CCVS: Context-aware Controllable Video Synthesis

Guillaume Le Moing¹, * Jean Ponce¹, ² Cordelia Schmid¹
¹Inria and Department of Computer Science, ENS, CNRS, PSL Research University
²Center for Data Science, New York University

Abstract

This presentation introduces a self-supervised learning approach to the synthesis of new video 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 [38], 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 [68], for their simplicity, and their ability to model long-range dependencies and learn from large volumes of data [4, 16]. First introduced for natural language processing (NLP) and then successfully applied to visual data [18], 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 [11, 39, 50], using such methods directly on visual data is still limited to low resolutions and impractical without considerable computational power [9, 79].

Some recent works [20, 53] 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 [26] mitigate this issue by “hallucinating” plausible local details in image synthesis [20]. But latent video

*corresponding author: guillaume.le-moing@inria.fr

transformers [53] 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.

We follow the problem decomposition from [20, 53], 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 [26] 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 [53, 79],
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 [20, 54], and we extend it to a
variety of video synthesis tasks by guiding the prediction with different annotations as illustrated in
Figure 2. Code, pretrained models, and video samples synthesized by our approach are available at
the url https://16lemoing.github.io/ccvs. Our main contributions are 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 [71] and ProgressiveVGAN [1], adapt the traditional GAN framework [26] from
image to video synthesis by simply using 3D instead of 2D convolutions. These approaches, including
recent attempts such as G^3AN [76], 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 [56], or a recurrent neural network (RNN) [12, 57, 65], 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], sub-
sampling [56, 57], reducing the dimension of the discriminator [36], or redefining the task as finding
a trajectory in the latent space of a pre-trained image generator [63]. Compression, together with
efficient context-aware reconstruction allows us to synthesize videos at high resolution.

**Controllable video synthesis.** Some of the aforementioned works [12, 56, 57, 76] handle synthesis
conditioned on a class label. Another popular control is to use a few priming frames to set off
the generation process. This task, known as **future video prediction**, has received a lot of attention
recently. Methods based on variational autoencoders (VAE) [2, 15] have been proposed to account for
the stochastic nature of future forecasting, i.e., the plurality of possible continuations, disregarded in
deterministic predictive models [21, 47, 70]. Yet, their blurry predictions have motivated the incorpo-
ration of adversarial training [44], hierarchical architectures [7, 80], fully latent temporal models [22],
or normalizing flows [42]. 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 [33])
in RGB space [21, 24, 28, 72, 81] or in feature space [45], typically using a refinement step to handle
occlusions. Lately, autoregressive methods [53, 79] leveraging a self-attention mechanism [68]
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 [75], a variant of future video prediction, specifies both start and end frames. State or
action-conditioned synthesis [28, 48] guides the frame-by-frame evolution with high-level commands.
We consider video data, such that an individual frame \( x \) which can be encoded as some feature \( E \) with \((output) \) synthesized video. Learnable encoder and decoder modules \( T \) are estimated in an autoregressive manner by the quadruple \( \Delta_i \) and \( \Delta_s \) to ensure that the frames synthesized are both realistic (\( \Delta_i \)) and temporally coherent (\( \Delta_s \)). The two discriminators are then discarded, and the parameters of the transformer \( T \) are estimated with \( E, F \) and \( D \) frozen. At inference time, the transformer is used only once, latter frames being estimated in an autoregressive manner by the quadruple \( D, S, E, F \). See text for more details.

Additional works consider video synthesis based on one [51] or multiple layouts [46, 74, another video [8], or sound [10, 34, 73]. To account for the variety of potential controls, we leverage the flexibility of transformers, and propose an unifying approach to tackle all of these tasks.

3 Context-aware controllable video synthesis

3.1 Overview of the proposed approach

We consider video data, such that an individual frame \( x \) is natively an element of \( \mathcal{X} = \mathbb{R}^{H \times W \times 3} \), which can be encoded as some feature \( z \) in \( \mathcal{Z} = \mathbb{R}^{h \times w \times F} \) with reduced spatial resolution and increased number of channels. We assume that we are given \( X_c \) in \( \mathcal{X}^m \) corresponding to \( m \) successive frames, and our goal is to synthesize a plausible representation of the following \( n \) frames \( X_s \) which lies in \( \mathcal{X}^n \). Given \( X_c \), we can compute the corresponding features \( Z_c \) in \( \mathcal{Z}^m \) using some encoder \( E : \mathcal{X} \rightarrow \mathcal{Z} \) on each frame individually, then use an autoregressive model (a transformer coupled with a quantization step in our case) to predict features \( Z_s \) in \( \mathcal{Z}^n \) formed by the \( n \) features 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 features can finally be converted one by one into the corresponding frames \( X_s \) using some decoder \( D : \mathcal{Z} \rightarrow \mathcal{X} \).

The overall approach is illustrated by Figure 1. The use of a quantizer over a learned codebook in our implementation complicates the architecture a bit, but has several advantages, including the reuse of familiar NLP technology [68] and, perhaps more importantly, affording simple mechanisms for handling different types of inputs (from images to sound for example [9, 31]) and the intrinsically uncertain and multimodal nature of future prediction (by sampling different codebook elements according to their likelihood [30]). Concretely, this choice simply amounts to inserting in our architecture, right after the encoder \( E \), a nearest-neighbor quantizer \( Q : \mathbb{R}^F \rightarrow [1, q] \) parameterized by an \( F \times q \) codebook which, given the embedding \( z \) of a frame, returns a \( h \times w \) matrix of corresponding tokens, that is, the indices of the closest entry in the codebook for feature vector \( z_{i,j} \) for all spatial locations \((i, j)\) in \([1, h] \times [1, w]\). We abuse notation, and identify \( Q \) with its parameterization by this codebook, so we can optimize over \( Q \) just as we optimize over \( E \) instead of naming explicitly its parameters in the rest of this presentation. An “un-quantizer” \( U : [1, q] \rightarrow \mathbb{R}^F \), also parameterized (implicitly) by the codebook and associating with each token the corresponding entry of the codebook, is also inserted right before the decoder \( D \). In this setting, \( T : [1, q]^m \rightarrow [1, q]^n \) takes as input a sequence of tokens, and outputs the tokens for subsequent frames, 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 [30] for related approaches).

The elements of the architecture described so far are by now (individually) rather classical, with learnable, parametric functions $E$, $D$, $Q$ and $T$. Besides putting them all together, and as detailed in the rest of this section, we add several original elements: (a) The encoding/decoding scheme is improved by the use of two discriminators, $\Delta_e$ and $\Delta_d$, respectively, trained in an adversarial manner to ensure that the predicted frames are realistic and temporally consistent. (b) The context frames $X_c$ are themselves updated each time a new frame is predicted in an autoregressive manner (iteratively fill the sequence up to some predefined capacity, then shift to the left, forgetting the first frame and adding the latest synthetic one on the right). (c) The encoder and decoder are linked through a learnable flow module $F$, allowing the context frames to guide the prediction of the synthetic ones. (d) Additional control variables, ranging from object trajectories to audio tracks, can be used in the form of sequence- or frame-level annotations to drive the synthesis by adding the corresponding tokens to the ones passed on to the autoregressive model $T$.

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 [68], illustrated in Figure 2, which we adapt to account for outside control signals (Section 3.3), once again with the corresponding procedure. Further architectural choices are also detailed in Appendix A.

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

**Architecture.** $E$ and $D$ respectively decreases and increases the spatial resolution by using convolutional residual blocks [29], with $(r_k)_{k \in [1:K]}$ the $K$ corresponding resolution levels ($r_k = h_k \times w_k$). It is common practice to augment, as in U-Net [55], the autoencoder with long skip connections between $E$ and $D$ to share information across the two models at these intermediate levels and escape the lossy compression bottleneck. Although we cannot apply this directly to our setting since information only flows through $D$ for predicted timesteps, such skip connections can be established from the encoding stage of a context frame $x_c$ to the decoding stage of a new frame $x_s$ (resulting from features $z_s$). Similar mechanisms [15, 19] have been proposed in the past for video synthesis but they are only copying static background features from a single context frame. We follow works on semantic segmentation [23] and face frontalization [78] and use a flow module $F$ to warp features and produce temporally consistent outputs despite motion. We extend this to multi-frame contexts, with significant performance gains and no additional parameter to be learned.

Concretely, let $e^k$ be features being encoded from $x_c$, and $d^k$ features being decoded from $z_s$ at a given intermediate resolution $r_k$. We first compute all intermediate context features $e^k$ by applying $E$ to $x_c$. We then progressively decode features $d^k$ for the new frame from low ($k = 1$) to high resolution ($k = K$) by iterating over the following steps: (a) apply one decoding sub-module to get $d^k_s$ from $d^{k-1}_s$, (b) use $F$ to refine the optical flow $f^k_c$ (in $\mathbb{R}^{2 \times r_k}$) which estimates the displacement field from $e^k_c$ to $d^k_s$ (as a proxy to the one from $x_c$ to $x_s$), and a fusion mask $m^k_c$ (in $\mathbb{R}^{1 \times r_k}$) which indicates the expected similarity between aligned features $e^k_c = W(e^k_c, f^k_c)$ and $d^k_s$ (also as a proxy for the one in image domain) with $W$ corresponding to a standard warping operation, (c) use $e^k_c$ and $m^k_c$ to update $d^k_s$ with context information (see update rule (1) below), (d) move to resolution level $r_{k+1}$ by going back to (a). We note that $F$ estimates $f^k_c$ and $m^k_c$ in a coarse-to-fine fashion by refining $f^{k-1}_c$ and $m^{k-1}_c$ (see Appendix A for further details). Temporal skip connections at a given resolution level $r_k$ are defined as the following in-place modification of $d^k_s$:

$$d^k_s = \sigma(m^k_c) \odot d^k_s + (1 - \sigma(m^k_c)) \odot e^k_c,$$

where $\sigma$ is the Sigmoid function, and $\odot$ the element-wise product. We note that update rule (1) is quite standard for warping and fusing two streams of spatial information [23, 28, 74]. For concrete implementation of $F$, we build upon LiteFlowNet [32], 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 $E$ and $D$ both in the mask and flow estimation, and in the update (1). This process readily generalizes to multi-frame contextual information (see Appendix D for details). Similar to spatial transformers [33], the warping operation $W$ is differentiable. As a result, gradients from the training losses can backpropagate from $D$ to $E$ through $F$. This allows end-to-end training of the autoencoder even with information from different timesteps in $E$ and $D$. 

4
Training procedure. The global objective is the linear combination of four auxiliary ones:
\[ \mathcal{L} = \lambda_q \mathcal{L}_q + \lambda_r \mathcal{L}_r + \lambda_a \mathcal{L}_a + \lambda_c \mathcal{L}_c, \]  
(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):
\[ \mathcal{L}_q(\mathcal{E}, \mathcal{Q}) = \| \text{sg}(z) - z_q \|_2^2 + \beta \| \text{sg}(z_q) - z \|_2^2, \]  
(3)
where \( \text{sg}(\cdot) \) is the step 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 [67], 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 \( \tilde{x} = \mathcal{D}(z_q) \) 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 [59] pretrained on ImageNet [14]:
\[ \mathcal{L}_r(\mathcal{E}, \mathcal{Q}, \mathcal{F}, \mathcal{D}) = \| x - \tilde{x} \|_1 + \| \text{VGG}(x) - \text{VGG}(\tilde{x}) \|_1. \]  
(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 \( \Delta_i \), made of downsampling residual blocks [29], to encourage realistic outputs. \( \Delta_i \) tries to distinguish real images from reconstructed ones (\( \mathcal{L}_d \)), while \( \mathcal{E} \) and \( \mathcal{D} \) fools \( \Delta_i \) into assuming reconstructed images are as good as real ones (\( \mathcal{L}_a \)):
\[ \mathcal{L}_d(\Delta_i) = \ln(1 + e^{\Delta_i(x)}) + \ln(1 + e^{-\Delta_i(\mathcal{E}(x))}), \]  
(5)
\[ \mathcal{L}_a(\mathcal{E}, \mathcal{Q}, \mathcal{F}, \mathcal{D}) = \ln(1 + e^{\Delta_i(\mathcal{E})(\mathcal{F})}). \]  
(6)

To improve training.

The success of our method relies on accurate motion estimation in \( \mathcal{F} \), a difficult task which benefits from self-supervision [35]. Therefore, we train the autoencoder with augmented views of the input frames as context. Custom augmentations functions \( A : \mathcal{X} \rightarrow \mathcal{X} \) include: rotation, scaling, translation, elastic deformation, and combinations of these. Augmented views are obtained by warping \( x \) by the suitable flow \( a_c : A(x) = W(x, a_c) \), and the inverted flow \( f_c \) from \( A(x) \) 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 \( B : \mathcal{X} \rightarrow \mathcal{X} \) and an occlusion mask \( o_c \) to the augmented frames \( x_c = o_c \odot B(A(x)) \). This augmentation strategy is illustrated in Appendix F. We define the contextual loss as:
\[ \mathcal{L}_c(\mathcal{E}, \mathcal{Q}, \mathcal{F}, \mathcal{D}) = \| o_c - \tilde{f}_c \|_2^2 + \| o'_c - \sigma(\tilde{m}_c) \|_2^2; \]  
(7)
where \( o'_c = W(o_c, f_c) \), and \( \tilde{f}_c \) and \( \tilde{m}_c \) are the flow and mask estimated by \( \mathcal{F} \). In practice, this loss is applied at intermediate resolution levels \( r_k \) for improved training.

3.3 Second stage: predicting temporal dynamics with transformers

Architecture. We follow [20] and adopt an architecture similar to Image-GPT [9] for \( T \). Instead of modeling a single annotated frame as in [54], we design our model to account for sequences of \( N \) such frames to allow prediction of temporal dynamics controlled by ancillary information in the form of video- and frame-level annotations. We have shown, in Section 3.2, how to represent a frame as a sequence of \( h \times w \) 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 \( L = l_v + N * (l_f + h + w) \), 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 \( \hat{o}_i \) (of size \( q \), the number of possible tokens) which scores the likelihood of the \( i + 1 \)th token given all preceding ones.

\(^1\)Our implementation of flow inversion approximation is detailed in Appendix C.
Training procedure. To learn the parameters of $T$, 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:

$$
\mathcal{L}(T) = - \sum_{i=2}^{L} \log \left( \frac{\exp\left( \langle \hat{\tau}_{i-1}, e\tau(i) \rangle \right)}{\sum_j \exp\left( \langle \hat{\tau}_{i-1}, e_j \rangle \right)} \right),
$$

where $\tau(i)$ is the $i^{th}$ ground-truth token, and $e_j$ a vector of 0’s with a 1 in its $j^{th}$ coordinate.

Inference. During inference we use $T$ 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 $\hat{o}$. Although $T$ 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 $T$ (not predicted, even for future frames) to guide the synthesis. We show in the experiments how this control can translate 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 [19] 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 [5], a large and diverse action-recognition dataset with approximately 500K videos. We also test our method on AudioSet-Drums [25] 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 Appendix E.

Metrics. We use the Fréchet video distance (FVD) [66] which measures the distribution gap between real and synthetic videos in the feature space of an Inception3D network [6] pretrained on Kientics-400 [40]. It estimates the visual quality and temporal consistency of samples as well as the diversity in unconditioned scenarios. In conditioned ones, we use another metric for diversity (DIV) which is the mean pixel-wise distance among synthetic trajectories conditioned on the same input. 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) [77] which evaluates a per-frame
We adopt notations and evaluation setups from Table 1. 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 [59] produces sharper videos than using the same loss in the RGB space alone. Predicted frames using image and temporal discriminators ($\Delta_i$, and $\Delta_t$) display greater realism and temporal consistency. The flow module $F$ significantly improves the performance on all metrics by allowing context frames to guide the reconstruction of synthetic ones. The self-supervision of $F$, the use of larger context windows, and longer training times, further improve the quality of the synthesis. $\Delta_t$ 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.

### 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). The use of larger context windows, and longer training times, further improve the quality of the synthesis. $\Delta_t$ 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.
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.

<table>
<thead>
<tr>
<th>Method</th>
<th>Cond.</th>
<th>FVD ↓</th>
<th>Code Avail.</th>
<th>Memory, compute</th>
</tr>
</thead>
<tbody>
<tr>
<td>MoCoGAN [65]</td>
<td>4</td>
<td>503</td>
<td>✓</td>
<td>16GB, 23h</td>
</tr>
<tr>
<td>SVG-FP [15]</td>
<td>2</td>
<td>315</td>
<td>✓</td>
<td>12GB, 6 to 24h*</td>
</tr>
<tr>
<td>CDNA [21]</td>
<td>2</td>
<td>297</td>
<td>✓</td>
<td>10GB, 20h*</td>
</tr>
<tr>
<td>SV2P [2]</td>
<td>2</td>
<td>263</td>
<td>✓</td>
<td>16GB, 24 to 48h*</td>
</tr>
<tr>
<td>SRVP [22]</td>
<td>2</td>
<td>181</td>
<td>✓</td>
<td>36GB, 168h*</td>
</tr>
<tr>
<td>VideoFlow [42]</td>
<td>3</td>
<td>131</td>
<td></td>
<td>12GB, 336h*</td>
</tr>
<tr>
<td>LVT [53]</td>
<td>1</td>
<td>126±3</td>
<td>✓</td>
<td>128GB, 48h</td>
</tr>
<tr>
<td>SAVP [44]</td>
<td>2</td>
<td>116</td>
<td>✓</td>
<td>32GB, 144h</td>
</tr>
<tr>
<td>DVD-GAN-FP [12]</td>
<td>1</td>
<td>110</td>
<td></td>
<td>2TB, 24h*</td>
</tr>
<tr>
<td>Video Transformer (S) [79]</td>
<td>1</td>
<td>106±3</td>
<td></td>
<td>256GB, 33h*</td>
</tr>
<tr>
<td>TriVD-GAN-FP [45]</td>
<td>1</td>
<td>103</td>
<td></td>
<td>1TB, 280h*</td>
</tr>
<tr>
<td>Low res. CCVS (ours)</td>
<td>1</td>
<td>99±2</td>
<td>✓</td>
<td>128GB, 40h</td>
</tr>
<tr>
<td>Video Transformer (L) [79]</td>
<td>1</td>
<td>94±2</td>
<td></td>
<td>512GB, 336h*</td>
</tr>
</tbody>
</table>

High res.** CCVS (ours) 1 80±3 0.729 0.683
+ end frame 2 81±2 0.766 0.839
+ state 1 50±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.

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 \(T\). Note that using a smaller temporal window in \(T\) may allow additional speed-ups.
Table 4: Future video prediction on Kinetics (64 × 64) of 16-frame videos from 5 consecutive input frames.

<table>
<thead>
<tr>
<th>Method</th>
<th>FVD↓</th>
<th>GPU/TPU Mem.</th>
</tr>
</thead>
<tbody>
<tr>
<td>LVT [53]</td>
<td>225</td>
<td>128GB, 48h</td>
</tr>
<tr>
<td>Video Transformer [79]</td>
<td>170±5</td>
<td>2TB, 336h*</td>
</tr>
<tr>
<td>DVD-GAN-FP [12]</td>
<td>69±1</td>
<td>2TB, 144h*</td>
</tr>
<tr>
<td>CCVS (ours)</td>
<td>55±1</td>
<td>128GB, 300h</td>
</tr>
<tr>
<td>TriVD-GAN-FP [45]</td>
<td>26±1</td>
<td>16TB, 160h*</td>
</tr>
</tbody>
</table>

Table 5: Codebook size on Kinetics.

<table>
<thead>
<tr>
<th>Size</th>
<th>FVD↓</th>
<th>SSIM↑</th>
<th>PSNR↑</th>
<th>FVD↓</th>
</tr>
</thead>
<tbody>
<tr>
<td>64</td>
<td>55±1</td>
<td>0.907</td>
<td>31.2</td>
<td>66±2</td>
</tr>
<tr>
<td>256</td>
<td>54±1</td>
<td>0.917</td>
<td>31.6</td>
<td>64±1</td>
</tr>
<tr>
<td>1024</td>
<td>52±1</td>
<td>0.923</td>
<td>32.0</td>
<td>59±1</td>
</tr>
<tr>
<td>4096</td>
<td>51±1</td>
<td>0.928</td>
<td>32.2</td>
<td>61±1</td>
</tr>
<tr>
<td>16384</td>
<td>-</td>
<td>0.963</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>65536</td>
<td>-</td>
<td>0.967</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>∞</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

Table 6: Sound-conditional video synthesis on AudioSet-Drums (64 × 64).

<table>
<thead>
<tr>
<th>Method</th>
<th>Cond. Audio</th>
<th>SSIM↑</th>
<th>PSNR↑</th>
</tr>
</thead>
<tbody>
<tr>
<td>SVG-LP [15]</td>
<td>15</td>
<td>0.971±0.017</td>
<td>0.661±0.011</td>
</tr>
<tr>
<td>Vougioukas et al. [73]</td>
<td>15</td>
<td>√</td>
<td>0.940±0.017</td>
</tr>
<tr>
<td>Sound2Sight [10]</td>
<td>15</td>
<td>√</td>
<td>0.984±0.009</td>
</tr>
<tr>
<td>CCVS (ours)</td>
<td>15</td>
<td>√</td>
<td>0.987±0.001</td>
</tr>
</tbody>
</table>

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 [79]), but requires much less computing resources, and outperforms [79] under similar resources. We also propose high res. CCVS which is not strictly comparable to the prior arts as we use 256 × 256 image resolution for 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 (with the end frame as a video-level annotation) and state-conditioned synthesis (with 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 require realistic video continuations, but also ones which explain the end position of all visible objects. Hence, FVD score is constant despite the additional input. Still, this yields better SSIM for mid-point and one-before-last frames. State-conditioned synthesis improves on FVD and mid-synthesis SSIM as motion becomes near-deterministic. Some synthetic frames 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 [48] 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 [54] 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 $T$ (instead of ranking the likelihood of possible tokens). It allows better reconstructions, yet the prediction FVD is high. Figure 5 shows examples of synthetic continuations conditioned on...
Figure 6: Diversity results for conditioned scenarios. The diversity metric (DIV) measures the mean pixel-wise distance among 10 15-frame synthetic trajectories conditioned on the same frame. We report the mean and std over 100 runs: on whole frames as in [82], or moving parts only by masking static regions (optical flow magnitude between consecutive frames < 20% of the max magnitude) as in [69]. We show, for the same input, the 15th frame of various randomly seeded synthetic trajectories.

5-frame unseen test sequences. CCVS produces realistic and temporally coherent outputs which display various types of motion (e.g., body, hand, camera).

AudioSet-Drums. CCVS achieves top performance on sound-conditioned video synthesis on AudioSet Drums (Table 6). Figure 6 shows quantitative and qualitative insights on the diversity of the synthetic trajectories conditioned on the same input for the three datasets. The diversity metrics (DIV) computed on whole frames is lower on AudioSet Drums than on the other two datasets. This is explained by the fact that motion is quite repetitive and involves a limited portion of the frame. The same metric on moving parts only, and the end position of the drummer’s hand and upper left cymbal in qualitative samples, show the diversity of synthetic trajectories. An ablation of CCVS with/without audio guidance as well as more qualitative results on diversity can be found in Appendix E.

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 [17, 61]. Methods sparing GPU-hours like ours are crucial to make AI less polluting [43, 61], and move from a “Red” to a “Green” AI [58]. Future work will include exploring new codebook strategies and synthesis guided by textual information.

Limitations. CCVS uses a complex architecture and a two-stage training strategy. Simplification of both is an interesting direction for improving the method. Moreover, CCVS lacks global regularization of motion (flow computed on pairs of timesteps), and its efficiency relies on recycling context information such that synthesizing content from scratch (i.e., no input frame given) remains difficult.

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 [49], 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.
Acknowledgements

This work was granted access to the HPC resources of IDRIS under the allocation 2020-AD011012227 made by GENCI. It was funded in part by the French government under management of Agence Nationale de la Recherche as part of the “Investissements d’avenir” program, reference ANR-19-P3IA-0001 (PRAIRIE 3IA Institute). JP was supported in part by the Louis Vuitton/ENS chair in artificial intelligence and the Inria/NYU collaboration. We thank the reviewers for useful comments.

References


