CCVS: Context-aware Controllable Video Synthesis

Anonymous Author(s)
Affiliation
Address
email

Abstract

This presentation introduces a self-supervised learning approach to the synthesis of new videos clips from old ones, with several new key elements for improved spatial resolution and realism: It conditions the synthesis process on contextual information for temporal continuity and ancillary information for fine control. The prediction model is doubly autoregressive, in the latent space of an autoencoder for forecasting, and in image space for updating contextual information, which is also used to enforce spatio-temporal consistency through a learnable optical flow module. Adversarial training of the autoencoder in the appearance and temporal domains is used to further improve the realism of its output. A quantizer inserted between the encoder and the transformer in charge of forecasting future frames in latent space (and its inverse inserted between the transformer and the decoder) adds even more flexibility by affording simple mechanisms for handling multimodal ancillary information for controlling the synthesis process (*e.g.*, a few sample frames, an audio track, a trajectory in image space) and taking into account the intrinsically uncertain nature of the future by allowing multiple predictions. Experiments with an implementation of the proposed approach give very good qualitative and quantitative results on multiple tasks and standard benchmarks.

1 Introduction

Feeding machines with extensive video content, and teaching them to create new samples on their own, may deepen their understanding of both the physical and social worlds. Video synthesis has numerous applications from content creation (*e.g.*, deblurring, slow motion) to human-robot interaction (*e.g.*, motion prediction). Despite the photo-realistic results of modern image synthesis models [32], video synthesis is still lagging behind due to the increased complexity of the additional temporal dimension.

An emerging trend is to use autoregressive models, for example transformer architectures [58], for their simplicity, and their ability to model long-range dependencies and learn from large volumes of data [3, 14]. The strength of transformers is grounded in a self-attention mechanism which considers all pairwise interactions within the data. The price to pay is a computational complexity which grows quadratically with the data size, which itself depends linearly on the temporal dimension and quadratically on the spatial resolution in the image domain. Although there have been some efforts to reduce the complexity of self-attention [10, 33, 44], using such methods directly on visual data is still limited to low resolutions and impractical without considerable computational power [8, 67].

Some recent works [17, 46] address this problem by using an autoencoder to compress the visual data, and apply the autoregressive model in the latent space. This greatly reduces the memory footprint and computational cost, yet, the greater the compression, the harder it is to faithfully reconstruct frames. The corresponding trade-offs may undermine the practical usability of these approaches. GANs [22] mitigate this issue by "hallucinating" plausible local details in image synthesis [17]. But latent video transformers [46] decode frames independently, which prevents local details from being temporally coherent. Hence, using GANs in this setting may result in flickering effects in the synthesized videos.

Submitted to 35th Conference on Neural Information Processing Systems (NeurIPS 2021). Do not distribute.

We follow the problem decomposition from [17, 46], but introduce a more elaborate compression strategy with CCVS (for *Context-aware Controllable Video Synthesis*), an approach that takes advantage of "context" frames (*i.e.*, both input images and previously synthesized ones) to faithfully reconstruct new ones despite lossy compression. As shown in Figure 1, CCVS relies on optical flow estimation between context and new frames, within temporal skip connections, to let information be shared across timesteps. New content, which cannot be retrieved from context, is synthesized directly from latent compressed features, and adversarial training [22] is used to make up realistic details. Indeed, information propagates in CCVS to new frames as previously synthesized ones become part of the context. Like other video synthesis architectures based on autoregressive models [46, 67], CCVS can be used in many tasks besides future video prediction. Any data which can be expressed in the form of a fixed-size sequence of elements from a finite set (*aka*, tokens) can be processed by a transformer. This applies to video frames (here, via compression and quantization) and to other types of data. Hence, one can easily fuse modalities without having to build complex or task-specific architectures. This idea has been used to control image synthesis [17, 47], and we extend it to a variety of video synthesis tasks by guiding the prediction with different annotations as illustrated in Figure 2. Our code will be made publicly available. We summarize our main contributions as follows:

1. an optical flow mechanism within an autoencoder to better reconstruct frames from context,
2. the use of ancillary information to control latent temporal dynamics when synthesizing videos,
3. a performance on par with or better than the state of the art, while being more memory-efficient.

2 Related Work

Video synthesis. In its simplest form, videos are produced without prior information about their content. GAN-based approaches map Gaussian noise into a visually plausible succession of frames. For example, VGAN [60] and ProgressiveVGAN [1], adapt the traditional GAN framework [22] from image to video synthesis by simply using 3D instead of 2D convolutions. These approaches, including recent attempts such as G^3AN [65], are computationally expensive, and, by nature, restricted to synthesizing a fixed number of frames due to the constraints of their architecture. Other approaches predict latent motion vectors with a CNN [49], or a recurrent neural network (RNN) [11, 50, 55], and generate frames with individual 2D operations. To avoid the shortcomings of information loss in the sequential processing of RNNs, we use an attention-based autoregressive model instead. Since we forecast temporal dynamics in a compressed space, a large temporal window can be used when predicting new frames, without having to resort to expensive 3D computations. Previous works have attempted to scale video synthesis to higher resolution by using progressive training [1], subsampling [49, 50], reducing the dimension of the discriminator [31], or redefining the task as finding a trajectory in the latent space of a pre-trained image generator [54]. Compression, together with efficient context-aware reconstruction allows us to synthesize videos at high resolution.

Controllable video synthesis. Some of the aforementioned works [11, 49, 50, 65] have proposed to condition the synthesis on class labels. Another popular control is to use a few priming frames to set off the generation process. Also known as *future video prediction*, this task has received a lot of attention recently. Methods based on variational autoencoders (VAE) [2, 13] have been proposed to account for the stochastic nature of future forecasting, *i.e.*, the plurality of possible continuations, disregarded in deterministic predictive models [18, 41, 59]. Yet, their blurry predictions have motivated the incorporation of adversarial training [38], hierarchical architectures [6, 68], fully latent temporal models [19], or normalizing flows [36]. Another line of work infers spatial transformation parameters (*e.g.*, optical flow), and predicts the future by warping past frames (*i.e.*, grid sampling and interpolation as in [28]) in RGB space [18, 20, 23, 61, 69] or in feature space [39], typically using a refinement step to handle occlusions. Lately, autoregressive methods [46, 67] leveraging a self-attention mechanism [58] have also been applied to this task. Our method benefits from the context-efficiency of approaches based on spatial transformation modules and the modeling power of autoregressive networks. In the meantime, other forms of control with interesting applications have emerged. Point-to-point generation [64], a variant of future video prediction, specifies both start and end frames. State or action-conditioned synthesis [23, 42] allows to guide the frame-by-frame evolution with high-level commands. Additional works consider video synthesis based on one [45] or multiple layouts [40, 63], another video [7], or sound [9, 29, 62]. To account for the variety of potential controls, we leverage the flexibility of transformers, and propose an unifying approach which can adapt to all of these tasks.

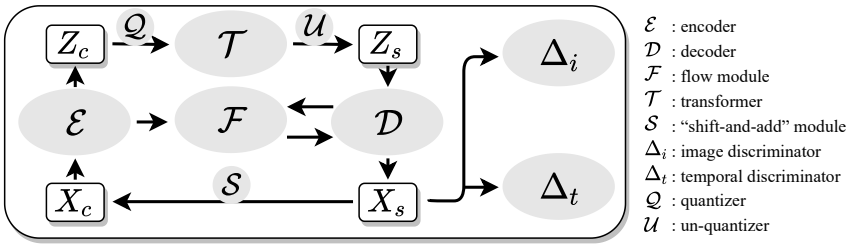


Figure 1: Proposed architecture. Here X_c and X_s respectively stand for the (input) context and (output) synthesized video. Learnable encoder and decoder modules $\mathcal{E}$ and $\mathcal{D}$ are linked by a learnable flow estimation module $\mathcal{F}$ ensuring spatio-temporal consistency between context and synthesized frames. The architecture is doubly autoregressive, with a transformer $\mathcal{T}$ responsible for predicting the codes Z_s associated with future frames X_s from the codes Z_c associated with context frames X_c, and a simple, parameterless, "shift-and-add" module $\mathcal{S}$ updating X_c as each new frame is generated. The arcitecture is trained in two steps: The parameters of $\mathcal{E}$, $\mathcal{D}$ and $\mathcal{F}$ are first estimated (without any future forecasting from the transformer) in an adversarial manner using two discriminators Δ_i and Δ_t to ensure that the frames synthesized are both realistic (Δ_i) and temporally coherent (Δ_t). The two discriminators are then discarded, and the parameters of the transformer $\mathcal{T}$ are estimated with $\mathcal{E}$, $\mathcal{F}$ and $\mathcal{D}$ frozen. At inference time, the transformer is used only once, latter frames and embeddings being estimated in an autoregressive manner by the quadruple $\mathcal{D}$ $\mathcal{S}$, $\mathcal{E}$, $\mathcal{F}$. See text for more details.

3 Context-aware controllable video synthesis

3.1 Overview of the proposed approach

Let us first set up our problem in an abstract setting: We consider sequential data (a video, a sentence, a speech time series), such that an individual datum x is natively an element of $\mathbb{R}^d$, which can be encoded (or embedded) as some feature (or code) z in $\mathbb{R}^f$ with, typically, $f \ll d$ (we will use encoding and embedding, feature and code interchangeably in the rest of this presentation). We assume that we are given a $d \times m$ *context* matrix X_c, whose columns are embeddings of m successive data frames, and our goal is to *synthesize* a plausible representation of the following n frames in the form of another $d \times n$ matrix X_s. Given X_c, we can compute the corresponding $f \times m$ embedding matrix Z_c using some encoder $\mathcal{E} : \mathbb{R}^d \to \mathbb{R}^f$, then use an autoregressive model (a transformer in our case) $\mathcal{T} : \mathbb{R}^{f \times m} \to \mathbb{R}^{f \times n}$ to predict the $f \times n$ matrix Z_s formed by the n codes corresponding to time ticks $m+1$ to $m+n$. (Note that the values of m and n can be arbitrary, using [by now] traditional masking and sliding window techniques.) These codes can finally be converted one by one into the corresponding frames of X_s using some decoder $\mathcal{D} : \mathbb{R}^f \to \mathbb{R}^d$.

The overall approach is illustrated by Figure 1. In practice, we have chosen in our implementation to insert a quantizer over a learned codebook between the encoder and the transformer. This complicates the architecture a bit, but has several advantages, including the reuse of familiar NLP technology [58] and, perhaps more importantly, affording simple mechanisms for handling different types of inputs (from images to sound for example [8, 26]) and the intrinsically uncertain and multimodal nature of future prediction (by sampling different codebook elements according to their likelihood [25]). Concretely, this choice simply amounts to inserting in our architecture, right after the encoder $\mathcal{E}$, a nearest-neighbor quantizer $\mathcal{Q} : \mathbb{R}^f \to [\![1, q]\!]^l$ parameterized by an $p \times q$ codebook which, given the embedding z of a frame, returns a sequence of corresponding *tokens*, that is, the indices of the closest entry in the codebook for l consecutive data chunks with $f = p \times l$. In our particular setting, these chunks correspond to low-resolution spatial features obtained from images via compression. We abuse notation, and identify $\mathcal{Q}$ with its parameterization by this codebook, so we can optimize over $\mathcal{Q}$ just as we optimize over $\mathcal{E}$ instead of naming explicitly its parameters in the rest of this presentation. An "un-quantizer" $\mathcal{U} : [\![1, q]\!]^l \to \mathbb{R}^f$, also parameterized (implicitly) by the codebook and associating with each token the corresponding entry of the codebook, is also inserted right before the decoder $\mathcal{D}$. In this setting, $\mathcal{T}$ now takes as input an array of $m \times l$ successive tokens, and outputs $n \times l$ tokens, each one of them chosen among q possibilities as either the one with the highest score, or drawn randomly from the top-k scores to account for the multimodal nature of future forecasting (here $k \leq q$ is some predefined constant, see [25] for related approaches).

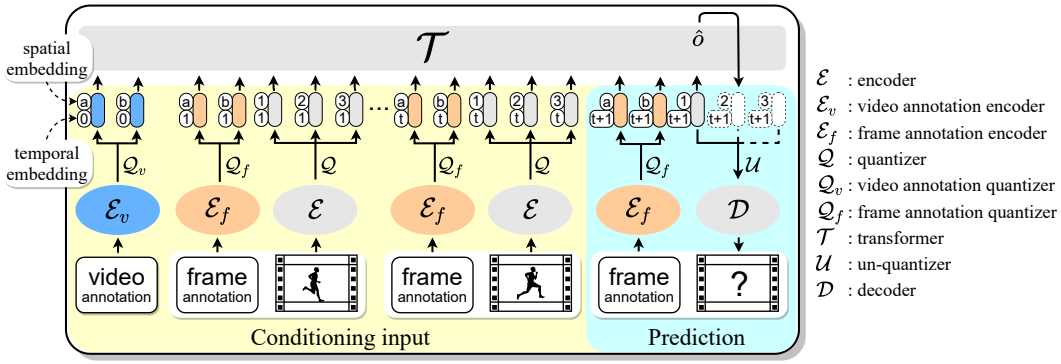


Figure 2: Illustration of the transformer to predict latent temporal dynamics with control-related data. For clarity, we omit the flow module $\mathcal{F}$ between $\mathcal{E}$ and $\mathcal{D}$. Input video-level annotation, frame-level annotations, and conditioning frames are encoded and quantized to form the initial token sequence, and mapped to corresponding embeddings. Subsequent frame tokens are obtained autoregressively with $\mathcal{T}$, while the ones corresponding to video- and frame-level annotations guide the prediction.

The elements of the architecture described so far are by now (individually) rather classical, with learnable, parametric functions $\mathcal{E}$, $\mathcal{D}$, $\mathcal{Q}$ and $\mathcal{T}$. Besides putting them all together, and as detailed in the rest of this section, we add several original elements: **(1)** The encoding/decoding scheme is improved by the use of two discriminators, Δ_i and Δ_t respectively, trained in an adversarial manner to ensure that the predicted frames are realistic and temporally consistent. **(2)** The context matrix X_c is itself updated each time a new frame is predicted in an autoregressive manner (iteratively fill the matrix up to some predefined capacity, then shift to the left, forgetting the first frame and adding the latest synthetic one on the right). **(3)** The encoder and decoder are linked through a learnable *flow module* $\mathcal{F}$ allowing the context frames to guide the prediction of the synthetic ones. **(4)** Additional *control variables*, in the form of sequence- or frame-level annotations, and tokens corresponding to their embeddings can be added to the codes passed on to the autoregressive model $\mathcal{T}$.

The data sequences we focus on in the following are video clips, and the control signals range from object trajectories to audio tracks. We detail in this section the concrete components of the approach sketched above, including the autoencoder and quantizer architectures and their training procedure (Section 3.2), and the implementation of the autoregressive model by a transformer [58], illustrated in Figure 2, which we adapt to account for outside control signals (Section 3.3), once again with the corresponding procedure.

3.2 First stage: training the context-aware autoencoder and the quantizer

Architecture. Following the notations introduced in Section 3.1, individual RGB frames x and their latent representations z are elements of $\mathbb{R}^d$ and $\mathbb{R}^f$ respectively with $d = H \times W \times 3$ and $f = h \times w \times F$, such that the length of the token sequence for each frame is $l = h \times w$. $\mathcal{E}$ and $\mathcal{D}$ increase and decrease the spatial resolution progressively by using convolutional residual blocks [24], with $(l_k)_{k \in [\![1;K]\!]}$ the K corresponding intermediate levels ($l_k = h_k \times w_k$). It is common practice to augment, as in U-Net [48], the autoencoder with long skip connections between $\mathcal{E}$ and $\mathcal{D}$ to share information across the two models at these intermediate levels despite the lossy compression bottleneck. Although we cannot apply this directly to our setting since information only flows through $\mathcal{D}$ for predicted timesteps, we argue that such skip connections can be established from the encoding stage of a context frame x_c to the decoding stage of a new frame x_s. Similar mechanisms [13, 16] have been proposed in the past but they are only copying static background features from a single context frame. Our approach addresses this issue by leveraging a flow module, which can be extended to multi-frame context without significant overhead, other than having to store intermediate features.

Concretely, let e_c be features being encoded from x_c, and d_s features being decoded from z_s at a given intermediate level l_k. $\mathcal{F}$ outputs an optical flow f_c (in $\mathbb{R}^{2 \times h_k \times w_k}$) which estimates the displacement field from e_c to d_s (as a proxy to the one from x_c to x_s), and a fusion mask m_c (in $\mathbb{R}^{1 \times h_k \times w_k}$) which indicates the expected similarity between aligned features $e'_c = \mathrm{W}(e_c, f_c)$ and d_s (also as a proxy for the one in image domain) with W corresponding to a standard warping operation. We define

temporal skip connections as the following in-place modification of d_s:

$$d_s = \sigma(m_c) \otimes d_s + (\mathbf{1} - \sigma(m_c)) \otimes e'_c, \tag{1}$$

where σ is the Sigmoid function, and $\otimes$ the element-wise product. For concrete implementation of $\mathcal{F}$, we build upon LiteFlowNet [27], an optical flow estimation model which also combines pyramidal extraction and progressive warping of features. For simple integration into our framework, we use features from $\mathcal{E}$ and $\mathcal{D}$ both in the mask and flow estimation, and in the update (1). This process readily generalizes to multi-frame contextual information (see supplementary material for details).

Training procedure. The global objective is the linear combination of four auxiliary ones:

$$\mathcal{L} = \lambda_q \mathcal{L}_q + \lambda_r \mathcal{L}_r + \lambda_a \mathcal{L}_a + \lambda_c \mathcal{L}_c, \tag{2}$$

namely a quantization loss ($\mathcal{L}_q$), a reconstruction loss ($\mathcal{L}_r$), an adversarial loss ($\mathcal{L}_a$), and a contextual loss ($\mathcal{L}_c$), detailed in the next paragraphs.

The codebook is trained by minimizing the reconstruction error between encoded features $z = \mathcal{E}(x)$ and quantized features $z_q = \mathcal{U}(\mathcal{Q}(z))$ (with notations introduced in Section 3.1):

$$\mathcal{L}_q(\mathcal{E}, \mathcal{Q}) = \| \operatorname{sg}(z) - z_q \|_2^2 + \beta \| \operatorname{sg}(z_q) - z \|_2^2, \tag{3}$$

where $\operatorname{sg}(.)$ is the stop gradient operation which constrains its operand to remain constant during backpropagation. The first part moves the codebook entries closer to the encoded features. The second part, known as the *commitment loss* [57], reverses the roles played by the two variables.

In regions with complex textures and high frequency details, local patterns shifted by a few pixels in x and its reconstruction $\widehat{x} = \mathcal{D}(z_q)$ may result in large pixel-to-pixel errors, while being visually satisfactory. We thus define the recovery objective as the L_1 loss in both RGB space and between features from a VGG network [52] pretrained on ImageNet [12]:

$$\mathcal{L}_r(\mathcal{E}, \mathcal{Q}, \mathcal{D}) = \|x - \widehat{x}\|_1 + \| \operatorname{VGG}(x) - \operatorname{VGG}(\widehat{x}) \|_1. \tag{4}$$

To tackle cases where information cannot be recovered from context due to occlusion, and the compressed features are insufficient to create plausible reconstructions due to lossy compression, we supplement our architecture with an image discriminator Δ_i, made of downsampling residual blocks [24], to encourage realistic outputs. Δ_i tries to distinguish real images from reconstructed ones ($\mathcal{L}_d$), while $\mathcal{E}$ and $\mathcal{D}$ fools Δ_i into assuming reconstructed images are as good as real ones ($\mathcal{L}_a$):

$$\mathcal{L}_d(\Delta_i) = \ln(1 + e^{\Delta_i(x)}) + \ln(1 + e^{-\Delta_i(\widehat{x})}), \tag{5}$$

$$\mathcal{L}_a(\mathcal{E}, \mathcal{Q}, \mathcal{D}) = \ln(1 + e^{\Delta_i(\widehat{x})}). \tag{6}$$

We employ a similar strategy on sequences of consecutive frames to improve the temporal consistency using a 3D temporal discriminator Δ_t, a direct extension of 2D image discriminator Δ_i.

The success of our method relies on accurate motion estimation in $\mathcal{F}$, a difficult task which benefits from self-supervision [30]. Therefore, we train the autoencoder with augmented views of the input frames as context. Custom augmentations functions $\mathrm{A} : \mathbb{R}^d \to \mathbb{R}^d$ include: rotation, scaling, translation, elastic deformation, and combinations of these. Augmented views are obtained by warping x by the suitable flow a_c: $\mathrm{A}(x) = \mathrm{W}(x, a_c)$, and the inverted flow f_c from $\mathrm{A}(x)$ to x can be approximated.[1] We resort to flow inversion because directly reconstructing distorted views may encourage similar defects during inference. Moreover, we balance between self-recovery and context-recovery objectives by additionally applying a blurring function $\mathrm{B} : \mathbb{R}^d \to \mathbb{R}^d$ and an occlusion mask o_c to the augmented frames $x_c = o_c \otimes \mathrm{B}(\mathrm{A}(x))$. We define the contextual loss as:

$$\mathcal{L}_c(\mathcal{E}, \mathcal{Q}, \mathcal{F}, \mathcal{D}) = \|f_c - \widehat{f}_c\|_2^2 + \|o'_c - \sigma(\widehat{m}_c)\|_2^2, \tag{7}$$

where $o'_c = \mathrm{W}(o_c, f_c)$, and $\widehat{f}_c$ and $\widehat{m}_c$ are the flow and mask estimated by $\mathcal{F}$. In practice, this loss is applied at intermediate resolution levels for improved training.

[1] Our implementation of flow inversion approximation is detailed in the supplementary material.

3.3 Second stage: predicting temporal dynamics with transformers

Architecture. We follow [17] and adopt an architecture similar to Image-GPT [8] for $\mathcal{T}$. Instead of modeling a single annotated frame as in [47], we design our model to account for sequences of N such frames to allow prediction of temporal dynamics controlled by ancillary information. We have shown, in Section 3.2, how a video frame can be represented by a sequence of l tokens (indices in $[\![1, q]\!]$) through encoding and quantization. Similar strategies can be applied to cater to other types of data, with or without compression depending on their complexity, and, thereby, turn ancillary information into tokens as well. The capacity of the transformer (the maximum sequence length it can process) is $L = l_v + N * (l_f + l)$, where l_v and l_f are the size of video- and frame-level annotations respectively. The final layer of the model predicts, for every i in $[\![1, L-1]\!]$, a vector $\widehat{o}_i$ (of size q, the number of possible tokens) which scores the likelihood of the $i+1^{\text{th}}$ token given all preceding ones.

Training procedure. To learn the parameters of $\mathcal{T}$, we load a complete sequence (N frames corresponding to L tokens), and try to predict the $L-1$ last tokens based on the $L-1$ first ones. This is done by maximizing the log-likelihood of the data using the cross-entropy loss:

$$\mathcal{L}(\mathcal{T}) = -\sum_{i=2}^{l} \log \left(\frac{\exp(\langle \widehat{o}_{i-1}, e_{\tau(i)} \rangle)}{\sum_j \exp(\langle \widehat{o}_{i-1}, e_j \rangle)} \right), \tag{8}$$

where $\tau(i)$ is the ith ground-truth token, and e_j a vector of 0's with a 1 in its jth coordinate.

Inference. During inference we use $\mathcal{T}$ to complete autoregressively an input sequence of tokens. To avoid known pitfalls (*e.g.*, repetitive synthesis) and allow diverse outcomes, we use top-k sampling whereby the next token is randomly chosen among the k most likely ones, weighted by their scores $\widehat{o}$. Although $\mathcal{T}$ processes sequences of N annotated frames, we predict ones of arbitrary length by using a temporal sliding window. Tokens corresponding to video- and frame-level annotations are given as input to $\mathcal{T}$ (not predicted, even for future frames) to guide the synthesis. We show in the experiments how this control can translates into a variety of interesting tasks.

4 Experiments

We assess the merits of CCVS in the light of extensive experiments on various video synthesis tasks.

Datasets. BAIR Robot Pushing [16] consists of 43k training and 256 test videos of a robotic arm interacting with objects from a fixed viewpoint. A high-resolution version has recently been released. We manually annotate, in 500 frames, the (x, y) location of the arm in image space to train a position estimator, which we use for state-conditioned synthesis. To account for real world scenarios, we evaluate on Kinetics-600 [4], a large and diverse action-recognition dataset with approximately 500K videos. We also test our method on AudioSet-Drums [21] for sound-conditioned synthesis on music performance, containing 6k and 1k video clips in train and test splits respectively. Other datasets and tasks are covered in the supplementary material.

Metrics. We use the Fréchet video distance (FVD) [56] which measures the distribution gap between real and synthetic videos in the feature space of an Inception3D network [5] pretrained on Kientics-400 [34]. It estimates the visual quality, diversity and temporal consistency of samples. For near-deterministic motions (*e.g.*, in reconstructions, or constrained tasks), there is a one-to-one mapping between real video frames and synthetic ones, and we include pairwise image quality assessments: the structural similarity index measure (SSIM) [66] which evaluates a per-frame conformity (combination of luminance, contrast and structure), and the peak signal-to-noise ratio (PSNR) which is directly related to the root mean squared error. For each metric, we compute the mean and standard deviation (std) over 5 evaluation runs (2 for Kinetics due to its voluminous 50k video test set). For clarity, the std value is shown only when it is greater than the reported precision.

Training details. All our models are trained on 4 Nvidia V100 GPUs (32GB VRAM each), using ADAM [35] optimizer, for multiple 20 hour runs. We adapt the batch size to fill the memory available. We use a learning rate of 0.02 to train the autoencoder, and exponential moving average [70] to obtain its final parameters. We use weighting factors $(1, 10, 1, 1)$ and 0.25 for $(\lambda_q, \lambda_r, \lambda_a, \lambda_c)$ and β in Equations (2) and (3) respectively. We use a learning rate of 10^{-5} to train the transformer.

Table 1: Ablation study of the autoencoder on BAIR (256×256). We evaluate self- and context-recovery modules in different scenarios: synthesizing 16-frame videos from known compressed features ("Reconstruction"), by inferring compressed features with $\mathcal{T}$ given the real trajectory of the robotic arm ("State-conditioned"), or without the trajectory ("Pred." and "Unc."). The first real frame is used as initial context in all cases, except for "Unc." where it is synthesized by StyleGAN2 [32].

Self-recovery				Ctxt.-recovery			Reconstruction		State-conditioned		Pred.	Unc.
RGB	VGG	Δ_i	Δ_t	$\mathcal{F}$	Sup.	Ctxt.	FVD↓	PSNR↑	FVD↓	PSNR↑	FVD↓	FVD↓
✓						0	$1200_{\pm 6}$	20.0	$1238_{\pm 24}$	17.8	$1265_{\pm 22}$	$1321_{\pm 10}$
✓	✓					0	$700_{\pm 11}$	17.9	$714_{\pm 5}$	17.4	$704_{\pm 7}$	$765_{\pm 12}$
✓	✓	✓				0	$323_{\pm 3}$	17.8	$355_{\pm 5}$	16.6	$377_{\pm 11}$	$566_{\pm 22}$
✓	✓	✓	✓			0	$389_{\pm 12}$	18.2	$401_{\pm 4}$	16.9	$407_{\pm 10}$	$627_{\pm 9}$
✓	✓	✓	✓	✓		1	$98_{\pm 2}$	22.7	$106_{\pm 2}$	22.2	$142_{\pm 6}$	$350_{\pm 11}$
✓	✓	✓	✓	✓	✓	1	$87_{\pm 3}$	24.4	$97_{\pm 4}$	22.1	$128_{\pm 4}$	$301_{\pm 10}$
✓	✓	✓	✓	✓	✓	8	$62_{\pm 1}$	25.4	$76_{\pm 3}$	22.7	$109_{\pm 6}$	$299_{\pm 4}$
✓	✓	✓	✓	✓	✓	15	$60_{\pm 1}$	25.6	$75_{\pm 2}$	**22.8**	$110_{\pm 3}$	$297_{\pm 7}$
Training longer (num. epochs $\times 3$)							$\mathbf{45}_{\pm 1}$	**26.8**	$\mathbf{67}_{\pm 1}$	22.3	$\mathbf{100}_{\pm 2}$	$\mathbf{293}_{\pm 7}$

"Sup.": self-supervision of $\mathcal{F}$; "Ctxt.": number of context frames taken into account (in $\mathcal{F}$) when decoding current frame (in $\mathcal{D}$).

Table 2: Ablation study of the transformer on BAIR. We adopt notations and evaluation setups from Table 1.

Architecture			Top-k		State-conditioned		Pred.	Unc.
Layer	Head	Dec.	Frame	State	FVD↓	PSNR↑	FVD↓	FVD↓
6	4		1	1	$73_{\pm 2}$	23.0	$281_{\pm 7}$	$474_{\pm 17}$
12	8		1	1	$73_{\pm 3}$	23.1	$262_{\pm 7}$	$435_{\pm 8}$
24	16		1	1	$70_{\pm 3}$	**23.3**	$331_{\pm 9}$	$521_{\pm 19}$
24	16	✓	1	1	$69_{\pm 2}$	23.2	$321_{\pm 9}$	$479_{\pm 34}$
24	16	✓	10	1	$\mathbf{65}_{\pm 2}$	22.4	$127_{\pm 7}$	$308_{\pm 19}$
24	16	✓	100	1	$67_{\pm 1}$	22.3	$121_{\pm 2}$	$314_{\pm 12}$
24	16	✓	100	10	$67_{\pm 1}$	22.3	$\mathbf{100}_{\pm 2}$	$\mathbf{293}_{\pm 7}$

"Dec.": spatio-temporal decomposition of positional embeddings.

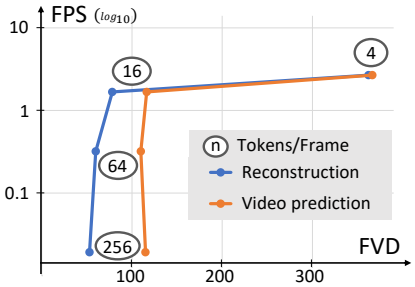


Figure 3: Quality and speed of synthesis vs. compression on a Nvidia V100 GPU.

4.1 Ablation study

We conduct an ablation study of CCVS to show the individual contribution of the key components of the proposed autoencoder (Table 1), transformer (Table 2), and the effect of compression (Figure 3).

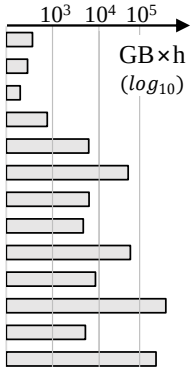
First, we fix the transformer and observe the incremental improvements in synthesis quality when adding self- and context-recovery modules to the autoencoder (Table 1). In particular, L_1 loss in the feature space of a VGG net [52] produces sharper videos than using the same loss in the RGB space alone. Predicted frames using image and temporal discriminators (Δ_i and Δ_t) display greater realism and temporal consistency. The flow module $\mathcal{F}$ significantly improves the performance on all metrics by allowing context frames to guide the reconstruction of synthetic ones. The self-supervision of $\mathcal{F}$, the use of larger context windows, and longer training times, further improve the quality of the synthesis. Δ_t seems to deteriorate the FVD at first, but when all modules are combined it improves FVD by almost a factor 2 as it encourages better temporal consistency in the presence of context.

We fix the autoencoder, and compare different architectures and sampling strategies for the transformer (Table 2). Increasing the model capacity by adding layers and increasing expressivity (number of attention heads) along with a spatio-temporal decomposition of positional embeddings (shown in Figure 2, detailed in supplementary material) yields small improvements. Top-k sampling is beneficial for stochastic tasks (video prediction and unconditional synthesis), whereas always selecting the most probable tokens results in little to no motion. Guiding temporal dynamics with state-conditioned synthesis reduces the advantage of sampling by narrowing down possible outcomes.

Finally, we explore the effect of compression (in terms of the number of tokens for each frame) on synthesis speed (FPS) and quality (FVD) of reconstructed and predicted videos (Figure 3). We use a compression of 64 tokens in our default setup since it gives the best FVD while retaining a reasonable speed. The critical drop of FPS for low compression ratios is due to the pairwise consideration of all input tokens in $\mathcal{T}$. Note that using a smaller temporal window in $\mathcal{T}$ may allow additional speed-ups.

Table 3: Future video prediction on BAIR (64×64), synthesizing 16-frame videos given a few conditioning frames ("Cond."). We include some extensions of our method at higher resolution.

Method	Cond.	FVD ↓	Code Avail.	Memory, compute
MoCoGAN [55]	4	503	✓	16GB, 23h*
SVG-FP [13]	2	315	✓	12GB, 6 to 24h*
CDNA [18]	2	297	✓	10GB, 20h*
SV2P [2]	2	263	✓	16GB, 24 to 48h*
SRVP [19]	2	181	✓	36GB, 168h*
VideoFlow [36]	3	131		128GB, 336h*
LVT [46]	1	$126_{\pm 3}$	✓	128GB, 48h
SAVP [38]	2	116	✓	32GB, 144h
DVD-GAN-FP [11]	1	110		2TB, 24h*
Video Transformer (S) [67]	1	$106_{\pm 3}$		256GB, 33h*
TriVD-GAN-FP [39]	1	103		1TB, 280h*
Low res. CCVS (*ours*)	1	$99_{\pm 2}$	✓	128GB, 40h
Video Transformer (L) [67]	1	$94_{\pm 2}$		512GB, 336h*
			SSIM ↑ ($t = 8$)	SSIM ↑ ($t = 15$)
*High res.*** CCVS (*ours*)	1	$80_{\pm 3}$	0.729	0.683
+ end frame	2	$81_{\pm 2}$	0.766	0.839
+ state	1	$50_{\pm 1}$	0.885	0.863



*: value confirmed by authors; **: training / inference / SSIM at 256×256, FVD at 64×64.

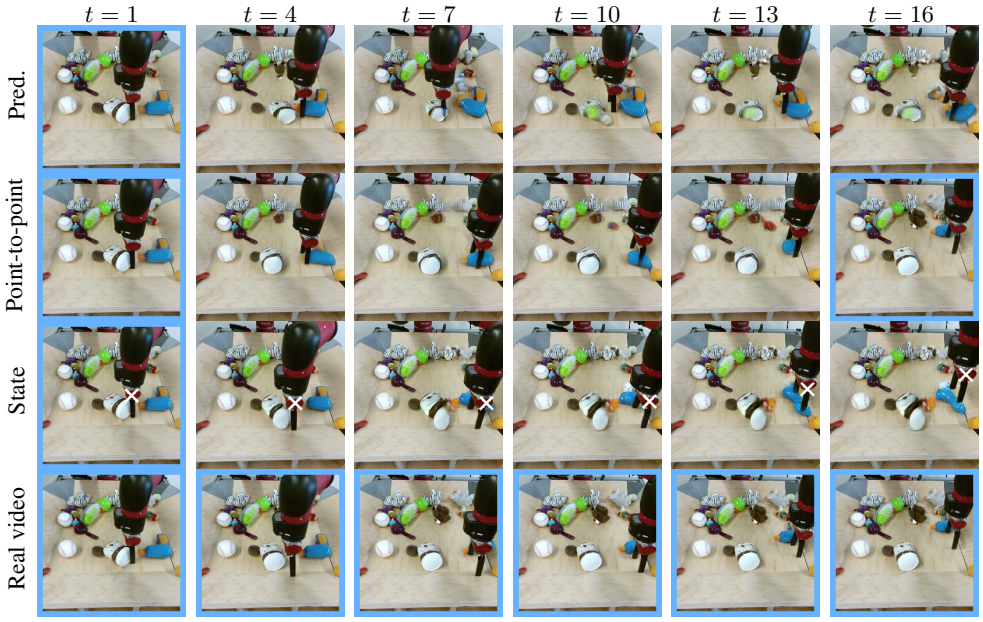


Figure 4: Qualitative samples for different types of control on BAIR (256×256). Zoom in for details.

4.2 Quantitative and qualitative studies

BAIR. For future video prediction on BAIR (Table 3), CCVS trained at 64×64 resolution (*low res.*) is on par with the best method (L-size version of [67]), but requires much less computing resources, and outperforms [67] under similar resources. We also propose *high res.* CCVS which is not strictly comparable to the prior arts because we use 256×256 image resolution during training and test, and resize the synthesized frames to 64×64 for computing FVD. However, using this variant demonstrates the performance gains that can arise by scaling CCVS. We additionally address point-to-point synthesis (by using the end frame as a video-level annotation) and state-conditioned synthesis (by using the estimated 2D position of the arm as a frame-level annotation). Point-to-point synthesis is more difficult than video prediction: Not only does it requires realistic video continuations, but also ones which explain the end position of all existing objects. Hence, FVD score is constant despite the additional input. Still, this yields better SSIM for mid-point and one-before-last frames.

Table 4: Future video prediction on Kinetics (64×64) of 16-frame videos from 5 consecutive input frames.

Method	FVD ↓	GPU/TPU Mem.
LVT [46]	225	128GB, 48h
Video Transformer [67]	$170_{\pm 5}$	2TB, 336h*
DVD-GAN-FP [11]	$69_{\pm 1}$	2TB, 144h*
CCVS (*ours*)	$55_{\pm 1}$	128GB, 300h
TriVD-GAN-FP [39]	$26_{\pm 1}$	16TB, 160h*

*: value confirmed by authors.

Table 5: Codebook size on Kinetics.

Size	Reconstruction			Pred.
	FVD ↓	SSIM ↑	PSNR ↑	FVD ↓
1024	$58_{\pm 1}$	0.907	31.2	$66_{\pm 2}$
4096	$54_{\pm 1}$	0.917	31.6	$64_{\pm 1}$
16384	$49_{\pm 1}$	0.923	32.0	$55_{\pm 1}$
65536	$45_{\pm 1}$	0.928	32.2	$61_{\pm 1}$
∞	$12_{\pm 1}$	0.963	34.5	229

Table 6: Sound-conditional video synthesis on AudioSet-Drums at resolution 64×64.

Method	Cond.	Audio	SSIM-16	SSIM-30	SSIM-45	PSNR-16	PSNR-30	PSNR-45
SVG-LP [13]	15		$0.971_{\pm 0.017}$	$0.661_{\pm 0.010}$	$0.510_{\pm 0.008}$	$30.0_{\pm 1.1}$	$16.6_{\pm 0.3}$	$13.5_{\pm 0.1}$
Vougioukas *et al.* [62]	15	✓	$0.940_{\pm 0.017}$	$0.904_{\pm 0.007}$	$0.896_{\pm 0.015}$	$26.2_{\pm 1.0}$	$23.8_{\pm 0.2}$	$23.3_{\pm 0.3}$
Sound2Sight [9]	15	✓	$0.984_{\pm 0.009}$	$0.954_{\pm 0.007}$	$\mathbf{0.947}_{\pm 0.007}$	$33.2_{\pm 0.1}$	$27.9_{\pm 0.5}$	$27.0_{\pm 0.3}$
CCVS (*ours*)	15	✓	$\mathbf{0.987}_{\pm 0.001}$	$\mathbf{0.956}_{\pm 0.006}$	$0.945_{\pm 0.008}$	$\mathbf{33.7}_{\pm 0.4}$	$\mathbf{28.4}_{\pm 0.6}$	$\mathbf{27.3}_{\pm 0.5}$

State-conditioned synthesis is much better on FVD (synthetic videos have greater correlation with real ones) and mid-synthesis SSIM (near-deterministic motion). Some sample frames synthesized for these tasks are shown in Figure 4. CCVS creates plausible high-quality videos in various settings, and true interactions with objects compared to previous attempts [42] at the same resolution.

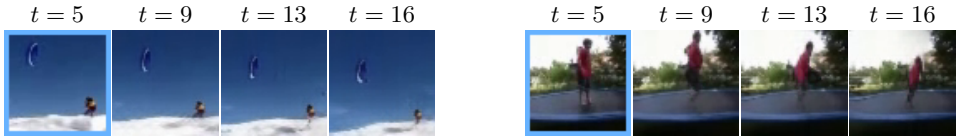


Figure 5: Qualitative samples on future video prediction on Kinetics (64×64). Zoom in for details.

Kinetics. CCVS ranks second on Kinetics video prediction benchmark. Kinetics contains more diversity than BAIR, and the reconstruction is thus more difficult. A solution [47] is to increase the codebook size (Table 5) but it stops translating into better prediction FVD at some point. We also try removing the quantization step (equivalent to an infinite codebook), and directly regressing latent features with $\mathcal{T}$ (instead of ranking the likelihood of possible tokens). It allows better reconstructions, yet the prediction FVD is high. We show some examples of synthetic video predictions in Figure 5.

AudioSet-Drums. CCVS achieves top performance on sound-conditioned video synthesis on AudioSet Drums (Table 6). For more details see supplementary material.

5 Discussion

CCVS is on par or better than the state-of-the-art on standard benchmarks, uses less computational resources, and scales to high resolution. Training neural networks is environmentally costly, due to the carbon footprint to power processing hardware [15, 53]. Methods sparing GPU-hours like ours are crucial to make AI less polluting [37, 53], and move from a "Red" to a "Green" AI [51]. Future work will include exploring new codebook strategies and synthesis guided by textual information.

Broader impact. The increased accessibility and the many controls CCVS offers could accelerate the emergence of questionable applications, such as "deepfakes" (*e.g.*, a video created from someone's picture and an arbitrary audio) which could lead to harassment, defamation, or dissemination of fake news. On top of current efforts to automate their detection [43], it remains our responsibility to grow awareness of these possible misuses. Despite these worrying aspects, our contribution has plenty of positive applications which outweigh the potential ethical harms. Our efficient compression scheme is a step in the direction of real-time solutions: *e.g.*, enhancing human-robot interactions, or improving the safety of self-driving cars by predicting the trajectories of people and vehicles nearby.

References

[1] 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*, 2018.

[2] Mohammad Babaeizadeh, Chelsea Finn, Dumitru Erhan, Roy H. Campbell, and Sergey Levine. Stochastic variational video prediction. In *ICLR*, 2018.

[3] Tom B Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *arXiv preprint*, 2020.

[4] Joao Carreira, Eric Noland, Andras Banki-Horvath, Chloe Hillier, and Andrew Zisserman. A short note about Kinetics-600. *arXiv preprint*, 2018.

[5] Joao Carreira and Andrew Zisserman. Quo vadis, action recognition? a new model and the kinetics dataset. In *CVPR*, 2017.

[6] Lluis Castrejon, Nicolas Ballas, and Aaron Courville. Improved conditional VRNNs for video prediction. In *ICCV*, 2019.

[7] Caroline Chan, Shiry Ginosar, Tinghui Zhou, and Alexei A. Efros. Everybody dance now. *ICCV*, 2019.

[8] Mark Chen, Alec Radford, Rewon Child, Jeffrey Wu, Heewoo Jun, David Luan, and Ilya Sutskever. Generative pretraining from pixels. In *ICML*, 2020.

[9] Anoop Cherian, Moitreya Chatterjee, and Narendra Ahuja. Sound2Sight: Generating visual dynamics from sound and context. In *ECCV*, 2020.

[10] Krzysztof Choromanski, Valerii Likhosherstov, David Dohan, Xingyou Song, Andreea Gane, Tamas Sarlos, Peter Hawkins, Jared Davis, Afroz Mohiuddin, Lukasz Kaiser, David Belanger, Lucy Colwell, and Adrian Weller. Rethinking attention with performers. In *ICLR*, 2021.

[11] Aidan Clark, Jeff Donahue, and Karen Simonyan. Adversarial video generation on complex datasets. *arXiv preprint*, 2019.

[12] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. ImageNet: A large-scale hierarchical image database. In *CVPR*, 2009.

[13] Emily Denton and Rob Fergus. Stochastic video generation with a learned prior. In *ICML*, 2018.

[14] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In *ACL*, 2019.

[15] Payal Dhar. The carbon impact of artificial intelligence. *Nature Machine Intelligence*, 2020.

[16] Frederik Ebert, Chelsea Finn, Alex X. Lee, and Sergey Levine. Self-supervised visual planning with temporal skip connections. In *CoRL*, 2017.

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

[18] Chelsea Finn, Ian Goodfellow, and Sergey Levine. Unsupervised learning for physical interaction through video prediction. In *NeurIPS*, 2016.

[19] Jean-Yves Franceschi, Edouard Delasalles, Mickaël Chen, Sylvain Lamprier, and Patrick Gallinari. Stochastic latent residual video prediction. In *ICML*, 2020.

[20] Hang Gao, Huazhe Xu, Qi-Zhi Cai, Ruth Wang, Fisher Yu, and Trevor Darrell. Disentangling propagation and generation for video prediction. In *ICCV*, 2019.

[21] Jort F Gemmeke, Daniel PW Ellis, Dylan Freedman, Aren Jansen, Wade Lawrence, R Channing Moore, Manoj Plakal, and Marvin Ritter. Audio set: An ontology and human-labeled dataset for audio events. In *ICASSP*, 2017.

[22] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *NeurIPS*, 2014.

[23] Zekun Hao, Xun Huang, and Serge Belongie. Controllable video generation with sparse trajectories. In *CVPR*, 2018.

[24] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, 2016.

[25] Ari Holtzman, Jan Buys, Li Du, Maxwell Forbes, and Yejin Choi. The curious case of neural text degeneration. In *ICLR*, 2020.

[26] Cheng-Zhi Anna Huang, Ashish Vaswani, Jakob Uszkoreit, Ian Simon, Curtis Hawthorne, Noam Shazeer, Andrew M. Dai, Matthew D. Hoffman, Monica Dinculescu, and Douglas Eck. Music transformer. In *ICLR*, 2019.

[27] Tak-Wai Hui, Xiaoou Tang, and Chen Change Loy. LiteFlowNet: A lightweight convolutional neural network for optical flow estimation. In *CVPR*, 2018.

[28] Max Jaderberg, Karen Simonyan, Andrew Zisserman, and Koray Kavukcuoglu. Spatial transformer networks. In *NeurIPS*, 2015.

[29] Amir Jamaludin, Joon Son Chung, and Andrew Zisserman. You said that?: Synthesising talking faces from audio. *IJCV*, 2019.

[30] Rico Jonschkowski, Austin Stone, Jonathan T. Barron, Ariel Gordon, Kurt Konolige, and Anelia Angelova. What matters in unsupervised optical flow. In *ECCV*, 2020.

[31] Emmanuel Kahembwe and Subramanian Ramamoorthy. Lower dimensional kernels for video discriminators. *Neural Networks*, 2020.

[32] Tero Karras, Samuli Laine, Miika Aittala, Janne Hellsten, Jaakko Lehtinen, and Timo Aila. Analyzing and improving the image quality of stylegan. In *CVPR*, 2020.

[33] Angelos Katharopoulos, Apoorv Vyas, Nikolaos Pappas, and François Fleuret. Transformers are RNNs: Fast autoregressive transformers with linear attention. In *ICML*, 2020.

[34] Will Kay, Joao Carreira, Karen Simonyan, Brian Zhang, Chloe Hillier, Sudheendra Vijayanarasimhan, Fabio Viola, Tim Green, Trevor Back, Paul Natsev, Mustafa Suleyman, and Andrew Zisserman. The kinetics human action video dataset. *arXiv preprint*, 2017.

[35] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *ICLR*, 2015.

[36] Manoj Kumar, Mohammad Babaeizadeh, Dumitru Erhan, Chelsea Finn, Sergey Levine, Laurent Dinh, and Durk Kingma. VideoFlow: A conditional flow-based model for stochastic video generation. In *ICLR*, 2020.

[37] Alexandre Lacoste, Alexandra Luccioni, Victor Schmidt, and Thomas Dandres. Quantifying the carbon emissions of machine learning. In *NeurIPS workshop*, 2019.

[38] Alex X. Lee, Richard Zhang, Frederik Ebert, Pieter Abbeel, Chelsea Finn, and Sergey Levine. Stochastic adversarial video prediction. *arXiv preprint*, 2018.

[39] Pauline Luc, Aidan Clark, Sander Dieleman, Diego de Las Casas, Yotam Doron, Albin Cassirer, and Karen Simonyan. Transformation-based adversarial video prediction on large-scale data. *arXiv preprint*, 2020.

[40] Arun Mallya, Ting-Chun Wang, Karan Sapra, and Ming-Yu Liu. World-consistent video-to-video synthesis. In *ECCV*, 2020.

[41] Michael Mathieu, Camille Couprie, and Yann LeCun. Deep multi-scale video prediction beyond mean square error. In *ICLR*, 2016.

[42] Willi Menapace, Stéphane Lathuilière, Sergey Tulyakov, Aliaksandr Siarohin, and Elisa Ricci. Playable video generation. In *CVPR*, 2021.

[43] Thanh Thi Nguyen, Quoc Viet Hung Nguyen, Cuong M. Nguyen, Dung Nguyen, Duc Thanh Nguyen, and Saeid Nahavandi. Deep learning for deepfakes creation and detection: A survey. *arXiv preprint*, 2019.

[44] Anselm Levskaya Nikita Kitaev, Lukasz Kaiser. Reformer: The efficient transformer. In *ICLR*, 2020.

[45] Junting Pan, Chengyu Wang, Xu Jia, Jing Shao, Lu Sheng, Junjie Yan, and Xiaogang Wang. Video generation from single semantic label map. In *CVPR*, 2019.

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

[47] Aditya Ramesh, Mikhail Pavlov, Gabriel Goh, Scott Gray, Chelsea Voss, Alec Radford, Mark Chen, and Ilya Sutskever. Zero-shot text-to-image generation. *arXiv preprint*, 2021.

[48] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *MICCAI*, 2015.

[49] Masaki Saito, Eiichi Matsumoto, and Shunta Saito. Temporal generative adversarial nets with singular value clipping. In *ICCV*, 2017.

[50] Masaki Saito, Shunta Saito, Masanori Koyama, and Sosuke Kobayashi. Train sparsely, generate densely: Memory-efficient unsupervised training of high-resolution temporal GAN. *IJCV*, 2020.

[51] Roy Schwartz, Jesse Dodge, Noah A. Smith, and Oren Etzioni. Green AI. *ACM*, 2020.

[52] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. In *ICLR*, 2015.

[53] Emma Strubell, Ananya Ganesh, and Andrew McCallum. Energy and policy considerations for deep learning in NLP. In *ACL*, 2019.

[54] 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 *ICLR*, 2021.

[55] Sergey Tulyakov, Ming-Yu Liu, Xiaodong Yang, and Jan Kautz. MoCoGAN: Decomposing motion and content for video generation. In *CVPR*, 2018.

[56] 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*, 2018.

[57] Aaron van den Oord, Oriol Vinyals, and Koray Kavukcuoglu. Neural discrete representation learning. In *NeurIPS*, 2017.

[58] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *NeurIPS*, 2017.

[59] Carl Vondrick, Hamed Pirsiavash, and Antonio Torralba. Anticipating visual representations from unlabeled video. In *CVPR*, 2016.

[60] Carl Vondrick, Hamed Pirsiavash, and Antonio Torralba. Generating videos with scene dynamics. In *NeurIPS*, 2016.

[61] Carl Vondrick and Antonio Torralba. Generating the future with adversarial transformers. In *CVPR*, 2017.

[62] Konstantinos Vougioukas, Stavros Petridis, and Maja Pantic. End-to-end speech-driven facial animation with temporal GANs. In *BMVC*, 2018.

[63] Ting-Chun Wang, Ming-Yu Liu, Jun-Yan Zhu, Guilin Liu, Andrew Tao, Jan Kautz, and Bryan Catanzaro. Video-to-video synthesis. In *NeurIPS*, 2018.

[64] Tsun-Hsuan Wang, Yen-Chi Cheng, Chieh Hubert Lin, Hwann-Tzong Chen, and Min Sun. Point-to-point video generation. In *ICCV*, 2019.

[65] Yaohui Wang, Piotr Bilinski, Francois Bremond, and Antitza Dantcheva. G3AN: Disentangling appearance and motion for video generation. In *CVPR*, 2020.

[66] Zhou Wang, Alan C. Bovik, Hamid R. Sheikh, and Eero P. Simoncelli. Image quality assessment: from error visibility to structural similarity. *TIP*, 2004.

[67] Dirk Weissenborn, Oscar Täckström, and Jakob Uszkoreit. Scaling autoregressive video models. In *ICLR*, 2020.

[68] Bohan Wu, Suraj Nair, Roberto Martin-Martin, Li Fei-Fei, and Chelsea Finn. Greedy hierarchical variational autoencoders for large-scale video prediction. In *CVPR*, 2021.

[69] Yue Wu, Rongrong Gao, Jaesik Park, and Qifeng Chen. Future video synthesis with object motion prediction. In *CVPR*, 2020.

[70] Yasin Yaz, Chuan-Sheng Foo, Stefan Winkler, Kim-Hui Yap, Georgios Piliouras, and Vijay Chandrasekhar. The unusual effectiveness of averaging in GAN training. In *ICLR*, 2019.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes] See the corresponding Abstract and Section 1. Our claims are clearly stated at the end of the introduction. Technical content corresponding to Claim-1 is detailed in Section 3.2 and there is Table 1 to back it up. Technical content corresponding to Claim-2 is detailed in Section 3.3 and there is Table 2 as well as Figure 4 to back it up. Claim-3 is backed up by Table 3, Table 4 and Table 6.
   (b) Did you describe the limitations of your work? [Yes] See Section 4.1 and Figure 3 where we evidence a trade-off between quality and speed of synthesis, and Section 4.2 and Table 5 where we show that, confronted to high diversity datasets like Kinetics, the reconstruction error of our method with standard codebook strategies limits its performance.
   (c) Did you discuss any potential negative societal impacts of your work? [Yes] See Section 5 where we discuss potential misuses.
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes] We have read them carefully and taken them into consideration.
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [N/A]
   (b) Did you include complete proofs of all theoretical results? [N/A]
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes] The code is provided in the supplemental material with detailed instruction, and will be made available on the public release of the article as stated in the Section 1.
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] Datasets (with their respective splits) and training details are given in Section 4.
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes] As mention in the "Metrics" paragraph in Section 4 all methods were evaluated multiple times and we report corresponding error bars.
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes] See Table 3 and Table 4. Note that we asked authors of related methods for their own training setup and conducted a short analysis of memory requirements to train video synthesis models.
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes] See Section 4.
   (b) Did you mention the license of the assets? [Yes] In the code folder.
   (c) Did you include any new assets either in the supplemental material or as a URL? [N/A]
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A]
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A]
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]