Capstone Project Jesse Xing
Machine Learning Engineer NanoDegree September 13, 2019

Epileptic Seizure Detection

Definition

Project Overview

Epilepsy is among the most common neurological disorders in the world, second only to stroke, and affects over 50 million people around the globe (with 3 million just in the US). Unpredictable seizures are one of the most debilitating aspects of epilepsy and when uncontrolled, can cause severe limitations to an individual's ability to lead a normal life. During the event of a seizure, a subject may become confused or lose consciousness, resulting in complete loss of memory of the event. Stronger seizures can even cause spasms and uncontrollable muscle twitches which combined with nonconsciousness may result in severe injuries to the sufferer or even death in the worst case. In such scenarios, a seizure detection device which enables a caretaker to quickly aid the subject or triggers real time therapy that limits the complications of the seizure, could have huge life-saving potential. While many current methods of seizure detection have high levels of sensitivity (true positive rates >90%), they also suffer primarily from low levels of specificity resulting in a high number of false alarms. As a result, a significant amount of research in recent years has been focused on exploring new methodologies to detect seizures using EEG (brain), ECG (heart), and EMG (muscle) signals (1). In this project, I test a number of different machine learning methods on a dataset consisting of EEG from 500 epileptic patients and access their ability to identify the presence of a seizure.

Problem Statement

The goal of this project is to compare and analyze the performance of various machine learning methods in their ability to detect seizures from EEG signals which can be obtained from commercially available wearable devices. Both traditional classification (e.g. naïve bayes, logistic regression, and random forest) and deep learning (neural networks) methods were assessed to determine the strengths and weaknesses of each at recognizing seizure events from time series EEG data of 500 epileptic patients collected from the UCI Machine Learning Repository. Both models were expected to perform a binary classification of seizure or no seizure and their test accuracy scores were evaluated against the labels provided in the dataset. This research would hopefully aid in the development of wearable devices that more accurately signal the event of seizures in epileptic patients which could greatly aid in their mobility and reduce their risk of injury or death.

Metrics

Accuracy score was used as one of the evaluation methods for optimization as it takes into account the ability of the model to identify both labels of seizure and no seizure correctly. In this score, both true positives and true negatives get equal weighting, and the higher the ratio, the fewer false positives and false negatives there are to trigger false alarms or fail to alarm at all.

$$accuracy = \frac{true\ positives + \ true\ negatives}{Total\ dataset}$$

F- beta score was used to find out the precision $(\frac{\text{true positives}}{\text{all positives}})$ and recall $(\frac{\text{true positives}}{\text{true positive + false negative}})$ of the models. (Precision refers to the percentage of results that are relevant and recall gives the percentage of relevant results that are correctly classified). For this application, a higher recall may be necessary to reduce the number of false negatives which may result in injury to epileptic patient. Since the data set is imbalanced towards non-seizure samples (see data exploration), F-score may be crucial for ensuring the model accurately classifies seizure patients correctly as a high accuracy model may simply classify patients as non-seizure correctly.

$$F_\beta = (1+\beta^2) \cdot \frac{\text{precision} \cdot \text{recall}}{(\beta^2 \cdot \text{precision}) + \text{recall}}.$$

Analysis

Data Exploration

For this experiment, we use data obtained from Kaggle under the title Epileptic Seizure Recognition (2), which is a preprocessed/re-structured version of a commonly used dataset featuring epileptic seizure detection. The original dataset consists of a total of 500 EEG recordings, each from a different subject, containing time series data lasting a duration of 23.6 seconds and sampled into 4097 data points. In the Kaggle repository, each of these EEG recordings are further subdivided into 23 chunks containing 178 data points representing 1 second intervals of time. As a result a total of 500 * 23= 11,500 rows of data, each with 178 time series data points of EEG values and 1 response variable (marked as classes 1 – 5) was obtained and then stored in a csv file for easy accessibility to researchers. Note: Only subjects in class 1 are experiencing a seizure. Classes 2, 3, 4, and 5 represent different states of subjects who did not have an epileptic seizure. Each class contains an equal 2300 samples. For the purposes of this study, subjects in classes 2-5 were marked as 0 to represent a state of no seizures while labels of 1 will be left alone to represent a state of seizures. As a result, there was a 4:1 class imbalance between non seizure and seizure patients.

The original dataset is collected from the UCI Machine Learning Repository

Exploratory Visualization

Below (Figure 1) are plots of 4 samples of the dataset, with the upper two being patients experiencing no seizure and the lower two of subjects experiencing seizure. From an early preliminary analysis, it seems that the frequency of the EEG signal is much higher in the subjects who don't experience any seizure compared to the ones that are in a state of seizure. As a result, a simple Fourier transform analysis measuring the predominant frequency component in the EEG signal may be the best feature for seizure detection (input for traditional classification models).

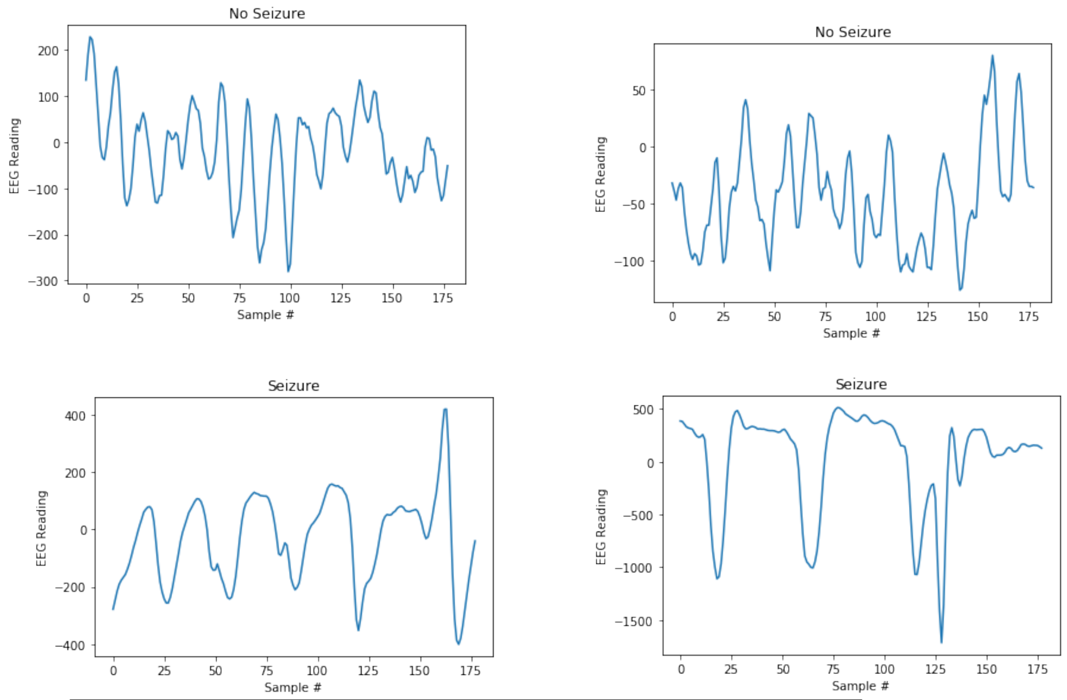


Figure 1: Raw EEG Signals of Seizure and Non-Seizure Patients

Algorithms and Techniques

Two distinct classes of machine learning algorithms were be used for training on the EEG dataset for seizure detection: traditional classification models and deep learning models. To create a proper training and testing set, the 11,500 samples was be split at a ratio of 2 : 1 (training : testing), at random using the train_test_split function embedded in sci-kit learn. These data sets were then be used consistently throughout the rest of the protocol to train and validate the aforementioned models under two distinct paradigms, one suited for each class (more details in Methodology section). Parameters for each model were adjusted to allow for optimal performance and a comparison of all models was conducted at the conclusion using a set number of evaluation metrics (as defined above).

The classification models were trained on a series of features that can be extracted from the time series data since individual EEG sample values have little relationship with the final output. These included max and min values, average signal frequency, standard deviation, and median among other basic descriptive statistics. Four key classification models were tested for their effectiveness – random forest, logistic regression, support vector machines, and Naïve Bayes (at first using just the default parameters in sci-kit learn). The most effective model out of the four (random forest))had its training parameters optimized using grid search and cross validation with the goal of comparing its performance to that of neural networks.

Contrarily, since convolutional neural networks have been shown to extract relevant features from image data on their own accord, it was passed time series EEG data that had not been preprocessed as inputs with the expectation that certain layers in the model would autonomously filter out relevant information. Neural networks will assign a probability for each class, which will allow the setting of a threshold to optimize true positive rates while minimizing false positive rates. This classification threshold along with the overall neural network architecture (layers types, activation functions, number of nodes at each layer, etc.) and other training parameters (epochs, solver type, loss function, etc.) were altered until an optimal solution is reached. A TowardsDataScience tutorial (3), that creates a convolutional neural network in Keras for sleep stage classification was used to establish the initial base algorithm (and set initial parameters) for the deep learning seizure detection model.

Benchmark

For the benchmark comparison, we will use data collected from a scientific paper written by Ramgopal et al in 2014 (1) which summarizes a number of different machine learning methods (e.g. neural networks, linear regression, SVM, etc.) used to train EEG data for seizure detection and prediction as well as their level of sensitivity (TP/P rate) and specificity (TN/N rate). Both these metrics can be calculated for the models that we train in our experiment.

From an initial analysis of the model performance as outlined in the paper, it seems that a good target for both sensitivity and specificity is any value greater than 95%. Only the best models outlined in the paper can perform with a sensitivity level between 90-100 and a specificity nearing 100. (See table 1 in Appendix for model performance). As data sets may vary, for the purposes of this study, if we can get within 5 basis points of the best performing models, we would have proven that fairly basic ML models implementable through the sci-kit learn library can achieve almost as good or even better (if probably optimized) results than those created by professionals for the task of seizure detection.

Methodology

Data Preprocessing

For both paradigms, the 11,500 samples in the randomized data set is divided using the train_test_split function at a ratio of 2:1 (random_state=42), so that 7705 of those samples are training data and 3795 are testing data.

In addition, within the output vector, labels 2-5 were all relabeled as 0 to represent subjects in a state of non-seizure while labels of 1 were left alone to represent a state of seizures. (see Data exploration section for reasoning)

Paradigm 1: Traditional Classification Models

A series of different features are first extracted from the raw EEG signals before any traditional classification model can be trained. The reason why this preprocessing step must be conducted is because the raw EEG values are time series data. Although each sample has a consistent number of feature points (178 EEG values for every 1 second of data), patterns of meaning within the data set are not attributed to any one feature and are really an amalgamation of every value within the time interval of interest. Also data points (features) of interest may be shifted depending on when the EEG signal was being recorded. As a result only group statistics of the entire 178 raw EEG values are taken and input as features into our models.

The first step that needs to be conducted is a Fourier transform of each sample to convert the time series data into the frequency domain. Since we observed from the direct plots of the EEG signals that seizure patients seemingly have a lower frequency than their non seizure counterparts, a conversion into the frequency domain would allow us to directly analyze the frequency differences between subjects. The plots below show the Fourier transform or a seizure and non-seizure patient. From the plots, we see that the non-seizure patient has a peak in frequency that is much higher than that of the seizure patient, which corroborates our initial observations from the raw signal.

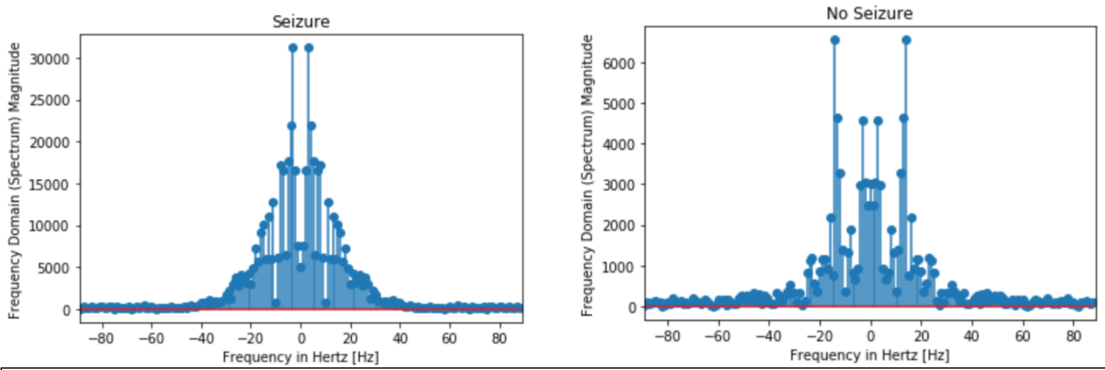
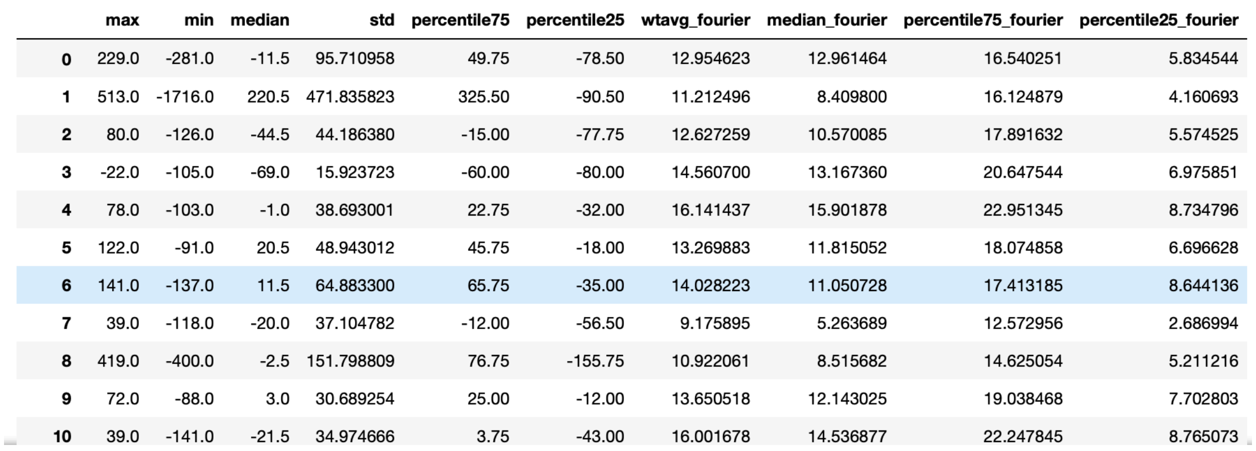


Figure 2: Fourier Transform of Raw EEG Signal for Seizure and Non-Seizure Patient

The final feature set that is extracted to serve as training inputs into the classification models includes the following metrics:

- Max, min, and standard deviation of original EEG signal
- 25th percentile, median, and 75th percentile values of original signal
- Weighted average (frequency) of Fourier transform
- 25th percentile, median, and 75th percentile of Fourier transform

The table below shows those extracted features for a set of 11 samples.



	max	min	median	std	percentile75	percentile25	wtavg_fourier	median_fourier	percentile75_fourier	percentile25_fourier
0	229.0	-281.0	-11.5	95.710958	49.75	-78.50	12.954623	12.961464	16.540251	5.834544
1	513.0	-1716.0	220.5	471.835823	325.50	-90.50	11.212496	8.409800	16.124879	4.160693
2	80.0	-126.0	-44.5	44.186380	-15.00	-77.75	12.627259	10.570085	17.891632	5.574525
3	-22.0	-105.0	-69.0	15.923723	-60.00	-80.00	14.560700	13.167360	20.647544	6.975851
4	78.0	-103.0	-1.0	38.693001	22.75	-32.00	16.141437	15.901878	22.951345	8.734796
5	122.0	-91.0	20.5	48.943012	45.75	-18.00	13.269883	11.815052	18.074858	6.696628
6	141.0	-137.0	11.5	64.883300	65.75	-35.00	14.028223	11.050728	17.413185	8.644136
7	39.0	-118.0	-20.0	37.104782	-12.00	-56.50	9.175895	5.263689	12.572956	2.686994
8	419.0	-400.0	-2.5	151.798809	76.75	-155.75	10.922061	8.515682	14.625054	5.211216
9	72.0	-88.0	3.0	30.689254	25.00	-12.00	13.650518	12.143025	19.038468	7.702803
10	39.0	-141.0	-21.5	34.974666	3.75	-43.00	16.001678	14.536877	22.247845	8.765073

Figure 3: Table of Features inputted into Traditional Classification Model

Paradigm 2: Neural Network Models

The preprocessing for the Neural Network Models is comparatively simpler, as the complexity of the model itself should allow it to extract out meaningful information from the time series EEG data on its own accord. As a result the input to the convolutional neural network was simply the raw EEG values in vectors of size 178 to represent 1 second of data for each subject.

In a comparison study we observe whether or not the performance of the convolutional network can be improved by training on only the Fourier transform of the EEG signal (each data point represents a separate frequency value from -89-89 Hz). The theory is that the Fourier transform of the EEG signal may be better for training because all features would only be frequency dependent (which we saw had an impact on the detection of seizure or non-seizure) and not time dependent, so any time shifts in the data collection process would not have any impact on the result. As a result, a training and testing input set of the EEG samples converted to the frequency domain using a Fourier transform was obtained. (This data set was also used to train a significantly simpler Neural Network without any convolutional layers to see if a less complicated and therefore faster model could return similar results).

For neural network training, a validation set is also required. In order to obtain this, we divide the training set further using the train_test_split function (again with a 2:1 split for training:validation), so that 5162 samples of the training set are now marked as the new training set and 2543 samples are marked as a separate validation set.

Implementation

Paradigm 1: Traditional Classification Models

To implement our traditional classification models (Logistic Regression, Gaussian Naïve Bayes, AdaBoost Classifier, and Random Forest Classifier), first we declare them (and save them as models 1-4) in a code block with default parameters. We pass each model to a function train_pred (for training and prediction purposes - further discussion below) and save the results of each into a dictionary called results, so that we can plot the model metrics at a latter step.

```
model = LogisticRegression(random_state=1)
model2 = GaussianNB()
model3 = AdaBoostClassifier()
model4 = RandomForestClassifier(n_estimators=100,max_depth=5,random_state=10)

results = {}

for mod in [model,model2,model3,model4]:
    clf_name = mod.__class__.__name__
    results[clf_name] = train_pred(mod,processed_train,y_train_np,processed_test,y_test_np)
```

The train_pred function below takes as input a model (learner), training feature set (X_train), training output set (y_train), testing feature set (X_test), and testing output set (y_test). It calls the fit function on the learner using the training feature and output set (X_train and y_train) to properly train the model. A prediction using the fitted model is then performed on both the test feature set and training feature set (X_test and X_train) and the results are saved in a dictionary for future reference. Both the accuracy score and F-beta score (beta =.5) for the training and testing predictions in comparison to their desired outputs in y_train and y_test are calculated and saved to the same dictionary. Lastly the function saves the confusion matrix as a vector of (True Negative, False Positive, False Negative, and True Positive) for both training and test predictions.

```
def train_pred(learner,X_train,y_train,X_test,y_test):

    results={}
    learner = learner.fit(X_train,y_train)
    predictions_test = learner.predict(X_test)
    predictions_train = learner.predict(X_train)

    results['y_test'] = predictions_test

    results['y_train'] = predictions_train

    results['acc_train'] = accuracy_score(y_train,predictions_train)
    results['acc_test'] = accuracy_score(y_test,predictions_test)
    results['f_train'] = fbeta_score(y_train,predictions_train,beta=.5)
    results['f_test'] = fbeta_score(y_test,predictions_test,beta=.5)

    tn,fp,fn,tp = confusion_matrix(y_train,predictions_train).ravel()

    results['confusion_train'] = (tn,fp,fn,tp)

    tn,fp,fn,tp = confusion_matrix(y_test,predictions_test).ravel()

    results['confusion_test'] = (tn,fp,fn,tp)

    return results
```

Paradigm 2: Neural Network Models

Note: The following models are largely derived from a tutorial written by Mansar, Youness (3).

Part A:

In order to create our convolutional neural network (CNN) for seizure detection, first we build a base model using Keras that consists of alternating layers of 1D convolutional filters, Max Pooling layers, and spatial dropout layers. The purpose of the convolutional filters is to extract out features in the EEG signal (by taking some combination of data points within a rolling window of a specified kernel size). The pooling layer limits the number of extracted features (by taking the maximum value over a window) and reduces the spatial dimensionality of the network which speeds up its computational performance and lowers its chances of overfitting (limits the number of trainable parameters in subsequent layers). The dropout layer further reduces the final output of the convolutional neurons by selecting only a certain number of max pool outputs to keep and ignoring the rest. This is used to again reduce spatial dimensionality and the chances of overfitting by eliminating neurons that are codependent on one another. A total of 4 layers of convolutional neurons (with associated max pool and dropout layers) are stacked on top of each other in our base model, with each additional layer filtering out a larger set of features. Layers closer to the output (deeper) contain a larger number of filters to extract out more features than those closer to the input (earlier) similar to the way networks are typically setup for image processing (4). This allows us to filter out more complex features that are an amalgamation of basic features as we get deeper into the network while keeping the model relatively simple (fewer optimizable parameters) so that it can be trained quickly.

```
def get_base_model():
    inp = Input(shape=(WINDOW_SIZE, 1))
    img_1 = Convolution1D(16, kernel_size=5, activation=activations.relu, padding="valid")(inp)
    img_1 = Convolution1D(16, kernel_size=5, activation=activations.relu, padding="valid")(img_1)
    img_1 = MaxPool1D(pool_size=2)(img_1)
    img_1 = SpatialDropout1D(rate=0.01)(img_1)
    img_1 = Convolution1D(32, kernel_size=3, activation=activations.relu, padding="valid")(img_1)
    img_1 = Convolution1D(32, kernel_size=3, activation=activations.relu, padding="valid")(img_1)
    img_1 = MaxPool1D(pool_size=2)(img_1)
    img_1 = SpatialDropout1D(rate=0.01)(img_1)
    img_1 = Convolution1D(32, kernel_size=3, activation=activations.relu, padding="valid")(img_1)
    img_1 = Convolution1D(32, kernel_size=3, activation=activations.relu, padding="valid")(img_1)
    img_1 = MaxPool1D(pool_size=2)(img_1)
    img_1 = SpatialDropout1D(rate=0.01)(img_1)
    img_1 = Convolution1D(256, kernel_size=3, activation=activations.relu, padding="valid")(img_1)
    img_1 = Convolution1D(256, kernel_size=3, activation=activations.relu, padding="valid")(img_1)
    img_1 = GlobalMaxPool1D()(img_1)
    img_1 = Dropout(rate=0.01)(img_1)

    dense_1 = Dropout(0.01)(Dense(64, activation=activations.relu, name="dense_1")(img_1))

    base_model = models.Model(inputs=inp, outputs=dense_1)
    opt = optimizers.Adam(0.001)

    base_model.compile(optimizer=opt, loss=losses.sparse_categorical_crossentropy, metrics=['acc'])
    #model.summary()
    return base_model
```

Part B:

The following code depicts how we utilize the base model we created in part A. We add a TimeDistributed layer which allows us to input time series EEG data over multiple 1 second periods at 178 Hz into our base model and achieve a result that is the combination of several seconds worth of data. It works by cutting a long time series EEG dataset into 1 second long intervals, each with 178 data points, and passing individual sections separately into our base model. For each 1 second interval, a different output is produced by our base model, and all outputs are passed through a dropout layer and convolution layer to determine the final class designation (either seizure or non-seizure). For our particular use case, the TimeDistributed layer may be overkill as all our data is only 1 second long, but for use in the real world, where data can be collected continuously over an indefinite period, a model that uses historical data to predict the presence of a seizure may be more robust.

```
import keras.backend as K

def get_model_cnn():
    nclass = 2

    seq_input = Input(shape=(None, WINDOW_SIZE,1))#Input(shape=(None, WINDOW_SIZE, 1))
    base_model = get_base_model()

    base_model.summary()
    encoded_sequence = TimeDistributed(base_model)(seq_input)
    encoded_sequence = SpatialDropout1D(rate=0.01)(Convolution1D(128,
                                                          kernel_size=3,
                                                          activation="relu",
                                                          padding="same")(encoded_sequence))
    encoded_sequence = Dropout(rate=0.05)(Convolution1D(128,
                                                          kernel_size=3,
                                                          activation="relu",
                                                          padding="same")(encoded_sequence))

    out = Convolution1D(nclass, kernel_size=3, activation="softmax",
padding="same")(encoded_sequence)

    model = models.Model(seq_input, out)

    def custom_loss_function(y_true,y_pred):
        #predicted = np.argmax(y_pred)
        #print(y_true)
        #print(y_pred)
        pred = K.slice(y_pred,[0,0,1],[-1,-1,1])
        return losses.binary_crossentropy(y_true,pred)

    def accuracy_mod_func(y_true,y_pred):
        return K.mean(K.equal(y_true,K.round(K.slice(y_pred,[0,0,1],[-1,-1,1]))))

    model.compile(optimizers.Adam(0.001), custom_loss_function, metrics=[accuracy_mod_func])
    model.summary()
    return model
```

The model is finally compiled and uses the binary_crossentropy loss function, accuracy metric, and Adam's optimizer for training on the raw EEG and Fourier transform data. (Note: custom loss and accuracy function wrappers were written to compensate for the loss of two dimensions in the y_true dataset (which is only 1 dimensional) due to the requirement of 3 dimensions in the output of the convolutional neural net)

Refinement

Paradigm 1: Traditional Classification Models

Grid search and cross validation is used optimize the parameters for the best classification model out of the four that we tried. In our case Random Forest had by far the best test accuracy and F-score at 0.978 and 0.950 respectively. In comparison the next runner up, an Adaboost classifier, had corresponding accuracy and F-scores of 0.975 and 0.937 (see results section for full comparison of unoptimized models). We use the shuffle-split function to create the cross validations sets on the training data (10 splits with 80:20 train:test ratio in each split). Next we set the grid search parameters to vary the number of estimators (5,10,50,100,500) and max depth (1,10,20,30) of the random forest algorithm. A lower value for both parameters will result in a simpler model while a higher value will cause the model to overfit the dataset. The GridSearchCV function is used to optimize the parameters of the Random Forest model on the training set and to return the optimal model.

```
def performance_metric(y_true,y_predict):
    score = accuracy_score(y_true,y_predict)
    return score

def fit_model(X,y):
    cv_sets = ShuffleSplit(X.shape[0],n_iter=10,test_size=.2,random_state=0)
    regressor = RandomForestClassifier()
    scoring_fnc = make_scorer(performance_metric)

    params = {'n_estimators':[5,10,50,100,500],'max_depth':[1,10,20,30]}

    grid = GridSearchCV(regressor,params,scoring_fnc,cv=cv_sets)

    grid = grid.fit(X,y)

    return grid.best_estimator_
```

After running the optimization, the following model is obtained with optimal parameters for n_estimators and max_depth set at 500 and 30 respectively (the highest values in my parameter set). This suggests that a more complicated model (that likely overfits) seems to yield better results on the training data (split at 80:20 into training:validation sets). A slightly better test accuracy and f-score of 0.982 and 0.954 is obtained.

```
RandomForestClassifier(bootstrap=True, class_weight=None, criterion='gini',

            max_depth=30, max_features='auto', 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, n_estimators=500, n_jobs=1,

            oob_score=False, random_state=None, verbose=0,

            warm_start=False)
```

Paradigm 2: Neural Network Models

Since the original model performed so well (with well over 98 % test accuracy), only slight modifications were made to the convolutional neural net in order to optimize its performance. Since it seemed that a more complex Random Forest model worked the best at achieving the highest test accuracy, the same idea was applied to the modifying the CNN structure. By increasing the number of outputs in the final dense layer of our base model to 128, and then doubling the kernel size of our two 1D convolutional neural nets to 256 within the final processing layer, we increase the number of features that the model would extract before determining an output. In doing so, we also increase the number of trainable parameters from 325,170 to 563,826. In order to avoid a slowdown in the training process, I decided to only change the latter layers of the neural networks as that is where most of the final feature accumulation occurs. The changes yielded an improvement to the test accuracy from 0.98419 to 0.9884 (a change of roughly .004). While this change seems small, it is encouraging that it supports the initial hypothesis that by overfitting the training set, we can continue to improve upon the test accuracy. As such it can be reasoned that with a more powerful machine and an increasingly complex neural net, we can extract even better performance out of this particular dataset (although its generalization to real world seizure detection may not fully manifest to the same extent).

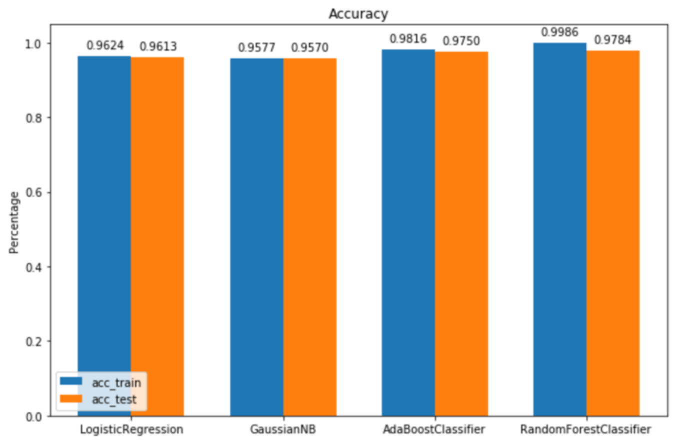
```
def get_base_model_refined():
    . . .
    dense_1 = Dropout(0.01)(Dense(128, activation=activations.relu, name="dense_1")(img_1))

    base_model = models.Model(inputs=inp, outputs=dense_1)
    opt = optimizers.Adam(0.001)
    . . .

def get_model_cnn_refined():
    . . .
    encoded_sequence = TimeDistributed(base_model)(seq_input)
    encoded_sequence = SpatialDropout1D(rate=0.01)(Convolution1D(256,
                                                              kernel_size=3,
                                                              activation="relu",
                                                              padding="same")(encoded_sequence))
    encoded_sequence = Dropout(rate=0.05)(Convolution1D(256,
                                                              kernel_size=3,
                                                              activation="relu",
                                                              padding="same")(encoded_sequence))

    out = Convolution1D(nclass, kernel_size=3, activation="softmax",
padding="same")(encoded_sequence)
    . . .
```

Results

Model Evaluation and Validation

Paradigm 1: Traditional Classification Models

Below (Figure 4) are plots of the accuracy and F-beta scores (beta =.5) for all four classification techniques (Logistic Regression, Gaussian Naïve Bayes, AdaBoost, and Random Forest) applied to both the training and testing sets using default Sci-kit learn parameters. In all the models, the training scores are slightly higher than the test scores which suggests that they are properly overfitted to their respective training sets. By far, the best accuracy and F-score for the testing set is achieved by the Random Forest model at .9784 and .9501 respectively in comparison to the next runner up which is the AdaBoost classifier at .9650 and .9367.



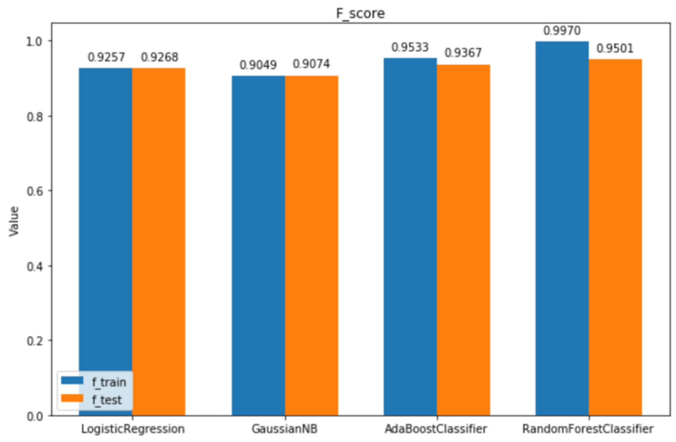
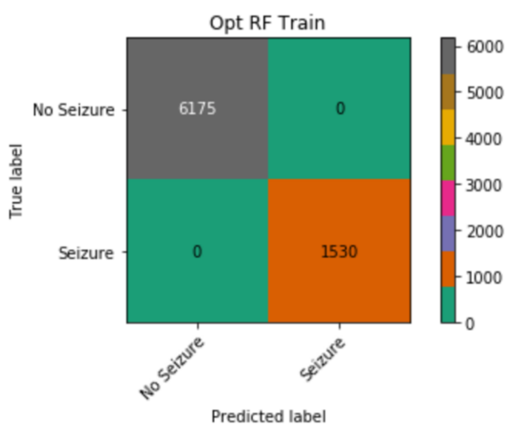


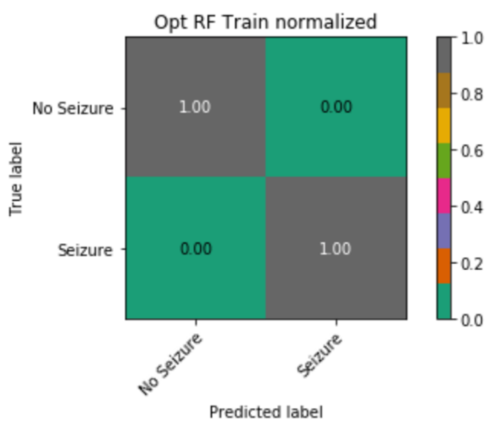
Figure 4: Accuracy and F-Score (Beta=0.5) for Classification Models on Training and Test Set with Default Parameters

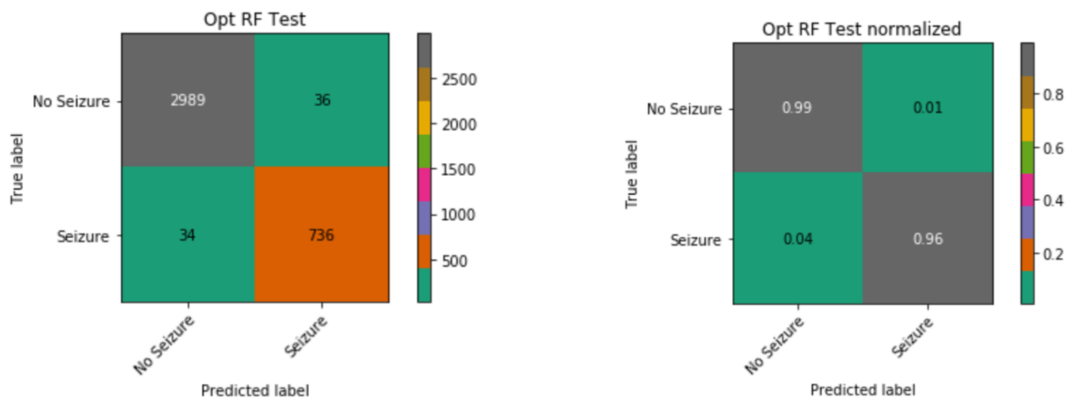
As previously discussed, the Random Forest Model was optimized over a range of different estimator values and max depth cutoffs, until the best possible model was achieved. This model had the maximum number of estimators and highest max depth in our parameter range at 500 and 30 respectively which suggests that a more complex model was better for our dataset. (Given a more powerful computer, even larger parameter sets can be tested to see whether a more optimal solution can be achieved with a more sophisticated model that heavily overfits the training set). The test accuracy and f-score of the optimal model obtained in this experiment was 0.982 and 0.954 respectively.

Optimized Random Forest	
Accuracy	F - score
0.982	0.954

Figure 5: Confusion Matrices for Optimal Random Forest Model on Training and Testing data sets



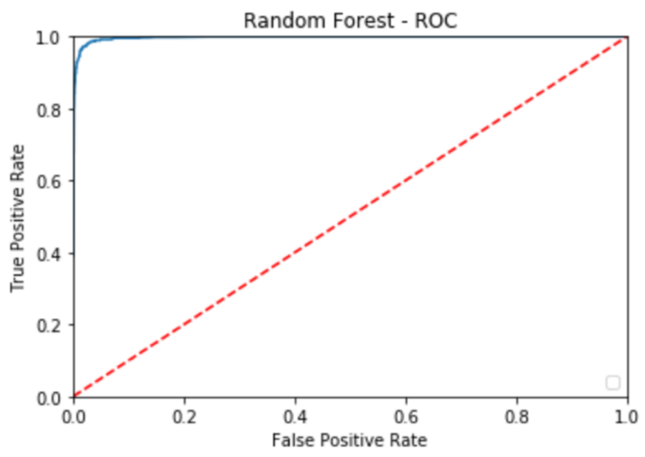




In Figure 5 above, we observe the confusion matrices for the optimized random forest model on both the training and testing data. From this, we can see that the model already fits our training set perfectly as there are no false-positives or false-negatives and the detection of seizure and non-seizure is at 100 %. However for test data set, the Random Forest was only able to achieve a 99 % true negative score and a 96% true positive score. While both these metrics are higher than our initial benchmark which only called for a specificity and sensitivity higher than 95 %, there is still room for improvement in both these areas as a 4% false negative rate may lead to seizures going undetected and a 1% false positive rate could lead to unwanted seizure warnings.

One possible means of further optimizing our model is by changing the probability threshold by which we classify events as seizure vs. non-seizure. Below in Figure 6 is a plot of the receiver operating characteristics curve for our optimized Random Forest Model which plots the false positive rate against the true positive rate for our test data set at different probability thresholds. The best model will have an area under the curve (AUC) of 1, hug the upper left hand corner, and allow for a 100% true positive rate with 0 false positives. This particular model has an AUC of .99745 and seemingly has a true positive rate of 90% when the false positive rate (which leads to unwanted seizure warnings) is 0.

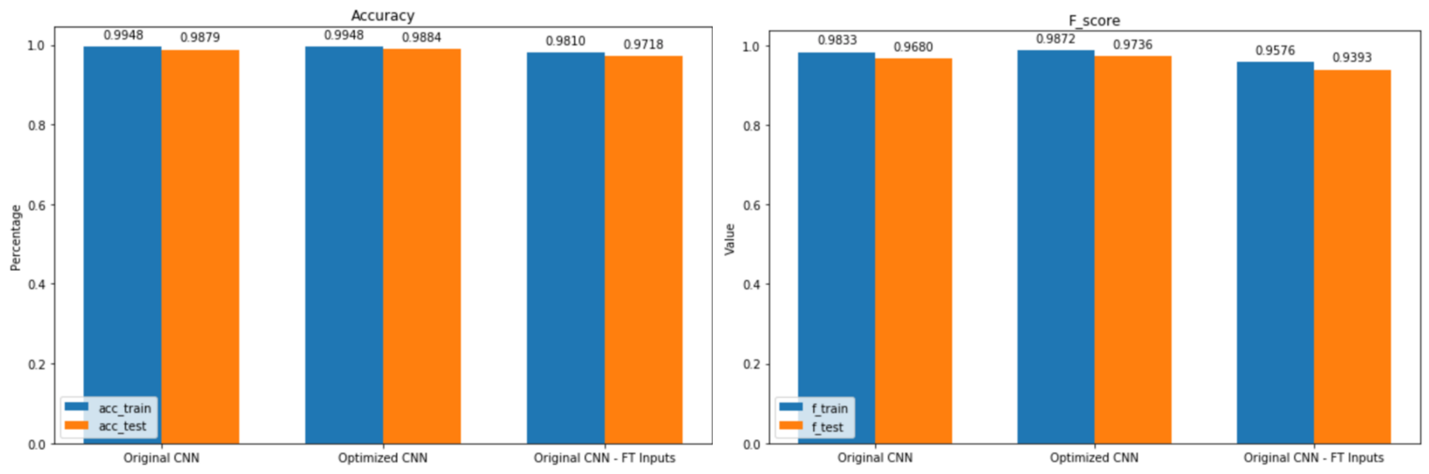
Figure 6: Receiver Operating Characteristics Curve for Optimized Random Forest Model



Paradigm 2: Neural Network Models

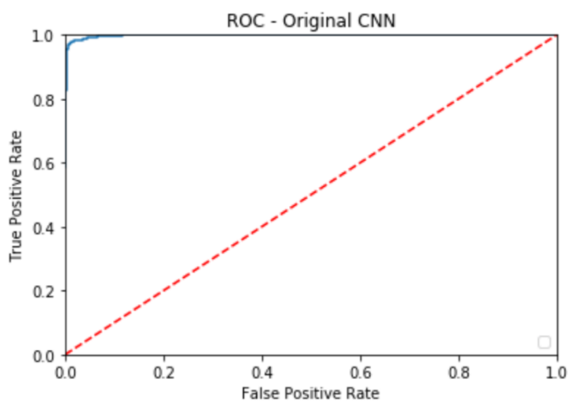
As discussed in the Methodology section, two CNN networks were trained on the raw EEG data, a default convolution neural net using parameters largely derived from a tutorial written by Mansar, Youness (3), and an optimized version that doubles the number of filters within the neural net in its final 3 layers. We also train the original neural net on the Fourier transform of the raw EEG signals to see if normalized input data set would yield better results (as discussed in Data preprocessing section). Figure 7 shows the accuracy and F-score of the training and test data sets for all three models. The optimized CNN only surpassed the original CNN by only a small margin in test accuracy (.9884 vs .9849), but performed slightly better by the F-score metric (.9736 vs .9680). This is somewhat expected, as the original CNN was already performing extremely well and small changes in its neural architecture could only have caused it to overfit the training set slightly. Interestingly, the Fourier transform of the EEG data as input into the original CNN had the worst training and test performance in terms of accuracy and F-score with test values of only .9718 and .9393 respectively. Contrary to our initial conjecture, this suggest that performing a Fourier transform before training a CNN is a redundant procedure that inhibits the natural ability of the neural nets to extract features and properly fit our dataset (as CNN are themselves implicitly applying a Fourier transform to the data set in a rolling window).

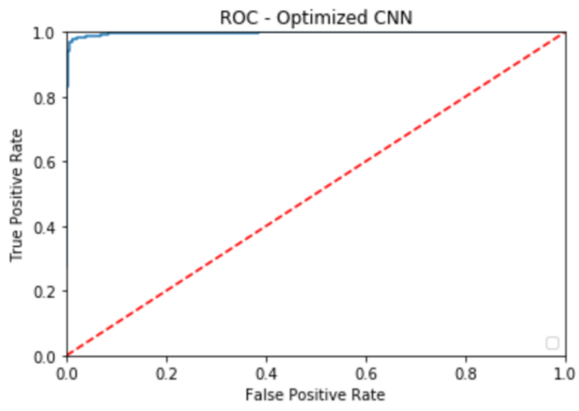
Figure 7: Accuracy and F-Score (Beta=0.5) for Neural Networks on Training and Test Set

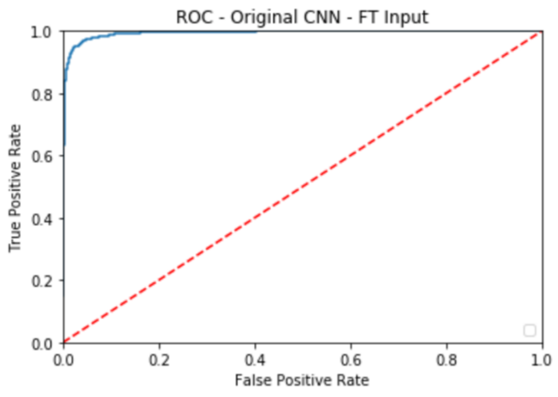


The Receiver Operating Characteristics curves in Figure 8 below further validates our initial observations. Both the Original and Optimized CNN models trained on unprocessed raw EEG data have a better ROC (curve hugs upper left corner) than then CNN trained on Fourier Transform data. However, it seems that the original CNN has a better AUC at 0.9988 than the optimized CNN at 0.9975. While the difference is small, it suggests that the original CNN may function better than the optimized version at different threshold levels (at the default threshold – 50 % the optimized version is better from accuracy and f-score analysis above). The optimized version may have been overfitting the training set and therefore only performs well under certain conditions whereas the original CNN is a bit more flexible. Both models can provide a sensitivity level of nearly 97% with a 0 percent false positive rate (100 percent specificity).

Figure 8: Receiver Operating Characteristics Curve/AUC for Neural Network Models







AUC of NN Models		
Original CNN	Optimized CNN	Original CNN- FT Input
0.99838	0.99748	0.99365

Justification

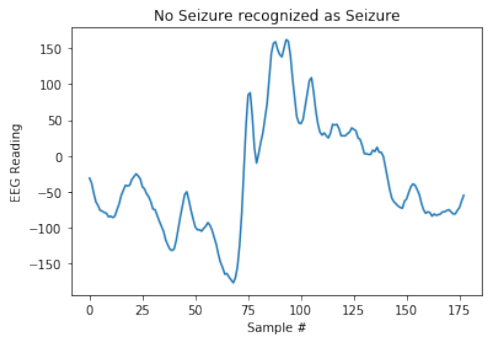
Both our Random Forest and Convolutional Neural Network models performed better than our target which only required a sensitivity and specificity higher than 95% as discussed earlier. In the real world, a very high specificity is required so as to not trigger alerts when the subject isn't seizing as this could interfere with his/her daily activities. It seems that Convolutional Neural Networks are the best solution to this problem as in this test, they can achieve a sensitivity level greater than 97% with nearly 100% specificity. (Note that this result is among the best of any of the benchmark models in our reference table see Appendix). The sensitivity could be even higher if the model was fed a sequence of EEG data longer than 1 second (while the model may not be able to detect a seizure in the first second, EEG data over multiple seconds may be more easily identifiable). Given that these results are extensible to real world EEG devices, with performance in this range, we can comfortably trigger seizure alerts on patients without worrying that it would be overly intrusive. However, since we cannot confirm that the EEG data used in this study is fully extensible, this study can only serve as a recommendation for the best model to be potentially used in a real world seizure detection device (as per our original goal).

Conclusion

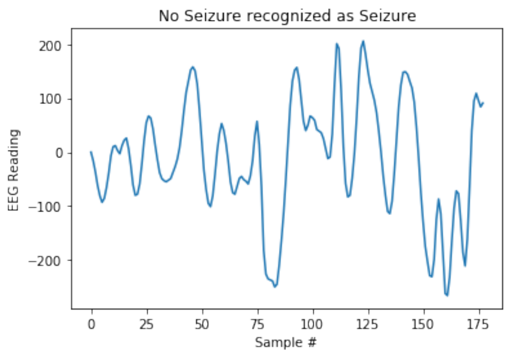
Free-Form Visualization

Since our models (both CNN and traditional classifiers) performed so well, it is more interesting in viewing the EEG samples of subjects that were misclassified than those that were classified correctly. Figure 9 below shows the plots of 5 subjects that were mislabeled as having a seizure when none were occurrent (False Positives) by our Optimized CNN model (our best performer). Other than the plot 2, the rest seem to have fairly low frequencies of oscillation, which by our initial conjecture should classify that subject in a state of seizure (as the model has done). Plot 2 seems to show a higher frequency of oscillation but the magnitudes are inconsistent which may make the signal a bit ambiguous. It seems that the samples that are misclassified by the model are also somewhat hard to classify visually by a human.

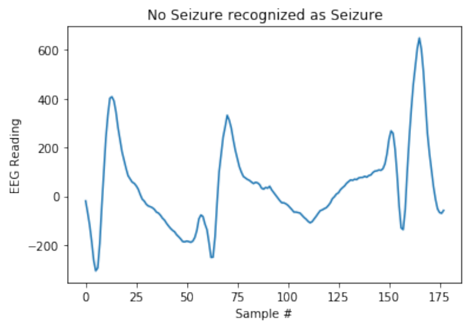
Figure 9: Analysis of False Positives in Test data (No Seizure being recognized as Seizure)



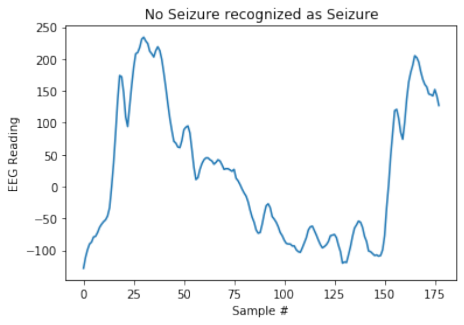
1



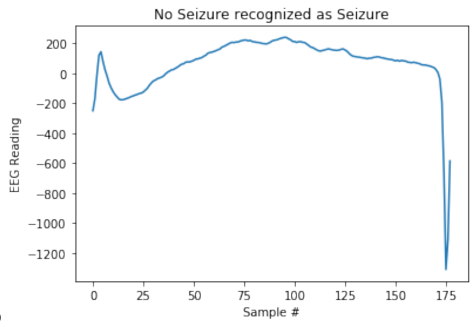
2



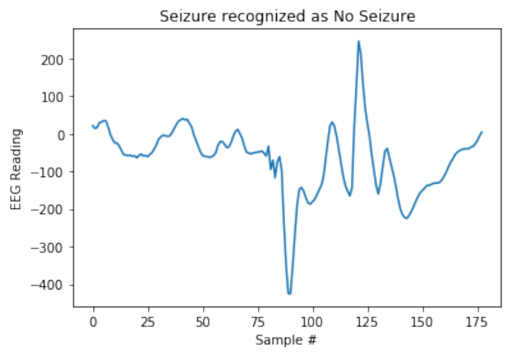
3



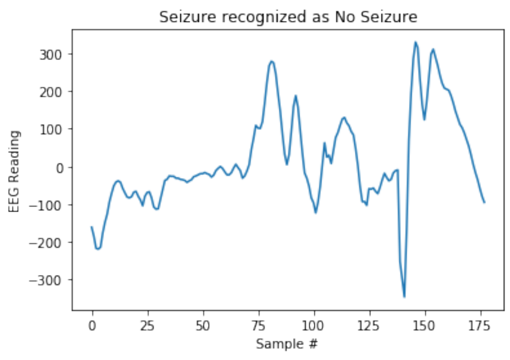
4



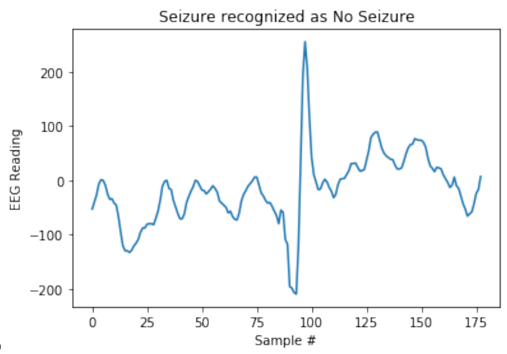
5



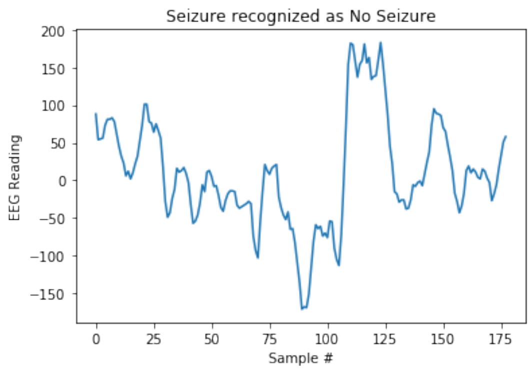
1



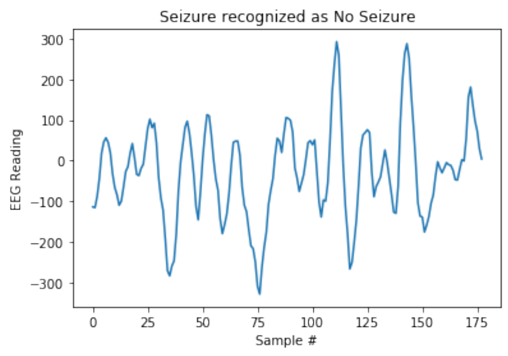
2



3



4



5

Figure 10: Analysis of False Negatives in Test data (Seizure being recognized as No Seizure)

Similarly, Figure 10 shows the plots of 5 subjects that were mislabeled as having no seizure when in fact they were seizure (False Negative) by our Optimized CNN model. In this case, it seems that most of the plots have a fairly high frequency component mixed into the EEG signal, although they may be noisy and a bit irregular. By initial visual analysis of the datasets (see Exploratory Visualization section), it was determined that most non seizing subjects have a fairly high frequency that is fairly regular in oscillation. As a result, it is understandable that the model might mistake models that have a higher frequency of oscillation (though less regular) as an occurrence of a seizure. From both False Positive and False Negative analysis, it seems that the model has the most trouble identifying seizures in EEG data that has an irregular signal signature. This issue may be mitigatable if we feed multiple seconds worth of data to our predict (rather than training and prediction on just 1 second of data).

Reflection

The process used can be summarized as follows:

1. EEG signal data of seizure and non-seizure patients was collected from an open source repository, Kaggle, under the title Epileptic Seizure Recognition (2)
2. Preprocessing of the EEG data to divide into training (validation) and test sets/ extract features for training of traditional classification models (also apply Fourier Transform to EEG data for training of Neural Net)
3. Benchmarks accuracy score and F-Score were created to assess both traditional and convolutional neural network models in their ability to properly classify seizure patients
4. Both the traditional classification techniques and convolutional neural nets were trained on the data sets, the former being fed extracted features from the EEG data, the latter being fed the raw EEG data itself (and Fourier transform data).
5. Optimal parameters were achieved for the best traditional classification method (Random Forest) and the CNN neural net by increasing the model complexity.
6. Performance metrics were recorded and compared between models with the Optimized Convolutional Neural Network performing the best with a test accuracy and F-score of .9884 and .9736 respectively
7. False Positives and False Negatives on the test data were plotted for our optimal models to assess the conditions under which it fails to properly identify seizures.

The most difficult parts were steps 2, 4, 5 because I had to learn how to use multiple libraries including the fftpack library in scipy, the weighted class in quantiles library, and the numerous functions in Sci-kit learn to create/optimize CNNs and traditional classifiers. The most interesting parts were coming up with the different frameworks for these models, establishing the proper set of extracted features for training, and then assessing the performance of each model using preconceived metrics.

Although the models tested in this study were fairly rudimentary, they showed that even simple methods that can be trained with a laptop computer have the ability to identify seizures in EEG with a relatively high degree of accuracy (better than many models written in scientific journals). While the dataset may not be fully extensible to the real world, this at least opens up the possibility for real time seizure detection using wearable EEG sensors and provides a direction for models to be used in such a scenario.

Improvement

This study seems to suggest that more complex models that overfit the training data have a better ability to predict seizures than simpler models. For both Random Forest and our CNNs, we observed that by increasing model complexity, we were able to improve test accuracy and F-Score by a small but significant margin. Given more processing power, I would have liked to train additional layer in my CNN and increased the number of estimators and max depth cutoff parameters in my Random Forest model to see at which point test accuracy and F-score start to decline due to overfit. I think SVM which is used in many of the models in our benchmark table has also shown good results in Seizure Detection, so a next step may be to create an SVM model of our own to see if we can reproduce (or improve) the results.

Appendix

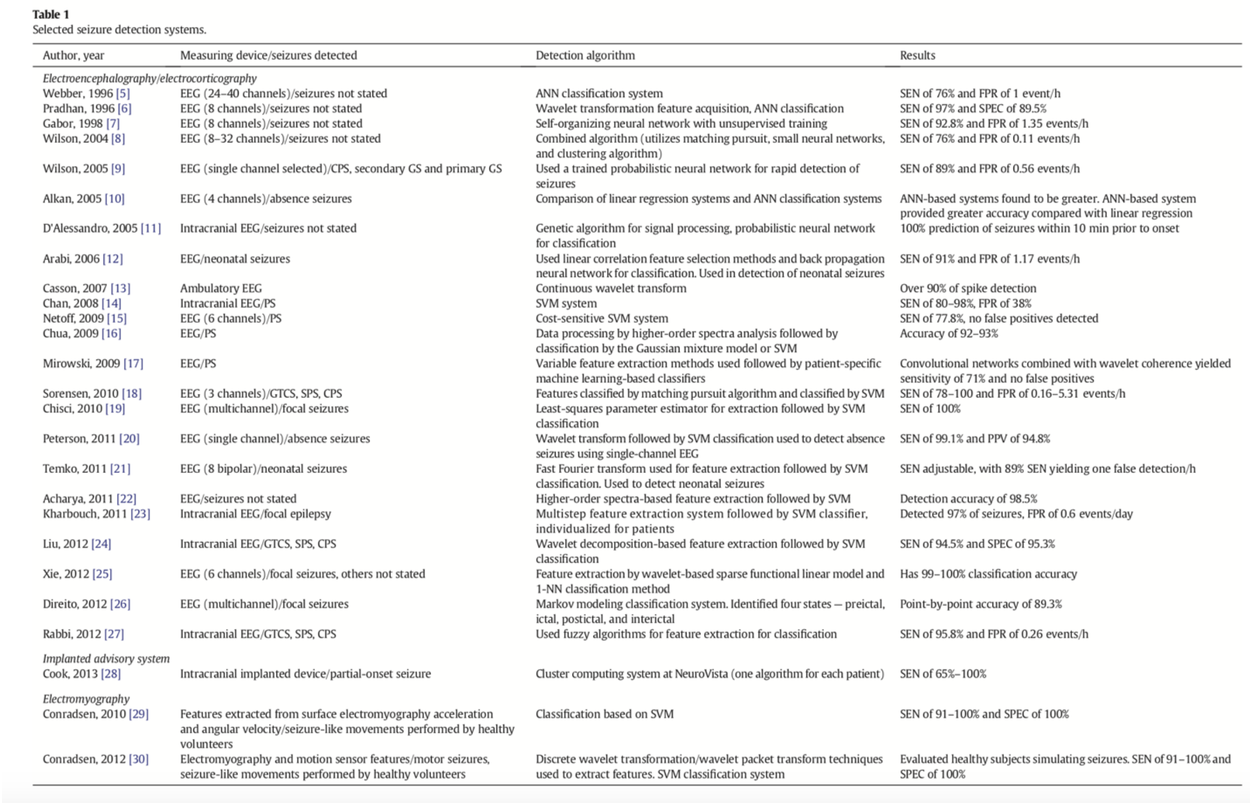


Table 1
Selected seizure detection systems.

Author, year	Measuring device/seizures detected	Detection algorithm	Results
Electroencephalography/electrocorticography			
Webber, 1996 [5]	EEG (24–40 channels)/seizures not stated	ANN classification system	SEN of 76% and FPR of 1 event/h
Pradhan, 1996 [6]	EEG (8 channels)/seizures not stated	Wavelet transformation feature acquisition, ANN classification	SEN of 97% and SPEC of 89.5%
Gabor, 1998 [7]	EEG (8 channels)/seizures not stated	Self-organizing neural network with unsupervised training	SEN of 92.8% and FPR of 1.35 events/h
Wilson, 2004 [8]	EEG (8–32 channels)/seizures not stated	Combined algorithm (utilizes matching pursuit, small neural networks, and clustering algorithm)	SEN of 76% and FPR of 0.11 events/h
Wilson, 2005 [9]	EEG (single channel selected)/CPS, secondary GS and primary GS	Used a trained probabilistic neural network for rapid detection of seizures	SEN of 89% and FPR of 0.56 events/h
Alkan, 2005 [10]	EEG (4 channels)/absence seizures	Comparison of linear regression systems and ANN classification systems	ANN-based systems found to be greater. ANN-based system provided greater accuracy compared with linear regression
D'Alessandro, 2005 [11]	Intracranial EEG/seizures not stated	Genetic algorithm for signal processing, probabilistic neural network for classification	100% prediction of seizures within 10 min prior to onset
Arabi, 2006 [12]	EEG/neonatal seizures	Used linear correlation feature selection methods and back propagation neural network for classification. Used in detection of neonatal seizures	SEN of 91% and FPR of 1.17 events/h
Casson, 2007 [13]	Ambulatory EEG	Continuous wavelet transform	Over 90% of spike detection
Chan, 2008 [14]	Intracranial EEG/PS	SVM system	SEN of 80–98%, FPR of 38%
Netoff, 2009 [15]	EEG (6 channels)/PS	Cost-sensitive SVM system	SEN of 77.8%, no false positives detected
Chua, 2009 [16]	EEG/PS	Data processing by higher-order spectra analysis followed by classification by the Gaussian mixture model or SVM	Accuracy of 92–93%
Mirowski, 2009 [17]	EEG/PS	Variable feature extraction methods used followed by patient-specific machine learning-based classifiers	Convolutional networks combined with wavelet coherence yielded sensitivity of 71% and no false positives
Sorensen, 2010 [18]	EEG (3 channels)/GTCS, SPS, CPS	Features classified by matching pursuit algorithm and classified by SVM	SEN of 78–100 and FPR of 0.16–5.31 events/h
Chisci, 2010 [19]	EEG (multichannel)/focal seizures	Least-squares parameter estimator for extraction followed by SVM classification	SEN of 100%
Peterson, 2011 [20]	EEG (single channel)/absence seizures	Wavelet transform followed by SVM classification used to detect absence seizures using single-channel EEG	SEN of 99.1% and PPV of 94.8%
Temko, 2011 [21]	EEG (8 bipolar)/neonatal seizures	Fast Fourier transform used for feature extraction followed by SVM classification. Used to detect neonatal seizures	SEN adjustable, with 89% SEN yielding one false detection/h
Acharya, 2011 [22]	EEG/seizures not stated	Higher-order spectra-based feature extraction followed by SVM	Detection accuracy of 98.5%
Kharbouch, 2011 [23]	Intracranial EEG/focal epilepsy	Multistep feature extraction system followed by SVM classifier, individualized for patients	Detected 97% of seizures, FPR of 0.6 events/day
Liu, 2012 [24]	Intracranial EEG/GTCS, SPS, CPS	Wavelet decomposition-based feature extraction followed by SVM classification	SEN of 94.5% and SPEC of 95.3%
Xie, 2012 [25]	EEG (6 channels)/focal seizures, others not stated	Feature extraction by wavelet-based sparse functional linear model and 1-NN classification method	Has 99–100% classification accuracy
Direito, 2012 [26]	EEG (multichannel)/focal seizures	Markov modeling classification system. Identified four states — preictal, ictal, postictal, and interictal	Point-by-point accuracy of 89.3%
Rabbi, 2012 [27]	Intracranial EEG/GTCS, SPS, CPS	Used fuzzy algorithms for feature extraction for classification	SEN of 95.8% and FPR of 0.26 events/h
Implanted advisory system			
Cook, 2013 [28]	Intracranial implanted device/partial-onset seizure	Cluster computing system at NeuroVista (one algorithm for each patient)	SEN of 65%–100%
Electromyography			
Conradsen, 2010 [29]	Features extracted from surface electromyography acceleration and angular velocity/seizure-like movements performed by healthy volunteers	Classification based on SVM	SEN of 91–100% and SPEC of 100%
Conradsen, 2012 [30]	Electromyography and motion sensor features/motor seizures, seizure-like movements performed by healthy volunteers	Discrete wavelet transformation/wavelet packet transform techniques used to extract features. SVM classification system	Evaluated healthy subjects simulating seizures. SEN of 91–100% and SPEC of 100%

Table of EEG seizure detection models and associated sensitivity/specificity.
Source: *Seizure detection, seizure prediction, and closed-loop warning systems in epilepsy.* **Sriram Ramgopal, Sigride Thome-Souza, et al.** 2014, Epilepsy & Behavior, pp. 292.

Bibliography

1. *Seizure detection, seizure prediction, and closed-loop warning systems in epilepsy.* **Sriram Ramgopal, Sigride Thome-Souza, et al.** 2014, Epilepsy & Behavior, pp. 291-307 .
2. **Harun-Ur-Rashid.** https://www.kaggle.com/harunshimanto/epileptic-seizure-recognition. [Online] October 11, 2018. [Cited: August 16, 2019.]
3. **Mansar, Youness.** https://towardsdatascience.com/sleep-stage-classification-from-single-channel-eeg-using-convolutional-neural-networks-5c710d92d38e. [Online] October 1, 2018. [Cited: August 15, 2019.]
4. **Rosebrock, Adrian.** Keras Conv2D and Convolutional Layers . *pyimagesearch.com.* [Online] December 31, 2018. [Cited: September 23, 2019.] https://www.pyimagesearch.com/2018/12/31/keras-conv2d-and-convolutional-layers/.