# BE-521: Final Project Report

HADES
Ankit Billa, Daniel Kang, Harsh Parekh

## Abstract

*Finger flexion is controlled by firing rate of neurons, and the rate of neuronal firing is proportionate to the momentum of the finger. This means that the intracranial EEG signal along with the current position of the finger determine the future motion of the finger, this relation is quite complex in nature. Hence, for the purpose of predicting finger flexion movements from the intracranial recordings, we employ the use of a Multi-layer Perceptron Regressor, in a way that takes previous motion of the finger into account. We evaluate the performance of this algorithm compared with others standard machine learning models, by computing the Mean Squared Error (MSE), along with the correlation score between the true finger flexions and the model outputs. In terms of the final correlation score, our approach achieves an* `r` *value of* ***0.4659****. The code for the project can be found here.*

## 1. Introduction

This is the final project report for the Spring 2022 cohort of the Brain Computer Interfaces course at the *University of Pennsylvania*, submitted by the team **HADES**. The project involved predicting finger flexion movements from intracranial EEG recordings in three subjects. The data and tasks are derivatives of the *4th International Brain Computer Interfaces Competition* [9]. In the following sections we present our analysis of the given dataset (Section 2.1), our final algorithm and design choices (Section 2.2), an explanation of our design choices and alternative methods considered (Section 3), a short discussion on the physiological structure of the hand and how it effects finger flexion (Section 5), and our conclusions regarding this project. The final models used for evaluation have all been implemented using the `scikit-learn` library [8].

## 2. Method

Our final algorithm utilizes a Multi-Layer Perceptron Regressor, trained on the ECoG input data and the data glove ground truth traces. The flow chart for our algorithm is illustrated in Figure 1.

### 2.1. Dataset Analysis

We analyze the dataset to better understand the underlying structure in the data to effectively exploit it during pre and post processing, model and feature engineering. For this purpose, we study the spectrogram of each channel alongside the finger flexion. This gives us two critical insights: most neural activity occurs in lower frequency bands, and some triples of channel, finger, and frequency band have a high positive or negative co-relation.

### 2.2. Proposed Pipeline

The details of our proposed pipeline are discussed below:

#### 2.2.1 Pre-Processing

We normalize the raw data so that each channel has unit variance and zero mean, so as to eliminate biases introduced by attenuation of the signal reaching a recording site. The signal is then divided into overlapping windows, where the overlap of each window is chosen to be twice the sampling rate of the data glove.

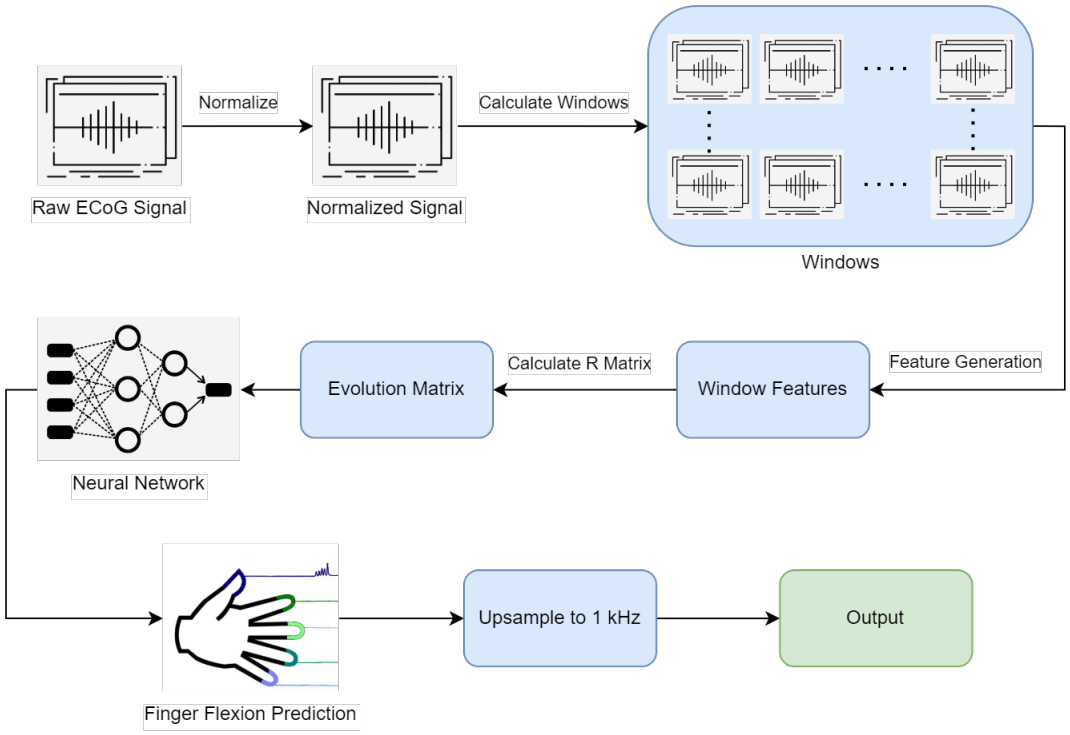


Figure 1. Flow Chart summarizing our Proposed Pipeline

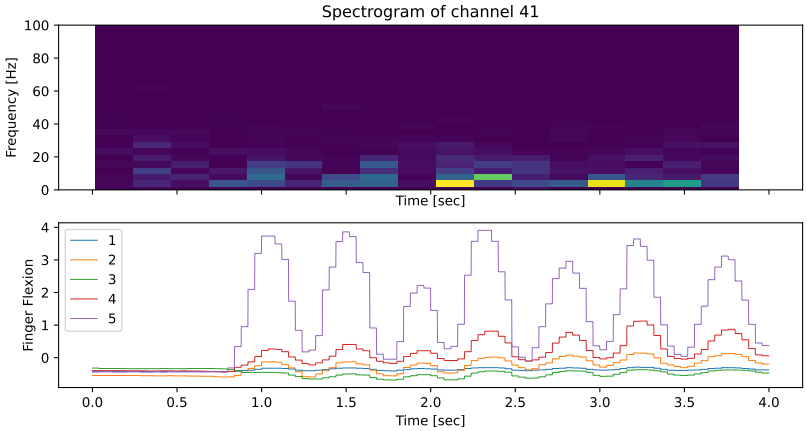


Figure 2. We see that the most of the interesting activity occurs in the lower frequency bands and the amplitude in the band is higher around peaks.

#### 2.2.2 Feature Generation

We compute a number of features for each window, across each channel to capture various aspects of the signal. The functions for these features are illustrated in Figure 3. These features are then combined into an Evolution Matrix, using the method described in Kubanek *et al.* [4] to capture the features of previous 300 ms of signal corresponding to every window.

#### 2.2.3 Feature Selection

The data has numerous recording channels and using our approach of pre-processing, we get a high dimensional feature vector for each window of data. If we rely on our model to ignore irrelevant data from the feature vector, the

$$\texttt{line-length}(x) = \sum |\Delta \vec{x}|$$
$$\texttt{signed-area}(x) = \sum \vec{x}$$
$$\texttt{area}(x) = \sum |\vec{x}|$$
$$\texttt{energy}(x) = \|\vec{x}\|_2^2$$
$$\texttt{max-deflection}(x) = \max(\vec{x}) - \min(\vec{x})$$
$$\texttt{signal-deviation}(x) = \sigma(\vec{x})$$
$$\texttt{band-power}(\text{pass-band}, x) = \|\text{filter-noise}(\text{pass-band}, \vec{x})\|_2^2$$

Figure 3. Feature Functions

learning problem increases in complexity. To avoid this we carefully prune the features using the information we learn during data analysis. We know that most relevant neural activity takes place in the low-frequency range, so we pick features that capture low frequency information such as `signed-area`, `area`, and `energy`. We also found that adding some higher frequency features improves the model-loss via cross-validation. Therefore, we selected the following features for our model: `signed-area`, `area`, `energy`, `line-length`, and `signal-deviation`.

#### 2.2.4 Multi-layer Perceptron Regressor

A Multi-Layer Perceptron (MLP) is used to predict finger flexions in each of the 3 subjects. The MLP is an extension of a Linear Perceptron, with 4 hidden layers. The model's hidden layers have hidden dimensions of 256, 1024, 512, and 256 respectively, which were selected by an extensive cross-validation study.

Simpler models like like Linear Regression or K-Nearest Neighbor fail to fit the training data efficiently due to lack of trainable parameters as well as low model complexity. Deep Neural Networks are well known to learn complex hypothesis functions over data similar to the given ECoG signal. We employ MLP to find a good balance of learning the intricacies of the training data while ensuring that we do not over-fit to the training set by using a 90:10 cross-validation split.

#### 2.2.5 Post-Processing

Our model generates predictions at twice the frequency of the data glove (50 Hz), and hence we use cubic interpolation to upsample the 50 predictions to 1 kHz, as recommended in the Project Description.

## 3. Alternative Methods Explored

### 3.1. Models

#### 3.1.1 K-Nearest Neighbor

K-Nearest Neighbor models can be used for both classification and regression tasks, assigning new values for inputs based on their similarity to other data points in the training set. With lower values of `k`, like 5, we expect the model to fit the training data well, since lower values of `k` are more prone to overfitting. However, as evident in Table 1, the model is not able to fit even the training data well. Hence it's performance on the hidden leaderboard set was subpar, and we discarded this approach.

#### 3.1.2 Least Squares Linear Regression

Least Squares Linear Regression was the original algorithm we used to clear Checkpoint 1, using the `LinearRegression` model from `scikit-learn`. The equation of the linear model can be described as $Y = WX + B$, where $Y$ is a vector of response variables, $X$ is a matrix of input variables, $W$ is the matrix of regression coefficients or weights, and $B$ is a bias term.

However, the approximated hypothesis function of Linear Regression is not complex enough to give predictions with higher correlation values, and we see comparatively high $MSE$ loss values as evident in Table 1. Hence we started to explore Neural Network approaches for more complex hypothesis functions.

#### 3.1.3 Video Vision Transformer

Since the ECoG signal is similar to an audio signal, the process of converting the signal to a spectrogram and essentially converting it into a series of images is a popular approach, as illustrated in work like VGG-Sound [2] and Palanisamy *et al.* [7]. Following a similar approach, we experimented with the ViViT model [1], given the recent success of vision-based transformers [3]. Transformers often require large amounts of training data to generalize over

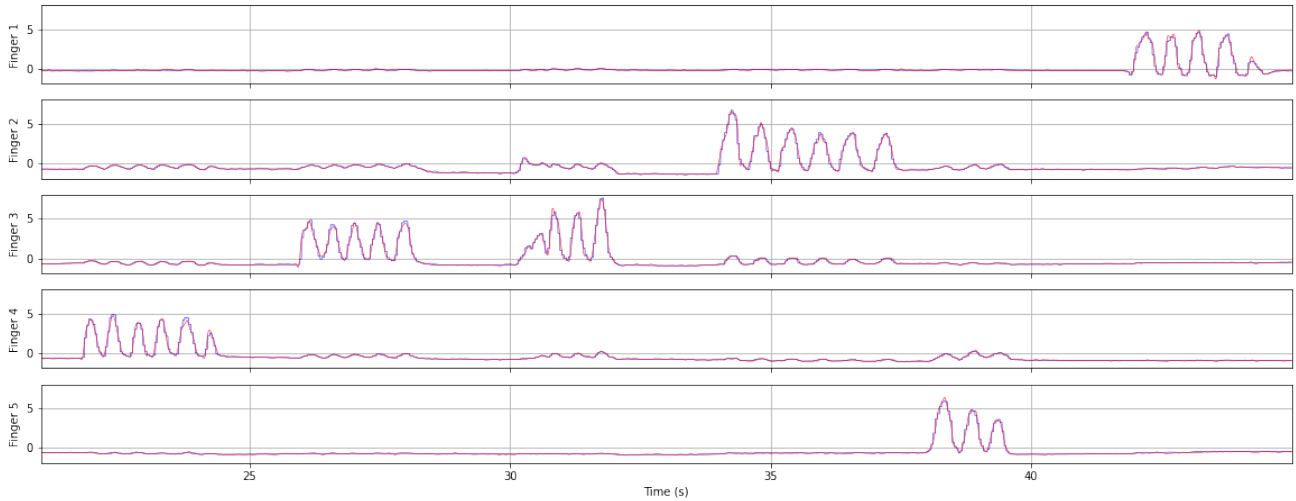


Figure 4. Comparison of Finger Flexion Predictions. Red indicates model predictions, and Blue denotes the actual data glove traces. We see that the model is able to learn the training data while not over-fitting to the sampling artefacts.

unseen examples. However, since our application is data constrained, the model does not perform well.

### 3.2. Filtering Methods

#### 3.2.1 Noise Removal

We tried to remove electrical noise at 60 Hz using a notch filter [5] and following our dataset analysis we removed high-frequency noise using a Butterworth Filter of order 100. However, upon calculating the new features and running our pipeline, we decided to drop the noise filtering approach and calculated features from the raw ECoG signal itself. Filtering the data as such causes us to loose some information in exchange for a simpler problem. Our Multilayer Perceptron model as well as our choice of feature is expressive enough to ignore high frequency noise on it's own, we felt like it was a reasonable decision to drop the filtering step.

#### 3.2.2 Channel Band Splitting

Given our observation during data analysis we tried to split each ECoG channel into various frequency band. In particular we decomposed the signal into $Delta$, $Theta$, $Alpha$, $Beta$, and $Gamma$ bands. We drop the $Gamma$ band and computed all features independently for each channel and band. These features were then used to evaluate numerous models, and we noticed that the models take longer to train and generally had a higher loss value as compared to corresponding models using combined data. We speculate that the increase in input dimensions had a larger negative effect than the positive effect of decomposing the signal.

## 4. Evaluation and Analysis

We evaluate the performance of the models on a validation set using the **Mean Squared Error** (MSE) averaged over all samples. The MSE represents the square root of the second sample moment of the differences between predicted values and observed values or the quadratic mean of these differences. Since the leaderboard was updated periodically, we used the MSE values of each model to evaluate their performance on a validation set broken off from the training data.

Our experiments also included Channel selection for each subject initially in our pipeline; however we did not notice any improvement in performance, and hence it was removed from the pipeline.

Quantitative results of the baseline models are illustrated in Table 1, and Table 2 illustrates the performance of our proposed method.

Through our empirical results in Table 2, we observed that the `(256, 1024, 512, 256)` architecture has the best performance out of all other evaluated models. Observing Figure 4, we see that the predicted signal is very similar to the ground truth, hence the model is slightly over-fitting.

| Method | History (ms) | Subject 1 ↓ | Subject 2 ↓ | Subject 3 ↓ |
|---|---|---|---|---|
| 5-Nearest Neighbor + Evolution Matrix | 300 | 0.2146 | 0.1368 | 0.1180 |
| Linear Regression + Raw Data (1kHz) | None | 0.9332 | 0.8511 | 0.8428 |
| Linear Regression + Raw Data (50Hz) | None | 0.8120 | 0.7918 | 0.7234 |
| Linear Regression + Evolution Matrix | 150 | 0.6255 | 0.6221 | 0.4536 |
| Linear Regression + Evolution Matrix | 300 | 0.4877 | 0.4998 | 0.3280 |

Table 1. Baseline Model Performance

| Architecture | Subject 1 ↓ | Subject 2 ↓ | Subject 3 ↓ |
|---|---|---|---|
| (180, 10) | 0.0450 | 0.0328 | 0.0287 |
| (128, 256, 256, 256) | 0.0251 | 0.0114 | 0.0084 |
| (256, 512, 512, 256) | 0.0233 | 0.0128 | 0.009 |
| **(256, 1024, 512, 256)** | **0.0181** | **0.0077** | **0.0072** |

Table 2. Multi-Layer Perceptron Performance with an Evolution Matrix and History of 300ms

We also observe that some models perform better for certain fingers in certain subjects, which could be a potential improvement to our pipeline.

## 5. Discussions

We also observe a correlation between the fourth, third and fifth finger. Physiologically, control of the fourth finger uses muscles shared with the third and fifth finger. These shared muscle include two long flexors in the forearm and a set of muscles named lumbricals in the hand. Meanwhile, the thumb, index, and little fingers have independent extensors, the middle finger and fourth finger do not have independent flexors or extensors, as opposed to the rest of the digits, as demonstrated in the work of Van Duinen *et al.* [10]. The general correlation of the fourth finger's flexion with the third and fifth fingers can be further explained by the fingers' connection to the brain through two nerves: the radial nerve, which connects the thumb, index finger and one side of the middle finger, and the ulnar nerve, which connects with the other side of middle finger, ring finger, and the little finger. Because the nerves for the ring and little finger are intertwined, it becomes difficult to move each of these fingers separately. The same phenomenon occurs between the ring and middle finger. The branching between these nerves results in the dependence of the three fingers on each other for movement therefore we observe a strong co-relation between their flexions.

For further details about the topics discussed in the report, please refer to the Appendices.

## 6. Conclusions

In this project, we devised a method to predict finger flexion movements of a human hand from raw ECoG signals. Various feature extraction techniques and algorithms used in current literature were examined and evaluated for data processing and improving predictions. Finally, our model parameters were tuned simultaneously for all 5 fingers of a particular patient. We tried to find a balance between model complexity and data processing, to yield optimal results.

Although we obtained decent results from our algorithm, there is much room for improvement. Some prospective approaches which could improve results include identifying neuronal cells and spiking patterns from the data to improve model inputs, tuning individual model architectures for each finger of a particular subject, exploring *Convolutional Neural Networks* [6] for the spectrograms of the ECoG signal, and using more optimal filtering techniques to process the input signal.

Although the competitive format of the project resulted in a more intense and slightly rushed approach to solving the problem, devising a structured pipeline which was compatible with different models we employed made our overall algorithm more modular and have cleaner code. Our pipeline has hence been open-sourced and is available on `Github`.

## References


[1] Anurag Arnab, Mostafa Dehghani, Georg Heigold, Chen Sun, Mario Lučić, and Cordelia Schmid. Vivit: A video vision transformer, 2021. 3

[2] Honglie Chen, Weidi Xie, Andrea Vedaldi, and Andrew Zisserman. Vggsound: A large-scale audio-visual dataset. *CoRR*, abs/2004.14368, 2020. 3

[3] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. *CoRR*, abs/2010.11929, 2020. 3

[4] J Kubánek, K J Miller, J G Ojemann, J R Wolpaw, and G Schalk. Decoding flexion of individual fingers using electrocorticographic signals in humans. *J. Neural Eng.*, 6(6):066001, Dec. 2009. 2

[5] Gang Li. A stable and efficient adaptive notch filter for direct frequency estimation. *IEEE Transactions on Signal Processing*, 45(8):2001–2009, 1997. 4

[6] Xiangmin Lun, Zhenglin Yu, Tao Chen, Fang Wang, and Yimin Hou. A simplified cnn classification method for mi-eeg via the electrode pairs signals. *Frontiers in Human Neuroscience*, 14, 2020. 5

[7] Kamalesh Palanisamy, Dipika Singhania, and Angela Yao. Rethinking CNN models for audio classification. *CoRR*, abs/2007.11154, 2020. 3

[8] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011. 1

[9] Gerwin Schalk, Jan Kubanek, Kai Miller, N.R. Anderson, Eric Leuthardt, Jeffrey Ojemann, D Limbrick, DW Moran, Lester Gerhardt, and Jonathan Wolpaw. Decoding two-dimensional movement trajectories using electrocorticographic signals in humans. *Journal of neural engineering*, 4:264–75, 10 2007. 1

[10] Hiske van Duinen, Wei Shin Yu, and Simon C Gandevia. Limited ability to extend the digits of the human hand independently with extensor digitorum. *J. Physiol.*, 587(Pt 20):4799–4810, Oct. 2009. 5


## A. Appendix A

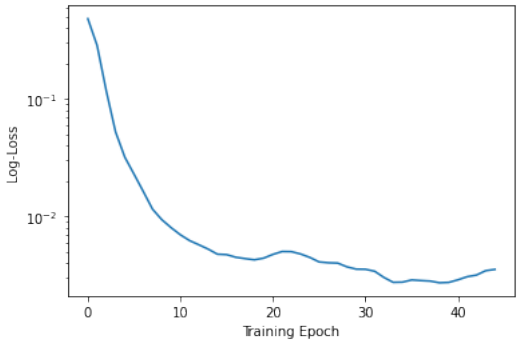


Figure 5. Training Loss of MLP for Subject 1

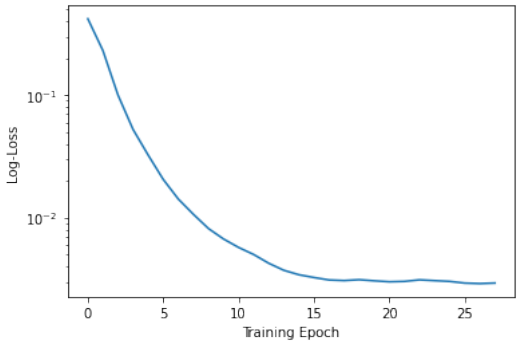


Figure 6. Training Loss of MLP for Subject 2

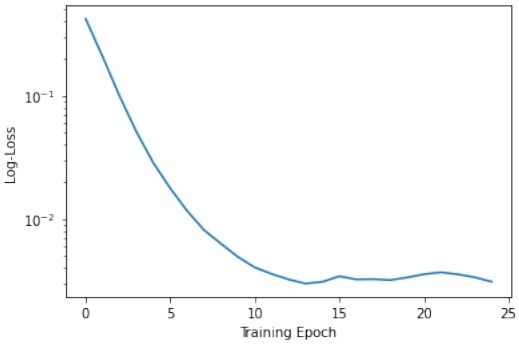


Figure 7. Training Loss of MLP for Subject 3