7 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.

$$\begin{aligned}
-\nabla_\phi \mathrm{KL}(p_\theta || q_\phi) &= \nabla_\phi \int p_\theta(\mathbf{x}_{1:T}|\mathbf{y}_{1:T}) \log q_\phi(\mathbf{x}_{1:T}|\mathbf{y}_{1:T}) d\mathbf{x}_{1:T} && (13)\\
&= \int p_\theta(\mathbf{x}_{1:T}|\mathbf{y}_{1:T}) \nabla_\phi \log q_\phi(\mathbf{x}_{1:T}|\mathbf{y}_{1:T}) d\mathbf{x}_{1:T} && (14)\\
&= \int p_\theta(\mathbf{x}_{1:T}|\mathbf{y}_{1:T}) \nabla_\phi \left(\sum_t \log q_\phi(\mathbf{x}_t|\mathbf{x}_{t-1}, \mathbf{y}_{t:T}) \right) d\mathbf{x}_{1:T} && (15)\\
&= \sum_t \int p_\theta(\mathbf{x}_{1:T}|\mathbf{y}_{1:T}) \nabla_\phi \log q_\phi(\mathbf{x}_t|\mathbf{x}_{t-1}, \mathbf{y}_{t:T}) d\mathbf{x}_{1:T} && (16)\\
&= \sum_t \mathbb{E}_{p_\theta(\mathbf{x}_{1:T}|\mathbf{y}_{1:T})} \left[\nabla_\phi \log q_\phi(\mathbf{x}_t|\mathbf{x}_{t-1}, \mathbf{y}_{t:T}) \right] && (17)
\end{aligned}$$

We use the assumption that there are no shared parameters in the second equality.

Gradient of the Marginal Likelihood We derive the gradients for the marginal likelihood. This identity is known as Fisher's identity.

$$\begin{aligned}
\nabla_\theta \log p(\mathbf{y}_{1:T}) &= \nabla_\theta \log \int p_\theta(\mathbf{x}_{1:T}, \mathbf{y}_{1:T}) d\mathbf{y}_{1:T} && (18)\\
&= \frac{1}{p_\theta(\mathbf{y}_{1:T})} \nabla_\theta \int p_\theta(\mathbf{x}_{1:T}, \mathbf{y}_{1:T}) d\mathbf{x}_{1:T} && (19)\\
&= \frac{1}{p_\theta(\mathbf{y}_{1:T})} \int \nabla_\theta p_\theta(\mathbf{x}_{1:T}, \mathbf{y}_{1:T}) d\mathbf{x}_{1:T} && (20)\\
&= \frac{1}{p_\theta(\mathbf{y}_{1:T})} \int p_\theta(\mathbf{x}_{1:T}, \mathbf{y}_{1:T}) \nabla_\theta \log p_\theta(\mathbf{x}_{1:T}, \mathbf{y}_{1:T}) d\mathbf{x}_{1:T} && (21)\\
&= \int p_\theta(\mathbf{x}_{1:T}|\mathbf{y}_{1:T}) \nabla_\theta \log p_\theta(\mathbf{x}_{1:T}, \mathbf{y}_{1:T}) d\mathbf{x}_{1:T} && (22)\\
&= \int p_\theta(\mathbf{x}_{1:T}|\mathbf{y}_{1:T}) \nabla_\theta \sum_t \log p_\theta(\mathbf{y}_t, \mathbf{x}_t|\mathbf{x}_{t-1}) d\mathbf{x}_{1:T} && (23)\\
&= \sum_t \int p_\theta(\mathbf{x}_{1:T}|\mathbf{y}_{1:T}) \nabla_\theta \log p_\theta(\mathbf{y}_t, \mathbf{x}_t|\mathbf{x}_{t-1}) d\mathbf{x}_{1:T} && (24)\\
&= \sum_t \mathbb{E}_{p_\theta(\mathbf{x}_{1:T}|\mathbf{y}_{1:T})} \left[\nabla_\theta \log p_\theta(\mathbf{y}_t, \mathbf{x}_t|\mathbf{x}_{t-1}) \right] && (25)
\end{aligned}$$

The key steps were the log-derivative trick and Bayes rule.

8 LGSSM

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

$$p(\mathbf{x}_{1:T}, \mathbf{y}_{1:T}) = \mathcal{N}(\mathbf{x}_1; 0, \sigma_x^2) \prod_{t=2}^{T} \mathcal{N}(\mathbf{x}_{t+1}; \mathbf{x}_t, \sigma_x^2) \prod_{t=1}^{T} \mathcal{N}(\mathbf{y}_t; \mathbf{x}_t, \sigma_y^2). \quad (26)$$

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

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

$$q(x_{1:T}) = \prod_{t=1}^{T} q_t(x_t) = \prod_{t=1}^{T} \mathcal{N}(x_t; \mu_t, \sigma_t^2), \tag{27}$$

with parameters $\mu_{1:T}$ and $\sigma^2_{1:T}$ 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 26. 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).

9 rSLDS

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

$$p(z_{t+1} = i \mid z_t = j, x_t) \propto \exp\left(r_i + R_i^T x_{t-1}\right) \tag{28}$$

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.

$$x_{t+1} = A_{z_{t+1}} x_t + b_{z_{t+1}} + v_t, \qquad v_t \sim^{\text{iid}} \mathcal{N}(0, Q_{z_{t+1}}) \tag{29}$$

Here $A_k, Q_k \in \mathbb{R}^{2x2}$ and $b_k \in \mathbb{R}^2$. Importantly, from Equations 29 and 28 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.

$$y_t = C x_t + d + w_t, \qquad v_t \sim^{\text{iid}} \mathcal{N}(0, S) \tag{30}$$

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 $t = 0$.

10 Inference in Squid Giant Axon Model

10.1 HH Model Definition

For the inference experiments (Section 5.3.1) we used a probabilistic version of the squid giant axon model 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 dynamics defined by

$$C_m \frac{dv}{dt} = I_{\text{ext}} - \bar{g}_{\text{Na}} m^3 h (v - E_{\text{Na}}) - \bar{g}_{\text{K}} n^4 (v - E_{\text{K}}) - g_{\text{leak}}(v - E_{\text{leak}}) \tag{31}$$

$$\frac{dm}{dt} = \alpha_m(v)(1-m) - \beta_m(v)m \tag{32}$$

$$\frac{dh}{dt} = \alpha_h(v)(1-h) - \beta_h(v)h \tag{33}$$

$$\frac{dn}{dt} = \alpha_n(v)(1-n) - \beta_n(v)n \tag{34}$$

where C_m is the membrane capacitance; v is the potential difference across the membrane; I_{ext} is the external current; $\bar{g}_{\text{Na}}$, $\bar{g}_{\text{K}}$, and $\bar{g}_{\text{leak}}$ are the maximum conductances for sodium, potassium, and leak channels; E_{Na}, E_{K}, and E_{leak} are the reversal potentials for the sodium, potassium, and leak channels; m and h are subunit states for the sodium channels and n is the subunit state for the potassium channels. The functions α and β that define the dynamics for n, m, and h are defined as

$$\alpha_m(v) = \frac{-4 - v/10}{\exp(-4 - v/10) - 1}, \quad \beta_m(v) = 4 \cdot \exp((-65 - v)/18) \tag{35}$$

$$\alpha_h(v) = 0.07 \cdot \exp((-65 - v)/20), \quad \beta_h(v) = \frac{1}{\exp(-3.5 - v/10) + 1} \tag{36}$$

$$\alpha_n(v) = \frac{-5.5 - v/10}{\exp(-5.5 - v/10) - 1}, \quad \beta_n(v) = 0.125 \cdot \exp((-65 - v)/80) \tag{37}$$

This system of ordinary differential equations defines a nonlinear dynamical system with a four-dimensional state space: the instantaneous membrane potential v and the ion gate subunit states n, m, and h.

As in Lawson et al. [2022], we use a probabilistic version of the original HH model that adds zero-mean Gaussian noise to both the membrane voltage v and the "unconstrained" subunit states. The observations are produced by adding Gaussian noise with variance σ_y^2 to the membrane potential v.

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

$$p(\mathbf{x}_{1:T}, \mathbf{y}_{1:T}) = p(\mathbf{x}_1) \prod_{t=2}^{T} p(\mathbf{x}_t \mid \varphi_{dt}(\mathbf{x}_{t-1})) \prod_{t=1}^{T} \mathcal{N}(\mathbf{y}_t; \mathbf{x}_{t,1}, \sigma_y^2) \tag{38}$$

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

$$p(\mathbf{x}_t \mid \varphi_{dt}(\mathbf{x}_{t-1})) = \mathcal{N}(\mathbf{x}_{t,1}; \varphi_{dt}(\mathbf{x}_{t-1})_1, \sigma_{x,1}^2) \prod_{i=2}^{4} \text{LogitNormal}(\mathbf{x}_{t,i}; \varphi_{dt}(\mathbf{x}_{t-1})_i, \sigma_{x,i}^2). \tag{39}$$

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.

Problem Setting For the inference experiments 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.

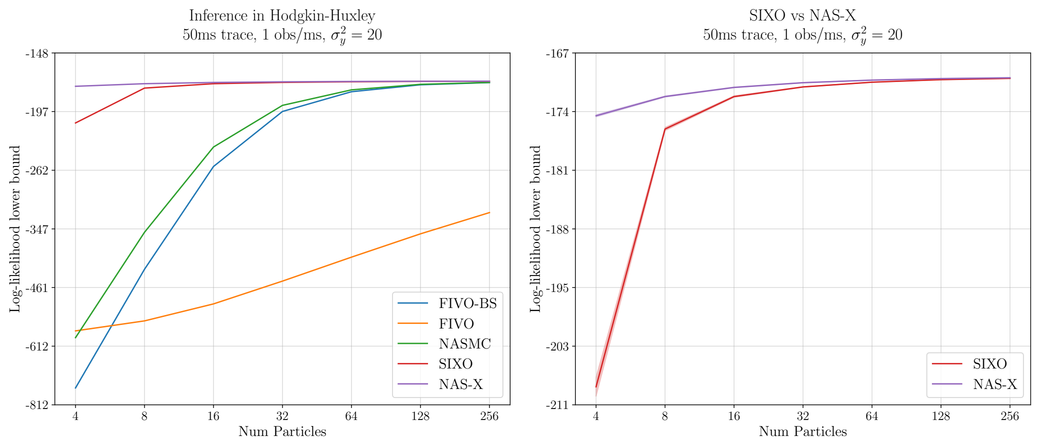

Figure 5: **HH inference performance across different numbers of particles.**
(left) Log-likelihood lower bounds for proposals trained with 4 particles and evaluated across a range of particle numbers. NAS-X's inference performance decays only minimally as the number of particles is decreased, while 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.

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.

10.2 Experimental Results

In Figure 5 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 model and has no trainable parameters. FIVO-BS far outperforms standard FIVO, and is only marginally worse than NASMC in this setting.

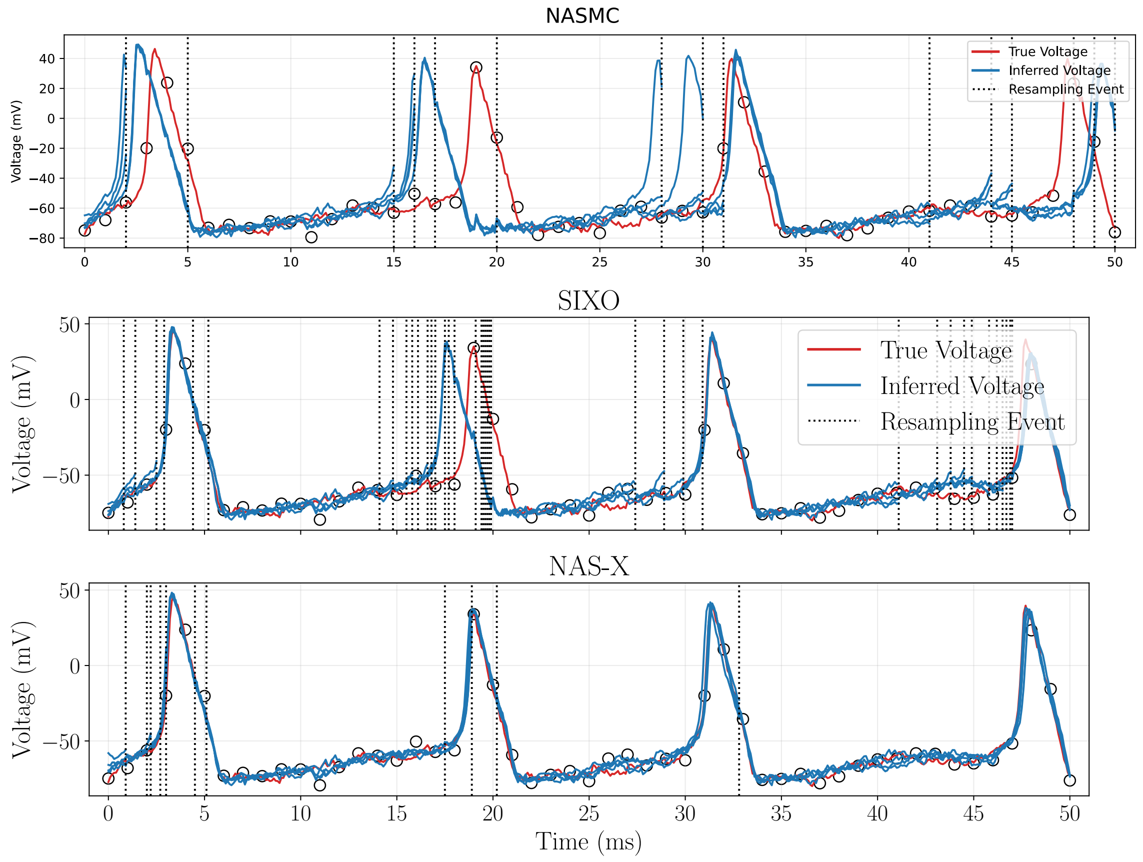

Figure 6: **Inferred voltage traces for NASMC, SIXO, and NAS-X.**
(top) NASMC exhibits poor performance, incorrectly inferring the timing of most spikes. **(middle)** SIXO's inferred voltage traces are more accurate than NASMC's with only a single mistimed spike, but SIXO generates a high number of resampling events leading to particle degeneracy. **(bottom)** NAS-X perfectly infers the latent voltage with no mistimed spikes, and resamples very infrequently.

In Figure 6 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 5 — SIXO's high resampling rate and NASMC's filtering approach lead to lower bound values.

11 Model Learning in Mouse Pyramidal Neuron Model

11.1 Model Definition

For the model learning experiments in Section 5.3.2 we used a generalization of the Hodgkin-Huxley model developed for modeling mouse visual cortex neurons by the Allen Institute for Brain Science Wang et al. [2020], AIBS [2017]. Specifically we used the perisomatic model with ID 482657528 developed to model cell 480169178. The model is detailed in the whitepaper AIBS [2017] and the accompanying code, but we reproduce the details here to ensure our work is self-contained.

Table 2: Train Bound comparison

Metric	NAS-X	SIXO
$\mathcal{L}_{\text{BPF}}^{256}$	-660.7003	-636.2579
$\mathcal{L}_{\text{train}}^{4}$	-664.3528	-668.6865
$\mathcal{L}_{\text{train}}^{8}$	-662.8712	-653.6352
$\mathcal{L}_{\text{train}}^{16}$	-662.0753	-644.8764
$\mathcal{L}_{\text{train}}^{32}$	-661.5387	-639.5388
$\mathcal{L}_{\text{train}}^{64}$	-660.8040	-636.5131
$\mathcal{L}_{\text{train}}^{128}$	-660.5102	-633.7875
$\mathcal{L}_{\text{train}}^{256}$	-660.3423	-632.1377

Similar to the squid giant axon model, the mouse visual cortex model is composed of ion channels that affect the current flowing in and out of the cell. Let $\mathcal{I}$ be the set of ions $\{\text{Na}^+, \text{Ca}^{2+}, \text{K}^+\}$. Each ion has associated with it

1. A set of channels that transport that ion, denoted C_i for $i \in \mathcal{I}$.
2. A reversal potential, E_i.
3. An instantaneous current density, I_i, which is computed by summing the current density flowing through each channel that transports that ion.

Correspondingly, let $\mathcal{C}$ be the set of all ion channels so that $\mathcal{C} = \bigcup_{i \in \mathcal{I}} C_i$. Each $c \in \mathcal{C}$ has associated with it

1. A maximum conductance density, $\overline{g}_c$.
2. A set of subunit states, referred to collectively as the vector λ_c. Let $\lambda_c \in [0,1]^{d_c}$, i.e. λ_c is a d_c-dimensional vector of values in the interval $[0, 1]$.
3. A function g_c that combines the gate values to produce a number in $[0, 1]$ that weights the maximum conductance density, $\overline{g}_c \cdot g_c(\lambda_c)$.
4. Functions $A_c(\cdot)$ and $b_c(\cdot)$ which compute the matrix and vector used in the ODE describing λ_c dynamics. A_c and b_c are functions of both the current membrane voltage v and calcium concentration inside the cell $[\text{Ca}^{2+}]_i$. If the number of subunits (i.e. the dimensionality of λ_c) is d_c, then the output of $A_c(v, [\text{Ca}^{2+}]_i)$ is a $d_c \times d_c$ diagonal matrix and the output of $b_c(v, [\text{Ca}^{2+}]_i)$ is a d_c-dimensional vector.

With this notation we can write the system of ODEs

$$C_m \frac{dv}{dt} = \frac{I_{\text{ext}}}{SA} - g_{\text{leak}}(v - E_{\text{leak}}) - \sum_{i \in \text{ions}} I_{\text{i}} \tag{40}$$

$$I_{\text{i}} = \sum_{c \in \text{C}_{\text{i}}} \overline{g}_c g_c(\lambda_c)(v - E_{\text{i}}) \tag{41}$$

$$\frac{d\lambda_c}{dt} = A_c(v, [\text{Ca}^{2+}]_i)\lambda_c + b_c(v, [\text{Ca}^{2+}]_i) \quad \forall c \in \mathcal{C} \tag{42}$$

$$\frac{d[\text{Ca}^{2+}]_i}{dt} = -kI_{\text{Ca}^{2+}} - \frac{[\text{Ca}^{2+}]_i - [\text{Ca}^{2+}]_{\text{min}}}{\tau}. \tag{43}$$

Most symbols are as described earlier, SA is the membrane surface area of the neuron, $[\text{Ca}^{2+}]_i$ is the calcium concentration inside the cell, $[\text{Ca}^{2+}]_{\text{min}}$ is the minimum interior calcium concentration with a value of 1 nanomolar, τ is the rate of removal of calcium with a value of 80 milliseconds, and k and is a constant with value

$$k = 10000 \cdot \frac{\gamma}{2 \cdot F \cdot \text{depth}} \tag{44}$$

where 10000 is a dimensional constant, γ is the percent of unbuffered free calcium, F is Faraday's constant, and depth is the depth of the calcium buffer with a value of 0.1 microns.

Because the concentration of calcium changes over time, this model calculates the reversal potential for calcium $E_{\mathrm{Ca}^{2+}}$ using the Nernst equation

$$E_{\mathrm{Ca}^{2+}} = \frac{G \cdot T}{2 \cdot F} \log\left(\frac{[\mathrm{Ca}^{2+}]_o}{[\mathrm{Ca}^{2+}]_i}\right) \tag{45}$$

where G is the gas constant, T is the temperature in Kelvin ($308.15°$), F is Faraday's constant, and $[\mathrm{Ca}^{2+}]_o$ is the extracellular calcium ion concentration which was set to 2 millimolar.

Probabilistic Model The probabilistic version of the deterministic ODEs was constructed similarly to the probabilistic squid giant axon model — Gaussian noise was added to the voltage and unconstrained gate states. One difference is that the system state now includes $[\mathrm{Ca}^{2+}]_i$ which is constrained to be greater than 0. To noise $[\mathrm{Ca}^{2+}]_i$ we added Gaussian noise in the log space, analagous to the logit-space noise for the gate states.

Model Size The 38 learnable parameters of the model include:

1. Conductances $\overline{g}$ for all ion channels (10 parameters).
2. Reversal potentials of sodium, potassium, and the non-specific cation: E_{K^+}, E_{Na^+}, and E_{NSC^+}.
3. The membrane surface area and specific capacitance.
4. Leak channel reversal potential and max conductance density.
5. The calcium decay rate and free calcium percent.
6. Gaussian noise variances for the voltage v and interior calcium concentration $[\mathrm{Ca}^{2+}]_i$.
7. Gaussian noise variances for all subunit states (16 parameters).
8. Observation noise variance.

The 18-dimensional state includes:

1. Voltage v
2. Interior calcium concentration $[\mathrm{Ca}^{2+}]_i$
3. All subunit states (16 dimensions)

11.2 Channel Definitions

In this section we provide a list of all ion channels used in the model. In the following equations we often use the function exprel which is defined as

$$\mathrm{exprel}(x) = \begin{cases} 1 & \text{if} \quad x = 0 \\ \dfrac{\exp(x) - 1}{x} & \text{otherwise} \end{cases} \tag{46}$$

A numerically stable implementation of this function was critical to training our models.

Additionally, many of the channel equations below contain a 'temperature correction' q_t that adjusts for the fact that the original experiments and Allen Institute experiments were not done at the same temperature. In those equations, T is the temperature in Celsius which was $35°$.

11.2.1 Transient Na$^+$

From Colbert and Pan [2002].

$$\lambda_c = (m, h), \quad g_c(\lambda_c) = m^3 h$$
$$\frac{1}{q_t}\frac{dm}{dt} = \alpha_m(v)(1 - m) - \beta_m(v)m$$
$$\frac{1}{q_t}\frac{dh}{dt} = \alpha_h(v)(1 - h) - \beta_h(v)h$$
$$q_t = 2.3^{\left(\frac{T-23}{10}\right)}$$

$$\alpha_m(v) = \frac{0.182 \cdot 6}{\text{exprel}(-(v+40)/6)}, \quad \beta_m(v) = \frac{0.124 \cdot 6}{\text{exprel}((v+40)/6)}$$
$$\alpha_h(v) = \frac{0.015 \cdot 6}{\text{exprel}((v+66)/6)}, \quad \beta_h(v) = \frac{0.015 \cdot 6}{\text{exprel}(-(v+66)/6)}$$

11.2.2 Persistent Na⁺

From Magistretti and Alonso [1999].

$$\lambda_c = h, \quad g_c(\lambda_c) = m_\infty h$$
$$m_\infty = \frac{1}{1+\exp(-(v+52.6)/4.6)}$$
$$\frac{1}{q_t}\frac{dh}{dt} = \alpha_h(v)(1-h) - \beta_h(v)h$$
$$q_t = 2.3^{\left(\frac{T-21}{10}\right)}$$

$$\alpha_h(v) = \frac{2.88\times10^{-6}\cdot 4.63}{\text{exprel}((v+17.013)/4.63)}, \quad \beta_h(v) = \frac{6.94\times10^{-6}\cdot 2.63}{\text{exprel}(-(v+64.4)/2.63)}$$

11.2.3 Hyperpolarization-activated cation conductance

From Kole et al. [2006]. This channel uses a 'nonspecific cation current' meaning it can transport any cation. In practice, this is modeled by giving it its own special ion NSC$^+$ with resting potential E_{NSC^+}.

$$\lambda_c = m, \quad g_c(\lambda_c) = m$$
$$E_{\text{NSC}^+} = -45.0$$
$$\frac{dm}{dt} = \alpha_m(v)(1-m) - \beta_m(v)m$$

$$\alpha_m(v) = \frac{0.001\cdot 6.43\cdot 11.9}{\text{exprel}((v+154.9)/11.9)}, \quad \beta_m(v) = 0.001\cdot 193\cdot\exp(v/33.1)$$

11.2.4 High-voltage-activated Ca²⁺ conductance

From Reuveni et al. [1993]

$$\lambda_c = (m,h), \quad g_c(\lambda_c) = m^2 h$$
$$\frac{dm}{dt} = \alpha_m(v)(1-m) - \beta_m(v)m$$
$$\frac{dh}{dt} = \alpha_h(v)(1-h) - \beta_h(v)h$$

$$\alpha_m(v) = \frac{0.055\cdot 3.8}{\text{exprel}(-(v+27)/3.8)}, \quad \beta_m(v) = 0.94\cdot\exp(-(v+75)/17)$$
$$\alpha_h(v) = 0.000457\cdot\exp(-(v+13)/50), \quad \beta_h(v) = \frac{0.0065}{\exp(-(v+15)/28)+1}$$

11.2.5 Low-voltage-activated Ca²⁺ conductance

From Avery and Johnston [1996], Randall and Tsien [1997].

$$\lambda_c = (m,h), \quad g_c(\lambda_c) = m^2 h$$
$$\frac{1}{q_t}\frac{dm}{dt} = \frac{m_\infty - m}{m_\tau}$$
$$\frac{1}{q_t}\frac{dh}{dt} = \frac{h_\infty - h}{h_\tau}$$
$$q_t = 2.3^{(T-21)/10}$$

$$m_\infty = \frac{1}{1+\exp(-(v+40)/6)}, \quad m_\tau = 5 + \frac{20}{1+\exp((v+35)/5)}$$
$$h_\infty = \frac{1}{1+\exp((v+90)/6.4)}, \quad h_\tau = 20 + \frac{50}{1+\exp((v+50)/7)}$$

11.2.6 M-type (Kv7) K$^+$ conductance

From Adams et al. [1982].

$$\lambda_c = m, \quad g_c(\lambda_c) = m$$
$$\frac{1}{q_t}\frac{dm}{dt} = \alpha_m(v)(1-m) - \beta_m(v)m$$
$$q_t = 2.3^{\left(\frac{T-21}{10}\right)}$$

$$\alpha_m(v) = 0.0033\exp(0.1(v+35)), \quad \beta_m(v) = 0.0033\cdot\exp(-0.1(v+35))$$

11.2.7 Kv3-like K$^+$ conductance

$$\lambda_c = m, \quad g_c(\lambda_c) = m$$
$$\frac{dm}{dt} = \frac{m_\infty - m}{m_\tau}$$

$$m_\infty = \frac{1}{1+\exp(-(v-18.7)/9.7)}, \quad m_\tau = \frac{4}{1+\exp(-(v+46.56)/44.14)}$$

11.2.8 Fast inactivating (transient, Kv4-like) K$^+$ conductance

From Korngreen and Sakmann [2000].

$$\lambda_c = (m,h), \quad g_c(\lambda_c) = m^4h$$
$$\frac{1}{q_t}\frac{dm}{dt} = \frac{m_\infty - m}{m_\tau}$$
$$\frac{1}{q_t}\frac{dh}{dt} = \frac{h_\infty - h}{h_\tau}$$
$$q_t = 2.3^{(T-21)/10}$$

$$m_\infty = \frac{1}{1+\exp(-(v+47)/29)}, \quad m_\tau = 0.34 + \frac{0.92}{\exp(((v+71)/59)^2)}$$
$$h_\infty = \frac{1}{1+\exp((v+66)/10)}, \quad h_\tau = 8 + \frac{49}{\exp(((v+73)/23)^2)}$$
$$\overline{g} = 1\times 10^{-5}$$

11.2.9 Slow inactivating (persistent) K$^+$ conductance

From Korngreen and Sakmann [2000].

$$\lambda_c = (m,h), \quad g_c(\lambda_c) = m^2h$$
$$\frac{1}{q_t}\frac{dm}{dt} = \frac{m_\infty - m}{m_\tau}$$
$$\frac{1}{q_t}\frac{dh}{dt} = \frac{h_\infty - h}{h_\tau}$$
$$q_t = 2.3^{(T-21)/10}$$

$$\begin{aligned}
m_\infty =& \frac{1}{1+\exp(-(v+14.3)/14.6)} \\
m_\tau =& \begin{cases} 1.25 + 175.03 \cdot e^{0.026v}, & \text{if } v < -50 \\ 1.25 + 13 \cdot e^{-0.026v}, & \text{if } v \geq -50 \end{cases} \\
h_\infty =& \frac{1}{1+\exp((v+54)/11)} \\
h_\tau =& \frac{24v+2690}{\exp(((v+75)/48)^2)} \\
& \overline{g} = 1 \times 10^{-5}
\end{aligned}$$

11.2.10 SK-type calcium-activated K⁺ conductance

From Köhler et al. [1996]. Note this is the only calcium-gated ion channel in the model.

$$\begin{aligned}
\lambda_c = z, \quad g_c(\lambda_c) = z \\
\frac{dz}{dt} = \frac{z_\infty - z}{z_\tau}
\end{aligned}$$

$$z_\infty = \frac{1}{1+(0.00043/[\text{Ca}^{2+}]_i)^{4.8}}, \quad z_\tau = 1$$

11.3 Training Details

Dataset The dataset used to fit the model was a subset of the stimulus/response pairs available from the Allen Institute. First, all stimuli and responses were downloaded for cell 480169178. Then, sections of length 200 milliseconds were extracted from a subset of the stimuli types. The stimuli types and sections were chosen so that the neuron was at rest and unstimulated at the beginning of the trace. We list the exclusion criteria below.

1. Any "Hold" stimuli: Excluded because these traces were collected under voltage clamp conditions which we did not model.
2. Test: Excluded because the stimulus is 0 mV for the entire trace.
3. Ramp/Ramp to Rheobase: Excluded because the cell is only at rest at the very beginning of the trace.
4. Short Square: 250 ms to 450 ms.
5. Short Square — Triple: 1250 ms to 1450 ms.
6. Noise 1 and Noise 2: 1250 ms to 1450 ms, 9250 ms to 9450 ms, 17250 ms to 17450 ms.
7. Long Square: 250 ms to 450 ms.
8. Square — 0.5ms Subthreshold: The entire trace.
9. Square — 2s Suprathreshold: 250 ms to 450 ms.
10. All others: Excluded.

For cell 480169178, the criteria above selected 95 stimulus/response pairs of 200 milliseconds each. Each trace pair was then downsampled to 1 ms (from the original 0.005 ms per step) and corrupted with mean-zero Gaussian noise of variance 20 mV^2 to simulate voltage imaging conditions. Finally, the 95 traces were randomly split into 72 training traces and 23 test traces.

Proposal and Twist The proposal and twist hyperparameters were broadly similar to the squid axon experiments, with the proposal being parameterized by a bidirectional RNN with a single hidden layer of size 64 and an MLP with a single hidden layer of size 64. The RNN was conditioned on the observed response and stimulus voltages at each timestep, and the MLP accepted the RNN hidden state, the previous latent state, and a transformer positional encoding of the number of steps since the last voltage response observation. The twist was similarly parameterized using an RNN run in reverse across the stimulus and response, combined with an MLP that accepted the RNN hidden state, the latent state being evaluated, and a transformer positional encoding of the number of steps elapsed since the last voltage response observation. The positional encodings were used to inform the twist and proposal of the number of steps elapsed since the last observation because the model was integrated with a stepsize of 0.1ms while observations were received once every millisecond.

Hyperparameter Sweeps To evaluate the methods we swept across the parameters

1. Initial observation variance: e^2, e^3, e^5
2. Initial voltage dynamics variance: e, e^2, e^3
3. Bias added to scales produced by the proposal: e^2, e^5

We also evaluated the models across three different data noise variances (20, 10, and 5) but the results were similar for all values, so we reported only the results for variance 20. This amounted to $3 \cdot 3 \cdot 3 \cdot 2$ different hyperparameter settings, and 5 seeds were run for each setting yielding a total of 270 runs.

When computing final performance, a hyperparameter setting was only evaluated if it had at least 3 runs that achieved 250,000 steps without NaN-ing out. For each hyperparameter setting selected for evaluation, all successful seeds were evaluated using early stopping on the train 4-particle log likelihood lower bound.

12 Strang Splitting for Hodgkin-Huxley Models

Because the Hodgkin-Huxley model is a *stiff* ODE, integrating it can be a challenge, especially at large step sizes. The traditional solution is to use an implicit integration scheme with varying step size, allowing the algorithm to take large steps when the voltage is not spiking. However, because our model adds noise to the ODE state at each timestep adaptive step-size methods are not viable as the different stepsizes would change the noise distribution.

Instead, we sought an explicit, fixed step-size method that could be stably integrated at relatively large stepsizes. Inspired by Chen et al. [2020], we developed a splitting approach that exploits the conditional linearity of the system. The system of ODEs describing the model can be split into two subsystems of linear first-order ODEs when conditioned on the state of the other subsystem. Specifically, the dynamics of the channel subunit states $\{\lambda_c \mid c \in \mathcal{C}\}$ is a system of linear first-order ODEs when conditioned on the voltage v and interior calcium concentration $[\mathrm{Ca}^{2+}]_i$. Similarly, the dynamics for v and $[\mathrm{Ca}^{2+}]_i$ is a system of linear first-order ODEs when conditioned on the subunit states.

Because the conditional dynamics of each subsystem are linear first-order ODEs, an exact solution to each subsystem is possible under the assumption that the states being conditioned on are constant for the duration of the step. Our integration approach uses these exact updates in an alternating fashion, first performing an exact update to the voltage and interior calcium concentration while holding the subunit states constant, and then performing an exact update to the subunit states while holding the voltage and interior calcium concentration constant. For details on Strang and other splitting methods applied to Hodgkin-Huxley type ODEs, see Chen et al. [2020].