MYOELECTRIC CONTROL FOR ACTIVE PROSTHESES VIA DEEP NEURAL NETWORKS AND DOMAIN ADAPTATION

Elahe Rahimian Najafabadi

A thesis
in
The Department
of
Concordia Institute for Information Systems Engineering (CIISE)

Presented in Partial Fulfillment of the Requirements
For the Degree of Doctor of Philosophy
Concordia University
Montréal, Québec, Canada

October 2022

CONCORDIA UNIVERSITY
School of Graduate Studies

This is to certify that the thesis prepared

By:	**Elahe Rahimian Najafabadi**
Entitled:	**Myoelectric Control for Active Prostheses via Deep Neural Networks and Domain Adaptation**

and submitted in partial fulfillment of the requirements for the degree of

Doctor of Philosophy (Information and Systems Engineering)

complies with the regulations of this University and meets the accepted standards with respect to originality and quality.

Signed by the final examining commitee:

Dr. Bruno Lee	Chair
Dr. Mahdi Tavakoli	External Examiner
Dr. Amir Asif	External to Program
Dr. Abdessamad Ben Hamza	Examiner
Dr. Nizar Bouguila	Examiner
Dr. Arash Mohammadi	Supervisor

Approved by

Dr. Zachary Patterson, Graduate Program Director

09/29/2022

Dr. Mourad Debbabi, Dean
Faculty of Engineering and Computer Science

Abstract

Myoelectric Control for Active Prostheses via Deep Neural Networks and Domain Adaptation

Elahe Rahimian Najafabadi, Ph.D.
Concordia University, 2022

Recent advances in Biological Signal Processing (BSP) and Machine Learning (ML), in particular, Deep Neural Networks (DNNs), have paved the way for development of advanced Human-Machine Interface (HMI) systems for decoding human intent and controlling artificial limbs. Myoelectric control, as a subcategory of HMI systems, deals with detecting, extracting, processing, and ultimately learning from Electromyogram (EMG) signals to command external devices, such as hand prostheses. In this context, hand gesture recognition/classification via Surface Electromyography (sEMG) signals has attracted a great deal of interest from many researchers. Despite extensive progress in the field of myoelectric prosthesis, however, there are still limitations that should be addressed to achieve a more intuitive upper limb prosthesis. Through this Ph.D. thesis, first, we perform a literature review on recent research works on pattern classification approaches for myoelectric control prosthesis to identify challenges and potential opportunities for improvement. Then, we aim to enhance the accuracy of myoelectric systems, which can be used for realizing an accurate and efficient HMI for myocontrol of neurorobotic systems. Beside improving the accuracy, decreasing the number of parameters in DNNs plays an important role in a Hand Gesture Recognition (HGR) system. More specifically, a key factor to achieve a more intuitive upper limb prosthesis is the feasibility of embedding DNN-based models into prostheses controllers. On the other hand, transformers are considered to be powerful DNN models that have revolutionized the Natural Language Processing (NLP) field and showed great potentials to dramatically improve different computer vision tasks. Therefore, we propose a Transformer-based neural network architecture to classify and recognize upper-limb hand gestures. Finally, another goal of this thesis is to design a modern DNN-based gesture detection model that relies on minimal training data while providing high accuracy. Although DNNs have shown superior accuracy compared to conventional methods when large amounts of data are available for training, their performance substantially decreases when data are limited. Collecting large datasets for training may be feasible in research laboratories, but it is not a practical approach for real-life applications. We propose to solve this problem, by designing a framework which utilizes a combination of temporal convolutions and attention mechanisms.

Acknowledgments

First and foremost, I would like to express my sincere gratitude to my Supervisor, Dr. Arash Mohammadi for the continuous support of my Ph.D study and research, for his patience, motivation, enthusiasm, and immense knowledge. I deeply express my gratitude to the committee members, Dr. Abdessamad Ben Hamza, Dr. Amir Asif, Dr. Nizar Bouguila, and Dr. Mahdi Tavakoli, for evaluating this dissertation and their thoughtful feedback. I would like to express my gratitude to my parents for their incredible support, inspiration, and understanding, and last but not least, my husband whose unconditional support and patience helped me believe in myself.

Contents

List of Figures

List of Tables

List of Abbreviations

AI	Artificial Intelligence
BERT	Bidirectional Encoder Representations from Transformers
BSP	Biological Signal Processing
CCA	Canonical Correlation Analysis
CNN	Convolutional Neural Network
CV	Computer Vision
DNNs	Deep Neural Networks
DoF	Degree of Freedom
ECG	Electrocardiogram
EEG	Electroencephalogram
ELU	Exponential Linear Unit
EMG	Electromyography
FC	Fully Connected
GPT	Generative Pre-Training
GRU	Gated Recurrent Unit
HCFs	Hand Crafted Features
HD-sEMG	High-Density sEMG
HGR	Hand Gesture Recognition
HMI	Human-Machine Interface
IMs	Inertial Measurements
IQR	Interquartile Range
KNN	K-Nearest Neighbours
LDA	Linear Discriminant Analysis
LSTM	Long Short-Term Memory
MC	Monte Carlo

ML Machine Learning

MS Multi-Stream

MUAPs Motor Unit Action Potentials

NLP Natural Language Processing

PCA Principal Component Analysis

ReLU Rectified Linear Unit

RF Random Forests

RL Reinforcement Learning

RNNs Recurrent Neural Networks

sEMG surface EMG

SNR Signal to Noise Ratio

SRC Sparse Representation Classification

STFT Short-Time Fourier Transform

SVM Support Vector Machine

TCN Temporal Convolutional Network

TL Transfer Learning

ViT Vision Transformer

Chapter 1

Overview of the Thesis

1.1 Introduction

An upper-limb difference, be it due to an amputation, or by birth, can deteriorate the quality of life. The individual can rely on a passive or robotic extension of their residual limb to perform activities of daily living and to secure a manageable degree of independence. Despite centuries of advancements, passive prostheses have shown limited functionality, while active robotic prostheses have a significant potential to (re-)enable the individual to conduct versatile tasks. For actuating the robotic prostheses, Electromyography (EMG) [1–9] has been conventionally studied as a potential signal modality that can translate the peripheral responses of residual muscle activations into control commands.

Myoelectric control via surface EMG (sEMG) signals [10] has found several applications of significant engineering importance for the development of neuro-rehabilitation devices, such as multi-function prostheses. These sEMG-based neuro assistive machines are generated for the replacement, restoration, and/or modulation of lost or impaired function in research and clinical settings [11]. Due to its non-invasiveness and richness of neural information, sEMG has been used extensively in Human-Machine Interface (HMI) systems for commercial/clinical upper-limb prosthetic control. Although sEMG has significant utility for control of prosthetic systems, it is affected by several factors, many of which degrade the quality of the signal and its information content [12]. Such factors include pervasive electromagnetic noise, electrode repositioning, dynamic motions artifacts, changes in electrode-skin impedance (e.g., because of sweating), and relative movement of the associated muscle with respect to electrodes (including muscle fiber lengthening and shortening, which changes the mappings in a stochastic fashion). Moreover, distribution and shape of the underlying Motor Unit Action Potentials (MUAPs) that comprise the sEMG signal vary greatly among subjects because of the anatomical variations in locations of the motor units within the muscle tissue [13]. In summary, the efficacy and acceptance of myoelectric prostheses are influenced by a number of factors such as noise, the expense of servicing, and specifically intuitiveness of control. These factors can substantially influence the performance of myoelectric control algorithms [13].

To tackle existing technical and computational challenges, researchers have been developing advanced Biological Signal Processing (BSP) and Machine Learning (ML) algorithms aimed towards enhancing a high spatio-temporal resolution for decoding the intention of the user. In brief, although a major amount of research has been conducted on the design and implementation of sophisticated BSP and ML techniques, only marginal progress has been made in the clinically-viable commercial solutions for prosthetic control. More effort is, therefore, needed to address the prosthesis users needs within multidisciplinary initiatives. An ideal HMI needs to provide consistent, direct, intuitive, and accurate mechanisms for control of a multi-function prosthetic system with minimal need for training and calibration. Furthermore, an ideal HMI should provide sensory feedback mechanisms to enable bidirectional interaction with the environment, thus creating a closed control loop. More recent advanced HMI systems, developed in research laboratories, have become increasingly sophisticated to address some of the above-mentioned objectives. This has been achieved through the use of embedded mechatronics systems in combination with state-of-the-art ML modules and real-time BSP pipelines designed to achieve high dexterity and versatility. Recent evolution in Deep Neural Networks (DNNs) coupled with advancements in rehabilitation technologies has resulted in a promising future to develop intuitive myoelectric prostheses. In this context, the sEMG signals derived from the muscle fibers' action potentials, have been used in the literature for Hand Gesture Recognition (HGR) in advanced myoelectric prostheses. In this regard, gesture recognition and classification has attracted a great deal of interest from many researchers due to its high potential for improving the quality of control over the actions of prostheses, which can significantly enhance the quality of lives of hand amputated individuals.

1.2 Research Objectives

This Ph.D. thesis targets addressing the aforementioned challenges associated with HGR via sparse multichannel sEMG signals through design and development of novel DNN-based architectures (also referred to as deep MyoLearning). In this context, the thesis targets achieving the following three main research objectives:

- ***Improving the Overall Accuracy of HGR***: The first objective to design new sEMG-based DNN-based models for prosthesis control is to achieve the highest possible classification accuracy. In fact, high accuracy is of great importance to achieve the ultimate goal of effectively tracking static and dynamic patterns and converting classification results into smooth prosthetic actions. The sEMG signals can be collected based on *sparse multichannel sEMG* or in more advanced cases using *High-Density sEMG (HD-sEMG)* devices. Although high recognition accuracy is achievable by using HD-sEMG data, the performance drops when the data is collected based on sparse sEMG devices. As a result, the existing solutions developed based on sparse sEMG are still far from being optimal.

- ***Reducing Complexity of DNN Architectures***: A key factor for clinical adoption of sEMG-based DNN architectures is the feasibility of embedding such models into prostheses controllers. Computational cost requirements needed to train conventional DNN architectures such as Convolutional Neural Network (CNN), Long Short-Term Memory (LSTM), or CNN+LSTM models can easily exceed many thousands of parameters, especially with the very deep models. As a result, we are still far away from embedding a DNN model with thousands of parameters in a prosthetics device. This motivates more research on reducing complexity of the underlying architectures to design compact DNN models for deep MyoLearning.

- ***Developing Adaptive Learning Solutions***: Another key factor for practical utilization of sEMG-based DNN architectures is reducing their training burden. There is, therefore, a need to develop adaptive learning solutions with a focus on designing a DNN model, which can be adopted for new subjects based on only a few examples through a fast learning approach, i.e., reducing the training load. More specifically, to fill the gap between Source data (train data) and Target data (Test data) a concept called domain adaptation is used. It is the ability to apply an algorithm that is trained on one or more source domains to a different target domain. In particulat, it is a subcategory of transfer learning. In domain adaptaion, the source and target data have the same feature space but from different distributions, while transfer learning includes cases where target feature space is different from source feature space. More specifically, given data or experience on previous tasks, learning a new task more quickly and proficiently is another key factor for the practical use of sEMG-based DNNs architectures. Moreover, fast learning is another factor that should be improved in the field of deep MyoLearning. Fast learning is an indication of human intelligence, which involves recognizing an object just by looking at a few examples or quickly repeating an action a few times. This is a challenging task since many factors such as electrode location and muscle fiber lengthening/shortening can affect the collected sEMG signals. Moreover, the neurophysiology differences between users and the changes caused by amputations result in more discrepancies between different conditions.

1.3 Targeted Challenges

Despite recent advancements in the field of sEMG-based HGR and increase of its potential clinical applications, there are still several open problems and challanges, which require extensive investigations, including:

C1. ***Lack of Generalizability of DNN-based Architectures***: Existing data-driven approaches designed for HGR are developed based on a single deep model that can hardly maintain proper generalization performance across various prognostic scenarios.

C2. ***Joint Incorporation of Spatial and Sequential Information***: A common strategy used for sEMG-based HGR is to convert the multichannel sEMG recording over fix time windows into images and then use CNN-based image classification models. The problem with such an approach is that only the spatial information of sEMG signals are captured without considering the sequential nature of the sEMG signals. There is, therefore, a need to capture both temporal and spatial information of the sEMG signals without the need for data augmentation and/or manual design of feature extraction to perform the recognition task.

C3. ***High Memory Requirement and Parallelization***: While the recurrent-based models, such as LSTM, are advanced approaches to sequence modeling, they do not allow parallelization during the training phase due to their sequential nature.

C4. ***Design of Compact DNN Models***: Although DNN models have shown promising results for HGR, these data-driven models have been challenged by the need for a large number of parameters and structural complexity. The challenge is to design DNN architectures with small number of trainable parameters.

C5. ***Handling Sequential Inputs with Long-Time Dependencies***: Designing Recurrent Neural Networks (RNNs) to accurately handle sequential inputs with long-term dependencies is very challenging because of the exploding and vanishing gradient problem. There is, therefore, a need to develop transformer architectures for HGR task because the transformers do not have this problem as the distance to each element in the sequence is always *O(1)* sequential operations away.

C6. ***Introducing Robustness against sEMG Signal Variations***: The classification accuracy of DNN-based models is affected by several factors such as users' characteristics, which are not fully considered in the scientific field. Such variabilities that exists in the nature of sEMG signals can affect myoelectric prosthesis performance. There is, therefore, a need to develop adaptive methods that combine prior knowledge gathered from the source domain with new information to provide robustness.

C7. ***Reducing the Training Load***: Although DNNs have shown superior accuracy compared to conventional methods when large amounts of data are available for training, their performance substantially decreases when data are limited. Overcoming the issue of having insufficient training data for the system recalibrations is of great importance for sEMG-based HGR. However, collecting large training data is time consuming (can take up to several days), and is cumbersome for the end-users. Collecting large datasets for training may be feasible in research laboratories, but it is not a practical approach for real-life applications.

1.4 Thesis Contributions

Below, the contributions of the thesis are briefly outlined:

- **Chapter 3 [2,7–9]: Accurate and Efficient sEMG-based Deep Learning Models for HGR**
  - Motivated by the potentials of deep learning models in significantly improving myoelectric control of neuro-prosthetic robotic limbs, a novel deep learning architecture, namely the Hybrid Recognition Model (**HRM**) for performing HGR via multi-channel sEMG signals [2, 7], is introduced in Sub-section 3.1.2 targeting challenges C1 and C2. The HRM architecture is aimed at enhancing the accuracy of myoelectric systems, which can be used for realizing an accurate HMI for myocontrol of neuro-robotic systems. The HRM is developed based on an innovative, unconventional, and particular hybridization of two parallel paths (one convolutional and one recurrent) coupled via a fully-connected multilayer network acting as the fusion center providing robustness across different scenarios. The hybrid design is specifically proposed to treat temporal and spatial features in two parallel processing pipelines and to augment the discriminative power.
  - A second architecture, referred to as Temporal Convolutional Network (**TCN**) [2, 8] is designed, which is developed based on dilated causal convolutions. It is worth mentioning that efficiency of a designed deep model, especially its memory usage, is as important as its achievable accuracy in practice. The TCN has significantly less memory requirement in training when compared with the HRM due to the implementation of novel dilated causal convolutions that gradually increase the receptive field of the network and utilize shared filter parameters. The training step of the proposed architecture is considerably faster than that of RNNs due to the absence of recurrent connections. Moreover, different from the existing deep learning methods for HGR, in this approach, by applying Conv1d,

the network itself is able to extract the hidden correlations existing between different signal sequences. This contribution targets challenges C2 and C3 in Sub-section 3.1.3.

  - **XceptionTime** architecture [9] is designed by integration of depthwise separable convolutions, adaptive average pooling, and a novel non-linear normalization technique. By utilizing the depthwise separable convolutions, the XceptionTime network has far fewer number of trainable parameters resulting in a more efficient and less complex network. This novel XceptionTime architecture is designed to improve the performance of HGR both in terms of the recognition accuracy and the complexity of the system. This contribution targets challenge C4 in Section 3.2.

- **Chapter 4 [3, 4]: Attention-Based Models for HGR**

  - To address the challenges with recurrent architectures and achieve high accuracy for sparse multichannel sEMG, we proposed the design of the **TC-HGR** [4], which is based on self-attention mechanism and temporal convolution to address the aforementioned challenges with the recurrent architectures (targeting challenge C3 in Section 4.2). The TC-HGR reduces the number of parameters, which is a key step forward to embed the DNN models into prostheses controllers. Moreover, the TC-HGR divides the sEMG signals into patches, which reduces the computational burden of the system. The TC-HGR can access a long history through temporal convolutions and also can pinpoint specific information in the sEMG signals through the attention mechanism.
  - The **TEMGNet** [3] is designed on the basis of transformers to enhance the accuracy of EMG classification. To the best of our knowledge, transformer models remain unexplored for the task of sEMG-based hand gestures recognition, making it an urgent quest to investigate their potentials within this domain. The TEMGNet architecture has been proposed to eliminate recurrence or convolution using a self-attention mechanism that targets challenge C5. The proposed TEMGNet architecture was designed based on the Vision Transformer (ViT) architecture to improve recognition accuracy and to reduce structural complexity. In Section 4.3, we show

that the proposed TEMGNet architecture outperforms its state-of-the-art counterparts in terms of overall recognition accuracy and complexity.

- **Chapter 5 [1, 5, 6]: Adaptive-Based Models for HGR**
  - The **FS-HGR** framework [1] provides a novel venue for adopting few-shot learning, to not only reduce the training time, but also to eventually mitigate the significant challenge of variability in the characteristics of sEMG signals. In other words, the proposed FS-HGR framework allows a myoelectric controller, that has been built based on background data, to adapt to the changes in the stochastic characteristics of sEMG signals using a small number of new observations. This class of architectures is introduced for sEMG meta-learning, where the meta-learner, via adaptation, quickly incorporates and refers to the experience based on just few training observations. This contribution targets challenges C6 and C7 in Chapter 5.

1.5 Organization of the Thesis

Chapter 1 (this chapter) provided an overview and a summary of important contributions made in the thesis. The reminder of the thesis is organized as follows:

- **Chapter 2** provides the literature review of pattern recognition-based myoelectric control leading to more advanced DNN-based frameworks (Deep MyoLearning).
- In **Chapter 3**, we concentrate on presenting our proposed accurate and efficient deep learning-based solutions for sEMG-based HGR. In particular, HRM and TCN architectures are introduced in Sub-section 3.1.2 and Sub-section 3.1.3, respectively. Furthermore, in Section 3.2, we propose the XceptionTime architecture.
- Attention-based models for HGR is investigated in **Chapter 4**. In this chapter, we introduce two architectures named TC-HGR and TEMGNet in Section 4.2 and Section 4.3, respectively.
- Developing an adaptive learning method is provided in **Chapter 5**. This chapter focuses on the FS-HGR architecture in particular.

- In **Chapter 6**, we conclude the thesis and remaining work will be discussed.

1.6 Publications

Journal Publications

J3. **E. Rahimian**, S. Zabihi, A. Asif, D. Farina, S.F. Atashzar, and A. Mohammadi, "FS-HGR: Few-shot Learning for Hand Gesture Recognition via ElectroMyography", *IEEE Transactions on Neural Systems and Rehabilitation Engineering (TNSRE)*, vol. 29, pp. 1004-1015, 2021.

J2. **E. Rahimian**, S. Zabihi, S.F. Atashzar, A. Asif, and A. Mohammadi, "Surface EMG-Based Hand Gesture Recognition via Hybrid and Dilated Deep Neural Network Architectures for Neurorobotic Prostheses", *Journal of Medical Robotics Research*, pp.1-12, 2020.

J1. **E. Rahimian**, S Zabihi, A Asif, D. Farina, S.F. Atashzar, A Mohammadi, "TEMGNet: Deep Transformer-based Decoding of Upperlimb sEMG for Hand Gestures Recognition", *Submitted to IEEE Transactions on Medical Robotics and Bionics (TMRB)*, 2021.

Conference Publications

C6. **E. Rahimian**, S Zabihi, A Asif, D. Farina, S.F. Atashzar, A Mohammadi, "Hand Gesture Recognition Using Temporal Convolutions and Attention Mechanism", *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2022.

C5. **E. Rahimian**, S. Zabihi, A. Asif, S.F. Atashzar, and A. Mohammadi, "Trustworthy Adaptation with Few-Shot Learning for Hand Gesture Recognition", *IEEE International Conference on Autonomous Systems (ICAS)*, pp. 1-5, 2021.

C4. **E. Rahimian**, S. Zabihi, A. Asif, S.F. Atashzar, and A. Mohammadi, "Few-Shot Learning for Decoding Surface Electromyography for Hand Gesture Recognition", *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, pp. 1300-1304, 2021.

C3. **E. Rahimian**, S. Zabihi, A. Asif, and A. Mohammadi, "Hybrid Deep Neural Networks for Sparse Surface EMG-Based Hand Gesture Recognition", *Asilomar Conference on Signals, Systems, and Computers*, pp. 371-374, 2020.

C2. **E. Rahimian**, S. Zabihi, S.F. Atashzar, A. Asif, and A. Mohammadi, "Xceptiontime: Independent Time-window Xceptiontime Architecture for Hand Gesture Classification", *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 1304-1308, 2020.

C1. **E. Rahimian**, S. Zabihi, S.F. Atashzar, A. Asif, and A. Mohammadi, "Semg-based Hand Gesture Recognition via Dilated Convolutional Neural Networks", *IEEE Global Conference on Signal and Information Processing (GlobalSIP)*, pp. 1-5, 2019.

Chapter 2

Literature Review

The main focus of this chapter is on different myoelectric pattern-recognition control algorithms, especially DNN-based approaches. As shown in Fig. 2.1, myoelectric control algorithms can be classified into three main categories, namely:

- Conventional myoelectric control systems: Conventional myoelectric control systems (e.g., on/off control or direct-proportional) offer several benefits such as straightforward implementation. However difficulties such as limited Degree of Freedom (DoF) due to crosstalk have lead researchers to focus on improving myoelectric control systems by adopting the other categories.
- Pattern recognition control systems, which is the main focus of this research project and will be described in details.
- Systems that map decoded MUAP pulse trains to control commands. This category encompasses methodologies that map the extracted neural code from sEMG to a proportional command signal.

Fig. 2.2 illustrates the schematic of a myoelectric pattern-recognition control pipeline, which provides a platform for dexterous control of robotic arms and prosthesis. Generally speaking, pattern recognition control systems can be classified into:

(a) ML-based approaches,

(b) DNN-based methods (Fig. 2.1).

Myoelectric control techniques belonging to the former category (ML-based) typically, consist of data acquisition, data segmentation, feature extraction, and classification

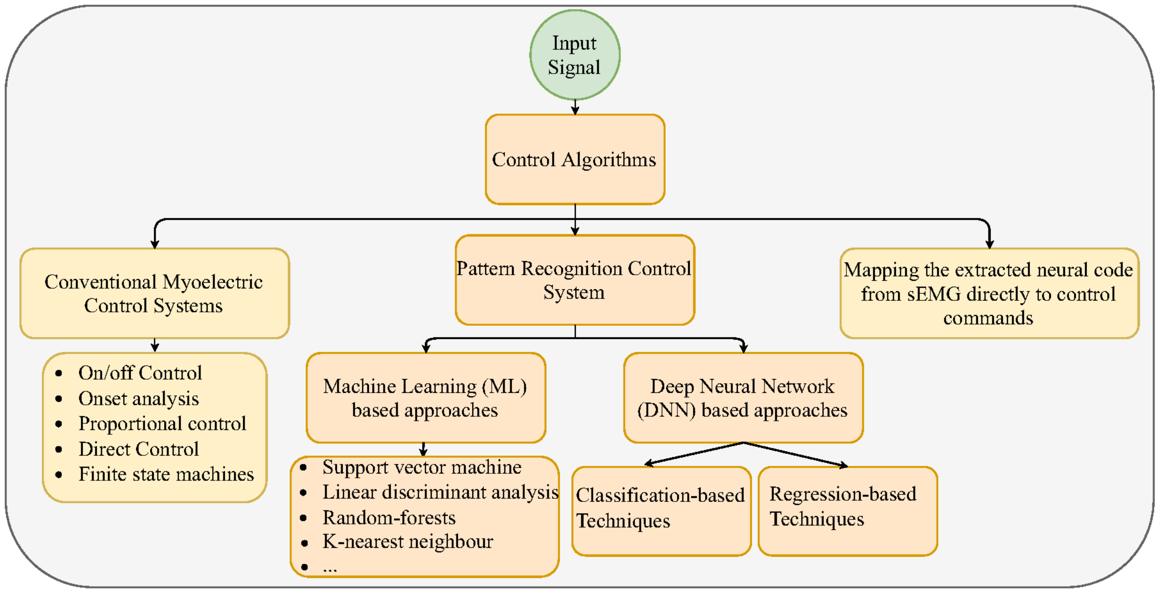

Input Signal
Control Algorithms
Conventional Myoelectric Control Systems
Pattern Recognition Control System
Mapping the extracted neural code from sEMG directly to control commands
On/off Control
Onset analysis
Proportional control
Direct Control
Finite state machines
Machine Learning (ML) based approaches
Deep Neural Network (DNN) based approaches
Support vector machine
Linear discriminant analysis
Random-forests
K-nearest neighbour
...
Classification-based Techniques
Regression-based Techniques

Figure 2.1: Different myoelectric prosthetic control algorithms. Our main focus in this research is on the pattern recognition control systems, specially the DNN-based approaches (the orange blocks). The non-pattern recognition control systems (the yellow blocks) are briefly discussed in this section.

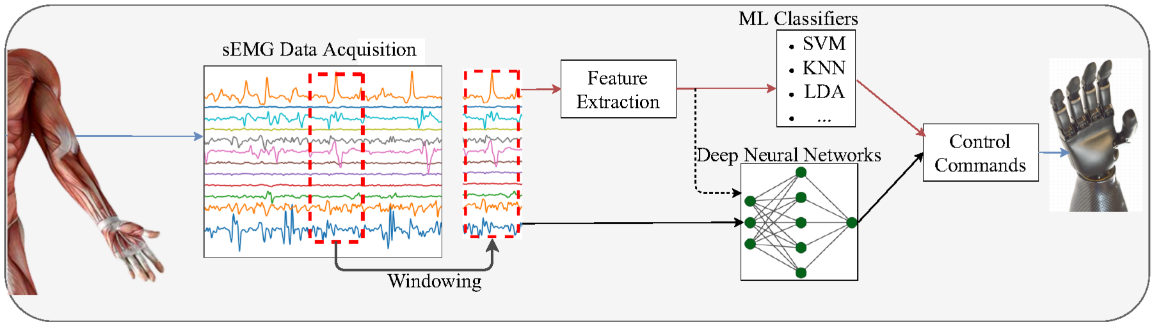


Figure 2.2: Schematic illustration of myoelectric pattern-recognition control. This kind of control can be done by classical ML or DNN approaches, where after sEMG data acquisition, at first raw signals are segmented by a window. For classical ML approaches, HCFs are extracted and then fed to the ML classifier. For DNN approaches, however, either raw sEMG signals or HCF features can be used as the input for DNNs.

steps. Such traditional ML approaches have been thoroughly explored in the literature [14]. Hand Crafted Features (HCFs) are, typically, extracted from sEMG signals, which are then used as inputs for ML classifiers such as Support Vector Machine (SVM), Linear Discriminant Analysis (LDA), K-Nearest Neighbours (KNN), and random-forests, to name but a few [14].

While pattern-recognition control models developed based on conventional ML techniques have shown promising results, it is important to develop control models that are capable of meeting specific requirements of clinical practice, including: (i) high accuracy, (ii) robustness, and (iii) capability of adaptation to environmental changes. Therefore, recent evolution of neural networks coupled with advancements in rehabilitation and assisted technologies have created a trending shift from classical ML-based approaches to DNN-based myoelectric control systems (Category (b) above). Here, we refer to DNN-based myoelectric control systems as "*deep MyoLearning*". These systems rely on the fact that the sEMG signal patterns are specific and repetitive for particular movements. As such, several research works have focused on development of advanced deep MyoLearning models for the task of myoelectric pattern-recognition control. Deep MyoLearning can be categorized into regression-based techniques (presented later in Section 2.2) and classification-based techniques (described below in Section 2.3). Advanced deep MyoLearning technologies are potentially capable of achieving the aforementioned requirements of clinical practice for adoption in real-world applications.

In the following subsections, first, we provide a brief overview of public databases as a unified platform to train and evaluate new deep MyoLearning models. Second, we present state-of-the-art deep MyoLearning solutions together with their challenges and opportunities for successful translation into practical settings.

2.1 Myoelectric Control Prosthesis Supporting Databases

Table 2.1 summarizes public scientific benchmark datasets that can be utilized to develop, implement, and evaluate new DNN/ML architectures. These databases are introduced for movement recognition and force control with the ultimate goal of facilitating development of non-invasive and naturally controlled robotic hand prosthesis. The main focus of these datasets is to achieve one or more of the following objectives: Provide a common evaluation platform, along with a repository of myoelectric signals [15]; Increase the number of both non-disabled and amputated subjects participating in the data collection [16–18]; Increase the type and number of gestures [16]; Collect sEMG signals with different acquisition setups [19], and; Increase the number of sessions to explore repeatability in sEMG data and providing a unified platform

Table 2.1: Public available sEMG datasets for development of advanced Pattern Recognition Control.

Reference	Year	Number of gestures	Intact subjects	Amputated subjects	Number of sEMG channels	Number of Movement Repetitions	Repetitions for training	Repetitions for testing	Sampling rate	sEMG setup	Sessions for each subject
NinaPro DB1 [17]	2013	53 (rest included)	27	0	10	10	1,3,4,6,8,9,10	2,5,7	100 Hz	Otto Bock 13E200	1
NinaPro DB2 [16]	2014	50 (rest included)	40	0	12	6	1,3,4,6	2,5	2000 Hz	Delsys Trigno	1
NinaPro DB3 [16]	2014	50 (rest included)	0	11	12	6	1,3,4,6	2,5	2000 Hz	Delsys Trigno	1
NinaPro DB4 [19]	2017	53 (rest included)	10	0	12	6	1,3,4,6	2,5	2000 Hz	Cometa + Dormo	1
NinaPro DB5 [19]	2017	53 (rest included)	10	0	16	6	1,3,4,6	2,5	200 Hz	Thalmic Myo armbands	1
NinaPro DB6 [21]	2017	7	10	0	14	120	1,3,...,119	2,4,...,120	2000 Hz	14 Delsys Trigno	5
NinaPro DB7 [18]	2017	41 (rest included)	20	2	12	6	1,3,4,6	2,5	2000 Hz	Delsys Trigno IM	1
NinaPro DB8 [22]	2019	10 (rest included)	10	2	16	12	1,2,...,10	11,12	1111 Hz	Delsys Trigno IM	1
MDS1 [23]	2020	10	30	15	12	4	4-fold cross validation (one fold is used for test)		1926 Hz	Delsys Trigno	1
BioPatRec DB1 [15]	2013	10	20	0	4	3	1	2	2000 Hz	in-house device	-
BioPatRec DB2 [15]	2013	27 (rest included)	17	0	8	3	1	2	2000 Hz	in-house device	-
BioPatRec DB3 [15]	2013	10	8	0	4	3	1	2	2000 Hz	in-house device	-
BioPatRec DB4 [15]	2013	8	8	0	16	3	1	2	2000 Hz	in-house device	-
CapgMyo [24,25]	2016	22	23	0	128 (HD)	10	1,3,5,7,9	2,4,6,8,10	1000 Hz	in-house device	1
SEEDS [26]	2019	13	25	0	126 (HD)+8	6	10-fold cross-validation)		2048 Hz	BioSemi ActiveTwo EMG amplifier	3
CSL-HDEMG [20]	2015	27	5	0	192 (HD)	10 (in every session)	5-fold cross-validation on the sessions (one fold is used for test)		2048 Hz	in-house device	5

for improving the robustness of robotic prosthetic hands [20, 21]. Other databases such as [22] are collected to be used for estimation/reconstruction of finger movement rather than motion/grip classification. Recently introduced dataset in [23] is a multimodal database including EMG, inertial, gaze tracking, visual, behavioral and clinical data, which provides a medium for prosthetic and phantom limb sensation analyses. Finally, datasets introduced in References [20, 24–26] consist of HD-sEMG measurements providing the foundation to develop and test DNN-based models based on the HD-sEMG signals.

2.2 Pattern Regression for Myoelectric Control System

Regression-based techniques (typically, via combination of several regressors) have been used to identify and estimate different movements in a continuous space across several DoFs. Unfortunately, the prediction accuracy of such techniques is not yet at a level to be used in practical settings. As an alternative, some researchers such as [27] adopt a regression CNN to estimate wrist motions based on sEMG signals. Unlike CNN-based classification methods, regression CNNs allow for independent and simultaneous control so that several DoFs can be manipulated concurrently with different magnitude. This allows for more dexterous and realistic prosthetic motion than low-DoF proportional control or discrete classification control. This CNN-based regression technique was validated by real-time control tests that demonstrated superior performance compared to SVM-based techniques [27]. Other research works (e.g., [28]) predict grasping force levels based on a combination of Principal Component Analysis (PCA) and DNN for control of a prosthetic hand. More specifically, dimension reduction of time domain features are achieved by PCA, and then a sEMG-force regression model is generated by using DNN. This is an important step toward improving the grip of prosthetic hand based on force prediction.

2.3 Pattern Classification for Myoelectric Control System

Table 2.2 summarizes the recent DNN-based algorithms for the task of pattern classification for multifunctional upper limb prosthesis control. DNN models are developed to address one or more of the following three objectives:

- Achieving high accuracy [2, 24, 29–32],
- Improving robustness [12, 25, 33, 34],
- Adaptability [25, 35–38].

We explain these three objectives along with a brief overview of recent efforts devoted to achieve each one.

2.3.1 High Accuracy

Several recent DNN-based pattern classification architectures (deep MyoLearning) have been motivated by the observation that a small group of muscles play a significant role in specific hand movements. Typically, the main focus of new DNN-based models for prosthesis control is to achieve the highest possible classification accuracy. High accuracy is indeed of paramount importance to achieve the ultimate goal of efficiently tracking both static and dynamic patterns and translating the classification results into smooth prosthesis actions. Early DNN-based works investigated the application of HCF coupled with different ML classifiers such as LDA, SVM, random forests, and KNN [16]. It was shown that among these traditional classification models, random forests provide the highest accuracy. Next, a transition from conventional ML models to deep MyoLearning took place, starting with design of simple DNN architectures. The first design step for construction of DNN-based architectures is to select the type of input for feeding deep learning models. Options include:

(i) Raw EMG signals resulting in an end-to-end deep model,

(ii) Time domain features,

(iii) Time-frequency features such as spectrograms using Short-Time Fourier Transform (STFT).

Table 2.2: State-of-the-art researches of pattern classification for prosthetic control.

	Reference	Input	Method
Foucs on Accuracy	[29]	Raw sEMG	Basic CNN
	[24]	Instantaneous sEMG images	CNN
	[30]	sEMG image	Attention-based hybrid CNN-LSTM
	[31]	sEMG image	Multi-stream CNN
	[32]	Hand Crafted Features	Multi-stream CNN
	[2]	Raw sEMG	TCN/ hybrid CNN-LSTM
Foucs on Robustness	**Reference**	**Input**	**Method**
	[33]	Hand Crafted Features	CNN + transfer learning
	[34]	Raw sEMG	CNN+ transfer learning
	[12]	Hand Crafted Features	Extreme learning machine with adaptive sparse representation classification
	[25]	sEMG image	CNN + unsupervised adaptation
Foucs on Adaptivity	**Reference**	**Input**	**Method**
	[25]	sEMG image	CNN + unsupervised adaptation
	[35]	Hand Crafted Features	TCN
	[38]	Raw EMG/Spectrograms/Continuous Wavelet Transform	CNN + transfer learning
	[36]	Hand Crafted Features	Canonical Correlation Analysis
	[37]	Hand Crafted Features	Bilinear Modeling

Initial approaches focused on utilization of DNN models in an end-to-end fashion without the need to use predefined HCFs [2,24,29–31] (Item (i) above). For example, it was shown in [29] that a simple CNN architecture (trained in an end-to-end fashion based on raw signals) is capable of providing comparable results to those obtained by classical ML techniques for classification of sEMG signals. Such results where further extended to HD-sEMG signals, where end-to-end CNN architectures are designed achieving significantly high accuracies [24]. Capitalizing on the high accuracy achieved by CNN architectures with HD-sEMG data, it was concluded that patterns present within HD-sEMG signals make them an ideal option for the development of highly accurate DNN-based pattern classification models [24]. Such DNN-based models can include one of the following structures: (a) CNNs; (b) RNNs, or; (c) combined architectures (i.e., hybrid CNN-RNN). More recently, the Temporal Convolutional Network (TCN) was adopted in [2], which not only provides several advantages over RNNs such as lower memory requirement and faster training but also increases the receptive field of the network capturing the temporal information of sEMG. Alternatively [31], defines the sEMG-based pattern classification as an image classification problem to take advantage of Multi-Stream (MS) CNN architectures. The sEMG image can be constructed based on traditional sEMG or HD-sEMG signals. HD-sEMG signals are sampled at a high frequency, therefore, the size of instantaneous sEMG images is the same as that of the electrode array used to record the HD-sEMG signals. However, traditional sEMG signals are collected at a low sampling rate, therefore, a sliding window of specific length is required to segment these signals and then convert them to sEMG images. Alternatively, some researchers such as [30] proposed hybrid CNN and RNN architecture to better capture spatial and temporal features of sEMG signals for pattern classification tasks. It is shown [30] that hybrid CNN-RNN architecture has a higher classification accuracy in comparison to networks in which only CNN is adopted when both HD-sEMG and sEMG signals are used as the input. Results obtained from hybrid DNN architectures compared favourably against those from state-of-the-art classical ML approaches, showing that deep MyoLearning outperforms its classical counterparts [30].

The most recent trend to further improve accuracy of pattern classification algorithms is the integration of HCFs and deep features (Items (ii)-(iii) above). Such algorithms [32] start by extracting HCFs, which are then used by DNN architectures to enhance the overall accuracy. Here, the focus is on learning from multiple feature sets instead of designing an end-to-end learning model (i.e., learning only from raw sEMG signals). For example, reference [32] first extracted multiple engineered feature sets from raw sEMG data, providing different representations of the underlying sEMG signals. Extracted HCF sets were then converted to an image, which was provided as the input to a CNN-based framework for performing pattern classification. Reference [39] concluded that combination of conventional feature extraction-based methods with advanced DNN models, when coupled smartly and in an efficient fashion, has the potential to achieve better results (compared to the case that either used alone).

Despite the aforementioned developments and although existing scientific publications report high accuracy, the existing gap can be attributed, in part, to the fact that classification accuracy of DNN-based models is affected by several factors not being considered fully in scientific domain. The underlying factors include: Effects of utilized pre-processing techniques; Nature of selected feature representations and classification model; Number of participants in data collection both in terms of intact and trans-radial amputees subjects; Number of gestures; Number of repetitions of each gesture (increasing the number of repetitions for each movement will result in fatigue); Acquisition setup ; Clinical parameters related to the amputation (e.g., remaining forearm percentage, phantom limb sensation, use of prostheses); Subject's characteristics; Electrode shifts; Effect of limb position, and; Non-stationary nature of sEMG signals varying over days. Therefore, improving the accuracy of DNN models is not the biggest hurdle limiting their use in clinical applications. Instead, there is an urgent need to focus on alternative ML/DNN models, e.g., compact DNNs with far less number of trainable parameters. To design a more proportional, intuitive, and dexterous prostheses, the robustness of a DNN-based model is considered in some research works, while others pay more attention to adaptability. Next, we review these two research directions.

2.3.2 Robustness

Enhancing robustness for control of myoelectric prosthesis [29] is an effective approach to bridge the gap between real-world usage and scientific developments under laboratory conditions. One of the factors that affects robustness of myoelectric control is electrode shift, which occurs during prosthesis donning and doffing. Therefore, some researches focus on compensating the effects of electrode shifts. Reference [33], for instance, is one of the state-of-the-art research works, which improves the robustness of a myoelectric control system by proposing an adaptive calibration of electrode array shifts via Transfer Learning (TL). More specifically, CNN along with TL is used to reduce the amount of sEMG data required, and to determine the patterns at the initial position of HD-sEMG electrode array. Then, overlapped regions between the training and testing images are identified and matched to estimate electrode array shifts in an unsupervised fashion. When this system was then compared with five classical methods, it was demonstrated to be robust to designated shifts, while the performance of other methods dropped when there were shifts in electrode array during the testing phase.

The use of the TL approach can, to some extent, overcome the issue of having insufficient training data for the system recalibrations. Along a similar path, a supervised approach based on TL and CNNs is constructed in [34] to improve robustness to electrode shifts. More specifically, a CNN network is pre-trained, and then fine-tuned with the data obtained after electrode shifting. The CNN-TL approach demonstrates superior performance over the architectures that train a CNN from scratch. Adverse effects resulting from electrode shifts can similarly be introduced by limb position changes. The authors in [12] proposed a robust Sparse Representation Classification (SRC) model that is more robust to untrained effects, such as limb position changes, that affect myoelectric prosthesis performance.

Another direction of recent research is to introduce robustness against variation of HD-sEMG signals between different sessions, which is observed even for the same user in the same experimental conditions across different sessions. Although high recognition accuracy is achievable by using HD-sEMG data, the performance drops when the data is collected in different sessions. In this regard, a deep domain adaptation is proposed in [25], which is robust to the inter-session recognition settings. Furthermore, robustness enhancement is targeted by adopting a long-term approach instead of focusing on short-term conditions. Commonly, short-term recordings of sEMG data are used to train and evaluate different DNN-based frameworks. This is a critical barrier to achieve robustness because of the day-to-day variant nature of sEMG signals. Therefore, sEMG signals recorded on different days can be used to develop and test for robust DNN models. Despite recent targeted focus on improving robustness in DNN-based myoelectric control models, more research is still needed, particularly on the design of new classification models, a unified framework for TL and domain adaptation, and long-term robustness.

2.3.3 Adaptability

Employing adaptive methods with a focus on transferring information and knowledge between source and target domains (despite their inherit differences) is another attractive direction for addressing the gap between scientific achievements and clinical application of myoelectric prosthesis between these domains. More precisely, the goal of adaptive approaches is to develop a learning algorithm that combines the prior knowledge and experience gathered from the source domain with new information. Generally speaking, the following two issues of prosthetic hand control can (partially and possibly) be addressed by adaptive approaches:

(i) *Training Time*: The first problem is the extended training time required by the end-user to mitigate the differences between the desired and performed movements. Such a training process, which is time consuming, tedious and unpleasant, can take up to several days.

(ii) *Variability in the Nature of sEMG Signals*: The second issue is the variability in the nature of the sEMG signals. This variability is caused by:

   – The time-dependent and stochastic nature of human neurophysiology, which changes the mapping between neural drive and the sEMG recording over time (partly due to fatigue).
   – The dependency of neurophysiology on the dynamics and kinematics of tasks.
   – The variability of neurophysiology between different users and the changes caused by amputations.

   In addition, sEMG recordings vary based on electrode location. Given such variations, therefore, probability distributions of sEMG signals obtained at each time could be completely different from another observation. Consequently, models trained based on some specific observations may not consistently and directly be reused over time. This would require retraining and recalibration, which makes it difficult for the user. In many cases, this challenge limits the use of these advanced myoelectric prostheses to research laboratories, where calibration and recalibration can be done recurrently.

Capitalizing on potentials of adaptive DNN-based solutions, there has been a recent surge of interest on adaptability of pattern classification DNN models. For example, in [25], the authors defined HGR as a multi-source domain adaptation problem in which the HD-sEMG signals are collected through multiple sessions. The proposed deep domain adaption algorithm starts to adapt while the device is being worn and stops when it is removed by the user; therefore, it is compatible to new sessions or subjects.

As another example, the authors in [35] not only used TCNs to attain high performance, but also utilized adaptive reinforcement training to integrate previously unseen classes into model predictions. TL is another method that can be used to reduce training data by taking advantage of inter-subject data for the purpose of sEMG-based gesture recognition [38]. More precisely, the data-hungry nature of DNN models is addressed by mapping a function between source and target domains, as such facilitating pattern classification for a new user by pre-training the model on multiple subjects.

Before the emergence of deep learning, the second issue was tackled intelligently in [36, 37] via feature engineering rather than using different twists of data and classifier combinations. More specifically, they proposed the multi-user myoelectric that can adapt to new users. Incorporation of such methodologies within the context of DNN-based frameworks could potentially be a fruitful direction for future research.

2.4 Conclusion

As discussed previously in Section 2.3, concentrated efforts have been devoted to provide a pattern recognition control system for upper limb prosthesis with the following characteristics: (i) high accuracy, (ii) robustness (e.g., against electrode shift or limb position), and; (iii) adaptability. Another topic of recent focus is simultaneous and proportional control through use of regression-based techniques (discussed in Section 2.2). In addition, another key factor is the feasibility of embedding DNN-based models into prostheses controllers. Computational cost requirements needed to train DNN-based models can easily exceed many thousands of parameters, especially with the very deep models. As a result, we are still far away from embedding a model with thousands of parameters in a prosthetics device, and more research on compact DNN models is required. Despite extensive progress in the field of myoelectric prosthesis, however, there are still limitations that should be addressed to achieve a more intuitive upper limb prosthesis.

Chapter 3

Accurate and Efficient sEMG-based Deep Learning Models for HGR

The problem of accurately and efficiently (computation-wise) recognizing a (possibly large) set of hand gestures using sparse multichannel sEMG signals for myoelectric control of neuro-prosthetic robotic limbs is still far from being solved. Motivated by potentials of deep learning models in improving performance of such systems, this chapter focuses on development of hand gesture recognition via deep learning-based architectures. In this context, first in Section 3.1, we focus on improving gesture recognition accuracy. In Section 3.2, then, we focus on reducing gesture recognition complexity.

3.1 Improving Gesture Recognition Accuracy

While deep learning models have great potentials to improve the overall hand gesture recognition accuracy, their performance for accurately recognizing a set of hand gestures is still far from being acceptable. This section addresses this gap by proposing two novel deep learning architectures, namely the Hybrid Recognition Model (HRM) (Sub-section 3.1.2), and the Temporal Convolutional Network Model (TCN) architecture (Sub-section 3.1.3). The proposed HRM framework aims to enhance accuracy of myoelectric systems with the eventual goal of realizing an accurate and resilient man-machine interface for myocontrol of neuro-robotic systems. In contrary to the existing data-driven approaches, which are commonly developed based on a single deep model, the proposed hybrid architecture consists of two parallel blocks (one convolutional and one recurrent). A fully connected multilayer fusion neural network, which acts as the fusion centre, combines the output of the two parallel blocks to form the target gesture classification. Such a hybrid architecture is specifically designed to separately process temporal and spatial features via two parallel learning pipelines to augment the discriminative power of each individual model.

In practice, training speed of a designed deep model, especially its memory usage, is as important as its achievable accuracy. To maintain (or even improve) the achieved accuracy of the HRM architecture, the TCN framework is developed based on dilated causal convolutions, which increase the receptive field of the network. In other words, the training step of the TCN architecture is considerably faster than that of HRM due to the absence of recurrent connections. Moreover, different from HRM and existing deep learning methods, by applying Conv1d, the TCN network itself can extract the hidden correlations existing between different signal sequences.

3.1.1 Material and Methods

Before presenting the proposed hand gesture recognition architectures, below, we discuss the utilized dataset for training purposes followed by the pre-processing step used to prepare training, validation, and test sets.

Database: For development of the proposed DNN architectures, we used the second dataset from Ninapro [16], which is called the DB2. Ninapro is a publicly available hand gesture recognition dataset widely used in this field, which is collected via Delsys Trigno Wireless EMG System with 12 wireless electrodes (channels). The electrical activities of the muscles are recorded at the sampling rate of 2 KHz. The DB2 dataset is collected from 40 healthy individuals performing 50 different gestures such as hand, functional, wrist, and grasping movements coupled with different force patterns. The population used for data collection consists of 12 females and 28 males within 29.9 ± 3.9 age range. The DB2 dataset consists of three different exercise sets referred to as Exercise set B, C, and D, where individuals participating in the data collection task perform repeated movements (6 times, each lasting for 5 seconds). Each movement task is followed by a rest period of 3 seconds. Here, we use Exercise B to design and implement the proposed HGR models (i.e., the HRM and the TCN). Exercise B consists of 17 different movements. As shown in Fig. 3.1, the utilized dataset

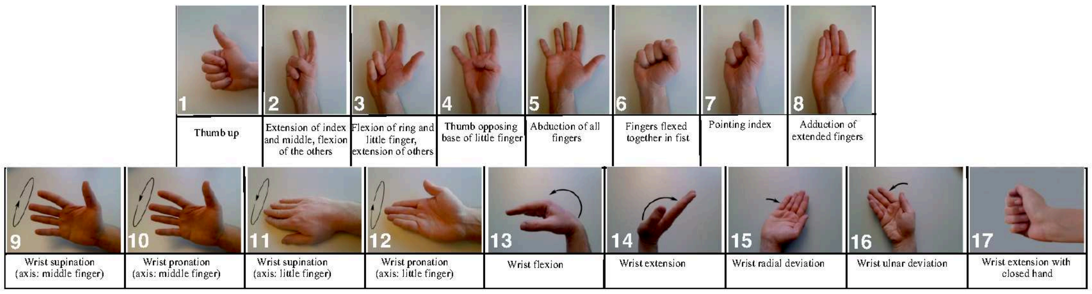

1
Thumb up
2
Extension of index and middle, flexion of the others
3
Flexion of ring and little finger, extension of others
4
Thumb opposing base of little finger
5
Abduction of all fingers
6
Fingers flexed together in fist
7
Pointing index
8
Adduction of extended fingers
9
Wrist supination (axis: middle finger)
10
Wrist pronation (axis: middle finger)
11
Wrist supination (axis: little finger)
12
Wrist pronation (axis: little finger)
13
Wrist flexion
14
Wrist extension
15
Wrist radial deviation
16
Wrist ulnar deviation
17
Wrist extension with closed hand

Figure 3.1: Hand and wrist movements (Exercise B) [16].

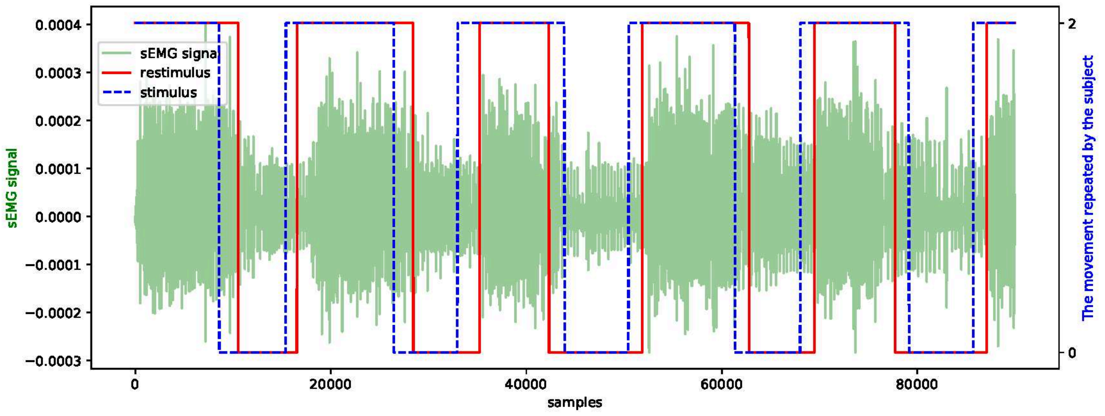


Figure 3.2: Movement 2 performed by Subject 1. sEMG signal are illustrated with the green color. The stimulus signal is shown with blue color, showing the movement repeated by the subject in the "Prior dataset". The restimulus signal is shown with red color, showing the refined movement repeated by the subject in the "Posterior dataset".

includes 9 basic movements of the wrist together with 8 isometric and isotonic hand movements. Another aspect of the DB2 dataset is the availability of the second set of labels, where these refined labels are post-processed in order to more accurately represent the actual movement. In what follows, we refer to the original labels as the *Prior Dataset*, while the refined labels are referred to as the *Posterior Dataset*.

The process of labeling the Prior dataset is performed based on the timestamps generated by the stimulus signal. In Fig. 3.2, the stimulus signal is shown with the dashed blue line for Movement 2 performed by Subject 1. As shown in Fig. 3.2, each repetition of Movement 2 lasts for 5 seconds, followed by a rest period of 3 seconds. In this case, the duration of trials for the active signals are exactly equal to 5 seconds or 10,000 samples. However, some delays caused by the volunteer's reaction time tended to create mismatches and frequently incorrect labels for some samples at the beginning and end of each movement. To improve the signal's labeling, the Posterior database is constructed, where labels are refined a-posteriori to only represent real movements. As illustrated in Fig. 3.2, the restimulus signal, which is shown via solid red lines, represents the actual movements. As shown in Fig. 3.2, in the "Posterior database" the duration of trials for the active signals are not the same as individuals have different response time to react to the incoming stimulus.

We follow the recommended configuration provided by the dataset producers and use Repetitions 1, 3, 4, and 6 of the *Prior Dataset* for training purposes. Repetitions 2 and 5 from the *Prior Dataset* and the *Posterior Dataset* are then utilized to test the trained models.

Pre-processing Step: Before implementation and training the two proposed architectures, we need to pre-process the input multi-channel sEMG dataset to remove noise, artifacts, and also to transform the signals in a particular format to be used as the input to each of the proposed architectures. In this regard, we pre-process the Exercise B data from DB2 by applying a 4$^{\text{th}}$ order Butterworth low-pass filter with a 20 Hz cutoff frequency. This filtering step is then followed by a normalization step using the Z-score approach. To compare the proposed method with previous literature and also to satisfy the acceptable delay proportional to motor intention reaction time (which should be under 300 ms [40], the sEMG data for each channel is segmented by a window of length of 100 ms (200 samples per window). Furthermore, we consider sliding window with steps of 10 ms to construct the training, validation and test data.

This completes a brief introduction to the utilized dataset and the pre-processing step. Next, we introduce the proposed HRM followed by the design of the TCN in Section 3.1.3.

3.1.2 Hybrid Recognition Model (HRM)

Fig. 3.3 illustrates a schematic of the proposed hybrid HRM architecture. The input to the HRM is multi-channel sEMG signals. The multi-channel sEMG signal space is used as the input to the network while applying the sliding window strategy mentioned in Sub-section 3.1.1. The input of the model, therefore, is a two-dimensional matrix composing of r_{tw} number of columns (representing the size of the sliding window), and r_s number of rows (representing the number of channels (sEMG sensors)). The

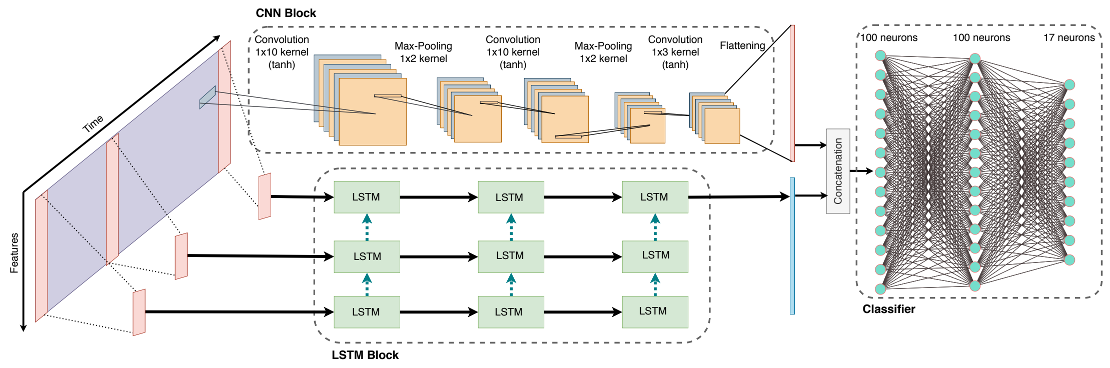


Figure 3.3: (a) The schematic of the proposed hybrid architecture, which consists of three main blocks called the CNN Block, the LSTM Block, and Classifier Block.

proposed HRM architecture consists of three main blocks, i.e., the CNN block; the LSTM block, and; the Fusion block. The details of these blocks are described in details below:

(i) **The CNN Block**: This block is used to extract informative spatial features from multi-channel sensory data. For this purpose, although, 2-D convolutional layers are used in the model, the kernels have one dimensional (1-D) structure. Using 1-D kernels within the CNN block makes the extracted features independent from each other. In other words, the sensors' data arrangement in the segmented input window will not affect the final CNN block's extracted features. This part of the model is constructed by three convolutional layers. The first and second CNN layers both comprise of 10 kernels with size of (1×10). Each of these two CNN layers are then followed by Max-Pooling layers with the filter of size of (1×2). The third and last CNN layer has one filter with size of (1×3). After each convolutional layers, *tanh* activation function is used to add non-linearity to the model.

(ii) **The LSTM Block**: Many to one sequential mapping structure is employed for the LSTM path, which is used in parallel to the CNN block to extract the hidden temporal correlations existing in multi-channel sEMG time series. The LSTM block consists of 3 LSTM layers. The first two layers of the LSTM consist of 32 cell structures followed by the third LSTM layer having 64 cell structure.

(iii) **The Fusion Block**: The features extracted by the CNN path is flattened and then concatenated with the temporal features obtained by the LSTM block.

Finally, the resulting vector is fed to the final block of the model, referred to as the Fusion Block. This part of the proposed model is composed of three fully connected layers, which allow the features resulted from the CNN and the LSTM blocks to be incorporated by the proposed architecture. Both the first and the second fully connected layers have 100 neurons followed by Rectified Linear Unit (ReLU) function while the third layer consists 17 neurons representing the 17 output gesture classes.

It is worth clarifying the rational behind designing the HRM as parallel architecture and not in a series format is that:

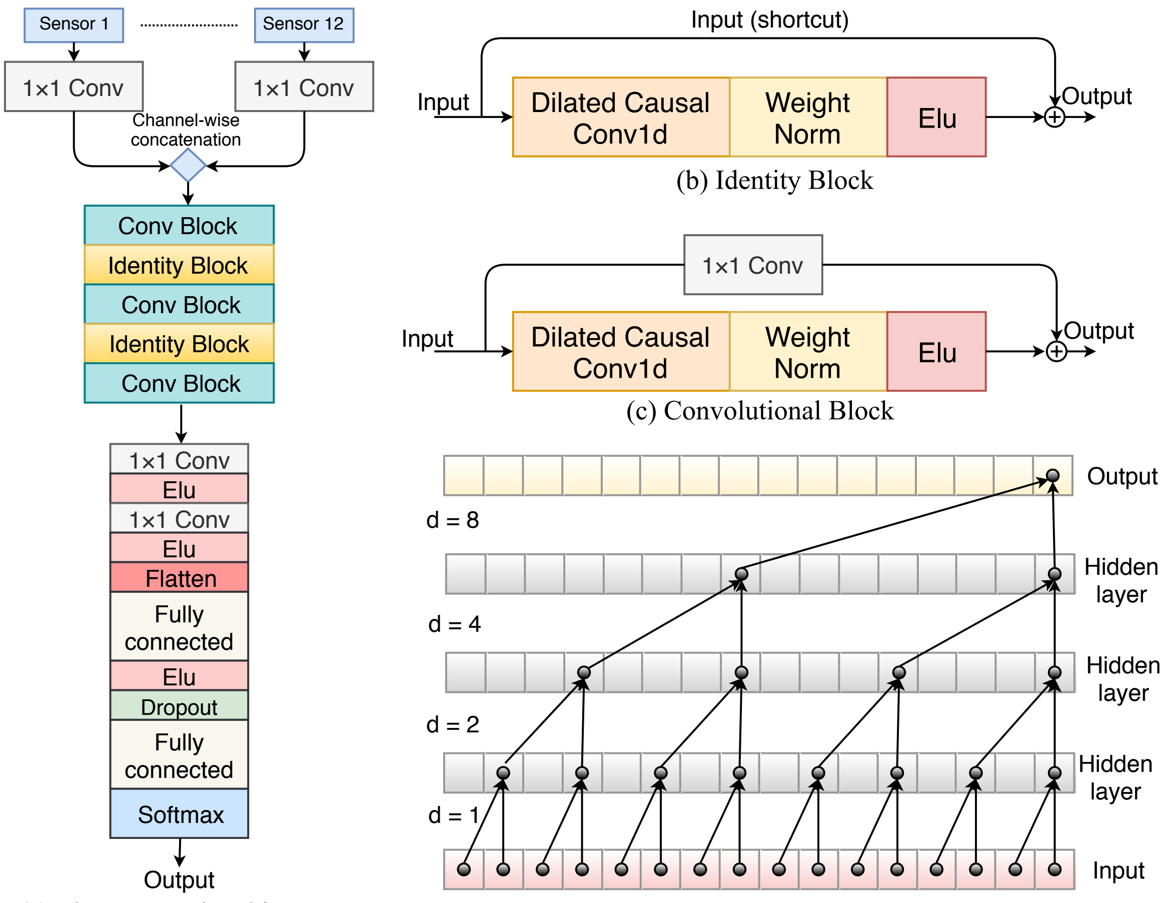
(a) Using a series architecture will increase the latency of the overall model as the second model needs to wait for computations of the first model to be completed, which is a negative factor for real-time HGR tasks; and,

(b) The error will more boldly propagate through the network once the output features of one model are used as the input to the second one.

In the proposed parallel architecture, however, both the CNN and the LSTM paths start with raw measurements as such error from one model will not propagate to the second one directly. This completes the presentation of the proposed HRM. Next, we present the TCN.

3.1.3 Temporal Convolutional Network (TCN)

In this section, we present details of the second model, the TCN architecture, which can be considered as a compact DNN in comparison to the HRM and as such has a faster training phase. In the TCN, different from the HRM, after the pre-processing step, sEMG signals from the recording sensors (12 sensors used in the collection of the DB2 dataset) are provided, separately, as inputs to a (1×1) convolution layer with 10 kernels. The features are concatenated channel-wise, which are then fed to the first block of the TCN architecture. This input strategy will result in increasing the number of input features, i.e., with signals from 12 sensors and using 10 kernels we end up with 120 input features. In other words, this approach allows us to train 120 different features on the first layer of the network. Furthermore, as instead of using Conv2D, we employed Conv1D, the network is equipped with the capability to extract



(a) The proposed architecture

(b) Identity Block

(c) Convolutional Block

(d) Dilated Causal Convolution

Figure 3.4: (a) The schematic of the TCN architecture. (b) The identity block. (c) The Convolutional block. (d) The dilated causal convolution (dilation factor d is equal to $[1, 2, 4, 8]$ for the layers). The receptive field of the last neuron in the output layer is shown in "pink".

hidden correlations between different signals without using the permutation approach introduced in Reference 41. Fig. 3.4(a) illustrates the proposed TCN architecture. As can be seen from Fig. 3.4(a), the proposed architecture consists of two main building blocks, i.e., the Identity block (shown in Fig. 3.4(b)), and the Convolutional Block (shown in Fig. 3.4(c)) which includes Dilated Causal Conv1D (shown in Fig. 3.4(d)). These blocks are designed as follows:

(i) **The Identity Block**: This Block, shown in Fig. 3.4(b), is one of the main blocks of the TCN architecture. This block consists of the following components:

– *Shortcut (Skip Connection)*: Although very deep networks are capable of extracting highly complicated features, using deep network may not always lead to achieving a better result. This is because highly deep networks, typically, suffer from the issue of vanishing gradient, referred to as the *Degradation* problem. In this work, by inspiring from ResNets [42], we address the degradation problem by adding a shortcut or a skip connection to the identity block, where the dimension of input and output are identical.

– *Dilated Causal Convolutions*: A critical issue for the time-series modeling of sEMG signals for hand gesture recognition is how to prevent leakage of data from future samples to past instances. Causal convolutions [43, 44] are used to address this problem. Another advantage of incorporating the causal convolutions is that they do not have recurrent connections. Therefore, the required training time is substantially less than that of RNNs. Despite having these benefits, however, the receptive field of causal convolutions is small. By inspiring from [43, 44], we used dilated causal convolutions within the Identity block of the proposed architecture to address the issue related to the limited receptive field of casual convolutions. Fig. 3.4(d) illustrates how dilated causal convolutions can increase the receptive field. In general, for one dimensional sEMG input sequence $\mathbf{x} \in \mathbb{R}^n$ with a filter $f : \{0, \ldots k-1\} \rightarrow \mathbb{R}$ on element s of the sequence, dilated causal convolution DCC is defined as follows:

$$DCC(\mathbf{x} \ *_d \ f)(s) = \sum_{i=0}^{k-1} f(i) \ . \ \mathbf{x}_{s-d \cdot i}, \tag{1}$$

where d is the dilation factor, k is the filter size, and $s - d \cdot i$ shows the past direction. When the depth of the model is increased, the dilation factor d is exponentially increased; i.e., $d = 2^l$ at layer l of the model. More specifically, the dilation factors d is exponentially increased by a factor of

2 in each layer $l = 1 \ldots L$ where L is the number of layers of the dilated causal convolutions in the proposed TCN model (Figure 3.4). The dilation factor in the TCN model is as follows:

$$d \in \left[2^0, 2^1, \ldots 2^{L-1}\right] \tag{2}$$

A dilated convolution reduces to a regular convolution when $d = 1$.

– *Weight Normalization*: To increase the training process and following the intuition provided by Reference [45], weight normalization beside normalizing the input signals are utilized within the Identity Block.

– *Activation Function (ELU)*: Exponential Linear Unit (ELU) [46] is used as the nonlinear activation function and is the last component within this block.

This completes the description of the Identity block. Next, we present the Convolutional Block.

(ii) **The Convolutional Block**: This block is employed when the input and output dimensions do not match. This happens as the number of filters used within the dilated causal components can vary from one block to another, which in turn results in a miss-match between the dimension of the input sequence and that of the output sequence. For instance, such a miss-match happens when, for example, 64 filters are used in the first block while the number of filters utilized in the second block is 128. To deal with such scenarios, the Convolutional block is introduced. The difference between this block and the identity block is that there is a (1×1) convolution layer in the shortcut path, which resizes the input to a different dimension to match up with the output sequence. Fig. 3.4(c) illustrates a Convolutional block.

(iii) **TCN Network Structure**: As stated previously, the signals provided as inputs into the TCN architecture are separately fed to a (1×1) convolutions layer with 10 kernels. The outputs are then concatenated channels-wise and fed to the first residual block of the network. The proposed architecture has 3 Convolutional Blocks and 2 Identity Blocks together 2 (1×1) Convolutions layers and 2 fully connected layers. The details of the TCN architecture are provided below:

- The first Convolutional Block has 64 kernels with size 5. The dilation factor for dilated causal convolutions is set equal to 1.
- Similarly, the first Identity Block of the architecture has 64 kernels with size 5. On the contrary to the previous convolutional block, here the factor for the dilated causal convolutions is set equal to 2. Therefore, the output neurons in this block have extended the receptive field.
- The second Convolutional Block has 128 kernels with size 5 and an increased dilation factor of 4.
- The second Identity Block has 128 kernels with size 5 with now dilation factor of 8.
- The third Convolutional Block has 256 kernels with size 5 and dilation factor of 16. The receptive fields of the blocks are increased as we move deeper into the network.
- The outputs from the third Convolutional Block, which has 256 channels, are fed to (1×1) Convolutions with 64 kernels, and then ELU activation function is applied on the output.
- There are another (1×1) Convolutions followed by ELU activation, which reduces the size of the input channels to 2.
- The first fully connected layer reduces the size of its input features to 150.
- The second fully connected reduces further reduces the size of the features, i.e., from 150 features to 17, which is equal to the number of classes.

We used Adam optimizer as the optimizer algorithm with a learning rate set of 0.001. The learning rate changes in a cycle with a length of 100 epochs. After 20 epochs, we divided the learning rate by 2, but after 100 epochs instead of dividing the learning rate by 2, we multiply it by 14.4. Therefore, the learning rate at the beginning of each cycle will be 90% of the learning rate at the beginning of the previous cycle. Finally, the dropout[47] is set equal to 0.3.

3.1.4 Experiments and Results

In this section, we evaluate the performance of the proposed HRM and TCN architectures based on different criteria and provide comparisons with recent state-of-the-art results [48] obtained based on the same dataset.

Existing research works [24, 30, 38, 49] used different datasets for evaluation purposes ranging from private ones, e.g., Reference [49], or public datasets such as Ninapro, *Capgmyo* [24], or *CSL-HDEMG* datasets. Ninapro consists of sparse multichannel sEMG signals, while the Capgmyo and CSL-HDEMG are based on HD-sEMG signals. As stated previously, the focus of the thesis is on hand gesture recognition via sparse multi-channel sEMG. Therefore, we used Ninapro database. More specifically, in this part, we focused on classifying 17 hand and wrist movements of Exercise B from DB2 database and followed the recommended configuration provided by the dataset producers to construct train and test sets. For evaluation purposes and to have a fair comparison, the following criteria needs to be considered:

(i) The results should be evaluated over the same dataset;

(ii) Training and testing sets must be similar for fair comparison, and;

(iii) The window length used to segment sEMG signals should be the same.

Although References [24, 30, 38] used the Ninapro database, different sections such as DB1, DB3, DB5, or other sub-database of the DB2 database, i.e., Exercises C or D are used rendering fair comparison impossible. Some other works developed based on the Ninapro database, used different train/test sets other than the recommended ones or used other window lengths to segment the sEMG signals. Consequently, we have focused our comparison on Reference [48].

In brief, Reference [48] is considered a hybrid setting but with two CNN paths designed in parallel. The first CNN path (referred to as Block 1 or B1 in short) consisted of five convolution layers and two maximum pooling layers, while the second parallel CNN path (referred to as Block 2 or B2 in short) only has five convolution layers. The main architecture, referred to as the *C-B1B2*, which achieved maximum accuracy of 83.79% on the *Posterior Dataset*, no results are provided for the *Prior Dataset*. Reference [48] also considered alternative architectures of similar nature by changing different features of the network such as convolution kernel size. For

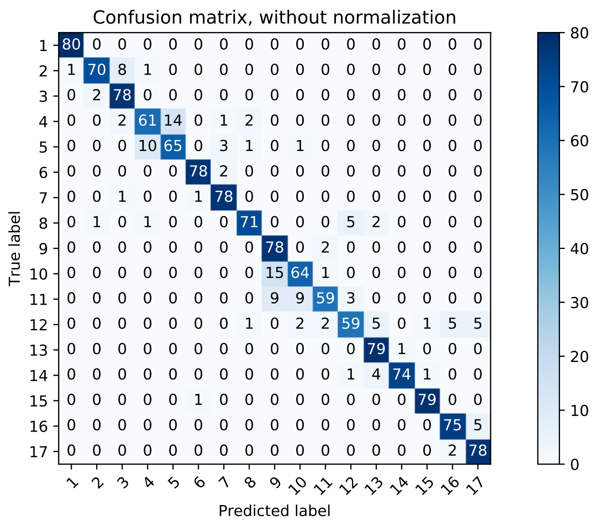
Table 3.1: The Accuracy between our model and Reference [48].

	Data	**Method**	**Accuracy** (%)
Ours	Prior data	Hybrid Model	98.01
	Posterior data	Hybrid Model	97.35
Ours	Prior data	TCN Model	90.14
	Posterior data	TCN Model	92.5
[48]	Prior data	-	-
	Posterior data	C-B1B2	83.79
		CC	77.96
		C-2B1	82.16
		C-2B2	82.49
		C-DK	79.23
		C-SK	78.52
		C-Sk2	81.3

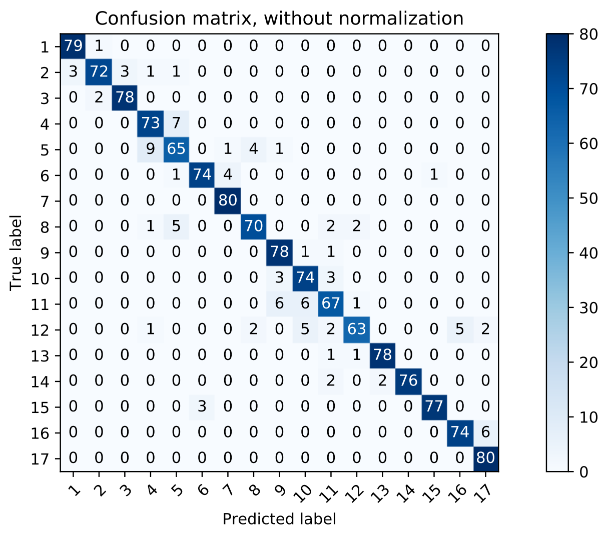
example, in the *C-2B1* model, *Block1* is used in parallel with itself, while in the *C-2B2* model, *Block2* is used in parallel with itself. In the *C-DK* model, convolution is conducted with a different kernel. Finally, models *C-SK* and *C-SK2* use smaller kernel sizes. Results based on the aforementioned variations are provided in Table 3.1 for comparison purposes.

We evaluate the proposed models based on both *Posterior* and *Prior* datasets. In computing the overall accuracies, we use trials (repetitions) to evaluate our architecture. Table 3.1 shows the average accuracy obtained from the proposed HRM architecture. It can be observed that the proposed hybrid architecture consisting of a CNN block designed in parallel with an LSTM path significantly outperforms its counterpart and improves the recognition accuracy from 83.79% to 97.35% on the *Posterior* data.

Our proposed HRM achieves 98.01% accuracy on the *Prior* dataset, which is even higher than the best result reported in Reference [48] for the *Posterior* dataset. We would like to point out that having better results on the *Prior* dataset in comparison to the *Posterior* dataset is due to the fact that the model is trained based on the



(a) Prior data.



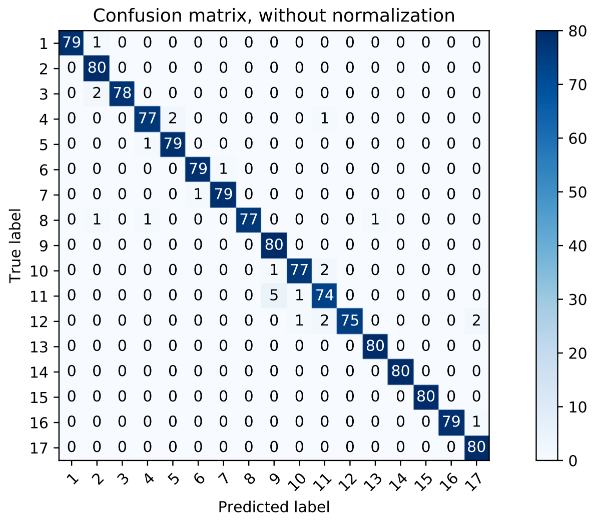
(b) Posterior data.

Figure 3.5: Confusion matrix of the HRM

Prior dataset. This further shows the superiority of the HRM in comparison to its counterpart, considering the fact that the results obtained from the *Prior* dataset are expected to be less than those obtained based on the *Posterior* dataset.

Along the same trend, the TCN improves the recognition accuracy from 83.79% to 92.5% on the *Posterior* data. It can be observed that the proposed architecture considerably improves the recognition accuracies in the *Posterior* dataset when compared to its existing solutions. As expected, accuracy on the posterior data is better than that obtained from the *Prior* dataset because the labels associated with the duration of the movement are refined, therefore, representing the real movements. Our proposed framework archives 90.14% accuracy on the *Prior* dataset, which is even higher than the best result reported in Reference [48] for the *Posterior* dataset.

The HRM's confusion matrix on the *Prior* dataset is obtained, as shown in Fig. 3.5(a), to analyze the classification performance on testing data. Fig. 3.5(b) shows the HRM's confusion matrix for the *Posterior* dataset. There are 17 rows (and correspondingly 17 columns) in the confusion matrices each associated with one of the 17 movements. We would like to point out that the total number of repetitions for each movement (we have the total number of 17 movements) in the test set (as specified by the DB2 dataset) is equal to 80. This is because, 2 repetitions are specified to be used for test purposes (i.e., Repetitions 2 and 5) each with 40 subjects. Therefore,


(a) Prior data.

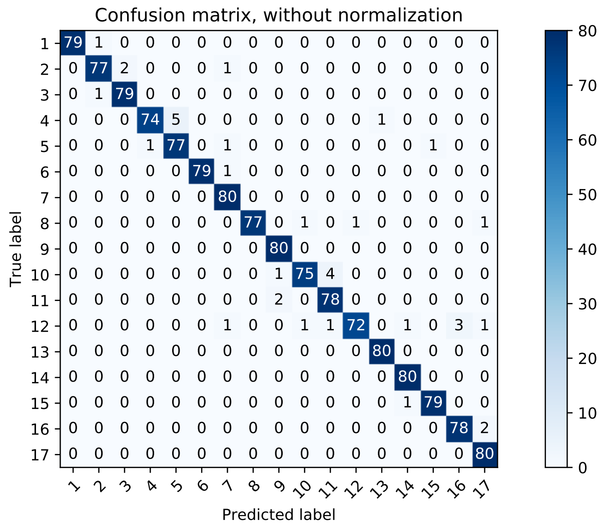

(b) Posterior data.

Figure 3.6: Confusion matrix of the TCN

in Figs. 3.5(a) and 3.5(b) the maximum number of the diagonal entries is equal to 80 showing the perfect classification of each movement. It can be observed that the diagonal entries in these confusion matrices obtained from the proposed architecture are fairly close to 80 indicating its superior classification performance.

Similarly, the TCN's confusion matrix on the *Prior* dataset is obtained, as shown in Fig. 3.6(a), while Fig. 3.6(b) shows the TCN's confusion matrix for the *Posterior* dataset. These results corroborate our earlier conclusion on the superiority of the proposed architecture.

This completes developments and discussions on proposed DNN-based architectures for the task of improving hand gesture recognition accuracy based on sEMG signals. Next, we shift the focus to designing a less complex network for the task at hand.

3.2 Reducing Gesture Recognition Complexity

This section considers the problem of reducing the number of parameters of the designed DNN-based model for HGR. In this context, a novel deep learning model, referred to as the XceptionTime architecture is proposed (Sub-section 3.2.2). The proposed XceptionTime is designed by integration of depthwise separable convolutions, adaptive average pooling, and a novel non-linear normalization technique. At the heart of the proposed architecture is several XceptionTime modules concatenated in series fashion designed to capture both temporal and spatial information-bearing contents of the sparse multichannel sEMG signals without the need for data augmentation and/or manual design of feature extraction. In addition, through integration of adaptive average pooling, Conv1D, and the non-linear normalization approach, XceptionTime is less prone to over-fitting, and more importantly is independent from the input window size. Finally, by utilizing the depthwise separable convolutions, the XceptionTime network has far fewer number of parameters resulting in a compact (less complex) network.

3.2.1 Material and Methods

In this section, first, the database on which the proposed model is evaluated is described. Then, the pre-processing approach for preparing the data set will be explained.

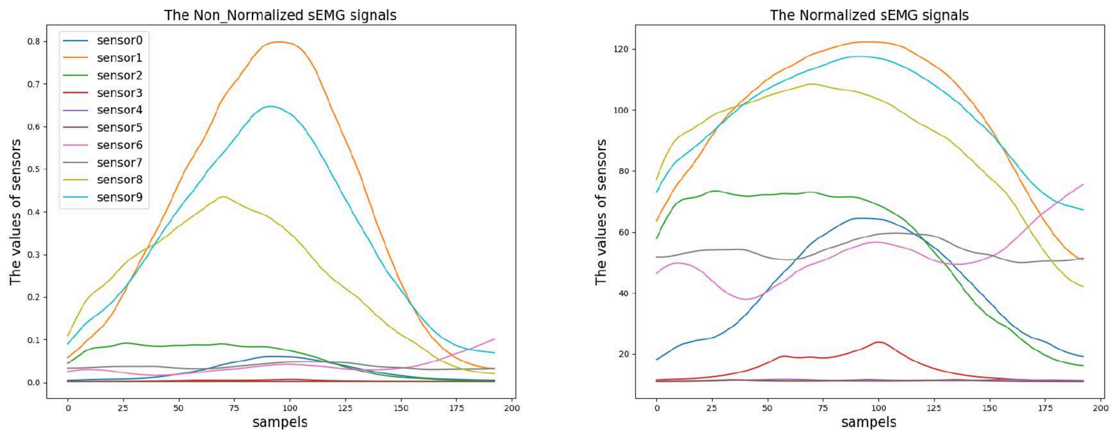
Database: Performance of deep learning techniques using sparse multichannel sEMG is yet far being optimal in terms of:

(i) Recognition accuracy;

(ii) Complexity of the system, and;

(iii) Sufficiency of number of subjects and movements.

Therefore, the proposed architecture will be evaluated on a public identified scientific benchmark database, Ninapro [16, 17], which is the most widely accepted benchmark for evaluation of different models developed based on sparse multichannel sEMG signals. The first Ninapro database [16, 17], referred to as the DB1, is used in this work, where the sEMG signals are acquired using Otto Bock MyoBock 13E200 with 10 wireless electrodes (channels) at a sampling rate of 100 Hz. The DB1 consists of 27 intact (healthy) subjects, where each subject has to repeat 52 gestures including finger, hand, and wrist movements. The subjects repeated each gestures 10 times, each time lasted for 5 seconds followed by 3 seconds of rest. For the sake of comparison and following the recommendations provided by the database and also previous



(a) (b)

Figure 3.7: The electrical activity of muscles obtained from 10 sensors: (a) The sEMG signals before normalization. (b) The sEMG signals after μ-law nonlinear normalization.

studies [16, 24, 29, 31], the testing set consists of repetitions 2, 5, and 7, where the remaining repetitions are considered as the training set. Evaluating the proposed model based on the sufficient number of subjects and hand gestures, shows its capability to generalize the results for practical use in daily life.

Pre-processing Step: Following the proposed preprocessing procedure in the previous studies [16, 24, 29, 31], we adopted a 1st order 1 Hz low-pass Butterworth filter to preprocess the electrical activities of muscles. However, we develop and propose a new approach for the normalization, referred to as the μ-law normalization, of sEMG signals in a nonlinear fashion based on μ-law transformation [50]. This normalization approach has been used traditionally in speech and communication domains for quantization purposes. We propose for the first time to use it for normalization in the context of sEMG processing. The μ-law normalization is performed based on the following formulation:

$$F(x_t) = \text{sign}(x_t)\frac{\ln\big(1+\mu|x_t|\big)}{\ln\big(1+\mu\big)}. \tag{3}$$

where x_t denotes the input scaler to be normalized, and $\mu = 256$ is utilized. The nonlinear normalization preprocesses the sEMG signals significantly better than linear normalization such as Minmax normalization, which is commonly used. In contrary to commonly used Minmax normalization, which linearly distributed signal values between the pre-defined range, the proposed μ-law normalization magnifies the outputs of sensors with small magnitude (in a logarithmic fashion), while keeping the scale with those sensors having larger values over time. As an illustrative example, Fig. 3.7 shows the 1 Hz low-pass filtered sEMG signals obtained from 10 sensors corresponding to the first repetition from Subject 1 performing the second gesture. As can be observed in Fig. 3.7(a), except from sensors 1, 8, and 9, the values of the remaining sensors are close to zero. However, by using the proposed μ-law normalization (as shown in Fig. 3.7(b)), the outputs of the sensors will be amplified more non-linearly. Moreover, compared to Minmax normalization, which is very sensitive to the presence of outliers, the μ-law normalization can handle the outliers. In summary, the μ-law normalization is a strategy to preserve relative changes among the sensor values while capturing the magnitude of the sensors.

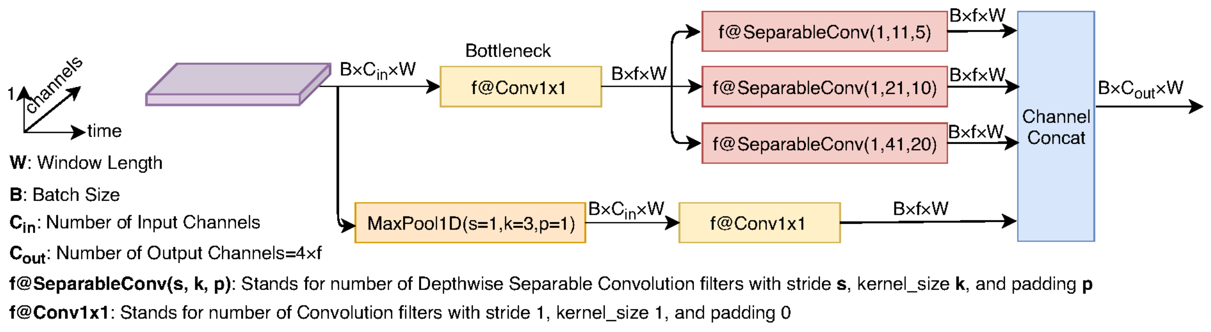
3.2.2 XceptionTime Architecture

To provide the related context, we should point out that a common strategy [24, 29, 31, 51] used for hand gesture recognition is to convert the multichannel sEMG recording over fix time windows into images and then use CNN-based image classification models to perform the recognition task. The problem with such an approach is that only the spatial information of sEMG signals are captured without considering the sequential nature of the sEMG signals. Motivated by this fact, Reference [30] proposed a hybrid CNN and RNN architecture where both spatial and temporal features of the sEMG signals are captured. However, in [30], raw signals are first converted to images (via six sEMG image representation approaches) and then fed to the hybrid CNN-RNN architecture. The results obtained in [30] show that accuracy in classification depends critically on the characteristic of the constructed images, revealing that there is still a major question what is the optimal approach for converting sEMG signals into images and if this is subject dependent [52]. Moreover, in this work [30], the algorithm proposed in Reference [41] is utilized, which fuses various signal sequences as an activity image used for training purposes. Although utilization of the aforementioned algorithm allows each sEMG sequence to be adjacent to all other sequences, this requires readjustment of the input signals adding to the complexity of the model.

To overcome these problems, in the Sub-Section 3.1.3 we developed a new composite architecture, the TCN [8], to eliminate the need for converting the raw sEMG signals into images. Such an approach directly fed the sEMG signals into the proposed temporal-convolutional network architecture capitalizing on the time-series nature of the underlying signals. Although this approach has its advantages, i.e., there is no need for readjustment, and the number of parameters is much less than their counterparts using RNN modules, high accuracy can only be achieved by using the complete sEMG sequence (a large window of sEMG sequence). On the other hand, the model in [53] is trained separately for each subject limiting its generalization capabilities to be used as a subject-independent model. Finally, in [32], the authors extracted 11 classical sEMG feature sets and then combined these features with a CNN framework. Although this can help with the computational expense of the technique, extraction of optimal engineered features and construction of optimal classifier are particularly challenging and can saturate the achievable accuracy in many cases.

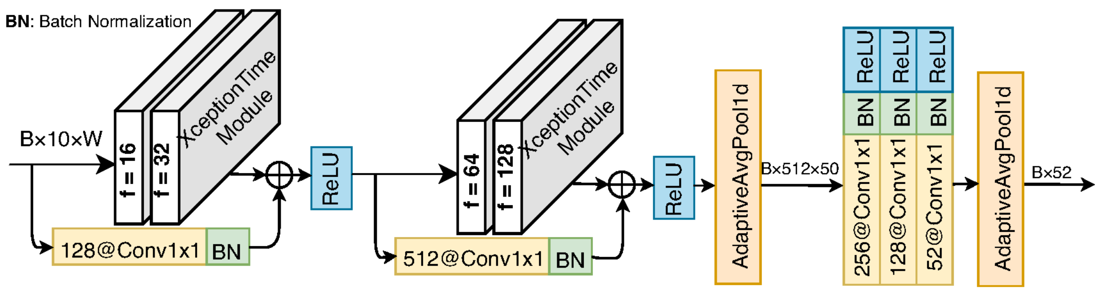
In [54], inspired by the Inception V4 architecture, a new deepnet model has been recently proposed and named as "InceptionTime" for time series classification. In [54] it is shown that InceptionTime, which is an equivalent of AlexNet for time series data, is more accurate and faster than its existing counterparts in time series classification. On the other hand, in [55], by replacing the Inception modules with depthwise separable convolution, a new architecture is designed and named as Xception, which has better performance than Inception V3 on a large image classification dataset. Motivated by the prior works [54, 55], we propose a novel deep architecture, the XceptionTime, which is more accurate than the existing model for sparse sEMG-based hand gesture recognition. Furthermore, by deploying adaptive average pooling, the proposed end-to-end XceptionTime architecture is independent of the time window, meaning that for utilization of different time windows, e.g., 50 ms, 100 ms, or 150 ms, there is no need to reconfigure and retrain the XceptionTime model. Besides, by replacing the fully connected layers with adaptive average pooling, the proposed XceptionTime model is less prone to overfitting because there are no extra parameters to optimize [56]. By deploying adaptive average pooling, the proposed architecture is more robust to temporal translation of the inputs as the temporal information will sum out.

In the following, first, the proposed XceptionTime module is introduced followed



channels
1
time
W: Window Length
B: Batch Size
Cin: Number of Input Channels
Cout: Number of Output Channels=4×f
Bottleneck
B×Cin×W
f@Conv1x1
B×f×W
f@SeparableConv(1,11,5)
f@SeparableConv(1,21,10)
f@SeparableConv(1,41,20)
MaxPool1D(s=1,k=3,p=1)
f@Conv1x1
Channel Concat
B×Cout×W
f@SeparableConv(s, k, p): Stands for number of Depthwise Separable Convolution filters with stride s, kernel_size k, and padding p
f@Conv1x1: Stands for number of Convolution filters with stride 1, kernel_size 1, and padding 0

(a) XceptionTime Module



(b) XceptionTime Architecture

Figure 3.8: (a) ***XceptionTime Module***, which consists of two parallel paths, the first path includes three Depthwise Separable Convolutions, while the second path includes a MaxPooling followed by a Conv1×1. (b) ***XceptionTime Architecture***, which includes series of XceptionTime modules with residual connections followed by Adaptive Average Pooling layers and Conv1×1 layers.

by a description of the XceptionTime architecture consisting of stacked XceptionTime modules and adaptive average pooling layers.

XceptionTime Module: One of the challenging tasks in designing CNNs, is selecting the right kernel size, which has an important role in extracting global or local information. However, inspired by Inception [57], as shown in Fig. 3.8(a), instead of committing ourselves to pick a filter with a specific size, we adopt multiple one-dimensional filters with different kernel sizes to extract short and long time series' features simultaneously with the resulted feature maps being concatenated to construct the output features. Moreover, for mitigating the computational cost problems, as well as lessening the overfitting problems, the *bottleneck* layer is used as the first component within the proposed XceptionTime Module. In the bottleneck layer, f number of one-dimensional filters with kernel size one is utilized to transform the input with C_{in} channels into another time series with f channels.

One key difference between the proposed XceptionTime module and InceptionTime module previously proposed in [54], is deploying Depthwise Separable Convolutions, which significantly mitigates the required number of parameters in the network. In Depthwise Separable Convolution [58, 59], two convolutions are deployed, i.e., the *Depthwise Convolution*, and the *Pointwise Convolution*. In Depthwise Convolution, each channel of the input is convolved separately and then stacked together; therefore, the temporal convolution is done without changing the depth. The consequent output from the Depthwise convolution is fed to the Pointwise convolution, where 1×1 convolutions are utilized to transform the number input channels from the Depthwise convolution into a new channel depth. Later in Section 3.2.3, it will be shown that by using the depthwise separable convolutions, not only the recognition accuracy will be increased, but also the number of parameters will be reduced significantly.

To summarize, as shown in Fig. 3.8(a), the time series input with C_{in} number of channels is first fed to two parallel paths. The first path consists of a bottleneck, reducing the dimensionality of the input, followed by three sets of depthwise separable convolution each with f number of filters with kernel size l, where l is set to 11, 21, or 41. In the second path, the input is fed to a MaxPooling layer followed by a Conv1×1 component, which produces an output with f channels. Finally, the resulted feature maps of Depthwise Separable Convolutions and skip connections are concatenated in a channel-wise fashion. As shown in Fig. 3.8(a), the time series input with C_{in} channels are transformed to output with C_{out} number of channels, where C_{out} is four times that of the number of filters (f) used in the bottleneck as well as in the depthwise separable convolutions.

XceptionTime Architecture: The XceptionTime architecture is constructed based on the proposed XceptionTime modules (Fig. 3.8(a)). More specifically, after preprocessing, sEMG signals acquired from 10 sensors are segmented by a window with a length of $W \in \{50\,\text{ms}, 100\,\text{ms}, 150\,\text{ms}, 200\,\text{ms}\}$. The sliding window with steps of 10 ms is considered for segmentation of the sEMG signals. The proposed XceptionTime architecture (Fig. 3.8(b)), includes 4 XceptionTime modules where the number of filters (f) are set to 16, 32, 64, and 128, respectively. Moreover, two residual connections are deployed in the XceptionTime architecture to address the degradation problem. Each residual connection consists of a Conv1 $\times$ 1 layer, to match up the input and output dimensions, followed by Batch Normalization, which is for regularization and also to reduce the internal covariate shift effect. Moreover, in order to learn the complex structure of data, ReLU is applied to the summation of outputs of residual connection and the XceptionTime module.

As stated previously, one of the novelties of the proposed XceptionTime architecture is the independency of the architecture from the length of the time window. In other words, for an arbitrary window length, the XceptionTime architecture remains unchanged without any need of reconfiguration. To realize this objective, the output yielded from summation of the 4^{th} XceptionTime Module and residual Connection is fed to an Adaptive Average Pooling layer, which transforms the input with window size W to a fixed length of 50. Then, the dimension of the time series input will be reduced to the number of the classes (i.e., 52 in our settings) by stacking three Conv1×1, each followed by Batch Normalization and ReLU. Finally, a second Adaptive Average Pooling layer is used to convert the length of the input signal to one. We use Adam optimizer for training purpose with learning rate of 0.001. The learning rate changes in a cycle with a length of 20 epochs. After 20 epoch, we divided the learning rate by 2. These models are trained with a mini-batch size of 32. For measuring the classification performance the Cross-entropy loss is considered.

3.2.3 Experiments and results

In this section, the performance of the proposed XceptionTime architecture is evaluated through a comprehensive set of different experiments and provide comparisons with 6 state-of-the-art models [24,29–32,53] developed recently on the same dataset to illustrates superior performance of the proposed XceptionTime over its counterparts.

Experiment 1: In this experiment, referred to as "First Exp." in the results, the objective is to validate our claim that by incorporation of Depthwise Separable Convolutions within the proposed XceptionTime architecture, a much smaller model size with significantly reduced complexity will be achieved. For this purpose, we implemented a variant of the proposed architecture, where referred to as XceptionTime-V2, where standard convolutions are deployed within the XceptionTime Module instead of Depthwise Separable Convolutions. Table 3.2 shows the results, where it can be

Table 3.2: (First Exp.): Comparison between the proposed XceptionTime model and XceptionTime-V2 model. (Second Exp.): Result of the proposed model and XceptionTime-V2 when the input is normalized by Minmax.

Exp.	**Normalization**	**Model**	**Accuracy (%)**					**Model Parameters**
			50 ms	**100 ms**	**150 ms**	**200 ms**	**Trial**	
First	μ**-law**	XceptionTime	81.71	87.40	90.76	92.30	95.43	413,516
		XceptionTime-V2	81.24	86.81	89.79	91.71	94.59	1,918,476
Second	**Minmax**	XceptionTime	71.49	82.63	88.94	90.51	92.20	413,516
		XceptionTime-V2	68.95	79.56	87.17	89.61	90.08	1,918,476

Table 3.3: Accuracy when the proposed XceptionTime Model is trained on a combination of different window lengths (i.e. 50, 100, 150, 200) and then tested on different windows.

Exp.	**Model**	**Accuracy (%)**				
		50 ms	**100 ms**	**150 ms**	**200 ms**	**Trial**
Third	XceptionTime	77.87	87.64	91.81	93.91	95.44

observed that while the accuracy associated with the XceptionTime is slightly better than XceptionTime-V2, the number of parameters is significantly reduced. For example, the accuracy for the proposed XceptionTime model for window length 200 ms is 95.43% using $413,516$ number of parameters, while XceptionTime-V2 achieves accuracy of 94.59% but using extensively higher $1,918,476$ of parameters.

Experiment 2: In this experiment, referred to as "Second Exp." in the results, the objective is to validate the effectiveness of using the proposed non-linear μ-normalization within the proposed XceptionTime architecture. In this regard, in Table 3.2, results trained by using Minmax normalization is shown for both variants of the proposed framework. From Table 3.2, it is observed that the accuracy of the model will decrease when Minmax normalization is applied to the input. For instance, accuracy of the proposed XceptionTime framework with μ-law normalization in window length of 50 ms is 81.71%, whereas using Minmax normalization within the proposed XceptionTime framework reduces the accuracy to 71.49%. Another observation is that the degradation effect of discarding the proposed nonlinear normalization approach on XceptionTime-V2 is higher.

Experiment 3: The third experiment is performed to validate our claim that the

Table 3.4: Comparison of the proposed XceptionTimel with the state-of-the-art literature (number of parameters for [24, 29, 31] are reported from [53]).

	Accuracy (%)					**Model Parameters**
	50 ms	**100 ms**	**150 ms**	**200 ms**	**Trial**	
XceptionTime (First Exp.)	81.71	87.4	90.76	92.3	95.43	413,516
XceptionTime (Third Exp.)	77.87	87.64	91.81	93.91	95.44	413,516
GengNet [24]	-	-	-	77.8	76.1	500,000
WeiNet [31]	81.7	83.4	84.4	85	-	2.5M
HuNet [30]	-	-	86.8	87	97.3	-
AtzoriNet [29]	-	-	66.59	-	-	85,000
TsinganosNet [53]	-	-	-	-	89.76	85,000
WeiNet [32]	85.8	86.8	87.4	88.2	-	-

proposed XceptionTime is applicable to different window sizes without the need for re-configuration. We evaluate the performance when the proposed architecture is trained based on a combination of different window sizes. In other words, instead of training the XceptionTime model just with a specific time window (as is done for reporting the results in Table 3.2), inputs with different window sizes are fed into network to increase the robustness of the network during training. However, for the effectiveness of the training process, only windows with the length of 50, 100, 150, and 200 are used as input. Table 3.3 illustrates the results obtained from XceptionTime trained with a combination of different window lengths and then tested separately on each window size. As can be seen, the performance of the model, except for time window 50, is improved in comparison to the case where the model was just trained with a specific window length (Table 3.2(First Exp.)). In other words, not only the proposed model can handle different window sizes simultaneously, by utilizing this property the performance can be boosted. Finally, Table 3.4 shows performance of the proposed model in comparison to the state-of-the art results obtained over the same DB1 dataset of 52 hand gestures. As shown in Table 3.4, our architecture outperforms existing solutions while maintaining a reduced number of parameters.

3.3 Conclusion

In this chapter, we targeted the problem of hand gesture classification using deep learning-based architectures by focusing on improving the accuracy in Section 3.1 and reducing the overall complexity (number of trainable parameters) in Section 3.2. For the former category, we have proposed two architectures, referred to as the HRM and the TCN. Generally speaking, although the performance of deep learning algorithms can motivate their use for multichannel electrode space, applying/training deep models based on signals obtained from sparse multichannel sEMG devices is very challenging as such datasets are typically shallow. The HRM architecture is proposed to address this gap by designing a hybrid architecture. Moreover, we have proposed the TCN architecture, which is based on dilated causal convolutions to increase the receptive field of the network and process the input sequence as a whole, instead of sequentially as in HRM. For the problem of reducing the computational burden, we proposed the XceptionTime architecture, which is designed by integration of depthwise separable convolutions, adaptive average pooling, and a novel non-linear normalization technique. To the best of our knowledge, it is the first time that the proposed innovative XceptionTime architecture is introduced and has not been designed/utilized previously in sEMG-based HGR recognition. Its performance is evaluated via the benchmark sparse sEMG dataset outperforming its state-of-the-art counterparts.

Chapter 4

Attention-Based Models for HGR

In Chapter 3, we focused on improving accuracy and reducing complexity of deep neural networks for sEMG-based hand gesture recognition. In this chapter, we focus on attention-based models to further improve the performance for the task at hand. The reminder of the chapter is organized as follows: The database and pre-processing step are described in Section 4.1. Afterwards, Section 4.2 describes the proposed TC-HGR architecture, which is designed based on the attention mechanism and temporal convolution networks. Finally, the proposed TEMGNet framework, which is based on the transformer architecture is presented in Section 4.3.

4.1 Material and Methods

Before presenting the proposed TC-HGR and TEMGNet architectures, below, we discuss the utilized dataset for training purposes followed by the pre-processing step used to prepare training, validation, and test sets.

Database: To train and evaluate the proposed TC-HGR and TEMGNet architectures, we used the second Ninapro dataset [16] called DB2, which is a widely used public dataset. While the dataset has been previously described in Subsection 3.1.1, we briefly present it here for completeness and ease of reference. As stated in Subsection 3.1.1, DB2 is collected by the Delsys Trigno Wireless EMG system, which has 12 channels and records the electrical activities of muscles at 2 kHz. Moreover, the DB2 dataset consists of the sEMG signals from 40 healthy subjects performing 50 different hand gestures. Each gesture is repeated 6 times, each lasting for 5 seconds followed by 3 seconds of rest. The 50 different gestures in the DB2 dataset are presented in three sets of exercises (i.e., B, C, and D). In this chapter, we focus on Exercise B, which consists of 17 different gestures. More specifically, Exercise B consists of 9 basic wrist movements with 8 isometric and isotonic hand movements. Following the recommendations provided by the Ninapro dataset, the training set consists of 2/3 of the repetitions of each gesture (i.e., $1, 3, 4$, and 6), and the test set consists of the remaining repetitions (i.e., 2 and 5).

Pre-processing Step: For developing TC-HGR architecture, we used the raw time-domain sEMG signals. The sEMG signals are pre-processed and smoothed using a 1^{st} order 1 Hz low-pass Butterworth filter. Moreover, to amplify the magnitude of sensors with small values, we scaled the sEMG signals logarithmically as stated previously in Subsection 3.2.1. More specifically, we utilized the *μ-law* transformation, which is primarily used for quantization in the speech processing domain and re-purposed in our works for normalizing the sEMG signals. This completes the steps performed to pre-process the sEMG signals and prepare the input to be provided to the TC-HGR and TEMGNet architectures. Next, we present the detailed structure of the former architecture.

4.2 Temporal Convolutions-based Hand Gesture Recognition (TC-HGR)

Developing a DNN architecture with fewer parameters can tackle the challenge of structural complexity and reduce the gap between academic research and practical settings for myoelectric prosthesis control. DNN models have shown promising results with respect to other algorithms for decoding muscle electrical activity, especially for recognition of hand gestures. Such data-driven models, however, have been challenged by their need for a large number of trainable parameters and their structural complexity. In the previous chapter (Chapter 3), we have targeted this issue by incorporating different advanced deep models such as hybrid CNN-LSTM architecture, dilated causal convolutions, and depthwise separable convolutions. In this section, instead, we focus on the use of attention mechanism and propose the novel Temporal Convolutions-based Hand Gesture Recognition (TC-HGR) architecture to reduce this computational burden.

4.2.1 The TC-HGR Architecture

In this section, first, we provide an introduction, followed by presentation of the proposed TC-HGR framework. Finally, we discuss detailed architecture of the TC-HGR. As stated previously, HGR via sEMG signals [9, 53, 60, 61] has been investigated in the literature as the most promising approach for myoelectric control of prosthetic systems. In particular, HGR has been the focus of different research works [2,8], given its unique potentials to improve the quality of control and consequently to enhance the quality of life of amputees. Although academic researchers have used advanced ML and DNN models to achieve promising laboratory results in HGR, translating these new techniques into the daily lives of amputees has faced several critical challenges [13,62]. One of the key challenges is the dependency of DNN models on a large number of trainable parameters, which leads to structural complexity and limits their applicability to clinical settings [63]. Therefore, there is an urgent and unmet quest to develop DNN-based learning frameworks that focus on reducing the number of parameters and maintaining high performance.

Following our discussion in the previous chapters, performance of DNN models developed based on sparse multi-channel sEMG is still significantly lower than that of HD-sEMG systems, which has a high number of densely located electrodes to significantly increase the information rate [24, 30, 31]. For instance, in Reference [31], the HGR accuracy of 99.7% is reported using HD-sEMG, which is reduced to 84.4% when sparse multi-channel sEMG signals are used. In this context, we aim to design a novel DNN architecture using sparse multi-channel sEMG signals provided by the Ninapro [16, 17] database, which is one of the most widely accepted sparse multi-channel sEMG benchmark datasets. We designed the novel TC-HGR to reduce computational burden while maintaining high accuracy, which is of paramount importance to translate the classification results into smooth actions.

A common strategy for classifying hand movements with DNN-based algorithms is converting sEMG signals into images and then using CNNs to detect hand movements [24, 31, 32, 48]. sEMG signals are, however, sequential in nature, and CNNs cannot extract temporal features. In this regard, recent literature [30, 63] used recurrent architectures such as LSTM networks to consider the sequential nature of sEMG signals. In addition, LSTMs and CNNs can be combined as a hybrid architecture [2] to jointly extract the temporal and spatial properties of sEMG signals. Although sequence modeling with recurrent networks is a common approach, it can have disadvantages, such as lack of parallelism during training, exploding/vanishing gradient, and extensive memory/computation requirements [43]. Therefore, Reference [43] proposed TCs for extracting temporal information in time series tasks while addressing the aforementioned challenges of the recurrent architectures. In addition, Reference [63] proposed the concept of temporal dilation of LSTM to reduce the computational cost, memory, requirement, and model complexity. On the other hand, attention-based architectures such as Transformers [64,65] show great potentials for widespread adoption in different Artificial Intelligence (AI) applications. In particular, the transformer-based architectures could improve the recognition accuracy compared to their state-of-the-art counterparts (where LSTM or hybrid LSTM-CNN are adopted). However, transformers are limited by the memory and computation requirements of the quadratic operation in attention for long sequences or images. Therefore, in [64], the authors proposed dividing the images into patches and then using the flattened patches as the input for the transformers. Inspired by the progress of attention mechanism and TCs [66], we aim to design a DNN-based architecture based on the combination of attention and temporal convolutions.

TC-HGR architecture: After pre-processing, we segment the sEMG signals based on a window of size $W \in \{200\,ms, 300\,ms\}$, resulting in the dataset $\mathcal{D} = \{(\boldsymbol{X}_i, \mathrm{y}_i)\}_{i=1}^{M}$. More specifically, $\boldsymbol{X}_i \in \mathbb{R}^{C \times L}$ is the i^{th} segment with label y_i, for $(1 \leq i \leq M)$. Here, C indicates the number of channels in the input segment, and L is the length of the segmented sequence, which represents the number of samples obtained at a frequency of 2 kHz for a window of size W. As illustrated in Fig. 4.1, the TC-HGR framework has been developed based on "Embedded Patches" and two modules, namely *Temporal Convolution Block* and *Self-Attention Module with Residual Connection*, which is described below:

(i) **Embedded Patches:** In a similar way to the ViT architecture [64], the input segment $\boldsymbol{X}_i$ is divided into N non-overlapping patches. Here, $N = L/P$, where P shows the size of each patch. This patching mechanism helps reduce memory and computation requirements. As shown in Fig. 4.1, the sequence of linear projections of these patches are fed as input to the TC-HGR architecture. More

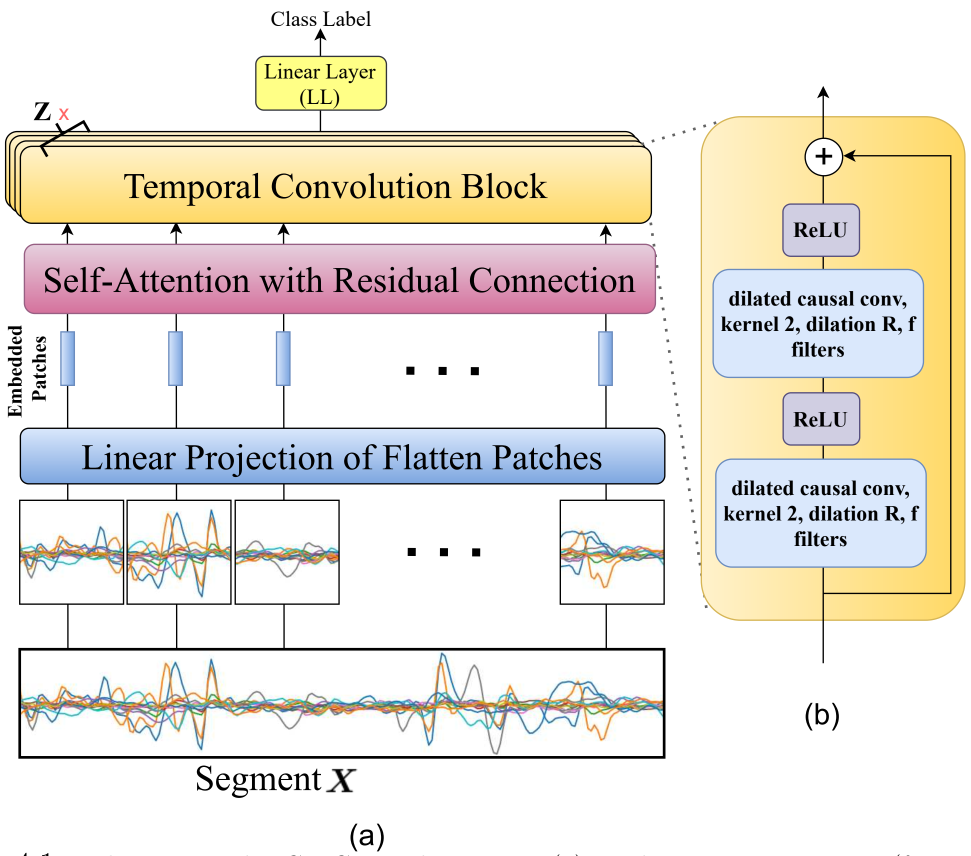

Class Label
Linear Layer (LL)
Z x
Temporal Convolution Block
Self-Attention with Residual Connection
Embedded Patches
Linear Projection of Flatten Patches
Segment $\boldsymbol{X}$
(a)
ReLU
dilated causal conv, kernel 2, dilation R, f filters
ReLU
dilated causal conv, kernel 2, dilation R, f filters
(b)

Figure 4.1: The proposed TC-HGR architecture: (a) Each input segment $\boldsymbol{X}$ (for simplicity, we dropped the index i) is divided into N non-overlapping patches. Then, each patch is flattened and mapped to model dimension D (blue block). We refer to the output of this process as Embedded Patches. The sequence of the Embedded Patches is passed into the Self-Attention module, which includes the residual connection (purple block). Afterward, we used Z number of Temporal Convolution Blocks to access a long history (orange block). (b) Each Temporal Convolution Block consists of two dilated causal convolutions, each followed by a ReLU activation function. Again, we used residual connections to concatenate the output and input. Finally, a Linear Layer (LL) is adopted to output the class label.

specifically, each patch is first flattened and then is mapped into the model dimension D with a trainable linear projection. The output of this projection is called the “Embedded Patches”.

(ii) **Temporal Convolution Block:** In recent literature (such as [43]), the authors represented TCs for the sequence modeling tasks and showed that temporal convolutions could outperform recurrent networks such as LSTMs in a wide range of datasets and time-series tasks. More specifically, TCs offer several advantages over recurrent networks such as processing the input sequence as a whole rather than sequential training, low memory requirements, stable gradient, and capturing the past information with flexible receptive field size. Inspired by the performance of TCs for the sequential data, we used "Temporal Convolution Block" (represented via an orange box in Fig. 4.1) instead of recurrent networks for the sEMG-based gesture recognition. As shown in Fig. 4.1, each Temporal Convolution Block consists of dilated causal convolutions, where the dilation rate R increases exponentially, i.e., $(1, 2, 4, 8, \ldots)$, to access a receptive field that exceeds the length of the input sequence. Moreover, each dilated causal convolution is followed by a ReLU activation function. The number of Temporal Convolution Blocks are based on the logarithmic scale with the number of patches N. More specifically, we used $Z = \lceil \log_2 N \rceil$ number of Temporal Convolution Block for an input segment $\boldsymbol{X}$ with a sequence length of L.

(iii) **Self-Attention Module with Residual Connection:** In the proposed TC-HGR architecture, we used the "Temporal Convolution Block" along with the "Attention" mechanism. In [65], the authors showed that the attention mechanism allows a model to present important information in a given input sequence. Moreover, the attention mechanism has recently been used [6] in the context of sEMG-based hand gesture recognition, where experiments have demonstrated the ability of attention to identifying specific pieces of information in the sequential nature of the sEMG signals. On the other hand, in References [1,66], it was shown that temporal convolutions and attention are complementary mechanisms, i.e., the former captures a long history while the latter identifies a specific type of information. An attention block measures the pairwise similarity of each query and all keys to assign a weight to each value. Then, the output is computed, which is the weighted sum of the values [65]. The keys, values, and queries are packed together into matrices $\boldsymbol{K}$, $\boldsymbol{V}$, and $\boldsymbol{Q}$, respectively. The output matrix is then computed as follows:

$$\text{Attention}(\boldsymbol{Q}, \boldsymbol{K}, \boldsymbol{V}) = \text{softmax}(\frac{\boldsymbol{Q}\boldsymbol{K}^T}{\sqrt{d_k}})\boldsymbol{V}, \tag{4}$$

where d_k denotes the dimension of $\boldsymbol{K}$ and $\boldsymbol{Q}$. In the TC-HGR architecture, we also used residual connections to concatenate the output and input.

This completes the description of the proposed TC-HGR architecture, next, we present our results to evaluate its HGR performance.

Table 4.1: Descriptions of TC-HGR architecture variants.

Window size W	**Model ID**	**Number of Patches N**	**Model dimension D**	**Params**
200 ms	**1**	10	12	49,186
	2	10	16	68,445
	3	16	12	69,076
	4	16	16	94,965
300 ms	**1**	10	12	52,066
	2	10	16	72,285
	3	15	12	67,651
	4	15	16	92,945

4.2.2 Experiments and Results

In this section, the performance of the proposed TC-HGR architecture is evaluated through a comprehensive set of experiments as presented below.

Results and Discussions: In Table 4.1, different variants of the TC-HGR architecture are presented based on a window size of $W \in \{200\,ms, 300\,ms\}$. For training, Adam optimizer was used across all the models with a learning rate of 0.0001. Furthermore, we used a mini-batch size of 32. The performance of each model is evaluated using Cross-entropy loss. In Table 4.2, the averaged recognition accuracy of the proposed TC-HGR architecture and its variants are reported over all subjects. In what follows, we focus on three different experiments:

(i) **Experiment 1 - Effect of the Model's Dimension D:** In this experiment, the objective is to investigate the effect of D of the proposed TC-HGR architecture on the recognition accuracy. In this regard, Table 4.2 has shown the results for $D \in \{12, 16\}$ for both window sizes. From Table 4.1 and Table 4.2, it is observed that the accuracy of the model will improve when the D is increased from 12 to 16 for the same Number of Patches N. More specifically, "Model 2 versus Model 1" and "Model 4 versus Model 3" are more accurate in both the 200 ms and 300 ms window sizes. However, from Table 4.1, it can be observed that the number of trainable parameters has increased when D is increased

Table 4.2: Classification accuracies for TC-HGR architectures variants for different Window Size (W).

	Model ID	**1**	**2**	**3**	**4**
$W =$ 150 ms	Accuracy (%)	79.78	80.22	80.00	**80.72**
	STD (%)	6.5	6.5	6.5	**6.4**
$W =$ 200 ms	Accuracy (%)	80.29	80.63	80.51	**80.83**
	STD (%)	6.7	6.8	6.7	**6.6**
$W =$ 250 ms	Accuracy (%)	80.34	80.99	80.73	**81.35**
	STD (%)	6.4	6.5	6.6	**6.6**
$W =$ 300 ms	Accuracy (%)	80.84	81.59	80.95	**81.65**
	STD (%)	6.4	6.5	6.5	**6.7**

from 12 to 16, which leads to more complexity. For instance, for $W = 200$ ms and $N = 10$, Model 2 has $68,445$ parameters, while this number is $49,186$ for Model 1 (Table 4.1). While increasing D can potentially improve performance, the implementation of prosthetic controllers is limited by its structural complexity.

(ii) **Experiment 2 - Effect of the Number of Patches N:** This experiment is included to evaluate the effect of increasing N on the performance of the proposed TC-HGR. From Tables 4.1 and 4.2, it is observed that for the same W and D, accuracy increases as the number of patches N increases from 10 to 16. More specifically, "Model 3 versus Model 1" and "Model 4 versus Model 2" classified the hand gestures with higher accuracies in both window sizes. This is because use of more patches results in a larger effective sequence length, which in turn improves the overall performance. Increasing the number of patches, however, makes the structure more complex. For instance, for $W = 300$ ms and $D = 12$, Model 3 has $67,651$ parameters, while Model 1 has $52,066$ parameters (Table 4.1).

(iii) **Experiment 3 - Effect of Window size W:** As shown in Table 4.2, increasing the window size leads to more accuracy. This is because the larger the window size, the more information is provided for the proposed TC-HGR architecture. In other words, machine learning model would have higher exposure to the signals from the gesture. For instance, for both "Models 2 and 1", the window size 300 ms leads to greater accuracy and complexity. However, both "Models 3 and 4" classified hand gestures more accurately while reducing the complexity because number of patches for $W = 300$ ms is less than $W = 200$ ms. We would like to point out the window length is required to be under 300 ms to satisfy the acceptable delay time [40]. With a larger window, the results would likely improve. However, the use of shorter windows (e.g., 200 ms) provides an extra time (100 ms) to perform the pre-processing and classification tasks, which allows staying within the target 300 ms. We have also trained and evaluated two other window size; i.e. 150 ms and 250 ms for our proposed architecture.

Statistical Comparisons of Different TC-HGR Variants for Window Size of 300 ms: To examine the importance and significance of TC-HGR variants, we perform statistical tests for all models considering $W = 300$ ms. By following [1], the Wilcoxon signed-rank test [67] is used in which each user is considered as a separate dataset. As illustrated in Fig. 4.2, we conduct comparison between the model with the least number of parameters (i.e., Model 1) and other models. (i.e., Model 2, 3, and 4). The performance distribution between 40 users for each model is illustrated in Fig. 4.2. We adjusted the resulted p-values using Benjamini-Hochberg false discovery rate. The results are given in Table 4.3 for window size 300 ms.

Comparison with the State-of-the-art Research [63]: We have also compared the results with a recent state-of-the-art model [63] in which the same dataset is used for performance evaluations. More specifically, Reference [63] proposed models based on recurrent architectures (i.e., LSTM) with dilation. As shown in Table 4.4, for window size of 200 ms, our methodology can outperform both recurrent networks and traditional ML approaches such as SVM. For instance, it can be observed that the accuracy for our proposed Model 4 is 80.72% with only $94,965$ number of parameters, while the best accuracy for the Reference [63] is 79% with $1,102,801$ parameters. Moreover, as shown in Table 4.4, for window size of 300 ms, the accuracy of the proposed Model 4 is 81.65%, while for "pure LSTM with dilation" and SVM proposed in [63], the accuracy is 79.7% and 30.7%, respectively. Although in [63] the authors reached to 82.4% accuracy with "4-layer 3rd Order Dilation" which is a hybrid dilation-based LSTM, they used $1,102,801$ number of parameters which is 11.9 times larger

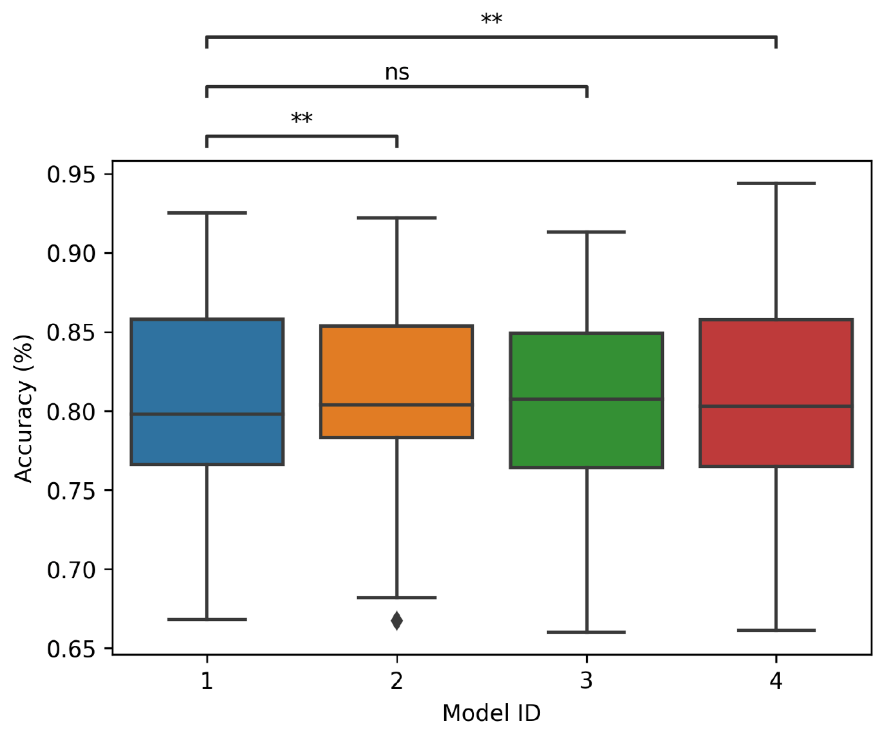


Figure 4.2: The accuracy boxplots for all TC-HGR architecture variants for window size 300 ms. Each boxplot is representing the IQR of each model across 40 users. (ns: $0.05 < \text{p-value} \leq 1$, $^{*}: 0.01 < \text{p-value} \leq 0.05$, $^{**}: 0.001 < \text{p-value} \leq 0.01$, $^{***}: 0.0001 < \text{p-value} \leq 0.001$, $^{****}: \text{p-value} \leq 0.0001$).

Table 4.3: Adjusted Wilcoxon signed rank test p-values by Benjamini-Hochberg method for window size 300 ms.

	Model 1	Model 2	Model 3
Model 2	0.007		
Model 3	0.572	0.007	
Model 4	0.017	0.572	0.017

than the number of parameters used in our proposed Model 4. Therefore, we provided a compact DNN model with a far fewer number of trainable parameters compared to previous works, and took a step forward towards designing more proportional, intuitive, and dexterous prostheses for clinical applications.

We have also obtained the average of classification results per each gesture for our proposed Model 4. As shown in Table 4.5, each gesture is presented with an index (e.g., 1-17) exactly matched with the Ninapro dataset. We followed References [16,17] and presented each gesture with an index. The full description of these movements is

Table 4.4: Comparison between the proposed TC-HGR methodology and previous works [63], which used recurrent architectures (LSTM).

		200 ms		**300 ms**	
		Params	**Accuracy (%)**	**Params**	**Accuracy (%)**
Reference [63]	4-layer 3rd Order Dilation	1,102,801	79.0	1,102,801	82.4
	4-layer 3rd Order Dilation (pure LSTM)	_	_	466,944	79.7
	SVM	_	26.9	_	30.7
Our Method	Model 1	**49,186**	80.29	**52,066**	80.84
	Model 4	94,965	**80.72**	92,945	**81.65**

Table 4.5: Gestures' classification results for Model 4. The description of the gestures are presented in the References [16, 17].

Gesture Index	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
Accuracy (%)	89.39	84.63	84.20	82.51	74.80	84.39	82.32	77.72	83.19	77.16	72.25	76.55	84.25	83.56	87.76	77.08	86.22

given in References [16, 17]. As shown in Table 4.5, the accuracy of model degrades when the gestures are similar. For example, the model distinguish index 11 and 12 which corresponds to the wrist supination and pronation with a lower accuracy. This result is not far from expectation as the muscle groups which are predominantly used during these two gestures are exactly the same [68]. Finally, we evaluated effectiveness and robustness of the proposed method through Monte Carlo (MC) simulations [69]. For evaluating the convergence of proposed approach, in particular 100 MC runs are performed where at each run sensor measurements are contaminated by additive Gaussian noise based on a specific level of Signal to Noise Ratio (SNR). MC simulation results (100 times and SNR = 25 dB) for the proposed Model 4 is 81.19% ± 6.6%, while without MC simulation the accuracy for the same model is 81.65% ± 6.7%. The achieved accuracy show remarkably stable performance of the proposed model. The MC simulations motivate further research to improve the overall model stability to deal with higher noise levels (low SNR values).

This completes developments and discussions on TC-HGR architecture for the task of improving HGR accuracy based on sEMG signals. Next, we shift the focus to designing a transformer-based architecture for the task at hand.

4.3 Transformer-based EMG Architecture (TEMGNet)

As stated previously, DNN-based models require large training sets and, typically, have high structural complexity, i.e., they depend on a large number of trainable parameters. To address these issues, in this part, we develop a framework based on the Transformer architecture for processing sEMG signals.

4.3.1 The TEMGNet Architecture

The TEMGNet framework is developed for processing sEMG signals by capitalizing on recent advancements of the Transformer architecture. Transformers are considered to be powerful DNN models that have revolutionized Natural Language Processing

(NLP) and Computer Vision (CV). However, transformer models have remained unexplored for the task of myoelectric control of bionic limbs. In this regard, we propose a ViT-based neural network architecture (referred to as the TEMGNet) to classify and recognize upper-limb hand gestures from sEMG to be used for myocontrol of prostheses. To evaluate its efficacy, the second subset (exercise B) of the NinaPro DB2 dataset was utilized, where the proposed TEMGNet framework achieved a recognition accuracy of 82.93% and 82.05% for window sizes of 300 ms and 200 ms, respectively, outperforming its state-of-the-art counterparts. Moreover, the proposed framework is superior in terms of structural capacity while having seven times fewer number of trainable parameters.

Prior Research: As discussed in the previous chapters, recognizing limb motions using sEMG signals allows for the control of rehabilitation and assistive systems (such as bionic limbs and exoskeletons). Surface EMG signals are obtained non-invasively by sensors on the skin surface that measure the electrical activity of the muscle's motor units [60, 61]. The information obtained from sEMG signals is used to decode discriminative and repeatable patterns that can be utilized to effectively classify the intended motor commands of the user. In this context, several attempts have been made to classify hand movements using traditional ML methods. While these conventional approaches (such as LDA and SVM [29]) have been successfully implemented, their performance might degrade when applied to large-scale data set consisting of a sizable number of movements. This has motivated a recent surge of interest in applying DNNs within this domain, with the aim of addressing the shortcomings of traditional ML solutions.

Processing sEMG signals using DNN architectures has the potentials to provide significantly improved performance. CNN is the commonly used DNN architecture for recognizing upper-limb hand gestures in which sEMG signals are translated into images [9,24,31]. CNN-based models, however, target learning spatial features and are ineffective for the extraction of temporal features from sEMG sequential data. RNNs, such as LSTM, have been therefore proposed in recent studies [63, 70] to capture the temporal information from sEMG signals. To capture both spatial and temporal characteristics of sEMG signals, LSTM and CNN can be combined [2, 7], resulting in a hybrid solution such as the one proposed in Chapter 3 (Section 3.1.2). Another example is Reference [30], where authors composed six image representations of raw sEMG signals that were then fed as input to a hybrid CNN-LSTM architecture. We showed in Chapter 3 (Section 3.1.3) that dilated causal convolutions in CNN-based architectures have great potential to surpass the pure RNN architecture in term of overall accuracy [8, 43]. In addition, dilated LSTM models have shown the potential to enhance accuracy and reduce computational cost [63].

While the above-mentioned models are advanced approaches to sequence modeling, they do not allow parallelization during the training phase due to their sequential nature. For this reason, recurrent-based models are slow to train. The transformer neural network architecture has been proposed to eliminate recurrence or convolution using a self-attention mechanism [65]. In brief, the main building block of the transformer architecture, which makes it a unique and powerful DNN model, is the attention mechanism. Attention is a familiar word to use based on which we selectively focus on one piece of information and ignore others. A neural network is seen as an attempt to mimic the computational power of the human brain. The attention mechanism is a way to mimic and execute the function of selective focus. Transformers were first applied to NLP tasks, with the goal of solving sequence-by-sequence tasks while handling long-range dependencies [65]. The transformer model architectures, such as Bidirectional Bidirectional Encoder Representations from Transformers (BERT) [71] and Generative Pre-Training (GPT) [72], achieved state-of-the-art results in many NLP tasks. In addition to the NLP field, Transformers have been applied to address a variety of other problems, including CV tasks [64]; Electroencephalogram (EEG)-based speech recognition [73]; EEG decoding [74]; Electrocardiogram (ECG)-based heartbeat classification [75]; sleep stages classification [76], acoustic modeling [77], and ECG classification [78].

Contributions: In the above context, we hypothesize that novel models, designed on the basis of transformers, would enhance the accuracy of EMG classification. This will be a major step towards the ultimate utilization of deep learning models for commercial prosthetic systems. Therefore, here we propose, design, and evaluate the performance of a novel transformer model for hand gesture recognition. The proposed TEMGNet framework was designed based on the ViT architecture to improve recognition accuracy and to reduce structural complexity. Here:

- We illustrate that transformers can be trained for sEMG dataset without the need to use pre-trained models and fine-tuning.

- We show that the proposed TEMGNet architecture outperforms its state-of-the-art counterparts in terms of overall recognition accuracy and complexity.

In summary, the main contributions of this work are as follows:

- For the first time, we propose a novel deep learning architecture (the TEMGNet) for hand gesture recognition via the utilization of transformers for processing sEMG signals.
- The proposed transformer-based architecture improves the recognition accuracy compared to its state-of-the-art counterparts (where LSTM or CNN are adopted) with the significantly reduced number of trainable parameters. This can be considered a significant step forward towards feasibly embedding DNN-based models into prostheses controllers.

We examined the effect of several variants of the TEMGNet architectures on both recognition accuracy and the number of required trainable parameters. We also conducted statistical tests to assess the significance of the proposed architecture. Finally, we present a visualization of position embedding similarities, showing that the model is capable of encoding position information and considering the sequential nature of sEMG signals.

The TEMGNet Framework: The main fundamental block of the Transformer architecture is the attention mechanism. It is noteworthy to mention that attention along with other architectures such as CNN and/or LSTM has recently been utilized [1,4–6,30] to classify hand movements based on sEMG signals, where the results showed the ability of attention to learn the temporal information of multi-channel sEMG data. Unlike prior works that aimed to combine CNN or LSTM architectures with self-attention, in this section, we will show that the proposed ViT-based architecture, which is solely based on the attention mechanism, has the capability to outperform the previous networks. The overall structure of the proposed TEMGNet architecture is shown in Fig. 4.3. In the following, first, we describe the building blocks of the proposed architecture, and then the overall structure of the network.

The proposed architecture is inspired by ViT [64], which closely follows the original transformer model. Within the TEMGNet framework, after completion of the pre-processing step, the collected sEMG data was segmented via a sliding window of

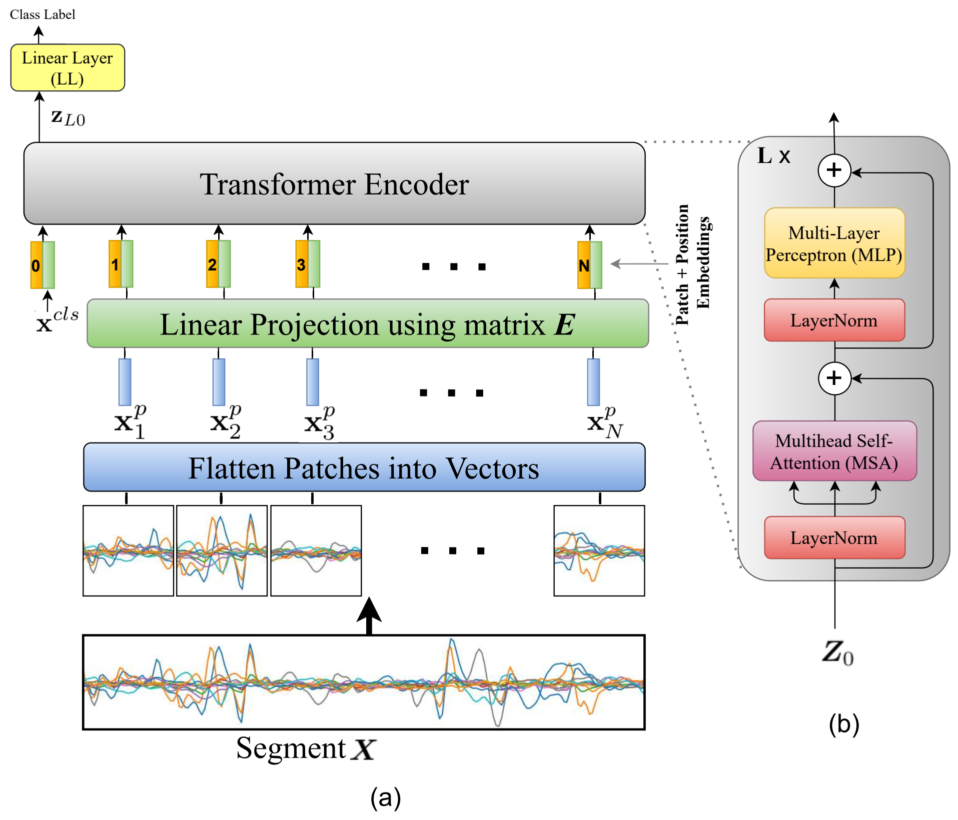


Figure 4.3: The proposed TEMGNet architecture: (a) Each segment of sEMG signal $\boldsymbol{X}$ is split into a sequence of fixed-size non-overlapping patches. The constructed patches are then flattened (the blue one) and projected linearly (the green block). The output of this step is referred to as "Patch Embedding." Afterward, position embeddings are added to the patch embeddings. The resulting sequence of these embedded patches is then fed to the transformer encoder (the gray block). For the classification, a trainable $[cls]$ token $\mathbf{x}^{cls}$ is added to the sequence. (b) The transformer encoder: This module consists of L layers, each consisting of two LayerNorm modules, an MLP, and an MSA module.

length 200 ms with steps of 10 ms (for comparison purposes, the results for a window of 300 ms are also provided). The segmentation step transforms the sEMG dataset into $\mathcal{D} = \{(\boldsymbol{X}_i, \mathrm{y}_i)\}_{i=1}^{M}$, consisting of M segments, where the i^{th} segment is denoted by $\boldsymbol{X}_i \in \mathbb{R}^{S\times W}$, for $(1 \leq i \leq M)$, with its associated label denoted by y_i. Here, S denotes the number of sensors, and W shows the number of samples collected at 2 kHz for a window of 200 ms (or 300 ms). The main objective of the TEMGNet architecture is to learn the mapping from the sequence of segment patches to its corresponding label y_i. As shown in Fig. 4.3, the TEMGNet architecture consists of the following modules; i.e., Patch Embeddings, Position Embedding, Transformer encoder, and finally a Multi-Layer Perceptron (MLP) head. The description of these blocks are as follows:

(i) **Patch Embeddings**: As shown in Fig. 4.3(a), at first, we split the segmented input $\boldsymbol{X}$ (for simplicity, we drop the segment index i) into N number of non-overlapping patches. Here, we set the size of each patch to $(S \times S)$; therefore, the number of patches will be equal to $N = W/S$. Each patch is then flattened into a vector $\mathbf{x}_j^p \in \mathbb{R}^{S^2}$, for $(1 \leq j \leq N)$. A linear projection is then applied to embed each vector into the model dimension d. For the linear projection, we used a matrix $\boldsymbol{E} \in \mathbb{R}^{S^2 \times d}$, which is shared among different patches. The output of this projection is called patch embeddings (Eq. 5 below).

In a similar way as in the BERT framework [71], the beginning of the sequence of embedded patches is appended with a trainable $[cls]$ token $\mathbf{x}^{cls}$, to capture the meaning of the entire segmented input. Finally, we will add position embeddings denoted by $\boldsymbol{E}^{pos} \in \mathbb{R}^{(N+1) \times d}$ to the patch embeddings that will allow the transformer to capture the positional information. The formulation governing patch and position embeddings is given by

$$\boldsymbol{Z}_0 = [\mathbf{x}^{cls}; \mathbf{x}_1^p \boldsymbol{E}; \mathbf{x}_2^p \boldsymbol{E}; \ldots; \mathbf{x}_N^p \boldsymbol{E}] + \boldsymbol{E}^{pos}. \tag{5}$$

(ii) **Position Embeddings**: The sEMG signal is sequential data that is presented in a specific order. If we change this order, the meaning of the input might also change. The transformer does not process the input sequentially and for each element, it combines information from the other elements through self-attention. Since the architecture of the transformer does not model the positional information, there is a need to explicitly encode the order of the input sequence so that the transformer knows that one piece is after another and not in any other permutation. This is where positional embedding comes in. Positional embedding is a form of identifier, a clear reference for the transformer that encodes the location information within the sequence. Therefore, positional embeddings are order or position identifiers added to the input to identify the relative position of each element in the sequence for the transformer. There are different ways to encode spatial information into the input of transformer using positional embedding, e.g., Sinusoidal positional embedding, Relative positional embeddings, Convolutional embedding, 1-dimensional positional embedding, and 2-dimensional positional embedding [64, 77]. Following [64], the standard trainable 1-dimensional positional embeddings are used in this study.

(iii) **Transformer Encoder**: The resulting sequence of vectors $\boldsymbol{Z}_0$ is fed as an input to a standard transformer encoder. We are basically treating all of the constructed patches as simple tokens provided into the transformer. In fact, the encoder block is similar to the main transformer encoder block proposed by [65]. As shown in Fig. 4.3(b), the transformer encoder consists of L identical layers. Each layer consists of two modules, i.e., a Multihead Self-Attention Mechanism (MSA) and an MLP module (defined later in Eqs. 6, 7). MSA is built based on the Self-Attention (SA) mechanism. The MLP module consists of two linear layers and a Gaussian Error Linear Unit (GELU) activation function.

To address the degradation problem, a layer-normalization [79] is applied, which is then followed by residual skip connections

$$\begin{aligned} \boldsymbol{Z}_l^{'} &= MSA(LayerNorm(\boldsymbol{Z}_{l-1})) + \boldsymbol{Z}_{l-1}, && (6) \\ \boldsymbol{Z}_l &= MLP(LayerNorm(\boldsymbol{Z}_l^{'})) + \boldsymbol{Z}_l^{'}, && (7) \end{aligned}$$

for $l = 1 \ldots L$. The final output of the transformer can be represented as follows

$$\boldsymbol{Z}_L = [\mathbf{z}_{L0}; \mathbf{z}_{L1}; \ldots; \mathbf{z}_{LN}], \tag{8}$$

where $\mathbf{z}_{L0}$ is used for classification purposes. Finally, $\mathbf{z}_{L0}$ is passed to a Linear Layer (LL), i.e.,

$$\mathbf{y} = LL(LayerNorm(\mathbf{z}_{L0}). \tag{9}$$

This completes the description of the proposed TEMGNet architecture. Next, we present the description of SA and MSA, respectively.

(iv) **Self-Attention (SA)**: Let us define the input sequence as $\boldsymbol{Z} \in \mathbb{R}^{N \times d}$ consisting of N vectors, each with an embedding dimension of d. The SA mechanism was first introduced in [65]. Generally speaking, the SA mechanism is defined with the aim of capturing the interaction between different vectors in $\boldsymbol{Z}$. In this regard, three different matrices, namely Queries $\boldsymbol{Q}$, Keys $\boldsymbol{K}$, and Values $\boldsymbol{V}$ are computed via a linear transformation, i.e.,

$$[\boldsymbol{Q}, \boldsymbol{K}, \boldsymbol{V}] = \boldsymbol{Z}\, W^{QKV}, \tag{10}$$

where $W^{QKV} \in \mathbb{R}^{d \times 3d_h}$ shows the learnable weight matrix. Here, d_h denotes the size of each vector in $\boldsymbol{Q}$, $\boldsymbol{K}$, and $\boldsymbol{V}$. After that, the pairwise similarity between each query and all keys is obtained using the dot-product of $\boldsymbol{Q}$ and $\boldsymbol{K}$, which is then scaled by $\sqrt{d_h}$ and translated into the probabilities $\boldsymbol{P} \in \mathbb{R}^{N \times N}$ using the softmax function as follows

$$\boldsymbol{P} = \text{softmax}(\frac{\boldsymbol{Q}\boldsymbol{K}^T}{\sqrt{d_h}}). \tag{11}$$

Finally, for each vector in the input sequence, the corresponding output vector resulted from the SA mechanism is computed by taking the weighted sum over all values $\boldsymbol{V}$ as follows

$$SA(\boldsymbol{Z}) = \boldsymbol{P}\boldsymbol{V}, \tag{12}$$

where $SA(\boldsymbol{Z}) \in \mathbb{R}^{N \times d_h}$. Such an attention mechanism helps the model to focus on important parts from a given sEMG input sequence.

(v) **Multihead Self-Attention (MSA)**: In the MSA, the SA mechanism is applied h times in parallel, allowing the model to attend to parts of the input sequence for each head differently. More specifically, MSA consists of h heads where each head has its own learnable weight matrix $\{W_i^{QKV}\}_{i=1}^{h}$. For the input sequence $\boldsymbol{Z}$, we applied the SA mechanism for each head (Eqs. (10)-(12)). Then the outputs of h heads are concatenated into a single matrix $[SA_1(\boldsymbol{Z}); SA_2(\boldsymbol{Z}); \ldots; SA_h(\boldsymbol{Z})] \in \mathbb{R}^{N \times h.d_h}$ and once again projected to obtain the final values as follows

$$MSA(\boldsymbol{Z}) = [SA_1(\boldsymbol{Z}); SA_2(\boldsymbol{Z}); \ldots; SA_h(\boldsymbol{Z})]W^{MSA}, \tag{13}$$

where $W^{MSA} \in \mathbb{R}^{h.d_h \times d}$ and d_h is set to d/h, this description completes the

Table 4.6: Descriptions of TEMGNet architecture variants.

Window size	Model ID	Layers	Model dimension d	MLP size	Params
200 ms	**1**	1	32	128	20,049
	2	2	32	128	32,657
	3	3	32	128	45,265
	4	1	64	256	64,625
300 ms	**1**	1	32	128	20,593
	2	2	32	128	33,201
	3	3	32	128	45,809
	4	1	64	256	65,713

proposed TEMGNet architecture.

Next, we present the results and experiments.

4.3.2 Experiments and Results

In this section, the performance of the proposed TEMGNet architecture is evaluated through a comprehensive set of experiments. In particular, we evaluated different variants of the TEMGNet architecture. The results are summarized in Table 4.6, where the performance of the model for window sizes of 200 ms and 300 ms is shown. For all model variants, we set the size of the input patch to 12×12. All models were trained using Adam optimizer [80] with betas = $(0.9, 0.999)$, and the weight decay set to 0.001. These models were trained with a batch size of 512. Cross-entropy loss was used for measuring classification performance.

Table 4.7 summarizes the classification accuracies for different TEMGNet architecture variants. The computed gesture recognition accuracy was averaged over all subjects. As shown in Table 4.7, by increasing the model dimension d from 32 to 64 (Model 1 to Model 4), the accuracy improved approximately by 2% for both window sizes. However, Model 4 had higher number of trainable parameters in comparison to Model 1, resulting in higher complexity (Table 4.6). In a second experiment, we examined the effect of increasing the number of layers. As shown in Table 4.7, for both window sizes of 200 ms and 300 ms, Model 2 had a higher accuracy than Model 1.

Table 4.7: classification accuracies for TEMGNet architectures variants. The STD represents the standard variation in accuracy over the 40 users.

	Model ID	1	2	3	4
Window 200 ms	Accuracy (%)	80.39	81.23	80.85	**82.05**
	STD (%)	5.86	6.31	6.32	**5.78**
	Model ID	**1**	**2**	**3**	**4**
Window 300 ms	Accuracy (%)	80.88	81.54	81.42	**82.93**
	STD (%)	5.97	5.99	5.94	**5.83**

However, by increasing the number of layers to 3, no improvements had been observed in classification accuracy. We used the statistical tests to show the significance level of different model variants, i.e., Model 1, 2, 3, and 4. Therefore, we followed [1,38] and used the Wilcoxon signed-rank test [67], considering each participant as a separate dataset. As shown in Fig. 4.4, the difference in accuracy between Model 1 and Model 4, for both window sizes of 200 ms and 300 ms was considered statistically significant by the Wilcoxon signed-rank test as the (**** : $p \leq 1.00e-4$).

In Fig. 4.4, a p-value is annotated by:

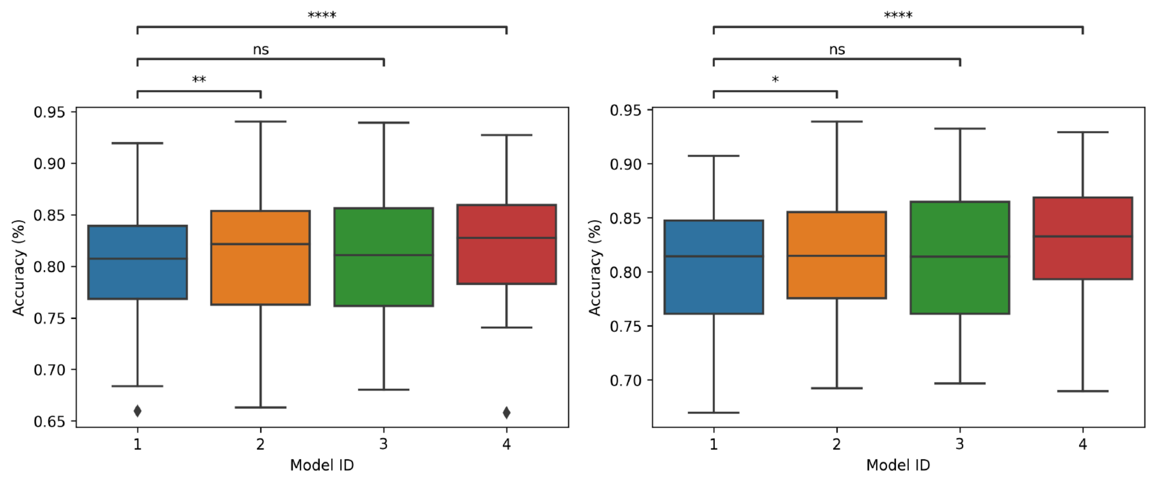
- Not significant (ns): $5.00\mathrm{e}-02 < \mathrm{p} \leq 1.00\mathrm{e}+00$,
- * : $1.00\mathrm{e}-02 < \mathrm{p} \leq 5.00\mathrm{e}-02$,
- ** : $1.00\mathrm{e}-03 < \mathrm{p} \leq 1.00\mathrm{e}-02$,
- *** : $1.00\mathrm{e}-04 < \mathrm{p} \leq 1.00\mathrm{e}-03$,
- **** : $\mathrm{p} \leq 1.00\mathrm{e}-04.\mathrm{s}$

In Fig. 4.4, the performance distribution across 40 users for each model is shown. The boxplot for each model shows the Interquartile Range (IQR), which is based on dividing the performance of each model for 40 users into quartiles. The median performance is shown by a horizontal line in each boxplot.

In Table 4.8, we provide comparisons with the state-of-the-art models [63] developed recently on the same dataset to illustrate the superior performance of the proposed TEMGNet architecture over its counterparts. The proposed TEMGNet

Table 4.8: Comparison between our methodology (TEMGNet) and previous works [4, 63].

		200 ms		300 ms	
		Params	**Accuracy (%)**	**Params**	**Accuracy (%)**
Reference [63]	4-layer 3rd Order Dilation	1, 102, 801	79.0	1, 102, 801	82.4
	4-layer 3rd Order Dilation (pure LSTM)	_	_	466, 944	79.7
	SVM	_	26.9	_	30.7
The TC-HGR	Model 1	49, 186	80.29	52, 066	80.84
	Model 4	94, 965	80.72	92, 945	81.65
The TEMGNet	Model 1	**20,049**	80.39	**20,593**	80.88
	Model 2	32, 657	81.23	33, 201	81.54
	Model 3	45, 265	80.85	45, 809	81.42
	Model 4	64, 625	**82.05**	65, 713	**82.93**

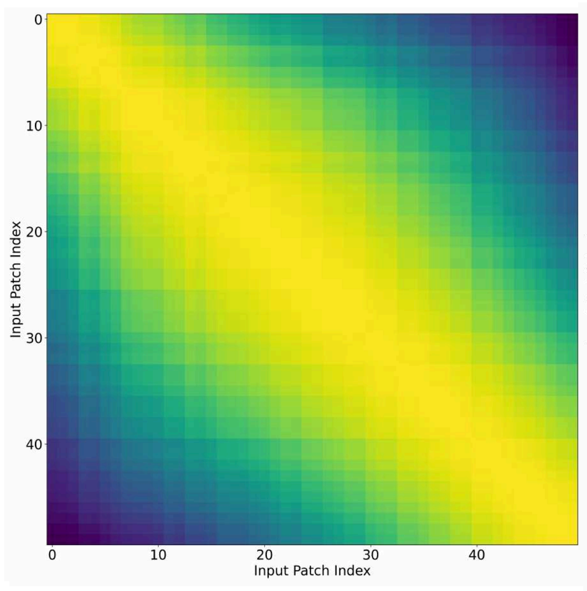


(a) Window size of 200 ms (b) Window size of 300 ms

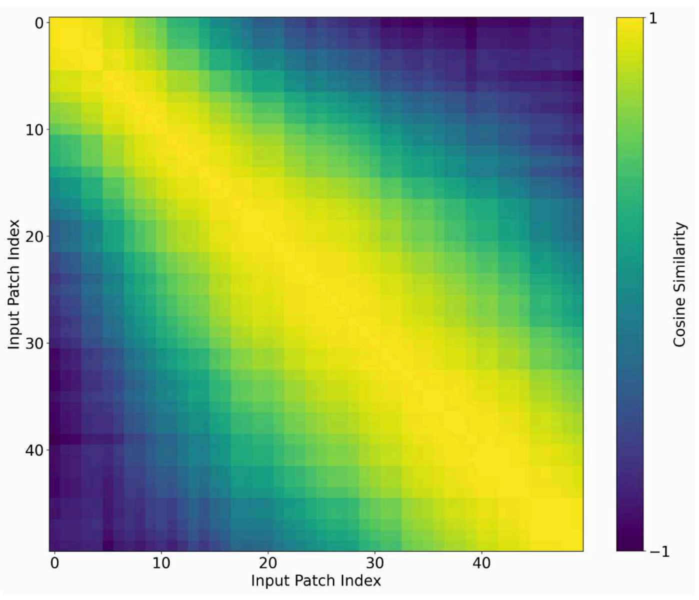
Figure 4.4: The accuracy boxplots for all TEMGNet architecture variants. Each boxplot shows the IQR of each model for 40 users. The Wilcoxon signed-rank test is used to compare the Model 1 (with a minimum number of parameters) with other models; i.e., Model 2, Model 3, and Model 4 (ns: $5.00e-02 < p \leq 1.00e+00$, $^{*}: 1.00e-02 < p \leq 5.00e-02$, $^{**}: 1.00e-03 < p \leq 1.00e-02$, $^{***}: 1.00e-04 < p \leq 1.00e-03$, $^{****}: p \leq 1.00e-04$).

method outperforms classical and state-of-the-art DNN-based models. More specifically, for a window size of 300 ms, the classification accuracy was 82.93% with only 65,713 number of trainable parameters, while in Reference [63], the authors reached 82.4% accuracy with 1,102,80 number of parameters. Moreover, for window size 200 ms, the accuracy for Model 1 and Model 4 was 80.39% and 82.05%, respectively. However, with the same window size, the best accuracy reported in Reference [63] was 79.0%.

Moreover, as shown in Table 4.8, we provide the comparisons with the TC-HGR architecture which was proposed in Section 4.2. More specifically, in Section 4.2 [4], we illustrated that by taking advantage of dilated causal convolutions, the receptive field of the network can be increased which results in extracting the temporal features of the sEMG signals. Although the TC-HGR architecture is an effective way to reduce the number of parameters while improving accuracy, the results for both window size of 200 ms and 300 ms is lower than the TEMGNet Architecture. More specifically, for a window size of 300 ms, the classification accuracy in the TC-HGR architecture is 81.65 with 92,945 number of parameters; while the accuracy in our proposed TEMGNet method is 82.93 with 65,713 number of parameters. This shows the capability of transformers in extracting the sEMG features for hand gesture classification.



(a) Model 1 for a Window size of 300 ms



(b) Model 4 for a Window size of 300 ms

Figure 4.5: Visualization of position embedding similarities for Model 1 and 4 in the Window size of 300 ms. The number of samples (W) collected at a frequency of 2 kHz for a window of 300 ms is 600. The size of each patch is set to (12×12); therefore, the number of patches is $N = 50$; i.e., the input patch index is from 0 to 49. Each row in (a) and (b) represents the similarity of the position embedding vector of each patch to all positional embeddings. It is shown that position embedding vectors learn distance in a segment. This means that the neighbors have higher similarities.

Fig. 4.5 shows the position embedding similarities for the window size of 300 ms. To encode position information, learnable position embedding was added to the patch embeddings. This is a key factor for the transformer to consider the sequential nature of the sEMG signals. Fig. 4.5 shows that position embedding vectors learn distance in a segment of sEMG signals. More specifically, the size of each patch was set to 12×12. Therefore, there are $N = 50$ patches for a window of 300 ms, i.e., the input patch index varied from 0 to 49. Each row in Fig. 4.5(a) and Fig. 4.5(b) represents the similarity of the position embedding vector of each patch to all positional embeddings. It is observed that the main diagonal in Fig. 4.5(a), (b) shows higher similarities between the neighboring ones. Moreover, position embedding similarities in Fig. 4.5(b) are greater than in Fig. 4.5(a), which means that the proposed Model 4 encode the sequential nature of sEMG signals better than Model 1.

4.4 Conclusion

In this chapter, first, we proposed a novel architecture referred to as the TC-HGR for HGR from sparse multichannel sEMG signals. The proposed model showed strong capability in addressing several existing challenges of gesture recognition based on the temporal convolutions and attention mechanism. We showed that by proper design of convolution-based architectures, we can extract temporal information of the sEMG signal and improve the performance. Moreover, the proposed architecture can reduce the required number of trainable parameters with respect to the state-of-the-art, which is a key enabling factor to reduce the complexity and embed DNN-based models into prostheses controllers. In the second part of the chapter, we proposed a novel transformer-based framework, the TEMGNet, for HGR from sEMG signals. We showed that the proposed architecture has the capacity to reduce the number of trainable parameters by 55 times and 2.45 times with respect to the state of the art [63] and the proposed TC-HGR, respectively, while improving the performance. This is a major step toward the utilization of deep learning for the control of prosthetic systems. It is noteworthy to mention that misplacement or displacement of sensors and time variability between days is an open topic of our research which is not addressed here. This can be considered as a limitation for this research work and a future research direction.

Chapter 5

Adaptive-Based Models for HGR

In the previous chapters, we focused on improving accuracy and reducing complexity of DNN-based models for the problem of sEMG-based HGR. The main objective of this chapter is to design a modern DNN-based gesture detection model that relies on minimal training data while providing high accuracy [6]. In this chapter, we introduce (for the first time) the concept of few-shot training for myoelectric systems. Few-shot learning is a variant of domain adaptation with the goal of inferring the required output based on just one or a few training observations. Few-shot learning minimizes the need for recalibration and would allow the user to retrain the ML core of control, by only few basic exercises instead of extensive recalibration procedures. For this purpose, we propose an innovative Few-Shot learning- Hand Gesture Recognition (FS-HGR). The proposed meta-learning FS-HGR architecture takes advantage of domain knowledge and requires a small amount of training data (when compared with traditional counterparts) to decode new gestures of the same or new users. In other words, the proposed FS-HGR generalizes after seeing very few observations from each class by combining temporal convolutions with attention mechanisms. This allows the meta-learner to aggregate contextual information from experience and to pinpoint specific pieces of information within its available set of inputs.

The chapter is organized as follows: In Section 5.1, we present the dataset used in development of the proposed FS-HGR framework together with the pre-processing step. The proposed FS-HGR architecture is developed in Section 5.2. Experimental results and different evaluation scenarios are presented in Section 5.3. Finally, Section 5.4 concludes the chapter.

5.1 Material and Methods

In this section, first, the database on which the proposed model is evaluated is described. Then, the pre-processing approach for preparing the data set will be explained.

Database: The proposed FS-HGR architecture was evaluated on the Ninapro [16, 17, 81] benchmark database, which is a publicly available dataset for hand gesture recognition tasks. Similar to the previous chapters, in this work, the second Ninapro database [16] referred to as the DB2 was utilized. As stated before, Delsys Trigno Wireless EMG system with 12 wireless electrodes (channels) was used in the DB2 dataset to collect electrical activities of muscles at a rate of 2 kHz. The dataset consists of signals collected from 28 men and 12 women with age 29.9 ± 3.9 years, among whom 34 are right-handed and 6 are left-handed. The DB2 consists of 50 gestures including wrist, hand, grasping, and functional movements along with force patterns from 40 healthy (intact-limb) subjects. The subjects repeated each movement 6 times, each time lasted for 5 seconds followed by 3 seconds of rest. More detail on the Ninapro database are described in Reference [16].

In addition to DB2, the 5th Ninapro database [19] referred to as the DB5 is also used for evaluation of the FS-HGR architecture. DB5 dataset is recorded with two Thalmic Myo armbands including 16 active single-differential wireless electrodes, recording muscular activity at a rate of 200 Hz. More specifically, the DB5 dataset consists of signals collected from 10 intact-limb users performing 52 movements including basic movements of the fingers, isometric, isotonic hand configurations, basic wrist movements, and grasping and functional movements. Each movement in the DB5 dataset is repeated 6 times, each lasting for 5 seconds followed by 3 seconds of rest. The DB5 dataset was presented in three sets of exercises which more details are provided in [19]. It is noteworthy to say that to follow the same criteria in [38,87] and also to have a fair comparison, in this chapter, we only consider the lower armband in DB5.

Pre-processing Step: Following the pre-processing procedure established in Subsection 3.2.1, we used a 1st order low-pass Butterworth filter to smooth the electrical activities of muscles. Along a similar path described in Subsection 3.2.1, we applied μ*-law* transformation to magnify the output of sensors with small magnitude (in a logarithmic fashion), while keeping the scale of those sensors having larger values over time.

As discussed in Subsection 3.2.1, we empirically observed that the normalization of the scaled sEMG signals is better than non-scaled sEMG signals. For example, the results obtained without scaling for a window of length 50 ms was 71.49%, while normalization of scaled sEMG signals has improved the results to 81.71%. In this work, we noticed a similar trend and as such continued using normalization of scaled sEMG signals. This completes a brief introduction of the utilized dataset and the introduced pre-processing step. Next, we develop the proposed Meta Learning-based FS-HGR framework.

5.2 The FS-HGR Architecture

Prior Research: A common strategy used for hand gesture recognition in recent works is applying DNN with the focus on improving hand gestures classification performance on "*never-seen-before repetitions*". Along this line of research, several state-of-the-art works [2, 8, 9, 24, 29–32, 48, 82, 83] mainly used the Ninapro database [16, 17, 81], which is a public dataset providing kinematic and sEMG signals from 52 finger, hand, and wrist movements. The Ninapro database is similar to data obtained in real-world conditions, and as such it allows development of advanced DNN-based recognition frameworks.

The common approach in recent studies [2, 8, 9, 24, 29–32, 48, 82, 83], following the recommendations provided by the Ninapro database, is to train DNN-based models on a training set consisting of approximately 2/3 of the gesture trials of each subject. The evaluation is then performed on the remaining trials constituting the test set. Although existing DNN techniques achieve promising performance on never-seen-before repetitions, they fail to function properly if the repetition is not extensively explored [66, 84, 85]. For example, in Reference [38], the authors reported the average accuracy over the 10 participants of the Ninapro DB5 for one to four training repetitions (one repetition is equal to 5 seconds of data). The accuracy decreases and the model fails to function properly if the repetition is not extensively explored. For example, for one to four training repetitions the accuracies are equal to 49.41 ± 5.82; 60.12 ± 4.79; 65.16 ± 4.46, and; 68.98 ± 4.46, respectively. Thus, for a new user or a new gesture, a significant amount of training should be conducted and the whole learning process should be redone, assuming a small variation between the new class and the previous classes. If the aforementioned change is more than minimal, there may be the need to recalibrate the whole process for all classes. In addition, existing DNN-based methodologies require large training datasets and perform poorly on tasks with only a few observations being available for training purposes.

In Reference [86], the authors proposed a domain adaptation method that maps both the original and target data into a common domain, while keeping the topology of the input data probability distributions. For this purpose, the authors used a local dataset, where the sEMG data was acquired by repetitive gripping tasks while data was collected from 8 subjects. In addition to the above, TL was also used to adopt a pre-trained model and leverage the knowledge acquired from multiple subjects and speed up the training process for the new users. In [38, 87], the authors proposed a TL-based algorithm adopting CNN to transfer knowledge across multiple subjects for sEMG-based hand gesture recognition. The authors in [38, 87], applied the Myo armband to collect sEMG signals and used the fifth Ninapro database, which contains data from 10 intact-limb subjects. The pre-training for each participant was done employing the training sets of the remaining nine participants and the average accuracy was obtained over the 10 participants of the Ninapro DB5 [19]. Finally, References [25, 49] applied deep learning along with domain adaptation techniques for inter-session classification to improve the robustness for the long-term uses. Due to the variability of the signal space, the generalizability of existing techniques is questionable and it is not clear how they would perform in real-life scenarios when the training data is limited. It is not clear how these models would perform on scenarios with larger number of subjects and postures. For example, in References [49], 7 subjects participated in the experiment and 7 movements were classified. As another example, in References [25], the authors separately used three datasets to train and evaluate their model based on HD-sEMG signals. The first dataset referred to as the CSL-HDEMG consists of 5 subjects performing 27 gestures. The second and third datasets referred to as CapgMyo DB-b and DB-c, respectively, consist of 23 subjects performing 8 gestures for DB-b and 12 gestures for DB-c. In summary, the number of subjects and movements in previous studies were relatively small - particularly in comparison with the Ninapro database, therefore, making it difficult to explore the generalizability of the existing techniques and has motivated us to focus on this relatively large-scale sEMG database.

In summary, there is an urgent need to develop adaptive learning methods with the focus on designing a classifier which can be adopted for new subjects based on only a few observations through a fast learning approach. This is a challenging task since many factors, such as electrode location and muscle fiber lengthening/shortening, can affect the collected sEMG signals. Moreover, the differences between users and the changes caused by amputations result in discrepancies between different conditions [13, 61]. To the best of our knowledge, this is the first time that *Few-shot Learning* is adopted in the literature to classify 49 hand gestures on *new* subjects using a small (one to five) number of training observations.

Contributions: Although classical pattern-recognition-based myoelectric control has been widely studied in academic settings over the last decades, the advanced methodologies have not been used in many commercial examples. This is due to a noticeable gap [13, 62] between real-world challenges and existing methodologies. Among the reasons for this gap are:

(i) *Training Time*: The first problem is the extended training time required by the end-user to mitigate the differences between the desired and performed movements. Such a training process, which is time consuming, tedious and unpleasant, can take up to several days in practice.

(ii) *Variability in the characteristics of sEMG Signals*: The second issue is the variability in nature of the sEMG signals. This variability is caused by (a) Time-dependent and stochastic nature of the neural drive to muscles; (b) Dependency of the neural drive to the dynamic and kinematics of tasks, and; (c) Variability in neural control strategies between different users and the changes caused by amputations. In addition, sEMG recording could vary based on electrode location. Given such variations, therefore, the probability distributions of sEMG signals may be different over time. Consequently, models trained based on some specific observations may not consistently and directly be reused over time. This would require retraining and recalibration, which cannot be done often in real-life applications.

Recently, DNNs have been designed and used by our team [2,8,9,88] and other research groups [29–32,82,89,90], for myocontrol, achieving superior classification performance than conventional approaches. For example, in Reference [29], which is among the first DNN-based methods developed for the analysis of sEMG data, it was shown that results of a DNN with a very simple architecture are comparable to the average result of classical methods. More specifically, the average classification accuracy obtained using a simple CNN architecture on Ninapro DB2 was reported as $60.27 \pm 7.7\%$. The average classification accuracy obtained using all the classical methods on this dataset is $60.28 \pm 6.51\%$. The best classical classification method (Random Forests (RF) with all features) obtained an average classification accuracy of $75.27 \pm 7.89\%$. It is worth noting that the DNN performance depends on several factors such as pre-processing and the designed architecture. The optimization of parameters is, therefore, fundamental for the performance. Consequently, there have been several efforts to design advanced DNN architectures. For instance, the average classification accuracy obtained based on recent works [30,32] is 82.95%, which shows their superior performance compared to classical methods. However, DNNs need large training data to achieve high performance. This may be feasible in laboratory conditions but poses constraints in the practical use of prostheses in real-life applications. There is an unmet need for the design of a modern gesture detection technique that relies on minimal training data while achieving high performance.

Meta-learning can be formalized as a sequence-to-sequence learning problem. The bottleneck is in the meta-learner's ability to internalize and refer to experience. To address this shortcoming for the gesture recognition task based on sparse multichannel sEMG, inspired by [66], we proposed a class of model architectures by combining temporal convolutions with attention mechanisms to enable the meta-learner to aggregate contextual information from experience. This integrated architecture allows the meta-learner to pinpoint specific pieces of information within its available set of inputs. Our main goal is to construct and train a hand gesture recognition model that can achieve rapid adaptation. Next, we first elaborate on the meta-learning concept.

The Meta-Learning Problem: A supervised learning task starts with a given dataset $\mathcal{D} = \{(\boldsymbol{X}_i, \mathrm{y}_i)\}_{i=1}^{M}$, consisting of M observations, where the i^{th} observation is denoted by $\boldsymbol{X}_i$, for $(1 \leq i \leq M)$, with its associated label denoted by y_i. The main objective is to learn a (possibly non-linear) function $f(\cdot)$ defined based on its

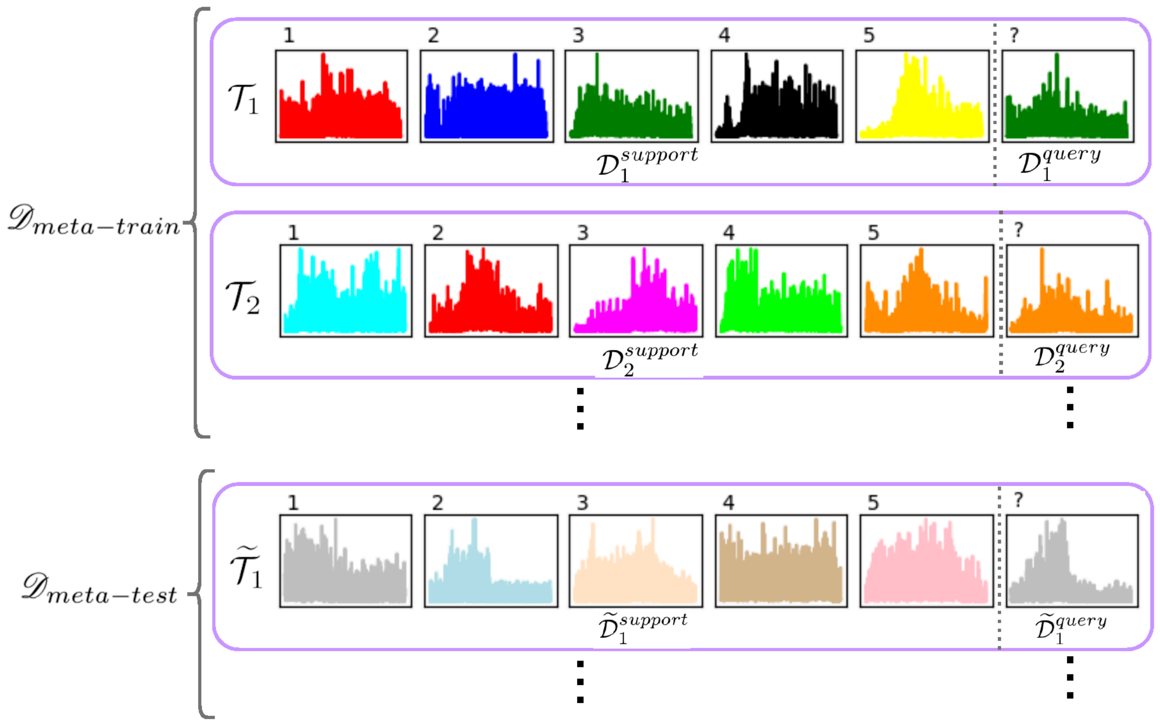

$\mathscr{D}_{meta-train}$
$\mathcal{T}_1$
1
2
3
4
5
?
$\mathcal{D}_1^{support}$
$\mathcal{D}_1^{query}$
$\mathcal{T}_2$
$\mathcal{D}_2^{support}$
$\mathcal{D}_2^{query}$
$\mathscr{D}_{meta-test}$
$\widetilde{\mathcal{T}}_1$
$\widetilde{\mathcal{D}}_1^{support}$
$\widetilde{\mathcal{D}}_1^{query}$

Figure 5.1: 5-way 1-shot classification: Each task $\mathcal{T}_j$, represented in a purple box, is associated with a support-set $\mathcal{D}^{support}$ and a query-set $\mathcal{D}^{query}$. Here, for constructing $\mathcal{D}^{support}$, first, 5 classes are selected from the $\mathscr{D}_{meta-train}$, and then one observation from each of these 5 classes are selected (each class is represented with a different colour and is associated with a label 1-5). $\mathcal{D}^{query}$ consists of 1 observation selected from one of those 5 classes. The $\mathscr{D}_{meta-test}$ is represented in the same approach, covering a different set of datasets, which do not include any classes presented in any of the datasets in $\mathscr{D}_{meta-train}$. Finally, $\mathscr{D}_{meta-val}$ is defined in the same way to determine the hyper-parameters of the model.

underlying parameters θ that maps each observation $\boldsymbol{X}_i$ to its corresponding label, $\mathrm{y}_i = f(\boldsymbol{X}_i; \theta)$. In a supervised learning approach, the dataset is divided into:

(a) The training data $\mathcal{D}_{train}$ used for learning the parameters θ of the model,

(b) The validation data $\mathcal{D}_{val}$ utilized for tuning the hyper-parameters of the model,

(c) The test data $\mathcal{D}_{test}$ for model evaluation.

In this context, we focused on meta-supervised learning, where the goal is generalization across tasks rather than across data points. Therefore, instead of using the aforementioned conventional data subsets (Items (a)-(c) above), we have a meta-set denoted by $\mathscr{D}$, which in turn splits into meta-train $\mathscr{D}_{meta-train}$, meta-validation

$\mathscr{D}_{meta-val}$, and meta-test $\mathscr{D}_{meta-test}$ sub-datasets. Furthermore, one needs to construct different tasks (as shown in Fig. 5.1) within each meta-dataset. Task $\mathcal{T}_j \in \mathscr{D}$ is episodic and is defined by two components, a support-set $\mathcal{D}_j^{support}$ and a query-set $\mathcal{D}_j^{query}$, i.e., $\mathcal{T}_j = (\mathcal{D}_j^{support}, \mathcal{D}_j^{query})$.

Within the context of meta-learning, our focus is specifically on few-shot learning (typically referred to as k-shot learning with k being a small integer), which is briefly described next. In a N-way k-shot classification, our goal is training on $\mathscr{D}_{meta-train}$, where the input is the support-set $\mathcal{D}_j^{support}$ and, a query instance $\boldsymbol{X}_j^{query} \in \mathcal{D}_j^{query}$. To be more precise, $\mathcal{D}_j^{support} = \{(\boldsymbol{X}_i, \mathrm{y}_i)\}_{i=1}^{k\times N}$, where N classes are selected from the meta-train set, and then k observations are selected from each of these classes. To make predictions about a new test data point, $\boldsymbol{X}_j^{query} \in \mathcal{D}_j^{query}$, we produce a mapping function $f(\cdot)$ that takes as input $\mathcal{D}_j^{support}$ and $\boldsymbol{X}_j^{query}$ to produce the label $\hat{\mathrm{y}}_j^{query} = f(\mathcal{D}^{support}, \boldsymbol{X}_j^{query}; \theta)$. Hyper-parameter selection is performed by using $\mathscr{D}_{meta-val}$. Generalization performance of the meta-learner is then evaluated on the $\mathscr{D}_{meta-test}$ [84].

Fig. 5.1 shows a $N = 5$-way $k = 1$-shot classification task, where inside each purple box is a separate dataset $\mathcal{T}_j$ consisting of the support-set $\mathcal{D}_j^{support}$ (on the Left-Hand Side (LHS) of the dashed line) and the query-set $\mathcal{D}_j^{query}$ (on the Right-Hand Side (RHS) of the dashed line). In the illustrative example of Fig. 5.1, we are considering a 5-way 1-shot classification task where for each dataset, we have one observation from each of the 5 classes (each given a label 1 to 5) in the support-set and 1 observation for evaluation from the query-set of that specific task. For training the model $\mathscr{D}_{meta-train}$ is used, where each Task $\mathcal{T}_j$ is drawn from $p\,(\mathcal{T})$ distribution, while during the test procedure $\mathscr{D}_{meta-test}$ is used, which consists of unseen tasks randomly sampled from a different distribution (i.e., $\widetilde{\mathcal{T}}_j \sim p\,(\widetilde{\mathcal{T}})$), where $p\,(\widetilde{\mathcal{T}})$ is similar in nature to $p\,(\mathcal{T})$. As shown in Fig. 1, task $\widetilde{\mathcal{T}}_j$ is associated with a dataset $\widetilde{\mathcal{D}}_j$ splitting into two parts, i.e., the support-set $\widetilde{\mathcal{D}}_j^{support}$ and the query-set $\widetilde{\mathcal{D}}_j^{query}$ (Fig. 1). For each task, we measure performance on the $\widetilde{\mathcal{D}}_j^{query}$ based on the knowledge of its cosponsoring $\widetilde{\mathcal{D}}_j^{support}$.

Description of the FS-HGR Model: In few-shot classification, the goal is to reduce the prediction error on data observations with unknown labels given a small training set. Inspired by [66], the proposed FS-HGR network receives as input a sequence of observation-label pairs $\mathcal{D}_j^{support} = \{(\boldsymbol{X}_i, \mathrm{y}_i)\}_{i=1}^{k\times N}$, followed by $\mathcal{D}_j^{query}$, which consists of an unlabelled observation. The meta-learning model predicts the label

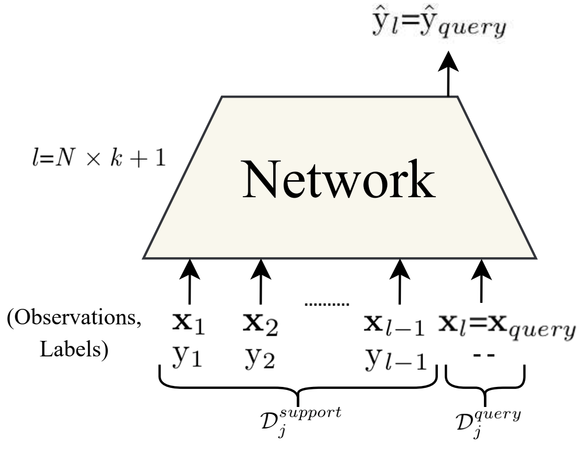

$\hat{y}_l = \hat{y}_{query}$
$l = N \times k + 1$
Network
(Observations, Labels)
$\mathbf{x}_1$ y_1
$\mathbf{x}_2$ y_2
$\mathbf{x}_{l-1}$ y_{l-1}
$\mathbf{x}_l = \mathbf{x}_{query}$ --
$\mathcal{D}_j^{support}$
$\mathcal{D}_j^{query}$

Figure 5.2: For each task $\mathcal{T}_j$, the set of observations and labels are concatenated together and sequentially fed to the model. The final observation is concatenated with a null label instead of True label. The network is supposed to predict the missing label of final observation based the previous labels that it has seen. In N-way k-shot classification, N shows the number of classes which are selected from whole set of labels, and k shows the observations that are sampled from each of those N classes.

of the final observation based on the previous labels that it has seen. During the training phase, first, we select N classes, with k observations per $\mathcal{D}_j^{support}$ (in terms of our running illustrative example, for each task, we have $k = 1$ observation from each of the underlying $N = 5$ classes). For constructing the $\mathcal{D}_j^{query}$, we select an extra observation from one of those selected classes. Afterwards, each set of the observations and labels are concatenated together (the final observation is concatenated with a null label instead of the ground truth label as it is used for evaluation purposes), and then all $(N \times k + 1)$ are sequentially fed to the network. Finally, the loss $\mathcal{L}_j$ is computed between the predicted and ground truth label of the $(N \times k + 1)^{th}$ observation. During such a training mechanism, the network learns how to encode the first $N \times k$ observations to make a prediction about the final observation [66]. The training procedure is described in Algorithm 1 and the schematic of the model is shown in Fig. 5.2.

5.2.1 The Building Modules of the FS-HGR Architecture

After completion of the pre-processing step, sEMG signals acquired from N_S number of sensors are segmented by a window of length of $W = 200$ ms selected to satisfy the acceptable delay time [40], i.e., the window length W is required to be under 300 ms. With a larger window of 300 ms, the results would likely improve. However,

Algorithm 1 The training procedure

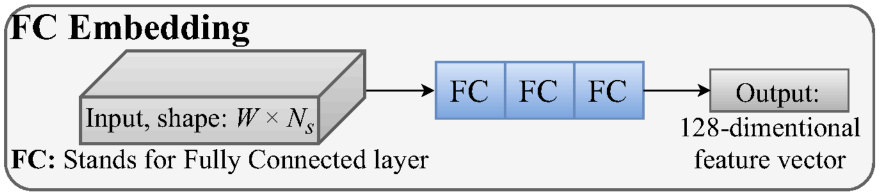
```
Input: $\mathscr{D}_{meta-train}$, and; mapping function $f(\cdot)$ with parameters $\theta$.
Require. $p(\mathcal{T})$: distribution over tasks
1: while not done do
2:     Sample batch of tasks $\mathcal{T}_j \sim p(\mathcal{T})$
3:     for all $\mathcal{T}_j$ do
4:         Split $\mathcal{T}_j$ into $\mathcal{D}_j^{support}$ and $\mathcal{D}_j^{query}$
5:         Predict the missing label of final observation of $\mathcal{T}_j$: $\hat{y}^{query} = f(\mathcal{D}_j^{support}, \mathcal{D}_j^{query}; \theta)$
6:     end for
7:     Update $\theta$ using $\Sigma_{\mathcal{T}_j \sim p(\mathcal{T})} \mathcal{L}_{\mathcal{T}_j}(\hat{y}^{query}, y^{query})$
8: end while
```

the use of shorter windows (e.g., 200 ms or 260 ms) provides an extra time (100 ms and 40 ms, respectively) to perform the pre-processing and classification tasks, which allows staying within the target 300 ms. A second reason for using a window of duration 200 ms is to perform a fair comparison with prior works [24,29,30,32,48,83] reported in Table 5.1. Finally, sliding window with steps of 50 ms is considered for segmentation of the sEMG signals. By using overlapping, there are more observations for training the underlying architecture. In [30, 83], a sliding window with steps of 100 ms was considered for segmentation of the sEMG signals. On the other hands, in [32,48] a sliding window with steps of 10 ms was used. In all these previous studies, the window size was 200 ms. The larger the overlapping size, the more training data are available, i.e., the more augmentation. However, extended augmentation increases the training time. In our work, for providing a fair comparison and to keep a reasonable training time, we considered a sliding window with steps of 50 ms, which is approximately between the values considered in the prior works. The building modules of the FS-HGR architecture are as follows:

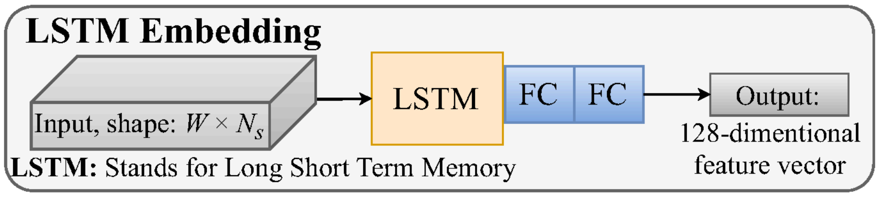
1) The Embedding Module: To develop the FS-HGR for few-shot learning, we aimed to first extract a 128-dimensional feature vector from each observation with size of ($W = 200 \times N_S = 12$), The "Embedding Module" is, therefore, used to extract a 128-dimensional feature vector, which is then provided as input to the proceeding modules within the proposed architecture.

   Adopting a proper Embedding Module has a significant effect on the results. For validating our claim, therefore, we utilized four different Embedding Modules:

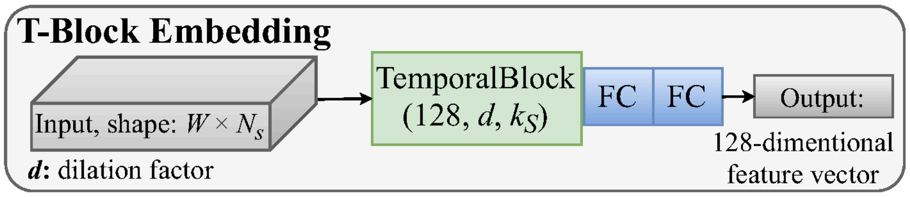
   (i) The first Embedding Module, referred to as the *FC Embedding*, consists


FC Embedding
Input, shape: $W \times N_s$
FC
FC
FC
Output:
128-dimentional feature vector
FC: Stands for Fully Connected layer

(a) FC Embedding Module


LSTM Embedding
Input, shape: $W \times N_s$
LSTM
FC
FC
Output:
128-dimentional feature vector
LSTM: Stands for Long Short Term Memory

(b) LSTM Embedding Module


T-Block Embedding
Input, shape: $W \times N_s$
TemporalBlock $(128, d, k_S)$
FC
FC
Output:
128-dimentional feature vector
d: dilation factor

(c) T-Block Embedding Module

Figure 5.3: **The Embedding Module**, which converts an input with size $(W \times N_S)$, W stands the window length and N_S shows the number of sensors (input features), to a 128-dimensional feature vector. (a) **FC Embedding Module**, which uses three FC layers to outputs a 128-dimensional feature vector. (b) **LSTM Embedding Module**, which adopts a LSTM layer followed by two FC layers. (c) **T-Block Embedding Module**, which consists of a Temporal Block with number of filters $f = 128$, kernel size $k_S = 2$, and dilation factor d, followed by two FC layers.

of three Fully Connected (FC) layers to output a 128-dimensional feature vector from each observation. The first FC layer in the FC Embedding Module is used to increase the input dimensional to $(W \times 128)$. Subsequently, the second (which is followed by ReLU activation function) and third FC layers with output size of 100 and 1, respectively, are adopted to reduce the sequence length of each observation to (1×128) (Fig. 5.3(a));

(ii) *LSTM Embedding*: Fig. 5.3(b) illustrates the second Embedding Module, referred to as the *LSTM Embedding*, which utilizes a LSTM layer as its first block followed by two FC layers. The LSTM layer takes the observation with input size 12 and converts it to an output with 128 features. Then, the two FC layers are adopted to reduce the observation's sequence length to 1;

(iii) *T-Block Embedding I*: This third Embedding Module utilizes the *TemporalBlock Module* (which will be described in next sub-section) consisting of $f = 128$ 1D-Convolutions with kernel size $k_S = 2$, and dilation factor $d = 1$ as its first block. The *TemporalBlock Module* is followed by two FC layers to decrease the input's sequence length to 1 as shown in Fig. 5.3(c), and;

(iv) *T-Block Embedding II*: This embedding is similar in nature to the one described above in Item (iii), however, here the goal is to examine the effect of increasing the size of the receptive field. As such, the fourth Embedding Module utilizes two *TemporalBlock Modules* with $d = 1$ and $d = 2$. It is noteworthy to mention that the first FC layer in both LSTM and T-Block Embedding modules are followed by ReLU activation function.

2) The TemporalBlock Module: Inspired by [2,8,43,44,66], the proposed FS-HGR few-shot learning architecture utilizes *Dilated Causal 1D-Convolutions* over the temporal dimension. The proposed architecture, therefore, provides several advantages over RNNs such as low memory requirement and faster training. In addition, and unlike conventional CNNs, by incorporation of dilated causal convolutions, we increased the receptive field of the network and as such benefit from the time-series nature of the input. As shown in Fig. 5.4(a), each *TemporalBlock* consists of two dilated causal 1D-convolutions, each with dilation factor d, kernel size k_S, and f number of filters. To learn the complex structure of the underlying data, each dilated causal 1D-convolutions is followed by a ReLU activation function. More details are provided in Algorithm 2. Finally, by concatenating the results and the input, the training speed can be considerably improved. This module takes an input with size $(C_{in} \times l)$ and output a tensor with size $(C_{out} \times l)$. Here, l denotes the sequence length and is equal to $(N \times k + 1)$.

3) The TemporalConvNet Module: The benefit that comes with the designed "TemporalConvNet" module is that its training procedure is much faster and efficient compared to LSTM or Gated Recurrent Unit (GRU) architectures. In other words, through this approach one complete sequence can be processed through only one forward pass, while in RNN-based models this, typically, needs

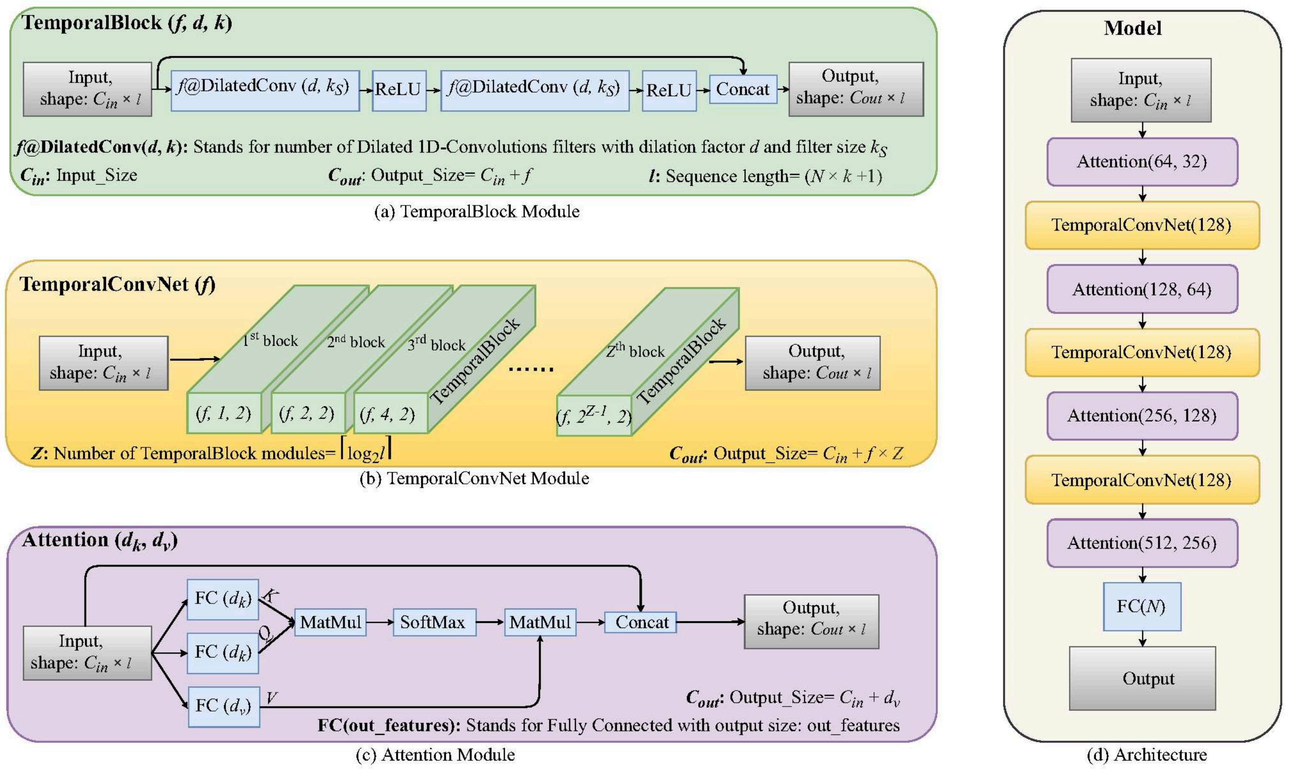

TemporalBlock (f, d, k)
Input, shape: $C_{in} \times l$
f@DilatedConv (d, k_S)
ReLU
f@DilatedConv (d, k_S)
ReLU
Concat
Output, shape: $Cout \times l$
f@DilatedConv(d, k): Stands for number of Dilated 1D-Convolutions filters with dilation factor d and filter size k_S
C_{in}: Input_Size
C_{out}: Output_Size= $C_{in} + f$
l: Sequence length= $(N \times k + 1)$
(a) TemporalBlock Module
TemporalConvNet (f)
Input, shape: $C_{in} \times l$
1st block
2nd block
3rd block
TemporalBlock
(f, 1, 2)
(f, 2, 2)
(f, 4, 2)
......
Zth block
TemporalBlock
(f, 2^{Z-1}, 2)
Output, shape: $Cout \times l$
Z: Number of TemporalBlock modules= $\lceil \log_2 l \rceil$
C_{out}: Output_Size= $C_{in} + f \times Z$
(b) TemporalConvNet Module
Attention (d_k, d_v)
Input, shape: $C_{in} \times l$
FC (d_k)
FC (d_k)
FC (d_v)
K
Q
V
MatMul
SoftMax
MatMul
Concat
Output, shape: $Cout \times l$
C_{out}: Output_Size= $C_{in} + d_v$
FC(out_features): Stands for Fully Connected with output size: out_features
(c) Attention Module
Model
Input, shape: $C_{in} \times l$
Attention(64, 32)
TemporalConvNet(128)
Attention(128, 64)
TemporalConvNet(128)
Attention(256, 128)
TemporalConvNet(128)
Attention(512, 256)
FC(N)
Output
(d) Architecture

Figure 5.4: (a) **The TemporalBlock Module**, which consists of f Dilated Causal 1D-Convolutions with dilation factor d and kernel size k_S. This module converts an input with C_{in} features to an output with C_{out} features. The sequence length of the input l is equal to $(N \times k + 1)$, which N shows the number of classes and k denotes the number of observations of each class. (b) **The TemporalConvNet Module**, which consists of a series of TemporalBlock modules (green ones). The kernel size of each TemporalBlock Module is equal to 2; however, their dilation factor d increases exponentially. (c) **The Attention Module**, which consists of three FC layers with output size d_k, d_k, and d_v, respectively, to produce matrix $\boldsymbol{Q}$, $\boldsymbol{K}$, and $\boldsymbol{V}$. (d) **The Architecture**, consisting of three TemporalConvNet modules (yellow ones), and four Attention modules (purple ones). Here, 128 denotes the number of filters f in Dilated 1D-Convolutions. The architecture is supposed to predict the missing label of the $(N \times k + 1)^{th}$ observation in each task $\mathcal{T}_j$.

Algorithm 2 The TemporalBlock Module

```
function: TemporalBlock(input, dilation factor $d$, kernel size $k_S$, number of filters $f$):
1: output1 = DilatedConv(input, $d$, $k_S$, $f$)
2: activation1 = relu(output1)
3: output2 = DilatedConv(activation1, $d$, $k_S$, $f$)
4: activation2 = relu(output2)
return concat(input, activation2)
```

Algorithm 3 The TemporalConvNet Module

```
function: TemporalConvNet(input, sequence length $l = (N \times k + 1)$, kernel size $k_S$, number of filters $f$):
1: $Z = \lceil \log_2 l \rceil$
2: for $i$ in $0, .., Z - 1$ do
3:     input = TemporalBlock(input, $2^i$, $k_S$, $f$)
4: end for
return input
```

several passes due to temporally linear hidden state dependency. The TemporalConvNet module consists of a series of TemporalBlock modules with exponentially growing dilation factors d. More specifically, as shown in Fig. 5.4(b), for an input with sequence length $l = (N \times k + 1)$, the TemporalConvNet consists of $Z = \lceil \log_2 l \rceil$ number of TemporalBlock modules. The dilation factors d for the TemporalBlock modules are equal to $[1, 2, 4, ..., 2^{Z-1}]$, respectively. The algorithm of this module is provided in Algorithm 3.

Algorithm 4 The Attention Module

```
function: Attention(input, key size $d_k$, value size $d_v$):
1: $\boldsymbol{K}$ = affine(input, $d_k$)
2: $\boldsymbol{Q}$ = affine(input, $d_k$)
3: $\boldsymbol{V}$ = affine(input, $d_v$)
4: logits = matmul($\boldsymbol{Q}$, transpose($\boldsymbol{K}$))
5: probs = softmax($\frac{\text{logits}}{\sqrt{d_k}}$)
6: output = matmul(probs, $\boldsymbol{V}$)
return concat(input, output)
```

4) The Attention Module: The final constituent module within the proposed FS-HGR architecture is referred to as the “Attention Module," included with the objective of pinpointing a specific type of information within the available

(possibly significantly large) context [65]. Attention mechanism has been recently utilized [89] within the context of sEMG-based hand gesture recognition, where the experiments showed attention's capability to learn a time-domain representation of multichannel sEMG data. By integrating the TemporalConvNet, described above, and the Attention Module, essentially we provided the FS-HGR architecture with the capability to access the past experience without any limitations on the size of experience that can be used effectively. Furthermore, in the FS-HGR framework we used the Attention Module at different stages to provide the model with the ability to learn how to identify and select pieces of useful information and its appropriate representation from its experience. As shown in Fig. 5.4(c), to get queries, keys, and values, three linear transformations are applied to the input. The attention mechanism then compares queries to each of the key values with a dot-product, scaled by $\sqrt{d_k}$, which results compatibility scores. To obtain attention distribution over the values, softmax function is applied to the scores. Then, we computed the weighted average of the values, weighted by the attention distribution. In practice, the keys, values, and queries are packed together into matrices $\boldsymbol{K}$, $\boldsymbol{V}$, and $\boldsymbol{Q}$, respectively. Then, the results and inputs are concatenated together. More details are provided in Algorithm 4 and Reference [65].

This completes description of the modules incorporated to construct the proposed FS-HGR framework. Next, we present its overall architecture.

The FS-HGR Architecture: The overall structure of the proposed FS-HGR architecture consists of four Attention modules, where the first three ones are followed by a TemporalConvNet module. The final Attention module is followed by a FC layer to produce the label of the final observation in each task $\mathcal{T}_j$. More specifically, after feeding each observation with size $(W \times N_S)$ to an Embedding Module, we obtained a 128-dimensional feature vector (Fig. 5.3). Then, for constructing each task $\mathcal{T}_j$ with sequence length l (Fig. 5.2), the set of observations (each observation is converted to a 128-dimensional feature vector) and labels are concatenated. The final observation in the sequence is concatenated with a null label instead of a True label. The network is supposed to predict the missing label of the final observation based on the previous labels that it has seen. In summary, to perform the hand gesture recognition task, the FS-HGR framework is constructed based on different modules as shown in

Fig. 5.4(d).

5.3 Experiments and Results

In this section, we describe a comprehensive set of experiments to analyse and evaluate the proposed FS-HGR framework. At stated previously, in few-shot classification, we would like to classify inputs in N classes when we have just k observations per class. More specifically, for N-way k-shot classification, to construct each Task $\mathcal{T}_j = (\mathcal{D}_j^{support}, \mathcal{D}_j^{query})$, for $(1 \leq j \leq N_{\text{Tasks}})$, first, we randomly select N classes from the total number of $\mathcal{N}$ available classes in the meta-set $(N \ll \mathcal{N})$. Then, we select k observations from each of those selected N classes. These k observations together constitute the support-set $\mathcal{D}_j^{support}$. An additional observation is randomly selected to form the query-set $\mathcal{D}_j^{query}$. In the experiments, there are a total of $\mathcal{N} = 49$ classes in the meta-set, each class corresponding to a specific hand gesture. For example, consider the $N = 10$-way $k = 5$-shot classification scenario. To construct each Task, we randomly select $N = 10$ out of the $\mathcal{N} = 49$ available classes and then from each class randomly select $k = 5$ observations to form the support-set $(\mathcal{D}_j^{support})$ for the j^{th} Task. Additionally, one extra observation is randomly selected from one of the $N = 10$ classes to form the query-set $(\mathcal{D}_j^{query})$ for the j^{th} Task. In the $N = 10$-way $k = 5$-shot classification scenario, therefore, each task consists of $N \times k + 1 = 51$ number of observations selected randomly. In a training experiment, we consider $10,000$ iterations per epochs. Therefore, with a batch-size of 64, we create $640,000$ tasks per epoch. We consider 25 epochs for training. As a final note, within the few-shot learning context, it is common to report the results based on different values of N and k. In the experiments, we followed the common practice of using $N = 5$ and $N = 10$ together with $k = 1$ and $k = 5$. By increasing the number of classes (N) in each Task $\mathcal{T}_j$, the classification accuracy will decrease. At the same time, by increasing the number of observations per class (k), the classification accuracy is expected to improve as there are more observations from each of the underlying N classes.

In all experiments, Adam optimizer was used for training purposes with learning rate of 0.0001. Different models were trained with a mini-batch size of 64 except in 10-way 5-shot classification where mini-batch size of 32 was used. For measuring

Table 5.1: Experiment 1: 5-way, 1-shot, 5-shot, and 10-shot classification accuracies on *new repetitions with few-shot observation*. The classification on new repetitions with few-shot observation are performed by using Meta-supervised Learning approach. This table also shows a comparison between our methodology (Meta-supervised) learning and previous works where Supervised learning methodology, i.e., DNN and Classical ML methods are used.

Proposed Method (FS-HGR)		**5-way Accuracy (%) ± STD**		
Meta-Supervised Learning	**Embedding Module**	**1-shot**	**5-shot**	**10-shot**
	FC Embedding	72.59 ± 0.15	85.13 ± 0.15	89.26 ± 0.14
	LSTM Embedding	**75.03** ± 0.15	84.06 ± 0.14	88.45 ± 0.12
	T-Block Embedding I	73.46 ± 0.15	**85.94** ± 0.13	89.40 ± 0.14
	T-Block Embedding II	74.89 ± 0.15	85.88 ± 0.14	**89.70** ± 0.15
Previous Works		**Method**	**Accuracy (%)**	
DNN-based Methods	Wei *et al.* [32]	CNN	**83.70**	
	Hu et al. [30]	Hybrid CNN-RNN	82.20	
	Ding *et al.* [48]	CNN	78.86	
	Zhai *et al.* [83]	CNN	78.71	
	Geng *et al.* [24]	CNN	77.80	
Classical Methods	Zhai *et al.* [83]	SVM	**77.44**	
	Atzori *et al.* [29]	RF	75.27	
	Pizzolato *et al.* [19]	RF	72.25	

the classification performance, the loss $\mathcal{L}_j$ was computed between the predicted and ground truth label of $(N \times k + 1)^{th}$ observation in each task $\mathcal{T}_j$. The average loss was computed using Cross-entropy loss. Finally, the average accuracy is reported on the $(N \times k + 1)^{th}$ observation. In the following, we present three evaluation scenarios.

Experiment 1: Classification on New-Repetitions with Few-Shot Observation. The first experiment shows that our proposed network is applicable when we had new repetitions with few-shot observation on the target. We evaluated our proposed architecture when $\mathscr{D}_{meta-train}$ consisted of the 2/3 of the gesture trials of each subject (following Reference [29], repetitions 1, 3, 4, and 6 repetitions were used for training purposes), and $\mathscr{D}_{meta-test}$ consisted of the remaining repetitions. Table 5.1 shows our results when using few-shot classification as well as previous works which used supervised learning. From Table 5.1, it can be observed that the proposed FS-HGR architecture outperformed existing methodologies when evaluated based on the same setting, i.e., 89.70% best accuracy with the FS-HGR compared to 83.70%

Table 5.2: Train and Test time for one Task $\mathcal{T}_j$ using 5-way 1-shot, 5-way 5-shot, and 5-way 10-shot for Experiment 1.

The Embedding Module	**5-way Accuracy**					
	1-shot		**5-shot**		**10-shot**	
	train time	**test time**	**train time**	**test time**	**train time**	**test time**
FC Embedding	0.736 ms	0.175 ms	3.034 ms	0.856 ms	6.378 ms	1.428 ms
LSTM Embedding	1.242 ms	0.537 ms	4.930 ms	1.598 ms	10.417 ms	2.815 ms
T-Block Embedding I	1.41 ms	0.382 ms	6.46 ms	1.352 ms	13.59 ms	2.396 ms
T-Block Embedding II	4.27 ms	0.488 ms	19.05 ms	1.841 ms	38.43 ms	3.613 ms

best accuracy achieved by the state-of-the-art.

Reference [29], used different classifiers such as KNN, SVM, RF, and LDA. The average classification accuracy obtained using all the classical methods on the DB2 dataset is $60.28 \pm 6.51\%$. They show that the highest average classification accuracy is $75.27 \pm 7.89\%$, obtained with RF. Reference [83] showed that the average accuracy of SVM on all movement types is 77.44%. Finally, in Reference [19], the best accuracy is reported with RF classifier, which is $72.25 \pm 7.13\%$. It can be seen from Table 5.1 that DNN-based methods provide improved performance.

The average of training and testing times for one Task $\mathcal{T}_j$ using 1-shot, 5-shot, and 10-shot for Experiment 1 are summarized in Table 5.2. It is noteworthy to say that the time of processing depends on the hardware. In this work, we used a "NVIDIA's GeForce GTX 1080 Ti Graphic Cards".

Experiments 2: Classification on New-Subject with Few-Shot Observation. In this scenario, like the previous experiment, the second Ninapro database DB2 was utilized. It consists of 49 gestures plus rest from 40 intact-limb subjects. In this experiment, to validate our claim that the proposed FS-HGR architecture can classify hand gestures of new subjects just by training with a few observations, we split the DB2 database into $\mathscr{D}_{meta-train}$, $\mathscr{D}_{meta-val}$, and $\mathscr{D}_{meta-test}$ such that the subjects in these meta-sets are completely different (i.e., there is no overlap between the meta-sets). In other words, $\mathscr{D}_{meta-train}$ consists of the first 27 subjects, while $\mathscr{D}_{meta-val}$ includes the sEMG signals from the 28^{th} subject to 32^{ed} subject (5 subjects). Finally, we evaluated our model on the remaining subjects, i.e., $\mathscr{D}_{meta-test}$ consists of the final 8 subjects in the DB2 database.

It is noteworthy to mention that the proposed network is trained once and shared across all participants (which is different from previous works that trained the model

Table 5.3: Experiment 2(a): 5-way and 10-way, 1-shot and 5-shot classification accuracies based on *new subjects with few-shot observation.* In this experiment, we adopted four different Embedding Modules: (i) FC Embedding; (ii) LSTM Embedding; (iii) T-Block Embedding I, and; (iv) T-Block Embedding II.

The Embedding Module	**5-way Accuracy (%) ± STD**		**10-way Accuracy (%) ± STD**	
	1-shot	**5-shot**	**1-shot**	**5-shot**
FC Embedding	62.87 ± 0.13	78.90 ± 0.15	43.47 ± 0.15	68.59 ± 0.14
LSTM Embedding	64.46 ± 0.16	79.82 ± 0.14	49.58 ± 0.17	69.93 ± 0.19
T-Block Embedding I	**67.81** ± 0.14	81.08 ± 0.14	50.31 ± 0.14	69.94 ± 0.18
T-Block Embedding II	66.98 ± 0.15	**81.29** ± 0.17	**52.05** ± 0.15	**70.71** ± 0.18

Table 5.4: Comparison of 5-way, 1-shot and 5-shot classification accuracies between the Experiment 2(a) and 2(b) based on *new subjects with few-shot observation.*

The Embedding Module	**Experiment 2(a)**		**Experiment 2(b)**	
	5-way Accuracy (%) ± STD			
	1-shot	**5-shot**	**1-shot**	**5-shot**
FC Embedding	62.87 ± 0.13	78.90 ± 0.15	72.69 ± 0.15	86.08 ± 0.14
LSTM Embedding	64.46 ± 0.16	79.82 ± 0.14	75.56 ± 0.17	89.14 ± 0.12
T-Block Embedding I	**67.81** ± 0.14	81.08 ± 0.14	75.11 ± 0.14	89.66 ± 0.13
T-Block Embedding II	66.98 ± 0.15	**81.29** ± 0.17	**77.08** ± 0.15	**90.47** ± 0.12

separately for each participant). For constructing task $\mathcal{T}_j$, however, we can feed data in two different approaches:

- *Experiment 2(a)*: In the first approach, for constructing $\mathcal{D}_j^{support}$ for each task $\mathcal{T}_j$, we selected all of the N classes from a specific user, which was randomly selected from the existing participants. This is the more realistic and practical scenario.

- *Experiment 2(b)*: In the second approach, for constructing $\mathcal{D}_j^{support}$, N classes were selected from different participants.

Table 5.3 shows few-shot classification accuracies for Experiment 2(a) based on four different embedding modules. The adaptive learning method of the proposed FS-HGR focuses on transfer learning information between a source and a target domain despite the existence of a distribution mismatch between $\mathscr{D}_{meta-train}$ and $\mathscr{D}_{meta-test}$. The results reported in Table 5.3 show that the proposed mechanism achieves acceptable results despite the fact that the sEMG signals are user-dependent. Table 5.4 shows a comparison of 5-way classification accuracies between Experiments 2(a) and 2(b). As was it expected, Experiment 2(b) achieved better results, which is due to the presence of variations among the probability distribution of sEMG signals obtained from different subjects. However, this is not a practical setting as in practice all of the N classes in $\mathcal{D}_j^{support}$ comes from the same user (i.e., Experiment 2(a)).

It is worth mentioning that Experiment 2(a) is the more realistic and challenging one, while Experiment 2(b) is included for completeness and comparison purposes. The rational behind Experiment 2(a) is that a prosthesis hand will be utilized by a user, therefore, the model should be able to distinguish different hand movements of this specific person. In this sense, Experiment 2(a) is more realistic than Experiment 2(b). In Experiment 2(a), for constructing $\mathcal{D}_j^{support}$, we selected all the N classes from a specific user, which was randomly selected from the participants. However, in Experiment 2(b), for constructing $\mathcal{D}_j^{support}$, N classes were selected from different participants, which implies that observations were obtained from different distributions/people, so the model would more easily discriminate these observations. For Experiment 2(a), we have conducted a statistical hypothesis test to evaluate if there is significant evidence to reject the hypothesis that lower and higher shots have similar accuracies. We followed Reference [38] and used the Wilcoxon signed-rank test [67] considering each participant as a separate dataset. Table 5.5 compares accuracy for each subject in the test set in Experiment 2(a) for 5-way 1-shot and 5-way 5-shot. The difference in accuracy between 1 and 5 shots was considered statistically significant by the Wilcoxon signed rank test as the *($p < 0.05$)*.

Experiment 3: Classification on New-Gestures with Few-Shot Observations: In this scenario, the goal is evaluating the capability of the proposed FS-HGR architecture when the target consists of solely out-of-sample gestures (i.e., new gestures with few-shot observation). Performing well in this task allows the model to evaluate new observations, exactly one per novel hand gesture class. In this experiment, the Ninapro database DB2 was used. The DB2 dataset includes three sets

Table 5.5: 5-way, 1-shot and 5-shot classification accuracies based on *new subjects with few-shot observation* (Experiment 2(a)). In this experiment, we obtained the accuracy for each subject in the test set using T-Block Embedding II. The Wilcoxon signed rank test is applied to compare the different shots (e.g., 1-shot and 5-shot). Null hypothesis is rejected when $H_0 = 0$ ($p < 0.05$).

The Subject	**5-way Accuracy (%)**	
	1-shot	**5-shot**
Subject 33	71.94	85.62
Subject 34	73.48	85.49
Subject 35	64.84	77.68
Subject 36	64.58	78.53
Subject 37	64.25	82.52
Subject 38	63.75	77.81
Subject 39	61.82	76.50
Subject 40	71.29	86.30
$H_0 = 0$ ($p < 0.05$)	0 (0.01172)	-

Table 5.6: Experiment 3: 5-way, 1-shot, 5-shot, and 10-shot classification accuracies based on *new gesture with few-shot observation.*

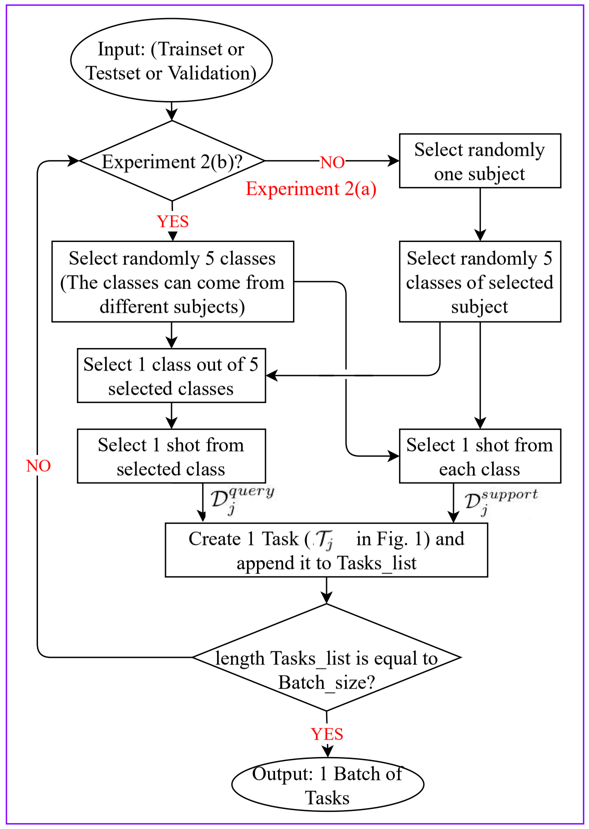
The Embedding Module	**5-way Accuracy (%) ± STD**		
	1-shot	**5-shot**	**10-shot**
FC Embedding	45.94 ± 0.16	67.20 ± 0.14	79.87 ± 0.15
LSTM Embedding	46.05 ± 0.16	71.76 ± 0.14	81.58 ± 0.16
T-Block Embedding I	**49.78** ± 0.15	71.57 ± 0.13	83.41 ± 0.14
T-Block Embedding II	45.48 ± 0.17	**73.36** ± 0.17	**83.97** ± 0.17

of exercises denoted by Exercise B, C, and D. Exercise B includes 8 isometric and isotonic hand configurations and 9 basic movements of the wrist; Exercise C consists of 23 grasping and functional movements; and finally, Exercise D consists of 9 force patterns. For training purposes, $\mathscr{D}_{meta-train}$ consisted of the first 34 gestures of each user, which is equal to approximately 68% of the total gestures. $\mathscr{D}_{meta-val}$ included 6 gestures or 12% of the total gestures. The remaining gestures (9 gestures), were used in $\mathscr{D}_{meta-test}$ for evaluation purposes. Exercises B and C were, therefore, used for training and validation, and Exercises D, with different gestures, were used for test purposes. Table 5.6 shows the efficiency of the proposed model when we had out-of-sample gestures in the target. The model predicted unknown class distributions in scenarios where few observations from the target distribution were available.

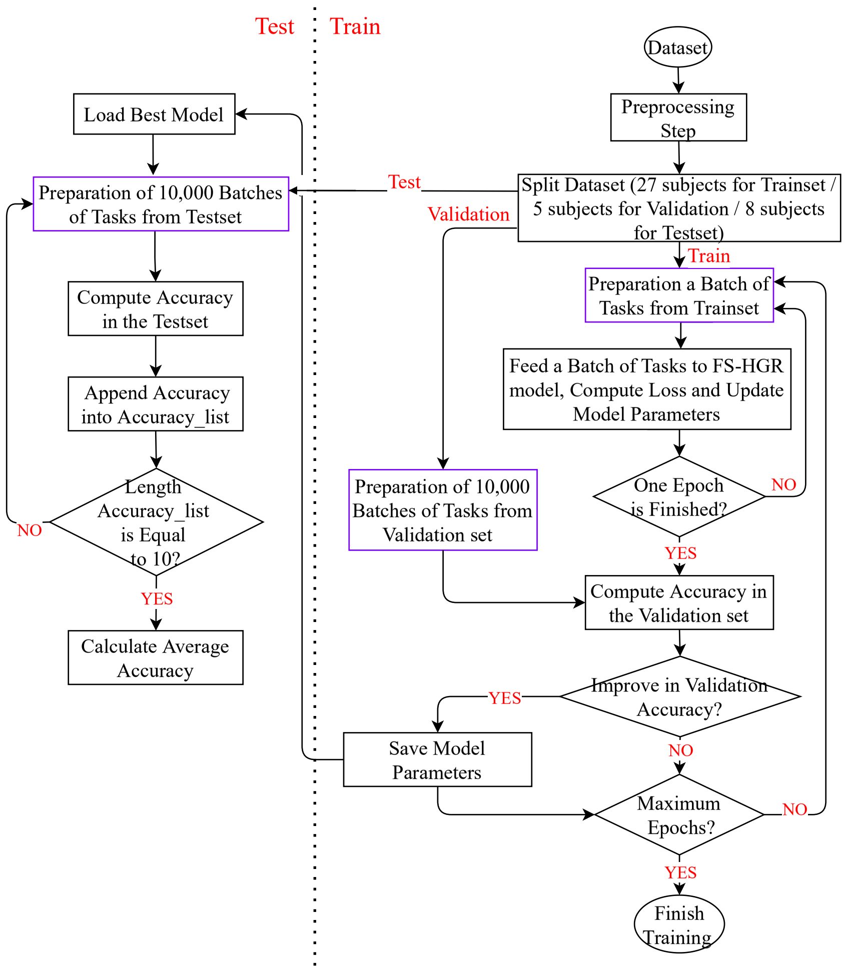
Flowchart of the FS-HGR Framework: We have included a flowchart (Fig. 5.5) for the proposed method applied to Experiment 2. Experiments 1 and 3 are similar in nature to Experiment 2. Fig. 5.5(a) shows the preparation of a batch of tasks for Experiment 2(a) and 2(b). Fig. 5.5(b) shows the training, validation, and test steps.

Comparison with State-of-Art TL-based Model: Moreover, we have compared the proposed FS-HGR framework with the TL technique of Reference [38] for cross-user scenarios. To provide a fair comparison, we have used the same public dataset utilized in [38], i.e., Ninapro DB5. The Ninapro DB5 [19] was recorded with the Myo Armband, and contains data from 10 healthy participants performing a total of 53 movements (rest included) divided into three exercise sets. The performance of their proposed TL-based architecture is investigated based on the second exercise set of DB5, which contains 18 (rest included) number of gestures. More specifically, in their proposed approach, the pre-training for each participant was performed by employing the training sets of the remaining nine participants. Finally, the average accuracy over the 10 participants was reported. Furthermore, the data is first separated by applying sliding windows of 52 samples (260 ms) with an overlap of 235 ms. We followed the same criteria and obtained the average accuracy over the 10 participants for cross-user model (Experiment 2(a)) for 5-way 5-shot and 5-way 10-shot classifications. In addition to the results of Reference [38], we have included results of cross-user models based on classical hand-crafted features developed in References [16,19,91–93], coupled with traditional classifiers (RF, and LDA). Results are reported in Table 5.7. It is observed that the proposed FS-HGR method outperforms classical and state-of-the-art TL-based models over this cross-user scenario.

Providing the Results for Ninapro DB5: First, we evaluate the performance of the proposed FS-HGR architecture for *new repetitions* with few-shot examples for DB5 dataset. We considered 2/3 of the gesture repetitions of each subject for training and validation purpose. The remaining repetitions constitute the test set. Table 5.8 shows the performance of model when FS-HGR is applied for HGR task. In this experiment, we used two Embedding Modules, i.e., FC and LSTM Embedding.


Input: (Trainset or Testset or Validation)
Experiment 2(b)?
NO
Experiment 2(a)
Select randomly one subject
YES
Select randomly 5 classes (The classes can come from different subjects)
Select randomly 5 classes of selected subject
Select 1 class out of 5 selected classes
Select 1 shot from selected class
Select 1 shot from each class
NO
$\mathcal{D}_j^{query}$
$\mathcal{D}_j^{support}$
Create 1 Task ($\mathcal{T}_j$ in Fig. 1) and append it to Tasks_list
length Tasks_list is equal to Batch_size?
YES
Output: 1 Batch of Tasks

(a) Flow chart showing the preparation of a Batch of Tasks for Experiments 2(a) and 2(b)


Test
Train
Dataset
Preprocessing Step
Split Dataset (27 subjects for Trainset / 5 subjects for Validation / 8 subjects for Testset)
Test
Validation
Train
Load Best Model
Preparation of 10,000 Batches of Tasks from Testset
Compute Accuracy in the Testset
Append Accuracy into Accuracy_list
Length Accuracy_list is Equal to 10?
NO
YES
Calculate Average Accuracy
Preparation a Batch of Tasks from Trainset
Feed a Batch of Tasks to FS-HGR model, Compute Loss and Update Model Parameters
One Epoch is Finished?
NO
YES
Preparation of 10,000 Batches of Tasks from Validation set
Compute Accuracy in the Validation set
Improve in Validation Accuracy?
YES
NO
Save Model Parameters
Maximum Epochs?
NO
YES
Finish Training

(b) Flow chart for training, validation, and test steps for Experiment 2

Figure 5.5: (a) Flowchart for preparation of a batch of tasks for Experiment 2.(b) Flowchart for training, validation, and test steps of Experiment 2. Each purple box in (b) represents one or more repetitions of (a).

Table 5.7: Comparison of Experiment 2(a) on NinaproDB5 database between Classical Methods [16, 19, 91–93], DNN Methods [38], and our proposed few-shot learning approach (FS-HGR).

Duration	**Our Proposed Few-shot Learning Method**	
	T-Block Embedding I	T-Block Embedding II
5-way 5-shot (1.3 seconds)	61.63 ± 5.09	64.65 ± 4.88
5-way 10-shot (2.6 seconds)	65.71 ± 4.79	**67.69 ± 4.58**

Duration	**Classical Methods**				**DNN Methods**	
	[91] (RF)	[92] (LDA)	[16, 19] (LDA)	[93] (RF/LDA)	[38]	[38] + TL
10 seconds	50.85 ± 4.29	50.08 ± 4.63	46.85 ± 4.81	53.00 ± 3.85 (RF)	55.65 ± 4.38	60.12 ± 4.79
5 seconds	40.70 ± 5.84	40.86 ± 6.91	37.60 ± 6.67	42.26 ± 5.78 (LDA)	46.06 ± 6.09	49.41 ± 5.82

Table 5.8: 5-way 1-shot, 5-way 5-shot, and 5-way 10-shot classification accuracies based on *new repetitions with few-shot observation* for Ninapro DB5. In this experiment, we adopted two different Embedding Modules: (i) FC Embedding and (ii) LSTM Embedding

The Embedding Module	**5-way Accuracy (%) ± STD**		
	1-shot	**5-shot**	**10-shot**
FC Embedding	66.75 ± 0.14	79.48 ± 0.13	79.56 ± 0.15
LSTM Embedding	69.61 ± 0.15	78.75 ± 0.14	83.17 ± 0.15

Table 5.9: 5-way 1-shot, 5-way 5-shot, and 5-way 10-shot classification accuracies based on *new gestures with few-shot observation* for Ninapro DB5. In this experiment, we adopted two different Embedding Modules: (i) FC Embedding and (ii) LSTM Embedding

The Embedding Module	**5-way Accuracy (%) ± STD**		
	1-shot	**5-shot**	**10-shot**
FC Embedding	40.23 ± 0.14	65.95 ± 0.15	75.29 ± 0.15
LSTM Embedding	39.56 ± 0.15	65.14 ± 0.14	74.91 ± 0.16

It can be observed that the Embedding Module could affect the overall accuracy. Moreover, by increasing the number of shots, the accuracy will be improved. It is noteworthy to mention that the performance of the proposed FS-HGR architecture for this experiment is evaluated based on the second set of exercises DB5, which includes 17 movements.

Moreover, it is shown that the proposed FS-HGR model can be used for new movements based on only a few examples. More precisely, the test set consists of completely new gestures. This is a challenging task because unlike the experiment for *new repetitions*, here, the probability distribution of the test set is not the same as training set. In other words, the model leverage the experience from source gestures to adapt to the variations in the new gestures in the target. Table 5.9 shows the efficiency of the proposed model when we had out-of-sample movements in the test set. The model predicted unknown class distributions in scenarios where few observations from the target distribution were available. It is worth to mention that in this experiment, 52 movements in DB5 are used. More specifically, 68% of total gestures are considered for the training purpose, while the validation set consists of 12% of the total gestures. The test set consists of the remaining gestures (20% of the total gestures).

Discussions on the Training Time of the FS-HGR Framework: It is worth noting that adoption of few-shot learning within the FS-HGR framework has resulted in reduction in the required training time for users. In previous studies, such as [24,29,30,32,48,83], for each user's gesture, 4 repetitions, each one lasting 5 seconds, were required for calibrating (fine-tuning) the model for a new user. Therefore, for a dataset consisting of 49 gestures, $49 \times 4 \times 5 = 980$ seconds of data must be collected to calibrate (fine-tune) the model for a new user. On the other hand, traditional hand-crafted methods, such as the Canonical Correlation Analysis (CCA)-based approach proposed in [36], require 3-to-5 seconds of EMG data per movement class (determined empirically) for calibration. While this calibration set is much smaller than the training set previously used [24, 29, 30, 32, 48, 83], it is developed for a smaller number of subjects. In the proposed FS-HGR approach, however, the model is tuned to a new user by seeing a small number of observations (each of duration 200 ms). For a new user in a N-way k-shot classification problem, we just need to see k-shot from each gesture. Each shot is a window of size 200 ms. For example, in 5-way 1-shot scenario, the duration of the required data from a new user for calibration (fine-tuning) is 49×200 ms $= 9.8$ seconds (which is $980/9.8 = 100$ times less than in previous methods). It is worth noting that increasing the number of shots (while improving the accuracy) increases the required number of training observations. For instance, in a 5-way 5-shot problem, the 5- shots (windows of length 200 ms) come from one repetition, i.e., for each new user only one repetition from each class, lasting 1 second, is required. In other words, we do not need to collect data for the same number of repetitions as in previous works. Therefore, the total duration of the required data from a new user would be $49 \times 5 \times 200$ ms $= 49$ seconds, which is still much lower than conventional deep learning-based approaches. In a practical setting, the number of utilized shots can be adjusted based on the required level of accuracy and training time, which provides flexibility for practical use.

The Average Prediction Accuracy for Each Class: Moreover, Table 5.10 shows the average prediction accuracy by the network for each class for 5-way, 1-shot, 5-shot, and 10-shot classification accuracies on *new repetitions with few-shot observation* for T-Block Embedding II.

Investigating the Effect of Input Modalities: The raw time-domain sEMG signals and time-channel-frequency spectrograms are the two different modalities as

Table 5.10: The average prediction accuracy by the network for each class for 5-way, 1-shot, 5-shot, and 10-shot classification accuracies on *new repetitions with few-shot observation.* In this experiment, we adopted T-Block Embedding II.

5-way k-shot Accuracy ± STD %											
Movement	**1-shot**	**5-shot**	**10-shot**	**Movement**	**1-shot**	**5-shot**	**10-shot**	**Movement**	**1-shot**	**5-shot**	**10-shot**
1	85.81 ± 0.17	93.04 ± 0.23	95.59 ± 0.26	18	67.30 ± 0.37	77.80 ± 0.45	85.05 ± 0.34	35	64.80 ± 0.44	81.15 ± 0.32	84.73 ± 0.34
2	83.15 ± 0.20	90.99 ± 0.33	94.61 ± 0.40	19	71.91 ± 0.32	80.78 ± 0.44	86.71 ± 0.37	36	69.13 ± 0.51	84.40 ± 0.44	87.83 ± 0.54
3	85.43 ± 0.17	91.59 ± 0.22	93.86 ± 0.17	20	68.74 ± 0.34	80.44 ± 0.46	85.55 ± 0.36	37	69.27 ± 0.23	85.37 ± 0.54	88.03 ± 0.36
4	79.84 ± 0.30	89.79 ± 0.56	92.63 ± 0.23	21	68.60 ± 0.29	81.46 ± 0.32	86.65 ± 0.38	38	64.11 ± 0.51	81.42 ± 0.40	86.11 ± 0.37
5	75.56 ± 0.20	86.31 ± 0.46	91.03 ± 0.35	22	68.20 ± 0.34	78.90 ± 0.62	84.73 ± 0.39	39	76.72 ± 0.22	89.06 ± 0.37	90.18 ± 0.25
6	80.26 ± 0.33	88.60 ± 0.40	92.96 ± 0.35	23	61.34 ± 0.26	74.96 ± 0.34	81.84 ± 0.45	40	76.35 ± 0.31	88.28 ± 0.32	90.66 ± 0.27
7	81.90 ± 0.22	89.96 ± 0.27	92.47 ± 0.29	24	66.87 ± 0.31	79.99 ± 0.34	84.63 ± 0.28	41	85.58 ± 0.28	94.95 ± 0.14	96.32 ± 0.21
8	79.35 ± 0.27	89.28 ± 0.23	92.41 ± 0.29	25	67.06 ± 0.49	81.08 ± 0.47	87.34 ± 0.32	42	86.35 ± 0.31	94.89 ± 0.31	95.79 ± 0.17
9	73.27 ± 0.38	83.83 ± 0.41	88.87 ± 0.17	26	65.36 ± 0.51	78.96 ± 0.71	84.24 ± 0.33	43	86.47 ± 0.18	95.11 ± 0.25	95.66 ± 0.27
10	69.24 ± 0.28	80.71 ± 0.33	88.19 ± 0.29	27	63.34 ± 0.38	77.53 ± 0.37	83.42 ± 0.31	44	89.12 ± 0.33	96.27 ± 0.15	96.99 ± 0.17
11	63.16 ± 0.42	75.48 ± 0.28	83.37 ± 0.39	28	64.82 ± 0.51	78.46 ± 0.44	83.73 ± 0.27	45	88.32 ± 0.28	95.18 ± 0.20	96.97 ± 0.26
12	71.62 ± 0.32	84.35 ± 0.36	89.23 ± 0.37	29	63.20 ± 0.60	76.74 ± 0.57	81.61 ± 0.40	46	86.28 ± 0.24	93.20 ± 0.26	96.18 ± 0.22
13	80.10 ± 0.33	90.46 ± 0.29	93.09 ± 0.30	30	64.28 ± 0.47	78.63 ± 0.63	83.83 ± 0.35	47	89.76 ± 0.16	94.11 ± 0.18	96.65 ± 0.13
14	78.00 ± 0.33	88.76 ± 0.45	91.99 ± 0.21	31	70.89 ± 0.43	83.26 ± 0.57	86.78 ± 0.33	48	91.89 ± 0.18	95.17 ± 0.28	97.01 ± 0.27
15	78.26 ± 0.28	87.87 ± 0.26	92.10 ± 0.25	32	73.32 ± 0.40	85.88 ± 0.29	87.97 ± 0.26	49	90.64 ± 0.22	93.50 ± 0.26	97.59 ± 0.13
16	75.32 ± 0.40	85.38 ± 0.50	88.12 ± 0.41	33	63.61 ± 0.51	79.58 ± 0.61	83.15 ± 0.61				
17	76.72 ± 0.25	88.90 ± 0.23	91.35 ± 0.26	34	68.91 ± 0.30	83.07 ± 0.44	87.71 ± 0.48				

Table 5.11: Parameters used in design of *Spect-Embed1*, *Raw-Embed*, and *Spect-Embed1*2. Each module repeats the following block for 3 or 4 times {3×3 conv, batch norm, ReLU, MaxPool 2D}.

Name	Input	# Blocks	# Channels in each block	Kernel-size of MaxPool in each block
Spect-Embed1	Spectrogram	4	128, 128, 128, 128	(1,3), (2,3), (2,3), (2,3)
Raw-Embed	Raw sEMG	3	8, 16, 32	(1,2), (3,4), (2,25)
Spect-Embed2	Spectrogram	3	8, 16, 32	(1,2), (3,4), (2,5)

inputs for the FS-HGR architecture. For utilizing raw sEMG as the input, we follow the pre-processes approach described in Section 5.1 and use a 1st order low-pass Butterworth filter to smooth the sEMG signals. Moreover, for scaling the sEMG signals, the *μ-law* transformation is applied to sensors with small values, amplifying their output in a logarithmic fashion (Eq. 3). This transformation keeps the scale of sensors with larger magnitudes over time. Next, the Minmax normalization is applied to the scaled inputs. It has been experimentally observed that scaling sEMG signals with the mentioned pre-processing approach leads to better output. The spectrogram is calculated using a 256-point Fast Fourier Transform (FFT) with a Hamming window and hop-length of 72. This procedure converts each segment of raw sEMG to a spectrogram with 129 frequencies covering the range ($0 - 1,000$ Hz). Then and following the procedure in [38, 83], to decrease the baseline drift, the first frequency is removed. Moreover, we removed the frequencies in range ($700 - 1,000$ Hz) because the majority of the sEMG energy does not lie in this band.

To make multidimensional input compatible with the proposed architecture, we develop the Embedding Module, which repeats the following block several times {3×3 conv, batch norm, ReLU, Max Pool 2D}. This module extracts a 128-dimensional feature vector from raw sEMG signals or corresponding spectrogram. Each feature vector is then concatenated with its corresponding label to form the input for the next stage. It is noteworthy to mention that raw sEMG signals acquired from N_S number of sensors which are segmented by a window of length of $W = 200$ ms. We adopt three Embedding Modules, namely *Spect-Embed1*, *Raw-Embed*, and *Spect-Embed1*2 as detailed in Table 5.11.

Table 5.12 shows the performance of the proposed FS-HGR when faced with new repetitions with few-shot observations on the target. Following *Experiment 1* in Section 5.3, we evaluate the FS-HGR when $\mathscr{D}_{meta-train}$ consists of the 2/3 of the

Table 5.12: 5-way, 1-shot and 5-shot classification accuracies on new repetitions with few-shot observations, performed by using meta-learning approach. The table also shows a comparison between the proposed FS-HGR Meta-supervised learning methodology and previous works where Supervised learning is used.

		Embedding Module	5-way Accuracy	
			1-shot	5-shot
Meta-Supervised	Learning	Spect-Embed1	70.71%	80.40%
		Raw-Embed	**71.23%**	**83.99%**
		Spect-Embed2	61.51%	74.54%
Supervised Learning	Previous Works	Previous Works	**Accuracy**	
		Wei *et al.* [32]	**83.70%**	
		Hu et al. [30]	82.20%	
		Ding *et al.* [48]	78.86%	
		Zhai *et al.* [83]	78.71%	
		Geng *et al.* [24]	77.80%	

Table 5.13: 5-way, 1-shot and 5-shot classification accuracies based on *new subjects with few-shot observation.*

The Embedding Module	5-way Accuracy	
	1-shot	5-shot
Spect-Embed1	52.31%	69.61%
Raw-Embed	**66.45%**	**76.39%**
Spect-Embed2	54.64%	70.77%

gesture trials of each subject, repetitions 1, 3, 4, and 6 are used for training), and $\mathscr{D}_{meta-test}$ consists of the remaining repetitions. Table 5.12 also shows the results associated with the proposed few-shot classification approach and previous supervised learning-based methodologies. From Table 5.12, it can be observed that the proposed FS-HGR architecture outperforms existing methodologies when evaluated based on the same setting, i.e., 83.99% best accuracy from the FS-HGR compared to 83.70% best accuracy achieved by the state-of-the-art.

Moreover, to classify the hand gestures of new subjects, we followed *Experiment 2* in Section 5.3 and split the DB2 database into $\mathscr{D}_{meta-train}$, $\mathscr{D}_{meta-val}$, and $\mathscr{D}_{meta-test}$ such that the subjects in these meta-sets are completely different (i.e., there is no overlap between the meta-sets). More specifically, $\mathscr{D}_{meta-train}$ consists of the first 27

Table 5.14: 5-way, 1-shot and 5-shot classification accuracies based on *new gesture with few-shot observation*.

The Embedding Module	5-way Accuracy	
	1-shot	**5-shot**
Spect-Embed1	38.35%	57.1%
Raw-Embed	**47.28%**	**72.19%**
Spect-Embed2	38.78%	58.06%

subjects, while $\mathscr{D}_{meta-val}$ includes the sEMG signals from the 28^{th} subject to 32^{ed} subject. Finally, we evaluated our model on the remaining 8 subjects. For constructing each task $\mathcal{T}_j$, we constrained ourselves to sample all of N classes from a specific user, randomly selected from the existing participants, which is a more realistic and practical scenario. Table 5.13 shows the results for the *classification on new-subject with few-shot observations* based on three Embedding Modules. The results of Table 5.13 show that the proposed FS-HGR achieved acceptable results in transferring information between a source and a target domain despite the existence of a distribution mismatch between them.

Moreover, we evaluate the capability of the proposed FS-HGR architecture when the target consists of solely out-of-sample gestures (i.e., a new gesture with few shot observations). We followed the same procedure for the *Experiment 3* in Section 5.3 here. Therefore, Exercises B and C were used for training and validation, and Exercises D with totally different gestures were used for test purposes. Table 5.14 shows the efficiency of the FS-HGR when provided with out-of-sample gestures. It can be observed that the model can predict unknown class distributions in scenarios where few examples from the target distribution is available.

As shown in Table 5.12, Table 5.13, and Table 5.14 , the network has better performance when it receives the raw sEMG signal as the input. The Raw-Embed has similar structure to that of the Spect-Embed2, and the main difference is the input format. Therefore, it can be concluded that the network itself can extract more informative features, improving the overall performance. Moreover, in Experiment 1, where $\mathscr{D}_{meta-train}$ and $\mathscr{D}_{meta-test}$ have the same distribution, the Spect-Embed1 has better performance than Spect-Embed2. The intuition is that deeper structure of Spect-Embed1 helps it to mimic data distribution better than Spect-Embed2. In contrast, as shown in Table 5.13 and Table 5.14, the performance of Spect-Embed1 drops in comparison with Spect-Embed2. The main reason is that data distribution in

$\mathscr{D}_{meta-train}$ and $\mathscr{D}_{meta-test}$ is not the same anymore. Therefore, the simpler structure in Spect-Embed2 contributes to its generalization resulting in better performance on the test-set having a different distribution from that of the train-set.

5.4 Conclusion

In this chapter, we proposed a novel few-shot learning recognition approach for the task of hand gesture recognition via sEMG signals. The proposed FS-HGR framework could quickly generalize after seeing very few observations from each class. This is achieved by exploiting the knowledge gathered from previous experiences to accelerate the learning process performed by a new subject. The experience gained over several source subjects is leveraged to reduce the training time of a new target user. The ability to learn quickly based on a few observations is a key characteristic of the proposed FS-HGR framework that distinguishes this novel architecture from its previous counterparts. A second contribution of the chapter is its capability to address the user-dependent nature of the sEMG signals. The proposed FS-HGR framework transfers information between a source and a target domain despite the existence of a distribution mismatch among them. This would dramatically reduce the number of required cumbersome training sessions leading to a drastic reduction in functional prosthesis abandonment. In this chapter, we have shown that for a new user/gesture when the distributions of sEMG signals is different from that of the training data, the model can still classify the hand movements. This is because the knowledge gained during the training phase is leveraged for the new users/gestures. Other factors such as time variability between days, and type of amputations affect the distributions of sEMG signals. These factors were not investigated in the proposed architecture while are relevant to study in future research. A third factor that can affect distributions of sEMG signals is misplacement or displacement of sensors (electrode locations/shift), which is an open topic of research that has not been addressed in this chapter. This can be applied to any existing research focusing on the processing of sEMG. We believe that the proposed approach has the potential to also address this problem, however, we have not completed our experiments and thus cannot strongly mention that. We consider this as a limitation for our current study and a future research direction. As a final note, we would like to mention that multi-channel EMG recording has become a common trend and there are commercialized wearable sensors. It is correct that the higher number of sensors results in higher complexity of electronics, but thanks to recent advances in the area of wearable sensors, this has been achieved and is progressing.

Chapter 6

Conclusion and Future Direction

While passive prostheses are typically used for aesthetic purposes, active prostheses can perform different functionalities such as gripping, grasping, and functional movements, to name but a few. Active prostheses can be categorized into: (i) *Body-Powered Prostheses*, in which the opening and closing are achieved by a harness and cables that are worn over the user's shoulder, and; (ii) *Electrically-Powered Prostheses*, which are controlled by neural information sources and are referred to as myoelectric prostheses. In the latter case, once an amputee wants to move their phantom limb, the intent is first encoded in a nerve impulse, which is communicated naturally from the brain to the residual muscles, producing electrical activities. Myoelectric control is performed by measuring these electrical activities and actuating a prosthesis device to provide natural and intuitive control for the user. The main focus throughout this Ph.D. thesis was on active myoelectric prostheses. We first presented an review of different myoelectric pattern-recognition control algorithms, especially DNN-based approaches. Then we focused on three important research objectives associated with HGR, i.e., improving overall accuracy of sparse sEMG-based HGR, reducing complexity of DNN architectures, and developing adaptive learning solutions. In the following, we first summarize the thesis contributions, and then discuss potential directions for future research.

6.1 Summary of Contributions

The thesis contributions can be summarized as follows:

- **Accurate and Efficient sEMG-based Deep Learning Models:** In Chapter 3, we discussed one of the most important HGR research objectives, which is improving the accuracy of myoelectric systems, and proposed the HRM architecture. Since this network is developed based on a hybrid deep learning technique, it outperforms CNNs in the HGR problem. We expanded on this research by presenting TCN architecture developed based dilated causal convolutions. The proposed TCN architecture is capable of processing the input sequence as whole, rather than sequentially as in RNNs. Moreover, unlike LSTMs that can easily consume a lot of memory to store partial results, in the TCN architecture, the filters are shared in a layer resulting in low memory requirements for training. Our results show that the two proposed approaches are capable of increasing the recognition accuracy when compared to the existing solutions. Then, we moved on to the second research goal of myoelectric control, which is to design an efficient DNN-based architecture for the HGR problem. In this context, we presented the XceptionTime architecture, which incorporates depthwise separable convolutions, adaptive average pooling, and a novel non-linear normalization technique to classify the hand gestures. The proposed XceptionTime architecture has reduced number of trainable parameters resulting in a more compact (reduced complexity) and efficient solution.

- **Attention-Based Models:** Capitalizing on the fact that attention-based models have revolutionized several ML fields, including NLP, Computer Vision, and speech recognition, we demonstrated that attentions-based architectures have the potential to improve the analysis of sEMG signals and bridge the gap between recent academic research and clinical settings. In this regard, in Chapter 4, we focused on increasing the accuracy and reducing the number of parameters for the HGR problem by utilizing attention-based architectures. We presented TC-HGR architecture, which is based on self-attention mechanism and temporal convolutions to classify the hand gestures. Then, we proposed the TEMGNet architecture based on transformer. In particular, the TEMGNet architecture is entirely based on attention mechanisms, with no recurrence or convolutions. We demonstrated that the TEMGNet architecture outperforms its existing state-of-the-art counterparts in terms of overall recognition accuracy and complexity.

- **Adaptive-based Model:** One important, and at the same time, challenging problem, within the hand gesture classification scope, is developing a novel framework based on the formulation of Few-Shot Learning to infer the required output given only one or a few number of training examples. Therefore, in Chapter 5, we proposed an architecture (refered to as the FS-HGR) that learns such a mapping using a small number of data and quickly adapts to a new user/gesture by combing its prior experience. The proposed FS-HGR architecture's ability to learn quickly based on a few observations is a key feature that distinguishes this novel architecture from its predecessors.

6.2 Future Direction

Below, we present few fruitful directions for future research:

- Factors such as time variability between days and type of amputations affect the distributions of sEMG signals. These factors were not investigated in the thesis while are relevant to study in future research. A third factor that can affect distributions of sEMG signals is misplacement or displacement of sensors, which is an open topic of research that has not been addressed in this thesis. We consider this as a limitation for our current study and a future research direction.

- Most of the literature concentrate on algorithm development based on intact (healthy) subjects. No or few amputees were used to validate some approaches [94]. Pattern recognition for prosthesis control is inherently dependent on the unique and repeatable patterns of sEMG signals. It is, therefore, critical to investigate the effects of different levels of amputation on the capabilities of DNN-based solutions.

- There is a need to develop advanced prostheses for people who have a high level of amputation. More specifically, there are more solutions available for the transradial prostheses than the transhumeral ones, although both make up a similar percentage of the population [94]. This can be attributed to the fact that it is easier to restore hand function for the lower level of amputation since more muscles are available. Therefore, there is an unmet need to improve myoelectric prostheses control for transhumeral amputees. Moreover, most of available databases provide sEMG signals for intact subjects. Only a few ones have collected signals from transradial amputated patients and none exist for transhumeral patients. Consequently, to develop robust pattern recognition-based control algorithms for myoelectric prostheses, there is a need for public databases, which focus on amputee subjects, especially the transhumeral subjects [14, 94].

- Enrichment of sEMG signals with other measurement modalities can be considered a fruitful direction for future research. For example, sensor fusion techniques such as coupling sEMG electrodes with Inertial Measurements (IMs) can increase classification performance [32]. Moreover, it was shown [95] that replacing the EMG signals with IM information rather than combining the two, has the potential to increase the classification accuracy for some specific gestures. These findings are important, combining sEMG signals with IMs can help researchers resolve the non-stationarity problems with the sEMG signals. On the other hand, the combination of computer vision and sEMG signals is one of the most recent developments in this field and offers promising prospects for hand movement classification and future study. More specifically, the performance of sEMG-based hand gesture recognition is negatively impacted by the changes in arm posture or muscle fatigue. On the other hand, by utilizing vision sensors, a significant stream of environmental data can be provided, which might be crucial in determining the intended gestures. However, visual data can be affected by its own artifacts, including changes in lighting. Therefore, combining sEMG signals with visual sensors is a promising potential for future study given the complimentary properties of these modalities.

- While DNN-based approaches have shown promising recognition results for myoelectric control, such data-driven solutions, typically, fail to provide sensory feedback to the limb amputee end users. In other words, hand motions combined with a sense of touch can enhance the smoothness of human interactions with the environment. It is, therefore, critical to recover the lost sensory touch in the upper limb prosthesis through sensory neuro-prosthesis, enhancing the dexterous control of myoelectric prosthesis. Capturing the human hand's sensory perception can be achieved by electronic skin (e-skin), which has embedded sensors providing sensory precepts (e.g., touch, temperature, pressure, and pain) from the environment. Consequently, several recent works such as [96] have focused on providing a sensation feedback loop such that amputees can interact with their environment more easily. This is another fruitful direction for future research.

- Another possible research direction to improve myoelectric prostheses is real-time ML, where prosthetic devices are considered as goal-seeking agents. Following an autonomous system perspective, agents can adapt to the environmental changes. For instance, incorporation of Reinforcement Learning (RL)-based methods can allow an amputee interacting with the prosthetics agent, which in turn learns, over time, how to perform different patterns. Use of RL-based models has the potential of providing personalized myoelectric prosthesis control.

- When designing myoelectric prosthesis, restoring functionality of a hand is not the only important factor. Other concerns such as cost, comfort, cosmetic acceptance, intuitiveness, stability, and durability are important to fulfill the amputated user's needs [14].

Bibliography

[1] E. Rahimian, S. Zabihi, A. Asif, D. Farina, S.F. Atashzar, and A. Mohammadi, "FS-HGR: Few-shot Learning for Hand Gesture Recognition via ElectroMyography," *IEEE Trans. Neural Syst. Rehabil. Eng.*, 2021.

[2] E. Rahimian, S. Zabihi, S. F. Atashzar, A. Asif, and A. Mohammadi, "Surface EMG-Based Hand Gesture Recognition via Hybrid and Dilated Deep Neural Network Architectures for Neurorobotic Prostheses," *Journal of Medical Robotics Research*, 2020, pp. 1-12.

[3] E. Rahimian, S Zabihi, A Asif, D. Farina, S.F. Atashzar, A Mohammadi, "TEMGNet: Deep Transformer-based Decoding of Upperlimb sEMG for Hand Gestures Recognition", *arXiv:2109.12379*, 2021.

[4] E. Rahimian, S. Zabihi, A. Asif, D. Farina, S.F. Atashzar, and A. Mohammadi, "Hand Gesture Recognition Using Temporal Convolutions and Attention Mechanism," *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2022, pp. 1196-1200.

[5] E. Rahimian, S. Zabihi, A. Asif, S.F. Atashzar, and A. Mohammadi, "Trustworthy Adaptation with Few-Shot Learning for Hand Gesture Recognition," *IEEE International Conference on Autonomous Systems (ICAS)*, 2021, pp. 1-5.

[6] E. Rahimian, S. Zabihi, A. Asif, S.F. Atashzar, and A. Mohammadi, "Few-Shot Learning for Decoding Surface Electromyography for Hand Gesture Recognition," *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2021, pp. 1300-1304.

[7] E. Rahimian, S. Zabihi, A. Asif, and A. Mohammadi, "Hybrid Deep Neural Networks for Sparse Surface EMG-Based Hand Gesture Recognition," *Asilomar Conference on Signals, Systems, and Computers*, 2020, pp. 371-374.

[8] E. Rahimian, S. Zabihi, S. F. Atashzar, A. Asif, A. Mohammadi, "sEMG-Based Hand Gesture Recognition via Dilated Convolutional Neural Networks," *GlobalSIP*, 2019.

[9] E. Rahimian, S. Zabihi, F. Atashzar, A. Asif, A. Mohammadi, "XceptionTime: Independent Time-Window XceptionTime Architecture for Hand Gesture Classification," *International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, 2020.

[10] F.V. Tenore, A. Ramos, A. Fahmy, S. Acharya, R. Etienne-Cummings, and N.V. Thakor, "Decoding of Individuated Finger Movements Using Surface Electromyograph," *IEEE Transactions on Biomedical Engineering*, vol. 56, no. 5, pp. 1427-1434, 2009.

[11] N. Jiang, D. Falla, A. D'Avella, B. Graimann, and D. Farina, "Myoelectric Control in Neurorehabilitation," *Critical Reviews in Biomedical Engineering*, vol. 38, no. 4, pp. 381-391, 2010.

[12] J.L. Betthauser, C.L. Hunt, L.E. Osborn, M.R. Masters, G. Levay, R.R. Kaliki, and N.V. Thakor, "Limb Position Tolerant Pattern Recognition for Myoelectric Prosthesis Control with Adaptive Sparse Representations From Extreme Learning," *IEEE Transactions on Biomedical Engineering*, vol. 65, no. 4, pp. 770-778, 2017.

[13] D. Farina, N. Jiang, H. Rehbaum, A. Holobar, B. Graimann, H. Dietl, and O.C. Aszmann, "The Extraction of Neural Information from the Surface EMG for the Control of Upper-Limb Prostheses: Emerging Avenues and Challenges," *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, vol. 22, no. 4, pp. 797-809, 2014.

[14] N. Parajuli, N. Sreenivasan, P. Bifulco, M. Cesarelli, S. Savino, V. Niola, D. Esposito, T.J. Hamilton, G.R. Naik, U. Gunawardana, and G.D. Gargiulo, "Real-time EMG based Pattern Recognition Control for Hand Prostheses: A Review on Existing Methods, Challenges and Future Implementation," *Sensors*, vol. 19, no. 20, p.4596, 2019.

[15] M. Ortiz-Catalan, R. Branemark, and B. Hakansson, "A Modular Research Platform for the Control of Artificial Limbs based on Pattern Recognition Algorithms," *Source code for biology and medicine*, vol. 8, no. 1, p.11, 2013.

[16] M. Atzori, A. Gijsberts, C. Castellini, B. Caputo, A.G.M. Hager, S. Elsig, G. Giatsidis, F. Bassetto, and H. Muller, "Electromyography Data for Non-Invasive Naturally-controlled Robotic Hand Prostheses," *Scientific data*, vol. 1, no. 1, pp.1-13, 2014.

[17] M. Atzori, A. Gijsberts, I. Kuzborskij, S. Heynen, A.G.M Hager, O. Deriaz, C. Castellini, H. MÃijller, and B. Caputo, "A Benchmark Database for Myoelectric Movement Classification," *Transactions on Neural Systems and Rehabilitation Engineering*, 2013.

[18] A. Krasoulis, I. Kyranou, M.S. Erden, K. Nazarpour, and S. Vijayakumar, "Improved Prosthetic Hand Control with Concurrent Use of Myoelectric and Inertial Measurements," *neuroengineering and rehabilitation*, vol. 14, no. 1, p.71, 2017.

[19] S. Pizzolato, L. Tagliapietra, M. Cognolato, M. Reggiani, H. Muller, and M. Atzori, "Comparison of Six Electromyography Acquisition Setups on Hand Movement Classification Tasks," *PloS one*, vol. 12, no. 10, 2017.

[20] C. Amma, T. Krings, J. Boer, and T. Schultz, "Advancing Muscle-computer Interfaces with High-density Electromyography," *In Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems*, pp. 929-938, 2015.

[21] F. Palermo, M. Cognolato, A. Gijsberts, H. MÃijller, B. Caputo, and M. Atzori, "Repeatability of Grasp Recognition for Robotic Hand Prosthesis Control based on sEMG Data," *International Conference on Rehabilitation Robotics*, pp.1154-1159, 2017.

[22] A. Krasoulis, S. Vijayakumar, and K. Nazarpour, "Effect of User Adaptation on Prosthetic Finger Control with an Intuitive Myoelectric Decoder," *Frontiers in Neuroscienc*, vol. 13, p.891, 2019.

[23] M. Cognolato, A. Gijsberts, V. Gregori, G. Saetta, K. Giacomino, A.G.M. Hager, A. Gigli, D. Faccio, C. Tiengo, F. Bassetto, and B. Caputo, "Gaze, Visual, Myoelectric, and Inertial Data of Grasps for Intelligent Prosthetics," *Scientific Data*, vol. 7, no. 1, pp.1-15, 2020.

[24] W. Geng, Y. Du, W. Jin, W. Wei, Y. Hu, and J. Li, "Gesture Recognition by Instantaneous Surface EMG Images," *Scientific reports*, p.36571, 2016.

[25] Y. Du, W. Jin, W. Wei, Y. Hu, and W. Geng, "Surface EMG-based Inter-session Gesture Recognition Enhanced by Deep Domain Adaptation," *Sensors*, vol. 17, no. 3, p.458, 2017.

[26] A. Matran-Fernandez, I.J.R. Martinez, R. Poli, C. Cipriani, and L. Citi, "SEEDS, Simultaneous Recordings of High-density EMG and Finger Joint Angles During Multiple Hand Movements," *Scientific data*, vol. 6, no. 1, pp.1-10, 2019.

[27] A. Ameri, M.A. Akhaee, E. Scheme, and K. Englehart, "Regression Convolutional Neural Network for Improved Simultaneous EMG Control," *Journal of Neural Engineering*, vol. 16, no. 3, p.036015, 2019.

[28] C. Li, J. Ren, H. Huang, B. Wang,Y. Zhu, and H. Hu, "PCA and Deep Learning based Myoelectric Grasping Control of a Prosthetic Hand," *Biomedical engineering online*, vol. 17, no. 1, p.107, 2018.

[29] M. Atzori, M. Cognolato, and H. Muller, "Deep Learning with Convolutional Neural Networks Applied to Electromyography data: A Resource for the Classification of Movements for Prosthetic Hands," *Frontiers in neurorobotics*, vol. 10, no. 9, 2016.

[30] Y. Hu, Y. Wong, W. Wei, Y. Du, M. Kankanhalli, and W. Geng, "A Novel Attention-based Hybrid CNN-RNN Architecture for sEMG-based Gesture Recognition," *PloS one*, vol. 13, no. 10, 2018.

[31] W. Wei, Y.Wong, Y. Du, Y. Hu, M. Kankanhalli, and W. Geng, "A Multi-stream Convolutional Neural Network for sEMG-based Gesture Recognition in Muscle-computer Interface," *Pattern Recognition Letters*, pp.131-138, 2019.

[32] W. Wei, Q. Dai, Y. Wong, Y. Hu, M. Kankanhalli, and W. Geng, "Surface Electromyography-based Gesture Recognition by Multi-view Deep Learning," *IEEE Transactions on Biomedical Engineering*, vol. 66, no. 10, pp.2964-2973, 2019.

[33] X. Zhang, L. Wu, B. Yu, X. Chen, and X. Chen, "Adaptive Calibration of Electrode Array Shifts Enables Robust Myoelectric Control," *IEEE Transactions on Biomedical Engineering*, 2019.

[34] A. Ameri, M.A. Akhaee, E. Scheme, and K. Englehart, "A Deep Transfer Learning Approach to Reducing the Effect of Electrode Shift in EMG Pattern Recognition-Based Control," *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, vol. 28, no. 2, pp.370-379, 2019.

[35] J.L. Betthauser, J.T. Krall, S.G. Bannowsky, G. Levay, R.R. Kaliki, M.S. Fifer, and N.V. Thakor, "Stable Responsive EMG Sequence Prediction and Adaptive Reinforcement With Temporal Convolutional Networks," *IEEE Transactions on Biomedical Engineering*, vol. 67, no. 6, pp.1707-1717, 2019.

[36] R.N. Khushaba, "Correlation Analysis of Electromyogram Signals for Multiuser Myoelectric Interfaces," *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, vol. 22, no. 4, pp.745-755, 2014.

[37] T. Matsubara and J. Morimoto, "Bilinear Modeling of EMG Signals to Extract User-Independent Features for Multiuser Myoelectric Interface," *IEEE Transactions on Biomedical Engineering*, vol. 60, no. 8, pp. 2205-2213, 2013.

[38] U. Cote-Allard, C.L. Fall, A. Drouin, A. Campeau-Lecours, C. Gosselin, K. Glette, F. Laviolette, and B. Gosselin, "Deep learning for Electromyographic Hand Gesture Signal Classification Using Transfer Learning," *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, vol. 27, no. 4, pp.760-771, 2019.

[39] A. Olsson, N. Malesevic, A. Bjorkman and C. Antfolk, "Exploiting the Intertemporal Structure of the Upper-Limb sEMG: Comparisons between an LSTM Network and Cross-Sectional Myoelectric Pattern Recognition Methods," *Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*, pp.6611-6615, 2019.

[40] B. Hudgins, P. Parker, and R.N. Scott, "A New Strategy for Multifunction Myoelectric Control", *IEEE Transactions on Biomedical Engineering*, **40**(1) (1993) pp. 82–94.

[41] W. Jiang, and Z. Yin, "Human Activity Recognition using Wearable Sensors by Deep Convolutional Neural Networks", *ACM international Conference on Multimedia*, pp. 1307–1310, 2015.

[42] K. He, X. Zhang, S. Ren, and J. Sun, "Deep Residual Learning for Image Recognition", *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 770–778, 2016.

[43] S. Bai, J.Z. Kolter, and V. Koltun, "An Empirical Evaluation of Generic Convolutional and Recurrent Networks for Sequence Modeling", *arXiv preprint arXiv:1803.01271*, 2018.

[44] A.V.D. Oord and *et al.*, "Wavenet: A Generative Model for Raw Audio", *ArXiv preprint arXiv:1609.03499*, 2016.

[45] T. Salimans and D. P. Kingma, “Weight Normalization: A Simple Re-parameterization to Accelerate Training of Deep Neural Networks”, *In Advances in Neural Information Processing Systems*, pp. 901–909, 2016.

[46] D. A. Clevert, T. Unterthiner, and S. Hochreiter, “Fast and Accurate Deep Network Learning by Exponential Linear Units (ELUs)”,*arXiv preprint arXiv:1511.07289, 2015.*

[47] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, “Dropout: A Simple Way to Prevent Neural Networks from Overfitting”, *The Journal of Machine Learning Research*, **15**(1) (2014) pp. 1929–1958.

[48] Z. Ding, C. Yang, Z. Tian, C. Yi, Y. Fu, , and F. Jiang, “sEMG-Based Gesture Recognition with Convolution Neural Networks”, *Sustainability* **10**(6) (2018) p.1865.

[49] M. Zia ur Rehman, A. Waris, S. Gilani, M. Jochumsen, I. Niazi, M. Jamil, D. Farina, and E. Kamavuako, “Multiday EMG-based Classification of Hand Motions with Deep Learning Techniques,” *Sensors*, vol. 18, no. 8, p. 2497, 2018.

[50] TU-T. Recommendation G. 711 “Pulse Code Modulation (PCM) of Voice Frequencies,” *ITU*, 1988.

[51] P. Tsinganos, B. Cornelis, J. Cornelis, B. Jansen, and A. Skodras, “Deep Learning in EMG-based Gesture Recognition,” *PhyCS*, pp. 107-114, 2018.

[52] P. Tsinganos, B. Cornelis, J. Cornelis, B. Jansen, A. and Skodras, “A Hilbert Curve Based Representation of sEMG Signals for Gesture Recognition,” *International Conference on Systems, Signals and Image Processing (IWSSIP)*, 201-206, 2019.

[53] P. Tsinganos, B. Cornelis, J. Cornelis, B, Jansen, and A. Skodras, “Improved Gesture Recognition Based on sEMG Signals and TCN,” *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2019, pp. 1169-1173.

[54] H. I. Fawaz, B. Lucas, G. Forestier, C. Pelletier, D. F. Schmidt, J. Weber, G. I. Webb, L. Idoumghar, P. Muller, F. Petitjean, “InceptionTime: Finding AlexNet for Time Series Classification,” *arXiv:1909.04939*, 2019.

[55] F. Chollet, “Xception: Deep Learning with Depthwise Separable Convolutions,” *In Proceedings of the IEEE conference on computer vision and pattern recognition* , pp.1251-1258, 2017.

[56] M. Lin, Q. Chen, and S. Yan, “Network in Network,” *arXiv preprint arXiv:1312.4400*, 2013.

[57] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, “Going Deeper with Convolutions,” *In Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 1-9.

[58] L. Sifre, S. and Mallat, “Rigid-motion Scattering for Image Classification,” *Ph. D. dissertation*, 2014.

[59] V. Vanhoucke, "Learning Visual Representations at Scale," *ICLR invited talk*, 2014.

[60] N. Jiang, S. Dosen, K.R. Muller, D. Farina, "Myoelectric Control of Artificial Limbs. Is There A Need to Change Focus?" *IEEE Signal Processing Magazine*, vol. 29, pp. 150-152, 2012.

[61] D. Farina, R. Merletti, R.M. Enoka, "The Extraction of Neural Strategies from the Surface EMG," *Journal of Applied Physiology*, vol. 96, pp. 1486-95, 2004.

[62] C. Castellini, *et al.*, "Proceedings of the First Workshop on Peripheral Machine Interfaces: Going Beyond Traditional Surface Electromyography," *Frontiers in neurorobotics*, 8, p. 22, 2014.

[63] T. Sun, Q. Hu, P. Gulati, and S.F. Atashzar, "Temporal Dilation of Deep LSTM for Agile Decoding of sEMG: Application in Prediction of Upper-limb Motor Intention in NeuroRobotics.," *IEEE Robotics and Automation Letters*, 2021.

[64] A. Dosovitskiy, et al., "An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale," *arXiv preprint arXiv:2010.11929*, 2020.

[65] A. Vaswani, N. Shazeer, J. Uszkoreit, L. Jones, A. Gomez N., L. Kaiser, and I. Polosukhin, "Attention is All You Need," *arXiv preprint arXiv:1706.03762*, 2017a.

[66] N. Mishra, M. Rohaninejad, X. Chen, and P. Abbeel, "A Simple Neural Attentive Meta-Learner," *arXiv preprint arXiv:1707.03141*, 2017.

[67] F. Wilcoxon, "Individual Comparisons by Ranking Methods," *Biometrics Bull.*,vol. 1, no. 6, pp. 80-83, 1945.

[68] McIntosh, et al., "EMPress: Practical Hand Gesture Classification with Wrist-Mounted EMG and Pressure Sensing," *In Proceedings of the CHI Conference on Human Factors in Computing Systems*, pp. 2332-2342, 2016.

[69] X. Si, T. Li, Q. Zhang, X. Hu., "An Optimal Condition-based Replacement Method for Systems with Observed Degradation Signals," *IEEE Trans. Reliab.*, vol. 67, no. 3, pp. 1281-1293, 2018.

[70] F. Quivira, *et al.*, "Translating sEMG Signals to Continuous Hand Poses Using Recurrent Neural Networks.," *in Proc. IEEE EMBS Int. Conf. Biomed. Health Informat.*, 2018, pp. 166-169.

[71] J. Devlin, M.W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of Deep Bidirectional Transformers for Language Understanding," *arXiv preprint arXiv:1810.04805*, 2018.

[72] T.B. Brown., *et al.*, "Language Models are Few-shot Learners," *arXiv preprint arXiv:2005.14165*, 2020.

[73] G. Krishna, C. Tran, M. Carnahan, and A.H. Tewfik, "EEG based Continuous Speech Recognition using Transformers," *arXiv preprint arXiv:2001.00501*, 2019.

[74] Y. Song, X. Jia, L. Yang, and L. Xie, "Transformer-based Spatial-Temporal Feature Learning for EEG Decoding," *arXiv preprint arXiv:2106.11170*, 2021.

[75] B. Wang, C. Liu, C. Hu, X. Liu, and J. Cao, "Arrhythmia Classification with Heartbeat-Aware Transformer," *International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2021, pp. 1025-1029.

[76] R. Casal, L.E. Di Persia, and G. Schlotthauer, "Temporal Convolutional Networks and Transformers for Classifying the Sleep Stage in Awake or Asleep using Pulse Oximetry Signals," *arXiv preprint arXiv:2102.03352*, 2021.

[77] Y. Wang, *et al.*, "Transformer-based Acoustic Modeling for Hybrid Speech Recognition," *International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2020, pp. 6874-6878.

[78] J. Guan, W. Wang, P. Feng, X. Wang, and W. Wang, "Low-Dimensional Denoising Embedding Transformer for ECG Classification," *International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2021, pp. 1285-1289.

[79] JL Ba, JR Kiros, and G.E. Hinton, "Layer Normalization," *arXiv preprint arXiv:1607.06450*, 2016.

[80] DP. Kingma, and J. Ba, "Adam: A Method for Stochastic Optimization," *ICLR*, 2015.

[81] A. Gijsberts, M. Atzori, C. Castellini, H. Muller, and B. Caputo, "Movement Error Rate for Evaluation of Machine Learning Methods for sEMG-based Hand Movement Classification," *IEEE Trans. Neural Syst. Rehabil. Eng.*, vol. 22, no. 4, pp. 735-744, 2014.

[82] L. Chen, J. Fu, Y. Wu, H. Li, and B. Zheng, "Hand Gesture Recognition using Compact CNN via Surface Electromyography Signals," *Sensors*, vol. 20, no.3, p. 672, 2020.

[83] X. Zhai, B. Jelfs, R. H. Chan, and C. Tin, "Self-recalibrating Surface EMG Pattern Recognition for Neuroprosthesis Control based on Convolutional Neural Network," *Frontiers in neuroscience*, 11, p.379, 2017.

[84] S. Ravi, and H. Larochelle, "Optimization as a Model for Few-shot Learning," *International Conference on Learning Representations (ICLR)*, 2016.

[85] C. Finn, P. Abbeel, and S. Levine, "Model-agnostic Meta-learning for Fast Adaptation of Deep Networks," *In Proceedings of the 34th International Conference on Machine Learning-Volume 70*, vol. 100, pp. 1126-1135, 2017.

[86] R. Chattopadhyay, N. C. Krishnan, and S. Panchanathan, "Topology Preserving Domain Adaptation for Addressing Subject based Variability in SEMG Signal," *in Proc. AAAI Spring Symp., Comput. Physiol.*, 2011, pp. 4-9.

[87] U. Cote-Allard, C. L. Fall, A. Campeau-Lecours, C. Gosselin, F. Laviolette, and B. Gosselin, "Transfer learning for SEMG Hand Gestures Recognition using Convolutional Neural Networks," *Proc. IEEE Int. Conf. Syst., Man, Cybern.*, Oct. 2017, pp. 1663-1668.

[88] A. K. Clarke *et al.*, "Deep Learning for Robust Decomposition of High-Density Surface EMG Signals," *IEEE Transactions on Biomedical Engineering*, 2020, In Press.

[89] D. Josephs, C. Drake, A. Heroy, and J. Santerre, "sEMG Gesture Recognition with a Simple Model of Attention," *arXiv preprint arXiv:2006.03645*, 2020.

[90] Y. Peng, H. Tao, W. Li, H. Yuan and T. Li, "Dynamic Gesture Recognition based on Feature Fusion Network and Variant ConvLSTM," *IET Image Processing*, vol. 14, no. 11, pp. 2480-2486, 2020.

[91] K. Englehart and B. Hudgins, "A Robust, Real-time Control Scheme for Multifunction Myoelectric Control" *IEEE Trans. Biomed. Eng.*, vol. 50, no. 7, pp. 848-854, 2003.

[92] R. N. Khushaba and S. Kodagoda, "Electromyogram (EMG) Feature Reduction Using Mutual Components Analysis for Multifunction Prosthetic Fingers Control" *in Proc. 12th Int. Conf.*

[93] A. Phinyomark, *et al.*, "EMG Feature Eevaluation for Improving Myoelectric Pattern Recognition Robustness" *Expert Syst. Appl.*, vol. 40, no. 12, pp. 4832-4840, 2013.

[94] O. Barron, M. Raison, and S. Achiche, "Control of Transhumeral Prostheses based on Electromyography Pattern Recognition: From Amputees to Deep Learning," *Academic Press*, p. 1-21, 2020.

[95] R. N. Khushaba, A. Krasoulis, A. Al-Jumaily, and K. Nazarpour, "Spatio-Temporal Inertial Measurements Feature Extraction Improves Hand Movement Pattern Recognition without Electromyography," *International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*, 2018.

[96] M.M. Iskarous, and N.V. Thakor, "E-Skins: Biomimetic Sensing and Encoding for Upper Limb Prostheses," *Proceedings of the IEEE*, vol. 107, no. 10, pp. 2052-2064, 2019.