50.039: Theory & Practice of Deep Learning

Project Report

Harmful Brain Activity Classification

Group 6

Abram TAN, 1005057
Benjamin LUO, 1005368
SIM Shang Hong, 1005500

Spring 2024

Contents

1 - Introduction

Electroencephalography (EEG) is a non-invasive medical imaging technique that records the electrical activity of the brain using sensors placed on the scalp [6]. The use of EEG is crucial in the diagnosis and monitoring of neurological disorders such as epilepsy [7] and sleep disorders [2], as well as for research in cognitive neuroscience[3]. The technique captures fluctuations in electrical potential caused by ionic current flows within the neurons of the brain over time, providing physicians highly valuable data to assess brain function and detect potential abnormalities in brain wave patterns, which may lead to improvements in neurocritical care and drug development.

Like many things, the 'no free lunch' idea applies to EEG analysis as well, where the extraction of useful information from raw EEG data is a difficult, slow, laborious, expensive, and ambiguous task, even for trained medical professionals. EEG signals are used in practice by physicians with critically ill patients for seizure detection as well as other brain-damaging activities through the classification of EEG signals into 6 pattern types, namely seizure (SZ), generalized periodic discharges (GPD), lateralized periodic discharges (LPD), lateralized rhythmic delta activity (LRDA), generalized rhythmic delta activity (GRDA), and an 'others' category.

Hence, with the nature of the challenge at hand being classification of signals as well as the presence of historical EEG data and labels, there is an opportunity for deep learning to tackle and potentially improve upon the status quo. With that in mind, this paper aims to tackle the task of classifying EEG data in the form of raw signals as well as their equivalent spectrograms into the 6 pattern types mentioned above, and is based on the Harvard Medical School - Harmful Brain Activity Classification Kaggle competition, leveraging the large dataset provided. This paper covers the whole deep learning pipeline, from exploratory data analysis and data pre-processing to experimentation of various model architectures and parameters, as well as analysis of the results.

2 - Literature Review

The classification of EEG signals and spectrograms using deep learning techniques has been a significant area of research in recent years. The most common deep learning techniques used in the field for the analysis of EEG data generally involve convolutional neural networks (CNNs) [11] and recurrent neural networks (RNNs), including Long Short-Term Memory (LSTM) networks [12], as these model architectures excel in capturing the spatial and temporal dependencies in EEG data. Additionally, transfer learning has also been explored [13], demonstrating potential in leveraging pre-trained networks to overcome the challenge of limited labeled EEG data. Overall, the integration of deep learning with EEG analysis promises significant advancements.

2.1 EEG Data for Deep Learning

EEG data captures the dynamic processes of the brain through electrical signals, offering a temporal resolution unmatched by other neuroimaging techniques. However, its use in deep learning is hampered by several intrinsic and extrinsic limitations. Despite these challenges, EEG's high temporal resolution and non-invasiveness make it a valuable tool.

The spatial resolution of EEG signals is relatively poor, which complicates the precise localization of neural activity and can affect the accuracy of classification and prediction models. In addition to that, EEG signals are highly susceptible to various types of noise, including environmental interference and biological artifacts from muscle movements or eye blinks [5]. Variability in electrode placement, individual differences in skull thickness, and other biological factors contribute to significant variations in EEG recordings, even across similar experimental conditions. These artifacts require robust pre-processing and can lead to increased complexity to achieve good performance.

Another factor limiting the potential of EEG data is class imbalance. Many deep learning applications in EEG suffer from imbalanced datasets, particularly in medical diagnostics where abnormal patterns may be rare [8]. With the challenges and nature of EEG data in mind, there is more context to understand the task in the next steps.

2.2 CNNs

The use of convolutional neural networks (CNNs) is highly rated in the field of computer vision and although EEGs are not visual images with the same kind of spatial encoding, there has been growing use of CNNs in EEG data and spectrogram classification.

EEG signals are complex and highly variable, presenting significant challenges in signal processing and classification. Traditional machine learning techniques often fall short in handling the spatial and temporal dynamics inherent in EEG data. Hence, CNNs, which are known for their prowess in image and signal processing, have been adapted to tackle these challenges, leveraging both raw EEG data and transformed representations like spectrograms. A notable approach involves the use of deep CNNs that apply multiple convolutional and pooling layers to capture hierarchical features from the EEG signals [4].

Transforming EEG data into time-frequency representations such as spectrograms allows CNNs to treat the data similarly to image inputs. This transformation leverages the CNN's strength in handling grid-like data structures. However, one of the recurrent challenges in EEG data analysis is the presence of class imbalance and high levels of noise. Studies have adapted CNN architectures to be more robust against these issues by incorporating techniques such as dropout, batch normalization, and regularization [1].

CNNs have advanced the field of EEG signal classification, offering tools that outperform traditional methods in both accuracy and efficiency, hence have been chosen as a model architecture that this paper will experiment with for the specific application of classifying harmful brain activity.

2.3 EfficientNet

Narrowing down from generic CNNs, EfficientNet is a type of CNN that has recently been explored in the context of EEG data and spectrogram classification. Why EfficientNet? Because EfficientNet models are able to be highly accurate and computationally efficient through balancing scaling of depth, width, and resolution, and offer a promising approach to handle EEG signals, which are difficult to analyze due to their high dimensionality and variability.

EfficientNet's architecture leverages a coefficient to uniformly scale network width, depth, and resolution, which has proven effective in maintaining model efficiency and accuracy when adapted to EEG data[10]. EfficientNet models represent a significant step forward in the classification of EEG data and spectrograms, offering a scalable and efficient solution that meets the demands of both accuracy and computational constraints and is definitely an architecture to explore in this field.

2.4 Contrastive Loss & Siamese Networks

Finally, a novel but highly promising method to approach EEG signal classification is the use of contrastive loss and Siamese networks[9].

Siamese networks involve twin networks which compare paired inputs, making them particularly suitable for tasks that require learning from the similarity or dissimilarity between samples. When combined with contrastive loss, these networks effectively learn to distinguish between different mental states, neurological conditions, or cognitive tasks by emphasizing the differences in learned feature representations and show great potential in this field.

Siamese networks and contrastive loss offer a promising direction for EEG signal classification, providing robustness against noise and variability inherent in EEG data, and is also explored in this paper.

3 - Exploratory Data Analysis

3.1 Dataset Overview

The dataset used for this paper is the Harvard Medical School - Harmful Brain Activity Classification dataset, which consists of two key types of files. Firstly, it contains a train.csv file which has metadata to generate a dataset from raw EEG and spectrogram data. The need for the metadata file is because the annotators reviewed 50 second long EEG samples plus matched spectrograms covering 10 a minute window with many of these samples overlapping. The second type of files are the raw EEG and equivalent spectrogram data, both of which are in .parquet formats. With those files, a dataset can be extracted with the original subsets that the annotators used. Another point to note is that there are also test.csv and raw test data, but they are not used in this paper as they are not provided publicly, as the test data is used for competition evaluation and cannot be known to participants. Hence, in this paper, the train.csv and train raw data is split into training, validation, and test sets. The following are descriptions of the relevant raw EEG and spectrogram data:

1. train_eegs
   EEG data from one or more overlapping samples. Use the metadata in train.csv to select specific annotated subsets. The column names are the names of the individual electrode locations for EEG leads, with one exception. The EKG column is for an electrocardiogram lead that records data from the heart. All of the EEG data (for both train and test) was collected at a frequency of 200 samples per second.

2. train_spectrograms
   Spectrograms assembled EEG data. Use the metadata in train.csv to select specific annotated subsets. The column names indicate the frequency in hertz and the recording regions of the EEG electrodes. The latter are abbreviated as LL = left lateral; RL = right lateral; LP = left parasagittal; RP = right parasagittal.

3.2 Exploring the train.csv Metadata File

Taking a look at the metadata file, there are 106,800 datapoints with no missing values, which can later be separated into train, validation and test datasets. The 106,800 datapoints seen in train.csv are not the actual EEG or spectrogram data, but rather contains labels (seizure_vote, lpd_vote, gpd_vote, lrda_vote, grda_vote, other_vote) and also

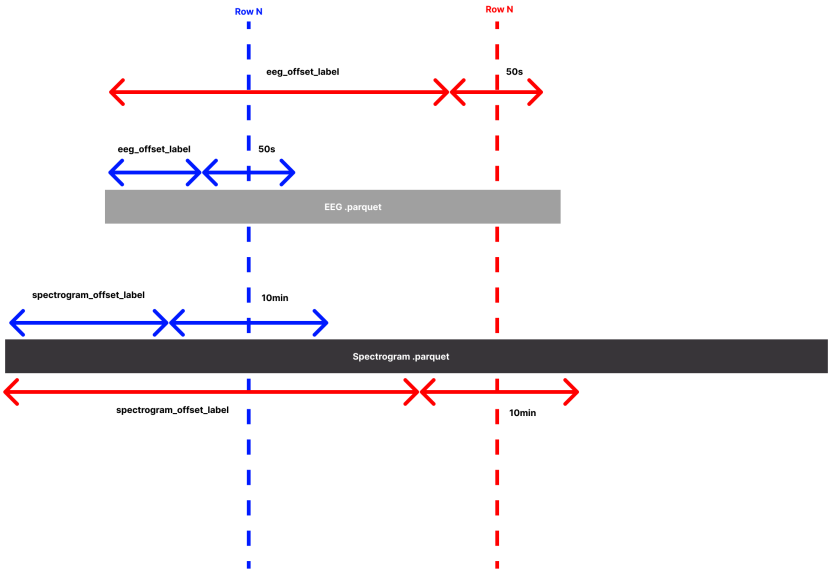

FIGURE 3.1: Relationships between .parquet files and metadata.

provides the information (eeg_label_offset_seconds, spectrogram_label_offset_seconds) needed to extract the portions of the raw data (e.g. EEG and spectrograms sub-samples) that correspond to the labels. Using this information, a dataset with the corresponding EEG, spectrogram data and labels was generated. The following is a list of headers in the train.csv file:

1. eeg_id
   A unique identifier for the entire EEG recording.

2. eeg_sub_id
   An ID for the specific 50 second long sub-sample this row's labels apply to.

3. eeg_label_offset_seconds
   The time between the beginning of the consolidated EEG and this sub-sample.

4. spectrogram_id
   A unique identifier for the entire EEG recording.

5. spectrogram_sub_id
   An ID for the specific 10 minute sub-sample this row's labels apply to.

6. spectogram_label_offset_seconds
   The time between the beginning of the consolidated spectrogram and this sub-sample.

7. label_id
   An ID for this set of labels.
8. patient_id
   An ID for the patient who donated the data.
9. expert_consensus
   The consensus annotator label.
10. (seizure/lpd/gpd/lrda/grda/other)_vote

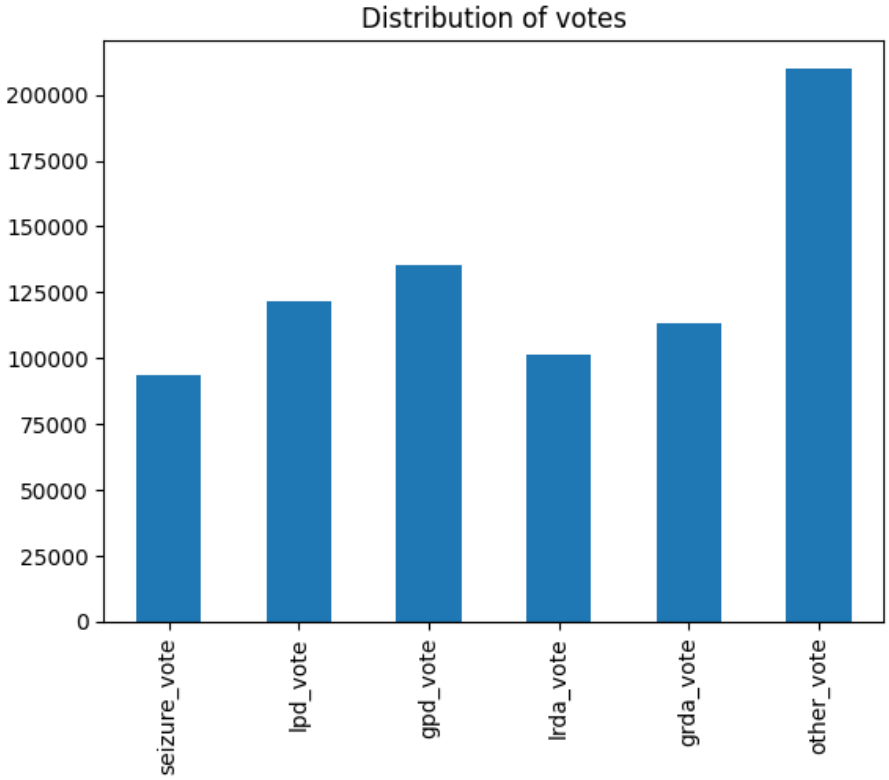

FIGURE 3.2: Frequency distribution of votes.

3.3 Labels

In terms of labels, based on the information on maximum and minimum values provided by the df.describe() function, it can be seen that they range from 0 to a maximum of 25, hence, doing some normalization on the labels might potentially improve training. Taking a look at the bar plot of the frequency distribution of the categorical vote features (3.2), it can be seen that the distribution of datapoints within each category is not particularly skewed in any direction except for the "other_vote" column which may or may not be alright to use for training, validation and testing without modification, but removing some datapoints to ensure an even distribution (prevent class imbalance)

across categories may prove useful. In this case, since the vote features are the features used in the KLDivergence loss metric (as required by the competition) that determines model performance, balancing amongst them makes more sense then balancing the classes around the "expert_consensus" labels.

3.4 EEG and Spectrogram Samples

As seen in sample plots for EEGs (3.3) and spectrograms (3.4), there are missing values present as shown by the white/empty portions of the plots. Hence, analysis of the NaN values present in the data is important and should be dealth with. In addition to that, the range of values present in the data varies a lot for both the EEGs and spectrograms, which is not ideal for training, aka should be normalized before subsequent steps.

3.5 NaN and EEG + spectrogram range check

The following are the number of files containing NaNs and the widely varying EEG and spectrogram values obtained from the .parquet files in the dataset:

- NaN EEGs: 3542
- NaN Spectrograms: 7743
- Minimum EEG value: -1408502.25
- Maximum EEG value: 1942108.125
- Minmum spectrogram value: 0.0
- Maximum spectrogram value: 120858360.0

Digging further into the extent of missing values and value ranges of the dataset, it can be seen that there are 3542 EEGs with NaNs and 7743 spectrograms with NaNs. Three potential ways to tackle this would be to replace the NaNs with zeros, replace the NaNs with interpolated values, or to simply drop the entire datapoint. While dropping 3542 EEGs and 7743 spectrograms may seem to be significant, it only represents around 3.3% and 7.3% of EEGs and spectrograms, which make it a plausible technique to utilize. The following shows the number of samples left if NaN datapoints are dropped:

1. Original number of datapoints: 106800
2. Number of datapoints after dropping NaN EEGs and spectrograms: 95971

Through the exploratory data analysis conducted above, the recommendations are implemented in the next step, data pre-processing.

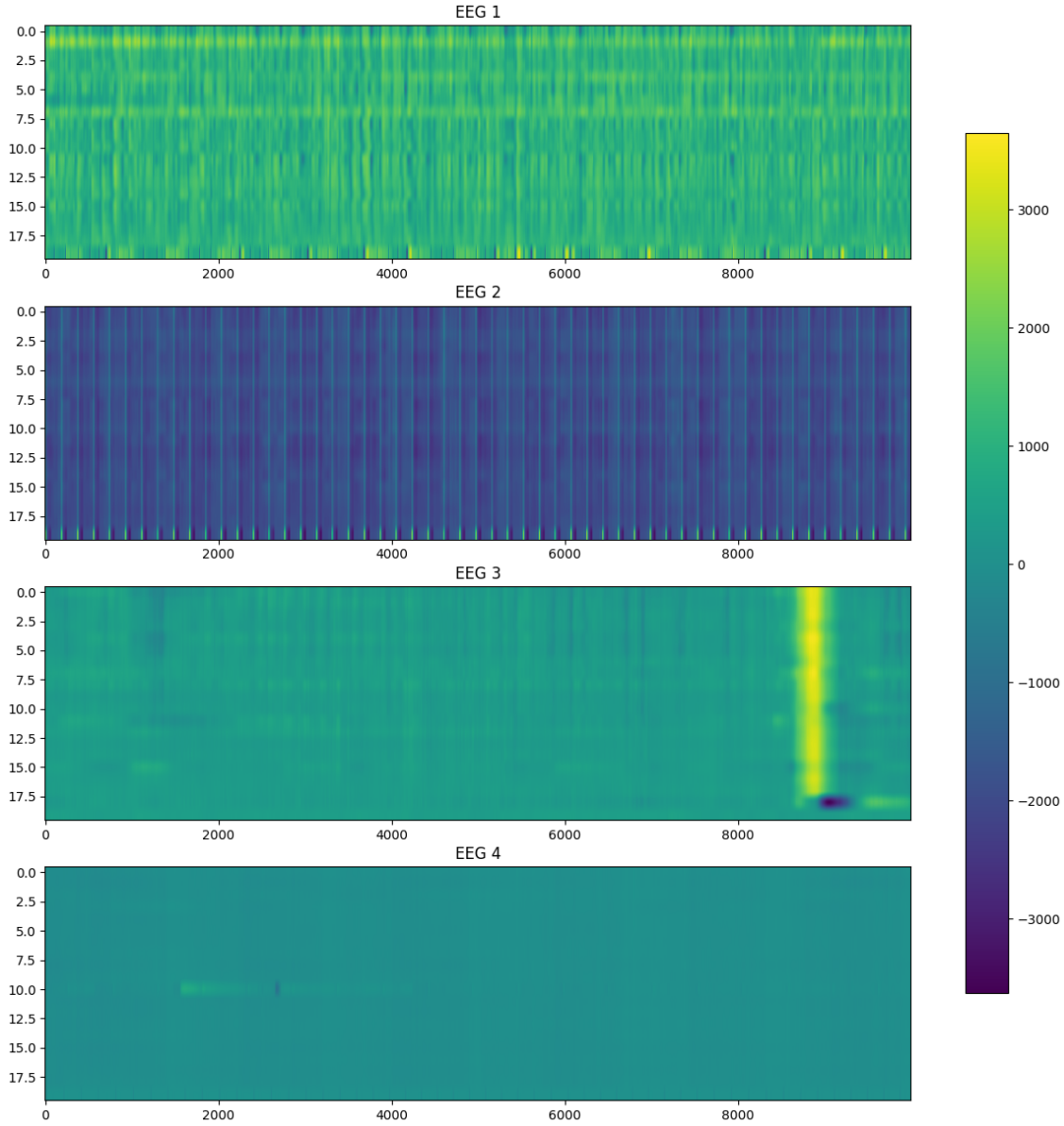

FIGURE 3.3: Visualization of 4 random EEG samples

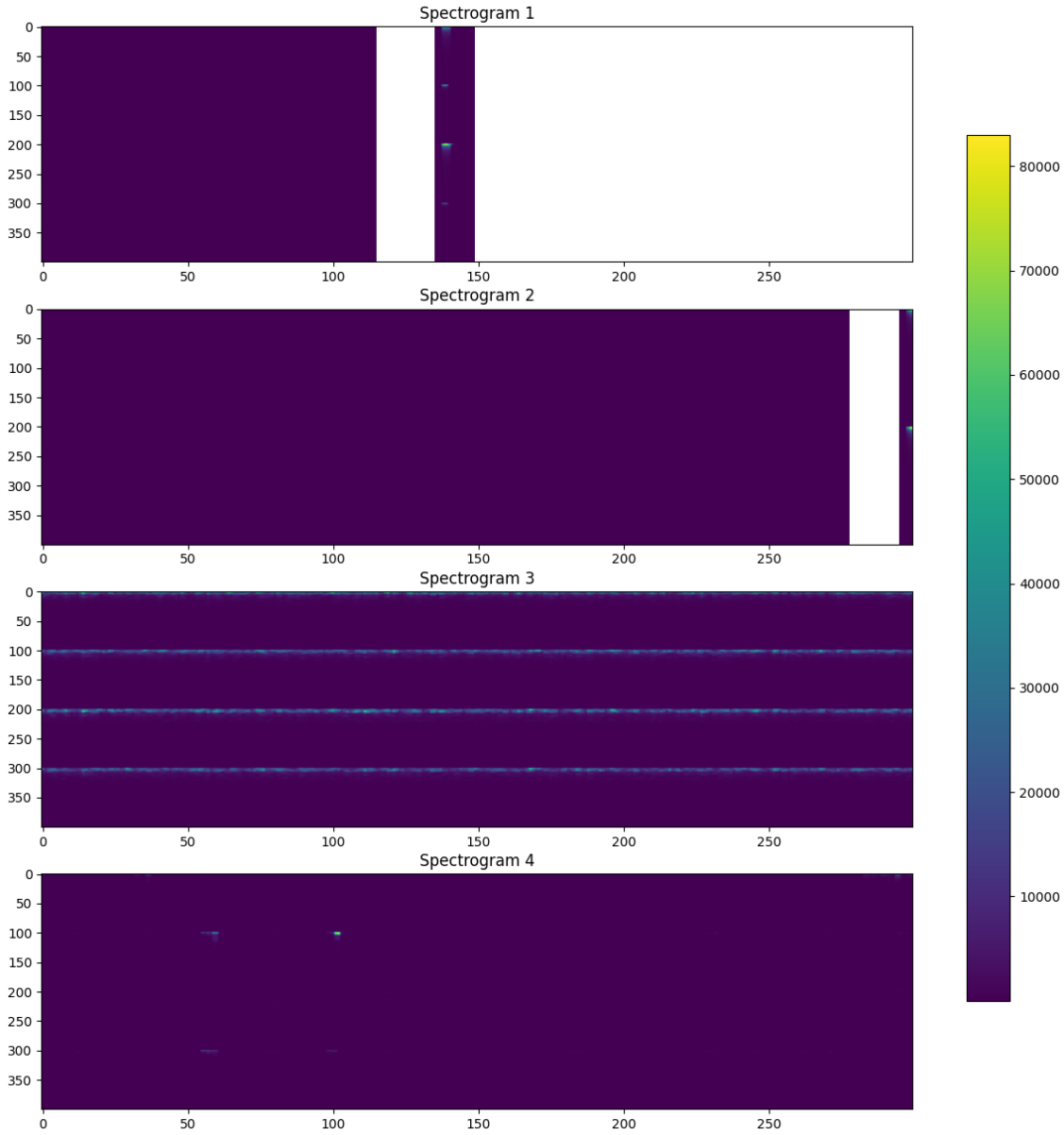

FIGURE 3.4: Visualization of 4 random spectrogram samples, showing missing portions in the data.

3.6 Data Pre-processing

From the original dataset of 106,800 datapoints, the first step in the pre-processing pipeline was to halve the dataset, resulting in 53,400 datapoints. The rationale behind this was not because doing so would improve performance in any way but rather for practicality due to the lack of compute, as using the full dataset takes up around 32GB of RAM.

3.6.1 Data Split

As mentioned above, the competition's test data is not publicly available and hence, the training data needs to be split. The dataset was split by a 90/10 ratio into train and test sets, resulting in 48,060 training datapoints and 5,340 test datapoints, each comprising of EEG, spectrogram and label data. The train dataset was further split by group K-folds into 5 uniform groups. One group will be chosen randomly for validation while the rest will be used for training every epoch. The partially processed data was then saved to local storage as 2 .pkl files, one for EEG and the other for spectrogram data, for faster access in subsequent runs since accessing 1 file is faster than accessing many.

3.6.2 Normalization

Normalization was conducted on several fronts, the labels being the first to be normalized. The 6 labels per datapoint were normalized by dividing each label by the sum of the other 5 labels, resulting in a probability distribution amongst the 6 labels. Normalization was also done on the EEG and spectrogram data through the use of z-score normalization, whereby the mean is subtracted from each data point and divided by the standard deviation for each feature, as established by this equation: $z = \frac{x-\mu}{\sigma}$. The decision to use z-score normalization was due to several factors, including the known effect of boosting algorithm performance, faster convergence, and especially that z-score normalization has no defined range. Unlike min-max scaling which scales data to a specific range, z-score normalization does not bind values to a fixed range, allowing it to handle outliers better, which is important as the range of EEG and spectrogram values are extremely large (-1,408,502.25 to 1,942,108.125 for EEGs, 0.0 to 120,858,360.0 for spectrograms).

3.6.3 Removal of NaNs

Both EEGs and spectrograms had NaNs but were replaced with 0.

3.6.4 Clipping

Clipping was done to spectrogram data to limit it within the range of e^{-4} to e^{8} as the spectrogram data is highly dynamic and of large range, and when being normalized, may cause smaller values to diminish.

3.6.5 Data Augmentation

Simple data augmentation in the form of random horizontal flips was also used to further create a robust dataset.

3.6.6 Sub-sampling

Sub-sampling was done on the raw data according to the metadata train.csv file. As shown in Fig. 3.1, samples of EEG signals and their spectrograms have to be extracted from a longer signal using the correct offsets to ensure that the model learns to make predictions based on the correct portions of the signals.

4 - Methodology

4.1 Models

4.1.1 Naive CNN

The first model architecture used was a naive CNN. The idea behind the naive CNN was to serve as a good starting point, a baseline, for comparisons with other models as well as a simple way to experiment with and test other code. Hence, the naive CNN had to be easy and quick to implement, and also be able to perform somewhat adequately for the task of harmful brain activity classification. Knowing that while EEG is valued for its high temporal resolution, and despite having relatively low spatial resolution, the temporal information that EEGs possess can be transformed into a spectrogram, which would enable techniques that work well with spatial information, like CNNs, to work well with the temporal nature of EEG data. Thus, an arbitrary naive CNN was designed, attempting to use common CNN-related techniques, while also keeping the number of trainable parameters small so that it can be trained quickly and used as a benchmark. The following table describes the architecture of the naive CNN:

Layer	**Configuration**
conv1	Conv2d(3, 32, kernel_size=(3, 3), stride=(1, 1))
batch_norm1	BatchNorm2d(32, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
pool1	MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
conv2	Conv2d(32, 32, kernel_size=(3, 3), stride=(1, 1))
batch_norm2	BatchNorm2d(32, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
pool2	MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
avg_pool	AvgPool2d(kernel_size=5, stride=5, padding=0)
flatten	Flatten(start_dim=1, end_dim=-1)
fc1	Linear(in_features=8512, out_features=512, bias=True)
drop1	Dropout(p=0.1, inplace=False)
fc2	Linear(in_features=512, out_features=6, bias=True)

TABLE 4.1: NaiveCNN Model Architecture

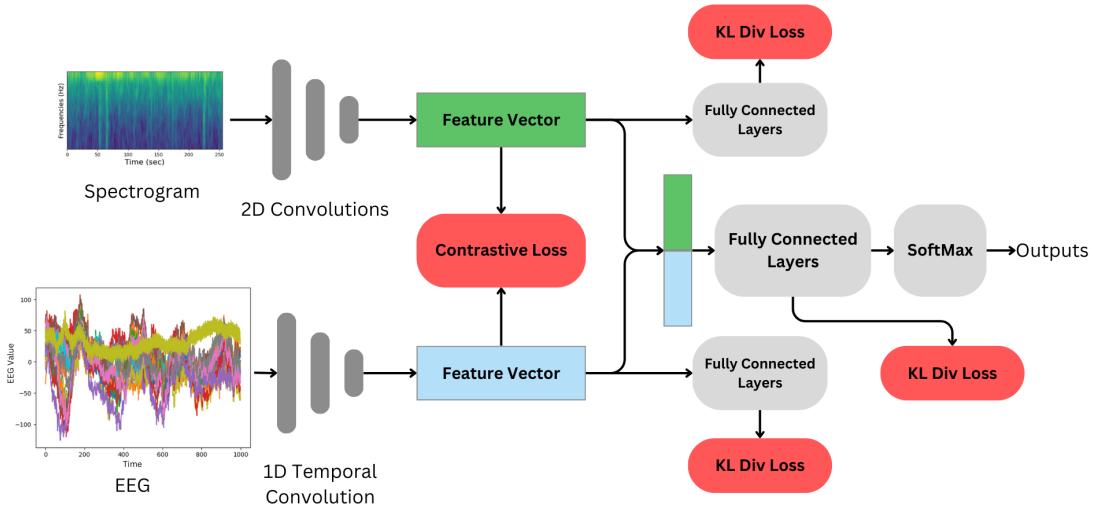

FIGURE 4.1: Contrastive Model architecture

4.1.2 EfficientNet

More State-of-the-Art CNNs such as EfficientNet-b0 were trained on the EEG spectrograms. The mono-channel spectrograms were expanded into three channels of the same values to follow the image architecture of the CNNs as closely as possible and to allow for pretrained weights to be used to speed up training (more details in section 4.2.1).

4.1.3 Contrastive Loss

To help find the correlation between the spectrogram and its EEG signal, we introduce additional layers between the output of the CNN backbones extracting features from the spectrogram and EEG signals as shown in Fig. 4.1. The flatten-ed feature vectors produced by the CNN backbones are compared using a contrastive loss and concatenated to be passed into another set of fully connected layers to output the final prediction classes.

The feature vectors of the two CNN backbones are also forwarded into their own fully connected layers to output predictions based on the spectrogram and EEG signals separately. All three prediction classes are compared with the ground truth using the Kullback-Leibler divergence loss to ensure that the features extracted by the CNN backbones are relevant to the output and the model learns to produce the right output classes.

The contrastive loss we used in our network is the negative of cosine similarity (inner product between two vectors) which is intuitive in comparing the similarity in directions of two vectors in a feature space. A high cosine similarity will result in a low contrastive loss, which we hypothesise is a result of the CNN backbones learning

to extract similar features from the spectrogram and EEG signals which could be more common features between sets of spectrograms and EEG signals.

4.2 Evaluation

4.2.1 Loss Function

The loss function chosen to evaluate model performance in the HMS Competition is the Kullback-Leibler divergence loss, given by the following between two distributions P and Q:

$$KL(P||Q) = \sum_{x=\text{class}} P(x) \log \frac{P(x)}{Q(x)}$$

This measures the difference between the predicted distribution of output classes and the ground truth distribution. Hence, the lower the KL divergence, the more confident we are that the model can approximate the ground truth function relating EEG signals to their harmful brain activity class.

Other common test metrics used in multi-class classification were implemented to test the performance of our models such as:

Accuracy

We opted for macro accuracy to take into account any class imbalances that may occur in the dataset (as shown in Fig. 3.2).

$$\text{Accuracy} = \frac{1}{N} \sum_{\text{class}} \frac{\text{TP} + \text{TN}}{\text{Total count for class}}$$

where N is the total number of classes.

F1-Score

$$\text{Accuracy} = \frac{1}{N} \sum_{\text{Class}} \frac{\text{TP}}{\text{TP} + \frac{1}{2}(\text{TP} + \text{TN})}$$

where TP and TN stand for the True Positives and True Negatives of predictions of each class.

4.3 Training and Validation

As our dataset size has to be limited as mentioned in section 3.5 and is only moderate compared to similar computer vision tasks, we opted to fine-tune on backbones pre-trained on image classification like IMAGENET1K.

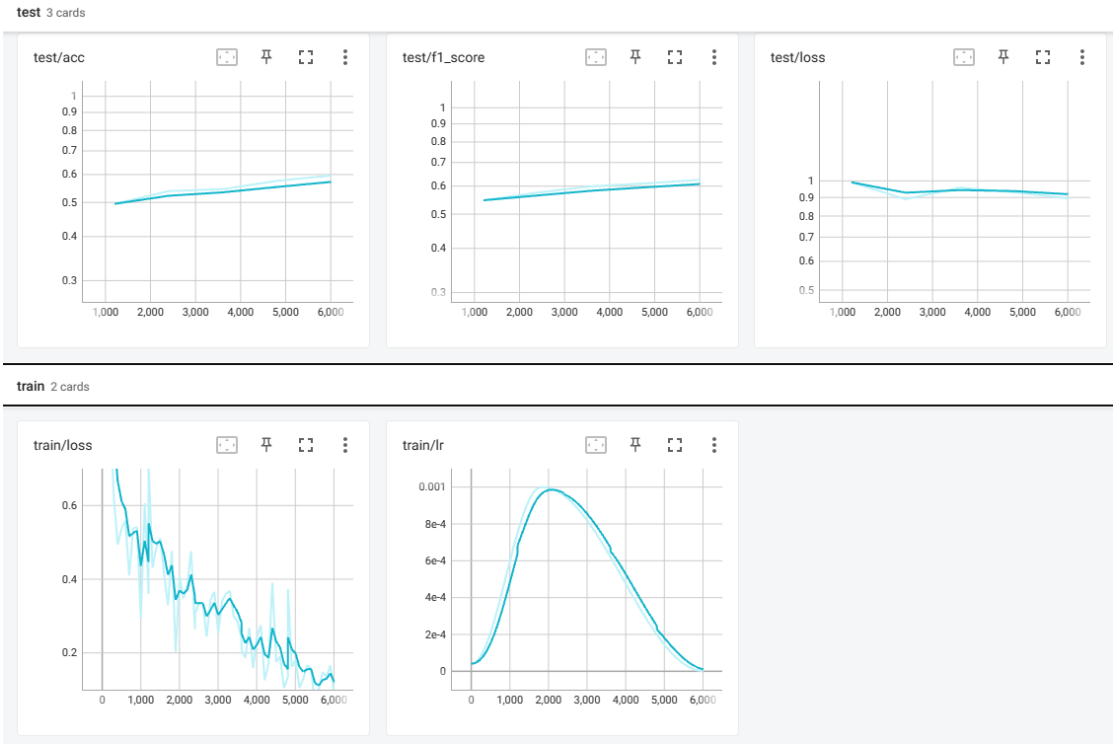

FIGURE 4.2: Training plots for EfficientNet-b0 training on only spectrogram data of each eeg signal

Each model is trained for 5 epochs using a batch size of 32 with the Adam optimiser and oneCycleLR scheduler. We adopt the K-Folds training strategy where 1 random group of training data is loaded as the validation dataset while the rest are used for training. Models are evaluated based on their validation loss every 500 steps to monitor if they are over-fitting on the training dataset.

4.3.1 Optimiser

The Adam optimiser is chosen for its versatility and completeness. A weight decay coefficient of 1e-4

4.3.2 Learning Rate Scheduler

We chose the 1 Cycle LR policy which increases the learning rate from near zero initially to a maximum value before gradually decreasing to zero again, as can be seen in our training graphs (Fig. 4.2).

5 - Results & Discussion

5.1 Results

Testing was done at the end of each epoch and the test metrics at the end of 5 epochs are shown. Judging from the validation loss in the training plots shown in Fig. 4.2, none of the models are overfitting the training data at the end of 5 epochs.

Architecture	Model	Test Accuracy	Test F1-Score	Test Loss
CNN w/ spec	Naive	0.4902	0.5549	0.9662
	EfficientNet-b0	0.5962	0.6252	0.8949
	EfficientNet_V2s	0.5972	0.6334	0.871
	ConvNext Tiny	**0.6171**	**0.6441**	0.9095
Dual CNNs	Contrastive CNNs	0.6128	0.6411	**0.8432**

TABLE 5.1: Test Metrics at the end of 5 epochs, test loss is the same as that in training - KL Divergence Loss.

5.1.1 Evaluation

As can be seen in table 5.1, the contrastive CNN using both the spectrogram and EEG data outperforms models without using EEG data in terms of test loss.

5.2 Limitations

1. Limited computational resources to train on the entire train dataset
2. Limited time to train models, from the training graphs in Fig. 4.2, it can be seen that the training loss has not fully converged. Hence, more performance could have been obtained with more epochs of training.

5.3 Future Work

To improve the performance and generalisability of the models, various data augmentation techniques can be explored:

- Adding random noise to the spectrogram and/or EEG data
- Adding random dropout to the data to remove over-reliance on any part of the signals.

- Signal filtering to remove noise and isolate certain frequency bands of EEG signals.
- Adding synthetic data generated from GANs and Autoencoders to add regularisation to the dataset.

Other State-of-the-Art computer vision architectures like vision transformers could be trained and evaluated as well.

6 - Conclusion

In this project, we took on the challenge of analysing EEG signals and their spectrograms to detect evidence of Harmful Brain Activities. It is interesting to see that CNNs can model not just spatial relationships as in computer vision tasks like image classification, but also temporal relationships present in time-based signals like EEG.

From the results of our models, we can also observe that while no additional information is added from signal processing as in the case of generating a spectrogram of the EEG signal, it can emphasise useful aspects of the data and expose hidden relationships between data features that would otherwise be difficult to learn from scratch.

References

[1] F. Lotte, M. Congedo, A. Lécuyer, F. Lamarche, and B. Arnaldi, "A review of classification algorithms for eeg-based brain–computer interfaces," *Journal of Neural Engineering*, vol. 4, no. 2, R1, Jan. 2007. DOI: 10.1088/1741-2560/4/2/R01. [Online]. Available: https://dx.doi.org/10.1088/1741-2560/4/2/R01.

[2] I. G. Campbell, "Eeg recording and analysis for sleep research," *Current Protocols in Neuroscience*, vol. Chapter 10, Unit10.2, 2009. DOI: 10.1002/0471142301.ns1002s49.

[3] A. Dietrich and R. Kanso, "A review of eeg, erp, and neuroimaging studies of creativity and insight," *Psychological Bulletin*, vol. 136, no. 5, pp. 822–848, 2010. DOI: 10.1037/a0019749. [Online]. Available: https://doi.org/10.1037/a0019749.

[4] R. T. Schirrmeister, J. T. Springenberg, L. D. J. Fiederer, *et al.*, "Deep learning with convolutional neural networks for eeg decoding and visualization," *Human Brain Mapping*, vol. 38, no. 11, pp. 5391–5420, 2017. DOI: https://doi.org/10.1002/hbm.23730. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/hbm.23730. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/hbm.23730.

[5] C. Q. Lai, H. Ibrahim, M. Z. Abdullah, J. M. Abdullah, S. A. Suandi, and A. Azman, "Artifacts and noise removal for electroencephalogram (eeg): A literature review," pp. 326–332, 2018. DOI: 10.1109/ISCAIE.2018.8405493.

[6] "What happens during an electroencephalogram (eeg)?," 2018, Accessed: 2023-04-14.

[7] H. Chen and M. Z. Koubeissi, "Electroencephalography in epilepsy evaluation," *Continuum (Minneapolis, Minn.)*, vol. 25, no. 2, pp. 431–453, 2019. DOI: 10.1212/CON.0000000000000705.

[8] J. Fan, C. Sun, C. Chen, *et al.*, "Eeg data augmentation: Towards class imbalance problem in sleep staging tasks," *Journal of Neural Engineering*, vol. 17, no. 5, p. 056 017, Oct. 2020. DOI: 10.1088/1741-2552/abb5be. [Online]. Available: https://dx.doi.org/10.1088/1741-2552/abb5be.

[9] S. Shahtalebi, A. Asif, and A. Mohammadi, "Siamese neural networks for eeg-based brain-computer interfaces," pp. 442–446, 2020. DOI: 10.1109/EMBC44109.2020.9176001.

[10] Z. Xiong, H. Wang, L. Zhang, *et al.*, "A study on seizure detection of eeg signals represented in 2d," *Sensors*, vol. 21, no. 15, 2021, ISSN: 1424-8220. DOI: 10.3390/s21155145. [Online]. Available: https://www.mdpi.com/1424-8220/21/15/5145.

[11] T. Mari, J. Henderson, M. Maden, S. Nevitt, R. Duarte, and N. Fallon, "Systematic review of the effectiveness of machine learning algorithms for classifying pain intensity, phenotype or treatment outcomes using electroencephalogram data," *The Journal of Pain*, vol. 23, no. 3, pp. 349–369, 2022. DOI: 10.1016/j.jpain.2021.07.011. [Online]. Available: https://doi.org/10.1016/j.jpain.2021.07.011.

[12] C. Wang, X. Wang, X. Jing, *et al.*, "Towards high-accuracy classifying attention-deficit/hyperactivity disorders using cnn-lstm model," *Journal of Neural Engineering*, vol. 19, no. 4, 10.1088/1741–2552/ac7f5d, 2022. DOI: 10.1088/1741-2552/ac7f5d. [Online]. Available: https://doi.org/10.1088/1741-2552/ac7f5d.

[13] S. Sidharth, A. A. Samuel, R. H, J. T. Panachakel, and S. Parveen K, "Emotion detection from eeg using transfer learning," Jul. 2023. DOI: 10.1109/embc40787.2023.10340389. [Online]. Available: http://dx.doi.org/10.1109/EMBC40787.2023.10340389.