EE40098: Neural Spike Sorting Assignment

Candidate No: 12847

This document details the development of a spike sorting algorithm written in Python.

Operation: Once the training and submission data have been imported, the locations of the spikes within the training data are sorted into ascending order. Two filters are applied to both the training and submission data sets: a Butterworth band-pass filter and a Savitzky-Golay filter. This is done to reduce noise and to smooth the data. A spike detector is then run on both of the filtered data sets. This uses scipy's 'find_peaks' function, which returns the location of the peaks within the data set. To match the given spike locations within the training dataset, the gradients of the previous 15 values are calculated, and the point at which the gradient is steepest is where the beginning of the spike is defined. The complete training data is then divided into a training dataset and a validation dataset. This is so the network can be tested on data it has not seen before. A 90%/10% respective split was chosen to ensure the network has sufficient data for both training and validation, whilst providing a reliable success rate on validation classification. The spikes detected within the validation data can now be compared against the known spikes within this region. This is achieved by checking if the detected spike exists within the list of known spikes, allowing for a degree of error. 50 datapoints are stored for each spike and the multilayer perceptron neural network is trained on the training data set. An MLP neural network was chosen due to the ease of implementation and because of the success achieved with an MLP neural network for the MNIST classification problem, which is similar to this spike classification assignment. Before running the network on the entirety of the submission data, it is tested against the validation data set and against the first 25 manually identified spikes of the submission data set.

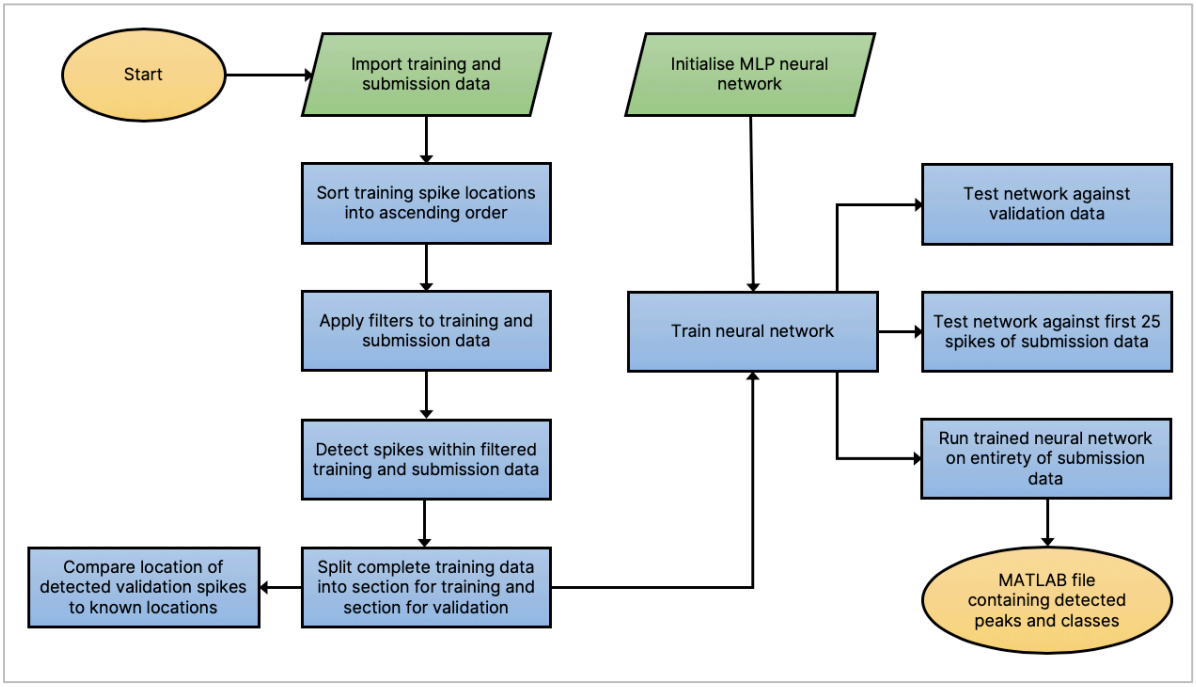

Figure 1: Spike sorting algorithm code flow chart diagram

Results: 94% of the spikes within the validation dataset were identified correctly by the peak detector function. Of those peaks correctly identified, approximately 94% are classified correctly. Therefore, approximately 88% of the spikes within the validation data are classified correctly. 92% of the first 25 peaks within the submission data were correctly classified. These metrics indicate that the classifier should perform well on the entirety of the submission data. 2080 spikes are detected within the entire submission data set with an approximate class breakdown of: 620 Type 1, 450 Type 2, 450 Type 3 and 560 Type 4.

Future Improvements: The performance depends greatly on various filter, peak detection, and neural network parameters which could be optimised through the use of another machine learning technique. Principal Component Analysis could be utilized to potentially improve the classifier performance. The spike detector function should be run only on the validation data, rather than all of the training data, to improve efficiency.