Report 1 - Machine Learning in Practice
HMS - Harmful Brain Activity Classification

Jasper Pieterse (s1017897)
Rachel Knol (s1010850)
Stefan Boneschanscher (s1011532)

March 15, 2024

1 Introduction

The goal of this competition is to automate the classification of harmful brain activity using Electroencephalography (EEG) signals. Despite requiring years of training, expert consensus can vary, making automation desirable. Our task involves categorizing EEG and spectrogram data into six categories based on expert-labelled harmful brain activity types. We aim to train a deep learning model to replicate the distribution of expert votes accurately [1].

2 Method

We developed an ensemble model to classify six categories of harmful brain activity: seizure (SZ), generalized periodic discharges (GPD), lateralized periodic discharges (LPD), lateralized rhythmic delta activity (LRDA), generalized rhythmic delta activity (GRDA), and 'other'. The ensemble consists of three different models: EfficientNetB0, a 2D convolutional model trained on spectrogram data; EEGNet, a 1D hybrid model trained directly on EEG data; and WaveNet, a 1D convolutional model trained on EEG data. This report will focus specifically on the improvements made to the EfficientNet and EEGNet models as these are the models we extensively studied and re-implemented, whereas the WaveNet model is excluded from our scope.

2.1 Evaluation Metrics

We evaluated the performance of our models using the KL Divergence between the predicted distribution among the six categories and the expert vote distribution. To ensure robustness, we used 5-fold GroupKFold cross-validation during the training of all models. The KL Divergence on the out-of-fold validation set yielded the Cross-Validation (CV) score. Additionally, the KL Divergence evaluated on the private test set from Kaggle corresponds to the public leaderboard (PB) score.

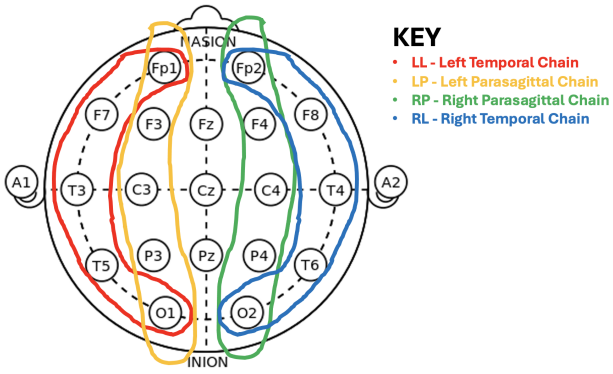

Figure 1: Overview of the 20 EEG channels that record activity. Data processing is based on the temporal chains visualized by the four colours. Image from [2].

2.2 EEGNet

The baseline EEGNet model by Nischay Dhankhar uses a hybrid architecture combining a convolutional pathway and a Gated Recurrent Unit (GRU) pathway. Convolutional layers capture local patterns, while GRU layers model long-term dependencies within the EEG signals. The model processes input EEG data through parallel convolutions with varying kernel sizes to extract temporal features at different scales. A schematic representation of the model and its pathways is shown in Figure 2.

Our main modification, named 'EEGNet Parallel,' processes each temporal chain independently before combining them for classification. This alteration allows the model to handle each channel of data separately, with appropriate downsizing of feature maps and GRU hidden sizes proportional to the number of channels. The model is visualized in Figure 3.

Preprocessing involved filtering the training data to include only one EEG recording per unique ID, quantizing data using μ-law encoding with $\mu = 1$, and applying a low-pass Butterworth filter with a cutoff frequency of 20 Hz. Manual feature extraction involves computing residuals between features of a chain. For example, for the first LP chain, we created custom features `x1 = FP1 - T3` and `x2 = T3 - O1` to represent the residuals along that chain.

Our model was trained using the EEG dataset provided by the HMS Kaggle competition. To optimize performance, we used the KL Divergence Loss combined with a logarithm-softmax transformation to produce raw class scores (logits). We used an ADAM optimizer with a cosine annealing scheduler and an initial learning rate of $\eta = 0.008$. To enhance generalization and prevent overfitting, we used weight decay with $\alpha = 0.01$ and coloured noise applied to the training data with a probability of $p = 0.15$ as regularization techniques.

2.3 EfficientNet

As a starting point, the EfficientNetB0 baseline provided by C Deotte was re-implemented. EfficientNet is a convolutional network model that is designed to be easily scaled [3]. The specific version used is B0, the smallest version of the network. To implement EfficientNet we used the deep-learning library timm. Timm provides pretrained versions of EfficientNet that can be rescaled to eight channels to match the number of spectrograms we have.

For preprocessing, we merged the multiple labels of a spectrogram into a single one for training to prevent bias. It was suggested in a discussion that the test data contains only one labelled event per spectrogram, making the training data more comparable to the test data. We grouped all events with the same EEG identifier, averaged their minimum and maximum offset, and summed the expert votes. Then, we normalized the number of expert votes for each event to ensure a total vote sum of one, accounting for variations in the total number of votes between events.

We used KL divergence with the Adam optimizer for training. We tested three different learning rates $(0.1, 0.01, 0.001)$ Training lasted for 10 epochs, however early stopping based on the average epoch loss was added. We also added gradient scaling with auto-casting to minimise gradient underflow in addition to gradient clipping. We tested both the first and the second versions of EfficientNet. In addition, we tried replacing the final two layers of the model with new layers.

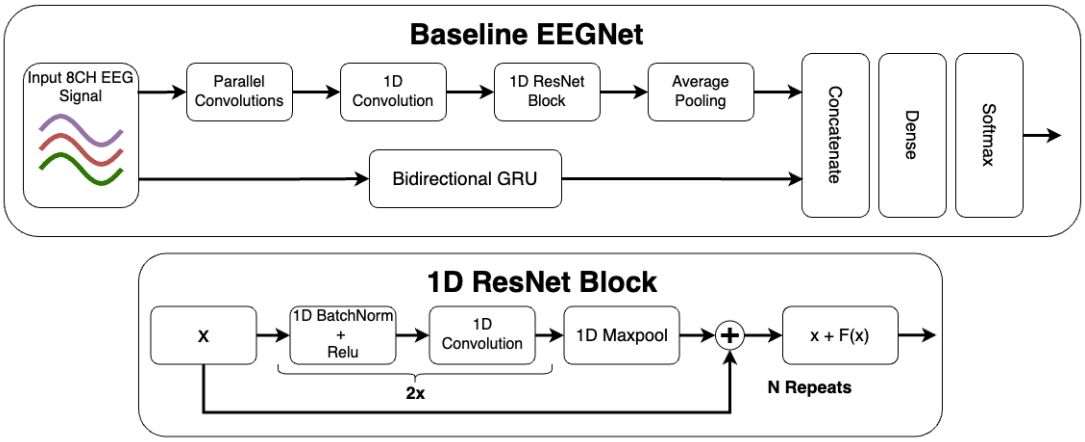

Figure 2: Schematic of the baseline EEGNet implementation.

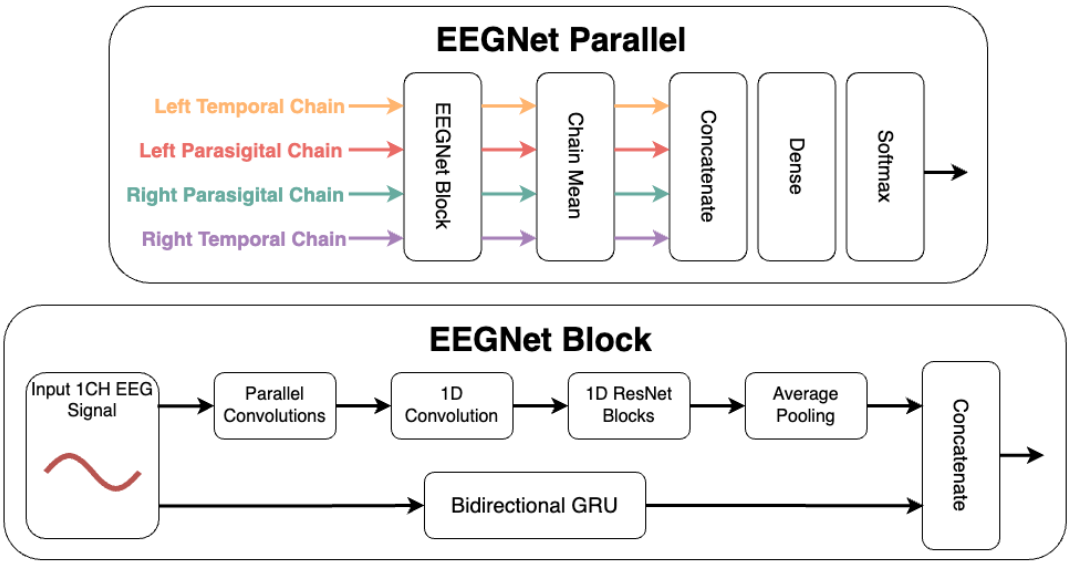

Figure 3: Schematic representation of the Parallel EEGNet model.

2.4 Ensemble Inference model

The test samples are loaded into an ensemble model which consists of the inference models of EfficientNetB0, WaveNet and EEGNet. At the start the necessary data for all models is loaded. These are the spectrograms for the EfficientNet and the EEG signals for the WaveNet and EEGNet. When data is no longer needed it is deleted to prevent memory issues.

3 Results

3.1 EEGNet

We performed 3 main experiments for the EEGNet implementation:

Using raw EEG data In our initial experiment, we evaluated our baseline model's performance on raw 20-channel EEG data, questioning the common practice of using only 8 selected features based on temporal chains. We adjusted the baseline notebook to handle any specified number of features and processed data using float16 precision due to memory constraints. Despite these efforts, both 8-channel and 20-channel experiments yielded a public leaderboard (PB) score of 0.72, notably worse than the baseline PB of 0.51. Additionally, the 20-channel model required double the training time, possibly due to float16 data handling issues. Given these challenges, we decided to focus solely on 8-channel data with float32 precision for further experiments.

Temporal Chain Feature Engineering In the second experiment, our goal was to explore the impact of manual feature engineering on the performance of our model, because the EEGNet implementation lacked any feature engineering. Guided by the established temporal chain formalism, we introduced handcrafted features created by taking residuals between features along the temporal chains, as described in the Methods section. We observed that manual feature engineering led to a slight improvement in both CV ($0.778 \rightarrow 0.764$) and PB scores ($0.52 \rightarrow 0.50$) compared to the same model without feature engineering. This suggests that feature engineering could indeed be beneficial. Thus, we decided to incorporate manual feature engineering into our approach moving forward.

Parallel Chain Processing In our third experiment, we implemented the parallel EEGNet model, aiming to incorporate information about temporal chains directly into the model structure. This was motivated by the success of the previous experiment. Despite this, both cross-validation (CV) scores (from 0.764 to 0.781) and public leaderboard (PB) scores (from 0.52 to 0.52) remained unchanged. The doubled training time of the Parallel model suggests potential issues with parameter downscaling, while incomplete convergence of the validation loss indicates the increased

complexity of Parallel EEGNet. Future investigations may explore lighter model versions and increased epochs for better convergence, although time constraints prevented further exploration in this direction.

3.2 EfficientNet

The self-implemented EfficientNet scored very poorly in testing reaching scores of 2.13 and 2.2. This means that most results from our own EfficientNet provide very little insight. As we were unable to solve the problem in time the EfficientNet model of C. Deotte is implemented in the ensemble model instead of the self-implemented EfficientNet.

Furthermore, we implemented small changes to the starter EfficientNetB0 of C. Deotte. Without adjustments, the EfficientNetB0 achieved a PB 0.43. When pseudocounts of +0.5 were added to the votes, the score significantly decreased to a PB 0.52. In another experiment, the pseudocounnts were set to +0.05 which gave a PB 0.44. Lastly, the epochs were set to 5 instead of 4. Again this resulted in a less favorable score with a PB of 0.45.

3.3 Ensemble inference model

The ensemble model weights the individual scores of all three individual models. The EfficientNet individual score of 0.43, is weighted 0.55 to the final score, the WaveNet with an individual score of 0.50 is weighted by 0.25 and EEGNet is weighted by 0.20. The optimal score of the ensemble model is PB 0.39. The EfficientNet has a relatively high weight towards the final score. This is due to its better individual performance and the expectation that EEGNet and WaveNet perform relatively similarly and therefore should not be overly emphasized.

4 Discussion

Pseudocounts After noticing that there are large differences in the number of votes given to samples, we came up with the idea to add pseudocounts to each categorical vote to smoothen the result. In this matter, a sample with e.g. only 3 votes, all seizure votes, becomes less stable when pseudocounts compared to an event with more votes. Unfortunately, this smoothing did not work and thus was not used.

Memory and disk space Multiple times we ran into the problem of not having enough disk space. The initial ensemble model existed of only a EfficientNetB0 and WaveNet model, and since it seemed to work well a ResNet34d and an EEGNet were added as well. Unfortunately, the addition of these models resulted in submission errors due to disk space. To solve this problem, we deleted some of the memory taken by each model after the model was runn and we optimized the data acquisition. However, mainly the ResNet24d seemed to create problems and in combination with limited time it was decided to remove this model from the ensemble.

EfficientNet The effect of our changes to EfficientNet are hard to judge as the performance was very poor for all iterations of the model. When we analyse the batch loss we see that almost all training progress is achieved at the beginning of the first epoch followed by fluctuation in loss during further training. To us, this suggests that the learning rate was likely too large. However, this is likely not the only issue as earlier versions with lower learning rates also performed poorly.

Further Improvements We had several planned implementations that we didn't manage to execute. One big aspect we overlooked was data cleaning, in particular implementing artifact removal techniques. We also considered experimenting with data augmentations like stretching spectrograms or masking frequency bands, which could have provided valuable insights. Additionally, we haven't explored hyperparameter optimization yet. However, to undertake this task effectively, we might need to employ a more efficient model or utilize a subset of the data.

5 Conclusion

We created an ensemble model by combining EfficientNet, WaveNet, and EEGNet. While this approach gave better results than any individual component, we encountered challenges in achieving scores above the baseline for the individual models. Our EEGNet experimentation revealed that leveraging domain knowledge through temporal chain feature engineering enhanced model performance. However, integrating this knowledge directly into the model structure did not yield the desired outcomes. Similarly it was hard to understand the EfficientNet well enough to be able to improve the baseline model.

6 Source code URLs

- Our EfficientNet model can be found here.
- Our EEGNet model[1] can be found here.
- Our Parallel EEGNet model can be found here
- Our ensemble model can be found here.

7 Author Contributions

Jasper focused primarily on the implementation of EEGNet. Stefan worked on the EfficientNet model. Rachel started working on the EfficientNet model and later focused on the ensemble model.

8 Evaluation of the Process

We underestimated the 'in practice' aspect of the course, which involved setting up the pipeline and getting familiar with it and the Kaggle enviroment workflow. For this, we should have allocated more time. As we started slowly in the first few weeks of the project, we had a substantial workload towards the end of the project. For the next project it would be better to spread the workload more equally throughout the weeks. Since we are now familiar with the Kaggle environment, we expect this to be a lot less of an issue for the following competition.

We initially included EEGNet experiments where we compared `float16` and `float32` implementations, but discovered later these were implemented incorrectly and thus we could not make any substantial claims based on these experiments, so we decided to leave them out. This could have been prevented by using a more thorough and methodical approach. We were thoughtful in deciding om what we wanted to implement next, but not thoughtful in implementing said improvement. For the next competition, it is better to first think through the entire procedure of what needs to be altered and why, instead of starting to just tinker away on the model.

Another example of this is that initial 20CH EEGNet experiment set us on a road that might not have been most optimal. The main thing we overlooked is that using 20CH would take use into the realm of data precision and memory management, of which we did not have any experience with whatsoever. This means that the very first experiment already needed quite some changes to the baseline model, without any intermediate testing. We wasted quite some time on getting Automatic Mixed Precision and DDP Multi-GPU implementations to work, without any fruitful results. In hindsight, we should've thought a bit more about our strenghts and weaknessed before following our curiosity.

Furthermore, it would be nice to start with a smaller model for the next competition. We were aware of the warning to start with a simple model, but we naively assumed a baseline model would be simple. This increases the amount of possible things to do and thus the chance of going along a less effective path. Also, more complicated models take longer to run, making them less efficient for trying out a lot of things and running experiments.

9 Evaluation of the Supervision

We received useful feedback on implementing the parallel EEGNet and the memory issues in the ensemble model. We could have made a more methodical approach about what we planned on improving and how we would go to do it and discuss that plan with the TA. This would have forced us to think our experiments through more and the extra check from the TAs could have helped keep us on the right track. Finding errors early on saves a lot of time downstream.

References

[1] J. Jing, L. Zhen, C. Yang, *et al.*, *Hms - harmful brain activity classification*, 2024. [Online]. Available: https://kaggle.com/competitions/hms-harmful-brain-activity-classification.

[2] C. Deotte, *How to create spectrogram from eeg? solved? boost cv and lb!* [Online]. Available: https://www.kaggle.com/competitions/hms-harmful-brain-activity-classification/discussion/467877.

[3] B. Koonce and B. Koonce, "Efficientnet," *Convolutional neural networks with swift for Tensorflow: image recognition and dataset categorization*, pp. 109–123, 2021.

[1] The accounts "Ringooo", "Myrthe Kouwenhoven" and "marijn_kluen" are shadow accounts used to get more GPU time.