Faculty of Engineering
Cairo University

Medical Instrumentations
Project 2
Signal Acquisition and BCI of Grasp and Lift

Team 16

Nouran Mahmoud

Nada Amr

Hadeer Sherif

Mariam Hatem

Sarah Mohamed Ragaei

Introduction

People undergoing neuromuscular dysfunctions and amputated limbs require automatic prosthetic appliances. In developing such prostheses, the precise detection of brain motor actions is imperative for the Grasp-and-Lift (GAL) tasks. Because of the low-cost and non-invasive essence of Electroencephalography (EEG), it is widely preferred for detecting motor actions during the controls of prosthetic tools.

In this project, we developed a system that utilizes EEG data to accurately classify motor actions as either grasp or release. The EEG signals are processed and fed into a Convolutional Neural Network (CNN) model designed to identify these specific motor events. Once the model predicts the action it is translated into physical movement of a prototype hand via an Arduino-controlled servo motor.

This project demonstrates a low-cost, efficient approach to translating brain signals into physical movements, providing a potential pathway for developing advanced prosthetic devices that can respond to the user's neural commands with high precision. By leveraging EEG and machine learning techniques, we aim to enhance the quality of life for individuals with motor impairments, offering them greater autonomy and functionality in their daily activities.

Dataset description Dataset

The dataset is intended to decode sensation, intention, and action from scalp EEG signals.

- The rows represent the time frame (500HZ)
- The columns represent channels(32)+labels(6).
- There are 12 subjects in total
- 10 series of trials for each subject (1-8 train, 9-10 test)
- Approximately 30 trials within each series.
- The dataset contains 32-channel EEG signals, having six different events, which are enlisted in Table I.
- In the training set, there are two files for each subject + series combination:
    - The ***_data.csv** files contain the raw 32 channels of EEG

data (sampling rate 500Hz).

- the ***_events.csv** files contain the ground truth frame-wise labels for all events.

Although the dataset includes six different events, our project specifically focuses on two events: "FirstDigitTouch" and "BothRelease." Therefore, we preprocessed the data to exclude the unimportant events and concatenated the series of all patients into one file to increase the amount of data available for training the model.

GAL Events	Description
HandStart (HS)	Reaching for the object
FirstDigitTouch (FDT)	Grasp the object using thumb and index finger
BothStartLoadPhase (BSP)	Lifting an object for a couple of seconds
LiftOff (LO)	Lift force
Replace (R)	Set the object backward on the support surface.
BothRelease (BR)	Free the object and place hand at the starting location

TABLE I: Details of six events in the dataset

Signal Preprocessing

1. Denoising

- **Wavelet denoising** is a technique used to remove noise from a signal by utilizing the properties of wavelet transform. It is particularly effective in eliminating noise that has a non-stationary or time-varying characteristic.

- A function named **`wavelet_denoise()`** is defined to perform wavelet denoising on the data. It applies wavelet decomposition using the **`pywt.wavedec()`** function, sets a threshold value based on the standard deviation of coefficients, and applies soft thresholding to the detail coefficients. Finally, it reconstructs the denoised signal using **`pywt.waverec()`.**

2. Filtration

- **Band-pass filtration** is a signal processing technique used to selectively filter out frequency components outside a specific frequency range while allowing the desired frequency range to pass through.
- We chose the low cutoff frequency is 7Hz and the high cutoff frequency is 30 Hz to suit our data and fit our model.
- The sampling frequency is 500 Hz. A function named **`bandpass_filter()`** is defined to perform band-pass filtration on the data. It designs a Butterworth band-pass filter using the **`signal.butter()`** function and applies the filter to the data using **`signal.filtfilt()`.**

Model Development (CNN)

Introduction

To accurately interpret grasp and release EEG signals to control an artificial hand via an Arduino, we employed a Convolutional Neural Network (CNN) due to its powerful feature extraction and pattern recognition capabilities.

Model description

The CNN model processes the EEG signals to determine whether the signal indicates a grasp or lift action. The model's input size is (150, 32, 1), representing the height, width, and number of channels of the EEG data.
The model consists of the following layers:

- **Convolutional Layers:** These layers apply filters of size (3x3) to extract features from the EEG signals. Each convolutional layer uses the ReLU activation function to introduce non-linearity and ensure positive feature values.

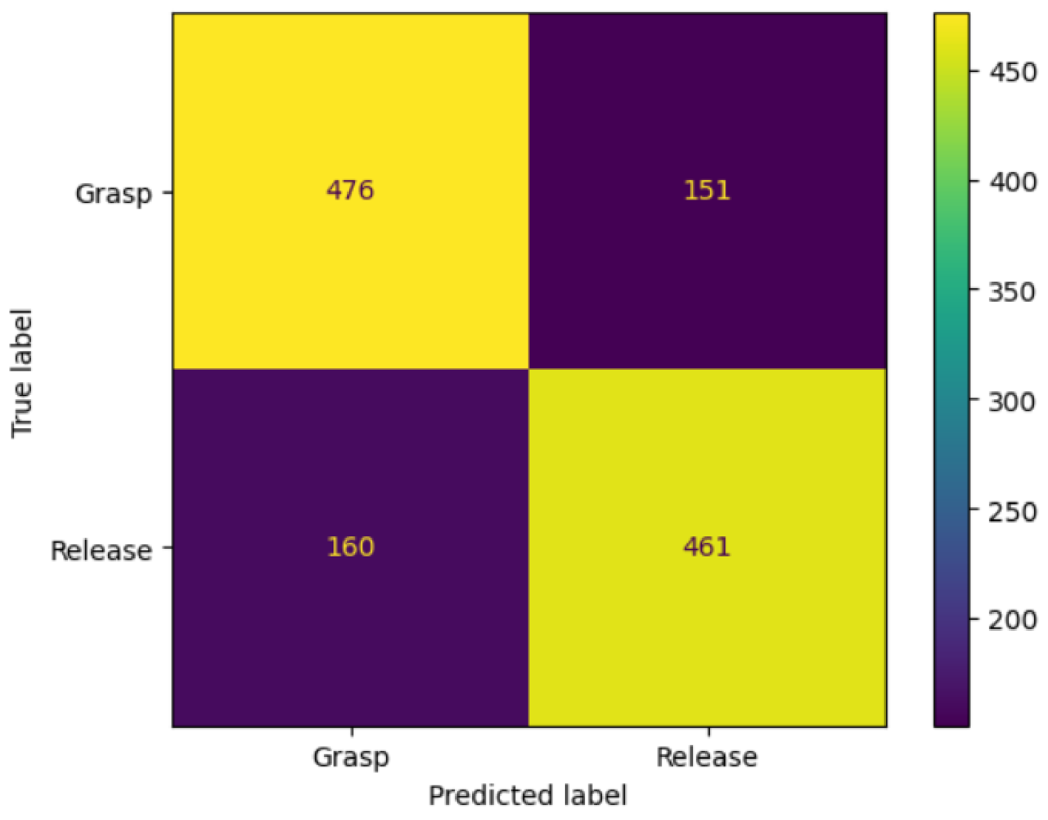
- **Average Pooling Layers:** These layers reduce the spatial dimensions of the feature maps while retaining the essential features. This down-sampling process helps in reducing computational complexity and mitigating overfitting.
- **Flattening Layer:** This layer converts the 2D feature maps into a 1D vector, preparing the data for the fully connected layers.
- **Fully Connected Layers:** These layers further process the extracted features to make final classifications. The last fully connected layer uses the SoftMax activation function to output probabilities for the grasp and lift classes.
- **Dropout Layer:** This layer, with a dropout rate of 0.6, prevents overfitting by randomly deactivating 60% of the neurons during each training iteration, ensuring robust learning.

Compilation and Training

- **Loss Function**: The model uses Sparse Categorical Crossentropy to measure the error between the true and predicted classes.
- **Optimizer**: The Adam optimizer, with a learning rate of 0.000001, updates the network's parameters efficiently, combining the benefits of SGD with momentum and RMSProp.

Results

By the end of 80 epochs, the training accuracy was 89.8% and validation accuracy was 75%



Hardware Integration

1. Arduino and Serial Communication

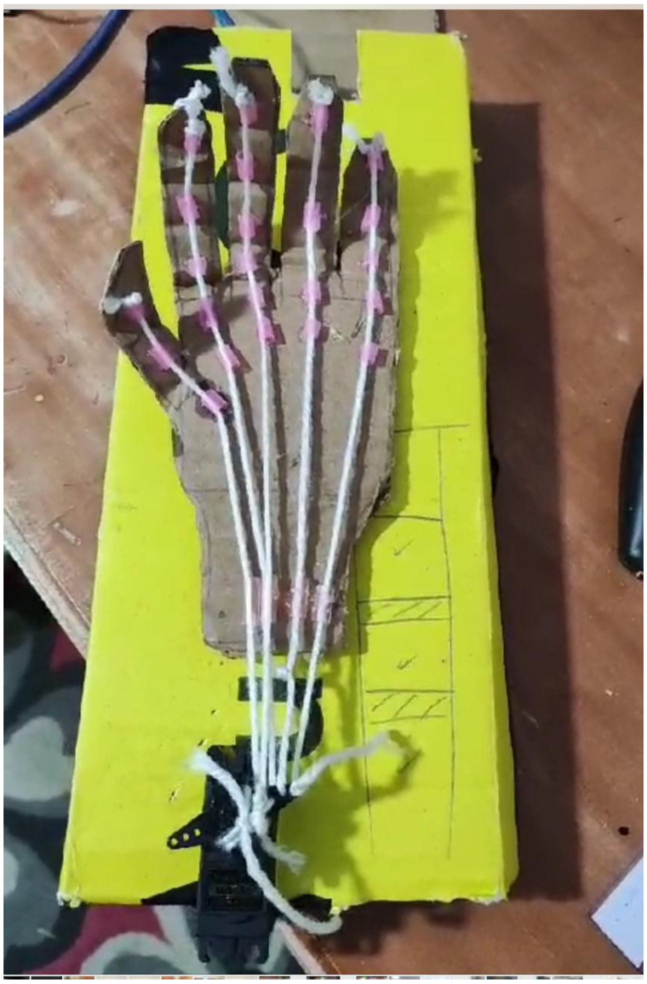
To bridge the gap between the EEG signal predictions and the physical movements of the prosthetic hand, we used an Arduino microcontroller to interpret the output from our CNN model and control the servo motor accordingly. The serial communication between the Python environment and the Arduino is established using the pyserial library. The predicted grasp and release signals from the CNN model are written to a CSV file, which is then read and sent to the Arduino through serial communication. The Arduino code listens for incoming serial data, interprets it, and adjusts the servo motor's position based on the received command. The commands are simple: a '0' to indicate a release action and a '1' to indicate a grasp action.

2. Hand Prototype

The hand prototype is designed to mimic human finger movements using a simple yet effective mechanism. The prototype consists of:

- **Cartoon Hand Frame**: The frame is constructed with straws to represent the bones of the hand. These straws provide both structure and flexibility, allowing realistic bending of the fingers.
- **Threads and Servo Motor**: Threads are connected to the fingertips, passing through the straws and fixed at their tips. When the servo motor rotates in a certain direction, it pulls the threads, causing the fingers to bend. This simulates the grasp action. When the servo motor rotates in the opposite direction, it releases the tension on the threads, allowing the fingers to return to their resting position, simulating the release action.

The servo motor is controlled by the Arduino based on the commands received from the serial communication, which are derived from the CNN model's predictions.



Integration and Workflow

The integration of the CNN model, Arduino, and hand prototype follows a systematic workflow:

1. **EEG Data Processing**: EEG data is processed using wavelet denoising and band-pass filtration.
2. **Prediction with CNN**: The preprocessed EEG data is fed into the CNN model, which predicts whether the action is a grasp or release.
3. **Serial Communication**: The prediction is written to a CSV file, read by the Python script, and sent to the Arduino via serial communication.
4. **Servo Motor Control**: The Arduino receives the command and adjusts the servo motor accordingly. A '0' command rotates the servo to release the fingers, and a '1' command rotates it to grasp.
5. **Hand Movement**: The servo motor's movement pulls or releases the threads in the hand prototype, resulting in realistic finger movements.

This integration ensures that the predicted motor actions are accurately translated into physical movements, providing a functional and responsive prosthetic hand prototype.