Revisiting Structured Variational Autoencoders

Yixiu Zhao [1] **Scott W. Linderman** [2]

Abstract

Structured variational autoencoders (SVAEs) (Johnson et al., 2016) combine probabilistic graphical model priors on latent variables, deep neural networks to link latent variables to observed data, and structure-exploiting algorithms for approximate posterior inference. These models are particularly appealing for sequential data, where the prior can capture temporal dependencies. However, despite their conceptual elegance, SVAEs have proven difficult to implement, and more general approaches have been favored in practice. Here, we revisit SVAEs using modern machine learning tools and demonstrate their advantages over more general alternatives in terms of both accuracy and efficiency. First, we develop a modern implementation for hardware acceleration, parallelization, and automatic differentiation of the message passing algorithms at the core of the SVAE. Second, we show that by exploiting structure in the prior, the SVAE learns more accurate models and posterior distributions, which translate into improved performance on prediction tasks. Third, we show how the SVAE can naturally handle missing data, and we leverage this ability to develop a novel, self-supervised training approach. Altogether, these results show that the time is ripe to revisit structured variational autoencoders.

1. Introduction

Variational autoencoders (VAEs, Kingma and Welling, 2013; Rezende et al., 2014) are deep generative models that use neural networks to link latent variables to high dimensional observations. *Structured* variational autoencoders (SVAEs) (Johnson et al., 2016) use probabilistic graphical models to capture dependencies in the prior distribution over latent variables. For example, when working with sequential data, graphical models can capture latent dynamics (Krishnan et al., 2015; Archer et al., 2015). When domain knowledge is available, graphical models offer an expressive means of incorporating it (Pandarinath et al., 2018; Lopez et al., 2018). When inferring the posterior distribution over latent variables, graphical models offer an ancillary benefit: SVAEs can leverage the rich toolkit of message passing algorithms for graphical models (Wainwright et al., 2008) to aid in approximate inference.

Specifically, the SVAE leverages a structure-exploiting amortized variational inference algorithm. Rather than producing a full posterior distribution over latent variables, the recognition network (also referred to as an encoder) outputs *conjugate potentials*. When the prior is composed of exponential family distributions, the conjugate potentials can be combined with the prior using message passing algorithms to obtain an approximate posterior. Essentially, the SVAE learns to approximate complex deep generative models with simpler exponential family models where fast and efficient algorithms can be applied.

Despite their conceptual elegance, two issues have limited the adoption of SVAEs. The first is that the structure-exploiting inference algorithm is difficult to implement in practice, requiring gradients through message passing algorithms and fixed-point iterations. The second is a more basic question: what is the value of structure-exploiting amortized inference when we can construct arbitrarily flexible recognition networks? In practice, some methods retain a structured prior but use more general recognition networks (Krishnan et al., 2015; Dilokthanakul et al., 2016; Casale et al., 2018). Other methods use more flexible priors as well; e.g., sequential latent variable models often leverage recurrent neural network priors (e.g. Chung et al., 2015; Karl et al., 2016; Buesing et al., 2018; Pandarinath et al., 2018; Kosiorek et al., 2018; Hafner et al., 2019; Saxena et al., 2021)

In this work we address both of these concerns. First, we develop a modern implementation of the SVAE that is fast and scalable with CPU, GPU, or TPU hardware. We show how the SVAE can leverage parallel message passing algorithms to obtain orders of magnitude speedup over less structured alternatives. Second, we study SVAEs for sequential data

[1]Applied Physics Department, Stanford University [2]Department of Statistics and the Wu Tsai Neurosciences Institute, Stanford University. Correspondence to: Yixiu Zhao <yixiuz@stanford.edu>.

Proceedings of the 40^{th} International Conference on Machine Learning, Honolulu, Hawaii, USA. PMLR 202, 2023. Copyright 2023 by the author(s).

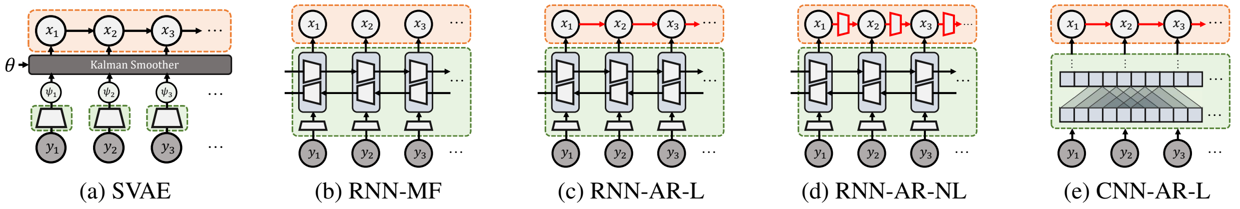


Figure 1: Different recognition network architectures for sequential VAEs. (a) In the structured variational autoencoder (SVAE), a neural network outputs recognition potentials ψ_t for each time step, which are combined with the prior parameters θ via a message passing algorithm (e.g. Kalman smoothing) to form the variational posterior $q(x_{1:T})$. Note that although ψ_t only depends on y_t, the posterior has correlations across time due to message passing. (b) In the mean field RNN recognition (RNN-MF) posterior family, a bi-directional RNN carries information forwards and backwards in time, generating a mean field variational posterior. (c-e) Temporal dependencies in the posterior can be captured with linear (c) or nonlinear (d) autoregressive dependencies that, again, depend on the outputs of a bi-directional RNN. This temporal dependency can also be approximated with a temporal CNN network with a finite kernel size (e).

and demonstrate many advantages over more commonly used alternatives. We show that structure-exploiting inference yields more accurate posterior approximations. These improvements lead to better model learning and more accurate prediction, especially with higher dimensional latent states and in lower signal-to-noise regimes. Finally, we highlight how SVAEs can naturally handle missing data, which motivates a novel, self-supervised training regimen to further improve model learning and predictive performance. With a modern implementation and multiple improvements, it is time to revisit structured variational autoencoders.[1]

2. Background

We begin with an introduction to structured variational autoencoders (SVAEs) (Johnson et al., 2016), with a particular emphasis on modeling sequential data. We then contrast the SVAE with more commonly used VAEs for sequential data, which use more general approaches to approximate posterior inference.

2.1. Structured Variational Autoencoders

The SVAE (Johnson et al., 2016) is both a modeling idea and an inference idea. The modeling idea has been independently proposed many times: to combine structured prior distributions on latent variables with complex dependencies implemented using deep neural networks (Khan and Lin, 2017; Archer et al., 2015; Klushyn et al., 2019; Kosiorek et al., 2018; Burgess et al., 2019; Greff et al., 2017; Van Steenkiste et al., 2018; Greff et al., 2019, e.g.). The inference idea is more unique. Like standard variational autoencoders (Kingma and Welling, 2013), the SVAE uses a recognition network to aid in inferring the latent variables. But rather than producing the full posterior, the SVAE recognition network only produces conjugate potentials. The conjugate potentials are combined with the prior, and the posterior is computed using classical message passing algorithms (Wainwright et al., 2008).

[1]Our implementation of the SVAE is available at https://github.com/lindermanlab/SVAE-Revisited.

For example, consider an SVAE with a linear Gaussian dynamical system prior (LDS-SVAE) for sequential data. The generative model specifies a joint distribution over a sequence of latent variables $x_{1:T}$ and emissions $y_{1:T}$,

$$p_{\theta,\gamma}(x_{1:T}, y_{1:T}) = p_\theta(x_1) \prod_{t=2}^{T} p_\theta(x_t \mid x_{t-1}) \prod_{t=1}^{T} p_\gamma(y_t \mid x_t),$$

where θ are the prior parameters and γ are the parameters of the decoder. The prior is a linear dynamical system,

$$p_\theta(x_1) = \mathcal{N}(x_1; \mu_1, Q_1),$$
$$p_\theta(x_t \mid x_{t-1}) = \mathcal{N}(x_t; A x_{t-1}, Q),$$

where $\theta = (\mu_1, Q_1, A, Q)$. However, the emission distribution, $p_\gamma(y_t \mid x_t) = \mathcal{N}(y_t; f_\gamma(x_t), R)$, may be nonlinear, since $f_\gamma(\cdot)$ can be parameterized with a neural network with weights γ.

The SVAE parameters are learned by maximizing an evidence lower bound (ELBO),

$$\mathcal{L}(\theta, \gamma, \phi) = \mathbb{E}_{q_{\theta,\phi}(x_{1:T}; y_{1:T})}\Big[\log p_{\theta,\gamma}(x_{1:T}, y_{1:T}) - \log q_{\theta,\phi}(x_{1:T}; y_{1:T})\Big] \leq \log p_{\theta,\gamma}(y_{1:T}),$$

where $q_{\theta,\phi}(x_{1:T}; y_{1:T})$ is a variational posterior distribution that is determined by both the variational parameters ϕ and the prior parameters θ.

The SVAE defines the variational posterior in a clever way. It defines the posterior as the implicit solution to a *surrogate variational inference problem*,

$$q_{\theta,\phi}(x_{1:T}; y_{1:T}) = \arg\min_{\tilde{q}\in\mathcal{Q}} \mathrm{KL}\left(\tilde{q}(x_{1:T}) \,||\, \tilde{p}_{\theta,\phi}(x_{1:T}; y_{1:T})\right).$$

The target, $\tilde{p}_{\theta,\phi}(x_{1:T}; y_{1:T})$, combines the prior with conjugate potentials, $\psi_\phi(x_t; y_t)$, from the recognition network,

$$\tilde{p}_{\theta,\phi}(x_{1:T}; y_{1:T}) \propto p_\theta(x_1)\prod_{t=2}^{T} p_\theta(x_t \mid x_{t-1})\prod_{t=1}^{T}\psi_\phi(x_t; y_t),$$
$$\psi_\phi(x_t; y_t) = \mathcal{N}(x_t; m_\phi(y_t), V_\phi(y_t)).$$

Here, $m_\phi(y_t)$ and $V_\phi(y_t)$ are the mean and covariance of a Gaussian potential output by a neural network with weights ϕ.

The key is that the surrogate inference problem is easy by design. Since the target is the product of a linear Gaussian prior and conjugate Gaussian potentials, the surrogate problem admits an exact solution via the Kalman smoother, a classic message passing algorithm. Figure 1a illustrates this structured-exploiting amortized inference approach.

The SVAE exploits structure in the prior distribution by incorporating it into the target and only learning to produce conjugate potentials. If the conjugate potentials are good approximations to the likelihoods, $p_\gamma(y_t \mid x_t)$, the solution to the surrogate problem will approximate the true posterior.

The principal challenge in implementing an SVAE — and we suspect the main reason why it has not been more widely adopted — is that maximizing the ELBO requires computing gradients of the implicitly defined variational posterior with respect to the prior and variational parameters. In the case of the linear Gaussian prior above, this amounts to back-propagating gradients through a Kalman smoother. More generally, it may require back-propagating gradients through inference algorithms like coordinate-ascent variational inference (Blei et al., 2017).

2.2. Alternative Inference Approaches for Sequential VAEs

A simpler and more commonly used approach is to have the recognition network output the full posterior directly. A canonical example of this approach is the deep Kalman filter (DKF) (Krishnan et al., 2015). DKFs are deep generative models for sequential data, allowing for nonlinear latent variable dynamics and emissions. For posterior inference, the key difference is that in a DKF, the encoder is a bidirectional recurrent neural network (RNN) that directly maps the data $y_{1:T}$ to a variational posterior,

$$q_\phi(x_{1:T}; y_{1:T}) = \prod_{t=1}^{T}\mathcal{N}(x_t; m_{\phi,t}, V_{\phi,t}),$$

where $\{m_{\phi,t}, V_{\phi,t}\}_{t=1}^{T}$ are the means and covariances output by a bidirectional RNN with weights ϕ running over the data $y_{1:T}$. We call this the **RNN-MF** posterior family (fig. 1b), since the RNN outputs a mean-field (i.e., independent) posterior.

It is straightforward to extend this family to include temporal dependencies,

$$q_\phi(x_{1:T}; y_{1:T}) = \mathcal{N}(x_1; m_{\phi,1}, V_{\phi,1})\prod_{t=2}^{T}\mathcal{N}(x_t; A_{\phi,t}x_{t-1} + m_{\phi,t}, V_{\phi,t}),$$

where $\{m_{\phi,t}, A_{\phi,t}, V_{\phi,t}\}_{t=1}^{T}$ are outputs of a bidirectional RNN with weights ϕ, and their dependence on the input data $y_{1:T}$ is omitted. We call this the **RNN-AR-L** posterior family (fig. 1c), since it extends the RNN-MF family with linear autoregressive dependencies. Though we do not know of examples where this family has been used, it is of theoretical interest for our experiments since it contains the true posterior for linear Gaussian state space models.

A more commonly used posterior approximation is,

$$q_\phi(x_{1:T}; y_{1:T}) = \mathcal{N}(x_1; m_{\phi,1}, V_{\phi,1}) \times \prod_{t=2}^{T}\mathcal{N}(x_t; g_\phi(x_{1:t-1}, u_{\phi,t}), Q_\phi(x_{1:t-1}, u_{\phi,t})),$$

where g and Q are neural networks, and $\{u_{\phi,t}\}_{t=1}^{T}$ are the outputs of a bidirectional RNN. This **RNN-AR-NL** posterior family (fig. 1d) extends the one above by allowing *nonlinear* autoregressive dependencies. For example, this style of posterior is used in LFADS (Pandarinath et al., 2018), a sequential VAE for modeling neural spike train data. A similar posterior family is employed by PlaNet (Hafner et al., 2019), a deep architecture for model-based planning from image data.

Of course, the RNNs could be replaced with other architectures as well, like convolutional neural networks (CNNs) (Bai et al., 2018), Transformers (Vaswani et al., 2017), or state space layers (Gu et al., 2021; Smith et al., 2022) for sequence-to-sequence mapping. As another baseline, we consider a CNN recognition network with fixed-sized convolution kernels over the time dimension, paired with the linear Gaussian posterior. We call this the **CNN-AR-L** posterior family (fig. 1e). While temporal CNNs are limited by the finite kernel size, they can still offer effective means of capturing temporal dependencies (Bai et al., 2018).

3. Modernizing the SVAE

We revisit the SVAE with three contributions that address the challenges that limited the original work:

- We provide a JAX implementation that's modular and easy to use.
- We leverage parallel Kalman filtering and smoothing for significant speed-ups on parallel hardware.
- We propose a simple self-supervised scheme for learning latent dynamics that plays into the strengths of the SVAE in handling missing data.

3.1. Efficient JAX Implementation

When the SVAE was first proposed, it did not see much practical usage because of its inherent complexity of implementation and the lack of hardware acceleration in the original implementation. With the advent of JAX (Bradbury et al., 2018), a Python library allowing automated compilation for efficient gradient computation through arbitrary compositions of functions, the barriers to implementing sophisticated algorithms like those in the SVAE are lowered significantly. Furthermore, the GPU and TPU support provided by JAX makes training more efficient. The JAX ecosystem provides libraries like JaxOpt (Blondel et al., 2022) for automatic *implicit* differentiation, and Dynamax (Chang et al., 2022) for message passing in probabilistic state space models. In this work we provide a JAX implementation of SVAEs for sequential data, enabling fast and flexible model building and training.

3.2. Parallel Message Passing

In this section we introduce a strategy for efficiently parallelizing SVAE inference with a linear Gaussian dynamical system prior that can also be applied more generally. In the LDS-SVAE described above, the variational posterior is obtained by a Kalman smoother. Standard implementations of the algorithm use a forward-backward recursion, with complexity that is linear in the length of the time series (Särkkä, 2013). However, this seemingly sequential algorithm can be efficiently parallelized by casting the process of forward filtering and backward smoothing/sampling as an associative operation on Gaussian distributions. We can leverage the prefix sum algorithm for parallel computational span that is only *logarithmic* in sequence length (Särkkä and García-Fernández, 2020).

To illustrate the idea behind this, consider the following linear Gaussian chain as a simple case:

$$x_1 \sim \mathcal{N}(b_1, Q_1), \quad x_t \mid x_{t-1} \sim \mathcal{N}(A_t x_{t-1} + b_t, Q_t).$$

To compute $p(x_T)$, note that the linear Gaussian model implies Gaussian marginals, $p(x_t) = \mathcal{N}(x_t; \mu_t, \Sigma_t)$ for some mean μ_t and covariance Σ_t. The naive idea would be to start from $\mu_1 = b_1$, $\Sigma_1 = Q_1$ and iterate over t, marginalizing the x_t's one at a time:

$$\mu_t = A_t \mu_{t-1} + b_t, \quad \Sigma_t = A_t \Sigma_{t-1} A_t^\top + Q_t,$$

which seems like an inherently sequential process. Writing out another iteration, we see more structure to the problem:

$$\begin{aligned} \mu_{t+1} &= A_{t+1}\mu_t + b_{t+1} \\ &= A_{[t,t+1]}\mu_{t-1} + b_{[t,t+1]} \\ \Sigma_{t+1} &= A_{t+1}\Sigma_t A_{t+1}^\top + Q_{t+1} \\ &= A_{[t,t+1]}\Sigma_{t-1} A_{[t,t+1]}^\top + Q_{[t,t+1]}, \end{aligned}$$

where $A_{[t,t+1]} = A_{t+1}A_t$, $b_{[t,t+1]} = A_{t+1}b_t + b_{t+1}$, and $Q_{[t,t+1]} = A_{t+1}Q_t A_{t+1}^\top + Q_{t+1}$.

The crucial observation is that this process of marginalization is *closed* and *associative*: the composition of two linear Gaussian conditional distributions gives another linear Gaussian conditional distribution, and the order of compositions does not affect the result. Instead of iterating on the means and covariances, we keep track of the parameters of the conditional distribution $a_t \equiv (A_t, b_t, Q_t)$, and define the following *associative operator* $\otimes$ on them:

$$(A_i, b_i, Q_i) \otimes (A_j, b_j, Q_j) \equiv (A_{[ij]}, b_{[ij]}, Q_{[ij]})$$
$$\text{where } \begin{cases} A_{[ij]} = A_j A_i \\ b_{[ij]} = A_j b_i + b_j \\ Q_{[ij]} = A_j Q_i A_j^\top + Q_j \end{cases}$$

Notice that here the subscripts represent subsequences of $(1, 2, \ldots, T)$, and we used $[ij]$ to represent a concatenation of subsequences i and j. It is easy to see that $\mu_t = b_{1:t}$ and $\Sigma_t = Q_{1:t}$. More importantly, the associative nature of the operation allows us to compute $a_{1:t}$ for all $t \in [1, T]$ in parallel, via the well-known prefix-sum algorithm (Ladner and Fischer, 1980). Dynamax offers convenient JAX implementations of parallel Kalman filtering and smoothing (Chang et al., 2022), enabling SVAE inference and training to have logarithmic parallel span. That is, the computation can be performed in $\mathcal{O}(\log T)$ time on a parallel machine.

3.3. Self-supervised Learning and Missing Data

As observed in previous works, learning accurate dynamics from high-dimensional observations is a non-trivial task for latent variable generative models (Karl et al., 2016; Klushyn et al., 2021). Practitioners often explicitly define losses based on current model extrapolations and true future observations. While this is an effective strategy, it is also a departure from the standard generative modeling approach.

To train a sequential generative model to learn accurate dynamics, we use a simple self-supervised training method inspired by (Kenton and Toutanova, 2019). During each training iteration, we apply a random mask to the input data that zeros out a continuous subsequence of observations. Since the model is required to reconstruct the original unmasked data, it must rely on its learned dynamics to fill in the missing observations.

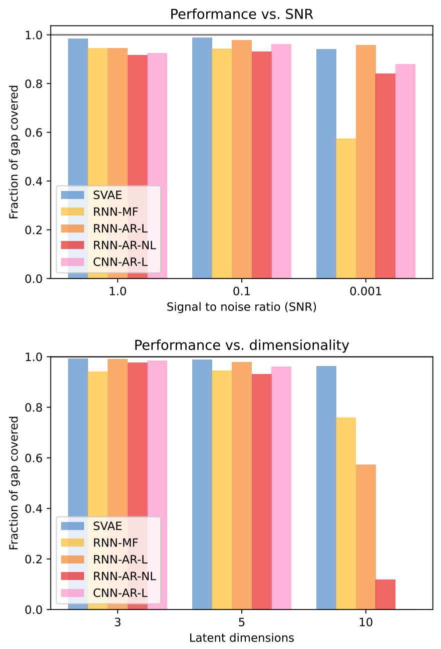


Figure 2: Top results on the LDS dataset. Performance between different methods is on par for most regimes, with the SVAE consistently achieving the best performance. Latent dimension size and SNR are 5 and 0.1 for the top and bottom plots respectively.

The SVAE has a unique advantage with this self-supervised scheme due to its structured approach to inference. Since each observation produces a separate conjugate potential, we can apply the mask by simply dropping the corresponding potentials (or equivalently, taking the potential covariance to infinity). This is a principled way to accommodate missing data in the SVAE and force it to rely on prior dynamics.

In practice, we found that masking out as much as 40% of the input sequence encourages learning of the dynamics. We train all SVAE instances with masks applied throughout training, and we will show it yields substantial increase in predictive performance. For RNN and CNN-based models, however, removing such a large chunk of data at the beginning of training can be disastrous, resulting in posterior collapse (Lucas et al., 2019a).

4. Results

4.1. Linear Dynamical System Dataset

We first test all of the inference architectures from Section 2 on toy datasets sampled from randomly generated linear dynamical systems. We test accuracy of the learned models under different state and observation dimensions, as well as different signal-to-noise (SNR) ratios.

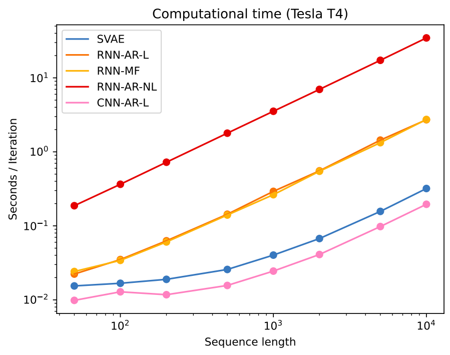


Figure 3: Wallclock time per iteration of the different methods on a Tesla T4 GPU. The SVAE with parallel scan achieves nearly ten-fold speedup to the faster RNN methods.

Dataset We consider the following linear dynamical system with stationary dynamics:

$$
\begin{aligned}
x_1 &\sim \mathcal{N}(0, Q_1), \\
x_t \mid x_{t-1} &\sim \mathcal{N}(Ax_{t-1} + b, Q), \\
y_t \mid x_t &\sim \mathcal{N}(Cx_t, R),
\end{aligned}
$$

for $t \in [1, T]$, where $x_t \in \mathbb{R}^D$ and $y_t \in \mathbb{R}^N$. Furthermore, we sample A as a rotation around a random axis in the D-dimensional latent space and set $Q = qI$, $R = rI$ to be diagonal covariance matrices.

We test the different recognition network architectures under three different noise scale combinations $(q, r) \in \{(0.1, 0.1), (0.1, 1.0), (0.01, 10.0)\}$ and three dimensionality settings $(D, N) \in \{(3, 5), (5, 10), (10, 20)\}$. We define the SNR as q/r.

For all of the parameter settings, the sequence length is 200 frames and we sample 100 sequences as the training data.

Model architecture For the SVAE, we use one linear layer with two linear readouts for the mean and covariance of the potential, since we know that the optimal recognition potential for LDS data is linearly related to the inputs. For the RNN models, we use various different recurrent state sizes $H \in \{10, 20, 30\}$, and we also add one to two hidden layers with ReLU nonlinearity to the output heads $A_{\phi,t}$, $b_{\phi,t}$ and $V_{\phi,t}$. For RNN-AR-NL we implement the nonlinear conditional distribution as a gated recurrent unit (GRU) cell. For CNN-AR-L, we use a 3-layer architecture with 32 features in each layer, and convolution kernels in the time dimension of sizes up to 50 timesteps. We use the same decoder network for all of the models, which is a one layer linear network with linear readouts for the output mean and covariance.

Null model To have a basis for comparison for the SVAE and RNN-based methods, we establish a static baseline model:

$$p_{\mathsf{st}}(x_{1:T}, y_{1:T}) = \prod_t p_{\hat{\theta}_t}(x_t) p_{\hat{\gamma}_t}(y_t \mid x_t),$$

where $p_{\hat{\theta}_t}(x_t) = \mathcal{N}(x_t; 0, \hat{Q}_t)$ is the marginal distribution of x_t and $p_{\hat{\gamma}_t}(y_t|x_t) = \mathcal{N}(y_t; Cx_t, R)$ is the emissions distribution under the true model. Since both terms are linear and Gaussian, the marginal distribution of the static model is $p_{\mathsf{st}}(y_{1:T}) = \prod_{t=1}^{T} \mathcal{N}(y_t; 0, \hat{Q} + CRC^{\top})$. This baseline is the best ELBO achievable by a "static" model, e.g. a VAE that treats each of the latent variables as independent across time. Intuitively, the noisier the observations, the harder it is for the models to learn the dynamics. Therefore, we expect the learned models to outperform the static baseline by a smaller margin in these regimes.

Results We perform hyperparameter search for all of the methods and show the best ELBO achieved in figure 2. We normalize the ELBO by looking at the percentage of the gap covered between the marginal data log likelihood under the null model $p_{\text{st}}(y_{1:T})$ and the true model $p_{\text{true}}(y_{1:T})$. Overall, the SVAE is the most consistent and achieves the highest ELBO. Unsurprisingly, we notice that low SNR and high dimensionality negatively impact model performance, resulting in decreased performance for all methods, with the SVAE being the least impacted. Note that these trends also scale higher dimensions, which we explore in Appendix A.

4.2. Computational Efficiency of SVAE with Parallel Kalman Filtering

Here we examine the practical benefits of the parallel Kalman filter in the SVAE. We run all of the methods on synthetic LDS data with three-dimensional latent states and five-dimensional observations. We scale up the sequence length and measure the wallclock time it takes for each method to complete an iteration over a batch of ten sequences on a Tesla T4 GPU. For the RNN and CNN-based methods, we use the same architecture as Section 4.1 with ten-dimensional latent states for the RNNs and a kernel size of 10 for the CNN.

The results are shown in figure 3. While we do not achieve the theoretical efficiency of $\mathcal{O}(\log T)$ in sequence length T due to practical limits on the amount of parallel resources units available, the SVAE still obtained almost ten-fold speedup over the mean-field and autoregressive RNN methods. Unsurprisingly, the nonlinear autoregressive RNN is the slowest, with an additional RNN for sequential sampling from the posterior alongside the sequential BiRNN inference network. The CNN-AR-L is slightly faster than the SVAE but follows the same trend, however with finite kernel size it cannot capture the same temporal dependencies.

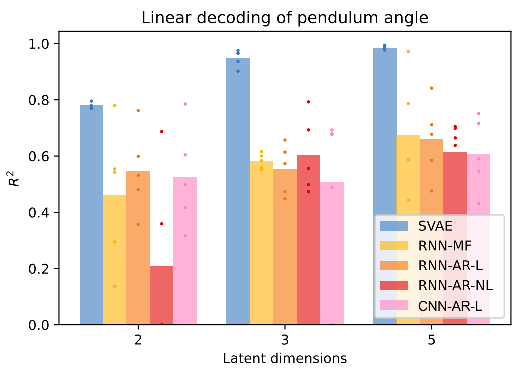
4.3. Synthetic Pendulum Dataset

It is perhaps no surprise that the SVAE performs well on data generated from the same prior class with linear emissions. In general, we are more interested in the value of SVAEs in more complex settings, where the emissions are high dimensional and nonlinear. Furthermore, it would be interesting to see the SVAE perform in a regime where the dynamics are *nonlinear*.

We apply the SVAE and the RNN methods to the synthetic pendulum dataset adapted from Schirmer et al. (2022). A 24×24 pixel movie of a swinging pendulum is rendered with noise, and the task is to learn the underlying low-dimensional dynamics in an unsupervised fashion. We take regularly spaced observations of the pendulum and apply uniform Gaussian noise stationary across time to each pixel. Since the true physical state (angle and angular momentum) of the pendulum can be described in a two-dimensional state space, we use models with small latent dimensionality $D \in (2, 3, 5)$. It is important to note that for most of this dataset the small-angle approximation does not apply, and therefore the underlying dynamics are nonlinear.

One challenge in learning dynamics is that angular velocity information is missing from the individual frames. Since the pendulum observations exist in a one-dimensional manifold, the models do not need dynamics information to be able to reconstruct the observations. To encourage the models to recover the underlying dynamics, we apply the self-supervised masking introduced in Section 3.3 to all of the models. We found that RNN-based recognition networks failed to learn accurate models when significant masking was used in early stages of training; thus, we gave these architectures full observations for the first 10-100 epochs before applying the masks. For the SVAE, we found it trained reliably even with masks applied from the start. This is likely due to the inference algorithm of the SVAE, which allows it to handle missing data in a principled way. We compare the results of the SVAE, RNN-AR-L and RNN-AR-NL on 100 sequences of pendulum renderings, each with 100 frames. We pick the top five runs in ELBO on the test set and analyze the learned models. The achieved ELBO values with and without the self supervised masking is reported in appendix B. The SVAE achieves comparable ELBO values to other methods, but its learned models yield better representations of the data dynamics, as we show next.

SVAE learns linearly decodable representations We find that the true physical states of the pendulum can be linearly decoded from the latent states inferred by the SVAE. Figure 4 shows R^2 values of a linear regression from the learned latents to true angles and angular velocities. Across different latent sizes, the SVAE admits representations that are more linearly related to the true physical states. Note



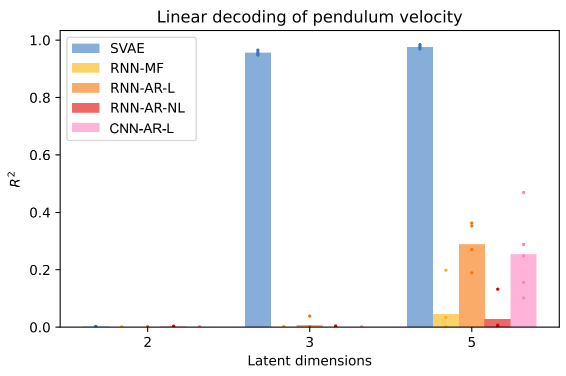


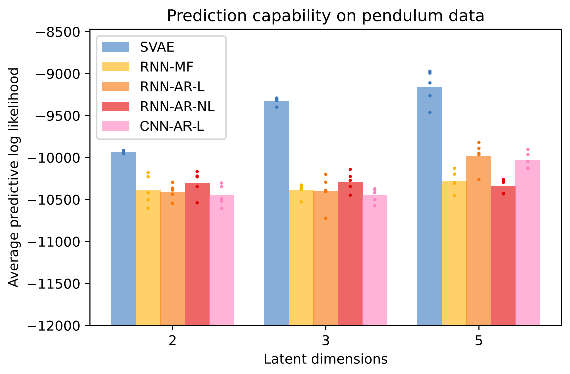
Figure 4: SVAE learns representations of pendulum data that are informative about the dynamics. We run linear regression from the latent representations to the true pendulum angles (left) and angular velocities (right) and report the R^2 values. Notably, the SVAE is the only method that learns a representation of angular velocity with 3 dimensional latents.

that none of the models are able to encode information about angular velocity given two-dimensional latents, while the SVAE is able to explain almost all of the variance in angular velocity given an additional dimension.

Model performance on prediction Given the first 50 frames of a test sequence, We forecast latent states with the learned dynamics model $p_\theta(x_{t+1} \mid x_t)$ and estimate the log marginal likelihood of true data $\log p(y_{51:100} \mid y_{1:50})$ by approximating the integral over future latent states with Monte Carlo. Across different latent space sizes, the SVAE yields higher likelihoods for future data (fig. 5). Given the linear decodability of the SVAE latents, we also visualize an example predicted trajectory of angles and angular velocities in figure 5.

We show sample predictions in the image space in figure 6. To show how the models represent uncertainty in their predictions, we sample 200 latent trajectories $\{x_{1:T}^{(i)}\}_{i=1}^{200}$ of length $T = 100$. We train on the first 50 time steps and forecast the last 50. Fig. 6 shows both the most accurate predictions and the average predicted frames. While the best prediction from the RNN-AR-L are sharp and reasonably accurate, the average prediction quickly becomes diffuse. The SVAE, on the other hand, is able to maintain its spread over angular positions throughout the sequence, demonstrating the accuracy of its learned dynamics model.

This pendulum example would not be possible with the existing, pure Python SVAE implementation, since it requires convolutional neural networks for generating the recognition potentials and implementing the decoder. With our JAX implementation, the model can be trained on a single GPU in tens of minutes. The promising results on this image forecasting task suggest that the SVAE can be a valuable model for more complex, high-dimensional time series as well.



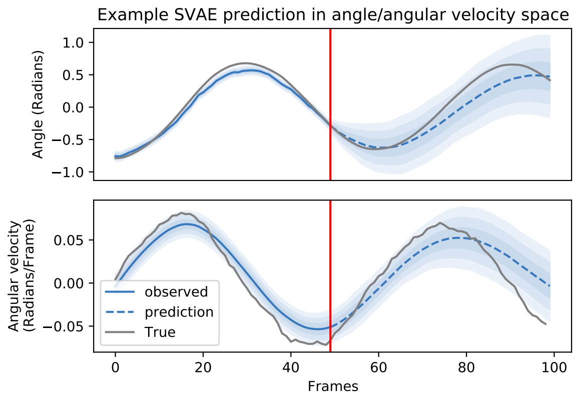


Figure 5: SVAE yields better predictive performance. We sample many latent trajectories from the learned dynamics and estimate the marginal log likelihood of the future 50 frames give the past 50 frames (top). Given the linear regression weights, we can make predictions in the phase space of the pendulum (bottom). Each shade in the plot corresponds to one standard deviation.

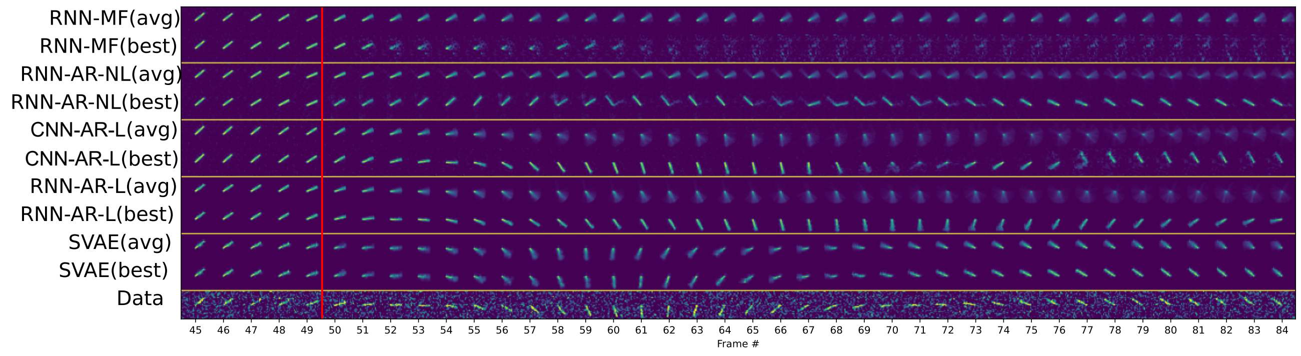


Figure 6: Sample model predictions for pendulum data in image space. For all of the models, we show both the average prediction (avg) and the best prediction (best) among the 200 sample trajectories. We see that the SVAE alone predicts pendulum angles that are concentrated around the true angles.

5. Related Works

In recent years, there has been a plethora of work on deep sequential LVMs, with applications ranging from speech modeling, and video compression to prediction and planning in real and simulated environments (Chung et al., 2015; Yingzhen and Mandt, 2018; Marino et al., 2018; Hafner et al., 2019; Babaeizadeh et al., 2018; Karl et al., 2016; Saxena et al., 2021). We focus on the LDS-SVAE since it is a simple yet interesting model that demonstrates the advantages of structure-exploiting inference for complex, high-dimensional datasets. It is worth noting that the SVAE also allows for more complex and expressive PGM priors such as the switching linear dynamical system (SLDS) (Murphy, 1998; Fox et al., 2008) and recurrent switching linear dynamical system (rSLDS) (Linderman et al., 2017) for the purposes of analyzing sequential data.

Amortization gap Amortized inference with inflexible encoder networks results in significant *amortization gaps* — the difference between the ELBO and the true marginal likelihood that arises from the recognition network outputting suboptimal variational parameters (Cremer et al., 2018; Marino et al., 2018). A failure mode that often accompanies this amortization gap is the posterior collapse, which happens when the model learns to ignore the latents completely (Alemi et al., 2018; Lucas et al., 2019b). Turner and Sahani (2011) studied the effects of approximation gap on model learning, and more recent work has considered this issue in the context of amortized variational inference (Cremer et al., 2018; Shu et al., 2018; He et al., 2019). The SVAE addresses this gap by incorporating the prior into the implicit definition of the posterior distribution.

Structure-exploiting recognition networks Some inference frameworks try to address the amortization gap issue by having the variational posterior depend on the prior parameters (Marino et al., 2018; Johnson et al., 2016; Lin et al., 2018; Tomczak and Welling, 2018). Amortized variational filtering (AVF) (Marino et al., 2018) uses inner-loop gradient updates to correct the variational posterior which accounts for the changing prior in an expectation-maximization-like fashion. Other approaches like the SVAE use structured exponential family priors and variational posteriors computed with an inner structured inference step involving the prior, offloading computation from the inference network. In this work we focus on the SVAE as one representative of structure-exploiting amortized inference and demonstrate how it performs more accurate inference and supports better parameter estimation over its non-structured counterparts. Interestingly, despite having the capacity to learn the true posterior in our linear dynamical systems experiments, we found that the RNN-based approaches failed to attain the true marginal log likelihood. In essence, it appears hard for RNNs to learn to perform Kalman smoothing with noisy, high-dimensional observations in the small data regimes that we consider.

Sequence modeling beyond RNNs In the past few years there has been a burst of new model and architectures that tackles sequence modeling. Transformers (Vaswani et al., 2017; Kenton and Toutanova, 2019; Brown et al., 2020) have become the new standard in language modeling, outperforming RNNs. For very long sequences, structured state space model for sequences (S4) (Gu et al., 2021) and its variants (Smith et al., 2022; Hasani et al., 2022; Gupta et al., 2022) are becoming more prominent in solving hard sequence tasks with high parallel efficiency. Notably, Zhou et al. (2022) uses S4 in place of the prior, recognition and decoder components of a sequential VAE. Although we included the CNN as a simple alternative to the RNN-based models, it would be natural to follow up with comparisons between SVAEs and these new types of sequence modeling techniques that have more complex architectures and stronger modeling power.

6. Conclusion

We revisit the structured variational autoencoder, a structure-exploiting amortized variational inference framework that combines the flexibility of neural networks with the modeling benefits of probabilistic graphical models. Our JAX implementation runs efficiently on modern hardware and leverages parallel Kalman filtering to achieve an order of magnitude speed-up over RNN-based unstructured inference methods.

We showed that the SVAE with linear Gaussian dynamics yields competitive performance on synthetic datasets. Not only does it perform fast and accurate inference in the case of well-specified models, it also yields superior predictive performance for high-dimensional observations with nonlinear underlying dynamics. With its advantage for handling missing data, we use self-supervised masking to encourage learning of dynamics. Compared to its RNN and CNN-based counterparts, the SVAE learns representations that are informative about the dynamics, and we were able to linearly decode the true states of the system from its latent states. These results show much promise for the SVAE in deep generative modeling tasks.

Acknowledgements

This work was supported by the Stanford Interdisciplinary Graduate Fellowship (SIGF), along with grants from the Simons Collaboration on the Global Brain (SCGB 697092), the NIH (U19NS113201, R01NS113119, and R01NS130789), the Sloan Foundation, and the Stanford Center for Human and Artificial Intelligence. We thank Matt Johnson for his constructive feedback and insightful discussions with us on the paper. Additionally, we thank Blue Sheffer and Dieterich Lawson for their contribution in the early stages of the project, Andy Warrington and Jimmy Smith for their extensive feedback in the paper writing process, and the rest of the Linderman Lab for their support and feedback throughout the project.

References

Matthew J Johnson, David K Duvenaud, Alex Wiltschko, Ryan P Adams, and Sandeep R Datta. Composing graphical models with neural networks for structured representations and fast inference. *Advances in Neural Information Processing Systems*, 29:2946–2954, 2016.

Diederik P Kingma and Max Welling. Auto-encoding variational Bayes. *arXiv preprint arXiv:1312.6114*, 2013.

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

Rahul G Krishnan, Uri Shalit, and David Sontag. Deep Kalman filters. *arXiv preprint arXiv:1511.05121*, 2015.

Evan Archer, Il Memming Park, Lars Buesing, John Cunningham, and Liam Paninski. Black box variational inference for state space models. *arXiv preprint arXiv:1511.07367*, 2015.

Chethan Pandarinath, Daniel J O'Shea, Jasmine Collins, Rafal Jozefowicz, Sergey D Stavisky, Jonathan C Kao, Eric M Trautmann, Matthew T Kaufman, Stephen I Ryu, Leigh R Hochberg, et al. Inferring single-trial neural population dynamics using sequential auto-encoders. *Nature Methods*, 15(10):805–815, 2018.

Romain Lopez, Jeffrey Regier, Michael B Cole, Michael I Jordan, and Nir Yosef. Deep generative modeling for single-cell transcriptomics. *Nature Methods*, 15(12): 1053–1058, 2018.

Martin J Wainwright, Michael I Jordan, et al. Graphical models, exponential families, and variational inference. *Foundations and Trends® in Machine Learning*, 1(1–2): 1–305, 2008.

Nat Dilokthanakul, Pedro AM Mediano, Marta Garnelo, Matthew CH Lee, Hugh Salimbeni, Kai Arulkumaran, and Murray Shanahan. Deep unsupervised clustering with Gaussian mixture variational autoencoders. *arXiv preprint arXiv:1611.02648*, 2016.

Francesco Paolo Casale, Adrian Dalca, Luca Saglietti, Jennifer Listgarten, and Nicolo Fusi. Gaussian process prior variational autoencoders. *Advances in Neural Information Processing Systems*, 31, 2018.

Junyoung Chung, Kyle Kastner, Laurent Dinh, Kratarth Goel, Aaron C Courville, and Yoshua Bengio. A recurrent latent variable model for sequential data. *Advances in Neural Information Processing Systems*, 28, 2015.

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*, 2016.

Lars Buesing, Theophane Weber, Sébastien Racaniere, SM Eslami, Danilo Rezende, David P Reichert, Fabio Viola, Frederic Besse, Karol Gregor, Demis Hassabis, et al. Learning and querying fast generative models for reinforcement learning. *arXiv preprint arXiv:1802.03006*, 2018.

Adam Kosiorek, Hyunjik Kim, Yee Whye Teh, and Ingmar Posner. Sequential attend, infer, repeat: Generative modelling of moving objects. *Advances in Neural Information Processing Systems*, 31, 2018.

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.

Vaibhav Saxena, Jimmy Ba, and Danijar Hafner. Clockwork variational autoencoders. *Advances in Neural Information Processing Systems*, 34:29246–29257, 2021.

Mohammad Khan and Wu Lin. Conjugate-computation variational inference: Converting variational inference in non-conjugate models to inferences in conjugate models. In *Artificial Intelligence and Statistics*, pages 878–887, 2017.

Alexej Klushyn, Nutan Chen, Richard Kurle, Botond Cseke, and Patrick van der Smagt. Learning hierarchical priors in VAEs. *Advances in Neural Information Processing Systems*, 32, 2019.

Christopher P Burgess, Loic Matthey, Nicholas Watters, Rishabh Kabra, Irina Higgins, Matt Botvinick, and Alexander Lerchner. Monet: Unsupervised scene decomposition and representation. *arXiv preprint arXiv:1901.11390*, 2019.

Klaus Greff, Sjoerd Van Steenkiste, and Jürgen Schmidhuber. Neural expectation maximization. *Advances in Neural Information Processing Systems*, 30, 2017.

Sjoerd Van Steenkiste, Michael Chang, Klaus Greff, and Jürgen Schmidhuber. Relational neural expectation maximization: Unsupervised discovery of objects and their interactions. *arXiv preprint arXiv:1802.10353*, 2018.

Klaus Greff, Raphaël Lopez Kaufman, Rishabh Kabra, Nick Watters, Christopher Burgess, Daniel Zoran, Loic Matthey, Matthew Botvinick, and Alexander Lerchner. Multi-object representation learning with iterative variational inference. In *International Conference on Machine Learning*, pages 2424–2433, 2019.

David M Blei, Alp Kucukelbir, and Jon D McAuliffe. Variational inference: A review for statisticians. *Journal of the American statistical Association*, 112(518):859–877, 2017.

Shaojie Bai, J Zico Kolter, and Vladlen Koltun. An empirical evaluation of generic convolutional and recurrent networks for sequence modeling. *arXiv preprint arXiv:1803.01271*, 2018.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in Neural Information Processing Systems*, 30, 2017.

Albert Gu, Karan Goel, and Christopher Re. Efficiently modeling long sequences with structured state spaces. In *International Conference on Learning Representations*, 2021.

Jimmy TH Smith, Andrew Warrington, and Scott W Linderman. Simplified state space layers for sequence modeling. *arXiv preprint arXiv:2208.04933*, 2022.

James Bradbury, Roy Frostig, Peter Hawkins, Matthew James Johnson, Chris Leary, Dougal Maclaurin, George Necula, Adam Paszke, Jake VanderPlas, Skye Wanderman-Milne, and Qiao Zhang. JAX: composable transformations of Python+NumPy programs, 2018. URL `http://github.com/google/jax`.

Mathieu Blondel, Quentin Berthet, Marco Cuturi, Roy Frostig, Stephan Hoyer, Felipe Llinares-López, Fabian Pedregosa, and Jean-Philippe Vert. Efficient and modular implicit differentiation. *Advances in Neural Information Processing Systems*, 35:5230–5242, 2022.

Peter Chang, Giles Harper-Donnelly, Aleyna Kara, Xinglong Li, Scott Linderman, and Kevin Murphy. Dynamax: a library for probabilistic state space models in jax, 2022. URL `http://github.com/probml/dynamax`.

Simo Särkkä. *Bayesian filtering and smoothing*. Number 3. Cambridge university press, 2013.

Simo Särkkä and Ángel F García-Fernández. Temporal parallelization of bayesian smoothers. *IEEE Transactions on Automatic Control*, 66(1):299–306, 2020.

Richard E Ladner and Michael J Fischer. Parallel prefix computation. *Journal of the ACM (JACM)*, 27(4):831–838, 1980.

Alexej Klushyn, Richard Kurle, Maximilian Soelch, Botond Cseke, and Patrick van der Smagt. Latent matters: Learning deep state-space models. *Advances in Neural Information Processing Systems*, 34:10234–10245, 2021.

Jacob Devlin Ming-Wei Chang Kenton and Lee Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of NAACL-HLT*, pages 4171–4186, 2019.

James Lucas, George Tucker, Roger Grosse, and Mohammad Norouzi. Understanding posterior collapse in generative latent variable models, 2019a.

Mona Schirmer, Mazin Eltayeb, Stefan Lessmann, and Maja Rudolph. Modeling irregular time series with continuous recurrent units. In *International Conference on Machine Learning*, pages 19388–19405, 2022.

Li Yingzhen and Stephan Mandt. Disentangled sequential autoencoder. In *International Conference on Machine Learning*, pages 5670–5679, 2018.

Joseph Marino, Milan Cvitkovic, and Yisong Yue. A general method for amortizing variational filtering. *Advances in Neural Information Processing Systems*, 31, 2018.

Mohammad Babaeizadeh, Chelsea Finn, Dumitru Erhan, Roy H. Campbell, and Sergey Levine. Stochastic variational video prediction. In *International Conference on Learning Representations*, 2018.

Kevin P Murphy. Switching Kalman filters. Technical report, Compaq Cambridge Research, 1998.

Emily Fox, Erik Sudderth, Michael Jordan, and Alan Willsky. Nonparametric Bayesian learning of switching linear dynamical systems. *Advances in Neural Information Processing Systems*, 21, 2008.

Scott Linderman, Matthew Johnson, Andrew Miller, Ryan Adams, David Blei, and Liam Paninski. Bayesian learning and inference in recurrent switching linear dynamical systems. In *Artificial Intelligence and Statistics*, pages 914–922, 2017.

Chris Cremer, Xuechen Li, and David Duvenaud. Inference suboptimality in variational autoencoders. In *International Conference on Machine Learning*, pages 1078–1086, 2018.

Alexander Alemi, Ben Poole, Ian Fischer, Joshua Dillon, Rif A Saurous, and Kevin Murphy. Fixing a broken ELBO. In *International Conference on Machine Learning*, pages 159–168, 2018.

James Lucas, George Tucker, Roger B Grosse, and Mohammad Norouzi. Don't blame the ELBO! A linear VAE perspective on posterior collapse. *Advances in Neural Information Processing Systems*, 32, 2019b.

R. E. Turner and M. Sahani. Two problems with variational expectation maximisation for time-series models. In D. Barber, T. Cemgil, and S. Chiappa, editors, *Bayesian Time series models*, chapter 5, pages 109–130. Cambridge University Press, 2011.

Rui Shu, Hung H Bui, Shengjia Zhao, Mykel J Kochenderfer, and Stefano Ermon. Amortized inference regularization. *Advances in Neural Information Processing Systems*, 31, 2018.

Junxian He, Daniel Spokoyny, Graham Neubig, and Taylor Berg-Kirkpatrick. Lagging inference networks and posterior collapse in variational autoencoders. *arXiv preprint arXiv:1901.05534*, 2019.

Wu Lin, Nicolas Hubacher, and Mohammad Emtiyaz Khan. Variational message passing with structured inference networks. In *International Conference on Learning Representations*, 2018.

Jakub Tomczak and Max Welling. VAE with a VampPrior. In *International Conference on Artificial Intelligence and Statistics*, pages 1214–1223, 2018.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in Neural Information Processing Systems*, 33:1877–1901, 2020.

Ramin Hasani, Mathias Lechner, Tsun-Hsuan Wang, Makram Chahine, Alexander Amini, and Daniela Rus. Liquid structural state-space models. *arXiv preprint arXiv:2209.12951*, 2022.

Ankit Gupta, Albert Gu, and Jonathan Berant. Diagonal state spaces are as effective as structured state spaces. *Advances in Neural Information Processing Systems*, 35: 22982–22994, 2022.

Linqi Zhou, Michael Poli, Winnie Xu, Stefano Massaroli, and Stefano Ermon. Deep latent state space models for time-series generation. *arXiv preprint arXiv:2212.12749*, 2022.

A. Learning higher dimensional linear dynamics

To investigate if the performance of the SVAE can scale to higher dimensions, we include results on learning higher dimensional linear dynamical systems in Table 1. For the data, we sample from 32 and 64 dimensional linear dynamical systems with 64 and 128 dimensional observations respectively. For the 128 dimensional dataset, we also sampled 10 times more data, since we expect the sample complexity of parameter estimation to be greater. We use the same architecture as in the LDS experiments in Section 4.1, but scaling up the hidden layer sizes to 64 and 128. We also use diagonal matrices for the dynamics noise covariance and recognition potentials for efficiency. None of the models achieve the true marginal likelihood in this regime, but the SVAE still outperforms the other methods considerably.

Table 1: Evidence lower bound (ELBO) on higher dimensional linear dynamical systems.

METHOD	ELBO	
	32D	64D
TRUE MLL	$\mathbf{-1.905}$	$\mathbf{-2.039}$
SVAE	$\mathbf{-2.599}$	$\mathbf{-3.419}$
RNN-AR-L	-4.331	-4.665
RNN-MF	-4.317	-4.781
RNN-AR-NL	-4.697	-6.395
CNN-AR-L	-4.387	-7.005

B. Ablation studies for unsupervised masking

Here we examine the effects of the unsupervised masking approach to learning dynamics. We provide results in validation ELBO and predictive log likelihoods for all of the methods with and without masks applied in Table 2. The results shown here correspond to models with 5 dimensional latent spaces, but the same trends appear across the different dimensionalities studied in the paper. It is clear that while applying the masks hurt the overall ELBO achieved, due to having less access to and more uncertainty in the training data, it is beneficial for learning the correct model for predicting the future.

Table 2: Ablation results for the unsupervised masking method.

METHOD	ELBO	PREDICTION
SVAE	-0.3015	$\mathbf{-0.1558}$
SVAE (NO MASK)	$\mathbf{-0.2763}$	-0.1910
RNN-AR-L	-0.2886	$\mathbf{-0.1717}$
RNN-AR-L (NO MASK)	$\mathbf{-0.2757}$	-0.1862
RNN-MF	-0.2906	$\mathbf{-0.1815}$
RNN-MF (NO MASK)	$\mathbf{-0.2761}$	-0.1899
RNN-AR-NL	-0.2914	$\mathbf{-0.1783}$
RNN-AR-NL (NO MASK)	$\mathbf{-0.2779}$	-0.1798
CNN-AR-L	-0.2902	$\mathbf{-0.1719}$
CNN-AR-L (NO MASK)	$\mathbf{-0.2796}$	-0.1861