DEEP DIRECT DISCRIMINATIVE DECODERS FOR HIGH-DIMENSIONAL TIME-SERIES DATA ANALYSIS

Anonymous authors
Paper under double-blind review

ABSTRACT

The state-space models (SSMs) are widely utilized in the analysis of time-series data. SSMs rely on an explicit definition of the state and observation processes. Characterizing these processes is not always easy and becomes a modeling challenge in many instances, such as when the dimension of observed data grows or the observed data distribution deviates from the normal distribution. New variants of SSMs try to address these challenges by utilizing the scalability of deep neural networks (DNNs) in the SSM formulation. Here, we propose a new formulation of SSM for high-dimensional observation processes with a heavy-tailed distribution. We call this solution the deep direct discriminative process (D4). The D4 utilizes discriminative models like DNN in characterizing the observation process. With this formulation, we bring DNNs' expressiveness and scalability to the SSM formulation letting us build a novel solution that efficiently estimates the underlying state processes through high-dimensional observation signal. For the D4, we define the Bayesian filter solution and develop a training algorithm that finds the model-free parameters. We demonstrate the D4 solutions in simulated and real data such as Lorenz attractors, Langevin dynamics, random walk dynamics, and rat hippocampus spiking neural data and show that the D4's performance precedes traditional SSMs and RNNs. The D4 can be applied to a broader class of time-series data where the connection between high-dimensional observation and the underlying latent process is hard to characterize.

1 INTRODUCTION

The state-space model is one of the well-established dynamical latent variable modeling frameworks successfully applied in the analysis of dynamical time-series data Paninski et al. (2010). The Kalman filter, the most widely known form of SSMs, has been frequently utilized in characterizing a wide range of time-series data from healthcare Zhang et al. (2015), navigation Grewal et al. (1990), machine vision Kiriy & Buehler (2002), and neuroscience Eden et al. (2004). The standard SSM consists of a state process that defines how the state, i.e. dynamical latent variables, evolves in time, and an observation process that defines the conditional distribution of the observed signals given the state variable(s) Durbin & Koopman (2012). A modeling challenge in SSMs is to build an accurate conditional probability distribution of the observed signal given the state, specifically, for the cases where the dimension of the observed signal is high or it has a heavy-tailed distribution. In practice, the observation distribution model is simplified with assumptions such as conditional independence between dimensions of the signal or normal distribution for the observed signal. These assumptions might not hold in many datasets, including our research domain where we have neural activity of thousands of neurons. New advances in SSMs focused on circumventing this problem by increasing the dimension of the state variables, characterizing the observation conditional distribution becomes easy given a large set of state variables Zoltowski et al. (2020); Linderman et al. (2017); Glaser et al. (2020). Although this approach increases the expressive power of SSMs, we lose the interpretability of the expanded state variables, given the extra state variable might not represent the temporal dynamics of the underlying physical or biological systems. In addition, identifying the optimal number of the state dimension is computationally intractable, and the proposed solutions lack a mechanism for identifying the number of states.

Most recently, discriminative models such as deep neural networks (DNNs) and recurrent neural networks (RNNs) Burkhart et al. (2020); Krishnan et al. (2015); Rezaei et al. (2018; 2022); Glaser et al.

(2020) are introduced to SSM for increasing its scalability in the analysis of high dimensional data. A modeling concern with these solutions is that the DNN expressive power is not efficiently embedded in their solutions; as a result, they will not necessarily address the challenges of characterizing high-dimensional observation. For instance, the discriminative Kalman filters (DKFs) Burkhart et al. (2020) and direct discriminative decoders (DDD) Rezaei et al. (2022) use a simple discriminative model, such as a linear model with an additive Gaussian noise or the history of observed signals is discarded in the estimation of the state. In other solutions like the deep Kalman filter (Deep-KF)Krishnan et al. (2015), the linear observation and state transition processes are replaced with DNNs Krishnan et al. (2015); Rangapuram et al. (2018) and mainly ignore temporal dynamics of the underlying biological or physical systems. This modeling assumption is prohibitive when a proper prior is available for the state dynamics, or the interpretability of the inferred state is of interest.
A critical advantage of SMMs is their explicit definition of the state dynamics and observation processes that allow information carried by the state and observation to be optimally combined to infer the underlying latent dynamics. In the meantime, the advantage of the DNNs and RNNs is expressive power and scalability. The deep direct discriminative decoder (D4) solution proposed here leverages both SSM and DNN advantages in characterizing observed signals and inference of the underlying dynamics. The D4 incorporates a state transition process like the way is being utilized in SSMs and utilizes DNN or RNNs to augment the conditional distribution of observation by a discriminative process. In contrast to the previous solutions, our discriminative process can maintain DNN's expressiveness as it optimally combines the current and history of the observation in its estimation of the underlying state. We argue that the D4 will not only address scalability issues we face in SSMs for high-dimensional time-series data but also will reach a high level of accuracy in the estimation of state dynamics. We tested D4 on simulation and real datasets, where its performance preceded traditional DNN and SSM models. The D4 applications are broader and can be applied to different modalities of time-series data without constraints on the modality or distribution of observed data. We argue D4 solution facilitates the analysis of high-dimensional data where the connection between the high-dimensional observation and the underlying state process needs to be interpretable.

1.1 BACKGROUNDS

Consider the problem of modeling time-series data $\boldsymbol{y}_{1:K}, \boldsymbol{y}_k \in \mathbf{R}^N$, where $k = 1, ..., K$, using dynamical latent variables $\boldsymbol{x}_{1:K}, \boldsymbol{x}_k \in \mathbf{R}^M$ with a Markovian property. Under the SSM modeling framework, the joint probability distribution of latent variables and observations can be factorized by conditional probabilities of a generative processes defined by

$$p(\boldsymbol{x}_{1:K}, \boldsymbol{y}_{1:K}) = p(\boldsymbol{x}_1)p(\boldsymbol{y}_1|\boldsymbol{x}_1) \prod_{k=2}^{K} p(\boldsymbol{x}_k|\boldsymbol{x}_{k-1})p(\boldsymbol{y}_k|\boldsymbol{x}_k). \tag{1}$$

The posterior distribution of $\boldsymbol{x}_{1:k}$ given $\boldsymbol{y}_{1:k}$, the filter solution, is defined by the following recursive solution

$$p(\boldsymbol{x}_{1:k}|\boldsymbol{y}_{1:k}) \propto p(\boldsymbol{y}_k|\boldsymbol{x}_k)p(\boldsymbol{x}_k|\boldsymbol{x}_{k-1})p(\boldsymbol{x}_{1:k-1}|\boldsymbol{y}_{1:k-1}), \tag{2}$$

where the first term is the likelihood function given the current observation, the second term is the state process conditional distribution, and the last term is the posterior from the previous time index Chen et al. (2003). A modeling challenge in SSM is to build the conditional probability of the observed signal. In practice, the likelihood function is simplified by an assumption like observations are conditionally independent given the state or they follow normal distributions. These assumptions are prohibitive in characterizing datasets such as neural data, where the observed signals include the spiking activity which can not be properly characterized by a normal distribution. These simplifications will potentially induce bias in the estimation of the underlying state process, causing an inaccurate inference of the state variables. The SSM modeling structure, specifically its generative model for the observation process, will avoid the utilization of many powerful tools such as DNNs in its characterization of observed signals and estimation of the underlying states. With this in mind, we propose a new variant of SMMs called the deep direct discriminative decoder (D4) model; a new modeling framework that fuses the advantages of DNN in the SSM to build a scalable and potentially accurate solution for characterizing high-dimensional dynamical time-series data.

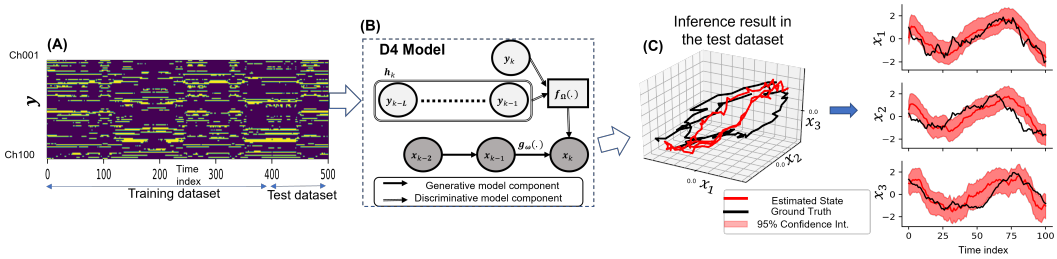

Figure 1: The D4 latent state inference for the Lorenz system with point-process observations described in section 3.2. A) 100 simulated spiking channels using a point process intensity model, see Appendix C, from a trajectory consists of 500 data samples. B) Graphical representation of the D4 model. $\boldsymbol{g}_\omega(.)$ is the state transition process parameterized by ω and $\boldsymbol{f}_\Omega(.)$ is a discriminative process parameterized by Ω. C) The D4 inferred 3-D trajectory for the Lorenz latent states. The D4 successfully inferred the latent state trajectory as most of the points in the trajectory are covered by the 95% HPD of the D4's predictions.

2 D4 MODEL

Let's assume $\boldsymbol{h}_k = \boldsymbol{y}_{1:k-1}$. With this assumption, we can rewrite the posterior distribution of $\boldsymbol{x}_{1:k}$ given $\boldsymbol{y}_{1:k}$ as

$$p(\boldsymbol{x}_{1:k}|\boldsymbol{y}_k, \boldsymbol{h}_k) = \frac{p(\boldsymbol{x}_{1:k}, \boldsymbol{y}_k, \boldsymbol{h}_k)}{p(\boldsymbol{y}_k, \boldsymbol{h}_k)} = \frac{p(\boldsymbol{y}_k|\boldsymbol{x}_k, \boldsymbol{x}_{1:k-1}, \boldsymbol{h}_k)p(\boldsymbol{x}_k, \boldsymbol{x}_{1:k-1}, \boldsymbol{h}_k)}{p(\boldsymbol{y}_k, \boldsymbol{h}_k)}. \tag{3}$$

With the Markovian assumption of the state process and factorization rule, we can rewrite equation 3 as

$$p(\boldsymbol{x}_{1:k}|\boldsymbol{y}_k, \boldsymbol{h}_k) =$$
$$\frac{p(\boldsymbol{y}_k|\boldsymbol{x}_k, \boldsymbol{h}_k)p(\boldsymbol{x}_k, \boldsymbol{x}_{1:k-1}, \boldsymbol{h}_k)}{p(\boldsymbol{y}_k, \boldsymbol{h}_k)} = \frac{p(\boldsymbol{y}_k|\boldsymbol{x}_k, \boldsymbol{h}_k)p(\boldsymbol{x}_k|\boldsymbol{x}_{k-1})p(\boldsymbol{x}_{1:k-1}|\boldsymbol{h}_k)p(\boldsymbol{h}_k)}{p(\boldsymbol{y}_k, \boldsymbol{h}_k)}, \tag{4}$$

where we replace $p(\boldsymbol{x}_k|\boldsymbol{x}_{1:k-1}, \boldsymbol{h}_k)$ with $p(\boldsymbol{x}_k|\boldsymbol{x}_{k-1})$. By applying the Bayes rule once again for $p(\boldsymbol{y}_k|\boldsymbol{x}_k, \boldsymbol{h}_k)$ and replacing $\boldsymbol{h}_k$ with $\{\boldsymbol{y}_{k-1}, \boldsymbol{h}_{k-1}\}$, we can rewrite the posterior as

$$p(\boldsymbol{x}_{1:k}|\boldsymbol{y}_k, \boldsymbol{h}_k) = \frac{p(\boldsymbol{x}_k|\boldsymbol{y}_k, \boldsymbol{h}_k)p(\boldsymbol{y}_k, \boldsymbol{h}_k)p(\boldsymbol{x}_k|\boldsymbol{x}_{k-1})p(\boldsymbol{x}_{1:k-1}|\boldsymbol{h}_k)p(\boldsymbol{h}_k)}{p(\boldsymbol{y}_k, \boldsymbol{h}_k)p(\boldsymbol{x}_k|\boldsymbol{h}_k)p(\boldsymbol{h}_k)} =$$
$$\frac{p(\boldsymbol{x}_k|\boldsymbol{y}_k, \boldsymbol{h}_k)}{p(\boldsymbol{x}_k|\boldsymbol{h}_k)}p(\boldsymbol{x}_k|\boldsymbol{x}_{k-1})p(\boldsymbol{x}_{1:k-1}|\boldsymbol{y}_{k-1}, \boldsymbol{h}_{k-1}) \tag{5}$$

where $p(\boldsymbol{x}_k|\boldsymbol{x}_{k-1})$ is the state transition process and we call $p(\boldsymbol{x}_k|\boldsymbol{y}_k, \boldsymbol{h}_k)$ the prediction process. The $p(\boldsymbol{x}_{1:k-1}|\boldsymbol{y}_{k-1}, \boldsymbol{h}_{k-1})$ becomes the posterior distribution from the previous time index and the denominator term, $p(\boldsymbol{x}_k|\boldsymbol{h}_k)$, can be expanded by $p(\boldsymbol{x}_k|\boldsymbol{h}_k) = \int d\boldsymbol{x}_{k-1}p(\boldsymbol{x}_k|\boldsymbol{x}_{k-1})p(\boldsymbol{x}_{k-1}|\boldsymbol{h}_{k-1}, \boldsymbol{y}_{k-1})$. Equation 5 gives a recursive solution to calculate the posterior at time index k. In this formulation, the ratio term $p(\boldsymbol{x}_k|\boldsymbol{y}_k, \boldsymbol{h}_k)/p(\boldsymbol{x}_k|\boldsymbol{h}_k)$ is equivalent to the update term in the SSM and represents the amount of information carried by the current observation. If the current observation is not informative, the ratio is close to one which corresponds to a flat likelihood in the SSM for the observed signal; on the other hand, it will push the one-step prediction distribution toward a new domain of states implied by the observed data.
As Figure1.B shows, the D4 is comprised of two equations: a) a state transition equation, and b) a prediction process equation. The state transition equation at time index k is defined by

$$\boldsymbol{x}_k|\boldsymbol{x}_{k-1} \sim g(\boldsymbol{x}_{k-1}; \boldsymbol{\omega}), \tag{6}$$

where $\boldsymbol{\omega}$ is the model parameters of the conditional distribution. The prediction process is defined by

$$\boldsymbol{x}_k|\boldsymbol{y}_k, \boldsymbol{h}_k \sim f(\boldsymbol{y}_k, \boldsymbol{h}_k; \boldsymbol{\Omega}). \tag{7}$$

In practice, $\boldsymbol{h}_k$ is assumed to be a subset of the observation from previous time points, and $\boldsymbol{\Omega}$ is the model free parameters of prediction process defined by the discriminative function f.
Equations 6 and 7 define the D4 model, where, the observation equation of SSM is replaced by a

discriminative process. The discriminative process can be built using the state-of-the-art discriminative models such DNNs, RNNs, CNNs, or variants of these models. The flexibility in picking the discriminative process allows us to benefit from the scalability and expressiveness of these models, where these choices make the D4 agnostic to the modality of the input data.
Here we defined the D4 model structure, equations 6, 7, and 5 give a recursive solution to compute the posterior distribution of the states variables. Due to involvement of an integration term in equation 5, the calculations in equation 5 is not strait forward. Note that the estimation of posterior distribution, equation 5, involves an integral term; with this term, deriving a closed-form solution is becoming a challenging problem for the posterior. As a result, in the next sections, we first propose efficient sampling solution that allows to draw sample of state trajectories recursively for the state posterior distribution, defined by $P(\boldsymbol{x}_{1:K} \mid \boldsymbol{y}_{1:K})$. We also discuss the computational complexity of sampling solution for the posterior estimation in the D4 and compare it with the SSM one. We then discuss the training solution for the D4; we will see how the $P(\boldsymbol{x}_{1:K} \mid \boldsymbol{y}_{1:K})$ samples will be used in the training of the D4 components and discuss how in practice we can reduce the D4 computational cost by controlling the $\boldsymbol{h}_k$ term.

2.1 SMOOTHED SEQUENTIAL IMPORTANCE SAMPLING

Equation 5 defines the posterior distribution of the state variables given the observation. Like sequential Monte Carlo sampling Doucet et al. (2009), we assume there is a proposal distribution define by $q_k(\boldsymbol{x}_k \mid \boldsymbol{x}_{k-1})$. Let's assume at time index k, we draw D samples using the proposal distribution give the samples form the previous time point. With this assumption, the weight for dth sample, $w_{k|k}^{(d)}$, is defined by

$$\begin{aligned} \boldsymbol{x}_k^{(d)} &\sim q(\boldsymbol{x}_k^{(d)}|\boldsymbol{x}_{k-1}^{(d)}, \boldsymbol{y}_k), \\ w_{k|k}^{(d)} &:= \frac{p(\boldsymbol{x}_k^{(d)}|\boldsymbol{x}_{k-1}^{(d)}) p(\boldsymbol{x}_k^{(d)}|\boldsymbol{y}_k, \boldsymbol{h}_k)}{q(\boldsymbol{x}_k^{(d)}|\boldsymbol{x}_{k-1}^{(d)}, \boldsymbol{y}_k) \int d\boldsymbol{x}_{k-1}^{(d)} p(\boldsymbol{x}_k^{(d)} \mid \boldsymbol{x}_{k-1}^{(d)}) p(\boldsymbol{x}_{k-1}^{(d)}|\boldsymbol{h}_k)} \end{aligned} \tag{8}$$

The integral term in the denominator of equation 8 can be numerically approximated using the samples from the previous time index, $\boldsymbol{x}_{k-1}^{(d)}$. In the cases where $f(., \boldsymbol{\omega})$ is smooth and invertible Rezende & Mohamed (2015), we can approximate the integral term by

$$p(\boldsymbol{x}_k^{(d)}|\boldsymbol{h}_k, \{\boldsymbol{\omega}, \boldsymbol{\Omega}\}) = \int d\boldsymbol{x}_{k-1}^{(d)} p(\boldsymbol{x}_k^{(d)} \mid \boldsymbol{x}_{k-1}^{(d)}, \boldsymbol{\omega}) p(\boldsymbol{x}_{k-1}^{(d)}|\boldsymbol{h}_{k-1}, \boldsymbol{y}_{k-1}, \boldsymbol{\Omega}) \approx p(f(\boldsymbol{x}_{k-1}^{(d)}, \boldsymbol{\omega}^*)|\boldsymbol{h}_k, \boldsymbol{\Omega}) \tag{9}$$

where $\boldsymbol{\omega}^*$ satisfies

$$\boldsymbol{\omega}^* = argmin_{\boldsymbol{\omega}} \mathbb{KL}(p(\boldsymbol{x}_k^{(d)}|\boldsymbol{h}_k, \{\boldsymbol{\omega}, \boldsymbol{\Omega}\}) \parallel p(f(\boldsymbol{x}_{k-1}^{(d)}, \boldsymbol{\omega})|\boldsymbol{h}_k, \boldsymbol{\Omega}) \tag{10}$$

Using equation 8, we draw samples from the filter estimation. To draw samples from the smoother estimate of the state, we use forward filtering and backwards smoothing (FFBS) formula suggested in Kitagawa (1996). FFBS first runs the filter solution define by equation 8 for the D4; it then reweighs particles with the backward recursion defined by

$$w_{k|K}^{(d)} = \overline{w}_{k|k}^{(d)} \left[\sum_{i=1}^{K} w_{k+1|K}^{(i)} \frac{p(\boldsymbol{x}_{k+1}^{(i)}|\boldsymbol{x}_k^{(k)})}{\sum_{j=1}^{K} \overline{w}_{k|k}^{(j)} p(\boldsymbol{x}_{k+1}^{(i)}|\boldsymbol{x}_j^{(k)})} \right] \tag{11}$$

where $\overline{w}_{K|k}^{(d)} = w_{K|k}^{(d)} / \sum_{j=1}^{D} w_{K|k}^{(j)}$ with $w_{K|K}^{(d)} = \overline{w}_{K|K}^{(d)}$. Using equations 8 and 11, we can draw samples from the posterior of the state given the whole observation. Note that we will do the resampling at each time index of the filter solution.

2.2 COMPUTATIONAL COMPLEXITY OF SMMS VS D4

The filtering for SSM, defined in equation 2 for SSM requires $O(NK)$ operations to sample one trajectory of the state approximately distributed according to $p(\boldsymbol{x}_{1:K}|\boldsymbol{y}_{1:K})$Doucet et al. (2009). On the other hand, the D4 requires $O(NK^2)$ operations to sample one path for the same distribution, if we consider $\boldsymbol{h}_k = \boldsymbol{y}_{1:k-1}$. In practice, we replace the $\boldsymbol{h}_k$ with a fixed length history $\boldsymbol{h}'_k = \{\boldsymbol{y}_j\}_{j=k-1}^{j=k-L-1}$ which can reduce the computational cost to $Q(NKL)$. In the following sections, we discuss the

training algorithm where we can estimate the model free parameters (ω, Ω) along with the optimal history length, as a result, the training algorithm outcome for the history length will impact the computation complex of the filter and smoother solution.

2.3 D4 MODEL TRAINING

With the D4 training, we maximize the likelihood of the observation data by tuning the model free parameters. The D4 free parameters include the state process parameters - e.g. ω, and the discirminative model free parameters - e.g Ω. Note that the history term, $\boldsymbol{h}_k$, needs to be identified, which will change the optimal values of $\boldsymbol{\omega}$ and $\boldsymbol{\Omega}$ of the D4 model. Note that the state variables, $\boldsymbol{x}_k$, are not directly observed; thus, the training requires the estimation of the state process Kenny & Judd (1984). For this setting, we can use the EM algorithm, which let us find the ML estimates of the model parameters Myung (2003). The EM alternates between computing the expected complete log-likelihood according to the posterior estimation of the state (the E step) and maximizing this expectation, $\boldsymbol{Q}$ function, by updating the model free parameters (the M step). The $\boldsymbol{Q}$ is defined by

$$\boldsymbol{Q}(\boldsymbol{\theta}|\boldsymbol{\theta}^{(r)}) = \mathbb{E}_{\boldsymbol{x}_{0:K}|\boldsymbol{y}_{1:K};\boldsymbol{\theta}^{(r)}}[\log(p(\boldsymbol{x_0};\boldsymbol{\omega}_0)\prod_{k=1}^{K} p(\boldsymbol{x}_k|\boldsymbol{x}_{k-1};\boldsymbol{\omega})\prod_{k=1}^{K}\frac{p(\boldsymbol{x}_k|\boldsymbol{y}_k,\boldsymbol{h}_k;\boldsymbol{\Omega})}{p(\boldsymbol{x}_k|\boldsymbol{h}_k;\boldsymbol{\Omega})})] \tag{12}$$

where $\boldsymbol{\theta}^r = \{\boldsymbol{\omega}_0^r, \boldsymbol{\Omega}^r, \boldsymbol{\omega}^r\}$ represents the model parameters estimated by maximizing $\boldsymbol{Q}$ at the previous iteration of the EM algorithm Yousefi et al. (2015). For the simplicity of notation, we use $\mathbb{E}_K$ for $\mathbb{E}_{\boldsymbol{x}_{0:K}|\boldsymbol{y}_{1:K};\boldsymbol{\theta}^{(r)}}$ in the rest of the paper. We can expand the $\boldsymbol{Q}$ function, as

$$\boldsymbol{Q}(\boldsymbol{\theta}|\boldsymbol{\theta}^{(r)}) = \mathbb{E}_K[\log\ p(\boldsymbol{x}_0;\boldsymbol{\omega}_0)+$$
$$\sum_{k=1}^{K}\log\ p(\boldsymbol{x}_k|\boldsymbol{x}_{k-1};\boldsymbol{\omega}) + \sum_{k=1}^{K}\log\ p(\boldsymbol{x}_k|\boldsymbol{y}_k,\boldsymbol{h}_k;\boldsymbol{\Omega})] - \mathbb{E}_K[\sum_{k=1}^{K}\log\ p(\boldsymbol{x}_k|\boldsymbol{h}_k;\boldsymbol{\Omega})] \tag{13}$$

For the cases where the state process is linear with an additive Gaussian noise and the prediction process is a multi-variate Gaussian, we can find a closed-form solution for different terms of the $\boldsymbol{Q}$ function. However, for almost all other cases such as when the prediction process is a non-linear function of the observation or the state transition represents a non-linear dynamics, it is hard to find a closed-form solution for $\boldsymbol{Q}$ function given the model free parameters. Instead, we can use the sampling technique, described in the section 2.1, to calculate the expectation for the first term, $\mathbb{E}_K[.]$, of equation 13; however, finding the expectation for the last term is still challenging given it involves a second integration over the state variable. To address this challenge, we show that we can find an approximation for the last term which turns into a lower bound for the $\boldsymbol{Q}$. We can write the last term in $\boldsymbol{Q}$ function as

$$\mathbb{E}_K[\sum_{k=1}^{K}\log\ p(\boldsymbol{x}_k|\boldsymbol{h}_k;\boldsymbol{\Omega})] = \sum_{k=1}^{K}\int d\boldsymbol{x}_k p(\boldsymbol{x}_k|\boldsymbol{y}_{1:K},\boldsymbol{\theta}^{(r)})\log p(\boldsymbol{x}_k|\boldsymbol{h}_k;\boldsymbol{\Omega}) =$$
$$= \sum_{k=1}^{K}\int p(\boldsymbol{x}_k|\boldsymbol{y}_{1:K},\boldsymbol{\theta}^{(r)})\log\frac{p(\boldsymbol{x}_k|\boldsymbol{h}_k;\boldsymbol{\Omega})p(\boldsymbol{x}_k|\boldsymbol{y}_{1:K},\boldsymbol{\theta}^{(r)})}{p(\boldsymbol{x}_k|\boldsymbol{y}_{1:K},\boldsymbol{\theta}^{(r)})}d\boldsymbol{x}_k$$
$$= \sum_{k=1}^{K}(\int d\boldsymbol{x}_k p(\boldsymbol{x}_k|\boldsymbol{y}_{1:K},\boldsymbol{\theta}^{(r)})\log\frac{p(\boldsymbol{x}_k|\boldsymbol{h}_k;\boldsymbol{\Omega})}{p(\boldsymbol{x}_k|\boldsymbol{y}_{1:K},\boldsymbol{\theta}^{(r)})} + \int d\boldsymbol{x}_k p(\boldsymbol{x}_k|\boldsymbol{y}_{1:K},\boldsymbol{\theta}^{(r)})\log p(\boldsymbol{x}_k|\boldsymbol{y}_{1:K},\boldsymbol{\theta}^{(r)}), \tag{14}$$

where the first term here is the negative of KL divergence between $p(\boldsymbol{x}_k|\boldsymbol{y}_{1:K},\boldsymbol{\theta}^{(r)})$ and $p(\boldsymbol{x}_k|\boldsymbol{h}_k;\boldsymbol{\Omega})$, and the second term is the negative of the entropy of the $\boldsymbol{x}_k$ posterior distribution given the whole observation. Notice that $\mathbb{H}(p(\boldsymbol{x}_k|\boldsymbol{y}_{1:K},\boldsymbol{\theta}^{(r)})$ is independent of the free parameters $\boldsymbol{\theta}$. As a result, we can rewrite $\boldsymbol{Q}$ as

$$\boldsymbol{Q}(\boldsymbol{\theta}|\boldsymbol{\theta}^{(r)}) = \mathbb{E}_K[\log\ p(\boldsymbol{x}_0;\boldsymbol{\omega}_0) + \sum_{k=1}^{K}\log\ p(\boldsymbol{x}_k|\boldsymbol{x}_{k-1};\boldsymbol{\omega})+$$
$$\sum_{k=1}^{K}\log\ p(\boldsymbol{x}_k|\boldsymbol{y}_k,\boldsymbol{h}_k;\boldsymbol{\Omega})] + \sum_{k=1}^{K}\mathbb{KL}(p(\boldsymbol{x}_k|\boldsymbol{y}_{1:K},\boldsymbol{\theta}^{(r)})\ \|\ p(\boldsymbol{x}_k|\boldsymbol{h}_k;\boldsymbol{\Omega})) + \mathbb{H}(p(\boldsymbol{x}_k|\boldsymbol{y}_{1:K},\boldsymbol{\theta}^{(r)}) \tag{15}$$

Algorithm 1 D4 Learning Algorithm

procedure REGULARIZED-EM-FOR-D4($\boldsymbol{y}_{1:K}, \boldsymbol{\theta}^{(0)}, \lambda, D, L$)
  $\boldsymbol{h}_k \leftarrow \{\boldsymbol{y}_{k-L:k-1}\}, \boldsymbol{Q}^0 \leftarrow 0$
  Do
    $\boldsymbol{Q}^{max} \leftarrow \boldsymbol{Q}^r$
    $\tilde{\boldsymbol{x}}_{1:K}^{1:D} \leftarrow \boldsymbol{x}_{1:K}^{1:D}$
    Sample D smoothed trajectories using equations 8 and 11, $\boldsymbol{x}_{1:K}^{1:D} \sim p_{\boldsymbol{\theta}}(\boldsymbol{x}_{1:K}^{1:D} \mid \boldsymbol{y}_{1:K})$
    $\boldsymbol{\theta}^{(r)}, \boldsymbol{Q}^r = Update - Model(\boldsymbol{y}_{1:K}, \tilde{\boldsymbol{x}}_{1:K}^{1:D}, \boldsymbol{x}_{1:K}^{1:D}, \boldsymbol{\theta}^{(r-1)}, \lambda)$
  DoWhile$\{\boldsymbol{Q}^r > \boldsymbol{Q}^{max}\}$
  return $\boldsymbol{\theta}^{(r-1)}, \boldsymbol{Q}^{max}$
end procedure
procedure UPDATE-MODEL($\boldsymbol{y}_{1:K}, \tilde{\boldsymbol{x}}_{1:K}^{1:D}, \boldsymbol{x}_{1:K}^{1:D}, \boldsymbol{\theta}^{(r-1)}, \lambda$)
  $\{\boldsymbol{\omega}_0^{(r-1)}, \boldsymbol{\omega}^{(r-1)}, \boldsymbol{\Omega}^{(r-1)}\} = \boldsymbol{\theta}^{(r-1)}$
  Update $\boldsymbol{Q}^r$ using equation 17 evaluate at $\boldsymbol{\theta}^{(r-1)}$
  Update $\boldsymbol{\Omega}^{(r)}$, $\boldsymbol{\omega}^{(r)}$, and $\boldsymbol{\omega}_0^{(r)}$ using gradients calculated by equations 18, 26, and 27; respectively
  return $\{\boldsymbol{\omega}_0^{(r)}, \boldsymbol{\omega}^{(r)}, \boldsymbol{\Omega}^{(r)}\}, \boldsymbol{Q}^r$
end procedure

The KL term in equation15 is positive; thus, a lower bound for the $\boldsymbol{Q}$ can be defined as the expectation of the likelihood terms defined by the state transition and prediction processes. The KL term can not be zero; however, it can generate smaller values as the length of the history term in the discriminative model grows.

Note that the history term has two contrasting effects in the $\boldsymbol{Q}$ function. As the history length grows, the $\boldsymbol{Q}$ value grows as the prediction process, the $\log p(\boldsymbol{x}_k|\boldsymbol{y}_k, \boldsymbol{h}_k; \boldsymbol{\Omega})$, will better fit the state, wheres it brings the value of the KL down. Thus, we can assume the KL term in equation 15, penalizes the length of $\boldsymbol{h}_k$ and prevents unconstrained growth of the $\boldsymbol{h}_k$. This dynamics of KL suggests that there is an optimal history length that not only contributes to a better prediction of the state but also pushes $\boldsymbol{Q}$ to a higher value. We assume that the sum of KL and H over the whole time indices should be smaller than a pre-set threshold ϵ. With this assumption, we can approach our model training as a regularized maximum likelihood problem defined by

$$\max_{\boldsymbol{\theta}} \mathbb{E}_K[\log\ p(\boldsymbol{x}_0; \boldsymbol{\omega}_0) + \sum_{k=1}^{K} \log\ p(\boldsymbol{x}_k|\boldsymbol{x}_{k-1}; \boldsymbol{\omega}) + \sum_{k=1}^{K} \log\ p(\boldsymbol{x}_k|\boldsymbol{y}_k, \boldsymbol{h}_k; \boldsymbol{\Omega})]$$

$$s.t.\ \sum_{k=1}^{K} \mathbb{KL}(p(\boldsymbol{x}_k|\boldsymbol{y}_{1:K}, \boldsymbol{\theta}^{(r)}) \parallel p(\boldsymbol{x}_k|\boldsymbol{h}_k; \boldsymbol{\Omega})) + \mathbb{H}(p(\boldsymbol{x}_k|\boldsymbol{y}_{1:K}, \boldsymbol{\theta}^{(r)})) < \epsilon \tag{16}$$

Re-writing equation 16 as a Lagrangian under the KKT conditions Higgins et al. (2016), we obtain

$$\boldsymbol{Q}(\boldsymbol{\theta}, \lambda, \mid \boldsymbol{\theta}^{(r)}) = \mathbb{E}_K[\log\ p(\boldsymbol{x}_0; \boldsymbol{\omega}_0) + \sum_{k=1}^{K} \log\ p(\boldsymbol{x}_k|\boldsymbol{x}_{k-1}; \boldsymbol{\omega}) + \sum_{k=1}^{K} \log\ p(\boldsymbol{x}_k|\boldsymbol{y}_k, \boldsymbol{h}_k; \boldsymbol{\Omega})]$$

$$-\lambda(\sum_{k=1}^{K} \mathbb{KL}(p(\boldsymbol{x}_k|\boldsymbol{y}_{1:K}, \boldsymbol{\theta}^{(r)}) \parallel p(\boldsymbol{x}_k|\boldsymbol{h}_k; \boldsymbol{\Omega})) + \mathbb{H}(p(\boldsymbol{x}_k|\boldsymbol{y}_{1:K}, \boldsymbol{\theta}^{(r)})) \tag{17}$$

$\lambda \geq 0$ keeps the KL non-zero by putting pressure to shrink history length.

2.4 STOCHASTIC OPTIMIZATION FOR PARAMETER ESTIMATION

We can optimize the objective function defined in equation 17 by using a stochastic optimization algorithm. Stochastic optimization algorithms follow noisy gradients to reach the optimum of an objective function. The gradient with respect to the state transition parameters is already derived in Kingma & Welling (2019), which can be found in equations 27 and 26 of Appendix A. We also require to find the gradient of $\boldsymbol{Q}$ with respect to $\boldsymbol{\Omega}$. This gradient is defined by

$$\nabla_{\boldsymbol{\Omega}} \boldsymbol{Q}(\boldsymbol{\theta}|\boldsymbol{\theta}^{(r)}) \simeq \frac{1}{D} \sum_{d=1}^{D} \sum_{k=1}^{K} \nabla_{\boldsymbol{\Omega}} \log p(\hat{\boldsymbol{x}}_k^{(d)}|\boldsymbol{y}_k, \boldsymbol{h}_k; \boldsymbol{\Omega}) + \lambda \nabla_{\boldsymbol{\Omega}} \log p(\hat{\boldsymbol{x}}_k^{(d)}|\boldsymbol{h}_k; \boldsymbol{\Omega}), \tag{18}$$

where $\{\hat{\boldsymbol{x}}_k^{(d)}\}_{k=1}^{k=K}$ is dth state trajectory path derived from the smoothed posterior with parameter set $\boldsymbol{\theta}^{(r)}$, see more details in Appendix A. Given equation18, we can optimize the $\boldsymbol{\Omega}$ using a stochastic back-propagation similar to the ones proposed for variational-autoencoder models Kingma & Welling (2013).
Algorithm 1 presents the learning steps for the D4 using stochastic optimization. In developing the learning algorithm, we assumed that $\boldsymbol{x}_k$ is unobserved, when $\boldsymbol{x}_k$ is known, the learning procedure follows the same steps, whilst the expectation, $\mathbb{E}_K$, is replaced by the state values constructing the trajectory of the state.

3 EXPERIMENTS

We demonstrate utility of the D4 by testing its different modeling steps on simulated data created for non-linear dynamical systems with high-dimensional observations. The simulated experiments are 1) Langevin dynamics with point-process observations, 2) Lorenz attractor with point-process observations, and 3) 1-D random walk model with with a non-linear mapping onto 20 dimensional space as the observed signal (results fort this example are shown Appendix B). We selected these experiments to cover diverse non-linear dynamical systems with physically meaningful state variables and different types of observations and noise processes. We show that the D4 accurately estimate the non-linear latent dynamics. We also apply the D4 on a decoding problem using the rat hippocampus data while the rat forages a W-shaped maze for food Rezaei et al. (2021). For this problem, we compared D4's performance with the current state-of-the-art solutions including traditional point-process SSMs Truccolo et al. (2005) and GRU-RNNs Chung et al. (2014). In our assessment of different decoder models' performance, we consider mean squared error (MSE), mean absolute error (MAE) Chai & Draxler (2014), correlation coefficients (CC), and the 95% highest posterior density region (HPD) metrics Yao & Lindsay (2009).

3.1 LANGEVIN DYNAMICS WITH POINT-PROCESS OBSERVATIONS

Langevin dynamics is used to describe the acceleration of a particle in a liquid. Here we consider one-dimensional harmonic oscillators, with potential function $U(q) = \frac{Kq^2}{2}, q \in \mathbb{R}$ which is a standard test case for Langevin dynamics Leimkuhler & Matthews (2013),

$$dq = M^{-1}pdt, \;\; dp = -\nabla U(q)dt - \gamma pdt + \sigma M^{0.5}dW \tag{19}$$

where $q, p \in \mathbb{R}^{3N}$ are vectors of instantaneous position and momenta, respectively, $W = W(t)$ is a vector of $3N$ independent Wiener processes, $\gamma > 0$ is a free (scalar) parameter, and M is a constant diagonal mass matrix. We simulated 100-dimensional spiking time-series using a point process intensity model, see Appendix C. Figure1.A shows the simulated spiking data. For the D4, we assume the prediction process is a Gaussian process, $p(x_k|\boldsymbol{y}_k, \boldsymbol{h}_k, \boldsymbol{\Omega}) \sim N(\mu_{\boldsymbol{\Omega}}(\boldsymbol{y}_k, \boldsymbol{h}_k), \sigma_{\boldsymbol{\Omega}}(\boldsymbol{y}_k, \boldsymbol{h}_k))$. $\mu_{\boldsymbol{\Omega}}(\boldsymbol{y}_k, \boldsymbol{h}_k)$ and $\sigma_{\boldsymbol{\Omega}}(\boldsymbol{y}_k, \boldsymbol{h}_k)$ are the mean and standard deviation of the Gaussian predictor which are nonlinear functions of the current, $\boldsymbol{y}_k$, and a history of observed spiking data, $\boldsymbol{h}_k$. Given the observed spiking data, Algorithm 1 simultaneously updates the state transition and prediction process parameters that maximize the $\boldsymbol{Q}$ function given $\boldsymbol{y}_{1:K}$. Figure 2.C shows the performance of D4, with the optimal hyper-parameter λ identified in Figure 2.B, along with the DDD and DKF. The D4 successfully inferred the latent state trajectory as most of the points in the trajectory are covered by the 95% HPD of the D4's predictions. The D4 gives a higher performance measures, meaning that it gives a better estimation of the underlying states.

3.2 LORENZ ATTRACTOR WITH POINT-PROCESS OBSERVATIONS

Lorenz attractor is a chaotic system Afraimovich et al. (1977) with its nonlinear dynamics are defined by,

$$\dot{x_1} = \sigma(x_2 - x_1) + \epsilon_1, \;\; \dot{x_2} = x_1(\rho - x_3) - x_2 + \epsilon_1, \;\; \dot{x_3} = x_1x_2 - \beta x_3 + \epsilon_1. \tag{20}$$

where the $\{\epsilon_1, \epsilon_2, \epsilon_3\}$ are Gaussian white noises. The model were set to $\{\sigma = 10, \rho = 28, \beta = 8/3\}$ to have a complex trajectory. We simulated 100 spiking data using a point process intensity model, see AppendixC, Figure1.A shows the spiking data.
Here, we assume the prediction process is a Gaussian process, $p(x_k|\boldsymbol{y}_k, \boldsymbol{h}_k, \boldsymbol{\Omega}) \sim$

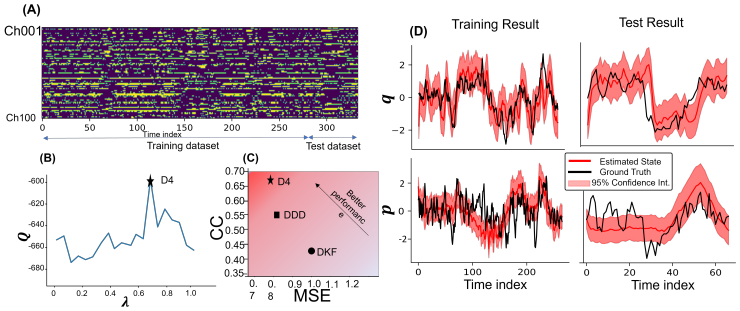

Figure 2: The D4 inference result for the Langevian system. A) 100 simulated spiking observations based on equations 19. B) λ hyperparameter optimization for equation 17. C) The D4's performance comparison with DDD and DKF. D) The D4 inferred trajectory for the Langevian latent states.

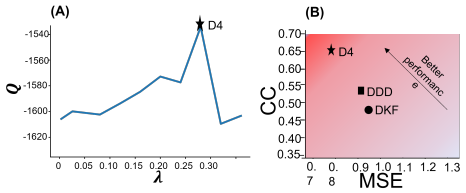

Figure 3: Th D4's performance compared with DDD and DKF models in Lorenz Attractor problem. A) λ hyper-parameter optimization for D4, equation 17. B) The D4's inference performance compared with DDD and DKF.

$N(\mu_{\boldsymbol{\Omega}}(\boldsymbol{y}_k, \boldsymbol{h}_k), \sigma_{\boldsymbol{\Omega}}(\boldsymbol{y}_k, \boldsymbol{h}_k))$, where $\mu_{\boldsymbol{\Omega}}(\boldsymbol{y}_k, \boldsymbol{h}_k)$ and $\sigma_{\boldsymbol{\Omega}}(\boldsymbol{y}_k, \boldsymbol{h}_k)$ are the mean and standard deviation of the Gaussian noise, both nonlinear functions of the current, $\boldsymbol{y}_k$, and the history, $\boldsymbol{h}_k$. Figure 3.B shows the D4, with optimal value of λ hyper-parameter identified in Figure3.A, and inference performance compared to DDD and DKF models. Similar to the D4's results for Langevin dynamics, shown in Figure 1.C, the D4 accurately inferred the latent state trajectories for the Lorenz system with a better performance compared to DDD and DKF.

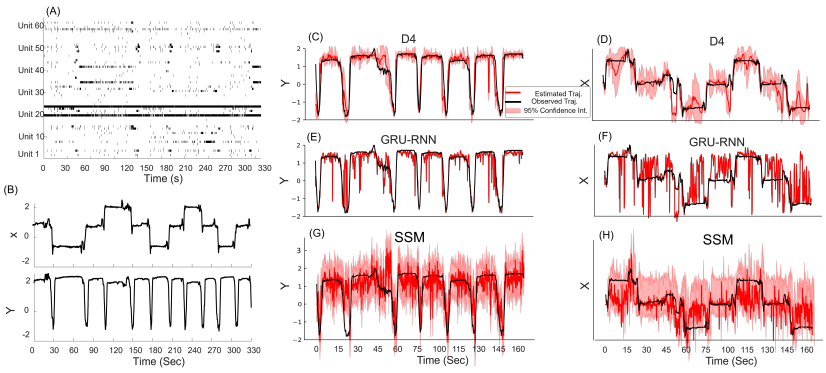

Figure 4: Decoding result for the rat movement trajectory using D4, SSM, and GRU-RNN. A) Raster plot of the 62 place cells spiking activity for a 330 seconds period. B) The rat position during time period shown in A. The rat position is measured using a camera installed over the maze. C-H) Results of D4, SSM, and GRU-RNN for the test and training dataset.

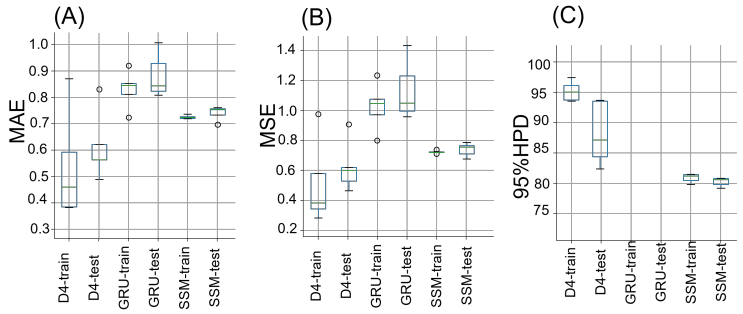

Figure 5: The D4, SSM, and GRU-RNN performance decoding performance in the rat dataset. I) The MAE, J) The MSE, K) The 95% highest posterior density (HPD) performance results for D4, SSM, and GRU-RNN for the 2D dimensions.

3.3 Decoding rat hippocampus data

In this problem, we seek to decode the 2-D movement trajectory of a rat traversing through a W-shaped maze from the ensemble spiking activity of 62 hippocampal place cells. The neural data are recorded from 62 place cells in the CA1 and CA2 regions of the hippocampus brain area of a rat, more details are available in AppendixD. Figure4.A shows the spiking activity of these 62 units, $\boldsymbol{y}_k \in \mathbb{R}^{62}$. Here, the state variable $\boldsymbol{c}_k$ represents the rat coordinates in the 2D spaces, see Figure4.B. For the state process, we use a 2-dimensional random walk with a multivariate normal distribution as the state process. The prediction process is a nonlinear multi-variate regression model for 2D-coordinates where the mean and variance are a function of the current spiking activity of 62 cells and their spiking history. Along with the D4, we also build point-process SSM, explained in Truccolo et al. (2005) and GRU-RNN Chung et al. (2014) models, see Figure 4 for decoding results of these models.
Figure5 shows performance of different decoding solutions. Among these models, the D4 gives a significantly better 95% HPD compared to the SSM model. Note that this comparison is not possible for GRU-RNN, as its output is deterministic. The D4 model outperforms both SSM and GRU-RNN in the performance measures. The D4 performance precedes other models given it optimally combines the information carried by spiking activity and state model of the movement at different temporal scale. These results are aligned with the physiology of neurons where their spiking activity is dependent on their previous spikes, a phenomenon which is not necessarily addressed in SSMs Truccolo et al. (2005). The other attribute of D4 is its robustness to the overfitting issue. As it shown in Figure5, the difference in D4 decoding performance for training and test data are significantly less than those in GRU-RNN model. This result suggests that we can utilize the D4 in modeling problems when the size of the data is limited, whereas, RNNs will faces over-fitting issues.

4 Conclusion

In this research, we introduced a new modeling framework that extends the utility of SSMs in the analysis of high-dimensional time-series data. We call the solution the Deep Direct Discriminative Decoder, or D4, model. For this model, we demonstrated its solution in both simulated and real datasets, where its performance precedes RNN and SSM models. The D4 is a variant of SSMs, in which the conditional distribution of the observed signals is replaced by a discriminative process. The D4 inherits attributes of SSM, while it provides a higher expressive power in its prediction of the state variables. In sum, the D4 inherits the advantages of both SSM and DNNs. The D4 can incorporate any information from the history of observed data at different time scales in computing the estimate of this state process. D4 is fundamentally different from SSMs, where the information at only two time scales: a) fast, which is carried by the observation, and b) slow, defined by the state process, are combined in estimating the state process. We expect the D4 performance to be better than SSM and DNNs, as a D4 without the state process turns to a DNN, and the D4 without the history term will (potentially) correspond to the DKF and/or SSM. Here, we showed that the D4 performance in the simulated data with high-dimensional observation and non-linear state processes

preceded the state-of-art solutions including SSM and RNNs. The D4 decoding performance in the neural data is significantly better than SSM and RNNs, showing its applicability in the analysis of complex and high-dimensional time-series data. In the neural data, the D4 performance shows less discrepancy in the test and training data, pointing to its robust prediction. The D4 modeling pipeline integrated with the training solution built in this research provides a scalable and versatile modeling solution applicable in the analysis of dynamical high-dimensional time-series data, where a high level of interpretability and performance are needed.

REFERENCES

Valentin S Afraimovich, VV Bykov, and Leonid P Shilnikov. On the origin and structure of the lorenz attractor. In *Akademiia Nauk SSSR Doklady*, volume 234, pp. 336–339, 1977.

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

Michael C Burkhart, David M Brandman, Brian Franco, Leigh R Hochberg, and Matthew T Harrison. The discriminative kalman filter for bayesian filtering with nonlinear and nongaussian observation models. *Neural Computation*, 32(5):969–1017, 2020.

Tianfeng Chai and Roland R Draxler. Root mean square error (rmse) or mean absolute error (mae)?–arguments against avoiding rmse in the literature. *Geoscientific model development*, 7(3): 1247–1250, 2014.

Zhe Chen et al. Bayesian filtering: From kalman filters to particle filters, and beyond. *Statistics*, 182 (1):1–69, 2003.

Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555*, 2014.

Arnaud Doucet, Adam M Johansen, et al. A tutorial on particle filtering and smoothing: Fifteen years later. *Handbook of nonlinear filtering*, 12(656-704):3, 2009.

James Durbin and Siem Jan Koopman. *Time series analysis by state space methods*, volume 38. OUP Oxford, 2012.

Uri T Eden, Loren M Frank, Riccardo Barbieri, Victor Solo, and Emery N Brown. Dynamic analysis of neural encoding by point process adaptive filtering. *Neural computation*, 16(5):971–998, 2004.

Joshua Glaser, Matthew Whiteway, John P Cunningham, Liam Paninski, and Scott Linderman. Recurrent switching dynamical systems models for multiple interacting neural populations. *Advances in neural information processing systems*, 33:14867–14878, 2020.

Mohinder S Grewal, Vinson D Henderson, and Randy S Miyasako. Application of kalman filtering to the calibration and alignment of inertial navigation systems. In *29th IEEE Conference on Decision and Control*, pp. 3325–3334. IEEE, 1990.

Irina Higgins, Loic Matthey, Arka Pal, Christopher Burgess, Xavier Glorot, Matthew Botvinick, Shakir Mohamed, and Alexander Lerchner. beta-vae: Learning basic visual concepts with a constrained variational framework. 2016.

David A Kenny and Charles M Judd. Estimating the nonlinear and interactive effects of latent variables. *Psychological bulletin*, 96(1):201, 1984.

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

Diederik P Kingma and Max Welling. An introduction to variational autoencoders. *arXiv preprint arXiv:1906.02691*, 2019.

Evgeni Kiriy and Martin Buehler. Three-state extended kalman filter for mobile robot localization. *McGill University., Montreal, Canada, Tech. Rep. TR-CIM*, 5:23, 2002.

Genshiro Kitagawa. Monte carlo filter and smoother for non-gaussian nonlinear state space models. *Journal of computational and graphical statistics*, 5(1):1–25, 1996.

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

Benedict Leimkuhler and Charles Matthews. Robust and efficient configurational molecular sampling via langevin dynamics. *The Journal of chemical physics*, 138(17):05B601_1, 2013.

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

In Jae Myung. Tutorial on maximum likelihood estimation. *Journal of mathematical Psychology*, 47 (1):90–100, 2003.

Liam Paninski, Yashar Ahmadian, Daniel Gil Ferreira, Shinsuke Koyama, Kamiar Rahnama Rad, Michael Vidne, Joshua Vogelstein, and Wei Wu. A new look at state-space models for neural data. *Journal of computational neuroscience*, 29(1):107–126, 2010.

Syama Sundar Rangapuram, Matthias W Seeger, Jan Gasthaus, Lorenzo Stella, Yuyang Wang, and Tim Januschowski. Deep state space models for time series forecasting. *Advances in neural information processing systems*, 31, 2018.

Mohammad R Rezaei, Anna K Gillespie, Jennifer A Guidera, Behzad Nazari, Saeid Sadri, Loren M Frank, Uri T Eden, and Ali Yousefi. A comparison study of point-process filter and deep learning performance in estimating rat position using an ensemble of place cells. In *2018 40th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*, pp. 4732–4735. IEEE, 2018.

Mohammad R. Rezaei, Alex E. Hadjinicolaou, Sydney S. Cash, Uri T. Eden, and Ali Yousefi. Direct Discriminative Decoder Models for Analysis of High-Dimensional Dynamical Neural Data. *Neural Computation*, pp. 1–36, 03 2022. ISSN 0899-7667. doi: 10.1162/neco_a_01491. URL `https://doi.org/10.1162/neco_a_01491`.

Mohammad Reza Rezaei, Kensuke Arai, Loren M Frank, Uri T Eden, and Ali Yousefi. Real-time point process filter for multidimensional decoding problems using mixture models. *Journal of Neuroscience Methods*, 348:109006, 2021.

Danilo Rezende and Shakir Mohamed. Variational inference with normalizing flows. In *International conference on machine learning*, pp. 1530–1538. PMLR, 2015.

Wilson Truccolo, Uri T Eden, Matthew R Fellows, John P Donoghue, and Emery N Brown. A point process framework for relating neural spiking activity to spiking history, neural ensemble, and extrinsic covariate effects. *Journal of neurophysiology*, 93(2):1074–1089, 2005.

Weixin Yao and Bruce G Lindsay. Bayesian mixture labeling by highest posterior density. *Journal of the American Statistical Association*, 104(486):758–767, 2009.

Ali Yousefi, Angelique C Paulk, Thilo Deckersbach, Darin D Dougherty, Emad N Eskandar, Alik S Widge, and Uri T Eden. Cognitive state prediction using an em algorithm applied to gamma distributed data. In *2015 37th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*, pp. 7819–7824. IEEE, 2015.

Ali Yousefi, Anna K Gillespie, Jennifer A Guidera, Mattias Karlsson, Loren M Frank, and Uri T Eden. Efficient decoding of multi-dimensional signals from population spiking activity using a gaussian mixture particle filter. *IEEE Transactions on Biomedical Engineering*, 66(12):3486–3498, 2019.

Fan Zhang, Junwei Cao, Samee U Khan, Keqin Li, and Kai Hwang. A task-level adaptive mapreduce framework for real-time streaming data in healthcare applications. *Future generation computer systems*, 43:149–160, 2015.

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

A CALCULATING Q FUNCTION GRADIENTS

First we start with calculation of the gradients of the $\boldsymbol{Q}(\boldsymbol{\theta}|\boldsymbol{\theta}^{(r)})$ with respect to $\boldsymbol{\Omega}$ which is defined by

$$\nabla_{\boldsymbol{\Omega}}\boldsymbol{Q}(\boldsymbol{\theta}|\boldsymbol{\theta}^{(r)}) = \nabla_{\boldsymbol{\Omega}}(\mathbb{E}_K[\log\ p(\boldsymbol{x}_0;\boldsymbol{\omega}_0) + \sum_{k=1}^{K}\log\ p(\boldsymbol{x}_k|\boldsymbol{x}_{k-1};\boldsymbol{\omega}) + \sum_{k=1}^{K}\log\ p(\boldsymbol{x}_k|\boldsymbol{y}_k,\boldsymbol{h}_k;\boldsymbol{\Omega})]$$
$$-\lambda\sum_{k=1}^{K}\mathbb{KL}(p(\boldsymbol{x}_k|\boldsymbol{y}_{1:K},\boldsymbol{\theta}^{(r)})\parallel p(\boldsymbol{x}_k|\boldsymbol{h}_k;\boldsymbol{\Omega})))$$

We can change the order of expectation - $\mathbb{E}_K$- and gradient; thus, we have

$$\nabla_{\boldsymbol{\Omega}}\boldsymbol{Q}(\boldsymbol{\theta}|\boldsymbol{\theta}^{(r)}) = \mathbb{E}_K[\nabla_{\boldsymbol{\Omega}}(\log\ p(\boldsymbol{x}_0;\boldsymbol{\omega}_0))+\sum_{k=1}^{K}\nabla_{\boldsymbol{\Omega}}(\log\ p(\boldsymbol{x}_k|\boldsymbol{x}_{k-1};\boldsymbol{\omega}))+\sum_{k=1}^{K}\nabla_{\boldsymbol{\Omega}}(\log\ p(\boldsymbol{x}_k|\boldsymbol{y}_k,\boldsymbol{h}_k;\boldsymbol{\Omega}))]$$
$$-\lambda\sum_{k=1}^{K}\nabla_{\boldsymbol{\Omega}}(\mathbb{KL}(p(\boldsymbol{x}_k|\boldsymbol{y}_{1:K},\boldsymbol{\theta}^{(r)})\parallel p(\boldsymbol{x}_k|\boldsymbol{h}_k;\boldsymbol{\Omega})))$$

The gradient of the first two term with respect to $\boldsymbol{\Omega}$ is zero; as a result, we can rewrite the gradient as

$$\nabla_{\boldsymbol{\Omega}}\boldsymbol{Q}(\boldsymbol{\theta}|\boldsymbol{\theta}^{(r)}) = \mathbb{E}_K[\sum_{k=1}^{K}\nabla_{\boldsymbol{\Omega}}(\log\ p(\boldsymbol{x}_k|\boldsymbol{y}_k,\boldsymbol{h}_k;\boldsymbol{\Omega}))]$$
$$-\lambda\sum_{k=1}^{K}\nabla_{\boldsymbol{\Omega}}(\mathbb{KL}(p(\boldsymbol{x}_k|\boldsymbol{y}_{1:K},\boldsymbol{\theta}^{(r)})\parallel p(\boldsymbol{x}_k|\boldsymbol{h}_k;\boldsymbol{\Omega}))) \tag{21}$$

The $\mathbb{KL}(.)$ term can be rewritten as Blei et al. (2017)

$$\mathbb{KL}(p(\boldsymbol{x}_k|\boldsymbol{y}_{1:K},\boldsymbol{\theta}^{(r)})\parallel p(\boldsymbol{x}_k|\boldsymbol{h}_k;\boldsymbol{\Omega}))$$
$$=\mathbb{E}_{p(\boldsymbol{x}_k|\boldsymbol{y}_{1:K},\boldsymbol{\theta}^{(r)})}[\log p(\boldsymbol{x}_k|\boldsymbol{y}_{1:K},\boldsymbol{\theta}^{(r)})]-\mathbb{E}_{p(\boldsymbol{x}_k|\boldsymbol{y}_{1:K},\boldsymbol{\theta}^{(r)})}[\log p(\boldsymbol{x}_k|\boldsymbol{h}_k;\boldsymbol{\Omega})] \tag{22}$$

By replacing equation 22 inside equation 21, we derive

$$\nabla_{\boldsymbol{\Omega}}\boldsymbol{Q}(\boldsymbol{\theta}|\boldsymbol{\theta}^{(r)}) = \mathbb{E}_K[\sum_{k=1}^{K}\nabla_{\boldsymbol{\Omega}}(\log\ p(\boldsymbol{x}_k|\boldsymbol{y}_k,\boldsymbol{h}_k;\boldsymbol{\Omega}))]$$
$$-\lambda\sum_{k=1}^{K}\nabla_{\boldsymbol{\Omega}}(\mathbb{E}_{p(\boldsymbol{x}_k|\boldsymbol{y}_{1:K},\boldsymbol{\theta}^{(r)})}[\log p(\boldsymbol{x}_k|\boldsymbol{y}_{1:K},\boldsymbol{\theta}^{(r)})]-\mathbb{E}_{p(\boldsymbol{x}_k|\boldsymbol{y}_{1:K},\boldsymbol{\theta}^{(r)})}[\log p(\boldsymbol{x}_k|\boldsymbol{h}_k;\boldsymbol{\Omega})]) \tag{23}$$

The $\mathbb{E}_{p(\boldsymbol{x}_k|\boldsymbol{y}_{1:K},\boldsymbol{\theta}^{(r)})}[p(\boldsymbol{x}_k|\boldsymbol{y}_{1:K},\boldsymbol{\theta}^{(r)})]$ is not dependent on $\boldsymbol{\Omega}$; therefore, the $\nabla_{\boldsymbol{\Omega}}(\mathbb{E}_{p(\boldsymbol{x}_k|\boldsymbol{y}_{1:K},\boldsymbol{\theta}^{(r)})}[\log p(\boldsymbol{x}_k|\boldsymbol{y}_{1:K},\boldsymbol{\theta}^{(r)})]) = 0$. Now, we move the gradient inside the expectation term one more time which gives us

$$\nabla_{\boldsymbol{\Omega}}\boldsymbol{Q}(\boldsymbol{\theta}|\boldsymbol{\theta}^{(r)}) = \sum_{k=1}^{K}\mathbb{E}_{p(\boldsymbol{x}_k|\boldsymbol{y}_{1:K},\boldsymbol{\theta}^{(r)})}[\nabla_{\boldsymbol{\Omega}}(\log\ p(\boldsymbol{x}_k|\boldsymbol{y}_k,\boldsymbol{h}_k;\boldsymbol{\Omega})) + \lambda\nabla_{\boldsymbol{\Omega}}(\log p(\boldsymbol{x}_k|\boldsymbol{h}_k;\boldsymbol{\Omega}))] \tag{24}$$

Therefore, we can calculate $\nabla_{\boldsymbol{\Omega}\boldsymbol{Q}(\boldsymbol{\theta}|\boldsymbol{\theta}^{(r)})}$ by

$$\nabla_{\boldsymbol{\Omega}\boldsymbol{Q}(\boldsymbol{\theta}|\boldsymbol{\theta}^{(r)})} \simeq \frac{1}{D}\sum_{d=1}^{D}\sum_{k=1}^{K}\nabla_{\boldsymbol{\Omega}}\log p(\hat{\boldsymbol{x}}_k^{(d)}|\boldsymbol{y}_k,\boldsymbol{h}_k;\boldsymbol{\Omega}) + \lambda\nabla_{\boldsymbol{\Omega}}\log p(\hat{\boldsymbol{x}}_k^{(d)}|\boldsymbol{h}_k;\boldsymbol{\Omega}) \tag{25}$$

The equation 25 is a Monte Carlo estimator of the equation25, where $\{\hat{\boldsymbol{x}}_k^{(d)}\}_{k=1}^{k=K}$ is dth sample trajectory from the smoothed posterior with parameter set $\boldsymbol{\theta}^{(r)}$.
Similarly, the gradient of the $\boldsymbol{Q}(\boldsymbol{\theta}|\boldsymbol{\theta}^{(r)})$ with respect to $\boldsymbol{\omega}$ can be derived by

$$\nabla_{\boldsymbol{\omega}}\boldsymbol{Q}(\boldsymbol{\theta}|\boldsymbol{\theta}^{(r)}) \simeq \frac{1}{D}\sum_{d=1}^{D}\sum_{k=1}^{K}\nabla_{\boldsymbol{\Omega}}\log p(\hat{\boldsymbol{x}}_k^{(d)}|\hat{\boldsymbol{x}}_{k-1}^{(d)};\boldsymbol{\omega}) \tag{26}$$

Finally, the gradient of the $\boldsymbol{Q}(\boldsymbol{\theta}|\boldsymbol{\theta}^{(r)})$ with respect to $\boldsymbol{\omega}_0$ can be derived by

$$\nabla_{\boldsymbol{\omega}_0}\boldsymbol{Q}(\boldsymbol{\theta}|\boldsymbol{\theta}^{(r)}) \simeq \frac{1}{D}\sum_{d=1}^{D}\nabla_{\boldsymbol{\Omega}}\log p(\hat{\boldsymbol{x}}_k^{(d)}|\hat{\boldsymbol{x}}_{k-1}^{(d)};\boldsymbol{\omega}) \tag{27}$$

B PERFORMANCE COMPARISON IN 1D SIMULATION DATA WITH HIGH-DIMENSIAL OBSERVATIONS

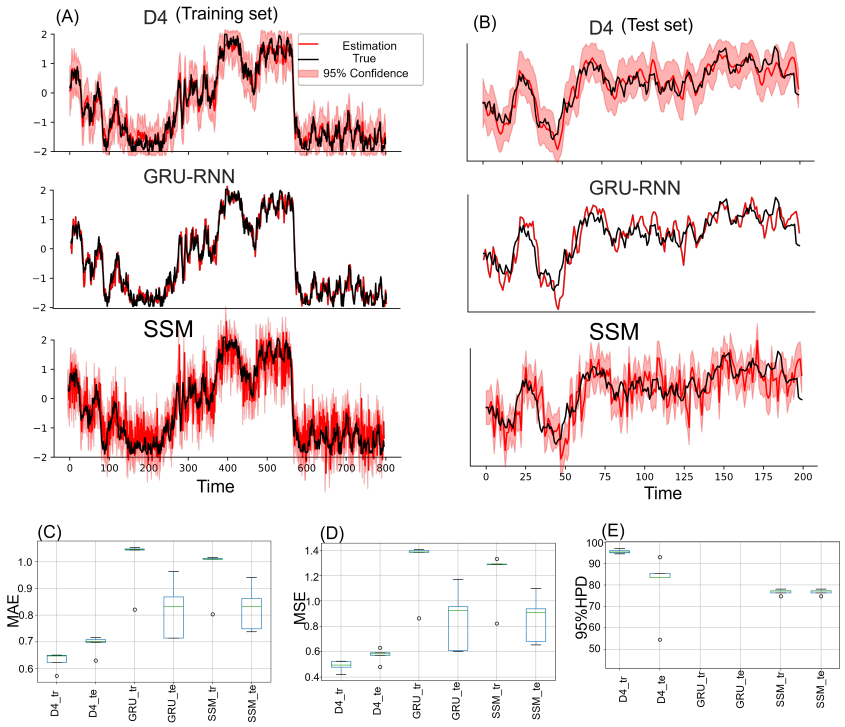

Figure 6: Decoding results for simulation dataset using D4, GRU-RNN, and SSM models. A) Decoding results for the training set. B) Decoding results for the test set. C) MAE. D) MSE. E) 95% HPD. Note that the GRU-RNN model has a deterministic output; as a result, there is no 95% HPD measure for the GRU-RNN model.

We generate the simulation dataset by assuming the state of interest, x_k, is a one-dimensional random walk model which is defined by $P(x_k|x_{k-1}) \sim N(ax_{k-1}+b, \sigma_x^2)$, where a is the state transition and b is the bias coefficient for the random walk with a normal additive noise with a standard deviation of σ_x. Using the state, we then generate a 20-dimensional temporal signal with a conditional distribution defined by

$$P(\boldsymbol{y}_k|x_k) \sim N(\boldsymbol{g}(x_k, x_{k-1}, ..., x_{k-l_m}), \boldsymbol{\Sigma}_s) \tag{28}$$

where $\boldsymbol{g}(.)$ is a 20-dimensional vector of non-linear functions, like tanh, and cosine, with an argument defined by a subset of state processes at the current and previous time points. The l_m is the maximum length of data points used in $g(.)$ function. For each channel of data, we pick a l uniformly from 0 to l_m randomly, which is used for data generation. For example, in our simulation, the g_1 function corresponding to the first channel of observation $\boldsymbol{g}$, is a $\tanh(.)$ with the argument $x_k + 0.8 * x_{k-1} + 0.6 * x_{k-2} + 0.4 * x_{k-3} + 0.2 * x_{k-4}$. $\boldsymbol{\Sigma}_s$ is the stationary covariance matrix with size 20×20. $\boldsymbol{\Sigma}_s$'s non-diagonal terms are non-zeros, implying observations across channels are correlated as well. In our simulation, we picked $a = .98$, $b = 0$, and $\sigma_x = 0.1$. We generate the data for 1000 data points. The observation signal in this simulation has a complex dynamics while the undelying state is low dimensional; this setting was purposefully picked to demonstrate D4 prediction power and build a clear comparison across models.

We tested SSM and GRU-RNN model decoding performance along with the D4. For the training, we assumed the state process is observed. We consider the prediction process is a Gaussian process, $p(x_k|\boldsymbol{y}_k, \boldsymbol{h}_k, \boldsymbol{\Omega}) \sim N(\mu_{\boldsymbol{\Omega}}(\boldsymbol{y}_k, \boldsymbol{h}_k), \sigma_{\boldsymbol{\Omega}}(\boldsymbol{y}_k, \boldsymbol{h}_k))$. $\mu_{\boldsymbol{\Omega}}(\boldsymbol{y}_k, \boldsymbol{h}_k)$ and $\sigma_{\boldsymbol{\Omega}}(\boldsymbol{y}_k, \boldsymbol{h}_k)$ are the mean and standard deviation of the Gaussian predictor which both are nonlinear functions of the current, $\boldsymbol{y}_k$, and the history of observed spiking data, $\boldsymbol{h}_k$. This is because we needed the state for GRU-RNN training, and with the values of the state known, the training of the SSM becomes simple. For SSM

training, we can find the state and observation process parameters using the MLE technique, if the state trajectory is known. Figure6 shows the D4, GRU-RNN, and SSMs decoding results on this data. The D4 model MAE and MSE measures outperform both SSM and GRU-RNN models. The D4 model also gives even a better 95% HPD compared to the SSM model.

C POINT-PROCESS OBSERVATIONS FOR LORENZ DYNAMICAL SYSTEM

We generate simulated M channels of spiking data using a point process intensity model that is governed by a nonlinear mapping of the state values $\boldsymbol{X}$, where the conditional intensity for each channel of the data is a function of the state process defined by a problem, such as Langevin problem, $\boldsymbol{X} = \{p, q\}$ and Lorenz attractor problem, $\boldsymbol{X} = \{x_1, x_2, x_3\}$, as

$$\lambda_j(\boldsymbol{X}) = \exp[a_j - \sum_{x_k \in \boldsymbol{X}} \frac{(x_k - \mu_{j,x_k})^2}{2\sigma^2_{j,x_k}}], j = 1, ..., M \tag{29}$$

where μ_{j,x_k} and σ^2_{j,x_k} define the center and width for a hypothetical receptive field model of x_k, and a_j is the peak firing rates. The history-dependent terms for ith channel is defined by another intensity function as

$$\lambda_{j,H} = \sum_{s_n \in S_j} 1 - \exp(-\frac{(k - s_n)^2}{2\sigma^2_j}), \tag{30}$$

where S_j is the set containing all the spike times of jth channel. Therefore, the intensity function for jth point-process channel, $\hat{\lambda}_j$, is calculated by $\hat{\lambda}_j = \lambda_j * \lambda_{j,H}$.
μ_{j,x_k} are drawn from uniform distributions that cover the domain of the state values, $\sim U(mean(x_k) - 2 * std(x_k), mean(x_k) + 2 * std(x_k))$. $\{\sigma_{j,x_k}, \sigma_i\}$ are drawn from a uniform distribution, $\sim U(min_{sigma}, 1/M)$. a_j are drawn from a uniform distribution, $\sim U(min_{firing-rate}, max_{firing-rate})$.

D DETAILS ABOUT THE NEURAL DATA

In this example, we seek to decode the 2D movement trajectory of a rat traversing a W-shaped maze from the ensemble spiking activity of 62 hippocampal place cells Yousefi et al. (2019). The neural data were recorded from an ensemble of place cells in the CA1 and CA2 regions of the hippocampus of a single Long-Evans rat, aged approximately 6 months. The rat was trained to traverse between a home box and the outer arms to receive a liquid reward (condensed milk) at the reward locations. The spiking activity of these 62 units was detected offline by identifying events with peak-to-peak amplitudes above a threshold of 80 uV in at least one of the tetrode channels. The rat's position was measured using video tracking software and was used for training the models and as the ground truth for the decoded position. We used a 15-minute section of the experiment, with a time resolution of 33 milliseconds, to analyze multiple decoding methods. The first 82% of the recording (about 12.5 minutes) was used to train the discriminative and state process models and the remaining 18% (about 2.5 minutes) of data was used to test the model's decoding performance.