



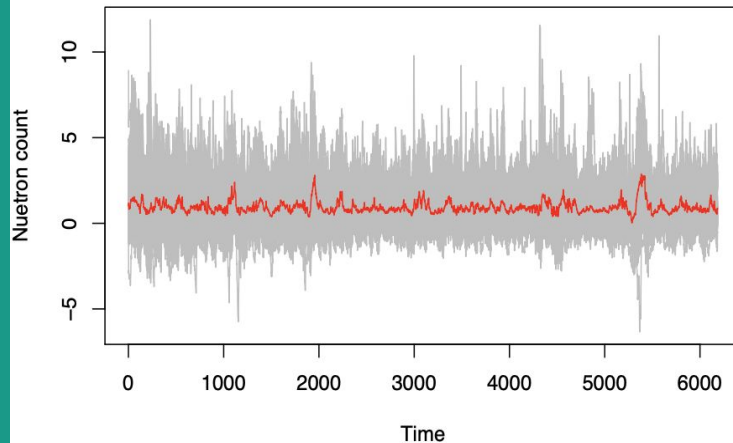
Mouse Project

Lihao Liao, Ruining Jia
Client: Cruz-Martin Alberto
Professor: Yajima Masanao

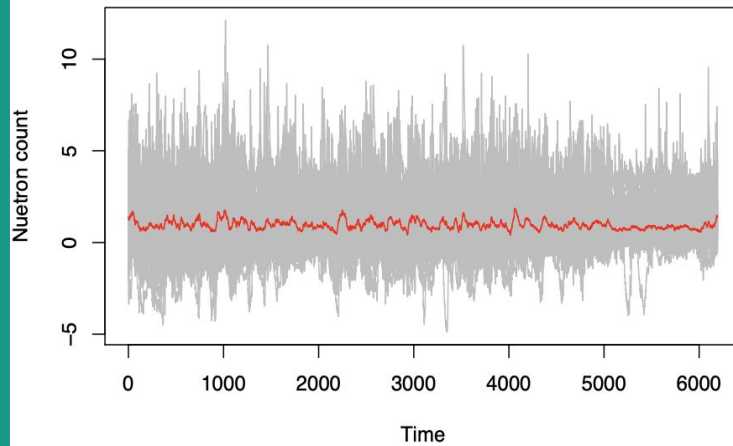
Introduction

Here are the plots of neutron counts over time for Mouse 1 and Mouse 2, in which the red line repents the mean neutron counts over time across all the cells.

Nuetron count over time for Mouse 1



Nuetron count over time for Mouse 2






Baseline Model

- Build of logistic regression model to predict and classify the behavior of each mouse (open arm =1, close arm 0). We need to only keep the column representing open arm in the behavior data.
- We need to combine data for Z score (neutron activity) and behavior into one data set.
- We can build a logistic regression model with mouse behavior (open arm or close arm) as outcome and neutron activity (Z scores) of each cell as predictor.
- Split the data into two data sets: Training and Test for each mouse. Training data were used to build the logistics regression model and test data were used to check the performance of the logistic regression model.

Table 1: Error rate and precision for logistic regression model of the test data

	False Negative rate	False Positive rate	Precision rate
Mouse 1	0.0875764	0.1723238	0.8949353
Mouse 2	0.0474334	0.1222571	0.9397201
Mouse 3	0.2638332	0.3138564	0.7199350
Mouse 4	0.2237569	0.3934837	0.7395777
Mouse 5	0.0366252	0.1269350	0.9476242
Mouse 6	0.0628723	0.1563422	0.9200000
Mouse 7	0.2951699	0.2427184	0.7343750
Mouse 8	0.0398366	0.1032028	0.9460317
Mouse 9	0.0250587	0.0456432	0.9716733
Mouse 10	0.1001410	0.1287356	0.8889860
Mouse 11	0.0537634	0.0811594	0.9388235
Mouse 12	0.1953125	0.2638889	0.7823795
Mouse 13	0.0893372	0.0659459	0.9276730



A Basic Neural Network Approach (Flattened Layer Network)

Aim: Predict the future behavior of a mouse given its current neural activity as well previous behaviors and neural activity.

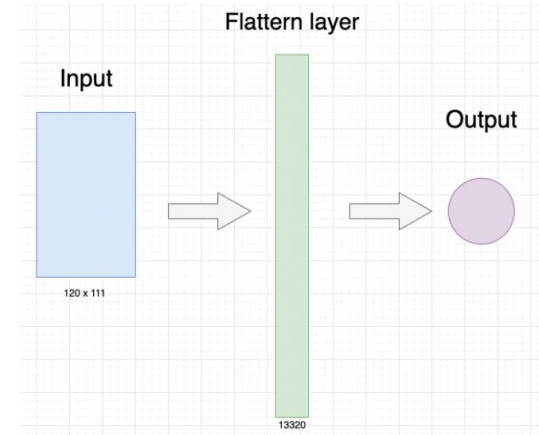
Framework: We choose to look back as far as 40 time points ago and draw one data point every 10 time points for predicting the behavior of a mouse for the next time point.

A Basic Neural Network Approach

An essential part of the neural network is that we need to process the raw data into a format that a neural network can ingest.

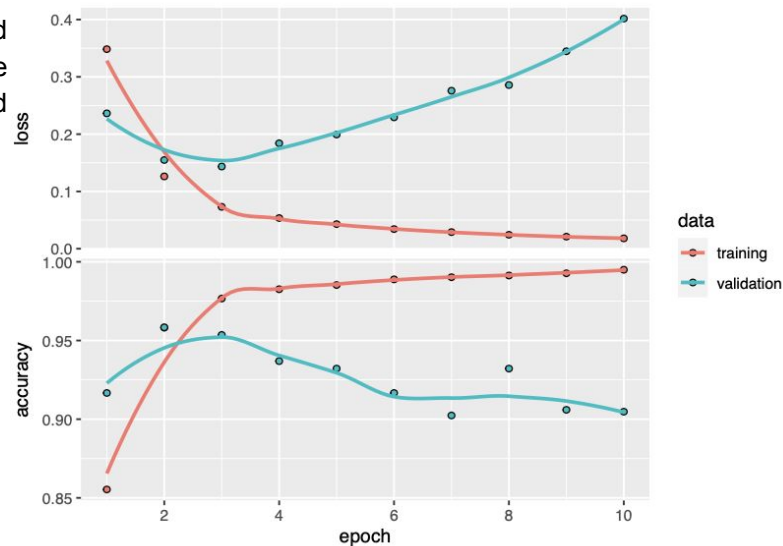
- Ways:**
1. Make sure that the data are numerical
 2. Normalized the raw data to make each time series be on the same scale
 3. Obtain a batch of data containing data in the past 40 time points as well as the “current” time point

We first fit a basic neural network model, which is a flattened layer network. A flatten layer will take an array input, as a flatten layer, i.e., an vector.



A Basic Neural Network Approach

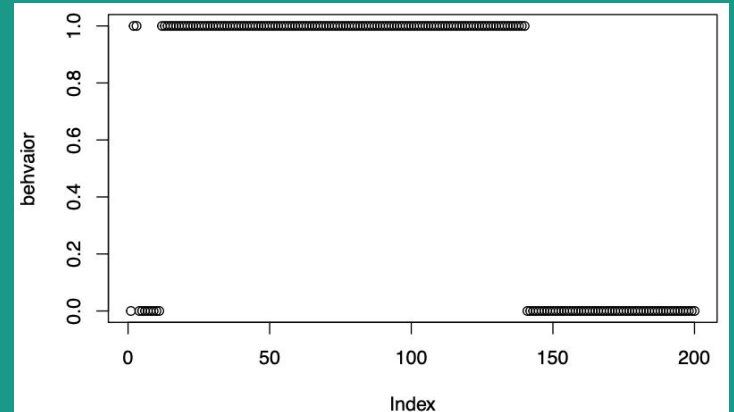
We split the data into three parts: training, validation, and test data. Data obtained at the first 3000 time points will be used as training data. Data obtained at the next 1000 time points will be used as validation data. The rest data will be used as test data.



Recurrent Neural Network (RNN)

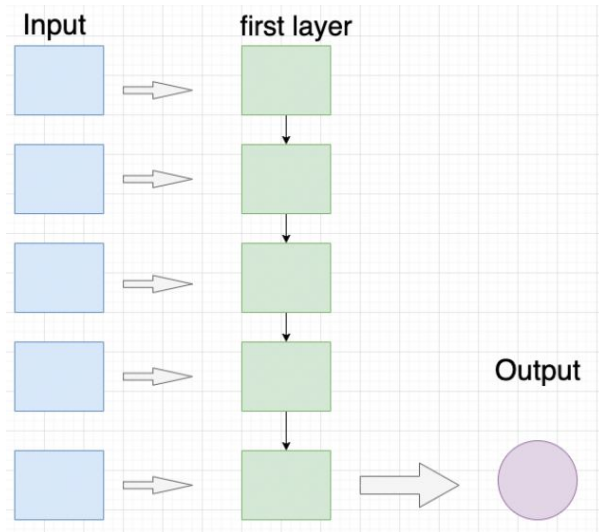
- A recurrent neural network (RNN) has often been used to predict an event that does not only depends on the predictors of the current time point but also depends on the previous events.
- In this project, the behavior of the mouse does not only depend on its current neural activity but can also be affected by its previous behaviors.

From the plot of the behavior of Mouse 1 during the first 200 time points, we can see that its behavior in the next time point tended to be consistent with its previous time points.

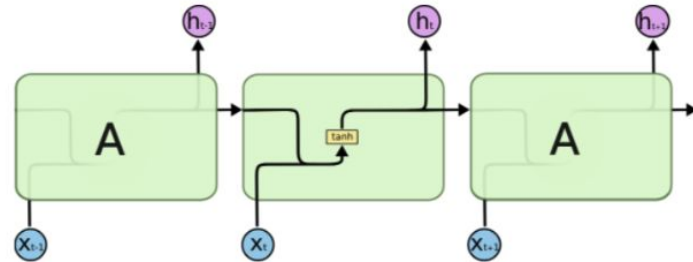


RNN

The following plot shows the basic idea for an one-layer many-to-one RNN.



Here is the unfolded-version of a simple RNN.





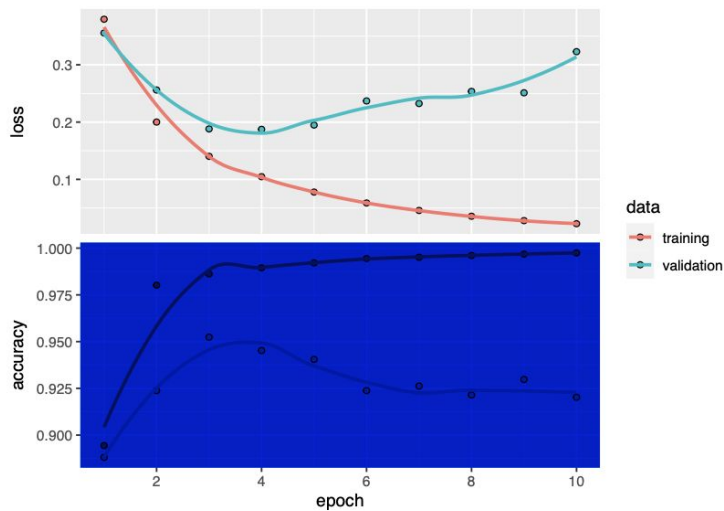
RNN

General Steps:

1. Input $X(t_1)$ is fed into RNN and generate an output $h(t_1)$.
2. Output $h(t_1)$ and $X(t_2)$ will both serve as input for the second step, which generate output $h(t_2)$.
3. Output $h(t_1)$ and $X(t_3)$ will both serve as input for the second step, which generate output $h(t_3)$.
4. This goes on until hit the final output $h(t_{k+1})$, where k is the number of time points we would like to look back.

Simple RNN

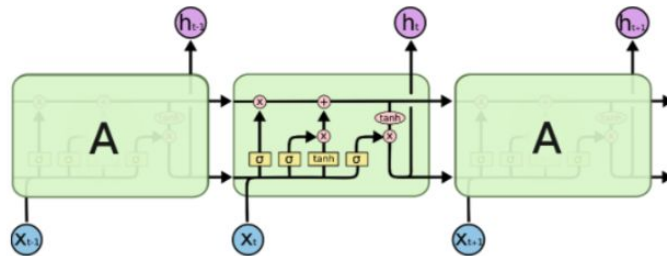
We first fit a simple RNN that look back as far as 40 time points ago to predict the behavior at the next time point. The accuracy of validation is about 0.9, which is not bad can be better.



Long Short Term Memory RNN

Long Short Term Memory RNN model (LSTM) will remember the previous information (long-term dependency) and use it for processing current input, as it is illustrated in the following picture. LSTM will “carry out” the previous output and feed it into the current training process.

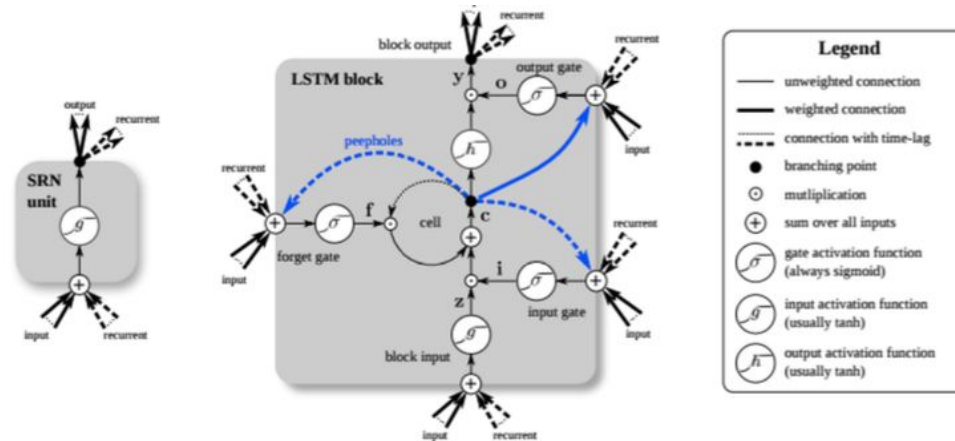
The basic idea of LSTM RNN is illustrated in the following diagram. Within each recurrent layer, there are four more interacting layers that controls the information fed in the cell state.



Long Short Term Memory RNN

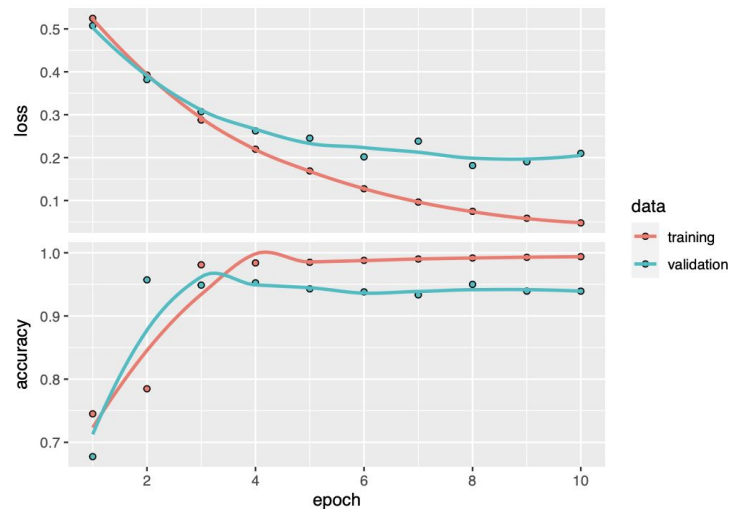
Detailed Framework:

- The Forget gate layer decides which information will be tossed away from the cell state.
- The Input gate layer decides which new information will be fed into the cell state.
- This information has then been passed into a tanh layer to create a vector of new candidate information, which will be fed into the new cell state.
- The output gate decides which information will be output



Long Short Term Memory RNN

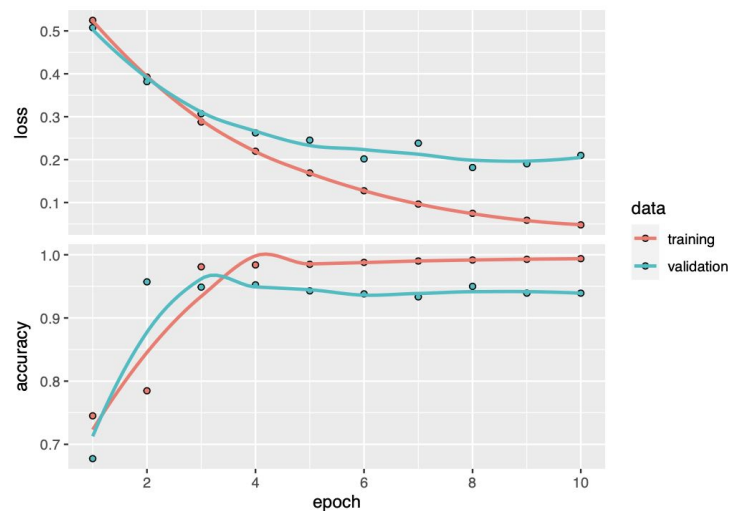
The accuracy of validation for the LSTM RNN is about 0.75, which is quite poor. However, this might be caused by overfitting issues, which will be addressed later



Gated Recurrent Unit RNN (GRU)

Gated recurrent unit is very similar to LSTM as they share the same principle. However GRU usually requires fewer parameters than LSTM, which makes it easier to run but also makes it have less representational power than LSTM.

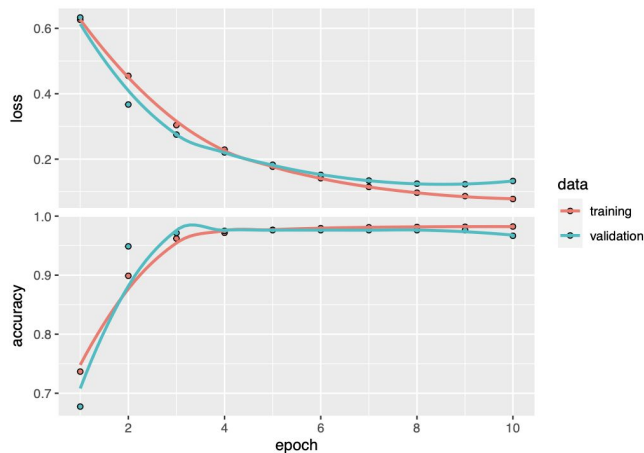
The accuracy of validation for the GRU RNN is about the same as LSTM, which is quite poor. However, this might be caused by overfitting issues, which will be addressed later



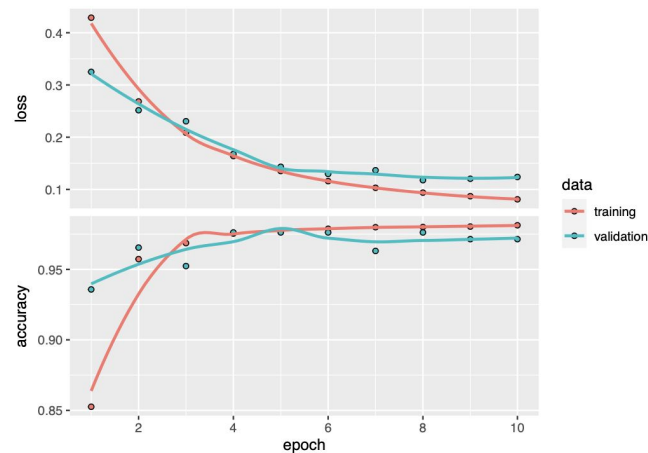
Address Overfitting

It is clear that there is overfitting issues for simple, LSTM, and GRU RNN models as the training and validation curves start to diverge from each other after a few training epochs. For the simple RNN model, we can randomly drop out a certain percentage of connections between layers to prevent overfitting issues. For LSTM and GRU RNN, such dropout should happen at each time step. In addition, a temporally constant dropout was applied to the inner recurrent activation, which will help to regularize the representation of the recurrent gates. In this project, we decided to drop out 20% of the connections between the input and first hidden layer and drop out 20% of the inner recurrent activation for the LSTM and GRU RNN.

LSTM



GRU





Conclusion

We decided to perform both LSTM and GRU RNN on all the mouse since GRU RNN requires less computation power and showed very similar or even better results than LSTM RNN.

Compare to Baseline Model:

Accuracy of LSTM RNN, GRU RNN, and Logistic Regression model of the Test Data

LSTM	GRU	Logistic
0.7000000	1.000	0.8949353
1.0000000	1.000	0.9397201
0.4833333	1.000	0.7199350
0.5916667	1.000	0.7395777
1.0000000	1.000	0.9476242
1.0000000	1.000	0.9200000
1.0000000	0.575	0.7343750
0.1000000	1.000	0.9460317
0.7416667	0.875	0.9716733
1.0000000	0.875	0.8889860
1.0000000	1.000	0.9388235
1.0000000	1.000	0.7823795
1.0000000	1.000	0.9276730

Thank you.
