arXiv:1903.01434v2 [cs.CV] 10 Jun 2019

VideoFlow: A Flow-Based Generative Model for Video

Manoj Kumar [1] **Mohammad Babaeizadeh** [2] **Dumitru Erhan** [1] **Chelsea Finn** [1] **Sergey Levine** [1] **Laurent Dinh** [1] **Durk Kingma** [1]

Abstract

Generative models that can model and predict sequences of future events can, in principle, learn to capture complex real-world phenomena, such as physical interactions. In particular, learning predictive models of videos offers an especially appealing mechanism to enable a rich understanding of the physical world: videos of real-world interactions are plentiful and readily available, and a model that can predict future video frames can not only capture useful representations of the world, but can be useful in its own right, for problems such as model-based robotic control. However, a central challenge in video prediction is that the future is highly uncertain: a sequence of past observations of events can imply many possible futures. Although a number of recent works have studied probabilistic models that can represent uncertain futures, such models are either extremely expensive computationally (as in the case of pixel-level autoregressive models), or do not directly optimize the likelihood of the data. In this work, we propose a model for video prediction based on normalizing flows, which allows for direct optimization of the data likelihood, and produces high-quality stochastic predictions. To our knowledge, our work is the first to propose multi-frame video prediction with normalizing flows. We describe an approach for modeling the latent space dynamics, and demonstrate that flow-based generative models offer a viable and competitive approach to generative modeling of video.

1. Introduction

Exponential progress in the capabilities of computational hardware, paired with a relentless effort towards greater insights and better methods, has pushed the field of machine learning from relative obscurity into the mainstream. Progress in the field has translated to improvements in various capabilities, such as classification of images (Krizhevsky et al., 2012), machine translation (Vaswani et al., 2017) and super-human game-playing agents (Mnih et al., 2013; Silver et al., 2017), among others. However, the application of machine learning technology has been largely constrained to situations where large amounts of supervision is available, such as in image classification or machine translation, or where highly accurate simulations of the environment are available to the learning agent, such as in game-playing agents. An appealing alternative to supervised learning is to utilize large unlabeled datasets, combined with predictive generative models. In order for a complex generative model to be able to effectively predict future events, it must build up an internal representation of the world. For example, a predictive generative model that can predict future frames in a video would need to model complex real-world phenomena, such as physical interactions. This provides an appealing mechanism for building models that have a rich understanding of the physical world, without any labeled examples. Videos of real-world interactions are plentiful and readily available, and a large generative model can be trained on large unlabeled datasets containing many video sequences, thereby learning about a wide range of real-world phenoma. Such a model could be useful for learning representations for further downstream tasks (Mathieu et al., 2016), or could even be used directly in applications where predicting the future enables effective decision making and control, such as robotics (Finn et al., 2016). A central challenge in video prediction is that the future is highly uncertain: a short sequence of observations of the present can imply many possible futures. Although a number of recent works have studied probabilistic models that can represent uncertain futures, such models are either extremely expensive computationally (as in the case of pixel-level autoregressive models), or do not directly optimize the likelihood of the data.

In this paper, we study the problem of stochastic prediction, focusing specifically on the case of conditional video prediction: synthesizing raw RGB video frames conditioned on a short context of past observations (Ranzato et al., 2014; Srivastava et al., 2015; Vondrick et al., 2015; Xingjian et al., 2015; Boots et al., 2014). Specifically, we propose a new class of video prediction models that can provide exact

[1]Google Brain [2]University of Illinois at Urbana-Champaign. Correspondence to: Manoj Kumar <mechcoder@google.com>.

likelihoods, generate diverse stochastic futures, and accurately synthesize realistic and high-quality video frames. The main idea behind our approach is to extend flow-based generative models (Dinh et al., 2014; 2016) into the setting of conditional video prediction. Although methods based on variational autoencoders (Babaeizadeh et al., 2017; Denton & Fergus, 2018; Lee et al., 2018), and pixel-level autoregressive models (Hochreiter & Schmidhuber, 1997; Graves, 2013; van den Oord et al., 2016b;c; Van Den Oord et al., 2016) have previously been studied for stochastic predictive generation, flow-based models have received comparatively much less attention, and to our knowledge have been applied only to generation of non-temporal data, such as images (Kingma & Dhariwal, 2018), and to audio sequences (Prenger et al., 2018). Conditional generation of videos presents its own unique challenges: the high dimensionality of video sequences makes them difficult to model as individual datapoints. Instead, we learn a latent dynamical system model that predicts future values of the flow model's latent state. This induces Markovian dynamics on the latent state of the system, replacing the standard unconditional prior distribution. We further describe a practically applicable architecture for flow-based video prediction models, inspired by the Glow model for image generation (Kingma & Dhariwal, 2018), which we call VideoFlow.

Our empirical results show that VideoFlow achieves results that are competitive with the state-of-the-art in stochastic video prediction on the action-free BAIR dataset, with quantitative results that rival the best VAE-based models. VideoFlow also produces excellent qualitative results, and avoids many of the common artifacts of models that use pixel-level mean-squared-error for training (e.g., blurry predictions), without the challenges associated with training adversarial models. Compared to models based on pixel-level autoregressive prediction, VideoFlow achieves substantially faster test-time image synthesis [1], making it much more practical for applications that require real-time prediction, such as robotic control (Finn & Levine, 2017). Finally, since VideoFlow directly optimizes the likelihood of training videos, without relying on a variational lower bound, we can evaluate its performance directly in terms of likelihood values.

2. Related Work

Early work on prediction of future video frames focused on deterministic predictive models (Ranzato et al., 2014; Srivastava et al., 2015; Vondrick et al., 2015; Xingjian et al., 2015; Boots et al., 2014). Much of this research on deterministic models focused on architectural changes, such as incorporating pixel transformations (Finn et al., 2016; De Brabandere et al., 2016; Liu et al., 2017) and predictive coding architectures (Lotter et al., 2017), as well as different generation objectives (Mathieu et al., 2016; Vondrick & Torralba, 2017; Walker et al., 2015). With models that can successfully model many deterministic environments, the next key challenge is to address stochastic environments by building models that can effectively reason over uncertain futures. Real-world videos are always somewhat stochastic, either due to events that are inherently random, or events that are caused by unobserved or partially observable factors, such as off-screen events, humans and animals with unknown intentions, and objects with unknown physical properties. In such cases, since deterministic models can only generate one future, these models either disregard potential futures or produce blurry predictions that are the superposition or averages of possible futures.

A variety of methods have sought to overcome this challenge by incorporating stochasticity, via three types of approaches: models based on variational auto-encoders (VAEs) (Kingma & Welling, 2013; Rezende et al., 2014), generative adversarial networks (Goodfellow et al., 2014), and autoregressive models (Hochreiter & Schmidhuber, 1997; Graves, 2013; van den Oord et al., 2016b;c; Van Den Oord et al., 2016). Among these models, techniques based on variational autoencoders have been explored most widely (Babaeizadeh et al., 2017; Denton & Fergus, 2018; Lee et al., 2018). These models use latent random variables to represent stochastic events. They are trained by maximizing the evidence lower bound using an inference network, which estimates the posterior distribution over these latent variables and is typically conditioned on the current or future frames, such that the whole model resembles a sequence-level autoencoder. Unlike our proposed method, none of these models maximize the log-likelihood directly, since they rely on optimizing the evidence lower bound.

To our knowledge, the only prior class of video prediction models that directly maximize the log-likelihood of the data are auto-regressive models (Hochreiter & Schmidhuber, 1997; Graves, 2013; van den Oord et al., 2016b;c; Van Den Oord et al., 2016), which can be applied to model the joint distribution of raw video pixels by means of an autoregressive model that generates the video one pixel at a time (Kalchbrenner et al., 2017). However, synthesis with such models is typically inherently sequential, making synthesis substantially inefficient on modern parallel hardware. Prior work has aimed to speed up training and synthesis with such auto-regressive models (Reed et al., 2017; Ramachandran et al., 2017). However, Babaeizadeh et al. (2017) show that the predictions from these models are sharp but noisy and that the proposed VAE model produces substantially better predictions, especially for longer horizons. In contrast to autoregressive models, we find that our proposed method exhibits faster sampling, while still directly optimizing the log-likelihood and producing high-quality long-term

[1] We generate 64x64 videos of 20 frames in less than 3.5 seconds on a NVIDIA P100 GPU.

predictions.

3. Preliminaries: Flow-Based Generative Models

Flow-based generative models (Dinh et al., 2014; 2016) have received comparatively little attention in the research community. However, these models have a unique set of advantages: exact latent-variable inference, exact log-likelihood evaluation, and efficiency in terms of both inference and synthesis. The basic principles behind flow-based generative models were first described by Deco & Brauer (1995), but were re-discovered and more fully developed in a modern context by Dinh et al. (2014) as *Non-linear Independent Component Estimation* (NICE), with further refinements and extensions proposed by Dinh et al. (2016) (RealNVP). To our knowledge, in the domain of image generation, prior work has only applied such models to generate static images (Kingma & Dhariwal, 2018) or sound (Prenger et al., 2018), while we propose a dynamics-enabled normalizing flow model in our work. Here, we first summarize the foundations of modern normalizing flow models.

Let $\mathcal{D} = \{\mathbf{x}^{(i)}\}_{i=1}^{N}$ be our dataset of i.i.d. observations of a random variable $\mathbf{x}$ with an unknown true distribution $p^*(\mathbf{x})$. Our data consist of 8-bit videos, with each dimension rescaled to the domain $[0, 255/256]$. We add a small amount of uniform noise to the data, $\mathbf{u} \sim \mathcal{U}(0, 1/256.)$, matching its discretization level (Dinh et al., 2016; Kingma & Dhariwal, 2018). Let $q(\mathbf{x})$ be the resulting empirical distribution corresponding to this scaling and addition of noise. Note that additive noise is required to prevent $q(\mathbf{x})$ from having infinite densities at the datapoints, which can result in ill-behaved optimization of the log-likelihood; it also allows us to recast maximization of the log-likelihood as minimization of a KL divergence.

Let $p_{\boldsymbol{\theta}}(\mathbf{x})$ be our model of the data with parameters $\boldsymbol{\theta}$. Maximization of the log-likelihood w.r.t. $\boldsymbol{\theta}$, is then equivalent to minimization the KL divergence w.r.t. the parameters $\boldsymbol{\theta}$: $D_{KL}(q(\mathbf{x})||p_{\boldsymbol{\theta}}(\mathbf{x}))$. This objective measures the expected per-datapoint compression cost in nats or bits (depending on the base); see Dinh et al. (2016).

In flow-based generative models (Dinh et al., 2014; 2016), we model the data as if it was first generated from a latent space $p_{\boldsymbol{\theta}}(\mathbf{z})$, then transformed to $\mathbf{x}$ through an *invertible* transformation:

$$\mathbf{z} \sim p_{\boldsymbol{\theta}}(\mathbf{z}) \tag{1}$$

$$\mathbf{x} = \mathbf{g}_{\boldsymbol{\theta}}(\mathbf{z}) \tag{2}$$

where $\mathbf{z}$ is the latent variable and $p_{\boldsymbol{\theta}}(\mathbf{z})$ has a simple, tractable density, such as a spherical multivariate Gaussian distribution: $p_{\boldsymbol{\theta}}(\mathbf{z}) = \mathcal{N}(\mathbf{z}; 0, \mathbf{I})$. The function $\mathbf{g}_{\boldsymbol{\theta}}(..)$ is invertible, also called *bijective*, such that given a datapoint $\mathbf{x}$, latent-variable inference is done by $\mathbf{z} = \mathbf{f}_{\boldsymbol{\theta}}(\mathbf{x}) = \mathbf{g}_{\boldsymbol{\theta}}^{-1}(\mathbf{x})$. We will omit subscript $\boldsymbol{\theta}$ from $\mathbf{f}_{\boldsymbol{\theta}}$ and $\mathbf{g}_{\boldsymbol{\theta}}$.

We focus on functions where $\mathbf{f}$ (and, likewise, $\mathbf{g}$) is composed of a sequence of invertible transformations: $\mathbf{f} = \mathbf{f}_1 \circ \mathbf{f}_2 \circ \cdots \circ \mathbf{f}_K$. Under the *change of variables* of Eq. (2), the probability density function (PDF) of the model given a datapoint can be written as:

$$\log p_{\boldsymbol{\theta}}(\mathbf{x}) = \log p_{\boldsymbol{\theta}}(\mathbf{z}) + \log |\det(d\mathbf{z}/d\mathbf{x})| \tag{3}$$

$$= \log p_{\boldsymbol{\theta}}(\mathbf{z}) + \sum_{i=1}^{K} \log |\det(d\mathbf{h}_i/d\mathbf{h}_{i-1})| \tag{4}$$

where $\mathbf{h}_0 \triangleq \mathbf{x}$ and $\mathbf{h}_K \triangleq \mathbf{z}$. The scalar value $|\det(d\mathbf{h}_i/d\mathbf{h}_{i-1})|$ is the absolute value of the determinant of the Jacobian matrix $(d\mathbf{h}_i/d\mathbf{h}_{i-1})$, also called the *Jacobian determinant*. This value is the change in log-density when going from $\mathbf{h}_{i-1}$ to $\mathbf{h}_i$ under transformation $\mathbf{f}_i$. While computation of the Jacobian determinant is expensive in the general case, its value can be surprisingly simple to compute for certain choices of transformations, as explored in prior work (Deco & Brauer, 1995; Dinh et al., 2014; 2016; Rezende & Mohamed, 2015; Kingma et al., 2016; Kingma & Dhariwal, 2018). The basic idea used in this work, is to choose transformations whose Jacobian $d\mathbf{h}_i/d\mathbf{h}_{i-1}$ is a triangular matrix, diagonal matrix or a permutation matrix. For permutation matrices, the Jacobian determinant is one. For triangular and diagonal Jacobian matrices $L = d\mathbf{h}_i/d\mathbf{h}_{i-1}$, the determinant is simply the product of diagonal terms, such that:

$$\log |\det(L)| = \sum_j \log |L_{j,j}| \tag{5}$$

where $\log()$ takes the element-wise logarithm, and $L_{j,j}$ is the j-th element on the diagonal of matrix L.

4. Proposed Architecture

We propose a generative flow for video, extending the recently proposed *Glow* (Kingma & Dhariwal, 2018) and *RealNVP* (Dinh et al., 2016) architectures.

In our model, we break up the latent space $\mathbf{z}$ into separate latent variables per timestep: $\mathbf{z} = \{\mathbf{z}_t\}_{t=1}^{T}$. The latent variable $\mathbf{z}_t$ at timestep t is an invertible transformation of a corresponding frame of video: $\mathbf{x}_t = \mathbf{g}_{\boldsymbol{\theta}}(\mathbf{z}_t)$. Furthermore, like in (Dinh et al., 2016; Kingma & Dhariwal, 2018), we use a multi-scale architecture (Fig. 1): the latent variable $\mathbf{z}_t$ is composed of a stack of multiple levels: where each level l encodes information about frame $\mathbf{x}_t$ at a particular scale: $\mathbf{z}_t = \{\mathbf{z}_t^{(l)}\}_{l=1}^{L}$, one component $\mathbf{z}_t^{(l)}$ per level.

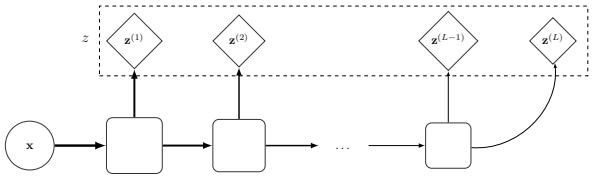

Figure 1: The flow model uses a multi-scale architecture using several levels of stochastic variables. At each level, the input is passed through K flows to output a stochastic variable and the input to the next series of flow. The final series of flow just output the final stochastic variable.

4.1. Autoregressive latent dynamics model

As in equation (1), we need to choose a form of latent prior $p_{\boldsymbol{\theta}}(\mathbf{z})$. We use the following autoregressive factorization for the latent prior:

$$p_{\boldsymbol{\theta}}(\mathbf{z}) = \prod_{t=1}^{T} p_{\boldsymbol{\theta}}(\mathbf{z}_t|\mathbf{z}_{<t}) \tag{6}$$

where $\mathbf{z}_{<t}$ denotes the latent variables of frames prior to the t-th timestep: $\{\mathbf{z}_1, ..., \mathbf{z}_{t-1}\}$. We specify the conditional prior $p_{\boldsymbol{\theta}}(\mathbf{z}_t|\mathbf{z}_{<t})$ as having the following factorization:

$$p_{\boldsymbol{\theta}}(\mathbf{z}_t|\mathbf{z}_{<t}) = \prod_{l=1}^{L} p_{\boldsymbol{\theta}}(\mathbf{z}_t^{(l)}|\mathbf{z}_{<t}^{(l)}, \mathbf{z}_t^{(>l)}) \tag{7}$$

where $\mathbf{z}_{<t}^{(l)}$ is the set of latent variables at previous timesteps and at the same level l, while $\mathbf{z}_t^{(>l)}$ is the set of latent variables at the same timestep and at higher levels. See figure 2 for a graphical illustration of the dependencies.

We let each $p_{\boldsymbol{\theta}}(\mathbf{z}_t^{(l)}|\mathbf{z}_{<t}^{(l)}, \mathbf{z}_t^{(>l)})$ be a conditionally factorized Gaussian density:

$$p_{\boldsymbol{\theta}}(\mathbf{z}_t^{(l)}|\mathbf{z}_{<t}^{(l)}, \mathbf{z}_t^{(>l)}) = \mathcal{N}(\mathbf{z}_t^{(l)}; \boldsymbol{\mu}, \sigma) \tag{8}$$

$$\text{where } (\boldsymbol{\mu}, \log\sigma) = NN_{\boldsymbol{\theta}}(\mathbf{z}_{<t}^{(l)}, \mathbf{z}_t^{(>l)}) \tag{9}$$

where $NN_{\boldsymbol{\theta}}()$ is a deep residual network (He et al., 2015) described in section 4.3.

4.2. Invertible neural networks

As explained in the section 3, the observed variables $\mathbf{x}$ are modeled as an invertible function of the latent variable $\mathbf{z}$. We let each individual frame in the video be modeled as function (a normalizing flow) of the set of corresponding latent variable: $\mathbf{x}_t = \mathbf{g}_{\boldsymbol{\theta}}(\mathbf{z}_t) = \mathbf{g}_{\boldsymbol{\theta}}(\{\mathbf{z}_t^{(l)}\}_{l=1}^{L})$; see figure 1 for an illustration. For this flow $\mathbf{g}_{\boldsymbol{\theta}}$, we use the multi-scale Glow architecture as introduced in (Kingma & Dhariwal, 2018), which builds upon the multi-scale flow introduced in (Dinh et al., 2016). We refer to (Dinh et al., 2016; Kingma & Dhariwal, 2018) for more details.

Note that in our architecture we have chosen to let the prior $p_{\boldsymbol{\theta}}(\mathbf{z})$, as described in eq. (6), model temporal dependencies in the data, while constraining the flow $\mathbf{g}_{\boldsymbol{\theta}}$ to act on separate frames of video. We have experimented with using 3-D convolutional flows, but found this to be computationally overly expensive compared to an autoregressive prior; in terms of both number of operations and number of parameters.

A separate concern is that of temporal border effects. Due to memory limits, we found it only feasible to perform SGD with a small number of sequential frames per gradient step. In case of 3-D convolutions, this would make the temporal dimension considerably smaller during training than during synthesis; this would change the model's input distribution between training and synthesis, which often leads to various artifacts. This temporal border effect is not present in our architecture. Using 2-D convolutions in our flow $\mathbf{f}_{\boldsymbol{\theta}}$, and with autoregressive priors, allows us to synthesize arbitrarily long sequences with introducing temporal border effects.

4.3. Residual Network Architecture

Here we'll describe the architecture for the residual network $NN_{\boldsymbol{\theta}}()$ that maps $\mathbf{z}_{<t}^{(l)}, \mathbf{z}_t^{(>l)}$ to $(\boldsymbol{\mu}_t^{(l)}, \log\sigma_t^{(l)})$. Let $\mathbf{h}_t^{(>l)}$ be the tensor representing $\mathbf{z}_t^{(>l)}$ after the split operation between levels in the multi-scale architecture. We apply a 1×1 convolution over $\mathbf{h}_t^{(>l)}$ and concatenate this across channels to each latent from the previous time-step and the same-level independently. In this way, we obtain $((W\mathbf{h}_t^{(>l)}; \mathbf{z}_{t-1}^{(l)}), (W\mathbf{h}_t^{(>l)}; \mathbf{z}_{t-2}^{(l)}) \dots (W\mathbf{h}_t^{(>l)}; \mathbf{z}_{t-n}^{(l)}))$. We transform these values into $(\boldsymbol{\mu}_t^{(l)}, \log\sigma_t^{(l)})$ via a stack of residual blocks. We obtain a reduction in parameter count by sharing parameters across every 2 time-steps via 3-D convolutions in our residual blocks.

Each 3-D residual block consists of three layers. The first layer has a filter size of 2x3x3 with 512 output channels followed by a ReLU activation. The second layer has two $1 \times 1 \times 1$ convolutions via the Gated Activation Unit (Van Den Oord et al., 2016; van den Oord et al., 2016a). The third layer has a filter size of $2 \times 3 \times 3$ with the number of output channels determined by the level. This block is replicated three times in parallel, with dilation rates 1, 2 and 4, after which the results of each block, in addition to the input of the residual block, are summed.

The first two layers are initialized using a Gaussian distribution and the last layer is initialized to zeroes. In that way, the residual network behaves as an identity network during initialization allowing stable optimization. After applying a sequence of residual blocks, we use the last temporal activation that should capture all context. We apply a final 1×1 convolution to this activation to obtain $(\Delta\mathbf{z}_t^{(l)}, \log\sigma_t^{(l)})$. We then add $\Delta\mathbf{z}_t^{(l)}$ to $\mathbf{z}_{t-1}^{(l)}$ to a *temporal skip connection* to out-

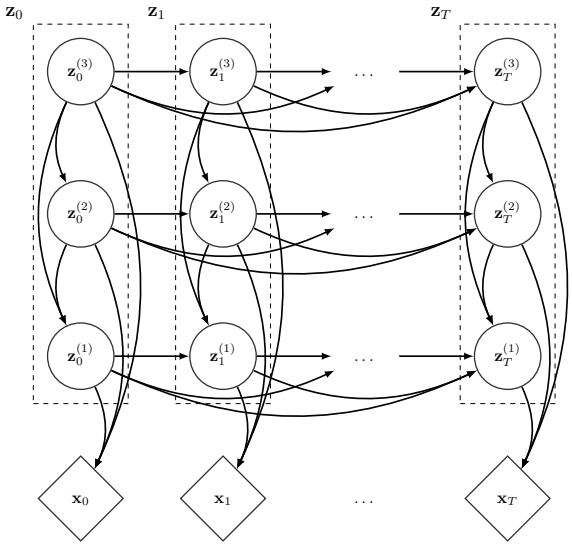

Figure 2: The input at each timestep $\mathbf{x}_t$ is encoded into multiple levels of stochastic variables $(\mathbf{z}_t^{(1)}, \ldots, \mathbf{z}_t^{(L)})$. We model those levels through a sequential process $\prod_t \prod_l p(\mathbf{z}_t^{(l)} \mid \mathbf{z}_{<t}^{(l)}, z_t^{(>l)})$.

put $\boldsymbol{\mu}_t^{(l)}$. This way, the network learns to predict the change in latent variables for a given level.

5. Quantitative Experiments

We evaluate the performance of VideoFlow on a toy Stochastic Movement Dataset (Babaeizadeh et al., 2017) and the BAIR robot pushing dataset (Ebert et al., 2017). We provide ablations of the key components of our model to quantify their effect. Finally, we provide quantitative comparisons to previous state-of-the-art stochastic video generation baselines. The full set of hyperparameters of the VideoFlow model is described in the supplementary material.

Dataset	Bits-per-pixel
BAIR action free	1.87

Table 1: We report the average bits-per-pixel across 10 target frames with 3 conditioning frames for the BAIR action-free dataset.

5.1. Video modelling with the Stochastic Movement Dataset

We use VideoFlow to model the Stochastic Movement Dataset used in prior work (Babaeizadeh et al., 2017). In this dataset, the first frame of every video consists of a shape placed near the center of a 64x64x3 resolution gray background with its type, size and color randomly sampled. The shape then randomly moves in one of eight directions, (up, down, left, right, up-left, upright, down-left, down-right)

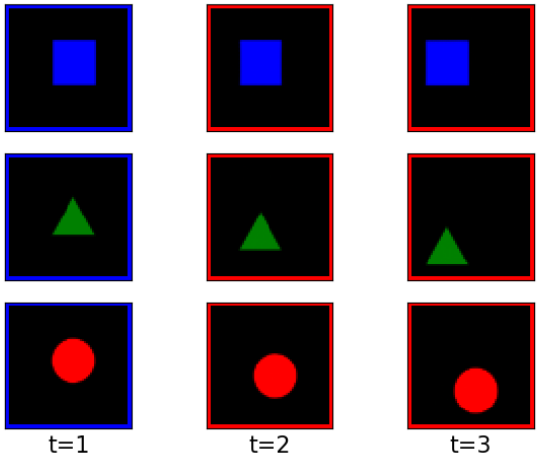

Figure 3: We condition the VideoFlow model with the frame at $t = 1$ and display generated trajectories at $t = 2$ and $t = 3$ for three different shapes.

with constant speed. Babaeizadeh et al. (2017) show that conditioned on the first frame, their latent variable stochastic model is able to generate all plausible trajectories of the shape while a deterministic model just averages out all eight possible directions in pixel space.

Since the shape moves with a uniform speed, we should be able to model the position of the shape at the $(t+1)^{th}$ step using only the position of the shape at the t^{th} step. More specifically, given the frame at $t = 1$, i.e if the shape is at the center, the model should learn a distribution over 8 positions to generate the frame at $t = 2$. Given a frame at any other t the model should learn a deterministic position of the shape for $t + 1$. Using this insight, we extract random temporal patches of 2 frames from each video of 3 frames. We then use the VideoFlow model to maximize the log-likelihood of the second frame given the first, i.e the model looks back at just one frame. We observe that the bits-per-pixel on the holdout set reduces to a very low values between 0.05 and 0.09 bits-per-pixel across multiple hyperparameter runs. We then generate videos conditioned on the first frame with the shape at the center. On inspection of these videos, we observe that the model consistently predicts the future trajectory of the shape to be one of the eight random directions.

5.2. Video Modeling with the BAIR Dataset

We use the action-free version of the BAIR robot pushing dataset (Ebert et al., 2017) that contain videos of a Sawyer robotic arm with resolution 64x64. In the absence of actions, the task of video generation is completely unsupervised with multiple plausible trajectories due to the partial observability of the environment and stochasticity of the robot actions.

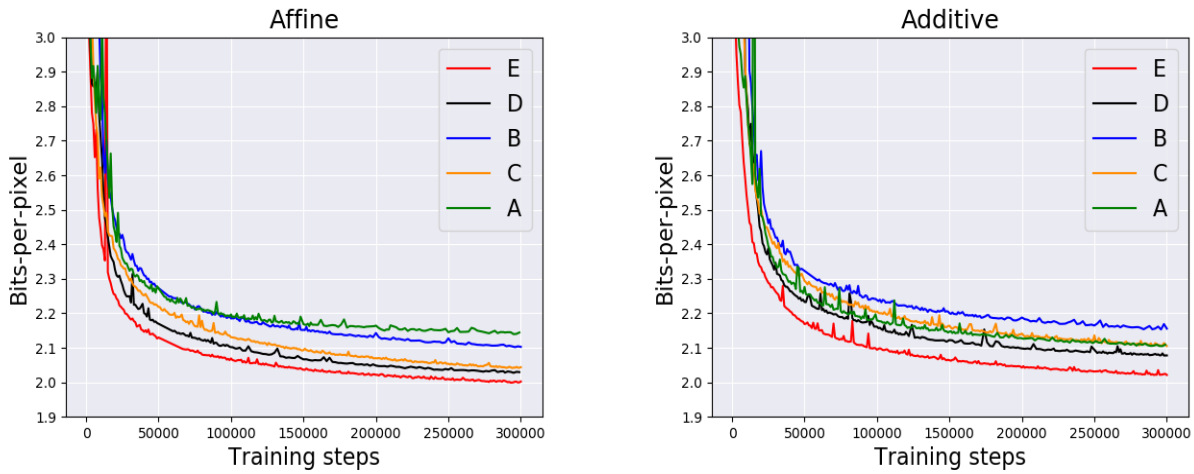

Figure 4: **B: baseline**, **A: Temporal Skip Connection**, **C: Dilated Convolutions + GATU**, **D: Dilation Convolutions + Temporal Skip Connection**, **E: Dilation Convolutions + Temporal Skip Connection + GATU**. We plot the holdout bits-per-pixel on the BAIR action-free dataset for different ablations of our VideoFlow model.

For each video we extract the first 13 frames and take a random temporal patch of 4 frames due to memory constraints. Using Equation 6, we then train our VideoFlow model to maximize the log-likelihood of the 4th frame given the context of 3 previous frames; the residual network in section 4.3 looks back $n = 3$ frames. This stochastic objective gives is an unbiased estimator of the log-likelihood of frame 4 to 13, conditioned on the first three frames. We constrained the range to the first 13 frames in order to be compatible with the results with previous models of this dataset (Babaeizadeh et al., 2017; Lee et al., 2018). We set apart 512 videos from the training set as a validation set on which hyper-parameters are optimized.

For evaluation, we use the first 3 frames as ground-truth conditioning frames. For each of the remaining 10 target frames, we compute the bits-per-pixel given the window of 3 previous frames. We then average this across all the 10 target frames and the test set.

5.3. Ablation Studies

Through an ablation study, we experimentally evaluate the importance of the following components of our VideoFlow model: (1) the use of temporal skip connections, (2) the use Gated Activation Unit (GATU) instead of ReLUs in the residual network (section 4.3), and (3) the use of dilations in $NN_{\boldsymbol{\theta}}()$.

We start with a VideoFlow model with 256 channels in the coupling layer, 16 steps of flow and remove the components mentioned above to create our baseline. We use four different combinations of our components (described in Fig. 4) and keep the rest of the hyperparameters fixed across those combinations. For each combination we plot the mean bits-per-pixel on the holdout BAIR-action free dataset over 300K training steps for both affine and additive coupling in Figure 4. For both the coupling layers, we observe that the VideoFlow model with all the components provide a significant boost in bits-per-pixel over our baseline.

We also note that other combinations—dilated convolutions + GATU (C) and dilated convolutions + the temporal skip connection —improve over the baseline. Finally, we experienced that increasing the receptive field in $NN_{\boldsymbol{\theta}}()$ using dilated convolutions alone in the absence of the temporal skip connection or the GATU makes training highly unstable.

5.4. Comparison with stochastic video-generation baselines

We compare against two state-of-the-art stochastic video generation models, SAVP-VAE (Lee et al., 2018) and SV2P (Babaeizadeh et al., 2017). We use the implementation of these models in the open-source Tensor2Tensor (Vaswani et al., 2018) library. We train these baseline video models to predict ten frames given three conditioning frames, ensuring that all the video models have seen a total of 13 frames during training.

Both these models use variations of temporal VAEs which optimize a lower bound on the log-likelihood and hence are not directly comparable to our model. To make a quantitative comparison with the baselines, we follow the metrics proposed in prior work (Babaeizadeh et al., 2017; Lee et al., 2018). For a given set of conditioning frames in the BAIR action-free test-set, we generate 100 videos from each of the

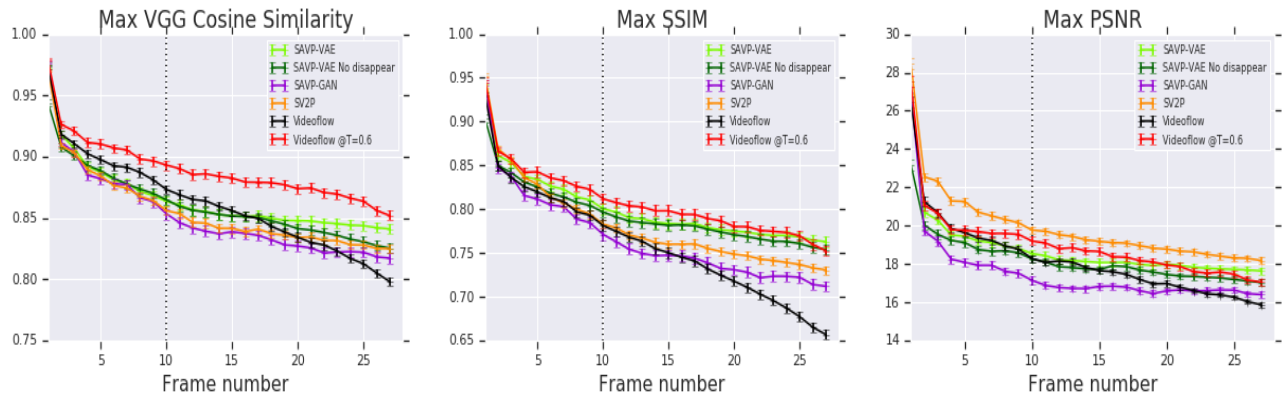

Figure 5: For a given set of conditioning frames on the BAIR action-free we sample 100 videos from each of the stochastic video generation models. We choose the video closest to the ground-truth on the basis of PSNR, SSIM and VGG perceptual metrics and report the best possible value for each of these metrics. All the models were trained using ten target frames but are tested to generate 27 frames. For all the reported metrics, **higher is better**.

stochastic models. We then compute the closest of these generated videos to the ground truth according to three different metrics, PSNR (Peak Signal to Noise Ratio), SSIM (Structural Similarity) (Wang et al., 2004) and cosine similarity using features obtained from a pretrained VGG network (Dosovitskiy & Brox, 2016; Johnson et al., 2016). [2]. We report our findings in Figure 5. This metric helps us understand if the true future lies in the set of all plausible futures according to the video model (and the implicit embedding space of each of the metrics).

In prior work, (Lee et al., 2018) and (Babaeizadeh et al., 2017) do not train a stochastic decoder to learn the variance in pixel space, rather they use a deterministic decoder and effectively treat this variance as a hyperparameter. They search for the variance on a grid of extremely small values on a log-scale using a two stage training procedure. They show that this greatly improves training stability and removes pixel-level noise during generation.

We can remove pixel-level noise in our VideoFlow model resulting in higher quality videos at the cost of diversity by sampling videos at a lower temperature, analogous to the low-temperature procedure in (Kingma & Dhariwal, 2018). For a network trained with additive coupling layers, we can sample the t^{th} frame x_t from $P(x_t|x_{<t})$ with a temperature T simply by scaling the standard deviation of the latent gaussian distribution $P(z_t|z_{<t})$ by a factor of T. To achieve a balance between quality and diversity, we tune the temperature using the maximum VGG similarity across 100 video samples with the ground-truth as a metric[3]. We report results with a temperature of 1.0 and the optimal temperature in Figure 5.

For SAVP-VAE, we notice that the hyperparameters that perform the best on these metrics are the ones that have disappearing arms. For completeness, we report these numbers as well as the numbers for the best performing SAVP models that do not have disappearing arms. Our model with optimal temperature performs better or as well as the SAVP-VAE model on the VGG-based similarity metrics, which correlate well with human perception (Zhang et al., 2018) and SSIM. Our model with temperature $T = 1.0$ is also competent with state-of-the-art video generation models on these metrics. PSNR is explicitly a pixel-level metric, which the VAE models incorporate as part of its optimization objective. VideoFlow on the other-hand models the conditional probability of the distribution of frames, hence as expected it underperforms on PSNR.

We also computed the variational bound of the bits-per-pixel loss, via importance sampling, from the posteriors for the SAVP-VAE and SV2P models. Neither of these models estimate a pixel-level variance, which is required for estimating the loss; we estimated the optimal pixel-level variance for both models. We obtain high values of bits-per-pixel, larger than 6, for these models. We attribute this to the optimization objective of these models: they do not optimize the variational bound on the log-likelihood directly due to the presence of a $\beta \neq 1$ term in their objective and scheduled sampling (Bengio et al., 2015).

5.5. Generation time

For our model used to demonstrate qualitative results using additive coupling layers, sampling 20 frames of 64x64 resolution takes less than 3.5 seconds on an NVIDIA P100

[2]Our baselines are also tuned using this VGG-based cosine similarity metric on a search grid available in the appendix

[3]The temperature was tuned on a linear scale between 0.1 and 1.0 on the validation set.

GPU. To our knowledge, the fastest autoregressive model for video (Reed et al., 2017) that models log-likelihood directly generates a frame every 3 seconds[4].

5.6. Out-of-sequence detection

We use our trained VideoFlow model, conditioned on 3 frames as explained in Section 5.2, to detect the plausibility of a temporally inconsistent frame to occur in the immediate future. To do this, we condition the model on the first three frames of a test-set video $X_{<4}$ to obtain a distribution $P(X_4|X_{<4})$ over its 4th frame X_4. We then compute the likelihood of the t^{th} frame X_t of the same video to occur as the 4th time-step using this distribution. i.e, $\mathcal{P}(X_4 = X_t|X_{<4})$ for $t = 4 \dots 13$. We average the corresponding bits-per-pixel values across the test set and report our findings in Figure 6. We find that our model assigns a monotonically decreasing log-likelihood to frames that are more far out in the future and hence less likely to occur in the 4th time-step.

Secondly, for the distribution $P(X_4|X_{<4})$ obtained from each test-set video as explained above, we then randomly sample another video from the test-set and choose it's 4th frame which we describe as $X_4^{'}$. We then compute the mean bits-per-pixel obtained by $\mathcal{P}(X_4 = X_4^{'}|X_{<4})$ across the test set. We repeat this experiment 1000 times and observe the mean across the 1000 trials to be **8.876** with a standard error of **0.002**. Our results reflect the intuition that the frames from a different video should be less likely to occur in the 4th timestep than the same video but from a different time-step.

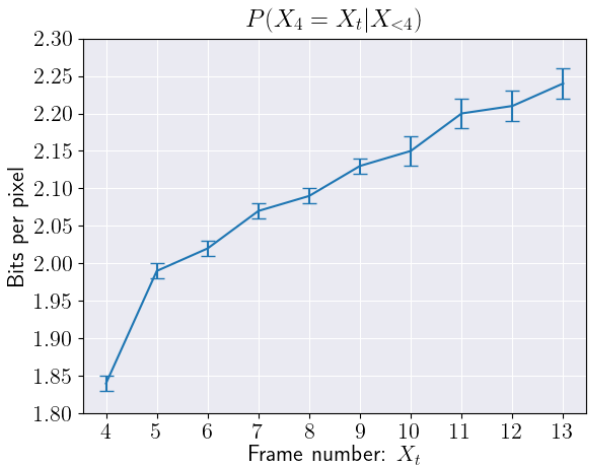

Figure 6: For a given test video, we compute the likelihood of the t^{th} target frame X_t belonging to $\mathcal{P}(X_4 = X_t|X_{<4})$ for $t = 4 \dots 13$ using our model to detect temporal anomalies. We average the corresponding bits-per-pixel across the test-set and plot error bars.

6. Qualitative Experiments

We demonstrate qualitative results by generating videos conditioned on input frames and interpolations in latent space for both datasets. The qualitative results can be viewed at this website. In the generated videos, a border of blue represents the conditioning frame, while a border of red represents the generated frames.

6.1. Effect of temperature

We study the effect of temperature on the quality of generated videos in Figure 7. For each temperature, we sample 100 videos from the model. We then compute the max cosine similarity across these 100 videos based on features obtained from a pretrained VGG network with the ground truth as described in Section 5.4. We display the worst and best videos according to this metric. On inspection, we observe that even our "worst" videos across temperatures according to this metric are temporally cohesive and the robot arm looks sharp and realistic. We believe that though these videos are of high quality and are physically plausible, they are far from the ground truth, which itself represents just one plausible future in VGG feature space.

At lower temperatures, the arm exhibits slow motion with the background objects remaining static and clear while at higher temperatures, the arm moves much more rapidly, with the background objects becoming much noisier. We obtain a tradeoff between these two properties at a temperature of 0.5 via our qualitative experiments.

6.2. Longer predictions

We generate 100 frames into the future using our model trained on 13 frames with a temperature of 0.5. We display our results in Figure 8. On the top, even 100 frames into the future, the generated frames remain in the image manifold maintaining temporal consistency.

We additionally display a failure mode on the bottom. In the presence of occlusions, the arm remains super-sharp but the background objects become noisier and blurrier. We hypothesize that this can be due to following reason. Our VideoFlow model has a bijection between the $\mathbf{z}_t$ and $\mathbf{x}_t$ meaning that the latent state $\mathbf{z}_t$ cannot store information other than that present in the frame $\mathbf{x}_t$. This, in combination with the Markovian assumption in our latent dynamics means that the model can forget objects if they have been occluded for a few frames. In future work, we would address this drawback by incorporating longer memory in our VideoFlow model; for example by parameterizing $NN_{\boldsymbol{\theta}}()$

[4]An important caveat is that code and hardware differences make these numbers not directly comparable.

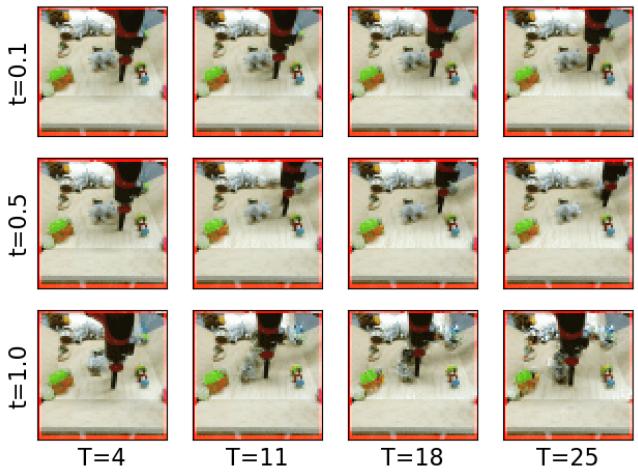

Figure 7: We generate videos with temperatures 0.1, 0.5 and 1.0. For each temperature, we display generated frames at different time-steps into the future.

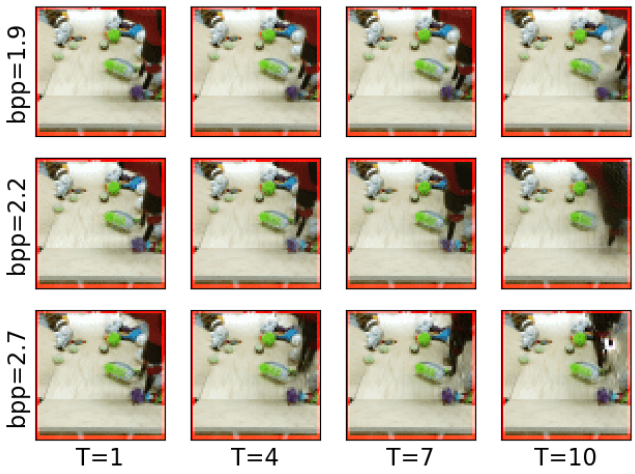

Figure 9: We provide a comparison between training progression (measured in the mean bits-per-pixel objective on the test-set) and the quality of generated videos.

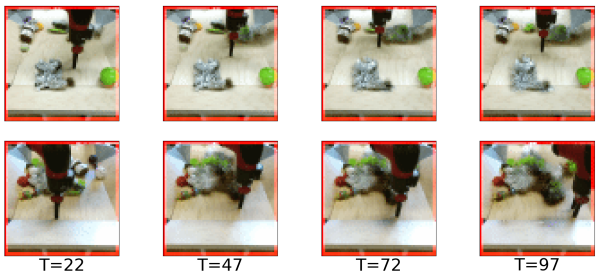

Figure 8: We generate 100 frames into the future with a temperature of 0.5. The top and bottom row correspond to generated videos in the absence and presence of occlusions respectively.

as a recurrent neural networks instead of residual networks in our autoregressive prior (eq. 9). Training on larger temporal patches could also potentially be made feasible by using more memory-efficient backpropagation algorithms for invertible neural networks, as initially explored by (Gomez et al., 2017).

6.3. Likelihood vs Quality

We show correlation between training progression (measured in bits per pixel) and quality of the generated videos in Figure 9. We display the videos generated by conditioning on frames from the test set for three different values of bits-per-pixel on the test-set. As we approach lower bits-per-pixel, our VideoFlow model learns to model the structure of the arm with high quality as well as its motion resulting in high quality video.

6.4. Latent space interpolation

BAIR robot pushing dataset: We encode the first input frame and the last target frame into the latent space using our trained VideoFlow encoder and perform interpolations. We find that the motion of the arm is interpolated in a temporally cohesive fashion between the initial and final position. Further, we use the multi-level latent representation to interpolate representations at a particular level while keeping the representations at other levels fixed. We find that the bottom level interpolates the motion of background objects which are at a smaller scale while the top level interpolates the arm motion.

Stochastic Movement Dataset: We encode two different shapes with their type fixed but a different size and color into the latent space. We observe that the size of the shape gets smoothly interpolated. During training, we sample the colors of the shapes from a uniform discrete distribution which is reflected in our experiments. We observe that all the colors in the interpolated space lie in the set of colors in the training set.

7. Code for reproducing results

Our code to reproduce the experimental results is available in the publicly available Tensor2Tensor repository

8. Conclusion and Discussion

We describe a practically applicable architecture for flow-based video prediction models, inspired by the Glow model for image generation (Kingma & Dhariwal, 2018), which we call VideoFlow. We introduce a latent dynamical sys-

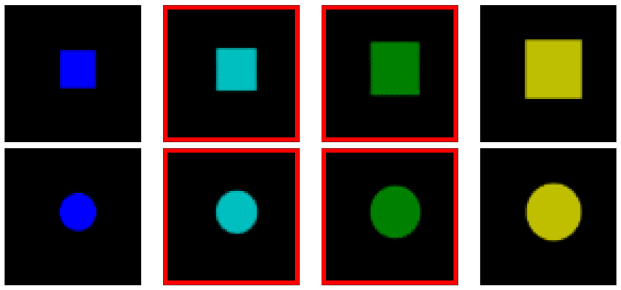

Figure 10: We display interpolations between a) a small blue rectangle and a large yellow rectangle b) a small blue circle and a large yellow circle

Figure 11: We display interpolations between the first input frame and the last target frame of two test videos in the BAIR robot pushing dataset.

tem model that predicts future values of the flow model's latent state replacing the standard unconditional prior distribution. Our empirical results show that VideoFlow achieves results that are competitive with the state-of-the-art VAE models in stochastic video prediction. Finally, our model optimizes log-likelihood directly making it easy to evaluate while achieving faster synthesis compared to pixel-level autoregressive video models, making our model suitable for practical purposes. In future work, we plan to incorporate memory in VideoFlow to model arbitrary long-range dependencies and apply the model to challenging downstream tasks.

References

Babaeizadeh, M., Finn, C., Erhan, D., Campbell, R. H., and Levine, S. Stochastic variational video prediction. *arXiv preprint arXiv:1710.11252*, 2017.

Bengio, S., Vinyals, O., Jaitly, N., and Shazeer, N. Scheduled sampling for sequence prediction with recurrent neural networks. In *Advances in Neural Information Processing Systems*, pp. 1171–1179, 2015.

Boots, B., Byravan, A., and Fox, D. Learning predictive models of a depth camera & manipulator from raw execution traces. In *International Conference on Robotics and Automation (ICRA)*, 2014.

De Brabandere, B., Jia, X., Tuytelaars, T., and Van Gool, L. Dynamic filter networks. In *Neural Information Processing Systems (NIPS)*, 2016.

Deco, G. and Brauer, W. Higher order statistical decorrelation without information loss. *Advances in Neural Information Processing Systems*, pp. 247–254, 1995.

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

Dinh, L., Krueger, D., and Bengio, Y. Nice: non-linear independent components estimation. *arXiv preprint arXiv:1410.8516*, 2014.

Dinh, L., Sohl-Dickstein, J., and Bengio, S. Density estimation using Real NVP. *arXiv preprint arXiv:1605.08803*, 2016.

Dosovitskiy, A. and Brox, T. Generating images with perceptual similarity metrics based on deep networks. In *Advances in Neural Information Processing Systems*, pp. 658–666, 2016.

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

Finn, C. and Levine, S. Deep visual foresight for planning robot motion. In *International Conference on Robotics and Automation (ICRA)*, 2017.

Finn, C., Goodfellow, I., and Levine, S. Unsupervised learning for physical interaction through video prediction. In *Advances in Neural Information Processing Systems*, 2016.

Gomez, A. N., Ren, M., Urtasun, R., and Grosse, R. B. The reversible residual network: Backpropagation without storing activations. In *Advances in Neural Information Processing Systems*, pp. 2211–2221, 2017.

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. Generative adversarial nets. In *Advances in Neural Information Processing Systems*, pp. 2672–2680, 2014.

Graves, A. Generating sequences with recurrent neural networks. *arXiv preprint arXiv:1308.0850*, 2013.

He, K., Zhang, X., Ren, S., and Sun, J. Deep residual learning for image recognition. *arXiv preprint arXiv:1512.03385*, 2015.

Hochreiter, S. and Schmidhuber, J. Long Short-Term Memory. *Neural computation*, 9(8):1735–1780, 1997.

Johnson, J., Alahi, A., and Fei-Fei, L. Perceptual losses for real-time style transfer and super-resolution. In *European Conference on Computer Vision*, pp. 694–711. Springer, 2016.

Kalchbrenner, N., van den Oord, A., Simonyan, K., Danihelka, I., Vinyals, O., Graves, A., and Kavukcuoglu, K. Video pixel networks. *International Conference on Machine Learning (ICML)*, 2017.

Kingma, D. P. and Dhariwal, P. Glow: Generative flow with invertible 1x1 convolutions. In *Advances in Neural Information Processing Systems*, pp. 10236–10245, 2018.

Kingma, D. P. and Welling, M. Auto-encoding variational Bayes. *Proceedings of the 2nd International Conference on Learning Representations*, 2013.

Kingma, D. P., Salimans, T., Jozefowicz, R., Chen, X., Sutskever, I., and Welling, M. Improved variational inference with inverse autoregressive flow. In *Advances in Neural Information Processing Systems*, pp. 4743–4751, 2016.

Krizhevsky, A., Sutskever, I., and Hinton, G. Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems 25*, pp. 1106–1114, 2012.

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

Liu, Z., Yeh, R., Tang, X., Liu, Y., and Agarwala, A. Video frame synthesis using deep voxel flow. *International Conference on Computer Vision (ICCV)*, 2017.

Lotter, W., Kreiman, G., and Cox, D. Deep predictive coding networks for video prediction and unsupervised learning. *International Conference on Learning Representations (ICLR)*, 2017.

Mathieu, M., Couprie, C., and LeCun, Y. Deep multi-scale video prediction beyond mean square error. *International Conference on Learning Representations (ICLR)*, 2016.

Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D., and Riedmiller, M. Playing Atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*, 2013.

Prenger, R., Valle, R., and Catanzaro, B. Waveglow: A flow-based generative network for speech synthesis. *CoRR*, abs/1811.00002, 2018. URL http://arxiv.org/abs/1811.00002.

Ramachandran, P., Paine, T. L., Khorrami, P., Babaeizadeh, M., Chang, S., Zhang, Y., Hasegawa-Johnson, M. A., Campbell, R. H., and Huang, T. S. Fast generation for convolutional autoregressive models. *arXiv preprint arXiv:1704.06001*, 2017.

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

Reed, S., Oord, A. v. d., Kalchbrenner, N., Colmenarejo, S. G., Wang, Z., Belov, D., and de Freitas, N. Parallel multiscale autoregressive density estimation. *arXiv preprint arXiv:1703.03664*, 2017.

Rezende, D. and Mohamed, S. Variational inference with normalizing flows. In *Proceedings of The 32nd International Conference on Machine Learning*, pp. 1530–1538, 2015.

Rezende, D. J., Mohamed, S., and Wierstra, D. Stochastic backpropagation and approximate inference in deep generative models. In *Proceedings of the 31st International Conference on Machine Learning (ICML-14)*, pp. 1278–1286, 2014.

Silver, D., Schrittwieser, J., Simonyan, K., Antonoglou, I., Huang, A., Guez, A., Hubert, T., Baker, L., Lai, M., Bolton, A., et al. Mastering the game of go without human knowledge. *Nature*, 550(7676):354, 2017.

Srivastava, N., Mansimov, E., and Salakhudinov, R. Unsupervised learning of video representations using lstms. In *International Conference on Machine Learning*, 2015.

Van Den Oord, A., Dieleman, S., Zen, H., Simonyan, K., Vinyals, O., Graves, A., Kalchbrenner, N., Senior, A., and Kavukcuoglu, K. Wavenet: A generative model for raw audio. *arXiv preprint arXiv:1609.03499*, 2016.

van den Oord, A., Kalchbrenner, N., Espeholt, L., Vinyals, O., Graves, A., et al. Conditional image generation with PixelCNN decoders. In *Advances in Neural Information Processing Systems*, pp. 4790–4798, 2016a.

van den Oord, A., Kalchbrenner, N., and Kavukcuoglu, K. Pixel recurrent neural networks. *arXiv preprint arXiv:1601.06759*, 2016b.

van den Oord, A., Kalchbrenner, N., Vinyals, O., Espeholt, L., Graves, A., and Kavukcuoglu, K. Conditional image generation with PixelCNN decoders. *arXiv preprint arXiv:1606.05328*, 2016c.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. Attention is all you need. In *Advances in Neural Information Processing Systems*, pp. 5998–6008, 2017.

Vaswani, A., Bengio, S., Brevdo, E., Chollet, F., Gomez, A. N., Gouws, S., Jones, L., Kaiser, Ł., Kalchbrenner, N., Parmar, N., et al. Tensor2tensor for neural machine translation. *arXiv preprint arXiv:1803.07416*, 2018.

Vondrick, C. and Torralba, A. Generating the future with adversarial transformers. In *Computer Vision and Pattern Recognition (CVPR)*, 2017.

Vondrick, C., Pirsiavash, H., and Torralba, A. Anticipating the future by watching unlabeled video. *arXiv preprint arXiv:1504.08023*, 2015.

Walker, J., Gupta, A., and Hebert, M. Dense optical flow prediction from a static image. In *International Conference on Computer Vision (ICCV)*, 2015.

Wang, Z., Bovik, A. C., Sheikh, H. R., and Simoncelli, E. P. Image quality assessment: from error visibility to structural similarity. *IEEE transactions on image processing*, 2004.

Xingjian, S., Chen, Z., Wang, H., Yeung, D.-Y., Wong, W.-K., and Woo, W.-c. Convolutional lstm network: A machine learning approach for precipitation nowcasting. In *Advances in Neural Information Processing Systems*, 2015.

Zhang, R., Isola, P., Efros, A. A., Shechtman, E., and Wang, O. The unreasonable effectiveness of deep features as a perceptual metric. *arXiv preprint*, 2018.

A. VideoFlow - BAIR Hyperparameters

A.1. Quantitative - Bits-per-pixel

To report bits-per-pixel we use the following set of hyperparameters. We use a learning rate schedule of linear warmup for the first 10000 steps and apply a linear-decay schedule for the last 150000 steps.

Hyperparameter	Value
Flow levels	3
Flow steps per level	24
Coupling	Affine
Number of coupling layer channels	512
Optimier	Adam
Batch size	40
Learning rate	3e-4
Number of 3-D residual blocks	5
Number of 3-D residual channels	256
Training steps	600K

A.2. Qualitative Experiments

For all qualitative experiments and quantitative comparisons with the baselines, we used the following sets of hyperparameters.

Hyperparameter	Value
Flow levels	3
Flow steps per level	24
Coupling	Additive
Number of coupling layer channels	392
Optimier	Adam
Batch size	40
Learning rate	3e-4
Number of 3-D residual blocks	5
Number of 3-D residual channels	256
Training steps	500K

B. Hyperparameter grid for the baseline video models.

We train all our baseline models for 300K steps using the Adam optimizer. Our models were tuned using the maximum VGG cosine similarity metric with the ground-truth across 100 decodes.

SAVP-VAE and SV2P: We use three values of latent loss multiplier 1e-3, 1e-4 and 1e-5. For the SAVP-VAE model, we additionally apply linear decay on the learning rate for the last 100K steps.
SAVP-GAN: We tune the gan loss multiplier and the learning rate on a logscale from 1e-2 to 1e-4 and 1e-3 to 1e-5 respectively.

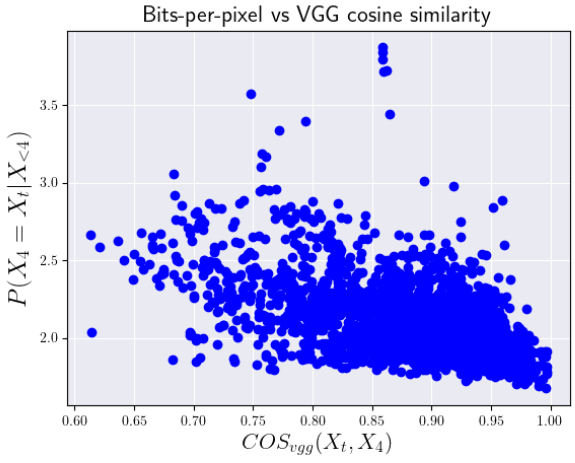

Figure 12: We compare $\mathcal{P}(X_4 = X_t|X_{<4})$ and VGG cosine similarity between X_4 and X_t for $t = 4 \dots 13$

C. Correlation between VGG perceptual similarity and bits-per-pixel

We plot correlation between cosine similarity using a pre-trained VGG network and bits-per-pixel using our trained VideoFlow model. We compare $\mathcal{P}(X_4 = X_t|X_{<4})$ as done in Section 5.6 and the VGG cosine similarity between X_4 and X_t for $t = 4 \dots 13$. We report our results for every video in the test set in Figure 6. We notice a weak correlation between VGG perceptual metrics and bits-per-pixel with a correlation factor of -0.51.