A Novel Dynamic Deep Decoder Model for High-Dimensional Neural Data

Sasoun Krikorian
Dept. of Computer Science
Worcester Polytechnic Institute
Worcester, MA, USA
smkrikorian@wpi.edu

Maceo Richards
Dept. of Computer Science
Worcester Polytechnic Institute
Worcester, MA, USA
mdrichards@wpi.edu

Navid Ziaei
Dept. of Electrical & Computer Engineering
Isfahan University of Technology
Isfahan, Iran
n.ziaei@iut.ac.ir

Loren Frank
Dept. of Physiology
University of California San Francisco
San Francisco, CA, USA
loren@phy.ucsf.edu

Uri Eden
Dept. of Mathematics and Statistics
Boston University
Boston, MA, USA
tzvi@bu.edu

Ali Yousefi
Dept. of Computer Science
Worcester Polytechnic Institute
Worcester, MA, USA
ayousefi@wpi.edu

***Abstract*—Decoding neural activity into useful information poses a modeling challenge, given the data's inherent noise and high dimensionality. Recent advancements in machine learning (ML) have introduced novel decoder models that leverage the expressiveness of deep neural network models. Among these models, the Direct Discriminative Decoder (DDD) has been recently proposed as a potential means for precisely decoding dynamical latent variables within high-dimensional neural data. The formulation of DDD relies on replacing the observation process in state-space models with a discriminative process that directly predicts the conditional distribution of latent variables given observed data at current and previous time points. Despite its preceding accuracy in prediction tasks, the framework's implementation suggests a tendency to make inaccurate predictions for the latent variable, resulting in varying decoding performance. In this research, we explore possible solutions to avoid inaccurate predictions and investigate these solutions in both simulated and real datasets. We demonstrate that the refined model alleviates the robustness of decoding performance, further showcasing the utility of advanced ML in the analysis of high-dimensional neural data.**

***Index Terms*—Neural Decoding, Time-Series Prediction, Deep Learning, Hippocampus, Place Cells**

I. INTRODUCTION

Many neuroscience experiments yield high-dimensional recordings of neural activity that represent underlying dynamic processes, for example, movement and learning. For instance, in neural recordings of rat hippocampus place cells, we are interested in decoding rat movement trajectories, or from the motor cortex neural recordings, we are interested in decoding projected arm or limb movements. Established solutions such as a linear state-space model (SSM) and point-process SSM [1] show promising results in decoding the latent processes representing movement or learning state vs neural recordings.

The SSM framework is built upon multiple modeling assumptions, such as conditional independence of current observed data from the previous time data given the state. This assumption does not hold in most neuroscience experiments, given the dependency of neural activity on previous time activities; thus, this assumption introduces bias in inferences and predictions. Given the high dimension of observed data, we might need other assumptions like the independence of neural recording channels, which might not be valid as neural nodes interact in shaping functions.

Recent advancements in the field of machine learning have introduced novel algorithms such as Direct Discriminative Decoder (DDD) [2], which leverage advantages of both deep neural networks (DNN) and the SSM to address modeling challenges pointed out before. Models such as DDD introduce a new framework that is scalable to large recordings and provides decent decoding accuracy and faster processing time in the decoding tasks. Despite these advancements, these methods carry their own challenges and need remedies to make their performance more robust and generalizable across datasets. In particular, there is a potential for numerical stability in model likelihood estimation of state-given neural data, which can significantly degrade the DDD decoding performance.

In this research, we introduce different implementations of the DDD model and evaluate their performance in both simulated and neural data. In Deep Direct Discriminative Decoder (D4), a framework implementation using two neural networks is suggested [2]. As we pointed out before, there are different ways that DDD can be implemented. We would like to know which method is the most robust and better handles the numerical instability in the likelihood term estimation. Our main objective is to find the best method to ensure the robustness of the likelihood estimation while maintaining high accuracy.

In assessing different implementations of DDD, we will study different aspects of decoding accuracy, which includes mean squared error, 95% highest posterior density (HPD) region, and 95% HPD coverage [3]. We examine these measures in simulated data where the current observation is not necessarily independent of the previous observations given the current state. We also revisit the performance analysis in rat

hippocampus data recorded from place cells as it traverses a W-shaped maze.

The following sections discuss the DDD method in detail and show performance evaluations. We then discuss how the simulated data is being created and compare the performance of different implementations. We then reflect on our findings about the robust method in decoding a rat movement trajectory given place cell data.

II. METHOD

A. Definitions and DDD formulation

Let's assume $x_{0:T}$ is a sequence of latent variables such that $x_k \in \mathbb{R}^N$ for $k = 0, 1, \ldots, T$, and $y_{1:T}$ is a sequence of observed data such that $y_k \in \mathbb{R}^M$ for $k = 0, 1, \ldots, T$ — in general, $M \gg N$. Let h_k, a suffix of $y_{1:k-1}$, represent a fixed-length history with length d. Under the DDD assumptions, two models are used to characterize the dynamics of $x_{0:T}$, which are defined as

$$x_k|x_{k-1} \sim g\left(x_{k-1}; \Omega\right) \quad \text{(state process)} \quad (1)$$

$$x_k|y_k, h_k \sim f\left(y_k, h_k; \Theta\right) \quad \text{(prediction process)} \quad (2)$$

where both Ω and Θ are a set of model-free parameters. In DDD, the state process is assumed to be Markovian. With Baye's rule and the Markovian assumption, a filter solution is defined by

$$p\left(x_{0:k}|y_k, h_k\right) = \frac{p\left(x_k|y_k, h_k\right)}{p\left(x_k|h_k\right)} p\left(x_k|x_{k-1}\right) p\left(x_{0:k-1}|y_{k-1}, h_{k-1}\right) \quad (3)$$

This solution is similar to the solution in a traditional SSM. Here, the ratio $\frac{p(x_k|y_k,h_k)}{p(x_k|h_k)}$ corresponds to the likelihood term, which, in an SSM, would be $p\left(y_k|x_k\right)$, which is defined by the observation process. Using Equation (3), the posterior distribution of $x_{0:k}$ can be inferred using sampling techniques such as Markov Chain Monte Carlo (MCMC) [4].

The main advantage of DDD over SSM is that DDD does not need to build $p\left(y_k|x_k\right)$, which becomes challenging to compute when the data is high-dimensional. Note that DDD directly estimates $x_{0:k}$ as a function of current and a subset of previous time activity; thus, we expect to reach a higher accuracy in predicting the underlying state process. Moreover, DDD grants the flexibility to choose how to build the prediction process. For instance, one could use a DNN to compute the numerator and another DNN to build the prediction process.

While DDD does not need to compute $p\left(y_k|x_k\right)$, it needs to compute $p\left(x_k|h_k\right)$ along with $p\left(x_k|y_k, h_k\right)$ and take their ratio, which could lead to numerical instability. For instance, when the mode of numerator and denominator distributions are different, the likelihood of x_k might significantly grow for the range of x_k not presented by either of the modes. To address these shortcomings of DDD, we explore possible ways of computing the denominator to find a solution that is the most robust and maintains high prediction accuracy.

TABLE I: Performance results using different variants of DDD in simulated Data

method	mse	hpd%	hpd range
kalman	19.15±22.17	65.21%±24.76%	8.16±0.77
$p(x_k\|y_k, h_k)$	1.75±11.91	93.34%±16.70%	3.73±0.63
method 1	1.08±4.01	87.67%±19.59%	2.75±0.72
method 2	1.06±4.06	88.28%±19.91%	2.77±0.74
method 3	1.17±3.33	89.91%±20.40%	3.30±1.19
method 4	1.38±3.23	88.41%±19.98%	3.49±1.54
method 5	1.52±4.12	83.97%±26.32%	2.69±0.84

B. Robust Denominator Solution

The denominator term is the conditional probability of the state given the history of neural activity, ignoring the current observation. There are different approximations for this term, and we want to identify robust approximations and detour the numerical degeneracy that introduces larger variability in the decoding step, such as numerical degeneracy or large variance in the prediction. The five potential approximations are listed as follows:

Method 1: Set $p\left(x_k|h_k\right) \coloneqq 1$ (disregard denominator)

Method 2: separate model to predict $p\left(x_k|h_k\right)$

Method 3: $p\left(x_k|h_k\right) \approx \int p(x_k|x_{k-1}) p\left(x_{k-1}|y_{1:k-1}\right)\, dx$ (one-step prediction using the full history)

Method 4: $p\left(x_k|h_k\right) \approx \int p(x_k|x_{k-1}) p\left(x_{k-1}|y_{k-1}, h_{k-1}\right)\, dx$ (one-step prediction using previous time process)

Method 5: $p\left(x_k|h_k\right) = \int p(x_k|y_k, h_k) p\left(y_k|h_k\right)\, dy_k$ (marginal distribution using using $p(xk|yk, hk)$

We discussed an MCMC method for a filter solution under DDD. Note that we can use particles in the filter solution to calculate these terms presented in the five methods numerically. In implementing MCMC, we have an extra term that will change the weight of each particle given the method we pick. Thus, we expect different filter estimates using different approximations of the denominator.

III. DDD ON SIMULATED DATA

For the simulation, we assumed the state process is one-dimensional (N=1) and the observation dimension is 4 (M=4). The observation process is defined by the sum of the current state, the observation noise, and a nonlinear transformation of the previous two observations. For the simulation, we assume each measurement noise for different observations is uncorrelated from other channels. The first 50,000 simulation observations are used for training the DNNs. The last 300 observations are used for testing. We run the Kalman filter along with the DDD model, examining five different approximations for the denominator calculation. We use three metrics to assess prediction accuracy: 95%

Note that the observation is a function of current and previous states in the simulated data. Thus, we expect the Kalman filter to be suboptimal. For the DDD, we set history

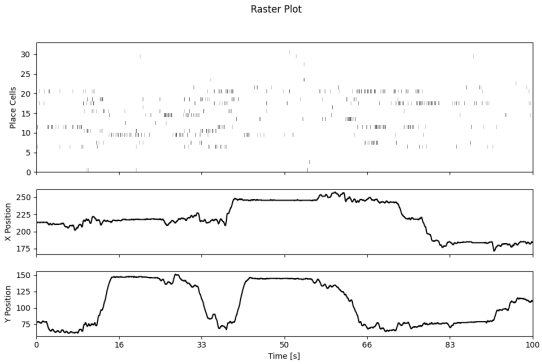

Fig. 1: Raster plot of place cell spiking activity for the test dataset. The rat traverses a maze and a camera captures its location. Recordings are extracted from 33 place cells as the rat traverses the maze.

length to be 1, a suboptimal choice. This choice corroborates the assumption of DDD; using the prediction process, we will capture intermediate and mostly suboptimal temporal dependency between state and observed data [2]. Thus, we expect the Kalman filter to perform worse in the simulation. We ran the simulation 100 times with different observation and noise values. Table I shows the performance measures for different approaches.

Given the results, we see Methods 1 and 2 provide a balance between MSE and HPD measures. DDD performs better than simply using the prediction process model alone; this is expected, as the state process will carry longer time information about the latent variable. For Method 2, we see a reduction in MSE with a bit of growth in the HPD range, which is unsurprising, given the denominator is less certain. In the simulation, the denominator term will create more spread over the state space than the numerator since the denominator cannot access the current observation.

Methods 3 and 4 occasionally push the likelihood peak to a new latent space that does not conform to the prediction process (numerator) and the one-step prediction (denominator). Method 5 is computationally expensive given that we need to build a mapping from h_k to y_k, both of which are high dimensional and will require having to sample the y_k's

IV. DDD ON RAT HIPPOCAMPUS DATA

An application of revised DDD is explored in decoding rat hippocampus data. The objective is to decode the rat movement trajectory given the neural activity of place cell ensembles.

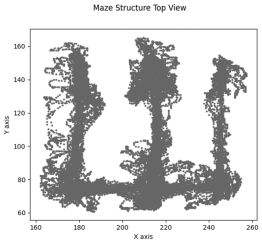

Fig. 2: Top view of the w-shaped maze.

This dataset consists of intracortical recordings collected from 33 neurons within an ensemble of place cells in the rat hippocampus. The experiment length is approximately 1,850 seconds, of which the first 1,750 are used for training. DDD is used to decode a two-dimensional latent state that represents the rat position. It is worth noting DDD can be used in decoding position and movement speed, which then characterizes a four-dimensional latent state. The maze is shaped like the letter "W", with three arms and one base area. Figure 1 shows the raster plot of place cell spiking activity and Figure 2 shows a top view of the maze.

For this application, the prediction process $p\left(x_k|y_k, h_k\right)$ is a DNN that produces Gaussian mixture model (GMM) parameters describing possible latent rat positions through the maze given the neural observations. The input to the prediction process is spiking activity at the current timestep and some history of recent spiking activity. The GMM includes five mixtures, where the DNN predicts the mixing coefficient, mean, and covariance of each mixture. The choice of five mixtures represents each of the three arms and both halves of the maze base, as we expect different movement dynamics to be observed at different maze regions. Using GMM for the prediction step allows for properly capturing spatial information encoded by place cells that are not specific to one position and represent different (non-overlapping) maze segments.

The state transition process $p\left(x_k|x_{k-1}\right)$ is characterized by a multivariate normal distribution such that $p\left(x_k|x_{k-1}\right) \sim \mathcal{N}\left(x_{k-1}, \Sigma_p\right)$ where the covariance structure Σ_q is determined by the natural covariance of the movement data used for training. In a future DDD implementation, this state transition process will be characterized by a DNN such that $p\left(x_k|x_{k-1}\right) \sim \mathcal{N}\left(x_{k-1}, \Sigma_p\left(x_{k-1}\right)\right)$, where the covariance structure is produced by a nonlinear function of the latent state at the previous timestep.

For the MCMC implementation, a proposal distribution $q\left(x_k|x_{k-1}\right) \sim \mathcal{N}\left(x_{k-1}, \Sigma_q\right)$ is used. The covariance structure of this proposal distribution Σ_q is an upscaling of Σ_p and promotes the decoder to explore new maze areas. In the

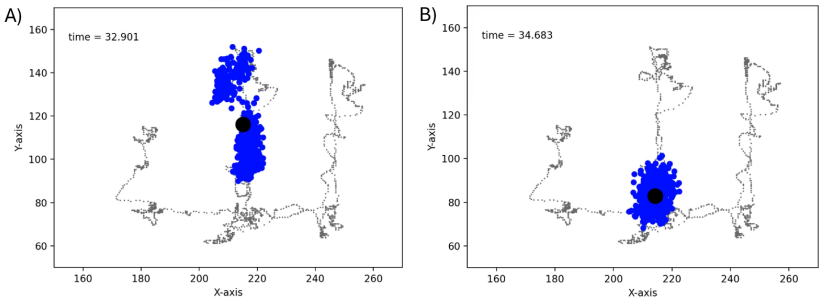

Fig. 3: Decoding results at two instances in the test dataset. The particle cloud reasonably captures the current position of the rat. In (A), particles also suggest decoding toward the top of the middle arm. This is a possible scenario as place cells demonstrate multimodal encoding mechanisms.

MCMC implementation, 1000 samples are used.

For the training, maximum likelihood estimation is used to learn the prediction process parameters with $p\left(x_k|y_k, h_k\right)$ as the objective function. The prediction process DNN and training scheme are implemented using the PyTorch machine learning library, which provides a comprehensive framework for building neural network models. The PyTorch library also handles gradient calculations and parameter updates during backpropagation.

Method 3 is used in this revised DDD implementation and characterizes the denominator term $p\left(x_k|h_k\right)$. This application demonstrates this method's robustness and high decoding accuracy. The decoder is tested in a data recording segment that does not overlap with recordings used for training. Figure 2 shows decoding results for two sample time points. The decoder yields a mean-squared error performance of 12.6614 $\pm$ 0.6008 cm. DDD has previously been applied in decoding rat movement trajectory. However, the revised solution for DDD demonstrates increased robustness and performance, showing the result from [2].

V. DISCUSSION

We observed that different methods will provide the best results depending on the situation. In the simulated data, where the probability densities of numerator and denominator models are unimodal, method 2 performed the best. We think this is because the ratio term was able to remain stable because the variance of the denominator was always larger than the variance of the numerator. However, in the case of multimodal probabilities, like in the rat data, method two did not perform as well, as it was a lot more sensitive to the prediction of the denominator. As a result, method 3 performed the best because the denominator was more stable. This suggests that, depending on the experiment, the user must determine which method would work. Thus, we might suggest exploring different methods to check which one can reach a more robust decoding performance.

Multiple nonlinear mechanisms, including a latent process and previous activity, generally drive neural activities. The simulated experiment is a simple example of this observation; we observed that the Kalman filter performs poorly in this case. The DDD decoding results are much better in general, suggesting the utility of DDD in decoding tasks where there are complex and temporal dependencies in observed neural data. The simulated data has a nonlinear dependency carried by neural activity at two previous time points; we build DDD models based on only one step history term. It is essential to explore a larger history term in building the prediction process and how the choice of history term will affect the decoding performance and robustness. We have a similar scenario for the rat decoder model; we heuristically searched the history term and picked a length of 8 time points, which is about 264 msec. To better understand the DDD solution, we need to examine how the length of the history term will change the prediction model training and decoding performance.

VI. FUTURE RESEARCH

The decoding results based on the place cell dataset show promising outcomes for the DDD framework in decoder tasks. The synthetic data and place cell decoding analytical results suggest two different approximations for the likelihood term in the DDD framework. Thus, we need to expand the simulation to a multidimensional state process and observation process with a multimodal distribution to understand further which approximation method will generalize better in different decoding problems. This will be the scope of our future research. We have other place cell datasets with much higher dimensions of recordings; we will study decoding results in these datasets to further examine other aspects of the DDD framework, such as its training, computational cost, and generalization.

REFERENCES

[1] U. T. Eden and E. N. Brown, "Continuous-time filters for state estimation from point process models of neural data," *Statistica Sinica*, vol. 18, no. 4, p. 1293, 2008.

[2] M. Rezaei, M. Popovic, U. Eden, M. Lankarany, and A. Yousefi, "Deep direct discriminative decoders for high-dimensional time-series data analysis," *Neurons, Behavior, Data analysis, and Theory*, pp. 1–22, 8 2023.

[3] R. J. Hyndman, "Computing and graphing highest density regions," *The American Statistician*, vol. 50, no. 2, pp. 120–126, 1996.

[4] A. Doucet and A. Johansen, "A tutorial on particle filtering and smoothing: Fifteen years later," *Handbook of Nonlinear Filtering*, vol. 12, 01 2009.