Bayesian Decoder Models with a Discriminative Observation Process

Mohammad R. Rezaei[1], Alex E. Hadjinicolaou[2], Sydney S. Cash[2], Uri T. Eden[3], Ali Yousefi[4*]

[1]Institute of Biomedical Engineering, University of Toronto, Toronto, ON, Canada
[2]Department of Neurology, Massachusetts General Hospital, and Harvard Medical School, Boston, MA
[3]Department of Mathematics and Statistics, Boston University, Boston, MA
[4]Department of Computer Science, Worcester Polytechnic Institute (WPI), Worcester, MA

Keywords: Direct Decoder, Dynamical Auto-encoder, Multi-dimensional Neural Decoder, General Rescaling Theorem

Abstract

The Bayesian state-space neural encoder-decoder modeling framework is an established solution to reveal how changes in brain dynamics encode physiological covariates like movement or cognition. Although the framework is increasingly being applied to progress the field of neuroscience, its application to modeling high-dimensional neural data continues to be a challenge. Here, we propose a novel solution that avoids the complexity of encoder models that characterize high-dimensional data as a function of the underlying state processes. We build a discriminative model to estimate state processes as a function of current and previous observations of neural activity. We then develop the filter and parameter estimation solutions for this new class of state-space modeling framework called the "direct decoder" model. We applied the model to decode movement trajectories of a rat in a W-shaped maze from the ensemble spiking activity of place cells and achieve comparable performance to modern decoding solutions, without needing an encoding step in the model development. We further demonstrate how a dynamical auto-encoder can be built using the direct decoder model; where the underlying state process links the high-dimensional neural activity to the behavioral readout. We applied the dynamical auto-encoder model in estimating the intention to verbally communicate of an epileptic participant and their companions. The result shows that the dynamical auto-encoder can optimally estimate the low-dimensional dynamical manifold which represents the relationship between the brain and behavior.

1 Introduction

The rapid development of neural recording technologies over the last few decades has enabled the simultaneous recording of neural activity from an ever-increasing number of brain regions. For research groups interested in relating brain activity to higher-level processes, these data are often recorded during some sorts of experimental tasks, together with behavioral or cognitive observations that are influenced by the task [1, 2]. The higher dimension and multi-modality of these data necessitate the development of analytical solutions capable of making statistically robust inferences about the underlying brain dynamics and their relationship to the observed correlates [3, 4]. A wide variety of statistical and machine learning techniques, broadly known as neural encoder-decoder models, have been developed to address this particular type of problem [5-7]. Such models are built in two stages: first, a neural encoder model builds the conditional distribution of observed neural data given the underlying neural correlates (such as movement or cognitive state), and then, newly observed neural data are decoded to estimate those correlates by applying Bayes' theorem to the encoder model [8-13]. Although traditional neural encoder-decoder models have been successfully applied to gain insights from low-dimensional data [8, 14, 15], they face multiple modeling challenges when applied to high-dimensional data. One of such problems appears in the encoding step, in which the conditional joint distribution of neural data is built. Due to the large dimension of data, it is hard to properly characterize this distribution. Proposed solutions for this distribution are mainly built upon naive assumptions such as the conditional independence of the individual neural data given the correlates. Even with this assumption, it is not always possible to characterize the distributions of neural data and associated noise components (e.g., they may not be stationary), which introduces further complications and difficulties in building the encoder model. The fact that the neural correlates generally have a lower dimension compared to the neural data might help us to address some of the challenges tied to the classical encoder-decoder modeling methodologies [16-19].

In this work, we propose a Bayesian filter solution for the decoder model in which we build the model directly from the neural data ensemble, as opposed to first formulating the encoder model. A specific variation of this modeling approach has been recently proposed by Harrison et al. [20], in which the decoder model is defined as a function of the current-time neural observation. Given this assumption, the Bayes filter solution is derived for the steady-state condition. Here, we introduce a more general framework of this modeling approach, in which the decoder model is defined as a function of current and previous neural data. In our proposed framework, the filter solution accounts for time variability present in the observed data; in addition to that, we derive the maximum likelihood (ML) estimation of the model parameters using a revised expectation-maximization (EM) technique [21].

Recently, new techniques in machine learning like deep neural networks (DNNs) has been used in neuroscience data analysis to address a similar class of decoding and inference problems. DNNs are used to characterize the direct input and output relationship between neural activity and physiological or neurological states [11, 12, 22]. DNNs are flexible models that are able to extract information from high dimensional and complex data. Regardless of the extensive utilization of DNNs in neural encoding/decoding problems, they still have significant pitfalls like lack of generalizability and interpretation [23]. Many techniques are used to address these issues like using dropout [24], or regularization [25]; they even go further by making DNNs' parameters stochastic by assigning a probability distribution to each parameter of the DNNs, known as Bayesian DNNs (B-DNNs) [26, 27]. B-DNNs maintain their generalizability even when trained by a small number of data and prevent overfitting issues [28]. In this research, we incorporate DNNs and B-DNNs into our framework to add more flexibility and generalizability to the direct decoder model to capture complex dynamics present in neural data.

We apply our modeling framework to decode the 2-D trajectories of a rat moving through a W-shaped maze from the ensemble spiking activity of place cells; our proposed methodology demonstrates decoding results that are comparable to the state-of-art models, a point-process encoder-decoder model [29]. It is worth emphasizing that no encoder model is being built in our proposed modeling framework.

Our proposed Bayesian filter solution can be applied to a broader class of neural encoding-decoding problems, in which the connection between brain dynamics and neural correlates are defined through a low-dimensional dynamical manifold [19, 29-31]. For instance, when behavioral readouts are used as the correlates, the manifold will represent the cognitive states that underlie these behaviors. To address these class of problems, we propose a modeling solution in which a behavioral encoder model is used together with the direct neural decoder model to find the dynamical manifold linking behavior and the underlying neural activity. The proposed encoder-decoder model can be viewed as a dynamical auto-encoder model with the cognitive states as the latent manifold, and the behavior and neural data as different measures of the same dynamical latent structure. We conclude our findings with an application of this solution to a novel decoding problem, in which we seek to decode the communicative intentions of an epileptic study participant (with electrodes implanted for clinical purposes) from their neural activity, by modeling communicative intent as the underlying cognitive state. Whilst we demonstrate our modeling result, we compare our proposed modeling solution with the state-of-art neural decoding solutions including exact point-process [8, 15, 19, 29, 30], generalized linear models [9, 32], and DNNs/B-DNNs [11, 12, 22] .

2 Problem Formulation

Here, we begin by formulating the direct decoder (D-D) model using a discriminative observation process. We derive the Bayesian filter and parameter estimation solution for this model and expand it to a more generalized form of a *dynamical auto-encoder* model. For the D-D model, we propose a revised EM [21] algorithm that helps us to find a maximum likelihood estimate of the dynamical auto-encoder model-free parameters.

In the state-space modeling solution, care must be taken to build an accurate model of the observation process. We focus on the class of problems for which the dimensionality of neural observations is much larger than the number of state processes, an appropriate constraint for the wide range of problems in neuroscience that deal with multi-electrode neural recordings [33-35].

2.A Direct Decoder Model

Let us assume we have K observations from $k = 1$ to K. Let's assume $\boldsymbol{x}_\mathrm{k}$ represents the state (latent process, or underlying cognitive process) at time index k, and $\boldsymbol{s}_\mathrm{k}$ is the observed neural activity (observation process) at the same time index k. We define the history term $\boldsymbol{h}_\mathrm{k}$ as the subset of previous neural observations, $\boldsymbol{h}_\mathrm{k} \subset \{\boldsymbol{s}_1, \cdots, \boldsymbol{s}_{\mathrm{k}-1}\}$. As with a Bayes filter solution, our objective is to estimate

$$p(\boldsymbol{x}_\mathrm{k}|\boldsymbol{s}_{1:\mathrm{k}}) \tag{1}$$

which is the posterior distribution of $\boldsymbol{x}_\mathrm{k}$ given observed neural activity till time k.

Using a recursive filter solution [36], the filter update rule at time index k is defined by

$$p(\boldsymbol{x}_\mathrm{k}|\boldsymbol{s}_{1:\mathrm{k}}) \propto p(\boldsymbol{s}_\mathrm{k}|\boldsymbol{s}_{1:\mathrm{k}-1}, \boldsymbol{x}_\mathrm{k}) \times p(\boldsymbol{x}_\mathrm{k}|\boldsymbol{s}_{1:\mathrm{k}-1}) = p(\boldsymbol{s}_\mathrm{k}|\boldsymbol{h}_\mathrm{k}, \boldsymbol{x}_\mathrm{k}) \times p(\boldsymbol{x}_\mathrm{k}|\boldsymbol{s}_{1:\mathrm{k}-1}) \tag{2}$$

Given the definition of the history term, we can rewrite the filter update rule as

$$p(\boldsymbol{x}_\mathrm{k}|\boldsymbol{s}_{1:\mathrm{k}}) \propto \frac{p(\boldsymbol{x}_\mathrm{k}|\boldsymbol{h}_\mathrm{k}, \boldsymbol{s}_\mathrm{k})}{p(\boldsymbol{x}_\mathrm{k}|\boldsymbol{h}_\mathrm{k})} \times p(\boldsymbol{x}_\mathrm{k}|\boldsymbol{s}_{1:\mathrm{k}-1}) = \frac{p(\boldsymbol{x}_\mathrm{k}|\boldsymbol{h}_\mathrm{k}, \boldsymbol{s}_\mathrm{k})}{p(\boldsymbol{x}_\mathrm{k}|\boldsymbol{h}_{\mathrm{k}-1}, \boldsymbol{s}_{\mathrm{k}-1})} \times p(\boldsymbol{x}_\mathrm{k}|\boldsymbol{s}_{1:\mathrm{k}-1}) \tag{3}$$

Now, we can build a recursive solution for the update rule using the Chapman-Kolmogorov equation [37]:

$$p(\boldsymbol{x}_{\mathrm{k}}|\boldsymbol{s}_{1:\mathrm{k}}) \propto \frac{p(\boldsymbol{x}_{\mathrm{k}}|\boldsymbol{h}_{\mathrm{k}}, \boldsymbol{s}_{\mathrm{k}})}{\int p(\boldsymbol{x}_{\mathrm{k}}|\boldsymbol{x}_{\mathrm{k}-1})p(\boldsymbol{x}_{\mathrm{k}-1}|\boldsymbol{h}_{\mathrm{k}-1}, \boldsymbol{s}_{\mathrm{k}-1})d\boldsymbol{x}_{\mathrm{k}-1}} \times \int p(\boldsymbol{x}_{\mathrm{k}}|\boldsymbol{x}_{\mathrm{k}-1})p(\boldsymbol{x}_{\mathrm{k}-1}|\boldsymbol{s}_{1:\mathrm{k}-1})d\boldsymbol{x}_{\mathrm{k}-1} \tag{4}$$

The fraction term on the right-hand-side of equation (4) represents the likelihood function in the standard state-space model. It is the ratio of two likelihood functions for each value of $\boldsymbol{x}_{\mathrm{k}}$. The denominator defines the likelihood of $\boldsymbol{x}_{\mathrm{k}}$ given the history of observation until time k and the numerator is the likelihood of $\boldsymbol{x}_{\mathrm{k}}$ when considering the current observation together with the history term. This likelihood can be large or small depending on the information being carried by $\boldsymbol{s}_{\mathrm{k}}$ about $\boldsymbol{x}_{\mathrm{k}}$, which changes the posterior distribution of $\boldsymbol{x}_{\mathrm{k}}$ given the observation until time k. Note that the two Chapman-Kolmogorov equations in equation (4) define the likelihood of $\boldsymbol{x}_{\mathrm{k}}$ given two different history terms; these two likelihoods cancel each other if $\boldsymbol{h}_{\mathrm{k}} = \{ \boldsymbol{s}_1, \cdots, \boldsymbol{s}_{\mathrm{k}-1}\}$. In practice, modeling solutions to characterize $p(\boldsymbol{x}_{\mathrm{k}}|\boldsymbol{h}_{\mathrm{k}}, \boldsymbol{s}_{\mathrm{k}})$ are misspecified and $\boldsymbol{h}_{\mathrm{k}}$ will be limited to a subset of $\{ \boldsymbol{s}_1, \cdots, \boldsymbol{s}_{\mathrm{k}-1}\}$. For $p(\boldsymbol{x}_{\mathrm{k}}|\boldsymbol{h}_{\mathrm{k}}, \boldsymbol{s}_{\mathrm{k}})$ models with a long history term, the two likelihoods become similar, and the filter estimate is being mainly driven by $p(\boldsymbol{x}_{\mathrm{k}}|\boldsymbol{h}_{\mathrm{k}}, \boldsymbol{s}_{\mathrm{k}})$**.** When the history term is short, the dynamics of these likelihoods become important in the filter estimation.

As a part of the state-space model, we define the state transition process at time index k by

$$\boldsymbol{x}_{\mathrm{k}}|\boldsymbol{x}_{\mathrm{k}-1} \sim f(\boldsymbol{x}_{\mathrm{k}-1}; \boldsymbol{\theta}) \tag{5}$$

where$\boldsymbol{x}_{\mathrm{k}}$ is the cognitive state variable at time index k and $\boldsymbol{\theta}$ is the set of free parameters of the state equation.

Now, we can use the D-D filter solution derived in equation (4) to build the conditional distribution of state for the discriminative model, given the current observation and observation history. We call this the “prediction process”, which is described as

$$\boldsymbol{x}_{\mathrm{k}}|\boldsymbol{s}_{\mathrm{k}}, \boldsymbol{h}_{\mathrm{k}} \sim f(\boldsymbol{s}_{\mathrm{k}}, \boldsymbol{h}_{\mathrm{k}};\ \boldsymbol{\Omega}) \tag{6}$$

where $\boldsymbol{s}_{\mathrm{k}}$ and $\boldsymbol{h}_{\mathrm{k}}$ are the neural activity and history term at time index k, and $\boldsymbol{\Omega}$ is the set of free parameters for the discriminative model.

The D-D model (represented by the schematic in **Figure 1.A**) is comprised of the state and prediction processes defined by equations (5) and (6). With the prediction process, we no longer require an explicit description of the observation process, or the conditional distribution of the observation. The noise process in the prediction process in general is well-defined – note that the noise process for the state is already defined in the state process – and thus the direct decoder model can be easily constructed. The prediction process itself can be modeled using a variety of solutions including a generalized linear model (GLM) [16], a neural network [12], or a linear regression with regularization [38]. It can also incorporate non-linear terms like interaction terms defined by $\boldsymbol{s}_{\mathrm{k}}$ and $\boldsymbol{h}_{\mathrm{k}}$ along with their higher-order combinations.

For the model identification step (defined in the next section), we also require a smoother solution of the state which is defined by

$$p(\boldsymbol{x}_{\mathrm{k}}|\boldsymbol{s}_{1:\mathrm{K}}) = p(\boldsymbol{x}_{\mathrm{k}}|\boldsymbol{s}_{1:\mathrm{k}}) \int \left[\frac{p(\boldsymbol{x}_{k+1}|\boldsymbol{x}_{k})p(\boldsymbol{x}_{\mathrm{k}+1}|\boldsymbol{s}_{1:\mathrm{K}})}{p(\boldsymbol{x}_{\mathrm{k}+1}|\boldsymbol{s}_{1:\mathrm{k}})}\right] d\boldsymbol{x}_{\mathrm{k}+1} \tag{7}$$

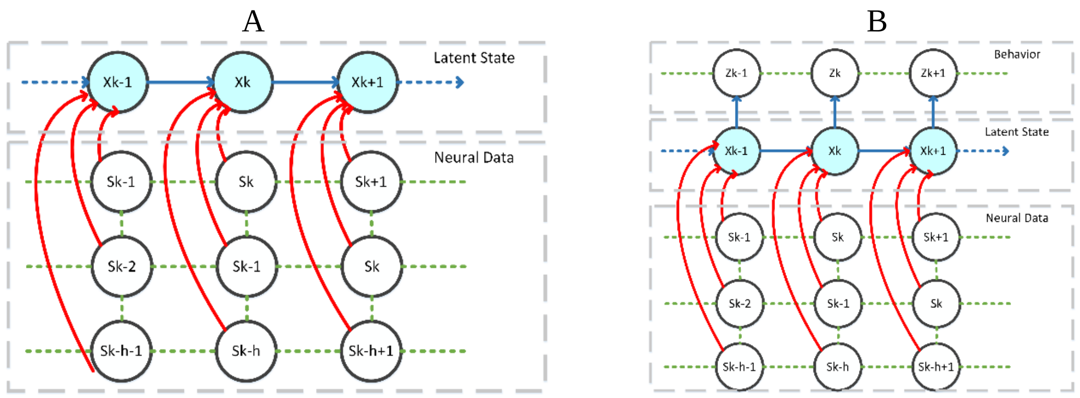

Figure 1. Direct Decoder Model. A. Schematic representation of the direct-decoder model. s_k represents neural activity at time index k, h represents the number of previous time points in the history term (determined by model selection techniques), and $\mathbf{x}_\mathrm{k}$ is the state variable. **B.** Schematic representation of the dynamical auto-decoder model. $\mathrm{z}_{\boldsymbol{k}}$ represents the behavioral readout at time index k, which is defined as a function of the state variables. The other parts of the model are the same as the D-D model.

2.B Dynamical auto-encoder model

Here, we discuss how the D-D model can be expanded to a dynamical auto-encoder model [19] by incorporating behavioral readouts. Let's assume that $\boldsymbol{z}_k$ represents the behavioral observation at time index k; given $\boldsymbol{z}_k$ and $\mathbf{s}_\mathrm{k}$, our objective is to estimate

$$p(\boldsymbol{x}_\mathrm{k}|\boldsymbol{s}_{1:\mathrm{k}}, \boldsymbol{z}_{1:k}) \tag{8}$$

As with the D-D methodology, we can express the filter solution defined in equation (8) as

$$p(\boldsymbol{x}_\mathrm{k}|\boldsymbol{s}_{1:\mathrm{k}}, \boldsymbol{z}_{1:\mathrm{k}}) \propto p(\boldsymbol{s}_\mathrm{k}, \boldsymbol{z}_\mathrm{k}|\boldsymbol{h}_\mathrm{k}, \boldsymbol{x}_\mathrm{k}) \times p(\boldsymbol{x}_\mathrm{k}|\boldsymbol{s}_{1:\mathrm{k}-1}, \boldsymbol{z}_{1:\mathrm{k}-1}) \tag{9}$$

We further assume that the $\boldsymbol{s}_\mathrm{k}$ and z_k are independent given $\boldsymbol{x}_\mathrm{k}$ [19]; as a result, we can rewrite equation (9) as

$$p(\boldsymbol{x}_\mathrm{k}|\boldsymbol{s}_{1:\mathrm{k}}, \boldsymbol{z}_{1:\mathrm{k}}) = p(\boldsymbol{s}_\mathrm{k}|\boldsymbol{h}_\mathrm{k}, \boldsymbol{x}_\mathrm{k}) \times p(\boldsymbol{z}_\mathrm{k}|\boldsymbol{h}_\mathrm{k}, \boldsymbol{x}_\mathrm{k}) \times p(\boldsymbol{x}_\mathrm{k}|\boldsymbol{s}_{1:\mathrm{k}-1}, \boldsymbol{z}_{1:\mathrm{k}-1}) \tag{10}$$

Given the definition of the history term, we can rewrite the filter update rule as

$$p(\boldsymbol{x}_\mathrm{k}|\boldsymbol{s}_{1:\mathrm{k}}, \boldsymbol{z}_{1:\mathrm{k}}) \propto \frac{p(\boldsymbol{x}_\mathrm{k}|\boldsymbol{h}_\mathrm{k}, \boldsymbol{s}_\mathrm{k})}{p(\boldsymbol{x}_\mathrm{k}|\boldsymbol{h}_\mathrm{k})} \times p(\boldsymbol{z}_\mathrm{k}|\boldsymbol{x}_\mathrm{k}) \times \ p(\boldsymbol{x}_\mathrm{k}|\boldsymbol{s}_{1:\mathrm{k}-1}, \boldsymbol{z}_{1:\mathrm{k}-1}) \tag{11}$$

where $\boldsymbol{z}_\mathrm{k}$, behavioral readout, is assumed to be independent of $\boldsymbol{h}_\mathrm{k}$ given the state process, $\boldsymbol{x}_\mathrm{k}$. With this assumption, we can build a recursive solution for the filter update rule [36]:

$$p(\boldsymbol{x}_\mathrm{k}|\boldsymbol{s}_{1:\mathrm{k}}, \boldsymbol{z}_{1:\mathrm{k}}) \propto \frac{p(\boldsymbol{x}_\mathrm{k}|\boldsymbol{h}_\mathrm{k}, \boldsymbol{s}_\mathrm{k})}{\int p(\boldsymbol{x}_\mathrm{k}|\boldsymbol{x}_{\mathrm{k}-1})p(\boldsymbol{x}_{\mathrm{k}-1}|\boldsymbol{h}_{\mathrm{k}-1}, \boldsymbol{s}_{\mathrm{k}-1})d\boldsymbol{x}_{\mathrm{k}-1}} \times p(\boldsymbol{z}_\mathrm{k}|\boldsymbol{x}_\mathrm{k}) \times$$

$$\int p(\boldsymbol{x}_\mathrm{k}|\boldsymbol{x}_{k-1})p(\boldsymbol{x}_{\mathrm{k}-1}|\boldsymbol{s}_{1:\mathrm{k}-1}, \boldsymbol{z}_{1:\mathrm{k}-1})d\boldsymbol{x}_{\mathrm{k}-1} \tag{12}$$

As with the D-D model, we describe the state transition process at time index k as a function of the previous state value and $\boldsymbol{\theta}$, the set of free parameters of the state equation.

$$\boldsymbol{x}_\mathrm{k}|\boldsymbol{x}_{\mathrm{k}-1} \sim f(\boldsymbol{x}_{\mathrm{k}-1};\ \boldsymbol{\theta}) \tag{13}$$

For the auto-encoder model, we have two processes: (1) a prediction process similar to what that has been derived for the D-D model, and (2) a set of observation processes. These processes are described by

$$\boldsymbol{x}_{\mathrm{k}}|\boldsymbol{s}_{\mathrm{k}},\boldsymbol{h}_{\mathrm{k}} \sim f(\boldsymbol{s}_{\mathrm{k}},\boldsymbol{h}_{\mathrm{k}};\boldsymbol{\Omega}) \tag{14.a}$$

$$\boldsymbol{z}_{\mathrm{k}}|\boldsymbol{x}_{\mathrm{k}} \sim f(\boldsymbol{x}_{\mathrm{k}};\boldsymbol{\Gamma}) \tag{14.b}$$

where equation (14.a) is analogous to equation (6) and equation (14.b) is $\boldsymbol{z}_{\mathrm{k}}$'s observation process. In (14.b), $\boldsymbol{\Gamma}$ is the set of free parameters that describe the behavioral encoder model. In the D-D and dynamical auto-encoder models described so far, we assume the model parameters and the state dimension are known. In the next section, we describe how the model parameters can be estimated given either $\boldsymbol{s}_k$ or both $\boldsymbol{s}_k$ and $\boldsymbol{z}_k$. In the following sections, we assume the dimension of state process, $\boldsymbol{x}_{\mathrm{k}}$**,** is pre-known. In general, identifying the dimension of the state process is a challenging modeling problem; in the discussion section, we will discuss possible solutions that can be used or developed in the search for an optimal dimension of the state process.

2.C Model Parameter Estimation

We use the EM algorithm [39] to find maximum likelihood estimates of the model-free parameters, a subset of $\{\boldsymbol{\theta}, \boldsymbol{\Omega}, \boldsymbol{\Gamma}\}$. The EM algorithm is an established solution to perform maximum likelihood estimation of model parameters when there is an unobservable process or missing observations [21]. The other possible solution includes fully Bayesian or Variational Bayes approaches, which can applied to our modeling framework where a Bayesian prior exist per the model parameters [40]. Here, we present the EM solution for the auto-encoder model, given that a D-D model is a specific form of this model. The EM solution recursively estimates the model parameters $\{\boldsymbol{\theta}^{(\mathrm{r})}, \boldsymbol{\Omega}^{(\mathrm{r})}, \boldsymbol{\Gamma}^{(\mathrm{r})}\}$ – here, superscript r is the iteration of the EM procedure, based on an updated posterior distribution of $\boldsymbol{x}_{0:\mathrm{K}}$ and parameter estimates from the previous EM iteration, $\{\theta^{(\mathrm{r}-1)}, \boldsymbol{\Omega}^{(\mathrm{r}-1)}, \boldsymbol{\Gamma}^{(\mathrm{r}-1)}\}$. The EM algorithm includes two steps: expectation (E-step), and maximization (M-step) [41]. The E-step is defined by

$$\mathrm{Q} = \mathrm{E}_{\boldsymbol{x}_{0:\mathrm{K}}|\boldsymbol{s}_{1:\mathrm{K}},\boldsymbol{z}_{1:\mathrm{K}}}\left[\log\left(\mathrm{p}(\boldsymbol{x}_0)\prod_{\mathrm{k}=1}^{\mathrm{K}}\mathrm{p}(\boldsymbol{x}_{\mathrm{k}}|\boldsymbol{x}_{\mathrm{k}-1}) \times \mathrm{p}(\boldsymbol{z}_{\mathrm{k}}|\boldsymbol{x}_{\mathrm{k}}) \times \mathrm{p}(\boldsymbol{s}_{\mathrm{k}}|\boldsymbol{x}_{\mathrm{k}},\boldsymbol{h}_k)\right)\right] \tag{15}$$

here, the right side is the full likelihood of the state, neural and behavioral readout.
The Q function can be rewritten as

$$\mathrm{Q} = \mathrm{E}_{\boldsymbol{x}_{0:\mathrm{K}}|\boldsymbol{s}_{1:\mathrm{K}},\boldsymbol{z}_{1:\mathrm{K}}}\left[\log\left(\mathrm{p}(\boldsymbol{x}_0)\prod_{\mathrm{k}=1}^{\mathrm{K}}\mathrm{p}(\boldsymbol{x}_{\mathrm{k}}|\boldsymbol{x}_{\mathrm{k}-1}) \times \mathrm{p}(\boldsymbol{z}_{\mathrm{k}}|\boldsymbol{x}_{\mathrm{k}}) \times \frac{\mathrm{p}(\boldsymbol{x}_{\mathrm{k}}|\boldsymbol{s}_{\mathrm{k}},\boldsymbol{h}_{\mathrm{k}})}{\int \mathrm{p}(\boldsymbol{x}_{\mathrm{k}}|\boldsymbol{x}_{\mathrm{k}-1})\mathrm{p}(\boldsymbol{x}_{\mathrm{k}-1}|\boldsymbol{s}_{\mathrm{k}-1},\boldsymbol{h}_{\mathrm{k}-1})d\boldsymbol{x}_{\mathrm{k}-1}}\right)\right] \tag{16}$$

given the prediction process and the behavioral observation process. Expanding the Q function yields

$$\mathrm{Q} = \mathrm{E}_{\boldsymbol{x}_{0:\mathrm{K}}|\boldsymbol{s}_{1:\mathrm{K}},\boldsymbol{z}_{1:\mathrm{K}}}\Big[\log\mathrm{p}(\boldsymbol{x}_0) + \sum_{\mathrm{k}=1}^{\mathrm{K}}\log\mathrm{p}(\boldsymbol{x}_{\mathrm{k}}|\boldsymbol{x}_{\mathrm{k}-1}) + \sum_{\mathrm{k}=1}^{\mathrm{K}}\log\mathrm{p}(\boldsymbol{z}_{\mathrm{k}}|\boldsymbol{x}_{\mathrm{k}}) + \sum_{\mathrm{k}=1}^{\mathrm{K}}\log\mathrm{p}(\boldsymbol{x}_{\mathrm{k}}|\boldsymbol{s}_{\mathrm{k}},\boldsymbol{h}_{\mathrm{k}}) - \sum_{\mathrm{k}=1}^{\mathrm{K}}\log\int\mathrm{p}(\boldsymbol{x}_{\mathrm{k}}|\boldsymbol{x}_{\mathrm{k}-1})\,\mathrm{p}(\boldsymbol{x}_{\mathrm{k}-1}|\boldsymbol{s}_{\mathrm{k}-1},\boldsymbol{h}_{\mathrm{k}-1})d\boldsymbol{x}_{\mathrm{k}-1}\Big] \tag{17}$$

The Chapman-Kolmogorov equation in (17) can be expressed as

$$\int \mathrm{p}(\boldsymbol{x}_{\mathrm{k}}|\boldsymbol{x}_{\mathrm{k}-1})\,\mathrm{p}(\boldsymbol{x}_{\mathrm{k}-1}|\boldsymbol{s}_{\mathrm{k}-1},\boldsymbol{h}_{\mathrm{k}-1})\mathrm{d}\boldsymbol{x}_{\mathrm{k}-1} = \mathrm{E}_{\boldsymbol{x}_{\mathrm{k}-1}|\boldsymbol{s}_{\mathrm{k}-1},\boldsymbol{h}_{\mathrm{k}-1}}[\mathrm{p}(\boldsymbol{x}_{\mathrm{k}}|\boldsymbol{x}_{\mathrm{k}-1})] \tag{18}$$

Note that when the state process is linear with an additive Gaussian noise and the prediction process is a multi-variate normal, there is a closed form solution for this expectation. To derive a more general solution, when the state process does not follow a multi-variate normal process, we can use the Jensen inequality – e.g., $\log(\mathrm{E}[\mathrm{g}(x)]) \geq \mathrm{E}\big[\log\big(\mathrm{g}(x)\big)\big]$, to exchange the log and expectation operations for the last term in equation (17), which is rewritten in equation (18). This yields a lower bound for Q, which can be written as

$$\mathrm{Q} \geq \mathrm{E}_{\boldsymbol{x}_{0:\mathrm{K}}|\boldsymbol{s}_{1:\mathrm{K}},\boldsymbol{z}_{1:\mathrm{K}}}\Big[\log \mathrm{p}(\boldsymbol{x}_0) + \textstyle\sum_{\mathrm{k}=1}^{\mathrm{K}} \log \mathrm{p}(\boldsymbol{x}_{\mathrm{k}}|\boldsymbol{x}_{\mathrm{k}-1}) + \sum_{\mathrm{k}=1}^{\mathrm{K}} \log \mathrm{p}(\boldsymbol{z}_{\mathrm{k}}|\boldsymbol{x}_{\mathrm{k}}) + \sum_{\mathrm{k}=1}^{\mathrm{K}} \log \mathrm{p}(\boldsymbol{x}_{\mathrm{k}}|\boldsymbol{s}_{\mathrm{k}},\boldsymbol{h}_{\mathrm{k}})\Big] - \textstyle\sum_{\mathrm{k}=1}^{\mathrm{K}} \mathrm{E}_{\boldsymbol{x}_{\mathrm{k}}|\boldsymbol{s}_{1:\mathrm{K}},\boldsymbol{z}_{1:\mathrm{K}}} \mathrm{E}_{\boldsymbol{x}_{\mathrm{k}-1}|\boldsymbol{s}_{\mathrm{k}-1},\boldsymbol{h}_{\mathrm{k}-1}}[\log \mathrm{p}(\boldsymbol{x}_{\mathrm{k}}|\boldsymbol{x}_{\mathrm{k}-1})] \tag{19}$$

where, the expectation with respect to $\boldsymbol{x}_{\mathrm{k}-1}$ in the last term, defined by the prediction process, will be a function of the model free parameters. In the last term of equation (19), the order of log and expectation has changed, and this will make the estimation of Q easier; note that the expectation of $\log \mathrm{p}(\boldsymbol{x}_{\mathrm{k}}|\boldsymbol{x}_{\mathrm{k}-1})$ appears twice in the Q function and its expectation is easy to calculate when the state process is a linear multivariate normal.

The updated parameter set at iteration (r) can be found by maximizing Q, which is the M-step. This can be done analytically if there is a closed-form solution for the expectation defined in equation (18), or maximizing its lower bound defined in equation (19). The maximization is defined by

$$\{\boldsymbol{\theta}^{(\mathrm{r})},\boldsymbol{\Omega}^{(\mathrm{r})},\boldsymbol{\Gamma}^{(\mathrm{r})}\} = \arg\max_{\boldsymbol{\theta},\boldsymbol{\Omega},\boldsymbol{\Gamma}} \mathrm{Q} \tag{20}$$

The optimization step in equation (20) can be calculated analytically or numerically, e.g. gradient descent [42]. After each iteration, a new set of parameters are estimated, and the EM routine is stopped when a stopping criterion based on the likelihood of growth or parameter changes is satisfied [21].

2.D DNN as direct-decoder model

While a Gaussian linear process can be a proper choice for the direct-decoder model, a more flexible model for the prediction process might capture the complex dynamics presented in high-dimensional data better. To address this, we can utilize DNNs to predict $\boldsymbol{x}_{\mathrm{k}}$ distribution given the current and previous neural features. This modeling viewpoint is also aligned with recent advances in the field, where machine learning techniques are used to better understand neural data [11, 12, 22].

A DNN model can be defined by,

$$\boldsymbol{x}_{\mathrm{k}}|\boldsymbol{s}_{\mathrm{k}},\boldsymbol{h}_{\mathrm{k}} \sim f(\boldsymbol{s}_{\mathrm{k}},\boldsymbol{h}_{\mathrm{k}};\boldsymbol{\Omega}) \tag{21}$$

where $\boldsymbol{x}_{\boldsymbol{k}}$ is the state process and $\boldsymbol{\Omega}$ is the network free parameters. This is the same model being used to build the prediction process in equation (14); as a result, we can use DNN in our direct-decoder and auto-encoder model given its free parameters are known. Thus, the objective is to estimate the DNN free parameters in the context of a dynamical auto-encoder model. Note that if $\boldsymbol{x_k}$ is known, the DNN parameter estimation turns to a supervised problem, which has established solutions [12, 22].

For the auto-encoder model, we applied EM routine to recursively update the model parameters. We can expand this technique to train the DNN by drawing samples from the state posterior distribution. Let's assume $\boldsymbol{x}_{0:\mathrm{K}}^{(\boldsymbol{m})}$ is the m^{th} sample trajectory from the state given the neural and behavioral data, and we have M, $m = 1 \dots M$, trajectories of the state process. Using the state trajectories, we can calculate Q function, defined in equation (19). Using the M samples, Q is defined by

$$\mathrm{Q} \approx \frac{1}{M}\sum_{m=1}^{M}\left[\log \mathrm{p}\left(\boldsymbol{x}_0^{(m)}\right) + \sum_{\mathrm{k}=1}^{\mathrm{K}}\log \mathrm{p}\left(\boldsymbol{x}_{\mathrm{k}}^{(\boldsymbol{m})}\middle|\boldsymbol{x}_{\mathrm{k}-1}^{(\boldsymbol{m})}\right) + \sum_{\mathrm{k}=1}^{\mathrm{K}}\log \mathrm{p}\left(\boldsymbol{z}_{\mathrm{k}}\middle|\boldsymbol{x}_{\mathrm{k}}^{(\boldsymbol{m})}\right) + \sum_{\mathrm{k}=1}^{\mathrm{K}}\log \mathrm{p}\left(\boldsymbol{x}_{\mathrm{k}}^{(\boldsymbol{m})}\middle|\boldsymbol{s}_{\mathrm{k}},\boldsymbol{h}_{\mathrm{k}}\right)\right] - \frac{1}{\mathrm{MN}}\sum_{m=1}^{M}\sum_{n=1}^{N}\sum_{\mathrm{k}=1}^{\mathrm{K}}\left[\log\left(\mathrm{p}\left(\boldsymbol{x}_{\mathrm{k}}^{(\boldsymbol{m})}\middle|\boldsymbol{x}'^{(\boldsymbol{n})}_{\mathrm{k}-1}\right)\right)\right] \quad (22)$$

where $\boldsymbol{x}'^{(\boldsymbol{n})}_{\mathrm{k}-1}$ are samples drawn from the prediction process, not the state posterior distribution – note that, we draw N samples for it. While Q is maximized, the DNN is trained using samples of the state trajectory, which turns to a supervised learning problem. Note that the DNN is trained on multiple state measure, $\boldsymbol{x}_{\mathrm{k}}^{(\boldsymbol{m})}$, per each time index given $\boldsymbol{s}_{\mathrm{k}}$ and $\boldsymbol{h}_{\mathrm{k}}$. In other words, the DNN training corresponds to maximize the average of log of likelihood function on multiple samples of state trajectory.

To draw samples from the state trajectory, we can use the conditional distribution of $\boldsymbol{x_k}$ given $\boldsymbol{x}_{k+1}$, $\boldsymbol{s}_{1:\mathrm{K}}$, and $\boldsymbol{z}_{1:K}$. This distribution can be computed by [43]

$$p(\boldsymbol{x}_k|\boldsymbol{x}_{k+1},\boldsymbol{s}_{1:\mathrm{K}},\boldsymbol{z}_{1:K}) = p(\boldsymbol{x}_{k+1}|\boldsymbol{x}_k) \times p(\boldsymbol{x}_k|\boldsymbol{s}_{1:\mathrm{k}},\boldsymbol{z}_{1:k})/p(\boldsymbol{x}_{k+1}|\boldsymbol{s}_{1:\mathrm{k}},\boldsymbol{z}_{1:k}) \quad (23)$$

We first draw sample from $\mathrm{p}(\boldsymbol{x}_K|\boldsymbol{s}_{1:\mathrm{K}},\boldsymbol{z}_{1:K})$, defined in equation (7), and recursively draw samples from equation (23) for time steps $K-1$ to 0. We then use these samples in equation (22) to find updated model parameters maximizing Q.

In the previous section, we discussed a numerical solution for the filter and smoother steps. In cases where the state is high-dimensional and computing the integrals in filter and smoother equations using simple numerical methods become computationally expensive, we can use sequential Monte Carlo (SMC) methods, otherwise known as particle filters, as an alternate approach for filter and smoother estimation [44, 45]. If we use the particle filter, we can use smoother samples in the Q function defined in equation (22).

Here, we described a DNN with a fixed set of parameters. In order to avoid possible overfitting issues in our DNN training, it is suggested to use a B-DNN in the auto-encoder model [26]. In B-DNN, there will be a probability distribution over the network weights instead of fixed weights. It is possible to build a fully Bayesian auto-encoder model, where not only DNN's weights are probabilistic but also the observation and state processes parameters are defined through prior distributions. In our modeling solution, we have already derived a fully Bayesian solution for the state process and a maximum likelihood estimate for model-free parameters, including the DNN weights. Extending the DNN parameter estimate to MAP estimate is easy, and it can be done through a penalized likelihood estimate in the context of our EM algorithm [46]. However, solving a fully Bayesian solution for DNN is generally a complex and computationally intractable modeling problem. In Appendix A, we provide a suboptimal solution based on the EM solution we already developed here. The proposed solution is based on the MCMC solution and it might provide a reasonable solution when DNN has a limited number of wights or the network weights are significantly correlated.

3 Datasets

We applied our methodology to different decoding problems. In the first problem, the goal is to decode the movement trajectory of a rat from invasive neural recordings, while the rat is foraging in a W-shaped maze for food (Hippocampus dataset) [12, 22, 29, 30]. The second problem investigates how invasive neural recordings and behavioral observations from a human participant can be processed to infer a dynamical internal cognitive process, representing the human participant's intent to communicate with a companion. In the following section, we describe each dataset in more detail.

3.A Movement Dataset

In this dataset, 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 [29]. The neural data were recorded from 62 place cells in the CA1 and CA2 regions of the hippocampus brain area of a Long-Evans rat, aged approximately 6 months. The rat has been trained to traverse between the home box and the outer arms to receive a liquid reward (condensed milk) at the reward locations. **Figure 2.A** shows the maze structure and the rat's movement trajectory in 2-D spaces, where the rat position at each time step is represented by (x, y) coordinates. The spiking activity of these 62 units was detected offline by choosing events that their peak-to-peak amplitudes were above a threshold of 80uV in at least one of the tetrode channels (see **Figure 2.B**). In the experiment process, the actual rat’s position was measured by video tracking software which was used as the ground truth for the position (see **Figure 2.C**). We used a 15-minute-long recording of the experiment, with a time resolution of 33 milliseconds, to analyze different decoding solutions. The first 85% of the recording (~13 minutes) was used to train the prediction and state processes’ models, and the remaining 15% (~2 minutes) of the data was used to test the model’s decoding performance.

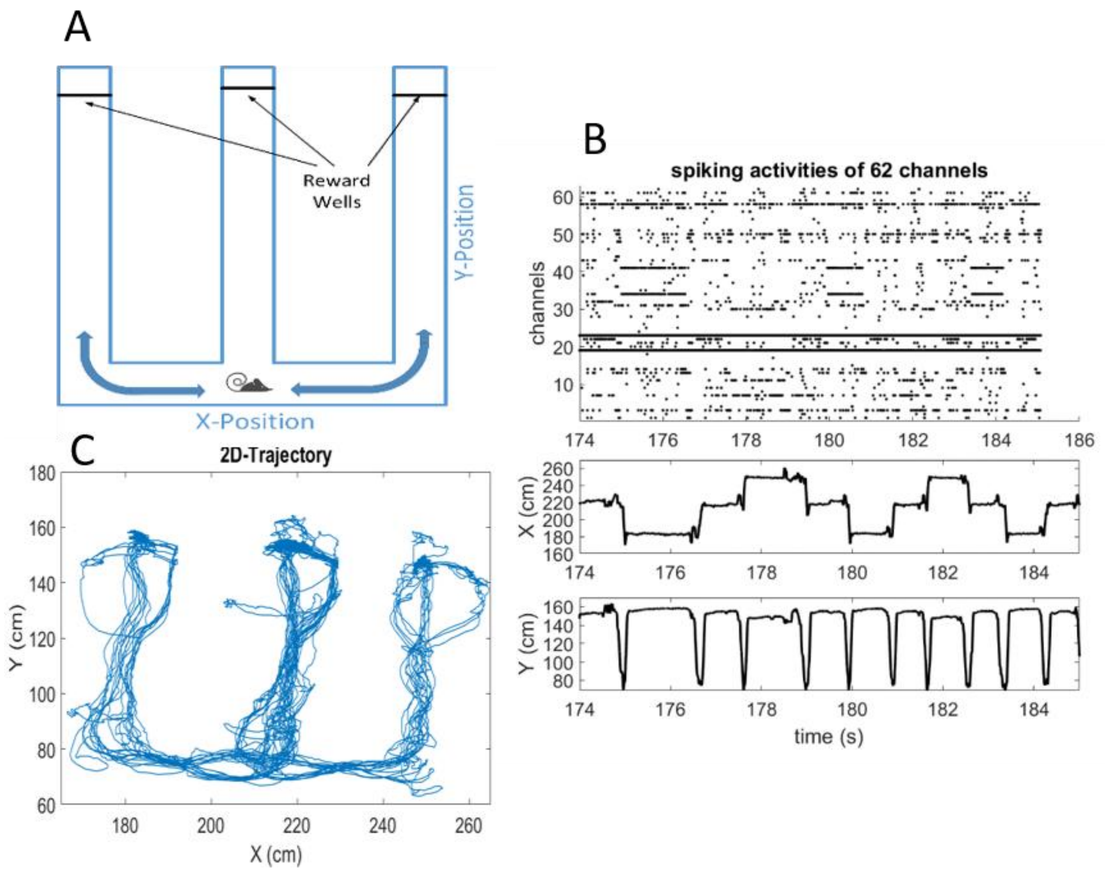

Figure 2 Movement Dataset: the maze topology, the rat movement trajectory, and sample neural data. **A.** W-maze topology, the rat moves from the center arm to the left and right arms to get a food reward. **B.** Both movement trajectories along with *x* and *y* directions and neural activity of the 62 channels. **C.** 2-D trajectory of the rat movement during the experiment. The experiment is about 3 minutes, and during this 3 minutes, the rat has traversed 3 times inside the maze.

3.B Conversation Dataset

In this experiment, we investigated how the neural recordings and spoken words from a human study participant can be processed to infer the dynamics of a cognitive state related to verbal communication. In this dataset (unpublished data; Hadjinicolaou, Cash, et al., Massachusetts General Hospital), study participants were implanted with intracranial (sEEG) electrodes for clinical monitoring of their epilepsy, for the duration of their stay in the telemetry ward. The raw neural data were acquired at a sampling rate of 2 kHz using a 128-channel neural signal processor recording system (Cerebus, Blackrock Microsystems, UT) and neighboring channels were re-referenced with a bipolar montage to mitigate volume conduction [47]. The spectral power was estimated for each channel for the theta (4-8 Hz) and gamma (70-115 Hz) frequency bands (Chronux). Neural activity was recorded during conversations between the participants and their companions, including hospital staff, family, friends, and study investigators. All spoken dialog within each recording interval was captured and transcribed to yield individual word timings that are synchronized to the neural data (see **Figure 3**). We used this data – neural activity and spoken dialogs – to examine our dynamical auto-encoder model in search of the underlying intent state of the patient to communicate.

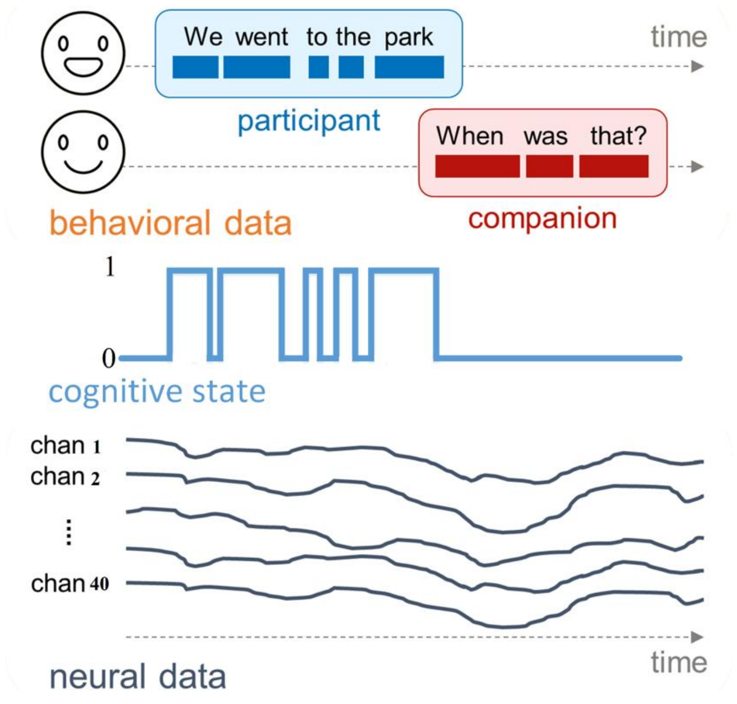

Figure 3 Explanation of the conversation dataset. The diagram shows the relationship between the neural recordings, spoken words from a human study participant, and the cognitive state related to verbal communication during the experiment period.

3.C Decoding Problems

Having the direct-decoder modeling framework, we now discuss how the model can be used to decode two different correlates of neural data already described. We describe model identification for these two decoding problems and use two metrics to compare the performance of our proposed solution along with other established decoding techniques. The performance metrics include: mean-squared-error (RMSE), and 95% highest posterior density (HPD) region [48].

3.C.1 Decoding 2-D movement trajectories using the direct-decoder model

We assume the rat's position in the maze at time interval k is specified by the state variable $\boldsymbol{X}_\mathrm{k} = (x_\mathrm{k}, y_\mathrm{k})$, where x_k and y_k represent the rat's x- and y-coordinates. We define the state process by

$$\boldsymbol{X}_\mathrm{k} = \mathbf{A}\,\boldsymbol{X}_{\mathrm{k}-1} + \mathbf{Q} \qquad \mathbf{A} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \qquad \mathbf{Q} \sim N\left(\begin{bmatrix} 0 \\ 0 \end{bmatrix}, \begin{bmatrix} \sigma_x^2 & 0 \\ 0 & \sigma_y^2 \end{bmatrix}\right) \qquad (24)$$

where the covariance matrix $\mathbf{Q}$ is assumed to be diagonal with σ_x^2 and σ_y^2 terms representing the movment variances. We estimate these two variances empirically using the rat's movement during the training session. For the prediction process, we assume the state $\boldsymbol{X}_\mathrm{k}$ can be predicted using a linear regression model where the predictor variables are the ensemble spiking activity of 62 cells, where each spike train is filtered with a Gaussian window with length 20 ms. We further assume the noise process follows a normal distribution. We build two regression models; one per each coordinate, as a function of ensemble spiking activity. The prediction process and its decomposition into two predictor models are defined by

$$p(\boldsymbol{X}_{\mathrm{k}}|\boldsymbol{s}_{\mathrm{k}},\boldsymbol{h}_{\mathrm{k}}) = p(x_{\mathrm{k}}|y_{\mathrm{k}},\boldsymbol{s}_{\mathrm{k}},\boldsymbol{h}_{\mathrm{k}}) \times p(y_{\mathrm{k}}|\boldsymbol{s}_{\mathrm{k}},\boldsymbol{h}_{\mathrm{k}}) \tag{25}$$

where $\boldsymbol{h}_{\mathbf{k}}$ represents the history of the ensemble spiking activity from the previous time intervals. Note that given the rat's movement is bounded by the maze, the state process defined in equation (24) is a misspecified model. To address this issue, we add a penalty term to the prediction process which accounts for the topology of the maze – a detailed explanation of the penalty term can be found in our previous work [29]. The revised prediction process is expressed as

$$p(\boldsymbol{X}_{\mathrm{k}}|\boldsymbol{s}_{\mathrm{k}},\boldsymbol{h}_{\mathrm{k}}) \sim p(x_{\mathrm{k}}|y_{\mathrm{k}},\boldsymbol{s}_{\mathrm{k}},\boldsymbol{h}_{\mathrm{k}}) \times p(y_{\mathrm{k}}|\boldsymbol{s}_{\mathrm{k}},\boldsymbol{h}_{\mathrm{k}}) \times L(x_{\mathrm{k}},y_{\mathrm{k}}) \tag{26}$$

where $L(x_{\mathrm{k}},y_{\mathrm{k}})$ is close to zero for *x*-*y* coordinates outside the maze area and one otherwise (**Figure 4.F**). Note that adding the penalty term does not change any aspects of the modeling pipeline.

To find the optimal length of the history term for the *x*- and *y*-coordinate regression models, we use a forward model selection process. For the *y*-coordinate model, the null model is defined by the current observation of the spiking activity and the ensemble spiking activity of the previous time points are added recursively. We use a BIC criterion [42] to determine when the increase in the length of the history term does not improve the model fit. For the *x*-coordinate model, the null model is y_{k}, and the ensemble spiking activity of the current and previous time points are added to the model in search of the optimal length of the history term. As it was done for the y-coordinate, we use a BIC criterion to find the proper length of the history term for the *x*-coordinate model. For this dataset, the history term for the *x*-coordinate ends up to 12 previous time points (~ 400 milliseconds) and the *y*-coordinate includes 18 time points (~ 600 milliseconds) (**Figure 4.A**).

To assess the performance of our modeling framework, we decoded the rat movement trajectory in the test dataset using four different models: (1) an exact point process decoder model, described in [29], (2) a D-D model with one-step history terms (**Figure 4.C)**, (3) a D-D model with optimal history term lengths (**Figure 4.D** and **Figure 4.B**), and (4) a B-DNN model with optimal history term length (**Figure 4.E**). The performance results in **Table. 1** shows comparable performance between the (exact) point process model and the D-D model with the optimal history term. The performance result also suggests the necessity of incorporating proper history terms in building a more robust decoder model. We expect B-DNN to have a better prediction accuracy compared to other D-D models; however, this is not the case. A possible reason is a limited dataset we have to train the B-DNN model, and thus, the model performance drops in the test dataset despite attaining a superb performance in the training dataset.

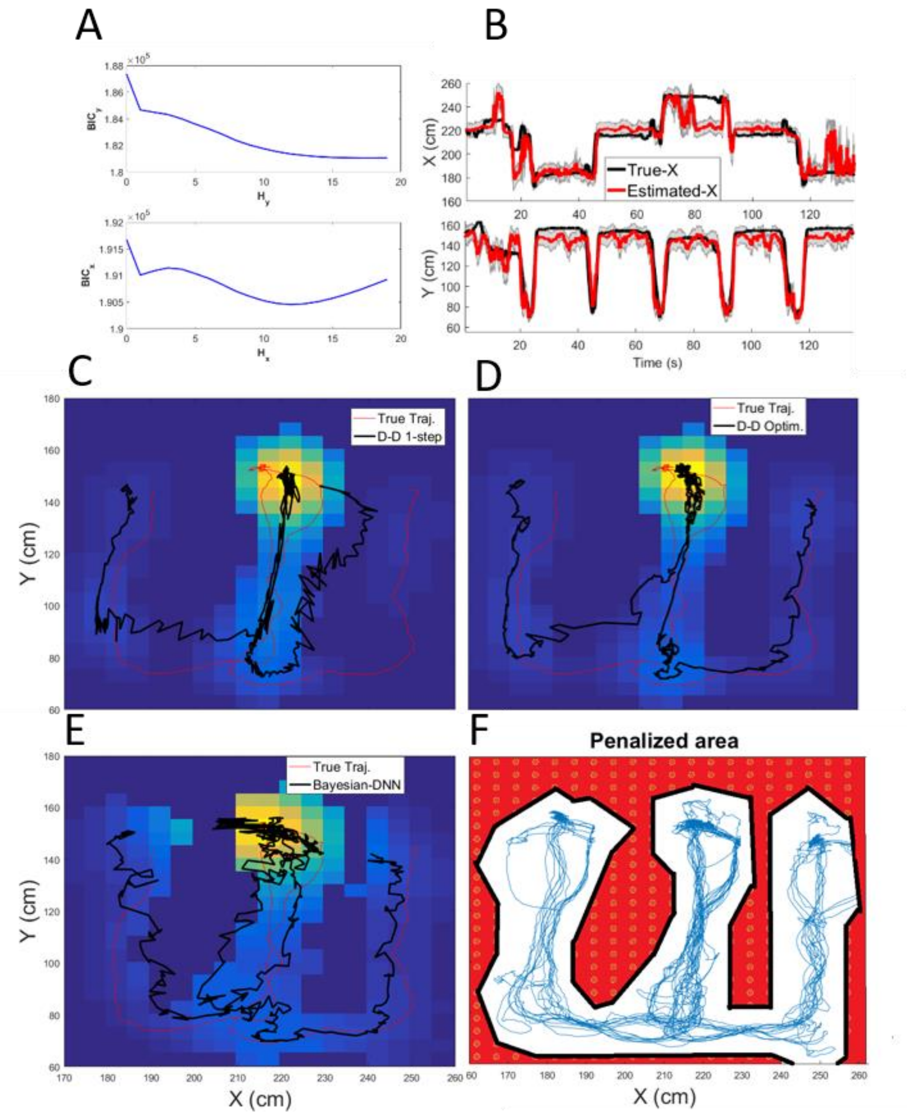

Figure 4. **Decoding results in 2-D movement trajectory. A.** BIC curve for different *x*- and *y*-coordinate model history term lengths. The optimum lengths (corresponding to the minimum BIC) for hx and hy are 12 and 18, respectively. **B.** The D-D model result for *x* and *y* coordinates with optimal history lengths. **C.** The D-D model result with one-step history terms. **D.** The D-D model result with optimal history term lengths. **E.** The B-DNN model results with optimal history term length. **F.** Visualization of rat movement trajectories (blue traces) together with an overlay of the penalty term $L(x_k, y_k)$ (red area), defined in equation (30). The penalty term is close to zero for *x*-*y* coordinates outside the maze area (identified by red dots) and one otherwise.

Table 1 Different Models Decoding Performance

Method	**RMSE (cm)**		**95%HPD**	
	Train	Test	Train	Test
Exact numerical solution	--	12.7	--	85.8%
D-D result with hx=1 and hy=1	--	17.7	--	85.9%
D-D result with optimum hx and hy	--	13.8	--	88.7%
B-DNN	14.5	15.1	88.7%	88.4%

3.C.2 Decoding cognitive state using the dynamical autoencoder

The onset times of words spoken by the participant comprise the behavioral signal z_k, which is specified as a point-process observation model [15]. The auto-encoder model aims to build the relationship between intention state x_k, and neural features, in conjunction with the behavioral observation. The neural features $\boldsymbol{s}_k$ consist of spectral power at different frequency bands from a subset bipolar recording channels, which add up to 40 features [49].

We assume the prediction process follows a normal distribution, where the expected value is a linear function of the neural features [19]. The intention state x_k is characterized by a random walk model [14]. The auto-encoder model is defined by

$$x_0 \sim \mathcal{N}(m_0, \sigma_0^2) \tag{27.a}$$

$$x_\mathrm{k} | x_\mathrm{k-1} \sim \mathcal{N}(x_\mathrm{k-1}, \sigma_\varepsilon^2) \tag{27.b}$$

$$p(z_\mathrm{k} = 1) \approx \lambda_\mathrm{k}\,\Delta \qquad \lambda_\mathrm{k} = \exp(a_0 + a_1\, x_\mathrm{k} + a_2\, p_\mathrm{k} + a_3\, q_\mathrm{k}) \tag{27.c}$$

$$x_\mathrm{ck} | \boldsymbol{s}_\mathrm{ck}, \boldsymbol{h}_\mathrm{ck} \sim \mathcal{N}(w_0 + \mathbf{w}_1^T \boldsymbol{s}_\mathrm{ck} + \mathbf{w}_2^T \boldsymbol{h}_\mathrm{ck}, \sigma_\mathrm{v}^2) \tag{27.d}$$

where $\{\sigma_\varepsilon^2, (m_0, \sigma_0^2), (a_0, a_1, a_2, a_3), (w_0, \boldsymbol{w}_1, \boldsymbol{w}_2, \sigma_\mathrm{v}^2)\}$ are the model free parameters – to avoid model identifiability issue, we set a_1 to 1. Here, we assume the history term for the conditional intensity of spoken words can be defined by p_k and q_k, that corresponds to the number of spoken words from the participant and their companion respectively, over the last 400 milliseconds [8, 50]. The processing interval Δ is set to 50 milliseconds; small enough that the probability of more than one word inside each interval is negligible [15]. Neural features are updated at a slower rate than the behavioral one; they are updated once every 2 seconds. As a result, we have a multi-rate auto-encoder model, and this has been addressed with the c term in equation (27.d). The c term is equal to 40 – e.g., 2 seconds/50 msec; this implies that we have neural activity updated for every 40 times updates of behavioral data.

We use our modeling solution to estimate the state process, which simultaneously maximizes the likelihoods of the behavioral readout and neural recording. **Figure 5.B** shows the intent state estimation given behavioral readout z_k (see **Figure 5.A)**, neural activity $\boldsymbol{s}_\mathrm{k}$, and optimal history term h_k, which is selected by the LASSO regularization method [38, 49]. The decoding result using only neural activity is shown in **Figure 5.B.** The intention state increases when the participant starts to talk and it decreases when they stop or the other companion starts to talk – or, the participant starts to listen, which is aligned with our expectations. As it can be seen from **Figure**

5.B, by adding the behavioral signals to the decoder model, auto-encoder model, we get more salient state estimation compared to the state estimation run only by the D-D model.

We use the model parameters using the auto-encoder model which is optimized by using both behavioral and neural data. We update the model parameters using the EM technique described in section 2.C. Initial values and optimized values of the D-D model parameter using the EM algorithm are shown in **Figure 5.C**. We assume the history term is 1, and find a sparse set of the weights. This corresponds to a MAP estimate in EM with a Laplace prior to D-D weights. The D-D model weights can also reflect physiological mechanisms of intention, like which neural activity is positively/negatively correlated with the intention and which neural activity is not predictive of the intention – this is not the scope of this research.

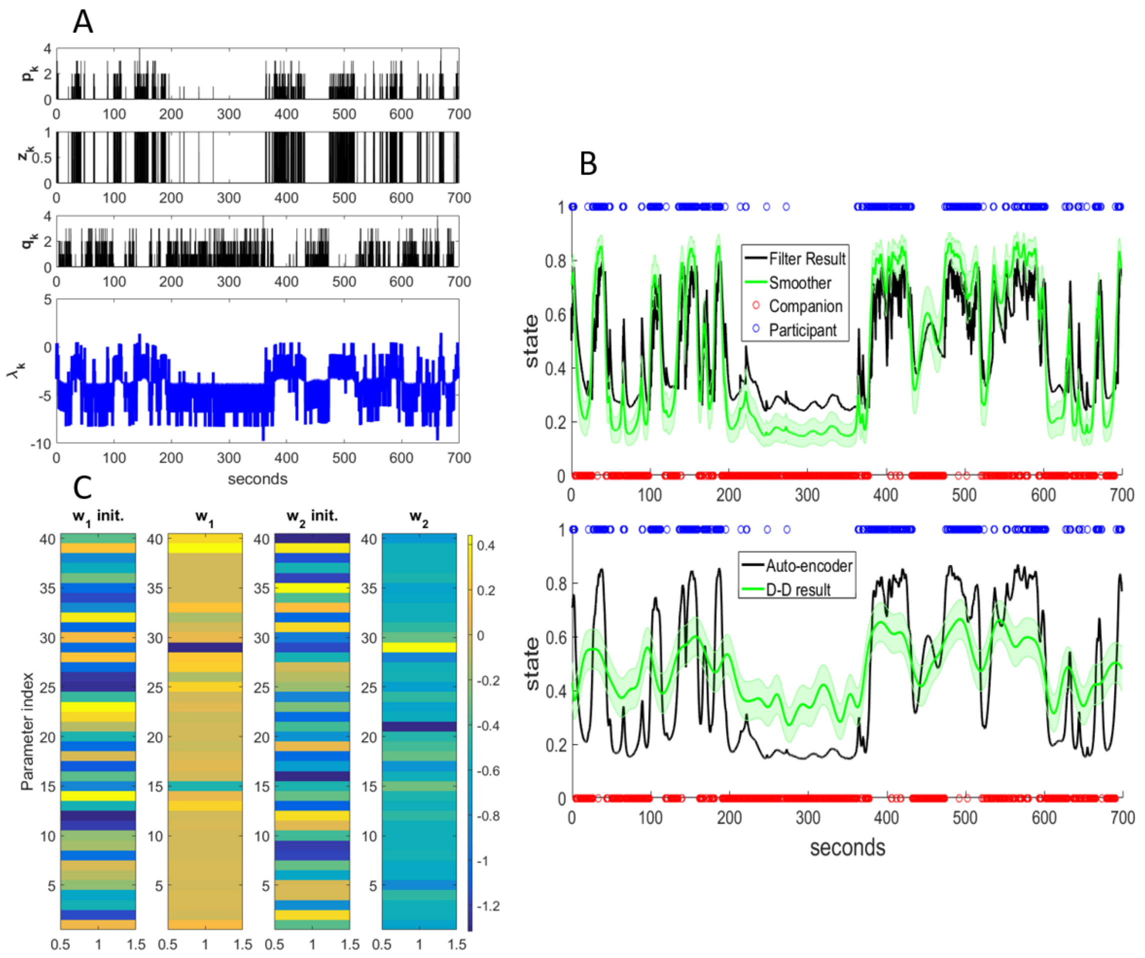

Figure 5. **Decoding communication intent using D-D model**. **A.** Temporal evolution of p_k, z_k , q_k, and λ_k variables, defined in equation (27). **B.** Conversation intent estimation using the dynamical auto-encoder model . (top) Estimated intent state using both behavioral readout z_k, and neural activity s_k; (bottom) Estimated intent state using only neural activity. **C.** MLE estimate of the direct-decoder model parameters.

5 Discussion

In this paper, we introduced a state-space decoder model with a discriminant observation process. The discriminant observation process called D-D model characterizes the state process as a function of current and previous neural data. The model filter solution accounts for time variability present in the observed data; in addition to that, we derive the maximum likelihood (ML) estimation of the model parameters using a revised EM technique [21]. A distinct difference between our work and previous work like those being suggested by Harrison et al. [20] is generalizing of the discriminat process by including the history terms. For this, we not only show the necessity of the history terms through mathematical derivation but also demonstrate the need through a neural decoding problem – e.g., 2-D decoding problem. We then expanded the D-D model to a dynamical auto-encoder model, which lets us link the behavioral readout and high-dimensional neural recording in pursuit of a low-dimensional manifold representing underlying emotional or cognitive states. We discussed how DNNs, including B-DNNs, can be added as the direct-decoder model, which increases the flexibility of the framework to characterize complex dynamics present in the high-dimensional data. Not only we demonstrated how DNNs can be added in the auto-encoder pipeline, but also provided training procedure for DNNs and B-DNNs – Appendix A. Finally, we demonstrated the application of the auto-encoder model in a novel and potentially complex clinical experiment and showed that participant communication intent can be estimated through neural data.

Whilst characterizing the relationship between brain dynamics and behavioral or physical correlates like movement in the context of high dimensional recording is of great importance; a more significant research direction is to estimate underlying cognitive or mental processes that shape the relationship between distributed neural activity and behavior. The dynamical auto-encoder model proposed in this work is well suited for this research objective and can be applied to complex and novel tasks like the communication intent task that we discussed here. The decoding examples presented here highlight the flexibility of our proposed modeling framework, and the fact that, it can be applied to different modalities of neural and behavioral data across different tasks and domains.

In our previous research – Yousefi et al. [19], we discussed two-step encoding and decoding solution to characterize the relationship between brain dynamics and behavior thorough a dynamical cognitive process. In that work, we first estimate the underlying cognitive states using behavioral readout and we then utilize the estimated cognitive state to build a neural encoder and potentially decoder. There, the temporal dynamics of the cognitive state are mandated by the behavior and the neural encoding process looks for neural features that represent dynamics of cognitive state-driven solely by behavior. In dynamical auto-encoder solution; the underlying cognitive state dynamics is not solely driven by the behavior readout; instead, it is defined through joint neural and behavioral readout. This viewpoint can improve our estimation of cognitive state, where different processes come to an agreement in their estimation of the latent dynamical process representing cognitive or emotional states. It is worth mentioning, we already benefit from the D-D process as a part of the dynamical auto-encoder model letting us avoid building many auto-encoder models, each per neural activity.

In both experiments, we used a Gaussian process to build the parametric D-D model. Note that we have a priori knowledge on the observation process noise model, as we define what state process model constructs the underlying dynamical process. The state processes in both decoding problems are defined by Gaussian processes; as a result, we used a Gaussian process for the observation process. The noise process for both decoder problems are assumed to be stationary, whilst the noise process characteristics can change over time. To address this, we can build a more flexible model like DNN to capture changes present in the noise process. Parametric models like those used in two experiments might fail to capture the complex and time-varying dynamics present in the state processes, and using DNNs with their non-linear mapping will boost the model prediction power. Our proposed framework shows how DNNs and B-DNNs can be incorporated into the modeling framework, and how they can be trained as well. A challenge with DNNs is their higher level of flexibility, which will lead to an overfitting problem. To address this, we discussed how to utilized B-DNN in our dynamical auto-encoder model. We also provided the training procedure – Appendix A – to find the posterior distribution of DNN weights, which to the best of authors' knowledge is novel and can help to build more robust decoder models.

In developing the methodology, we hypothesized that there is a dynamical low-dimensional manifold – state-process – present in the data, where its estimation will help to better understand the relationship between complex and distributed brain dynamics and behavioral readout. In our solution, we assumed that the dimension of the state is pre-known; however, identifying the dimension of the state process is of great interest. There are fewer established solutions that provide principle solutions to find the dimension of the state. In the auto-encoder model, we set the dimension of the state as a part of the behavioral observation process model. However, it is still of great importance to assess the optimal dimension of the state process. Note that we derive the posterior distribution of the state given the data, and we can study attributes of the posterior distribution to increase/decrease dimension of the state. Another possible solution is defining multiple state processes with independent noise processes; we also set a sparse prior on these states' noise processes and use the Bayesian posterior to find posteiro of their noise variance. The noise variance posterior will help in search for optimal dimension. This will be one of our future research directions.

For D-D direct and auto-encoder model, we focused on deriving model identification and state estimation processes. We used BIC and other model selection algorithms to pick a proper history term for the observation processes. However, we did not discuss the goodness-of-fit process to better examine the extent of the model fit to data. Utilizing simulation data and using goodness-of-fit techniques to better assess the model fit is another research direction we pursue to better address the pros and cons of proposed framework.

Here, we mainly focused on the model development and how the proposed techniques will help to analyze complex and high dimensional data like the conversation dataset. Understandingthe neural mechanisms of those cognitive processes has significant prioritiess for experimental and clinincal neuroscientist, which is not discussed in this research. We puruse another research to better understand underlying neural mechanisms of intention using the inference being drawn from our modeling results.

Acknowledgments

We thank Prof. Loren Frank and Dr. Anna K. Gillespie from the University of California at San

Francisco who provided spiking activity dataset for decoding movement trajectory problem (sections 3.A and 3.B).

References

1. Lebedev, M.A., and M.A. Nicolelis, *Brain-machine interfaces: From basic science to neuroprostheses and neurorehabilitation.* Physiological Reviews, 2017. **97**(2): p. 767-837.
2. Stevenson, I.H. and K.P. Kording, *How advances in neural recording affect data analysis.* Nature Neuroscience, 2011. **14**(2): p. 139.
3. Jorgenson, L.A., et al., *The BRAIN Initiative: developing technology to catalyse neuroscience discovery.* Philosophical Transactions of the Royal Society B: Biological Sciences, 2015. **370**(1668): p. 20140164.
4. Krakauer, J.W., et al., *Neuroscience needs behavior: correcting a reductionist bias.* Neuron, 2017. **93**(3): p. 480-490.
5. Wang, C. and M.M. Shanechi, *Estimating multiscale direct causality graphs in neural spike-field networks.* IEEE Transactions on Neural Systems and Rehabilitation Engineering, 2019. **27**(5): p. 857-866.
6. Abbaspourazad, H., H.-L. Hsieh, and M.M. Shanechi, *A multiscale dynamical modeling and identification framework for spike-field activity.* IEEE Transactions on Neural Systems and Rehabilitation Engineering, 2019. **27**(6): p. 1128-1138.
7. Hsieh, H.-L., et al., *Multiscale modeling and decoding algorithms for spike-field activity.* Journal of neural engineering, 2018. **16**(1): p. 016018.
8. Truccolo, W., et al., *A point process framework for relating neural spiking activity to spiking history, neural ensemble, and extrinsic covariate effects.* 2005. **93**(2): p. 1074-1089.
9. Paninski, L., J. Pillow, and J.J.P.i.b.r. Lewi, *Statistical models for neural encoding, decoding, and optimal stimulus design.* 2007. **165**: p. 493-507.
10. Coleman, T.P., et al., *A mixed-filter algorithm for dynamically tracking learning from multiple behavioral and neurophysiological measures.* The dynamic brain: An exploration of neuronal variability and its functional significance, 2011: p. 1-16.
11. Glaser, J.I., et al., *Machine learning for neural decoding.* 2017.
12. Rezaei, M.R., et al. *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)*. 2018. IEEE.
13. Yousefi, A., et al., *Real-Time Point Process Filter for Multidimensional Decoding Problems Using Mixture Models.* 2018: p. 505289.
14. Brown, E.N., et al., *A statistical paradigm for neural spike train decoding applied to position prediction from ensemble firing patterns of rat hippocampal place cells.* 1998. **18**(18): p. 7411-7425.
15. Eden, U.T., et al., *Dynamic analysis of neural encoding by point process adaptive filtering.* 2004. **16**(5): p. 971-998.

16. Park, I.M., et al., *Encoding and decoding in parietal cortex during sensorimotor decision-making.* 2014. **17**(10): p. 1395.
17. Prerau, M.J., et al., *A mixed filter algorithm for cognitive state estimation from simultaneously recorded continuous and binary measures of performance.* Biological cybernetics, 2008. **99**(1): p. 1-14.
18. Vyas, S., et al., *Neural population dynamics underlying motor learning transfer.* Neuron, 2018. **97**(5): p. 1177-1186. e3.
19. Yousefi, A., et al., *Decoding hidden cognitive states from behavior and physiology using a bayesian approach.* Neural computation, 2019. **31**(9): p. 1751-1788.
20. Burkhart, M.C., et al., *The Discriminative Kalman Filter for Bayesian Filtering with Nonlinear and Nongaussian Observation Models.* 2020. **32**(5): p. 969-1017.
21. Yousefi, A., et al. *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)*. 2015. IEEE.
22. Rezaei, M.R., B. Nazari, and S. Sadri, *Deep Recurrent Neural Network and Point Process Filter Approaches in Multidimensional Neural Decoding Problems.* bioRxiv, 2020.
23. Najafabadi, M.M., et al., *Deep learning applications and challenges in big data analytics.* Journal of big data, 2015. **2**(1): p. 1-21.
24. Srivastava, N., et al., *Dropout: a simple way to prevent neural networks from overfitting.* The journal of machine learning research, 2014. **15**(1): p. 1929-1958.
25. Scardapane, S., et al., *Group sparse regularization for deep neural networks.* Neurocomputing, 2017. **241**: p. 81-89.
26. Zhu, Y. and N. Zabaras, *Bayesian deep convolutional encoder–decoder networks for surrogate modeling and uncertainty quantification.* Journal of Computational Physics, 2018. **366**: p. 415-447.
27. Gal, Y., R. Islam, and Z. Ghahramani. *Deep bayesian active learning with image data*. in *International Conference on Machine Learning*. 2017. PMLR.
28. Gal, Y. and Z. Ghahramani, *Bayesian convolutional neural networks with Bernoulli approximate variational inference.* arXiv preprint arXiv:1506.02158, 2015.
29. Yousefi, A., et al., *Efficient decoding of multi-dimensional signals from population spiking activity using a Gaussian mixture particle filter.* 2019. **66**(12): p. 3486-3498.
30. Rezaei, M.R., et al., *Real-Time Point Process Filter for Multidimensional Decoding Problems Using Mixture Models.* Journal of Neuroscience Methods, 2020. **348**.
31. Shine, J.M., et al., *The low dimensional dynamic and integrative core of cognition in the human brain.* bioRxiv, 2018: p. 266635.
32. Pillow, J.W. and E.P. Simoncelli, *Dimensionality reduction in neural models: an information-theoretic generalization of spike-triggered average and covariance analysis.* Journal of vision, 2006. **6**(4): p. 9-9.
33. Byron, M.Y., et al. *Gaussian-process factor analysis for low-dimensional single-trial analysis of neural population activity*. in *Advances in neural information processing systems*. 2009.
34. Churchland, M.M., et al., *Techniques for extracting single-trial activity patterns from large-scale neural recordings.* Current opinion in neurobiology, 2007. **17**(5): p. 609-618.
35. Byron, M.Y., et al., *Mixture of trajectory models for neural decoding of goal-directed*

movements. 2007.

36. Chen, Z.J.S., *Bayesian filtering: From Kalman filters to particle filters, and beyond.* 2003. **182**(1): p. 1-69.

37. Koks, D. and S. Challa, *An introduction to Bayesian and Dempster-Shafer data fusion*. 2003, DEFENCE SCIENCE AND TECHNOLOGY ORGANISATION SALISBURY (AUSTRALIA) SYSTEMS ….

38. Ogutu, J.O., T. Schulz-Streeck, and H.-P. Piepho. *Genomic selection using regularized linear regression models: ridge regression, lasso, elastic net and their extensions*. in *BMC proceedings*. 2012. Springer.

39. Dempster, A.P., N.M. Laird, and D.B.J.J.o.t.R.S.S.S.B. Rubin, *Maximum likelihood from incomplete data via the EM algorithm.* 1977. **39**(1): p. 1-22.

40. Kingma, D.P. and M. Welling, *Auto-encoding variational bayes.* arXiv preprint arXiv:1312.6114, 2013.

41. Van Dyk, D.A.J.J.o.C. and G. Statistics, *Fitting mixed-effects models using efficient EM-type algorithms.* 2000. **9**(1): p. 78-98.

42. Ruder, S.J.a.p.a., *An overview of gradient descent optimization algorithms.* 2016.

43. Särkkä, S., *Bayesian filtering and smoothing*. 2013: Cambridge University Press.

44. Doucet, A., S. Godsill, and C. Andrieu, *On sequential Monte Carlo sampling methods for Bayesian filtering.* Statistics and computing, 2000. **10**(3): p. 197-208.

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

46. Sun, J., N.-N. Zheng, and H.-Y. Shum, *Stereo matching using belief propagation.* IEEE Transactions on pattern analysis and machine intelligence, 2003. **25**(7): p. 787-800.

47. Bastos, A.M. and J.-M.J.F.i.s.n. Schoffelen, *A tutorial review of functional connectivity analysis methods and their interpretational pitfalls.* 2016. **9**: p. 175.

48. Joseph, L., D.B. Wolfson, and R.D. Berger, *Sample size calculations for binomial proportions via highest posterior density intervals.* Journal of the Royal Statistical Society: Series D (The Statistician), 1995. **44**(2): p. 143-154.

49. Fonti, V. and E. Belitser, *Feature selection using lasso.* VU Amsterdam Research Paper in Business Analytics, 2017. **30**: p. 1-25.

50. Yousefi, A., et al., *Assessing Goodness-of-Fit in Marked Point Process Models of Neural Population Coding via Time and Rate Rescaling.* Neural Computation, 2020. **32**(11): p. 2145-2186.

51. Blundell, C., et al., *Weight uncertainty in neural networks.* arXiv preprint arXiv:1505.05424, 2015.

Appendix A. Moving to a Fully B-DNN

Let's assume that there is a prior probability over the network parameters $\boldsymbol{\Omega}$, $p(\boldsymbol{\Omega})$. Now, we require to build the joint posterior over the states value and the DNN model parameters defined by $p(\boldsymbol{x}_{0:K}, \boldsymbol{\Omega} | \boldsymbol{s}_{1:K}, \boldsymbol{z}_{1:K})$. The posterior distribution is proportionate to full likelihood of the observation which can be defined by

$$p(\boldsymbol{x}_{0:K}, \boldsymbol{\Omega} | \boldsymbol{s}_{1:K}, \boldsymbol{z}_{1:K}) \propto p(\boldsymbol{x_0}) \prod_{k=1}^{K} p(\boldsymbol{x_{k+1}} | \boldsymbol{x_k}) \times p(\boldsymbol{z_k} | \boldsymbol{x_k}) \times p(\boldsymbol{s_k} | \boldsymbol{x_k}, \boldsymbol{h_k}) \times p(\boldsymbol{\Omega})$$

$$= p(\boldsymbol{x_0}) \times \prod_{k=1}^{K} p(\boldsymbol{x_{k+1}}|\boldsymbol{x_k}) \times p(\boldsymbol{z_k}|\boldsymbol{x_k}) \times \frac{p(\boldsymbol{x_k}|\boldsymbol{s_k}, \boldsymbol{h_k}; \boldsymbol{\Omega})}{p(\boldsymbol{x_k}|\boldsymbol{h_k}; \boldsymbol{\Omega})} \times p(\boldsymbol{\Omega}) \tag{A.1}$$

where $\boldsymbol{z_k}$ and $\boldsymbol{s_k}$ are the neural activity and the behavioral observations at the time index k; respectively. Note that constant term will be $p(\boldsymbol{s}_{1:K}, \boldsymbol{z}_{1:K})$; as we see later, this term is not needed in our DNN training process.

Given the fact that the exact solution for $p(\boldsymbol{x}_{0:K}, \boldsymbol{\Omega}|\boldsymbol{s}_{1:K}, \boldsymbol{z}_{1:K})$ due to a large dimension of $\boldsymbol{\Omega}$ is intractable, we can approximate the true posterior $p(\boldsymbol{x}_{0:K}, \boldsymbol{\Omega}|\boldsymbol{s}_{1:K}, \boldsymbol{z}_{1:K})$ by a variational distribution defined by

$$p(\boldsymbol{x}_{0:K}, \boldsymbol{\Omega}|\boldsymbol{s}_{1:K}, \boldsymbol{z}_{1:K}) \sim q(\boldsymbol{x}_{0:K}|\boldsymbol{s}_{1:K}, \boldsymbol{z}_{1:K}, \boldsymbol{\Omega}) \times q(\boldsymbol{\Omega}) \tag{A.2}$$

Where $q(\boldsymbol{\Omega})$ has a known functional form with a set of fixed parameters. Note that, we have already derived the solution for the first term on the right side of equation (A.2), which is the smoother solution with a known model parameter set. Equation (7) provides the smoother solution with neural observation; this solution can be extended for observation process including both neural and behavioral data. To find the proper set of parameters for the DNN weights' distribution, we can minimize the Kullback-Leibler (KL) divergence between $q(\boldsymbol{x}_{0:K}|\boldsymbol{s}_{1:K}, \boldsymbol{z}_{1:K}, \boldsymbol{\Omega}) \times q(\boldsymbol{\Omega})$ and the true posterior $p(\boldsymbol{x}_{0:K}, \boldsymbol{\Omega}|\boldsymbol{s}_{1:K}, \boldsymbol{z}_{1:K})$ [51]. The optimization corresponds to minimizing the following metrics

$$\mathrm{KL}(q(\boldsymbol{x}_{0:K}|\boldsymbol{s}_{1:K}, \boldsymbol{z}_{1:K}, \boldsymbol{\Omega}) \times q(\boldsymbol{\Omega}) \parallel p(\boldsymbol{x}_{0:K}, \boldsymbol{\Omega}|\boldsymbol{s}_{1:K}, \boldsymbol{z}_{1:K})) \tag{A.3}$$

where, we require to find new set of paramaters for a specific functional form of $q(\boldsymbol{\Omega})$. As we discussed $q(\boldsymbol{x}_{0:K}|\boldsymbol{s}_{1:K}, \boldsymbol{z}_{1:K}, \boldsymbol{\Omega})$ can be replaced by $p(\boldsymbol{x}_{0:K}|\boldsymbol{s}_{1:K}, \boldsymbol{z}_{1:K}, \boldsymbol{\Omega})$, which we already have a solution for it.
The KL described in equation (A.2) can be written as

$$\begin{aligned}\mathrm{KL}(p(\boldsymbol{x}_{0:K}|\boldsymbol{s}_{1:K}, \boldsymbol{z}_{1:K}, \boldsymbol{\Omega}) \times q(\boldsymbol{\Omega}) \parallel p(\boldsymbol{x}_{0:K}, \boldsymbol{\Omega}|\boldsymbol{s}_{1:K}, \boldsymbol{z}_{1:K}))\\ = \mathrm{E}_{p(\boldsymbol{x}_{0:K}|\boldsymbol{s}_{1:K}, \boldsymbol{z}_{1:K}, \boldsymbol{\Omega})q(\boldsymbol{\Omega})}\left[\log \frac{p(\boldsymbol{x}_{0:K}|\boldsymbol{s}_{1:K}, \boldsymbol{z}_{1:K}, \boldsymbol{\Omega})q(\boldsymbol{\Omega})}{p(\boldsymbol{x}_{0:K}, \boldsymbol{\Omega}|\boldsymbol{s}_{1:K}, \boldsymbol{z}_{1:K})}\right]\end{aligned} \tag{A.4}$$

We can rewrite the KL distance as – moving forward, for simplicity, we use KL without its arguments.

$$\begin{aligned}\mathrm{KL} = \mathrm{E}_{q(\boldsymbol{\Omega})}[\log q(\boldsymbol{\Omega})] \ldots\\ +\mathrm{E}_{q(\boldsymbol{\Omega})}\mathrm{E}_{p(\boldsymbol{x}_{0:K}|\boldsymbol{s}_{1:K}, \boldsymbol{z}_{1:K}, \boldsymbol{\Omega})}\left[\log \frac{p(\boldsymbol{x}_{0:K}|\boldsymbol{s}_{1:K}, \boldsymbol{z}_{1:K}, \boldsymbol{\Omega})}{p(\boldsymbol{x}_{0:K}, \boldsymbol{\Omega}|\boldsymbol{s}_{1:K}, \boldsymbol{z}_{1:K})}\right]\end{aligned} \tag{A.5}$$

We can replace $p(\boldsymbol{x}_{0:K}, \boldsymbol{\Omega}|\boldsymbol{s}_{1:K}, \boldsymbol{z}_{1:K})$ term using equation (A.1).

$$\begin{aligned}\mathrm{KL} = \mathrm{E}_{q(\boldsymbol{\Omega})}[\log q(\boldsymbol{\Omega})] \ldots\\ +\mathrm{E}_{q(\boldsymbol{\Omega})}\{\mathrm{E}_{p(\boldsymbol{x}_{0:K}|\boldsymbol{s}_{1:K}, \boldsymbol{z}_{1:K}, \boldsymbol{\Omega})}[\log p(\boldsymbol{x}_{0:K}|\boldsymbol{s}_{1:K}, \boldsymbol{z}_{1:K}, \boldsymbol{\Omega}) - \log p(\boldsymbol{\Omega}) -\\ \textstyle\sum_{k=1}^{K} \log p(\boldsymbol{x_k}|\boldsymbol{s_k}, \boldsymbol{h_k}, \boldsymbol{\Omega}) + \sum_{k=1}^{K} \log \int \mathrm{p}(\boldsymbol{x}_\mathrm{k}|\boldsymbol{x}_{\mathrm{k}-1})\, \mathrm{p}(\boldsymbol{x}_{\mathrm{k}-1}|\boldsymbol{s}_{\mathrm{k}-1}, \boldsymbol{h}_{\mathrm{k}-1}, \boldsymbol{\Omega}) \mathrm{d}\boldsymbol{x}_{\mathrm{k}-1}]\} + \mathbf{C}\end{aligned} \tag{A.6}$$

where, the terms which are not a function of the DNN weights' parameters are represented by C. Let's assume that $\boldsymbol{\Omega}$ distribution paramaters are defined by $\boldsymbol{\omega}$ and $\boldsymbol{\omega}^{(\mathbf{0})}$ is our initial guess of these parameters defining $p(\mathbf{\Omega}; \boldsymbol{\omega}^{(\mathbf{0})})$. Our objective is to minimize KL by tuning $\boldsymbol{\omega}$ parameters along with other paramaters of the auto-encoder model. We optimize other paramaters of the model using the iterative EM algoirthm. As a result, KL needs to be minimized iteratively as the model paramaters and state estimation are updated per eahc iteration of EM step. With this assumption, we can rewrite the KL by

$$\mathrm{KL}^{(\mathrm{r})} = \mathrm{E}_{q(\mathbf{\Omega};\boldsymbol{\omega})}\left[\log q(\mathbf{\Omega};\boldsymbol{\omega}) + \mathrm{E}_{p(\boldsymbol{x}_{0:K}|\boldsymbol{s}_{1:K}, \boldsymbol{z}_{1:K}, \mathbf{\Omega})}[\log p(\boldsymbol{x}_{0:K}|\boldsymbol{s}_{1:K}, \boldsymbol{z}_{1:K}, \mathbf{\Omega})]\right] \dots$$
$$-\mathrm{E}_{q(\mathbf{\Omega};\boldsymbol{\omega})}\mathrm{E}_{p(\boldsymbol{x}_{0:K}|\boldsymbol{s}_{1:K}, \boldsymbol{z}_{1:K}, \mathbf{\Omega})}\Big[-\log p(\mathbf{\Omega}; \boldsymbol{\omega}^{(\mathbf{0})}) + \textstyle\sum_{k=1}^{K}\log p(\boldsymbol{x}_k|\boldsymbol{s}_k, \boldsymbol{h}_k, \mathbf{\Omega}) +$$
$$\textstyle\sum_{k=1}^{K}\log\int \mathrm{p}(\boldsymbol{x}_\mathrm{k}|\boldsymbol{x}_{\mathrm{k}-1})\, \mathrm{p}(\boldsymbol{x}_{\mathrm{k}-1}|\boldsymbol{s}_{\mathrm{k}-1}, \boldsymbol{h}_{\mathrm{k}-1}, \mathbf{\Omega})\mathrm{d}\boldsymbol{x}_{\mathrm{k}-1}\Big] + \mathbf{C} \quad \text{(A.7)}$$

To minimize $\mathrm{KL}^{(\mathrm{r})}$, we require to find a new set of paramaters which are called, $\boldsymbol{\omega}^{(\boldsymbol{r})}$. Solving this optimization problem is computationally interactable; as a result, we provide a suboptimal solution using the EM solution we already proposed to find ML estimate of the DNN weights. As we discussed, this solution might be a proper apporach when the numer of weights in DNN is limited or the DNN weights are correlated. The solution is similar to a coordinate descent optimization, where we assume the auto-encoder models are updated recursively. Some of recent efforts to address Bayesian learning in dynamical auto-encoder model can be found in [40].

Let's assume we have $q(\mathbf{\Omega}; \boldsymbol{\omega}^{(\boldsymbol{r})})$ for the r^{th} iteration; we can define

$$p(\boldsymbol{x}_{0:K}|\boldsymbol{s}_{1:K}, \boldsymbol{z}_{1:K}) = \int p(\boldsymbol{x}_{0:K}|\boldsymbol{s}_{1:K}, \boldsymbol{z}_{1:K}, \mathbf{\Omega})\ q(\mathbf{\Omega}; \boldsymbol{\omega}^{(\boldsymbol{r})})\ d\mathbf{\Omega} \quad \text{(A.8)}$$

Let's assume, we draw *D* samples from $q(\mathbf{\Omega}; \boldsymbol{\omega}^{(\boldsymbol{r})})$, $d = 1 \dots D$; using these samples, we can find the posterior estimation of state given behavioral and neural data. Note that, $\mathbf{\Omega}_\mathbf{d}^{(r)}$ is d^{th} sample of DNN network weights at iteration r. In other word, we have *D* DNN networks, where for each network, we can have $p\left(\boldsymbol{x}_{0:K}\middle|\boldsymbol{s}_{1:K}, \boldsymbol{z}_{1:K}, \mathbf{\Omega}_\mathbf{d}^{(r)}\right)$ and the state posterior is defined as weighted sum of these distribution

$$p(\boldsymbol{x}_{0:K}|\boldsymbol{s}_{1:K}, \boldsymbol{z}_{1:K}) = \textstyle\sum_{d=1}^{D} p\left(\boldsymbol{x}_{0:K}\middle|\boldsymbol{s}_{1:K}, \boldsymbol{z}_{1:K}, \mathbf{\Omega}_\mathbf{d}^{(r)}\right) q\left(\mathbf{\Omega}_\mathbf{d}^{(r)}; \boldsymbol{\omega}^{(r)}\right) \Big/ \textstyle\sum_{d=1}^{D} q\left(\mathbf{\Omega}_\mathbf{d}^{(r)}; \boldsymbol{\omega}^{(r)}\right) \quad \text{(A.9)}$$

To calculate (A.9), we require to have $q(\mathbf{\Omega}; \boldsymbol{\omega}^{(r)})$.

To find $q(\mathbf{\Omega}; \boldsymbol{\omega}^{(\boldsymbol{r})})$, we assume there are *D* dynamical auto-ecnoder models and each model is trained using the EM alogithm defined in section **2.D**. Note that each DNN is trained using a regularizaion term defined by $p(\mathbf{\Omega}; \boldsymbol{\omega}^{(\mathbf{0})})$ – adding $p(\mathbf{\Omega}; \boldsymbol{\omega}^{(\mathbf{0})})$ to EM algorithm is straight forward. It is worth to mention that DNN training at iteration *r* can start with weights estimated at iteration $r - 1$. Training *D* DNN will provide new set of paramaters $\mathbf{\Omega}_\mathbf{d}^{(r)}$ for each iteration; we can find updated paramaters for $\boldsymbol{\omega}^{(\boldsymbol{r})}$ through *D* DNN models, defined by

$$\boldsymbol{w}^{(r)} = \mathrm{argmin}_w\ \mathrm{KL}^{(\mathrm{r})}$$
$$= \mathrm{argmin}_w\ \textstyle\sum_{d=1}^{D} q\left(\mathbf{\Omega}_\mathbf{d}^{(r)}; \boldsymbol{\omega}\right)\left[\log q\left(\mathbf{\Omega}_\mathbf{d}^{(r)}; \boldsymbol{\omega}\right) - \log p\left(\mathbf{\Omega}_\mathbf{d}^{(r)}; \boldsymbol{\omega}^{(\mathbf{0})}\right) - \mathrm{Q}_\mathrm{d}^{(r)}\right] \quad \text{(A.10)}$$

where, $\mathrm{Q}_\mathrm{d}^{(r)}$ is defined by equation (22).

As a result, our B-DNN training includes *D* auto-encoder model trained in parallel. Using (A.10), we can find approximate posterior for DNN weights, and use that posterior to build a more robust posterior distribution of the state. When we use full B-DNN as part of our EM algorithm, we can replace the posteiror distribution in EM with the postrior distribution defined in equation (A.9), and use this distribution to find the state process and behavioral model paramaters.