POLITECNICO DI MILANO
Master of Science in Biomedical Engineering
Department of Electronic, Information and Bioengineering

Classification of EMG signals for hand movement intention detection

NearLab

Supervisor: Prof. Alessandra Pedrocchi
Co-supervisor: Marta Gandolla, PhD.

Thesis of:
Rahil Soroushmojdehi, ID: 903720
Sina Javadzadeh No, ID: 903669

Academic year 2018-2019

Acknowledgment

We would like to thank professor Pedrocchi for her kindness and support.
We would like to thank Dr. Gandolla for her guidance and availability, despite the maternity leave.
Finally, we want to thank our dear friends and all the students who participated in the experimental sessions.

Abstract

For people with motor disabilities or amputations, natural control of assistive devices or prostheses have significant importance. Over the past several years, Electromyography (EMG) signals have been used as a natural interface to control artificial limbs. While decoding the subject's movement intention from EMG signals using traditional machine learning methods have shown promising results. Recently, deep learning algorithms such as Convolutional Neural Networks (CNNs) have gained interest as decoding strategies and have shown superior performance and robustness in comparison with traditional machine learning methods. However, CNNs require a big dataset to train properly. Creating such a database for a single subject could be very time consuming. Hence, researchers have proposed Transfer Learning as a solution for this challenge.

In this thesis, we introduce multiple CNN-based architectures for hand movement intention detection and compare their performance with classical machine learning algorithms such as support vector machines, multi-layer perceptron, linear discriminant analysis and k-nearest neighbor. Classifiers are tested on Nearlab dataset which is a sEMG hand/wrist movement dataset created in this research work, and also on publicly available sEMG dataset “NinaPro”. Moreover, we propose two transfer learning approaches namely “Freeze & fine-tune” and “Parallel networks“, to eliminate the need to acquire large datasets from a single subject, leveraging available data from other subjects. Finally the results obtained over Nearlab dataset, indicated that deep learning algorithms can produce higher classification accuracy comparing to classical machine learning algorithms, with the maximum accuracy of 93.24% achieved by one of the proposed CNNs. In addition, transfer learning algorithm referred to as “Parallel networks“ was able to improve the average accuracy of the best performing deep learning network, by obtaining an average classification accuracy of 93.48%.

Astratto

Per le persone con disabilità motorie o amputazioni, il controllo naturale dei dispositivi di assistenza o delle protesi ha un'importanza significativa. Negli ultimi anni, i segnali dell'elettromiografia (EMG) sono stati utilizzati come interfaccia naturale per controllare gli arti artificiali. Mentre decodifica l'intenzione di movimento del soggetto dai segnali EMG usando i metodi tradizionali di apprendimento automatico hanno mostrato risultati promettenti. Recentemente, algoritmi di deep learning come Convolutional Neural Networks (CNNs) hanno guadagnato interesse come strategie di decodifica e hanno mostrato prestazioni e robustezza superiori rispetto ai metodi di apprendimento automatico tradizionali. Tuttavia, le CNN richiedono un grande set di dati per allenarsi correttamente. La creazione di un database di questo tipo per un singolo argomento potrebbe richiedere molto tempo. Pertanto, i ricercatori hanno proposto l'apprendimento del trasferimento come soluzione per questa sfida.
In questo tesi, introduciamo più architetture basate sulla CNN per il rilevamento dell'intenzione di movimento della mano e confrontiamo le loro prestazioni con gli algoritmi di apprendimento automatico classici come macchine vettoriali di supporto, percettrone multistrato, analisi discriminante lineare e vicino k-più vicino. I classificatori sono testati sul set di dati Nearlab che è un set di dati di movimento mano / polso sEMG creato in questo lavoro di ricerca, e anche sul set di dati sEMG "NinaPro" disponibile pubblicamente. Inoltre, proponiamo due approcci per l'apprendimento del trasferimento, vale a dire "Congela e perfeziona" e "Reti parallele", per eliminare la necessità di acquisire grandi set di dati da una singola materia, sfruttando i dati disponibili da altre materie. Infine, i risultati ottenuti sul set di dati Nearlab, hanno indicato che gli algoritmi di deep learning possono produrre una maggiore precisione di classificazione rispetto agli algoritmi di machine learning classici, con una precisione massima del 93,24% raggiunta da una delle CNN proposte. Inoltre, l'algoritmo di apprendimento di trasferimento denominato "Reti parallele" è stato in grado di migliorare la precisione media della rete di apprendimento profondo con le migliori prestazioni, ottenendo una precisione di classificazione media del 93,48%.

Summary

Aim - The aim of this thesis is to work towards an accurate and robust hand gesture recognition algorithm for myo-controlled upper limb assistive devices. In addition, the proposed algorithm should be practical and feasible for real life applications.
To this aim, the contribution of this work is twofold: I) develop novel deep network architectures specific for target application and II) use different transfer learning approaches with limited amount of data available for each participant. Transfer learning is employed as a possible solution to the problem of large databases necessary for training deep networks.
In this thesis, 4 deep learning networks and 2 transfer learning methods are introduced. The proposed approaches are tested on a hand/wrist movement surface electromyography (sEMG) database created by authors of this work (referred to as Nearlab dataset) and on a publicly available EMG dataset (Ninapro database 2) to provide results comparable with the state of art. A comparison has also been carried out with well-known classical hand gesture recognition algorithms.
In the end, a program simulating online classification has been designed to display the real-time implementation of a hand gesture recognition algorithm in a myo-controlled prosthesis.

Background - For people with upper limb disabilities, independently performing daily tasks that require hand function such as holding objects, opening/closing doors and eating meals is a major challenge. For this population, the use of an assistive device targeting in particular the hand could be beneficial. According to the type of disability, this device can be a prosthesis (in the case of replacing a missing limb) or an orthosis (in the case of supporting an existing limb). Among different kinds of hand prosthesis, myo-controlled hand prosthesis has gained rising interest among researchers. Myo-controlled technique uses signals acquired from residual limb muscles to control the assistive device [1]. In myo-controlled hand prostheses, the signals acquired from users' muscles is classified to predict hand movement intention. Then the predicted movement will be used to control the artificial hand. Although myo-controlled devices have been introduced for many years, due to their insufficient classification accuracy and robustness, they have not yet been widely accepted by a considerable portion of the targeted population [2]. The number and variety of hand gestures as well as the complex anatomical configuration of muscles in the forearm are the elements that make the classification task particularly challenging.
Traditionally, EMG signals picked up by electrodes placed on the surface of the skin (sEMG) were used for pattern recognition of myo-controlled hand prostheses. SEMG signals were pre-processed and segmented into windows and signal features were calculated over each window. Signal features would then be fed to a classifier to be classified [3]. One significant challenge in this approach is choosing the right combination of features. Many researchers have tackled this issue by analyzing different feature combinations and evaluating their performance in terms of accuracy, time efficiency and robustness [3].
Recently, a rising attention has been given to deep learning approaches, originally applied to image recognition. For hand gesture recognition using EMG signals, shifting the methodology from feature engineering to feature learning [4, 5]. Although the approach is different, the goals remain the same: improving accuracy, time efficiency and robustness of classification. An important factor when using deep

learning algorithms is that obtaining accurate results is highly dependent on the size of the training database [5].
One of the major problems in deep learning which this thesis is trying to address is the fact that, creating a sufficiently large and reliable EMG dataset for each individual is not practical. Moreover, considering a single subject, due to high inter-session variability (due to e.g. skin-electrode impedance time variability, and changes of the position of the electrodes), it may be, in principle, necessary to create a training database at the start of each session. The mentioned problems could introduce a serious challenge to the real-life translation of deep learning algorithms in myo-controlled hand prostheses.
To compare the performance of different techniques proposed by researchers around the world, the need for a publicly available benchmark dataset was always evident. Hence, Atzori in 2014 [6] published a dataset including several hand and wrist movements from 78 subjects, called “Ninapro”, which was divided into three different subsets based on acquisition system and characteristics of subjects. DataBase 2 (DB2), includes three sets of different exercises, one of these, exercise B, consists of basic wrist movements and isotonic hand configuration. In this work, this dataset (DB2) is targeted for evaluation, and for the comparison of our results with classical machine learning [7] and deep learning methods [8, 9] that have been tested on the same exercise from Ninapro DB2.

sEMG DATASETS:

Nearlab Dataset:

This database includes electrical activities of hand muscles picked up by surface electrodes during a series of predefined hand movements. The Nearlab dataset includes 11 healthy subjects including 6 male and 5 female participants (age 25±3 years). The only inclusion criteria were the absence of history of neuro-muscular disorders. The data acquisition protocol was approved by the research ethical committee of Politecnico di Milano, on October 16th, 2019. All participants had been briefed about the experiments and gave informed consent.

A. Acquisition setup

The experimental setup is composed by a screen for visual cue display, an EMG acquisition system, a laptop that receives the acquired signal via USB cable, disposable gel-based electrodes and synchronization circuitry. The EMG acquisition system used in this project was high-density EMG acquisition system “Porti” from TMSi company [10]. The cables used to connect electrodes to the device were equipped with active shielding which significantly increased signal to noise ratio. The electrodes were passive EMG Ag/AgCl electrodes with conductive gel inside them. 10 differential channels were employed in this study (Figure 0-1). The signal was sampled at the rate of 2048 Hz.
A Matlab interface was used both to visualize the acquired signals in real-time and to store the data in the PC. The synchronization code was also coded in Matlab.
The general purpose of electrode placement was to both consider muscle anatomies and achieve simplicity in placement. The exact positions of electrodes were determined according to SENIAM (Surface EMG for non-invasive assessment of muscles) [11]: 6 electrode pairs (corresponding to 6 channels) are placed around the upper forearm equally-spaced along the forearm circumference. Each differential pair is arranged along the length of the arm with 2cm distance from each other. The first electrode pair is placed 3 cm distal to the elbow (medial epicondyle),

other 5 are arranged to have same distance with respect to each other using the measured forearm circumference. The 4 remaining electrodes pairs are placed 3 cm distal to the previous electrodes. All these electrodes were placed on the dominant hand of the participant. Reference electrode is placed on the back of the wrist as suggested by SENIAM directions [11]. Position of electrodes was set by simple geometrical directions and not complicated anatomical considerations.

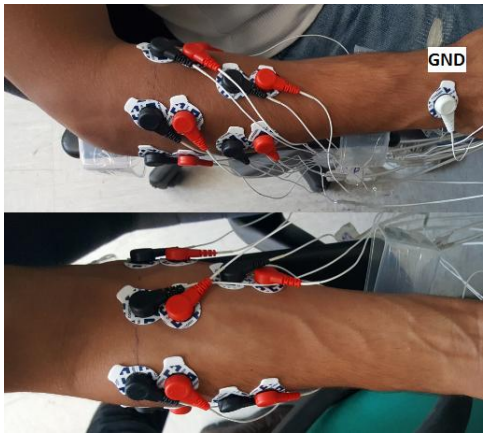

Figure 0-1 Electrode positioning

Skin preparation and electrode placement procedures took 20-25 minutes.

B. Acquisition protocol

Subjects were instructed to perform 8 classes of movement through a video cue with random order, having 3 second rest and 2 seconds preparation between movements. Figure 0-2 displays classes of movement. Each movement was repeated 5 times starting from a specific hand posture. 3 different hand posture were defined as following:

- Round 1: upward starting position, where the palm is faced upward;
- Round 2: sideway starting position, where the palm is faced medially;
- Round 3: downward starting position, where the hand palm is faced to ground.

With these 3 hand positions (upwards, downwards and sideways) each movement is repeated for total number of 15 (5x3) times by each subject.

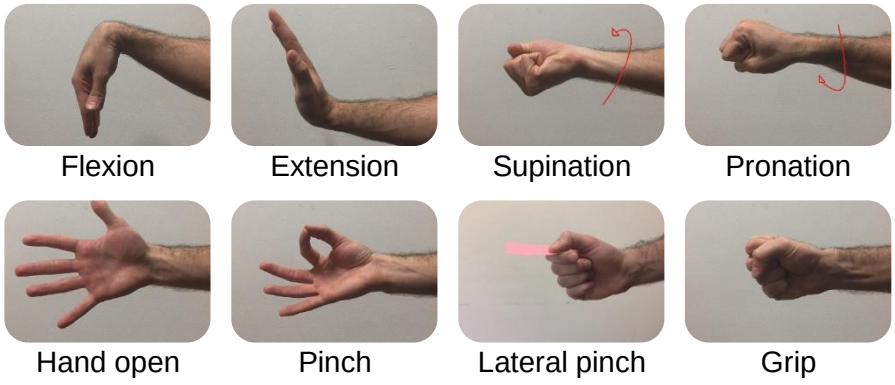

Figure 0-2 Movement classes

Synchronization: One of the most important aspects of the experiment is the synchronization of the acquired signal and the time tags of the movement execution cues. This synchronization is crucial to the process of labeling the input signals. In order to do so, a trigger input of the acquisition system is utilized. A micro-controller board (Arduino board) is used to send a pulse to the trigger channel upon receiving the instruction from PC through serial connection. The computer will start the video and send the serial command at the same time. Thus, the time of starting the video is tagged in the acquired signal by a pulse in the trigger channel.

Observation process: an observer was always monitoring the experiments. His/her role was to check the correctness of the movements (according to video cue) and record it for future steps of preparing the database with a correct movements labelling. Incorrect movements are discarded based on these reports from observations at the end of each experimental session, prior to any processing.

Ninapro Dataset:
In this study, DataBase2 (DB2) [6] consisting of 40 able-body participants from Ninapro datasets is used. DB2 is collected using 12 active double–differential wireless electrodes with a Delsys Trigno Wireless EMG system [12], which has 2k sample/second sample rate. This database includes 3 sets of exercises. The first exercise (called exercise B) which includes 17 basic movements of fingers and wrist is targeted in this thesis, due to its similarity to movement classes in Nearlab dataset. Each movement, which lasts for 5 seconds and is followed by a rest period of 3 seconds, is repeated 6 times.

PRE-PROCESSING AND DATA SEGMENTATION:
Filtering and movement onset-detection:
Raw data collected in Nearlab dataset are filtered using a 10-500Hz band pass filter (Butterworth order 4th) and a 50 Hz Notch filter (Butterworth order 2nd) in order to remove unwanted signals and power line interference [13]. Filtered data are then divided in labeled movement windows using the video cue time markers. Moreover, windows are further trimmed using a threshold-based onset detection algorithm. Three times of the rest activity obtained by electrodes was used as a threshold to determine beginning of movement execution. Furthermore, since the steady-state signal of muscles is targeted, a small part (100ms) from the beginning and end of windows are removed to eliminate the transient part of the movement signal [1].

Data Segmentation:
The labeled trimmed windows (after onset detection and transient removal) should be further segmented to be fed to network. Considering that for online applications window length plus processing time to generate classified control commands should be less than 300ms [1], window size of 250ms (512 samples) was selected for this project.

Data augmentation is a necessary step towards increasing the database size in order to be used in deep learning methods. As suggested by Côté-Allard et al. [5], sliding window approach is the most effective augmentation technique for sEMG classification. As a result, windows of 250ms (512 samples) are shifted 62.5ms (128 samples), creating 187.5ms (384 samples) overlap.
Thus in the case of traditional classifiers, the features are calculated on windows of 250ms (512 samples) over 10 channels resulting the input shape of classifier to be: number of features x number of channels (e.g. in the case of full feature set 15 x

10). And in the case of deep learning methods which uses the raw signal the input sample is in the shape of number of data points x number of channels (512 x 10).

TRADITIONAL CLASSIFIERS:

Feature Selection:

Based on the recent literature available on feature selection for sEMG classification [14, 15, 16] and datasets exploration, 15 features of time and frequency domains are used to create 4 feature sets as inputs of classifiers. Features were calculated for each movement considering the same preprocessing and segmentation of signal presented above. Feature sets are listed in table 0-1:

Table 0-1 Selected feature sets

Name	FEATURES	Number of features
Time Domain (TD)	MAV, ZC, SSC, WL	4
Improved Time Domain (ITD)	MAV, ZC, SSC, WL, RMS, IEMG, HP_A, HP_M, HP_C	9
Correlation Based (CB)	CC1, ZC, SSC, WL , HP_M, HP_C and SampEn	7
Full dataset (Full)	MAV, ZC, SSC, WL, HP_A, HP_M, HP_C, SampEn, CC1-4, RMS, IEMG, SKEW	15

MAV=Mean Absolute Value, ZC= Zero Crossing, SSC= Slope Sign Change, WL= Waveform Length, HP_A/HP_M/HP_C=Hjorth Parameters, SampEn=Sample Entropy, CC1-4=Cepstral Coefficient order 4, RMS= Root Mean Square, IEMG=Integrated EMG, SKEW= Skewness

Train and test set:

At this stage, the training data and the testing data should be separated. In each hand orientation (upward, downward, sideway) subjects were requested to repeat each movement for 5 times. In each round, 2/3 of repetitions of each movement is added to training set, while the remaining part is included in testing set. Consequently, both training and testing datasets include all hand orientations for each movement.

Outlier removal and scaling (using mean and standard deviation) are the measures taken to improve the classifier performance in this study [17]. Testing dataset remains intact during outlier removal, while during scaling, it will scale with the parameters fitted on the training data.

Classifiers:

Among most common classifiers, K-Nearest Neighbors (KNN), Support Vector Machine (SVM), Multilayer Perceptron (MLP) and Linear Discriminant Analysis (LDA) are chosen for this work. In addition to being well-known, these classifiers have shown promising results in majority of the research in sEMG classification, such as Zhai et al. [7] who used SVM on sEMG spectrograms.

To optimize the classifiers' hyper parameters, a grid search was performed. Selected hyper-parameters for each classifier are listed in table 0-2.

Table 0-2 Classifiers and thier selected hyper-parameters

Classifier	Parameter 1	Parameter 2	Parameter 3	Parameter 4
KNN	K: 40	weights: uniform		
MLP	hidden layers: (100)	Learning rate: 0.0001	activation function: tanh	solver: sgd
SVM	Regularization: 1	kernel: linear	None	gamma: auto

LDA	solver: svd

DEEP LEARNING METHODS:

There are 4 deep learning architectures proposed in this thesis, 3 of which are based on typical CNN and one is inspired by Residual CNN. All architectures can be divided into 2 stages. First stage is an inter-connected network of convolutional blocks working as a “feature extractor” and the second stage is consisted of few fully connected layers serving as the “classifier”.

The activation function used in this study is randomized rectified linear (RReLU), which was introduced in a recent Kaggle National Data Science Bowl (NDSB) competition [18]. To prevent over-fitting, the following 3 pre-cautions have been taken:

Drop out: Srivastava et al. [19] presented dropout technique, in which some random neurons with probability of p (e.g., 0.3) are eliminated from hidden layers. As a result, complex coadaptation of features between neurons can be prevented during training, leading to reduction of over-fitting.

Batch normalization: Introduced by Ioffe et al. [20], Batch Normalization (BN) was targeted to solve the need of low learning rate and careful parameter initialization in training of deep neural networks. It is a type of regularization technique, which performs input normalization in each training mini-batch.

Early stopping: The validation error in each update is monitored. When it reaches a minimum, the learner would continue training only for a certain number of iterations and then stops the training. Meanwhile, if a new minimum is observed, it will restart counting iterations before stopped. This mentioned number of iterations is referred to as Patience and is set by user.

1. Cnet2D: This architecture includes 3 convolution blocks (Conv) connected after each other constructing the feature extractor stage, followed by 2 fully connected blocks (FC) as classifier stage. Each convolutional block consists of a convolution layer with 2D filter shape (e.g., first layer’s filter size is (3,13)), BN, RReLU activation layer, max pooling and dropout. First fully connected block includes dense layer, BN, RReLU and dropout, while the second fully connected block does not include dropout. At the end, a Softmax layer has been included to create the output of classifier. Adam optimizer [21] is used as optimization method. During training, the model with minimum validation (20% of training data is randomly selected as validation set) loss is saved and used for testing; this technique is called Model Check Point and is used in this and all following networks. Figure 0-3 demonstrates the sequence of the layers in Cnet2D. The final output shape in this and other deep networks is 8, representing probabilities of the sample belonging to each of the 8 movement class.

Cnet2D’s performance is dependent on electrode positioning due to its 2 dimensional filter shapes. This fact, should be considered when applying this network to other databases.

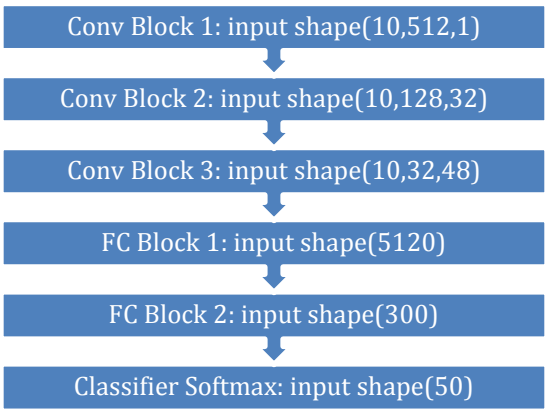

Figure 0-3 Architecture of Cnet2D

2. Cnet1D: The architecture of Cnet1D, is similar to that of Cnet2D. However, the shape of filter is such that it does not exploit the relations between channels in the feature extraction stage (e.g., filter size of first layer is (1,11)). Similar to Cnet2D Adam optimizer with same learning rate is used. The architecture of this network is the same as Cnet2D depicted in figure 0-3.

3. CnetComb: In this architecture, feature extractor stages of Cnet1D and Cnet2D extract the 1D and 2D features then the features would be concatenated and fed to one classifier stage similar to the ones used before. The architecture can be seen in figure 0-4.

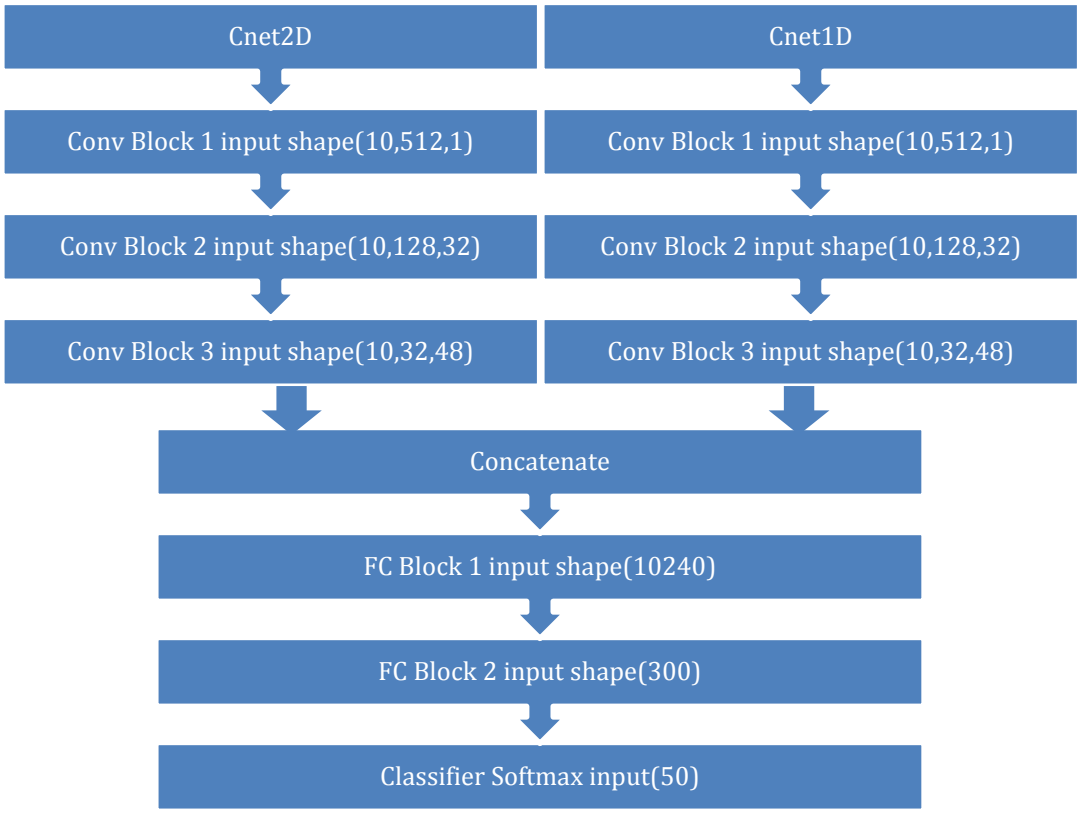

Figure 0-4 Architecture of CnetComb

4. RESnet:

This architecture is inspired by Residual Neural Network [22]. It includes skip connections to transfer the influence of deeper layers into shallower ones. The feature extractor stage of this proposed network has two branches. The left branch is consisted of 2 convolutional blocks, followed by a single convolutional layer. Right branch includes one convolutional block followed by a single convolutional layer. The output of the two single convolutional layers of two branches are summed together and the result is given to a batch normalization (BN) layer, RRelu, average pooling layer and dropout ("referred to as β block"). A flatten layer is used to conclude the feature extractor stage. At the end, classifier stage is added. The classifier stage is 2 fully connected blocks followed by a Softmax layer, similar to previous networks. Figure 0-5 demonstrates the architecture of this network.

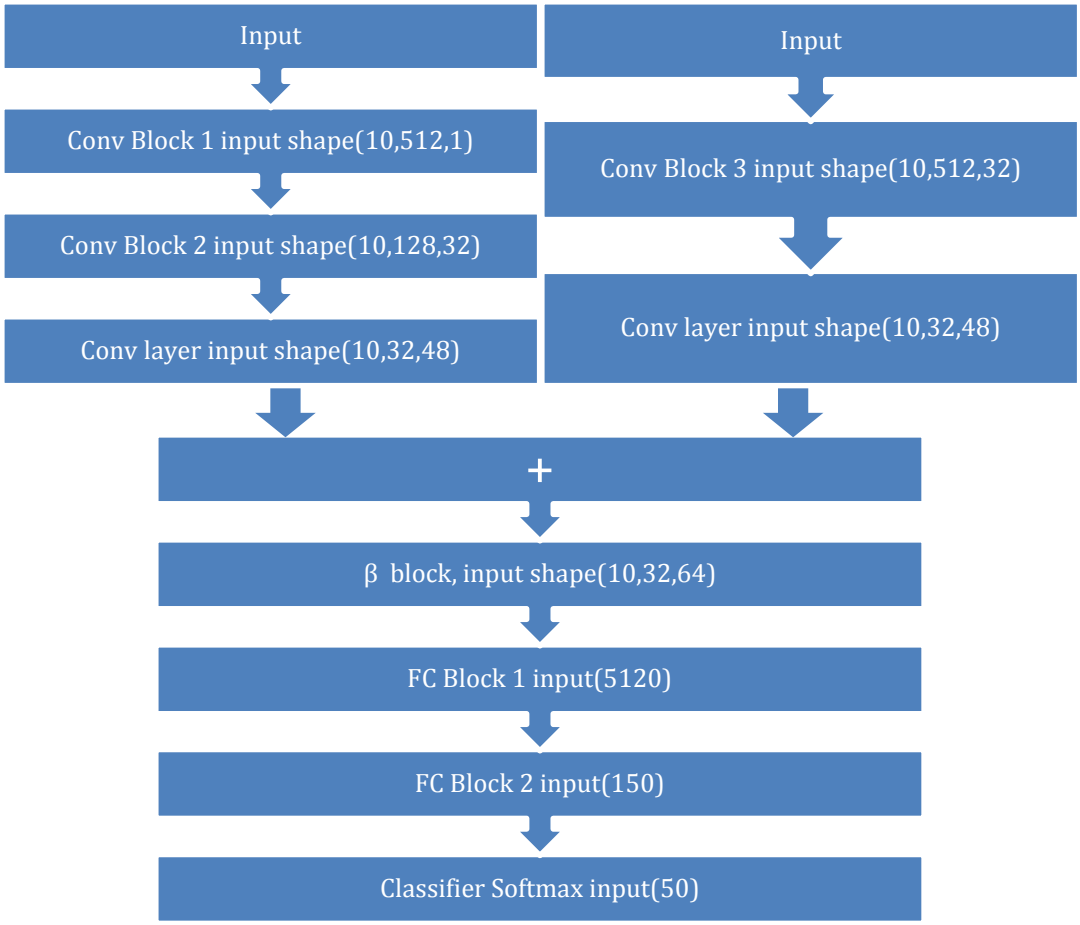

Figure 0-5 Architecture of RESnet

TRANSFER LEARNING:

In this study, TL is used to leverage the shared information among different subjects to obtain bigger training dataset to train deep neural networks. There are two transfer learning schemes designed in this study.

Method I: Freeze & fine-tune

The idea is to use a pre-trained network (trained in the source domain), remove the classifier stage and attach a new adapted (according to target domain) classifier stage. Finally, the network should be re-trained in the target domain. The process

of re-training is done by freezing the deeper layers' weights and fine-tuning the shallower layers. Fine-tuning can be achieved by initializing with previous weights and choosing very low learning rates.
This technique is employed on the deep learning network with highest average accuracy. First, network is pre-trained with the data of 5 selected subjects with the highest single subject accuracy except the target subject. Afterwards, the first 2 convolutional blocks (except for Batch Normalization layer) are frozen and other layers (including the last convolutional block plus all the fully connected blocks) are fine-tuned for targeted subject using the subject's database. Figure 0-6 displays the architecture of network.

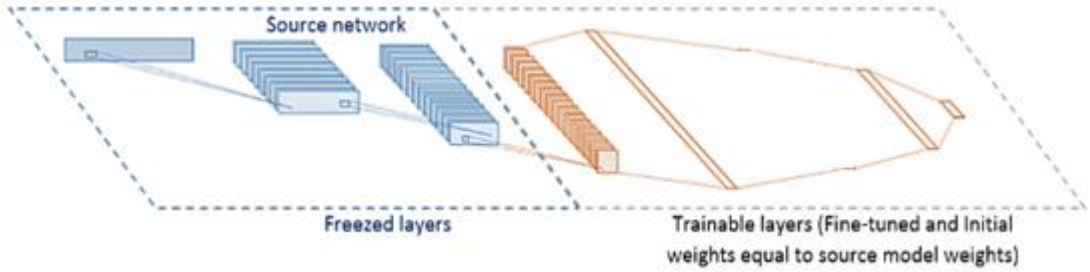

Figure 0-6 Architecture of Freeze & fine-tune method

Method II: Parallel networks
In this method, two networks are used in parallel. One is trained on the 5 selected subjects with the highest single subject accuracy except the target subject (referred to as "Source Network"), while the other is trained on the target subject's database (referred to as "Target Network"). The features extracted by the mentioned networks would be concatenated, while their classifier stages would be disregarded and a new classifier stage is added after feature layer. Finally, the two parallel feature extractors would be frozen and the classifier stage would be trained using the target subject's database (with random initialization). Figure 0-7 illustrates the final model.

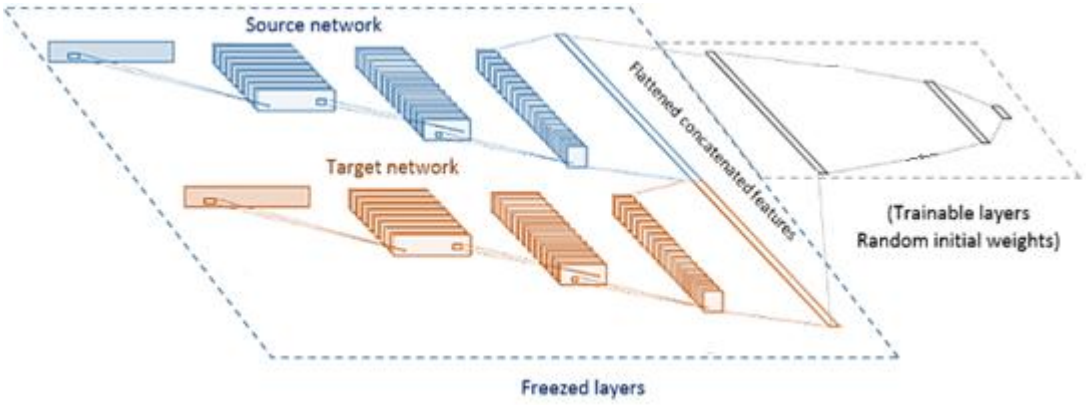

Figure 0-7 Architecture of Parallel networks method

EXPERIMENTAL RESULTS:
Training and testing databases:

In order to separate database into training and testing datasets, 3 out of 5 repetitions (~2/3 of repetitions) in the case of Nearlab dataset and 4 out of 6 in the case of Ninapro DB2 are considered for training and the remaining for test.

Runtime analysis:

An estimation of processing time is needed when approaching an online classification problem. Utilizing a 64-bit Windows based computer, with a 6 GB RAM and an Intel® Core™ i5 CPU, the required time for extracting each individual feature and also proposed feature sets is reported in table 0-3.

Table 0-3 Extraction time (ms) of each feature over all channels

Feature	MAV	ZC	SSC	WL	HP_A	HP_M
Time (ms)	7.925×10^{-2}	8.927×10^{-2}	6.776×10^{-2}	9.979×10^{-2}	8.788×10^{-2}	1.833×10^{-1}

Feature	HP_C	SampEn	CCs	RMS	IEMG	Skew
Time (ms)	2.558×10^{-1}	147.9	125.3	9.5456×10^{-2}	3.1459×10^{-2}	1.3

Feature sets	TD	ITD	CB	Full
Time (ms)	~0.5	~1	~270	~270

According to table 0-3, extracting Full feature set requires almost 270ms (sum of all calculation times) while Improved Time Domain (ITD) feature set takes only 1ms. Hence, comparing to Full, ITD feature set can be regarded as fast calculating feature set.

In table 0-4 prediction time of each classification method (using Full feature set) for one sample is listed. These delays should be summed up with feature set calculation times. Table 0-5 reports the prediction time for deep learning methods. Prediction time was acquired by averaging the time needed for predicting 1000 samples. This process for traditional classifiers was performed on the mentioned CPU. However, for deep learning methods a Tesla P100-PCIE-16GB GPU was used.

Table 0-4 Prediction time (ms) of each traditional classifier using Full feature set (should be added to feature set calculation time)

Classifier	KNN	LDA	MLP	SVM
Time (ms)	3.554	0.801	1.021	0.879

Table 0-5 Prediction time (ms) of all deep learning networks

Classifier	Cnet1D	Cnet2D	CnetComb	RESnet
Time (ms)	4.81	4.97	6.34	5.16

Performance analysis:

The results of all the traditional, proposed deep learning and transfer learning approaches, tested on Nearlab dataset are reported in the table 0-6. Moreover, results of traditional and deep learning methods on Ninapro DB2 dataset is listed in table 0-7. The results are average accuracy over all subjects included in each dataset (11 subjects in Nearlab dataset and 40 in Ninapro DB2). The highest average accuracy in each classification family is shown in bold format. It should be considered that the number of movement classes in Nearlab database is 8 and in Ninapro DB2 exercise B is 17. The transfer learning approaches applied on Nearlab dataset used Cnet1D architecture as base model since Cnet1D obtained the highest accuracy (without transfer learning) when testes on Nearlab dataset.

Table 0-6 Classification results on Nearlab dataset

Classification family	Classifier(feature set)	Average accuracy (%)	Standard deviation (%)
Traditional machine learning	KNN (ITD)	89.20	4.35
	LDA (Full)	**92.55**	4.23
	MLP (ITD)	91.45	2.97
	SVM (ITD)	91.72	3.60
Deep learning	Cnet1D	**93.23**	2.77
	Cnet2D	92.81	3.22
	CnetComb	92.41	3.26
	RESnet	93.20	3.42
Transfer learning (using Cnet1D)	Freeze & fine-tune	93.27	3.69
	Parallel networks	**93.48**	2.86

Table 0-7 Classification results on Ninapro DB2 dataset

Classification family	Classifier(feature set)	Average accuracy (%)	Standard deviation (%)
Traditional machine learning	KNN (Full)	75.71	6.52
	LDA (Full)	79.95	5.73
	MLP (Full)	**80.97**	5.44
	SVM (Full)	79.50	6.25
Deep learning	Cnet1D	77.69	6.54
	Cnet2D	**80.34**	6.30
	CnetComb	79.93	5.98
	RESnet	77.03	7.09

According to table 0-6, comparing average accuracy of traditional classifiers and deep learning methods shows that Cnet1D has highest average accuracy specifically comparing to classifiers using time-domain features which are fast in calculation (KNN, MLP and SVM). In order to compare Cnet1D with LDA, in addition to table 0-6, prediction times listed in table 0-3 to 0-5 must be considered. As table 0-6 points out, Cnet1D has higher average accuracy than LDA. Also, LDA with Full feature set needs almost 270ms for producing a single prediction, which is much slower than Cnet1D that produces single prediction in almost 5ms. Hence, both in terms of calculation time and average accuracy Cnet1D outperforms LDA. It is worth to mention that LDA with ITD feature set (much faster than LDA with Full feature set) has average accuracy of 91.62%. Also from table 0-6 it can be seen that applying transfer learning on Cnet1D has improved the classification accuracy to 93.48%, at the cost of increased training time.

Analyzing accuracy/loss curves obtained during training of Ninapro DB2 (table 0-7) reveals that the proposed deep networks are possibly over-fitting on the available data and thus reducing the performance comparing to machine learning methods. The mentioned accuracy/loss curves demonstrated very low correlation between training accuracy/loss and testing accuracy/loss (high offset between the two plus

high fluctuations in testing accuracy when there is low fluctuations on training accuracy). This could be due to the fact that deep learning methods are designed for Nearlab database with much higher samples per movement class ratio.
Table 0-7 indicates that Cnet2D obtained the highest average accuracy comparing to other networks on Ninapro DB2 dataset. The success of 2D filter shapes here, may be related to the electrode positions in Ninapro database which differs from Nearlab dataset.

Conclusion and feature work:
In this thesis, multiple pattern recognition techniques have been designed, implemented and tested in order to classify 8 hand movements using Surface EMG signals. Classical machine learning methods along with deep learning techniques are employed for this task. To tackle the problem of training deep networks with limited databases, transfer learning approach has been applied.
A comprehensive comparison between classical machine learning methods and deep learning methods has been conducted. In addition, the feasibility of employing such methods is investigated using runtime analysis. It has been shown that when considering fast algorithms (using time domain features), deep learning algorithms outperform classical machine learning algorithms with no significant increase of cost in terms of time consumption.
Transfer learning approach has been also deployed to answer the following question: could knowledge learned from other subjects' datasets be useful to train a classifier for a given subject? To this aim, two methods have been designed and implemented to integrate previous subjects' knowledge into the classifier for the targeted subject. It has been demonstrated that the proposed transfer learning algorithms are able to improve performance in terms of accuracy.
To compare the results, the results published by studies on Ninapro DB2 exercise B for classifying hand movement gestures are reported in table 0-8. Table 0-8 compares the related research works with the obtained accuracies with best methods used in this study. MLP with Full feature set and Cnet2D are reported, since these two algorithms displayed the highest accuracy among the proposed classical machine learning and deep learning algorithms respectively. According to this table, simple classical approaches (MLP) used in this study shows comparable results with respect to state of the art studies. Zhai et al. [7] classified spectrograms of DB2 sEMG signals combined with PCA for dimension reduction. They used SVM as classifier and obtained 75.74% accuracy on exercise B. Later, they improved their work [8] by proposing a self-recalibrating CNN to eliminate the need of user training over time. The classification accuracy of their new method on Ninapro DB2 was 82.22%, when tested on exercise B. Moreover, in 2019 Huang et al. [9] used a CNN-LSTM network in order to fully capture the spatial and temporal features of sEMG spectrograms, the resulting accuracy on DB2 exercise B was 80.93%.

Table 0-8 Classification accuracy (%) comparison with related work over Ninapro DB2

	Classification method	Accuracy (%)
Related work	Spectrogram with SVM [7]	75.74
	Spectrogram with CNN [8]	82.22
	Spectrogram with CNN and LSTM [9]	80.93
This study	MLP with Full feature set	**80.97**
	Cnet2D	**80.34**

In future studies, probabilities provided by output of Softmax layer can be used to assign each sample with two movement classes instead of one. For example, a

classifier trained for pinch and wrist rotation target classes could assign pinch + wrist rotation (similar to holding a key and rotating it in the lock) along with single classes. One other aspect that could be investigated in future is to test the proposed networks in an online classification setting while providing visual feedback representing the output of the classifier to the user.

Other transfer learning algorithms could be explored in future work. An image representation of sEMG signal such as Spectrograms could be used as input of classifiers. In this way, in a transfer learning framework the strong pre-trained image classifiers could be leveraged. Another example can be creating several classifiers trained on other subjects EMG data and use a voting procedure to classify data from the target subject.

Table of Contents

Abstract 5
Astratto 6
Summary 7
1 Introduction 23
1.1 Thesis outline 23
1.2 Problem statement 23
1.3 Aim of the thesis 24
1.4 Theoretical foundation 24
1.4.1 Electromyogram signal 24
1.4.2 Myoelectric prosthesis 26
1.4.3 Traditional classifiers 28
1.4.4 Artificial neural network 31
1.4.5 Convolutional neural network 34
1.4.6 Transfer learning 36
1.4.7 Overfitting 36
1.5 State of the art 37
2 Materials and methods 39
2.1 Nearlab Dataset 39
2.1.1 Participants 39
2.1.2 Protocol definition 39
2.1.3 Hardware and software 41
2.1.4 Electrode placement 42
2.1.5 Database creation 43
2.2 Ninapro Dataset 43
2.3 Data analysis 43
2.3.1 Data pre-processing 43
2.3.2 Data segmentation 48
2.3.3 Classical machine learning 49
2.3.4 Deep learning 55
2.3.5 Transfer learning 63
2.3.6 Summary of explored methods 65
2.4 Performance measures 65
2.4.1 Basic metrics 65
2.4.2 Multiclass metrics 66
2.4.3 Variance analysis 66
3 Classical machine learning results 67

3.1 Classification performance analysis 67
3.1.1 Nearlab dataset 67
3.1.2 Ninapro DB2 69
3.2 Class specific analysis 73
3.2.1 Modification on movement classes 75
3.3 Run time analysis 76
4 Deep learning results 78
4.1 Classification performance analysis 78
4.1.1 Nearlab dataset 78
4.1.2 Ninapro DB2 dataset 83
4.2 Class specific analysis 86
4.3 Run time analysis 88
4.4 Transfer learning 89
4.4.1 Nearlab dataset 89
4.4.2 With and without transfer learning 92
4.4.3 Run time analysis 93
5 Simulating online classification 95
5.1 Control panel 95
5.2 Plots 95
5.3 Real-time classification 96
6 Conclusion 98
Reference list 103
List of Figures 107
List of tables 110

1 Introduction

1.1 Thesis outline

This thesis is arranged in 6 chapters. The first chapter starts with defining the problem addressed in this work and the goals of the thesis. Next, it provides the necessary scientific background relevant to the context of this project. The background includes the origin of EMG signal, signal acquisition methods, a review of myoelectric prostheses, well-known classification algorithms, transfer learning technique and other relevant theoretical concepts. First chapter ends by presenting state of the art and recent advances in hand gesture recognition using sEMG signals.
The second chapter thoroughly discusses the methods and algorithms used in this study. To start, all the technical details of how the custom EMG database was created is reported. Here, the participant population, experimental protocol, electrode positioning, hardware and software used in the experiments are covered. Moving on, it introduces all the pattern recognition algorithms used in this study, covering traditional machine learning, deep learning and transfer learning approaches. Chapters 3 and 4 are dedicated to evaluation of all the methods that were introduced in second chapter. Chapter 3 evaluates classical machine learning algorithms, chapter 4 covers deep learning and transfer learning approaches. Chapter 5, introduces the graphical user interface designed in this thesis for simulating online classification schemes. Chapter 6 discusses the obtained results throughout this work, conclusions and suggestions for possible future work in the continuation of this thesis work.

1.2 Problem statement

For people with upper limb disabilities, independently performing daily tasks that require hand function such as holding objects, opening/closing doors and eating meals is a major challenge. For this population, the use of an assistive device targeting in particular the hand could be beneficial. According to the type of disability, this device can be a prosthesis (in the case of replacing a missing limb) or an orthosis (in the case of supporting an existing limb). Among different kinds of hand prosthesis, myo-controlled hand prosthesis has gained rising interest among researchers. Myo-controlled technique uses signals acquired from limb muscles to control the assistive device [1]. In myo-controlled hand prostheses, the signals acquired from users' muscles is classified to predict hand movement intention. Then the predicted movement will be used to control the artificial hand. Although myo-controlled devices have been introduced for many years, due to their insufficient classification accuracy and robustness, they have not yet been accepted by a considerable portion of the targeted population [2].
Traditionally, EMG signals were pre-processed to remove unwanted signals. Then signal is segmented into windows and signal features were calculated over each window. Signal features would then be fed to a classifier to be classified [3]. One significant challenge, which is present to this day, is choosing the right combination of features. Many researchers have tackled this issue by analyzing different feature combinations and evaluating their performance in terms of accuracy, time efficiency and robustness [3].
Recently, a rising attention has been given to deep learning approaches for hand gesture recognition using EMG signals, shifting the methodology from feature engineering to feature learning [4, 5]. Although the approach is different, the goals

remain the same: improving accuracy, time efficiency and robustness. An important factor when using deep learning algorithms is that obtaining accurate results is highly dependent on the size of the training database [5].
One of the major problems in deep learning which this thesis is trying to address is the fact that, creating a sufficiently large EMG dataset for each individual is not practical. Moreover, considering a single subject, due to high inter-session variability, it may be necessary to create a training database at the start of each session. The mentioned problems could introduce a serious challenge to the real life application of deep learning algorithms in myo-controlled hand prostheses.

1.3 Aim of the thesis

The aim of this thesis is to work towards an accurate and robust hand gesture recognition algorithm for myo-controlled upper limb assistive devices. In addition, the proposed algorithm should be practical and feasible for real life applications. Finally, in order to get as close as possible to real-life applications, we aim to implement an online classification software to be used by myo-controlled prostheses in real-time.
To this aim, the contribution of this work is twofold: I) develop novel deep network architectures specific for target application and II) use different transfer learning approaches with limited amount of data available for each participant. Transfer learning is employed as a possible solution to the problem of large databases necessary for training deep networks.
Additionally, an upper limb EMG database has been created through a series of experimental sessions. This database includes electrical activities of hand muscles picked up by surface electrodes during a series of predefined hand movements. The proposed approaches are tested on this custom database and also on a publicly available EMG dataset (Ninapro database 2) to provide results comparable with the state of art. A comparison has also been carried out with well-known classical hand gesture recognition algorithms.
In the end, a program simulating online classification has been designed to display the real-time implementation of a hand gesture recognition algorithm in a myo-controlled prosthesis.

1.4 Theoretical foundation

In order to fully understand hand gesture classification, necessary topics are covered in this section. Along with electromyogram signal and myo-controlled hand prosthesis, the definition and terminology of methods utilized in this work are discussed in order to familiarize the reader with essential concepts.

1.4.1 Electromyogram signal

Motor neurons are neuronal cells located in the central nervous system (CNS) controlling a variety of downstream targets. According to their location, motor neurons are classified as (i) upper motor neurons, that originate from the cerebral cortex and (ii) lower motor neurons, that are located in the brainstem and spinal cord. Alpha (α) motor neuron is a type of lower motor neuron that innervate skeletal muscles responsible for movements, and which is the key of muscle contraction [23].
The human motor control system is organized in a hierarchical way and its smallest functional unit is the motor unit. A motor unit consists of an α-motor neuron in the spinal cord and the muscle fibers it innervates, as indicated in Figure 1-1 [24].

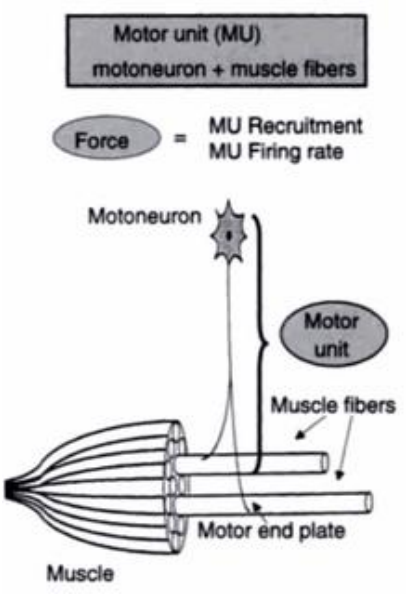

Figure 1-1 Motor unit and its components [24]

Muscle contraction is initiated by activation of the lower motor neuron. When an action potential is generated in motor neuron and is propagated through its axon, it produces an action potential in muscle fiber that is connected to it. Repeated activation of lower motor neuron, produces trains of action potentials that propagates along the muscle fiber membrane and then contracts the muscle [13, 25]. This electrical activity of the muscle during contraction and force production is called electromyographic (EMG) signal.
EMG could be collected by either I) surface electrodes or II) needle electrodes [26]. Figure 1-2 illustrates the two categories.

Figure 1-2 Needle electrode (right) (https://www.ambu.com/neurology/emg-electrodes/product/ambu-neuroline-concentric) and surface electrode (left) (https://bio-medical.com/covidien-kendall-disposable-surface-emg-ecg-ekg-electrodes-1-3-8-35mm-50pkg.html)

Surface electrodes are applied on the surface of the skin; hence, they can't target the activity of a single muscle fiber. On the other hand, needle electrodes, penetrate the skin. Since they are inserted into the contracting muscle, they are able to record MUAPs. However, since surface electrodes are non-invasive, they are the most common method of EMG recording [26,24].
A sample surface EMG signal is shown in Figure 1-3. The amplitude of sEMG signal ranges from µV to low mV (0-10 mV peak to peak) depending on the muscle types and conditions during the recording [27]. The factors causing sEMG signal changes

are electrode re-positioning, changes in skin electrode impedance (e.g. sweating), changes in relative position of electrode with respect to muscle fibers [2]. The most used frequency band of EMG signal is between 0 to 500 Hz, considering that lower frequencies (0-20 Hz) are often disturbed by noises [11]. The main sources of this noise are motion artifacts, electrical noises from electrical components inside the circuitry and quasi-random nature of the firing rate of MUs [13].

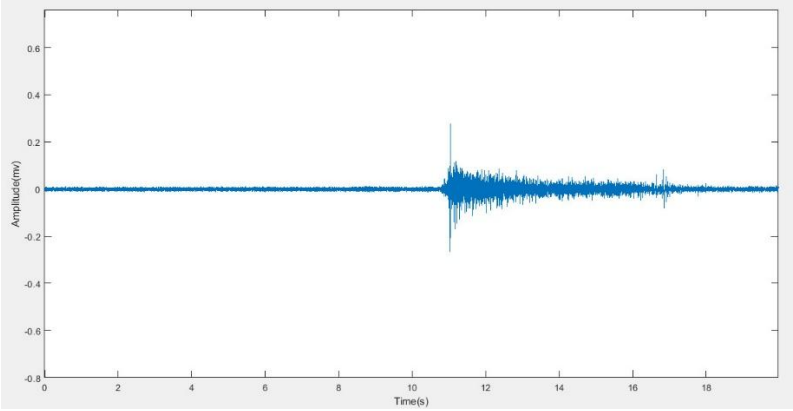

Figure 1-3 A sample of surface EMG signal

1.4.2 Myoelectric prosthesis

Considering the application of EMG signals in improving the quality of life of people with hand amputation, the focus of this part is on hand prosthetic. These assistive devices are categorized in passive and active prostheses. In passive prosthesis, the force to control the grasping mechanism is applied externally, for example, by the functional hand, in Figure 1-4 an example is shown. In active prostheses, this force is applied internally, for example, by an electric actuator or a body-powered cable [33]. Active prostheses themselves are divided in two groups, depending on the means of generating movement in the joints: body-powered and electrically powered. In the body-powered prostheses, voluntary movements of shoulders and/or limb stump, controls the movement of prosthetic hand attachment using a body harness and a cable system [28], an example is demonstrated in Figure 1-5. On the other hand, electrically powered hand prosthesis employs electrical actuators inside each joint to create motion, a sample of this prosthesis is shown in Figure 1-6.

Figure 1-4 An example of passive prosthetic hand (http://rehabindyo.tripod.com/be-passive.html)

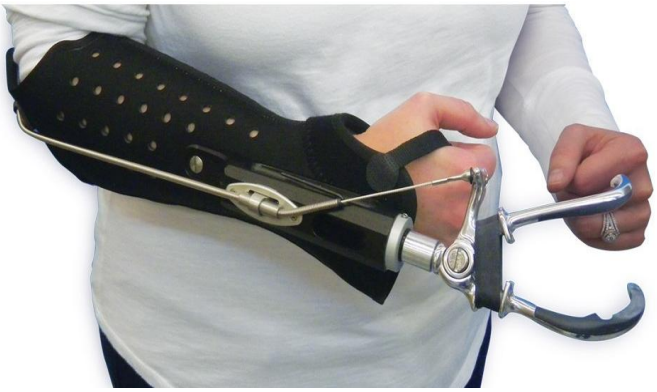

Figure 1-5 An example of body-powered ahnd prosthesis (www.trsprosthetics.com/)

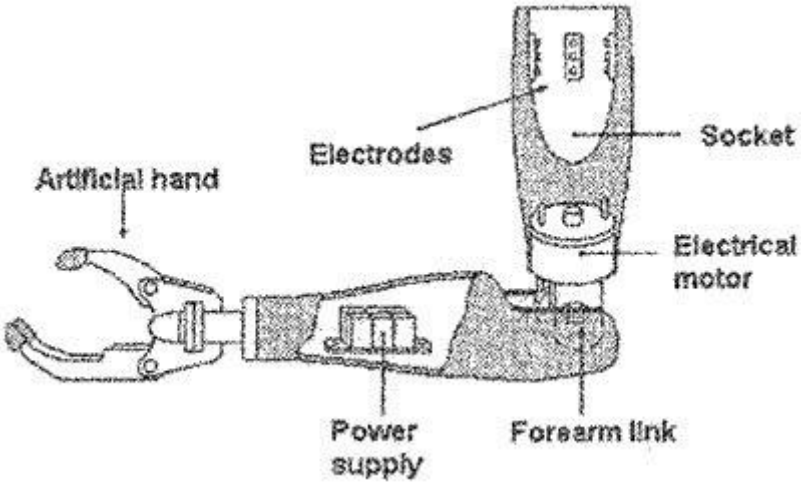

Figure 1-6 An example of how the electonic powere hand prosthesis is actuated (https://www.rehab.research.va.gov)

The EMG signal acquired from the muscle of the functional part of the user's hand is used to detect the subject's intention and control the electrical actuators. The prostheses which use such control mechanism are called myoelectric prostheses. Myoelectric prostheses offer advantages in many aspects but the most important one is that they operate based on user's intention providing a natural interface between user and assistive device. For this reason, myoelectric prosthesis are targeted in this study.

Controlling schemes of myoelectric hand prostheses can be split into two categories: pattern recognition- and non-pattern recognition-based. In pattern recognition group, the pattern produced in EMG signals during muscle contraction or limb movement is used to detect the user's intention and control the prosthetic hand. A classification module classifies specific patterns into specific movement categories. Based on classification performance, this method could offer several function classes. In contrast, non-pattern recognition methods (e.g. on/off controllers and/or finite state machines) provide as output limited and predefined control commands based on a sequence of muscle contractions [1].

Figure 1-7 illustrates a typical pattern recognition based control system. The process starts with picking up electrical activity from the skin of the user. Then the signal is prepared before digitalization. A typical analog preparation includes amplification, filtering and sampling. At this point, the signal to noise ratio is increased as much as

possible in the analog signal. The prepared signal is then digitalized to be processed by the pattern recognition algorithm whether its applied by a computer or an embedded system. Pattern recognition algorithms generally start by windowing the input signal, then the features are calculated for each window and the classier categorizes the input in predefined groups. Using the result of classifier, a controller activates the relevant physical (e.g. prosthesis) or virtual (in the case of computer interfaces) actuator. Adaptive systems also include a feedback that uses the outcome of the system to modify the controller.

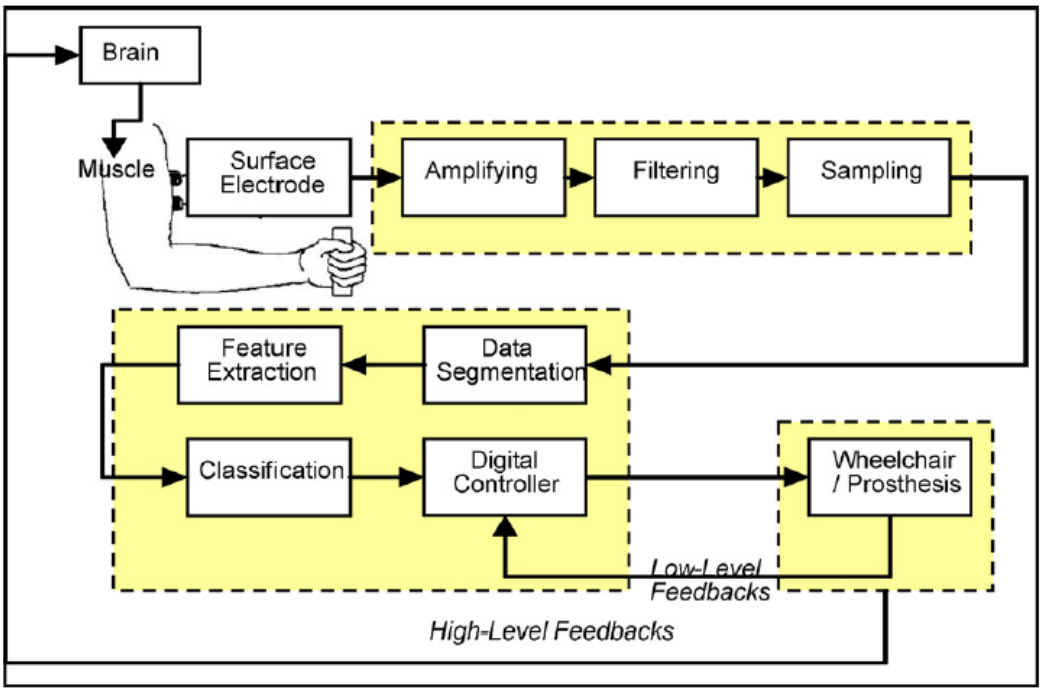

Figure 1-7 Diagram of typical pattern recognition-based hand prosthesis [1]

1.4.3 Traditional classifiers

Generally, pattern recognition can be divided into two steps: finding distinct characteristics of signal (feature extraction) and categorize signals into different groups (classification) [29].
To construct the feature vector, *n* unique characteristics of the signal must be identified, that fully conveys the information included in the signal [30], some examples will be investigated in section 2.2.3.
Classifiers fall into the supervised category of machine learning algorithms. As opposed to non-supervised techniques, supervised methods use labeled datasets to learn in order to solve the problem. In order to do so, researchers divide the available labeled datasets in the format of {(input, label)} into training and testing datasets. The training set will be used by the classifier to learn to improve its performance and the unseen test datasets is used to evaluate the final classification performance. The learning procedure starts with calculating a loss function (error) which represents the error between the output of the classifier and the desired output (label). The loss value calculated over each sample is then used to modify the parameters of classifier in order to decrease the total error value in the training dataset.

In past years, various machine learning classification techniques have been proposed for the hand gesture recognition based on sEMG signal. The most promising approaches are hereby introduced to the reader.

K-nearest neighbor: One of the simplest classification procedures is k-nearest-neighbor (K-NN). For this network no training procedure is required. This supervised method assigns the class of a new sample based on its K-th nearest samples. The most repeated class among its neighbors would be the predicted class for that new sample [31]. The whole algorithm is shown in Figure 1-8.

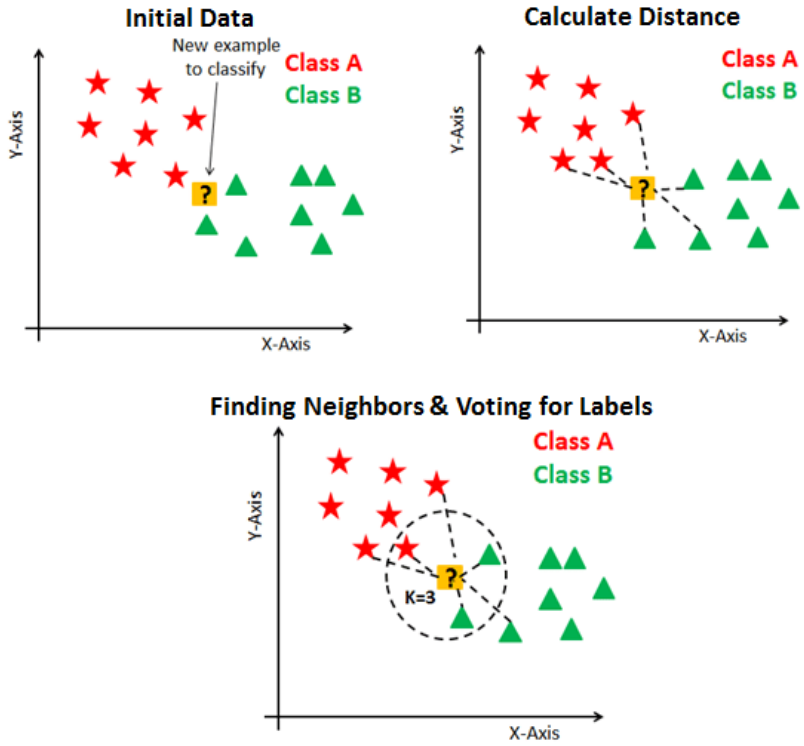

Figure 1-8 K-NN classification algorithm for samples with 2 features and K=3

The hyper-parameters in this method can be number of considered neighbors (K) and the distance measure. These parameters are not learnable and are chosen by the researcher by testing the algorithm performance on unseen labeled data. In the case of choosing a small number of neighbors, the noise will have a higher influence on the result, and a large number of neighbors make it computationally expensive. As for distance measures, Euclidean, Manhattan and Minkowski are the most popular ones.

Linear discriminant analysis: Discriminant analysis (DA) is a classification technique that uses the features of the new sample to calculate the joint probabilities of that sample with the training groups (posterior probability). Then it will assign the new sample to the group with maximum joint probability. In the linear discriminant analysis (LDA) each category is assumed to have a multivariate normal density (prior probability) with a shared covariance matrix.
Consider sample **x** with d number of features ($\boldsymbol{x} = [x_1, x_2, ..., x_d]$), using Bayes' rule for each class k (total numbers of classes is C) we have prior probability ($P(\boldsymbol{x}|y = k)$) and posterior probability ($P(y = k|\boldsymbol{x})$) as:

$$P(\boldsymbol{x}|y=k) = \frac{1}{(2\pi)^{d/2}|S_k|^{1/2}} \exp\left(-(\boldsymbol{x}-\boldsymbol{\mu}_k)^T {S_k}^{-1}(\boldsymbol{x}-\boldsymbol{\mu}_k)\right) \quad (1.2)$$

$$P(y=k|\boldsymbol{x}) = \frac{P(\boldsymbol{x}|y=k)\,P(y=k)}{\sum_j^C P(\boldsymbol{x}|y=j)P(y=j)} \quad (1.3)$$

Where S_k is the covariance matrix and $\boldsymbol{\mu}_k = [\mu_{1d}, \mu_{2d}, ..., \mu_{kd}]$ is the mean values of $\boldsymbol{x}$ within k-th class. As mentioned before, covariance matrices of all groups are equal to $S_k = S$. Using log function and replacing (1.2) in (1.3), we obtain a linear function as in (1.4):

$$logP(y=k|\boldsymbol{x}) = -(\boldsymbol{x}-\boldsymbol{\mu}_k)^T {S_k}^{-1}(\boldsymbol{x}-\boldsymbol{\mu}_k) + logP(y=k) + Cst \quad (1.4)$$

Where the constant term Cst corresponds to the denominator of $P(y=k|\boldsymbol{x})$, in addition to other constant terms from normal distribution. The predicted class is the one with maximum log-posterior [32].

Support Vector Machine: The original support vector machine (SVM) is a binary classifier able to define a hyperplane in a high-dimensional space to separate two classes and assign the new sample to one of them.
Let d-dimensional training sample $\boldsymbol{x}_i$ (i = 1, 2, 3, ... n, where n is the number of samples) belong to one of the two classes 1 or 2 and the associated labels be 1 ($y_i = +1$) and -1 ($y_i = -1$), respectively. For linearly separable data, a linear separating hyperplane (f(**x**)) can be defined as:

$$f(\boldsymbol{x}) = \boldsymbol{w}\cdot\boldsymbol{x} + b = \sum_{i=1}^{d} w_i x_i + b = 0 \quad (1.5)$$

Where the weight vector ($\boldsymbol{w}$) has the same dimension as the sample ($\boldsymbol{x}$) and b is an additional scalar parameter. Considering the $sgn(f(\boldsymbol{x}))$ as decision function, the following constraints must be met by hyperplane:

$$y_i(\boldsymbol{x}_i\cdot\boldsymbol{w}+b)-1 \geq 0 \iff \begin{cases} f(\boldsymbol{x}_i) = \boldsymbol{w}\cdot\boldsymbol{x}_i + b \geq 1 & y_i = +1 \\ f(\boldsymbol{x}_i) = \boldsymbol{w}\cdot\boldsymbol{x}_i + b \leq -1 & y_i = -1 \end{cases} \quad (1.6)$$

The optimal hyperplane must be defined in a way to maximize the margin of separation (δ). The margin is equal to minimum distance between training points and the separating hyperplane. The optimal separating line for a 2-D space is shown in Figure 1-9.

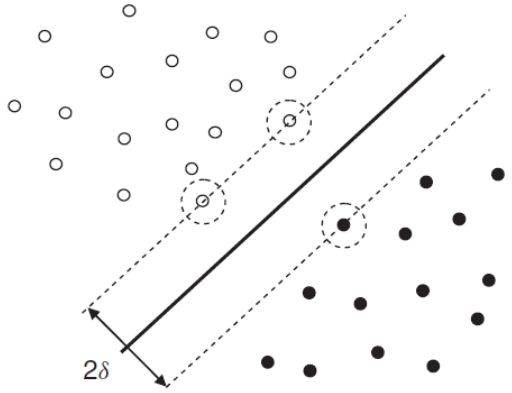

Figure 1-9 The optimal seperating line and maximum margin in a 2-D input space [17]

Only a number of training data are at the minimum distance from the hyperplane and thus define the hyperplane; they are called support vectors (SV). The number of SVs are lower than number of training data which makes this method considerably fast.
For non-separable cases (few points placed on the wrong sides), taking into account the noise with slack variables ξ_i and error penalty λ, optimal hyperplane must minimize the following optimization problem:

$$\phi(\boldsymbol{w}.\xi) = \frac{1}{2}(\boldsymbol{w}\cdot\boldsymbol{w}) + \lambda(\sum_{i=1}^{n}\xi_i) \qquad (1.7)$$

Subject to:

$$y_i(\boldsymbol{x}_i\cdot\boldsymbol{w}+b) - 1 \geq \xi_i. \qquad i = 1.2.\cdots.n \qquad (1.8)$$

Where ξ_i is measuring the distance between the margin and the sample $\boldsymbol{x}_i$ placed on the wrong side of the margin.
An advantage of this method is that the input space can be mapped into a higher dimensional space (called feature space) using a kernel function. The function of kernel is to take data as input and transform it into the required form. It means that with a kernel function this method can produce features from input data. In cases that data in their original space is not linearly separable this transformation can result in linear separation in feature space.
In a multi-class problem, multiple binary classifiers are used to categorize one class from another, ignoring all others (one versus one), then each input is classified by all the binary classifiers and the final classification result is driven from approaches like majority voting [34].

1.4.4 Artificial neural network

Artificial neural networks (ANN) were designed to simulate human central nervous system. The basic component of nervous system is neuron. A neuron is mainly consisted of cell body (processor), dendrite (input) and axon (output). Neurons are connected to each other through dendrites and axons, the junctions of dendrites and axons are called synapses. A dendrite can be connected to multiple axons of different neurons which makes the interconnections between neurons very complex. When the sum of signals received from other neurons reaches a certain threshold, a pulse will be generated as output and propagates through axon in a given neuron [30].
In ANN, nodes and arcs resemble cell bodies and their connections through dendrite and axons, respectively. Each arc links two nodes and is associated with a weight. A node produces an output by applying an activation function on the received input from other nodes. Inputs from other nodes are adjusted by the weight of the arc that linked these two nodes [17].
The first artificial neuron model was proposed by McCulloch and Pitts in 1943 [35], who used a sign function as activation function, resulting in a binary output. In 1957, Rosenbaltt [36] developed a neural network with only one neuron, called *perceptron* which is depicted in Figure 1-10 and can be used for linearly separable problems. Output value of this network is the predicted class for the input sample. The parameters of this network are the weights associated to each input feature and the bias ϑ. These parameters should be learned during training [17]. The training

procedure again takes place using the training set and optimizing the parameters to reduce loss function.

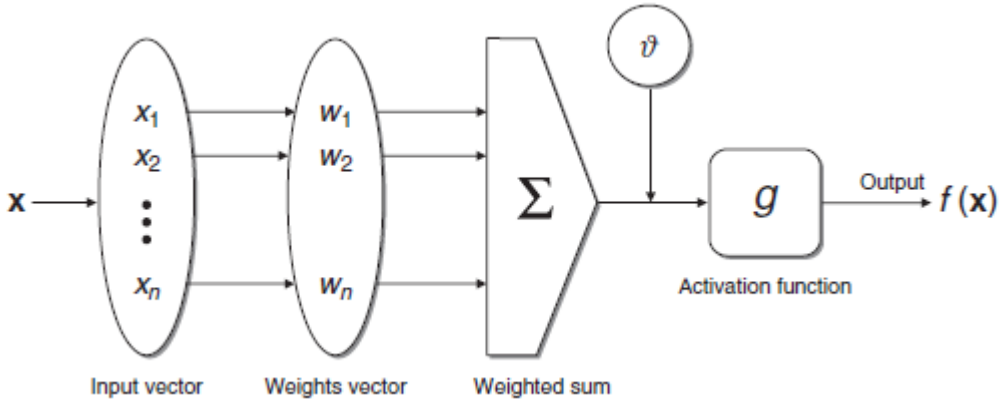

Figure 1-10 Operating model of perceptron network [17]

Assuming the parameters ($w_1,\ldots, w_n$ and ϑ) are known, the predicted output of each n-dimensional sample is calculated by applying activation function on weighted summation of input features subtracting the bias (threshold):

$$f(\boldsymbol{x}) = sgn(w_1x_1 + w_2x_2 + \cdots + w_nx_n - \vartheta) = g(\boldsymbol{w}'\boldsymbol{x} - \vartheta) \qquad (1.9)$$

In this binary classification case, the prediction error which is the difference between desired value (target) and the predicted value, can be obtained by [17]:

$$\varepsilon = y - f(\boldsymbol{x}) \qquad (1.10)$$

For an incorrect output, this error is non-zero. Network should be able to learn from these mistakes. The training procedure is an iterative algorithm based on the prediction error, trying to minimize the error by modifying weights after each training sample following the so-called perceptron rule.
While perceptron rule is only applicable with sign activation functions, delta rule is applicable for perceptron network with generic activation functions like sigmoid. Delta rule performs minimization of the mean squared error between the target and predicted value. The error function, also called loss/cost function, can be seen in equation (1.11):

$$E(\boldsymbol{w}) = \frac{1}{2}\sum_{t=1}^{l}(y_{predicted}^{t} - y_{target}^{t})^2 \qquad (1.11)$$

Where l is the number of training samples and t is a training cycle (epoch) index. Weights are adjusted in the opposite direction of the gradient of error function after each iteration (gradient descent method) [30]:

$$\boldsymbol{w}_{t+1} = \boldsymbol{w}_t - \eta\frac{\partial E(\boldsymbol{w})}{\partial \boldsymbol{w}} \qquad (1.12)$$

The updated value of the weight is w_{t+1}, the current value is w_t and η is the gain which controlls convergance rate (learning rate) and it's common range is between 0.1 to 1.0 [30].

Multilayer perceptron (MLP): As mentioned before, the single perceptron was only applicable for linearly separable populations. However, a multilayer perceptron with a nonlinear activation function can be employed to solve the nonlinearly separable problems [30]. The most common non-linear activation functions are sigmoid (also called logistic function), hyperbolic tangent, Softmax (only for output layer) and rectified linear unit (ReLU). A multilayer perceptron consists of an input layer, an output layer and one or more layers in between, called hidden layers. In Figure 1-11 a three-layer network with one hidden layer can be seen.

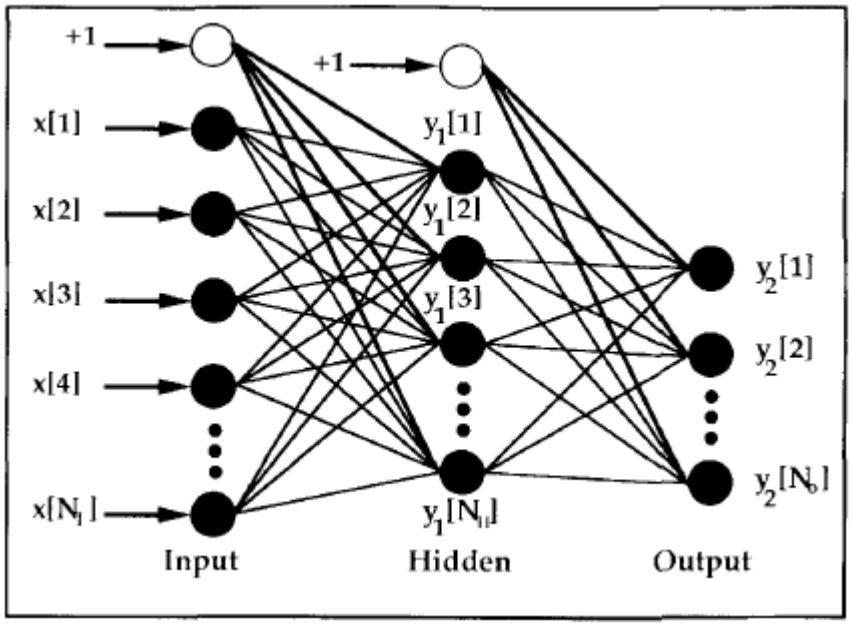

Figure 1-11 Multilayer network [30]

The learning rule for MLP network is called *backpropagation* which permits to train networks with more than one hidden layers. As always, the goal of learning is to minimize the total loss function. Calculating the error in output nodes is feasible since the target value is given. However, in the case of hidden nodes, there is no defined desired value, hence the error cannot be calculated [30].
Backpropagation has two steps at each iteration. First, given an input sample, the error between target value and output value is calculated. Second, layer by layer error is back-propagated working backwards from output layer through hidden layers, and to input layer following the backpropagation rule. At this point, the back-propagated errors can be used to adjust the weights in hidden and output layers. This process is repeated until the network learns to produce the correct output.

Softmax loss function: In case of multi-class problems (only one class can be chosen between all classes), the activation function of output nodes is usually Softmax. Softmax produces a number in range of (0,1) for each node's output, in a way that outputs of all nodes add up to 1. In these cases, instead of mean squared error, other types of loss functions are employed such as cross-entropy. Categorical cross-entropy (also called Softmax loss) is a cross-entropy loss function applied to the output of Softmax activation, as shown in Figure 1-12. [37].

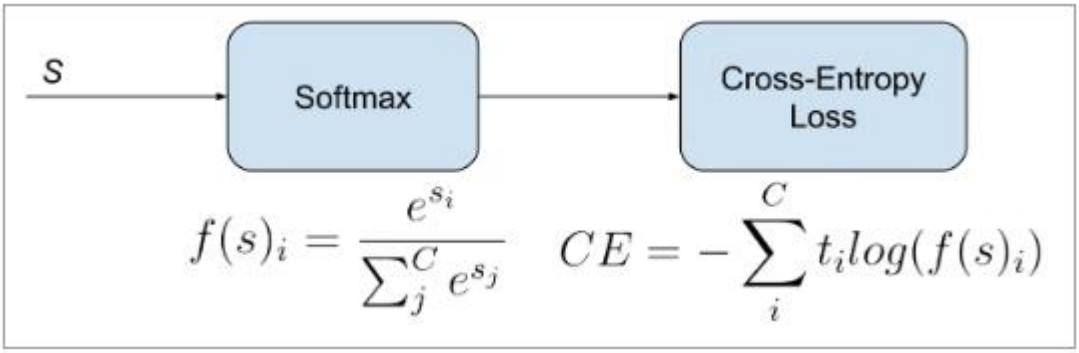

Figure 1-12 categorical cross-entropy is Softmax plus cross-entropy [37]

Considering having C number of classes (equal to number of output nodes), s is a vector with C dimensions and is the output of network. The target vector is presented as t. If only one class is supposed to be chosen between C classes, target vector will only have one non-zero element for $i = p$. In this way, the cross-entropy (CE) can be re-written as in (1.13) [37]:

$$CE = -\log(\frac{e^{sp}}{\sum_j^C e^{sj}}) \qquad (1.13)$$

Optimization: In order to minimize the total loss function, an optimizer must define how to update the network's parameters. Usually with Softmax loss, some type of stochastic gradient descent (SGD) is employed [38]. SGD uses the gradient of loss function to update the weights, inspired by perceptron rule. Adam is similar to SGD in the sense that it has stochastic nature, but it can automatically update parameters based on adaptive estimates of lower-order moments [21, 38]. Other recently introduced optimizers are RMSprop and AdaGrad, the details of which does not fit in the context of this work.

1.4.5 Convolutional neural network

Convolutional neural networks (CNNs) are a type of ANNs but in contrast to fully connected networks, they are sparsely connected. CNNs are specially interesting for researchers because they don't require prior feature engineering due to the fact that they have feature learning capability embedded in their architecture [39].
The architecture of CNN was inspired by biological knowledge of human's visual system. Specifically, findings about alternating layers of simple and complex cells by Hubel and Wiesel [40] motivated CNN's architecture. Visual system's structure is evolved in a way that they encode the visual pattern layer by layer. The extracted features gradually become more specific layer by layer. The same happens in a CNN.
In general, CNN architecture consist of blocks of convolutional and pooling layers, followed by one or more fully connected layers [38]. In Figure 1-13 a general CNN architecture with one convolutional block and 3 fully connected layers can be seen.

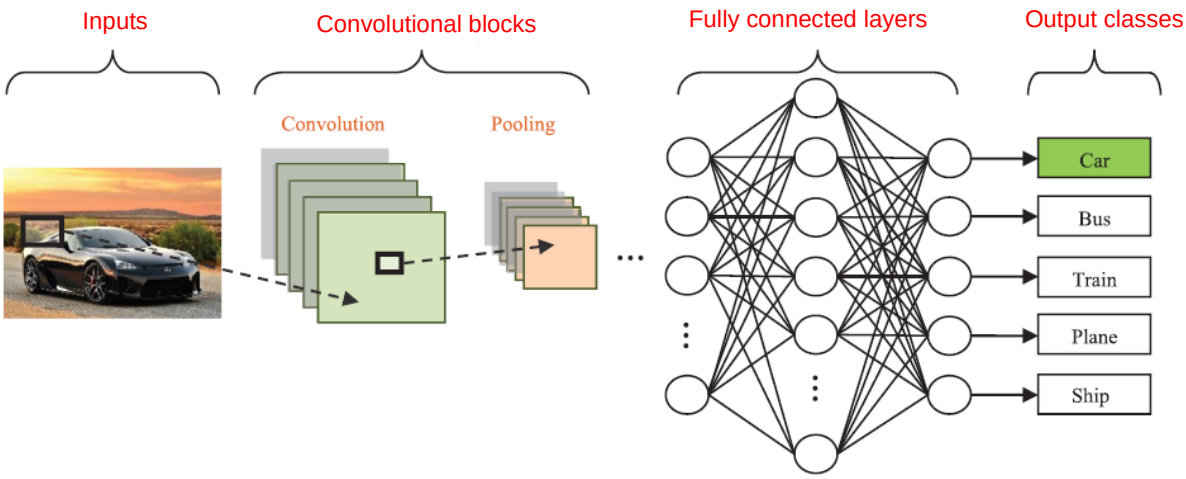

Figure 1-13 General CNN architecture [38]

Convolutional layer: The convolutional layers can be regarded as feature extractors. The goal is to calculate a convolution between a set of filters and input data. One convolutional layer has multiple learnable filters (n) (All the weights in a single filter are regarded as parameters to be learned). Every filter has limited extend in width and height but extends through full depth of input. For example, if the size of input data is 218×218×3 (3 is red, green and blue colors in images) one filter size could be 5×5×3. The width and height size of filter is also called receptive field which in this example is 5x5. As shown in Figure 1-14, a filter scans the whole input by sliding in width and height dimensions according to a predefined stride [39].

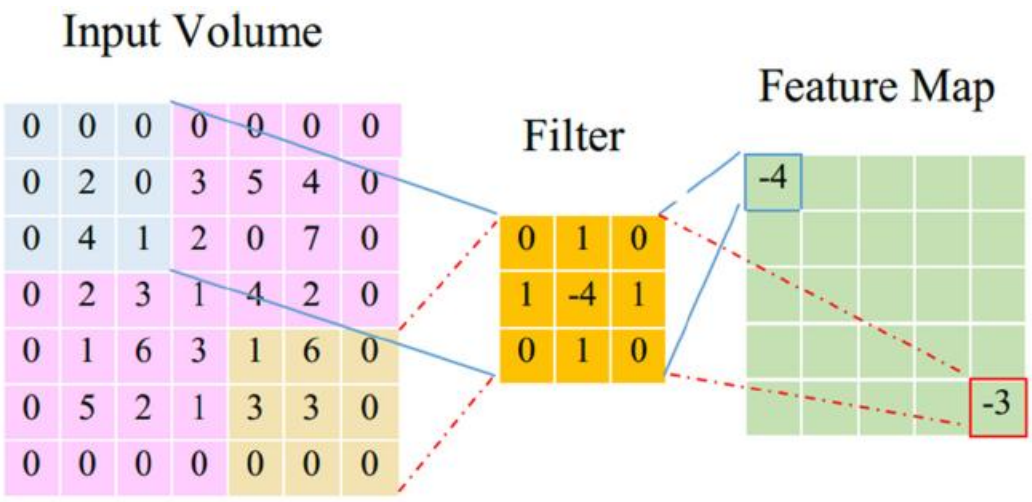

Figure 1-14 Convolution of a 2-D input data with a (3×3) filter [39]

The results of convolution between filter and input data will be given to a nonlinear activation function to form the non-linear feature map. This procedure is formulated as in (1.14) [38]:

$$Y_k = f(W_k * x) \qquad (1.14)$$

Input is shown as x, Y_k is the kth feature map, W_k is the weights of the kth filter and $f(\cdot)$ is the non-linear activation function. The operator $*$ denotes the convolution operation.
Filter weights remain the same when sliding the kernel on the input image; however, different filters in a convolutional layer assume different weights [38]. The generated feature maps from n filters will be appended to produce the output of that convolutional layer [39].

Pooling layer: In order to reduce sensitivity to input distortions and translations, and to reduce spatial resolution of feature maps, a pooling layer can be employed after each convolution layer. This can be considered as a subsampling in the number of neurons and is performed by averaging or taking the maximum of adjacent neurons. The latter is called max pooling and outputs the largest element in each receptive field, such that [38]:

$$Y_{kij} = \max_{(p.q)\in R_{ij}} x_{kpq} \qquad (1.15)$$

Where region R_{ij} is the receptive field in the kth feature map, $(p.q)$ is a location contained in that region and x_{kpq} denotes the element at that location. Y_{kij} would be the output of max pooling of kth feature map.

Fully connected layers: After multiple blocks of convolution and pooling, fully connected layers interpret the extracted features and perform high-level reasoning. At the final layer, typically a Softmax activation function is used [38]. Only when using Softmax layer number of neurons should match the number of output classes.

1.4.6 Transfer learning

Researchers often face a common challenge when using deep learning. Training a deep network with insufficient training data could produce very poor results, even lower than classical classification methods with handcrafted features [41]. This problem becomes more severe in the cases that acquiring more samples is costly in terms of both time and money. Transfer learning (TL) techniques that apply knowledge learnt from one task to other related tasks have been proven helpful in these situations [42]. As humans have the ability to transfer knowledge from one domain to another (e.g. learning violin from the knowledge gained while learning piano), TL aims to leverage knowledge from a related domain (called source domain) to improve the training performance or minimize the number of training samples required in a target domain [43].

1.4.7 Overfitting

Overfitting is one of the common problems in deep networks which occurs when the network is trained by small training set. In this situation, network can't generalize its performance on unseen data and the resulting accuracy on test dataset decreases radically [38]. This issue is addressed by employing regularization techniques such as adding drop out layer, batch normalization layer and early stopping which will be explained in chapter 2.

1.5 State of the art

SEMG-based hand gesture recognition was first introduced by Hudgins [14]. Hudgins used time domain (TD) features such as Mean Absolute Value (MAV), Mean Absolute Value Slope (MAVS), Zero Crossing (ZC), Slope Sign Change (SSC) and Waveform Length (WL) as input to an ANN classifier. In an attempt to find more effective features, Englehart [15] compared the performance of Hudgins' TD features, and those derived from short-time Fourier transform (STFT), wavelet transform (WT), and wavelet packet transform (WPT). He came to the conclusion that the combination of WPT and PCA dimension reduction, given to LDA classifier outperforms other combinations of feature sets and classifiers. Phinyomark et al. [16] investigated the classification result of LDA, using single or multiple time-domain and frequency-domain features. When comparing single features, sample entropy (SampEn) was chosen as the best feature among all 50 considered features. Further features and further classifiers such as Random Forest (RF), K-Nearest Neighbors (k-NN), LDA and SVMs were studied by Atzori et al. [6] who announced that random forest displays the best performance when fed with all the features used in his study. Discovering the best feature set remains an open problem and researchers have tackled this issue evaluating diverse combination of features [4, 16, 44].

With the outburst of deep learning studies in image classification and speech recognition, scientists became interested to apply such networks in sEMG classification. The first time that raw EMG signals (for hand movement recognition) were classified by a deep convolutional neural network (CNN) can be considered when Park and Lee [45] tried to resolve inter-subject variability on Ninapro database with CNN in 2016 and concluded it outperformed SVM by 12%–18%. Indeed, a survey on deep learning techniques for sEMG-based hand gesture classification published in 2019 (considering the papers from 2014) [46], states that the most widely used methods were CNN [4], Auto-encoder (AE) [47], Recurrent Neural Network (RNN) [48] and Deep Belief Network (DBN) [49], respectively. It was also shown in [50] that in terms of robustness over time, CNN shows better performance comparing LDA with handcrafted features.

Over the last few years, many novel deep network architectures were designed based on CNN, aiming to improve its performance. Residual Neural Network (ResNet) [22] is a variation of ANN in which skip connections are used to amplify the effects of deeper layers. The main contribution of such model was to resolve the vanishing gradient problem in very deep networks. The idea of using skip connections can also be applied to CNNs to create a Residual Convolutional Neural Network. To the best of our knowledge this kind of network is rarely employed in sEMG classification. ResNet could potentially pave the way to explore deeper networks in this field.

In the context of bio-signal classification, deep learning is certainly not the only technique borrowed from image classification. Transfer learning has gained increasing interest as an approach to solve a common issue in bio-signal classification. In contrast to image classifiers, bio-signal classifiers are confined to single session recordings as training data, due to high session variability which limits the size of dataset. Recently, TL has been employed to leverage recordings from other sessions or even other subjects to train the deep network for the targeted subject [51, 52, 5]. Du et al. [51] proposed an unsupervised deep domain adaptation (adaptive batch normalization) for inter-session and inter-subject gesture recognition using high-density EMG recordings. Furthermore, to target inter-subject continues gesture recognition [52] employed a RNN with adversarial domain

adaptation. In another study, [5] used an element-wise summation methodology on a CNN architecture to combine source network (trained on all participants) and a second network to generate the target network for each subject. Inspired by the mentioned studies, two TL approaches are introduced in this work.

To compare the performance of different techniques proposed by researchers around the world, the need for a publicly available benchmark dataset was always evident. Hence, Atzori in 2014 published a dataset including several hand and wrist movements from 78 subjects, called “Ninapro”, which was divided into three different subsets based on acquisition system and characteristics of subjects [6]. DataBase 2 (DB2) acquired by 12 Delsys electrodes, includes three sets of different exercises, from which exercise B consists of basic wrist movements and isotonic hand configuration. From that point on, this database has been used by many researchers as a benchmark. Zhai proposed a self-recalibrating CNN to eliminate the need of user training over time. The classification accuracy of his method on Ninapro DB2 was 82.22% when tested on exercise B [8]. Moreover, in 2019 Huang used a CNN-LSTM network in order to fully capture the spatial and temporal features of sEMG spectrograms, the resulting accuracy on DB2 exercise B was 81% [9]. To the knowledge of authors this is the best accuracy obtained on this database. The same database is targeted as a benchmark in this work.

2 Materials and methods

2.1 Nearlab Dataset

As mentioned in section 1.2, a custom hand gesture EMG dataset was acquired to test the proposed methods.

2.1.1 Participants

The dataset includes 11 healthy subjects including 6 male participants and 5 female participants (age 25±3 years). The only inclusion criteria were the absence of history of neuro-muscular disorders. The data acquisition protocol was approved by the research ethical committee of Politecnico di Milano, on October 16th, 2019. All participants had been briefed about the experiments and gave informed consent.
Subjects were asked to answer the questions of a questionnaire. The questionnaire contained questions about medical history of the participants that could affect the outcome of the experiments.

2.1.2 Protocol definition

Eight movement classes have been targeted in this thesis work. Movement classes have been selected to include the most frequent upper limb movements performed in daily life. According to Vergara et al. [53] pinch, cylindrical and lateral pinch are among the most frequent handgrips in daily life. Therefore, “pinch” and “lateral pinch” were included as two movement classes. Whereas, cylindrical grip was replaced with closed palm or “fist”. The fourth hand movement was “hand open”. “Hand open” is characterized by separated extended fingers as opposed to rest, where the hand is relaxed. Moreover, wrist flexion, extension, pronation and supination have been included. The 8 movement classes have been depicted in Figure 2-1 and listed below:
1) hand flexion, 2) hand extension, 3) wrist supination, 4) wrist pronation, 5) hand open, 6) pinch, 7) lateral pinch and 8) fist.

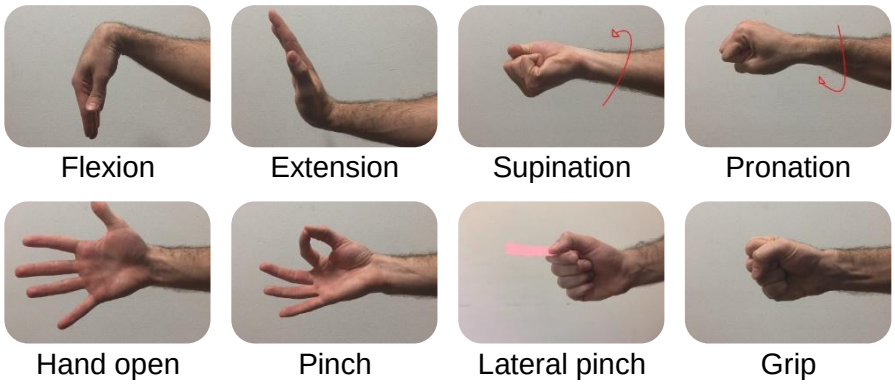

Figure 2-1 Movement classes selected for experimental protocol

In order to create a more general dataset leading to a robust pattern recognizer, subjects were required to repeat selected hand movements three times (i.e., three round), characterized by three different hand starting positions (Figure 2-2):

- Round 1: upward starting position, where the palm is faced upward;
- Round 2: sideway starting position, where the palm is faced sideway;

- Round 3: downward starting position, where the hand palm is faced to ground.

Palm sideway Palm upward Palm downward

Figure 2-2 Hand open movement in 3 predefined hand orientations

A video was created and displayed to the participants as a video cue to instruct them to perform the expected tasks in the correct time. No audio cues were used in this process.
As mentioned before, the dataset is composed of 8 basic movements. In a single experimental session, the 8 basic movements are repeated 5 times in each starting hand orientation (8 movements*5 repetitions*3 hand starting positions=120 total movements). Subjects are instructed to hold each movement for 5 seconds. The movements' order in the experiment is randomized and unknown to the participants. Before each movement instruction, participants have 3 second of rest, followed by the preparation time (2 seconds) where the subject is informed about the next movement, but is instructed to stay relaxed until the execution cue is shown.
The 5 second rest periods followed by 5 second movements are called "movement blocks" (Figure 2-3).

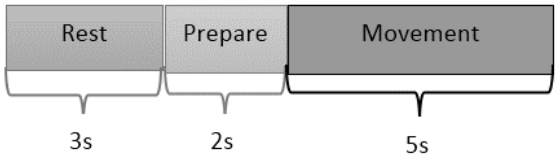

Figure 2-3 One movement block (10 seconds in total)

8 movement blocks form a "trial". A "round" consists of 5 "trials", separated by 15 second free periods. The 3 aforementioned "rounds" are separated by 100 seconds periods. The protocol related to basic movements is demonstrated in Figure 2-4.

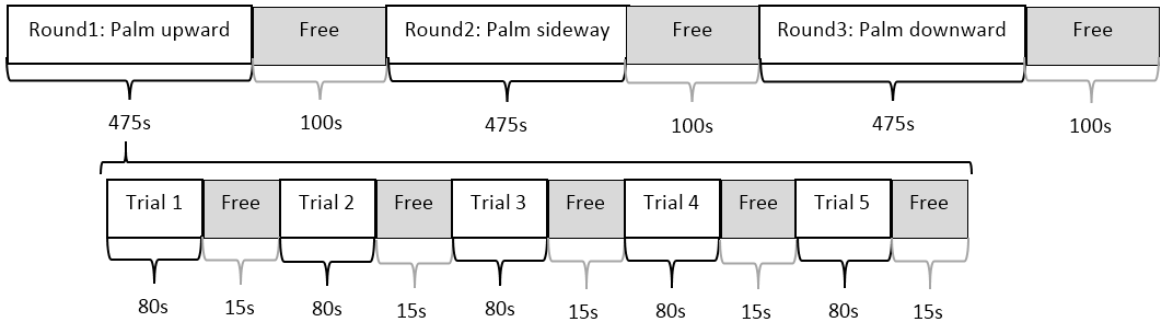

Figure 2-4 Construction of whole protocol for basic movements: Basic movement protocol is divided in 3 rounds and each round consists of 5 trial

In order to prevent muscle fatigue as much as possible, rest and free periods are carefully arranged in between movements. In the free periods, subjects are free to do anything and their signals will not be used for offline analysis. In contrast, in rest periods that are crucial for the data analysis steps, the participants are instructed to relax their muscle. Furthermore, subjects were asked to use a normal and constant speed to reach to the final pose of the movement to their best of ability.
The whole video shown to participants lasts 34 minutes.

The total arrangement of the experiment is illustrated in Figure 2-5.

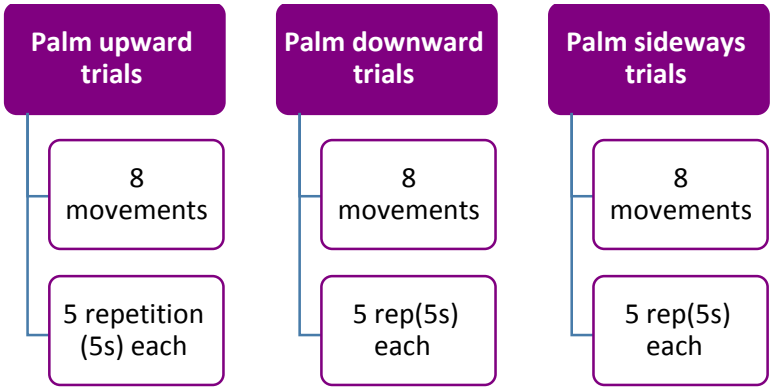

Figure 2-5 Trial time organization

Synchronization: One of the most important aspects of the experiment is the synchronization of the acquired signal and the time tags of the movement execution cues. This synchronization is crucial to the process of labeling the input signals. In order to do so, a trigger input of the acquisition system is utilized. A micro controller board (Arduino board) is used to send a pulse to the trigger channel upon receiving the instruction from PC through serial connection. The computer will start the video and send the serial command at the same time. Thus, the time of starting the video is tagged in the acquired signal by a pulse in the trigger channel.

Observation process: an observer was always monitoring the experiments. His/her role was to check the correctness of the movements (according to video cue) and record it for future steps of preparing the database. The main reason for such an observation is that, in some cases, the participants confused the movements. The most frequent example is executing pronation as opposed to supination or vice versa. In these cases, the actual executed movement is detected by the observer and recorded. Incorrect movements are discarded based on these reports from observations at the end of each experimental session, prior to any processing.

2.1.3 Hardware and software

The experimental setup was composed by a screen for visual cue display, an EMG acquisition system, a laptop that receives the acquired signal via USB cable, disposable gel-based electrodes and synchronization circuitry. The EMG acquisition system used in this project was Porti from TMSi company [10]. The device had 32 input channels, 16 of which were bipolar inputs. In addition to the 32 channels it had a trigger input, which is used in this project for synchronization. The cables used to

connect electrodes to the device were equipped with active shielding which significantly increased signal to noise ratio.
The electrodes were passive EMG Ag/AgCl electrodes with conductive gel inside them. 10 differential channels were employed in this study. The signal was sampled at the rate of 2048 Hz.
A Matlab interface was used both to visualize the acquired signals in real-time and to store the data in the PC. The synchronization code was also coded in Matlab.

2.1.4 Electrode placement

10 bipolar channels are used for signal acquisition. The general purpose of electrode placement was both to consider muscle anatomies and achieve simplicity in placement. Regarding muscle anatomy, the aim was to record from highest number of muscles while avoiding places with highest probability of cross-talk. The exact positions of electrodes were determined according to SENIAM (Surface EMG for non-invasive assessment of muscles) [11]. The final configuration of electrodes is as follows: 6 electrode pairs (corresponding to 6 channels) are placed around the upper forearm equally-spaced along the forearm circumference. Each differential pair is arranged along the length of the arm with 2cm distance from each other. The first electrode pair is placed 3 cm distal to the elbow (medial epicondyle), other 5 are arranged to have same distance with respect to each other using the measured forearm circumference. The 4 remaining electrode pairs are placed 3 cm distal to the previous electrodes. All these electrodes were placed on the dominant hand of the participant. Electrodes' spatial distribution on the forearm is displayed in Figure 2-6. Reference electrode is placed on the back of the wrist as suggested by SENIAM directions [11]. Figure 2-7 depicts the hand stance prepared for the experiments.

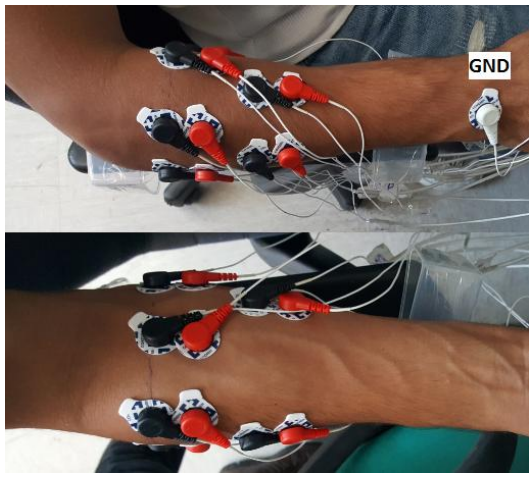

Figure 2-6 Electrode positions

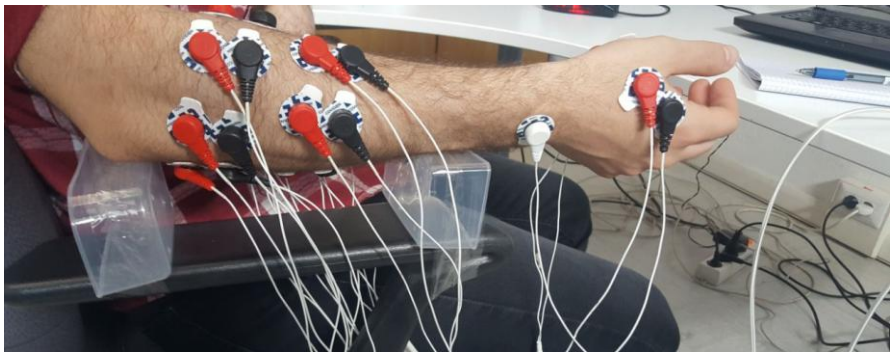

Figure 2-7 Hand stance

In order to properly prepare for signal acquisition, the relevant areas of the skin are cleaned twice with alcohol and cotton before placing the electrodes on the skin. Skin preparation and electrode placement procedures took 20-25 minutes.

2.1.5 Database creation

Recorded signals contain 10 channel of EMG data sampled at 2048 Hz and 1 trigger channel which marks the start of the video cue movie. The recorded data, video cue movie and the report filled by the observer form the custom-made dataset referred to as “Nearlab dataset”.

2.2 Ninapro Dataset

Moreover, to have comparable results with state of the art, a well-known publicly available dataset (Ninapro) is also be employed. Ninapro dataset is one of the largest and most well-known publicly available sEMG datasets covering hand and finger movements and it is used in many studies as a common benchmark [6]. In this thesis work, DataBase2 (DB2) consisting of 40 able-body participants from Ninapro datasets is used. DB2 is collected using 12 active double–differential wireless electrodes with a Delsys Trigno Wireless EMG system [12], which has 2kHz sample rate. This database includes 3 sets of exercises. The first exercise, which includes 17 basic movements of fingers and wrist, is targeted in this thesis, due to its similarity to movement classes in Nearlab dataset. Each movement, which lasts for 5 seconds and is followed by a rest period of 3 seconds, is repeated 6 times.

2.3 Data analysis

This part of the document is dedicated to the data processing technics used in the project.

2.3.1 Data pre-processing

Up to this point the raw EMG signal is obtained and is ready for processing. Figure 2-8 displays an example of raw EMG signal of one of the channels for the entire duration of experiment (~35 minutes).

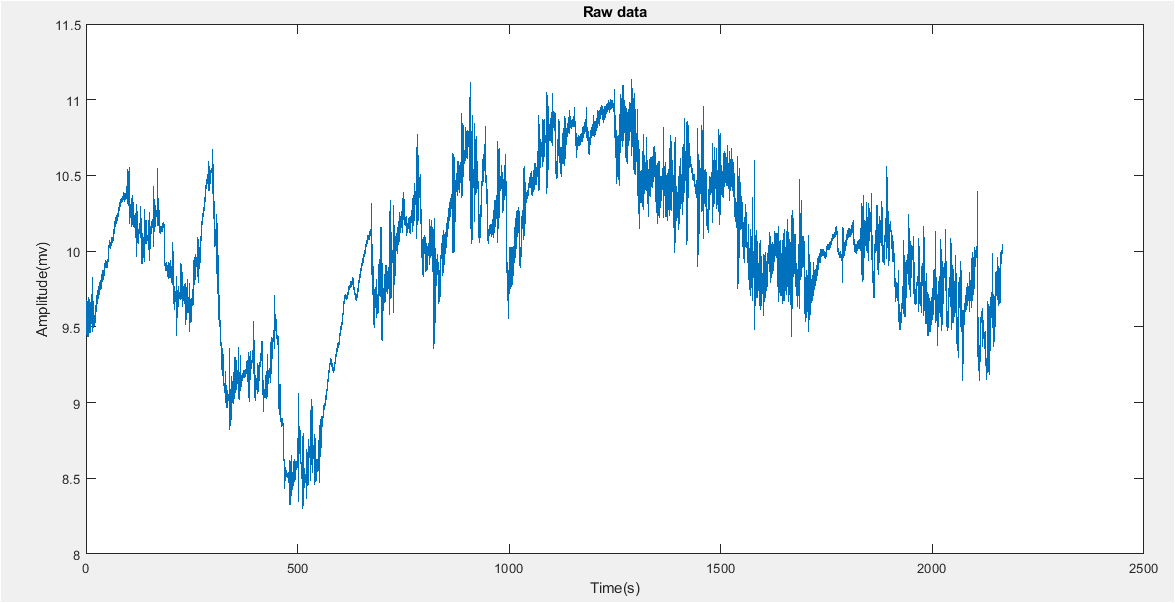

Figure 2-8 Raw EMG signal acquired by one channel during the entire experiment session

It can be noticed from the Figure 2-8 that signal acquired with Porti system is highly attenuated by noise. Therefore, the raw signal is filtered using a band pass filter and a notch filter, in order to remove unwanted signals and power line interference [13]. The specifications of the filters are shown in table 2-1.

Table 2-1 Preprocessing filter specifications

	Band-pass filter	Notch filter
Filter type	Butterworth	Butterworth
Filter order	4	2
Cut-off frequency 1	10	49
Cut-off frequency 2	500	51

Figure 2-9 illustrates the raw and filtered data in the same graph. The three main activity periods in the experiment (movements with upward, sideways and down ward starting orientations) can be distinguished in the Figure 2-9 filtered data.

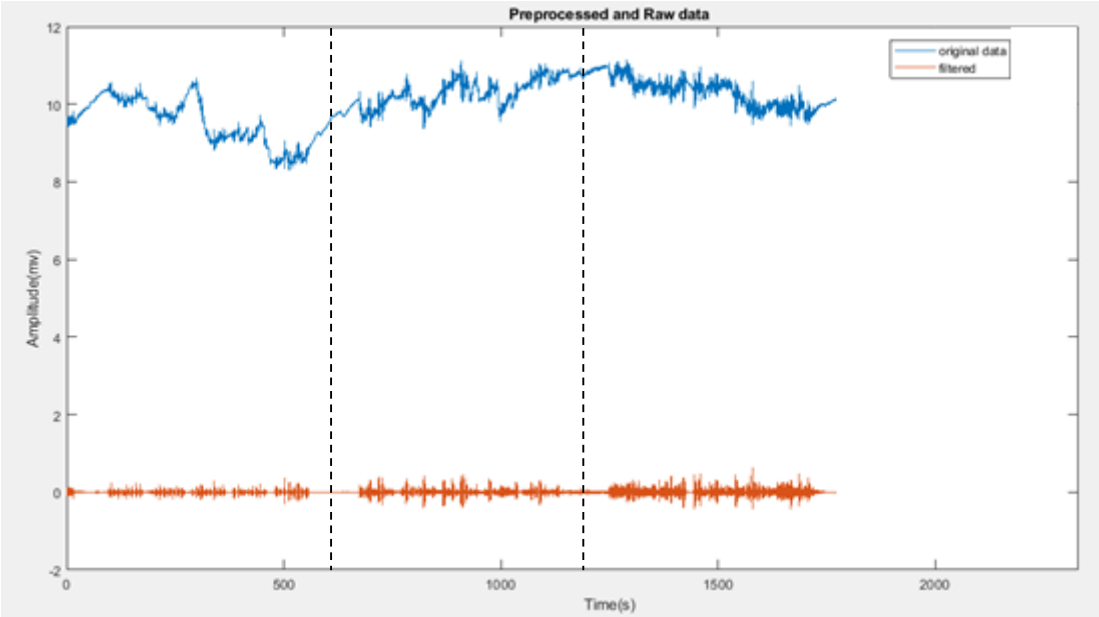

Figure 2-9 Filtered data vs raw data

Figure 2-10 demonstrates the frequency content of the signal before (top) and after (down) filtering. It can be seen that the large low-frequency and 50 Hz component is removed.

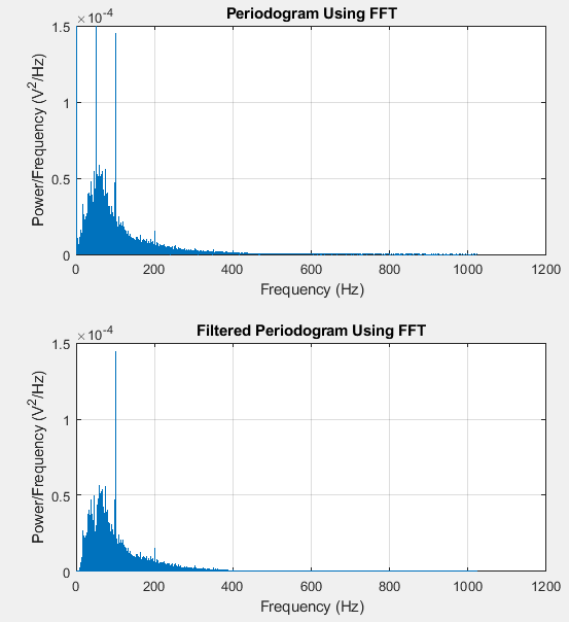

Figure 2-10 Frequency content of the signal before (top) and after (down) filtering

Next step is related to synchronization. Channel 13 which is the trigger input of the Porti is used to extract the time of the start of the video cue movie. The acquired data before the trigger pulse is removed in all 10 channels. The new signal completely coincides with the timings of the video cues in the movie. Thus, the signals could be labeled with their associated class of movements. One approach for labeling the data is to use to the video cue timings to label movement and rest periods. However, this algorithm will take for granted that movement execution is done immediately after the video cue, which is not realistic. Thus, a different

approach has been considered. As the first step, based on the video cues indicating start of each movement, windows with length of 7 seconds are extracted. The window will include roughly 5 seconds of movement and 2 seconds of rest. Then the window is further trimmed using a threshold-based on-set detection algorithm to extract the actual movement (around 5 seconds) from the 7 second window. This algorithm is depicted in figure 2-11. The on-set detection algorithms requires a threshold. This threshold is calculated from a rest period in the signal. The signal of all the channels are rectified in this period. Then a moving average is used to make the signals smoother. Next, the signal is averaged in time within the window and also among all channels in order to reach a single value referred to as the “Rest value”.

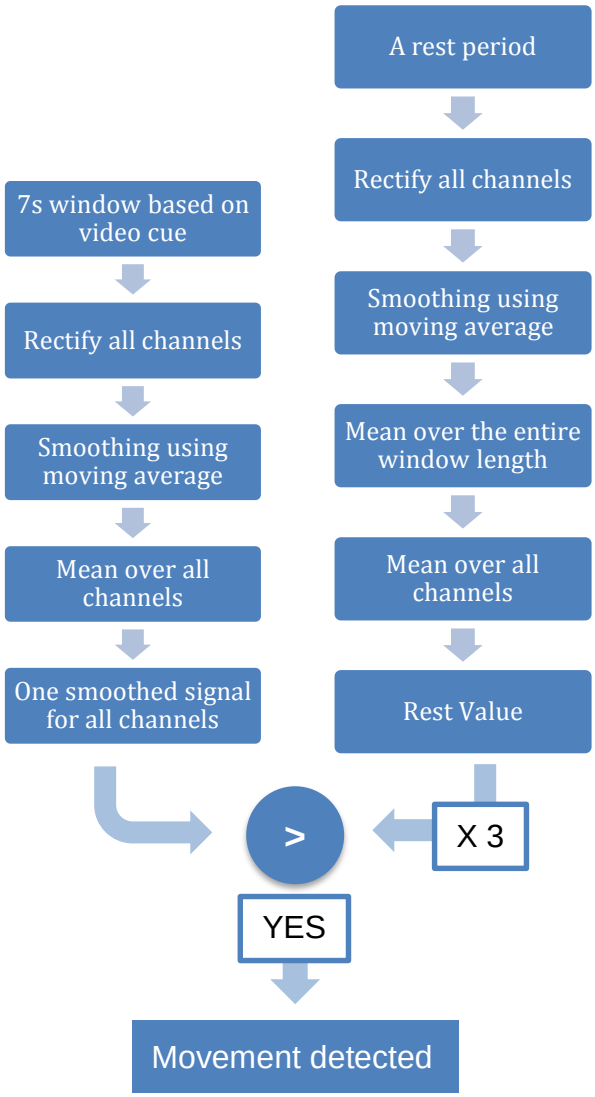

Figure 2-11 Diagram of on-set or ending detedction algorithm

With the rest value available, the on-set and ending detection algorithm use a threshold (3 times the rest value) to extract the actual movement from the 7 second window. This stage is necessary since the subject movement delay varies from each task to the next. Figure 2-12 displays the first channel’s 7 second windows throughout an entire round, super imposed on each other. Figure 2-13 displays the on-set and end detection process on the signals. Start and finish points are indicated with green triangles. The red envelope is the smoothed signal.

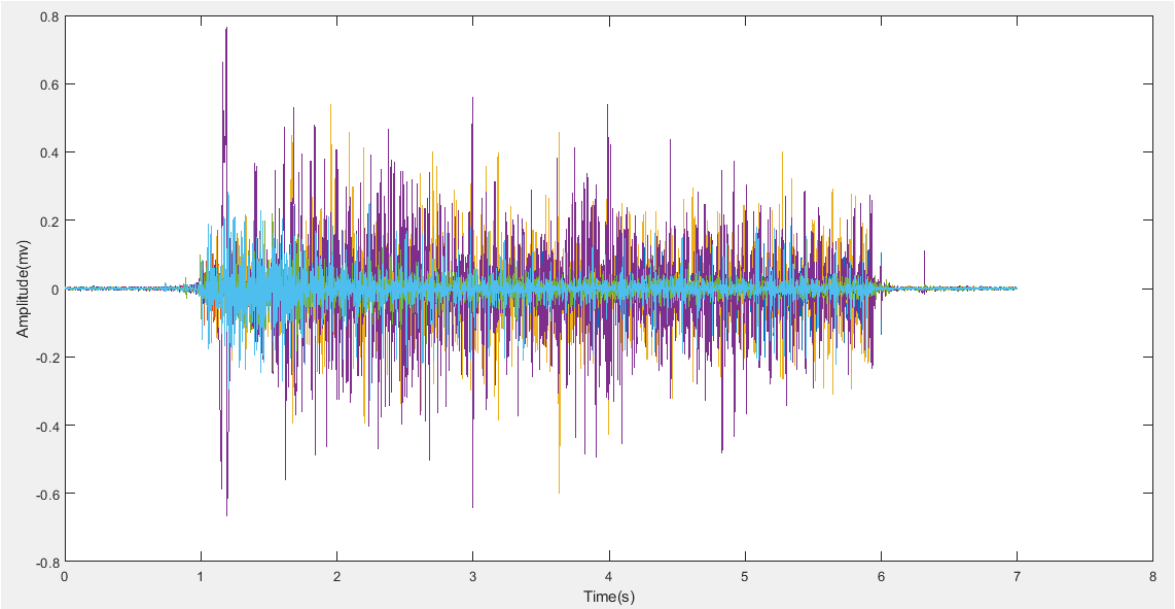

Figure 2-12 7 second windows super imposed from one channel in different movements

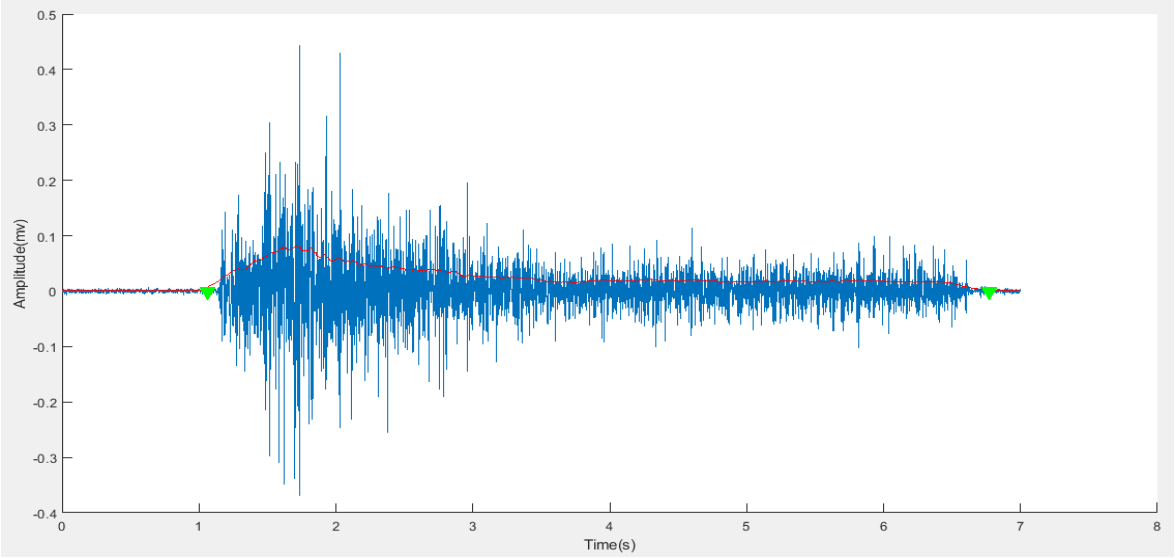

Figure 2-13 The process of on-set and ending detection

The last preprocessing step is to remove the first and the last 100ms of the movement to eliminate the transient part of the signal; since the main focus of the data analysis in this thesis is the stationary characteristics of the signal. This decision is derived from conclusions Englehart et al. made in [54] showing that steady-state data is classified more accurately than transient data. Figure 2-14 shows the final step. The red signal is removed due to on-set and ending detection while the orange part is removed to eliminate the transient part of the signal.

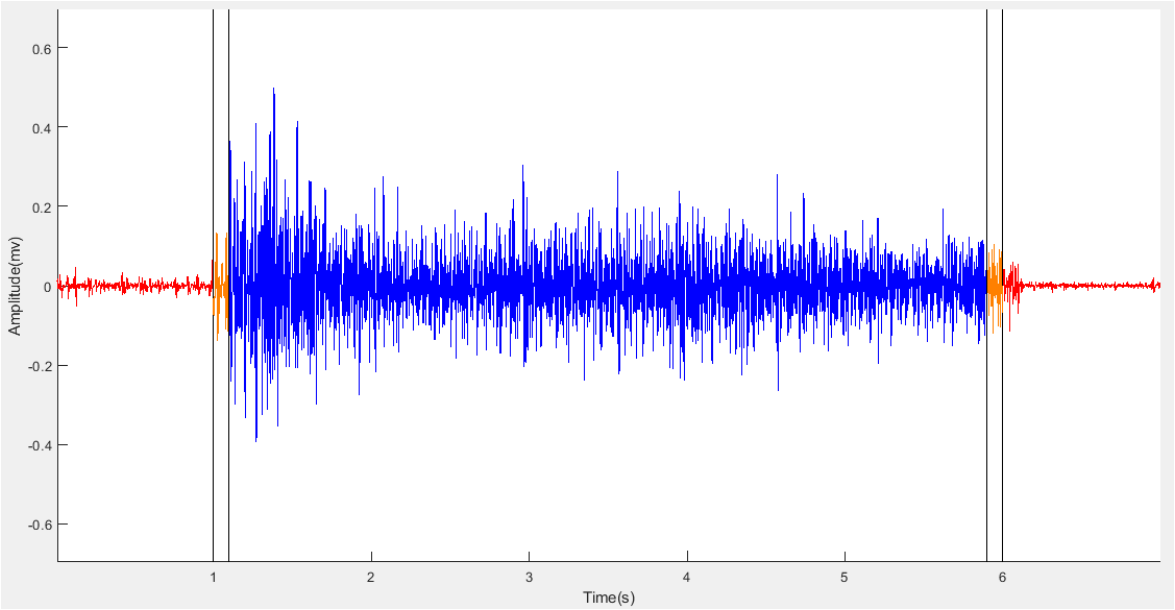

Figure 2-14 the process of on-set and ending detection and removing the transiant signal

This remaining window can be now labeled with its true movement class according to the video cues.

2.3.2 Data segmentation

Since one of the goals of this project is online classification, windows should become very small to allow for fast classification in real-time application. Considering that for online applications window length plus processing time to generate classified control commands should be less than 300ms [1], window size of 250ms (512 samples) was selected for this project.
Data augmentation is a necessary step towards increasing the database size in order to be used in deep learning methods. As suggested by authors of [5], sliding window approach is the most effective augmentation technique for sEMG classification. Moreover, in order to allow for majority voting method, it was decided that the sliding windows method should be used in this study as well. Majority voting could assist in achieving a smooth online classification system by eliminating scarce errors. In this method, instead of dividing the movements in to small windows (in our case with the size of 250ms), we produce overlapping windows with strides smaller than the window size (in this case 62.5ms). Having 250ms window and 62.5ms strides will result in 187.5ms overlap between two consecutive windows.
In an online classification system, utilizing this method means that instead of classifying 250ms windows of movements we are classifying each 62.5ms of movement, which will cause more frequent predictions resulting in a smoother control. The predicted class of each 62.5ms will be calculated using majority voting on 4 predicted values for 250ms windows which include that specific 62.5ms interval. Figure 2-15 displays an instance of majority voting to determine the prediction of the 62.6ms window.

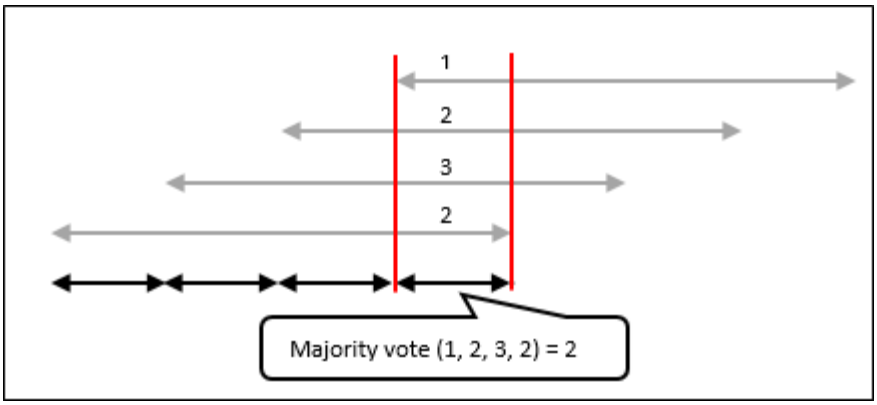

Figure 2-15 An example of majority voting method for online classification

At the end of data segmentation, the database will consist of labeled signal windows with 512 samples per channel for 10 channels representing 250ms of EMG signals.

2.3.3 Classical machine learning

In this section, the following traditional classification methods will be used to classify the EMG signal: Support Vector Machine (SVM), Linear Discriminant Analysis (LDA), K-Nearest Neighbors (KNN) and Multi-Layer Perceptron (MLP). These classifiers require features extracted from the raw signal (250ms windows).
Based on the recent literature available on feature selection for sEMG classification [14, 15, 16] and datasets exploration, 15 features of time and frequency domains are used to create 4 feature sets as inputs of classifiers. Each feature should be calculated for all the channels separately.

- Mean Absolute Value: This feature is the average of the rectified signal.
- Zero Crossing: number of times the signal crosses the zero line.
- Slope Sign change: number of times that the signal slope changes, detected in signal derivative
- Waveform Length (WL): This is a feature that offers a simple characterization of the signal's waveform. It is calculated as in equation (2-1), where $x_{i,k}$ is kth data point in the ith channel and L is the number of data points.

$$WL(x_i) = \sum_{k=1}^{L} |x_{i.k} - x_{i.k-1}| \qquad (2\text{-}1)$$

- Hjorth Parameters (HP_m, HP_M, HP_C): Hjorth Parameters are indicators of statistical properties used in signal processing in the time domain, introduced by Bo Hjorth in 1970 [55]. They are commonly used in the analysis of electroencephalography signals for feature extraction, improving dataset quality. These parameters are Activity, Mobility and Complexity and are calculated as in equations (2-2), (2-3) and (2-4).

$$Activity(x_i) = \frac{1}{L}\sum_{k=1}^{L}(x_{i.k} - \overline{x_i})^2 \qquad (2\text{-}2)$$

$$Mobility(x_i) = \sqrt{\frac{Activity(\dot{x}_i)}{Activity(x_i)}} \qquad (2\text{-}3)$$

$$Complexity(x_i) = \frac{Mobilty(\dot{x_i})}{Mobility(x_i)} \quad (2\text{-}4)$$

Where $x_{i,k}$ is kth data point in the ith channel and L is the number of data points.

- Sample Entropy (SampEn): This feature is a measure of orderliness/randomness of the signal. SampEn is a method which estimates entropy of the signal and is calculated as in equation (2-5).

$$SampEn(x_i.m.r) = -\ln\left(\frac{A^m(r)}{B^m(r)}\right) \quad (2\text{-}5)$$

Where x_i is the ith channel signal, m is the embedding dimension and r is the tolerance.

- Cepstral Coefficient (CC): The Cepstrum of a signal is the inverse Fourier transform of the log power spectrum magnitude of the signal. The coefficients of the Cepstral Coefficients are employed as features. In our case we used the forth order CC features. They can be directly derived from Auto-Regression (AR) coefficients as in equations (2-6) and (2-7).

$$C1 = -a1 \quad (2\text{-}6)$$
$$C_i = a_i - \sum_{n=1}^{i-1}\left(1 - \frac{n}{i}\right)a_n C_{i-n} \ .with\ 1 < i \leq P \quad (2\text{-}7)$$

Where C_i is the ith Cepstral coefficient, a_i is the ith auto-regression coefficient and P is the order.

- Root Mean Square (RMS): This feature, also known as the quadratic mean, is calculated as in equation (2-8).

$$RMS(x_i) = \sqrt{\frac{1}{L}\sum_{k=1}^{L} x_{i.k}^2} \quad (2\text{-}8)$$

Where $x_{i,k}$ is kth data point in the ith channel and L is the number of data points.

- Integrated EMG (IEMG): This feature will represent the sum of fully rectified signal and is calculated as in equation (2-9).

$$IEMG(x_i) = \sum_{k=1}^{L}|x_{i.k}| \quad (2\text{-}9)$$

Where $x_{i,k}$ is kth data point in the ith channel and L is the number of data points.

- Skewness (SKEW): The Skewness is the third central moment of a distribution, which measures the overall asymmetry of a distribution. It is calculated as in equation (2-10).

$$SKEW(x_i) = \frac{1}{L}\sum_{k=1}^{L}\left(\frac{x_{i.k} - \overline{x}_i}{\sigma}\right)^3 \quad (2\text{-}10)$$

Where $x_{i,k}$ is kth data point in the ith channel and L is the number of data points.

In order to visualize the effectiveness of the features as a mean to distinguish different classes, MAV feature is depicted for 3 movements, as a way of example. Figures 2-17, 2-18 and 2-19 present the MAV feature calculated for all 10 channels in the 250ms windows extracted from pronation, supination and hand open movement, respectively. For each 250ms window, MAV feature of all 10 channels are calculated and connected in this plot, hence each curve belongs to one 250ms window. All the features of 250ms windows from a movement are super imposed in a single plot, making several curves in one Figure. From these graphs, the patterns could be distinguished from one another in different movements. The x axis represents the channel, while the y axis represents the value of the features.

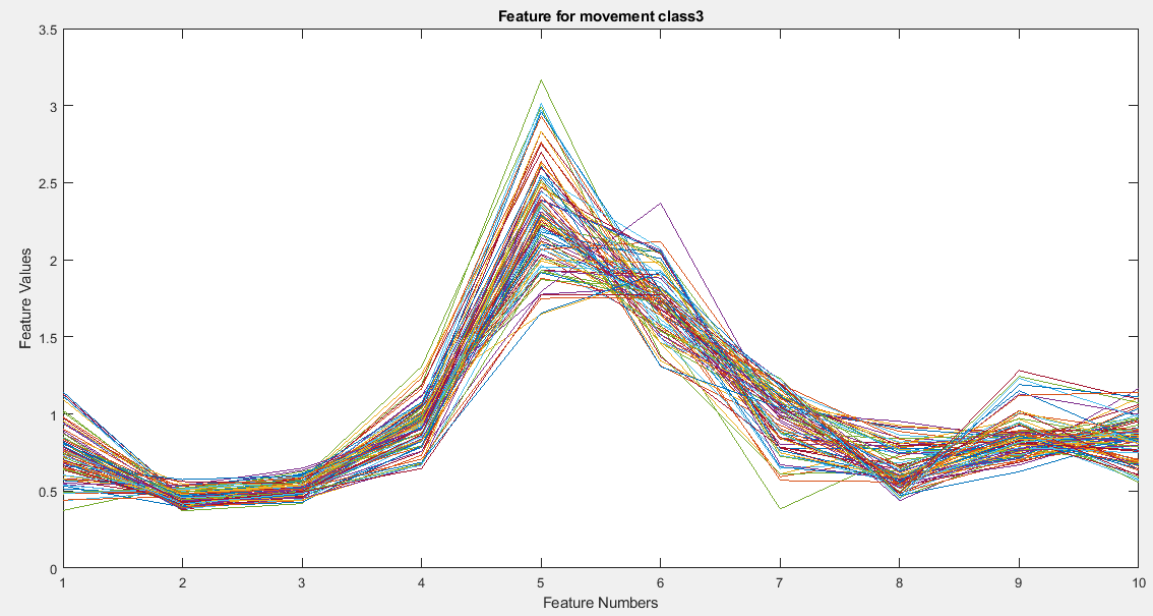

Figure 2-16 MAV features for 10 channels calculated for the multiple windows belonging to pronation movement

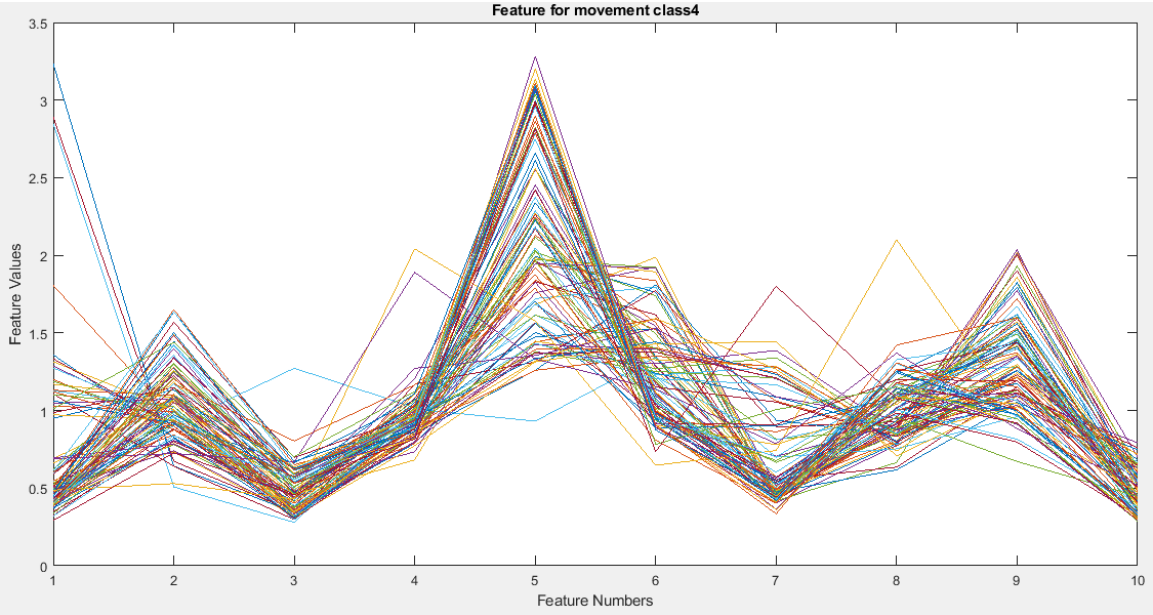

Figure 2-17 MAV features for 10 channels calculated for the multiple windows belonging to supination movement

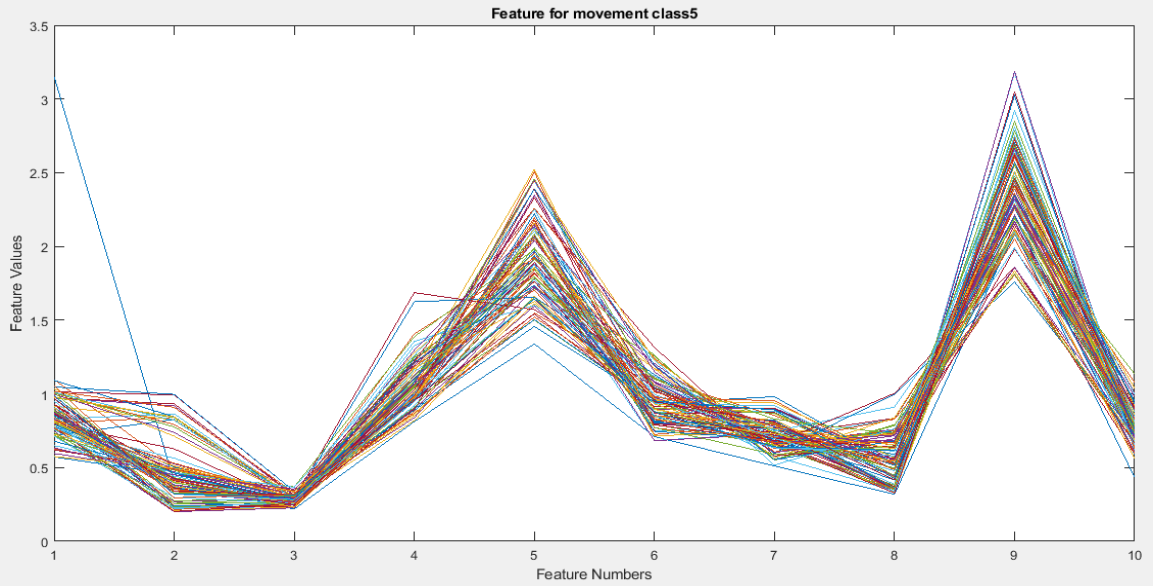

Figure 2-18 MAV features for 10 channels calculated for the multiple windows belonging to hand open movement

The 4 features sets designed for this study are introduced in table 2-2 and further explained bellow:

Table 2-2 Feature sets

Name	FEATURES	Number of features
Time Domain (TD)	MAV, ZC, SSC, WL	4
Improved Time Domain (ITD)	MAV, ZC, SSC, WL, RMS, IEMG, HP_A, HP_M, HP_C	9
Correlation Based (CB)	CC1, ZC, SSC, WL , HP_M, HP_C and SampEn	7
Full dataset (Full)	MAV, ZC, SSC, WL, HP_A, HP_M, HP_C, SampEn, CC1-4, RMS, IEMG, SKEW	15

MAV=Mean Absolute Value, ZC= Zero Crossing, SSC= Slope Sign Change, WL= Waveform Length, HP_A/HP_M/HP_C=Hjorth Parameters, SampEn=Sample Entropy, CC1-4=Cepstral Coefficient order 4, RMS= Root Mean Square, IEMG=Integrated EMG, SKEW= Skewness

- Time Domain (TD): This feature set consists of 4 well-known and often used time domain features.
- Improved Time Domain (ITD): This feature set consists of 9 features including previously mentioned TD features and 5 additional time domain features. These are particularly fast in calculation, making this feature set a good candidate for online classification.
- Correlation Based (CB): A set of 7 features is handpicked after investigating the cross-correlation of all features. In case of high correlation between two features, it was inferred that they are mostly explaining same information. Hence, the feature with more computation cost is removed, in order to decrease the total computation time. Figure 2-19 illustrates the cross-correlation of different features for channel 5.
- Full feature set (Full): This feature set includes all the 15 mentioned features, representing time and frequency domain characteristics.

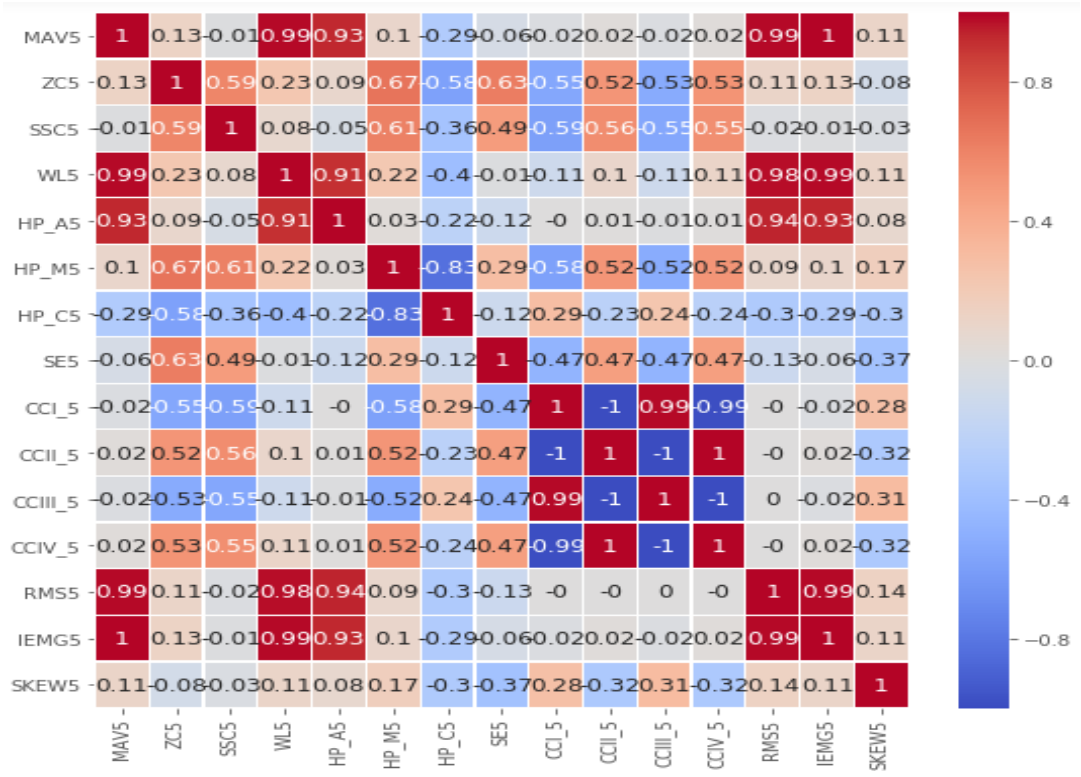

Figure 2-19 Cross correlation between features

All the steps up to this point including feature extraction has been coded in Matlab. From this point forward, all steps are implemented using Python including the designing of classifiers which was implemented with “sklearn” (v 0.21.2) library [56]. After extracting the features from the raw EMG signal we have a new database that has features as inputs and labels as outputs. The final shape of the input feature vector is (1xN), where N is 10 times the number of selected features. For example, for full feature set, a vector of 150 elements is given to the classifier and represents a 250ms window.

At this stage, the training data and the testing data should be separated. In each hand orientation (upward, downward, sideway) subjects were requested to repeat each movement for 5 times. It is important to mention, sometimes not all 5 repetitions were included in the database. The repetitions could decrease to 4 or even 3, depending on the number of correct movements executed by subject. As previously mentioned in section 2.1.2, some movements were discarded prior to signal processing by the observer. In each round, 2/3 of repetitions of each movement is added to training set, while the remaining part is included in testing set. Consequently, both training and testing datasets include all hand orientations for each movement.

Outlier removal and scaling are the measures taken to improve the classifier performance in majority of related studies. In what follows, a brief elaboration on these methods which were also deployed in this work is presented:

- Outlier removal: While the testing database is intact, the training performance is boosted by removing outliers from training set. Removing too much of outliers would endanger the robustness of model, hence it’s important to find the right threshold for determining outliers. The outliers were determined based on the values of their MAV and WL features. Samples outside of the bounds defined by 2.5 times of standard deviation around mean are labeled as outliers.
- Scaling: Almost all classifiers are sensitive to highly different ranges among their inputs. Each feature of training set is scaled to have zero mean and standard deviation equal to one. A significant consideration here, is that for online classification we do not have access to testing dataset and it is produced in time as the experiment goes on. As a result, the same scaling parameters which is created and fitted on the training set is used to scale testing set. This scaling strategy can be directly applied in an online classification scheme. Figure 2-20 shows the range of a group of features before and after scaling.

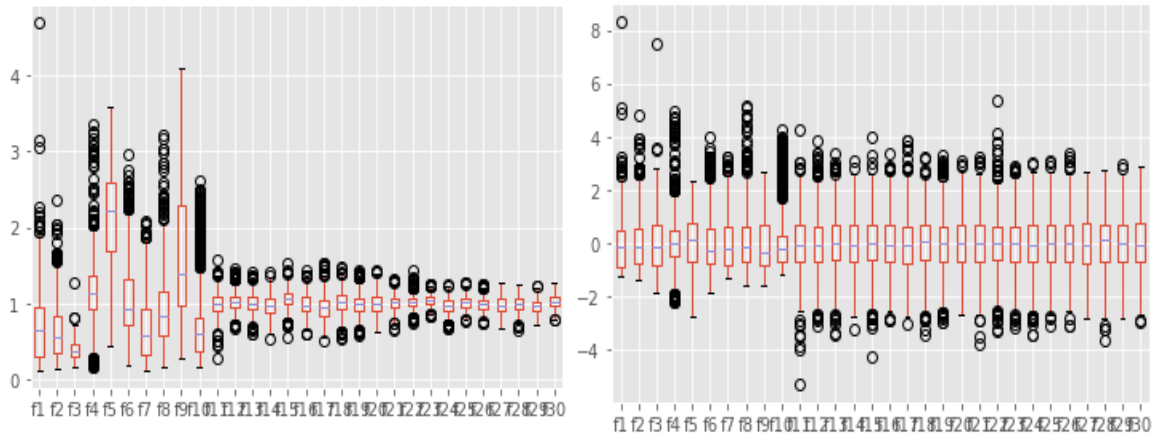

Figure 2-20 the effect of scaling on the ranges of the features

The improved data set is ready to be trained and tested. In regards to the classifiers, the hyper-parameters should be optimized to improve the results. A grid search has been done on the hyper-parameters for each classifier to find the best performing classifiers using the training set of a randomly chosen subject with 3 fold cross-validation. The ranges of the hyper-parameter search and selected hyper-parameters are displayed in tables 2-3 and 2-4, respectively.

Table 2-3 Range of hyper-parameter for classifiers

Classifier	Parameter 1	Parameter 2	Parameter 3	Parameter 4
KNN	K: 10, 20, 30, 40, 50	weights: uniform, distance		
MLP	hidden layers: (100,50,20), (50,20), (10,8), (100), (200), (100,20)	Alpha: 0.001,0.0001,0.00001	activation function: tanh, identity, relu, logistic	solver: sgd, adam
SVM	Regularization: 0.1,1,10,100	kernel: linear, poly, rbf, sigmoid	degree for poly:1,2,3,4	gamma: auto, 0.1 to 10e-7
LDA	solver: svd, lsqr, eigen			

Table 2-4 Selected hyper-parameters

Classifier	Parameter 1	Parameter 2	Parameter 3	Parameter 4
KNN	K: 40	weights: uniform		
MLP	hidden layers: (100)	Alpha: 0.0001	activation function: tanh	solver: sgd
SVM	Regularization: 1	kernel: linear	None	gamma: auto
LDA	solver: svd			

2.3.4 Deep learning

The deep learning algorithms are completely implemented in Python. “TensorFlow” [57] and “Keras” [58] libraries are employed for developing networks. These libraries are well-known, powerful and open-source software libraries for developing neural networks and machine learning methods.

Train and test separation is performed exactly as explained in section 2.2.3 for traditional classifiers. However, as opposed to traditional classifiers, deep learning algorithms don’t require features. The raw EMG signal or a representation of it (e. g. spectrogram of the EMG signal) can be used as input of the network.

There are 4 deep learning architectures proposed in this thesis. 3 of which are based on typical CNN and one is inspired by Residual CNN. All architectures can be divided into 2 stages. First stage is an inter-connected network of convolutional blocks working as a “feature extractor” and the second stage is consisted of few fully connected layers serving as the “classifier”.

Activation function: Deep neural networks take advantage of non-linear activation functions to extract non-linear features from input data. Traditionally, saturated functions such as sigmoid and tanh were used. However, recently, non-saturated activation functions such as Rectified Linear Units (ReLU) have gained popularity in order to solve exploding/vanishing gradient problem and increase the speed of convergence [59]. There are several modifications of ReLU such as leaky rectified linear (Leaky ReLU), parametric rectified linear (PReLU) and randomized rectified linear (RReLU). RReLU, which was introduced in a recent Kaggle National Data

Science Bowl (NDSB) competition [18], has proven to reduce the problem of overfitting of ReLU due to its randomized nature. According to the comparison published by [59] on an image classification task, RReLU displayed better results among other ReLUs mentioned above, which motivated the choice of RReLU for the classification problem in this project.
Overfitting: Due to their high number of parameters, deep neural networks particularly tend to over-fit. In order to overcome this issue, many techniques have been suggested by literature, among which the following 3 pre-cautions have been employed:
1) Drop out: Srivastava and colleagues [19] presented dropout technique, in which some random neurons with probability of p (e.g. 0.3) are eliminated from hidden layers. As a result, complex coadaptation of features between neurons can be prevented during training, leading to reduction of over-fitting.
2) Batch normalization: Introduced by Ioffe et al. [20], Batch Normalization (BN) was targeted to solve the need of low learning rate and careful parameter initialization in training of deep neural networks. It is a type of regularization technique, which performs input normalization in each training mini-batch. Ioffe et al. [20] showed that utilizing BN in classification of ImageNet can improve the results comparing to the state-of-art.
3) Early stopping: In iterative learning methods, after each iteration, the network is more fitted to training data. However, if the number of iterations exceeds a point, although the model will perfectly fit the training data, it would no longer be able to classify the unseen data correctly, thus the generalization error will increase. Early stopping provides a rule to limit the number of iterations in order to avoid this overfitting problem. In this work, the validation error in each update is monitored. When it reaches a minimum, the learner would continue training only for a certain number of iterations and then stops the training. Meanwhile, if a new minimum is observed, it will restart counting iterations before stopped. The mentioned number of iterations is referred to as “patience” and is set by user.

In recent years, many architectures of CNN for hand movement recognition based on sEMG have been proposed, among which [4, 5] are the main source of inspiration for the following network. Multiple modifications have been made by trial and error to boost the accuracy of the classifier. The main modifications are the size of filters and using RReLU activation function among others. Three approaches have been tested. One is 2-dimensional filter shape (Cnet2D), another is 1-dimensional filter (Cnet1D) and the last one is a combination of previous networks (CnetComb).

1. Cnet2D: Figure 2-21 displays an example of the proposed network for Cnet2D. 3 convolution blocks (Conv) are connected after each other, followed by 3 fully connected blocks (FC). However, the number of blocks can be regarded as a hyper-parameter to be optimized.

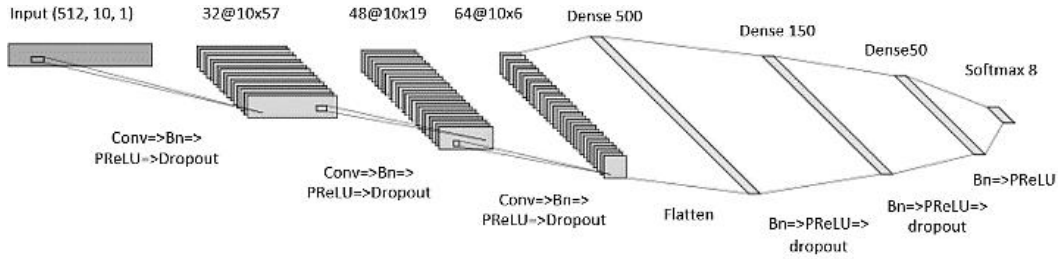

Figure 2-21 An example of Cnet2D/Cnet1D architectures

In this study exploration of different number of layers resulted in choosing 3 convolution blocks and 2 fully connected blocks. After 3 convolutional blocks, a flatten layer turns the 3-dimensional output of previous layer into a feature vector. The feature extractor stage concludes with the flatten layer. Then, 2 fully connected blocks are followed. Each Conv block consists of a convolution layer with 2D filter shape (e.g. (3,3)), BN, RReLU activation layer, max pooling and dropout. First fully connected block includes dense layer, BN, RReLU and dropout, while the last fully connected block does not include dropout. At the end, a Softmax layer has been included to create the 8 output of classifier. Softmax layer produces 8 outputs ranging from 0 to 1, indicating the probability of sample belonging to each class. The highest probability corresponds to predicted class for the input sample.
Adam optimizer [21] is used as optimization method for this and all the deep networks. During training, the model with minimum validation (20% of training data is randomly selected as validation set) loss is saved and used for testing; this technique is called Model Check Point and is used in this and all following networks. In order to find the hyper-parameters of this network, different combinations of hyper-parameters were tested on validation set of Nearlab database. In table 2-5 the hyper-parameters related to this architecture found by trial and error, is shown. The hyper-parameters can be divided into 3 groups. The ones related only to feature extractor stage, the ones only related to classifier stage and the ones that can be associated to the general design of the network.

Table 2-5 Hyper-parameters of architecture Cnet2D on Nearlab dataset

	Hyper-parameter	Block number: Value
Feature extractor stage	Filter numbers of 3 Conv blocks	1: 32, 2: 48, 3: 64
	Filter kernel sizes of 3 Conv blocks	1: (3,13), 2: (3,9), 3: (3,5)
	Filter stride sizes of 3 Conv blocks	1: (1,1), 2: (1,1), 3: (1,1)
	Max pooling sizes of 3 Conv blocks	1: (1,4), 2: (1,4), 3: (1,4)
Classifier stage	Dense layer sizes of 2 FC blocks	1: 300, 2: 50
General hyper-parameters	RRelu parameter range	All: 1/8 to 1/7
	Dropout rate	All: 0.5

In figure 2-22 a schematic of the network is shown. The sequence of the blocks from top to bottom shows the layers from deep to shallow. In this figure the input shape of each block is also mentioned. The input shape of the network is 10 x 512 x 1 which is channel x data points along the sub-window x number of feature maps. In the case of image classification, feature maps coincide with red, green and blue maps in an rgb input picture, however as here raw EMG signal is used, this number is equal to one. In the subsequent blocks, this number is equal to the number of filters used in the convolution blocks.

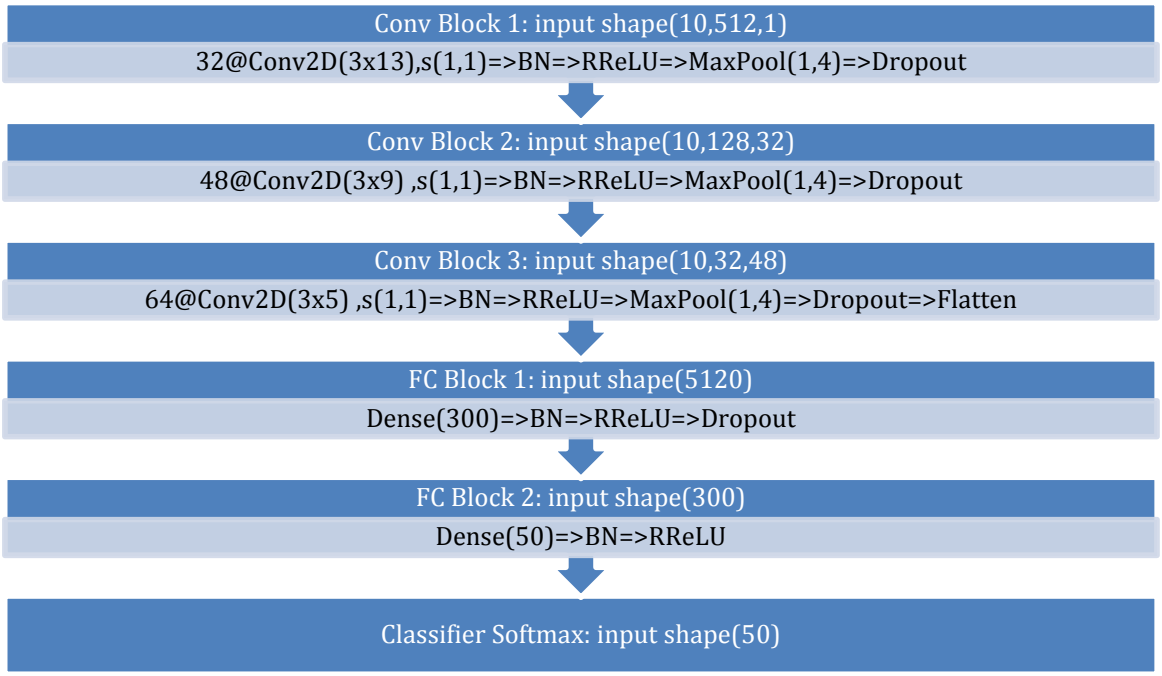

Figure 2-22 Schematic of Cnet2D architecture for Nearlab dataset

2. Cnet1D: The architecture of Cnet1D, is similar to that of Cnet2D. However, the shape of filter is such that it does not exploit the relations between channels in the feature extraction stage (e.g., (1,3)). Focusing on features extracted from the information of individual channels, it could potentially increase the accuracy of classifier.

In order to find the hyper-parameters of this network, different combinations of hyper-parameters were tested on validation set of Nearlab database (20 percent of training set). In table 2-6 the hyper-parameters related to this architecture are shown. Besides the filter shape, the dropout of this network is different from Cnet2D. Similar to Cnet2D, the hyper-parameters are divided into 3 groups: feature extractor stage, classifier stage and general hyper-parameters.

Table 2-6 Hyper-parameters of architecture Cnet1D for Nearlab dataset

	Hyper-parameter	Block number: Value
Feature extractor stage	Filter numbers of 3 Conv blocks	1: 32, 2: 48, 3: 64
	Filter kernel sizes of 3 Conv blocks	1: (1,13), 2: (1,9), 3: (1,5)
	Filter stride sizes of 3 Conv blocks	1: (1,1), 2: (1,1), 3: (1,1)
	Max pooling sizes of 3 Conv blocks	1: (1,4), 2: (1,4), 3: (1,4)
Classifier stage	Dense layer sizes of 2 FC blocks	1: 300, 2: 50
General hyper-parameters	RRelu parameter range	All: 1/8 to 1/7
	Dropout rate	All: 0.3

In figure 2-23 a schematic of the network is shown. The sequence of the blocks from top to bottom shows the sequence from deep layers to shallow ones. The construction of the blocks is as previously explained for architecture Cnet2D. Here again the output is produced by Softmax layer.

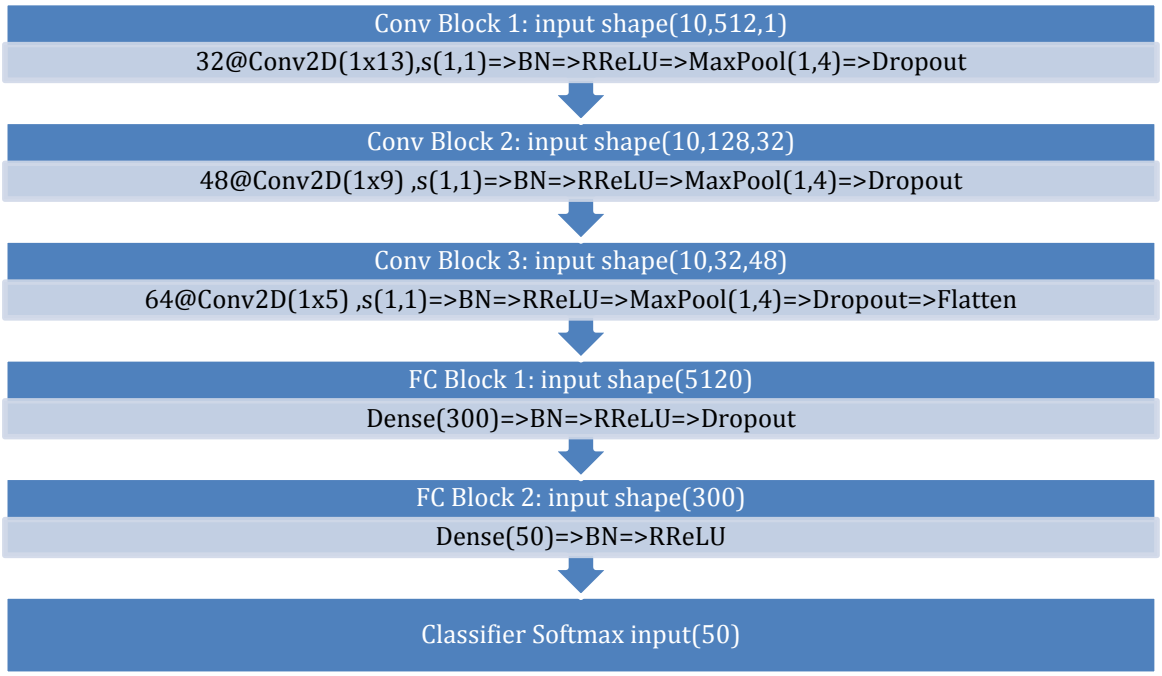

Figure 2-23 Schematic of Cnet1D architecture for Nearlab dataset

3. CnetComb: Provided that Cnet1D extracts features related to individual channels and Cnet2D exploits the relationship between channels, it seems reasonable to introduce an architecture leveraging both strategies for features extraction. In this architecture, feature extractor stages of Cnet1D and Cnet2D extract the 1D and 2D features. Afterwards, the features would be concatenated, flattened and fed to one classifier stage similar to the ones used before. An example of this network can be seen in Figure 2-24.

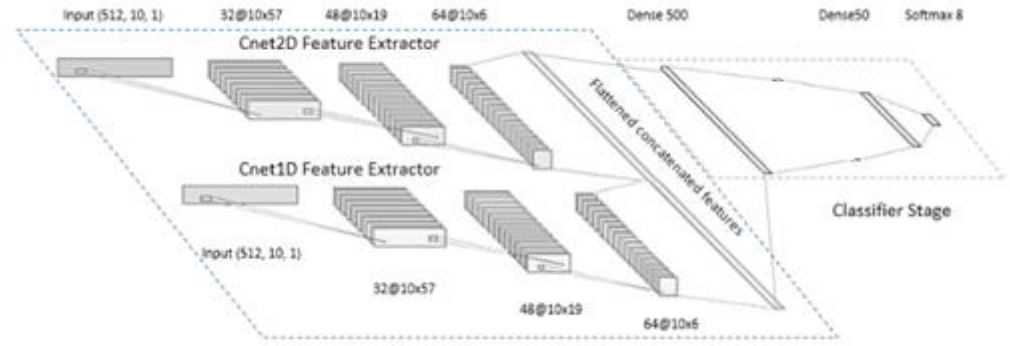

Figure 2-24 CnetComb general architecture

In order to find the hyper-parameters of this network, different combinations of hyper-parameters were tested on validation setoff Nearlab database. In table 2-7 the hyper-parameters related to this architecture are shown. Similar to Cnet2D, the hyper-parameters are divided into 3 groups: feature extractor stage, classifier stage and general hyper-parameters. Feature extractor stage itself is consisted of 1D feature extractor and 2D feature extractor.

Table 2-7 Hyper-parameters of architecture CnetComb on Nearlab dataset

Hyper-parameter	Block number: Value
Filter numbers of 3 Conv blocks	1: 32, 2: 48, 3: 64
Filter kernel sizes of 3 Conv blocks	1: (3,13), 2: (3,9), 3: (3,5)

Feature extractor stage	Cnet2D feature extractor	Filter stride sizes of 3 Conv blocks	1: (1,1), 2: (1,1), 3: (1,1)
		Max pooling sizes of 3 Conv blocks	1: (1,4), 2: (1,4), 3: (1,4)
	Cnet1D feature extractor	Filter numbers of 3 Conv blocks	1: 32, 2: 48, 3: 64
		Filter kernel sizes of 3 Conv blocks	1: (1,13), 2: (1,9), 3: (1,5)
		Filter stride sizes of 3 Conv blocks	1: (1,1), 2: (1,1), 3: (1,1)
		Max pooling sizes of 3 Conv blocks	1: (1,4), 2: (1,4), 3: (1,4)
Classifier stage		Dense layer sizes of 2 FC blocks	1: 300, 2: 50
General hyper-parameters		RRelu parameter range	All: 1/8 to 1/7
		Dropout rate	All: 0.3

In figure 2-25 a schematic of the network is shown. The construction of the convolutional blocks and fully connected blocks are similar to the architecture Cnet2D.

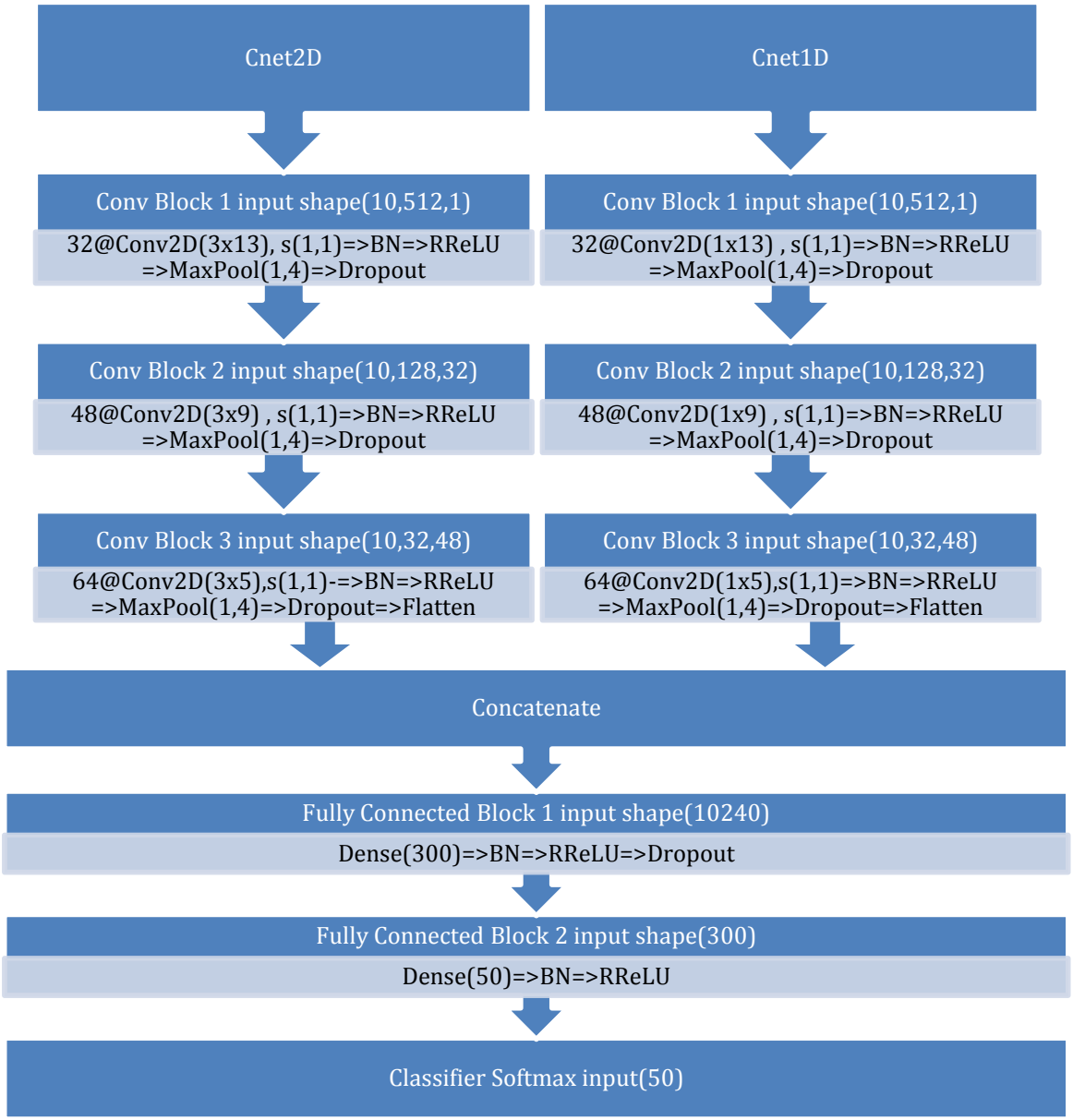

Figure 2-25 Schematic of CnetComb architecture on Nearlab dataset

4. RESnet: This architecture is inspired by the Residual Neural Networks [22] as a solution for vanishing gradients in deep neural networks. In residual neural networks, skip identity connections are utilized around convolutional blocks in order to transfer the influence of shallower layers into deeper layers. The skip identity connections allow for training deeper layers, which enables having more convolutional blocks in this proposed architecture. Figure 2-26 is illustrating an example of a general residual neural network.

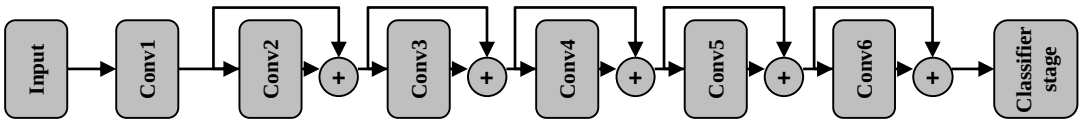

Figure 2-26 A general residual neural network architecture

However, in this study a modification has been applied to the original proposed residual neural network. The modification is including convolutional layers in the skip connection from input. Figure 2-27 shows the structure of the proposed network. In contrast to Cnet1D and Cnet2D, which had 3 convolutional blocks as feature extractor stage, this architecture uses two branches for extracting features. The left branch is consisted of 2 convolutional blocks, followed by a single convolutional layer (referred to as “Conv layer α”), as shown in figure 2-27. Right branch includes one convolutional block followed by the same single convolutional layer (Conv layer α). The output of the two single convolutional layers of two branches are summed together and the result is given to a batch normalization (BN) layer, RRelu, average pooling layer and dropout (referred to as “β block”). A flatten layer is used to conclude the feature extractor stage. At the end, classifier stage is added. The classifier stage is 2 fully connected blocks followed by a Softmax layer, similar to previous networks.

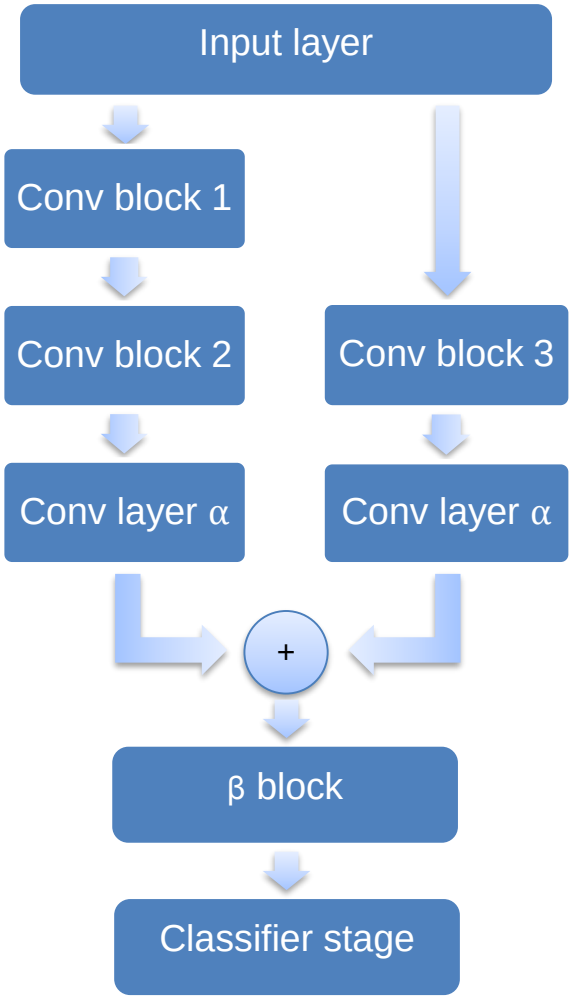

Figure 2-27 General schematic of proposed RESnet structure

In order to find the hyper-parameters of this network, different combinations of hyper-parameters were tested on validation set (20 percent of training set) and the best setting is reported in table 2-8.

Table 2-8 Hyper-parameters of architecture RESnet on Nearlab dataset

		Hyper-parameter	Block number: Value
Feature extractor stage	Conv blocks	Filter numbers of Conv blocks 1,2 and 3	1: 32, 2: 48, 3: 48
		Filter kernel sizes of Conv blocks 1,2 and 3	1: (1,13), 2: (1,9), 3: (1,13)
		Filter stride sizes of Conv blocks 1,2 and 3	1: (1,4), 2: (1,1), 3: (1,4)
		Max pooling sizes of Conv blocks 1,2 and 3	1: None, 2: (1,4), 3: (1,4)
	Conv layer α	Filter number	64
		Filter kernel size	(1,5)
		Filter stride size	(1,1)
	β block	Average pooling size	(1,4)
Classifier stage		Dense layer sizes of 2 FC blocks	1: 150, 2: 50
General hyper-parameters		RRelu parameter range	All: 1/10 to 1/9
		Dropout rate	All: 0.3

In figure 2-28 a schematic of the network is shown.

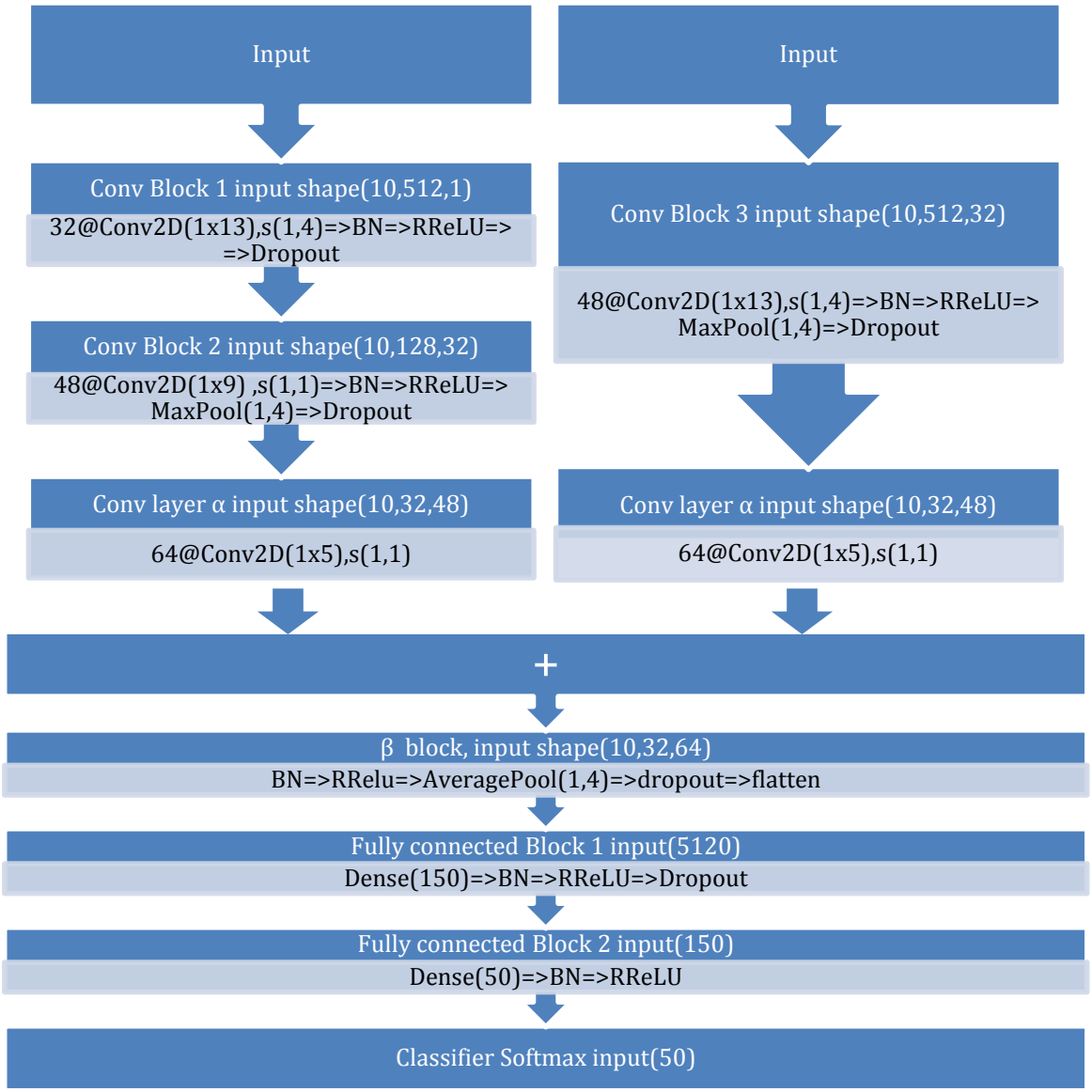

Figure 2-28 Schematic of RESnet architecture on Nearlab dataset

2.3.5 Transfer learning

In some studies, transfer learning (TL) is used to transfer the knowledge of pre-trained image classifiers (VGG16 and AlexNet) into sEMG-based physical action recognition [60]. In other research works such as [51], the knowledge learned in one session of recording was transferred to another session. In this study, TL is used to leverage the shared information among different subjects to obtain bigger training dataset to train deep neural networks. There are two transfer learning schemes designed in this study.

Method I: Freeze & fine-tune
The idea is to use a pre-trained network (trained in the source domain), remove the last layer (classifier stage) and attach the new adapted (according to target domain) classifier stage. Finally, the network should be re-trained in the target domain. The process of re-training is done by freezing the deeper layers' weights and fine-tuning

the shallower layers. Fine-tuning can be achieved by initializing with previous weights and choosing very low learning rates [43].

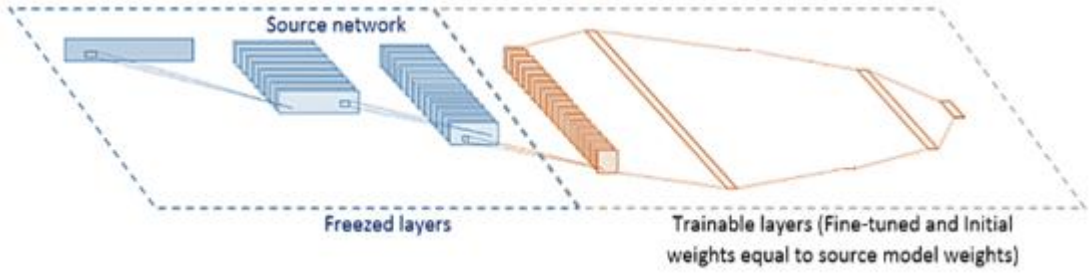

Figure 2-29 Freeze & fine-tune architecture

First, the selected network (e.g. Cnet1D) is pre-trained with the data of 5 selected subjects with the highest single subject accuracy not including the target subject. Afterwards, the first 2 convolutional blocks (except for Batch Normalization layers) are frozen and other layers (including the last Conv block plus all the FC blocks) are fine-tuned for targeted subject using the subject's database. Figure 2-29 displays the architecture of network. Since the aim of this thesis is to only apply transfer learning to the best deep learning network, the exact architecture should be chosen after observing the results of deep learning models. Hence, the hyper-parameters of this network will be discussed in chapter 4.

Method II: Parallel networks
In this method, two networks are used in parallel. One is trained on the data of 5 selected subjects with the highest single subject accuracy not including the target subject (referred to as "Source Network"), while the other is trained on the target subject's database (referred to as "Target Network"). The features extracted by the mentioned networks would be concatenated, while their classifier stages would be disregarded and a new classifier stage with 2 fully connected blocks is added after feature layer making the final model referred to as "Final Network". Finally, in order to preserve the learned features from the two pre-trained networks, the weights of their feature extractor stage would be frozen (not learnable) and the classifier stage would be trained using the target subject's database (with random initialization). Figure 2-30 demonstrates an example of the final model with 3 fully connected blocks. Similar to method I the exact architecture of this network will be discussed in chapter 4.

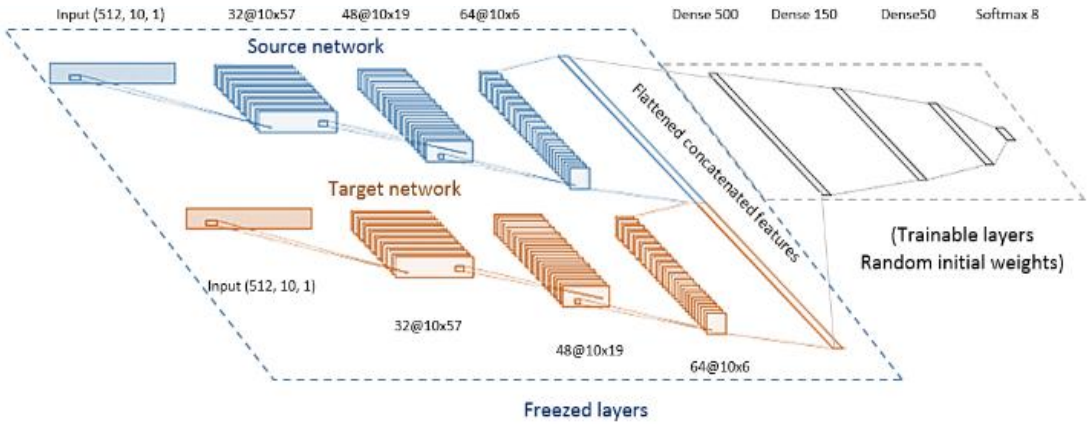

Figure 2-30 Parallel networks architecture

2.3.6 Summary of explored methods

Table 2-9 summarizes all the methods introduced in this chapter which are going to be used in this study.

Table 2-9 Summary of explored methods

Classification family	Input shape	Input	Classifier
Traditional machine learning	Number of channels x number of selected features (e.g. 15 x 10)	One of the 4 defined feature sets (TD, ITD, CB, Full)	KNN LDA MLP SVM
Deep learning	Number of channels x number of data points (10 x 512)	Raw EMG signal	Cnet1D Cnet2D CnetComb RESnet
Transfer learning	Number of channels x number of data points (10 x 512)	Raw EMG signal	Parallel networks Freeze & fine-tune

2.4 Performance measures

2.4.1 Basic metrics

The most common performance measure in a classification problem is “accuracy”, which is the ratio of correct classification over the total number of classifications. Though this measure is very common, it has its own draw backs. As an example, if the classifiers is testing an unbalanced dataset, accuracy could create misleading results. In other cases, accuracy might not indicate the essential information a researcher is looking for. For example, in some problems, false negative predictions are highly costly (e.g., incorrectly labeling an ill patient as healthy). Thus, two classification algorithms with same accuracy but different false positive rate could

differ radically in such cases. Therefore, to solve the mentioned problems other performance measures are introduced, such as “Recall”, “Precision” and “F1 score”.

2.4.2 Multiclass metrics

Multi-class performance measurement, could also be done using accuracy metric. However, in order to have a more descriptive measure and have class specific analysis, confusion matrix has also been used in this project. Figure 2-31 is an example of a confusion matrix. Using this metric, the performance of the classifier could be analyzed in each class. Vertical axis indicates the true classes of the samples while the horizontal axis indicates the output of classifier. The total number of samples truly belonging to a certain class can be obtained by summing all the elements in the row of the mentioned class. Similarly, the total number of samples which are predicted to belong to the same class can be obtained by summing all the elements in the column of the mentioned movement.

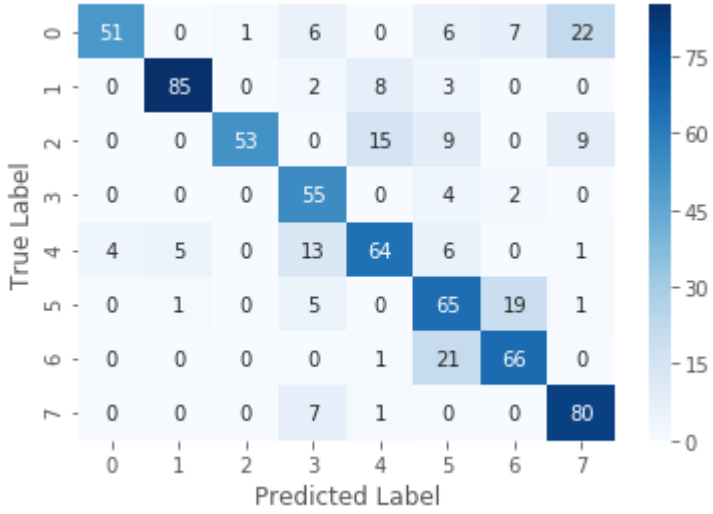

Figure 2-31 Example of confusion matrix

2.4.3 Variance analysis

In order to compare the performance of two methods tested on a population of subjects and draw conclusions about their differences, a scientific metric should be defined. In this thesis, Wilcoxon signed-rank test is used as a metric to determine if two methods have significant statistical difference in their performance. Wilcoxon signed-rank test is a non-parametric statistical hypothesis test which is very appropriate for a repeated measure design where the same subjects are evaluated under two different conditions [61]. It is also very useful when normality in data cannot be assumed [62].

3 Classical machine learning results

3.1 Classification performance analysis

3.1.1 Nearlab dataset

All 4 feature sets are classified by the 4 classifiers on all 11 subjects included in Nearlab dataset and their accuracy result is reported in table 3-1. Since the output of multilayer perceptron (MLP) depends on the initial weights and the order of training samples, accuracy results were averaged over 20 runs of training and testing.

Table 3-1 Classification accuracy over all 11 subjects of Nearlab dataset

Model	KNN				LDA			
Feature Set	TD	ITD	CB	Full	TD	ITD	CB	Full
S 1	95.24	96.54	96.54	96.27	94.96	96.09	**97.88**	96.39
S 2	92.29	93.50	92.44	93.27	93.03	94.12	94.15	94.77
S 3	80.52	83.45	83.48	84.60	84.94	88.20	84.85	88.69
S 4	84.58	84.22	87.63	84.41	89.46	90.27	91.44	90.89
S 5	88.93	90.37	90.85	90.13	88.96	90.97	**92.54**	92.12
S 6	88.21	88.69	87.00	86.31	88.21	88.39	90.87	88.63
S 7	82.03	83.56	79.99	79.93	83.20	83.20	83.13	83.93
S 8	88.56	88.29	86.00	86.02	89.08	95.46	93.49	95.59
S 9	89.60	91.87	90.39	92.25	88.66	92.88	92.50	94.71
S 10	91.62	93.10	93.26	93.32	95.17	97.63	96.90	**98.60**
S 11	85.16	87.62	88.05	88.81	87.65	90.66	92.90	**93.76**

Model	MLP				SVM			
Feature Set	TD	ITD	CB	Full	TD	ITD	CB	Full
S 1	93.94	95.32	96.34	96.06	92.14	93.78	96.36	95.72
S 2	93.14	93.53	89.01	91.13	96.16	**96.35**	92.29	94.77
S 3	86.94	87.97	87.05	**88.81**	86.52	86.89	86.89	86.89
S 4	88.88	89.55	**91.47**	90.44	87.17	88.06	91.08	89.23
S 5	89.12	90.17	91.43	90.65	87.88	89.23	89.17	89.89
S 6	**91.51**	91.24	89.95	90.65	90.33	90.75	90.27	91.48
S 7	85.96	85.92	82.71	84.56	**88.88**	88.33	83.81	85.46
S 8	94.20	93.84	90.98	91.91	95.17	**95.96**	91.09	92.86
S 9	91.34	92.00	90.54	91.23	93.23	**93.38**	91.18	92.00
S 10	93.57	95.32	94.15	94.98	97.06	96.57	94.87	96.02
S 11	89.47	91.10	93.09	92.56	89.43	89.64	92.22	91.86

In this table, considering all the combinations of classifiers and features, the result of the best performing combination is shown in bold format for each subject. It can be seen that there is no single combination of classifier and feature set that works best for all subjects (no column has elements all in bold format). This shows the high variety between sEMG signals of different subjects, also referred to as inter-subject variability.

In order to better compare the classifiers and feature sets, average accuracy over all subjects should be considered. Figure 3-1 visualizes the average accuracy of all the combinations.

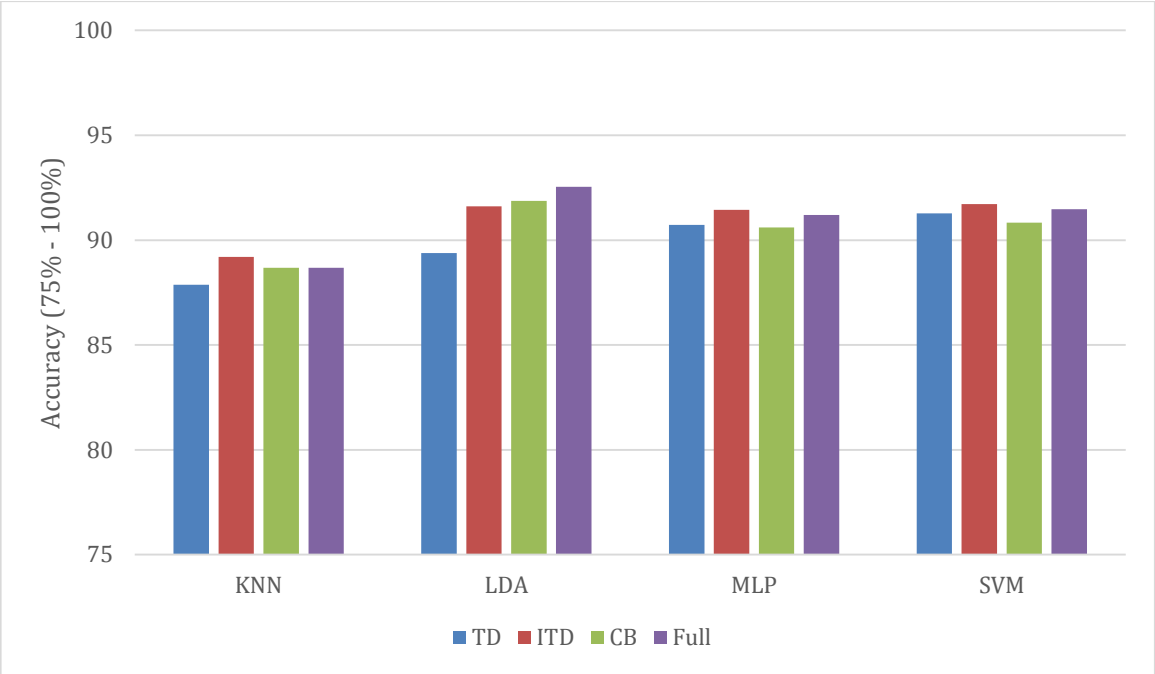

Figure 3-1 Comparison between average classification accuracies of Nearlab dataset

Table 3-2 illustrates average accuracy, standard deviation (std), median, interquartile range (IQR) (over 11 subjects) and result of non-parametric statistical significance analysis. Pairwise Wilcoxon Signed-Rank Test is applied to compare performance of features sets with the best performing feature set in each classifier. For example, for LDA classifier, the considered pairs are (Full vs. TD), (Full vs. ITD) and (Full vs. CB). (Null hypothesis is rejected when H0 = 0 ($p < 0.05$)).

Table 3-2 Comparison between classifiers and feature sets on Nearlab dataset (Pairwise Wilcoxon Signed-Rank Test is applied to compare performance of features sets with the best performing feature set in each classifier) (Null hypothesis is rejected when H0 = 0 ($p < 0.05$))

Classifier	KNN				LDA			
Feature set	TD	**ITD**	CB	Full	TD	ITD	CB	**Full**
Average(%)	87.88	**89.20**	88.69	88.68	89.39	91.62	91.88	**92.55**
Std(%)	4.46	4.35	4.67	4.89	3.75	4.18	4.45	4.23
Median(%)	88.56	88.69	88.05	88.81	88.96	90.97	92.54	93.76
IQR(%)	5.74	6.57	5.15	7.45	3.32	5.46	2.66	5.39
H0 (P-value)	0(0.008)	-	1	1	0(0.003)	0(0.003)	1	-

Classifier	MLP				SVM			
Feature set	TD	**ITD**	CB	Full	TD	**ITD**	CB	Full
Average(%)	90.73	**91.45**	90.61	91.18	91.27	**91.72**	90.84	91.47
Std(%)	2.86	2.97	3.62	3.02	3.72	3.60	3.45	3.42
Median(%)	91.34	91.24	90.98	91.13	90.33	90.75	91.09	91.86
IQR(%)	4.35	3.83	2.80	1.69	5.82	6.09	2.54	4.26
H0 (P-value)	0(0.016)	-	1	1	1	-	1	1

In table 3-2, the feature set which gives the highest average accuracy for each classifier is shown in bold format. This table demonstrates that ITD feature set gives the best average accuracy among other feature sets for SVM, MLP and KNN. However, for LDA it seems that as the number of features in a feature set increases, so does the classification accuracy. The statistical analysis performed within each

classification method indicates that there is no significant difference between top two feature sets in each classification method.
Table 3-3 provides a comparison between classifiers with their best performing feature sets. Although the LDA shows the best performance in average, the statistical analysis indicates that there is no significant difference between LDA using Full feature set and MLP and SVM using ITD feature set. he considered pairs are (LDA vs. KNN), (LDA vs. MLP) and (LDA vs. SVM).

Table 3-3 Comparison of classifiers with their best performing feature sets
(Pairwise Wilcoxon Signed-Rank Test is applied to compare performance of LDA with other classifiers)
(Null hypothesis is rejected when Ho = 0 ($p < 0.05$))

Classifier	KNN	LDA	MLP	SVM
Feature set	ITD	Full	ITD	ITD
Average(%)	89.20	**92.55**	91.45	91.72
Std(%)	4.35	4.23	2.97	3.60
Median(%)	88.69	93.76	91.24	90.75
IQR(%)	6.57	5.39	3.83	6.09
H0 (P-value)	0(0.008)	-	1	1

In Figure 3-2 the classification accuracy of each subject using LDA classifier (as the best classifier) and the 4 feature sets is compared. It can be seen that in almost all subjects, combining time domain features with frequency domain features as in CB and Full feature sets, improve the classification accuracy for LDA classifier. This result is in line with statistical data presented in table 3-2 which indicates that there is significant difference in Full feature set comparing to TD and ITD feature sets.

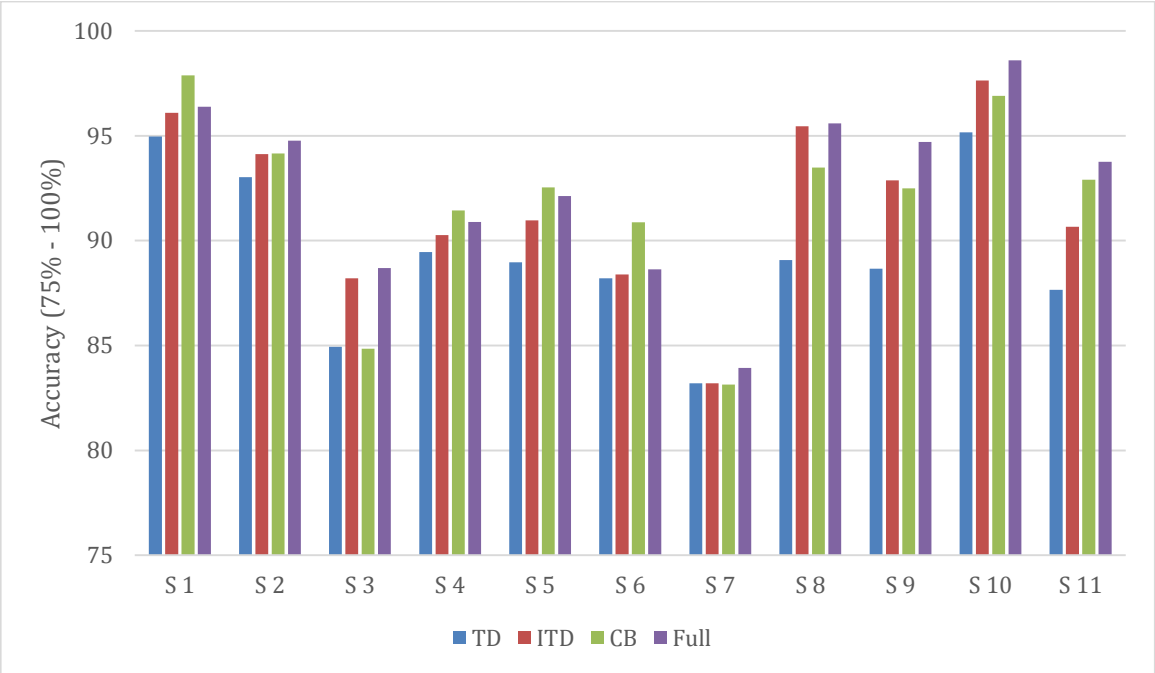

Figure 3-2 Comparison between feature sets for each subject of Nearlab dataset, using LDA classifier

3.1.2 Ninapro DB2

In order to be able to compare the performance of proposed methods with other studies, same classifier and feature set combinations were tested on Ninapro DB2

(explained in section 2.1.5). The results are averaged over all 40 subjects of this database. In Figure 3-3 the average accuracy of each combination is visualized.

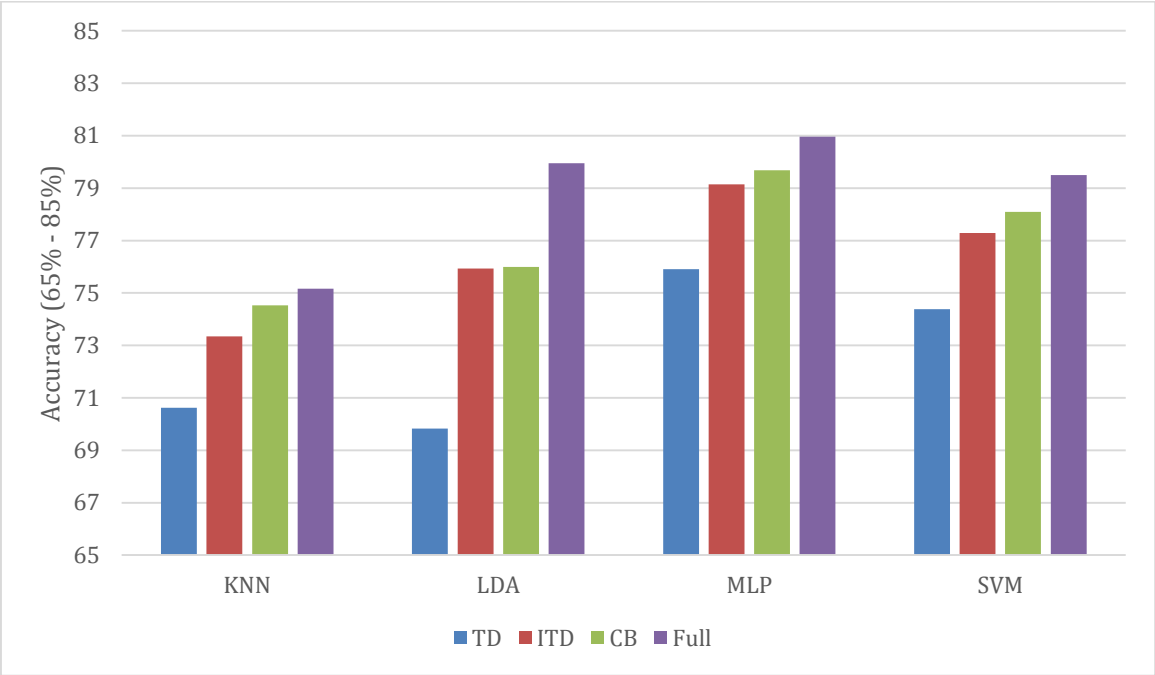

Figure 3-3 Comparison between average classification accuracies of Ninapro DB2 database

In table 3-4 the average accuracy, standard deviation (std), median, IQR and statistical analysis of all classifiers with all feature sets can be seen. Pairwise Wilcoxon Signed-Rank Test is applied to compare performance of features sets with the best performing feature set in each classifier. For example, for LDA classifier, the considered pairs are (Full vs. TD), (Full vs. ITD) and (Full vs. CB). (Null hypothesis is rejected when H0 = 0 ($p < 0.05$)).

Table 3-4 Comparison between classifiers and feature sets on Ninapro DB2
(Pairwise Wilcoxon Signed-Rank Test is applied to compare performance of features sets with the best performing feature set in each classifier)
(Null hypothesis is rejected when H0 = 0 ($p < 0.05$))

Classifier	KNN				LDA			
Feature set	TD	ITD	CB	**Full**	TD	ITD	CB	**Full**
Average(%)	70.62	73.34	74.53	**75.17**	69.83	75.93	75.99	**79.95**
Std(%)	6.13	6.27	6.34	6.52	6.25	6.00	6.28	5.73
Median(%)	71.44	73.72	74.53	75.29	70.83	76.01	76.68	80.62
IQR(%)	7.87	9.72	8.36	9.48	8.69	7.89	7.81	7.45
H0 (P-value)	0(0.000)	0(0.000)	0(0.010)	-	0(0.000)	0(0.000)	0(0.000)	-

Classifier	MLP				SVM			
Feature set	TD	ITD	CB	**Full**	TD	ITD	CB	**Full**
Average(%)	75.91	79.14	79.68	**80.97**	74.38	77.29	78.09	**79.50**
Std(%)	5.39	5.59	5.54	5.44	6.39	6.17	6.14	6.25
Median(%)	75.98	79.09	79.65	80.92	73.42	77.29	77.94	79.98
IQR(%)	8.17	7.81	7.21	6.79	9.00	9.72	9.18	9.55
H0 (P-value)	0(0.000)	0(0.000)	0(0.000)	-	0(0.000)	0(0.000)	0(0.000)	-

According to this table, in Ninapro database the best feature set for all classifiers is Full and the best classifier is MLP with 80.97% average accuracy. The statistical analysis indicates that, there are significant differences between full feature set and other feature sets in all classification methods.
Table 3-5 provides a comparison between classifiers with their best performing feature sets. MLP shows the best performance in average and the statistical analysis indicates that there is significant difference between MLP and other classifiers using Full feature set.

Table 3-5 Comparison of classifiers with their best performing feature sets (Pairwise Wilcoxon Signed-Rank Test is applied to compare performance of MLP with other classifiers) (Null hypothesis is rejected when H0 = 0 ($p < 0.05$))

Classifier	KNN	LDA	MLP	SVM
Feature set	Full	Full	Full	Full
Average(%)	75.17	79.95	80.97	79.50
Std(%)	6.52	5.73	5.44	6.25
H0 (P-value)	0(0.000)	0(0.001)	-	0(0.000)

In Figure 3-4 (a-d) classification result of all 40 subjects with MLP classifier and all feature sets is depicted. It is important to keep in mind that, the reported results for each subject is the average over 20 repetitions of training and testing.

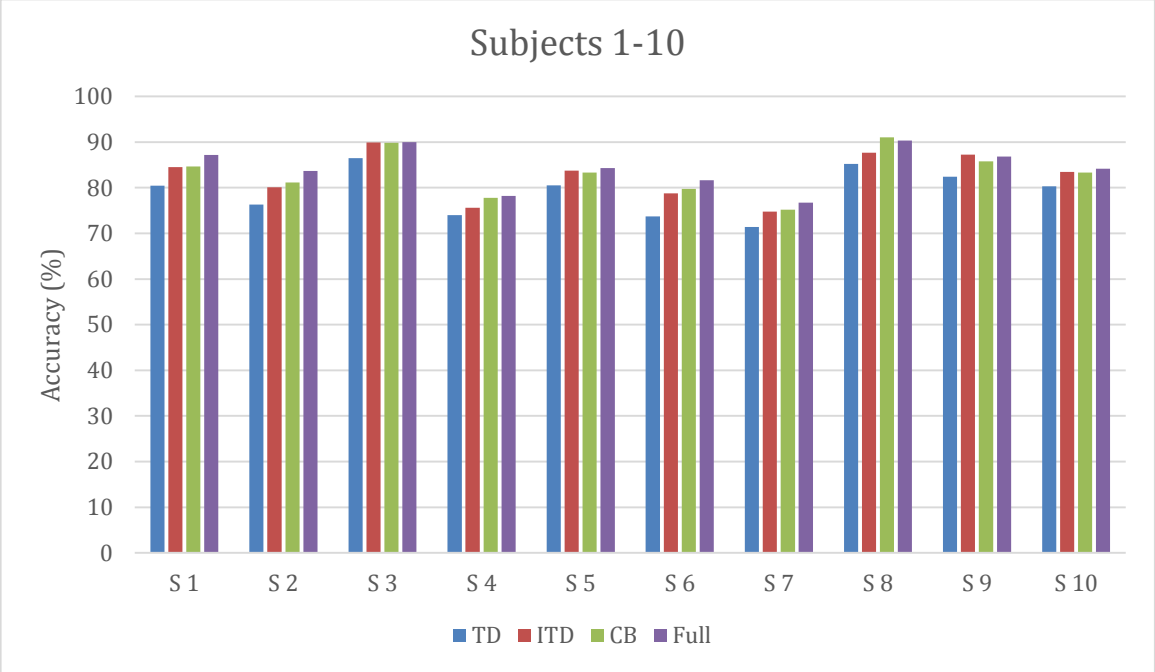

Figure 3-4-a Classification accuracy (%) of Ninapro DB2 subjects 1-10 with MLP

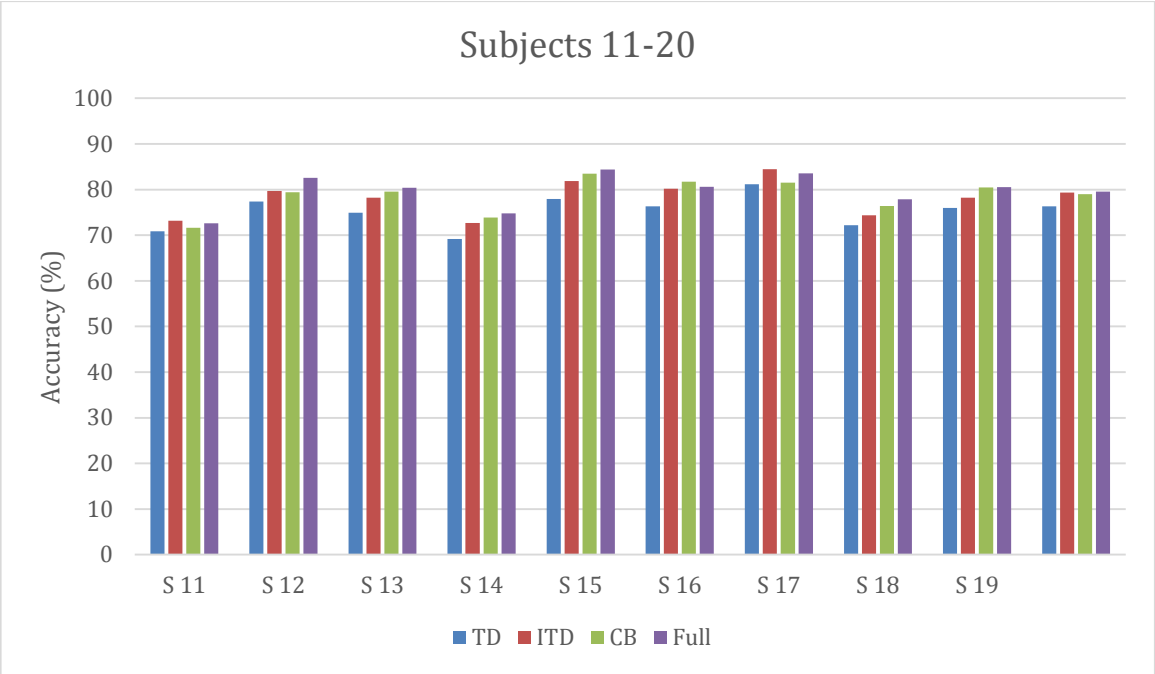

Figure 3-4-b Classification accuracy (%) of Ninapro DB2 subjects 11-20 with MLP

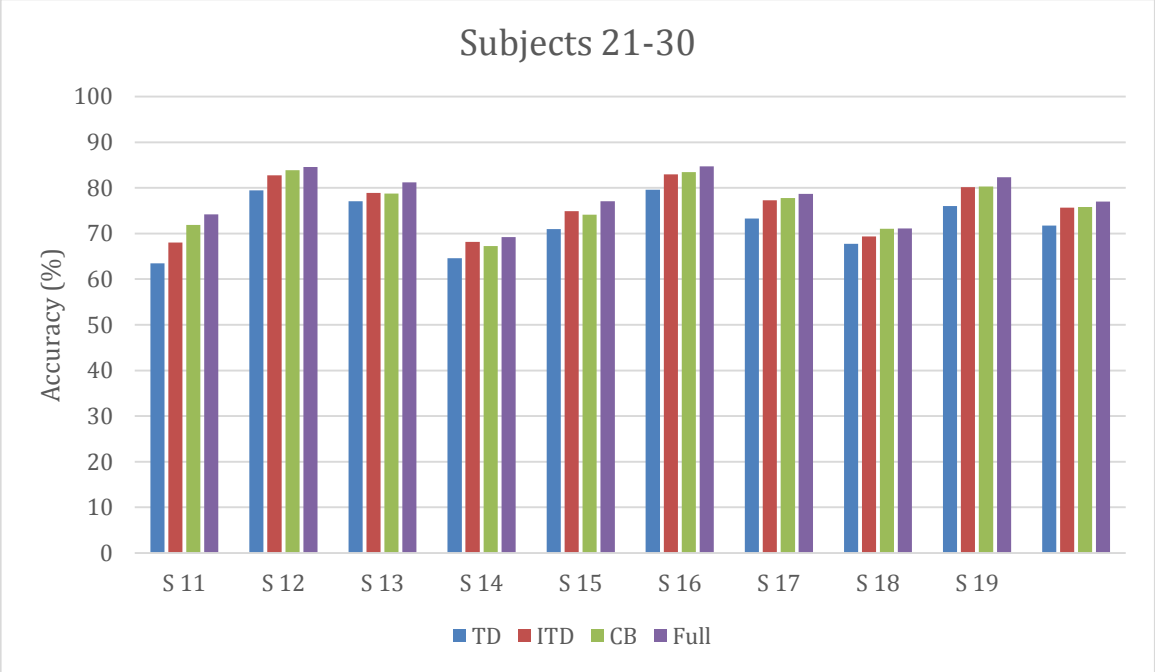

Figure 3-4-c Classification accuracy (%) of Ninapro DB2 subjects 21-30 with MLP

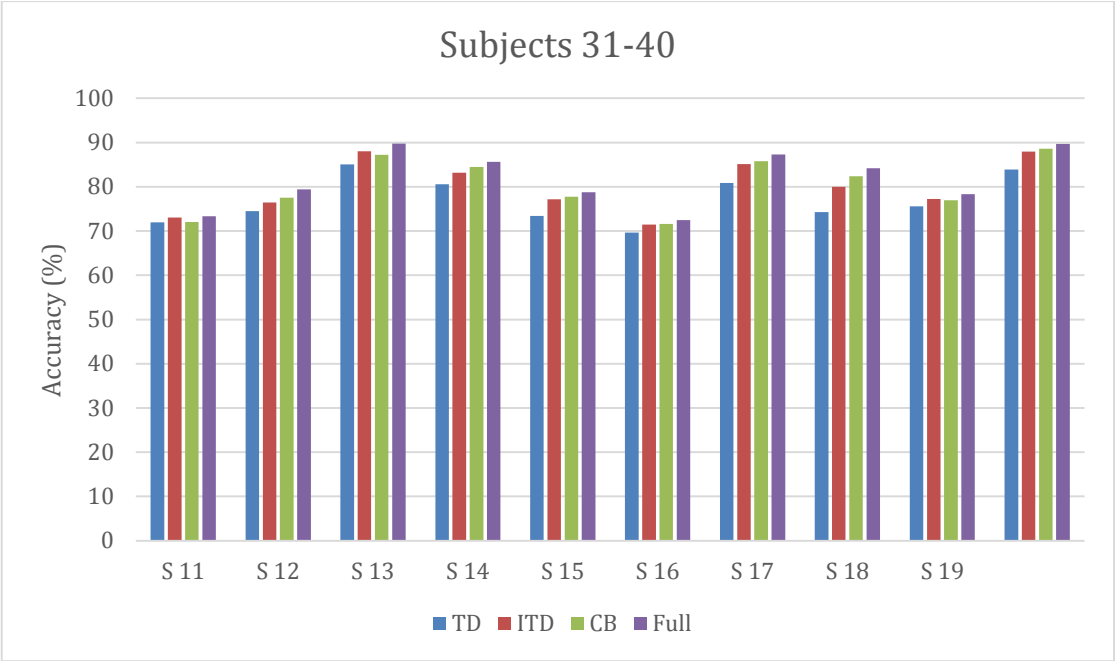

Figure 3-4-d Classification accuracy (%) of Ninapro DB2 subjects 31-40 with MLP

From the Figure 3-4 it can be inferred that Full feature set is generally the best feature set among most of the subjects. Again, the inter-subject variability is evident from different ranges of accuracy considering Full feature classification (from 69.25% till 90.31%).

3.2 Class specific analysis

In Figure 3-5, confusion matrix is used to demonstrate the ability of the best classifier (LDA with Full feature set) to classify individual movements. In order to generalize the performance measure over all subjects of Nearlab database, the confusion matrix is drawn in Figure 3-5 based on the average result over all subjects. In other words, each element of the confusion matrix is the average of corresponding elements of all subjects' confusion matrix.

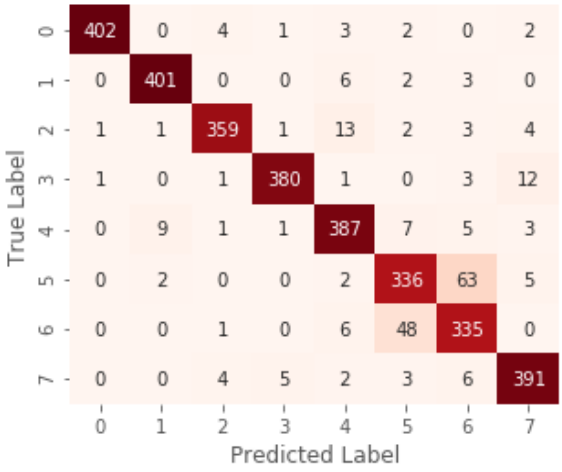

Figure 3-5 Confusion matrix averaged over all 11 subjects of Nearlab dataset

In Figure 3-5, class labels 0 to 7 correspond to hand flexion, extension, supination, pronation, open, pinch, lateral pinch and fist, respectively. Based on this Figure, the most frequent misclassifications happened between classes labeled with 5 and 6, i.e., pinch and lateral pinch. The possible reason behind this misclassification is the similarity of muscle activation patterns involved in the execution of the mentioned movements. In addition to intrinsic similarities between these two movements, in the subjects that did not execute these two movements in accordance with the instructions of video cue, similarity increases. In other words, they perform these two movements very similar to each other.
In Figure 3-6, the confusion matrix of subject 10 and 7 as best (98.60%) and worst (83.93%) performing subjects, respectively, are displayed based on LDA classification with Full feature set. The right graph represents the classification of subject 7 and the left is for subject 10. In accordance with the right graph, the most misclassifications in subject 7 are between the two pinch classes (labels 5 and 6). However, subject 10 does not have this problem. Since there is no significant misclassification in subject 10, there should be another reason apart from similarities in the movement classes for misclassification of classes 5 and 6 in other subjects.

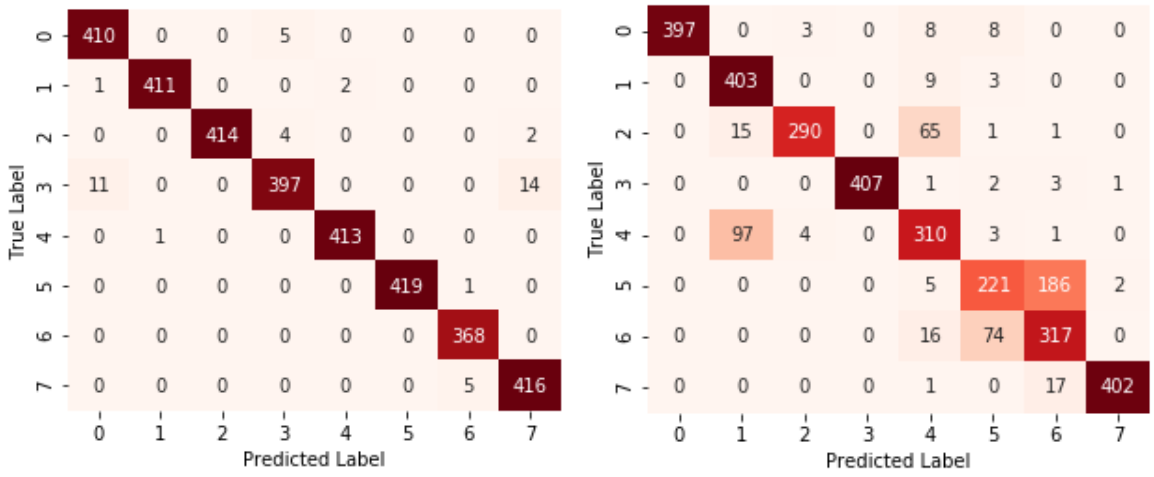

Figure 3-6 confusion matrixes for subject 10 (left) and subject 7 (right)

As mentioned before, an observer was responsible to monitor the movements of each subject. According to observer, if a subject performed incorrect movements according to video cue, the movement would be discarded from the database. This case happened often when the subject incorrectly performed the wrong pinch movement (e.g. lateral pinch instead of pinch). The effect of discarding movements is decreasing the number of available samples for training of that specific class. Figure 3-7 shows the number of available samples for each class (only round 1 dataset) of subjects 10 (left) and 7 (right).
According to protocol, each movement execution lasted almost 5 seconds (5x2048 samples) and the movements were segmented in 512 samples sub-windows with stride of 128 samples, hence each movement must include about 77 sub-windows. In each round, each movement class was repeated 5 times, therefore in total each movement class must have maximum 385 sub-windows. However, some movements were discarded in the case of incorrectness of execution, thus decreasing the number of available repetitions for each movement. From Figure 3-7 left associated to subject 10, it can be concluded that almost all movements were

executed correctly. On the contrary, in Figure 3-7 right related to subject 7, it is obvious that classes labeled 2, 5, and 6 have less than 5 repetitions.

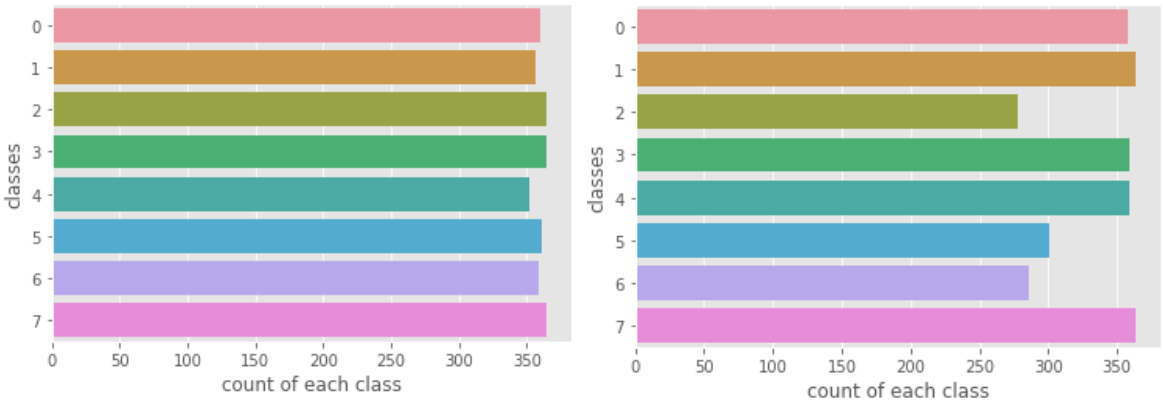

Figure 3-7 Round 1 class distributions of subject 10 (left) and subject 7 (right)

The less the number of available samples of one class, the less the network is able to train to discriminate that class from others. Hence, we can infer that network for subject 7 was not able to train well enough on classes labeled 2, 5 and 6. This can be the reason of misclassification between classes 5 and 6, and also class labeled 2 with others.

3.2.1 Modification on movement classes

Based on the idea that the pinch and lateral pinch classes are very similar and can be confused with each other, a modification of movement classes has been proposed and investigated in this section. These two classes were combined to create a general pinch class. Therefore, the database in this section has 7 classes instead of 8. In Figure 3-8 classification result on this modified database is demonstrated. The classification is performed using LDA with Full feature set.

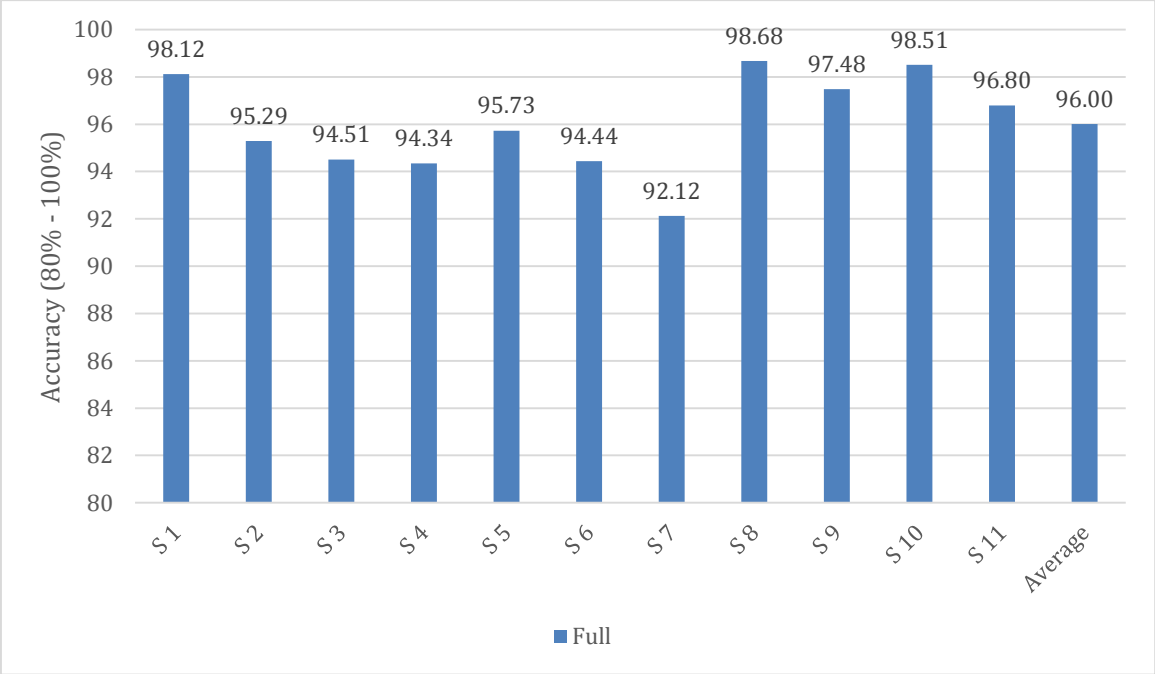

Figure 3-8 Classification accuarcy of LDA with Full feature set, using modified dataset

Figure 3-8 illustrates that after this modification, average accuracy of all subjects with LDA and Full feature set, has improved from 92.55% (table 3-2) to 96.00%.

In Figure 3-9 the new confusion matrix averaged on all subjects is displayed.

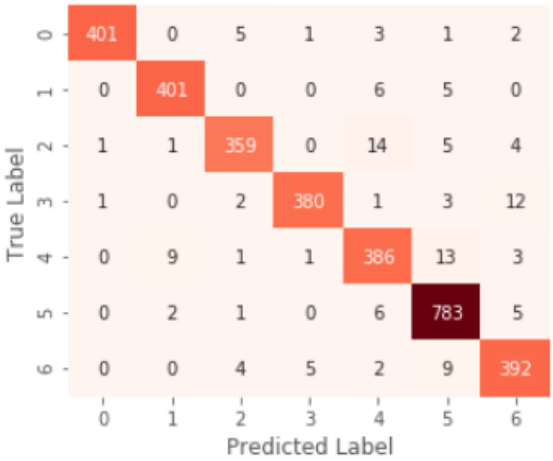

Figure 3-9 Confusion matrix over all subjects of modified Nearlab dataset

According to Figure 3-9, the misclassification of classes does not include any special classes and classifier was able to distinguish all classes almost equally. It is worth mentioning that, since the samples of two classes were combined, the number of samples for this new class labeled as 5 is more than others.

3.3 Run time analysis

An estimation of processing time is needed when approaching an online classification problem. One of the steps that might require high amount of time is feature extraction, which is addressed here. One random subject is chosen and the time needed for extracting each feature of all channels for one sub-window is reported in table 3-6. The reported time is averaged over 1000 repetitions of the procedure. The system used for these measurements was a 64-bit Windows based computer, with a 6 GB RAM and an Intel® Core™ i5 CPU.

Table 3-6 Extraction time (ms) of each feature over all channels

Feature	MAV	ZC	SSC	WL	HP_A	HP_M
Time (ms)	7.925×10^{-2}	8.927×10^{-2}	6.776×10^{-2}	9.979×10^{-2}	8.788×10^{-2}	1.833×10^{-1}

Feature	HP_C	SampEn	CCs	RMS	IEMG	Skew
Time (ms)	2.558×10^{-1}	147.9	125.3	9.5456e-02	3.1459e-02	1.3

Table 3-6 shows that, extracting time-domain features are orders of magnitude faster than extracting frequency-domain features (e.g. CCs). This fact, proves the superior performance of time-domain features in an online classification application. Regarding the feature sets considered in this project, TD and ITD present the fastest solutions since they only contain time-domain features.
In online classification, it is also necessary to limit the required time for producing a prediction over one input sample. Hence, the prediction time of one sample of a random subject over different classifiers has been calculated. The time is averaged

over predicting 1000 samples. The Full feature set was chosen for this comparison since it has all 15 features included. The result is shown in table 3-7.

Table 3-7 Prediction time (ms) of each classifier using Full feature set

Classifier	KNN	LDA	MLP	SVM
Time (ms)	3.554	0.801	1.021	0.879

According to table 3-7, KNN is the slowest classifier. This can be explained by the fact that in KNN classifier, for each prediction it is necessary to calculate the distance of new sample with all training samples, hence making the prediction time dependent to number of training samples and calculating the distances. LDA, MLP and SVM have similar performance regarding predicted time consumption.

4 Deep learning results

4.1 Classification performance analysis

4.1.1 Nearlab dataset

Cnet2D:
The hyper-parameters of this network optimized on Nearlab database was introduced in section 2.3.4. The learning parameters used for training this network on Nearlab dataset are listed in table 4-1. Adam optimizer was used for this and other proposed deep networks. As it can be seen in this network, reduced learning rate has been used. Learning rate decreases with a factor of 0.5, if after 30 epochs the validation loss does not decrease. The minimum of learning rate is set to 0.0005.

Table 4-1 Learning parameters for Cnet2D on Nearlab dataset

Parameter	Value
Learning rate	From 0.003 to 0.0005, factor = 0.5, patience = 30
Epoch	400, early stopping patience = 100
Batch size	128

For each possible combination of learning parameters, the curves for loss and accuracy was investigated and the parameters were modified accordingly. In figure 4-1 loss and accuracy curve of subject number 4 is shown as an example. The loss curve (left) is comparing the loss (total error) of network on training and validation sets through the training epochs. Accuracy curve (right) similarly compares the accuracy of training and validation sets through the training epochs. It is desired to have a decrease in loss and increase in accuracy as the number of epochs increases until both converge to a plateau. Not reaching plateau in the curves could mean the need for more epochs in training procedure. Moreover, the absence of correlation between training and validation curves could indicate presence of over-fitting or under-fitting. A significant bias between two curves or high oscillation in validation curve when the training curve has no considerable oscillations, are two examples of absence of correlation.

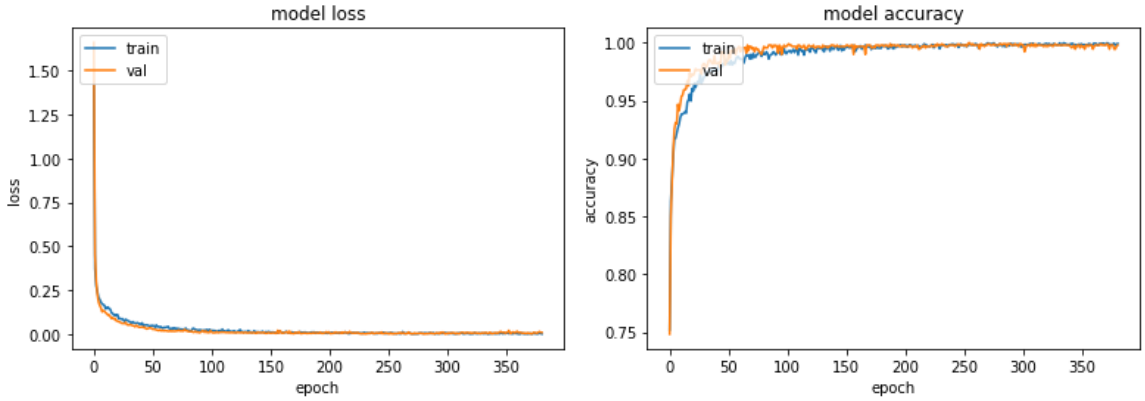

Figure 4-1 Loss (left) and accuracy (right) curves during training of Cnet2D for subject 4 of Nearlab dataset.

In figure 4-2 the classification accuracy over all 11 subjects of Nearlab dataset using the proposed Cnet2D is shown. Since training deep networks have stochastic nature, the classification accuracy of each subject is different every time network is trained. In order to decrease this effect, the network has been trained and tested for

5 times, and the reported values is the average of these 5 repetitions for each subject.

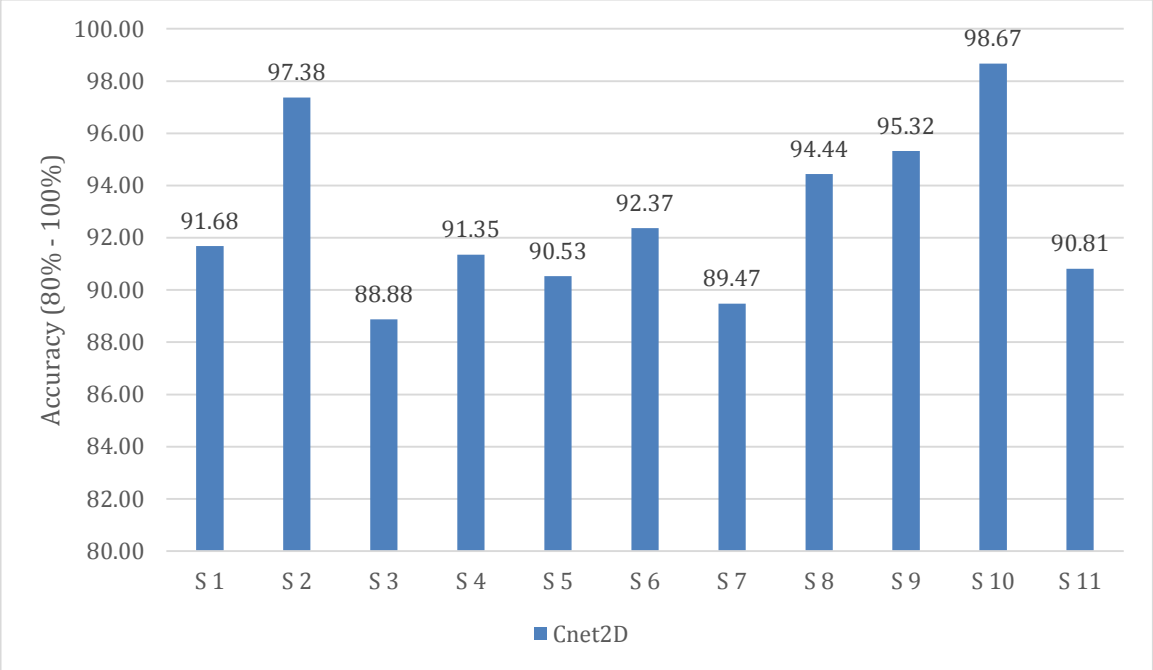

Figure 4-2 Classification accuracy (%) of Cnet2D over Nearlab dataset subjects

Cnet1D:
The learning parameters used for training this network are listed in table 4-2. In this network, a fixed learning rate is used. These parameters have been obtained by trial and error and investigating loss and accuracy, following the same procedure previously illustrated for parameters of Cnet2D.

Table 4-2 Learning parameters for Cnet1D on Nearlab dataset

Parameters	Values
Learning rate	0.001
Epoch	400, early stopping patience = 100
Batch size	128

In figure 4-3 the classification accuracy over all 11 subjects of Nearlab dataset using the proposed Cnet1D is shown. The reported results are the average of 5 repetitions of training and testing the model for each subject in order to decrease the effect of stochastic nature of the model.

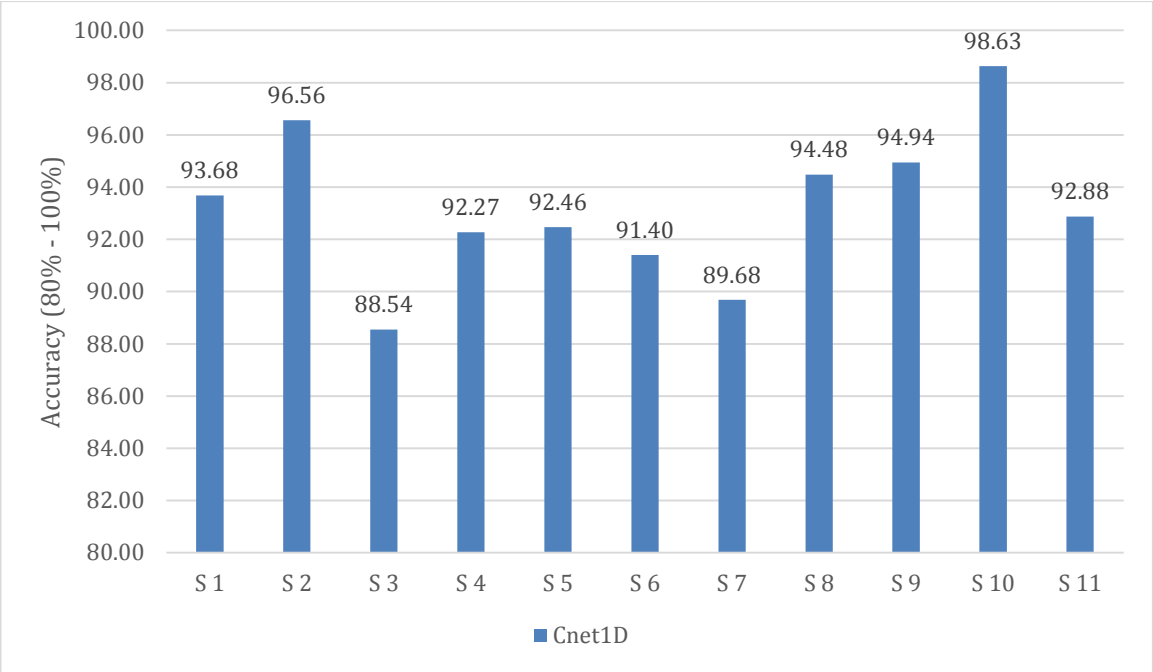

Figure 4-3 Classification accuracy (%) of Cnet1D over Nearlab dataset subjects

CnetComb:
This architecture is a combination of two previous networks, in order to combine the features extracted from 1-dimaensional filters and 2-dimensional filters. The parameters used for training this network are listed in table 4-3. Reducing learning rate approach is used for this network. The parameters search was performed as previous networks.

Table 4-3 Learning parameters for CnetComb on Nearlab dataset

Parameters	Values
Learning rate	From 0.003 to 0.0005, factor = 0.5, patience = 30
Epoch	400, early stopping patience = 100
Batch size	128

In figure 4-4 the classification accuracy over all 11 subjects of Nearlab dataset using the proposed CnetComb is shown. Here again the reported results are the average of 5 repetitions of training and testing.

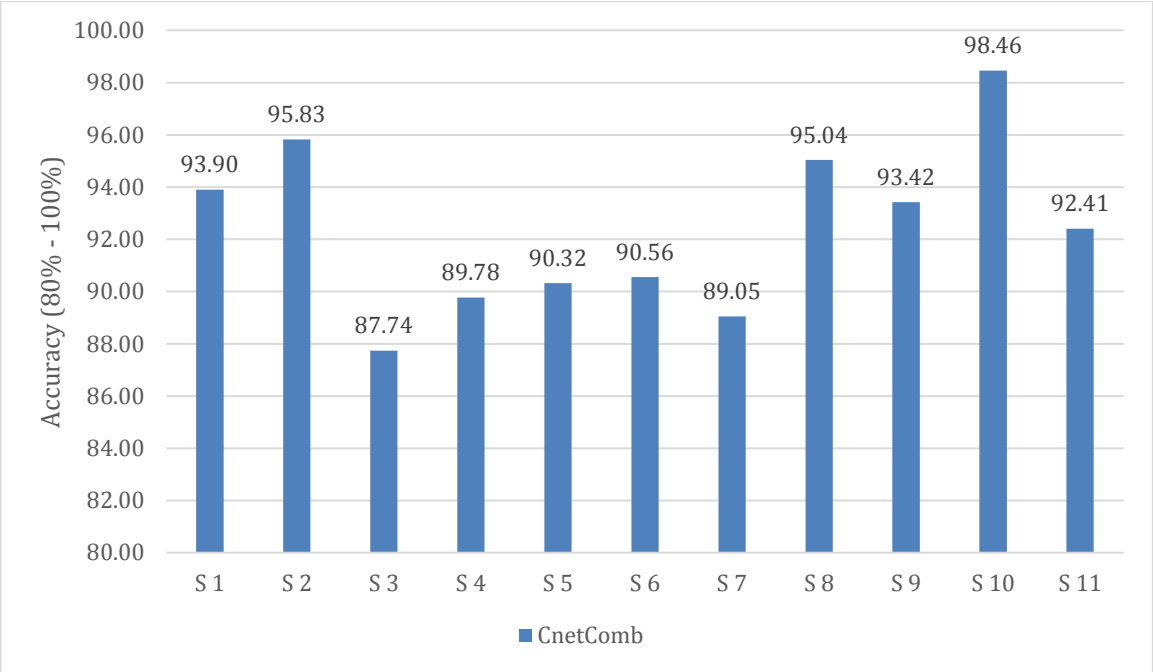

Figure 4-4 Classification accuracy (%) of CnetComb over Nearlab dataset subjects

RESnet:
The parameters used for training this network are listed in table 4-4.

Table 4-4 Learning parameters for RESnet on Nearlab dataset

Parameters	Values
Learning rate	From 0.003 to 0.0005, factor = 0.5, patience = 30
Epoch	400, early stopping patience = 150
Batch size	32

In figure 4-5, the classification accuracy over all 11 subjects of Nearlab dataset using the proposed RESnet is shown. The reported results are the average of 5 repetitions of training and testing of the model for each subject.

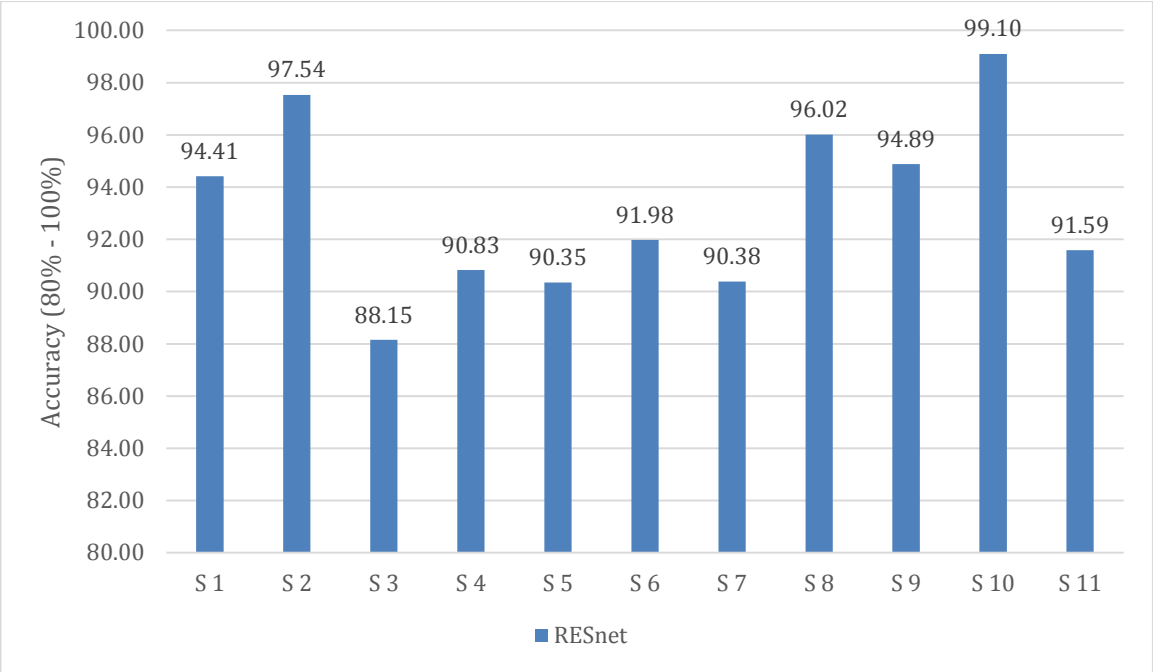

Figure 4-5 Classification accuracy (%) of RESnet over Nearlab dataset subjects

In the following, the 4 proposed networks are compared. Figure 4-6 demonstrates the classification accuracy of the proposed methods on all subjects.

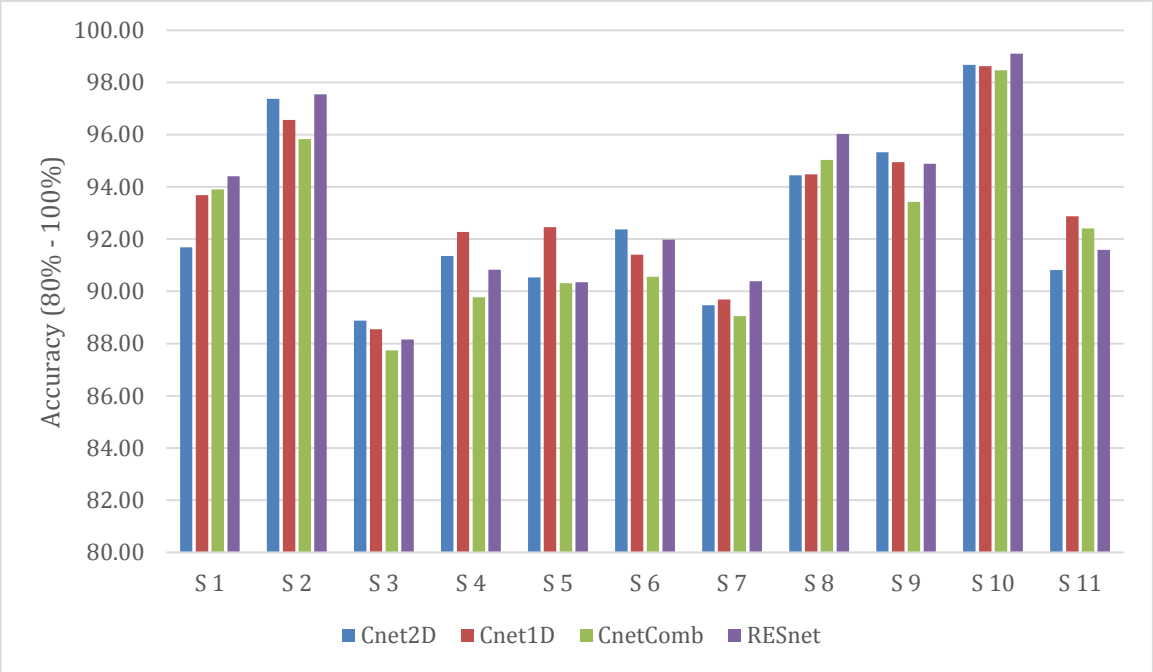

Figure 4-6 Comparison between classifiers via classification accuracy (%) on Nearlab dataset subjects

Table 4-5 compares the average accuracy, standard deviation (std), median and interquartile range (IQR) over all 11 subjects and the statistical significance of these methods. The Pairwise Wilcoxon Signed-Rank Test is applied to compare statistical significance of the classifier with highest average accuracy (Cnet1D) and others.

Hence, the pairs considered for this test are: (Cnet2D, Cnet1D), (RESnet, Cnet1D) and (CnetComb, Cnet1D). (Null hypothesis is rejected when H0 = 0 ($p < 0.05$)).

Table 4-5 Comparison of proposed deep networks on Nearlab database subjects

Classifier	Cnet1D	Cnet2D	CnetComb	RESnet
Average(%)	93.23	92.81	92.41	93.20
Std(%)	2.91	3.22	3.26	3.42
Median (%)	92.88	91.68	92.41	91.98
IQR(%)	2.87	4.21	4.42	4.85
H0 (P-value)	-	1	0(0.014)	1

It can be inferred form table 4-6 that although Cnet1D has the highest average accuracy, there is no significant difference between Cnet1D, Cnet2D and RESnet.

A comparison based on average accuracy and Pairwise Wilcoxon Signed-Rank Test, has been also conducted between Cnet1D as the method with highest average accuracy and all machine learning methods with their best feature set (section 3.1.1) in table 4-6. The considered pairs for Wilcoxon tests are: (Cnet1D vs LDA), (Cnet1D vs MLP), (Cnet1D vs SVM) and (Cnet1D vs KNN)

Table 4-6 Comprison of best models of classical machine learning and Cnet1D on Nearlab database

Classifier	Cnet1D	LDA with Full	MLP with ITD	SVM with ITD	KNN with ITD
Average(%)	93.23	92.55	91.45	91.72	89.20
Std(%)	2.91	4.23	2.97	3.60	4.35
Median(%)	92.88	93.76	91.24	90.75	88.69
IQR(%)	2.87	5.39	3.83	6.09	6.57
H0 (P-value)	-	1	0(0.007)	0(0.014)	0(0.005)

According to table 4-6 the Cnet1D, has the highest average when comparing with all the traditional classifiers. The variance analysis indicates that, the Cnet1D has significant difference (improvement) when comparing to MLP, SVM and KNN methods while there is no significance when comparing to LDA method. A noteworthy point is that, deep learning method is extracting features from raw signal, which is in time-domain, while LDA is using frequency-domain features.

4.1.2 Ninapro DB2 dataset

On this database, Cnet1D, Cnet2D, CnetComb and RESnet has been applied. However, since electrodes placement, acquisition protocol and movement classes are different, some minor modifications should be considered to fit the models on DB2 database. The most important difference between Ninapro DB2 and Nearlab dataset is that the number of samples per class, is significantly higher in the Nearlab dataset comparing to DB2. Therefore, even in the case of minor modifications, sometimes the proposed networks show overfitting on Ninapro DB2, since the hyper-parameters of base models were designed according to Nearlab dataset characteristics (e.g., number of samples per target class). Another important difference is the total number of movement classes (8 in Nearlab vs. 17 in Ninapro). The hyper-parameter modifications of Cnet2D, Cnet1D, CnetComb and RESnet networks are reported in table 4-7 to 4-10, respectively. The modifications in hyper-parameters on Ninapro DB2 were performed with the goal of decreasing the number of trainable parameters in order to prevent overfitting. As a result, the size of stride and maxpool was changed by trial and error.

Moreover, learning parameters of all the networks are listed in tables 4-11.

Table 4-7 Hyper-parameters of Cnet2D for Ninapro DB2 dataset

	Hyper-parameter	Block number: Value
Feature extractor stage	Filter numbers of 3 Conv blocks	1: 32, 2: 48, 3: 64
	Filter kernel sizes of 3 Conv blocks	1: (3,13), 2: (3,9), 3: (3,5)
	Filter stride sizes of 3 Conv blocks	1: (1,3), 2: (1,1), 3: (1,1)
	Max pooling sizes of 3 Conv blocks	1: (1,3), 2: (1,3), 3: (1,3)
Classifier stage	Dense layer sizes of 2 FC blocks	1: 300, 2: 50
General hyper-parameters	RRelu parameter range	All: 1/8 to 1/7
	Dropout rate	All: 0.3

Table 4-8 Hyper-parameters of network Cnet1D for Ninapro DB2 dataset

	Hyper-parameter	Block number: Value
Feature extractor stage	Filter numbers of 3 Conv blocks	1: 32, 2: 48, 3: 64
	Filter kernel sizes of 3 Conv blocks	1: (1,13), 2: (1,9), 3: (1,5)
	Filter stride sizes of 3 Conv blocks	1: (1,3), 2: (1,1), 3: (1,1)
	Max pooling sizes of 3 Conv blocks	1: (1,3), 2: (1,3), 3: (1,3)
Classifier stage	Dense layer sizes of 2 FC blocks	1: 300, 2: 50
General hyper-parameters	RRelu parameter range	All: 1/8 to 1/7
	Dropout rate	All: 0.3

Table 4-9 Hyper-parameters of network CnetComb for Ninapro DB2 dataset

		Hyper-parameter	Block number: Value
Feature extractor stage	Cnet2D feature extractor	Filter numbers of 3 Conv blocks	1: 32, 2: 48, 3: 64
		Filter kernel sizes of 3 Conv blocks	1: (3,13), 2: (3,9), 3: (3,5)
		Filter stride sizes of 3 Conv blocks	1: (1,3), 2: (1,1), 3: (1,1)
		Max pooling sizes of 3 Conv blocks	1: (1,3), 2: (1,3), 3: (1,3)
	Cnet1D feature extractor	Filter numbers of 3 Conv blocks	1: 32, 2: 48, 3: 64
		Filter kernel sizes of 3 Conv blocks	1: (1,13), 2: (1,9), 3: (1,5)
		Filter stride sizes of 3 Conv blocks	1: (1,3), 2: (1,1), 3: (1,1)
		Max pooling sizes of 3 Conv blocks	1: (1,3), 2: (1,3), 3: (1,3)
Classifier stage		Dense layer sizes of 2 FC blocks	1: 300, 2: 50
General hyper-parameters		RRelu parameter range	All: 1/8 to 1/7
		Dropout rate	All: 0.3

Table 4-10 Hyper-parameters of network RESnet for Ninapro DB2 dataset

		Hyper-parameter	Block number: Value
Feature extractor stage	Conv blocks	Filter numbers of Conv blocks 1,2 and 3	1: 24, 2: 32, 3: 24
		Filter kernel sizes of Conv blocks 1,2 and 3	1: (1,7), 2: (1,5), 3: (1,5)
		Filter stride sizes of Conv blocks 1,2 and 3	1: (1,5), 2: (1,1), 3: (1,5)
		Max pooling sizes of Conv blocks 1,2 and 3	1: None, 2: (1,4), 3: (1,4)
	Conv layer α	Filter number	48
		Filter kernel size	(1,3)
		Filter stride size	(1,1)
	β block	Average pooling size	(1,3)
Classifier stage		Dense layer sizes of 2 FC blocks	1: 300, 2: 50
General hyper-parameters		RRelu parameter range	All: 1/10 to 1/9
		Dropout rate	All: 0.3

Table 4-11 Learning parameters for deep networks on Ninapro DB2 dataset

Parameter	Value
Learning rate	0.0001 (for Cnet1D, Cnet2D, CnetComb) / 0.00005 (for RESnet)
Epoch	400, early stopping patience = 100 (for all)
Batch size	128 (for all)

As mentioned before, the curves for loss and accuracy were investigated, in order to modify learning parameters accordingly. In figure 4-7 loss (left) and accuracy (right) curves of subject 2 during training Cnet2D is shown. According to figure 4-7 there is very low correlation between training and validation accuracy/loss. The absence of correlation could be an indicator of overfitting. Overfitting effect was predictable due to the fact that the number of samples per class is much lower in Ninapro database comparing to Nearlab database while the model complexity is the same in both cases. It is important to mention that other subjects also displayed similar patterns in their accuracy/loss curves.

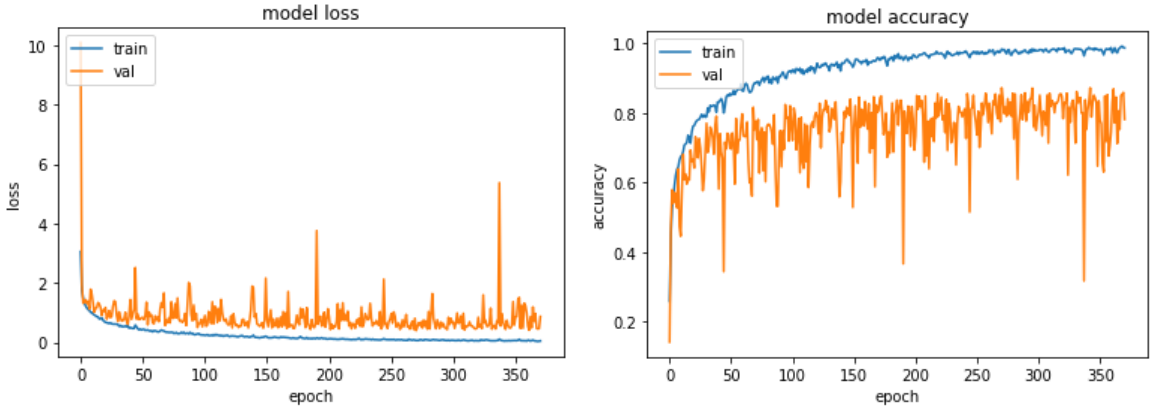

Figure 4-7 Loss (left) and accuracy (right) curves during training of Cnet2D on random subject of Ninapro DB2

In table 4-12 the classification accuracy of Ninapro DB2 subjects on the 4 networks is reported. For each subject, the best classification accuracy is shown in bold.

Table 4-12 Classification accuracy (%) of all Ninapro DB@ subjects

	Cnet1D	Cnet2D	CnetComb	**RESnet**		Cnet1D	Cnet2D	CnetComb	RESnet
S1	84.16	**87.50**	83.94	82.10	**S21**	70.29	69.21	**74.46**	70.14
S2	82.61	**83.35**	79.15	79.20	**S22**	77.39	79.21	77.8	**79.90**
S3	**90.28**	88.98	84.82	86.37	**S23**	75.18	**77.99**	77.77	76.48
S4	**76.67**	76.52	73.62	70.98	**S24**	**67.81**	67.32	67.12	65.20
S5	82.64	**86.36**	82.42	81.98	**S25**	76.98	79.42	**79.66**	76.44
S6	79.5	**81.71**	79.88	74.97	**S26**	79.96	84.48	**85.94**	80.44
S7	72.06	**77.84**	74.83	68.25	**S27**	**81.64**	79.85	79.77	75.75
S8	89.76	90.84	89.36	**91.64**	**S28**	65.07	66.85	66.72	**67.11**
S9	86.4	**87.79**	85.14	84.87	**S29**	78.1	**79.99**	79.41	77.00
S10	83.43	**86.56**	84.68	84.47	**S30**	71.71	**76.93**	74.18	69.83
S11	70.51	**73.36**	71.44	71.35	**S31**	64.88	72.83	**73.93**	63.05
S12	78.2	**81.32**	79.99	78.24	**S32**	74.62	79.17	**80.46**	76.52
S13	78.83	**80.92**	78.55	76.66	**S33**	84.93	89.44	**90.36**	89.09
S14	71.99	**74.75**	71.09	69.24	**S34**	81.31	83.82	**85.37**	84.24
S15	83.46	**86.75**	84.27	82.90	**S35**	67.55	73.92	**78.58**	64.34
S16	80.82	80.00	**81.89**	81.28	**S36**	69.41	73.59	**74.53**	68.36
S17	80.48	**83.77**	83.63	81.41	**S37**	80.32	84.38	**85.53**	79.35

S18	73.37	75.46	**76**	73.90	**S38**	74.46	82.49	**85.9**	78.53
S19	80.17	81.15	**82.63**	79.86	**S39**	74.26	81.95	**83.87**	76.70
S20	**76.31**	73.41	75.24	73.06	**S40**	90.04	92.68	**93.18**	90.17

Moreover, table 4-13 compares the average accuracy and standard deviation over all subjects and the statistical significance of these methods. The Pairwise Wilcoxon Signed-Rank Test is applied to compare performance of Cnet2D vs. Cnet1D and Cnet2D vs. CnetComb and Cnet2D vs. RESnet. Cnet2D was chosen as the base for comparison since it has the highest average accuracy. (Null hypothesis is rejected when H0 = 0 ($p < 0.05$)).

Table 4-13 Comparison of deep networks over Ninapro DB2 subjects

Classifier	Cnet1D	Cnet2D	CnetComb	RESnet
Average(%)	77.69	**80.34**	79.93	77.03
Std(%)	6.54	6.30	5.98	7.09
Median(%)	78.15	80.46	79.82	76.85
IQR(%)	8.84	8.15	9.23	10.29
H0 (P-value)	0(0.000)	-	0(0.001)	0(0.000)

From table 4-12 it can be inferred that, Cnet2D has the highest average accuracy and has significant statistical difference comparing to others. Moreover, a comparison based on classification accuracy and statistical test (Pairwise Wilcoxon Signed-Rank Test), has been also conducted between the best classical machine learning methods (section 3.1.2) and the best deep learning method (Cnet2D) in table 4-14. The pairs considered for Wilcoxon test are: (Cnet2D vs LDA), (Cnet2D vs MLP), (Cnet2D vs SVM) and (Cnet2D vs KNN). According to table 4-13, although MLP has highest average accuracy, MLP, LDA and Cnet2D have no significant statistical difference in their performance. The lower average accuracy of Cnet2D comparing MLP could be due to overfitting problem of Ninapro DB2 database using a deep learning method.

Table 4-14 Comprison of best models of classical machine learning and Cnet2D on Ninapro DB2 database

Classifier	Cnet2D	LDA with Full	MLP with Full	SVM with Full	KNN with Full
Average(%)	80.34	79.95	**80.97**	79.50	75.17
Std(%)	6.30	5.73	5.44	6.25	6.52
Median(%)	80.46	80.62	80.92	79.98	75.29
IQR(%)	8.15	7.45	6.79	9.55	9.48
H0 (P-value)	-	1	1	0(0.015)	0(0.000)

4.2 Class specific analysis

Similar to section 3.2 the classification performance should also be investigated by the means of confusion matrix on Nearlab dataset. In order to do so, the confusion matrix of all subjects of Nearlab dataset has been obtained using Cnet1D classifier and averaged. In the figure 4-8, the averaged confusion matrix over all 11 subjects, can be seen. According to this figure, the misclassification between the two pinch classes labeled 5 and 6 remains a problem in deep learning as well.

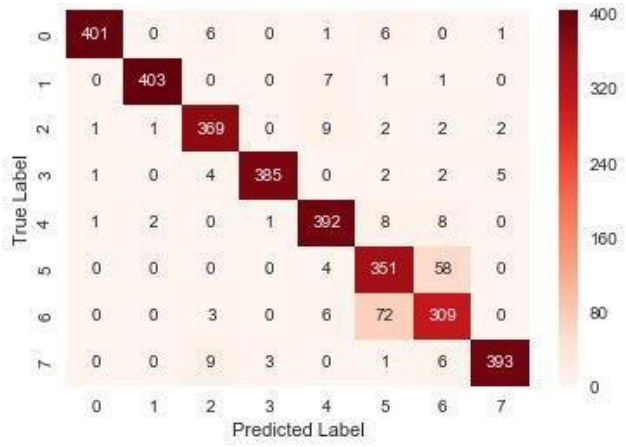

Figure 4-8 Confusion matrix of Cnet2D averaged over Nearlab database subjects

Hence, the suggested modification in section 3.2.1 will be repeated here on Nearlab dataset. The two classes of pinch will be treated as one single class in training and prediction and the averaged confusion matrix of new results over all 11 subjects is shown in figure 4-9.

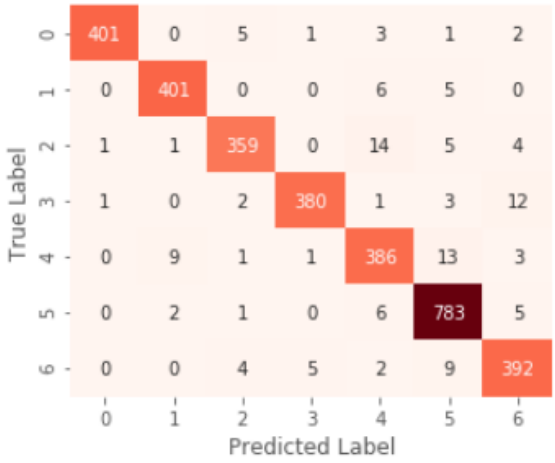

Figure 4-9 Confusion matrix of Cnet1D averaged over modified database

The corresponding classification accuracy of this modified database using Cnet1D over all Nearlab dataset subjects is presented in figure 4-10. The average accuracy over all subjects is increased to 96.5% using this modified movement classes.

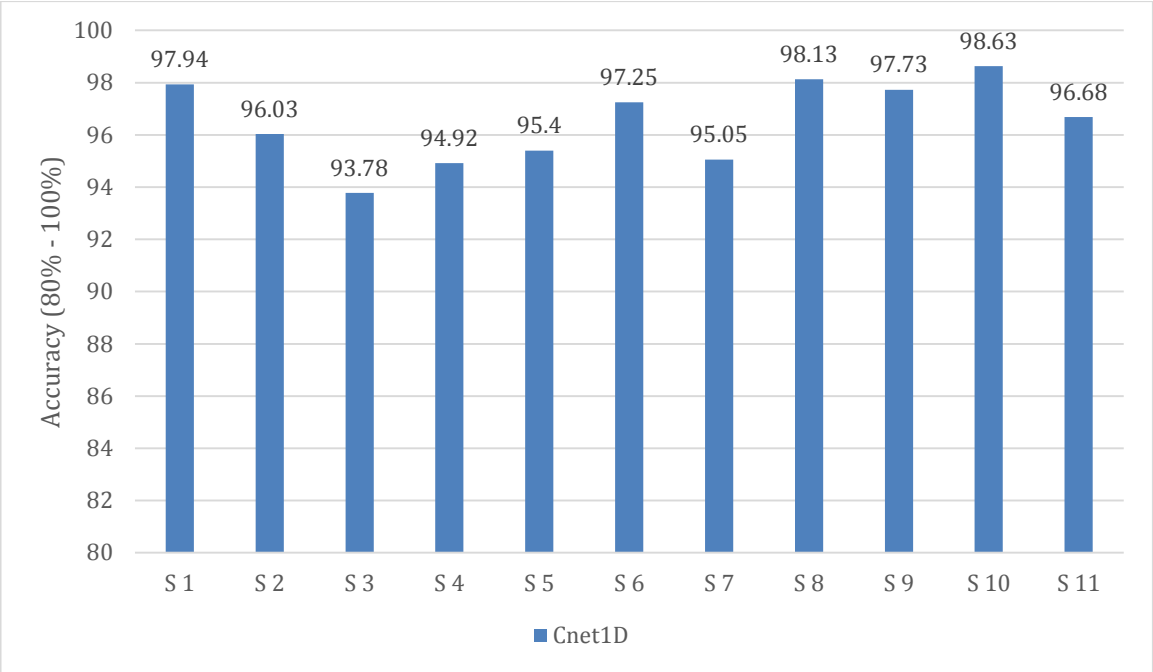

Figure 4-10 Classification accuracy (%) over all subjects using modified databse

The classification result on this modified database over Nearlab dataset subjects, using Cnet1D and LDA with Full feature set is compared in table 4-15. The pairwise Wilcoxon test between these two methods have also been conducted. From this table it can be inferred that although Cnet1D has higher accuracy, it does not have significant statistical difference comparing LDA with Full feature set. The same conclusion was made before modifying the database.

Table 4-15 Comparing Cnet1D and LDA on modified Nearlab dataset

Classifier	Cnet1D	LDA with Full
Average(%)	96.50	96.00
Std(%)	1.57	2.09
Median(%)	96.68	95.73
IQR(%)	2.61	3.33
H0 (P-value)	-	1

4.3 Run time analysis

In online classification, it is necessary to limit the required time for producing a prediction over one input sample. Hence, the prediction time of one sample of a random subject over different classifiers has been calculated. The time is averaged over predicting 1000 samples. In table 4-16, the prediction time of each classifier is presented. Moreover, since the prediction time depends on the number of the total parameters of the model, this number is also reported. Deep learning training and testing were executed on google Colaboratory Jupyter notebook service [63]. The allocated GPU from google Colaboratory to this project while running the code for time calculation was a Tesla P100-PCIE-16GB.

Table 4-16 Prediction time (ms) comparison over all deep networks

Classifier	Cnet1D	Cnet2D	CnetComb	RESnet
Time (ms)	4.81	4.97	6.34	5.16
Number of total parameters	1,583,022	1,642,222	3,208,086	822,908

From table 4-16 it can be inferred that all networks produce sample predictions fast enough to meet the time limitations of online classification, provided that a GPU similar to the one in this test (Tesla P100-PCIE-16GB) is utilized.

4.4 Transfer learning

In this study, transfer learning (TL) has been employed as an approach to solve the challenge of limited available data to train deep neural networks. TL is used to leverage the shared information among different subjects to obtain bigger training dataset.

4.4.1 Nearlab dataset

The two TL methods introduced in section 2.2.5 will be trained and tested on Nearlab database and the result will be reported in this section. The base architecture for both of the methods are Cnet1D, which exhibits the best average accuracy on Nearlab dataset.

Freeze & fine-tune: In this study, the source domain is the data of 5 selected subjects with the highest single subject accuracy (acquired by Cnet1D) except the target subject. While, target domain is corresponding data to target subject. (For more detail about training procedure please refer to section 2.2.5). The architecture of the feature extractor and classifier stage is based on Cnet1D, which was concluded to be the best deep network in this project. Hence, the hyper-parameters and the architecture of the network remains the same, table 2-6 and figure 2-23, respectively. However, the training parameters change according to the problem. Table 4-17 refers to learning parameters of this network.

Table 4-17 Learning parameter of Freeze & fine-tune network for Nearlab database

	Parameters	Values
Source Network	Learning rate	0.001
	Epoch	100, early stopping patience = 50
	Batch size	128
Target Network	Learning rate	From 0.001 to 0.0001, factor = 0.5, patience = 30
	Epoch	300, early stopping patience = 150
	Batch size	128

In figure 4-11 the classification accuracy of the source network (Cnet1D) and its comparison when applying freeze & fine-tune (target network) over all subjects can be seen.

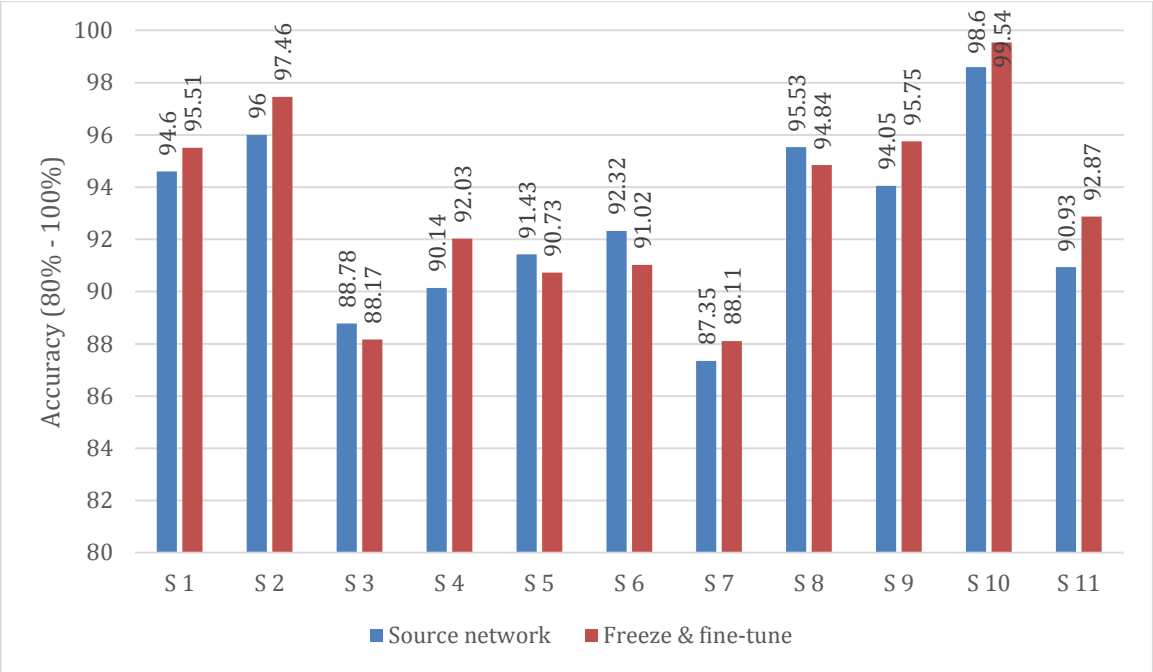

Figure 4-11 Classification accuracy (%) of Freeze & fine-tune method on Nearlab database subjects

Parallel networks: This network utilizes combinations of features learned by two separate networks. The target network uses the architecture of Cnet1D which was chosen as the best deep network for Nearlab dataset subjects, hence the hyper-parameters are equal to Cnet1D (table 2-6). On the other hand, the source network has 4 convolutional blocks as feature extractor stage. The feature extractor stage of final architecture is based on two previous networks, while the classifier stage is 2 fully connected blocks with random initial values. In table 4-18 and table 4-19 the hyper-parameters of source and final network can be found, respectively.

Table 4-18 Hyper-parameters of source network of Parallel method

	Hyper-parameter	Block number: Value
Feature extractor stage	Filter numbers of 4 Conv blocks	1: 32, 2: 48, 3: 56, 4: 64
	Filter kernel sizes of 4 Conv blocks	1: (1,11), 2: (1,9), 3: (1,7), 4: (1,5)
	Filter stride sizes of 4 Conv blocks	1: (1,1), 2: (1,1), 3: (1,1), 4: (1,1)
	Max pooling sizes of 4 Conv blocks	1: (1,4), 2: (1,4), 3: (1,2), 4: (1,2)
Classifier stage	Dense layer sizes of 2 FC blocks	1: 300, 2: 50
General hyper-parameters	RRelu parameter range	All: 1/8 to 1/7
	Dropout rate	All: 0.5

Table 4-19 Hyper-parameters of subject network of Parallel method

		Hyper-parameter	Value
Feature extractor stage	Source network	Feature extractor stage of table 4-21	
	Target network	Feature extractor stage of table 4-3	
Classifier stage		Dense layer sizes of 2 FC blocks	1: 150, 2: 50
General hyper-parameters		RRelu parameter range	All: 1/8 to 1/7
		Dropout rate	All: 0.5

In figure 4-12 a schematic of the final network including the hyper-parameters is shown.

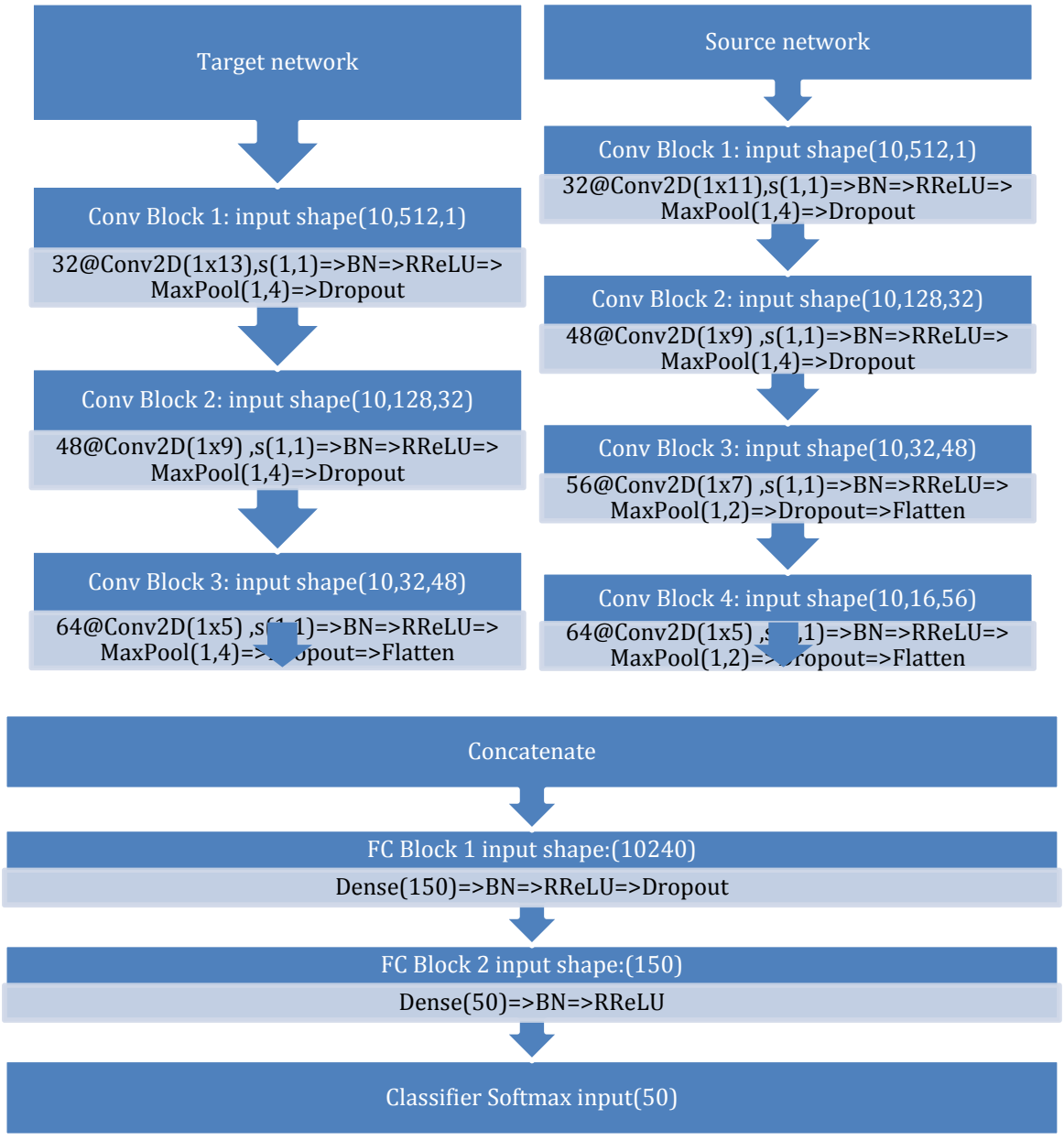

Figure 4-12 Schematic of target network architecture of Parallel method

The learning parameters of target network is same as Cnet1D listed in table 4-2. The learning parameters of source and final networks are reported in tables 4-20 and 4-21, respectively.

Table 4-20 Learning parameters of source network from Parallel method

Parameters	Values
Learning rate	0.003
Epoch	100, early stopping patience = 50
Batch size	128

Table 4-21 Learning parameters of target network from Parallel method

Parameters	Values
Learning rate	0.001
Epoch	200, early stopping patience = 80
Batch size	128

The result of classification accuracy of target network (Cnet1D) and its improvement thorough applying parallel networks method over Nearlab database subjects is demonstrated in figure 4-13.

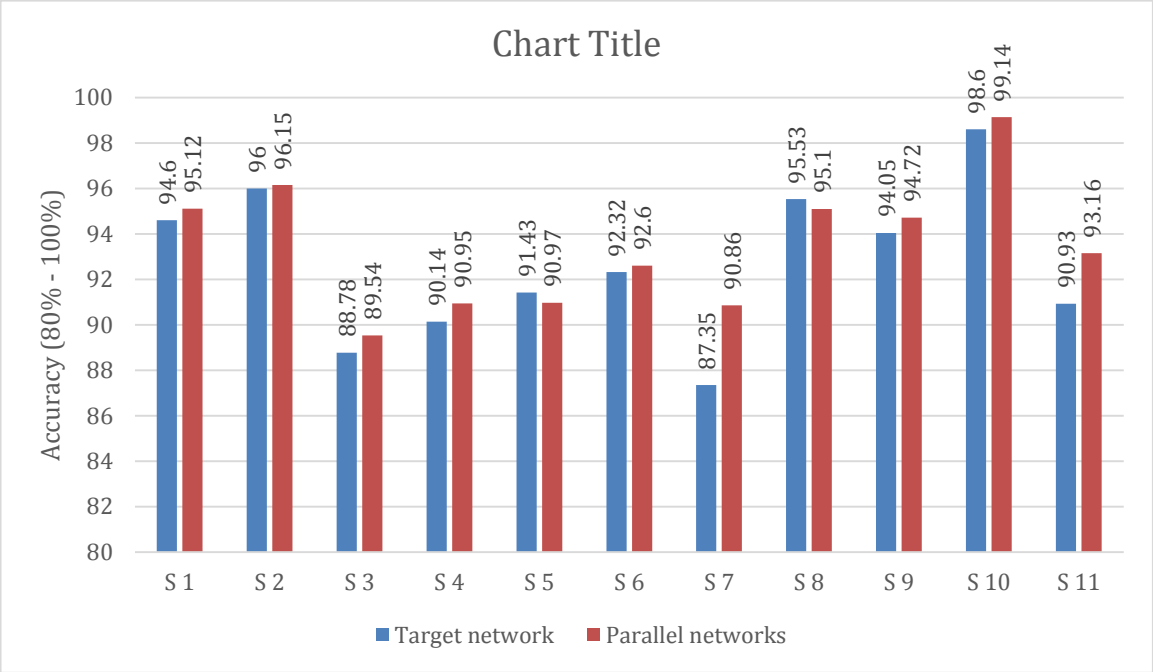

Figure 4-13 Classification accuracy (%) of Parallel networks method over all Nearlab dataset subjects

4.4.2 With and without transfer learning

Figure 4-14 displays classification accuracy of all subjects using the two proposed TL methods. Since the same Cnet1D model was used for further applying the two TL methods, a direct comparison can be considered to conclude which method improved the subjects' classification accuracy.

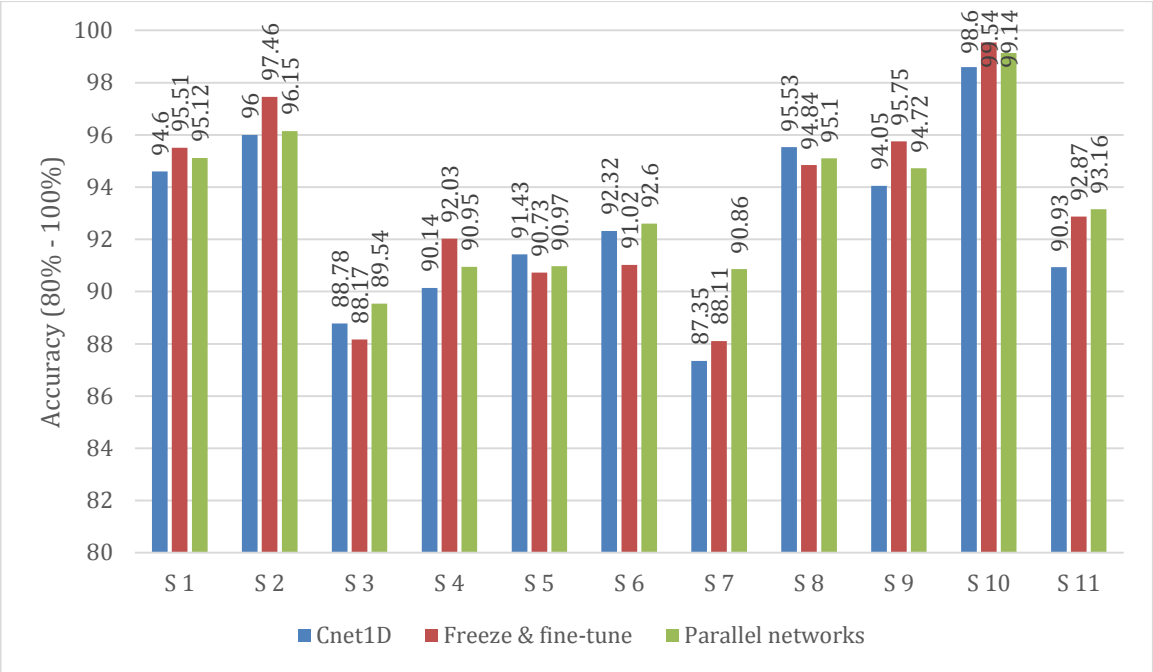

Figure 4-14 Comparison of classification accuracy (%) on Cnet1D and two TL methods

In table 4-22 this comparison was made with the average accuracy and statistical analysis. The pairwise Wilcoxon statistical analysis has been applied to investigate the significance of the two methods comparing to the Cnet1D without employing transfer learning. The considered pairs are (Cnet1D vs. Freeze & fine-tune), (Cnet1D vs. Parallel networks).

Table 4-22 Comparison of Freeze & fine-tune and Parallel networks

Classifier	Cnet1D	Freeze & fine-tune	Parallel networks
Average(%)	92.70	93.27	**93.48**
Std(%)	3.38	3.69	2.86
Median (%)	92.32	92.87	93.16
IQR (%)	4.53	4.76	4.15
H0 (P-value)	-	1(0.083)	0(0.019)

According to table 4-22, parallel networks method has improved the average accuracy from 92.70% to 93.48% and it has significant statistical difference comparing Cnet1D.

4.4.3 Run time analysis

It is necessary to make sure these methods can satisfy the time limitation condition for online applications. One random subject is chosen and the time needed for producing prediction along with total number of parameters related to each network is reported in table 4-23. The reported time is averaged over 1000 repetitions of the procedure. The GUI appointed by google colab for this procedure was Tesla P100-PCIE-16GB.

Table 4-23 Time performance comparison of TL methods and Cnet1D

Classifier	Cnet1D	Freeze & fine-tune	Parallel networks
Time (ms)	4.81	2.15	4.21
Number of total parameters	1,583,022	1,597,342	1,626,044

From table 4-23 it can be inferred that both two transfer learning methods provide the sample prediction fast enough to meet the time requirements of online applications.

5 Simulating online classification

The initial goal of the thesis was to perform online classification experiments. Unfortunately, due to the outbreak of the Coronavirus disease (COVID-19) the possibility of executing of such experiments was limited. Thus, this section of the thesis has changed to simulating online classification. We have used the raw recordings from Nearlab dataset and went through the signal with a real time speed, reading samples at their correct time and classifying them in real time. Every 62.5ms, the program should read its last 250ms available data and classify the window. It is important to mention the recordings used in online classifications are the primary raw EMG data acquired from the Porti device without any filtering or manipulation.
The graphical interface has been coded in Python. PyQtGraph library [64] has been used for fast plotting in order to achieve real-time plotting and classification.

5.1 Control panel

In what follows, options available in the control panel will be discussed.

- Start: This button will start the program to go through the recording.
- Subject: This button is for choosing the recordings among the 11 available subjects' datasets.
- Filters: This section is for controlling the filters applied to the signal.
- Channels: This section offers the option to visulize or hide each channel.
- Speed: This section controls the speed of going through the recording. The users can choose between normal speed, 1/2, 1/4 and 1/8 normal speed.
- Exit: Exit button is for quiting the application.

5.2 Plots

Several plots have been implemented in this program. All these plots are completely synchronized in time and are as follows:

- Live 10-channel data: This plot is dedicated to displaying the EMG data. It can show all the selected channles. The yellow line in the middle indicates the exact time of that point in the recording. While this plot also shows 1 second after the yellow line (just for visualiztion purposes), the classification algorithm is only using the data before yellow line. This plot has also interactive mouse functions with which users can change the visualization of the data.
- Live video cue: This plot shows the protocol and instructions shown to the subject in time. Blue rectangles represent free periods. Green rectangles represent rest periods. Yellow rectangles are representing the movements and the instructed class of movement is written inside the yellow rectangles.
- Live classifiction results: This plot is dedicated to display the classification results. The classifier has access only to the data shown before the yellow line. The green rectangles show rest class, while white rectangle with numbers inside them show movement classes.

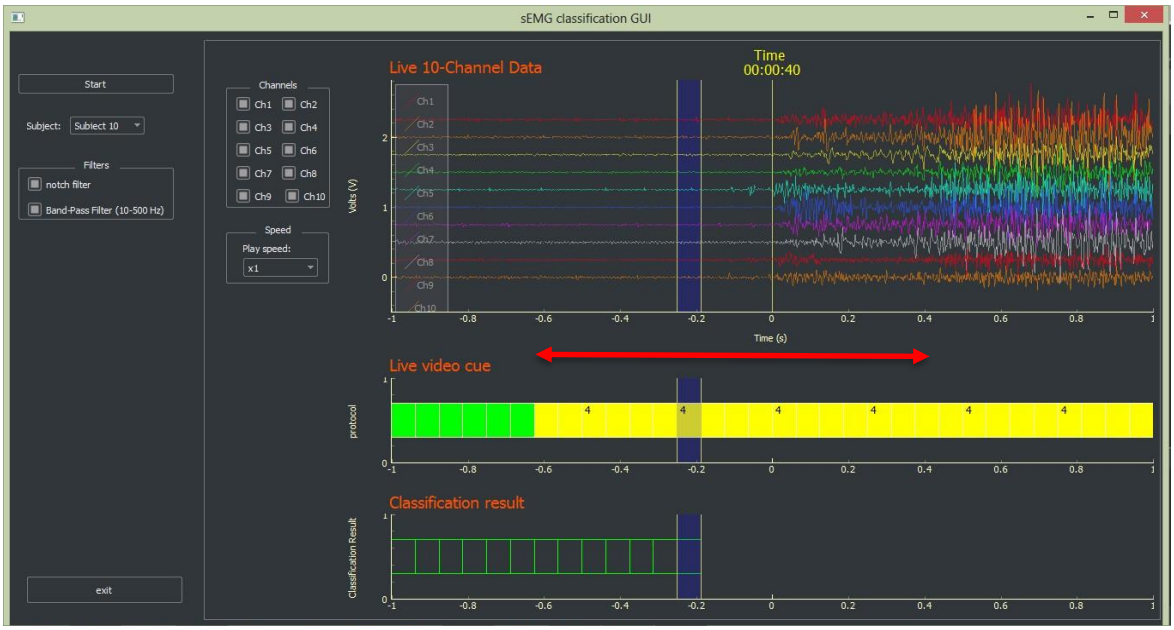

Figure 5-1 A picture of the user interface program

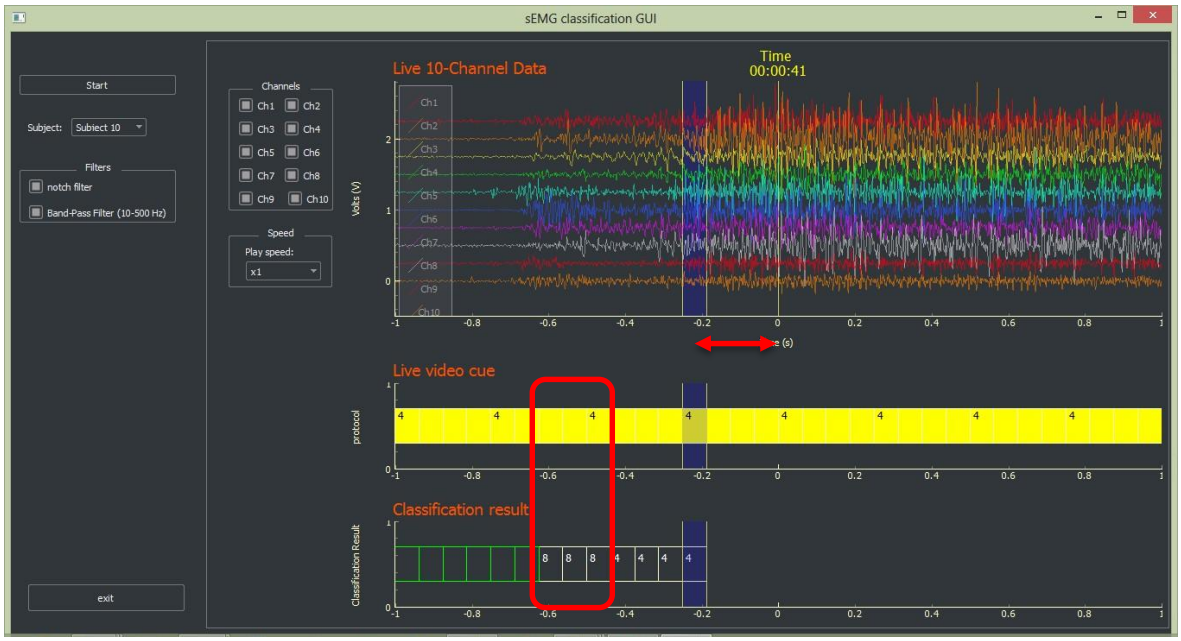

Figure 5-2 A picture of user interface, showing movement classification

5.3 Real-time classification

The classifier used in this program is LDA, since it outperformed other traditional classification methods and also is one of the fast ones to execute. The classification algorithm does not use any knowledge about the video cues and their timings in the classification. Each 512 samples window (250ms) is regarded as a window to classify. A threshold is defined as 3 times of the mean absolute values in a rest period. Each window is checked with this threshold, if it is higher than the threshold, it will be classified as one of the movements using LDA. If not, it will be classified as rest. That is why it may happen that, in a period of 5 seconds rest, the classifier detects a movement class from one or more of the 125ms windows included in that

period or it detects rest class windows in the middle of a movement period. The protocol plot is for the user to see which movement was shown to the subject at each moment in time.

It can be seen in the Figures 5-1 and 5-2 that the last classification result has a certain distance from the yellow line (marked with a two headed red arrow in Figure 5-2). The reason for this delay, is that the classifier is using the majority voting method over 62.5ms slots of time. Thus in order to acquire all the votes from different 250ms windows that include the targeted 62.5ms slot, 187.5ms should pass after that slot. Therefore, this algorithm introduces a 187.5ms delay in classification of a 62.5ms slot. Figure 2-15 in section 2.3.3 represents an example of the delay imposed by majority voting.

Moreover, it can be seen from the Figures 5-1 that there is a delay between the video cue and start of the actual movement indicated by increased activity in EMG signals (marked with a two headed red arrow in Figure 5-1). This delay can be associated with subjects' decision making time and electro-mechanical delay after receiving the visual cue.

Furthermore, the classifier produces wrong results more frequently in the start and ending of a movement. The red rectangle in Figure 5-2 is an indication of this statement, since first classification results are class 8 while video cue was class 4. This phenomenon can be related to the fact that the stationary characteristics of EMG signal was the main focus of this study. Thus the start and ending of the movements are more prone to be classified incorrectly due to their transient characteristics which the classifiers are not trained for.

6 Conclusion

In this thesis work, multiple pattern recognition techniques have been designed, implemented and tested in order to classify 8 hand movement classes using surface EMG signals. Classical machine learning methods along with deep learning techniques are employed for this task. To tackle the problem of training deep networks with limited databases, transfer learning approach has been applied. Moreover, a graphical user interface has been designed to classify acquired signal in real-time.

A comprehensive comparison between classical machine learning methods and deep learning methods is conducted. In addition, the feasibility of employing such methods is investigated using run-time analysis. It has been shown that when considering fast algorithms (using time domain features), deep learning algorithm out performs classical machine learning algorithms with no significant increase of cost in terms of time consumption.

Transfer learning approach has been also deployed to answer the following question: Could knowledge learned from other subjects' EMG data be useful to train a classifier for target subject? In order to answer, 2 methods are designed and implemented to integrate other subjects' knowledge into the classifier for the targeted subject. It has been demonstrated by this study that the proposed transfer learning algorithms are able to improved performance in terms of accuracy.

Another conclusion drawn from this thesis work is the importance of choosing the correct movement classes to classify. In this study, it has been shown that by combining two very similar movement classes (pinch and lateral pinch) the accuracy of the classifier is increased by 3-4%, which is significant improvement when considering accuracies above 90%.

Discussion:

Looking at results from classical machine learning algorithms, the classification performance of 4 proposed classifiers combined with 4 feature sets on all Near dataset subjects, demonstrated that LDA with full feature set obtains the best average accuracy of 92.55%. However, pairwise Wilcoxon Signed-Rank Test proved that there is no statistical significance between LDA combined with Full feature set and MLP or SVM combined with ITD feature set. From the time consumption point of view, calculation of ITD features are orders of magnitude faster than Full. Therefore, in an online application using MLP or SVM with ITD feature set might have higher priority over LDA with Full feature set.

Comparing the classification results of same classifiers and feature sets on Ninapro DB2 revealed that, the best combination was MLP with 80.97% using Full feature set. This demonstrates the high potential of MLP classifier as reliable classifier. When comparing the results of the Ninapro DB2 and Near database, the number of movement classes, which is much higher in Ninapro DB2 (17 as opposed to 8) should also be considered.

It is important to mention, in an online classification scheme, all the processing steps needed for classifying a new sample should be performed in a limited amount of time, defined by the requirements of the problem. In the case of hand movement online classification which was targeted in this study, the time limitation is 62.5 ms (already mentioned in section 2.3.2). According to run time analysis, section (3.3) using time-domain features will easily meet this limitation. However, frequency-domain features exceed the time limitation. A noteworthy point is that the system used in this study is a regular laptop with medium performance. A high performance

system or using accelerated algorithms in feature extraction, could potentially solve this problem.
Moving forward to deep learning algorithms, the results from section 4 reveal that out of 4 proposed networks, Cnet1D and RESnet both show high accuracy, 93.23% and 93.20% respectively. The results obtained by Cnet1D is compared with results obtained with traditional machine learning algorithms. Although the average accuracy is improved comparing to all 4 traditional classifiers, statistical analysis only provides evidence of performance improvement when comparing to MLP, SVM and KNN. An important point to mention is that the best performing LDA algorithm was using Full feature (frequency and rime domain) set while other classifiers performed best when using ITD feature set (only time domain).
Looking at runtime analysis of deep learning algorithms we can conclude that the time consumption of all proposed deep learning algorithms falls well within the time limitations of online classification, if a similar GPU to the one used in this study (mentioned in chapter 4) is employed.
When comparing the performance of all the developed classifiers considering both accuracy and time consumption, the proposed deep convolutional neural network referred to as “Cnet1D” outperforms all other algorithms having 93.23% accuracy and a runtime equal to 4.81ms in average per sample, while LDA needs about 270ms only to calculate its 15 features.
The same networks were also tested on publicly available sEMG dataset for hand and wrist movements “Ninapro DB2”. Cnet2D architecture achieved highest accuracy of 80.34%. Statistical analysis indicated no significant statistical difference between best deep learning method and best classical approach in the case of Ninapro dataset. However, probing the loss/accuracy curves obtained from training procedure of the convolutional neural networks, revealed that the proposed methods are probably over-fitting on Ninapro DB2 dataset. Ninapro DB2 has 17 different hand movements and 6 repetitions per movement class. This means comparing to Near dataset, Ninapro has more than twice the number of movement classes and less than half the repetitions per movement available. Thus, it’s logical to conclude that networks designed for Near dataset are over-fitting on the Ninapro dataset.
Class specific analysis was repeated for deep learning algorithms on Near dataset. the same modification proposed before (combining the two pinch classes) was applied and an increase of more than 3% was achieved. Cnet1D architecture was able to obtain 96.5% accuracy on the modified Near dataset with 7 movements.
Finally, two transfer learning methods were proposed to take advantage of knowledge learned from other subjects’ EMG data in order to improve performance in the target subject classification problem. Variance analysis revealed that the performance of Cnet1D classifier has significant statistical improvement when used with transfer learning algorithm proposed in this study named “Parallel networks”. The other method introduced as “Freeze & fine-tune” also improved the average accuracy of Cnet1D classifier however there was no significant statistical difference between the performance of augmented and not-augmented Cnet1D network. Looking at run-time analysis of transfer learning methods, both algorithms meet the specifications required for online classification implementation, having a runtime of under 5ms per sample.

Comparison with related work:
SEMG hand movement database, Ninapro DB2, introduced by Atzori et al. [6] has been used by many researchers as a benchmark. Zhai et al. [7] classified spectrograms of DB2 sEMG signals combined with PCA for dimension reduction.

They used SVM as classifier and obtained 75.74% accuracy on exercise B (same exercise used in this study). Later, they improved their work [8] by proposing a self-recalibrating CNN to eliminate the need of user training over time. The classification accuracy of their new method on Ninapro DB2 was 82.22%, when tested on exercise B. Moreover, in 2019 Huang et al. [9] used a CNN-LSTM network in order to fully capture the spatial and temporal features of sEMG spectrograms, the resulting accuracy on DB2 exercise B was 80.93%.
Table 3-8 compares these related research works with the best obtained classification accuracies obtained in this study. Specifically, MLP with Full feature set from classical machine learning techniques and Cnet2D from deep learning techniques. All the reported results are on 17 movements of exercise B and on 40 subjects. According to this table, although Zhai et al [8] has the nest average accuracy, proposed MLP with Full feature set shows comparable results with respect to other state of the art studies. This approach has the advantage of faster execution time while utilizing simpler algorithms. The deep network applied on Ninapro DB2 did not improve the results due to possible over-fitting over data.

Table 6-1 Classification accuracy (%) comparison with related work over Ninapro DB2 exeB

	Classification method	Accuracy (%)
Related work	Spectrogram with SVM [7]	75.74
	Spectrogram with CNN [8]	82.22
	Spectrogram with CNN and LSTM [9]	80.93
This study	MLP with Full feature set	**80.97**
	Cnet2D	80.34

Côté-Allard et al. [5] have also used transfer learning as an approach to solve the issue of training deep networks for databases with limited amount of data in sEMG hand movement classification. Their defined database includes 7 movement classes: neutral, hand close, hand open, wrist extension, wrist flexion, ulnar deviation, radial deviation. They proposed a transfer learning approach called progressive neural network which after pre-training the network on source domain, trains another network for the new task with random initialization and then connects these two networks laterally to each layer. When applying this method on raw EMG data, their classification accuracy increased from 97.08% to 97.39%. In this study, the proposed parallel networks method applied on the Nearlab dataset increased the accuracy of a Cnet1D model from 92.70% to 93.48%. The reason that the overall accuracy of Nearlab dataset is lower than their proposed database could be the presence of two pinch movement classes. This was evident when the two pinch classes combined and the accuracy of Cnet1D improved to 96.5%.

Future work:

- One of the advantages of using neural networks, as a pattern recognition algorithm, is that in addition to assigning a class to each input sample, it can provide a series of probabilities of the sample belonging to each target class. This advantage can prove very beneficial in the case of classifying hand movements based on sEMG data. As an example, consider the classification problem in this thesis. The original movement class groups were 8 commonly used hand/wrist movements including pinch and wrist rotation. However, the combination of pairs of the original movements could be also very useful. For example, the combination of pinch and wrist rotation could mimic holding a

key and locking/unlocking a door. Of course adding all the combinations of 8 movements to the problem would increase the number of target classes leading to a significantly harder classification problem. However, by making use of the probabilities provided by the neural networks output Softmax layer, we could try to classify combination of movements using a network that is only trained on the original movement classes. Using a threshold on the output probabilities (e. g., 0.3), we can declare 2 output classes for one input sample and thus creating combination movement outputs based on a given dataset of original movement for training.
In the future studies the proposed method could be used to cover a broader range of hand/wrist movements using limited databases for training neural networks.

- As mentioned before, the graphical user interface designed in this thesis was not tested in an actual real-time classification scheme. Thus, a natural path to continue this thesis work is to recruit subjects to test the classification algorithms developed in this thesis in real-time. The performance of the classifier can be improved in an online setting, due to user adaptation derived by visual feedback. In an offline classification algorithm, the user is not informed about the output of the classifier while in an online classification setting the user could use the visual feedback to adopt the muscle activation patterns to increase the classifier accuracy and reliability.

- The transfer learning algorithms developed in this thesis work are just 2 examples of methods trying to integrate the knowledge learned in another domain into the target domain. There are several other transfer learning algorithms proposed by the literature. In the following paragraph, 2 of the most promising ones from the view point of authors of this thesis are proposed for future work.

  1- Deep learning algorithms are intensively explored in the field of image classification. As oppose to sEMG signal classification, creating large databases to train very deep networks is not a significant challenge in majority of image classification problems. Having this fact in mind, it seems logical to use transfer learning to transfer the knowledge from an image classifier to our problem of sEMG classification. However, one problem remains, which is the intrinsic difference between images and 1D sEMG time series data. This problem could be solved using a representation of the EMG data other than the raw signals, such as spectrograms. Spectrograms represent the time and frequency content of a time series signal in the form of an image. With the new image representation of the sEMG data available, the very deep networks for image classification problems trained on millions of images can be used to extract detailed features from sEMG data and improve classification accuracy. In [60] authors employed short-time Fourier transform to represent EMG signals as a time-frequency image. Then pre-trained convolutional neural networks (popular pre-trained image classifiers) are used to extract features from the time-frequency image.

  2- Another interesting transfer learning approach is to use classifiers trained on other subjects directly on the data acquired from the target subject,

without further modifications on target subject. Without doubt, these classifiers will have poor results since they are tuned on the training data from other subjects. However, the accumulation of multiple weak learners could create a very strong learner. One approach to test this theory is to use majority voting on the outputs of all the weak networks. Authors of [65] proposed a similar approach. In this paper multiple supportive convolutional neural networks are pre-trained on EMG data of multiple subjects and fine-tuned on the targeted subject. The final output is obtained by voting the supportive networks' predictions.

Reference list

[1] Oskoei, M.A. and Hu, H., 2007. Myoelectric control systems—A survey. *Biomedical signal processing and control*, *2*(4), pp.275-294.

[2] Farina, D., Jiang, N., Rehbaum, H., Holobar, A., Graimann, B., Dietl, H. and Aszmann, O.C., 2014. 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*, *22*(4), pp.797-809.

[3] Abbaspour, S., Lindén, M., Gholamhosseini, H., Naber, A. and Ortiz-Catalan, M., 2020. Evaluation of surface EMG-based recognition algorithms for decoding hand movements. *Medical & Biological Engineering & Computing*, *58*(1), pp.83-100.

[4] Atzori, M., Cognolato, M. and Müller, H., 2016. Deep learning with convolutional neural networks applied to electromyography data: A resource for the classification of movements for prosthetic hands. *Frontiers in neurorobotics*, *10*, p.9.

[5] Côté-Allard, U., Fall, C.L., Drouin, A., Campeau-Lecours, A., Gosselin, C., Glette, K., Laviolette, F. and Gosselin, B., 2019. Deep learning for electromyographic hand gesture signal classification using transfer learning. *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, *27*(4), pp.760-771.

[6] Atzori, M., Gijsberts, A., Castellini, C., Caputo, B., Hager, A.G.M., Elsig, S., Giatsidis, G., Bassetto, F. and Müller, H., 2014. Electromyography data for non-invasive naturally-controlled robotic hand prostheses. *Scientific data*, *1*(1), pp.1-13.

[7] Zhai, X., Jelfs, B., Chan, R.H. and Tin, C., 2016, August. Short latency hand movement classification based on surface EMG spectrogram with PCA. In *2016 38th annual international conference of the IEEE engineering in medicine and biology society (EMBC)* (pp. 327-330). IEEE.

[8] Zhai, X., Jelfs, B., Chan, R.H. and Tin, C., 2017. Self-recalibrating surface EMG pattern recognition for neuroprosthesis control based on convolutional neural network. *Frontiers in neuroscience*, *11*, p.379.

[9] Huang, D. and Chen, B., 2019, September. Surface EMG Decoding for Hand Gestures Based on Spectrogram and CNN-LSTM. In *2019 2nd China Symposium on Cognitive Computing and Hybrid Intelligence (CCHI)* (pp. 123-126). IEEE.

[10] Twente Medical Systems International B.V. (TMSi). https://www.tmsi.com/. Accessed: June 04,2020

[11] Stegeman, D. and Hermens, H., 2007. Standards for surface electromyography: The European project Surface EMG for non-invasive assessment of muscles (SENIAM).

[12] DELSYS wearable sensors for movement sciences. https://delsys.com/. Accessed: June 04,2020

[13] Wang, J., Tang, L. and Bronlund, J.E., 2013. Surface EMG signal amplification and filtering. *International Journal of Computer Applications*, *82*(1).

[14] Hudgins, B., Parker, P. and Scott, R.N., 1993. A new strategy for multifunction myoelectric control. *IEEE transactions on biomedical engineering*, *40*(1), pp.82-94.

[15] Englehart, K., Hudgins, B., Parker, P.A. and Stevenson, M., 1999. Classification of the myoelectric signal using time-frequency based representations. *Medical engineering & physics*, *21*(6-7), pp.431-438.

[16] Phinyomark, A., Quaine, F., Charbonnier, S., Serviere, C., Tarpin-Bernard, F. and Laurillau, Y., 2013. EMG feature evaluation for improving myoelectric pattern recognition robustness. *Expert Systems with applications*, *40*(12), pp.4832-4840.

[17] Vercellis, C., 2009. *Business intelligence: data mining and optimization for decision making* (pp. 1-420). New York: Wiley.

[18] Kaggle: world's largest data science community. https://www.kaggle.com/. Accessed: June 04,2020.

[19] Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I. and Salakhutdinov, R., 2014. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, *15*(1), pp.1929-1958.

[20] Ioffe, S. and Szegedy, C., 2015. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*.

[21] Kingma, D.P. and Ba, J., 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

[22] He, K., Zhang, X., Ren, S. and Sun, J., 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 770-778).

[23] Stifani, N., 2014. Motor neurons and the generation of spinal motor neurons diversity. *Frontiers in cellular neuroscience*, *8*, p.293.

[24] Merletti, R. and Parker, P.J. eds., 2004. *Electromyography: physiology, engineering, and non-invasive applications* (Vol. 11). John Wiley & Sons.

[25] Goyal, N. and Chad, D.A., 2014. Motor Control, Peripheral.

[26] Day, S., 2002. Important factors in surface EMG measurement. *Bortec Biomedical Ltd publishers*, pp.1-17.

[27] Nazmi, N., Abdul Rahman, M.A., Yamamoto, S.I., Ahmad, S.A., Zamzuri, H. and Mazlan, S.A., 2016. A review of classification techniques of EMG signals during isotonic and isometric contractions. *Sensors*, *16*(8), p.1304.

[28] Popovic, M.B., 2019. *Biomechatronics*. Academic Press.

[29] Al-Timemy, A.H., Bugmann, G., Escudero, J. and Outram, N., 2013. Classification of finger movements for the dexterous hand prosthesis control with surface electromyography. *IEEE journal of biomedical and health informatics*, *17*(3), pp.608-618.

[30] Corne, S.A., 1996. Artificial neural networks for pattern recognition. *Concepts in Magnetic Resonance*, *8*(5), pp.303-324.

[31] Dudani, S.A., 1976. The distance-weighted k-nearest-neighbor rule. *IEEE Transactions on Systems, Man, and Cybernetics*, (4), pp.325-327.

[32] Cao, J. and Sanders, D.B., 1996. Multivariate discriminant analysis of the electromyographic interference pattern: statistical approach to discrimination among controls, myopathies and neuropathies. *Medical and Biological Engineering and Computing*, *34*(5), pp.369-374.

[33] Maat, B., Smit, G., Plettenburg, D. and Breedveld, P., 2018. Passive prosthetic hands and tools: A literature review. *Prosthetics and orthotics international*, *42*(1), pp.66-74.

[34] Wang, A., Yuan, W., Liu, J., Yu, Z. and Li, H., 2009. A novel pattern recognition algorithm: Combining ART network with SVM to reconstruct a multi-class classifier. *Computers & Mathematics with Applications*, *57*(11-12), pp.1908-1914.

[35] McCulloch, W.S. and Pitts, W., 1990. A logical calculus of the ideas immanent in nervous activity. *Bulletin of mathematical biology*, *52*(1-2), pp.99-115.

[36] Rosenblatt, F., 1957. *The perceptron, a perceiving and recognizing automaton Project Para*. Cornell Aeronautical Laboratory.

[37] Gómez, R., 2018. Understanding categorical cross-entropy loss, binary cross-entropy loss, Softmax loss, logistic loss, focal loss and all those confusing names. *URL: https://gombru. github. io/2018/05/23/cross_ entropy_loss/(visited on 29/03/2019)*.

[38] Rawat, W. and Wang, Z., 2017. Deep convolutional neural networks for image classification: A comprehensive review, Neural computing.

[39] Ameri, A., Akhaee, M.A., Scheme, E. and Englehart, K., 2019. Regression convolutional neural network for improved simultaneous EMG control. *Journal of neural engineering*, *16*(3), p.036015.

[40] Hubel, D.H. and Wiesel, T.N., 1962. Receptive fields, binocular interaction and functional architecture in the cat's visual cortex. *The Journal of physiology*, *160*(1), p.106.

[41] Ge, W. and Yu, Y., 2017. Borrowing treasures from the wealthy: Deep transfer learning through selective joint fine-tuning. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 1086-1095).

[42] Cote-Allard, U., Fall, C.L., Campeau-Lecours, A., Gosselin, C., Laviolette, F. and Gosselin, B., 2017, October. Transfer learning for sEMG hand gestures recognition using convolutional neural networks. In *2017 IEEE International Conference on Systems, Man, and Cybernetics (SMC)* (pp. 1663-1668). IEEE.

[43] Zhuang, F., Qi, Z., Duan, K., Xi, D., Zhu, Y., Zhu, H., Xiong, H. and He, Q., 2019. A comprehensive survey on transfer learning. *arXiv preprint arXiv:1911.02685*.

[44] Khushaba, R.N. and Kodagoda, S., 2012, December. Electromyogram (EMG) feature reduction using mutual components analysis for multifunction prosthetic fingers control. In *2012 12th International Conference on Control Automation Robotics & Vision (ICARCV)* (pp. 1534-1539). IEEE.

[45] Park, K.H. and Lee, S.W., 2016, February. Movement intention decoding based on deep learning for multiuser myoelectric interfaces. In *2016 4th International Winter Conference on Brain-Computer Interface (BCI)* (pp. 1-2). IEEE.

[46] Buongiorno, D., Cascarano, G.D., Brunetti, A., De Feudis, I. and Bevilacqua, V., 2019, August. A survey on deep learning in electromyographic signal analysis. In *International Conference on Intelligent Computing* (pp. 751-761). Springer, Cham.

[47] ur Rehman, M.Z., Gilani, S.O., Waris, A., Niazi, I.K. and Kamavuako, E.N., 2017, December. A novel approach for classification of hand movements using surface EMG signals. In *2017 IEEE International Symposium on Signal Processing and Information Technology (ISSPIT)* (pp. 265-269). IEEE.

[48] He, Y., Fukuda, O., Bu, N., Okumura, H. and Yamaguchi, N., 2018, July. Surface emg pattern recognition using long short-term memory combined with multilayer perceptron. In *2018 40th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)* (pp. 5636-5639). IEEE.

[49] Shim, H.M., An, H., Lee, S., Lee, E.H., Min, H.K. and Lee, S., 2016. EMG pattern classification by split and merge deep belief network. *Symmetry*, *8*(12), p.148.

[50] Zia ur Rehman, M., Waris, A., Gilani, S.O., Jochumsen, M., Niazi, I.K., Jamil, M., Farina, D. and Kamavuako, E.N., 2018. Multiday EMG-based classification of hand motions with deep learning techniques. *Sensors*, *18*(8), p.2497.

[51] Du, Y., Jin, W., Wei, W., Hu, Y. and Geng, W., 2017. Surface EMG-based inter-session gesture recognition enhanced by deep domain adaptation. *Sensors*, *17*(3), p.458.

[52] Sosin, I., Kudenko, D. and Shpilman, A., 2018, November. Continuous gesture recognition from sEMG sensor data with recurrent neural networks and adversarial domain adaptation. In *2018 15th International Conference on Control, Automation, Robotics and Vision (ICARCV)* (pp. 1436-1441). IEEE.

[53] Vergara, M., Sancho-Bru, J.L., Gracia-Ibáñez, V. and Pérez-González, A., 2014. An introductory study of common grasps used by adults during performance of activities of daily living. *Journal of Hand Therapy*, *27*(3), pp.225-234.

[54] Englehart, K., Hudgin, B. and Parker, P.A., 2001. A wavelet-based continuous classification scheme for multifunction myoelectric control. *IEEE Transactions on Biomedical Engineering*, *48*(3), pp.302-311.

[55] Hjorth, B., 1970. EEG analysis based on time domain properties. *Electroencephalography and clinical neurophysiology*, *29*(3), pp.306-310.

[56] Scikit-learn: python machine learning library. https://scikit-learn.org/stable/. Accessed: June 04,2020.

[57] TensorFlow: A python machine leaning library. https://www.tensorflow.org/. Accessed: June 04,2020.

[58] Keras: A python machine learning library. https://keras.io/. Accessed: June 04,2020.

[59] Xu, B., Wang, N., Chen, T. and Li, M., 2015. Empirical evaluation of rectified activations in convolutional network. *arXiv preprint arXiv:1505.00853*.

[60] Demir, F., Bajaj, V., Ince, M.C., Taran, S. and Şengür, A., 2019. Surface EMG signals and deep transfer learning-based physical action classification. *Neural Computing and Applications*, *31*(12), pp.8455-8462.

[61] Scheff, S.W., 2016. *Fundamental statistical principles for the neurobiologist: A survival guide*. Academic Press.

[62] McDonald, J.H., 2009. *Handbook of biological statistics* (Vol. 2, pp. 6-59). Baltimore, MD: sparky house publishing.

[63] Google Colaboratory: free Jupyter notebook environment provided by Google. https://colab.research.google.com/notebooks/intro.ipynb. Accessed: June 04,2020.

[64] PyQtGraph: Python library for graphics and GUI. http://www.pyqtgraph.org/. Accessed: June 04,2020.

[65] Kim, K.T., Guan, C. and Lee, S.W., 2019. A Subject-Transfer Framework Based on Single-Trial EMG Analysis Using Convolutional Neural Networks. *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, *28*(1), pp.94-103.

List of Figures

Figure 0-1 Electrode positioning 9
Figure 0-2 Movement classes 9
Figure 0-3 Architecture of Cnet2D 13
Figure 0-4 Architecture of CnetComb 13
Figure 0-5 Architecture of RESnet 14
Figure 0-6 Architecture of Freeze & fine-tune method 15
Figure 0-7 Architecture of Parallel networks method 15
Figure 1-1 Motor unit and its components [24] 25
Figure 1-2 Needle electrode (right) (https://www.ambu.com/neurology/emg-electrodes/product/ambu-neuroline-concentric) and surface electrode (left) (https://bio-medical.com/covidien-kendall-disposable-surface-emg-ecg-ekg-electrodes-1-3-8-35mm-50pkg.html) 25
Figure 1-3 A sample of surface EMG signal 26
Figure 1-4 An example of passive prosthetic hand (http://rehabindy0.tripod.com/be-passive.html) 26
Figure 1-5 An example of body-powered ahnd prosthesis (www.trsprosthetics.com/) 27
Figure 1-6 An example of how the electonic powere hand prosthesis is actuated (https://www.rehab.research.va.gov) 27
Figure 1-7 Diagram of typical pattern recognition-based hand prosthesis [1] 28
Figure 1-8 K-NN classification algorithm for samples with 2 features and K=3 29
Figure 1-9 The optimal seperating line and maximum margin in a 2-D input space [17] 30
Figure 1-10 Operating model of perceptron network [17] 32
Figure 1-11 Multilayer network [30] 33
Figure 1-12 categorical cross-entropy is Softmax plus cross-entropy [37] 34
Figure 1-13 General CNN architecture [38] 35
Figure 1-14 Convolution of a 2-D input data with a (3×3) filter [39] 35
Figure 2-1 Movement classes selected for experimental protocol 39
Figure 2-2 Hand open movement in 3 predefined hand orientations 40
Figure 2-3 One movement block (10 seconds in total) 40
Figure 2-4 Construction of whole protocol for basic movements: Basic movement protocol is divided in 3 rounds and each round consists of 5 trial 40
Figure 2-5 Trial time organization 41
Figure 2-6 Electrode positions 42
Figure 2-7 Hand stance 43
Figure 2-8 Raw EMG signal acquired by one channel during the entire experiment session 44
Figure 2-9 Filtered data vs raw data 45
Figure 2-10 Frequency content of the signal before (top) and after (down) filtering 45
Figure 2-11 Diagram of on-set or ending detedction algorithm 46
Figure 2-12 7 second windows super imposed from one channel in different movements 47
Figure 2-13 The process of on-set and ending detection 47
Figure 2-14 the process of on-set and ending detection and removing the transiant signal 48
Figure 2-15 An example of majority voting method for online classification 49

Figure 2-16 MAV features for 10 channels calculated for the multiple windows belonging to pronation movement 51
Figure 2-17 MAV features for 10 channels calculated for the multiple windows belonging to supination movement 52
Figure 2-18 MAV features for 10 channels calculated for the multiple windows belonging to hand open movement 52
Figure 2-19 Cross correlation between features 53
Figure 2-20 the effect of scaling on the ranges of the features 54
Figure 2-21 An example of Cnet2D/Cnet1D architectures 56
Figure 2-22 Schematic of Cnet2D architecture for Nearlab dataset 58
Figure 2-23 Schematic of Cnet1D architecture for Nearlab dataset 59
Figure 2-24 CnetComb general architecture 59
Figure 2-25 Schematic of CnetComb architecture on Nearlab dataset 60
Figure 2-26 A general residual neural network architecture 61
Figure 2-27 General schematic of proposed RESnet structure 62
Figure 2-28 Schematic of RESnet architecture on Nearlab dataset 63
Figure 2-29 Freeze & fine-tune architecture 64
Figure 2-30 Parallel networks architecture 65
Figure 2-31 Example of confusion matrix 66
Figure 3-1 Comparison between average classification accuracies of Nearlab dataset 68
Figure 3-2 Comparison between feature sets for each subject of Nearlab dataset, using LDA classifier 69
Figure 3-3 Comparison between average classification accuracies of Ninapro DB2 database 70
Figure 3-4-a Classification accuracy (%) of Ninapro DB2 subjects 1-10 with MLP 71
Figure 3-5 Confusion matrix averaged over all 11 subjects of Nearlab dataset ... 73
Figure 3-6 confusion matrixes for subject 10 (left) and subject 7 (right) 74
Figure 3-7 Round 1 class distributions of subject 10 (left) and subject 7 (right) ... 75
Figure 3-8 Classification accuarcy of LDA with Full feature set, using modified dataset 75
Figure 3-9 Confusion matrix over all subjects of modified Nearlab dataset 76
Figure 4-1 Loss (left) and accuracy (right) curves during training of Cnet2D for subject 4 of Nearlab dataset. 78
Figure 4-2 Classification accuracy (%) of Cnet2D over Nearlab dataset subjects 79
Figure 4-3 Classification accuracy (%) of Cnet1D over Nearlab dataset subjects 80
Figure 4-4 Classification accuracy (%) of CnetComb over Nearlab dataset subjects 81
Figure 4-5 Classification accuracy (%) of RESnet over Nearlab dataset subjects 82
Figure 4-6 Comparison between classifiers via classification accuracy (%) on Nearlab dataset subjects 82
Figure 4-7 Loss (left) and accuracy (right) curves during training of Cnet2D on random subject of Ninapro DB2 85
Figure 4-8 Confusion matrix of Cnet2D averaged over Nearlab database subjects 87
Figure 4-9 Confusion matrix of Cnet1D averaged over modified database 87
Figure 4-10 Classification accuracy (%) over all subjects using modified databse 88
Figure 4-11 Classification accuracy (%) of Freeze & fine-tune method on Nearlab database subjects 90
Figure 4-12 Schematic of target network architecture of Parallel method 91

Figure 4-13 Classification accuracy (%) of Parallel networks method over all Nearlab dataset subjects 92
Figure 4-14 Comparison of classification accuracy (%) on Cnet1D and two TL methods 93
Figure 5-1 A picture of the user interface program 96
Figure 5-2 A picture of user interface, showing movement classification 96

List of tables

Table 0-1 Selected feature sets 11
Table 0-2 Classifiers and thier selected hyper-parameters 11
Table 0-3 Extraction time (ms) of each feature over all channels 16
Table 0-4 Prediction time (ms) of each traditional classifier using Full feature set (should be added to feature set calculation time) 16
Table 0-5 Prediction time (ms) of all deep learning networks 16
Table 0-6 Classification results on Nearlab dataset 17
Table 0-7 Classification results on Ninapro DB2 dataset 17
Table 0-8 Classification accuracy (%) comparison with related work over Ninapro DB2 18
Table 2-1 Preprocessing filter specifications 44
Table 2-2 Feature sets 53
Table 2-3 Range of hyper-parameter for classifiers 55
Table 2-4 Selected hyper-parameters 55
Table 2-5 Hyper-parameters of architecture Cnet2D on Nearlab dataset 57
Table 2-6 Hyper-parameters of architecture Cnet1D for Nearlab dataset 58
Table 2-7 Hyper-parameters of architecture CnetComb on Nearlab dataset 59
Table 2-8 Hyper-parameters of architecture RESnet on Nearlab dataset 62
Table 2-9 Summary of explored methods 65
Table 3-1 Classification accuracy over all 11 subjects of Nearlab dataset 67
Table 3-2 Comparison between classifiers and feature sets on Nearlab dataset. 68
Table 3-3 Comparison of classifiers with their best performing feature sets 69
Table 3-4 Comparison between classifiers and feature sets on Ninapro DB2..... 70
Table 3-5 Comparison of classifiers with their best performing feature sets 71
Table 3-6 Extraction time (ms) of each feature over all channels 76
Table 3-7 Prediction time (ms) of each classifier using Full feature set..... 77
Table 4-1 Learning parameters for Cnet2D on Nearlab dataset..... 78
Table 4-2 Learning parameters for Cnet1D on Nearlab dataset..... 79
Table 4-3 Learning parameters for CnetComb on Nearlab dataset..... 80
Table 4-4 Learning parameters for RESnet on Nearlab dataset..... 81
Table 4-5 Comparison of proposed deep networks on Nearlab database subjects 83
Table 4-6 Comprison of best models of classical machine learning and Cnet1D on Nearlab database 83
Table 4-7 Hyper-parameters of Cnet2D for Ninapro DB2 dataset 84
Table 4-8 Hyper-parameters of network Cnet1D for Ninapro DB2 dataset 84
Table 4-9 Hyper-parameters of network CnetComb for Ninapro DB2 dataset 84
Table 4-10 Hyper-parameters of network RESnet for Ninapro DB2 dataset 84
Table 4-11 Learning parameters for deep networks on Ninapro DB2 dataset..... 85
Table 4-12 Classification accuracy (%) of all Ninapro DB@ subjects..... 85
Table 4-13 Comparison of deep networks over Ninapro DB2 subjects 86
Table 4-14 Comprison of best models of classical machine learning and Cnet2D on Ninapro DB2 database 86
Table 4-15 Comparing Cnet1D and LDA on modified Nearlab dataset 88
Table 4-16 Prediction time (ms) comparison over all deep networks 89
Table 4-17 Learning parameter of Freeze & fine-tune network for Nearlab database 89
Table 4-18 Hyper-parameters of source network of Parallel method 90
Table 4-19 Hyper-parameters of subject network of Parallel method..... 90

Table 4-20 Learning parameters of source network from Parallel method 91
Table 4-21 Learning parameters of target network from Parallel method..... 92
Table 4-22 Comparison of Freeze & fine-tune and Parallel networks 93
Table 4-23 Time performance comparison of TL methods and Cnet1D..... 93
Table 6-1 Classification accuracy (%) comparison with related work over Ninapro DB2 exeB 100