Student names: Peichen Liu
SCIPER: 375504

Electroencephalography based epileptic seizure detection

Abstract

Our project used a new framework combining data standardization, annotation extraction, model training for machine learning and deep learning, and performance visualization. In our project, we test 4 different algorithms in 2 datasets(SIENA and CHBMIT) in the new framework including the Feature-based algorithm, CNN algorithm, Lightweight CNN algorithm, and Transformer algorithm. We also standardize the input and output. All of these algorithms run smoothly in the new framework. Actually, we also tested for the SeizIT1 dataset, but it is hard to get a good performance of the SeizIT1 dataset, so, we didn't show the SeizIT1 result in this report. We published the code in the GitHub `https://github.com/Peich-Liu/Epileptic_Seizure_Project`

1 Introduction

Epilepsy affects 50 million people worldwide, with an estimated 2–3 million living in the United States, 6 million in Europe, and at least 40 million in the developing world.Galanopoulou et al. (2012) Sudden and abrupt seizures that cause momentarily lapses of consciousness can have a significant impact on the daily life of sufferers, it may even put the patient's life at risk. The seizure will also cause some unforeseeable side effects, such as memory loss, depression, and other psychological disordersChen et al. (2020b). So, it is important to monitor the seizures of the patients. However, the seizure is hard to predict and it will cause such bad effects, therefore, using EEG signals and machine learning/deep learning algorithms to predict and monitor epileptic seizures is a popular topic in different research units.

Both feature-based machine learning and deep learning have potential in the seizure detection area. But feature-based machine learning and deep learning have different main problems. The main problem in feature-based machine learning is the feature extraction process and training process, and the main problem in deep learning is the data splitting and training part, so, we want to make a new frame to combine the feature-based machine learning and deep learning. The users can use different commands to do the tasks they need.

In the previous research, Chen et al. (2020a) proposed a framework for the combination of seizure detection and diagnosis. However, they did not focus on the standardization and evaluation. Winterhalder et al. (2003) also proposed a framework for the evaluation of seizure detection, but, they focus on the single signal evaluation. So, we want to propose a framework that includes the data standardization, the training, validation, and testing of machine learning and deep learning algorithms, and the evaluation and visualization of the different algorithms for the whole different dataset.

Therefore, in this project, we designed a framework that is suitable for different algorithms and different datasets. Firstly, we use a new function that includes a standardized list of channels, sampling frequency, and data format. Secondly, we used three different algorithms with these formalized data to train the different models and did the cross-validation of the different models. Finally, we use the new evaluation function to evaluate the performance of the different algorithms.

2 Literature Review

To check how far we are in epilepsy seizure detection, we performed a systematic literature review to understand the status quo. To understand the status quo of seizure detection, we resort to a set of keywords to search for relevant publications in popular repositories. The keywords are: *epilepsy seizure detection algorithm*, *Deep learning*, *machine learning*, *EEG*. The algorithm of seizure detection could be divided into whether they need feature extraction.

The first part is the feature-based algorithm, Temko et al. (2011) used multiple SVM classifiers to analyze the feature vectors of a single channel. Next, the output of the SVM is smoothed by a moving average method, and a threshold judgment is applied to determine whether there is a possibility of epileptic seizure. These binary decision vectors are finally combined through a logical OR operation, and the final output result is produced after a decision process. Moreover, Seiffert et al. (2010) used RusBoost to do the classification tasks, RusBoostMarín et al. (2021) is an algorithm that is focused on the imbalanced dataset which is very suitable for epileptic seizure detection.

For the non-feature-based algorithms, we will also divide them into a time-series-focused algorithm and a non-time-series-focused algorithm. In the time-series-focused algorithm, Hu et al. (2020) proposed an algorithm using the Bidirectional Long Short-Term Memory Network (Bi-LSTM). They used the simple LSTM algorithm to gain a good performance. Potter et al. (2022) proposed a transformer-based unsupervised learning method, which uses the unsupervised learning method to strengthen the performance of the imbalanced data.

There are also several researches using the non-time-series-focused algorithm to do the seizure detection. Zhou et al. (2018) introduced a method based on CNN to process EEG signals to detect epileptic seizures. This method does not rely on manual feature extraction but directly uses raw EEG signals. The effectiveness of this method in two-classification and three-classification problems was verified by analyzing the time domain and frequency domain of signals in the intracranial Freiburg database and scalp CHB-MIT database. Thuwajit et al. (2022) proposed an algorithm in 2022 that uses a CNN algorithm. Compared with the normal CNN algorithm, they use a Multiscale Convolution before the convolution layer which has a good performance in the different datasets.

3 Background

3.1 EEG Montages

From the individual discharges detected by electrodes, EEGs give rise to a rich set of data across the entire span of the scalp. This is done via connecting all the electrodes in what is termed a montage, and there are two broad categories: bipolar and referential. Valentine

3.2 Double Banana

Double Banana is an EEG signal-collected method. There are two chains per side in the double banana (from which it derives its name): an outside temporal chain involving Fp2→F8→T4→T6→O2, and then an inside parasagittal chain involving Fp2→ F4→C4→P4→O2. The "z" electrodes Fz→Cz→Pz form a small central chain. The Double banana picture is shown in Fig1 Valentine.

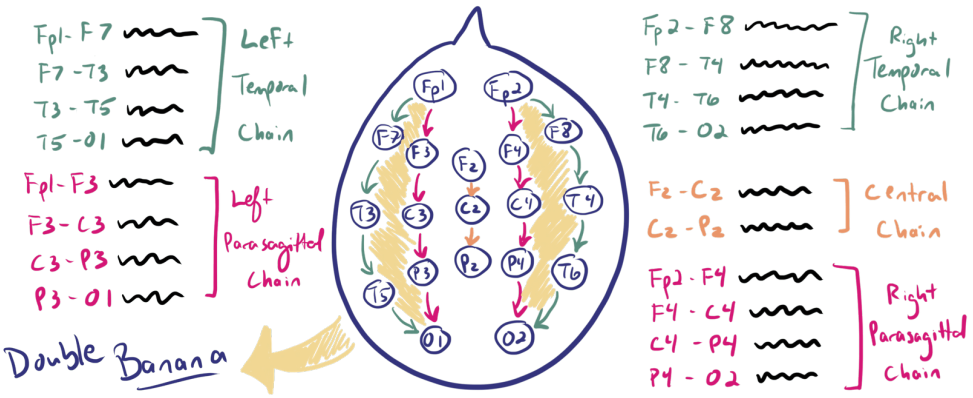

Figure 1: Double Banana

3.3 Cross Validation

Cross-validation is a statistical method used to evaluate and improve the performance of a model on independent data sets. This approach provides a more comprehensive assessment of model performance by splitting the data into small subsets and iteratively training and validating the model. We used Kfolder validation and leave-one-out cross-validation in the project.

Kfloder Validation: In k-fold cross-validation, the available learning set is partitioned into k disjoint subsets of approximately equal size. Here, fold refers to the number of resulting subsets. This partitioning is performed by randomly sampling cases from the learning set without replacement. The model is trained using k-1 subsets, which, together, represent the training setBerrar et al. (2019).

Leave-one-out cross-validation: For k = n, we obtain a special case of k-fold cross-validation, called leave-one-out cross-validation (LOOCV). Here, each individual case serves, in turn, as hold-out case for the validation set. Thus, the first validation set contains only the first case, x1, the second validation set contains only the second case, x2, and so onBerrar et al. (2019).

3.4 Performance Metrics

In classification tasks, performance evaluation usually involves the understanding of the following four basic concepts:
True Positive (TP): The number of times the model correctly classified a positive example as a positive example.
False Positive (FP): The number of times the model incorrectly classified a negative example as a positive example.
True Negative (TN): The number of times the model correctly classified a negative example as a negative example.
False Negative (FN): The number of times the model incorrectly classified a positive example as a negative example.
Sensitive: The measurement metric can identify positive examples, calculated as Eq1

$$TP/(TP + FN) \tag{1}$$

Precision: is the proportion of True positive samples among all positive samples, calcu-

lated as Eq2

$$TP/(TP + FP) \tag{2}$$

F1-score: is the harmonic mean of precision and sensitivity, calculated as Eq3

$$2 * (Precision * Sensitivity)/(Precision + Sensitivity) \tag{3}$$

Fp-rate: It can measure how often a model incorrectly labels negative examples as positive, calculated as Eq4

$$FP/(FP + TN) \tag{4}$$

4 Methodology

4.1 Design Framework

The main body of the software can be divided into three parts: preprocessing data, train-test processing, and evaluation. The workflow of the framework is shown in Fig2.

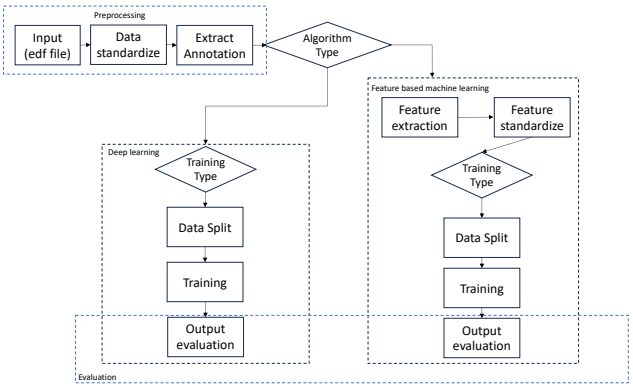

Figure 2: Framework of the system

The detail of data preprocessing is shown in paragraph 4.3, the detail of the algorithm and train-test is shown in paragraph 4.4, and the evaluation part is shown in paragraph 4.6

4.2 Standardised Input and Output

To ensure the reproducibility of our experiments, we provide different configuration files (conf) for different algorithms that define all relevant parameters, as well as a command line interface that allows users to specify basic running configurations. Our framework supports the command-based running. Users can use this format's command to run the different algorithms with different validation types and datasets. The example of the command is shown in Fig3 There are several parameters in the conf files for the different algorithms. The parameters include the window parameters, the channel parameters, and the feature-based machine learning parameters. Window parameters include the window length and the window step size. Channel parameters include the montage electrode and normalization. Feature-based machine parameters include the name of the features and normalization of the features.

```
python --algorithm CNN --dataset CHBMIT --trainType Kfolder
```

Figure 3: Example of command line usage

4.3 Data Acquisition and Standardize

The preprocessing data included the file load and the data standardized. At first, in our framework, we have two different standardized functions bipolar montages and referential montages. The user of the framework can choose the different standardized functions in the different tasks. Secondly, the data after standardizing needs to be cut to different windows.

4.4 Algorithms

To validate the correctness of the framework, we realized 3 different algorithms in this section. The algorithms include feature-based machine learning (4.4.1) and deep learning (4.4.2 and 4.4.3).

4.4.1 Feature Based Algorithm

In the feature-based algorithm, we used the RusBoost as the algorithm.Marín et al. (2021). The algorithm is aimed to train the imbalanced data.Seiffert et al. (2010) RusBoost is an algorithm that focuses on the imbalance of data. The detail of the algorithm is shown in Algorithm0.

Algorithm 1 RUSBoost(Seiffert et al. (2010))

Given: Set S of examples $(x_1, y_1), \ldots, (x_m, y_m)$ with minority class $y' \in Y, |Y| = 2$
Weak learner, *WeakLearn*
Number of iterations, T
Desired percentage of total instances to be represented by the minority class, N
Initialize $D_1(i) = \frac{1}{m}$ for all i.
for $t = 1, 2, \ldots, T$ **do**
Create temporary training dataset S'_t with distribution D'_t using random undersampling
Call *WeakLearn*, providing it with examples S'_t and their weights D'_t.
Get back a hypothesis $h_t : X \times Y \to [0, 1]$.
Calculate the pseudo-loss (for S and D_t):
$\epsilon_t = \sum_{(i,y):y_i \neq y} D_t(i)(1 - h_t(x_i, y_i)) + h_t(x_i, y)$
Calculate the weight update parameter:
$\alpha_t = \frac{\epsilon_t}{1-\epsilon_t}$.
Update D_t:
$D_{t+1}(i) = D_t(i) \left(\frac{1}{2}\right)^{1+h_t(x_i,y_i)-h_t(x_i,y_i \neq y)}$
Normalize D_{t+1}: Let $Z_t = \sum_i D_{t+1}(i)$.
$D_{t+1}(i) = \frac{D_{t+1}(i)}{Z_t}$.
end for
Output the final hypothesis:
$H(x) = argmax_{y \in Y} \left(\sum_{t=1}^{T} h_t(x, y) \log \frac{1}{\alpha_t} \right)$.

Feature in the algorithm: We use the same feature in the algorithm in the paper Marín et al. (2021). The features are Standard deviation, Mean absolute deviation with respect to the median, Skewness, Second-order difference plot, Katz fractal dimension, Mean network degree, Mean network betweenness, and Mean network closeness. The details of the features are shown in the appendix4.4.1.
Our Fine-tuning: In their paper, they only use the signal channel to evaluate the performance of the algorithm, and they use the different channels for the different patients. We want to test what is the algorithm performance in a general model that uses the same channel and the same test threshold.

4.4.2 CNN

In the deep learning part, we used the CNN as the traditional CNN algorithm as the test.Thuwajit et al. (2022). The network architecture is shown in Fig52 in the appendix. This algorithm proposed an EEGWaveNet to test the epileptic seizure. The network contains multiple Depthwise Convolution layers, and the number of output channels of each layer gradually increases from 16 at the top to 1024. These layers are able to extract features while keeping the number of parameters low. The middle of the network consists of standard Conv2d and BatchNorm2d alternately. Each convolutional layer is followed by a batch normalization layer, and all convolutional layers appear to have the same number of output channels (32) and a kernel size of 1x4. The output part of the network is each convolutional layer followed by a Global Average Pooling, and finally, a softmax layer is used to classify the data and output two categories: normal and epileptic seizure.

4.4.3 Transformer

Transformer uses the deep learning network structure of the unsupervised Transformer encoder to process multivariate time series data.Potter et al. (2022) The structure can be divided into the following parts:
Multivariate input: The input layer receives time series data from multiple channels.
Positional encoding: After the input layer, positional encoding is added to preserve the sequential information in the time series.
Affine dimensionality reduction: The positionally encoded data is reduced in dimensionality through an affine transformation layer to obtain latent features.
Transformer encoder: The latent features are fed into the transformer encoder, which consists of multiple layers, each layer includes: **Multi-head self-attention mechanism**: allows the model to focus on different parts of the input at different locations at the same time. **Batch normalization and fully connected neural network**: standardizes, stabilizes, and accelerates the training process. Provides additional nonlinear processing capabilities. **Repeated transformer layers**: The above transformer encoding layer is repeated multiple times to improve the model's learning ability.
Affine Transformation: The output of the encoder is passed through another affine transformation layer to prepare the final output.
Output layer: The final output layer produces reconstructed time series data.

The architecture detail is shown in Fig53 in the appendix.

Our fine-tuning: The original paper they published uses different window lengths for the different datasets, the window length and step are based on the shortest seizure window in the datasets. But we use the same length window for every different dataset

to standardize the window. We want to use this standardized method to validate the performance of the algorithm.

4.4.4 Light Weight Deep Learning with CNN

In Huang et al. (2023) paper, they proposed a lightweight method to detect seizures, they tried to use the proportional train-test data to train the model. And they got a good performance in the different datasets. They use the method based on the random forest, we want to test the performance in the deep learning algorithm, so, we test this algorithm with the CNN algorithm, we train the model on a training set three times the size of the test set.

4.5 Annotation Extraction

The different publicly available datasets all use a different format to store seizure annotation information. Some of them use a binary format, others text based with different ways of encoding the information (csv, tsv, txt, ...). Here we propose a standardized format for seizure annotations of EEG recordings along with scripts to convert the formats from the different datasets to this standardized format. The format is used both for loading annotations from datasets and as the output of seizure detection algorithms. The details of this part are already shown on the ESL website.

4.6 Evaluation and Visualization

4.6.1 Score of seizure detection

Hypnosis and Reference: Hypnosis is the annotation of a seizure detection algorithm. Reference is the annotations from an expert neurologist. So, the validation processing is the processing to compare the hypnosis and reference.
Sample-based Score: Sample-based score is that only judges consistency between the hypnosis windows and reference windows
Event-based Score: To evaluate the performance of our model, we define a set of event-based performance metric parameters that provide control over temporal accuracy when dealing with onset prediction:

tolerance before: This is the tolerance time window before the seizure starts, in seconds. If a false positive prediction (FP) occurs within this time frame, it will not be counted. We allow 30 seconds.

tolerance after: Similarly, this is the tolerance time window after the episode ends, also used to control false positives. We allow 60 seconds

minimum overlap: Specifies the minimum relative overlap ratio required between predicted and true onsets in order for the two to be considered a match. In our case, this minimum overlap ratio is set to 0.

max Event Duration: Defines the maximum duration of a single event in seconds. Any event exceeding this duration will be automatically split. We set this threshold to 300 seconds.

min Duration Between Events: If the time between two consecutive events is less than the number of seconds specified by this parameter, they will be automatically merged. In our setup, this interval is set to 90 seconds.

In addition, we also calculated the final prediction frequency predictionFreq, which is based on the reciprocal of the feature parameter window step and represents how often the model makes predictions.

4.6.2 Improve signal prediction

We use two different functions to improve its accuracy and reliability. The first function is Moving Average Smoothing. This method smoothes the classifier's predictions by averaging the labels over a set window length. When the average label value within a window exceeds the set threshold, the final output label of the window is determined to be 1, otherwise, it is 0. The second function is the Bayesian Smoothing, which uses predicted probabilities to calculate the cumulative probability of an event occurring within a time window. The final label is determined by calculating the logarithmic ratio between the cumulative product of positive probabilities and the cumulative product of negative probabilities within the window and comparing it to a preset threshold. If this ratio exceeds the threshold, an event is considered to have occurred within that window.

4.6.3 Individual Evaluation

In the individual evaluation, we use the time series classifier performance graph to show the performance of the different individual patients. The graph shows the True Positive, False Positive, False Negative, and the Sensitive, Precision, F1-score for the individual patient. Meanwhile, the individual visualization will show the three different signal processing methods including the original signal, moving average post-processed signal, and Bayesian post-processed signal. The example of the individual evaluation is shown in Fig4 in paragraph 5.3.1.

4.6.4 General Evaluation

The general evaluation includes 2 figures. First, we use box plots to show the performance distribution of different evaluation metrics. The boxplot intuitively reflects the prediction performance of the model through the median line, the interquartile range of the box, and outlier points. Moreover, we use the line chart to integrate every individual result. The results of these two plots include the sensitive, precious, F1 score of the event-based and sample-based evaluation. In the line chart, we also show the average of the F1 score of sample-based and event-based evaluation and the event-based false positive rate. These figures are shown in paragraph5.3

5 Experiment

5.1 Datasets

In our experiment, we use 3 datasets to test the correctness of our framework. The datasets are CHB-MITShoeb (2009), Siena Scalp EEG. The parameters of these datasets are shown in Table1

Table 1: Dataset detail

Dataset	subjects	duration [h]	seizures
CHB-MIT	24	982	198
Siena Scalp EEG	14	128	47

CHB-MIT Dataset: The first dataset we used is the CHB-MIT dataset which was collected from Children's Hospital Boston. Subjects were monitored for up to several days following withdrawal of anti-seizure medication in order to characterize their seizures and assess their candidacy for surgical intervention.Shoeb (2009).

Siena Scalp EEG: The second dataset is Siena Scalp EEG. The database consists of EEG recordings of 14 patients acquired at the Unit of Neurology and Neurophysiology of the University of Siena. Subjects include 9 males (ages 25-71) and 5 females (ages 20-58)Detti (2020).

5.2 Data Preprocessing

5.2.1 Data Standardize

We use the different channels of information in the different datasets to validate the correctness of our framework. There are two different. We used the Bipolar montage in the CHB-MIT dataset and use the Unipolar montage in the SIENA dataset.
The Biploar montage we used in **CHB-MIT** are $Fp1-F3$, $F3-C3$, $C3-P3$, $P3-O1$, $Fp1-F7$, $F7-T3$,$T3-T5$, $T5-O1$, $Fz-Cz$, $Cz-Pz$, $Fp2-F4$, $F4-C4$, $C4-P4$, $P4-O2$, $Fp2-F8$, $F8-T4$, $T4-T6$, $T6-O2$.
The Biploar we used in **SIENA** are $Fp1$, $F3$, $C3$, $P3$, $O1$, $F7$, $T3$, $T5$, Fz, Cz, Pz, $Fp2$, $F4$, $C4$, $P4$, $O2$, $F8$, $T4$, $T6$ and the reference electrode is Cz.

5.2.2 Window

The window sizes are different in the different algorithms. The parameters of the windows' size in different algorithms are shown in Table2

Table 2: Window Parameter in Algorithms

	Window Size	Step Size	Sample Frequency
RusBoost	0.391s	0.391s	256Hz
CNN	4s	1s	256Hz
Unsupervised Transformer	4s	4s	256Hz
CNN with Light Weight	4s	1s	256Hz

5.3 Training Results

5.3.1 Individual Result example

Fig4 is an example of the individual performance example. The figure shows the different signal processing methods for the PN00 in SIENA with the algorithm RusBoost. The green dots in the figure show the true positive windows, the red dots show the false

positive windows and the purple dots show the false negative windows. It shows the different results in sample-based and event-based. And the figure also shows the sensitivity, precision, and F1 score.

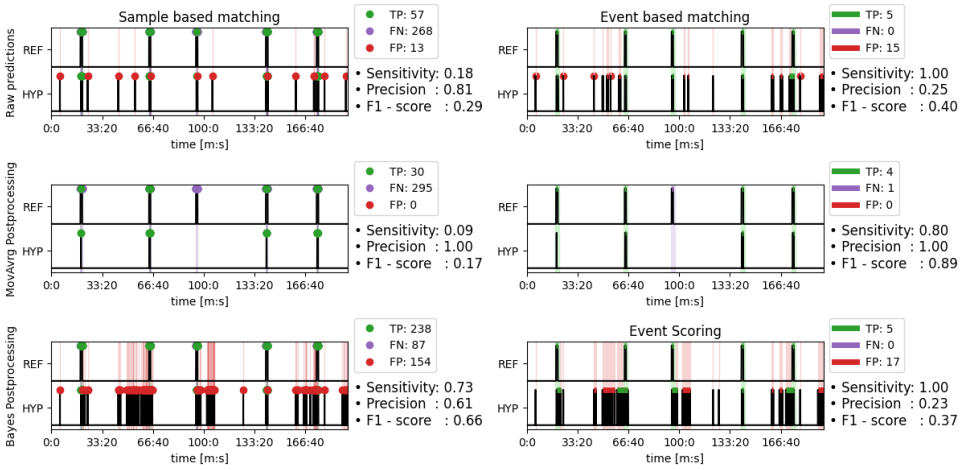

Figure 4: Example of the individual result

We also generate the ROC curve for every individual patient in every different algorithm and cross-validation, there is an example for the individual ROC curve in Fig5. They will be stored in the same folder with the individual results.

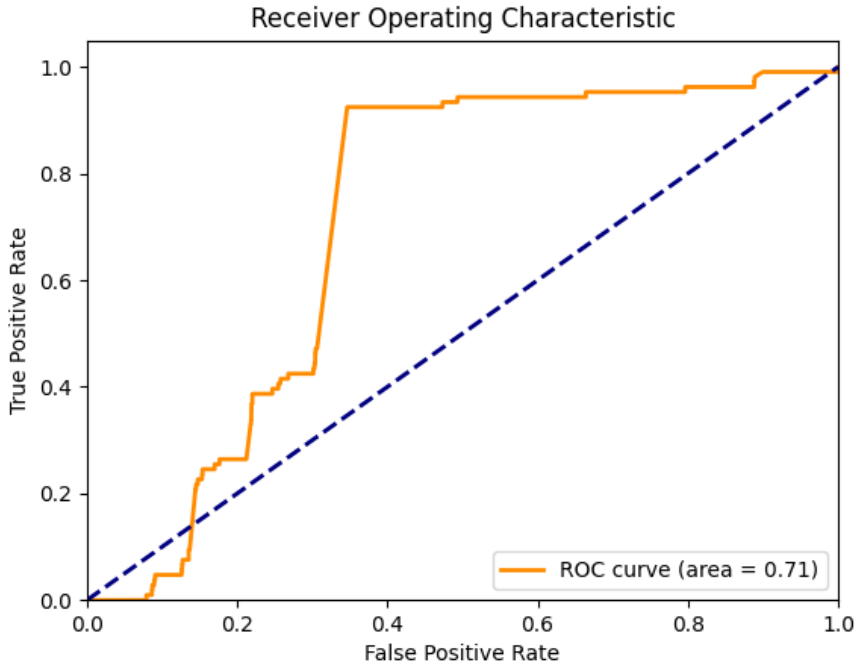

Figure 5: Example of the individual ROC

5.3.2 Results in RusBoost

In the RusBoost algorithm, we use three different cross-validation methods(Leave-one-out, Kfolder, Personal). The leave-one-out Box plot and line-chart are shown in Fig41

and Fig17 and other validation results are shown in appendix 4

Evaluate figure: boxplot, validation type: Leave-one-out:

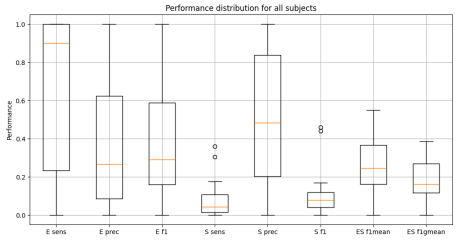

Figure 6: Dataset: SIENA

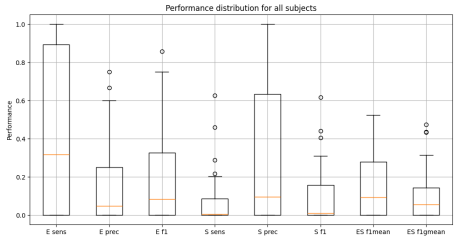

Figure 7: Dataset: CHBMIT

Figure 8: Boxplot figure, leave-one-out,RusBoost

Algorithm:RusBoost, Evaluate figure: line chart, validation type: Leave-one-out:

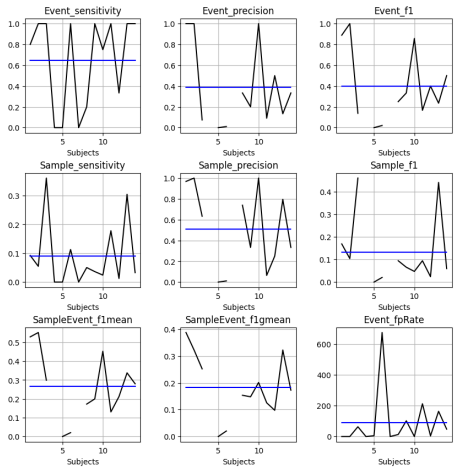

Figure 9: Dataset: SIENA

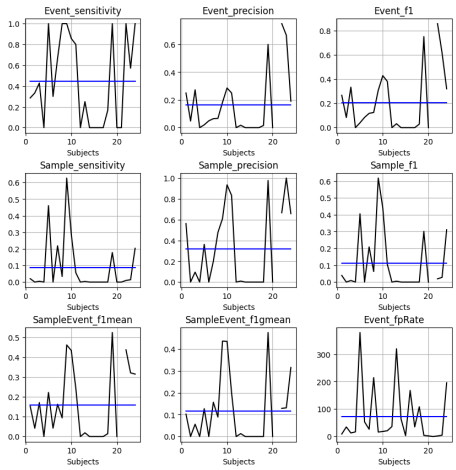

Figure 10: Dataset: CHBMIT

Figure 11: Line figure, leave-one-out, RusBoost

5.3.3 Results in CNN

In the algorithm CNN, we just use the Kfolder to validate the model because the data is too large to validate in the other methods.
Evaluate figure: box plot, validation type: Kfolder, algorithm: CNN:

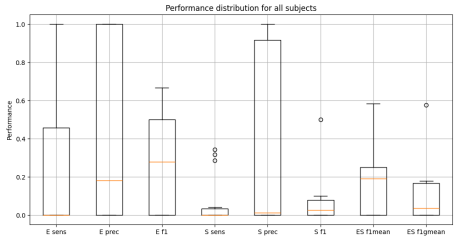

Figure 12: Dataset: SIENA

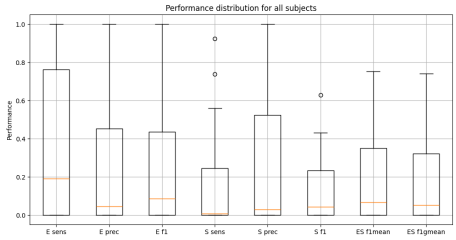

Figure 13: Dataset: CHBMIT

Figure 14: Boxplot, Kfolder, CNN

Evaluate figure: line chart, validation type: Kfolder, algorithm: CNN:

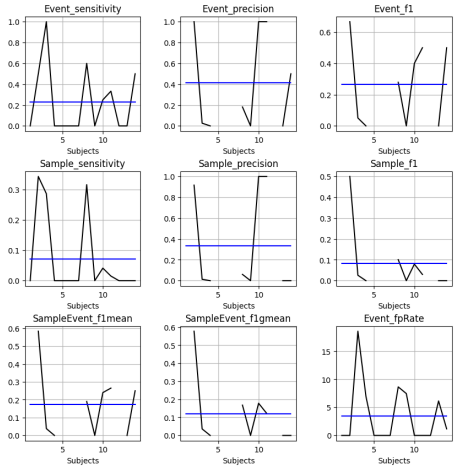

Figure 15: Dataset: SIENA

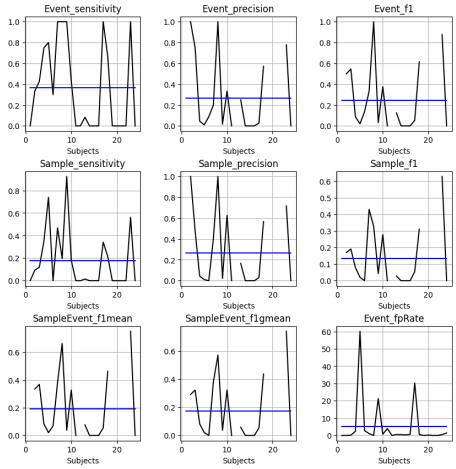

Figure 16: Dataset: CHBMIT

Figure 17: Line chart, Kfloder, CNN

5.3.4 Results in Light-Weight CNN

In Light-Weight CNN, we use all of the three different validation method to validate the model, the Kfolder is shown in Fig20 and 23. The Leave-one-out and personal validation are shown in Appendix Fig44, Fig47, Fig **Evaluate figure: box plot, validation type: Kfolder, Light-Weight CNN:**

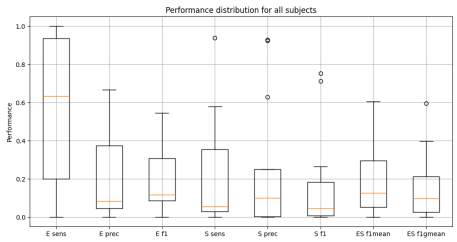

Figure 18: Dataset: SIENA

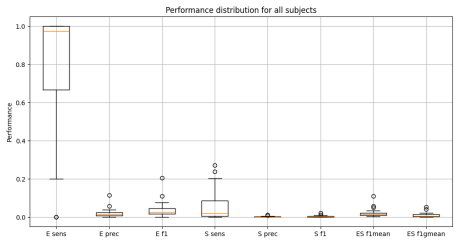

Figure 19: Dataset: CHBMIT

Figure 20: Boxplot, Kfolder, Light-Weight CNN

Evaluate figure: line chart, validation type: Kfolder, Light-Weight CNN:

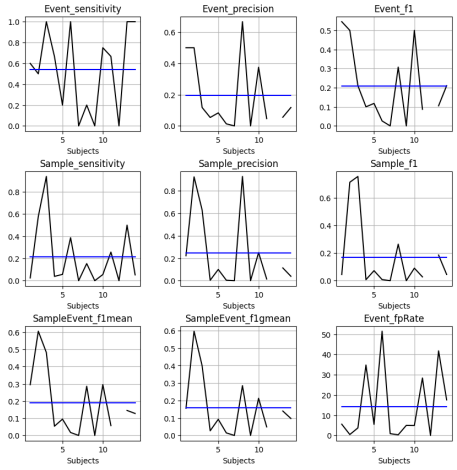

Figure 21: Dataset: SIENA

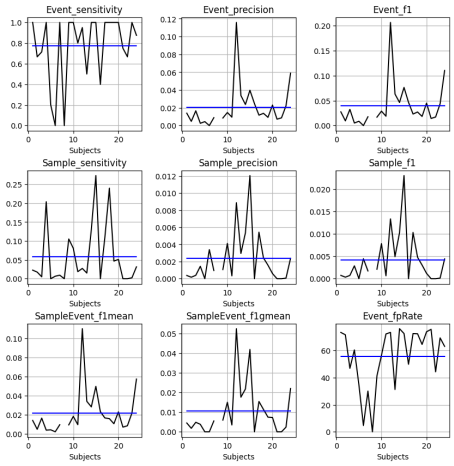

Figure 22: Dataset: CHBMIT

Figure 23: Line chart, Kfloder, Light-Weight CNN

5.3.5 Results in Transformer

In the transformer algorithm, we only test the leave-one-out and personal validation, because it is an unsupervised algorithm and the window's number is limited in their paper, it is not necessary to validate as Kfolder. The leave-one-out validation has a similar performance to the Kfolder. And in the personal validation, the CHBMIT is too large to run, so, we only validate the SIENA dataset. The results of leave-one-out validation are shown in Fig26 and Fig29. The result of personal validation is shown in Fig50 and Fig51
Evaluate figure: box plot, validation type: Leave-one-out:

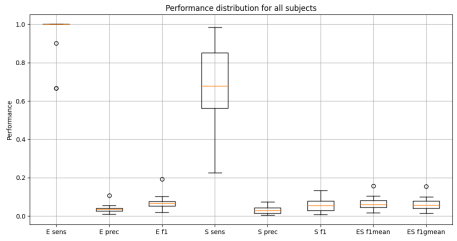

Figure 24: Dataset: SIENA

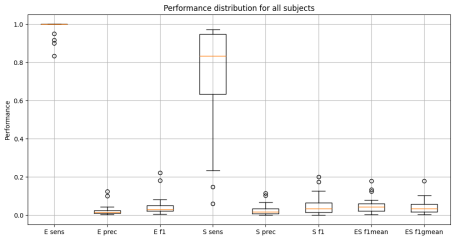

Figure 25: Dataset: CHBMIT

Figure 26: Line chart, Kfloder

Evaluate figure: box plot, validation type: Leave-one-out:

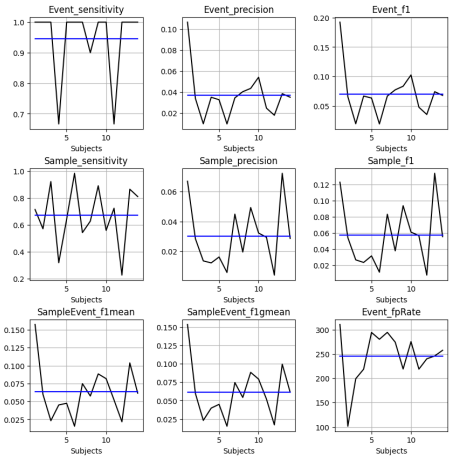

Figure 27: Dataset: SIENA

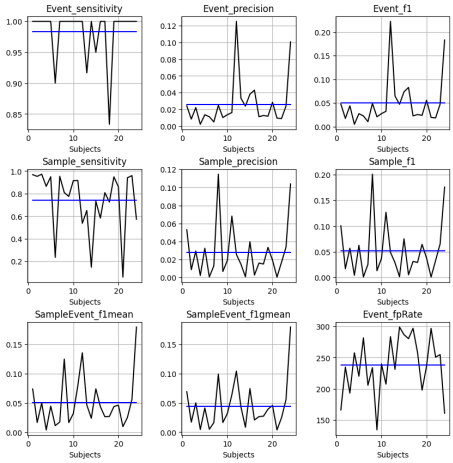

Figure 28: Dataset: CHBMIT

Figure 29: Line chart, Kfloder

5.4 Result Overview and Analysis

In the overview part, we chose two different ways to calculate the results. At first, we calculate all of the averages of the dataset. These results are shown in Tab3 and Tab**??** Secondly, we also calculate the averages except the NaN number.

Table 3: Event Based Result

Algorithm	Dataset	Validate	Sensitive	Precsion	F-score	Fp rate
RusBoost	SIENA	Left-one	0.86	0.12	0.17	328
RusBoost	SIENA	Kfolder	1.0	0.08	0.13	507
RusBoost	SIENA	personal	0.54	0.03	0.06	487
RusBoost	CHBMIT	Left-one	0.59	0.61	0.50	8.85
RusBoost	CHBMIT	Kfolder	0.67	0.23	0.30	75
RusBoost	CHBMIT	personal	1.0	0.04	0.08	442
CNN	SIENA	Kfolder	0.53	0.61	0.40	8.16
CNN	CHBMIT	Kfolder	0.52	0.33	0.29	10.7
lightCNN	SIENA	Left-one	0.7	0.04	0.07	47
lightCNN	SIENA	Kfolder	0.68	0.23	0.25	15.4
lightCNN	SIENA	Personal	0.99	0.48	0.59	4.5
lightCNN	CHBMIT	Left-one	Nan	Nan	Nan	Nan
lightCNN	CHBMIT	Kfolder	0.67	0.39	0.36	7.49
Transformer	SIENA	Left-one	0.74	0.03	0.06	237
Transformer	SIENA	personal	0.74	0.02	0.05	218
Transformer	CHBMIT	Left-one	0.98	0.02	0.04	237

Table 4: Sample Based Result

Algorithm	Dataset	Validate	Sensitive	Precsion	F1-score
RusBoost	SIENA	left-one	0.12	0.30	0.12
RusBoost	SIENA	Kfolder	0.65	0.16	0.19
RusBoost	SIENA	personal	0.11	0.03	0.05
RusBoost	CHBMIT	left-one	0.22	0.76	0.30
RusBoost	CHBMIT	Kfolder	0.16	0.57	0.20
RusBoost	CHBMIT	personal	0.83	0.19	0.28
CNN	SIENA	Kfolder	0.27	0.29	0.20
CNN	CHBMIT	Kfolder	0.34	0.42	0.21
lightCNN	SIENA	Left-one	0.68	0.02	0.05
lightCNN	SIENA	Kfolder	0.20	0.59	0.14
lightCNN	SIENA	personal	0.84	0.54	0.63
lightCNN	CHBMIT	Left-one	NaN	NaN	NaN
lightCNN	CHBMIT	Kfolder	0.34	0.42	0.21
Transformer	SIENA	left-one	0.55	0.02	0.04
Transformer	SIENA	personal	0.47	0.02	0.03
Transformer	CHBMIT	left-one	0.74	0.03	0.05

In the RusBoost algorithm, we realized all the 4 different cross-validation methods, the CHB-MIT dataset always has better performance than the SIENA dataset and with the increasing of training data, the performance of the model is better.

In the CNN algorithm, we only try to work in the Kfolder cross-validation because the data is too large to use the leave-one-out and personal validation, and the paper only uses the Kfolder as the cross-validation method. In this algorithm, the model shows better performance in SIENA than the CHBMIT dataset, which has 0.61 precision in SIENA which is the highest performance in all the algorithms.

In the CNN-light algorithm, the personal and Kfolder validation always have good performance in these two algorithms (whose F1 score is larger than 0.25), but the left-one-out validation didn't have good performance. there is a hypnosis: Because it used a part of the data to train the model, the data scale is too small and quite dispersion, and the model can not learn the feature from these data.

In the transformer algorithm, it doesn't have good performance even in the personal validation. Our hypothesis is that the window is different from the original paper, we just chose the same windows in the different datasets to ensure the validation is fair, so, it may have a bad influence on the algorithm. To make the algorithm performance better, we think change the calculate function of threshold will be a choice.

6 Conclusion and Future works

In the project, we use a seizure detection framework to validate 4 different algorithms. All of the algorithms run smoothly in the new validation framework in our experiment. The RusBoost, CNN, and CNN-Light all have good performance as well as the original research. The Transformer didn't show a very brilliant performance as their paper, the reasons are shown in paragraph5.4. In the future, we think it is necessary to validate more different algorithms in the framework and upload the framework into docker.

A Another cross-validation results

A.1 RusBoost

Evaluate figure: Box plot, validation type: Kfolder

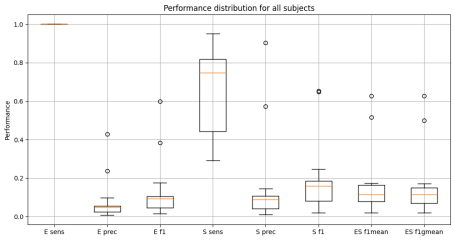

Figure 30: Dataset: SIENA

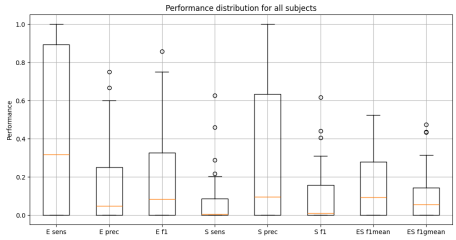

Figure 31: Dataset: CHBMIT

Figure 32: Boxplot figure, Kfolder

Evaluate figure: line chart, validation type: Kfolder

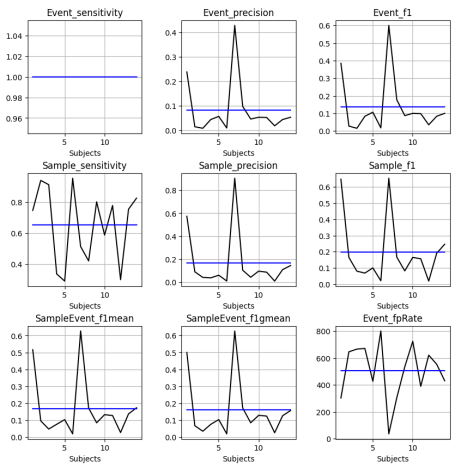

Figure 33: Dataset: SIENA

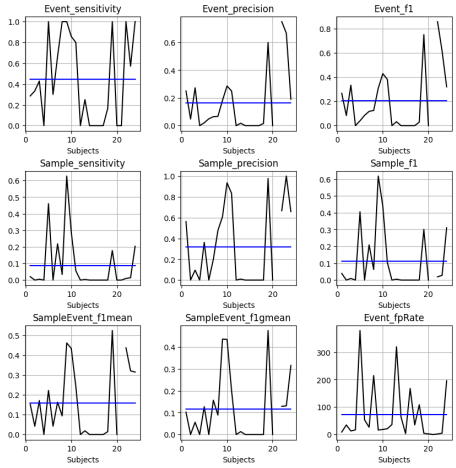

Figure 34: Dataset: CHBMIT

Figure 35: line figure, Kfolder

Evaluate figure: Box plot, validation type: Personal

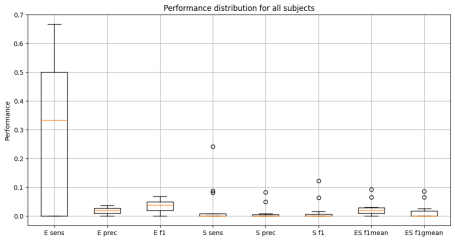

Figure 36: Dataset: SIENA

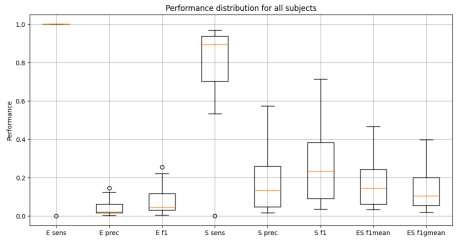

Figure 37: Dataset: CHBMIT

Figure 38: Boxplot figure, Kfolder

Evaluate figure: line chart, validation type: Personal

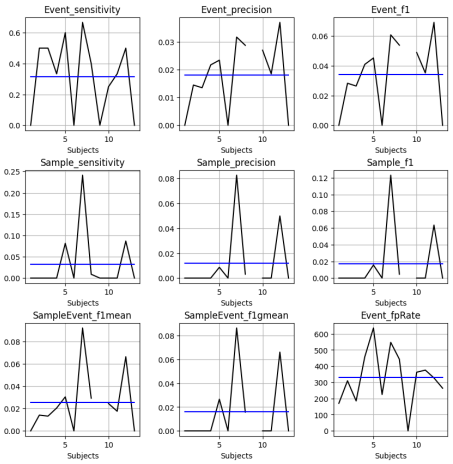

Figure 39: Dataset: SIENA

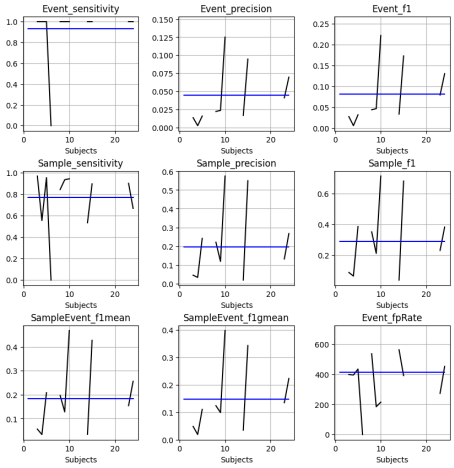

Figure 40: Dataset: CHBMIT

Figure 41: Boxplot figure, Kfolder

A.2 CNN Light

Evaluate figure: Box plot, validation type: Leave-one-out

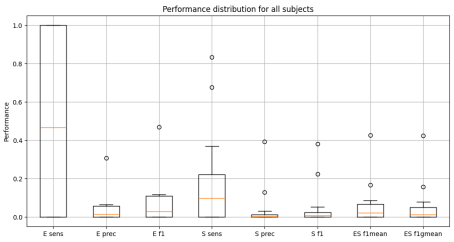

Figure 42: Dataset: SIENA

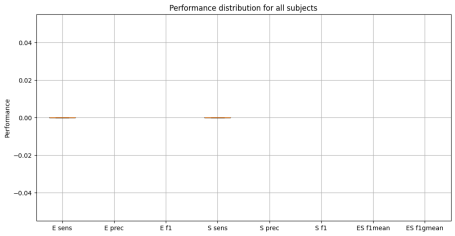

Figure 43: Dataset: CHBMIT

Figure 44: Box plot, Leave-one-out, CNN-light-weight

Evaluate figure: line chart, validation type: Leave-one-out

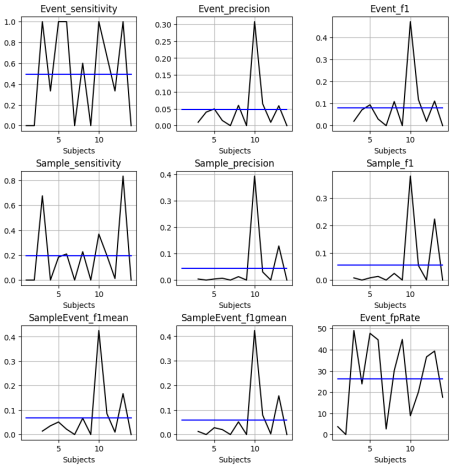

Figure 45: Dataset: SIENA

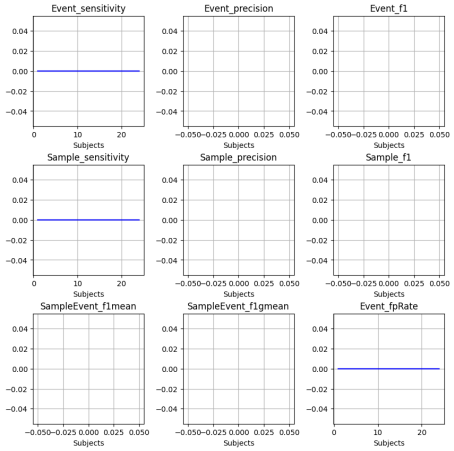

Figure 46: Dataset: CHBMIT

Figure 47: line figure, Leave-one-out, CNN-light-weight

Evaluate figure: Box plot, validation type: Personal

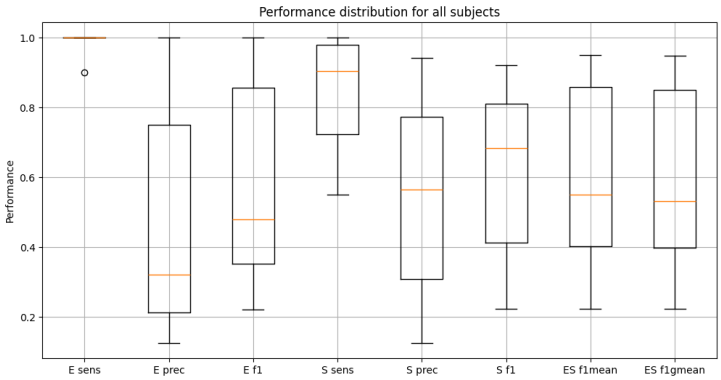

Figure 48: CNNLight, SIENA, personal

Evaluate figure: line chart, validation type: Personal

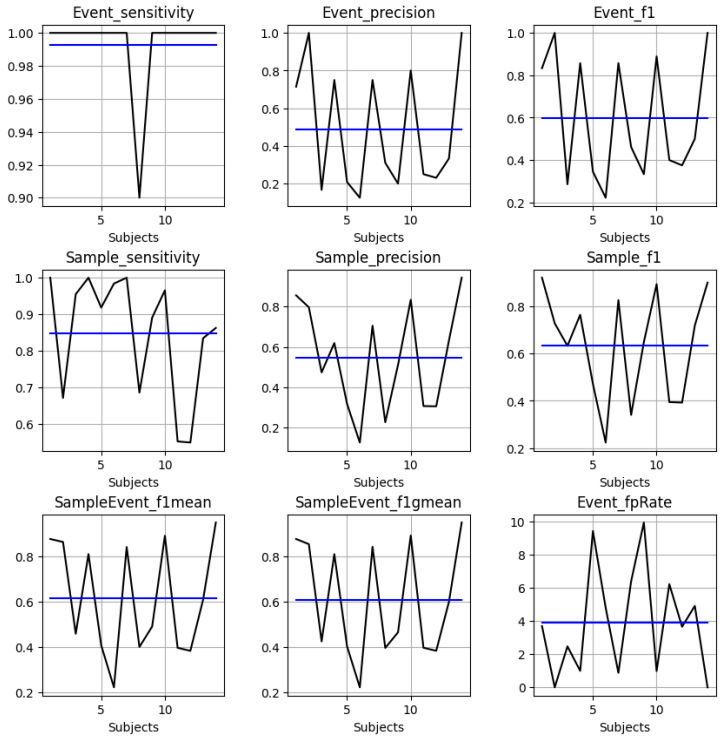

Figure 49: CNNLight, SIENA, personal

A.3 Transformer

Evaluate figure: Box plot, validation type: Personal

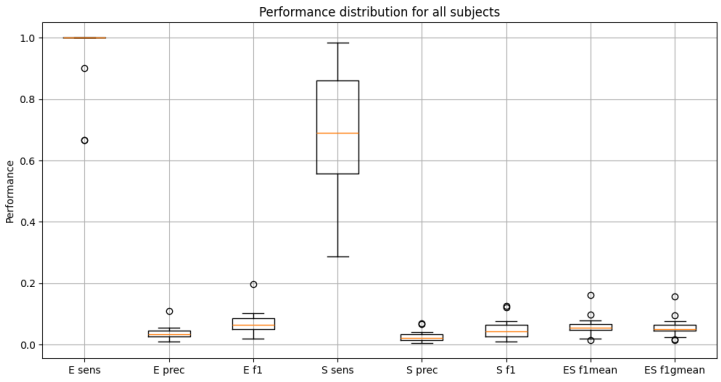

Figure 50: Transformer, SIENA, personal

Evaluate figure: line chart, validation type: Personal

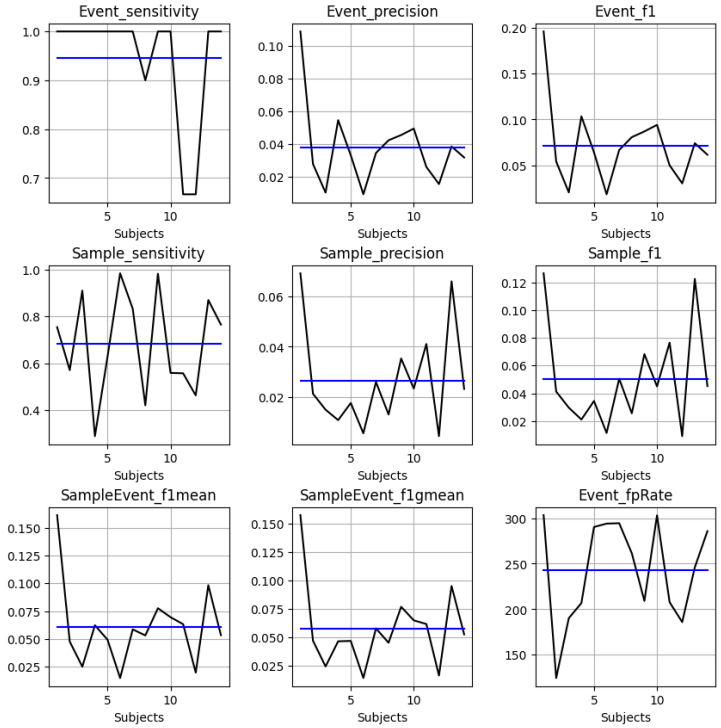

Figure 51: Transformer, SIENA, personal

B Feature in RusBoost

This part is a cite from the Marín et al. (2021), the detail of the feature can also be found in their paper appendix. **Standard deviation:**

$$\sigma = \frac{1}{\sqrt{n_w - 1}} \sqrt{\sum_{t=1}^{n_w} (x_t - \overline{x})^2} \tag{5}$$

Mean absolute deviation with respect to the median

$$D_{Me} = \frac{1}{n_w} \sum_{t=1}^{n_w} |x_t - Me| \tag{6}$$

Skewness coefficient:

$$Skw = \frac{1}{\sigma^3}\sum_{t=1}^{n_w}(x_t - \bar{x})^3 \tag{7}$$

Katz fractal dimension:

$$KFD = \frac{\log(n_w - 1)}{\log(n_w - 1) + \log(d/L)} \tag{8}$$

$$d = \max\left\{\sqrt{(1-t)^2 + (x_1 - x_t)^2}\right\}_{t=1}^{n_w} \tag{9}$$

$$L = \sum_{t=1}^{n_w - 1}\sqrt{1 + (x_i - x_{t+1})^2} \tag{10}$$

Area of the second-order difference plot:

$$Y_t = x_t + 1 - x_t \tag{11}$$

$$Z_t = x_t + 2 - x_t + 1 \tag{12}$$

$$A = 3\pi\sqrt{(S_Y^2 + S_Z^2)^2 - D^2} \tag{13}$$

$$S_Y = \frac{1}{\sqrt{n_w - 2}}\sqrt{\sum_{t=1}^{n_w-2} Y_t^2} \tag{14}$$

$$S_Z = \frac{1}{\sqrt{n_w - 2}}\sqrt{\sum_{t=1}^{n_w-2} Z_t^2} \tag{15}$$

$$S_{YZ} = \frac{1}{n_w - 2}\sum_{t=1}^{n_w-2} Y_t Z_t \tag{16}$$

$$D = \sqrt{S_Y^2 + S_Z^2 - 4(S_Y^2 S_Z^2 - S_{YZ}^2)} \tag{17}$$

Mean betweenness centrality of the recurrent network

$$\text{bet} = \frac{1}{2|\mathcal{V}|}\sum_{u=1}^{|\mathcal{V}|}\sum_{t=1,s=1}^{|\mathcal{V}|}\frac{n_{st}(u)}{N_{st}} \tag{18}$$

Mean closeness of the recurrent network

$$\text{clo} = \frac{1}{|\mathcal{V}|(|\mathcal{V}| - 1)^2}\sum_{t=1}^{|\mathcal{V}|}\frac{M_t^2}{C_t} \tag{19}$$

C Figures in other paper

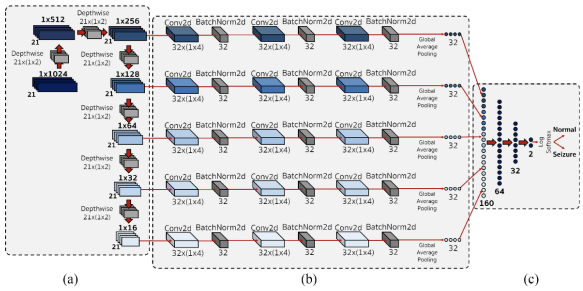

Figure 52: Framework of CNN

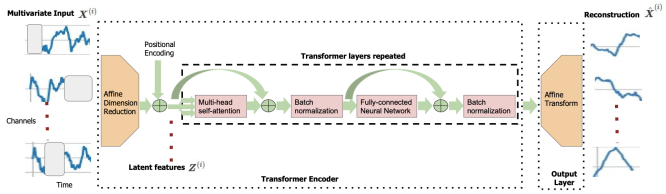

Figure 53: Framework of Transformer

References

D. Berrar et al. Cross-validation., 2019.

Z. Chen, G. Lu, Z. Xie, and W. Shang. A unified framework and method for eeg-based early epileptic seizure detection and epilepsy diagnosis. *IEEE Access*, 8:20080–20092, 2020a. doi: 10.1109/ACCESS.2020.2969055.

Z. Chen, G. Lu, Z. Xie, and W. Shang. A unified framework and method for eeg-based early epileptic seizure detection and epilepsy diagnosis. *IEEE Access*, 8:20080–20092, 2020b.

P. Detti. Siena scalp eeg database. *PhysioNet. doi*, 10, 2020.

ESL. Framework for validation of epileptic seizure detection algorithms. `https://eslweb.epfl.ch/epilepsybenchmarks/framework/#annotation`.

A. S. Galanopoulou, P. S. Buckmaster, K. J. Staley, S. L. Moshé, E. Perucca, J. Engel Jr, W. Löscher, J. L. Noebels, A. Pitkänen, J. Stables, et al. Identification of new epilepsy treatments: issues in preclinical methodology. *Epilepsia*, 53(3):571–582, 2012.

X. Hu, S. Yuan, F. Xu, Y. Leng, K. Yuan, and Q. Yuan. Scalp eeg classification using deep bi-lstm network for seizure detection. *Computers in Biology and Medicine*, 124:103919, 2020. ISSN 0010-4825. doi: https://doi.org/10.1016/j.compbiomed.2020.103919. URL `https://www.sciencedirect.com/science/article/pii/S0010482520302614`.

B. Huang, A. Abtahi, and A. Aminifar. Lightweight machine learning for seizure detection on wearable devices. In *ICASSP 2023 - 2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 1–2, 2023. doi: 10.1109/ICASSP49357.2023.10096280.

M. R. Marín, I. V. Martínez, G. R. Bermúdez, and M. Porfiri. Integrating old and new complexity measures toward automated seizure detection from long-term video eeg recordings. *Iscience*, 24(1), 2021.

Y. Potter, G. Zerveas, C. Eickhoff, and D. Duncan. Unsupervised multivariate time-series transformers for seizure identification on eeg. In *2022 21st IEEE International Conference on Machine Learning and Applications (ICMLA)*. IEEE, Dec. 2022. doi: 10.1109/icmla55696.2022.00208. URL http://dx.doi.org/10.1109/ICMLA55696.2022.00208.

C. Seiffert, T. M. Khoshgoftaar, J. Van Hulse, and A. Napolitano. Rusboost: A hybrid approach to alleviating class imbalance. *IEEE Transactions on Systems, Man, and Cybernetics - Part A: Systems and Humans*, 40(1):185–197, 2010. doi: 10.1109/TSMCA.2009.2029559.

A. H. Shoeb. *Application of machine learning to epileptic seizure onset detection and treatment*. PhD thesis, Massachusetts Institute of Technology, 2009.

A. Temko, E. Thomas, W. Marnane, G. Lightbody, and G. Boylan. Eeg-based neonatal seizure detection with support vector machines. *Clinical Neurophysiology*, 122(3):464–473, 2011. ISSN 1388-2457. doi: https://doi.org/10.1016/j.clinph.2010.06.034. URL https://www.sciencedirect.com/science/article/pii/S1388245710006036.

P. Thuwajit, P. Rangpong, P. Sawangjai, P. Autthasan, R. Chaisaen, N. Banluesombatkul, P. Boonchit, N. Tatsaringkansakul, T. Sudhawiyangkul, and T. Wilaiprasitporn. Eegwavenet: Multiscale cnn-based spatiotemporal feature extraction for eeg seizure detection. *IEEE Transactions on Industrial Informatics*, 18(8):5547–5557, 2022. doi: 10.1109/TII.2021.3133307.

D. Valentine. Learning eeg. https://www.learningeeg.com/montages-and-technical-components,.

M. Winterhalder, T. Maiwald, H. Voss, R. Aschenbrenner-Scheibe, J. Timmer, and A. Schulze-Bonhage. The seizure prediction characteristic: a general framework to assess and compare seizure prediction methods. *Epilepsy Behavior*, 4(3):318–325, 2003. ISSN 1525-5050. doi: https://doi.org/10.1016/S1525-5050(03)00105-7. URL https://www.sciencedirect.com/science/article/pii/S1525505003001057.

M. Zhou, C. Tian, R. Cao, B. Wang, Y. Niu, T. Hu, H. Guo, and J. Xiang. Epileptic seizure detection based on eeg signals and cnn. *Frontiers in Neuroinformatics*, 12, 2018. ISSN 1662-5196. doi: 10.3389/fninf.2018.00095. URL https://www.frontiersin.org/articles/10.3389/fninf.2018.00095.