CS230

Predicting Monkey Arm Movements via Electrode Signals

Ziyad Gawish
Stanford University
zgawish@stanford.edu

Mariane Mousa
Stanford University
marianem@stanford.edu

Abstract

Prosthetic limbs have allowed some patients to recover the majority of the function of the original limb. Medical providers utilize these artificial limbs for patients who either lost a limb in an accident or were born without a limb. However, this use case does not completely extend to those who are paralyzed, as artificial limbs cannot currently move on their own through brain signals. In this project, we explore this possibility by utilizing neural network models to predict the direction of a monkey's arm movement using electrode time series data. Our long short-term memory (LSTM) model scores a maximum accuracy of 20.13% with only 506 data points, leading to the conclusion that more data is needed to properly train a LSTM model with such requirements and thus that this domain is especially difficult to apply deep learning models. However, since the accuracy is above a guessing accuracy of 12.5%, we believe that it is possible to use our model to accurately predict such movements. Our baseline model, which we improved after gaining insight and intuition from our LSTM model, had a final accuracy of 42.2%. We believe that this shows that our smaller dataset works better on simpler models, but the overall problem would be better suited for an LSTM given we had a larger dataset.

1 Introduction

The brain utilizes action potentials, or the fast movement of electrical current into a cell, to send signals across neurons. Measured through electrodes on a monkey, we are utilizing the rate and pattern of action potentials to predict the direction of arm movement. To obtain the data in each trial, the monkey is cued to move in a specific direction, waits 500 milliseconds, then actually moves in the specified direction. After moving, the monkey returns to its central "resting" position, then repeats as it is cued. We are using the electrode data from their motor cortex during the "planning" period, which is the period after the monkey has been cued but does not start moving yet, as an input to our algorithm. We then use an LSTM to output one of 8 predicted directions that a monkey will reach towards. These directions include the cardinal directions and the intercardinal directions.

Our project has many applications, making it an interesting problem. As discussed in the abstract, applying this to technology to improve the lives of paralyzed patients. If scientists are able to predict how a human wants to move, this can be transmitted to a prosthetic device attached to the arm, allowing someone to move their arm once again.

CS230: Deep Learning, Winter 2018, Stanford University, CA. (LateX template borrowed from NIPS 2017.)

2 Related work need to do

We based our project on that worked on utilizing delayed activity in the premotor cortex to predict the reach direction of monkey's arm. However, none of these papers used a neural network approach to these predictions. One uses an SVM to predict the movement of the monkey's arm [1], while the other uses a dedicated BCI (Brain Computer Interface) in addition to varying integration periods to make these predictions. One paper's approach was clever in the sense that it used simple trends found in the data such as firing rates, the number of neural spikes in a binned period, and the use of an Support Vector Machine (SVM) to accurately predict the monkey's reach directions [1]. This approach is less complicated than our base NN approach and far less complicated to our LSTM approach. While the experiment shown in the first paper shows that there are obvious trends found in the signals to extract useful information, we thought that using a more complicated model can find more subtle details in the data. Given the relatively low complexity of the task and the sparse amount of data present, the SVM approach proved to be more accurate than our LSTM approach with an accuracy of 32.2% versus our 20.03%. Our base neural net on the other hand, did better than the SVM with an accuracy of 42.2%. We used another paper to help fine tune which part of the neural data we should consider for training and testing [6]. Initially, we were making little progress training the entire sequence of neural spikes but after some thoughtful considering and help from the paper, we were able to gain improvements in accuracy by limiting our scope of the data to just 300ms starting at the 200ms mark. The paper shows us that the delay between the monkey being shown where to reach and the actual movement is important in predicting the reach direction because it is clear from any motor cortex noise that can be attributed by the actual motion of the arm. Another two papers helped us gain an intuition of how to design the architecture of our model. We noticed that we need very few layers as shown in the experiments in the papers [7][8], with one layer being an LSTM layer and another being a fully connected layer. While they both examine different topics in brain activity, we saw that the paper predicting student confusion, had an accuracy of around 70% which showed us that an LSTM approach may be viable[8].

3 Dataset and Features

Our dataset contains 506 trials and is time series data, which we separated into 70% train set and 30% test set. We ensured that there were the same proportion of each class (reach direction) in each set. We did not use a validation set due to our limited dataset size. There are 3 components per trial: an array of 1's and 0's documenting the presence/absence of an action potential at a given time for a specific electrode, where each index represents its respective millisecond, the target reach direction, which is an (x, y) coordinate mapped to one of 8 directions, and the live position documented over time of the monkey's hand during the trial.[1] We are utilizing only the first two components to make our predictions.

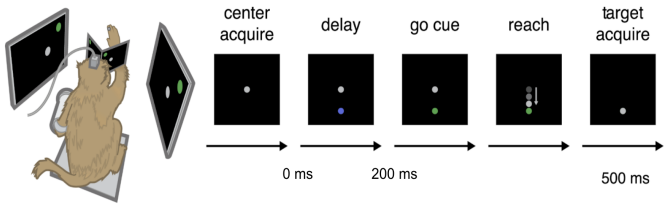

Figure 1. Data collection [1]

The first component of each trial, which is an array documenting the timings of the neuron spikes, is of size 96x300 as we are documenting the action potentials from 96 neurons over the course of 300 milliseconds. An example for one neuron is shown in figure 2 below, where it visualizes the action potentials over the 506 trials. There are 96 equivalent electrodes measuring action potentials in its respective area. This example shows electrode 17. During data collection, a monkey starts with their arm positioned at the center position. After this, there is a 200 millisecond delay where nothing occurs, and following this, the monkey is cued to move in an indicated direction. The monkey then recalibrates to the center position to prepare for the next trial.

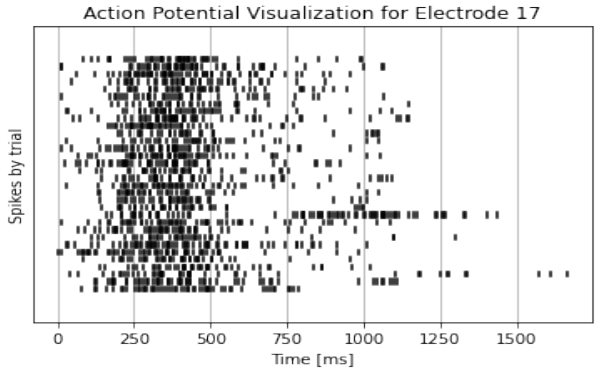

Figure 2. Visualization of array showing action potential times. Each darker spot indicates that there was an action potential at its respective time and trial at electrode 17. [1]

We do not use the data from the first 200 milliseconds to allow for the monkey to restabilize from the previous trial and to account for reaction time. This is called the "T-skip" portion. The second component for a trial is the reach direction, which is correctly labeled for each trial as a (x, y) coordinate. These (x, y) coordinates are organized into 8 classes, which are the cardinal and intercardinal directionst. The center point is used to recalibrate the monkey and also as a target direction. The true label is documented as a one-hot vector, where each index represents a class.

To satisfy the novelty portion of the project, we conducted pre-training on the first component of each trial and performed data augmentation. We implemented time series smoothing, which is intended to help models identify patterns and trends by suppressing short-term variability and capturing long-term trends. Short term variability usually stems from noise and random variations, and smoothing helps the model avoid these parts of the data. To implement time series smoothing in our model, we used the tsmoothie python package [2]. Additionally, we augmented our data. Originally, we decided to remove the recalibration data points where the monkey returns to the center position as the direction of travel would be different based on the previous location. These points made up half of the dataset, making models pick the center position almost every trial. We needed to counteract this behavior without causing our daatset size to decrease by 50%. In order to add back these data points, we changed the true label from "center" to the direction the monkey would be traveling from its previous position to the center position. For example, if the monkey's arm was in the "North" position after a trial, then the next datapoint where the monkey returns to the center position would be labeled "South." This data augmentation allowed us to utilize all of the datapoints.

4 Methods

As a base model, we used a standard feed forward neural network. We started with this model as a building block to our final LSTM model. We accomplished a test accuracy of 30% and a training accuracy of 95% on this model. Our baseline model outputted a one hot vector, showing which of the 8 possible the directions of monkey movement. This basic neural net had 3 layers, which had the following number of cells per layer: W1 : [40, 28800], b1 : [40, 1], W2 : [18, 40], b2 : [18, 1], W3 : [9, 18], b3 : [9, 1]. We hoped that using a more complicated model would help improve our accuracy on this model, as we required the neural network to capture the pattern of all action potentials over time rather than each individual point.

After working on our standard feed forward neural network, we decided on using an LSTM to take advantage of the feedback connections. Additionally, LSTMs avoid long-term dependency issues that other recurrent neural networks face, where a vanishing gradient causes long term data to not be utilized throughout the network. Since our prediction criteria depends on the pattern of action potentials over 200 milliseconds, we decided to use an LSTM to ensure that the data was well captured. Lastly, LSTMs are used to efficiently model sequential data, learning high level representations that capture the structure of the data. This allows our model to identify the pattern of electrode data for each possible reach direction.

To measure loss in both our baseline and LSTM models, we utilized the categorical cross entropy formula. We chose this formula because it is conventionally used when true labels are one-hot encoded and when optimizing classification models. Both of these criteria are components in our model. The formula for cross entropy is the following: $Loss = -\sum_{i=1}^{i=N} y_i log(\hat{y}_i)$.

In this formula, y_i represents true labels and $\hat{y}_i$ represents labels predicted by our model. Both inputs are arrays of size 8x1, where each entry is the probability of that class label being correct. In y_i, there is a single 1 entry and the rest are 0's. In $\hat{y}_i$, it is more likely for the array to have multiple non-zero entries, but all of the entries must sum to 1. This formula increases as the numerical differences at each index increases between $\hat{y}_i$ and y_i.

5 Experiments/Results/Discussion

With our LSTM model, we had a test accuracy of about 20.03%, a train accuracy of 88.8%, and a top-3 categorical accuracy of about 52.59%. The latter score indicates that the correct answer was in the top 3 candidates over half of the time. This shows that the model does have the potential to learn based on our data, but it needs more data to properly train the LSTM. Additionally, our model has a precision of 0.20, and and F1 score of 0.24. These values indicate that our model has a high number of false positives, which is confirmed by the F1 score and accuracy. This figure also shows how the precision levels out as the recall value increases. The confusion matrix on the left side of figure 4 shows that there are a few directions that the model is more accurate in identifying, including East, North, and West. This could be due to many factors, including that these directions are all cardinal rather than intercardinal. The cardinal directions could be easier to connect to electrode patterns as those movements are simpler and require less actions potentials [1]. Our test accuracy fluctuated with each run, ranging from 15-20%, where 20.03% was the highest we have seen.

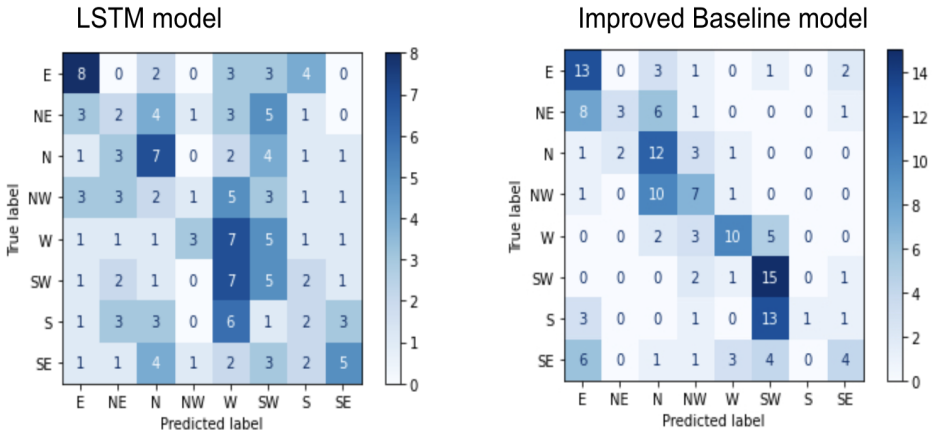

Figure 5. Precision vs. Revall graphs for LSTM model (left) and improved baseline model (right) [5].

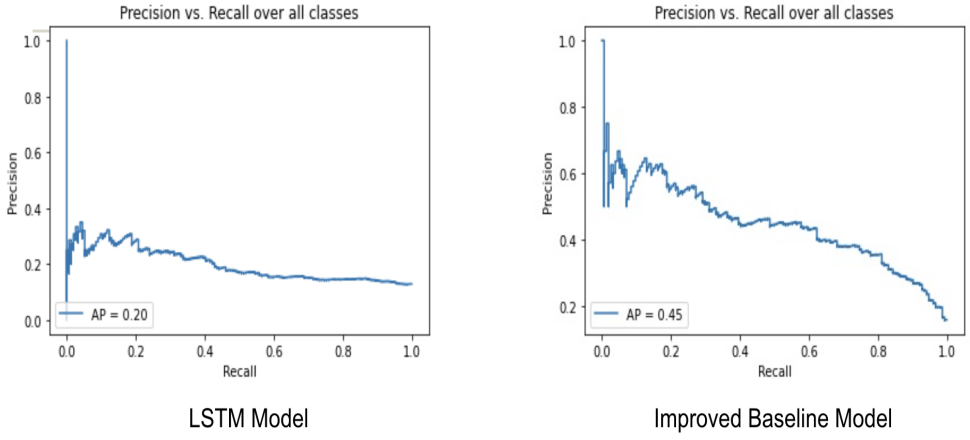

Figure 4. Confusion matrices for our LSTM model (left) and our improved baseline model(right) [5].

We fine tuned several hyperparameters. These include batch size, number of layers, number of cells per layer, number of epochs, optimizer and dropout. After experimenting with dozens of hyperparamters using nested for loops, we utilized a batch size of 32, 70 epochs, 3 layers, 64 cells per layer, the Adam optimizer, and a dropout rate of 0.7. Larger or smaller batch sizes made it more difficult to optimize our accuracy in conjunction with the number of epochs. We experimented with several optimizers, including adamax, root mean square propagation (RMSprop), gradient descent with momentum, and Adamax. We utilized the adam optimizer as it performed the best during experimentation. We believe this is the case because the adam optimizer combines the benefits of RMSprop and gradient descent with momentum. After noticing overfitting to our training model, we decided to include dropout at a higher rate of 0.7. This higher rate allowed us to counteract the over-fitting to the training model. The exact value was determined through experimentation. A combination of these hyperparameters allowed us to prevent overfitting. Before these changes, our training accuracy was 95% and our test was 15.5%. After, our training accuracy became about 89%, decreasing the difference between the test and train accuracies and thus counteracting overfitting to the training set.

After fine tuning the hyperparameters, adding dropout and implementing pretraining on our LSTM model, we decided to apply similar additions to our baseline model. We found that our baseline model was more accurate than our LSTM model, resulting in a test accuracy of 42.2%, a train accuracy of 99.7%, and a top-3 accuracy of 85.7%. The confusion matrix for our improved baseline model is shown on the right side of figure 4, showing a clear improvement from the LSTM model. The precision vs. recall graph shown in figure 5 for the improed baseline model (right) is also better than that for the LSTM model (left) as shown by the higher area under the curve. Lastly, the improved baseline performed better numerically on f1 score (0.356), overall precision (0.322), and overall recall (0.397) than on the LSTM model. We conclude that our smaller data set worked better with the simpler model, but if we had a much larger data set, the LSTM model would ultimately perform better.

6 Conclusion/Future Work

Using an LSTM, we were able to utilize patterns in neurological signals (actions potentials) to start to predict the direction of movement by a monkey's arm. Our LSTM model was theoretically better than the base model due to the nature of our dataset, but the small size of our dataset ultimately allowed the simpler model to perform with a higher accuracy. However, the LSTM allowed us to capture the pattern of action potentials rather than each individual one, which led us to the conclusion that the LSTM model would perform better than the base if we had a larger dataset.

While we were disappointed with our final accuracy of both our LSTM and baseline models, we believe that our results are still relevant and highlight the necessity for more resources to be allocated to healthcare related deep learning projects. In our case, having several more monkeys completing the same exercise could have allowed us to collect much more data and thus a much more accurate model, so we would use resources to expand our dataset. From our limited dataset using the LSTM, we accomplished an accuracy 20.13%, which is above a guessing accuracy of 12.5%. This shows that an LSTM has the capacity to utilize electrode data to make predictions, but it likely needed more data to properly train the model. Our top-3 accuracy of over 50% also supports that this model However, increasing our dataset size to the sizes of other datasets used for deep learning would be extremely expensive and time consuming as is the nature of health related data. It cannot be automated and requires real animal subjects. Even though these types of deep learning projects require high levels of resources, it is likely that the results could be very fruitful for the use of biomechanics for patients needing artificial limbs.

7 Contributions

Mariane and Ziyad contributed equally to the project. We conducted similar amounts of research and literature review, brainstormed ideas together, implemented the code together on the same screen, and wrote this report and our slides in parallel.

References

[1] Even-Chen N, Sheffer B, Vyas S, Ryu SI, Shenoy KV (2019) Structure and variability of delay activity in premotor cortex. PLoS Computational Biology. 15(2): e1006808.
[2] Marco Cerliani, Jitendra Mishra (2020) Recoil [Source Code]. https://github.com/cerlymarco/tsmoothie
[3] Harris, C.R., Millman, K.J., van der Walt, S.J. et al. Array programming with NumPy. Nature 585, 357–362 (2020). DOI: 10.1038/s41586-020-2649-2.
[4] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Rafal Jozefowicz, Yangqing Jia, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dan Mané, Mike Schuster, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org.
[5] Pedregosa et al., Scikit-learn: Machine Learning in Python, JMLR 12, pp. 2825-2830, 2011.
[6] Santhanam, Gopal, et al. "A High-Performance Brain–Computer Interface." Nature, vol. 442, no. 7099, July 2006, pp. 195–98. www.nature.com, https://doi.org/10.1038/nature04968.
[7]Bano, Lorela. "LSTM-Based Model For Human Brain Decisions Using EEG Signals Analysis." Electronic Theses and Dissertations, Jan. 2021, https://digitalcommons.georgiasouthern.edu/etd/2291.
[8] Ni, Zhaoheng, et al. "Confused or Not Confused? Disentangling Brain Activity from EEG Data Using Bidirectional LSTM Recurrent Neural Networks." Proceedings of the 8th ACM International Conference on Bioinformatics, Computational Biology,and Health Informatics, Association for Computing Machinery, 2017, pp. 241–46. ACM Digital Library, https://doi.org/10.1145/3107411.3107513.