Classification on EEG Data with Deep Networks

Sundara Rajan Srinivasavaradhan
UID: 004-760-906
sundar@ucla.edu

Keerthana Sankar
UID: 705-227-735
keerthanasankar@ucla.edu

Raja Mathanky Sankaranarayanan
UID: 705-227-740
mathanky04@ucla.edu

Amruth Varshinee Mohan
UID: 105-225-800
amruth06@ucla.edu

March 18, 2019

Abstract

In this project, we perform classification of motor-imagery tasks using the dataset from the BCI competition [2]. We experiment with a variety of neural network architectures, and describe here the intuitions and ideas we developed through the course of this project.

1. Introduction

Deep neural networks have become ubiquitous in various machine learning tasks, especially those related to computer vision. In the past few years, models based on deep networks have, rather surprisingly, achieved superhuman performance in visual pattern recognition (image classification, for instance). Inspired by these recent developments, researchers have successfully used the leverage provided by these models for big-data analysis to advance other fields - computational neuroscience, AI based gaming engines to name a few. In this project, we apply tools from deep learning for a task in brain-machine interface - the classification of motor imagery tasks from EEG (and potentially EOG) data [2]. In particular, our architectures involve convolutional and recurrent layers and combinations thereof. Finally, we also propose an architecture that combines the *soft-outputs* (namely the class probabilities) of independent convolutional and recurrent models to boost the overall performance; indeed this could be envisioned as a form of ensemble learning meta heuristic since it aggregates the prediction of multiple independent models.

Motivation for considering sophisticated neural network models:

There exist only a few theoretical results on artificial neural network models, yet they work astoundingly well in practice. One of the first theoretical guarantees provided in this context was the *Universal Approximation Theorem* for feed-forward networks, which states that a feed-forward network with single hidden layer with sufficient number of neurons can approximate a wide variety of interesting functions [4]. On the other hand, there seem to be no such guarantees for CNNs and RNNs. This naturally begs the question of why sophisticated NN models (CNN, RNN etc.) are even needed; why not employ just feed-forward layers for the task? To answer this, we constructed a feed-forward network with two fully-connected (FC) hidden layers, each comprising of 512 units and we trained/tested the EEG data on this model - the results are summarized below.

Train accuracy	Val. Accuracy	Test Accuracy
0.93	0.40	0.41

The training accuracy approaches 1 yet the model does not learn well as could be seen from the test accuracy. Since the data comprises of time-series, we used *discrete wavelet transform* (see for example, [3]) to extract essential time-frequency features of the data before classification. The results are summarized below and as we can see, it did much worse than using just the time-domain features[1]. In contrast, as can be

Train accuracy	Val. Accuracy	Test Accuracy
0.95	0.32	0.30

seen in the results on pages 4 and 5, models based on CNNs and RNNs perform much better, justifying their use in this situation.

The rest of report is organized as follows:

- Section 2 briefly describes the various architectures we explored.
- Section 3 discusses our decoding performance on the BCI dataset while also giving some explanations, and answers a few other questions that emerged through the course of this project.
- Section 4 describes potentially exciting future directions.
- Pages 4 and 5 detail the architectures and results.

[1]The code for this part and all other architectures are submitted as a zip file on CCLE.

2. Architectures

The architectures are detailed on page 4. Here we briefly describe them:

- Architecture 1: This architecture mainly consists of (possibly) multiple 1-D CNN layers (with ReLu, Maxpool, Dropout and Batchnorm) followed by a fully connected layer.
- Architecture 2: This architecture mainly consists of (possibly) multiple unidirectional LSTM layers (with ReLu, Dropout and Batchnorm) followed by a fully connected layer.
- Architecture 3: This architecture mainly consists of (possibly) multiple bidirectional LSTM layers (with ReLu, Dropout and Batchnorm) followed by a fully connected layer.
- Architecture 4: This architecture mainly consists of (possibly) multiple 2-D and 1-D CNN layers followed by unidirectional LSTM layers.
- Architecture 5: This architecture is comprised of one architecture 1 model, and one architecture 3 model. During training, these two models are trained independently. During prediction, the class probabilities from the two models are averaged before final classification.

Next we briefly comment on some predominant components present in all of our architectures.

- *Data preprocessing:* We normalize the input features across the batch before feeding it to the model, though this had no observable effect on the performance of the models.
- *Batch size:* We used batch sizes of 50 and 100 for training. Moving to larger batch sizes resulted in a slight degradation in performance.
- *Dropout:* It is well-known that dropout helps in the generalizability of a neural network by acting as a regularizer. Adding a dropout layer helped in improving the accuracy in almost all cases. For instance, as could be seen in the table of results for architecture 4, adding a dropout of 0.3 improved the test accuracy (68.4% vs. 67.5%) on comparable models.
- *Batch normalization:* BatchNorm was used in all of our architectures. We used a BatchNorm layer just prior to the input of a given layer since that modifies the input features of the layer to have a zero-mean, unit-variance distribution.

3. Results and Discussions

Results: More results can be found in page 5 of this report.

Architecture 1: For the CNN architecture, the best test accuracy achieved was 61% when using only the EEG signals (22 electrodes) and 68% when the EOG signals also were also used in training (25 electrodes). We used a stride of 1 for the convolutional layers and stride of 2 for the max-pool layers. Even though low values of stride come at a price of increased computational complexity and the risk of not being general enough, they perform better at capturing the finer details. We also used filter sizes of 3,4 and 5 as larger filter sizes did not help in improving the performance, yet significantly increased the number of trainable parameters.

Architecture 2: For the unidirectional LSTM architecture, the best test accuracy achieved was 60% with the data from 22 electrodes and 73% with 25 electrodes. Some of the best performance numbers we obtained are with architecture 2. We also comment that this may not necessarily mean that architecture 2 is the best for this task as we may not have investigated enough with other architectures (especially with architecture 5).

Architecture 3: For the bi-directional LSTM architecture, the best test accuracy achieved was 57% with the data from 22 electrodes and 71% with 25 electrodes. Note that these numbers are quite close to those obtained with architecture 2.

Architecture 4: For the CNN + RNN based architecture, the best test accuracy achieved was 53% with the data from 22 electrodes and 68% with 25 electrodes.

Architecture 5: For the probability aggregation architecture, we obtained a test accuracy of 55% with the data from 22 electrodes and 68% with 25 electrodes. We comment here that we did not experiment as much as we would have liked with this architecture, and that it has a potential to perform better than the others.

Motivation for considering architecture 5: Although ensemble methods have been known to improve the generalization performance of machine learning models, our motivation for considering this architecture stems from an observation we made during the course of this project. Consider Fig. 1 and Fig. 2 shown below. As can be seen from the confusion matrix heatmaps (darker regions correspond to more datapoints), the CNN based architecture seems to effectively recognize the signals corresponding to two of the classes (legs and tongue) while the RNN based architecture seemingly recognizes the signals corresponding

to the other two classes (left and right hand).

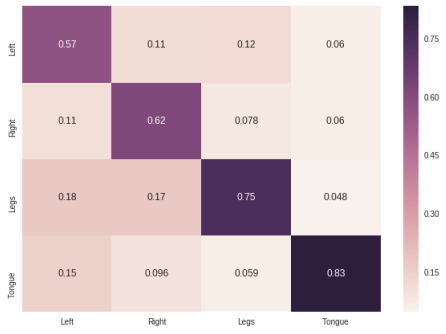


Figure 1: The confusion matrix heatmap for a particular model based on architecture 1 (CNN based).

Figure 2: The confusion matrix heatmap for a particular model based on architecture 3 (Bi-RNN based).

Thus, it would be reasonable to expect that combining the predictions from two independently trained models (one based on arch. 1 and the other based on arch. 3) would aid in the decoding performance. One way to do this is to aggregate the soft-outputs (the class probabilities) from these two models, by taking their average and then decode based on this "average class probability" (indeed, if we had more than two models, we could invoke a majority rule for decoding). This seems to improve over both the underlying models as seen from 'Architecture 5 results' table on page 5.

Is the information encoded in EEG generalizable to unseen subjects?: Suppose that we train a classification model with the data from a few subjects, can we still use this model to predict from the data of an unseen subject? To answer this, we do a small experiment: using the EEG data (22 electrodes) from only 7 subjects, we train a simple model based on architecture 1. But in the testing phase, we only use the data from the 8th subject to make predictions, the results are shown in the following table.

The prediction on the data from the unseen subject has an accuracy significantly greater than chance accuracy, but is also much less than the validation accuracy. These results imply that there could be some correlation in the way information is encoded across different subjects.

Train accuracy	Val. Accuracy	Test Accuracy
0.91	0.58	0.46

4. Future Directions

We here describe potential directions we intended to try, but could not due to time-constraints.

- The first idea hopes to build upon the impressive empirical performance of CNNs on image related tasks. Since CNNs seem to effectively extract essential visual patterns in datasets, it might help to transform the temporal component of our data into a "visual" object: one way to accomplish this is via topographical constructs called *Hilbert space-filling curves*, which map a 1-dimensional space onto a 2-dimensional space while preserving locality (i.e., points close to each other in the 1-D space are also close to each other in the 2-D space with this mapping)[2]. These are hence extensively used to visualize long time-series data effectively, and it would be interesting to check if such a transformation helps in our case.
- Building upon our probability aggregation architecture and our observations on the confusion-matrix, it would be worthwhile to see if constructing a number of CNN based and RNN based models and invoking a majority rule for prediction helps to improve the decoding performance.

References

[1] C. V. S. Baback Moghaddam, Kenneth J. Hintz. Space-filling curves for image compression, 1991.

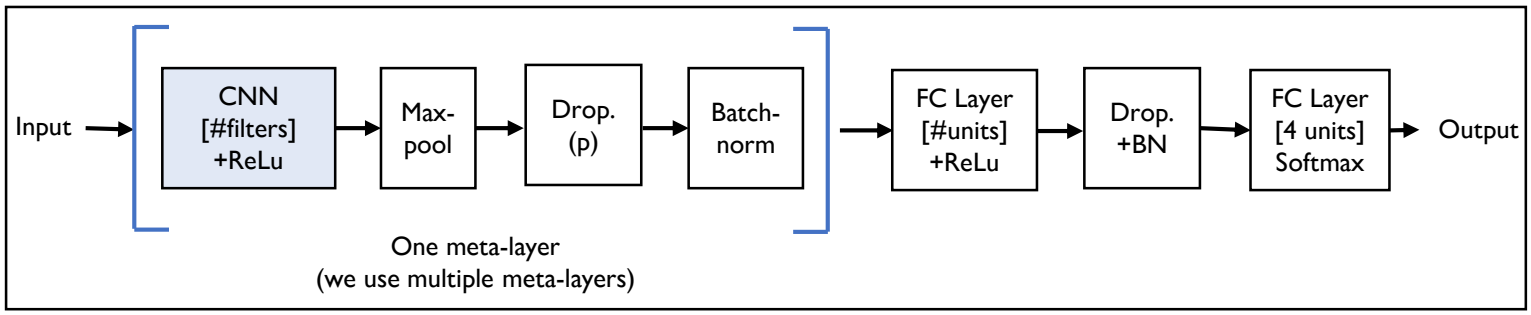
[2] C. Brunner, R. Leeb, G. Muller-Putz, A. Schlogl, and G. Pfurtscheller. Bci competition 2008 - graz data set a, 2008. bbci.de.

[3] K. H. Ghazali, M. F. Mansor, M. M. Mustafa, and A. Hussain. Feature extraction technique using discrete wavelet transform for image classification. In *2007 5th Student Conference on Research and Development*, pages 1–4, Dec 2007.

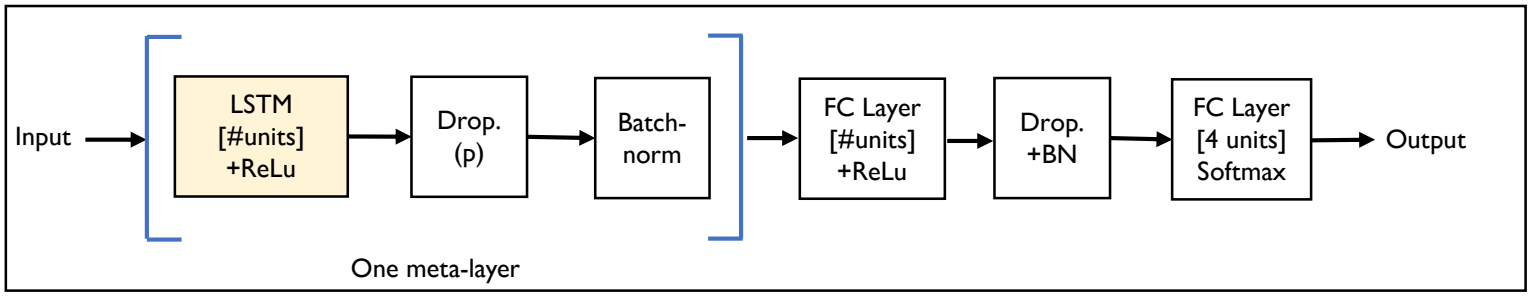
[4] K. Hornik, M. Stinchcombe, and H. White. Multilayer feedforward networks are universal approximators. *Neural Networks*, 2(5):359 – 366, 1989.

[2]There have been works which use Hilbert curves for image processing, see [1] for instance.

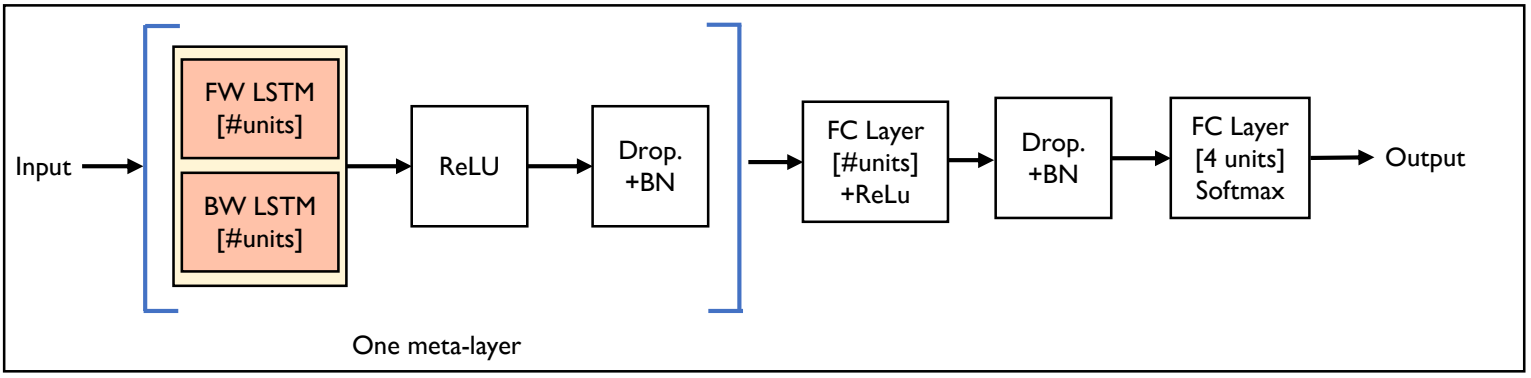
Architecture 1: CNN based architecture



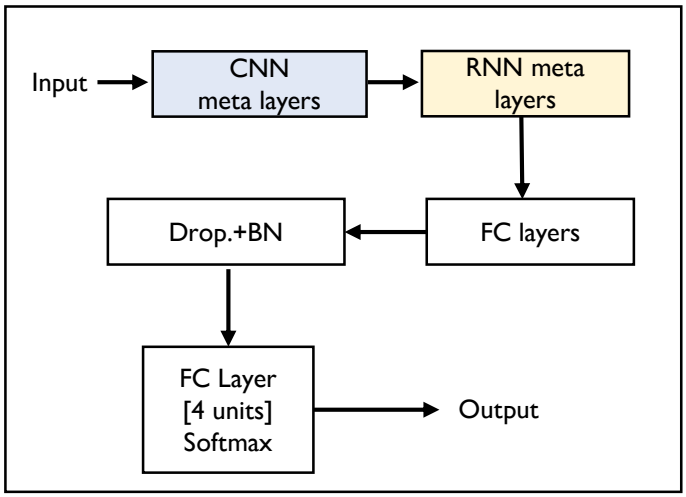
Architecture 2: RNN based architecture



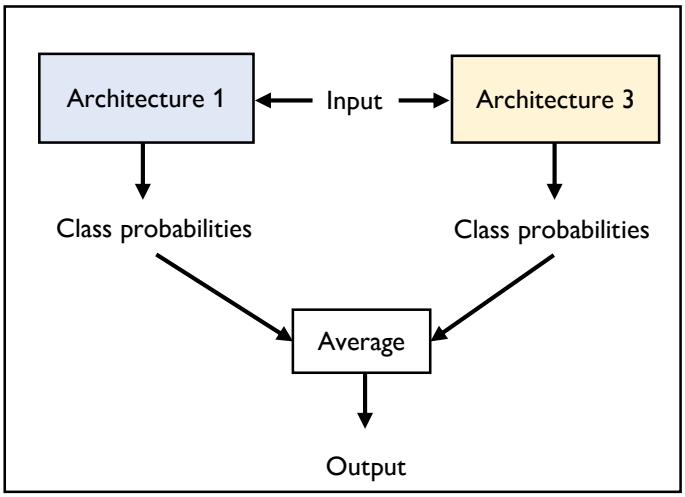
Architecture 3: Bidirectional RNN based architecture



Architecture 4:
CNN + RNN layers



Architecture 5:
Probability aggregation



Architecture 1 results

Electrodes (22 for EEG, 25 for EEG+EOG)	Number of filters in each layer	Number of units FC Layer	Dropout	Train accuracy	Validation accuracy	Test accuracy
22	[64,128,32]	[32,4]	0.5	0.99	0.66	0.61
22	[64,128,32]	[32,4]	0.3	0.85	0.58	0.59
22	[32,128,64]	[32,4]	0.3	0.80	0.54	0.56
25	[64,64]	[32,4]	0.2	0.95	0.65	0.64
25	[64,128,32]	[32,4]	0.3	0.89	0.78	0.68
25	[64,128,128,64]	[32,4]	0.3	0.82	0.68	0.68

Architecture 2 results

Electrodes (22 for EEG, 25 for EEG+EOG)	#Units in each unidirectional LSTM Layer	#Units in each FC Layer	Dropout	Train accuracy	Val accuracy	Test accuracy
22	[128,128]	[4]	0.3	0.78	0.5	0.58
22	[128,128]	[64,4]	0.3	0.96	0.58	0.58
22	[128,128]	[4]	0.1	0.87	0.58	0.6
22	[128,128]	[4]	0.4	0.99	0.59	0.6
25	[128,128]	[4]	0.5	0.98	0.70	0.72
25	[128,128]	[64,4]	0.5	0.92	0.7	0.73

Architecture 3 results

Electrodes (22 for EEG, 25 for EEG+EOG)	#Units in each bidirectional LSTM Layer [Fw1,Bw1,Fw2,Bw2,…]	#Units in each FC Layer	Dropout	Train accuracy	Val accuracy	Test accuracy
22	[128,128]	[512,4]	0.6	0.88	0.55	0.57
22	[64,64,64,64]	[4]	0.5	0.88	0.54	0.53
25	[128,128,128,128]	[64,4]	0.5	0.9	0.7	0.7
25	[32,32,32,32]	[4]	0.4	0.91	0.69	0.7
25	[64,64,64,64]	[64,4]	0.5	0.89	0.72	0.71

Architecture 4 results

Electrodes (22 for EEG, 25 for EEG+EOG)	# Filters in each 2D-CNN layer	# Filters in each 1D CNN layer	#Unites in each LSTM layer	Dropout	Train accuracy	Validation accuracy	Test accuracy
22	[32, 64,128, 64, 32,16]	[64]	[128,128]	0.5	0.74	0.52	0.53
22	[64,128,128]	[0]	[128]	0.3	0.81	0.54	0.53
25	[32, 64,128, 64, 32,16]	[64]	[128]	0.5	0.83	0.73	0.68
25	[32, 64,128, 64, 32,16]	[64]	[128]	0.3	0.95	0.78	0.68
25	[32, 64,128, 64, 32,16]	[64]	[128]	0.0	0.95	0.79	0.67

Architecture 5 results

Electrodes (22 for EEG, 25 for EEG+EOG)	Corresponding accuracy on architecture 1	Corresponding accuracy on architecture 3	Corresponding accuracy on architecture 5
22	0.53	0.52	0.55
25	0.65	0.67	0.68