Project Mouse Final Report

Zening Ye, Qiannan Shen

Department of Mathematics and Statistics, Boston University

MA 679 A1: Applied Statistical Machine Learning

May 03, 2022

Brain Behavior Prediction with Mouse Data

Abstract

This report is based on the mouse's neuron activity and its behavior in an elevated zero maze. The goal of this paper is to predict whether the mouse would be in the open arm (anxiogenic) or the closed arm (anxiolytic) in the elevated zero maze by using the z-score of cells for the mouse chosen randomly. After analyzing the characteristics of the data, our group first use logistic regression as a baseline model for predicting the behavior one hour into the future. We create a generator for training our model. Then, the report uses two types of Recurrent Neural Network (RNN) for more accurate prediction — a single-step model by time unit and a multi-step model by time period. In Particular, our simple model has five layers and uses 1.5 seconds of the period to predict the next time point. Eventually, compared to the baseline model, the single-step RNN model gets 20% more improvement.

Introduction

A hallmark of the anterior cingulate cortex (ACC) is its functional heterogeneity, and it has been implicated in several complex cognitive functions, such as empathy, impulse control, emotion, and decision-making. Some studies emphasize its importance in the encoding of anxiety-related and social stimuli. After knowing this, we are interested in how the ACC, which is related to neuron activity, affects behavior. In this report, we focus on a related experiment about the mouse's behavior in an elevated zero maze. The elevated zero maze is a circular channel with four parts — two parts are closed arm with a wall and two parts are open arm without a wall. The experiment with multiple mice records the neuron activity of different cells in each mouse over time and whether each mouse is in the closed arm (anxiolytic) or in the open arm (anxiogenic) by time. This experiment is related to social behavior. When the mouse is in the open arm, which means the mouse is possibly being eaten by a bird in the light space, the anxiety level of the mouse increases. Hence, this report would predict the mouse's behavior in the elevated zero maze by using the data about the neuron activity and analyzing the effect.

Data Processing

The data of the elevated zero maze experiment records 13 mice's neuron activity and their behavior over time. Under our report, we randomly selected one mouse, 608102_414, as our priority object to design the analysis and prediction.

The single-cell recording for neuron activity has been recorded into two files, binned z-score and behavior. In the single-cell recording, each column represents a different cell and each row is a different temporal sample in the time series. In the behavior recording, column 1 indicates when the mouse is in the closed arm and column 2 indicates the mouse is in the open arm. We first use the readMat function to read the corresponding data and make it a data frame. Then, add an "Average" column that calculates the average z-score of cells. Though the z-score changes frequently in each cell, the overall z-score doesn't have much difference. Next, we combined the single-cell and behavior recording together by adding two new columns ‘closed’ and ‘open’ to indicate whether the mouse is in the open arm or the closed arm. There are only two conditions under this experiment, whether the mouse would be in the open arm or not. The data has two 0 or 1 because researchers did not observe mouse behavior during that time period. Therefore, we removed these data and continued our analysis.

Exploratory Data Analysis

Figure 1 indicates that the mouse 608102_414 would have 60.9% probability in the closed arm and 39.1% probability in the open arm. The mouse spent most of the time in the closed arm. It is reasonable that in the open arm without a wall the mouse is afraid of being eaten by a bird and so it prefers the closed arm with a wall. In order to further prove this hypothesis, we graph the proportion of the mouse whether in the open arm or in the closed arm for all the 13 mice (Figure 2). And the graph below shows that the mice did spend most of the time in the closed arm, though there is one mouse that would prefer to be in the open arm.

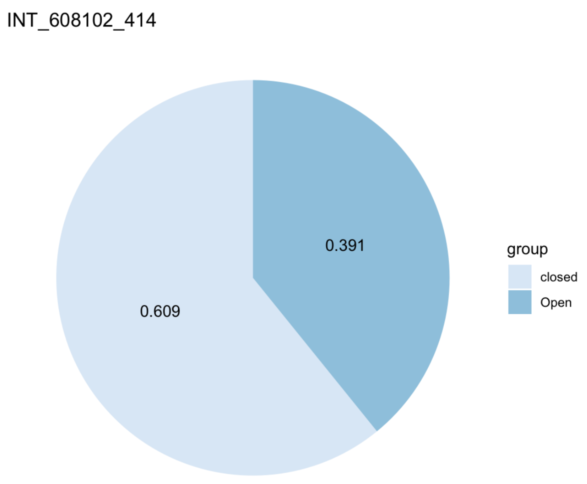
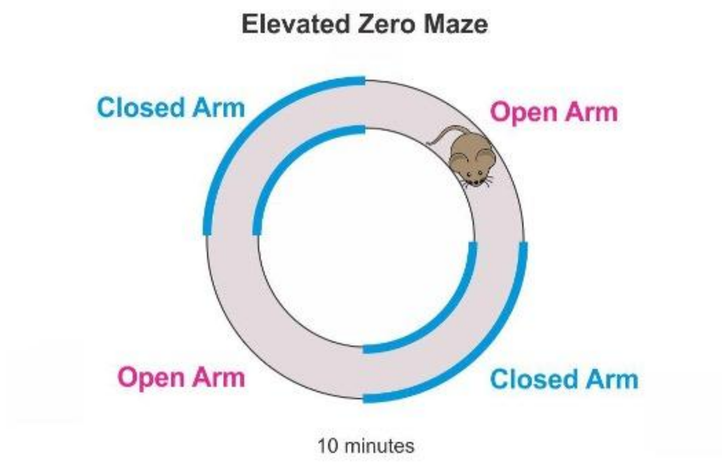


Figure 1: Percentage of two conditions



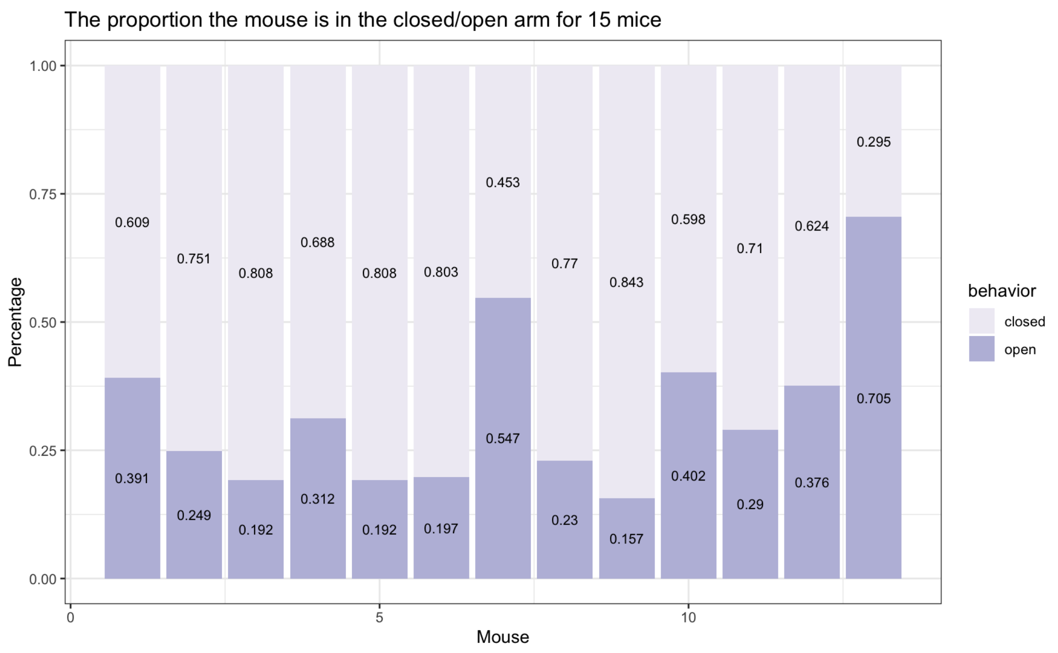


Figure 2: Proportion of mouse under two conditions

These two graphs below indicate the change in z-score, and the color shows where the mouse is in the elevated zero maze. The black line shows the average z-score of cells in the mouse 608034-409. The z-score, which is related to the neuron activity, fluctuated rapidly with no obvious pattern. We could see the neuron activity in the mouse's brain is complicated and unstable. The orange color and light purple color in the following graph represent the mouse in the closed arm (orange) and the mouse in the open arm (light purple). Combined with the behavior of the mouse and the neuron activity, there is no obvious pattern we could conclude. So, it is important to do further data analysis and build a proper model.

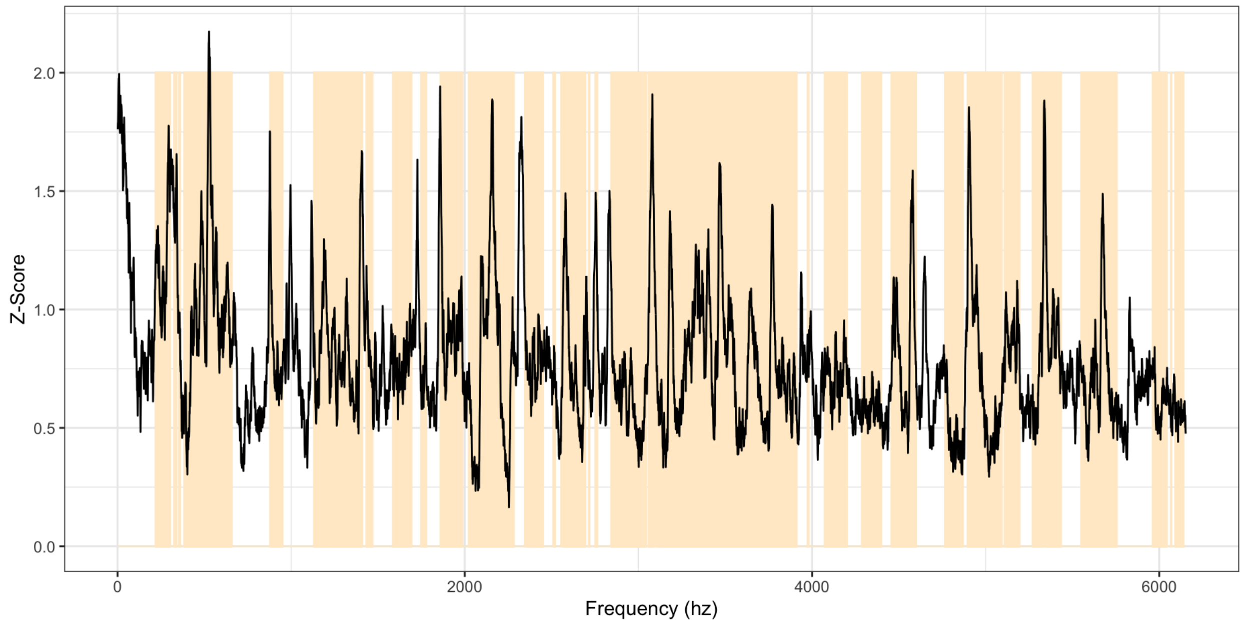


Figure 3: Z-Score vs. frequency(close arm)

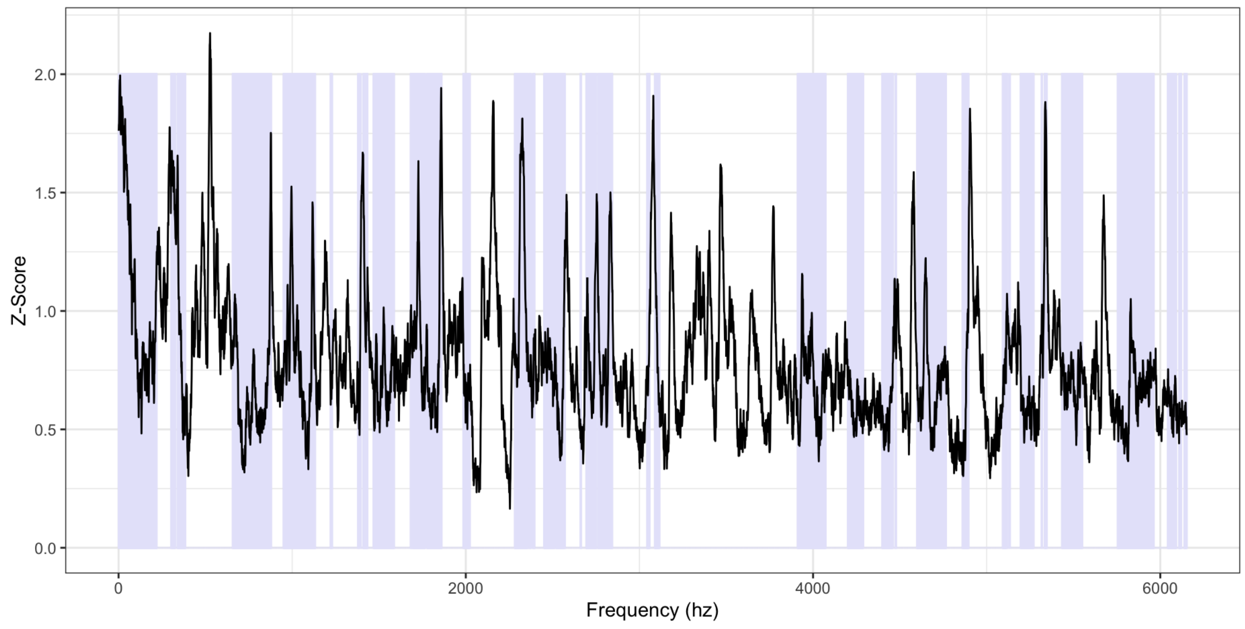


Figure 4: Z-Score vs. frequency(open arm)

Figure 5 illustrates the density of the average z-score of cells. The z-score at most of the time is around 0.2-0.75. The distribution of the z-score gradually decreases as the value goes up after 0.75.

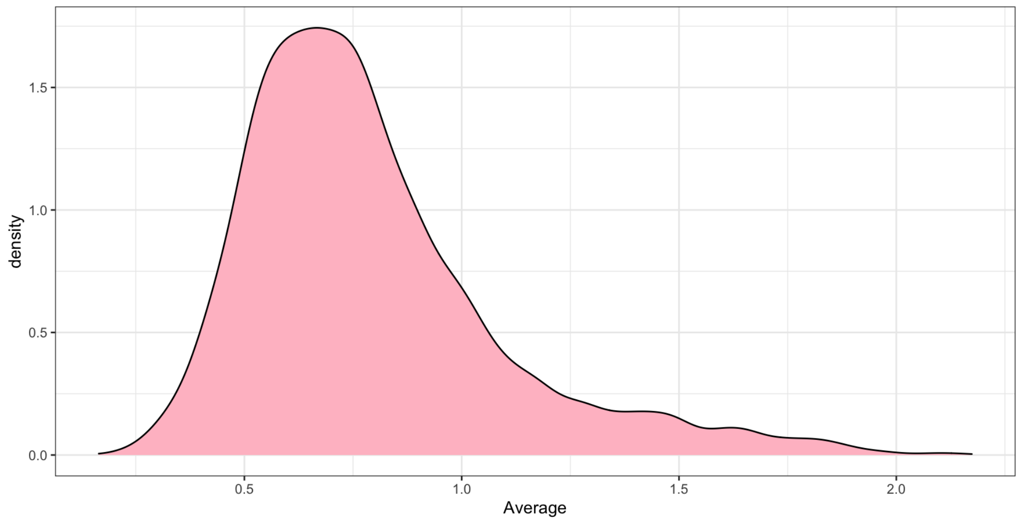
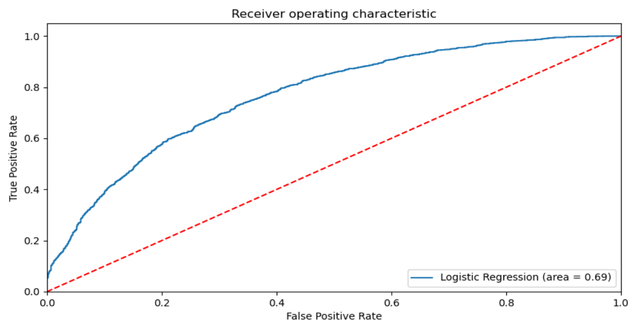
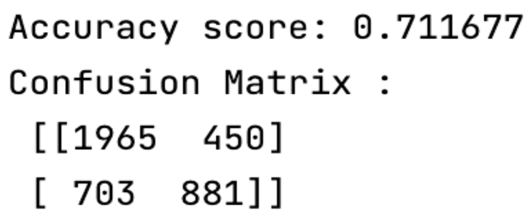


Figure 5: Density of Average Z-Score

Logistic Regression

Before building a trainable model, it would be plausible to have a baseline/preliminary as a point of comparison with the latter complicated model. There is a little different between the data we used for EDA and Logistic, RNN. We combined “open arm” and “close arm” into one column “behavior”. The baseline model is designed for predicting the behavior one hour into the future, given the current behavior of all behavior. In general, we use logistic regression as our baseline model. The prediction of the baseline model around 71.17%, which is a good starting point. Below is the accuracy and ROC plot we made through the baseline model.



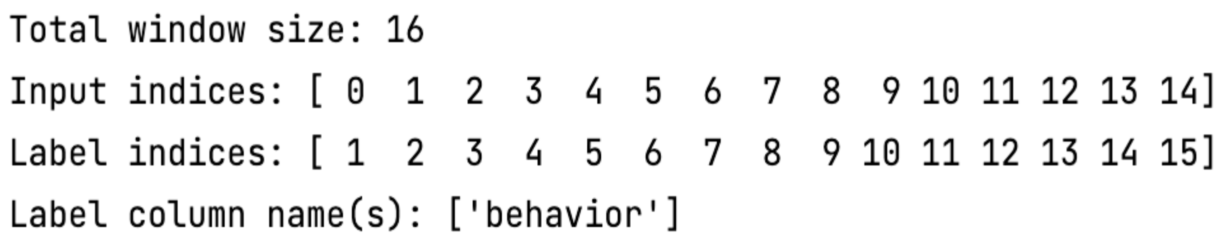


```
Accuracy score: 0.711677
Confusion Matrix :
 [[1965  450]
 [ 703  881]]
```

Figure 6: ROC Plot, Accuracy, and Confusion Matrix

Generator

Every model needs to have a suitable window slice for training. Therefore, our group created a window generator to generate the appropriate windows for the model we created further. There are some parameters we can set up with, input_width, label_witdth, step, and label_column. The brain neurons respond quickly, so we are unable to set the time interval of the baseline model to predict every second of behavior because there would be a time lag problem. Figure 7 is the framework of the windows:



```
Total window size: 16
Input indices: [ 0  1  2  3  4  5  6  7  8  9 10 11 12 13 14]
Label indices: [ 1  2  3  4  5  6  7  8  9 10 11 12 13 14 15]
Label column name(s): ['behavior']
```

Figure 7: Window Information(Single)

Recurrent Neural Network

Introduction

Recurrent Neural Network(RNN) is a type of artificial neural network which uses sequential data or time series data (IBM Cloud Education, 2020). This is the main reason that our group chose to use RNN to perform a binary classification with time series questions. There are several parts included in RNN model, windows, training model, accuracy, and some plots. The following image indicates how recurrent neural networks work.

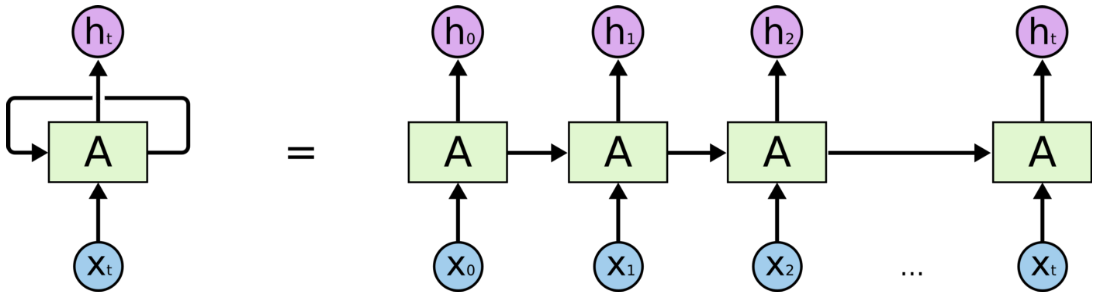


Figure 8: Recurrent Neural Network(Radhakrishnan, 2020)

Model

We originally conducted two types of RNN model, one is a single step model, and the other is a multi-step model. The single-step model uses a period of time to predict the behavior at the next point in time, input time is the input_width in the generator. On the other hand, a multi-step model predicts the behavior at the same time as input time period by time period. Generally speaking, the output of a single-step model only contains one unit, while a multi-step model has multiple outputs.

There are five different layers contained in a single model, two gate recurrent unit layers, one dropout layer and two fully connected layers. Figure 9 illustrates the order of the layers. To be more precise, each layer has a different unit in the model. The first GRU layer has 64 units/neural with ReLU activation function; the dropout layer with level 0.1 to prevent overfitting; the second GRU layer has 32 units with ReLU activation; next is the fully connected layer contains 64 units with ReLU activation; last but not least, the output layer, a fully connected layer, has one unit and Sigmoid activation function.

Tuning the hyperparameters is the most complicated part of training the whole model. To train an appropriate model, we tried different combinations of hyperparameters. Considering the brain's response is instantaneous, then we decided that input_width and label_width are 15 units and step is 1 unit. In other words, we used 1.5 second of the period to predict the next time point(0.1s). Due to the size of data, the epoch is 20, batch_siz is 11, and the learning rate is 0.0001. The reason we choose this combination is we want the model learning more carefully and as slow as possible. When the key information is extracted, the whole model can recognize the data based on more subtle features, and thus read out more accurate predictions. Since it is a binary classification problem, we used BinaryCrossentropy as the loss function and RMSprop(Root Mean Squared Propagation) to be the optimizer for the model.

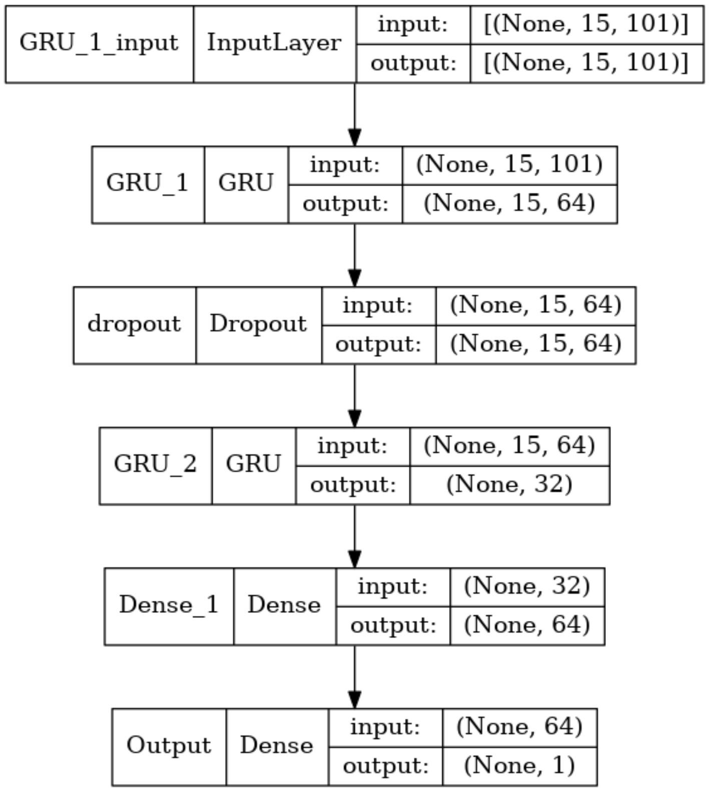


Figure 9: Single-step RNN Model

Figure 10 illustrates the loss value and accuracy between training set and validation set, both loss and accuracy are approaching each other continuously, which aliases the entire model training process. In the model, we should care more about the validation set's loss and accuracy, because we need to evaluate the accuracy of the test set by the validation set. Under the validation set, we got 0.22598 and 0.93695 for the loss and accuracy, when we evaluated the model by using the testing set, we got 0.21338 and 0.94398.

Comparing to the baseline model, single step RNN model has a better performance to predict the behavior of mice by using their neural signal. 20% more of improvement verified the advantages of RNN models in dealing with time series problems.

In order to predict the multi behavior for a sequence of time, we also develop a multi-step RNN model. The different between these two models is the “step”. For the single model, since we only need to predict a single time stamp, the step for the single model is 1. However, the purpose of multi step is to predict the behavior in a time period, the step would be the same as the input_width. Unfortunately, no matter how we adjust the hyperparameters, number of layers and loss function, the multi-step model did not perform a good forecasting. The entire accuracy between 0.60 to 0.62, and the model will overfit after 5 epochs. Figure 11 illustrate the architecture of multi-step RNN.

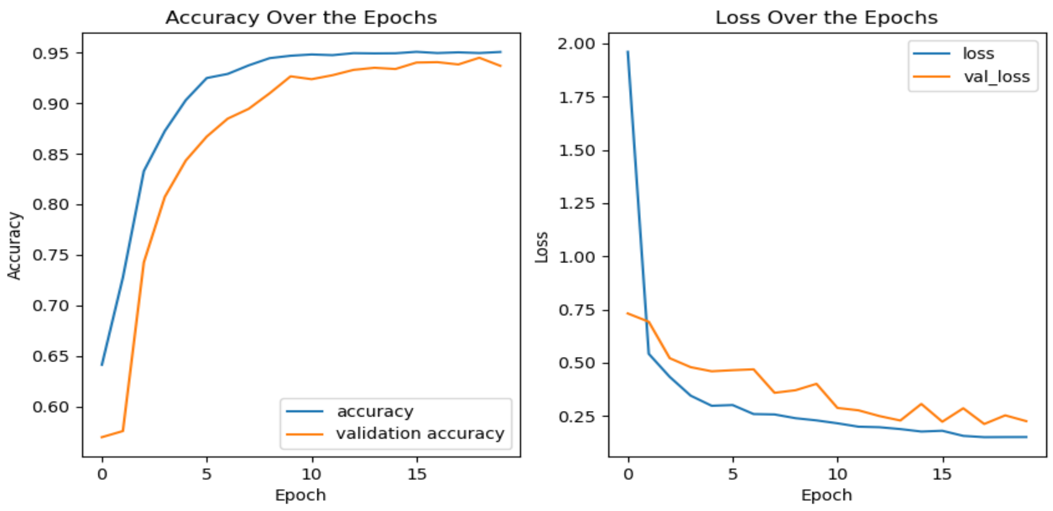


Figure 10: Accuracy and Loss Over Each Epoch(Single Step)

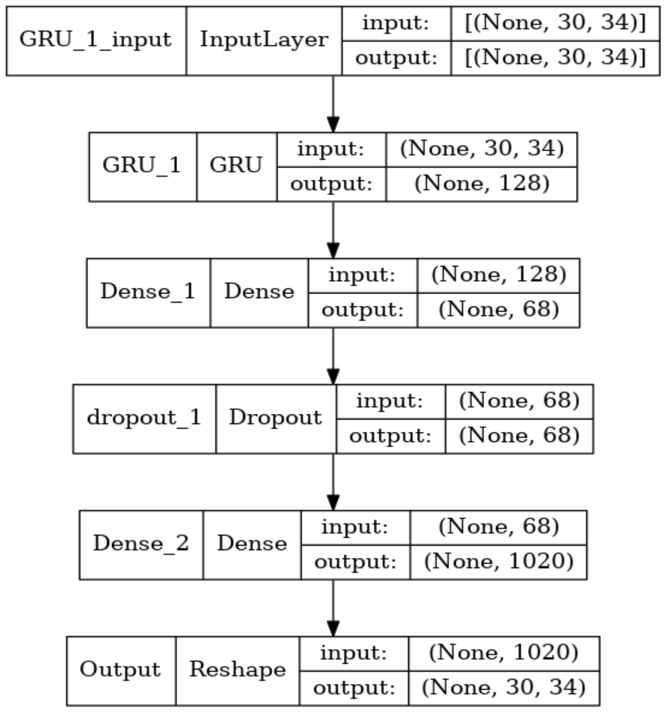

GRU_1_input | InputLayer | input: [(None, 30, 34)] | output: [(None, 30, 34)]
GRU_1 | GRU | input: (None, 30, 34) | output: (None, 128)
Dense_1 | Dense | input: (None, 128) | output: (None, 68)
dropout_1 | Dropout | input: (None, 68) | output: (None, 68)
Dense_2 | Dense | input: (None, 68) | output: (None, 1020)
Output | Reshape | input: (None, 1020) | output: (None, 30, 34)

Figure 11: Multi-Step RNN Model

Result

Both models only use one mice data to conduct the experiment. As you can see, the single-step RNN outperforms logistic regression and multi-step RNN in instantaneous prediction. We controlled the training process by setting different batch_size, epochs and learning rate to reduce the loss value. In general, recurrent neural networks outperform other models for time series prediction problems, which is a characteristic of RNNs.

Discussion

There are a lot of things that we did not achieve in this project. First, we only use one mouse to conduct the RNN model, which means we did not use other mouse data to test our model. We would like to build a generalized model to predict the behavior of all rats, which makes it necessary to consider carefully the selection of layers of the neural network and the setting of hyperparameters. Second, we did not conduct a good many-to-many model(multi-step RNN) for the advance topic. Third, the neural response of the brain is very fast, and we are not able to ensure that the values we use in creating windows represent the behavior of the whole brain. Furthermore, when cleaning the data, we removed all the data that were observed abnormal, close and open arm both equal 00 or 11, to create the dataset we used for neural network. What if we use these data as a validation set, to predict the mouse behavior during these times. It will be a challenging task to validate whether the prediction is correct or not. Last but not least, there is one architecture we did not use for multi-step model, which is Autoregression model. Autoregression is a time series model that uses observations from previous time steps as input to a regression equation to predict the value at the next time step (Brownlee, 2017). We believed Autoregression might be better for multi-step forecasting for the brain data.

References

Brownlee, J. (2017, January 2). *Autoregression Models for Time Series Forecasting With Python*. Machine Learning Mastery. Retrieved May 4, 2022, from https://machinelearningmastery.com/autoregression-models-time-series-forecasting-python/

IBM Cloud Education. (2020, September 14). *What are Recurrent Neural Networks?* IBM. Retrieved May 4, 2022, from https://www.ibm.com/cloud/learn/recurrent-neural-networks

Radhakrishnan, P. (2020, August 20). *Introduction to Recurrent Neural Network | by Pranoy Radhakrishnan*. Towards Data Science. Retrieved May 4, 2022, from https://towardsdatascience.com/introduction-to-recurrent-neural-network-27202c3945f3

Appendix:

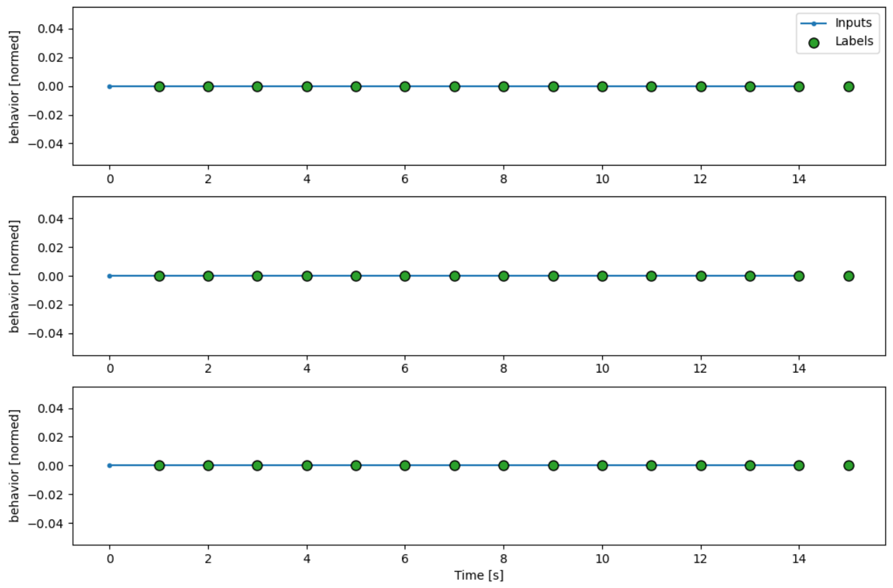


Figure 12: Single-Step Window: input=15, step=1

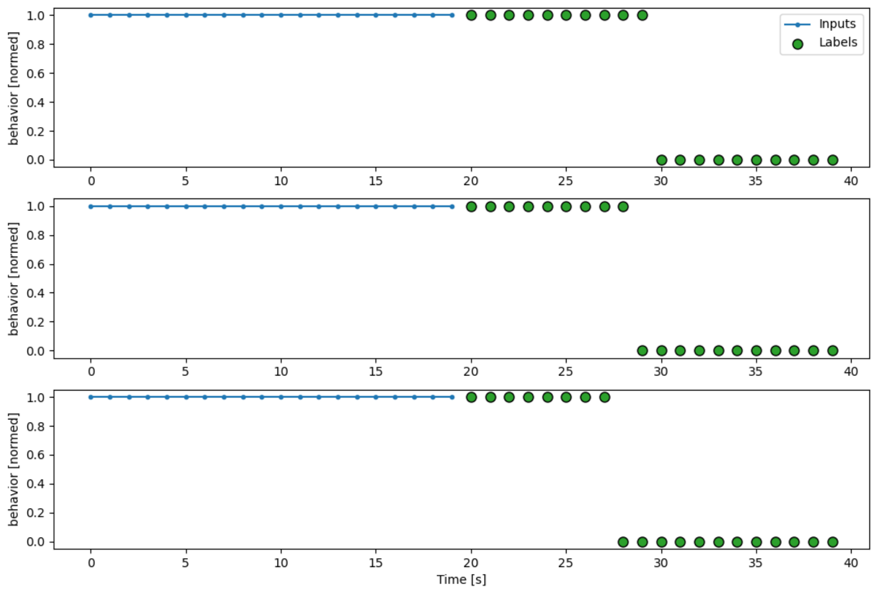


Figure 13: Multi-Step Window: input=label=step=20

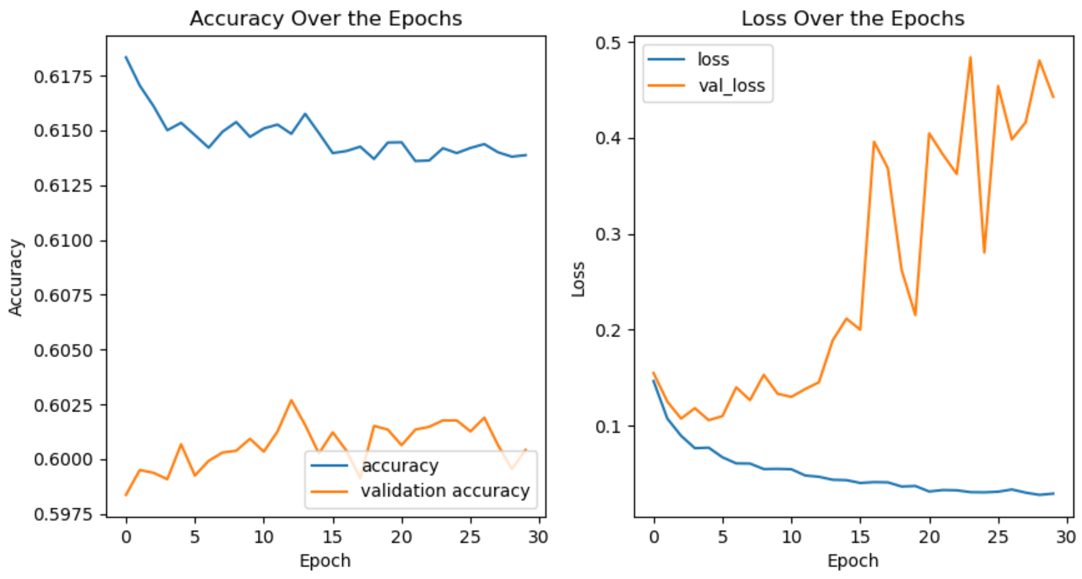
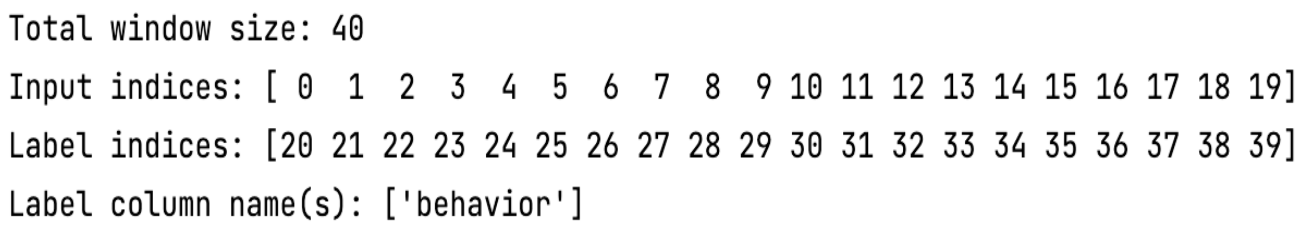


Figure 14: Accuracy and Loss Over Each Epoch(Multi Step)



```
Total window size: 40
Input indices: [ 0  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19]
Label indices: [20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39]
Label column name(s): ['behavior']
```

Figure 15: Window Information(Multiple)