UPenn and Mayo Clinic's Seizure Detection Challenge

CS 534 Machine Learning Project Report

Team: Malvern Madondo, Yazhuo Zhang, Ramraj Chandradevan

Abstract

Physicians and researchers working in epilepsy must often review large quantities of continuous electroencephalography (EEG) data in order to identify seizures. Recent advances in machine learning have enabled the development of automated seizure detection algorithms. In this paper, we report our findings on tackling the UPenn and Mayo Clinic's Seizure Detection Challenge hosted on Kaggle. In particular, we analyze and implement various algorithms, such as Convolutional Neural Networks and Random Forests, in order to detect seizures using intracranial electroencephalography from canines and humans with epilepsy.

Keywords: Kaggle, epilepsy, seizure detection, intracranial EEG

0. Introduction

Epilepsy is the third most common neurologic disorder. Each year, about 150,000 Americans are diagnosed with epilepsy. Over a lifetime, 1 in 26 U.S. people will be diagnosed with the disease. Epileptic seizures are defined as episodes of excessive or abnormal synchronous neuronal activity in the brain. Common causes of epilepsy include brain tumors and infections, strokes, and head traumas. Some treatment options for epilepsy include surgery and administration of antiepileptic drugs. However, for individuals with drug-resistant epilepsy, responsive neurostimulation is often employed. This is the application of electrical stimulation to terminate the seizure in its early stages, which is done by means of an implanted device. This approach requires the early detection of seizures with high accuracy based on the intracranial recording of the brain generated electric field potentials, called electroencephalography (EEG).

Detecting seizures from EEG data is a difficult and laborious task. Physicians and researchers working in epilepsy must often review large quantities of continuous EEG data to identify seizures, which in some patients may be quite subtle. As such, algorithms that can automatically detect seizures in large EEG datasets with low false positive and false negative rates would greatly assist clinical care and basic research. To this end, the University of Pennsylvania and Mayo Clinic jointly hosted a Kaggle competition, sponsored by the National Institutes of Health (NINDS), and the American Epilepsy Society, to crowdsource the development of seizure detection algorithms using intracranial electroencephalography from canines and humans with epilepsy.

Our goal is to tackle this challenge, which comprises two major tasks. The first task is to predict whether a given clip, collected from eight human patients with epilepsy and four canines with naturally-occurring epilepsy, is a seizure or not (ictal or interictal). The second task is to predict the probability that the clip occurred within 15 seconds of its respective seizure.

This report is organized as follows: Section 1 describes the data and the corresponding preprocessing and feature extraction techniques; Section 2 highlights the various classification algorithms applied on the computed features; Section 3 is about the evaluation of the algorithms on the dataset and results; Section 4 highlights the issues of prediction task; Section 5 discusses the model selection and parameter optimization. Section 6 concludes our work.

1. Signal Processing

1.0 Dataset

The seizure detection challenge dataset consists of two subjects (human and dog) and two classes (ictal and interictal). The data is organized in folders containing training and testing data for each human or canine subject. The training data is organized into 1-second EEG clips labeled "Ictal" for seizure data segments, or "Interictal" for non-seizure data segments. Ictal training and testing data segments are provided covering the entire seizure, while interictal data segments are provided covering approximately the mean seizure duration for each subject. The ground truth labels for the testing data for both subject categories is made available in a csv file.

The human data is from patients with temporal and extratemporal lobe epilepsy undergoing evaluation for epilepsy surgery. The iEEG recordings are from depth electrodes implanted along the anterior-posterior axis of hippocampus, and from subdural electrode grids in various locations. Data sampling rates vary from 500 Hz to 5,000 Hz. The canine data is from an implanted device acquiring data from 16 subdural electrodes. Two 4-contact strips are implanted over each hemisphere in an antero-posterior orientation. The data was recorded continuously at a sampling frequency of 400 Hz and referenced to the group average.

1.1 Preprocessing

Data preprocessing is a crucial step in any data-centric applications. Since our dataset is a time-series dataset, we need to preprocess the dataset and at the same time extract handcrafted features manually. The latter part is called feature engineering. We sought to extract as many features as possible in order to capture the static and dynamic nature of the signal. This proved to be helpful as most of the machine learning models we applied were able to classify the data effectively.

We paid attention to the following important preprocessing techniques in signal processing:

1. Noise removal
2. DC component removal
3. Re-sampling

1.1.1 Noise removal

Noise is an unwanted component in every practical signal attained by any sensors. It has the characteristics of being additive and white. We should maintain a trade-off where we should remove the noise that is the high

frequency components in our signal, but we shouldn't remove the informative high frequency components. One way to find out the cut-off frequency to avoid noise is by looking at the spectrogram plot in Matlab.

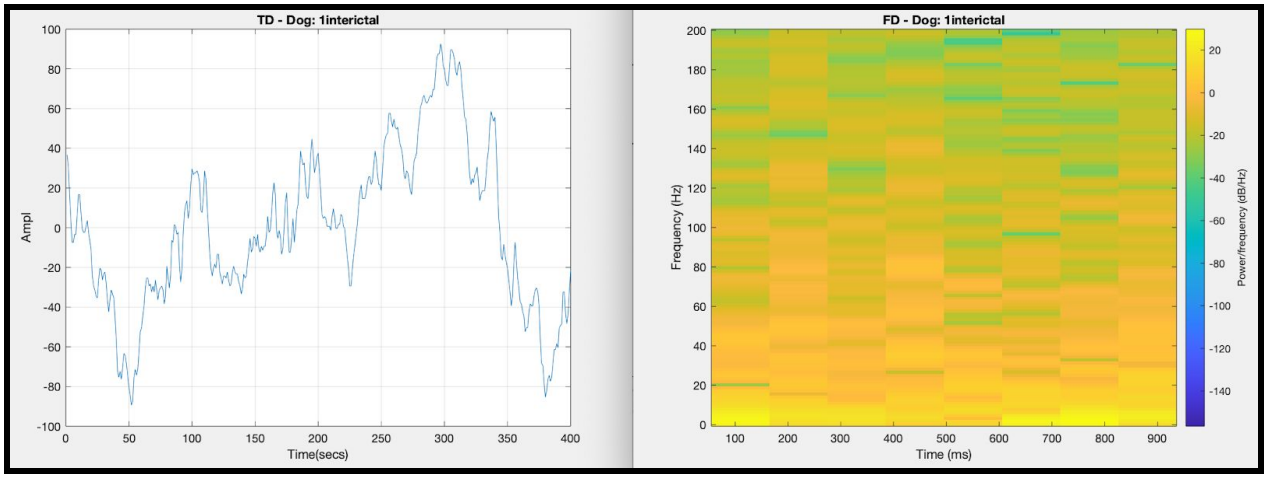


Fig 1.1 : Plot of time-domain signal and spectrogram(STFT) of Dog-interictal data

From the above figure, we can see the more power is concentrated in low frequency bands, and beyond 60Hz, the color degrades to a lower power region. This way by observing all 4 types of data (Dog-ictal, Dog-interical, Patient-ictal, and Patient-interictal), we were able to conclude 60Hz for low-pass filtering.

Once we decide upto which frequency we are going to allow the frequency components, we have two techniques to remove these noises.

1. Low-pass filtering
2. Discrete wavelet transform based multi-resolution analysis

1.1.1.1 Low-pass filtering

Filtering is a common procedure people follow in signal processing at the beginning. In our dataset, we have decided to go with low-pass filter especially FIR type because even though both FIR and IIR filters have group delay of latency in the resulting signal, the group delay of FIR filter is constant regardless of the signal frequency components, while IIR filter group delay is non-linear with respect to frequency components. Since we can adjust the group delay in the resulting signal, we picked the constant group delay filter to avoid distorting the output signal heavily.

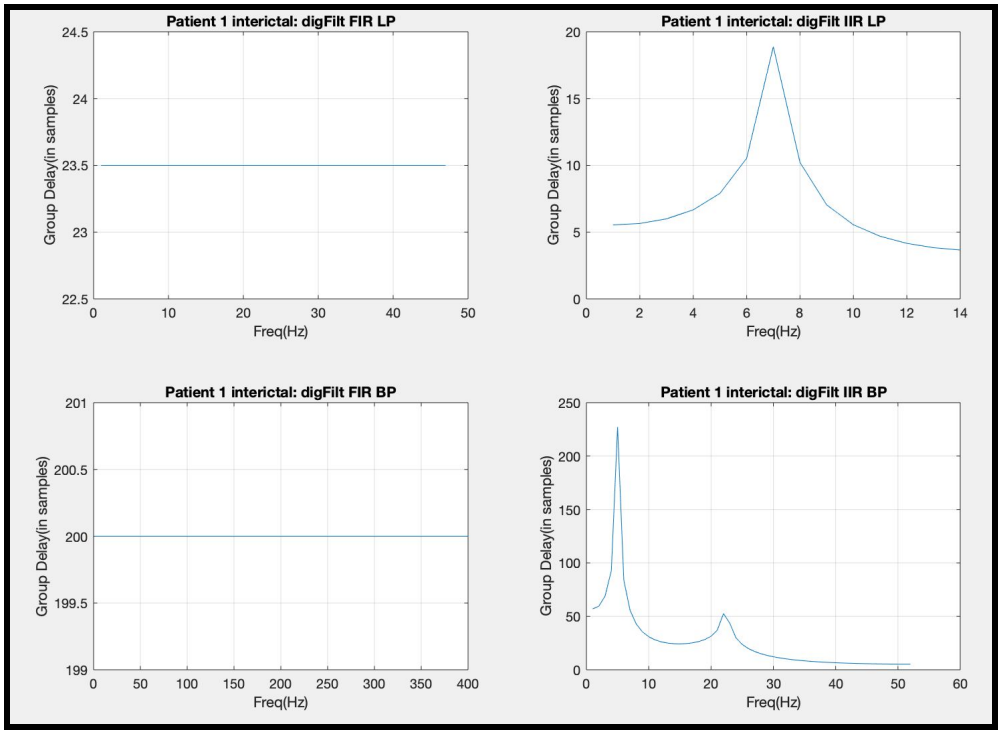


Fig 1.2: phase-response across different filters: FIR-LP, IIR-LP, FIR-BP, and IIR-LP. LP:=low-pass filter and BP:=band-pass filter

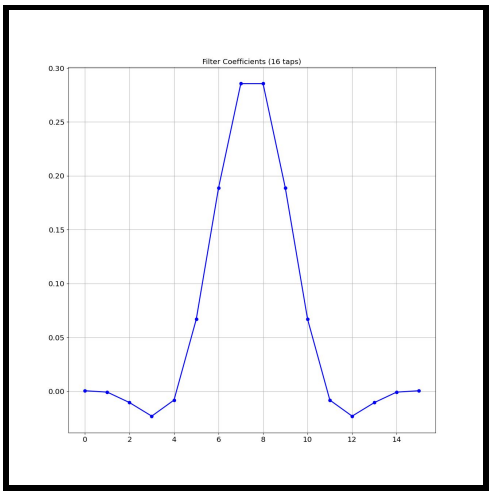


Fig 1.3: FIR-LP filter discrete coefficients

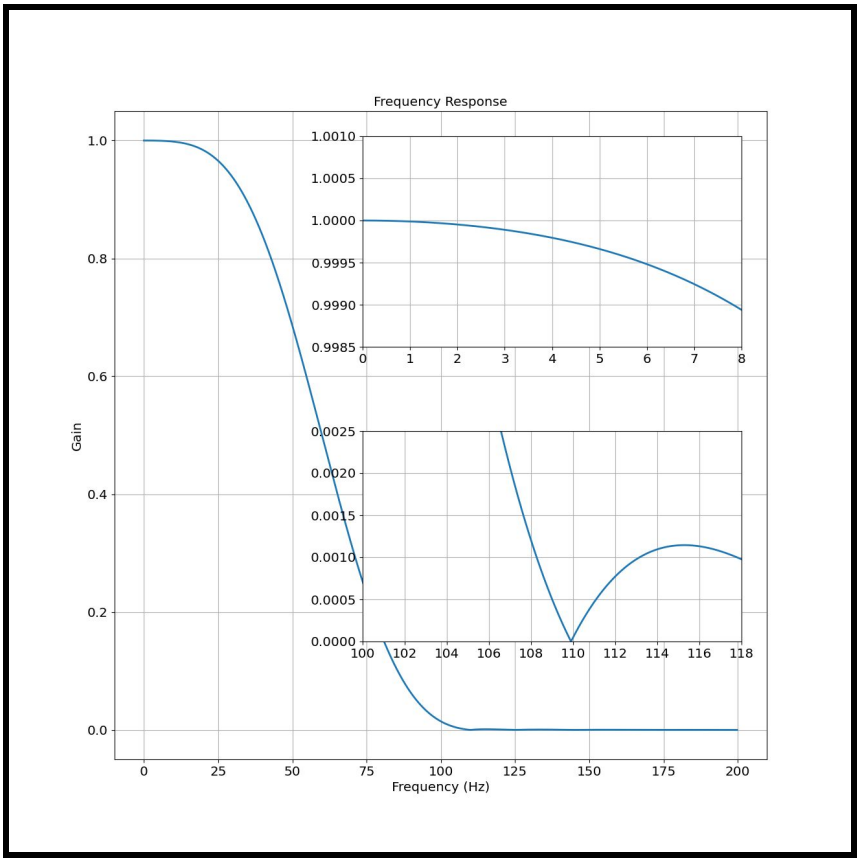


Fig 1.4: FIR-LP filter magnitude response. Upper subplot shows zoomed pass-band curve, and lower subplot shows zoomed stop-band ripples at ground-level

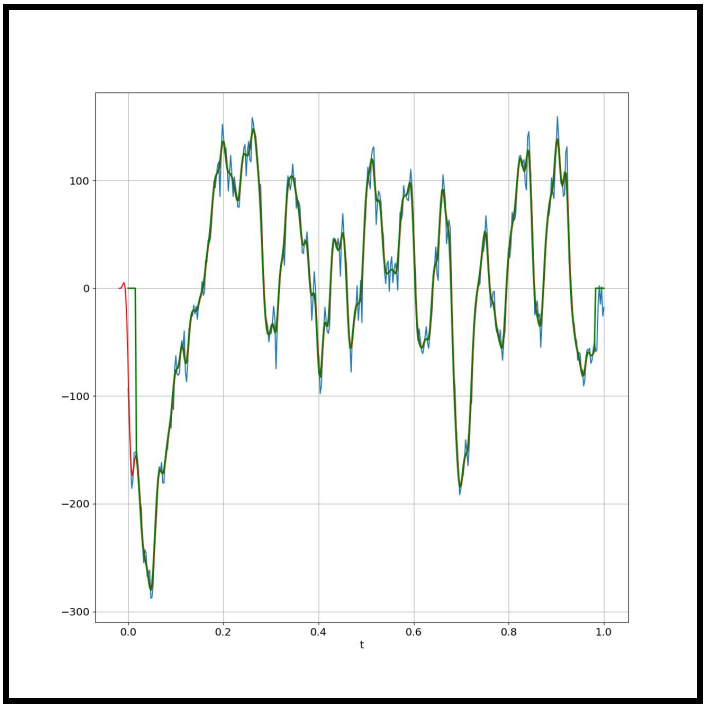


Fig 1.5: Plot shows original signal(blue), low-pass filtered signal with zero phase response(green), and low-pass filtered signal with phase response(red).

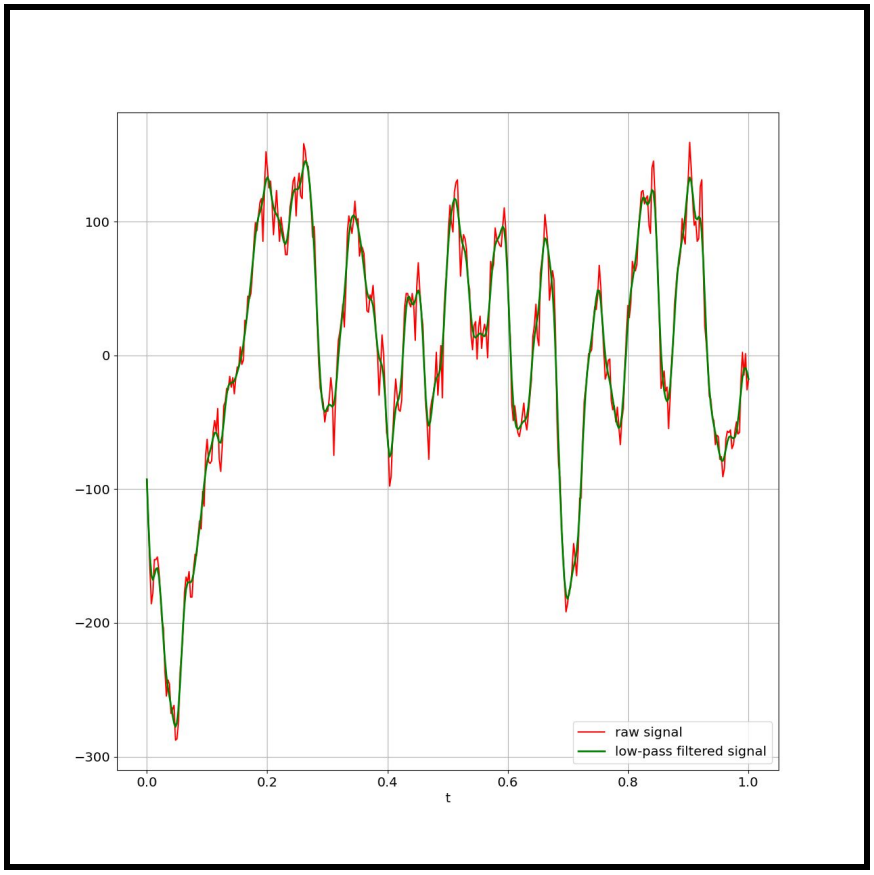


Fig 1.6: FIR-LP filtered signal shows smoothed signal without noise components.

1.1.1.2 Discrete wavelet transform based multi-resolution analysis

Since the wavelet transform topic covers a wide range of concepts and theories, we picked its important property of multi-resolution analysis. Forward analysis of wavelet transform is where the input signal is decomposed into low-pass and high-pass subbands. The coefficients responsible for both regions are called approximate and detailed coefficients. Then the downsampled low-pass subband is again decomposed into low-pass and high-pass subbands with lower **non-overlapping** split frequency. This procedure can be followed until the downsampling of the signal can be possible with Nyquist sample rate. Since we have both Patient and Dog sensor data which differs with 5000Hz and 400Hz sampling rate, we could evaluate the maximum number of wavelet levels are 10 and 7. Therefore, in the purpose of constructing a constant number of features eventually, we used a maximum number of levels as 7.

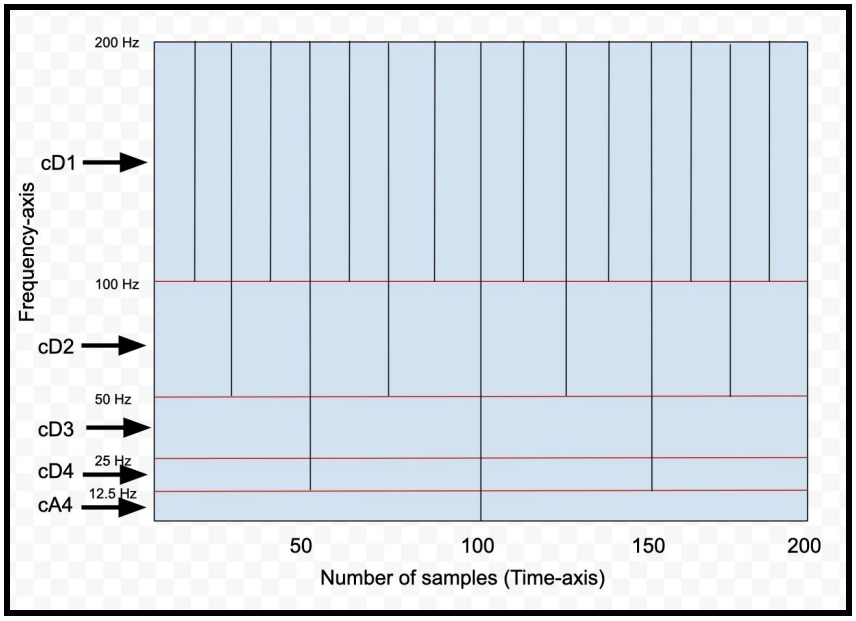


Fig 1.7: Time-frequency resolution of wavelet pyramid construction results and their respective approximate and detailed coefficients set.

Once we decompose the signal into approximate and detailed coefficients depending on the maximum number of levels, we apply inverse analysis where the reconstruction of the original signal takes place by combining low-pass and high-pass frequency component signals along with upsampling while going from lower to higher level. Even though we could get a fully reconstructed signal, by removing certain thresholded high-pass subband regions, we can remove the noisy frequency components during the reconstruction procedure. The final version of the reconstructed signal will apparently look like the original signal except the noise is removed.

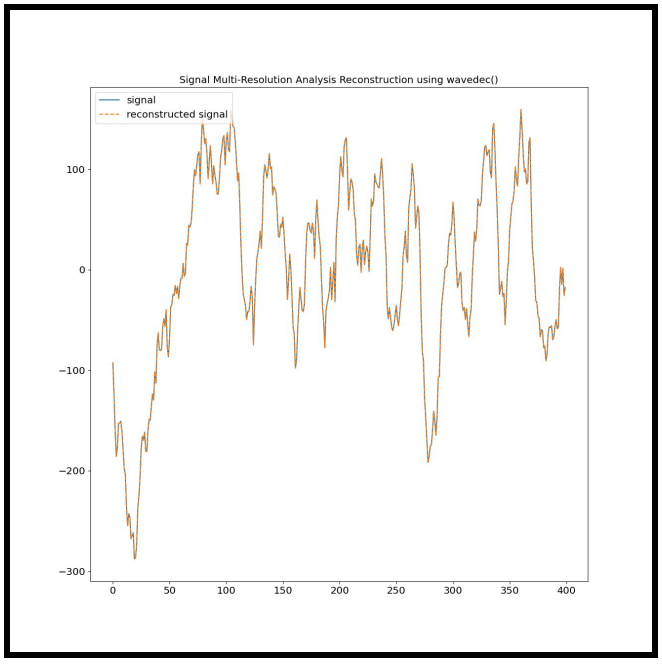


Fig 1.8: Perfect signal reconstruction after wavelet decomposition.

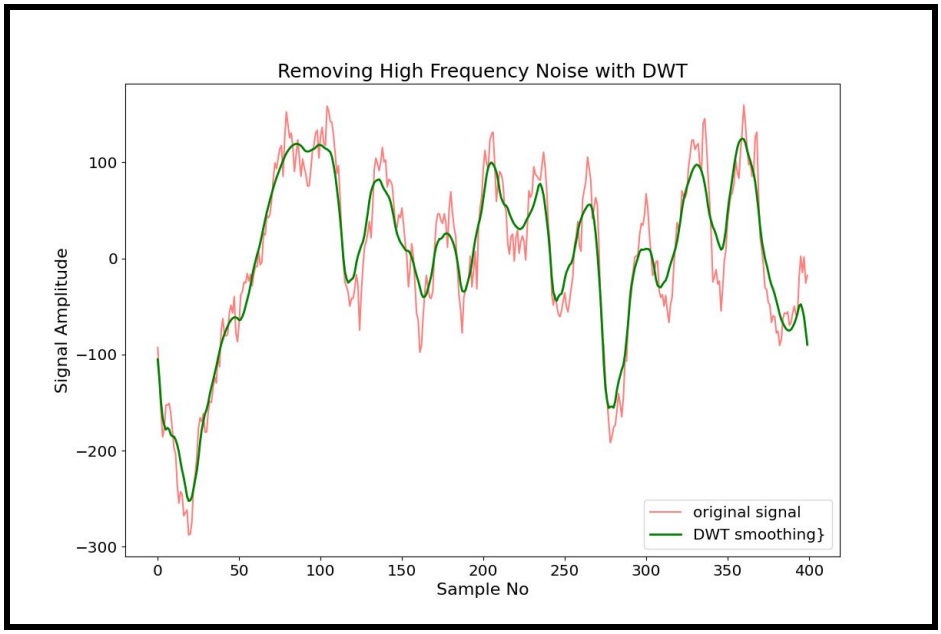


Fig 1.9: Signal reconstruction with defined threshold removes the noise components.

1.1.2 DC Component Removal

In addition to removing highpass frequency components, we have to remove mean value, which corresponds to the DC component of the signal. We applied linear and constant detrend function, and picked linear detrending because the change of signal trend looked linear.

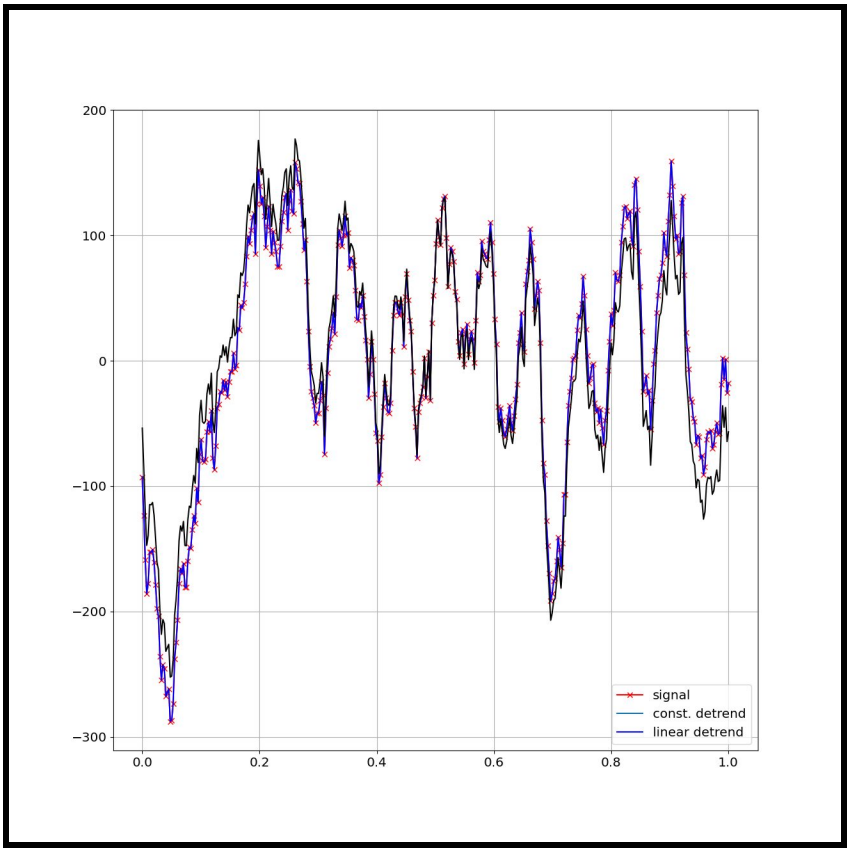


Fig 1.10: Comparison of constant and linear detrend against raw signal

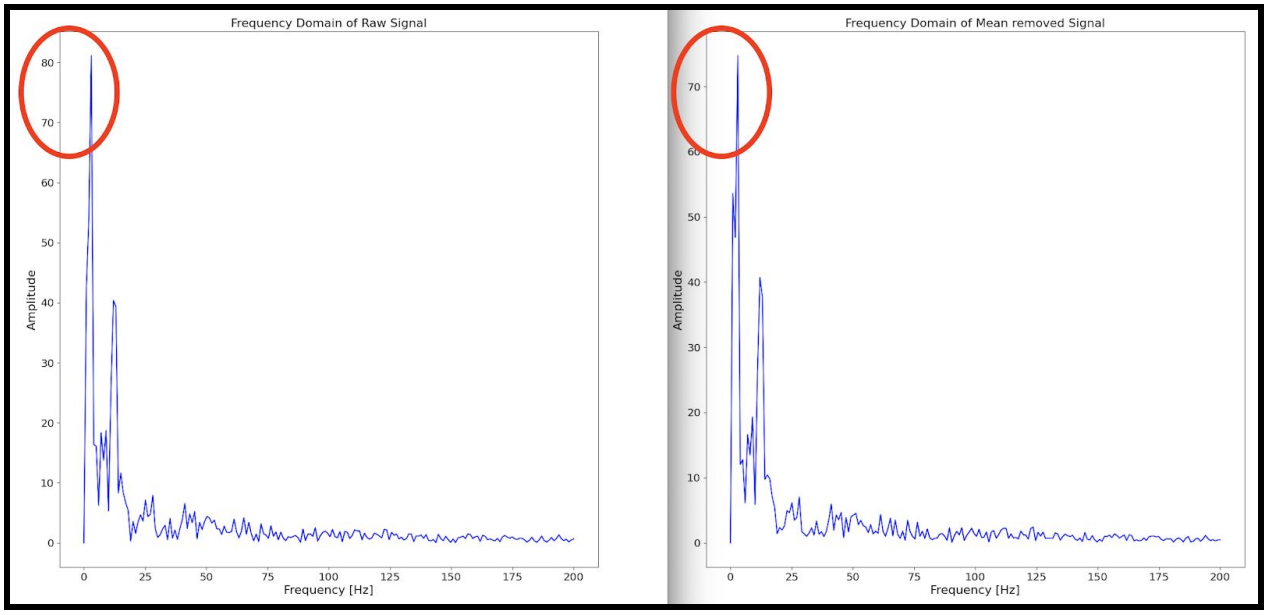


Fig 1.11: Comparison of before DC component removal and after removal in frequency-domain.

1.1.3 Downsampling

The algorithms we use depend on the spectrum of frequency the signal data possess. For dogs, all data consistently have 400Hz, while Patients data have varying frequency from 500Hz to 5000Hz. Therefore, we determined to downsample Patients data from 5000Hz to 500Hz to make all data consistent. The usage of scipy.signal library has a function called decimation, which downsamples the signal after applying an anti-aliasing filter.

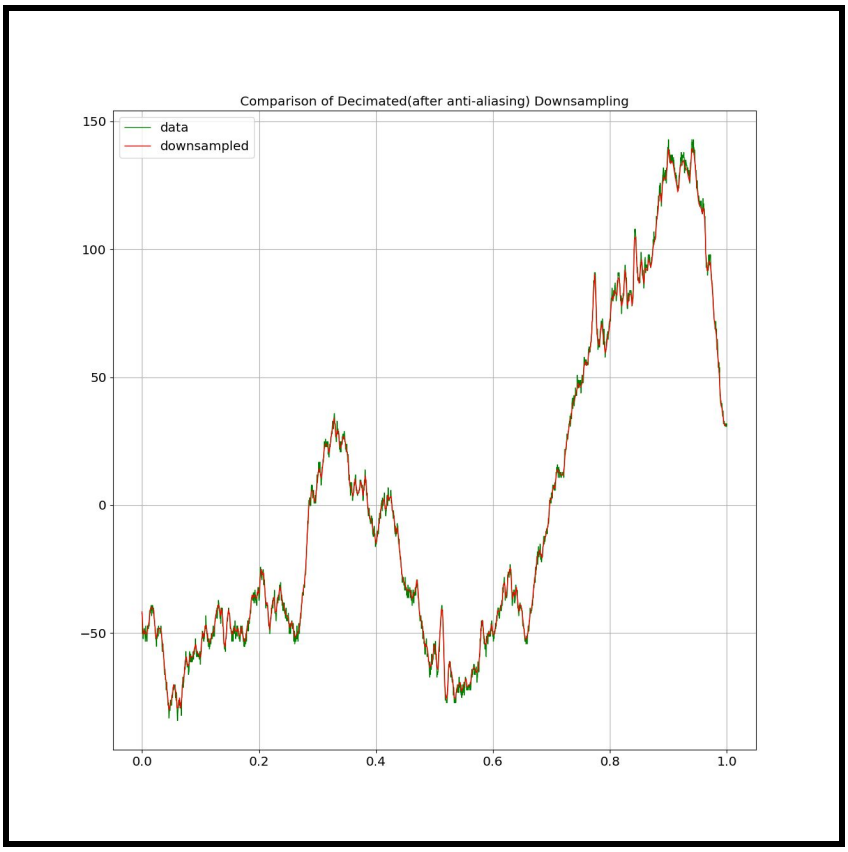


Fig 1.12: Comparison of downsampled filter after applying anti-aliasing filter

1.2 Feature Extraction

1.2.1 Frequency-based feature extraction

In terms of features, we selected top-k peaks from the graphs driven for different transformation of time domain signal. We extracted top-2 peaks to limit our number of features from below graphs,

1. Power spectral density (PSD) plot: y_values (#2)
2. Fast Fourier transform (FFT) plot: y_values (#2)

All together, we extracted 6 features per channel. We earlier experimented by taking x and y values as well, but since the number of features exceeded the number of samples, we dropped x coordinates because empirically the position of where the signal peak occurs makes little sense.

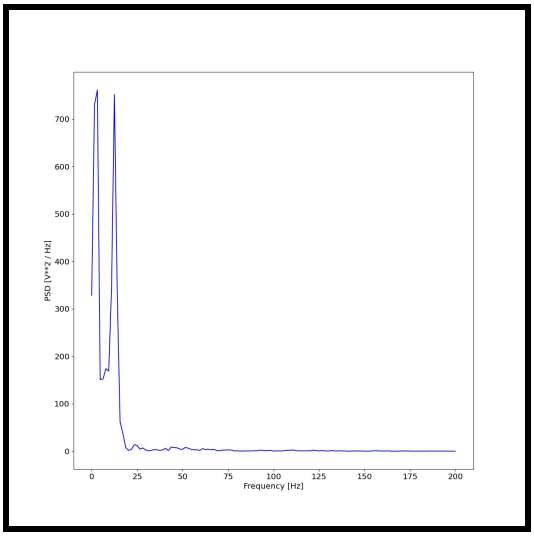


Fig 1.13: PSD plot

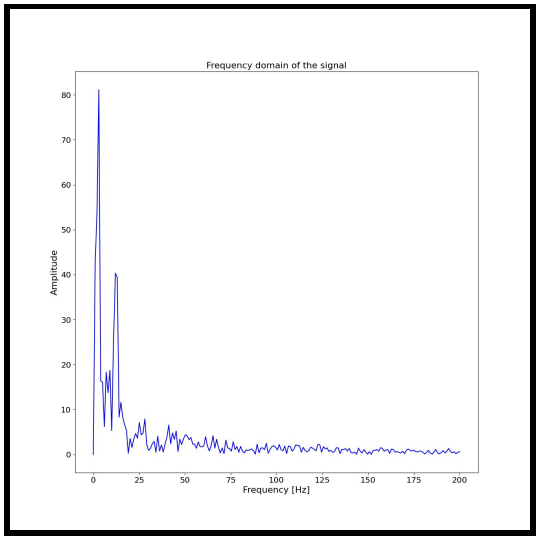


Fig 1.14: FFT plot

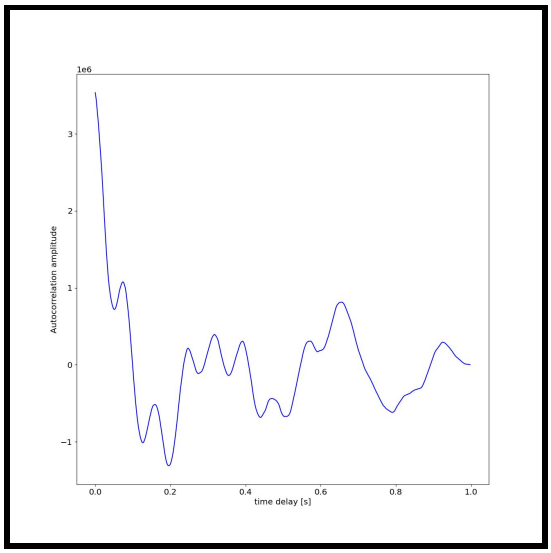


Fig 1.15: Autocorrelation plot

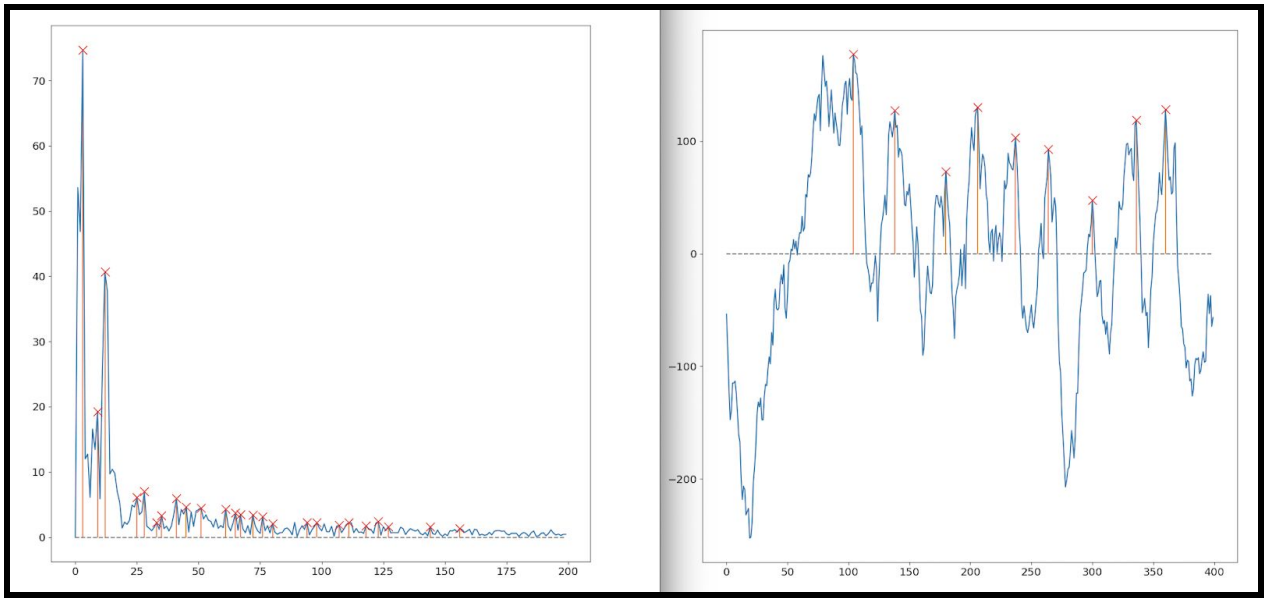


Fig 1.16: Peaks found in FFT plot and time-domain signal. Certain thresholds for height and weight were provided to find-peak function.

1.2.2 Wavelet-based feature extraction

In order to decouple the signal into wavelet components, we selected 4 numbers of levels since we don't want to increase the total feature size. While those coefficients are denoted as [cA4, cD4, cD3, cD2, cD1], and their subbands are denoted as (0→ 12.5Hz), (12.5→ 25Hz), (25→ 50Hz), (50→ 100Hz), (100→ 200Hz). In order to avoid taking noise into account, we excluded the high-pass subband, which is cD1 and covers subband of (100→ 200Hz).

In each subband coefficients, we extracted features such as,

1. Statistical features (#8)
    a. Percentile value @5
    b. Percentile value @25
    c. Percentile value @75
    d. Percentile value @95
    e. Median
    f. Mean
    g. Variance
    h. Root mean square (RMS)
2. Number of crossings as features (#2)
    a. Zero-crossings
    b. Mean-crossings
3. Entropy based features (#1)
    a. Entropy

All together, we were able to extract 11 features per wavelet component per channel. Therefore, across 4 wavelet decompositions, totally 44 features were extracted. In total, for Dog across 16 channels, we were able to get 800 features, and for Patients, since the number of channels are varying, per channel 50 features were extracted.

Along with the wavelet transform, so far we considered discrete wavelet transform (DWT). However, there is another family of wavelet transforms, called continuous wavelet transform (CWT). Unlike DWT, CWT has a continuous mother-wavelet, which results in a continuous spectrum of wavelet coefficients. Therefore, we only would be able to visualize the results with scalogram, which plots the time-frequency composition of the signal along the scale range.

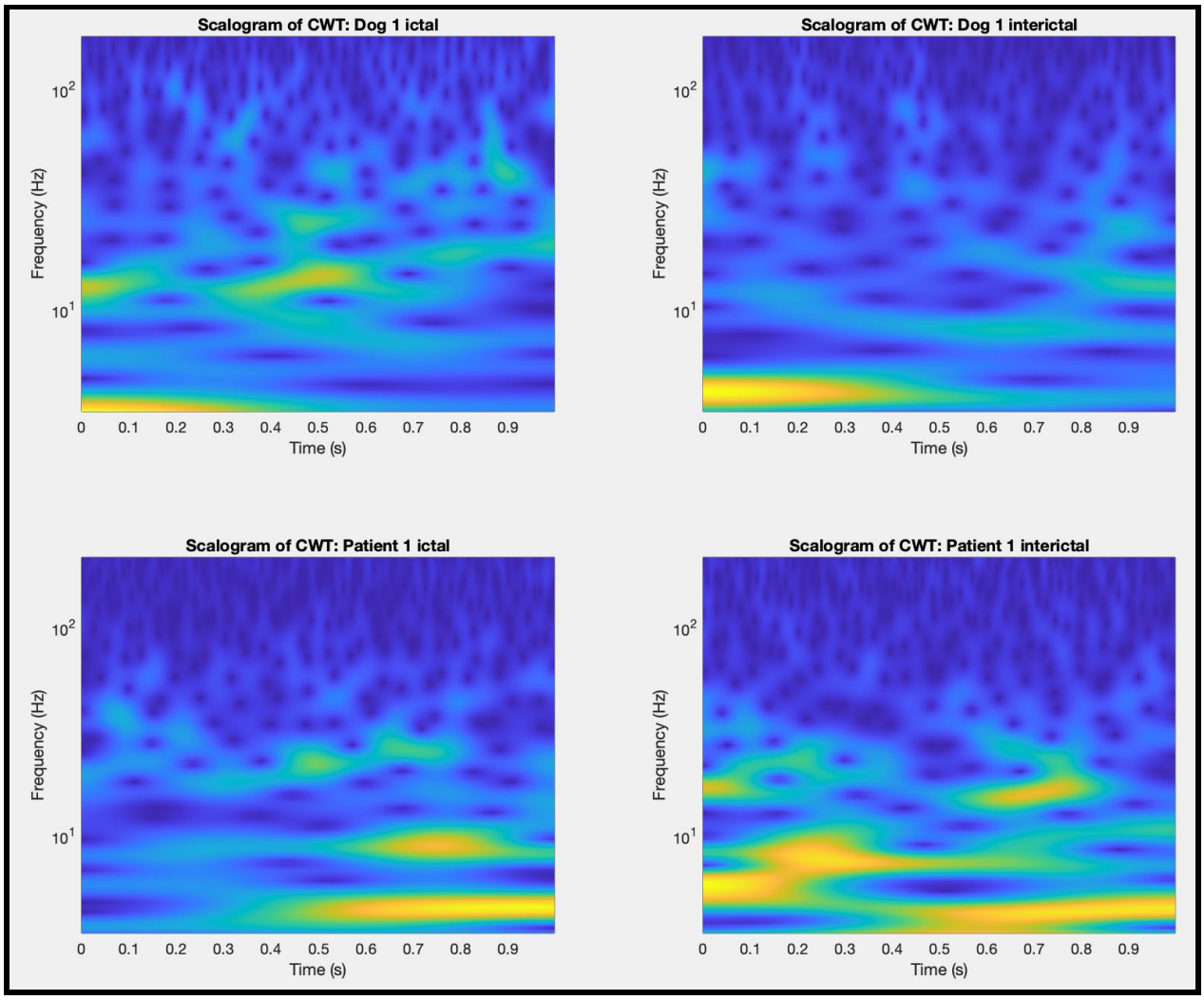


Fig 1.17: Comparison of scalograms for 4 different types of data: Dog-ictal, Dog-interical, Patient-ictal, and Patient-interictal.

We have made a hypothesis to generate a scalogram for our data, train a convolutional neural network (CNN), and based on the performance try to select the most prominent region from the input image that corresponds to the final classification. The later part can be realized as regularizer or feature selection. But incorporating the

feature selection algorithm into a complex CNN network was laborious and time consuming, thus we have to skip that in our methods and experiments.

2. Classification Methods

2.0 Baseline: Logistic Regression

Logistic regression (LR) is a widely used statistical modeling technique in which the model uses a logistic function to describe the relationship between a binary response variable (e.g., 'seizure' or 'no seizure) and a set of predictor variables. In the binary case, the logistic model fits a linear decision boundary for both classes, then is passed through a sigmoid function to transform from the log of odds to the probability that the sample belongs to the positive class. Because the model tries to find the best separation between the positive class and negative class, this model performs well when the data separation is noticeable.

Model Architecture:

```
LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
                   intercept_scaling=1, l1_ratio=None, max_iter=100,
                   multi_class='auto', n_jobs=None, penalty='l2',
                   random_state=None, solver='saga', tol=0.0001, verbose=0,
                   warm_start=False)
```

2.1 Convolutional Neural Networks

Convolutional neural networks (CNNs) have been extensively studied for classification. We use a generalized framework proposed by Nhan et al. (2018). In their work, they pointed out the feature extraction is limited to specific patients in which the best combination of features and classifiers are not known for each patient. Therefore, the authors provided a generalized approach for seizure prediction, which we apply for seizure detection in our work. They also gave a detailed framework in this paper. There are three convolution blocks in this CNN, and each convolution block consists of a batch normalization, a convolution layer with a rectified linear unit activation function, and a max pooling layer.

Instead of just working on the framework described in this paper, we changed the input shape in which the x-axis represents the features and y-axis represents different channels. Besides, we modified the parameters in the model to get better results. The parameter tuning is mainly aimed at solving the overfitting since the limited available datasets.

Model Architecture:

```
input_shape = (num_channel, num_feature, 1)
model = Sequential()

#C1
model.add(Conv2D(16, 3,  padding='valid',activation='relu', input_shape=input_shape))
model.add(MaxPooling2D(pool_size=2, padding='same'))
model.add(BatchNormalization())

#C2
model.add(Conv2D(32, 2, padding='valid',activation='relu'))
model.add(MaxPooling2D(pool_size=2, padding='same'))
model.add(BatchNormalization())

#C3
model.add(Conv2D(64, 2, padding='valid',activation='relu'))
model.add(MaxPooling2D(pool_size=2, padding='same'))
model.add(BatchNormalization())

model.add(Flatten())
model.add(Dropout(0.5))
model.add(Dense(256, activation='sigmoid'))
model.add(Dropout(0.5))
model.add(Dense(2, activation='softmax', kernel_regularizer=l2(0.01), activity_regularizer=l2(0.01)))
```

2.2 Random Forests

A Random Forest (RF) is an ensemble of decision tree classifiers. It consists of bootstrapping the dataset and using a random subset of features for each decision tree to reduce the correlation of each tree, hence reducing the probability of overfitting. The generic approach we use is based on the work by Mursalin et al. (2017) in which the authors proposed a novel analysis method for detecting epileptic seizures from EEG signals. The authors present a new approach to feature selection, called ICFS (Improved Correlation-based Feature Selection) that uses a correlation based heuristic to evaluate the utility of selected features. The authors used both time and frequency domain features to feed the proposed ICFS method which selected the most prominent features for automatic seizure detection using the RF classifier. This approach seeks to improve the performance of conventional CFS methods and select the minimum number of features from a large feature space.

The authors use a simple Random Forest classifier as proposed by L. Breiman (*Random forests, Mach. Learn. 45 (1) (2001) 5–32*). They set the number of trees to be generated as 100 and the random seed as 1. In order to reduce the bias of training and test data, they propose using the k-fold cross-validation technique with k = 10.

Model Architecture:

```
RandomForestClassifier(bootstrap=True, ccp_alpha=0.0, class_weight=None,
                       criterion='gini', max_depth=None, max_features='auto',
                       max_leaf_nodes=None, max_samples=None,
                       min_impurity_decrease=0.0, min_impurity_split=None,
                       min_samples_leaf=1, min_samples_split=2,
                       min_weight_fraction_leaf=0.0, n_estimators=100,
                       n_jobs=None, oob_score=False, random_state=1, verbose=0,
                       warm_start=False)
```

2.3 Random Forests with Grid Search Optimization

Another approach that we experimented with is based on the work by Wang et al. (2019). The authors proposed an automatic detection framework for epileptic seizure based on multiple time-frequency analysis approaches. Their approach also involves a random forest classifier, but one based on the Random Forest (RF) by Archer and Kimes, 2008. The authors feed the extracted features into the model using cross-validation to obtain three classification categories of seizure, light-seizure and non-seizure. This is different from our case, where we only have two categories.

In order to improve the classification performance of the RF algorithm, the paper proposes an improved grid search algorithm to optimize and configure the model parameters. To reduce the influence of the selected training data and test data on the model evaluation, the authors used 10-fold cross validation. Another measure that they implemented in their approach was

Model Architecture:

```
GridSearchCV(cv=10, error_score=nan,
             estimator=RandomForestClassifier(bootstrap=True, ccp_alpha=0.0,
                                              class_weight=None,
                                              criterion='gini', max_depth=None,
                                              max_features='auto',
                                              max_leaf_nodes=None,
                                              max_samples=None,
                                              min_impurity_decrease=0.0,
                                              min_impurity_split=None,
                                              min_samples_leaf=1,
                                              min_samples_split=2,
                                              min_weight_fraction_leaf=0.0,
                                              n_estimators=100, n_jobs=None,
                                              oob_score=False, random_state=42,
                                              verbose=0, warm_start=False),
             iid='deprecated', n_jobs=-1,
             param_grid={'max_depth': [5, 10, 20, 100],
                         'max_features': ['auto', 'sqrt'],
                         'min_samples_leaf': [1, 2, 4],
                         'n_estimators': [30, 65, 80, 100]},
             pre_dispatch='2*n_jobs', refit=True, return_train_score=False,
             scoring='roc_auc', verbose=0)
```

According to the authors, the penalty parameters *min_sample_leaf, max_features,* and *n_estimators* were the key parameters affecting the performance of the RF classifier. We tuned these parameters, in addition to the number of trees, which we also found to be crucial.

2.4 Other Models

We also experimented with a number of classification algorithms in order to get a better intuition on what techniques work on the data. Some of these methods proved to be just as effective, if not more, than the major algorithms we have presented above.

2.4.1 Support Vector Machines

Support vector machines (SVMs) are a class of supervised learning methods. They are commonly used in classification, regression, and outlier detection. The goal is to optimally learn the best decision surface or hyperplane that maximizes the margin between classes. They use a kernel transform to achieve nonlinear classification of data. In our case, we use the Radial Basis Function (RBF). SVM is non-probabilistic, rendering a categorical output, preictal versus interictal, binary.

SVMs are memory efficient as they use a subset of training points in the decision function (called support vectors). They are effective in high dimensional spaces and in cases where the number of dimensions is greater than the number of samples. However, SVMs are prone to overfitting when the number of features is much greater than the number of samples. They also do not directly provide probability estimates.

To get the probabilities, we use k-fold cross-validation, which is an expensive operation for large datasets. By setting the "`probability=True`" in the SVM constructor, we enable class membership probability estimates. The probabilities are then calibrated using Platt scaling: logistic regression on the SVM's scores, fit by an additional cross-validation on the training data.

Model Architecture:

```
SVC(C=1.0, break_ties=False, cache_size=200, class_weight=None, coef0=0.0,
    decision_function_shape='ovr', degree=3, gamma='scale', kernel='rbf',
    max_iter=-1, probability=True, random_state=None, shrinking=True, tol=0.001,
    verbose=False)
```

2.4.2 Extreme Random Forest (ExtraTrees)

The ExtraTrees Classifier is similar to Random Forest Classifier, in that it builds multiple trees and splits nodes using random subsets of features, except that when choosing a variable at the split, samples are drawn from the entire training set rather than bootstrapping samples. In addition, node splits are selected at random, instead of being specified as is the case in Random Forests. It builds multiple trees with '`bootstrap=False`' by default, which means it samples without replacement. Thus, the ExtraTrees Classifier is less prone to overfitting, and can often produce a more generalized model than the Random Forest Classifier.

Model Architecture:

```
ExtraTreesClassifier(bootstrap=False, ccp_alpha=0.0, class_weight=None,
                     criterion='gini', max_depth=None, max_features='auto',
                     max_leaf_nodes=None, max_samples=None,
                     min_impurity_decrease=0.0, min_impurity_split=None,
                     min_samples_leaf=1, min_samples_split=2,
                     min_weight_fraction_leaf=0.0, n_estimators=65, n_jobs=None,
                     oob_score=False, random_state=None, verbose=0,
                     warm_start=False)
```

2.4.3 Extreme Gradient Boosting (XGBoost)

XGBoost stands for e*Xtreme Gradient Boosting*. Gradient boosting methods (GBM) are a novel model that combats the overfitting of decision trees. Unlike Random Forests, gradient boosting builds shorter trees, one at a time, and each new tree reduces the error the previous tree has made. XGBoost is an ensemble tree method that applies the principle of boosting weak learners. XGBoost is different from standard GBM in that it uses a more regularized model to control over-fitting since the standard GBM has no regularization, which gives it better performance. Also, the trees have a varying number of terminal nodes and the leaf weights of the trees that are calculated with less evidence are shrunk more heavily. Furthermore, an extra randomization parameter is often used to reduce the correlation between trees. XGBoost implements parallel processing and is therefore much faster than GBM. It also comes with built-in cross validation.

Model Architecture:

```
XGBClassifier(base_score=0.5, booster='gbtree', colsample_bylevel=1,
              colsample_bynode=1, colsample_bytree=1, gamma=0,
              learning_rate=0.1, max_delta_step=0, max_depth=3,
              min_child_weight=1, missing=None, n_estimators=100, n_jobs=1,
              nthread=None, objective='binary:logistic', random_state=0,
              reg_alpha=0, reg_lambda=1, scale_pos_weight=1, seed=None,
              silent=None, subsample=1, verbosity=1)
```

2.4.4 AdaBoost Classifier

AdaBoost, short for *Adaptive Boosting*, is an iterative ensemble method. It builds a strong classifier by combining multiple poorly performing classifiers. The basic concept behind Adaboost is to set the weights of classifiers and training the data sample in each iteration such that it ensures the accurate predictions of unusual observations. AdaBoost is best used to boost the performance of decision trees on binary classification problems. In our approach, we use decision trees as the weak learners.

Model Architecture:

```
AdaBoostClassifier(algorithm='SAMME',
                   base_estimator=DecisionTreeClassifier(ccp_alpha=0.0,
                                                         class_weight=None,
                                                         criterion='gini',
                                                         max_depth=5,
                                                         max_features=10,
                                                         max_leaf_nodes=None,
                                                         min_impurity_decrease=0.0,
                                                         min_impurity_split=None,
                                                         min_samples_leaf=1,
                                                         min_samples_split=2,
                                                         min_weight_fraction_leaf=0.0,
                                                         presort='deprecated',
                                                         random_state=None,
                                                         splitter='best'),
                   learning_rate=1.0, n_estimators=100, random_state=None)
```

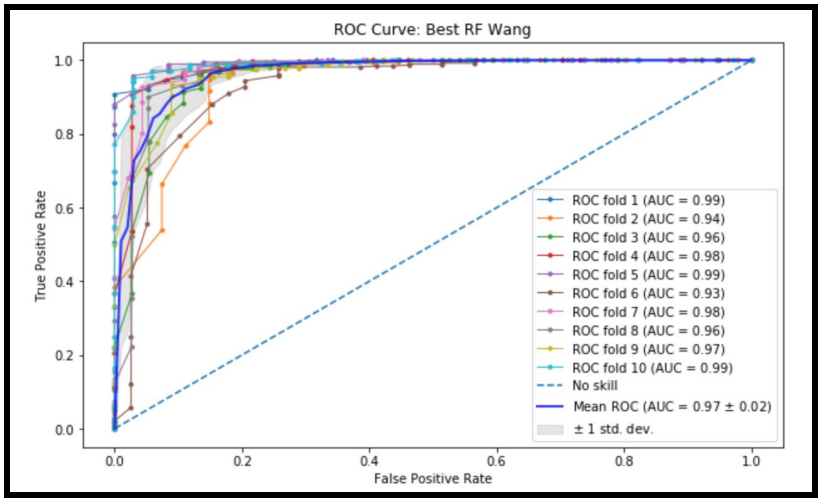
3. Results and Analysis

3.1 Performance Evaluation

Besides the accuracy, we evaluated all the classifiers using the Area Under the ROC Curve (AUC) as a metric. We also paid attention to various evaluation indicators such as the confusion matrix for each model as well as the classification report (includes precision, recall, f1-score, and support).

3.2 Results: Dog Subject Data

In general, all the models obtained good classification results. For example, the Random Forest with Grid Search Optimization model by Wang et al. (2019) had the following ROC plot, confusion matrix, and performance metrics:



```
=== Classification Report ===
              precision    recall  f1-score   support

           0       0.97      0.90      0.93       103
           1       0.99      1.00      0.99       945

    accuracy                           0.99      1048
   macro avg       0.98      0.95      0.96      1048
weighted avg       0.99      0.99      0.99      1048
```

Fig 3.1: ROC and classification report for Dog_3 experiment with the Random Forest with grid search optimization model.

The figure below shows the general comparison of the baseline Logistic Regression model with the Random Forest with Grid Search Optimization model on an experiment with Dog_3 data:

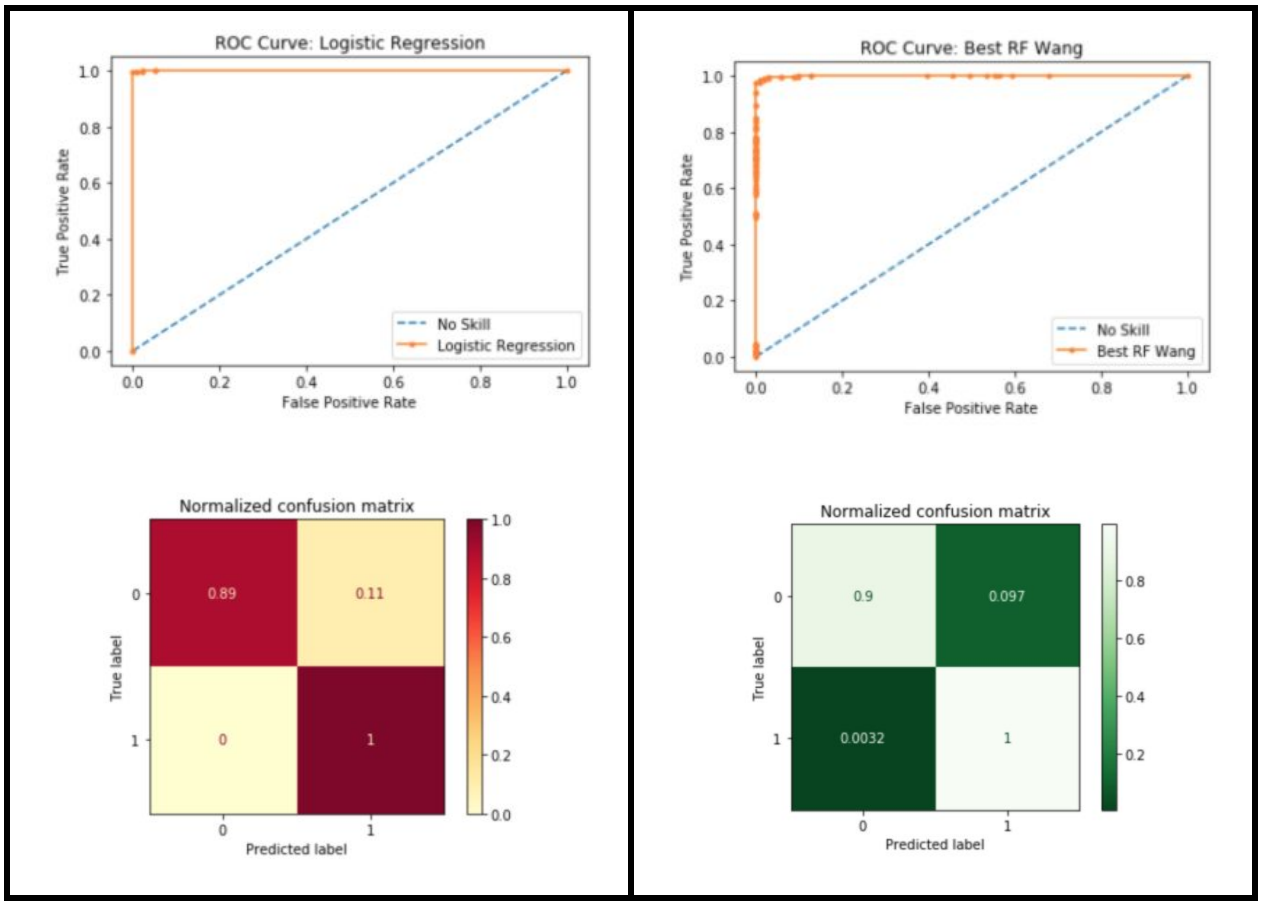


Fig 3.2: Results of baseline VS RF+GSO on experiment with Dog_3

Below is a summary of the ROC AUC scores obtained for each model:

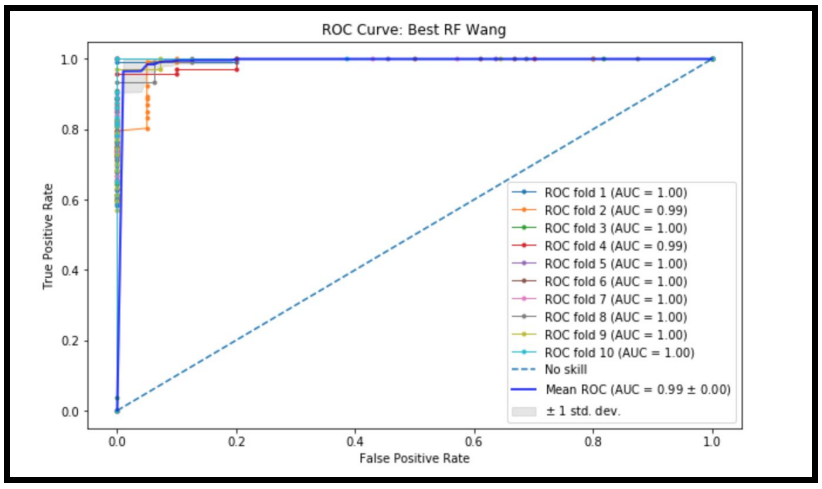
Dataset \Model	Dog_1	Dog_2	Dog_3	Dog_4	Dog_2_3_4*
Baseline	0.981	0.973	0.995	0.996	0.893
CNN	0.970	0.942	0.986	0.926	0.883
Mursalin RF	0.988	0.998	0.998	0.999	0.905
Wang RF+GSO	0.987	0.996	0.999	1.000	0.914
SVM	0.995	0.999	0.988	1.000	0.891
ExtraTrees	1.000	1.000	0.990	1.000	0.910
XGBoost	0.997	1.000	0.997	0.997	0.936
Adaboost	1.000	0.999	0.997	1.000	0.918

Table 3.1: ROC AUC scores on dog subject data. *For subject *Dog_2_3_4*, we used Dog_1 for testing and the other three dogs for training.

An interesting observation from this is that all the models obtained subpar scores when we switched from training and testing per subject basis to using data from dogs 2 - 4 for training and dog 1 for testing. However, the scores are high enough to earn some confidence that the models can generalize well with new subject data. In practice, this matters more in situations where clinicians have new patients.

3.3 Results: Human Subject Data

Despite the human data being more complicated, we observed that classification errors were just as low as on the dog data. For example, the same Random Forest with Grid Search Optimization model by Wang et al. (2019) which obtained good results on the seizure detection on dog data had the following results on the human data (patient 8):



```
=== Classification Report ===
              precision    recall  f1-score   support

           0       1.00      0.90      0.95        31
           1       0.99      1.00      1.00       347

    accuracy                           0.99       378
   macro avg       1.00      0.95      0.97       378
weighted avg       0.99      0.99      0.99       378
```

Fig 3.3: ROC curves and classification report for Patient_8 experiment with the Random Forest with grid search optimization model.

Below we show a tabular summary of how each model performed on human patient data.

Dataset\ Model	Pat_1	Pat_2	Pat_3	Pat_4	Pat_5	Pat_6	Pat_7	Pat_8
Baseline	1.000	0.956	1.000	1.000	1.000	1.000	1.000	0.980
CNN	0.923	0.915	0.924	0.912	0.922	0.893	0.955	0.969
Mursalin RF	1.000	0.961	1.000	1.000	1.000	1.000	1.000	0.979
Wang RF+GSO	1.000	0.971	1.000	1.000	1.000	1.000	1.000	0.976
SVM	1.000	0.952	0.999	1.000	0.998	1.000	0.995	0.966
ExtraTrees	1.000	0.962	1.000	1.000	1.000	1.000	1.000	1.000
XGBoost	0.997	0.956	1.000	1.000	1.000	1.000	1.000	0.998
Adaboost	1.000	0.993	1.000	1.000	1.000	1.000	1.000	1.000

Table 3.2: ROC AUC scores on human subject data.

We take the Patient 8 as an example to show the ROC AUC score and confusion matrix of different algorithms.

- **Baseline vs CNN**

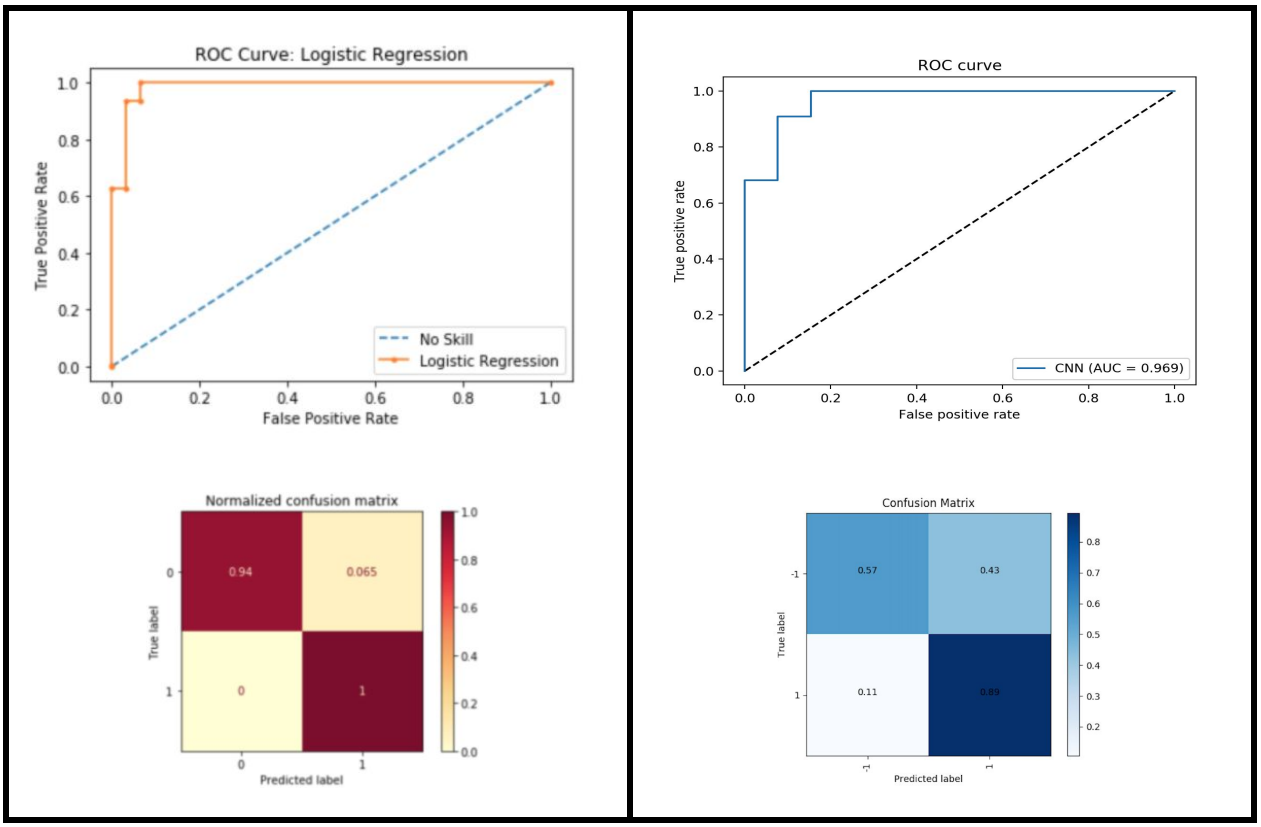


Fig 3.4: Results of baseline vs CNN on experiment with Patient_8

- **Baseline vs SVM vs ExtraTreesClassifier**

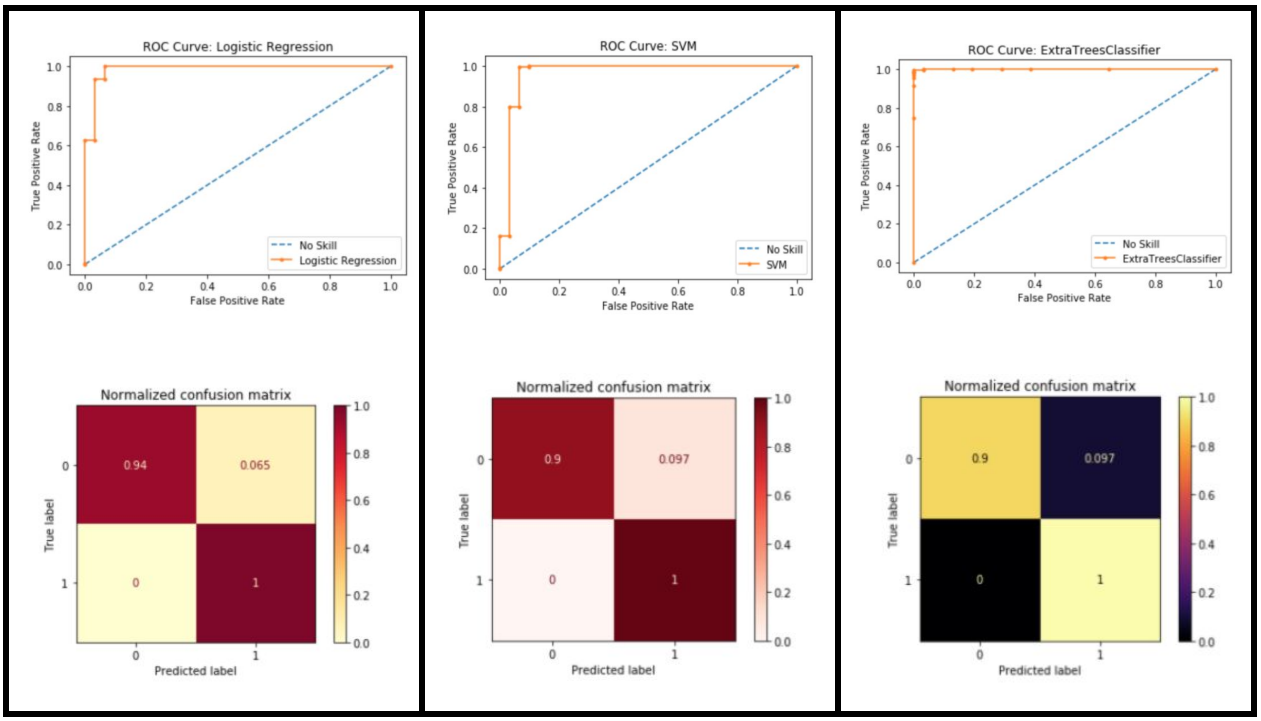


Fig 3.5: Results of baseline vs SVM vs ExtraTreesClassifier on experiment with Patient_8

- **Baseline vs XGBoost vs Adaboost**

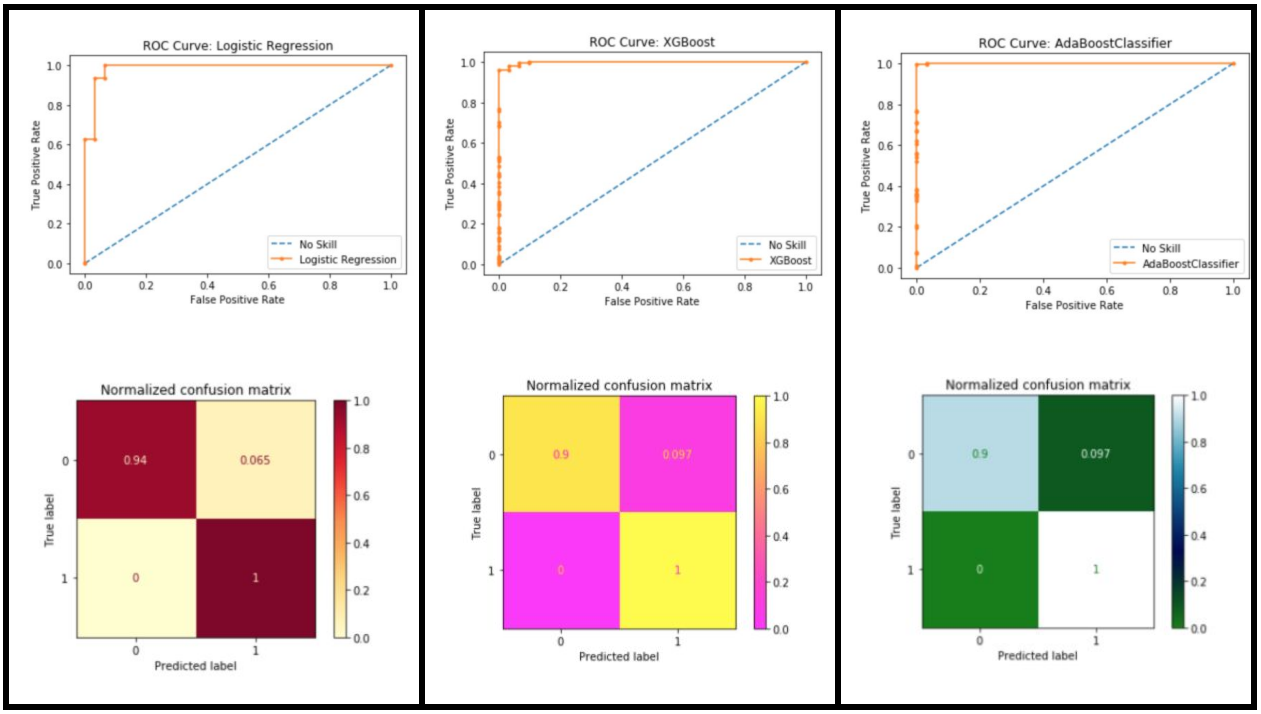


Fig 3.6: Results of baseline vs XGBoost vs Adaboost on experiment with Patient_8

4. Prediction Tasks

Now that we successfully completed the first task of developing an algorithm with high sensitivity and specificity that is able to accurately predict whether a given clip is a seizure or not; we consider the second task of predicting the probability that the clip occurred within 15 seconds of its respective seizure. In each of our classification models, we compute the prediction probabilities already. The challenge then becomes: if a segment is ictal, further categorize the seizure as 'early seizure' if the seizure occurs within the first 15 seconds, otherwise classify as 'non-early seizure'. Our models currently do not compute the probability of a clip being 'early' given that it is 'ictal'. To do this, we would adapt the model used in the first task such that instead of predicting 'ictal' or 'interictal', it predicts 'early' or 'late' given 'ictal' data. Then use Bayesian formula as below, once we have the predicted probability:

$$P('early' \mid x) = P('ictal' \mid x) * P('early' \mid 'ictal') \qquad (1)$$

where x is a given feature vector; $P('ictal'|x)$ is the probability computed from the first task; $P('early' \mid 'ictal')$ is the probability that a clip has early seizure given that it is ictal (basically occurred within 15 seconds) and we obtain this from applying the model on task 2. That's all that is needed for the second task. Since we weren't originally planning on submitting on the Kaggle leaderboard, we only focused on the first task, which would be easily adapted for task two.

Likewise, it is possible to extend our implementation to the Kaggle American Epilepsy Society Seizure Prediction Challenge. The goal of this challenge is to predict, for each clip in the test set, a real-valued probability that a given clip is *preictal*. Akin to the Seizure Detection Challenge which we focused on, the training data for this competition is organized into ten minute EEG clips labeled "Preictal" for pre-seizure data segments, or "Interictal" for non-seizure data segments.

5. Discussion

Data Processing and Feature Extraction: The time series data we used for seizure detection contains different numbers of channels and different sampling frequencies. Therefore, we extracted features and conducted analysis per subject wise. As a preprocessing step, we only applied mean-removal and handled low-pass filtering during our feature extraction process. During feature extraction, we explored the data with different tools and different analysis before designing the extraction pipeline. By visualizing the results at each step in our pipeline, we could verify the validity of our analysis and extraction. Eventually, we extracted top-k features from frequency domain values and statistical features from wavelet domain values per channel per subject. The concatenation of features across channels per subject is used to generate the complete training and testing data of our classification model.

Model Selection: We did a lot of experimentation with the various models in order to identify the best model and techniques for classification. For each experiment, we computed the ROC AUC score, which is the scoring criteria used by the competition leaderboard. For the most part, we used the Scikit-learn machine learning library to implement many of the classifiers. We also used k-fold Cross Validation in our approach, but that greatly increased the training time. Among the three implementations (Truong - CNN, Mursalin-RF, and Wang - RF with Grid Search Optimization), the latter had the best results overall and was our preferred algorithm for seizure detection. However, training this algorithm is computationally expensive and its usage in real life might not be practical. From experiments with other models, we found out that AdaBoost was not only fast but tended to be consistently more accurate in experiments with both dog and human data.

Parameter Optimization: For CNN and Random Forest with Grid Search Optimization, we performed some parameter tuning in order to determine the best set of parameters that maximizes performance. For the latter, this tended to vary based on the subject. It was interesting to note that our implementation obtained great results yet used way less estimators than the winning solution by Michael Hills which had the following architecture:

```
RandomForestClassifier(n_estimators=3000, min_samples_split=1,
        bootstrap=False, random_state=0)
```

Although the Random Forest with Grid Search Optimization gave optimal and consistent performances, it took way too long to train. In some cases, the training time exceeded that of all the models combined. Besides this, the other limitation of this method, according to the authors, is that if the noise in the EEG signals is too high, it will affect the detection of epilepsy. However, our preprocessing steps overcame this limitation, as explained by the results.

For CNN, several parameters need to be tuned to get a good result. Training CNN on a small data set (less than the number of parameters) will greatly affect the ability of CNN to generalize, usually leading to overfitting. In our work, we first refer to the relevant paper and use the parameters given in the paper as our initial parameters. Then we tune the learning rate and regularization terms, trying in order of 10. We also change the layers of CNN from shallow to deep. Besides, batch size and dropout are both important parameters for CNN.

6. Conclusion

Collectively, our findings in this project show that automated algorithms can successfully detect seizures in intracranial EEG data. We implemented three papers corresponding to three models: convolutional neural networks (CNN) by Truong et al, 2018, random forests (Mursalin et al., 2017), and random forests with grid search optimization (Wang et al, 2019). Our experiments also included a few other algorithms such as support vector machines, Extra Trees classifier, Extreme Gradient Boosting (XGBoost), and Adaptive Boosting (AdaBoost). As our experiments revealed, among our three implementations, Random Forest based approaches offered the best performance overall on detecting seizures compared to CNN. Surprisingly, the AdaBoost classifier proved not only to be faster, but more effective and consistent in both experiments on human and dog data. This then would be more desirable where we want a model that is capable of generalizing to new subjects.

One difficulty with the Random Forest with Grid Search Optimization model, by Wang et al., was the prolonged training time when using k-fold cross-validation. Though the process is computationally expensive, it could be useful to further explore other cross-validation techniques such as importance-weighted cross-validation and hopefully lower the training time. Other extensions to this project could include applying the Random Forest models on the prediction challenge, which we could not complete due to time constraints.

Source Code:
We pushed our project code to https://github.com/yazhuo/Seizure-Detection

References

Mursalin et al. (2017). Automated epileptic seizure detection using improved correlation-based feature selection with random forest classifier. Neurocomputing, ISSN: 0925-2312, Vol: 241, Page: 204-214.
https://www3.nd.edu/~dial/publications/mursalin2017automated.pdf

Truong, N. D., Nguyen, A. D., Kuhlmann, L., Bonyadi, M. R., Yang, J., Ippolito, S., & Kavehei, O. (2018). Convolutional neural networks for seizure prediction using intracranial and scalp electroencephalogram. *Neural Networks*, *105*, 104-111.
https://www.sciencedirect.com/science/article/abs/pii/S0893608018301485

Wang, X., Gong, G., Li, N., & Qiu, S. (2019). Detection Analysis of Epileptic EEG Using a Novel Random Forest Model Combined With Grid Search Optimization. Frontiers in human neuroscience, 13, 52.
https://doi.org/10.3389/fnhum.2019.00052 or https://www.ncbi.nlm.nih.gov/pmc/articles/PMC6393755/