Performance of Long Short-Term Memory (LSTM) and Multi-Headed Self-Attention (MHSA) layers with Convolutional Neural Networks (CNNs) in EEG Decoding

Anonymous CVPR submission

Paper ID

Abstract

We implement the vanilla CNN, CNN+MHSA, and CNN+LSTM architectures for the EEG decoding task, and train these models on the data of subject 1 and all subjects to evaluate their relative performances. The testing accuracies of the vanilla CNN and CNN+MHSA achieve the desired 70% threshold, but CNN+LSTM does not. In addition, we analyze the models with respect to classification accuracy as a function of time. Given a training time limit of 100 epochs, we find that CNN+MHSA requires 77 epochs, the longest among the three models, to reach testing accuracy of 70%, with its performance possibly continuing to increase after epoch 100. In contrast, the vanilla CNN converges to the testing accuracy benchmark of 70% early on epoch 36, whereas CNN+LSTM fails to even reach the threshold by the epoch 100 cutoff. Additionally CNN and CNN+LSTM models both show early significant divergences between training accuracy and validation accuracy.

1. Introduction

In this report, we investigate the relative advantages and disadvantages of several deep learning algorithms on the EEG decoding task, where the EEG signals have to be resolved on a spatio-temporal basis.

1.1. Convolutional Neural Networks (CNNs)

An advantage of CNNs is that they perform well in image classification tasks, strongly capturing the short-range spatial dependencies of input images via convolutional layers [3]. Another advantage is that CNNs can downsample input data (via pooling layers), potentially reducing processing costs [2,3]. A major disadvantage, however, is that a vanilla CNN architecture may not efficiently capture temporal dependencies in data [3].

1.2. Long Short-Term Memory (LSTM) networks

An alternative algorithm that would better capture temporal dependencies is the Long Short-Term Memory (LSTM) architecture, which is a widely-used recurrent neural network (RNN) architecture [4,5].

RNNs learn by recurrently processing sequential input data (e.g. text strings or video), storing previously computed past information in the current state via the Markov property [4,5]. A disadvantage of LSTMs (and RNNs in general) however is that the temporal dependency range they are able to learn may be limited, with the range of input tokens only strongly influencing the next few nearest tokens and dying off quickly for more distant embeddings [6]. To enable earning of long-term dependencies, LSTMs utilise gate functions or require weights for long-term dependencies to be learnt to be higher [5,6].

1.3. Transformers and Multi-Headed Self-Attention (MHSA) layer

A major improvement of transformers is that they avoid recurrence entirely—instead transformers process the entire input dataset collectively in one pass [7]. This allows for parallel processing, leading to significantly faster training times versus RNNs. This also explains why transformers do not face limitations in long-term dependency learning as in RNNs, by avoiding recurrently storing past information in hidden states [7,8].

Transformers utilize a “self-attention” layer, which computes the self-similarity scores between tokens. Multiple self-attention layers in transformers have been shown to be able to capture local n-gram-like contexts as well as global, long-term dependencies [7,9].

We focus on the implementation of the multi-headed self-attention (MHSA) layer used in transformers, which computes the self-similarity of the tokens using several parallel attention layers [7,9]. MHSA allows for faster learning of richer structural relations between tokens and hypothetically offers better learning of long-term dependencies.

1.4. EEG Dataset Visualization and Preprocessing

The EEG dataset has 2115 trials; with each trial containing EEG data from 22 electrodes over x=1000 time-bins/time-steps. When visualizing the average representations of the 9th channel, we note that from x=500 to x=1000, the plots of the 4 class labels overlap each other randomly and are difficult to resolve. However from x=0 to x=500, the plots are better distinguishable, so we trim the dataset from x=1000 down to x=500 time-steps.

The trimmed dataset was further preprocessed for data augmentation, using Maxpooling (stride=2), Average-pooling (stride=2) + Noise and lastly, Sub-sampling (stride=2) + Noise twice. This yields 4 distinct datasets (each of x=250 time-steps) from a single trimmed dataset. The dataset dimensionality reduction reduces computation costs, while the injection of Gaussian noise enhances robustness in our model learning process against overfitting.

1.5. Algorithm Model Choice and Architectures

As noted in Section 1.2, a vanilla CNN may efficiently learn spatial but not necessarily temporal dependencies [3]. We select a CNN architecture as the base of our algorithm architectures due to its strength in spatial learning. We also augment the CNN backbone with additional layers to better learn long-term temporal dependencies, incorporating elements from LSTMs and MHSA layers from transformers. The proposed "post-CNN" algorithms, namely CNN+LSTM and CNN+MHSA architectures, are subsequently evaluated against a vanilla CNN benchmark. The different model architectures, including relevant design and hyperparameter choices, are summarized in the Appendix.

2. Results

We evaluate and compare the presented algorithms across three rubrics (1) How optimization for a single subject behaves when used against all subjects (2) How optimization behaves when we use all subjects as opposed to any one subject and (3) How each model performs in terms of a comparison of accuracy as a function of time. Detailed graphs of the results are provided in the Appendix.

2.1. Optimizing the Classification Accuracy for Subject 1

The results for Section 2.1 are summarized in Tables 1 and 2 (see above).

Testing accuracy of the CNN and CNN+MHSA models on subject 1's data when they are trained on subject 1's data (CNN: 67.5%, CNN+MHSA: 60.5%), and their testing accuracy on subject 1's data when they are trained on all subjects' data (CNN: 64%, CNN+MHSA: 61%), are very similar and lay within a ± 3.5% range.

Model	Accuracy
CNN	67.5%
CNN + MHSA	60.5%
CNN + LSTM	64.0%

Table 1. Results of testing accuracies of models trained on subject 1 and evaluated on subject 1.

Model	Accuracy
CNN	64%
CNN + MHSA	61%
CNN + LSTM	54%

Table 2. Results of testing accuracies of models trained on all subjects and evaluated only on subject 1.

However, the testing accuracy of CNN+LSTM model on subject 1's data when it is trained on subject 1's data (64%) is significantly higher than its testing accuracy on subject 1's data when it is trained on all subjects' data (54%), with a significant 10% difference.

When all three models are trained on subject 1's data, their validation accuracies are very high, exceeding 95%. This is probably due to the fact that the size of subject 1's dataset is very small, resulting in a correspondingly small validation set. Moreover, we note that the models' training accuracies are very high as well. All three models clearly overfit when they are trained on subject 1's data.

2.2. Optimizing the Classification Accuracy Across all Subjects

When the three models are trained on all subjects and evaluated across all subjects, CNN has the highest test accuracy of 71%. CNN+MHSA has an accuracy of 70%. CNN+LSTM performs the worst, with an accuracy of only 64%. The training accuracy of CNN+LSTM is 94% while the validation accuracy is only 61%.

2.3. Classification Accuracy as a Function of Time

The results for Section 2.3 are summarized in Table 3.

We use inference step, which is the measure of how long it takes for our model to operate on a single batch from the dataset, as an evaluation metric. This clues us in to the overall computational complexity of the network as well as provides us an approximation for the wall time in deployment. Across our models we found that the vanilla CNN architecture had the lowest average inference step at 9 milliseconds on our hardware, CNN+MHSA the second lowest at 11 milliseconds and CNN+LSTM the longest at 204 milliseconds.

Model	Inference Time (ms)
Epoch with 70% Accuracy	Highest Accuracy
CNN	9
36	71%
CNN + MHSA	11
77	70%
CNN + LSTM	204
N/A	64%

Table 3. Results for the models' classification accuracy as a function of time (100 epoch limit). The models were trained and evaluated on all subjects. It is noted that the CNN model yields higher accuracy in fewer epochs, whereas the CNN+LSTM model failed to reach the 70% threshold by the epoch 100 cutoff.

We expect that this performance is hardware agnostic.

Each model was given a limit of 100 epochs of training time which we selected empirically, observing that each network would tend to reduce the loss to near 0 on the training set within this timeframe (with the notable exception of CNN+MHSA which likely could have been trained longer). CNN+MHSA required the largest number of epochs to reach our desired testing accuracy of 70% which occurred on epoch 77. The vanilla CNN reached the desired result on epoch 36 and CNN+LSTM failed to reach the 70% threshold in testing by the end of the epoch 100 limit.

3. Discussion

3.1. Architecture

We included batchnormalization layer in our CNN model to ensure zero mean and unit variance, which makes the whole training process stabler. Dropout layer is included to make the model generalizes well. We also found out that CNN generally yields better performance if we increase the number of filters of convolutional layers as the CNN goes deeper. We used a batch-normalization layer which appeared to make the training process more stable.

3.2. Model Selection and Comparison

Our results indicate that overall the model architectures yield approximately equivalent results between the vanilla CNN and CNN+MHSA with respect to time under the provided conditions, but given a more stringent testing accuracy requirement CNN+MHSA would likely be the best choice. This is largely a question of computational resources versus testing performance tradeoff: if the task was purely to achieve 70% testing accuracy, the CNN would likely perform marginally better than CNN+MHSA.

However, if computational time were not a major limitation, allowing the CNN+MHSA model additional training time would likely eventually yield significantly better results. There are some specific considerations to make, however—while CNN's performance was superior in inference step and wall time, it exhibited significantly higher divergence between the training and validation accuracy when compared to the CNN+MHSA (divergence refers to the scenario that the training accuracy increased but the validation accuracy remained the same or grew at a slower rate). More particularly the validation performance did not improve with time after epoch 36 for the CNN but the performance of CNN+MHSA may improve past the epoch 100 cutoff.

This may be indicative that the CNN was overly expressive and should have had fewer trainable parameters and that with more time CNN+MHSA would surpass the CNN in performance (at the cost of training time duration).

CNN+LSTM peaked at around 64% accuracy on epoch 25 and did not show improvement in continued training, instead exhibiting the widest divergence between training and validation accuracy. We suspect that this is because LSTM layers are very prone to overfitting - the smaller dataset we are presented with, coupled with the CNN architecture backbone that we already noted was overly complex, likely made the CNN+LSTM architecture a poor fit for the task.

3.3. Limitations and Future Directions

The major limiting factor of our research is a lack of access to suitable hardware (i.e GPUs) for training significantly more dense or computationally expensive models and a lack of time for solving the problem for more complex grids of parameters. We suspect that our solutions are effective solvers for the dataset, but that with a much large parameter grid or the training capacity to include MHSA layers earlier in the network we might have yielded substantively better results. Additional data (and by extension additional training time) would also likely have further improved performance.

In the future, we would have liked to continue exploring the CNN+MHSA model, especially given that it showed the least propensity to overfit. Future extensions to this project may have taken the form of additional epochs as well as a more sophisticated set of callbacks like cosine learning rate decay. Additionally, we would have liked to have expanded the subject 1 to all subjects analysis to a permutation of subjects that we could average over to see what the typical performance of subject x to all subjects was. The subject 1 to all subjects performance suggests that the average result might be slightly better than training with all subjects.

References

[1] Alzubaidi, L. et al. (2021). ”Review of deep learning: concepts, CNN architectures, challenges, applications, future directions.” *J. Big Data*. vol. 8, no. 53.

[2] An, S. et al. (2020). ”An Ensemble of Simple Convolutional Neural Network Models for MNIST Digit Recognition.” https://arxiv.org/abs/2008.10400

[3] Wang, S., Cao, J. and Yu, P.S. (2020). ”Deep Learning for Spatio-Temporal Data Mining: A Survey.” *IEEE Trans. Knowl. Data Eng.* vol. 34, no. 8.

[4] Hochreiter, S., and Schmidhuber, J. (1997). ”Long short-term memory.” *Neural Computation.* vol. 9, no. 8, pp. 1735–1780.

[5] Yu, Y., Si, X., Hu, C. and Zhang, J. (2019). ”A Review of Recurrent Neural Networks: LSTM Cells and Network Architectures.” *Neural Computation*. vol. 31, no. 7, pp. 1235-1270.

[6] Sherstinsky, A. (2021). ”Fundamentals of Recurrent Neural Network (RNN) and Long Short-Term Memory (LSTM) Network.” *Physica D: Nonlinear Phenomena.* vol. 404, 2020.

[7] Vaswani, A. et al. (2017). ”Attention is All You Need.” *Adv. in Neur. Info. Proc. Sys. 2017.* pp. 6000-6010.

[8] Sheikh, I., Vincent, E. and Illina, I. (2022). ”Transformer versus LSTM Language Models trained on Uncertain ASR Hypotheses in Limited Data Scenarios.” *LREC 2022*. pp. 393-399.

[9] Irie, K., Zayer, A., Schlüter, R. and Ney, H. (2019). ”Language Modeling with Deep Transformers.” https://doi.org/10.48550/arXiv.1905.04226