C247 Final Project - EEG Classifications

Guanyu Qian
UID: 606076335
gyqian@ucla.edu

Sihao Yang
UID: 205837588
sihaoyang@ucla.edu

Hexi Meng
UID: 406300552
fury1120@g.ucla.edu

Abstract

This project focuses on the classification of EEG signals using different neural network architectures, including CNNs, RNNs, and a combination of both. Our main objective is to enhance classification accuracy by employing the machine-learning techniques we acquired in the C247 lectures. Our findings demonstrate that 2D CNNs slightly outperform 1D CNNs by leveraging spatial relationships. Hybrid models incorporating depthwise convolutions, LSTM, and GRU layers do not significantly enhance performance.

1. Introduction

We aim to enhance the accuracy of classifying EEG signals through advanced data augmentation techniques and exploration of various neural network architectures, such as CNN, RNN, and hybrid models. Furthermore, we employed many essential concepts, including L1& L2 regularization, batch normalization, dropout, pooling, and so on, to analyze and tune the models.

This report summarizes our understanding of the course material and aims to contribute to the broader field by optimizing model performance for real-world EEG data analysis.

2. Results

This section elucidates the results of our data preprocessing and the comparative analysis of different model architectures implemented.

We explored multiple neural network architectures, including 1D Convolutional Neural Networks (CNNs) for temporal feature extraction, 2D CNNs to capture spatial dependencies across EEG channels, and hybrid models combining CNNs with recurrent neural layers (LSTM and GRU) to leverage spatial and temporal dynamics simultaneously.

For rigorous evaluation methodologies, we include two training processes across different time bins and train each model architecture on the entire dataset and data from a single subject (Subject 1) to investigate the impact of inter-subject variability on model performance. This dual approach allowed us to discover the models' capacities for generalized and personalized EEG data classification.

We include partial results from our implemented architectures to illustrate our results visually. Full results and comparisons are included in the Appendix, highlighting distinct performance trade-offs among the architectures.

2.1. Data Processing & Augmentation

Recognizing the characteristics of the EEG data, as shown in the discussion session, we identify three issues. First, data in the rear bins appear idle, potentially prolonging the training process without contributing meaningful information. Second, the original dataset size (2115 trials) may be insufficient for robust model training. Lastly, the essential features may be under-represented without any pooling extractions.

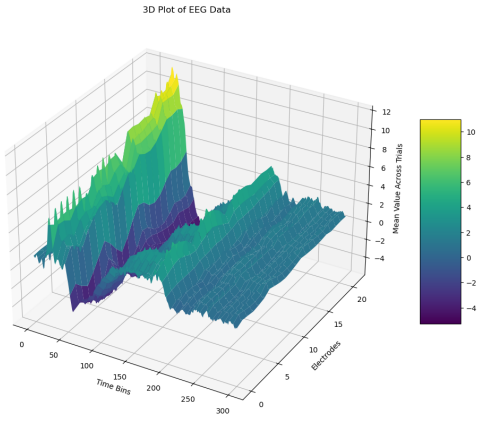


Figure 1. 3D plot of the pre-processed EEG over 22 electrodes and 300 time bins

Therefore, We employed cropping, noise addition, max pooling, and sub-sampling to address these challenges and create a more robust dataset for training our models. Fig 1 visualize the training data after our processing in 3D.

After implementing our model, we tested it across different time intervals and found that cropping at 600 bins

resulted in the highest accuracy.

To avoid class imbalance, we also visualize the class distribution of four actions, including *Cue Onset left, right, foot, and tongue.* The visualization of this distribution, as illustrated in Fig 2, reveals a balanced distribution among the four classes.

Figure 2. Class Distributions in the Training Set

2.2. 1D & 2D CNN

Our exploration of neural network architectures for EEG data classification starts with implementing 1D Convolutional Neural Networks (CNNs) to assess the efficacy of 1D structures in capturing relevant features from EEG data and to compare their performance with 2D CNNs.

In the 1D implementation, we treat EEG data as 22-length vectors (electrodes) with 300 time bins. The input dimension can be represented as (samples, time bins (after max pooling), electrodes), which is **(8460,300,22)** for our processed training data.

To enhance our model's ability to leverage spatial relationships between EEG channels, we subsequently adopted 2D CNNs. The 2D architecture treats EEG data like images, with each "pixel" representing an electrode's measurement at a specific time. To facilitate this, we introduce a height dimension of 1, thus reshaping our training data input shape to **(8460, 300, 1, 22)**. This adjustment allows the 2D CNN to exploit spatial information across electrodes in addition to the temporal dynamics, potentially leading to a more comprehensive understanding of the EEG signals.

The detailed 1D/2D CNN is included in the appendix. It has four convolutional layers, incorporating max pooling, batch normalization, and dropout. Filters' sizes progressively increase from 32 to 256.

2.2.1 Performance

Our appendix shows detailed model architecture and parameters. After comparing the performance and training time of both 1D and 2D CNNs [1], we found that 1D CNN has fewer parameters and slightly lower testing accuracy (0.7) than 2D CNN (0.72). However, both models took almost the same time to train (6-7ms/step) with a GPU. After considering all the factors, we decided to continue with the 2D CNN model for further RNN and CNN hybrid implementation.

Different training and testing approaches are shown in the appendix. We observed the testing accuracy of the model trained solely on subject 1 significantly dropped from 70% to 30 %.

Table 1. CNN Model Accuracies

Model	Training (%)	Validation (%)	Test (%)
1D CNN	78.0	91.9	70.4
2D CNN	78.8	89.0	72.2

2.3. RNN (Pure LSTM)

Using RNNs, specifically LSTM (Long Short-Term Memory) networks, to analyze EEG data is a strategic choice for several reasons. EEG data is sequential and temporal, capturing fluctuating electrical activity in the brain over time. RNNs, particularly LSTMs, are designed to handle sequential data by maintaining a 'memory' of past inputs. This allows them to capture temporal dynamics and dependencies in the data, which are crucial for understanding brain activity patterns that evolve over time. The detailed architecture of RNN is included in the appendix.

2.3.1 Performance

The relatively low accuracy indicates that the model struggles to classify the EEG data according to the labels [2]. LSTMs process data sequentially, which is ideal for capturing temporal dependencies. However, they may not adequately capture spatial relationships between features because they treat each feature independently without considering its interactions with others. As a result, it is better to incorporate convolutional layers before the LSTM layers.

Table 2. RNN Model Accuracies

Model	Training (%)	Validation (%)	Test (%)
Pure LSTM	76.9	78.8	47.6

2.4. CNN + Depthwise Convolution+ LSTM

Based on the results from the CNN, we can appreciate its strong capability in recognizing spatial features, which allows for a clear interpretation of the relationships and interactions between different channels. As discussed in the previous section, adding a flowing LSTM layer may excel in processing and predicting based on time series data. By integrating both models, we create a hybrid model capable of concurrently handling spatial and temporal elements.

Furthermore, each convolutional neural network layer incorporates a depthwise convolution, effectively preventing overfitting. Combining these three elements and their synergistic effect will provide a detailed understanding of both the spatial and temporal aspects of EEG signals. The detailed architecture of this model is included in the appendix.

2.4.1 Performance

The outcomes did not meet our initial expectations. Surprisingly, the test accuracy achieved by this model was lower than that of a pure CNN model [3]. This result prompts a critical evaluation of several factors that could contribute to this discrepancy. First, integrating LSTM layers, while theoretically advantageous for capturing temporal dependencies in EEG data, may introduce additional complexity and parameters to the model. This increased complexity can lead to challenges in training, potentially requiring more data or more sophisticated optimization strategies to learn from the temporal dynamics of the data effectively. Second, the depthwise convolutions, designed to enhance the model's ability to process spatial features efficiently and reduce the risk of overfitting, may also reduce the model's capacity to generalize from the training data to unseen test data if not correctly tuned. This delicate balance between model complexity, capacity, and generalization needs careful calibration.

Table 3. CNN+ Depthwise Convolution+ LSTM Model Accuracies

Model	Training (%)	Validation (%)	Test (%)
CNN+D+LSTM	87.4	97.4	70.7

2.5. CNN with GRU

In addition, we conducted an investigation on the impact of the RNN layer, where we replaced the LSTM with the GRU layer after the CNN and pooling layers. This allowed us to create a model that can effectively process both spatial features through the CNN layers and temporal or sequential patterns through the GRU layers. The gating mechanism of the GRUs enables them to manage the flow of information between time steps, thereby enabling them to capture long-term dependencies while mitigating the vanishing gradient problem that typically affects standard RNNs.

2.5.1 Performance

As a result, with the GRU layer, the model achieves significantly higher training and validation accuracy, as shown in Table [4]. However, the test accuracy remains at 70.2%, which is close to the CNN model; we think this result might be because of the more complex model architecture or the different learning dynamics. The more complex CNN+GRU model might capture intricate patterns in the training and validation sets but might not generalize well to the test set. Because of random chances, we need to increase the size of the test dataset to find out if the decrease in performance can be explained by statistical significance.

Table 4. CNN + GRU Hybrid Model Accuracies

Model	Training (%)	Validation (%)	Test (%)
CNN + GRU	76.5	91.6	71.8

This model has a test accuracy above 70%, yet still slightly lower than the CNN model. We conclude that GRU could add complexity that might not be necessary when dealing with multiple subjects, where simpler models like CNNs can generalize better across diverse data, and when trained solely on one subject, the data sequences are more likely to be aligned and consistent in terms of the temporal sequence, which plays to the strengths of the GRU.

3. Discussion

Our implementations of various neural network architectures have revealed some critical insights into the complex nature of EEG classification. While 2D CNNs have a slight advantage over 1D CNNs in capturing spatial relationships, implementing hybrid models such as CNN-LSTM or CNN-GRU does not significantly improve classification accuracy compared to the pure CNN model.

Furthermore, models trained exclusively on data from Subject 1 performed much worse in generalizing compared to the same model trained on the entire dataset, indicating a tendency toward overfitting when models are tailored to a single subject's data. This observation aligns with concepts discussed in C247, highlighting the risks of overfitting and the importance of diverse training samples for robust model performance.

Each of our models is trained using a separate CPU and GPU. We observed a significant training speed boost on GPU compared to CPU ($10^{1\sim2}ms$/step to $5ms$/step), underscoring GPUs' superior performance in handling the parallel processing demands of neural network training.

Future studies should focus on refining hybrid models and improving regularization techniques to enhance generalizability. Personalization and transfer learning offer promising avenues for improving classification accuracy by tailoring models to individual variability. Additionally, examining specialized models like EEGNet, designed explicitly for EEG data, may provide valuable insights for optimizing our approaches. [1]

References

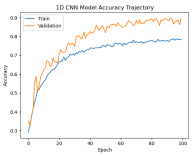
[1] Robin Tibor Schirrmeister, Jost Tobias Springenberg, Lukas Dominique Josef Fiederer, Martin Glasstetter, Katharina Eggensperger, Michael Tangermann, Frank Hutter, Wolfram Burgard, and Tonio Ball. Deep learning with convolutional neural networks for eeg decoding and visualization. *Human Brain Mapping*, 38:5391–5420, 08 2017. 3

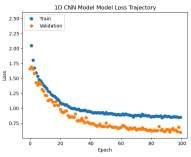
Model: 1D CNN 4 layers

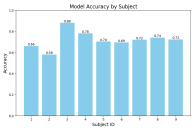
Layer (type)	Output Shape	Param #
conv1d_4 (Conv1D)	(None, 300, 32)	3552
max_pooling1d_4 (MaxPooling 1D)	(None, 100, 32)	0
batch_normalization_12 (Bat chNormalization)	(None, 100, 32)	128
dropout_12 (Dropout)	(None, 100, 32)	0
conv1d_5 (Conv1D)	(None, 100, 64)	10304
max_pooling1d_5 (MaxPooling 1D)	(None, 34, 64)	0
batch_normalization_13 (Bat chNormalization)	(None, 34, 64)	256
dropout_13 (Dropout)	(None, 34, 64)	0
conv1d_6 (Conv1D)	(None, 34, 128)	41088
max_pooling1d_6 (MaxPooling 1D)	(None, 12, 128)	0
batch_normalization_14 (Bat chNormalization)	(None, 12, 128)	512
dropout_14 (Dropout)	(None, 12, 128)	0
conv1d_7 (Conv1D)	(None, 12, 256)	164096
max_pooling1d_7 (MaxPooling 1D)	(None, 4, 256)	0
batch_normalization_15 (Bat chNormalization)	(None, 4, 256)	1024
dropout_15 (Dropout)	(None, 4, 256)	0
flatten_3 (Flatten)	(None, 1024)	0
dense_3 (Dense)	(None, 4)	4100

Total params: 225,060
Trainable params: 224,100
Non-trainable params: 960

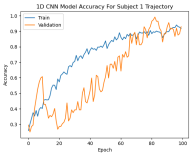
1D-CNN Model

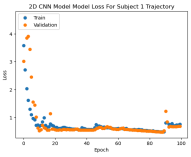


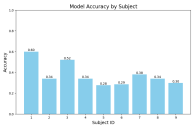




Train on all subjects







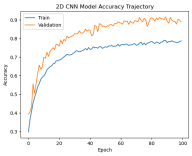
Train on subject 1

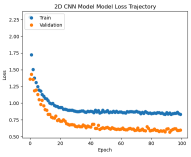
Model: 2D CNN 4 layers

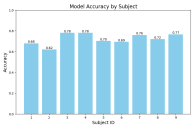
Layer (type)	Output Shape	Param #
conv2d_4 (Conv2D)	(None, 300, 1, 32)	17632
max_pooling2d_4 (MaxPooling 2D)	(None, 100, 1, 32)	0
batch_normalization_4 (Batc hNormalization)	(None, 100, 1, 32)	128
dropout_4 (Dropout)	(None, 100, 1, 32)	0
conv2d_5 (Conv2D)	(None, 100, 1, 64)	51264
max_pooling2d_5 (MaxPooling 2D)	(None, 34, 1, 64)	0
batch_normalization_5 (Batc hNormalization)	(None, 34, 1, 64)	256
dropout_5 (Dropout)	(None, 34, 1, 64)	0
conv2d_6 (Conv2D)	(None, 34, 1, 128)	204928
max_pooling2d_6 (MaxPooling 2D)	(None, 12, 1, 128)	0
batch_normalization_6 (Batc hNormalization)	(None, 12, 1, 128)	512
dropout_6 (Dropout)	(None, 12, 1, 128)	0
conv2d_7 (Conv2D)	(None, 12, 1, 256)	819456
max_pooling2d_7 (MaxPooling 2D)	(None, 4, 1, 256)	0
batch_normalization_7 (Batc hNormalization)	(None, 4, 1, 256)	1024
dropout_7 (Dropout)	(None, 4, 1, 256)	0
flatten_1 (Flatten)	(None, 1024)	0
dense_1 (Dense)	(None, 4)	4100

Total params: 1,099,300
Trainable params: 1,098,340
Non-trainable params: 960

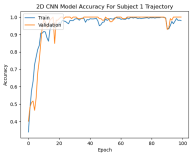
2D-CNN Model

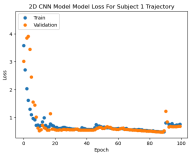






Train on all subjects





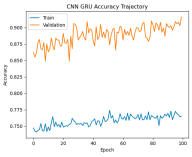
Train on subject 1

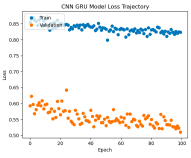
Model:CNN+GRU

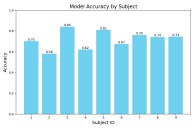
Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 300, 1, 25)	13775
max_pooling2d (MaxPooling2 D)	(None, 100, 1, 25)	0
batch_normalization (Batch Normalization)	(None, 100, 1, 25)	100
dropout (Dropout)	(None, 100, 1, 25)	0
conv2d_1 (Conv2D)	(None, 100, 1, 50)	31300
max_pooling2d_1 (MaxPoolin g2D)	(None, 34, 1, 50)	0
batch_normalization_1 (Bat chNormalization)	(None, 34, 1, 50)	200
dropout_1 (Dropout)	(None, 34, 1, 50)	0
conv2d_2 (Conv2D)	(None, 34, 1, 100)	125100
max_pooling2d_2 (MaxPoolin g2D)	(None, 12, 1, 100)	0
batch_normalization_2 (Bat chNormalization)	(None, 12, 1, 100)	400
dropout_2 (Dropout)	(None, 12, 1, 100)	0
conv2d_3 (Conv2D)	(None, 12, 1, 200)	500200
max_pooling2d_3 (MaxPoolin g2D)	(None, 4, 1, 200)	0
batch_normalization_3 (Bat chNormalization)	(None, 4, 1, 200)	800
dropout_3 (Dropout)	(None, 4, 1, 200)	0
flatten (Flatten)	(None, 800)	0
dense (Dense)	(None, 40)	32040
reshape (Reshape)	(None, 40, 1)	0
bidirectional (Bidirection al)	(None, 20)	780
dense_1 (Dense)	(None, 4)	84

Total params: 704779 (2.69 MB)
Trainable params: 704029 (2.69 MB)
Non-trainable params: 750 (2.93 KB)

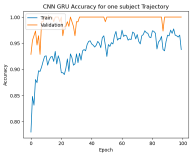
CNN + GRU Model

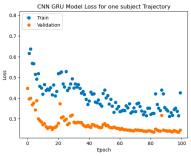


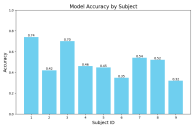




Train on all subjects





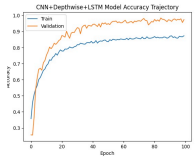


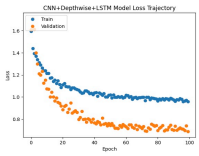
Train on subject 1

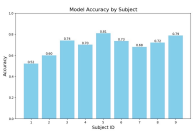
Model: CNN+Depthwise+LSTM

Layer (type)	Output Shape	Param #
conv2d_8 (Conv2D)	(None, 300, 1, 32)	17632
depthwise_conv2d_4 (Depthw iseConv2D)	(None, 300, 1, 64)	64
max_pooling2d_8 (MaxPoolin g2D)	(None, 100, 1, 64)	0
batch_normalization_8 (Bat chNormalization)	(None, 100, 1, 64)	256
dropout_10 (Dropout)	(None, 100, 1, 64)	0
conv2d_9 (Conv2D)	(None, 100, 1, 64)	102464
depthwise_conv2d_5 (Depthw iseConv2D)	(None, 100, 1, 128)	128
max_pooling2d_9 (MaxPoolin g2D)	(None, 34, 1, 128)	0
batch_normalization_9 (Bat chNormalization)	(None, 34, 1, 128)	512
dropout_11 (Dropout)	(None, 34, 1, 128)	0
conv2d_10 (Conv2D)	(None, 34, 1, 128)	409728
depthwise_conv2d_6 (Depthw iseConv2D)	(None, 34, 1, 256)	256
max_pooling2d_10 (MaxPooli ng2D)	(None, 12, 1, 256)	0
batch_normalization_10 (Ba tchNormalization)	(None, 12, 1, 256)	1024
dropout_12 (Dropout)	(None, 12, 1, 256)	0
conv2d_11 (Conv2D)	(None, 12, 1, 256)	1638656
depthwise_conv2d_7 (Depthw iseConv2D)	(None, 12, 1, 512)	512
max_pooling2d_11 (MaxPooli ng2D)	(None, 4, 1, 512)	0
batch_normalization_11 (Ba tchNormalization)	(None, 4, 1, 512)	2048
dropout_13 (Dropout)	(None, 4, 1, 512)	0
flatten_3 (Flatten)	(None, 2048)	0
dense_6 (Dense)	(None, 40)	81960
reshape_3 (Reshape)	(None, 40, 1)	0
bidirectional_4 (Bidirecti onal)	(None, 20)	960
dense_7 (Dense)	(None, 4)	84

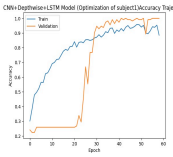
CNN + Depthwise + LTSM Model

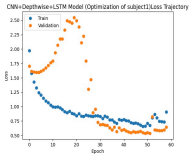


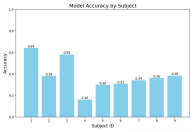




Train on all subjects



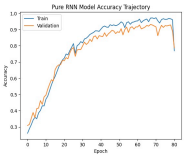


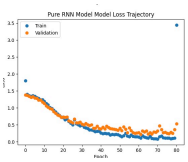


Train on subject 1

Model:RNN(LTSM)

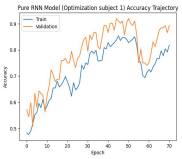
Layer (type)	Output Shape	Param #
flatten (Flatten)	(None, 6600)	0
dense (Dense)	(None, 40)	264040
reshape (Reshape)	(None, 40, 1)	0
bidirectional (Bidirectional)	(None, 40, 20)	960
dropout (Dropout)	(None, 40, 20)	0
bidirectional_1 (Bidirectional)	(None, 20)	2480
dropout_1 (Dropout)	(None, 20)	0
dense_1 (Dense)	(None, 4)	84

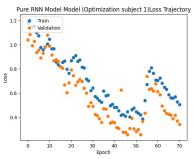


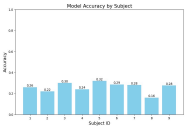


Train on all subjects

RNN(LTSM) Model







Train on subject 1

Model Performance Summary Table:

Model Name	Test accuracy across all subjects (train on entire data)	Test accuracy on Subject 1 (train on entire data)	Test accuracy across all subjects (train on subject 1 data)	Test Accuracy on Subject 1 (train on subject 1 data)
1D CNN	0.70	0.66	0.40	0.60
2D CNN	0.72	0.68	0.43	0.64
CNN + GRU	0.72	0.70	0.50	0.74
CNN + Depthwise + LTSM	0.71	0.52	0.40	0.64
Pure LTSM	0.48	0.36	0.26	0.26

Original Data Visualization (without processing):