EEG BRAIN STATE DECODING

Eigil Bagger 3035425540, Nathaniel Rose 3032538367

University of California, Berkeley

ABSTRACT

Motor imagery is of vital importance within the field of brain-computer-interface and EEG is one of the most used modalities in the research. In this paper, we attempt to repeat, reproduce and assess the findings of Tayeb et al [1] through the scope of memory equivalent capacities. The original authors have used traditional and deep learning methods and have achieved impressive binary classification accuracies around 95%, while state-of-the-art methods arrive at maximally 85%. The main findings of the paper concludes the results cannot be repeated or reproduced for the shallow methods and that the high accuracies for the deep models originates from the flawed combination of sliding-window preprocessing and 5-fold cross validation, which results in the same data in training and test sets. In addition, several aspects of generalization of deep learning methods are discussed through the use of the concept of capacity.

Index Terms— Motor imagery, EEG, Deep learning, Generalization, Reproducibility

0. COLLABORATION

This project has been carried out in close cooperation with Vilde Arntzen and Susan Hao. While the scope of this project is to assess the shallow models *Naïve Bayes* (NB) and *Quadratic Linear Discriminant Analysis* (QLDA) and the deep learning architechture denoted dCNN in the original paper, the other authors are working on the more shallow architechtures *pCNN* and *sCNN*

Responsibility:
Eigil Bagger - dCNN experiments, generalization discussion, capacity calculations and capacity discussion
Nate Rose - Naïve Bayes, QLDA, EEG discussion, set up of computational environments,

1. INTRODUCTION

Unveiling of the relationsship between intent and brain activity remains a challenge with tremendous potential utility within Brain-Computer Interface (BCI) [2]. BCI underpins many novel applications that are important to people's daily life, especially to people with psychological/physical illnesses or disabilities [3]. This paper is concerned with such brain decoding of *motor imagery tasks*, where a subject produces the thoughts of certain muscle movement and a computer accurately classifies the subjects intent based off of *electroencephalogram* (EEG) signals. EEG is non-invasive, inexpensive, portable and has a high temporal resolution but suffers from a low signal-to-noise ratio, from being sensitive to external sources of noise and from lack of deep brain sophistication and spatial resolution compared to i.e. fMRI [2, 4, 5]. Motor imagery is the most studied EEG classification task (22% of papers [6]), and while recent advances in deep learning has permeated the field of motor imagery [6], there is no consensus on deep learning algorithms or approach. In this project, we repeat, reproduce, assess and evaluates some of the approaches by Tayeb et. al [1]. The paper, which is concerned with binary classification of hand movement, is interesting because it claims to have accuracy up of to 95.72%, while state-of-the-art approaches normally arrives at 70-85% [2, 3]. It was found that the high accuracies can be credited due to flawed train-test practice.

2. THEORY

2.1. Brain State Decoding with EEG

EEG measures brain activity by utilizing electrodes placed on the scalp to measure voltage osciliations resulting from neuronic action potentials firing in the brain. The use of deep learning models or other generative models to decode EEG signals can prove to be a difficult task due to the feedback from environmental electrical current, muscle tension, wavering electrodes and other data artifacts that may distort the signal. EEG data is normally captured using 32 or 64 electrodes and a subset of these are then used for further processing. In this paper, 3 electrodes (chanels) have been chosen. While EEG is very inexpensive and has a high time resolution (Samples 250 times per second) it suffer from the following limitaitons:

- Low signal-to-noise ratio
- Non-stationary signal, which poses a big challenge for online use
- High inter-subject variability

2.2. Reproduction in EEG papers

A recent meta study found that only 11 out of 156 (7%) EEG brain decoding studies were easily reproducible [7]. The main reason for this is the lack of data availability, but also that only 19% of studies make their code available.

2.3. Generalization in deep neural networks

Deep learning has shown excellent representation learning ability and is empowered to learn distinct high-level representations from raw brain signals without manual feature selection [3]. This is highly desirable in EEG, as it allows for online use and removes the need for very time-intensive human preprocessing. In addition, it has been found that deep learning models give a median accuracy gain of 5.4% over traditional methods (156 papers reviewed) [7]. However, a range of authors have been concerned with the lack of reproducibility and possible lack of generalization for the networks. Often, the EEG networks have far more trainable model parameters than number of samples that they are trained on.

Zhang et al. recently demonstrated through randomization tests that deep neural networks are able to fit pure noise without the need for substantially longer training time [8]. This is curious, because over-parameterized models normally achieve excellent generalization [9]. The reason for this is not well understood, however [9] showed that deep neural networks normally do not just memorize data, but is *content-aware* through its optimization. Furthermore, [8] shows how their results rule out traditional complexity measures for controlling generalization error including the VC dimension. Nevertheless, in this paper we are mostly concerned with the VC dimension, following the lead of Friedland et al [10]. They argue that that the randomization tests show that the VC dimension of the analyzed networks is above the size of the used dataset.

2.4. VC dimension in convolutional neural networks

Friedland et al treats neural networks as encoders in a Shannon communication model and through this sets up upper bound of the VC dimension [10]. Bits are used, because in a binary classification setting, $2^{D_{VC}}$ different labellings can be memorized by the neural networks. Friedland et al then sets up 4 rules for calculation of upper bound of VC dimension for feed-forward neural networks. For convolutional neural networks, these VC bounds cannot be directly used. In this paper we might use a naïve method, that might underestimate the upper bound:

First idea is that, with the exception of the first layer, all subsequent layers are restricted by the information content in layers before them. Following Rishi's comment, a convolutional layer with a subsequent maxpool layer restricts capacity of subsequent layers by $xbits \cdot out_w \cdot out_h \cdot out_{ch}$, where $xbits$ is the information in each measurements. For EEG data, while the measurements are in floating points, the resolution is usually taken to be 12 bits.

2.5. Dropout-layers

Regularization operates as a capacity reducer. One such regularization, that is not well understood yet, is drop-out layers. The original authors [1] use dropout layers in their models, but in order to being able to measure and assess capacity

metrics, we have excluded dropout in this paper, with the exception of one experiment. This experiment was conducted to see the difference in performance with the exclusion of the dropout layers. Dropout prevents co-adaption of different units and can be seen as analogous to training an ensemble of networks.

3. DATA

In the original paper [1], two datasets are used: One that the original authors have collected and the *GrazB* dataset [11] from a 2008 BCI competition. Since Tayeb et al have only made their data partially publicly accesible, this paper only conduct analyses on the GrazB dataset. The data consist of EEG data from 9 right-handed subjects, sitting in an armchair watching a flat screen monitor placed approximately 1m away at eye level. The subjects were instructed to mentally simulate a left or right hand movement according to instructions (an arrow pointing left or right) showcased on the screen, see figure 1.

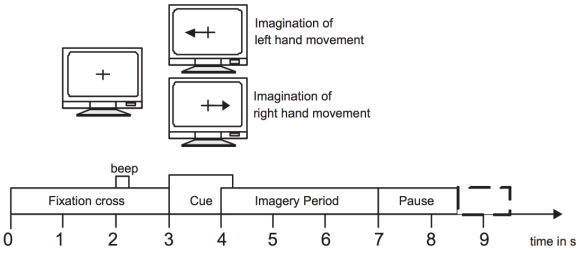

Fig. 1. Overview of experimental setup

Instruction time was 1.25 seconds and mental simulate time was 4 seconds. The equipment used to capturing the EEG signal operated at 250Hz and three electrodes were captured C3, Cz and C4. In total, this resulted in up to 360 trials per subject. The data was cleaned from electrooculography (EOG) potential, bandpass filtered between 0.5Hz and 100Hz, and a notch filter of 50Hz was enabled.

3.1. Preprocessing

In addition to the above mentioned preprocessing, Tayeb et al [1] conducts two types of preprocessing steps: The signal is filtered between 2 and 60 Hz using a 5th order zero-phase Butterworth filter and the data is clipped to $\mu(x_i) \pm 6\sigma(x_i)$ to eliminate outliers. Then the data was normalized per channel (zero mean and unit variance), and a thresholding method was utilized to remoeve artifacts.

Secondly, Tayeb et al used a moving window of 4 second with a stride of 125 ms to crop the 8 second signal into 25 4 second windows, see figure 2. This cropping technique yields

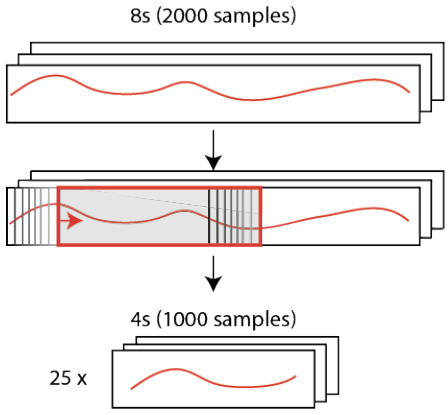

Fig. 2. Visualization of the moving window approach. Created in Illustrator

25 times as many observations and is often used both in EEG brain decoding [7] and in image recognition [12]. Some of our critique points of this paper is concerned with letting different windows from the same trial (out of the 25 available) persist in both training and testing set.

4. METHODS

4.1. Original method - Experimental setup

The architecture of the dCNN model of the original paper can be seen in figure 3. The architechture was originally proposed by Schirrmeister et al. [12]. The original authors estimate the performance of their neural networks by conducted 5-fold cross validation on their training set and reporting heir mean accuracy of these 5 folds. For the GrazB dataset, the accuracy of the dCNN architechture is reported to be 95.72%

4.2. Reproducibility of Gumpy models

In an effort to address the absense of reproducibility in the field of machine learning, Tayeb et al produced an open

Layer	Input	Operation	Output	Parameters
1	$E \times T$	$25 \times$ Conv2D (1×10)	$E \times 1015 \times 25$	275
2	$E \times 1015 \times 25$	Reshape	$E \times 1015 \times 25 \times 1$	-
	$E \times 1015 \times 25 \times 1$	$25 \times$ Conv3d $(3 \times 1 \times 25)$	$1 \times 1015 \times 1 \times 25$	1900
	$1 \times 1015 \times 1 \times 25$	BatchNorm	$1 \times 1015 \times 1 \times 25$	100
	$1 \times 1015 \times 1 \times 25$	ELU	$1 \times 1015 \times 1 \times 25$	-
	$1 \times 1015 \times 1 \times 25$	MaxPool2D (3×1)	$338 \times 25 \times 1$	-
	$338 \times 25 \times 1$	Dropout (0.5)	$338 \times 25 \times 1$	-
3	$338 \times 25 \times 1$	$50 \times$ Conv2D (10×25)	$329 \times 1 \times 50$	12,550
	$329 \times 1 \times 50$	BatchNorm	$329 \times 1 \times 50$	200
	$329 \times 1 \times 50$	ELU	$329 \times 1 \times 50$	-
	$329 \times 1 \times 50$	MaxPool2D (3×1)	$109 \times 1 \times 50$	-
	$109 \times 1 \times 50$	Dropout (0.5)	$109 \times 1 \times 50$	-
4	$109 \times 1 \times 50$	Reshape	$109 \times 50 \times 1$	-
	$109 \times 50 \times 1$	$100 \times$ Conv2D (10×50)	$100 \times 1 \times 100$	50,100
	$100 \times 1 \times 100$	BatchNorm	$100 \times 1 \times 100$	400
	$100 \times 1 \times 100$	ELU	$100 \times 1 \times 100$	-
	$100 \times 1 \times 100$	MaxPool2D	$33 \times 1 \times 100$	-
	$33 \times 1 \times 100$	Dropout (0.5)	$33 \times 1 \times 100$	-
5	$33 \times 1 \times 100$	Reshape	$33 \times 100 \times 1$	-
	$33 \times 100 \times 1$	$200 \times$ Conv2D (10×100)	$24 \times 1 \times 200$	200,200
	$24 \times 1 \times 200$	BatchNorm	$24 \times 1 \times 200$	800
	$24 \times 1 \times 200$	ELU	$24 \times 1 \times 200$	-
	$24 \times 1 \times 200$	MaxPool2D (3×1)	$8 \times 1 \times 200$	-
6	$8 \times 1 \times 200$	Flatten	1600	-
	1600	Softmax	K	3202
Total				268,977

Fig. 3. Architechture of the dCNN

source python software package named gumpy. The gumpy toolkit contains a subset of models, preprocessing methods, and plotting functions that target BCI signals used in their study. Our team seeks to reproduce the quadratic linear discriminant analysis (QLDA) model and the naive bayes model to compare if measurements obtained with stated precision by the Tayeb et all using the same measurement procedure is congruent in results.

4.3. Assessment of input space capacity

We estimate the capacity of 10%, 20%, .., 100% of the input data, both the original and the cropped (slided time window). This is done both to see if the numbers converge, but also to see where the expected capacity of a neural network that is able to memorize 100% of the training data lies. To do so, the script `capacityreq.py` was used [13].

4.4. Assessment of network capacity

The capacity of the original dCNN, figure 3, is estimated through the naïve method:

Conv1: 250 filter weights + 25 bias = 275 bits.
Output feature: $3 \cdot 1015 \cdot 25$ = 7125 bits

Conv2: 1875 filter weights + 25 bias = 1900 bits. min(1900, 7125) = 1900 bits
Output: $1015 \cdot 1 \cdot 25$ = 25375 bits
Maxpooling: $338 \cdot 25 \cdot 1$ = 8450bits. min(1900, 8450) = 1900

Conv3: 12500 filter weights + 50 bias = 12550 bits. min(12550, 1900) = 1900
Output: $329 \cdot 1 \cdot 50$ = 16450 bits
Maxpooling: $109 \cdot 1 \cdot 50$ = 5450 bits. min(1900, 5450) = 1900

Conv4: 50000 filter weights + 100 bias = 50100 bits. min(1900, 50100) = 1900
Output: $100 \cdot 1 \cdot 100$ = 10000 bits
Maxpooling: $33 \cdot 100$ = 3300 bits. min(1900, 3300) = 1900

Conv5: 200000 filter weights + 200 bias = 200200 bits. min(200200, 1900) = 1900
Output: $24 \cdot 1 \cdot 200$ = 4800 bits
Maxpool $8 \cdot 200$ = 1600. min(1900, 1600) = 1600

FC: 1600 bits
The overall bound on the capacity is then calculated as

$$Cap_{orig} = 275 + 1900 + 1900 + 1900 + 1900 + 1600 = 9485 \quad (1)$$

It can be seen, that by calculating the capacity as above (which may be wrong, we're unsure of that), the number of filters in the second convolutional layer restricts the capacity of all remaining layers. For this reason, we reduce capacity by simply reducing the number of filters in the first and second layers. In total, we run 6 such experiments on the cropped to see the effect of capacity on training and testing accuracies. The deep learning architechtures (where we have excluded all dropout-layrs) we run are as follows (denoted by the number of filters in each convolution)

- Original network: (25, 25, 50, 100, 200). Capacity: 9485
- (20, 20, 50, 100, 200). Capacity: $5 \cdot 1540 + 220 = 7920$ bits
- (15, 15, 50, 100, 200). Capacity: $5 \cdot 1140 + 165 = 5865$ bits

- (10, 10, 50, 100, 200). Capacity: $5 \cdot 760 + 110 = 3910$ bits
- (5, 5, 50, 100, 200). Capacity: $5 \cdot 380 + 55 = 1955$ bits
- (1, 1, 50, 100, 200). Capacity: $5 \cdot 76 + 11 = 391$ bits

4.5. Assessment of split procedure

Looking through the code of the original authors [14], we suspected that their 5-fold cross validation was set up in a flawed way. The authors conduct their sliding window, yielding a total of 10044 observations and then split these observations randomly in training and test sets. However, it was hypothesized that this splitting results in the same data points being in the training and testing set with the difference of a translation of 125 ms, which can be seen as a small additive noise. To see if this gives artifically high accuracies, we implemented the capacity estimation as in section 4.4, but where we split and then roll the window so that crops from the same trial always end up in the same set. We did this for the same networks architectures as covered in 4.4.

5. RESULTS

5.1. Reproducibility of models with Graz dataset

In the original paper [1], balanced accuracy was chosen as the evaluation metric for the trained models and a stratied 5-fold cross-validation was applied during the validation phase.

The VC Dimension is defined as the maximum number of points that can be arranged so that the classifier can shatter them. To start, the team used the Ipython notebooks provided through the gumpy toolkit to run preprocessing and model training on a gpu enabled cluster pool. We reviewed the naive bayes and QLDA model for validating the reproducibility of their experiment given the simplicity of the model's linear VC Dimension,$2^{D_{VC}}$[10]

The naive bayes model performed at a mean average accuracy of 63.49% while the QLDA model that was tested, achieved a 59.17% accuracy score after following an identical preprocessing configuration to Tayeb et al. This outcome compared to the expected model accuracy of 79% and 74% for QLDA and the naive bayes model respectively, demonstrates an inability reproduce the scores of Tayeb et all within the targeted standard error using the gumpy toolkit notebooks they provide.

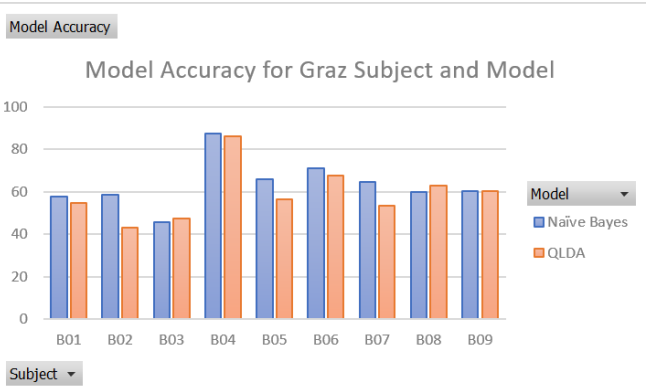

Fig. 4. Meeasured testing score fter 5 fold cross validation per subject

5.2. Capacity estimation of input space

The estimated capacity at 10%, 20%, ..., 100% of the original data can be seen in table 1. It can be seen that the estimated capacity doesn't converge at higher percentages of the data and thus, it can be concluded, that there isn't enough data.

Split ratio [%]	Max capacity [bits]	Estimated capacity [bits]
10	78026	24000
20	174058	30000
30	282094	36000
40	330110	36000
50	420140	42000
60	522174	42000
70	606202	42000
80	702234	42000
90	780260	48000
100	834278	48000

Table 1. Max capacity and estimated capacity on raw data (324 samples). Estimated capacity doesn't converge

Likewise, after having conducted the rolling window on the data set (transforming the data set from $\mathbb{R}^{322\times2000\times3}$ to $\mathbb{R}^{10044\times1000\times3}$, the capacity of the input space is calculated at 10%, 20%, ... up to 100%. The result can be seen in Table

1. A side-by-side comparison can be seen in figure 5

Split ratio [%]	Max capacity [bits]	Estimated capacity [bits]
10	1165164	27009
20	2420418	30010
30	3636633	33011
40	4831827	33011
50	6141135	33011
60	7237230	36012
70	8258250	36012
80	9489480	36012
90	10642632	36012
100	11789778	36012

Table 2. Max capacity and estimated capacity on augmented data (10044 samples). Estimated capacity converges at 60% of the data

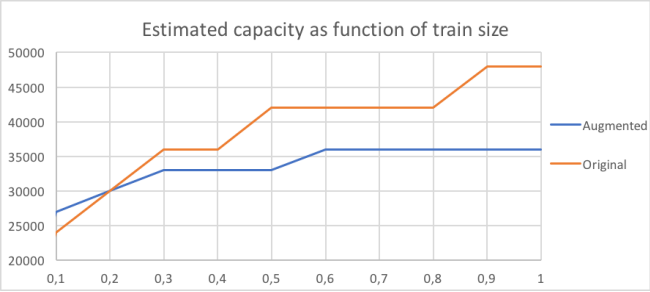

Fig. 5. Estimated capacity for original data and rolling window

One very interesting conclusion here is, that despite the dataset growing intensely in size, the estimated capacity actually decreases quite drastically. The maximum capacity of course increases drastically (a factor of 14), but the interesting metric here is the estimated capacity. In addition, the estimated capacity for the augmented data converges while that for the original data does not, which points towards the augmented data set being big enough. This is very interesting as the augmented dataset simply are crops of the original data set.

Furthermore, it can be seen that the maximum capacity scales approximately with the size of the data set.

It should be noticed that these estimated capacities are way above that of the original dCNN and its reduced forms.

5.3. Performance as function of capacities

The performance of the models can be seen in figure 6

Several very interesting points can be made here.

Firstly, it can be seen that with no trial mixing (that is, conducting the moving window *after* splitting), the testing accuracy is reduced significantly in all reductions of the original model, even though the splitting seeds remain the same. As an example, consider the original model. The testing accuracies here are close to 90%, while the actual testing accuracy should be around 75%. This means, that the splitting of the original authors is flawed. This is a serious problem, as all models in the paper uses the same train-test-splitting code and thus, **none** of the results in the paper can be seen as reliable. This underlines the accute need of keeping training and testing sets seperate. We are in the process of contacting the authors to let them know about the mistake.

Secondly, it can be seen that the capacity needed to memorize all data is quite low: At 8000 bits, all training set is memorized which is in big contrast to the estimated memorization capacity of 36000 bits. Friedland et al [10] showed that several different model reached perfect memorization very close to the estimated capacity, so the big discrepancy here is interesting. We suspect that the estimated capacity from the script might not be adequate to floating point input spaces with thousands of redundant variables, but future work should focus on that causality.

Thirdly, it can be seen that the anticipated behavior of the training and testing accuracy on capacity is fulfilled. The training accuracy is the biggest, when the model overfit, the testing accuracy is the biggest when the sum of bias and variance errors are minimized (*the sweet spot*), and both accuracies fall drastically, when the models is too simple to capture a good representation. If we continued reducing the capacity towards 0, the training accuracy would converge towards the baseline prediction of 50%.

5.4. The effect of drop-out

One single dropout experiment was run on the original dCNN model, but with the redefined way of doing sliding window (first splitting, the sliding window). It was stopped early due the computational speed, but arrived at a mean training ac-

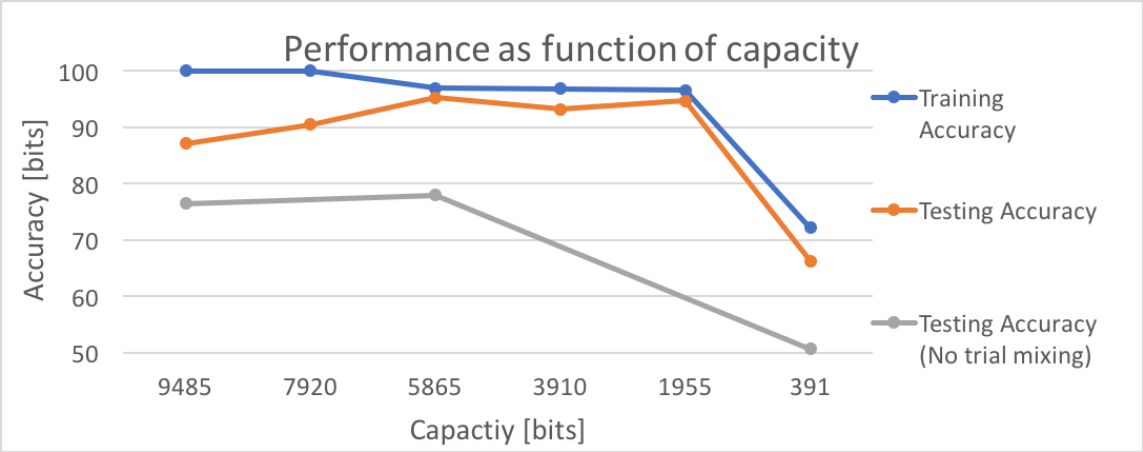

Fig. 6. Caption

curacy of 91.69% (substantially lower than the perfect memorization in the other runs) and a mean testing accuracy of 80.55%, so significantly higher than the unregularized counterpart. These results pinpoint the role of dropout layers as regularizers, that forces the network to generalize rather than memorize. The role of dropout was not examined further

6. FUTURE WORK

Future work would seek to look further into the calculation of capacity for convolutional neural networks. We don't feel like we have a good enough understanding of the MacKay information model and the derivations in the paper by Friedland et al [10] to derive better bounds for convolutional neural networks yet, but future work should look into this. That might also be able to explain the big discprancy between estimated memory equivalent capacity of the input space and the actual capacity needed to memorize all data, if the 'actual capacity' was calculated to be too low. In addition, the role of regularization should be examined much further, as we in this paper have compared regularized results with unreguralized, without being able to quantify how the theory changes with dropout layers.

7. CONCLUSION

?? Thus, several aspects of the paper by Tayeb et al. was discussed [1]. It was found, that the shallow model accuracies could not be reproduced, but was estimated to be substantially lower. In addition, it was found that the accuracies of the deep models were overly optimistic in their accuracy assessment due to an errorneous implementation of train-test-split, which undermines the results of the paper. It was also found, that while there isn't enough data in the original training set to make the capacity requirements converge, the cropping method yielded enough data for the capacity to stabilize. Lastly, it was found that there was a big discpreancy between estimated capacity required to memorize all the data and the actual capacity required. The actual capacity was approximately 1/4 of the estimated. Future work should look into the reasons for this, but we suspect that it might be due to wrong calculations of the capacity of convolutional neural networks.

8. PROJECT QUESTIONS

(1) *What is the variable to be predicted?*

Using motor imagery, determining whether the subjects were instructed to mentally simulate moving his/her hand to the left or the right.

(2) *How much data do we have to train the prediction of the variable?Are the classes balanced? How many modalities could be exploited in the data? Is there temporal information? How much noise are we expecting? Do you expect bias?*

We have approximately 400 data points with 6000 features in total and 4/5 of this will be used for training. Using the moving window method, these data points are transformed into 10000 data points with 3000 features. The classes are balanced. There is temporal information as each feature corresponds to the measurement at a time step in a time series. We are expecting a very high amount of noise as EEG is famous for high Noise-to-signal ratio.

3) *How well is the data annotated (anecdotally)? What is the annotator agreement (measured)?*
The data is annotated 100% as the annotation occurs automatically in the experimental setting (no need for human judgement). However, subjects not doing as instructed might be a problem

4) *Given questions 1-3: Are we reducing information (pattern matching) or do we need to infer information (statistical machine learner)? As a consequence, what seems the best choice for the type of machine learner per modality?*
We are heavily reducing information from 6000 features down to a binary classification.
5) *Estimate the memory equivalent capacity needed for the machine learner of your choice. What is the expected generalization? How does the progression look like.*
Done
6) *Train your machine learner for accuracy at memory equivalent capacity. Can you reach near 100% (diagnose)?* We trained at capacities much lower than the estimated MEC and could easily reach 100%
7)-10) Mostly completed, but havne't tried any other machine learners.

9. REFERENCES

[1] Zied Tayeb, Juri Fedjaev, Nejla Ghaboosi, Christoph Richter, Lukas Everding, Xingwei Qu, Yingyu Wu, Gordon Cheng, and Jorg Conradt, "Validating deep neural networks for online decoding of motor imagery movements from eeg signals," *Sensors*, vol. 19, no. 1, pp. 210, 2019.

[2] Xiang Zhang, Lina Yao, Quan Z Sheng, Salil S Kanhere, Tao Gu, and Dalin Zhang, "Converting your thoughts to texts: Enabling brain typing via deep feature learning of eeg signals," in *2018 IEEE International Conference on Pervasive Computing and Communications (PerCom)*. IEEE, 2018, pp. 1–10.

[3] Xiang Zhang, Lina Yao, Xianzhi Wang, Jessica Monaghan, and David Mcalpine, "A survey on deep learning based brain computer interface: Recent advances and new frontiers," *arXiv preprint arXiv:1905.04149*, 2019.

[4] Greta Tuckute, Sofie Therese Hansen, Nicolai Pedersen, Dea Steenstrup, and Lars Kai Hansen, "Single trial decoding of scalp eeg under natural conditions," *BioRxiv*, p. 481630, 2019.

[5] Luis Fernando Nicolas-Alonso and Jaime Gomez-Gil, "Brain computer interfaces, a review," *sensors*, vol. 12, no. 2, pp. 1211–1279, 2012.

[6] Alexander Craik, Yongtian He, and Jose L Contreras-Vidal, "Deep learning for electroencephalogram (eeg) classification tasks: a review," *Journal of neural engineering*, vol. 16, no. 3, pp. 031001, 2019.

[7] Yannick Roy, Hubert Banville, Isabela Albuquerque, Alexandre Gramfort, Tiago H Falk, and Jocelyn Faubert, "Deep learning-based electroencephalography analysis: a systematic review," *Journal of neural engineering*, 2019.

[8] Chiyuan Zhang, Samy Bengio, Moritz Hardt, Benjamin Recht, and Oriol Vinyals, "Understanding deep learning requires rethinking generalization," *arXiv preprint arXiv:1611.03530*, 2016.

[9] Devansh Arpit, Stanisław Jastrzebski, Nicolas Ballas, David Krueger, Emmanuel Bengio, Maxinder S Kanwal, Tegan Maharaj, Asja Fischer, Aaron Courville, Yoshua Bengio, et al., "A closer look at memorization in deep networks," in *Proceedings of the 34th International Conference on Machine Learning-Volume 70*. JMLR. org, 2017, pp. 233–242.

[10] Gerald Friedland, Alfredo Metere, and Mario Krell, "A practical approach to sizing neural networks," *arXiv preprint arXiv:1810.02328*, 2018.

[11] R Leeb, C Brunner, G Müller-Putz, A Schlögl, and G Pfurtscheller, "Bci competition 2008–graz data set b," .

[12] Robin Tibor Schirrmeister, Jost Tobias Springenberg, Lukas Dominique Josef Fiederer, Martin Glasstetter, Katharina Eggensperger, Michael Tangermann, Frank Hutter, Wolfram Burgard, and Tonio Ball, "Deep learning with convolutional neural networks for eeg decoding and visualization," *Human brain mapping*, vol. 38, no. 11, pp. 5391–5420, 2017.

[13] "nntailoring," https://github.com/fractor/nntailoring/tree/master/capacityreq, 2018, [GitHub Repository. Online; accessed 23-December-2019].

[14] "Gumpy-deeplearning," https://github.com/gumpy-bci/gumpy-deeplearning, 2018, [GitHub Repository. Online; accessed 23-December-2019].