

**Final Year Project to Obtain the Diploma of
Engineering**

Field

Automation

Speciality

Automation and Industrial Computing

- Subject -

**Contribution to Gestures Recognition Using Multimodal
Signals through Deep Learning**

Contribution à la reconnaissance des gestes en utilisant des
signaux Multimodales par Deep Learning

By

ZITOUNI Nour Elislam

SERAICHE Oubada

Examination Committee :

M. Cherabit Nourdine	Chair	M.C.B
Ms. REBAI Karima	Supervisor	M.C.A
M. Choutri Kheireddine	Examiner	M.C.B
Ms. Chouaf Seloua	Examiner	M.A.A

Algiers, the 04/04/2023

Academic year 2022 –2023

Dedication

“

In the name of Allah, the Most Merciful and the Most Gracious,

First of all, I would like to extend my heartfelt appreciation to my beloved parents, family, and friends, whose unwavering support and encouragement have been a constant source of strength. Your love, prayers, and belief in me have propelled me forward and given me the confidence to pursue my dreams,

To all the mentors, teachers, and colleagues who have played a pivotal role in shaping my professional growth, I am truly grateful for your wisdom, guidance, and expertise. Your invaluable contributions have helped me develop new skills, expand my knowledge, and overcome challenges along the way.

Lastly, I acknowledge the countless blessings and opportunities that have come my way, and I humbly pray that my work and efforts serve as a means to make a positive impact on others and contribute to the betterment of society,

Thanks.

”

- Islem

Dedication

“

In the name of Allah, the Most Merciful and the Most Gracious,

First and foremost, I express my heartfelt gratitude to the Almighty for granting me the inspiration, guidance, and the right path to achieve this position,

I dedicate this humble work, with sincere devotion, to the loving memory of my late father, seeking Allah's mercy upon him, Allah yarhamo.

I also extend my heartfelt dedication to my dear mother and brothers, wishing them all the success in their lives. To every member of my family, may this work serve as a testament to my profound respect and admiration for each one of you.

Also, I offer this dedication to all those who have placed their trust in me and offered their well-wishes,

Thanks.

”

- Oubada

Acknowledgments

First and foremost, we offer our thanks and praises to the Almighty God for granting us the patience, determination, and success to complete this humble endeavor.

In these concise words, we extend our heartfelt gratitude to all those who have provided us with assistance and contributed to the development of this work.

We would like to express our deep appreciation to Ms. **REBAI Karima** for graciously accepting the responsibility of leading this project with attentiveness and care. We are truly grateful for her availability, meticulous guidance, and support throughout the various challenges we encountered.

Furthermore, we extend our special thanks to the esteemed members of the Jury for their genuine interest in our project and their willingness to examine our work and enhance it with their valuable suggestions and proposals.

Contents

Dedication	II
Dedication	III
Acknowledgments	IV
General Introduction	1
1 Multimodal perception of hand gestures	3
1.1 Introduction	4
1.2 Electromyogram (EMG) signals	4
1.2.1 The characteristics of EMG signals	5
1.2.2 Hand Gesture Recognition using EMG	6
1.3 Accelerometer data	6
1.3.1 The characteristics of Accelerometer Data	7
1.3.2 Hand Gesture Recognition Using Accelerometer Data	8
1.4 Eye tracking data	8
1.4.1 The characteristics of Eye tracking in gesture recognition	9
1.4.2 Hand Gesture Recognition Using Eye tracking	10
1.5 Description of the dataset MeganePro[12]	10
1.5.1 Dataset Acquisition	11
1.5.2 Dataset size and organisation	14
1.5.3 Data pre-processing	15
1.5.4 Data Segmentation	16
1.5.5 Data preparation	17
1.6 Conclusion	17
2 Deep Learning	18
2.1 Introduction	19
2.2 Neural networks	19
2.2.1 Neuron model	19
2.2.2 Forward propagation	22
2.3 Loss function	23
2.4 Backpropagation	23
2.5 Optimization Algorithms	23
2.5.1 Stochastic Gradient Descent	23
2.5.2 Adam (Adaptive Moment Estimation)	24
2.6 Regularisation	24

2.6.1	Dropout	24
2.6.2	Early stopping	24
2.7	Deep Learning Foundations	25
2.8	Convolutional neural networks (CNNs)	26
2.8.1	The basic components of a CNN	26
	Convolutional Layer	26
	Pooling layers	26
	Fully connected layers	27
2.8.2	Understanding Conv1D and Conv2D	27
2.8.3	Advantages and limitations of CNNs	28
2.9	Recurrent neural network (RNN)	29
2.9.1	Types of Recurrent Neural Networks	29
	2.9.1.1 One-to-One RNN	30
	2.9.1.2 One-to-Many	30
	2.9.1.3 Many-to-One	30
	2.9.1.4 Many-to-Many	30
2.9.2	Advantages and limitations of Recurrent Neural Networks (RNNs)	31
2.9.3	Long Short-Term Memory (LSTM)	31
2.10	Multimodal Deep Learning	32
2.11	Evaluation metrics for performance	33
2.12	Conclusion	35
3	Hand gestures recognition using Single Modality (sEMG)	36
3.1	Introduction	37
3.2	CNN-LSTM model for sEMG-based hand gesture recognition	37
	3.2.1 Convolutional Layers	38
	3.2.2 LSTM Layers	38
	3.2.3 Regularization methods	39
	3.2.4 Fully Connected Layers:	39
	3.2.5 Hyperparameter Tuning	39
	3.2.6 Configuration 1	40
	3.2.7 Configuration 2	40
	3.2.8 Configuration 3	41
3.3	Performances evaluation of the final single modality CNN-LSTM model	42
	3.3.1 Performances evaluation using training data	42
	3.3.2 Performances evaluation using test data	43
3.4	Effect of manipulated object in recognition performance	46
3.5	Conclusion	49
4	Hand gestures recognition using Multimodality	51
4.1	Introduction	52
4.2	Fusion of sEMG with Gaze	52
	4.2.1 Model training and performances evaluation	53
4.3	Fusion of sEMG with Accelerometer measurements	55
4.4	Fusion of sEMG with Gaze and Accelerometer modalities	57
4.5	Test on Variability	61

Contents

4.5.1	Object-split	62
4.5.2	Trial-split	62
4.6	Models Training and performance evaluation considering the Amputees data	63
4.7	Model's Performances comparison with Related works	64
4.8	conclusion	66
Appendices		69
A Software and Hardware Tools		70
B Model's configuration parameters		72

List of Figures

1.1	illustrates an example of sEMG signal.	5
1.2	Robotic Hand Prostheses Controlled by Electromyography Data	6
1.3	The flat movement measurement of the accelerometer (x, y and z axis)[6]	7
1.4	Sensor of a three-axis accelerometer enclosed in a waterproof case	8
1.5	Overview of the gaze quantities calculated by a head mounted eye tracking system[10]	9
1.6	Visualizing Gaze Points Overlapped onto Scene Camera Video: An Illustrative Example[12]	10
1.7	Eye tracking data device and sEMG electrodes for an acquisition[12]	11
1.8	Tobii Pro Glasses 2 and sEMG electrodes [10]	12
1.9	Overview of the acquisition protocol and setup during the execution of a task[10]	12
1.10	sEMG signals	14
1.11	Eye tracking signals	14
1.12	Accelerometer signals	15
1.13	Data segmentation	16
1.14	Sliding window technique	17
2.1	The various components of a neuron	20
2.2	Multi-layer neural network[21]	22
2.3	Dropout Strategy. (a) A standard neural network. (b) Applying dropout to the neural network on the left by dropping the crossed units[23]	25
2.4	Early Stopping Strategy[24]	25
2.5	Example of a max pooling operation with spatial extent $F = 2 \times 2$ and stride $S=2$	27
2.6	Illustration of a typical CNN architecture with two convolution layers, two max pooling layers, one fully connected layer, and one soft max classification layer	27
2.7	Data Structure Requirements for 1D and 2D Convolutional Neural Networks[28]	28
2.8	Architecture of a traditional RNN	29
2.9	Summary of the main RNN models[30]	30
2.10	LSTM gates[30]	31
2.11	Example of Multimodal deep learning where different types of NN are used to extract features	33
2.12	Structure of a 2x2 Confusion Matrix	33
3.1	A Hybrid Model Approach for Hand Gesture Recognition using sEMG Signals	38

3.2	Accuracy and Loss Curves of the obtained results for the training and validation data(Configuration 1)	40
3.3	Accuracy and Loss Curves of the obtained results for the training and validation data (Configuration 2)	41
3.4	Accuracy and Loss Curves of the obtained results for the training and validation data (Configuration 3)	41
3.5	Evaluation of the Hybrid Model Performance: Confusion Matrix for training data	42
3.6	Predicted and True Labels in the training data model	43
3.7	Comparison of Actual and Predicted Label Frequencies for Ten Classes in the Training Data Model	43
3.8	Evaluation of the Hybrid Model Performance: confusion matrix of test data	44
3.9	Predicted and True Lables Comparison	45
3.10	Comparison of Actual and Predicted Label Frequencies for Ten Classes in the Test Data Model	45
4.1	Integrating Gaze and sEMG Signals in Hand Gesture Recognition: A Hybrid Model Approach	53
4.2	Accuracy and Loss Curves for the Hybrid Model of sEMG and Gaze	54
4.3	Evaluation of the Hybrid Model Performance: Confusion Matrix for sEMG and Gaze	55
4.4	Integrating Accelerometer, and sEMG Signals in Hand Gesture Recognition: A Hybrid Model Approach	56
4.5	Accuracy and Loss Curves for the Hybrid Model of sEMG & Accelerometer	57
4.6	Evaluation of the Hybrid Model Performance: Confusion Matrix for sEMG and Accelerometer	57
4.7	A Hybrid Model Approach for Integrating Accelerometer, Gaze, and sEMG Data in Hand Gesture Recognition	58
4.8	Accuracy and Loss Curves for the Hybrid Model of sEMG & Acceleromete & Gaze	59
4.9	Evaluation of the Hybrid Model Performance: Confusion Matrix for sEMG & Acceleromete and Gaze	60
4.10	Predicted and True Labels Comparison	60
4.11	Comparison of Actual and Predicted Label Frequencies for Ten Classes in the Model	61
4.12	Accuracy Comparison of Modalities	62
4.13	Accuracy and Loss Curves for sEMG-only Model	64
4.14	Accuracy and Loss Curves for sEMG & Gaze	64
4.15	Accuracy and Loss Curves for sEMG & Accelerometer	64
4.16	Classification accuracies for able-bodied and amputated subjects when predicting the grasp type with three different types of classifiers using sEMG signals [10] and our model	65
4.17	Classification accuracies for intact (intact ¹) and amputated (Amputees ¹) subjects when predicting the grasp type using only sEMG and while integrating also the visual information[35]	65

List of Tables

1.1	Overview of the grasp types and objects for the static condition of the exercise	13
2.1	Popular activation functions	21
3.1	Different hyperparameters	39
3.2	The individual performances of each class	44
3.3	The global performances of the developed model	45
3.4	The individual performances of each class	46
3.5	Subject's accurecies while manipulating different objects	47
3.6	Overview of the objects and the grasps of the dataset	48
4.1	Fusion of sEMG with Gaze: Model parameters	53
4.2	Performances comparison of the sEMG and the fusion of sEMG and Gaz models	54
4.3	Performances comparison of the sEMG, sEMG & Gaze and sEMG & Accelerometers models	56
4.4	Models performances	59
4.5	Test results for each class	61
4.6	Accuracy Results	63
B.1	Parameter Configuration 1 for Training a CNN-LSTM Model	72
B.2	Parameter Configuration 2 for Training a CNN-LSTM Model	73
B.3	Parameter Configuration 3 for Training a CNN-LSTM Model	73

List of abbreviations and acronyms

EMG	<i>ElectroMyoGraphy</i>
ECG	<i>Electrocardiogram</i>
EEG	<i>Electroencephalogram</i>
CNN	<i>Convolutional Neural Network</i>
FN	<i>False Negative</i>
FP	<i>False Positive</i>
TP	<i>True Positive</i>
TN	<i>True Negative</i>
MSE	<i>Mean-Squared Error</i>
RNN	<i>Recurrent Neural Network</i>
sEMG	<i>Surface Electromyography</i>
SGD	<i>Stochastic gradient descent</i>
Adam	<i>Adaptive Moment Estimation</i>
SNR	<i>signal to noise ratio</i>
LSTM	<i>Long Short-Term Memory</i>
NN	<i>Neural Network</i>
MLP	<i>Multi-Layer Perceptron</i>
Conv1D	<i>1-dimensional convolutional neural network</i>

List of Tables

Conv2D	<i>2-dimensional convolutional neural network</i>
ReLU	<i>Rectified Linear Unit</i>
HCLSTM	<i>Hybrid CNN LSTM Model</i>
MeganePro	<i>Myo-Electricity, Gaze And Artificial-intelligence for Neurocognitive Examination & Prosthetics</i>

General Introduction

A person's life can be significantly affected both physically and psychologically by losing a hand. The physical effects, particularly when the amputation is not fully healed or when pain from phantom limbs manifests, can make it difficult to do daily duties and cause discomfort and anguish. However, advancements in prosthetics have provided amputees with opportunities to regain physical fitness and independence. Myoelectric prosthetics, in particular, allow individuals to restore some lost limb function by utilizing surface Electromyography (sEMG) signals from antagonistic muscles to control basic gripping movements. While progress has been made in improving the function of complex prosthetics through pattern recognition methods, their full potential for practical clinical applications is yet to be realized.

The employment of gestures with multimodal signals is a promising advancement in the field of prosthetics that enables amputees to control their prosthetic devices more organically and intuitively. Eye-tracking technology and numerous sensors, including sEMG and accelerometer sensors, can be used as multimodal signals to identify and decipher the intentions and movements of users. The EMG sensor captures electrical signals from the arm muscles, providing information about the user's intended hand movements. The accelerometer data measures hand acceleration, enhancing the accuracy of tracking, while eye-tracking technology detects the user's gaze for more precise interactions.

In other applications, users can interact with computers and other gadgets in a more intuitive and natural way by integrating these sensors. For instance, people can move their hands to control a robotic arm or their head and hands to move around a virtual environment.

Deep Learning, a field within Machine Learning, plays a crucial role in this pursuit, aiming to bring machine learning closer to the realm of artificial intelligence. Deep Learning is a valuable component of data science, encompassing statistics and predictive modeling, and greatly aids scientists in efficiently collecting, analyzing, and interpreting vast amounts of data. By leveraging Deep Learning techniques, these signals, can be fused to enhance gesture recognition, paving the way for more effective prosthetic control.

The primary objective of this work is to develop intelligent solutions capable of recognizing hand gestures from sEMG signals, accelerometer data, and other complementary signals. This approach can improve the accuracy and efficiency of prosthetic control, enabling amputees to perform a wider range of daily activities with greater ease and confidence.

To present the work conducted, this engineering thesis is structured as follows:

- The first chapter provides a comprehensive introduction to the employed signals, including sEMG, eye tracking, and accelerometer data, and their significance in hand gesture recognition. It also discusses the dataset utilized in this work.
- The second chapter introduces deep learning and its fundamental concepts, such as neural networks, convolutional neural networks (CNN), and recurrent neural networks (RNN). It explores the application of multimodal deep learning techniques to integrate and analyze data from multiple sources.
- The third chapter focuses on developing a deep learning-based approach for hand gesture recognition using sEMG signals as a unique modality. It presents the proposed model, which combines CNN and Long and Short Term Memory (LSTM), and explores the search for optimal hyperparameters to enhance system performance.
- In chapter four, the study concentrates on developing a hand gesture recognition system that utilizes sEMG signals, accelerometer data, and gaze information as inputs from multiple sources. This chapter presents the evolution of our solution from one source to three modalities. The performances of the proposed models are evaluated and compared to other approaches discussed in the related works.

Chapter 1

Multimodal perception of hand gestures

1.1 Introduction

Gesture recognition using wearable sensors and deep learning methods has emerged as a prominent field in human-computer interaction. However, one persistent challenge in this domain is the limited accuracy and generalizability of recognition models, especially when it comes to recognizing a wide range of gestures in various scenarios. To overcome this limitation, researchers are actively exploring innovative approaches.

One promising avenue is the utilization of wearable sensors, such as surface ElectroMyoGraphy (sEMG) sensors. These sensors capture electrical signals generated by muscle activity during hand gestures. By analyzing these signals, distinct patterns can be identified, enabling the recognition of specific gestures. Deep learning models, with their ability to extract complex features and to learn hierarchical representations, can effectively capture the intricate relationships within the sEMG signals and improve the accuracy of gesture recognition.

In addition to sEMG, other wearable sensors like accelerometers and gyroscopes offer valuable insights into hand motion and orientation. Accelerometers provide information about the acceleration and movement of the hand. Currently, some research-oriented sEMG electrodes are already equipped with three-axis accelerometers. By integrating data from these sensors with sEMG signals, a more comprehensive understanding of hand gestures can be achieved. Deep learning algorithms can be trained on multimodal data to capture the intricate dynamics between muscle activity, hand movement, and orientation, leading to robust and accurate gesture recognition.

Additionally, the use of eye-tracking technology expands the capabilities of multimodal gesture detection. It becomes possible to determine the user's focus of attention and associate it with particular gestures by observing the user's eye movements. The user's gaze direction can offer helpful hints about the intended meaning of a gesture, improving the interpretability and accuracy of gesture recognition systems. This association of multiple sensors types can build systems that are more resilient and adaptable to various users, environments, and applications by merging data from many sources.

The aim of this chapter is to study these sensors properties and the principal characteristic of the generated signals.

1.2 Electromyogram (EMG) signals

Electromyography (EMG) is an experimental technique that involves recording and analyzing electrical signals generated by muscles [1]. These signals, known as myoelectric signals, are produced by changes in the muscle fiber's physiological state. EMG can be measured non-invasively using electrodes on the skin (surface EMG) or invasively using needles inside the muscle (implanted EMG). Surface EMG is commonly used as it is safe and easy to conduct. The EMG signal represents the bioelectrical events associated with muscle contraction and is measured in millivolts (mV) [2]. By analyzing the EMG signal,

insights into muscle activity can be gained. A voluntary signal from the central nervous system that activates specific muscle fibers through the spinal cord and motor neurons causes a muscle contraction. The resulting EMG signal is the total of all the muscular action potentials and offers important details on the activity of the muscles during various motions, aiding in the comprehension of the mechanics of the human body.

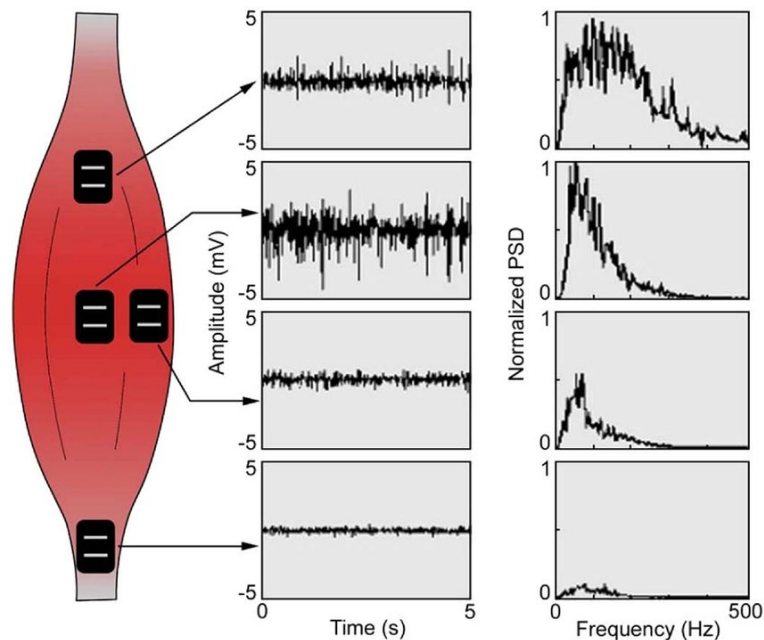


Figure 1.1: illustrates an example of sEMG signal.

1.2.1 The characteristics of EMG signals

The EMG (Electromyography) signal is indeed a complex physiological signal that reflects the electrical activity of muscles. It exhibits the several following characteristics [2], [3]:

- **Random:** The EMG signal can appear random due to the stochastic nature of muscle activation and motor unit recruitment.
- **Non-stationary:** The EMG signal is non-stationary because its statistical properties, such as mean and variance, change over time. This variability arises from factors like changes in muscle contraction levels and movement artifacts.
- **Non-linear:** The EMG signal is non-linear, meaning its waveform does not follow a linear relationship. The behavior of muscle fibers and the interactions between them result in non-linear characteristics.
- **Multi-component:** The EMG signal is composed of multiple components originating from different sources. Besides muscle activity, it may also contain interference from the heart's electrical activity (ECG) and brain activity (EEG) within the same frequency range. To separate EMG signals from ECG and EEG signals, a dominant frequency range of 20 Hz to 500 Hz is commonly chosen. This range ensures reasonable separation from other signals in the same frequency range.

- Regarding the amplitude and frequency ranges, the EMG signal can vary significantly. The amplitude typically ranges from $10 \mu\text{V}$ to 3 mV , depending on factors like muscle size, location, and the degree of contraction.

It's important to note that these characteristics and ranges can vary slightly depending on the specific context, measurement techniques, and the muscles being monitored.

1.2.2 Hand Gesture Recognition using EMG

Hand Gesture Recognition using EMG involves detecting hand gestures through analyzing the electrical signals generated by muscle movements. Different solutions have been developed using conventional methods and Deep learning-based approaches. It can be in a variety of applications, including robotics control, virtual reality systems, and prosthetic devices, the technique has yielded encouraging results.

The figure 1.2 demonstrates the use of robotic hand prostheses controlled by Electromyography (EMG) data. It shows a robotic hand, and electrodes attached to the user's arm, capturing electrical signals from the arm muscles during intentional hand movements. Advanced algorithms and control systems process and analyze the EMG data, allowing the robotic hand prosthesis to mimic the user's hand movements.

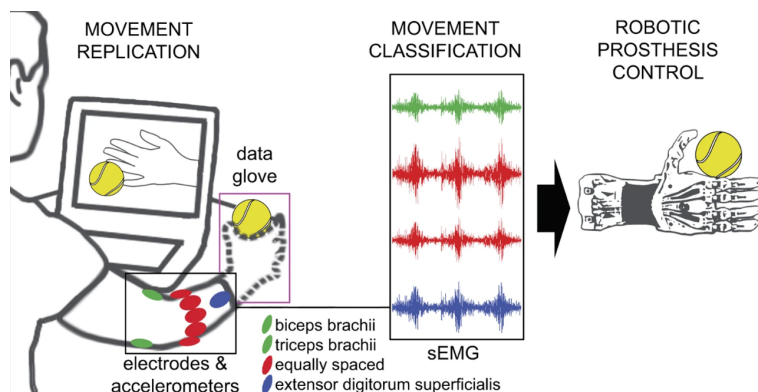


Figure 1.2: Robotic Hand Prostheses Controlled by Electromyography Data

1.3 Accelerometer data

Accelerometer data refers to the measurements and readings obtained from an accelerometer sensor that detects and quantifies the acceleration experienced by a device or wearable in three dimensions [4]. The accelerometer data typically includes acceleration values along the x, y, and z axes, representing the device's movement or changes in velocity in different directions. These measurements are often recorded as numerical values or time-series data, allowing for analysis and interpretation of motion patterns, trends, and specific motions. By analyzing accelerometer data, various applications and algorithms can leverage the information to enable motion detection, gesture recognition, activity tracking, and other functionalities that rely on understanding and utilizing motion-related information

[5].

The figure 1.3 depicts a graphical representation of labeled axes for an accelerometer. The x-axis represents movement in the horizontal plane, the y-axis represents movement perpendicular to the horizontal plane (e.g., vertical movement), and the z-axis represents movement perpendicular to both the x and y axes (e.g., depth or forward/backward movement).

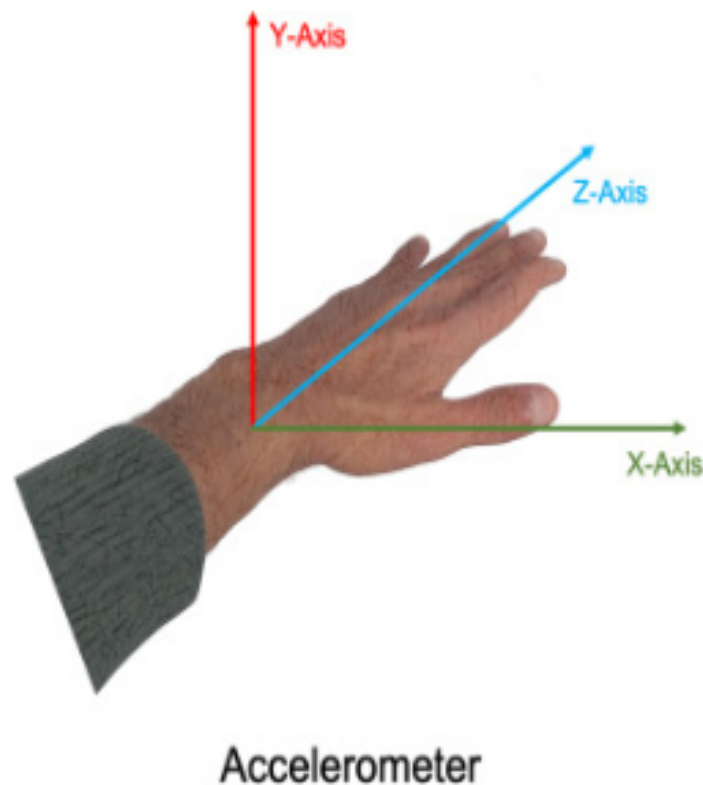


Figure 1.3: The flat movement measurement of the accelerometer (x, y and z axis)[6]

1.3.1 The characteristics of Accelerometer Data

Accelerometer data for hand gesture recognition possesses several characteristics that are essential for accurate classification. Here are some key characteristics [7], [8]:

- **Three-Axis Data:** Accelerometers capture data along three axes (X, Y, and Z), representing the three dimensions of physical space. This comprehensive data can provide a detailed representation of hand movements.
- **Amplitude and Magnitude:** Accelerometer data reflects the acceleration experienced by the hand during gestures. The amplitude of the acceleration indicates the intensity or force of the gesture, while the magnitude combines the acceleration values from all three axes to represent the overall movement.
- **Noise and Artifacts:** Accelerometer data is susceptible to noise and artifacts. Factors like sensor quality, device orientation, and external disturbances can introduce

unwanted noise. Preprocessing techniques, such as filtering and noise reduction, are used to improve the signal quality.

- **Multiclass Classification:** Hand gesture recognition often involves differentiating between multiple gestures or classes. The accelerometer data needs to be labeled with corresponding gesture classes for supervised learning.

Considering these characteristics, accelerometer data provides valuable information for accurate hand gesture recognition. It captures the dynamics, intensity, and orientation of hand movements.

1.3.2 Hand Gesture Recognition Using Accelerometer Data

In the context of hand gesture recognition for prosthetics, accelerometer data refers to the measurements obtained from an accelerometer sensor that is integrated into a prosthetic hand or limb. In the context of hand gesture recognition for prosthetics, accelerometer data refers to the measurements obtained from an accelerometer sensor that is integrated into a prosthetic hand or limb. Figure 1.4 illustrates the positioning of accelerometers on the hand to capture the hand's acceleration. The accelerometer data is processed and analyzed to identify the patterns and movements associated with different hand gestures. Algorithms and machine learning techniques can be applied to the accelerometer data to train models that can recognize and classify the intended gestures accurately. Prosthetic hand systems can comprehend the user's intended hand movements and transform them into appropriate actions by regulating the motors or actuators inside the prosthetic hand by gathering and analyzing accelerometer data. As a result, users are able to restore dexterity and control over their prosthetic limb, improving their ability to carry out various duties and activities.

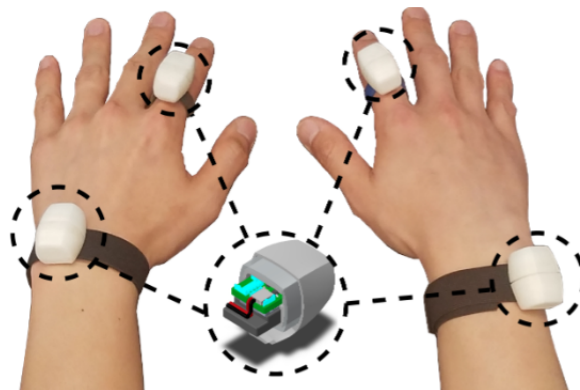


Figure 1.4: Sensor of a three-axis accelerometer enclosed in a waterproof case

1.4 Eye tracking data

Eye tracking is a method for observing and analyzing eye movements to determine a person's direction of gaze and visual attention. Cameras are commonly used to capture images of the eyes, which are then processed to calculate properties related to eye

movements. The gaze direction and point of gaze in 3D coordinates can be estimated by analyzing the pupil and corneal reflection. Eye tracking techniques can be classified as intrusive or non-intrusive, depending on whether the recording device makes direct contact with the user [9]. Invasive methods offer higher accuracy but require direct contact, while non-invasive methods use remote cameras. Infrared corneal reflection technology is commonly used [10], analyzing pupillary reflectance and corneal gloss to determine the gaze direction (Figure 1.5). Eye tracking provides real-time data on eye movements, enabling insights into cognitive, emotional, and physiological states. It has various applications in rehabilitation, assistance, and understanding human behavior and interactions.

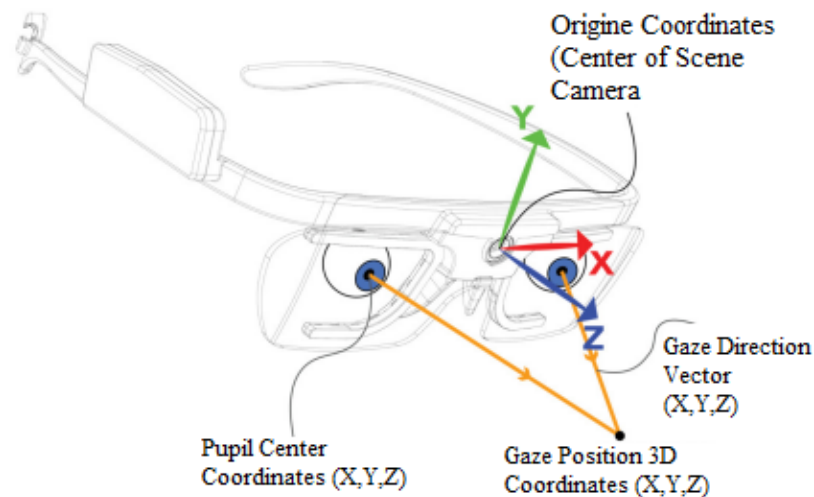


Figure 1.5: Overview of the gaze quantities calculated by a head mounted eye tracking system[10]

1.4.1 The characteristics of Eye tracking in gesture recognition

The characteristics of eye tracking data for hand gesture recognition include [10]–[12]:

- Eye movement information: Eye tracking captures saccades, fixations, and gaze direction, providing insights into visual attention and intention.
- Spatial data: Eye tracking measures eye positions, aiding in determining the relative positions of hands and gestures.
- Temporal data: Eye tracking data collected over time enables analysis of temporal patterns and sequences of eye movements, crucial for recognizing dynamic gestures.
- Non-invasiveness: Eye tracking is non-intrusive, relying on optical or infrared sensors for comfortable user experience.
- Robustness to lighting conditions: Eye tracking technology works effectively under various lighting conditions, ensuring reliable gesture recognition.
- Individual uniqueness: Eye movement patterns are unique to individuals, allowing for personalized gesture recognition and adaptations.

Leveraging these characteristics, eye tracking data enhances hand gesture recognition, enabling intuitive and natural human-computer interactions.

1.4.2 Hand Gesture Recognition Using Eye tracking

Eye tracking technology can enhance hand gesture detection systems by providing additional contextual information. It records eye motions, such as pupil size, gaze direction, and movement patterns, using specialized cameras or sensors . This data can be used in various ways, including anticipating gestures based on gaze direction and detecting subtle hand motions that may be challenging to capture with hand tracking alone . By combining hand and eye tracking data, the system can improve accuracy and provide user feedback, such as visual or auditory confirmation [12]. Eye-tracking data can also aid in the segmentation of hand regions, reducing false positives or negatives in gesture detection. Overall, eye tracking enhances hand gesture recognition by providing insights into user intent and attention, resulting in a more intuitive and natural user experience.

The figure 1.6 show gaze points are superimposed onto the scene camera video, showcasing the subject's visual fixations. Each fixation is represented by a circle, with its diameter indicating the duration of the fixation, and a number indicating the order of the fixations. The subject's task in this particular scenario was to grasp the door handle and the bottle

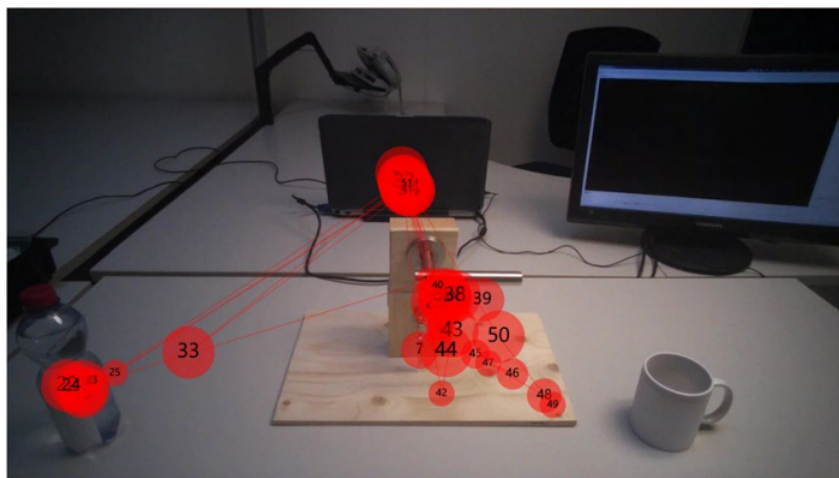


Figure 1.6: Visualizing Gaze Points Overlapped onto Scene Camera Video: An Illustrative Example[12]

1.5 Description of the dataset MeganePro[12]

The dataset "MeganePro Dataset" was published by the University of Applied Sciences Western Switzerland (HES-SO Valais), Istituto Italiano di Tecnologia (IIT), and University Hospital Zurich (UHZ) in 2019 [12]. It has been gathered using Myo-Electricity, accelerometers and Gaze.

The MeganePro team created a set-up (Figure I.7) for collecting sEMG from the forearm, gaze and visual data.

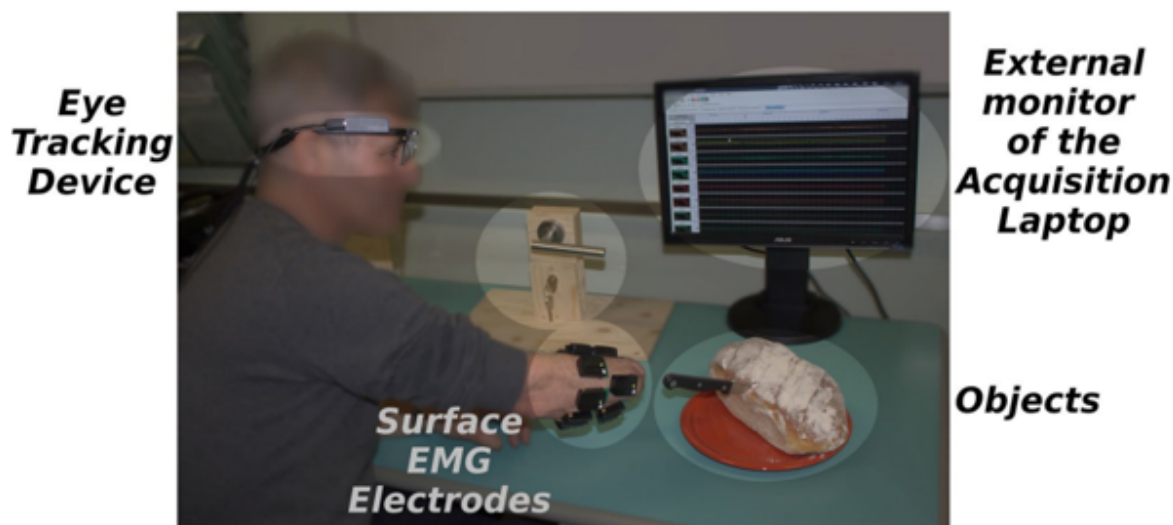


Figure 1.7: Eye tracking data device and sEMG electrodes for an acquisition[12]

1.5.1 Dataset Acquisition

The Delsys Trigno Wireless sEMG System (Delsys Inc., USA) is made up of sEMG electrodes that are utilized to record muscle activity from the participants' forearms. The inter-sensor delay is less than 500 s, the baseline noise is less than 750 nV RMS, and the sampling rate for the sEMG signal is 1926 Hz [10]. A three-axial accelerometer that gathers data at a sampling rate of 148 Hz is built into each electrode. The electrodes and a base station that is joined to the laptop communicate wirelessly. Figure 1.8 illustrates the electrodes and the Tobii Pro Glasses 2 used for the acquisition process. To collect data on gaze and vision, the Tobii Pro Glasses 2 developed by Tobii AB, Sweden (see Figure 1.8) were utilized. The gaze and information connected to the gaze are captured at 100 Hz. The data from the glasses is wirelessly sent to the laptop and locally stored on a portable recording device that is linked to the glasses via a cable. Additionally, the recording device has a battery that has a maximum recording time of about 120 minutes.



Figure 1.8: Tobii Pro Glasses 2 and sEMG electrodes [10]

The dataset was built from 29 people while they made a variety of static and dynamic hand gestures. For these experiments, 10 grasps are chosen, each of which may be used to move repeatedly three objects in a natural way. Standing and sitting positions were used for all static grasps. During the dynamic portion, a subset of the earlier objects was functionally handled using the same grasps.

Twelve sEMG electrodes with incorporated three-axial accelerometers were implanted in two arrays around the right forearm before the activity as shown in figure I.8 A detailed description of this dataset is available in[12]

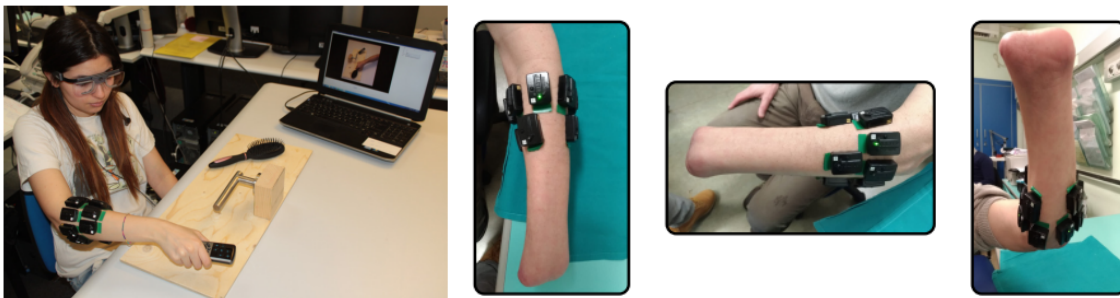


Figure 1.9: Overview of the acquisition protocol and setup during the execution of a task[10]

The table 1.1 [12] summarizes the ID and name of different grasp types , along with the ID and name of the objects involved in the grasping action. In some cases, there is further refinement specifying the ID and name of the specific object parts used in the grasping. Additionally, the table includes a column indicating the vocal command given to the subjects.

Table 1.1: Overview of the grasp types and objects for the static condition of the exercise

Grasp		Object		Object Part		Vocal Instruction	
1	medium wrap	1	bottle	1	bottle	take the bottle	
		2	can	2	can		can
		3	door handle	3	door handle		door handle
2	lateral	4	mug	23	handle		mug
		5	key	5	key		key
		24	pencil case	6	zip		zip
3	parallel extension	7	plate	7	plate		plate
		8	book	8	book		book
		9	drawer	9	drawer		drawer
4	tripod grasp	1	bottle	10	cap		cap of the bottle
		4	mug	4	mug		mug
		9	drawer	11	knob		knob of the drawer
5	power sphere	12	ball	12	ball		ball
		13	bulb	13	bulb		light bulb
		5	key	5	key		keys
6	precision disk	15	jar	26	lid		jar
		13	bulb	13	bulb		light bulb
		12	ball	12	ball		ball
7	prismatic pinch	16	clothespin	16	clothespin		clothespin
		5	key	27	keyring		keys
		2	can	25	pull tab		can
8	index finger extension	21	remote	17	button	point at a button of the remote	
		18	knife	18	knife	take the knife	
		19	fork	19	fork		fork
9	adducted thumb	20	screwdriver	20	screwdriver		screwdriver
		21	remote	21	remote		remote
		22	wrench	22	wrench		wrench
10	prismatic four finger	18	knife	18	knife		knife
		19	fork	19	fork		fork
		22	wrench	22	wrench		wrench

1.5.2 Dataset size and organisation

First of all, all modalities (sEMG, Accelerometer, Gaz) were resampled at the original 1926 Hz sampling rate of the sEMG stream. Each modality has 1926 data points per second.

The dataset size can be defined as the number of signals per gesture per subject. Each subject performed 10 grasps each grasp with 10 repetitions of the manipulation of the same object. Therefore, it is composed of 100 signals per subject, with a total of 29 subjects the data contains 2900 signals each one with around 15000 data points. Finally, the dataset has a size of 40M row for each modality.

sEMG dataset contains 12 columns for 12 electrodes each one with three columns representing the three axial accelerometer (x,y,z), and 2 columns for eye-tracking data representing the x and y coordinates of the plane. Figures 1.10, 1.11, 1.12 shows examples of sEMG, eye tracking and accelerometer signals.

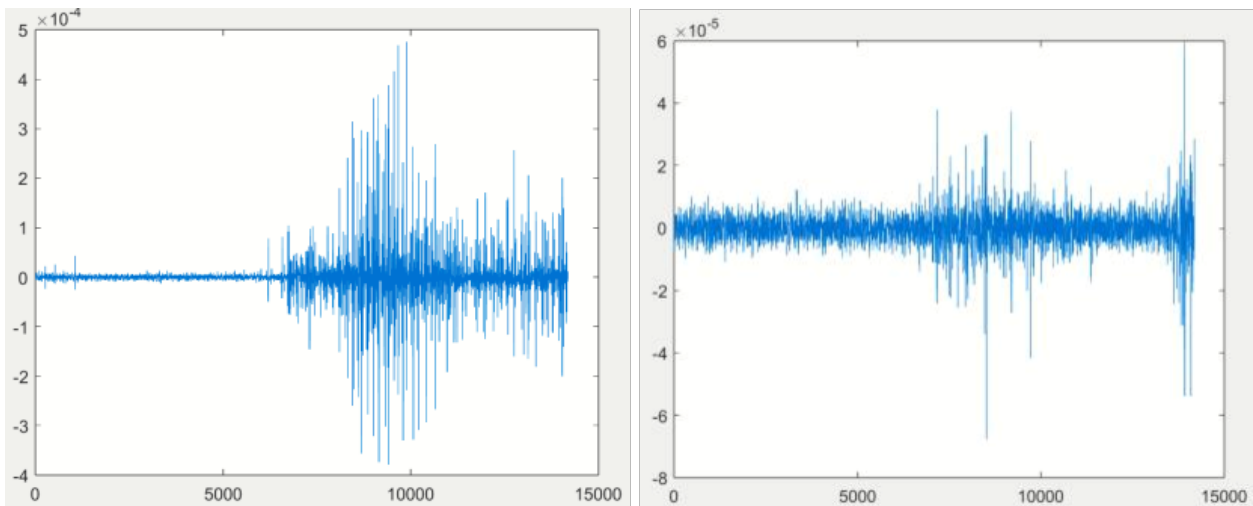


Figure 1.10: sEMG signals

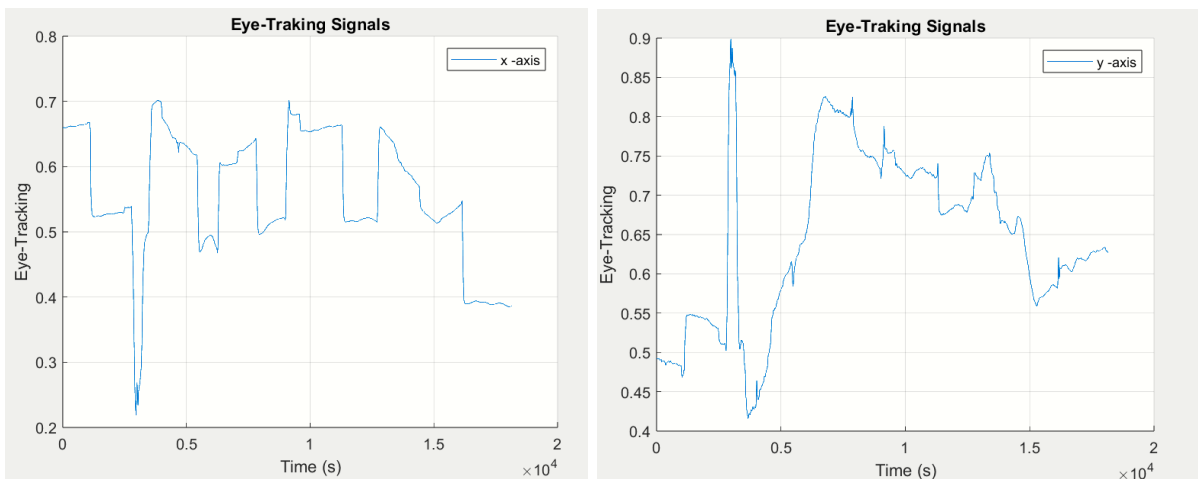


Figure 1.11: Eye tracking signals

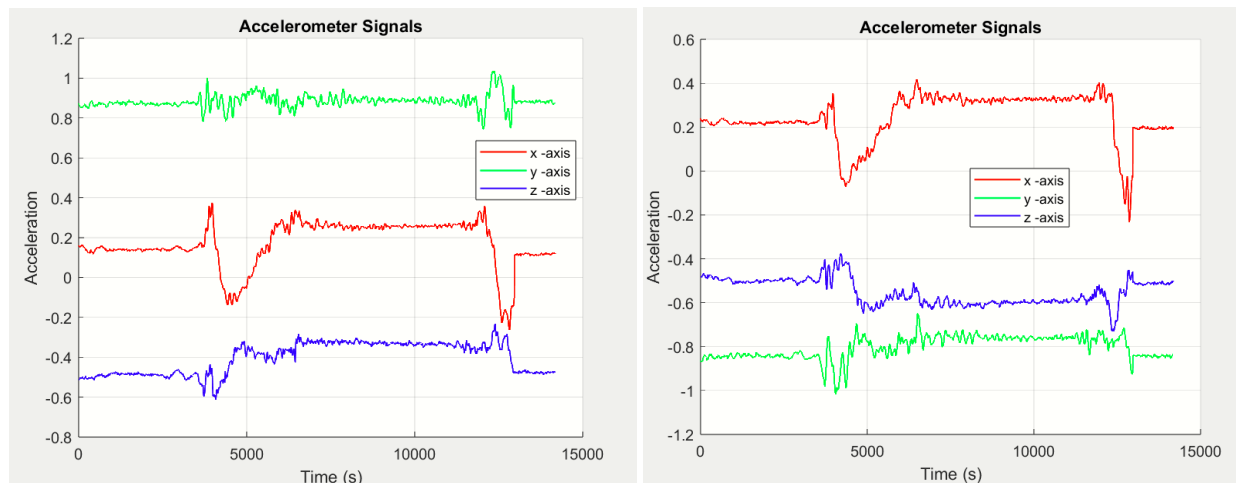


Figure 1.12: Accelerometer signals

The sEMG signals are down-sampled (reduce the sampling frequency). This process is frequently used to reduce the size of a dataset while keeping the important details and properties of the original data. The sampling rate may be fairly high when working with high-frequency data, such as sensor measurements or time series data, producing a large amount of data. In this work, we made downsampling with a factor of 10. This factor can reduce the number of data points while attempting to retain the essential features and dynamics of the original signal. This reduction in data size can lead to computational efficiency, faster processing times, and improved resource utilization, especially when dealing with large datasets or limited computing resources [12].

Figure 1.10 shows examples of sEMG signals.

1.5.3 Data pre-processing

Working with EMG (Electromyography) data requires many pre-processing steps, including data scaling or normalization. The amplitude of the EMG signals can vary greatly based on a number of variables, including the degree of muscle activation, the location of the electrodes, and individual variances.

For many EMG studies, such as feature extraction, classification, or pattern recognition, data scaling or normalization is crucial. It enhances the stability of algorithms or models, helps to mitigate the effect of amplitude changes, and makes the results more generalizable. It made the range and the distribution of the EMG data more uniform and acceptable for analysis. To ensure that the size of the data does not create bias or distort the analysis results, the fundamental objective of data scaling is to scale all of the EMG signals to a similar scale [13], [14].

The z-score normalization, commonly referred to as standardization, is a popular technique for scaling EMG data [14]. Using this method, the data is transformed to have a mean of zero and a standard deviation of one. As a result, the EMG signals have a uniform scale and are centred on zero, making it simpler to compare and analyse the data

from various participants and trials.

When working with a multimodal model that takes input from multiple sources with different scales, it is generally advisable to standardize or normalize each modality separately. This ensures that the data from different modalities are on a similar scale, enabling the model to learn from all modalities effectively [15], [16].

To ensure consistency in the scaling across all modalities, we may consider applying scaling or normalization techniques to the accelerometer and eye-tracking data as well. This will help ensure that all modalities have similar scales. Scaling the data from each modality individually before feeding it into the multimodal model helps prevent any single modality from dominating the learning process due to differences in scales [17]

1.5.4 Data Segmentation

A crucial step in signal processing is signal segmentation, which entails breaking a continuous signal into manageable windows or smaller pieces. This method is frequently used to examine and process signals in a more focused and useful way, such as time series data or sensor measurements [18], [19].

The windowing technique is one often used method for segmenting signals. Windowing is the process of dividing a continuous signal into fixed-duration windows that may or may not overlap [17], [18] as shown in figure 1.13. Each window represents a short segment of the original signal.

Windowing has many benefits for signal processing. It allows for the use of localized analytic methods inside each window, enabling the extraction of time-varying characteristics, the discovery of patterns, or the computing of statistical measures within certain signal segments. The continuity of the studied data can also be improved by the overlap between adjacent windows, which can assist minimize edge effects and provide a smoother transition between segments.

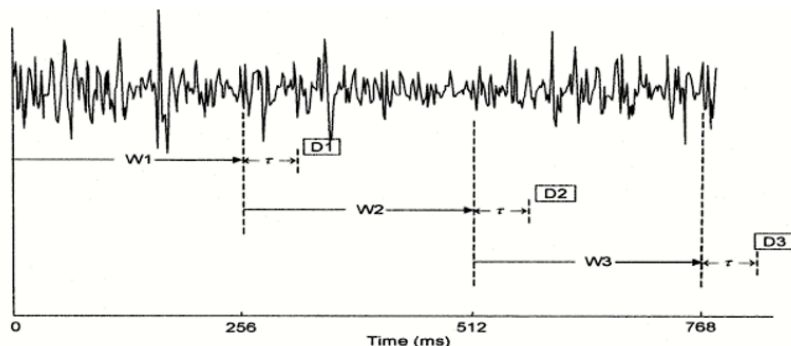


Figure 1.13: Data segmentation

In signal processing and data analysis, a sliding window is a method that makes that overlap between windows including moving a fixed-size window or interval through a se-

ries of data points. The window moves progressively, usually by one data point at a time, enabling the study of successive data segments.

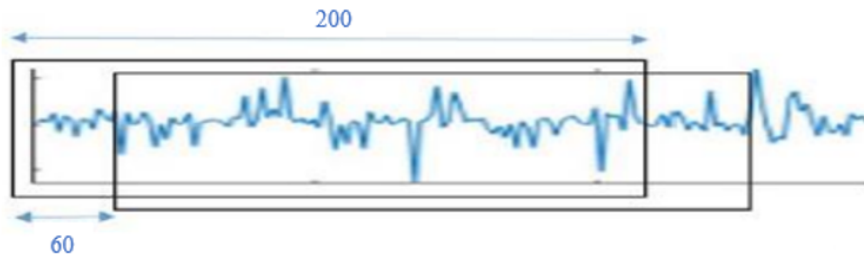


Figure 1.14: Sliding window technique

1.5.5 Data preparation

The initial dataset included 40 million rows for each modality, which made analysis quite computationally intensive. We used downsampling methods to keep a representative sample of the original data while reducing the dataset size to 4 million rows. The dataset was then divided into smaller, more manageable windows of 200 rows each using data segmentation. This window size allows for the recording of temporal data within each window, which corresponds to a time period of 1 second. A 30% overlap between successive windows was established, resulting in a 150-millisecond gap, to assure continuity and prevent information loss at window borders. The final dataset size was reduced to 11 Millions raws greatly enhancing computing performance without compromising important data.

1.6 Conclusion

In conclusion, this chapter has provided an overview of Electromyography (EMG), accelerometer data, and eye tracking signals. We discussed the generalities of these signals and their relevance to hand gesture recognition. Additionally, we presented different approaches for hand gesture recognition using sEMG, accelerometers, and eye tracking.

The review of related work highlighted the important properties of these signals that need to be considered in the classification process. Furthermore, we gained insights into the dataset used in our work and discussed the preprocessing steps involved.

By understanding the characteristics and preprocessing requirements of these signals, we can effectively analyze and classify hand gestures, laying the foundation for further research and development in the field of gesture recognition.

Chapter 2

Deep Learning

2.1 Introduction

Deep learning is a subfield of machine learning that focuses on train artificial neural networks to learn and make decisions similarly to human brains. To handle and interpret complicated data, it entails creating and training artificial neural networks with numerous layers of interconnected nodes.

Deep learning algorithms may handle tasks like picture and speech recognition, natural language processing, and predictive analytics by automatically learning and extracting meaningful patterns or representations from massive amounts of labeled or unlabeled data. The "deep" in deep learning refers to the complexity of neural networks, which include several hidden layers that let them pick up hierarchical data representations.

Deep learning has several advantages over traditional machine learning methods. First, it acts directly on raw signals, avoiding time-consuming preprocessing and feature engineering.

Furthermore, deep learning can learn high-quality representations even when the input is corrupted and noisy [20]. In addition, deep neural networks can capture high-level representative features and latent dependencies through deep structures.

This chapter give the theoretical background and introduces the concepts used in the development of our solution.

2.2 Neural networks

2.2.1 Neuron model

A neuron model is a mathematical representation of the behavior of a biological neuron (the basic unit of the nervous system). Figure 2.1 depicts the mathematical model of a single neuron.

Input the data provided to a model for training, prediction, or classification. This data can take various forms such as images, text, audio or digital data.

Weight In machine learning, weights are used to assign importance to the different features in the input data.

Bias The bias term is similar to the intercept in a linear function. It is added to the weighted summation of its inputs.

Activation Function The activation function is used to introduce nonlinearity in the model's output. Without this nonlinearity, the model produces only linear combinations

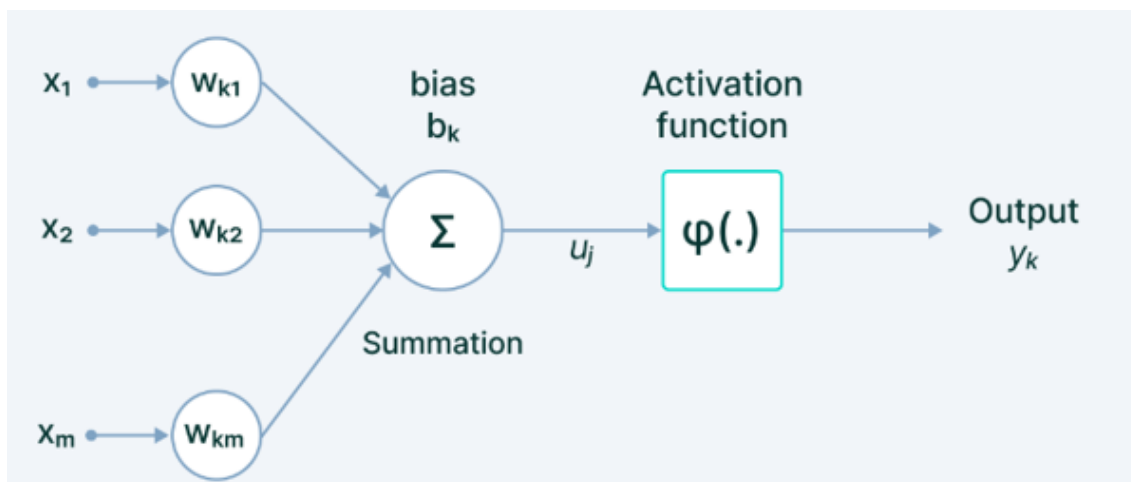


Figure 2.1: The various components of a neuron

of the input values, limiting its ability to model complex relationships in the data.

$$Y = \delta \left(\sum_i w_i x_i + b \right) \quad (2.1)$$

Where, y represents the output of the neuron, δ is the activation function (which is typically non-linear), w_i are the weights, x_i are the inputs, and b is the bias of the neuron. However, it should be noted that a single neuron is only capable of solving binary linear classification problems and is not able to simulate more complex non-linear functions.

Table 2.1 resumes the more popular activation functions

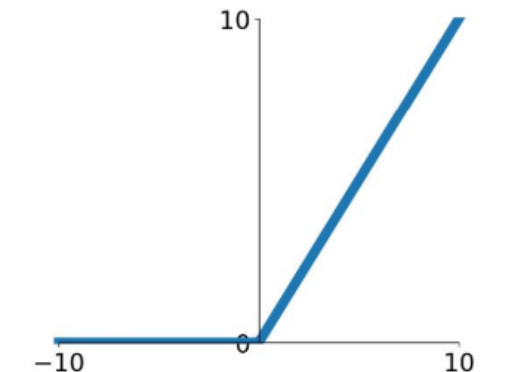
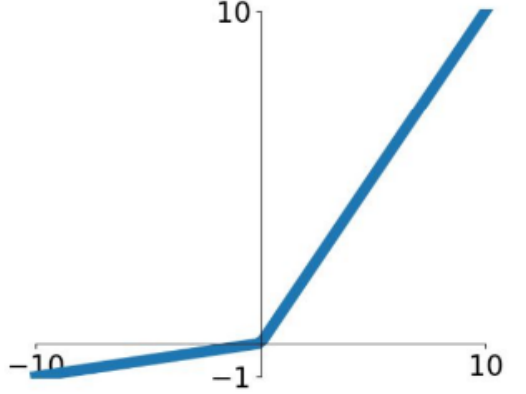
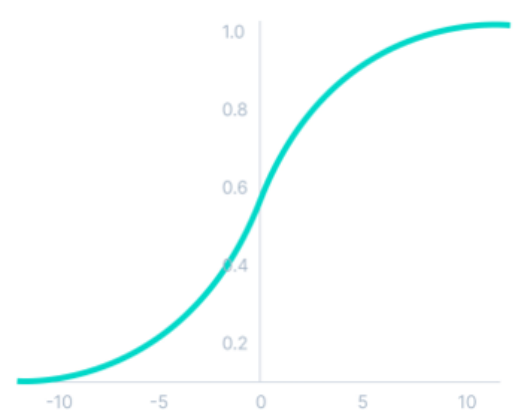
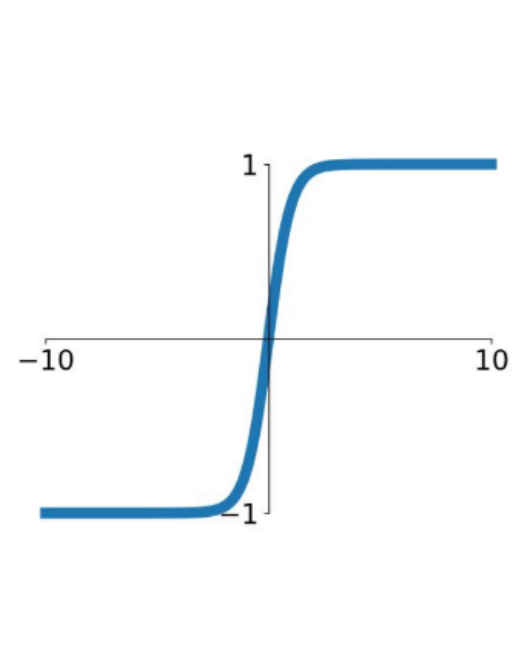
<p>Rectified Linear Unit (ReLU) ReLU sets the output to zero for negative input values and retains the input value for non-negative inputs. The mathematical representation of ReLU is as follows:</p> $\sigma(x) = \max(0, x) \quad (2.2)$	
<p>Leaky ReLU The Leaky ReLU function is similar to ReLU, but has a small slope for negative input values. This is meant to address the "dead ReLU" problem where some neurons become permanently inactive. Mathematically it can be represented as:</p> $f(x) = \max(0.1x, x) \quad (2.3)$	
<p>Softmax Function The softmax function is often used in the output layer of a multi-class classification task. It maps the input values to a probability distribution over the possible classes. Mathematically it can be represented as:</p> $\text{softmax}(z_i) = \frac{\exp(z_i)}{\sum_j \exp(z_j)} \quad (2.4)$	
<p>Hyperbolic Tangent Function Tanh maps input values to a range between -1 and 1, creating a smooth and symmetric function centered around zero. Tanh introduces non-linearity, enabling the network to capture complex relationships. It has a strong gradient around zero, facilitating faster convergence during training. Tanh is commonly employed in hidden layers to introduce non-linearities and capture intricate patterns. Mathematically it can be represented as:</p> $f(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}} \quad (2.5)$	

Table 2.1: Popular activation functions

2.2.2 Forward propagation

Equation (2.6) refers to the calculation of the activations or outputs of each layer in a neural network during forward propagation. Specifically, given the input features or sample vector x , the activations of the hidden layers $a^{(l)}$ ($l=1,2,\dots,L-1$) can be calculated recursively as:

$$a^{(l)} = \phi(W^{(l)}a^{(l-1)} + b^{(l)}) \quad (2.6)$$

where $W^{(l)}$ is the weight matrix for layer l , $b^{(l)}$ is the bias vector for layer l , ϕ is the activation function (e.g., sigmoid, ReLU, etc.), $a^{(l-1)}$ is the input to layer l , and $a^{(l)}$ is the output or activation of layer l . The input layer is typically not counted as a layer in the network, so the first hidden layer is considered to be layer 1 (i.e. the input layer is layer 0). Therefore, $a^{(0)} = x$ refers to the activation of the input layer, which is simply the input features or sample vector itself.

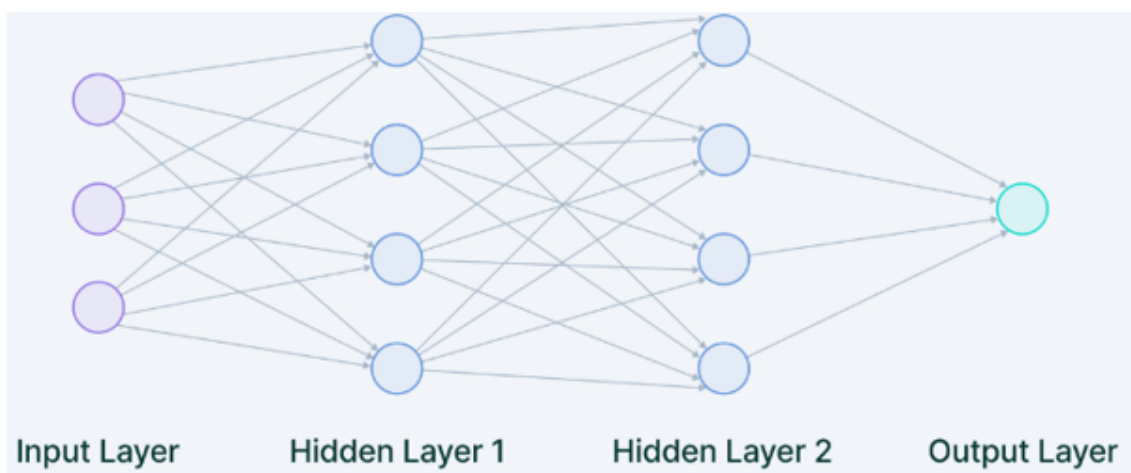


Figure 2.2: Multi-layer neural network[21]

Input Layer The input layer is the first layer in a neural network that receives input data. It passes this information to the next layers.

Hidden Layers Hidden layers in a neural network are responsible for processing and extracting features from the input data. They are called "hidden" because their computation is not directly visible in the output. Deep learning networks have multiple interconnected hidden layers that enable them to search for increasingly complex features in data.

Output Layer The output layer takes input from the preceding hidden layers and produces the final prediction based on the model's learned features. In classification or regression models, the output layer typically has a single node or multiple nodes, depending on the problem's complexity. The number of nodes in the output layer and the activation function used depends on the specific problem being solved..

In neural networks, forward propagation involves passing input features through the network to obtain predictions. Activations are computed layer-by-layer using weights, biases, and activation functions.

2.3 Loss function

The loss function in a neural network quantifies the difference between predicted and actual outputs [22]. It guides training by minimizing this difference. Cross-entropy loss is popular for classification tasks, penalizing high-confidence incorrect predictions. Mean squared error (MSE) is common for regression tasks. Other loss functions like mean absolute error (MAE) can also be used. The choice depends on the problem.

2.4 Backpropagation

The aim of the backpropagation algorithm is to calculate the impact of every weight and bias in the network on the loss function. By doing so, it becomes possible to modify the weights and biases using an appropriate learning algorithm such as stochastic gradient descent. The goal of backpropagation is to find:

$$\frac{\partial L}{\partial W_{jk}^{(l)}} \tag{2.7}$$

Where $W_{jk}^{(l)}$ represents the weight associated with the connection between the k-th neuron in the (l-1) th layer and the j-th neuron in the lth layer of a neural network. These values can be found by recursively applying the chain rule from the output of the network to the input.

2.5 Optimization Algorithms

Adaptive optimization algorithms improve neural network performance by adjusting the learning rate over time or accumulating the gradient. This is different from traditional optimization techniques that use a fixed learning rate. Popular neural network optimizers include Stochastic Gradient Descent (SGD), SGD with momentum, RMSProp, RMSProp with momentum, AdaDelta, and Adam. The choice of algorithm depends on the problem and data characteristics.

2.5.1 Stochastic Gradient Descent

Stochastic Gradient Descent (SGD) is an optimization algorithm used in machine learning to train models by updating their parameters based on gradients calculated from randomly selected mini-batches of training data. It balances computational efficiency with convergence speed and enables effective training of large-scale and deep learning models. SGD gradually minimizes the loss function by adjusting parameters in the opposite direction of the gradients. It can escape local minima and explore the parameter space efficiently but may result in slower convergence and fluctuations. Variants like mini-batch SGD and momentum-based methods have been developed to improve performance and stability.

2.5.2 Adam (Adaptive Moment Estimation)

Adam is an optimization algorithm for deep learning that combines Momentum and RM-Sprop. It adjusts learning rates based on past gradients and second moments to optimize parameter updates. By incorporating momentum and adaptive rates, Adam improves training by maintaining the right direction and speed. It converges faster and delivers good results by default, surpassing traditional stochastic gradient descent (SGD). It is widely used in deep learning tasks.

2.6 Regularisation

During the process of training a model, the aim is to minimize the loss function based on a training set. However, it is possible for the model to start overfitting on the training data, which means it can perfectly classify the training data but perform poorly on test data due to memorizing peculiarities of the training set, such as noise [22]. To prevent overfitting, several regularization strategies have been developed to improve the generalization of the model. Common regularization methods used in deep learning include:

- Data augmentation
- Dropout
- L1 regularization
- L2 regularization
- Early stopping

Some methods are presented below.

2.6.1 Dropout

The key idea behind dropout is to randomly drop units and their connections during the training phase [23]. This technique has been shown to effectively improve the generalization of the model by introducing noise into the system, forcing it to learn more global features instead of focusing on a single feature. The random dropping of nodes can be seen as training multiple smaller models in an ensemble, further enhancing the model's performance. Overall, dropout is a powerful technique that can help prevent overfitting and improve the robustness of the model.

2.6.2 Early stopping

During training, models are usually evaluated on a validation set separate from the training set. As the model trains, its performance on the validation set is monitored, and if validation accuracy begins to decline or plateaus, training is terminated early, before the model begins to overfit the training data as illustrated in figure 2.4. Stopping training

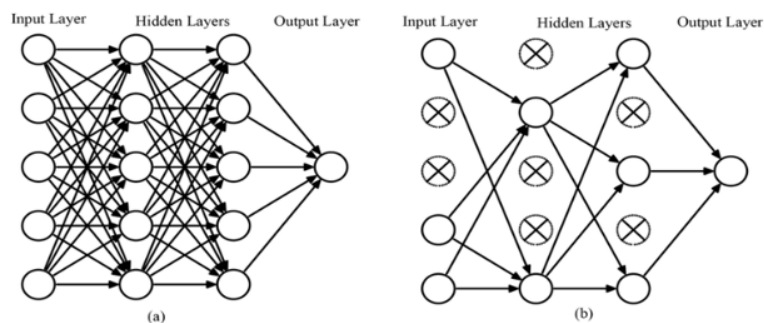


Figure 2.3: Dropout Strategy. (a) A standard neural network. (b) Applying dropout to the neural network on the left by dropping the crossed units[23]

early prevents the model from remembering the training data and generalizing better to unseen data. This can improve the overall performance of the model on new data. Although this method does not improve validation accuracy during training, it results in the best performing model. Early stopping is a simple but effective way to prevent overfitting and can be easily implemented in most machine learning frameworks [24].

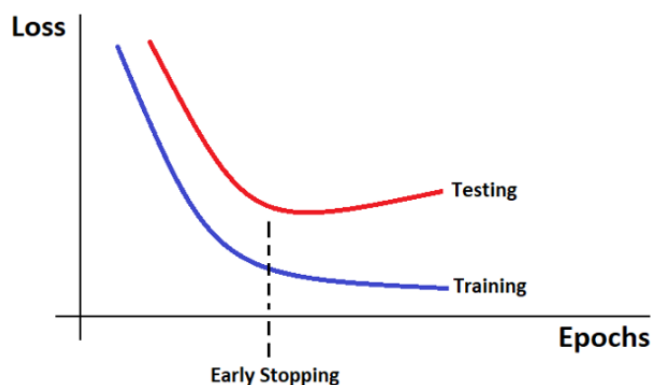


Figure 2.4: Early Stopping Strategy[24]

2.7 Deep Learning Foundations

Deep learning algorithms are divided into several subcategories based on their purpose: **Discriminative deep learning models** are able to classify the input data according to a predefined label based on the adaptively learned discriminative features. These discriminative algorithms are able to learn distinctive features by nonlinear transformation, and perform classification through probabilistic prediction[25]. Thus, these algorithms can play the role of both feature extraction and classification. The most common discriminative architectures include multilayer perceptron (MLP), recurrent neural networks (RNN), and convolutional neural networks (CNN), as well as their variations.

Hybrid deep learning models are models that combine multiple deep learning models. An example of a common hybrid deep learning model uses an algorithm to extract features and discriminative algorithms to perform classification for example a CNN-RNN

model.

Discriminative deep learning models are the most popular and powerful algorithms in muscle signal research because recognition of muscle signals is their main task. For a dataset of muscle signal samples x , y , where x represents the set of EMG signal observations and y represents the set of ground truth samples (i.e. labels). The goal of discriminative deep learning models is to learn a function $x \rightarrow y$ mapping. In summary, discriminative models receive the input data and output the corresponding category or label. All the discriminative model techniques presented in this section are supervised learning techniques, which require information from observations and ground truth.

2.8 Convolutional neural networks (CNNs)

A Convolutional Neural Network (CNN) is a deep learning architecture designed for processing grid-like input, like images or sounds. CNNs are widely used in computer vision tasks such as image classification, object detection, and image segmentation. They utilize convolutional layers to extract meaningful features by applying filters across the input grid, capturing local patterns and spatial hierarchies. This allows CNNs to automatically learn relevant representations for complex visual tasks. Pooling layers are employed to reduce spatial dimensions, improving translation robustness and computational efficiency. CNNs have revolutionized computer vision and are a crucial component in modern deep learning models for visual recognition.

2.8.1 The basic components of a CNN

A typical CNN consists of several layers, including convolutional layers, pooling layers, and fully connected layers (figure 2.6). The input data is fed into the network and processed layer by layer until the final output is produced. The layers of a CNN are interconnected in a way that allows the network to automatically learn and extract features from the input data.

Convolutional Layer

These layers perform the operation of convolution, which involves applying filters or kernels to the input data to extract features [25]. The filters slide over the input data, calculating dot products between the filter values and input values, resulting in a feature map that represents the presence of specific features in different regions of the input.

Pooling layers

These layers reduce the spatial dimensions (width and height) of the feature maps by aggregating neighboring values using operations such as maximum (max-pooling) or average (average-pooling). This helps reduce computational complexity while preserving important features and providing some translation invariance.

In Max pooling layer, the maximum element is selected from each window or region of the input feature map. This process helps identify the most dominant features within each window (Figure 2.5).

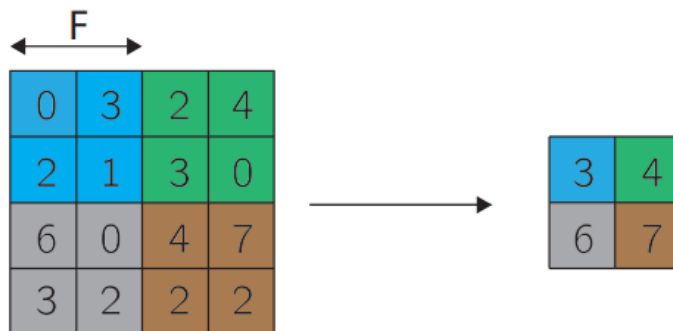


Figure 2.5: Example of a max pooling operation with spatial extent $F = 2 \times 2$ and stride $S=2$

In average pooling, the average value of the elements within a specific region of the feature map is calculated. It simply averages the features from the feature map.

Fully connected layers

These layers act like traditional Multilayer Perceptron where each neuron is connected to all neurons in the previous layer. Fully connected layers are typically used at the end of a CNN to combine extracted features and make predictions or classifications based on them.

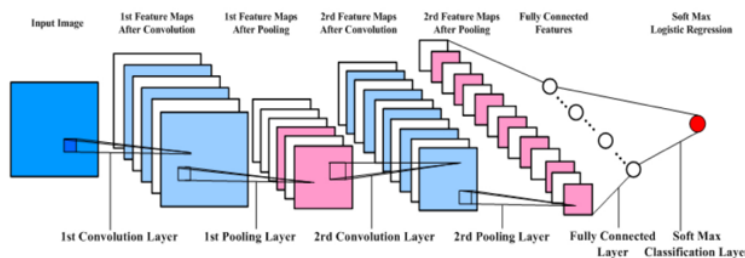


Figure 2.6: Illustration of a typical CNN architecture with two convolution layers, two max pooling layers, one fully connected layer, and one soft max classification layer

2.8.2 Understanding Conv1D and Conv2D

Conv1D and Conv2D are convolutional operations used in deep learning for processing sequential and image data, respectively. Conv1D, short for one-dimensional convolution, is applied to 1D inputs, such as sequences, time series, or text. It involves sliding a set of

filters (kernels) along the input sequence, capturing local patterns and extracting relevant features. Conv1D is commonly used in tasks like natural language processing, speech recognition, and audio analysis [26], [27].

Conv2D, on the other hand, refers to two-dimensional convolution. It operates on 2D inputs, typically images or spatial data represented as matrices. By sliding convolutional filters across the input in both horizontal and vertical directions, Conv2D effectively captures spatial relationships, detecting edges, textures, and higher-level image features. Conv2D is widely used in computer vision tasks, including image classification, object detection, and image segmentation.

Both Conv1D and Conv2D rely on shared weights within each convolutional filter, enabling the models to learn and generalize spatial patterns across different locations of the input. These convolutional operations have been instrumental in achieving state-of-the-art performance in various domains by automatically learning hierarchical representations from raw data.

The figure 2.7 illustrates that when using a 1D CNN, the data must have 1 spatial dimension. This implies that each sample needs to be 2D, consisting of a spatial dimension and channels. Consequently, the X-train input should be a 3D tensor, with dimensions including batch size, spatial dimensions, and channels.

Likewise, in the case of a 2D CNN, there would be 2 spatial dimensions, such as height and width (H, W), and the samples would be 3D, with dimensions including height, width, and channels (H, W, Channels). The X-train input would then be represented as (Samples, H, W, Channels).

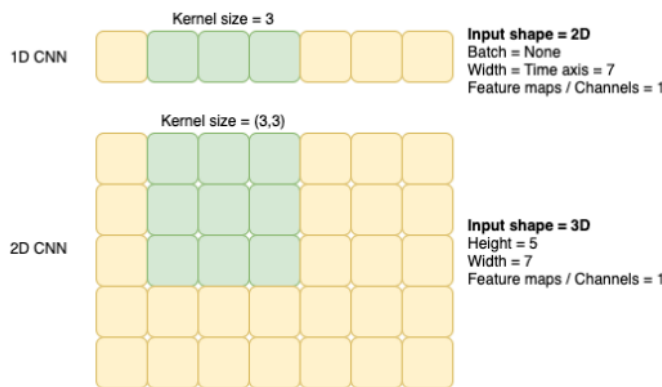


Figure 2.7: Data Structure Requirements for 1D and 2D Convolutional Neural Networks[28]

2.8.3 Advantages and limitations of CNNs

CNNs have several advantages, including the ability to automatically learn and extract features from input data, scalability to large datasets, and high accuracy in image recognition tasks. However, CNNs also have some limitations, such as their high computational cost and dependence on large amounts of training data. Additionally, CNNs may not

perform well in cases where the input data is very different from the training data, or where the task requires reasoning or understanding beyond simple pattern recognition.

2.9 Recurrent neural network (RNN)

Recurrent Neural Networks (RNNs) are neural networks designed to efficiently model sequence data by incorporating information from previous elements in the sequence [21]. Unlike traditional neural networks, RNNs have a recurrent structure that allows them to persist information over time, similar to how humans build upon past thoughts. This "memory" enables RNNs to influence current inputs and outputs based on previous inputs [29]. RNN contains a hidden layer that stores and recalls information over time, making them effective for tasks involving sequential data. for each timestep t , the activation

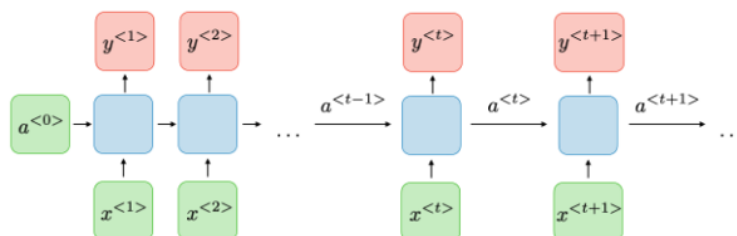


Figure 2.8: Architecture of a traditional RNN

$a^{<t>}$ and the output $y^{<t>}$ are expressed as follows:

$$a^{<t>} = g_1(W_{aa}a^{<t-1>} + W_{ax}x^{<t>} + b_a) \quad (2.8)$$

$$y^{<t>} = g_2(W_{ya}a^{<t>} + b_y) \quad (2.9)$$

Where $W_{ax}, W_{aa}, W_{ya}, b_a, b_y$ are coefficients that are shared temporally and g_1, g_2 are the activation functions.

2.9.1 Types of Recurrent Neural Networks

Sequential data cannot be effectively processed by traditional neural networks, as they have independent input and output layers that do not allow for information to be retained between sequential inputs[29], [30]. Recurrent Neural Networks (RNNs) were developed to address this limitation by incorporating an internal memory that can store information from previous outputs. There are four commonly used types of RNNs:

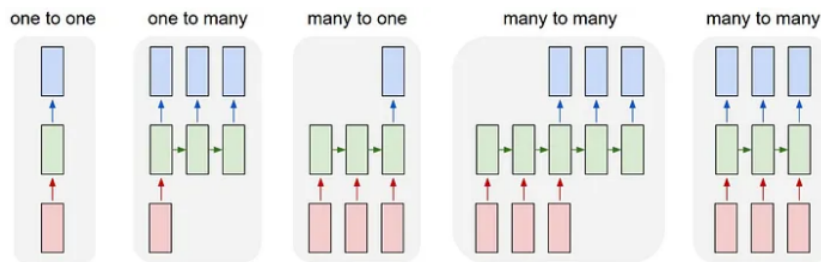


Figure 2.9: Summary of the main RNN models[30]

2.9.1.1 One-to-One RNN

The One-to-One recurrent neural network (RNN) is a simple type of RNN architecture that takes a single input and produces a single output [29]. Unlike other RNN architectures, the One-to-One RNN has fixed input and output sizes and behaves like a standard neural network. One common use case of the One-to-One RNN architecture is in image classification tasks.

2.9.1.2 One-to-Many

The One-to-Many architecture is a recurrent neural network (RNN) model that produces multiple outputs from a single input [29]. This model is designed to handle fixed input sizes and generate a sequence of data outputs. It has a range of applications in fields such as Music Generation and Image Captioning, where it is used to generate music or captions from a given input image.

2.9.1.3 Many-to-One

A Many-to-One RNN is a type of recurrent neural network architecture that takes a sequence of inputs and produces a single output by learning relevant features through a series of hidden layers [29]. This model is commonly used in various natural language processing tasks such as Sentiment Analysis.

2.9.1.4 Many-to-Many

The Many-to-Many architecture is a recurrent neural network (RNN) model that generates a sequence of output data from a corresponding sequence of input units [29]. It can be further divided into two subcategories:

Many-to-Many (equal size) where the input and output layers of the RNN have the same number of units.

Many-to-Many (unequal size) where the input and output sequences have different numbers of units. This type of RNN is often used in tasks such as machine translation.

2.9.2 Advantages and limitations of Recurrent Neural Networks (RNNs)

RNNs have several advantages such as the ability to process inputs of any length, remembering information throughout time, sharing weights across dependent hidden layers. It is an effective solution for pixel neighborhood prediction when combined with Convolutional Neural Networks. However, RNNs also have some drawbacks, such as slow computation due to their recurrent nature, difficulty in training, problems with processing long sequences using certain activation functions, and issues with exploding and vanishing gradients. Additionally, RNNs cannot be stacked into very deep models and struggle to keep track of long-term dependencies [29][31].

2.9.3 Long Short-Term Memory (LSTM)

Long Short-Term Memory (LSTM) networks are a type of recurrent neural network specifically designed to overcome the vanishing gradient problem. The vanishing gradient problem refers to the issue where the gradient term diminishes exponentially during back-propagation, hindering the learning of long period dependencies and leading to sub-optimal performance. LSTMs tackle this problem by incorporating specialized memory cells and gating mechanisms that selectively retain or forget information over multiple time steps. By effectively mitigating the vanishing gradient problem, LSTMs excel at capturing and utilizing long-term dependencies in sequential data. They achieve this through the use of three types of gates (Figure 2.10), which play a crucial role in selectively remembering or forgetting information [29]. These gates are:

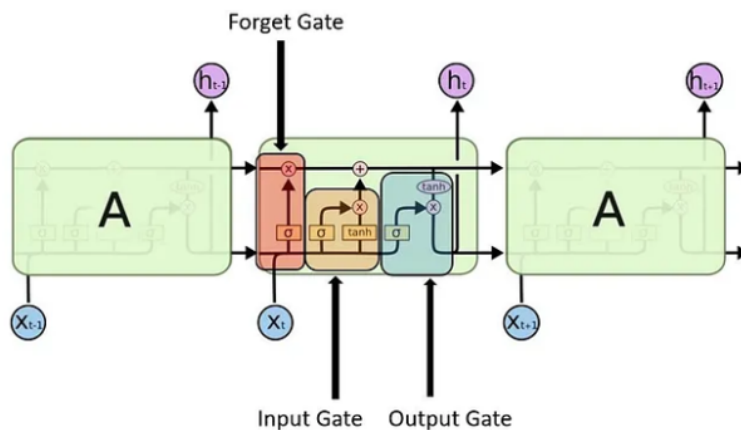


Figure 2.10: LSTM gates [30]

The Input gate is responsible for determining which values from the input will be used to modify the memory in an LSTM cell. The gate uses a sigmoid function to decide which values to let through, with a range of 0 to 1. Additionally, a tanh function is used to assign weights to the values that are passed through the gate, determining their level of importance on a scale of -1 to 1.

$$i_t = \sigma(W_i[h_{t-1}, x_t] + b_i) \quad \text{and} \quad (2.10)$$

$$\tilde{C}_t = \tanh(W_C[h_{t-1}, x_t] + b_C) \quad (2.11)$$

Where σ is the sigmoid function, $h_{(t-1)}$ is the output at time instant t-1, x_t is the current input at time instant t, b bias, and w the weights, \tilde{C}_t Candidate value.

The Forget gate in an LSTM cell is responsible for determining which information should be discarded or forgotten from the memory cell. This is done using a sigmoid function that looks at the previous hidden state (h_{t-1}) and the current input (x_t), and outputs a number between 0 (discard this) and 1 (keep this) for each value in the cell state (C_{t-1}) [30]. The forget gate allows the LSTM network to selectively retain or forget information from the memory cell as needed, improving its ability to learn and remember patterns in sequential data [30].

$$f_t = \sigma(W_f[h_{t-1}, x_t] + b_f) \quad (2.12)$$

The Output gate in an LSTM cell is responsible for computing the output of the current cell state, based on both the current input and the information stored in memory. Like the input gate, the output one uses the same activation function (sigmoid and tanh). The output of the sigmoid function is then multiplied by the output of the tanh function, producing the final output of the LSTM cell. This process enables the network to selectively output relevant information from the memory cell, improving its ability to process sequential data.

$$o_t = \sigma(W_o[h_{t-1}, x_t] + b_o) \quad \text{and} \quad (2.13)$$

$$h_t = o_t \cdot \tanh(C_t) \quad (2.14)$$

Where h_t is the output and C_t internal state.

2.10 Multimodal Deep Learning

Multimodal deep learning involves training deep learning models on diverse data types, such as images, text, and audio, to improve performance. This approach can combine popular architectures like CNNs and RNNs to leverage different types of information. For example, in a hybrid architecture, a CNN can be used to process visual input and extracts high-level features and a RNN to handle sequential data and models temporal dependencies.

By integrating both modalities, multimodal deep learning enables a more comprehensive understanding of data, capturing complementary information and improving accuracy, generalization, and robustness. This approach is applicable in various domains, including image analysis, speech recognition, and natural language processing. Figure 2.11 shows an example of these applications.

Multimodal deep learning provides a powerful framework for effectively modeling and

comprehending multimodal data by leveraging spatial and temporal correlations [32], [33].

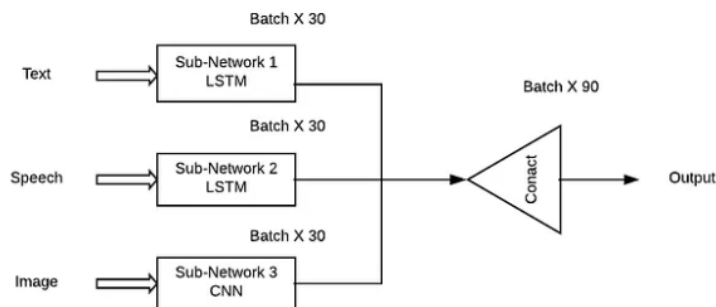


Figure 2.11: Example of Multimodal deep learning where different types of NN are used to extract features

2.11 Evaluation metrics for performance

It is important to evaluate the performance of the trained model to determine its generalization capabilities on unseen data. By employing various performance evaluation metrics, it is possible to enhance the model’s predictive accuracy before utilizing it for making predictions on new data.

Confusion matrix

A confusion matrix is a tabular representation that summarizes the performance of a classification model by displaying the counts of true positive (TP), true negative (TN), false positive (FP), and false negative (FN) predictions (figure 2.12). It is an important tool for evaluating the performance of a classification model and understanding its predictive capabilities.

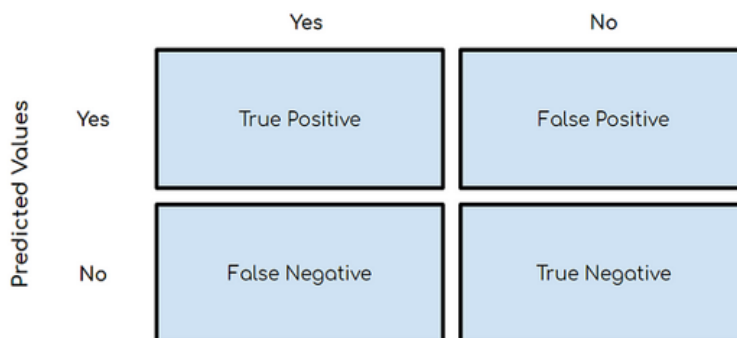


Figure 2.12: Structure of a 2x2 Confusion Matrix

In the confusion matrix:

True Positive (TP): The model correctly predicted the positive class.

True Negative (TN): The model correctly predicted the negative class.

False Positive (FP): The model incorrectly predicted the positive class when the actual class was negative (Type I error).

False Negative (FN): The model incorrectly predicted the negative class when the actual class was positive (Type II error).

By analyzing the values in the confusion matrix, several evaluation metrics can be derived, including [34]:

Accuracy: The overall proportion of correct predictions, calculated as:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \quad (2.15)$$

Precision: The proportion of true positive predictions out of the total positive predictions, calculated as:

$$Precision = \frac{TP}{TP + FP} \quad (2.16)$$

It measures the model's ability to correctly identify positive instances.

Recall (Sensitivity or True Positive Rate): The proportion of true positive predictions out of the total actual positive instances, calculated as:

$$Recall = \frac{TP}{TP + FN} \quad (2.17)$$

It measures the model's ability to capture positive instances.

Specificity (True Negative Rate): The proportion of true negative predictions out of the total actual negative instances, calculated as:

$$Specificity = \frac{TN}{TN + FP} \quad (2.18)$$

It measures the model's ability to correctly identify negative instances.

F1 score: The harmonic mean of precision and recall, calculated as:

$$F1 = 2 \cdot \frac{Precision \cdot Recall}{Precision + Recall} \quad (2.19)$$

The confusion matrix and the derived metrics help provide a comprehensive understanding of a model's performance, particularly in classification tasks, by capturing information about true positives, true negatives, and the types of errors made by the model.

2.12 Conclusion

In conclusion, this chapter has provided an overview of the theoretical concepts of deep learning relevant to this project. We explored the different architectures of artificial neural networks, specifically the Convolutional Neural Network (CNN) and the Recurrent Neural Network (RNN). These networks serve as the foundation for developing the hand gesture recognition algorithm.

While many of these concepts can be implemented using predefined functions, it is crucial to understand their underlying principles to design efficient algorithms. Deep learning offers various theories, methods, and approaches, each suited for specific applications and data characteristics.

The next chapter will focus on building the hand gesture recognition algorithm using the CNN and LSTM networks explained in this chapter. By leveraging these architectures, we aim to develop an accurate and robust algorithm for recognizing hand gestures.

Chapter 3

Hand gestures recognition using Single Modality (sEMG)

3.1 Introduction

Muscles signals are highly sensitive to different biological (e.g., muscle artifacts, fatigue, and attention) and environmental (e.g., noise) interferences, resulting in acquired sEMG signals with low Signal to Noise Ratio (SNR). A low SNR cannot be easily corrected by traditional methods due to the time required and the associated risk of information loss. Therefore, it is crucial to be able to derive informative representations (relevant features) from distorted signals.

Features extraction from medical signals relies heavily on human expertise in related fields. The conventional method is to analyze and extract information needed for diagnosis and providing the adequate solutions. However, manual analysis presents time issues due to the low availability of experts. Therefore, a method that can automatically extract the most representative features from the input data is highly desired. Different approaches using signal processing methods and machine learning algorithms are developed over the last decades. The most existing machine learning research has focused on static data and thus cannot accurately classify rapidly changing brain signals. For example, the classification accuracy of multi-class gesture recognition based on sEMG signals in the state-of-the-art is generally lower than 80% [35].

The potential of Deep learning methods and their capacity of learning meaningful representations from input data by constructing layered network. These methods can learn high-level and complex representations by combining a sequence of simpler representations, each of which extracts partial information related to the complex representation. In recent years, deep learning algorithms have been widely used and achieved great success in different fields.

The aim of this chapter is to develop a deep learning based approach for hand gesture recognition using sEMG signals. The following sections describe the proposed model, its implementation and the search of the best hyperparameters to achieve higher performances.

3.2 CNN-LSTM model for sEMG-based hand gesture recognition

Hand gestures and object manipulation take a time to be established. Therefore, the recognition of gestures process shall consider sequential characteristic of the sEMG signals. Thus, we proposed a hybrid CNN-LSTM architecture that combines recurrent layers composed of LSTM cells with convolution layers. Figure 3.1 shows this model. This architecture combines CNN layers for extracting spatial features from the input matrix of EMGs signals and LSTM layers for capturing temporal dependencies in the sequence of features. It concludes with fully connected layers to perform classification based on the extracted features.

3.2.1 Convolutional Layers

- The first layer is a Conv2D layer with 64 filters, a kernel size of (3, 3), and a hyperbolic tangent (tanh) activation function. It takes an input of shape (window-size, num-channels, 1), where the window width of the input is 12 channels and its height is 200 data point. The num-channels is the number of sEMG channels (12 channels are available in the used dataset).
- MaxPooling2D layer with a pool size of (2, 2) follows the first Conv2D layer, which reduces the spatial dimensions of the feature maps.
- The second Conv2D layer has 32 filters, a kernel size of (3, 3), and a tanh activation function.
- MaxPooling2D layer with a pool size of (2, 2) follows the second Conv2D layer.

Applied regularization methods: batch normalization is used after the first MaxPooling layer in order to normalize the activations of the previous layer and improving the training speed and performance. A dropout of 40% is applied after the second convolution layer. It is a well-known method to avoid overfitting.

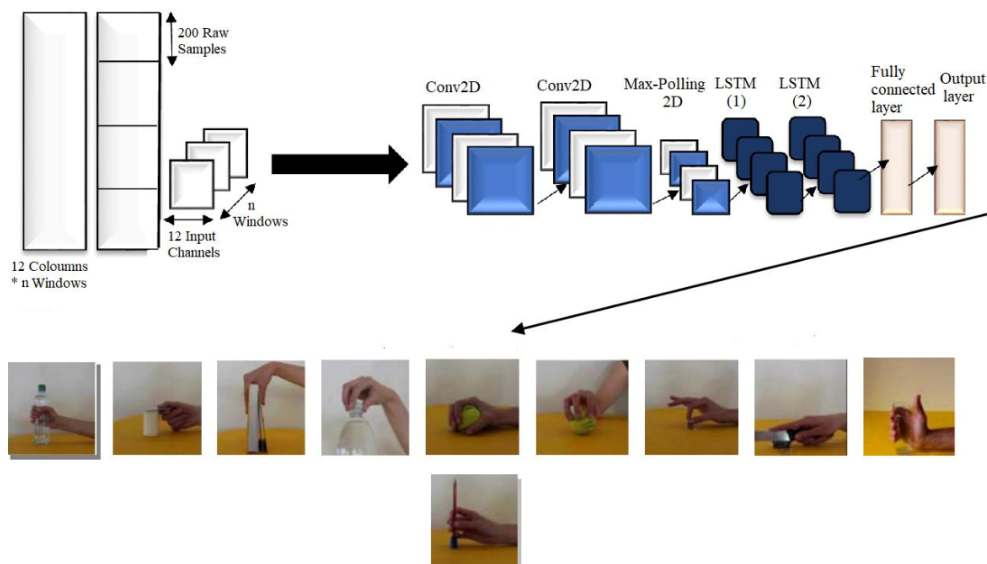


Figure 3.1: A Hybrid Model Approach for Hand Gesture Recognition using sEMG Signals

3.2.2 LSTM Layers

The second part of this model consists of LSTM layers for the extraction of temporal features from the sequential data. It is composed of two LSTM layers.

- The first LSTM layer has 128 units and is set to return sequences
- The second LSTM layer has 64 units and also returns sequences.

3.2.3 Regularization methods

Dropout layer with a rate of 0.4 is applied to the output of the first and second LSTM layers. After the first layer, the BatchNormalization is also applied.

3.2.4 Fully Connected Layers:

The function of this part is to classify the inputs on classes (Hand gestures). It is composed of several fully connected layers, each one of them having a different number of neurons and a tanh as an activation function. The output layer is composed of ten cells for considering the ten hand gesture recognition considered in the dataset (chapter 1), a softmax is used as an activation function.

The problem of hand gesture recognition is a classification problem, the loss function to be minimized is cross-entropy.

The implementation and training of our model were carried out in a specific hardware and software environment, as described in A .

3.2.5 Hyperparameter Tuning

The section highlights the importance of investigating hyperparameters to improve the performance and flexibility of a hybrid model. The objective is to enhance the accuracy of analyzing sEMG data by employing a combination of Convolutional Neural Networks (CNNs) and Long Short-Term Memory (LSTM) networks. The study conducts experiments, analyzes the outcomes to assess the hybrid model’s effectiveness. Table 3.1 presents the hyperparameters that were modified during the research.

Activation Function	Optimizers and Learning Rate	Batch Size
Tanh (Tangente Hyperbolique)	SGD (Stochastic Gradient Descent) with different learning rates	Tested different batch sizes during training
	Adam with different learning rates	

Table 3.1: Different hyperparameters

3.2.6 Configuration 1

Figure 3.2 illustrates the accuracy and loss curves depicting the results obtained for the training and validation data using the parameters from this configuration (B).

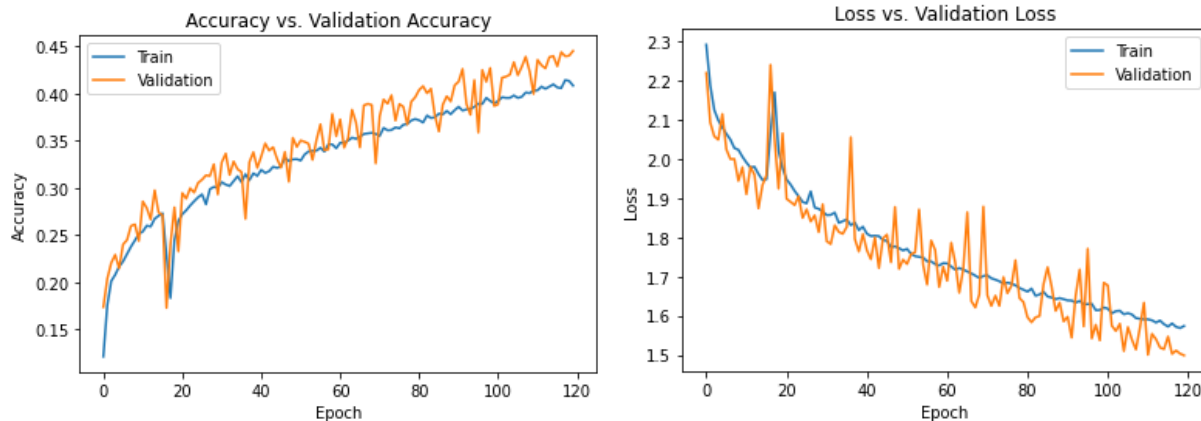


Figure 3.2: Accuracy and Loss Curves of the obtained results for the training and validation data(Configuration 1)

The obtained results revealed a poor accuracy of 40% for the initial training. Several factors may have contributed to this performance. Firstly, it is possible that the hybrid CNN-LSTM model architecture might have been unable to capture the information present in the sEMG data. This suggests the need for searching other hyperparameters that work better by running additional tests with other combinations of hyperparameters.

More experimentation is required to increase the hybrid CNN-LSTM model's accuracy and we start by changing the optimizer and learning rate.

3.2.7 Configuration 2

The network training is repeated using the new model and optimizer (B). The obtained results of the test using the Adam optimizer and a reduced learning rate of 0.002 demonstrated a significant improvement in the model's performance (Figure 3.3). The achieved accuracy of 80% marked a substantial increase compared to the previous test's accuracy of 40%, on the same number of epochs.

A comparison of the SGD and Adam optimizers' performances produced insightful results.

The Adam optimizer's adaptive learning rate allowed for efficient adjustment of the model's parameters and enabled faster convergence. In addition, it turned out that lowering the learning rate to 0.002 was a wise change. The model's weights could be adjusted more precisely thanks to the decreased learning rate. The model could better converge to an optimal solution and prevent overshooting or becoming stuck in sub optimal regions by reducing the step size during weight updates.

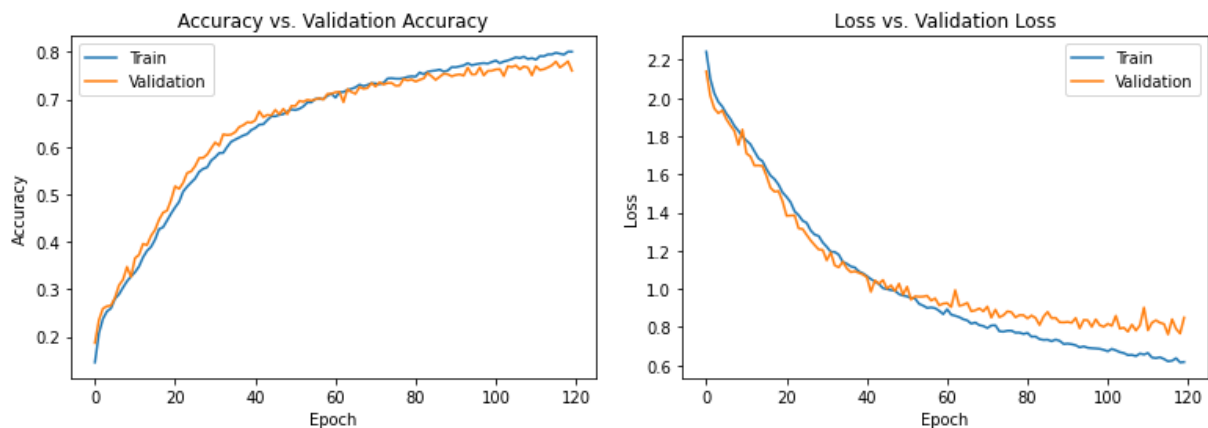


Figure 3.3: Accuracy and Loss Curves of the obtained results for the training and validation data (Configuration 2)

3.2.8 Configuration 3

Utilizing the characteristics of this configuration(B), the training of the model leads to its regeneration. This process reveals valuable insights regarding the interplay between learning rate and various factors, such as model convergence, stability, and the potential for achieving higher accuracies. The accuracy and loss curves are visually depicted in Figure 3.4.

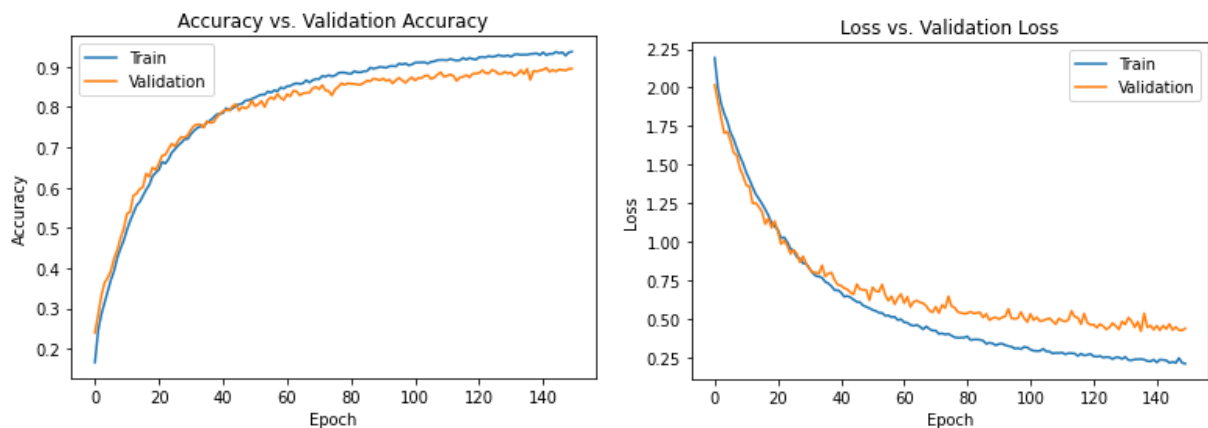


Figure 3.4: Accuracy and Loss Curves of the obtained results for the training and validation data (Configuration 3)

Furthermore, the achieved accuracy of 94% with a learning rate of 0.001 at the 120 epoch compared to the previous accuracies on the same number of epochs surpassed those obtained in previous tests. This suggested that the model’s ability to capture complex relationships within the sEMG data improved as the learning rate decreased.

Additionally, the impact of the number of epochs is assessed by comparing accuracies across different epoch settings while keeping the learning rate constant. Thus, we can see that the model needs more training epochs to converge to the optimal solution, but on this part, we only focused on the impact of lower learning rates.

3.3 Performances evaluation of the final single modality CNN-LSTM model

The performance of the developed model is conducted in detail for both training and validation data and test data.

3.3.1 Performances evaluation using training data

The final model is evaluated, initially, using examples representing the ten considered hand gestures. The confusion matrix of the obtained results (figure 3.6) shows a very small confusion rate between different classes and a very acceptable performance of recognition (between 92% and 98%) without overfitting.

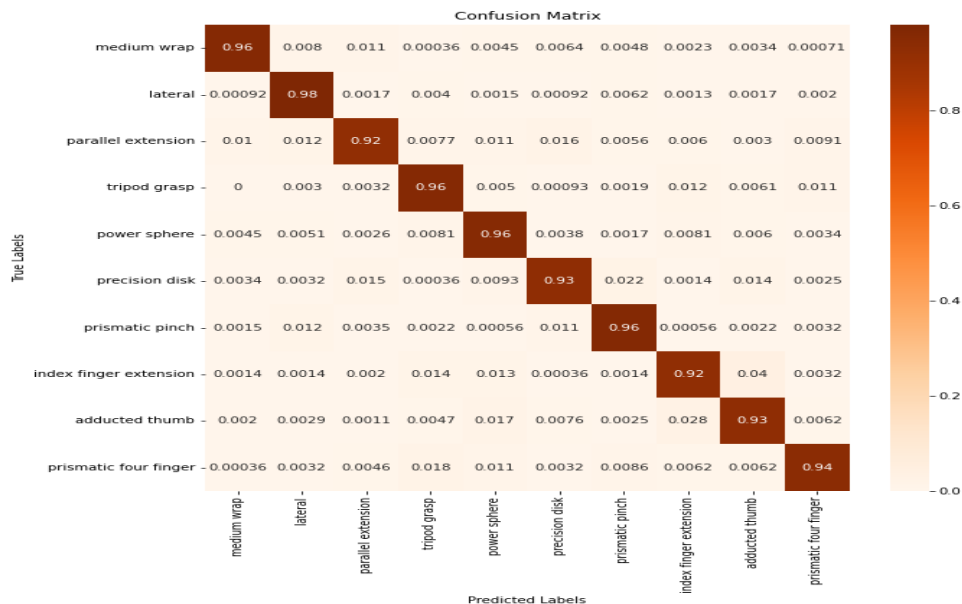


Figure 3.5: Evaluation of the Hybrid Model Performance: Confusion Matrix for training data

The confusion matrix shows that the classification mistakes are fairly evenly spread across all classes, indicating that the constructed CNN-LSTM model has performed well for the ten considered hand gestures.

The Pie plot presented in figure 3.6 shows the distribution of the examples over the ten classes and also the ratio of the predicted classes. The used data for the model evaluation are correctly distributed (10% approximately for each class).

For best visualization and interpretation of the sensibility of the developed model to each hand gesture, we present a Bar diagram in figure 3.7. This diagram shows, for each class, the difference between really available class (True) and the predicted one by the developed model, that the difference between them is small.

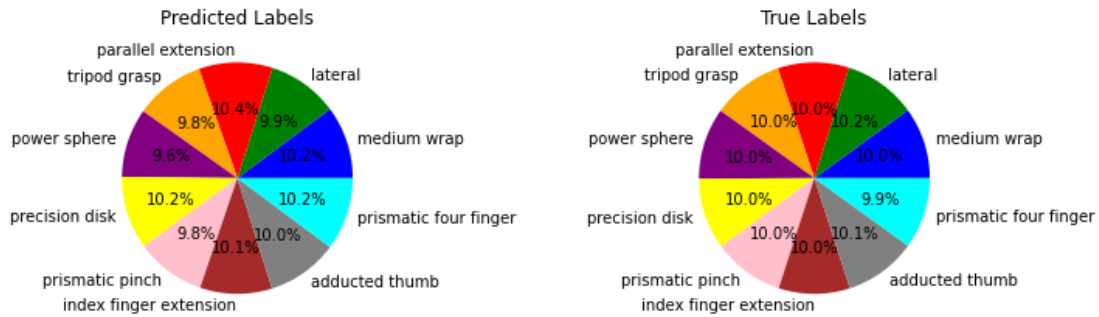


Figure 3.6: Predicted and True Labels in the training data model

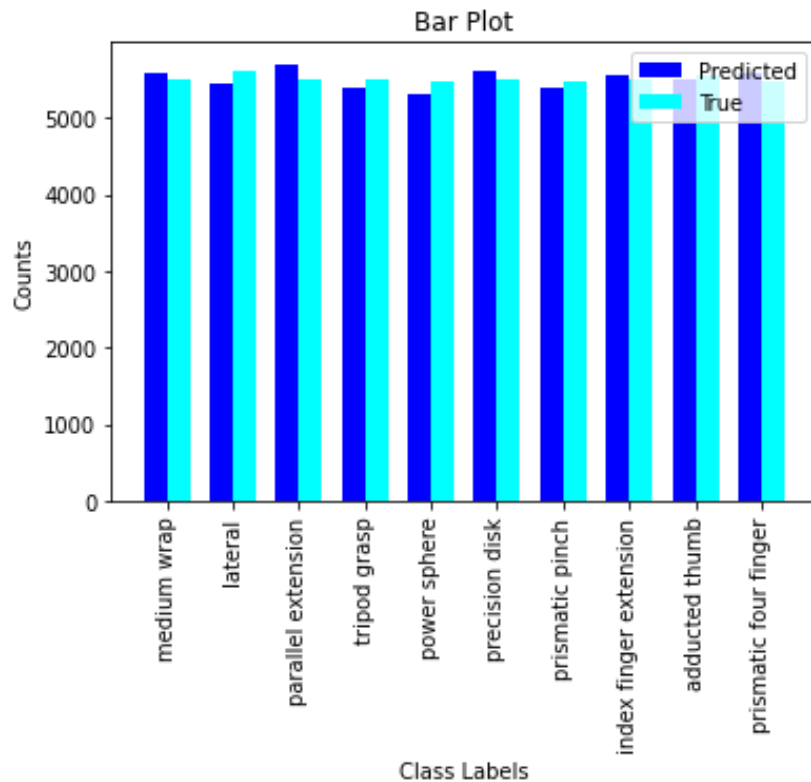


Figure 3.7: Comparison of Actual and Predicted Label Frequencies for Ten Classes in the Training Data Model

For quantitative evaluation of the model for the ten hand gestures, the measured performances are resumed in table 3.4.

3.3.2 Performances evaluation using test data

After performing a train-test split on our dataset, where we allocated 10% of the data to the test set, we evaluate the trained model’s performance on unseen test data to assess its ability to make predictions on new instances. In this subsection, the performances are presented in the same order as in the previous paragraph.

Classes	Precision	Recall	F1 score	Support
medium wrap	0.96	0.98	0.97	5502
lateral	0.98	0.95	0.97	5620
parallel extension	0.92	0.96	0.94	5494
tripod grasp	0.96	0.94	0.95	5494
power sphere	0.96	0.93	0.94	5491
precision disk	0.93	0.95	0.94	5494
prismatic pinch	0.93	0.95	0.94	5480
index finger extension	0.93	0.93	0.93	5500
adducted thumb	0.93	0.92	0.93	5559
prismatic four fingers	0.94	0.96	0.95	5480

Table 3.2: The individual performances of each class

The confusion matrix is presented in figure 3.8. The test accuracy, for each class, is slightly lower than the training accuracy. The diagonal elements varied from 84% to 94%. These values can still be considered reasonably good compared to the presented ones in figure 3.6. It indicates that the model is capable of making accurate predictions on new and unseen data.

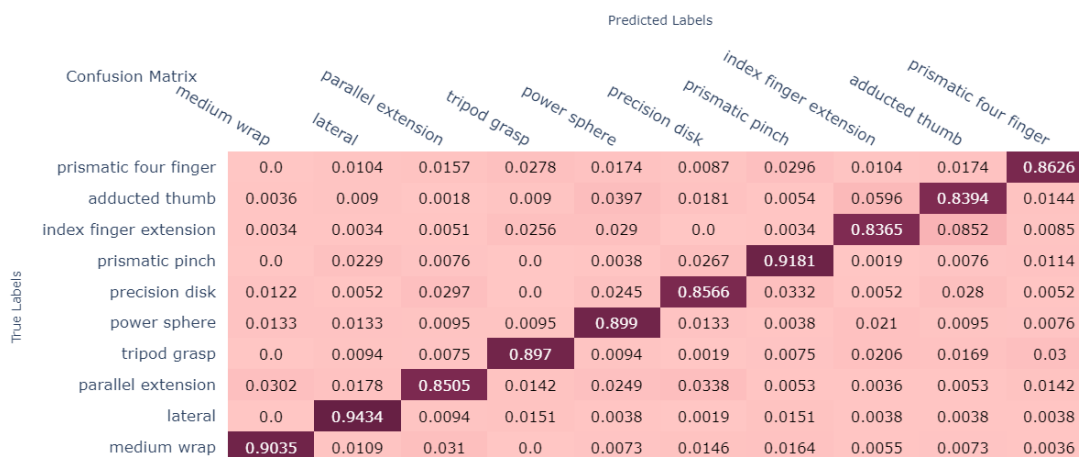


Figure 3.8: Evaluation of the Hybrid Model Performance: confusion matrix of test data

The Pie plot and Bar diagram presented in figure 3.9 and 3.10 show the distribution of the examples used for the model test for the ten classes and also the ratio of the predicted classes. The used data for the model evaluation are correctly distributed (10% approximately for each class). The obtained results are very close to those presented in the previous paragraph using training data.

The global performances of the developed model are summarized in table 3.3 and the individual ones for each class are presented in table 3.4.

Based on the results we have achieved, it appears that our classification model has per-

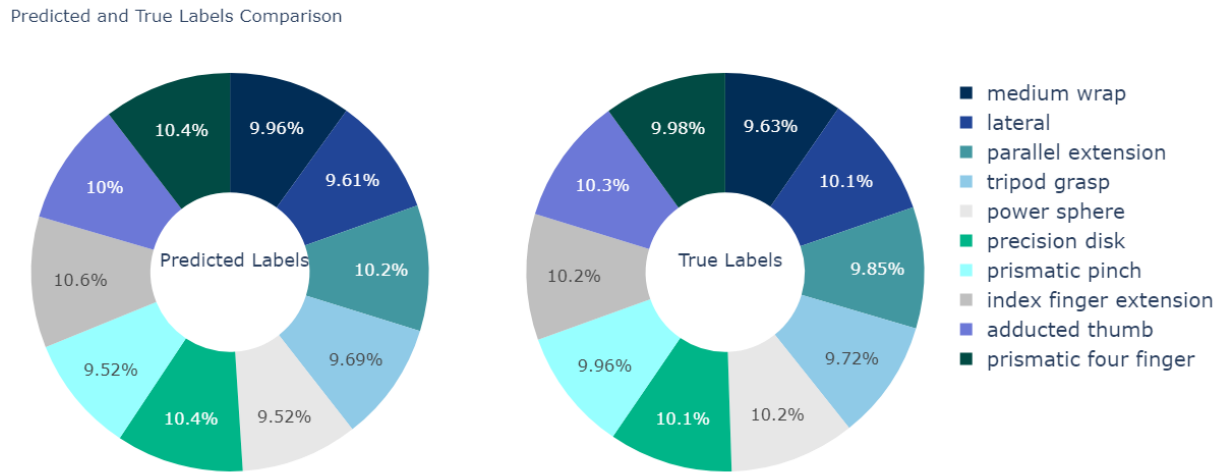


Figure 3.9: Predicted and True Labels Comparison

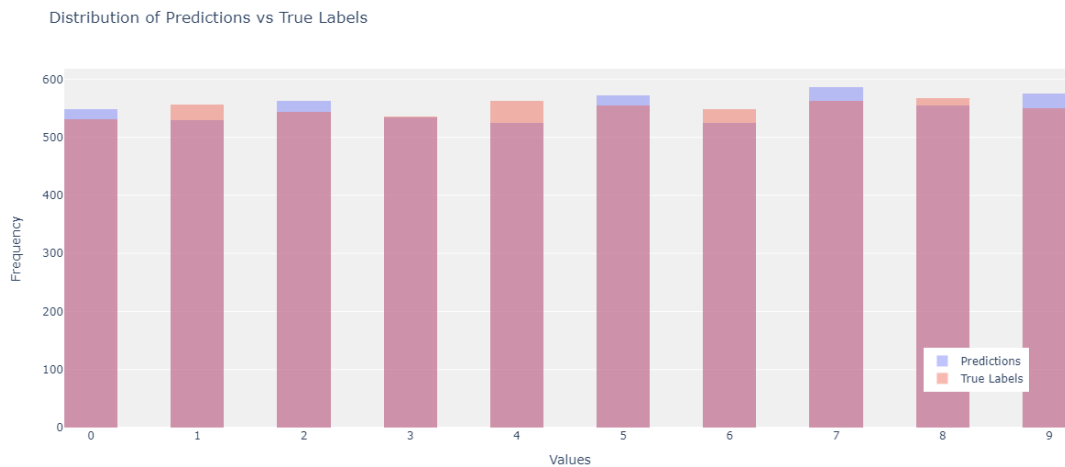


Figure 3.10: Comparison of Actual and Predicted Label Frequencies for Ten Classes in the Test Data Model

Accuracy	Precision	F1 score	Recall
88%	88.04%	87.95%	87.95%

Table 3.3: The global performances of the developed model

formed reasonably well with an 88% test accuracy, precision, recall, and F1 score. These metrics indicate that our model is able to accurately classify instances into their respective classes with a relatively high level of overall accuracy.

An accuracy of 88% implies that our model correctly predicted the class labels for 88% of the observations in the test dataset. This suggests that the model is effective in distinguishing between the different grasp classes.

Classes	Precision	Recall	F1 score	Support
medium wrap	0.9	0.93	0.92	531
lateral	0.94	0.9	0.92	556
parallel extension	0.85	0.88	0.87	543
tripod grasp	0.9	0.89	0.9	536
power sphere	0.9	0.84	0.87	562
precision disk	0.86	0.88	0.87	555
prismatic pinch	0.92	0.88	0.9	549
index finger extension	0.84	0.87	0.85	563
adducted thumb	0.84	0.82	0.83	568
prismatic four fingers	0.86	0.9	0.88	550

Table 3.4: The individual performances of each class

The recall, also known as sensitivity or true positive rate, is a measure of how well our model captures instances of a particular class. An 87.5% recall implies that our model successfully identified 87.5% of the instances belonging to each grasp class, indicating a good ability to detect positive cases.

The F1 score combines precision and recall into a single metric and is useful for evaluating the overall performance of the model. With an F1 score of 88%, our model demonstrates a balanced trade-off between precision and recall, indicating a robust classification capability.

Overall, our model’s performance is quite promising, achieving high accuracy, precision, recall, and F1 score.

3.4 Effect of manipulated object in recognition performance

It is crucial to take object variation into account when working with sEMG signal data gathered from participants performing various grasps with different objects. It is important to study and analyze the impact of object variation on the model’s performance and its ability to generalize.

During the learning process, we are considered for each hand grasp gesture one type of object. Additionally, we evaluated the trained model’s performances when other objects are manipulated.

We perform a test for 6 subjects named from S10 to S15:

Our experiment yielded results indicating poor performance of the model when attempting to predict outcomes for unseen data from the same individual using a different object. This finding highlights the considerable difficulty in applying the model’s predictions to new objects within the same grasp. It suggests that the sEMG signal patterns linked

Subject	Accuracy (%)
S10	19
S11	26
S12	33
S13	32
S14	26
S15	19

Table 3.5: Subject’s accurecies while manipulating different objects

to distinct objects may vary significantly, posing challenges for the model to accurately identify the same grasp when manipulating different objects.

Moreover, the model’s performance deteriorates with increased variability, such as when the subject transitions from a seated to a standing position or from a static to a dynamic task while maintaining the same grasp. The model needs to learn to adapt to diverse contexts and situations, and the introduction of these variations in the training set presents additional challenges [12].

To provide an overview of the objects and associated grasps within the dataset, Table 3.8 illustrates the various objects used in both the training and test sets.

Table 3.6: Overview of the objects and the grasps of the dataset

Grasp	Static Tasks	Train Objects	Test Objects
Wedium wrap 	take the can	bottle 	can 
Lateral 	take the zip of pencilcase	key 	pencilcase 
Parallel extension 	take the book	plate 	book 
Tripod grasp 	take the bottle	bottle 	mug 
Ower sphere 	take the ball	ball 	key 
Precision disk 	take the light bulb	bulb 	ball 
Prismatic pinch 	take the key	key 	can 
Index finger extension 	take the knife	remote 	knife 
Adducted thumb 	take the wrench	wrench 	screwdriver 
Prismatic four finger 	take the fork	knife 	fork 

3.5 Conclusion

A hybrid model composed of CNN and LSTM cells is proposed, trained and evaluated for the recognition of ten hand gestures grasps using 12 sEMG channels. The obtained results are very acceptable (accuracy of 88%).

In the following chapter, we present an approach for data fusion using deep learning to improve the quality of single modality hand grasp gestures recognition.

However, an important question to be considered in the development of this kind of application: What is the required accuracy of pattern recognition to be considered reliable by the user?

The classification accuracy does not consider the dynamic effects of real-time control[35]. Thus, there are various difficulties and constraints to take into account when employing surface Electromyography (sEMG) data for classification in the context of a myoelectric prosthesis hand such as:

- **Electrode Shift:** The electrodes will probably settle in a slightly different position in relation to the underlying muscle each time a user puts on a prosthesis. The loading and placement of the limb during usage may potentially cause the electrodes to move[35].
- **Signal Variability:** sEMG signals can vary significantly between subjects and even within the same subject due to factors such as electrode placement, and muscle fatigue, This variability can make it challenging to accurately classify the intended hand movements [35].
- **Adaptability:** The user's muscle patterns and strength may change over time, which may have an effect on how well the classification system performs. To account for these changes, the prosthesis may need to be periodically calibrated or retrained, which can be time-consuming [35].

To address these challenges, it may be necessary to consider some strategies such as :

- **Improved Data Collection:** Collect sEMG signals from more participants and record a variety of gesture and object-handling scenarios to improve the quality and diversity of the training data. This would help increase the model's ability to generalize across different users.
- **Signal Processing:** Develop advanced signal processing techniques that can effectively handle the variability and noise present in sEMG signals.
- **Transfer learning:** using transfer learning techniques to initialize or improve the myoelectric prosthesis models by using pre-trained models from huge datasets. It is also possible to think to extend the developed model in this project by transfer learning for adapting this model to various other situations. This can enable faster adaptation to new users.

- Multimodal Sensing: Investigate the incorporation of other sensor modalities to provide complementary information about hand position using for example inertial measurement units (IMUs) and visual information.

Chapter 4

Hand gestures recognition using Multimodality

4.1 Introduction

The difficulty of interpreting and employing sEMG as a single modality was emphasized as one of the constraints of myoelectric prostheses in the previous chapter. The anatomical characteristics of the user, the placement and contact of the electrodes with the skin, variations in arm positions, and fatigue all have a significant impact on the signals [10]. Various studies have suggested overcoming the controllability problems by integrating or replacing sEMG with modalities that are more stable, to improve the model accuracy on one hand and on the other hand to overcome some of sEMG limitations.

We present in this chapter a multimodal integration techniques that combines different signals such as sEMG, accelerometer signals and the eye motion (gaze).

Eye tracking (Gaze) is the practice of measuring eye movements or positions to determine where people are focusing during an activity or in other words where people are looking before grasping an object [36].

Also, the orientation and motion of the arm can be mapped using accelerometry in the inertial reference frame. Since accelerometers are inexpensive and some electrodes already have a three-axial accelerometer incorporated, obtaining this information is affordable. Numerous studies have demonstrated that the use of sEMG and accelerometry together performs better than using just the first modality alone to recognize hand gestures. [10].

In this chapter, we present different models that combine these modalities (sEMG & Gaze, sEMG & Accelerometer, sEMG, Gaze & Accelerometer). These models share some properties such as the global form of architectures and parameters and are trained and tested using same data for performances evaluation.

4.2 Fusion of sEMG with Gaze

The previous studies provided solid evidence that the grasp of an object is often preceded by a visual fixation on the same object [10]. Thus this visual information could be used to improve the model performance and why not improve the control of a prosthetic device. Gaze tracking data, combined with surface Electromyography (sEMG) data, can serve as a supplementary modality to improve the accuracy of hand gesture recognition by deep learning models. This combination can improve the performance of recognition. Even still, gaze tracking data may not be sufficient on their own to provide precise control or accurate predictions.

Figure 4.1 illustrates the developed architecture for combining the two modalities. It is composed of two branches one for each modalities and some full connected layers to concatenate the outputs of the previous branches and classified the signal according to the considered classes. Ten hand gestures grasp have been considered in this work.

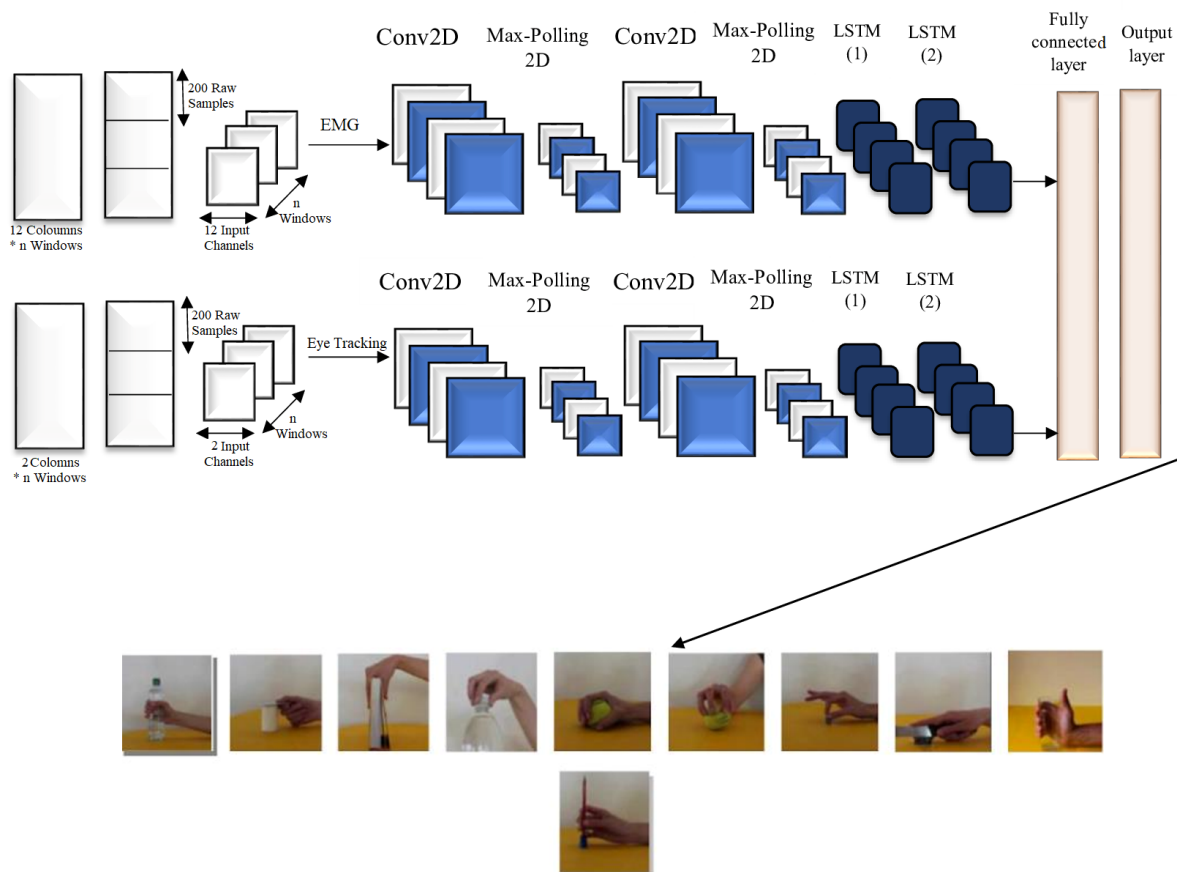


Figure 4.1: Integrating Gaze and sEMG Signals in Hand Gesture Recognition: A Hybrid Model Approach

4.2.1 Model training and performances evaluation

To select the model hyperparameters, many experiment are carried out and the best ones are presented chosen. The selected ones are used for all the upcoming experiments presented in this part. Table 4.1 summarises these parameters:

Activation	Optimizer	Learning rate	Batch size	Epochs
Tanh	Adam	0.001	256	150

Table 4.1: Fusion of sEMG with Gaze: Model parameters

The figure 4.2 shows the accuracy and Loss evolution during the training phase. The training and validation accuracies and losses curves show a good training without overfitting. The comparison between the performance of this model with the one presented in chapter 3 (using only sEMG modality) is in table 4.2. The training accuracy is improved by 1.7%, going from 94% to 95.7% when gaze data was combined with sEMG data. The model was able to use the combined insights from the two modalities by adding gaze information as an additional modality to EMG, which improved performance during training. This improvement highlights the complementary nature of gaze and sEMG data and highlights how effective they are when used together to increase model accuracy.

When comparing the test accuracies, the model integrating both gaze and sEMG data obtained a test accuracy of 90% while the model using only sEMG data only obtained an accuracy of 88%. This indicates that the integration of gaze data alongside sEMG data has slightly improved the model's performance on unseen test data. The additional data from the gaze data appears to be helpful in improving the model's capacity to generate precise predictions on untested examples.

It's crucial to remember that the increase in test accuracy (a 2% increase) is quite small. It suggests that although the incorporation of gaze data has provided certain benefits, its contribution in this particular situation may have some limitations.

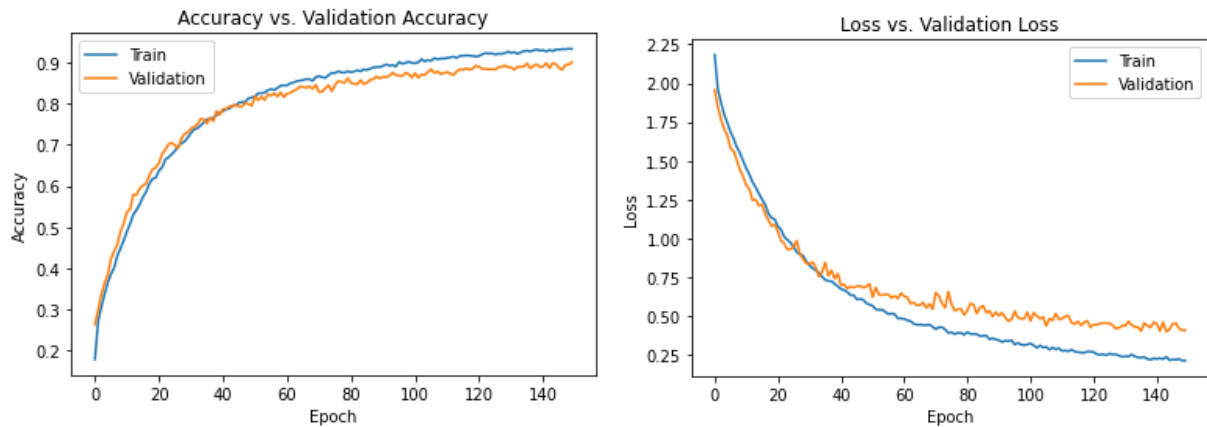


Figure 4.2: Accuracy and Loss Curves for the Hybrid Model of sEMG and Gaze

Accuracy	sEMG	sEMG & Gaze
Training	94%	95.7%
Test	88%	90%

Table 4.2: Performances comparison of the sEMG and the fusion of sEMG and Gaz models

To analyse the model performance for all the considered hand gestures grasps, we present the confusion matrix in figure 4.3. This matrix is obtained for test data. It is clear that the model can distinct between different classes and the confusion ratio is very small. The higher confusion rate is of 6%.

Confusion Matrix		Predicted Labels									
		parallel extension	lateral	tripod grasp	power sphere	precision disk	prismatic pinch	index finger extension	adducted thumb	prismatic four finger	medium wrap
True Labels	prismatic four finger	0.0	0.0086	0.0138	0.031	0.0172	0.0017	0.0344	0.0017	0.0172	0.8744
	adducted thumb	0.0091	0.0036	0.0091	0.0018	0.0236	0.0054	0.0036	0.0399	0.8947	0.0091
	index finger extension	0.0067	0.0033	0.0	0.0184	0.0467	0.0	0.0	0.8514	0.0618	0.0117
	prismatic pinch	0.0098	0.0217	0.002	0.002	0.002	0.0118	0.9429	0.002	0.0	0.0059
	precision disk	0.0215	0.0099	0.0397	0.0017	0.0083	0.8446	0.0463	0.0033	0.0165	0.0083
	power sphere	0.0019	0.0038	0.0076	0.0114	0.9242	0.0227	0.0	0.0133	0.0114	0.0038
	tripod grasp	0.0	0.0111	0.0111	0.8961	0.0186	0.0019	0.0056	0.0223	0.0093	0.0241
	parallel extension	0.0235	0.0126	0.8773	0.0144	0.0036	0.0325	0.0072	0.0108	0.0126	0.0054
	lateral	0.0	0.9552	0.0037	0.0093	0.0056	0.0	0.0187	0.0019	0.0	0.0056
	medium wrap	0.957	0.0059	0.0137	0.0039	0.0039	0.0059	0.0059	0.002	0.0	0.002

Figure 4.3: Evaluation of the Hybrid Model Performance: Confusion Matrix for sEMG and Gaze

4.3 Fusion of sEMG with Accelerometer measurements

Unlike using gaze data, Accelerometer data can be more informative and be a valuable input source for training and enhancing accuracy. Accelerometer data frequently offers more direct and accurate information on physical movements compared to gaze tracking data, making it a reliable modality for controlling prosthetic devices or improving model performance.

The developed model in this case is also a two branch model. The first one (sEMG madality) is the same developed in chapter 3 and used for the combination sEMG and Gaze. The second branch has accelerometer data as input and has the same architecture (as for Gaze) Figure 4.4. The difference is only in the number of inputs.

The figure 4.5 shows the accuracy and Loss evolution during the training phase. The training and validation accuracies and losses curves show a good training without overfitting.

The combination of sEMG and accelerometer data has produced the highest accuracy compared to the two previous scenarios (sEMG only, sEMG & Gaze) as was previously anticipated.

The comparison between the performance of this model with the other models is in table 4.3. The model’s training and testing accuracy were both 97% when the accelerometer and sEMG data were combined. It exceeds the accuracy of the previously developed models. This shows also that the model’s performance on both the training and testing datasets has significantly improved as a result of the incorporation of accelerometer data

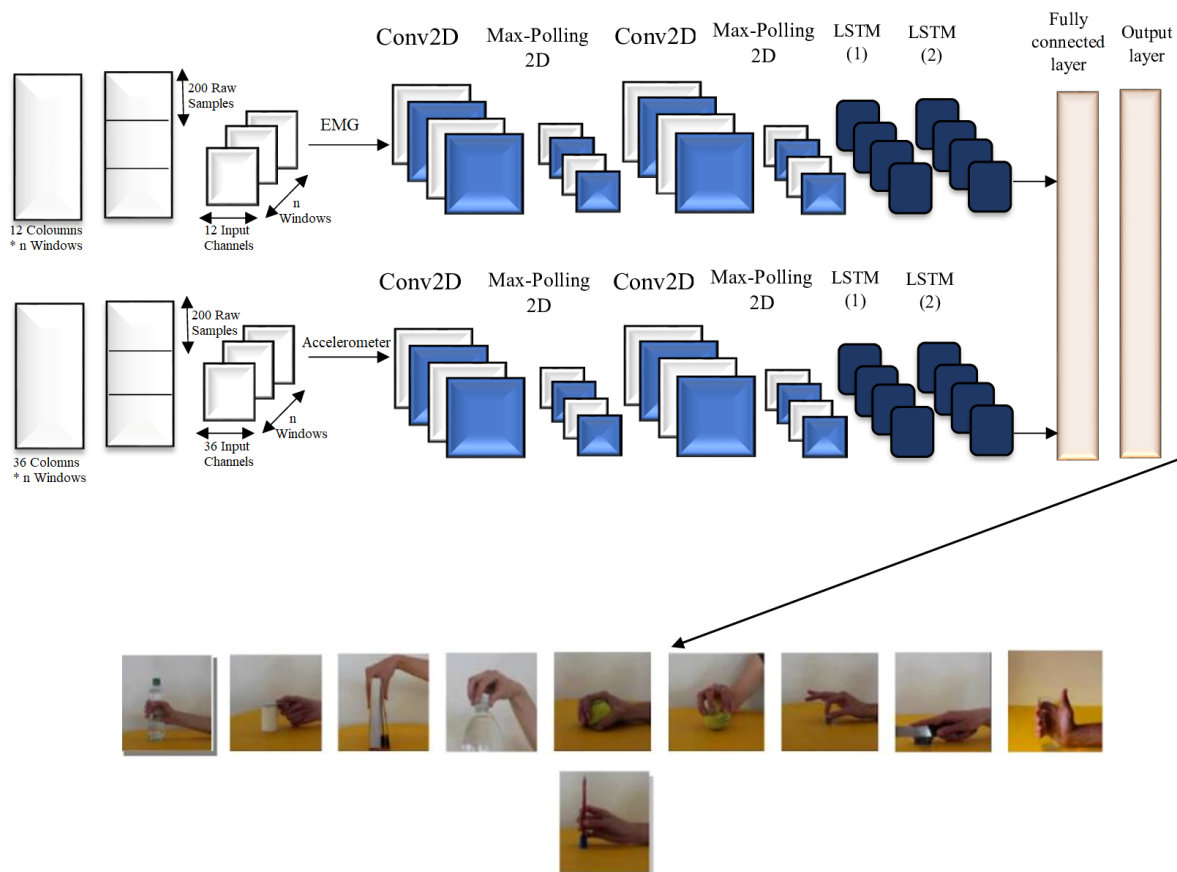


Figure 4.4: Integrating Accelerometer, and sEMG Signals in Hand Gesture Recognition: A Hybrid Model Approach

The integration of accelerometer data most likely give the model new information and features that enhanced its capacity to recognize patterns and make precise predictions. The model’s overall accuracy have been considerably improved by the integration of accelerometer data.

It’s important to note that increasing accuracy by combining different modalities shows the potential advantages of utilizing many data sources for a more accurate prediction of hande gestures recognition.

Accuracy	sEMG	sEMG & Gaze	sEMG & Accelerometer
Training	94%	95.7%	97%
Test	88%	90%	97%

Table 4.3: Performances comparison of the sEMG, sEMG & Gaze and sEMG & Accelerometers models

To analyse the model performance for all the considered hand gestures grasps, figure 4.6 illustrates the obtained confusion matrix using test data. Like the previous model, it is also clear here that this model can distinct between different classes and the confusion ratio is very small. The higher confusion rate is less than 3%.

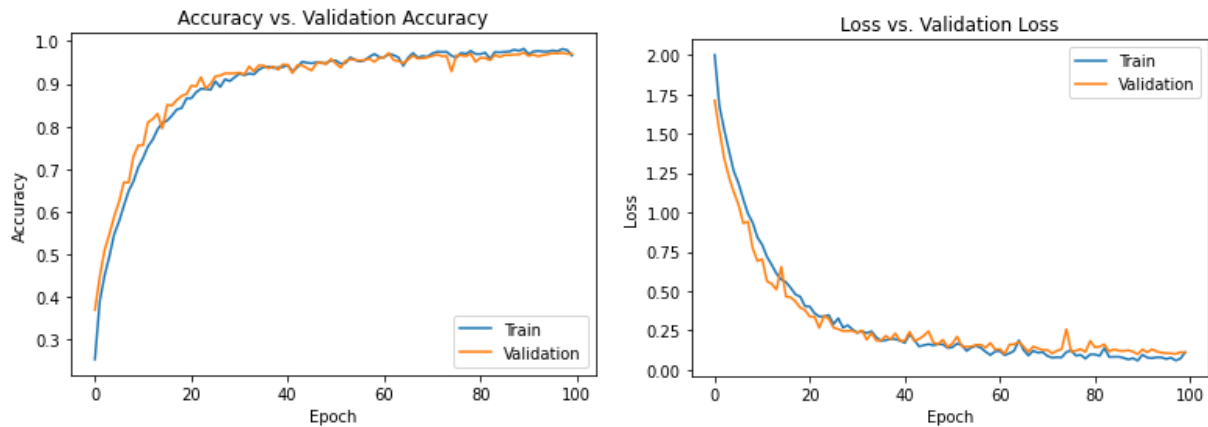


Figure 4.5: Accuracy and Loss Curves for the Hybrid Model of sEMG & Accelerometer

Confusion Matrix		Predicted Labels									
		medium wrap	lateral	parallel extension	tripod grasp	power sphere	precision disk	prismatic pinch	index finger extension	adducted thumb	prismatic four finger
True Labels	prismatic four finger	0.0	0.0	0.0	0.0	0.0	0.0	0.0055	0.0055	0.0037	0.9853
	adducted thumb	0.0018	0.009	0.0	0.0	0.0018	0.0	0.0072	0.0072	0.9657	0.0072
	index finger extension	0.0	0.0017	0.0017	0.0035	0.0052	0.0	0.0035	0.9529	0.0297	0.0017
	prismatic pinch	0.0037	0.0128	0.0018	0.0	0.0	0.0037	0.967	0.0055	0.0018	0.0037
	precision disk	0.0	0.0	0.0	0.0089	0.0036	0.9715	0.0125	0.0036	0.0	0.0
	power sphere	0.0018	0.0018	0.0036	0.0053	0.9698	0.0	0.0	0.0036	0.0107	0.0036
	tripod grasp	0.0	0.0	0.0019	0.9776	0.0112	0.0056	0.0	0.0019	0.0019	0.0
	parallel extension	0.0109	0.0018	0.962	0.0018	0.0054	0.0054	0.0018	0.0	0.0072	0.0036
	lateral	0.0018	0.9853	0.0018	0.0018	0.0018	0.0	0.0073	0.0	0.0	0.0
	medium wrap	0.9665	0.0074	0.0112	0.0	0.0	0.0037	0.0	0.0037	0.0037	0.0037

Figure 4.6: Evaluation of the Hybrid Model Performance: Confusion Matrix for sEMG and Accelerometer

4.4 Fusion of sEMG with Gaze and Accelerometer modalities

The obtained results when combining sEMG with Gaze and with Accelerometer measurement are very attractive. We present in this section a global system that combines the three modalities. Its architecture is illustrated in figure 4.7.

The performance and capabilities of machine learning models can be further improved by integrating gaze and accelerometer with sEMG data. By combining these multiple modalities, we may take advantage of the unique information of each type and develop a more comprehensive understanding of the task.

We may evaluate the effect of incorporating various types of modalities on the model’s performance by training the model with the integration of three modalities (EMG, gaze, and accelerometer) while using the same hyperparameters as the previous models.

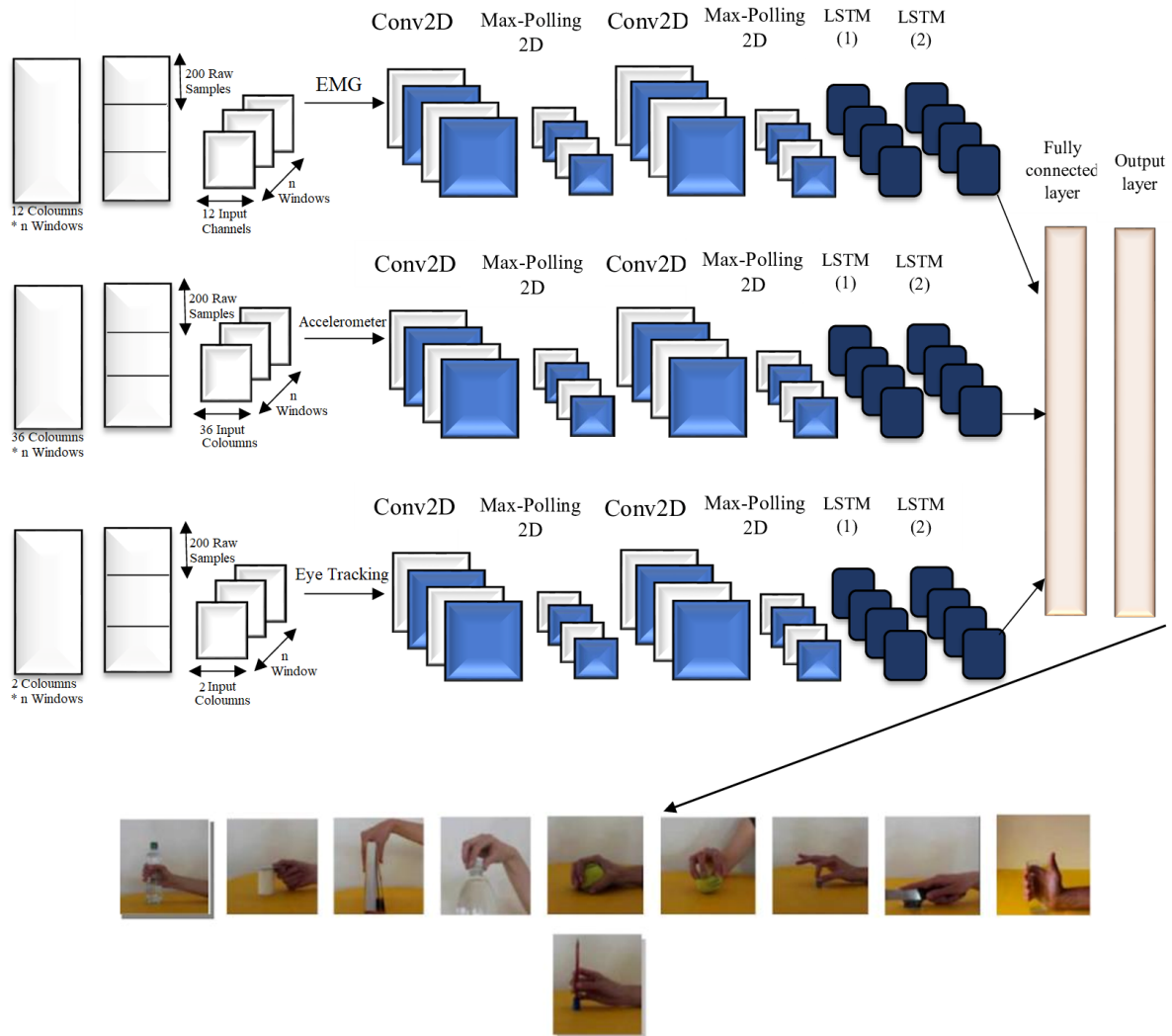


Figure 4.7: A Hybrid Model Approach for Integrating Accelerometer, Gaze, and sEMG Data in Hand Gesture Recognition

Figure 4.8 shows the accuracy and Loss evolution during the training phase and table 4.4 summarises the model performances for both training and test data. The comparison between the performances of the four developed models enables us to understand the additional advantages provided by including multiple modalities in the training process.

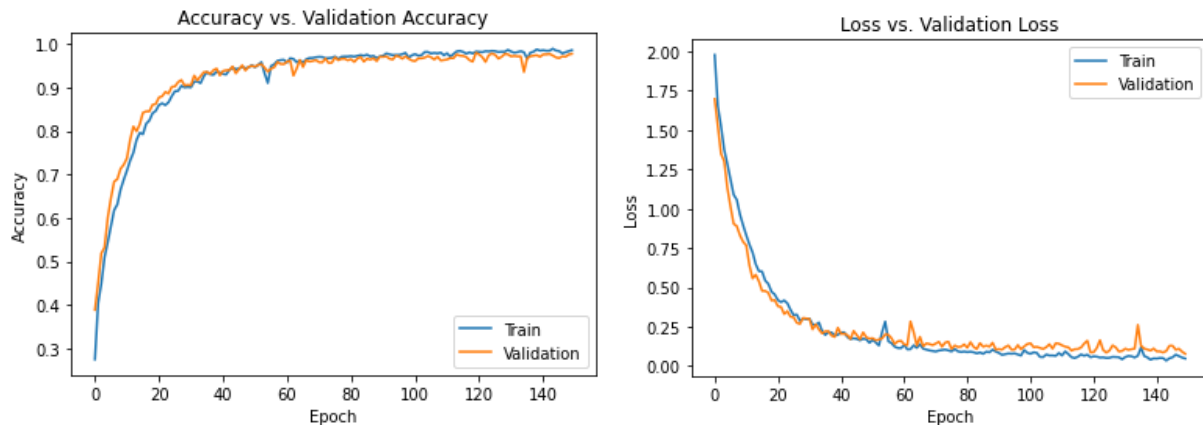


Figure 4.8: Accuracy and Loss Curves for the Hybrid Model of sEMG & Acceleromete & Gaze

Comparing all of the previous scenarios, the model that combines three modalities (sEMG, gaze, and accelerometer) achieves the highest accuracy, coming in at 98%. This notable increase in accuracy emphasizes the benefit of combining multiple data types in the training process. The test and training accuracies are similar, indicating that the model generalizes well and effectively uses the combined information from the three modalities to produce accurate predictions.

Accuracy	sEMG	sEMG & Gaze	sEMG & Accelerometer	sEMG & Accelerometer & Gaze
Training	94%	95.7%	97%	98%
Test	88%	90%	97%	98%

Table 4.4: Models performances

Figure 4.9 shows the confusion matrix obtained with this model. It is clear that the model can easily separate between different classes.

The Pie plot of predicted and true labels (figure 4.10) provides a visual representation of the distribution of different classes or categories within the predicted and true labels.

The degree to which the predictions of the model agree with the actual labels can be determined by comparing the pie charts of predicted and true labels side by side.

The model’s predictions are consistent with the actual labels if the pie charts resemble one another or strongly overlap. The model may be biased or inaccurate in predicting certain class labels, on the other hand, if there are obvious differences between the pie plots. These, the quality of model’s prediction is very acceptable.

Confusion Matrix		Predicted Labels									
		medium wrap	lateral	parallel extension	tripod grasp	power sphere	precision disk	prismatic pinch	index finger extension	adducted thumb	prismatic four finger
True Labels	prismatic four finger	0.0018	0.0	0.0036	0.0018	0.0036	0.0	0.0018	0.0	0.0018	0.9854
	adducted thumb	0.0035	0.0052	0.0017	0.0017	0.0017	0.0	0.0017	0.007	0.9738	0.0035
	index finger extension	0.0	0.0	0.0	0.0	0.0036	0.0	0.0018	0.9822	0.0107	0.0018
	prismatic pinch	0.0	0.0018	0.0072	0.0	0.0054	0.0054	0.9747	0.0018	0.0018	0.0018
	precision disk	0.0	0.0	0.0018	0.0089	0.0	0.9839	0.0036	0.0018	0.0	0.0
	power sphere	0.0	0.0	0.0036	0.0	0.9804	0.0	0.0036	0.0089	0.0	0.0036
	tripod grasp	0.0	0.0	0.0019	0.9887	0.0056	0.0019	0.0	0.0019	0.0	0.0
	parallel extension	0.0019	0.0019	0.9886	0.0038	0.0019	0.0	0.0	0.0	0.0	0.0019
	lateral	0.0	0.9892	0.0018	0.0018	0.0	0.0	0.0036	0.0	0.0018	0.0018
	medium wrap	0.9688	0.0037	0.0202	0.0	0.0018	0.0	0.0	0.0	0.0037	0.0018

Figure 4.9: Evaluation of the Hybrid Model Performance: Confusion Matrix for sEMG & Acceleromete and Gaze

Predicted and True Labels Comparison

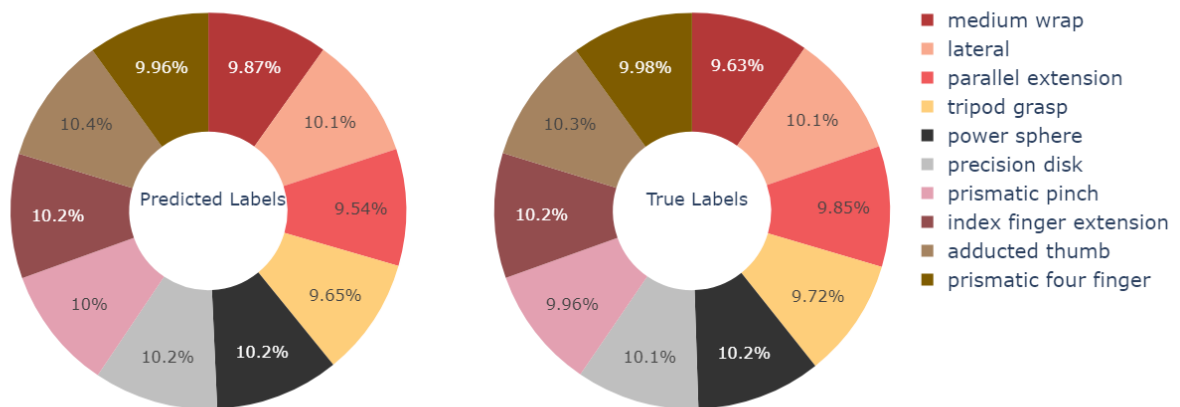


Figure 4.10: Predicted and True Labels Comparison

The global performances of this model considering all the ten classes are resumed in the following table. For a more illustration of these results, a bar diagram is plotted in figure 4.11. In this diagram, a categorical variable’s counts or frequencies of various class labels, such as actual labels or predicted labels, are represented. The closed bar heights for both actual and predicted labels for the ten classes shows that the model has similar performances for all the classes.

Classes	Precision	Recall	F1-score	Support
Medium wrap	0.97	0.99	0.98	531
Lateral	0.99	0.99	0.99	556
Parallel extension	0.99	0.96	0.97	543
Tripod grasp	0.99	0.98	0.99	536
Power sphere	0.98	0.98	0.98	562
Precision disk	0.98	0.99	0.98	555
Prismatic pinch	0.97	0.98	0.98	549
Index finger extension	0.98	0.98	0.98	563
Adducted thumb	0.97	0.98	0.98	568
Prismatic four fingers	0.99	0.98	0.98	550

Table 4.5: Test results for each class

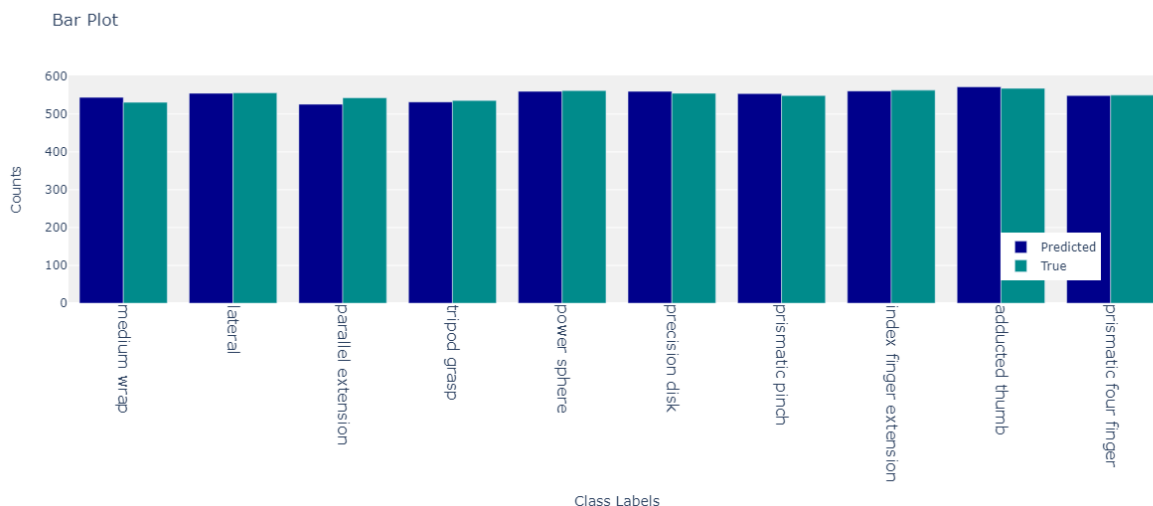


Figure 4.11: Comparison of Actual and Predicted Label Frequencies for Ten Classes in the Model

4.5 Test on Variability

In this section, we perform different experiments to study the model performance when training and test data were acquired in different experimental conditions listed in previous sections. This analysis aims to examine whether the sEMG or the accelerometer as well as the gaze data recorded when performing a specific grasp result depend on task-related factors, such as the grasped object.

To do this, two parameters are taken into consideration, each of them represents a different train-test split of the data.

4.5.1 Object-split

We included one of the three objects used for each grasp type in the training set and another object is considered in the test set and we look at the best results across all subjects. The aim is to understand how an object's characteristics influence our data and the model performances.

4.5.2 Trial-split

10% of our data is kept for the test set, while the remaining 90% is used for training. In this case, the level of variability in the train and test data is the same.

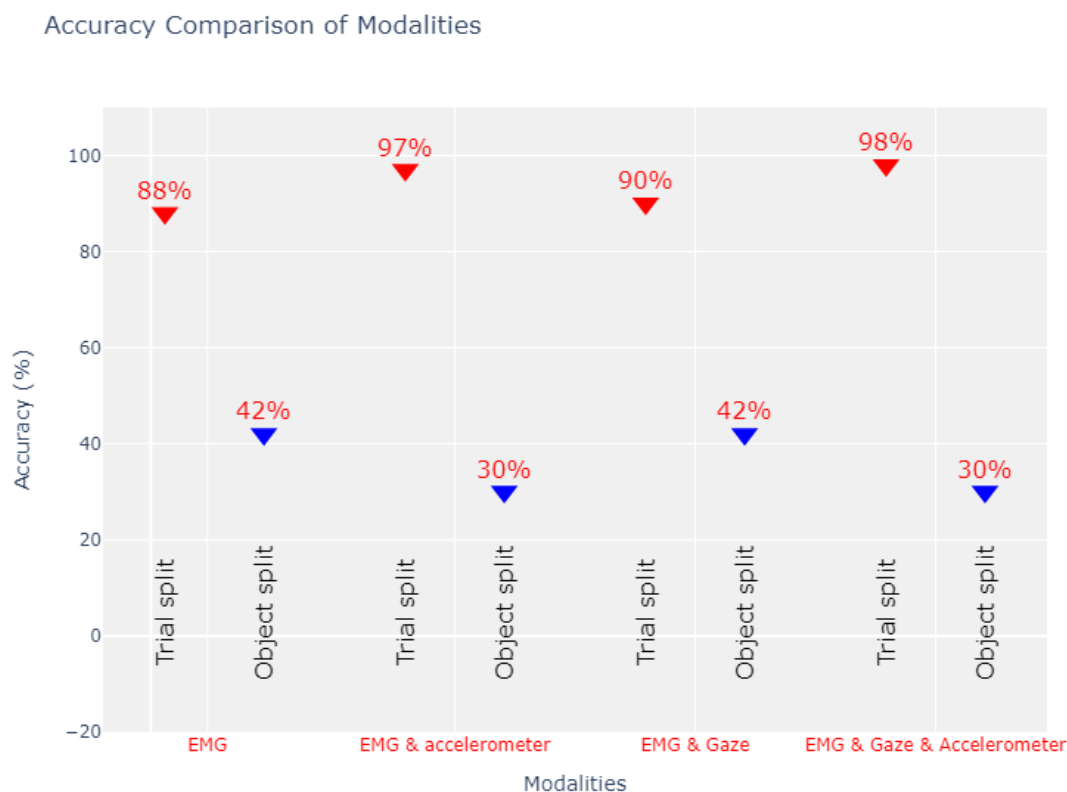


Figure 4.12: Accuracy Comparison of Modalities

As expected the model performs poorly on object-split, this problem turned around the challenge of variability in signals between subjects and within the same subject when manipulating different objects with the same grasp.

The variations in signals seen between various participants are referred to as subject variability. While performing the same grasp, each person may have different muscle activation patterns, eye movement characteristics, and accelerometer readings. This variability makes it difficult for a model trained on data from one subject to generalize well to other subjects, and that leads to resulting in poor performance. In addition, while manipulating several objects with the same grasp, there may be variances in signal patterns even

within the same person. Different objects have different physical characteristics, including different shapes, weights, and textures, which can have an impact on how individuals execute the grasp. These variances present the model with extra difficulties because it must take into account the differences caused by the items while still identifying the desired grab.

Even the integration of multiple modalities could not effectively overcome the limitations posed by the subject’s variability. The signals from different modalities might still exhibit significant variations between subjects while manipulating different objects, making it challenging for the model to generalize accurately.

Finding ways to reduce the impact of signal variability is necessary to solve this issue. This can involve collecting larger and more diverse datasets that cover a variety of subjects and objects. Additionally, to enhance the model’s generalization capabilities, methods like data augmentation, transfer learning, or fine-tuning might be used. Further study and the use of novel methods are required to improve the model’s capacity to handle these variances efficiently in order to solve this issue.

4.6 Models Training and performance evaluation considering the Amputees data

All the previously models are trained using examples of measurements obtained with intact persons. The aim of this section is to train the same final model with its optimized parameters using the dataset of amputee participants. This step enables us to evaluate the model’s performance on a different set of participants and compare it to the results obtained from the intact participant data.

The accuracy and loss curves obtained during the training of the model that used only sEMG data, sEMG and Gaze, and sEMG and Accelerometer are plotted in figures 4.13 4.14 4.15 respectively. The performances of these models using train and test data are summarised in table 4.6. The three models achieves a training accuracy slightly higher compared to the one obtained with intact participant’s data. It seems that the model achieves a higher testing accuracy when trained on the amputee data compared to the intact participant data. This observation suggests that the amputee participant’s data contains more informative patterns or features that contribute to the overall performance of the model.

	sEMG		sEMG & Gaze		sEMG & Accelerometer	
Accuracy	Intact	Amputees	Intact	Amputees	Intact	Amputees
Training	94%	95%	95%	96%	97%	98%
Test	88%	91%	90%	95%	97%	98%

Table 4.6: Accuracy Results

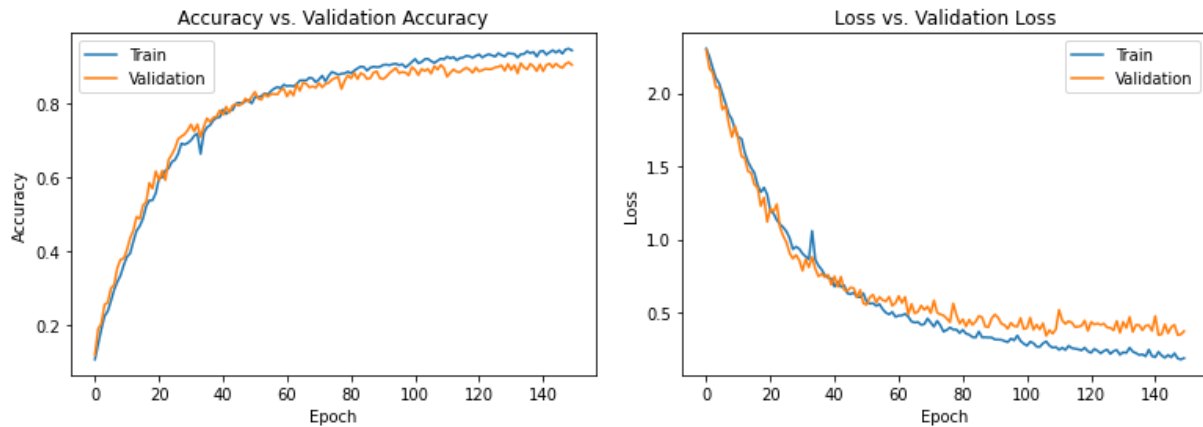


Figure 4.13: Accuracy and Loss Curves for sEMG-only Model

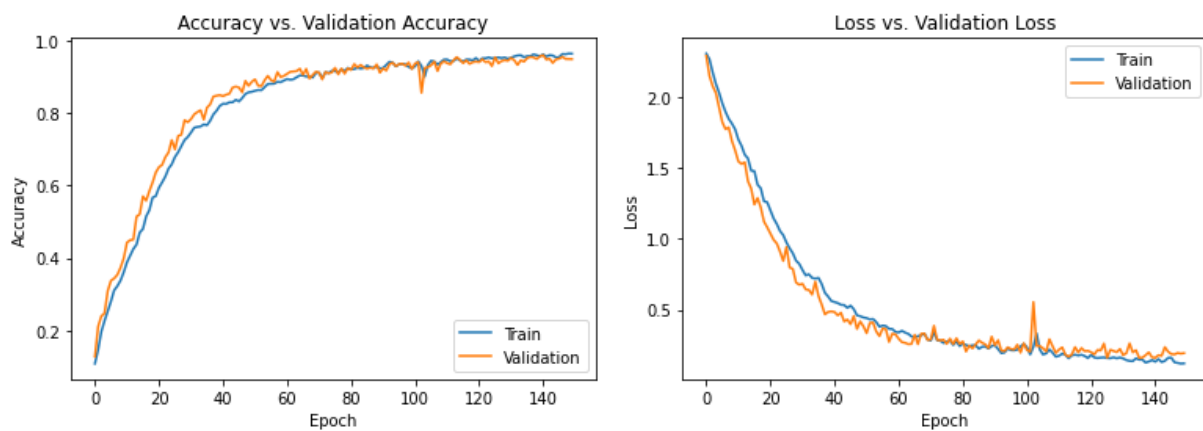


Figure 4.14: Accuracy and Loss Curves for sEMG & Gaze

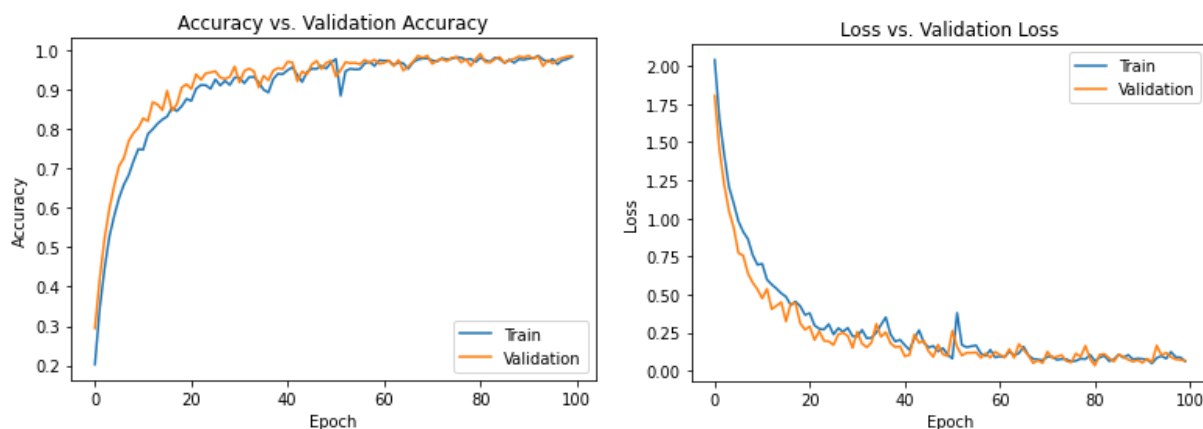


Figure 4.15: Accuracy and Loss Curves for sEMG & Accelerometer

4.7 Model’s Performances comparison with Related works

Figure 4.16 illustrates the performances obtained with different classifiers (LDA: KRLS mDWT, K-NN) published in [10] and that use the same dataset as in our work. These

performances are obtained considering only sEMG signals.

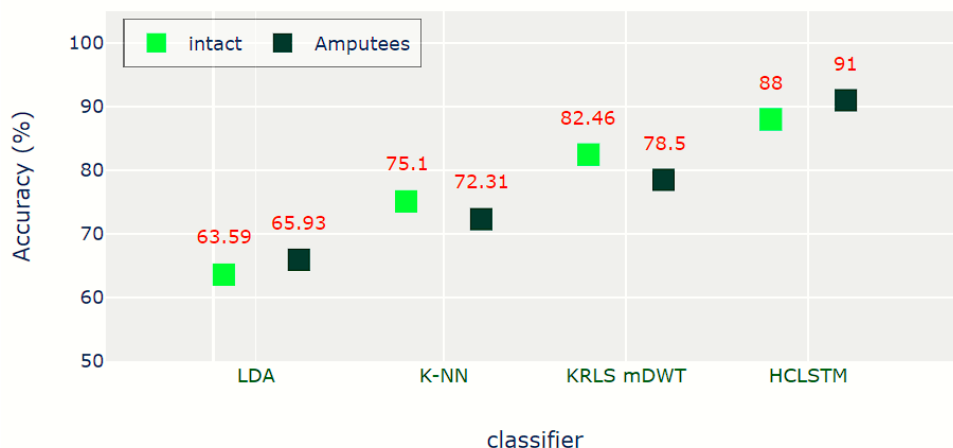


Figure 4.16: Classification accuracies for able-bodied and amputated subjects when predicting the grasp type with three different types of classifiers using sEMG signals [10] and our model

The results of classification accuracy, reported in Figure 4.14, show accuracy between 63% and 82% for both intact and amputated subjects, depending on the classification method (LDA, K-NN, or KRLS mDWT).[10]. Overall, the higher accuracy achieved by our HCLSTM classifier suggests its potential for effective modelling and prediction in the given domain, showing the advantages of leveraging deep learning techniques for sequential data analysis.

The figure 4.17 illustrates another comparison between our proposed model and the previously considered methods of classification [10] when the sEMG was combined with gaze data.

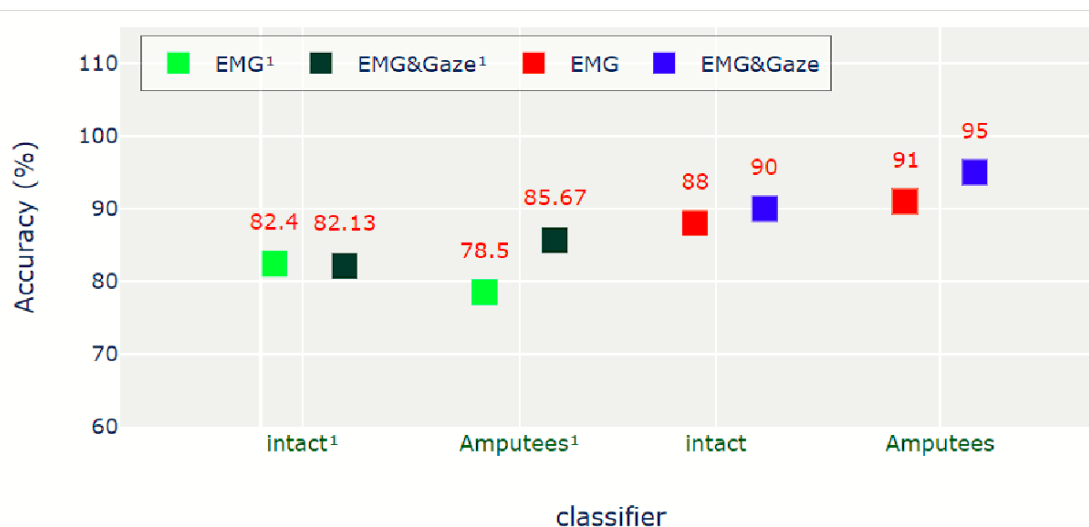


Figure 4.17: Classification accuracies for intact (intact¹) and amputated (Amputees¹) subjects when predicting the grasp type using only sEMG and while integrating also the visual information[35]

Figure 4.17 shows the classification accuracy for both intact and amputated subjects. The integration of vision with muscular information increases the performance by about 4% and 7% for intact subjects and amputees, respectively in comparison of the case of only sEMG signals [10].

Also in the case of using our proposed classifier, the performance of the model increases by about 2% and 4% for intact subjects and amputees when we integrate gaze information.

4.8 conclusion

This chapter was focused on the integration of gaze and accelerometer data alongside sEMG data to perform multimodal classification for predicting ten different grasp types. The study conducted a series of experiments, starting with the integration of sEMG with gaze data, which resulted in an accuracy of 95%. While the fusion of sEMG and Accelerometer can achieve an accuracy of 97%. Finally, by combining all three modalities (EMG, gaze, and accelerometer) into a multimodal, an impressive accuracy of 98% was obtained for both training and test data. These results highlight the effectiveness of incorporating multiple modalities for grasp classification.

Comparing our results to related work that used the same dataset, it would be essential to consider the specific methodologies, experimental setups, and evaluation metrics used in those studies. However, our approach shows promising results.

General Conclusion

In conclusion, this work has provided a comprehensive study of hand gesture recognition, covering various aspects related to signal analysis, deep learning architectures, classification models, and multimodal integration. The study emphasized the relevance of signals such as surface Electromyography (sEMG), accelerometers, and eye tracking in hand gesture recognition. Theoretical concepts of deep learning, including Convolutional Neural Networks (CNNs) and Recurrent Neural Networks (RNNs), were explored as the foundation for developing accurate and robust gesture recognition algorithms.

A hybrid model combining CNN and LSTM cells was proposed for recognizing ten hand gesture grasps using sEMG data as a single modality, achieving an acceptable accuracy of 88%. The study addressed the challenges and constraints associated with sEMG data classification, employing strategies such as improved data collection, signal processing, transfer learning, and multimodal sensing to overcome them.

The effectiveness of integrating gaze and accelerometer data with EMG data was demonstrated, resulting in a significant improvement in classification accuracy. By combining all three modalities, a promising accuracy of 98% was achieved for both training and test datasets.

Considering these achievements, there are several perspectives for future work in the field of hand gesture recognition for myoelectric prostheses. Exploration of advanced deep learning architectures, such as attention mechanisms, graph neural networks, or transformer-based architectures, could be conducted to improve the accuracy and robustness of classification algorithms. Enhancing the real-time applicability of gesture recognition systems is essential, and the development of efficient and lightweight models for real-time classification on prosthetic limbs would significantly enhance usability and practicality.

The development of personalized and adaptive models should also be pursued to improve accuracy and user experience. Techniques like transfer learning or personalized fine-tuning could be investigated to adapt pre-trained models to individual users. Additionally, integrating contextual information and semantic understanding could enhance gesture recognition systems by incorporating knowledge about user intentions, task context, or environmental factors.

To evaluate the usability, acceptance, and effectiveness of the developed systems in real-

world scenarios, conducting user studies and clinical trials is crucial. Gathering feedback from amputees and healthcare professionals can provide valuable insights to refine the technology and address user-specific needs.

By considering these perspectives, future research can contribute to advancing the field of hand gesture recognition for myoelectric prostheses, ultimately improving the quality of life and autonomy for individuals with limb amputation.

Appendices

Appendix A

Software and Hardware Tools

A.1 Software

A reliable software environment is necessary for the development of efficient deep learning models for time series analysis.

The software environment used to train deep learning models for time series analysis is covered in this section. The study made use of the TensorFlow framework and the flexible programming language Python, as well as crucial modules like pandas, numpy, matplotlib, and Keras. The work was done using Spyder as the Integrated Development Environment (IDE) in the Anaconda environment.

The software environment is built on the popular programming language Python. It is a popular option for data analysis and deep learning tasks due to its user-friendly syntax, substantial community support, and rich ecosystem of libraries. In this study, tools like pandas, numpy, and matplotlib were essential for manipulating data, performing calculations, and visualizing the results. Numpy supplied crucial capabilities for numerical computing and array operations, while Pandas supported effectively data pretreatment, cleaning, and transformation. Data exploration and pattern recognition were made easier with the help of Matplotlib's ability to create a variety of visuals.

This research used TensorFlow as the underlying deep learning framework, together with Keras, a high-level neural network library. By offering an accessible API and abstracting away low-level operations, Keras made the process of developing, testing, a deep-learning models simpler. The model training and deployment in production contexts were made easier by the scalable and effective framework TensorFlow. Its extensive toolkit enabled operations including distributed computing, model optimization, and data preparation.

The Anaconda environment, a complete data science platform, was used to perform this research. Package management, environment management, and other crucial capabilities provided by Anaconda allowed for the easy integration of libraries and quick project setup. The IDE of choice, Spyder, offered a user-friendly and potent environment for Python data analysis and scientific computing. With its features, which included code editing,

debugging, and variable exploration, model creation, and experimentation became more productive.

A.2 Hardware

This part explores the hardware configuration utilized for training machine learning models in time series analysis.

A.2.1 AMD Ryzen 5 3500U CPU

The research utilized an AMD Ryzen 5 3500U CPU as the central processing unit. This CPU is equipped with 4 cores and 8 threads, allowing for parallel execution of tasks. With a base clock speed of 2.9 GHz, the CPU provides a solid foundation for efficient data processing and model training in time series analysis.

A.2.2 AMD Radeon Vega 8 GPU

The system incorporated an integrated AMD Radeon Vega 8 GPU, which offers accelerated computations for machine learning tasks. The GPU provides additional processing power specifically designed for parallel operations, enabling faster training and inference of models. The inclusion of the AMD Radeon Vega 8 GPU enhances the overall computational capabilities of the system, particularly for tasks involving large-scale time series data.

A.2.3 Memory

The hardware configuration included DDR4 memory with a capacity of 8GB and a clock speed of 2400 MHz. This high-speed memory facilitated efficient data storage, retrieval, and processing during model training. The ample memory capacity allowed for the handling of large datasets and the storage of intermediate computations, contributing to improved training performance and reduced processing time.

Appendix B

Model’s configuration parameters

B.1 Configuration 1

For this, the first step made was to test the CNN-LSTM model with the parameters indicated in Table B.1.

Activation function	Tanh
Optimizer	SGD
Learning rate	0.2
Number of epochs	120
Batch size	256

Table B.1: Parameter Configuration 1 for Training a CNN-LSTM Model

B.2 Configuration 2

With a learning rate of 0.2, the SGD optimizer’s initial training test produced a low accuracy metric. This inspired the investigation of further optimization methods to perhaps enhance the performance of the model. Adam Optimizer was selected as a strong contender because of its adaptable learning rate and momentum. It becomes used with a decreased learning rate of 0.002. The goal was to compare the performance of SGD and Adam optimizers as well as look into the advantages of using a slower learning rate. All considered parameters in this case are in table B.2.

Activation function	Tanh
Optimizer	Adam
Learning rate	0.002
Number of epochs	120
Batch size	256

Table B.2: Parameter Configuration 2 for Training a CNN-LSTM Model

B.2.1 Configuration 3

In this part, we consider Adam optimizer with a reduced learning rate (0.001) compared to the used one in the previous test. The number of epochs is also increased (Table B.3).

Activation function	Tanh
Optimizer	Adam
Learning rate	0.001
Number of epochs	150
Batch size	256

Table B.3: Parameter Configuration 3 for Training a CNN-LSTM Model

Bibliography

- [1] S. Micera, J. Carpaneto, and S. Raspopovic, “Control of hand prostheses using peripheral information,” *IEEE Reviews in Biomedical Engineering*, vol. 3, pp. 48–68, 2010.
- [2] M. B. I. Reaz, M. S. Hussain, and F. Mohd-Yasin, “Techniques of EMG signal analysis: Detection, processing, classification and applications,” *Biological Procedures Online*, vol. 8, no. 1, pp. 11–35, Dec. 2006.
- [3] M. Atzori, A. Gijsberts, C. Castellini, B. Caputo, A.-G. M. Hager, S. Elsig, G. Giatsidis, F. Bassetto, and H. Müller, “Electromyography data for non-invasive naturally-controlled robotic hand prostheses,” *Scientific Data*, vol. 1, no. 1, Dec. 2014.
- [4] “Accelerometer Basics - SparkFun Learn — learn.sparkfun.com,”
- [5] S. Zhang, Y. Li, S. Zhang, F. Shahabi, S. Xia, Y. Deng, and N. Alshurafa, “Deep learning in human activity recognition with wearable sensors: A review on advances,” *Sensors*, vol. 22, no. 4, p. 1476, Feb. 2022.
- [6] N. Constant, G. Cay, V. Ravichandran, R. Diouf, U. Akbar, and K. Mankodiya, “Data analytics for wearable IoT-based telemedicine,” in *Wearable Sensors*, Elsevier, 2021, pp. 357–378.
- [7] X. Zhang, X. Chen, Y. Li, V. Lantz, K. Wang, and J. Yang, “A framework for hand gesture recognition based on accelerometer and EMG sensors,” *IEEE Transactions on Systems, Man, and Cybernetics - Part A: Systems and Humans*, vol. 41, no. 6, pp. 1064–1076, Nov. 2011.
- [8] A. Akl, C. Feng, and S. Valaee, “A novel accelerometer-based gesture recognition system,” *IEEE Transactions on Signal Processing*, vol. 59, no. 12, pp. 6197–6205, Dec. 2011.
- [9] “Understanding Eye Tracking How It Can Work For You: Definitions, Metrics, And Applications — eyeware.tech,”
- [10] V. Gregori and B. Caputo, *An Analysis of the Visuomotor Behavior of Upper Limb Amputees to Improve Prosthetic Control*. Rome: thesis. Sapienza – University of Rome, 2019.
- [11] D. W. Patmore and R. B. Knapp, “Towards an EOG-based eye tracker for computer control,” in *Proceedings of the third international ACM conference on Assistive technologies*, ACM, Jan. 1998.

- [12] G. Saetta, M. Cognolato, M. Atzori, D. Faccio, K. Giacomino, A.-G. M. Hager, C. Tiengo, F. Bassetto, H. Müller, and P. Brugger, “Gaze, behavioral, and clinical data for phantom limbs after hand amputation from 15 amputees and 29 controls,” *Scientific Data*, vol. 7, no. 1, Feb. 2020.
- [13] S. Said, A. S. Karar, T. Beyrouthy, S. Alkork, and A. Nait-ali, “Biometrics verification modality using multi-channel sEMG wearable bracelet,” *Applied Sciences*, vol. 10, no. 19, p. 6960, Oct. 2020.
- [14] T. Tanaka, I. Nambu, Y. Maruyama, and Y. Wada, “Sliding-window normalization to improve the performance of machine-learning models for real-time motion prediction using electromyography,” *Sensors*, vol. 22, no. 13, p. 5005, Jul. 2022.
- [15] M. Moshawrab, M. Adda, A. Bouzouane, H. Ibrahim, and A. Raad, “Reviewing multimodal machine learning and its use in cardiovascular diseases detection,” *Electronics*, vol. 12, no. 7, p. 1558, Mar. 2023.
- [16] K. Yan, T. Li, J. A. L. Marques, J. Gao, and S. J. Fong, “A review on multimodal machine learning in medical diagnostics,” *Academic Editor: Mariano Torrisi*, vol. 10, no. 3, pp. 123–145, Mar. 2023.
- [17] D. Singh and B. Singh, “Investigating the impact of data normalization on classification performance,” *Applied Soft Computing*, vol. 97, p. 105 524, Dec. 2020.
- [18] K. H. Lee, J. Y. Min, and S. Byun, “Electromyogram-based classification of hand and finger gestures using artificial neural networks,” *Sensors*, vol. 22, no. 1, p. 225, Dec. 2021.
- [19] D. Xiong, D. Zhang, X. Zhao, and Y. Zhao, “Deep learning for EMG-based human-machine interaction: A review,” *IEEE/CAA Journal of Automatica Sinica*, vol. 8, no. 3, pp. 512–533, Mar. 2021.
- [20] “Deep Learning 101: Introduction [Pros, Cons Uses] — v7labs.com,”
- [21] “The Essential Guide to Neural Network Architectures — v7labs.com,”
- [22] R. de Jong, “Multimodal deep learning for the classification of human activity radar and video data fusion for the classification of human activity,” Delft University of Technology, Jan. 2019.
- [23] Z.-S. Wang, J. Lee, C. G. Song, and S.-J. Kim, “Efficient chaotic imperialist competitive algorithm with dropout strategy for global optimization,” *Symmetry*, 2020.
- [24] M. S. Rohith, “Keras EarlyStopping Callback to train the Neural Networks Perfectly — pub.towardsai.net,”
- [25] “Convolutional Neural Networks: Architectures, Types Examples — v7labs.com,”
- [26] A. Thakur, “Intuitive understanding of 1D, 2D, and 3D convolutions in convolutional neural networks. — towardsdatascience.com,”
- [27] S. Verma, “Understanding 1d and 3d convolution neural network (keras) towards-datascience.com,” Sep 20, 2019.
- [28] “Difference between the input shape for a 1D CNN, 2D CNN and 3D CNN — stackoverflow.com,”
- [29] “The Complete Guide to Recurrent Neural Networks — v7labs.com,”

- [30] purnasai gudikandula, “Recurrent Neural Networks and LSTM explained — purnasaigudikandula.medium.com,”
- [31] “Introduction to Recurrent Neural Network - GeeksforGeeks — geeksforgeeks.org,”
- [32] J. Summaira, X. Li, A. M. Shoib, O. Bourahla, L. Songyuan, and J. Abdul, “Recent advances and trends in multimodal deep learning: A review,” 2018.
- [33] “Multimodal Deep Learning: Definition, Examples, Applications — v7labs.com,”
- [34] A. Bhandari, “Understanding Interpreting Confusion Matrices for Machine Learning (Updated 2023) — analyticsvidhya.com,”
- [35] E. Scheme and K. Englehart, “Electromyogram pattern recognition for control of powered upper-limb prostheses: State of the art and challenges for clinical use,” *Journal of Rehabilitation Research and Development*, vol. 48, no. 6, 2011.
- [36] A. Poole and L. J. Ball, “Eye tracking in human-computer interaction and usability research: Current status and future prospects,” in *Encyclopedia of Human Computer Interaction*. Idea Group Reference, 2006, pp. 211–219.

ملخص

يهدف هذا العمل إلى تعزيز التعرف على الإيماءات من خلال الجمع بين الإشارات متعددة الوسائط، بما في ذلك التخطيط الكهربائي السطحي (sEMG)، ومقياس التسارع، وبيانات تتبع العين، باستخدام تقنيات التعلم العميق. تتضمن البنية الهجينة المقترحة الشبكات العصبية التلافيفية (CNNs) لاستخراج الميزات وشبكات الذاكرة طويلة المدى (LSTM) لمعالجة السلاسل الزمنية، بهدف تعزيز دقة التعرف على الإيماءات. يتضمن التقييم مرحلتين: أولاً، تقييم الأداء باستخدام بيانات sEMG وحدها، وثانياً، تقييم مجموعة بيانات متعددة الوسائط باستخدام مقياس التسارع وتتبع العين. تُستخدم مجموعة بيانات MeganePro لتدريب خوارزميات التعلم العميق لتحسين التعرف على إيماءات اليد وتطوير آليات بديهية للتحكم في التفاعل بين الإنسان والحاسوب. يساهم هذا البحث بشكل كبير في مجال التعرف على الإيماءات.

الكلمات المفتاحية: التعرف على الإيماءات، الإشارات متعددة الوسائط، sEMG، مقياس التسارع، تتبع العين، التعلم العميق، الشبكات العصبية التلافيفية (CNN)، الذكرة الطويلة قصيرة المدى (LSTM).

Abstract

This work aims to advance gesture recognition by combining multimodal signals, including surface electromyography (sEMG), accelerometer, and eye-tracking data, using deep learning techniques. The proposed hybrid architecture incorporates Convolutional Neural Networks (CNNs) for feature extraction and Long Short-Term Memory (LSTM) networks for time series processing, aiming to enhance gesture recognition accuracy. The evaluation involves two stages: first, assessing performance with sEMG data alone, and second, evaluating a multimodal dataset with accelerometers and eye-tracking. The MeganePro dataset is used for training deep learning algorithms to improve hand gesture recognition and develop intuitive human-computer interaction control mechanisms. This research significantly contributes to the field of gesture recognition.

Keywords: Gesture recognition, multimodal signals, sEMG, accelerometer, eye tracking, deep learning, Convolutional neural networks (CNN), Long Short Time Memory (LSTM).

Résumé

Ce travail vise à faire progresser la reconnaissance gestuelle en combinant des signaux multimodaux, notamment des données d'électromyographie de surface (sEMG), d'accéléromètre et de suivi oculaire, à l'aide de techniques d'apprentissage profond. L'architecture hybride proposée intègre des réseaux de neurones convolutifs (CNN) pour l'extraction de caractéristiques et des réseaux LSTM (Réseaux de neurones à mémoire à court terme) pour le traitement des séries temporelles, dans le but d'améliorer la précision de la reconnaissance gestuelle. L'évaluation comprend deux étapes : d'abord, évaluer les performances uniquement avec les données sEMG, puis évaluer un ensemble de données multimodal comprenant des accéléromètres et du suivi oculaire. Le jeu de données MeganePro est utilisé pour entraîner des algorithmes d'apprentissage profond afin d'améliorer la reconnaissance des gestes de la main et de développer des mécanismes de contrôle intuitifs pour l'interaction homme-machine. Cette recherche contribue de manière significative au domaine de la reconnaissance gestuelle.

Mots-clés : Reconnaissance des gestes, signaux multimodaux, sEMG, accéléromètre, suivi oculaire, apprentissage profond, réseaux neuronaux convolutifs (CNN), Réseaux de neurones à mémoire à court terme (LSTM).