D model [54].

The official FVD implementation [68] does not provide a *complete* evaluation pipeline, but rather an inference script for a single batch of videos, which are required to be *already resized to* 256^2 *and loaded into memory*. This creates discrepancies in the evaluation protocols used by previous works since FVD (similar to FID [46]) is very sensitive to the subsampling and data processing procedures. We implement, document (see Appx C) and release a complete FVD evaluation protocol and use it to evaluate all the methods.

Baselines. We use 5 baselines for comparison: MoCoGAN [67], MoCoGAN [67] with the StyleGAN2 [30] backbone, VideoGPT [79], MoCoGAN-HD [65] and DIGAN [80]. For MoCoGAN with the StyleGAN2 backbone (denoted as MoCoGAN-SG2), we replaced its generator and image-based discriminator with the corresponding StyleGAN2's components, leaving its video discriminator unchanged. We also used the training scheme and regularizations from StyleGAN2. MoCoGAN was trained for 5 days on a single GPU since its lightweight DC-GAN [50] backbone makes it fast to train, while MoCoGAN+SG2 was trained for 2 days on $\times 4$ GPUs to reach 25M real images seen by its image-based discriminator. MoCoGAN-HD is trained for $\approx$4.5 days on $\times 4$ v100 GPUs, as specified in the original paper (Appx B of [65]). We trained VideoGPT for the maximum affordable total time of 32 GPU-days in our resource constraints. DIGAN [80] was trained for $\approx$4 days since after that its FVD score either did not change or exploded (on RainbowJelly). We also replaced its weighted sampling strategy (selecting clips from longer videos with higher probabilities) with the uniform one, which is used by other methods [65, 67, 79]. For each method, we used the checkpoint with the lowest FVD_{16} value.

4.1. Main experiments

For the main evaluation, we train our method and all the baselines from scratch on the described 256^2 datasets.

Table 1. Quantitative performance and training cost of different methods. We trained *all* the methods from scratch on 256^2 resolution datasets using the official codebases and evaluated them under the unified evaluation protocol (see §4). Training was done on $\times 4$ 32 GB NVidia V100 GPUs for all the methods except VideoGPT, which was trained on $\times 4$ NVidia A6000 GPUs (with 48.5 GB of memory each) due to its high memory consumption. For 2-stage methods, we report their training cost in the "$X + Y$" format. [†]VideoGPT was trained for our maximum resource constraint of 32 GPU-days which was detrimental to its performance on 256^2 resolution. Vanilla StyleGAN2 training time on 256^2 resolution (with mixed precision and optimizations [28]) is 7.72 GPU-days in our environment.

Method	FaceForensics		SkyTimelapse		UCF101		RainbowJelly		Training cost
	FVD_{16}	FVD_{128}	FVD_{16}	FVD_{128}	FVD_{16}	FVD_{128}	FVD_{16}	FVD_{128}	(GPU-days)
MoCoGAN [67]	124.7	257.3	206.6	575.9	2886.9	3679.0	1572.9	549.7	**5**
+ StyleGAN2 backbone	55.62	309.3	85.88	272.8	1821.4	2311.3	638.5	463.0	8
MoCoGAN-HD [65]	111.8	653.0	164.1	878.1	1729.6	2606.5	579.1	628.2	7.5 + 9
VideoGPT [79][†]	185.9	N/A	222.7	N/A	2880.6	N/A	**136.0**	N/A	16 + 16
DIGAN [80]	62.5	1824.7	83.11	**196.7**	1630.2	2293.7	436.6	369.0	16
StyleGAN-V (ours)	**47.41**	**89.34**	**79.52**	197.0	**1431.0**	**1773.4**	195.4	**262.5**	8

Each model is trained on $\times 4$ NVidia V100 32 GB GPUs, except for VideoGPT, which is very demanding in terms of GPU memory for 256^2 resolution and we had to train it on $\times 4$ NVidia A6000 instead (with the overall batch size of *4*). For our method and MoCoGAN+SG2, we use exactly the same optimization scheme as StyleGAN2, including the loss function, Adam optimizer hyperparameters and R1 regularization [36]. We reduce the learning rate by 10 for the D_V module of MoCoGAN+SG2 since it does not have equalized learning rate [27]. We use $\delta^z = 16$ for all the experiments except for SkyTimelapse, where we used $\delta^z = 256$. See other training details in Appx B. We evaluate all the methods under the same evaluation protocol, described in Appx C and report the results in Table 1.

To measure the efficiency, we use the amount of GPU days required to train a method. We build on top of the official StyleGAN2 implementation.[4] The training cost of the image-based StyleGAN2 to reach its specified 25M images is 7.72 NVidia V100 GPU-days in our environment. StyleGAN-V is trained for 2 days, which corresponds to ≈23M real frames seen by the discriminator. MoCoGAN-HD is built on top of `stylegan2-pytorch`'s codebase[5], which is ≈2 times slower than the highly optimized NVidia's implementation. That's why in Table 1 we report its training cost *reduced* by a factor of 2 to account for this.

Our method significantly outperforms the existing ones on almost all the benchmarks in terms FVD_{16} and FVD_{128}. We visualize the samples in Fig 1 and Fig 7. Our method is able to generate hour-long plausibly looking videos, though the motion diversity and global motion coherence for them would be limited (see Appx A). MoCoGAN-HD suffers from the LSTM instability when unrolled to large lengths and does not produce diverse motions. DIGAN produces high-quality videos on SkyTimelapse because its inductive bias of having joint spatio-temporal positional information is well suited for videos that have an entire scene moving. But for FaceForensics, this leads to a "head flying away" effect (see Appx H). To generate 1-hour long videos from MoCoGAN-HD, we unroll its LSTM model to the required depth (≈90k steps) and synthesize frames only in the necessary time positions, while DIGAN, similar to our method, is able to generate frames non-autoregressively.

[4]https://github.com/NVlabs/stylegan2-ada-pytorch

[5]https://github.com/rosinality/stylegan2-pytorch

4.2. Ablations

To ablate the core components, we replaced G or D modules with their MoCoGAN+SG2 counterparts. In the both cases, their removal leads to poor short-term and long-term video quality, as specified by the corresponding metrics in Table 2 and video samples in the supplementary.

Replacing continuous motion codes $\boldsymbol{v}(t)$ with $\boldsymbol{u}(t)$, produced by LSTM hurts the performance, especially when the distance δ^z between motion codes is small. This happens due to unnaturally abrupt transitions between frames and we provide the corresponding samples in the supplementary. The corresponding results are in Table 2.

We also verify the importance of the conditioning in D and denote the experiment where it's disabled as "w/o time conditioning" in Table 2. Removing the time conditioning hurts the performance, because it constrains the ability of D to understand the temporal scale it is currently operating on.

An important design choice is how many samples per video one should use during training. We try different values of k for $k = 2, 3, 4, 8$ and 16 and report the corresponding results in Table 3. As being discussed in §3.3, for existing video generation benchmarks, it might be enough to sample only several frames per each video, and our experiments confirm this observation. The performance is decreased for larger k, but this might be attributed to a weaker temporal aggregation procedure of D, which simply concatenates features together. It is surprising to see that modern datasets can be fit with as few as 2 samples per video.

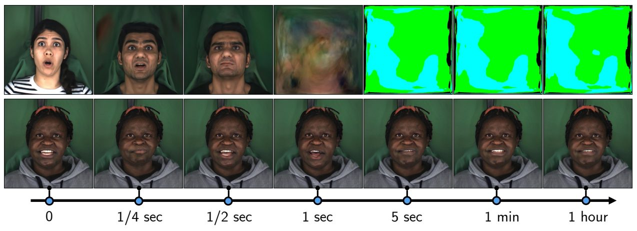

Figure 8. Generations on MEAD 1024^2 [72] for MoCoGAN-HD [65] and our method. MoCoGAN-HD cannot preserve the identity and diverges for long LSTM unrolling. (Note that all videos in the dataset have static head positions — see Appx E).

4.3. Properties

Our generator is able to generate arbitrarily long videos. Our design of motion codes allows StyleGAN-V not to suffer from stability problems when unrolled to large (potentially infinite) video lengths. This is verified by visualizing the video clips for the extremely large timesteps in Fig 1 and Fig 8. We also demonstrate its ability to produce videos in arbitrarily high frame-rate in the supplementary.

Our model has the same latent space manipulation properties as StyleGAN2. To show this, we conduct two experiments: embedding, editing and animating an off-the-shelf image and editing and animating the first frame of a generated video. To embed an image, we used the optimization procedure similar to [1], but considering it to be positioned at $t = 0$. To edit an image with CLIP, we used the procedure of [47]. The results of these experiments are visualized in Fig 2 and we provide the details in Appx B and more examples in the supplementary. Apart from showing the good properties of its latent space, these experiments demonstrate the extrapolation potential of our generator.

StyleGAN-V has almost the same training efficiency and image quality as StyleGAN2. In Fig 3, we plot the FID scores (computed from 16-frames videos) and training costs of modern video generators on FaceForensics 256^2 by their corresponding FVD_{16} scores. Our method comes very close to StyleGAN2: it converges to FID of 9.44 in 8 GPU-days compared to FID of 8.42 in 7.72 GPU-days for StyleGAN2, which is only $\approx$10% worse. This raises the question whether video generators can be as computationally efficient and good in terms of image quality as image ones.

Our model is the first one which is directly trainable on 1024^2 resolution. We provide the generations on MEAD 1024^2 for our method and for MoCoGAN-HD. MoCoGAN-HD cannot preserve the identity of a speaker and diverges for large video lengths, while our method achieves comparable image quality and coherent motions. For this dataset, our model was trained for 7 days on $\times 4$ NVidia v100 GPUs and obtained FID of 24.12 and FVD_{16} of 156.1. Image generator for MoCoGAN-HD was trained for 14 days on $\times 4$ A6000 GPUs, while its video generator was trained for only 5 days since it didn't require high-resolution training.

Table 2. Ablating architectural components of our model.

Method	FaceForensics 256^2 FVD_{16}	FaceForensics 256^2 FVD_{128}	SkyTimelapse 256^2 FVD_{16}	SkyTimelapse 256^2 FVD_{128}
Default StyleGAN-V	47.41	89.34	79.52	197.0
w/o our G	65.88	41.77	109.1	240.2
w/o our D	154.0	139.1	236.9	258.0
w/o time conditioning in D	95.4	236.0	102.1	210.3
w LSTM codes, $\delta^z = 1$	131.9	159.1	135.7	196.1
w LSTM codes, $\delta^z = 16$	180.3	94.55	95.71	165.8

Table 3. Ablating the amount of frames k per clip used during training. Sparse training provides better results for our method.

Number of frames	FaceForensics 256^2 FVD_{16}	FaceForensics 256^2 FVD_{128}	SkyTimelapse 256^2 FVD_{16}	SkyTimelapse 256^2 FVD_{128}
$k = 2$	60.41	93.5	50.5	209.9
$k = 3$ (default)	47.41	89.34	79.52	197.0
$k = 4$	51.84	114.9	65.7	194.5
$k = 8$	101.9	211.4	73.12	215.9
$k = 16$	92.52	192.8	107.6	254.3

Our discriminator provides more informative learning signal to G. Fig 4 visualizes the gradient signal to the generator from our discriminator and the `conv3d`-based video discriminator of MoCoGAN-HD, measured at $\approx$50% of training for our method (at 10M images seen by D) and MoCoGAN-HD (at the 300-th epoch). In our case, one can easily see fine-grained details of the face structure, perceived by D, while in case of MoCoGAN-HD, most of the gradient is redundant and lack any structural information.

Content and motion decomposition. Similar to MoCoGAN [67], our generator captures content and motion variations in a disentangled manner: altering motion codes $z^{\text{m}}_{t_0}, ..., z^{\text{m}}_{t_n}$ while fixing z^{c} does not change the appearance variations (like, a speaker's identity). Similarly, resampling z^{c} does not influence motion patterns on a video, but only its content. We provide the corresponding visualizations on the project website.

5. Conclusion

In this work, we provided a different perspective on time for video synthesis and built a continuous video generator using the paradigm of neural representations. For this, we developed motion representations through the lens of positional embeddings, explored sparse training of video generators and redesigned a typical dual structure of a video discriminator. Our model is built on top of StyleGAN2 and features a lot of its perks, like efficient training, good image quality and editable latent space. We hope that our work would serve as a solid basis for building more powerful video generators in the future. The limitations and potential negative impact are discussed in Appx A.

References

[1] Rameen Abdal, Yipeng Qin, and Peter Wonka. Image2stylegan: How to embed images into the stylegan latent space? In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 4432–4441, 2019. 8, 12, 13

[2] Dinesh Acharya, Zhiwu Huang, Danda Pani Paudel, and Luc Van Gool. Towards high resolution video generation with progressive growing of sliced wasserstein gans. *arXiv preprint arXiv:1810.02419*, 2018. 3

[3] Abhishek Aich, Akash Gupta, Rameswar Panda, Rakib Hyder, M Salman Asif, and Amit K Roy-Chowdhury. Non-adversarial video synthesis with learned priors. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6090–6099, 2020. 2, 3

[4] Ivan Anokhin, Kirill Demochkin, Taras Khakhulin, Gleb Sterkin, Victor Lempitsky, and Denis Korzhenkov. Image generators with conditionally-independent pixel synthesis. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 14278–14287, 2021. 3, 13

[5] Rajat Arora and Yong Jae Lee. Singan-gif: Learning a generative video model from a single gif. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, pages 1310–1319, January 2021. 3

[6] Wenbo Bao, Wei-Sheng Lai, Chao Ma, Xiaoyun Zhang, Zhiyong Gao, and Ming-Hsuan Yang. Depth-aware video frame interpolation. In *IEEE Conferene on Computer Vision and Pattern Recognition*, 2019. 3

[7] Piotr Bojanowski, Armand Joulin, David Lopez-Paz, and Arthur Szlam. Optimizing the latent space of generative networks. *arXiv preprint arXiv:1707.05776*, 2017. 3

[8] Andrew Brock, Jeff Donahue, and Karen Simonyan. Large scale gan training for high fidelity natural image synthesis. *arXiv preprint arXiv:1809.11096*, 2018. 1

[9] Joao Carreira and Andrew Zisserman. Quo vadis, action recognition? a new model and the kinetics dataset. In *proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6299–6308, 2017. 13

[10] Lluis Castrejon, Nicolas Ballas, and Aaron Courville. Hierarchical video generation for complex data. *arXiv preprint arXiv:2106.02719*, 2021. 3, 5

[11] Eric R Chan, Marco Monteiro, Petr Kellnhofer, Jiajun Wu, and Gordon Wetzstein. pi-gan: Periodic implicit generative adversarial networks for 3d-aware image synthesis. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5799–5809, 2021. 3

[12] Hao Chen, Bo He, Hanyu Wang, Yixuan Ren, Ser-Nam Lim, and Abhinav Shrivastava. Nerv: Neural representations for videos. *arXiv preprint arXiv:2110.13903*, 2021. 3

[13] Aidan Clark, Jeff Donahue, and Karen Simonyan. Adversarial video generation on complex datasets. *arXiv preprint arXiv:1907.06571*, 2019. 1, 2, 3, 5

[14] Yilun Du, Katherine M. Collins, Joshua B. Tenenbaum, and Vincent Sitzmann. Learning signal-agnostic implicit manifolds. In *Thirty-Fifth Conference on Neural Information Processing Systems*, 2021. 3

[15] Emilien Dupont, Yee Whye Teh, and Arnaud Doucet. Generative models as distributions of functions. *arXiv preprint arXiv:2102.04776*, 2021. 3

[16] Chelsea Finn, Ian Goodfellow, and Sergey Levine. Unsupervised learning for physical interaction through video prediction. *Advances in neural information processing systems*, 29:64–72, 2016. 3

[17] Gereon Fox, Ayush Tewari, Mohamed Elgharib, and Christian Theobalt. Stylevideogan: A temporal generative model using a pretrained stylegan. *arXiv preprint arXiv:2107.07224*, 2021. 3

[18] Kyle Genova, Forrester Cole, Daniel Vlasic, Aaron Sarna, William T Freeman, and Thomas Funkhouser. Learning shape templates with structured implicit functions. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 7154–7164, 2019. 3

[19] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. *Advances in neural information processing systems*, 27, 2014. 3

[20] Shir Gur, Sagie Benaim, and Lior Wolf. Hierarchical patch vae-gan: Generating diverse videos from a single sample, 2020. 3

[21] Niv Haim, Ben Feinstein, Niv Granot, Assaf Shocher, Shai Bagon, Tali Dekel, and Michal Irani. Diverse generation from a single video made possible. *arXiv preprint arXiv:2109.08591*, 2021. 3

[22] Amir Hertz, Or Perel, Raja Giryes, Olga Sorkine-hornung, and Daniel Cohen-or. SAPE: Spatially-adaptive progressive encoding for neural optimization. In *Thirty-Fifth Conference on Neural Information Processing Systems*, 2021. 13, 15

[23] Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. Gans trained by a two time-scale update rule converge to a local nash equilibrium. *Advances in neural information processing systems*, 30, 2017. 2, 6

[24] Huaizu Jiang, Deqing Sun, Varun Jampani, Ming-Hsuan Yang, Erik Learned-Miller, and Jan Kautz. Super slomo: High quality estimation of multiple intermediate frames for video interpolation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 9000–9008, 2018. 3

[25] Emmanuel Kahembwe and Subramanian Ramamoorthy. Lower dimensional kernels for video discriminators. *Neural Networks*, 132:506–520, 2020. 3

[26] Nal Kalchbrenner, Aäron Oord, Karen Simonyan, Ivo Danihelka, Oriol Vinyals, Alex Graves, and Koray Kavukcuoglu. Video pixel networks. In *International Conference on Machine Learning*, pages 1771–1779. PMLR, 2017. 3

[27] Tero Karras, Timo Aila, Samuli Laine, and Jaakko Lehtinen. Progressive growing of gans for improved quality, stability, and variation. *arXiv preprint arXiv:1710.10196*, 2017. 7, 12

[28] Tero Karras, Miika Aittala, Janne Hellsten, Samuli Laine, Jaakko Lehtinen, and Timo Aila. Training generative adversarial networks with limited data. *arXiv preprint arXiv:2006.06676*, 2020. 1, 3, 7, 12, 13, 15

[29] Tero Karras, Samuli Laine, and Timo Aila. A style-based generator architecture for generative adversarial networks. In

Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pages 4401–4410, 2019. 12, 14

[30] Tero Karras, Samuli Laine, Miika Aittala, Janne Hellsten, Jaakko Lehtinen, and Timo Aila. Analyzing and improving the image quality of stylegan. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8110–8119, 2020. 2, 5, 6, 13

[31] Adam R Kosiorek, Heiko Strathmann, Daniel Zoran, Pol Moreno, Rosalia Schneider, Soňa Mokrá, and Danilo J Rezende. Nerf-vae: A geometry aware 3d scene generative model. *arXiv preprint arXiv:2104.00587*, 2021. 3

[32] Zhengqi Li, Simon Niklaus, Noah Snavely, and Oliver Wang. Neural scene flow fields for space-time view synthesis of dynamic scenes. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6498–6508, 2021. 3

[33] Gidi Littwin and Lior Wolf. Deep meta functionals for shape representation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 1824–1833, 2019. 3

[34] Ce Liu, Jenny Yuen, and Antonio Torralba. Sift flow: Dense correspondence across scenes and its applications. *IEEE transactions on pattern analysis and machine intelligence*, 33(5):978–994, 2010. 2

[35] Michael Mathieu, Camille Couprie, and Yann LeCun. Deep multi-scale video prediction beyond mean square error. *arXiv preprint arXiv:1511.05440*, 2015. 3

[36] Lars Mescheder, Andreas Geiger, and Sebastian Nowozin. Which training methods for gans do actually converge? In *International conference on machine learning*, pages 3481–3490. PMLR, 2018. 7

[37] Lars Mescheder, Michael Oechsle, Michael Niemeyer, Sebastian Nowozin, and Andreas Geiger. Occupancy networks: Learning 3d reconstruction in function space. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4460–4470, 2019. 3

[38] Ben Mildenhall, Pratul P Srinivasan, Matthew Tancik, Jonathan T Barron, Ravi Ramamoorthi, and Ren Ng. Nerf: Representing scenes as neural radiance fields for view synthesis. In *European conference on computer vision*, pages 405–421. Springer, 2020. 2, 3, 4, 13, 19

[39] Takeru Miyato and Masanori Koyama. cgans with projection discriminator. *arXiv preprint arXiv:1802.05637*, 2018. 5, 18

[40] Andres Munoz, Mohammadreza Zolfaghari, Max Argus, and Thomas Brox. Temporal shift gan for large scale video generation. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 3179–3188, 2021. 3

[41] Michael Niemeyer, Lars Mescheder, Michael Oechsle, and Andreas Geiger. Differentiable volumetric rendering: Learning implicit 3d representations without 3d supervision. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3504–3515, 2020. 3

[42] Simon Niklaus, Long Mai, and Feng Liu. Video frame interpolation via adaptive separable convolution. In *IEEE International Conference on Computer Vision*, 2017. 3

[43] Jeong Joon Park, Peter Florence, Julian Straub, Richard Newcombe, and Steven Lovegrove. Deepsdf: Learning continuous signed distance functions for shape representation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 165–174, 2019. 3

[44] Keunhong Park, Utkarsh Sinha, Jonathan T Barron, Sofien Bouaziz, Dan B Goldman, Steven M Seitz, and Ricardo Martin-Brualla. Deformable neural radiance fields. *arXiv preprint arXiv:2011.12948*, 2020. 3, 13, 15

[45] Sunghyun Park, Kangyeol Kim, Junsoo Lee, Jaegul Choo, Joonseok Lee, Sookyung Kim, and Edward Choi. Vid-ode: Continuous-time video generation with neural ordinary differential equation. *arXiv preprint arXiv:2010.08188*, page online, 2021. 3

[46] Gaurav Parmar, Richard Zhang, and Jun-Yan Zhu. On buggy resizing libraries and surprising subtleties in fid calculation. *arXiv preprint arXiv:2104.11222*, 2021. 2, 6, 13, 14

[47] Or Patashnik, Zongze Wu, Eli Shechtman, Daniel Cohen-Or, and Dani Lischinski. Styleclip: Text-driven manipulation of stylegan imagery. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 2085–2094, 2021. 8, 13

[48] Songyou Peng, Michael Niemeyer, Lars Mescheder, Marc Pollefeys, and Andreas Geiger. Convolutional occupancy networks. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part III 16*, pages 523–540. Springer, 2020. 3

[49] Albert Pumarola, Enric Corona, Gerard Pons-Moll, and Francesc Moreno-Noguer. D-nerf: Neural radiance fields for dynamic scenes. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10318–10327, 2021. 3

[50] Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv preprint arXiv:1511.06434*, 2015. 6

[51] Ruslan Rakhimov, Denis Volkhonskiy, Alexey Artemov, Denis Zorin, and Evgeny Burnaev. Latent video transformer. *arXiv preprint arXiv:2006.10704*, 2020. 3

[52] MarcAurelio Ranzato, Arthur Szlam, Joan Bruna, Michael Mathieu, Ronan Collobert, and Sumit Chopra. Video (language) modeling: a baseline for generative models of natural videos. *arXiv preprint arXiv:1412.6604*, 2014. 3

[53] Andreas Rössler, Davide Cozzolino, Luisa Verdoliva, Christian Riess, Justus Thies, and Matthias Nießner. Faceforensics: A large-scale video dataset for forgery detection in human faces. *arXiv*, 2018. 2, 5, 12, 16, 19

[54] Masaki Saito, Eiichi Matsumoto, and Shunta Saito. Temporal generative adversarial nets with singular value clipping. In *Proceedings of the IEEE international conference on computer vision*, pages 2830–2839, 2017. 1, 3, 6, 15

[55] Masaki Saito, Shunta Saito, Masanori Koyama, and Sosuke Kobayashi. Train sparsely, generate densely: Memory-efficient unsupervised training of high-resolution temporal gan. *International Journal of Computer Vision*, 128:2586–2606, 2020. 1, 2, 3, 5

[56] Katja Schwarz, Yiyi Liao, Michael Niemeyer, and Andreas Geiger. Graf: Generative radiance fields for 3d-aware image synthesis. *arXiv preprint arXiv:2007.02442*, 2020. 3

[57] Aliaksandr Siarohin, Stéphane Lathuilière, Sergey Tulyakov, Elisa Ricci, and Nicu Sebe. First order motion model for image animation. *Advances in Neural Information Processing Systems*, 32:7137–7147, 2019. 15

[58] Vincent Sitzmann, Julien Martel, Alexander Bergman, David Lindell, and Gordon Wetzstein. Implicit neural representations with periodic activation functions. *Advances in Neural Information Processing Systems*, 33, 2020. 2, 3, 4, 13, 19

[59] Vincent Sitzmann, Michael Zollhöfer, and Gordon Wetzstein. Scene representation networks: Continuous 3d-structure-aware neural scene representations. In *Advances in Neural Information Processing Systems*, 2019. 3

[60] Ivan Skorokhodov, Savva Ignatyev, and Mohamed Elhoseiny. Adversarial generation of continuous images. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10753–10764, 2021. 3, 4, 12, 13, 15, 18

[61] Ivan Skorokhodov, Grigorii Sotnikov, and Mohamed Elhoseiny. Aligning latent and image spaces to connect the unconnectable. *arXiv preprint arXiv:2104.06954*, 2021. 3, 12, 13

[62] Khurram Soomro, Amir Roshan Zamir, and Mubarak Shah. Ucf101: A dataset of 101 human actions classes from videos in the wild. *arXiv preprint arXiv:1212.0402*, 2012. 2, 5, 16, 19

[63] Nitish Srivastava, Elman Mansimov, and Ruslan Salakhudinov. Unsupervised learning of video representations using lstms. In *International conference on machine learning*, pages 843–852. PMLR, 2015. 3

[64] Matthew Tancik, Pratul P Srinivasan, Ben Mildenhall, Sara Fridovich-Keil, Nithin Raghavan, Utkarsh Singhal, Ravi Ramamoorthi, Jonathan T Barron, and Ren Ng. Fourier features let networks learn high frequency functions in low dimensional domains. *arXiv preprint arXiv:2006.10739*, 2020. 2, 3, 4, 13, 19

[65] Yu Tian, Jian Ren, Menglei Chai, Kyle Olszewski, Xi Peng, Dimitris N. Metaxas, and Sergey Tulyakov. A good image generator is what you need for high-resolution video synthesis. In *International Conference on Learning Representations*, 2021. 1, 2, 3, 5, 6, 7, 8, 15

[66] Du Tran, Lubomir Bourdev, Rob Fergus, Lorenzo Torresani, and Manohar Paluri. Learning spatiotemporal features with 3d convolutional networks. In *Proceedings of the IEEE international conference on computer vision*, pages 4489–4497, 2015. 15

[67] Sergey Tulyakov, Ming-Yu Liu, Xiaodong Yang, and Jan Kautz. Mocogan: Decomposing motion and content for video generation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1526–1535, 2018. 1, 2, 3, 5, 6, 7, 8, 15

[68] Thomas Unterthiner, Sjoerd van Steenkiste, Karol Kurach, Raphael Marinier, Marcin Michalski, and Sylvain Gelly. Towards accurate generative models of video: A new metric & challenges. *arXiv preprint arXiv:1812.01717*, 2018. 2, 6, 14

[69] Carl Vondrick, Hamed Pirsiavash, and Antonio Torralba. Generating videos with scene dynamics. *Advances in neural information processing systems*, 29:613–621, 2016. 3

[70] Jacob Walker, Abhinav Gupta, and Martial Hebert. Patch to the future: Unsupervised visual prediction. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pages 3302–3309, 2014. 2

[71] Jacob Walker, Ali Razavi, and Aäron van den Oord. Predicting video with vqvae. *arXiv preprint arXiv:2103.01950*, 2021. 3

[72] Kaisiyuan Wang, Qianyi Wu, Linsen Song, Zhuoqian Yang, Wayne Wu, Chen Qian, Ran He, Yu Qiao, and Chen Change Loy. Mead: A large-scale audio-visual dataset for emotional talking-face generation. In *European Conference on Computer Vision*, pages 700–717. Springer, 2020. 2, 5, 8, 16, 19

[73] Ting-Chun Wang, Ming-Yu Liu, Jun-Yan Zhu, Guilin Liu, Andrew Tao, Jan Kautz, and Bryan Catanzaro. Video-to-video synthesis. *arXiv preprint arXiv:1808.06601*, 2018. 3

[74] Yaohui Wang, Piotr Bilinski, Francois Bremond, and Antitza Dantcheva. G3AN: Disentangling appearance and motion for video generation. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020. 3

[75] Yaohui Wang, Francois Bremond, and Antitza Dantcheva. Inmodegan: Interpretable motion decomposition generative adversarial network for video generation. *arXiv preprint arXiv:2101.03049*, 2021. 3

[76] Dirk Weissenborn, Oscar Täckström, and Jakob Uszkoreit. Scaling autoregressive video models. In *International Conference on Learning Representations*, 2020. 3

[77] Wenqi Xian, Jia-Bin Huang, Johannes Kopf, and Changil Kim. Space-time neural irradiance fields for free-viewpoint video. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9421–9431, 2021. 3

[78] Wei Xiong, Wenhan Luo, Lin Ma, Wei Liu, and Jiebo Luo. Learning to generate time-lapse videos using multi-stage dynamic generative adversarial networks. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018. 2, 5, 12, 16, 19

[79] Wilson Yan, Yunzhi Zhang, Pieter Abbeel, and Aravind Srinivas. Videogpt: Video generation using vq-vae and transformers. *arXiv preprint arXiv:2104.10157*, 2021. 3, 6, 7, 13, 15

[80] Sihyun Yu, Jihoon Tack, Sangwoo Mo, Hyunsu Kim, Junho Kim, Jung-Woo Ha, and Jinwoo Shin. Generating videos with dynamics-aware implicit generative adversarial networks. In *International Conference on Learning Representations*, 2022. 1, 3, 6, 7, 12, 15, 17

[81] Richard Zhang, Phillip Isola, Alexei A Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *CVPR*, 2018. 13