ECE C147/247 Final Project

Ethan Keshishian Matthew Loussinian Zachary Wong

University of California, Los Angeles
420 Westwood Plaza, Los Angeles CA 90095

Abstract

The use of electroencephalographic (EEG) data to overcome motor inabilities is an increasingly studied field. Optimized and efficient neural network architectures provide an accessible route for EEG classification. This report considers five distinct neural network architectures trained on the BCI Graz data set. It is shown that residual and stacked gated recurrent unit (GRU) modifications to purely convolutional neural networks (CNNs) offer similar testing performance with greatly reduced model sizes when trained for both single subjects and collections of subjects at a slight cost of robustness to data noise.

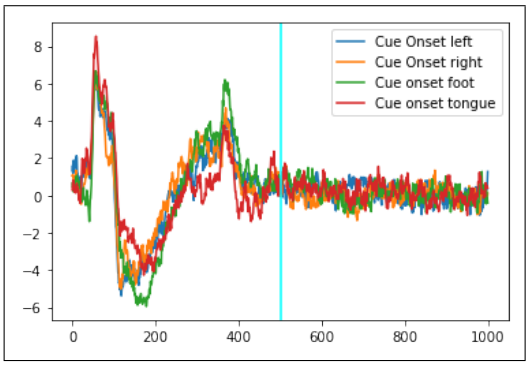


Figure 1. Example data with timestep cutoff

1. Introduction

1.1. BCI Graz Data Set

The dataset used is a reformatted version of the BCI Graz electroencephalographic data collected for nine individuals. Each was fitted with twenty-two electrodes sampled at 250Hz across 1000 timesteps [2], [3]. The subjects were then asked to envision one of four actions corresponding to a specific motor response. The resulting data was post-processed by bandpass filtering between 0.5Hz and 100Hz. An additional 50Hz notch filter was enabled to suppress line noise. For the purposes of this study, the data was compiled in a (2115, 22, 1000) array and sorted per-patient. An examination of the data reveals noticeable distinct features among patient cues up until near 500 seconds. Due to this issue and a small dataset, preprocessing was used to expand the input data to the tested models. This is described in greater detail in Appendix A.

1.2. Neural Network Architectures

Several neural network architectures were developed for training and testing the EEG data. Although each is distinct, several commonalities persist. The models were implemented in Tensorflow Keras with added callbacks for weight monitoring and learning rate monitoring. Each model has convolution layers, with added functionality to increase depth or reduce parameter counts. All were optimized using Adam to minimize a categorical cross entropy loss. Further model details as well as a discussion on architecture and hyperparameter optimization can be found in greater detail in Appendix A. These models were developed through architecture and hyperparameter optimization described in greater detail in Appendix A.

1.2.1 Architecture 1: CNN

A classical CNN architecture was implemented as a baseline model. CNNs have been previously shown to learn EEG features well [7] [9]. The CNN used in this project has four convolutional layers with batch normalization and dropout regularization after each, and a final softmax activation. The model shows no characteristic signs of overfitting and consistently yields test accuracies near seventy percent.

1.3. Architecture 2: ResCNN

The first modified architecture utilizes residual connections after the second and fourth layer to promote gradient highways through the architecture. A single residual connection had no effect, since residuals are usually best used when there are more layers on the model. Adding more

residuals did not have a linear effect on accuracy, since that entailed adding more convolutional layers, and accuracy decreased. When training and testing on all subjects, ResCNN achieves similar or slightly better accuracy and lower loss than Basic CNN (above seventy percent) despite training with half the amount of parameters.

1.3.1 Architecture 3: Hybrid CNN-LSTM

A secondary variation introduced recurrence to the already temporal dataset. Recurrence was hypothesized to improve classification due to the its inbuilt accounting for temporal changes. The architecture uses four CNN layers as in the base model but adds a recurrent layer (LSTM or GRU) at the end. Both options provided similar testing performance and in general fell below the testing errors of architecture 1.

1.3.2 Architecture 4: Stacked GRU with Bidirectionality

Additionally, several stacked recurrence models are explored as an alternative to the hybrid model. Neural networks with stacked LSTM layers have shown strong performance for radiotherapy data [8] and EEG data [4], [1]. This is likely because additional recurrent layers are able to more deeply encode temporal patterns learned from prior layers [6]. GRUs use the same gated operations as LSTMs without a forget gate, and have been shown to have similar to superior performance on smaller datasets. This justification as well as the minimized computational requirements informed the choice of the GRU. These layers may be further enhanced by adding bidirectionality, in which recurrent layers are able to communicate trends in both directions to other layers [10]. The formulated models are highly sensitive to model capacity and tend to overfit the training set, however after many iterations through potential architectures, this modified recurrent network was able to reach over sixty-eight percent classification accuracy when trained across the entire dataset.

1.3.3 Transformer

Finally, transformers are a novel architecture for sequence-to-sequence tasks capable of handling long-range dependencies. Addressed in further detail below, these features may be especially favorable for the classification of this type of dataset. Hybrid transformer networks are being used with greater regularity for EEG data, with much success [11], [12]. The transformer implemented builds on CNN layers and uses multiheaded attention. Multiple versions adjusting the placement of the attention layer were implemented. Tests resulted in classification accuracies of sixty-eight percent, but further iteration was paused due to CoLab GPU memory limitations.

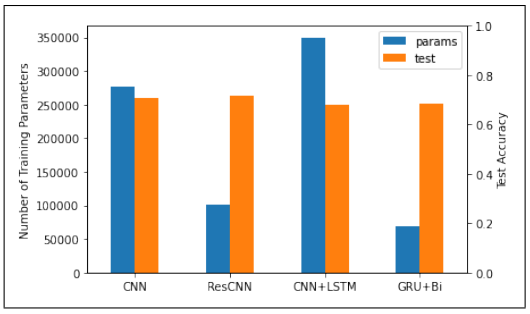


Figure 2. NN Architectures, Training Parameters, and Optimized Test Accuracy

Model	Single Subject	Full Dataset
CNN	0.7340	0.7553
ResCNN	0.7600	0.7400
CNN+LSTM	0.6649	0.3279
GRU+Bi	0.7021	0.3053

Table 1. Test Accuracy of Models Trained Using Single Subject Data

2. Results

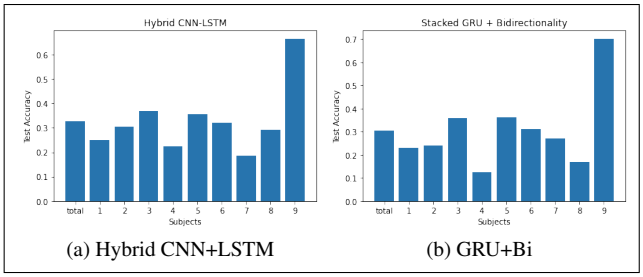
2.1. Single Subject Optimization

Utilizing the four architectures outlined, models were optimized for a single subject and tested using separate data from that same subject as well as combined data from all nine subjects. Results are provided in Table 1. All models, when trained and tested on the smaller datasets perform above larger datasets by at least 3-4%. However, while both architecture 1 and 2 are able to generalize well when tested across the entire dataset, recurrent architectures 3 and 4 perform significantly worse.

The generalization of single-subject training to other individual subjects was also considered. Results, summarized in Figure 3 show sub-normal prediction rates across different subjects, with many falling below the random guessing rate of twenty-five percent. This indicates that while there are collective commonalities among EEG data, individual EEG data varies widely from person to person.

2.2. Optimization Across All Subjects

In addition, all models were trained and tested across the entire dataset, each performing above 68%. The residual CNN network performed above all other models, reaching 0.7161. The full results may be found in Appendix B. As expected, models trained on the full dataset also perform well across all individuals. This is a markedly different trend than the one shown for single-subject optimization.



(a) Hybrid CNN+LSTM (b) GRU+Bi

Figure 3. Testing Accuracy for Recurrent Architectures Trained on Subject 9

2.3. Classification Across Time

Finally, the impact of preprocessing on the predictive capability of the models was undertaken, in which the input data to the models started at different times. As described above, the dataset appears to be more distinctive and rich up until 500s which determined the default cutoff time. As shown in Figure 4 the cutoff time was altered in increments of 100 and used to train architectures 1, 3, and 4. While both recurrent networks perform well with 400-500 timesteps of data, there is a sharp drop off near 200-300 timesteps and above 500 timesteps. The CNN architecture is far more robust to noisy datasets and maintains test accuracies above sixty-five percent until below a truncation point below 200.

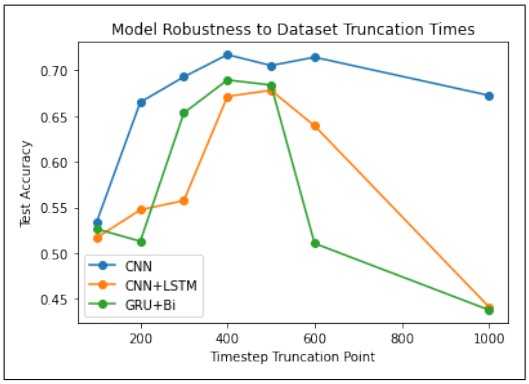


Figure 4. Impact of Temporal Dimension on Testing Accuracy

3. Discussion

Although recurrent networks were able to perform nearly as well as convolutional networks, testing accuracies consistently fell under those of architectures 1 and 2. CNN and ResCNN achieved maximum testing accuracies of 0.7054 and 0.7161 respectively when compared to 0.6783 and 0.6840 of CNN+LSTM and GRU+Bi when trained and tested across the entire dataset. This is contrary to initial expectations because recurrence has inbuilt accounting for temporal dynamics. An understanding of the divide might be traced to the nature of the dataset, which is noisy due to intrinsic sensor fidelity and human limitations. Furthermore, EEG classification for recognition of single motion likely relies on features across the temporal window rather than those within a small time window before and after the dataset. This may be distinctly different from recurrent networks used for speech recognition [5]. Further steps could be taken to test these hypotheses. Sensor noise may be mitigated using smoothing signals or additional frequency filters to downsample data. Recurrent models may also need more linear layers near the input for increased feature selection. Bidirectionality is shown to aid when added to a single layer (Appendix B) but does not lead to significant improvement when included in two or more layers.

A consideration of the optimization results across single subjects also provides insight into the challenges of the dataset and distinctive features of each model. For these tests, both CNN and ResCNN are able to learn features that generalize well across the entire dataset while the recurrent networks are unable to. Interestingly this divide does not appear to be tied to the number of trainable parameters each model has but rather to the architecture.

These considerations are not apparent when comparing testing results for models trained across the entire dataset. Residual networks tend to overfit, marked by a validation loss curve that rapidly decreases and then rises to a plateau in conjunction with a slowly rising validation accuracy.

The third study also reveals more robust performance of the CNN architecture to data noise. This is likely once again due to the nature of the architecture in being able to select key features which generalize. Recurrent networks assume implicitly that there are contextual dynamics important for classification which may not be the case for EEG data. In light of the trends indicated above, these results are expected.

References

[1] Tejas Agarwal, Shubham Raturi, TK Vybhav, and Mukhtiar Singh. Frobnication. *2020*. 2

[2] C. Brunner, R. Leeb, G.R. Müller-Putz, A. Schlögl, and G. Pfurtscheller. Bci competition 2008 - graz data set a. 1

[3] C. Brunner, M. Naeem, R. Leeb, B. Graimann, and G. Pfurtscheller. Separability of four-class motor imagery data using independent components analysis. 1

[4] Yu Chen Wenliang Che Gaowei Xu, Tianhe Ren. A one-dimensional cnn-lstm model for epileptic seizure recognition using eeg signal analysis. 14. 2

[5] Saeedeh Hashemnia, Lukas Grasse, Shweta Soni, and Matthew S. Tata. Human eeg and recurrent neural networks exhibit common temporal dynamics during speech recognition. 3

[6] KyungHyun Cho Yoshua Bengio Junyoung Chung, Caglar Gulcehre. Empirical evaluation of gated recurrent neural networks on sequence modeling. 2

[7] Vernon J Lawhern, Amelia J Solon, Nicholas R Waytowich, Stephen M Gordon, Chou P Hung, and Brent J Lance. Eegnet: A compact convolutional neural network for eeg-based brain-computer interfaces. 1

[8] Min Ma, Chenbin Liu, Ran Wei, Bi Liang, and Jianrong Dai. Predicting machine's performance record using the stacked long short-term memory (lstm) neural networks. 2

[9] 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 brain mapping and decoding of movement-related information from the human eeg. 1

[10] M. Schuster and K.K. Paliwal. Bidirectional recurrent neural networks. 2

[11] Gourav Siddhad, Anmol Gupta, Debi Prosad Dogra, and Partha Pratim Roy. Efficacy of transformer networks for classification of raw eeg data. 2

[12] Yonghao Song, Xueyu Jia, Lie Yang, and Longhan Xie. Transformer-based spatial-temporal feature learning for eeg decoding. 2

Appendix A - Summary of Architectures

The contents below describe the neural network architectures referenced throughout the report. There are several notable commonalities across all architectures. First, all datasets were trained and tested on pre-processed data using an augmentation process as described below. Additionally, all models utilize at least one convolutional layer with fixed kernel size of (10,1) and are optimized using Adam for a categorical cross-entropy loss function. During training, two callbacks were utilized for ease of monitoring results. The first is a learning rate monitoring callback which reduces the initial learning rate by a factor of 0.5 if the validation loss does not show improvement after ten epochs. The second callback tracks the lowest validation loss and saves the weights for future reference. All models were optimized across hyperparameters of epochs, learning rate, dropout, and batch size.

Dataset Augmentation

All architectures explored EEG data pre-processed through two primary steps. Initially, the data is truncated in time, averaged, and then added to noise. Then, to provide additional data, the dataset is split along in time and reshaped to provide four times the data. Truncation time was treated as an adjustable parameter and studied within the final report.

Architecture 1 - Basic CNN

The first architecture, as outlined in “C247 Final Project_ML.ipynb”, uses Elu activations for all convolutional filters. Each convolution has a kernel of size (10,1) and doubles the number of filters of the previous convolution, beginning with twenty-five. A 2D max pooling layer follows every convolutional layer with a (3,1) pool size. Batch normalization and dropout of 0.5 are used after every max pool layer. A fully connected layer with softmax activation is used as the output layer.

Architecture 2 - CNN with Residual Connections

The second architecture, as outlined in “C247 Final Project_ML.ipynb”, uses Elu activations for all convolutional filters. Each convolution has a kernel of size (10,1) and doubles the number of filters of *the one before the previous* convolution, beginning with twenty-five (25,25,50,50,100). Residual connections are placed after layers 2 and 4. A 2D max pooling layer follows every convolutional layer with a (3,1) pool size, except on layers preceding residual connections, in which case pool size is (1,1). Batch normalization and dropout of 0.5 are used after every max pool layer.

Architecture 3 - Hybrid CNN+LSTM

The third architecture, as outlined in “C247 Final Project_ZW.ipynb”, uses Elu activations for all convolutional filters. Each convolution has a kernel of size (10,1) and doubles the number of filters of the previous convolution, beginning with twenty-five. A 2D max pooling layer follows every convolutional layer with a (3,1) pool size. Batch normalization and dropout of 0.5 are used after every max pool layer. For the LSTM layer, a dropout of 0.6 and a recurrent dropout of 0.1 is applied.

Architecture 4 - Stacked GRU+Bidirectionality

The fourth architecture, as outlined in “C247 Final Project_ZW.ipynb”, uses Elu activations for all convolutional filters. Each convolution has a kernel of size (10,1) and doubles the number of filters of the previous convolution, beginning with twenty-five. A 2D max pooling layer follows every convolutional layer with a (3,1) pool size. Batch normalization and dropout of 0.5 are used after every max pool layer. Following the convolutional layers are a FC layer and 0.5 dropout layer. The primary architecture uses three GRU layers with 16-16-10 activation units respectively. Recurrent dropout of 0.1 is added to the first two units. Notably, bidirectionality is added to the middle GRU to improve model capacity. Dropout layers of 0.1 are included between each of the GRUs.

Several variations of these architectures were tested prior to selecting the primary stacked GRU, each with the same convolutional input outlined above (unless stated).

4a (Dense 100 3x GRU 63-32-10)

This model had a much larger FC layer and three stacked GRUs with 64, 32, and 10 activation units. Hyperparameter optimization over nine models revealed consistent performance near 63% validation accuracy. However, the model was prone to overfitting, potentially indicating excessive model complexity. Iterations on this model aimed to adjust model capacity to solve the problem of overfitting. All other models listed below follow the same convention. Results of preliminary architecture tests and hyperparameter optimization are given in Appendix B under the same naming convention.

4b (Dense 50 3x GRU 32-32-10)

4c (Dense 25 3x GRU 16-16-10)

4d (Dense 32 3x GRU 16-16-10); 2x Bidirectional This subset of models utilized bidirectionality in two of the layers: the first and second as well as the second and third. Results indicated that the added bidirectionality did not add much to

4e (Dense 32 4x GRU 16-16-16-10)

4f (Dense 32 2x GRU 16-10)

4g (Dense 32 3x GRU 16-16-10) This model took the primary stacked GRU architecture and reduced the number of convolution layers to one. The attempt at reducing model complexity did not improve performance, as this model was likely too simplistic.

Architecture 5 - CNN+Transformer

The final architecture, as outlined in “C247 Final Project_EK.ipynb”, builds on the same convolutional layering as in architecture 3, but replaces the LSTM layer with a multi-headed attention layer with 64 heads, 32 key dimensions, and dropout of 0.6.

Appendix B: Table of Testing Accuracies for Different Architectures

Table B.1 - Maximum Testing Accuracies for Different Architectures

Basic CNN			
Architecture	**Training Dataset**	**Max Test Accuracy (Subject 1)**	**Max Test Accuracy (Full Dataset)**
1	Subject 1	0.7340	0.7553
1	Entire Dataset	0.6700	0.7054
ResCNN			
Architecture	**Training Dataset**	**Max Test Accuracy (Subject 1)**	**Max Test Accuracy (Full Dataset)**
2	Subject 1	0.7600	0.7400
2	Entire Dataset	0.6900	0.7161
CNN+LSTM Hybrid			
Architecture	**Training Dataset**	**Max Test Accuracy (Subject 9)**	**Max Test Accuracy (Full Dataset)**
3	Subject 9	0.6649	0.3279
3	Entire Dataset	0.7500	0.6783
Stacked GRU+Bidirectionality			
Architecture	**Training Dataset**	**Max Test Accuracy (Subject 1)**	**Max Test Accuracy (Full Dataset)**
4	Subject 9	0.7021	0.3053
4	Entire Dataset		0.6840
4a	Entire Dataset		0.6637
4b	Entire Dataset		0.6417
4c	Entire Dataset		0.6417
4d	Entire Dataset		0.6304
4e	Entire Dataset		0.6490
4f	Entire Dataset		0.6631
4g	Entire Dataset		0.5271
CNN+Transformer			
Architecture	**Training Dataset**	**Max Test Accuracy (Subject 1)**	**Max Test Accuracy (Full Dataset)**
5	Entire Dataset		0.6823

Figure B.1 - Hyperparameter Optimization for Architecture 4 - The tables below are included in the Jupyter notebook and are given as representative of the hyperparameter and architecture optimization done before selection of each model. Further undocumented optimization was performed for this architecture for different input data and additional hyperparameters.

B.1.a - Architecture 4a

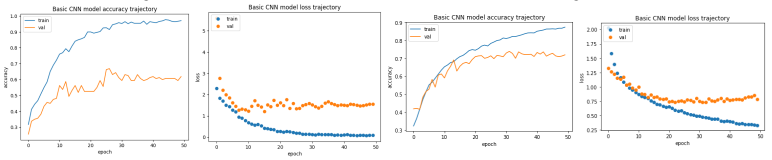
Model#	Batch	Dropout	Epochs	LR	TestAcc1	TestAcc2	MinValLoss	Epoch
0	32	0.1	50	1e-3	0.6309	0.6287	0.9084	8
1	32	0.3	50	1e-3	0.6236	0.6259	0.9024	8
2	32	0.6	50	1e-3	0.6309	0.6332	0.9690	6
3	64	0.1	50	1e-3	0.6552	0.6637	0.8254	10
9	64	0.2	50	1e-3	0.6292		1.0405	9
4	64	0.3	50	1e-3	0.6388	0.6270	0.9402	11
5	64	0.6	50	1e-3	0.5745	0.5745	1.0289	6
6	128	0.1	50	1e-3	0.5756	0.5880	1.0100	11
7	128	0.3	50	1e-3	0.6163	0.6270	1.0105	14
8	128	0.6	50	1e-3	0.6473	0.6253	1.0573	12

B.1.b - Architecture 4b

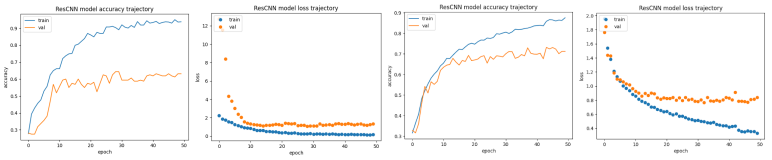
Model#	Batch	Dropout	Epochs	LR	TestAcc1	TestAcc2	MinValLoss	Epoch
10	64	0.2	50	1e-3	0.6417		1.0165	16
11	64	0.1	50	1e-3	0.6337		0.8088	10
13	64	0.1	50	1e-3	0.6275	0.6174	1.0001	10
15	64	0.1	50	1e-3	0.6095	0.5570	1.0704	7

Figure B.2 - Training Accuracy and Loss Trajectories for Primary Architectures

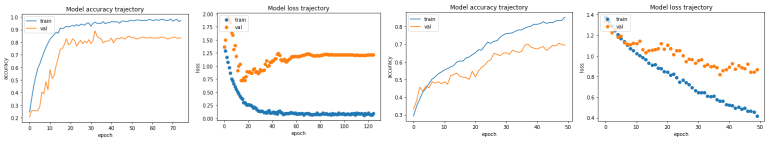
Basic CNN Subject 1 **Basic CNN on All Subjects**



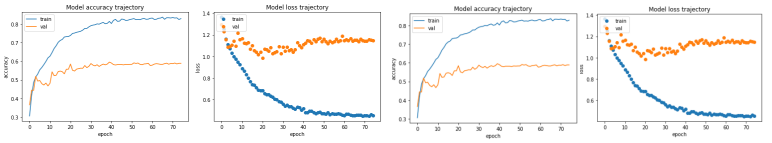
ResCNN on Subject 1 **ResCNN on All Subjects**



CNN+LSTM on Subject 9 **CNN+LSTM on All Subjects**



GRU+Bi on Subject 9 **GRU+Bi on All Subjects**



CNN+Transformer on All Subjects