**Project report**

**Project name**:

Idea2 :EOG based simple game interface.

**Team Members :**

Nermeen Saber Fawzy 20191700694

Amir Mamdouh Helmy Shaker 20191700138

Abanob Gamal Fekry Fawzy 20191700001

Mohamed Sabih Mahmoud 20191700543

Youssef Saad Shaker 20191700771

1. **Data preparation:**

* We prepared lists of pairs for the 5 classes, where each pair signal for a single movement ( signal.h , signal.v)

We extracted the pairs of signals that have the same name with “h” & “v” suffixes, by looping over the data file and for each item that ends with "h", we created the name of its pair by removing the "h" suffix and replacing it with "v", then we checked if the pair name is in the data list. If it is, we added the pair as a tuple to the pairs list of the corresponding class.

Where each pair has the directory path of its signals,

We ignored signals which don’t belong to any class of the 5 classes.

* We note that the data is balanced so we didn’t need to balance it.

1. **Data Preprocessing:**

First, we read each signal and append all its amplitude values in a list.

**Normalization**:

Then we **normalize** the input signal to a range of 0 to 100, by subtracting the minimum value of the signal and dividing by the range between the maximum and minimum values. The result is then multiplied by 100 to scale the normalized signal to the desired range of 0 to 100.

**Filtering:**

**Butterworth** **band pass filter** is the most suitable filter for EOG signals ,so we used it.

Both horizontal and vertical signals are filtered by **Butterworth** **band pass filter** to where the preserved EOG band from 0.5 to 20 HZ.

**filter** **order** = 2, as in many studies orders **2** is sufficient.

**sampling** **rate = 176, which** is the number of samples per unit time.

**Btype parameter** = bandpass filter , that passes frequencies within a range and rejects frequencies outside that range.

First, we defined the **Nyquist frequency** = ½ sampling rate, to avoid aliasing.

Where it refers to the maximum frequency that can be accurately represented in a digital signal.

Then we **normalized the cutoff frequencies**(low , high) by dividing them by the Nyquist frequency,

So, when we normalized them ,we can ensure that the same cutoff frequencies correspond to the same portion of the frequency spectrum, regardless of the sampling rate.

Next, we used the **butter function** from the SciPy. Signal module to calculate the filter coefficients (numinator and demonator) based on the specified filter order, normalized cutoff frequencies, and filter type (bandpass in this case).

Finally, we used **the filtfilt function** that applies the filter coefficients to the input signal, where this function performs a zero-phase forward and reverse filtering to eliminate any phase distortion.

**Down Resampling:**

Resampling is the process of changing the sampling rate of a digital signal, which involves changing the number of samples taken per unit time.

We applied down sampling as it more suitable with EOG signals.

We know that the minimum suitable sampling rate should be above double the

maximum frequency preserved according to the considered frequency band for EOG(0-20)

So, we produced a new signal with 128 samples, where #samples =250 in the original signal. And this is when we extract features by Wavelets.

and resampled by 50 samples when we extracted them by Autoregressive.

As we note it affects accuracy.

1. **Feature Extraction:**

* **Wavelets:**

Each signal is resampled by 128 sample , so when we used wavelets with level 2 and got the first coefficient of it of signals in range 0-32 that match with EOG signals.

First, defined the wavelet function to be used (db1) and the desired decomposition level (level=2).

next, we used wavedec function to perform the wavelet decomposition of the input signal , where the input signal is decomposed into a set of high-pass and low-pass filtered signals at multiple scales.

Finally, the approximation coefficients at level 2 are returned as the output where the approximation coefficients represent the low-frequency components of the signal , they are typically used as a feature.

* **Auto Regression:**

We used its coefficients as the features where the coefficients of regressive model considers time domain features , if the signal isn’t complex .

The lag parameter specifies the order of the autoregression, or the number of lagged values used in the model. We used lags=4,where the AutoReg model will use the four previous values of the time series as input variables to predict the current value of the time series.

We concatenated each pair of signals after extracting the feature of them by appending them in one list respectively.

**Note:**

We note that the way of concatenation affects accuracy,

When we concatenated 2 lists of features by ‘+’ the accuracy was lower than when we concatenated them by appending them manually.

Then we saved all the extracted features in data\_x list and their labels in data\_y list and shuffled them.

where ,shuffling the data before splitting it ensures that the order of the data points does not influence the split.

1. **Splitting data into training , validation, and testing data:**

60-20-20

After shuffling the data, we split the data into training and testing sets using train\_test\_split with test\_size=0.2.

then Split the training data into training and validation sets using train\_test\_split again, with test\_size=0.2.

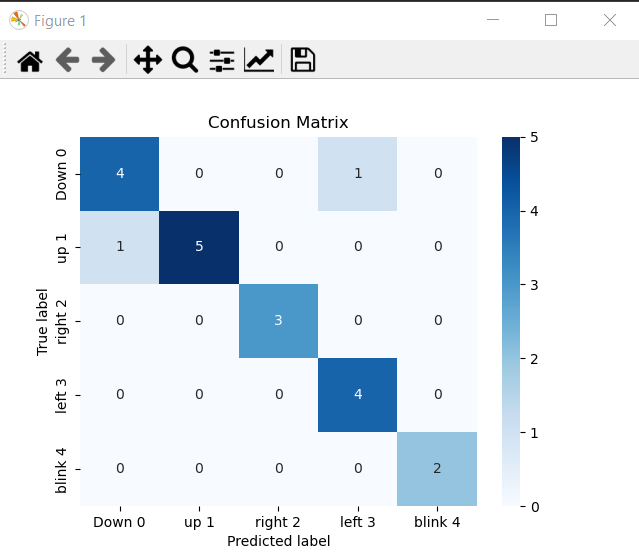
1. **Classifiers & its parameters:**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Classifier** | **Feature extraction method** | **Training accuracy** | **Validation accuracy** | **Testing accuracy** | **Resampling** |
| Gaussian Naive Bayes | wavelets | **0.984375** | **0.875** | **0.9** | **128** |
| SVM : svm.SVC(kernel='linear') | wavelets | 1.0 | 0.875 | 0.85 | 128 |
| RandomForestClassifier(n\_estimators=100, random\_state=42) | wavelets | 1.0 | 0.875 | 0.9 | 128 |
| SVM : svm.SVC(kernel='linear') | Auto regression | **0.75** | **0.6875** | **0.55** | **128** |
| RandomForestClassifier(n\_estimators=100, random\_state=42) | Auto regression | **1.0** | **0.8125** | 0.55 | 128 |
| Gaussian Naive Bayes | Auto regression | 0.890625 | 0.875 | 0.8 | 50 |
| **RandomForestClassifier(n\_estimators=100, random\_state=42)** | **Auto regression** | **1.0** | **0.875** | **0.9** | **50** |

The best classifier is RandomForestClassifier

1. **Screen shots for your running interface :**

**Confusion Matrix:**



**User Interface:**

The game starts with an object in the upper-left of the Game World.

The goal of the game is to pass the object through the end line.

The only way to move the object is by predicting the EOG signals that is coo-responded to the object.

So the object starts moving by reading signals from ‘Move’ Button then the object moves according to prediction of the move respect to the selected signals (h,v).

