ELEC3810 (Data Science for Neural Engineering) Final Project Report

Tara Relan (20716928)

1. Introduction

The aim of this project is to classify the rat's lever-press state (press or rest) using neural firing information (spikes). We developed a neural network model in MATLAB that could accurately decode the rat's state based on the recorded neural activity.

2. Data Visualisation and Preprocessing

We extracted the data from the dataset, that is, spikes corresponding to rest (0) / press (1) state. After removing the NaN values from trainSpike and trainState, there were 1500 valid datapoints out of 13145 total datapoints. Therefore, trainSpike became a 16*1500 matrix, and trainState became a 1*1500 matrix.

In machine learning it is important to separate the classes so there is accurate classification and understanding of the underlying patterns and relationships in the data. Therefore, we used PCA (Principal Component Analysis) and t-SNE (t-distributed Stochastic Neighbour Embedding) to visualise the data (red indicates rest (0) and green indicates press (1)). As Figure 1 shows, t-SNE shows a greater separability than PCA, meaning the machine learning model is more likely to correctly classify the rat's lever-press state.

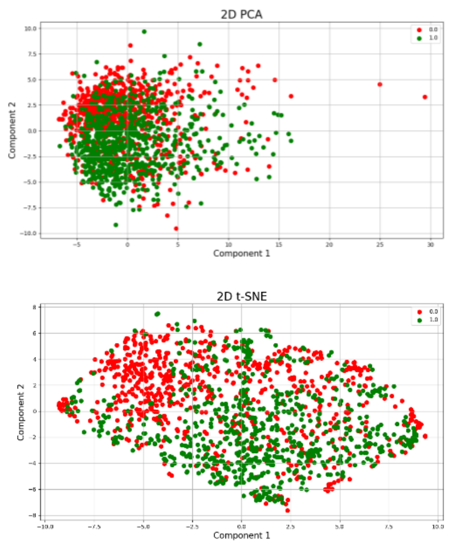


Figure 1: Result of applying PCA and t-SNE on the dataset.

3. Model Architecture and Hyperparameter Tuning

The model architecture used in this project is a feedforward neural network with multiple hidden layers, as follows:

$x \rightarrow bn \rightarrow fl \rightarrow fc \rightarrow af \rightarrow dr \rightarrow (fc \rightarrow af) * 5 \rightarrow fc \rightarrow sm \rightarrow y$

Where:

Variable	Meaning	Dimension/Type
x	Input	16
bn	Batch Normalisation	
fl	Flatten	
fc	Fully Connected	32
af	Activation Function	ReLU
dr	Dropout Rate	0.1
sm	SoftMax	
y	Output	2

3.1 Activation Function

The activation function of a neural network introduces non-linearity of the model and determines the output of a neuron based on the weighted sum of its inputs. In this project ReLU was used, which is computationally efficient and helps alleviate the vanishing gradient problem, allowing for faster convergence of the model.

3.2 Optimisation Function

In this project, the optimisation function used was Adam (Adaptive Movement Estimation), an algorithm that combines adaptive learning rates and momentum. Adam dynamically adjusts the learning rate for each parameter based on past gradients, ensuring the learning rate is appropriate and convergence is achieved.

3.3 Learning rate

The learning rate determines the step size at which the model adjusts its parameters during the training process. In this project, a learning rate of $1 * 10^{-2}$ was used. A smaller learning rate suggests a more cautious and gradual adjustment of parameter values, which helps when dealing with complex or noisy data.

3.4 Number of folds

This project used k-fold Cross-Validation with k=5, meaning that the input data was split into 1200 datapoints for training and 300 datapoints for validation. This will ensure the model's ability to generalise across multiple subsets of the data.

3.5 History

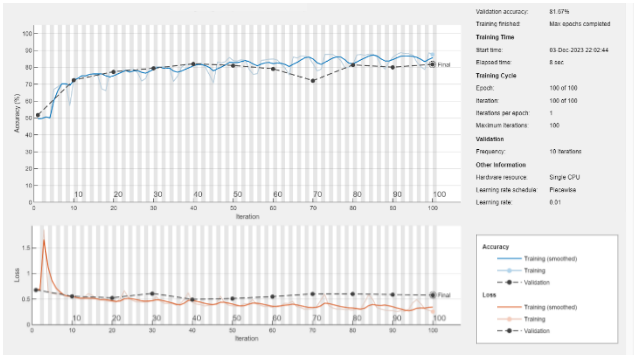
As the historical spike rate is densely correlated with the state information, it is important to account for this while training the model. A longer history length can capture more temporal dependencies in the data, which can be beneficial for modelling sequential patterns. This project used history values in the range of [5, 12], and found that the histories giving the best training and validation accuracies were in the range of [6, 10].

3.6 Random seed

The random seed, set as 3810 in the code, is a value used to initialise the random number generator. Using a random seed ensures the sequence of random numbers generated during training and validating becomes reproduceable, meaning the results obtained from the model will be consistent every time it's run with the same seed.

4. Training and Validation

In each k-fold training step, the model with the best validation accuracy was saved. The best model can be seen in Figure 2 as the training accuracy was 90%, validation accuracy 82%, training loss was 0.2 and validation loss 0.5.



Validation accuracy: 81.67%
Training finished: Max epochs completed
Training Time
Start time: 03-Dec-2023 22:02:44
Elapsed time: 8 sec
Training Cycle
Epoch: 100 of 100
Iteration: 100 of 100
Iterations per epoch: 1
Maximum iterations: 100
Validation
Frequency: 10 iterations
Other Information
Hardware resource: Single CPU
Learning rate schedule: Piecewise
Learning rate: 0.01

Figure 2: Accuracy and Loss Plots for Training and Validation

5. Performance Evaluation

5.1 Confusion Matrix

As mentioned earlier, the training model shows an overall accuracy of 90%, with a confusion matrix in Table 1.

		Predicted	
		1	0
Actual	1	507	89
	0	32	572

Table 1: Confusion Matrix

From the confusion matrix, the following metrics can be measured:

True positive (TP): 507

False negative (FN): 89

False positive (FP): 32

True negative (TN): 572

5.2 Sensitivity & Specificity

Sensitivity, also known as the true positive rate, is the proportion of actual positive instances that are correctly identified as positive by the model. It measures the model's ability to avoid false negatives.

$$Sensitivity = \frac{TP}{TP + FN} = 0.851$$

Specificity, also known as the true negative rate, is the proportion of actual negative instances that are correctly identified as negative by the model. It measures the model's ability to avoid false positives.

$$Specificity = \frac{TN}{TN + FP} = 0.947$$

5.3 gMean

The gMean (geometric mean) is sometimes used as a metric in evaluating classification models, and is calculated as:

$$gMean = \sqrt{\frac{TP}{TP + FN} * \frac{TN}{TN + FP}} = 0.898$$

As the gMean is close to 1, it indicates that the model’s performance in terms of balancing the true positive rate and true negative rate is relatively high.

6. Decoding Results

Using the best model, it was then applied to the test spike dataset, resulting in the decoded states as seen in result.mat.

7. Conclusion

This project aimed to classify a rat’s lever-press state based on recorded neural activity and a feedforward neural network was developed to do so. Data visualisation techniques such as PCA and t-SNE were implemented to gain insights into the data distribution and found t-SNE to be more effective in separating the rest and press states. The model architecture consisted of multiple hidden layers with ReLU activation function and Adam optimization. A training accuracy of 90% and a validation accuracy of 82% was achieved, indicating the model's ability to generalize well. The sensitivity and specificity of the model were 0.851 and 0.947, respectively, showing its effectiveness in avoiding false negatives and false positives. The gMean value of 0.898 further confirmed the balanced performance of the model.