The University of Melbourne - Department of Mechanical Engineering

Activity Detection from EEG signals using CNN and LSTM

Tuan Khoi Nguyen (1025294) Fawad Ahmed Malik (1004550) Sean Breukel (1080348)
Victor Bouchet-Hibbert (915058)

Introduction

Electroencephalography (EEG) are signals across the brain, which have been known to change according to different activities and emotions of human. Current technologies have been able to capture EEG as multiple signals at different areas of the brain called channels, hinting a potential of using them to detect human activities. However, data in EEG channels can be massive and noisy, which makes it hard to be able to incorporate all information or discover representations effectively. In this report, we propose the use of wavelet in noise filtering process, and subsequently use deep learning to capture information in both spatial and temporal domain with the use of 2 common neural network architectures - Convolutional Neural Network (CNN) and Long-Short Term Memory (LSTM). The methods will be introduced in details with reproducible steps available for the base code of the project, followed by throughout result analysis and suggestions.

1 Background

1.1 EEG & Related Works

Electroencephalography (EEG) are activity signals of the brain's neural system, which can be represented as a heatmap-like structure distributed around human skull [1]. Different behaviors and feelings can be reflected on a sequence of these heatmaps, which can vary to each specifically. Due to this nature, it can be said that EEG would contain information in both spatial and time domain [1, 2]. Several studies have investigated into the extraction of information from EEG signals for human activity recognition using domain knowledge to process data into different features [2, 3]. However, these methods requires expertise knowledge in neurology studies, making these approaches limited its development to specialists. Such problems can be solved by approaches that utilise more on deep learning, where arbitrary representations can be learned with fewer fed-in domain knowledge.

1.2 Dataset & Task

The dataset used from this report is recorded on a 500-Hertz frequency from a grasp-and-lift trial of 12 participants, where multiple series are represented as 32 channels across 200,000 time frames per series on average [1]. The task is to detect 6 possible activities that may have occurred in the series.

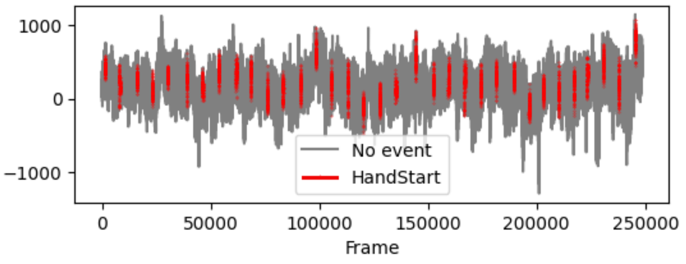


Figure 1: Activity data example from a sequence channel

2 Pipeline Structure

2.1 Data Format & Preprocessing

HS	FDT	BSLP	LO	R	BR	Count
0	0	0	0	0	0	15687584
0	0	0	0	0	1	321905
0	0	0	0	1	0	321905
0	0	0	0	1	1	146095
0	0	0	1	0	0	388279
0	0	1	0	0	0	134690
0	0	1	1	0	0	49296
0	1	0	0	0	0	184452
0	1	0	1	0	0	204
0	1	1	0	0	0	253123
0	1	1	1	0	0	30221
1	0	0	0	0	0	468000

Table 1: Distribution count of events in the training dataset

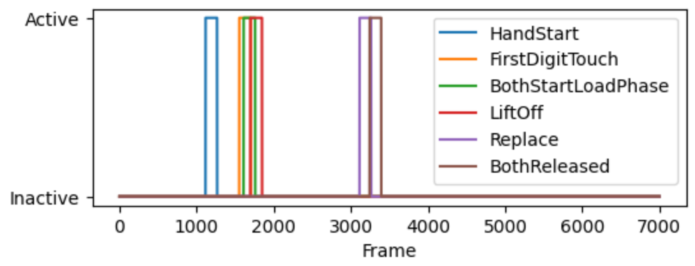


Figure 2: Signal sample where multiple activities happened in the same duration

Figure 1 shows an example data series from the dataset. The majority of the time frames have no activity, suggesting that some measures are needed for label balancing. Part of its ground truth activity in Figure 2 shows that multiple activities can overlap. Label distribution in Table 1 confirms this, as high appearance can also be found in some specific combinations such as *FDT* and *BSLP*. Therefore, it shows a potential of activities may also have close correlation to each other, where doing one can likely follow up with another activity happening next. To make the best use of this information, the problem will be treated as a multi-label classification problem. The next paragraphs will also demonstrate some preprocessing on the data based on its characteristics with the aim of improving the prediction quality.

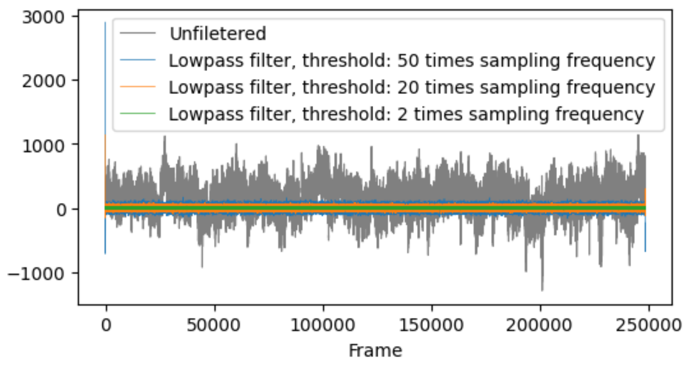


Figure 3: Signal with Fourier Transform Low-pass filtering at different frequency thresholds

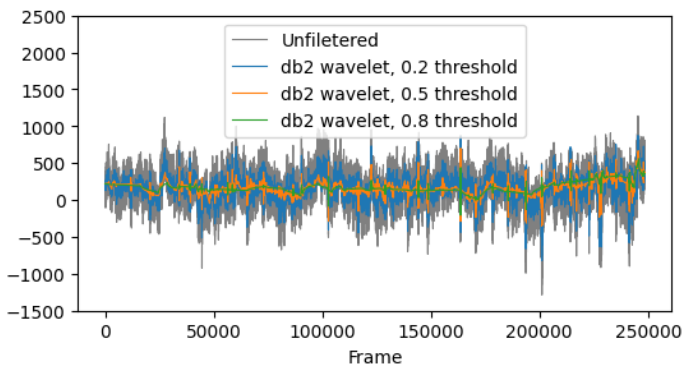


Figure 4: Signal filtered with 2nd-order Daubechies (db2) wavelet at different thresholds

Wavelet Noise Filtering EEG data consists of signal, which makes it prone to noisy data coming from measurement uncertainty or other external factors, which may increase the risk of model overfitting. In order to prevent this, 2 common frequency-based signal processing methods are investigated: Fourier transform and wavelets. Figure 3 shows results of applying a lowpass filter to remove high frequency oscillations from the data. Despite the frequency thresholds are at least 2 times larger than the sampling rate to satisfy the Shannon-Nyquist theorem and maintain all presumed important information [4], no clear signal characteristics are observed in the data. Wavelet transform on the other hand, maintains many notable details of the signal such as low-frequency fluctuations, as shown in Figure 4. This can be due to the fact that Fourier transform only considers frequency information, while wavelet transform considers both frequency information and their positions [5], making it useful in the dataset where patterns over time are common. Therefore, Wavelet transform is chosen as the noise removal method. Different thresholds will be tested for the optimal one, to ensure that noise is reduced suffciently, but not removing too much useful data information.

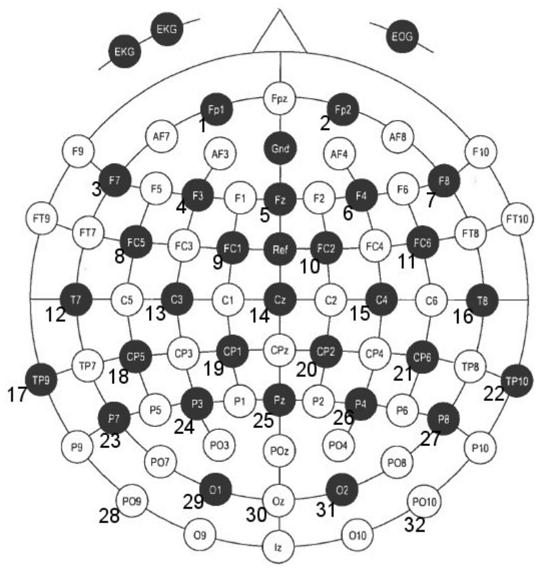
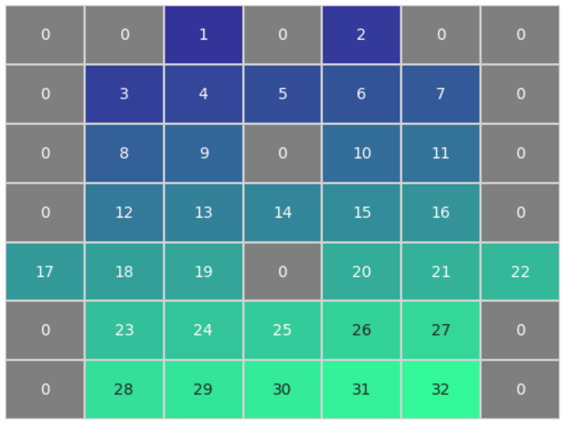


Figure 5: Real-world EEG structure with selected channel features in the dataset labelled as numbers [1]



0	0	1	0	2	0	0
0	3	4	5	6	7	0
0	8	9	0	10	11	0
0	12	13	14	15	16	0
17	18	19	0	20	21	22
0	23	24	25	26	27	0
0	28	29	30	31	32	0

Figure 6: 7×7 feature arrangement. '0' is assigned 0 value, other numbers correspond to marked numbers in Figure 5

Spatial Arrangement 32 channels in each time instance will be arranged into a 7×7 image as shown in Figure 6 to mimic the real world arrangement as shown in Figure 5. This ensures spatial information between channels can be maintained well and maximise information retrieval for models.

Channel	Min	Max	Mean	StDev
Fz	-245	594	131.79	127.00
PO9	-1287	1092	69.32	246.19
P4	-613	943	174.88	172.32
TP10	-669	1541	343.53	312.28
TP9	-953	1218	192.01	286.83

Table 2: Distribution of 5 random channels in a series

Scaling The channels are sections of the heatmap, hence share the same units [1]. However, inspection (Table 2) shows that the distribution between channels can be quite varied in range and span. To ensure optimal learning process by avoiding numerical imbalance, training data is unified by scaling to a normal distribution of mean 0 and standard deviation of 1, and the testing dataset is fitted to the new distribution.

Data Split Due to having no ground truth available for the test dataset, the validation will be done on part of the training data. Each subject in the training dataset have 8 labels with labelled chronological orders [1]. The train data will contain 1st to 6th series of each subject, and validation set will consist of the 7th and 8th series, making the split ratio roughly 75-25. This is to make the process goes in line with real life, where past data is used to predict upcoming future signals.

2.2 Model Architecture

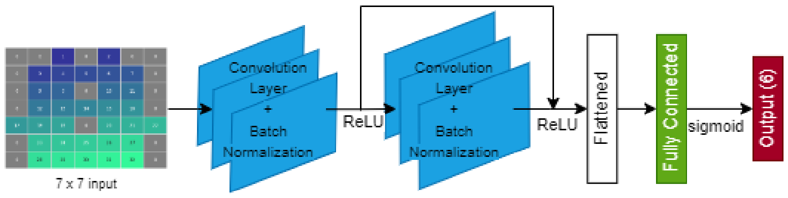

7 x 7 input
Convolution Layer + Batch Normalization
ReLU
Convolution Layer + Batch Normalization
ReLU
Flattened
Fully Connected
sigmoid
Output (6)

Figure 7: Flowchart of CNN architecture used in this report

2.2.1 Convolutional Neural Network (CNN)

Structure We implemented a CNN architecture to capture the spatial relationships between features (in this case EEG channels) and local patterns in the data. Initially, it was thought that the relationship between features and the label might be really complex, therefore, we decided to use a modified ResNet34 that was able to take in a structured 7×7 input data as shown in Figure 6. ResNet34 is a specific implementation of Residual CNN that uses residual and skip connections to overcome the problem of gradient vanishing in deep neural networks [6]. However, upon testing, it was found that the model was too complex for the task as it was severely overfitting to the training dataset, even with various regularisation techniques. Consequently, the complexity of the model was reduced and the implemented residual CNN is shown in Figure 7 and Appendix Figure 14. As seen in Figure 7, the simplified network uses a single residual block and skip connection. It also uses multiple 2D convolution, batch normalization and ReLU activation layers before flattening into dense layer.

Input Since CNNs are particularly effective for processing structured grid-like data, we pass in the spatially structured 7×7 EEG data as input to the model. With label imbalance mentioned in the previous section, we had to balance the dataset to improve model learning. So, for training the Residual CNN, we kept all event instances in the dataset and include the same amount of non-event instances, balancing the train dataset. After that, the train set was standardised and shuffled to prevent each event from being clustered together and spreading them across the dataset so that the model learns the general relationship between features and labels throughout the learning process instead of focusing on learning one event in some proportion of the learning and then focus on another later on in the process. On the other hand, the validation dataset, used to evaluate the model, was only standardised and not balanced or shuffled.

Output To determine the labels, the model's dense layer network connects to an output layer with 6 neurons and correspond to each label. For our multi-label problem, Sigmoid function is used as the activation function that gives us a probability of each label happening in the time instance. This is important since there can be more than one labels present at an event instance as seen previously.

Tuning In order to optimise the learning, we carried out hyperparameter tuning on the initial learning rate for an exponentially decaying learning rate, the optimiser, the dropout rate (between the dense hidden layer and the output layer) and kernel regulariser. A grid search with the following hyperparameters was carried out:

- Initial learning rate: 5e-4, 1e-3
- Optimiser: ADAM, NADAM
- Dropout rate: 0.0, 0.25, 0.5
- Regulariser: L1, L2

The results of the 10 best hyperparameter combinations can be seen in the Table 5 in the appendix. The best selected one has an initial learning rate of 1e-3, ADAM optimiser, a dropout rate of 0.25 and L2 kernel regularisation.

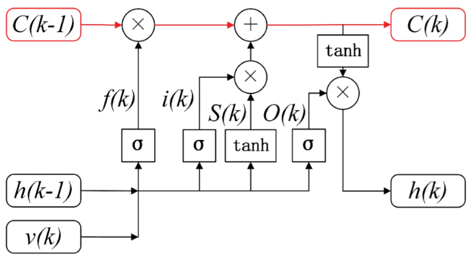


Figure 8: Structure of LSTM component [7]

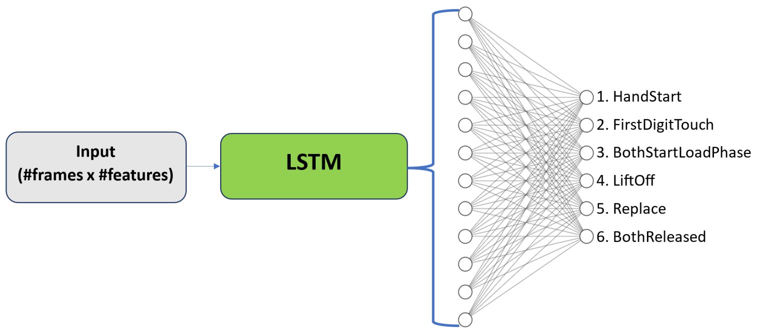


Figure 9: LSTM Architecture

2.2.2 Long-Short Term Memory (LSTM)

Description We decided to explore recurrent neural networks (RNN) in order to capture the historical data in the time series, which fits intuitively with the EEG task that we are working on as there are effectively 32 time series, one for each feature of the EEG data. LSTM (Figure 8) was our RNN of choice to avoid the common vanishing gradient problems that are common with RNN [8]. A simple LSTM architecture shown in Figure 9 is used which includes only one LSTM block and one hidden layer. This is done to reduce the chance of our model overfitting by having an overly complicated network which was observed in theory and in our initial model experimentation to lead to overfitting. The structure was selected through ad-hoc/rudimentary trialing of different parameters. We will stick to this simple model architecture throughout our training methodology in order to provide a clearer comparison between the hyperparameters which we are investigating, and also avoid making this section of the report overly convoluted by introducing further variables unrelated to the task-specific hyperparameters which we are hoping to test for.

Structure The model includes one LSTM block which takes in a fixed window of size $n_{rewind} \times n_{features}$, where n_{rewind} describes how many historical frames in the time series we wish to look back for in the data that LSTM operates with; this will be dictated by the hyperparameters that specify how far back to look when predicting for a frame (in seconds), and how much to subsample (to reduce the computational load of the default high sample rate of 500Hz). This LSTM block will output to a 64-neuron hidden layer, which will then be followed by a 50% dropout layer. Similar to CNN, the network will have its output layer of 6 neurons, having sigmoid as activation function. We train the model through a binary cross-entropy loss function, which allows us to treat each label prediction as a separate binary classification problem. We also use the Adam optimisation with learning rate set to default of 0.001.

Input & Tuning Before handing the training data into the network, we first perform scaling to the time series. As mentioned in section 2.1, scaling helps removing 'preference' when training based on the different features by allowing different inputs to have roughly similar magnitude. Once this has been done, we generate all the training samples, with one sample being for the prediction of a single frame - utilising the EEG data of that frame plus some historical frames. There are two hyperparameters that dictate how many frames are sent as input into the network:

1. $T_{lookback}$: how many seconds we want to look back in order to make a prediction of the current frame.
2. r_{sample}: downsampling from the original 500Hz to a less computationally expensive rate.

For the training data samples, we mostly select frames in the EEG data that are event frames (i.e., the presence of one or more of the events at this frame). To avoid unbalanced training towards the prediction of events occurrences, we introduce another hyperparameter, $r_{non-event}$, which specifies a proportion of the non-event frames to include in the training of the network. It should be noted that since we are predicting 6 distinct events at once, there will already be a lot of training for the absence of some of the labels/events even without the $r_{non-event}$, so this hyperparameter needed to be tuned along with the other two. To evaluate the model, we generate input samples for each frame of the test dataset to match the form that is set by $T_{lookback}$ and r_{sample} in training. Since the first few frames when the EEG recording starts does not have enough historical data, we left-pad the initial entries with 0s to account for this. The output from this inference is a prediction time series of the 6 events at each frame, giving a 0 to 1 value for the probability that each event is occurring at a given frame. We evaluate this holistically using the AUC-ROC metric, which will be further described in the next section.

3 Results

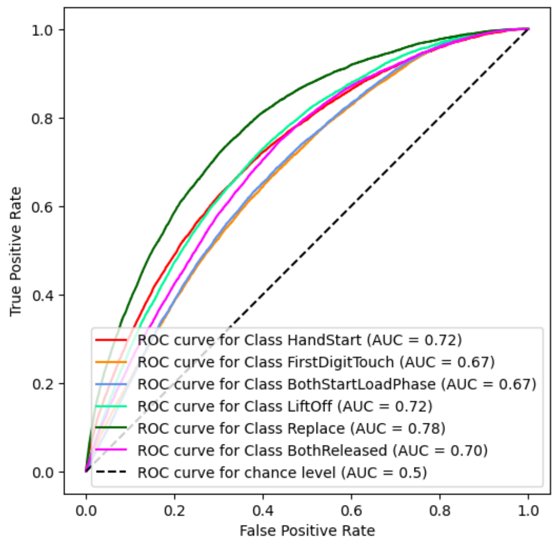


Figure 10: ROC Curve of Unfiltered CNN Model

Models	HS	FDT	BSLP	LiftOff	Replace	BR	Macro-average
CNN (unfiltered)	0.721	0.671	0.674	0.721	0.777	0.702	0.7110
CNN (filtered)	0.669	0.586	0.583	0.597	0.661	0.615	0.6186
LSTM (unfiltered)	0.80	0.85	0.84	0.81	0.86	0.85	0.8324
LSTM (filtered)	0.77	0.79	0.79	0.77	0.80	0.79	0.7841

Table 3: AUC Scores of Various Events for CNN and LSTM Models

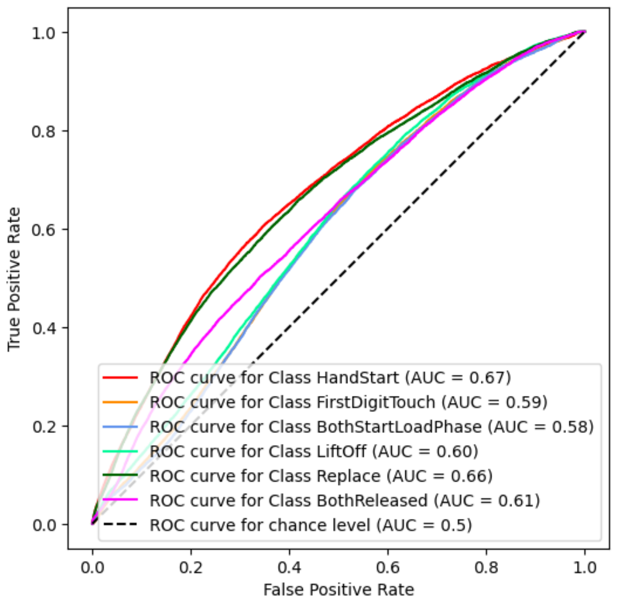


Figure 11: ROC Curve of Filtered CNN Model

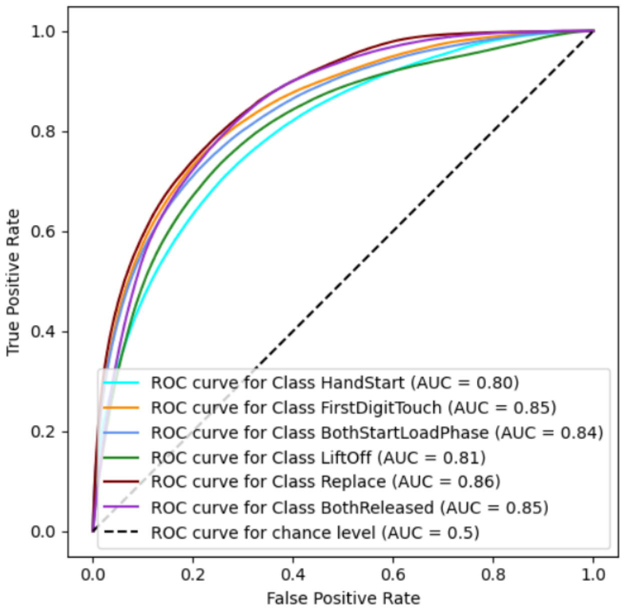


Figure 12: ROC Curve of Unfiltered LSTM Model

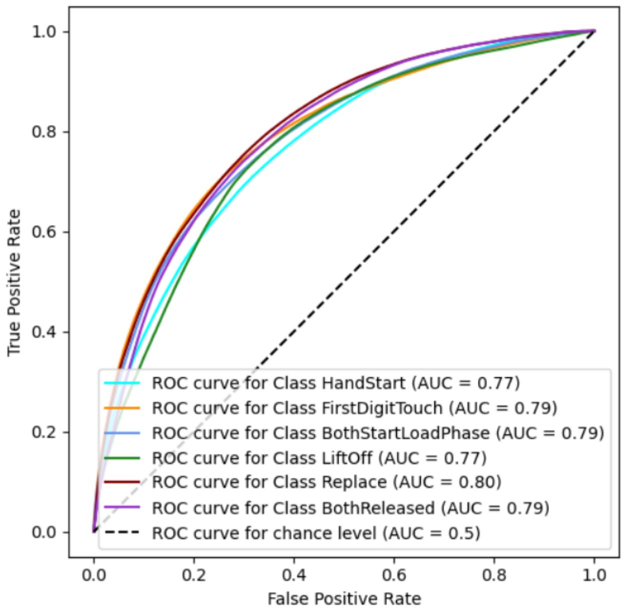


Figure 13: ROC Curve of Filtered LSTM Model

Metric Choice In this section we display Receiver Operating Characteristic (ROC) curves of each label for the trained residual CNN and LSTM models. The curve plots true positive rate against the false positive rate, and acts as a visual indicator on the results' degree of separability, that is the models' ability to distinguish classes from each other [9]. The area under the curve (AUC), which falls into a range of 0 to 1.0 is thus a useful numerical metric to evaluate the performance of the models, since the classification task is multi-label, where multiple classes can apply to a single instance at once. Here, Figures 10, 11, 12, and 13 display their respective model's ability isolate each individual class label from the rest. As an AUC score of 0.50 indicates that a model is completely unable to distinguish classes, models with AUC scores as close to 1.0 as possible are evaluated more highly. Results from these ROC curves have also been tabulated in Table 3.

4 Discussion

LSTM Performance For the LSTM model, experiments were performed (see Appendix Table 4) under the assumption that the default hyperparameters values of $T_{lookback} = 2$, $r_{sample} = 50$, and $r_{non-event} = 0.25$ provided a baseline macro-averaged AUC score, 0.831 for the unfiltered dataset. The AUC scores across the various actions were macro, as opposed to micro or weighted, averaged as all the individual actions are considered equally important. Increasing the threshold of the filter applied to the dataset in the preprocessing stage induced greater decreases in average AUC score for the LSTM model the higher the threshold applied. Across thresholds of 0.15, 0.3, 0.5, and 0.8, the AUC score on the unfiltered data decreased to 0.521. This indicates that the filtering causes important historical to be removed that would have allowed the LSTM to establish useful connections.

LSTM Tuning In the hyperparameter tuning, when adjusting values for the $T_{lookback}$ and r_{sample}, care had to be taken not to have the model track too much data for the RAM to handle, as was the case for $T_{lookback} = 2$, $r_{sample} = 25$, and $r_{non-event} = 0.25$, which resulted in twice the amount of data as the default values and went above the RAM's capacity. Increasing the $T_{lookback}$ for the LSTM had generally positive effects on raising the AUC score, even when r_{sample} had to be increased in response, giving a 0.014 increase. This indicates a temporal relationship between the data instances that spans a considerable length of time. The influence of $T_{lookback}$ also supercedes that of r_{sample} as, despite the additional density of data it provided, it was not enough to the stop the model's AUC score from decreasing. The impact of $r_{non-event}$ was neither as notable nor as straightforward as $T_{lookback}$, as decreasing the threshold decreased AUC by 0.007, and increasing it decreased AUC by 0.010, indicating that the events themselves are relatively self-contained temporally. The highest AUC from hyperparameter testing was 0.784, a 0.24 increase from the default value results of 0.760. This was achieved using $T_{lookback} = 3$, $r_{sample} = 50$, and $r_{non-event} = 0.10$, with the $r_{non-event}$ decreased to offset the increase in data for the RAM to handle from the higher $T_{lookback} = 3$. Ultimately however, the hyperparameter tuning was not enough to have the filtered dataset achieve a higher AUC than that of the unfiltered dataset.

CNN Performance The residual CNN model with tuned hyper-parameters provided a macro-averaged AUC score of 0.7110 for the unfiltered data and 0.6186 for the filtered data with a threshold of 0.15. Similar to LSTM, it was observed that data with higher filtering thresholds returned lower AUC scores, indicating the loss of important feature information during the filtering process as mentioned previously. These AUC scores are much lower than the validation AUC scores observed during the tuning (see Table 3 and 5). Therefore, we can say that even after using tuned hyper-parameters and multiple regularisation techniques, the model overfits. However, we have to note that the validation set was shuffled and balanced whereas the test set was neither shuffled, nor balanced. Consequently, the validation AUC score was not reflective of the actual sequence of the EEG data.

CNN Tuning Hyper-parameters tuned for the residual CNN focused on how the model learns. Initial testing showed that the models severely overfit to the training dataset, hence regularisation techniques, optimiser and learning rates were chosen to be the hyper-parameters to be tuned. According to the tuning results, it was evident that the regularised models performed better than their original counterparts, even though the difference was not too significant in the AUC score.

Observation In general, results indicate that the temporal information in the data is significant and since the CNN only takes spatial information into account while making predictions, it misses out on the important temporal relationship between the features and the labels. This argument can further be supported by comparing the AUC scores of the LSTM and CNN models. LSTM consistently achieves higher AUC scores for each label. This hints the future improvement by combining them together to further utilise information retrieval.

5 Limitations and Future Improvements

5.1 Time constraints

The time given to perform the tasks are limited in a duration of 3 weeks, making it difficult to develop a strong method that has proper tuning or validation. Furthermore, online platforms such as Google Colab - the main engine to carry out the tasks, were limiting the time for GPU processor and code usage, making any operation feasible only if they could finish all preprocess, train and validation process within a timeframe of roughly 2 hours. Further research and experiments would be possible when more timing allows.

5.2 Computational constraints

When tuning the hyperparamters for the LSTM, some combinations will yield more memory required. For instance, looking back further, less downsampling and including more non event frames will make the training samples more memory intensive. At one stage 48 gigabytes of RAM was used by the script. This partially limited the range that we were able to investigate for the hyperparameters. More memory optimised code should be used in future, such as using a generator to produce the training data.

Additionally, we were not able to test for all combinations of hyperparameters and filtered data to find an optimal methodology for LSTM. With greater resources, this would be a simple (though time consuming) extension to our project. To do this we could perform something like grid search or randomly sampling the hyperparameter combinations to attempt to find the best overall combination. However, we can use the results in this report to guide us towards what sort of selections may be more applicable to the task (for example how far to look back, etc.).

As for CNN, we could not train and therefore test the model on all the subjects due to RAM limitations. The maximum number of subjects that could be trained and tested at a time was 6.

5.3 Inadvertently changing LSTM architecture

Additionally, changing the hyperparameters affects the input size to the LSTM network, meaning that we are not quite comparing apples to apples in the different trials. To constrain the architecture to be identical between various hyperparameter tests, we should fix the input size to the largest that we used, and then pad whenever a smaller input size is encountered.

5.4 Changing the input to the CNN model

In order to allow CNN to capture temporal data, we can look into stacking multiple input instances into single input to the CNN model (for example: stack 32 instances, each containing 32 features, to get an input of size 32x32). However, this will mean that CNN has to wait for a particular amount of time to accumulate enough instances to generate input and then give a single prediction for the stack of multiple instances.

5.5 Considering time complexity for Realtime inferencing

Something that has not been considered in the report is the time complexity of our prediction since the use case in classifying the EEG data is to be used in real-time for desired event/action detection based on brain waves. Future work/extensions may involve optimisations to speed up performance and making this a metric that is prioritised when comparing models since a model that is too slow to keep up with the input data will not be very useful for future practical applications in EEG data interpretation.

5.6 Enhancing the Model architecture

Since we only considered a very basic LSTM architecture, it may be worth investigating whether deeper (or simply different) networks are able to outmatch the performance that we have obvserved in this project. This architecture fine-tuning can involve using more complex layers such as bidirectional LSTM layers, other forms of RNN such as GRU, chaining together more LSTM blocks or even other layers such as mixing in CNN + LSTM to combine both technologies that we investigated. Furthermore, adding in other regularisation techniques and tuning the standard hyperparameters of the network such as learning rate and loss function would be included in these future endeavors.

5.7 Wavelet filtering

Further testing can also be performed on the filtering process for the data, as the results (see Table 3) display greater model performance on the unfiltered data. Since the process only made use of second-order Daubechies wavelets, additional wavelet families and thresholds could potentially be implemented instead to determine whether higher AUC scores than the unfiltered datasets can be obtained.

6 Conclusion

Ultimately, much of what was accomplished with the current CNN and LSTM models minimally covers the surface of possible approaches to this multi-label classification task. Both models also have a long way to go to increase their respective macro-averaged AUC scores, achieving 0.711 for CNN and 0.832 for LSTM.

Various improvements can be made, through implementation means such as greater computation resources, more efficient processing times, more complex model architecture, and inter-model coordination. Consequently, many potential improvements when timing and resource allow are available as mentioned in section 5, which includes trying out more models and methods, testing different architecture modification or new parameters, and improving the code base to improve inference speed.

As such, these detection result would be really meaningful for further research, especially in brain and neurology studies, where further activities or feelings of one can be identified.

REFERENCES

[1] Matthew D Luciw, Ewa Jarocka, and Benoni B Edin. Multi-channel EEG recordings during 3, 936 grasp and lift trials with varying weight and friction. *Scientific Data*, 1(1), November 2014.

[2] Shafaq Zia, Ali Nawaz Khan, Mayyda Mukhtar, and Shan E Ali. Human activity recognition using portable EEG sensor and support vector machine. In *2021 International Conference on Engineering and Emerging Technologies (ICEET)*. IEEE, October 2021.

[3] Marianna Koctúrová and Jozef Juhár. A novel approach to EEG speech activity detection with visual stimuli and mobile BCI. *Applied Sciences*, 11(2):674, January 2021.

[4] Jing Huang, Krishnan Padmanabhan, and Oliver M. Collins. The sampling theorem with constant amplitude variable width pulses. *IEEE Transactions on Circuits and Systems I: Regular Papers*, 58(6):1178–1190, June 2011.

[5] Ingrid Daubechies. *Ten Lectures on Wavelets*. Society for Industrial and Applied Mathematics, January 1992.

[6] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, June 2016.

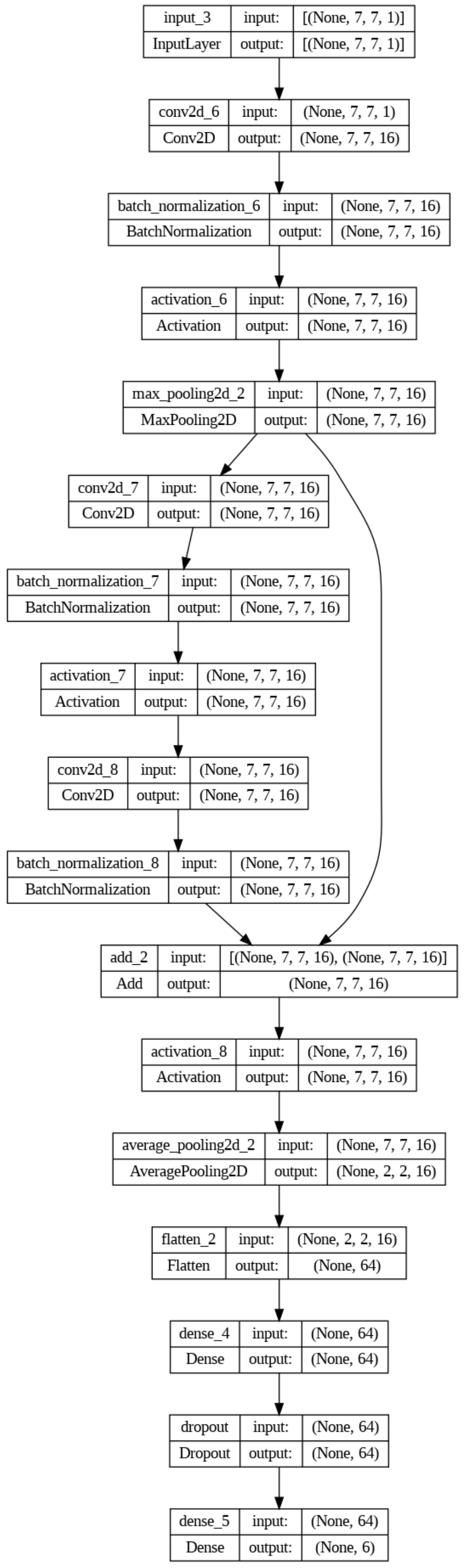
[7] Tong Liu, Tailin Wu, Meiling Wang, Mengyin Fu, Jiapeng Kang, and Haoyuan Zhang. Recurrent neural networks based on lstm for predicting geomagnetic field. In *2018 IEEE International Conference on Aerospace Electronics and Remote Sensing Technology (ICARES)*, pages 1–5, 2018.

[8] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Computation*, 9(8):1735–1780, 1997.

[9] Michael H Ferris, Michael McLaughlin, Samuel Grieggs, Soundararajan Ezekiel, Erik Blasch, Mark Alford, Maria Cornacchia, and Adnan Bubalo. Using ROC curves and AUC to evaluate performance of no-reference image fusion metrics. In *2015 National Aerospace and Electronics Conference (NAECON)*. IEEE, June 2015.

APPENDIX

A Detailed model architecture



input_3 | InputLayer | input: [(None, 7, 7, 1)] | output: [(None, 7, 7, 1)]
conv2d_6 | Conv2D | input: (None, 7, 7, 1) | output: (None, 7, 7, 16)
batch_normalization_6 | BatchNormalization | input: (None, 7, 7, 16) | output: (None, 7, 7, 16)
activation_6 | Activation | input: (None, 7, 7, 16) | output: (None, 7, 7, 16)
max_pooling2d_2 | MaxPooling2D | input: (None, 7, 7, 16) | output: (None, 7, 7, 16)
conv2d_7 | Conv2D | input: (None, 7, 7, 16) | output: (None, 7, 7, 16)
batch_normalization_7 | BatchNormalization | input: (None, 7, 7, 16) | output: (None, 7, 7, 16)
activation_7 | Activation | input: (None, 7, 7, 16) | output: (None, 7, 7, 16)
conv2d_8 | Conv2D | input: (None, 7, 7, 16) | output: (None, 7, 7, 16)
batch_normalization_8 | BatchNormalization | input: (None, 7, 7, 16) | output: (None, 7, 7, 16)
add_2 | Add | input: [(None, 7, 7, 16), (None, 7, 7, 16)] | output: (None, 7, 7, 16)
activation_8 | Activation | input: (None, 7, 7, 16) | output: (None, 7, 7, 16)
average_pooling2d_2 | AveragePooling2D | input: (None, 7, 7, 16) | output: (None, 2, 2, 16)
flatten_2 | Flatten | input: (None, 2, 2, 16) | output: (None, 64)
dense_4 | Dense | input: (None, 64) | output: (None, 64)
dropout | Dropout | input: (None, 64) | output: (None, 64)
dense_5 | Dense | input: (None, 64) | output: (None, 6)

Figure 14: Residual CNN - Detailed Architecture

B Model tuning results

Filter Threshold	$T_{lookback}$	r_{sample}	$r_{non-event}$	Validation AUC
0.8	2	50	0.25	0.521
0.5	2	50	0.25	0.606
0.3	2	50	0.25	0.700
0.15	2	50	0.25	0.760
0	2	50	0.25	**0.831**
0.15	2	100	0.25	0.748
0.15	2	25	0.25	NA
0.15	1	25	0.25	0.722
0.15	0.4	10	0.25	0.658
0.15	0.2	5	0.25	0.616
0.15	4	100	0.25	**0.774**
0.15	4	100	0.10	0.767
0.15	4	100	0.50	0.764
0.15	2	50	0.15	0.767
0.15	2	100	0.40	0.749
0.15	3	50	0.10	**0.784**

Table 4: Hyperparameter tuning results of the LSTM model

Regulariser	Optimiser	Dropout Rate	Learning Rate	Validation AUC
L2	ADAM	0.25	1e-3	**0.927**
L2	NADAM	0.0	1e-3	0.926
L2	ADAM	0.0	1e-3	0.925
L2	NADAM	0.25	1e-3	0.919
L2	ADAM	0.5	1e-3	0.914
L2	NADAM	0.5	1e-3	0.913
L2	ADAM	0.0	5e-4	0.909
L2	ADAM	0.25	5e-4	0.908
L2	NADAM	0.25	5e-4	0.906
L2	NADAM	0.0	5e-4	0.905

Table 5: Hyperparameter tuning results of the 10 best combinations for residual CNN model