In [ ]:
```
from google.colab import drive
drive.mount('/content/drive')
from IPython.display import Image, display
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, cal
l drive.mount("/content/drive", force_remount=True).
```

Instructions

This document is a template, and you are not required to follow it exactly. However, the kinds of questions we ask here are the kinds of questions we want you to focus on. While you might have answered similar questions to these in your project presentations, we want you to go into a lot more detail in this write-up; you can refer to the Lab homeworks for ideas on how to present your data or results.

You don't have to answer every question in this template, but you should answer roughly this many questions. Your answers to such questions should be paragraph-length, not just a bullet point. You likely still have questions of your own -- that's okay! We want you to convey what you've learned, how you've learned it, and demonstrate that the content from the course has influenced how you've thought about this project.

Harmful Brain Activity Classification

Project mentor: Aayush Mishra

Team member: Yun You yyou17@jh.edu, Weiyu Li wli145@jh.edu, Fanshu Zhou fzhou21@jh.edu, Ziqi Chen zchen140@jh.edu https://github.com/IvyYY00/Mach-learning-675-Project

Outline and Deliverables

Uncompleted Deliverables

1. "Expect to complete #1": Utilize optimized CNN or LSTM models alongside 1d-CNN to capture temporal dynamics of EEG. The LSTM model requires the output of the 1-D CNN model as input, and we do not have enough time to complete it.
2. "Expect to complete #3": Attempt to resolve overfitting issues caused by deep learning models. Our model is experiencing underfitting.
3. "Would like to complete #1": Utilize novel hierarchical neural networks to capture spatial-temporal characteristics of brain EEG. Brain EEG data is typically in 1-D form,

and there is no solid evidence to suggest the existence of spatial-temporal characteristics.

4. "Would like to complete #2": Use adversarial techniques to improve model robustness. We do not have time to improve the robustness of the model.
5. "Would like to complete #3": Understand the metric formula behind the model in order to understand what aspects of the input EEG data is most important to detecting harmful brain activity. Deep neural networks lack interpretability, so it is difficult to determine which features in EEG data are particularly important for detecting harmful brain activity. Additionally, we also do not have enough time to learn about signal processing.

Completed Deliverables

1. "Must complete #1": Exploring data and using reliable prepossessing method to increase model performance(s). We discuss our dataset pre-processing in "Pre-processing" below.
2. "Must complete #2": Establish a baseline classification performance for predicting the probability of seizures using raw EEG data with a 1-D CNN. Evaluate this performance using the cross-entropy loss between the predicted probabilities and the observed targets. We discuss building and training 1-D CNN model in "Models and Evaluation" below.
3. "Must complete #3": Conduct performance evaluation and validation of the model. For validation we will randomly sample a different subset from the training set, and we will use the given test set for test evaluation. We discuss evaluating 1-D CNN model in "Results" below.
4. "Expect to complete #2": Exploring methods to further denoise and filter the data. We discuss denoising and filtering the data in "Pre-processing" below.

Additional Deliverables

1. We have decided to add a comparative model that splits the original EEG data into overlapping segments and computes the power spectral density to reduce dimensions to 3D, and uses an SVM model. We discuss this in "Models and Evaluation" below.
2. We have decided to use the original spectrogram data with a 2-D CNN model to compare the results with those from using EEG data with a 1-D CNN model. We discuss this in "Models and Evaluation" below.

Preliminaries

What problem were you trying to solve or understand?

In this project, we are trying to build a model with careful preprocessing that can use complex time-series data recorded from critically ill patients to identify patterns associated with abnormal brain activity. The main purpose of this project is to classify seizures and other harmful brain activities by selecting the label with the highest probability given EEG data.

By learning more about the data and building a robust model for prediction, these models could be tested in simulated clinical environments to assess their practicality and stability under real-world conditions. Meanwhile, this can enhance the speed and accuracy of diagnosing brain-related illnesses, help make decisions in emergency scenarios, and could reduce healthcare costs.

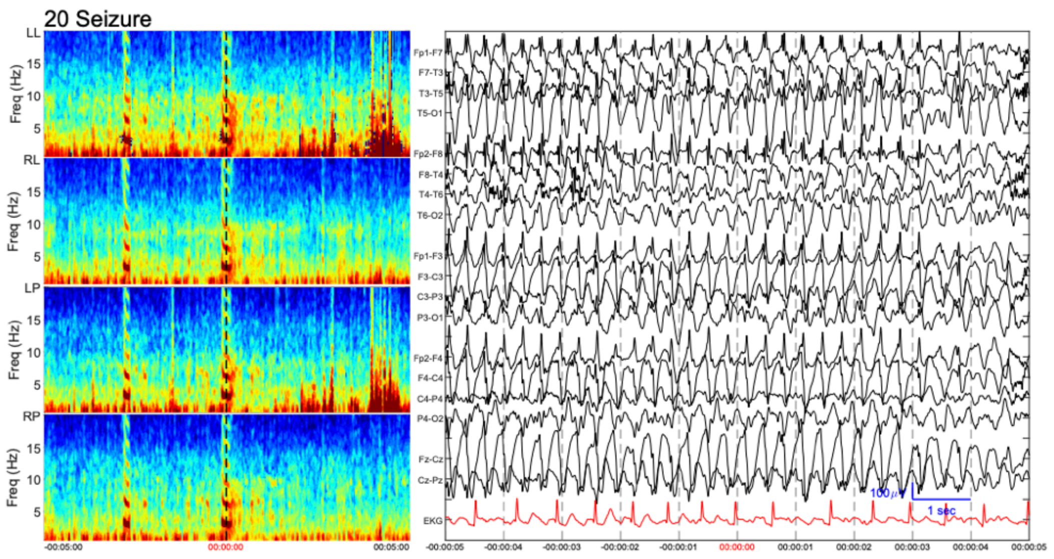
We have learned how to use CNNs and SVMs to detect images in class. We will apply these two models to this task to see their performance and limitations.

We choose this topic in hopes of exploring the application of the rapidly developing field of artificial intelligence in medical field, aiming to improve the early diagnosis rate of abnormal brain activity in patients.

However, when applying this method to the real world, we need to be careful with the data sources as all data are coming from sensitive medical data.

Here is an example of how eeg signal looks like

```
In [ ]: display(Image('/content/drive/MyDrive/Images/intro dataset.png', width=500))
```



Dataset(s)

We are using two related datasets in our project, both selected from Kaggle competition Harmful Brain Activity(HMS) Classification. The first dataset, named 'train.csv', includes reviews of 50-second long EEG samples by expert annotators. This dataset consists of 15 attributes across 106800 records. The target column, 'expert_consensus', indicates the majority vote among the annotators. The second file, 'subset_train_eegs', contains 653 '.parquet' raw EEG data files, each corresponding to a 10-second sub-EEG data segment extracted from the longer EEG samples. The way to link 'train.csv' to 'subset_train_eegs.csv' is by using 'eeg_id' and 'eeg_label_offset_seconds' to create a unique key. This allows us to read the specific EEG and its corresponding diagnostic result.

The dataset has 14,856 records for LPD class, 18,808 for Other, 16,702 for GPD, 20,933 for Seizure, 16,640 for LRDA, and 18,861 for GRDA.

The reason why we choose dataset from this competition is because it include enough records for us to learn the pattern of the dataset, and dataset is reliable and follow the data ethical. Moreover, ample datasets can effectively prevent potential overfitting issues in deep learning with CNNs.

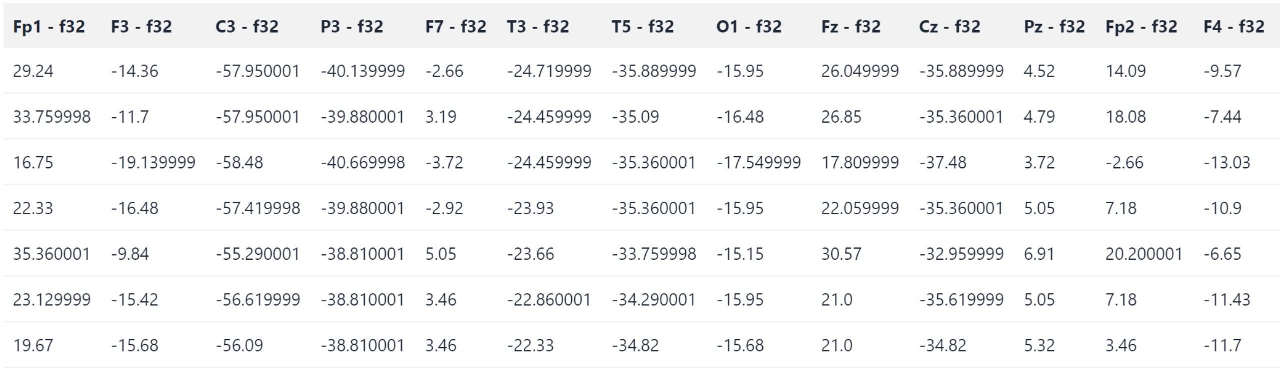
Here is an example of train.csv

```
In [ ]:
# Load your data and print 2-3 examples
import pandas as pd

csv_file_path = 'train.csv'
data = pd.read_csv(csv_file_path)

# Display the first 2 examples of the data
print(data.head(2))
```

```
       eeg_id  eeg_sub_id  eeg_label_offset_seconds  spectrogram_id  \
0  1628180742           0                       0.0          353733   
1  1628180742           1                       6.0          353733   

   spectrogram_sub_id  spectrogram_label_offset_seconds    label_id  \
0                   0                               0.0   127492639   
1                   1                               6.0  3887563113   

   patient_id expert_consensus  seizure_vote  lpd_vote  gpd_vote  lrda_vote  
\
0       42516          Seizure             3         0         0          0   
1       42516          Seizure             3         0         0          0   

   grda_vote  other_vote  
0          0           0  
1          0           0  
```

Here is an example of subset train eegs. These are 16 chanel(electrode) of eeg

```
In [ ]: display(Image('/content/drive/MyDrive/Images/spectrogram.png', width=700))
```



Fp1 - f32	F3 - f32	C3 - f32	P3 - f32	F7 - f32	T3 - f32	T5 - f32	O1 - f32	Fz - f32	Cz - f32	Pz - f32	Fp2 - f32	F4 - f32
29.24	-14.36	-57.950001	-40.139999	-2.66	-24.719999	-35.889999	-15.95	26.049999	-35.889999	4.52	14.09	-9.57
33.759998	-11.7	-57.950001	-39.880001	3.19	-24.459999	-35.09	-16.48	26.85	-35.360001	4.79	18.08	-7.44
16.75	-19.139999	-58.48	-40.669998	-3.72	-24.459999	-35.360001	-17.549999	17.809999	-37.48	3.72	-2.66	-13.03
22.33	-16.48	-57.419998	-39.880001	-2.92	-23.93	-35.360001	-15.95	22.059999	-35.360001	5.05	7.18	-10.9
35.360001	-9.84	-55.290001	-38.810001	5.05	-23.66	-33.759998	-15.15	30.57	-32.959999	6.91	20.200001	-6.65
23.129999	-15.42	-56.619999	-38.810001	3.46	-22.860001	-34.290001	-15.95	21.0	-35.619999	5.05	7.18	-11.43
19.67	-15.68	-56.09	-38.810001	3.46	-22.33	-34.82	-15.68	21.0	-34.82	5.32	3.46	-11.7

Pre-processing

First, since each EEG file corresponds to one patient, we determine each patient's 'true' disease based on a majority vote from expert consensus, and remove the EEG entries that have incorrect expert labels for each patient. Additionally, we retained only the data where there was unanimous agreement among all experts regarding a patient to ensure the certainty of the class assigned to each patient. Then, we found that there is a significant overlap between each subset of EEG segments—for example, some start at timestamp 0 and last for 10 seconds, while others start at 2 seconds and also last for 10 seconds. Therefore, we exclude the less ideal of two temporally close samples (≤ 2 seconds apart) from the same ID.

After this preliminary work, we conducted some data exploration and found that the new dataset has 1,036 records for LPD, 801 for Other, 586 for GPD, 338 for Seizure, 41 for LRDA, and 12 for GRDA. This indicates a highly unbalanced dataset. Consequently, we removed the classifications with the fewest samples, which are 'GRDA' and 'LRDA', since we cannot up-sample the dataset by simply make up the EEG record.

Additionally, we removed samples that contain little information by applying a total votes count threshold of 5. We are also developing a filtering method for the raw, time-based EEG data. We implemented a band-pass filter on the raw EEG signals to retain frequencies between 1-40 Hz, which are most informative for detecting seizures. We applied a moving average filter (using NumPy.roll) to reduce high-frequency noise. We also standardized the data and applied μ-law encoding, which is a dynamic range compression strategy to make them more distinguishable from noise.

Lastly, we removed the columns 'Fz, Cz, Pz, EKG' since these positions are not related to our task.

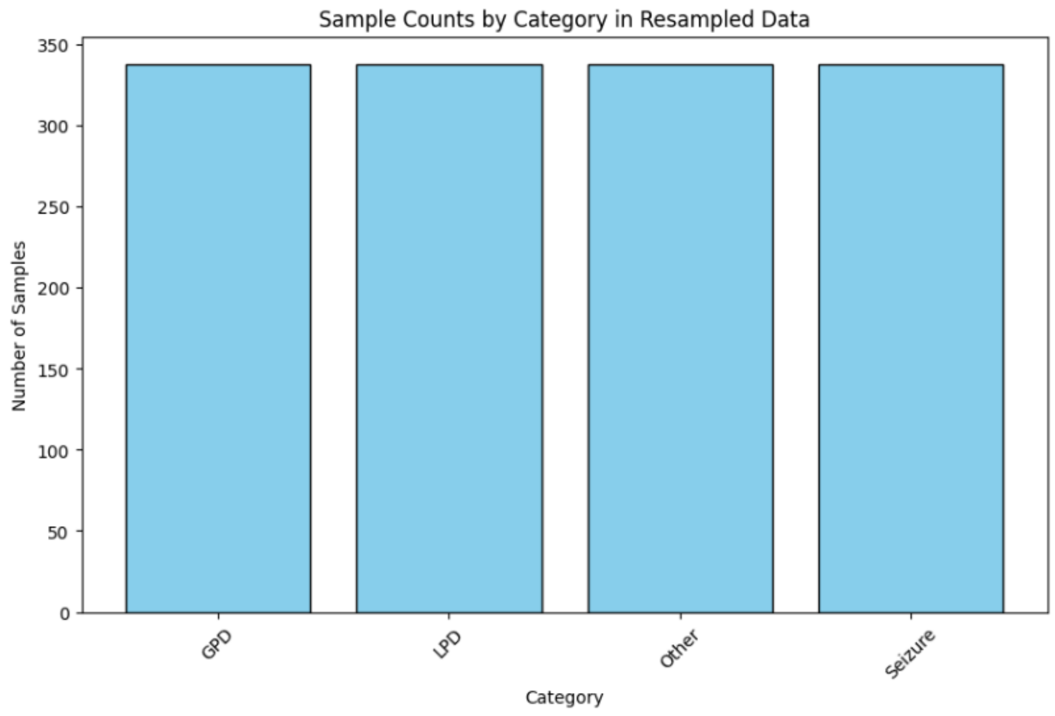
The imbalance in class distribution can lead to bias during the training of the dataset. To avoid this, we under-sampled the larger classes to match the number of records in the smallest class, thereby achieving a balanced dataset. The new dataset now has 338 records for each class, and we will apply this to following models.

By examining the statistical information, we found no missing values, and we cannot identify any records as outliers. Since we are predicting results using time-series signal data, we are utilizing continuous features for the prediction. Now, we have processed data, and the classes are unbalanced.We will put this dataset to each of the models and see their performances.

In 2DCNN, We extract the middle 50 seconds from the raw EEG waveform data corresponding to each 'eeg-id', store spectrograms corresponding to different EEG channel montages, and optionally denoise the signal using wavelet functions to obtain spectrogram image files.

Here's the class distribution after preprocessing

```
In [ ]: display(Image('/content/drive/MyDrive/Images/processed_dataset.png', width=5
```



Models and Evaluation

Experimental Setup

For both the 1-D CNN and SVM models, we determine the accuracy of the models in correctly classifying the EEG signals. In the 1DCNN, we use cross-entropy loss to measure the probability of the four classes. The reason we choose this is because cross-entropy loss measures the performance of a classification model whose output is a probability value between 0 and 1. In our project, we need to determine the probability of each class. Meanwhile, even if the predicted class is correct, using cross-entropy loss will also penalize those unconfident correct predictions to enhance performance.

We use cross validation and set to 10 fold to train model first, and randomly select 20% of the data for validation and 80% for training in both 1DCNN and SVM model.

Moreover, we also selected the records we removed during the preprocessing part and tested them with the 1D CNN to assess the model's generalization ability.

How did you evaluate your methods? Why is that a reasonable evaluation metric for the task?

For the 2DCNN model on spectrograms, we used validation accuracy along with visualization of model prediction confusion matrix to evaluate model performance. In addition to validation accuracies, a confusion matrix is a reasonable evaluation metric for supervised classification, as it contains information about bias, class imbalance, and whether there is reliable class separation in model performance. For our project, class separation was challenging in the raw EEG signals domain because there was overlap between `Other` and `Seizure` and other classes. Here, the high percentages of correct predictions along the diagonal of the confusion matrix indicate good class separation, and that spectrogram data is a useful input domain for our task of EEG brain activity classification.

What did you use for your loss function to train your models? Did you try multiple loss functions? Why or why not?

How did you split your data into train and test sets? Why? The CNN model spectrogram used `GroupKFold` to split the data into train and validation set. Within each epoch, the model trained on the train subset and both train and validation accuracies were recorded. The Kaggle provided test set did not contain

```
In [ ]:
```

Baselines

We use 1DCNN as baseline, apply it to a simple pre-preccessing data set and get accuracy around 41%.The reason why we want to use this as baseline is becasue it is a robust moodel to predict sequence data like EEG signals, it can detect and learn important features automatically.

In most related work, people are also using CNN related model to start the work.The main difference between ours and would be the detail of

What baselines did you compare against? Why are these reasonable?

Did you look at related work to contextualize how others methods or baselines have performed on this dataset/task? If so, how did those methods do?

1D-CNN

We chose a 1-D CNN model to classify harmful brain activity. 1-D CNNs are effective at capturing temporal patterns and dependencies in time-series data like EEG signals. We train this model using a dataset of labeled EEG data, employing a batch-wise training approach with a loss function like cross-entropy, suitable for multi-class classification. The training runs for 50 epochs, iterating over batches of data, where each batch undergoes a forward pass, loss computation, backward pass, and parameter update. We calculates the accuracy of the model as the percentage of correctly predicted labels per epoch to evaluate for our classification tasks. Finding the right architecture and tuning the hyperparameters such as the number of layers, size of the kernels, and dropout rate are major challenges in building and training our 1-D CNN model.

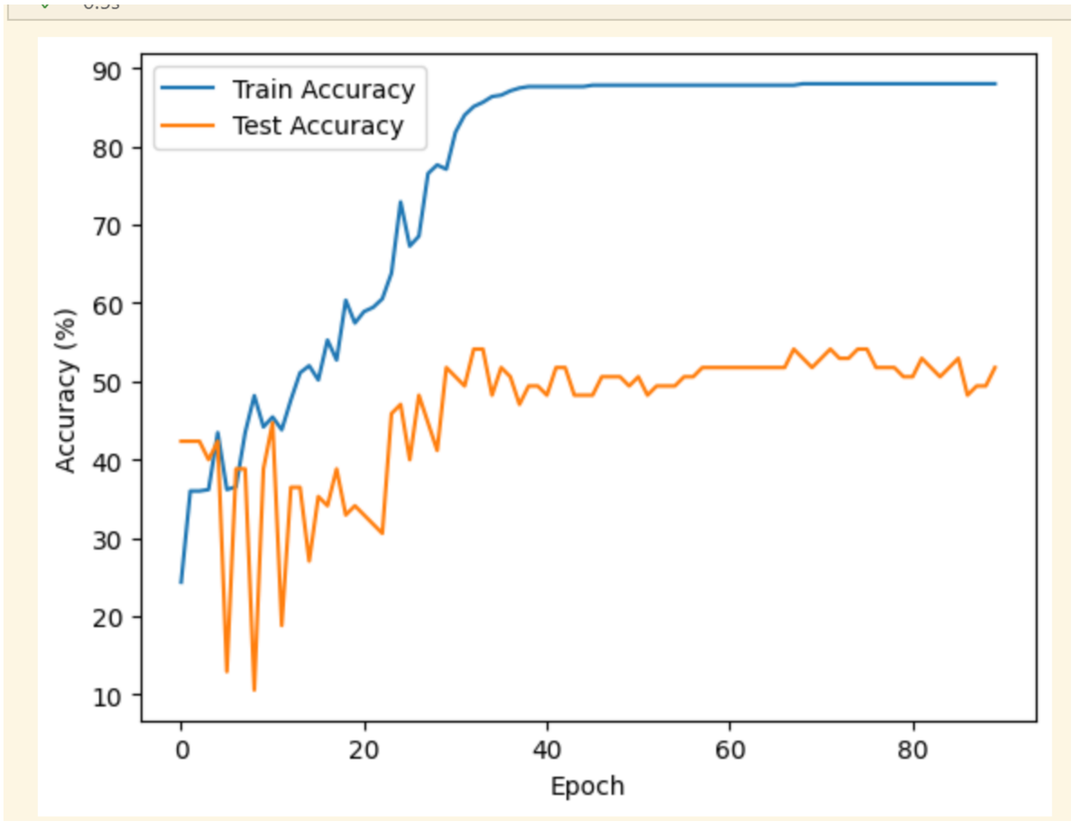
In [ ]:
```
# Code for training models, or link to your Git repository
```

In [ ]:
```
# Show plots of how these models performed during training.
#  For example, plot train loss and train accuracy (or other evaluation metr
#  with number of iterations or number of examples on the x-axis.
```

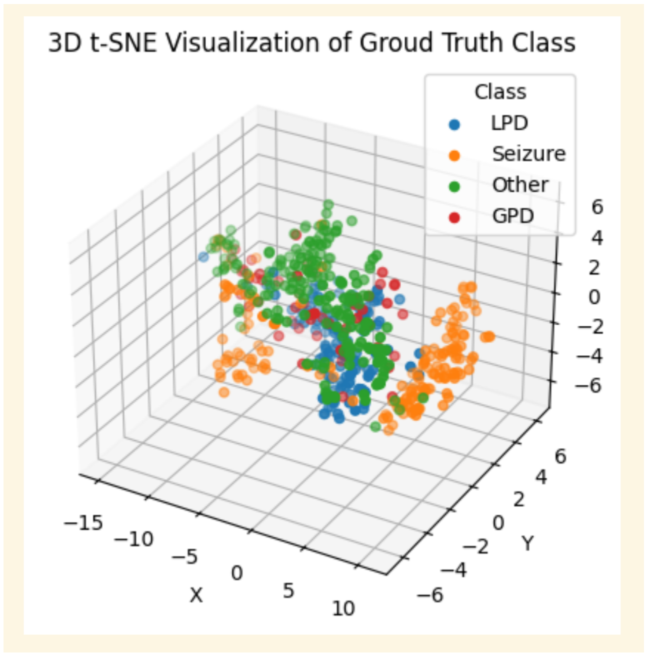
Results

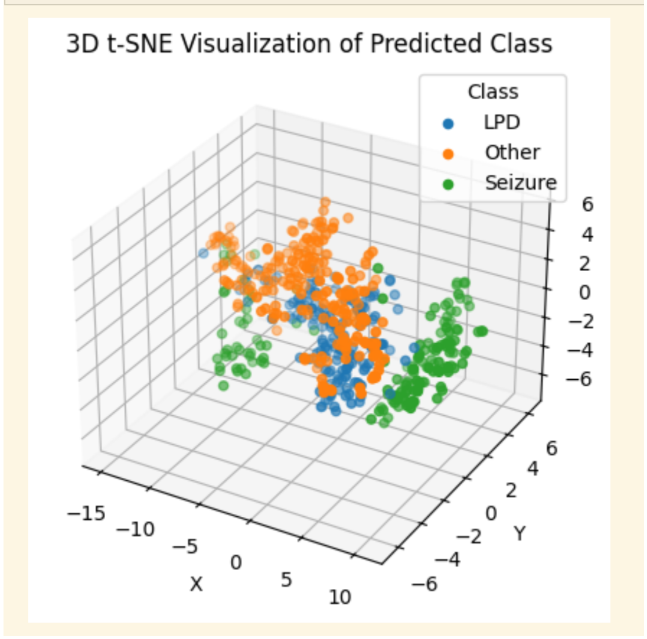
The validation accuracy (blue line) reaches high levels, close to 90%, while the test accuracy (orange line) plateaus around 50% and does not show significant improvement as training progresses. The gap between these two is surprisingly high. Usually, a large gap between validation and test accuracy indicates overfitting. However, we don't believe this is the case here; we think it's more likely due to the patterns of the two datasets being slightly different, causing the model to not learn the entire pattern of the whole dataset, which leads to underfitting. In this case, we may use more data to enhance the model's generalization ability and learn more patterns. The fulctuation in testing accuracy may be because the test data is too sparse and high dimensional.

```
In [ ]: display(Image('/content/drive/MyDrive/Images/1D accuracy update.png', width=
```



```
In [ ]: display(Image('/content/drive/MyDrive/Images/1D CNN output.png', width=300))
        display(Image('/content/drive/MyDrive/Images/1D CNN predict output.png', wid
```

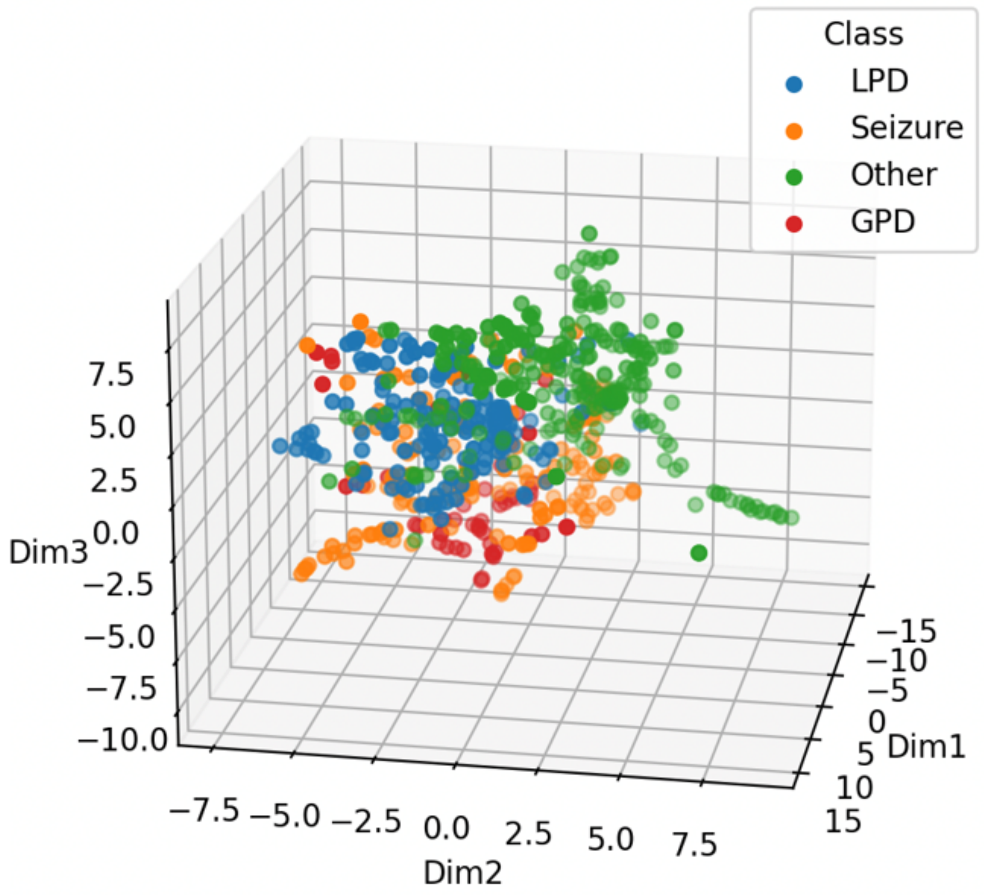


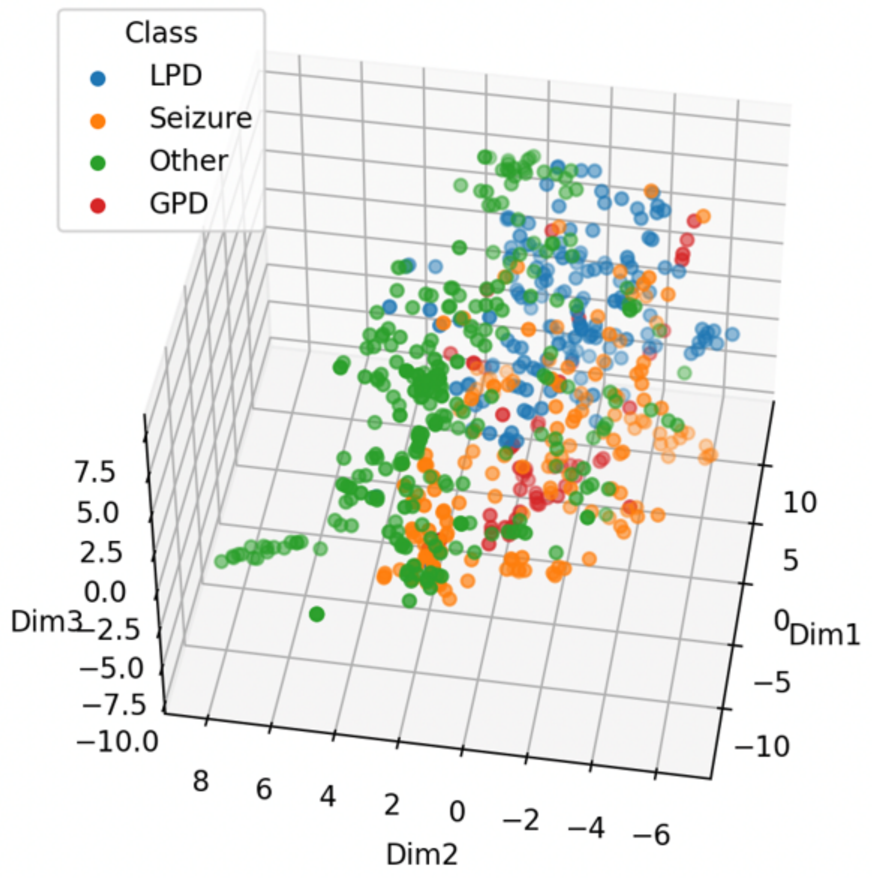


SVM

We want to gain a visualization of the 16*2500 eeg signals (in time domain), where 16 is number of channels and 2500 is the number of timestamps. We average the signal over 16 channels, and using the welch's method to convert the the signals into frequencies, and get a frequency count as feature vector for each frequency band. Then we use the TSNE decomposition to project it into 3D to gain a raw data representation.

```
In [ ]: display(Image('/content/drive/MyDrive/Images/raw_3d_1.png', width=500))
        display(Image('/content/drive/MyDrive/Images/raw_3d_2.png', width=500))
```


Class
LPD
Seizure
Other
GPD
Dim1
Dim2
Dim3
−15
−10
−5
0
5
10
15
−7.5
−5.0
−2.5
0.0
2.5
5.0
7.5
−10.0



Results

We can see that the class "others" is the most indistinguishable. Then apply a SVM model, we used a 10-fold cross validation, where 20% are used for validation dataset. Mean cross-validation accuracy: 0.6636363636363637 Accuracy on test set: 0.7 Classification Report: precision recall f1-score support

```
         GPD       0.00      0.00      0.00        11
         LPD       0.69      0.69      0.69        26
       Other       0.82      0.82      0.82        40
     Seizure       0.60      0.79      0.68        33

    accuracy                           0.70       110
```

This model was previously underfitting, so we used 10-fold cross validation. And changing from Linear kernel to an RBF kernel does not help. On different data, the model may be very unstable.

2D-CNN

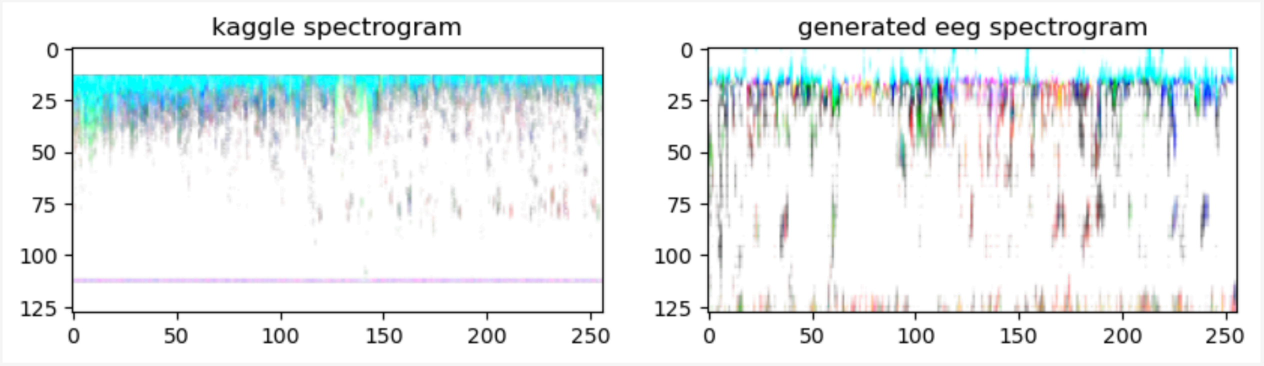
For the 2DCNN model on spectrograms, we used validation accuracy along with visualization of model prediction confusion matrix to evaluate model performance. In addition to validation accuracies, a confusion matrix is a reasonable evaluation metric for supervised classification, as it contains information about bias, class imbalance, and whether there is reliable class separation in model performance. For our project, class separation was challenging in the raw EEG signals domain because there was overlap between `Other` and `Seizure` and other classes. Here, the high percentages of correct predictions along the diagonal of the confusion matrix indicate good class separation, and that spectrogram data is a useful input domain for our task of EEG brain activity classification.

What did you use for your loss function to train your models? Did you try multiple loss functions? Why or why not?

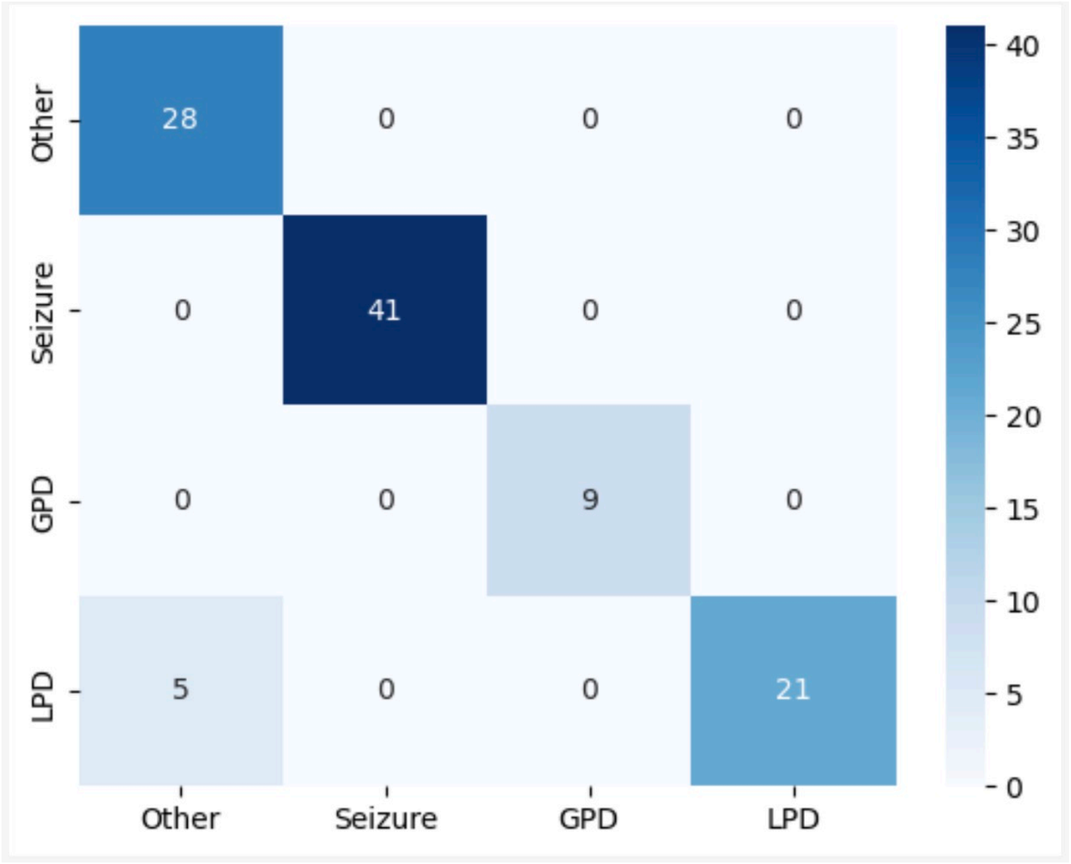
How did you split your data into train and test sets? Why? Our test set did not contain true labels, and evaluation of test performance on Kaggle is with hidden test evaluation metrics. We opted to use validation splits from our training set as test sets.

The spectrogram 2DCNN used `GroupKFold` to split the data into train and validation sets with `n_splits = 10`, the number of training epochs, and `group = train_ids` where `train_ids` is a list of `eeg_ids`. The main reason for this method is to ensure the model is evaluated on diverse samples from different patients and becomes more reliable in future performance on unseen data. Given the spectrogram data for this model was trimmed to one file per `eeg_id`, this method ensures each validation `eeg_id` is used once across folds, and there is diversity in patient samples in the validation set.

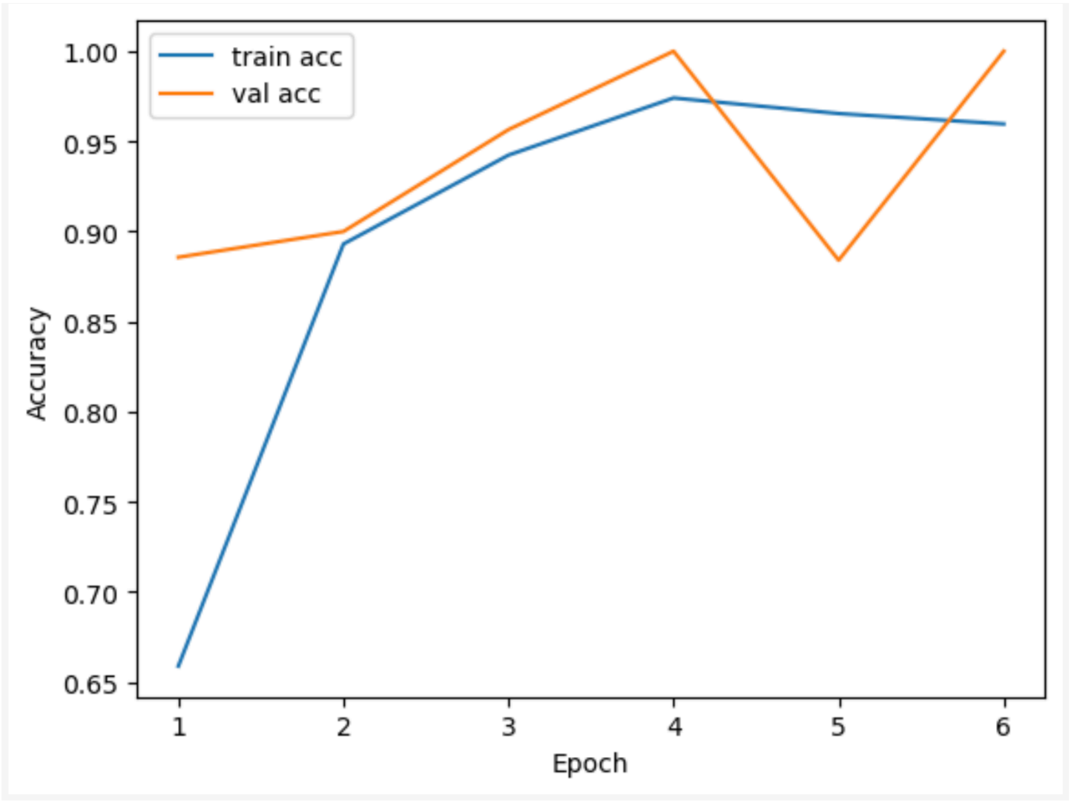
```
In [ ]: display(Image('/content/drive/MyDrive/Images/2D CNN input.png', width=800))
```



```
In [ ]: display(Image('/content/drive/MyDrive/Images/2D accuracy.jpg', width=500))
```



```
In [ ]: display(Image('/content/drive/MyDrive/Images/2D CNN accuracy 2.png', width=5
```



Results

For the spectrograms input format, we chose a CNN model. In this dataset spectrograms are processed to be two-dimensional, and pixels contain information about frequency-transformed signal densities. Thus, spectrograms can be viewed as images, and a 2DCNN model is well-suited to image-based classification tasks.

One challenge in using spectrograms as input was understanding the nature of spectrograms and the dimensions present in our spectrogram data. We consulted with our project mentor as well as referenced and cited the spectrogram preprocessing methods from publicly available Kaggle notebooks.

Once we had the spectrogram data ready, the implementation of our 2DCNN model is straightforward as we could rely on lectures and labs on CNN architecture. It did require some care to calculate the right dimensions after input passes through the convolutional layers. We trained the 2DCNN using batch gradient descent with a supervised learning paradigm.

Based on lab 3, we chose leaky relu as the activation functions and initialized weights to stabilize model performance.

The hyperparameters we evaluated for this 2DCNN were the learning rate and number of epochs. The model was sensitive to both parameters. With the larger learning rate of two rates, the model was less stable and performance fluctuated. When the model trained for more epochs, it started to overfit and validation performance dropped.

Discussion

What you've learned

Note: you don't have to answer all of these, and you can answer other questions if you'd like. We just want you to demonstrate what you've learned from the project.

In class, we have learned about Support Vector Machines (SVM) and Neural Network-based models. Both can be used to classify data. We apply these two models to differentiate between normal and abnormal EEG signals. Moreover, we use the same concepts, such as the loss function and accuracy, to evaluate our models.

In our project, we suprisly fond that he results from the 1-D CNN model show that while the training data can achieve an accuracy of close to 90%, the accuracy for the test data is less than 50%.

By working on this project, we have gained a deeper understanding of the construction of SVM and CNN models. Moreover, we have realized the importance of having balanced data. If the classes are imbalanced, the model will be biased toward the majority class, which can lead to poor performance. We will always keep this in mind for future machine learning projects.

Throughout the entire project, we received substantial assistance from our mentor, who provided us with many insights on how to handle EEG data. We are very appreciative of this support. The most helpful advice was his suggestion to use t-SNE to make the model more explainable, which greatly enhanced our understanding and presentation of the results.

We hope to try a multi-model method to classify data within the next two weeks, which involves taking the results from the 1-D CNN, segmenting them into fixed-size tokens, and inputting them into a transformers model to obtain the final results. At the same time, we will also attempt to improve the model's accuracy on test data by using more data for generalization.