ECE C147 Project Report - J3MNet

Matthew Craig, Jayson Shinn, Jeffrey Ma, Jonathan Carlson
University of California, Los Angeles
{mcraig90505, jayshinn, jma777, jonathancarlson}@g.ucla.edu

Abstract

In recent years, researchers have been intensively searching for better computational methods to classify and generate electroencephalography (EEG) data. Here, we propose a novel architecture, J3M-Net, that combines a Deep Convolutional Neural Network (ConvNet) with Long Short-Term Memory (LSTM) layers for the purpose of classifying EEG motor imagery data. Our results report a small increase over a solely convolutional approach (74.66% test accuracy J3M-Net, 72.97% test accuracy Depp ConvNet). We also explore augmenting limited EEG data samples with a Variational Autoencoder (VAE), and show that this approach does not perform well. Finally, we explore how training on single subject data performs worse than multiple subjects, as models trained on a single subject do not generalize well to new subjects.

1. Introduction

In this project, we sought to make improvements on neural network architectures for decoding EEG signals (BCI competition IV dataset 2a [Brunner et al., 2008]) [1]. We took as a starting point for our proposed architectures the Deep ConvNet approach used by Schirrmeister et al. (2017) [2]. We expanded on the original ConvNet architecture by training a recurrent neural network (RNN) on the output, hoping to capture more of the time series dynamics of EEG data. We compared our architecture, which we propose as J3M-Net (pronounced like “Gem Net”), against the original Deep ConvNet architecture, and a pure recurrent architecture built from LSTM layers. We also examined the accuracy of training our J3M-Net architecture on a single subject in the dataset.

Finally, we examined strategies for increasing prediction accuracy when a limited number of samples are available to train on. We used a variational autoencoder (VAE) to reduce the dimensionality of our data so we could train our models more quickly while maintaining the key features of the dataset (as demonstrated in Krishna et. al, 2020) [3]. In contrast to a normal autoencoder, we hoped that by using a VAE to encode our data before training, we could find a regularized latent space that would allow us to generate new samples by adding noise to our encoded training samples. We theorized that this would provide a better regularization effect than training on the original dataset with noise added, since the latent space could encode entirely new examples.

1.1. Results

We found a marginal improvement using our J3M-Net compared to a modified version of the Deep ConvNet proposed by Schirrmeister et al. (2017). We tested against a Deep ConvNet without a spatial filter in the first convolutional block, opting instead for only a temporal convolution. We also used different data preparation methods, as suggested by TA Tonmoy Monsoor. We achieved a classification accuracy of 74.66% on the processed dataset with J3M-Net, compared to 72.97% with the Deep ConvNet on the same dataset. We note that Schirrmeister et al. were able to achieve classification accuracies exceeding 80% in their original paper, which we were not able to reproduce with our dataset preparation methods. Both the Deep ConvNet and J3M-Net far outperformed the pure LSTM model, which achieved approximately 40% accuracy.

Our J3M-Net architecture performed poorly when trained on the output of our VAE. We tested different latent dimension sizes (100 and 250) and found similar performance on both, with approximately 30% classification accuracy on the test set.

Our final experiment examined the performance of training on a single subject compared to the full training dataset. We found that training on only a single subject decreased classification accuracy to 62.5% for that subject’s test set, and to approximately 32% on the entire test set.

1.2. Discussion

1.2.1 J3M-Net

The J3M-Net architecture uses the same convolutional blocks as the Deep ConvNet, so we find it probable that the success of J3M-Net is due to the same time series feature maps that the Deep ConvNet extracts. We hoped that by including LSTM layers after the convolutional output, we could capture some time series dynamics still present in the features maps which the classification output of Deep ConvNet is unable to exploit. However, the input to the LSTM layers consists of only 11 time steps, so we find it likely that there were very few time series dynamics left in the convolutional output. Thus, the LSTM layers provided a very marginal improvement at best, since the output of Deep ConvNet is already well suited to successful multiclass classification on EEG data.

Compared to the pure LSTM model, we found a significant improvement in classification accuracy by adding convolutional blocks before the LSTM input, which allowed us to reduce the number of LSTM units and increase the training speed. We believe this is because the EEG data, even after preprocessing, is relatively “noisy”, which results in poor state updates for the pure LSTM model. The temporal convolutional filters, combined with max pooling operations, are good at extracting temporal features from the noisy data and creating a well-ordered sequence that is much better suited to LSTM updates.

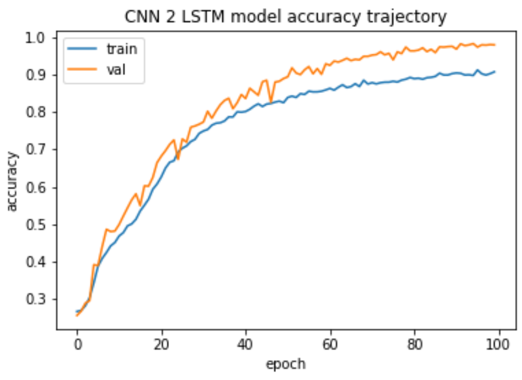

Figure 1. J3M-Net Accuracy Trajectory

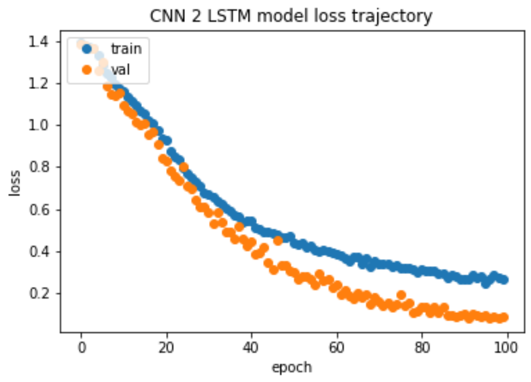

Figure 2. J3M-Net Loss Trajectory

1.2.2 Using a VAE for training J3M-Net

Our hope with using a VAE was to find a latent space that allowed us to generate new training data via sampling, to increase the size of our dataset. For this to be feasible, we required that similar classes be colocated in the latent space, so that we could expect samples from the same area to have the same labels. Because of the temporal nature of EEG data, we experimented with a VAE based on temporal convolution and LSTM cells (based on Lawhern et al., 2016) [4], to hopefully build a latent space that could be constructed and reconstructed in sequence.

Unfortunately, the VAE models that we tested were unable to find such a well regularized latent space. Figure 3 below shows a scatter plot of the first two dimensions of the latent space. It is clear from the figure that there is no clustering between examples of the same class. Thus, we were unable to generate additional data to augment our training. We found similar results across pairwise latent dimensions for both VAE sizes we tested (100 and 250).

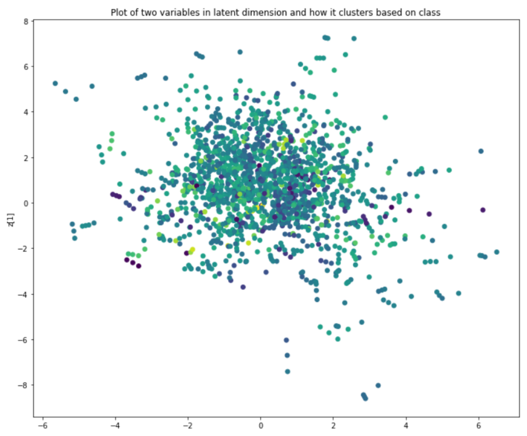

Figure 3. VAE Latent Space

When we examined the reconstructed output from our VAE compared with the ground truth in Figure 4, we found that the VAE appears to have learned a denoising function. This held true for all of the reconstructions we examined, across classes and across electrodes.

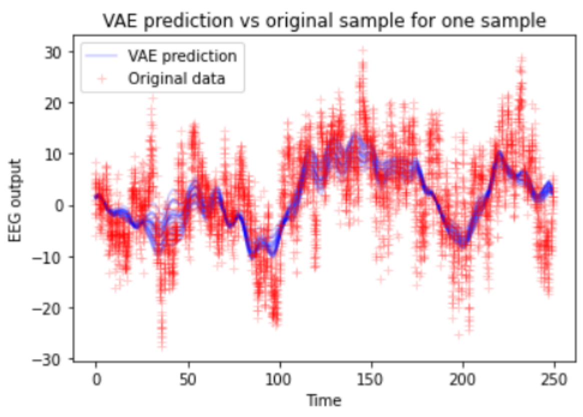

Figure 4. VAE Reconstructed Output vs. Ground Truth for one sample

These results seem to indicate that while the VAE does approximately reconstruct the input data, it is unable to fully capture some features or variations that are essential to classify the data, resulting in the poor performance that we observed.

It is also possible that our J3M-Net architecture is not well suited to classify examples in the latent space of the VAE. Perhaps the sequential nature of the original EEG data is not preserved in the same way in the latent space, and as a result the temporal convolutions and LSTM layers that performed well on the original dataset are not suited to the encoded data. Future research could examine alternative classification schemes on the VAE encoded output.

1.2.3 Single-Subject Training and Testing

Our single-subject training results were unsurprising. Due to differences between subjects, training on only a single subject's data yielded worse results than training on the entire dataset, as the training failed to capture unique features of each subject. Our results are summarized in Table 1, but the average performance is around 36% accuracy on the full test set when training with only a single subject's training set.

Alternatively, we hypothesized that the effects we observed could be due to the amount of training data available for a single subject (759 examples for one subject, after augmentation and separating validation sets), which could lead to the effects we observed. We tried training our J3M-Net model with 759 random examples on the full dataset, and obtained a worse accuracy on the subject 1 test set (44.5%) than the model trained only on the subject 1 training set (62.5%). Figures 5 and 6 below show the loss trajectory and the accuracy trajectory of the model on the randomly selected data.

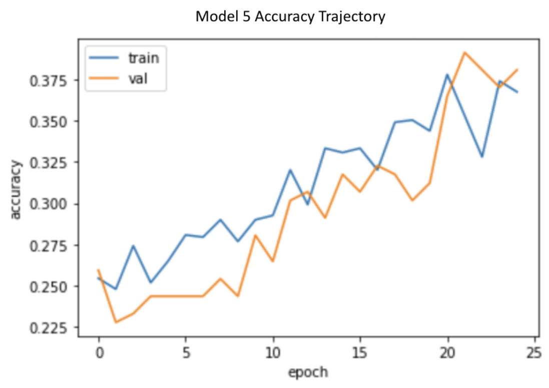

Figure 5. Random Sample Accuracy Trajectory

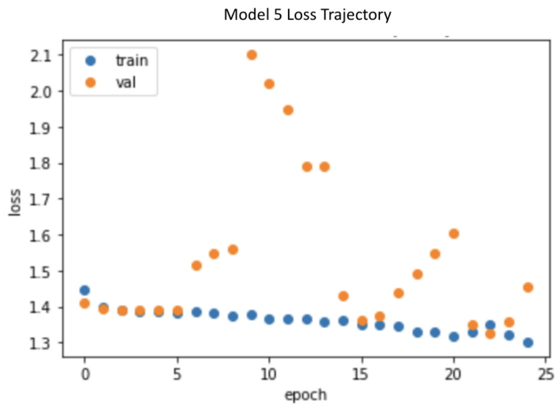

Figure 6. Random Sample Loss Trajectory

This may indicate that a lack of training examples is not the primary reason for the drop in performance compared to the full training set. Instead, we hypothesize that different subjects exhibit different enough behavior that examples from one subject do not generalize as well to another subject. A generalized model may struggle to classify all subjects with a high degree of accuracy, since it would have to account for each subject's peculiarities. This is perhaps another area for future research.

References

[1] Brunner C, Leeb R, Müller‑Putz G, Schlögl A, Pfurtscheller G (2008): BCI Competition 2008–Graz Data Set A. Institute for Knowledge Discovery (Laboratory of Brain‑Computer Interfaces), Graz University of Technology; pp 136–142.

[2] Schirrmeister, R. T., Springenberg, J. T., Fiederer, L. D. J., Glasstetter, M., Eggensperger, K., Tangermann, M., Hutter, F. & Ball, T. (2017). Deep learning with convolutional

neural networks for EEG decoding and visualization. Human Brain Mapping, Aug. 2017. Online: http://dx.doi.org/10.1002/hbm.23730

[3] Krishna, G., Tran, C., Carnahan, M., & Tewfik, A.H. (2020). Constrained Variational Autoencoder for improving EEG based Speech Recognition Systems. ArXiv, abs/2006.02902.

[4] Lawhern, Vernon & Solon, Amelia & Waytowich, Nicholas & Gordon, Stephen & Hung, Chou & Lance, Brent. (2016). EEGNet: A Compact Convolutional Network for EEG-based Brain-Computer Interfaces. Journal of Neural Engineering. 15. 10.1088/1741-2552/aace8c.

Appendix A. Model Architectures

Notation: POOL are all MaxPool, NORM means BatchNormalization, CONV means convolution, CONVT means convolution transpose, UPS means upsampling. All models have input shape of [375x1x22] (except for decoder and LSTM RNN).

J3M-Net
[375x1x25] CONV: 25 filters of size 10x1 with ELU
[125x1x25] POOL: 3x1 filters with stride 3
[125x1x25] NORM-Dropout rate of 0.4
[42x 1x 50] CONV: 50 filters of size 10x1 with ELU
[42x1x25] POOL: 3x1 filters with stride 3
[42x1x25] NORM-Dropout rate of 0.4
[42x1x100] CONV: 100 filters of size 10x1 with ELU
[21x1x100] POOL: 2x1 filters with stride 2
[21x1x100] NORM-Dropout rate of 0.5 BatchNorm layer
[21x1x200] CONV: 200 filters of size 10x1 with ELU
[11x1x200] POOL: 2x1 filters with stride 2
[11x1x200] NORM-Dropout rate of 0.5 BatchNorm layer
[11x1x50] TimeDistributed(FC-ReLU) with 50 units
[11x20] LSTM-ReLU-Dropout with 20 units, dropout rate of 0.5, recurrent dropout of 0.1
[20] LSTM-ReLU-Dropout with 20 units, dropout rate of 0.5, recurrent dropout of 0.1
[4] FC with 4 units
[4] OUT Softmax layer

Our tested version of Deep ConvNet is the same as the convolutional layers of J3M-Net, with a softmax layer on the output of the last convolutional block.

VAE Encoder
Dropouts have dropout rate of 0.4 unless otherwise stated, LeakyReLUs have alpha of 0.2
[124x1x36] CONV: 36 filters of size 5x1x22 with stride 4
[42x1x36] POOL: maxpool size 3, stride 3
[42x1x36] NORM-LeakyReLU-Dropout
[38x1x72] CONV: 72 filters of size 5x1x36 with stride 1
[13x1x72] POOL: maxpool size 3, stride 3
[13x1x72] NORM-LeakyReLU-Dropout
[13x1x32] TimeDistributed(FC-ReLU) of 32 units
[20] LSTM-ReLU-dropout: 20 units, dropout rate of 0.5, recurrent dropout rate of 0.1
[latent_dim] FC with latent_dim units (for both mean and log variance)
[latent_dim] OUT sampling layer

VAE Decoder
Same Dropout rates as the encoder
[latent_dim] Input
[(latent_dim/10)x32] TimeDistributed(FC-ReLU) of 32 units
[20] LSTM-ReLU-dropout: 20 units, dropout rate of 0.5, recurrent dropout rate of 0.1
[13x1x72] FC with 936 units
[39x1x72] UPS: pools of size (3, 1)
[42x1x72] CONVT-ReLU: 72 filters of size 4x1x72, s1
[42x1x72] NORM-Dropout
[126x1x72] UPS: pools of size (3, 1)
[375x1x36] CONVT-ReLU: 36 filters size 125x1x72, s1
[375x1x36] NORM-Dropout
[375x1x22] CONVT-ReLU: 22 filters size 5x1x36, s1
[375x1x22] OUT

LSTM RNN
[250x22] Input
[250x32] LSTM, dropout 0.5, recurrent dropout 0.1
[250x64] LSTM, dropout 05, recurrent dropout 0.1
[1x4] LSTM
[4] OUT Softmax layer

Table 1. Model Results

Model	Dataset	Test set	Best Test Accuracy
Deep ConvNet	Full	Full Testset	72.97%
J3M-Net	Full	Full Testset	74.66%
RNN (LSTM)	Full	Full Testset	39.16%
J3M-Net	VAE output of full dataset (100 latent dimensions)	Full Testset	30.3%
J3M-Net	VAE output of full dataset (250 latent dimensions)	Full testset	30.9%
J3M-Net	Subject 1	Subject 1 testset	62.5%
J3M-Net	759 random samples from the full dataset	Subject 1 testset	44.5%
J3M-Net	Subject 0	Full Testset	39.22%
J3M-Net	Subject 1	Full Testset	32.17%
J3M-Net	Subject 2	Full Testset	40.86%
J3M-Net	Subject 3	Full Testset	32.96%
J3M-Net	Subject 4	Full Testset	34.48%
J3M-Net	Subject 5	Full Testset	32.39%
J3M-Net	Subject 6	Full Testset	42.49%
J3M-Net	Subject 7	Full Testset	37.53%
J3M-Net	Subject 8	Full Testset	31.43%