Final Project Report: Capacity Estimation of Convolutional Neural Networks for Classifying Motor Imagery Movements

Susan Hao and Vilde Roland Arntzen

University of California, Berkeley

1 Introduction and Background

The experimental design of machine learning models and neural networks provide solutions to problems ranging from facial and speech recognition to classification tasks within an environmental control unit for disabled patients. However, one critical aspect of this design that is often overlooked is model and network capacity and its relationship to the data's capacity. In many instances where deep learning is used, models are not tailored to the data. That is, models may have a much larger capacity than that of the data that is used for training. In these instances, the model is ultimately overfitting and not capable of generalizing to anything outside of the training data.

One field using deep learning that may be particularly susceptible to the fallacy described above is the field of brain-computer interface (BCI). BCI aims to decode human neural data using supervised learning methods such as convolutional neural networks, recurrent neural networks, and support vector machines. However, the amount of data that is acquired from humans is often far less than the number of parameters in these models. In our experiment, we aim to examine whether data acquired from an EEG-based BCI experiment [7] is prone to overfitting when using deep learning models such as convolutional neural networks to decode neural activity.

1.1 EEG Based BCI

Electroencephalography (EEG) is a non-invasive technique that measures electrical activity of the brain through electrodes placed on the scalp. More specifically, EEG records voltage fluctuations which result from ionic current within neurons near the scalp of the brain [6]. In the past decade, researchers in the field of BCI have been utilizing EEG as a way to decode human mental processes. Tayeb and colleagues [7] utilized EEG to decode a motor imagery task. Subjects were instructed to imagine left or right arm movements as indicated by a visual left or right arrow. EEG data recordings were then decoded using a variety of deep learning and classical methods of EEG processing.

1.2 Our Experiment

This project examines the generalization capacity of selected deep learning neural networks that were used in Tayeb and colleagues' experiment when solving the task of classifying motor imagery movements into left and right hand movements. The neural networks examined from the experiment are: Pragmatic Convolutional Neural Network (pCNN) and

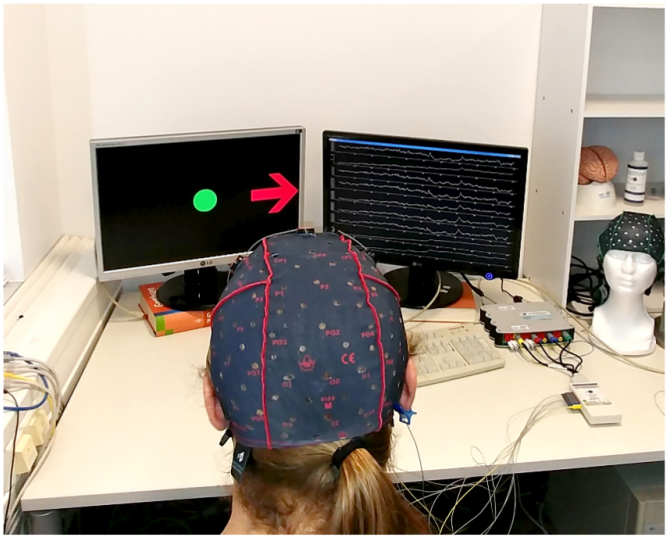

Fig. 1: Tayeb et al (2019) experimental setup. Subjects were instructed to fixate at the center of the screen on the green dot. Subjects were then given a visual cue that indicated which hand movment to imagine 4 seconds.

Shallow Convolutional Neural Network (sCNN). In addition to understanding the generalization capacity, we seek to investigate how Tayeb and colleagues were able to produce such high validation accuracy results for each subject even when their models were prone to overfitting.

1.3 Data

In the final part of their experiment, Tayeb and colleagues used a publicly available dataset to further evaluate several of their classical and deep learning models. The Graz Data set B is a publicly available EEG-based BCI data set from a 2009 BCI Competition [5]. Tayeb and colleagues trained and evaluated their models on a within subject basis so that each model was trained and tested for each individual subject. Thus, we deemed it sufficient to restrict our analysis to a single subject, B01, from the Graz data set due to restriction of time and computational resources.

The number of instances of imagined left hand and right hand movements for B01 was balanced, thus indicating that there is little to no bias in the data set. Three bipolar electrodes (located above the motor cortex) were used out of a possible 32 electrodes with a sampling frequency of 250Hz. 324 trials were collected for subject B01 with each trial length lasting for 8 seconds with 4 seconds of imagery. As with all EEG-based experiments, there is low signal to noise so extra preprocessing steps were taken to augment the data. Please see Leeb and colleagues' paper [5] for a deeper description on the data and recording process.

1.4 Measurements

We explore the models' ability to generalize using the generalization capacity defined as

$$G = \frac{n}{MEC}, \tag{1}$$

where n is the number of correctly classified instances in the binary classifier and MEC is the memory-equivalent capacity. MEC is used to practically estimate the size of the neural network architectures on the given EEG training data set[1]. The MEC of a neural network is calculated using the following rules:

1. The output of a single neuron is maximally one bit[1].
2. The memory capacity of a single neuron is the number of parameters in bits [1].
3. The memory capacity of neurons is additive[1].
4. The memory capacity of neurons in a subsequent layer is limited by the output of the layer it depends on[1].

The neural network architectures considered, pCNN and sCNN, consist of a combination of convolutional layers and max/average pooling layers. To calculate the capacity of the convolutional layers, we considered the number of filters multiplied by the filter size. We then add the number of filters to the result because each filter has one bias term. After calculating the capacity of the convolutional layer, we can then calculate the activation size of each layer. If max or average pooling is applied, then the activation size of the convolutional layer is limited by the activation size of the pooling layer. The following input of the next layer is then restricted by the output of the previous layer (the minimum of the capacity and the activation size). Finally, following rule 3, we can add the capacities of all the layers to get the total MEC of the network.

Although capacity measurements are the main focus of this project, accuracy is used to measure the number of correct classified instances. From an experimental point of view, a classification accuracy that is significantly above 50% is reasonable as it is significantly above chance. Given the application of classifying movements for BCI, an accuracy of approximately 100% is required to be useful in a control unit for disabled patients. However, most EEG-based BCI experiments obtain an accuracy of 70-80% for two class classification[4].

1.5 Regularization

The pCNN and sCNN models produced in Tayeb and colleagues' paper apply dropout layers and batch normalization which regularize the neural networks. These methods reduce overfitting by preventing complex co-adaptions on the training data [3]. In the reproduction of models, we froze dropout layer so that it did not affect the models. However, as in the original experiment, batch normalization and early stopping were implemented to prevent overfitting. These methods reduce the amount of memorization and therefore contribute to an unknown amount of reduction in MEC.

The same effect can be seen implementing the models using the Keras library in Python. Keras was used in the original experiment, and is therefore also used to reproduce the pCNN and sCNN in this project. Keras uses a form of implicit regularization to avoid overfitting, meaning it is not able to fit to the data completely[2]. This regularization also reduces MEC by an unknown amount which is not captured in the MEC measurments.

There is not yet discovered an official method to measure the amount of reduction in MEC by the regularization effects mentioned. This topic is out of scope for this project and is therefore left for future research. However, to avoid the implicit regularization from keras, the implementation of the models could be done using another Python Library such as PyTorch that does not result in this effect.

2 Methods

2.1 Data prepossessing

The data was preprocessed as described in the original experiment. To remove power line interference, a notch filter at 50 Hz was applied. A band pass filter between 2Hz and 60Hz was then applied in order to remove baseline drift. Finally, a data augmentation was performed; using a time-window of 4s that resulted in different crops with a stride of 125ms which gave 25 new sub trials from each individual trial. Each time-window was shifted iteratively such that it overlapped with the previous time-window. This sliding-window augmentation allows for more data points per subject albeit the data points are now extremely correlated.

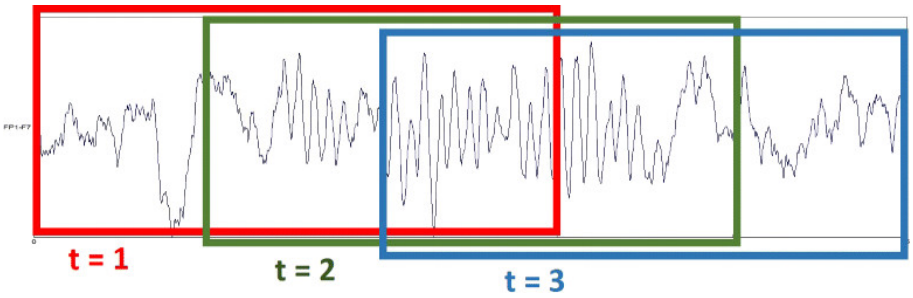

Fig. 2: Example of sliding window design for EEG data [8]. T represents a single time window.

2.2 Models

In this experiment, we implemented two deep learning models: shallow convolutional neural network (sCNN) and pragmatic convolutional neural networks (pCNN). Since we are convolving data in convolutional layers, we reduce the amount of information from the input which thus imply that the sCNN and pCNN are pattern matching models.

The sCNN comprised two convolutional blocks and one fully connected layer. The first convolutional block includes a 2D convolutional layer. The second block consists of a 3D convolutional layer, a batch normalization layer, an activation layer, and an average pooling layer. This is finally followed by a fully connected layer. Figure 3 shows the keras model summary output.

The pCNN comprised one spectrogram layer, three convolutional blocks, and one fully connected layer. The spectrogram layer converts the input into spectrograms. This is done by using a short-time Fourier transform (STFT) which is a technique borrowed from audio signal processing. After the spectrogram layer, there are three convolutional blocks that all

have a 2D convolutional layer, a batch normalization layer, an activation layer, and a max pooling layer. Finally, there is a fully connected layer. From figure 4, one can see that the architect of the pCNN is inefficient. The number of parameters increase as we go toward the latter convolutional layers. Following rule 4, we know that the the memory capacity in subsequent layers is limited by the previous layers it depends on. Thus, it is superfluous to have more parameters in subsequent layers compared to previous layers.

Layer (type)	Output Shape	Param #
conv2d_7 (Conv2D)	(None, 3, 976, 40)	1040
reshape_12 (Reshape)	(None, 3, 976, 40, 1)	0
conv3d_6 (Conv3D)	(None, 1, 976, 1, 40)	4840
batch_normalization_6 (Batch	(None, 1, 976, 1, 40)	160
activation_12 (Activation)	(None, 1, 976, 1, 40)	0
flatten_9 (Flatten)	(None, 39040)	0
reshape_13 (Reshape)	(None, 976, 40, 1)	0
average_pooling2d_4 (Average	(None, 61, 40, 1)	0
activation_13 (Activation)	(None, 61, 40, 1)	0
flatten_10 (Flatten)	(None, 2440)	0
dense_4 (Dense)	(None, 2)	4882
activation_14 (Activation)	(None, 2)	0

Total params: 10,922
Trainable params: 10,842
Non-trainable params: 80

Fig. 3: Keras model summary for sCNN.

Layer (type)	Output Shape	Param #
static_stft (Spectrogram)	(None, 65, 63, 3)	16640
normalization2d_1 (Normaliza	(None, 65, 63, 3)	0
conv1 (Conv2D)	(None, 65, 63, 24)	10392
batch_normalization_1 (Batch	(None, 65, 63, 24)	260
activation_1 (Activation)	(None, 65, 63, 24)	0
max_pooling2d_1 (MaxPooling2	(None, 32, 31, 24)	0
conv2 (Conv2D)	(None, 32, 31, 48)	73776
batch_normalization_2 (Batch	(None, 32, 31, 48)	128
activation_2 (Activation)	(None, 32, 31, 48)	0
max_pooling2d_2 (MaxPooling2	(None, 16, 15, 48)	0
conv3 (Conv2D)	(None, 16, 15, 96)	73824
batch_normalization_3 (Batch	(None, 16, 15, 96)	64
activation_3 (Activation)	(None, 16, 15, 96)	0
max_pooling2d_3 (MaxPooling2	(None, 8, 7, 96)	0
flatten_1 (Flatten)	(None, 5376)	0
dense_1 (Dense)	(None, 2)	10754
activation_4 (Activation)	(None, 2)	0

Total params: 185,838
Trainable params: 168,972
Non-trainable params: 16,866

Fig. 4: Keras model summary for pCNN.

2.3 Training and Evaluation of Models

As in the original Tayeb and colleagues' experiment, we implemented 5-fold cross validation for training and testing of the models. They did not use an independent test set to validate their model, but rather reported the mean accuracies across all five folds. One fold was held out for testing whereas the remaining four folds were used for training. Training data were split 90% and 10% over 100 epochs with the implementation of early stopping. This was done iteratively over each of the five folds.

One issue with Tayeb and colleagues' approach was that they randomly split the training data into five folds after data augmentation. Since a sliding window was used to augment the data with multiple data points occurring across several time windows, there are likely the same points in two or more folds. Thus, the training and testing data are not completely independent which substantially limits what we can infer from the results.

3 Results

3.1 Data Capacity

The estimated data capacity for the augmented data is 36000 bits. This number was obtained using the scripts and methods as described by Friedland, Metere, and Krell [1]. Figure 5 shows the progression of the expected generalization. The expected capacity converges around 60% of the original training data size which shows that there is enough training data to generalize.

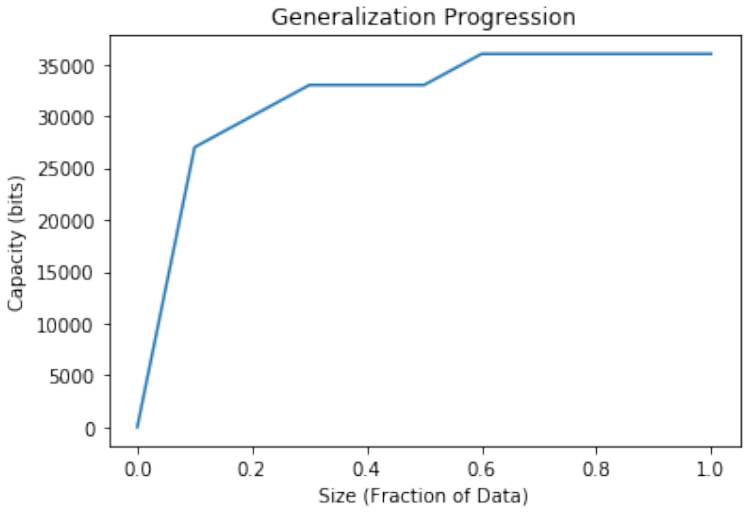

Fig. 5: Generalization capacity of training data. The capacity stabilizes around 0.6 times the size of the original training data.

3.2 Shallow CNN

Figure 6 presents the results of the shallow CNN (sCNN) with varying capacities. The original capacity of the sCNN was calculated to be 8630 bits which is lower than the calculated data capacity (36000 bits).

We increased the capacity of the network by adding neurons from a given layer until the network capacity was close to the data's capacity. We then iteratively decreased the network capacity several times by approximately half. The cross validated training accuracy reached almost 100% for the sCNNs with capacities 32220 and 16080 bits. We found that the sCNN with capacity 3388 bits had the highest cross validated test accuracy (80.1482%).

3.3 Pragmatic CNN

Figure 7 presents the results of the pragmatic CNN (pCNN) with varying capacities. The original capacity of the pCNN was calculated to be 53192 bits which is larger than the calculated capacity of the data. We then reduced the capacity of the pCNN to approximately the capacity of the data and then iteratively decreased the capacity by half. We found

A

Capacity (bits)	CV Train Acc	CV Test Acc
32220	98.39	65.013
16080	99.035	79.52
8630*	97.157	75.471
3388	94.886	80.148
1770	93.5308	71.4067

B

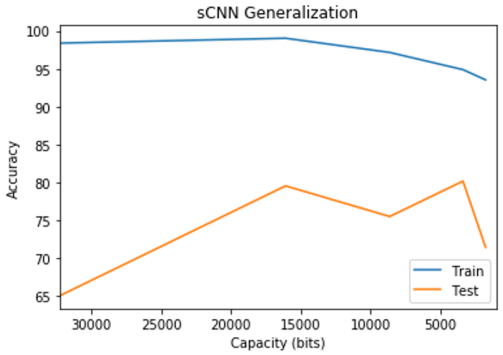

Fig. 6: A) This table shows several sCNNs with varying capacities along with the cross validation (CV) train accuracy and CV test accuracy. The asterisked capacity was the capacity of the model used in the original paper. B) CV train and test accuracy as a function of capacity of network. At 3388 bits, the test accuracy is the highest at 80.1482%

that the pCNN with capacity 36450 bits (approximately the capacity of the data) had the highest CV test accuracy of 91.926%. However, the training accuracy reached 100% indicating substantial overfitting. This puzzling result may be due to the data being split into folds after augmentation. There may be overlapping data points across folds thus resulting in a high test accuracy even when the pCNN is overfitting.

A

Capacity (bits)	CV Train Acc	CV Test Acc
53192*	100	68.252
36450	100	91.926
18418	100	90.552
9346	100	90.442
3670	96.51	79.64

B

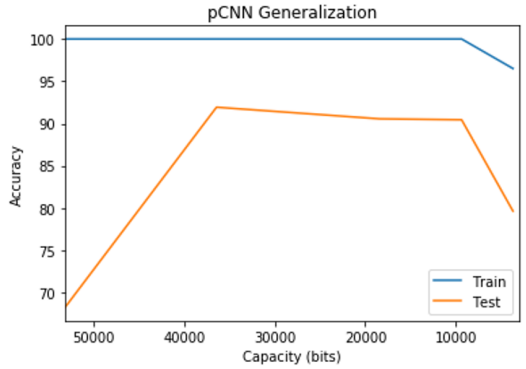

Fig. 7: A) This table shows several sCNNs with varying capacities along with the cross validation (CV) train accuracy and CV test accuracy. The asterisked capacity was the capacity of the model used in the original paper. B) CV train and test accuracy as a function of capacity of network. At 36450 bits, the test accuracy is the highest at 91.926%

4 Discussion

In this paper, we investigated whether two convolutional neural models taken from Tayeb and colleagues' paper [7] were appropriate for classifying EEG motor imagery data (imag-

ined right and left hand movements). We found that that the augmented data derived from using the sliding-window technique was sufficient to generalize in machine learning models. We, however, also discovered a fallacy in the original paper regarding how they split the augmented data in cross validation. Since using a sliding window results in overlapping points within each window, there are several repeats of the same data point within the training matrix. When the training matrix is split into five folds for cross validation, the authors did not control for the same data points appearing across more than one fold. Thus, there may be repeating points in both the training and test data in a given iteration of cross validation.

This is corroborated by our measurements of cross validated training and testing accuracy for our pCNN model. While varying the memory equivalent capacity of the model, we found that the pCNN with capacity 36450 (approximately our data capacity) had the highest cross validated test accuracy of 91.926%. However, the training accuracy for the pCNN at this capacity was 100% indicating overfitting. One way that an overfitted model can have an excellent test accuracy score is if there is overlapping data in the train and test set. This additionally addresses why some of the deep learning models within this paper had unusually higher test accuracy compared to other EEG-based BCI experiments.

Although we observed that the pCNN with the highest test accuracy had a 100% training accuracy, we found that this was not the case for the sCNN. The sCNN accuracy-capacity curve behaved much more typically than that of the pCNN. The sCNN with capacity 3388 bits had the highest cross validated test accuracy at 80.148%. We observed that the training accuracy started to decrease significantly around this capacity as well.

Although the generalization process of the sCNN looks promising, we cannot make concrete conclusions about the validity of the expected generalization of this model unless we use completely independent cross validated train and test sets. In addition, we know that the regularization applied to the models reduces the capacity of an unknown amount which is not taken into account in the calculated MEC's. For future work, we plan to split training and testing data prior to augmenting the data and train both the sCNN and pCNN on this data in addition to do more research on finding a method to measure the exact amount on reduction in capacity using regularization.

5 Contributions

5.1 Collaboration

We collaborated extensively with Eigil Bagger and Nathaniel Rose. While we worked on separate models, both our group and Bagger and Roses' group based our experiments off the same paper. Additionally, we have consulted with them in estimating data capacity, MEC of our models, and many other project related questions.

5.2 Responsibility

Susan Hao: sCNN and pCNN experiments, MEC calculations, data capacity calculations, generalization and capacity discussions, computational environement setup

Vilde Roland Arntzen: exploratory RNN and RCNN experiments, MEC calculations, generalization and capacity discussions, paper latex setup

References

1. Friedland, G., Krell, M.M.: A capacity scaling law for artificial neural networks (2018)
2. Friedland, G., Krell, M.M., Metere, A.: A practical approach to sizing neural networks. LLNL-TR-758456 (2018)
3. Hinton, G.E., Srivastava, N., Krizhevsky, A., Sutskever, I., Salakhutdinov, R.R.: Improving neural networks by preventing co-adaptation of feature detectors (2012)
4. Hong, K.S., Khan, M.J.: Hybrid brain–computer interface techniques for improved classification accuracy and increased number of commands: A review (2017)
5. Leeb, R., Brunner, C., Muller-Putz1, G.R., Schlogl2, A., Pfurtscheller, G.: Bci competition 2008 – graz data set b (2008)
6. Shanbhag, A., Kholkar, A., Sawant, S., Vicente, A., Martires, S., Patil, S.: P300 analysis using deep neural network (2017)
7. Tayeb, Z., Fedjaev, J., Ghaboosi, N., Richter, C., Everding, L., Qu, X., Wu, Y., Cheng, G., Conradt, J.: Validating deep neural networks for online decoding of motor imagery movements from eeg signals (2019)
8. Xun, G., Jia, X., Zhang, A.: Detecting epileptic seizures with electroencephalogram via a context-learning model (2016)