ENS 492 – Graduation Project (Implementation) Progress Report II

Project Title: Unsupervised SSVEP Signal Classification

Group Members:
Deniz Küçükahmetler
Pelinsu Çiftçioğlu
Giray Coşkun

Supervisor:
Hüseyin Özkan

Date: 14.11.2021

Faculty of Engineering and Natural Sciences
Sabancı University

1 Project Summary

Spellers (i.e. a visual representation of character matrix, can be seen in Figure 1) via Brain-Computer Interfaces (BCI), aims to establish direct communication between the brain and a computer for individuals who need complete assistance to communicate[4]. By using the visual presentation of alphanumeric characters through the speller, the computer interface allows exploiting brain signals through electroencephalogram (EEG). The collected EEG data utilized enables the user to type solely based on his/her brain activity and therefore does not require communication through speech. Presented as visual stimuli, each character on the speller flickers with a unique frequency that generates Steady-State Visually Evoked Potentials (SSVEP) which are captured by EEG. Target identification from these collected data is a multi-channel, multi-class classification problem[4]. State-of-arts solutions offer various approaches one of which is deep learning models used with various statistical techniques in order to employ training with supervised learning methods. An unsupervised learning approach would allow training without using labeled data which is the inspiration for our main goal of the project. Our aim is to build a deep learning architecture developed by employing both supervised and unsupervised learning methods to improve information transfer rate (ITR) with maintaining high accuracy in terms of character identification. ITR (see below for the formula) is a function of model accuracy and visual stimulation duration[4] i.e. amount of transferred information per unit time which is a metric we utilize to measure performance. Consequently, we aim to establish an unsupervised deep learning architecture which to outperform current training-free methods.

$$\text{I(P,T)} = \log_2 M + P \times \log_2 P + (1-P) \times \log_2[\tfrac{1-P}{M-1}) \times \tfrac{60}{T}$$

where *P=accuracy of model, T=stimulation duration, M=number of classes*

Our project utilizes 2 different datasets, namely Benchmark[2] and BETA [5] each containing brain activity data gathered from EEG while performing character identification tasks, a total of 40 characters including alphanumeric and a few crucial characters (i.e. punctuation). Benchmark dataset consists of 35 subject data performed in 6 block trials with each block including a collection of 3 seconds of data per character[2], whereas BETA consists of 40 subjects including 4 blocks of trials[5].

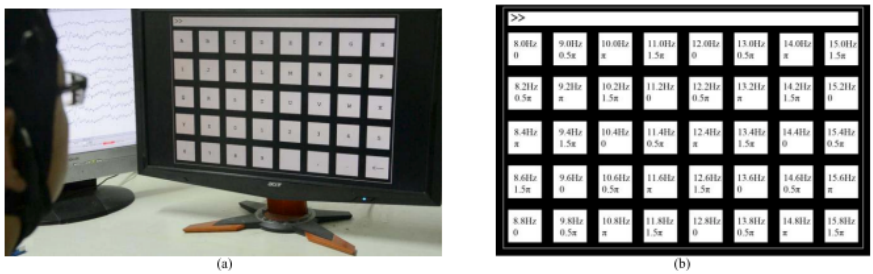

Figure 1: Visual Representation of the characters matrix and their corresponding unique frequencies. [2]

2 Scientific/Technical Developments

First stage of the project has been completed by implementing a Fast Fourier Transform (FFT) and a Canonical Correlation Analysis (CCA) which is a training-free multivariable statistical method [1] to realize the motivation of target identification through SSVEP multi-channel signals and to have a deeper understanding towards application. This step also has established a base to compare with developed models. As the CCA method to predict the target character is a training-free method, the developed approach is expected to outperform this method as a base comparison.

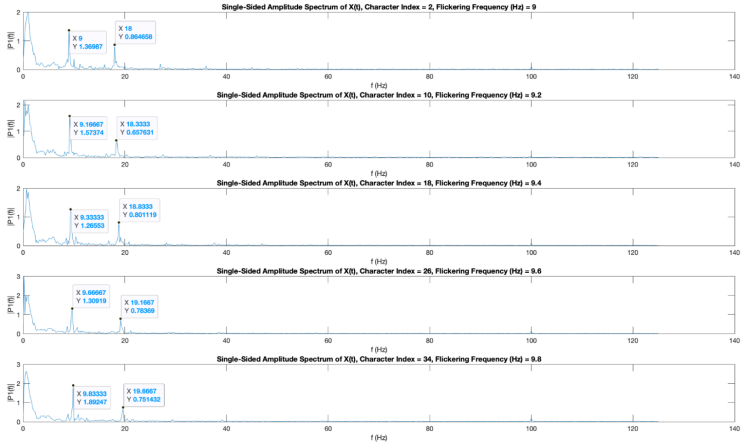

Figure 2: FFT results for example instances

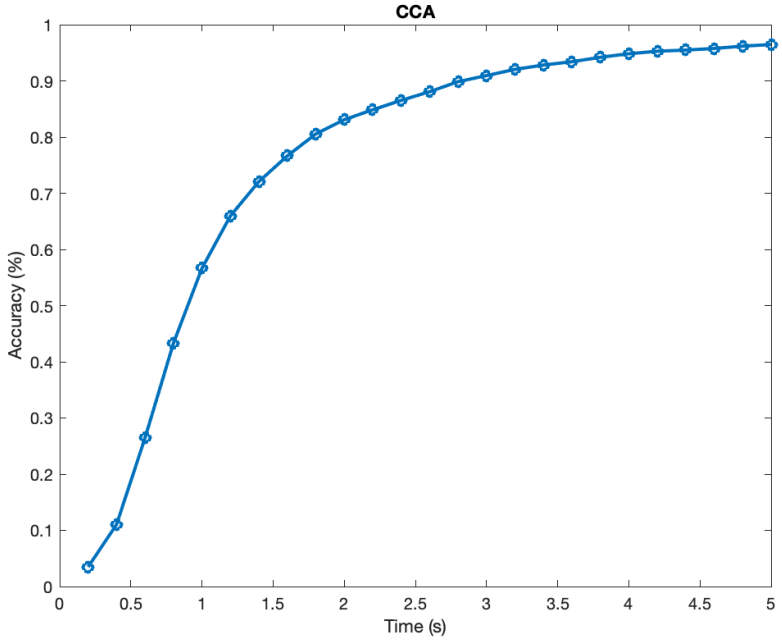

Figure 3: CCA Method Accuracy

Second stage of the project has been completed by evaluating a different neural network based on CCA. Network designed such that M branch is parallel developed to differentiate 40 characters. After fine-tune of the model however the performance was under the original neural network[4] and due to cost of training the model project is directed to new methods of ensemble and an unsupervised learning strategy.

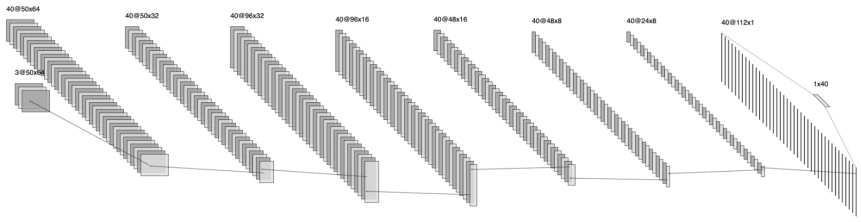

Figure 4: M-branch Neural Network Structure

Two main approaches are continued first the unsupervised learning with a custom

loss and another with ensemble strategy via auto-encoders. Principal Component Analysis is conducted first to understand more about data distribution and separation of classes. However the dimension of data is too high to utilize PCA therefore reconstruction error is also very high.

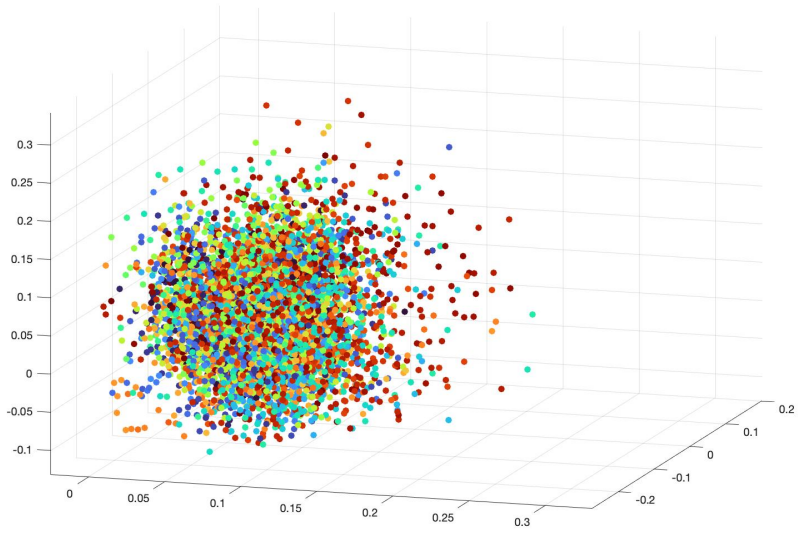

Figure 5: PCA Analysis with 3 coefficients

2.1 Utilizing Ensemble Method

Ensemble idea has been utilized such that the best neural network at hand is trained specifically to each subject in the training data. This process consisted of implementing an ensemble approach which is inspired by the idea that some data distributions could be similar to certain data distributions while being very different from others. A metric is chosen to find the best models to vote for classification label which are called voters. The ensemble idea have been utilized and enhanced the accuracy by utilizing the SAMME algorithm [6]. The project has utilized auto-encoders and minimum-squared metric to choose the voter models. An auto-encoder is trained with one-leave-out strategy. There are total of 35 subjects in Benchmark data set [2]. 34 subjects are used at training the global auto-encoder and each global auto-encoder fine-tuned for each subject separately in total of 34 auto-encoders for each subject.

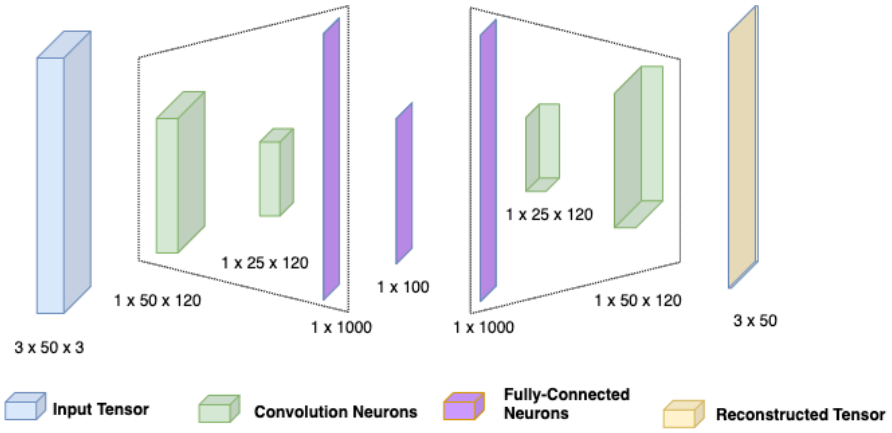

Figure 6: Auto-encoder Structure

After the training process of auto-encoders a classifier is trained from auto-encoder bottleneck. Both single layer and multiple layer classifiers have been tested. The voter classifiers are chosen based on mean squared error $\frac{1}{n} \times \sum_{i=1}^{n} (Y_i - \hat{Y}_i)^2$ Results show a weak correlation between mean-squared error of auto-encoders and accuracy of the projects.

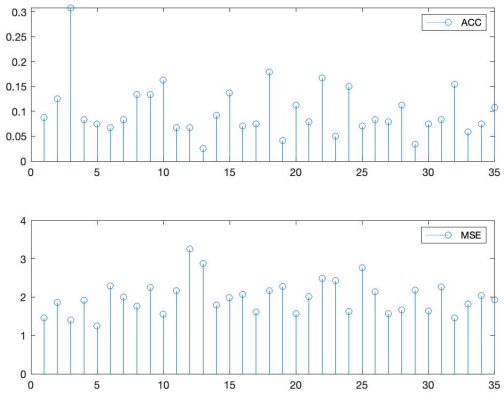

Figure 7: Accuracy vs Minimum Squared Error on average

2.2 Utilization of similarity-based approaches for training (neighborhood selection)

Main focus of the project was on developing an unsupervised learning strategy therefore a custom loss function is suggested by our supervisor with 2 important hyper-parameters. λ parameter is set to give weights to global model and neighbors. Number of neighbors is another hyper-parameter which can be chosen as a constant or dynamically based on instance. The loss function is based on on the pseudo-labels of the test subject and the neighbor instances chosen by a common channel combination.

$$Cross - Entropy \rightarrow H(p,q) = -\sum_{x \in X} p(x) \times \log q(x)$$

$$Loss = \lambda \times (H(g(x), g_0(x))) + (1-\lambda) \times (\sum_{i=1}^{k} H(g(x), g_0(x_k)) \times \alpha_k)$$

$k =$ number of neighbors, $\lambda =$ hyper parameter to set weights on global model or neighbors

The global model has been initialized with weights and re-trained by using constant neighbor count and λ hyper-parameters. A grid search is done for optimum λ value from 0 to 1. The results have shown maximum accuracy increase under $\lambda = 1$ which indicates the chosen neighbors under constant hyper-parameter are not effecting positively. Therefore a dynamic lambda and neighbor choosing algorithms have been developped.

2.2.1 Choosing Parameters via CCA

Each label given by the global model is used to create a CCA template. CCA is applied to the signal instance and the template for each instance in the test subject. Each CCA operation gives a channel combination which maximizes the correlation. This procedure is handled separately for each sub-band. A single channel combination is chosen by cosine similarity[3]. From a single channel combination corrections are calculated by which neighbors are chosen. Different methods are planned to use correlation values to pick a dynamic neighbor count for each signal instance.

2.2.2 Choosing Parameters via Channel Combination Layer

Global network provides 120 different channel combinations trained by the data of 34 subjects. Therefore a single best channel combination can be chosen to use in the procedures described above. For each channel combination in-class correlation is calculated for every class predicted by the model.

3 Encountered Problems

Even though it was anticipated that the network would require much time to train, it also became beyond our laptop's computational capacity, and therefore nearly all of the computations moved to the high-performance computing (HPC) device. However, our

training models required GPU processing in order to complete the tasks in a reasonable time and the GPU node in the HPC was not always ready to use. That is because when the GPU node is allocated to some other user's computations, we had to wait until it is freed. During summer, we continued our improvements on the project and it was relatively easier to get the calculations done on HPC, yet when the semester started it became very difficult to find it empty. Another problem we encountered was a mistake in the code. As the computations get more complex, the code follows and we make sure to double-check with each other before making any further changes in the implementation. Besides what we write as a code, there are some parts we do not and use some libraries in order to perform computations. And we realized that there was an issue with our usage of one of the functions from those methods we use and it made us question a lot of past computations and results. It was very disappointing to realize such a mistake especially because it would take too much time to run all the codes again. Luckily, it didn't affect the core results and previous calculations before summer, hence we learned how to deal with such problems, fixed the issue, and moved on with further calculations. We also gained the experience of checking all the functions we use from outside sources closely. Another problem was about the solution of the problem we had to overcome which motivated our works during the summer. Up until some point we were seeing improvement as much as I developed our model and made it more custom to the input data. However, at some level, it became very challenging to generate new ideas in order to reach better results, and could not go further. Our supervisor had very valuable feedback and developing inputs throughout the process and after following closely what he suggested and conducting our experiments, and weeks of not reaching better results, we finally broke that threshold. As we also realized and solved some issues about our code, we plan to finalize our findings soon. Even it was a fallback, we understand that there could be always unpredictable aspects of projects and therefore hope to continue the project sound with our current experience level.

4 Tasks To Be Completed Before Final Report

As discussed in the previous sections, it is an important aspect of the project that modeling the neighbor selection algorithm. Therefore, our goal is to conduct more experiments with our current model and based on the performance we achieve and to merge this algorithm with our best working training approach. While conducting experiments, we also plan to observe some relationships between various metrics that we think could give valuable insights such as examining the correlation between the prediction data by the network and the template signal created from the actual data frequency. Therefore, another goal is to finalize the architecture of the main training algorithm. We are planning to utilize another loss function that is inspired by the past loss configurations and neighborhood selection by introducing new elements. After doing so, the new network might again need some more fine-tuning, both during the experiments and after collecting the best suiting configuration, so we plan to adjust the model accordingly. We tried a lot of configurations and experimented with different sample lengths for the model alongside using second stage training techniques, ensembling techniques and different preprocessing strategies. Now our aim is to collect all our ideas from previous findings

to finalize the last version of our model. After that, we will conduct tests on the BETA data set which was kept from the network from the beginning. The reason to do that was to avoid overfitting and make sure our model performs as expected in other data sets as well. We will be doing appropriate tunes to the model in order to make it compatible with the sizes of the new data but will not adjust anything in the network. And we plan to report our results on BETA and compare them with the results from Benchmark up until the project ends.

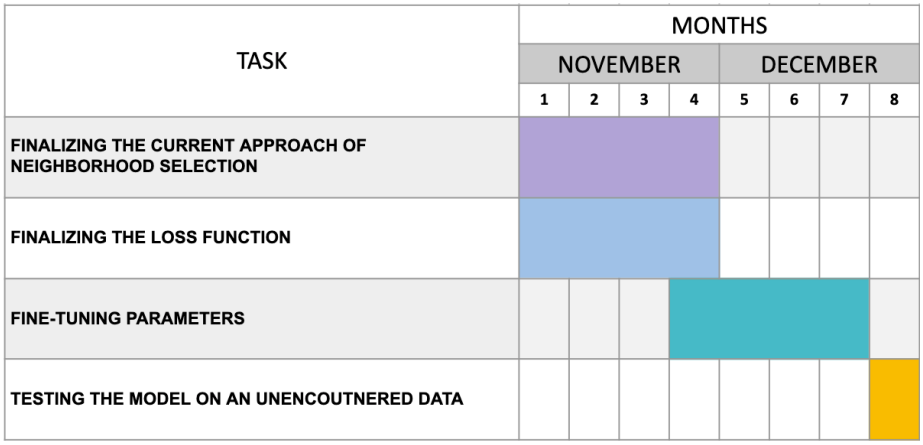

Figure 8: Gantt Chart

References

[1] Z. Lin, C. Zhang, W. Wu, and X. Gao, "Frequency recognition based on canonical correlation analysis for ssvep-based bcis," *IEEE Transactions on Biomedical Engineering*, vol. 54, no. 6, pp. 1172–1176, 2007. DOI: 10.1109/TBME.2006.889197.

[2] Y. Wang, X. Chen, X. Gao, and S. Gao, "A benchmark dataset for ssvep-based brain–computer interfaces," *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, vol. 25, no. 10, pp. 1746–1752, 2016.

[3] K. F. Lao, C. M. Wong, Z. Wang, and F. Wan, "Learning prototype spatial filters for subject-independent ssvep-based brain-computer interface," in *2018 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, 2018, pp. 485–490. DOI: 10.1109/SMC.2018.00092.

[4] O. B. Guney, M. Oblokulov, and H. Ozkan, *A deep neural network for ssvep-based brain-computer interfaces*, 2020. arXiv: 2011.08562 [cs.LG].

[5] B. Liu, X. Huang, Y. Wang, X. Chen, and X. Gao, "Beta: A large benchmark database toward ssvep-bci application," *Frontiers in Neuroscience*, vol. 14, p. 627, 2020, ISSN: 1662-453X. DOI: 10.3389/fnins.2020.00627. [Online]. Available: https://www.frontiersin.org/article/10.3389/fnins.2020.00627.

[6] O. B. Güney, E. Koç, C. Aksoy, Y. Çatak, Ş. S. Arslan, and H. Özkan, "Adaptive boosting of dnn ensembles for brain-computer interface spellers," in *2021 29th Signal Processing and Communications Applications Conference (SIU)*, 2021, pp. 1–4. DOI: 10.1109/SIU53274.2021.9477841.