Classifying EEG Signals via Ablation Analysis: A 2D Laplacian and Deep Learning Approach

Cristian Rodriguez, Adrik Shamlonian, Adnan Armouti, Giacomo Fratus
University of California, Los Angeles

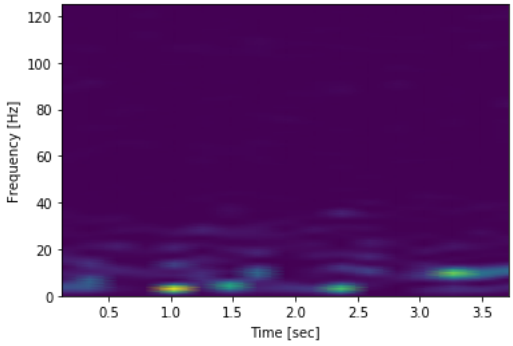
Abstract

In this paper we aim to demonstrate different deep learning architectures in order to classify what type of object a subject is imagining, by analyzing Electroencephalography samples taken during a short period of time. We have implemented different architectures and techniques in order to maximize the prediction accuracy of our models, and report results. The 4 general models used are: Shallow CNN, Deep CNN, CNN+GRU, and CNN+LSTM. The results obtained are promising and demonstrate how our deep learning implementations may be used to predict what object a person is imagining with up to a 72.42% accuracy.

1. Introduction

The electroencephalography (EEG) datasets reflect the coordinated activity of millions of neurons that are located near a non-invasive scalp electrode used to capture the data. Moreover, it is believed that EEG records dipoles transmitted through the scalp. The data consists of 22 channels, each with 1000 time bins per sample. Since the channels come from physical placement of electrodes on the subject's scalp, so the data contain both spatial and temporal features. Additionally, we looked at the frequency components of the data over time. As we can see in Figure 1, most of the signal power is located between 0 and 40Hz. Initial tests on the data with a fully connected network yielded test accuracies of about 35%, meaning dense layers alone could not accurately capture these features. Another obstacle is the small overall size of the data set, as training deeper models can be difficult on small datasets. We decided to evaluate the performance of both shallow and deep CNNs on the data set, as well as RNNs, GRUs, and LSTMs. We also implemented several data preprocessing and augmentation techniques to boost performance on certain different models. We implemented 5 different types of models: Shallow 2D CNN, Deep 2D CNN, Shallow 3D CNN, CNN + GRU, CNN + LSTM. The preprocessing techniques along with their description and justification can be found in Figure 2.

Figure 1. Spectogram of Random Channels



After implementing different combinations of models

Figure 2. Data Processing Techniques

Method	Description	Reason
Noise Augmentation	Add normally distributed noise to a copy of training samples	Increase training sample size [7]
Spatial Filter	From each electrode, subtract the average of its neighbors.	Improve the spatio-temporal resolution [2].
Low Pass Filter	Apply a low pass filter with cutoff frequency f_c	Majority of signal power is at $(2 - 30Hz)$
Subsampling	Subsample every N time bins to produce N new training examples	Increase training set size, and decrease sample length
Time Window	Take only a subset of the time bins (first 312 or last 688)	Earlier time bins seemed to contain richer information

and preprocessing techniques, we came to the following brief conclusions. Deep 2D and 3D CNN models with no preprocessing or data augmentation struggled to generalize on the small data set. RNN, GRU, and LSTM (without CNN) did not perform well on the raw data because of the long sequence of length 1000 and caused vanishing gradients by the last time sequence. A shallow 3D CNN with a low pass filter with $f_c = 25Hz$ performed significantly better than previous models. Deep CNN with preprocessing, batch normalization, and dropout performed better than a deep CNN with no preprocessing or data augmentation. The combination of a CNN and GRU with preprocessing worked well.

Our best performing model utilizes a 3D convolutional layer in order to capture both the electrodes' spatial and temporal localities. We rearranged the 22 electrode inputs into a 7x6 matrix on which the location of the electrodes are mapped with the same arrangement that covers the subject's head, and zero padded rest of the matrix elements. The 3D convolutional layer consists of 40 filters with a 25x3x3 kernel size receiving a 3D tensor of one trial and as a result outputting 40 channels. The 3D CNN layer is followed by two fully connected layers and an average pooling layer in between. This model's performance was improved by using a spatial filter which is explained in the discussion section.

2. Results

See Tables 1-4 for the results obtained from different combinations of model architectures and preprocessing techniques. The best result that we were able to get, i.e. the combination of options that provided the best results for each model, was the Shallow 3D CNN with the 2D channels and average neighbors preprocessing steps. For a more in depth description see introduction. The performance of our best model can be seen in Figure 3.

Figure 3. Performance of Best Model

Run	Test Accuracy (%)
1	72.2
2	72.9
3	70.7
4	73.6
5	72.7
Average	72.4

This best final model was then used to evaluate subjects 0-8, by training the model on samples from the first 1777 samples of X_train_valid, and evaluated on the relevant time bins from X_test.

3. Discussion

3.1. Working with the raw data

We found that shallow CNNS performed better than deeper CNNs, which suggests that more complicated models perform worse than simpler models on this dataset. Moreover, shallow Cnns outperformed deep CNNs paired with GRU or LSTM units, both in parallel and sequentially. We noticed that while recurrent architectures are typically good at classifying sequential data, the sequence length of 1000 time bins seemed to be a big issue in the parallel configuration. The sequential model still had a long sequence after the convolutional layers, and suffered from the same problem.

Deep CNNs outperformed Deep CNNs paired with GRU or LSTM units, both in parallel and sequentially. Even with several layers of convolutions before the recurrent layer, the temporal resolution going into the LSTM/GRU was likely poor. Adding this layer only increased the model complexity without providing any significant benefit.

CNNs outperformed Vanilla RNNs, LSTMs, and GRUs on this raw dataset. This may be due to the length of the time bins (LSTMs and GRUs have empirically been shown to perform better for 250 time bins.

Pairing shallow CNNs with GRU/LSTM units sequentially performed better than in parallel. In the sequential model, we believe the majority of the work is done by the convolutional layers. Pairing deep CNNs with GRU/LSTM units in parallel performed better than sequentially. This is because CNNS are good with dealing with high dimensional data such as the 22 electrode inputs in our dataset, while the GRU is better at keeping the temporal locality of the dataset [3].

3.2. Focusing on Shallow CNNs

Based on these preliminary insights, we decided to focus our attention on Shallow CNNs (both standalone and when paired sequentially with GRE/LSTM). We decided to discard the Shallow CNN + GRU/LSTM units models when paired in parallel. We then decided to pre-process the data by introducing spatial information to the dataset. This was done by rearranging the 22 electrode channels into a sparse 6x7 image. By doing this, performance for Shallow CNN improved compared to the 22 single dimension electrode case. By introdicing more spatial information to the shallow CNN, the CNN layer was able to perform better spatial feature extraction and therefore better spatiotemporal feature extraction as well. Performance for Deep CNN worsened compared to the 22 single dimension case, indicating that there is a simpler underlying distribution to the data, and that smaller moderls are less likely to overcomplicate and overfit to this distribution. Performance for Shallow CNN paired with GRU/LSTM units improved compared to the 22

single dimension electrode case. This is because in general, the 3D CNN extracted better features than the 2D CNN, so the overall performance after adding a recurrent layer was still better than the 2D CNN.

3.3. Prioritizing time bins

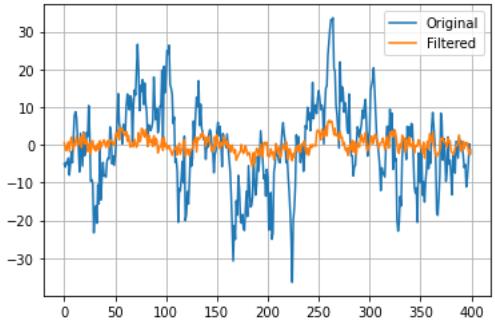
After seeing our improved model performance with the 2D electrode configuration, we explored whether ceratin time bins were more important than others. We found that Models trained on all 1000 time bins performed better than the first 312 time bins (when subject is responding to stimulus and "planning"), which in turn performed better than the last 688 time bins (when the subject begins to take "action" in response to the stimuli). This suggests that information encoded in the first 312 time bins are more important than the last 618 time bins. However, because all 1000 time bins are important, we cannot discard these last 618 time bins, and must instead focus on implementing a pre-processing algorithm that acts more heavily on the last 618 time bins.

3.4. Data preprocessing and augmentation

Having narrowed down our two best performing models to the Shallow CNN and the Shallow CNN + GRU, we began investigating different data preprocessing and augmentation techniques. We chose 4 methods: low pass filtering, average neighboring, subsampling, and a denoising variational enconder.

Average neighboring provided the best performing data preprocessing method. As seen in Figure 4, removing the average of the neighbours from a given channel gets rid of the underlying low frequency waveforms. This proved to be very effective, improving our test accuracies by 1-2% on our best model.

Figure 4. Channel 10 Signals



Low pass filtering using a butterwoth filter was the also useful. EEG data has been empirically shown to reside in the $0 - 30Hz$ frequency range and the last 618 time bins seem to specifically have higher frequency noise.

The Variational Autoencoder, trained to perform low pass filtering, performed poorly relative to other data preprocessing methods. This can be attributed to the method in which the Denoising VAE (DVAE) was trained; The 2D data had to be flattened to be passed into a fully connected network of 3 layers in the encoder, and then unflattened during the decoder. It was trained to perform low pass filtering. This reduction in dimensionality may have made it more difficult for the DVAE to decode and re-encode the spatial information of the electrodes. Although DVAEs have been empirically shown to remove Gaussian and Specular noise, a much more complex DVAE may be required to remove high frequency noises. With more resources (e.g. unlimited GPU usage by Google Colab with no "cooldown" period, and more time to work on this project), this may have been possible to achieve.

Subsampling did not improve either the Shallow CNN or the Shallow CNN + GRU models. We performed subsampling after the data set was split into both training and validation, to avoid "cross-contamination" of the samples and overfitting to the combined training and validation sets. This also ensures the degreee better generalization capacity and robustness of our trained models. This was surprising, because we expected subsampling to at least improve Shallow CNN + GRU performance, since a reduction in the time bins of the input data would have improved the performance of the individual GRU layer. In hindsight, however, this is expected because either the GRU or Shallow CNN layers must have relied more heavily on the adjacent time bins (either from the CNN layer for the former, or the input data for the latter). Subsampling would have removed the information encoded in the adjacent time bins.

References

[1] Schirrmeister, R.T., Springenberg, J.T., Fiederer, L.D.J., Glasstetter, M., Eggensperger, K., Tangermann, M., Hutter, F., Burgard, W. and Ball, T. *Deep Learning With Convolutional Neural Networks for EEG Decoding and Visualization* Hum. Brain Mapp., 38:5391-5420, (2017) https://doi.org/10.1002/hbm.23730

[2] Hjorth, B. *An on-line transformation of EEG scalp potentials into orthogonal source derivations* Electroencephalography and Clinical Neurophysiology 39:526-530, (1975)

[3] Lizhen Wu, Chun Kong, Xiaohong Hao, Wei Chen *A Short-Term Load Forecasting Method Based on GRU-CNN Hybrid Neural Network Model* Mathematical Problems in Engineering, vol. 2020 Article ID 1428104 (2020) https://doi.org/10.1155/2020/1428104

[4] K. K. Dutta *Multi-class time series classification of EEG signals with Recurrent Neural Networks* 2019 9th International Conference on Cloud Computing, Data Science Engineering (Confluence), Noida, India, 2019, pp. 337-341 doi: 10.1109/CONFLUENCE.2019.8776889

[5] Fazle Karim et al. *Multivariate LSTM-FCNs for time series classification* Neural Networks, Volume 116, (2019) Pages 237-245, ISSN 0893-6080 https://doi.org/10.1016/j.neunet.2019.04.014

[6] Vincent, P., Larochelle, H., Bengio, Y., Manzagol, P.-A. *Extracting and composing robust features with denoising autoencoders* ICML 2008.

[7] Wang F, Zhong S H, Peng J, Jiang J and Liu Y *Data augmentation for eeg-based emotion recognition with deep convolutional neural networks* Lecture Notes Comput. Sci. 10705 LNCS 82–93 (2018)

[8] Pascal Vincent, Hugo Larochelle, Isabelle Lajoie, Yoshua Bengio, and Pierre-Antoine Manzagol *Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion* The Journal of Machine Learning Research, 9999:3371–3408, 2010.

[9] https://github.com/titu1994/MLSTM-FCN

[10] https://bit.ly/3vvAGWI

[11] https://bit.ly/2NpmU6M

Table 1: No Preprocessing (1D Channel)

Model	Time Bins	Test Accuracy (%)
Shallow 2D CNN	1000	69.1
Deep 2d CNN	1000	68.8
Vanilla RNN	1000	23.5
GRU	1000	21.0
LSTM	1000	21.2
Sequential: Shallow 2d CNN + GRU	1000	67.5
Sequential: Shallow 2D CNN + LSTM	1000	61.6
Sequential: Deep 2D CNN + GRU	1000	65.5
Sequential: Deep 2D CNN + LSTM	1000	63.0
Parallel (MTSC): Shallow 2D CNN + GRU	1000	60.0
Parallel (MTSC): Shallow2d CNN + LSTM	1000	56.7
Parallel (MTSC): Deep 2D CNN (+ SE Block) + GRU	1000	66.6
Parallel (MTSC): Deep 2D CNN (+ SE Block) + LSTM	1000	65.5

Table 2: Preprocessing (2D Channels)

Model	Time Bins	Test Accuracy (%)
Shallow 3D CNN	1000	70.0
Shallow 3D CNN	688	50.1
Shallow 3D CNN	First 312	69.8
Deep 3D CNN	1000	65.5
Deep 3D CNN	Last 688	49.2
Deep 3D CNN	First 312	64.3
Sequential: Shallow 3D CNN + GRU	1000	70.0
Sequential: Shallow 3D CNN + GRU	Last 688	47.4
Sequential: Shallow 3D CNN + GRU	First 312	67.9
Sequential: Shallow 3D CNN + LSTM	1000	64.1
Sequential: Shallow 3D CNN + LSTM	Last 688	43.8
Sequential: Shallow 3D CNN + LSTM	First 312	60.3

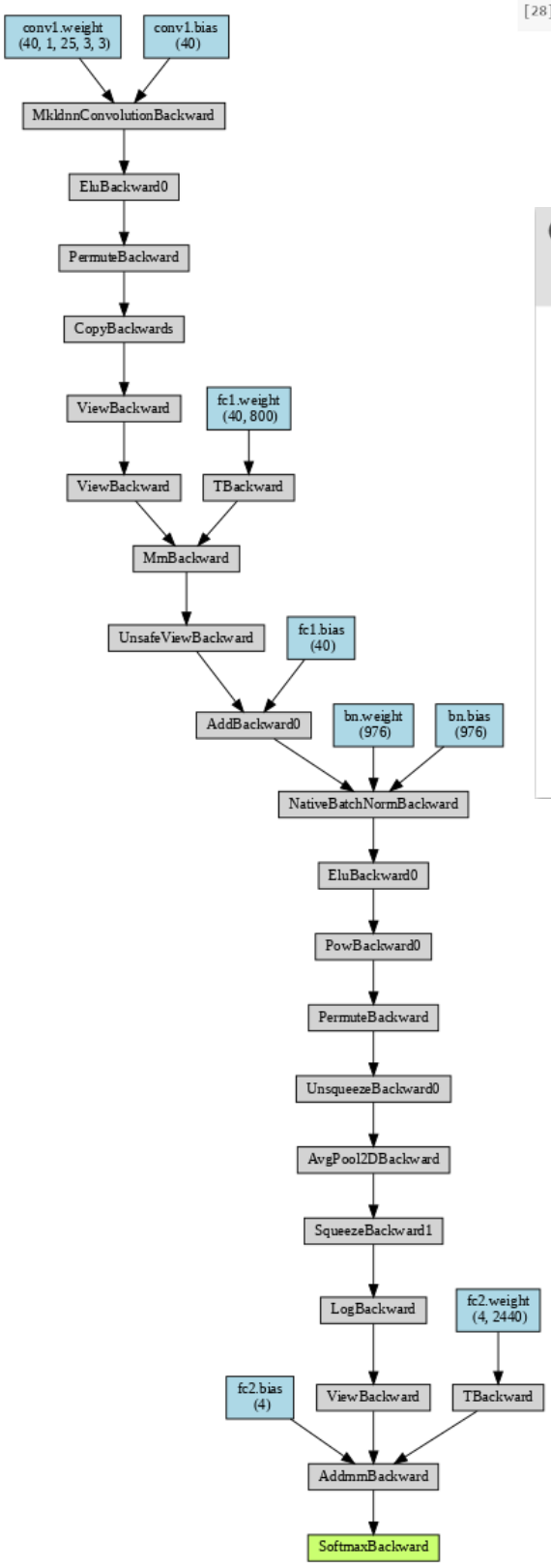
Table 3: Preprocessing (2D Channels, All 1000 Time Bins)

Model	Preprocessing	Test Accuracy (%)
Shallow 3D CNN	2D channels, average neighbors	71.6
Shallow 3D CNN	2D channels, low pass filter	69.5
Shallow 3D CNN	2D channels, DVAE	24.2
Deep 3D CNN	2D channels, subsampling	67.0
Deep 3D CNN	2D channels, average neighbors	66.8
Deep 3D CNN	2D channels, low pass filter	69.1
Deep 3D CNN	2D channels, DVAE	24.6
Deep 3D CNN	2D channels, average neighbors	67.6

Table 4: Test Accuracy of Model on Different Subjects

Subject	Test Accuracy (%)
0	70.0
1	54.4
2	90.0
3	40.0
4	61.7
5	57.1
6	80.0
7	82.0
8	87.2

Architecture of Best Performing Model




```
[28] print(model)

ShallowConv(
  (conv1): Conv3d(1, 40, kernel_size=(25, 3, 3), stride=(1, 1, 1))
  (fc1): Linear(in_features=800, out_features=40, bias=True)
  (elu): ELU(alpha=1.0)
  (bn): BatchNorm1d(976, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
  (avgpool): AvgPool1d(kernel_size=(75,), stride=(15,), padding=(0,))
  (fc2): Linear(in_features=2440, out_features=4, bias=True)
  (softmax): Softmax(dim=1)
)
```

```
from torchsummary import summary

summary(model, (32, 6, 7, 1000))
```

```
----------------------------------------------------------------
        Layer (type)               Output Shape         Param #
================================================================
            Conv3d-1       [-1, 40, 976, 4, 5]           9,040
              ELU-2       [-1, 40, 976, 4, 5]               0
            Linear-3            [-1, 976, 40]          32,040
       BatchNorm1d-4            [-1, 976, 40]           1,952
              ELU-5            [-1, 976, 40]               0
       AvgPool1d-6             [-1, 40, 61]               0
            Linear-7                 [-1, 4]           9,764
           Softmax-8                 [-1, 4]               0
================================================================
Total params: 52,796
Trainable params: 52,796
Non-trainable params: 0
----------------------------------------------------------------
Input size (MB): 5.13
Forward/backward pass size (MB): 12.83
Params size (MB): 0.20
Estimated Total Size (MB): 18.15
----------------------------------------------------------------
```