Faculty of Engineering
Cairo University

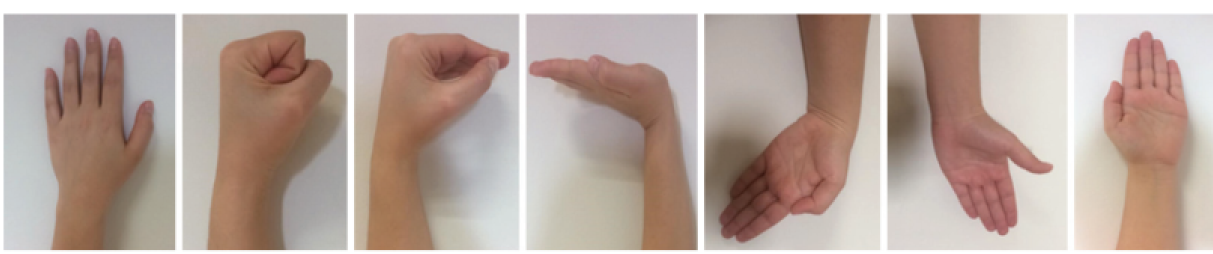
Machine Learning-Based Hand Gesture Recognition via EMG Data

Team 6 (iAI)

Clinical decision support systems final project

Dr.Eman Ayman

22/05/2023



Contents

Literature review

1.1 Introduction

Gesture recognition has gained significant attention in recent years due to its potential applications in various fields, such as human-computer interaction, rehabilitation engineering, and virtual reality. It offers a natural and intuitive way of controlling devices and interfaces through hand movements. One promising approach for gesture recognition is the utilization of electromyography (EMG) signals.

EMG signals are generated by the electrical activity of muscles during voluntary contractions. They reflect the activation patterns of muscles, making them a suitable modality for capturing hand movements and gestures. By analyzing these signals, it is possible to infer the intended hand gestures and translate them into meaningful commands or actions.

EMG-based gesture recognition systems have several advantages. They are non-invasive and can be captured using surface electrodes placed on the skin surface above targeted muscles. This makes them accessible and easy to use. Furthermore, EMG signals can provide real-time feedback, enabling interaction with devices and interfaces.

1.2 About the paper

The paper we are discussing titled "Machine Learning-Based Hand Gesture Recognition via EMG Data" by Zehra Karapinar Senturk and Melahat Sevgul Bakay focuses on a machine learning-based approach for hand gesture recognition using EMG data

The authors explore the use of machine learning algorithms to recognize and classify specific hand gestures based on the patterns observed in the EMG signals. The machine learning algorithms learn from a labeled dataset, which contains examples of different hand gestures along with their corresponding EMG signals.

The key steps involved in EMG-based gesture recognition, including signal acquisition, preprocessing, feature extraction, and classification. The importance of proper electrode placement, noise reduction techniques, and the selection of appropriate features are also discussed in this paper to enhance the accuracy of the system.

The models used in this paper : support vector machines (SVM), artificial neural networks (ANN), and random forests (RF). These algorithms are trained using the extracted features to classify and recognize hand gestures.

Problem Definition

The focus is on recognizing and classifying hand gestures based on the patterns observed in the EMG signals. The aim is to develop a system that can distinguish between different gestures and accurately identify the intended gesture in real-time.

The EMG data for the gesture recognition application was sourced from the UCI Machine Learning Repository. The dataset was collected using the Myo Thalmic armband, which was placed on the forearm. The data was then transferred from the armband to a PC using Bluetooth connectivity. The Myo armband, manufactured by Thalmic Labs, consists of 8 equidistant sensors. These 8 EMG channels from the Myo Thalmic bracelet were utilized as input features for the classifiers.

Myo armband and its placement on the forearm (Wahid et al., 2018)

Main medical applications using gesture recognition associated with VR includes:

1- Virtual Rehabilitation: Gesture recognition combined with VR technology is used in virtual rehabilitation to provide interactive and engaging therapy experiences. Patients can perform therapeutic movements and exercises in virtual environments, while their hand gestures are tracked and interpreted in real-time.

2-Surgeon Assistance: Gesture recognition integrated with VR enables touchless interaction with medical imaging data and control of surgical systems. Surgeons can navigate and manipulate complex imaging datasets using hand gestures, ensuring a sterile environment and enhancing surgical precision.

Major companies in this field includes:

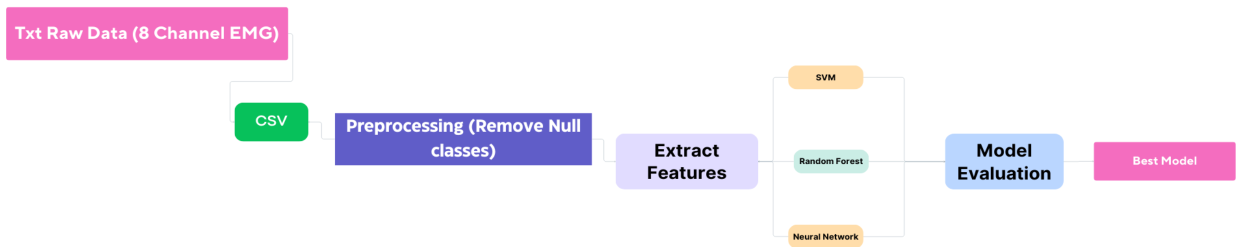
1-Theator.io

2-Augmedics

3-Kinestica

Methodology

3.1 Block Diagram



3.2 Pre-processing

Raw EMG data was recorded by MYO Thalamic bracelet for 36 subjects saved in text format, each subject performed two series (each series represents a text file), each of which consists of six (seven) basic gestures. Each gesture was performed for 3 seconds with a pause of 3 seconds between gestures.

Each text file was converted to csv format and then merging all these files into one csv file to perform our processing and feature extraction, then we dropped the records with unknown class & class number 7 as we already have a large dataset of 18464383 records, so our accuracy won't be affected. The final csv file had 10 columns, represented by the time taken for each subject in microseconds, the 8 EMG signal channels and finally the class of that gesture performed by the subject. Where class (0) represents unmarked data, class (1) represents hand at rest, class (2) represents hand clenched in a fist, class (3) represents wrist flexion, class (4) represents wrist extension, class (5) represents radial deviations, class (6) represents ulnar deviations, class (7) represents extended palm (the gesture was not performed by all subjects).

```python
import pandas as pd
import numpy as np
from scipy.fft import fft
```

[2] Python

```python
df = pd.read_csv('C:/Users/saraa/OneDrive/Desktop/CDSS - Final Project/data/csv files/merged_files.csv')
```

[3] Python

```python
df
```

[3] Python

	time	channel1	channel2	channel3	channel4	channel5	channel6	channel7	channel8	class	label
0	710	-0.00002	-0.00002	-0.00001	0.00000	-0.00002	-0.00001	0.00002	-0.00001	1.0	1.0
1	711	-0.00002	-0.00002	-0.00001	0.00000	-0.00002	-0.00001	0.00002	-0.00001	1.0	1.0
2	712	-0.00002	-0.00002	-0.00001	0.00000	-0.00002	-0.00001	0.00002	-0.00001	1.0	1.0
3	713	-0.00002	-0.00002	-0.00001	0.00000	-0.00002	-0.00001	0.00002	-0.00001	1.0	1.0
4	714	-0.00002	-0.00002	-0.00001	0.00000	-0.00002	-0.00001	0.00002	-0.00001	1.0	1.0
...	...	...	...	...	...	...	...	...	...	...	...
18464378	59208	-0.00002	-0.00005	-0.00013	-0.00021	-0.00014	-0.00004	-0.00001	-0.00002	0.0	9.0
18464379	59209	-0.00002	-0.00005	-0.00013	-0.00021	-0.00014	-0.00004	-0.00001	-0.00002	0.0	9.0
18464380	59211	-0.00002	-0.00005	-0.00013	-0.00021	-0.00014	-0.00004	-0.00001	-0.00002	0.0	9.0
18464381	59212	-0.00002	-0.00005	-0.00013	-0.00021	-0.00014	-0.00004	-0.00001	-0.00002	0.0	9.0
18464382	59213	-0.00002	-0.00005	-0.00013	-0.00021	-0.00014	-0.00004	-0.00001	-0.00002	0.0	9.0

18464383 rows × 11 columns

Original Dataset

```python
index_numbers_1 = df[df["class"] == 0].index
df.drop(index_numbers_1, inplace=True)
```

[5] Python

```python
df['class'].value_counts()
```

[6] Python

```
6.0    1265045
5.0    1258665
4.0    1257850
1.0    1250275
3.0    1247470
2.0    1215965
7.0      68480
Name: class, dtype: int64
```

```python
index_numbers_2 = df[df["class"] == 7].index
df.drop(index_numbers_2, inplace=True)
```

[7] Python

```python
df['class'].value_counts()
```

[8] Python

```
6.0    1265045
5.0    1258665
4.0    1257850
1.0    1250275
3.0    1247470
2.0    1215965
Name: class, dtype: int64
```

After dropping the classes with values of 0 & 7

3.2.1 Filtration

Firstly, we performed signal filtration, where we passed our signal through a notch filter & then through bandpass filter after normalizing the data, however we found that there was a significant decrease in the accuracy of our models after the filtration step, so we proceeded without it.

```
from sklearn.svm import SVC

# Train the SVM classifier
svm = SVC()
svm.fit(X_train, Y_train)

# Evaluate the SVM classifier
svm_accuracy = svm.score(X_test, Y_test)
print("Accuracy:", svm_accuracy)

Accuracy: 0.9545454545454546

from sklearn.ensemble import RandomForestClassifier

# Train the Random Forest classifier
rf = RandomForestClassifier()
rf.fit(X_train, Y_train)

# Evaluate the Random Forest classifier
rf_accuracy = rf.score(X_test, Y_test)
print("Accuracy:", rf_accuracy)

Accuracy: 0.9318181818181818

from sklearn.neural_network import MLPClassifier
mlp = MLPClassifier(hidden_layer_sizes=(15,20,25), activation='relu',
solver='adam', max_iter=500)
mlp.fit(X_train,Y_train)
predict_train = mlp.predict(X_train)
predict_test = mlp.predict(X_test)

# Evaluate the Random Forest classifier
mlp_accuracy = mlp.score(X_test, Y_test)
print("Accuracy:", mlp_accuracy)

Accuracy: 0.9772727272727273
```

Before Filtration

```
from sklearn.svm import SVC

# Train the SVM classifier
svm = SVC()
svm.fit(X_train, Y_train)

# Evaluate the SVM classifier
svm_accuracy = svm.score(X_test, Y_test)
print("Accuracy:", svm_accuracy)

Accuracy: 0.7727272727272727

from sklearn.ensemble import RandomForestClassifier

# Train the Random Forest classifier
rf = RandomForestClassifier()
rf.fit(X_train, Y_train)

# Evaluate the Random Forest classifier
rf_accuracy = rf.score(X_test, Y_test)
print("Accuracy:", rf_accuracy)

Accuracy: 0.9090909090909091

from sklearn.neural_network import MLPClassifier
mlp = MLPClassifier(hidden_layer_sizes=(10,15,20), activation='relu',
solver='adam', max_iter=500)
mlp.fit(X_train,Y_train)
predict_train = mlp.predict(X_train)
predict_test = mlp.predict(X_test)

# Evaluate the Random Forest classifier
mlp_accuracy = mlp.score(X_test, Y_test)
print("Accuracy:", mlp_accuracy)

Accuracy: 0.022727272727272728
```

After Filtration

3.3 Feature Extraction

Secondly, for each gesture performed by each subject extract the 12 features for the corresponding 8 channels, the features extracted for each channel were the root mean square (RMS), variance, mean absolute value (MAV), simple square integrated (SSI), waveform length (WL), integrated EMG (IEMG), difference absolute mean value (DAMV), difference absolute standard deviation value (DASDV), Willison amplitude (WAMP), peak to peak (PTP), min & max.

```python
def IEMG(data):
    return np.sum(np.abs(data), axis=0)

def SSI(data):
    return np.sum(data**2, axis=0)

def rms(data):
      return  np.sqrt(np.mean(data**2, axis=0))

def MAV(data):
    return np.mean(np.abs(data), axis=0)

def variance(data):
    return np.var(data, axis=0)

def WL(data):
    return np.sum(np.abs(np.diff(data,axis=0)),axis = 0)

def DAMV(data):
    return np.mean(np.abs(np.diff(data, axis=0)), axis=0)

def DASDV(data):
    return np.std(np.abs(np.diff(data, axis=0)), axis=0)

def WAMP(data):
    differences = np.abs(np.diff(data, axis=0))
    threshold = np.mean(np.mean(differences, axis=0))
    return np.sum(differences > threshold, axis=0)
```
[12] Python

Features

```python
df = df.reset_index()
df
```
[14] Python

	label	class	channel1								...	channel8									
			min	max	IEMG	SSI	rms	MAV	variance	WL	...	IEMG	SSI	rms	MAV	variance	WL	ptp	DAMV	DASDV	WAMP
0	-1.0	1.0	-0.00008	0.00008	0.08866	0.000002	0.000019	0.000015	2.915012e-10	0.01321	...	0.09464	0.000002	0.000019	0.000015	2.844973e-10	0.01298	0.00011	0.000002	0.000008	596
1	-1.0	2.0	-0.00128	0.00127	2.76972	0.002112	0.000581	0.000442	3.372732e-07	0.45666	...	1.40258	0.000564	0.000300	0.000224	9.010515e-08	0.22614	0.00205	0.000036	0.000138	665
2	-1.0	3.0	-0.00128	0.00127	1.57019	0.000817	0.000358	0.000247	1.280456e-07	0.26005	...	1.22648	0.000423	0.000258	0.000193	6.647597e-08	0.20233	0.00216	0.000032	0.000124	628
3	-1.0	4.0	-0.00037	0.00027	0.38221	0.000040	0.000078	0.000059	5.977212e-09	0.06433	...	0.54181	0.000078	0.000109	0.000083	1.181490e-08	0.08512	0.00091	0.000013	0.000049	663
4	-1.0	5.0	-0.00103	0.00112	0.54509	0.000142	0.000147	0.000083	2.152237e-08	0.08445	...	0.27746	0.000023	0.000060	0.000042	3.464970e-09	0.04145	0.00062	0.000006	0.000026	654
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
325	71.0	2.0	-0.00128	0.00127	1.62043	0.000708	0.000318	0.000231	1.006916e-07	0.27527	...	3.30172	0.002372	0.000581	0.000470	3.375245e-07	0.56014	0.00255	0.000080	0.000286	737
326	71.0	3.0	-0.00112	0.00075	1.42510	0.000464	0.000247	0.000187	6.070625e-08	0.24359	...	1.36494	0.000542	0.000266	0.000179	7.091400e-08	0.23177	0.00234	0.000030	0.000129	746
327	71.0	4.0	-0.00054	0.00017	0.27272	0.000022	0.000053	0.000035	2.729972e-09	0.04484	...	0.28793	0.000021	0.000052	0.000037	2.619938e-09	0.05295	0.00053	0.000007	0.000026	843
328	71.0	5.0	-0.00096	0.00110	1.77422	0.000699	0.000303	0.000233	9.191612e-08	0.31631	...	1.11028	0.000355	0.000216	0.000146	4.660867e-08	0.19548	0.00192	0.000026	0.000108	764
329	71.0	6.0	-0.00128	0.00110	1.16537	0.000582	0.000270	0.000146	7.268614e-08	0.20381	...	0.87744	0.000216	0.000164	0.000110	2.682055e-08	0.15987	0.00176	0.000020	0.000082	875

330 rows × 98 columns

After Feature Extraction

3.4 Feature selection

Thirdly, after extracting your features select the best 80 features.

```python
# Feature Selection
from sklearn.feature_selection import SelectKBest
from sklearn.feature_selection import f_classif

# feature extraction
test = SelectKBest(score_func=f_classif, k=80)

Y = Y.ravel()
fit = test.fit(X, Y)

# summarize scores
np.set_printoptions(precision=1)
print(fit.scores_)
```

Select K best features

```python
# Visualization
import matplotlib.pyplot as plt

plt.figure(figsize=(10, 6))
plt.bar(range(len(feature_scores)), feature_scores)
plt.xlabel("Feature Index")
plt.ylabel("Feature Score")
plt.title("Feature Scores")
plt.show()
```

Features' Scores

3.5 Classification

- Fourthly, for obtaining the best output from our data we used the best 3 classifiers for EMG signal to compare between them which are the SVM, Random Forest Classifier & Neural Network. For Neural Network we chose 3 hidden layers, the rectified linear unit function as the Activation function for the hidden layer, stochastic gradient-based optimizer proposed by Kingma, Diederik, and Jimmy Ba as the solver for weight optimization as it works well on relatively large datasets (with thousands of training samples or more) in terms of both training time and validation score, 500 Maximum number of iterations, where the solver iterates until convergence this number of iterations.

```python
from sklearn.model_selection import train_test_split

# Split the data into training and testing sets
X_train, X_test, Y_train, Y_test = train_test_split(selected_features, Y, test_size=0.2, random_state=42)
```

[511] Python

Split the data to 80% train & 20% test

```python
from sklearn.svm import SVC

# Train the SVM classifier
svm = SVC()
svm.fit(X_train, Y_train)

# Evaluate the SVM classifier
svm_accuracy = svm.score(X_test, Y_test)
print("Accuracy:", svm_accuracy)
```

[512] Python

```
Accuracy: 0.9545454545454546
```

SVM Classifier

```python
from sklearn.ensemble import RandomForestClassifier

# Train the Random Forest classifier
rf = RandomForestClassifier()
rf.fit(X_train, Y_train)

# Evaluate the Random Forest classifier
rf_accuracy = rf.score(X_test, Y_test)
print("Accuracy:", rf_accuracy)
```

[513] Python

```
... Accuracy: 0.9090909090909091
```

Random Forest Classifier

```python
from sklearn.neural_network import MLPClassifier
mlp = MLPClassifier(hidden_layer_sizes=(10,15,20), activation='relu',
solver='adam', max_iter=500)
mlp.fit(X_train,Y_train)
predict_train = mlp.predict(X_train)
predict_test = mlp.predict(X_test)

# Evaluate the Random Forest classifier
mlp_accuracy = mlp.score(X_test, Y_test)
print("Accuracy:", mlp_accuracy)
```

[531] Python

```
... Accuracy: 0.9772727272727273
```

Neural Network

Model Evaluation

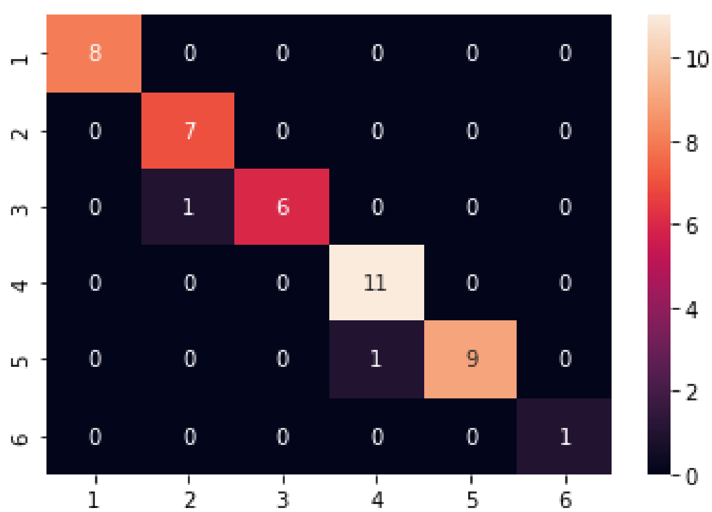
3.6 Performance Comparison

Comparing the performance of each model using multiple evaluation metrics:

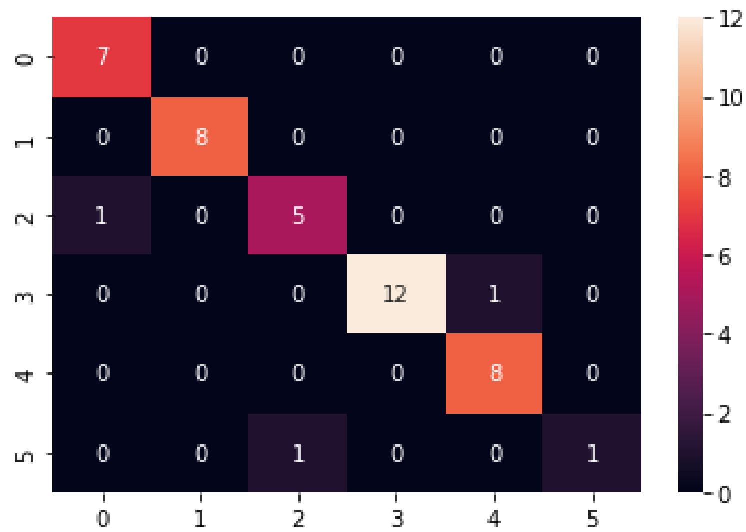
Metric	Classifier	SVM	RF	MLP
Accuracy (%)		95.45	93.18	95.45
Precision (%)		96.0	94.49	95.45
Recall (%)		95.45	93.18	95.45
F1 score (%)		95.47	93.46	95.45
Root Mean Squared Error		0.213	0.564	0.213

Confusion Matrix

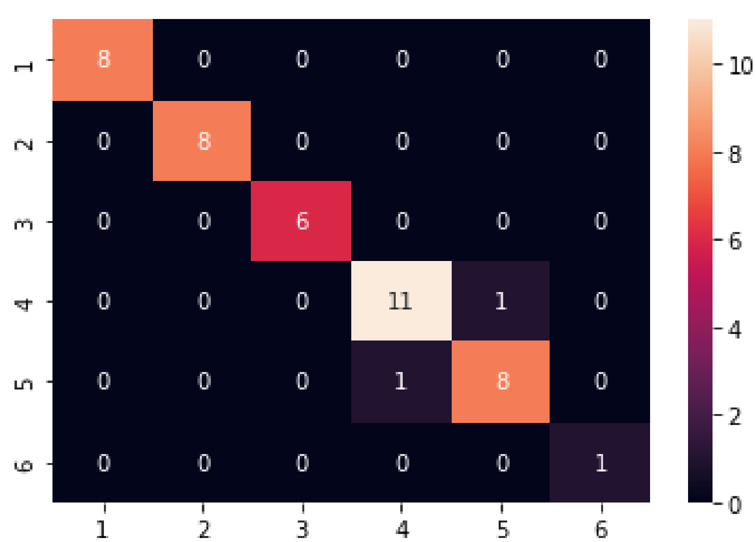
- SVM



- RF


0
1
2
3
4
5
7 0 0 0 0 0
0 8 0 0 0 0
1 0 5 0 0 0
0 0 0 12 1 0
0 0 0 0 8 0
0 0 1 0 0 1
0 1 2 3 4 5
12
10
8
6
4
2
0

- MLP


1
2
3
4
5
6
8 0 0 0 0 0
0 8 0 0 0 0
0 0 6 0 0 0
0 0 0 11 1 0
0 0 0 1 8 0
0 0 0 0 0 1
1 2 3 4 5 6
10
8
6
4
2
0

Results And Discussion

4.1 Comparison

In our methodology we used three classifiers (SVM, RF and NN) to predict the class of gesture and we compare the three classifiers, the findings are in the table below:

comparison / classifier	SVM	RF	NN
Model Complexity	Is a relatively simple model that constructs a hyperplane to separate classes.	Is an ensemble of decision trees.	Is a complex model with multiple layers of interconnected nodes.
Feature Engineering	Is well-suited for handling handcrafted features extracted from EMG signals.	Is well-suited for handling handcrafted features extracted from EMG signals.	Can learn features automatically through its layers, but is better to use extracted features with it for accuracy improvement.
Training Data Requirements	Can perform well even with relatively small training datasets and can handle high-dimensional feature spaces efficiently.	Can perform well even with relatively small training datasets and can handle high-dimensional feature spaces efficiently.	Require large amounts of labeled data to generalize effectively and avoid overfitting.
Interpretability	Is more interpretable compared to NN. Determines the decision boundary based on the support vectors.	Is more interpretable compared to NN. Provides feature importance measures.	Internal workings are often considered as "black boxes," making it difficult to interpret how decisions are made based on the input features.
Robustness to Noise	Can handle noise to some extent by using appropriate regularization and kernel functions.	Is generally robust to noise and outliers due to its ensemble nature and the voting mechanism.	Can be sensitive to noisy data, and additional preprocessing steps or regularization techniques may be required.
Computational Efficiency	Is computationally efficient, especially for smaller datasets.	Training and inference times can be higher, depending on the number of trees in the forest.	Particularly deep neural networks, can be computationally expensive to train and require powerful hardware or distributed computing

			resources.
Scalability	Is relatively scalable and can handle larger datasets with limited computational resources.	Is relatively scalable and can handle larger datasets with limited computational resources.	Depends on the architecture, but training and inference times can increase significantly with larger datasets.

.

4.2 Input Data

The input data will be an array of type double of size 96 (12x8), where each element will represent the feature extracted for each of the EMG signal 8 channels.

Team members

Name	ID
Aya Amr	9202342
Sara Ayman	9202615
Shuaib Abdulsalam	9204062
Abdelrahman Saeed	9202839
Mahmoud Rabea	9203396

References

Acharya, U. R., Dua, S., Du, X., Sree S, V., and Chua, C. K., (2011). Automated diagnosis of glaucoma

using texture and higher order spectra features. IEEE Transactions on Information Technology in

Biomedicine, 15(3):449–455. ISSN 10897771. doi:10.1109/TITB.2011.2119322.

Bian, F., Li, R., and Liang, P., (2017). SVM based simultaneous hand movements classification using

sEMG signals. In 2017 IEEE International Conference on Mechatronics and Automation, ICMA

2017, pages 427–432. Institute of Electrical and Electronics Engineers Inc. ISBN 9781509067572.

doi:10.1109/ICMA. 2017.8015855.

Donovan, I., Valenzuela, K., Ortiz, A., Dusheyko, S., Jiang, H., Okada, K., and Zhang, X., (2017).

MyoHMI: A low-cost and flexible platform for developing real-time human machine interface for

myoelectric controlled applications. In 2016 IEEE International Conference on Systems, Man, and

Cybernetics, SMC 2016 - Conference Proceedings, pages 4495–4500. Institute of Electrical and

Electronics Engineers Inc. ISBN 9781509018970. doi:10.1109/SMC.2016.7844940.

Englehart, K., Hudgins, B., and Parker, P. A., (2001). A wavelet-based continuous classification scheme

for multifunction myoelectric control. IEEE Transactions on Biomedical Engineering, 48(3):302–

311. ISSN 00189294. doi:10.1109/10.914793.

Eshitha, K. V. and Jose, S., (2018). Hand Gesture Recognition Using Artificial Neural Network. In

2018 International Conference on Circuits and Systems in Digital Enterprise Technology, ICCSDET

2018. Institute of Electrical and Electronics Engineers Inc. ISBN 9781538605769. doi:10.1109/

ICCSDET.2018.8821076.

Faust, O. and Bairy, M. G., (2012). Nonlinear analysis of physiological signals: A review. Journal of

Mechanics in Medicine and Biology, 12(4). ISSN 02195194. doi:10.1142/S0219519412400155.

Faust, O., Hagiwara, Y., Hong, T. J., Lih, O. S., and Acharya, U. R., (2018). Deep learning for healthcare

applications based on physiological signals: A review. Computer Methods and Programs in

Biomedicine, 161:1–13. ISSN 18727565. doi:10.1016/j.cmpb.2018.04.005.

Guo, W., Sheng, X., Liu, H., and Zhu, X., (2017). Toward an Enhanced Human-Machine Interface for

Upper-Limb Prosthesis Control with Combined EMG and NIRS Signals. IEEE Transactions on

Human-Machine Systems, 47(4):564–575. ISSN 21682291. doi:10.1109/THMS.2016.2641389.

Jayalakshmi, T. and Santhakumaran, A., (2011). Statistical Normalization and Back Propagationfor

Classification. International Journal of Computer Theory and Engineering, 3(1):89–93. ISSN

17938201. doi:10.7763/ ijcte.2011.v3.288.

Jiang, S., Gao, Q., Liu, H., and Shull, P. B., (2020). A novel, co-located EMG-FMG-sensing wearable

armband for hand gesture recognition. Sensors and Actuators, A: Physical, 301. ISSN 09244247.

doi:10.1016/j.sna. 2019.111738.

Kim, J. H., Hong, G. S., Kim, B. G., and Dogra, D. P., (2018). deepGesture: Deep learning-based gesture

recognition scheme using motion sensors. Displays, 55:38–45. ISSN 01419382. doi:10.1016/j.

displa.2018. 08.001.

Liang, H., Yuan, J., Thalmann, D., and Nadia, M. T., (2015). AR in hand: Egocentric palm pose tracking

and gesture recognition for augmented reality applications. In MM 2015 - Proceedings of the 2015

ACM Multimedia Conference, pages 743–744. Association for Computing Machinery, Inc. ISBN

9781450334594. doi:10.1145/2733373.2807972.

Mi, J., Sun, Y., Wang, Y., Deng, Z., Li, L., Zhang, J., and Xie, G., (2016). Gesture recognition based

teleoperation framework of robotic fish. In 2016 IEEE International Conference on Robotics and