Using EEG Scans to Detect Alcoholism in Patients

Physics 180M Final Project, Winter 2022

Zooey Nguyen	Suyash Kumar	Max Kroft	Nick Millkey
zooeyn@ucla.edu	suyashsep12@gmail.com	revelent0@gmail.com	nmillkey@gmail.com
SECTIONS I, IV	SECTIONS VI, VII	SECTIONS II, III	SECTIONS V, VIII

***Abstract*— In this paper, we present an alternative, objective method for identifying Alcohol Use Disorder in patients by applying machine learning to electroencephalogram (EEG) scans. Currently, the most widely used method is to give the patient the Alcohol Use Disorder Identification Test (AUDIT) survey. Although this achieves accuracies upwards of 90%, it is a subjective method that relies on the patients' answers. Our goal is to classify patients through EEG scans in an objective way. Due to the complexity of the data, simpler machine learning models may lose out on useful information relevant to brain activity. We take advantage of convolutional neural networks' ability to preserve spatial information and automatically identify important features in data in order to reach classification accuracies ($\sim$87%) near that of the AUDIT survey.**

I. Big Picture

Roughly 5.3% of Americans aged 12 and over suffer from an Alcohol Use Disorder (AUD) [1]. The risk of developing AUD is widely theorised to be driven by genetic predisposition in addition to other environmental and psychological risk factors for addiction. While the actual neurophysiology of AUD is not as well understood, exploring its effects on brain activity can be a helpful tool in mapping out its potential characteristic psychophysiological processes as well as evaluating a patient's risk of developing AUD.

Our goal is to create a classifier that can identify whether a patient is an alcoholic or not based on an electroencephalogram (EEG) scan, which involves attaching electrodes to the scalp to detect the electrical impulses that come from the brain [2], while being exposed to one of three possible types of visual stimuli. In a single trial, patients are shown a single picture, two identical pictures, or two different pictures from the Snodgrass and Vanderwart image set. The EEG scan samples the voltages from each of the 64 electrodes placed on the scalp at 256 Hz for one second after the patient is shown the stimulus. Our model takes in the 64x256 data points from a single trial and outputs a classification of "alcoholic" or "non-alcoholic."

A. Previous Work

There is strong evidence implicating differential brain activity in areas of the brain relating to reward pathways [3], decision-making, and emotional responses [4] in altering behavior in alcoholics. Since EEG electrodes are placed all around the brain, the spatial information in the data can indicate if these areas are in fact activated differently in alcoholics in response to a stimulus.

Additionally, studies have shown that there are multiple possible effects of alcoholism on response time, either raising it due to slower information processing or decreasing it due to higher impulsiveness [5]. The high temporal resolution of EEG compared to other brain-scanning techniques such as Magnetic Resonance Imaging (MRI) provides the benefits of looking at reaction time on a much shorter timescale closer to the initial stimulus, rather than using physical or active responses to record differential behavior [6].

The most widespread alcohol dependence screening instrument is the Alcohol Use Disorders Identification Test (AUDIT) [8]. The AUDIT is a short questionnaire that effectively distinguishes alcoholics and non-alcoholics in responses, with its assessments of alcoholism aligning with actual diagnoses of AUD at a rate of 92% [9].

Machine learning techniques have been extensively explored in the problem of automatically identifying alcoholism from brain scans. Ahmadi et al. 2017 [10] applied wavelet transform and PCA to EEG signals to then reduce dimensionality of the data, achieving high accuracies (>90%) with SVM, random forest, and gradient boosting. Mumtaz et al. 2017 [11] similarly applies rank-based feature selection to EEG features corresponding to inter-hemispheric coherences and spectral power in different bands before building the classification model, reaching high accuracy, sensitivity, and specificity (>80% for all). These results, although based on different stimuli, show promising EEG indicators to focus on in feature extraction and that machine learning techniques can effectively distinguish patients.

B. Problem Framework

In the full data set, we have EEG scans from 122 patients over 120 trials for each. The patients are divided into two groups: control and alcoholic. A trial involves the patient being exposed to one of three kinds of stimuli (Figure 1):

1) S1: patient is shown a single image
2) S2 match: patient is shown two identical images
3) S2 non-match: patient is shown two distinct images

Where these images are from the Snodgrass and Vanderwart image set.

It is helpful to visualize each trial as a 2-dimensional color map with time stamps on the x-axis and electrode number on the y-axis (Figure 2). The sampling frequency is 256 hertz over a 1 second period, and the samples are taken over 64 electrodes attached to the scalp of a given patient.

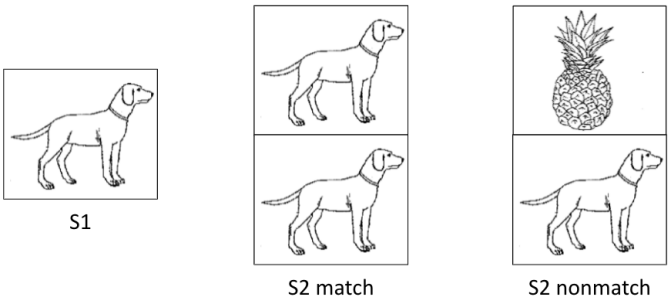


Fig. 1: Examples of three classes of stimuli that the patients were shown in the study.

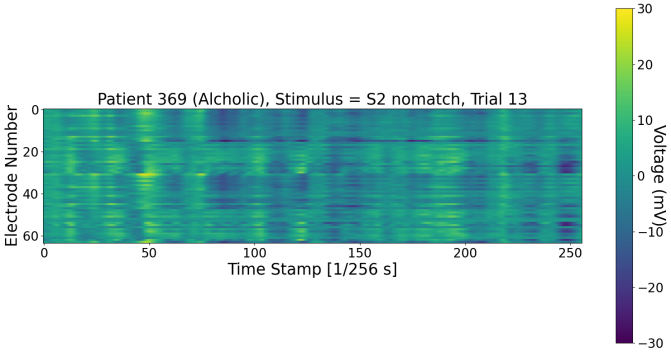


Fig. 2: Example of a 2-dimensional color map representation of the data. Voltage is plotted as color vs Time Stamps and Electrode Numbers.

As for the classification task, the idea is to look at the EEG scan and predict whether the subject of the scan is an alcoholic or not. Since responses to different stimuli can vary significantly, we decided to build separate classifiers for S1, S2 match, and S2 non-match. Therefore, given a patient's EEG scan and knowledge about what stimulus the patient was exposed to, we will feed the scan into the appropriate classifier and the classifier will attempt to decide if the patient suffers from AUD.

II. Data Acquisition

We obtained our data from the the UCI Machine Learning Repository [7]. As stated on the data description page [1], there are four data sets available:

1) smni97_eeg_data (4 MB): This is a very small subset of the full data set, mainly meant for getting acquainted with the structure of the data set. We have 6 folders, 3 for an alcoholic patient and 3 for a control patient. For each kind of patient, there is a folder for each kind of stimulus, and for each stimulus we have 10 trials (description of a file for a given trial will be given after this list).
2) SMNI_CMI_TRAIN (38 MB): This contains 40 folders, 20 for alcoholic patients and 20 for controls. For each patient, we have 30 trials, 10 of each stimulus type.
3) SMNI_CMI_TEST (38 MB): Identical to SMNI_CMI_TRAIN in strucutre, just a different subset of the full data set
4) eeg_full (720 MB): This has folders for 122 patients with 120 trials for each.

The file for a given trial is a compressed text file that has information about the patient name and stimulus type in the header, and thereafter has voltages for each of the 64 electrodes over 256 timestamps. We wrote a Python script that loops over all trial files in a given data set to construct separate summary CSVs/HDF5 files for trials belonging to each of the three stimulus types. The summary CSVs contain a list of patient names, their patient type, and the trial number. The HDF5 files contain a $N \times 64 \times 256$ NumPy array, where N is the number of trials in a given stimulus category within a given data set.

While smni97_eeg_data, SMNI_CMI_TRAIN, and SMNI_CMI_TEST don't have any files containing errors, eeg_full does. To remove these files, we did a search in the headers of all trial files in the full data set and to identify any keywords indicating errors in the files or if the contents of the file are empty.

III. Data Exploration

We began by looking at a small sample of the data provided by the UCI Machine Learning Repository [7] (smni97_eeg_data, or "the small data set" as described in the data set description page. The small data-set contains only two patients, one control and one alcoholic, with thirty trials each. First, we compared the time series for each patient using the same stimulus, looking through one electrode at a time. The initial results were very encouraging, as the alcoholic patient generally had voltage peaks or dips at significantly different time stamps than the control in many of the electrodes. This led us to believe that we could create a single feature out of each electrode, reducing our data set significantly and allowing us to try out simpler machine learning models, such as random forests.

We then began to look at the provided training set (SMNI_CMI_TRAIN as described in the data description page, check footnote). This data set contained 20 patients (10 alcoholic, 10 control), and 30 trials for each patient (10 of each stimulus class).

Considering the fact that our EEG scans exist in 1.6E+4 (64 $\times$ 256) dimensional space, dimensionality reduction seemed to be a natural strategy: what if all the electrodes of an EEG scan (Figure 2) are not equally important? Then, perhaps, we wouldn't have to feed the entire 64 $\times$ 256 image into our classifier and could make do with a subset of it.

First, we wanted to see if there were certain timestamps in each kind of patient which were less effective in identifying alcoholism. Therefore, we create Figure 3. It does seem from the constructed figure that certain timestamps are more

[1] https://archive.ics.uci.edu/ml/datasets/eeg+database

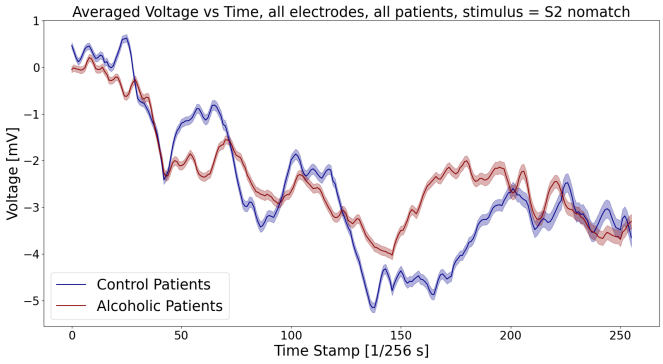


Fig. 3: Signals for alcoholics and control patients, averaged over all patients in the given category, over all respective trials and all electrodes. The shaded bands around the central dark lines represent the error on the mean.

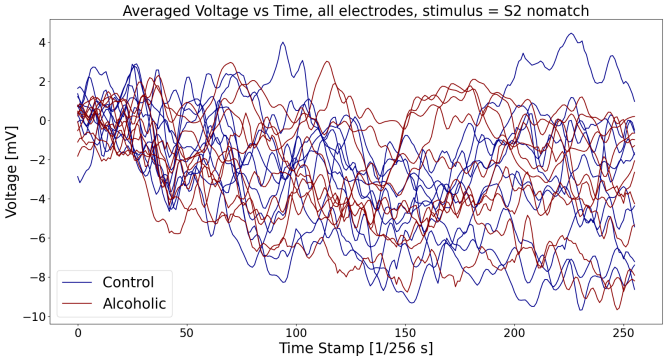


Fig. 4: Signals for all alcoholics and control patients, averaged over over all respective trials and electrodes. This is the same plot as Figure 3, except that the curves are separated out by patient. This shows the high variance among patients.

effective at differentiating between alcoholics and controls. For instance, we can see that between the timestamps 150 and 200, the average voltage from alcoholic patients is significantly higher than that of control patients.

But if we avoid averaging over patients (Figure 4), we notice a high variance across patients in each category. For instance, between the timestamps 150 and 200, while there are alcoholic patients exhibiting higher voltages than control patients, there are also a significant number of alcoholics exhibiting lower signals than controls. Thus, feature engineering we were hoping to perform using Figure 3 might give inaccurate results. However, this problem could potentially be avoided by using enough of these types of features. Unfortunately, finding enough features in the voltage series to make a working model would take hours of manually combing through plots.

Our next attempt was to find trends between trials of the same patient. We looked for patterns between electrodes by plotting the voltages as an image in time stamp vs electrode coordinates. Unfortunately, this yielded little information as well (see Figure 5); even between almost consecutive trials

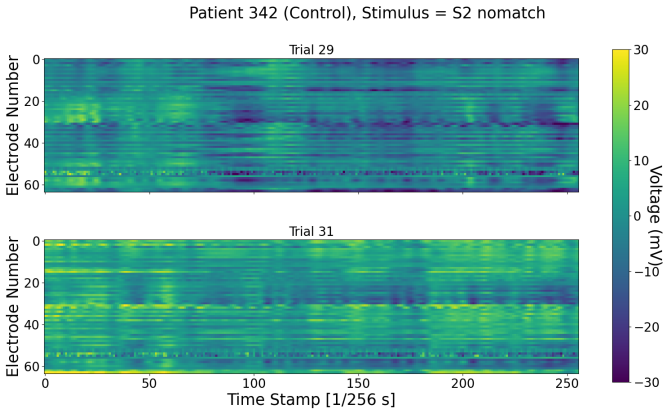


Fig. 5: Voltage maps vs time and electrode number for patient 342. The images show trials 29 and 31, both of which used the S2 non-match stimulus.

on the same patient, with the same stimulus, the images look very different. This method showed even less promise than the time series data, so we moved on once again.

In conclusion, performing a novel dimensionality reduction into useful features manually was turning out to be quite tedious. In the interest of time, we wanted to use a model that would not require manual feature engineering, but instead would pick up on important features (timestamps/electrodes) on its own. To do this, we decided to use a neural network called a Multilayer-Perceptron (MLP) classifier.

IV. Data Preparation

EEG data has been studied extensively over the years, which has resulted in a number of standard pre-processing steps for such data.

There are some frequencies that show up in EEG scans that are not useful when mapping to actual brainwaves, so scientists typically apply thresholding filters and band-pass filters under a Fourier frequency transform [12] to remove them. A high-pass filter usually removes frequencies below 0.1-1 Hz, the range which is affected by the patient's breathing, variation in the skin-electrode impedance, or other baseline interferences. A low-pass filter can vary depending on the range of activity being studied, but typically caps frequencies at around 30-100 Hz to remove activity from muscle movement and blinking artifacts. A notch filter can be applied at around 50-60 Hz to deal with power-line interference from the scanner itself [12]. In modern EEG datasets, this filtering may be applied at the time of collection.

Neuroscientists have also identified five primary regions of brain-wave frequencies that correspond to functionally distinct brain activities. From lowest to highest frequency, they are named delta, theta, alpha, beta, and gamma [13]. A common dimensionality reduction technique on EEG scans employs these frequency bands to decompose the signal into five parts and then calculate the bandpower in each one [14]. This allows for analysis of which areas of function are excited in the brain.

For example, deep sleep usually correlates to a higher relative bandpower in the delta band compared to wakefulness.

Another important method to consider is normalisation of signals across patients. Because baseline activity and activity levels can vary from patient to patient, EEG signals will typically be standardised using techniques such as z-scoring or min-max scaling [15]. This is useful if the timing of high activity and not the absolute level above baseline is necessary for analysis.

Finally, there are numerous techniques for denoising. While EEG scans are taken in well-controlled environments, noise is inevitable beyond just the standard filtering due to the complexity of brain activity. Singular-value decomposition (SVD) is commonly employed in this task to separate noise and activity into different subspaces and can also be used in dimensionality reduction [16]. Another method known as wavelet decomposition separates a signal into wavelets, which are standalone oscillations that are localised in time, like a moving spike [17]. Unlike a Fourier transform that decomposes a signal into global frequency components, which do not capture irregular spikes in signals that are relevant to analysis like in EEG scans, wavelet transforms capture both frequency information in the shape of the wavelet and time information in the scale factor of a wavelet at a certain point. One complication in this technique is finding a suitable mother wavelet set, which defines the shapes of the wavelets to break the signal into and sufficiently remove noise. Another is that in decomposition, there are fewer time evolution coefficients for low-frequency wavelets, so the output of the decomposition has a different dimension for each frequency-band of wavelets [18]. However, if these complications are handled, it is a powerful technique for retaining complex information about the signal.

We tested the impact of transforming the data using frequency transform, average bandpower, scaling, and wavelet decomposition. However, with a large enough dataset, a model that is equipped to "smooth out" noise and baseline variation may be able to pick up on the important underlying features of EEG without having to heavily pre-process the data. Because we wanted to preserve as much spatial and temporal information as possible, and thereby not risk losing information via dimensionality reduction or accidentally overstepping while denoising, our final model actually did not employ these techniques. Our preliminary models built with pre-processed data did not have significantly different results, indicating that our final model in fact was able to distinguish important information for prediction.

V. Model Selection and Training

A. Training on the smaller data set (SMNI_CMI_TRAIN)

1) Multi-Layer Perceptron (MLP) Classifier: Our first attempt at trying to overcome the difficulty of manual feature engineering for this project was with the TensorFlow implementation of an MLP network. MLPs are simple neural networks that consist of layers of perceptrons connected by weights that are fitted from training data via back propagation.

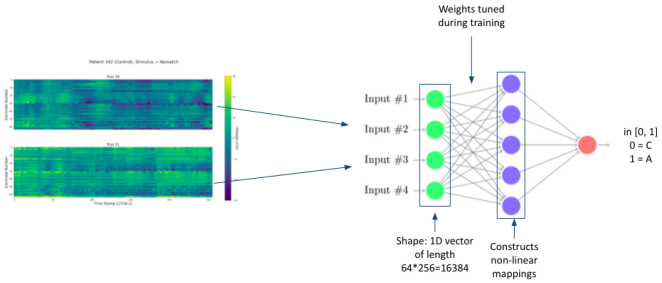


Fig. 6: Schematic diagram of an MLP classifier that is trained to identify whether a given EEG scan belongs to an alcoholic or not. The MLP architecture is taken from [22].

These models are advantageous because they are able to detect trends that can be difficult to find by eye. The inputs for our network were the 64x256 pixel images created from the time-series data from each electrode (Figure 6). We started with a simple MLP in order to obtain a baseline performance to use as a benchmark for all further models and hyperparameter tuning. The initial MLP was constructed using TensorFlow and contained 2 hidden Dense layers (25 and 15 nodes, respectively) with ReLU activation functions and a final Dense layer that uses a sigmoid to perform the binary classification.

As the full data set is quite large, we first used the more manageable SMNI_CMI_TRAIN data set that was provided by the repository. It is a smaller subset of trials from 20 subjects taken from the full data set. The S2 non-match was the stimuli that we predicted would be the most useful of the three, so we trained the MLP using only those trials first. The model had an accuracy score of 67% on the S2 non-match trials in the test set obtained from the repository (see Figure 7). While 67% isn't a good enough accuracy to get useful results, it was encouraging that the model was picking up on some trends in the data. It was also promising that the model appeared to do a better job of correctly identifying alcoholic subjects (72%) than control patients (62%). This indicates that our model is tends towards doing what we hoped to accomplish – identifying patients with AUDs.

The relatively poor performance of the MLP on the SMNI_CMI_TEST set is likely explained by the large variance in the data that was found during the data exploration phase (see Figure 4). With so much variety in the signals for each trial – even for the same patient with the same stimuli – it would be extremely difficult to obtain a subset of the data that would generalize well to the full data set.

The MLP performance could have been improved by altering its structure (adding or taking away additional layers) and by hyperparameter tuning (changing the number of nodes in each layer, introducing regularization, and trying different activation functions), but we decided that it would not be a productive use of time to try this. The main reason for this decision is that in order to use our 256x64 pixel images as input for the model, each 2D image had to be flattened into a 1D array of length $256 \times 64 = 16,384$. In performing this

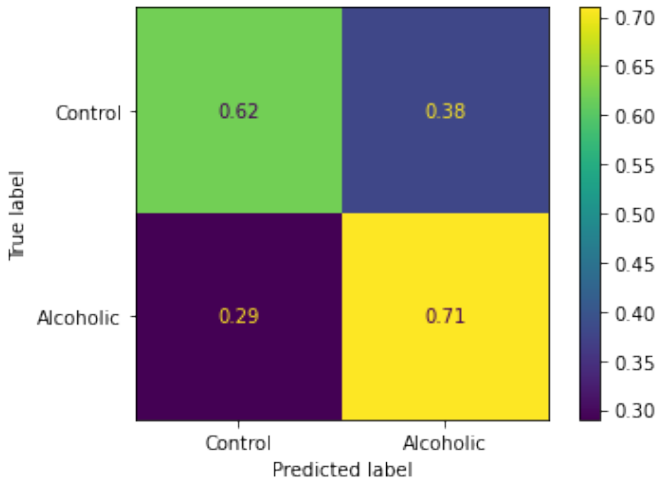


Fig. 7: Confusion Matrix Demonstrating Poor Performance of Initial MLP. The MLP performed with an accuracy of 67% on the S2 non-match trials in the test set (SMNI_CMI_TEST).

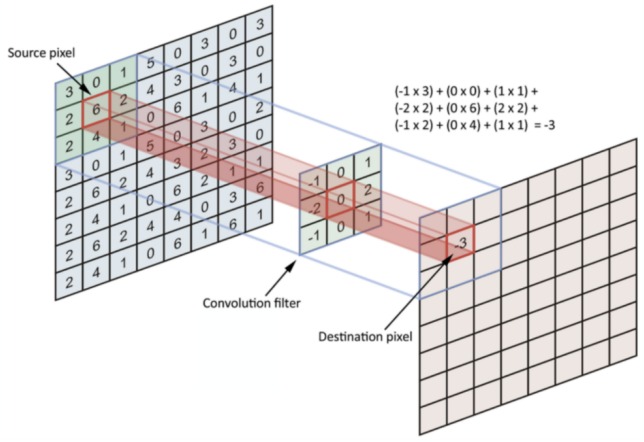


Fig. 8: Visualization of a Kernel Extracting a Feature from an Input to a Convolutional Layer. These layers are key to our model as they preserve the spacial information of our 2-dimensional images. Image from [19].

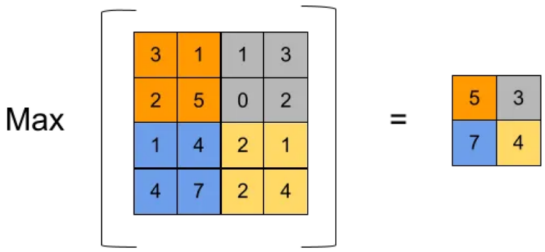


Fig. 9: Visualization of a Max Pooling Layer Performing Dimensionality Reduction by Selecting the Highest Values from each Feature. Image from [20].

flattening, the model loses spacial information from the images (check performance results in 7). In this way, an MLP cannot learn from spacial features in the images, which in the case of our 2D images can contain key relationships between adjacent channels, and therefore will not perform well in classifying each image. We decided instead to shift our attention to a model that can account for the spacial information in the 2D images, convolutional neural networks (CNN).

2) Convolutional Neural Network (CNN): CNNs are similar to MLPs, but the main difference between the two models is that CNNs are able to take in 2-dimensional images as input. To do this, there are (generally) two types of layers that are unique to CNNs:

1) Convolutional Layer: These layers take in the 2 dimensional image and applies different kernels (typically visualized as 3x3 matrices) which act on the entire input image in such a way to pick out certain features from the image. This process preserves crucial spacial information between nearby pixels and has important translational invariance. Different kernels contain different numbers and therefore detect different features. These features are then fed into the next layer. One example of a kernel is an edge-detector (see Figure 8).
2) Pooling Layer: A pooling layer is typically placed after a convolutional layer. It performs dimensionality reduction on the output of the preceding convolutional layer. The most common way this is done is by using MaxPooling, which extracts the maximum value(s) from each feature found by the convolutional layer and outputting these values into the next layer (see Figure 9).

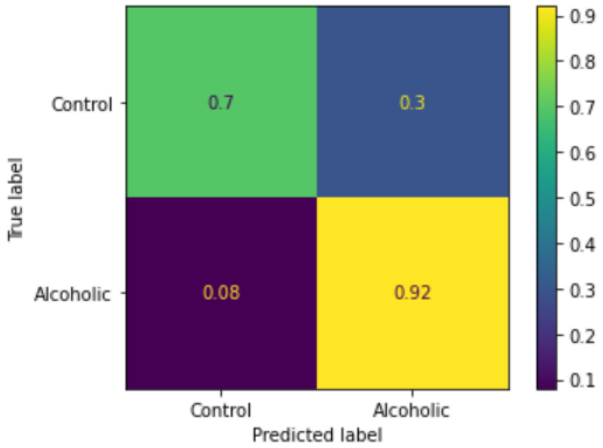
Again, the initial model investigation was done using the SMNI_CMI_TRAIN set. After various hyperparameter tuning, the initial CNN trained on the SMNI_CMI_TRAIN set was determined. The architecture of this model can be seen in Figure 10. The model consists of several sequences of layers in the following order: convolution layer, leakyReLU (selected for its better performance compared to the tanh and sigmoid activation functions), and a max pooling layer. Finally, a dense layer was again used to get binary output. The inclusion of dropout layers and regularization was tested to combat overfitting. However, these do not appear in the final CNN trained with SMNI_CMI_TRAIN because they did not appear to improve performance. Note that the number of nodes in each layer was not tuned at this point. This model was then tested on both the SMNI_CMI_TEST set and part of the full data set (eeg_full), using an 80-20 train-test split, and the results can be visualized in Figure 11.

The model performed very well on the SMNI_CMI_TEST set, achieving an accuracy of 81%. It also correctly identified 92% of alcoholic subjects. This was a much better result than the MLP, as 81% is solidly in the range of accuracy achieved by some AUDIT surveys.

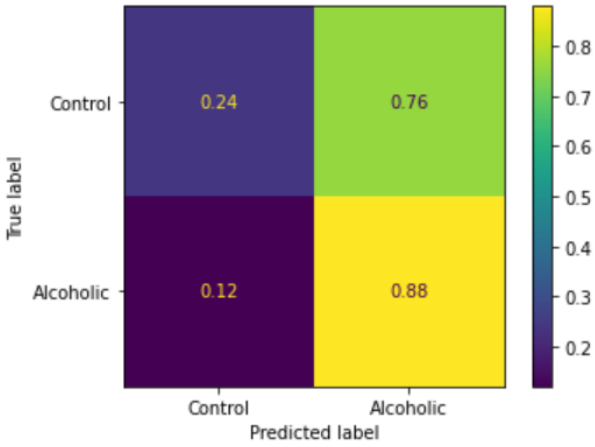
Encouraged by these results, we also tested the predictions of this model on the full data set. The model achieved an accuracy score of 62% (Figure 11). This disappointing result confirmed that models trained on the SMNI_CMI_TRAIN set do not generalize well to the full data set. However, it did

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 64, 256, 32)       320

 leaky_re_lu (LeakyReLU)     (None, 64, 256, 32)       0

 max_pooling2d (MaxPooling2D  (None, 32, 128, 32)      0
 )

 conv2d_1 (Conv2D)           (None, 32, 128, 64)       18496

 leaky_re_lu_1 (LeakyReLU)   (None, 32, 128, 64)       0

 max_pooling2d_1 (MaxPooling  (None, 16, 64, 64)       0
 2D)

 conv2d_2 (Conv2D)           (None, 16, 64, 128)       73856

 leaky_re_lu_2 (LeakyReLU)   (None, 16, 64, 128)       0

 max_pooling2d_2 (MaxPooling  (None, 8, 32, 128)       0
 2D)

 flatten (Flatten)           (None, 32768)             0

 dense (Dense)               (None, 128)               4194432

 leaky_re_lu_3 (LeakyReLU)   (None, 128)               0

 dense_1 (Dense)             (None, 2)                 258

=================================================================
Total params: 4,287,362
Trainable params: 4,287,362
Non-trainable params: 0
_________________________________________________________________
```

Fig. 10: Architecture of the Initial CNN Trained Using SMNI_CMI_TRAIN



(a) Performance of CNN trained with SMNI_CMI_TRAIN on SMNI_CMI_TEST. Results show a promising accuracy of 81%, correctly identifying 92% of alcoholic subjects.



(b) Performance of CNN trained using SMNI_CMI_TRAIN on the full data set. Results show an accuracy score of 62%. This is significantly worse than the model did on the SMNI_CMI_TEST, which confirms that the model does not generalize well to the full data set, and that future models and parameter tuning need to be done with the full data.

Fig. 11: Confusion matrices for predictions of the CNN trained on on SMNI_CMI_TRAIN on various unseen sets (SMNI_CMI_TEST and the test split from eeg_full)

generalized better than the MLP, achieving a score over 50%, demonstrating that it was still learning something from the data.

B. Using a CNN on full data set

Considering the fact that the CNN performed very well on the smaller data set but was unable to generalize to the full data set, it was worth going forward and training our chosen CNN architecture over part of the full data set (eeg_full), using an 80-20 train-test split. When the same model architecture (Figure 10) was trained using the full data set, the accuracy of the model tested on the full data set improved to 72% (Figure 12), although it did perform worse at identifying alcoholic subjects with only 58% accuracy.

The combination of the high accuracy of the model when tested on the SMNI_CMI_TEST set and the ability of the model to generalize – at least to some extent – to the full data set, convinced us that CNNs were the optimal model to pursue. It was also clear that we must do all further hyperparameter tuning and architecture selection on the full data set, as training with the SMNI_CMI_TRAIN set was not producing models that could accurately predict the full data.

VI. Fine Tuning the Model

Training the CNN on the full data set did improve accuracy, however there are several things we could do to better attune the model to the large data. Below we discuss several strategies/components for fine tuning the CNN (note that all the architecture choices choices except the number of nodes in the convolutional/dense layer are common to all three classifiers).

Firstly, considering the large size of the full data set we decided to go with a minimal architecture in terms of layers. The hyperparameters for each layer (except number of nodes) were determined by exhaustive trial and error. Below is a description of each layer and the hyperparameter values (except the number of nodes), see Figure 13 for a visualization of the model.

1) A convolutional input layer that accepts 64 $\times$ 256 images using the default kernel with a size of 5 $\times$ 20

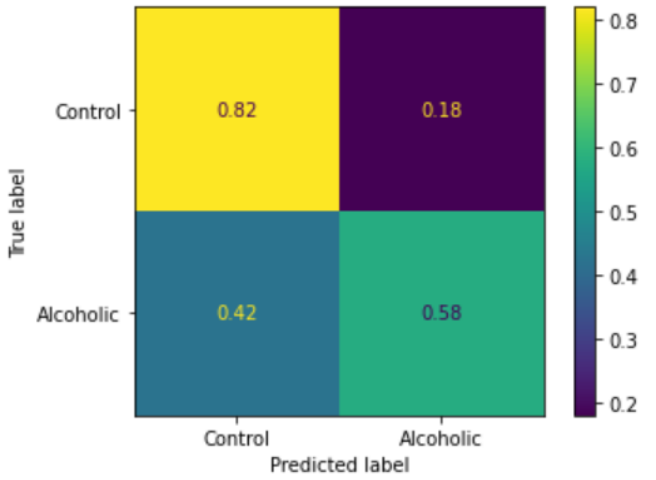


Fig. 12: Performance of the initial CNN when trained and tested using an 80-20 split of the full data set. The model achieved an accuracy score of 72%.

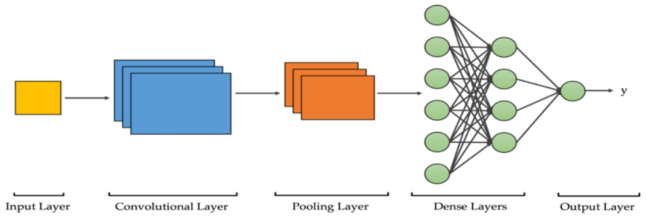


Fig. 13: Visualization of Final CNN. The structure was left simple to conserve computational time and consists of a convolutional layer, a max-pooling layer, a dense layer, and an output perceptron. Image from [21].

(keeping in mind the 1:4 aspect ratio of input images). We use the ReLu activation and also introduce fixed regularizations (kernel, bias, and activity) of 10^{-5}.

2) A max-pooling layer of size 3×12.
3) A dense layer using ReLu activation with fixed regularizations (kernel, bias, and activity) of 10^{-5}.
4) A single perceptron (sigmoid activated) providing a probabilistic output between 0 (control) and 1 (alcoholic)

As for the number of nodes in the convolutional layer and dense layer, we determined an ideal number for each classifier by trial and error. Thereafter, we let the number of nodes vary in a range covering ± 5 centered at the ideal number of nodes in the convolutional and dense layer that we determined by trial and error. This creates a grid of 100 models for each classifier, which we cover using grid search empowered by keras_tuner. The final model summaries are shown in 14.

In terms of the model compilation, we make the following choices.

1) Optimizer: We decided to use stochastic gradient descent because it scales well with big data sets. We fix a learning rate of 10^{-4} (also determined through trial and error). Note that the regularizations of the network layers and the optimizer learning rate are chosen in a way to avoid overfitting to the training data while also ensuring that we converge to a solution fast enough.
2) Loss and metric: We compile the model using binary cross entropy as our loss function and binary accuracy as our metric considering the fact that we are dealing with a binary classification problem.

Finally, we train our network for 400 epochs (courtesy to GPU quota on Google Colaboratory[2]) using a batch size

[2]https://colab.research.google.com/

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 60, 237, 23)       2323

 max_pooling2d (MaxPooling2D  (None, 20, 19, 23)       0
 )

 flatten (Flatten)           (None, 8740)              0

 dense (Dense)               (None, 19)                166079

 dense_1 (Dense)             (None, 1)                 20

=================================================================
Total params: 168,422
Trainable params: 168,422
Non-trainable params: 0
_________________________________________________________________
```

(a) Model summary for the S1 classifier

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 60, 237, 21)       2121

 max_pooling2d (MaxPooling2D  (None, 20, 19, 21)       0
 )

 flatten (Flatten)           (None, 7980)              0

 dense (Dense)               (None, 13)                103753

 dense_1 (Dense)             (None, 1)                 14

=================================================================
Total params: 105,888
Trainable params: 105,888
Non-trainable params: 0
_________________________________________________________________
```

(b) Model summary for the S2 match classifier

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 60, 237, 29)       2929

 max_pooling2d (MaxPooling2D  (None, 20, 19, 29)       0
 )

 flatten (Flatten)           (None, 11020)             0

 dense (Dense)               (None, 13)                143273

 dense_1 (Dense)             (None, 1)                 14

=================================================================
Total params: 146,216
Trainable params: 146,216
Non-trainable params: 0
_________________________________________________________________
```

(c) Model summary for the S2 non-match classifier.

Fig. 14: Model summaries

of 10 samples and a validation split of size .2. The model losses as a function of the number of epochs are presented in Figure 15. The model accuracies as a function of epochs is presented in Figure 16. Note that for the purposes of accuracy determination, an implicit assumption about the classification threshold being .5 is made; i.e. probabilities below .5 are reduced to 0 while those above are raised to 1.

VII. Final Model

Recall that to compute evaluation metrics (accuracy, precision, recall, F_1 scores, area under ROC curve etc.) we first need to make a prediction. What our neural networks provide us are a probability of a given patient being alcoholic. To actually make a prediction, we must choose a threshold $c \in [0, 1]$ such that if the network's probability $p < c$, then we predict the patient to be from the control group (0), otherwise we predict them to be an alcoholic (1).

Therefore, while the model accuracies in Figure 16 were evaluated using a threshold score of .5, we can try to choose an alternate threshold which accuracy. We present in Figure 17 the prediction accuracies for different thresholds. Using these thresholds, we construct confusion matrices for each classifier, presented in Figure 18.

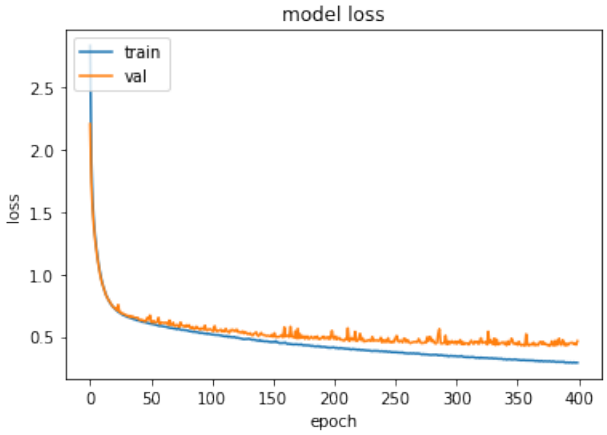
There are a few observations that can be made from the confusion matrices.

1) The accuracy of classifying alcoholics is greater than the overall accuracy for each classifier. Since the main purpose of this classification task is identifying alcoholism, our classifiers are doing their job well!
2) A portion of control patients get classified as alcoholics, but a majority of controls are still classified correctly, and the situation is not as bad as in Figure 11b.
3) The S1 classifier has a somewhat higher overall accuracy than the classifiers for S2 match and S2 non-match. This can be explained by S1 having roughly twice the number of samples in the full data set compared to both S2 match/non-match. Since a CNN's performance depends heavily on the volume of data provided to it, it makes sense that S1 performs slightly better than S2 match/ non-match. In fact, S2 match/non-match perform almost as well as S1 with half the data, so one can still argue that S2 non-match is the best discriminator among alcoholics and controls.

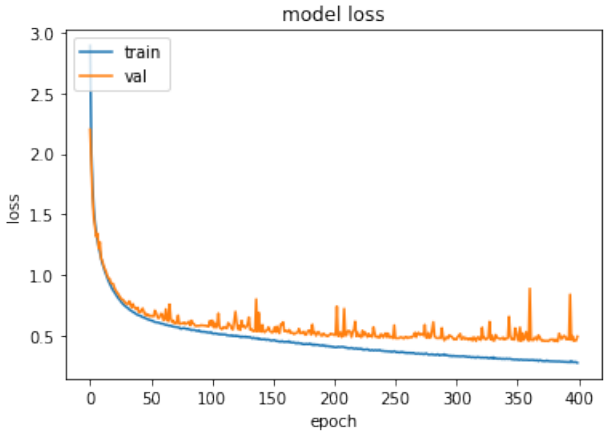
We can also evaluate some other metrics for our maximum-accuracy predictions. First, we would like to look at precision (P), recall (R), and F_1 scores. Here are the definitions for all these metrics.

1) $P = \frac{T_p}{T_p+F_p}$, where T_p and F_p are respectively true and false positives.
2) $R = \frac{T_p}{T_p+F_n}$, where F_n are false negatives.
3) $F_1 = \frac{2PR}{P+R} = \frac{T_p}{T_p+.5(F_p+F_n)}$

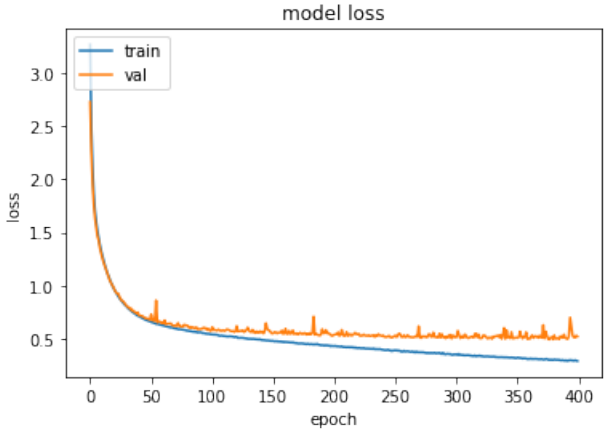
Precision is intuitively a model's ability to avoid labelling positive samples as negative, while recall is the model's ability to find all positive samples. The F_1 score captures the trade-off between precision and recall, since a high F_1 score can only



(a) Model losses as a function of epochs for the S1 classifier

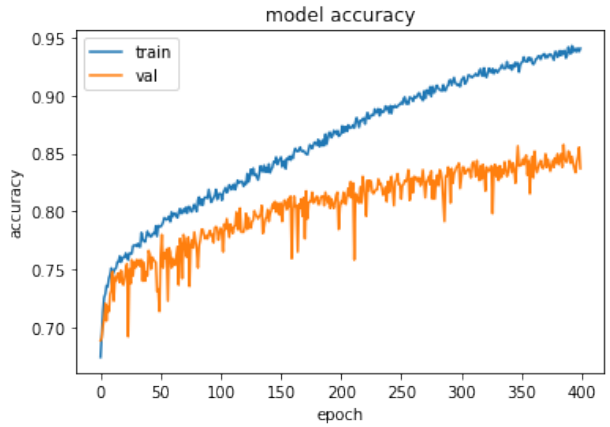


(b) Model losses as a function of epochs for the S2 match classifier

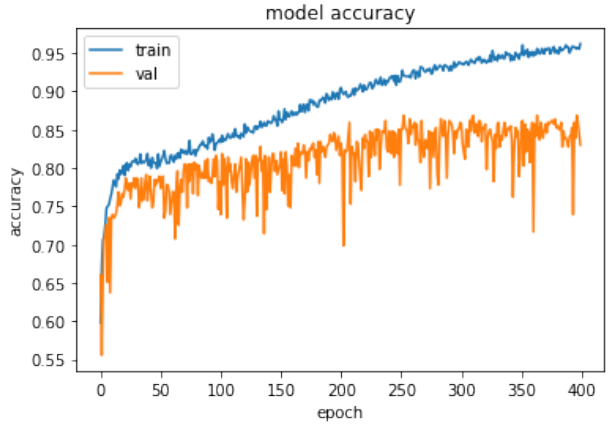


(c) Model losses as a function of epochs for the S2 non-match classifier.

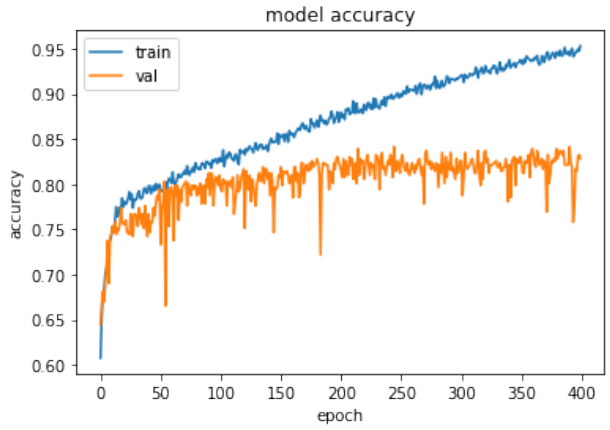
Fig. 15: Model losses as a function of epochs



(a) Model accuracies as a function of epochs for the S1 classifier

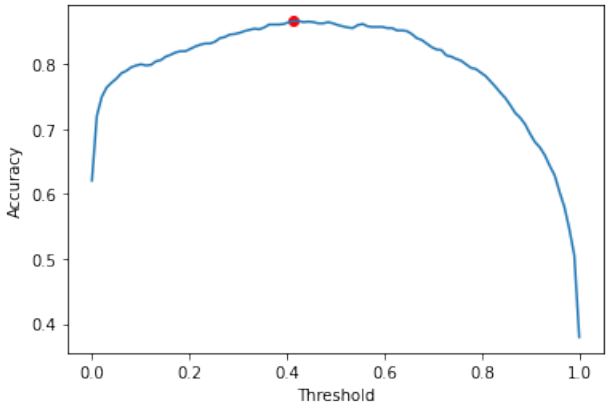


(b) Model accuracies as a function of epochs for the S2 match classifier

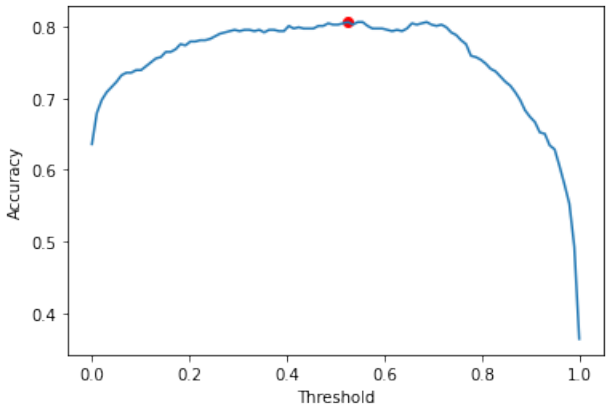


(c) Model accuracies as a function of epochs for the S2 non-match classifier.

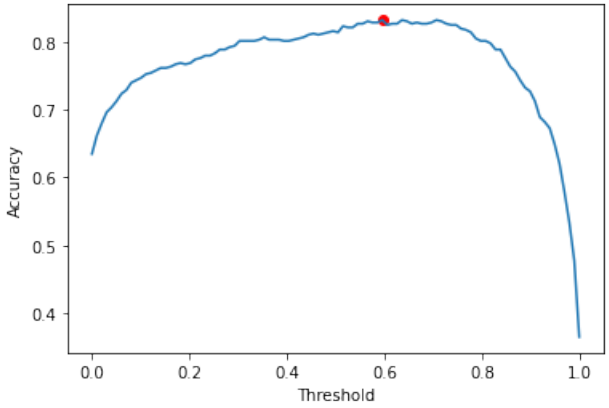
Fig. 16: Model accuracies as a function of epochs



(a) Model accuracies as a function of threshold for the S1 classifier

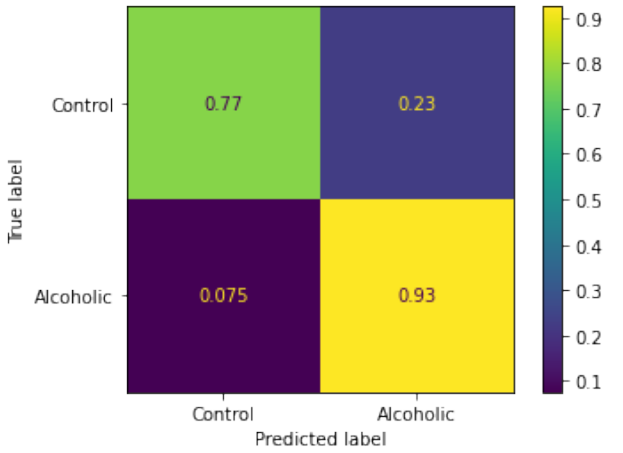


(b) Model accuracies as a function of threshold for the S2 match classifier

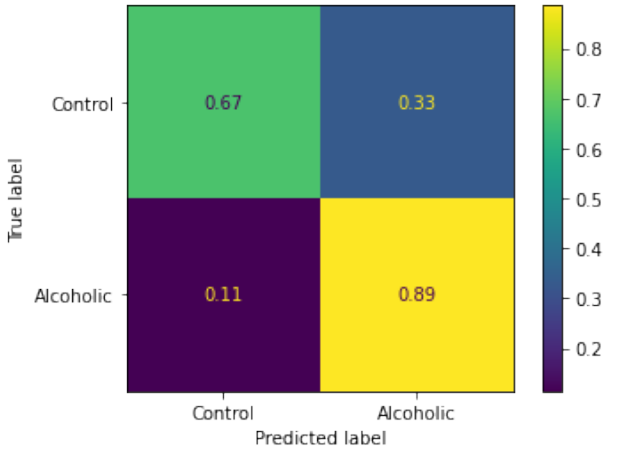


(c) Model accuracies as a function of threshold for the S2 non-match classifier

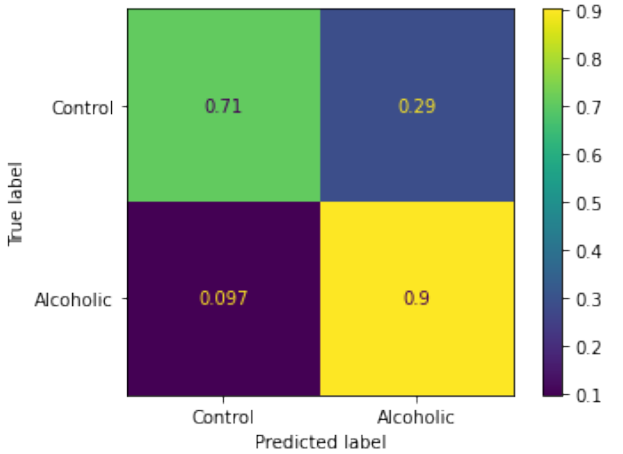
Fig. 17: Model accuracies as a function of threshold. The scatter points present the threshold for which accuracies are maximized



(a) Confusion matrix for the S1 classifier. The classifier achieves an accuracy of 86.59% using a threshold of 0.414.



(b) Confusion matrix for the S2 match classifier. The classifier achieves an accuracy of 80.62% using a threshold of 0.525.



(c) Confusion matrix for the S2 non-match classifier. The classifier achieves an accuracy of 83.12% using a threshold of 0.596.

Fig. 18: Confusion matrices for all three classifiers.

be achieved by having both high P and R. Thus, a model should be considered to be a good classifier only if it has a fairly high F_1 score. We can see in Figure 19 that all classifiers have high P, R, and therefore F_1 scores at the threshold of maximum accuracy, so our classifiers are well trained.

VIII. Conclusions

Overall, we were very successful in accomplishing our goal of applying machine learning to EEG time series data to create a more objective test for alcoholism. Our final CNNs achieved accuracy scores of approximately 87%, 81%, and 84% when trained on the S1, S2 match, and S2 non-match stimuli, respectively. 87% puts our model on the same order of accuracy as the AUDIT exam while avoiding the subjectivity of a survey. It is interesting to note that the model trained using the S1 stimuli achieved the highest accuracy despite our prediction that the S2 non-match stimuli would be the most effective. This can be explained by the fact that there were twice as many S1 trials than S2match or S2 non-match trials. CNNs rely heavily on the amount of data they have to train on, so it makes sense that S1 would get the highest accuracy.

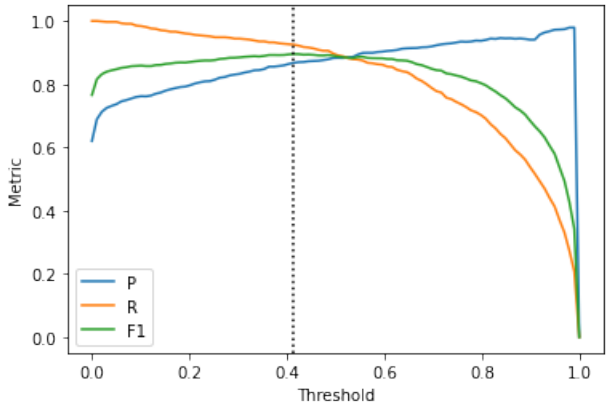
By using the previously mentioned statistic that 5.3% of Americans suffer from an AUD, we can now calculate the true positive rate of our model – the likelihood that a patient that is classified as alcoholic by our model is actually an alcoholic. Using Bayes' Theorem, A represents an alcoholic patient while M indicates that the model identified the patient as an alcoholic:

$$P(A|M) = \frac{P(M|A) \times P(A)}{P(M)} = \frac{(0.93) \times (0.053)}{(0.93) \times (0.053) + (0.23) \times (1 - 0.053)} = \frac{(0.93) \times (0.053)}{(0.2671)} = 0.1845 \quad (1)$$

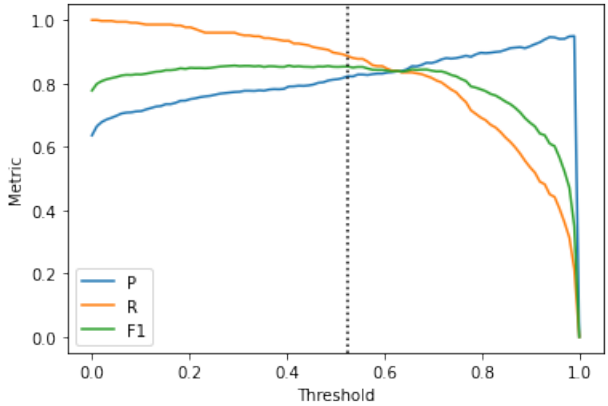
This means that only 18% of the times that a patient is identified as an alcoholic by our model will the patient actually turn out to have an AUD. This number seems alarmingly low, however it is explained by the fact that the prior is very low (5.3%) – only a small (but significant) portion of the US population suffers from an AUD. This concept keeps the true positive rate low for many medical tests, and is not an indicator that the model is not performing well.

There are several things we can try out going forward to potentially improve the performance of our three classifiers -

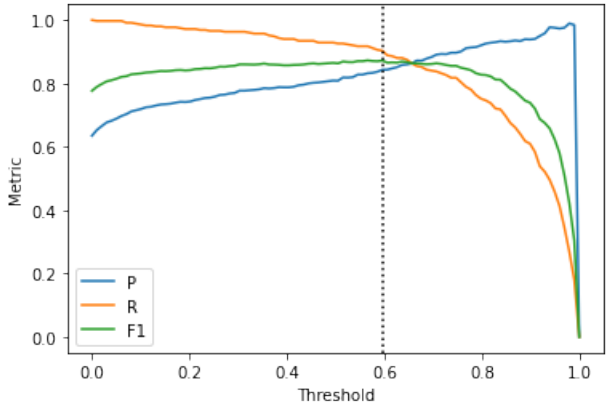
1) We should continue to understand how CNNs work and come up with better network architectures guided by how various hyperparameters affect CNN performance, and not mere trial and error.
2) We should look at feature maps produced by the CNNs to see if they can help in dimensionality reduction. We may also consider using PCA/ICA/SVD/wavelet transforms for denoising our data to see if it helps the network pick up features more easily.



(a) Precision, recall, and F1 scores for the S1 classifier as a function of classification threshold.



(b) Precision, recall, and F1 scores for the S2 match classifier as a function of classification threshold.



(c) Precision, recall, and F1 scores for the S2 non-match classifier as a function of classification threshold.

Fig. 19: Precision, recall, and F1 scores for all three classifiers. The vertical black dotted line is the threshold for which the classifier accuracy is maximum.

3) By treating this problem as an image classification task, we ignore the importance of time in the problem: at the end, we have a set of signals, and causality matters. CNNs are not able to account for this. We could use Recurrent Neural Networks (RNNs), possibly combining them with CNNs to predict alcoholism.

In the end, we created a model that was able to reliably distinguish alcoholic subjects from control subjects; the eventual goal is to be able to detect signs of predisposition to alcoholism in order to help people before they develop an AUD. More research needs to be done before this can be obtained, but our model is a step in the right direction for achieving this goal.

Acknowledgment

We thank everybody in the group for their contributions to the project. We also thank you our TA Evan Jones and Professor Tuan Do for guiding us on this project and throughout the quarter. We also acknowledge the resources provided by UCLA IDRE and Google Colaboratory that allowed us to carry out data-intensive investigations to construct our best possible machine learning model.

References

[1] https://www.niaaa.nih.gov/publications/brochures-and-fact-sheets/alcohol-facts-and-statistics
[2] https://www.hopkinsmedicine.org/health/treatment-tests-and-therapies/electroencephalogram-eeg
[3] https://www.ncbi.nlm.nih.gov/pmc/articles/PMC1236699/
[4] https://www.nature.com/articles/1300295
[5] https://www.karger.com/Article/Abstract/72881
[6] https://www.frontiersin.org/articles/10.3389/fneur.2019.00848/full
[7] https://archive.ics.uci.edu/ml/datasets/eeg+database
[8] https://auditscreen.org/
[9] https://onlinelibrary.wiley.com/doi/10.1111/j.1360-0443.1993.tb02093.x
[10] https://ieeexplore.ieee.org/document/8104182
[11] https://www.ncbi.nlm.nih.gov/pmc/articles/PMC5350086/
[12] https://iopscience.iop.org/book/978-0-7503-3279-8/chapter/bk978-0-7503-3279-8ch6
[13] https://www.nature.com/articles/s41598-017-15966-6
[14] https://raphaelvallat.com/bandpower.html
[15] http://www.saujs.sakarya.edu.tr/tr/download/article-file/833143
[16] http://eceweb1.rutgers.edu/ laleh/files/EMBC_2015.pdf
[17] https://towardsdatascience.com/the-wavelet-transform-e9cfa85d7b34
[18] https://www.medscimonit.com/abstract/index/idArt/4828
[19] https://towardsdatascience.com/simple-introduction-to-convolutional-neural-networks-cdf8d3077bac
[20] https://androidkt.com/explain-pooling-layers-max-pooling-average-pooling-global-average-pooling-and-global-max-pooling/
[21] https://medium.com/swlh/an-overview-on-convolutional-neural-networks-ea48e76fb186
[22] https://miro.medium.com/max/1000/1*BQ0SxdqC9Pl_3ZQtd3e45A.png