EEG Decoding Applications with Deep Neural Networks

Fabrice Harel-Canada
University of California, Los Angeles
fabricehc@g.ucla.edu

Abstract

This paper evaluates EEG decoding performance via the use of a variety of Deep Neural Networks (DNN) and has found that models consisting only of CNN components outperform those composed of RNN components as well as hybrid models that include components of both types. Furthermore, this study shows that single subject training is not better for evaluation for that specific subject than training on multiple subjects simultaneously. Rather, joint training across multiple subjects is observed to generalize performance more broadly and improve upon training on only a given subjects EEG signals. Finally, this study shows that the duration of a trial's time is not strongly correlated to model performance.

1. Introduction

This paper evaluates EEG decoding performance for classification tasks via the use of a variety of Deep Neural Networks (DNN) - CNNs, RNNs, and Hybrid Networks that combine elements of both. Specifically, the following research questions will be evaluated:

1.1. What model architectures perform the best on EEG data spanning multiple individuals?

1.2. Is classification accuracy better when trained and evaluated on the EEG data belonging to a specific person or over multiple people?

1.3. Does training on the EEG data of multiple people allow for acceptable accuracy on the individual level?

1.4. Does classification accuracy improve or degrade as a function of time?

In order to process EEG timeseries data successfully, I start with an analysis of different DNN architectures and the motivations for selecting them for these experiments. This analysis is conducted over all individuals in the BCI Competition dataset because the generalizability and therefor broad usefulness of any EEG classifier would be best served by accounting for an approximation of the average human's EEG signals, not those belonging to any specific person.

The waveforms of EEG data are particularly difficult to understand, and it would be incredibly difficult to engineer any features by hand, even for experts in the field. Fortunately, Convolutional Neural Networks (CNNs) are well known for their ability to extract features from complex data structures and so incorporating CNNs into the early steps of any model architecture is expected to improve prediction accuracy.

One notable limitation of traditional CNNs is that they have difficulties learning temporal dependencies between sequences of inputs. Recurrent Neural Networks (RNNs) were developed to overcome this limitation and fits well with the problem domain of EEG timeseries data. I hypothesize that RNNs will outperform CNNs because of the internal dynamics that maintain state across timesteps. I further hypothesize that the combination of both architectures into a hybrid model, with CNN layers acting as encoders / feature extractors that precede the RNN layers, will do the best of all.

Lastly, the two main types of RNNs that will be explored are Long Short-Term Memory (LSTM) RNNs and Gated Recurrent Unit (GRU) RNNs. The latter is being considered primarily because of having fewer parameters and the consequent time savings, which is important in environments limited to CPU training.

2. Results

This section presents the results of research questions in the order introduced. Unless otherwise specified, performance is always considered as accuracy on the test set unseen during training and validation.

As an opening caveat, certain configurations were *not* used in this analysis due to computational constraints associated with single processor (CPU / GPU) environments. Nonetheless, I will list following settings that were observed to have a positive effect on classification

accuracy on small scale tests and should be tried by others with more computing power and time:

1. Statefulness in RNNs were observed to have a 2-3% improvement in performance, but at the cost of forcing a smaller batch size (batch_size=1 in my case) and a training time increased by about 45x.
2. Cropped training allows for many segments of the original 1000 timesteps to be sampled at a certain rate. The authors of [1] selected a crop size of 500, which would increase the number of trials by a factor of 625. I picked a much more conservative crop size of 950, resulting in a 50-fold increase in the number of trials. In small trials, cropping increased performance by 4-6%, but at the cost of being 50x slower.

2.1. Best model architecture trained on multiple subjects

By far the best model was the Shallow CNN, followed by the Deep CNN. Overall, the models comprised only of CNN components did better than those comprised of any or all RNN components. See Table 1. for more details.

2.2. Subject specific training vs. Multiple subject training

This study indicates that training only on a single subject and evaluating the performance on that subject alone generates worse results than training on multiple subjects simultaneously. Figure 1. shows the performance of the best performing model, the Shallow CNN, when trained on individual people relative to being trained by all of them simultaneously (the orange baseline). Only one subject would be better off under the single subject training approach. See Table 2. for more details.

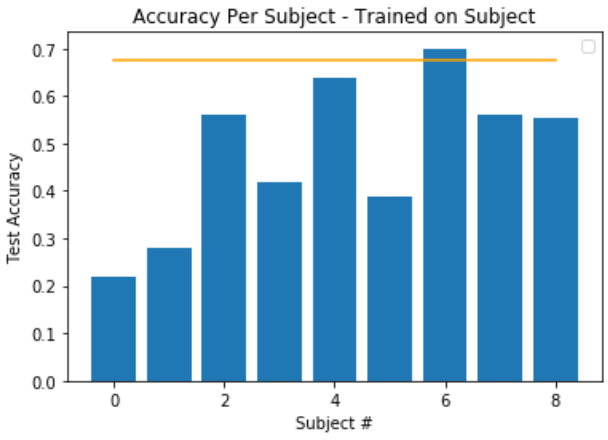

Figure 1.

2.3. Multiple subject training transfer to individuals

This study indicates that a model trained on the EEG data of multiple subjects can preserve – and in most cases exceed – the performance of the joint test evaluation when evaluated on subjects directly. Figure 2. shows the performance of the best performing model, Shallow CNN, and its performance across all subjects simultaneously (0.6749) in orange and the performance of each of the 9 subjects as bars.

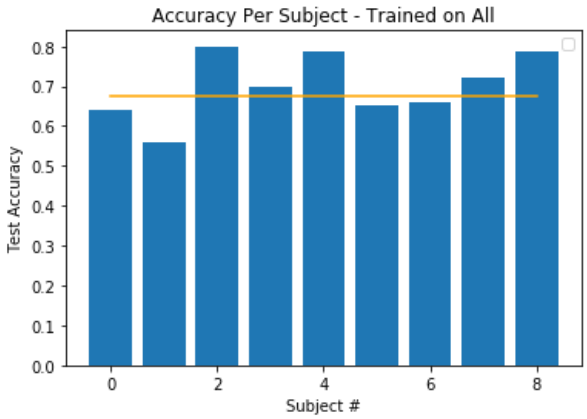

Figure 2.

2.4. Accuracy as a function of time

This experiment was conducted with a single layer LSTM model to determine whether accuracy generally increases of decreases as the length of the trial increases. The number of time bins was increased from 100 to 1000 in steps of 100, resulting in 10 trials that show a relatively flat correlation of -0.08 between accuracy and the number of time bins considered during training. Figure 3. displays the erratic relationship.

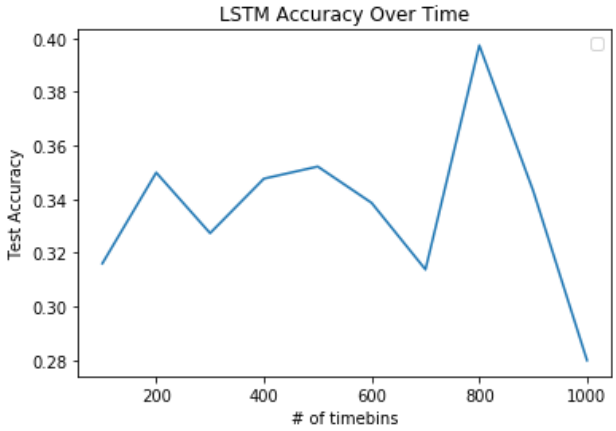

Figure 3.

3. Discussion

This study yielded some rather unintuitive results related to what models perform best with EEG data. The original hypothesis that RNNs would perform better overall was overturned considering the strong performance of CNNs with the special input block designed specifically for EEG timeseries. We can still see some support for the original hypothesis in the fact that training accuracy was highest amongst all the models, but this did not translate into a high accuracy on the unseen test set. This suggests a greater capacity to overfitting within RNNs and a highly irregular decision boundary for the 4 target classes. This propensity to overfit was noticed early on in my experiments and attempts to regularize the models by introducing dropout layers in between the stacked RNN layers and increasing the dropout rate did not significantly alleviate the problem. Overall, the CNN components do a better job of extracting the important information from the underlying data, as evidenced in the superiority of the pure CNN models and in the boost to performance when the CNN components were placed in the beginning of the Hybrid LSTM + CNN model.

Regarding the research questions of single subject training vs. multiple subject training and transferability, the results strongly suggest that there are overarching patterns that exist across the EEG spectrums of multiple people that help make model performance more robust for classification tasks. One of the challenges of using EEG data for real-time functionality is the extended calibration times needed to customize the decoder to the nuances of a particular subject's brain activity [2]. However, this issue is significantly mitigated by the success of cross-subject joint training and performance is improved by an average of 0.2209 across all subjects. See Table 2. for more details.

Finally, regarding the research question concerning accuracy as function of the length of the trial (i.e. the number of time bins), it was rather surprising to see that there wasn't any significant correlation. One would expect that the more data available, the more likely we are to learn the patterns that indicate some action over another, but having just the first 0.4 seconds of the trial was about as good as having all 4 seconds of it. This is especially bizarre considering that the cue for the target action hadn't actually been given at this point.

As for the task of hyper parameter optimization, most of the time and effort went into permuting through the many options available in Keras for LSTMs and GRUs. Since the primary issue related to them was overfitting, multiple units were tested in the range of 16 to 1000 and any count less than 20 tended to stunt learning enough for the model to underfit. Unfortunately, there was a quick swing to overfitting and unwieldly training times, so a balanced value of 50 was chosen for all RNN components. Essentially all of the settings in Section 6. Methods were selected via a time intensive grid search over at least 4 alternatives (e.g. for optimizers, Adam, Adamax, Nadam, and RMSProp were considered) and were tested directly against LSTMs. The selections made for LSTMs were held constant for GRUs, which might be contributing to their slightly lower performance relative LSTMs.

4. Future Work

One particular line of future research would be to adapt a Transformer network [4] to the task of timeseries analysis. A lot of attention within the machine learning community has been given to transformers since the work in [4] was shown to be comparable to Google's state-of-the-art language translation services while requiring significantly less training time and memory overhead. This is possible due to the "attention" mechanism it employs, which allows the network to keep track of important signals and use them to provide context and disambiguate subsequent decoding steps. Since EEG data is temporally dependent and it's likely that past signals are better understood in the context of nearby waveforms, investigating this novel application of it promises to be fruitful. [1]

References

[1] Schirrmeister, Robin Tibor et al. "Deep Learning with Convolutional Neural Networks for EEG Decoding and Visualization." Human Brain Mapping 38.11 (2017): 5391–5420.

[2] Halme, Hanna-Leena & Parkkonen, Lauri. (2018). Across-subject offline decoding of motor imagery from MEG and EEG. Scientific Reports. (2018) 8. 10.1038/s41598-018-28295-z.

[3] Tangermann, Michael et al. "Review of the BCI Competition IV" Frontiers in neuroscience vol. 6 55. 13 Jul. 2012, doi:10.3389/fnins.2012.00055

[4] Vaswani, Ashish & Shazeer, Noam & Parmar, Niki & Uszkoreit, Jakob & Jones, Llion & N. Gomez, Aidan & Kaiser, Lukasz & Polosukhin, Illia. (2017). Attention Is All You Need.

[1] This paper was initially going to pursue this line of research, but time constraints necessitated a more orthodox analysis.

5. Performance

5.1. Performance of all classifiers on all subjects.

Model Name	Layers	Test Loss	Test Accuracy	Final Train Loss	Final Train Accuracy	Diff	Overfitting[2]
Shallow CNN	3	0.8532	**0.6749**	0.4608	0.8310	0.1561	Medium
Deep CNN	6	0.9824	**0.6095**	0.6374	0.7512	0.1417	Small
Single LSTM	2	1.5346	0.3882	0.4679	0.8333	0.4451	High
Stacked LSTM	4	1.3889	0.3363	0.3035	0.9279	0.5916	High
Single GRU	2	1.4581	0.3273	0.8256	0.6720	0.3447	High
Stacked GRU	4	1.3922	0.3025	0.4144	0.8788	0.5763	High
Hybrid CNN-LSTM	7	1.3270	0.4176	0.9438	0.6105	0.1929	Medium

Table 1.

5.2. Performance of Shallow CNN on each subject.

Subjects	Test Loss	Test Accuracy	Final Train Loss	Final Train Accuracy	Diff	Overfitting	Joint Test Loss	Joint Test Accuracy	Improvement
1	1.5213	0.2200	0.7111	0.7316	0.5116	High	0.9719	0.6400	**0.4200**
2	1.4580	0.2800	0.6435	0.7238	0.4438	High	1.0671	0.5600	0.2800
3	1.1075	0.5600	0.0982	0.9734	0.4134	High	0.5925	0.8000	0.2400
4	1.3917	0.4200	0.5908	0.8043	0.3843	High	0.7485	0.7000	0.2800
5	0.8718	**0.6383**	0.1598	0.9632	0.3249	High	0.6068	0.7872	0.1489
6	1.4001	0.3878	0.9508	0.601	0.2132	Medium	0.8705	0.6531	0.2653
7	0.7584	**0.7000**	0.213	0.9239	0.2239	Medium	0.7185	0.6600	-0.0400
8	1.2915	0.5600	0.3541	0.8912	0.3312	High	0.7317	0.7200	0.1600
9	0.9105	0.5532	0.1066	0.9886	0.4354	High	0.7791	0.7872	0.2340

Table 2.

[2] Overfitting is estimated empirically as follows: if (train error – test error) < 15% = small, if < 25% = medium, else High.

6. Methods

All neural network architectures were implemented in Tensorflow with Keras and is publicly available at https://github.com/fabriceyhc/EC-239AS. The following configurations apply to all evaluations by default:

- Epochs: 50
- Batch_size: 16 (or 1 for stateful LSTM)
- Early Stopping: True
  - Monitor: val_acc
  - Min_delta: 0.005
  - Patience: 10
  - Restore_best_weights:True
- Splits: StratifiedShuffleSplit (test_size:0.2)
- Kernel_initializaer: glorot_normal (Xavier)
- Cropping: False[3]
- Scaling: True (minmax)
- Smoothing: True (Butterworth, order:3)
- Activations:
  - ConvNets: ELU
  - RNNs: tanh
- Optimizer:
  - ConvNets: adam
  - RNNs: RMSProp
  - Learning Rate: (default)

6.1. ConvNets

6.1.1 Deep CNN

The Deep CNN architecture was inspired by [1]. The architecture consists of four convolutional-pooling blocks with 25, 50, 100, and 200 filters (feature maps) in each layer, respectively. Each block also features a batch normalization layer and ELU activation between the convolution and max pooling, which has a pool size and stride of 3x1. Finally, each block is concluded with a dropout of 0.5. Then an affine soft-max classification layer outputs the model's prediction.

One particularly notable aspect of this architecture is that the specially designed (for EEG data) input block that contains two Conv2D layers – one for processing data across time (kernel_size=10x1) and the other for processing it across all electrode channels (kernel_size=1x22) – with no activation function in between them, which implicitly regularizes our objective function.

6.1.2 Shallow CNN

The Shallow CNN architecture was also inspired by [1], which was itself inspired by the FBCSP pipeline. The architecture consists of the same special input block as the Deep CNN, but because there is only one convolutional block, increases the filters to 40 and uses a larger kernel_size of 25x1 in the time convolution. Batch normalization and ELU activation precede an average pooling layer with a relatively large size of 75x1 and stride of 15x1. As before, dropout concludes the convolutional block. At this point the data is flattened and passed into an affine soft-max classification layer.

6.2. RNNs[4]

All LSTM and GRU cells were set to have 50 units by default based on hyperparameter evaluation of 16, 20, 25, 50, 100, 150, 200, 500, and 1000 units, judging for speed and accuracy.

6.2.1 1-Layer / 3-Layer LSTMs

Two models that consist of a single LSTM layer in the first case, and three in the second, both connected to an affine soft-max classification layer. State was not maintained between layers (stateful: False) for computational cost reasons, even though it was noted to contribute 2-3% improvements in small scale tests.

6.2.2 1- Layer / 3-Layer GRU

Two models that consist of a single GRU layer in the first case, and three in the second, both connected to an affine soft-max classification layer. Again, state was not maintained between layers (stateful: False).

6.2.3 Hybrid 3-Layer LSTM with 2 Conv2D Blocks

This model contains an input convolutional block similar to those used in the ConvNets, but with a larger number of feature maps and smaller kernels. This is followed by a second convolutional pool block with 50 more filters and otherwise the same setup as the Deep ConvNet blocks. The output is then reshaped for 3 stacked stateless LSTM layers with 50 units each before concluding in an affine soft-max classification layer.

[3] Cropping added about 4-6% performance increase on average, but at the cost of 50x slower training. Due to limited computational resources, cropping was not performed in any of the standard model analyses.

[4] Other configurations were evaluated, but performance was worse than random guessing and are therefore omitted. However, these configurations explore one and two single dimensional convolutions prior to LSTM layers.