Machine Learning Models for Probability Classification in Spectrographic EEG Seizures Dataset

DENIS MANOLESCU, NEIL BUCKLEY, EMANUELE LINDO SECCO
School of Mathematics, Computer Science and Engineering,
Liverpool Hope University,
Hope Park, L16 9JD, Liverpool,
UK

Abstract: - The examination of brain signals, namely the *Electroencephalogram* (EEG) signals, is an approach to possibly detect seizures of the brain. Due to the nature of these signals, deep learning techniques have offered the opportunity to perform automatic or semi-automatic analysis which could support decision and therapeutical approaches. This paper focuses on the possibility of classifying EEG seizure using convolutional layers (namely *EfficientNetV2* architectures, i.e., *EfficientNetV2S* and *EfficientNetV2B2*), *Long Short-Term Memory* (LSTM) units, and fine-tuned mechanisms of attention. We use these techniques to untangle the complexity of these signals and accurately predict seizures. The proposed system provided interesting results with an 86.45% accuracy under the *Kullback-Leibler Divergence* loss of 0.95. Moreover, these results showed that embedding LSTM layers deeply increases the quality of the results since these layers support the analysis of the spatial-temporal dynamics of the EEG signals. On the other hand, it is important to mention that hardware limitations could affect these results and therefore it is important, when setting this architectural system, to fine-tune the data set and balance the performance vs the computational cost of the process.

Key-Words: - Machine Learning Models, Attention Mechanism, Probabilistic Classification, EEG, Support-Decision System, Kullback-Leibler Divergence.

Received: January 16, 2024. Revised: July 17, 2024. Accepted: August 19, 2024. Published: September 23, 2024.

1 Introduction

Electroencephalography (EEG), an approach to the analysis brain signals and, in particular, electrical brain signals, developed in the 19th century and in the time of the Industrial Revolution, [1]. Around the 1920s, Has Berger proposed the first recording tool for the EEG signals, [2], which then brought to the possibility of analyzing brain activity and correlating this information with the functionality of the brain as well as with its pathologies, [3], [4]. In particular, new discoveries follow regarding epilepsy, sleep patterns, [5], [6], and other seizures and neurological disorders, [7], [8], [9].

Due to the nature of the EEG signals, it is sometimes difficult to find overall consistency in its interpretation, [10], [11], and therefore the importance of standard methodologies which can support the decision and visualization has grown especially vs the detection of epilepsy, which is one of the main interest of this work, [12]. In this context, *Machine Learning* (ML) techniques are a useful tool, together with the classic approaches of the literature, such as the analysis in the frequency and time domains, the use of *Convolutional Neural Networks* (CCNs), and *Support Vector Machines* (SVMs), [13], [14].

CNNs-based techniques showed robustness vs the inherently physiological variability of EEG signals, [15], especially when integrated with the mechanism of attention, [16], [17]. Another interesting aspect of this approach is the possibility to transfer these learning strategies as it has been reported, for example, in the analysis of datasets – e.g. g., UC Irvine ML Repository, imageNet or similar – for seizure detection of [18] and [19]. In addition, by carefully managing the learning rate adjustments throughout the training process using a scheduler mechanism, some models have been shown to optimize convergence and mitigate overfitting, [20], [21].

While numerous models have made significant progress, a critical limitation persists: many focus on binary deterministic classifications (e.g., seizure vs. non-seizure). This approach leaves clinicians without vital information on the uncertainty embedded in model predictions. *Probability-Based Classification* (PBC) addresses this drawback by quantifying the confidence associated with each prediction, [22]. PBC offers advantages for clinical decision-making by potentially flagging cases with lower model confidence for expert review. Evaluation metrics that align with PBC approaches, such as Kullback-Leibler divergence (KLD), are meant to provide a more comprehensive assessment of model performance than traditional accuracy-based metrics, [23]. In this study, we explore the integration of Kullback-Leibler Divergence (KLD) within our machine learning models aimed at improving seizure detection to enhance their practical utility in healthcare settings

As the literature suggests, pre-processing and data augmentation strategies go beyond the structural optimization of the model to directly enhance seizure detection capabilities. These strategies are vital in addressing overfitting and improving the generalizability of models, which are particularly important when working with limited datasets, [24]. In this perspective, it is also important to mention the importance of proper analysis in the time and frequency domain with, for example, *Short-Time Fourier Transform* (STFT), [14], [25], [26] and *Continuous Wavelet Transform* (CWT), [27], which has shown limitation in the management of the resolution vs the high dynamics of these signals. At the same time, there has been interesting progress on the refinement of the EfficientNetV2 system with new challenges which we want to face in this work.

The paper is organized as follows: in the following section, we introduce the dataset and the main pre-processing approach with a focus on the implementation's details. Further, this section highlights the use of Kullback-Leibler divergence (KLD) as the primary evaluation metric, augmented by an adaptable learning rate scheduler to refine training dynamics. The exploration of model architectures is grounded on two pre-trained EfficientNetV2 variants (B2 and S), each trained and initialized with ImageNet weights, serving as the core foundational backbone. This chapter continues with our investigation, examining a range of model configurations, including 2D, 3D, and pointwise convolutional neural networks, alongside recurrent neural network (RNN-LSTM) structures, all configured to optimize transfer learning. Here, we diligently assess the impact of attention mechanisms, focusing on how adjustments in headcounts and dimensionality influence performance. Further refinement of our models is assessed using Keras AutoML to systematically explore a broad spectrum of hyperparameter settings, seeking configurations that elevate model performance. The subsequent Results chapter presents a comprehensive analysis of the findings from our training processes. Finally, the conclusion reports some observations about the limits of our approach and how we could extend that vs the detection of EEG seizures.

2 Materials & Methods

2.1 Project Environment and Setup

This work is structured around Kaggle, [28], a platform that properly suits machine learning studies applied to EEG datasets. Data are provided by the Medical School of Harvard University and partners for the HMS Harmful Brain Activity Classification competition [29]. The competition made available a hosted notebook with a hardware configuration of 70GB of disk storage, with 29GB of CPU and 32GB GPU power (T4 x2 or P100 x1). This setup was powerful enough to facilitate the development, training, and experimentation phases of the model, the handling of the large dataset (25 GB), and the efficient training of the complex neural network architectures.

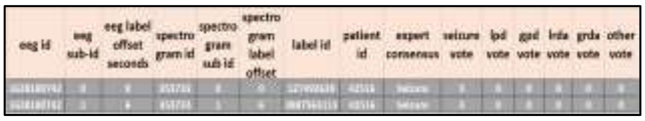
2.2 Database

2.2.1 Database Description

The dataset for this study contains real raw EEG recordings along with the metadata that links the brain signals to expert classifications. This design was structured to challenge and evaluate models on their ability to detect and classify seizures and other harmful brain activity. The difficulty of this task is increased by the variability in expert consensus, reflecting the complex nature of EEG data interpretation. The classification involves different patterns, such as seizure, generalized and lateralized periodic discharges, generalized and lateralized rhythmic delta activity and others. The data are sampled with overlapping time frames or windows of 10 second each and then, in order to provide a ground truth reference, the data are classified by professional experts as it is shown in Figure 1.

2.2.2 Optimize, Convert, Segmentation, Partition, Augmenting

Data are available in *parquet* format: they are then processed into *.npy* format by means of a *process_spec*() function. A *joblib* librabry is also used in order to optimize the CPU workload.



eeg id	eeg sub-id	eeg label offset seconds	spectro gram id	spectro gram sub id	spectro gram label offset	label id	patient id	expert consensus	seizure vote	lpd vote	gpd vote	lrda vote	grda vote	other vote
1628180742	0	0	353733	0	0	127492639	42516	Seizure	3	0	0	0	0	0
1628180742	1	6	353733	1	6	3887563113	42516	Seizure	3	0	0	0	0	0

Fig. 1: Network Dataset Layout before K-Fold

In order to optimise the analysis and the efficiency of the *EfficientNetV2* pattern recognition, 10 seconds overlapping and non-overlapping set of samples are saved. Data are partitioned into a training set and testing set as well.

Additionally, during data loading for model training, we incorporated signal augmentation techniques such as *MixUp* and *Random Cutout*. *MixUp* creates synthetic examples by blending images and labels, forcing the model to learn feature interpolations. In parallel, *Random Cutout* strategically obscure portions of images, forcing the model to recognize patterns beyond dominant features. Both these techniques enrich the training dataset, helping our models improve robustness and consistent performance across diverse scenarios.

The complete code for the conversion process, data segmentation, and the augmentation techniques used are presented in *Appendix 1* and they were implemented according to the Kaggle guidelines and applications, [28], [29]. In the Appendix, we show the Data Cleaning & Conversion (panels B, D of Figure 11B, Appendix), the Segmentation process (panel C, Figure 11B, Appendix), and finally the Augmentation (panel A, Figure 11A, Appendix).

2.3 Building the Models

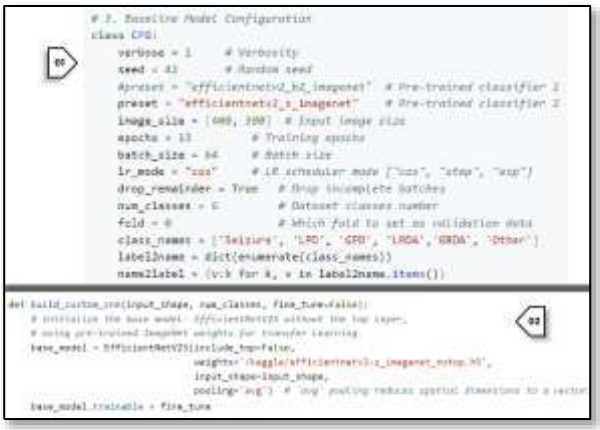
2.3.1 Baseline, Transfer Learning, and Base Model

This research adopts a transfer learning strategy with the *EfficientNetV2* architecture, pre-trained on the ImageNet dataset, to classify EEG spectrograms. The approach of recognizing brain signals as complex time-frequency images uses *EfficientNetV2* as the backbone for its advanced visual pattern recognition capabilities, proven effective and scalable across various image datasets, [30].

In practice, the study employs *"no-top"* versions of both *EfficientNetV2-B2* and *EfficientNetV2-S*, which removes the pre-trained final classification layer of the models (Figure 1, panel 02). This modification allows the two variants to be more adaptable for EEG probability seizure classification, diverging from their original purpose of general image classification on ImageNet. Custom output layers were subsequently designed to fine-tune the architecture, optimizing the ability of the systems to process the convoluted and high-dimensional information characteristic of EEG spectrograms, [31].

By applying transfer learning, the study aims to leverage the broad features these models have learned from ImageNet. This transfer is adjusting them to identify distinctive patterns associated with seizures and other neurological phenomena of interest from the database. In addition, the base model, represented by the CFG class, supports simple experimental comparisons between the *EfficientNetV2-S* and *EfficientNetV2-B2* (Figure 2, panel 01).

Changing the *preset* parameter within the base model facilitates a controlled evaluation of each model, ensuring a fair baseline comparison in the context of neurological pattern recognition. This strategy exploits the EfficientNetV2 architecture with its learned visual pattern detection capabilities, maximizing the utility of its design for the specialized task at hand.



```
# 3. Baseline Model Configuration
class CFG:
    verbose = 1      # Verbosity
    seed = 42        # Random seed
    #preset = "efficientnetv2_b2_imagenet"  # Pre-trained classifier 1
    preset = "efficientnetv2_s_imagenet"    # Pre-trained classifier 2
    image_size = [400, 300]  # Input image size
    epochs = 13          # Training epochs
    batch_size = 64      # Batch size
    lr_mode = "cos"      # LR scheduler mode ["cos", "step", "exp"]
    drop_remainder = True   # Drop incomplete batches
    num_classes = 6         # Dataset classes number
    fold = 0                # Which fold to set as validation data
    class_names = ['Seizure', 'LPD', 'GPD', 'LRDA','GRDA', 'Other']
    label2name = dict(enumerate(class_names))
    name2label = {v:k for k, v in label2name.items()}
```

```
def build_custom_cnn(input_shape, num_classes, fine_tune=False):
    # Initialize the base model: EfficientNetV2S without the top layer,
    # using pre-trained ImageNet weights for transfer learning
    base_model = EfficientNetV2S(include_top=False,
                                 weights='/kaggle/efficientnetv2-s_imagenet_notop.h5',
                                 input_shape=input_shape,
                                 pooling='avg')  # 'avg' pooling reduces spatial dimensions to a vector
    base_model.trainable = fine_tune
```

Fig. 2: Baseline CFG Class (01); No-Top Base Model for Transfer Learning (02)

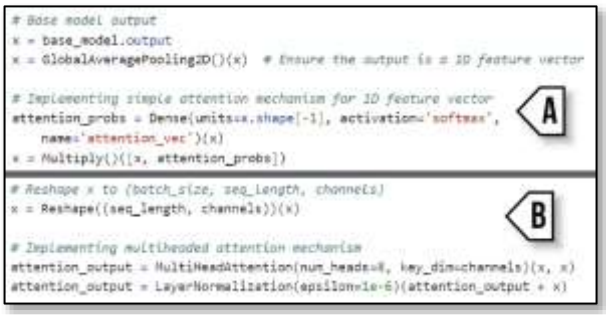
2.3.2 Attention Mechanism

Simple Attention Mechanism (Model 2)

At its core, *Attention Mechanisms* (AM) allow neural networks to selectively focus on the most informative regions within input data. It enables the model to dynamically weigh the significance of different input features, an ability that closely mirrors how the human brain processes complex stimuli, [32]. In the context of EEG spectrogram classification, attention helps models identify and prioritize the spectral features most strongly associated with seizures and other neurological activity.

In developing Model-2, the study began by integrating a simple AM (Figure 3, panel A) into a *no-top* EfficientNetV2-B2 architecture to refine the classification of EEG spectrograms. By generating an attentional vector that assigns weights to different features in the EEG spectrogram, the model highlights areas likely to contain critical information for classification. In addition, with the EfficientNetV2-B2 backbone maintained in a non-trainable state, the model leverages the depth of pre-trained features while concentrating adaptive efforts on the subtle features of attention and refinement through custom top layers.

From the backbone base, the Model-2 structure continues with a *GlobalAveragePooling2D* to condense the output into a 1D feature vector, applies a simple AM to emphasize critical features, and completes with custom *Dense-SoftMax* layers for precise probability classification output.



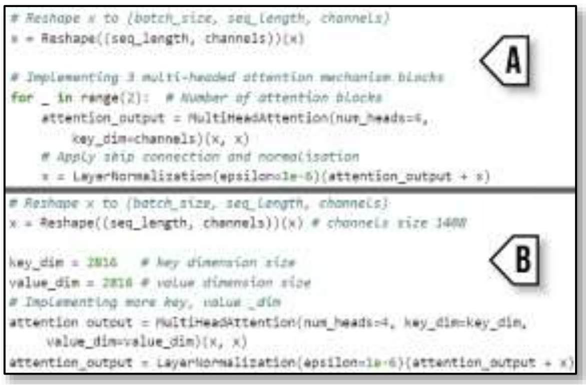
```
# Base model output
x = base_model.output
x = GlobalAveragePooling2D()(x)  # Ensure the output is a 1D feature vector

# Implementing simple attention mechanism for 1D feature vector
attention_probs = Dense(units=x.shape[-1], activation='softmax',
    name='attention_vec')(x)
x = Multiply()([x, attention_probs])
```

```
# Reshape x to (batch_size, seq_length, channels)
x = Reshape((seq_length, channels))(x)

# Implementing multiheaded attention mechanism
attention_output = MultiHeadAttention(num_heads=8, key_dim=channels)(x, x)
attention_output = LayerNormalization(epsilon=1e-6)(attention_output + x)
```

Fig. 3: Simple (A) and Multiheaded (B), Attention Mechanism Implementation

Multiheaded Attention Block (Models 3 & 4)

Next, with the building of Model-3 and Model-4, the research explored the complexities of multi-headed AMs (Figure 3, panel B). This refined version allows the models to attend to multiple parts of the input data simultaneously through separate *"heads"* that operate in parallel. Each head can capture different aspects of the input data, providing a composite understanding of the input features and further refined by subsequent normalization and pooling layers (*LayerNormalization* and *GlobalAveragePooling1D*).



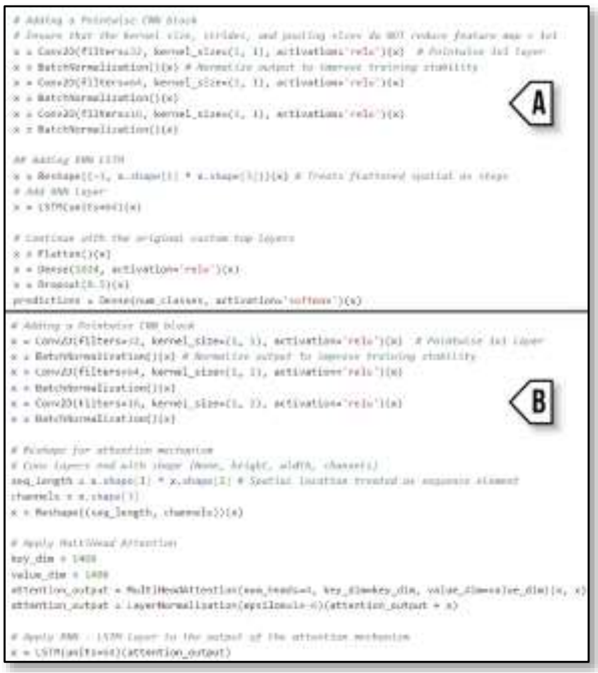
```
# Reshape x to (batch_size, seq_length, channels)
x = Reshape((seq_length, channels))(x)

# Implementing 3 multi-headed attention mechanism blocks
for _ in range(2):  # Number of attention blocks
    attention_output = MultiHeadAttention(num_heads=6,
        key_dim=channels)(x, x)
    # Apply skip connection and normalisation
    x = LayerNormalization(epsilon=1e-6)(attention_output + x)
```

```
# Reshape x to (batch_size, seq_length, channels)
x = Reshape((seq_length, channels))(x) # channels size 1408

key_dim = 2016   # key dimension size
value_dim = 2016 # value dimension size
# Implementing more key, value _dim
attention output = MultiHeadAttention(num_heads=4, key_dim=key_dim,
    value_dim=value_dim)(x, x)
attention_output = LayerNormalization(epsilon=1e-6)(attention_output + x)
```

Fig. 4: Code Example for Attention Blocks (A) and Expanded Key and Value Dimensions (B)

Multidimensional Attention (Model 5 to 8)

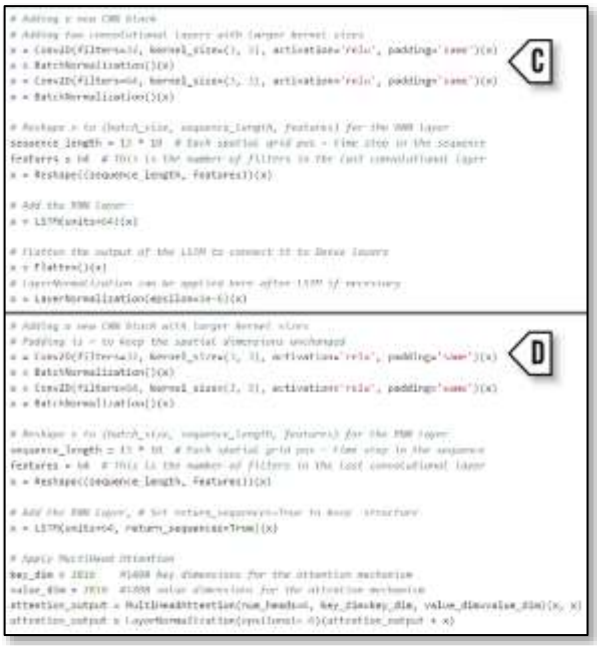
In order to improve the performance of the system and better characterize and represent the main features of the data, we also embedded multiple blocks of attention (Figure 4, panels A and B). By increasing dimensionality, the systems can discover critical, yet subtle, features for accurate EEG spectrogram classification.



```
# Adding a Pointwise CNN block
# Ensure that the kernel size, strides, and pooling sizes do NOT reduce feature map < 1x1
x = Conv2D(filters=32, kernel_size=(1, 1), activation='relu')(x)  # Pointwise 1x1 layer
x = BatchNormalization()(x) # Normalize output to improve training stability
x = Conv2D(filters=64, kernel_size=(1, 1), activation='relu')(x)
x = BatchNormalization()(x)
x = Conv2D(filters=16, kernel_size=(1, 1), activation='relu')(x)
x = BatchNormalization()(x)

## Adding RNN LSTM
x = Reshape((-1, x.shape[1] * x.shape[3]))(x) # Treats flattened spatial as steps
# Add RNN layer
x = LSTM(units=64)(x)

# Continue with the original custom top layers
x = Flatten()(x)
x = Dense(1024, activation='relu')(x)
x = Dropout(0.5)(x)
predictions = Dense(num_classes, activation='softmax')(x)
```

```
# Adding a Pointwise CNN block
x = Conv2D(filters=32, kernel_size=(1, 1), activation='relu')(x)  # Pointwise 1x1 layer
x = BatchNormalization()(x) # Normalize output to improve training stability
x = Conv2D(filters=64, kernel_size=(1, 1), activation='relu')(x)
x = BatchNormalization()(x)
x = Conv2D(filters=16, kernel_size=(1, 1), activation='relu')(x)
x = BatchNormalization()(x)

# Reshape for attention mechanism
# Conv layers end with shape (None, height, width, channels)
seq_length = x.shape[1] * x.shape[2] # Spatial location treated as sequence element
channels = x.shape[3]
x = Reshape((seq_length, channels))(x)

# Apply MultiHead Attention
key_dim = 1400
value_dim = 1400
attention_output = MultiHeadAttention(num_heads=4, key_dim=key_dim, value_dim=value_dim)(x, x)
attention_output = LayerNormalization(epsilon=1e-6)(attention_output + x)

# Apply RNN - LSTM layer to the output of the attention mechanism
x = LSTM(units=64)(attention_output)
```

Fig. 5A: Model-9 (A) with 2D CNN-Pointwise Layers & LSTM; Model-11 (B) with CNN-Pointwise Layers A M and LSTM



```
# Adding a new CNN block
# Adding two convolutional layers with larger kernel sizes
x = Conv2D(filters=32, kernel_size=(3, 3), activation='relu', padding='same')(x)
x = BatchNormalization()(x)
x = Conv2D(filters=64, kernel_size=(3, 3), activation='relu', padding='same')(x)
x = BatchNormalization()(x)

# Reshape x to (batch_size, sequence_length, features) for the RNN layer
sequence_length = 12 * 10  # Each spatial grid pos = time step in the sequence
features = 64  # This is the number of filters in the last convolutional layer
x = Reshape((sequence_length, features))(x)

# Add the RNN layer
x = LSTM(units=64)(x)

# Flatten the output of the LSTM to connect it to Dense layers
x = Flatten()(x)
# LayerNormalization can be applied here after LSTM if necessary
x = LayerNormalization(epsilon=1e-6)(x)
```

```
# Adding a new CNN block with larger kernel sizes
# Padding is = to keep the spatial dimensions unchanged
x = Conv2D(filters=32, kernel_size=(3, 3), activation='relu', padding='same')(x)
x = BatchNormalization()(x)
x = Conv2D(filters=64, kernel_size=(3, 3), activation='relu', padding='same')(x)
x = BatchNormalization()(x)

# Reshape x to (batch_size, sequence_length, features) for the RNN layer
sequence_length = 12 * 10  # Each spatial grid pos = time step in the sequence
features = 64  # This is the number of filters in the last convolutional layer
x = Reshape((sequence_length, features))(x)

# Add the RNN layer, # Set return_sequences=True to keep structure
x = LSTM(units=64, return_sequences=True)(x)

# Apply MultiHead Attention
key_dim = 2016    #1400 key dimensions for the attention mechanism
value_dim = 2016  #1400 value dimensions for the attention mechanism
attention_output = MultiHeadAttention(num_heads=4, key_dim=key_dim, value_dim=value_dim)(x, x)
attention_output = LayerNormalization(epsilon=1e-6)(attention_output + x)
```

Fig. 5B: Model-12 (C) with 3D CNN Layers & RNN LSTM; Model-13 (D) with 3D CNN Layers, Layers AM and LSTM

Models 5 through 8 represent a deeper exploration of AMs. We introduced variations in multi-headed attention configurations and expanded the dimensionality of key and value pairs to investigate the complexity and capacity of the attention mechanism to differentiate and prioritize information within EEG signals. All these models utilize the *EfficientNetV2B2* architecture as their foundation and incorporate up to eight attentional heads. Models 5 and 7 specifically examine the effects of doubling the key and value dimensions, while Model 8 explores tripling these dimensions to achieve an even higher level of detail in feature processing. Additionally, Model 6 integrates multiple attentional blocks, layering the attention mechanism to higher convolutions and deepening the analysis of EEG features.

Each model employs a sequence of steps starting from the base-model output, reshaping it to align with the multi-head AM requirements, followed by a *Normalization* layer to stabilize learning. *GlobalAveragePooling1D* is consistently used to condense the data into a more manageable form for the final classification layers. This sequence ends in custom top layers, including *Dense* and *Dropout* layers, leading to a *SoftMax* classification output. These enhancements in AMs refine the focus on relevant EEG features, aiming for a fine and highly accurate classification of neurological patterns.

CNN-Pointwise & RNN-LSTM Layers (Model 9 to 13)

Building on the attentional mechanisms explored earlier, Models 9–13 investigate the complex relationship between spatial feature extraction, temporal analysis, and attentional focus for EEG spectrogram classification (Figure 5A & Figure 5B). These aspects are implemented by integrating convolutional neural networks and long short-term memory features. In addition, each model was evaluated across both structural backbone architectures, *EfficientNetV2S* and *EfficientNetV2B2*. As a strategic shift, the models incorporated LSTM layers to capture inherent temporal dependencies and dynamics specific to the sequential nature of brain activity patterns, crucial for accurate seizure detection.

A central line of analysis was the optimization placement of AMs. Model-11 uniquely applied attention before the RNN layer, allowing us to investigate whether the model benefits from identifying crucial spectral features prior to sequential analysis. In contrast, Model-13 applied AMs after the LSTM, testing if focusing on the most significant temporal patterns enhanced classification. Additionally, all these five models have been used to experiment with different convolutional architectures, including the 2D 1x1 pointwise convolution layers for fine-grained feature mapping and larger kernel convolutional layers for broader spatial relationships. This diversified approach aims to identify the spatial feature resolutions most descriptive of seizure activities within EEG spectrograms.

Models 12 and 13 uniquely adopt a 3D convolutional approach, treating the spectrogram as an inherently spatiotemporal representation (Figure 5B). This aspect aimed to determine whether modelling frequency and temporal dynamics simultaneously held advantages. Finally, attention mechanisms were refined with varying key and value dimensions, potentially allowing systems to identify more complex relationships within the EEG data.

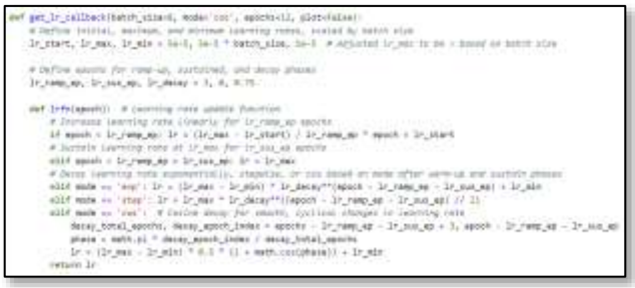

```
def get_lr_callback(batch_size=8, mode='cos', epochs=13, plot=False):
    # Define initial, maximum, and minimum learning rates, scaled by batch size
    lr_start, lr_max, lr_min = 5e-5, 5e-5 * batch_size, 1e-5  # Adjusted lr_max to be [illegible] based on batch size

    # Define epochs for ramp-up, sustained, and decay phases
    lr_ramp_ep, lr_sus_ep, lr_decay = 3, 0, 0.75

    def lrfn(epoch):  # Learning rate update function
        # Increase learning rate linearly for lr_ramp_ep epochs
        if epoch < lr_ramp_ep: lr = (lr_max - lr_start) / lr_ramp_ep * epoch + lr_start
        # Sustain learning rate at lr_max for lr_sus_ep epochs
        elif epoch < lr_ramp_ep + lr_sus_ep: lr = lr_max
        # Decay learning rate exponentially, stepwise, or cos based on mode after warm-up and sustain phases
        elif mode == 'exp': lr = (lr_max - lr_min) * lr_decay**(epoch - lr_ramp_ep - lr_sus_ep) + lr_min
        elif mode == 'step': lr = lr_max * lr_decay**((epoch - lr_ramp_ep - lr_sus_ep) // 2)
        elif mode == 'cos':  # Cosine decay for smooth, cyclical changes in learning rate
            decay_total_epochs, decay_epoch_index = epochs - lr_ramp_ep - lr_sus_ep + 3, epoch - lr_ramp_ep - lr_sus_ep
            phase = math.pi * decay_epoch_index / decay_total_epochs
            lr = (lr_max - lr_min) * 0.5 * (1 + math.cos(phase)) + lr_min
        return lr
```

Fig. 6: Learning Rate Scheduler

2.4 LR Scheduler, Kullback-Leibler Divergence & Other Metrics

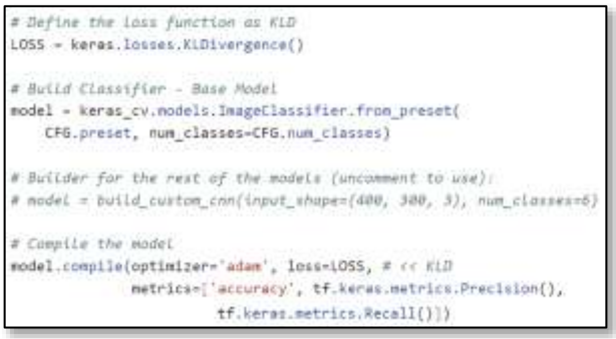
To optimize model convergence, a learning rate (LR) scheduler was implemented with an initial warmup phase (*lr_ramp_ep*) to establish model stability (Figure 6). This phase transitions into a sustained period of maximum learning rate (*lr_sus_ep*), intensifying the model's focus on critical EEG patterns. Subsequently, a customizable decay function (*cos*) is applied to gradually reduce the learning rate, facilitating precise weight adjustments and reducing the risk of overfitting in later training epochs. The learning rate hyperparameters (*lr_start, lr_max, lr_min*) were chosen by both best practices in deep learning and the batch size used, ensuring the schedule was adaptable to the specific dataset and training dynamics.

We also incorporated the *Kullback-Leibler Divergence* (KLD) function of loss within the training of the system (Figure 7 and Eq. (1)):

$$D_{KL}(P||Q) = \sum_{x}^{\infty} \left(P(i) \log \frac{P(i)}{Q(i)} \right) . \qquad (1)$$

Where P – true distribution; Q – predicted distribution.

This parameter is a proper marker of the classification performance in terms of accuracy, in terms of *precision,* and *recal*l, namely the correct detection of seizures and the minimization of false negatives, respectively.

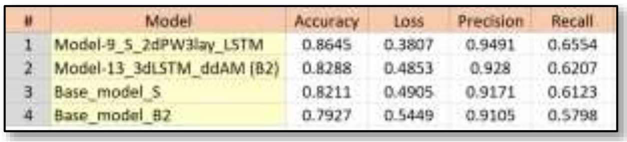


```
# Define the loss function as KLD
LOSS = keras.losses.KLDivergence()

# Build Classifier - Base Model
model = keras_cv.models.ImageClassifier.from_preset(
    CFG.preset, num_classes=CFG.num_classes)

# Builder for the rest of the models (uncomment to use):
# model = build_custom_cnn(input_shape=(400, 300, 3), num_classes=6)

# Compile the model
model.compile(optimizer='adam', loss=LOSS, # << KLD
              metrics=['accuracy', tf.keras.metrics.Precision(),
                       tf.keras.metrics.Recall()])
```

Fig. 7: Kullback-Leibler Divergence (KLD) implementation

2.5 AutoML Hyperparameter Tuner

Another step in order to adjust the system consists of introducing *Keras AutoML* which optimizes the overall configuration of the classifier such as the setting of the learning rate and the optimization of the parameters. Through an interactive process, we refine the setting and obtain an optimal configuration.



#	Model	Accuracy	Loss	Precision	Recall
1	Model-9_S_2dPW3lay_LSTM	0.8645	0.3807	0.9491	0.6554
2	Model-13_3dLSTM_ddAM (B2)	0.8288	0.4853	0.928	0.6207
3	Base_model_S	0.8211	0.4905	0.9171	0.6123
4	Base_model_B2	0.7927	0.5449	0.9105	0.5798

Fig. 8: Top 4 Model Performance

2.6 Training, Evaluation, Testing and Inference

For training the system, the number of epochs was defined (*CFG.epochs=13*) together with the usual parameters such as the learning rate and verbose output (*CFG.verbose*). A validation set was also established (*valid_ds*) together with a *model.evaluate* method. Finally, another set of data was prepared for testing (*test_ds*)).

Finally, the trained model was used to generate predictions (*model.predict*) on the test dataset and save the output into a CVS file. This process involved pairing the predictions with required EEG identifiers (*eeg_id*), for possible later analysis and evaluation procedures.

3 Results & Discussion

Analysis of the results revealed Model-9 (configured with *EfficientNetV2S* backbone, 2D pointwise convolution, and LSTM layers) and Model-13 (composed of *EfficientNetV2B2* backbone, 3D convolution, and LSTM layers, with double dimension AM factors and 6 attention heads) to be the most accurate architectures, achieving an accuracy of 0.8645 and 0.8288, respectively (Figure 9). These two models outperformed both the *EfficientNetV2-B2* and *EfficientNetV2-S* baselines (Figure 8). This difference suggests their improved capacity to decode and learn from complex, time-sensitive patterns in EEG spectrograms based on their integrated spatial-temporal processing layers. In addition, both configurations, 9 and 13, achieve the lowest KLD loss, which indicates confidence in their predictions and confirms their superior certainty in classifications. This characteristic is desirable in real-world EEG seizure detection systems, where minimizing both false alarms and missed seizures is crucial.

In all cases, integrating convolution layers and LSTM units improved the models. The effectiveness of this approach was universally observed, suggesting that additional investigation into the optimal configurations with this technique could yield even better results.

Further examination of the performance results reveals a significant decrease in accuracy for Model-6 and Model-8, which could be attributed to several factors:

Model-6, with its multiple attention blocks, each featuring four heads, indicates increased complexity issues, leading to difficulties in training. Each additional attention block introduces more parameters to learn, which requires more data and computational power to optimize effectively. When not regulated, this complexity results in the model overfitting to the training data or not converging on an optimal solution. In this particular case, the training data was not diverse or large enough to support learning these additional parameters.

Likewise, Model-8 encountered similar issues with its 4-head attention mechanism and tripled key-value dimensions. While intended to provide a more detailed feature processing capability, the substantial increase in dimensionality has caused the model to become too specialized in the training data variations, failing to generalize well to validation data. This phenomenon is known as the *“curse of dimensionality”*, [33], where adding more features increases the volume of the feature space exponentially.

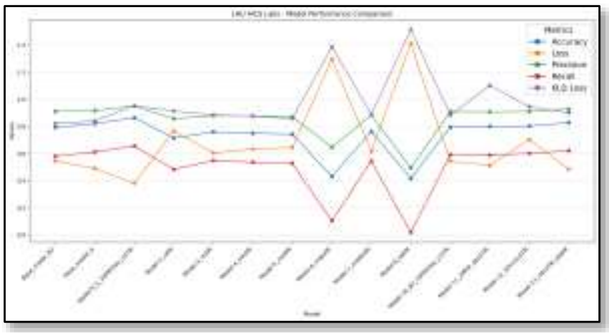


Fig. 9: Performance results of all 13 Models

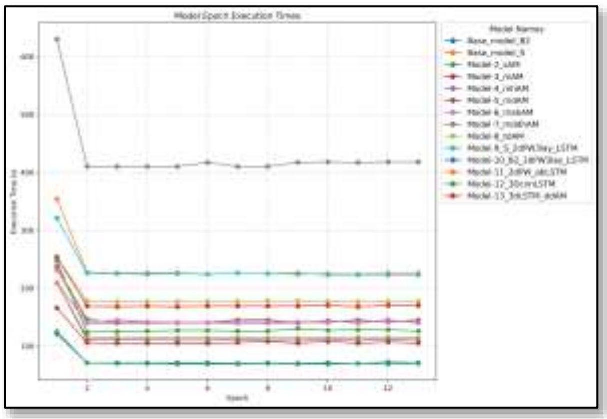


Fig. 10: Execution times/epoch during 13-train cycles (all models)

Without enough training data to cover this space, the model performance can deteriorate considerably. Furthermore, Model-6 and 8 demonstrate the risks of overfitting. Their poor accuracy is intensified by very high KLD loss, indicating incorrect predictions made with excessive certainty. This result highlights the importance of balancing model complexity with the need for reliable and well-calibrated prediction.

Overall, multi-headed models with fine-tuning generally outperformed those with simple attention design and showed promising results against the baselines. Similarly, expanding the key-value dimensions in the attentional modules led to stable accuracy, highlighting the value of capturing detailed feature relationships.

However, as *Attention Mechanisms* (AM) became more complex, the systems encountered *Out-Of-Memory* (OoM) errors during training. This event is also reflected in Model-7 training time per epoch, with its multi-dimensional 8-head AM configuration (Figure 10). Interestingly, these resource limitations become more pronounced when increasing the number of attentional heads compared to expanding key-value dimensions. This indicates that, given hardware constraints (29 GB of CPU and 32 GB of GPU power with a T4 GPU), adjusting the dimensionality presents a more computationally efficient method to boost the representational power of models within attention mechanisms.

The performance results of both baseline systems indicate that transfer learning is a highly effective strategy in machine learning developments for brain signal classification tasks. The foundational designs, *EfficientNetV2S* and *EfficientNetV2B2*, provided a solid starting point for testing and improving complex pattern recognition concepts. These *no-top* pre-trained models can accelerate the learning process, allowing for a direct focus on the subtle characteristics of EEG data. *Appendix 2* provides a detailed summary of the performance metrics in heatmap format (Figure 12, Appendix) and also offers a more comprehensive comparison of the models through bars and spiderweb charts (Figure 13, Appendix). Additionally, the appendix includes a confusion matrix that presents the performance metrics of the model.

The overall results of this work indicate that a stable approach to EEG multi-class classification can be achieved through transfer learning with *EfficientNetV2*, refined by specialized layer architectures.

4 Conclusion

This work analyses the possibility of using convolutional and LSTM layers, combined with attention mechanisms, in order to classify EEG seizure. The proposed system shows an accuracy of 86.45% with a KL divergence loss of 0.95. Moreover we showed that *EfficientNetV2S* and *EfficientNetV2B2* with the integration of LSTM and convolutional layers significantly improves the classification performance. However, while incorporating attention mechanisms with higher local dimensionalities (keys and values) further enhanced accuracy, producing richer and more informative outputs, we noticed that distributing these dimensionalities across multiple attention heads led to decreased performance and unsustainable computational demands.

These findings emphasize the need for more research into strategies to adjust model depth complexity with computational efficiency to balance performance without overextending system capabilities.

Future work will concentrate on enhancing the performance of the EEG seizure detection systems by exploring the benefits of learning from larger and more diverse datasets, as well as on looking at proper systems and software for EEG data acquisition, [34], [35], [36]. A particular focus will be on researching techniques to reduce the KL divergence to increase system confidence and accuracy. Additionally, adopting incremental learning strategies could allow for further model improvements. All these development efforts hold the potential to improve the reliability of machine learning systems and their integration while supporting a more personalized and superior quality of healthcare.

Acknowledgement:
This work was completed by Denis Manolescu as part of his coursework requirements for the MEng in Robotics at Liverpool Hope University's Robotics Laboratory within the School of Mathematics, Computer Science, and Engineering.

References:

[1] Caton, R. (1875, August 28). The Electric Current of the Brain, Forty-Third Annual Meeting of the British Medical Association, Br Med J. 1875, 2(765):257–79, [Online]. https://www.ncbi.nlm.nih.gov/pmc/articles/PMC2297516/?page=22 (Accessed Date: May 30, 2024).

[2] Kaplan, Kaplan RM. The Mind Reader: the Forgotten Life of Hans Berger, Discoverer of the EEG. *Australasian Psychiatry*. 2011, 19(2), pp.168-169, https://doi.org/10.3109/10398562.2011.561495.

[3] Diukova, G. M., Makarov, S. A., Golubev, V. L., Tyutina, R. R., Degterev, D. A., & Danilov, A. B. (2020). Psychogenic Seizure Imitating Narcolepsy. *Case Rep Neurol*., (2021) 12 (3), pp.472–481. 2021, https://doi.org/10.1159/000510517.

[4] Ye, E.M., Sun, H., Krishnamurthy, P.V., Lam, A.D. and Westover, M.B. (2021), Dementia detection from brain activity during sleep. Alzheimer's *Dement*., 17, https://doi.org/10.1002/alz.058718.

[5] Nielsen, J. M., Zibrandtsen, I. C., Masulli, P., Sørensen, T. L., Andersen, T. S., & Kjær, T. W. Towards a wearable multi-modal seizure detection system in epilepsy: A pilot study. *Clinical Neurophysiology*, 136, pp.40-48, 2022, https://doi.org/10.1016/j.clinph.2022.01.005.

[6] Berdina, O., Madaeva, I., & Rychkova, L. (2023). Sleep EEG pattern in childhood: from newborn through adolescent, *Eur. Phys. J. Spec. Top*., (2024) 233, pp.705–716, https://doi.org/10.1140/epjs/s11734-023-01071-5.

[7] Munjal NK, Bergman I, Scheuer ML, Genovese CR, Simon DW, Patterson CM. Quantitative Electroencephalography (EEG) Predicting Acute Neurologic Deterioration in the Pediatric Intensive Care Unit: A Case Series. *Journal of Child Neurology*. 2022, 37(1), pp.73-79, https://doi.org/10.1177/08830738211053908.

[8] Shelig M, Ames M, Young GB. Detection of Atrial Fibrillation in Routine EEG Recordings. Canadian Journal of Neurological Sciences, *Journal Canadien des Sciences Neurologiques*. 2023, 50(1), pp.23-27, https://doi.org/10.1017/cjn.2021.241.

[9] Weng, N., Plomecka, M., Kaufmann, M., Kastrati, A., Wattenhofer, R., & Langer, N. (2023). *An Interpretable and Attention-based Method for Gaze Estimation Using Electroencephalography*, arXiv:2308.05768 2023, https://arxiv.org/abs/2308.05768.

[10] Arthur C. Grant, Samah G. Abdel-Baki, Jeremy Weedon, Vanessa Arnedo, Geetha Chari, Ewa Koziorynska, Catherine Lushbough, Douglas Maus, Tresa McSween, Katherine A. Mortati, Alexandra Reznikov, Ahmet Omurtag, EEG Interpretation Reliability and Interpreter Confidence: A Large Single Center Study. *Epilepsy Behav.*, 2014 Mar; 32:102-7, https://doi.org/10.1016%2Fj.yebeh.2014.01.011.

[11] Pan, Y., Laohathai, C., & Weber, D. J. (2021). The effectiveness of neurology resident EEG training for seizure recognition in critically ill patients. *Epilepsy & Behavior Reports*, 1-3, 15, 2021, https://doi.org/10.1016%2Fj.ebr.2020.100408.

[12] Ng, M. C., Jing, J., & Westover, M. B., A Primer on EEG Spectrograms. *J. Clin., Neurophysiol*., 2022 Mar 1, 39(3), pp.177-183, https://doi.org/10.1097%2FWNP.0000000000000736.

[13] Tawhid, M. N., Siuly, S., Wang, H., Whittaker, F., Wang, K., & Zhang, Y. (2021). A spectrogram image based intelligent technique for automatic detection of autism spectrum disorder from EEG. *PLoS ONE*, 16(6): e0253094. https://doi.org/10.1371/journal.pone.0253094.

[14] Khan, M. S., Salsabil, N., Alam, M. G., Dewan, M. A., & Uddin, M. Z., CNN-

XGBoost fusion-based affective state recognition using EEG spectrogram image analysis. *Scientific Reports*, (2022) 12:14122, https://doi.org/10.1038/s41598-022-18257-x.

[15] Biscione, V., & Bowers, J. S., Convolutional Neural Networks Are Not Invariant to Translation, but They Can Learn to Be, *Journal of Machine Learning Research*, 22 (2021) 1-28, [Online]. https://www.jmlr.org/papers/volume22/21-0019/21-0019.pdf (Accessed Date: May 30, 2024).

[16] Yan, J.; Li, J.; Xu, H.; Yu, Y.; Xu, T. Seizure Prediction Based on Transformer Using Scalp Electroencephalogram. *Appl. Sci.*, 2022, 12, 4158, https://doi.org/10.3390/app12094158.

[17] Lu, X., Wen, A., Sun, L., Wang, H., Guo, Y., & Ren, Y., An Epileptic Seizure Prediction Method Based on CBAM-3D CNN-LSTM Model, *IEEE Journal of Translational Engineering in Health and Medicine*, 11, pp.417-423, 2023, https://doi.org/10.1109%2FJTEHM.2023.3290036.

[18] Xiong, Z.; Wang, H.; Zhang, L.; Fan, T.; Shen, J.; Zhao, Y.; Liu, Y.; Wu, Q. A Study on Seizure Detection of EEG Signals Represented in 2D. *Sensors*, 2021, 21, 5145, https://doi.org/10.3390%2Fs21155145.

[19] Ilias, L., Askounis, D., & Psarras, J. (2023). Multimodal detection of epilepsy with deep neural networks, *Expert Systems with Applications*, 213(B), 2023, https://doi.org/10.1016/j.eswa.2022.119010.

[20] Benfenati, L., Unsupervised and Self-Supervised Machine-Learning for Epilepsy Detection on EEG Data, *Data Science and Engineering*, 2023, [Online]. https://webthesis.biblio.polito.it/27685/ (Accessed Date: May 30, 2024).

[21] García, F. P., & UCL., *Towards a data-driven treatment of epilepsy: computational methods to overcome low-data regimes in clinical settings*, Dept. of Medical Physics and Biomedical Engineering, University College London, 2023, [Online]. https://discovery.ucl.ac.uk/id/eprint/10164304/2/FernandoPerez-Garcia_PhD_thesis.pdf (Accessed Date: May 30, 2024).

[22] Park, S., & Medium.com. (2021). *Predicting the true probability in Neural Networks: Confidence Calibration*, [Online]. https://medium.com/codex/predicting-the-true-probability-in-neural-networks-confidence-calibration-fa6c6d712ff (Accessed Date: May 30, 2024).

[23] Wildberger, J., Siyuan Guo, A. B., & Schölkopf, B., *On the Interventional Kullback-Leibler Divergence*. arXiv:2302.05380, 2023, https://arxiv.org/abs/2302.05380v1

[24] Chen, J., Tam, D., Raffel, C., Bansal, M., & Yang, D., An Empirical Survey of Data Augmentation for Limited Data Learning in NLP. *Transactions of the Association for Computational Linguistics*, 2023; 11 191–211, https://doi.org/10.1162/tacl_a_00542.

[25] Maksimenko, Maksimenko, V.A., van Heukelum, S., Makarov, V.V. et al. Absence Seizure Control by a Brain Computer Interface. *Sci. Rep.*, 7, 2487 (2017), https://doi.org/10.1038/s41598-017-02626-y

[26] Tuncer, S. A., & Alkan, A., Classification of EMG signals taken from arm with hybrid CNN-SVM architecture. *Concurrency and Computation: Practice and Experience*, 34(5), pp.1-11, 2022, https://doi.org/10.1002/cpe.6746.

[27] Faust, O., Acharya, U. R., Adeli, H., & Adeli, A., Wavelet-based EEG processing for computer-aided seizure detection and epilepsy diagnosis, *Seizure*, 26, 56-64, 2015, https://doi.org/10.1016/j.seizure.2015.01.012.

[28] Shah, K., & Kaggle.com. (2020). *Data Augmentation Tutorial: Basic, Cutout, Mixup.*, [Online]. https://www.kaggle.com/code/kaushal2896/data-augmentation-tutorial-basic-cutout-mixup (Accessed Date: May 30, 2024).

[29] Jing, J., Lin, Z., Yang, C., Chow, A., Dane, S., Sun, J., & Westover, M. B. (2024). *HMS - Harmful Brain Activity Classification*, [Online]. https://kaggle.com/competitions/hms-harmful-brain-activity-classification (Accessed Date: May 30, 2024).

[30] Kim, B., & Seo, S., EfficientNetV2-based dynamic gesture recognition using transformed scalogram from triaxial acceleration signal. *Journal of Computational Design and Engineering*, 10(4), 1694–1706, 2023, https://doi.org/10.1093/jcde/qwad068.

[31] Tan, M., & Le, Q. V., EfficientNetV2: Smaller Models and Faster Training. *Proc. of the 38th International Conf on Machine Learning, PMLR*, 139, 2021, https://arxiv.org/pdf/2104.00298.pdf.

[32] Li, S., Wang, Z., An, Y., Zhao, J., Zhao, Y., & Zhang, Y.-D., EEG emotion recognition based

on the attention mechanism and pre-trained convolution capsule network. *Knowledge-Based Systems*, 265, 2023, https://doi.org/10.1016/j.knosys.2023.110372.

[33] Altman, N., & Krzywinski, M., The curse(s) of dimensionality. *Nature Methods*, 15, 399–400 2018, https://doi.org/10.1038/s41592-018-0019-x.

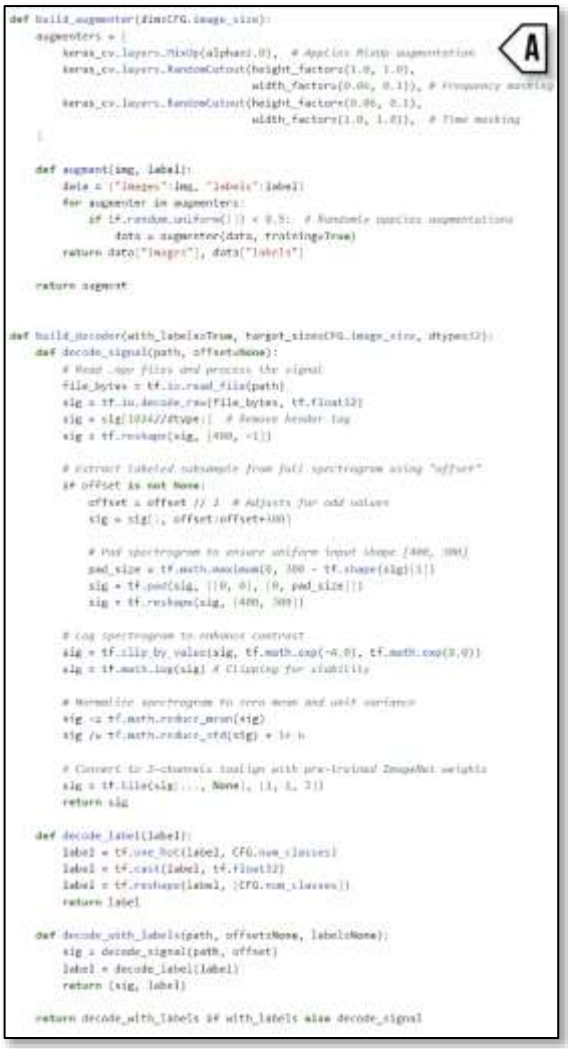
[34] Elstob, D., Secco, E.L, A low cost EEG based BCI Prosthetic using motor imagery, *International Journal of Information Technology Convergence and Services*, 6(1), 23-36, 2016.

[35] Chu, T.S., Chua, A.Y., Secco, E.L., Performance Analysis of a Neuro Fuzzy Algorithm in Human Centered & Non-Invasive BCI, *Lecture Notes in Networks and Systems*, 2, 241-252, 2021.

[36] Chu, T.S., Chua, A.Y., Secco, E.L., A Study on Neuro Fuzzy Algorithm Implementation on BCI-UAV Control Systems, *ASEAN Engineering Journal (AEJ)*, 12(4), 75-81, 2022, 10.11113/aej.v12.16900.

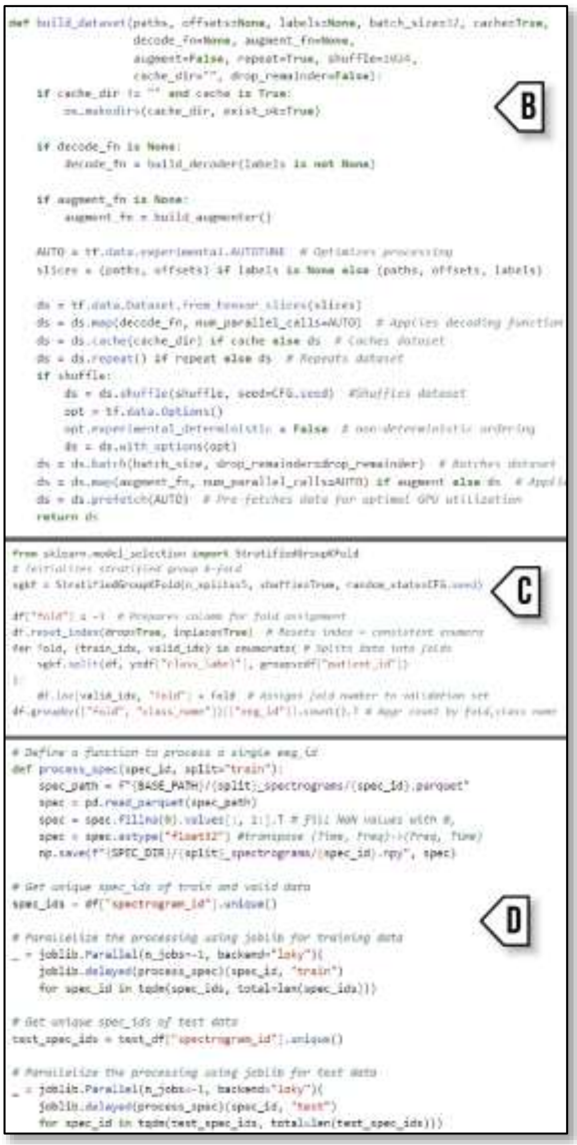
APPENDICES

Appendix 1



```
def build_augmenter(dim=CFG.image_size):
    augmenters = [
        keras_cv.layers.MixUp(alpha=1.0),  # Applies MixUp augmentation
        keras_cv.layers.RandomCutout(height_factor=(1.0, 1.0),
                                     width_factor=(0.06, 0.1)), # Frequency masking
        keras_cv.layers.RandomCutout(height_factor=(0.06, 0.1),
                                     width_factor=(1.0, 1.0)),  # Time masking
    ]

    def augment(img, label):
        data = {"images":img, "labels":label}
        for augmenter in augmenters:
            if tf.random.uniform([]) < 0.5:  # Randomly applies augmentations
                data = augmenter(data, training=True)
        return data["images"], data["labels"]

    return augment


def build_decoder(with_labels=True, target_size=CFG.image_size, dtype=32):
    def decode_signal(path, offset=None):
        # Read .npy files and process the signal
        file_bytes = tf.io.read_file(path)
        sig = tf.io.decode_raw(file_bytes, tf.float32)
        sig = sig[1024//dtype:]  # Remove header tag
        sig = tf.reshape(sig, [400, -1])

        # Extract labeled subsample from full spectrogram using "offset"
        if offset is not None:
            offset = offset // 2  # Adjusts for odd values
            sig = sig[:, offset:offset+300]

            # Pad spectrogram to ensure uniform input shape [400, 300]
            pad_size = tf.math.maximum(0, 300 - tf.shape(sig)[1])
            sig = tf.pad(sig, [[0, 0], [0, pad_size]])
            sig = tf.reshape(sig, [400, 300])

        # Log spectrogram to enhance contrast
        sig = tf.clip_by_value(sig, tf.math.exp(-4.0), tf.math.exp(8.0))
        sig = tf.math.log(sig) # Clipping for stability

        # Normalize spectrogram to zero mean and unit variance
        sig -= tf.math.reduce_mean(sig)
        sig /= tf.math.reduce_std(sig) + 1e-6

        # Convert to 3-channels to align with pre-trained ImageNet weights
        sig = tf.tile(sig[..., None], [1, 1, 3])
        return sig

    def decode_label(label):
        label = tf.one_hot(label, CFG.num_classes)
        label = tf.cast(label, tf.float32)
        label = tf.reshape(label, [CFG.num_classes])
        return label

    def decode_with_labels(path, offset=None, label=None):
        sig = decode_signal(path, offset)
        label = decode_label(label)
        return (sig, label)

    return decode_with_labels if with_labels else decode_signal
```

Fig. 11A: Augmentation (A)



```
def build_dataset(paths, offsets=None, labels=None, batch_size=32, cache=True,
                  decode_fn=None, augment_fn=None,
                  augment=False, repeat=True, shuffle=1024,
                  cache_dir="", drop_remainder=False):
    if cache_dir != "" and cache is True:
        os.makedirs(cache_dir, exist_ok=True)

    if decode_fn is None:
        decode_fn = build_decoder(labels is not None)

    if augment_fn is None:
        augment_fn = build_augmenter()

    AUTO = tf.data.experimental.AUTOTUNE  # Optimizes processing
    slices = (paths, offsets) if labels is None else (paths, offsets, labels)

    ds = tf.data.Dataset.from_tensor_slices(slices)
    ds = ds.map(decode_fn, num_parallel_calls=AUTO)  # Applies decoding function
    ds = ds.cache(cache_dir) if cache else ds  # Caches dataset
    ds = ds.repeat() if repeat else ds  # Repeats dataset
    if shuffle:
        ds = ds.shuffle(shuffle, seed=CFG.seed)  #Shuffles dataset
        opt = tf.data.Options()
        opt.experimental_deterministic = False  # non-deterministic ordering
        ds = ds.with_options(opt)
    ds = ds.batch(batch_size, drop_remainder=drop_remainder)  # Batches dataset
    ds = ds.map(augment_fn, num_parallel_calls=AUTO) if augment else ds  # Appli
    ds = ds.prefetch(AUTO)  # Pre-fetches data for optimal GPU utilization
    return ds
```

```
from sklearn.model_selection import StratifiedGroupKFold
# Initializes stratified group k-fold
sgkf = StratifiedGroupKFold(n_splits=5, shuffle=True, random_state=CFG.seed)

df["fold"] = -1  # Prepares column for fold assignment
df.reset_index(drop=True, inplace=True)  # Resets index = consistent enumeration
for fold, (train_idx, valid_idx) in enumerate( # Splits data into folds
    sgkf.split(df, y=df["class_label"], groups=df["patient_id"])
):
    df.loc[valid_idx, "fold"] = fold  # Assigns fold number to validation set
df.groupby(["fold", "class_name"])[["eeg_id"]].count().T # Aggr count by fold,class name
```

```
# Define a function to process a single eeg_id
def process_spec(spec_id, split="train"):
    spec_path = f"{BASE_PATH}/{split}_spectrograms/{spec_id}.parquet"
    spec = pd.read_parquet(spec_path)
    spec = spec.fillna(0).values[:, 1:].T # Fill NaN values with 0,
    spec = spec.astype("float32") #transpose (Time, Freq)->(Freq, Time)
    np.save(f"{SPEC_DIR}/{split}_spectrograms/{spec_id}.npy", spec)

# Get unique spec_ids of train and valid data
spec_ids = df["spectrogram_id"].unique()

# Parallelize the processing using joblib for training data
_ = joblib.Parallel(n_jobs=-1, backend="loky")(
    joblib.delayed(process_spec)(spec_id, "train")
    for spec_id in tqdm(spec_ids, total=len(spec_ids)))

# Get unique spec_ids of test data
test_spec_ids = test_df["spectrogram_id"].unique()

# Parallelize the processing using joblib for test data
_ = joblib.Parallel(n_jobs=-1, backend="loky")(
    joblib.delayed(process_spec)(spec_id, "test")
    for spec_id in tqdm(test_spec_ids, total=len(test_spec_ids)))
```

Fig. 11B: Data Cleaning & Conversion (B, D), Segmentation (C)

Appendix 2

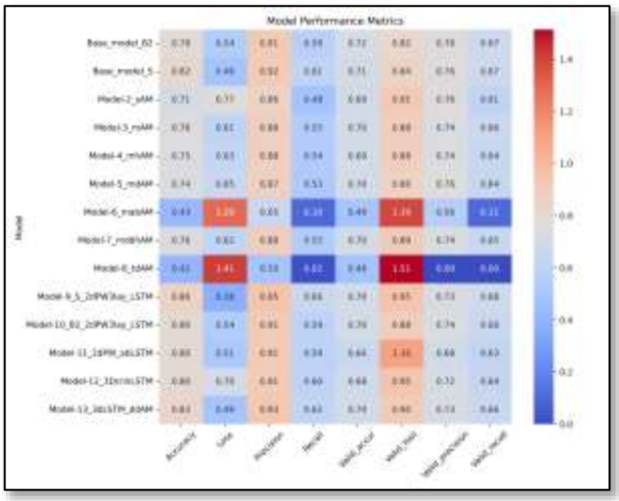
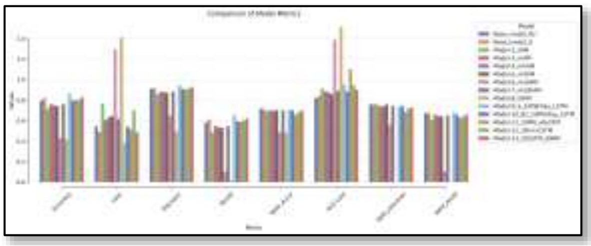


Fig. 12: Heatmap metrics results



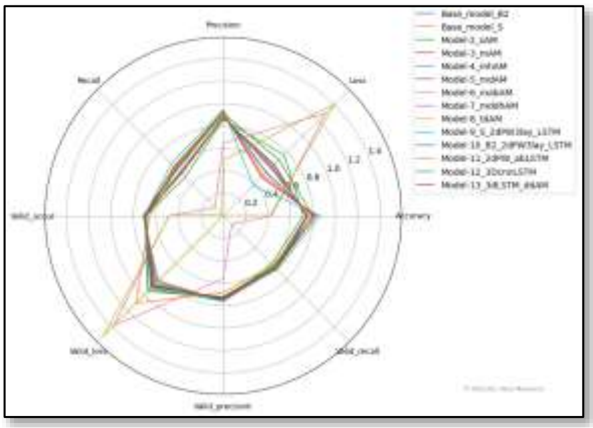


Fig. 13: Bars-chart (top panel) and Spiderweb-chart metrics comparison (bottom panel) between models

Contribution of Individual Authors to the Creation of a Scientific Article (Ghostwriting Policy)

DM and NB have conceived the classifier. DM has implemented the software, prepared the dataset and carried out all the data processing, as well as prepared the report with the description and results. NB has supervised the work. ELS has supervised the paper preparation.

Sources of Funding for Research Presented in a Scientific Article or Scientific Article Itself

No funding was received for conducting this study.

Conflict of Interest

The authors have no conflicts of interest to declare.