Improving Sequential Latent Variable Models with Autoregressive Flows

Joseph Marino

California Institute of Technology
jmarino@caltech.edu

Lei Chen

Simon Fraser University
lei_chen_4@sfu.ca

Jiawei He

Simon Fraser University
jha203@sfu.ca

Stephan Mandt

University of California Irvine
mandt@uci.edu

Abstract

We propose an approach for improving sequence modeling based on autoregressive normalizing flows. Each autoregressive transform, acting across time, serves as a moving frame of reference, removing temporal correlations and simplifying the modeling of 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 flow-based models and as a component within sequential latent variable models. Results are presented on three benchmark video datasets, where autoregressive flow-based dynamics improve log-likelihood performance over baseline models. Finally, we illustrate the decorrelation and improved generalization properties of using flow-based dynamics.

1 Introduction

Data often contain sequential structure, providing a rich signal for learning models of the world. Such models are useful for representing sequences [40, 21] and planning actions [6, 22]. Recent advances in deep learning have facilitated learning sequential probabilistic models directly from high-dimensional data [20], like audio and video. A variety of techniques have emerged for learning deep sequential models, including memory units [25] and stochastic latent variables [3, 7]. These techniques have enabled sequential models to capture increasingly complex dynamics. In this paper, we explore the complementary direction, asking *can we simplify the dynamics of the data to meet the capacity of the model?* To do so, we aim to learn a *frame of reference* to assist in modeling the data.

Frames of reference are an important consideration in sequence modeling, as they can simplify dynamics by removing redundancy. For instance, in a physical system, the frame of reference that moves with the system’s center of mass removes the redundancy in displacement. Frames of reference are also more widely applicable to arbitrary sequences. Indeed, video compression schemes use predictions as a frame of reference to remove temporal redundancy [45]. By learning and applying a similar type of temporal normalization for sequence modeling, the model can focus on aspects that are not predicted by the low-level frame of reference, thereby simplifying dynamics modeling.

We formalize this notion of temporal normalization through the framework of autoregressive normalizing flows [37, 46]. In the context of sequences, these flows form predictions across time, attempting to remove temporal dependencies [58]. Thus, autoregressive flows can act as a pre-processing technique to simplify dynamics. We preview this approach in Figure 1, where an autoregressive flow modeling the data (top) creates a transformed space for modeling dynamics (bottom). The transformed space is largely invariant to absolute pixel value, focusing instead on capturing deviations and motion.

Figure 1: **Sequence Modeling with Autoregressive Flows.** **Top:** Pixel values (solid) for a particular pixel location in a video sequence. An autoregressive flow models the pixel sequence using an affine shift (dashed) and scale (shaded), acting as a frame of reference. **Middle:** Frames of the data sequence (top) and the resulting “noise” (bottom) from applying the shift and scale. The redundant, static background has been largely removed. **Bottom:** The noise values (solid) are modeled using a base distribution (dashed and shaded) provided by a higher-level model. By removing temporal redundancy from the data sequence, the autoregressive flow simplifies dynamics modeling.

We empirically demonstrate this modeling technique, both with standalone autoregressive normalizing flows, as well as within sequential latent variable models. While normalizing flows have been applied in sequential contexts previously, our main contributions are in **1)** showing how these models can act as a general pre-processing technique to improve dynamics modeling, **2)** empirically demonstrating log-likelihood performance improvements, as well as generalization improvements, on three benchmark video datasets. This technique also connects to previous work in dynamics modeling, probabilistic models, and sequence compression, enabling directions for further investigation.

2 Background

2.1 Autoregressive Models

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 [15, 4] use the chain rule of probability to express the joint distribution over time steps as the product of T conditional distributions. These models are often formulated in forward temporal order:

$$p_{\theta}(\mathbf{x}_{1:T}) = \prod_{t=1}^T p_{\theta}(\mathbf{x}_t | \mathbf{x}_{<t}). \quad (1)$$

Each conditional distribution, $p_{\theta}(\mathbf{x}_t | \mathbf{x}_{<t})$, models the dependence between time steps. For continuous variables, it is often assumed that each distribution takes a simple form, such as a diagonal Gaussian: $p_{\theta}(\mathbf{x}_t | \mathbf{x}_{<t}) = \mathcal{N}(\mathbf{x}_t; \boldsymbol{\mu}_{\theta}(\mathbf{x}_{<t}), \text{diag}(\boldsymbol{\sigma}_{\theta}^2(\mathbf{x}_{<t})))$, where $\boldsymbol{\mu}_{\theta}(\cdot)$ and $\boldsymbol{\sigma}_{\theta}(\cdot)$ are functions denoting the mean and standard deviation. These functions may take past observations as input through a recurrent network or a convolutional window [60]. When applied to spatial data [61], autoregressive models excel at capturing local dependencies. However, due to their restrictive forms, such models often struggle to capture more complex structure.

2.2 Autoregressive (Sequential) Latent Variable Models

Autoregressive models can be improved by incorporating latent variables [43], often represented as a corresponding sequence, $\mathbf{z}_{1:T}$. The joint distribution, $p_\theta(\mathbf{x}_{1:T}, \mathbf{z}_{1:T})$, has the form:

$$p_\theta(\mathbf{x}_{1:T}, \mathbf{z}_{1:T}) = \prod_{t=1}^T p_\theta(\mathbf{x}_t | \mathbf{x}_{<t}, \mathbf{z}_{\leq t}) p_\theta(\mathbf{z}_t | \mathbf{x}_{<t}, \mathbf{z}_{<t}). \quad (2)$$

Unlike the Gaussian form, evaluating $p_\theta(\mathbf{x}_t | \mathbf{x}_{<t})$ now requires integrating over the latent variables,

$$p_\theta(\mathbf{x}_t | \mathbf{x}_{<t}) = \int p_\theta(\mathbf{x}_t | \mathbf{x}_{<t}, \mathbf{z}_{\leq t}) p_\theta(\mathbf{z}_{\leq t} | \mathbf{x}_{<t}) d\mathbf{z}_{\leq t}, \quad (3)$$

yielding a more flexible distribution. However, performing this integration in practice is typically intractable, requiring approximate inference techniques, like variational inference [31], or invertible models [38]. Recent works have parameterized these models with deep neural networks, e.g. [7, 18, 14, 32], using amortized variational inference [35, 52]. Typically, the conditional likelihood, $p_\theta(\mathbf{x}_t | \mathbf{x}_{<t}, \mathbf{z}_{\leq t})$, and the prior, $p_\theta(\mathbf{z}_t | \mathbf{x}_{<t}, \mathbf{z}_{<t})$, are Gaussian densities, with temporal conditioning handled through recurrent networks. Such models have demonstrated success in audio [7, 14] and video modeling [65, 19, 9, 23, 40]. However, as noted by Kumar et al. [38], such models can be difficult to train with standard log-likelihood objectives, often struggling to capture dynamics.

2.3 Autoregressive Flows

Our approach is based on affine autoregressive normalizing flows [37, 46]. Here, we continue with the perspective of temporal sequences, however, these flows were initially developed and demonstrated in *static* settings. Kingma et al. [37] noted that sampling from an autoregressive Gaussian model is an invertible transform, resulting in a *normalizing flow* [55, 10, 11, 51]. Flow-based models transform simple, *base* probability distributions into more complex ones while maintaining exact likelihood evaluation. To see their connection to autoregressive models, we can express sampling a Gaussian random variable using the reparameterization trick [35, 52]:

$$\mathbf{x}_t = \boldsymbol{\mu}_\theta(\mathbf{x}_{<t}) + \boldsymbol{\sigma}_\theta(\mathbf{x}_{<t}) \odot \mathbf{y}_t, \quad (4)$$

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

$$\mathbf{y}_t = \frac{\mathbf{x}_t - \boldsymbol{\mu}_\theta(\mathbf{x}_{<t})}{\boldsymbol{\sigma}_\theta(\mathbf{x}_{<t})}, \quad (5)$$

where division is element-wise. The inverse transform in Eq. 5, shown in Figure 2, normalizes (hence, *normalizing flow*) $\mathbf{x}_{1:T}$, removing statistical dependencies. Given the functional mapping between $\mathbf{y}_t$ and $\mathbf{x}_t$ in Eq. 4, the change of variables formula converts between probabilities in each space:

$$\log p_\theta(\mathbf{x}_{1:T}) = \log p_\theta(\mathbf{y}_{1:T}) - \log \left| \det \left(\frac{\partial \mathbf{x}_{1:T}}{\partial \mathbf{y}_{1:T}} \right) \right|. \quad (6)$$

By the construction of Eqs. 4 and 5, the Jacobian in Eq. 6 is triangular, enabling efficient evaluation as the product of diagonal terms:

$$\log \left| \det \left(\frac{\partial \mathbf{x}_{1:T}}{\partial \mathbf{y}_{1:T}} \right) \right| = \sum_{t=1}^T \sum_i \log \sigma_{\theta,i}(\mathbf{x}_{<t}), \quad (7)$$

where i denotes the observation dimension, e.g. pixel. For a Gaussian autoregressive model, the base distribution is $p_\theta(\mathbf{y}_{1:T}) = \mathcal{N}(\mathbf{y}_{1:T}; \mathbf{0}, \mathbf{I})$. We can improve upon this simple set-up by chaining transforms together, i.e. parameterizing $p_\theta(\mathbf{y}_{1:T})$ as a flow, resulting in hierarchical models.

Figure 2: **Affine Autoregressive Transform.** Computational diagram for an affine autoregressive transform [46]. 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, shown here, is capable of converting a *correlated* input, $\mathbf{x}_{1:T}$, into an *uncorrelated* output, $\mathbf{y}_{1:T}$.

Figure 3: **Redundancy Reduction.** (a) Conditional densities for $p(x_2|x_1)$. (b) The marginal, $p(x_2)$ differs from the conditional densities, thus, $\mathcal{I}(x_1; x_2) > 0$. (c) In the normalized space of y , the corresponding densities $p(y_2|y_1)$ are identical. (d) The marginal $p(y_2)$ is identical to the conditionals, so $\mathcal{I}(y_1; y_2) = 0$. Thus, in this case, a conditional affine transform removed the dependencies.

2.4 Related Work

Autoregressive flows were initially considered in the contexts of variational inference [37] and generative modeling [46]. These approaches are generalizations of previous approaches with affine transforms [10, 11]. While autoregressive flows are well-suited for sequential data, these approaches, as well as many recent approaches [26, 44, 36], were initially applied to static data, such as images.

Recent works have started applying flow-based models to sequential data. van den Oord et al. [62] and Ping et al. [47] *distill* autoregressive speech models into flow-based models. Prenger et al. [49] and Kim et al. [33] instead train these models directly. Kumar et al. [38] use a flow to model individual video frames, with an autoregressive prior modeling dynamics across time steps. Rhinehart et al. [53, 54] use autoregressive flows for modeling vehicle motion, and Henter et al. [24] use flows for motion synthesis with motion-capture data. Ziegler & Rush [66] model 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 sequences. However, we demonstrate that autoregressive flows can serve as a general-purpose technique for improving dynamics models. To the best of our knowledge, our work is the first to use flows to pre-process sequences to improve sequential latent variable models.

We utilize affine flows (Eq. 4), a family that includes methods like NICE [10], RealNVP [11], IAF [37], MAF [46], and Glow [36]. However, there has been recent work in non-affine flows [26, 30, 12], which offer further flexibility. We chose to investigate affine flows because they are commonly employed and relatively simple, however, non-affine flows could result in additional improvements.

3 Method

We now describe our approach for improving sequence modeling. First, we motivate using autoregressive flows to reduce temporal dependencies, thereby simplifying dynamics. We then show how this simple technique can be incorporated within sequential latent variable models.

3.1 Motivation: Temporal Redundancy Reduction

Normalizing flows, while often utilized for density estimation, originated from data pre-processing techniques [16, 28, 5], which remove dependencies between dimensions, i.e. *redundancy reduction* [2]. Removing dependencies simplifies the resulting probability distribution by restricting variation to individual dimensions, generally simplifying downstream tasks [39]. Normalizing flows improve upon these procedures using flexible, non-linear functions [8, 10]. While flows have been used for spatial decorrelation [1, 64] and with other models [27], this capability remains under-explored.

Our main contribution is showing how to utilize autoregressive flows for temporal pre-processing to improve dynamics modeling. Data sequences contain dependencies in time, for example, in the redundancy of video pixels (Figure 1), which are often highly predictable. These dependencies define the *dynamics* of the data, with the degree of dependence quantified by the multi-information,

$$\mathcal{I}(\mathbf{x}_{1:T}) = \sum_t \mathcal{H}(\mathbf{x}_t) - \mathcal{H}(\mathbf{x}_{1:T}), \quad (8)$$

where $\mathcal{H}$ denotes entropy. Normalizing flows are capable of reducing redundancy, arriving at a new sequence, $\mathbf{y}_{1:T}$, with $\mathcal{I}(\mathbf{y}_{1:T}) \leq \mathcal{I}(\mathbf{x}_{1:T})$, thereby reducing temporal dependencies. Thus, rather than

Figure 4: **Model Diagrams.** (a) An autoregressive flow pre-processes a data sequence, $\mathbf{x}_{1:T}$, to produce a new sequence, $\mathbf{y}_{1:T}$, with reduced temporal dependencies. This simplifies dynamics modeling for a higher-level sequential latent variable model, $p_\theta(\mathbf{y}_{1:T}, \mathbf{z}_{1:T})$. Empty diamond nodes represent deterministic dependencies, not recurrent states. (b) Diagram of the autoregressive flow architecture. Blank white rectangles represent convolutional layers (see Appendix). The three stacks of convolutional layers within the blue region are shared. cat denotes channel-wise concatenation.

fit the data distribution directly, we can first simplify the dynamics by pre-processing sequences with a normalizing flow, then fitting the resulting sequence. Through training, the flow will attempt to remove redundancies to meet the modeling capacity of the higher-level dynamics model, $p_\theta(\mathbf{y}_{1:T})$.

Example To visualize this procedure for an affine autoregressive flow, consider a one-dimensional input over two time steps, x_1 and x_2 . For each value of x_1 , there is a conditional density, $p(x_2|x_1)$. Assume that these densities take one of two forms, which are identical but shifted and scaled, shown in Figure 3. Transforming these densities through their conditional means, $\mu_2 = \mathbb{E}[x_2|x_1]$, and standard deviations, $\sigma_2 = \mathbb{E}[(x_2 - \mu_2)^2|x_1]^{1/2}$, creates a normalized space, $y_2 = (x_2 - \mu_2)/\sigma_2$, where the conditional densities are identical. In this space, the multi-information is

$$\mathcal{I}(y_1; y_2) = \mathbb{E}_{p(y_1, y_2)} [\log p(y_2|y_1) - \log p(y_2)] = 0,$$

whereas $\mathcal{I}(x_1; x_2) > 0$. Indeed, if $p(x_t|x_{<t})$ is linear-Gaussian, inverting an affine autoregressive flow exactly corresponds to Cholesky whitening [48, 37], removing all linear dependencies.

In the example above, μ_2 and σ_2 act as a *frame of reference* for estimating x_2 . More generally, in the special case where $\mu_\theta(\mathbf{x}_{<t}) = \mathbf{x}_{t-1}$ and $\sigma_\theta(\mathbf{x}_{<t}) = \mathbf{1}$, we recover $\mathbf{y}_t = \mathbf{x}_t - \mathbf{x}_{t-1} = \Delta \mathbf{x}_t$. Modeling finite differences (or *generalized coordinates* [17]) is a well-established technique, (see, e.g. [6, 38]), which is generalized by affine autoregressive flows.

3.2 Modeling Dynamics with Autoregressive Flows

We now discuss utilizing autoregressive flows to improve sequence modeling, highlighting use cases for modeling dynamics in the data and latent spaces.

3.2.1 Data Dynamics

The form of an affine autoregressive flow across sequences is given in Eqs. 4 and 5, again, equivalent to a Gaussian autoregressive model. We can stack hierarchical chains of flows to improve the model capacity. Denoting the shift and scale functions at the m^{th} transform as $\mu_\theta^m(\cdot)$ and $\sigma_\theta^m(\cdot)$ respectively, we then calculate $\mathbf{y}^m$ using the inverse transform:

$$\mathbf{y}_t^m = \frac{\mathbf{y}_t^{m-1} - \mu_\theta^m(\mathbf{y}_{<t}^{m-1})}{\sigma_\theta^m(\mathbf{y}_{<t}^{m-1})}. \quad (9)$$

After the final (M^{th}) transform, we can choose the form of the base distribution, $p_\theta(\mathbf{y}_{1:T}^M)$, e.g. Gaussian. While we could attempt to model $\mathbf{x}_{1:T}$ completely using stacked autoregressive flows,

Figure 5: **Decreased Temporal Correlation.** (a) Affine autoregressive flows result in sequences, $\mathbf{y}_{1:T}$, with decreased temporal correlation, corr_y , as compared with that of the original data, corr_x . (b) For SLVM + 1-AF, corr_y decreases during training on KTH Actions.

these models are limited to affine element-wise transforms that maintain the data dimensionality. Due to this limited capacity, purely flow-based models often require many transforms to be effective [36].

Instead, we can model the base distribution using an expressive sequential latent variable model (SLVM), or, equivalently, we can augment the conditional likelihood of a SLVM using autoregressive flows (Fig. 4a). Following the motivation from Section 3.1, the flow can remove temporal dependencies, simplifying the modeling task for the SLVM. With a single flow, the joint probability is

$$p_\theta(\mathbf{x}_{1:T}, \mathbf{z}_{1:T}) = p_\theta(\mathbf{y}_{1:T}, \mathbf{z}_{1:T}) \left| \det \left(\frac{\partial \mathbf{x}_{1:T}}{\partial \mathbf{y}_{1:T}} \right) \right|^{-1}, \quad (10)$$

where the SLVM distribution is given by $p_\theta(\mathbf{y}_{1:T}, \mathbf{z}_{1:T}) = \prod_{t=1}^T p_\theta(\mathbf{y}_t | \mathbf{y}_{<t}, \mathbf{z}_{\leq t}) p_\theta(\mathbf{z}_t | \mathbf{y}_{<t}, \mathbf{z}_{<t})$. If the SLVM is itself a flow-based model, we can use maximum log-likelihood training. If not, we can resort to variational inference [7, 14, 42]. We derive and discuss this procedure in the Appendix.

3.2.2 Latent Dynamics

We can also consider simplifying latent dynamics modeling using autoregressive flows. This is relevant in hierarchical SLVMs, such as VideoFlow [38], where each latent variable is modeled as a function of past and higher-level latent variables. Using $\mathbf{z}_t^{(\ell)}$ to denote the latent variable at the ℓ^{th} level at time t , we can parameterize the prior as

$$p_\theta(\mathbf{z}_t^{(\ell)} | \mathbf{z}_{<t}^{(\ell)}, \mathbf{z}_t^{(>\ell)}) = p_\theta(\mathbf{u}_t^{(\ell)} | \mathbf{u}_{<t}^{(\ell)}, \mathbf{z}_t^{(>\ell)}) \left| \det \left(\frac{\partial \mathbf{z}_t^{(\ell)}}{\partial \mathbf{u}_t^{(\ell)}} \right) \right|^{-1}, \quad (11)$$

where we convert $\mathbf{z}_t^{(\ell)}$ into $\mathbf{u}_t^{(\ell)}$ using the inverse transform $\mathbf{u}_t^{(\ell)} = (\mathbf{z}_t^{(\ell)} - \boldsymbol{\alpha}_\theta(\mathbf{z}_{<t}^{(\ell)})) / \boldsymbol{\beta}_\theta(\mathbf{z}_{<t}^{(\ell)})$. As noted previously, VideoFlow uses a special case of this procedure, setting $\boldsymbol{\alpha}_\theta(\mathbf{z}_{<t}^{(\ell)}) = \mathbf{z}_{t-1}^{(\ell)}$ and $\boldsymbol{\beta}_\theta(\mathbf{z}_{<t}^{(\ell)}) = \mathbf{1}$. Generalizing this procedure further simplifies dynamics throughout the model.

4 Evaluation

We demonstrate and evaluate the proposed technique on three benchmark video datasets: Moving MNIST [59], KTH Actions [57], and BAIR Robot Pushing [13]. Experimental setups are described in Section 4.1, followed by a set of analyses in Section 4.2. Further details and results can be found in the Appendix. Accompanying code is available in the supplementary material.

4.1 Experimental Setup

We empirically evaluate the improvements to downstream dynamics modeling from temporal pre-processing via autoregressive flows. For data space modeling, we compare four model classes: **1**) standalone affine autoregressive flows with one (AF-1) and **2**) two (AF-2) transforms, **3**) a sequential latent variable model (SLVM), and **4**) SLVM with flow-based pre-processing (SLVM + AF). As we are not proposing a specific architecture, but rather a general modeling technique, the SLVM architecture is representative of recurrent convolutional video models with a single latent level [9, 21, 22]. Flows

Figure 6: **Flow Visualization** for SLVM + 1-AF on Moving MNIST (left) and KTH Actions (right).

Figure 7: **Improved Generated Samples.** Random samples generated from (a) VideoFlow and (b) VideoFlow + AF, each conditioned on the first 3 frames. Using AF produces more coherent samples. The robot arm blurs for VideoFlow in samples 1 and 4 (red), but does not blur for VideoFlow + AF.

are implemented with convolutional networks, taking in a fixed window of previous frames (Fig. 4b). These models allow us to evaluate the benefits of temporal pre-processing (SLVM vs. SLVM + AF) and the benefits of more expressive higher-level dynamics models (AF-2 vs. SLVM + AF).

To evaluate latent dynamics modeling with autoregressive flows, we use the `tensor2tensor` library [63] to compare 1) VideoFlow¹ and 2) the same model with affine autoregressive flow latent dynamics (VideoFlow + AF). VideoFlow is significantly larger ($3\times$ more parameters) than the one-level SLVM, allowing us to evaluate whether autoregressive flows are beneficial in this high-capacity regime.

To enable a fairer comparison in our experiments, models with autoregressive flow dynamics have comparable or fewer parameters than baseline counterparts. We note that autoregressive dynamics adds only a constant computational cost per time-step, and this computation can be parallelized for training and evaluation. Full architecture, training, and analysis details can be found in the Appendix. Finally, as noted by Kumar et al. [38], many previous works do not train SLVMs with proper log-likelihood objectives. Our SLVM results are consistent with previously reported log-likelihood values [42] for the Stochastic Video Generation model [9] trained with a log-likelihood bound objective.

4.2 Analyses

Visualization In Figure 1, we visualize the pre-processing procedure for SLVM + 1-AF on BAIR Robot Pushing. The plots show the RGB values for a pixel before (top) and after (bottom) the transform. The noise sequence is nearly zero throughout, despite large changes in the pixel value. We also see that the noise sequence (center, lower) is invariant to the static background, capturing the moving robotic arm. At some time steps (e.g. fourth frame), the autoregressive flow incorrectly predicts the next frame, however, the higher-level SLVM compensates for this prediction error.

We also visualize each component of the flow. Figure 4b illustrates this for SLVM + 1-AF on an input from BAIR Robot Pushing. We see that μ_θ captures the static background, while σ_θ highlights regions of uncertainty. In Figure 6 and the Appendix, we present visualizations on full sequences, where we see that different models remove varying degrees of temporal structure.

¹We used a smaller version of the original model architecture, with half of the flow depth, due to GPU memory constraints.

Table 1: **Quantitative Comparison.** Average test negative log-likelihood (lower is better) in *nats per dimension* for Moving MNIST, BAIR Robot Pushing, and KTH Actions.

	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 + 1-AF	$\leq \mathbf{1.86}$	$\leq \mathbf{2.35}$	$\leq \mathbf{2.39}$
VideoFlow	–	1.53	–
VideoFlow + AF	–	1.50	–

Figure 8: **Improved Generalization.** The low-level reference frame improves generalization to unseen sequences. Train and test negative log-likelihood bound histograms for (a) SLVM and (b) SLVM + 1-AF on KTH Actions. (c) The generalization gap for SLVM + 1-AF remains small for varying amounts of KTH training data, while it becomes worse in the low-data regime for SLVM.

Temporal Redundancy Reduction To quantify temporal redundancy reduction, we evaluate the empirical correlation (linear dependence) between frames, denoted as corr , for the data and noise variables. We evaluate corr_x and corr_y for SLVM + 1-AF and 1-AF. The results are shown in Figure 5a. In Figure 5b, we plot corr_y for SLVM + 1-AF during training on KTH Actions. Flows decrease temporal correlation, and base distributions without temporal structure (1-AF) yield comparatively more decorrelation. Temporal redundancy is progressively removed throughout training. However, due to the limited capacity of the flows, they are not capable of completely removing temporal correlations on the more complex datasets, necessitating the use of higher-level dynamics models.

Performance Comparison Table 1 reports average test negative log-likelihood results. Standalone flow-based models perform surprisingly well. Increasing flow depth from AF-1 to AF-2 generally results in improvement. SLVM + 1-AF outperforms the baseline SLVM despite having *fewer* parameters. Incorporating autoregressive flows into VideoFlow results in a modest but noticeable improvement, demonstrating that removing spatial dependencies, through VideoFlow, and temporal dependencies, through autoregressive flows, are complementary techniques.

Improved Samples The quantitative improvement over VideoFlow is less dramatic, as this is already a high-capacity model. However, qualitatively, we observe that incorporating autoregressive flow dynamics improves sample quality (Figure 15). In these randomly selected samples, the robot arm occasionally becomes blurry for VideoFlow but remains clear for VideoFlow + AF.

Improved Generalization Our temporal normalization technique also improves generalization to unseen examples, a key benefit of normalization schemes, e.g. batch norm [29]. Intuitively, higher-level dynamics are often preserved, whereas lower-level appearance is not. This is apparent on KTH Actions, which contains a substantial degree of train-test mismatch (different identities and activities). NLL histograms on KTH are shown in Figure 8, with greater overlap for SLVM + 1-AF. We also train SLVM and SLVM + 1-AF on subsets of KTH Actions. In Figure 8c, we see that autoregressive flows enable generalization in the low-data regime, whereas SLVM becomes worse.

5 Conclusion

We have presented a technique for improving sequence modeling using autoregressive flows. Learning a frame of reference, parameterized by autoregressive transforms, reduces temporal redundancy in input sequences, simplifying dynamics. Thus, rather than expanding the model, we can simplify the input to meet the capacity of the model. This approach is distinct from previous works with normalizing flows on sequences, yet contains connections to classical modeling and compression. We hope these connections lead to further insights and applications. Finally, we have analyzed and empirically shown how autoregressive pre-processing in both the data and latent spaces can improve sequence modeling and lead to improved sample quality and generalization.

References

- [1] Siddharth Agrawal and Ambedkar Dukkipati. Deep variational inference without pixel-wise reconstruction. *arXiv preprint arXiv:1611.05209*, 2016.
- [2] Horace B Barlow et al. Possible principles underlying the transformation of sensory messages. *Sensory communication*, 1:217–234, 1961.
- [3] Justin Bayer and Christian Osendorfer. Learning stochastic recurrent networks. In *NeurIPS 2014 Workshop on Advances in Variational Inference*, 2014.
- [4] 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.
- [5] Scott Saobing Chen and Ramesh A Gopinath. Gaussianization. In *Advances in neural information processing systems*, pages 423–429, 2001.
- [6] 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.
- [7] 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.
- [8] Gustavo Deco and Wilfried Brauer. Higher order statistical decorrelation without information loss. In *Advances in Neural Information Processing Systems*, pages 247–254, 1995.
- [9] Emily Denton and Rob Fergus. Stochastic video generation with a learned prior. In *International Conference on Machine Learning*, pages 1182–1191, 2018.
- [10] Laurent Dinh, David Krueger, and Yoshua Bengio. Nice: Non-linear independent components estimation. In *International Conference on Learning Representations*, 2015.
- [11] Laurent Dinh, Jascha Sohl-Dickstein, and Samy Bengio. Density estimation using real nvp. In *International Conference on Learning Representations*, 2017.
- [12] Conor Durkan, Artur Bekasov, Iain Murray, and George Papamakarios. Neural spline flows. *arXiv preprint arXiv:1906.04032*, 2019.
- [13] Frederik Ebert, Chelsea Finn, Alex X Lee, and Sergey Levine. Self-supervised visual planning with temporal skip connections. In *Conference on Robot Learning*, 2017.
- [14] Marco Fraccaro, Søren Kaae Sønderby, Ulrich Paquet, and Ole Winther. Sequential neural models with stochastic layers. In *Advances in neural information processing systems*, pages 2199–2207, 2016.
- [15] 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.
- [16] Jerome H Friedman. Exploratory projection pursuit. *Journal of the American statistical association*, 82(397):249–266, 1987.

- [17] Karl Friston. Hierarchical models in the brain. *PLoS computational biology*, 4(11):e1000211, 2008.
- [18] Zhe Gan, Chunyuan Li, Ricardo Henao, David E Carlson, and Lawrence Carin. Deep temporal sigmoid belief networks for sequence modeling. In *Advances in Neural Information Processing Systems*, 2015.
- [19] 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.
- [20] Alex Graves. Generating sequences with recurrent neural networks. *arXiv preprint arXiv:1308.0850*, 2013.
- [21] David Ha and Jürgen Schmidhuber. Recurrent world models facilitate policy evolution. In *Advances in Neural Information Processing Systems*, pages 2450–2462, 2018.
- [22] Danijar Hafner, Timothy Lillicrap, Ian Fischer, Ruben Villegas, David Ha, Honglak Lee, and James Davidson. Learning latent dynamics for planning from pixels. In *International Conference on Machine Learning*, pages 2555–2565, 2019.
- [23] 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.
- [24] Gustav Eje Henter, Simon Alexanderson, and Jonas Beskow. Moglow: Probabilistic and controllable motion synthesis using normalising flows. *arXiv preprint arXiv:1905.06598*, 2019.
- [25] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.
- [26] Chin-Wei Huang, David Krueger, Alexandre Lacoste, and Aaron Courville. Neural autoregressive flows. In *International Conference on Machine Learning*, pages 2083–2092, 2018.
- [27] Chin-Wei Huang, Ahmed Touati, Laurent Dinh, Michal Drozdal, Mohammad Havaei, Laurent Charlin, and Aaron Courville. Learnable explicit density for continuous latent space and variational inference. *arXiv preprint arXiv:1710.02248*, 2017.
- [28] Aapo Hyvärinen and Erkki Oja. Independent component analysis: algorithms and applications. *Neural networks*, 13(4-5):411–430, 2000.
- [29] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International Conference on Machine Learning*, pages 448–456, 2015.
- [30] Priyank Jaini, Kira A Selby, and Yaoliang Yu. Sum-of-squares polynomial flow. In *International Conference on Machine Learning*, pages 3009–3018, 2019.
- [31] 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.
- [32] Maximilian Karl, Maximilian Soelch, Justin Bayer, and Patrick van der Smagt. Deep variational bayes filters: Unsupervised learning of state space models from raw data. In *International Conference on Learning Representations*, 2017.
- [33] 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.
- [34] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.
- [35] Diederik P Kingma and Max Welling. Stochastic gradient vb and the variational auto-encoder. In *Proceedings of the International Conference on Learning Representations*, 2014.

- [36] Durk P Kingma and Prafulla Dhariwal. Glow: Generative flow with invertible 1x1 convolutions. In *Advances in Neural Information Processing Systems*, pages 10215–10224, 2018.
- [37] 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.
- [38] 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.
- [39] Valero Laparra, Gustavo Camps-Valls, and Jesús Malo. Iterative gaussianization: from ica to random rotations. *IEEE transactions on neural networks*, 22(4):537–549, 2011.
- [40] Yingzhen Li and Stephan Mandt. A deep generative model for disentangled representations of sequential data. In *International Conference on Machine Learning*, 2018.
- [41] Salvator Lombardo, Jun Han, Christopher Schroers, and Stephan Mandt. Deep generative video compression. In *Advances in Neural Information Processing Systems*, pages 9283–9294, 2019.
- [42] 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.
- [43] Kevin P Murphy. *Machine learning: a probabilistic perspective*. MIT press, 2012.
- [44] 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.
- [45] BM Oliver. Efficient coding. *The Bell System Technical Journal*, 31(4):724–750, 1952.
- [46] George Papamakarios, Theo Pavlakou, and Iain Murray. Masked autoregressive flow for density estimation. In *Advances in Neural Information Processing Systems*, pages 2338–2347, 2017.
- [47] 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.
- [48] Mohsen Pourahmadi. Covariance estimation: The glm and regularization perspectives. *Statistical Science*, pages 369–387, 2011.
- [49] 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.
- [50] Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv preprint arXiv:1511.06434*, 2015.
- [51] Danilo Rezende and Shakir Mohamed. Variational inference with normalizing flows. In *International Conference on Machine Learning*, pages 1530–1538, 2015.
- [52] 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.
- [53] Nicholas Rhinehart, Kris M Kitani, and Paul Vernaza. R2p2: A reparameterized pushforward policy for diverse, precise generative path forecasting. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 772–788, 2018.
- [54] Nicholas Rhinehart, Rowan McAllister, Kris Kitani, and Sergey Levine. Precog: Prediction conditioned on goals in visual multi-agent settings. In *Proceedings of the International Conference on Computer Vision (ICCV)*, 2019.
- [55] Oren Rippel and Ryan Prescott Adams. High-dimensional probability estimation with deep density models. *arXiv preprint arXiv:1302.5125*, 2013.

- [56] Florian Schmidt, Stephan Mandt, and Thomas Hofmann. Autoregressive text generation beyond feedback loops. In *Empirical Methods in Natural Language Processing*, pages 3391–3397, 2019.
- [57] Christian Schuldt, Ivan Laptev, and Barbara Caputo. Recognizing human actions: a local svm approach. In *International Conference on Pattern Recognition*, 2004.
- [58] Mandyam Veerambudi Srinivasan, Simon Barry Laughlin, and Andreas Dubs. Predictive coding: a fresh view of inhibition in the retina. *Proceedings of the Royal Society of London. Series B. Biological Sciences*, 216(1205):427–459, 1982.
- [59] Nitish Srivastava, Elman Mansimov, and Ruslan Salakhudinov. Unsupervised learning of video representations using lstms. In *International conference on machine learning*, pages 843–852, 2015.
- [60] Aaron van den Oord, Sander Dieleman, Heiga Zen, Karen Simonyan, Oriol Vinyals, Alex Graves, Nal Kalchbrenner, Andrew Senior, and Koray Kavukcuoglu. Wavenet: A generative model for raw audio. *arXiv preprint arXiv:1609.03499*, 2016.
- [61] Aaron van den Oord, Nal Kalchbrenner, and Koray Kavukcuoglu. Pixel recurrent neural networks. In *International Conference on Machine Learning*, pages 1747–1756, 2016.
- [62] 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.
- [63] Ashish Vaswani, Samy Bengio, Eugene Brevdo, Francois Chollet, Aidan N. Gomez, Stephan Gouws, Llion Jones, Łukasz Kaiser, Nal Kalchbrenner, Niki Parmar, Ryan Sepassi, Noam Shazeer, and Jakob Uszkoreit. Tensor2tensor for neural machine translation. *CoRR*, abs/1803.07416, 2018.
- [64] Christina Winkler, Daniel Worrall, Emiel Hoogeboom, and Max Welling. Learning likelihoods with conditional normalizing flows. *arXiv preprint arXiv:1912.00042*, 2019.
- [65] Tianfan Xue, Jiajun Wu, Katherine Bouman, and Bill Freeman. Visual dynamics: Probabilistic future frame synthesis via cross convolutional networks. In *Advances in Neural Information Processing Systems*, 2016.
- [66] Zachary Ziegler and Alexander Rush. Latent normalizing flows for discrete sequences. In *International Conference on Machine Learning*, pages 7673–7682, 2019.

A Lower Bound Derivation

Consider the model defined in Section 3.3, with the conditional likelihood parameterized with autoregressive flows. That is, we parameterize

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

yielding

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

The joint distribution over all time steps is then given as

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

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

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

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

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

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

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

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

Finally, in the filtering setting, we can rewrite the expectation, bringing it inside of the sum (see [19, 42]):

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

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.

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

B Experiment Details

B.1 Flow Architecture

The affine autoregressive flow architecture is shown in Figure 9. The shift and scale of the affine transform are conditioned on three previous inputs. For each flow, we first apply 4 convolutional layers with kernel size (3, 3), stride 1, and padding 1 on each conditioned observation, preserving the input shape. The outputs are concatenated along the channel dimension and go through another 4 convolutional layers 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 output shift and log-scale. We use ReLU non-linearities for all convolutional layers.

Figure 9: **Affine Autoregressive Flow Architecture.** Each blank white rectangle represents a single convolutional layer (see text). The three stacks of convolutional layers within the blue shaded region are shared across all three time steps. cat denotes channel-wise concatenation.

B.2 Sequential Latent Variable Model Architecture

For sequential latent variable models, we use a DCGAN [50] encoder architecture (Figure 10a), with 4 convolutional layers of kernel size (4, 4), stride 2, and padding 1 followed by another convolutional layer of kernel size (4, 4), stride 1, and no padding. The encoding is sent to one or two LSTM layers [25] followed by separate linear layers to output the mean and log-variance for $q_\phi(\mathbf{z}_t | \mathbf{x}_{\leq t}, \mathbf{z}_{<t})$. The architecture for the conditional prior, $p_\theta(\mathbf{z}_t | \mathbf{x}_{<t}, \mathbf{z}_{<t})$, shown in Figure 10b, contains two fully-connected layers, which take the previous latent variable as input, followed by one or two LSTM layers, and separate linear layers to output the mean and log-variance. The decoder architecture, shown in Figure 10c, mirrors the encoder architecture, using transposed convolutions. In SLVM, we use two LSTM layers for modeling the conditional prior and approximate posterior distributions, while in SLVM + 1-AF, we use a single LSTM layer for each. We use leaky ReLU non-linearities [?] for the encoder and decoder architectures and ReLU non-linearities in the conditional prior architecture.

B.3 VideoFlow Architecture

For VideoFlow experiments, we use the official code provided by [38] in the `tensor2tensor` repository [63]. Due to memory and computational constraints, we use a smaller version of the model architecture used by [38] for the BAIR Robot Pushing dataset. We change depth from 24 to 12 and `latent_encoder_width` from 256 to 128. This reduces the number of parameters from roughly 67 million to roughly 32 million. VideoFlow contains a hierarchy of latent variables, with the latent variable at level l at time t denoted as $\mathbf{z}_t^{(l)}$. The prior on this latent variable is denoted as $p_\theta(\mathbf{z}_t^{(l)} | \mathbf{z}_{<t}^{(l)}, \mathbf{z}_{>t}^{(>l)}) = \mathcal{N}(\mathbf{z}_t^{(l)}; \boldsymbol{\mu}_t^{(l)}, \text{diag}((\boldsymbol{\sigma}_t^{(l)})^2))$, where $\boldsymbol{\mu}_t^{(l)}$ and $\boldsymbol{\sigma}_t^{(l)}$ are functions of $\mathbf{z}_{<t}^{(l)}$ and $\mathbf{z}_{>t}^{(>l)}$. We note that [38] parameterize $\boldsymbol{\mu}_t^{(l)}$ as $\boldsymbol{\mu}_t^{(l)} = \mathbf{z}_{t-1}^{(l)} + \tilde{\boldsymbol{\mu}}_t^{(l)}$, where $\tilde{\boldsymbol{\mu}}_t^{(l)}$ is the function. [38] refer to this as `latent_skip`. This is already a special case of an affine autoregressive flow, with a hard-coded shift of $\mathbf{z}_{t-1}^{(l)}$ and a scale of 1. We parameterize an affine autoregressive flow at each latent level, with a shift, $\boldsymbol{\alpha}_t^{(l)}$, and scale, $\boldsymbol{\beta}_t^{(l)}$, which are function of $\mathbf{z}_{<t}^{(l)}$, using the same 5-block ResNet architecture as [38]. In practice, these functions are conditioned on the variables at the past three time steps. The affine autoregressive flow produces a new variable:

$$\mathbf{u}_t^{(l)} = \frac{\mathbf{z}_t^{(l)} - \boldsymbol{\alpha}_t^{(l)}}{\boldsymbol{\beta}_t^{(l)}},$$

Figure 10: **SLVM Architecture**. Diagrams are shown for the (a) approximate posterior, (b) conditional prior, and (c) conditional likelihood of the sequential latent variable model (for SLVM + 1-AF). 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. SLVM contains one additional LSTM layer in both the approximate posterior and conditional prior.

which we then model using the same prior distribution and architecture as [38]: $p_\theta(\mathbf{u}_t^{(l)} | \mathbf{z}_{<t}^{(l)}, \mathbf{z}_t^{(>l)}) = \mathcal{N}(\mathbf{u}_t^{(l)}; \boldsymbol{\mu}_t^{(l)}, \text{diag}((\boldsymbol{\sigma}_t^{(l)})^2))$, where $\boldsymbol{\mu}_t^{(l)}$ and $\boldsymbol{\sigma}_t^{(l)}$, again, are functions of $\mathbf{z}_{<t}^{(l)}$ (or, equivalently $\mathbf{u}_{<t}^{(l)}$ and $\mathbf{z}_t^{(>l)}$).

B.4 Training Set-Up

We use the Adam optimizer [34] 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 SLVM and 100,000 iterations for AF-1, AF-2 and SLVM + 1-AF. For BAIR Robot Pushing, we use a batch size of 8 and train for 200,000 iterations for all models. For KTH Actions, we use a batch size of 8 and train for 90,000 iterations for all models. Batch norm [29] is applied to all convolutional layers that do not output distribution or affine transform parameters. We randomly crop sequences of length 13 from all sequences and evaluate on the last 10 frames. For AF-2 models, we crop sequences of length 16 in order to condition both flows on three previous inputs. For VideoFlow experiments, we use the same hyper-parameters as [38] (with the exception of the two architecture changes mentioned above) and train for 100,000 iterations.

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 two LSTM layers for modeling the prior and approx. posterior distribution of the latent variable, while in SLVM + 1-AF, we use a single LSTM layer for each.

Model	1-AF	2-AF	SLVM	SLVM + 1-AF
Moving Mnist	343k	686k	11,302k	10,592k
BAIR Robot Pushing	363k	726k	11,325k	10,643k
KTH Action	343k	686k	11,302k	10,592k

B.5 Quantifying Decorrelation

To quantify the temporal redundancy reduction resulting from affine autoregressive pre-processing, we evaluate the empirical correlation between successive frames for the data observations and noise

variables, averaged over spatial locations and channels. 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

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

where the term inside the expectation is

$$\xi_{t,t+1}(i,j,k) \equiv \frac{(x_t^{(i,j,k)} - \mu^{(i,j,k)})(x_{t+1}^{(i,j,k)} - \mu^{(i,j,k)})}{(\sigma^{(i,j,k)})^2}. \quad (23)$$

Here, $x_t^{(i,j,k)}$ denotes the image at location (i,j) and channel k at time t , $\mu^{(i,j,k)}$ is the mean of this dimension, and $\sigma^{(i,j,k)}$ is the standard deviation. H , W , and C respectively denote the height, width, and number of channels of the observations, and $\mathcal{D}$ denotes the dataset. We define an analogous expression for $\mathbf{y}$, denoted $\text{corr}_{\mathbf{y}}$.

C Additional Experimental Results

Table 3: **Training Quantitative Comparison.** Average training negative log-likelihood in *nats per dim.* for Moving MNIST, BAIR Robot Pushing, and KTH Actions.

	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 + 1-AF	$\leq \mathbf{1.85}$	$\leq \mathbf{2.31}$	$\leq \mathbf{2.21}$
VF	–	1.50	–
VF + AF	–	$\mathbf{1.49}$	–

C.1 Additional Qualitative Results

Figure 11: **Autoregressive Flow Visualization on KTH Action.** Visualization of the flow component for (a) standalone flow-based models and (b) sequential latent variable models with flow-based conditional likelihoods for KTH Actions. From top to bottom, each figure shows 1) the original frames, $\mathbf{x}_t$, 2) the predicted shift, $\mu_\theta(\mathbf{x}_{<t})$, for the frame, 3) the predicted scale, $\sigma_\theta(\mathbf{x}_{<t})$, for the frame, and 4) the noise, $\mathbf{y}_t$, obtained from the inverse transform.

Figure 12: **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, $\mu_\theta^1(\mathbf{x}_{<t})$, for the frame, 3) the predicted scale, $\sigma_\theta^1(\mathbf{x}_{<t})$, for the frame. On the right side, from top to bottom, we have 1) the higher-level predicted shift, $\mu_\theta^2(\mathbf{x}_{<t})$, for the frame, 3) the predicted scale, $\sigma_\theta^2(\mathbf{x}_{<t})$, for the frame and 4) the noise, $\mathbf{y}_t$, obtained from the inverse transform.

Figure 13: **Generated Moving MNIST Samples.** Sample frame sequences generated from a 2-AF model.

Figure 14: **Generated BAIR Robot Pushing Samples.** Sample frame sequences generated from SLVM + 1-AF. Sequences remain relatively coherent throughout, but do not display large changes across frames.

(a) VideoFlow

(b) VideoFlow + AF

Figure 15: **VideoFlow Generated Samples.** Sample frame sequences generated from (a) VideoFlow and (b) VideoFlow + AF. Each sample is conditioned on the first 3 frames, which are the same across both figures. Conditioning frames were selected randomly. Qualitatively, adding autoregressive flows tends to produce more coherent sequences. Comparing across figures, we see that the robotic arm disappears for VideoFlow in sequences 1 and 4, but does not disappear in any sequence for VideoFlow + AF.