Task Classification of EEG Data with Neural Networks

Trung Le 205223834
Hazar Benan Unal 605229363
Hsien-Chih Hung 104466136
Alimzhan Sultangazin 005035952

Abstract

In brain-computer interfacing, there has been an increased interest in the problem of end-to-end electroencephalogram (EEG) signal decoding. In this project, we have studied deep neural networks with a range of different architectures for the purpose of decoding four imagined tasks from raw EEG taken with 22 electrodes. We have mainly focused on architectures, which are based on convolutional neural networks (CNN), recurrent neural networks (RNN), and combinations of both. First, two purely CNN architectures with different network depths were considered. Next, performance of several architectures based on LSTM and GRU was studied. The experiments have shown that the best testing accuracy of 62.8% is achieved by using Deep ConvNet. This is unexpected because RNNs are known to handle temporal series, like EEG, better than CNNs. Moreover, we have discussed some insights as to how special architecture design choices may help the classification task.

1. Introduction

In recent years, electroencephalography (EEG) has attracted researchers as a highly desirable pathway for brain-computer interfacing (BCI) due to its non-invasive nature and low cost. Motivated by this, we have studied the dataset, which consists of several 4-second trials wherein a subject imagines to perform one of four physical actions. Here, we have decided to view this data in two mutually exclusive ways:

1. treat each trial as an image and try to extract the features for each task;
2. treat each trial as temporal data over each electrode, exploiting the temporal information to do classification.

These views encourage us to use convolutional neural networks (CNNs) and recurrent neural networks (RNNs), respectively.

The use of CNN-based architectures as a solution to the BCI problem is well-studied. A good example of study about CNNs in EEG signal processing is presented in [2]. In this paper, the authors used a dataset similar to the one we are using. To validate efficacy of this approach, we started with two CNN models described in this work, namely ShallowConvNet and DeepConvNet. These two models are inspired by architectures in computer vision research, and incorporated two convolution layers to learn both temporal and spatial features in EEG data.

Next, being motivated to see how RNN can exploit temporal features inherent to EEG signal, we explored models built with LSTM and GRU, and implemented variations of GRU-based architectures from [1]. Roy et al. in [1] came up with ChronoNet, an architecture consisting of multiple 1-D convolution layers of varying sizes, in hope that these layers learn features across multiple timescales and, thus, are able to adapt to the unique temporal features of a different dataset. Learning from the success of residual blocks, ChronoNet also solved the vanishing gradient problem by adding skipped connections to create a gradient highway between layers. Finally, we put DeepConvNet and ChronoNet in sequence, hypothesizing that ChronoNet may help further extract sophisticated features obtained from DeepConvNet and improve the overall accuracy.

To observe the effects of different scenarios, we:

- trained the models for only 1 subject;
- trained the models for all subjects;
- used cropped and uncropped training set;
- plotted the accuracy as a function of trial time.

The details of the architectures can be found in supplementary materials at the end of this report.

2. Results

The testing accuracy for the models we trained can be seen in Table 1. Testing accuracies as a function of time and confusion matrix for our best performance can also be found in Figures 1, 2, 3, and 4. Note that only half of the total duration is plotted for DeepConvNet accuracy as a function of time due to long running time it needs.

According to these results, when the network is trained with 1 subject only, we get the best performance with DeepConvNet (60.93% accuracy). For the networks trained with all subjects, we get 62.75% accuracy as the best accuracy with DeepConvNet. Overall, we observe that DeepConvNet is the most reasonable choice for task classification from EEG data.

3. Discussion

The given dataset is small in size and, therefore, we augmented the data by cropping each trial along time axis, where each crop contains 250 time points and is strided by 10 time points. Although we use smaller temporal data in cropped version, we observe that using 15 times larger training dataset from cropping greatly improved accuracy. This shows that only a small portion of all temporal information is sufficient to train the network. This result is also in line with plots that show accuracies as a function of time: The accuracy does not change much after 1/4 of the temporal data. Secondly, we observe substantial improvement for shallow and deep convnets when the network is trained and tested on all subjects. However, this is untrue for other networks. We can hypothesize that these networks are too simple to deal with a large amount of data. We keep them as they are not to make the running time too long.

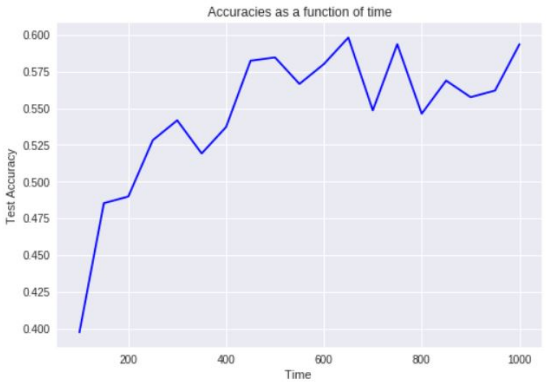

Figure 1. Accuracy as a function of time for shallow ConvNet

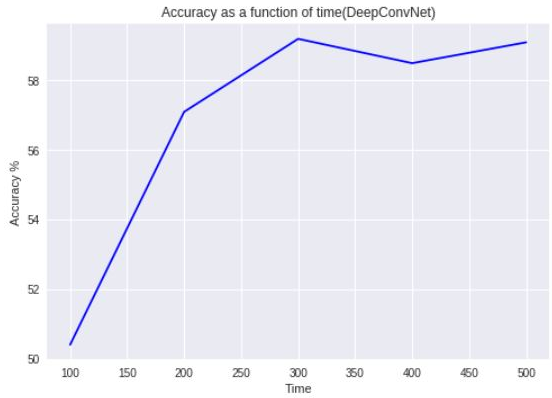

Figure 2. Accuracy as a function of time for deep ConvNet

We see that CNN architectures perform better than RNN architectures. This goes against our initial hypothesis that EEG as a time series signal should be better handled by RNN as RNN has memory of the state. The reason could be that for EEG, the spatial features might be much more informative. EEG is well-known to have low spatial resolution, thus it typically needs data from multiple electrode channels to fully represent the cognitive process. In our RNN architectures, even in ChronoNet, we performed convolution over the time series only, while for CNN architectures we also performed convolution across electrode channels.

Among RNN and CNN-RNN hybrid architectures, we saw no significant difference in testing accuracy, but more sophisticated models (from CRNN up to DeepConvChronoNet) seem to outperform simple models (LSTM, DGRU). This is expected, as complex models have higher capacity to extract meaningful features from the dataset.

Besides the overall accuracy, the confusion matrix can also be informative to design a better network or to use DeepConvNet only for more reliable predictions. In our confusion matrix, which is generated from our best prediction 62%, we see that the network is somewhat biased for class 1. Therefore, it might be useful to take this into account when evaluating the reliability of task classifications.

Since, even after data augmentation, we use a relatively small dataset, we sometimes encounter overfitting during the training process. Using a larger dataset may result in better performances for all the models if the architectures are optimized accordingly. In addition, some other observations about the EEG data may lead to improvements for these models. For example, if a periodic behavior is ob-

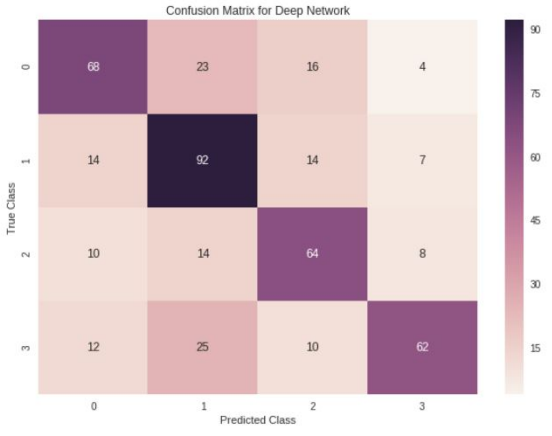

Figure 3. Confusion Matrix for the predictions with the best performance,i.e. DeepCNN 62%

served in electrode recordings, RNN architectures with better performance can be designed.

Design choices for experimented architectures and their effects on the results are discussed below:

3.1. Shallow ConvNet

The shallow convnet consists of 2 convolutional layers. The filters for these convolutional layers are chosen to be horizontal and vertical, i.e. (1,25) and (22,1) respectively. The main purpose of choosing these shapes is to extract features in temporal dimension, as well as features across electrode channels. As can be seen in results, shallow convnet achieves good accuracy, near that of deep convnet. This is consistent with the results in [2], where they achieve 1% lower accuracy for shallow network. We choose 'elu' as activation function (similarly to [2]) since it works better than 'relu' for this particular task. One possible reason for this is that 'relu' removes all the negative values while 'elu' results in small negative gradients, which allow the network to learn features that are only slightly different across electrodes.

3.2. Deep ConvNet

In deep convnet, we simply repeat the shallow network with some adjustments. The main difference from the shallow network, except being deeper, is that filter sizes are longer along the dimension of electrodes. In particular, we convert 2 successive convolutional layers with filter sizes (1,25) and (22,1) into, for example, one convolutional layer with filter size (10,50). This allows the network to encode information between different electrodes simultaneously. One possible advantage of this approach is that the network becomes more robust to noise in the electrode recordings. As a result, we achieve the best performance when we use deep convnet. For the same reason, we choose 'elu' activation in the shallow network. Due to extremely long training time with limited computation capacity, for cropped, all-subject dataset we needed to stop early in the middle of training and reported the testing accuracy there. The final testing accuracy could have increased further if we had not stopped early, as observed by the trend of validation accuracy.

3.3. LSTM

Due to the size of the EGG data set, it is easy for a model to overfit the training data and for the model to become less general. When training LSTM models, it is noted that the performance of the model does not increase as more layers are added to it. In the end, the chosen architecture included one LSTM layer, one dense layer with batch normalization before both. The performance of the model on the testing set is roughly 35% accurate. We used both cropped and uncropped data, as well as sub-sampled data. It turns our uncropped and sub-sampled data outperform the cropped data. When applied to Subject 1 data only, sub-sampled data allow the model to achieve 42% accuracy. For all subjects, network trained on sub-sampled data achieves 35% accuracy, while the one trained on cropped data - below 30%. With increased time data, we see a consistent increase in loss, however, the model accuracy remains roughly 35%.

3.4. DGRU + CRNN + ICRNN + CDRNN + ChronoNet + DeepConvChronoNet

These architectures are mirrored from those discussed in [1]. DGRU is 4 layers of GRU stacked together. We selected 50 units instead of 32 as used in the original paper, since these values gave us the best performance empirically for our dataset. This is equivalent to remembering in the model's hidden states 200ms worth of EEG data (we assumed this interval contains the most meaningful features of EEG). CRNN develops DGRU by adding three 1-D Convolution layers, which the authors in [1] attested will extract temporal data from raw EEG and feed it to the GRU layers. ICRNN further tweaked CRNN by building multiple 1-D Convolution layers of exponentially varying filter length, hoping to adapt different dataset with different inherent temporal characteristics. CDRNN implemented skipped connections between GRU blocks, supposedly solving the vanishing gradient problem. Finally, ChronoNet is built on top of ICRNN and added skipped connections as in CDRNN. Aside from the hyperparameters discussed above, we kept other hyperparameters as presented in the original paper.

Our custom DeepConvChronoNet is built based on the hypothesis that once we cascade ChronoNet after DeepConvNet, ChronoNet could further process the extracted features given by DeepConvNet. However empirically this model perform just well as ChronoNet and worse than DeepConvNet. Perhaps further hyperparameter tuning could help improve the performance of this model.

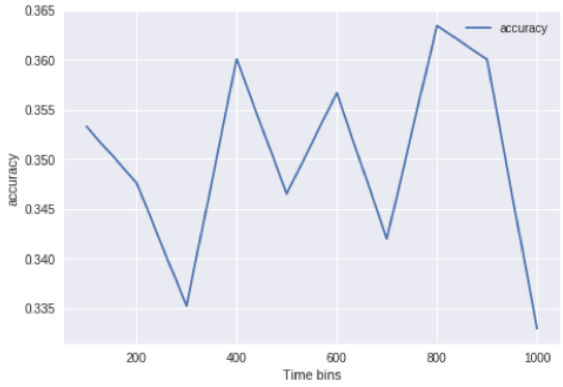

Figure 4. Accuracy with increasing time input LSTM

References

[1] S. Roy, F. I. Kiral-Kornek, and S. Harrer. Chrononet: A deep recurrent neural network for abnormal eeg identification. 01 2018.

[2] R. Schirrmeister, L. Gemein, K. Eggensperger, F. Hutter, and T. Ball. Deep learning with convolutional neural networks for decoding and visualization of eeg pathology. *2017 IEEE Signal Processing in Medicine and Biology Symposium (SPMB)*, 2017.

Architectures	Accuracy (%)			
	Subject 1		All subjects	
	no crop	cropped	no crop	cropped
ShallowConvNet	44.00	60.67	56.88	47.93
DeepConvNet	46.00	60.93	62.75	39.57(early stopped)
DGRU	–	44.4	–	30.4
CRNN	–	49.46	–	32.5
ICRNN	–	52.79	–	34.66
CDRNN	–	53.33	–	32.59
ChronoNet	–	52.53	–	31.99
DeepConvChronoNet	–	52.80	–	32.84

Table 1. Testing accuracies of considered architectures for different datasets

Block	Layer	# filters	size	# params	Output	Activation	Options
1	Conv2D	40	(1,25)	1040	(22,1000,40)	None	
	Conv2D	40	(22,1)	35240	(22,1000,40)	ELU	
	BatchNorm			160	(22, 1000, 40)		
	Lambda (Square)				(22,1000,40)		
	AveragePool2D		(1,75)		(22,62,40)		stride = (1,15)
	Lambda (ReLU)				(22,62,40)		
	Dropout				(22,62,40)		$p = 0.5$
	Flatten				54560		
Classifier	Dense	4			218244	Softmax	

Table 2. ShallowConvNet

Block	Layer	# filters	size	# params	Output	Activation	Options
1	Conv2D	25	(1,10)	275	(22,1000,25)	None	
	Conv2D	25	(22,1)	13775	(22,1000,25)	None	
	BatchNorm			100	(22, 1000, 25)		
	Activation (ELU)				(22, 1000, 25)		
	MaxPool2D		(1,3)		(22,333,25)		stride = (1,3)
	Dropout				(22,333,25)		$p = 0.5$
2	Conv2D	50	(25,10)	312550	(22,333,50)	None	
	BatchNorm			200	(22, 333, 50)		
	Activation (ELU)				(22, 333, 50)		
	MaxPool2D		(1,3)		(22,111,50)		stride = (1,3)
	Dropout				(22,111,50)		$p = 0.5$
3	Conv2D	50	(50,10)	1250050	(22,111,50)	None	
	BatchNorm			200	(22, 111, 50)		
	Activation (ELU)				(22, 111, 50)		
	MaxPool2D		(1,3)		(22,37,50)		stride = (1,3)
	Dropout				(22,37,50)		$p = 0.5$
4	Conv2D	200	(100,10)	10000200	(22,37,200)	None	
	BatchNorm			800	(22, 37, 200)		
	Activation (ELU)				(22, 37, 200)		
	MaxPool2D		(1,3)		(22,12,200)		stride = (1,3)
	Dropout				(22,12,200)		$p = 0.5$
	Flatten				52800		
Classifier	Dense	4			218244	Softmax	

Table 3. DeepConvNet

Block	Layer	# filters	# params	Output	Activation
1	GRU	50	786000	(1000,50)	
2	GRU	50	1503000	(1000,50)	
3	GRU	50	1503000	(1000,50)	
4	GRU	50	1503000	50	
Classifier	Dense	4	2004		Softmax

Table 4. DGRU architecture

Block	Layer	# filters	size	# params	Output	Activation	Options
1	Input				(1000,22)		
	Conv1D	4	32	2820	(500,4)	None	stride = 2
	Conv1D	4	32	516	(250,4)	None	stride = 2
	Conv1D	4	32	516	(125,4)	None	stride = 2
2	GRU	50		759000	(125,50)		
3	GRU	50		1503000	(125,50)		
4	GRU	50		1503000	(125,50)		
5	GRU	50		1503000	50		
Classifier	Dense	4	2004		Softmax		

Table 5. CRNN architecture

Block	Layer	# filters	size	# params	Output	Activation	Options
1	Input				(1000,22)		
	Conv1D	8/4/2	32	5640/2820/1410	(500,8)/(500,4)/(500,2)	None	stride = 2
	Concatenate				(500,14)		
	BatchNorm			56	(500,14)		
2	Conv1D	8/4/2	32	3592/1796/898	(250,8)/(250,4)/(250,2)	None	stride = 2
	Concatenate				(250,14)		
	BatchNorm			56	(250,14)		
3	Conv1D	8/4/2	32	3592/1796/898	(125,8)/(125,4)/(125,2)	None	stride = 2
	Concatenate				(125,14)		
	BatchNorm			56	(125,14)		
4	GRU	50		9900	(125,50)		
5	GRU	50		15300	(125,50)		
6	GRU	50		15300	(125,50)		
7	GRU	50		15300	50		
Classifier	Dense	4	204		Softmax		

Table 6. ICRNN architecture

Block	Layer	# filters	size	# params	Output	Activation	Options	Connected to
1	Input				(1000,22)			
a	Conv1D	4	32	2820	(500,4)	None	stride = 2	Input
b	Conv1D	4	32	516	(250,4)	None	stride = 2	Conv1D (1a)
c	Conv1D	4	32	516	(125,4)	None	stride = 2	Conv1D (1b)
2a	GRU	50		759000	(125,50)			Conv1D (1c)
b	GRU	50		1503000	(125,50)			GRU (2a)
c	Concatenate				(125,100)		GRU (2a, 2b)	
3a	GRU	50		2253000	(125,50)			Concatenate (2c)
b	Concatenate				(125,150)		GRU (2a,2b,3a)	
4a	GRU	50		30030000	50			Concatenate (3b)
Classifier	Dense	4	2004		Softmax			GRU(4a)

Table 7. C-DRNN architecture

Block	Layer	# filters	size	# params	Output	Options	Connected to
1	Input				(1000,22)		
a	Conv1D	8/4/2	32	5640/2820/1410	(500,8)/(500,4)/(500,2)	stride = 2	
b	Concatenate				(500,14)		Conv1D(1a $\times$ 3)
c	BatchNorm			56	(500,14)		
2a	Conv1D	8/4/2	32	3592/1796/898	(250,8)/(250,4)/(250,2)	stride = 2	
b	Concatenate				(250,14)		Conv1D(1a $\times$ 3)
c	BatchNorm			56	(250,14)		
3a	Conv1D	8/4/2	32	3592/1796/898	(125,8)/(125,4)/(125,2)	stride = 2	
b	Concatenate				(125,14)		Conv1D(1a $\times$ 3)
c	BatchNorm			56	(125,14)		
4a	GRU	50		759000	(125,50)		BatchNorm (3c)
b	GRU	50		1503000	(125,50)		GRU (4a)
c	Concatenate				(125,100)		GRU (4a, 4b)
5a	GRU	50		2253000	(125,50)		Concatenate (4c)
b	Concatenate				(125,150)		GRU (4a,4b,5a)
6a	GRU	50		30030000	500		Concatenate (5b)
Classifier	Dense	4	2004	Softmax			

Table 8. ChronoNet architecture