Exploring the Performance of CRNN, LSTM, GAN, and CNN on EEG Data

Mehrab Beikzadeh[1§], Reza Rezvani[2§], Mohammad Askari[3§]
[1,2]Department of Computer Science
[3]Department of Civil Engineering
University of California, Los Angeles
[1]mehrabbeikzadeh@cs.ucla.edu, [2]rezar@cs.ucla.edu, [3]askari@ucla.edu

Abstract

In this project, we designed and compared the performance of four different models for EEG-based task classification: Naive Convolutional Neural Networks (CNN), Naive Long Short-Term Memory (LSTM), a combined approach of Generative Adversarial Networks and CNN (GAN + CNN), and a combined approach of CNN and LSTM (CNN + LSTM). We applied these models to a dataset containing EEG signals from four different task classes. The performance of each model was evaluated using metrics such as accuracy.

1. Introduction

Electroencephalography (EEG) is a non-invasive method of measuring the electrical activity of the brain. It involves the placement of electrodes on the scalp to record the electrical signals generated by the neurons in the brain. EEG is widely used in clinical settings to diagnose and monitor neurological disorders such as epilepsy, sleep disorders, and brain injuries. It is also used in research to study brain function and activity, including cognitive and emotional processes, and is a valuable tool in the field of neuroscience. EEG technology continues to evolve, with advances in digital signal processing and machine learning algorithms enhancing its diagnostic and research capabilities. The dataset provided by BCI [1] for training includes 2115 trials of time series data, consisting of 1000 data points recorded from 22 electrodes placed on the scalps of 9 subjects. The dataset is designed to predict the outcomes of 4 distinct classes of motor imagery tasks: left hand (class 1), right hand (class 2), both feet (class 3), and tongue (class 4). Upon analyzing the dataset, we have opted to construct neural networks to facilitate the prediction process.

[§]Equal contribution

2. Architecture Design

2.1. Data Augmentation and Preprocessing

To visualize the data in our research, we first loaded the dataset and selected one channel (in this case, channel 9). We then separated the data into the four different motor imagery task classes and plotted all of them on the same graph for comparison. From this visualization, we discovered that the first half of the time series signals were more useful for classification, as the signals in the second half were largely indistinguishable from each other due to their high similarity.

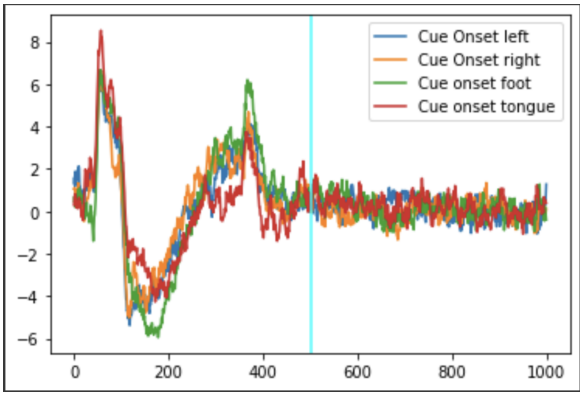


Figure 1. Data visualization for our four classes in one channel

After observing that the second half of the time series signals were less informative for classification, we decided to trim the data. The original shape of the

data was (2115, 22, 1000), but we opted to keep only the first half of the time series signals, resulting in a trimmed dataset with a shape of (2115, 22, 500).

To augment the dataset, we employed various techniques including max pooling with a stride of 2, averaging with noise addition, and subsampling. Additionally, since we intended to use 2D convolution, we added an extra dimension to the data with a width of 1. These methods helped to increase the diversity of the dataset, enhance the accuracy of our models, and prevent overfitting.

2.2. Model Selection

During the model selection phase, we investigated various architectures for our project and the details can be found in Figure 7. In this phase, the models were trained using data from all participants and we also reviewed previous studies on EEG data classification. To train and test all models, we utilized Google Colab with TPU runtime.

2.2.1 Naive-CNN

In particular we looked at the DeepConv architecture in [2] that used a pure-CNN architecture to capture the features from raw EEG data.The method that has been suggested comprises four convolutional layers. With our basic data pre-processing and no RNN layers, the model had a test accuracy of 70.49%.

2.2.2 Naive-LSTM

The LSTM model's architecture includes three bidirectional LSTM layers and a fully connected layer. As the input data is a time series, the RNN is employed to perform the classification task, which is expected to perform better than CNN. The reason for this is that RNN can extract information more efficiently across the time steps dimension compared to CNN.

2.2.3 CNN + LSTM

The proposed model consists of our convolutional layers combinned with one bidirectional LSTM layer, and a fully-connected layer in a sequential manner. In this model, the CNN component is able to extract features across time steps first. This reduces the difficulty of the LSTM in directly extracting features from a long time series with high correlation between adjacent time steps.

2.2.4 GAN + CNN

In this method, a GAN was utilized on top of CNN to augment the data, and a conditional GAN was employed to address the four different classes. A one-hot vector representation was inserted for the generator, and a one-hot tensor was provided, which consists of a set of matrices where the nth matrix for the nth class contains ones, and the rest of the matrices contain zeros. The objective of the discriminator was adjusted to determine whether the image of class X looked real or fake instead of just discerning whether the image looked real or fake.While it is possible to use the GAN method separately for each class, employing a conditional GAN is generally better. This is because a conditional GAN can be executed more quickly and is easier to implement, especially when dealing with a larger number of classes.

3. Result

3.1. Best Model Performance

In the first phase of the study, the performance of multiple architectures, which were previously discussed in the preceding section, was assessed using all subjects and a maximum time step of 500ms. The time series was truncated at T=500ms due to the presence of noise beyond this point, making it difficult to distinguish meaningful signals. Table 1 presents the tabulated data of the test accuracy. The findings indicate that the Naive CNN model outperformed other architectures, achieving a test accuracy of 70.49%.

Model Name	Accuracy
CNN	70.49%
CNN+LSTM	60.95%
LSTM	39.62%
GAN+CNN	68.23%

Table 1. Architectures Accuracies

3.2. Testing for Different Subjects

Secondly, the study employed the optimal model to determine the accuracy of the test dataset for each subject at a 500 millisecond time step, as depicted in Figure 2. It should be noted that due to the predominance

of datapoints pertaining to the first subject, the precision obtained by solely considering this subject is similar to that achieved by a model including all subjects as input. However, utilizing the model trained solely on the first subject for the entire dataset may result in comparable accuracy but with potentially greater error, as this model is less generalized.

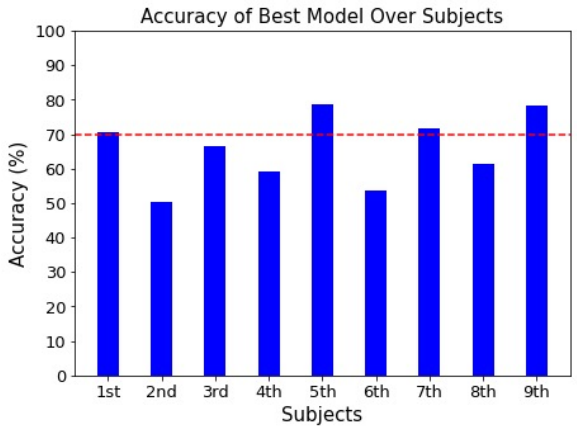


Figure 2. Accuracy of best model trained for each individual subject data

3.3. Testing for Different Time Durations

Five different time durations were examined to determine the optimal point to cut the data. Analysis of the Figure 3 suggests that removing data too early, such as from T=100ms, may result in significant loss of information for classification. Conversely, retaining nearly all or all of the data, such as from T=700ms and T=1000ms, may lead to overfitting noise, which is suboptimal for classification. Therefore, the most favorable time durations to trim the data are the median durations, namely T=300ms and T=500ms.

4. Discusion

4.1. Choice of hyperparameters and Architecture

To find the best combination of hyperparameters and architectures, we conducted a grid search, experimenting with different numbers of CNN and LSTM layers, their orders, the number of filters in CNN layers, and the hidden dimension in LSTM layers. We discovered that increasing the number of convolutional layers or decreasing the number of max pooling layers could lead to overfitting, while too much data loss could occur if too much information was removed

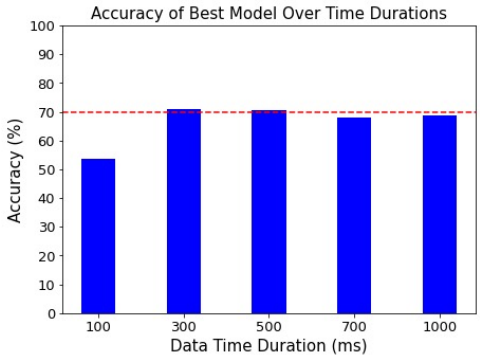


Figure 3. Accuracy of best model trained for each data time duration

from the time series data. Furthermore, increasing the filter size could enlarge the reception field of CNN, but this might also cause overfitting. We observed that bidirectional LSTM performed better than the normal one, and we experimented with a variety of activation functions for both CNN and LSTM. Our results showed that Exponential Linear Unit (ELU) was a better choice for the CNN part, and the default activation function (tanh) worked well for the LSTM part. Ultimately, we selected the hybrid architecture with the optimal hyperparameters and used the Adam optimizer with default parameters.

4.2. Effect of Data Augmentation

To reduce training and testing error and prevent overfitting, we employed various data augmentation techniques. During data preprocessing, we used methods such as averaging, noise adding, and max pooling, which resulted in a CNN model accuracy of over 70%. Additionally, we utilized GAN for data augmentation, but the accuracy table showed that the pure CNN approach performed better in this instance. We hypothesize that GAN might perform better on more complex data.

4.3. Next Steps

By harnessing greater hardware capabilities, we can advance our research efforts by refining temporal or attention-based structures, thus mitigating overfitting and reinforcing the outcomes observed in DeepConv + LSTM. An alternative avenue to explore is the application of VAEs to approximate data distribution and expand the scale of the training dataset.

References

[1] Clemens Brunner, Robert Leeb, Gernot Müller-Putz, Alois Schlögl, and Gert Pfurtscheller. Bci competition 2008–graz data set a. *Institute for Knowledge Discovery (Laboratory of Brain-Computer Interfaces), Graz University of Technology*, 16:1–6, 2008. 1

[2] Robin Tibor Schirrmeister, Jost Tobias Springenberg, Lukas Dominique Josef Fiederer, Martin Glasstetter, Katharina Eggensperger, Michael Tangermann, Frank Hutter, Wolfram Burgard, and Tonio Ball. Deep learning with convolutional neural networks for eeg decoding and visualization. *Human brain mapping*, 38(11):5391–5420, 2017. 2

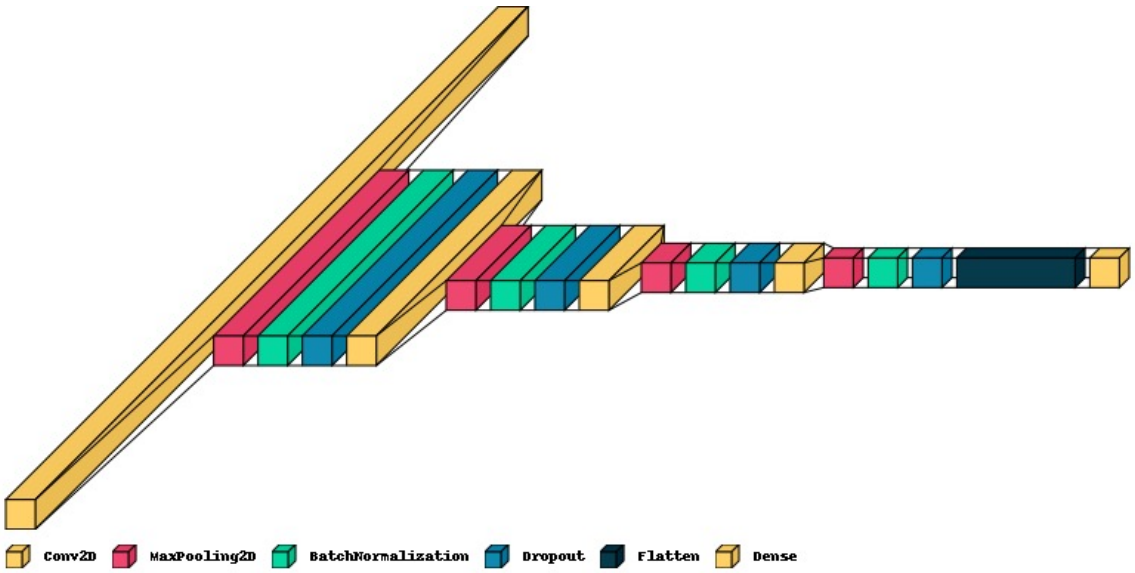
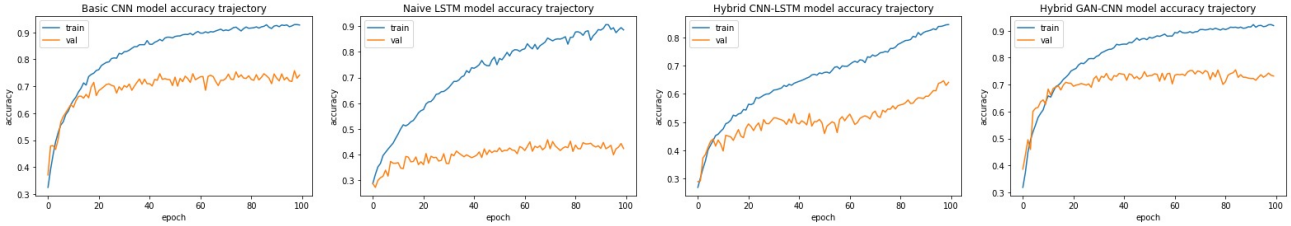
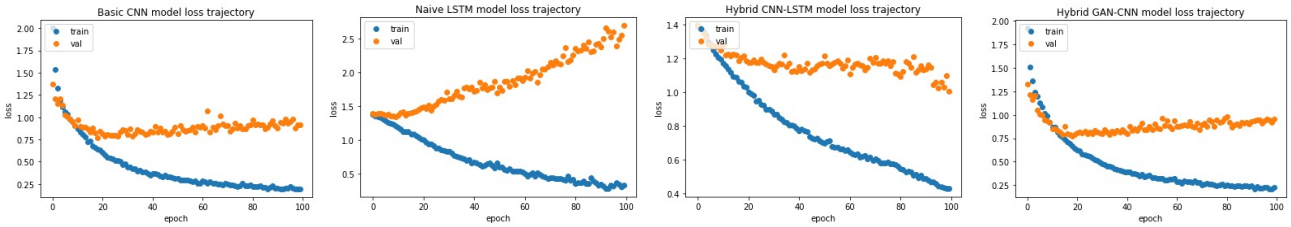


Figure 4. Naive CNN architecture layers illustration



(a) Naive CNN (b) Naive LSTM (c) CNN+LSTM (d) GAN+CNN

Figure 5. Models accuracy



(a) Naive CNN (b) Naive LSTM (c) CNN+LSTM (d) GAN+CNN

Figure 6. Models loss

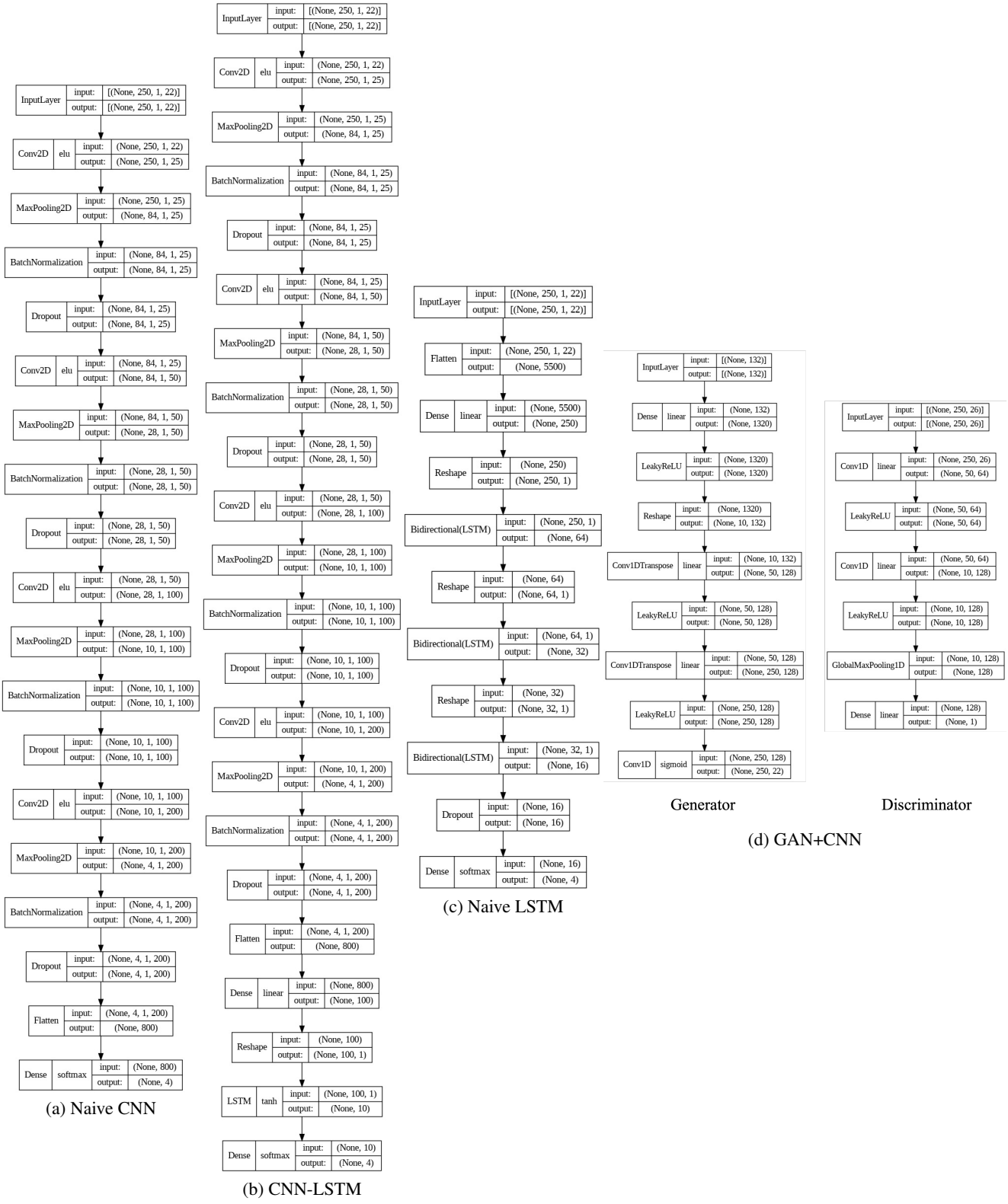


Figure 7. Models architectures