Machine Learning Approach for EEG-based Person Identification

Team: Harsha Santhanam, Aditya Rathi, Sandhya Bellary. **Project Mentor TA:** Ricardo

1) Abstract

Biometric scanners, such as face or finger-print recognition, have become increasingly less secure. With these technologies struggling to maintain the needed amount of security people desire, new biometric technologies are needed. EEG recognition is much harder to fake, which is why it could be used for secure biometric identification. Our primary target contribution is to develop image classification models to accurately classify EEG signals. Second, we would like to evaluate the efficacy of our model by testing it on a dataset of 105 individuals. We successfully built a logistic regression model to perform classification at an accuracy of 78%. Additionally, we were able to create a Feed Forward Network to classify at an even higher accuracy of 94%. More interestingly, we find that CNNs struggle to perform this classification with an accuracy of only 71%.

2) Introduction

Conventional biometric identification techniques include fingerprint detection, face recognition, and speech recognition. However, with each of these methods it is rather easy to circumvent the protection mechanism. For example, with speech recognition someone could easily record another person’s voice. Even with facial recognition, computer scientists have been able to make deep fake technologies that are able to successfully trick face recognition sensors. With these mechanisms failing to maintain security, we must look for new identification methods.

EEGs are brain wave signals collected using electrodes placed on an individual’s head. These signals have been used to perform person identification; however, not as much work has been done on this compared to other recognition techniques. With EEG signals, it’ll be substantially harder for someone to replicate as these signals are highly variable depending on the individual. With this in mind, our work seeks to develop a Machine Learning algorithm to accurately identify individuals based on their EEG signals.

We use a dataset containing 105 individuals performing a variety of tasks. Each signal is associated with a specific individual. The input to our model will eventually be an individual's EEG signals, and the output will be the person’s identity.

To predict the person’s identity based on an EEG signal, we will be treating this as an image classification problem. EEG’s have characteristic portions which can hopefully be used to train a classification model. With this in mind, we seek to identify these portions and use them to identify a person.

3) Background

EEG based recognition is relatively new. Researchers have created classification algorithms on this subject; however, we see very few people using it for actual security measures. Typical models used to perform this classification include LDAs and RNNs. RNN’s struggle to remember long-term contexts, which can be troublesome especially when you are analyzing EEGs (which are essentially time based). In addition, baseline work with logistic regression has been performed relatively effectively. For this reason, we wanted to build off of the logistic regression classification approach with more powerful models like a Feed Forward Network as well as a Convolutional Neural Network. By using architectures like CNNs, we also look at this classification problem through a new lens: image classification. We hope that by interpreting this question in a new and unique way, we will be able to improve on the accuracy that other researchers have achieved by using more conventional techniques.

1. Isuru Jayarathne, Michael Cohen, Senaka Amarakeerthi et al, “BrainID: Development of an EEG-based biometric authentication system”. What the authors do in this paper is use LDA to develop their model and it is relevant to our research because we know that new techniques, like utilizing transformers, can produce better results, and hence we think it would be interesting for us to compare our results with this model.
2. Zhang et. al, “DeepKey: A Multimodal Biometric Authentication System via Deep Decoding Gaits and Brainwaves”. In this paper, the authors compare the accuracies of SVMs, LDAs, LSTMs, CNN, Att-RNNs, which are 0.77, 0.31, 0.85, 0.76, and 0.93 respectively (on the test sets). We hope to perform better than these architectures.

4) Summary of Our Contributions

1. Code - Rather than using conventional methods of classification such as RNNs, we decided to build off of prior work and utilize FFNs as well as CNNs to perform the classification. Then we compared our accuracies to that of an RNN.
2. Application - Applying typical image classification models to a new set of data (EEG signals)
3. Data - N/A
4. Algorithm - Defined a new algorithm to perform EEG classification. We use image classification models to perform the task.
5. Analysis - N/A

5) Detailed Description of Contributions

We first implemented a Logistic Regression model as it is much easier to implement compared to the RNN model that is traditionally used for EEG signal identification models. However, there are several drawbacks to a Logistic Regression model. Firstly, there is an assumption of linearity between the independent and dependent variable which is not necessarily the case with our data. Due to this Logistic Regression models are not able to identify complex relationships within the data and assume the data is linearly separable which we cannot say is the case for our data. These limitations led us to believe it would have a low accuracy in comparison to our other model; however, we found that it actually yielded accuracies that, while lower, were comparable to those of the RNN models in the research.

We propose, however, that using CNN and FNN models will be more accurate in their predictions. Firstly for FNN models, the main difference between this model and RNN models is the presence or lack thereof of a feedback loop. For RNN models, this feedback loop can be a cause for concern when dealing with lengthy sequences of data. Thus, we believe FNN models are better at dealing with large sequences which would be useful for time-series data. FNN models are more flexible in this way. Feed Forward Networks can be prone to overfitting; however, our model proved to be both more effective on both validation accuracy and testing accuracy compared to the attention based RNN provided in DeepKey [1].

FNN Architecture:

Fully Connected Layer, Activation Layer, Fully Connected Layer, Activation Layer, Fully Connected Layer, Activation Layer, Fully Connected Layer

For FNN and CNN models, we are able to analyze EEG data as an image. However, for CNN models specifically, we will be able to extract the important features rather than collecting extensive memory of information. In addition, CNN models look at data from a broad outlook; whereas, RNN models are focused on a step-by-step process that is heavily dependent on past history/data. Specifically for EEG data, CNN models look at data in batches which allows it to look at the data in the form of neighborhoods. This is important since EEG data may rely on this reasoning rather than simply a compilation of information from the past timesteps. One potential issue with this pooling method, however, is the reduction of a portion of data to a singular item rather than a part of the whole dataset. Without putting a batch in the context of the entire image, the CNN model may incorrectly categorize an image on the basis of certain features being present at high volumes rather than understanding the exact location of these features and how they fit together to form an image. In addition, CNN models require large amounts of training data which raises concerns for overfitting of the data. However, this issue can be fixed by adding dropout layers.

CNN Architecture:

Sequential(Convolutional Layer, Activation Layer, Max Pooling Layer
(x6)Sequential(Convolutional Layer, Activation Layer)
Dropout(0.5), Fully Connected Layer, Fully Connected Layer, Fully Connected Layer

The CNN model took a long time to fully train due to the increase of complexity and shift from layer to layer. Thus, this increased our training/evaluation time as well due to constant iterations in not only layer parameters but also the permutations of layers we chose to use. In addition, we encountered issues with the structure of the data. By restructuring the data to the form of 96 x 96 images rather than time series data, we ran into some formatting issues but were able to clean the data to fit an image classification model. While training the model, we found that through editing the parameters within the layers, we were still not able to reach a high accuracy, so we decided to vary the learning rate to achieve better results.

For a baseline, we used the research study cited and compared accuracies of their models to the accuracies of our models. However, we also found that our dataset was consistently used in several other research papers and compared against their model results as well. This facilitated our ability to measure the impact of our algorithm and gave us the information necessary to understand where our contribution could actively change the EEG biometric scanning industry. Due to the disruptive nature of CNNs in machine learning academia, we felt that it would contribute significantly to this industry as its success rate in general has shown to be promising. In addition, it allows us to think about EEG data in a new way: that of image processing rather than time-series. Its capabilities far exceed RNN models due to its ability to process long sequences of EEG signals likely yielding better results for identification.

5.1 Methods

The following steps were taken to complete this project:

1. Download data files and add them to a shared drive. Then, connect it to a colab document by mounting it. Find the path of these files, and save it.
2. Extract each of the .edf files, convert to .csv, and then preprocess:
    a. Remove rows that are all 0s.

    b. Normalize (L2 Norm)
    c. Do fast fourier transform to remove noise from the EEG data
3. Reshape each channel into 96x96 “images”
4. Create a train-test split to have ~80% train and ~20% test.
5. Create the train_loader, test_loader from DataLoader
6. Declare loss criterion as CrossEntropyLoss
7. Create LogisticRegression from torch.nn.Module (just a single linear layer)
8. Create a train_model and test_model function that will give accuracy and loss and plot this to visualize the model training and test.
9. Create Feed-Forward NN (multiple linear layers with Relu)
10. Create a train_model and test_model function that will give accuracy and loss and plot this to visualize the model training and test.
11. Create CNN (multiple linear layers with Convolution, Relu and maxPool and then some FC linear layers as well as optionally dropout)
12. Create a train_model and test_model function that will give accuracy and loss and plot this to visualize the model training and test.

5.2 Experiments and Results

The key question our experiments were trying to answer was to observe other approaches of identifying individuals based on their EEG signal, that is different to the BrainAI paper that uses RNN for classification. Hence, we decided to use LR, FFNN and CNN to observe how well they would do in this use case.

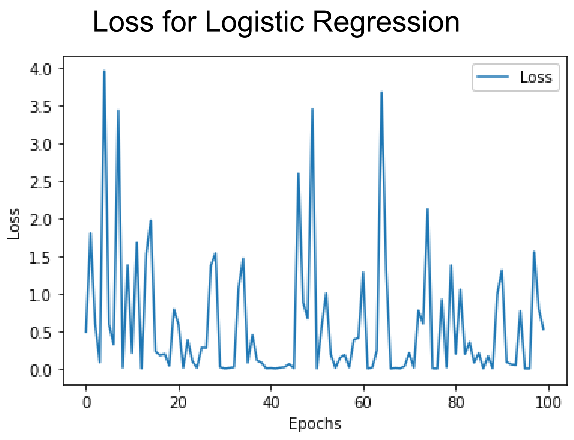
Seeing as CNN does a good job in identifying attributes of images, we had to make some design choices for our data. This included doing some denoising, but more importantly, reshaping single time-series data sets into “images” of 96x96. We decided on square “images” so it is easier to do the required mathematics when we are making decisions about the layers in the CNN. The reason I put “images” in quotation marks is because our data isn’t really images but formatting it this way may allow the CNN to better identify those peaks and valleys in the time-series data that makes solving this problem possible. Algorithmically, there isn’t much to be changed in single or multiple linear layers for LR and FFNN but for the CNN, we tried to keep our kernel size and stride fairly low (1-3) to ensure we weren’t losing too much information. We also tried to reduce the number of max-pools we do to once again prevent loss of information, but also reducing the dimensionality enough so the NN doesn’t overload the free GPU we are given by colab.

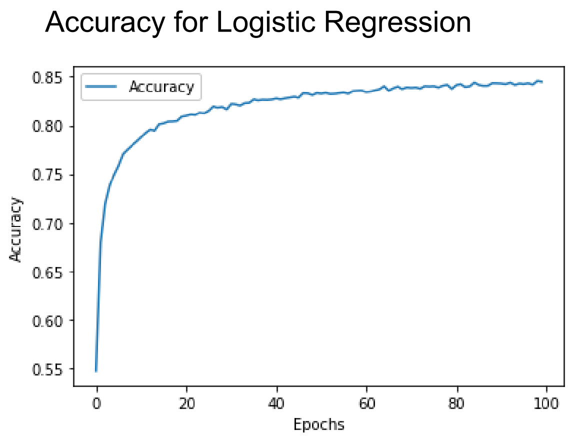
Our baseline approach was to compare each of our models to the 92% achieved by the attention based RNN from DeepKey [1]. We expected Logistic Regression and Feed Forward Network to be close but slightly worse, but CNN to be similar or better than RNN. However, we observed a substantially lower CNN accuracy than expected. We have several guesses for this, from google colab RAM being filled up, too few epochs, too few layers, etc. However, due to time constraints, we weren’t able to fully flesh out an improved CNN!

We tested our models on a dataset that had EEG information about 105 individuals, over 14 trials, with 64 electrodes per trial. The appropriate performance metric for us would be just accuracy (Acc = $\frac{TP + TN}{TP+TN+FP+FN}$) since this is a multi-class classification problem so we mainly just want to evaluate how many classifications are being done correctly.

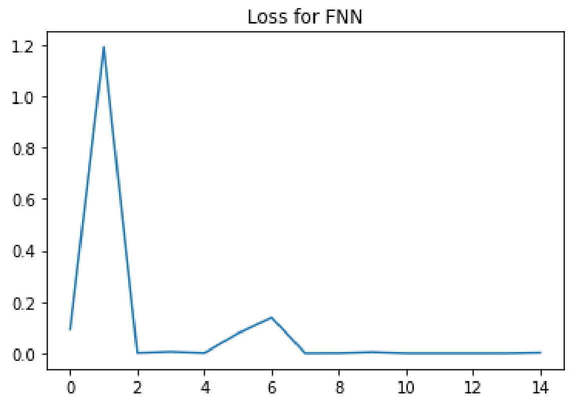
Below, we display the results for the three networks used. Presenting information for both loss and accuracy.

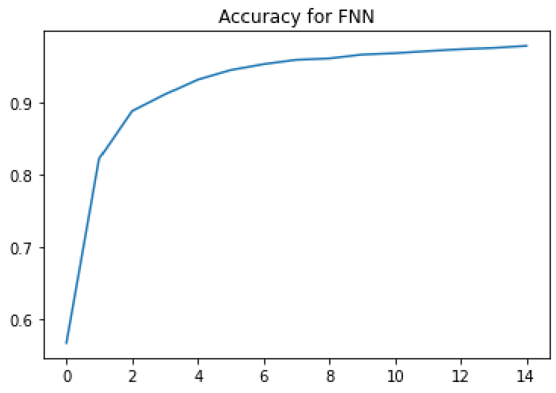
Logistic Regression Model



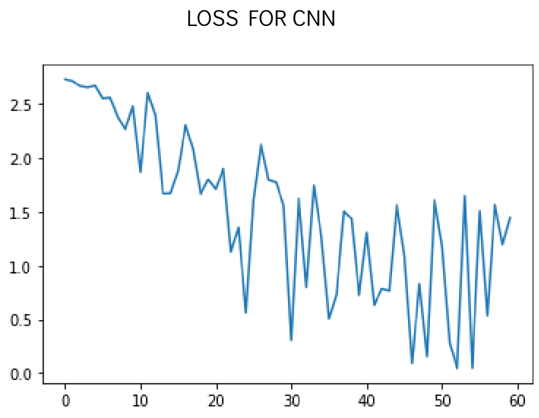


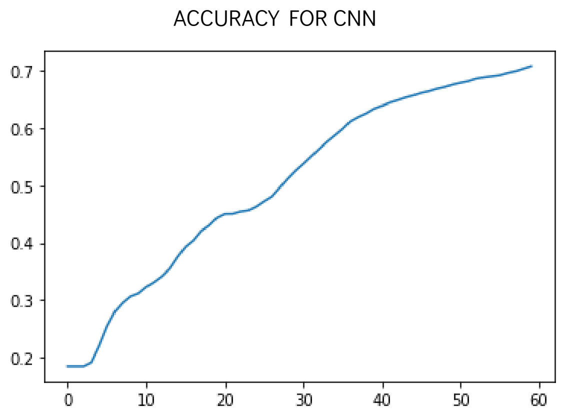
Feed Forward Network





Convolutional Neural Network





	Logistic Regression	Feed-Forward Network	Convolutional Neural Network
Training Loss	*0.527*	*0.002*	*1.44*
Training Accuracy	*0.844*	*0.978*	*0.72*
Testing Loss	*0.108*	*4.77e-07*	*1.72*
Testing Accuracy	*0.821*	*0.943*	*0.71*

6) Compute/Other Resources Used

We had to use additional software such as PyCharm to assimilate all the data appropriately. Everytime we tried to load data into google colab, we would fill up the RAM and not have enough space to even start running the model. Hence, we decided to take pre-processing out of the pipeline into PyCharm. We did all the preprocessing in there and loaded the train and train data sets onto data loaders. Then, we save this into .pkl files and we can then easily extract this using torch.load() and use the data in the models.

7) Conclusions

The main outcome of this project was our model successfully being able to predict individuals based on single electrode values! We were able to do this with variable amounts of success, whether with LR (84% train accuracy, 82% test accuracy), FNN (97% train accuracy, 93% test accuracy) or CNN (72% train accuracy, 71% test accuracy). We definitely believe that the project has produced models that would be very useful. If there are ways to detect brain waves without the need to place electrodes on the subject, we can easily replace other forms of biometric identification like facial scan and fingerprints. Research has shown certain facial recognition algorithms are prone to biases if they aren't trained particularly well on certain ethnicities and while that is certainly something to still be explored with these models, we assume brain waves to be less biased (especially our model since we had no informations about the ethnicities of the 105 test subjects)

In hindsight, we learned a lot from the different feedback we got from our TA, Ricardo. One of the most integral feedbacks we received was in terms of data cleaning. As someone who has already worked with EEG data before, he was quick to tell us that we need to do some sort of filtering. Heeding that advice, we found the appropriate low-pass and high-pass filter frequencies and performed FFT on our data. Another roadblock that we encountered was a low CNN accuracy at first, and after discussing this with Ricardo, we figured out that it might be our implementation of the "images". After running through a few variations of reformatting the train and test data, we settled one the current one which is described in the methods.

In the future, we would definitely like more research in this field. Testing and training our models further on additional participants, doing more tasks than the 14 described in the study we took this data from and individuals from different backgrounds, ethnicities and nationalities to filter out bias.

Our project's primary concern, as described above, would be to adopt the biases of other biometric systems. By being race-blind in our current model training, and in future model training ensuring our dataset is diverse, we can try and ensure our models do not reflect any personal biases, as well as any biases of future model editors.

(Exempted from page limit) Other Prior Work / References (apart from Sec 3) that are cited in the text:

1. Zhang, Yao, Huang, “DeepKey: A Multimodal Biometric Authentication System via Deep Decoding Gaits and Brainwaves”
2. Isuru Jayarathne, Michael Cohen, Senaka Amarakeerthi et al, “BrainID: Development of an EEG-based biometric authentication system”.

Broader Dissemination Information:

Your report title and the list of team members will be published on the class website. Would you also like your pdf report to be published?
No

If your answer to the above question is yes, are there any other links to github / youtube / blog post / project website that you would like to publish alongside the report? If so, list them here.
N/A

(Exempted from page limit) **Work Report: This may look like your GANTT chart from the midway report, with more completed steps now. Okay to modify.** (Mark completed steps in green, as shown here. For convenience, you may split into two charts, one till Nov 8, and another for after Nov 8, placed one below the other.)

		Wk5				Wk6				Wk7				Wk8				Wk9			
		NOV-DEC																	DEC		
PERSON (S)	**TASK (S)**	S 3	M 4	W 6	Th 7	S 10	M 11	W 13	Th 14	S 17	M 18	W 20	Th 21	S 24	M 25	W 27	Th 28	S 31	M 1	W 3	Th 4
Aditya, Harsha	CNN on Small Dataset		🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩							
Aditya, Harsha, Sandhya	Convert 105 Dataset		🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩										
Aditya, Harsha	Run Log Reg on 105									🟩	🟩	🟩	🟩								
Aditya, Harsha, Sandhya	Run FNN on 105									🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩			
Harsha, Aditya	Optimize CNN on 105								🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩
Sandhya	Contribution Write-up								🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩
Aditya	Method Write-up								🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩
...	Task 8																				
...	Task 9																				
...	Task 10																				

(Exempted from page limit) Attach your midway report here, as a series of screenshots from Gradescope, starting with a screenshot of your main evaluation tab, and then screenshots of each page, including pdf comments. This is similar to how you were required to attach screenshots of the proposal in your midway report.

Personal Identification through EEG Signals with Machine Learning

Team: Harsha Santhanam, Aditya Rathi, Sandhya Bellary. **Project Mentor TA: Ricardo**

1.1) Introduction:

We propose to predict the unique identification of a person based on a wide array of EEG signals such as eye-blinks/movements, working memory, motor imagery, etc. Therefore, the inputs to our system will be the EEG signal, and the output will be the person's unique identifier. The data we have chosen to train and test on is a Motor Movement/Imagery Dataset, which has 109 volunteers, 64 electrodes, two baseline tasks (eye-open, eye-close), motor movement, and motor imagery. To use this data, we had to do significant preprocessing to get our data in appropriate features and labels. We will then divide up our data set as 60% training, 20% validation, and 20% testing. After defining our test set, we will treat this as an image classification problem. This means we will break down the EEG signals (composed of 64 channels) into arrays. Next, we will use CNNs as our "image" classification model.

For our model, identifying a person correctly is important (in terms of how well the model is doing), and hence performance metrics like F1 score, as well as average accuracy for added confirmation. We will also use MSE to observe our errors and try to minimize them. We would also deem it successful if it gets close, or surpasses the accuracy of the papers we are drawing inspiration from.

1.2) Motivation:

Biometric scanners/sensors have been implemented in many aspects of our lives. For example, to unlock your phone, there is a camera that analyzes your facial features to identify if you are the owner of your phone. Many of these technologies have proven to be vulnerable; for example, contact lenses can trick iris recognition.

EEG based systems will circumvent many of these issues. EEGs can measure an individual brain's response to stimuli. These signals will be extremely hard to fake as they are unique to an individual's brain. Thus, EEGs could replace many of the current bio-recognition systems as it is not exposed to the same issues.

2) How We Have Addressed Feedback From the Proposal Evaluations

The first point mentioned in our feedback was that we should be careful about the sources of our data. To resolve this issue, we decided to take all of our data from one dataset, that way all of the testing remains uniform.

The next point mentioned was about where our data is coming from. The dataset we ended up using did not mention any preprocessing. However, as referenced by other literature (including the paper that gave us this idea), we can attempt using an L2 normalization of the dataset.

The last point mentioned in our feedback was about an "mvp" (as using Transformers could be very difficult). To address this, we have decided to stay away from an Attention based model (for now) and proceed with CNNs. If we are able to get our CNN model working effectively, we will attempt to use Transformers to perform the classification.

3) Prior Work We are Closely Building From

While several papers have been written on using EEG data for biometric authentication, the following was the most interesting and the one we will be building our project off:

1. Isuru Jayarathne, Michael Cohen, Senaka Amarakeerthi et al, "BrainID: Development of an EEG-based biometric authentication system". What the authors do in this paper is use

LDA to develop their model and it is relevant to our research because we know that new techniques, like utilizing transformers, can produce better results, and hence we think it would be interesting for us to compare our results with this model.

4) What We are Contributing

The most important distinction between our work and prior research is the type of model we will be using. The paper mentioned in prior work uses LDA, or linear discriminant analysis, to perform classification. In addition, other researchers have used RNNs which struggle to remember prior information (which is why LSTMs and Transformers were built). To address the downfalls of both of these models, we decided to treat this as an image classification problem using CNNs.

5) Detailed Description of Each Proposed Contribution, Progress Towards It, and Any Difficulties Encountered So Far

We are proposing two algorithm contributions: a CNN model and the usage of Transformers(if time permits). The first is much easier to implement, and thus, our first contribution. Currently, we are in the process of developing the CNN model and comparing it to Logistic Regression and an FFNN model, of which we hypothesize that the CNN model will do the best. As mentioned before, while researching, we found that most of these algorithms include RNN models or LDA models rather than CNN models. However, if we model the EEG signals as a matrix compiling an image, we can utilize a CNN model that has more feature compatibility than the RNN model.

Pseudocode:

```
class CNN(nn.Module):
def _init_(self):
super().__init__()
self.conv = nn.Conv2d(in_channels=TBD, out_channels=(TBD), kernel_size=TBD, stride=TBD)
self.relu = nn.ReLU()
self.mp = nn.MaxPool2d(kernel_size=TBD)
self.flatten = nn.Flatten(start_dim=1)
self.fc = nn.Linear(in_features=TBD0, out_features=TBD)
def forward(self, x):
outputs = self.mp(self.relu(self.conv(x)))
outputs = self.fc(self.flatten(outputs))
return outputs
cnn_model = CNN()
```

Pseudocode for Statistics:

```
cnn_optimizer = torch.optim.Adam(cnn_model.parameters(), lr=TBD)
cnn_training_loss, cnn_training_accuracy = train_cnn(cnn_model, cnn_optimizer, criterion,
cnn_epochs)
```

We are still adjusting the parameters for the CNN model, but once completed, if we still have the time, we will be pursuing another algorithm contribution: a model with Transformers. For the Transformer model, we predict that it will be more accurate than the CNN model as we believe its attention mechanism will improve this model's accuracy in predicting the identity of the person. This Transformer model will be utilizing sequence to sequence methodology to convert the sequence of the EEG signal to a sequence of the person's identity. It will be interesting to see whether modeling the data as an image versus a sequence will prove to yield more accurate models. By using the attention mechanism, we can analyze at every step what parts of the EEG signal are most important to the current position in the sequence. Among the many benefits we see from the Transformer model the most important are its ability to understand the relationships between parts of a sequence that are far away and dissect the most important parts of a sequence which reduces processing time. That way our model will be both accurate and fast. While we are still understanding the implementation of the Transformer model, we are hoping to have the time to compare its accuracy to our CNN model.

5.1 Methods

Our eventual goal is to train a Machine Learning model to identify a person based on their EEG signal. To accomplish this, we must first find a suitable database that uses the same experimental techniques throughout their data collection. After parsing through many open datasets, we selected a database containing samples from 109 individuals, each with over 64 channels of EEG data. This will provide a more than sufficient amount of data to train our model with.

The next step is to build a profile of our data meaning we must understand what the data is showing and how we can make use of it. The format of the data is presented as a edf file type. This is rather inconvenient when it comes to building data structures (i.e. numpy arrays), so to resolve this issue we converted the file into a .csv filetype. This allows us to create numpy arrays or data frames depending on the type of processing we choose to do. The data collected on a single individual (for one experiment) was divided up into 64 columns and had thousands of rows. Each column corresponded to a different channel, while the rows represented the value of the EEG signal at a certain point in time. In order to standardize the columns, we performed L2 normalization. In addition, some of the rows contained all zeroes, which could imply the experimentation had stopped, so we removed all of the rows containing only zeros. This was the last step in our data processing.

An approach not conventionally discussed with respect to EEGs is treating it like an image classification problem. We can essentially use the information generated from all 64 channels to build a large matrix, which is very similar to turning an image into a matrix of RGB values. With this in mind, we can use conventional image classification techniques like CNNs to build a network that can classify EEGs. Our reason for selecting this approach is that it avoids issues RNNs face. RNNs struggle to remember previous information (which is why LSTMs were later developed). CNNs are able to extract important features from images (in our case an array of EEG signals), and we can hopefully improve on prior RNN work with this new approach (*Zhang et. al*).

5.2 Experiments and Results

The primary question our experiments are trying to answer is can we organize the dataset in a form that would be usable by a convolutional neural network. Since CNNs usually are used for image classification, is it possible for us to get some N x N matrices that can be used for something similar. Our hypothesis is that we should be able to get the data into our desired format and the model should accept the data without many errors. The datasets are basically all our EEG data (109 x 14 x 10000 x 64 - described in detail below) and performance metric would be how accurately the model can predict the labels for each given set of 64 x 64 EEG data. We have loaded our data in data loaders and created the models (LR, FFNN and CNN) as well as the train and test model functions that will do that same.

Since our project has an enormous amount of data (109 participants, 14 trials each, ~10,000 EEG recordings each on 64 nodes), the biggest challenge was definitely cleaning this data and getting our train and test datasets ready.

Hence, we first began by sharing a google collab with the team, connected our google drive and began uploading all the .edf files from our dataset onto the drive. We also switched our runtime to GPU for faster training later on in the assignment. Now that we can access these files in our python notebook, we created the following ETL pipeline:

a. Obtain list of all files in google drive folder, this will be the identifying number for each individual (so 109 people)
b. We can then enter these folders, and obtain only the .edf files
c. Then, for each of the 14 trials of .edf files in there, we can extract it into a .csv file and read the data as a dataframe.
d. We do some cleaning here, removing rows that are all 0s and we normalize the data.
e. Then each of these data frames are divided into equal 64 x 64 smaller frames
f. ~15% of these smaller frames for each person are added to the testing dataset as tuples that are tensors that represent the data, as well as its respective person label. The rest ~85% is the training dataset
g. We use Dataloader to now represent these as train_dataloader and test_dataloader

Once we finished our ETL pipeline, we began constructing our model. We will be testing how our data performs using torch.nn.Module for 3 models - LogisticRegression, Feed-Forward Neural Network and finally Convolutional Neural Network which we hypothesize to do the best. While we are still working out the hyperparameters here, we have started constructing the training loop for our model. This would involve flattening our data, sending it to GPU, sending labels to GPU, training the model, optimizing (zero_grad()), using CrossEntropyLoss as loss criterion, loss.backward(), optimizer.step() and printing our loss and accuracy throughout each epoch to track how the model is doing.

6) Risk Mitigation Plan

One risk we have noticed early on is that the model implemented through Transformers may be difficult to implement (*Vaswani et. al*). In order to mitigate this, we have already begun working on the three other models, specifically the CNN model and will be optimizing these models first in order to have a useful project to turn in. While we have scheduled out ample time to finish the Transformer model and expect to have the CNN model complete within a week, we are acknowledging the possible difficulties by ensuring our CNN model produces an accurate prediction of the person's identity based on the features we input. This will provide us with an understanding of which model serves the best for EEG data: Logistic Regression, FFNN, or CNN. We already suspect that the CNN model will be more accurate than the RNN models that are currently being used in this field. In addition, we noticed early on that gathering the data from different sources may lead to us needing to standardize the data and ensure that the metrics are the same. Combining these datasets would prove to be a difficult task and may lead to us building an inaccurate model given that we did not preprocess the data correctly. To mitigate this, we are using a smaller dataset from one source in order to get the most accurate results. This has simplified our preprocessing and allowed us to focus on optimizing our model. The model can then be generalized to other datasets through preprocessing of those datasets(We will not be attempting this but rather it is an application of our project).

(Exempted from page limit) Other Prior Work / References (apart from Sec 3) that are cited in the text:

1. Vaswani, Ashish et al. “Attention is All you Need.” *ArXiv* abs/1706.03762 (2017): n. Pag.
2. Zhang, X., Yao, L., Kanhere, S. S., Liu, Y., Gu, T., & Chen, K. (2018). Mindid. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, *2*(3), 1–23. https://doi.org/10.1145/3264959

(Exempted from page limit) **Full Work Plan, including the previous work plan with completed/incomplete steps (okay to modify from the proposal), and the remaining steps:** (create additional columns with deadlines for steps towards the final report, assigning responsibilities to individual team members to the extent possible. The GANTT chart you used in the proposal will be a good starting point. Mark completed steps in green, as shown here. For convenience, you can split into two charts, one till Nov 8, and another for after Nov 8, placed one below the other.)

PERSON (S)	TASK (S)	Wk5				Wk6				Wk7				Wk8				Wk9			
		OCT-NOV8																	NOV		
		S 3	M 4	W 6	Th 7	S 10	M 11	W 13	Th 14	S 17	M 18	W 20	Th 21	S 24	M 25	W 27	Th 28	S 31	M 1	W 3	Th 4
All	Preprocessing	🟩	🟩	🟩	🟩																
Aditya	Log Reg			🟩	🟩	🟩	🟩	🟦													
Harsha, Sandhya	FFNN			🟩	🟩	🟩	🟩	🟦													
All	CNN					🟩	🟩	🟦	🟦	🟦	🟦	🟦									
Sandhya	Accuracy Stats								🟩	🟩	🟦	🟦									
All	Hyperparameters								🟩	🟩	🟦	🟦	🟦	🟦	🟦	🟦	🟦				
All	Transformer																🟦	🟦	🟦	🟦	🟦
Aditya	Accuracy Stats															🟦	🟦	🟦	🟦	🟦	🟦
...	Task 9																				
...	Task 10																				

(Exempted from page limit) Supplementary Materials if any (but not guaranteed to be considered during evaluation):