ECE C247 Course Project Report

Deniz Orkun Eren
UID: 905624625
deren@g.ucla.edu

Arda Okan
UID: 505630082
ardaokan97@g.ucla.edu

Kaan Ozkara
UID: 905431368
kaan@ucla.edu

Mohamad Rida Rammal
UID: 005428723
ridarammal@g.ucla.edu

Abstract

In this project, our goal is to train various neural network based classifiers for classifying EEG data from BCI Competition dataset IV. To this end, we consider 4 different models: CNN's, CNN's with residual connections, CNN's combined with LSTM, and CNN's combined with LSTM's with residual connections. As an added approach, we utilize Gramian Angular Field method to preprocess the data and using the resulting images as input for the CNN combined with LSTM model. We compare the performance of our approaches with and without data augmentation across different objectives and different data lengths. In the end, the architecture that utilizes residual CNNs obtains the best results.

1. Introduction

We consider a dataset consisting of EEG data collected from 9 subjects. For this dataset, each subject was asked to think of one of four tasks, namely moving their left hand, right hand, both feet, or tongue. We are interested in classifying each instance of the EEG data to the task it corresponds to. For this purpose, we propose a variety of neural network architectures, including vanilla Convolutional Neural Networks (CNNs), CNNs with residual connections, and CNNs combined with a recurrent architecture such as Long Short Term Memory (LSTM). We utilize CNNs as they have been demonstrated to be effective in EEG classification, and we add residual connections to our CNN models, which allows us to better train our architectures without having to worry about the vanishing gradient problem that occurs when training deep models. Furthermore, we include recurrent architectures to make better use of the time-series nature of our data. As an additional approach, we consider transforming the time-series data to images using the Gramian Angular Field (GMF) method before feeding the data to a CNN architecture to extract features and perform classification. Moreover, we consider the effect of data augmentation and preprocessing on the classification task for our different models and observe the changes in the performance of our systems.

We test our designed architectures using data from all subjects to see their generalization capabilities. Then, we consider trimming the dataset to different lengths to see the change in performance for each approach. Finally, we attempt to optimize our models to work well with the test data for a single subject to see if the models are suitable when used for classifying the data of a single individual.

1.1. Data Augmentation and Preprocessing

For data augmentation, we consider trimming, max pooling, averaging, and subsampling as possible approaches. For trimming, we discard the data after certain time steps, which allows us to shorten the length of the data we will be working with. For max pooling, we separate the data to sections of equal length and only consider the maximum value within each section. This process allows us to shorten the dataset without losing important information. For averaging, as we did for max pooling we seperate our data to sections. However, this time we consider the average value of the points in each section. We then add Gaussian noise to the results to allow our models to be more robust. Finally, we consider subsampling, where the data is subsampled by a specified rate and Gaussian noise is added.

We utilize a combination of some or all of the aforementioned strategies with different values and observe the change in performance. In the end, we choose different combinations of the preprocessing and augmentation strategies separately for each model based on the effect on performance.

1.2. CNN+LSTM with Gramian Angular Field

Gramian Angular Field(GAF) is a technique proposed by Wang and Oates [1] to transform time-series to images. It first normalizes the time-series to [-1,1], and then converts the data to a polar coordinate system. Afterwards, the temporal correlations of adjacent points are calculated to get a Gram matrix, which is then treated as an image. We decided to utilize this approach as it has been demonstrated to be effective for classifying EEG data [2].

In our implementation, we utilized max pooling, averag-

ing and subsampling to process the dataset. We then normalized each of the 22 features of the data. Subsections of these features were then converted to 30×30 images using the difference GAF technique, which resulted in (5) 30×30 images for each feature. The resulting array of images was treated as a time-series vector with 5 elements. We stacked the images for each feature and provided the result as an input to a CNN for feature extraction. The output of the CNN was connected to an LSTM network, which was used to classify the data after the 5 steps have been processed. More details about our architecture such as the parameters of each layer can be found in the Model Architectures section of our report.

1.3. CNN

EEG is a signal that depends on time but instead of treating it as a time-series data, we can stack each feature vertically and treat the data as an 22×1000 gray-scale image. With this in mind, we can train a CNN on this modified EEG dataset to perform feature extraction and classification. This apprach has been shown to be effective for EEG classification [3]. As we have 22 different channels for the EEG signal, we wanted to extract the features in our data that are present both across time steps and across channels. To do that, we chose filter sizes for the first two convolutional layers to be (1,11) and (22,1). In order to avoid common problems that can be encountered during training such as long training times or overfitting, we used batch normalization and dropout layers within our architecture. For the activation function of the CNN layers, we chose to use ELU as we observed empirically better results with this function compared to other common activations such as ReLU.

1.4. CNN with Residual Connections

When training CNN models with a large number of layers, a commonly encountered issue is the vanishing gradient problem, where the information backpropagated to the initial layers is not enough to perform reliable updates. A solution to this problem is to introduce residual connections between layers to allow the gradients to flow without alteration.

To allow better training of our CNN architecture created for Section 1.3 of our report, we decided to incorporate residual connections in our implementation. Here, the architecture of the model was not significantly altered, however as the residual connections allowed performing better updates on the initial layers, we hypothesized that this will lead to an increase in performance. Furthermore, it has also been shown that utilizing residual connections can improve classification performance on EEG data when compared to using regular CNNs [4], which supports our hypothesis.

There was a slight change in architecture between the model that was trained with augmented data and the model that was trained with regular data. For the one using augmented data, the pool size of the last max pooling layer was changed from (3,1) to (2,1) to accommodate the change of dimension in the input to this layer.

1.5. CNN + LSTM

As our data is time-dependent in nature, we decided that using recurrent architectures can be beneficial to capture this aspect of our data. However, instead of directly feeding the data to such an architecture, we decided to first transform the data to 22×1000 gray-scale images as explained in the CNN section of our report. This image data was fed to a CNN model in order to perform feature extraction. The extracted features were then passed to a bidirectional LSTM network for classification. We decided to include CNNs in our approach since we observed high performance with this model in previous experiments. Furthermore, we decided to use the LSTM as it does not suffer from the vanishing and exploding gradient problem as much as vanilla recurrent networks.

As combining CNNs with the LSTM network has been demonstrated to improve performance when compared to just using CNNs for EEG classification [5], it seemed logical to include this approach in our experiments.

1.6. CNN+LSTM with Residual Connections

As explained in Section 1.5 of the report, we hypothesized that adding an LSTM layer to the output of the CNN could enable us to make better use of the time related information within our dataset. Therefore, we decided to combine the residual CNN model with LSTM to perform classification as an added approach. Since the residual connections allow better training of the CNN part of our model, the extracted features were expected to be better suited for classification. Therefore, using these features in the LSTM was expected to lead to desirable accuracy levels and this approach was expected to give the best performance out of all our approaches.

2. Results

2.1. Performance Across All Subjects

The performance of each approach when trained on data acquired from all subjects and tested on data from all subjects can be found on Table 1. Here, we utilized all of the available data without trimming. However, certain preprocessing techniques were utilized depending on the model to optimize performance.

2.2. Different Data Trimmings

For this experiment, instead of utilizing the entire time-series data, we only included a section of specified length before we performed preprocessing and model training. We

examined the change in performance for each model when only the first 500 and 750 time steps were utilized. The results can be found on Table 2.

2.3. Testing and Training on One Subject

Here, we trained our model on the data obtained from only one subject (subject 1), and tested the performance on the test data of subject 1. For comparison, we then trained our models on all available data and again tested the performance on the test data obtained from subject 1. The results for this experiment can be found on Table 3.

3. Discussion

3.1. Comparison of the Different Approaches

Based on the results shown on Table 1, we can conclude that a residual CNN offers the best performance. Using residual connections allows us to better train our CNN models which allows them to capture better features from the input images compared to regular CNN models. Adding a further LSTM layer led to a slightly lower accuracy. We expected the LSTM layer to capture the time dependent features of the data as it did when there were no residual connections, but it did not seem to do so in this case. One could argue that the adding an LSTM layer to the residual CNN lead to an unnecessarily complex network, which resulted in overfitting and worse performance

Across all models, we observe that performing data augmentation improves classification results. This is expected as using data augmentations allows us to have a training set that has both more data points and more variance, which makes models more robust to changes in the data.

The worst performance was obtained with the GAF CNN+LSTM model. This could be related to the fact that although GAF has been shown to be effective for EEG classification, it may have led to loss of information in our dataset. Moreover, due to memory constraints, the architecture for this approach was kept relatively simple. A more sophisticated approach could lead to better results.

3.2. Comparison Across Trimmings

From Table 2, we can see that the best performance is again obtained using residual CNNs. By examining the results, we can conclude that the models that perform well when all the data is utilized also perform well on trimmed data.

Overall, we can infer that as the length of the data increases, the performance of most models generally also improves. This trend can be explained by the fact that having access to more data allows the models to better capture the patterns in the dataset, which leads to higher performance. Moreover, it should be considered that the used preprocessing and augmentation methods already decrease the length of the data. When these methods are performed on trimmed data, there is a significant loss of information, which leads to decreased accuracies for our models.

3.3. Comparison of Performance on Test Data From Subject 1

From Table 3, we can see that the best performance when trained and tested on the data from subject 1 is obtained when the regular CNN was trained with augmented data. Because complex models tend to overfit on small data, the simpler CNN did best. When trained on all the available data and tested on the test data of subject 1, best performance is obtained when a residual CNN without LSTM is used. However, the difference between this model and the residual CNN with LSTM is negligible.

There appears to be a significant boost in performance for almost all models when training is performed across all subjects instead of just using the data of subject 1. This can be attributed to the fact that in general, using more data when training deep and sophisticated networks is beneficial. Moreover, the patterns and features learned from the data of different subjects can be utilized to predict data from subject 1, as the patterns exhibited in the data do not vary much across subjects. As a final note, one needs to also consider that the networks designed to perform well for the task described in Section 2.1. are used for this experiment. These models are complex have a large number of parameters. Using only the relatively small amount of training data obtained from subject 1 leads to overfitting. This explains the trend that we observe.

The only approach that performs better on the test data of subject 1 when trained only with the train data of subject 1 is the GAF CNN+LSTM. This is related to the fact that for this experiment, we changed the architecture of the model when it was trained only with subject 1 in an attempt to improve performance, so the overfitting issue that affected the other models was not present. The details of this new architecture can be found in the Model Architectures section of our report. Moreover, as discussed in Section 3.1, the model that we use for this approach is relatively simple due to memory constraints. Since the model is simple, it fails to adequately extract features from the other subjects that are applicable to test data of subject 1. However, it is more capable of capturing trends from the training data of subject 1 due to the fact that the training data for this case is limited so a simple model is able to extract useful features.

References

[1] Wang, Z., & Oates, T., (2015). Imaging time-series to improve classification and imputation. In. Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence (IJCAI 2015), 3939- 3945. arXiv:1506.00327

[2] Thanaraj, K. Palani, et al. "Implementation of deep neural networks to classify EEG signals using gramian angular summation field for epilepsy diagnosis." arXiv preprint arXiv:2003.04534 (2020).

[3] Tang, Zhichuan, Chao Li, and Shouqian Sun. "Single-trial EEG classification of motor imagery using deep convolutional neural networks." Optik 130 (2017): 11-18.

[4] Hasan, Md Junayed, et al. "Sleep state classification using power spectral density and residual neural network with multichannel EEG signals." Applied Sciences 10.21 (2020): 7639.

[5] Shahbazi, Mohamad, and Hamid Aghajan. "A generalizable model for seizure prediction based on deep learning using CNN-LSTM architecture." 2018 IEEE Global Conference on Signal and Information Processing (GlobalSIP). IEEE, 2018.

Obtained Results

Model	Test Accuracy in %
GAF CNN+LSTM	44.24
CNN	67.49
CNN+aug	70.88
CNN+LSTM	67.27
CNN+LSTM+aug	72.46
Res CNN	77.36
Res CNN+aug	78.03
Res CNN+LSTM	75.08
Res CNN+LSTM+aug	75.96

Table 1. Test accuracy results when all subjects and time points are used.

Model	500 sec	750 sec
GAF CNN+LSTM	41.0	43.1
CNN	68.17	62.98
CNN+aug	67.27	68.17
CNN+LSTM	64.11	67.72
CNN+LSTM+aug	67.04	69.75
Res CNN	72.91	74.94
Res CNN+aug	73.51	75.36
Res CNN+LSTM	72.23	73.59
Res CNN+LSTM+aug	71.33	70.43

Table 2. Test accuracy results when data is trimmed along time axis to simulate a shorter experiment time.

Model	Trained on Subject 1	Trained on All Data
GAF CNN+LSTM	42.00	38.00
CNN	28.00	60.00
CNN+aug	62.00	66.00
CNN+LSTM	32.00	52.00
CNN+LSTM+aug	46.00	68.00
Res CNN	48.00	78.00
Res CNN+aug	50.00	81.00
Res CNN+LSTM	52.00	74.00
Res CNN+LSTM+aug	50.00	78.00

Table 3. Test accuracy results on subject 1's test set.

Model Architectures

Layer	**Parameters**
Input Layer	InputDim = 5x30x30x22
Conv2D	filters=8, kernelSize = 2x2, activation=ELU, L2 = 0.1
MaxPooling2D	poolSize=2x2
BatchNormalization	
Dropout	dropoutProbability=0.5
Conv2D	filters=10, kernelSize = 2x2, activation=ELU, L2 = 0.1
MaxPooling2D	poolSize=2x2
BatchNormalization	
Dropout	dropoutProbability=0.5
Flatten	
Bidirectional LSTM	units=32, returnSequences=False
Dropout	dropoutProbability=0.5
Dense	units=4, activation = Softmax, L2 = 0.1

Table 4. The Employed Model Architecture for GAF CNN+LSTM Model(Trained with cross-entropy loss and Adam optimizer)

Layer	**Parameters**
Input Layer	InputDim = 5x30x30x22
Conv2D	filters=32, kernelSize = 2x2, activation=ELU, L2 = 0.1
MaxPooling2D	poolSize=2x2
BatchNormalization	
Dropout	dropoutProbability=0.5
Flatten	
Bidirectional LSTM	units=16, returnSequences=False
Dropout	dropoutProbability=0.5
Dense	units=4, activation = Softmax, L2 = 0.1

Table 5. The Employed Model Architecture for GAF CNN+LSTM Model for training on subject 1 data only(Trained with cross-entropy loss and Adam optimizer)

Layer	Parameters
Input Layer	InputDim = 22xTime
Conv2D	filters=25, kernelSize = 1x11
Conv2D	filters=50, kernelSize = 22x1
MaxPooling2D	poolSize=1x3
Dropout	dropoutProbability=0.55
Conv2D	filters=50, kernelSize = 1x10, activation=ELU, L2 = 0.001
Dropout	dropoutProbability=0.55
BatchNormalization	
Conv2D	filters=50, kernelSize = 1x10, activation=ELU, L2 = 0.001
MaxPooling2D	poolSize=1x3
Dropout	dropoutProbability=0.55
Conv2D	filters=100, kernelSize = 1x10, activation=ELU, L2 = 0.001
MaxPooling2D	poolSize=1x3
Dropout	dropoutProbability=0.55
BatchNormalization	
Conv2D	filters=100, kernelSize = 1x10, activation=ELU, L2 = 0.001
Dropout	dropoutProbability=0.55
BatchNormalization	
Conv2D	filters=100, kernelSize = 1x10, activation=ELU, L2 = 0.001
MaxPooling2D	poolSize=1x3
Dropout	dropoutProbability=0.55
Conv2D	filters=200, kernelSize = 1x10, activation=ELU, L2 = 0.001
MaxPooling2D	poolSize=1x2 (1x3 w/o aug)
Dropout	dropoutProbability=0.55
BatchNormalization	
Flatten	
Dense	units=4, activation = Softmax, L2 = 0.001

Table 6. The Employed Model Architecture for CNN Model (Trained with cross-entropy loss and Adam optimizer)

Layer	Parameters
Input Layer	InputDim = 22xTime
Conv2D	filters=25, kernelSize = 1x11
Conv2D	filters=50, kernelSize = 22x1
MaxPooling2D	poolSize=1x3
Dropout	dropoutProbability=0.55
Res_1: BatchNorm	
Conv2D	filters=50, kernelSize = 1x10, activation=ELU, L2 = 0.001
Dropout	dropoutProbability=0.55
BatchNormalization	
Conv2D	filters=50, kernelSize = 1x10, activation=ELU, L2 = 0.001
Add	Res_1
MaxPooling2D	poolSize=1x3
Dropout	dropoutProbability=0.55
Conv2D	filters=100, kernelSize = 1x10, activation=ELU, L2 = 0.001
MaxPooling2D	poolSize=1x3
Dropout	dropoutProbability=0.55
Res_2: BatchNorm	
Conv2D	filters=100, kernelSize = 1x10, activation=ELU, L2 = 0.001
Dropout	dropoutProbability=0.55
BatchNormalization	
Conv2D	filters=100, kernelSize = 1x10, activation=ELU, L2 = 0.001
Add	Res_2
MaxPooling2D	poolSize=1x3
Dropout	dropoutProbability=0.55
Conv2D	filters=200, kernelSize = 1x10, activation=ELU, L2 = 0.001
MaxPooling2D	poolSize=1x2 (1x3 w/o aug)
Dropout	dropoutProbability=0.55
BatchNormalization	
Flatten	
Dense	units=4, activation = Softmax, L2 = 0.001

Table 7. The Employed Model Architecture for Res CNN

Layer	Parameters
CNN or CNN+res	w/o Flatten and Dense layers
Permute	(2,3,1)
TimeDistributed	
Bidirectional LSTM	units=32, returnSequences=True
Bidirectional LSTM	units=32, returnSequences=False
Dropout	dropoutProbability=0.5
Dense	units=4, activation = Softmax, L2 = 0.1

Table 8. The Employed Model Architecture for CNN+LSTM/Res LSTM(Trained with cross-entropy loss and Adam optimizer)