2nd Symposium on Advances in Approximate Bayesian Inference, 2019 1–16

Improving Sequential Latent Variable Models with Autoregressive Flows

Joseph Marino JMARINO@CALTECH.EDU
California Institute of Technology

Lei Chen and **Jiawei He** {LEI_CHEN_4, JIAWEI_HE_2}@SFU.CA
Simon Fraser University

Stephan Mandt MANDT@UCI.EDU
University of California Irvine

Abstract

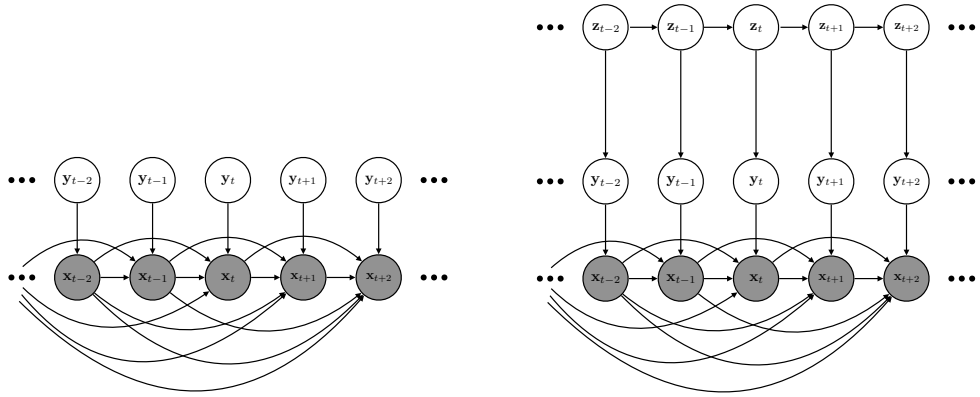
We propose an approach for sequence modeling based on autoregressive normalizing flows. Each autoregressive transform, acting across time, serves as a moving reference frame for modeling higher-level dynamics. This technique provides a simple, general-purpose method for improving sequence modeling, with connections to existing and classical techniques. We demonstrate the proposed approach both with standalone models, as well as a part of larger sequential latent variable models. Results are presented on three benchmark video datasets, where flow-based dynamics improve log-likelihood performance over baseline models.

1. Introduction

Sequential structure in data provides a rich learning signal. Recent improvements in computational techniques, primarily deep networks, have facilitated learning sequential models from high-dimensional data (Graves, 2013; Chung et al., 2015), particularly video and audio. However, such models attempt to capture all sequential dependencies with relatively unstructured dynamics. Intuitively, the model should use its dynamical components to track *changes* in the input instead of simultaneously modeling the entire signal. Rather than expanding the computational capacity of the model, we seek a method for altering the representation of the data to provide a more structured form of dynamics.

To incorporate more structured dynamics, we propose an approach for sequence modeling based on autoregressive normalizing flows (Kingma et al., 2016; Papamakarios et al., 2017), consisting of one or more autoregressive transforms in time. A single transform is equivalent to a Gaussian autoregressive model, and stacking additional transforms or latent variables on top results in more expressive models. Each autoregressive transform serves as a *moving reference frame* in which higher-level structure is modeled. This provides a general mechanism for separating different forms of dynamics, with higher-level stochastic dynamics modeled in the simplified space provided by lower-level deterministic transforms. This approach generalizes the classical technique of modeling temporal derivatives to simplify dynamics estimation (Friston, 2008) (Appendix B).

We empirically demonstrate this approach, both with standalone autoregressive normalizing flows, as well as by incorporating these flows within more flexible sequential latent



(a) (b)

Figure 1: **Graphical Models**. Diagrams for **(a)** a single-transform affine autoregressive flow-based model, with random variables, $\mathbf{y}_{1:T} \sim \mathcal{N}(\mathbf{y}_{1:T}; \mathbf{0}, \mathbf{I})$, and **(b)** a sequential latent variable model with a flow-based conditional likelihood.

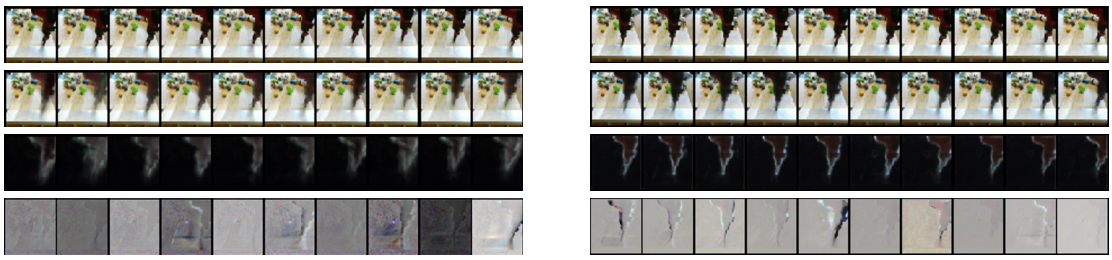
variable models. While normalizing flows have been applied in a few sequential contexts previously, we emphasize the use of these models in conjunction with sequential latent variable models. We present experimental results on three benchmark video datasets, showing improved quantitative performance in terms of log-likelihood.

2. Background

Consider modeling discrete sequences of observations, $\mathbf{x}_{1:T} \sim p_{\text{data}}(\mathbf{x}_{1:T})$, using a probabilistic model, $p_\theta(\mathbf{x}_{1:T})$, with parameters θ. Autoregressive models (Frey et al., 1996; Bengio and Bengio, 2000) use the chain rule of probability: $p_\theta(\mathbf{x}_{1:T}) = \prod_{t=1}^{T} p_\theta(\mathbf{x}_t|\mathbf{x}_{<t})$. Each conditional distribution, $p_\theta(\mathbf{x}_t|\mathbf{x}_{<t})$, models the temporal dependence between time steps. We often assume that each distribution takes a relatively simple form, such as a diagonal Gaussian density: $p_\theta(\mathbf{x}_t|\mathbf{x}_{<t}) = \mathcal{N}(\mathbf{x}_t; \boldsymbol{\mu}_\theta(\mathbf{x}_{<t}), \text{diag}(\boldsymbol{\sigma}^2_\theta(\mathbf{x}_{<t})))$, where $\boldsymbol{\mu}_\theta(\cdot)$ and $\boldsymbol{\sigma}_\theta(\cdot)$ are functions for the mean and standard deviation, often sharing parameters over time steps.

Autoregressive models can be improved by incorporating latent variables, often represented as a corresponding sequence, $\mathbf{z}_{1:T}$, yielding a sequential latent variable model. The joint distribution, $p_\theta(\mathbf{x}_{1:T}, \mathbf{z}_{1:T})$, has the following form: $\prod_{t=1}^{T} p_\theta(\mathbf{x}_t|\mathbf{x}_{<t}, \mathbf{z}_{\leq t}) p_\theta(\mathbf{z}_t|\mathbf{x}_{<t}, \mathbf{z}_{<t})$. Evaluating $p_\theta(\mathbf{x}_t|\mathbf{x}_{<t})$ now requires integrating over the latent variables, yielding a more flexible distribution. Such models have demonstrated success in audio (Chung et al., 2015) and video modeling (He et al., 2018). However, design choices for these models remain an active area of research, with each model proposing new combinations of deterministic and stochastic dynamics.

Our approach is based on affine autoregressive normalizing flows (Kingma et al., 2016; Papamakarios et al., 2017). Here, we briefly discuss this concept, continuing with the perspective of temporal sequences. Kingma et al. (2016) noted that sampling from an autoregressive Gaussian model is an invertible transform, resulting in a *normalizing flow* (Rezende and Mohamed, 2015). Flow-based models transform between simple, $p_\theta(\mathbf{y}_{1:T})$, and



(a) (b)

Figure 2: **Flow Visualization**. Visualization of the flow component for a **(a)** standalone flow-based model and **(b)** sequential latent variable model with flow-based conditional likelihood for BAIR Robot Pushing. From top to bottom, each figure shows 1) the original frames, $\mathbf{x}_t$, 2) the predicted shift, $\boldsymbol{\mu}_\theta(\mathbf{x}_{<t})$, for the frame, 3) the predicted scale, $\boldsymbol{\sigma}_\theta(\mathbf{x}_{<t})$, for the frame, and 4) the noise, $\mathbf{y}_t$, obtained from the inverse transform.

complex, $p_\theta(\mathbf{x}_{1:T})$, probability distributions while maintaining exact likelihood evaluation (Dinh et al., 2015). In the case of affine autoregressive models, this transform is given by $\mathbf{x}_t = \boldsymbol{\mu}_\theta(\mathbf{x}_{<t}) + \boldsymbol{\sigma}_\theta(\mathbf{x}_{<t}) \odot \mathbf{y}_t$, where $\odot$ denotes element-wise multiplication. We can improve upon this simple set-up by chaining multiple transforms together. We provide a more in-depth discussion of autoregressive flows and related work in Appendix A.

3. Method: Sequence Modeling with Autoregressive Flows

We now describe our approach for sequence modeling with autoregressive flows, showing how this simple technique can be incorporated within sequential latent variable models to improve dynamics modeling. We offer a motivating example in Appendix B, discussing how affine transforms can temporally decorrelate sequences, simplifying dynamics estimation.

We apply autoregressive flows across time steps within a sequence, $\mathbf{x}_{1:T} \in \mathbb{R}^{T \times D}$. That is, the observation at each time step, $\mathbf{x}_t \in \mathbb{R}^D$, is modeled as an autoregressive function of past observations, $\mathbf{x}_{<t} \in \mathbb{R}^{t-1 \times D}$, and a random variable, $\mathbf{y}_t \in \mathbb{R}^D$ (Fig. 1a). We consider flows of the form $\mathbf{x}_t = \boldsymbol{\mu}_\theta(\mathbf{x}_{<t}) + \boldsymbol{\sigma}_\theta(\mathbf{x}_{<t}) \odot \mathbf{y}_t$, where $\boldsymbol{\mu}_\theta(\mathbf{x}_{<t})$ and $\boldsymbol{\sigma}_\theta(\mathbf{x}_{<t})$ are parameterized by neural networks. The inverse transform is thus $\mathbf{y}_t = (\mathbf{x}_t - \boldsymbol{\mu}_\theta(\mathbf{x}_{<t}))/\boldsymbol{\sigma}_\theta(\mathbf{x}_{<t})$. We can also consider chains of multiple transforms (Appendix A). We can parameterize the base distribution, $p_\theta(\mathbf{y}_{1:T})$, with a parametric distribution or a separate model.

Consider parameterizing the conditional likelihood, $p_\theta(\mathbf{x}_t|\mathbf{x}_{<t}, \mathbf{z}_{\leq t})$, within a latent variable model using an autoregressive flow (Figure 1b). To do so, we express a base conditional distribution for $\mathbf{y}_t$, denoted as $p_\theta(\mathbf{y}_t|\mathbf{y}_{<t}, \mathbf{z}_{\leq t})$, which is then transformed into $\mathbf{x}_t$ via the affine transform. Using the change of variables formula, we can express the latent variable model's log-joint distribution as $\log p_\theta(\mathbf{x}_{1:T}, \mathbf{z}_{1:T}) = \log p_\theta(\mathbf{y}_{1:T}, \mathbf{z}_{1:T}) - \log \left| \det \left(\frac{\partial \mathbf{x}_{1:T}}{\partial \mathbf{y}_{1:T}} \right) \right|$, where the joint distribution over $\mathbf{y}_{1:T}$ and $\mathbf{z}_{1:T}$, in general, is given as $p_\theta(\mathbf{y}_{1:T}, \mathbf{z}_{1:T}) = \prod_{t=1}^T p_\theta(\mathbf{y}_t|\mathbf{y}_{<t}, \mathbf{z}_{\leq t}) p_\theta(\mathbf{z}_t|\mathbf{y}_{<t}, \mathbf{z}_{<t})$. The latent prior, $p_\theta(\mathbf{z}_t|\mathbf{y}_{<t}, \mathbf{z}_{<t})$, can be equivalently conditioned on $\mathbf{x}_{<t}$ or $\mathbf{y}_{<t}$, as there is a one-to-one mapping between these variables.

We can train the sequential latent variable model using variational inference (Jordan et al., 1998), which introduces an approximate posterior distribution, $q(\mathbf{z}_{1:T}|\mathbf{x}_{1:T})$,

Table 1: **Quantitative Comparison.** Average test log-likelihood (higher is better) in *nats per pixel per channel* for Moving MNIST, BAIR Robot Pushing, and KTH Actions. For flow-based models (1-AF and 2-AF), we report the average log-likelihood. For sequential latent variable models (SLVM and SLVM w/ 1-AF), we report the average lower bound on the log-likelihood.

	M-MNIST	BAIR	KTH
1-AF	-2.15	-3.05	-3.34
2-AF	-2.13	-2.90	-3.35
SLVM	≥ -1.92	≥ -3.57	≥ -4.63
SLVM w/ 1-AF	$\geq \mathbf{-1.86}$	$\geq \mathbf{-2.35}$	$\geq \mathbf{-2.39}$

providing a lower bound on the marginal log-likelihood: $\log p_\theta(\mathbf{x}_{1:T}) \geq \mathcal{L}(\mathbf{x}_{1:T}; q, \theta) \equiv \mathbb{E}_{q(\mathbf{z}_{1:T}|\mathbf{x}_{1:T})}\left[\log p_\theta(\mathbf{x}_{1:T}, \mathbf{z}_{1:T}) - \log q(\mathbf{z}_{1:T}|\mathbf{x}_{1:T})\right]$. We consider the case of filtering inference, with $q(\mathbf{z}_{1:T}|\mathbf{x}_{1:T}) = \prod_{t=1}^{T} q(\mathbf{z}_t|\mathbf{x}_{\leq t}, \mathbf{z}_{<t})$. Again, we can condition q on $\mathbf{x}_{\leq t}$ or $\mathbf{y}_{\leq t}$. We derive the corresponding objective in Appendix C:

$$\mathcal{L} = \sum_{t=1}^{T} \mathbb{E}_{q(\mathbf{z}_{\leq t}|\mathbf{y}_{\leq t})}\left[\log p_\theta(\mathbf{y}_t|\mathbf{y}_{<t}, \mathbf{z}_{\leq t}) - \log \frac{q(\mathbf{z}_t|\mathbf{y}_{\leq t}, \mathbf{z}_{<t})}{p_\theta(\mathbf{z}_t|\mathbf{y}_{<t}, \mathbf{z}_{<t})} - \log \left|\det\left(\frac{\partial \mathbf{x}_t}{\partial \mathbf{y}_t}\right)\right|\right] \tag{1}$$

4. Results

We demonstrate and evaluate the proposed method on three benchmark video datasets: Moving MNIST (Srivastava et al., 2015), KTH Actions (Schuldt et al., 2004), and BAIR Robot Pushing (Ebert et al., 2017). Experiment details can be found in Appendix D, additional results are in Appendix E, and code is available here[1].

To qualitatively understand autoregressive flows on sequences, we visualize each component as an image (Fig. 2 & Fig. 8). The shift parameters (second row) tend to capture the static background, blurring around regions of uncertainty. The scale parameters (third row), on the other hand, tend to focus on regions of higher uncertainty. The resulting noise variables (bottom row) display any remaining structure not modeled by the flow. In comparing standalone flow-based models with flow-based conditional likelihoods in sequential latent variable models, we see that the latter qualitatively contains more structure in $\mathbf{y}$. This is expected, as the noise distribution is more expressive in this case. In Appendix E.1, we quantify the degree of temporal decorrelation performed by flow-based models by evaluating the empirical correlation between frames at successive time steps for both the data, $\mathbf{x}$, and the noise variables, $\mathbf{y}$. In Appendix E.2, we provide additional qualitative results.

Quantitative results for each model class are shown in Table 1. We report the average test log-likelihood in *nats per pixel per channel* for flow-based models and the lower bound on this quantity for sequential latent variable models. Standalone flow-based models perform surprisingly well, even outperforming sequential latent variable models in some cases. Increasing flow depth from 1 to 2 generally results in improved performance. Sequential

1. https://github.com/joelouismarino/sequential_flows

latent variable models with flow-based conditional likelihoods outperform their baseline counterparts with Gaussian conditional likelihoods, despite having *fewer* parameters.

The quantitative results in Table 1 are for a representative sequential latent variable model with a standard convolutional encoder-decoder architecture and fully-connected latent variables. However, many previous works do not evaluate proper lower bounds on log-likelihood, using techniques like down-weighting KL divergences (Denton and Fergus, 2018; Ha and Schmidhuber, 2018; Lee et al., 2018). Indeed, Marino et al. (2018) train SVG (Denton and Fergus, 2018) with a proper lower bound and report a lower bound of -2.86 *nats per pixel* on KTH Actions, on-par with our results. Kumar et al. (2019) report log-likelihood results on BAIR Robot Pushing, obtaining -1.3 *nats per pixel*, substantially higher than our results. However, their model is significantly larger than the models presented here, consisting of 3 levels of latent variables, each containing 24 steps of flows.

5. Conclusion

We have presented a technique for improving sequence modeling based on autoregressive normalizing flows. This technique uses affine transforms to temporally decorrelate sequential data, thereby simplifying the estimation of dynamics. We have drawn connections to classical approaches, which involve modeling temporal derivatives. Finally, we have empirically shown how this technique can improve sequential latent variable models.

References

Yoshua Bengio and Samy Bengio. Modeling high-dimensional discrete data with multi-layer neural networks. In *Advances in Neural Information Processing Systems*, pages 400–406, 2000.

Kurtland Chua, Roberto Calandra, Rowan McAllister, and Sergey Levine. Deep reinforcement learning in a handful of trials using probabilistic dynamics models. In *Advances in Neural Information Processing Systems*, pages 4754–4765, 2018.

Junyoung Chung, Kyle Kastner, Laurent Dinh, Kratarth Goel, Aaron C Courville, and Yoshua Bengio. A recurrent latent variable model for sequential data. In *Advances in neural information processing systems*, pages 2980–2988, 2015.

Marc Peter Deisenroth, Dieter Fox, and Carl Edward Rasmussen. Gaussian processes for data-efficient learning in robotics and control. *IEEE transactions on pattern analysis and machine intelligence*, 37(2):408–423, 2013.

Emily Denton and Rob Fergus. Stochastic video generation with a learned prior. In *International Conference on Machine Learning*, pages 1182–1191, 2018.

Laurent Dinh, David Krueger, and Yoshua Bengio. Nice: Non-linear independent components estimation. In *International Conference on Learning Representations*, 2015.

Laurent Dinh, Jascha Sohl-Dickstein, and Samy Bengio. Density estimation using real nvp. In *International Conference on Learning Representations*, 2017.

Conor Durkan, Artur Bekasov, Iain Murray, and George Papamakarios. Neural spline flows. *arXiv preprint arXiv:1906.04032*, 2019.

Frederik Ebert, Chelsea Finn, Alex X Lee, and Sergey Levine. Self-supervised visual planning with temporal skip connections. In *Conference on Robot Learning*, 2017.

Brendan J Frey, Geoffrey E Hinton, and Peter Dayan. Does the wake-sleep algorithm produce good density estimators? In *Advances in neural information processing systems*, pages 661–667, 1996.

Karl Friston. Hierarchical models in the brain. *PLoS computational biology*, 4(11):e1000211, 2008.

Mevlana Gemici, Chia-Chun Hung, Adam Santoro, Greg Wayne, Shakir Mohamed, Danilo J Rezende, David Amos, and Timothy Lillicrap. Generative temporal models with memory. *arXiv preprint arXiv:1702.04649*, 2017.

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

David Ha and Jürgen Schmidhuber. Recurrent world models facilitate policy evolution. In *Advances in Neural Information Processing Systems*, pages 2450–2462, 2018.

Jiawei He, Andreas Lehrmann, Joseph Marino, Greg Mori, and Leonid Sigal. Probabilistic video generation using holistic attribute control. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 452–467, 2018.

Gustav Eje Henter, Simon Alexanderson, and Jonas Beskow. Moglow: Probabilistic and controllable motion synthesis using normalising flows. *arXiv preprint arXiv:1905.06598*, 2019.

Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

Chin-Wei Huang, David Krueger, Alexandre Lacoste, and Aaron Courville. Neural autoregressive flows. In *International Conference on Machine Learning*, pages 2083–2092, 2018.

Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*, 2015.

Priyank Jaini, Kira A Selby, and Yaoliang Yu. Sum-of-squares polynomial flow. In *International Conference on Machine Learning*, pages 3009–3018, 2019.

Michael I Jordan, Zoubin Ghahramani, Tommi S Jaakkola, and Lawrence K Saul. An introduction to variational methods for graphical models. *NATO ASI SERIES D BEHAVIOURAL AND SOCIAL SCIENCES*, 89:105–162, 1998.

Rudolph Emil Kalman et al. A new approach to linear filtering and prediction problems. *Journal of basic Engineering*, 82(1):35–45, 1960.

Sungwon Kim, Sang-Gil Lee, Jongyoon Song, Jaehyeon Kim, and Sungroh Yoon. Flowavenet: A generative flow for raw audio. In *International Conference on Machine Learning*, pages 3370–3378, 2019.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Diederik P Kingma and Max Welling. Stochastic gradient vb and the variational auto-encoder. In *Proceedings of the International Conference on Learning Representations*, 2014.

Durk P Kingma and Prafulla Dhariwal. Glow: Generative flow with invertible 1x1 convolutions. In *Advances in Neural Information Processing Systems*, pages 10215–10224, 2018.

Durk P Kingma, Tim Salimans, Rafal Jozefowicz, Xi Chen, Ilya Sutskever, and Max Welling. Improved variational inference with inverse autoregressive flow. In *Advances in neural information processing systems*, pages 4743–4751, 2016.

Manoj Kumar, Mohammad Babaeizadeh, Dumitru Erhan, Chelsea Finn, Sergey Levine, Laurent Dinh, and Durk Kingma. Videoflow: A flow-based generative model for video. *arXiv preprint arXiv:1903.01434*, 2019.

Alex X Lee, Richard Zhang, Frederik Ebert, Pieter Abbeel, Chelsea Finn, and Sergey Levine. Stochastic adversarial video prediction. *arXiv preprint arXiv:1804.01523*, 2018.

Joseph Marino, Milan Cvitkovic, and Yisong Yue. A general method for amortizing variational filtering. In *Advances in Neural Information Processing Systems*, pages 7857–7868, 2018.

Junier Oliva, Avinava Dubey, Manzil Zaheer, Barnabas Poczos, Ruslan Salakhutdinov, Eric Xing, and Jeff Schneider. Transformation autoregressive networks. In *International Conference on Machine Learning*, pages 3895–3904, 2018.

George Papamakarios, Theo Pavlakou, and Iain Murray. Masked autoregressive flow for density estimation. In *Advances in Neural Information Processing Systems*, pages 2338–2347, 2017.

Wei Ping, Kainan Peng, and Jitong Chen. Clarinet: Parallel wave generation in end-to-end text-to-speech. In *International Conference on Learning Representations*, 2019.

Ryan Prenger, Rafael Valle, and Bryan Catanzaro. Waveglow: A flow-based generative network for speech synthesis. In *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 3617–3621. IEEE, 2019.

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

Danilo Rezende and Shakir Mohamed. Variational inference with normalizing flows. In *International Conference on Machine Learning*, pages 1530–1538, 2015.

Danilo Jimenez Rezende, Shakir Mohamed, and Daan Wierstra. Stochastic backpropagation and approximate inference in deep generative models. In *Proceedings of the International Conference on Machine Learning*, pages 1278–1286, 2014.

Nicholas Rhinehart, Kris M Kitani, and Paul Vernaza. R2p2: A reparameterized push-forward policy for diverse, precise generative path forecasting. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 772–788, 2018.

Nicholas Rhinehart, Rowan McAllister, Kris Kitani, and Sergey Levine. Precog: Prediction conditioned on goals in visual multi-agent settings. 2019.

Oren Rippel and Ryan Prescott Adams. High-dimensional probability estimation with deep density models. *arXiv preprint arXiv:1302.5125*, 2013.

Christian Schuldt, Ivan Laptev, and Barbara Caputo. Recognizing human actions: a local svm approach. In *International Conference on Pattern Recognition*, 2004.

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

Aaron van den Oord, Yazhe Li, Igor Babuschkin, Karen Simonyan, Oriol Vinyals, Koray Kavukcuoglu, George Driessche, Edward Lockhart, Luis Cobo, Florian Stimberg, et al. Parallel wavenet: Fast high-fidelity speech synthesis. In *International Conference on Machine Learning*, pages 3915–3923, 2018.

Zachary Ziegler and Alexander Rush. Latent normalizing flows for discrete sequences. In *International Conference on Machine Learning*, pages 7673–7682, 2019.

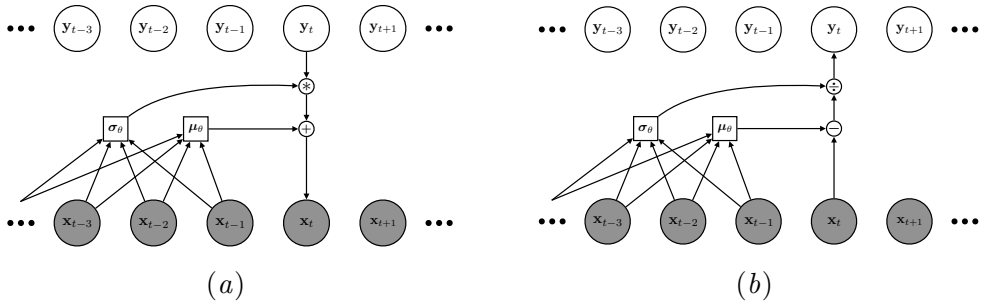


Figure 3: **Affine Autoregressive Transforms**. Computational diagrams for **(a)** forward and **(b)** inverse affine autoregressive transforms (Papamakarios et al., 2017). Each $\mathbf{y}_t$ is an affine transform of $\mathbf{x}_t$, with the affine parameters potentially non-linear functions of $\mathbf{x}_{<t}$. The inverse transform is capable of converting a *correlated* input, $\mathbf{x}_{1:T}$, into a *less correlated* variable, $\mathbf{y}_{1:T}$.

Appendix A. Autoregressive Flows & Related Work

Kingma et al. (2016) noted that sampling from an autoregressive Gaussian model is an invertible transform, resulting in a *normalizing flow* (Rippel and Adams, 2013; Dinh et al., 2015, 2017; Rezende and Mohamed, 2015). Flow-based models transform between simple and complex probability distributions while maintaining exact likelihood evaluation. To see their connection to autoregressive models, we can express sampling a Gaussian random variable, $\mathbf{x}_t \sim p_\theta(\mathbf{x}_t|\mathbf{x}_{<t})$, using the reparameterization trick (Kingma and Welling, 2014; Rezende et al., 2014):

$$\mathbf{x}_t = \boldsymbol{\mu}_\theta(\mathbf{x}_{<t}) + \boldsymbol{\sigma}_\theta(\mathbf{x}_{<t}) \odot \mathbf{y}_t, \tag{2}$$

where $\mathbf{y}_t \sim \mathcal{N}(\mathbf{y}_t; \mathbf{0}, \mathbf{I})$ is an auxiliary random variable and $\odot$ denotes element-wise multiplication. Thus, $\mathbf{x}_t$ is an invertible transform of $\mathbf{y}_t$, with the inverse given as

$$\mathbf{y}_t = \frac{\mathbf{x}_t - \boldsymbol{\mu}_\theta(\mathbf{x}_{<t})}{\boldsymbol{\sigma}_\theta(\mathbf{x}_{<t})}, \tag{3}$$

where division is performed element-wise. The inverse transform in Eq. 3 acts to normalize (hence, *normalizing* flow) and therefore decorrelate $\mathbf{x}_{1:T}$. Given the functional mapping between $\mathbf{y}_t$ and $\mathbf{x}_t$ in Eq. 2, the change of variables formula converts between probabilities in each space:

$$\log p_\theta(\mathbf{x}_{1:T}) = \log p_\theta(\mathbf{y}_{1:T}) - \log \left| \det \left(\frac{\partial \mathbf{x}_{1:T}}{\partial \mathbf{y}_{1:T}} \right) \right|. \tag{4}$$

By the construction of Eqs. 2 and 3, the Jacobian in Eq. 4 is triangular, enabling efficient evaluation as the product of diagonal terms:

$$\log \left| \det \left(\frac{\partial \mathbf{x}_{1:T}}{\partial \mathbf{y}_{1:T}} \right) \right| = \sum_{t=1}^{T} \sum_i \log \sigma_{\theta,i}(\mathbf{x}_{<t}), \tag{5}$$

where i denotes the observation dimension, e.g. pixel. For a Gaussian autoregressive model, $p_\theta(\mathbf{y}_{1:T}) = \mathcal{N}(\mathbf{y}_{1:T}; \mathbf{0}, \mathbf{I})$. With these components, the change of variables formula (Eq. 4) provides an equivalent method for sampling and evaluating the model, $p_\theta(\mathbf{x}_{1:T})$.

We can improve upon this simple set-up by chaining together multiple transforms, effectively resulting in a hierarchical autoregressive model. Letting $\mathbf{y}^m_{1:T}$ denote the variables after the m^{th} transform, the change of variables formula for M transforms is

$$\log p_\theta(\mathbf{x}_{1:T}) = \log p_\theta(\mathbf{y}^M_{1:T}) - \log \left| \det \left(\frac{\partial \mathbf{x}_{1:T}}{\partial \mathbf{y}^1_{1:T}} \right) \right| - \sum_{m=1}^{M-1} \log \left| \det \left(\frac{\partial \mathbf{y}^m_{1:T}}{\partial \mathbf{y}^{m+1}_{1:T}} \right) \right|. \tag{6}$$

Autoregressive flows were initially considered in the contexts of variational inference (Kingma et al., 2016) and generative modeling (Papamakarios et al., 2017). These approaches are, in fact, generalizations of previous approaches with affine transforms (Dinh et al., 2015, 2017). While autoregressive flows are well-suited for sequential data, as mentioned previously, these approaches, as well as many recent approaches (Huang et al., 2018; Oliva et al., 2018; Kingma and Dhariwal, 2018), were initially applied in static settings, such as images.

More recent works have started applying flow-based models to sequential data. For instance, van den Oord et al. (2018) and Ping et al. (2019) *distill* autoregressive speech models into flow-based models. Prenger et al. (2019) and Kim et al. (2019) instead train these models directly. Kumar et al. (2019) use a flow to model individual video frames, with an autoregressive prior modeling dynamics across time steps. Rhinehart et al. (2018) and Rhinehart et al. (2019) use autoregressive flows for modeling vehicle motion, and Henter et al. (2019) use flows for motion synthesis with motion-capture data. Ziegler and Rush (2019) learn distributions over sequences of discrete observations (e.g., text) by using flows to model dynamics of continuous latent variables. Like these recent works, we apply flow-based models to sequential data. However, we demonstrate that autoregressive flows can serve as a useful, general-purpose technique for improving sequence modeling as components of sequential latent variable models. To the best of our knowledge, our work is the first to focus on the aspect of using flows to pre-process sequential data to improve downstream dynamics modeling.

In this paper, we utilize affine flows (Eq. 2). This family of flows includes methods like NICE (Dinh et al., 2015), RealNVP (Dinh et al., 2017), IAF (Kingma et al., 2016), MAF (Papamakarios et al., 2017), and GLOW (Kingma and Dhariwal, 2018). However, there has been recent work in non-affine flows (Huang et al., 2018; Jaini et al., 2019; Durkan et al., 2019), which may offer further flexibility. We chose to investigate affine flows for their relative simplicity and connections to previous techniques, however, the use of non-affine flows could result in additional improvements.

Appendix B. A Motivating Example

Consider the discrete dynamical system defined by the following set of equations:

$$\mathbf{x}_t = \mathbf{x}_{t-1} + \mathbf{u}_t, \tag{7}$$
$$\mathbf{u}_t = \mathbf{u}_{t-1} + \mathbf{w}_t, \tag{8}$$

where $\mathbf{w}_t \sim \mathcal{N}(\mathbf{w}_t; \mathbf{0}, \mathbf{\Sigma})$. We can express $\mathbf{x}_t$ and $\mathbf{u}_t$ in probabilistic terms as

$$\mathbf{x}_t \sim \mathcal{N}(\mathbf{x}_t; \mathbf{x}_{t-1} + \mathbf{u}_{t-1}, \mathbf{\Sigma}), \tag{9}$$
$$\mathbf{u}_t \sim \mathcal{N}(\mathbf{u}_t; \mathbf{u}_{t-1}, \mathbf{\Sigma}). \tag{10}$$

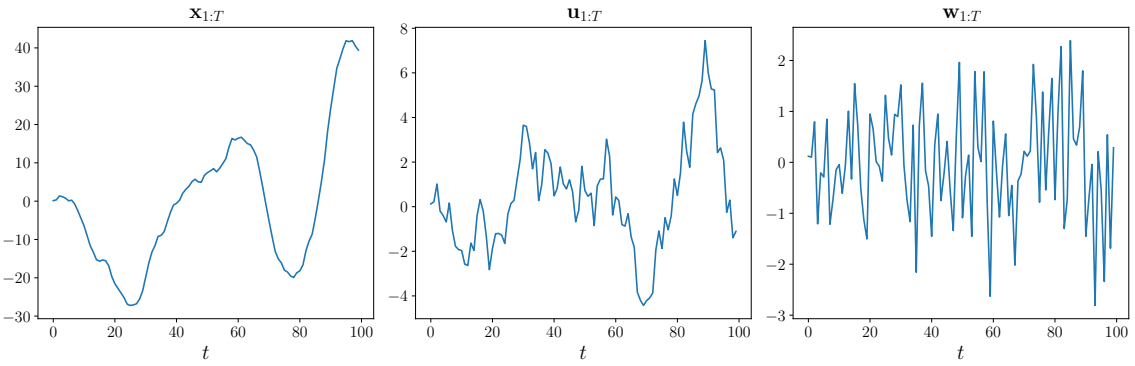


Figure 4: **Motivating Example**. Plots are shown for a sample of $\mathbf{x}_{1:T}$ (left), $\mathbf{u}_{1:T}$ (center), and $\mathbf{w}_{1:T}$ (right). Here, $\mathbf{w}_{1:T} \sim \mathcal{N}(\mathbf{w}_{1:T}; \mathbf{0}, \mathbf{I})$, and $\mathbf{u}$ and $\mathbf{x}$ are initialized at 0. Moving from $\mathbf{x} \rightarrow \mathbf{u} \rightarrow \mathbf{w}$ via affine transforms results in successively less temporal correlation and therefore simpler dynamics.

Physically, this describes the noisy dynamics of a particle with momentum and mass 1, subject to Gaussian noise. If we consider the dynamics at the level of $\mathbf{x}$, we can use the fact that $\mathbf{u}_{t-1} = \mathbf{x}_{t-1} - \mathbf{x}_{t-2}$ to write

$$p(\mathbf{x}_t|\mathbf{x}_{t-1}, \mathbf{x}_{t-2}) = \mathcal{N}(\mathbf{x}_t; \mathbf{x}_{t-1} + \mathbf{x}_{t-1} - \mathbf{x}_{t-2}, \mathbf{\Sigma}). \tag{11}$$

Thus, we see that in the space of $\mathbf{x}$, the dynamics are second-order Markov, requiring knowledge of the past two time steps. However, at the level of $\mathbf{u}$ (Eq. 10), the dynamics are first-order Markov, requiring only the previous time step. Yet, note that $\mathbf{u}_t$ is, in fact, an affine autoregressive transform of $\mathbf{x}_t$ because $\mathbf{u}_t = \mathbf{x}_t - \mathbf{x}_{t-1}$ is a special case of the general form $\frac{\mathbf{x}_t - \boldsymbol{\mu}_\theta(\mathbf{x}_{<t})}{\boldsymbol{\sigma}_\theta(\mathbf{x}_{<t})}$. In Eq. 7, we see that the Jacobian of this transform is $\partial \mathbf{x}_t / \partial \mathbf{u}_t = \mathbf{I}$, so, from the change of variables formula, we have $p(\mathbf{x}_t|\mathbf{x}_{t-1}, \mathbf{x}_{t-2}) = p(\mathbf{u}_t|\mathbf{u}_{t-1})$. In other words, an affine autoregressive transform has allowed us to convert a second-order Markov system into a first-order Markov system, thereby simplifying the dynamics. Continuing this process to move to $\mathbf{w}_t = \mathbf{u}_t - \mathbf{u}_{t-1}$, we arrive at a representation that is entirely temporally decorrelated, i.e. no dynamics, because $p(\mathbf{w}_t) = \mathcal{N}(\mathbf{w}_t; \mathbf{0}, \mathbf{\Sigma})$. A sample from this system is shown in Figure 4, illustrating this process of temporal decorrelation.

The special case of modeling temporal changes, $\mathbf{u}_t = \mathbf{x}_t - \mathbf{x}_{t-1} = \Delta \mathbf{x}_t$, is a common pre-processing technique; for recent examples, see Deisenroth et al. (2013); Chua et al. (2018); Kumar et al. (2019). In fact, $\Delta \mathbf{x}_t$ is a finite differences approximation of the generalized velocity (Friston, 2008) of $\mathbf{x}$, a classic modeling technique in dynamical models and control (Kalman et al., 1960), redefining the state-space to be first-order Markov. Affine autoregressive flows offer a generalization of this technique, allowing for non-linear transform parameters and flows consisting of multiple transforms, with each transform serving to successively decorrelate the input sequence in time. In analogy with generalized velocity, each transform serves as a *moving reference frame*, allowing us to focus model capacity on less correlated fluctuations rather than the highly correlated raw signal.

Appendix C. Lower Bound Derivation

Consider the model defined in Section 3, with the conditional likelihood parameterized with autoregressive flows. That is, we parameterize

$$\mathbf{x}_t = \boldsymbol{\mu}_\theta(\mathbf{x}_{<t}) + \boldsymbol{\sigma}_\theta(\mathbf{x}_{<t}) \odot \mathbf{y}_t \tag{12}$$

yielding

$$p_\theta(\mathbf{x}_t|\mathbf{x}_{<t}, \mathbf{z}_{\leq t}) = p_\theta(\mathbf{y}_t|\mathbf{y}_{<t}, \mathbf{z}_{\leq t}) \left|\det \left(\frac{\partial \mathbf{x}_t}{\partial \mathbf{y}_t}\right)\right|^{-1}. \tag{13}$$

The joint distribution over all time steps is then given as

$$p_\theta(\mathbf{x}_{1:T}, \mathbf{z}_{1:T}) = \prod_{t=1}^{T} p_\theta(\mathbf{x}_t|\mathbf{x}_{<t}, \mathbf{z}_{\leq t}) p_\theta(\mathbf{z}_t|\mathbf{x}_{<t}, \mathbf{z}_{<t}) \tag{14}$$

$$= \prod_{t=1}^{T} p_\theta(\mathbf{y}_t|\mathbf{y}_{<t}, \mathbf{z}_{\leq t}) \left|\det \left(\frac{\partial \mathbf{x}_t}{\partial \mathbf{y}_t}\right)\right|^{-1} p_\theta(\mathbf{z}_t|\mathbf{x}_{<t}, \mathbf{z}_{<t}). \tag{15}$$

To perform variational inference, we consider a filtering approximate posterior of the form

$$q(\mathbf{z}_{1:T}|\mathbf{x}_{1:T}) = \prod_{t=1}^{T} q(\mathbf{z}_t|\mathbf{x}_{\leq t}, \mathbf{z}_{<t}). \tag{16}$$

We can then plug these expressions into the evidence lower bound:

$$\mathcal{L} \equiv \mathbb{E}_{q(\mathbf{z}_{1:T}|\mathbf{x}_{1:T})} \left[\log p_\theta(\mathbf{x}_{1:T}, \mathbf{z}_{1:T}) - \log q(\mathbf{z}_{1:T}|\mathbf{x}_{1:T})\right] \tag{17}$$

$$= \mathbb{E}_{q(\mathbf{z}_{1:T}|\mathbf{x}_{1:T})} \left[\log \left(\prod_{t=1}^{T} p_\theta(\mathbf{y}_t|\mathbf{y}_{<t}, \mathbf{z}_{\leq t}) \left|\det \left(\frac{\partial \mathbf{x}_t}{\partial \mathbf{y}_t}\right)\right|^{-1} p_\theta(\mathbf{z}_t|\mathbf{x}_{<t}, \mathbf{z}_{<t}) \right) \right.$$
$$\left. - \log \left(\prod_{t=1}^{T} q(\mathbf{z}_t|\mathbf{x}_{\leq t}, \mathbf{z}_{<t}) \right) \right] \tag{18}$$

$$= \mathbb{E}_{q(\mathbf{z}_{1:T}|\mathbf{x}_{1:T})} \left[\sum_{t=1}^{T} \log p_\theta(\mathbf{y}_t|\mathbf{y}_{<t}, \mathbf{z}_{\leq t}) - \log \frac{q(\mathbf{z}_t|\mathbf{x}_{\leq t}, \mathbf{z}_{<t})}{p_\theta(\mathbf{z}_t|\mathbf{x}_{<t}, \mathbf{z}_{<t})} - \log \left|\det \left(\frac{\partial \mathbf{x}_t}{\partial \mathbf{y}_t}\right)\right| \right]. \tag{19}$$

Finally, in the filtering setting, we can rewrite the expectation, bringing it inside of the sum (see Gemici et al. (2017); Marino et al. (2018)):

$$\mathcal{L} = \sum_{t=1}^{T} \mathbb{E}_{q(\mathbf{z}_{\leq t}|\mathbf{x}_{\leq t})} \left[\log p_\theta(\mathbf{y}_t|\mathbf{y}_{<t}, \mathbf{z}_{\leq t}) - \log \frac{q(\mathbf{z}_t|\mathbf{x}_{\leq t}, \mathbf{z}_{<t})}{p_\theta(\mathbf{z}_t|\mathbf{x}_{<t}, \mathbf{z}_{<t})} - \log \left|\det \left(\frac{\partial \mathbf{x}_t}{\partial \mathbf{y}_t}\right)\right| \right]. \tag{20}$$

Because there exists a one-to-one mapping between $\mathbf{x}_{1:T}$ and $\mathbf{y}_{1:T}$, we can equivalently condition the approximate posterior and the prior on $\mathbf{y}$, i.e.

$$\mathcal{L} = \sum_{t=1}^{T} \mathbb{E}_{q(\mathbf{z}_{\leq t}|\mathbf{y}_{\leq t})} \left[\log p_\theta(\mathbf{y}_t|\mathbf{y}_{<t}, \mathbf{z}_{\leq t}) - \log \frac{q(\mathbf{z}_t|\mathbf{y}_{\leq t}, \mathbf{z}_{<t})}{p_\theta(\mathbf{z}_t|\mathbf{y}_{<t}, \mathbf{z}_{<t})} - \log \left|\det \left(\frac{\partial \mathbf{x}_t}{\partial \mathbf{y}_t}\right)\right| \right]. \tag{21}$$

Appendix D. Experiment Details

We store a fixed number of past frames in the buffer of each transform, to generate the shift and scale for the transform. For each stack of flow, 4 convolutional layers with kernel size $(3, 3)$, stride 1 and padding 1 are applied first on each data observation in the buffer, preserving the data shape. The outputs are concatenated along the channel dimension and go through another four convolutional layers also with kernel size $(3, 3)$, stride 1 and padding 1. Finally, separate convolutional layers with the same kernel size, stride and padding are used to generate shift and scale respectively.

For latent variable models, we use a DC-GAN structure (Radford et al., 2015), with 4 layers of convolutional layers of kernel size $(4, 4)$, stride 2 and padding 1 before another convolutional layer of kernel size $(4, 4)$, stride 1 and no padding to encode the data. The encoded data is sent to an LSTM (Hochreiter and Schmidhuber, 1997) followed by fully connected layers to generate the mean and log-variance for estimating the approximate posterior distribution of the latent variable, $\mathbf{z}_t$. The conditional prior distribution is modeled with another LSTM followed by fully connected layers, taking the previous latent variable as input. The decoder take the inverse structure of the encoder. In the SLVM, we use 2 LSTM layers for modelling the conditional prior and approximate posterior distributions, while in the combined model we use 1 LSTM layer for each.

We use the Adam optimizer (Kingma and Ba, 2014) with a learning rate of 1×10^{-4} to train all the models. For Moving MNIST, we use a batch size of 16 and train for $200,000$ iterations for latent variable models and $100,000$ iterations for flow-based and latent variable models with flow-based likelihoods. For BAIR Robot Pushing, we use a batch size of 8 and train for $200,000$ iterations for all models. For KTH dataset we use a batch size of 8 and train for $90,000$ iterations for all models. Batch norm (Ioffe and Szegedy, 2015) is applied to all convolutional layers that do not directly generate distribution or transform parameters. We randomly crop sequence of length 13 from all sequences and evaluate on the last 10 frames. (For 2-flow models we crop sequence of length 16 to fill up all buffers.) Code is available here[2].

Table 2: **Number of parameters for each model on each dataset.** Flow-based models contain relatively few parameters as compared with the SLVM, as our flows consist primarily of 3×3 convolutions with limited channels. In the SLVM, we use 2 LSTM layers for modelling the prior and posterior distribution of latent variable while in the combined model we use 1 LSTM layer for each.

Model	1-AF	2-AF	SLVM	SLVM w/ 1-AF
Moving Mnist	343k	686k	11302k	10592k
BAIR Robot Pushing	363k	726k	11325k	10643k
KTH Action	343k	686k	11302k	10592k

2. https://github.com/joelouismarino/sequential_flows

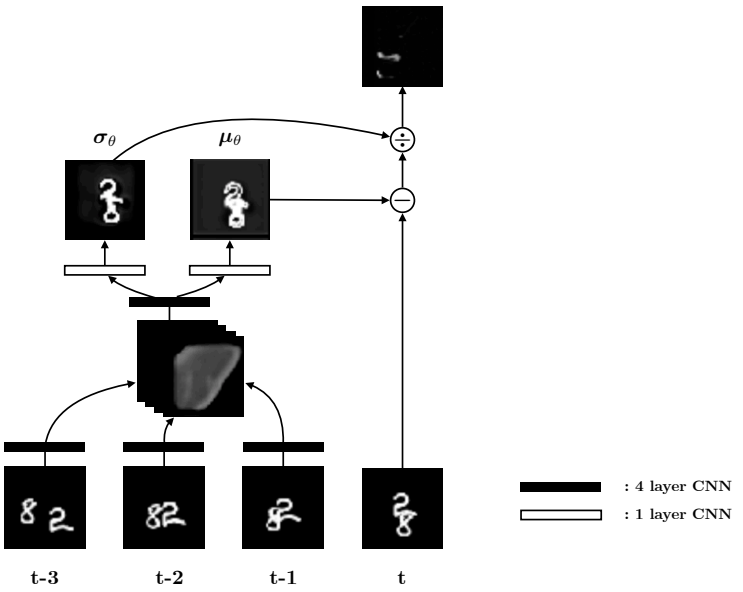


Figure 5: Implementation Visualization of the autoregressive flow.

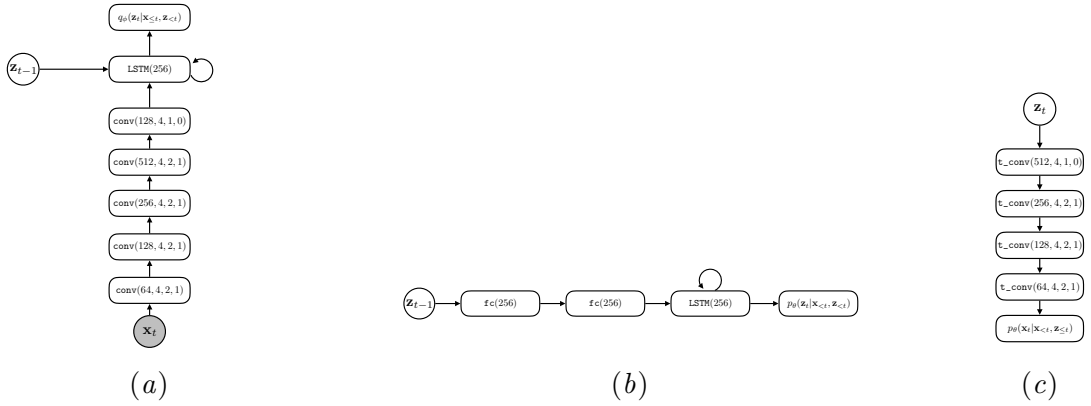


Figure 6: **Model Architecture Diagrams**. Diagrams are shown for the (**a**) approximate posterior, (**b**) conditional prior, and (**c**) conditional likelihood of the sequential latent variable model. `conv` denotes a convolutional layer, `LSTM` denotes a long short-term memory layer, `fc` denotes a fully-connected layer, and `t_conv` denotes a transposed convolutional layer. For `conv` and `t_conv` layers, the numbers in parentheses respectively denote the number of filters, filter size, stride, and padding of the layer. For `fc` and `LSTM` layers, the number in parentheses denotes the number of units.

Table 3: **Training Quantitative Comparison.** Average **training** log-likelihood (higher is better) in *nats per pixel per channel* for Moving MNIST, BAIR Robot Pushing, and KTH Actions. For flow-based models (1-AF and 2-AF), we report the average log-likelihood. For sequential latent variable models (SLVM and SLVM w/ 1-AF), we report the average lower bound on the log-likelihood.

	M-MNIST	BAIR	KTH
1-AF	-2.06	-2.98	-2.95
2-AF	-2.04	-2.76	-2.95
SLVM	≥ -1.93	≥ -3.46	≥ -3.05
SLVM w/ 1-AF	$\geq \mathbf{-1.85}$	$\geq \mathbf{-2.31}$	$\geq \mathbf{-2.21}$

Appendix E. Additional Experimental Results

E.1. Quantifying Temporal Decorrelation

The qualitative results in Figures 2 and 8 demonstrate that flows are capable of removing much of the structure of the observations, resulting in *whitened* noise images. To quantitatively confirm the temporal decorrelation resulting from this process, we evaluate the empirical correlation between successive frames, averaged over spatial locations and channels, for the data observations and noise variables. This is an average normalized version of the *auto-covariance* of each signal with a time delay of 1 time step. Specifically, we estimate the temporal correlation as

$$\text{corr}_{\mathbf{x}} \equiv \frac{1}{C * W * H} \cdot \sum_{i,j,k}^{H,W,C} \mathbb{E}_{x_t^{(i,j,k)}, x_{t+1}^{(i,j,k)} \sim \mathcal{D}} \left[\frac{(x_t^{(i,j,k)} - \mu^{(i,j,k)})(x_{t+1}^{(i,j,k)} - \mu^{(i,j,k)})}{\left(\sigma^{(i,j,k)}\right)^2} \right], \quad (22)$$

where $x^{(i,j,k)}$ denotes the value of the image at location (i, j) and channel k, $\mu^{(i,j,k)}$ denotes the mean of this dimension, and $\sigma^{(i,j,k)}$ denotes the standard deviation of this dimension. H, W, and C respectively denote the height, width, and number of channels of the observations.

Table 4: **Temporal Correlation.** Temporal correlation (Eq. 22) between successive time steps for data observations, $\mathbf{x}$, and noise variables, $\mathbf{y}$, for SLVM w/ 1-AF.

	M-MNIST	BAIR	KTH
$\text{corr}_{\mathbf{x}}$	0.24	0.87	0.96
$\text{corr}_{\mathbf{y}}$	0.02	0.43	0.31

We evaluated this quantity for data examples, $\mathbf{x}$, and noise variables, $\mathbf{y}$, for SLVM w/ 1-AF. The results for training sequences are shown in Table 4. In Figure 7, we plot this quantity during training for KTH Actions. We see that flows do indeed result in a decrease in temporal correlation. Note that because correlation is a measure of *linear* dependence, one cannot conclude from these results alone that the flows have resulted in simplified temporal structure. However, these results agree with the qualitative and quantitative

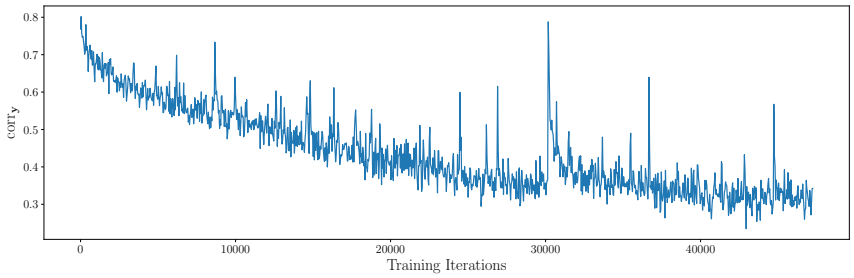


Figure 7: **Temporal Correlation During Training**. $\text{corr}_{\mathbf{y}}$ during training for SLVM w/ 1-AF on the KTH Actions. Temporal correlation decreases substantially during training.

results presented in Section 4, suggesting that autoregressive flows can yield sequences with simpler dynamics.

E.2. Additional Qualitative Experiments

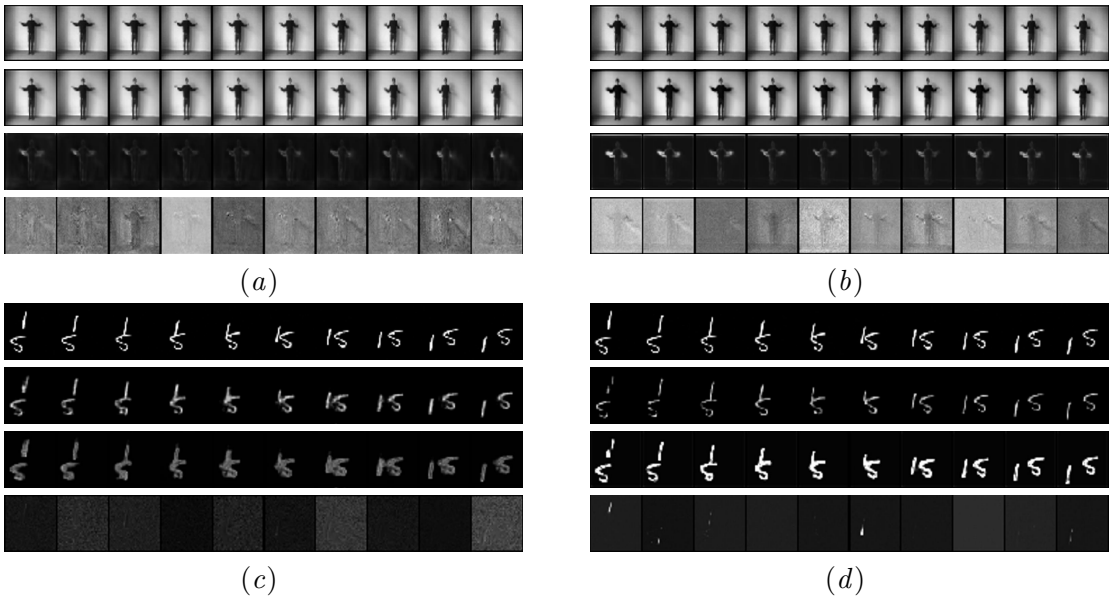


Figure 8: **Flow Visualization**. Visualization of the flow component for a **(a, c)** standalone flow-based model and **(b, d)** sequential latent variable model with flow-based conditional likelihood for KTH Actions and Moving MNIST. From top to bottom, each figure shows 1) the original frames, $\mathbf{x}_t$, 2) the predicted shift, $\boldsymbol{\mu}_\theta(\mathbf{x}_{<t})$, for the frame, 3) the predicted scale, $\boldsymbol{\sigma}_\theta(\mathbf{x}_{<t})$, for the frame, and 4) the noise, $\mathbf{y}_t$, obtained from the inverse transform.

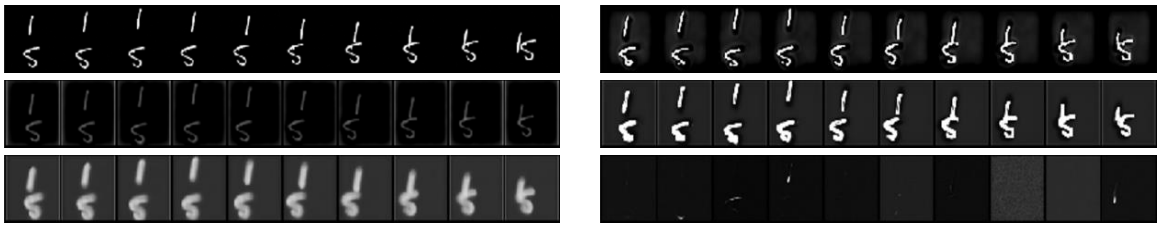
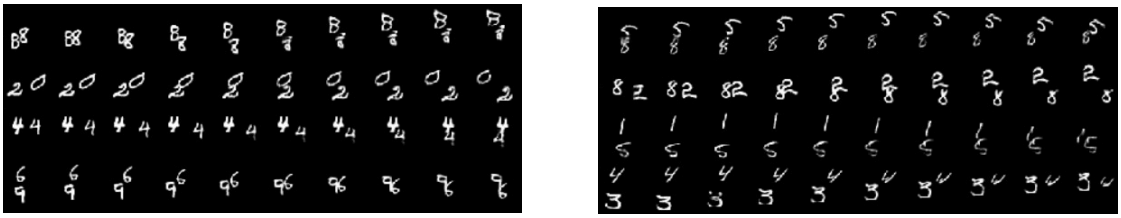


Figure 9: **SLVM w/ 2-AF Visualization on Moving MNIST**. Visualization of the flow component for sequential latent variable models with 2-layer flow-based conditional likelihoods for Moving MNIST. From top to bottom on the left side, each figure shows 1) the original frames, $\mathbf{x}_t$, 2) the lower-level predicted shift, $\boldsymbol{\mu}_\theta^1(\mathbf{x}_{<t})$, for the frame, 3) the predicted scale, $\boldsymbol{\sigma}_\theta^1(\mathbf{x}_{<t})$, for the frame. On the right side, from top to bottom, we have 1) the higer-level predicted shift, $\boldsymbol{\mu}_\theta^2(\mathbf{x}_{<t})$, for the frame, 3) the predicted scale, $\boldsymbol{\sigma}_\theta^2(\mathbf{x}_{<t})$, for the frame and 4) the noise, $\mathbf{y}_t$, obtained from the inverse transform.



(a) (b)

Figure 10: **Generated Moving MNIST Samples**. Samples frame sequences generated from a 2-AF model.

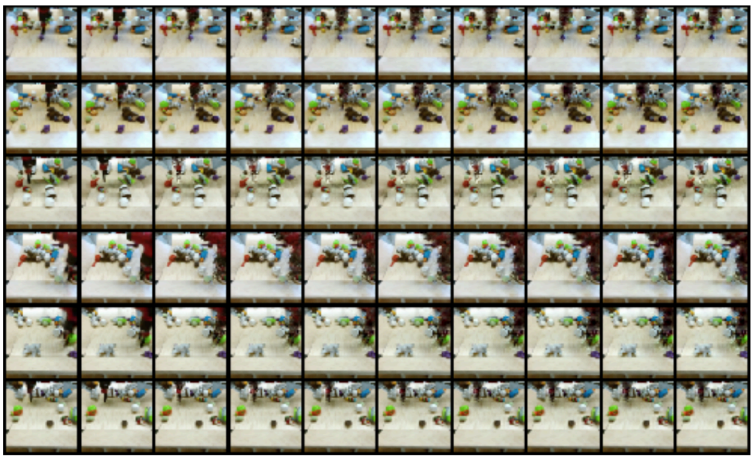


Figure 11: **Generated BAIR Robot Pushing Samples**. Samples frame sequences generated from SLVM w/ 1-AF.