[Unnamed Company] Assessment

Arnold Yeung

September 1st, 2020

In this project, we implement various machine learning and deep learning models for the classification of closed-eye and opened-eye EEG recordings.

Files

The following files were produced in these experiments:

- **q1_test_labels.npy**: labels of the test dataset of the best-performing traditional machine learning model (i.e., random forest)
- **q1_rnn_outputs.npy**: prediction probabilities of the test dataset of the LSTM model
- **q1_rnn_labels.npy**: labels of the test dataset of the LSTM model
- **q1_fft_outputs.npy**: prediction probabilities of the test dataset of the LSTM-FFT model
- **q1_fft_labels.npy**: labels of the test dataset of the LSTM-FFT model
- **q1_tflite_outputs.npy**: prediction probabilities of the test dataset of the LSTM model ran through the TFLite Interpreter
- **q1_tflite_labels.npy**: labels of the test dataset of the LSTM model ran through the TFLite Interpreter

The trained models are stored in **./models/**.

Traditional Machine Learning Models

We use three conventional machine learning classification models: logistic regression, support vector machine, and random forest.

Feature Processing

While the EEG data is time-series by nature, there is no obvious indication of the importance of the relative temporal position of each sample. Therefore, we use a Fast Fourier Transformation (FFT) to convert the EEG data into the frequency domain, a common domain for analyzing EEG signals. The power of each frequency (i.e., each 1 Hz range) is used a feature, with the frequency range from 2 Hz to 35 Hz.

Conversion to the frequency domain is also convenient because the deployed traditional machine learning models are not used typically as time-series models.

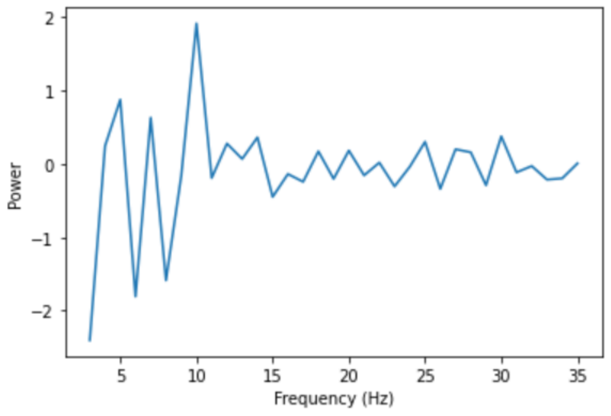

Figure 1: Example of a FFT transform from temporal EEG data

Results

The training dataset is split into 3 subsets with an 80-10-10 split: training, validation, and test. The training subset is used to train the model, the validation subset is used for hyperparameter tuning, and the test subset is used for evaluating the expected accuracy of the model.

Table 1: Optimal ML model accuracies

	Validation Accuracy	Test Accuracy
Logistic Regression	0.645	0.60
Support Vector Machine	0.811	0.70
Random Forest	0.756	0.68

The results suggest that Random Forest performs best across the 3 machine learning models. There are large differences between the validation and test accuracies, suggesting that models do not generalize well across the two subsets. This may be caused by over-fitting of the model (poor generalization) or the subsets having too few data samples to obtain an accuracy which generalizes well across the expected data distribution.

Observing the ROC curves of the test subset, it is more noticeable that the performances of models are only marginally better than a random baseline.

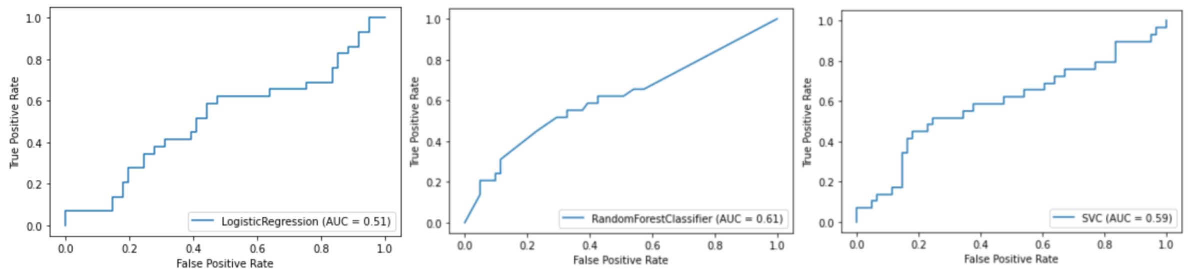

Figure 2: ROC curves of each machine learning model

Deep Learning Models

We implement 2 deep learning architectures: a time-series model using an LSTM-unit and a model which attempts to include both time-series and FFT components of the data.

Architectures

The time-series model includes two linear layers, before and after the LSTM-unit. The first linear layer converts the inputted data into an embedding and the second linear layer (with sigmoid activation) produces a binary classification. The inclusion of the LSTM-unit attempts to capture any distinguishing temporal characteristics within the 1 window epoch.

From the effectiveness of the FFT features observed in the traditional machine learning models, we also include a time-series and FFT model. This model is similar to the time-series model by incorporating an LSTM-unit to capture temporal characteristics, but also includes a parallel branch which converts the time-series data into the frequency domain and applies linear transformations. The outputs of both branches are concatenated and passed into a linear-sigmoid layer for binary classification.

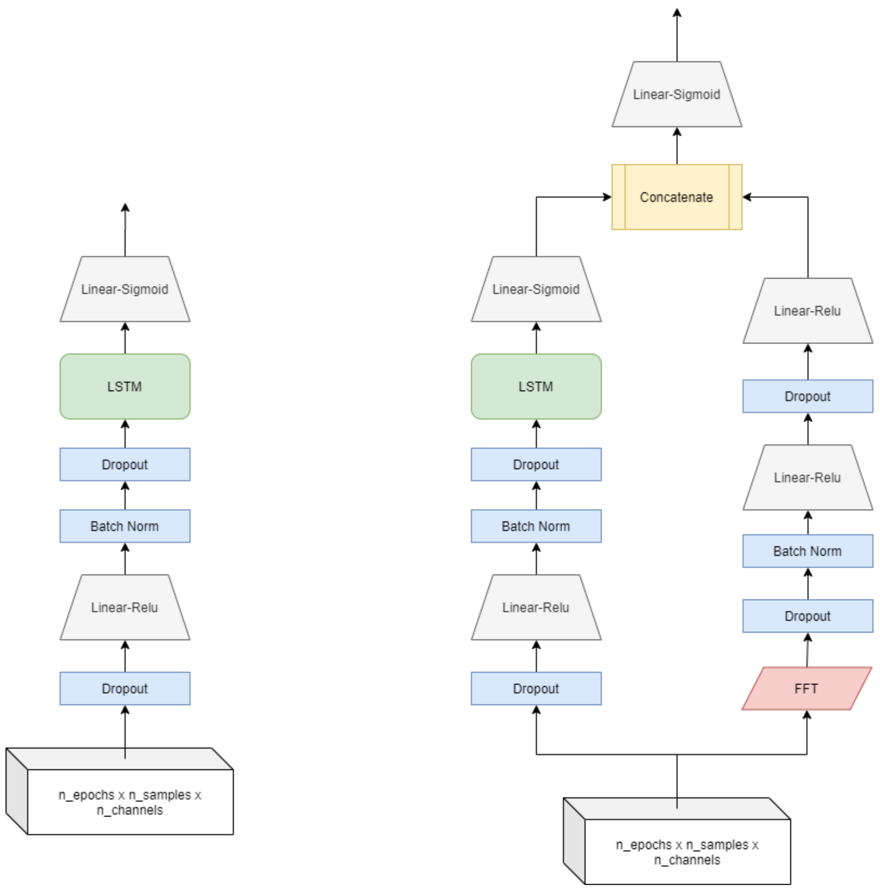

Figure 3: Two deep learning architectures: a LSTM network (left) and a LSTM-FFT network (right)

After initial runs of the models, we notice significant over-fitting from the models. Dropout and normalization layers are included into both models to reduce over-fitting and reduce exploding/vanishing gradients.

Both models are trained with an Adam optimizer across 100 iterations. Due to the slight imbalance across the data labels, the binary cross-entropy loss is class-weighted such that the losses of both classes are equal. Ideally, a grid-search (or Bayesian optimization) of the hyperparameters (e.g., batch size, learning rate, dropout rate, number of units) would be used to conduct hyperparameter tuning. However, due to limitations in time and computational resources, hyperparameter tuning is conducted manually. Therefore, we cannot be certain that the observed results represent the architectures optimally.

Results

The training dataset is split into 3 subsets with an 80-10-10 split: training, validation, and test. The training subset is used to train the model, the validation subset is used for monitoring the progress of the training, and the test subset is used for evaluating the expected accuracy of the model.

Table 2: Deep learning model accuracies

	Validation Accuracy (last iteration)	Test Accuracy
LSTM	0.700	0.689
LSTM-FFT	0.722	0.722

The LSTM-FFT model obtains an accuracy greater than that of the LSTM model, suggesting that the inclusion of FFT features may improve classification accuracy.

When we observe the training of the LSTM model, we see that while the training loss decreases overtime, the validation loss does not. This suggests that the model fails to generalize between the data within these two subsets, which may be potentially caused by the small amount of data within each subset or the inability of the LSTM architecture in capturing any generalizable features within the data distributions.

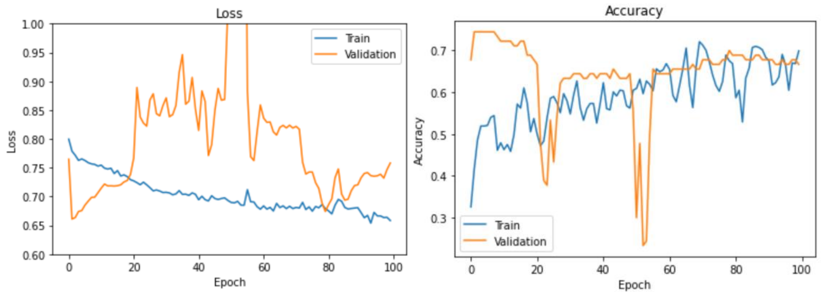

Figure 4: Loss and Accuracy of the LSTM model during training

When we observe the training of the LSTM-FFT model, we see that both the training and validation losses decrease overtime. However, while the training loss continues to decrease after epoch 40, the validation loss stays relatively the same, suggesting overfitting. This may be due to the complexity of the architecture and the small amount of data for the model to generalize to. Nonetheless, we observe that both the training and validation accuracies increase during training.

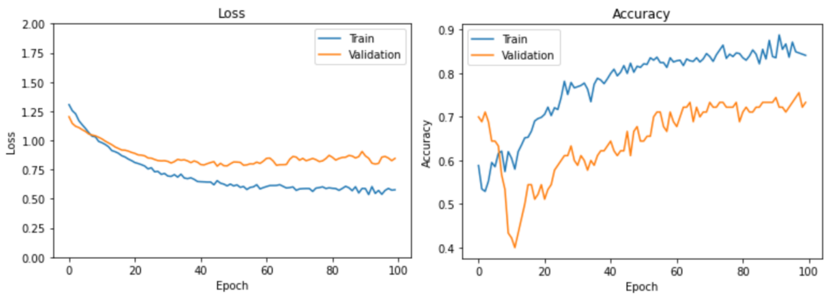

Figure 5: Loss and accuracy of the LSTM-FFT model during training

Comparison

From the test accuracies of each model, we observe that the deep learning LSTM-FFT model performs best, with an accuracy of 0.722. While the traditional machine learning models may have higher validation accuracies, their test accuracies are significantly different, suggesting the inability to generalize across the different subsets of data. The deep learning models, on the contrary, show similar validation and test accuracies, suggesting generalizable results.

The inclusion of the temporal and frequency components of the signals in the LSTM-FFT model may be a potential reason why this model outperformed the others. The traditional models suggest that there may be features in the frequency domain which may be relevant for classification, while the LSTM model suggest (less effectively) that there may be temporal features which may also be relevant. The LSTM-FFT model captures both components.

However, due to the limited data available (~900 instances within the training dataset), it may be difficult for more complex models, such as deep learning architectures, to generalize. This may result in over-fitting, as observed in the LSTM-FFT model, and erratic changes in the loss during training, as observed in the LSTM model. Traditional machine learning models are less complex and may be less prone to over-fitting given the limited data size, while only compromising a certain level of accuracy.

Limitations

Significant limitations of these experiments include the available data and the lack of proper hyperparameter tuning for the deep learning models.

The available training dataset contains ~900 instances, which may be an insufficient amount for training (especially for complex models), due to the variability across EEG data. Several instances also contained incomplete data (i.e., NaN values) and were removed from the training dataset during experimentation. The available EEG may also contain unwanted artifacts which should be removed prior to input into machine learning models.

Due to limited computational resources (e.g., GPU) and the time required for training deep learning architectures, the performance of the proposed models may not be optimal. Neither grid-search nor Bayesian optimization were conducted for hyperparameter tuning due to the expected computation and time required. Hyperparameter tuning for deep learning architectures were instead conducted manually and iteratively.