ELEC3810 Final Project Report

Name: Young, James Yang (20740589)

1. Data Check and Pre-processing

After loading the trainSpike and trainState matrices, the first step I did was filter the data by removing all NaN values from both matrices, trimming both dataset length from 13145 to 1500 valid data.

Then, data checking was performed with PCA and t-SNE to see the separability of the filtered dataset. Figure 1 shows the 2D PCA and 2D t-SNE scatter plots with button press states shown on the plot. As the plots show, the dataset has decent separability especially in the t-SNE plot, meaning I should see relatively good performance with most classifiers.

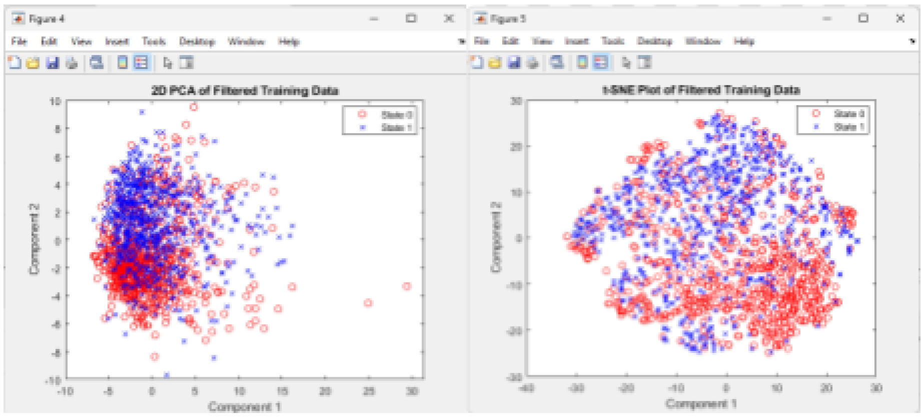


Figure 1: PCA and t-SNE Plots

Since the rat's movement is continuous, it is likely that previous spikes might contribute to the current movement. Therefore for data pre-processing I concatenated history spikes to the current spike to increase the dimension of the input from 16 to 16xN, where N is the number of history. Section 4 will discuss more about choosing an optimal history size.

2. Algorithm: LSTM

For choosing the algorithm to perform the binary classification, I decided to use Long Short-Term Memory (LSTM) recurrent neural network (RNN). RNNs use sequential and time-series data, which in this case is well-suited as the rat's movement is continuous making it a suitable model for this classification problem. Although simpler models such as SVM would probably perform well on this dataset as seen in the separability of the dataset in the section above, I wanted to try and use a neural network instead.

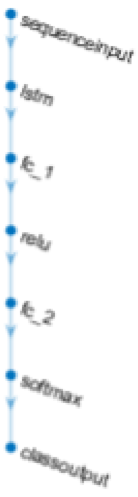


Figure 2: Network Architecture

2.1. Network Architecture

The architecture of the neural network is shown on Figure 2. The input layer has an input dimension of 16xN, where N is the number of history. Then, it passes to an LSTM layer with 64 hidden units and is fed forward to a fully-connected layer with a ReLu activation function and output dimension size of 32. Then it is passed to another fully-connected layer with a softmax activation function and outputs size of 2. Finally it is passed to a classification layer that computes the cross-entropy loss for classification.

3. Training

For training the model, I split the training dataset into 80% training and 20% validation, meaning that 1200 of the 1500 filtered data will be used for training and 300 will be used for validation. I used the Matlab cvpartition function for extracting indices for cross-validation. Each run goes through 30 epochs and with a batch size 64 means that there are 540 iterations (18 iterations per each epoch).

4. Parameter Exploration

Finding the optimal hyperparameters was done manually and mainly consisted of trial and error with trying different values and evaluating the performance.

4.1 Learning Rate

For tuning the learning rate, I tried different learning rates set to 1e^{-1}, 1e^{-2}, 1e^{-3}, and 1e^{-4}. Learning rate 1e^{-1} converged early and performed poorly, most likely as the rate is too high. Learning rates 1e^{-2}, 1e^{-3}, and 1e^{-4} had close performances, but ultimately I found that learning rate 1e^{-2} had the best validation accuracy even when tuning other parameters. Figure 3 shows the learning rate comparison with validation and training accuracy shown.

4.2 History Size

For the input history data, I tested N=[0, 1, 5, 8, 10]. Overall, the best size was found to be 8 as shown in Figure 4 below.

4.3 Figures

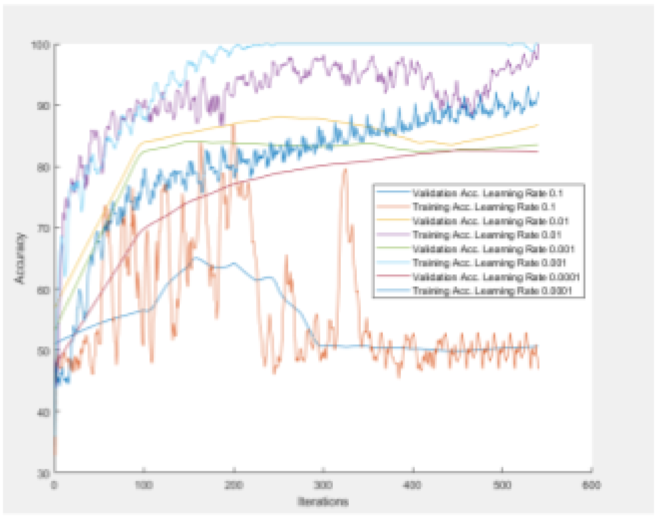


Figure 3: Learning Rate Comparison

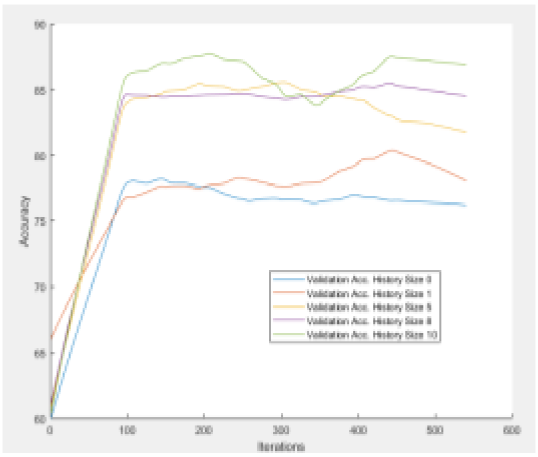


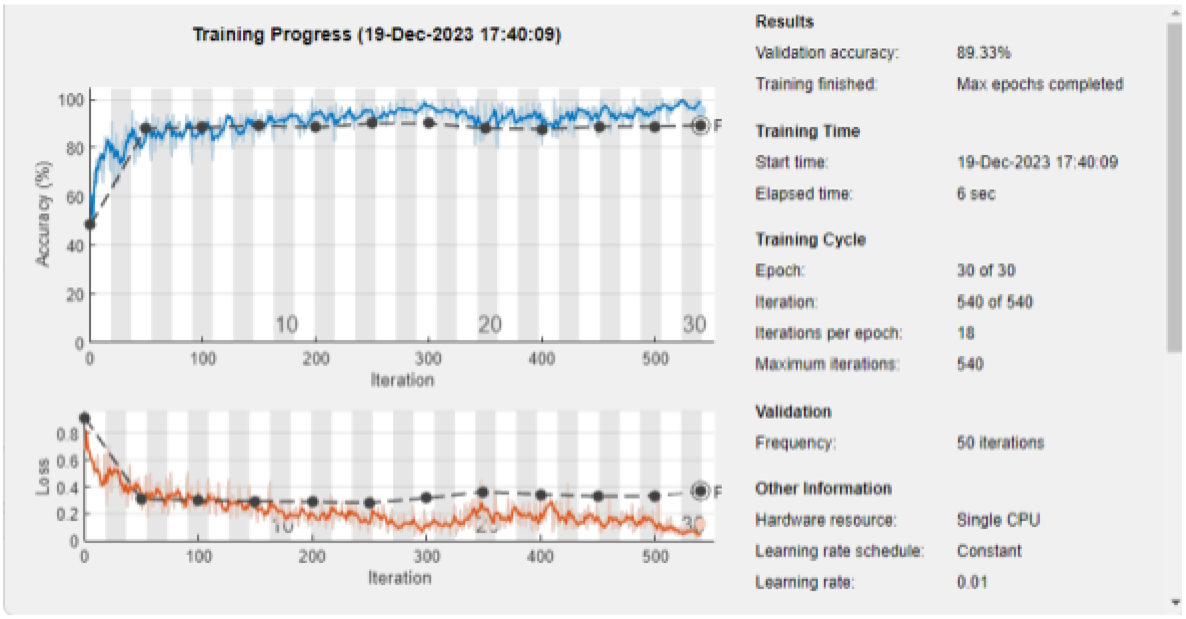
Figure 4: History Size Validation Accuracy

4.4 Other Parameters

Other parameters such as batch size were tuned but I found less significant difference when tuning these parameters as compared to the parameters above. Also, adding more layers made only minimal improvements to validation accuracy whilst increasing computation time and hence why I didn't add any more layers to my final neural network. Using Adam performed better than RMSProp from my testing. For the LSTM hidden units I chose 64 as low values performed worse and choosing too high values may overfit the data.

5. Performance

In the end, my highest validation accuracy achieved was 89.33% as shown in Figure 5 with the following hyperparameters: learning rate (1e^{-2}), optimizer (Adam), history size (10), batch-size (64), LSTM hidden units (64). The confusion matrix is shown in Figure 6 below.



Results

Validation accuracy:	89.33%
Training finished:	Max epochs completed

Training Time

Start time:	19-Dec-2023 17:40:09
Elapsed time:	6 sec

Training Cycle

Epoch:	30 of 30
Iteration:	540 of 540
Iterations per epoch:	18
Maximum iterations:	540

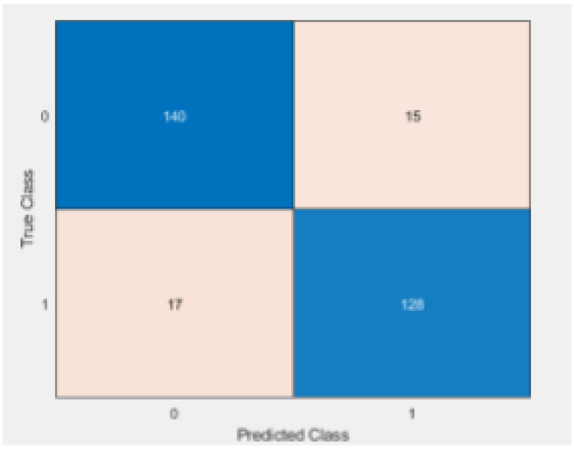
Validation

Frequency:	50 iterations

Other Information

Hardware resource:	Single CPU
Learning rate schedule:	Constant
Learning rate:	0.01

Figure 5: Training Result of Best Run



True Class \ Predicted Class	0	1
0	140	15
1	17	128

Figure 6: Confusion Matrix