Motor Imagery Classification and Subject Identification

Shruti Mohanty
ECE Department, UCLA
UID: 705494615
shrutimohanty@ucla.edu

Boya Ouyang
MSE Department UCLA
UID: 005037574
boyao@ucla.edu

Yashwanth Bharadwaj Vedula
ECE Department, UCLA
UID: 705627385
yash2699@ucla.edu

Abstract

This project focuses on learning the representations for the Electroencephalogram (EEG) signals that are responsible for motor imagery. This can be utilized in multiple prediction tasks. The first task explored in this project is motor imagery classification. Various neural network architectures are implemented and tuned for this purpose - CNN, LSTM, and GRUs. The insights from motor imagery task classification are leveraged to identify the subject given the EEG data, which is our second task. A combination of CNN-LSTM architecture is used to learn the EEG distribution by training on the entire dataset. Using the tuned model, we perform the task of transfer learning to predict subject identities given their EEG representations. We have discussed the results from training on every subject, and the entire dataset, and have compared the various architectures implemented. Analysis has been performed on the data after certain pre-processing deduced from the effect of time period on accuracy. We achieve an accuracy of 73% for motor imagery classification, and 85% for subject identification.

1. Introduction

Electroencephalography (EEG) is a noninvasive method to measure brain activity through the recording of electrical activity across a patient's skull. Based on EEG data from nine subjects, we employ neural network to predict the imagery task performed by each subject. We will discuss the neural network architectures in the following sections.

1.1. Convolutional Neural Network (CNN)

Similar to ordinary Neural Networks, CNNs are made up of neurons that have learnable weights and biases.A simple CNN is a sequence of layers, and every layer of a CNN transforms one volume of activations to another through a differentiable function. Specifically, the CNN is a combination of two basic building blocks: the convolution block and the fully connected Block. The convolution block consists of the Convolution Layer and the Pooling Layer. This layer forms the essential component of Feature-Extraction. The Fully Connected Block Consists of a fully connected simple neural network architecture. This layer performs the task of Classification based on the input from the convolutional block.

1.2. Recurrent Neural Network(RNN)

Recurrent neural network is another type of neural network in which the output form the previous step is fed as input to the current step. Different from traditional neural networks in which all the inputs and outputs are independent of each other, RNNs have a hidden stage which used to capture information about the previous inputs. Thus RNN is useful to analyze sequence of data. However, RNN model has a major drawback, called vanishing gradient problem, which prevents it from being accurate. To solve the issue, Long Short Term Memory networks(LSTM) and Gated recurrent unit (GRU) has been proposed.

LSTM is a special kind of RNN, capable of learning long-term dependencies.It is explicitly designed to avoid the long-term dependency problem. Similar to standard RNN, LSTM forms a chain of repeating modules of neural network. However, the repeating module has a different structure. Instead of having a single neural network layer, LSTM has a cell state, carefully regulated by a forget gate, an input gate, and an output gate to optionally let information through.

GRU is very similar to LSTM, it uses gates to control the flow of information. But unlike LSTM, it does not have a separate cell state (Ct). It only has a hidden state. Besides, there are primarily two gates in a GRU as opposed to three gates in an LSTM cell. The first gate is the reset gate and the other one is the update gate. The Reset Gate is responsible for the short-term memory of the network, while the Update gate is responsible for long-term memory. Due to the simpler architecture, GRU is faster to train than LSTM.

Since we used shorter subsequences to train the model it avoids the vanishing and exploding gradient problem. However, using subsequences reduced information from the

past, which can limit the ability of the model to learn. To alleviate this problem, we proposed using a combination of CNN with RNN. The hyper parameters are optimized by performing a grid search on the validation data.

1.3. Subject Identification and Transfer Learning

The previous sections explore the different architectures for motor imagery classification. We further investigate if our learnings from previous architecture can help in other tasks - one of them is to identify a subject given their EEG data. The dataset provided has 9 subjects, so we perform our analysis on them using the successful architecture for Motor Imagery Classification (MIC). We draw inspiration from [1] for this task.

We used transfer learning to take us from MIC to Subject Identification (SI). The features are extracted from the CNN-LSTM model for MIC to transfer knowledge. The hidden layer values for each raw data point are extracted, and the output from the last convolutional layer is stored. A softmax classifier is trained with these representations as input for SI.

2. Pre processing

An important step in the process of building a machine learning model is the pre-processing of its data. The model is as good as the data it has worked upon. The pre-processing steps can either be normalizing, resizing or cropping etc.. or in the case that the data is scarce, then we can use data augmentation techniques to obtain more of the data. In our model, we examined the motor imagery data for a single channel and deduced that imagery data for time stamps greater than 500 have not shown any significant change across the four different tasks. So we trimmed the data to the first 500 time steps in our classification model.We have trained our models considering the first 750 time steps and all the time stamps but did not observe any significant change in the testing accuracy. The original dataset has 2115 samples which was considerably less, so we used data augmentation techniques to get more data. The trimmed data is passed through a max pooling function, an average filter and added noise to be robust to subtle changes observed during testing and concatenated the data obtained from both the above processes. We also subsampled the trimmed data twice and concatenated this data with the data from max pooling and averaging filter obtaining a total of 8640 samples for classification.

3. Results

3.1. Motor Imagery Classification

In Section 5, we provide details of the architectures. We trained these architectures with the data for all the subjects. In Table 1, we report the accuracy over the subjects separately and the overall accuracy, i.e., the accuracy computed on the entire test data.

Results on DeepCNN- The CNN used in this project consists of four convolutionary blocks and one fully connected layer. Dropout and batch normalization layers are also added to prevent overfitting the data. The parameters of the CNN is tuned using grid search. The averaged accuracy is 70% for the entire dataset.

Results on LSTM- A standard LSTM architecture was first used to train the entire dataset. The architecture had 2 layers of LSTM with 200 and 50 units each. The accuracy for this model was poor with the average accuracy on the entire dataset being 50%. We suspect the reason for this is the exploding gradients problem (or vanishing gradients). Hyper parameter tuning didn't help much, and the model would either overfit or test accuracy would increase only by 1-2%.

Results on CNN-LSTM- CNN-LSTM cascaded architecture showed an increased accuracy when compared to just LSTM. The dataset overfitting problem was also largely reduced by carefully selecting the dropout parameters and the number of layers in the model. The input size of the LSTM was restricted to avoid the model gradients from blowing up. A fully Connected network was used before LSTM to reduce convolution layers dimension to 100. Only one layer of LSTM with 20 units and a dropout of 0.6 is added. The average accuracy on the entire dataset was 73%. The subject was evaluation results are elaborated in Table 1. Some subjects perform better than the others like Subjects 3,5, and 9. It could be that the pre-processing was ideal for their EEG bands. In Figure 1, we show the training and validation loss plot for this mode over 85 epochs.

In addition to training the model on the entire dataset, we also trained this model subject-wise. The performance for this has been reported in Table 2. Accuracies ranged from 30% to 57%. Our analysis from this was that training the model on the entire dataset is definitely more helpful than subject-wise training.

Results on CNN-GRU- We trained using two different CNN - GRU combination as well as a GRU model without any convolutional layer. The latter model did not perform well compared to the former combination model owing to the fact that convolutional layers generate the important features that were necessary for the model to generalize. GRU model with its lesser parameters is faster to train compared to LSTM but is not necessarily accurate when the dataset is large. Since we used a CNN

first to extract the key features, we get a sufficiently higher testing accuracy. The model with 3 GRU Units obtained an overall testing accuracy of 66% compared to 64% that was obtained when we use a single GRU unit which indicates that we can achieve better testing accuracy when we consider the number of GRU units as a hyper-parameter to be tuned. We obtained reasonable subject-wise accuracies as presented in Table 1.

3.2. Analysis with respect to time period

In Figure 2, we plotted accuracy as a function of time to understand how many segments of the data are important for this task. From the graph, we see that after 250 to 300 index of the time signal the accuracy saturates with a maximum increase by 1% for the entire data sequence. We have sub sampled our data to 250th index and concatenated it as mentioned in Section 2.

3.3. Subject Identification

For this task, we use the subject from the person_train and person_test datasets as the labels. CNN-LSTM was the best performing overall architecture for MIC. This architecture was directly used and the softmax classifier, in the end, had 9 units for the 9 subjects. The accuracy for this model is 83%. This direct learning was compared with results from transfer learning. The CNN-LSTM model representation was extracted from the last convolution layer. This was given as an input to the softmax classifier with the LSTM unit for classification. We were able to achieve an accuracy of 85% using this. Transfer Learning is a very useful algorithm when the data available at hand is limited, like EEG data for example. In Figure 3, we show that transfer learning accuracy is almost similar or better than the direct learning approach, and this can be explored more even for motor imagery classification.

4. Discussion

The usage of dropout and batch normalization helped all architectures tackle overfitting to some extent and the problem of covariate shift. The effect of this was noticed as we added more layers to our network. A deep network makes more abstract and non-linear feature constructions that could be related to the electrodes in the dataset. These patterns are deduced by the black box networks to some extent. Using the CNN to learn features and compress them, and then feed them to LSTM gave us the best accuracy, leveraging the advantage of LSTM to learn long-term dependencies. On arriving at a good architecture for motor imagery, we could extend our model to other tasks. We chose subject identification as we have a dataset present for that as well with labels. Transfer Learning performed comparatively well for this task when compared to raw data. We could conclude that the black box feature representations learnt not only gave us "accuracy" as a number but are actually meaningful. From this, we also inferred that EEG bands for motor imagery are different amongst people, and that's why we could perform subject identification from our EEG representations learned from the model. This would mean that the neural architectures acquire the frequency band-specific information as well.

References

[1] W. X. Y. Z. Mao and Y. Huang. Eeg-based biometric identification with deep learning. In 2017 8th International IEEE/EMBS Conference on Neural Engineering (NER), pages 609–612. IEEE.

5. Architecture

Conv2D: 2 Dimensional Convolution, FCNet: Fully Connected Network. Across all the architectures the padding was kept as 'same' to preserve the dimension of the tensor.

5.1. Data Augmentation

Discussed in Section 2.

5.2. Loss Function

Categorical cross entropy loss function is used. Labels are one hot encoded to be evaluated by this function later on.

5.3. Optimiser

Adam Optimiser is used along with the tuned learning rate parameter for each architecture

5.4. Deep CNN

Learning Rate : 0.0073, **Epochs** : 55

Layer 1: 25 Conv2D filters kernel : (10,1), stride: (1,1), Activation: ReLU $->$ MaxPool: size: (3,1) $->$ Batch Normalization $->$ Dropout : 0.6

Layer 2: 50 Conv2D filters kernel : (10,1), stride: (1,1), Activation: ReLU $->$ MaxPool: size: (3,1) $->$ Batch Normalization $->$ Dropout : 0.6

Layer 3: 100 Conv2D filters kernel : (10,1), stride: (1,1), Activation: ReLU $->$ MaxPool: size: (3,1) $->$ Batch Normalization $->$ Dropout : 0.6

Layer 4: 200 Conv2D filters kernel : (10,1), stride: (1,1), Activation: ReLU $->$ MaxPool: size: (3,1) $->$ Batch Normalization $->$ Dropout : 0.6

Layer 5: Dense Fully Connected Network $->$ Output Shape : 100

5.5. LSTM

Learning Rate : 0.001, **Epochs** : 40

Layer 1: LSTM 200 Units, dropout: 0.6, recurrent_dropout: 0.6

Layer 2: LSTM 50 Units, dropout: 0.4, recurrent_dropout: 0.4

Layer 3: Dense FCNet $->$ Output shape: 100 $->$ Batch Normalisation $->$ Activation: ReLU $->$ Dropout:0.5

Layer 4: FCNet with 4 outputs $->$ Softmax activation

5.6. CNN-LSTM

Learning Rate : 0.0073, **Epochs** : 85

Layer 1: 25 Conv2D filters kernel : (10,1), stride: (1,1), Activation: ReLU $->$ MaxPool: size: (3,1) $->$ Batch Normalization $->$ Dropout : 0.6

Layer 2: 50 Conv2D filters kernel : (10,1), stride: (1,1), Activation: ReLU $->$ MaxPool: size: (3,1) $->$ Batch Normalization $->$ Dropout : 0.6

Layer 3: 100 Conv2D filters kernel : (10,1), stride: (1,1), Activation: ReLU $->$ MaxPool: size: (3,1) $->$ Batch Normalization $->$ Dropout : 0.6

Layer 4: 200 Conv2D filters kernel : (10,1), stride: (1,1), Activation: ReLU $->$ MaxPool: size: (3,1) $->$ Batch Normalization $->$ Dropout : 0.6

Layer 5: Dense Fully Connected Network $->$ Output Shape : 100

Layer 6: LSTM with 20 Units, dropout : 0.6, recurrent_dropout:0.15, activation: tanh, recurrent_activation: sigmoid.

Layer 7: FCNet with 4 outputs $->$ Softmax activation Total Model Parameters : 351,819,

5.7. GRU

Learning Rate : 0.001, **Epochs** : 50

Layer 1: 25 Conv2D filters, kernel : (10,1), stride: (1,1), Activation: ReLU $->$ MaxPool: size: (3,1) $->$ Batch Normalization $->$ Dropout : 0.5

Layer 2: GRU 10 Units, dropout: 0.5, recurrent_dropout: 0.5, return_sequences=False

Layer 3: Dense layer with 4 outputs $->$ Softmax activation

5.8. CNN-GRU

Learning Rate : 0.001, **Epochs** : 50

Layer 1: 50 Conv2D filters kernel : (10,1), stride: (1,1), Activation: ReLU $->$ MaxPool: size: (3,1) $->$ Batch Normalization $->$ Dropout : 0.5

Layer 2: 100 Conv2D filters kernel : (10,1), stride: (1,1), Activation: ReLU $->$ MaxPool: size: (3,1) $->$ Batch Normalization $->$ Dropout : 0.5

Layer 3: Dense Fully Connected Network $->$ Output Shape : 100

Layer 4: GRU with 32 Units, dropout : 0.5, recurrent_dropout:0.5, return_sequences=True

Layer 5: GRU with 64 Units, dropout : 0.5, recurrent_dropout:0.5,return_sequences=True

Layer 6: GRU with 128 Units, dropout : 0.5, recurrent_dropout:0.5,return_sequences=True

Layer 7: Dense Fully Connected Network $->$ Output Shape : 1024

Layer 8: FCNet with 4 outputs $->$ Softmax activation

6. Performance

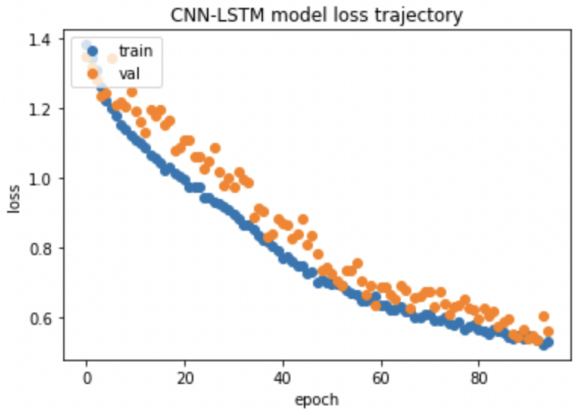

Figure 1. Model loss curve for tuned CNN-LSTM model with training and validation data

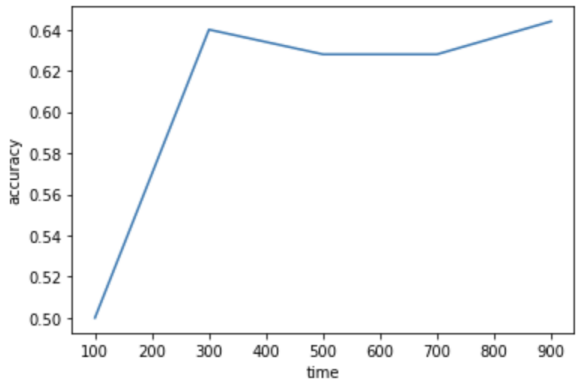

Figure 2. Accuracy v/s time- Change in accuracy not much after 250 to 300 data points

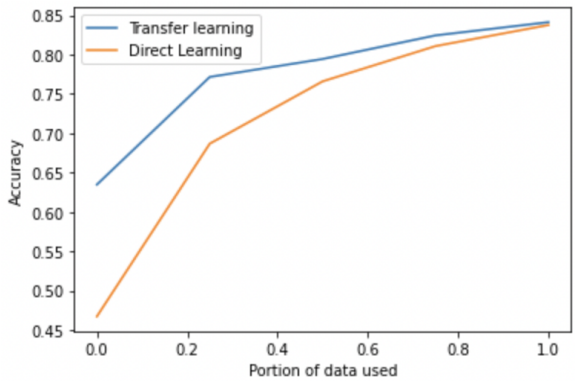

Figure 3. Subject Identification accuracy comparison for Motor Imagery Transfer Learning v/s Direct Learning

Table 1. Average Test accuracy for model trained on all subjects

Subject − >	1	2	3	4	5	6	7	8	9	Overall
Model	Accuracy									
DeepCNN	0.60	0.53	0.75	0.79	0.73	0.66	0.78	0.74	0.74	0.70
LSTM	0.38	0.38	0.62	0.49	0.56	0.54	0.57	0.49	0.61	0.51
CNN-LSTM	0.65	0.60	0.82	0.75	0.82	0.73	0.72	0.71	0.79	0.73
GRU	0.58	0.47	0.63	0.65	0.73	0.64	0.66	0.65	0.72	0.64
CNN-GRU	0.67	0.50	0.61	0.78	0.71	0.66	0.68	0.56	0.73	0.66

Table 2. Test accuracy for model trained on only one subject

Subject − >	1	2	3	4	5	6	7	8	9	Overall
Model	Accuracy									
CNN-LSTM	0.31	0.37	0.51	0.31	0.52	0.40	0.57	0.43	0.44	0.43