IMPROVING SEQUENTIAL LATENT VARIABLE MODELS WITH AUTOREGRESSIVE FLOWS

Anonymous authors
Paper under double-blind review

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

Data often contain sequential structure, providing a rich signal for learning models of the world. Such models are useful for learning self-supervised representations of sequences (Li & Mandt, 2018; Ha & Schmidhuber, 2018) and planning sequences of actions (Chua et al., 2018; Hafner et al., 2019). While sequential models have a longstanding tradition in probabilistic modeling (Kalman et al., 1960), it is only recently that improved computational techniques, primarily deep networks, have facilitated learning such models from high-dimensional data (Graves, 2013), particularly video and audio. Dynamics in these models typically contain a combination of stochastic and deterministic variables (Bayer & Osendorfer, 2014; Chung et al., 2015; Gan et al., 2015; Fraccaro et al., 2016), using simple distributions (e.g. Gaussian) to directly model the likelihood of data observations. However, attempting to capture all sequential dependencies with relatively unstructured dynamics may make it more difficult to learn such models. 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. However, by stacking additional transforms or latent variables on top, we can arrive at 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. In fact, as we discuss, this approach generalizes the technique of modeling temporal derivatives to simplify dynamics estimation (Friston, 2008).

We empirically demonstrate this approach, both with standalone autoregressive normalizing flows, as well as by incorporating these flows within more flexible sequential latent 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. In formulating this general technique for improving dynamics estimation in the framework of normalizing flows, we also help to contextualize previous work.
Consider modeling discrete sequences of observations, \( x_{1:T} \sim p_{\text{data}}(x_{1:T}) \), using a probabilistic model, \( p_\theta(x_{1:T}) \), with parameters \( \theta \). Autoregressive models (Frey et al., 1996; Bengio & Bengio, 2000) use the chain rule of probability to express the joint distribution over all time steps as the product of \( T \) conditional distributions. Because of the forward nature of the world, as well as for handling variable-length sequences, these models are often formulated in forward temporal order:

\[
p_\theta(x_{1:T}) = \prod_{t=1}^{T} p_\theta(x_t|x_{<t}).
\]

Each conditional distribution, \( p_\theta(x_t|x_{<t}) \), models the temporal dependence between time steps, i.e. a prediction of the future. For continuous variables, we often assume that each distribution takes a relatively simple form, such as a diagonal Gaussian density:

\[
p_\theta(x_t|x_{<t}) = \mathcal{N}(x_t; \mu_\theta(x_{<t}), \text{diag}(\sigma^2_\theta(x_{<t}))),
\]

where \( \mu_\theta(\cdot) \) and \( \sigma_\theta(\cdot) \) are functions denoting the mean and standard deviation, often sharing parameters over time steps. While these functions may take the entire past sequence of observations as input, e.g. through a recurrent neural network, they may also be restricted to a convolutional window (van den Oord et al., 2016a). Autoregressive models can also be applied to non-sequential data (van den Oord et al., 2016b), where they excel at capturing local dependencies. However, due to their restrictive distributional forms, such models often struggle to capture higher-level structure.

### 2.2 Autoregressive Latent Variable Models

Autoregressive models can be improved by incorporating latent variables, often represented as a corresponding sequence, \( z_{1:T} \). Classical examples include Gaussian state space models and hidden Markov models (Murphy, 2012). The joint distribution, \( p_\theta(x_{1:T}, z_{1:T}) \), has the following form:

\[
p_\theta(x_{1:T}, z_{1:T}) = \prod_{t=1}^{T} p_\theta(x_t|x_{<t}, z_{<t})p_\theta(z_t|x_{<t}, z_{<t}).
\]

Unlike the simple, parametric form in Eq. 2, evaluating \( p_\theta(x_t|x_{<t}) \) now requires integrating over the latent variables,

\[
p_\theta(x_t|x_{<t}) = \int p_\theta(x_t|x_{<t}, z_{<t})p_\theta(z_{<t}|x_{<t})dz_{<t},
\]

yielding a more flexible distribution. However, performing this integration in practice is typically intractable, requiring approximate inference techniques, like variational inference (Jordan et al.,
an invertible transform, resulting in a normalizing flow. Kingma et al. (2016) noted that sampling from an autoregressive Gaussian model is possible in static settings. Here, we review this basic concept, continuing with the perspective of temporal conditioning handled through deterministic recurrent networks and the stochastic latent space. Our approach is based on affine autoregressive normalizing flows (Kingma et al., 2016; Papamakarios et al., 2017). 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, \( p_\theta(x_1|x_{<1}) \) (Eq. 2), using the reparameterization trick (Kingma & Welling, 2014; Rezende et al., 2014):

\[
x_t = \mu_\theta(x_{<t}) + \sigma_\theta(x_{<t}) \odot y_t,
\]

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

\[
y_t = \frac{x_t - \mu_\theta(x_{<t})}{\sigma_\theta(x_{<t})},
\]

where division is performed element-wise. The inverse transform in Eq. 6 acts to normalize (hence, normalizing flow) and therefore decorrelate \( x_{1:T} \). Given the functional mapping between \( y_t \) and \( x_t \) in Eq. 5, the change of variables formula converts between probabilities in each space:

\[
\log p_\theta(x_{1:T}) = \log p_\theta(y_{1:T}) - \log \left| \det \frac{\partial y_{1:T}}{\partial x_{1:T}} \right|.
\]

By the construction of Eqs. 5 and 6, the Jacobian in Eq. 7 is triangular, enabling efficient evaluation as the product of diagonal terms:

\[
\log \left| \det \frac{\partial y_{1:T}}{\partial x_{1:T}} \right| = \sum_{i=1}^{T} \sum_{i} \log \sigma_{\theta,i}(x_{<i}),
\]

where \( i \) denotes the observation dimension, e.g. pixel. For a Gaussian autoregressive model, \( p_\theta(y_{1:T}) = \mathcal{N}(y_{1:T}; 0, I) \). With these components, the change of variables formula (Eq. 7) provides an equivalent method for sampling and evaluating the model, \( p_\theta(x_{1:T}) \), from Eqs. 1 and 2.

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

\[
\log p_\theta(x_{1:T}) = \log p_\theta(y_{M,1:T}) - \log \left| \det \frac{\partial y_{M,1:T}}{\partial x_{1:T}} \right| - \sum_{m=1}^{M-1} \log \left| \det \frac{\partial y_{m+1,T}}{\partial y_{m,T}} \right|.
\]

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 & 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.
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.

Finally, we utilize affine flows (Eq. 5) in this work. 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 & 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.

3 Method

We now describe our approach for sequence modeling with autoregressive flows. Although the core idea is a relatively straightforward extension of autoregressive flows, we show how this simple technique can be incorporated within autoregressive latent variable models (Section 2.2), providing a general-purpose approach for improving dynamics modeling. We first motivate the benefits of affine autoregressive transforms in the context of sequence modeling with a simple example.

3.1 A Motivating Example

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

\[ x_t = x_{t-1} + u_t, \]
\[ u_t = u_{t-1} + w_t, \]

where \( w_t \sim \mathcal{N}(w_t; 0, \Sigma) \). We can express \( x_t \) and \( u_t \) in probabilistic terms as

\[ x_t \sim \mathcal{N}(x_t; x_{t-1} + u_{t-1}, \Sigma), \]
\[ u_t \sim \mathcal{N}(u_t; u_{t-1}, \Sigma). \]

Physically, this describes the noisy dynamics of a particle with momentum and mass 1, subject to Gaussian noise. That is, \( x \) represents position, \( u \) represents velocity, and \( w \) represents stochastic forces. If we consider the dynamics at the level of \( x \), we can use the fact that \( u_{t-1} = x_{t-1} - x_{t-2} \) to write

\[ p(x_t|x_{t-1}, x_{t-2}) = \mathcal{N}(x_t; x_{t-1} + x_{t-1} - x_{t-2}, \Sigma). \]
Thus, we see that in the space of $x$, the dynamics are second-order Markov, requiring knowledge of the past two time steps. However, at the level of $u$ (Eq. 13), the dynamics are first-order Markov, requiring only the previous time step. Yet, note that $u_t$ is, in fact, an affine autoregressive transform of $x_t$ because $u_t = x_t - x_{t-1}$ is a special case of the general form $rac{X_t - \mu_\theta(x_{t-1})}{\sigma_\theta(x_{t-1})}$. In Eq. 10, we see that the Jacobian of this transform is $\frac{\partial x_t}{\partial u_t} = I$, so, from the change of variables formula, we have $p(x_t|x_{t-1}, x_{t-2}) = p(u_t|u_{t-1})$. 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 $w_t = u_t - u_{t-1}$, we arrive at a representation that is entirely temporally decorrelated, i.e. no dynamics, because $p(w_t) = N(0; 0, \Sigma)$. A sample from this system is shown in Figure 2, illustrating this process of temporal decorrelation.

The special case of modeling temporal changes, $u_t = x_t - x_{t-1} = \Delta x_t$, is a common pre-processing technique; for recent examples, see Deisenroth et al. (2013); Chua et al. (2018); Kumar et al. (2019). In fact, $\Delta x_t$ is a finite differences approximation of the generalized velocity (Friston, 2008) of $x$, 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 temporally correlated raw signal.

### 3.2 Autoregressive Flows on Sequences

We apply autoregressive flows across time steps within a sequence, $x_{1:T} \in \mathbb{R}^{T \times D}$. That is, the observation at each time step, $x_t \in \mathbb{R}^D$, is modeled as an autoregressive function of past observations, $x_{<t} \in \mathbb{R}^{<1 \times D}$, and a random variable, $y_t \in \mathbb{R}^D$ (Figure 3a). We consider flows of the form given in Eq. 5, where $\mu_\theta(x_{<t})$ and $\sigma_\theta(x_{<t})$ are parameterized by neural networks. In constructing chains of flows, we denote the shift and scale functions at the $m^{th}$ transform as $\mu_m^{\theta}(\cdot)$ and $\sigma_m^{\theta}(\cdot)$ respectively. We then calculate $y^m_t$ using the corresponding inverse transform:

$$y^m_t = \frac{y^{m-1}_t - \mu^m_\theta(y^{m-1}_{<t})}{\sigma^m_\theta(y^{m-1}_{<t})}. \quad (15)$$

After the final ($M^{th}$) transform, the base distribution, $p_\theta(y^M_{1:T})$, can range from a simple distribution, e.g. $N(0; \mathbf{0}, \mathbf{I})$, in the case of a flow-based model, up to more complicated distributions in the case of other latent variable models (Section 3.3). While flows of greater depth can improve model capacity, such transforms have limiting drawbacks. In particular, 1) they require that the outputs maintain the same dimensionality as the inputs, $\mathbb{R}^{T \times D}$, 2) they are restricted to affine transforms, and 3) these transforms operate element-wise within a time step. As we discuss in the next section, we can combine autoregressive flows with non-invertible sequential latent variable models (Section 2.2), which do not have these restrictions.

### 3.3 Latent Variable Models with Autoregressive Flows

We use autoregressive flows as a component in parameterizing the dynamics within autoregressive latent variable models. To simplify notation, we consider this set-up with a single transform, but a chain of multiple transforms (Section 3.2) can be applied within each flow.

#### 3.3.1 Model Formulation

Let us consider parameterizing the conditional likelihood, $p_\theta(x_t|x_{<t}, z_{<t})$, within a latent variable model using an autoregressive flow (Figure 3b). To do so, we express a base conditional distribution for $y_t$, denoted as $p_\theta(y_t|y_{<t}, z_{<t})$, which is then transformed into $x_t$ via the affine transform in Eq. 5. We have written $p_\theta(y_t|y_{<t}, z_{<t})$ with conditioning on $y_{<t}$, however, by removing temporal correlations to arrive at $y_{1:T}$, our hope is that these dynamics can be primarily modeled through $z_{1:T}$. Using the change of variables formula, we can express the latent variable model’s log-joint distribution as

$$\log p_\theta(x_{1:T}, z_{1:T}) = \log p_\theta(y_{1:T}, z_{1:T}) - \log \left| \det \left( \frac{\partial x_{1:T}}{\partial y_{1:T}} \right) \right|, \quad (16)$$
where the joint distribution over $y_{1:T}$ and $z_{1:T}$, in general, is given as

$$p_\theta(y_{1:T}, z_{1:T}) = \prod_{t=1}^{T} p_\theta(y_t | y_{<t}, z_{<t}) p_\theta(z_t | y_{<t}, z_{<t}). \quad (17)$$

Note that the latent prior, $p_\theta(z_t | y_{<t}, z_{<t})$, can be equivalently conditioned on $x_{<t}$ or $y_{<t}$, as there is a one-to-one mapping between these variables. We could also consider parameterizing the prior with autoregressive flows, or even constructing a hierarchy of latent variables. However, we leave these extensions for future work, opting to first introduce the basic concept here.

### 3.3.2 Variational Inference & Learning

Training a latent variable model via maximum likelihood requires marginalizing over the latent variables to evaluate the marginal log-likelihood of observations: $\log p_\theta(x_{1:T}) = \log \int p_\theta(x_{1:T}, z_{1:T}) dz_{1:T}$. This marginalization is typically intractable, requiring the use of approximate inference methods. Variational inference (Jordan et al., 1998) introduces an approximate posterior distribution, $q(z_{1:T} | x_{1:T})$, which provides a lower bound on the marginal log-likelihood:

$$\log p_\theta(x_{1:T}) \geq \mathcal{L}(x_{1:T}; q, \theta) = \mathbb{E}_{q(z_{1:T} | x_{1:T})} \left[ \log p_\theta(x_{1:T}, z_{1:T}) - \log q(z_{1:T} | x_{1:T}) \right], \quad (18)$$

referred to as the evidence lower bound (ELBO). Often, we assume $q(z_{1:T} | x_{1:T})$ is a structured distribution, attempting to explicitly capture the model’s temporal dependencies across $z_{1:T}$. We can consider both filtering or smoothing inference, however, we focus on the case of filtering, with

$$q(z_{1:T} | x_{1:T}) = \prod_{t=1}^{T} q(z_t | x_{\leq t}, z_{<t}). \quad (19)$$

The conditional dependencies in $q$ can be modeled through a direct, amortized function, e.g. using a recurrent network (Chung et al., 2015), or through optimization (Marino et al., 2018). Again, note that we can condition $q$ on $x_{<t}$ or $y_{<t}$, as there exists a one-to-one mapping between these variables. With the model’s joint distribution (Eq. 16) and approximate posterior (Eq. 19), we can then evaluate the ELBO. We derive the ELBO for this set-up in Appendix A, yielding

$$\mathcal{L} = \sum_{t=1}^{T} \mathbb{E}_{q(z_{\leq t} | y_{\leq t})} \left[ \log p_\theta(y_t | y_{<t}, z_{\leq t}) - \log \frac{q(z_t | y_{\leq t}, z_{<t})}{p_\theta(z_t | y_{<t}, z_{<t})} - \log \left| \frac{\partial x_t}{\partial y_t} \right| \right]. \quad (20)$$

This expression makes it clear that a flow-based conditional likelihood amounts to learning a latent variable model on top of the intermediate learned space provided by $y$, with an additional factor in the objective penalizing the scaling between $x$ and $y$. 
Figure 4: **Flow Visualization.** Visualization of the flow component for (a), (c) standalone flow-based models and (b), (d) sequential latent variable models with flow-based conditional likelihoods for Moving MNIST and BAIR Robot Pushing. From top to bottom, each figure shows 1) the original frames, $x_t$, 2) the predicted shift, $\mu_{\theta}(x_{<t})$, for the frame, 3) the predicted scale, $\sigma_{\theta}(x_{<t})$, for the frame, and 4) the noise, $y_t$, obtained from the inverse transform.

## 4 Evaluation

We demonstrate and evaluate the proposed framework 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). Experimental setups are described in Section 4.1, followed by a set of qualitative experiments in Section 4.2. In Section 4.3, we provide quantitative comparisons across different model classes. Further implementation details and visualizations can be found in Appendix B. Anonymized code is available at the following link.

### 4.1 Experimental Setup

We implement three classes of models: 1) standalone autoregressive flow-based models, 2) sequential latent variable models, and 3) sequential latent variable models with flow-based conditional likelihoods. Flows are implemented with convolutional networks, taking in a fixed window of previous frames and outputting shift, $\mu_{\theta}$, and scale, $\sigma_{\theta}$, parameters. The sequential latent variable models consist of convolutional and recurrent networks for both the encoder and decoder networks, following the basic form of architecture that has been previously employed in video modeling (Denton & Fergus, 2018; Ha & Schmidhuber, 2018; Hafner et al., 2019).

In the case of a regular sequential latent variable model, the conditional likelihood is a Gaussian that models the frame, $x_t$. In the case of a flow-based conditional likelihood, we model the noise variable, $y_t$, with a Gaussian. In our experiments, the flow components have vastly fewer parameters than the sequential latent variable models. In addition, for models with flow-based conditional likelihoods, we restrict the number of parameters to enable a fairer comparison. These models have fewer parameters than the baseline sequential latent variable models (with non-flow-based conditional likelihoods). See Appendix B for parameter comparisons and architecture details. Finally, flow-based conditional likelihoods only add a constant computational cost per time-step, requiring a single forward pass per time step for both evaluation and generation.
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.

<table>
<thead>
<tr>
<th></th>
<th>M-MNIST</th>
<th>BAIR</th>
<th>KTH</th>
</tr>
</thead>
<tbody>
<tr>
<td>1-AF</td>
<td>−2.15</td>
<td>−3.05</td>
<td>−3.34</td>
</tr>
<tr>
<td>2-AF</td>
<td>−2.13</td>
<td>−2.90</td>
<td>−3.35</td>
</tr>
<tr>
<td>SLVM</td>
<td>≥−1.92</td>
<td>≥−3.57</td>
<td>≥−4.63</td>
</tr>
<tr>
<td>SLVM w/ 1-AF</td>
<td>≥−1.86</td>
<td>≥−2.35</td>
<td>≥−2.39</td>
</tr>
</tbody>
</table>

### 4.2 Qualitative Evaluation

To better understand the behavior of autoregressive flows on sequences, we visualize each component as an image. In Figure 4, we show the data, $x_t$, shift, $\mu_\theta$, scale, $\sigma_\theta$, and noise variable, $y_t$, for standalone flow-based models (left) and flow-based conditional likelihoods (right) on random sequences from the Moving MNIST and BAIR Robot Pushing datasets. Similar visualizations for KTH Actions are shown in Figure 8 in the Appendix. In Figure 9 in the Appendix, we also visualize these quantities for a flow-based conditional likelihood with two transforms.

From these visualizations, we can make a few observations. 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, as expected. 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 $y$, e.g. dots (Figure 4b, fourth row) or sharper edges (Figure 4d, fourth row). This is expected, as the noise distribution is more expressive in this case. With a relatively simple dataset, like Moving MNIST, a single flow can reasonably decorrelate the input, yielding white noise images (Figure 4a, fourth row). However, with natural image datasets like KTH Actions and BAIR Robot Pushing, a large degree of structure is still present in these images, motivating the use of additional model capacity to model this signal.

In Appendix C.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, $x$, and the noise variables, $y$. In Appendix C.2, we provide additional qualitative results.

### 4.3 Quantitative Evaluation

Log-likelihood 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 latent variable models with flow-based conditional likelihoods outperform their baseline counterparts, despite having fewer parameters. One reason for this disparity is overfitting. Comparing with the training performance reported in Table 3, we see that sequential latent variable models with flow-based conditional likelihoods overfit less. This is particularly apparent on KTH Actions, which contains training and test sets with a high degree of separation (different identities and activities). This suggests that removing static components, like backgrounds, yields a reconstruction space that is better for generalization.

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 & Fergus, 2018; Ha & Schmidhuber, 2018; Lee et al., 2018). Indeed, Marino et al. (2018) train SVG (Denton & 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


A Lower Bound Derivation

Consider the model defined in Section 3.3.1, with the conditional likelihood parameterized with autoregressive flows. That is, we parameterize

\[
x_t = \mu_\theta(x_{<t}) + \sigma_\theta(x_{<t}) \odot y_t
\]

yielding

\[
p_\theta(x_t | x_{<t}, z_{<t}) = p_\theta(y_t | y_{<t}, z_{<t}) \left| \det \left( \frac{\partial x_t}{\partial y_t} \right) \right|^{-1}
\]

(22)

The joint distribution over all time steps is then given as

\[
p_\theta(x_{1:T}, z_{1:T}) = \prod_{t=1}^{T} p_\theta(x_t | x_{<t}, z_{<t}) p_\theta(z_t | x_{<t}, z_{<t})
\]

(23)

\[
= \prod_{t=1}^{T} p_\theta(y_t | y_{<t}, z_{<t}) \left| \det \left( \frac{\partial x_t}{\partial y_t} \right) \right|^{-1} p_\theta(z_t | x_{<t}, z_{<t}).
\]

(24)

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

\[
q(z_{1:T} | x_{1:T}) = \prod_{t=1}^{T} q(z_t | x_{<t}, z_{<t}).
\]

(25)

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

\[
\mathcal{L} \equiv \mathbb{E}_{q(z_{1:T} | x_{1:T})} \left[ \log p_\theta(x_{1:T}, z_{1:T}) - \log q(z_{1:T} | x_{1:T}) \right]
\]

(26)

\[
= \mathbb{E}_{q(z_{1:T} | x_{1:T})} \left[ \log \left( \prod_{t=1}^{T} p_\theta(y_t | y_{<t}, z_{<t}) \right) \left| \det \left( \frac{\partial x_t}{\partial y_t} \right) \right|^{-1} p_\theta(z_t | x_{<t}, z_{<t}) \right]
\]

\[
- \log \left( \prod_{t=1}^{T} q(z_t | x_{<t}, z_{<t}) \right)
\]

(27)

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)):

\[
\mathcal{L} = \sum_{t=1}^{T} \mathbb{E}_{q(z_t | x_{<t})} \left[ \log p_\theta(y_t | y_{<t}, z_{<t}) - \log \frac{q(z_t | x_{<t}, z_{<t})}{p_\theta(z_t | x_{<t}, z_{<t})} - \log \left| \det \left( \frac{\partial x_t}{\partial y_t} \right) \right| \right].
\]

(28)

Because there exists a one-to-one mapping between \(x_{1:T}\) and \(y_{1:T}\), we can equivalently condition the approximate posterior and the prior on \(y\), i.e.

\[
\mathcal{L} = \sum_{t=1}^{T} \mathbb{E}_{q(z_t | y_{<t})} \left[ \log p_\theta(y_t | y_{<t}, z_{<t}) - \log \frac{q(z_t | y_{<t}, z_{<t})}{p_\theta(z_t | y_{<t}, z_{<t})} - \log \left| \det \left( \frac{\partial x_t}{\partial y_t} \right) \right| \right].
\]

(29)
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 & Schmidhuber, 1997) followed by fully connected layers to generate the mean and log-variance for estimating the approximate posterior distribution of the latent variable, $z_t$. 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 & Ba, 2014) with a learning rate of $1 \times 10^{-4}$ 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 & 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.) Anonymized code is available at the following link.

![Diagram of the autoregressive flow](image)

All convolution layers are set up with kernel size $(3,3)$ stride 1 and padding 1.

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. \texttt{conv} denotes a convolutional layer, \texttt{LSTM} denotes a long short-term memory layer, \texttt{fc} denotes a fully-connected layer, and \texttt{t.conv} denotes a transposed convolutional layer. For \texttt{conv} and \texttt{t.conv} layers, the numbers in parentheses respectively denote the number of filters, filter size, stride, and padding of the layer. For \texttt{fc} and \texttt{LSTM} layers, the number in parentheses denotes the number of units.

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.

<table>
<thead>
<tr>
<th>Model</th>
<th>1-AF</th>
<th>2-AF</th>
<th>SLVM</th>
<th>SLVM w/ 1-AF</th>
</tr>
</thead>
<tbody>
<tr>
<td>Moving Mnist</td>
<td>343k</td>
<td>686k</td>
<td>11302k</td>
<td>10592k</td>
</tr>
<tr>
<td>BAIR Robot Pushing</td>
<td>363k</td>
<td>726k</td>
<td>11325k</td>
<td>10643k</td>
</tr>
<tr>
<td>KTH Action</td>
<td>343k</td>
<td>686k</td>
<td>11302k</td>
<td>10592k</td>
</tr>
</tbody>
</table>

C **ADDITIONAL EXPERIMENTAL RESULTS**

C.1 **QUANTITATIVE EVALUATION OF TEMPORAL DECORRELATION**

The qualitative results in Figures 4 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

\[
\text{corr}_x \equiv \frac{1}{C \times W \times H} \sum_{i,j,k}^{H,W,C} \mathbb{E}_{x^{(i,j,k)}_t, x^{(i,j,k)}_{t+1}} \left[ \frac{(x^{(i,j,k)}_t - \mu^{(i,j,k)}_t)(x^{(i,j,k)}_{t+1} - \mu^{(i,j,k)}_{t+1})}{\sigma^{(i,j,k)}_t} \right], \quad (31)
\]

where \(x^{(i,j,k)}_t\) 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.

We evaluated this quantity for data examples, \(x\), and noise variables, \(y\), 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
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.

<table>
<thead>
<tr>
<th></th>
<th>M-MNIST</th>
<th>BAIR</th>
<th>KTH</th>
</tr>
</thead>
<tbody>
<tr>
<td>1-AF</td>
<td>-2.06</td>
<td>-2.98</td>
<td>-2.95</td>
</tr>
<tr>
<td>2-AF</td>
<td>-2.04</td>
<td>-2.76</td>
<td>-2.95</td>
</tr>
<tr>
<td>SLVM</td>
<td>≥ -1.93</td>
<td>≥ -3.46</td>
<td>≥ -3.05</td>
</tr>
<tr>
<td>SLVM w/ 1-AF</td>
<td>≥ -1.85</td>
<td>≥ -2.31</td>
<td>≥ -2.21</td>
</tr>
</tbody>
</table>

Table 4: **Temporal Correlation.** Temporal correlation (Eq. 31) between successive time steps for data observations, \( x \), and noise variables, \( y \), for SLVM w/ 1-AF.

<table>
<thead>
<tr>
<th></th>
<th>M-MNIST</th>
<th>BAIR</th>
<th>KTH</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \text{corr}_x )</td>
<td>0.24</td>
<td>0.87</td>
<td>0.96</td>
</tr>
<tr>
<td>( \text{corr}_y )</td>
<td>0.02</td>
<td>0.43</td>
<td>0.31</td>
</tr>
</tbody>
</table>

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

that the flows have resulted in simplified temporal structure. However, these results agree with the qualitative and quantitative results presented in Section 4, suggesting that autoregressive flows can yield sequences with simpler dynamics.
C.2 ADDITIONAL QUALITATIVE RESULTS

Figure 8: **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, $x_t$, 2) the predicted shift, $\mu_\theta(x_{<t})$, for the frame, 3) the predicted scale, $\sigma_\theta(x_{<t})$, for the frame, and 4) the noise, $y_t$, 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, $x_t$, 2) the lower-level predicted shift, $\mu_1^1(x_{<t})$, for the frame, 3) the predicted scale, $\sigma_1^1(x_{<t})$, for the frame. On the right side, from top to bottom, we have 1) the higher-level predicted shift, $\mu_2^2(x_{<t})$, for the frame, 3) the predicted scale, $\sigma_2^2(x_{<t})$, for the frame and 4) the noise, $y_t$, obtained from the inverse transform.
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.