Analysis of Convolutional Neural Networks, Long Short Term Memory Networks and Variational Autoencoders in classifying movement-related information from EEG signals

Konark J S Kumar
UCLA
konarkjs@ucla.edu

Rahul Shenoy
UCLA
rahulshenoy@ucla.edu

Shwetha Srinath
UCLA
shwethas@ucla.edu

Abstract

We explore the possibilities of using deep learning techniques including Convolutional Neural Networks(CNN), Recurrent Neural Networks(RNN) such as Long Short Term Memory (LSTM), and briefly discuss the performance of Variational Autoencoders(VAE) and Generative Adversarial Networks (GAN) in the classification of movement related thoughts such as the imagination of movement of the left hand(class 1), right hand(class 2), both feet(class 3), and tongue(class 4). Architectures with Multiple layers of CNN are trained and optimized on varying data to correctly predict the task and accuracy is reported for each approach. We also considered using LSTM since the data provided consisted of observations at regular time-intervals. We use techniques of regularization such as batch normalization and dropout to improve the generalization accuracy of our models, and observe that deep CNN with LSTM does a very good job. VAE is used in the network to analyze their impact in data augmentation when the available training data is limited.

1. Introduction

It is seen that, CNNs give good performance on EEG data but the best accuracy we achieved suggests that it is unable to capture the temporal features that characterize the signal. This serves as a motivation to explore the use of LSTM, a unit of recurrent neural network as they are known to be well suited to classify, analyze and predict time series data. VAEs and GANs are used with the objective of augmenting the dataset and the encoding process of VAE might help extract useful features from raw data which could thereby help classification accuracy and combining VAE/GAN with CNN or CNN and LSTMs can help improve inter subject testing.

Section 2 provides a brief overview of the the dataset used in the project. In Section 3, we outline the network structures in detail and discuss the different techniques we used to achieve the best results. In Section 4, we present the results we obtained by experimenting on each method presented in Section 3. Finally, we conclude by discussing how the models can be further improved in Section 5.

2. Dataset

The dataset used is the BCI competition data (2008) [1]. We utilise data from 22 EEG electrodes and 2115 trials across 9 subjects to train, as well as individual subject data to train on one subject at a time. There are 4 possible class labels corresponding to movement of left hand, right hand, legs and tongue.

3. Network Architectures

3.1. CNN (Shallow Net)

In the shallow net CNN architecture we make use of three convolution layers. The first two CNN layers perform a temporal and spatial convolution. This architecture is inspired from [2] Addition of dropout and batch-norm layers were observed to improve the efficiency of the architecture. Xavier initialization was the preferred method of initialization. Optimum results were observed with a learning rate of 5e-4, and RMSProp optimizer was used. The architecture is shown in Figure 1.

The CNN model was trained and tested on different sets of data. Initially, it was trained on a single subjects data and tested on trials from the same subject. This procedure was repeated for all subjects. In the following step, the model was trained on inter subject data and tested on inter subject data.

3.2. Deep CNN

The architecture for the deep convolutional neural network for EEG signal classification is inspired from [2]. The major difference is that we have included a fully connected layer as the final layer as shown in Figure 1 in the appendix.

Layer	in_channels	out_channels	kernel_size	stride
Conv2d time	1	40	(25,1)	1
Conv2d spatial	40	40	(1,22)	1
BatchNorm2d layer				
Non-linear layer (Square)				
Mean Pool	40	40	(75,1)	(15,1)
Non-linear layer (Safe log)				
Dropout layer (p = 0.5)				
Conv2d	40	4	(61,1)	(1,1)
Softmax layer (Log Softmax)				

Figure 1. Shallow CNN

LSTM layer (2 layers, 20 units each)		
Fully Connected Layer	20000	4

Figure 2. LSTM

Layer	**in_channels**	**out_channels**	
FC Linear Layer	22000	400	ENCODER
FC Linear Layer	400	20	
FC Linear Layer	400	20	
FC Linear Layer	20	400	DECODER
FC Linear Layer	400	22000	

Figure 3. VAE

3.3. LSTM

In the LSTM architecture, we pass the input through two LSTM layers and then a fully connected layer. Each layer has 20 units. Optimization using AdamOptimizer and RMSPropOptimizer was done, with regularization using L2 loss.

3.4. Deep CNN + LSTM

We then tried to use a CNN to preprocess the data before sending it as input to the LSTM. The architecture is described below.

3.5. VAE

We used VAE to achieve data augmentation. The architecture of VAE is shown in Figure 3. We perform the encoding and decoding using this architecture. We encode a specific subjects data and generate more samples using the decoder. This data is then concatenated with available data for a subject and passed to the Shallow net CNN for classification. Due to time constraints, we were not able to perform classification on inter-subject data and optimize the results further.

Subject	Testing Accuracy
0	70%
1	50%
2	82%
3	62%
4	66%
5	65.3%
6	86%
7	80%
8	70.2%
Inter-subject Data	**66%**

Figure 4. Shallow CNN subject-wise results

3.6. GAN

GAN architecture with the discriminator and generator network is shown in Figures 4 and 5 of the appendix. The GAN is trained separately on the training data of all 4 labels to generate more training data in an attempt to increase the accuracy by generating artificial data to be fed to the deep CNN. The training of the network is extremely slow and thus was trained for limited iterations. A GAN with discriminator and generator models consisting of fully connected neural networks was also attempted but gave worse results when used alongside the experimental data.

4. Results and Discussion

4.1. Shallow CNN

The results for classification using Shallow net CNN are shown in Figure 4. The testing accuracy is reported for each of the 9 subjects. This is the result of the experiment in which the network was trained on data only from a particular subject and tested on data from the same subject.

In the next experiment we train the model on data from all subjects and test the model with data from every subject. The best model produces an accuracy of 66% and is a more generalized model compared to models that train just on a single subjects data.

4.2. Deep CNN

A systematic approach towards the choice of architecture, time bin length, down sampling ratio etc. is explained in Figure 2 in the appendix. The second column of the figure shows the test accuracy subject wise when the CNN is trained on just the EEG data of subject 6, while column 3 shows the test accuracy subject wise when the CNN is trained on all subjects. The test accuracy for subject 6 is the only one higher than that when trained on all subjects. This could imply that the inter-subject EEG data are not highly

Test Accuracy:	Trained on all subjects
Subject 0	52
Subject 1	50
Subject 2	73
Subject 3	64
Subject 4	56.38
Subject 5	45.91
Subject 6	65
Subject 7	67
Subject 8	77.66
All Subjects	61.06

Figure 5. Deep CNN + LSTM

Architecture	Training Accuracy	Testing Accuracy
VAE + CNN	48%	36%

Figure 6. VAE results

correlated and thus it is not suitable to use this model trained on one subject to test other subjects. Similar results were obtained for models trained on other individual subjects.

4.3. LSTM

Results obtained using just LSTM followed by a fully connected layer were not promising, since the model tended to overfit on the training data, and only managed to reach approximately 40% validation accuracy. Perhaps including data preprocessing before feeding it to the LSTM would improve results.

4.4. Deep CNN + LSTM

The model trained on all subject datasets and the figure shows subject-wise accuracies. The network performs on-par with deep CNN and is much better than plain LSTM.

4.5. VAE

The results obtained by using VAE for data augmentation and then classifying using Shallow net CNN is shown in Figure 6. In this experiment we werent able to run the CNN classifier for longer epochs due to time constraint and the values reported are when the model was trained for 30 epochs. We can see that this could be a good approach to follow when the available data for a test subject is very less.

4.6. GAN

1400 additional data points were added to the existing training dataset using GAN. The model was trained to one fifth the number of iterations as deep CNN due to limited

Architecture	Validation Accuracy	Testing Accuracy
GAN + CNN	56%	57%

Figure 7. GAN results

time, but was able to achieve an accuracy of 57% on test dataset already.

5. Conclusion

We believe that the models can further be improved by using some data pre-processing techniques and if we optimized the combination of VAE/GAN with CNN or CNN and LSTM we could achieve better results.

References

[1] C. Brunner, R. Leeb, G. Müller-Putz, A. Schlögl, and G. Pfurtscheller. Bci competition 2008–graz data set a. *Institute for Knowledge Discovery (Laboratory of Brain-Computer Interfaces), Graz University of Technology*, 16, 2008. 1

[2] R. Schirrmeister, J. Springenberg, L. Fiederer, M. Glasstetter, K. Eggensperger, M. Tangermann, F. Hutter, W. Burgard, and T. Ball. Deep learning with convolutional neural networks for brain mapping and decoding of movement-related information from the human eeg. arxiv, 2017. *arXiv preprint arXiv:1703.05051*. 1

Appendix

Layer	in-channels	out-channels	kernel_size	stride
Conv2d time	1	25	(3,1)	1
Conv2d spatial	25	25	(1,22)	1
Batch Normalization layer				
Activation layer (ELU)				
Max Pool	25	25	(3,1)	3
Dropout layer (p=0.7)				
Conv2d	25	50	(3,1)	1
Batch Normalization layer				
Activation layer (ELU)				
Max Pool	50	50	(3,1)	3
Dropout layer (p=0.7)				
Conv2d	50	100	(3,1)	1
Batch Normalization layer				
Activation layer (ELU)				
Max Pool	100	100	(3,1)	3
Dropout layer (p=0.7)				
Conv2d	100	200	(3,1)	1
Batch Normalization layer				
Activation layer (ELU)				
Max Pool	200	200	(3,1)	3
Dropout layer (p=0.5)				
Flatten				
Linear layer (output size = 4)				
Softmax layer				

Figure 1: Deep CNN architecture

Strategies		Testing Accuracy
Architecture Choice	Deep CNN without batch normalization and dropout layer and no L2 regularization	50.3
	Deep CNN without batch normalization but with dropout layer and L2 regularization	54.4
	Deep CNN with batch normalization and dropout layer and also L2 regularization	**56.4**
Time bin length	500	54.85
	600	57.33
	700	**58.24**
	800	56.2
	900	56.4
	1000	57.79
Downsampling ratio	1	**58.69**
	0.5	55.08
	0.33	50.79
Downsample and concatenate	Downsample ratio = 0.5	**59.6**
Cropped Training stride size	10	**64.67**
	25	63.88
	50	62.64
	75	60.15
	100	57.56

Figure 2: Strategies and results deep CNN

Layer	in-channels	out-channels	kernel_size	stride
Conv2d time	1	25	(3,1)	1
Conv2d spatial	25	25	(1,22)	1
Batch Normalization layer				
Activation layer (ELU)				
Max Pool	25	25	(3,1)	3
Dropout layer (p=0.7)				
Conv2d	25	50	(3,1)	1
Batch Normalization layer				
Activation layer (ELU)				
Max Pool	50	50	(3,1)	3
Dropout layer (p=0.7)				
Conv2d	50	100	(3,1)	1
Batch Normalization layer				
Activation layer (ELU)				
Max Pool	100	100	(3,1)	3
Dropout layer (p=0.7)				
Conv2d	100	200	(3,1)	1
Batch Normalization layer				
Activation layer (ELU)				
Max Pool	200	200	(3,1)	3
Dropout layer (p=0.5)				
LSTM (1 layer, 64 units)				
Linear layer (output size = 4)				
Softmax layer				

Figure 3: Deep CNN + LSTM architecture

Layer	in channels	out channels	Kernel_size	stride
Conv2d_T	100	1024	(3,3)	(1,1)
Batch Normalization layer				
Leaky Relu				
Conv2d_T	1024	512	(3,3)	(2,3)
Batch Normalization layer				
Leaky Relu				
Conv2d_T	512	256	(3,3)	(2,3)
Batch Normalization layer				
Leaky Relu				
Conv2d_T	256	128	(3,3)	(2,3)
Batch Normalization layer				
Leaky Relu				
Conv2d_T	128	64	(3,3)	(1,3)
Batch Normalization layer				
Leaky Relu				
Conv2d	64	1	(3,3)	(1,3)
Tanh Activation				

Figure 4: Generator architecture

Layer	in channels	out channels	Kernel_size	stride
Conv2d	1	64	(3,3)	(1,3)
Leaky Relu				
Conv2d	64	128	(3,3)	(1,3)
Batch Normalization layer				
Leaky Relu				
Conv2d	128	256	(3,3)	(2,3)
Batch Normalization layer				
Leaky Relu				
Conv2d	256	512	(3,3)	(2,3)
Batch Normalization layer				
Leaky Relu				
Conv2d	512	1024	(3,3)	(2,3)
Batch Normalization layer				
Leaky Relu				
Conv2d	1024	1	(3,3)	(1,1)
Sigmoid Activation				

Figure 5: Discriminator architecture

	Trained on:									
Test Accuracy on (down):	Subject 0	Subject 1	Subject 2	Subject 3	Subject 4	Subject 5	Subject 6	Subject 7	Subject 8	All Subjects
Subject 0	**71**	30	63	22	27	27	35	35	43	56
Subject 1	41	35	39	28	30	22	24	27	28	**46**
Subject 2	61	48	**80**	31	29	36	28	47	47	75
Subject 3	38	24	34	53	49	29	62	27	19	**67**
Subject 4	29.79	34.04	31.91	41.49	64.89	39.36	60.64	38.3	39.36	**65.96**
Subject 5	21.43	30.61	34.69	23.47	22.45	47.96	28.57	32.65	30.61	**50**
Subject 6	43	29	37	35	55	27	**78**	23	31	73
Subject 7	42	24	53	41	35	31	46	70	49	**71**
Subject 8	30.85	39.36	44.68	32.98	29.79	31.91	37.23	61.7	**85.11**	76.6
All Subjects	42.66	34.31	44.58	33.75	38.15	32.28	44.58	40.18	40.63	**64.45**

Figure 6: Deep CNN testing accuracies across all subjects