Diffusion-Driven State Space Models

Jack Ruder[1,*] **Michael Wojnowicz**[1,†]

[1]Montana State University
[*]jackruder@montana.edu
[†]michael.wojnowicz@montana.edu

Abstract

In many domains, practitioners seek models that produce accurate forecasts while faithfully capturing latent system dynamics. Existing approaches typically sacrifice one of these goals: deep state space models often assume Gaussian latent transitions, limiting fit and forecasting, while diffusion models are highly expressive but lack principled inference for the underlying dynamics. To combine the strengths of both, we introduce the Diffusion-Driven State Space Model (DDSSM), which replaces the conventional Gaussian transition distribution with a diffusion model. Our DDSSM resolves the open problem of how to jointly train an autoencoder and a diffusion model on sequential data, thereby extending the literature on latent diffusion models for time series. Moreover, we find that the DDSSM empirically outperforms a state-of-the-art deep SSM at fitting and forecasting a simulated time series with multimodal transitions.

1. Introduction

Diffusion models (Sohl-Dickstein et al., 2015; Ho et al., 2020) have emerged as a powerful class of generative models for high-dimensional data. For time series data, emerging evidence demonstrates state-of-the-art performance on forecasting, imputation, and sample-generation tasks (Yang et al., 2024), where diffusion models are typically used as generative samplers to produce future trajectories conditioned on past observations (Su et al., 2025). Some approaches perform diffusion directly in observation space (Tashiro et al., 2021; Rasul et al., 2021; Kollovieh et al., 2023; Price et al., 2025), while others apply diffusion in the latent space of an autoencoder (Feng et al., 2024; Liu et al., 2024; Blattmann et al., 2023), which can improve tractability for high-dimensional data. However, existing latent time-series diffusion methods typically rely on two-stage training: first learning the autoencoder, then training the diffusion model. This may produce an over-regularized latent space (Liu et al., 2024) that discards information needed for downstream generative dynamics, consistent with research on latent diffusion for static data where two-stage training leads to poorer representations and generative performance (Vahdat et al., 2021; Shmakov et al., 2023).

In contrast, State Space Models (SSMs) (Kalman, 1960; Shumway and Stoffer, 1982; Durbin and Koopman, 2001) provide a classical framework for temporal data that naturally supports end-to-end training, captures timestep-by-timestep dynamics, and admits well-studied extensions to diverse settings. SSMs typically assume observed data are generated from latent states that evolve over time via a Markov process, with each observation depending on the current state. Classically, the linear Gaussian state-space model assumes first-order Markovian linear dynamics and linear Gaussian observation models (Roweis and Ghahramani, 1999). To absolve linearity, transition and emission densities may be parameterized with deep neural networks (Krishnan et al., 2015, 2017), where training is performed via stochastic backpropagation (Rezende et al., 2014; Kingma and Welling,

2014). These neural-network-parameterized SSMs may be referred to as Deep SSMs, or Dynamical Variational Autoencoders (DVAEs); see Girin et al. (2022) for a review. While neural-networks allow the mean of each latent state to be a *nonlinear* function of the previous latent state, the transition densities in these models assume some parametric form (Karl et al., 2016), most often Gaussian (Krishnan et al., 2015; Johnson et al., 2016; Fraccaro et al., 2017; Krishnan et al., 2017; Ansari et al., 2021; Girin et al., 2022; Revach et al., 2022; Dowling et al., 2024; Hashempoorikderi and Choi, 2024; Chinellato and Marcuzzi, 2025), thereby limiting their expressivity. Moreover, Gaussian transition densities induce a Gaussian prior on the latent trajectories. Within the VAE framework the Gaussian prior is known to over-regularize, yielding latent representations that poorly represent the structure of the data (Tomczak and Welling, 2018; Klushyn et al., 2019; Chen et al., 2017).

Inspired by the framework of composing probabilistic graphical models with neural networks (Johnson et al., 2016), we propose to get the "best of both worlds" by composing SSMs with diffusion models. Our *diffusion-driven* SSMs (DDSSMs) naturally model the real-time dynamics of a system without imposing restrictive Gaussian assumptions about its latent dynamics. Further, we adapt SSM ideas to resolve the open problem of how to jointly train an autoencoder and a diffusion model on sequential data in a principled manner. In this way, we extend latent diffusion models for time series.

2. Method

Generative Model. Let $\mathbf{x}_{1:T} = (\mathbf{x}_1, \ldots, \mathbf{x}_T)$ with $\mathbf{x}_t \in \mathbb{R}^D$ be a time series of T observations, and let $\mathbf{u}_{1:T} = (\mathbf{u}_1, \ldots, \mathbf{u}_T)$ with $\mathbf{u}_t \in \mathbb{R}^V$ denote covariates observed at all time steps. We assume the observations $\mathbf{x}_{1:T}$ are generated from latent variables $\mathbf{z}_{1:T} = (\mathbf{z}_1, \ldots, \mathbf{z}_T)$ with $\mathbf{z}_t \in \mathbb{R}^M$ that capture the underlying system dynamics. After applying standard conditional independence assumptions, the density of the complete data likelihood $p(\mathbf{x}_{1:T}, \mathbf{z}_{1:T}|\mathbf{u}_{1:T})$ can be factorized in the form of a State Space Model (SSM) as

$$p(\mathbf{x}_{1:T}, \mathbf{z}_{1:T}|\mathbf{u}_{1:T}) = \underbrace{p_{\eta,\theta}(\mathbf{z}_{1:j}, \mathbf{x}_{1:j}|\mathbf{u}_{1:j})}_{\text{State Initialization}} \prod_{t=j+1}^{T} \underbrace{p_\psi^{(t)}(\mathbf{z}_t|\mathbf{z}_{t-j:t-1}, \mathbf{u}_{t-j:t})}_{\text{Transition Density}} \underbrace{p_\theta^{(t)}(\mathbf{x}_t|\mathbf{z}_{t-j+1:t}, \mathbf{u}_{t-j+1:t})}_{\text{Emissions Density}}, \quad (1)$$

where θ denotes the parameters of the emissions density, ψ the parameters of the transition density, and η the parameters of the initialization model. The hyperparameter j determines the size of the conditioning set for the transition and emissions densities, and in particular the Markov order of the latent sequence. The superscript t designates that the emissions and transitions are time-inhomogeneous. See Appendix A.1 for the derivation and assumptions leading to this factorization.

As mentioned in the introduction, most existing deep SSMs model the transitions with Gaussian densities where the means and variances are parameterized by neural networks. We depart from this assumption by modeling the transition density $p_\psi^{(t)}$ with a diffusion model, which allows us to learn arbitrary transition densities in the latent space. Our diffusion-driven transition model induces a wider prior over the latent trajectories, allowing us to learn more flexible latent representations of the data. The diffusion model introduces a sequence of intermediate (increasingly noisy) latent states $(\mathbf{z}_t^1, \mathbf{z}_t^2, \ldots, \mathbf{z}_t^K)$ for each time step t, giving an augmented transition model

$$p_\psi^{(t)}(\mathbf{z}_t^0, \mathbf{z}_t^{1:K}|\mathbf{z}_{t-j:t-1}, \mathbf{u}_{t-j:t}) = p(\mathbf{z}_t^K) \prod_{k=1}^{K} p_\psi^{(t)}(\mathbf{z}_t^{k-1}|\mathbf{z}_t^k, \mathbf{z}_{t-j:t-1}, \mathbf{u}_{t-j:t}), \quad (2)$$

where $\mathbf{z}_t^0 := \mathbf{z}_t$ and $\mathbf{z}_t^K$ is approximately distributed as $\mathcal{N}(\mathbf{0}, \boldsymbol{I})$. Marginalizing $\mathbf{z}_t^{1:K}$ recovers the transition density in Eq. (1) while enabling complex, multimodal transitions beyond Gaussian dynamics. Following (Tashiro et al., 2021), we include the time index t and the diffusion noise level k

as conditioning inputs, such that a single neural network with parameters ψ can parameterize the family of densities $(p_\psi^{(t)})_{t=1}^T$ across all time steps and noise levels[1].

For simplicity, we assume Gaussian emissions, with the diffusion model handling non-Gaussian structure and the emissions capturing any remaining uncertainty. We choose the standard parameterization of the emissions as $p_\theta^{(t)}(\mathbf{x}_t|\mathbf{r}_t) = \mathcal{N}(\mu_\theta(\mathbf{r}_t), \Sigma_\theta(\mathbf{r}_t))$, where μ_θ and Σ_θ are neural networks mapping recent latent states and covariates $\mathbf{r}_t = (\mathbf{z}_{t-j+1:t}, \mathbf{u}_{t-j+1:t})$ to the mean and covariance parameters of a Gaussian distribution in observation space. As with the transitions, a single neural network can be used to parameterize the emissions at all time steps by including the time index t within the covariates $\boldsymbol{u}_t$. We model the initialization density via the factorization $p_{\eta,\theta}(\mathbf{z}_{1:j}, \mathbf{x}_{1:j}|\mathbf{u}_{1:j}) = p_\eta(\mathbf{z}_{1:j}|\mathbf{u}_{1:j}) p_\theta^{(t)}(\mathbf{x}_{1:j}|\mathbf{z}_{1:j}, \mathbf{u}_{1:j})$. Then, $p_\eta(\mathbf{z}_{1:j}|\mathbf{u}_{1:j})$ is made expressive using a variational hierarchical prior (VHP) (Klushyn et al., 2019, 2021)—as demonstrated to be effective for deep SSMs. See Appendix G for further discussion and implementation details.

Inference. As the true posterior density $p(\mathbf{z}_{1:T}, \{\mathbf{z}_t^{1:K}\}_{t=j+1}^T|\mathbf{x}_{1:T}, \mathbf{u}_{1:T})$ is intractable, we approximate it with a variational density $q_\phi(\mathbf{z}_{1:T}, \{\mathbf{z}_t^{1:K}\}_{t=j+1}^T|\mathbf{x}_{1:T}, \mathbf{u}_{1:T})$ with learnable parameters ϕ . Our Lemma 1 (see Appendix C.1) along with the well-known fact that the optimal variational posterior factorizes identically to the true posterior (Wainwright and Jordan, 2008) implies that we can assume the following factorization for the variational posterior density $q_\phi(\mathbf{z}_{1:T}, \{\mathbf{z}_t^{1:K}\}_{t=j+1}^T|\mathbf{x}_{1:T}, \mathbf{u}_{1:T})$ without making any approximation other than the dependence on ϕ:

$$
\begin{aligned}
q_\phi(\mathbf{z}_{1:T}, \{\mathbf{z}_t^{1:K}\}_{t=j+1}^T|\mathbf{x}_{1:T}, \mathbf{u}_{1:T}) &= \underbrace{q_\phi(\mathbf{z}_{1:j}|\mathbf{x}_{1:T}, \mathbf{u}_{1:j})}_{\text{State Initialization}} \\
&\times \underbrace{\prod_{t=j+1}^{T} q_\phi^{(t)}(\mathbf{z}_t|\, \mathbf{z}_{t-j:t-1}, \mathbf{x}_{t:T}, \mathbf{u}_{t-j:t})}_{\text{Smoothed Transitions}} \times \underbrace{\prod_{t=j+1}^{T} q_{\text{chain}}(\mathbf{z}_t^{1:K}|\mathbf{z}_t)}_{\text{Diffusion Noising Process}} . \qquad (3)
\end{aligned}
$$

Notably, this factorization contains the Kalman smoothing factor $q_\phi^{(t)}(\mathbf{z}_t|\mathbf{z}_{t-j:t-1}, \mathbf{x}_{t:T}, \mathbf{u}_{t-j:t})$, so the optimal encoder's latent state at time t depends on the *past* latent states and covariates, but *future* observations. Also note that as with models for static data, the diffusion noising process does not require learning.

We jointly learn the variational parameters ϕ along with our model's generative parameters (θ, ψ, η) by maximizing an evidence lower bound (ELBO) on the marginal log-likelihood $\log p(\mathbf{x}_{1:T}|\mathbf{u}_{1:T})$. The ELBO for our model is

$$
\begin{aligned}
\log p\left(\mathbf{x}_{1:T}|\mathbf{u}_{1:T}\right) \geq &- \underbrace{D_{\mathrm{KL}}(q_\phi(\mathbf{z}_{1:j}|\mathbf{x}_{1:T}, \mathbf{u}_{1:T}) \,\|\, p_\eta(\mathbf{z}_{1:j}|\mathbf{u}_{1:j}))}_{\text{KL Divergence for Initialization}} \\
&+ \underbrace{\sum_{t=1}^{T} \mathbb{E}_{q_\phi(\mathbf{z}_{1:T}|\mathbf{x}_{1:T}, \mathbf{u}_{1:T})} \left[\log p_\theta^{(t)}\left(\mathbf{x}_t|\mathbf{z}_{\max(t-j+1,1):t}, \mathbf{u}_{\max(t-j+1,1):t}\right) \right]}_{\text{Negative Reconstruction Error}} \qquad (4a) \\
&- \underbrace{\sum_{t=j+1}^{T} D_{\mathrm{KL}}\left(q_\phi^{(t)}(\mathbf{z}_t|\mathbf{z}_{t-j:t-1}, \mathbf{x}_{1:T}, \mathbf{u}_{1:T}) \,\middle\|\, p_\psi^{(t)}(\mathbf{z}_t|\mathbf{z}_{t-j:t-1}, \mathbf{u}_{t-j:t}) \right)}_{\text{KL Divergence for Transition}}, \qquad (4b)
\end{aligned}
$$

[1]Some works instead include the time *difference* as a covariate, in which case the transition model is time-homogeneous (Krishnan et al., 2017).

where Eq. (4b) can be further rewritten as

$$\underbrace{\sum_{t=j+1}^{T} \mathbb{E}_{q_\phi(\mathbf{z}_{1:T}, \{\mathbf{z}_t^{1:K}\}_{t=j+1}^{T} | \mathbf{x}_{1:T}, \mathbf{u}_{1:T})} \Big[\log p(\mathbf{z}_t^K) + \sum_{k=1}^{K} - \log q(\mathbf{z}_t^k | \mathbf{z}_t^{k-1}) + \log p_\psi^{(t)}(\mathbf{z}_t^{k-1} | \mathbf{z}_t^k, \mathbf{c}_t) \Big]}_{\text{Diffusion Chain}} \tag{5a}$$

$$+ \underbrace{\sum_{t=j+1}^{T} \mathbb{E}_{q_\phi(\mathbf{z}_{1:T} | \mathbf{x}_{1:T}, \mathbf{u}_{1:T})} \Big[- \log q_\phi^{(t)} \left(\mathbf{z}_t | \mathbf{z}_{t-j:t-1}, \mathbf{x}_{t:T}, \mathbf{u}_{t-j:t} \right) \Big]}_{\text{Posterior Transition Entropy}}, \tag{5b}$$

with $\mathbf{c}_t = (\mathbf{z}_{t-j:t-1}, \mathbf{u}_{t-j:t})$ denoting the conditioning set for the diffusion model at time t. Unlike existing latent time-series diffusion models that model $q_\phi(\mathbf{z}_{1:T} | \mathbf{x}_{1:T}, \mathbf{u}_{1:T})$ without factorization, our objective mirrors the conditional independence structure of the true posterior, providing a strong inductive bias that narrows the ELBO approximation gap. Specifically per Eq. (3), the encoder constructs latent representations for $\mathbf{z}_t$ by connecting the past (via $\mathbf{z}_{t-j:t-1}$) to the future (via $\mathbf{x}_{t:T}$). The transition model does not observe $\mathbf{x}_{t:T}$, thus by jointly training the transition model and the encoder, Eq. (4b) encourages the encoder to produce latent states that are predictable under the dynamics learned by the transition model. Moreover, as our diffusion model is highly expressive, the regularization of $\mathbf{z}_t$ is no longer constrained towards a strict Gaussian prior.

Parameterization and Training. The design of the neural networks parameterizing these densities is detailed in Appendix H. We follow standard practice (Girin et al., 2022) in minimizing the negative ELBO with stochastic backpropagation. To ensure the flow of gradients through the diffusion model to the VAE are normalized across noise levels, we use preconditioning (Karras et al., 2022) as implemented in Appendix D.2 to rewrite Eq. (5a). Our full training algorithm, along with an explanation of how to compute each of the terms in the ELBO, can be found in Appendix I.

3. Related work

Deep state space models. Similarly to other Deep State Space Models (DSSMs) (Girin et al., 2022) such as the Deep Kalman Filter (DKF) (Krishnan et al., 2015, 2017), our Diffusion-Driven State Space Model (DDSSM) parametrizes the transitions and emissions densities of a state space model with a neural network. However, our model replaces the parametric (typically Gaussian) transition distribution of a DSSM with a diffusion model. This design enables complex, potentially multi-modal transition dynamics (validated in Section 4) and avoids the restrictive regularization of Gaussian transitions (Karl et al., 2016). Integration with advances such as the extended Kalman filter VAE (Klushyn et al., 2021) is left for future work.

Diffusion models for temporal data. Some works (Rasul et al., 2021; Tashiro et al., 2021; Kollovieh et al., 2023) apply diffusion directly in observation space, but the high computational cost of diffusion – motivating latent diffusion for static data (Rombach et al., 2022) – is exacerbated in time series due to the need to model both high-dimensional structure and temporal dynamics. This has led to latent diffusion approaches for time series (Qian et al., 2024; Suh et al., 2025; Liu et al., 2024; Feng et al., 2024), which typically rely on two-stage training (autoencoder then diffusion), potentially yielding over-regularized latent spaces that discard information needed for generative dynamics (Liu et al., 2024; Vahdat et al., 2021; Shmakov et al., 2023). Our approach instead follows Vahdat et al. (2021) and Shmakov et al. (2023) by training diffusion end-to-end in latent space, extended here to the temporal setting. For further details on related works, see Appendix J.

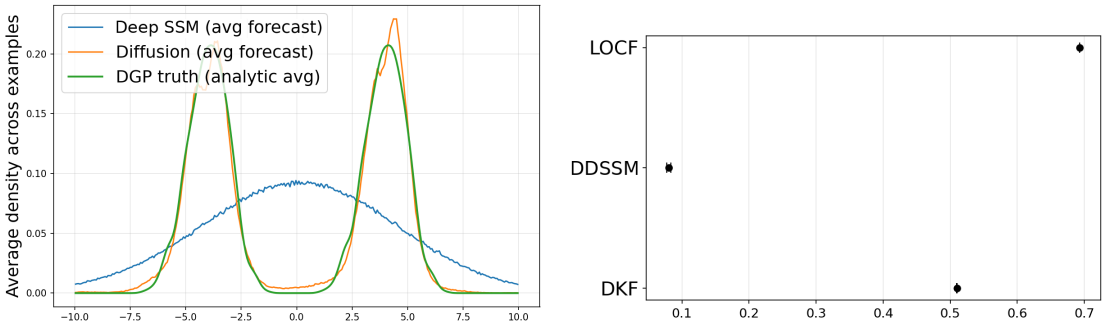
4. Simulation study: Bimodal Noise

To demonstrate the advantage of flexible transitions, we implement a latent dependent-step random walk with bimodal noise

$$
\begin{aligned}
z_t^{(i)} &= 0.9 z_{t-1}^{(i)} + 4 s_t^{(i)}, \quad & s_t^{(i)} &\sim \mathrm{Unif}(\{-1, 1\}). \\
x_t^{(i)} &= z_t^{(i)} + \epsilon_t^{(i)}, & \epsilon_t^{(i)} &\sim \mathcal{N}(0, 0.2),
\end{aligned}
$$

where $i = 1, \ldots, N$ indexes i.i.d sequences, and $t = 1, \ldots, T$ indexes the time steps of those sequences. We fix $T = 32$ and generate $N = 1024$ distinct sequences each for training, validation, and testing.

We fit three models to the data, fixing a latent dimension of 1. The first is the DKF equipped with a VHP, and second is our DDSSM model. As a naïve baseline we include a Last Observation Carried Forward (LOCF) strategy (Shao and Zhong, 2003), which carries forward the last observed value, i.e $\hat{x}_t^{(i)} = x_{t-1}^{(i)}$. Details of the exact training procedure are in Appendix I; model configurations and hyperparameter search procedure are provided in Appendix F. To evaluate the performance of the

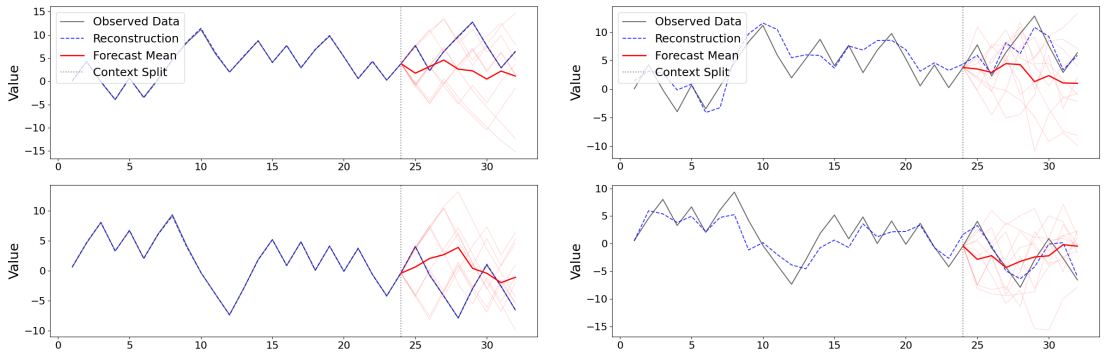


(a) Predictive distributions, centered on x_{31}

(b) Jensen-Shannon Divergence

Figure 1: On the left, the average predictive distribution of each model across the test set is plotted against the true distribution. On the right, the JSD is plotted as the mean across the test set for each model, with (small) error bars giving $\pm$ 1 standard error of the mean.

models, we draw 1024 samples from each model. Then, we compute the Jensen-Shannon Divergence (JSD) between the 1-step predictive distribution of each model and the true distribution of $x_{32}^{(i)}$ given $x_{1:31}^{(i)}$ for each example in the test set. Each sample is drawn by first drawing a latent sample $\hat{z}_{31}^{(i)}$ from the encoder, and then sampling the transition and emission distributions to produce $\hat{x}_{32}^{(i)}$. Figure 1a reveals, as expected, that the DKF places probability mass between the two modes of the true distribution, while the DDSSM is able to capture both modes. Qualitatively, this results in a significantly lower JSD as shown in Fig. 1b. To investigate the impacts of diffusion transitions on prediction, we plot the reconstructions and forecasts of two representative samples from the test set, chosen according to the median value of the JSD metric for the DDSSM model. Figure 2 shows that the DDSSM's trajectories are representative of the data, while the DKF's forecasts do not have a consistent pattern. Notably, the DKF fails to provide good reconstructions of the observed data, supporting our claim that a Gaussian prior over the latent trajectories is harmful for learning latent representations.



(a) DDSSM (b) DKF

Figure 2: Reconstructions and Forecasts. The dashed line shows $\hat{x}^{(i)}_{1:32}$ for each model, computed by sampling $\hat{z}^{(i)}_{1:32}$ from the encoder, and subsequently sampling $\hat{x}^{(i)}_{1:32}$ from the decoder. Given the encoder's sample $\hat{z}^{(i)}_{24}$, we generate trajectories of $\hat{x}^{(i)}_{25:32}$ by sampling from the transitions and emissions. Trajectories are shown in light red, and the mean of these trajectories is in solid red.

5. Discussion

In this work we have demonstrated that we can formulate latent diffusion models for time series by casting the problem as a state space model with a diffusion transition distribution. This allows end-to-end training, allowing flexible reconstructions without penalizing the learning of the latent dynamics. We aim to continue our work by applying the DDSSM to large-scale datasets, and by incorporating ideas from deep SSM literature to fully utilize the power of the diffusion transition distribution.

Acknowledgements

Funding. This material is based upon work supported in part by the National Science Foundation EPSCoR Cooperative Agreement OIA-2242802. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

Computing Resources. Computational efforts were performed on the Tempest High Performance Computing System, operated and supported by University Information Technology Research Cyberinfrastructure (RRID:SCR_026229) at Montana State University.

References

Takuya Akiba, Shotaro Sano, Toshihiko Yanase, Takeru Ohta, and Masanori Koyama. Optuna: A next-generation hyperparameter optimization framework. In *The 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 2623–2631, 2019.

Abdul Fatir Ansari, Konstantinos Benidis, Richard Kurle, Ali Caner Turkmen, Harold Soh, Alexander J Smola, Bernie Wang, and Tim Januschowski. Deep explicit duration switching models for time series. *Advances in Neural Information Processing Systems*, 34:29949–29961, 2021.

Andreas Blattmann, Robin Rombach, Huan Ling, Tim Dockhorn, Seung Wook Kim, Sanja Fidler, and Karsten Kreis. Align your latents: High-resolution video synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 22563–22575, June 2023.

Tianqi Chen, Bing Xu, Chiyuan Zhang, and Carlos Guestrin. Training deep nets with sublinear memory cost. *arXiv preprint arXiv:1604.06174*, 2016.

Xi Chen, Diederik P Kingma, Tim Salimans, Yan Duan, Prafulla Dhariwal, John Schulman, Ilya Sutskever, and Pieter Abbeel. Variational lossy autoencoder. In *International Conference on Learning Representations*, 2017.

Xiongjie Chen and Yunpeng Li. Normalizing flow-based differentiable particle filters. *IEEE Transactions on Signal Processing*, 73:493–507, 2025.

Erik Chinellato and Fabio Marcuzzi. State, parameters and hidden dynamics estimation with the deep kalman filter: Regularization strategies. *Journal of Computational Science*, 87:102569, 2025.

Emmanuel De Bezenac, Syama Sundar Rangapuram, Konstantinos Benidis, Michael Bohlke-Schneider, Richard Kurle, Lorenzo Stella, Hilaf Hasson, Patrick Gallinari, and Tim Januschowski. Normalizing kalman filters for multivariate time series analysis. *Advances in Neural Information Processing Systems*, 33:2995–3007, 2020.

Matthew Dowling, Yuan Zhao, and Il Memming Park. exponential family dynamical systems (xfads): Large-scale nonlinear gaussian state-space modeling. *Advances in Neural Information Processing Systems*, 37:13458–13488, 2024.

James Durbin and Siem Jan Koopman. *Time series analysis by state space methods*. Oxford University Press (UK), 2001.

Shibo Feng, Chunyan Miao, Zhong Zhang, and Peilin Zhao. Latent diffusion transformer for probabilistic time series forecasting. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 11979–11987, 2024.

Marco Fraccaro, Simon Kamronn, Ulrich Paquet, and Ole Winther. A disentangled recognition and nonlinear dynamics model for unsupervised learning. *Advances in neural information processing systems*, 30, 2017.

Laurent Girin, Simon Leglaive, Xiaoyu Bie, Julien Diard, Thomas Hueber, and Xavier Alameda-Pineda. Dynamical variational autoencoders: A comprehensive review. *Foundations and Trends in Machine Learning*, 15(1-2):1–175, 2022.

Hamidreza Hashempoorikderi and Wan Choi. Gated inference network: Inference and learning state-space models. *Advances in Neural Information Processing Systems*, 37:39036–39073, 2024.

Irina Higgins, Loic Matthey, Arka Pal, Christopher Burgess, Xavier Glorot, Matthew Botvinick, Shakir Mohamed, and Alexander Lerchner. beta-vae: Learning basic visual concepts with a constrained variational framework. In *International conference on learning representations*, 2017.

Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in neural information processing systems*, 33:6840–6851, 2020.

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, 2016.

Rudolph Emil Kalman. A new approach to linear filtering and prediction problems. 1960.

Maximilian Karl, Maximilian Soelch, Justin Bayer, and Patrick Van der Smagt. Deep variational bayes filters: Unsupervised learning of state space models from raw data. *arXiv preprint arXiv:1605.06432*, 2016.

Tero Karras, Miika Aittala, Timo Aila, and Samuli Laine. Elucidating the design space of diffusion-based generative models. *Advances in neural information processing systems*, 35:26565–26577, 2022.

Diederik P Kingma and Max Welling. Auto-encoding variational bayes. In *International Conference on Learning Representations (ICLR)*, 2014.

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.

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.

Marcel Kollovieh, Abdul Fatir Ansari, Michael Bohlke-Schneider, Jasper Zschiegner, Hao Wang, and Yuyang Bernie Wang. Predict, refine, synthesize: Self-guiding diffusion models for probabilistic time series forecasting. *Advances in Neural Information Processing Systems*, 36:28341–28364, 2023.

Rahul Krishnan, Uri Shalit, and David Sontag. Structured inference networks for nonlinear state space models. In *Proceedings of the AAAI conference on artificial intelligence*, volume 31, 2017.

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

Tuan Anh Le, Maximilian Igl, Tom Rainforth, Tom Jin, and Frank Wood. Auto-encoding sequential monte carlo. *arXiv preprint arXiv:1705.10306*, 2017.

Yuhang Liu, Yingxue Zhang, Xin Zhang, Yu Yang, Yiqun Xie, Sahar Ghanipoor Machiani, Yanhua Li, and Jun Luo. Align along time and space: A graph latent diffusion model for traffic dynamics prediction. In *2024 IEEE International Conference on Data Mining (ICDM)*, pages 271–280. IEEE, 2024.

George Papamakarios, Eric Nalisnick, Danilo Jimenez Rezende, Shakir Mohamed, and Balaji Lakshminarayanan. Normalizing flows for probabilistic modeling and inference. *Journal of Machine Learning Research*, 22(57):1–64, 2021.

Ilan Price, Alvaro Sanchez-Gonzalez, Ferran Alet, Tom R Andersson, Andrew El-Kadi, Dominic Masters, Timo Ewalds, Jacklynn Stott, Shakir Mohamed, Peter Battaglia, et al. Probabilistic weather forecasting with machine learning. *Nature*, 637(8044):84–90, 2025.

Jian Qian, Bingyu Xie, Biao Wan, Minhao Li, Miao Sun, and Patrick Yin Chiang. Timeldm: Latent diffusion model for unconditional time series generation. *arXiv preprint arXiv:2407.04211*, 2024.

Kashif Rasul, Calvin Seward, Ingmar Schuster, and Roland Vollgraf. Autoregressive denoising diffusion models for multivariate probabilistic time series forecasting. In *International conference on machine learning*, pages 8857–8868. PMLR, 2021.

Guy Revach, Nir Shlezinger, Xiaoyong Ni, Adria Lopez Escoriza, Ruud JG Van Sloun, and Yonina C Eldar. Kalmannet: Neural network aided kalman filtering for partially known dynamics. *IEEE Transactions on Signal Processing*, 70:1532–1547, 2022.

Danilo Jimenez Rezende and Fabio Viola. Taming vaes. *arXiv preprint arXiv:1810.00597*, 2018.

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. PMLR, 2014.

Maria Isabel Ribeiro. Kalman and extended kalman filters: Concept, derivation and properties. *Institute for Systems and Robotics*, 43(46):3736–3741, 2004.

Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 10684–10695, 2022.

Sam Roweis and Zoubin Ghahramani. A unifying review of linear gaussian models. *Neural computation*, 11(2):305–345, 1999.

Jun Shao and Bob Zhong. Last observation carry-forward and last observation analysis. *Statistics in medicine*, 22(15):2429–2441, 2003.

Alexander Shmakov, Kevin Greif, Michael Fenton, Aishik Ghosh, Pierre Baldi, and Daniel Whiteson. End-to-end latent variational diffusion models for inverse problems in high energy physics. In A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine, editors, *Advances in Neural Information Processing Systems*, volume 36, pages 65102–65127. Curran Associates, Inc., 2023.

Robert H Shumway and David S Stoffer. An approach to time series smoothing and forecasting using the em algorithm. *Journal of time series analysis*, 3(4):253–264, 1982.

Jascha Sohl-Dickstein, Eric Weiss, Niru Maheswaranathan, and Surya Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics. In *International conference on machine learning*, pages 2256–2265. pmlr, 2015.

Yang Song, Jascha Sohl-Dickstein, Diederik P Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-based generative modeling through stochastic differential equations. *arXiv preprint arXiv:2011.13456*, 2020.

Chen Su, Zhengzhou Cai, Yuanhe Tian, Zhuochao Chang, Zihong Zheng, and Yan Song. Diffusion models for time series forecasting: A survey. *arXiv preprint arXiv:2507.14507*, 2025.

Namjoon Suh, Yuning Yang, Din-Yin Hsieh, Qitong Luan, Shirong Xu, Shixiang Zhu, and Guang Cheng. Timeautodiff: A unified framework for generation, imputation, forecasting, and time-varying metadata conditioning of heterogeneous time series tabular data. *Transactions on Machine Learning Research*, 2025.

Yusuke Tashiro, Jiaming Song, Yang Song, and Stefano Ermon. Csdi: Conditional score-based diffusion models for probabilistic time series imputation. *Advances in neural information processing systems*, 34:24804–24816, 2021.

Jakub Tomczak and Max Welling. Vae with a vampprior. In *International conference on artificial intelligence and statistics*, pages 1214–1223. PMLR, 2018.

Arash Vahdat, Karsten Kreis, and Jan Kautz. Score-based generative modeling in latent space. *Advances in neural information processing systems*, 34:11287–11302, 2021.

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

Yiyuan Yang, Ming Jin, Haomin Wen, Chaoli Zhang, Yuxuan Liang, Lintao Ma, Yi Wang, Chenghao Liu, Bin Yang, Zenglin Xu, Jiang Bian, Shirui Pan, and Qingsong Wen. A survey on diffusion models for time series and spatio-temporal data, 2024.

Zachary Ziegler and Alexander Rush. Latent normalizing flows for discrete sequences. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 7673–7682. PMLR, 09–15 Jun 2019.

A. State Space Model Derivation

A.1. Derivation of the Generative Model

Our goal is to learn a model of the conditional distribution $p(\mathbf{x}_{1:T}|\mathbf{u}_{1:T})$. Following the general framework of Girin et al. (2022), we introduce a sequence of latent variables $\mathbf{z}_{1:T} = (\mathbf{z}_1, \mathbf{z}_2, \ldots, \mathbf{z}_T)$, where each $\mathbf{z}_t \in \mathbb{R}^d$ is a d-dimensional vector, and factorize the joint distribution $p(\mathbf{x}_{1:T}, \mathbf{z}_{1:T}|\mathbf{u}_{1:T})$ as

$$p(\mathbf{x}_{1:T}, \mathbf{z}_{1:T}|\mathbf{u}_{1:T}) = \prod_{t=1}^{T} p(\mathbf{x}_t|\mathbf{x}_{1:t-1}, \mathbf{z}_{1:t}, \mathbf{u}_{1:t}) p(\mathbf{z}_t|\mathbf{x}_{1:t-1}, \mathbf{z}_{1:t-1}, \mathbf{u}_{1:t}). \tag{6}$$

We make the Markovian assumption that the latent state at time t only depends on a fixed-length history of j previous latent states and covariates, rather than the entire history, i.e.

$$p\left(\mathbf{z}_t|\mathbf{z}_{1:t-1}, \mathbf{u}_{1:t}\right) = p\left(\mathbf{z}_t|\mathbf{z}_{t-j:t-1}, \mathbf{u}_{t-j:t}\right), \tag{7}$$

for $t > j$ where j is a fixed Markov order.

On the emissions model, we assume that the observation at time t only depends on a fixed-length history of j previous latent states and covariates, i.e.

$$p(\mathbf{x}_t|\mathbf{x}_{1:t-1}, \mathbf{z}_{1:t}, \mathbf{u}_{1:t}) = p(\mathbf{x}_t|\mathbf{z}_{t-j+1:t}, \mathbf{u}_{t-j+1:t}). \tag{8}$$

We explicitly choose the same conditioning set for the emissions model, arguing that the most recent j states carry the full information about the system's current state, and thus the most recent j states and covariates should be sufficient for predicting the current observation. Alternatively we could choose a smaller order on the emissions as well; this is not a key aspect of our model and leave this choice to future work.

Lastly, we assume that the initial j latent states are generated from a separate initial state distribution, i.e.

$$p(\mathbf{z}_{1:j}, \mathbf{x}_{1:j}|\mathbf{u}_{1:j}) = p_\eta(\mathbf{z}_{1:j}|\mathbf{u}_{1:j}) p_\theta(\mathbf{x}_{1:j}|\mathbf{z}_{1:j}, \mathbf{u}_{1:j}). \tag{9}$$

Rewriting Eq. (6) with the above assumptions, we have

$$\begin{aligned} p(\mathbf{x}_{1:T}, \mathbf{z}_{1:T}|\mathbf{u}_{1:T}) = & \underbrace{p_\eta(\mathbf{z}_{1:j}|\mathbf{u}_{1:j}) p_\theta(\mathbf{x}_{1:j}|\mathbf{z}_{1:j}, \mathbf{u}_{1:j})}_{\text{State Initialization}} \\ & \times \prod_{t=j+1}^{T} \underbrace{p_\psi^{(t)}(\mathbf{z}_t|\mathbf{z}_{t-j:t-1}, \mathbf{u}_{t-j:t})}_{\text{Transition Density}} \times \underbrace{p_\theta^{(t)}(\mathbf{x}_t|\mathbf{z}_{t-j+1:t}, \mathbf{u}_{t-j+1:t})}_{\text{Emissions Model}}, \end{aligned} \tag{10}$$

where we have indicated the parameters as explained in the main text.

The inclusion of j as a hyperparameter for our model is inspired by the GenCast model of Price et al. (2025), which found that a second-order Markov assumption (in observation space) was more effective for forecasting in contrast to a first-order assumption. We note that another option for increasing the Markov order would be to use a higher-dimensional latent state, where the transition density is still first-order in the latent space, but the latent state at time t includes the previous j states. However, the representation here allows for independently controlling the Markov order and the dimension of the latent state. Another motivation for this representation is that it allows us to view our method as a direct generalization of autoregressive diffusion models. For example, the model proposed by Price et al. (2025) can be viewed as our model with $j = 2$ and identity emissions.

B. Modeling the Transition Density with Diffusion Models

To model the transition density $p(\mathbf{z}_t|\mathbf{z}_{t-j:t-1})$ as a diffusion model, we must introduce the intermediate states used in the diffusion process. To simplify notation, we will denote the history of latent states and covariates as $\mathbf{c}_t := (\mathbf{z}_{t-j:t-1}, \mathbf{u}_{t-j:t})$, as this conditioning set remains fixed across the diffusion steps for a given time t.

Eq. (27) shows that we may write the transition density as

$$p_\psi^{(t)}(\mathbf{z}_t|\mathbf{c}_t) = \int p_\psi^{(t)}(\mathbf{z}_t^0|\mathbf{z}_t^1, \mathbf{c}_t) \prod_{k=2}^{K} p_\psi^{(t)}(\mathbf{z}_t^{k-1}|\mathbf{z}_t^k, \mathbf{c}_t)\, p(\mathbf{z}_t^K)\, d\mathbf{z}_t^{1:K}, \tag{11}$$

where $\mathbf{z}_t^0 := \mathbf{z}_t$ is the clean latent state, $p(\mathbf{z}_t^K)$ has standard normal density, and $p_\psi^{(t)}(\mathbf{z}_t^{k-1}|\mathbf{z}_t^k, \mathbf{z}_{t-j:t-1})$ is the learned reverse process of the diffusion model.

Thus, we augment Eq. (1) to include the intermediate diffusion states $\mathbf{z}_t^{1:K}$ for $t > j$.

The augmented joint distribution is

$$\begin{aligned} &p(\mathbf{x}_{1:T}, \mathbf{z}_{1:T}, \left\{\mathbf{z}_t^{1:K}\right\}_{t=j+1}^{T} |\mathbf{u}_{1:T}) \\ &\quad = p_{\eta,\theta}(\mathbf{z}_{1:j}, \mathbf{x}_{1:j}|\mathbf{u}_{1:j}) \times \prod_{t=j+1}^{T} \left[p(\mathbf{z}_t^K) \prod_{k=1}^{K} p_\psi^{(t)}(\mathbf{z}_t^{k-1}|\mathbf{z}_t^k, \mathbf{c}_t)) \times p_\theta^{(t)}(\mathbf{x}_t|\mathbf{z}_{t-j+1:t}, \mathbf{u}_{t-j+1:t}) \right]. \end{aligned} \tag{12}$$

We may recover Eq. (1) by marginalizing out the diffusion states $\mathbf{z}_t^{1:K}$ for $t > j$.

C. ELBO Derivation

C.1. Posterior Factorization

Learning our generative model in Eq. (12) requires performing inference over the latent variables $\mathbf{z}_{1:T}$ and $\{\mathbf{z}_t^{1:K}\}_{t=j+1}^{T}$. To do so, we follow works from the deep state-space modeling literature (Krishnan et al., 2015, 2017; Girin et al., 2022) and train under the VAE framework which allows jointly learning the generative model and the inference model. The primary objective is to maximize the marginal log-likelihood $\log p(\mathbf{x}_{1:T}|\mathbf{u}_{1:T})$ with respect to the parameters of the generative model. As is typical with latent-variable models, exact inference depends on the smoothing posterior distribution

$$p(\mathbf{z}_{1:T}, \{\mathbf{z}_t^{1:K}\}_{t=j+1}^{T} \mid \mathbf{x}_{1:T}, \mathbf{u}_{1:T}), \tag{13}$$

whose normalization involves integrating over a high-dimensional latent space and is generally intractable.

To obtain a tractable objective, we introduce a variational distribution

$$q_\phi(\mathbf{z}_{1:T}, \left\{\mathbf{z}_t^{1:K}\right\}_{t=j+1}^{T} |\mathbf{x}_{1:T}, \mathbf{u}_{1:T}) \tag{14}$$

to approximate the true posterior distribution. It is important to choose a family of variational distributions which matches the conditional independence structure of the true posterior distribution (Girin et al., 2022). If the variational distribution misses important dependencies required to capture the true posterior, then the variational distribution will not be able to approximate the true posterior well, resulting in a looser bound on the marginal log-likelihood.

Therefore, we examine the form of the true posterior distribution $p(\mathbf{z}_{1:T}, \{\mathbf{z}_t^{1:K}\}_{t=j+1}^{T}|\mathbf{x}_{1:T}, \mathbf{u}_{1:T})$ to determine the appropriate factorization for the variational distribution. The conditional independence structure of the true posterior distribution can be determined by applying d-separation to the directed graphical model corresponding to the generative model in Eq. (12), resulting in the following lemma.

Lemma 1. *For the generative model in Eq. (12), the exact smoothing posterior factorizes as*

$$p(\mathbf{z}_{1:T}, \{\mathbf{z}_t^{1:K}\}_{t=j+1}^T | \mathbf{x}_{1:T}, \mathbf{u}_{1:T}) \tag{15}$$

$$= \left[\prod_{t=1}^{j} p(\mathbf{z}_t | \mathbf{x}_{t:T}, \mathbf{z}_{1:t-1}, \mathbf{u}_{1:t})\right] \times \left[\prod_{t=j+1}^{T} p(\mathbf{z}_t | \mathbf{x}_{t:T}, \mathbf{z}_{t-j:t-1}, \mathbf{u}_{t-j:t})\right] \tag{16}$$

$$\times \left[\prod_{t=j+1}^{T} p(\mathbf{z}_t^{1:K} | \mathbf{z}_t, \mathbf{z}_{t-j:t-1}, \mathbf{u}_{t-j:t})\right], \tag{17}$$

where

$$p(\mathbf{z}_t^{1:K} | \mathbf{z}_t, \mathbf{z}_{t-j:t-1}, \mathbf{u}_{t-j:t}) \propto p(\mathbf{z}_t^K) \prod_{k=1}^{K} p_\psi^{(t)}(\mathbf{z}_t^{k-1} | \mathbf{z}_t^k, \mathbf{z}_{t-j:t-1}, \mathbf{u}_{t-j:t}), \tag{18}$$

with $\mathbf{z}_t^0 := \mathbf{z}_t$.

Proof. Let the reverse-time markov chain $\mathbf{z}_t^K \to \cdots \to \mathbf{z}_t^1 \to \mathbf{z}_t^0$ be parameterized by the time-homogeneous transition densities $p_\psi^{(t)}(\mathbf{z}_t^{k-1} | \mathbf{z}_t^k, \mathbf{z}_{t-j:t-1}, \mathbf{u}_{t-j:t})$.

Applying Bayes rule to $p(\mathbf{z}_{1:T}, \{\mathbf{z}_t^{1:K}\}_{t=j+1}^T | \mathbf{x}_{1:T}, \mathbf{u}_{1:T})$, we have

$$\begin{aligned}
&p(\mathbf{z}_{1:T}, \{\mathbf{z}_t^{1:K}\}_{t=j+1}^T | \mathbf{x}_{1:T}, \mathbf{u}_{1:T}) \\
&= \frac{p(\mathbf{x}_{1:T}, \mathbf{z}_{1:T}, \{\mathbf{z}_t^{1:K}\}_{t=j+1}^T | \mathbf{u}_{1:T})}{p(\mathbf{x}_{1:T} | \mathbf{u}_{1:T})} \\
&\propto p_{\eta,\theta}(\mathbf{z}_{1:j}, \mathbf{x}_{1:j} | \mathbf{u}_{1:j}) \\
&\qquad \times \prod_{t=j+1}^{T} \left[p(\mathbf{z}_t^K) \prod_{k=1}^{K} p_\psi^{(t)}(\mathbf{z}_t^{k-1} | \mathbf{z}_t^k, \mathbf{c}_t)) \cdot p_\theta^{(t)}(\mathbf{x}_t | \mathbf{z}_{t-j+1:t}, \mathbf{u}_{t-j+1:t})\right].
\end{aligned}$$

This induces a directed acyclic graph (DAG) over the variables $\mathbf{x}_{1:T}, \mathbf{z}_{1:T}, \{\mathbf{z}_t^{1:K}\}_{t=j+1}^T, \mathbf{u}_{1:T}$.

Applying the chain rule to the desired posterior form, we have

$$\begin{aligned}
p(\mathbf{z}_{1:T}, \{\mathbf{z}_t^{1:K}\}_{t=j+1}^T \mid \mathbf{x}_{1:T}, \mathbf{u}_{1:T}) \;\propto\; &\prod_{t=1}^{j} p(\mathbf{z}_t \mid \mathbf{x}_{1:T}, \mathbf{z}_{1:t-1}, \mathbf{u}_{1:T}) \\
\cdot \prod_{t=j+1}^{T} p(\mathbf{z}_t \mid \mathbf{x}_{1:T}, \mathbf{z}_{1:t-1}, \mathbf{u}_{1:T}) \;\cdot\; &\prod_{t=j+1}^{T} p(\mathbf{z}_t^{1:K} \mid \mathbf{x}_{1:T}, \mathbf{u}_{1:T}, \mathbf{z}_{1:T}, \{\mathbf{z}_s^{1:K}\}_{s<t}).
\end{aligned}$$

In the augmented DAG, any path from $\mathbf{z}_t^{1:K}$ to $\{\mathbf{x}_s\}_{s=1}^T$ or to latents outside of $\mathbf{c}_t$ must pass through $(\mathbf{z}_{t-j:t})$. Hence by d-separation,

$$p(\mathbf{z}_t^{1:K} \mid \mathbf{x}_{1:T}, \mathbf{u}_{1:T}, \mathbf{z}_{1:T}, \{\mathbf{z}_s^{1:K}\}_{s<t}) \;=\; p(\mathbf{z}_t^{1:K} \mid \mathbf{z}_t, \mathbf{c}_t).$$

Moreover, by the Markov structure of the reverse chain, and $p(\mathbf{z}_t^K)$ being a prior independent of $\mathbf{c}_t$,

$$p(\mathbf{z}_t^{1:K} \mid \mathbf{z}_t, \mathbf{c}_t) \;\propto\; p(\mathbf{z}_t^K) \prod_{k=1}^{K} p_\psi^{(t)}(\mathbf{z}_t^{k-1} \mid \mathbf{z}_t^k, \mathbf{c}_t), \qquad \mathbf{z}_t^0 \equiv \mathbf{z}_t,$$

with the same parameter set ψ for all t (time-homogeneous in t).

For $t > j$, by the j-th order Markov property, $\mathbf{z}_t \perp\!\!\!\perp \mathbf{z}_{1:t-j-1} \mid (\mathbf{z}_{t-j:t-1}, \mathbf{u}_{t-j:t})$. Conditioning on all observations, any influence of $\mathbf{x}_{1:t-1}$ on $\mathbf{z}_t$ is blocked by the parents $\mathbf{z}_{t-j:t-1}$; thus

$$\mathbf{z}_t \;\perp\!\!\!\perp\; \mathbf{x}_{1:t-1} \mid (\mathbf{z}_{t-j:t-1}, \mathbf{u}_{t-j:t}, \mathbf{x}_{t:T}).$$

Therefore,

$$p(\mathbf{z}_t \mid \mathbf{x}_{1:T}, \mathbf{z}_{1:t-1}, \mathbf{u}_{1:T}) \;=\; p(\mathbf{z}_t \mid \mathbf{x}_{t:T}, \mathbf{z}_{t-j:t-1}, \mathbf{u}_{t-j:t}), \qquad t > j.$$

For $t \leq j$, we similarly have

$$p(\mathbf{z}_t \mid \mathbf{x}_{1:T}, \mathbf{z}_{1:t-1}, \mathbf{u}_{1:T}) \;=\; p(\mathbf{z}_t \mid \mathbf{x}_{t:T}, \mathbf{z}_{1:t-1}, \mathbf{u}_{1:t}).$$

Substituting the two reductions back into the chain-rule factorization yields

$$p(\mathbf{z}_{1:T}, \{\mathbf{z}_t^{1:K}\}_{t=j+1}^{T} \mid \mathbf{x}_{1:T}, \mathbf{u}_{1:T}) \;\propto\; \left[\prod_{t=1}^{j} p(\mathbf{z}_t \mid \mathbf{x}_{t:T}, \mathbf{z}_{1:t-1}, \mathbf{u}_{1:t})\right] \cdot \left[\prod_{t=j+1}^{T} p(\mathbf{z}_t \mid \mathbf{x}_{t:T}, \mathbf{z}_{t-j:t-1}, \mathbf{u}_{t-j:t})\right] \cdot \left[\prod_{t=j+1}^{T} p(\mathbf{z}_t^{1:K} \mid \mathbf{z}_t, \mathbf{c}_t)\right].$$

Re-normalizing both sides gives the stated posterior factorization. □

Once again, if we marginalize the intermediate diffusion states $\mathbf{z}_t^{1:K}$, Eq. (18) recovers the original transition density $p_\psi^{(t)}(\mathbf{z}_t|\mathbf{z}_{t-j:t-1}, \mathbf{u}_{t-j:t})$. This serves to highlight that there are two perspectives to our model. First, we may view our model as replacing the transition density in a standard state-space model with a diffusion model, which allows for more flexible transitions. Alternatively, we may view our model as a state-space model with an augmented latent space which includes the intermediate diffusion states, where marginalization of the intermediate states recovers the more-general state-space model in Eq. (1).

The optimal variational posterior factorizes identically to the true posterior (Wainwright and Jordan, 2008). Hence, given Lemma 1, we assume the following factorization for the variational posterior density $q_\phi(\mathbf{z}_{1:T}, \{\mathbf{z}_t^{1:K}\}_{t=j+1}^{T}|\mathbf{x}_{1:T}, \mathbf{u}_{1:T})$ without making any approximation other than the dependence on ϕ:

$$q_\phi(\mathbf{z}_{1:T}, \{\mathbf{z}_t^{1:K}\}_{t=j+1}^{T}|\mathbf{x}_{1:T}, \mathbf{u}_{1:T}) = \underbrace{q_\phi(\mathbf{z}_{1:j}|\mathbf{x}_{1:T}, \mathbf{u}_{1:j})}_{\text{State Initialization}} \tag{19}$$

$$\times \underbrace{\prod_{t=j+1}^{T} q_\phi^{(t)}(\mathbf{z}_t|\, \mathbf{z}_{t-j:t-1}, \mathbf{u}_{t-j:t}, \mathbf{x}_{t:T})}_{\text{Smoothed Transitions}} \times \underbrace{\prod_{t=j+1}^{T} q_{\text{chain}}(\mathbf{z}_t^{1:K}|\mathbf{z}_t)}_{\text{Diffusion Noising Process}}. \tag{20}$$

Note the dependence of the true and variational posteriors of the latent state $\mathbf{z}_t$ on the future observations $\mathbf{x}_{t:T}$; the same structure is used in the smoothing posterior of the DKF (Krishnan et al., 2017) and other works on deep state-space models (Girin et al., 2022). To implement this dependence, we follow Krishnan et al. (2017) and employ a neural-network to summarize the future observations $\mathbf{x}_{t:T}$ for each timestep t into a fixed-dimensional vector $\mathbf{h}_t$. We define

$$(\mathbf{h}_T, \mathbf{h}_{T-1}, \ldots \mathbf{h}_1) = F_\phi(\text{concat}(\mathbf{x}_{1:T})) \tag{21}$$

The implementation of F_ϕ used by Krishnan et al. (2017) is a backward RNN, which produces a sequence of hidden states $\mathbf{h}_T, \mathbf{h}_{T-1}, \ldots, \mathbf{h}_1$. By design each $\mathbf{h}_t$ is intended to summarize the future observations for each time step. We employ a similar implementation, for which details can be found in Appendix H.4.

C.2. ELBO

We now derive an evidence lower bound (ELBO) for our model, by finding a lower bound on the marginal log-likelihood $\log p(\mathbf{x}_{1:T}|\mathbf{u}_{1:T})$. We have

$$
\begin{aligned}
\log p\left(\mathbf{x}_{1:T}|\mathbf{u}_{1:T}\right) &= \log \int p\left(\mathbf{x}_{1:T}, \mathbf{z}_{1:T}, \left\{\mathbf{z}_t^{1:K}\right\}_{t=j+1}^{T} |\mathbf{u}_{1:T}\right) d\mathbf{Z} \\
&= \log \int q_\phi\left(\mathbf{z}_{1:T}, \left\{\mathbf{z}_t^{1:K}\right\}_{t=j+1}^{T} |\mathbf{x}_{1:T}, u_{1:T}\right) \frac{p\left(\mathbf{x}_{1:T}, \mathbf{z}_{1:T}, \left\{\mathbf{z}_t^{1:K}\right\}_{t=j+1}^{T} |\mathbf{u}_{1:T}\right)}{q_\phi\left(\mathbf{z}_{1:T}, \left\{\mathbf{z}_t^{1:K}\right\}_{t=j+1}^{T} |\mathbf{x}_{1:T}, u_{1:T}\right)} d\mathbf{Z} \\
&\geq \mathbb{E}_{q_\phi\left(\mathbf{z}_{1:T}, \left\{\mathbf{z}_t^{1:K}\right\}_{t=j+1}^{T} |\mathbf{x}_{1:T}, \mathbf{u}_{1:T}\right)} \Bigg[\\
&\qquad \log p\left(\mathbf{x}_{1:T}, \mathbf{z}_{1:T}, \left\{\mathbf{z}_t^{1:K}\right\}_{t=j+1}^{T} |\mathbf{u}_{1:T}\right) \\
&\qquad - \log q_\phi\left(\mathbf{z}_{1:T}, \left\{\mathbf{z}_t^{1:K}\right\}_{t=j+1}^{T} |\mathbf{x}_{1:T}, \mathbf{u}_{1:T}\right) \Bigg],
\end{aligned}
$$

where for shorthand we have written $d\mathbf{Z} = d\mathbf{z}_{1:T}\, d\left\{\mathbf{z}_t^{1:K}\right\}_{t=j+1}^{T}$. Substituting in Eq. (1) and Eq. (19), we have

$$
\begin{aligned}
&\log p\left(\mathbf{x}_{1:T}|\mathbf{u}_{1:T}\right) \\
&\geq \mathbb{E}_{q_\phi}\Bigg[\\
&\qquad \log p_{\eta,\theta}(\mathbf{z}_{1:j}, \mathbf{x}_{1:j}|\mathbf{u}_{1:j}) \\
&\qquad + \sum_{t=j+1}^{T} \log p_\theta^{(t)}\left(\mathbf{x}_t|\mathbf{z}_{t-j+1:t}, \mathbf{u}_{t-j+1:t}\right)) \\
&+ \sum_{t=j+1}^{T} \log p(\mathbf{z}_t^K) + \sum_{t=j+1} \sum_{k=1}^{K} \log p_\psi^{(t)}(\mathbf{z}_t^{k-1}|\mathbf{z}_t^k, \mathbf{c}_t)) \\
&\qquad - \sum_{t=1}^{j} \log q_\phi\left(\mathbf{z}_{1:j}|\mathbf{x}_{1:T}, \mathbf{u}_{1:t}\right) \\
&\qquad - \sum_{t=j+1}^{T} \log q_\phi^{(t)}\left(\mathbf{z}_t|\mathbf{x}_{t:T}, \mathbf{z}_{t-j:t-1}, \mathbf{u}_{t-j:t}\right) \\
&\qquad - \sum_{t=j+1}^{T} \sum_{k=1}^{K} \log q(\mathbf{z}_t^k|\mathbf{z}_t^{k-1})\Bigg].
\end{aligned}
$$

Grouping terms,

$$
\begin{aligned}
&\log p\left(\mathbf{x}_{1:T}|\mathbf{u}_{1:T}\right) \\
&\geq \mathbb{E}_{q_\phi}\Big[\\
&\qquad \log p_{\eta,\theta}(\mathbf{z}_{1:j},\mathbf{x}_{1:j}|\mathbf{u}_{1:j}) \\
&\qquad - \log q_\phi\left(\mathbf{z}_{1:j}|\mathbf{x}_{1:T},\mathbf{u}_{1:t}\right) \\
&\qquad + \sum_{t=j+1}^{T} \log p_\theta^{(t)}\left(\mathbf{x}_t|\mathbf{z}_{t-j+1:t},\mathbf{u}_{t-j+1:t}\right)) \\
&+ \sum_{t=j+1}^{T} \Big[\log p(\mathbf{z}_t^K) \\
&\qquad + \sum_{k=1}^{K} \log p_\psi^{(t)}(\mathbf{z}_t^{k-1}|\mathbf{z}_t^k,\mathbf{c}_t) - \log q(\mathbf{z}_t^k|\mathbf{z}_t^{k-1})\Big]. \\
&\qquad - \sum_{t=j+1}^{T} \log q_\phi^{(t)}\left(\mathbf{z}_t|\mathbf{x}_{t:T},\mathbf{z}_{t-j:t-1},\mathbf{u}_{t-j:t}\right) \\
&= \mathbb{E}_{q_\phi(\mathbf{z}_{1:T}|\mathbf{x}_{1:T},\mathbf{u}_{1:T})}\left[\log p_{\eta,\theta}(\mathbf{z}_{1:j},\mathbf{x}_{1:j}|\mathbf{u}_{1:j}) - \log q_\phi\left(\mathbf{z}_{1:j}|\mathbf{x}_{1:T},\mathbf{u}_{1:t}\right)\right] \\
&+ \sum_{t=j+1}^{T} \mathbb{E}_{q_\phi(\mathbf{z}_{1:T}|\mathbf{x}_{1:T},\mathbf{u}_{1:T})}\Big[\log p_\theta^{(t)}\left(\mathbf{x}_t|\mathbf{z}_{t-j+1:t},\mathbf{u}_{t-j+1:t}\right)\Big] \\
&+ \sum_{t=j+1}^{T} \mathbb{E}_{q_\phi(\mathbf{z}_{1:T},\left\{\mathbf{z}_t^{1:K}\right\}_{t=j+1}^{T}|\mathbf{x}_{1:T},\mathbf{u}_{1:T})}\Big[\log p(\mathbf{z}_t^K) + \sum_{k=1}^{K} \log p_\psi^{(t)}(\mathbf{z}_t^{k-1}|\mathbf{z}_t^k,\mathbf{c}_t) - \log q(\mathbf{z}_t^k|\mathbf{z}_t^{k-1})\Big] \\
&- \sum_{t=j+1}^{T} \mathbb{E}_{q_\phi(\mathbf{z}_{1:T}|\mathbf{x}_{1:T},\mathbf{u}_{1:T})}\Big[\log q_\phi^{(t)}\left(\mathbf{z}_t|\mathbf{x}_{t:T},\mathbf{z}_{t-j:t-1},\mathbf{u}_{t-j:t}\right)\Big].
\end{aligned}
$$

Expanding $\log p_{\eta,\theta}(\mathbf{z}_{1:j},\mathbf{x}_{1:j}|\mathbf{u}_{1:j}) = \log p_\eta(\mathbf{z}_{1:j}|\mathbf{u}_{1:j}) + \log p_\theta^{(t)}(\mathbf{x}_{1:j}|\mathbf{z}_{1:j},\mathbf{u}_{1:j})$,

$$
\begin{aligned}
&\log p\left(\mathbf{x}_{1:T}|\mathbf{u}_{1:T}\right) \\
&\geq \mathbb{E}_{q_\phi(\mathbf{z}_{1:T}|\mathbf{x}_{1:T},\mathbf{u}_{1:T})}\left[\log p_\eta(\mathbf{z}_{1:j}|\mathbf{u}_{1:j}) - \log q_\phi\left(\mathbf{z}_{1:j}|\mathbf{x}_{1:T},\mathbf{u}_{1:t}\right)\right] \\
&+ \sum_{t=1}^{T} \mathbb{E}_{q_\phi(\mathbf{z}_{1:T}|\mathbf{x}_{1:T},\mathbf{u}_{1:T})}\Big[\log p_\theta^{(t)}\left(\mathbf{x}_t|\mathbf{z}_{\max(t-j+1,1):t},\mathbf{u}_{\max(t-j+1,1):t}\right)\Big] \\
&+ \sum_{t=j+1}^{T} \mathbb{E}_{q_\phi(\mathbf{z}_{1:T},\left\{\mathbf{z}_t^{1:K}\right\}_{t=j+1}^{T}|\mathbf{x}_{1:T},\mathbf{u}_{1:T})}\Big[\log p(\mathbf{z}_t^K) + \sum_{k=1}^{K} \log p_\psi^{(t)}(\mathbf{z}_t^{k-1}|\mathbf{z}_t^k,\mathbf{c}_t) - \log q(\mathbf{z}_t^k|\mathbf{z}_t^{k-1})\Big] \\
&- \sum_{t=j+1}^{T} \mathbb{E}_{q_\phi(\mathbf{z}_{1:T}|\mathbf{x}_{1:T},\mathbf{u}_{1:T})}\Big[\log q_\phi^{(t)}\left(\mathbf{z}_t|\mathbf{x}_{t:T},\mathbf{z}_{t-j:t-1},\mathbf{u}_{t-j:t}\right)\Big].
\end{aligned}
$$

To obtain a minimization objective we multiply by -1 yielding the negative ELBO,

$$
\begin{aligned}
&- \log p\left(\mathbf{x}_{1:T} | \mathbf{u}_{1:T}\right) \\
&\leq \underbrace{\mathbb{E}_{q_\phi(\mathbf{z}_{1:T}|\mathbf{x}_{1:T},\mathbf{u}_{1:T})}\left[\log q_\phi\left(\mathbf{z}_{1:j}|\mathbf{x}_{1:T},\mathbf{u}_{1:t}\right) - \log p_\eta(\mathbf{z}_{1:j}|\mathbf{u}_{1:j})\right]}_{\text{Initialization Loss}} && (22) \\
&+ \underbrace{\sum_{t=1}^{T} \mathbb{E}_{q_\phi(\mathbf{z}_{1:T}|\mathbf{x}_{1:T},\mathbf{u}_{1:T})}\Big[- \log p_\theta^{(t)}\left(\mathbf{x}_t | \mathbf{z}_{\max(t-j+1,1):t}, \mathbf{u}_{\max(t-j+1,1):t}\right)\Big]}_{\text{Reconstruction Loss}} && (23) \\
&+ \underbrace{\sum_{t=j+1}^{T} \mathbb{E}_{q_\phi(\mathbf{z}_{1:T},\{\mathbf{z}_t^{1:K}\}_{t=j+1}^{T}|\mathbf{x}_{1:T},\mathbf{u}_{1:T})}\Big[- \log p(\mathbf{z}_t^K) + \sum_{k=1}^{K} \log q(\mathbf{z}_t^k|\mathbf{z}_t^{k-1}) - \log p_\psi^{(t)}(\mathbf{z}_t^{k-1}|\mathbf{z}_t^k, \mathbf{c}_t)\Big]}_{\text{Diffusion Chain}} && (24) \\
&+ \underbrace{\sum_{t=j+1}^{T} \mathbb{E}_{q_\phi(\mathbf{z}_{1:T}|\mathbf{x}_{1:T},\mathbf{u}_{1:T})}\Big[\log q_\phi^{(t)}\left(\mathbf{z}_t | \mathbf{x}_{t:T}, \mathbf{z}_{t-j:t-1}, \mathbf{u}_{t-j:t}\right)\Big]}_{\text{Negative of Posterior Entropy}} .. && (25)
\end{aligned}
$$

For convenience, we will refer to each of the four terms in Eqs. (22) to (25) as $\mathcal{L}_{\text{init}}(\phi, \eta, \theta; \mathbf{x}_{1:T}, \mathbf{u}_{1:T})$, $\mathcal{L}_{\text{recon}}(\phi, \theta; \mathbf{x}_{1:T}, \mathbf{u}_{1:T})$, $\mathcal{L}_{\text{diff}}(\phi, \psi; \mathbf{x}_{1:T}, \mathbf{u}_{1:T})$, and $\mathcal{L}_{\text{entropy}}(\phi; \mathbf{x}_{1:T}, \mathbf{u}_{1:T})$, respectively.

Yet again we show the marginalized perspective, where marginalization of the intermediate diffusion states $\mathbf{z}_t^{1:K}$ in Eq. (24) recovers the transition density $p_\psi^{(t)}(\mathbf{z}_t|\mathbf{z}_{t-j:t-1})$. Combining Eq. (24) with Eq. (25) post-marginalization, we recover the KL divergence between the smoothed transition posterior and the transition density,

$$
D_{\text{KL}}\left(q_\phi^{(t)}(\mathbf{z}_t|\mathbf{x}_{t:T}, \mathbf{z}_{t-j:t-1}) \,\middle\|\, p_\psi^{(t)}(\mathbf{z}_t|\mathbf{z}_{t-j:t-1})\right), \tag{26}
$$

again making a j-th order extension of the form given by Krishnan et al. (2015, 2017).

D. Diffusion Modeling

D.1. Background for Conditional Diffusion models

Diffusion models are a class of generative models which learn to sample from some arbitrary data distribution by learning to reverse a gradual noising process. In the conditional setting, we wish to learn a model of the form $p(\mathbf{x}|\mathbf{c})$, where $\mathbf{c}$ is some conditioning information. To formulate the diffusion objective, we first must define a forward noising process. This process may be defined in either continuous or discrete time; we present the discrete-time case here. In discrete-time, the forward noising process is a Markov chain which gradually adds Gaussian noise to the data. As shown by Ho et al. (2020), the forward noising process first takes an initial data point $\mathbf{z}^0 := \mathbf{z}$ sampled from the data, and then iteratively adds noise to produce a sequence of noisy latent states $\mathbf{z}^1, \mathbf{z}^2, \ldots, \mathbf{z}^K$. The corruption process for each step k is defined by a transition density

$$
q(\mathbf{z}^k|\mathbf{z}^{k-1}) = \mathcal{N}(\mathbf{z}^k; \sqrt{1-\beta_k}\mathbf{z}^{k-1}, \beta_k \mathbf{I}),
$$

where $0 < \beta_1, \beta_2, \ldots, \beta_K < 1$ is a fixed noise schedule,

The noise schedule is chosen such that the terminal latent state $\mathbf{z}^K$ is approximately distributed as $\mathcal{N}(0, \mathbf{I})$.

This forward process allows direct sampling of a latent $\mathbf{z}_t^k$ for some time step t since $q(\mathbf{z}_t^k|\mathbf{z}_t^0) \sim \mathcal{N}(\mathbf{z}_t^k; \sqrt{\overline{\alpha}_k}\mathbf{z}_t^0, (1-\overline{\alpha}_k)\mathbf{I})$, where $\alpha_k = (1-\beta_k)$ and $\overline{\alpha}_k = \prod_{i=1}^k \alpha_i$.

For data $\mathbf{z}$, a reversal process may be defined as

$$p_\psi^{(t)}\left(\mathbf{z}^{k-1}|\mathbf{z}^k, \mathbf{c}\right),$$

such that

$$\begin{aligned} p_\psi^{(t)}(\mathbf{z}^0|\mathbf{c}) &= \int p_\psi^{(t)}\left(\mathbf{z}^{0:K-1}|\mathbf{z}^K, \mathbf{c}\right) p(\mathbf{z}^K) d\mathbf{z}^{1:K} \\ &= \int \prod_{k=K}^{1} p_\psi^{(t)}\left(\mathbf{z}^{k-1}|\mathbf{z}^k, \mathbf{c}\right) p(\mathbf{z}^K) d\mathbf{z}^{1:K} \end{aligned} \tag{27}$$

where $p_\psi^{(t)}(\mathbf{z}^{k-1}|\mathbf{z}^k, \mathbf{c})$ is learned by a neural network with parameters ψ. Typically, a single neural network with shared parameters ψ is used to learn the reverse process across all noise levels, with the noise level k fed in as an additional input to the network.

Training this model can be accomplished by minimizing the variational bound on the negative log-likelihood of the data, we provide details in Appendix C.2 as this is key to our methodology. Sampling from the diffusion model is accomplished by first sampling $\mathbf{z}^K \sim \mathcal{N}(0, \mathbf{I})$, and then sampling $\mathbf{z}^{K-1}, \mathbf{z}^{K-2}, \ldots, \mathbf{z}^0$ using the learned reverse process, however more efficient samplers are used in practice. We refer the reader to Ho et al. (2020); Song et al. (2020); Karras et al. (2022) for more details on diffusion models.

D.2. Diffusion Reparameterization

We follow prior works and reparameterize the diffusion model in terms of a noise prediction network $F_\psi(\cdot)$, which predicts the noise added at each step of the diffusion process, so that the diffusion loss may be written as

$$\begin{aligned} \mathcal{L}_{\widehat{\text{diff}}}(\phi, \psi; \mathbf{x}_{1:T}, \mathbf{u}_{1:T}) = \sum_{t=j+1}^{T} \mathbb{E}_{q_\phi(\mathbf{z}_{1:T}|\mathbf{x}_{1:T}, \mathbf{u}_{1:T})} \Big[& D_{\text{KL}}\big(q(\mathbf{z}_t^K|\mathbf{z}_t^0) \,\big\|\, p(\mathbf{z}_t^K)\big) \\ & + \mathbb{E}_{\epsilon\sim\mathcal{N}(0,\mathbf{I}), k\sim\text{Unif}\{1,\ldots,K\}} \left[K\tilde{w}_k ||F_\psi(\mathbf{z}_t^k, \mathbf{c}_t, k) - \mathbf{y}(\mathbf{z}_t^0, k)||_2^2 \right] \Big]. \end{aligned} \tag{28}$$

The derivation and motivation for Eq. (28) is given in Appendix E. In the above equation, F_ψ denotes the neural network parameterizing the reverse diffusion process, conditioned on the noisy latent $\mathbf{z}_t^k$, the Markov history $\mathbf{c}_t$, and the diffusion step k. The target $\mathbf{y}(\mathbf{z}_t^0, k)$ is a function of the clean latent $\mathbf{z}_t^0$ and the preconditioned noisy latent, as defined in Appendix E (see Eq. (37)). The coefficient $\tilde{w}_k$ weights the loss at each noise level and is given by Eq. (41) and its subsequent scaling.

We have written the diffusion loss as $\mathcal{L}_{\widehat{\text{diff}}}$ to distinguish it from the KL form in Eq. (26).

E. Derivation of the diffusion loss

E.1. Derivation of Noise-Prediction Loss

We now proceed with describing our specific loss for the diffusion transitions. Notice that the diffusion model is trained under an expectation over the variational distribution q_ϕ. This means that the loss of the diffusion model is informative to the encoder, so we must carefully consider the full ELBO objective during our derivation.

With the goal of maximizing the log-likelihood of $p_\psi^{(t)}(\mathbf{z}_t^0|\mathbf{c}_t)$, we can derive a variational bound on the negative log-likelihood of the transition density. We have

$$
\begin{aligned}
&-\log p_\psi^{(t)}(\mathbf{z}_t^0|\mathbf{c}_t) \\
&= -\log \int p_\psi^{(t)}(\mathbf{z}_t^{0:K-1}|\mathbf{z}_t^K, \mathbf{c}_t) p(\mathbf{z}_t^K) d\mathbf{z}_t^{1:K} && \text{(29a)} \\
&= -\log \int q(\mathbf{z}_t^{1:K}|\mathbf{z}_t^0) \frac{p_\psi^{(t)}(\mathbf{z}_t^{0:K-1}|\mathbf{z}_t^K, \mathbf{c}_t) p(\mathbf{z}_t^K)}{q(\mathbf{z}_t^{1:K}|\mathbf{z}_t^0)} d\mathbf{z}_t^{1:K} && \text{(29b)} \\
&\leq \mathbb{E}_{q(\mathbf{z}_t^{1:K}|\mathbf{z}_t^0)} \left[-\log \frac{p_\psi^{(t)}(\mathbf{z}_t^{0:K-1}|\mathbf{z}_t^K, \mathbf{c}_t) p(\mathbf{z}_t^K)}{q(\mathbf{z}_t^{1:K}|\mathbf{z}_t^0)} \right] && \text{(29c)} \\
&= \mathbb{E}_{q(\mathbf{z}_t^{1:K}|\mathbf{z}_t^0)} \left[-\log p(\mathbf{z}_t^K) - \sum_{k=1}^{K} \log \frac{p_\psi^{(t)}(\mathbf{z}_t^{k-1}|\mathbf{z}_t^k, \mathbf{c}_t)}{q(\mathbf{z}_t^k|\mathbf{z}_t^{k-1}, \mathbf{z}_t^0)} \right] && \text{(29d)} \\
&= \mathbb{E}_{q(\mathbf{z}_t^{1:K}|\mathbf{z}_t^0)} \left[-\log p(\mathbf{z}_t^K) - \sum_{k=2}^{K} \log \frac{p_\psi^{(t)}(\mathbf{z}_t^{k-1}|\mathbf{z}_t^k, \mathbf{c}_t)}{q(\mathbf{z}_t^k|\mathbf{z}_t^{k-1}, \mathbf{z}_t^0)} - \log \frac{p_\psi^{(t)}(\mathbf{z}_t^0|\mathbf{z}_t^1, \mathbf{c}_t)}{q(\mathbf{z}_t^1|\mathbf{z}_t^0)} \right] && \text{(29e)} \\
&= \mathbb{E}_{q(\mathbf{z}_t^{1:K}|\mathbf{z}_t^0)} \Big[-\log p(\mathbf{z}_t^K) - \sum_{k=2}^{K} \log \frac{p_\psi^{(t)}(\mathbf{z}_t^{k-1}|\mathbf{z}_t^k, \mathbf{c}_t)}{q(\mathbf{z}_t^{k-1}|\mathbf{z}_t^k, \mathbf{z}_t^0)} \cdot \frac{q(\mathbf{z}_t^{k-1}|\mathbf{z}_t^0)}{q(\mathbf{z}_t^k|\mathbf{z}_t^0)} \\
&\qquad - \log \frac{p_\psi^{(t)}(\mathbf{z}_t^0|\mathbf{z}_t^1, \mathbf{c}_t)}{q(\mathbf{z}_t^1|\mathbf{z}_t^0)} \Big] && \text{(29f)} \\
&= \mathbb{E}_{q(\mathbf{z}_t^{1:K}|\mathbf{z}_t^0)} \Big[-\log \frac{p(\mathbf{z}_t^K)}{q(\mathbf{z}_t^K|\mathbf{z}_t^0)} - \sum_{k=2}^{K} \log \frac{p_\psi^{(t)}(\mathbf{z}_t^{k-1}|\mathbf{z}_t^k, \mathbf{c}_t)}{q(\mathbf{z}_t^{k-1}|\mathbf{z}_t^k, \mathbf{z}_t^0)} - \log p_\psi^{(t)}(\mathbf{z}_t^0|\mathbf{z}_t^1, \mathbf{c}_t) \Big] && \text{(29g)} \\
&= D_{\mathrm{KL}}\big(q(\mathbf{z}_t^K|\mathbf{z}_t^0) \,\|\, p(\mathbf{z}_t^K)\big) + \sum_{k=2}^{K} \mathbb{E}_{q(\mathbf{z}_t^k|\mathbf{z}_t^0)} \left[D_{\mathrm{KL}}\Big(q(\mathbf{z}_t^{k-1}|\mathbf{z}_t^k, \mathbf{z}_t^0) \,\Big\|\, p_\psi^{(t)}(\mathbf{z}_t^{k-1}|\mathbf{z}_t^k, \mathbf{c}_t) \Big) \right] \\
&\qquad - \mathbb{E}_{q(\mathbf{z}_t^1|\mathbf{z}_t^0)} \left[\log p_\psi^{(t)}(\mathbf{z}_t^0|\mathbf{z}_t^1, \mathbf{c}_t) \right]. && \text{(29h)}
\end{aligned}
$$

This derivation is very similar to the one presented by Ho et al. (2020), except for the presence of conditioning information. The only additional step needed is made on the first line, where we utilize that the forward process is designed so that $\mathbf{z}_t^K \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$. This implies that $p(\mathbf{z}_t^K|\mathbf{z}_{t-j:t-1}) = p(\mathbf{z}_t^K)$, allowing the simplification seen after applying the chain rule in Eq. (29a). Also note that the use of conditioning information here is exactly the same in function as presented by Tashiro et al. (2021), and the included conditioning information is not limited to the previous latent states.

Equation (29h) reveals that training the diffusion model is accomplished by simply teaching the diffusion model to learn the reverse process by matching the reverse transition density to the forward transition density at each noise scale.

The first term in Eq. (29h) is constant with respect to ψ, and solely depends on the noise schedule and the initial latent state $\mathbf{z}_t^0$. Most works simply drop this term, since $\mathbf{z}_t^0$ in their case is usually data, and not trainable. In our case $\mathbf{z}_t^0$ is sampled from the encoder, and therefore we must keep this first term to respect the true ELBO.

We denote this term as $\mathcal{L}_{\text{forward}}(\mathbf{z}_t^0)$, and compute it in closed form as

$$
\begin{aligned}
D_{\mathrm{KL}}\big(q(\mathbf{z}_t^K|\mathbf{z}_t^0) \,\|\, p(\mathbf{z}_t^K)\big) &= D_{\mathrm{KL}}\big(\mathcal{N}(\sqrt{\overline{\alpha}_K}\mathbf{z}_t^0, (1-\overline{\alpha}_K)\mathbf{I}) \,\|\, \mathcal{N}(0, \mathbf{I})\big) \\
&= \frac{1}{2}\left[d(1-\overline{\alpha}_K) + \overline{\alpha}_K ||\mathbf{z}_t^0||^2 - d - d\log(1-\overline{\alpha}_K) \right] \\
&= \frac{1}{2}\left[\overline{\alpha}_K ||\mathbf{z}_t^0||^2 - d\overline{\alpha}_K - d\log(1-\overline{\alpha}_K) \right] \\
&:= \mathcal{L}_{\text{forward}}(\mathbf{z}_t^0).
\end{aligned}
$$

For the second term, Ho et al. (2020) uses Bayes rule to derive the parameterization of the fixed reverse direction kernel as

$$q(\mathbf{z}_t^{k-1}|\mathbf{z}_t^k, \mathbf{z}_t^0) = \mathcal{N}(\mathbf{z}_t^{k-1}; \tilde{\mu}_k(\mathbf{z}_t^k, \mathbf{z}_t^0), \tilde{\beta}_k \mathbf{I}),$$
$$\text{where } \tilde{\mu}_k(\mathbf{z}_t^k, \mathbf{z}_t^0) = \frac{\sqrt{\overline{\alpha}_{k-1}}\beta_k}{1-\overline{\alpha}_k}\mathbf{z}_t^0 + \frac{\sqrt{\alpha_k}(1-\overline{\alpha}_{k-1})}{1-\overline{\alpha}_k}\mathbf{z}_t^k \text{ and } \tilde{\beta}_k = \frac{1-\overline{\alpha}_{k-1}}{1-\overline{\alpha}_k}\beta_k.$$

To turn the training objective of the diffusion model into a regression problem, we must parameterize the reverse direction kernel $p_\psi^{(t)}(\mathbf{z}_t^{k-1}|\mathbf{z}_t^k, \mathbf{c}_t)$ in a way that allows us to reparameterize the KL divergence as a noise prediction error. We continue to follow Ho et al. (2020), and parameterize the reverse direction kernel as $p_\psi^{(t)}(\mathbf{z}_t^{k-1}|\mathbf{z}_t^k, \mathbf{c}_t) = \mathcal{N}(\mathbf{z}_t^{k-1}; \mu_\psi(\mathbf{z}_t^k, k, \mathbf{c}_t), \sigma^2\mathbf{I})$. Then, we may reparameterize the predicted mean μ_ψ as

$$\mu_\psi(\mathbf{z}_t^k, k, \mathbf{c}_t) = \frac{1}{\sqrt{\alpha_k}}\left(\mathbf{z}_t^k - \frac{\beta_k}{\sqrt{1-\overline{\alpha}_k}}\epsilon_\psi(\mathbf{z}_t^k, k, \mathbf{c}_t)\right), \tag{30}$$

where ϵ_ψ is a neural network with parameters ψ that predicts the noise added at step k.

Under this reparameterization, Ho et al. (2020) shows that the second KL term of Eq. (29h) for a fixed k is equivalent to

$$\mathbb{E}_{\mathbf{z}_t^0 \sim q, \epsilon \sim \mathcal{N}(\mathbf{0}, \mathbf{I})}\left[\frac{\beta_k^2}{2\sigma_k^2 \alpha_k (1-\overline{\alpha}_k)}\left\|\epsilon - \epsilon_\psi\left(\sqrt{\overline{\alpha}_k}\mathbf{z}_t^0 + \sqrt{1-\overline{\alpha}_k}\epsilon, k, \mathbf{c}_t\right)\right\|_2^2\right] + C_k, \tag{31}$$

where C_k is solely a function of the noise schedule.

For the last term of Eq. (29h), observe that when $k = 1$, that

$$D_{\mathrm{KL}}\left(q(\mathbf{z}_t^{k-1}|\mathbf{z}_t^k, \mathbf{z}_t^0)\,\middle\|\, p_\psi^{(t)}(\mathbf{z}_t^{k-1}|\mathbf{z}_t^k, \mathbf{c}_t)\right) = D_{\mathrm{KL}}\left(q(\mathbf{z}_t^0|\mathbf{z}_t^1, \mathbf{z}_t^0)\,\middle\|\, p_\psi^{(t)}(\mathbf{z}_t^0|\mathbf{z}_t^1, \mathbf{c}_t)\right).$$

The term $q(\mathbf{z}_t^0|\mathbf{z}_t^1, \mathbf{z}_t^0)$ is a Dirac delta, i.e., it has zero entropy and

$$D_{\mathrm{KL}}\left(q(\mathbf{z}_t^0|\mathbf{z}_t^1, \mathbf{z}_t^0)\,\middle\|\, p_\psi^{(t)}(\mathbf{z}_t^0|\mathbf{z}_t^1, \mathbf{c}_t)\right) = -\mathbb{E}_{q(\mathbf{z}_t^1|\mathbf{z}_t^0)}\left[\log p_\psi^{(t)}(\mathbf{z}_t^0|\mathbf{z}_t^1, \mathbf{c}_t)\right].$$

It follows that the last term in Eq. (29h) is equivalent to the second term of Eq. (29h) with $k = 1$, allowing us to combine these two terms into a single sum.

Rewriting Eq. (29h) with the above results, we have

$$\begin{aligned}&\mathbb{E}_{\mathbf{z}_t \sim q_\phi^{(t)}}[-\log p_\psi^{(t)}(\mathbf{z}_t^0|\mathbf{c}_t)] \\ &\leq \mathbb{E}_{\mathbf{z}_t^0 \sim q_\phi^{(t)}}\left[\mathcal{L}_{\text{forward}}(\mathbf{z}_t^0) + \underbrace{\sum_{k=1}^{K}\mathbb{E}_{\epsilon \sim \mathcal{N}(\mathbf{0}, \mathbf{I})}\left[w_k\left\|\epsilon - \epsilon_\psi\left(\tilde{\mathbf{z}}_t^k, k, \mathbf{c}_t\right)\right\|_2^2\right]}_{:=\mathcal{L}_{\text{diff}}} + C\right].\end{aligned} \tag{32}$$

where $w_k = \frac{\beta_k}{2\alpha_k(1-\overline{\alpha}_k)}$, $\tilde{\mathbf{z}}_t^k = \sqrt{\overline{\alpha}_k}\mathbf{z}_t^0 + \sqrt{1-\overline{\alpha}_k}\epsilon$, and C is summation of the C_k terms.

As C still is solely a function of the noise schedule, we can drop it from the optimization objective.

E.2. Preconditioning as a Variational Bound

While training standard diffusion models, the signal-to-noise ratio of the noisy inputs to the diffusion model changes drastically across different noise levels. This results in the magnitude of the regression targets for the noise prediction model to fluctuate, and thus the magnitude of the gradients of the diffusion model to fluctuate drastically as well. As the diffusion model is being trained alongside

the encoder and decoder, this results in high variance gradients for the encoder and decoder, which may lead to instability in training.

To sidestep this issue and ensure stability of the gradients, we employ the preconditioning introduced by Karras et al. (2022), where the proposed preconditioning re-scales the inputs and outputs of the neural network targets. This rescaling enforces that magnitudes are roughly unit variance across all noise levels, and thus the gradients of the diffusion model are more stable across noise levels.

The purpose of this section is to show that the preconditioning of Karras et al. (2022) is a proper bound on the original log-likelihood $\log p_\psi^{(t)}(\mathbf{z}_t^0|\mathbf{c}_t)$, given proper weighting of the objective. In turn, we detail the optimization objective for the diffusion transitions for our model, specifically Eq. (5a) from the main text.

To align with the framework from Karras et al. (2022), we re-scale our forward process, writing

$$\hat{\mathbf{z}}_t^k = \frac{\tilde{\mathbf{z}}_t^k}{\sqrt{\overline{\alpha}_k}} \tag{33}$$

$$= \frac{\sqrt{\overline{\alpha}_k}\mathbf{z}_t^0 + \sqrt{1-\overline{\alpha}_k}\epsilon}{\sqrt{\overline{\alpha}_k}} \tag{34}$$

$$= \mathbf{z}_t^0 + \sqrt{\frac{1-\overline{\alpha}_k}{\overline{\alpha}_k}}\epsilon, \tag{35}$$

where we have the additive noise as

$$\sigma_k = \sqrt{\frac{1-\overline{\alpha}_k}{\overline{\alpha}_k}}.$$

From Karras et al. (2022), the preconditioned denoiser is written as

$$D_\psi(\hat{\mathbf{z}}_t^k, \sigma_k, \mathbf{c}_t) = c_{\text{skip}}(\sigma_k)\hat{\mathbf{z}}_t^k + c_{\text{out}}(\sigma_k)F_\psi\left(c_{\text{in}}(\sigma_k)\hat{\mathbf{z}}_t^k, c_{\text{noise}}(\sigma_k), \mathbf{c}_t\right).$$

where F_ψ is a neural network with parameters ψ, and the coefficients c are those used by Karras et al. (2022) (see Table 1, in the "Ours" column) with $\sigma_{\text{data}} = 1$. For reference, these coefficients are

$$c_{\text{skip}}(\sigma_k) = \frac{1}{\sigma_k^2+1}, \qquad c_{\text{out}}(\sigma_k) = \frac{\sigma_k}{\sqrt{\sigma_k^2+1}}, c_{\text{in}}(\sigma_k) = \frac{1}{\sqrt{\sigma_k^2+1}}, \qquad c_{\text{noise}}(\sigma_k) = \frac{1}{4}\log(\sigma_k).$$

To recover our original objective, we use that $\epsilon = \frac{\hat{\mathbf{z}}_t^k - \mathbf{z}_t^0}{\sigma_k}$ to recover the predicted noise,

$$\epsilon_\psi\left(\tilde{\mathbf{z}}_t^k, k, \mathbf{c}_t\right) = \frac{\hat{\mathbf{z}}_t^k - D_\psi(\hat{\mathbf{z}}_t^k, \sigma_k, \mathbf{c}_t)}{\sigma_k}.$$

We now may find a new form for the noise prediction error term in Eq. (32). We have

$$\begin{aligned} w_k\left\|\epsilon - \epsilon_\psi\left(\tilde{\mathbf{z}}_t^k, k, \mathbf{c}_t\right)\right\|_2^2 &= w_k\left\|\frac{\hat{\mathbf{z}}_t^k - \mathbf{z}_t^0}{\sigma_k} - \frac{\hat{\mathbf{z}}_t^k - D_\psi(\hat{\mathbf{z}}_t^k, \sigma_k, \mathbf{c}_t)}{\sigma_k}\right\|_2^2 \\ &= \frac{w_k}{\sigma_k^2}\left\|D_\psi(\hat{\mathbf{z}}_t^k, \sigma_k, \mathbf{c}_t) - \mathbf{z}_t^0\right\|_2^2 \\ &= \frac{w_k}{\sigma_k^2}\left\|c_{\text{out}}(\sigma_k)F_\psi + c_{\text{skip}}(\sigma_k)\hat{\mathbf{z}}_t^k - \mathbf{z}_t^0\right\|_2^2 \\ &= \frac{w_k c_{\text{out}}(\sigma_k)^2}{\sigma_k^2}\left\|F_\psi - \frac{\mathbf{z}_t^0 - c_{\text{skip}}(\sigma_k)\hat{\mathbf{z}}_t^k}{c_{\text{out}}(\sigma_k)}\right\|_2^2, \end{aligned} \tag{36}$$

where to simplify notation going forward we will write the regression targets as

$$\mathbf{y}(\mathbf{z}_t^0, k) = \frac{\mathbf{z}_t^0 - c_{\text{skip}}(\sigma_k)\hat{\mathbf{z}}_t^k}{c_{\text{out}}(\sigma_k)}, \tag{37}$$

and the coefficient in front of the squared error as

$$\tilde{w}_k = \frac{w_k c_{\text{out}}(\sigma_k)^2}{\sigma_k^2}.$$

We substitute our value for $c_{\text{out}}(\sigma_k)$ and obtain our diffusion loss for a given t as

$$\mathcal{L}_{\text{diff}} = \sum_{k=1}^{K} \mathbb{E}_{\epsilon \sim \mathcal{N}(\mathbf{0},\mathbf{I})} \left[\tilde{w}_k \left\| F_\psi(\cdot) - \mathbf{y}(\mathbf{z}_t^0, k) \right\|_2^2 \right]. \tag{38}$$

Directly implementing the summation in Eq. (38) is prohibitively expensive since it requires a forward pass through the diffusion model K times per example. Instead, we employ a Monte-Carlo approximation by sampling k from an arbitrary density $\rho(k)$ on $\{1, \ldots, K\}$ such that

$$\mathcal{L}_{\text{diff}} = \sum_{k=1}^{K} \mathbb{E}_{\epsilon \sim \mathcal{N}(\mathbf{0},\mathbf{I})} \left[\tilde{w}_k \left\| F_\psi(\cdot) - \mathbf{y}(\mathbf{z}_t^0, k) \right\|_2^2 \right] \tag{39}$$

$$= \mathbb{E}_{k \sim \rho(k), \epsilon \sim \mathcal{N}(\mathbf{0},\mathbf{I})} \left[\frac{\tilde{w}_k}{\rho(k)} \left\| F_\psi(\cdot) - \mathbf{y}(\mathbf{z}_t^0, k) \right\|_2^2 \right]. \tag{40}$$

The choice of $\rho(k)$ affects the variance of the Monte-Carlo estimate. The coefficient $\tilde{w}_k$ is typically viewed as a weight, where in our case we have

$$\tilde{w}_k = \frac{w_k c_{\text{out}}(\sigma_k)^2}{\sigma_k^2} \tag{41}$$

$$= w_k \cdot \overline{\alpha}_k \tag{42}$$

$$= \frac{\beta_k \overline{\alpha}_k}{2\alpha_k(1 - \overline{\alpha}_k)}. \tag{43}$$

Many works drop the weights and perform uniform sampling, implicitly reweighting the objective. Dropping this term imbalances the terms in the ELBO. Vahdat et al. (2021) also trains a diffusion model end-to-end in a VAE framework, and highlight this issue as well; during training of the encoder, Vahdat et al. (2021) choose to retain the weights so that the encoder is encouraged to match the true posterior of the diffusion model. We provide an alternative perspective on this issue by noting that the weights $\tilde{w}_k$ are very large for small k (small noise levels), and very small for large k (large noise levels). Intuitively, this means that maximizing the ELBO penalizes errors in the noise prediction at small noise levels, where the signal-to-noise ratio is high. On the other hand, errors in noise prediction at large noise levels are not penalized as much, where the diffusion model is focused on denoising the lower-frequency structure of the sample. In the context of our full model, this property implies that the strongest regularization of the encoder comes from the small noise levels, penalizing the encoder for producing latent samples that cannot reveal the fine-grained structure of the data. We retain the weights in our implementation, as we do not separately train the diffusion model from the encoder. Lastly, we set $\rho(k) = \frac{1}{K}$ for uniform sampling, to ensure that the Monte-Carlo estimate is unbiased without requiring any additional weighting of the loss.

F. Experiment Details

To ensure a fair comparison for our synthetic experiments, we fix architectural hyperparameters across the DKF and DDSSM models, and tune hyperparameters for both models using Optuna (Akiba et al., 2019). The only difference in architecture is the implementation for the transition model, where the DKF uses a Gaussian transition module and the DDSSM uses a conditional U-Net parameterized diffusion model. We provide identical search spaces to the tuning algorithm, using the one-step JSD as the evaluation metric on a validation set of examples.

F.1. Synthetic Experiment Architectural Parameters

We construct our models using the modular elements described in Appendix H. Specifically, we define the *time mixer* (operating over the sequence length axis) and the *feature mixer* (operating across the latent or observation dimensions) for each individual module. Across all synthetic experiments, we fix the latent dimension to $M = 1$, the sequential latent lag to $j = 1$, the context channels to 16, and the dense hidden dimension to 16. Embeddings for continuous time steps are disabled, while discrete mask embeddings have dimension 8.

For both the baseline DKF and our DDSSM, the shared contextual architectures are summarized in Table 1.

Module	**Layers**	**Channels**	**Hidden Dim**	**Time Mixer**	**Feature Mixer**
Encoder Summary	3	–	16	GRU	–
Encoder Context	2	16	16	Conv ($k = 2$)	Conv ($k = 2$)
Decoder Context	2	16	16	Conv ($k = 2$)	Conv ($k = 2$)
Initialization Context	2	16	16	Identity	Conv ($k = 2$)
Gaussian Transition (DKF)	3	16	16	Identity	Conv ($k = 2$)

Table 1: Shared architectural parameters across models, including the baseline DKF transition model. The Gaussian transition relies on an identity time mixer since it only projects summary statistics for a single target step, whereas the encoder and decoder leverage temporal convolutions to process multi-step histories and future summaries.

Diffusion Transition Specifications. For the DDSSM, we substitute the Gaussian transition module with a conditional U-Net parameterized diffusion model. Unlike the Gaussian baseline, the diffusion denoiser utilizes a convolutional time mixer rather than an identity map. This is because at each diffusion step k, the transition model observes both the clean history states $\mathbf{z}_{t-j:t-1}$ and the noisy target state $\mathbf{z}_t^k$, requiring explicit temporal mixing. We summarize the network sizing and diffusion schedule parameters in Table 2.

Parameter	**Value**
U-Net Layers	3
U-Net Channels (Blocks)	16
Time Mixer	Conv ($k = 2$)
Feature Mixer	Conv ($k = 2$)
Diffusion Step Embedding Space	64 (projected to 128)
Feature Embedding Dimension	8
Diffusion Schedule	
Number of Diffusion Steps (K)	128
Noise Schedule	Linear

Table 2: Architectural and schedule parameters specific to the diffusion-based transition model in DDSSM.

F.1.1. Hyperparameters and Tuning

We use Optuna (Akiba et al., 2019) for hyperparameter tuning. Each trial consists of training a model for 1500 steps and evaluating the CRPS-sum metric on unseen validation data. We use 1024 training examples, 1024 validation examples, and 1024 test examples for the synthetic experiment, each generated by simulating the same underlying process with a fixed random seed.

The parameters we tune, their tuning ranges, and the optimal parameters found for the DKF and DDSSM models are summarized in Tables 3 and 4.

Parameter	**Tuning Range**	**DKF**	**DDSSM**
lambda_schedule	linear, cosine	cosine	cosine
lambda_warmup_steps	200 – 1200	867	889
lambda_end	0.3 – 2	1.245	1.243
enc_lr	5×10^{-5} – 10^{-2} (log)	0.00887	0.000864
dec_lr	5×10^{-5} – 10^{-2} (log)	0.00777	0.000300
trans_lr	5×10^{-5} – 10^{-2} (log)	0.000099	0.00941
zinit_lr	10^{-4} – 10^{-3} (log)	0.005	0.000801
S	1 – 4	2	2
batch_size	64, 128, 256, 512	512	128
transition.schedule.S_k	1 – 8 (Diffusion only)	1	5

Table 3: Tuning ranges and optimal parameters found for DKF and DDSSM models.

Hyperparameter	**Fixed Value**
Number of training steps	1500 (tuning), 3000 (final)
Number of Optuna trials	50
Number of startup trials	10
Encoder/decoder channels	16
Encoder/decoder layers	2
Weight decay	0.01

Table 4: Hyperparameters fixed across all models during tuning.

G. Variational Hierarchical Prior

Klushyn et al. (2021, 2019), consider the problem of defining the prior distribution over the initial latent state $p(\mathbf{z}_1)$ in the $j = 1$ case. Typically, the prior is chosen by DSSMs to be a standard Gaussian Krishnan et al. (2015, 2017); Karl et al. (2016). The prior over $\mathbf{z}_1$ however regularizes the rest of the latent trajectory, as both the transition model and the encoder posterior are conditioned on $\mathbf{z}_1$. This is experimentally validated by Klushyn et al. (2019, 2021), where it is demonstrated that a more flexible prior over the initial latent state can lead to better inference about the latent trajectory.

As our diffusion transition model can be highly expressive, we wish to avoid inhibiting the expressiveness of the transition model by imposing a simple prior over the initial latent states. We therefore follow Klushyn et al. (2021) and use an expressive initial distribution via a variational hierarchical prior (VHP).

In our case, we wish to handle higher Markov orders, so our initial state distribution is over the first j latents, $p(\mathbf{z}_{1:j})$. For the $j = 1$ case, Klushyn et al. (2021) uses a two-level hierarchical prior, where the initial latent state $\mathbf{z}_1$ is modeled as

$$p(\mathbf{z}_1) = \int p_\eta(\mathbf{z}_1|\mathbf{z}_0)p(\mathbf{z}_0)d\mathbf{z}_0,$$

where $p(\mathbf{z}_0)$ is a standard Gaussian. This choice mimics the fact that the optimal empirical bayes prior for a VAE is $p*(\mathbf{z}_1) = \mathbb{E}_{p(\mathbf{x}_{1:T})}[q_\phi(\mathbf{z}_1|\mathbf{x}_{1:T})]$.

We generalize this to the j-order Markovian setting by introducing j auxiliary variables $\mathbf{z}_{-j+1:0}$. The conditional distribution $p_\eta(\mathbf{z}_{1:j}|\mathbf{z}_{-j+1:0})$ is given by the chain rule,

$$\begin{aligned} p_\eta(\mathbf{z}_{1:j}|\mathbf{z}_{-j+1:0}) &= \prod_{t=1}^{j} p_\eta(\mathbf{z}_t|\mathbf{z}_{t-1}, \ldots, \mathbf{z}_{-j+1}) \\ &= \prod_{t=1}^{j} p_\eta(\mathbf{z}_t|\mathbf{z}_{t-j:t-1}), \end{aligned}$$

where we have used the j-order Markovian property in the second line. Marginalizing over the auxiliary variables, we have

$$\begin{aligned} p_\eta(\mathbf{z}_{1:j}) &= \int p_\eta(\mathbf{z}_{1:j}|\mathbf{z}_{-j+1:0}) p(\mathbf{z}_{-j+1:0}) d\mathbf{z}_{-j+1:0} \\ &= \int \prod_{t=1}^{j} p_\eta(\mathbf{z}_t|\mathbf{z}_{t-j:t-1}) p(\mathbf{z}_{-j+1:0}) d\mathbf{z}_{-j+1:0}. \end{aligned}$$

Continuing, we introduce the variational distribution $q_\Phi(\mathbf{z}_{-j+1:0}|\mathbf{z}_{1:j})$, parameterized by Φ, to approximate the posterior over the auxiliary variables. We find the bound on the optimal expected marginal log-likelihood of $p(\mathbf{z}_{1:j})$ as

$$\begin{aligned} &\mathbb{E}_{p*(\mathbf{z}_{1:j})}[\log p_\eta(\mathbf{z}_{1:j})] \\ &\geq \mathbb{E}_{p(\mathbf{x}_{1:T})} \mathbb{E}_{q_\phi(\mathbf{z}_{1:T}|\mathbf{x}_{1:T}, \mathbf{u}_{1:T})} \left[\mathbb{E}_{q_\Phi(\mathbf{z}_{-j+1:0}|\mathbf{z}_{1:j})} \left[\log \frac{p_\eta(\mathbf{z}_{1:j}|\mathbf{z}_{-j+1:0}) p(\mathbf{z}_{-j+1:0})}{q_\Phi(\mathbf{z}_{-j+1:0}|\mathbf{z}_{1:j})} \right] \right] \\ &= \mathbb{E}_{p(\mathbf{x}_{1:T})} \mathbb{E}_{q_\phi(\mathbf{z}_{1:T}|\mathbf{x}_{1:T}, \mathbf{u}_{1:T})} \left[\mathbb{E}_{q_\Phi(\mathbf{z}_{-j+1:0}|\mathbf{z}_{1:j})} \left[\log p_\eta(\mathbf{z}_{1:j}|\mathbf{z}_{-j+1:0}) + \log p(\mathbf{z}_{-j+1:0}) - \log q_\Phi(\mathbf{z}_{-j+1:0}|\mathbf{z}_{1:j}) \right] \right] \\ &= \mathbb{E}_{p(\mathbf{x}_{1:T})} \mathbb{E}_{q_\phi(\mathbf{z}_{1:T}|\mathbf{x}_{1:T}, \mathbf{u}_{1:T})} \left[\mathbb{E}_{q_\Phi(\mathbf{z}_{-j+1:0}|\mathbf{z}_{1:j})} \left[\sum_{t=1}^{j} \log p_\eta(\mathbf{z}_t|\mathbf{z}_{t-j:t-1}) \right] \right. \\ &\qquad \left. - D_{\mathrm{KL}}(q_\Phi(\mathbf{z}_{-j+1:0}|\mathbf{z}_{1:j}) \,\|\, p(\mathbf{z}_{-j+1:0})) \right], \end{aligned}$$

where in the last line we have expanded the conditional prior and recognized the KL divergence. In line with Klushyn et al. (2021), we set $p(\mathbf{z}_{-j+1:0}) = \mathcal{N}(0, \mathbf{I})$, and parameterize $q_\Phi(\mathbf{z}_{-j+1:0}|\mathbf{z}_{1:j})$ as a diagonal Gaussian with mean and variance given by neural networks that take $\mathbf{z}_{1:j}$ as input.

Note that in the variational posterior, $q_\phi(\mathbf{z}_{1:j}|\mathbf{x}_{1:T}, \mathbf{u}_{1:T})$ depends on all observations and inputs, while in the prior $p_\eta(\mathbf{z}_{1:j}|\mathbf{z}_{-j+1:0})$ only depends on the auxiliary variables. Therefore it is unnecessary to provide a hierarchical structure to the initial latent states for the variational posterior, as the variational posterior can already leverage the full context of the observations and inputs to infer the initial latent states. As we implement a network to parameterize $q_\phi^{(t)}(\mathbf{z}_t|\mathbf{z}_{t-j:t-1}, \mathbf{x}_{t:T}, \mathbf{u}_{t-j:t})$, we can simply use the same network to parameterize $q_\phi(\mathbf{z}_{1:j}|\mathbf{x}_{1:T}, \mathbf{u}_{1:T})$ by providing zero inputs for $\mathbf{z}_{t-j:t-1}$ when $t \leq j$. These drawn latent states are then used for drawing from $q_\Phi(\mathbf{z}_{-j+1:0}|\mathbf{z}_{1:j})$ to compute the hierarchical prior.

Rewriting $\mathcal{L}_{\text{init}}(\phi, \eta; \mathbf{x}_{1:T}, \mathbf{u}_{1:T})$ from Eq. (22) using the hierarchical prior, we have

$$\begin{aligned}
\mathcal{L}_{\text{init}}(\phi, \eta, \Phi; \mathbf{x}_{1:T}, \mathbf{u}_{1:T}) &= \mathbb{E}_{q_\phi(\mathbf{z}_{1:T}|\mathbf{x}_{1:T}, \mathbf{u}_{1:T})}\Big[\log q_\phi\left(\mathbf{z}_{1:j}|\mathbf{x}_{1:T}, \mathbf{u}_{1:t}\right) \\
&\qquad - p_\eta(\mathbf{z}_{1:j}|\mathbf{z}_{-j+1:0})p(\mathbf{z}_{-j+1:0})d\mathbf{z}_{-j+1:0}\Big] \\
&\leq \mathbb{E}_{q_\phi(\mathbf{z}_{1:T}|\mathbf{x}_{1:T}, \mathbf{u}_{1:T})}\Big[\log q_\phi\left(\mathbf{z}_{1:j}|\mathbf{x}_{1:T}, \mathbf{u}_{1:t}\right) \\
&\qquad - \mathbb{E}_{q_\Phi(\mathbf{z}_{-j+1:0}|\mathbf{z}_{1:j})}\left[\log p_\eta(\mathbf{z}_{1:j}|\mathbf{z}_{-j+1:0}) + \log p(\mathbf{z}_{-j+1:0}) - \log q_\Phi(\mathbf{z}_{-j+1:0}|\mathbf{z}_{1:j})\right]\Big] \\
&= \underbrace{\mathbb{E}_{q_\phi(\mathbf{z}_{1:T}|\mathbf{x}_{1:T}, \mathbf{u}_{1:T})}\Big[\log q_\phi\left(\mathbf{z}_{1:j}|\mathbf{x}_{1:T}, \mathbf{u}_{1:t}\right)\Big]}_{\text{negative entropy}} \\
&\qquad + \mathbb{E}_{q_\phi(\mathbf{z}_{1:T}|\mathbf{x}_{1:T}, \mathbf{u}_{1:T})}\Big[\mathbb{E}_{q_\Phi(\mathbf{z}_{-j+1:0}|\mathbf{z}_{1:j})}\Big[-\log p_\eta(\mathbf{z}_{1:j}|\mathbf{z}_{-j+1:0}) \\
&\qquad\qquad - \log p(\mathbf{z}_{-j+1:0}) + \log q_\Phi(\mathbf{z}_{-j+1:0}|\mathbf{z}_{1:j})\Big]\Big]\Big] \\
&= \underbrace{\mathbb{E}_{q_\phi(\mathbf{z}_{1:T}|\mathbf{x}_{1:T}, \mathbf{u}_{1:T})}\Big[\log q_\phi\left(\mathbf{z}_{1:j}|\mathbf{x}_{1:T}, \mathbf{u}_{1:t}\right)\Big]}_{\mathcal{L}_{\text{init}}^{\text{entropy}}(\phi; \mathbf{x}_{1:T}, \mathbf{u}_{1:T})} \\
&\qquad + \mathbb{E}_{q_\phi(\mathbf{z}_{1:T}|\mathbf{x}_{1:T}, \mathbf{u}_{1:T})}\Big[\mathbb{E}_{q_\Phi(\mathbf{z}_{-j+1:0}|\mathbf{z}_{1:j})}\Big[-\sum_{t=1}^{j}\log p_\eta(\mathbf{z}_t|\mathbf{z}_{t-j:t-1}) \\
&\qquad\qquad + D_{\text{KL}}(q_\Phi(\mathbf{z}_{-j+1:0}|\mathbf{z}_{1:j}) \,\|\, p(\mathbf{z}_{-j+1:0}))\Big]\Big]\Big] \\
&:= \mathcal{L}_{\text{init}}^{\text{entropy}}(\phi; \mathbf{x}_{1:T}, \mathbf{u}_{1:T}) + \mathcal{L}_{\text{VHP}}(\phi, \eta, \Phi; \mathbf{x}_{1:T}, \mathbf{u}_{1:T}), \qquad (44)
\end{aligned}$$

and notice that we have split the final line into two terms, the negative entropy of the variational posterior over the initial states, and the variational hierarchical prior (VHP) loss, where we have defined

$$\begin{aligned}
\mathcal{L}_{\text{VHP}}(\phi, \eta, \Phi; \mathbf{x}_{1:T}, \mathbf{u}_{1:T}) = \mathbb{E}_{q_\phi(\mathbf{z}_{1:T}|\mathbf{x}_{1:T}, \mathbf{u}_{1:T})}\Big[\mathbb{E}_{q_\Phi(\mathbf{z}_{-j+1:0}|\mathbf{z}_{1:j})}\Big[-\sum_{t=1}^{j}\log p_\eta(\mathbf{z}_t|\mathbf{z}_{t-j:t-1}) \\
+ D_{\text{KL}}(q_\Phi(\mathbf{z}_{-j+1:0}|\mathbf{z}_{1:j}) \,\|\, p(\mathbf{z}_{-j+1:0}))\Big]\Big].
\end{aligned}$$

Performing a substitution of the resulting inequality in Eq. (44) into the ELBO in Eq. (22), we obtain the ELBO which incorporates the variational hierarchical prior:

$$\begin{aligned}
&-\log p\left(\mathbf{x}_{1:T}|\mathbf{u}_{1:T}\right) \\
&\leq \mathcal{L}_{\text{init}}^{\text{entropy}}(\phi; \mathbf{x}_{1:T}, \mathbf{u}_{1:T}) + \mathcal{L}_{\text{VHP}}(\phi, \eta, \Phi; \mathbf{x}_{1:T}, \mathbf{u}_{1:T}) \\
&+ \mathcal{L}_{\text{recon}}(\phi, \theta; \mathbf{x}_{1:T}, \mathbf{u}_{1:T}) + \mathcal{L}_{\widehat{\text{diff}}}(\phi, \psi; \mathbf{x}_{1:T}, \mathbf{u}_{1:T}) + \mathcal{L}_{\text{entropy}}(\phi; \mathbf{x}_{1:T}, \mathbf{u}_{1:T}). \qquad (45)
\end{aligned}$$

which we minimize with respect to the parameters $\phi, \eta, \theta, \psi, \Phi$.

H. Model Architecture Details

The method of Tashiro et al. (2021), titled "CSDI: Conditional Score-based Diffusion Models for Probabilistic Time Series Imputation", uses a well-established architecture for modeling time-series data with diffusion models Tashiro et al. (2021), and we adapt it to our setting of autoregressive generation in the latent space. We choose to extend the architecture of Tashiro, Song, Song, and Ermon (2021), for several reasons. First, it is a well-established architecture for modeling time-series data with diffusion models, Second, its success in modeling directly in observation space suggests that it is sufficiently powerful to model the transition distribution in latent space, and it is simple

enough to allow us to modularize the architecture depending on the specific needs of the encoder, decoder, and transition modules. Specifically, we extend the U-net architecture from CSDI to not only our diffusion model, but also to the encoder, decoder, and initialization modules, which all operate on latent histories.

H.1. U-net Architecture

Because we operate in the latent space and generate autoregressively, we utilize a static mask indicating which time steps functionally serve as the conditioning history versus the generated output, rather than a dynamic missing-data mask. Our mask is thus a binary mask $\mathbf{m}_{\text{hist}}$ of shape $(j+1)$, which equals one for history steps and zero for the target step, since we are always generating a single step and conditioning on j history steps. The input to the denoiser is a tensor $\mathbf{Z} \in \mathbb{R}^{M \times (j+1)}$, where M is the latent dimension. The target noise is always located at the last time step, while the clean history is located in the preceding j time steps.

Input Projection Following Tashiro et al. (2021), we separate the input into two components via the mask: a clean history $\mathbf{Z}_{\text{hist}} = \mathbf{Z} \odot \mathbf{m}_{\text{hist}}$ and a noisy target $\mathbf{Z}_{\text{noisy}} = \mathbf{Z} \odot (1 - \mathbf{m}_{\text{hist}})$. These are concatenated along a new channel dimension to form a tensor in $\mathbb{R}^{2 \times M \times (j+1)}$. A 1×1 convolution and a ReLU activation project this input to a hidden channel dimension C, yielding an initial representation $\mathbf{H}_0 \in \mathbb{R}^{C \times M \times (j+1)}$.

Extra Conditioning Information Conditioning is injected via a continuous diffusion timestep embedding and a side information tensor. The scalar diffusion noise level is embedded via sinusoidal positional encodings and a two-layer multi-layer perceptron into $\mathbf{e}_{\text{diff}} \in \mathbb{R}^{E_{\text{diff}}}$. The side information tensor $\mathbf{S} \in \mathbb{R}^{E_{\text{side}} \times M \times (j+1)}$ concatenates dynamic covariates $\mathbf{u}_{t-j:t}$, absolute time embeddings, and the mask $\mathbf{m}_{\text{hist}}$. Since this diffusion operates on compressed latents rather than the original explicitly meaningful observation coordinates, in contrast to Tashiro et al. (2021), we do not utilize static feature embeddings here.

Residual Blocks The core of our network consists of L stacked residual blocks, each of which takes in a representation of shape $(C, M, j+1)$ and produces an output of the same shape as well as a skip connection. The residual block itself is mostly unchanged from CSDI, except we abstract the time-mixing and feature-mixing operations. This allows us to replace computationally heavy attention operations with architectures like a 1D convolution stack or a Gated Recurrent Unit (GRU) when the sequence length j is small. We refer the reader to Tashiro et al. (2021) for more details on the architecture of the residual block.

Output Projection We continue to follow CSDI for handling the residual block inputs and outputs, as well as the skip connections. However, we deviate from CSDI in the output projection step. Because we generate autoregressively, we only need to produce an output for the target time step, rather than a full sequence of length $j+1$. So, we slice the final output of the residual block stack to only keep the target time step, yielding a tensor of shape (M), containing the predicted noise for the target time step.

H.2. Context Producer

The encoder, decoder and initialization module, similar to the denoiser, operate on latent histories. Therefore for the encoder, decoder, and initialization module, we adapt the U-net architecture above as a general input-output architecture, where the inputs and outputs are tensors of shape (h, L) for some hidden dimension h and sequence length L.

The encoder, decoder, and initialization modules process sequential latent and observable data histories. To support this repetition, we abstract the U-net framework into a generalized feature-extraction module, which we call the *Context Producer*. The Context Producer accepts inputs of

shape (H_{seq}, L), where H_{seq} is the spatial or feature dimension and L is the temporal sequence length. The output is a vector representing the entire trajectory.

Input Projection Unlike the denoiser which separates clean and noisy data, the Context Producer takes a single sequence tensor $\mathbf{H}_{\text{in}} \in \mathbb{R}^{H_{\text{seq}} \times L}$. We treat this input sequence as having a single channel, flattening it temporarily to shape $(1, H_{\text{seq}} \times L)$, and apply a 1×1 convolution followed by a ReLU activation. This projects the input to a hidden channel dimension C, yielding an initial representation $\mathbf{H}_0 \in \mathbb{R}^{C \times H_{\text{seq}} \times L}$.

Extra Conditioning Information Because the Context Producer does not reverse a diffusion process, we completely omit the diffusion step embedding. As with the denoiser, we inject conditioning information via a side information tensor, but we adapt the construction of this tensor to a side-information tensor $\mathbf{S} \in \mathbb{R}^{E_{\text{side}} \times H_{\text{seq}} \times L}$, where E_{side} is the side information embedding dimension. We carefully construct this tensor to support both temporal and static covariates, as well as absolute time embeddings and the history/target mask. To construct the tensor we handle temporal and static information separately, and then concatenate them along the feature dimension. For each type of information we have:

- **Temporal Information.** Dynamic covariates, time embeddings, and valid/padding masks vary over time L but are uniform across the feature dimension H_{seq}. These are computed as a tensor of shape $(E_{\text{time}} + E_{\text{mask}}, L)$ and expanded along the feature dimension to $(E_{\text{time}} + E_{\text{mask}}, H_{\text{seq}}, L)$.
- **Static Information.** Static categorical or contextual embeddings vary across the features but are constant across time. These are mapped to a tensor $(E_{\text{static}}, H_{\text{seq}})$ and expanded along the time dimension to $(E_{\text{static}}, H_{\text{seq}}, L)$.

We concatenate the temporal and static information along the feature dimension to yield the final side information tensor $\mathbf{S} \in \mathbb{R}^{E_{\text{side}} \times H_{\text{seq}} \times L}$ where $E_{\text{side}} = E_{\text{time}} + E_{\text{mask}} + E_{\text{static}}$.

Residual Blocks The residual blocks of the context producer are unchanged from Appendix H.1, except the diffusion embedding is omitted. The utilization of the side information tensor is unchanged.

Output Projection The rest of the architecture is unchanged from our denoiser adaptation, except we do not project the (C, h, L) output back to $(1, h, L)$ before slicing as we wish to produce a dense summary of the input rather than a noise prediction for a single time step. To concisely collapse the temporal dimension, we reshape the tensor equivalently to $(C \times H_{\text{seq}}, L)$ and apply a 1D convolution over the time dimension. Standard 1D convolutions would mix representations across the feature dimension, so we strictly use a grouped 1D convolution with groups equal to $C \times H_{\text{seq}}$ and a kernel size equal to the sequence length L. This operation pools the sequence, outputting a flattened context vector in $\mathbb{R}^{C \times H_{\text{seq}}}$.

H.3. Decoder Architecture

The decoder architecture parameterizes the emissions density $p_\theta^{(t)}(\mathbf{x}_t | \mathbf{z}_{t-j+1:t}, \mathbf{u}_{t-j+1:t})$ as a diagonal Gaussian distribution over the observation $\mathbf{x}_t \in \mathbb{R}^D$. It takes the sequential latent history and extracts a summary using a Context Producer to predict the observation parameters.

Input Preparation The inputs include the length-j latent history $\mathbf{z}_{t:t-j+1}$ and covariates $\mathbf{u}_{t-j+1:t}$. For early time steps where $t < j$, the available history is left-padded with zero vectors to strongly enforce a fixed sequence length j. We include a binary mask to indicate which time steps are padded versus valid, and add this to the set of conditioning information.

The resulting sequence of shape (M, j) is processed step-wise by a linear layer mapping the latent dimension M to the Context Producer's spatial dimension H_{seq}. This outputs the primary input sequence $\mathbf{H}_{\text{in}} \in \mathbb{R}^{H_{\text{seq}} \times j}$ for the Context Producer.

Conditioning Construction The decoder's Context Producer requires side information aligned to its j-step temporal axis:

- **Temporal Information:** We compute absolute time embeddings for the j history steps. These are concatenated with the step-aligned dynamic covariates $\mathbf{u}_{t-j+1:t}$. As previously mentioned, we embed a binary padding mask—which flags valid history vectors versus left-padded zero vectors—into a continuous embedding space.
- **Static Information:** If static observation-level covariates are present, they are linearly projected from the data dimension D to the spatial hidden dimension H_{seq}. The purpose of this projection is to ease the decoder's task of matching its generated data to observed data, since the semantically meaningful static covariates are in the data space rather than the latent space.

These elements are integrated as the side information tensor $\mathbf{S}$ utilizing the mechanisms defined in Appendix H.2.

Gaussian Output The Context Producer condenses the sequence into a final structured summary vector of shape $(C \times H_{\text{seq}})$. Because the Context Producer utilizes several non-linear residual blocks, we preserve a residual path for available static covariates by mapping the flattened static covariates through an auxiliary linear layer and adding the result directly to the summary vector. This resulting context vector is passed into the Gaussian output head, which comprises two multi-layer perceptrons producing the distribution parameters: the mean $\mu_\theta(\mathbf{r}_t) \in \mathbb{R}^D$ and the log-variance $\log \Sigma_\theta(\mathbf{r}_t) \in \mathbb{R}^D$ for the components of $\mathbf{x}_t$.

H.4. Encoder Architecture

The encoder approximates the smoothed variational density $q_\phi^{(t)}(\mathbf{z}_t | \mathbf{z}_{t-j:t-1}, \mathbf{x}_{1:T}, \mathbf{u}_{t-j:t})$. As mentioned in Appendix C.1, we simplify the conditioning set by summarizing all future information into a sequence of continuous vectors $\mathbf{h}_{1:T}$, i.e. our variational family assumes $q_\phi^{(t)}(\mathbf{z}_t | \mathbf{z}_{t-j:t-1}, \mathbf{u}_{t-j:t}, \mathbf{x}_{t:T}) \approx q_\phi^{(t)}(\mathbf{z}_t | \mathbf{z}_{t-j:t-1}, \mathbf{u}_{t-j:t}, \mathbf{h}_t)$. Therefore, there are two stages to our encoder's operation: (i) a full-sequence *Future Summary* module that processes the entire future trajectory to produce a sequence of future summaries $\mathbf{h}_{1:T}$, and (ii) an autoregressive *Context Producer* that takes in the current latent history and the current future summary to produce the parameters of the variational distribution over z_t. We provide the algorithmic details of sampling a full trajectory from the encoder in Algorithm 2.

Future Summary Module The Future Summary module computes $\mathbf{h}_{1:T} = F_\phi(\mathbf{x}_{1:T}, \mathbf{m}_{\text{obs}})$. At each time step across the length-T sequence, we concatenate the observations $\mathbf{x}_t$, absolute time embeddings, observation missingness masks $\mathbf{m}_{\text{obs},t}$, and the flattened static covariates. This feature vector is linearly projected to a hidden dimension C_{summary}. In line with Krishnan et al. (2017), to ensure that $\mathbf{h}_t$ only absorbs information from the *future and present* (time$t \ldots T$), the entire projected sequence is reversed along the time dimension and passed through a causal sequence model (such as a Gated Recurrent Unit or a causally-masked Transformer). The output sequence is reversed again, restoring the original chronological order while guaranteeing that $\mathbf{h}_t \in \mathbb{R}^{C_{\text{summary}}}$ depends only on sequence indices $\geq t$.

Input Preparation Given the future summary $\mathbf{h}_t$, the encoder autoregressively infers the parameters for $\mathbf{z}_t$ contingent on previously sampled states $\mathbf{z}_{t-j:t-1}$. For the first j time steps, we left-pad the latent history with zero vectors and include a binary mask to indicate which time steps are

padded versus valid, and add this to the set of conditioning information. This is the same strategy taken by the decoder (see Appendix H.3).

We linearly project the future summary from C_{summary} to the Context Producer spatial dimension H_{seq}. Likewise, the j history vectors are mapped from the latent dimension M to H_{seq}. These are concatenated along the temporal axis to yield the primary input tensor $\mathbf{H}_{\text{in}} \in \mathbb{R}^{H_{\text{seq}} \times (j+1)}$ for the Context Producer, where the future summary is located at index 0 and the history vectors are located at indices $1 \dots j$. By placing the future summary at the beginning of the sequence, we allow the Context producer to attend to the future summary at each layer of its architecture. This choice is once again inspired by Krishnan et al. (2017), who project the future summary to the initial hidden state of their RNN-based encoder, which allows the future summary to influence the entire inference process. In fact, this is the behavior of the Context Producer when we use an RNN for the time-mixing operation in the residual blocks.

Conditioning Construction The side information for the encoder's Context Producer includes absolute time embeddings for the j history steps, as well as a binary mask indicating which of the j history steps are valid versus left-padded, and dynamical covariates for the j history steps. We additionally include static covariates, and similar to the encoder we project these from the data space to the Context Producer's spatial dimension to ease the task of matching the generated latent states to the observed data. To summarize these inputs, we have:

- **Temporal & Covariate Information:** We gather absolute time embeddings and dynamic covariates aligned with the exact $(j+1)$ slots: time t for the summary slot, and $t-j \dots t-1$ for the history slots.
- **Role Mask:** We employ a binary mask—set to 0 for the future summary slot and 1 for the history slots—which is projected through a learned linear layer. This explicitly instructs the residual blocks to treat the two modalities differently.
- **Padding Mask:** We use an additional binary mask representing valid versus zero-padded history steps, which is also passed through a learned embedding layer.
- **Static Information:** Similar to the decoder, static categorical or continuous covariates are mapped to the spatial attribute H_{seq} and expanded identically across the temporal axis.

Gaussian Output The sequence $\mathbf{H}_{\text{in}}$ and side information $\mathbf{S}$ are evaluated through the Context Producer described in Appendix H.2, returning a flattened representation vector in $\mathbb{R}^{C \times H_{\text{seq}}}$. Finally, this vector is processed by a Gaussian MLP head to produce the mean $\mu_t \in \mathbb{R}^M$ and diagonal log-variance $\log \Sigma_t \in \mathbb{R}^M$ defining the variational posterior for $\mathbf{z}_t$.

H.5. Transition Architecture (Baselines)

The Gaussian baseline transition model parameterizes $p_\psi^{(t)}(\mathbf{z}_t | \mathbf{z}_{t-j:t-1}, \mathbf{u}_{t-j:t})$ as a diagonal Gaussian, where the mean and log-variance are produced by a Context Producer followed by a Gaussian MLP head.

Input Preparation The transition model receives only the length-j latent history $\mathbf{z}_{t-j:t-1}$ as input (no future summary or observation information). Each latent vector $\mathbf{z}_{t-i} \in \mathbb{R}^M$ is linearly projected to the Context Producer's spatial dimension H_{seq}, yielding the primary input tensor $\mathbf{H}_{\text{in}} \in \mathbb{R}^{H_{\text{seq}} \times j}$. Unlike the encoder, no padding is required: the transition model is only invoked for $t > j$, so a full history of length j is always available.

Conditioning Construction Because the transition model has no future summary and no padded positions, its side information is considerably simpler than the encoder's:

- **Temporal Information:** Absolute time embeddings for the j history steps $t-j, \ldots, t-1$ are gathered and, if dynamic covariates $\mathbf{u}_{t-j:t}$ are present, concatenated along the feature dimension.
- **Mask Information:** No mask embeddings are used. The Context Producer receives a zero-dimensional mask tensor (i.e. `mask_tot_dim` $= 0$), so no role mask or padding mask is injected.
- **Static Information:** No static covariates are provided, since the transition operates entirely in the latent space and does not need to align with observation-level features.

Gaussian Output The Context Producer condenses the history into a flattened context vector in $\mathbb{R}^{C \times H_{\text{seq}}}$, which is passed to a Gaussian MLP head producing the mean $\mu_\psi(\mathbf{z}_{t-j:t-1}) \in \mathbb{R}^M$ and diagonal log-variance $\log \Sigma_\psi(\mathbf{z}_{t-j:t-1}) \in \mathbb{R}^M$. The transition density is then

$$p_\psi^{(t)}(\mathbf{z}_t | \mathbf{z}_{t-j:t-1}, \mathbf{u}_{t-j:t}) = \mathcal{N}(\mathbf{z}_t;\ \mu_\psi(\mathbf{z}_{t-j:t-1}),\ \operatorname{diag}(\exp \log \Sigma_\psi(\mathbf{z}_{t-j:t-1}))) \,.$$

H.6. VHP Architecture

The variational hierarchical prior (VHP) for the initial latent states $p_\eta(\mathbf{z}_{1:j} | \mathbf{z}_{-j+1:0})$ is implemented with two small Context Producers, one for the conditional prior p_η and one for the auxiliary variational posterior $q_\Phi(\mathbf{z}_{-j+1:0} | \mathbf{z}_{1:j})$. The architectures described below we implement with fewer layers and smaller hidden dimensions, given the simpler task of modeling only j time steps rather than the full sequence of length T. We refer to Appendix G for the mathematical derivation.

Conditional Prior $p_\eta(\mathbf{z}_t | \mathbf{z}_{t-j:t-1})$ The conditional prior for $t \leq j$ shares the same architecture as the Gaussian baseline transition described above: a Context Producer of temporal length j followed by a Gaussian MLP head producing $\mu_\eta, \log \Sigma_\eta \in \mathbb{R}^M$. Its side information consists of absolute time embeddings, optional dynamic covariates, and an embedded padding mask, since during the autoregressive generation of $\mathbf{z}_1, \ldots, \mathbf{z}_j$, the conditioning history may include auxiliary latents $\mathbf{z}_{-j+1:0}$ sampled from the VHP prior $p(\mathbf{z}_{-j+1:0}) = \mathcal{N}(\mathbf{0}, \mathbf{I})$ or from the auxiliary posterior q_Φ. Concretely, at step $t \in \{1, \ldots, j\}$, the history $\mathbf{z}_{t-j:t-1}$ is formed by concatenating the available auxiliary latents with any already-generated initial latents, left-padded with zeros when $t < j$. A binary padding mask indicating which positions are real versus padded is embedded through a learned linear layer and injected as mask side information, exactly as in the encoder (Appendix H.4).

Auxiliary Posterior $q_\Phi(\mathbf{z}_{-j+1:0} | \mathbf{z}_{1:j})$ The auxiliary posterior takes the encoder's initial latent samples $\mathbf{z}_{1:j}$ and produces a diagonal Gaussian over all j auxiliary latents jointly. Each latent $\mathbf{z}_t$ for $t \in \{1, \ldots, j\}$ is linearly projected from $\mathbb{R}^M$ to $\mathbb{R}^{H_{\text{seq}}}$, forming an input sequence of shape (H_{seq}, j) for a second, smaller Context Producer. This Context Producer uses the same residual-block architecture described in Appendix H.2, with temporal side information (absolute time embeddings and optional covariates for the first j steps) but no mask embeddings (`mask_tot_dim` $= 0$), since the full sequence $\mathbf{z}_{1:j}$ is always observed. The flattened output in $\mathbb{R}^{C \times H_{\text{seq}}}$ is passed to a Gaussian MLP head that outputs the mean $\mu_\Phi \in \mathbb{R}^{M \times j}$ and diagonal log-variance $\log \Sigma_\Phi \in \mathbb{R}^{M \times j}$ for the j auxiliary latents. Sampling is performed via the reparameterization trick:

$$\mathbf{z}_{-j+1:0} = \mu_\Phi(\mathbf{z}_{1:j}) + \exp\big(\tfrac{1}{2} \log \Sigma_\Phi(\mathbf{z}_{1:j})\big) \odot \boldsymbol{\epsilon}, \qquad \boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I}).$$

I. Implementation Details

I.1. Parameters

For clarity, we summarize the parameters being optimized in our model. The full parameter set consists of five groups:

1. **Encoder parameters** ϕ: all parameters of the encoder Context Producer, Future Summary module, and Gaussian MLP head (Appendix H.4).
2. **Decoder parameters** θ: all parameters of the decoder network (Appendix H.3).
3. **Transition parameters** ψ: all parameters of the transition Context Producer and Gaussian MLP head (Appendix H.5).
4. **VHP conditional prior parameters** η: all parameters of the conditional prior Context Producer and Gaussian MLP head used to model $p_\eta(\mathbf{z}_t|\mathbf{z}_{t-j:t-1})$ for $t \in \{1, \dots, j\}$ (Appendix H.6).
5. **VHP auxiliary posterior parameters** Φ: all parameters of the auxiliary posterior Context Producer and Gaussian MLP head used to model $q_\Phi(\mathbf{z}_{-j+1:0}|\mathbf{z}_{1:j})$ (Appendix H.6).

I.2. Optimization Techniques

It is a well-known difficulty of VAEs that the optimization of the ELBO may be viewed as a multi-objective optimization problem, where the two competing objectives are the reconstruction of the data and the regularization of the latent space via the prior Higgins et al. (2017); Rezende and Viola (2018). For example, in the most-general VAE setting, the ELBO may be written as

$$\mathrm{ELBO}(\phi, \theta) = -\mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}[\log p_\theta^{(t)}(\mathbf{x}|\mathbf{z})] + D_{\mathrm{KL}}(q_\phi(\mathbf{z}|\mathbf{x}) \,\|\, p(\mathbf{z}))\,,$$

or in the distortion-rate form,

$$\mathrm{ELBO}(\phi, \theta) = \mathcal{D}(\phi, \theta) + \mathcal{R}(\phi),$$

where the distortion term $\mathcal{D}(\phi, \theta) := -\mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}[\log p_\theta(\mathbf{x}|\mathbf{z})]$ measures the reconstruction error, and the rate term $\mathcal{R}(\phi) := D_{\mathrm{KL}}(q_\phi(\mathbf{z}|\mathbf{x}) \,\|\, p(\mathbf{z}))$ measures the complexity of the latent representation.

It is commonplace in the VAE literature and the Deep SSM literature to employ a weighting term β on the rate term to balance the tradeoff between reconstruction and regularization Higgins et al. (2017); Krishnan et al. (2017), this is particularly important for DSSMs with learnable priors. To emphasize learning a good reconstruction before regularization, it is common to anneal β from 0 to 1 during training. To avoid confusing notation with the diffusion schedule, we denote this regularization term as λ instead of β.

That is, we optimize the following objective:

$$\mathcal{L}_{\mathrm{ELBO}}(\phi, \eta, \theta, \psi, \Phi; \mathbf{x}_{1:T}, \mathbf{u}_{1:T}) = \mathcal{D}(\phi, \theta) + \lambda \mathcal{R}(\phi, \eta, \psi, \Phi),$$

where the distortion term $\mathcal{D}$ includes the reconstruction loss, and the rate term $\mathcal{R}$ collects the posterior entropy, diffusion KL, any initialization likelihoods, and auxiliary posterior KL contributions.

We utilize cosine annealing schedules for λ in our experiments. To find the optimal schedule, we use the hyperparameter λ_{end} to specify the ending value of λ, and use the hyperparameter $\lambda_{\mathrm{warmup}}$ to specify the number of training iterations until λ reaches λ_{end}. Empirically, we find that choosing a good schedule is crucial for performance, hence we include these parameters in our hyperparameter sweep during training (see Section F.1.1).

I.3. Computing MC Estimates of ELBO Terms

It still remains to detail the computation of the other ELBO terms, and to deal with the expectations in each term.

When any of the distributions involved are Gaussian, computing the KL divergences becomes straightforward as the Gaussian distribution admits closed-form expressions for likelihoods and KL divergences. In the case of non-Gaussian distributions, we must use a Monte Carlo estimates to compute the various expectations. To allow us to reduce the error of the Monte Carlo estimates, we draw several latent trajectory samples from our encoder during training. Let the S trajectories drawn from the encoder during training be denoted by $\mathbf{z}_{1:T}^{(s)} \sim q_\phi(\mathbf{z}_{1:T} \mid \mathbf{x}_{1:T}, \mathbf{u}_{1:T})$. We now detail the Monte-Carlo estimate for each term in the ELBO.

I.3.1. Initialization Loss

Following the decomposition in Eq. (44), we compute the initialization loss as the sum of the entropy term and the VHP loss. Using the sampled latent trajectories $\mathbf{z}_{1:T}^{(s)}$, we first compute the entropy estimate:

$$\mathcal{L}_{\text{init}}^{\text{entropy, MC}}(\phi; \mathbf{x}_{1:T}, \mathbf{u}_{1:T}) = \frac{1}{S} \sum_{s=1}^{S} \log q_\phi(\mathbf{z}_{1:j}^{(s)} | \mathbf{x}_{1:T}, \mathbf{u}_{1:T}). \tag{46}$$

For the VHP loss, we sample auxiliary variables $\mathbf{z}_{-j+1:0}^{(s)} \sim q_\phi(\mathbf{z}_{-j+1:0} | \mathbf{z}_{1:j}^{(s)})$ for each trajectory s. The KL divergence term is computed analytically, assuming Gaussian distributions.

$$\begin{aligned}\mathcal{L}_{\text{VHP}}^{\text{MC}}(\phi, \eta, \Phi; \mathbf{x}_{1:T}, \mathbf{u}_{1:T}) = \frac{1}{S} \sum_{s=1}^{S} \Bigg[&- \sum_{t=1}^{j} \log p_\eta(\mathbf{z}_t^{(s)} | \mathbf{z}_{t-j:t-1}^{(s)}) \\ &+ D_{\text{KL}}\Big(q_\phi(\mathbf{z}_{-j+1:0} | \mathbf{z}_{1:j}^{(s)}) \,\Big\|\, p(\mathbf{z}_{-j+1:0}) \Big) \Bigg],\end{aligned} \tag{47}$$

where the history $\mathbf{z}_{t-j:t-1}^{(s)}$ in the reconstruction term draws from the auxiliary variables $\mathbf{z}_{-j+1:0}^{(s)}$ when indices are non-positive. The total initialization loss is

$$\mathcal{L}_{\text{init}}^{\text{MC}} = \mathcal{L}_{\text{init}}^{\text{entropy, MC}} + \mathcal{L}_{\text{VHP}}^{\text{MC}}. \tag{48}$$

I.3.2. Sequence Reconstruction Loss

For the reconstruction loss, we use the same sampled latent trajectories $\mathbf{z}_{1:T}^{(s)} \sim q_\phi(\mathbf{z}_{1:T} \mid \mathbf{x}_{1:T}, \mathbf{u}_{1:T})$ as above.

Our decoder is Gaussian, and given most generally by

$$\begin{aligned}&p_\theta^{(t)}(\mathbf{x}_t \mid \mathbf{z}_{\max(1,t-j+1):t}, \mathbf{u}_{\max(1,t-j+1):t}) = \\ &\qquad \mathcal{N}(\mu_\theta(\mathbf{z}_{\max(1,t-j+1):t}, \mathbf{u}_{\max(1,t-j+1):t}), \Sigma_\theta(\mathbf{z}_{\max(1,t-j+1):t}, \mathbf{u}_{\max(1,t-j+1):t})),\end{aligned}$$

so the log-likelihood is available in closed form. For $t \leq j$, the conditioning window $\mathbf{z}_{\max(1,t-j+1):t}$ contains fewer than j entries; in practice, the decoder left-pads the history with zeros to maintain a fixed input size, consistent with the architecture described in Appendix H.3.

For trajectory sample $\mathbf{z}_{1:T}^{(s)}$, we denote the decoder mean and covariance as

$$\begin{aligned}\mu_\theta^{(s)} &:= \mu_\theta(\mathbf{z}_{\max(1,t-j+1):t}^{(s)}, \mathbf{u}_{\max(1,t-j+1):t}), \\ \Sigma_\theta^{(s)} &:= \Sigma_\theta(\mathbf{z}_{\max(1,t-j+1):t}^{(s)}, \mathbf{u}_{\max(1,t-j+1):t}).\end{aligned}$$

Using that

$$-\log \mathcal{N}(\mathbf{x}_t; \mu_\theta, \Sigma_\theta) = \frac{1}{2}\Big[(\mathbf{x}_t - \mu_\theta)^\top \Sigma_\theta^{-1}(\mathbf{x}_t - \mu_\theta) + \log\det(2\pi\Sigma_\theta)\Big],$$

if we have diagonal covariance Σ_θ, this reduces to a weighted squared error plus a log variance term and then

$$-\log \mathcal{N}\big(\mathbf{x}_t;\, \mu_\theta, \Sigma_\theta\big) = \frac{1}{2}\sum_{i=1}^{D}\left[\frac{(x_{t,i} - \mu_{\theta,i})^2}{\sigma_{\theta,i}^2} + \log(2\pi\sigma_{\theta,i}^2)\right],$$

Thus we have

$$\begin{aligned}
\mathcal{L}_{\text{recon}}(\phi, \theta; \mathbf{x}_{1:T}, \mathbf{u}_{1:T}) &= \sum_{t=1}^{T} \mathbb{E}_{q_\phi(\mathbf{z}_{1:T}|\mathbf{x}_{1:T}, \mathbf{u}_{1:T})}\Big[-\log p_\theta^{(t)}\big(\mathbf{x}_t \mid \mathbf{z}_{\max(1,t-j+1):t}, \mathbf{u}_{\max(1,t-j+1):t}\big)\Big] \\
&\approx \frac{1}{S}\sum_{s=1}^{S}\sum_{t=1}^{T}\Big[-\log \mathcal{N}\big(\mathbf{x}_t;\, \mu_\theta^{(s)}, \Sigma_\theta^{(s)}\big)\Big] \\
&= \frac{1}{2S}\sum_{s=1}^{S}\sum_{t=1}^{T}\sum_{i=1}^{D}\left[\frac{(x_{t,i} - \mu_{\theta,i}^{(s)})^2}{\sigma_{\theta,i}^{(s)2}} + \log(2\pi\sigma_{\theta,i}^{(s)2})\right] \\
&:= \mathcal{L}_{\text{recon}}^{\text{MC}}(\phi, \theta; \mathbf{x}_{1:T}, \mathbf{u}_{1:T}).
\end{aligned} \tag{49}$$

I.3.3. Posterior Entropy

Next, we compute the Monte Carlo estimate of the posterior entropy term.

$$\begin{aligned}
\mathcal{L}_{\text{entropy}}(\phi; \mathbf{x}_{1:T}, \mathbf{u}_{1:T}) &= \sum_{t=j+1}^{T} \mathbb{E}_{q_\phi(\mathbf{z}_{1:T}|\mathbf{x}_{1:T}, \mathbf{u}_{1:T})}\Big[\log q_\phi^{(t)}\big(\mathbf{z}_t \mid \mathbf{x}_{t:T}, \mathbf{z}_{t-j:t-1}, \mathbf{u}_{t-j:t}\big)\Big] \\
&\approx \frac{1}{S}\sum_{s=1}^{S}\sum_{t=j+1}^{T}\Big[\log q_\phi^{(t)}\big(\mathbf{z}_t^{(s)} \mid \mathbf{x}_{t:T}, \mathbf{z}_{t-j:t-1}^{(s)}, \mathbf{u}_{t-j:t}\big)\Big] \\
&:= \mathcal{L}_{\text{entropy}}^{\text{MC}}(\phi; \mathbf{x}_{1:T}, \mathbf{u}_{1:T}).
\end{aligned} \tag{50}$$

Again, for a Gaussian inference model, this term may be computed in closed form.

I.3.4. Diffusion Loss

The diffusion loss is given by Eq. (28), and may be computed using the sampled latent trajectories $\mathbf{z}_{1:T}^{(s)} \sim q_\phi(\mathbf{z}_{1:T} \mid \mathbf{x}_{1:T}, \mathbf{u}_{1:T})$ as above. We replace the expectation over the variational posterior with

a Monte Carlo estimate,

$$\begin{aligned}\mathcal{L}_{\widehat{\text{diff}}}(\phi,\psi;\mathbf{x}_{1:T},\mathbf{u}_{1:T}) &= \sum_{t=j+1}^{T} \mathbb{E}_{q_\phi(\mathbf{z}_{1:T}|\mathbf{x}_{1:T},\mathbf{u}_{1:T})}\Big[D_{\text{KL}}\big(q(\mathbf{z}_t^K|\mathbf{z}_t^0)\,\|\,p(\mathbf{z}_t^K)\big) \\ &\quad + \mathbb{E}_{\epsilon\sim\mathcal{N}(0,I),k\sim\text{Unif}\{1,\dots,K\}}\left[K\tilde{w}_k||F_\psi(\mathbf{z}_t^k,\mathbf{c}_t^{(s)},k)-\boldsymbol{y}(\mathbf{z}_t^0,k)||_2^2\right]\Big] \\ &\approx \frac{1}{S}\sum_{s=1}^{S}\sum_{t=j+1}^{T}\Big[D_{\text{KL}}\Big(q(\mathbf{z}_t^K|\mathbf{z}_t^{0(s)})\,\Big\|\,p(\mathbf{z}_t^K)\Big) \\ &\quad + \mathbb{E}_{\epsilon\sim\mathcal{N}(0,I),k\sim\text{Unif}\{1,\dots,K\}}\left[K\tilde{w}_k||F_\psi(\mathbf{z}_t^{k(s)},\mathbf{c}_t^{(s)},k)-\boldsymbol{y}(\mathbf{z}_t^{0(s)},k)||_2^2\right]\Big],\end{aligned}$$

where $\mathbf{c}_t^{(s)}$ denotes the Markov history constructed from the sampled latent trajectory $\mathbf{z}_{1:T}^{(s)}$. Note that for each of the sampled latent trajectories $\mathbf{z}_{1:T}^{(s)}$, we must run the forward diffusion process to obtain the noisy latents $\mathbf{z}_t^{k(s)}$ for $k=1,\dots,K$. That is, for each time step $t=j+1,\dots,T$ and sample $s=1,\dots,S$, we draw unique noise samples and diffusion steps and compute the corresponding noisy latents. To increase the amount of training signal to the diffusion model, we may draw multiple diffusion steps or noise samples per latent trajectory sample. During typical diffusion model training, the clean data is known, so over the course of training the diffusion model sees many noise samples per data point. Here, since the clean latent is itself sampled from a variational posterior, the diffusion model only sees one clean latent per data point per training iteration. To compensate, we may draw S_k diffusion steps per latent trajectory sample $\mathbf{z}_{1:T}^{(s)}$ (with corresponding noise samples), increasing the training signal to the diffusion model per iteration. Additionally, we note that the first term in the diffusion loss is computed in closed form for Gaussian distributions.

The resulting Monte Carlo estimate of the diffusion loss is finally

$$\begin{aligned}\mathcal{L}^{\text{MC}}_{\widehat{\text{diff}}}(\phi,\psi;\mathbf{x}_{1:T},\mathbf{u}_{1:T}) &= \frac{1}{S}\sum_{s=1}^{S}\sum_{t=j+1}^{T}\Big[D_{\text{KL}}\Big(q(\mathbf{z}_t^K|\mathbf{z}_t^{0(s)})\,\Big\|\,p(\mathbf{z}_t^K)\Big) \\ &\quad + \frac{1}{S_k}\sum_{m=1}^{S_k}\mathbb{E}_{\epsilon\sim\mathcal{N}(0,I),k\sim\text{Unif}\{1,\dots,K\}}\left[K\tilde{w}_k||F_\psi(\mathbf{z}_t^{k(s,m)},\mathbf{c}_t^{(s)},k)-\boldsymbol{y}(\mathbf{z}_t^{0(s)},k)||_2^2\right]\Big],\end{aligned} \tag{51}$$

where $\mathbf{z}_t^{k(s,m)}$ denotes the noisy latent at diffusion step k for time step t, sample s, and diffusion sample m.

I.3.5. Full ELBO Objective

Combining each of the above terms, we have the full ELBO objective

$$\begin{aligned}\mathcal{L}^{\text{MC}}_{\text{ELBO}}(\phi,\eta,\theta,\psi,\Phi;\mathbf{x}_{1:T},\mathbf{u}_{1:T}) &= \mathcal{L}^{\text{MC}}_{\text{init}}(\phi,\eta,\Phi;\mathbf{x}_{1:T},\mathbf{u}_{1:T}) + \mathcal{L}^{\text{MC}}_{\text{recon}}(\phi,\theta;\mathbf{x}_{1:T},\mathbf{u}_{1:T}) \\ &\quad + \mathcal{L}^{\text{MC}}_{\widehat{\text{diff}}}(\phi,\psi;\mathbf{x}_{1:T},\mathbf{u}_{1:T}) + \mathcal{L}^{\text{MC}}_{\text{entropy}}(\phi;\mathbf{x}_{1:T},\mathbf{u}_{1:T}).\end{aligned} \tag{52}$$

I.4. Training Algorithm

We consider the case of training with multiple i.i.d. time series of fixed length T. In this case we compute the ELBO over the full time series for each sequence in the minibatch, and average the ELBOs to get the minibatch ELBO. We describe the training procedure in Algorithm 1, and the sampling of latent trajectories in Algorithm 2.

Algorithm 1: Training for Diffusion State-Space Models

Input: Minibatch $\{(\mathbf{x}_{1:T}^{(b)}, \mathbf{u}_{1:T}^{(b)})\}_{b=1}^{B}$; untrained encoder q_ϕ (including F_ϕ); initial prior p_η; auxiliary posterior q_Φ; decoder $p_\theta^{(t)}$; diffusion model $p_\psi^{(t)}$; number of latent paths S; number of diffusion samples per path S_k; Markov order j

Output: Updated parameters $(\theta, \phi, \psi, \eta, \Phi)$

for $b \leftarrow 1$ **to** B **do**

  $\{\mathbf{z}_{1:T}^{(s,b)}\}_{s=1}^{S}, \{\mathbf{z}_{-j+1:0}^{(s,b)}\}_{s=1}^{S} \leftarrow \texttt{SampleLatentPaths}(\mathbf{x}_{1:T}^{(b)}, \mathbf{u}_{1:T}^{(b)}, F_\phi, q_\phi, q_\Phi, S, j)$

  Compute initialization loss $\mathcal{L}_{\text{init}}^{\text{MC},(b)}$ using Eq. (48) with $\{\mathbf{z}_{1:j}^{(s,b)}\}_{s=1}^{S}$ and $\{\mathbf{z}_{-j+1:0}^{(s,b)}\}_{s=1}^{S}$

  Compute reconstruction loss $\mathcal{L}_{\text{recon}}^{\text{MC},(b)}$ using Eq. (49) with $\{\mathbf{z}_{1:T}^{(s,b)}\}_{s=1}^{S}$

  Compute posterior entropy term $\mathcal{L}_{\text{entropy}}^{\text{MC},(b)}$ using Eq. (50) with $\{\mathbf{z}_{1:T}^{(s,b)}\}_{s=1}^{S}$

  Compute diffusion loss $\mathcal{L}_{\widehat{\text{diff}}}^{\text{MC},(b)}$ using Eq. (51) with $\{\mathbf{z}_{1:T}^{(s,b)}\}_{s=1}^{S}$ and S_k diffusion samples per path

  Compute λ from annealing schedule

  $\mathcal{L}_{\text{ELBO}}^{\text{MC},(b)} \leftarrow \mathcal{L}_{\text{recon}}^{\text{MC},(b)} + \lambda \left(\mathcal{L}_{\text{init}}^{\text{MC},(b)} + \mathcal{L}_{\widehat{\text{diff}}}^{\text{MC},(b)} + \mathcal{L}_{\text{entropy}}^{\text{MC},(b)} \right)$

end

$\mathcal{L}_{\text{ELBO}}^{\text{MC}} \leftarrow \frac{1}{B} \sum_{b=1}^{B} \mathcal{L}_{\text{ELBO}}^{\text{MC},(b)}$

Update $(\theta, \phi, \psi, \eta, \Phi)$ with gradient descent on $\mathcal{L}_{\text{ELBO}}^{\text{MC}}$

Algorithm 2: `SampleLatentPaths`: Sampling of S latent trajectories

Input: Observations $\mathbf{x}_{1:T}$, covariates $\mathbf{u}_{1:T}$, future summary network F_ϕ, encoder q_ϕ, VHP auxiliary posterior q_Φ, number of paths S, Markov order j

Output: Latent trajectories $\{\mathbf{z}_{1:T}^{(s)}\}_{s=1}^{S}$ and auxiliary latents $\{\mathbf{z}_{-j+1:0}^{(s)}\}_{s=1}^{S}$

Compute future summaries $(\mathbf{h}_T, \ldots, \mathbf{h}_1) \leftarrow F_\phi(\mathbf{x}_{1:T}, \mathbf{u}_{1:T})$

for $s \leftarrow 1$ **to** S **do**

  `// Step 1: Sample initial` j `latents`

  for $t \leftarrow 1$ **to** j **do**

    $k \leftarrow t - 1$ `// available history length`

    $m \leftarrow j - k$ `// missing history slots`

    Left-pad with zeros: $\mathbf{z}_{t-j:t-k-1}^{(s)} \leftarrow \mathbf{0}$

    Form length-j history $\mathbf{z}_{t-j:t-1}^{(s)} \leftarrow (\mathbf{z}_{t-j:t-k-1}^{(s)}, \mathbf{z}_{t-k:t-1}^{(s)})$

    Sample $\mathbf{z}_t^{(s)} \sim q_\phi^{(t)}(\mathbf{z}_t \mid \mathbf{z}_{t-j:t-1}^{(s)}, \mathbf{h}_t)$

    Append $\mathbf{z}_t^{(s)}$ to path s

  end

  `// Step 2: Sample auxiliary latents via VHP`

  Sample $\mathbf{z}_{-j+1:0}^{(s)} \sim q_\Phi(\mathbf{z}_{-j+1:0} \mid \mathbf{z}_{1:j}^{(s)})$

  `// Step 3: Sample remaining latents autoregressively`

  for $t \leftarrow j + 1$ **to** T **do**

    $\mathbf{z}_{t-j:t-1}^{(s)} \leftarrow$ last j latents from path s

    Sample $\mathbf{z}_t^{(s)} \sim q_\phi^{(t)}(\mathbf{z}_t \mid \mathbf{z}_{t-j:t-1}^{(s)}, \mathbf{h}_t)$

    Append $\mathbf{z}_t^{(s)}$ to path s

  end

end

return $\{\mathbf{z}_{1:T}^{(s)}\}_{s=1}^{S}, \{\mathbf{z}_{-j+1:0}^{(s)}\}_{s=1}^{S}$

We briefly describe the computational cost incurred by the diffusion model during training and sampling. Under Algorithm 1, we point out to the reader that the diffusion model receives an effective batch size of $B_{\text{eff}} = B \times (T - j) \times S \times S_k$, where B is the number of sequences in a batch, and $T - j$ is the number of diffusion-parameterized transitions in each sequence. Thus for training only a single evaluation of the diffusion model is required, assuming there is sufficient memory to handle the $T - j$ factor in the batch. As T scales, this factor increases necessitating an unavoidable cost in memory (unless gradient checkpointing (Chen et al., 2016) is used to trade off time for memory). However, the larger batch size provides a stronger per-batch training signal to the diffusion model during training, amortizing the cost of training the diffusion model across more transitions. The per-function-evaluation of the diffusion model depends on the size of the input tensor, which is $B_{\text{eff}} \times M \times j + 1$. Therefore given enough memory, the diffusion-incurred time complexity for a forward pass of Algorithm 1 is $O(g(M \times j + 1))$ where g is the time complexity of a pass through the diffusion model. It is only during sampling that we lose parallelism across time steps, in which case the total diffusion cost becomes $O(K \times T \times g(M \times j + 1))$, for K diffusion steps.

J. Extended Related Works

Deep state space models. Similarly to other Deep State Space Models (DSSMs) (Girin et al., 2022), our Diffusion-Driven State Space Model (DDSSM) parametrizes the transitions and emissions densities of a state space model with a neural network. However, our model replaces the parametric (typically Gaussian) transition distribution of a DSSM with a diffusion model. In particular, we extend the Deep Kalman Filter (DKF) of Krishnan et al. (2015, 2017) to have diffusion transitions. An immediate shortcoming of the usage of Gaussian transitions is the inability to model multi-modal transitions, which we verify in our simulation study in Section 4. Karl et al. (2016) observed the shortcoming of Gaussian transitions in the context of modeling physical systems, arguing that the regularization provided by Gaussian transitions harms reconstruction performance.

Several lines of work have attempted to make the DKF more expressive. Karl et al. (2016) proposed learning a more flexible transition by learning a deterministic transition of a stochastic variable. That is, they learn $\mathbf{z}_t = f_\psi(\mathbf{z}_{t-1}, \mathbf{u}_t, \beta_t)$ where f_ψ is a neural network and β_t is a stochastic parameter of the transition sampled from a simple distribution such as a Gaussian. This ensures that the latent states are sampled from the transition distribution, forcing the encoder to receive gradients directly from the transition model.

A separate track of work attempting to make the DKF more expressive has been to add auxiliary variables. For example the Kalman VAE (KVAE) (Fraccaro et al., 2017) introduces parameters $\mathbf{a}_{1:T}$ and propose the generative model $p(\mathbf{x}_{1:T}, \mathbf{a}_{1:T}, \mathbf{z}_{1:T}|\mathbf{u}_{1:T}) = p(\mathbf{x}_{1:T}|\mathbf{a}_{1:T})p(\mathbf{a}_{1:T}|\mathbf{z}_{1:T})p(\mathbf{z}_{1:T}|\mathbf{u}_{1:T})$, relying on linear Gaussian $p(\mathbf{a}_t|\mathbf{z}_t, \mathbf{u}_t)$ and $p(\mathbf{z}_t|\mathbf{z}_{t-1}, \mathbf{h}_t, \mathbf{u}_t)$ distributions. Klushyn et al. (2021) extends this model to have nonlinear Gaussian transitions, proposing the Extended Kalman VAE (EKVAE). The primary advantage to their method is that inference about the parameters $\mathbf{a}_t$ use the filtering/smoothing equations of the extended Kalman filter (Ribeiro, 2004), to allow the training objective for the transition model to be sample-based instead of likelihood-based. It may be possible to extend the EKVAE and other deep SSMs to use diffusion transitions as well, but we leave this to future work: we emphasize that our contribution is not on the design of the generative model, but rather on the usage of diffusion models as flexible transition distributions.

Lastly, we note that several works have extended deep SSMs using normalizing flows (NFs) (Papamakarios et al., 2021). While these admit exact-likelihoods, they are heavily constrained by the requirement of invertible neural network architectures, limiting applicability across many domains. One recent example is from Chen and Li (2025), who propose learning a differentiable particle filter (Le et al., 2017), allowing the transition and emission densities to be parameterized by NFs. We note that diffusion models would be unsuitable for this approach, as the particle filtering algorithm requires many samples through the transition for each time step. Another example is from De Bezenac et al. (2020), who assume linear transition dynamics and parameterize the emission density with a NF. This enables efficient inference using Kalman-style filtering/smoothing, but

does not allow for flexible transition dynamics. We mention Ziegler and Rush (2019) as well, who consider a variety of NF-based approaches for modeling discrete sequences, but they do not consider the true posterior form of their proposed generative models.

Models for temporal data which apply diffusion directly in observation space. Several lines of work model temporal data by applying diffusion methods directly in observation space, rather than incorporating a latent dynamic process. The advantage to this approach is that it does not require learning a latent representation of the data, and does not require performing inference about the latent state. For example, Rasul et al. (2021) employs autoregressive forecasting using a diffusion model, by conditioning the diffusion model on a summary of all previous time steps of the series. To create a summary, an RNN is run over the history of the time series, providing a single conditioning variable as an input to their diffusion model. The RNN summary is similar in spirit to the latent state of our model; in our model the conditioning set $\mathbf{c}_t$ fully summarizes the history of the time series up to the time t. The advantage of our approach is that we are able to operate the diffusion model fully in latent-space, which is by construction well-suited to modeling the dynamics of the time series. Additionally, the latent space can be designed to be low-dimensional, which can make training and sampling more efficient. Another advantage is that our latent representation may provide a more interpretable representation of the dynamics of the time series, whereas it is unclear how to interpret the RNN summary of Rasul et al. (2021). Another well known work is by Tashiro et al. (2021), who perform imputation using diffusion models by learning to denoise imputation targets while conditioning on observed data. Their method operates on fixed length windows of a time series, and handles forecasting by setting imputation targets to be future time steps through a conditioning mask. By changing the mask, their method can sample many timesteps into the future by conditioning on a fixed length history. One drawback of this approach is that the model cannot utilize past data beyond the fixed length window which is determined at training time. This is contrary to Rasul et al. (2021) or our method which attempt to infer the necessary parameters to accurately sample the next time step. More importantly, Tashiro et al. (2021) requires their diffusion model to jointly model the full conditional distribution of the entire future given the entire history in a fixed window, which is expensive to learn and sample from. We note that we have designed our diffusion model architecture to match Tashiro et al. (2021) closely, as their previously state-of-the-art performance suggests that their architecture is well-suited to modeling temporal data. We detail the differences between our architecture and theirs in Appendix H.1.

Latent diffusion models for static data. While diffusion models are already computationally expensive in the static setting—motivating latent diffusion models (Rombach et al., 2022)—the challenge is amplified for time series, where the model must capture both high-dimensional structure and temporal dynamics. Our work can be considered a temporal extension of the diffusion models of Vahdat et al. (2021) and Shmakov et al. (2023), who train a diffusion model end-to-end in the latent space of a VAE. Like Vahdat et al. (2021), we design the training objective of the diffusion model to account for the fact the model is being trained concurrently with the VAE. However, we employ preconditioning (Karras et al., 2022) of the denoising model to reduce the variance of the gradients across noise levels. This is important since the gradients of the diffusion model are used to train the VAE. Details of our approach to concurrent training are described in Appendix E.

Latent diffusion models for temporal data. (WIP) Qian et al. (2024) propose a latent diffusion model framework for unconditional time-series generation, which they call TimeLDM. They employ a VAE training objective, where the encoder accepts an entire time series as input, and then produces an entire latent time series as output, i.e $\mathbf{z}_{1:T} = \text{Encoder}(\mathbf{x}_{1:T})$. However, there is no mechanism in their model to capture any temporal structure within $\boldsymbol{z}_{1:T}$. The decoder is responsible for reconstructing the entire time series from the latent time series, and the diffusion model is trained to model the distribution of time-series examples in latent space. We note that their method cannot perform forecasting, and they train in a two-stage manner where the VAE is trained separately from the diffusion model. In comparison, our method does capture temporal structure in the latent space, as the latent state at time t is conditioned on the latent state at time $t-1$. Additionally, our model can perform any tasks available to a standard state-space model, including unconditional generation.

Suh et al. (2025) proposes a latent diffusion modeling framework to unify forecasting, imputation, unconditional generation, and conditional generation of time series. Their autoencoder similar to Qian et al. (2024) outputs an entire latent time series given an entire time series as input. The decoder however aims to reconstruct a target subset of observations given the latent time series, a conditioning set from the observations, and a mask to indicate which inputs are targets/conditional. The diffusion model is then trained to model the distribution of the latent time series, conditioned on the conditioning set and mask. This is to say that the diffusion model does not operate independently of the data, contrary to our method where the diffusion model lives fully in the latent space, conditioning only on latent features. The strongest similarity which Suh et al. (2025) has to our method is that their method constructs a full ELBO training objective to maximize the log-likelihood of the data, where the loss of the diffusion model is a component of the ELBO. Of interest is the observation that end-to-end training was detrimental to their model, instead they prefer first training the autoencoder and then training the diffusion model separately. This claim however is not validated within their paper, and it is unclear how they attempted to train end-to-end. As an artifact of training in stages, their variational posterior is regularized towards a standard Gaussian distribution. If training jointly, their objective instead would regularize the variational posterior towards the distribution of the diffusion model. Liu et al. (2024) and Feng et al. (2024) give other approaches to time-series diffusion modeling in the latent space of a pre-trained autoencoder. We note that Feng et al. (2024) applies regularization of the latent space during the training of the autoencoder to a standard Gaussian distribution to avoid learning an unregularized space.