Project 2 Report Paper: Harmful Brain activity Classification with MobileNetV3 using KerasCV

Joseph Martínez
Old Dominion University, jmart130@odu.edu

***Abstract* - The Kaggle challenge HMS - Harmful Brain Activity Classification focuses on automating the analysis of electroencephalography (EEG) signals to detect seizures and harmful brain activity in critically ill patients. Hosted by Sunstella Foundation, the competition aims to enhance EEG pattern classification accuracy, crucial for neurocritical care and drug development. Currently, manual EEG analysis is time-consuming and prone to errors. Participants are tasked with developing models to classify EEG patterns, including seizures and various discharges. Annotated EEG segments range from well-agreed idealized patterns to edge cases with disagreements among experts. Successful algorithms could expedite accurate treatments, benefiting neurology and pharmaceutical research.**

Our study focuses on the detection and classification of EEG patterns associated with seizures and related conditions. We adapted the KerasCV starter notebook, originally employing the EfficientNetV2 architecture, and made a significant modification by substituting the model with MobileNetV3. Our experiments aimed to explore the comparative performance of the original and modified model architectures, with a particular emphasis on model generalization and overfitting. The results of our study provide valuable insights into the challenges and strategies involved in developing robust deep learning models for EEG-based seizure detection, highlighting the importance of model selection, regularization techniques, and hyperparameter tuning. Our findings contribute to advancing the state-of-the-art in automated EEG analysis.

Index Terms – EEG signals, seizure detection, pattern classification, neurocritical care.

PROBLEM DEFINITION

The HMS - Harmful Brain Activity Classification Kaggle competition [1] addresses the pressing need for automating the detection and classification of seizures and other harmful brain activities in critically ill patients. Presently, electroencephalography (EEG) monitoring in neurocritical care heavily relies on manual interpretation by specialized neurologists. This process is not only time-consuming but also prone to errors, leading to delays in treatment and potential misdiagnoses. Moreover, the labor-intensive nature of manual EEG analysis poses significant challenges, including high costs, fatigue-related errors, and inconsistencies between different reviewers, even when they are experts in the field.

One of the primary challenges in EEG analysis is the variability in EEG patterns, which can range from well-defined seizures to subtle abnormalities. Additionally, there can be disagreements among experts regarding the classification of EEG segments, further complicating the diagnostic process. This variability and disagreement highlight the need for automated methods capable of accurately classifying EEG patterns in real-time.

The primary objective of this competition is to develop machine learning models that can accurately classify EEG segments into specific patterns, including seizures, generalized periodic discharges, lateralized periodic discharges, and other relevant categories.

By leveraging EEG data recorded from critically ill patients and advancing state-of-the-art machine learning techniques, participants are tasked with creating models that will not only expedite the diagnostic process but also improve the accuracy and reliability of EEG analysis, ultimately leading to better patient outcomes in neurocritical care settings.

METHODS

We adapted the code *HMS-HBAC: KerasCV Starter Notebook*[1] from by the Keras team, utilizing KerasCV[2] [2] and Keras[3] (JAX as preferred backend), changing the model but keeping the preprocessing, training process, and evaluation the same. Their solution presented for the HMS - Harmful Brain Activity Classification Kaggle competition employs a Deep Learning model, specifically EfficientNetV2 with spectrograms of the EEG data to classify the patterns.

KerasCV is a library that extends the capabilities of Keras, a popular deep learning framework. It provides additional

[1] https://www.kaggle.com/code/awsaf49/hms-hbac-kerascv-starter-notebook
[2] https://github.com/keras-team/keras-cv
[3] https://github.com/keras-team/keras

functionalities and pre-implemented models for various computer vision tasks, making it easier for developers to build and experiment with deep learning models for image-related tasks [2]. The original network used the preset for EfficientNetV2, but KerasCV has a model preset library[4] of over 20 models.
Our submission used the model MobileNetV3 instead since it generally achieves good performance on image classification tasks while maintaining efficiency. The model has a lightweight design optimized for mobile and edge devices, making it perfect for use outside of servers with big performance. With this change, we aim that with MobileNetV3, users can try and test the resulting model in mobile and smaller devices.
This section outlines the methodology, including data preprocessing, model architecture, training process, and evaluation.

Data Preprocessing:

The data preprocessing pipeline for the classification task involves several steps to prepare the spectrogram data for model training:

1. **Convert .parquet to .npy:** The original spectrogram data is stored in .parquet files. To facilitate easier data loading and processing, the spectrogram data is converted to .npy format. This process involves reading the .parquet files, filling any NaN values with zeros, transposing the data to rearrange the dimensions from (Time, Frequency) to (Frequency, Time), and saving the data as .npy files.
2. **Data Loading**: The DataLoader reads the preprocessed .npy spectrogram files and extracts labeled subsamples using specified offset values. These offset values determine the temporal segment of the spectrogram to be used for training. The DataLoader also converts the spectrogram data into log spectrograms to enhance feature representation.
3. **Data Augmentation:** Data augmentation techniques are applied to enhance the model's ability to generalize and improve robustness. Augmentation techniques such as MixUp, frequency masking, and time masking are employed to introduce variations in the spectrogram data. MixUp interpolates between two examples to create new training samples, while frequency and time masking randomly mask out portions of the spectrogram data.
4. **Data Split:** The data is divided into training and validation sets using a Stratified Group K-Fold cross-validation strategy. This ensures that each fold contains a balanced distribution of class labels and avoids potential data leakage issues by preventing overlap of patients between the training and validation sets.
5. **Build Train & Valid Dataset:** Only the first sample for each spectrogram_id is used to keep the dataset size manageable. The training and validation datasets are constructed using the DataLoader, specifying the paths to the spectrogram files, offsets for labeled subsamples, class labels, batch size, and other parameters. The datasets are then prepared for model training and validation.

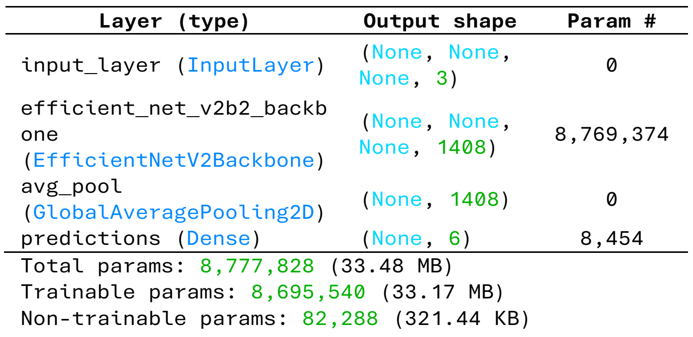
[4] https://keras.io/api/keras_cv/models/

Model Architecture:

The model architecture utilized the original notebook employs the EfficientNetV2 B2, a convolutional neural network (CNN) model from KerasCV's collection of pretrained models. EfficientNetV2 is known for its effectiveness in various computer vision tasks due to its balanced architecture, which efficiently scales the network's depth, width, and resolution. The architecture consists of several key components (see Figure 2):

- Input Layer: Accepts input images with three color channels (RGB).
- EfficientNetV2 B2 Backbone: The backbone of the model, responsible for extracting features from input images. It consists of multiple layers that perform convolutional operations to extract hierarchical representations of input images.
- Global Average Pooling 2D Layer (avg_pool): A pooling layer that reduces the spatial dimensions of the feature maps while retaining important information. It computes the average value of each feature map, resulting in a fixed-size output.
- Predictions Layer (predictions): The output layer of the model, which consists of a dense layer with six units, corresponding to the six classes of EEG patterns to be classified.

The total number of parameters in the model is 8,777,828, with 8,695,540 trainable parameters and 82,288 non-trainable parameters. This architecture has been pretrained on the ImageNet dataset, making it capable of extracting meaningful features from input EEG signals.



```
         Layer (type)              Output shape       Param #

input_layer (InputLayer)         (None, None,             0
                                 None, 3)
efficient_net_v2b2_backb         (None, None,      8,769,374
one                              None, 1408)
(EfficientNetV2Backbone)
avg_pool                         (None, 1408)             0
(GlobalAveragePooling2D)
predictions (Dense)              (None, 6)            8,454

Total params: 8,777,828 (33.48 MB)
Trainable params: 8,695,540 (33.17 MB)
Non-trainable params: 82,288 (321.44 KB)
```

FIGURE I
ORIGINAL MODEL ARCHITECTURE

Training Process:

Training was performed over 13 epochs with a batch size of 64 samples. A cosine learning rate scheduler was employed to adjust the learning rate during training. The training process was executed with verbosity level set to 1, providing detailed output information during each epoch.

Evaluation:

The evaluation metric in this competition is KL Divergence:

$$D_{KL}(P \,||\, Q) = \sum_i P(i) \log\left(\frac{P(i)}{Q(i)}\right)$$

Where P is the true distribution and Q is the predicted distribution. We used it directly as our loss function, removing the need of a third-party metric like Accuracy to evaluate our model. Therefore, valid loss can stand alone as an indicator for our evaluation. We used the implementation for KL Divergence loss in Keras:

```
keras.losses.KLDivergence()
```

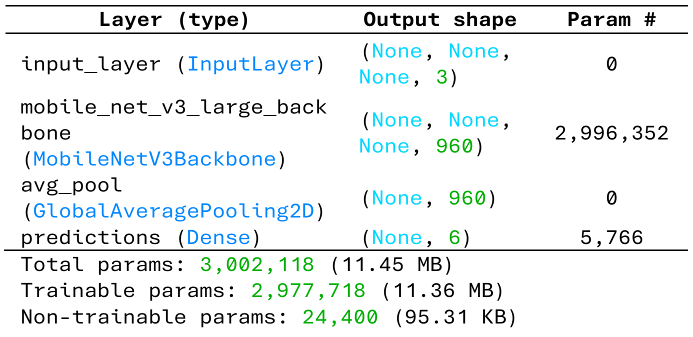
Changes to Original Architecture

Model Architecture:

In contrast to the original implementation utilizing EfficientNetV2, we opted for the utilization of MobileNetV3 for the model architecture. This substitution introduces alterations to the network's composition and parameterization, leading to distinctive performance characteristics. The modified architecture comprises the following components (See figure below):

- Input Layer: The initial layer of the network, accepting input images with three color channels (RGB).
- MobileNetV3 Large Backbone: Replacing the EfficientNetV2 backbone, the MobileNetV3 Large Backbone serves as the feature extraction component of the model. It consists of multiple layers specifically tailored to enhance efficiency and performance in mobile and edge computing environments.
- Global Average Pooling 2D Layer (avg_pool): A pooling layer responsible for reducing the spatial dimensions of the feature maps generated by the backbone. It computes the average value of each feature map across all spatial locations, resulting in a fixed-size output.
- Predictions Layer (predictions): The output layer of the model, consisting of a dense layer with six units corresponding to the six classes of EEG patterns to be classified.

The updated architecture configuration leads to a total of 3,002,118 parameters, with 2,977,718 parameters trainable and 24,400 non-trainable parameters. This substitution aims to explore the performance characteristics of MobileNetV3 in comparison to the original EfficientNetV2 architecture, potentially offering insights into the efficacy of different network architectures for the EEG pattern classification task.



```
 Layer (type)                     Output shape       Param #
─────────────────────────────────────────────────────────────
 input_layer (InputLayer)         (None, None,             0
                                  None, 3)
 mobile_net_v3_large_back         (None, None,     2,996,352
 bone                             None, 960)
 (MobileNetV3Backbone)
 avg_pool                         (None, 960)              0
 (GlobalAveragePooling2D)
 predictions (Dense)              (None, 6)            5,766
─────────────────────────────────────────────────────────────
 Total params: 3,002,118 (11.45 MB)
 Trainable params: 2,977,718 (11.36 MB)
 Non-trainable params: 24,400 (95.31 KB)
```

FIGURE II
MODIFIED MODEL ARCHITECTURE

Results

The performance of the model trained using MobileNetV3 Large as compared to the original model based on EfficientNetV2 is summarized below.

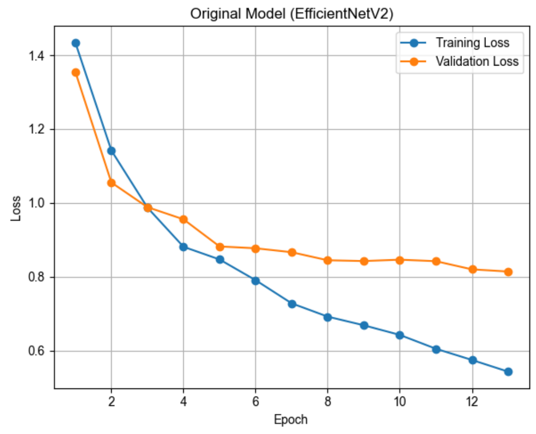


FIGURE III
LOSS VS EPOCH PLOT OF THE ORIGINAL MODEL

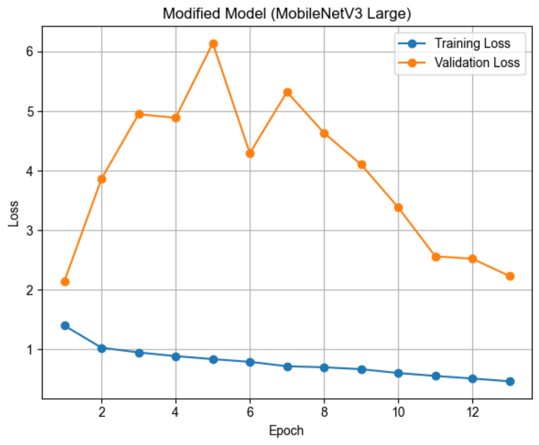


FIGURE IV
LOSS VS EPOCH PLOT OF THE NEW MODEL

It is evident from the results that both models exhibit a decrease in training loss and validation loss over the course of training epochs. However, the MobileNetV3 Large model demonstrates a slightly higher validation loss compared to the original model across all epochs. This suggests that the MobileNetV3 Large model may have slightly inferior performance in terms of generalization to unseen data compared to the original EfficientNetV2 model.

Further analysis could provide additional insights into the comparative performance of the two models. Additionally, experimentation with hyperparameters and model configurations may lead to improvements in the performance of both models.

The behavior observed in the validation loss curve of the modified model, where the loss initially increases before eventually decreasing after epoch 5, is indicative of a phenomenon commonly known as *overfitting* or *early overfitting*. Overfitting occurs when a model learns to perform well on the training data but fails to generalize its performance to unseen data, such as the validation set. This phenomenon often manifests as a decrease in training loss while the validation loss begins to increase, suggesting that the model is memorizing the training data rather than learning meaningful patterns. This could be attributed to several factors:

1. **Model Complexity:** The MobileNetV3 Large architecture might be more complex than necessary for the given task, leading to overfitting. This complexity could result in the model capturing noise or irrelevant patterns in the training data, which do not generalize well to unseen data.
2. **Insufficient Regularization:** The model may lack effective regularization techniques to prevent overfitting, such as dropout layers or weight decay. Without proper regularization, the model may become overly sensitive to the training data, resulting in poor generalization.
3. **Hyperparameter Tuning:** The hyperparameters of the model, such as learning rate or batch size, may not be optimized for the given dataset. Suboptimal hyperparameters can exacerbate overfitting and lead to poor generalization performance.

Lessons Learned

The experiments conducted provide valuable insights into the development and training of deep learning models for the task of seizure detection and classification using electroencephalography (EEG) signals. Several key lessons can be gleaned from our endeavors, shedding light on both the challenges encountered and the strategies employed to address them.

1. **Model Selection and Architecture**: The choice of model architecture plays a pivotal role in the performance of the seizure detection system. Our experiments comparing the EfficientNetV2 and MobileNetV3 architectures underscored the importance of selecting an appropriate model that balances complexity and effectiveness. While the EfficientNetV2 exhibited stable behavior throughout training, the MobileNetV3 demonstrated peculiar validation loss fluctuations, indicative of potential overfitting. This highlights the necessity of carefully evaluating and selecting model architectures tailored to the specific requirements and characteristics of the EEG dataset.
2. **Generalization and Overfitting:** The observed phenomenon of early overfitting in the modified model underscores the challenges associated with achieving generalization in deep learning models. Despite efforts to mitigate overfitting through regularization techniques, the MobileNetV3 architecture exhibited a divergence between training and validation loss after a few epochs. This emphasizes the importance of robust regularization strategies and hyperparameter tuning to prevent overfitting and enhance the model's generalization capabilities.

Conclusion

In this study, we explored the development of deep learning models for EEG-based harmful brain activity classification, with a focus on seizure detection and related patterns. Leveraging the HMS - Harmful Brain Activity Classification Kaggle competition dataset, we aimed to automate the classification of EEG segments into distinct patterns using machine learning techniques.

Our adaptation of the KerasCV starter notebook, originally designed with EfficientNetV2, featured a significant modification by substituting the model architecture with MobileNetV3. This change was motivated by the model´s efficiency and suitability for deployment on mobile and edge devices, offering potential advantages in real-world clinical applications.

The results from our experiments revealed notable insights into the comparative performance of the original and modified model architectures. While both models exhibited a decrease in training and validation loss over epochs, the MobileNetV3-based model demonstrated slightly inferior generalization performance compared to the EfficientNetV2 counterpart. This discrepancy, highlighted by the validation loss behavior, suggests potential challenges related to model complexity and overfitting.

The observed phenomenon of increasing validation loss followed by a subsequent decrease in the modified model's behavior underscores the importance of model selection, regularization strategies, and hyperparameter tuning. Addressing these factors is critical to enhancing model generalization and mitigating issues associated with overfitting, ultimately improving the reliability and accuracy of EEG pattern classification.

In summary, our study contributes valuable insights into the development and optimization of deep learning models for EEG-based harmful brain activity classification. Moving forward, further investigations into alternative model architectures, regularization techniques, and hyperparameter configurations are warranted to advance the state-of-the-art in automated EEG analysis, with the ultimate goal of improving patient care and outcomes in neurocritical care settings. The lessons learned from this study provide a foundation for future research aimed at refining and deploying machine learning solutions for real-time seizure detection and diagnosis.

AUTHOR INFORMATION

Joseph Martínez, Modeling and Simulation Engineering Master's student at Old Dominion University (ODU).

REFERENCES

1. Kaggle, *HMS - Harmful Brain Activity Classification*. https://www.kaggle.com/competitions/hms-harmful-brain-activity-classification.
2. Wood, L., et al., *KerasCV*. https://github.com/keras-team/keras-cv.