This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TPAMI.2022.3153225, IEEE Transactions on Pattern Analysis and Machine Intelligence

Streaming Variational Monte Carlo

Yuan Zhao*, Josue Nassar*, Ian Jordan, Mónica Bugallo, and Il Memming Park

Abstract—Nonlinear state-space models are powerful tools to describe dynamical structures in complex time series. In a streaming setting where data are processed one sample at a time, simultaneous inference of the state and its nonlinear dynamics has posed significant challenges in practice. We develop a novel online learning framework, leveraging variational inference and sequential Monte Carlo, which enables flexible and accurate Bayesian joint filtering. Our method provides an approximation of the filtering posterior which can be made arbitrarily close to the true filtering distribution for a wide class of dynamics models and observation models. Specifically, the proposed framework can efficiently approximate a posterior over the dynamics using sparse Gaussian processes, allowing for an interpretable model of the latent dynamics. Constant time complexity per sample makes our approach amenable to online learning scenarios and suitable for real-time applications.

Index Terms—Nonlinear state-space modeling, online filtering, Bayesian machine learning

1 Introduction

Nonlinear state-space models are generative models for complex time series with underlying nonlinear dynamical structure [1], [2], [3]. Specifically, they represent nonlinear dynamics in the latent state-space, x_t, that capture the spatiotemporal structure of noisy observations, y_t:

$$x_t = f_\theta(x_{t-1}, u_t) + \epsilon_t \quad \text{(state dynamics model)} \quad (1a)$$

$$y_t \sim P(y_t | g_\psi(x_t)) \quad \text{(observation model)} \quad (1b)$$

where f_θ and g_ψ are continuous vector functions, P denotes a probability distribution, and ϵ_t is intended to capture unobserved perturbations of the state x_t. Such state-space models have many applications (e.g., object tracking) where the flow of the latent states is governed by known physical laws and constraints or where learning an interpretable model of the laws is of great interest, especially in neuroscience [4], [5], [6], [7], [8], [9]. If the parametric form of the model and the parameters are known a priori, then the latent states x_t can be inferred online through the filtering distribution, $p(x_t|\mathbf{y}_{1:t})$, or offline through the smoothing distribution, $p(\mathbf{x}_{1:t}|\mathbf{y}_{1:t})$ [10], [11]. Otherwise the challenge is in learning the parameters of the state-space model, $\{\theta, \psi\}$, which is known in the literature as the system identification problem.

In a streaming setting where data is processed one sample at a time, joint inference of the state and its nonlinear dynamics has posed significant challenges in practice. In this study, we are interested in online algorithms that can recursively solve the dual estimation problem of learning both the latent trajectory, $\mathbf{x}_{1:t}$, in the state-space and the parameters of the model, $\{\theta, \psi\}$, from streaming observations [12].

Popular solutions such, as the extended Kalman filter (EKF) or the unscented Kalman filter (UKF) [13], build an online dual estimator using nonlinear Kalman filtering by augmenting the state-space with its parameters [13], [14], [15], [16]. While powerful, they usually provide coarse approximations to the filtering distribution and involve many hyperparameters to be tuned which hinder their practical performance. Moreover, they do not take advantage of modern stochastic gradient optimization techniques commonly used throughout machine learning and are not easily applicable to arbitrary observation likelihoods.

Recursive stochastic variational inference has been proposed for streaming data assuming either independent [17] or temporally-dependent samples [6], [18], [19]. However the proposed variational distributions are not guaranteed to be good approximations to the true posterior. As opposed to variational inference, sequential Monte Carlo (SMC) leverages importance sampling to build an approximation to the target distribution in a data streaming setting [20], [21]. However, its success heavily depends on the choice of proposal distribution and the (locally) optimal proposal distribution usually is only available in the simplest cases [20]. While work has been done on learning good proposals for SMC [22], [23], [24], [25] most are designed only for offline scenarios targeting the smoothing distributions instead of the filtering distributions. In [22], the proposal is learned online but the class of dynamics for which this is applicable to is extremely limited.

In this paper, we propose a novel sequential Monte Carlo method for inferring a state-space model for the streaming time series scenario that adapts the proposal distribution on-the-fly by optimizing a surrogate lower bound to the log normalizer of the filtering distribution. Moreover, we choose the sparse Gaussian process (GP) [26] for modeling the unknown dynamics that allows for $\mathcal{O}(1)$ recursive Bayesian inference. Specifically our contributions are:

1) We prove that our approximation to the filtering distribution **converges to the true filtering distribution**.
2) Our objective function allows for **unbiased gradients** which lead to improved performance.
3) To the best of our knowledge, we are the first to use particles to represent the posterior of inducing variables of the sparse Gaussian processes, which

* *equal contribution*

- *All authors are with Stony Brook University, Stony Brook, NY, 11794. Y. Zhao and I. M. Park are with the Department of Neurobiology and Behavior. J. Nassar and M. Bugallo are with the Department of Electrical and Computer Engineering. I. Jordan is with the Department of Applied Mathematics and Statistics.*

allows for **accurate Bayesian inference** on the inducing variables rather than the typical variational approximation and closed-form weight updates.

4) Unlike many efficient filtering methods that usually assume Gaussian or continuous observations, our method allows **arbitrary observational distributions**.

2 Streaming Variational Monte Carlo

Given the state-space model defined in (1), the goal is to obtain the latent state, $x_t \in \mathbb{R}^{d_x}$, given a new observation, $y_t \in \mathcal{Y}$, where $\mathcal{Y}$ is a measurable space (typically $\mathcal{Y} = \mathbb{R}^{d_y}$ or $\mathcal{Y} = \mathbb{N}^{d_y}$). Under the Bayesian framework, this corresponds to computing the filtering posterior distribution at time t

$$p(x_t|\mathbf{y}_{1:t}) = \frac{p(y_t|x_t)}{p(y_t|\mathbf{y}_{1:t-1})} p(x_t|\mathbf{y}_{1:t-1}) \tag{2}$$

which recursively uses the previous filtering posterior distribution, $p(x_t|\mathbf{y}_{1:t-1}) = \int p(x_t|x_{t-1})p(x_{t-1}|\mathbf{y}_{1:t-1})dx_{t-1}$.

However, the above posterior is generally intractable except for limited cases [12] and thus we turn to approximate methods. Two popular approaches for approximating (2) are sequential Monte Carlo (SMC) [20] and variational inference (VI) [27], [28], [29]. In this work, we propose to combine sequential Monte Carlo and variational inference, which allows us to utilize modern stochastic optimization while leveraging the flexibility and theoretical guarantees of SMC. We refer to our approach as *streaming variational Monte Carlo* (SVMC). For clarity, we review SMC and VI in the follow sections.

2.1 Sequential Monte Carlo

SMC is a sampling based approach to approximate Bayesian inference that is designed to recursively approximate a sequence of distributions $p(\mathbf{x}_{0:t}|\mathbf{y}_{1:t})$ for $t = 1, \ldots$, using samples from a proposal distribution, $r(\mathbf{x}_{0:t}|\mathbf{y}_{1:t}; \boldsymbol{\lambda}_{0:t})$ where $\boldsymbol{\lambda}_{0:t}$ are the parameters of the proposal [20]. Due to the Markovian nature of the state-space model in (1), the smoothing distribution, $p(\mathbf{x}_{0:t}|\mathbf{y}_{1:t})$, can be expressed as

$$p(\mathbf{x}_{0:t}|\mathbf{y}_{1:t}) \propto p(x_0) \prod_{j=1}^{t} p(x_t|x_{t-1}) p(y_t|x_t). \tag{3}$$

We enforce the same factorization for the proposal, $r(\mathbf{x}_{0:t}|\mathbf{y}_{1:t}; \boldsymbol{\lambda}_{\mathbf{0:t}}) = r_0(x_0; \lambda_0) \prod_{j=1}^{t} r_j(x_j|x_{j-1}, y_j; \lambda_j)$.

A naive approach to approximating (3) is to use standard importance sampling (IS) [30]. N samples are sampled from the proposal distribution, $\mathbf{x}_{0:t}^1, \cdots, \mathbf{x}_{0:t}^N \sim r(\mathbf{x}_{0:t}; \boldsymbol{\lambda}_{0:t})$, and are given weights according to

$$w_{0:t}^i = \frac{p(x_0^i) \prod_{j=1}^{t} p(x_j^i|x_{j-1}^i) p(y_j|x_j^i)}{r_0(x_0^i; \lambda_0) \prod_{j=1}^{t} r_j(x_j^i|x_{j-1}^i, y_j; \lambda_j)}. \tag{4}$$

The importance weights can also be computed recursively

$$w_{0:t}^i = \prod_{s=0}^{t} w_s^i, \tag{5}$$

where

$$w_s^i = \frac{p(y_s|x_s^i) p(x_s^i|x_{s-1}^i)}{r_s(x_s^i|x_{s-1}^i, y_s; \lambda_s)}. \tag{6}$$

The samples and their corresponding weights, $\{(\mathbf{x}_{0:t}^i, w_{0:t}^i)\}_{i=1}^N$, are used to build an approximation to the target distribution

$$p(\mathbf{x}_{0:t}|\mathbf{y}_{1:t}) \approx \hat{p}(\mathbf{x}_{0:t}|\mathbf{y}_{1:t}) = \sum_{i=1}^{N} \frac{w_{0:t}^i}{\sum_\ell w_{0:t}^\ell} \delta_{\mathbf{x}_{0:t}^i} \tag{7}$$

where δ_x is the Dirac-delta function centered at x. While straightforward, naive IS suffers from the weight degeneracy issue; as the length of the time series, T, increases all but one of the importance weights will go to 0 [20].

To alleviate this issue, SMC leverages sampling-importance-resampling (SIR). Suppose at time $t-1$, we have the following approximation to the smoothing distribution

$$\hat{p}(\mathbf{x}_{0:t-1}|\mathbf{y}_{1:t-1}) = \frac{1}{N} \sum_{i=1}^{N} \frac{w_{t-1}^i}{\sum_\ell w_{t-1}^\ell} \delta_{\mathbf{x}_{0:t-1}^i}, \tag{8}$$

where w_{t-1}^i is computed according to (6). Given a new observation, y_t, SMC starts by resampling ancestor variables, $a_t^i \in \{1, \ldots, N\}$ with probability proportional to the importance weights, w_{t-1}^j. N samples are then drawn from the proposal, $x_t^i \sim r_t(x_t|x_{t-1}^{a_t^i}, y_t; \lambda_t)$, and their importance weights are computed, w_t^i, according to (6). The introduction of resampling allows for a (greedy) solution to the weight degeneracy problem. Particles with high weights are deemed good candidates and are propagated forward while the ones with low weights are discarded.

The updated approximation to $p(\mathbf{x}_{0:t}|\mathbf{y}_{1:t})$ is now

$$\hat{p}(\mathbf{x}_{0:t}|\mathbf{y}_{1:t}) = \frac{1}{N} \sum_{i=1}^{N} \frac{w_t^i}{\sum_\ell w_t^\ell} \delta_{\mathbf{x}_{0:t}^i}, \tag{9}$$

where $\mathbf{x}_{0:t}^i = (x_t^i, \mathbf{x}_{0:t-1}^{a_t^i})$. Marginalizing out $\mathbf{x}_{0:t-1}$ in (9) gives an approximation to the filtering distribution:

$$\begin{aligned} p(x_t|\mathbf{y}_{1:t}) &= \int p(\mathbf{x}_{0:t}|\mathbf{y}_{1:t}) \mathrm{d}\mathbf{x}_{0:t-1} \\ &\approx \int \sum_{i=1}^{N} \frac{w_t^i}{\sum_\ell w_t^\ell} \delta_{\mathbf{x}_{0:t}^i} \\ &= \sum_{i=1}^{N} \frac{w_t^i}{\sum_\ell w_t^\ell} \delta_{x_t^i}. \end{aligned} \tag{10}$$

As a byproduct, the weights produced in an SMC run yield an unbiased estimate of the marginal likelihood of the smoothing distribution [21]

$$\mathbb{E}[\hat{p}(\mathbf{y}_{1:t})] = \mathbb{E}\left[\prod_{s=1}^{t} \frac{1}{N} \sum_{i=1}^{N} w_s^i\right] = p(\mathbf{y}_{1:t}), \tag{11}$$

and a biased but *consistent* estimate of the marginal likelihood of the filtering distribution [21], [31]

$$\mathbb{E}[\hat{p}(y_t|\mathbf{y}_{1:t-1})] = \mathbb{E}\left[\frac{1}{N} \sum_{i=1}^{N} w_t^i\right]. \tag{12}$$

For completeness, we reproduce the consistency proof of (12) in section A of the appendix. The recursive nature of SMC makes it constant complexity per time step and constant memory because only the samples and weights generated at time t are needed, $\{w_t^i, x_t^i\}_{i=1}^N$, making them a perfect candidate to be used in an online setting [32]. These attractive

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TPAMI.2022.3153225, IEEE Transactions on Pattern Analysis and Machine Intelligence

properties have allowed SMC to enjoy much success in fields such as robotics [33], control engineering [34] and target tracking [35].

The success of an SMC sampler crucially depends on the design of the proposal distribution, $r_t(x_t|x_{t-1}, y_t; \lambda_t)$. A common choice for the proposal distribution is the transition distribution, $r_t(x_t|x_{t-1}, y_t; \lambda_t) = p(x_t|x_{t-1})$, which is known as the bootstrap particle filter (BPF) [36]. While simple, it is well known that the BPF needs a large number of particles to perform well and suffers in high-dimensions [37]. In addition, BPF requires the knowledge of $p(x_t|x_{t-1})$ which may not be known

Designing a proposal is even more difficult in an online setting because a proposal distribution that was optimized for the system at time t may not be the best proposal K steps ahead. For example, if the dynamics were to change abruptly, a phenomenon known as concept drift [38], the previous proposal may fail for the current time step. Thus, we propose to adapt the proposal distribution online using variational inference. This allows us to utilize modern stochastic optimization to adapt the proposal on-the-fly while still leveraging the theoretical guarantees of SMC.

2.2 Variational Inference

In contrast to SMC, VI takes an optimization approach to approximate Bayesian inference. In VI, we approximate the target posterior, $p(x_t|\mathbf{y}_{1:t})$, by a class of simpler distributions, $q(x_t; \vartheta_t)$, where ϑ_t are the parameters of the distribution. We then minimize a divergence (which is usually the Kullback-Leibler divergence (KLD)) between the posterior and the approximate distribution in the hopes of making $q(x_t; \vartheta_t)$ closer to $p(x_t|\mathbf{y}_{1:t})$. If the divergence used is KLD, then minimizing the KLD between these distributions is equivalent to maximizing the so-called evidence lower bound (ELBO) [29], [27]:

$$\begin{aligned}\mathcal{L}(\vartheta_t) &= \mathbb{E}_q[\log p(x_t, \mathbf{y}_{1:t}) - \log q(x_t; \vartheta_t)], \\ &= \mathbb{E}_q[\log \mathbb{E}_{p(x_{t-1}|\mathbf{y}_{1:t-1})}[p(x_t, x_{t-1}, \mathbf{y}_{1:t})] - \log q(x_t; \vartheta_t)].\end{aligned} \tag{13}$$

For filtering inference, the intractability introduced by marginalizing over $p(x_{t-1}|\mathbf{y}_{1:t-1})$ in (13) makes the problem much harder to optimize, rendering variational inference impractical in a streaming setting where incoming data are temporally dependent.

2.3 A Tight Lower Bound

Due to the intractability of the filtering distribution, the standard ELBO is difficult to optimize forcing us to define a different objective function. As stated above, we know that the sum of importance weights is an unbiased estimator of $p(\mathbf{y}_{1:t})$. Jensen's inequality applied to (11) [25], [39] gives,

$$\log p(\mathbf{y}_{1:t}) = \log \mathbb{E}[\hat{p}(\mathbf{y}_{1:t})] \geq \mathbb{E}[\log \hat{p}(\mathbf{y}_{1:t})]. \tag{14}$$

Expanding (14), we obtain

$$\begin{aligned}&\log p(y_t|\mathbf{y}_{1:t-1}) + \log p(\mathbf{y}_{1:t-1}) \\ &\qquad \geq \mathbb{E}[\log \hat{p}(y_t|\mathbf{y}_{1:t-1})] + \mathbb{E}[\log \hat{p}(\mathbf{y}_{1:t-1})],\end{aligned} \tag{15}$$

$$\log p(y_t|\mathbf{y}_{1:t-1}) \geq \mathbb{E}[\log \hat{p}(y_t|\mathbf{y}_{1:t-1})] - \mathcal{R}_t(N) \tag{16}$$

Algorithm 1 Streaming Variational Monte Carlo (Step t)

Require: $\{x_{t-1}^i, w_{t-1}^i\}_{i=1}^N, \Theta_{t-1}, y_t, \alpha$
for $k = 1, \ldots, N_{\text{SGD}}$ **do**
  for $i = 1, \ldots, L$ **do**
    $a_t^i \sim \Pr(a_t^i = j) \propto w_{t-1}^j$ ▷ *Resample*
    $x_t^i \sim r(x_t|x_{t-1}^{a_t^i}, y_t; \Theta_{t-1})$ ▷ *Propose*
    $w_t^i \leftarrow \frac{p(x_t^i|x_{t-1}^{a_t^i}; \Theta_{t-1}) p(y_t|x_{t-1}^{a_t^i}; \Theta_{t-1})}{r(x_t^i|x_{t-1}^{a_t^i}, y_t; \Theta_{t-1})}$ ▷ *Weigh*
  end for
  $\tilde{\mathcal{L}}_t \leftarrow \log(\sum_i w_t^i)$
  $\Theta_t \leftarrow \Theta_{t-1} + \alpha \nabla_\Theta \tilde{\mathcal{L}}_t$ ▷ SGD
end for
Resample, propose and reweigh N particles
return $\{x_t^i, w_t^i\}_{i=1}^N, \Theta_t$

where $\mathcal{R}_t(N) = \log p(\mathbf{y}_{1:t-1}) - \mathbb{E}[\log \hat{p}(\mathbf{y}_{1:t-1})] \geq 0$ is the variational gap. Leveraging this we propose to optimize

$$\begin{aligned}\widetilde{\mathcal{L}}_t(\Theta_t) &= \mathbb{E}[\log \hat{p}(y_t|\mathbf{y}_{1:t-1})] - \mathcal{R}_t(N), \\ &= \mathbb{E}\left[\log\left(\sum_{i=1}^N w_t^i\right)\right] - \mathcal{R}_t(N).\end{aligned} \tag{17}$$

We call $\widetilde{\mathcal{L}}_t$ the *filtering ELBO*; it is a lower bound to the log normalization constant (log partition function) of the filtering distribution where $\mathcal{R}_t(N)$ accounts for the bias of the estimator (12). As $\mathcal{R}_t(N)$ is **not** a function of Θ_t, it can be ignored when computing gradients.

There exists an implicit posterior distribution that arises from performing SMC given by [40],

$$\begin{aligned}\tilde{q}(x_t|\mathbf{y}_{1:t}) &= p(x_t, \mathbf{y}_{1:t}) \mathbb{E}\left[\frac{1}{\hat{p}(\mathbf{y}_{1:t})}\right], \\ &= p(x_t, y_t|\mathbf{y}_{1:t-1}) \mathbb{E}\left[\hat{p}(y_t|\mathbf{y}_{1:t-1})^{-1} \frac{p(\mathbf{y}_{1:t-1})}{\hat{p}(\mathbf{y}_{1:t-1})}\right].\end{aligned} \tag{18}$$

As the number of samples goes to infinity (17) can be made arbitrarily tight; as a result, the *implicit* approximation to the filtering distribution (18) will become arbitrarily close to the true posterior, $p(x_t|\mathbf{y}_{1:t})$, almost everywhere which allows for a trade-off between accuracy and computational complexity. We note that this result is not applicable in most cases of VI due to the simplicity of variational families used. We summarize this result in the following theorem (see the proof in section B of the appendix).

Theorem 2.1 (Filtering ELBO)**.** *The filtering ELBO* (17)*, $\widetilde{\mathcal{L}}_t$, is a lower bound to the logarithm of the normalization constant of the filtering distribution, $p(x_t|\mathbf{y}_{1:t})$. As the number of samples, N, goes to infinity, $\widetilde{\mathcal{L}}_t$ will become arbitrarily close to* $\log p(y_t|\mathbf{y}_{1:t-1})$.

Theorem 2.1 leads to the following corollary [41] (proof in section C of the appendix).

Corollary 2.1.1. *Theorem 2.1 implies that the implicit filtering distribution, $\tilde{q}(x_t|\mathbf{y}_{1:t})$, converges to the true posterior, $p(x_t \mid \mathbf{y}_{1:t})$, as $N \to \infty$.*

2.4 Stochastic Optimization

As in variational inference, we fit the parameters of the proposal, dynamics and observation model, $\Theta_t = \{\lambda_t, \theta_t, \psi_t\}$,

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TPAMI.2022.3153225, IEEE Transactions on Pattern Analysis and Machine Intelligence

by maximizing the (filtering) ELBO (Alg. 1). While the expectations in (17) are not in closed form, we can obtain unbiased estimates of $\widetilde{\mathcal{L}}_t$ and its gradients with Monte Carlo. Note that when obtaining gradients with respect to Θ_t, we only need to take gradients of $\mathbb{E}[\log \hat{p}(y_t|\mathbf{y}_{1:t-1})]$. We also assume that the proposal distribution, $r(x_t|x_{t-1}, y_t; \lambda_t)$, is reparameterizable, i.e. we can sample from $r(x_t|x_{t-1}, y_t; \lambda_t)$ by setting $x_t = h(x_{t-1}, y_t, \epsilon_t; \lambda_t)$ for some function h where $\epsilon_t \sim s(\epsilon_t)$ and s is a distribution independent of λ_t. Thus we can express the gradient of (17) using the reparameterization trick [42] as

$$\begin{aligned}\nabla_{\Theta_t}\widetilde{\mathcal{L}}_t &= \nabla_{\Theta_t}\mathbb{E}_{s(\epsilon^{1:L})}[\log \hat{p}(y_t|\mathbf{y}_{1:t-1})],\\ &= \mathbb{E}_{s(\epsilon^{1:L})}[\nabla_{\Theta_t}\log \hat{p}(y_t|\mathbf{y}_{1:t-1})],\\ &= \mathbb{E}_{s(\epsilon^{1:L})}\left[\nabla_{\Theta_t}\log\big(\sum_{i=1}^{L} w_t^i\big)\right].\end{aligned} \tag{19}$$

where $L \leq N$ is the number of subsamples to accelerate calculations. In Algorithm 1, we perform N_{SGD} stochastic gradient descent (SGD) updates for each step.

While using more samples, N, will reduce the variational gap between the filtering ELBO, $\widetilde{\mathcal{L}}_t$, and $\log p(y_t|\mathbf{y}_{1:t-1})$, using more samples, L, for estimating (19) may be detrimental for optimizing the parameters, as it has been shown to decrease the signal-to-noise ratio (SNR) of the gradient estimator for importance-sampling-based ELBOs [43]. The intuition is as follows: as the number of samples used to compute the gradient increases, the bound gets tighter and tighter which in turn causes the magnitude of the gradient to become smaller. The rate at which the magnitude decreases is much faster than the variance of the estimator, thus driving the SNR to 0. In practice, we found that using a small number of samples to estimate (19), $L < 5$, is enough to obtain good performance.

2.5 Learning Dynamics with Sparse Gaussian Processes

State-space models allow for various time series models to represent the evolution of state and ultimately predict the future [44]. While in some scenarios there exists prior knowledge on the functional form of the latent dynamics, $f_\theta(x)$, this is usually never the case in practice; thus $f_\theta(x)$ must be learned online as well. While one can assume a parametric form for $f_\theta(x)$, i.e. a recurrent neural network, and learn θ through SGD, this does not allow uncertainty over the dynamics to be expressed which is key for many real-time, safety-critical tasks. An attractive alternative over parametric models are Gaussian processes (GPs) [45]. Gaussian processes do not assume a functional form for the latent dynamics; rather, general assumptions, such as continuity or smoothness, are imposed. Gaussian processes also allow for a principled notion of uncertainty, which is key when predicting the future.

A Gaussian process is defined as a collection of random variables, any finite number of which have a joint Gaussian distribution. It is completely specified by its mean and covariance functions. A GP allows one to specify a prior distribution over functions

$$f(x) \sim \mathcal{GP}(m(x), k(x, x')) \tag{20}$$

where $m(\cdot)$ is the mean function and $k(\cdot,\cdot)$ is the covariance function; in this study, we assume that $m(x) = x$. With the GP prior imposed on the dynamics, one can do Bayesian inference with data.

With the current formulation, a GP can be incorporated by augmenting the state-space to (x_t, f_t), where $f_t \equiv f(x_{t-1})$. The importance weights are now computed according to

$$w_t = \frac{p(y_t|x_t)p(x_t|f_t)p(f_t|\mathbf{f}_{1:t-1}, \mathbf{x}_{0:t-1})}{r(x_t, f_t|f_{t-1}, x_{t-1}, y_t; \lambda_t)}. \tag{21}$$

Examining (21), it is evident that naively using a GP is impractical for online learning because its space and time complexity are proportional to the number of data points, which grows with time t, i.e., $\mathcal{O}(t^2)$ and $\mathcal{O}(t^3)$ respectively. In other words, the space and time costs increase as more and more observations are processed.

To overcome this limitation, we employ the sparse GP method [26], [46]. We introduce M inducing points, $\mathbf{z} = (z_1, \ldots, z_M)$, where $z_i = f(u_i)$ and u_i are pseudo-inputs and impose that the prior distribution over the inducing points is $p(\mathbf{z}) = \mathcal{N}(\mathbf{0}, k(\mathbf{u}, \mathbf{u}'))$. In the experiments, we spread the inducing points uniformly over a finite volume in the latent space. Under the sparse GP framework, we assume that $\mathbf{z}$ is a sufficient statistic for f_t, i.e.

$$\begin{aligned}p(f_t|\mathbf{x}_{0:t-1}, \mathbf{f}_{1:t-1}, \mathbf{z}) &= p(f_t|x_{t-1}, \mathbf{z})\\ &= \mathcal{N}\left(f_t|m_t + K_{tz}K_{zz}^{-1}\mathbf{z}, K_{tt} - K_{tz}K_{zz}^{-1}K_{zt}\right),\end{aligned} \tag{22}$$

where $m_t = m(x_{t-1})$. Note that the inclusion of the inducing points in the model reduces the computational complexity to be constant with respect to time. Marginalizing out f_t in (22)

$$\begin{aligned}&p(x_t|x_{t-1}, \mathbf{z})\\ &= \int p(x_t|f_t)p(f_t|x_{t-1}, \mathbf{z})df_t\\ &= \mathcal{N}\left(x_t|m_t + K_{tz}K_{zz}^{-1}\mathbf{z}, K_{tt} - K_{tz}K_{zz}^{-1}K_{zt} + Q\right).\end{aligned} \tag{23}$$

Equipped with equation (23), we can express the smoothing distribution as

$$p(\mathbf{x}_{0:t}, \mathbf{z}|\mathbf{y}_{1:t}) \propto p(x_0)p(\mathbf{z})\prod p(y_t|x_t)p(x_t|x_{t-1}, \mathbf{z}), \tag{24}$$

and the importance weights can be computed according to

$$w_t = \frac{p(y_t|x_t)p(x_t|x_{t-1}, \mathbf{z})p(\mathbf{z}|\mathbf{x}_{0:t-1})}{r(x_t, \mathbf{z}|x_{t-1}, y_t; \lambda_t)}. \tag{25}$$

Due to the conjugacy of the model, $p(\mathbf{z}|\mathbf{x}_{0:t-1})$ can be recursively updated efficiently. Let $p(\mathbf{z}_t|\mathbf{x}_{0:t-1}) = \mathcal{N}(\mathbf{z}_t|\mu_{t-1}, \Gamma_{t-1})$. Given x_t and by Bayes rule

$$p(\mathbf{z}|\mathbf{x}_{0:t}) \propto p(x_t|x_{t-1}, \mathbf{z})p(\mathbf{z}|\mathbf{x}_{0:t-1}), \tag{26}$$

we obtain the recursive updating rule:

$$\begin{aligned}\Gamma_t &= \left(\Gamma_{t-1}^{-1} + A_t^\top C_t^{-1}A_t\right)^{-1},\\ \mu_t &= \Gamma_t\left[\Gamma_{t-1}^{-1}\mu_{t-1} + A_t^\top C_t^{-1}(x_t - m_t)\right],\end{aligned} \tag{27}$$

where $A_t = K_{tz}K_{zz}^{-1}$ and $C_t = K_{tt} - K_{tz}K_{zz}^{-1}K_{zt} + Q$.

To facilitate learning in non-stationary environments, we impose a diffusion process over the inducing variables. Letting $p(\mathbf{z}_{t-1}|x_{0:t-1}) = \mathcal{N}(\mu_{t-1}, \Gamma_{t-1})$, we impose the following relationship between $\mathbf{z}_{t-1}$ and $\mathbf{z}_t$

$$\mathbf{z}_t = \mathbf{z}_{t-1} + \eta_t, \tag{28}$$

where $\eta_t \sim \mathcal{N}(0, \sigma_z^2 I)$. We can rewrite (25)

$$w_t = \frac{p(y_t|x_t)p(x_t|x_{t-1}, \mathbf{z}_t)p(\mathbf{z}_t|\mathbf{x}_{0:t-1})}{r(x_t, \mathbf{z}_t|x_{t-1}, \mathbf{z}_{t-1}, y_t; \lambda_t)}, \tag{29}$$

where

$$\begin{aligned} p(\mathbf{z}_t|\mathbf{x}_{0:t-1}) &= \int p(\mathbf{z}_t|\mathbf{z}_{t-1})p(\mathbf{z}_{t-1}|\mathbf{x}_{0:t-1})d\mathbf{z}_{t-1} \\ &= \mathcal{N}(\mu_{t-1}, \Gamma_{t-1} + \sigma_z^2 I). \end{aligned} \tag{30}$$

To lower the computation we marginalize out the inducing points from the model, simplifying (29)

$$w_t = \frac{p(y_t|x_t)p(x_t|\mathbf{x}_{0:t-1})}{r(x_t|x_{t-1}, y_t; \lambda_t)}, \tag{31}$$

where

$$\begin{aligned} p(x_t|\mathbf{x}_{0:t-1}) &= \int p(x_t|x_{t-1}, \mathbf{z}_t)p(\mathbf{z}_t|\mathbf{x}_{0:t-1})d\mathbf{z}_t \\ &= \mathcal{N}(v_t, \Sigma_t) \end{aligned} \tag{32}$$

where $v_t = m_t + A_t\mu_{t-1}$ and $\Sigma_t = C_t + A_t(\Gamma_{t-1} + \sigma_x^2 I)A_t^\top$.

For each stream of particles, we store μ_t^i and Γ_t^i. Due to the recursive updates (27), maintaining and updating μ_t^i and Γ_t^i is of constant time complexity, making it amenable for online learning. The use of particles also allows for easy sampling of the predictive distribution (details are in section E of the appendix). We call this approach SVMC-GP; the algorithm is summarized in Algorithm 2.

Algorithm 2 SVMC-GP (Step t)

Require: $\{x_{t-1}^i, \mu_{t-1}^i, \Gamma_{t-1}^i, w_{t-1}^i\}_{i=1}^N, \Theta_{t-1}, y_t, \alpha$
for $k = 1, \ldots, N_{\text{SGD}}$ **do**
  for $i = 1, \ldots, L$ **do**
    $a_t^i \sim \Pr(a_t^i = j) \propto w_{t-1}^j$ ▷ *Resample*
    $x_t^i \sim r(x_t|y_t, x_{t-1}^{a_t^i}; \mu_{t-1}^{a_t^i}, \Gamma_{t-1}^{a_t^i}, \Theta_{t-1})$ ▷ *Propose*
    $w_t^i \leftarrow \frac{p(x_t^i|x_{t-1}^{a_t^i})p(y_t|x_{t-1}^{a_t^i}; \Theta_{t-1})}{r(x_t^i|x_{t-1}^{a_t^i}, y_t; \mu_{t-1}^{a_t^i}, \Gamma_{t-1}^{a_t^i}, \Theta_{t-1})}$ ▷ *Reweigh*
  end for
  $\tilde{\mathcal{L}}_t \leftarrow \log(\sum_i w_t^i)$
  $\Theta_t \leftarrow \Theta_{t-1} + \alpha\nabla_\Theta \tilde{\mathcal{L}}_t$ ▷ SGD
end for
Resample, propose and reweigh N particles
Compute μ_t^i and Γ_t^i
return $\{x_t^i, \mu_t^i, \Gamma_t^i, w_t^i\}_{i=1}^N, \Theta_t$

2.6 Design of Proposals

As stated previously, the accuracy of SVMC depends crucially on the functional form of the proposal. The (locally) optimal proposal is

$$p(x_t|x_{t-1}, y_t) \propto p(y_t|x_t)p(x_t|x_{t-1}), \tag{33}$$

which is a function of y_t and f_t [47]. In general (33) is intractable; to emulate (33) we parameterize the proposal as

$$r(x_t|x_{t-1}, y_t) = \mathcal{N}(\mu_{\lambda_t}(f_t, y_t), \sigma_{\lambda_t}^2(f_t, y_t)I), \tag{34}$$

where μ_{λ_t} and σ_{λ_t} are neural networks with parameters λ_t.

3 Related Works

Much work has been done on learning good proposals for SMC. The method proposed in [24] iteratively adapts its proposal for an auxiliary particle filter. In [22], the proposal is learned online using expectation-maximization but the class of dynamics for which the approach is applicable for is extremely limited. The method proposed in [23] learns the proposal by minimizing the KLD between the smoothing distribution and the proposal, $\mathbb{D}_{\text{KL}}[p(\mathbf{x}_{0:t}|\mathbf{y}_{1:t})\|r(\mathbf{x}_{0:t}; \boldsymbol{\lambda}_{0:t})]$; while this approach can be used to learn the proposal online, biased importance-weighted estimates of the gradient are used which can suffer from high variance if the initial proposal is bad. Conversely, [25] maximizes $\mathbb{E}[\log \hat{p}(\mathbf{y}_{1:t})]$, which can be shown to minimize the KLD between the proposal and the implicit smoothing distribution, $\mathbb{D}_{\text{KL}}[q(\mathbf{x}_{0:t}|\mathbf{y}_{1:t})\|p(\mathbf{x}_{0:t}|\mathbf{y}_{1:t})]$; biased gradients were used to lower the variance. In contrast, SVMC allows for *unbiased* and *low variance* gradients that target the filtering distribution as opposed to the smoothing distribution. In [48], the proposal is parameterized by a Riemannian Hamiltonian Monte Carlo and the algorithm updates the mass matrix by maximizing $\mathbb{E}[\log \hat{p}(\mathbf{y}_{1:t})]$. At each time step (and for every stochastic gradient), the Hamiltonian must be computed forward in time using numerical integration, making the method impractical for an online setting.

Previous works have also tackled the dual problem of filtering and learning the parameters of the model online. A classic approach is to let the parameters of the generative model evolve according to a diffusion process, $\theta_t = \theta_{t-1} + \upsilon_t$: one can then create an augmented latent state, $\tilde{x}_t = [x_t, \theta_t]$, and perform filtering over $\tilde{x}_t$ either using particle filtering [49] or joint extended/unscented Kalman filtering [16], [15]. One can also use a separate filter for the latent state and the parameters, which is known as dual filtering [16], [15]. As SVMC is a general framework, we could also let the parameters of the generative model evolve according to a diffusion process and do joint/dual filtering; the availability of the filtering ELBO allows us to learn the variance of the diffusion online, while previous approaches treat this a fixed hyper-parameter. Besides, as we demonstrate in later experiments, we can learn the parameters of a parametric model online by performing SGD on the filtering ELBO. In [50], they combine extended Kalman filtering (EKF) with Gaussian processes for dual estimation; the use of EKF involves approximations and restricts the observation models one can apply it on. Moreover, the use of a full Gaussian process–as opposed to a sparse alternative–prevents it from being deployed for long time series. In [2], particle filtering is combined with a sparse Gaussian process to learn the latent dynamics and the emission model online; while similar to SVMC-GP, there are important differences between the two works. Firstly–and most importantly–the latent fixed dynamics are **not** learned online in [2]; training data is collected a priori and used to pre-train the GP and is kept during the filtering process. While a priori training data can also be used for SVMC-GP, our formulation allows us to continuously learn the latent dynamics in a purely online fashion. Second, a fixed proposal–similar to the one found in bootstrap particle filtering–is used while SVMC-GP allows for the proposal to adapt on-the-fly. In [19], they

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TPAMI.2022.3153225, IEEE Transactions on Pattern Analysis and Machine Intelligence

tackle the problem of dual estimation by leveraging the recursive form of the smoothing distribution to obtain an ELBO that can be easily computed online, allowing for the parameters of the generative model to be inferred using SGD. While similar to SVMC, we note that their approach relies on simple parametric variational approximations which are not as expressive as the particle based ones used in SVMC.

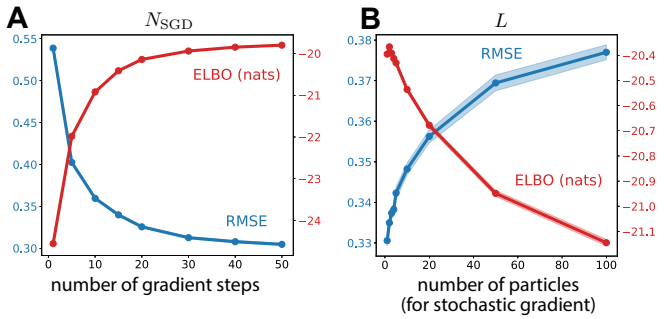

Figure 1. Investigating how the performance of SVMC–measured via RMSE (lower is better) and ELBO (higher is better)–depends on number of gradient steps, N_{SGD} and number of particles used to compute stochastic gradient, L. For each setting, we run 100 realizations of SVMC on the chaotic RNN data from sec. 4.1.2. Solids lines are the average where error bars are the standard error. A) For a fixed number of particles used to compute stochastic gradient, $L = 4$, the number of gradient steps, N_{SGD} taken at every time step is varied. B) For a fixed number of gradient steps, $N_{\text{SGD}} = 15$, the number of particles used to compute stochastic gradient, L, is varied.

4 Experiments

To showcase the power of SVMC, we employ it on a number of simulated and real experiments. For all experiments, the Adam optimizer was used [52].

4.1 Synthetic Data

4.1.1 *Linear Dynamical System*

As a baseline, we apply SVMC on data generated from a linear dynamical system (LDS)

$$\begin{aligned} x_t &= Ax_{t-1} + \epsilon_t, \quad \epsilon_t \sim \mathcal{N}(0, Q), \\ y_t &= Cx_t + \xi_t, \qquad \xi_t \sim \mathcal{N}(0, R). \end{aligned} \tag{35}$$

LDS is the *de facto* dynamical system for many fields of science and engineering due to its simplicity and efficient exact filtering (i.e., Kalman filtering). The use of an LDS also endows a closed form expression of the log marginal likelihood for the filtering and smoothing distribution. Thus, as a baseline we compare the estimates of the negative log marginal likelihood, $-\log p(\mathbf{y}_{1:T})$, produced by SVMC, variational sequential Monte Carlo (VSMC) [25] (which is an offline method) and BPF [36] in an experiment similar to the one used in [25]. We generated data from (35) with $T = 50$, $d_x = d_y = 10$, $(A)_{ij} = \alpha^{|i-j|+1}$, with $\alpha = 0.42$ and $Q = R = I$ where the state and emission parameters are fixed; the true negative log marginal likelihood is 1168.2. For SVMC and VSMC, we used the same proposal parameterization as [25]

$$r(x_t|x_{t-1}; \lambda_t) = \mathcal{N}(\mu_t + \text{diag}(\beta_t)Ax_{t-1}, \text{diag}(\sigma_t^2)), \tag{36}$$

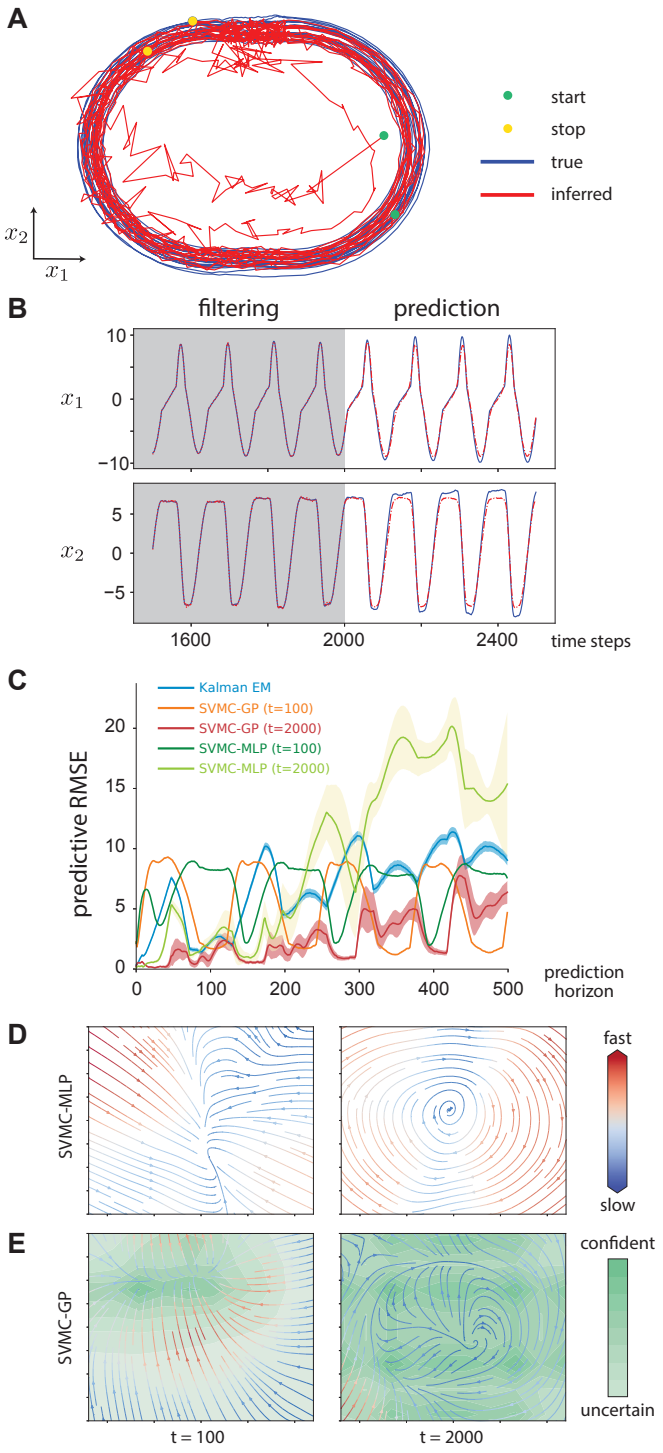

Figure 2. NASCAR® Dynamics [51]. (A) True and inferred latent trajectory using SVMC-GP. (B) Filtering and prediction. We show the last 500 steps of filtered states and the following 500 steps of predicted states. (C) Forecasting error. We compare the 500-step predictive MSE (averaged over 100 realizations) of SVMC-GP, SVMC-MLP, and Kalman filter. The transition matrix of the Kalman filter was learned by EM (offline). The periodic tendency is due to the periodic nature of ground truth. (D)–(E) Inferred dynamics as velocity field. For SVMC-GP, posterior variance of dynamics is additionally shown as uncertainty.

where $\lambda_t = \{\mu_t, \beta_t, \sigma_t^2\}$. To ensure VSMC has enough time to converge, we use 25,000 gradient steps. To equate

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TPAMI.2022.3153225, IEEE Transactions on Pattern Analysis and Machine Intelligence

the total number of gradient steps between VSMC and SVMC, 25,000/50 = 500 gradient steps were done at each time step for SVMC. For both methods, a learning rate of 0.01 was used where $L = 4$ particles were used for computing gradients, which was used in [25]. To equate the computational complexity between SVMC and BPF, we ran the BPF with 125,000 particles. We fixed the data generated from (35) and ran each method for 100 realizations; the average negative ELBO and its standard error of each the methods are shown in Table 1. To investigate the dependence of the ELBO on the number of particles, we demonstrate results for SVMC and VSMC using a varying number of particles.

From Table (1), we see that SVMC performs better than VSMC for all number of particles considered. While SVMC with 100 particles is outperformed by BPF, SVMC with 1,000 particles matches the performance of BPF with a smaller computational budget.

4.1.2 Chaotic Recurrent Neural Network

To show the performance of our algorithm in filtering data generated from a complex, nonlinear and high-dimensional dynamical system, we generate data from a continuous-time "vanilla" recurrent neural network (vRNN)

$$\tau \dot{x}(t) = -x(t) + \gamma W_x \tanh(x(t)) + \sigma(x) dW(t). \quad (37)$$

where $W(t)$ is Brownian motion. Using Euler integration, (37) can be described as a discrete time dynamical system

$$x_{t+1} = x_t + \Delta(-x_t + \gamma W_x \tanh(x_t))/\tau + \epsilon_t, \quad \epsilon_t \sim \mathcal{N}(0, Q) \quad (38)$$

where Δ is the Euler step. The emission model is

$$y_t = Cx_t + D + \xi_t, \quad \xi_{t,1}, \cdots, \xi_{t,d_y} \overset{\text{i.i.d}}{\sim} \mathcal{ST}(0, \nu_y, \sigma_y) \quad (39)$$

where each dimension of the emission noise, v_t, is independently and identically distributed (i.i.d) from a Student's t distribution, $\mathcal{ST}(0, \nu_y, \sigma_y)$, where ν_y is the degrees of freedom and σ_y is the scale.

We set $d_y = d_x = 10$ and the elements of W_x are i.i.d. drawn from $\mathcal{N}(0, 1/d_x)$. We set $\gamma = 2.5$ which produces chaotic dynamics at a relatively slow timescale compared to τ [53]. The rest of the parameters values are: $\tau = 0.025$, $\delta = 0.001$, $Q = 0.01I$, $\nu_y = 2$ and $\sigma_y = 0.1$, which are kept fixed. We generated a single time series of length of $T = 500$ and fixed it across multiple realizations. SVMC was ran using 200 particles with proposal distribution (34), where the neural network was a multi-layer perceptron (MLP) with 1 hidden layer of width 100 and relu nonlinearities; 15 gradient steps were performed at each time step with a learning rate of .001 with $L = 4$. For a comparison, a BPF with 10,000 particles and an unscented Kalman filter (UKF) was run. Each method was ran over 100 realizations. We compare the ELBO and root mean square error (RMSE) between the true latent states, $x_{1:T}$, and the inferred latent states, $\hat{x}_{1:T}$.[1] With a similar computational budget, SVMC can achieve better performance than a BPF using almost two orders of magnitude less samples. To investigate the effect the number of gradient steps has on the performance of SVMC, we plot the RMSE and ELBO as a function of number of gradient

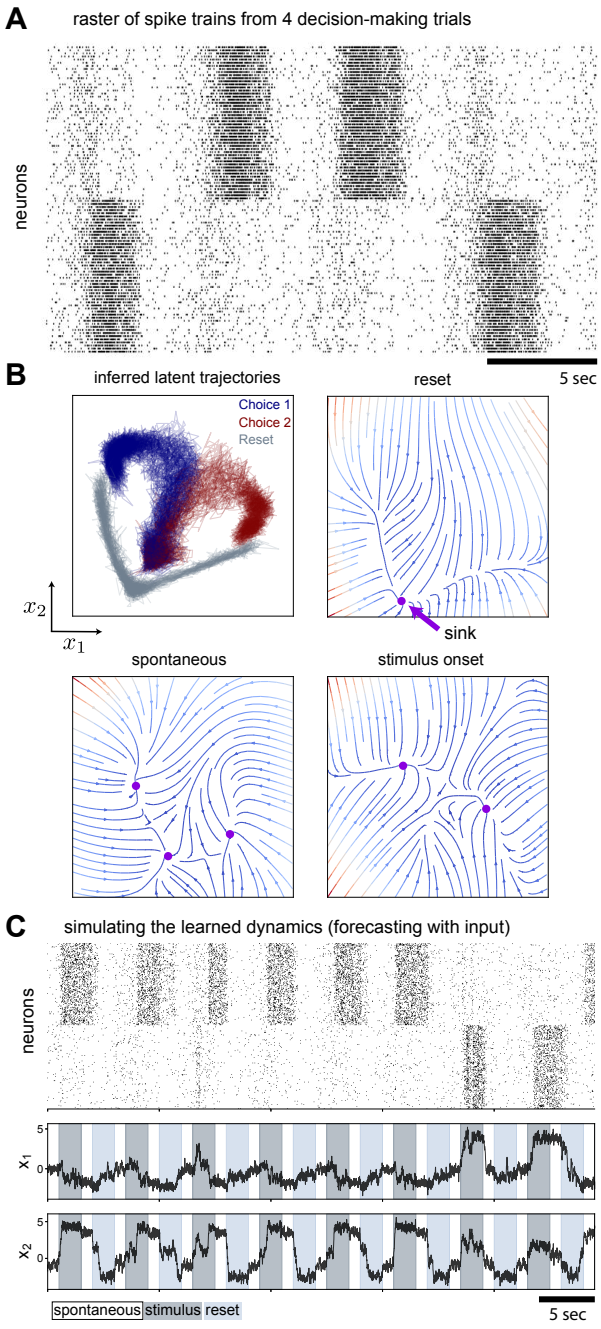

Figure 3. Winner-Take-All Spiking Neural Network. (A) 4 trials of training data. The neuronal activity was drawn over a 25 sec time window. Each row represents one neuron. Each dot represents that neuron fired within that time bin. (B) Inference. The top-left panel shows the inferred latent trajectories of several trials. In each trial the network started at the initial state, eventually reached either choice (indicated by the color) after the stimulus onset, and finally went back around the initial state after receiving reset signal. The rest three panels show the phase portraits of inferred dynamical system revealing the bifurcation and how the network dynamics were governed at different phases of the experiment. At the spontaneous phase (when the network receive no external input), the latent state is attracted by the middle sink. After the stimulus is onset, the middle sink disappears and the latent state falls into either side driven by noise to form a choice. When the reset is onset, the latent state is pushed back to the only sink that is close to the middle sink of the spontaneous phase, and then the network is ready for a new trial. (C) Simulation from the fitted model. We generated latent trajectory and spike train by replicating the experiments on the fitted model. The result shows that the model can replicate the dynamics of the target network.

1. We use the posterior mean as our estimate of the latent states.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TPAMI.2022.3153225, IEEE Transactions on Pattern Analysis and Machine Intelligence

Table 1
Experiment 1 (LDS) with 100 replication runs (true negative log-likelihood is 1168.12). The average negative ELBO and runtime are shown with the standard error for SVMC, VSMC and BPF where the number in parenthesis is the number of particles used.

	SVMC (100)	VSMC (100)	SVMC (1,000)	VSMC (1,000)	SVMC (10,000)	VSMC (10,000)	BPF (125,000)
−ELBO	1188.3 ± 0.5	1195.9 ± 0.5	1178.3 ± 0.3	1183.6 ± 0.3	**1173.8** ± 0.2	1179.8 ± 0.2	1177.0 ± 0.2
time (s)	47.5 ± 0.5	6390.2 ± 3.1	51.6 ± 0.3	6390.2 ± 3.1	64.5 ± 0.5	6390.2 ± 3.1	95.0 ± 0.7

Table 2
Experiment 2 (Chaotic RNN) with 100 replication runs. The average RMSE (lower is better), negative ELBO (lower is better) and runtime per step are shown with standard error.

	SVMC (200)	BPF (10,000)	UKF
RMSE	**.34 ± .001**	0.4 ± 0.002	3.9 ± 0.12
−ELBO (nats)	**20.42 ± .008**	24.16 ± 0.018	N/A
time (s)	18.78 ± 0.08	15.83 ± 0.09	0.8 ± 0.004

steps taken in figure 1A; taking more gradient steps leads to a decrease in RMSE and an increase in the ELBO. We next investigate the effect the number of samples used to compute the stochastic gradient, L, has on the performance 1B; as was demonstrated in [43], larger L leads to a decrease in performance.

4.1.3 Synthetic NASCAR® Dynamics

We test learning dynamics online with sparse GP on a synthetic data of which the underlying dynamics follow a recurrent switching linear dynamical systems [51]. The simulated trajectory resembles the NASCAR® track (Fig. 2A). We train the model with $2,000$ observations simulated from $y_t = Cx_t + \xi_t$ where C is a 50-by-2 matrix. The proposal is defined as $\mathcal{N}(\mu_t, \Sigma_t)$ of which μ_t and Σ_t are linear maps of the concatenation of observation y_t and previous state x^i_{t-1}. We use 50 particles, squared exponential (SE) kernel and 20 inducing points for GP and 1E-4 learning rate. We also run a SVMC (with the same setting on particles and learning rate as the former) with MLP (1 hidden layer and 20 hidden units) dynamics for comparison. GP dynamics not only estimate the velocity field but also give the uncertainty over the estimate while MLP dynamics is only a point estimate. To investigate the learning of dynamics, we control for other factors, i.e. we fix the observation model and other hyper-parameters such as noise variances at their true values. (See the details in section D of the appendix.)

Figure 2A shows the true (blue) and inferred (red) latent states. The inference quickly catches up with the true state and stays on the track. As the state oscillates on the track, the sparse GP learns a reasonable limit cycle (Fig. 2F) without seeing much data outside the track. The velocity fields in Figure 2D–F show the gradual improvement in the online learning. The 500-step prediction also shows that the GP captures the dynamics (Fig. 2B). We compare SVMC with Kalman filter in terms of mean squared error (MSE) (Fig. 2C). The transition matrix of the LDS of the Kalman filter (KF) is learned by expectation-maximization which is an offline method, hence not truly online. Yet, SVMC performs better than KF after 1000 steps.

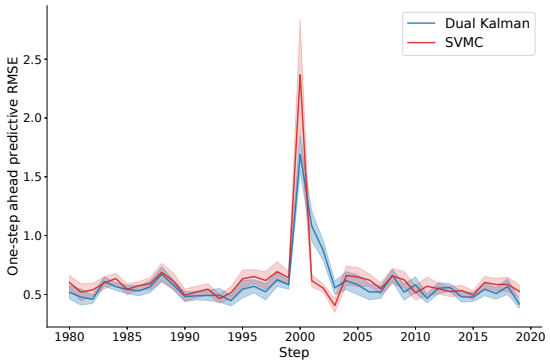

Figure 4. Prediction of nonstationary dynamical system. The colored curves (blue: EKF, red: SVMC) are the RMSEs (solid line: mean, shade: stderr) of one-step-ahead prediction of nonstationary system during online learning (50 trials each run, 10 runs). The linear system was changed and the state was perturbed at the 2000th step (center). Both online algorithms quickly learned the change after a few steps.

4.1.4 Winner-Take-All Spiking Neural Network

The perceptual decision-making paradigm is a well-established cognitive task where typically a low-dimensional decision variable needs to be integrated over time, and subjects are close to optimal in their performance. To understand how the brain implements such neural computation, many competing theories have been proposed [54], [55], [56], [57], [58]. We test our method on a simulated biophysically realistic cortical network model for a visual discrimination experiment [57]. In the model, there are two excitatory sub-populations that are wired with slow recurrent excitation and feedback inhibition to produce attractor dynamics with two stable fixed points. Each fixed point represents the final perceptual decision, and the network dynamics amplify the difference between conflicting inputs and eventually generates a binary choice.

The simulated data was organized into decision-making trials. We modified the network by injecting a 60 Hz Poisson input into the inhibitory sub-population at the end of each trial to "reset" the network for the purpose of uninterrupted trials to fit the streaming case because the original network was designed for only one-time experiment. In each trial the input to the network consisted of two periods, one 2-sec stimulus signal with different strength of visual evidence controlled by "coherence", and one 2-sec 60 Hz reset signal, each follows a 1-sec spontaneous period (no input). We subsampled 480 selective neurons out of 1600 excitatory neurons from the simulation to be observed by our algorithm.

Fig. 3 shows that SVMC (300 particles) with sparse GP dynamics (150 inducing points, squared exponential kernel) and MLP proposal (1 hidden layer, 1000 hidden units) with

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TPAMI.2022.3153225, IEEE Transactions on Pattern Analysis and Machine Intelligence

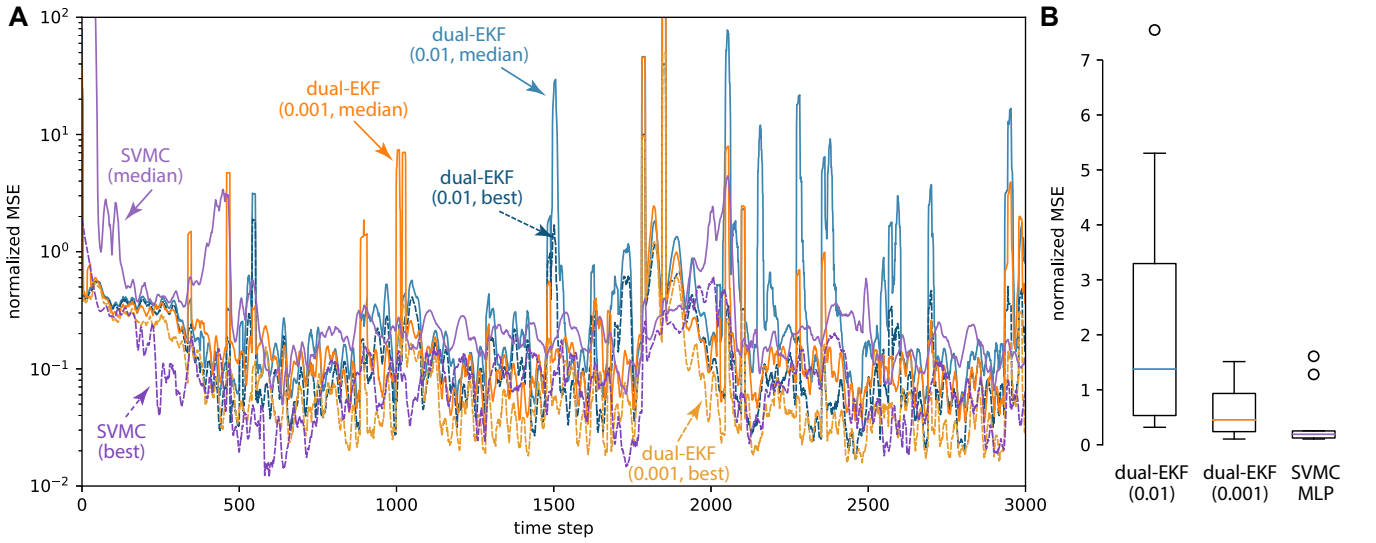

Figure 5. Prediction performance on 3D data generated from an analog stable oscillator circuit. We compare Dual-EKF and SVMC both with dynamics parameterized with MLP (2-20-2). (A) Normalized MSE of 100 time step prediction using the filtered system. Median and best out of 11 randomly initialized filters are shown. To estimate the normalized MSE, 11 realizations were used, and for ease of visual parsing 11 bin moving window averaging was applied. (B) Comparison of normalized MSE of the last 500 time steps.

$L = 2$, learning rate 1E-4 and 15 gradient steps, did well at learning the dynamics of target network. The inferred latent trajectories of several trials (Fig. 3B). In each trial the network started at the initial state, eventually reached either choice (indicated by the color) after the stimulus onset, and finally went back around the initial state after receiving reset signal. The other three panels (Fig. 3B) show the phase portraits of the inferred dynamical system revealing the bifurcation and how the network dynamics are governed during different phases of the experiment. In the spontaneous phase (when the network receives no external input), the latent state is attracted by the middle sink. After the stimulus is onset, the middle sink disappears and the latent state falls into either side driven by noise to form a choice. When the reset is onset, the latent state is pushed back to the only sink that is close to the middle sink of the spontaneous phase, and then the network is ready for a new trial. We generated a latent trajectory and corresponding spike train by replicating the experiments on the fitted model (Fig. 3C). The result shows that the model can replicate the dynamics of the target network.

The mean-field reduction of the network (Fig. 6 [58]) also confirms that our inference accurately captured the key features of the network. Note that our inference was done without knowing the true underlying dynamics which means our method can recover the dynamics from data as a bottom-up approach.

4.1.5 Nonstationary system

Another feature of our method is that its state dynamics estimate never stops. As a result, the algorithm is adaptive, and can potentially track slowly varying (nonstationary) latent dynamics. To test adaptation to perturbation to both the state and system, we compared a dual EKF and the proposed approach (50 particles, GP dynamics with 10 inducing points and squared exponential kernel, linear proposal, 1E-4 learning rate) on a 2D nonstationary linear dynamical system. A spiral-in linear system was suddenly changed from clockwise to counter-clockwise at the 2000th step and the latent state was perturbed (Fig. 4). To adapt EKF, we used Gaussian observations that were generated through linear map from 2-dimensional state to 200-dimensional observation with additive noise ($\mathcal{N}(0, 0.5)$). To focus on the dynamics, we fixed all the parameters except the transition matrix for both methods, while our approach still has to learn the recognition model in addition. Our method quickly adapts in a few steps.

4.2 Real Data: Learning an Analog Circuit

It has been verified that the proposed methodology is capable of learning the underlying dynamics from noisy streaming observations in the above synthetic experiments. To test it in real world, we applied our method to the voltage readout from an analog circuit [59]. We designed and built this circuit to realize a system of ordinary differential equations as follows

$$\begin{aligned} \dot{x} &= (5z - 5)[x - \tanh(\alpha x - \beta y)] \\ \dot{y} &= (5z - 5)[y - \tanh(\beta x + \alpha y)] \\ \dot{z} &= -0.5(z - \tanh(1.5z)) \end{aligned} \tag{40}$$

where $\cdot$ indicates the time derivative and $\alpha = \beta = 1.5\cos(\frac{\pi}{5})$. This circuit performed oscillation with a period of approximately 2 Hz. The sampling rate was 2000 Hz.

We assume the following model:

$$x_t = f(x_{t-1}) + \epsilon_t, \tag{41}$$

$$y_t = Cx + d + \psi_t, \tag{42}$$

where $x_t \in \mathbb{R}^2$, $y_t \in \mathbb{R}^3$, $\epsilon_t \sim \mathcal{N}(0, 10^{-3})$ and $\xi_t \sim \mathcal{N}(0, 10^{-3})$. We parameterize $f(\cdot)$ using an MLP (1 hidden layer, 20 hidden units) and perform dual estimation using SVMC and dual EKF on 3,000 time steps (Fig. 5A). We

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TPAMI.2022.3153225, IEEE Transactions on Pattern Analysis and Machine Intelligence

chose two different levels of diffusion $(0.001, 0.0001)$ on the parameters for dual EKF to implement different learning rates. We forecast 10 realizations of 100 steps ahead every filtering step and show the mean and standard deviation of the logarithm of MSE to the true observation (Fig.). As dual EKF has trouble learning the parameters of the observation model, we fixed C and d for dual EKF while we let SVMC (500 particles, lr 1E-4 and 15 gradient steps) learn both the parameters of the latent dynamics, C and d. Figure shows SVMC achieve the same level of performance but of less variance, and the slow convergence in the beginning was due to learning more parameters. The inferred dynamics shows that the limit cycle can implement the oscillation (Fig. 5B). The prediction of future observations (500 steps) resemble the oscillation and the true observation is covered by 100 repeated predictions (Fig. 5C). The predictions started at the final state of the training data, and we simulated the future observation trajectory from the trained model without seeing any new data. We repeated the procedure of prediction 100 times. Figure. 5D shows the normalized predictive MSE (relative to the mean observation over time). The solid line is the mean normalized MSE and the shade is the standard error. Since the simulation included the state noise, the prediction diverged from the true observations as time goes.

5 Discussion

In this study we developed a novel online learning framework, leveraging variational inference and sequential Monte Carlo, which enables flexible and accurate Bayesian joint filtering. Our derivation shows that our filtering posterior can be made arbitrarily close to the true one for a wide class of dynamics models and observation models. Specifically, the proposed framework can efficiently infer a posterior over the dynamics using sparse Gaussian processes by augmenting the state with the inducing variables that follow a diffusion process. Taking benefit from Bayes' rule, our recursive proposal on the inducing variables does not require optimization with gradients. Constant time complexity per sample makes our approach amenable to online learning scenarios and suitable for real-time applications. In contrast to previous works, we demonstrate our approach is able to accurately filter the latent states for linear / nonlinear dynamics, recover complex posteriors, faithfully infer dynamics, and provide long-term predictions. In future, we want to focus on reducing the computation time per sample that could allow for real-time application on faster systems. On the side of GP, we would like to investigate the priors and kernels that characterize the properties of dynamical systems as well as the hyperparameters.

Appendix A
Proof that $\hat{p}(y_t|\mathbf{y}_{1:t-1})$ is a consistent estimator for $p(y_t|\mathbf{y}_{1:t-1})$

Proof. To prove that $\hat{p}(y_t|\mathbf{y}_{1:t-1})$ is a consistent estimator, we will rely on the delta method [31]. From [60], we know that the central limit theorem (CLT) holds for $\hat{p}(\mathbf{y}_{1:t})$ and $\hat{p}(\mathbf{y}_{1:t-1})$

$$\sqrt{N}(\hat{p}(\mathbf{y}_{1:t-1}) - p(\mathbf{y}_{1:t-1})) \xrightarrow{d} \mathcal{N}(0, \sigma_{t-1}^2), \tag{43}$$

$$\sqrt{N}(\hat{p}(\mathbf{y}_{1:t}) - p(\mathbf{y}_{1:t})) \xrightarrow{d} \mathcal{N}(0, \sigma_{t}^2) \tag{44}$$

where we assume that σ_{t-1}^2 and σ_t^2 are finite. We can express $\hat{p}(y_t|\mathbf{y}_{1:t-1})$ as a function of $\hat{p}(\mathbf{y}_{1:t})$ and $\hat{p}(\mathbf{y}_{1:t-1})$,

$$\hat{p}(y_t|\mathbf{y}_{1:t-1}) = g(\hat{p}(\mathbf{y}_{1:t}), \hat{p}(\mathbf{y}_{1:t-1})) = \frac{\hat{p}(\mathbf{y}_{1:t})}{\hat{p}(\mathbf{y}_{1:t-1})}. \tag{45}$$

Since $\frac{p(\mathbf{y}_{1:t})}{p(\mathbf{y}_{1:t-1})} = p(y_t|\mathbf{y}_{1:t-1})$ and g is a continuous function, an application of the Delta method gives

$$\sqrt{N}(\hat{p}(y_t|\mathbf{y}_{1:t-1}) - p(y_t|\mathbf{y}_{1:t-1})) \xrightarrow{d} \mathcal{N}(0, \nabla g^\top \Sigma \nabla g), \tag{46}$$

where $\Sigma_{1,1} = \sigma_t^2$, $\Sigma_{2,2} = \sigma_{t-1}^2$ and $\Sigma_{1,2} = \Sigma_{2,1} = \sigma_{t,t-1}$ where by the Cauchy-Schwartz inequality, $\sigma_{t,t-1}$ is also finite [31]. Thus, as $N \to \infty$, $\hat{p}(y_t|\mathbf{y}_{1:t-1})$ will converge in probability to $p(y_t|\mathbf{y}_{1:t-1})$, proving the consistency of the estimator. □

Appendix B
Proof of Theorem 2.1

Proof. It is well known that the importance weights produced in a run of SMC are an unbiased estimator of $p(\mathbf{y}_{1:t})$ [21]

$$\mathbb{E}[\hat{p}(\mathbf{y}_{1:t})] = p(\mathbf{y}_{1:t}) \tag{47}$$

where $\hat{p}(\mathbf{y}_{1:t}) = \prod_{j=1}^{t} \frac{1}{N} \sum_{i=1}^{N} w_j^i$. We can apply Jensen's inequality to obtain

$$\log p(\mathbf{y}_{1:t}) \geq \mathbb{E}[\log \hat{p}(\mathbf{y}_{1:t})]. \tag{48}$$

Expanding both sides of (48)

$$\begin{aligned} &\log p(y_t|\mathbf{y}_{1:t-1}) + \log p(\mathbf{y}_{1:t-1}) \\ &\quad \geq \mathbb{E}[\log \hat{p}(y_t|\mathbf{y}_{1:t-1})] + \mathbb{E}[\log \hat{p}(\mathbf{y}_{1:t-1})]. \end{aligned} \tag{49}$$

Subtracting $\log p(\mathbf{y}_{1:t-1})$ from both sides gives

$$\begin{aligned} &\log p(y_t|\mathbf{y}_{1:t-1}) \\ &\quad \geq \mathbb{E}[\log \hat{p}(y_t|\mathbf{y}_{1:t-1})] + \mathbb{E}[\log \hat{p}(\mathbf{y}_{1:t-1})] - \log p(\mathbf{y}_{1:t-1}). \end{aligned} \tag{50}$$

Letting $\mathcal{R}_t(N) = \log p(\mathbf{y}_{1:t-1}) - \mathbb{E}[\log \hat{p}(\mathbf{y}_{1:t-1})]$, where N is the number of samples, we get

$$\log p(y_t|\mathbf{y}_{1:t-1}) \geq \mathbb{E}[\log \hat{p}(y_t|\mathbf{y}_{1:t-1})] - \mathcal{R}_t(N), \tag{51}$$

where by Jensen's inequality (48), $\mathcal{R}_t(N) \geq 0$ for all values of N. By the continuous mapping theorem [31],

$$\lim_{N\to\infty} \mathbb{E}[\log \hat{p}(\mathbf{y}_{1:t-1})] = \log p(\mathbf{y}_{1:t-1}). \tag{52}$$

As a consequence, $\lim_{N\to\infty} \mathbb{E}[\mathcal{R}_t(N)] = 0$. By the same logic, and leveraging that $\hat{p}(y_t|\mathbf{y}_{1:t-1})$ is a consistent estimator for $p(y_t|\mathbf{y}_{1:t-1})$, we get that

$$\lim_{N\to\infty} \mathbb{E}[\log \hat{p}(y_t|\mathbf{y}_{1:t-1})] = \log p(y_t|\mathbf{y}_{1:t-1}). \tag{53}$$

Thus $\mathcal{L}_t$ will get arbitrarily close to $\log p(y_t|\mathbf{y}_{1:t-1})$ as $N \to \infty$. □

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TPAMI.2022.3153225, IEEE Transactions on Pattern Analysis and Machine Intelligence

APPENDIX C
PROOF OF COROLLARY 2.1.1

Proof. The implicit *smoothing* distribution that arises from performing SMC [25] is defined as

$$\tilde{q}(\mathbf{x}_{1:t}) = p(\mathbf{x}_{1:t}, \mathbf{y}_{1:t})\mathbb{E}\left[\frac{1}{\hat{p}(\mathbf{y}_{1:t})}\right]. \tag{54}$$

We can obtain the implicit filtering distribution by marginalizing out $d\mathbf{x}_{1:t-1}$ from (54)

$$\begin{aligned}
&\tilde{q}(x_t|\mathbf{y}_{1:t}) \\
&= \int p(\mathbf{x}_{1:t}, \mathbf{y}_{1:t})\mathbb{E}\left[\frac{1}{\hat{p}(\mathbf{y}_{1:t})}\right] d\mathbf{x}_{1:t-1}, \\
&= p(x_t, \mathbf{y}_{1:t})\mathbb{E}\left[\frac{1}{\hat{p}(\mathbf{y}_{1:t})}\right], \\
&= p(x_t, y_t|\mathbf{y}_{1:t-1})\mathbb{E}\left[\hat{p}(y_t|\mathbf{y}_{1:t-1})^{-1}\frac{p(\mathbf{y}_{1:t-1})}{\hat{p}(\mathbf{y}_{1:t-1})}\right].
\end{aligned} \tag{55}$$

In [25], [39], it was shown that

$$\begin{aligned}
\log p(\mathbf{y}_{1:t}) &\geq \mathbb{E}_{q(\mathbf{x}_{1:t})}[\log p(\mathbf{x}_{1:t}, \mathbf{y}_{1:t}) - \log \tilde{q}(\mathbf{x}_{1:t})] \\
&\geq \mathbb{E}[\log \hat{p}(\mathbf{y}_{1:t})].
\end{aligned} \tag{56}$$

Rearranging terms in (56), we get

$$\log p(y_t|\mathbf{y}_{1:t-1}) \geq \hat{\mathcal{L}}_t \geq \mathcal{L}_t. \tag{57}$$

where

$$\begin{aligned}
\hat{\mathcal{L}}_t = \mathbb{E}_{q(\mathbf{x}_{1:t})}[\log p(x_t, y_t|\mathbf{y}_{1:t-1}, \mathbf{x}_{1:t-1}) - \log \tilde{q}(x_t|\mathbf{x}_{1:t-1})] \\
+ \mathbb{D}_{\mathrm{KL}}[\tilde{q}(\mathbf{x}_{1:t-1})\|p(\mathbf{x}_{1:t-1}, \mathbf{y}_{1:t-1})] - \log p(\mathbf{y}_{1:t-1}).
\end{aligned} \tag{58}$$

By Theorem 2.1, we know that $\lim_{N\to\infty} \mathcal{L}_t = \log p(y_t|\mathbf{y}_{1:t-1})$, and thus

$$\lim_{N\to\infty} \hat{\mathcal{L}}_t = \log p(y_t|\mathbf{y}_{1:t-1}). \tag{59}$$

Leveraging Theorem 1 from [25] we have

$$\lim_{N\to\infty} \mathbb{D}_{\mathrm{KL}}[\tilde{q}(\mathbf{x}_{1:t-1})\|p(\mathbf{x}_{1:t-1}, \mathbf{y}_{1:t-1})] = \log p(\mathbf{y}_{1:t-1}) \tag{60}$$

which implies that

$$\lim_{N\to\infty} \tilde{q}(\mathbf{x}_{1:t-1}) = p(\mathbf{x}_{1:t-1}) \text{ a.e.} \tag{61}$$

thus plugging this into (59)

$$\begin{aligned}
&\log p(y_t|\mathbf{y}_{1:t-1}) \\
&= \int -\tilde{q}(\mathbf{x}_{1:t}) \log \frac{p(x_t, y_t|\mathbf{y}_{1:t-1}, \mathbf{x}_{1:t-1})}{\tilde{q}(x_t|\mathbf{x}_{1:t-1})}\, d\mathbf{x}_{1:t} \\
&= \int -\tilde{q}(\mathbf{x}_{1:t}) \log \frac{p(\mathbf{x}_{1:t}|\mathbf{y}_{1:t})p(y_t|\mathbf{y}_{1:t-1})}{\tilde{q}(x_t|\mathbf{x}_{1:t-1})p(\mathbf{x}_{1:t-1}|\mathbf{y}_{1:t-1})}\, d\mathbf{x}_{1:t} \\
&= \log p(y_t|\mathbf{y}_{1:t-1}) \\
&\quad + \int -\tilde{q}(\mathbf{x}_{1:t}) \log \frac{p(x_t|\mathbf{x}_{1:t-1}, \mathbf{y}_{1:t})}{\tilde{q}(x_t|\mathbf{x}_{1:t-1})}\, d\mathbf{x}_{1:t}
\end{aligned} \tag{62}$$

which is true iff $\tilde{q}(x_t|\mathbf{x}_{1:t-1}) = p(x_t|\mathbf{x}_{1:t-1}, \mathbf{y}_{1:t})$ almost everywhere. Thus by Lebesgue's dominated convergence theorem [31]

$$\begin{aligned}
&\lim_{N\to\infty} \int \tilde{q}(x_t|\mathbf{x}_{1:t-1}) d\mathbf{x}_{1:t-1} \\
&= \int \lim_{N\to\infty} \tilde{q}(x_t|\mathbf{x}_{1:t-1}) d\mathbf{x}_{1:t-1} \\
&= p(x_t|\mathbf{y}_{1:t}).
\end{aligned} \tag{63}$$

□

APPENDIX D
SYNTHETIC NASCAR® DYNAMICS

An rSLDS [51] with 4 discrete states was used to generate the synthetic NASCAR® track. The linear dynamics for each hidden state were

$$A_1 = \begin{bmatrix} \cos(\theta_1) & -\sin(\theta_1) \\ \sin(\theta_1) & \cos(\theta_1) \end{bmatrix}, A_2 = \begin{bmatrix} \cos(\theta_2) & -\sin(\theta_2) \\ \sin(\theta_2) & \cos(\theta_2) \end{bmatrix}, \tag{64}$$

and $A_3 = A_4 = I$. The affine terms were $B_1 = -(A_1 - I)c_1$, ($c_1 = [2, 0]^\top$), $B_2 = -(A_2 - I)c_2$, ($c_2 = [-2, 0]^\top$), $B_3 = [0.1, 0]^\top$ and $B_4 = [-0.35, 0]^\top$. The hyperplanes, R, and biases, r, were defined as

$$R = \begin{bmatrix} 100 & 0 \\ -100 & 0 \\ 0 & 100 \end{bmatrix}, \quad r = \begin{bmatrix} -200 \\ -200 \\ 0 \end{bmatrix}.$$

A state noise of $Q = 0.001I$ was used.

APPENDIX E
PREDICTION USING SVMC-GP

Let $\tilde{w}_t^i = \frac{w_t^i}{\sum_\ell w_t^\ell}$ be the self-normalized importance weights. At time t, given a test point x_* we can approximately sample from the predictive distribution

$$\begin{aligned}
&p(f_*|x_*, \mathbf{y}_{1:t}) \\
&= \int p(f_*|x_*, \mathbf{z}_t)p(\mathbf{z}_t|\mathbf{y}_{1:t})d\mathbf{z}_t \\
&= \int p(f_*|x_*, \mathbf{z}_t)p(\mathbf{z}_t|\mathbf{x}_{0:t})p(\mathbf{x}_{0:t}|\mathbf{y}_{1:t})d\mathbf{z}_t d\mathbf{x}_{0:t} \\
&= \int p(f_*|x_*, \mathbf{x}_{0:t})p(\mathbf{x}_{0:t}|\mathbf{y}_{1:t})d\mathbf{x}_{0:t} \\
&\approx \sum_{i=1}^N \tilde{w}_t^i\, p(f_*|x_*, \mathbf{x}_{0:t}^i) \\
&= \sum_{i=1}^N \tilde{w}_t^i \mathcal{N}(v_*^i, \Sigma_*^i)
\end{aligned} \tag{65}$$

where

$$v_*^i = m(x_*) + A_*\mu_t^i, \tag{66}$$

$$\Sigma_*^i = C_* + A_*\Gamma_t^i A_*^\top \tag{67}$$

where $A_* = K_{*z}K_{zz}^{-1}$ and $C_* = K_{**} - K_{*z}K_{zz}^{-1}K_{z*} + Q$. The approximate predictive distribution is a mixture of SGPs, allowing for a much more richer approximation to the predictive distribution. Equipped with (65), we approximate the mean of the predictive distribution, μ_{f_*}, as

$$\begin{aligned}
\mu_{f_*} &= \int f_* p(f_*|x_*, \mathbf{y}_{1:t})df_* \\
&\approx \int f_* \sum_{i=1}^N \tilde{w}_t^i\, p(f_*|x_*, \mathbf{x}_{0:t}^i)df_* \\
&= \sum_{i=1}^N \tilde{w}_t^i \int f_* p(f_*|x_*, \mathbf{x}_{0:t}^i)df_* \\
&= \sum_{i=1}^N \tilde{w}_t^i\, \mathbb{E}_i[f_*] = \sum_{i=1}^N \tilde{w}_t^i v_*^i = \hat{\mu}_{f_*}
\end{aligned} \tag{68}$$

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TPAMI.2022.3153225, IEEE Transactions on Pattern Analysis and Machine Intelligence

where $\mathbb{E}_i[\cdot] = \mathbb{E}_{p(f_*|x_*,\mathbf{x}^i_{0:t})}[\cdot]$.

Similarly, we can also approximate the covariance of of the predictive distribution, Σ_{f_*}

$$\begin{aligned}
\Sigma_{f_*} &= \int (f_* - \mu_{f_*})(f_* - \mu_{f_*})^\top p(f_*|x_*, \mathbf{y}_{1:t}) df_* \\
&\approx \sum_{i=1}^{N} \tilde{w}_t^i \int (f_* - \mu_{f_*})(f_* - \mu_{f_*})^\top p(f_*|x_*, \mathbf{x}^i_{0:t}) df_* \\
&= \sum_{i=1}^{N} \tilde{w}_t^i \, \mathbb{E}_i[(f_* - \mu_{f_*})(f_* - \mu_{f_*})^\top] \\
&= \sum_{i=1}^{N} \tilde{w}_t^i (\mathbb{E}_i[f_* f_*^\top] - \mathbb{E}_i[f_*]\mu_{f_*}^\top - \mu_{f_*}\mathbb{E}_i[f_*]^\top + \mu_{f_*}\mu_{f_*}^\top) \\
&= \sum_{i=1}^{N} \tilde{w}_t^i (\Sigma_*^i + v_*^i v_*^{i\top} - v_*^i \mu_{f_*}^\top - \mu_{f_*} v_*^{i\top} + \mu_{f_*}\mu_{f_*}^\top) \\
&\approx \sum_{i=1}^{N} \tilde{w}_t^i (\Sigma_*^i + v_*^i v_*^{i\top} - v_*^i \hat{\mu}_{f_*}^\top - \hat{\mu}_{f_*} v_*^{i\top} + \hat{\mu}_{f_*}\hat{\mu}_{f_*}^\top).
\end{aligned} \tag{69}$$

APPENDIX F
WINNER-TAKE-ALL SPIKING NEURAL NETWORK

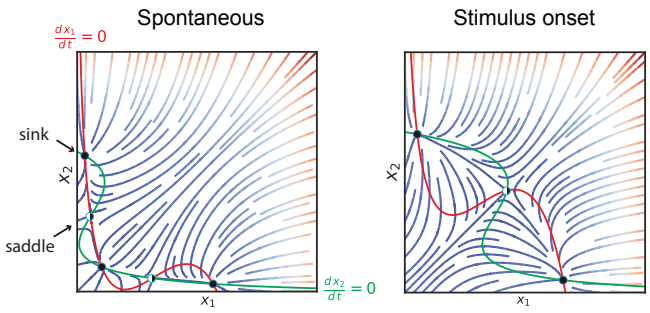

Figure 6. Mean field reduction of the Winner-Take-All spiking neural network.

In Figure 6 the mean-field reduction of the spiking network model is shown [58].

REFERENCES

[1] S. Haykin and J. Principe, "Making sense of a complex world [chaotic events modeling]," *IEEE Signal Processing Magazine*, vol. 15, no. 3, pp. 66–81, May 1998.

[2] J. Ko and D. Fox, "GP-BayesFilters: Bayesian filtering using gaussian process prediction and observation models," *Autonomous Robots*, vol. 27, no. 1, pp. 75–90, 5 2009.

[3] C. L. C. Mattos, Z. Dai, A. Damianou, J. Forth, G. A. Barreto, and N. D. Lawrence, "Recurrent gaussian processes," *International Conference on Learning Representations (ICLR)*, 2016.

[4] S. Roweis and Z. Ghahramani, *Learning nonlinear dynamical systems using the expectation-maximization algorithm*. John Wiley & Sons, Inc, 2001, pp. 175–220.

[5] D. Sussillo and O. Barak, "Opening the black box: Low-dimensional dynamics in high-dimensional recurrent neural networks," *Neural Computation*, vol. 25, no. 3, pp. 626–649, Mar. 2013.

[6] R. Frigola, Y. Chen, and C. E. Rasmussen, "Variational gaussian process state-space models," in *Proceedings of the 27th International Conference on Neural Information Processing Systems - Volume 2*, Montreal, Canada, 2014, pp. 3680–3688.

[7] B. C. Daniels and I. Nemenman, "Automated adaptive inference of phenomenological dynamical models," *Nature Communications*, vol. 6, pp. 8133+, Aug. 2015.

[8] Y. Zhao and I. M. Park, "Interpretable nonlinear dynamic modeling of neural trajectories," in *Advances in Neural Information Processing Systems (NIPS)*, 2016.

[9] J. Nassar, S. Linderman, M. Bugallo, and I. M. Park, "Tree-structured recurrent switching linear dynamical systems for multi-scale modeling," in *International Conference on Learning Representations*, 2019.

[10] Y. Ho and R. Lee, "A Bayesian approach to problems in stochastic estimation and control," *IEEE Transactions on Automatic Control*, vol. 9, no. 4, pp. 333–339, Oct. 1964.

[11] S. Särkkä, *Bayesian filtering and smoothing*. Cambridge University Press, 2013.

[12] S. S. Haykin, *Kalman filtering and neural networks*. Wiley, 2001.

[13] E. A. Wan and R. Van Der Merwe, "The unscented kalman filter for nonlinear estimation," in *Proceedings of the IEEE 2000 Adaptive Systems for Signal Processing, Communications, and Control Symposium (Cat. No. 00EX373)*. Ieee, 2000, pp. 153–158.

[14] E. A. Wan and A. T. Nelson, *Dual extended Kalman filter methods*. John Wiley & Sons, Inc, 2001, pp. 123–173.

[15] E. A. Wan, R. Van Der Merwe, and A. T. Nelson, "Dual estimation and the unscented transformation." in *NIPS*, vol. 99, 1999.

[16] E. A. Wan and A. T. Nelson, "Dual kalman filtering methods for nonlinear prediction, smoothing, and estimation," *Advances in neural information processing systems*, vol. 9, 1997.

[17] T. Broderick, N. Boyd, A. Wibisono, A. C. Wilson, and M. I. Jordan, "Streaming variational Bayes," in *Advances in Neural Information Processing Systems 26*, C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger, Eds. Curran Associates, Inc., 2013, pp. 1727–1735.

[18] Y. Zhao and I. M. Park, "Variational joint filtering," *arXiv:1707.09049*, 2017.

[19] A. Campbell, Y. Shi, T. Rainforth, and A. Doucet, "Online variational filtering and parameter learning," *arXiv preprint arXiv:2110.13549*, 2021.

[20] A. Doucet and A. M. Johansen, "A tutorial on particle filtering and smoothing: Fifteen years later," *Handbook of nonlinear filtering*, vol. 12, no. 656-704, p. 3, 2009.

[21] A. Doucet, N. de Freitas, and N. Gordon, *Sequential Monte Carlo Methods in Practice*. Springer Science & Business Media, Mar. 2013.

[22] J. Cornebise, E. Moulines, and J. Olsson, "Adaptive sequential monte carlo by means of mixture of experts," *Statistics and Computing*, vol. 24, no. 3, pp. 317–337, 2014.

[23] S. S. Gu, Z. Ghahramani, and R. E. Turner, "Neural adaptive sequential monte carlo," in *Advances in Neural Information Processing Systems*, 2015, pp. 2629–2637.

[24] P. Guarniero, A. M. Johansen, and A. Lee, "The iterated auxiliary particle filter," *Journal of the American Statistical Association*, vol. 112, no. 520, pp. 1636–1647, 2017.

[25] C. Naesseth, S. Linderman, R. Ranganath, and D. Blei, "Variational sequential monte carlo," in *Proceedings of the Twenty-First International Conference on Artificial Intelligence and Statistics*, ser. Proceedings of Machine Learning Research, A. Storkey and F. Perez-Cruz, Eds., vol. 84. Playa Blanca, Lanzarote, Canary Islands: PMLR, 09–11 Apr 2018, pp. 968–977.

[26] M. Titsias, "Variational learning of inducing variables in sparse gaussian processes," in *Artificial Intelligence and Statistics*, Apr. 2009, pp. 567–574.

[27] D. M. Blei, A. Kucukelbir, and J. D. McAuliffe, "Variational inference: A review for statisticians," *Journal of the American Statistical Association*, vol. 112, no. 518, pp. 859–877, 2017.

[28] C. Zhang, J. Butepage, H. Kjellstrom, and S. Mandt, "Advances in variational inference," *IEEE transactions on pattern analysis and machine intelligence*, 2018.

[29] M. J. Wainwright, M. I. Jordan *et al.*, "Graphical models, exponential families, and variational inference," *Foundations and Trends® in Machine Learning*, vol. 1, no. 1–2, pp. 1–305, 2008.

[30] A. B. Owen, *Monte Carlo theory, methods and examples*, 2013.

[31] A. W. Van der Vaart, *Asymptotic statistics*. Cambridge university press, 2000, vol. 3.

[32] T. Adali and S. Haykin, *Adaptive signal processing: next generation solutions*. John Wiley & Sons, 2010, vol. 55.

[33] S. Thrun, "Particle filters in robotics," in *Proceedings of the Eighteenth conference on Uncertainty in artificial intelligence*. Morgan Kaufmann Publishers Inc., 2002, pp. 511–518.

[34] A. Greenfield and A. Brockwell, "Adaptive control of nonlinear stochastic systems by particle filtering," in *2003 4th International Conference on Control and Automation Proceedings*. IEEE, 2003, pp. 887–890.

[35] F. Gustafsson, F. Gunnarsson, N. Bergman, U. Forssell, J. Jansson, R. Karlsson, and P.-J. Nordlund, "Particle filters for positioning, navigation, and tracking," *IEEE Transactions on signal processing*, vol. 50, no. 2, pp. 425–437, 2002.

[36] N. J. Gordon, D. J. Salmond, and A. F. M. Smith, "Novel approach to nonlinear/non-gaussian bayesian state estimation," in *IEE proceedings F (radar and signal processing)*, vol. 140, no. 2. IET, 1993, pp. 107–113.

[37] P. Bickel, B. Li, T. Bengtsson *et al.*, "Sharp failure rates for the bootstrap particle filter in high dimensions," in *Pushing the limits of contemporary statistics: Contributions in honor of Jayanta K. Ghosh*. Institute of Mathematical Statistics, 2008, pp. 318–329.

[38] I. Žliobaitė, M. Pechenizkiy, and J. Gama, "An overview of concept drift applications," in *Big data analysis: new algorithms for a new society*. Springer, 2016, pp. 91–114.

[39] T. A. Le, M. Igl, T. Rainforth, T. Jin, and F. Wood, "Auto-encoding sequential monte carlo," in *International Conference on Learning Representations*, 2018.

[40] C. Cremer, Q. Morris, and D. Duvenaud, "Reinterpreting Importance-Weighted Autoencoders," *arXiv e-prints*, Apr 2017.

[41] P. Del Moral, "Non-linear filtering: interacting particle resolution," *Markov processes and related fields*, vol. 2, no. 4, pp. 555–581, 1996.

[42] D. P. Kingma and M. Welling, "Auto-Encoding Variational Bayes," *arXiv:1312.6114 [cs, stat]*, May 2014, arXiv: 1312.6114.

[43] T. Rainforth, A. R. Kosiorek, T. A. Le, C. J. Maddison, M. Igl, F. Wood, and Y. W. Teh, "Tighter variational bounds are not necessarily better," *arXiv preprint arXiv:1802.04537*, 2018.

[44] R. H. Shumway and D. S. Stoffer, *Time Series Analysis and Its Applications: With R Examples (Springer Texts in Statistics)*. Springer, 2010.

[45] C. K. I. W. Carl Edward Rasmussen, *Gaussian Processes for Machine Learning*. MIT Press Ltd, 2006.

[46] E. Snelson and Z. Ghahramani, "Sparse Gaussian Processes using Pseudo-inputs," in *Advances in Neural Information Processing Systems 18*, Y. Weiss, B. Schölkopf, and J. C. Platt, Eds. MIT Press, 2006, pp. 1257–1264.

[47] A. Doucet, S. Godsill, and C. Andrieu, "On sequential monte carlo sampling methods for bayesian filtering," *Statistics and computing*, vol. 10, no. 3, pp. 197–208, 2000.

[48] D. Xu, "Learning nonlinear state space models with hamiltonian sequential monte carlo sampler," 2019.

[49] J. Liu and M. West, "Combined parameter and state estimation in simulation-based filtering," in *Sequential Monte Carlo methods in practice*. Springer, 2001, pp. 197–223.

[50] M. P. Deisenroth, R. D. Turner, M. F. Huber, U. D. Hanebeck, and C. E. Rasmussen, "Robust filtering and smoothing with gaussian processes," *IEEE Transactions on Automatic Control*, vol. 57, no. 7, pp. 1865–1871, 2011.

[51] S. Linderman, M. Johnson, A. Miller, R. Adams, D. Blei, and L. Paninski, "Bayesian learning and inference in recurrent switching linear dynamical systems," in *Artificial Intelligence and Statistics*, 2017, pp. 914–922.

[52] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *ICLR (Poster)*, 2015.

[53] D. Sussillo and L. Abbott, "Generating coherent patterns of activity from chaotic neural networks," *Neuron*, vol. 63, no. 4, pp. 544 – 557, 2009.

[54] O. Barak, D. Sussillo, R. Romo, M. Tsodyks, and L. F. Abbott, "From fixed points to chaos: three models of delayed discrimination." *Progress in neurobiology*, vol. 103, pp. 214–222, Apr. 2013.

[55] V. Mante, D. Sussillo, K. V. Shenoy, and W. T. Newsome, "Context-dependent computation by recurrent dynamics in prefrontal cortex," *Nature*, vol. 503, no. 7474, pp. 78–84, Nov. 2013.

[56] S. Ganguli, J. W. Bisley, J. D. Roitman, M. N. Shadlen, M. E. Goldberg, and K. D. Miller, "One-dimensional dynamics of attention and decision making in LIP," *Neuron*, vol. 58, no. 1, pp. 15–25, Apr. 2008.

[57] X.-J. Wang, "Probabilistic decision making by slow reverberation in cortical circuits," *Neuron*, vol. 36, no. 5, pp. 955–968, Dec. 2002.

[58] K.-F. Wong and X.-J. Wang, "A recurrent network mechanism of time integration in perceptual decisions," *The Journal of Neuroscience*, vol. 26, no. 4, pp. 1314–1328, Jan. 2006.

[59] I. D. Jordan and I. M. Park, "Birhythmic analog circuit maze: A nonlinear neurostimulation testbed," 2020.

[60] N. Chopin *et al.*, "Central limit theorem for sequential monte carlo methods and its application to bayesian inference," *The Annals of Statistics*, vol. 32, no. 6, pp. 2385–2411, 2004.

Yuan Zhao received his Ph.D. from Stony Brook University in 2016. He is a postdoc in the Department of Neurobiology and Behavior at Stony Brook University. His research interests lie in machine learning and computational neuroscience.

Josue Nassar received the B.S. and M.S. degree in electrical engineering from Stony Brook University in 2016 and 2018, respectively. He is currently a Ph.D. candidate in the department of electrical and computer engineering at Stony Brook University. His research interest lie at the intersection of computational neuroscience, control, signal processing and machine learning.

Ian Jordan received the BS degree in electrical engineering, specializing in control systems, from the New Jersey Institute of Technology in 2017. He is currently a PhD candidate at the Department of Applied Mathematics and Statistics at Stony Brook University. His research interests lie primarily in the field of applied dynamical systems theory, and the theory behind the underlying dynamics of recurrent neural networks.

Mónica Bugallo Mónica F. Bugallo is a Professor of Electrical and Computer Engineering and Associate Dean for Diversity and Outreach of the College of Engineering and Applied Sciences at Stony Brook University. She received her B.S., M.S, and Ph. D. degrees in Computer Science and Engineering from University of A Coruña, Spain. Her research interests are in the field of statistical signal processing, with emphasis on the theory of Monte Carlo methods and its application to different disciplines including biomedicine, ecology, sensor networks, and finance. In addition, she has focused on STEM education and has initiated several successful programs with the purpose of engaging students at all academic stages in the excitement of engineering and research, with focus on underrepresented groups. She is a senior member of the IEEE and the Vice Chair of the IEEE Signal Processing Theory and Methods Technical Committee and has served on several technical committees of IEEE conferences and workshops.

Il Memming Park is an Associate Professor in Neurobiology and Behavior at Stony Brook University. He is a computational neuroscientist trained in statistical modeling, information theory, and machine learning. He received his B.S. in computer science from KAIST, M.S. in electrical engineering and Ph.D. in biomedical engineering from the University of Florida (2010), and trained at University of Texas at Austin as a postdoctoral fellow (2010-2014).