ITSP Final Documentation

n3URL

The Mentalists

ITS21038

Keywords (Include 7 or more keywords that will help others find your documentation easily)
Brain-Computer Interface (BCI), Electroencephalogram (EEG), Machine Learning (ML), Convolution Neural Networks (CNN), Pygame, MNE

Team Member Name	Roll Number	Email-Id
Pranava Singhal	200070057	200070057@iitb.ac.in
Sankalp Parashar	200050127	200050127@iitb.ac.in
Prakhar Bansal	200070056	200070056@iitb.ac.in
Aditya Sriram	200070004	200070004@iitb.ac.in

Inspiration for Idea

When we first heard of the amazing research Neuralink has been doing we got interested in exploring Brain-Computer Interfacing. Our brains have unparalleled information processing capacity. However, it is our limbs that restrict how fast we can perform a task. BCI allows us to get rid of the intermediary, resulting in faster performance. This is very Matrix-like but has profound medical implications besides its WOW factor. The primary targets are patients suffering from both short-term and long-term paralysis. Other possible applications include security, control and regulation, marketing and advertising. Finally, we hope to gain a better understanding of the brain.
https://docs.google.com/document/d/1FqKziApLEUZzOYAvBbhaYBstyV6WlwcEHt-mOJ6TS24/edit

Problem Statement

This project is a proof of concept of the viability of Brain-Computer Interfacing as an alternative to conventional forms of control. We aim to accomplish something very similar to what Neuralink recently demonstrated - playing a game with our minds. We have chosen Atari's Breakout as the game of choice for this project.

Existing solutions in the Market

There are plenty of EEG devices available for home experimentation (OpenBCI, NeuroSky). However, there are not a lot of products that do what we intend on doing. Elon Musk's company Neuralink is at the forefront of the industry. Their progress is inspiring. (See their latest video for a taste of what they have accomplished) Unlike Neuralink, we do not plan on building a microchip to implant in someone's head. Our technology is non-invasive and hence, safer.

Proposed Solution

The proposed solution is for EEG data recordings. The model will also work for real-time data from an EEG headset with suitable modifications:

1. Data-preprocessing (Done in Python using MNE or MATLAB)
    a. Load the EEG data and apply a bandpass filter to remove the powerline noise artefact.
    b. Split the EEG track into segments 4 seconds long (this is to conform with the data recording procedure used to generate the Physionet dataset).
    c. Each segment corresponds to left, right, none motor imagery. Segregate the data for each class and select the appropriate electrodes.
    d. Create a train/validation/test split (we chose a train/test split since we were implementing a hyperparameter-tuned model).
2. Training phase (Done in Tensorflow & Keras or MATLAB)
    a. Implement the model (Pre-trained GitHub model, Transfer Learning with GoogLeNet, research-paper model).
    b. Train the model on the training data.
    c. Save model weights.
3. Testing Phase (Point of difference between real-time and simulation)
    a. Load model weights into the game.

b. Identify the move (left/right) which minimises the horizontal separation between the paddle and the ball (we have access to the game state variables).
c. Pick a random example from the test set corresponding to the desired move.
d. Return the prediction and move the paddle.

Brief Description

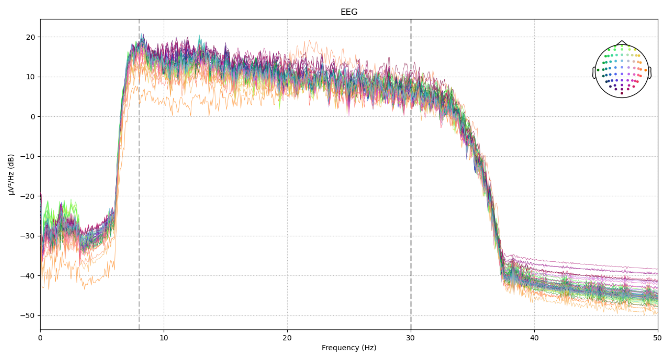
An Electroencephalography (EEG) controlled Machine Learning model to play Atari's Breakout game. Usually, the user plays the game using the left and right arrow keys on the keyboard. We propose an alternative form of control that relies only on your mind removing the need to physically press keys or detect movement. Here, we explore one potential application of Brain-Computer Interfaces (BCI) pertinent to the gaming and entertainment sector. This project can be described as our goal to understand Brain-Computer Interfacing (BCI) and introduce it as a viable alternative to existing technology.
To get a sense of what we intend on doing, please watch the following video: https://www.youtube.com/watch?v=Dgo7F-lpyYE

Progress

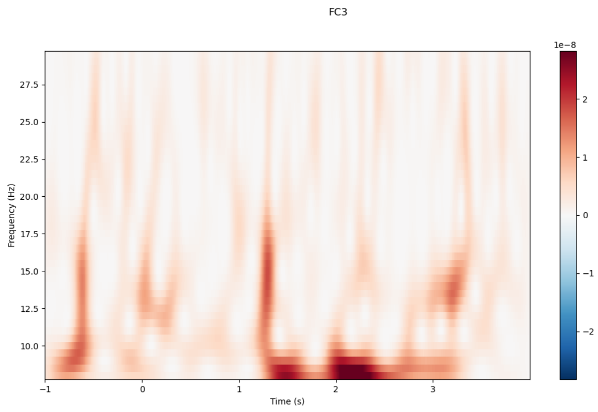
Work Done

We started by looking at Brain-Computer Interfacing (BCI) applications in Motor Imagery and Game Control. We spent a considerable amount of time exploring EEG datasets to identify one that best suited our requirements. We finally chose the Motor Movement and Imagery Dataset by Physionet recorded using the BCI2000 headset.

Power Spectral Density of EEG Data



Wavelet Transform of EEG Data



Having chosen our dataset, we had to identify the method of EEG signal classification. After careful consideration, we decided to use Convolutional Neural Networks (CNNs) for the following reasons:

1. Generally, classification pipelines in Machine Learning involve two main steps - Feature Extraction followed by the Classification algorithm, examples of which include Random Forest Classifiers, Support Vector Machines (SVMs). Convolutional Neural Networks effectively combine both these stages, achieving much higher accuracy.
2. We lacked the experience to train a suitable CNN from scratch and found successful implementations to guide us.
3. Prakhar and Sankalp were already doing a learning project on Convolutional Neural Networks and could effectively apply their learnings to our proposal.

We were inspired to combine CNNs with already existing Feature Extraction techniques to boost the capabilities of our model. We learned about various Signal Processing techniques - the Fast-Fourier Transform, Short-Time Fourier Transform and the Wavelet Transform. To this end, we explored two classes of models:

1. (Done in Keras & Tensorflow) A research paper and a pre-trained CNN model. The input consisted of raw EEG data with minimal preprocessing - bandpass filtering and resampling.
2. (Done in MATLAB) A transfer learning model based on GoogLeNet. The input consisted of Continuous Wavelet Transform Spectrogram images.

To our surprise, the Keras models outperformed the transfer learning model. We were able to achieve a maximum test accuracy of ~75% after hyperparameter tuning.
The final stage of our project consisted of integrating the trained CNN model with Breakout. Playing games has a significant feedback component where the user reacts to what appears on the screen. We did not have access to an EEG headset and could only simulate this experience with a clever scheme. We chose random examples from the test set, allowed the CNN model to classify and moved the paddle according to the prediction.
We met regularly to keep track of our progress. To document our work, we created a GitHub repository containing the epoched data that we generated, CNN models and some experimentation with other software packages.

None of us want ITSP to mark the end of our work and we hope to continue developing and maintaining the project. We plan on participating in ITSP++ and experimenting with real-time EEG data. We would also like to explore the hardware aspects of this project, like building an EEG.

Work distribution in the team

**Detailed description can be found in Contributions by each Team Member*

Work-Flow distributed across the duration between the review meets

1. Before review meet 1
    a. Choosing the final game to be played and implementing it in Pygame
    b. Exploring potential EEG datasets
    c. Reading about the basics of Machine Learning
    d. Understanding the fundamentals of Signal Processing
    e. Tutorials on Python packages like MNE, PyEDFlib
2. Before review meet 2
    a. Implement the data preprocessing stage of the pipeline
    b. Start learning about Deep Learning

    c. Read research papers on CNN models suited for this task. Implement a basic Keras Model
    d. Brush up on MATLAB and generate Wavelet Transform Spectrogram images
3. Until now
    a. Implement the Transfer Learning model using GoogLeNet
    b. Hyperparameter tuning to improve validation accuracy
    c. Integrate the final model with Breakout and prepare evaluation metrics

Challenges (Difficulties faced and how you overcame them)

- Poor accuracy in model training. Needed to do quite a lot of hyperparameter tuning in order to get the best possible results.
- Difficulty in procuring an EEG headset to generate real-time EEG data. Eventually simulated the EEG data using information from the game backend.

Model Architecture

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_1 (InputLayer)         [(None, 10, 640, 1)]      0
_________________________________________________________________
conv2d (Conv2D)              (None, 10, 636, 25)       150
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 1, 636, 25)        6275
_________________________________________________________________
batch_normalization (BatchNo (None, 1, 636, 25)        100
_________________________________________________________________
activation (Activation)      (None, 1, 636, 25)        0
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 1, 318, 25)        0
_________________________________________________________________
dropout (Dropout)            (None, 1, 318, 25)        0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 1, 314, 50)        6300
_________________________________________________________________
batch_normalization_1 (Batch (None, 1, 314, 50)        200
_________________________________________________________________
activation_1 (Activation)    (None, 1, 314, 50)        0
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 1, 157, 50)        0
_________________________________________________________________
dropout_1 (Dropout)          (None, 1, 157, 50)        0
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 1, 153, 100)       25100
_________________________________________________________________
batch_normalization_2 (Batch (None, 1, 153, 100)       400
_________________________________________________________________
activation_2 (Activation)    (None, 1, 153, 100)       0
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 1, 76, 100)        0
_________________________________________________________________
dropout_2 (Dropout)          (None, 1, 76, 100)        0
_________________________________________________________________
conv2d_4 (Conv2D)            (None, 1, 72, 200)        100200
_________________________________________________________________
batch_normalization_3 (Batch (None, 1, 72, 200)        800
_________________________________________________________________
activation_3 (Activation)    (None, 1, 72, 200)        0
_________________________________________________________________
max_pooling2d_3 (MaxPooling2 (None, 1, 36, 200)        0
_________________________________________________________________
dropout_3 (Dropout)          (None, 1, 36, 200)        0
_________________________________________________________________
flatten (Flatten)            (None, 7200)              0
_________________________________________________________________
dense (Dense)                (None, 2)                 14402
_________________________________________________________________
activation_4 (Activation)    (None, 2)                 0
=================================================================
```

Calculations involved

$$(10.1)\quad Accuracy = \frac{T_p + T_n}{T_p + T_n + F_p + F_n}$$

$$(10.2)\quad Precision = \frac{T_p}{T_p + F_p}$$

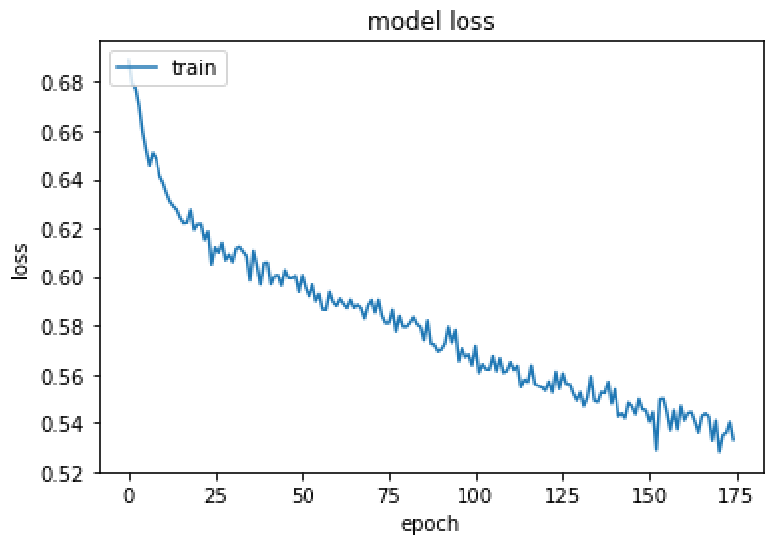
$$(10.3)\quad Recall = \frac{T_p}{T_p + T_n}$$

$$(10.4)\quad F_1 = 2 \cdot \frac{precision \cdot recall}{precision + recall}$$

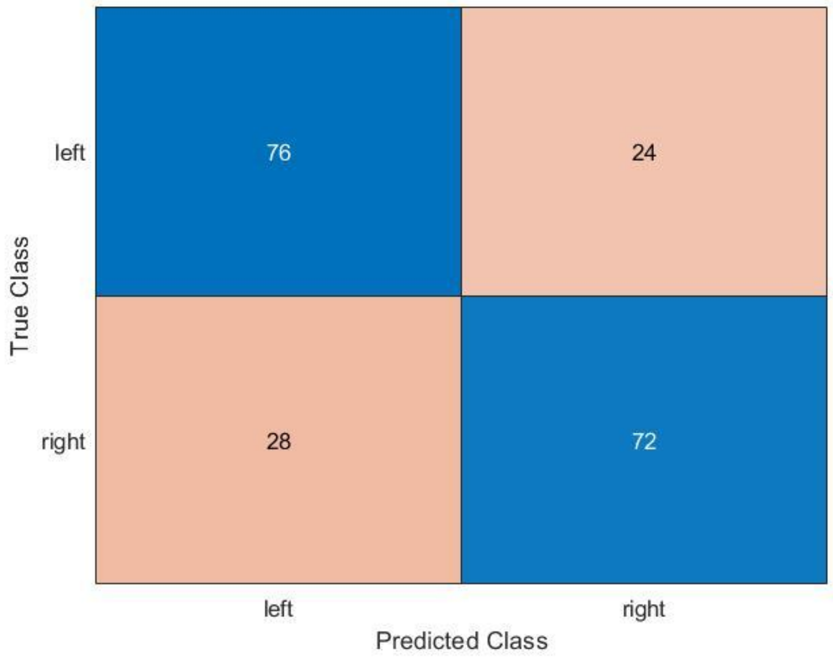
```
              precision    recall  f1-score   support

       right       0.76      0.72      0.74       471
        left       0.74      0.78      0.76       480

   micro avg       0.75      0.75      0.75       951
   macro avg       0.75      0.75      0.75       951
weighted avg       0.75      0.75      0.75       951
 samples avg       0.75      0.75      0.75       951
```

Results

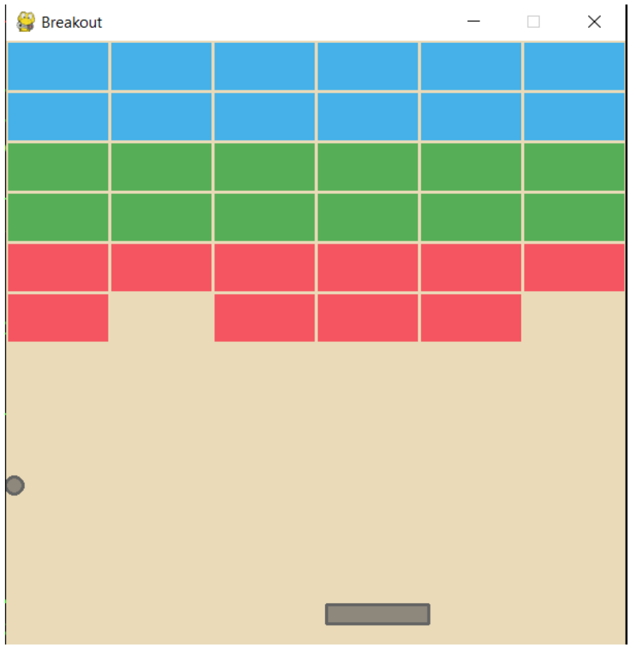
Training Curve



Confusion Matrix



Main Game



Project Repository

[Project Repository]

Simulations/ Project Video

Presentation

Final Presentation

Learning Value

1. Comprehension of Research Papers and Literature
2. Analysis of available datasets to determine their suitability for the task at hand
3. Game development using Python's Pygame Library
4. Understanding of EEG data - the connection between Frequency and Brain Activity
5. Signal Processing Techniques including the Discrete-Fourier Transform, Fast-Fourier Transform, Short-Term Fourier Transform, Wavelet Transform, Common Spatial Patterns
6. Noise Removal and Artefact Reduction for EEG data

7. Learned the basic concepts of Deep learning and applied these to build a CNN classifier from scratch
8. Proficiency in languages like Python and MATLAB after working on a large-scale application and understanding of source code documentation
9. Importance of Version Control using Git and GitHub in documentation and collaboration
10. Soft Skills - Working effectively as a team, delegation, effective time management and presentation

Software/ Hardware used

1. Main Game Implementation - Pygame
2. EEG Data Collection - BCI2000 Headset
3. EEG Data Epoching and Preprocessing -
    a. MNE, Python
    b. MATLAB
4. Convolutional Neural Network Implementation -
    a. Tensorflow & Keras (Research Paper Model, Pre-existing GitHub Model)
    b. MATLAB (Transfer Learning)
5. Data Visualisation - Matplotlib, MATLAB

Suggestions for others

- Rather than focusing on procuring an EEG headset to acquire real-time data try to work with pre-recorded data first. This is easier to obtain and also makes it easier to focus on developing the software first. It is not very difficult to integrate this with the hardware interface later.
- Start with a well-defined problem statement so that you are able to track progress through each stage.
- Organise work and collaborate using version control with Git.
- In order to implement the project, it is not essential to gain a deep understanding of the mathematics involved (in signal processing and deep learning). However, it is a good idea to understand what effect each process has on the data.

Contribution by each Team Member

Pranava Singhal

1. Coordinated tasks during the project duration
2. Explored the feasibility of online EEG datasets for the project
3. Studied various Signal Processing Techniques
4. Generated Wavelet Transform Spectrogram images in MATLAB
5. Implemented the GoogLeNet Transfer Learning model using MATLAB

Prakhar Bansal & Sankalp Parashar

1. Leveraged their expertise on Convolutional Neural Networks
2. Improved the accuracy of the original Keras model from 50% to nearly 70% by hyperparameter tuning
3. Implemented the final DeepConvNet model on Keras
4. Tuned the final model achieving a testing accuracy of 75%
5. Generated plots and calculated model metrics

Aditya Sriram

1. Added the implementation of Atari's Breakout in Pygame
2. Tested the capabilities of Python EEG Processing packages like PyEDFlib, MNE
3. Implemented various data preprocessing schemes using FFT, STFT, Wavelet Transform on python
4. Generated epoched EEG data from raw files to generate the initial dataset
5. Implemented the starting Convolutional Neural Network Architecture (CNN) on Keras without any hyperparameter tuning
6. Integrated the final CNN model with the game
7. Maintained the project GitHub repository for the duration of the project

References and Citations

1. https://github.com/russs123/Breakout (Breakout Game)
2. https://github.com/vlawhern/arl-eegmodels (Reference CNN Architecture)
3. *https://www.frontiersin.org/articles/10.3389/fnhum.2020.00338/full* (Reference CNN Architecture)
4. *https://github.com/Sentdex/BCI* (gave us an idea of project feasibility)
5. https://www.biorxiv.org/content/10.1101/326066v1.full.pdf (simulating eeg data)
6. https://www.frontiersin.org/articles/10.3389/fneur.2019.00325/full (modelling the problem and pre-processing)

7. https://www.frontiersin.org/articles/10.3389/fnhum.2020.00338/full (CNNs)
8. https://downloads.hindawi.com/journals/cmmm/2020/1981728.pdf (CNNs - alternative implementation)
9. https://mne.tools/stable/auto_examples/decoding/decoding_csp_eeg.html#sphx-glr-auto-examples-decoding-decoding-csp-eeg-py (MNE library for data processing)
10. https://www.frontiersin.org/articles/10.3389/fnins.2021.655599/full
11. PhysioNet - Goldberger AL, Amaral LAN, Glass L, Hausdorff JM, Ivanov PCh, Mark RG, Mietus JE, Moody GB, Peng C-K, Stanley HE. PhysioBank, PhysioToolkit, and PhysioNet: Components of a New Research Resource for Complex Physiologic Signals. Circulation 101(23):e215-e220 [Circulation Electronic Pages; http://circ.ahajournals.org/cgi/content/full/101/23/e215]; 2000 (June 13).
12. BCI2000 - Schalk, G., McFarland, D.J., Hinterberger, T., Birbaumer, N., Wolpaw, J.R. BCI2000: A General-Purpose Brain-Computer Interface (BCI) System. *IEEE Transactions on Biomedical Engineering* **51**(6):1034-1043, 2004. [In 2008, this paper received the Best Paper Award from IEEE TBME.]
13. EEG dataset - https://archive.physionet.org/pn4/eegmmidb/
14. https://subscription.packtpub.com/book/big-data-and-business-intelligence/9781785282287/10/ch10lvl1sec133/computing-precision-recall-and-f1-score

Disclaimer

Fair Use of the DeepConvNet Model
Fair Use of Physionet EEG Motor Movement/Imagery Dataset v1.0.0

Licenses

NumPy (BSD-3 Clause License)
Matplotlib (MDT)
MATLAB. (2010). version 9.10.0.1669831 (R2021a) Update 2. Natick, Massachusetts: The MathWorks Inc.