CS230

Using Deep Learning to Predict Locomotive Intention from Hippocampal Signals

Nicolai P. Ostberg
Department of Bioengineering
Stanford University
nostberg@stanford.edu

Max D. Melin
Department of Bioengineering
Stanford University
mmelin@stanford.edu

Abstract

Applying deep learning principles to the field of brain machine interface (BMI) has already resulted in several successful applications to decode neural activity. Most early successful cases utilize recordings from the motor cortex to decode arm movement in primates. This has proven a powerful paradigm for decoding upper limb movement, however, little work has been done to decode movement intention through two-dimensional space. Using hippocampal recordings from mice and a three-layer LSTM, we demonstrate that deep learning is a plausible approach to decode movement intention from the brain. Applying this technology in the clinical setting could enable patients with no motor function to perform tasks that require navigation through 2D space, such as controlling a wheelchair or driving a car.

1 Introduction

In the last decade, translational brain machine interface (BMI) research has made great strides to restore lost function to patients with neurological trauma or neurodegenerative disease. Using multi-electrode arrays (MEA's) and quickly improving algorithms, scientists can extract brain signals to control robot arms or computer cursors. Although early trials have seen encouraging results, these types of interfaces are limited in their scope because they generally rely on two brain areas: the premotor cortex and the primary motor cortex. These brain regions capture motor signals when the brain wants to move a part of the body, thus recording from these areas is highly suited to controlling a prosthetic arm or leg. Due to the limited size and invasiveness of electrode implants, recording from one electrode in motor cortex lacks sufficient information to control multiple the multiple body regions-the kinds of signals that are required to navigate through space.

Here, we propose that capturing higher order neural dynamics in the hippocampus may be used as a signal to decode spatial navigation intention via deep learning. The hippocampus has long been known to encode signals for spatial navigation and planning. Thus, we hypothesize that deep learning can be used to extract locomotion intent: where the animal wants to move within a two-dimensional space. The input to our algorithm is an local field potential measurements or binned spike data from 32 electrodes. We then use an LSTM model to output a predicted X and Y position. This works serves as a proof-of-concept that a LSTM based deep learning approach can decode hippocampal neural data in a continuous manner to predict locomotive intent of rats.

2 Related work

Extensive work has been doing in the past decade to improve neural activity decoding algorithms. Prior to the deep learning era, several regression based algorithms were implemented to decode neural signals. The current non-deep learning state of the art is a velocity Kalman filter (VKF) that assumes a linear relationship between the kinematic state of the subject (i.e. rat) and the neural data, with some added Gaussian noise. A VKF attempts attempts to estimate the Gaussian noise of the data and then attempts to predict velocity. A VKF only relies upon the last step of the data rather than the entire history, limiting its applications to neuroscience. One example of a VKF implementation is described in Wu et. al. [1], where a VKF was used to predict cursor motion trying to hit randomly paced targets with marginal success. Another example includes predicting retinal movement from visual cortex recordings [2]. The main advantage of VKFs are computational speed: only a limited set of weights are needed to make predictions. However, VKFs are unable to recapitulate more complex neural data considering the degree of noise and relationships between brain regions across time and space.

Deep learning, particularly recurrent neural networks (RNNs), is well suited for the task of decoding neural data due to its ability to capture long term temporal relationships and complex relationships between different brain regions. Echo state networks (ESN) is a particular RNN that has been used previously to predict neural behavior. An ESN relies upon sparsely connected and fixed randomly initialized hidden layers where input data can only alter the final output layer, making training extremely quick and able to capture noisy inputs. ESNs have been used to predict neuronal action potentials [3] and memory capacity [4] with limited success.

The current state of the art is an ESN RNN developed by Sussillo et. al. named FORCE [5]. FORCE stands for first order reduced and controlled error and applies an additional constraint that limits the amount of modification that each output neuron can change over the course of training. FORCE was used to decode cursor dynamics in a naturalistic hand movement task from hippocampal recordings with great success, which is the closest analogous task to locomotive prediction that we could find. We hoped to try a different deep learning framework in this paper - LSTM - to further validate a deep learning appraoch.

3 Dataset and Features

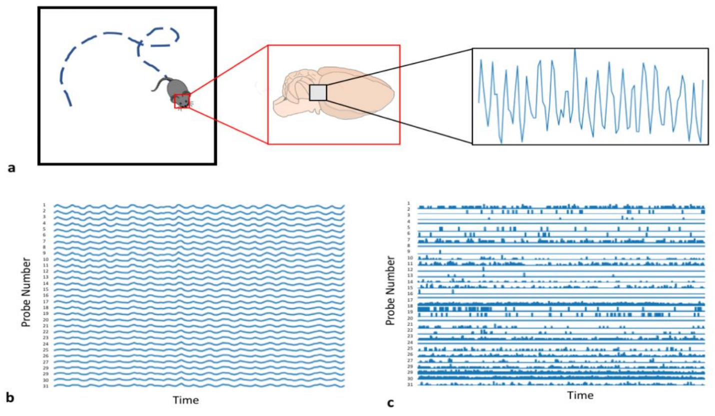

Figure 1: **Data acquisition pipeline from open field foraging task.** a. A Long-Evans rat explored a open field seeking water rewards while brain activity was electrical activity in the brain was monitored via shank probes. b. LFP from 32 probes plotted over 3 seconds. Characteristic 8 Hz theta waves were noted. c. Binned spike counts from probe locations over entire recording session.

The data for this project came from the Collaborative Research in Computational Neuroscience (CRCNS) database [6], where researchers will upload data from experiments for other researchers to use and analyze. A Long-Evans rat implanted with a 4 shank probes in the right dorsal hippocampus that recorded neural activity during an open field foraging task . Each shank probe recorded activity at 8 locations with one broken probe, resulting in 31 electrical recordings of hippocampal neurons lasting 17.7 minutes.

The dataset contains data regarding spike times, identity of the cluster producing the spikes, location data, and local field potentials (LFPs) for each electrode. Spikes are rapid inflections in the electrical activity of the brain region recorded, indicating a neuronal action potential in cluster the electrode was recording. The recording is then low pass filtered to obtain an LFP. LFPs are the summed electrical of local neurons, capturing the membrane polarization dynamics of a small brain region without noise from spike data.

The sampling rate of the spike data (20kHz), LFP data (1250 Hz), and location data (39.06Hz) were vastly different. In order to account for this and to make the spike matrix less sparse, we downsampled both the spike and LFP data to match the output frequency of the location data. Each point in the spike matrix represents the number of spikes the probe detected in a 25.5 ms time period. LFP data will be averaged over the same time frame. This is biologically relevant because a paradigm of neuroscience is that cumulative spike rate and average neuron behavior not isolated neuron firing dictates behavior. The end result was 40,554 time bins with average LFP and spike count for each probe coupled with X and Y position data.

Data was compiled and cleaned using Python and MATLAB scripts. A significant amount of the spike data was classified as “unidentified” cluster and was excluded from the input data. In addition, the location data had some points where the exact location of the mouse could not be determined from a video file; in these cases, linear interpolation was used to map between two known points.

An 85-15 train-test split was performed on the data, where the first 85 percent of the recording was used to train the model and then the last 15 percent was used to test. The number of time point samples for the train and test set are shown below:

Training Timepoints	Testing Timepoints
34471	6083

Table 1: **Summary of Train and Test Data**

4 Methods

In order to infer spatial location through neural data over time, we have adapted an Long Short Term Memory (LSTM) implementation from Jakob Aungiers [7]. This model was implemented in Keras with a TensorFlow backend [8,9]. LSTM models are able to capture long term dependencies across data, which could be helpful in predicting locomotive intent. LSTM’s capture long term dependencies by making use of three different gates: forget, input, and output. The forget gate uses the current timestep input and the previous cell state to determine which information should be carried over from the previous cell state at the prior timestep. This is the "memory" component of the LSTM that allows for long term storage of important predictive information. After this, information is modified by the input gate. The input gate makes use of input at the current timestep to modify the current cell state (which is really dependent on all of the previous cell states in the sequence). Finally, the output gate makes use of the current timestep to alter the hidden state of the LSTM cell. While the cell state acts as the "state memory" of the LSTM, the hidden state is used to access relevant information from previous timestep input values.

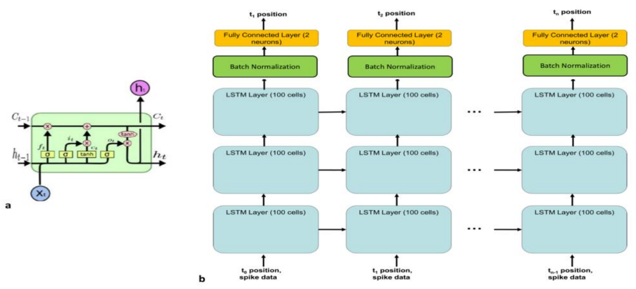

Figure 2: **Overview of LSTM Model.** a. A schematic version of a single LSTM node. b. A three layered LSTM model was constructed to process neural data with X and Y position predictions at each time point.

In our first model, LFP and binned spike inputs at each time step are sent to the first LSTM layer with 100 neurons. This then passes through a dropout layer with dropout rate = .2. Then we pass through two more LSTM layers of 100 neurons each before another dropout later with rate = .2. A final pass through Batch Normalization layer is performed before going through 2 output fully connected layer. A linear activation function is used for the output for the X and Y values. In following models, we varied the number of layers in the model, the sequence length that was fed into each neuron, and the type of input data (LFP or Spike Sorted).

The Adam optimizer was used for training. The loss function that was used was mean squared error (MSE) as is common in the BMI field. This loss function is defined as:

$$\mathcal{L} = \frac{\sum_{n=1}^{m}(y^i - \hat{y}^i)}{m}$$

Other possible models that could have been used included a bidirectional LSTM or a more complex deep LSTM model (i.e. adding more layers). A bidirectional LSTM does not fit the use case trying to be accomplished here; only previous data would be available to an algorithm trying to make predictions for the future. In addition, a more complex LSTM model would have resulted in slower predictions given that the potential use case here need to make rapid predictions using limited mobile computational resources. We therefore sought to find a model that represented a good balance of complexity and low error considering the aforementioned use case.

5 Results

Model	Data Input	Layers	Sequence Length	Mean Squared Error	Avg Point Distance
1	Spike	3	100	78.852	0.288
2	Spike	2	100	20.803	0.465
3	Spike	1	100	101.076	0.734
4	Spike	3	20	40.577	0.243
5	Spike	3	5	49.257	0.4688
6	LFP	3	100	173.474	3.03

Table 2: **Summary of model architecture and results.** "Spike" means that that input data was the binned spike sorted nuron firing and "LFP" means that the input data was local field potentials. Sequence length represents the number of previous points fed into each node.

Hyperparameter choices for this model was based on a previous model constructed to predict stock market prices (training epochs = 2, batch size = 32, LSTM Layer Neurons = 100) [7]. The parameters for the Adam optimizer were set as the defaults (learning rate = 0.001, $\beta_1 = 0.9$, $\beta_2 = 0.999$). As previously described, the number of LSTM layers (but not the number of neurons in each layer) and the sequence length fed into the model were varied across six models.

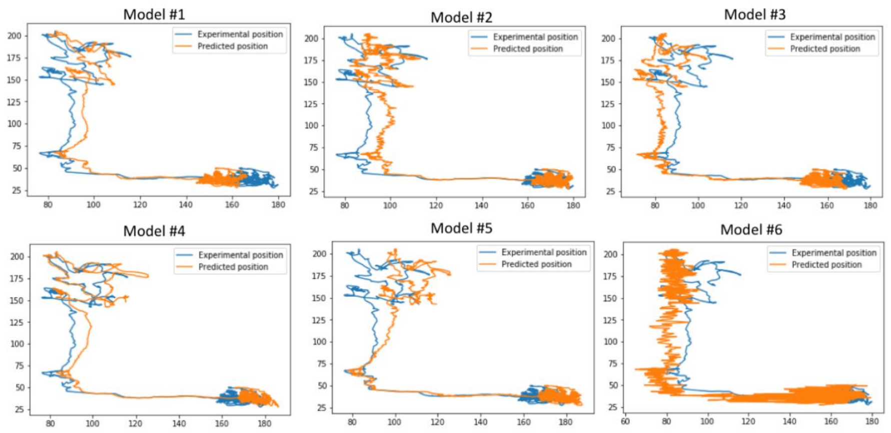

Figure 3: **Results from LSTM Models.** Six models were trained and the resulting predicting positions were overlaid with the ground truth experimental results.

We used MSE as our primary metric as well as the average distance between points to measure the "coarseness" of the resulting path that was predicted. We noted that there were local optimum for both the number of layers (2 in model 2) as well as the sequence input length (20 in model 4). Spike data has significantly less error than LFP data. This makes sense because LFP data was fairly constant throughout the entire timecourse (with the main features 8Hz theta waves), lowering its predictive value.

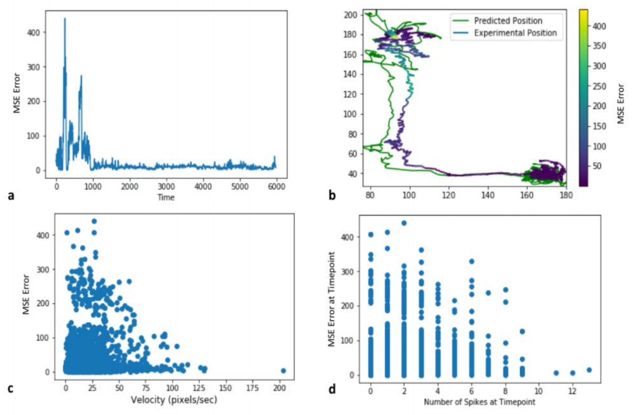

Figure 4: **Error Analysis of LSTM Model 2.**

The model with the lowest error (Model 2, 2 layers, 100 sequence length, spike data) was analyzed to determine where error was occurring. The MSE magnitude varied significantly over the course of the testing data (Figure 4a). To better localize where error was occuring, a colormap was applied to the predicted result positions, with yellow representing regions of maximum MSE (Figure 4b). Interestingly, straighter predicted sequences had higher error. In addition, there was a slight negative corelation between velocity and error ($R^2 = 0.4007$) (Figure 4c), meaning that faster velocity resulted in more accurate predictions. Fewer spikes correlated with increased error as expected ($R^2 = 0.14017$) (Figure 4d)..

6 Conclusion/Future Work

In our various LSTM implementations, we found that model 2 and model 4 were our most accurate methods, with MSE values of 20.8 and 40.6. Although the loss for model 2 is significantly lower than model 4, it is also worth noting that model 4 produces a much smoother output. Although model 2 was more accurate after two epochs, we believe that a sequence input length of 100 may have been too large. A sequence length of 20 (which equates to about .5 seconds of neural data) was more than adequate to predict the path with reasonable success. We hypothesize that the jagged solution of model 2 would have eventually smoothed out like in model 4, but this would have required many more epochs due to the increased input sequence length. In future iterations, we plan to create a loss function that also punishes jagged solutions. This will result in accurate but smooth predicted paths, which will certainly be important in navigating spaces effectively.

In the future, we also plan to assess our ability to predict velocity rather than position. This may require a more complex network, since velocity is abstracted away from position. However, we believe predicting velocity is important because it may enable us to generalize our LSTM to different spatial contexts. This current implementation must be trained individually for each context it is to be used in. By learning to predict velocity, we hope to find a generalized solution that allows us to extract navigation intent in a variety of contexts.

7 Contributions

Both MM and NO worked on data formatting and cleaning. MM primarily modified the LSTM model from GitHub with input from NO. NO worked on data visualization. Both worked on this report and the poster.

References

[1] Wilson, R and Leif Finkel. A Neural Implementation of the Kalman Filter. NIPS. (2009)

[2] Orban de Xivry JJ, Coppe S, Blohm G, Lefèvre P. Kalman filtering naturally accounts for visually guided and predictive smooth pursuit dynamics. J Neurosci. 2013;33(44):17301-13.

[3] Gürel T, Rotter S, Egert U. Functional identification of biological neural networks using reservoir adaptation for point processes. J Comput Neurosci. 2010;29(1-2):279-299.

[4] Rodriguez N, Izquierdo E, Ahn YY. Optimal modularity and memory capacity of neural reservoirs. Netw Neurosci. 2019;3(2):551-566.

[5] Sussillo D, Nuyujukian P, Fan JM, et al. A recurrent neural network for closed-loop intracortical brain-machine interface decoders. J Neural Eng. 2012;9(2):026027.

[6] Mizuseki K, Sirota A, Pastalkova E, Buzsáki G. (2009): Multi-unit recordings from the rat hippocampus made during open field foraging. http://dx.doi.org/10.6080/K0Z60KZ9

[7] Aungiers, Jakob. LSTM Neural Network for Time Series Prediction. (2018). GitHub repository.https://github.com/jaungiers/LSTM-Neural-Network-for-Time-Series-Prediction.

[8] Keras. Chollet, F. (2015).

[9] Martín Abadi and Paul Barham and Jianmin Chen and Zhifeng Chen and Andy Davis and Jeffrey Dean and Matthieu Devin and Sanjay Ghemawat and Geoffrey Irving and Michael Isard and Manjunath Kudlur and Josh Levenberg and Rajat Monga and Sherry Moore and Derek G. Murray and Benoit Steiner and Paul Tucker and Vijay Vasudevan and Pete Warden and Martin Wicke and Yuan Yu and Xiaoqiang Zheng. TensorFlow: A System for Large-Scale Machine Learning. 12th USENIX Symposium on Operating Systems Design and Implementation. (2016).