Università degli Studi di Padova

Dipartimento di Ingegneria dell'Informazione

Bioingegneria per le Neuroscienze

Multikernel convolutional neural network for sEMG based hand gesture classification

Relatore: Prof. Manfredo Atzori

Correlatore: Dott. Niccolò Marini

Candidato:

Riccardo Fratti

Data di laurea 09/10/2023

Anno Accademico 2022 - 2023

Abstract (EN)

Recognizing hand gestures is a complex topic that involves analyzing various techniques related to input signals and algorithms. Despite the ongoing research, constructing robust and reliable classifiers remains challenging due to the high variability among subjects and the unpredictability of daily factors. This project proposes and compare different strategies that utilizes forearm Electromyography (EMG) signals to address some of these issues.
The proposed classifier uses a Multi Kernel Convolutional Neural Network (MKCNN)and is tested on multiple open-access databases with 14 classes selected from the ADL (Activity of Daily Living). Few basic pre-processing steps are employed to minimize computational demands. This is particularly critical in small embedded devices, such as those used in prosthetic control, where excessive computation can lead to significant delay, energy consumption and heat generation.
Additionally, Triplet Margin Loss and Adversarial training via Reversal Gradient are exploited to maximize generalization capabilities of the MKCNN. The effectiveness of the two different algorithms are used to overcome the challenges posed by daily variables, such as skin impedance, sensor positioning, muscle variations, sweat, humidity, or interference. Finally, transfer learning is implemented to address the limitation of inter-subject variability by pre-train the model over multiple subjects and fine-tuning it on the final user.
Overall, this project represents an advancement in the field of hand gesture recognition and could pave the way for the development of more reliable and efficient prosthetic control systems.

Abstract (IT)

Il riconoscimento dei gesti delle mani costituisce un argomento complesso che richiede l'analisi di varie tecniche relative ai segnali di ingresso e agli algoritmi. Nonostante le ricerche in corso, rimane una sfida considerare classificatori robusti e affidabili a causa dell'alta variabilità tra i soggetti e dell'imprevedibilità dei fattori quotidiani. Questo progetto propone e confronta diverse strategie che impiegano i segnali dell'elettromiografia (EMG) dell'avambraccio per affrontare alcune di queste sfide. Il classificatore proposto utilizza una rete neurale convoluzionale multi-kernel (MKCNN) ed è stato testato su vari database accessibili al pubblico con 14 classi selezionate dalle attività quotidiane. Per ridurre al minimo i requisiti computazionali, sono state adottate poche fasi di pre-elaborazione, il che è particolarmente cruciale per dispositivi embedded di piccole dimensioni, come quelli utilizzati per il controllo delle protesi. Questo evita ritardi significativi, consumo di energia e surriscaldamento. Inoltre, sono state implementate tecniche come la perdita di margine della tripletta e l'addestramento avversario tramite gradiente inverso per massimizzare la capacità di generalizzazione della MKCNN, affrontando le sfide imposte dalle variabili quotidiane, come l'impedenza cutanea, il posizionamento del sensore, le variazioni muscolari, il sudore, l'umidità o le interferenze. Infine, l'apprendimento per trasferimento è stato utilizzato per mitigare la variabilità tra i soggetti, preaddestrando il modello su più soggetti e ottimizzandolo per l'utente finale. Nel complesso, questo progetto rappresenta un avanzamento significativo nel campo del riconoscimento dei gesti della mano e potrebbe aprire la strada allo sviluppo di sistemi di controllo protesico più affidabili ed efficienti.

1 | Introduction

The field of Human-Machine Interaction (HMI) has witnessed remarkable growth, with numerous applications and interfaces emerging to facilitate natural interactions between humans and computing systems. One pivotal aspect of HMI is Hand Gesture Recognition (HGR), which plays a crucial role in interpreting human commands and intentions. HGR models have found diverse applications, from enhancing intelligent prostheses to enabling sign language recognition, aiding rehabilitation devices, and facilitating device control.

To empower these HGR models, various methods for data acquisition have been explored, including gloves, vision sensors, inertial measurement units (IMUs), and surface electromyography (EMG) sensors, either individually or in combinations. While each of these technologies brings unique advantages, they also come with their own limitations. For instance, gloves can restrict natural hand movements while vision sensors can face challenges related to occlusion or lighting variations, for this reason they may not be suitable for amputees. IMUs and surface EMG sensors, on the other hand, offer a valuable advantage in capturing the intent behind hand movements. Surface EMG sensors, in particular, have proven effective even in cases where individuals are physically unable to execute specific movements while retaining the intention to do so [1]. This unique attribute has positioned them as a forefront technology in the quest for prosthetic control devices.

1.1 Electromyography

Electromyography (EMG) is a method to measure the electrical activity of skeletal muscles, which allow us to perform voluntary movements, and their activity is controlled by alpha motor neurons. When an alpha motor neuron is activated (i.e., discharges), signals from the brainstem/spinal cord are transmitted to the neuromuscular junction, which causes muscle fibers to contract [2]. This results in a transient change in the electrical potential, known as motor unit action potentials (MUAPs). It is possible to detect this signal using electrodes that are placed either on the skin or inserted into the muscles, as shown in **Figure** 1.1.
When a muscle contracts, charged particles (ions) flow across its fiber membrane, constituting an electrical current (I), which is measured in Amperes (A) or electric charge per second. These electrical currents affect the electrical potential in the surrounding tissue, creating a voltage difference, measured in Volts (V), between two points. The voltage recorded at the skin surface reflects this electrical activity. It's influenced by factors like resistance or impedance, quantified in Ohms (Ω), which is the obstacle to electric current flow posed by the muscle, subcutaneous tissue, and skin. The time-varying voltage distribution on the skin surface resulting from

muscle activity is known as the surface electromyography (sEMG) signal.
Using surface EMG, what is actually registered is the summation, in space and time, of multiple MUAPs [3], which provide a 2D representation of the electric potential across the surface of the skin, similar to the ECG on the chest or the EEG on the scalp. This electrical signal evolves over time and is obtained by sampling its analog version both spatially and temporally using an electronic sampler.

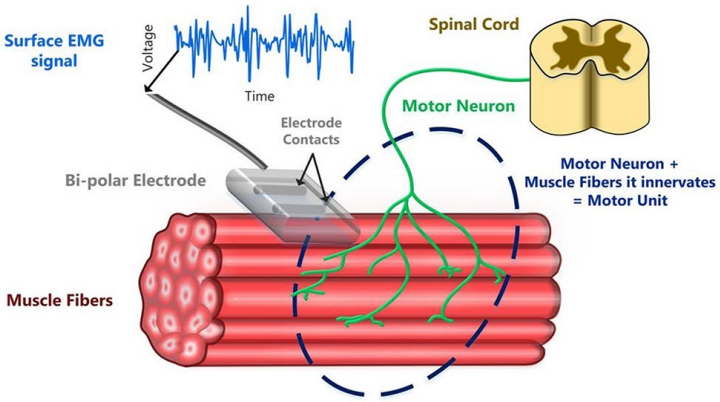

Figure 1.1: Motor Unit

1.1.1 Motor neuron action potential

A motor unit (MU) consists of an alpha motor neuron and the muscle fibers it controls. When this motor neuron discharges, all the muscle fibers within that unit activate together, creating a motor unit action potential (MUAP). The sEMG (surface electromyogram) signal recorded is the sum in space and time of these MUAPs. Depending on the required force and contraction speed, dozens to hundreds of motor units can be activated [4]. MUAPs typically last between 5 and 15 milliseconds [5], measured from their initial deviation from baseline and returning to it. These MUAPs vary in number, size, and shape, providing crucial information on the health and function of motor units. This information is valuable for diagnosing and monitoring neuromuscular disorders.

1.2 Applications

Surface Electromyography, or sEMG, is a powerful tool in the field of physiological and muscular studies. It provides valuable insights into the intricate interplay between the nervous system and muscles. By recording the electrical activity generated during muscle contractions, sEMG offers a window into the inner workings of the neuromuscular system.

sEMG data can reveal a wealth of information about muscle function, including muscle recruitment patterns, timing of muscle activation, and coordination of muscle groups.

Moreover, the versatility of sEMG extends far beyond research alone. Its applications span a wide range of fields, from healthcare to human-computer interaction. The data collected through sEMG can be harnessed to create innovative solutions

that improve our lives, improve healthcare practices, and open doors to new possibilities. In particular, in the domain of hand-gesture recognition, sEMG finds prominent application in the following contexts:

1. **Communication**: in some situations, individuals face challenges in using spoken language for communication due to health conditions such as muteness or certain impairments. To address this need, sign languages emerged as a forms of communication relying on hand gestures. However, the reach of sign languages is limited, varying from one country to another, and not all individuals are proficient in them. This limitation necessitates the presence of trained intermediaries for effective communication. In this context, the development of automated hand gesture recognition systems may have an high impact. Infact, these systems can interpret and translate dynamic hand gestures into corresponding words or sentences [6]. Essentially, they serve as digital translators [7], facilitating communication between individuals who rely on sign languages and those who do not. This innovation plays a crucial role in enhancing inclusivity and bridging communication gaps in the communication contexts.

2. **Rehabilitation**: the sEMG's ability to decode muscle symmetry and activation patterns during different movements is usually exploited by therapists, enabling them to precisely tailor rehabilitation programs [8]. Furthermore, sEMG biofeedback enhances patient control by increasing their awareness of muscle dynamics, which can help reduce spasticity, especially in individuals with neurological conditions such as stroke or cerebral palsy [9]. Another significant application involves the utilization of assistive robots, specifically sEMG-controlled exoskeletons [10]. These exoskeletons are frequently employed in cases of severe impairment, as they enable patients to perform precise and corrective movements, thereby significantly augmenting their rehabilitation progress [11]. Notably, these exoskeletons find extensive use in both upper and lower limb movements [12].

3. **Prosthetics control**: thanks to modern technology, it is possible now achieve remarkable control over prosthetic devices. This is made possible by capturing and interpreting electrical signals generated by the muscles remaining in the user's residual limb. Prostheses controlled via surface electromyography (sEMG) provide users with an unparalleled level of dexterity and control previously unattainable [13]. This newfound capability empowers individuals with limb loss to regain a natural and effortless grip on their daily activities. It not only enhances their quality of life but also redefines the potential of prosthetic hand functionality.

4. **Gaming and VR**: nowadays, the traditional gaming experience is not enough to satisfy the hunger of today's gamers. Hand-gesture recognition adds a layer of immersion to gaming experiences. Gamers can control characters and actions with natural hand movements, making gameplay more intuitive and exciting. Furthermore, within a virtual reality (VR) environment, gesture-based controls enable players to actively interact with games [14]. This interaction fosters a profound sense of presence and realism. Notably, these advan-

tages have found extensive application in the realm of rehabilitation [15], [16] through gamification, yielding highly promising outcomes.

1.3 Sensing modalities

Upper-limb EMG detection can be mainly categorized into two types: invasive and non invasive. Both of those modalities generally come within the form of electrode arrays or unpaired ones which can be strategically positioned inside (needles electrodes) or above target muscles. Even if the invasive methods has been proved to provide a more accurate, clean and stable signal [17], they come with the needing of surgical operations. There are sEMG devices that require conductive gel and others that use dry electrodes that stick to the skin directly, providing easy and quick implantation. Additionally, there are wireless sEMG systems[18] that improve client comfort and mobility, enabling them to participate in a variety of activities without tedious wires. Other sensing modalities for hand-gesture recognition include Force myography (FMG), Near-Infrared Spectroscopy (NIRS), Radiomyography (RMG), and Inertial measuring units (IMU).

1.4 Embedded system constrains

In real-time Hand Gesture Recognition (HGR) utilizing EMG signals, the concept of "controller delay" is of paramount importance. This term refers to the time it takes for the HGR system to recognize a performed gesture and subsequently respond to it. The significance of minimizing controller delay lies in providing users with a seamless and responsive interaction experience.

The ongoing challenge in the development of HGR systems is to strike a balance between minimizing controller delay and ensuring accurate gesture recognition.

However, in more intricate and fine-tuned tasks, like controlling a robotic hand or executing precise maneuvers, users can tolerate slightly longer delays, which may extend up to 300 ms [19]. Beyond this threshold, delays become noticeable and can significantly impact the user experience, leading to frustration and diminished usability.

Another significant constraint in the context of prosthetic devices, particularly for daily use, is power consumption. Prostheses are expected to operate efficiently on a daily basis, which necessitates the use of models and systems that are either simplistic or designed for low power consumption.

A promising advancement in this context has arisen from the University of Bologna. They have spearheaded techniques that involve quantization, which may result in a minor reduction in accuracy, but significantly enhance energy efficiency in prosthetic devices [20], [21].

1.5 Machine learning applications to sEMG

In recent years, deep learning has emerged as a potent tool in various fields such as speech recognition, computer vision, and natural language processing. This success has extended to electromyography (EMG) pattern recognition, where deep learning techniques have been applied to enhance the accuracy and robustness of EMG-based gesture recognition systems.

1.5.1 Classical machine learning approaches

Classical machine learning techniques encompass a variety of algorithms that have played a foundational role in the field of sEMG. It's common the usage of K-NN, which classifies data points by the considering the majority class of their nearest neighbors in the feature space. SVM, or Support Vector Machines, aims to find an optimal hyperplane that best separates data points into different classes. Decision Trees, a fundamental building block that maps out decision paths based on input features. Lastly, Random Forest, an ensemble method, takes a large number of decision trees and combines their outputs to enhance predictive accuracy (e.g. major voting). All this techniques usually employ hand-crafted feature, that need to be extracted from the data, thus, involving a significant computational burden int he case of real-time applications. Nonentless,these classical techniques have paved the way for more advanced machine learning approaches, offering valuable tools for tackling a wide range of real-world problems.

1.5.2 Deep learning approcahes

The landscape of sEMG gesture recognition has been significantly shaped by Convolutional Neural Networks (CNNs). Early pioneers [22] introduced user-adaptive CNN models, achieving remarkable classification accuracy improvements compared to conventional methods. Additionally, Atzori et al. [23] presented a modified LeNet CNN architecture that outperformed traditional techniques, such as Support Vector Machine, Random forest, Linear Discriminant Analysis and k-Nearest Neighbor.

Recent research efforts have explored innovative techniques in this domain. Wei et al. [24] delved into a "decomposition-and-fusion" strategy, employing a two-stage multi-stream CNN approach, proving its supremacy over random forests and basic CNNs. Moreover, multi-kernel structures [25] have demonstrated their effectiveness in enhancing classification.

To further boost CNN performance, several studies have meticulously tweaked various elements, encompassing pre-processing, hyperparameters, and network layouts [26]. Meanwhile, the realm of 3D CNNs has been harnessed to process intricate finger movement data, particularly in HDsEMG contexts [27].

Another dimension of deep learning methods has embraced Temporal Convolutional Networks (TCNs). These have gained traction in sEMG-based gesture recognition due to their innate ability to capture temporal dimensions. Their utility extends to enhancing motion intention prediction, especially during transitions between classes [28]. Notably, TCNs have exhibited superior performance compared to classical ML methods [29], making them an optimal choice for resource-constrained embedded devices.

Recurrent neural networks (RNNs), especially LSTM variants, have gained significant traction in time series analysis, offering a robust foundation for sEMG data processing as well. Promisingly, the fusion of CNN and RNN, referred to as hybrid architectures, presents substantial potential in sEMG applications. One notable approach integrates LSTM and CNN into a unified structure [**Karman2022**, 30]. Alternatively, other models leverages LSTM to preserve temporal information while complementing it with CNN for spatial feature extraction [31]. These hybrid models represent a compelling direction in sEMG-based research, combining the strengths of both convolutional and recurrent neural networks.

Recent trends in sEMG gesture recognition also encompass the adoption of unsupervised learning algorithms, requiring minimal to no data labeling. For instance, Vujaklija et al. [32] leveraged Auto-encoders to transform noisy input data into a cleaner format. Unsupervised learning is advantageous in eliminating the need for labor-intensive human data labeling. However, self-learning techniques may introduce classification errors, particularly in scenarios involving shifts in data distribution [33]. Other CNN-based studies in the sEMG domain are geared toward enhancing model scalability through domain adaptation [34], or self-recalibrating classification strategies [35].

1.6 Objectives

In this work, a preliminary and comprehensive analysis of the methodologies used for the pre-processing of surface electromyography (sEMG) signals is presented, with the aim of optimizing data quality and relevance. Deep investigations are conducted to acquire significant insights into the selection of appropriate normalization types and modes, as well as to address critical factors connected to the selection of the time window dimension.

Furthermore, the development of a sophisticated model for the classification of hand gestures based on sEMG data is tested over four publicly available datasets. This model employs cutting-edge techniques, particularly convolutional neural networks (CNNs), equipped with filters of varying sizes to capture intricate relationships among different frequency components. A systematic exploration of the influence of hyperparameters on the model's performance is shown to ensure optimal results.

One key objective is to address the pressing issue of generalization in the field of sEMG-based hand gesture recognition. To this end, a comprehensive comparison of the model's capabilities with three different methodologies is conducted. In particular, the aim consists of enhancing the adaptability of the model to new subjects while extracting universal relationships whenever possible. The work aims to provide valuable insights into the potential of deep learning in the realm of sEMGs and develop a classifier capable of real-time applications that surmount the challenges typical of modern devices. The structure of this work guides the reader through method explanations, followed by results, and critical discussions.

2 | Data

In this section, essential details about the datasets employed in the study are provided. Publicly available databases commonly used for sEMG signal research, particularly for hand gesture recognition, were selected. These databases were chosen to enable thorough cross-correlation analyses, assessing the effectiveness of various algorithms and overall data quality.

2.1 Databases

In this Master's thesis, four widely recognized public databases were selected, which are commonly used as benchmarks in the field of sEMG signal research, specifically for hand gesture recognition These databases are NinaPro DB1, DB2, DB3, and DB7 [36], [37]. An overview of databases main feature is visible in **Table 2.1**.

The choice was made primarily because they feature the same types of exercises, enabling comprehensive cross-correlated analyses. This approach allows for the evaluation of the effectiveness of various algorithms and the assessment of overall data quality.

Furthermore, it's worth noting that the last three databases (DB2, DB3, and DB7) share a common acquisition device. To enhance the volume of data available for the study, these databases were merged, contributing to a more extensive dataset for the research

2.1.1 Data acquisitions protocol

For DB2,DB3, and DB7, Muscle activity was monitored using 12 TrignoWireless electrodes from Delsys, Inc. These electrodes recorded sEMG signals at a sampling rate of 2 kHz. Among these electrodes, eight were evenly spaced around the forearm

Table 2.1: Description of Nina-Pro Databases

Name	Subjects	Channels	Classes	Repetitions	Frequency	Device
DB1	27	10	52	10	100 Hz	MyoBook 13E200
DB2	40	12	41	6	2000 Hz	Delsys Trygno
DB3	11 (A)	12	41	6	2000 Hz	Delsys Trygno
DB7	20+2 (A)	12	41	6	2000 Hz	Delsys Trygno

Note: In the context of this study, the letter A in *Subjects* designates the number of amputees.

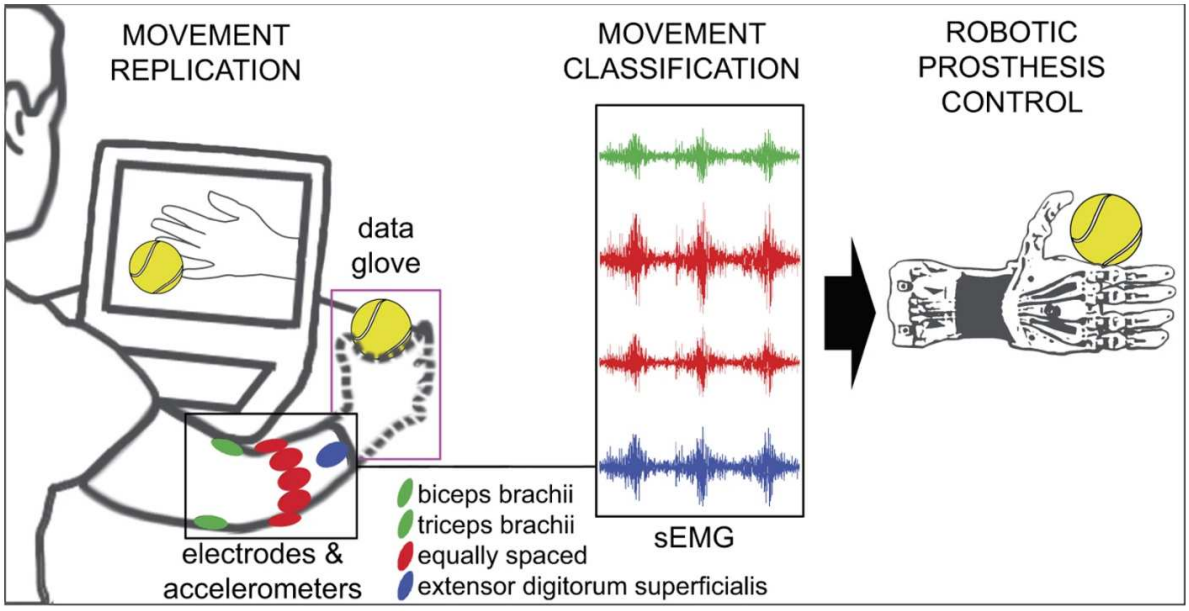

Figure 2.1: Acquisition set up

at the level of the radio-humeral joint. Two electrodes were strategically placed on the key activation spots of the flexor digitorum superficialis and the extensor digitorum superficialis muscles. The remaining two electrodes were positioned at the primary activity sites of the biceps brachii and the triceps brachii.
For DB1, instead, 10 Otto Bock MyoBock 13E200 electrodes were used. Same positions were used except for the last two missing. Electrode placement and an overview of the methodology adopted are shown in **Figure** 2.1.
All four databases used the same data acquisition protocol described by M. Atzori et al. in [36].
It's important do mention that, during acquisition, each exercise was interspersed with 3 seconds of rest to alleviate as much muscle fatigue as possible, which would have compromised the regularity of the signals [38].
Finally, because of the large drop in performance due to the number of gestures to be classified, a subset of 14 gestures was chosen from those available based on the ADL (activity daily living) visible in **Figure 2.2**.

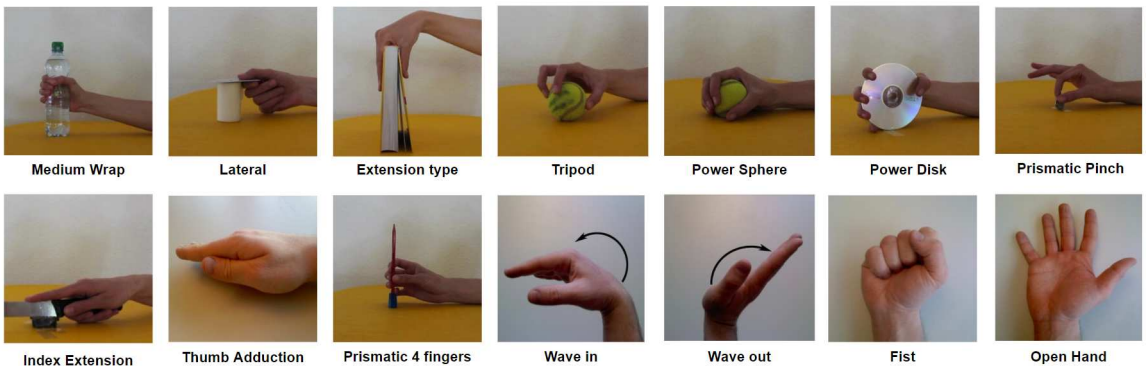

Figure 2.2: Selected gestures

2.2 Pre-processing

Regarding signal processing, the publicly available data undergo three essential steps:

1. **Filtering**: to eliminate interference from 50 Hz power-line signals (and harmonics), a Hampel filter was employed. This step was necessary because the Delsys electrodes were not shielded against such interferences.

2. **Synchronization**: to ensure consistency, the data streams were synchronized by up-sampling to the highest sampling frequency of 2 kHz using linear interpolation.

3. **Relabelling**: human movements may not always align precisely with the video stimuli used, due to delay response of the human brain. To address this, any inaccuracies in the movement labels were corrected using an offline generalized likelihood ratio algorithm.

These preprocessing steps helped ensure that the sEMG data was in a suitable and reliable format for subsequent analysis. Data from each database were extracted and reconstructed in the form of a dataframe-transformed csv file using Pandas python library and extracting and remapping only the exercises of interest.

2.2.1 Filtering

The dataframes underwent another filtration process that employed a 5th-order Butterworth bandpass filter within the (20 - 500) Hz range, adhering to the standard practice in sEMG signal processing [39]. In fact, approximately 95% of the spectral power in EMG signals resides within the 400 - 500 Hz range [40]. Higher frequencies are usually related to interference while values below 20 Hz are often associated with motion-related artifacts.

2.2.2 Windowing

Before being passed to the model, the signal underwent a windowing process. This involved creating a new dataframe with customized indexes based on the chosen window size.

The selection of window size was primarily influenced by the real-time application's requirements. For datasets sampled at 2000 Hz, window sizes of 200, 300, and 400 points were chosen, equivalent to time intervals of 100, 150, and 200 miliseconds, respectively, all with 75% overlap.

Additionally, a 20 ms window (comprising 40 points) was chosen, considering the potential use of a majority voting strategy, as it has been proved a valid option [41]

In contrast, for the 100Hz DB1 database, a single 20-point window (equivalent to 200 ms) was selected.

This diverse set of window sizes allowed for an assessment of the impact and significance of this pre-processing step.

2.2.3 Normalization and Rectification

In the pursuit of enhancing data homogeneity and handle inter-subject variability, three normalization types were tested: scaling data to the intervals (-1, 1) and (0, 1), as well as employing the z-score normalization technique. These methods, inherently subject-dependent, have been evaluated:

- **Collective**: compute basics parameters, such as maximum, minimum, mean, and standard deviation, from all channels collectively.
- **Channel-level**: retrieve the same set of parameters from each channel independently.

The choice of normalization methods prioritized simplicity and suitability for real-time applications. Moreover, the requisite parameters for normalization (maximum, minimum, mean, standard deviation) can be stored externally for direct utilization in the online application, circumventing the need for additional computations, and they can easily be updated for daily calibration.

Signal rectification was chosen because of its minimal computational cost and as it is a common practice for sEMG [**Xiong2020**]. Moreover, a preliminary experiment revealed that although there were negligible differences from a metric perspective, there was a slight trend towards faster convergence.

3 | Methods

3.1 Machine learning applications to sEMG

In recent years, deep learning has emerged as a powerful tool in various fields such as speech recognition, computer vision, and natural language processing. This success has extended to electromyography (EMG) pattern recognition, where deep learning techniques have been applied to enhance the accuracy and robustness of EMG-based gesture recognition systems.

3.1.1 Classical machine learning approaches

Classical machine learning techniques encompass a variety of algorithms that have played a foundational role in the field of sEMG. It's common the usage of K-NN, which classifies data points by the considering the majority class of their nearest neighbors in the feature space. SVM, or Support Vector Machines, aims to find an optimal hyperplane that best separates data points into different classes. Another common approach involves the use of Decision Trees, a fundamental building block that maps out decision paths based on input features. Lastly, Random Forest, an ensemble method, takes a large number of decision trees and combines their outputs to enhance predictive accuracy (e.g. major voting). All these techniques usually employ hand-crafted features that need to be extracted from the data, thus involving a significant computational burden in the case of real-time applications. Undoubtedly, these classical techniques have paved the way for more advanced machine learning approaches, offering valuable tools for tackling a wide range of real-world problems.

3.1.2 Deep learning approaches

The landscape of sEMG gesture recognition has been significantly shaped by Convolutional Neural Networks (CNNs). Early pioneers [22] introduced user-adaptive CNN models, achieving remarkable classification accuracy improvements compared to conventional methods. Additionally, Atzori et al. [23] presented a modified LeNet CNN architecture that outperformed traditional techniques, such as Support Vector Machine, Random Forest, Linear Discriminant Analysis and k-Nearest Neighbor.

Recent research efforts have explored innovative techniques in this domain. Wei et al. [24] delved into a "decomposition-and-fusion" strategy, employing a two-stage multi-stream CNN approach, proving its supremacy over random forests and basic CNNs. Moreover, multi-kernel structures [25] have demonstrated their effectiveness in enhancing classification.

To further boost CNN performance, several studies have meticulously tweaked

various elements, encompassing pre-processing, hyperparameters, and network layouts [26]. Meanwhile, the realm of 3D CNNs has been harnessed to process intricate finger movement data, particularly in HDsEMG contexts [27].

Another dimension of deep learning methods embrace Temporal Convolutional Networks (TCNs). These have gained traction in sEMG-based gesture recognition due to their innate ability to capture temporal dimensions. Their utility extends to enhancing motion intention prediction, especially during transitions between classes [28]. Notably, TCNs have exhibited superior performance compared to classical ML methods [29], making them an optimal choice for resource-constrained embedded devices.

Recurrent neural networks (RNNs), especially LSTM variants, have gained significant traction in time-series analysis, offering a robust foundation for sEMG data processing as well. Promisingly, the fusion of CNN and RNN, referred to as hybrid architectures, presents substantial potential in sEMG applications. One notable approach integrates LSTM and CNN into a unified structure [**Karman2022**, 30]. Alternatively, other models use LSTM to preserve temporal information while complementing it with CNN for spatial feature extraction [31]. These hybrid models represent a compelling direction in sEMG-based research, combining the strengths of both convolutional and recurrent neural networks.

Recent trends in sEMG gesture recognition also encompass the adoption of unsupervised learning algorithms, requiring minimal to no data labeling. For instance, Vujaklija et al. [32] leveraged auto-encoders to transform noisy input data into a cleaner format. Unsupervised learning is advantageous in eliminating the need for labor-intensive human data labeling. However, self-learning techniques may introduce classification errors, particularly in scenarios involving shifts in data distribution [33]. Other CNN-based studies in the sEMG domain are geared toward enhancing model scalability through domain adaptation [34], or self-recalibrating classification strategies [35].

3.2 Model Structure

The architecture of the proposed Multi-Kernel Convolutional Neural Network (MKCNN) is visually depicted in **Figure** 3.1. The MKCNN model consists of four distinct parts, each serving a unique purpose, elaborated as follows:

Part 1 This section comprises two blocks, as illustrated in **Figure** 3.2. The objective of Part 1 is to extract comprehensive features capable of capturing the spatio-temporal intricacies present within the sEMG signal. Each sample of the multichannel sEMG signal is interpreted as an image. In Block 1, a set of five diverse filters, of sizes 10×3, 20×3, 30×3, 40×3, and 50×3, is employed. This simultaneous extraction of temporal and spatial information from the input sEMG signals contributes to a richer representation. The filter sizes play a pivotal role in influencing the receptive field of the neural units, consequently impacting the temporal information of the extracted deep features. Larger filter sizes are generally associated with lower-frequency features. In Block 2, a comparable structure to that of Block 1 is replicated. Nevertheless, in Block 2, the number of neurons is increased twofold through the utilization of five convolution layers, each featuring uniform 3×3 filter dimensions. This helps us

map the extracted features into a higher-dimensional space. It is important to mention that the convolution layer configuration in Block 2 adopts separable convolution to accomodate the transition from single-channel images in Block 1 to multichannel feature maps in Block 2. Furthermore, the integration of Batch Normalization (BN) layers and dropout layers in both blocks aims to enhance the overall generalization capabilities of the MKCNN model.

Part 2 The primary goal of this component is to achieve dimensionality reduction on the feature maps emanating from *Part* 1, subsequently merging these maps. Utilization of 1×1 filters within this convolutional layer after *Part* 1 results in a reduction in the number of feature maps, decreasing the count from 320 (64×5) to 128.

Part 3 Building upon the preceding structure, *Part* 3 delves into the further extraction of deep features. Achieved through the implementation of Separable Convolutional layers, this facet comes into play despite the utilization of multi-scale convolutional layers in the previous stages. The rationale here is to tap into deep convolutional layers, enabling a more profound extraction of semantic information.

Part 4 The pivotal role played by this final part is in establishing a robust mapping relationship connecting deep features with the ultimate classification results. Through the integration of two fully connected layers, the generation of intricate fused deep representations is enabled, facilitating more refined classification results.

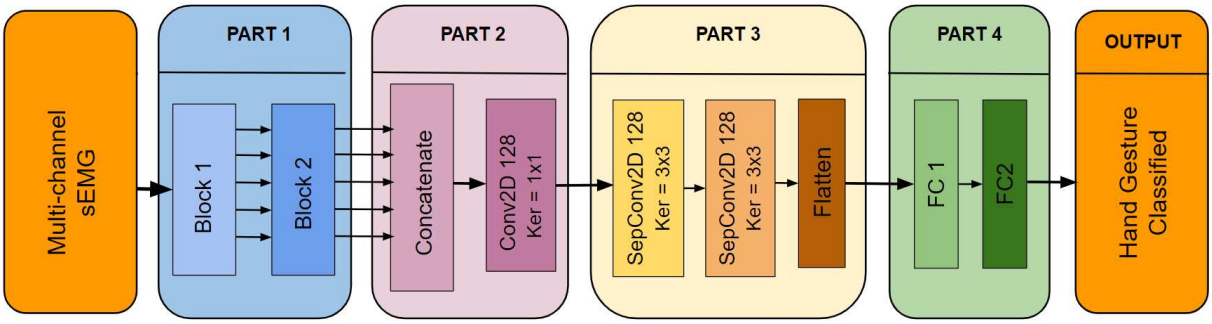

Figure 3.1: Architecture of model

A customized Convolution layer was strategically employed for the initial convolutions (*Conv2D_ Circular8*). The reason for this choice is that in all the databases used, there are 8 channels arranged in a circular fashion around the arm. When transitioning from this circular physical configuration to a linear computing arrangement (rows by columns), the spatial relationships between the first and last channels, as well as between the first and penultimate channels, and so on, are lost. In order to overcome this problem, this specialized layer uses circular padding along the first 8-channel axis and zero temporal padding with a specific purpose: to preserve the spatial relationship between electrodes. The architecture of the model, including its layer types, kernel sizes, output sizes, activation functions, and options, is presented in **Table 3.1** below.

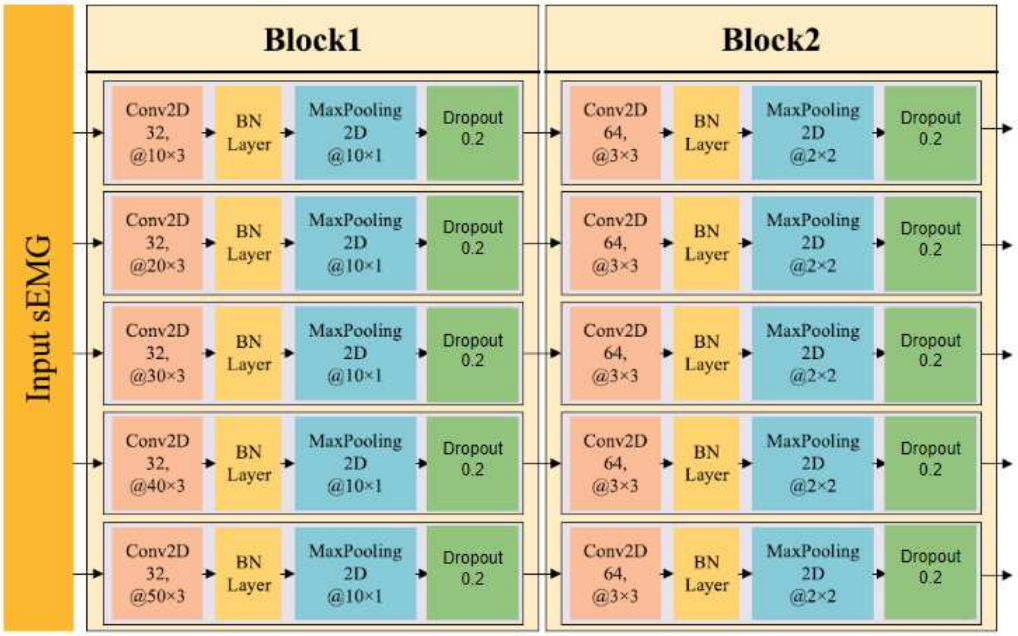

Figure 3.2: Enlargement of first two block of Part one

Table 3.1: Architectural Layout and Data Pathways

Part	Layer Type	Kernel Size	Output Size	Activ	Options
1	1-Input	-	1/300x12	-	-
	5-Conv2D_Circular8	[15 - 75] x3	32/300x14	ReLU	pad='circular'
	5-BatchNorm2D	-	32/300x14	-	-
	5-MaxPooling2D	15x1	20x14	-	-
	5-Dropout2D	-	-	20x14	rat e= 0.2
	5-SeparableConv2D	3x3	64/20x14	ReLU	pad=0
	5-BatchNorm2D	-	64/20x14	-	-
	5-MaxPooling2D	2x2	10x7	-	-
	5-Dropout2D	-	10x7	-	rate=0.2
2	1-Concatenate	-	320/10x7	-	-
	1-Conv2D	128/1x1	128/10x7	ReLU	-
3	1-SeparableConv2D	3x3	128/10x7	ReLU	pad=0
	1-BatchNorm2D	-	128/10x7	-	-
	1-MaxPool2D	2x2	128/5x3		-
	1-Dropout2D	-	128/5x3	-	rate =0.2
	1-SeparableConv2D	3x3	128/5x3	ReLU	pad=0
	1-BatchNorm2D	-	128/5x3	-	-
	1-MaxPool2D	2x2	128/2x1	-	-
	1-Dropout2D	-	128/2x1	-	rate= 0.2
4	1-Flatten	-	256	-	-
	1-Linear	-	128	ReLU	-
	1-Linear	-	14	SoftMax	-

Note: The number before layer type represents the number of this layer, e.g., 5-Conv2D represents there are five parallel conv2D layers. The Kernel size of the first Convolution uses a fixed second dimension of 3, while the first dimension varies from the lower value to the upper value in increments of 10.

3.2.1 Triplet Margin Loss

The triplet margin loss is a popular choice in deep learning for domain adaptation. It operates by learning embeddings or representations of data in a way that enhances the similarity between similar samples (positive pairs) and pushes dissimilar samples (negative pairs) apart in the high dimensional feature space. The loss is typically defined as:

$$L_{\text{triplet}} = \sum_{i} \left[\max\left(0, \text{margin} + d(a_i, p_i) - d(a_i, n_i)\right)\right] \tag{3.1}$$

where a_i represents the anchor sample, p_i represents a positive sample (with the same label as the anchor), n_i represents a negative sample (with a different label from the anchor), and margin is a margin parameter that enforces a minimum distance between positive and negative samples. See appendix for more information.

The anchor in this study consists of information on the subject and their label. Positive examples were labels that matched, but originated from distinct subjects. Negative samples, on the other hand, were chosen to represent various labels in order to emphasize dissimilarity in the embedding space. This method assisted the model in learning robust features that are domain-invariant as well as effective for the primary classification job. The balance between the two loss function is weighted by a factor (λ), which was set to 0.5 in these experiments.

3.2.2 Domain adaptation with Reversal Gradient variant

Domain adaptation is a technique used to adapt a machine learning model trained in one domain (source domain) to perform well in another domain (target domain) [42]. In the context of this study, the reverse gradient algorithm is employed for this purpose. This algorithm involves the use of a second classifier connected to the primary model. In particular, this connection is established at the end of *Part* 3 of the MKCNN architecture. The objective is to classify the data based on the subject to which it belongs, and it is achieved by minimizing a function based on cross-entropy loss.

The key idea behind the reverse gradient algorithm is to penalize the primary model's loss function based on the domain classifier's performance. This penalization encourages the model to learn features that are invariant to the domain, in this case different subjects. In mathematical terms, the objective function to be minimized can be expressed as:

$$\min_{\theta_f, \theta_c} \frac{1}{n} \sum_{i=1}^{n} \mathcal{L}_c(f(x_i; \theta_f); y_i) - \lambda \cdot \frac{1}{m} \sum_{j=1}^{m} \mathcal{L}_d(g(x_j; \theta_f); d_j) \tag{3.2}$$

Here, θ_f represents the parameters of the primary model, θ_c represents the parameters of the domain classifier, $f(x_i; \theta_f)$ is the output of the primary model for input x_i, y_i is the true label for x_i, $g(x_j; \theta_f)$ is the output of the domain classifier for input x_j, d_j is the domain label for x_j (indicating the subject to which the data belongs), n is the number of source domain samples, m is the number of target domain samples, $\mathcal{L}_c$ is the cross-entropy loss function for classification, and $\mathcal{L}_d$ is the cross-entropy loss function for domain classification. The hyperparameter λ controls the trade-off between the two loss terms, and in this work has been set as 0.5

In simpler terms, this equation combines two cross-entropy losses, one for classification and one for domain classification, with a trade-off controlled by the hyperparameter λ

In this implementation, the reversal gradient algorithm involves multiplying the gradients by an exponential scaling factor, which gradually increases from 0 to 1 over the course of training epochs. This scaling factor serves the purpose of gradually transitioning the model's focus from the main task (hand gesture classification), to domain generalization. Initially, with a scaling factor of 0, the model prioritizes learning domain-invariant features. As the scaling factor increases, the model progressively shifts its emphasis towards domain generalization while retaining the capacity to perform the primary task effectively.

3.3 Experimental set-up

Due to the significant variability among subjects and the ultimate goal of using the classifier in the prosthetic field, user-dependent fine-tuning was considered as essential from the outset. For this reason, 20% of the subjects were initially set aside and treated as new users for testing the fine-tuning capability through transfer learning.

Subsequently, each database was divided into three sets: training, validation, and testing using repetitions [2, 4, 6], [1, 5], and [3], respectively. This specific division was chosen to mitigate the effects of muscle fatigue and assess how effectively the model learns features that are not solely dependent on exercise conditions (e.g., changes in skin resistance, humidity, and electrodes shift).

3.3.1 Transfer Learning set-up

Transfer learning is a powerful technique in deep learning where a pre-trained model, often developed on a large dataset for a specific task, is fine-tuned or adapted for a different but related task.

To achieve this, certain layers of the pre-trained model are frozen, meaning their weights are kept fixed, while other layers are modified to suit the new task. In this study, transfer learning was employed to tailor the model to individual subjects, thereby exploring the model's adaptability. The process involved two distinct experiments for each of the three algorithms used:

In the first experiment, only the final two linear layers (*Part* 4) were trained on the new subject-specific data. In the second experiment, a more extensive retraining was carried out, taking advantage not only of the final layers but also *Part* 2 and *Part* 3 of the model.

The subject-specific dataset was once again divided based on repetition to observe changes in the model's accuracy as the amount of available data changed. The training procedure was gradual, beginning with one repetition, then to two, then three repetitions, and so on, while being tested on the others. The order of repetitions was set as [2, 1, 4, 5, 6, 3].

This methodical technique enabled us to investigate how the model's accuracy changes with increasing amounts of training data, revealing insights into its learning and generalization skills.

4 | Results

In the upcoming results chapter, we present the outcomes of our study, spanning the selection of normalization methods and time window choices with the rigorous data division technique employed. We highlight the effectiveness of convolutional neural networks (CNNs) with varying filter sizes, the influence of hyperparameters and asses comparisons between the two variants explored. These findings set the stage for experiments exploring transfer learning under different data scenarios.

4.1 Normalization types comparison

To ascertain the optimal normalization methods, experiments were carried out employing the model with the predefined initial parameters as detailed in the "Methods" section. Proper normalization of data is a critical step in training robust machine learning models, especially in domains like gesture recognition.

For the testing phase, Database 2 (DB2) was chosen, given its substantial number of subjects. An intermediate window of 150 ms (300 points) was employed for the analysis.
The results from five runs with the setup introduced in the same section showed superior generalization capabilities when employing channel-specific normalization mode, as recommended in [43] and [44], as depicted in **Table 4.1**. For what concern the normalization types, the z-score method was the one which demonstrated superior accuracy and slightly faster convergence rates compared to the others [45].

Hereafter, all experiments will incorporate z-score normalization with the 'by channel' modality.

Table 4.1: Performance Metrics for Different Normalization Modes and Types

Norm Mode	**Norm Type**	**Accuracy**	**Kappa**	**F1 Score**
Subject	(0 - 1)	0.6988	0.7656	0.6988
Channel	(0 - 1)	0.7192	0.7729	0.7192
Subject	(-1 - 1)	0.6794	0.7362	0.6794
Channel	(-1 - 1)	0.7086	0.7638	0.7086
Subject	Z-score	0.7040	0.7647	0.7040
Channel	Z-score	**0.7308**	**0.7873**	**0.7308**

4.2 Window comparison

Selecting the appropriate window size for data processing is a crucial aspect of signal-based tasks such as hand gesture recognition. The window size determines how the data is segmented and presented to the model, directly affecting its ability to capture temporal information and patterns. We employed again the same fixed parameters as before. Moreover, the model underwent the normalization steps outlined above. **Table 4.2** presents the results of these experiments, averaged across five trials.

As can be observed, the first three windows tested had no discernible change. Particularly in terms of Kappa, which considers the presence of an unbalanced dataset. One plausible explanation for this phenomenon could be attributed to the increased volume of data associated with smaller windows. In fact, when using the 200-point window, the model benefits from a greater number of training samples

All measures show a significant difference for the 20 ms span. However, when majority vote is used, it remains a feasible option. Finally, the window size was determined at an intermediate value of 300 points (corresponding to 150 ms), as it is the window size suggested by several studies [46].

Except for DB1, which, as previously stated, would utilize a fixed 20 point dataframe, all of the experiments shown will now use the dataframe with 300 point window samples.

Table 4.2: Window selection Results

Window Lenght	Best Val Loss	Accuracy	Kappa	F1 Score	Best Epoch
200	0.9760	0.7340	0.7951	**0.7399**	62
300	**0.9710**	**0.7366**	**0.7979**	0.7328	51
400	0.9885	0.7319	0.7891	0.7380	56
40	1.3404	0.5993	0.6890	0.6073	36

4.3 Grid Search for parameters optimization

The model was fully parameterized, maintaining the consistency of the data flow to determine the best combination of the remaining parameters. The same experimental setup, which consists of splitting training, validation, and test over repetitions, was applied. Subsequently, a comprehensive grid search was performed. The model was fully parameterized while retaining data flow consistency to discover the optimum combination of the remaining parameters. Subsequently, a comprehensive grid search with 50 alternative models was performed to assess the effectiveness of the hyperparameters stated in **Table 4.3**.

Table 4.3: Overview of the grid search parameters tested

Hyperparameter	**Values chosen**
batch size	64 128 256 512
optimizer	SGD, Adam
learning rate	1.47 | 2.37 | 3.40 | 6.77 | 15.5 | 34.2 | 60.1 | 82.2) $\times 10^{-4}$
activation function	ReLu, LeakyReLU, ELU
pooling type	MaxPooling, AveragePooling
N_multik	16, 32, 64, 128
N_Conv_conc	64, 128, 256
N_SepConv	64, 128, 256
Kernel_Multi_Dim	[10 - 50], [20 - 60], [30 - 70]

Note: This table presents the hyperparameters explored in the grid search. The keys *N_multik, N_Conv_conc, and N_SepConv* represent the number of neurons in the hidden layers of parts 1, 2, and 3, respectively. Notably, The Block 2 of Part 1 consistently uses twice as many hidden neurons as Block 1. In the table, *Kernel_Multi_Dim* represents the configuration for 5 parallel convolution operations. Each operation uses a fixed second dimension of 3, while the first dimension varies from the lower value to the upper value in increments of 10.

To ensure a comprehensive parameter analysis that remains valid across multiple databases, the grid search was performed on the unified dataset created by merging DB2, DB3 and DB7.

Figure 4.1 shows the results of the first 20 models, ordered by validation loss. Following a comprehensive evaluation, the parameters of model number 6 was chosen for additional testing, since they demonstrated a significant discrepancy across all metrics collected.
The Learning Rate column has been multiplied by 10^4 for visualization purposes.

N_model	Val Loss	Accuracy	Kappa	F1_score	Epoch	N_Params	N_multik	N_Conc	N_SepConv	Ker_multi_dim	Kern_Conc	Act_func	Batch_size	Learn Rate
6	1.022	0.748	0.802	0.748	41	238542	32	128	256	[[10 3] - [50 3]]	1	ReLU()	256	0.7588
5	1.052	0.729	0.783	0.729	35	288399	64	64	256	[[30 3] - [70 3]]	1	PReLU(num	512	1.6593
4	1.066	0.710	0.774	0.710	52	246414	64	128	128	[[10 3] - [50 3]]	1	ReLU()	256	0.7588
17	1.068	0.710	0.774	0.710	69	196622	64	64	128	[[30 3] - [70 3]]	1	LeakyReLU(	256	1.5030
1	1.078	0.706	0.756	0.706	64	265774	16	128	128	[[20 3] - [60 3]]	3	ReLU()	128	0.7588
15	1.081	0.714	0.779	0.714	60	475854	32	128	128	[[30 3] - [70 3]]	3	ReLU()	512	0.7977
3	1.085	0.722	0.785	0.722	28	1675919	64	256	128	[[50 3] - [90 3]]	3	PReLU(num	512	0.9793
14	1.086	0.719	0.774	0.719	36	901774	64	128	128	[[30 3] - [70 3]]	3	ReLU()	256	1.7135
10	1.093	0.704	0.765	0.704	69	292302	32	64	128	[[20 3] - [60 3]]	3	LeakyReLU(	512	1.0451
18	1.097	0.701	0.763	0.701	66	157774	32	128	128	[[50 3] - [90 3]]	1	LeakyReLU(	512	0.7710
37	1.098	0.704	0.763	0.704	64	292302	32	64	128	[[50 3] - [90 3]]	3	LeakyReLU(	512	1.0451
36	1.100	0.689	0.750	0.689	67	109262	32	64	128	[[10 3] - [50 3]]	1	ReLU()	256	0.7710
2	1.101	0.722	0.774	0.722	14	1792270	64	256	256	[[20 3] - [60 3]]	3	ReLU()	256	0.9793
28	1.115	0.693	0.754	0.693	66	447758	32	128	64	[[20 3] - [60 3]]	3	ReLU()	256	1.4704
33	1.127	0.686	0.746	0.686	68	174382	16	64	128	[[50 3] - [90 3]]	3	ReLU()	512	1.4704
39	1.132	0.678	0.736	0.678	67	87662	16	64	128	[[30 3] - [70 3]]	1	LeakyReLU(	256	1.7105
22	1.132	0.668	0.744	0.668	53	228079	16	128	64	[[10 3] - [50 3]]	3	PReLU(num	256	1.4704
20	1.135	0.699	0.766	0.699	67	428558	32	128	64	[[20 3] - [60 3]]	3	ELU(alpha=	512	0.7977
30	1.135	0.669	0.749	0.669	69	189518	64	128	64	[[10 3] - [50 3]]	1	LeakyReLU(	512	1.7105

Figure 4.1: Grid Search Results over the merged dataset

4.3.1 Results of standard fine-tuned model

Figures 4.2, 4.3, 4.4, 4.5 and **4.6** depict the performance of the standard model, with the parameters selected in the previous section, over the various databases. It is clear how DB3, composed exclusively by Amputee, heavily degrades the model's capability.

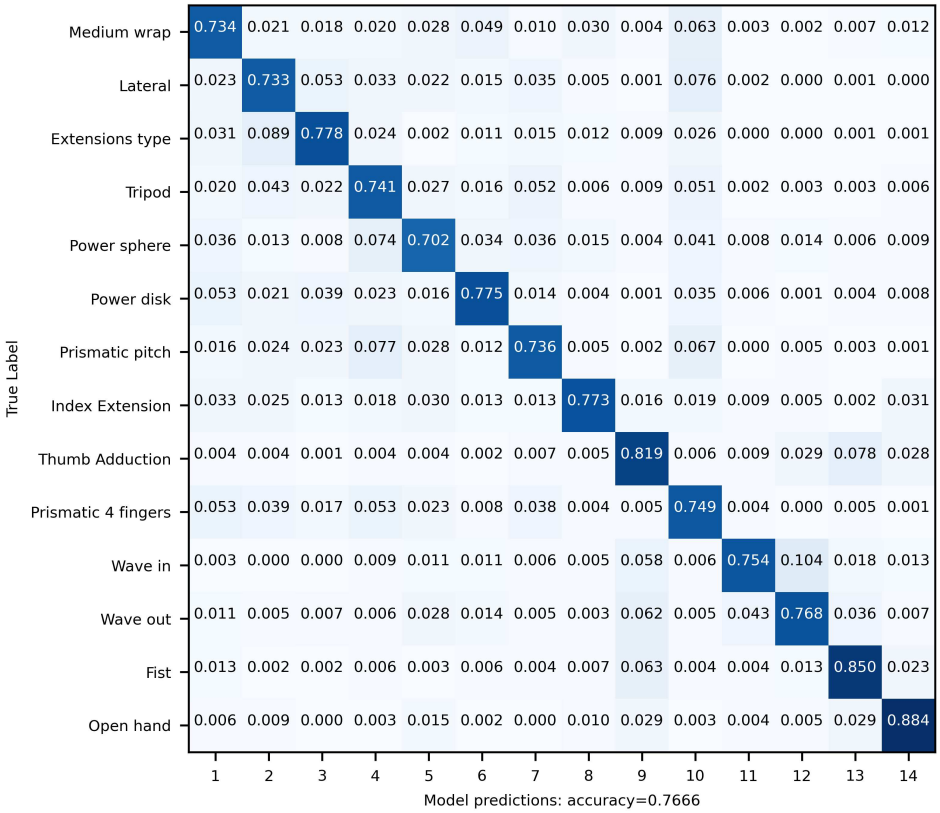

Figure 4.2: Confusion Matrix for DB2

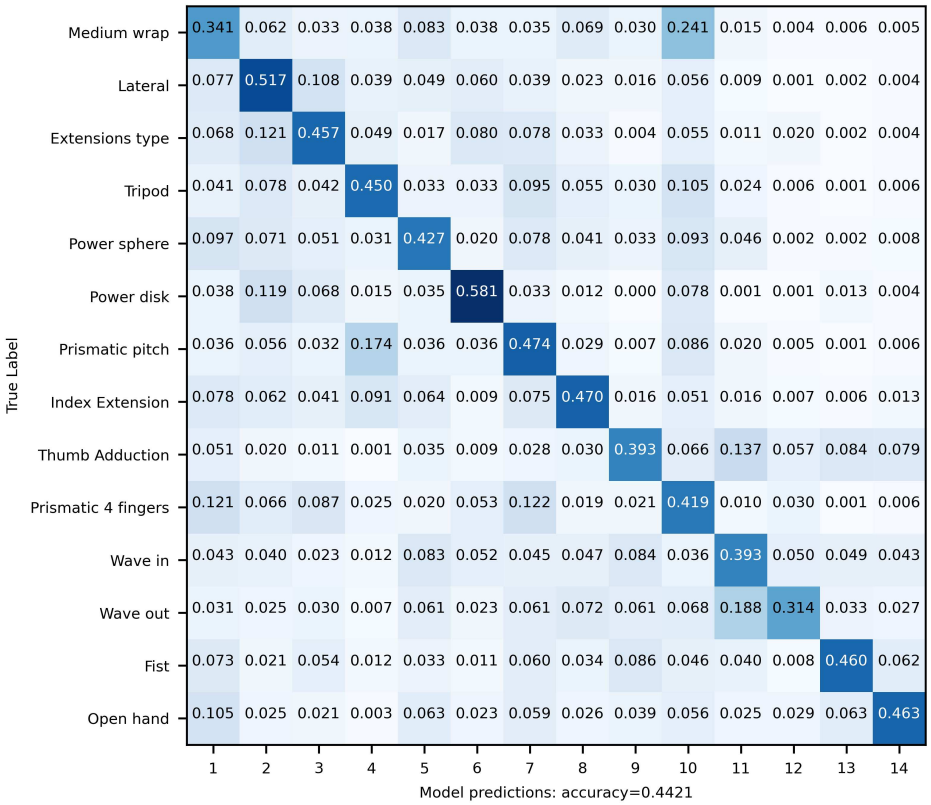

Figure 4.3: Confusion Matrix for DB3

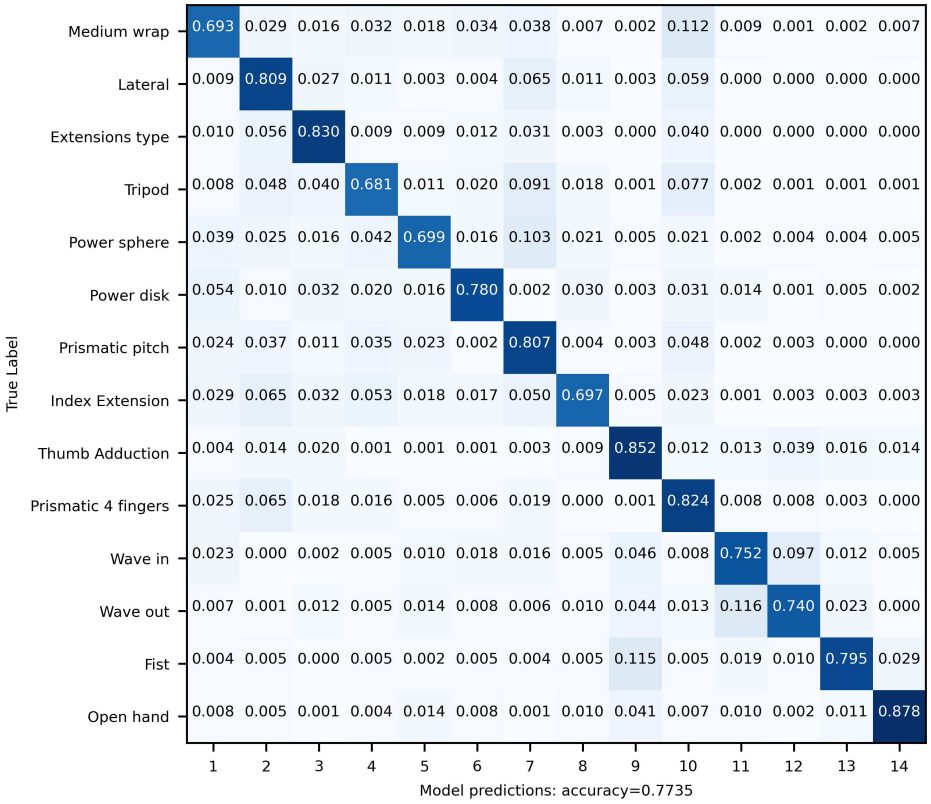

Figure 4.4: Confusion Matrix for DB7

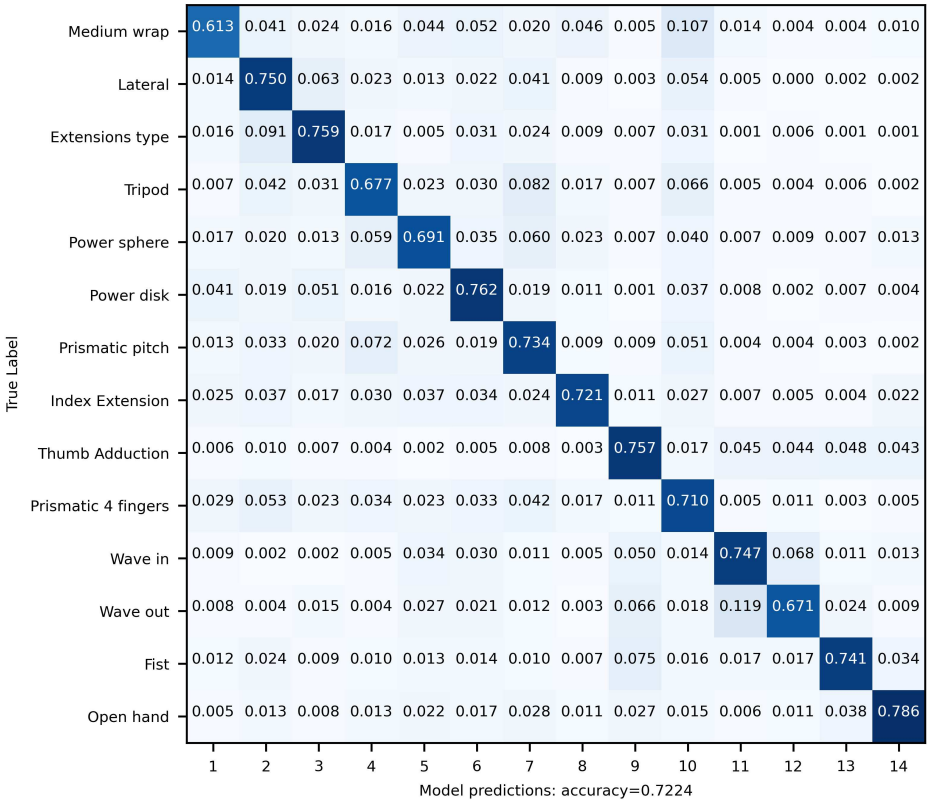

Figure 4.5: Confusion Matrix for DB2+DB3+DB7

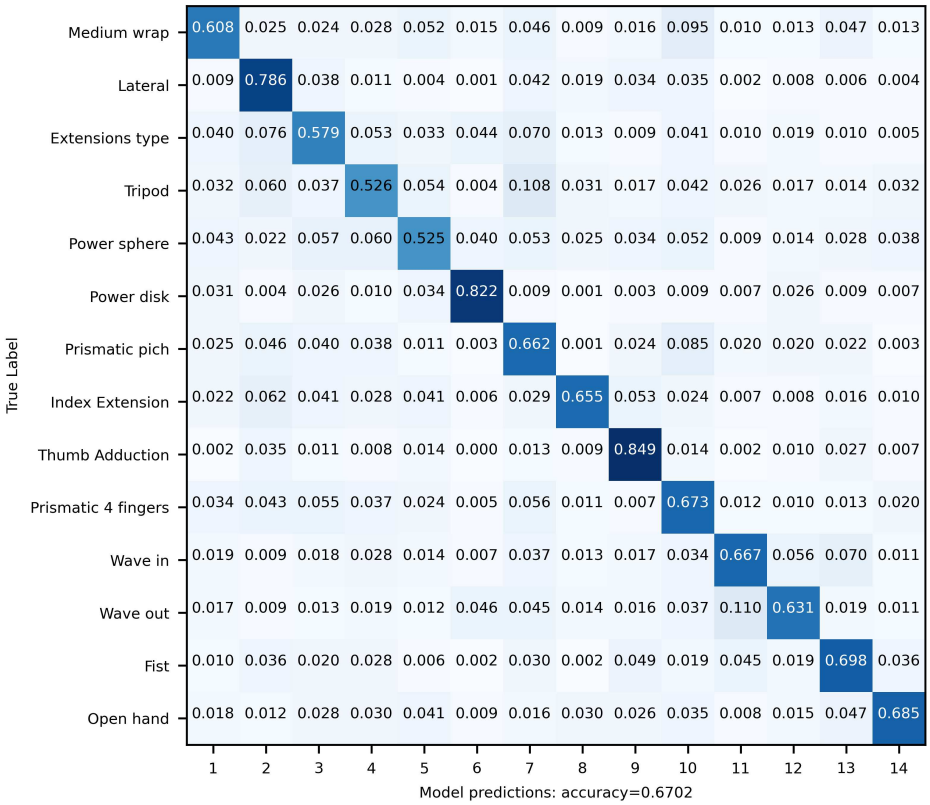

Figure 4.6: Confusion Matrix for DB1

4.4 Reversal Gradient

This section presents the outcomes of the domain adaptation implementation. To streamline the presentation, the emphasis will be placed on the confusion matrix and plots related to the amalgamated database (DB2+DB3+DB7). While larger datasets often lead to enhanced performance, the presence of individuals with amputations introduces a noticeable bias. **Figure 4.7** shows the plot of the domain classifier loss. First, the particular pattern of the curve match the expectations of using incremental gradient multiplier for the domain classifier loss as described in "Method" section. Indeed, in the first epochs, the classifier is learning to accurately predict from which patient the signals come from; then, when the multiplier starts to rise, penalizing this ability, it starts to increase the loss.

Furthermore, another significant consideration pertains to the observation that the validation loss consistently remains below the training loss curve. This phenomenon can be elucidated by the weight update process: validation data follow training data and the corresponding updates. As a result, with each iteration, there is a progressive divergence from the classifier's capability to accurately identify patients, leading to an increase in the training loss when compared to the validation loss.

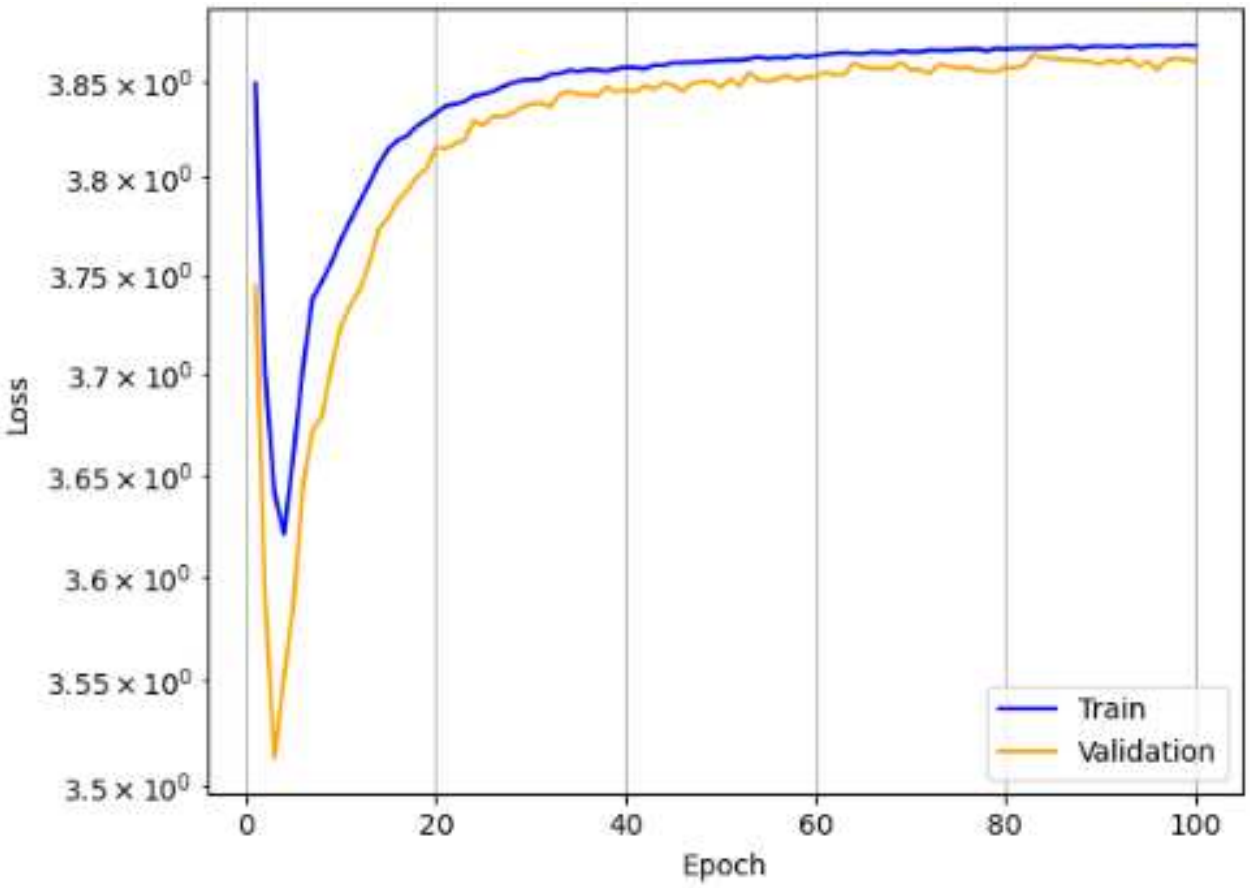

Figure 4.7: Domain Classifier loss for DB2+DB3+DB7 Dataset

In **Figure 4.8** and **Figure 4.9** are shown the plot of the task loss and the confusion matrix respectively.

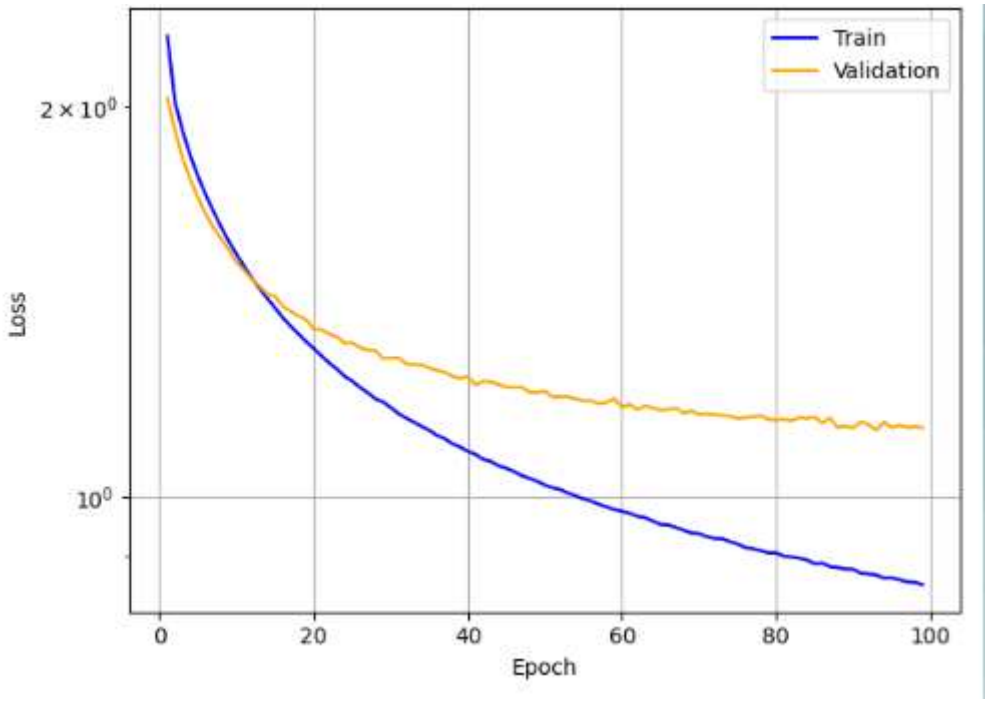

Figure 4.8: Task Loss only for DB2+DB3+DB7 Dataset with Reversal Gradient

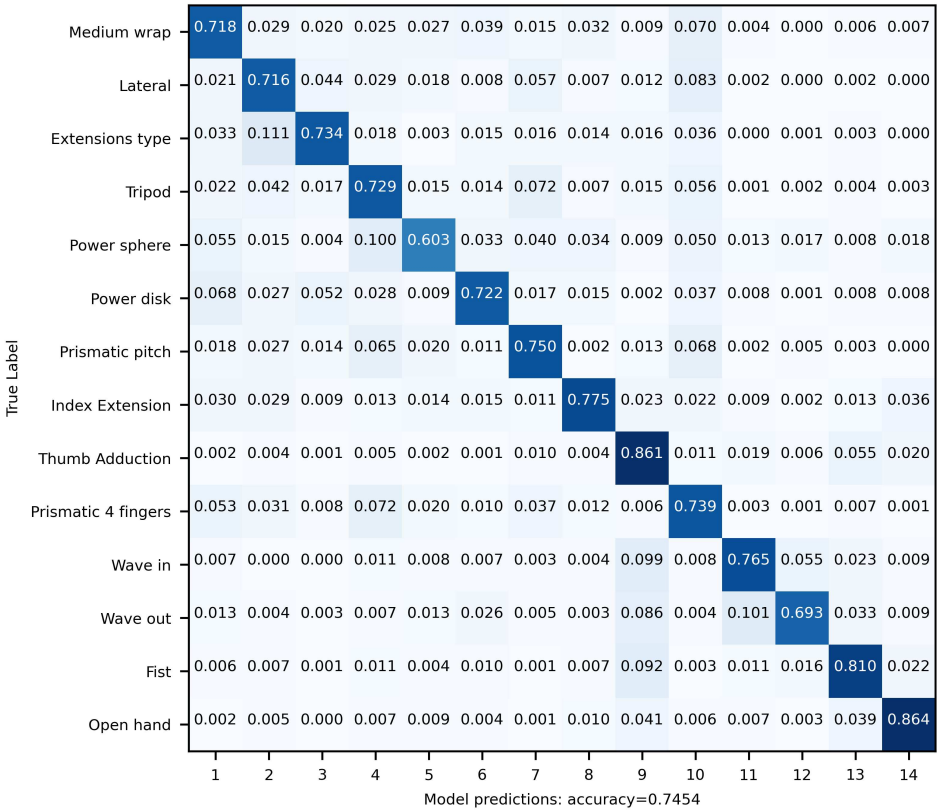

Figure 4.9: Confusion Matrix for DB2+DB3+DB7

4.5 Triplet Margin Loss

In this section, akin to the previous one, the focus will solely be on the presentation of results for the consolidated database in relation to the Triplet Margin Loss. **Figure 4.10** displays the loss plot of training and validation. In **Figure 4.11** and **Figure 4.12**the plots of the task loss exclusively and the confusion matrix, respectively, are available for observation.

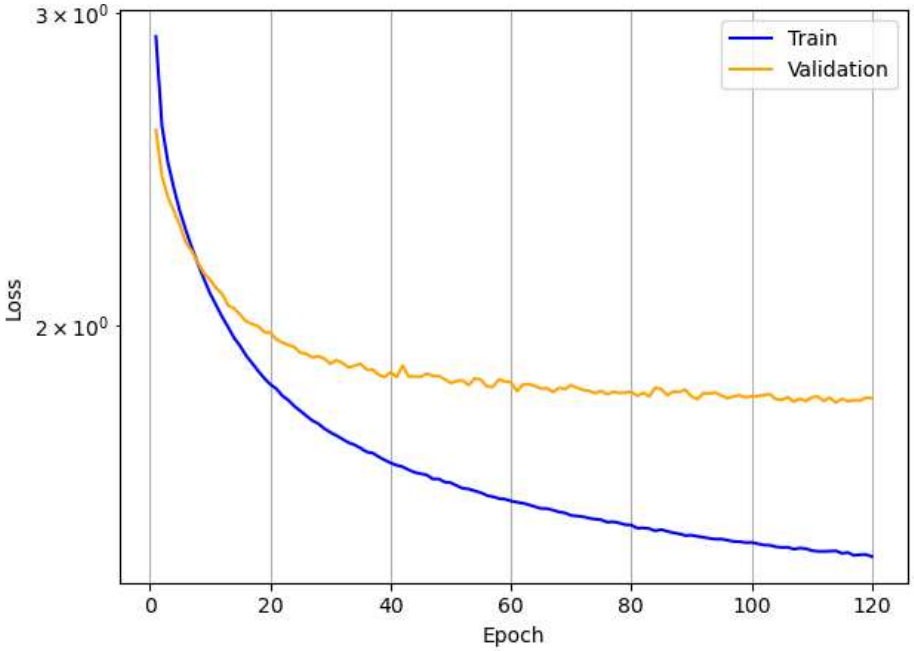

Figure 4.10: Triplet Margin loss for DB2+DB3+DB7 Dataset

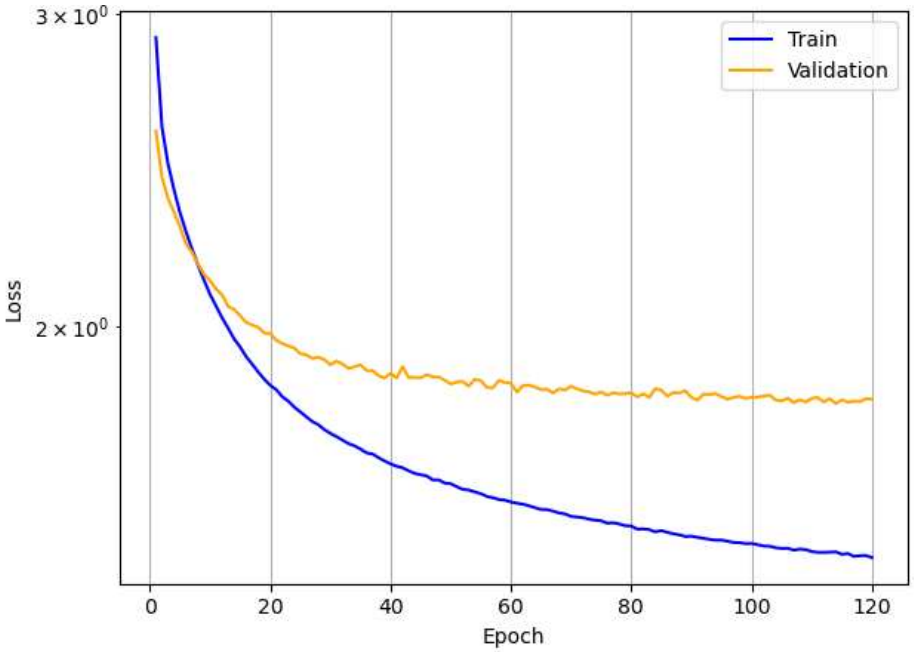

Figure 4.11: Task loss only for DB2+DB3+DB7 Dataset with Triplet Margin Loss

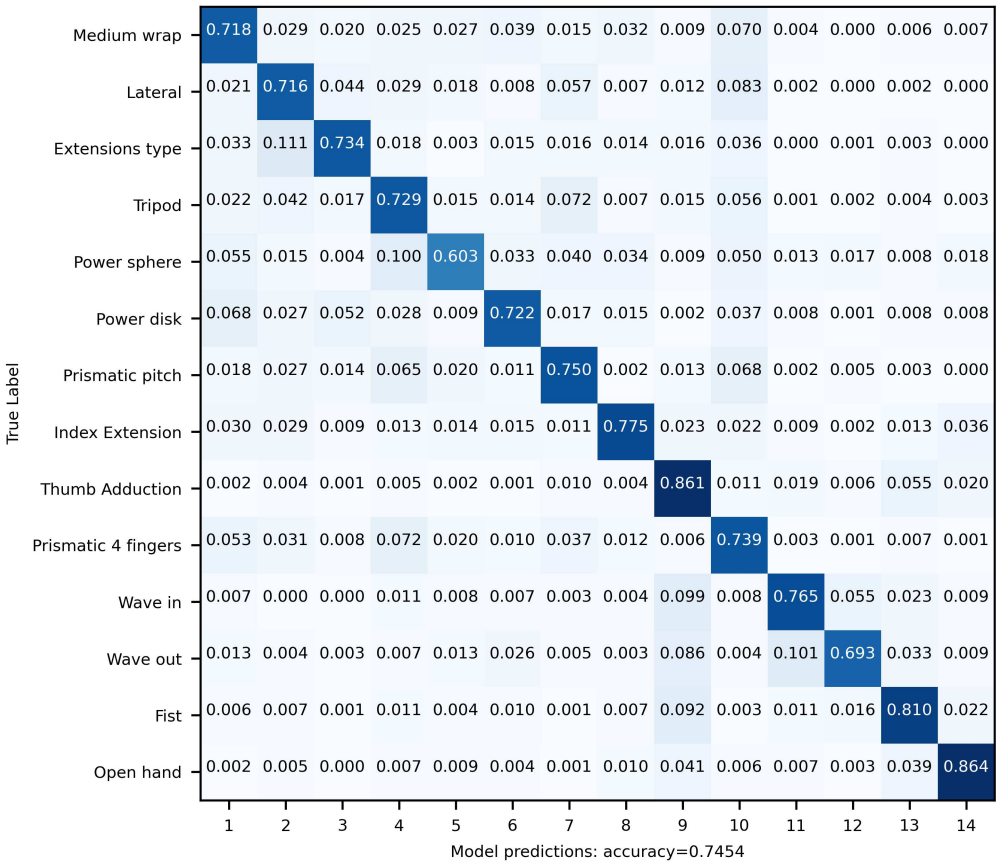

Figure 4.12: Confusion Matrix for DB2+DB3+DB7

4.6 Transfer learning

Patients who were previously excluded from the experiment are now included in the study for user-specific fine-tuning using Transfer Learning. This procedure involves retraining a part of the network to improve its adaption to a new task, specifically for a certain subject. As described in "Methods" section, for each of the three prediction algorithms used, two experiment were conducted:

- Experiment 1: retraining only the final two linear layers (Part 4).
- Experiment 2: retraining Parts 2 and 3 in addition to the final two linear layers (Part 4).

The initial model encompasses a total of 238,542 trainable parameters. In the first experiment, only 34,702 parameters remain trainable, whereas in the second experiment, a total of 111,374 parameters are unfrozen. The selected metric for visualization is Cohen's Kappa (Kappa), which proves to be a more informative choice, particularly when dealing with imbalanced datasets.
The results presented in this section leverage the model pre-trained in the merged database, which incorporates more data while providing a more objective overview.

4.6.1 Pre-trained Standard Model

The outcomes of the first experiment, averaged across the 11 patients tested, including two amputees, are presented in **Figure 4.13**.

The results of the second experiment are shown, instead, in **Figure 4.14**

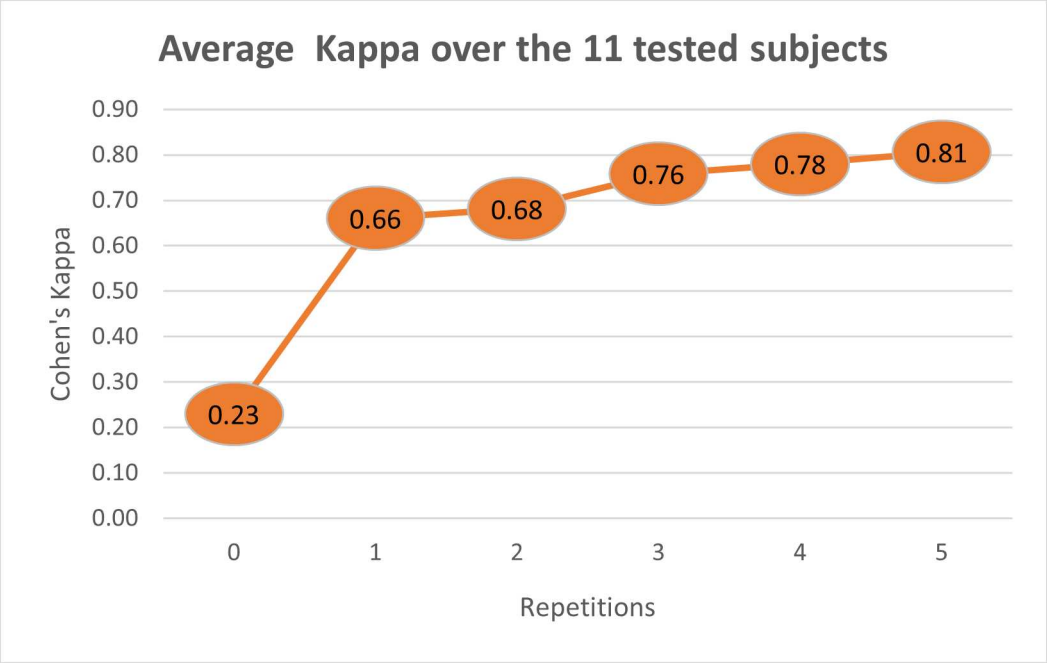

Figure 4.13: Variation in performance for experiment 1

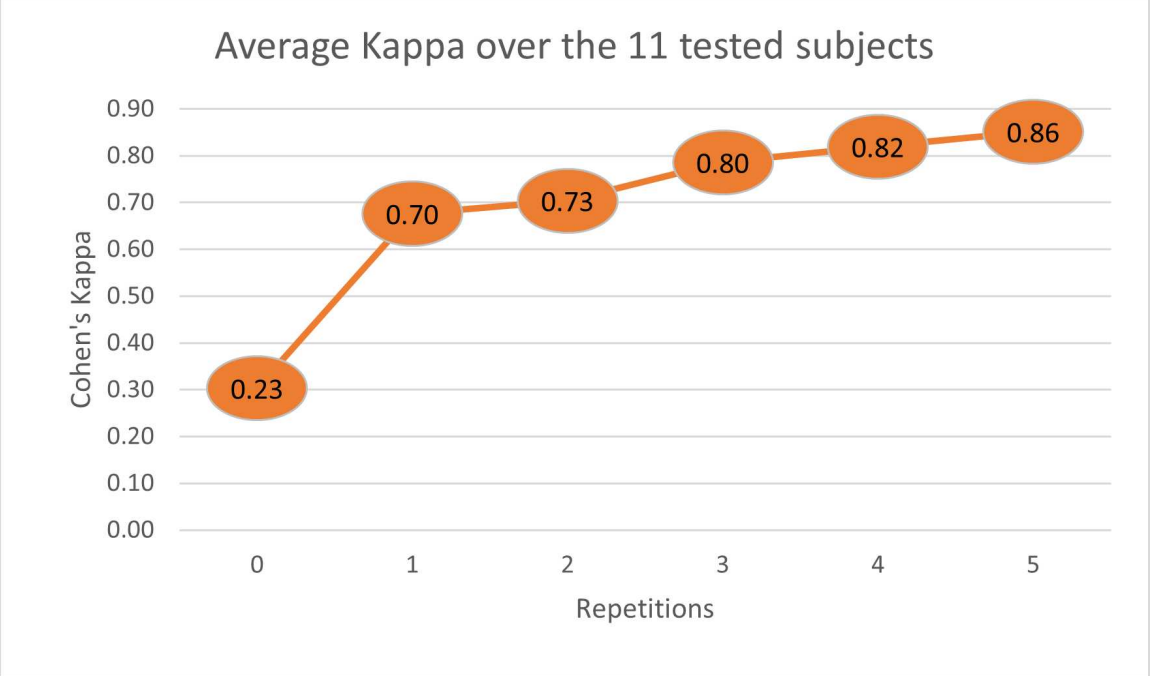

Figure 4.14: Variation in performance for experiment 2

4.6.2 Pre-trained with Reversal Gradient

The two results displayed in **Figure 4.15** and **4.16** represents the effect of the user-dependent fine-tuning, respectively for experiment 1 and experiment 2 for the model trained with Reversal Gradient algorithm.

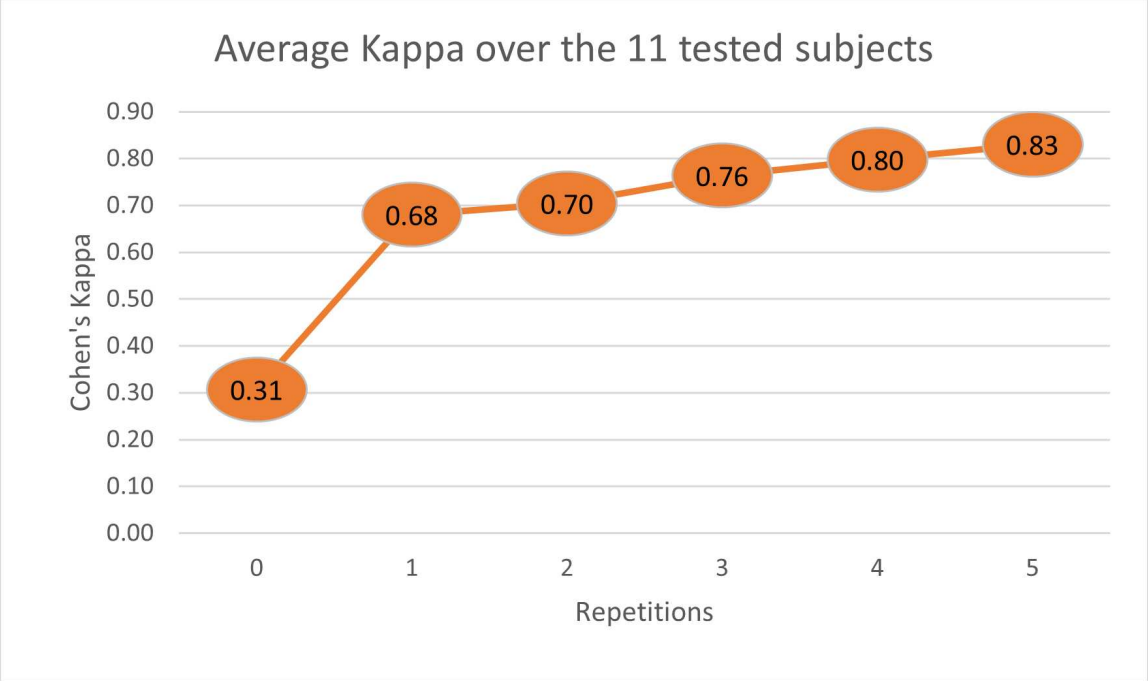

Figure 4.15: Variation in performance for experiment 1

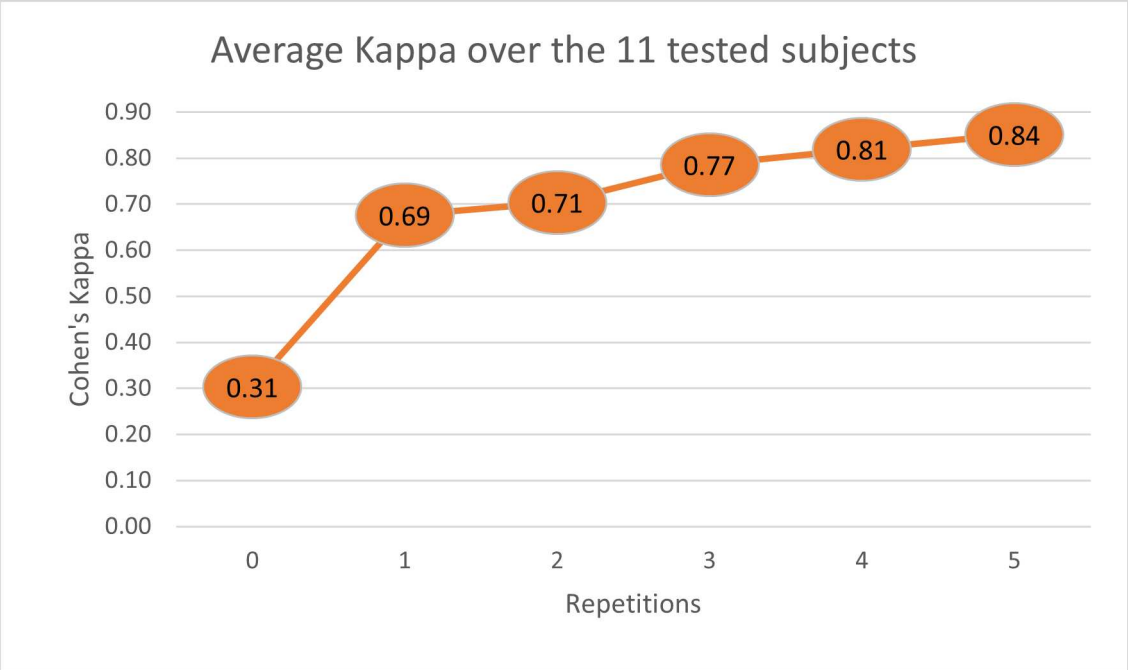

Figure 4.16: Variation in performance for experiment 2

4.6.3 Pre-trained with Triplet Margin Loss

In the following, in **Figure** 4.17 and **Figure 4.18**, it is possible to observe the results for the last tested algorithm, the Triplet Margin Loss.

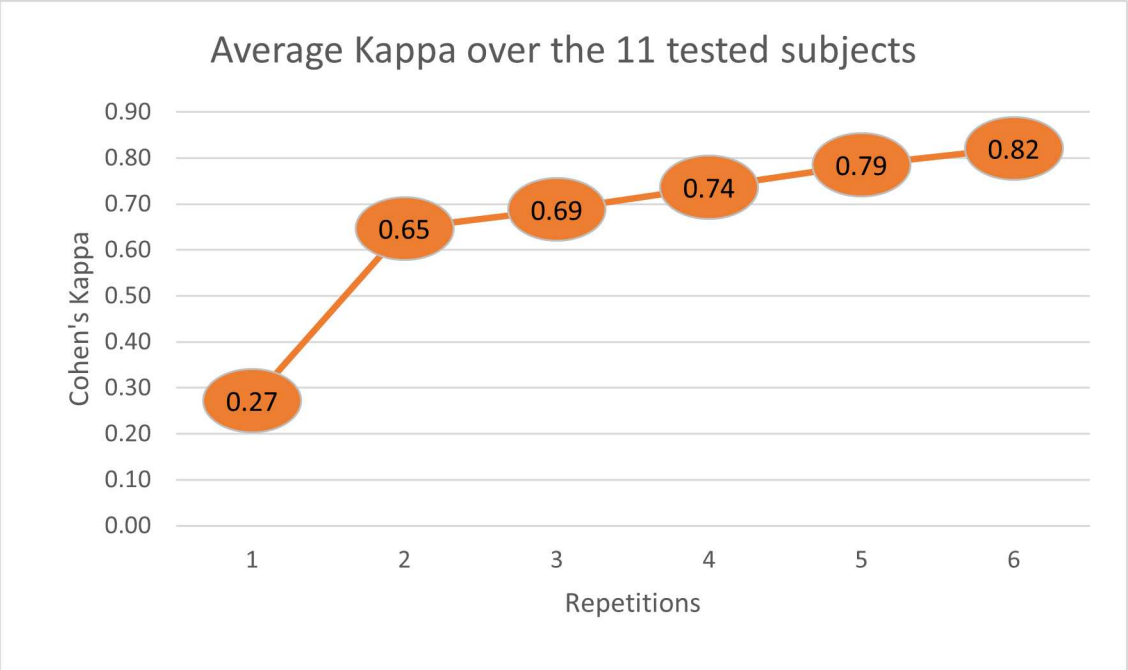

Figure 4.17: Variation in performance for experiment 1

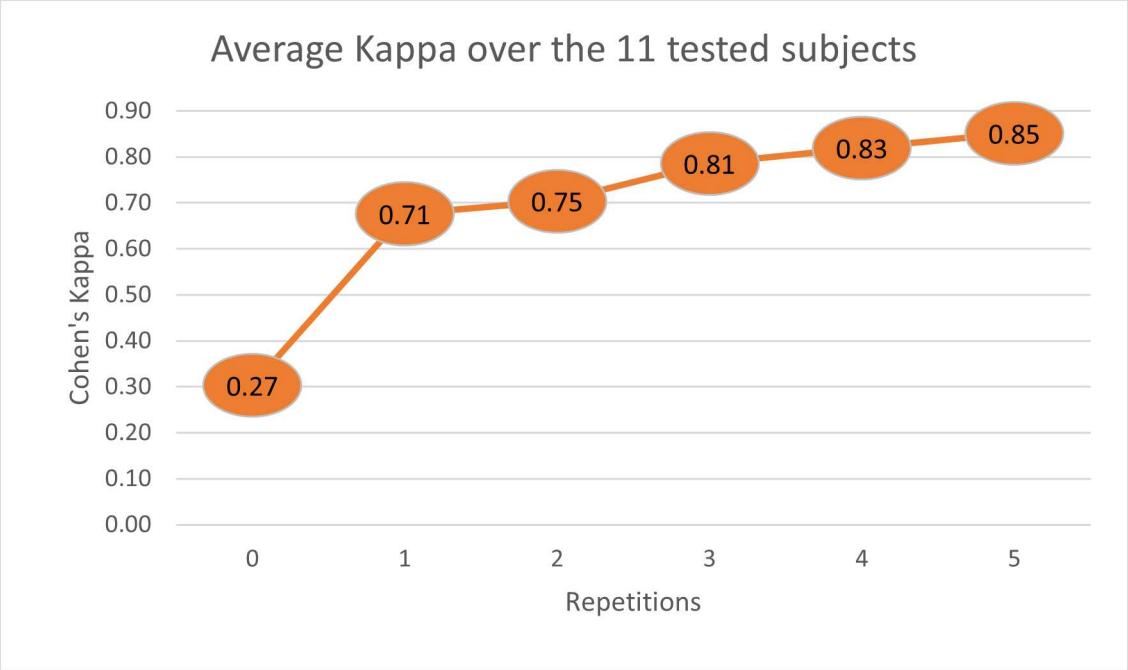

Figure 4.18: Variation in performance for experiment 2

5 | Discussion

The development and evaluation of any machine learning model are inevitably accompanied by a set of limitations and challenges that provide valuable insights for future improvements and research directions. In this section, various key limitations encountered during the design and analysis of the model are discussed.

5.1 Constrains and limitation of the base model

One primary constraint is the deliberate use of a low number of parameters. This choice was driven by both the real-time constrains and the desire to mitigate overfitting, a common issue in deep learning models. However, this limitation also restricts the model's capacity to capture intricate data relationships, potentially affecting its overall performance.

Another significant challenge arises from the substantial variability observed between patients and even within different sessions of the same person. This high inter-subject variability presents a formidable obstacle to achieving robust performance.

Furthermore, it's essential to emphasize that the experimental protocol utilized in this study does not involve model calibration for individual testing sessions, which has a substantial impact on the model's classification accuracy. This influence arises from variations in data distributions attributable to factors such as skin resistance, humidity, and electrode displacement, which are session-specific.
In this sense, the data generated offline closely resembles that of a real-time trial, without any prior operations before using the prosthesis.

Even using generalization methods such as triplet margin loss and a small number of parameters, the model demonstrates a tendency to overfit the data. Significant unpredictability and variations in data distributions among patients and sessions increase this predisposition.

5.1.1 Database considerations

In **Table 5.1** are summarized the performance of the fine-tuned standard model with respect to the different databases.

Database 1 (DB1) introduces an additional layer of complexity due to its low-frequency data. This characteristic can hinder the model's ability to accurately classify gestures, emphasizing the need for specialized signal processing techniques or architectures tailored to low-frequency data.

Database	Accuracy	Kappa	F1 Score
DB1	0.67	0.724	0.671
DB2	0.767	0.822	0.768
DB3	0.442	0.501	0.446
DB7	**0.774**	0.826	0.779
DB2+DB3+DB7	0.722	0.774	0.722

Table 5.1: Performance Metrics for Different Databases

The requirement for user-adaptive control of prosthetic devices became clear when considering the significant decline in performance metrics reported in DB3, which is primarily due to the presence of only amputee patients. These users pose unique challenges in gesture recognition due to differences in their physiological signals, making user-adaptive models a critical area of exploration for improved performance.

Furthermore, a shift in performance dynamics is observed, particularly for databases 2 and 7. Notably, database 7 stands out with the highest accuracy, even though it includes two subjects without arms. This phenomenon may be attributed to the fact that when using only half of the subjects, the model tends to overfit on them, resulting in enhanced performance on the included subjects.

Conversely, when considering the composite database, its performance naturally experiences a decline due to the inclusion of database 3. Nevertheless, a satisfactory level of performance is still maintained, probably thanks to leveraging a larger volume of data during training, which usually lead to the development of more robust models.

5.2 Transfer Learning comparisons

This section investigates the impact of various algorithms within the context of the transfer learning framework. The objective is to elucidate the effects of these algorithms and offer insights into the results they produce. **Table 5.2** summarizes the results over the merged database reported in section "Results."

One prominent observation is the notably low accuracy achieved by the model when utilizing 0 repetitions. It's important to note that the subjects tested were entirely distinct from those used for model training, making this column a cross-subject experiment. These results serve as a clear indication of the substantial variability present in the signals recorded from different subjects.

In this column the performance for both experiments remains identical since the model hasn't undergone retraining at this stage. However, it's evident that in this scenario, the implementation of the Domain Adversarial Neural Network with reversal gradient, followed by the triplet margin loss, exhibits a greater capability to generalize information across a broader range of subjects compared to the standard model.

Overall, when considering the re-training of only the last two linear layers (experiment 1), it becomes apparent that the DANN model with reversal gradient achieved the highest performance. This observation suggests that learning invariant features concerning session-related factors, such as repetitions, facilitated the

Table 5.2: Cohen's Kappa for the different algorithms and experiments in the Transfer Learning Framework, varying number of repetitions

	Number of repetitions used					
Pre-training type	**0**	**1**	**2**	**3**	**4**	**5**
Standard 1	0.23	0.66	0.68	0.76	0.78	0.81
DANN 1	**0.31**	0.68	0.7	0.76	0.8	**0.83**
Triplet 1	0.27	0.65	0.69	0.74	0.79	0.82
Standard 2	0.23	0.7	0.73	0.8	0.82	**0.86**
DANN 2	0.31	0.69	0.71	0.77	0.81	0.84
Triplet 2	0.27	0.71	0.75	0.81	0.83	0.85

Note: In the table, "DANN 1" represents the utilization of Domain Adversarial Neural Network, with the subsequent number denoting the specific fine-tuned experiment type.

fine-tuning process.

Now, let's shift the focus to experiment number two. In this scenario, DANN exhibited the poorest performance. To interpret this outcome, it's important to recall the information detailed in the "Methods" section. Specifically, the domain classifier is connected to the output of part 3, which occurs before the linear layers.

Unlike in experiment 1, in this case, the model is re-training not only the parameters of the last layers but also those of part 2 and part 3, all of which were subjected to the domain classifier's loss, implemented via reversal gradient. Consequently, during fine-tuning, the network might face more challenges in moving towards the optimal direction, in contrast to the behavior of the standard model or triplet margin loss.

5.3 Future improvement

Integration with virtual reality and gamification techniques opens up exciting avenues for dynamic hand gesture recognition. This combination has the potential to bring in a continuum of learning processes for models, improving adaptability and real-time interaction [47].
Furthermore, the adoption of 3D motion capture cameras offers the potential to shift from classification to regression tasks. This transition could introduce fluidity and precision to prosthetic devices, improving their functionality and user experience.

5.3.1 Major Voting Strategy for 20 ms Window

Interesting would be the possibility of implementing the major voting strategy for the 20 ms window, as it is a common practice in this field [48] In fact, although the accuracy of the standard model, without transfer learning or parameter tuning, is only about 60 percent, with this strategy and some considerations one could arrive at very good results [49]. Consider, for example, having the prosthesis only perform

a motion if there are at least 4 out of 5 correct classifications in a group. For this example, 4/5 consecutive correct classifications are considered due to the low accuracy of the classifier. Indeed, expecting the classifier to correctly classify 5 consecutive results in a row could be cumbersome for the user ($0.60^5 \approx 0.08$), potentially leading to muscle fatigue, increased burden and prosthesis abandonment. Alternatively, different criteria could be chosen, such as requiring 8/10 consecutive correct classifications. With this in mind, it is possible to use the binomial probability formula to calculate the probability of achieving exactly k successes in n trials:

$$P(X = k) = \binom{n}{k} \cdot p^k \cdot (1-p)^{n-k} \tag{5.1}$$

Where:

- $P(X = k)$ is the probability of getting exactly k successes in n trials.
- $\binom{n}{k}$ represents the binomial coefficient, calculated as $\frac{n!}{k!(n-k)!}$, which accounts for the number of ways to choose k successes from n trials.
- p is the probability of success in a single trial.
- k is the number of desired successes.
- $(1-p)$ is the probability of failure in a single trial.
- n is the total number of trials.

To calculate the probability of getting exactly 4 correct out of 5:

$$P(X = 4) = \binom{5}{4} \cdot (0.6^4) \cdot (0.4^1) \approx 0.2592 \tag{5.2}$$

To calculate the probability of getting all 5 correct:

$$P(X = 5) = (0.6^5) \approx 0.07776 \tag{5.3}$$

Finally, to calculate the probability of having at least 4 out of 5 correct:

$$P(\text{At least 4 correct out of 5}) \approx 0.2592 + 0.3888 \approx 0.3888 \tag{5.4}$$

This scenario can be summarized with a single equation, where the consideration of whether the 5th classification is correct or not is omitted:

$$P(X = 4 or X = 5) = \binom{5}{4} \cdot (0.6^4) \times 1 \approx 0.65 \tag{5.5}$$

In a similar way, it is possible calculate the probability of not classifying something that was actually correct:

$$P(X = 4) + P(X = 5) = \binom{5}{4} \cdot (0.4^4) \times 1 \approx 0.13 \tag{5.6}$$

The probability of making an incorrect movement, specifically selecting the same wrong class four times out of five, under the strong assumption that all wrong classes have an equal probability of misclassification, is as follows:

$$P(Y = 4) + P(Y = 5) = \binom{5}{4} \cdot 0.4 \times (0.4 \times 1/13)^3 \times 1 \approx 0.00 \tag{5.7}$$

Where Y in this case refers to the same wrong class and:

- First term represents the random position in which the 5th classes can be picked
- Second term refers to the probability of random wrong class
- Third term refers to probability of picking, between 13 classes (e.g. the wrong ones) the same one

Therefore, with at least four out of five correct classifications as threshold the majority voting strategy enhances accuracy by approximately 5%, significantly reducing the likelihood of performing the wrong task and thereby increasing its reliability. To better explore this option would be great for future improvements.

6 | Conclusion

In conclusion, this thesis project provides valuable insights into various techniques applicable in the field of sEMGs, particularly for prosthetic applications. It delves into the meticulous selection of normalization types and modes, along with considerations regarding time window choices. The establishment of a formal and replicable data division facilitates the creation of a comparative protocol across different techniques.

Furthermore, this research underscores the potential of employing convolutional neural networks with filters of varying sizes to capture intricate relationships among different frequency components, while also exploring the impact of hyperparameters on model performance. Additionally, it assesses the effectiveness of two distinct algorithms, aiming to enhance the model's adaptability to new subjects and extract universal relationships whenever possible. In addition, the research encompasses two distinct experiments that utilize transfer learning as a method for assessing the system's adaptability to new users. These experiments delve into the effectiveness of transfer learning techniques when faced with varying amounts of data, such as different numbers of sessions.

A | Appendix

A.1 Deep Learning

Deep Learning (DL) approaches represent relatively recent techniques that offer an end-to-end solution for developing predictive models or classifiers. Unlike traditional methods, DL models do not rely on the manual evaluation and extraction of hand-crafted features. However, they are often criticized for their lack of interpretability. Nevertheless, the growing interest in DL arises from its ability to surpass classical machine learning techniques in certain scenarios.

Deep neural networks function by autonomously identifying significant patterns or features within data during the training phase. This process involves learning the weights that characterize and define the layers, regardless of their type. These learned weights determine the activation maps (also known as feature maps) within each layer, generated from input data. The training of these weights relies on back-propagation, a fundamental concept in neural network training.

The training process of a neural network is organized into epochs. It commences by randomly initializing the weights in each layer. Within each epoch, the training data is divided into smaller data batches. For each batch, the network performs classification or regression tasks and subsequently compares the results with the true labels or values using specific loss functions. These loss functions vary according to the nature of the task at hand. In classification tasks, two commonly used loss functions are:

- Binary Cross Entropy: it is used if the network has to solve a binary problem:

$$L = -\frac{1}{N}\sum_{i=1}^{N}\left(y_i \log(\hat{y}_i) + (1-y_i)\log(1-\hat{y}_i)\right) \tag{A.1}$$

  Where y_i is the label referred to the ith class, and can be 1 or 0. $\hat{y}_i$ is the probability outputted by the classifier for the ith class.

- Categorical Cross Entropy: it is employed for multi-class classification problems:

$$L = -\sum_{i=1}^{N} y_i \log(p(\hat{y}_i)) \tag{A.2}$$

  Where y_i is the ground truth and $\hat{y}_i$ is the score for the ith class outputted by the network.

The loss assessment measures how much the network is making errors. To upgrade the weights, what is exploited in particular is the evaluation of the gradient

of the loss, which is the vector of partial derivatives with respect to all coordinates of the weights:

$$\nabla W_L(W) = \left[\frac{\partial L}{\partial W_1}, \frac{\partial L}{\partial W_2}, ..., \frac{\partial L}{\partial W_m}\right]^T \tag{A.3}$$

If $\nabla W_L(W) = 0$, it means that a local optimum minimum is reached, which is the situation desired. In order to reach the optimal minimum, it is necessary to move in the direction $-\nabla W_L(W)$. In particular, each time a batch of data is predicted, the weights are updated according to this formula:

$$W_t + 1 = W_t - \alpha \nabla W_L(W)$$

Where α is called the learning rate and determines how strongly the movement in the direction $-\nabla W_L(W)$ should be. When $\nabla W_L(W)$ is evaluated over a batch of data, it is called Stochastic Gradient Descent ($g(t)$).

There are several ways of updating weights:

- With Momentum. The objective is to make the trajectory of the weights update more stable, preventing time loss in oscillations. It uses momentum, which can be interpreted as a force proportional to the velocity of the object but acts in the opposite direction:

$$W_{t+1} = W(t) + p(t+1) \tag{A.4}$$

$$p(t+1) = \mu p(t) - \alpha g(t) \tag{A.5}$$

  Where: μ is in the range (0, 1) and $p(t)$ is the momentum.

- RMSprop (Root Mean Squared Propagation): It scales gradients by dividing them by a moving average Root Mean Squared gradient ($s(t)$), aiming to slow down learning in directions where the gradient is higher and speed up the process where the gradient is lower:

$$W(t+1) = W(t) - \alpha \frac{g(t)}{\sqrt{s(t) + \epsilon}} \tag{A.6}$$

$$s(t) = \rho s(t-1) + (1-\rho)(g(t))^2 \tag{A.7}$$

  Where ρ is a moving-average decay factor.

- Adaptive Gradient - ADAGRAD: It is similar to RMSprop, but it uses the cumulative sum of squared gradients $c(t)$.

$$W(t+1) = w(t) - \frac{\eta}{\sqrt{c(t)}} \cdot g(t) \tag{A.8}$$

$$c(t) = \sum_{j=1}^{t} (g(j))^2 \tag{A.9}$$

- Momentum and RMSprop - ADAM: It combines momentum and RMSprop moving averages.

$$W(t+1) = w(t) - \frac{\eta}{\sqrt{s(t)}} \cdot p(t) \tag{A.10}$$

$$p(t+1) = \beta_1 \cdot p(t) - (1-\beta_1) \cdot g(t) \tag{A.11}$$

$$s(t) = \beta_2 \cdot s(t-1) + (1-\beta_2) \cdot (g(t))^2 \tag{A.12}$$

A.2 Convolutional Neural Networks (CNNs)

The field in which CNNs were born is image analysis, which started from a simple classification task and evolved to computer vision. Convolutional Neural Networks have a structure composed of several layers, with convolutional layers being the most frequent. These layers allow the convolution (2D or 3D) of the input image with a determined number of filters, which have specific dimensions in terms of width and height (they could also have a depth dimension for 3D convolution).

Each convolutional layer is characterized by the application of a non-linear activation function to the filters' output, resulting in activation maps. Common activation functions include Rectified Linear Unit (ReLU), Exponential Linear Unit (ELU), Leaky ReLU, or Parameterized ReLU (PReLU). Each activation function has its own characteristics, which are summarized in **Table A.1**

Table A.1: Comparison of Non-linear Activation Functions

Name	**Function**	**Pros and Cons**
ReLU	$f(x) = \max(0, x)$	No saturation for $x > 0$ No activation for $x < 0$ Not zero-centered
ELU	$f(x) = \begin{cases} x & \text{for } x > 0 \\ \alpha(e^x - 1) & \text{for } x \leq 0 \end{cases}$	No saturation for $x > 0$ Not zero-centered No de-activation for $x < 0$
Leaky ReLU	$f(x) = \max(0.01x, x)$	-
PReLU	$f(x) = \max(\alpha x, x)$	-

A.2.1 Fully Connected Layers and Softmax

Typically, at the end of a neural network architecture, there exist fully connected layers that serve to consolidate the activation maps obtained from the final convolutional layer into a 1D vector. This vector is subsequently transformed into a probabilistic distribution, where each unit in the final fully connected layer corresponds to a distinct class. To achieve this, the Softmax activation function is applied, yielding the final probabilities:

$$f(x)_i = \frac{e^{x_i}}{\sum_{j=1}^{K} e^{x_j}} \tag{A.13}$$

In this case, x is the vector representing the last fully connected layer, and K is the number of its units, which corresponds to the number of distinct classes. The Softmax function normalizes the components of the output vector, ensuring their sum equals 1. This characteristic allows the final numbers to be interpreted as probabilities. The class with the highest probability value is chosen as the final forecast. A common CNN architecture is depicted in **Figure A.1**.

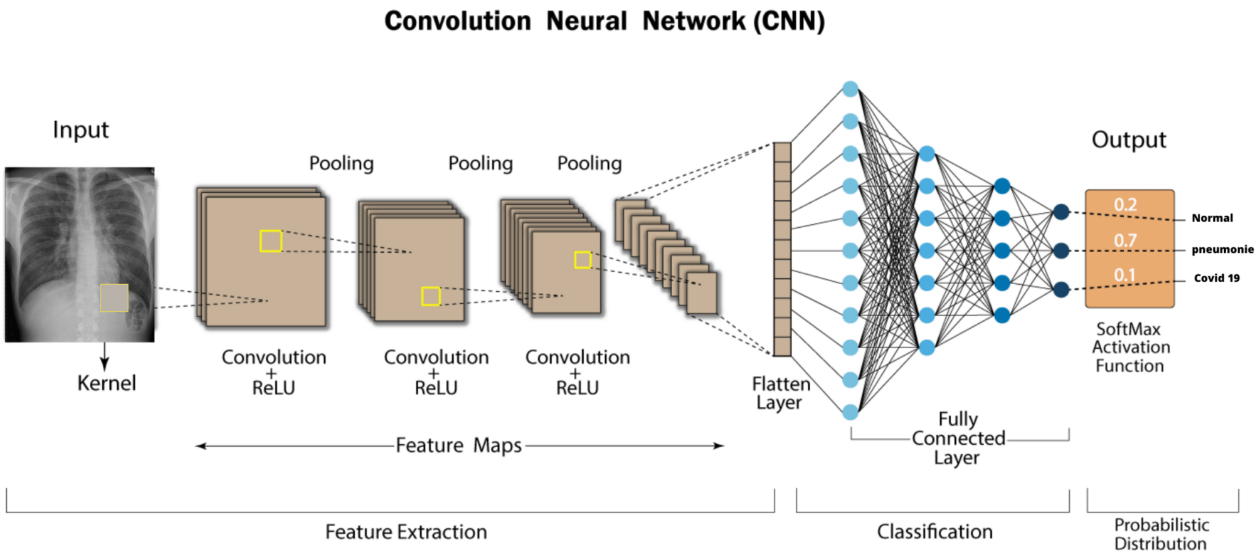

Figure A.1: Example of common CNN architecture

A.3 Separable Convolutional Layer

Unlike traditional CNNs, which employ standard convolutions that apply a set of learnable filters across the entire input volume, Separable CNNs decompose the convolution operation into two distinct stages: depthwise convolution and pointwise convolution. This separation reduces the number of parameters and computations, making Separable CNNs more efficient for tasks like image classification while maintaining competitive accuracy. Overall, Separable CNNs offer a practical trade-off between computational efficiency and model performance.

In a typical convolutional layer of a normal CNN, a 3D filter (e.g., a 3x3x3 filter for a color image with three channels) is applied at every spatial position across all input channels, generating a single output channel. In contrast, Separable CNNs first apply a depthwise convolution, where a separate 2D filter (e.g., 3x3) is applied independently to each input channel. This results in multiple output channels, one for each input channel. Subsequently, a pointwise convolution is performed to combine these channels linearly.

Mathematically, the depthwise convolution can be represented as:

$$Y_{i,j,k} = \sum_{m,n} X_{i+m,j+n,k} \cdot W_{m,n} \tag{A.14}$$

where X is the input tensor, Y is the output tensor, and W is the depthwise filter.

The pointwise convolution is then applied as:

$$Z_{i,j,l} = \sum_{k} Y_{i,j,k} \cdot V_{k,l} \tag{A.15}$$

where Z is the final output tensor, and V is the pointwise filter. In **Figure A.2**, a visual representation illustrates the concept explained above.

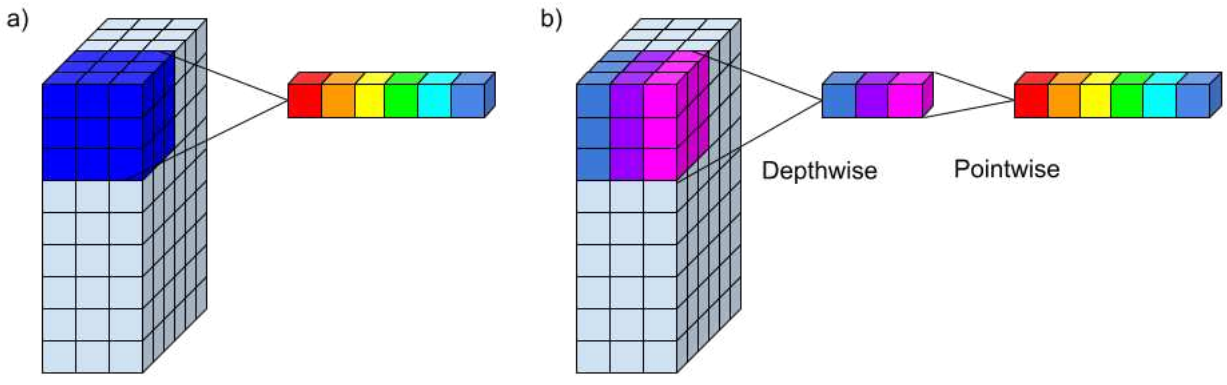

Figure A.2: Comparison between standard convolution **(a)** and separable convolution **(b)**

A.4 Domain Adversarial Network with Gradient Reversal

In the context of this research, a Domain Adversarial Network (DANN) is employed to facilitate domain-invariant feature learning. DANN is a type of neural network architecture that includes a domain classifier alongside the primary task classifier. Specifically, the second output of the model is dedicated to the domain classification task. A visualization of the structure con be seen in **Figure A.3**

To encourage the model to learn domain-invariant features, a critical step involves reversing the gradients during backpropagation when optimizing the domain classifier. This is achieved by introducing a gradient reversal layer. The purpose of reversing the gradient is to penalize the model when the domain classifier is correct, thus encouraging the feature extractor to generate features that are agnostic to the domain (e.g., different subjects).

Mathematically, the gradient reversal operation can be represented as follows, where L_d is the loss of the domain classifier:

$$\frac{\partial L_d}{\partial \theta} \rightarrow -\lambda \cdot \frac{\partial L_d}{\partial \theta} \tag{A.16}$$

Here, θ represents the model's parameters, and λ is a hyperparameter that controls the strength of the gradient reversal. By minimizing the domain classifier loss while simultaneously maximizing the primary task's performance, the DANN effectively learns features that are both discriminative for the main task and domain-invariant, allowing for improved generalization across different domains or subjects.

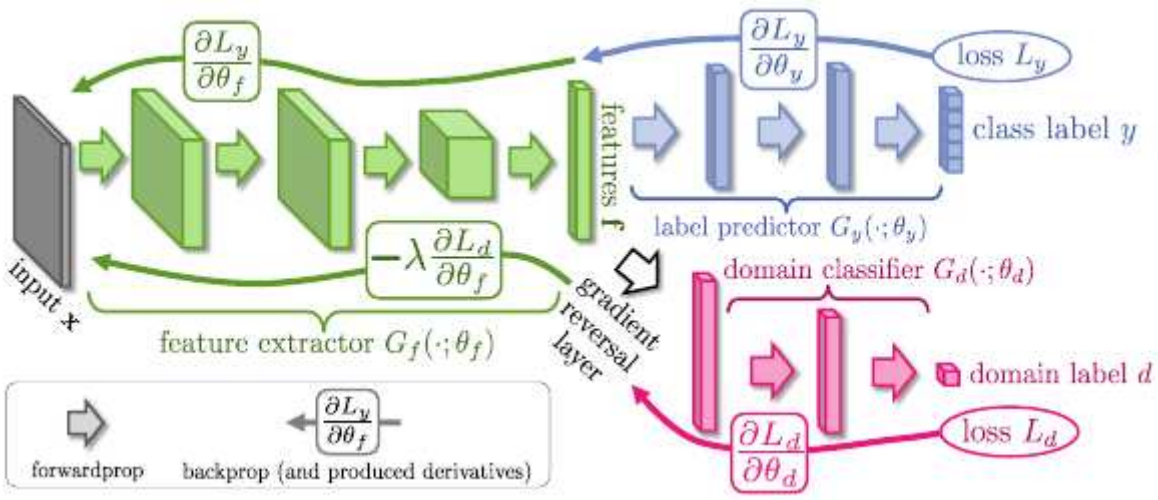

Figure A.3: Structure of the Domain Adversarial Network (DANN) architecture used for domain-invariant feature learning

A.5 Triplet Margin Loss

The basic goal of the Triplet Margin Loss is to train data embeddings or representations that increase the similarity of positive pairs of samples while driving negative pairs away in a high-dimensional feature space. The loss function is given as follows:

$$L_{\text{triplet}} = \sum_{i} \left[\max\left(0, \text{margin} + d(a_i, p_i) - d(a_i, n_i)\right)\right] \tag{A.17}$$

Where:

- L_{triplet} represents the triplet margin loss.
- a_i is the anchor sample.
- p_i is a positive sample (with the same label as the anchor).
- n_i is a negative sample (with a different label from the anchor).
- margin is a margin parameter controlling the desired minimum separation between positive and negative samples.
- The function $d(x, y)$ calculates the distance between feature vectors x and y in the high-dimensional space.

In this context, **Figure A.4**in provides a visual illustration of how triplet margin loss equations operate, offering deeper insights into its mathematical foundation and practical significance in domain adaptation tasks.

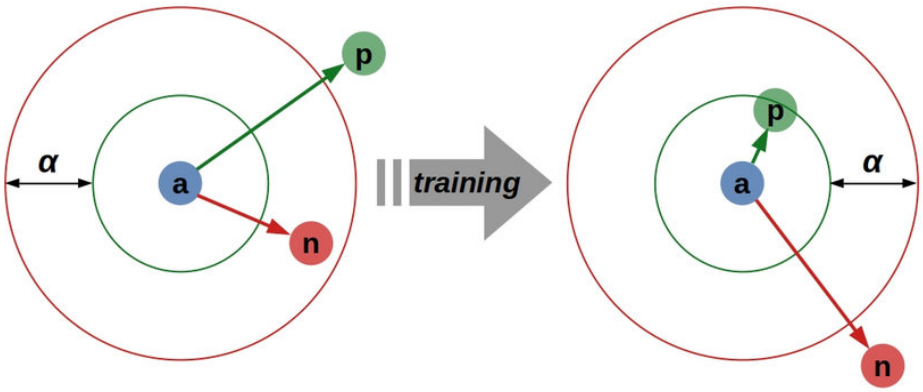

Figure A.4: Visualization of the Triplet Margin Loss Mechanism

Bibliography

[1] Niroj Parajuli et al. "Real-Time EMG Based Pattern Recognition Control for Hand Prostheses: A Review on Existing Methods, Challenges and Future Implementation". In: *Sensors* 19.20 (Oct. 2019). DOI: 10.3390/s19204596.

[2] Dick F Stegeman et al. "Surface EMG models: properties and applications". In: *Journal of Electromyography and Kinesiology* 10.5 (Oct. 2000), pp. 313–326. DOI: 10.1016/S1050-6411(00)00023-7.

[3] S. Muceli R. Merletti. "Surface EMG detection in space and time: Best practices". In: *Journal of Electromyography and Kinesiology* (2019). DOI: https://www.sciencedirect.com/science/article/pii/S1050641119302536.

[4] Alessandro Del Vecchio et al. "You are as fast as your motor neurons: speed of recruitment and maximal discharge of motor neurons determine the maximal rate of force development in humans". In: *Journal of Physiology* 597 (9 2019), pp. 2445–2456. DOI: 10.1113/JP277396. URL: https://doi.org/10.1113/JP277396.

[5] C. J. De Luca et al. "Decomposition of Surface EMG Signals". In: *Journal of Neurophysiology* (Sept. 2006). DOI: 10.1152/jn.00009.2006. URL: http://doi.org/10.1152/jn.00009.2006.

[6] H. Liu et al. "An epidermal sEMG tattoo-like patch as a new human–machine interface for patients with loss of voice". In: *Microsystems & Nanoengineering* (Mar. 2020). DOI: 10.1038/s41378-019-0127-5. URL: http://doi.org/10.1038/s41378-019-0127-5.

[7] J. Wu, L. Sun, and R. Jafari. "A Wearable System for Recognizing American Sign Language in Real-Time Using IMU and Surface EMG Sensors". In: *IEEE Journal of Biomedical and Health Informatics* (Sept. 2016). DOI: 10.1109/jbhi.2016.2598302. URL: http://doi.org/10.1109/jbhi.2016.2598302.

[8] I. Campanini et al. "Surface EMG in Clinical Assessment and Neurorehabilitation: Barriers Limiting Its Use". In: *Frontiers in Neurology* (Sept. 2020). DOI: 10.3389/fneur.2020.00934. URL: http://doi.org/10.3389/fneur.2020.00934.

[9] L. McManus, G. De Vito, and M. M. Lowery. "Analysis and Biophysics of Surface EMG for Physiotherapists and Kinesiologists: Toward a Common Language With Rehabilitation Engineers". In: *Frontiers in Neurology* (Oct. 2020). DOI: 10.3389/fneur.2020.576729. URL: http://doi.org/10.3389/fneur.2020.576729.

[10] B. Treussart et al. "Controlling an upper-limb exoskeleton by EMG signal while carrying unknown load". In: (2020). DOI: 10.1109/icra40945.2020.9197087. URL: http://doi.org/10.1109/icra40945.2020.9197087.

[11] C. Fang et al. "EMG-Centered Multisensory Based Technologies for Pattern Recognition in Rehabilitation: State of the Art and Challenges". In: *Biosensors* (July 2020). DOI: 10.3390/bios10080085. URL: http://doi.org/10.3390/bios10080085.

[12] K. Kiguchi and Y. Hayashi. "An EMG-Based Control for an Upper-Limb Power-Assist Exoskeleton Robot". In: *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)* (Aug. 2012). DOI: 10.1109/tsmcb.2012.2185843. URL: http://doi.org/10.1109/tsmcb.2012.2185843.

[13] O. W. Samuel et al. "Intelligent EMG Pattern Recognition Control Method for Upper-Limb Multifunctional Prostheses: Advances, Current Challenges, and Future Prospects". In: *IEEE Access* (2019). DOI: 10.1109/access.2019.2891350. URL: http://doi.org/10.1109/access.2019.2891350.

[14] J. W. Yoo et al. "Augmented effects of EMG biofeedback interfaced with virtual reality on neuromuscular control and movement coordination during reaching in children with cerebral palsy". In: *NeuroRehabilitation* (Mar. 2017). DOI: 10.3233/nre-161402. URL: http://doi.org/10.3233/nre-161402.

[15] A. Dash and U. Lahiri. "Design of Virtual Reality-Enabled Surface Electromyogram-Triggered Grip Exercise Platform". In: *IEEE Transactions on Neural Systems and Rehabilitation Engineering* (Feb. 2020). DOI: 10.1109/tnsre.2019.2959449. URL: http://doi.org/10.1109/tnsre.2019.2959449.

[16] D. Blana et al. "Feasibility of using combined EMG and kinematic signals for prosthesis control: A simulation study using a virtual reality environment". In: *Journal of Electromyography and Kinesiology* (Aug. 2016). DOI: 10.1016/j.jelekin.2015.06.010. URL: http://doi.org/10.1016/j.jelekin.2015.06.010.

[17] B. Ambikapathy and K. Krishnamurthy. "Analysis of electromyograms recorded using invasive and noninvasive electrodes: a study based on entropy and Lyapunov exponents estimated using artificial neural networks". In: *Journal of Ambient Intelligence and Humanized Computing* (2018). DOI: 10.1007/s12652-018-0811-6. URL: http://doi.org/10.1007/s12652-018-0811-6.

[18] S. Pancholi and A. M. Joshi. "Portable EMG Data Acquisition Module for Upper Limb Prosthesis Application". In: *IEEE Sensors Journal* (2018). DOI: 10.1109/jsen.2018.2809458. URL: http://doi.org/10.1109/jsen.2018.2809458.

[19] Angélica Jaramillo-Yánez, Milton E. Benalcázar, and Edison Mena-Maldonado. "Real-Time Hand Gesture Recognition Using Surface Electromyography and Machine Learning: A Systematic Literature Review". In: *Sensors* 20.9 (2020), p. 2467. DOI: 10.3390/s20092467.

[20] Alessio Burrello et al. "Bioformers: Embedding Transformers for Ultra-Low Power sEMG-based Gesture Recognition". In: *2022 Design, Automation Test in Europe Conference Exhibition (DATE)*. IEEE. 2022. DOI: 10.23919/date54114.2022.9774639.

[21] Chunyang Xie et al. "Reducing the Energy Consumption of sEMG-Based Gesture Recognition at the Edge Using Transformers and Dynamic Inference". In: *Sensors* 23.4 (2023), p. 2065. DOI: 10.3390/s23042065.

[22] K. Park and S. Lee. "Movement Intention Decoding Based on Deep Learning for Multiuser Myoelectric Interfaces". In: (2016), pp. 1–2. DOI: 10.1109/IWW-BCI.2016.7457459.

[23] M. Atzori, M. Cognolato, and H. Müller. "Deep learning with convolutional neural networks applied to electromyography data: a resource for the classification of movements for prosthetic hands". In: *Frontiers in Neurorobotics* 10 (2016), p. 9. DOI: 10.3389/fnbot.2016.00009.

[24] W. Wei et al. "A multi-stream convolutional neural network for sEMG-based gesture recognition in muscle-computer interface". In: *Pattern Recognition Letters* 119 (2019), pp. 131–138. DOI: 10.1016/j.patrec.2017.12.005.

[25] Y. Zou and L. Cheng. "A Transfer Learning Model for Gesture Recognition Based on the Deep Features Extracted by CNN". In: *IEEE Transactions on Artificial Intelligence* (Oct. 2021). DOI: 10.1109/tai.2021.3098253.

[26] Ali Raza Asif et al. "Performance Evaluation of Convolutional Neural Network for Hand Gesture Recognition Using EMG". In: *Sensors* 20.6 (Mar. 2020). DOI: 10.3390/s20061642.

[27] J. Chen et al. "High-density surface EMG-based gesture recognition using a 3D convolutional neural network". In: *Sensors* 20 (2020), p. 1201. DOI: 10.3390/s20041201.

[28] Jennifer L. Betthauser et al. "Stable Responsive EMG Sequence Prediction and Adaptive Reinforcement With Temporal Convolutional Networks". In: *IEEE Transactions on Biomedical Engineering* (2020). DOI: 10.1109/tbme.2019.2943309.

[29] Marco Zanghieri et al. "Robust Real-Time Embedded EMG Recognition Framework Using Temporal Convolutional Networks on a Multicore IoT Processor". In: *IEEE Transactions on Biomedical Circuits and Systems* (2020). DOI: 10.1109/tbcas.2019.2959160.

[30] Y. Hu et al. "A novel attention-based hybrid CNN-RNN architecture for sEMG-based gesture recognition". In: *PLOS ONE* (Oct. 2018). Ed. by H. He. DOI: 10.1371/journal.pone.0206049.

[31] Y. Wu, B. Zheng, and Y. Zhao. "Dynamic Gesture Recognition Based on LSTM-CNN". In: (Nov. 2018). DOI: 10.1109/cac.2018.8623035.

[32] Ivan Vujaklija et al. "Online Mapping of EMG Signals into Kinematics by Autoencoding". In: *Journal of NeuroEngineering and Rehabilitation* (2018). DOI: 10.1186/s12984-018-0363-1.

[33] Ulysse Cote-Allard et al. "Unsupervised Domain Adversarial Self-Calibration for Electromyography-Based Gesture Recognition". In: *IEEE Access* (2020). DOI: 10.1109/access.2020.3027497.

[34] Yubo Du et al. "Surface EMG-Based Inter-Session Gesture Recognition Enhanced by Deep Domain Adaptation". In: *Sensors* 17.3 (Feb. 2017). DOI: 10.3390/s17030458.

[35] Xiaolong Zhai et al. "Self-Recalibrating Surface EMG Pattern Recognition for Neuroprosthesis Control Based on Convolutional Neural Network". In: *Frontiers in Neuroscience* (2017). DOI: 10.3389/fnins.2017.00379.

[36] M. Atzori et al. "Electromyography data for non-invasive naturally-controlled robotic hand prostheses". In: *Scientific Data* (Dec. 2014). DOI: 10.1038/sdata.2014.53. URL: http://doi.org/10.1038/sdata.2014.53.

[37] A. M. Simon et al. "Target Achievement Control Test: Evaluating real-time myoelectric pattern-recognition control of multifunctional upper-limb prostheses". In: *The Journal of Rehabilitation Research and Development* (2011). DOI: 10.1682/jrrd.2010.08.014.

[38] Mario Cifrek et al. "Surface EMG based muscle fatigue evaluation in biomechanics". In: *Clinical Biomechanics* 24 (4 May 2009), pp. 327–340. DOI: 10.1016/j.clinbiomech.2009.01.010. URL: https://doi.org/10.1016/j.clinbiomech.2009.01.010.

[39] Wen Li, Peng Shi, and Haoyong Yu. "Gesture Recognition Using Surface Electromyography and Deep Learning for Prostheses Hand: State-of-the-Art, Challenges, and Future". In: *Frontiers in Neuroscience* (Apr. 2021). DOI: 10.3389/fnins.2021.621885. URL: http://doi.org/10.3389/fnins.2021.621885.

[40] E. A. Clancy, E. L. Morin, and R. Merletti. "Sampling, Noise-Reduction and Amplitude Estimation Issues in Surface Electromyography". In: *Journal of Electromyography and Kinesiology* (Feb. 2002). DOI: 10.1016/s1050-6411(01)00033-5. URL: http://doi.org/10.1016/s1050-6411(01)00033-5.

[41] M. F. Wahid, R. Tafreshi, and R. Langari. "A Multi-Window Majority Voting Strategy to Improve Hand Gesture Recognition Accuracies Using Electromyography Signal". In: *IEEE Transactions on Neural Systems and Rehabilitation Engineering* (Feb. 2020). DOI: 10.1109/tnsre.2019.2961706.

[42] Yaroslav Ganin and Victor Lempitsky. "Unsupervised Domain Adaptation by Backpropagation". In: 37 (2015), pp. 1180–1189.

[43] Yuzhou Lin et al. "A Normalisation Approach Improves the Performance of Inter-Subject sEMG-based Hand Gesture Recognition with a ConvNet". In: *IEEE* (2020). DOI: 10.1109/EMBC44109.2020.9175156.

[44] Reema Jain and Vijay Kumar Garg. "An Efficient Feature Extraction Technique and Novel Normalization Method to Improve EMG Signal Classification". In: *IEEE Transactions on Intelligent Engineering and Management* (2022). DOI: 10.1109/ICIEM54221.2022.9853101.

[45] Taichi Tanaka et al. "Sliding-Window Normalization to Improve the Performance of Machine-Learning Models for Real-Time Motion Prediction Using Electromyography". In: *Sensors* 22 (2022). DOI: 10.3390/s22135005.

[46] L. H. Smith et al. "Determining the Optimal Window Length for Pattern Recognition-Based Myoelectric Control: Balancing the Competing Effects of Classification Error and Controller Delay". In: *IEEE Transactions on Neural Systems and Rehabilitation Engineering* (Apr. 2011). DOI: 10.1109/tnsre.2010.2100828.

[47] Ulysse Côté-Allard et al. "Deep learning for electromyographic hand gesture signal classification using transfer learning". In: *IEEE Transactions on Neural Systems and Rehabilitation Engineering* (Apr. 2019).

[48] W. Geng et al. "Gesture recognition by instantaneous surface EMG images". In: *Scientific Reports* (Nov. 2016). DOI: `insert-doi-here`.

[49] S. Tam et al. "A Fully Embedded Adaptive Real-Time Hand Gesture Classifier Leveraging HD-sEMG and Deep Learning". In: *IEEE Transactions on Biomedical Circuits and Systems* (2020). DOI: `10.1109/tbcas.2019.2955641`.

Contents

List of Figures

List of Tables