Decoding Brain MEG Signals Using Deep Neural Networks

Parham Oghabi, Adam Khazi, Jacob Cohen-Setton

Abstract

Brain MEG and EEG signals are very complex high dimensional spatiotemporal data which contain a lot of noise. In this project, we have attempted to analyze MEG signals using various types of deep neural networks. Our motivation stems from the observation that historically SVMs and random forest models were applied to extracted discriminative features of MEG signals and have produced fairly mediocre results. Our project indicates that deep neural networks have a great potential in analyzing and decoding MEG data.

1. Introduction

Being able to understand how the brain works is of huge interest to neuroscientists and one approach is by analyzing neuroimaging techniques such as MEGs which record brain activities as multiple time series data. In our project, different subjects were shown a stimulus (2 different pictures in our case) and their concurrent brain activity was recorded using 306 sensors. The goal of this project is to decode the MEG recordings, which is to predict the category of the stimulus from the recorded brain activity. Pictures of the stimuli are shown below and the two categories to be predicted are "Face" and "Scrambled Face", which makes this a binary classification problem.

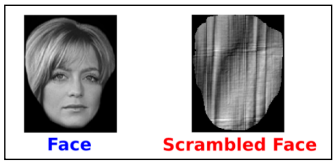

Figure 1. The two categories of the visual stimuli which were presented to the subjects and which had to be predicted by our model by analyzing MEG training data.

Deep Neural Nets have shown remarkable performance on complex data such as image (spatial) and speech recognition (temporal) problems with the help of its layer-wise non-linear activations. Conversely, it can be said that bioinformatic problems relating to brain activity are still being tackled with SVMs and random forest models and have not yet witnessed a deep learning breakthrough. While it is true that deep networks don't hold the best results in this area, the margin of improvement by feature engineering and SVMs is not very impressive and non-deep methods require complex feature extraction methods. In this project, we challenge this view and attempt to decode the MEG signals by applying various deep learning models.

2. Dataset and Difficulties

The training data consists of 9414 MEG recordings from 16 people who were shown the "Face" and "Scramble Face" visual stimuli multiple times (roughly 588 trials per subject). The test set comprises of 4058 MEG recordings from 7 other different people. For each trial the brain activity was recorded by 306 sensors at a sampling rate of 250Hz lasting for 1.5 seconds (starting from 0.5 seconds before the visual stimuli was shown). Therefore, each subject data has a 3D data matrix (trial number x channel x time) of size 588 x 306 x 375. A visual description of the data is shown below.

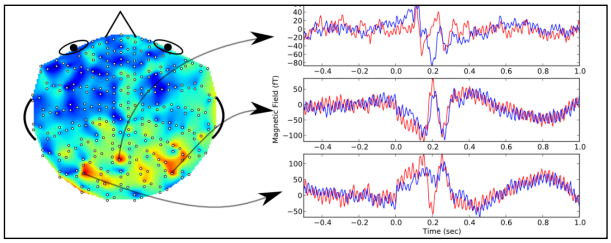

Figure 2. A visual description of the MEG recordings by each of the 306 sensors over the 1.5 second period. At each sensor location, two orthogonal gradiometers and one magnetometer record the magnetic field induced by the changing electric fields of the neuron currents.

While MEG is much more accurate than the historical EEG, they are still associated with a lot of inaccuracies. MEG includes a large amount of non-Gaussian noise since the 306 sensors pick up all brain activity, even those which are not related to the brain parsing the visual stimuli. For example, if the subject thought of something random or moved their hand during the 1.5 second period of being shown the visual stimuli, the sensors would pick up on the signals induced by those brain neurons. This can skew the results and reduce the accuracy of the models and finding a way to filter out the noise can be quite difficult. In addition to that, MEG signals are very subject-specific since the sensors will pick up different signals for different orientations of the cortical dipoles.

Also, the number of features greatly outnumber the number of training examples. For example, when we flattened the 2D matrix, we only had 9414 trials but 76,500 features for each trial.

Due to the reasons mentioned above, it is very challenging to build a generic classification model that works on all subjects and historically subject-specific models have been favoured. However, in this project we have attempted the generic classification model.

3. Data Pre-Processing

The first 0.5 seconds of the 306 channels were discarded and only the MEG signals from t=0.5 to t=1.5 seconds were used. This is because the visual stimuli was shown at t=0.5 and all information prior to that contained a lot of subject-specific noise.

Therefore, this would allow some of the noise to be discarded and resulted in the input being reduced to a 3D matrix of 588 x 306 x 250 for each person.

The data was also scaled up and normalized to have a mean of 0 and standard deviation of 1. This would allow gradient descent to converge faster.

PCA dimensionality reduction was also applied to the input data for some of the models in order to eliminate the noise. Since the noise for the MEG signals is non-gaussian, PCA was incapable of modelling the non-linear manifolds during dimensionality reduction but it did help in slightly improving the final accuracy. Stacked auto-encoders were attempted in order to learn low-dimensional, discriminative representations of the MEG signals but implementation was unsuccessful.

4. Experiments

In this section, we present a brief analysis of the accuracies we achieved when decoding MEG signals with different deep neural models. As primary benchmarks for evaluation comparison, we used the top 2 accuracy scores in Kaggle's Brain MEG decoding competition, all of which were achieved using non-deep neural models such as SVMS, Logistic Regression, and Random Forest.

4.1 Testing Environment and Framework

Due to the large size of the dataset (8 GB), all training and inference had to be run in the cloud. A python script was written to fetch the data from Amazon's S3 bucket each time a new EC2 instance was launched. EC2's GPU optimized p2.xlarge (4 vCPU and 61 GB RAM) were used for all training and inference purposes.

With regards to the code and commonality between the models, Keras was used to implement them. Adam was also used to minimize the categorical cross-entropy loss. All models were trained between 25-50 epochs to predict whether the subject was viewing the real face or the scrambled face image. A 20-fold cross validation set used to optimize the hyper-parameters.

4.2 Multi-Layer Perceptron

The first model that was attempted was a 3 layer multi-layer perceptron with 512 hidden units in each layer. Each layer was also followed by dropout layer with a probability of 0.5 to introduce regularization. ReLU was used as the activation function and all kernels were initialized with "He uniform". Both the pre-processed data and the PCA dimensionally reduced pre-processed data were used with the latter resulting in a slightly higher accuracy. The results were quite satisfactory for such a simple model as shown in below in Table 1.

It was advantageous to apply dimensionality reduction techniques such as PCA as it was able to reduce some of the non-Gaussian noise however using a stacked auto-encoder might have performed better for dimensionality reduction since it is capable of capturing non-linear manifolds. This was attempted but we were unsuccessful in implementing it. PCA is not as robust to non-Gaussian noise as other techniques. Adding more layers or neurons did not improve the test set or validation set accuracy.

Table 1. Test accuracies for the multi-layer perceptron both with and without PCA applied on the pre-processed data. We can see that PCA helped in reducing the noise and slightly increasing the accuracy.

Architecture	**Test Accuracy**
MLP with Dimensionality Reduction	66.5%
MLP without Dimensionality Reduction	65.7%

4.3 Convolutional Neural Network

A convolutional neural network followed by a multi-layer perceptron was attempted but the accuracy was not as high as the MLP. The architecture was as follows: 3 convolution layers, each with a 5 x 5 kernel and a max pooling layer, followed by 3 dense layers with 256 neurons in each layer. "He uniform" kernel initializer was used for all the layers since ReLU activation was used.

The fully connected layers were used to extract low dimensional representations. The CNN was used to extract local features to build hierarchical representations but a 5 x 5 kernel was capable of only finding local patterns and not able to exploit the 306 channel data. An accuracy of 58% was achieved. CNNs were expensive to train and took around 7 hours for 30 epochs of training.

4.4 Long Short-Term Memory

LSTM models were also applied in order to exploit the temporal patterns of the time series data and performed better than CNNs but still not as good as MLPs. Two various LSTM models were implemented. A time window of 250 was used for both LSTMs which span the entire 1 second in which the subject was shown the visual stimuli.

The first was a 3 layer LSTM (64 hidden units in each layer) followed by 2 Dense layers (128 hidden units in each layer).

The second was a 4 layer bi-directional LSTM (128 neurons in each layer) followed by 2 Dense layers each wrapped with concrete dropout.

Brain MEG signals can also be read in reverse and therefore a bi-directional LSTM seems to be more suitable to extract pattens and concrete dropout was used since the probability hyper-parameter can now be learnt with gradient descent since it is differentiable and need not be tuned manually.

Table 2. Test accuracies for an LSTM compared to a Bi-directional LSTM. The Bi-directional LSTM was slower to train but performed much better on the test set.

Architecture	**Test Accuracy**
4 Layer Bi-Directional LSTM Followed by 2 Dense Layers Wrapped With Concrete Dropout	61.1%
3 Layer LSTM Followed by 2 Dense Layers	59.5%

5. Comparison of Results and Discussion

Table 3 below lists the test accuracies of all the models we have tried and the top two test set accuracies achieved by competitors in Kaggle. The top two accuracies in Kaggle were achieved using some combination of logistic regression, random forest models, and SVMs and not using deep learning models. In addition to that, various complex forms of feature engineering was also used by them.

We can see that the simple 3 layer MLP with PCA dimensionality reduction applied to the pre-processed data works best and that the margin of improvement using non-deep learning models by Kaggle competitors is not that great.

Table 3. The test set accuracies of the various deep models compared to the non-deep models used by the top 2 competitors on Kaggle show that the margin of improvement is not that great over deep models.

Architecture	Test Accuracy
MLP with Dimensionality Reduction	66.5%
MLP without Dimensionality Reduction	65.7%
4 Layer Bi-Directional LSTM Followed by 2 Dense Layers wrapped with concrete dropout	61.1%
3 Layer LSTM Followed by 2 Dense Layers	59.5%
CNN Followed by 3 Dense Layers	58%
1st Position on Kaggle (Non-deep model)	75%
2nd Position on Kaggle (Non-deep model)	72%

Convolutional neural nets could not take advantage of the temporal information of the data and therefore achieved the lowest accuracy. LSTMs have cell states and can exploit this information and therefore resulted in a slightly higher accuracy but the noise in the data was quite high which prevented the model from getting high accuracies.

With all the deep models, we were able to overfit the training data and achieve up to 99% training accuracy. However, the validation set accuracy peaked at 71% and the test set accuracy peaked at 66.5%. We assume that they plateau since the optimization stops at a local minima or a saddle point.

Unfortunately, we did not have the domain knowledge to apply complex feature engineering and de-noising methods to the data but one would assume that using both feature extraction and better data pre-processing with deep models would result in a higher accuracy. Other dimensionality reduction methods like stacked auto-encoders and TSNE might also perform better by reducing the noise and increasing the SNR (signal to noise ratio) more than PCA did.

In addition to that, complex models useful for MEGs takes a long time to train due to the large number of features and due to the limited time that we had, elaborate models with more than 5-6 layers could not be attempted. However, our results shows that deep learning has huge potentials in brain decoding.

6. Conclusion

In this project, we have presented a brief study of different deep learning models for decoding brain MEG signals in order to predict the visual stimuli observed by the subject and their respective accuracies. Our results show that a 3 layer multi-layer perceptron with 512 neurons in each layer achieves satisfactory results compared to the SVM and other non-deep models used to achieve the top 2 accuracy scores on Kaggle. Whilst our study, models, and results were that of a novice, one can clearly see the potential of deep learning in the bio-informatics industry. As of now, many of the datasets are kept private due to the high cost of collecting brain MEG recordings and due to subject privacy concerns. In the future, as larger datasets become open-sourced like ImageNet, deep learning models can learn more features from them and lead to significant improvements in brain decoding.

References

(n.d.). Retrieved December 06, 2017, from https://www.kaggle.com/c/decoding-the-human-brain/data