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. 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. To our knowledge, our work is the first to propose multi-frame video prediction with normalizing flows, which allows for direct optimization of the data likelihood, and produces high-quality stochastic predictions. 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. Related Work

In this paper, we study the problem of stochastic conditional video prediction: synthesizing raw RGB video frames conditioned on a short context of past observations. 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 predicting high-level structure (Villegas et al., 2017b), incorporating pixel transformations (Finn et al., 2016; De Brabandere et al., 2016; Liu et al., 2017), predictive coding architectures (Lotter et al., 2017), as well as different generation objectives (Mathieu et al., 2016; Vondrick & Torralba, 2017; Walker et al., 2015) and disentangling representations (Villegas et al., 2017a; Denton & Birodkar, 2017). 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. In such cases, deterministic 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). The models based on VAEs (Babaeizadeh et al., 2017; Denton & Fergus, 2018; Lee et al., 2018; Xue et al., 2016; Li et al., 2018) do not maximize the log-likelihood directly, since they rely on optimizing the evidence lower bound while the autoregressive models (Hochreiter & Schmidhuber, 1997; Graves, 2013; van den Oord et al., 2016b;c; Van Den Oord et al., 2016; Reed et al., 2017; Ramachandran et al., 2017) are computationally expensive during generation.

2. Contribution

Our main contribution is a practically applicable architecture for flow-based video prediction models which we call VideoFlow. To our knowledge, prior work has only applied such models to generate static images (Kingma & Dhariwal, 2018) or sound (Prenger et al., 2018). In this work, we learn a latent dynamical system model that predicts future values of the flow model's latent state. Our empirical results show that VideoFlow can produce excellent qualitative predictions with quantitative results that are competitive with the best VAE techniques in stochastic video prediction on the action-free BAIR dataset. Our model further directly optimizes the likelihood of training videos, allowing us to evaluate its performance directly in terms of likelihood values, performs exact inference and is computationally efficient during inference and generation.[1].

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

First workshop on *Invertible Neural Networks and Normalizing Flows* (ICML 2019), Long Beach, CA, USA

[1]We generate 64x64 videos of 20 frames in less than 3.5 seconds on a NVIDIA P100 GPU as compared to (Reed et al., 2017)

3. Preliminaries: Flow-Based Generative Models

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, 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 $\mathbf{f}_i$. As explored in prior work (Deco & Brauer, 1995; Dinh et al., 2014; 2016; Rezende & Mohamed, 2015; Kingma et al., 2016; Kingma & Dhariwal, 2018), we choose transformations whose Jacobian $d\mathbf{h}_i/d\mathbf{h}_{i-1}$ is a triangular matrix, diagonal matrix or a permutation matrix for computational efficiency. 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. VideoFlow - Autoregressive latent dynamics model

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: 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.

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 1 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). We have described the architecture with ablation studies in the appendix with network figures at the website.

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 also experimented with 3-D convolutional flows, which we found to be computationally expensive in comparison to our architecture; In addition, due to memory limits, we found it only feasible to perform SGD with a small number of sequential frames per gradient step. In the case of a 3-D flow, this would also change the distribution between training and synthesis, leading to temporal artifacts.

5. Experiments

The code to reproduce our quantitative results along with the full set of hyperparameters is available in the Tensor2Tensor repository. Our 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.

which generate a frame every 3 seconds

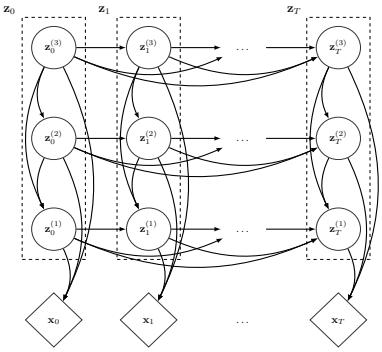

Figure 1: The input at each timestep $\mathbf{x}_t$ is encoded into multiple levels of stochastic variables $(\mathbf{z}_t^{(1)}, \ldots, \mathbf{z}_t^{(L)})$. We model those levels through a sequential process $\prod_t \prod_l p(\mathbf{z}_t^{(l)} \mid \mathbf{z}_{<t}^{(l)}, z_t^{(>l)})$.

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 moves in one of eight random directions. 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 its position at the t^{th} step. At $t = 2$, the position of the shape is probabilistic and for any other t the position is deterministic given previous timesteps. 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. 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. On generation of videos conditioned on the first frame with the shape at the center, we observe that the model consistently predicts the future trajectory of the shape to be one of 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. For each video we extract the first 13 frames to be compatible with previous work 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 described in the appendix 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 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. We report a value of **1.87 bpp** on the test set.

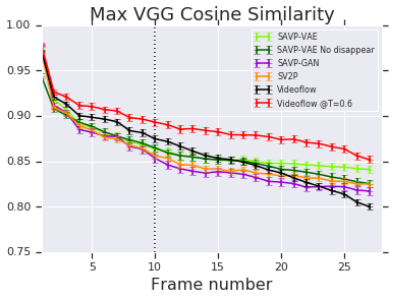

Figure 2: For a given set of conditioning frames on the BAIR action-free we sample 100 videos from each of the stochastic video generation models. We choose the video closest to the ground-truth on the basis of VGG perceptual metrics that correlate with human perception **higher is better** (Zhang et al., 2018; Johnson et al., 2016). All the models were trained using ten target frames but are tested to generate 27 frames.

5.3. 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) using their open-source implementations (Vaswani et al., 2018). 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 are intractable. To make a quantitative comparison with the baselines, we follow the metrics proposed in prior work (Babaeizadeh et al., 2017; Lee et al., 2018). We describe the metric and report our findings in Figure 2. In prior work, (Lee et al., 2018) and (Babaeizadeh et al., 2017) remove pixel-level noise by using a deterministic decoder and tuning this as a hyperparameter. In VideoFlow, we can remove pixel-level noise at the cost of diversity by sampling videos at a lower temperature (Kingma & Dhariwal, 2018), the procedure of which, we describe in the appendix. We report results with a temperature of 1.0 and the optimal temperature in Figure 2. Our model with optimal temperature outperforms the SAVP-VAE model while the model at temperature $T = 1.0$ is

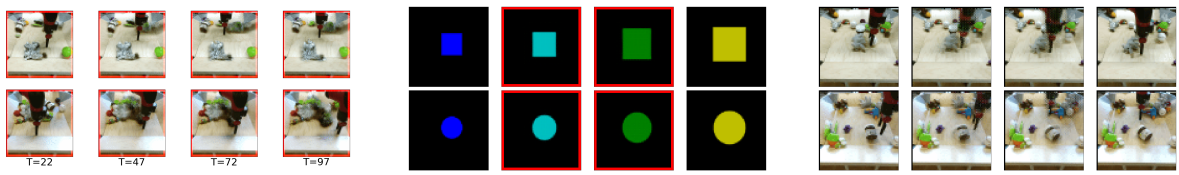

Figure 3: **Left:** The top and bottom row correspond to generated videos in the absence and presence of occlusions respectively. **Center**: Latent space interpolations on the Stochastic Movement Dataset. **Right**: Latent space interpolations on the BAIR robot pushing dataset

competent with other state-of-the-art models.

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; 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 have a true reconstruction loss due to scheduled sampling and they do not optimize the log-likelihood directly due to the presence of a $\beta \neq 1$ term in their objective.

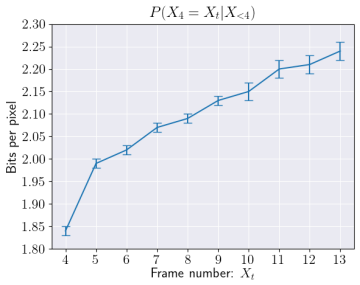

Figure 4: For a given test video, we compute the likelihood of the t^{th} target frame X_t belonging to $\mathcal{P}(X_4 = X_t|X_{<4})$ for $t = 4 \dots 13$ using our model to detect temporal anomalies. We average the corresponding bits-per-pixel across the test-set and plot error bars.

5.4. 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 4. 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.

5.5. Effect of temperature

We study the effect of temperature on the quality of generated videos. At lower temperatures, the arm exhibits slow motion with lower uncertainty and the background objects remaining static. At higher temperatures, the arm exhibits stochastic motion with more uncertainty, with the background objects becoming much noisier.

5.6. 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 3. Even 100 frames into the future, the generated frames remain in the image manifold maintaining temporal consistency. In the presence of occlusions, the arm remains super-sharp but the background objects become distorted. In future work, we would address this drawback by incorporating longer memory by using memory-efficient backpropagation algorithms for invertible networks (Gomez et al., 2017) or parameterizing $NN_{\boldsymbol{\theta}}()$ as a recurrent neural network instead of residual networks in our autoregressive prior (eq. 9).

5.7. 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 temporally-cohseive interpolations. On interpolating specific levels, 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, the colors of the shapes are discretized 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.

References

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

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 Birodkar, V. Unsupervised learning of disentangled representations from video. *arXiv preprint arXiv:1705.10915*, 2017.

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.

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., 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.

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.

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

Li, Y., Fang, C., Yang, J., Wang, Z., Lu, X., and Yang, M.-H. Flow-grounded spatial-temporal video prediction from still images. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 600–615, 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.

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.

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., 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.

Villegas, R., Yang, J., Hong, S., Lin, X., and Lee, H. Decomposing motion and content for natural video sequence prediction. *arXiv preprint arXiv:1706.08033*, 2017a.

Villegas, R., Yang, J., Zou, Y., Sohn, S., Lin, X., and Lee, H. Learning to generate long-term future via hierarchical prediction. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pp. 3560–3569. JMLR. org, 2017b.

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.

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.

Xue, T., Wu, J., Bouman, K., and Freeman, B. Visual dynamics: Probabilistic future frame synthesis via cross convolutional networks. In *Advances in Neural Information Processing Systems*, 2016.

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

B. 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 **??**), 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. 5) 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 5. 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.

C. Temperature

We can remove pixel-level noise in our VideoFlow model resulting in higher quality videos at the cost of diversity. 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[2].

D. Likelihood vs Quality

We show correlation between training progression (measured in bits per pixel) and quality of the generated videos in Figure 6. 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.

E. VideoFlow - BAIR Hyperparameters

E.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

E.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

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

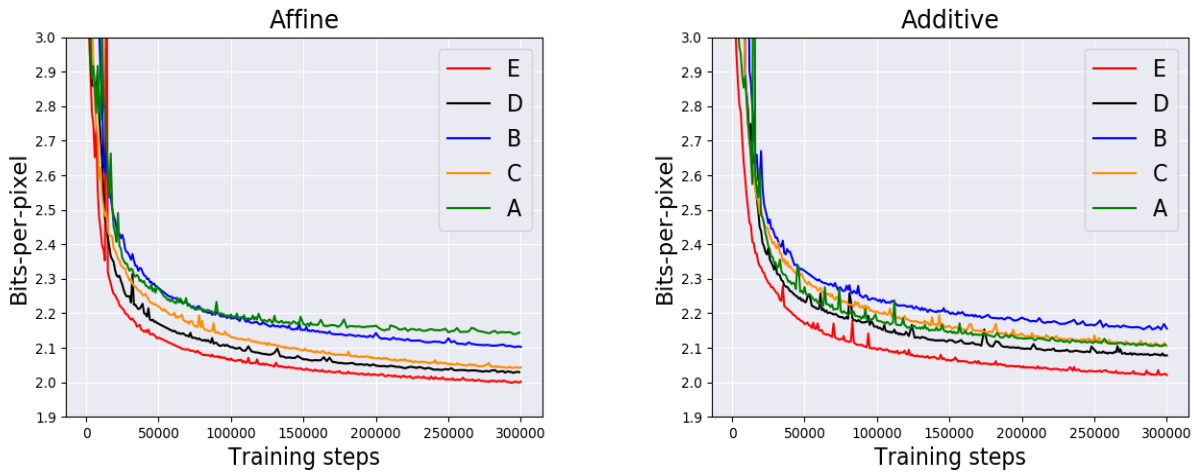

Figure 5: **B: baseline**, **A: Temporal Skip Connection**, **C: Dilated Convolutions + GATU**, **D: Dilation Convolutions + Temporal Skip Connection**, **E: Dilation Convolutions + Temporal Skip Connection + GATU**. We plot the holdout bits-per-pixel on the BAIR action-free dataset for different ablations of our VideoFlow model.

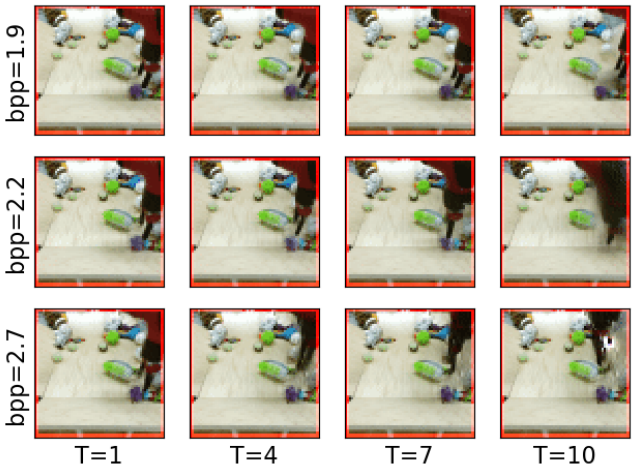

Figure 6: We provide a comparison between training progression (measured in the mean bits-per-pixel objective on the test-set) and the quality of generated videos.

F. 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.

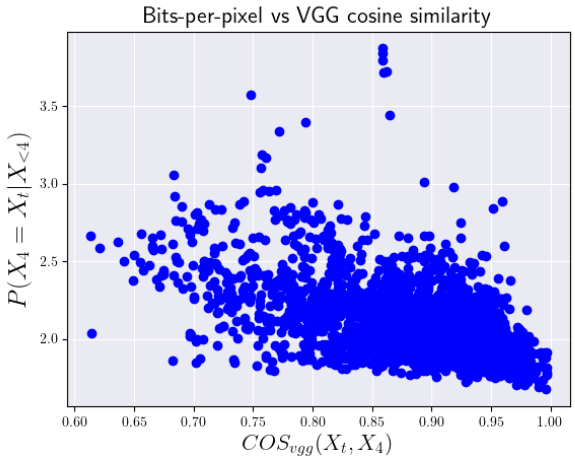

Figure 7: We compare $\mathcal{P}(X_4 = X_t|X_{<4})$ and VGG cosine similarity between X_4 and X_t for $t = 4 \dots 13$

G. 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 5.4 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 4. We notice a weak correlation between VGG perceptual metrics and bits-per-pixel

with a correlation factor of -0.51.