EEG Signal Classification Using Scalogram/Spectrogram based CNN

Rahul Dhiman (2001EE49), Aman Kumar (2001EE05)
Under Supervision of Dr.Sudhir Kumar
Indian Institue of Technology Patna

***Abstract*—This report describes a preliminary classification of EEG signals using convolutional neural networks (CNN) on scalograms and spectrograms. We analyzed sets of electroencephalographic (EEG) time series: surface EEG recordings from healthy volunteers with eyes closed and eyes open, and intracranial EEG recordings from epilepsy patients during the seizure free interval from within and from outside the seizure generating area as well as intracranial EEG recordings of epileptic seizures. Data contains five sets each containing 100 single channel EEG segments of 23.6 sec duration. The idea behind the project is to use convolutional neural networks on the features extracted from Continuous Wavelet Transform (CWT) and Short Time Fourier Transform (STFT) on the raw EEG signal and then classify it. The trained model achieved 90% accuracy using scalograms and 94% accuracy using spectrograms.**

I. Introduction

Epilepsy is a neurological condition that affects the central nervous system and can be challenging to visually detect. Seizures may cause an individual to experience lapses in attention or full-body convulsions. Frequent seizures can increase the risk of physical injury and may even be life-threatening. A device capable of quickly identifying and responding to seizures by delivering therapy or notifying a caregiver could potentially alleviate the burden of seizures. For that deep learning can be used to detect the surge of epileptic seizures.

There are various methods of detecting epileptic seizures in the early stages of the disease, with Neural Network (NN), Convolutional Neural Network (CNN), and Long Short-term Memory (LSTM) being among the most notable ones. We also came up with a classification technique using CNN on scalograms and spectrograms.

II. Dataset

The Data we used is the Bonn EEG dataset, it is a widely used benchmark dataset in the field of EEG-based brain computer interfaces (BCIs). It was developed at the University of Bonn, Germany, and contains EEG recordings from 5 healthy subjects and 5 patients with epilepsy. The recordings were made using a 40-electrode cap with a sampling rate of 173.61 Hz and include EEG signals from different mental states, such as relaxation, eyes-closed, eyes-open, and motor imagery.

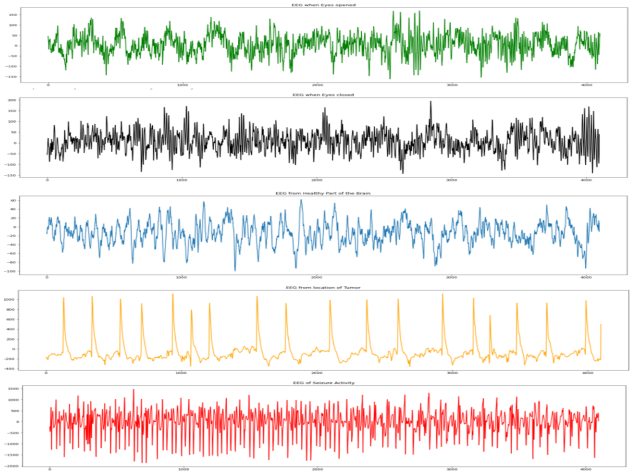
The data set consists of five sets, A, B, C, D, and E. The characteristics of each cluster are given in the Table below.

Different classes in Bonn EEG Datset

A	B	C	D	E
Epilepsy Patient	Epilepsy Patient	Epilepsy Patient	Healthy	Healthy
Total of 100 segments	Total of 100 segments	Total of 100 segments	Total of 100 segments	Total of 100 segments
Duration of each segment 23.6s	Duration of each segment 23.6s	Duration of each segment 23.6s	Duration of each segment 23.6s	Duration of each segment 23.6s
Record during the seizure	Pre-seizure, record from the epileptic area	Pre-Seizure, record from the hippocampal half sphere	Eyes closed recording	Eyes open recording

Fig. 1. Dataset

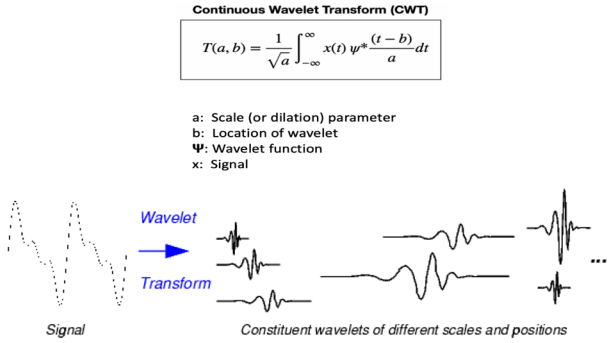
EEG recordings were taken using the 10–20 international electrode positioning system. Each cluster consists of 100 parts with a single channel of 23.6 s. The sampling rate of the filtered EEG signals is 173.61 Hz. But this data was not filtered through a low pass filter therefore we formed a new dataset by using 6th order Butterworth 0.53–40 Hz bandpass filter to filter the raw signals. The filtered sample signals for these five clusters are shown in the figure below. After that no further pre-processing was applied to the data sets.



III. Feature Extraction

A. *Continuous Wavelet Transform (CWT)*

Continuous Wavelet Transform (CWT) is a mathematical technique used in signal processing and data analysis to analyze non-stationary signals, which are signals that vary over time. The CWT decomposes a signal into a set of wavelet functions that are scaled and translated to analyze the different frequency components of the signal at different time intervals. Mathematically it is represented as:



We applied CWT transformation on the EEG data set using the "Morlet" Continuous Wave. After that each individual sample was converted into a scalogram and further into a scalogram image, a total of 500 images (100 each for 5 different classes) were included in the analysis.

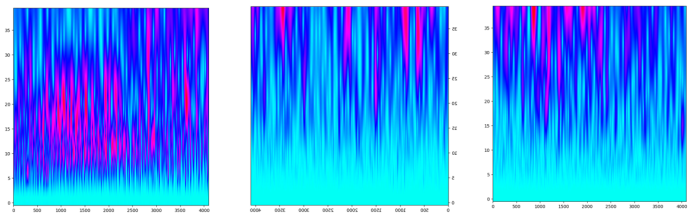
```
import numpy as np
import pywt

# Let signal = row of Dataframe
signal = df.iloc[n, :]
fs = 173.6

# Define the wavelet and its parameters
wavelet = 'morl'
scales = np.arange(1, 41)

# Compute the CWT
coef, freqs = pywt.cwt(signal, scales, wavelet, sampling_period = 1/fs)
```

Sample scalogram images –



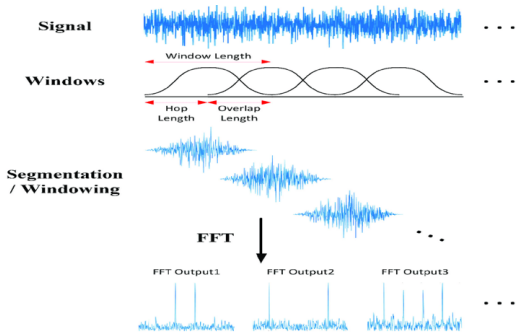
B. *Short Time Fourier Transform (STFT)*

Short Time Fourier Transform (STFT) is also a signal processing technique like CWT used to analyze non-stationary signals, which are signals that vary over time. The STFT breaks down a signal into a series of short-time windows and performs a Fourier Transform on each window to analyze its frequency components. The resulting spectrogram displays the frequency content of the signal as it changes over time.

The STFT is defined as follows:

$$\sum_{m=-\infty}^{\infty} X_m(\omega) \triangleq \sum_{m=-\infty}^{\infty} \sum_{n=-\infty}^{\infty} x(n)w(n-mR)e^{-j\omega n}$$
$$= \sum_{n=-\infty}^{\infty} x(n)e^{-j\omega n} \underbrace{\sum_{m=-\infty}^{\infty} w(n-mR)}_{1 \text{ if } w \in \mathrm{COLA}(R)}$$
$$= \sum_{n=-\infty}^{\infty} x(n)e^{-j\omega n}$$
$$\triangleq \mathrm{DTFT}_\omega(x) = X(\omega).$$

$x(n)$ = input signal at time n
$w(n)$ = length M window function (*e.g.*, Hamming)
$X_m(\omega)$ = DTFT of windowed data centered about time mR
R = hop size, in samples, between successive DTFTs.



The STFT coefficients are calculated over a series of time intervals, with each interval overlapping the previous one, to provide a continuous-time representation of the signal's frequency content. The STFT has several advantages over other time-frequency analysis techniques such as Fourier analysis, as it provides better time-frequency resolution and can handle non-stationary signals. However, the STFT has a trade-off between time and frequency resolution, which can be adjusted by choosing the length and overlap of the windows.

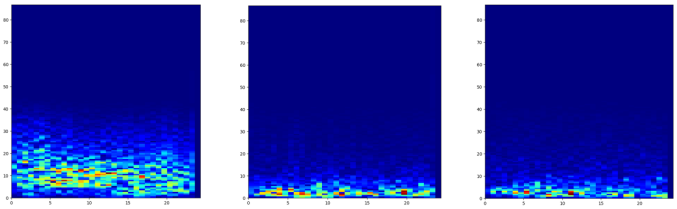
Again, we applied STFT to each sample to get 500 spectrogram images (100 each from 5 different classes).

```
import numpy as np
from scipy.signal import stft

# Let signal = row of Dataframe
signal = df.iloc[n, :]
fs = 173.6

# Compute the STFT
f, t, Zxx = stft(arr, fs = 173.6, window = 'hann', nperseg = 256, noverlap = None)
```

Sample spectrogram images –



C. *Resizing obtained images (Scalograms and Spectrograms)*

In this particular study, the frequency-time image was obtained by using the Continuous Wavelet Transform (CWT) on the raw EEG signals for each class. These images had dimensions of 662 by 536 (Default, depends on system). To generate input that could be used by a Convolutional Neural Network (CNN), the images were resized to 32 by 32 using cubic interpolation. Finally, these frequency-time images were fed as input to a CNN which is a widely used neural network architecture for image classification tasks. Also, the scalogram/spectrogram images formed were having 3 channels of RGB which was converted to single channel (RGB to GRAY).

IV. Model Architecture

A. *Convolutional Neural Network (CNN)*

A CNN (Convolutional Neural Network) layer is a fundamental building block of deep neural networks that are particularly useful for image recognition and processing. Convolutional, Pooling, and Activation layers are the three basic components of a CNN (Convolutional Neural Network).

Convolutional layer: The convolutional layer performs the convolution operation, which applies a set of filters to the input data to produce a set of feature maps. Each filter extracts a specific feature or pattern from the input data. The filter slides over the input data, computing the dot product between the filter and the local region of the input data at each location. The resulting output is a feature map that highlights the presence of the corresponding feature in the input data.

Pooling layer: The pooling layer down samples the output of the convolutional layer by reducing the spatial resolution of the feature maps while preserving the most important information. This is achieved by applying a pooling operation, such as max pooling or average pooling, to each local region of the feature maps. The resulting output is a set of pooled feature maps that are smaller in size than the original feature maps.

Activation layer: The activation layer applies a nonlinear activation function, such as ReLU (Rectified Linear Unit), to the output of the convolutional or pooling layer. This introduces nonlinearity to the network, enabling it to learn

more complex and abstract representations of the input data. The activation function applies a threshold to the output of the previous layer, producing an output value that is either activated (above the threshold) or not activated (below the threshold).

Together, these layers enable CNN to learn local features and hierarchical representations of the input data, making it particularly effective for image recognition and processing tasks.

B. Structure and Training of the Proposed CNN

This study was carried out in the Python environment by the Keras deep learning library. The obtained scalogram images were converted to pixel values in 32*32 array. 10% of the data is used as Test data and 20% of the remaining data set used for training is used as validation data. In the CNN structure we have 3 sets of 2 convolution layers followed by Batch Normalization layer, Pooling layer and a Dropout layer. After that we used Flatten layer followed by 2 dense layer having 1024 and 128 neurons and output layer having 5 neurons each representing different class.

Parameters for CNN Model-1 (Scalogram)

Layer	Filter Size	Number of Filters	Number of Neurons	Stride
Conv-1	7×7	16	-	1
Conv-2	7×7	16	-	1
BatchNorm	-	-	-	-
MaxPooling	-	-	-	2
Dropout	-	-	-	-
Conv-3	5×5	32	-	1
Conv-4	5×5	32	-	1
BatchNorm	-	-	-	-
MaxPooling	-	-	-	2
Dropout	-	-	-	-
Conv-5	3×3	64	-	1
Conv-6	3×3	64	-	1
BatchNorm	-	-	-	-
MaxPooling	-	-	-	2
Dropout	-	-	-	-
Flattened	-	-	-	-
Dense	-	-	1024	-
Dropout	-	-	-	-
Dense	-	-	128	-
Dropout	-	-	-	-
Dense	-	-	5	-

For the obtained spectrogram images, we first converted to pixel values in 32*32 array. 10% of the data is used as Test data and 20% of the remaining data set used for training is used as validation data. In the CNN structure we have 2 sets of 2 convolution layers followed by Batch Normalization layer, Pooling layer and a Dropout layer. After that we used Flatten layer followed by 2 dense layer having 1024 and 128 neurons and output layer having 5 neurons each representing different class.

For CNN, the learning rate is 0.001, the optimizer used is RMSprop, the epoch number is set to 200 and the appropriate batch size is 4. Also, we used learning rate reduction as Reduce LR On Plateau. EEG scalogram images were divided into 10 equal parts in the CNN structure, 9 parts of these parts were used as training and the remaining 1 were used as test data. In order to avoid overfitting, 20% of the training data was allocated as validation data.

Parameters for CNN Model-2 (Spectrograms)

Layer	Filter Size	Number of Filters	Number of Neurons	Stride
Conv-1	7×7	16	-	1
Conv-2	7×7	16	-	1
BatchNorm	-	-	-	-
MaxPooling	-	-	-	2
Dropout	-	-	-	-
Conv-3	3×3	64	-	1
Conv-4	3×3	64	-	1
BatchNorm	-	-	-	-
MaxPooling	-	-	-	2
Dropout	-	-	-	-
Flattened	-	-	-	-
Dense	-	-	1024	-
Dropout	-	-	-	-
Dense	-	-	128	-
Dropout	-	-	-	-
Dense	-	-	5	-

C. Performance Evaluation

We have used accuracy metrics with Categorical Cross entropy loss function because of multiclass classification. Besides that, we also plotted confusion matrix on test data. And calculated Accuracy, Recall and F1 score.

Confusion Matrix: A confusion matrix is a table used in machine learning and statistics to evaluate the performance of a classification model. It is used to show the number of true positive (TP), false positive (FP), true negative (TN), and false negative (FN) predictions made by a classification model.

Accuracy:The confusion matrix is commonly used in calculating accuracy, which is a measure of how well a classification model correctly identifies both positive and negative samples.

$$\textbf{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}}$$

Recall:The confusion matrix is also used in calculating recall, which is a measure of a classification model's ability to correctly identify positive samples out of all the actual positive samples.

$$\text{Recall} = \frac{TP}{TP + FN}$$

Precision:The confusion matrix is also used in calculating precision, which is a measure of a classification model's ability to correctly identify positive samples out of all the samples that the model has classified as positive.

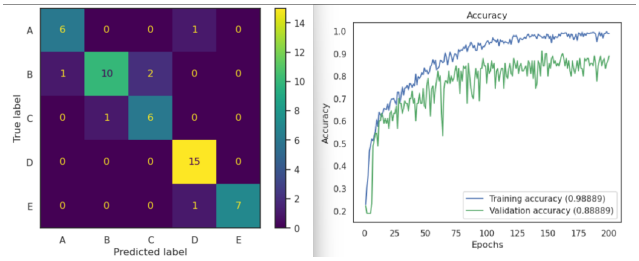
$$\text{Precision} = \frac{TP}{TP + FP}$$

F1 Score:The confusion matrix is also used in calculating the F1 score, which is a metric that combines both precision and recall into a single score. The F1 score provides a balance between the two metrics and is useful when we want to consider both false positives and false negatives in the evaluation of a classification model.

$$F_1\text{−score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} = \frac{2\text{TP}}{2\text{TP} + \text{FP} + \text{FN}}$$

V. Results

A. *Model-1*

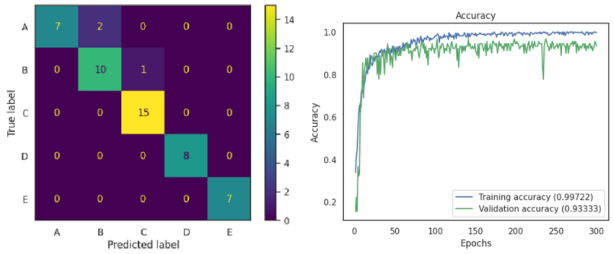



```
from sklearn.metrics import accuracy_score, recall_score, f1_score

print("Accuracy = ", accuracy_score(actual, predicted))
print("Recall = ", recall_score(actual, predicted,  average = None))
print("F1_Score = ", f1_score(actual, predicted, average = None))
```

```
Accuracy =  0.9
Recall =  [0.85714286 0.69230769 1.         1.         1.        ]
F1_Score =  [0.85714286 0.81818182 0.82352941 0.96774194 1.        ]
```

B. *Model-2*




```
[48] from sklearn.metrics import accuracy_score, recall_score, f1_score

     print("Accuracy = ", accuracy_score(actual, predicted1))
     print("Recall = ", recall_score(actual, predicted1,  average = None))
     print("F1_Score = ", f1_score(actual, predicted1, average = None))
```

```
Accuracy =  0.94
Recall =  [0.77777778 0.90909091 1.         1.         1.        ]
F1_Score =  [0.875      0.86956522 0.96774194 1.         1.        ]
```

VI. Conclusion

This project basically aims to show that despite using conventional complex models or deep neural networks on the raw EEG signals, there is also a way in analyzing the scalograms or spectrograms of the given EEG signal. Our model achieved good accuracy of around 90% and 94% by using CNN on scalograms and spectrograms by using some additional layers of Batch Normalization and Dropout layers. This can be further improved by using this technique for binary classification.

References

[1] Rashed-Al-Mahfuz, M., Moni, M. A., Uddin, S., Alyami, S. A., Summers, M. J., Eapen, V. (2021). A Deep Convolutional Neural Network Method to Detect Seizures and Characteristic Frequencies Using Epileptic Electroencephalogram (EEG) Data. IEEE Journal of Translational Engineering in Health and Medicine.

[2] The Bonn EEG time series download page. (n.d.). Nonlinear Time Series Analysis.

[3] An automated classification of EEG signals based on spectrogram and CNN for epilepsy diagnosis Article Full-t.

[4] M. K. M. Rabby, R. B. Eshun, S. Belkasim and A. K. M. K. Islam, "Epileptic Seizure Detection Using EEG Signal Based LSTM Models," 2021 IEEE Fourth International Conference on Artificial Intelligence and Knowledge Engineering (AIKE), Laguna Hills, CA, USA, 2021, pp. 131-132