ECE 239 Deep Learning Project

Fangyao Liu	Tianyi Liu	Xuan Hu	Yanzhe Xu
204945018	705035425	505031796	404946757
fangyaoliu@g.ucla.edu	liuty13@ucla.edu	x1310317@gmail.com	davidxu2019@gmail.com

Abstract

In this project, we trained different models to predict people's action according to the signals of their electrodes. First, we start with basic models including CNN, LSTM and AutoEncoder. Second, we explore some methods like shallow CNN, time series analysis and weighted regularization to optimize our model. At last, we summarize performances and analyze results. The best test accuracy we can get is 64.11%

1. Introduction

In this section, we are going to introduce the motivation of using these foundation deep-learning architectures in the project. Convolutional Neural Network and Recurrent Neural Network are used to classify signals. Autoencoder is used to extract useful features from EEG signals.

1.1. Convolutional Neural Netwrok

As we all know, the convolutional neural network is very good at image classification since its convolution and pooling layer would gain the capacity by training to reduce the images into a form which is easier to process without losing features which are critical for getting a good prediction. Those important features can be learned and selected by each filter. It is pretty intuitive that we could represent the EEG as a time series of topographically organized images. Each pixel of the image is a voltage. The whole image is a voltage distributions plot of several electrodes.Also, the other consideration to use CNN is that the pre-processing required in a CNN is much lower as compared to other classification algorithms. While in primitive methods filters are hand-engineered, CNN has the ability to learn these characteristics by itself. Because we barely know how to appropriately pre-process the electroen-cephalography data and we have no biological background knowledge to select useful features from it, CNN seems like a life-saving choice.

1.2. Long Short-Term Memory

Recurrent Neural Network is another classic artificial neural network for supervised learning. Compared to feed-forward network like CNN, it could use internal state to process sequences-like input, which allows it to exhibit temporal dynamic. It has been widely used in NLP, Speech Recognition and other areas. When we first realized that our dataset is signal in a period of time, our intuition told us RNN will be a more appropriate model than CNN.

1.3. Autoencoder

When we explored methods to optimize our LSTM model for classifying EEG signals, we found that many researches mentioned EEG signals were noisy. One of research papers explained that "Severe contamination of EEG activity by eye movements, blinks, muscle, heart and line noise is a serious problem for EEG interpretation and analysis."[1] It implies that our EEG signals are actually a combination of different movements. In order to remove such artifacts, researchers generally adopted Independent Component Analysis (ICA) to multichannel EEG data to remove such artifacts. Unfortunately, we didn't find any convenient package to perform ICA on our three dimensional data. Instead, we realize Autoencoder could also help us extract useful features and decrease the dimension from EEG data. Autoencoder consisted of encoder and decoder. Encoder is responsible for extracting features and decoder is responsible for recovering signals from these features.We try to explore whether AutoEncoder could recover signals from extracted features. If there exists such a Autoencoder , then we could utilize the output of encoder as our input into CNN/RNN architecture.

2. Results

The accuracy of each architecture is shown in table 1. Also, we predict the accuracy of each subject individually. The result is in table 2. Both CNN and RNN architecture can perform very well on subject 8. Besides that, same model performs different on different action. For Shallow CNN, it predicted action 771 very well. For rnn architec-

tures, it can achieve a 69% accuracy on action 772.

In the next section, we described how we refined and evolved our fundamental models with proper reasoning to reach an appropriate result.

3. Discussion

3.1. Convolutional Neural Network

3.1.1 Deep CNN

Initially, we just think of a very successful and simple CNN model called LetNet, which has 5 convolutional and pooling layer. We adjust some filter parameters to fit the EGG data. However, the model does not work very well.No matter how we change the hyperparamters, the model just does not have the ability to learn the generality of the data. It performs pretty well on training data, but the test error seems always high.We can easily see from the result of the test and train accuracy of the deep CNN that the architecture of the model is so complex that the model can easily get overfitted even though we try everything we can like dropout and L2 regularization to prevent that problem. Then we notice that the essential problem here is that we only have extremely small data. Since theres not enough data to feed the complex model, we should think of another idea.

3.1.2 Shallow CNN

Here, we only apply two convolutional layers to time and electrode dimension separately.[2] The filter applied for time dimension can be thought of as a way to detect a trend or type of voltage change among time, on the other hand, the filter for electrode dimension can select useful feature at a specific time point. Then we put features that are abstracted from the convolutional layer into a fully connected network and softmax to get the classification result. Figure 1 on extra page one shows the tendency of accuracy against epochs for Shallow CNN. Whats more, what needs to mention is that theres a huge difference between an image and an EEG which is the x-axes and y-axes of an image has the spatial relationship but for an EEG the time axes and electrodes axes have no spatial relationship in between. That means, for example, the relationship between electrode no.1 and electrode no.2 is not necessarily stronger than what in between electrode no.1 and electrode no.22. The number or the order of the electrode does not represent anything. Therefore we can only apply a filter for each dimension separately. We observe that when we apply a filter across two dimensions the training and testing accuracy would be always below 50%. We cant even overfit the model, which definitely shows that the way we select the feature is totally wrong.

3.2. Long Short-Term Memory

Here we start with 1 layer of vanilla LSTM followed by a ReLU layer as our basic architecture. At first, we only use the last output of LSTM layer, considering this is a classification problem. However, validation accuracy turned out to be like random guess. We realized that one output contains too less information. Therefore, we keep all 1000 output and add a fully connected layer to map data into 4 classes.During the training, we realize we need regularization because training accuracy is almost 90% just after several epoches, so we add dropout layer to prevent overfitting. In such an basic architecture, our model could reach 47% accuracy. Then we think about how to optimize our results.

3.2.1 Data Pre-processing

To start with, we need to process our dataset to make it more understandable to our model. We first check whether the dataset is balanced. Because an unbalanced dataset will bias model and reach a worse average testing accuracy. Fortunately, number of data samples from each class is almost equal. Second is signal scale. Considering signals are measured from 9 different subjects, we assume that the magnitude may differ in different subjects. Besides, when data are being recorded, there must be some noise. If we remove such shared noise and only keep the unique information of each signal, will it help? So we standardize the dataset by removing the mean and scaling to unit variance. After such a naive standardization, our validation and test accuracy increased by 5%, which also confirms my assumption.

3.2.2 Time Series Analysis

Time series analysis is an effective tool to process sequence. Because our signal are time series, we also consider taking advantage of time series analysis to improve accuracy. According to previous study[3],single-moving average will be an useful tool to help smooth the data. We set window length and shifting length as hyperparameters to tune. For the best set of hyperparameters, this technique could help increase the test accuracy to 59%. Another thing worth to mention is that by applying single-moving average, we also expedite the training and testing process because time series analysis decrease the time units from 1000 to a much smaller value.

3.2.3 Bi-directional Recurrent Neural Network

Considered the data points are related to time, a particular data point should be affected by previous points and also have a connection of the following data points. RNN only consider the influence of previous input, but failed to consider the factor of future input. Thus, we try to use BRNN to

solve this problem. BRNN just use two independent RNNs together. The input sequence is fed for one network in normal time order, and in reverse time order for another network. This structure allows the whole network not only have forward information but also backward information about the input at each step. However, the result does not show great improvement in performance.

3.2.4 Weight Regularization

When we divide our testset and get separate test accuracy by subject, we have found that model has different performance on different subject. Several subjects accuracy is clearly lower than others. Since our data points come from 9 different subjects, their electrodes may have different reaction to same action and then some of them are hard to recognize. Hence, we need to do regularization on the model. We choose to apply different weight on different subject to compute our loss. It means the penalty coefficient for subjects who have lower test accuracy are higher than others so that our model are more strict with their data and expected to have better performance. Besides, we also take a look at the actions. Similarly, we have noticed that test accuracy regarding action shows the same phenomenon. Typically, action 2 has the worst accuracy which is around 40% while accuracy of action 4 is above 65%. To avoid this and get better performance, we also apply different weights on different actions. Figure 2 in extra pages shows the result of weight regularization. From the Table 2 and Table 3 in extra pages, it's easy to noticed that this optimization improves the accuracy by about 3%.

3.2.5 Time duration

In this dataset, each sample has 22 electrodes and each electrode has 1000 data points of different time. More data means we can have a better fit for the data, but also means we need more time to process it. Therefore, we evaluate the accuracy as a function of time to find out the minimum requirement of data to achieve a reasonable accuracy.As the experiment shows, when we have less than 10 data points for each electrode, the accuracy is near 28% which means the model just give a random guess and has not learned features of data. However, as we use data from longer periods of time to feed the model, we get better performance than before. The Figure 1 shows the curve of the accuracy against time. And we can have a reasonable accuracy around 60% from at least 100 data points of time.

3.3. Autoencoder

3.3.1 Linear Autoencoder

At first, we try the simplest Autoencoder model. We just use linear model and relu to encode and decode data.We try

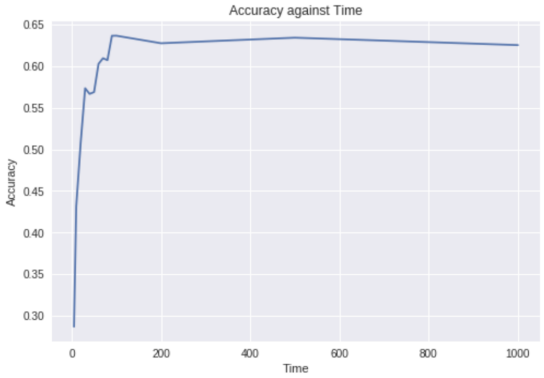

Figure 1. Accuracy vs Time

to recover the original data from the encode data. Then we want to minimize the loss between output and input. However, no matter how we modify the parameter, the error rate is still high and the performance remains unsatisfied.

3.3.2 Denoising Autoencoder

As we introduced above, our EEG data are actually contaminated with other artifacts. Therefore, these artifacts are actually noise. Then, we intuitively came up with this idea to use Denoising Autoencoder to clean noise and extract useful features. Denoising Autoencoders are consisted of encoder and decoder. Encoder part is consisted of several convolution layers, ReLU layers and batchnorm layers. Decoder is almost the same besides it replace convolution layers with deconvolution layers. Then we measure the mean square error of original signal and denoised signal. The error rate stays at same level and fluctuates in 300 epoches. Even when we tried different set of parameters, error rate did not decrease.

References

[1]EEG artifact removal with Blind Source Separation
[2]Schirrmeister R T, Springenberg J T, Fiederer L D J, et al. Deep learning with convolutional neural networks for EEG decoding and visualization[J]. Human brain mapping, 2017, 38(11): 539.
[3]Greaves A S. Classification of EEG with recurrent neural networks[J]. 2014.

Table 1: Performance of Model

Model	Test accuracy
CNN	32.05
Shallow CNN	57.78
RNN	47.18
RNN with Standardization	52.82
RNN with Standardization and Sliding Window	59.14
BRNN	60.04
RNN with weights	64.11

Table 2: Performance of Model for each Person

Model	0	1	2	3	4	5	6	7	8	all
CNN	30.00	38.00	32.00	40.00	29.78	32.65	24.00	24.00	38.29	32.05
Shallow CNN	66.00	44.00	70.00	60.00	57.45	46.94	54.00	56.00	65.95	57.78
BRNN	52.00	46.00	54.00	72.00	68.05	53.06	78.00	60.00	76.60	60.95
RNN with weights	54.00	52.00	60.00	62.00	72.34	59.18	64.00	56.00	74.47	64.11

Table 3: Performance of Model for each Action

Model	769	770	771	772	all
CNN	18.91	43.30	23.95	39.44	32.05
Shallow CNN	55.86	52.76	64.58	59.63	57.78
BRNN	57.66	51.97	66.67	69.72	60.95
RNN with weights	61.26	59.84	67.71	68.81	64.11

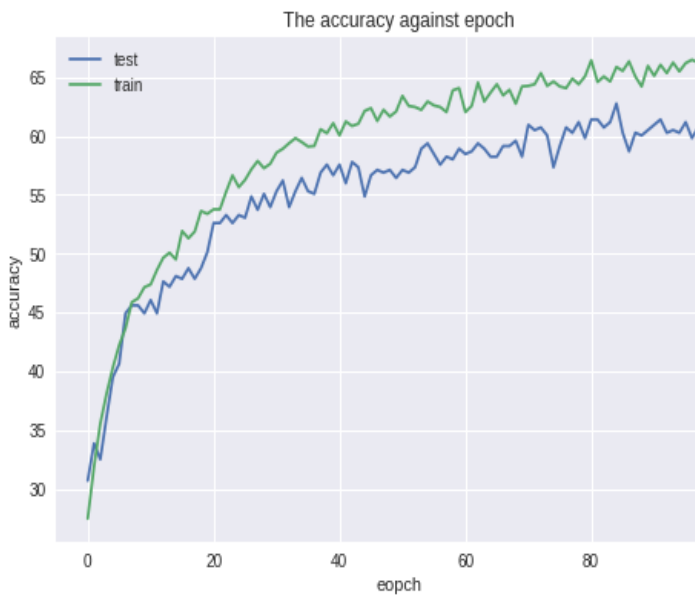

Figure 1: Shallow CNN

Figure 2: BRNN with Weight

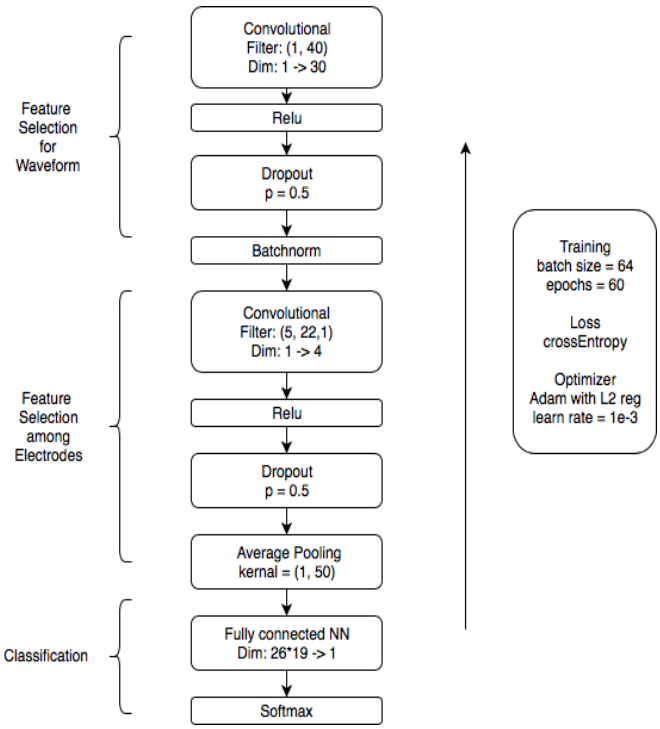

Figure 3: Shallow CNN

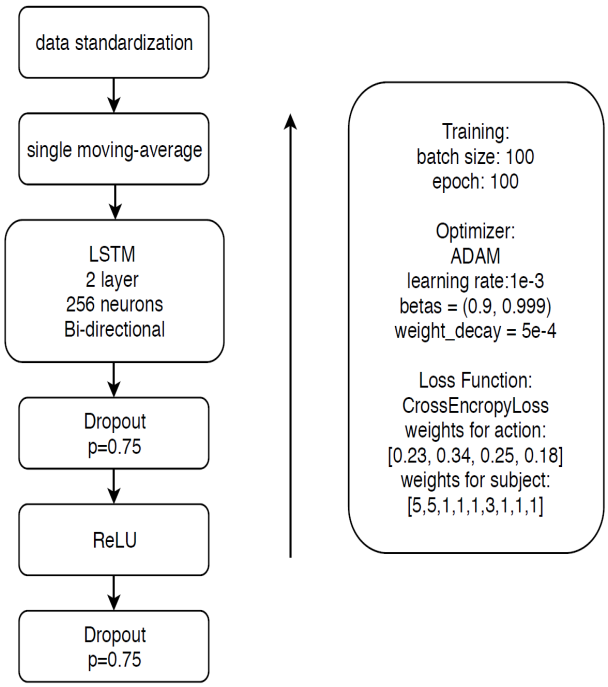

Figure 4: BRNN with Weight

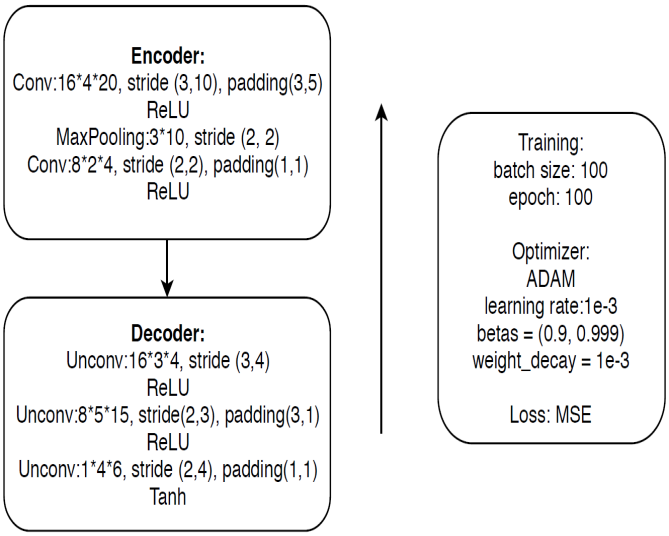

Figure 5: Linear Autoencoder

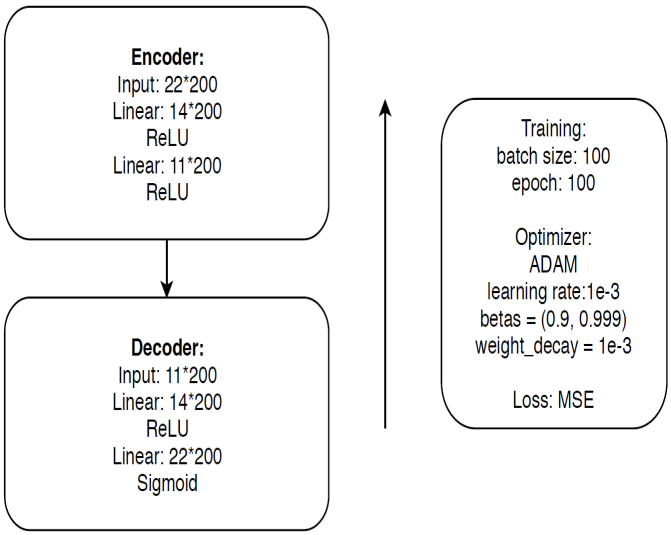

Figure 6: Conv Autoencoder