5th Symposium on Advances in Approximate Bayesian Inference, 2023 1–17

Neural Adaptive Smoothing via Twisting

Michael Y. Li MICHAELYLI@STANFORD.EDU
Dieterich Lawson JDLAWSON@STANFORD.EDU
Scott Linderman SCOTT.LINDERMAN@STANFORD.EDU

Abstract

We propose NAS-X, a method for sequential latent variable model learning and inference that uses smoothing sequential Monte Carlo (SMC) in a reweighted wake sleep (RWS) framework. Our method works with both discrete and continuous latent variables, and successfully fits a wider range of models than filtering SMC-based methods. We evaluate NAS-X on several tasks and find that it substantially outperforms existing methods in both inference and parameter recovery.

1. Introduction and Background

This work considers model learning and inference in sequential latent variable models with Markovian structure, i.e. models that factor as

$$p_{\boldsymbol{\theta}}(\mathbf{x}_{1:T}, \mathbf{y}_{1:T}) = p_{\boldsymbol{\theta}}(\mathbf{x}_1) p_{\boldsymbol{\theta}}(\mathbf{y}_1 \mid \mathbf{x}_1) \prod_{t=2}^{T} p_{\boldsymbol{\theta}}(\mathbf{x}_t \mid \mathbf{x}_{t-1}) p_{\boldsymbol{\theta}}(\mathbf{y}_t \mid \mathbf{x}_t), \tag{1}$$

with latent variables $\mathbf{x}_{1:T} \in \mathcal{X}^T$, observations $\mathbf{y}_{1:T} \in \mathcal{Y}^T$, and global parameters $\boldsymbol{\theta} \in \Theta$. We are specifically interested in *nonlinear* latent variable models where the conditional distributions $p_{\boldsymbol{\theta}}(\mathbf{x}_t \mid \mathbf{x}_{t-1})$ and $p_{\boldsymbol{\theta}}(\mathbf{y}_t \mid \mathbf{x}_t)$ depend nonlinearly on $\mathbf{x}_{t-1}$ and $\mathbf{x}_t$ respectively. This is an important class of models that encompass sequential versions of variational autoencoders (Rezende et al., 2014; Kingma and Welling, 2014; Krishnan et al., 2015; Lawson et al., 2019), financial volatility models (Chib et al., 2009), and biophysical models of neural activity (Hodgkin and Huxley, 1952).

Estimating the marginal likelihood $p_\theta(\mathbf{y}_{1:T})$ and posterior $p_\theta(\mathbf{x}_{1:T} \mid \mathbf{y}_{1:T})$ for this model class is difficult because it requires computing an intractable integral over the latents,

$$p_{\boldsymbol{\theta}}(\mathbf{y}_{1:T}) = \int_{\mathcal{X}^T} p_{\boldsymbol{\theta}}(\mathbf{y}_{1:T}, \mathbf{x}_{1:T}) \, \mathrm{d}\mathbf{x}_{1:T},$$

To tackle this problem, we extend Reweighted Wake Sleep (RWS) proposed in (Bornschein and Bengio, 2014; Hinton et al., 1995) to the sequential latent variable setting using recent advances in smoothing sequential Monte Carlo (Lawson et al., 2022).

1.1. Reweighted Wake Sleep

Our goals is to fit a model p_θ by maximizing the marginal likelihood. RWS uses a two-step coordinate ascent method to accomplish this. First, RWS estimates the gradients of the log marginal likelihood using self-normalized importance sampling (SNIS) with a proposal distribution q_ϕ. Second, to improve the quality of those gradient estimates, RWS optimizes the proposal q_ϕ by minimizing the inclusive Kullback-Leibler (KL) divergence from the posterior to the proposal which RWS also estimates with SNIS. Importantly, the gradients for both steps are posterior expectations, which RWS estimates using an SNIS particle approximation to the true posterior.

Let $q_\phi(\mathbf{x}_{1:T} \mid \mathbf{y}_{1:T})$ be a proposal distribution. RWS approximates the posterior distribution $p_\theta(\mathbf{x}_{1:T} \mid \mathbf{y}_{1:T})$ with N weighted particles sampled from q_ϕ,

$$\hat{p}(\mathbf{x}_{1:T} \mid \mathbf{y}_{1:T}) = \sum_{i=1}^{N} \overline{w}^{(i)} \delta_{\mathbf{x}_{1:T}^{(i)}}(\mathbf{x}_{1:T}), \quad \mathbf{x}_{1:T}^{(i)} \sim q_\phi(\mathbf{x}_{1:T} \mid \mathbf{y}_{1:T}), \quad w^{(i)} = \frac{p_\theta(\mathbf{x}_{1:T}^{(i)}, \mathbf{y}_{1:T})}{q_\phi(\mathbf{x}_{1:T}^{(i)} \mid \mathbf{y}_{1:T})} \tag{2}$$

where $w^{(i)}$ are unnormalized weights formed from a ratio of likelihoods and $\overline{w}^{(i)}$ are their self-normalized counterparts, i.e. $\sum_{i=1}^{N} \overline{w}^{(i)} = 1$.

RWS fits the proposal distribution q_ϕ by descending the gradients of the inclusive KL divergence, estimated as

$$\nabla_\phi \mathrm{KL}(p_\theta \mid\mid q_\phi) = -\mathbb{E}_{p_\theta(\mathbf{x}_{1:T} \mid \mathbf{y}_{1:T})} \left[\nabla_\phi \log q_\phi(\mathbf{x}_{1:T} \mid \mathbf{y}_{1:T})\right] \approx -\sum_{i=1}^{N} \overline{w}^{(i)} \nabla_\phi \log q_\phi(\mathbf{x}_{1:T}^{(i)} \mid \mathbf{y}_{1:T}). \tag{3}$$

The equality on the right hand side is approximate in the sense that the sum of weighted gradients is a biased but consistent ($N \to \infty$) Monte Carlo estimate of the expectation, as per the consistency of self-normalized importance sampling (Owen, 2013).

The model p_θ is fit using the posterior approximation (2) to estimate of the gradients of the marginal likelihood, i.e.

$$\nabla_\theta \log p_\theta(\mathbf{y}_{1:T}) = \mathbb{E}_{p_\theta(\mathbf{x}_{1:T} \mid \mathbf{y}_{1:T})} \left[\nabla_\theta \log p_\theta(\mathbf{x}_{1:T}, \mathbf{y}_{1:T})\right] \approx \sum_{i=1}^{N} \overline{w}^{(i)} \nabla_\theta p_\theta(\mathbf{x}_{1:T}^{(i)}, \mathbf{y}_{1:T}). \tag{4}$$

For a derivation of these identities, see Appendix A.

1.2. Estimating Posterior Expectations with Sequential Monte Carlo

As we saw in Equations 3 and 4, key quantities in RWS can be expressed as expectations w.r.t the posterior. Standard RWS uses SNIS to approximate these expectations, but in sequence models the variance of SNIS can scale exponentially in the sequence length. In this section, we review sequential Monte Carlo (SMC) (Doucet and Johansen, 2011; Naesseth et al., 2019), an inference algorithm that can produce estimators of posterior expectations with linear or even sub-linear variance scaling.

SMC approximates the posterior $p_{\boldsymbol{\theta}}(\mathbf{x}_{1:T} \mid \mathbf{y}_{1:T})$ with a set of N weighted particles $\mathbf{x}_{1:T}^{1:N}$ constructed iteratively by sampling from a sequence of target distributions $\{\pi_t(\mathbf{x}_{1:t})\}_{t=1}^{T}$.

Intuitively, sampling from these intermediate distributions may be "easier" than directly sampling from the posterior. Since these intermediate targets are often only available up to an unknown normalizing constant Z_t, SMC uses the unnormalized targets $\{\gamma_t(\mathbf{x}_{1:t})\}_{t=1}^T$, where $\pi_t(\mathbf{x}_{1:t}) = \gamma_t(\mathbf{x}_{1:t})/Z_t$. Provided the target distributions satisfy mild technical conditions and $\gamma_T(\mathbf{x}_{1:T}) \propto p_{\boldsymbol{\theta}}(\mathbf{x}_{1:T}, \mathbf{y}_{1:T})$, SMC returns a set of weighted particles that approximate the posterior $p_{\boldsymbol{\theta}}(\mathbf{x}_{1:T} \mid \mathbf{y}_{1:T})$ (Doucet and Johansen, 2011; Naesseth et al., 2019). These weighted particles can then be used to compute biased but consistent estimates of expectations of test functions with respect to the posterior distribution, similar to SNIS.

SMC repeats three steps: first, a set of latents $\mathbf{x}_{1:t}^{1:N}$ are sampled from a proposal distribution $q_\phi(\mathbf{x}_{1:t} \mid \mathbf{y}_{1:T})$. Then, each particle is weighted using the unnormalized target γ_t to form an empirical approximation of the normalized target distribution π_t. Finally, new particles $\mathbf{x}_{1:t}^{1:N}$ are drawn from this approximation to the normalized target. This last step is known as resampling and is crucial for SMC's success. For a thorough description of SMC, see Doucet and Johansen (2011); Naesseth et al. (2019); Del Moral (2004).

1.3. Smoothing SMC via Twisting Functions learned through Density Ratio Estimation

The most common choice of targets are the *filtering* distributions $\pi_t(\mathbf{x}_{1:t}) = p_{\boldsymbol{\theta}}(\mathbf{x}_{1:t} \mid \mathbf{y}_{1:t})$, and when SMC is run with these distributions as targets the resulting algorithm is known as filtering SMC. Filtering SMC has been used to estimate posterior expectations within the RWS framework in Neural Adaptive Sequential Monte Carlo (NASMC) (Gu et al., 2015), but a major disadvantage of filtering SMC is that it ignores future observations $\mathbf{y}_{t+1:T}$. Ignoring future observations can lead to particle degeneracy and high-variance estimates of test function integrals, frustrating approaches that use filtering SMC for model learning and inference (Maddison et al., 2017; Whiteley and Lee, 2014; Briers et al., 2010).

We could avoid these issues by using the *smoothing* distributions as targets, $\pi_t(\mathbf{x}_{1:t}) = p_{\boldsymbol{\theta}}(\mathbf{x}_{1:t} \mid \mathbf{y}_{1:T})$, but unfortunately the smoothing distributions are not available from the model. We can approximate the smoothing distributions by observing that $p_{\boldsymbol{\theta}}(\mathbf{x}_{1:t}, \mathbf{y}_{1:T})$ is proportional to the filtering distributions, $p_{\boldsymbol{\theta}}(\mathbf{x}_{1:t}, \mathbf{y}_{1:t})$, times the *lookahead* distributions, $p_{\boldsymbol{\theta}}(\mathbf{y}_{t+1:T} \mid \mathbf{x}_t)$. If the lookahead distributions are well-approximated by a sequence of *twists* $\{r(\mathbf{y}_{t+1:T}, \mathbf{x}_t)\}_{t=1}^T$, then running SMC with targets $\gamma_t(\mathbf{x}_{1:t}) = p_{\boldsymbol{\theta}}(\mathbf{x}_{1:t}, \mathbf{y}_{1:t})\, r(\mathbf{y}_{t+1:T}, \mathbf{x}_t)$ approximates smoothing SMC (Whiteley and Lee, 2014).

Learning the twists can be extremely challenging (Maddison et al., 2017; Lawson et al., 2018; Guarniero et al., 2017). To tackle these challenges, SIXO (Lawson et al., 2022) leverages a denstity-ratio estimation approach Sugiyama et al. (2012). This method is motivated by the observation that the lookahead distribution is proportional to a ratio of densities,

$$p_{\boldsymbol{\theta}}(\mathbf{y}_{t+1:T} \mid \mathbf{x}_t) = \frac{p_{\boldsymbol{\theta}}(\mathbf{x}_t \mid \mathbf{y}_{t+1:T})\, p_{\boldsymbol{\theta}}(\mathbf{y}_{t+1:T})}{p_{\boldsymbol{\theta}}(\mathbf{x}_t)} \propto \frac{p_{\boldsymbol{\theta}}(\mathbf{x}_t \mid \mathbf{y}_{t+1:T})}{p_{\boldsymbol{\theta}}(\mathbf{x}_t)}. \tag{5}$$

Therefore, the logits of a classifier trained to distinguish between samples from these densities will approximate the twists. We summarize the process for training twists below.

$$\begin{gathered}\tilde{\mathbf{x}}_{1:T} \sim p_{\boldsymbol{\theta}}(\mathbf{x}_{1:T}), \qquad \mathbf{x}_{1:T}, \mathbf{y}_{1:T} \sim p_{\boldsymbol{\theta}}(\mathbf{x}_{1:T}, \mathbf{y}_{1:T}) \\ \mathcal{L}_{\mathrm{DRE}}(\boldsymbol{\psi}) = \frac{1}{T-1} \sum_{t=1}^{T-1} \log \sigma(\log r_{\boldsymbol{\psi}}(\mathbf{y}_{t+1:T}, \mathbf{x}_t)) + \log(1 - \sigma(\log r_{\boldsymbol{\psi}}(\mathbf{y}_{t+1:T}, \tilde{\mathbf{x}}_t)))\end{gathered} \tag{6}$$

Importantly, SIXO returns a sequence of particle approximations that approximate the smoothing distributions, a crucial fact we will leverage in NAS-X.

2. NAS-X: Neural Adaptive Smoothing via Twisting

NAS-X uses SIXO's sequence of particle approximations of the smoothing distributions,

$$\hat{p}(\mathbf{x}_{1:t} \mid \mathbf{y}_{1:T}) = \sum_{i=1}^{N} \overline{w}_t^{(i)} \delta_{\mathbf{x}_{1:t}^{(i)}}(\mathbf{x}_{1:t}), \quad t = 1, \ldots, T \tag{7}$$

to estimate the posterior expectations required by the RWS framework. Specifically, NAS-X computes the gradients of the inclusive KL divergence for learning the proposal q_ϕ as

$$-\mathbb{E}_{p(\mathbf{x}_{1:T}|\mathbf{y}_{1:T})}\left[\nabla_\phi \log q_\phi(\mathbf{x}_{1:T} \mid \mathbf{y}_{1:T})\right] \approx -\sum_{t=1}^{T}\sum_{i=1}^{N} \overline{w}_t^{(i)} \nabla_\phi \log q_\phi(\mathbf{x}_t^{(i)} \mid \mathbf{x}_{t-1}^{(i)}, \mathbf{y}_{t:T}) \tag{8}$$

and computes the gradients of the model as

$$\mathbb{E}_{p_\theta(\mathbf{x}_{1:T}|\mathbf{y}_{1:T})}\left[\nabla_\theta \log p_\theta(\mathbf{x}_{1:T}, \mathbf{y}_{1:T})\right] \approx \sum_{t=1}^{T}\sum_{i=1}^{N} \overline{w}_t^{(i)} \nabla_\theta \log p_\theta(\mathbf{x}_t^{(i)}, \mathbf{y}_t \mid \mathbf{x}_{t-1}^{(i)}). \tag{9}$$

The particle weights $\overline{w}_t^{(i)}$ are directly available from running SMC with the twists. The twists are learned using density ratio estimation as described in the previous section. The full procedure is summarized in Algorithm 1.

3. Experiments

3.1. Linear Gaussian State Space Model

We first consider a one-dimensional linear Gaussian state space model with joint distribution

$$p(\mathbf{x}_{1:T}, \mathbf{y}_{1:T}) = \mathcal{N}(\mathbf{x}_1; 0, \sigma_x^2) \prod_{t=2}^{T} \mathcal{N}(\mathbf{x}_{t+1}; \mathbf{x}_t, \sigma_x^2) \prod_{t=1}^{T} \mathcal{N}(\mathbf{y}_t; \mathbf{x}_t, \sigma_y^2). \tag{10}$$

We compare NAS-X and NASMC by evaluating log marginal likelihood estimates and parameter recovery. For both methods we use a mean-field Gaussian proposal factored over time, $q(\mathbf{x}_{1:T}) = \prod_{t=1}^{T} q_t(\mathbf{x}_t) = \prod_{t=1}^{T} \mathcal{N}(\mathbf{x}_t; \mu_t, \sigma_t^2)$, with parameters $\mu_{1:T}$ and $\sigma_{1:T}^2$ corresponding to the means and variances at each timestep. We parameterize the twist as a

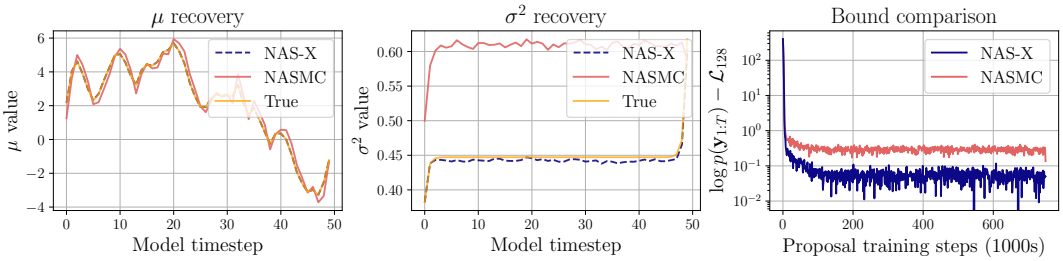

Figure 1: **Comparison of NAS-X vs NASMC on Inference in LG-SSM.** (**left**) Comparison of learned proposal means. (**middle**) Comparison of learned proposal variances. (**right**) Comparison of log-marginal likelihood bounds (lower is better). NAS-X's learned proposal captures the true posterior mean and variance and achieves tighter estimates of the log marginal likelihood.

quadratic function in $\mathbf{x}_t$ whose coefficients are functions of the observations and time step. We chose this form to match the functional form of the analytic log density ratio.

In the left and middle panels of Figure 1, we compare the posteriors learned by NAS-X and NASMC against the true posterior whose parameters can be obtained in closed form. NASMC exhibits a persistent bias in its parameter estimates and fails to capture the true posterior distribution because it employs filtering approximations, but NAS-X recovers the true posterior. In the right panel of Figure 1, we compare NAS-X and NASMC's bound gap, the difference between the true log marginal likelihood and the estimated log marginal likelihood. NAS-X's bound gap is lower, indicating it performs more accurate inference than NASMC. For details, see Appendix B.

3.2. Switching Linear Dynamical Systems

To explore NAS-X's ability to handle discrete latent variables, we consider a switching linear dynamical system (SLDS) model (Fox et al., 2008; Linderman et al., 2017). Specifically, we adapt the SLDS example from Linderman et al. (2017) in which the latent dynamics trace ovals in a manner that resembles cars racing on a NASCAR track. There are two coupled sets of latent variables: a discrete state $\mathbf{z}_t$, with $K = 4$ possible values, and a two-dimensional continuous state $\mathbf{x}_t$ that follows linear dynamics that depend on the discrete state. The observations are a noisy projection of $\mathbf{x}_t$ into a ten-dimensional observation space. For model details see Appendix C and Linderman et al. (2017).

For the proposal we factorize q over both time and the continuous/discrete states. The continuous distributions are parameterized by Gaussians while Categorical distributions are used for the discrete latent variables. For details on the proposal and twist, see Appendix C.

In the top of Figure 2, we compare NAS-X and NASMC on inference in the SLDS model. We report (average) posterior parameter recovery for the continuous and discrete latent states across 5 random samples from the generative model. NAS-X systematically recovers better estimates of both the discrete and continuous latent states.

We also present results from joint model learning and inference in the bottom of Figure 2. We compare the learned dynamics for NAS-X, NASCAR, and a Laplace-EM algorithm tailored for recurrent state space models (Zoltowski et al., 2020). In each panel, we plot the vector field of the learned dynamics and the posterior means, with different colors

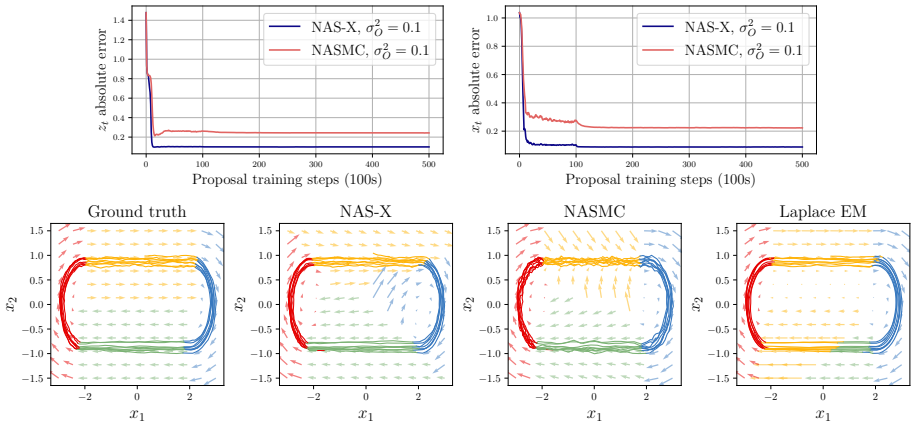

Figure 2: **Inference and model learning in switching linear dynamical systems (SLDS)**. (**top**) Posterior parameter recovery in the inference setting. (**bottom**) Comparison of learned dynamics and inferred latent states in model learning. Laplace EM sometimes learns incorrect segmentations, as seen here.

Table 1: Train $\mathcal{L}_{\text{BPF}}^{1024}$ for NAS-X and NASMC on the SLDS.

Method	$\sigma_O^2 = 0.001$	$\sigma_O^2 = 0.01$	$\sigma_O^2 = 0.1$
NAS-X	19.837 ± 0.0234	8.63 ± 0.0015	-2.79 ± 0.0009
NASMC	19.834 ± 0.0018	8.53 ± 0.001	-2.874 ± 0.0007
Laplace EM	19.154 ± 0.057	8.54 ± 0.0039	-2.765 ± 0.0012

corresponding to the four discrete states. In Table 1, we quantitatively compare the model learning performances across these three approaches by running a bootstrap proposal with the learned models and the true dynamics and observation variances. We normalize the bounds by the sequence length. NAS-X outperforms or performs on par with both NASMC and Laplace EM across the different observation noises σ_O^2. For details, see Appendix C.

3.3. Hodgkin Huxley model

In Appendix D we evaluate NAS-X on inference in the Hodgkin-Huxley model, a nonlinear dynamical system model of neural activity. In summary, NAS-X far outperforms FIVO, SIXO, and NASMC in terms of particle efficiency and inference quality.

4. Conclusion

In this work, we proposed NAS-X, a new method for approximate inference and model learning in sequential latent variable models. Our method extends the reweighted wake sleep framework to sequential settings by using Smoothing SMC to estimate posterior expectations. We evaluated our framework on a number of tasks, including inference in discrete latent variable models, where we outperformed prior methods. In future work, we will apply NAS-X to nonlinear dynamical systems models of neural data.

References

Jörg Bornschein and Yoshua Bengio. Reweighted wake-sleep. *arXiv preprint arXiv:1406.2751*, 2014.

Mark Briers, Arnaud Doucet, and Simon Maskell. Smoothing algorithms for state–space models. *Annals of the Institute of Statistical Mathematics*, 62(1):61–89, 2010.

Zhengdao Chen, Baranidharan Raman, and Ari Stern. Structure-Preserving Numerical Integrators for Hodgkin–Huxley-Type Systems. *SIAM Journal on Scientific Computing*, 42(1):B273–B298, 2020.

Siddhartha Chib, Yasuhiro Omori, and Manabu Asai. Multivariate stochastic volatility. In *Handbook of Financial Time Series*, pages 365–400. Springer, 2009.

Peter Dayan and Laurence F. Abbott. *Theoretical Neuroscience: Computational and Mathematical Modeling of Neural Systems*. MIT press, 2005.

Pierre Del Moral. *Feynman-Kac formulae: Genealogical and Interacting Particle Systems with Applications*, volume 88. Springer, 2004.

Arnaud Doucet and Adam M. Johansen. A tutorial on particle filtering and smoothing: Fifteen years later. In Dan Crisan and Boris Rozovsky, editors, *The Oxford Handbook of Nonlinear Filtering*, pages 656–704. Oxford University Press, 2011.

Emily Fox, Erik Sudderth, Michael Jordan, and Alan Willsky. Nonparametric Bayesian Learning of Switching Linear Dynamical Systems. In *Advances in Neural Information Processing Systems*, volume 21, 2008.

Marco Fraccaro, Søren Kaae Sønderby, Ulrich Paquet, and Ole Winther. Sequential neural models with stochastic layers. *Advances in Neural Information Processing Systems*, 29, 2016.

Shixiang Shane Gu, Zoubin Ghahramani, and Richard E. Turner. Neural Adaptive Sequential Monte Carlo. *Advances in Neural Information Processing Systems*, 28, 2015.

Pieralberto Guarniero, Adam M. Johansen, and Anthony Lee. The iterated auxiliary particle filter. *Journal of the American Statistical Association*, 112(520):1636–1647, 2017.

Geoffrey E Hinton, Peter Dayan, Brendan J Frey, and Radford M Neal. The "wake-sleep" algorithm for unsupervised neural networks. *Science*, 268(5214):1158–1161, 1995.

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

Alan L. Hodgkin and Andrew F. Huxley. A quantitative description of membrane current and its application to conduction and excitation in nerve. *The Journal of Physiology*, 117 (4):500, 1952.

Michael I Jordan. Serial order: A parallel distributed processing approach. In *Advances in Psychology*, volume 121, pages 471–495. Elsevier, 1997.

Diederik Kingma and Max Welling. Auto-encoding variational Bayes. In *2nd International Conference on Learning Representations*, 2014.

Diederik Kingma, Jimmy Ba, Yoshua Bengio, and Yann LeCun. Adam: A method for stochastic optimization. In *3rd International Conference on Learning Representations*, 2015.

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

Dieterich Lawson, George Tucker, Christian A. Naesseth, Chris Maddison, Ryan P. Adams, and Yee Whye Teh. Twisted Variational Sequential Monte Carlo. In *Third Workshop on Bayesian Deep Learning (NeurIPS)*, 2018.

Dieterich Lawson, George Tucker, Bo Dai, and Rajesh Ranganath. Energy-inspired models: Learning with sampler-induced distributions. *Advances in Neural Information Processing Systems*, 32, 2019.

Dieterich Lawson, Allan Raventós, Andrew Warrington, and Scott Linderman. SIXO: Smoothing Inference with Twisted Objectives. In Alice H. Oh, Alekh Agarwal, Danielle Belgrave, and Kyunghyun Cho, editors, *Advances in Neural Information Processing Systems*, 2022.

Scott W. Linderman, Matthew J. Johnson, Andrew C. Miller, Ryan P. Adams, David M. Blei, and Liam Paninski. Bayesian learning and inference in recurrent switching linear dynamical systems. In *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2017.

Chris J. Maddison, Dieterich Lawson, George Tucker, Nicolas Heess, Mohammad Norouzi, Andriy Mnih, Arnaud Doucet, and Yee Whye Teh. Filtering variational objectives. *Advances in Neural Information Processing Systems*, 30, 2017.

Christian A. Naesseth, Fredrik Lindsten, Thomas B. Schön, et al. Elements of Sequential Monte Carlo. *Foundations and Trends® in Machine Learning*, 12(3):307–392, 2019.

Art B Owen. *Monte Carlo Theory, Methods and Examples.* Stanford, 2013.

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

Masashi Sugiyama, Taiji Suzuki, and Takafumi Kanamori. *Density ratio estimation in machine learning.* Cambridge University Press, 2012.

Nick Whiteley and Anthony Lee. Twisted particle filters. *The Annals of Statistics*, 42(1): 115–141, 2014.

David Zoltowski, Jonathan Pillow, and Scott Linderman. A general recurrent state space framework for modeling neural dynamics during decision-making. In *Proceedings of the 37th International Conference on Machine Learning*, pages 11680–11691. PMLR, 2020.

Appendix A. Derivations

Gradient of Inclusive KL Divergence Below, we derive the gradient of the inclusive KL divergence for a generic Markovian model. In this derivation, we assume there are no shared parameters between the proposal and model.

$$-\nabla_\phi \mathrm{KL}(p_\theta||q_\phi) = \nabla_\phi \int p_\theta(\mathbf{x}_{1:T}|\mathbf{y}_{1:T}) \log q_\phi(\mathbf{x}_{1:T}|\mathbf{y}_{1:T}) d\mathbf{x}_{1:T} \tag{11}$$

$$= \int p_\theta(\mathbf{x}_{1:T}|\mathbf{y}_{1:T}) \nabla_\phi \log q_\phi(\mathbf{x}_{1:T}|\mathbf{y}_{1:T}) d\mathbf{x}_{1:T} \tag{12}$$

$$= \int p_\theta(\mathbf{x}_{1:T}|\mathbf{y}_{1:T}) \nabla_\phi \left(\sum_t \log q_\phi(\mathbf{x}_t|\mathbf{x}_{t-1}, \mathbf{y}_{t:T}) \right) d\mathbf{x}_{1:T} \tag{13}$$

$$= \sum_t \int p_\theta(\mathbf{x}_{1:T}|\mathbf{y}_{1:T}) \nabla_\phi \log q_\phi(\mathbf{x}_t|\mathbf{x}_{t-1}, \mathbf{y}_{t:T}) d\mathbf{x}_{1:T} \tag{14}$$

$$= \sum_t \mathbb{E}_{p_\theta(\mathbf{x}_{1:T}|\mathbf{y}_{1:T})} \left[\nabla_\phi \log q_\phi(\mathbf{x}_t|\mathbf{x}_{t-1}, \mathbf{y}_{t:T}) \right] \tag{15}$$

Gradient of the Marginal Likelihood

$$\nabla_\theta \log p(\mathbf{y}_{1:T}) = \nabla_\theta \log \int p_\theta(\mathbf{x}_{1:T}, \mathbf{y}_{1:T}) d\mathbf{y}_{1:T} \tag{16}$$

$$= \frac{1}{p_\theta(\mathbf{y}_{1:T})} \int \nabla_\theta p_\theta(\mathbf{x}_{1:T}, \mathbf{y}_{1:T}) d\mathbf{x}_{1:T} \tag{17}$$

$$= \frac{1}{p_\theta(\mathbf{y}_{1:T})} \int p_\theta(\mathbf{x}_{1:T}, \mathbf{y}_{1:T}) \nabla_\theta \log p_\theta(\mathbf{x}_{1:T}, \mathbf{y}_{1:T}) d\mathbf{x}_{1:T} \tag{18}$$

$$= \int p_\theta(\mathbf{x}_{1:T}|\mathbf{y}_{1:T}) \nabla_\theta \log p_\theta(\mathbf{x}_{1:T}, \mathbf{y}_{1:T}) d\mathbf{x}_{1:T} \tag{19}$$

$$= \int p_\theta(\mathbf{x}_{1:T}|\mathbf{y}_{1:T}) \nabla_\theta \sum_t \log p_\theta(\mathbf{y}_t, \mathbf{x}_t|\mathbf{x}_{t-1}) d\mathbf{x}_{1:T} \tag{20}$$

$$= \sum_t \int p_\theta(\mathbf{x}_{1:T}|\mathbf{y}_{1:T}) \nabla_\theta \log p_\theta(\mathbf{y}_t, \mathbf{x}_t|\mathbf{x}_{t-1}) d\mathbf{x}_{1:T} \tag{21}$$

$$= \sum_t \mathbb{E}_{p_\theta(\mathbf{x}_{1:T}|\mathbf{y}_{1:T})} \left[\nabla_\theta \log p_\theta(\mathbf{y}_t, \mathbf{x}_t|\mathbf{x}_{t-1}) \right] \tag{22}$$

Appendix B. LGSSM

Model Details We consider a one-dimensional linear Gaussian state space model with joint distribution

$$p(\mathbf{x}_{1:T}, \mathbf{y}_{1:T}) = \mathcal{N}(\mathbf{x}_1; 0, \sigma_x^2) \prod_{t=2}^{T} \mathcal{N}(\mathbf{x}_{t+1}; \mathbf{x}_t, \sigma_x^2) \prod_{t=1}^{T} \mathcal{N}(\mathbf{y}_t; \mathbf{x}_t, \sigma_y^2). \tag{23}$$

In our experiments we set the dynamics variance $\sigma_x^2 = 1.0$ and the observation variance $\sigma_y^2 = 1.0$.

Algorithm 1: NAS-X

Procedure `NAS-X(`θ_0, ϕ_0, ψ_0, $\mathbf{y}_{1:T}$`)`

  while *not converged* **do**

    $\psi \leftarrow$ `tilt-training(`θ, ψ`)`

    $\mathbf{x}_{1:T}^{1:N}, \overline{w}_{1:T}^{1:N} \leftarrow \mathrm{SMC}(\{p_\theta(\mathbf{x}_{1:t}, \mathbf{y}_{1:t}), r_\psi(\mathbf{x}_t, \mathbf{y}_{t+1:T})\}_{t=1}^T)$

    $\Delta\theta = \sum_{t=1}^T \sum_{i=1}^N \overline{w}_t^{(i)} \nabla_\theta \log p_\theta(\mathbf{x}_t^{(i)}, \mathbf{y}_t \mid \mathbf{x}_{t-1}^{(i)})$

    $\Delta\phi = -\sum_{t=1}^T \sum_{i=1}^N \overline{w}_t^{(i)} \nabla_\phi \log q_\phi(\mathbf{x}_t^{(i)} \mid \mathbf{x}_{t-1}^{(i)}, \mathbf{y}_{t:T})$

    $\theta =$ `grad-step(`θ, $\Delta\theta$`)`

    $\phi =$ `grad-step(`ϕ, $\Delta\phi$`)`

  end

return θ', ϕ'

Proposal Parameterization For both NAS-X and NASMC, we use a mean-field Gaussian proposal factored over time

$$q(x_{1:T}) = \prod_{t=1}^T q_t(x_t) = \prod_{t=1}^T \mathcal{N}(x_t; \mu_t, \sigma_t^2), \tag{24}$$

with parameters $\mu_{1:T}$ and $\sigma_{1:T}^2$ corresponding to the means and variances at each timestep. In total, we learn $2T$ proposal parameters.

Twist Parametrization We parameterize the twist as a quadratic function in x_t whose coefficients are functions of the observations and time step and are learned via the density ratio estimation procedure described in (Lawson et al., 2022). We chose this form to match the analytic log density ratio for the model defined in Eq 23. Given that $p(x_{1:T}, y_{1:T})$ is a multivariate Gaussian, we know that $p(x_t \mid y_{t+1:T})$ and $p(x_t)$ are both marginally Gaussian. Let

$$\begin{aligned} p(x_t \mid y_{t+1:T}) &\triangleq \mathcal{N}(\mu_1, \sigma_1^2) \\ p(x_t) &\triangleq \mathcal{N}(0, \sigma_1^2) \end{aligned}$$

Then,

$$\begin{aligned} \log\left(\frac{p(x_t \mid y_{t+1:T})}{p(x_t)}\right) &= \log \mathcal{N}(x_t; \mu_1, \sigma_1^2) - \log \mathcal{N}(x_t; 0, \sigma_2^2) \\ &= \log Z(\sigma_1) - \frac{1}{2\sigma_1^2} x_t^2 + \frac{\mu_1}{\sigma_1^2} x_t - \frac{\mu_1^2}{2\sigma_1^2} - \log Z(\sigma_2) + \frac{1}{2\sigma^2} x_t^2 \end{aligned}$$

where $Z(\sigma) = \frac{1}{\sigma\sqrt{2\pi}}$, so $\log Z(\sigma) = -\log(\sigma\sqrt{2\pi})$.

Collecting terms gives:

$$\begin{aligned} &-\log(\sigma_1\sqrt{2\pi}) + \log(\sigma_2\sqrt{2\pi}) \\ &-\frac{1}{2}\left(\frac{1}{\sigma_1^2} - \frac{1}{\sigma_2^2}\right) x_t^2 \\ &+\frac{\mu_1}{\sigma_1^2} x_t \\ &-\frac{\mu_1^2}{2\sigma_1^2} \end{aligned}$$

So we'll define

$$\begin{aligned} a &\triangleq -\frac{1}{2}\left(\frac{1}{\sigma_1^2} - \frac{1}{\sigma_2^2}\right) \\ b &\triangleq \frac{\mu_1}{\sigma_1^2} \\ c &\triangleq -\frac{\mu_1^2}{2\sigma_1^2} - \log(\sigma_1\sqrt{2\pi}) + \log(\sigma_2\sqrt{2\pi}) \end{aligned}$$

We'll explicitly model $\log \sigma_1^2$, $\log \sigma_2^2$ and μ_1. Both $\log \sigma_1^2$ and $\log \sigma_2^2$ are only functions of t, not of $y_{t+1:T}$, so those can be vectors of shape T initialized at 0. μ_1 is a linear function of $y_{t+1:T}$ and t, so that can be parameterized by a set of $T \times T$ weights, initialized to $1/T$ and T biases initialized to 0.

Training Details We use a batch size of 32 for the density ratio estimation step. Since we do not perform model learning, we do not repeatedly alternate between tilt training and proposal training for NAS-X. Instead, we first train the tilt for 3,000,000 iterations with a batch size of 32 using samples from the model. We then train the proposal for $750,000$ iterations. For the tilt, we used Adam with a learning rate schedule that starts with a constant learning rate of $1e-3$, decays the learning by 0.3 and 0.33 at $100,000$ and $300,000$ iterations. For the proposal, we used Adam with a constant learning rate of $1e-3$. For NASMC, we only train the proposal.

Evaluation In the right panel of Figure 1, we compare the bound gaps of NAS-X and NASMC averaged across 20 different samples from the generative model. To obtain the bound gap for NAS-X, we run SMC 16 times with 128 particles and with the learned proposal and twists. We then record the average log marginal likelihood. For NASMC, we run SMC with the current learned proposal (without any twists).

Appendix C. rSLDS

Model details The generative model is as follows. At each time t, there is a discrete latent state $z_t \in \{1, \ldots, 4\}$ as well as a two-dimensional continuous latent state $x_t \in \mathbb{R}^2$. The discrete state transition probabilities are given by

$$p(z_{t+1} = i \mid z_t = j, x_t) \propto \exp\left(r_i + R_i^T x_{t-1}\right) \tag{25}$$

Here R_i and r_i are weights for the discrete state z_i.

These discrete latent states dictates two-dimensional latent state $x_t \in \mathbb{R}^2$ which evolves according to linear Gaussian dynamics.

$$x_{t+1} = A_{z_{t+1}} x_t + b_{z_{t+1}} + v_t, \qquad v_t \sim^{\text{iid}} \mathcal{N}(0, Q_{z_{t+1}}) \tag{26}$$

Here $A_k, Q_k \in \mathbb{R}^{2x2}$ and $b_k \in \mathbb{R}^2$. Importantly, from Equations 26 and 25 we see that the dynamics of the continuous latent states and discrete latents are coupled. The discrete latent states index into specific linear dynamics and the discrete transition probabilities depend on the continuous latent state.

The observations $y_t \in \mathbb{R}^{10}$ are linear projections of the continuous latent state x_t with some additive Gaussian noise.

$$y_t = Cx_t + d + w_t, \qquad v_t \sim^{\text{iid}} \mathcal{N}(0, S) \tag{27}$$

Here $C, S \in \mathbb{R}^{10x10}$ and $d \in \mathbb{R}^{10}$.

Proposal Parameterization We use a mean-field proposal distribution factorized over the discrete and continuous latent variables (i.e. $q(\mathbf{z}_{1:T}, \mathbf{x}_{1:T}) = q(\mathbf{z}_{1:T})q(\mathbf{x}_{1:T})$). For the continuous states, $q(\mathbf{x}_{1:T})$ is a Gaussian factorized over time with parameters $\mu_{1:T}$ and $\sigma^2_{1:T}$. For the discrete states, $q(\mathbf{z}_{1:T})$ is a Categorical distribution over K categories factorized over time with parameters $p^{1:K}_{1:T}$. In total, we learn $2T + TK$ proposal parameters.

Twist Parameterization We parameterize the twists using a recurrent neural network (RNN) that is trained using density ratio estimation. To produce the twist values at each timestep, we first run a RNN backwards over the observations $\mathbf{y}_{1:T}$ to produce a sequence of encodings $\mathbf{e}_{1:T-1}$. We then concatenate the encodings of $\mathbf{x}_t$ and $\mathbf{z}_t$ into a single vector and pass that vector into an MLP which outputs the twist values at each timestep. The RNN has one layer with 128 hidden units. The MLP has 131 hidden units and ReLU activations.

Model Parameter Evaluation We closely follow the parameter initialization strategy employed by Linderman et al. (2017). First, we use PCA to obtain a set of continuous latent states and initialize the matrices C and d. We then fit an autoregressive HMM to the estimated continuous latent states in order to initialize the dynamics matrices $\{A_k, b_k\}$. Importantly, we do not initialize the proposal with the continuous latent states described above.

Training Details We use a batch size of 32 for the density ratio estimation step. We alternate between 100 steps of tilt training and 100 steps of proposal training for a total of 50,000 training steps in total. We used Adam and considered a grid search over the model, proposal, and tilt learning rates. In particular, we considered learning rates of $1e-4, 1e-3, 1e-2$ for the model, proposal, and tilt.

Bootstrap Bound Evaluation To obtain the log marginal likelihood bounds and standard deviations in Table 1, we ran a bootstrapped particle filter (BPF) with the learned model parameters for all three methods (NAS-X, NASMC, Laplace EM) using 1024 particles. We repeat this across 30 random seeds. Initialization of the latent states was important for a fair comparison. To initialize the latent states, for NAS-X and NASMC, we simply sampled from the learned proposal at time $t = 0$. To initialize the latent state for Laplace EM, we sampled from a Gaussian distribution with the learned dynamics variance at $t = 0$.

Appendix D. Hodgkin-Huxley Model

D.1. Model

We also evaluated NAS-X on the Hodgkin-Huxley (HH) biophysical model of neural action potentials (Hodgkin and Huxley, 1952; Dayan and Abbott, 2005). Our experimental setup was constructed to broadly match Lawson et al. (2022), and used a single-compartment model with voltage dynamics defined by

$$\begin{aligned} C_m \frac{dV}{dt} &= I_{\text{ext}} - I_{Na} - I_K - I_L \\ I_{Na} &= \bar{g}_{Na} m^3 h (V - E_{Na}) \\ I_K &= \bar{g}_K n^4 (V - E_K) \\ I_L &= \bar{g}_L (V - E_L) \end{aligned} \tag{28}$$

where C_m is the membrane capacitance, V is the membrane potential, I_{ext} is the injected current, I_{Na}, I_K, and I_L are the sodium, potassium, and leak currents respectively, $\bar{g}_{Na}$, $\bar{g}_K$, and $\bar{g}_L$ are the maximum conductances for sodium, potassium, and leak channels, and E_{Na}, E_K, and E_L are the reversal potentials for sodium, potassium, and leak channels.

The gating variables m, h, and n obey the following first-order kinetics:

$$\begin{aligned} \frac{dm}{dt} &= \alpha_m(V)(1 - m) - \beta_m(V)m \\ \frac{dh}{dt} &= \alpha_h(V)(1 - h) - \beta_h(V)h \\ \frac{dn}{dt} &= \alpha_n(V)(1 - n) - \beta_n(V)n \end{aligned} \tag{29}$$

where α_m, β_m, α_h, β_h, α_n, and β_n are the voltage-dependent rate constants for the gating variables. The specific forms for the α and β functions are available in Dayan and Abbott (2005).

This system of ordinary differential equations defines a nonlinear dynamical system with a four-dimensional state space: the instantaneous membrane potential and the activation states of the ion gates.

As in Lawson et al. (2022), we use a probabilistic version of the original HH model that adds zero-mean Gaussian noise to the unconstrained gates n, m, and h. The observations are produced by adding Gaussian noise with variance σ_y^2 to the instantaneous membrane potential. Specifically, let $\mathbf{x}_t$ be the state vector of the system at time t containing (V_t, m_t, h_t, n_t), and let $\varphi_{dt}(\mathbf{x})$ be a function that integrates the system of ODEs defined above for a step of length dt. Then the probabilistic HH model can be written as

$$p(\mathbf{x}_{1:T}, \mathbf{y}_{1:T}) = p(\mathbf{x}_1) \prod_{t=2}^{T} p(\mathbf{x}_t \mid \varphi_{dt}(\mathbf{x}_{t-1})) \prod_{t=1}^{T} \mathcal{N}(\mathbf{y}_t; \mathbf{x}_{t,1}, \sigma_y^2) \tag{30}$$

where the 4-D state distributions $p(\mathbf{x}_1)$ and $p(\mathbf{x}_t \mid \varphi_{dt}(\mathbf{x}_{t-1}))$ are defined as

$$p(\mathbf{x}_t \mid \varphi_{dt}(\mathbf{x}_{t-1})) = \mathcal{N}(\mathbf{x}_{t,1}; \varphi_{dt}(\mathbf{x}_{t-1})_1, \sigma_{x,1}^2) \prod_{i=2}^{4} \text{LogitNormal}(\mathbf{x}_{t,i}; \varphi_{dt}(\mathbf{x}_{t-1})_i, \sigma_{x,i}^2). \tag{31}$$

In words, we add Gaussian noise to the voltage ($\mathbf{x}_{t,1}$) and logit-normal noise to the gate states n, m, and h. The logit-normal is defined as the distribution of a random variable whose logit has a Gaussian distribution, or equivalently it is a Gaussian transformed by the sigmoid function and renormalized. We chose the logit-normal because its values are bounded between 0 and 1, which is necessary for the gate states. We implemented φ_{dt} using a Strang splitting approach as described in Chen et al. (2020).

D.2. Problem Setting

We evaluated NAS-X, NASMC, SIXO, and its predecessor FIVO on inference in the probabilistic HH model. For this task we sampled 10,000 noisy voltage traces from a fixed model and used each method to train proposals (and possibly twists) to compute the marginal likelihood assigned to the data under the true model.

As in Lawson et al. (2022), we sampled trajectories of length 50 milliseconds, with a single noisy voltage observation every millisecond. The stability of our ODE integrator allowed us to integrate at $dt = 0.1$ms, meaning that there were 10 latent states per observation.

Proposal and Twist Details Each proposal was parameterized using the combination of a bidirectional recurrent neural network (RNN) that conditioned on all observed noisy voltages as well as a dense network that conditioned on the RNN hidden state and the previous latent state $\mathbf{x}_{t-1}$ (Hochreiter and Schmidhuber, 1997; Jordan, 1997). The twists for SIXO and NAS-X were parameterized using an RNN run in reverse over the observations combined with a dense network that conditioned on the reverse RNN hidden state and the latent being 'twisted', $\mathbf{x}_t$. Both the proposal and twists were learned in an amortized manner, i.e. they were shared across all trajectories. All RNNs had a single hidden layer of size 64, as did the dense networks. All models were fit with ADAM (Kingma et al., 2015) with proposal learning rate of 10^{-4} and tilt learning rate of 10^{-3}.

A crucial aspect of fitting the proposals was defining them in terms of a 'residual' from the prior, a technique known as Res_q (Fraccaro et al., 2016). In our setting, we defined the true proposal density as proportional to the product of a unit-variance Gaussian centered at $\varphi(\mathbf{x}_t)$ and a Gaussian with parameters output from the RNN proposal.

D.3. Experimental Results

In Figure 3 we plot the performance of proposals and twists trained with 4 particles and evaluated across a range of particle numbers. All methods except FIVO perform roughly the same when evaluated with 256 particles, but with lower numbers of evaluation particles the smoothing methods emerge as more particle-efficient than the filtering methods. To achieve NAS-X's inference performance with 4 particles, NASMC would need 256 particles, a 64-times increase. NAS-X is also more particle-efficient than SIXO, achieving on average a 2x particle efficiency improvement.

The FIVO method with a parametric proposal drastically underperformed all smoothing methods as well as NASMC, indicating that the combination of filtering SMC and the exclusive KL divergence leads to problems optimizing the proposal parameters. To compensate, we also evaluated the performance of "FIVO-BS", a filtering method that uses a bootstrap proposal. This method is identical to a bootstrap particle filter, i.e. it proposes from the

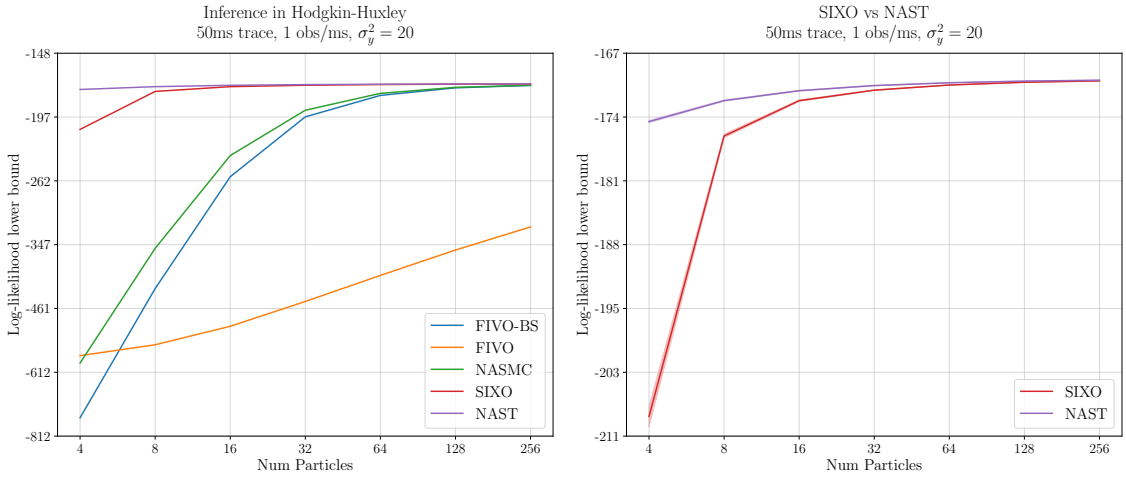

Figure 3: **HH inference performance across different numbers of particles.** **(left)** Log-likelihood lower bounds for proposals trained with 4 particles and evaluated across a range of particle numbers. NAS-X's inference performance decays only minimally as the number of particles is decreased, while all other methods experience significant performance degradation. **(right)** A comparison of SIXO and NAS-X containing the same values as the left panel, but zoomed in. NAS-X is roughly twice as particle efficient as SIXO, and outperforms SIXO by roughly 34 nats at 4 particles.

model and has no trainable parameters. FIVO-BS far outperforms standard FIVO, and is only marginally worse than NASMC in this setting.

In Figure 4 we investigate these results qualitatively by examining the inferred voltage traces of each method. We see that NASMC struggles to produce accurate spike timings and generates many spurious spikes, likely because it is unable to incorporate future information into its proposal or resampling method. SIXO performs better than NASMC, accurately inferring the timing of most spikes but resampling at a high rate. High numbers of resampling events can lead to particle degeneracy and poor inferences. NAS-X is able to correctly infer the voltage across the whole trace with no suprious or mistimed spikes. Furthermore NAS-X rarely resamples, indicating it has learned a high-quality proposal that does not generate low-quality particles that must be resampled away. These qualitative results seem to support the quantitative results in Figure 3 — SIXO's high resampling rate and NASMC's filtering approach lead to lower bound values.

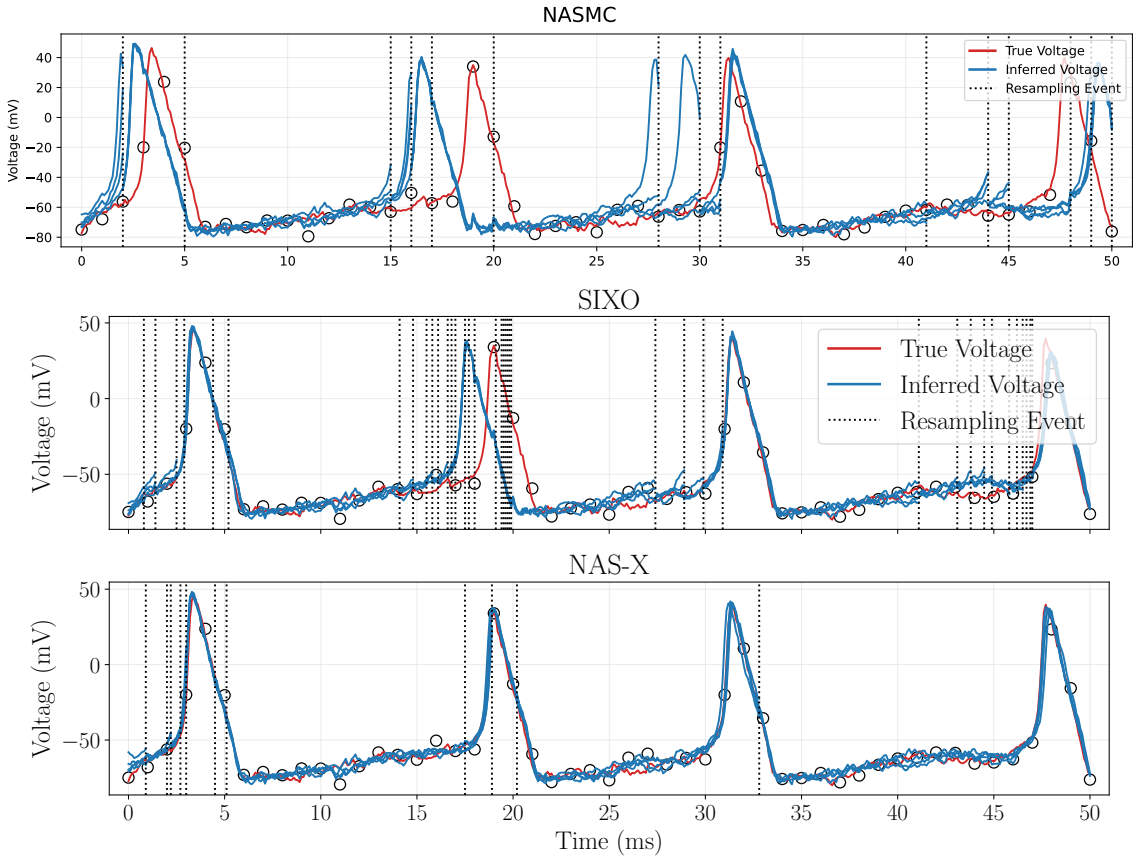

Figure 4: **Inferred voltage traces for NASMC, SIXO, and NAS-X.** **(top)** NASMC exhibits poor performance, incorrectly inferring the timing of most spikes. **(middle)** SIXO's inferred voltage traces are more accurate than NASMC's with only a single mistimed spike, but SIXO generates a high number of resampling events leading to particle degeneracy. **(bottom)** NAS-X perfectly infers the latent voltage with no mistimed spikes, and resamples very infrequently.