Seizure vs. non-seizure using MLP

1. **Executive Summary:**
   Early detection of seizures is important to prevent injuries and ensure better management for patients with epilepsy. However, traditional methods have yet to be efficient for real-time and early detection of seizures. In this project, I strive to build a binary classification model that can classify seizure and on-seizure events using data from the Bonn dataset.

2. **Colab Notebook:** link

3. **Methodology:**
   The first step for every machine learning model is to preprocess data for the overall project:
   - First, I imported the necessary libraries including NumPy, Pandas, and Scikit-learn. The dataset was uploaded as CSV files, containing the four files given by Professor Rachel Bergstrom.
   - Then, I combined the datasets to create a unified dataset for both training and testing using `pd.concat()`. I combined the EEG signal datasets along rows to append the short signal data to the short label data. Similarly, the labels for full signal were concatenated with the full label data. This is the most important step to diversify the volume and variety of data for the future robust model.
   - After loading the dataset and combining them, I tried `data.dropna()` to see if there were any rows with missing values. I then printed some first few rows to verify the structure of the combined datasets. However, the given datasets are already clean and do not need any other extensive data cleaning. Only StandardScaler was used to ensure the uniformity of the data and avoid potential scale dominance during training.
   - Next, I used `segment_data()` to slice the dataset into smaller segments. This is an essential step when dealing with time-series data like EEG, which is to ensure that the data is processed in a small and manageable way that captures the temporal nature of EEG signals. Seizures can happen at any moment and it's of extreme importance to be detected promptly.
   - Finally, I split the dataset into training and validation sets using the `train_test_split` function from Scikit-learn to measure the efficiency of the model.

4. **Model architecture:**
   This project uses Multi-layer Perceptrons (MLP) thanks to its suitable application for the tabular data in the EEG dataset which originally contained numerical time-series data. MLP is a type of feedforward neural network, consisting of an input layer, one or more hidden layers, and an output layer. MLP is not dependent on the sequential nature of data yet still versatile; therefore, it can still learn useful features through different trial-and-errors in the hidden layers. MLP is also optimal in handling non-linearity thanks to its ReLU activation functions in the hidden layers, allowing it to capture non-linear relationships between various features. Compared to other more complex models like Convolutional neural networks (CNN) or Recurrent neural networks (RNN),

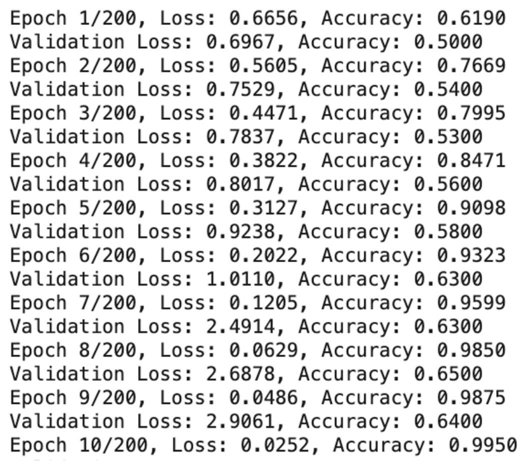
MLP is specifically simple and more time-efficient in training for these moderate-sized datasets. Although this model is simpler compared to the aforementioned two, it's still scalable to larger datasets and able to accommodate the increase in data size. Given that our EEG dataset is not that large, MLP is more than capable of providing an effective balance of performance and computational efficiency. Last but not least, aligning with the main objective of this project – binary classification – MLP is an excellent choice as the final output layer can simply represent a binary decision.

The breakdown of model architecture:

- Input layer: contains 178 features corresponding to EEG data measurements.
- Hidden layer: contains 3 hidden layers with sizes 128 → 64 → 32 neurons. These three layers are the results of numerous experiments and tuning based on performance and the model's ability to generalize the data. This trial-and-error process is crucial for any deep learning model development. With the three proposed neuron numbers for the hidden layer, the model can capture the broad patterns in the data. Moreover, after each linear layer, a ReLU activation was applied to enable the model to capture more complex relationships in the data.
- Output layer: contains a single neuron that results in a scalar value between 0 and 1. This layer is used for binary classification, with 1 representing seizure and a value close to 0 representing non-seizure.

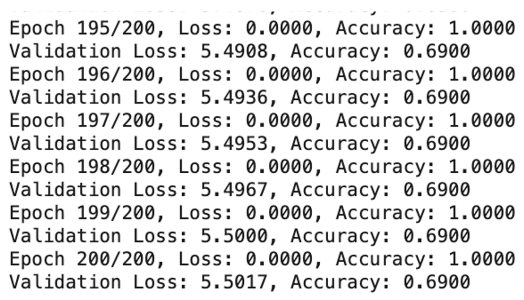
5. **Training Procedure:**

The model was trained using the PyTorch framework. The training loop includes Binary Cross-Entropy loss, Adam optimizer, and performance evaluation metrics based on accuracy, precision, recall, and F-1 score. After running 200 epochs, I received the results as follows:



```
Epoch 1/200, Loss: 0.6656, Accuracy: 0.6190
Validation Loss: 0.6967, Accuracy: 0.5000
Epoch 2/200, Loss: 0.5605, Accuracy: 0.7669
Validation Loss: 0.7529, Accuracy: 0.5400
Epoch 3/200, Loss: 0.4471, Accuracy: 0.7995
Validation Loss: 0.7837, Accuracy: 0.5300
Epoch 4/200, Loss: 0.3822, Accuracy: 0.8471
Validation Loss: 0.8017, Accuracy: 0.5600
Epoch 5/200, Loss: 0.3127, Accuracy: 0.9098
Validation Loss: 0.9238, Accuracy: 0.5800
Epoch 6/200, Loss: 0.2022, Accuracy: 0.9323
Validation Loss: 1.0110, Accuracy: 0.6300
Epoch 7/200, Loss: 0.1205, Accuracy: 0.9599
Validation Loss: 2.4914, Accuracy: 0.6300
Epoch 8/200, Loss: 0.0629, Accuracy: 0.9850
Validation Loss: 2.6878, Accuracy: 0.6500
Epoch 9/200, Loss: 0.0486, Accuracy: 0.9875
Validation Loss: 2.9061, Accuracy: 0.6400
Epoch 10/200, Loss: 0.0252, Accuracy: 0.9950
```

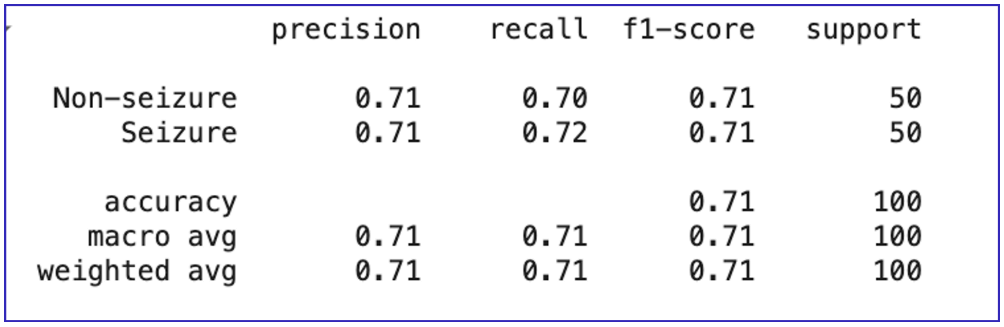
The training loss decreases steadily, indicating that the model is effective at learning and minimizing the error while being trained on the training dataset. Generally, this is a good sign when the model is learning the patterns in the data. Training accuracy also increases steadily, almost reaching 100% at the final epoch, meaning that the model is classifying almost all training data accurately, with very few misclassifications by the end of the 200 epochs. However, we can clearly see the overfitting of this mode; when the model's performance on the validation set does not match. This means that there's something wrong hindering the ability to generalize this model. In this case, the near-perfect accuracy of the training data shows that the model is memorizing the data rather than learning generalizable patterns.



```
Epoch 195/200, Loss: 0.0000, Accuracy: 1.0000
Validation Loss: 5.4908, Accuracy: 0.6900
Epoch 196/200, Loss: 0.0000, Accuracy: 1.0000
Validation Loss: 5.4936, Accuracy: 0.6900
Epoch 197/200, Loss: 0.0000, Accuracy: 1.0000
Validation Loss: 5.4953, Accuracy: 0.6900
Epoch 198/200, Loss: 0.0000, Accuracy: 1.0000
Validation Loss: 5.4967, Accuracy: 0.6900
Epoch 199/200, Loss: 0.0000, Accuracy: 1.0000
Validation Loss: 5.5000, Accuracy: 0.6900
Epoch 200/200, Loss: 0.0000, Accuracy: 1.0000
Validation Loss: 5.5017, Accuracy: 0.6900
```

Validation loss fluctuates and then starts to grow after a few epochs, suggesting the overfitting of this model. This means that the model becomes too specialized in predicting based on the training data and loses its capability to generalize the validation set – or let's say the unseen data. Furthermore, the validation accuracy plateaus around 68%, showing a stagnation in the model's performance. This is evident when the training metrics improve while the validation metrics worsen and are stagnant over time. This 68% is much lower than the near 100% accuracy seen on the training set, again proving the overfitting of this proposed model. While this model performs very well on the training set, it struggles to generalize the validation set and make validation performance plateau. This might be attributed to the mismatch between training and validation data distribution.

6. Classification Results:



```
                precision    recall  f1-score   support

  Non-seizure       0.71      0.70      0.71        50
      Seizure       0.71      0.72      0.71        50

     accuracy                           0.71       100
    macro avg       0.71      0.71      0.71       100
 weighted avg       0.71      0.71      0.71       100
```

Across the two classes (seizure and non-seizure), support (=50), macro average (=0.71), and weighted average (=0.71) are the same. Thus, the model is tested equally on both classes, enables all metrics to have similar performance, and gets equal support. Firstly, the precision measures how many true positives are out of all predicted positives (true positives + false positives), telling how many of the predicted non-seizures are actually non-seizures and seizures are seizures. With 0.71 for each, 71% of the events predicted as seizures by the model are indeed actual seizures, and the same explanation is applied to non-seizures. The model is equally confident and accurate in predicting both seizure and non-seizure events. Secondly, recall measures how many actual positives are correctly identified by the model, or shows how well the model detects seizures or non-seizures. This model accurately detects 72% of the actual seizure events and 70% of the actual non-seizure events. This model is slightly better at recalling seizure events than non-seizure events. Finally, the F-1 score is the mean of precision and recall, providing a balanced measure for both. Both classes have an F-1 score of 0.71, showing that the model balances precision and recall fairly well. With an accuracy of 71%, the model is performing decently, but this signals room for improvement, especially the solution for overfitting and the lack of ability to deal with the validation set.

7. Conclusion:

This is my first time pushing myself to build a machine learning model, learning and applying along the way, this model is a fresh start to dive into more advanced data science knowledge. This MLP model has been successfully developed to classify seizure and non-seizure events from the Bonn dataset, got trained on both full and short signal data, and showed promising performance on the training set. On the training data, accuracy reaches nearly 100%, and training loss decreases steadily over time. However, challenges always emerge when working on any project. Here, I encounter overfitting in my model. While this model performed exceptionally well on the training data, its performance on the validation set stagnated. This gap suggests that the model struggles to generalize unseen data, and instead of learning generalizable patterns, it's only able to memorize the data.