PHD

Thesis

On

Deep Learning and sEMG Signals based Human Hand Gesture Recognition

by

WEIJIE KE

(*Ph.D. Student Id. 201777615)*

Under the guidance of

Dr. Lykourgos Petropoulakis

(*First Supervisor*)

&

Prof. John J. Soraghan

(*Second Supervisor*)

Neuromorphic Lab for AI and Deep Learning Systems Centre for Signal and Image Processing (CeSIP) Department of Electronic and Electrical Engineering University of Strathclyde, Glasgow G1 1XQ, U.K.

This thesis is submitted for the award of the degree of

Doctor of Philosophy 2022

Declaration

This thesis is the result of the author's original research. It has been composed by the author and has not been previously submitted for examination which has led to the award of a degree.

Signed: WEIJIE KE
Date: 01/03/2022

Acknowledgements

I would like to express my sincere appreciation to my first supervisor, Dr. Lykourgos Petropoulakis, who gave me space and room to conduct my research and gain experience, kept encouraging me and providing me with guidance which brought benefits not only for my research work, but also for my life. Without his persistent help, the goal of this novel deep learning research would not have been realized.

I would also like to thank my second supervisor Prof. John J. Soraghan, as he was always friendly and approachable whenever I need assistance.

Additionally, I really want to give my special thanks to Dr Gaetano Di Caterina, who stood by my side whenever I needed him during the whole journey of my Ph.D. Seeing his smiling face always brought a vast boost on my moral.

Further, I owe my sincere appreciation to Prof StephanWeiss, who has been my annual reviewer twice, for his insight and knowledge that supported my research.

Last but not least, I owe my sincere appreciation to my colleagues, Paul Kirkland, Yannan Xing, Ming Gong, Amlan Basu, Keerati Kaewrak for the numerous ideas exchange, inspiration and help when I encountered difficulties. I also want to thank all other colleagues who shared RC351 laboratory over the last couple of years with their direct or indirect help in some way or another. It has been my pleasure to have you in my PHD journey.

Finally, a big thanks to my parents for supporting me to realize my dream. Also, my sincere appreciation to the University of Strathclyde, a wonderful place that teaches useful learning and provides a world learning research environment.

Abstract

Hand prosthesis have helped amputees to partially live normal lives as healthy people for ages since invented. However, due to the technical limitations such as the essential time required for signal sending and receiving, the state-of-the-art hand prosthesis still cannot fully restore real hand functions. As Artificial Intelligence (AI) technologies, especially deep learning and artificial neural networks, nowadays show an impressive performance when building smart machines, it is possible to use them to bring improvements to conventional hand prosthesis. This advanced AI prosthesis can learn real hand functions through self-learning and eventually, fully achieve all real hand functions. Moreover, with the help of current biologically inspired neural network models and spiking neural networks (SNN), the power consumption and reaction delays of a prosthesis can be further minimized.

A novel approach which attempts to address the long-existing problems such as frequent misclassification faced by current hand prosthesis is presented in this thesis. Through converting the raw surface electromyograph (sEMG) signals from amputees with different hand amputation levels and able-bodied people into heatmaps, with an applied properly designed convolution neural network which extracts and learns the features contained within the heatmaps, the mis-trigger disadvantage rapidly decreases. The experimental results, from 8 hand gestures so far, indicate that this novel approach is effective. Moreover, the relationship between the number of sensors, used to record the sEMG signals, and the recognition accuracy is also examined and presented in this thesis. According to the experimental results, the optimal or required number of sensors when recording sEMG signals can be minimized for different hand gestures without classification accuracy degradation.

This thesis also contains another novelty: a spiking sEMG signal maps-based hand gesture classification algorithm. The algorithm employs a spiking neural network as the classifier, as well as a common heatmap technique which efficiently converts the raw sEMG signals into common heatmaps that can be used by the spiking neural network. The classification results obtained by two different sEMG datasets denote high robustness of the novel algorithm. Moreover, further evaluation of the experimental

results denotes that both the mis-trigger issue as well as power consumption are reduced when applying this novel algorithm.

Additionally, to further utilize the potential of sEMG signals, another novel algorithm which aims at frequency domain features is presented in this thesis. This algorithm successfully extracts the frequency domain features by applying singular system analysis (SSA) to the raw sEMG signals and converts them into frequency density maps (FDM). With the help of a proposed SNN, the algorithm is proved to be efficient for different hand gestures. In addition, further evaluation of this algorithm indicates that it demonstrates a significant reduction in computational costs, training time, power consumption whilst, at the same time, results in lower classification errors/mis-triggers when compared to other state of the art hand gesture recognition methodologies.

Table of Contents

List of acronyms

AI	**artificial intelligence**
CV	**computer vision**
EMG	**electromyography**
EEG	**electroencephalography**
RNN	**recurrent neural network**
CNN	**convolutional neural network**
SNN	**spiking neural network**
CSNN	**convolutional spiking neural network**
SSA	**singular spectrum analysis**
FDM	**frequency density map**
sEMG	**surface electromyography**
SNR	**signal to noise ratio**
MUAP	**motor unit action potential**
WT	**Wavelet transform**
FFT	**Fast Fourier transform**
STFT	**short term Fourier transform**
WVD	**Wigner-Ville distribution**
AR	**autoregressive**
ARMA	**autoregressive moving average**
ARIMA	**integrated moving average**
MU	**motor unit**
MUNE	**motor unit number estimation**
IPIS	**inter-pulse intervals**
LSI	**large-scale integration**
ANN	**artificial neural network**
MLP	**multiple layer perception**
ReLu	**rectified linear unit**
KNN	**k-nearest neighbour**
NB	**naive bayes**

SVM	**support vector machine**
WFMM	**weighted fuzzy minimum-maximum**
BPTT	**back-propagation through time**
BIRNN	**bidirectional recurrent neural network**
LSTM	**long short-term memory**
LIF	**leaky integrate-and-fire**
SRM	**spike response model**
PSP	**post synaptic potential**
STDP	**spike time dependent plasticity**
RESUME	**remote supervised learning**
SPAN	**spike pattern association neuron**
LTP	**long-term potentiation**
LTD	**long-term depression**
DVS	**dynamic visual sensing**
IACNN	**intersected heatmap based CNN**
RCNN	**recurrent convolutional neural network**
BPNN	**back propagation artificial neural network**
SVD	**singular value decomposition**

List of Figures

List of tables

Chapter 1
Introduction

1.1 Preface

Through the past few decades, the Artificial Intelligence (AI) technologies flourished rapidly because of the achievements in computer science fields such as big data. Machine learning, one of the most prominent AI areas nowadays, has been applied on several aspects of classification or regression tasks in both research and industry. Considered as one of the most advanced machine learning branches, deep learning [1] also plays an important role in all these fields.

Since artificial neural networks achieved substantial success in pattern recognition, such as convolutional neural networks for human face recognition [2], species recognition [3], tracking tasks [4], etc., researchers wish to apply them on other challenging areas. As one of the novel research topics, the AI based hand gesture recognition for lower arm prothesis has drawn more and more attention in the last few years.

Nowadays, the AI technology brings convenience and effectiveness to daily lives. It provides a bridge between the machine and users' demands through establishing a channel that allows machines to acquire intentions from human behaviour such as spoken commands.

For medical use, the AI based hand gesture recognition provides a distinctive possibility for amputees with different hand damage levels to improve their situation and ultimately resume a near normal life. According to a report [5], just in the USA there are 185,000 new amputations every year. The most common is partial hand amputation with loss of 1 or more fingers, about 61000, followed by loss of one arm, 25000. Massive funds and efforts have been invested to provide aid to these amputees every year while the main problem still exists, as the current prosthetics cannot fully restore the functions of a real hand. However, with the AI based hand gesture recognition, the prosthesis would be able to learn to perform the acquired gesture with any certain inputs if it has learned features for that gesture.

Moreover, the AI based hand gesture recognition has considerable potential to be applied in several real-world human computer interaction applications like robot control to accomplish tasks where it is dangerous or inconvenient for human presence. Various surgeries could, potentially, be performed by trained robotic arms where a doctor's presence may not be possible, or bombs can be defused remotely. Some attempts for such applications have already be proposed and evaluated by research works [6][7][8].

1.2 Research Motivation

As mentioned in the preface, with the increasing amount of lower arm amputees every year, developing a prosthesis which can fully restore human hand functions are always a task of importance and priority for researchers. The current lack of intelligence for conventional prostheses causes their aid for amputees to remain at a basic level and, in many cases, can cause substantial inconvenience in their lives. With a high performance, efficient AI based hand gesture recognition system, such situation could be addressed, thus providing a better life for many amputees.

According to previous research works, there are various attempts which are based on classical machine learning methods addressed to create such prostheses. However, none of them has achieved satisfactory results. The main issues are: First, the conventional machine learning models require extensive training data that is structured to meet the model's requirements [9]. Second, the machine learning model requires special training which consists of learning algorithms designed by humans for that specific training process to fulfil its purpose [10]. Third, the machine learning model cannot deal with real-time biases [11]. All these factors result in the failure of conventional machine learning based hand gesture recognition approaches and, effectively, delay AI based hand gesture recognition development.

With the development and improvement of AI technology, mostly deep learning, we have never been closer to achieving this task. Unlike other AI machine learning means, such as decision tree or random forest, deep learning requires less human involvement in training to achieve a good recognition accuracy as it does not need extensive reserves of pre-structured data and learns on an incremental basis [12]. For example, images for different large dogs' species might give different classification results when applying conventional machine learning methods. For deep learning though, the differences/errors

between the prediction results and real classes would be returned to adjust parameters for the next training session. In other words, deep learning can learn from its own errors and make necessary adjustments rather than using fixed error values [13]

The applications of deep learning in hand gesture recognition in medical field, or prosthesis, can be achieved with two ways. One approach is computer vision (CV) based. Some researchers have tried to implement hand gesture recognitions through conventional RGB or Grey-scale images which contain captured gestures [14]. The other approach is based on different signals that are collected when a gesture is performed [15]. These signals are usually Electromyography (EMG) signals [16] or Electroencephalography (EEG) signals [17], usually processed by recurrent neural networks (RNNs), which are best for dealing with sequence data.

For the CV based methods, the quality of the recognition accuracy is mainly affected by the captured gesture images due to the differences which exist in the size of human hands, for example it is hard for convolutional neural networks (CNNs) to recognize a baby's hand gesture when they have been trained use adult's hand gestures. In addition, the angle of the camera while acquiring the gesture image also has a significant effect on recognition accuracy. Once the angle changes, the captured features that identify a gesture may look very different to a CNN. Comparing with the image approach, the other methods which use signals such as EMG signals or EEG signals are more reliable. As measured directly from the motor units or muscles or brain signals when gestures are performed, the patterns/waveforms of the signals for different persons are roughly the same as same position muscles are used, despite magnitude (muscle strength) differences [17]. Moreover, due to the EMG signals are easily acquire and more accurate when reflecting muscle activities with the sensors attached directly on the target body parts, comparing to EEG signals, the focuses of this research focus on the EMG based hand gesture recognition area.

Nonetheless, the EMG based approach also has its own limitations which reduce the recognition accuracy and the applications. As the muscle strength for humans are different, it is difficult to find common features for a single gesture for various people when using the collected signals directly. Additionally, the acquired raw signals not only contain the required motor units or muscle activations, when performing a gesture, but they also contain noise which is generated by the environment or the device or even the

muscle itself or a combination of such effects. This noise further increases the difficulty to use signals to perform hand gesture recognition [18]. Thus, algorithms which can extract features from raw signals while removing the noise are required for improving current EMG based hand gesture recognition approaches.

Moreover, there is another issue which relates to the power consumption and computation speed for deep learning-based hand gesture approach in real time. When applying statistical deep learning methods such as CNNs and RNNs to real time applications, the required power and computation time increase with the scale of the number of the gestures and sensors. Such extra power and time consumption limitations makes the deep learning-based hand gesture recognition difficult to implement and unable to beat conventional prosthesis methods, such as body-powered prostheses like hooks, terminal devices, gripping devices and wrists units [19][20][21]. However, these limitations could be surpassed by the next generation of the ANN, the spiking neural networks (SNNs). Compared to CNNs and RNNs, SNNs require less levels of power consumption when operating by employing a simplified bioinspired neuron model as the basic operating unit and an event-based training algorithm which uses spikes as the information carrier [22]. In addition, because of the spike-based training algorithm, the SNNs have shown significant potential in terms of computational speed [23]. Furthermore, there is another advantage to applying an SNN based hand gesture recognition approach. Unlike previous ANNs, there already exist specialized hardware for SNNs such as IBM TrueNorth [24], Intel Loihi [25], etc. Such chips consist of billions transistors with only mW (milliwatt) level power consumption, requiring only 1/10000 of the power of conventional processing units [26].

1.3 Aims and Objectives

The aim of this research is to investigate the current deep learning methodologies which includes various types of neural networks and evaluate their performance on sEMG based hand gesture recognition tasks. As current deep learning methodologies have better recognition performance, reduced complexity solution and time and energy consumption advantages when compared to conventional machine learning methods.

The objectives of this research are listed as follows:

1. Examining the features contained inside the sEMG signals and designing appropriate methods to acquire these features.

2. Investigating the noise effects that reduce the performance of sEMG based hand gesture recognition systems.

3. Designing algorithms which can improve the sEMG based hand gesture recognition performance through minimizing noise effects.

4. Examining key parameters that affect the performance of a neural network and designing appropriate ANN models and training algorithms which meet the engineering application requirements such as good performance, energy efficient etc.

5. Building sEMG based hand gesture recognition systems by combining designed noise removing and feature extracting algorithms of the raw sEMG signals with the designed ANN models.

6. Evaluating sEMG hand gesture recognition systems performances and applying them on open-source datasets to obtain more experiment results.

1.4 Original Contributions

The research works presented in this thesis provide several contributions in the sEMG based hand gesture recognition area which include:

i. Design the intersected heatmap algorithm which converts the time domain features of each channel of raw sEMG signals to grey scale maps where each pixel represents one channel. The use of the intersected heatmap algorithm not only reduces the required data dimensions by discovering the optimal number of sensors which produce the highest recognition accuracy, but also improves the performance of the hand gesture recognition system by minimizing the noise effects. (Chapter 4)

ii. Design a novel VGG based CNN structure. The comparison results to previous research works indicate that this novel network structure can achieve better recognition results with fewer parameters. (Chapter 4)

iii. Discover the common active sensors for each applied hand gesture and design the common heatmap algorithm which minimizes the network training and dataset processing time, while maintaining a desired hand gesture recognition performance. This is achieved by further reducing the required numbers of sensors. (Chapter 5)

iv. Design a novel convolutional spiking neural network (CSNN) structure and evaluate its performance with common heatmaps. The CSNN structure takes advantage of a convolution operation and a spike train to achieve reduced energy requirements and training time consumption with an expected testing accuracy, which could improve the performance of real time hand gesture recognition systems (hand prosthesis). (Chapter 5)

v. Apply the singular spectrum analysis algorithm (SSA) which efficiently reduces the noise contained inside raw sEMG signals and the novel frequency density map algorithm (FDM), that enables the use of the frequency domain features of the sEMG signals to the hand gesture recognition system. (Chapter 6)

Overall, the research presented in this thesis introduces a hand gesture recognition system that leverages deep learning and sEMG signals. It combines signal processing techniques to reduce the necessary input sensors and deep learning methodologies that employ spikes. The aim is to lower power consumption and reduce reaction time while maintaining a high level of recognition accuracy. In essence, this work offers a potential avenue for enhancing myoelectric prostheses to address their existing limitations.

1.5 Thesis Outlines

This thesis consists of 7 chapters. Chapter 1 includes the introduction, research motivation, research aims and objectives of the thesis.

Chapter 2 presents the basic concepts of the sEMG technique including detection, processing and related hand gesture research. In addition, the concept of the human hand prosthetics is also given in this chapter through explaining the reasons why they were first invented and how they developed through history. The disadvantages of some state-of-the-art hand prosthetics are also listed in this chapter.

Chapter 3 introduces the basic concepts of the artificial neural networks and their involvements. From the simple multiple layer perceptions to the SNNs, their structures and components such as neurons, layers etc. are all listed. Moreover, previous research works which relate to human gesture recognition are also mentioned in this chapter.

Chapter 4 presents the first novel contribution of this thesis, a novel processing algorithm which converts the raw sEMG signals to heatmaps and a novel VGG based CNN structure, which provide a potential solution to the misclassification problem for current hand gesture recognition tasks. The heatmaps algorithm reduces the noise contained inside the raw signals and the proposed CNN structure improves the hand gesture recognition performance.

Chapter 5 presents a method to minimize the required time consumption of the hand gesture recognition by reducing the needed sensor numbers when classifying. Additionally, a novel CSNN structure is also included in this chapter, which uses spikes rather than conventional RGB or Grey-scale images while training the network. The development technique enables hand prosthetics to have a faster response in real time.

Chapter 6 introduces a novel FDM and CSNN based hand gesture recognition approach, which applies SSA technique and heatmaps algorithm for frequency domain features of the sEMG signals. This approach was evaluated and proved to be effective with two sEMG datasets which include 16 different hand gestures.

Chapter 7 contains the conclusion of the contributions in this research, together with some relevant future works.

1.6 Author's Publications

Xing, Y., Ke, W., Di Caterina, G., & Soraghan, J. (2019). Noise reduction using neural lateral inhibition for speech enhancement. *In International Journal of Machine Learning and Computing*. (Accepted for publication)

Ke, W., Xing, Y., Di Caterina, G., Petropoulakis, L., & Soraghan, J. (2020, February). Intersected EMG heatmaps and deep learning-based gesture recognition. *In Proceedings of the 2020 12th International Conference on Machine Learning and Computing (pp. 73-78)*.

Ke, W., Xing, Y., Di Caterina, G., Petropoulakis, L., & Soraghan, J. Deep Convolutional Spiking Neural Network Based Hand Gesture Recognition. *In International Joint Conference on Neural Networks*. (Accepted for publication)

Ke, W., Di Caterina, G., Petropoulakis, L., & Soraghan, J. Singular Spectrum Analysis and Frequency Density Map based Hand Gesture Recognition on Deep Convolutional Spiking Neural Networks (Submitted for publication)

Chapter 2

Review of EMG Signals and Hand Prosthetics

2.1 Introduction

In this chapter, a review of EMG signals in terms of their history, technology evolution and how their significance in hand gesture recognition tasks is discussed. The development of hand prosthetics and its combination with EMG signals is also included in this chapter. Section 2.2 includes the history introduction and generation methodology, together with some previous research of detection and processing of EMG signals. Several detailed classification methods such as artificial intelligences technologies (machine learning and deep learning) of hand gestures based on EMG signals from former research through decades are also included. Section 2.3 reveals the history of hand prosthetics and section 2.4 talks about its development after combining EMG signals. Section 2.5 provides a detailed introduction for the applied dataset in this research work.

2.2 Electromyography (EMG) Signal

The sEMG is short form of surface Electromyography. It is a non-invasive procedure involving the detection, recording and interpretation of the electrical activity of groups of muscles at rest (i.e., static) and during activity (i.e., dynamic) [27]. The measured electric signals can reflect the active level of the according motor units directly. Also, the recorded signals usually are shown in the form of graphs which is easy for users to make analyses. Figure 2.1 below indicates an example of an Electromyography. The

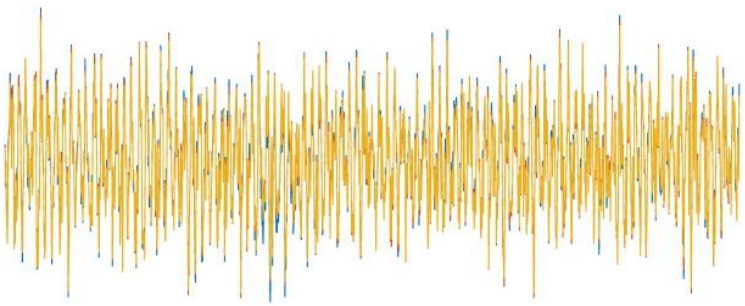

Figure 2.1: Example for electromyography signal.

pulses in figure 2.1 are composed of sEMG signal values recorded by electromyography sensors at different time points. Since sEMG sensors record discrete electrical potentials of muscles, a relatively high sampling frequency is normally required to reconstruct more accurate EMG signals.

2.2.1 History of EMG Signals

The earliest mention of surface electromyography (sEMG) dates back to 1666, when Francesco Redi documented the electric-generating muscle of an electric ray fish[28]. In 1773, Walsh observed that eel fish muscle tissue could produce electrical sparks [28]. Further research led to A. Galvani's 1792 publication "De Viribus Electricitatis in Motu Musculari Commentarius", revealing that electricity could trigger muscle contractions [29]. Dubois-Raymond made a significant breakthrough in 1849 by demonstrating that electrical activity during voluntary muscle contractions could be recorded [30]. Based on his theory, Marey successfully recorded this electrical activity in 1890 and named the method electromyography [31]. Gasser and Erlanger showed electrical signals from muscles through oscilloscopes. Because of the randomness of the EMG signals, only rough information was obtained from their observation [28]. However, the potential industrial and medical ability showed by them of the EMG signals attracted massive attention of various researchers and led to a steady development for the following two decades. Moreover, from 1930s to 1950s, improved electrodes were applied more widely for muscle research, which further improved the collected EMG signals' quality [28]. The first successful medical application of EMG signals was achieved by Hardyck and his colleagues for the treatment of specific disorders in 1966 [31]. In the early 1980s, a clinical method for scanning a variety of muscles through EMG sensing devices were proposed by Cram and Steger [31]. Through their discovery, the development of EMG based medical application began to flourish.

In 1980s, the integration techniques in electrode manufacturing became sufficiently advanced to allow batch production of small and light weight instrumentation and amplifiers [28]. The combination of these two-technique breakthrough further expanded EMG application scope. A better understanding for EMG signal recording technology was gained through past years' research and massive suitable hardwires are commercially available nowadays.

In recent years, the EMG technique has evolved to sEMG (surface electromyography) technique. By just placing electrodes in the surface of the muscles, one can easily record detailed muscle activities which previously could only be achieved through inserting intramuscular needles. This improvement not only makes the medical diagnosis and treatment in clinical protocols more convenient, but also is adopted by many prosthesis production companies.

Furthermore, the majority of methodological advancements have originated within specific regions, leading to a diverse array of methodologies across various user groups. This prompted the emergence of the European concerted action SENIAM (surface EMG for non-invasive muscle assessment) [32]. A crucial aim of SENIAM was not merely to foster collaboration among European entities, but also to develop guidelines pertaining to sensors, their placement, signal processing methodologies, and modeling strategies.

2.2.2 EMG Signal Detection

Normally, sEMG signal recording requires delving into the complex processes that occur in the human body, from the firing of motor neurons to the measurement of electrical signals.

Human muscle control involves motor neurons within the central nervous system [33]. These neurons transmit electrical impulses when the brain commands a muscle to contract or relax. Muscle fibers are organized into motor units for precise control [33]. When a motor neuron receives a signal for muscle contraction, it generates an action potential, which travels to the neuromuscular junction. There, it triggers the release of acetylcholine, leading to muscle cell membrane depolarization. This signal spreads through T-tubules to the sarcoplasmic reticulum, releasing calcium ions [34]. These ions facilitate the interaction between myosin and actin, causing muscle contraction—the sliding filament theory. As muscles contract and relax, they generate electrical activity, and this electrical activity is the actual sEMG signal [32].

There are two issues of concern which remain unsolved, but which have major influence on the signal fidelity. The first one is SNR (signal-to-noise-ratio), which is the ratio of the energy in EMG signals to the energy in noise signals. Normally, the noise in detection is often generated from electrical signals that are not part of the desired EMG

signal. The other one is the distortion of the signals, meaning that the relative component in the EMG signal should not be altered when recording.

When detecting EMG signals nowadays, two types of improved electrodes are commonly applied: invasive electrode and non-invasive electrodes. The non-invasive electrodes are located on the surface skin while invasive electrodes are normally in the form of needles and needed to be inserted into the muscle. Furthermore, both electrodes are mounted directly on the skin when recording. As the recorded signal is a composite of all the muscle fiber action potentials and these action potentials often occur at random intervals, the collected EMG signal might have either positive or negative voltage. For a single muscle fiber action potential, its obtained information is commonly limited so the combination of action potential from all the muscle fibers which named as the motor unit action potential (MUAP) is often used instead in today's detection process. Equation 2.1 indicates a simple model [35] of the EMG signal:

$$X(n) = \sum_{r=0}^{N-1} H(r)E(n-r) + W(n) \tag{2.1}$$

where $X(n)$ is the EMG signal. $E(n)$ represents the firing impulse. $H(r)$, indicates the MUAP. $W(n)$ represents zero mean additive white Gaussian noise and N is the number of motor unit firings.

There are various methods for EMG signal detection. The earliest used one is visual inspection by trained observers, meaning asking pre-trained professionals to check motor-related events visually [35]. The most commonly and intuitively computer-based method of time-locating the onset of muscle contraction activity is called "single threshold method", which compares the EMG signal with a fixed threshold. This methodology is based on comparison of the rectified raw signals and an amplitude threshold which depends on the mean power of the background noise [36]. Though this method is more advanced compared to the visual inspection, however, due to the measured signal values depending strongly on the choice of threshold, it sometimes could not provide accurate results. Moreover, the single threshold method often relies on criteria that are too heuristic and does not allow users to set the detection and false alarm

probabilities separately. To overcome the drawbacks of the single threshold method, Bornato et al. [37] introduced "double threshold detection method" in 1998. Providing higher detection probability, this method allows users to adopt the link between alarm and detection probability with a higher degree of freedom compared with signal threshold method. Furthermore, users can tune a detector according to different optimal criteria to make its performance adaptable for characteristics of specific signals and applications. In 2004, a new method was proposed based on the original double threshold detection method, by Lanyi and Adler [38], considering that Bornato's method is complex, computationally expensive and requiring a whitening of the signal. Lanyi's method is more sensitive, stable and efficient with decreased computation cost. One of

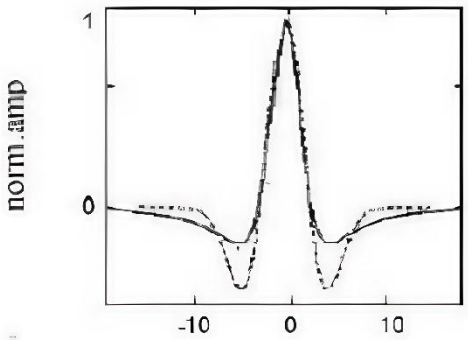

Figure 2.2: Mexican hat wavelet and typical unipolar MUAP shape [37].

the specific disadvantages to the method of Bornato is that to achieve the maximum detection probability, one of the thresholds must be set equal to "1". This implies that the double threshold detector turned into a single threshold detector. Another solved problem in Lanyi's method is that the signal whitening step is no more required. As this process occupies a lot of computation time and reduces detection probability of the signal, which will relate in missing part of activation interval, Lanyi's method marks a significant step forward in EMG detection.

2.2.3 EMG Signal Processing

After acquiring raw EMG signals, various signal processing methods are required as the contained useful information are hiding behind things like noises.

One of the commonly applied signal processing method is the Wavelet transform (WT) as wavelets can analyse both the time and frequency domain features [39]. The WT has been proved as an efficient mathematical tool for local analysis of nonstationary and fast transient signals. One main characteristic of WT is that it can be implemented by means

of a discrete time filter bank [40]. The work in [41] argues that if the wavelet analysis is chosen as to match the shape of the MUAP, the resulting WT yields the best possible energy localization in the time-scale plane. In 1997, Laterz and Olmo [42] discovered that WT is an alternative to other time frequency representations. It contains the advantage of being linear, producing a multiresolution representation and not being affected by crossterms when dealing with multicomponent signals. As EMG signals can be considered as the sum of scaled delayed versions of a single prototype, these advantages are particularly relevant. Laterz and Olmo applied the well-known Mexican hat wavelet in their approach, which is indeed the second-order derivative of a Gaussian distribution. The form of Mexican hat wavelet and typical unipolar MUAP shape are shown in figure 2.2.

The disadvantage of the Mexican hat wavelet is obviously shown in this figure: Mexican hat wavelet is not perfectly matched to the MUAP shape. Thus, other research kept aiming at creating a perfect matching. In 1998, Ismail and Asfour [43] came with a

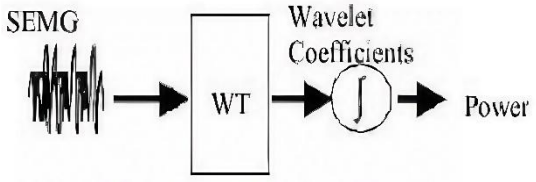

Figure 2.3: Processing flowchart for Kumar's approach [40].

theory that the optimal method to determine the frequency spectrum of EMG are the fast and short-term Fourier transforms (FFT and STFT). But they also pointed out the major drawback of this approach is the signal stationary assumption applied, while the EMG signals are nonstationary. Pattichis and Pattichis [44] found out that WT can also be used to analyse signals at different resolution levels. The continuous time signal expression for signal samples x_0, x_1, x_2, …, based on their theory is shown by equation 2.2,

$$f^0(t) = \sum_k x_k \emptyset(t-k) \tag{2.2}$$

where $f^0(t)$ is the continuous time signal, $\emptyset(t-k)$is called a scaling function. This indicates that the signal samples are weighted averages of the continuous signal. In 2003, Kumar et al. [45] produced an approach that decomposed the signal into several

multiresolution components according to a basis function called “wavelet function”. The processing flowchart is shown in figure 2.3. This method can be seen as a mathematical microscope that provides a tool to detect and characterize a short time component within a nonstationary signal. Utilizing Wavelet Transform (WT) in conjunction with various Wavelet Functions (WFs), they devised a method to decompose EMG signals. The output obtained from the power transform domain is then computed and employed as the pivotal criterion in selecting the optimal WF, which yields the most distinct contrast between different EMG scenarios. In accordance with Kumar's theory, this approach facilitates a straightforward determination of muscle fatigue, also known as muscle failure status.

Another commonly applied processing method is called Time-frequency approach. This kind of method focuses on gaining quantitative information from EMG signals in time domain. Cohen class transformation, Wigner-Ville distribution (WVD), and Choi-William’s distribution are famous time-frequency approaches used for EMG signal processing. Cohen class transformation proposed by Cohen in 1995 is good at analysing EMG signals recorded during dynamic contractions, which can be modelled as realizations of nonstationary stochastic process [46]. The Wigner-Ville distribution (WVD) [46] is a time-frequency that can display the frequency as a function of time, thus utilizing all available information contained in the EMG signal. Ricamato et al. proposed that WVD could be used to display the frequency ranges of the motor unit in 1992 [47]. Equation 2.3 presents the Wigner-Ville distribution formula.

$$W_x(t,\omega) = \int_{-\infty}^{\infty} x\left(t+\frac{\pi}{2}\right) x^*\left(t-\frac{\pi}{2}\right) e^{-j\omega\tau} d\tau \tag{2.3}$$

where $x(t)$ and $x^*(t)$ represents the signal and its complex conjugate. As WVD implementation needs a discrete form in digital computer, the fast Fourier transform is often applied, which produces a discrete-time or discrete-frequency representation. The Choi-William’s distribution theory [48] was proposed in 1993. Compared with previous methods, the major improvement of the Choi-William’s distribution is that it substantially reduces the interface for when detecting EMG signals.

The last commonly applied EMG processing method is Autoregressive model. This model enables input delayed intramuscular EMG to be formed in an AR model formation [28]. The autoregressive (AR) time series model was invented as surface electrodes will pick up EMG activity from all the active muscles in its vicinity, with only minimal crosstalk from adjacent muscles. In 1975, Graupe and Cline [49] first introduced the autoregressive moving average (ARMA) model to represent EMG signals. Based on their results, the EMG signals can be considered stationary over sufficient short time intervals. In 1980, Sherif replaced the model with an AR, integrated moving average (ARIMA) representation [50]. He also characterized the nonstationary nature of the EMG signal during different phase of muscle activity [50]. In 1986, Bernatoes et al. [51] employed a static nonlinear element with time-varying autoregressive moving average model. In 1989, Moser and Graupe [52] created a nonstationary identifier of time-varying AR parameters. However, the computation cost of ARIMA model is high and the determination of the model order is difficult and complex sometimes.

2.2.4 EMG Signal-Based Gesture Classification Techniques

Once obtained the processed EMG signals, various classification methodologies are used to create real world applications with them. The classification of the EMG data into their constituent motor unit action potential is normally a difficult task because of MUAPs waveform variability, jitter to single fiber potentials and MUPAs superposition [53]. In this section, three mainly used classification tools are introduced. They are based on a) Artificial Intelligence; b) Motor Unit Number Estimation methodology; c) hardware model.

Often, the common feature for classifying intramuscular EMG signal means the Euclidean distance is used between the MUAP waveforms [16]. The main features of the EMG signal are the number of active motor unit (MUs), the innervations time statistics and the MUAP waveforms. Based on Welling and Moschytz research results [54], the key points which dominating the classification results are MUAP waveforms and number of active MUs. Normally, the classification results can be influenced by noise such as background noise, offsets noise and white noise. According to Boualem and Peter's research, the time frequency representation of WVD provided high-resolution signal characterization in time-frequency space and good noise rejection performance [55].

The first popular classification method for EMG signal is based on Artificial Intelligence. In 1995, Christodoulou and Pattichis [56] proposed their three steps' ANN classification procedure: In the beginning, the unsupervised learning was applied based on the one-dimensional self-organizing feature map and competitive learning. Then, to improve pre-gained the classification results, learning vector quantization was introduced. Finally, the classification results were produced. Their research outputs drawn great attention because of the ability to adopt and create complex classification boundaries performed by their ANN. Figure 2.4 shows their classification strategy.

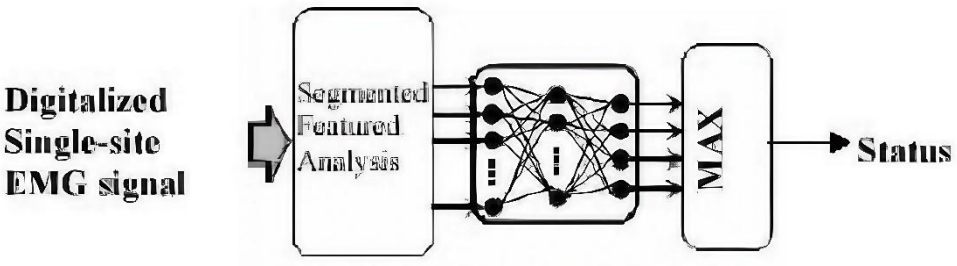

Figure 2.4: Classification strategy for ANN approach [51].

In 2000, Chan et al. published their work, a DRNN (deep recurrent neural network) which is more dedicated to EMG classification tasks [57]. Their output shows the designed DRNN successfully identifying the complex mapping between full wave rectified EMG signals and upper limb trajectory. Besides the neural network, Cheron et al. [58] proved that the fuzzy logic system can also deal with the classification problems

Table 2.1: Comparation results for fuzzy logic system and neural network.

Method	Accuracy Rate
Coefficients of AR ([58])	99%
Neural Networks ([58])	84%
Fuzzy System ([58])	85%

and might be superior compared with neural networks as the more consistent and non-over-training classification results the fuzzy logic system provided. The results comparation between fuzzy logic system and neural network are listed in table 2.1.

The second classification method is known as Motor Unit Number Estimation (MUNE). MUNE aims at evaluating the motor axons connected to a muscle. However, all MUNE techniques are based on assumptions which required to be fulfilled to produce valid estimates [59]. In 1971, McComas [60] proposed a neurophysiological methodology to estimate the number of motor units in a muscle. In his approach, a maximal bioelectric response of the muscle was recorded through EMG following a supramaximal electrical stimulation of the muscle's nerve. Then, the maximal EMG response was divided by the estimate average value of the single motor unit response. The result was an estimate number of the single motor unit responses that compose the maximal EMG response. In 1998, Zhengquan Xu and Shaojun Xiao [61] proposed an approach for estimating the mean and standard deviation of inter-pulse intervals (IPIs) of individual MUAP units. The firing parameters were estimated by a weighted matching between the observed IPI probability density function and the modelled function. The weighted function applied was to approximate the validity of IPI data as all valid information provided are utilized as far as possible. Their method was proved to provide reliable estimations even if the MUAP trains are extracted with significant errors. For this reason, the methodology is very good when estimating the firing statistics of surface EMG where the individual MUAP trains are difficult to be accurately identified. To address the problem that the basic shape of MUAP can be represented by a very small number of waveforms or wavelet functions, Ping and Rymer [62] provided methods to estimate the number of MUAPs present in standard surface EMG records through wavelet based matching techniques. Although the increase of the correctly estimated maximum MUAP number was small, their model still showed an advantage on correctly estimating the number of MUAPs in EMG signals at higher force level. In 2005, Major and Jones [63] applied four MUNE techniques: Incremental Simulation, Revised Incremental, Multiple Point Stimulation and Spike-Triggered Averaging on Ping and Rymer's model. The applied techniques allow a detailed testing of methodological assumptions in different MUNE techniques which will lead to a more accurate and reliable method of performing MUNE. According to their results, the estimated number of functional motor unites in a muscle can be represented as in equation 2.4.

$$N = \frac{(\max CMAP)}{(mean\ SMUP)} \tag{2.4}$$

where CMAP (Simulations of motor unit number estimation techniques) indicates the compound muscle action potential and SMUP indicates mean single motor unit response.

The third often used classification method which is also the earliest invention is the hardware model. The microprocessor system for myoelectric signal classification was proposed by Graupe et al. [64]. The device was built on an 8080 Intel Corporation microprocessor which is an 8-bit parallel central processing unit. A 40-pin dual in-line ceramic package which has a 2 μs instruction time was placed on a single Large-Scale Integration (LSI) chip with N-channel silicon gates, followed by a 4K-bytes semiconductor memory. Furthermore, a hardware multiplier unit was interfaced with the microprocessor in order to increase the processing speed. In 1999, Yen et al. [65] designed instrumentation amplifier, gain control stage and filters into a chip for processing EMG signals in adequate amplitude and limited bandwidth since they contain characteristics of low voltage amplitude and low frequency common mode noise. The chip has achieved goals of low cost, minimizing layout area and low power consumption. In 2001, a two-step incremental evaluation of a prosthetic hand controller that based on a floating-point CPU or a neural network was proposed by Torresen [66]. Through applying gate level EHW, such implementation can make it more feasible to be installed inside a prosthetic arm. The structure of the controller is shown in figure 2.5.

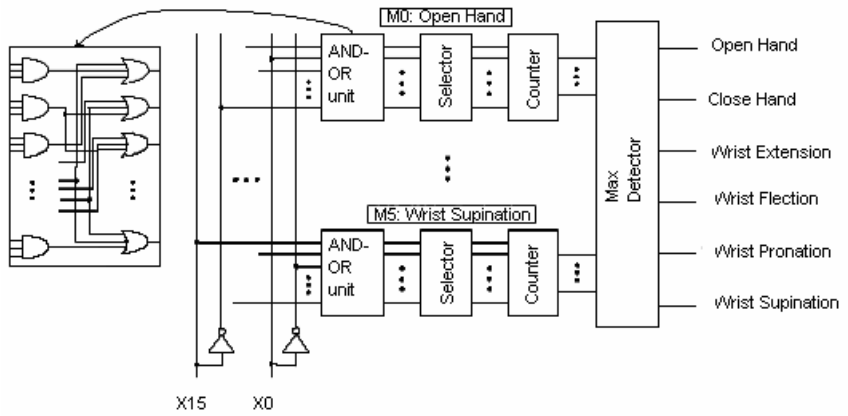

Figure 2.5: Structure of Yen's controller [60].

Each of the six prosthetic motions in figure 2.5 contains one sub-system. In each sub-system, the binary inputs are represented by a number of different units starting by the AND-OR units. Then, a layer of AND gates is formed, followed by a layer of OR gates. The input number for each gate is the same and can be chosen as two, three or four. The OR gates' outputs are routed to the selector which selects which of these outputs are to be counted by the succeeding counter. In the end, the max detector outputs which counter corresponding to one specific motion having the largest value. Each output from the Max Detector is connected to the corresponding motor and the input can be considered correctly classified only if the counter having the largest value corresponds to the correct hand motion.

The constraint imposed by the limited length of chromosome strings poses a significant challenge in the evolution of hardware systems. Typically, intricate systems necessitate the utilization of extended strings. Consequently, as the string length escalates, genetic algorithms generate vast quantities of chromosome generations [28]. Nevertheless, the primary advantage of the hardware model is undeniable: Evolution proceeds in a bottom-up fashion across identical devices, rather than being confined to a single operation on the entire evolvable unit.

2.3 Hand Prosthetics

Looking through human history records, various amputations such as front arm amputations have caused a huge disability globally. As the human body does not have a regeneration ability, the only solution is to replace the lost part with a prosthesis. All amputations are important, especially hand amputations because they prevent several preform functions for the amputees who unfortunately have it. Still, prosthetic hardware remains limited by weight, durability, and power supply. Human muscles, in contrast, are highly adaptive and precise, making their functions difficult to replicate. Modern state-of-the-art prostheses address this gap through myoelectric control, neural interfaces, and sensory feedback, offering more natural movement and improved quality of life.

Human upper limb is a vital part of the body, and any partial or complete loss of the upper limb can cause significant effort on a person's ability. For any human upper limb, there are three sections: hand, forearm, and arm. Each of these three sections

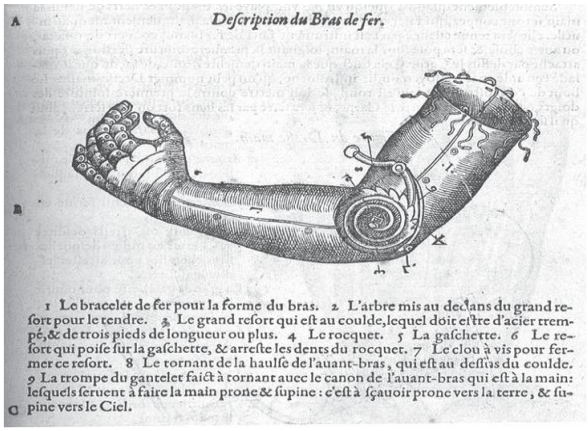

Figure 2.6: Götz von Berlichingen's iron hand [61].

contains massive nervous system, musculoskeletal systems, and its surroundings [67]. To perform various daily activities, all these three sections are essential.

Taking forearm as an example. The forearm muscles are biologically divided into two parts: the anterior flexor compartment and the posterior extensor compartment [67]. To achieve any hand movements such as moving the elbow, wrist and digits of the hand, all muscles of these two compartments are required. Additionally, according to these muscles' functions, they can also fall into two categories: intrinsic and extrinsic muscles [67]. The intrinsic muscles function to move the forearm by pronating and supinating the radius and ulna while the extrinsic muscles flex and extend the digits of the hand [67].

Until now, the only way for hand prosthetics to restore amputees' hand functions is imitating upper limb muscles' functions.

One of the earliest records of a prosthetic hand was described in 77AD by Roman scholar Pliny the elder in his encyclopaedia [68]. After losing a hand in the Second Punic War (218–201 BC), Marcus Sergius, a Roman general, received a prosthesis that enabled him to return successfully to battle.

One of the most famous earlier hand prothesis example in the west was the iron hand of a German knight Götz von Berlichingen [69]. After losing his hand during the Siege of Landshut, an artisan forged an iron hand with flexible and extendable digits at the metacarpophalangeal, proximal interphalangeal, distal interphalangeal joints and thumb interphalangeal joint for him. As presented in figure 2.6. The fashioned hand allowed Götz to re-hold weapons and return to the war. However, this

prothesis was considered as an armour instead of an assistance device that time and due to its own weight, thick leather straps were required when equipping.

Another famous record for early hand prothesis was from Italian historian and physician Paolo Giovio [69]. In his record, the Turkish pirate Horuk Barbarossa lost his right arm in the battle of Bugia when against Spain in 1577. With an iron replacement, he was able to re-join the fight and finish it. About 100 years later, the Duke Christian of Brunswick also received a same replacement for his left arm which lost in the battle of Fleury [69]. Yet, all prothesis from these examples were forged in the purpose of war. Despite being heavy and needing control by the abled hand of the amputee, the early hand prothesis still restored part of the real hand ability.

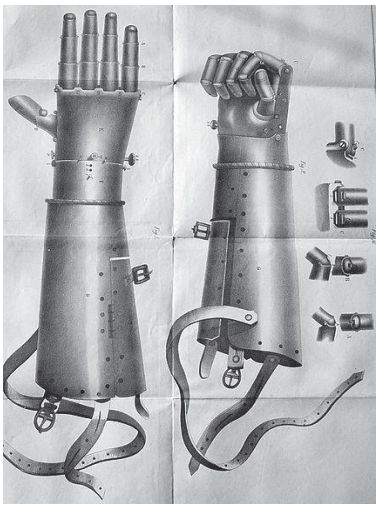

Figure 2.7: Le Petit Lorrain [63].

It was not until 1600 when Italian surgeon Giovanni Tommaso Minadoi [69] described a hand prothesis that could help an amputee to remove a hat, open a wallet or even write using a quill, that the noncombative hand prothesis began to attract people's attention. In the 16th century, the French military surgeon Ambroise Paré [70] drew the first detailed design of a spring-loaded prosthetic hand which later named "Le Petit Lorrain" shown in figure 2.7.

In conclusion, these hand prosthetics were not simple tools to hold objects, but they were specially designed and crafted in the human-hand form. However, the expensive forging price meant that only the wealthy could afford such kind of customized devices.

The concept of an automatic hand came from by a German dentist Peter Baliff in 1818 [71]. The device he invented was able to elicit motion for hand amputees by girdling the intact muscles of the trunk and shoulder in a terminal device attached to the amputation stump. For the first time the amputees could control their prosthesis with fluid body motions instead of a distinct foreign object. In 1860, Comte de Beaufort [72] from France improved Peter's design by passing through a loop to the contralateral axilla and missing limb with a strap which buttoned to the trousers. With this improvement, an amputee could open and close a double spring hook or flex and extend the thumb on a prosthesis with fused fingers through manipulating the strap tension. In 1916, German surgeon Ferdinand Sauerbruch [73] produced his upper arm muscle movements transmission-based digits-controlled prosthetic. According to his film captures, the amputees were able to drink from a teacup and even to remove a match from a box to light a cigarette through manipulating his invention. Unfortunately, like previous situations, due to the high cost of the production, only a few rich individuals could afford the device.

Both World War I and World War II caused casualties in numbers previously unimagined. After World War I, the Unite States created an amputee rehabilitation programs to help majority upper limb amputees to regain some working abilities by disturbing prosthetics with sockets and a universal terminal device allowed the attachment of various work tools [74]. Other countries such as Canada also realized the need to provide support to amputees and this led to the creation of the Amputations Association of the Great War [70]. All those country efforts brought rapid development for hand prosthetics during that time. By the end of World War II, the huge demand for upper limb prosthetics led to the creation of various artificial limb research and development organizations like the US Committee on Prosthetics Research and Development and the Canadian Association of Prosthetics and Orthotics [70]. In 1948, the Bowden cable body-powered prothesis was invented to replace straps with a sleek and sturdy cable [75]. With Bowden cable body-powered prothesis, an amputee could operate a terminal device in an impressive range of motion, speed and force by changing the tension in a cable via preserved shoulder and body movements. One of the main improvements shown by the new device was that it permitted users to complete tasks more efficiently by allowing them to use both hands simultaneously, rather than requiring an integral hand for controlling. Another

improvement was that the users were able to predict and adjust the position of the prosthetic through sensing cable tension without any visual feedback. Thus, with the durable, portable and relatively affordable characteristics, Bowden's prothesis became quite popular and widespread since invented. In fact, today's body-powered prostheses are essentially adaptations of Bowden's design. Despite the disadvantages such as that prolonged wearing can be uncomfortable, complicated motor tasks are limited e.g. using the equipment to grab an egg, the body-powered prostheses are widely used nowadays [76].

2.4 EMG Based Hand Prosthetics

When it comes to 20th century, scientists and engineers started to focus on combing traditional hand prosthetics with electromyography techniques due to the achievements on both fields in past decades. In 1919, a German book named "Ersatzglieder und Arbeitshilfen (Limb Substitutes and Work Aids)" was published [77]. The book contained conceptual designs for the first externally powered prostheses which began to be known by the whole world. Though, these revolutionary designs were too complex to be achieved with contemporary technology, they can still be considered as the first attempts of hybrid prosthetics.

About 29 years later a physics student Reinhold Reiter from Munich University created the first real hybrid prosthesis [78]. This kind of devices which was later to be known as myoelectric prosthesis, amplified surface electromyography signal potentials to power motorized parts. However, Reiter's published work was not

Figure 2.8 Bebionic Hand 8E7 [83].

widely appreciated, and this ground-breaking device did not receive any commercial or clinical acceptance.

The first successful industry myoelectric prosthesis was invented by Russian scientist Alexander Kobrinski in 1960 [79]. The applied transistors not only reduced the size but also allowed portability of the equipment through belt and wires. The device was even improved through the use of a skin-coloured rubber cosmetic glove as a cover. Although this device was quite popular in US and Canada, it also had numerous problems: heavy weight, slow movement, weak pinch force, easily broken wire connections and, most important, electrical interference compromised reliability.

By the 1980s, myoelectric prostheses were extensively used in rehabilitation centres worldwide. Nowadays, they have become the most common choice for hand amputees [80] due to their advantages which includes a reduction of harnessing, access to effortless strength and multiple grip patterns, more natural hand movements and more intuitive control abilities [81]. The breakthroughs in material fields allowed lighter and more ergonomic designs for the myoelectric prostheses [82]. Moreover, the driving power has evolved from compressed gas into rechargeable nickel-cadmium batteries. The bebionic hand, designed by Ottobock, is one of the state of art myoelectric prostheses [83]. As shown in figure 2.8, 14 selectable grip patterns and hand positions enable users to perform a huge number of daily activities. Also, companies such as Hosmer (part of Össur), Fillauer, TRS Prosthetics, College Park Industries etc. also produce myoelectric prostheses with excellent quality and durability [84][85].

Compared with earlier versions, modern myoelectric prostheses offer improvements in both comfort and aesthetics by eliminating external cables and incorporating life-like silicone overlays. The use of surface electromyography remains non-invasive and safe, with operational effort comparable to natural limbs [86]. Control muscles differ depending on the level of amputation; for instance, trans-radial amputees typically use preserved wrist flexors and extensors, while trans-humeral amputees also engage biceps and triceps [87] [88]. Current state-of-the-art prothesis enable multiple grip patterns, proportional finger control, and more natural appearance. Advances in lightweight materials, rechargeable batteries, and

microprocessor-based adaptive control further enhance usability, while ongoing research into sensory feedback aims to restore tactile perception.

Yet, several challenges remain unresolved in the myoelectric prosthesis field. Unlike body-powered devices, they rely on external power and require frequent recharging [89]. Training to isolate muscle signals is time-consuming, and complex movements often demand simultaneous finger, wrist, and elbow articulation. Response delays, electrode shifts, and skin conditions such as sweating can all disrupt sEMG signals [87]. The absence of sensory feedback forces constant visual monitoring, which is unnatural and error prone. Moreover, high cost remains a major barrier: in the 1990s, myoelectric prostheses were about six times more expensive than body-powered ones [70], and although material innovations have lowered prices, affordability is still limited [90][91][92]. Finally, each device must be customized for a single amputee, as sEMG signals vary widely between individuals.

2.5 Dataset Applied for Research

All experiment results demonstrated in this paper were based on two datasets. One is Strathclyde dataset, which includes forearm sEMG signals collected from 13 healthy people and 9 amputees. The other one is CapgMyo dataset, an open-source dataset with 23 able-bodied participants.

2.5.1 Strathclyde Dataset

The sEMG hand gesture dataset used in this chapter was collected by the DSP group members of the EEE department of Strathclyde University [93]. The whole dataset comprises of the summation of different MUAP with its amplitude which including some external noise due to the acquisition process. In total, 22 participants with 12 distinct hand gestures were included.

The raw sEMG datasets were first amplified and sampled with a frequency of 2048 HZ. Following was a denoising process that consisted of two parts. Most sEMG activity happens between 5 Hz and 450 Hz according to previous research [94]. The first part of the denoising was achieved by a hardware filter comprising a high pass filter with a 3Hz cut off frequency and a low-pass filter with cut off frequency at 900 Hz.

Table 2.2: Amputee Participant Information [93]

Participant ID	Age	Gender	Level of amputation	Time since amputation in years
B01	68	M	Mid TR	36
B02	64	M	Mid TR	31
B03	57	M	Kuechenberg	40
B04	75	M	Mid TR	74
B05	56	M	Long TR	22
B06	57	F	Short TR	Congenital
B07	37	M	Long TR	Congenital
B08	17	M	Long TR	Congenital
C01	44	F	5 Fingers	1
C02	41	M	5 Fingers	3
C03	45	M	5 Fingers	3
C04	45	M	5 Fingers	1
C06	43	M	5 Fingers	3

Among the participants, 9 of them are able bodied individuals and the rest are amputees with different levels of hand amputations. The amputee participants included 5 partial-hand (which had all five fingers amputated to some extent while the wrist joint was kept intact) and 8 trans-radial amputees (including three congenital cases) [93]. Detailed information about the level of amputation and time since amputation of the volunteers is shown in table 2.2.

Figure 2.9 indicates the schematic diagram of a signal acquisition device which consists of 2 separate bands. The left picture shows the upper band with the first 64 sensors placed on the top of the forearm and the right picture shows the lower band attached to the bottom of the forearm, with the remaining 64 sensors. Figure 2.10 shows an example of the recording process for 12 gestures.

The data collected from the participants was categorized as follows for analysis: (i) able-bodied participants, (ii) partial-hand amputees and (iii) trans-radial amputees [93].

The recording process starting with placing one high density electrode array consisting of 64 channels [95] over the flexor compartment muscles and another identical electrode array over the extensor compartment muscles [93]. Each electrodes inside these two arrays had a diameter of 2mm and was spaced at a distance of 8mm from one another [93]. The recording was performed in a floating monopolar configuration. Participants were seated comfortably and rested their forearms on a desk with their elbows flexed. They were requested to perform bilateral hand gestures when prompted by text displayed on a monitor placed directly in front of them. For each gesture, same gesture cue was

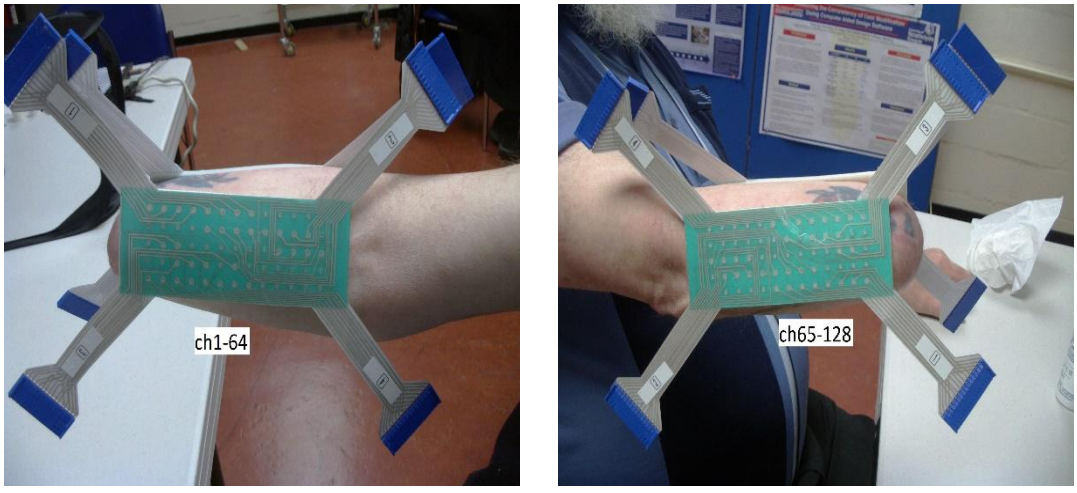

Figure 2.9: Structure of the collecting device.

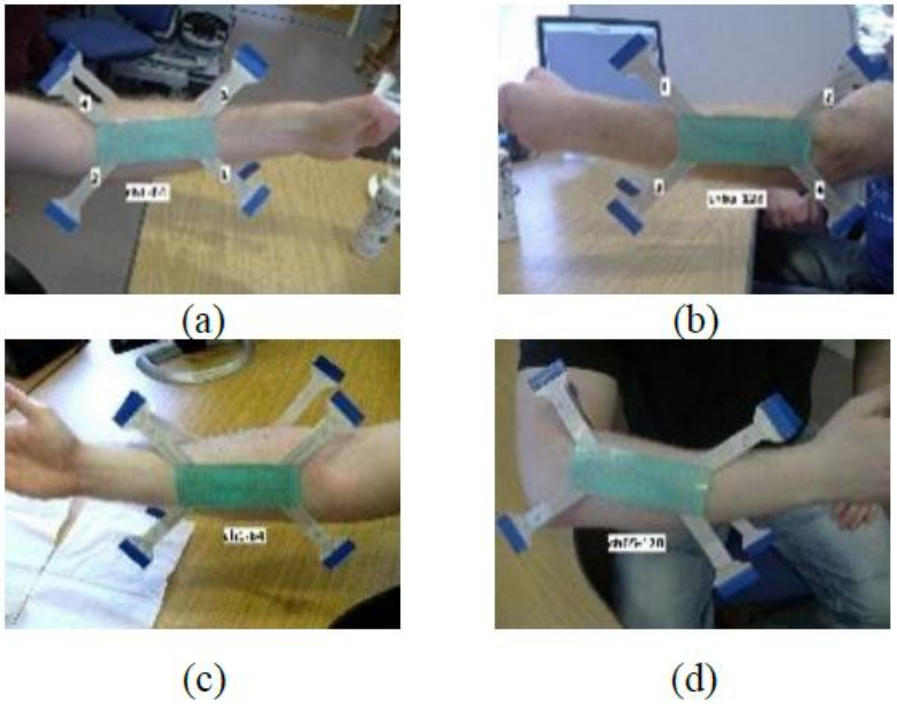

Figure 2.10: (a) and (b) show 1 to 128 sensors attached on amputee's forearm. (c) and (d) show 1 to 128 sensors attached on able-handed participant's forearm.

repeated 5 times [93]. When recording, participants were instructed to imagine the action being carried out and contract their muscles at their normal strength. In addition, for unilateral amputees, a mirror box was positioned over the amputated arm to hide that arm from the amputee's field of view to assist the amputee to visualize the required

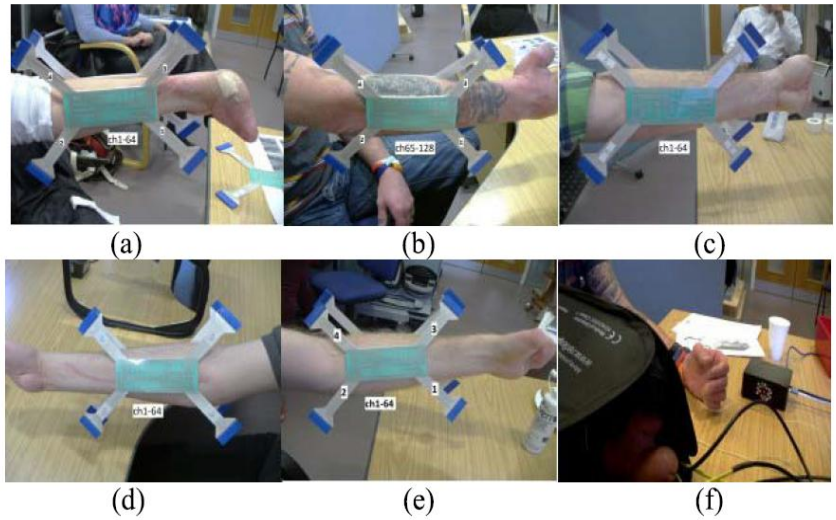

Figure 2.11: (a) and (b): Placement of the two 64-channel electrode arrays on the flexor compartment muscles and extensor compartment muscles, respectively. (a) to (e): Partial-hand amputation of participants C01, C02, C03, C04, and C06, respectively. (f): A mirror box positioned over the amputated arm.

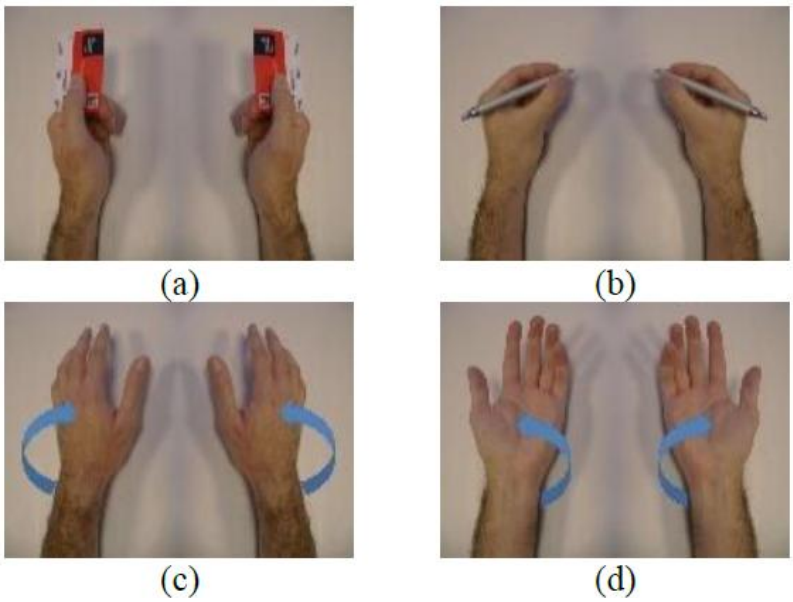

Figure 2.12: Classified gestures (a) lateral; (b) tripod closed; (c) palm down; (d) palm up.

gesture which happens on the side of his/her amputation [93]. The recording procedure is shown in figure 2.11.

Generally, the 12 distinct gestures performed by all the participants including the amputees. They were all performed from rest in an interval of 3s and maintained for 4-5 seconds, followed by a rest period of 3-4 seconds [93]. Additionally, the sensor locations remained the same for both amputees and able-bodied participants when recording for the same gesture (if the amputee does not have the muscles to perform the gesture, these sensors are abandoned and the rest sensors are placed at the same position as for the able-

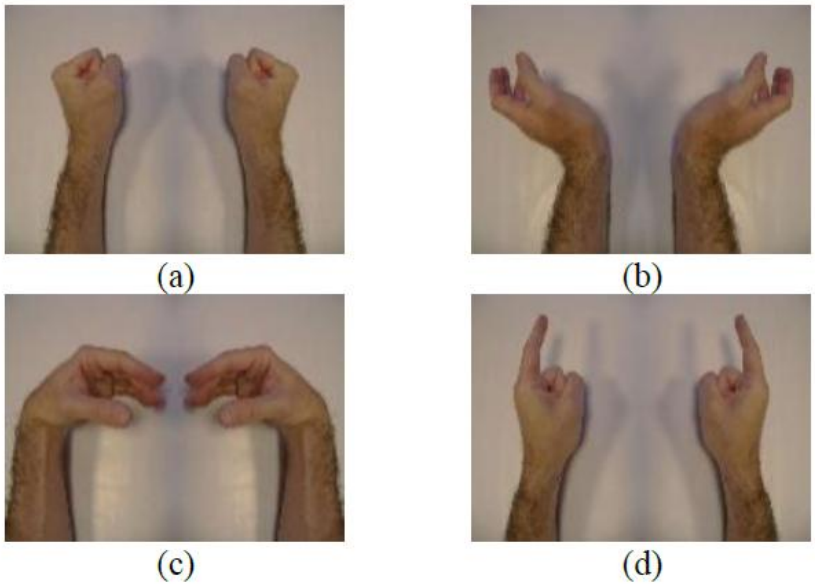

Figure 2.13: Classified gestures (a) close; (b) extension; (c) flexion; (d) point.

bodied participants). Though 12 gestures were recorded, as the strength of the sEMG signals of some gestures performed by amputees are too weak when analysing, we selected 8 gestures whose strength was strong for all participants for classification task. Figures 2.12 and 2.13 present all 8 applied gestures.

Table 2.3: Muscles for Each Gestures[96]

Gesture	Muscles
Close	Flexor Digitorum Superficialis; Flexor Digitorum Profundus; Flexor Carpi Ulnaris; Flexor Carpi Radialis
extension	Extensor Digitorum; Extensor Carpi Radialis Longus; Extensor Carpi Radialis Brevis; Extensor Carpi Ulnaris
flexion	Flexor Digitorum Superficialis; Flexor Digitorum Profundus; Flexor Carpi Ulnaris; Flexor Carpi Radialis
point	Extensor Digitorum; Extensor Carpi Radialis Longus; Extensor Carpi Radialis Brevis
lateral	Flexor Carpi Ulnaris; Flexor Carpi Radialis; Extensor Carpi Radialis Longus; Extensor Carpi Radialis Brevis
tripod closed	Flexor Digitorum Superficialis; Flexor Pollicis Longus; Flexor Digitorum Profundus; Flexor Carpi Radialis; Flexor Carpi Ulnaris
palm down	Extensor Digitorum; Extensor Carpi Radialis Longus; Extensor Carpi Radialis Brevis; Pronator Teres
palm up	Flexor Digitorum Superficialis; Flexor Digitorum Profundus; Flexor Carpi Ulnaris

Additionally, when recording data for the recognition of eight different gestures, nearly all the muscles in the forearm are engaged [96]. For instance, as illustrated in Figure 2.13, the "gesture close" involves the activation of the Flexor Carpi Ulnaris, Flexor Digitorum Superficialis, Flexor Digitorum Profundus, and Flexor Pollicis Longus muscles. On the other hand, the "palm-down" gesture not only engages the aforementioned four forearm muscles but also necessitates the involvement of the Pronator Teres muscle for forearm pronation. The specific muscles required for each gesture are detailed in Table 2.3.

2.5.2 CapgMyo Dataset

The Capgmyo dataset consists of 23 healthy, able-bodied participants ranging in age from 23 to 26 years. It was collected through a non-invasive wearable device. The device contains 8 acquisition modules. Each acquisition module includes a matrix-type (8x2) differential electrode array. On each side of the acquisition board, eight pairs of electrodes are situated. Each electrode within a pair measures 10mm × 10mm × 1mm, with a 20mm separation between the two acquisition electrodes. Additionally, to mitigate common-mode interference, a reference electrode sized at 10mm × 4mm × 1mm is positioned at the centre between the two acquisition electrodes [97]. The whole database consists of 3 sub-datasets (channel DB-a, channel DB-b and channel DB-c). Each gesture for DB-a was held for 3 to 10 seconds. Because readings for gestures were taken at different times, the electrodes cannot be placed at the exact same position each time so the collected signals have tiny differences [97]. The DB-c contains 12 basic hand

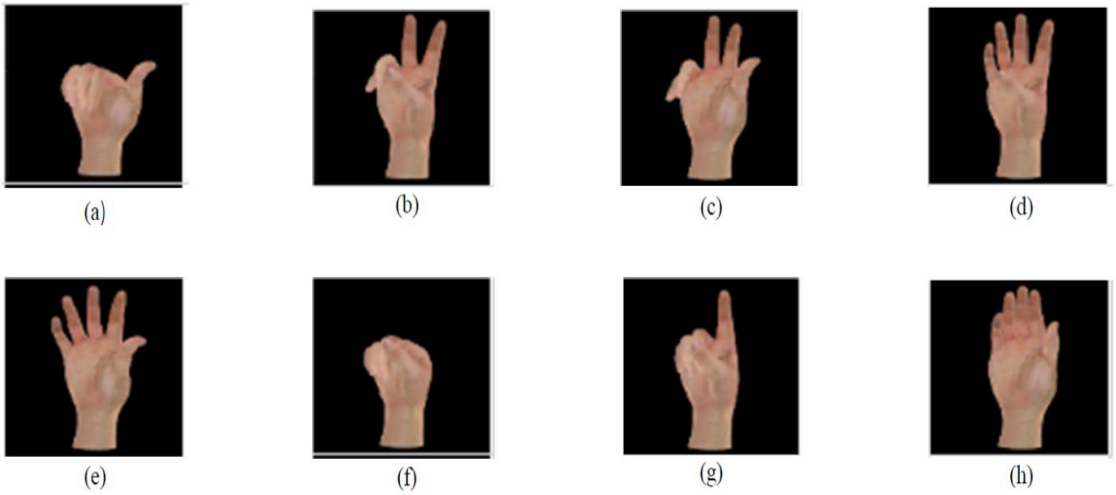

Figure 2.14: (a) Thumb up; (b) Extension of index and middle, flexion of the others; (c) Flexion of ring and little finger, extension of the others; (d) Thumb opposing; (e) Abduction of all fingers; (f) Fingers flexed together in fist; (g) Pointing index; (h) Adduction of extended fingers.

gestures' movements from 10 of the 23 participants. In total, 8 isometric and isotonic hand gestures were selected and applied from 18 of the 23 participants for the research from DB-a giving that the electrodes of DB-b cannot be guaranteed to place on the exact position each time and DB-c are just finger movements, so we just used DB-a as it contributes to the most hand gestures.

Because of characteristics such as the gesture performing time length and repeat performing numbers per hand gesture per participant, the denoising pre-processing of the CapgMyo dataset is different compared with our collected sEMG signals mentioned in section 2.5.1. Therefore, the format of the formed common heatmaps can be guaranteed unchanged (as it has the same sensor amount and similar attached positions as ours), we did the same denoising process to ensure as much compatibility as possible). Each selected sEMG signal from DB-a was passed through a band-pass filter with 20-380 HZ and sampled at 1000 HZ. Moreover, the obtained value of each signal was normalized to the [-1,1] range. The chosen 8 hand gestures performed are shown in figure 2.14.

2.5.3 Pre-processed sEMG Dataset with SSA

The research work presented in chapter 6 involves an improved technique, singular spectrum analysis (SSA) [98] for trends, smoothing and further extraction of periodic components for sEMG signals. As the recorded sEMG signals from both datasets (Strathclyde dataset and CapgMyo dataset) consist of the summation of values of different MUAPs plus some additional noise, after first applying traditional noise reduction methods to remove external noise (shown in section 2.5.1 and 2.5.2), the SSA is then applied.

As introduced in the beginning of this section, the SSA is a recent technique for time series analysis and forecasting which is able to decompose an original series into several independent components that are interpretable as varying trend, oscillations, or noise [99]. Since created, the SSA has been used on several applications such as effective feature extraction in hyperspectral imaging and achieved great success [100].

The conventional SSA can be applied in the following steps for a given 1D signal $x = [x_1, x_2, \cdots, x_n] \in R^n$.

- **Embedding**

Defining a window size $L \in Z$ where $L \in [1, n]$, a trajectory matrix of the original vector x can be formed as:

$$X = \begin{pmatrix} x_1 & x_2 & \cdots & x_K \\ x_2 & x_3 & \cdots & x_{K+1} \\ \vdots & \vdots & \ddots & \vdots \\ x_L & x_{L+1} & \cdots & x_N \end{pmatrix} = (C_1, C_2 \cdots, C_K) \tag{2.5}$$

where the columns of the matrix X are $C_k = [x_k, x_{k+1}, \cdots, x_{K+L-1}]^T \in R^L$, lagged vectors where $k \in [1, K]$ and $K = N - L + 1$. The matrix X has equal values in the antidiagonals. Additionally, based on X porosities, the SSA algorithm can be implemented symmetrically in two intervals [98].

- **Singular Value Decomposition (SVD)**

To perform SVD, another matrix S is required to be defined as $S = XX^T$, where the eigenvalues of S and their respective eigenvectors can be represented as $(\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_L)$ and $(U_1, U_2, \cdots, U_L)$. the SVD formular of the matrix X is shown below:

$$X = X_1 + X_2 + \cdots + X_d \tag{2.6}$$

where d is equal to the rank of the matrix X. According to equation 2.6, the matrix X consists of the addition of several matrices $X_i | i \in [1, d]$. These matrix are known as elementary matrices [98]. The related respective eigenvalue can then be defined as

$$X_i = \sqrt{\lambda_i} U_i V_i^T, V_i = \frac{X^T U_i}{\sqrt{\lambda_i}} \tag{2.7}$$

$$U = (U_1, U_2, \cdots, U_L) \in R^{L \times L}, V = (V_1, V_2, \cdots, V_L) \in R^{L \times L} \tag{2.8}$$

where U and V are known as matrix of empirical orthogonal functions and matrix of the principal components [98].

- **Grouping**

After the embedding and SVD, the total set of L individual components need to be grouped into M disjoint sets, which can be performed as $I_1, I_2, \cdots, I_M$ ($\sum|I_m| = l$ and $m \in [1, M]$). For a given disjoint set $I = [i_1, i_2, \cdots, i_p]$, the group matrix X_I can be defined as $X_I = X_{i1} + X_{i2} + \cdots + X_{ip}$. The trajectory matrix after the grouping can be presented as

$$X = X_{I1} + X_{I2} + \cdots + X_{IM} \tag{2.9}$$

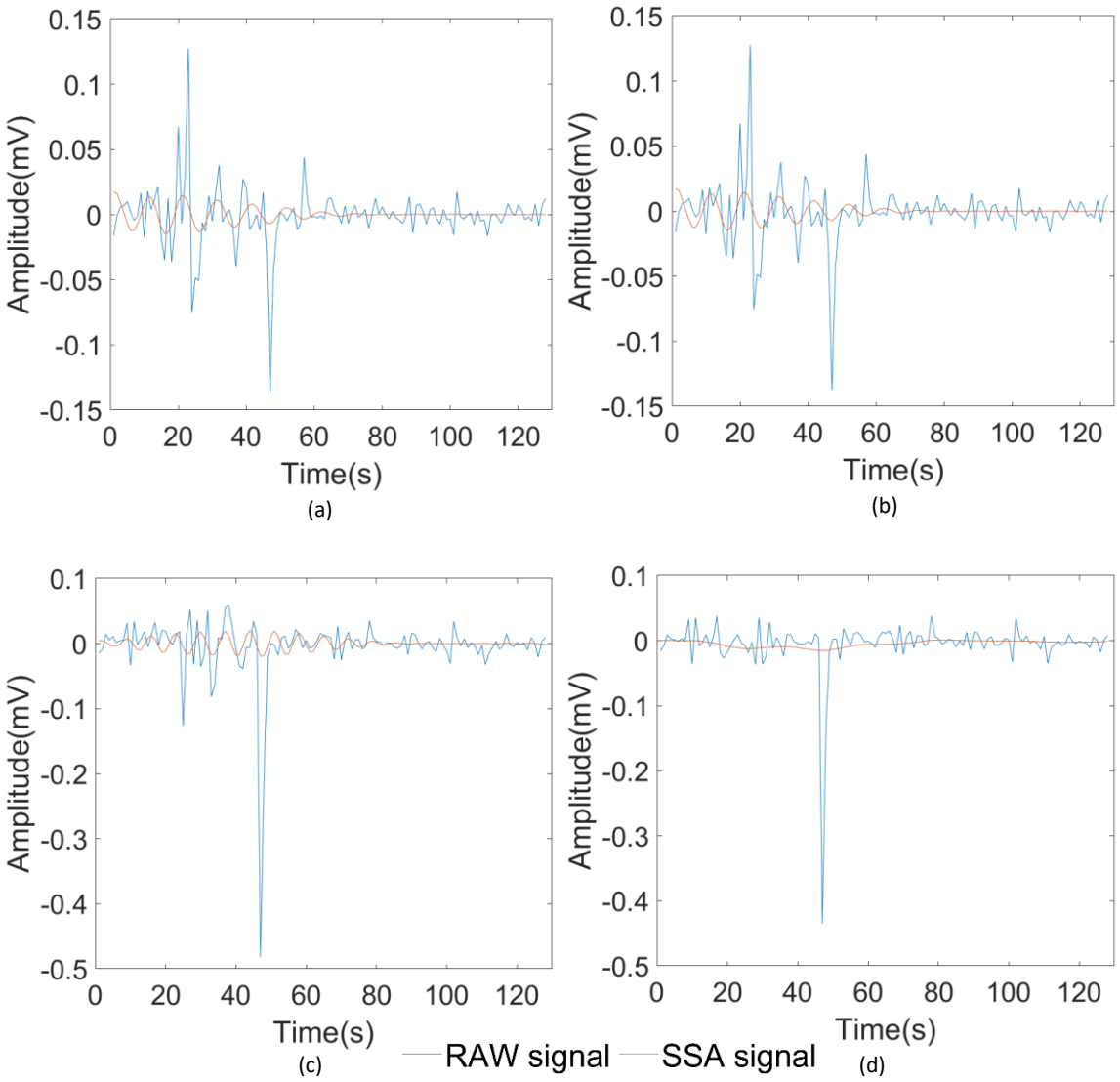

Figure 2.15: SSA Results for self-collected sEMG dataset; (a)channel 20, first record; (b) channel 20, second record; Gesture: close, participant 1. (c) channel 100, first record; (d) channel 80, first record; Gesture: extension, participant 1.

Additionally, the basic grouping happens under condition when $M = L$ and $p = 1$, where each set is made of just one component [98].

- **Diagonal Averaging**

Now comes the final step for the conventional SSA. As the obtained matrices shown in equation 2.9 are not necessary Hankel structure (not average in their antidiagonals), each one of them has to be Hankelized for the projection into 1D signals, due to the values in the antidiagonals of these matrices contributing to the same element in the derived 1D vector [101]. Through the means of the diagonal averaging shown in equation 2.10, the elements $a_{j,n-j+1}$ of the matrices X_{Im} can be projected to a given 1D signal $z_m = [z_{m1}, z_{m2}, \cdots, z_{mN}] \in R^N$.

$$z_{mn} = \begin{cases} \frac{1}{n}\sum_{j=1}^{n} a_{j,n-j+1}, 1 \le n \le L \\ \frac{1}{L}\sum_{j=1}^{L} a_{j,n-j+1}, L \le n \le K \\ \frac{1}{N-n+1}\sum_{j=n-K+1}^{L} a_{j,n-j+1}, K \le n \le N \end{cases} \tag{2.10}$$

when finishing grouping all the matrices, the original signal x can be formed as

$$X = z_1 + z_2 + \cdots + z_M = \sum_{m=1}^{M} z_m \tag{2.11}$$

According these four steps, the original signal can be reconstructed using specific components with noise or insignificant parts discarded.

The cleaned signals are then converted to frequency spike images through our frequency map technique. Figures 2.15 and 2.16 indicate signal examples for both Strathclyde sEMG dataset and CapgMyo dataset after applying SSA. As per the graphical representations, the prominent peaks present in the raw sEMG signal, which

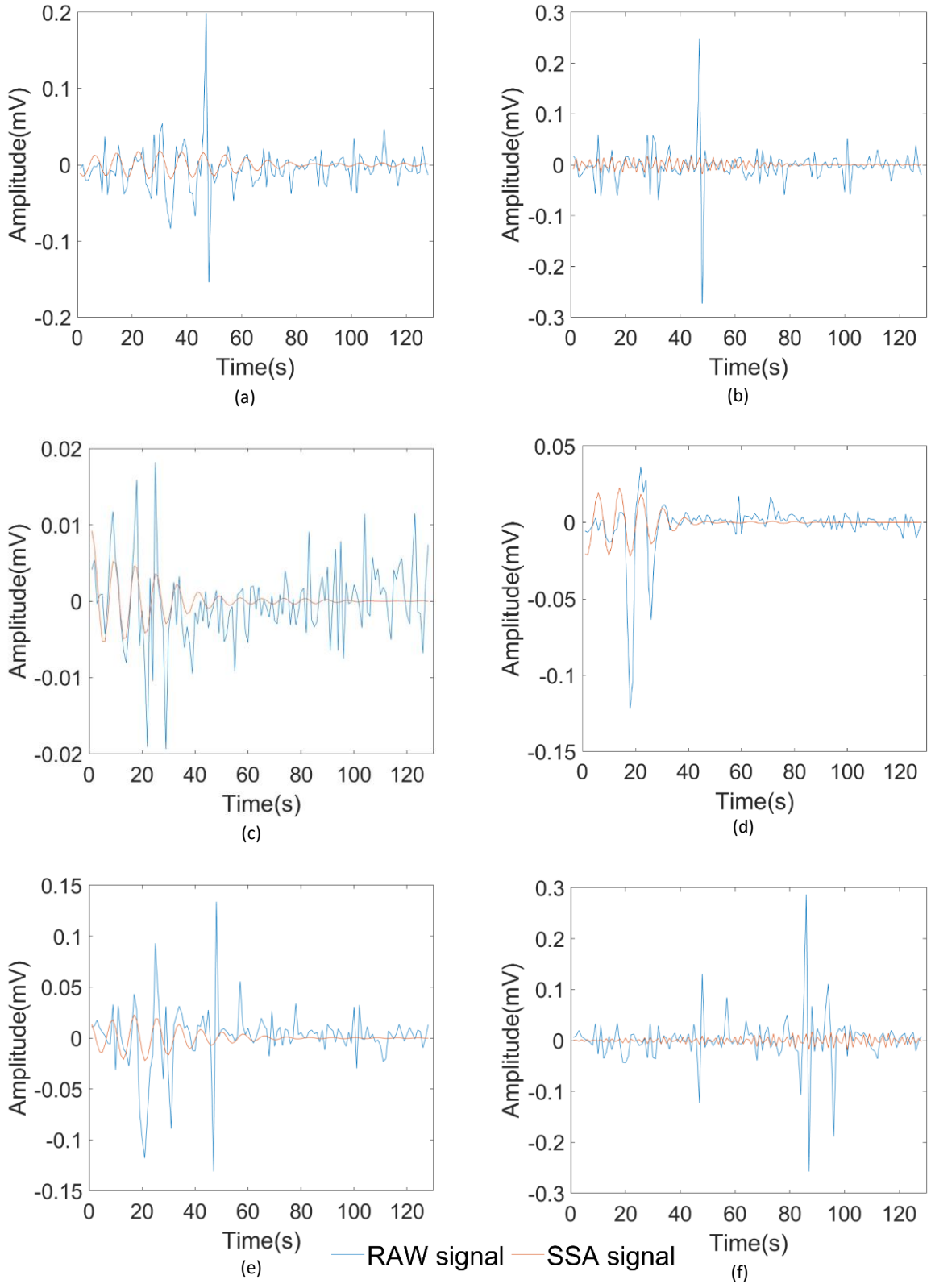

Figure 2.16: SSA Results for CapgMyo dataset; (a) channel 20; (b) channel 60; For participant 1, Gesture: thumb up, first record; (c) channel 40; (d) channel 60; For participant 1, Gesture: Extension of index and middle, flexion of the others, first record; (e) channel 20; (f) channel 60; For participant 2, Gesture: thumb up, first record.

are typically induced by noise or interference, have been effectively reduced. For

instance, in Figure 2.15a, two distinct sharp peaks (highlighted by the blue line) occurring around 43 seconds and 44 seconds are eliminated following the implementation of SSA (depicted by the red line).

2.6 Conclusion

This chapter has reviewed EMG signals developments and hand prosthesis techniques. The history of the EMG signals, their detection and processing methods, the applied sEMG datasets and pre-processing methods, together with the previous classification approaches were included. The history of the hand prosthesis and its development was also contained. The several issues mentioned in section 2.2, section 2.3 and section 2.4 form the basis for the chosen research inspirations and contributions in chapters 4-6 of this thesis. The combination of EMG signals and hand prosthesis i.e., EMG based hand prosthesis was reviewed due to it significantly contributing to the prosthetic improvement process. Additionally, it is also the fundamental theory of this thesis. Despite the presented defective classification results and unavoidable time delayed response, the use of artificial neural networks unlocks the ability of creating real automatic hand prosthetic which does not require human assistance and provides a key inspiration for the work which achieved through applying Strathclyde dataset, CapgMyo dataset and SSA algorithm in chapter 4, 5 and 6. Moreover, the early ANN works on EMG also inspired the application of neural networks in the research, which is introduced in chapter 3.

Chapter 3

Review of The DNN and SNN Techniques

3.1 Introduction

In this chapter, deep neural networks and certain gesture recognition applications that rely on both surface electromyography (sEMG) signals and deep neural networks (especially Spiking Neural Networks) are reviewed. Section 3.2 includes early stage deep artificial neural networks, known as the first generation of neural networks. Section 3.3 talks about the convolutional neural networks (CNN) and section 3.4 concludes the development of recurrent neural networks (RNN). The CNNs are commonly applied to deal with visual imaging tasks while the RNNs often appear in situations that require temporal sequence data analysis such as unsegmented and connected handwriting recognition or speech recognition. As the above two types of networks share similarities like operating through sending floating points and utilizing continuous activation functions such as Sigmoid or Tanh, they are termed as the second-generation neural networks.

The third generation of neural networks, spiking neural networks (SNN), is presented in this chapter.

The SNN is completely inspired by human brain neurons [102]. Thus, the differences between SNNs and other neural networks (e.g., CNNs and RNNs) exist in various aspects such as information carriers, basic processing units, sensing techniques and training methods. Moreover, SNN restores human brain neuron functions on computer science: communicating through broadcasting action potentials, also known as spikes, that have sample amplitude and are asynchronous.

From the science aspect, the representation of spikes in spatial and temporal domain allows SNNs to address event-based problems in a more “brain-like” way which unlocks its unique ability to better process temporal data. As for the engineering aspect, the binary spike events inside real neuron systems consumes only very little energy but can contain a high information content in the spike timing [22]. The same advantage in terms

of energy efficiency it is also inherited by SNNs and makes SNNs more suitable for neuromorphic chips applications than second-generation neural networks.

The detailed history of SNNs is introduced in section 3.5. Section 3.6 includes the current popular spiking neuron models. Section 3.7 discusses various encoding algorithms for SNNs and unique training methods and structures for SNNs are introduced in section 3.8. Section 3.9 summarizes the present state of the art in hand gesture implementations using Spiking Neural Networks (SNN), with several of them incorporating surface electromyography (sEMG) signals as well.

3.2 Artificial Neural Networks

The story of artificial neural networks begins in 1943. Neurophysiologist Warren McCulloch and his colleague, mathematician Walter Pitts [103] published a paper which introduced the operation function of the neurons. To support their theory and produce a better explanation for the readers, they modelled a simple neural network using electrical circuits. In 1949, Donald Hebb [104] discovered a fundamentally essential concept for human learning process, that neural pathways are strengthened each time when being used. The connection between two nerves will be enhanced if they are firing at the same time.

When it comes to 1950s, with the development of advanced computers, researchers were able to simulate neural networks which only existed in concept. Though failed, the attempt made by the IBM researcher Nathanial Rochester [105] still marks the birth of such hypothetical neural networks. In 1959, Bernard Widrow and Marcian Hoff [106] created models known as “ADALINE” and “MADALINE” [107] while their names represent the applied Multiple ADAptive LINear Elements [107]. The ADALINE was created for binary pattern recognition tasks such as predicting the next bit when reading streaming bits from a phone line. As for MADALINE, its main job is to deal with the real-world problems like eliminating echoes on phone lines through an adaptive filter. Though its design is quite crude for current neural network systems, this first neural network is still in commercial use. Only 3 years later, Widrow and Hoff [108] devised a learning process that initially scrutinizes the input value (0 or 1), prior to adjusting the weights according to the rule outlined in equation 3.1. This procedure facilitates the adjustment of weight values, enabling them to be distributed throughout the network or

among neighbouring perceptrons, in cases where a significant error is detected in one of the active perceptrons.

$$Weight\ Change = \ Pre - Weight\ line\ value \times \frac{Error}{Number\ of\ Inputs} \tag{3.1}$$

Despite the later success of the neural network, the traditional von Neumann architecture took over the computing scene for a quite long period while the neural network research was left far behind. A paper published at that time suggested there is no extension from single layered neural network to a multiple layered neural network [109]. Additionally, the learning function applied for researchers was fundamentally flawed as it was not differentiable across the entire line [110]. John von Neumann himself even suggested imitating neural functions through applying telegraph relays or vacuum tubes [110].

Fortunately, researchers who were interested in computer-based simulating neuron systems did not give up. In 1975, the first unsupervised multi-layered neural network eventually appeared [111]. In 1982, John Hopfield [112] presented a paper which provided an approach to create multiple layers by using bidirectional lines. A multi-layer hybrid network was created by Reilly and Cooper [113] in the same year. The problem-solving strategy for each individual layer inside that network is different. In 1986, the remained main issue for multiple layered neural networks was to extend the Widrow-Hoff rule to multiple layers [103]. A former member of Stanford's psychology department named David Rumelhart [114] proposed a method which distributes pattern recognition errors throughout the network. Now this method is known as back propagation. The results shown in his attempts indicated that the back propagation method requires possibly thousands of iterations to learn.

Nowadays, neural networks are involved in several applications and in many fields. Despite disadvantages such as slow learning processes, massive productivity has become possible by them [115]. However, due to the fact that the neurons inside a human brain actually work more like analog signals instead of digital signals [116], the future of neural networks lies in the development of hardware, in a faster, more efficient form.

3.2.1 Neurons

Neurons are the fundamental basements of the human brain. With the connections between each other, they endue us the ability of learning. However, their operation process remained a mystery until the birth of modern Neuroscience. A biological neuron consists of several different parts: Cell body, Nucleus, Endoplasmic reticulum, Mitochondrion, Axon hillock, Golgi apparatus, Dendrite, Dendritic branches, Axon, Telodendria and Synaptic terminals, as shown in figure 3.1.

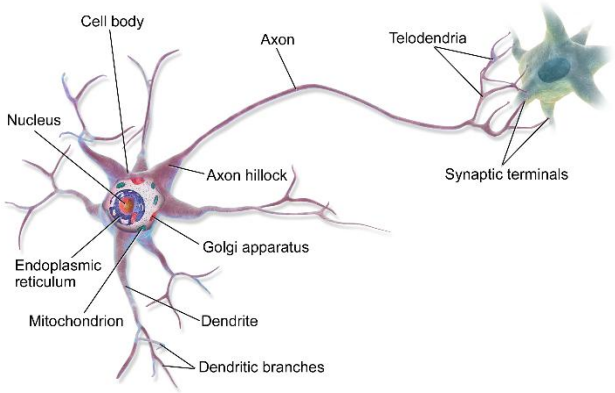

Figure 3.1: Biological neuron example [95].

The operative principles of neurons can be concluded as receiving signals and generating new signals. In other words, the neurons receive input data, perform some processing and then create a new output. Given stimuli in the brain, electrical signals will be transmitted from one end to the other i.e., from the dendrites to the axon terminals through the axon body [116]. Through this process, the electrical signals continue to be transmitted across the synapse from neuron to neuron. Thus, the information they carry would be recorded and analysed. In addition, the generated output only occurs when the input exceeds a certain threshold. Such function is known as an activation function, which will be introduced later.

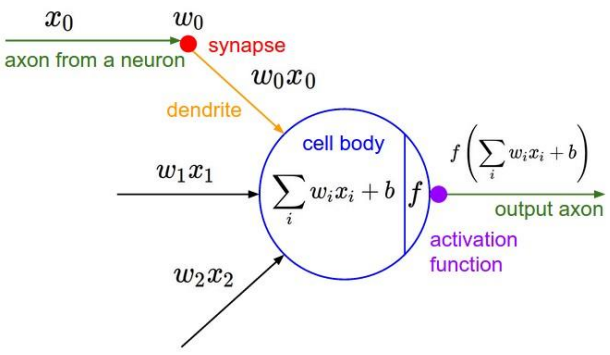

Figure 3.2: Common artificial neuron structure [96].

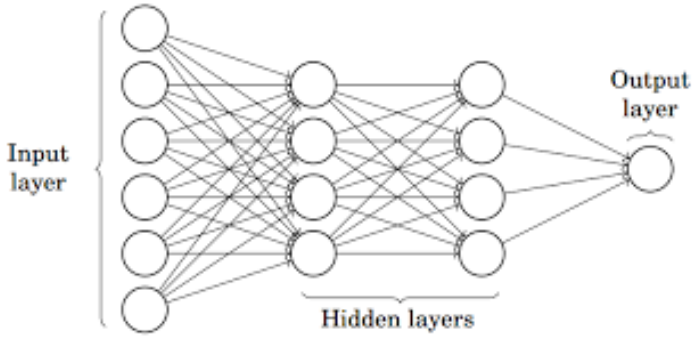

Figure 3.3: Multi-layer perception example [97].

As imitations from the biological neurons, the neurons inside neural network are exquisitely designed to achieve the same functionality. Figure 3.2 indicates the structure of common artificial neurons.

Compared with biological neurons, the inputs of the artificial neuron correspond to the dendrites, the transfer function, net input and activation function represent the cell body while the activation corresponds to the axon and synaptic terminal [117]. Like biological neurons, first, the input information such as images or digits are fed to the neuron. Next, the transfer function sums all the inputs together and generate values. The generated values are then compared with a specified threshold. If they reach the threshold, the activation function will generate an output which is a signal and send it to a post neuron or take it as the final output for the network depending on the artificial neural network (ANN) architecture. Otherwise, nothing will be generated or sent.

3.2.2 Multi-Layer Perception and Neural Networks

A multiple layer perception can be considered as the primary form of a deep, artificial neuron network. As shown by its name, it is composed of more than one perception. Usually, the first part of a multiple layer perception (MLP) is an input layer which receives the signals and transforms them. The last part normally is called output layer which makes a decision or prediction about the input. The middle part, that between those two layers, is often an arbitrary number of different hidden layers which are the true computational engine of the MLP. Additionally, MLPs with only one hidden layer are capable of approximating any situation though the results may not always be accurate. Figure 3.3 indicates the common structure of an MLP.

Most of the applications of the MLPs are located on supervised learning problems [118][119][120][121][122]. The MLPs are trained on a set of input-output pairs and learn to model the correlation or dependencies between input and output data [123]. The training process involves several steps such as adjusting the parameters, the weights or the biases of each layer to minimize the error. Back propagation is also applied in training process to adjust weights and biases relative to the error.

MLP is a typical example of the neural networks. These kind of networks normally have two motions, a constant back and forth [124]. In each forward path, the input layer transmits the signal through the hidden layers to the output layer. Then the output layer makes decision or prediction which will be measured against the pre-set ground truth labels. Then in each backward path, various weights and biases are backpropagated through the layers according to the chain rule of calculus and partial derivatives of the error function. Several gradient-based optimisation algorithms such as stochastic gradient descent can be applied in feedforward networks (feedforward means the connections between the neurons do not form any cycles) [125]. By using the differentiation, a gradient or a landscape of error will be obtained. In other words, the network parameters are adjusted to minimize the error. The gradient descent process is continued until the network error cannot be reduced further. This state is also called convergence.

3.2.3 Activation Functions

One of the key factors for all the training process for all kinds of neural networks is the activation function. Activation functions unlock the learning ability for the networks

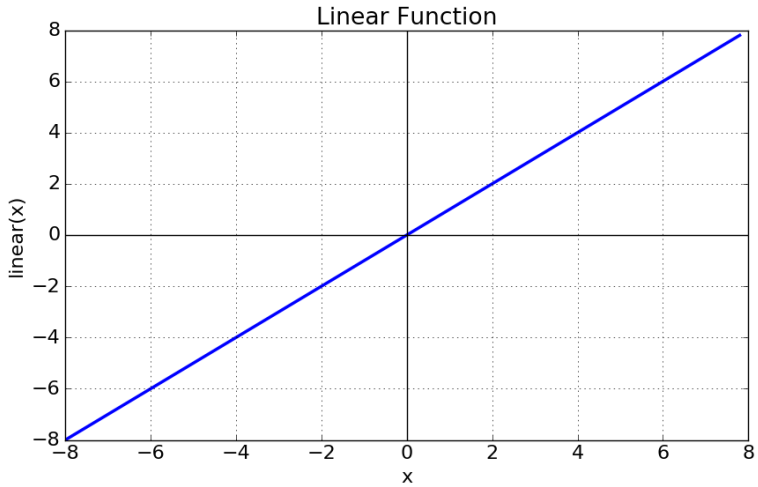

Figure 3.4: Linear activation function example [105].

and enable them to perform humanlike judgements. Activation functions are mathematical models that are attached to each neuron in a network and determine whether a neuron should be activated with regard to the relevance between the input and network prediction [126]. The output of the entire network is also normalized in a range dependent on the applied activation function type.

There are two types of activation functions. The first type is called linear or identity activation function. Figure 3.4 indicates the linear function form.

As shown, the output of the function will not be confined between any range, thus this kind of function does not help with the complexity or various parameters of usual data which is fed to the neural networks [127]. The second type of activation functions is named as non-linear activation function. This type of activation functions enables networks to generalise or adapt with variety of data and differentiate between the output [127], which makes them the most used activation functions in deep learning. Figure 3.5 indicates the shape of a non-linear activation function.

There are several commonly applied non-linear activation functions which are named according to their range or curves.

Sigmoid function is one of the most applied logistic/activation functions. All sigmoid functions have the property to map the entire number line into a small range such as between 0 and 1 or -1 and 1. So one main use of a sigmoid function is to convert a real value into a probability [127]. A sigmoid function is usually placed as the last layer of a

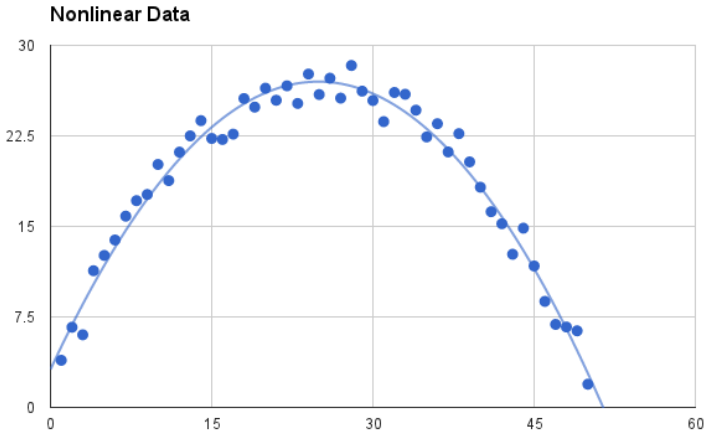

Figure 3.5: Non-linear activation function example [105].

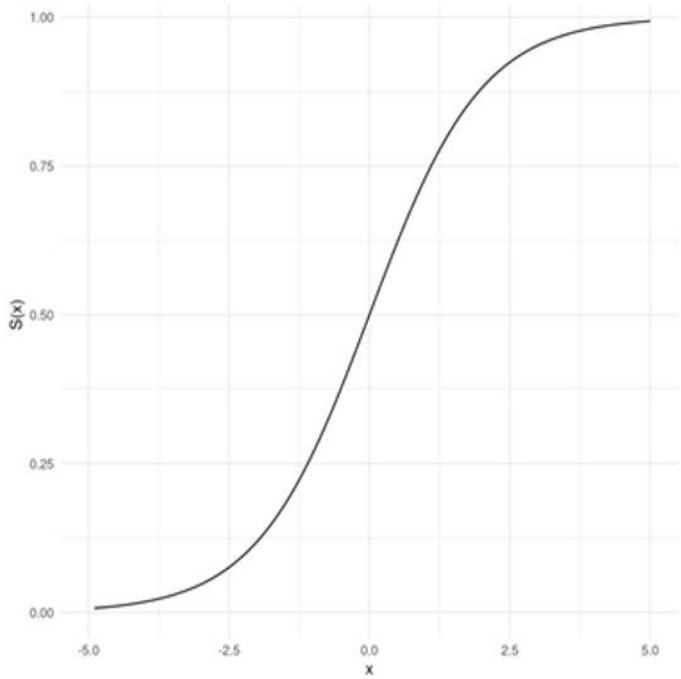

Figure 3.6: Sigmoid function [105].

neural network which can serve to convert its output into a probability score. Figure 3.6 below indicates an example of the most widely used sigmoid functions.

Tanh is another popular activation function. Comparing with sigmoid function, the advantage of the tanh function is that the negative inputs will be mapped strongly negative, and the zero inputs will be mapped near zero. In general, under the same circumstances, tanh function can provide much better performance than the sigmoid function.

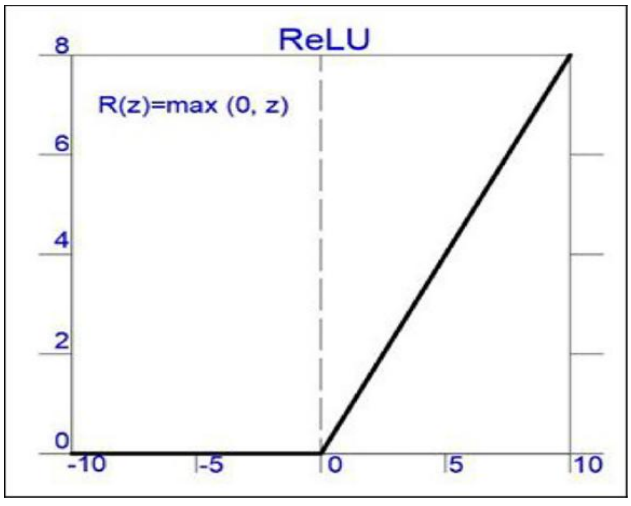

Figure 3.7: ReLu function [105].

Another famous activation function is ReLu (Rectified Linear Unit) activation function, which is the most commonly used one as it can be found in almost all deep neural networks. The ReLu function and its derivative are monotonic [127]. As shown in figure 3.7, the ReLu is half rectified. The function output is zero when the input is less than zero while the function output is equal to the input when the input is above zero. However, any negative input given to the ReLu function turns the output value to zero immediately which causes the inappropriate mapping of the negative values and decreases the ability of the network to be trained properly.

3.2.4 Loss Function Optimization

The learning problem for a neural network is mainly aiming at searching of a parameter vector weight at which the loss function takes the minimum value [128]. The necessary condition suggests that if the loss function reaches its minimum value, the gradient is the zero vector.

Normally, the training would start by creation of some random parameter vectors. Then, the parameters would reduce at each iteration according to the loss function, which is also called the loss decrement. The training would stop when the specified condition is satisfied.

The loss function is a non-linear function of the parameters. As it is not possible to find closed training algorithms for the minima, a search through the parameter space section is generated for each training. For each steps, the loss for the neural network parameters would decrease. Nowadays, five methodologies are commonly applied for

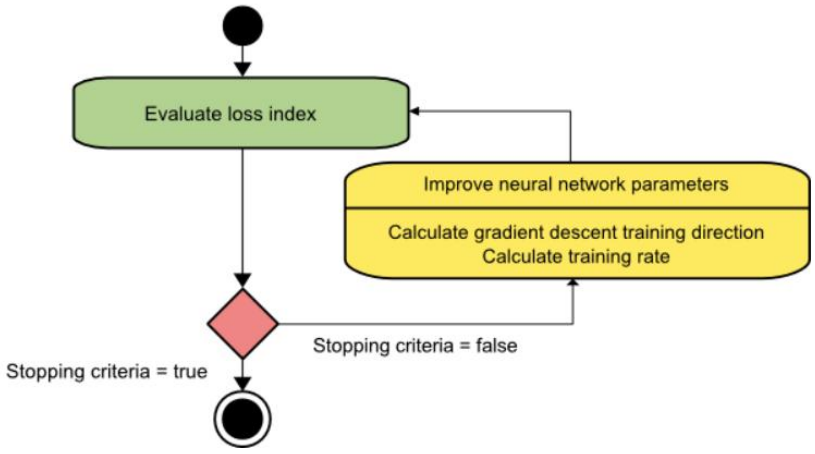

Figure 3.8: Gradient descent iteration process [107].

network training. They are gradient descent method [129], Newton's method [130], conjugate gradient [131], quasi-newton method [132] and Levenberg-Marquardt algorithm [133]. In this section, the three most popular ones are introduced.

The first one is gradient descent (steepest descent). Due to requirements of information from the gradient vector, this first order method is the most straightforward training algorithm [129]. Once begin, the gradient descent method will keep running to update the same weight until the stopping criterion is satisfied. The equation 3.2 and Figure 3.8 below indicate the iterations of the gradient descent.

$$w^{(i+1)} = w^{(i)} - g^{(i)}n^{(i)} \tag{3.2}$$

where w represents the weight for the network parameters, g indicates the training direction and n means the training rate. The training rate can be set to either a fixed value or decided by a one-dimensional optimization which follows the training direction at each step. Though quite straightforward, the gradient descent method also has several drawbacks such as requiring many iterations when functions with long narrow structures are applied. The optimal situation to apply gradient descent is when the network contains massive parameters as this method only stores the gradient vector instead of Hessian matrix [134].

The second one is Newton's method. This training method is a second order algorithm as it requires the Hessian matrix for iteration [130]. The main purpose for Newton's

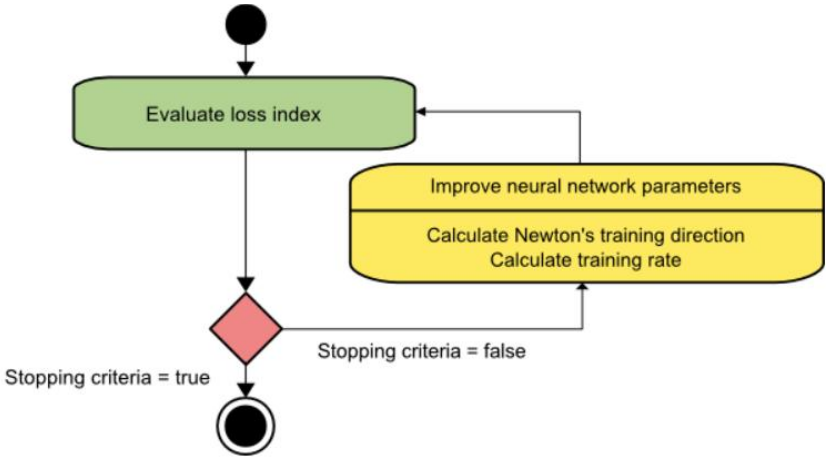

Figure 3.9: State diagram of newton's method [108].

method is to find better training directions through second derivatives of the loss function. Equation 3.3 below shows the mathematical representation of the Newton's method.

$$w^{(i+1)} = w^{(i)} - \left(H^{(i)-1} \cdot g^{(i)}\right) n^{(i)} \tag{3.3}$$

where w denotes the weights of the network parameters, H represents the applied Hessian matrix, g indicates the original training direction and n means the training rate. The vector $H^{(i)-1} \cdot g^{(i)}$ is also known as newton's step or newton's training direction. Compared with other training methods, less steps are required for newton's method to find the minimum value of the loss function. However, newton's method shows bad performances when exacting evaluation of the hessian and the inverse of it are quite expensive in computational terms [130]. Figure 3.9 reveals the state diagram of newton's method.

The last commonly applied training method is called conjugate gradient. This method can be considered as the hybrid of the gradient descent method and newton's method. The convergence inside conjugate gradient method is accelerated and the information requirements associated with the evaluation, storage and inversion of the Hessian matrix is also removed [131]. Equation 3.4 denotes the construction of conjugate gradient method.

$$w^{(i+1)} = w^{(i)} - d^{(i)} \cdot n^{(i)} \tag{3.4}$$

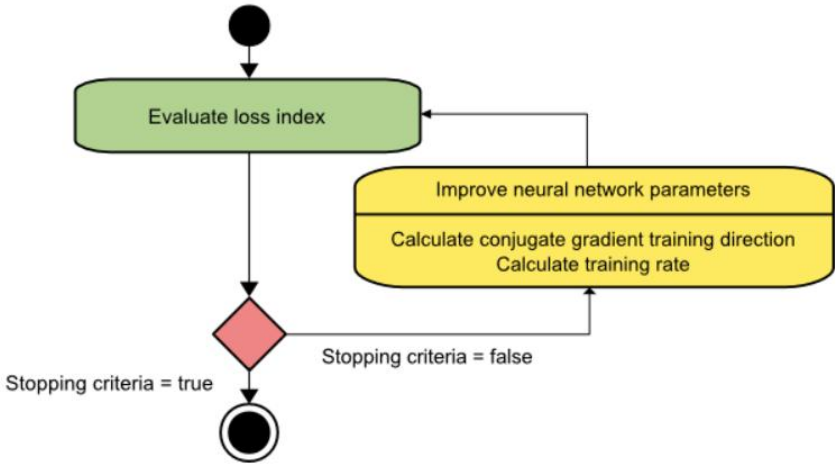

Figure 3.10: Training steps for conjugate gradient [109].

where w means the weights of the network parameters, d is the training direction which decided by the conjugate parameter according to Fletcher and Reeves [135] or Polak and Ribiere [136] calculation methods. n still denotes the selected learning rate. Figure 3.10 shows the diagram for training steps with conjugate gradient method.

It has been proved that the conjugate gradient method is more effective than the gradient descent method when training networks [137]. In addition, as hessian matrix is not required for conjugate gradient method, it is suitable to apply it when dealing with vast networks.

3.2.5 Back Propagation

Even though the optimization methods such as gradient descent can obtain the minimum loss, the initial input still directly links with the final output, which can be represented by a neural network with only input layers and output layers, no hidden layers. In addition, for a deep neural network, multiple hidden layers are normally required. Thus, to apply optimization methods in a deep neural network, the back propagation [138] is essential.

The idea of backpropagation is actually, for each training instance, pass it into the neural network and calculate its output; then measure the output error of the network (that is, the difference between the expected output and the actual output), and calculate how much error does each neuron in the previous hidden layer contributes to the current output result. Iteratively calculates from the next layer to the previous layer until the algorithm reaches the initial input layer. This reverse transfer process effectively measures the error gradients of all connection weights in the network, and finally optimizes the parameters of the layer by applying a gradient descent algorithm in each hidden layer.

3.3 Convolutional Neural Networks

Convolutional neural network is one of the famous neural networks. A convolutional neural network assigns learnable weights and biases to various aspects in an input image to differentiate it from the other [139]. Compared with other neural networks, the image pre-processing required for a ConvNet is much lower. The architecture of the ConvNet is inspired by the organization of the Visual Cortex. In other words, it is analogous version of the connectivity pattern of neurons in human brain. Each individual neuron only responds to stimuli from Receptive field, which is a restricted region of the whole visual field. By concatenating and overlapping all the receptive fields, the entire visual field can be obtained. Three layers play the most important role inside the ConvNet: convolutional layer (kernel), pooling layer and fully connected layer. Figure 3.11 denotes the structure of a convolution neural network.

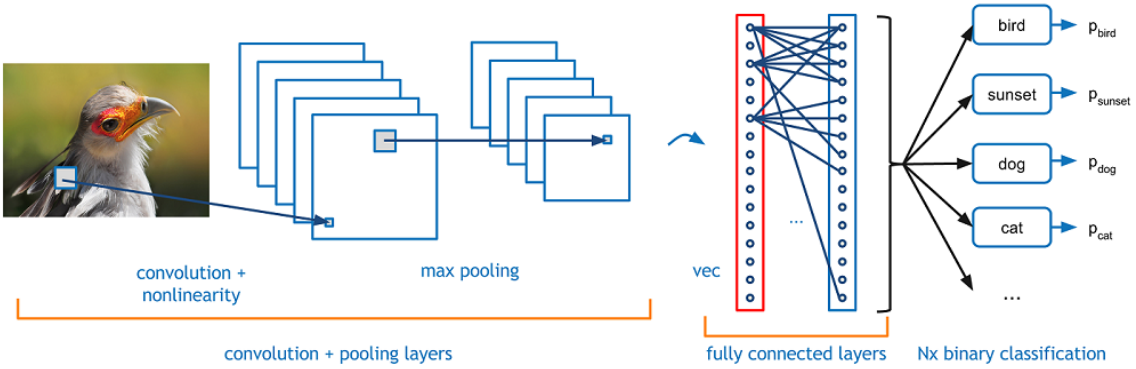

Figure 3.11: Convolution neural network structure [117].

3.3.1 Convolutional Layer

A convolutional layer is a layer which contains convolution operation inside. Imaging an input image with dimensions shown in figure 3.13: 6 (Height), 6 (Length) and 1 (number of input channels) and a selected kernel/filter with size: 3, 3 and 1 shown in figure 3.12. Once the convolution process begins, the kernel will move along the whole of the image from the first column to the last one while does the matrix multiplication operation [140]. After finishing the calculation for the first row, the kernel will move to the next one until the last row of the kernel meets the last row of the image. The kernel movement steps are called strides and can be selected for different whole numbers such as 1 or 2, etc. To allow more space for the kernel to cover the input image, a term called padding is added to the convolution operation. Padding refers to the number of pixels added to an image when it is being processed by the kernel. For example, if the padding

1	0	0
0	1	1
1	0	1

Figure 3.12: Applied Kernel.

1	0	0	1	0	1
0	1	1	0	0	0
1	1	1	0	1	0
0	0	0	0	0	0
0	0	0	0	0	0
0	0	0	0	0	0

Figure 3.13: Input image.

value is set to zero, then every pixel value that is added will be of value zero; however, if the padding value is set to one, a one-pixel border with a pixel value of zero will be added to the image. More specifically, if the 3x3x1 kernel shown in figure 3.12 is applied on the image shown in figure 3.13 with stride of 1 and padding value of 1, a 4x4x1 convolved feature matrix with 16 elements will eventually be created for this attempt. This process and the obtained results are shown in figure 3.14 (Same convolution operation repeats 16 times for this attempt. First 5 operations are represented by different colours. Blue lines represent the first convolution operation, red lines indicate the second operation, green lines represent the third operation, yellow lines indicate the fourth operation and pink lines demonstrate the fifth operation). However, when using multiple kernels, the size of the obtained feature map depends on the number of applied kernels. One kernel would generate one output feature map and two kernels would generate two feature maps etc. For example, three 3x3x1 kernels convolve a 6x6x1 image, the size of the obtained feature map would be 4x4x3 (each of the kernels would have a 4x4x1 feature map like the feature map shown in figure 3.14 and the output feature map would

be gained through the concatenation of the three 4x4x1 feature maps). The objective of the convolution operation is to extract the high-level features from the input image [140]. Usually, each individual convolution layer would capture features such as edges, colour, etc. With the concatenation of several convolution layers, these captured features would be adapted to the high-level features, thus the network would have the wholesome understanding of the entire image.

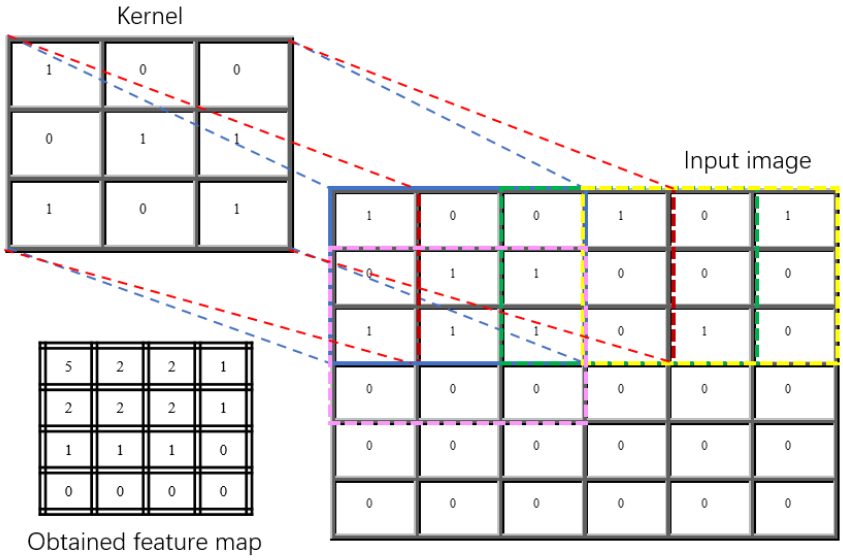

Figure 3.14: Convolution operation example.

3.3.2 Pooling Layer

A pooling layer aims at reducing the spatial size of the convolved feature map [140]. After the pooling layer, the computational power required to process the data is decreased due to dimensionality reduction. Moreover, the dominate features which are rotational and positional invariant are also easier extracted after the pooling layer. The pooling layer is often consisting of two types of pooling functions: max pooling and

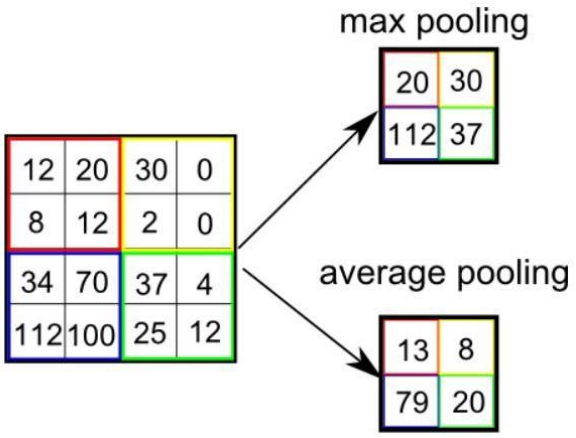

Figure 3.15: Max pooling and average pooling example [118].

average pooling. Additionally, the max pooling is also referred as noise suppressant [141]. That means the noisy activations are discarded together with dimensionality reduction after the max pool layer. The max pooling function returns the maximum value from the portion of the kernel covered image. As for the average pooling, it is simply a noise suppressing mechanism for dimensionality reduction [142]. The average pooling function returns the average of all the values from the portion of the image. In general, the performance of the max pooling is better than the average pooling. Figure 3.15 indicates the results comparison between max pooling and average pooling.

3.3.3 Fully Connected Layer

The fully connected layer performs as the classifier of a convolutional neural network. The way that a fully connected layer works is that it determines the most correlated class from all settled class based on the high-level features contained inside activation maps from previous layer [140]. For example, if dealing with plants classification task, the network predicts the input image is carrot, the values of the contained high level features such as red, triangle formed etc. would be quite high in the activation maps. For an input volume (normally the output from the previous convolutional layer or pooling layer), the fully connected layer generates an output vector with N dimensions where N is the number of classes of the required result. Each number of the fully connected layer represents a certain class. For example, if the output requires classification of 10 different animals, N should be chosen as 10 and each number inside 10 (0-9) represents an animal kind like dog, cat etc. the outputs form of the fully connected layer usually are digit numbers which indicates the prediction possibility. For example, if the output for a 10-digit classification task is [0, 0.1, 0.8, 0, 0, 0.05, 0, 0.05, 0, 0], that means the contained number inside the input image has 10% probability to be a 2, 80% probability to be a 3, 5% probability to be a 6 and 5% probability to be a 7. In total, the sum probability for all the numbers in this condition should equal to 1, which represents 100% probability.

3.3.4 Network Training Methods

The training process of convolutional neural network can be divided into two stages. The first stage is forward propagation which propagates data from low-level to high-level. The other stage is back propagation which propagates errors from high-level to low-level when the obtained results for the forward propagation do not match the expectations [143].

In the forward propagation process, the input graphic data is processed through multi convolutional layers and pooling layers to extract the features. The proposed features are fed into the fully connected layer to obtain the results for the pre-set classification task. The output results for the fully connected layer would become the network results only when they match the expectations. For the convolutional layer, the forward propagation process is to perform a convolutional operation on the input data through a kernel. The layer first convolves the entire input figure to form a local receptive field. Then combines the weighted sum of the weight matrix with the eigenvalues of the input figure (plus a bias). Finally, the obtained result would be passed through an activation function to produce the output. For the pooling layer, it provides pooling operation for features extracted from the connected convolutional layer. After this layer, the dimensionality of the data is reduced. For the fully connected layer, the down sampled features are classified according to pre-set classification model. The output of the fully connected layer will become the final output of the convolutional neural network if they match the expectations. Besides, only after all these layers' processing, a forward propagation can be considered as finished.

When the output of the fully connected layer does not match our expectation, the back propagation process is then carried out. The reason for the mismatch to occur is because of the inevitably information changes caused during the transmission process among layers. The main purpose of this process is to adjust the network weights through training samples and expected values. Generally, the back propagation finds the error between the result and the expected value (expectation), which is known as the total error of the network and returns it layer by layer to calculate each layer's contribution to the error. So that the weights can be updated and applied to reduce the error.

Table 3.1: Training process for convolutional neural network.

1.The network initializes the weight;
2.The input data is propagated forward through the convolutional layers, pooling layers, and fully connected layers to obtain the output value;
3.Evaluating the error between the output value of the network and the target value;
4. When the error is greater than expected value, the error is sent back to the network, and the errors of the fully connected layers, the pooling layers, and the convolutional layers are sequentially obtained. The errors of each layer can be understood as the total error of the network. When the error is equal to or less than the expected value, the training ends.
5. Weight update will be performed and restored based on the calculated error. Then the network will re-enter the second step and keep repeating the following steps.

The more direct training process explanation for a convolutional neural network is shown in table 3.1.

3.3.5 CNNs for Human Gesture Recognition

Because of the significant advancement occurred in computer vision field, CNNs has been widely investigated in human gesture classification tasks. Compared with other machine learning methodologies such as SVM (support vector machine) [144], K-nearest neighbour (KNN) [145] or Naive Bayes (NB) [146], CNN does not require manual recognition and pre-extraction of visual features.

In addition, higher quality features can be discovered and learned by CNN itself. Such advantages draw great attention for engineers. The authors Kim et al. [147] came up with a weighted fuzzy minimum-maximum (WFMM) combined CNN which significantly increased the efficiency of spatiotemporal pattern extraction for video-based hand gesture recognition tasks. Morchanov et al. [148] created a hand gesture recognition system which applies 3D version of the CNN. Raimundo F. Pinto et al. [149] proposed a CNN based hand gesture recognition system which achieved high success rates at a relatively low computational cost.

Rather than relying solely on RGB or grey scale figures, the integration of sEMG signals with CNNs has also garnered significant interest in hand gesture recognition. Notably, Chen H et al. [150] devised a hybrid classifier framework combining CNN and Support Vector Machine (SVM), which directly harnesses sEMG signals to categorize hand gestures. Wu Y et al. [151] further innovated by creating an LSTM-CNN model that efficiently utilizes pre-processed sEMG signals for dynamic gesture recognition, capitalizing on the complementary strengths of Long Short-Term Memory (LSTM) and CNNs.

Moreover, advanced digital signal processing (DSP) methodologies like wavelet transform [152] were also introduced by engineers for CNN based hand gesture recognition. These DSP methods not only facilitate sEMG signal-driven DNN/CNN research but also enhance their capabilities. For instance, Velandia N. S et al. [153] introduced a CNN architecture specifically designed to classify hand gestures using features extracted from sEMG signals through wavelet transform. Similarly, Shanmuganathan V et al. [154] developed an R-CNN (a temporally structured CNN) that

incorporates both sEMG signals and wavelet packet transform to recognize hand gestures, showcasing the versatility and potential of this approach.

Though achieved higher result accuracy, the required signal processing time for common CNN based gesture recognition systems is also enlarged. Thus, this existing problem motivated the applied research in chapter 4, 5 and 6.

3.4 Recurrent Neural Networks

Another popular neural network example is recurrent neural network. Unlike feedforward neural networks, which have no internal memory, RNNs have a form of memory that allows them to process sequences in a sequential and temporal manner [155]. In general, recurrent neural network contains the ability to remember sequence relations, such as connections between previous and post words in one sentence. For normal feedforward neural networks, their memories represent the learning result they learned during the training process. For example, the form of number '2' would be understanded by an image classifier after training it with an image contains number '2' inside. Then the classifier can use the learned knowledge to do classification tasks. Differently, a recurrent neural network remembers features learnt from prior inputs while generating outputs [156]. In other works, the output of the network is not only affected by the weights and biases but also decided by the hidden state vectors which indicate the context based on previous inputs and outputs. This characteristic enables one input to produce multiple types of outputs according to tuning previous inputs in the series. The recurrent neural network often uses 2-D signals such as voice or words as inputs. However, such fixed size vector inputs also limit the recurrent neural network application

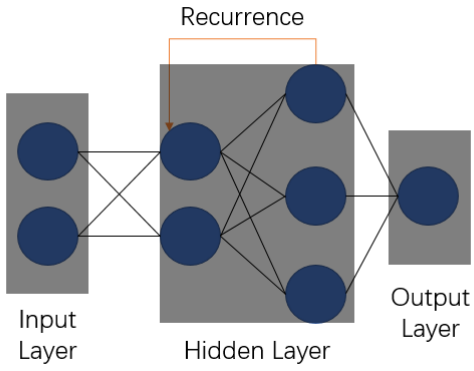

Figure 3.16: Recurrent neural network example [129].

situations as it is a ‘series’ input type which requires no predetermined size. Figure 3.16 indicates the architecture of a basic recurrent neural network.

3.4.1 Recurrent Layer Structure

For the usual neural networks, the parameters inside post hidden layer cannot be affected by previous hidden layer’s outputs directly due to disconnected nodes existing between each layer. Such network structure makes it difficult for the network to solve problems like predicting the next word in a given sentence as the words of that sentence

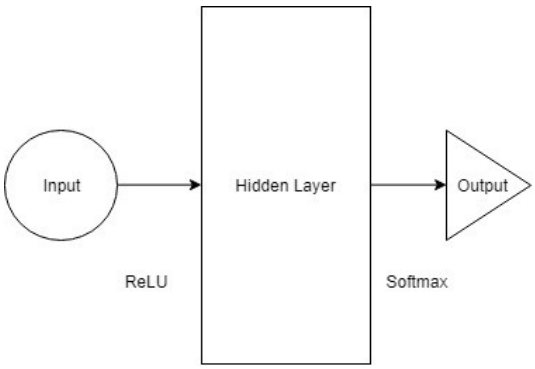

Figure 3.17: Recurrent multilayer perceptron unit example.

are not independent. In order to address this situation, the supervisor recurrent layer structure was created. Figure 3.17 represents the primary form of a recurrent multilayer perceptron unit.

As shown in figure 3.17, the unit consists of three parts, an input layer which feed the previous layer’s output to the hidden layer which contains a selected activation function like ReLU. An output layer which transfers the hidden layer’s results to the next layer.

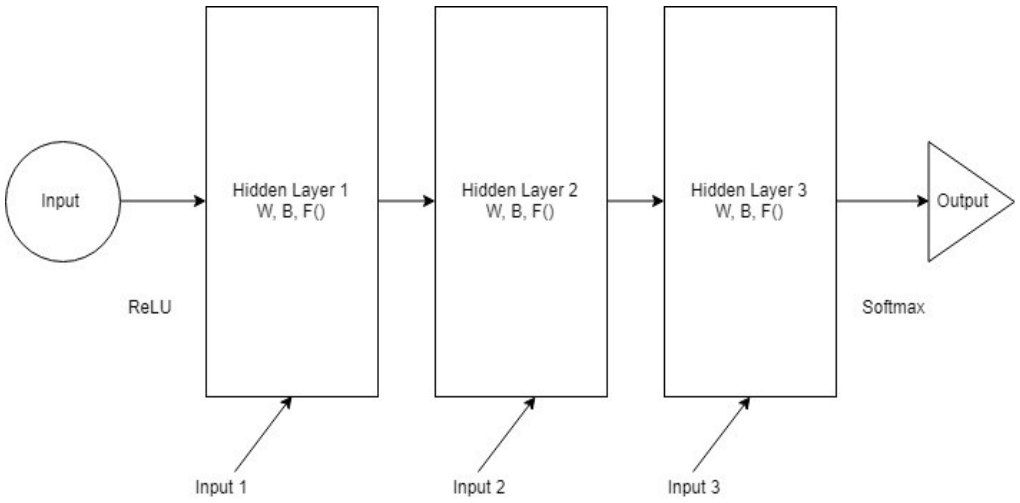

Figure 3.18: Basic form of a recurrent layer.

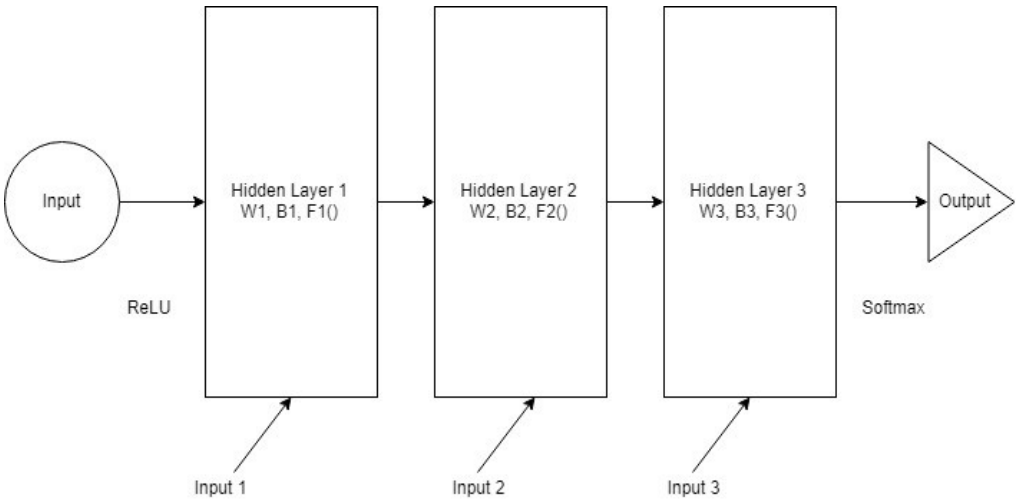

Figure 3.19: Basic form with different network parameters for recurrent neural networks.

By increasing the numbers of the hidden layers, the basic form of a recurrent layer can be achieved as figure 3.18, each hidden layer has its own input, weight and bias.

However, the inputs for each hidden layer currently are not functionable because of the differences between the activation functions, weights and biases. Thus, to make the hidden layers' inputs working, the value for weights, biases and activation functions for each hidden layer must be the same, as shown in figure 3.19.

Now, with the combination of all the hidden layers, the recurrent layer can be achieved as figure 3.20. For each training step, the information is provided to the hidden layers through their inputs. The recurrent layer stores all the input of the previous step and merges this information with the input of the current step. Thus, the correlation feature between the current data step and the previous step can be captured. In simple terms, the result obtained by previous step affects the result for the next step. For example,

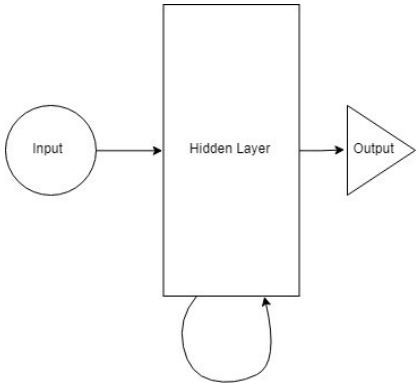

Figure 3.20: Example for a complete recurrent neural network structure.

if a recurrent network needs to predict the 5th letter for word “happy” after we input the 4th letter, the hidden layers inside the network will go through 5 times iteration, each iteration corresponds to a letter.

3.4.2 Network Training Methods

Training a recurrent neural network is almost the same as training other neural networks. The first step is calculating the network output value through forward propagation. The second step is calculating the error value between the obtained output and expectation. The third step is calculating weight gradients through back propagation. The last step is iteratively updating the weights until a minimum value of the error is achieved. However, due to the recurrent neural network aims at processing time series data, the back propagation used in the third step is based on time, which is named as back-propagation through time (BPTT) [157]. The central idea of BPTT is the same as that of the back propagation algorithm: it continuously searches for better values along the negative gradient direction of the parameters to be optimized until convergence. Since the input of the recurrent neural network contains two parts: the current input and the last output, the BPTT algorithm will propagate in two directions when calculating the weight gradient. One is the same as the normal back propagation algorithm which passed from the last layer to the first layer of the network [158]. The other one is passed along the timeline to the initial moment [158]. Thus, more parameters are required when training recurrent neural networks due to the added extra time direction.

3.4.3 Variations of RNN

After the creation of the recurrent neural network, several improvements were applied to further enhance its learning ability. In this section, two most well-known variant recurrent neural networks will be introduced and discussed.

The first one is the bidirectional recurrent neural network (BIRNN) [159]. The recurrent neural network model introduced earlier in this section assumes that the current time step is determined by the sequence of earlier time steps, so it passes information from the front to the back through the hidden state. Sometimes, the current time step may also be determined by a later time step. For example, when writing a sentence, the former words may need to be adjusted according to the later written words in a sentence. BIRNN was created to handle this type of tasks by adding an extra hidden layer that transmits

information from back to front, i.e., an extra loop. Thus, the hidden state (weights and biases) of the BIRNN at each time step depend on previous and past time step's sub-sequences (including the input of the current time step).

The other special recurrent neural network is the long short-term memory (LSTM) network [160]. Among all recurrent neural networks, LSTM is also famous because of its complex recurrent layer structure [161]. For normal RNNs, the training process of the input and hidden state are simply achieved by the tanh function, though, this simple transformation results in problems of keeping or resetting context. The LSTM improves this transformation through applying additional fine structures of the so-called "gates". The gates are composed of S-shaped neural network layers and point-by-point multiplication operations, and they can be selectively used to let information pass through. There are three types of gates that help LSTM to remember information: forgetting gate, input gate and output gate. A forgetting gate can forget or discard some useless information transferred from previous layer [162]. Its main task is to accept a long-term memory (the output passed from the previous unit module) and decide which part of it should be transferred to next layer and which part of it should be removed (forgotten). An input gate [163], which is also called memory gate, is able to determine what new information should be stored in cell for current state. It often consists of two parts: one function that determines what value needs to be updated and another function which creates a new candidate value vector to generate candidate memory. The main task for an input gate is to find the corresponding new information according to the information discarded in the forgetting gate and use it to supplement the discarded one. An output gate [163] determines the output value based on the current state. It also includes two parts: one function to determine which part of the current state needs to be output and another function to process the required output part. Through these gates, LSTM can address problems such as across sentences and the distance between such context resets.

3.5 Development of SNNs

Though inspired by biological neurons, at the implementation level, only marginal similarities between the second-generation neural networks such as CNNs and RNNs and brain-like computation can be recognized [164]. One of the main evidence is that the neurons in deep neural networks are mostly non-linear but continuous functions approximators that operate on a common clock cycle [164] while for the biological neurons, the transmitted information is represented in the form of digital and temporally precise action potentials, which are also known as spike trains to downstream neurons [165]. In other words, the way for the information propagations in biological neurons and second-generation neurons is different. This gap results in the appearance of various problems such as large power consumption and slow inference when applying deep neural networks. Thus, spiking neural networks with different types of spiking neuron models are created by scientists from several aspects [22][166][167] to address those problems.

In 1952, Hodgkin and Huxley [168] were the first to model the principle of the biological neuron operation: the pre-synaptic neurons modulate the membrane potential of post-synaptic neurons and generate action potentials of spikes when their membrane potential crosses a threshold. The model they created can generate action potential from the voltage gating properties of ion channels in a squid cell membrane of its axon. After the Hodgkin and Huxley proposed their model which contains extensive biological details [102], diverse spiking neuron models had been brought forward by engineers for the reduction of the high computational cost [169] and further enhancement, including the leaky integrate-and-fire (LIF) neuron model [170], the spike response model (SRM) [171] and the Izhikevich neuron model [172].

3.6 Spiking Neuron Models

The neurons are the fundamental information transmitting units in a spiking neural network and the features of the input images/signals will be processed and learned by them through sending and receiving spikes. The principle of modern spiking neuron models was proposed by P König et al. [173] in 1996 that a neuron can be regarded as an integrator or coincidence detector. The operation process inside a spiking neuron can be divided into several steps. Once receiving the input spikes, the spike neuron would

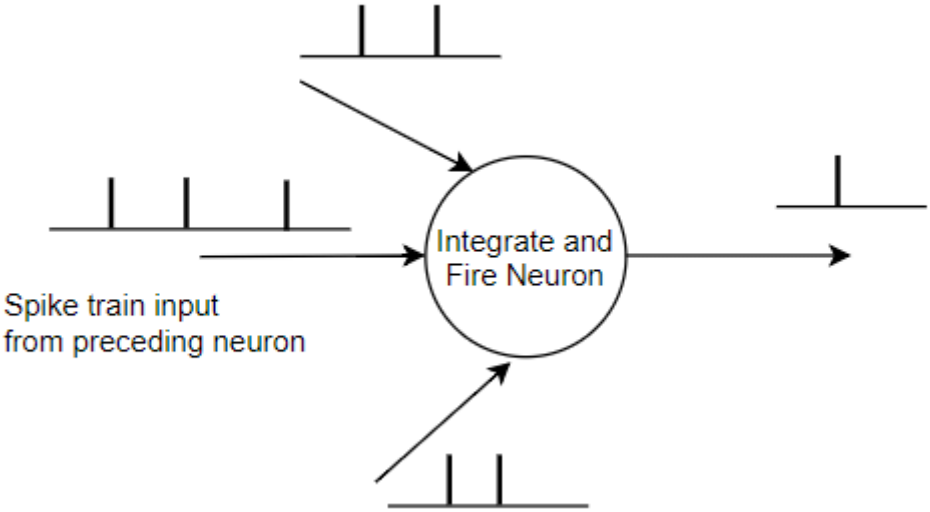

Figure 3.21: Internal process of a spiking neuron model [146].

integrate and transform these spikes into a voltage change which named as post synaptic potential (PSP). Then, the entire PSP of the neuron will be compared with a pre-defined threshold. If the PSP reaches or exceeds the threshold, a spike will be generated and emitted by current spiking neuron to the next neuron. In addition, the input spikes are only identified and integrated at the time instant when they arrived. The internal process of a spiking neuron model which receives input spike trains from 3 pre-synaptic neurons and generates output spikes is demonstrated in figure 3.21.

Figure 3.22 indicates the integrate and transfer process of the membrane potential dynamics $u(t)$ for an input spike inside a spiking neuron. The neuron would generate and emit spikes whenever the membrane potential equals or is above the threshold value ϑ. After emitting a spike, the membrane potential of that neuron would decrease to its resting value u_{rest} and the neuron would enter a phase called refractory period which lasts for a short time. During the refractory period, no further spikes would be generated

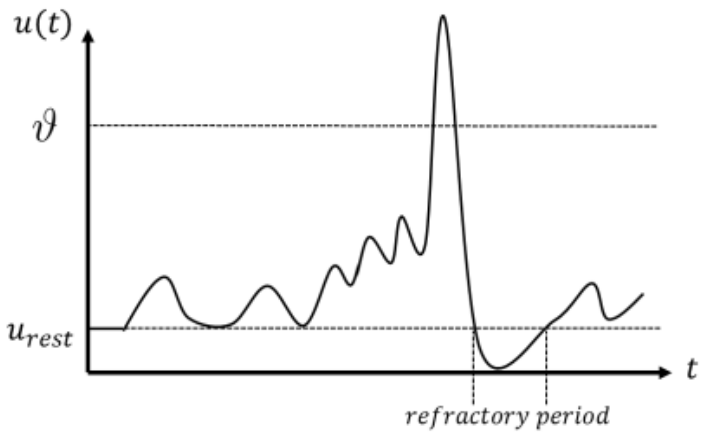

Figure 3.22: Integrate and transfer process of the membrane potential dynamics [146].

by the neuron until the membrane potential reaches its resting value again. Moreover, additional parameters that approximates the membrane potential changes in the neural cortex can be obtained by any typical spiking neuron models.

3.6.1 Hodgkin-Huxley Model

As mentioned in section 3.5, Hodgkin et al. [168] obtained this model from their experiment on a giant squid. Inside their experiment, three different types of ion currents were discovered: viz., sodium, potassium, and a leak current that consists mainly of Cl− ions [168]. One specific sodium ion-based channel and one potassium ion-based channel control the flow of the ions through the cell membrane while a semi-permeable cell membrane separates the interior of the cell and acts as a capacitor. The circuit form of the model is shown in figure 3.23. Both ion channels are represented by a resistor (sodium channel resistance R_{Na} and potassium channel resistance R_K) and the unspecified channel is represented by a resistor with leaky resistance R. Each battery inside figure 3.23 indicates the generated potential by the difference of the ion

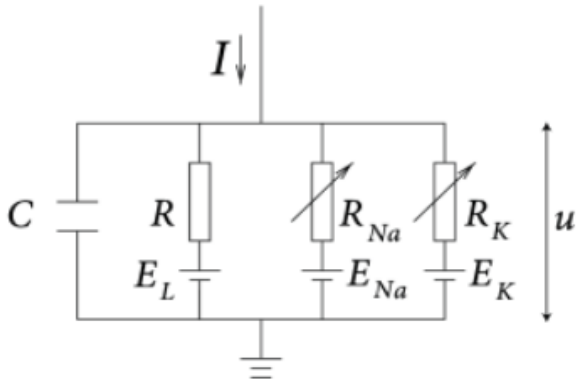

Figure 3.23 Circuit for generated potential by the difference of the ion concentration [141].

concentration (E_{Na} for sodium, E_K for potassium and E_L for the unspecified channel). Additionally, the arrows across the resistors indicate that these are variable resistors. An input current $I(t)$ for the cell could have different results, either it will further charge the capacitor or leak through the channels in the cell membrane [174].

Equation 3.5 demonstrates the mathematical equation of the Hodgkin-Huxley model [175].

$$I(t) = I_c(t) + \sum_K I_K(t) \tag{3.5}$$

where $I(t)$ denotes the input current, $I_c(t)$ represents the split capacitive current which charges the capacitor and $I_K(t)$ indicates the current which pass through the ion channel.

For a standard Hodgkin-Huxley model, there are three types of ion channels [175]: a sodium channel with index Na, a leakage channel with resistance R and a potassium channel with index K. In addition, the presented mathematical equation can be reformed as equation 3.6 according to the definition of a capacitor:

$$C\frac{du}{dt} = -\sum_K I_K(t) + I(t) \tag{3.6}$$

where C indicates the capacity, u denotes the voltage across the membrane and $\sum_K I_K$ represents the sum of the ionic currents which pass through the cell membrane.

Though Hodgkin-Huxley model may be quite simple or even rough from current perspective, its appearance still marks the birth of the spiking neural network.

3.6.2 Izhikevich Model

The previously introduced Hodgkin-Huxley model is computational prohibitive and can only simulate a small number of neurons in real time [176]. In order to enlarge the neuron numbers for the simulation, Russian scientist Eugene M. Izhikevich [172] proposed a superior spiking neuron model in 2003. Unlike the Hodgkin-Huxley model, this model doesn't apply biophysics neurons. Instead, it computes a wide range of spiking patterns for cortical neurons by mathematical equations and the output results for the model is incredibly realistic and biologically plausible [177]. Equation 3.7, 3.8 and 3.9 demonstrate the mathematical expression of the Izhikevich model.

$$\frac{dv}{dt} = c_1 v^2 + c_2 v + c_3 - c_4 u + c_5 I \tag{3.7}$$

$$\frac{du}{dt} = a(bv - u) \tag{3.8}$$

$$\text{If } v \geq 30mV, \text{ then } \begin{cases} v = c \\ u = u + d \end{cases} \tag{3.9}$$

where v denotes the membrane potential of the neuron, I indicates the synaptic currents or injected dc currents, u represents the membrane recovery variable which provides negative feedback to v through accounting the activation of K^+ ionic currents and inactivation of Na^+ ionic currents [178]. The remained four parameters a, b, c, and d are dimensionless parameters and used to determine the spiking and bursting behaviour of the known types of cortical neurons [178]. However, unlike most real neurons, there is no fixed threshold for the Izhikevich model, the neuron firing depends on the history of the membrane potential prior to a spike [172]. As the Izhikevich model can exhibit all neuron behaviours [179], it is widely applied in benchmarking and simulation of spiking neural networks nowadays.

3.6.3 Leaky Integrate-and-Fire (LIF) Neuron Model

Currently, the LIF neuron model is the most popular spiking neuron model applied by neuroscientists to prove their experiment results [180][181]. The LIF model is based on a fact that the neuron action potentials of a given neuron always have the same form [182]. Theoretically, if the form of an action potential remains the same, then the information is not carried by the action potential's shape, but it is contained in the presence or absence of the spikes [183]. Thus, the action potentials can be identified according to the spike events that happed at a precise moment in time. An RC circuit

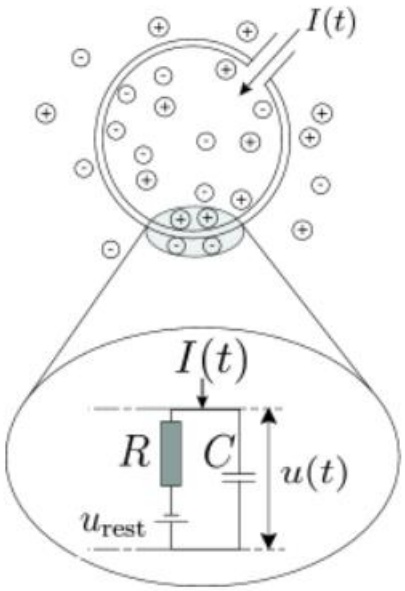

Figure 3.24: LIF neuron model example [155].

with resistor, capacitor and current source in parallel can represent the circuit form of an LIF model, as shown in figure 3.24.

In Figure 3.24 the big circle represents the neuron's cell membrane which acts like a capacitor in parallel with a resistor and a battery of potential u_{rest}. Once receiving an input current $I(t)$, the electrical charge in the cell increases. If the input current $I(t)$ vanishes, the voltage across the capacitor is provided by the battery u_{rest}. Furthermore, there are two separate components that are necessary to define a LIT model dynamics [183]. The first one is the equation that describes the evolution of the membrane potential. The second one is the mechanism that generate spikes. These two components are presented in equation 3.10 and 3.11.

$$\tau_m \frac{du}{dt} = -(u(t) - u_{rest}) + RI(t) \tag{3.10}$$

$$\text{If } u(t) \geq V_{th}, \text{ then } u(t) = u_{rest} \tag{3.11}$$

where τ_m indicates the membrane time constant (membrane capacitance in figure 3.23) of the neuron, $u(t)$ denotes the membrane potential, u_{rest} represents membrane

equilibrium potential, R represents the membrane resistance and $I(t)$ indicates the total currents flows inside the neuron.

Due to the simplicity characteristic of the LIF model e.g., depends on one variable, several neuron behaviours such as phasic spiking, bursting and rebound responses cannot be exhibit through it [184]. Thus, various improvements of the LIF model had been created by scientists [185][186][187]and each of the enhanced LIF model is based on a specific neuron behaviour.

3.6.4 Spike Response Model

The spike response model (SRM) represents a type of biologically, flexible model of the spiking neuron. The basement of this model is kernel functions that describe the effect of spike reception and emission on the membrane potential of the neuron [188]. Because of it is time-dependent and its kernel function is not restricted, the spike response model is considered more general than other classical neuron models. Also, the model is described in a more realistic manner [171]. Equation 3.12 and 3.13 reveal the mathematical representation of the spike response model [102].

$$v(t) = \eta(t - t^{\wedge}) + \int_{-\infty}^{+\infty} \kappa(t - t^{\wedge}, s)\, I(t - s)\, ds \tag{3.12}$$

$$\text{If } v(t) \geq V_{th} \text{ and } v(t) > 0, \text{ then } t^{\wedge} = t \tag{3.13}$$

where $v(t)$ indicates the neuron membrane potential, $t^{\wedge}$ represents the time when neuron fires its last spike, $I(t)$ means the external current, η is the form of the action potential and its spike after potential, κ denotes the linear response to the pulse input and V_{th} is the spike fire threshold.

3.7 Neural Coding Algorithms

Neuron coding aims at examining information processing inside neurons e.g., the relation between spike trains and the transmitted information forms. As the information propagation inside a spiking neural network is different from the conventional system, the external input stimulus must be converted to the spikes which contain the information. A number of neuron coding paradigms have been developed over the past decades. The

most famous coding methods, which are known as rate coding and spike count coding [141][142], codes the information in the form of frequency/rate of spikes in a limited time period. As an alternative to the rate coding, there is another famous coding strategy named temporal coding[189] or latency coding[190], which relies on the precise timing of action potentials or inter-spike intervals and codes the information in the form of the first spike's arrival time. Apart from the temporal coding, there are other spike coding methodologies such as population coding [191] which uses the joint activities of a number of neurons to represent stimuli.

3.7.1 Spike Count Coding and Rate coding

When applying spike count coding strategy, the neuron counts a number of input spikes until firing [102]. As the integrator (neuron) excludes the leakage contribution to the membrane potential in the situation, the neuron response is only dictated by the number of input spikes for the given synaptic weight irrespective of how fast the spikes arrive at the post synaptic neuron (firing rate) and the arriving time of the spike (spike timing). Different spike numbers can be induced by different input stimuli through applying different number of spikes, which endowing the neuron with input selectivity. Additionally, the inter spike interval in such scheme is most robust to variability among all the coding methods [192]. Figure 3.25 indicates the spike count coding for a sensory neuron.

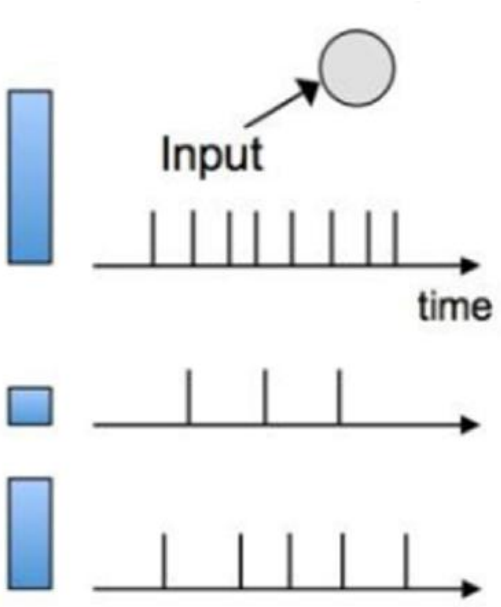

Figure 3.25: Spike count coding example for a sensory neuron [165].

Where the blue bar at the left side of the figure denotes the magnitude of the input stimuli. The larger (higher bar) the input is, the more spikes are generated.

The rate coding was first created by ED Adrian and Y Zotterman [193]. The sensory neuron is represented by means of a firing rate. The firing rate is the measurement of spikes in a certain time window and thus refers to the average spike number per unit time [192]. It should be note that the rate coding is often called spike count coding as opposed to temporal coding essentially count spikes to evaluate the temporal-average of incident spikes [194]. With the applied time window, a rate coding strategy can easily transfer into a spike count coding strategy. However, there are some disadvantages for the rate coding strategy as it mainly focusses on the magnitudes of the input spikes but ignores the encoded temporal structure inside spike trains. Additionally, rate coding strategy is essentially time consuming as it requires each neuron to integrate input spikes sufficient to evaluate the temporal average of spike number per unit time.

3.7.2 Temporal Coding

Temporal coding represents the spike trains through using spike timing and can efficiently map the input information into the sequence order of spike trains rather than the average firing rate. Researchers created temporal coding to replace the rate coding as rate coding is not sufficiently fast enough to duplicate the fast decision making of the biological neural systems. Various research proves that the typical temporal resolution

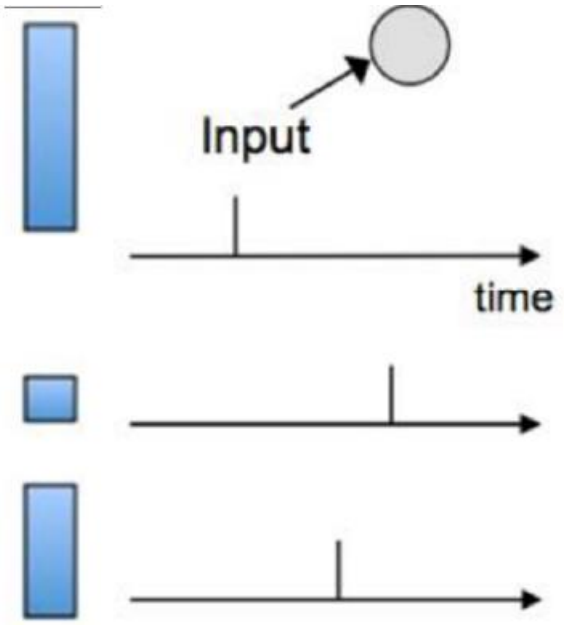

Figure 3.26: Latency coding example [168].

of neural coding is in a range of milliseconds [195] which indicates the importance of applying precise spike timing when coding. As the objection of the rate coding, a number of research have proved that the decision-making neurons can provide responses just after a few input spikes when applying temporal coding strategy [190][196].

In general, temporal coding strategy can express features of spike trains which cannot be represented by the firing rate. Figure 3.26 denotes an example of the temporal coding, latency coding scheme, which is proposed by Thorpe [190].

Where the blue bar still represents the magnitude of the input spikes. Unlike rate coding, the magnitude is converted into precise time of spikes and the information carried inside are represented by individual spikes rather than a spike train. High intensity of the stimulus will result in a low latency of spike generation and low magnitude spikes will results in delayed spike generations [197].

3.7.3 Population Coding

The population coding scheme uses the joint sequences of action potentials or spikes of multiple neurons to express input stimulus [198]. For population coding, each neuron is endued to a certain input sets and each of the input response value is determined by combination of many neurons [199]. Unlike other coding schemes, the unique coding strategies of population coding have been revealed by a number of experiments [200][201][202][203].

i. **Independence: each neuron responds independently to the stimulus.**
ii. **Decorrelation: neurons will interact and generate a decorrelated representation of the stimulus.**
iii. **Correction: neurons respond redundantly to combat noise, thus errors can be corrected.**
iv. **Synergistic coding: the unachievable information from separate neurons can be obtained through the group of neurons.**

3.7.4 Sparse Coding

The sparse coding strategy represents input stimulus through encoding the strong action potentials or spikes of a small group of neurons [204]. There are two types of

sparse coding based on different focusing objects. The first one is temporal sparseness, which means the sparse is achieved on a relatively small number of time periods. The second one is sparse for the population of neurons, which indicates the amount of neurons that generate strong action potentials relatives to the total neuron number in a time period [205]. When applying sparse coding scheme, the interference between distinct neuron outputs is much less likely to happen and the learning for the spiking network is much easier.

3.8 SNN Architecture

The structure of the SNN is similar to the traditional Artificial neural networks with different neuron activations functions and spikes, e.g., carrier of the information. Thus, for a common SNN, there are two parts which decide the network performance. One is the encoding schemes which were introduced in section 3.7. The other one is the applied neural model. Figure 3.27 represents an example of the architecture of a multiple layer spiking neural network.

As shown in figure 3.27 the structure is very similar to ANNs. With various coding layers which placed before the spiking layer, the unavailable event inputs can be transferred into understandable spike trains for the network.

Inside SNNs, the information carried by the spike trains is propagated via synaptic weights. These synaptic weights react to the relevance of the connections between the

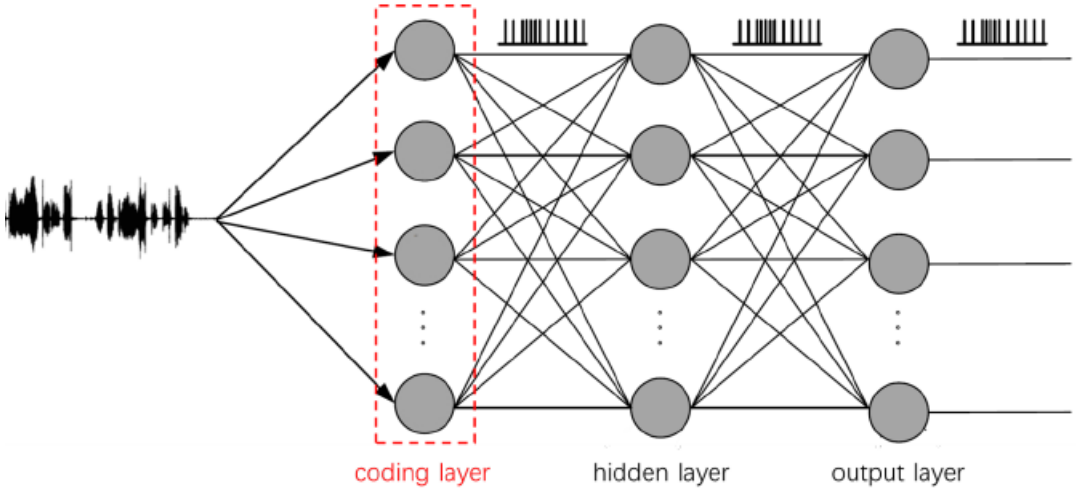

Figure 3.27: Example architecture of a multiple layer spiking neural network [180].

spiking neurons and are the key learning parameters inside a spiking neural network as they control the spike activations, and the output spike patterns.

Additionally, the SNN architecture is able to exceed standard feedforward networks through applying additionally signal process techniques or changing neuron connections. Due to this special characteristic, vast unique spiking networks have been developed among past fewer years. for example, convolutional spiking neural network (CSNN), which combines convolutional layer and spiking neural networks by using difference of Gaussian (DoG) function as edge detection and encoding method, together with weight sharing neuron groups (kernels) [206][207][208]. The recurrent spiking neural network, which enhanced the sequence data processing ability of the SNNs through introducing recurrent layer structure [209][210][211].

3.8.1 SNN Training Methods

During past years, three training methods have achieved successful results in various SNN applications. The first one is supervised learning, the most common training method for ANNs which involves feed-forward and back propagation strategy introduced in section 3.2. However, the supervised learning requires differentiable activation functions to reflect error gradients. As the discrete asynchronous event generated by the spiking neurons are non-differentiable, this training method is not easy to apply for SNNs. The second method is unsupervised bio-inspire learning, which often uses some forms of STDP (spike time dependent plasticity) as part of the learning procedure [212][213]. During the past decades, a number of works have demonstrated vary successful applications on SNN with STDP [214][215][216]. Yet, this research also denoted some drawbacks of the current unsupervised bio-inspire learning strategy. One of the main issues is that the Hebbian learning based bio-inspired learning algorithm does not achieve enough testing accuracy as supervised learning algorithm for classification or recognition of tasks. In addition, STDP training parameters must obey the spike distribution of the input data. Moreover, the not learning problem (neuron domination) is more likely to happen when excitatory and inhibitory actions which are generated by coupled spiking neurons appear. The reason is these actions are quite sensitive to the spike timing and weights of the spikes. The third training scheme is conversion of pre-trained ANN parameters. The parameters such as input data, weights and neurons are first trained in a conventional ANN with back propagation and then

transferred to spiking version. Moreover, most of the spiking training strategies nowadays are highly dependent on the applied spiking neural model and information encoding schemes introduced in chapter 4 since different learning behaviours can be resulted by different spiking models for the same event.

3.8.2 Supervised Learning

The earliest record for applying supervised learning on SNN training process came from Bohte et.al. [217]. Their published work, SpikeProp, accounts the spike timing in the cost function and successfully classified the XOR problem, a problem that predict the output of XOR logic gates by given two binary inputs. Like conventional ANNs, the supervised learning for SNN also requires gradient-based back propagation to minimize the error between the desired output spike trains for given inputs. The weights between the spiking neurons are justified through each iteration according to applied optimization methods as introduced in this chapter earlier. The advantage of the supervised learning is that it can achieve the equivalent accuracy for classification or recognition tasks to the ones of conventional ANNs. However, as the spikes are discrete in time i.e., the activation function for them is not derivable, the only way to enable supervised learning for SNNs is to find an approximate real-valued and differentiable surrogate to the activation function, which became the major limitation for developing supervised learning application on SNNs currently.

After the birth of the SpikeProp, many of its variants were created for multiple applications. For example, Lee et.al. [218] created a SNN which treated membrane potentials as differentiable signals. Mostafa et.al. [219] published their work which uses the timing of the first spike for each neuron as its activation value, thus the input and output function for that SNN is differentiable and became very sparse with the ability to process complex temporal information of input spikes. Also, there are many other attempts appeared among past decade to break through the limitations of the supervised learning for SNNs. One attempt was focusing on generating spike trains with excepted spike times through training of synapses, like ReSuMe [220][221] (remote supervised learning), SPAN [222] (spike pattern association neuron) etc. Another attempt was focusing on the management of alternatives of spike functions, like backpropagated membrane potentials of a spiking neuron. Spike Layer Error Reassignment (SLAYER) [223] training algorithm is one of the advanced spiking learning algorithms among these

works. It approximates the derivative of the spike functions according to neuron state changes and assigns the errors to previous layers which allows the network to adapt developed gradient descent methods such as RmsProp [224] etc.

3.8.3 Unsupervised Bio-Inspired Learning

The unsupervised learning means training the network without pre-existing labels. The most representative scheme for unsupervised bio-inspired learning for spiking neural network is STDP [225]. STDP is a Hebbian learning rule with very intuitive explanation. When applying STDP, the synapse's strength between two neurons is depends on the relative timing of the spikes from the pre-synaptic neuron and post-synaptic neuron. If the pre-synaptic spikes fires before the post-synaptic neuron, the weight value of this connection will increase i.e., the connection between these two neurons is strengthen. This progress is named long-term potentiation (LTP) in STDP. On the contrary, if the post-synaptic neuron fires before the pre-synaptic neuron, the weight value will decrease. Like LTP, this phenomenon is also authored a name: long-term depression (LTD). The mathematical form of the STDP is shown in equation 3.14.

$$\Delta w = \begin{cases} Ae^{\frac{-(|t_{pre} - t_{post}|)}{\tau}}, t_{pre} - t_{post} \leq 0 \; and \; A > 0 \\ Be^{\frac{-(|t_{pre} - t_{post}|)}{\tau}}, \; t_{pre} - t_{post} \geq 0 \; and \; B < 0 \end{cases} \tag{3.14}$$

where the upper equation represents the mathematical form of LTP, and the lower equation presents the LDP. For both, Δw indicates the weight changes, A and B are constant parameters, τ denotes the timing window constant and t_{pre}, t_{post} represent the absolute timing of the pre-synaptic and post-synaptic spikes.

Figure 3.28 shows the working principle of STDP. Due to the learning rule of STDP strictly depends on the connection between two layers which is lacking coordination with other parts of the network [226], the performance is not good enough in terms of accuracy for multi-layer spiking networks.

Moreover, since unsupervised learning is usually implemented for local events which do not have extra information to supervise the SNN, it may be created under one or more

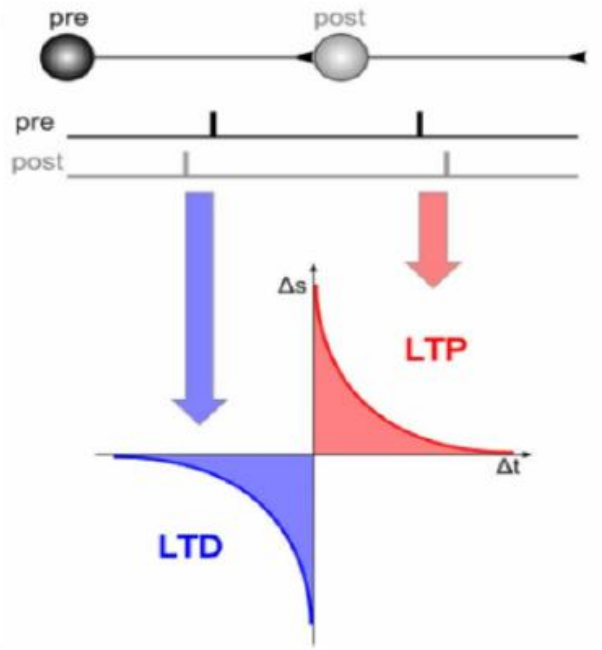

Figure 3.28: Working principle of STDP [199].

of the synaptic plasticity rules shown in table 3.2. However, only the Hebbian term-based learning scheme is currently applied in SNN, which is rule No.3. Other weight modulation rules are still too challenged to be used for SNN training.

Table 3.2: Synaptic plasticity rules for unsupervised learning

1.The reward or decay of the synaptic weights should base on the presence of the spiking activities [227].
2.The reward or decay of the synaptic weights must be independent of pre-synaptic spikes [228].
3.The synaptic weight modulation caused by both pre-synaptic and post-synaptic spiking activities [229].
4.The synaptic weight modulation caused by only pre-synaptic spikes but excludes post-synaptic spikes [230].

3.8.4 Learning by ANN Conversion

The development of ANN conversion training algorithm aims at avoiding the requirement of complex and dedicated training for SNNs. When applying, the first step for the methodology is to train a known ANN network. Then, adapting the pre-trained weights and parameters of the ANN network to its equivalent SNN counterpart [231]. Finally, mapping the input and output spike encoding and decoding of the converted SNN.

The main advantages of ANN conversion method exist in two parts. One of them is that various numbers of highly efficient training techniques developed for ANNs can be leveraged with a negligible training overhead [232]. The other aspect is that the performance of the SNN is ensured even for some challenging datasets (datasets that require massive parameters in network training process) such as ImageNet [233]. In other words, the accuracy loss for the SNN is quite small when compared with its original ANN. Originally, the ANN-SNN conversion requires modifications on the network topology of the formal network [234]. Due to later achievements, it is possible to do the transformation directly. However, the disadvantages of ANN-SNN conversion are obvious.

First, not all ANN operations can be transferred into SNN operations. Thus, the networks which contained these functions are hard to be converted. For example, the max pooling operation mentioned in section 3.3.2. Since the max pooling is not a linear function, it cannot be approximated by spike-to-spike basis.

Second, when converting the encoding and decoding parts, the vast required parameters with their normalization degrade the efficiency of the SNN and bring additional costs.

Generally, the ANN-SNN conversion requires finding the balance between the performance and the computational costs of the network.

3.9 Spiking Neural Networks for Human Hand Gesture Classification

Over past decades, vast number of applications of SNNs have been demonstrated by researchers among various aspects such as speech recognition [235], image classification [236] etc.

Among all the aspects, it is worth to outline the achievements for SNN on human hand gesture recognition tasks. As presented in section 3.3.5, the conventional ANN based hand gesture recognition methods demand extensive processing time which limits its real time application performance. Through SNN, time consuming problem can be easily addressed. Currently, the SNN based human hand gesture classification methodologies can be divided into two popular parts according to the applied data format. One is classifying hand gestures by employing the collected 2D signals like EMG or EEG. A work presented by Mukhopadhyay et.al [237] in 2018 demonstrated a 89.39% recognition accuracy for an objects hand gestures while the ANN achieved 93.33 % recognition accuracy under the same conditions. As a result, the processing time for this kind of approach is highly decreased but the accuracy is still uncompetitive compared to the results for conventional ANNs.

Similar research was also conducted by Haofeng Chen et al. [238] in 2021, involving conventional Artificial Neural Networks (ANN) and sEMG-based gesture recognition. Similar challenges or issues also encountered in their study. The other one is based on the image or video of hand gestures such as videos obtained by a dynamic visual sensing (DVS) camera. In 2018, Shrestha et.al. proposed their work on spiking backpropagation based hand gesture recognition tasks with a 96.49% accuracy for 10 classes [223]. Moreover, in 2017, Arnon Amir et al. [239] conducted a research study that utilized Spiking Neural Networks (SNN) and DVS camera-recorded hand gestures, known as the IBM Gesture Dataset. Their research paper reported an impressive accuracy rate of 97.8%. However, the required processing time is increased as the video or image input is more complex when converting to spike trains compared with 2D signals. Furthermore, an alternative approach exists. Enea Ceolini et al. [240] have introduced a fusion mechanism that combines DVS image data with sEMG signals. In their research, they have designed a Spiking Neural Network (SNN) structure tailored for the specific purpose of classifying five different gestures using synchronized visual and

electromyography data. Additionally, the SNN is also implemented on neuromorphic devices known as Loihi and ODIN + MorphIC. Their findings indicate that the SNN approach offers a notable enhancement in energy efficiency. However, it's worth noting that the inference time also experiences an increase, approximately in the range of 20% to 40%. These drawbacks inspired the research work in chapter 6 which finds a trade-off between the consumed time and performance of SNN based hand gesture recognition.

An additional reason that draws increasingly attention of applying SNN on human hand gesture recognition tasks is the energy efficiency characteristic of the SNN. Eventually, the designed network has to be equipped by a hardware such as electric prosthesis to help the disabled. With the low power consumption and high computing capability of the SNN, the SNN based electric prosthesis is capable of a longer standby period without extra power assistance compare with conventional electric prosthesis. Meanwhile, the difficulty of design of neuromorphic chips in architecture has been conquered by researchers and industrial partners. Various neuromorphic chips like TrueNorth [24], SpiNNaker [241], Loihi [25] have been designed and tested for real time applications, which further inspires the research on SNN based human hand gesture recognition.

3.10 Conclusion

In this chapter, the review of ANN and DNN techniques has been illustrated. In particular, the application for CNN in human gesture recognition tasks were introduced and discussed. The principles for the operations of neural networks in terms of neurons, neural models, training method, activation functions etc. were also included. The fundamentals of CNN and RNN were mentioned and discussed specifically in two separate parts as they represent the two main branches of current DNN technique. The CNN structure has significantly contributed to the visual based hand gesture recognition and was the inspiration for the chosen research in chapters 4-6 of this thesis.

In addition, as the spatial features can be well handled by the CNN structure, the idea of use spatial feature in gesture recognition is also applicable in the spiking neural network (SNN). By combining both CNN and SNN, the actual spatial-temporal feature of the EMG signals can be analysed which was the key inspiration for the work in chapter 4, 5 and 6.

The basic concepts and operation principle of SNN have also been reviewed in this chapter. As the most advanced generation of the neural network, the SNN shows high performance when receiving, processing and outputting the event-based information. The potential of SNN in terms of energy efficient and real time applications demonstrated in this chapter provides the theoretical fundamental of the research work in chapter 5 and 6.

Although the advantages of the SNN are conspicuous, there are still many challenges such as uncompetitive performance compared with conventional ANNs, lacking reliable training algorithms etc. exits and needs to be addressed. Besides, the standard framework of the SNN haven’t been accomplished yet.

Chapter 4

Intersected EMG Heatmaps and Deep Learning based Gesture Recognition

4.1 Introduction

A novel methodology which aims at reducing the difficulty of processing sEMG signals-based hand gesture recognition is presented in this chapter. Section 4.2 shows the pre-processing of the raw sEMG signals and a unique energy density map generation technique which designed for further removing of the noises and reducing signal complexity. Section 4.3 indicates the structure of the novel deep convolutional neural network for hand gesture recognition tasks. The output results of our experiment are discussed and compared with other relevant research works are presented in section 4.4. Section 4.5 contains the conclusion and discussion of the recognition system.

The raw sEMG sequence is signal sequences in time via magnitude forms. After converting raw signals into energy density maps, the information contained inside the combination of the sensor's locations can be observed and processed by the neural network. With these additional features, the recognition accuracy is further enhanced according to the obtained results. Besides, as mentioned in section 3.3, the convolutional neural networks are excellent on addressing image-based classification tasks. Thus, the CNN was chosen as the classifier for this chapter's research work. Moreover, the whole intersected heatmap procedure that is presented in section 4.2 is much faster compared with other existing methodologies. Based on the results, the neural network shows a faster processing speed when applying the datasets that have been processed by intersected heatmap methodology. Thus, the entire training and reacting time of the hand gesture recognition system is significantly decreased, as well as providing an improvement of the recognition accuracy.

4.2 Raw sEMG Processing and Intersected Heatmaps Algorithm

After collected by the device introduced in section 2.5.1, a 3rd order Butterworth digital filter which can remove electronic equipment noise and motion artefacts is applied for filtering. When finishing, the sEMG signals are formed in Hamming windows of 250ms window length and 50% overlap.

The intersected heatmap algorithm refers to one of the key inventions of the research described in this chapter. The heatmap indicate the strength of the electrical activity of the sEMG sensors as a results of different hand gestures which cause various muscle activities. According to section 2.5.1, the collecting device contains 128 sensors. Each of the sensors is represented by a channel which corresponds to a specific muscle area on the forearm. The values that represent the energy density are obtained from the mean absolute values which were computed over a 250ms window of the original sEMG signal based on the equation 4.1 shown below.

$$E = mean(\frac{S_f}{\max(|S_f|)}) \tag{4.1}$$

where S_f indicates the filtered sEMG signals, max function indicates the maximum value and $mean$ function represents the mean value. Additionally, the calculated energy is converted into RGB colours. The brighter the colour the higher the energy level the sensor contains. An example of the heatmap which represents a gesture performed by participant 5 is shown in figure 4.5. The left heatmap shows the array formed by the first

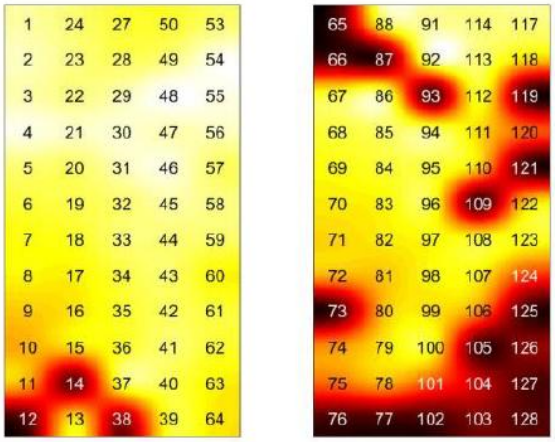

Figure 4.1: Heatmap for close gesture of participant 5.

64 sensors placed on the flexor muscles and the right part of the heatmap refers to the array contains the remaining 64 sensors placed on the extensor muscles.

Because most of the peak muscle activations occurred in small regions of the heatmap, as revealed by figure 4.1, vital information on certain areas where the muscle activity occurs can be lost when the arrays are divided into smaller rectangles or sensors are averaged. Through selecting sensors from the locations with high activation levels, the represent muscle activation patterns and omit readings from sensors which provide little or even sprues information can be improved. To serve this purpose, the RGB heatmaps were first converted into grey scales figures which remained in the same format, e.g., same sensor locations and arrangements to enhance visibility of the sensors that includes the highest energy densities. Only the most active sensors were re-coloured in white after the transformation, while less active and non-active sensors were represented by black. Moreover, based on previous experiments, there is an optimal number of sensors which provide useful information for all features. Identifying theses optimal number of sensors helps minimize overlapping common features to further reduce the required training time. Thus, an empirical strategy was applied to investigate and determine optimal sensor numbers.

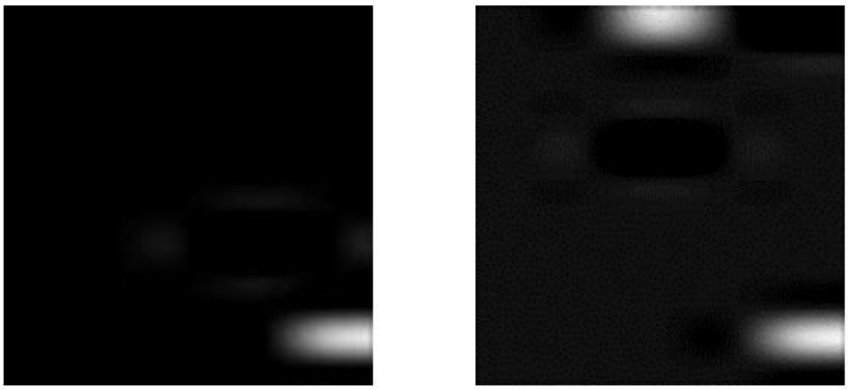

Figure 4.2: (a) for 1 sensor; (b) for 2 sensors.

Figure 4.3: (a) for 3 sensors; (b) for 4 sensors (2 of the sensors are close to each other).

The procedure starts by applying only one sensor which contains the highest energy strength for a given gesture for both bands to form the first partial heatmap. Then the additional partial heatmaps are formed by introducing additional sensors with the next highest energy level until 50 sensors as the model performance does not increase when going above. Figures 4.2 and 4.3 show the example of the heatmaps generated from 4 topmost sensors for a "close" gesture for the lower band. In general, 340 partial heatmaps (30 partial heatmaps for 9 able bodied participants and 5 for 14 amputees) were obtained after the first process for each set of the sensors selected (from 1 to 50 sensors). In total, 136000 partial heatmaps were generated (340 per-gesture × 8 gestures × 50 sensor selection conditions).

After obtaining the partial heatmaps with the optimal sensor numbers, the intersection algorithm is introduced. Given the mixture of healthy participants and amputees with varying degree of injuries, the partial heatmaps obtained at the current stage still contains spurious information. For example, outliers or signals with high degree of noise. These noises could cause misclassification during a training process and delay the network response time for real time applications. Due to the presence of misinterpreted messages necessitates the network to expend additional effort in filtering and distinguishing genuine features from the noise. The intersection methodology shown in table 4.1 that determines the common heatmap features for each gesture for all participants presented below successfully addressed these problems.

Table 4.1: Intersected Algorithm for Participants

Input: partial heatmaps (heatmaps which contains certain numbers of sensors) for all subjects' gestures for all numbers of sensors considered
1. Collect all partial heatmaps for same sensor number for all subjects' gestures. These heatmaps represent muscle activation patterns captured by a specific number of sensors.
2. Select one partial heatmap for one gesture from one subject. The heatmap will be used for comparison with all other heatmaps of the same gesture from different subjects.
3. Subtract (pixelwise) the heatmap of the selected subject's gesture from all remaining heatmaps (from all other users) for the same gesture. This step is performed pixel-wise to eliminate subject-specific features and highlight differences.
4. Inverse pixel value of obtained subtracted heatmaps to generate intersected heatmaps between the chosen subject and each of the other subjects. This inversion helps to identify common features between the chosen subject and each of the other subjects.
5. Repeat steps 2 to 4 for all other subjects
6. Repeat steps 2 to 5 for all remaining gestures
7. Repeat steps 1 to 6 for all remaining sensor number selections.

The intersected algorithm was applied for the selected number of sensors considered in each case for all participants, each gesture in turn. In this chapter, a maximum of 50

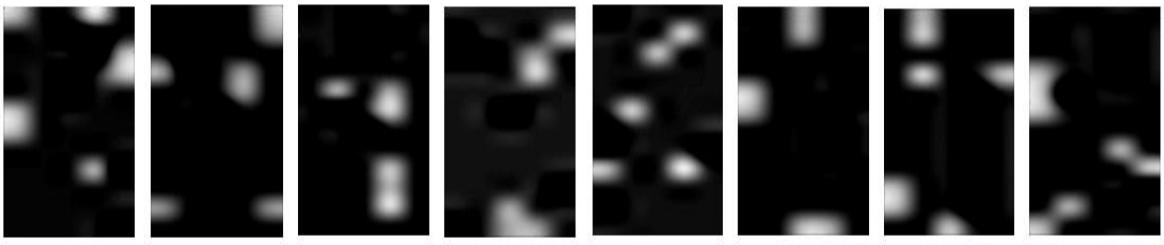

Figure 4.4: Intersected Heatmap for 35 channels for 8 gestures. Gestures from left to right: close; extension; flexion; point; lateral; tripod closed; palm down; palm up.

sensors and 8 gestures (1: palm up; 2: palm down; 3: tripod closed; 4: lateral; 5: point; 6: flexion; 7: extension; 8: close.) experiment was achieved and based on recognition results, the best accuracy was achieved by using 35 sensors.

57630 intersected heatmaps were obtained for each gesture after the intersection algorithm for each set of sensors were selected. In total, 16136400 intersected heatmaps were generated (57630 per-gesture × 8 gestures × 35 sensor selection conditions) for the research. The applied 50 sensors aim to find out what is the best number of sensors for intersection heatmaps, and the best result was achieved with 35 sensors, thus 16136400 intersected heatmaps were generated. An example of the intersected heatmaps for participant 9 is shown in figure 4.4.

4.3 Deep Convolutional Neural Network Structure

As the advantages of applying convolution neural network in image based deep learning tasks are significantly superior compared with other types of neural networks, was indicated in section 3.3.5, a novel CNN was created for the research work in this chapter.

The CNN's architecture was grounded in the VGG [242] framework owing to its straightforward design, ensuring that alterations to the network structure remain manageable and avoid potential complications. Furthermore, given that the prosthesis operates on battery power, minimizing energy consumption was crucial, hence the decision to incorporate just three convolutional blocks to keep the layer count as low as feasible. Shown in figure 4.5, the whole CNN can be divided into convolutional blocks, dropout layers and output layer. In total, three convolutional blocks were applied and each of them contains 3 convolutional layers with a fixed size 3x3 filter. A max-pooling

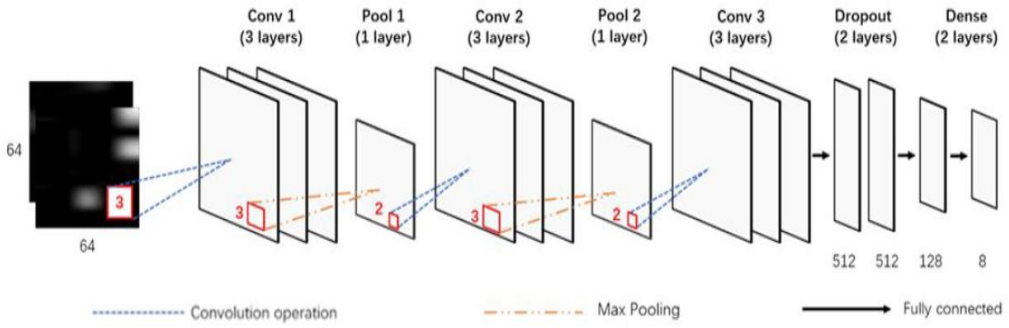

Figure 4.5: CNN network structure.

layer is also included in the first two convolutional blocks, which aims at substantially reducing the computing free-parameters. Two dropout layers [243] with 512 neurons are attached to the last convolution layer of the third convolutional block to prevent occurrence of overfitting issues. The output layer includes 8 neurons which correspond to the required predicting gesture numbers. In addition, the learning methodology that controls the learning process for the CNN is a SoftMax combined Adam gradient descent algorithm [244].

4.4 Experiment Results Comparations

For all the numerous experiments presented in this chapter, the CNN was trained for various experiments using different hyperparameters, while the training process was limited to 10 epochs. This approach was applied across a range of sensor inputs, from 1 to 50 sensors. It was observed that when the number of sensors exceeded 50, there was a decline in recognition accuracy. This decline is likely attributable to the increased level of noise present within the sensors.

The hand gesture recognition results for applied 8 hand gestures shown in figures 4.3 and 4.4 are presented in figure 4.6.

These results were obtained through intersected heatmaps which combined upper and lower bands. According to the figure, for the set of gestures considered, the recognition accuracy improved significantly with increasing numbers of the sensors, until 35 sensors are used. The best classification accuracy achieved for intersected heatmap algorithm was 98.96% when 35 sensors were considered.

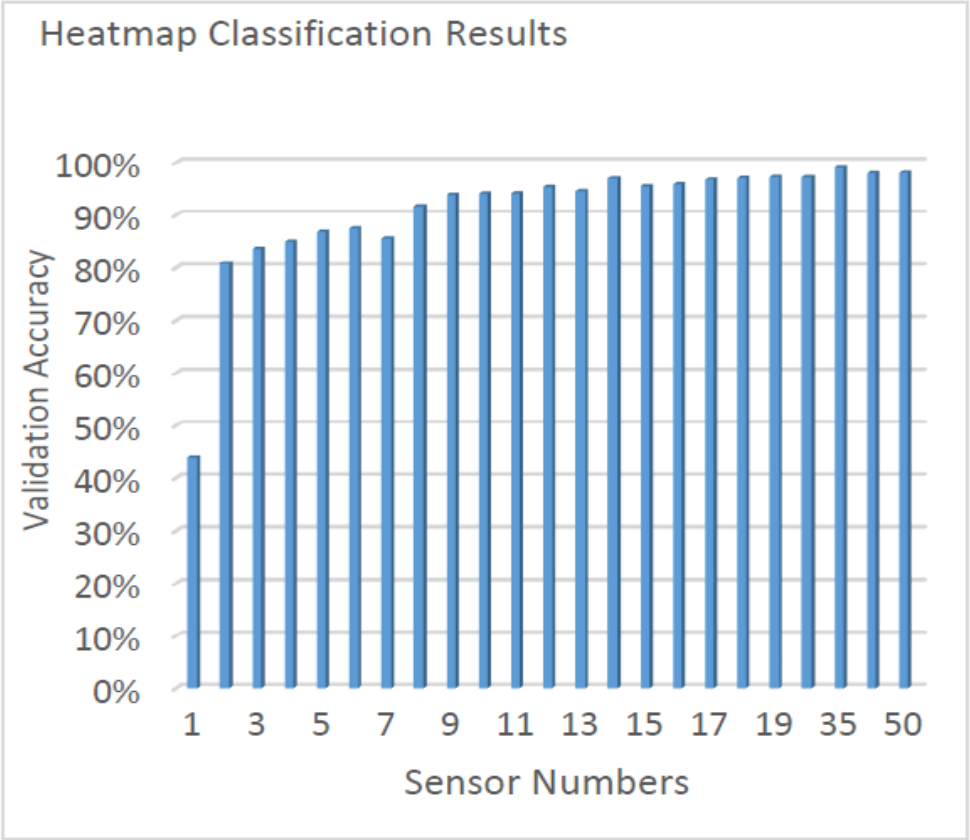

Figure 4.6: Classification results for combined two bands for 8 gestures.

The histogram in figure 4.6 indicates that beyond 35 sensors, the testing results appears to be similar. For some sensor numbers, such as 40 sensors, a slight reduction

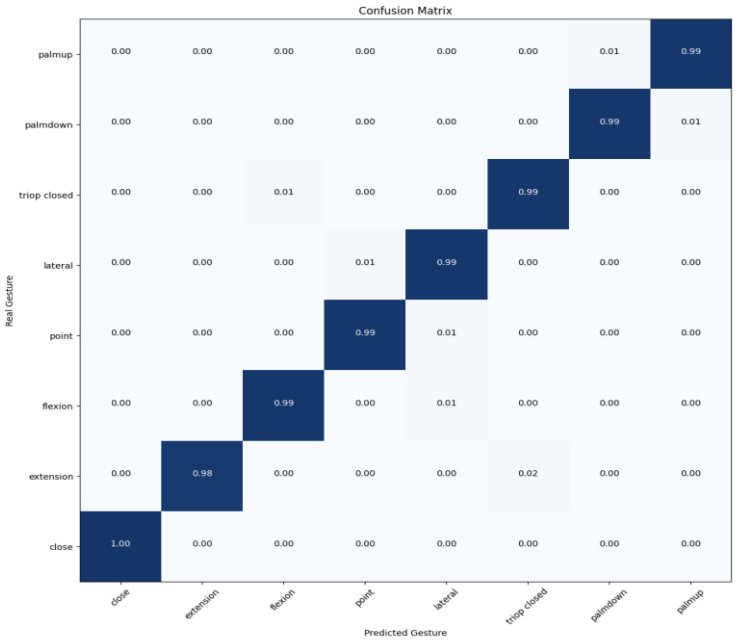

Figure 4.7: Confusion matrix for 35 applied sensors, 98.96% accuracy.

appeared in the classification rates. Hence, 35 is considered as the optimal number of sensors which can produce the best classification results in this research. Figure 4.7 demonstrate the obtained confusion matrix of the 8 gestures recognition task testing result for 35 sensors. According to figure 4.7, the most confused gesture for the CNN when doing recognition task was extension with 98.96% accuracy while the other 7 gestures achieved almost 1.00 accuracy.

Table 4.2: Common active 35 sensors for two bands

Gestures	Lower Band Common Sensor Locations	Upper Band Common Sensor Locations
palm up	48,49,50,53,54,55	103,104,105,117,118,119,125,126,127,128
palm down	24, 27, 50	107, 108, 109, 117,118,119,121,122,123,124,125,126,127,128
tripod closed	3, 4, 5, 6, 54, 55	95,96,97,105,106,107, 108,117,118,122,123,126,127,128
lateral	1, 2, 3, 4, 5, 6	67,105,106,113,114, 117,118,120,121,122,123,124,126,127
point	3, 4, 5, 6, 20, 21, 22	117,118,120,121,122,123,124,126,127,128
flexion	2, 3, 4, 20, 21, 22	91,92,93,94,95,96,97,117,118,126,127,128
extension	46,47,48,49,53,54,55,56,57	95,96,97,111,112,113, 117,118,119,125,126,127
close	3,4,5,6,7,8,27,28,29,49,50	110,111,112,113, 117,118,119,125,126,127

Table 4.3: Classification results comparison

Method	Gesture numbers	Amputee numbers	Input data type	Results
SVM ([245])	10	11	Spectrogram	77.44%
ANN ([246])	4	0	Time Frequency Features	88.40%
CNN ([247])	6	0	Wavelet Transformation	96.60%
JL ([248])	4	0	TD and other features	90.00%
CNN ([249])	7	0	Transfer Learning	97.81%
IA-CNN([250])	**6**	**14**	**Intersected Heatmap for both bands**	**98.24%**
IA-CNN([250])	**7**	**14**	**Intersected Heatmap for both bands**	**98.77%**
IA-CNN([250])	**8**	**14**	**Intersected Heatmap for both bands**	**98.96%**

Besides, the optimal sensor algorithm presented in this chapter identified the sensors which provide the best combination results for a particular gesture. This obtained optimality is used in chapter 6. In table 4.2, the common sensors for both bands for 8 hand gestures are listed.

It can be clearly seen that for most cases, only around 20 sensors are contributing significantly for all participants (both able-bodied and amputees) and many of them are contributing to more than one gesture. For example, the sensors 27 and 50 are both active for palm down and close. However, the combination of sensors for each gesture are unique. In other words, when a certain sensor combination occurs, other sensors can be ignored. Moreover, there are more contributing sensors from the upper band than the lower one.

As the research output demonstrated a nearly perfect result for the given dataset, it is interesting to do a comparison with previous works. The comparison between the intersected heatmap methodology and several methods is shown in table 4.3. The last 3 entries in the table indicate the results using the proposed approach when 6, 7 and 8 gestures were considered with corresponding classification rates of 98.24%, 98.77% and 98.96% respectively.

It should be noted that, most of the previous experimental results were achieved from different sEMG datasets. However, the equipment used to collect the data were identical in all compared cases (all collected via non-invasive electrodes, the only difference is the numbers of applied electrodes) and therefore the compared datasets are as similar as they can be. According to table 4.3, 77.44% accuracy was achieved by Xiaolong Zhai et al. [245] through SVM classification methodology on NinaPro dataset. Rezwanul et al.[246] successfully used the time frequency features of the raw sEMG signals on training an ANN and obtained 88.40% testing accuracy. Some DSP methods such as wavelet transform have also been applied on sEMG signals by Ugur Sahin et al. [247]. With 6 target hand gestures, they achieved 96.60% accuracy. Xun Chen et al. [248] tried several methods such as Hudgins' Time Domain features, autocorrelation and cross-correlation coefficients and spectral power magnitudes of EMG signals and achieved 90% real time recognition accuracy for various hand gestures collected from 6 participants. The last comparison happened between research accomplished by Côté-Allard et al.

[249], which involved a transfer learning algorithm for 17 participants and reached 97.8% accuracy for 7 aimed gestures-based recognition task.

As the dataset applied for the research in this chapter achieved as high as 98.96%, which contains 8 hand gestures from 23 able handed and amputee participants, the experimental results show that the intersected heatmap approach outperforms most of other methodologies in terms of accuracy.

In addition, considering the varying degrees of amputation among the participants, an additional experiment tailored specifically for those with diverse amputation levels was devised. The outcomes of this experiment are presented in table 4.4.

Table 4.4 Results of intersected heatmap algorithm for different amputation levels

Amputation level	Gesture Number	Subject Number	Input Data Format	Result
Able-bodied	8	9	Intersected Heatmap	99.02%
Partial hand	8	5	Intersected Heatmap	98.63%
Trans-radial	8	8	Intersected Heatmap	54.38%

Based on previous research conducted by Radhika Menon et al. [93], it was anticipated that able-bodied subjects would exhibit optimal performance due to the absence of any upper limb impairment. This was corroborated by the results in table 4.4, showcasing a classification accuracy of approximately 99.02%. Following closely were partial hand amputees, who achieved a commendable 98.63% accuracy. This is likely due to their retained intact forelimb, free from the muscle resection that occurs during surgical trans-

radial amputation or stump formation post-trauma. In contrast, trans-radial amputees experienced a significant drop in accuracy to 54.38%, which can be attributed to the loss of crucial muscles necessary for executing the specified gestures during the experiment.

4.5 Conclusion

In this chapter, a novel intersected heatmap method which designed for different participants etc. hand partial amputees, was presented for personalizing EMG-based models with deep learning techniques. The comparison between the output results for this method and some proposed prior works has indicated the advancement of the intersected heatmaps.

Furthermore, the research presented in chapters 5 and 6 are based on the intersected heatmap and common sensor algorithm presented within this chapter. The unique dual-band structure of the CapgMyo dataset [97], featuring 8 set of 16 channels, in contrast to the Strathclyde dataset's singular 64-channel band [93], proves the adaptability of intersected heatmap algorithm across diverse channel configurations. Moreover, acknowledging the potential limitations of data acquisition devices lacking 128 sensors, the intersected heatmap algorithm remains applicable, albeit with the acknowledgment that the identified common sensors may vary due to different sensor counts, potentially excluding certain sensors under specific sensor numbers.

The approach not only minimizes the input data dimensions by reducing the required number of attaching sensors without recognition accuracy deterioration, but also has the added advantage of minimizing the possibility of erroneous readings. In addition, the approach also reveals that the active sensors are not spread equally for upper band and lower band. Thus, the individual recognition accuracy for each band may differ for some gestures. Furthermore, given that there are several sensors which contribute to more than one gestures, it is reasonable to assume that judicious selection of sensor inputs can be used to minimize the number of sensors required to obtain very high classification accuracy. This finding inspired and led to the research works presented in chapter 5 and 6.

Chapter 5

Deep Convolutional Spiking Neural Network based Hand Gesture Recognition

5.1 Introduction

In this chapter, a novel hand gesture recognition technique based on the advantages of SNN as introduced in chapter 3 is presented. Because the power supply for hand prothesis is limited in reality and considering that SNNs are the most energy efficient networks nowadays, it is natural to attempt to use them in real-world applications like hand gesture recognition systems, in this case. Moreover, through applying SNNs, the required training time for the network is further reduced when compared with the research work presented in chapter 4, which used CNNs.

Due to the superior performances and advantages shown by the intersected heatmaps in chapter 4, a novel CSNN (convolutional spiking neural network) structure was created for this research to exploit these benefits. In addition, the common sensors for each gestures which were identified and presented in chapter 4 are also used in this chapter to form common heatmaps. With the help of the common heatmaps, the complexity of the network input data is greatly decreased. Thus, the essential training parameters and calculation process inside the SNNs are further improved.

The common heatmap generation procedure is presented in section 5.2. Section 5.3 indicates the novel CSNN structure. The comparation of the experimental results and some previous research works are shown in section 5.4. The conclusion and discussion of this research approach are included in section 5.5.

5.2 Common Heatmap Algorithm

The common heatmap algorithm shares a similar starting procedure as the intersected heatmap algorithm. The collected signals were first amplified and sampled at 2048Hz. Then, these sampled signals were passed through a high pass filter with cut off frequency 3Hz and a low pass filter with 500Hz cut off frequency as major EMG activity happens

between 5Hz and 450Hz. After locating the EMG activity, a 3rd order Butterworth digital filter was applied to remove electronic equipment noise and motion artefacts. The next step is to generate partial heatmaps which indicate the sensors with highest signal strength and their locations (details can be reviewed in chapter 4). However, because certain gestures cannot be performed by some amputees such as grip, only 8 gestures which can be performed by both set of participants (able bodied and amputees) were chosen for the experiment test. These 8 gestures (close; extension; flexion; point; lateral; tripod closed; palm down; palm up) are shown in figure 2.12 and 2.13 (chapter 2, section 2.5.1). The process stops when 50 (out of 64) sensors were selected for each band (100 sensors in total and the last 14 sensors were ignored as their signal strengths were quite low) [250]. The common sensor numbers for each gesture for Strathclyde sEMG dataset were identified through the common sensor locating algorithm below and is shown in table 4.2(chapter 4, section 4.4).

Due to different hand gestures applied in CapgMyo dataset, the common sensors for CapgMyo were examined through a similar process as for Strathclyde dataset (CapgMyo's collecting device also includes 128 sensors). As the acquired sEMG signals for the 8 gestures (thumb up; extension flexion; flexion extension; thumb opposing; abduction; fingers flexed; pointing index; adduction) were already band pass filtered at 20-380Hz and sampled at 1000 Hz, we only added the same 3rd order Butterworth digital filter for noise reduction.

By applying the common sensor locating technique, it is possible to determine and use for classification tasks, the sEMG values from sensors that are common to all users in each case. The obtained common sensors for each of the gestures for CapgMyo are shown in table 5.1 and the common sensor locating algorithm is presented in table 5.2.

Table 5.1: Common Active Sensors for CapgMyo Dataset

Gestures	Lower Band Common Sensor Locations	Upper Band Common Sensor Locations
Thumb up	2, 3, 4, 5	113,114, 117,118,120
Extension flexion	2, 3, 5, 46,47,48,49,53	95,96,97,111,112,113, 117,118,119
Flexion extension	2,3,4,5,46,47,48,49, 53,54,55,56	95,96,97,111,112,113, 117,118,119,125,126,
Thumb opposing	24,25,27,47,48,49,50	117,118,119,121,122, 123
Abduction	2,3,4,5,46,47,48,53, 54,55,56	95,96,97,111,112,113,117, 118,119,125,126,127
Fingers flexed	3,4,5,6,7,8,27,28,29, 49,50	110,111,112,113, 117,118,119,125,126, 127
Pointing index	3,4,5,20,21,23	117,118,120,121,122, 123,124,126,127,128
Adduction	48,49,50,53,54,55	103,104,105,117,118, 119,125,126,127,128

Table 5.2: Common Sensor Locating Algorithm for Sensors

Input: partial heatmaps for all subjects' gestures for all numbers of sensors considered
Output: common heatmaps
1. Collect all partial heatmaps for same sensor number for all subjects' gestures.
2. Select one partial heatmap for one gesture from one subject.
3. Intersect the heatmap of the selected subject's gesture from all remaining heatmaps (from all other users) for the same gesture to generate common heatmaps between the chosen subject and each of the other subjects. This generates common heatmaps between the chosen subject and each of the other subjects.
4. Repeat steps 2 to 3 for all other subjects to ensure all possible combinations of subjects are considered for the selected gesture.
5. Repeat steps 2 to 4 for all remaining gestures to generate common heatmaps for each gesture across all subjects.
6. Repeat steps 1 to 5 for all remaining sensor number selections to ensure the process is applied to all sensor configurations.

For each gesture in our self-collected dataset, 340 partial heatmaps were created (30 partial heatmaps from 9 able handed participants and 5 from 14 amputees). With 8 gestures performed by all participants, 136000 partial heatmaps were obtained (340 per-gesture× 8 gestures × 50 sensors selection conditions). With the common heatmaps algorithm for each set of the chosen sensors, 57630 common heatmaps were generated for each gesture[251]. In total, 23,052,000 common heatmaps were involved in this research (57,630 per-gesture × 8 gestures × 50 sensors election conditions). As for CapgMyo dataset, 38 partial heatmaps were obtained per-participants for each gesture

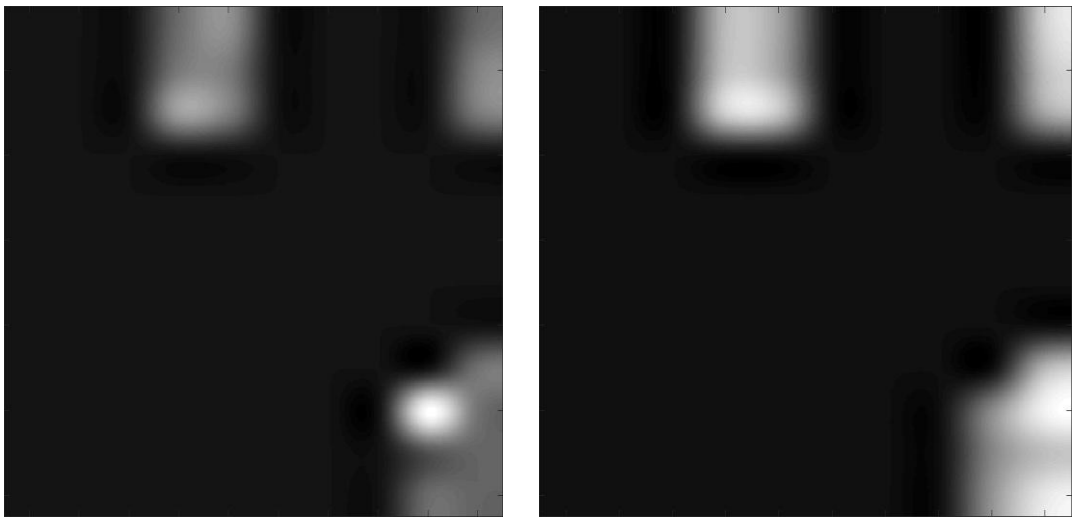

Figure 5.1: common heatmaps for participant 1 for CapgMyo.

for each set of sensors selected (from 1 to 50). In total, 15200 partial heatmaps were collected (38 per-gesture × 8 gestures × 50 sensors selection conditions). After applying the common sensor locating algorithm, 296,400 common heatmaps were obtained (741 per-gesture × 8 gestures × 50 sensors selection conditions).

Figure 5.1 indicates the examples of acquired common heatmaps for participant 1 from the CapgMyo dataset,

where the white colour indicates the location of the chosen common sensors which contain the common features, and the black colour represents the locations of the sensors whose readings were ignored after applying the common sensor location algorithm (non-common sensors). The higher the signal strength of the sensor, the brighter the colour is.

5.3 Deep Convolutional Spiking Neural Network Structure

The convolutional spiking neural network is the combination of two separate techniques: spiking convolution operation and encoding [251]. The first important part is a spike encoding which converts the pixel intensity value of the common heatmaps of the input figures to spike images/spike trains according to a threshold. The threshold was determined through various experiments and the best experiment results occurred when a threshold of 75 (ranges from 0 to 99; with 0 being total black and 99 being total white) was chosen for both datasets. For this research, the LIF encoding technique is applied to accomplish the encoding task. Once the pixel intensity value is above the threshold, a spike will be generated. With all the pixel intensities replaced by spikes, the common heatmap was turned into spike images.

The second major part of the spiking convolutional operation aims at extracting and learning the spike features that were converted by the spike encoding. The rule of the applied convolution operation is presented in equation 5.1, which dealing with spikes (0 and 1) instead of standard pixel values (0-255).

$$y[m,n] = \sum_{j=-\infty}^{\infty} \sum_{i=-\infty}^{\infty} x[i,j] \cdot h[m-i, n-j] \tag{5.1}$$

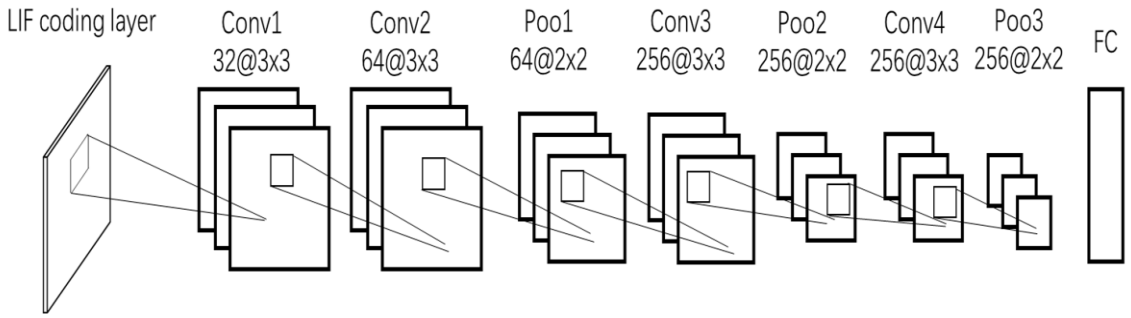

Figure 5.2: The Architecture of proposed CSNN.

where m denotes the horizontal coordinate of the selection point; n represents the vertical coordinate of the selection point; i represents the horizontal coordinates and j indicates the vertical coordinates for all pixels inside the input image [252].

The surrogate gradient method is used to train the CSNN to solve the training problem of the spiking neural network [253]. By using the surrogate gradient method to select a differentiable function with similar non-differentiable activation function as a surrogate function proxy (such as softplus) [254], the gradient, weights and bias of the CSNN can be updated during training. In addition, the trained CSNN can be deployed on neuromorphic hardware such as Loihi [25] to generate hand prosthesis in real world.

The whole structure of the CSNN is shown in figure 5.2. Similar to the normal CNN (inspired by the network in chapter 4), the network contains 4 convolutional layers which process spikes (second layer, third layer, fifth layer and seventh layer). There are 32 kernels (size 3×3) inside the second layer, 64 kernels (size 3×3) applied to the third layer, 128 kernels (size 3×3) in the fifth layer and 256 kernels (size 3×3) in the seventh layer. The first layer of the CSNN is formed by a group of LIF neurons to perform the encoding operation. The fourth layer is an average pooling layer with pooling size 2, with a stride of 2. For the sixth layer, the average pooling operation is done again with pooling size 2, and a stride of 2. The eighth layer is another pooling layer which has the same parameters as the previous ones. The last layer of the network, nineth layer, is a dense fully connected layer which produces the classification results. The rationale behind designing this network structure is similar to that outlined in chapter 4, aimed at ensuring manageability and minimizing power consumption.

5.4 Experimental Results

Various experimental results were obtained for both datasets after training the CSNN with ANN conversion. By comparing these results with prior research works, the superior qualities such as the improved testing accuracy of the common heatmap algorithm were verified.

Figure 5.3 shows three test results for CSNN of our collected sEMG dataset after applying the common heatmap algorithm. Figure 5.4 indicates the confusion matrix of classification results of Strathclyde dataset.

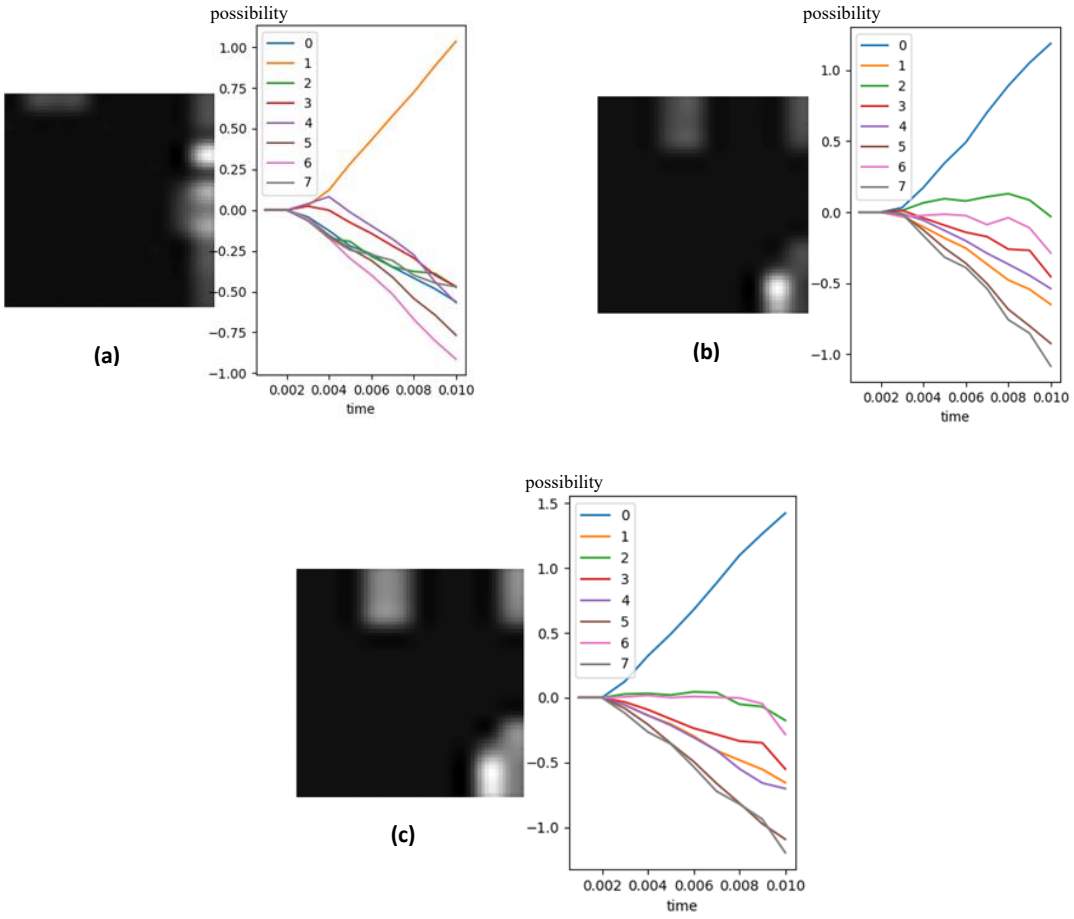

Figure 5.3: CSNN classification results for Strathclyde dataset.

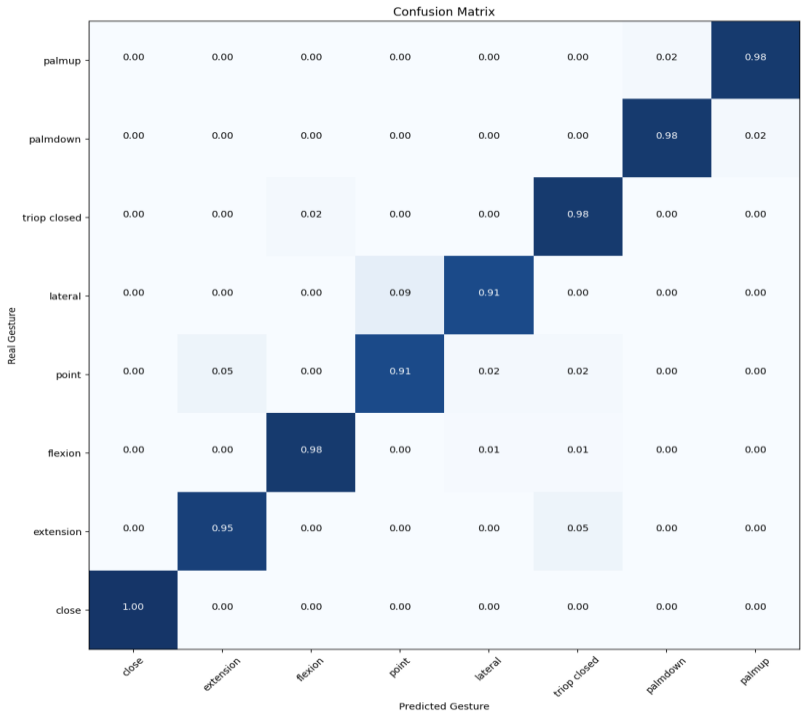

Figure 5.4: Confusion matrix for Strathclyde dataset via CSNN.

where the figure shows the input common heatmap and the ledger and response diagram indicate the recognition result. Each number in the ledger denotes a pre-defined hand gesture (0: palm up; 2: palm down; 3: tripod closed; 4: lateral; 5: point; 6: flexion; 7: extension; 8: close) represented by specified colour. The vertical axis of the line charts indicates the possibility (in percentage) of the gesture the input common heatmap stands for, and the horizontal axis of the line charts represents the processing time (in seconds).

Clearly, as shown in figure 5.3, the possibility for each gesture always starts with 0 when the common heatmap is fed into the CSNN. The time displayed on the x-axis, measured in seconds, illustrates how the classification results change over time as the amount of input data increases. After approximately 0.002s, the correct line which indicates the actual gesture that the input common heatmap represents starts to rise while other gesture lines are falling. This tendency continues until the testing time ends, at around 0.01s, where short analysis windows of approximately 10 ms have been shown sufficient for real-time sEMG classification according to previous research [255] [256]. as according to x axis shown in figure 5.3. Eventually, the line which contains the highest possibility would be selected by the CSNN as the actual recognition result for this whole set of gestures test. Additionally, the decision-making process of the CSNN is also

demonstrated in figure 5.3. For example, in figure (a), lines for gestures 1, 3 and 4 all go up from 0.002s to 0.004s, which means initially the CSNN predicts the input common heatmap represents multiple gestures. This phenomenon happens as some common sensors exist in multiple gestures (table 4.1 and 5.1). However, through the CSNN's further 'processing', the line representing gesture 1 keeps rising while the other two lines start to decline, which indicates the CSNN believes that the input heatmap belongs to gesture 1. In addition, figure 5.3 also displays some interference which cannot be removed even after applying common heatmap algorithm. For example, the common heatmaps obtained from the same performed gesture (palm up, shown in (b) and (c) in figure 5.4), they can appear to be slightly different in some sensors' energy intensities (colours). This is mainly because the muscle strength for each participant can be different. However, despite the interference, the CSNN achieves 98.76% average testing accuracy for our self-collected dataset after applying the common heatmap algorithm.

The recognition result examples for the CapgMyo dataset are presented in figure 5.5 and figure 5.6.

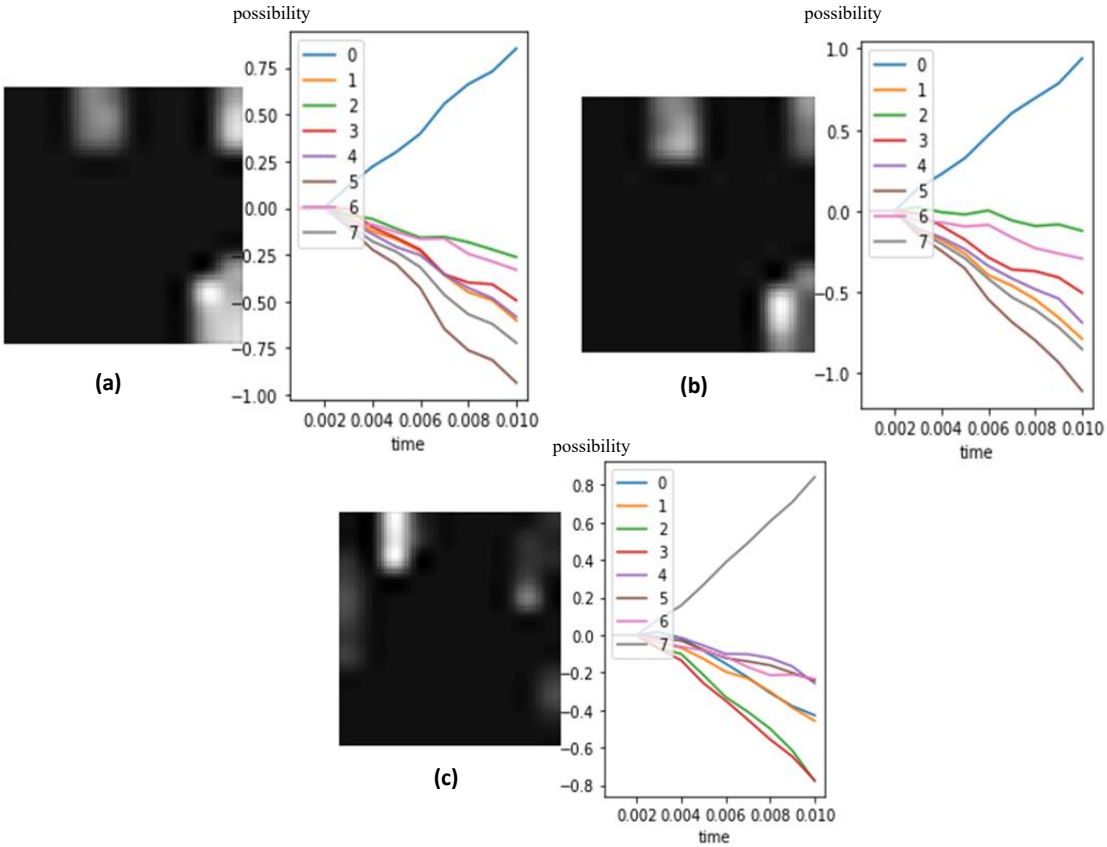

Figure 5.5: CSNN classification results for CapgMyo dataset.

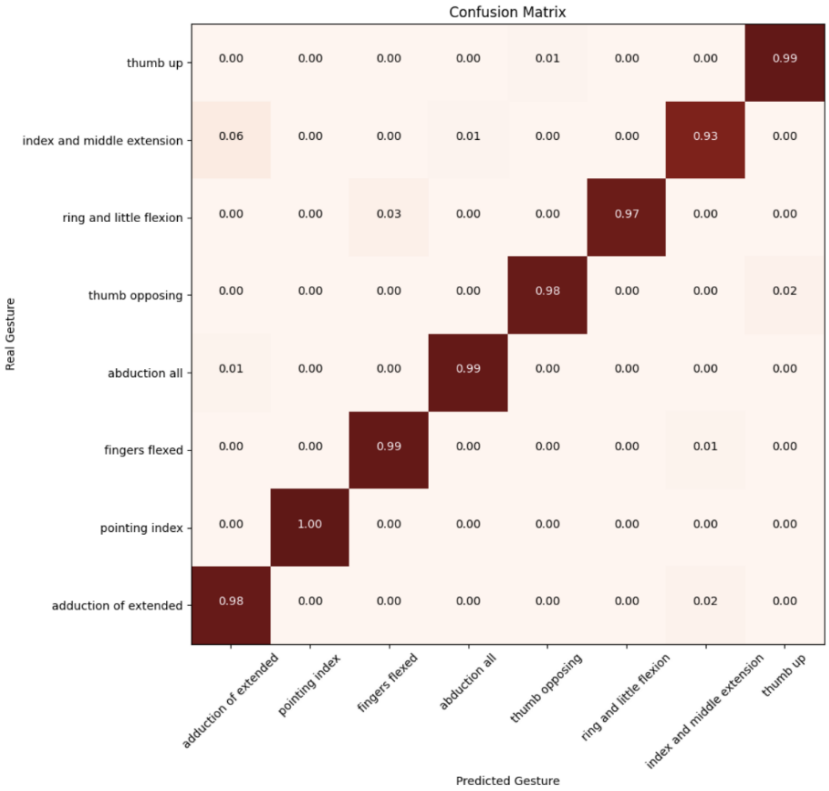

Figure 5.6: Confusion matrix for CapgMyo dataset via CSNN.

Again, the time displayed on the x-axis also illustrates how the classification results change over time as the amount of input data increases.

Where the figure still represents the input common heatmap and the ledger and response diagram show the recognition result. The numbers and colours still have the same meaning as figure 5.4 for 8 selected hand gestures from CapgMyo (0: thumb up; 1: extension flexion; 2: flexion extension; 3: thumb opposing; 4: abduction; 5: fingers flexed; 6: pointing index; 7: adduction). The vertical axis and the horizontal axis refer to the possibility of prediction (in percentage) and the processing time (in second).

Shown in figure 5.5, the CSNN starts the recognition process at around 0.0025s which is a bit slower than the previous experiment based on our self-collected dataset. Due to the muscle strength differences, the recorded signals for the CapgMyo are weaker compared to the signals from our self-collected dataset (the colour white is not as bright as our dataset). In other words, the features for CapgMyo's common heatmaps are less prominent compared to the common heatmaps generated based on our self-collected dataset. Thus, a longer recognition time appears in figure 5.5 as the CSNN requires more time to extract and recognize the information. However, the prediction process still ends

at 0.01s, which remains unchanged for both datasets. Like the previous test, the lines for the same gesture thumb up (represented by (a) and (b) in figure 5.5) are in different forms due to the small differences of the pixel intensity which exist in the common heatmaps. Eventually, the CSNN achieves an average 98.21% testing accuracy. Though it is marginally lower compared to the collected dataset, it still shows the efficiency and advantage of the common heatmap algorithm even when applied to different sEMG datasets.

The disparities in processing time for the two datasets can be attributed to variations in the sEMG signals. Since the gestures and participants contained in the two datasets differ, the resulting spikes inputted into the SNN also vary. Therefore, despite employing the same SNN structure for both experiments, the processing time required differs.

Once the test results were completed, they were first compared to our previous results obtained using the IACNN (intersected heatmap based convolutional neural network) methodology introduced in section 4.4. Table 5.3 provides the comparison details.

Table 5.3: Classification Results Comparison

Method	**Gesture Number**	**Subject Number**	**Dataset**	**Result**
IA-CNN([250])	8	23	Strathclyde	98.96%
IA-CNN([250])	8	10	CapgMyo	97.59%
CSNN([251]**)**	**8**	**23**	**Strathclyde**	**98.76%**
CSNN([251]**)**	**8**	**10**	**CapgMyo**	**98.21%**

As shown in table 5.3, the previous IACNN method achieved 98.96% accuracy for Strathclyde dataset. By contrast, for the CSNN, 98.76% accuracy was achieved for 5 times experiments. Hence, the common heatmap algorithm and CSNN produced a comparable performance (only 0.2% accuracy loss) to the intersected heatmaps algorithm and IACNN but using only just one sixth of the training of the latter approach. Additionally, the presence of the tiny accuracy loss is mainly because of the conversion process. As the threshold for the CSNN was essential when transferring normal figures into spiking images, any lower value features would be filtered out. Thus, the information contained inside these features cannot be absorbed by the network. With the improvement of the transfer methods, these kinds of losses can be further minimized.

Moreover, also shown inside the table, the recognition result achieved by CSNN on the CapgMyo dataset reached over 98% accuracy, which is higher than the IACNN's. In conclusion, the CSNN methodology is a superior and can be a more widely applicable technique for hand gesture recognition as it outperforms IACNN on both datasets which, in total, include 16 gestures.

The test results comparation between our research work presented in this chapter and some previous achievements accomplished by other researchers are listed in table 5.4.

Based on Table 5.4, the CSNN exhibits the third-highest testing accuracy, registering at 98.21%, among the six presented hand gesture recognition methods. Notably, the peak accuracy of 98.9% is achieved by Jiangcheng Chen et al. [257] utilizing 3DCNNs, while a close second of 98.7% is also secured by Jiangcheng Chen et al. [257] with 2DCNNs. Despite these two exceeding the CSNN's performance, their research focuses on a smaller set of objects, potentially contributing to their superior results due to reduced complexity.

Table 5.4: Classification Results Comparison for Multiple Methodologies

Method	Gesture Number	Subject Number	Input Data Format	Result
2DCNN([257])	8	6	sequential sEMG images	98.7%
3DCNN([257])	8	6	sequential sEMG images	98.9%
Quantization-based position Weight Matrix (QuPWM) based SVM ([258])	8	9	Quantization-based position Weight Matrix features	75%
CNN with spatial attention modules ([259])	8	10	sEMG images	96.06%
CSNN([251])	**8**	**10**	**common ED map**	**98.21%**
SVM, LDA, KNN and BP([260])	8	10	HD sEMG maps created by decoded signals	94.21%±4.84%

Yurong Li et al. [260] follow closely behind with an accuracy of 94.21%±4.84%, achieved through various classifiers like SVM, LDA, KNN, and BP. In addition, the method devised by Abderrazak Chahid et al. [258] applying a Quantization-based

Position Weight Matrix, demonstrates a significantly lower accuracy of 75%, compared to the 96.06% achieved by S. Hao et al. [259]. using CNNs with spatial attention modules. This disparity highlights the varying effectiveness of different classification approaches, as SVMs and CNNs employ distinct training strategies. It is plausible that the observed performance differences stem not only from the input features but also from the choice of classifier.

Importantly, all the reported results in Table 5.4 are based on evaluations conducted on the CapgMyo dataset.

5.5 Conclusion

In this chapter, a novel common heatmap (targeting the enhancement of hand gesture recognition performance for amputees by minimizing the number of sensors) based algorithm for personalizing sEMG based models with CSNN has been presented. Two high density sEMG datasets (Strathclyde dataset and CapgMyo dataset) were applied to evaluate the actual performance of the technique. The obtained testing accuracy is compared with proposed prior works that were based on different sEMG signal processing methodologies such as sequential sEMG images, Quantization-based position Weight Matrix features and self-designed auto encoder with various machine learning and deep learning techniques. The compared results indicate that the core advantage of the common heatmap based CSNN approach not only reflects the minimisation of network training time and dataset processing time, but also depicts the high accuracy of this hand gesture recognition approach. According to section 5.4, the experiment results denote that the common heatmap based CSNN approach grants faster user execution than other approaches without recognition accuracy loss, which is important when building real time hand gesture recognition systems (hand prosthesis). Additionally, the CSNN achievement presented in this chapter is based on the magnitude of the sEMG signals (heatmaps indicates the energy density which is the magnitude of the signals). In this chapter, we only looked at magnitude of the signals. However, as the previous works reviewed in this chapter prove that various features of the sEMG signals can be used for gesture recognition tasks, our work has also been extended into frequency features of the sEMG signals, which is presented in chapter 6.

Chapter 6

Deep Spiking Neural Network and Frequency Density Map based Hand Gesture Recognition

6.1 Introduction

This chapter presents the novel design and experiment process of a CSNN based hand gesture recognition system. The SSA technique is utilized to eliminate noise from the raw sEMG signal, as it effectively isolates the signal's primary component from the noise [98]. By reconstructing the signal using only its main component, the impact of noise can be further minimized. The SNN model was trained by ANN conversion algorithm (described in section 3.8.4) with a novel CSNN architecture. The frequency density maps (FDM) method (in section 6.2) is validated as one of the best input features for the proposed CSNN structure. The CapgMyo open-source dataset and our collected sEMG dataset were used for validating the proposed CSNN based hand gesture recognition technique. According to the experimental results, the proposed technique achieves improvements for both training speed and state-of-the-art hand gesture recognition accuracy compared to previous work.

Section 6.2 introduces the applied sEMG dataset with the essential sEMG processing. The detailed SSA procedure and FDM method are also presented in section 6.2 with examples. Section 6.3 indicates the proposed CSNN architecture, with the recognition accuracy comparation between our experiment results and some of the related previous research works. Section 6.4 includes the conclusion and discussion of the state-of-art CSNN based hand gesture recognition technique.

6.2 Frequency Density Map (FDM) Generation

With the cleaned sEMG signals and pre-identified common sensors shown in sections 4.3 and 5.3, the frequency density map of each of the hand gestures for both datasets can be obtained.

First, the fast Fourier transform was applied to convert the sEMG signals from time domain to frequency domain together with the contained features. Then, the mean absolute value for each common channel selected from all 128 channels for a chosen hand gesture in the form of sEMG signals was calculated and formed into a graph with 128 pixels. Each pixel represents one sensor channel. The common sensors' pre-calculated point values were then placed inside the graph according to the original location of their represented common sensors. For the rest of the points inside the graph, their point values were set to zeros. Each frequency map was resized before feeding it into the neural network. Some examples of the created frequency maps for our self-collected sEMG signals are shown in figure 6.1. In total, 21760 frequency maps were obtained from our self-collected sEMG dataset, and 140 frequency maps were generated from the CapgMyo dataset.

6.3 CSNN Architecture, Experimental Results and Comparation Comparison

A designed convolutional spiking neural network was applied in this research to test the novel sEMG signals' frequency-based hand gesture recognition approach. The applied CSNN shares a similar structure with the architecture introduced in section 5.3, with the same LIF coding neurons threshold settings and the same number of convolutional layer and pooling layer.

The CSNN consists of six layers. The first layer, the input layer, is formed by a group of LIF neurons which converts input frequency maps into spiking images (each input

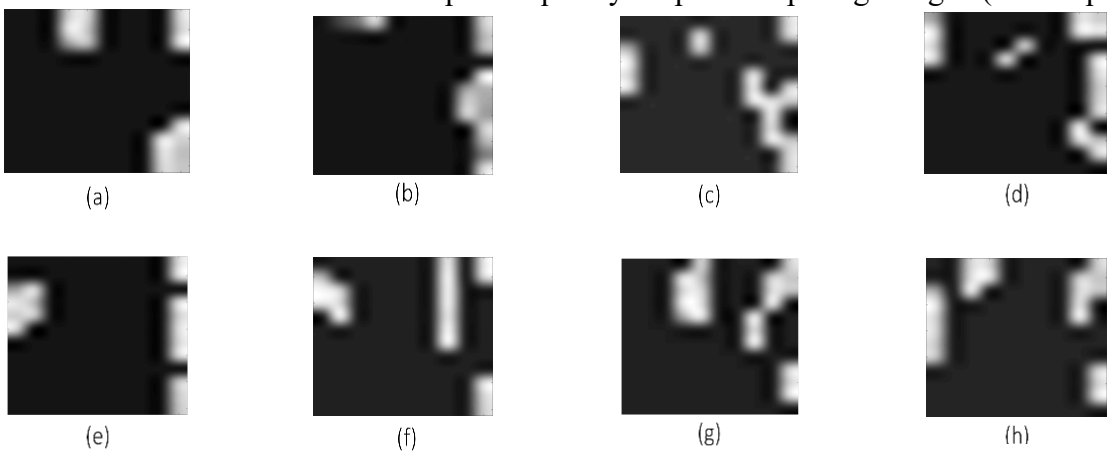

Figure 6.1: Frequency density maps for gestures: (a) palm up; (b) palm down; (c) tripod closed; (d) lateral; (e) point; (f) flexion; (g) extension; (h) close.

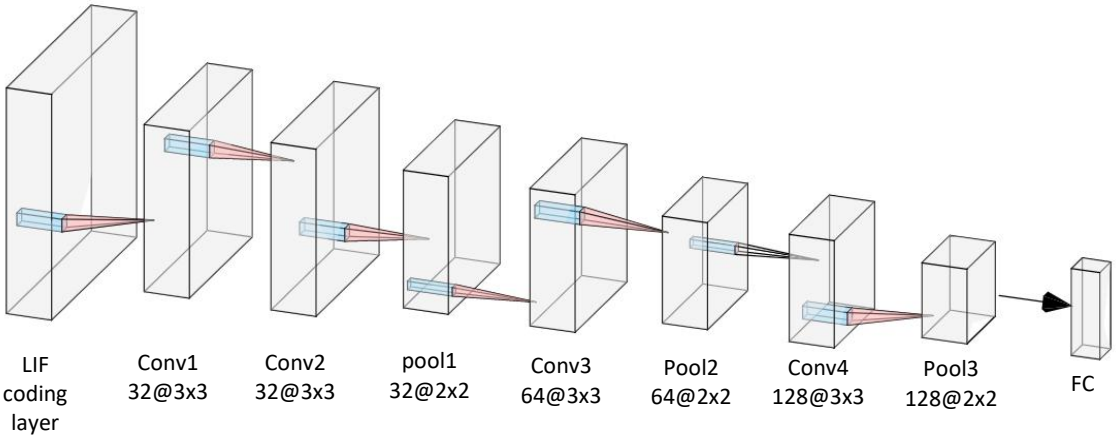

Figure 6.2: Convolutional spiking neural network structure.

pixel blocks are represented by spikes). The second layer is a convolutional layer applied to perform the first convolutional operation with 32 kernels (size 3*3). The third, fourth and fifth layers are all convolutional layers followed by a pooling layer. The only difference between these three layers is that the third convolutional layer has 32 kernels (size 3*3), the fourth convolutional layer contains 64 kernels (size 3*3), and the fifth convolutional layer has 128 kernels (size 3*3). All the pooling layers share a similar construction (size 2, strides 2). The last layer of the CSNN is a fully connected layer (dense layer) which classifies the input hand gestures' spike images. Figure 6.2 indicates the full CSNN structure (from left to right: input LIF coding layer, first convolutional layer, second convolutional and first pooling layer, third convolutional and second pooling layer, fourth convolutional and third pooling layer, dense layer).

6.3.1 Experimental Results and Comparison

Two experimental processes which comprise similar testing approaches were designed and executed on both Strathclyde dataset and the CapgMyo dataset as both datasets were obtained through similar methodologies and equipment.

Each of the two datasets was split into a training part (contains 80% of the images for training the network) and a testing part (includes the rest 20% of the images for testing the network performance once the training process is completed).

6.3.2 Experimental Results for Strathclyde Dataset

The SSA and FDM methods were first applied on Strathclyde sEMG dataset. The training process lasts approximately 20 minutes in average.

Figures 6.3 and 6.4 indicate typical testing recognition results after the training. As shown in figure 6.3, the grayscale figure was the test image randomly chosen from the testing part (for this example, it is the gesture extension). The ledger and response diagram show the network recognition result. The numbers attached on the left side of the result image denotes the possibility of the gesture to which this test image could belong (in percentage). Each of the colour lines, with its corresponding number, represents a selected hand gesture from the dataset (blue line, No.0 represents gesture palm up; orange line, No.1 indicates gesture palm down; green line, No.2 represents gesture tripod closed; red line No.3 indicates gesture lateral; purple line No.4 represents gesture point; brown line No.5 indicates gesture flexion; pink line No.6 represents gesture extension and grey line No.7 indicates gesture close).

The simulation time lasted 0.01s as shown in figure 6.3. At around 0.002s the CSNN started to classify the gestures as all the lines began to rise or fall. Until the end of the simulation, the pink line reached around 100% while all the other lines went down which indicates the network recognized that this test image belongs to the gesture extension.

In figure 6.4, the input image represented gesture palm up, and the whole simulation time also lasted 0.01s. Comparing with the previous example, the network started to

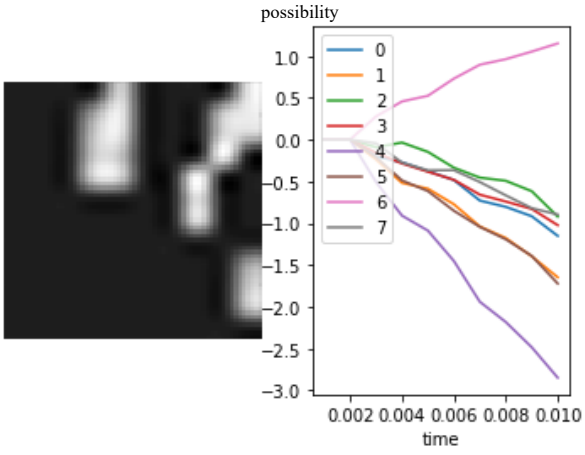

Figure 6.3: Convolutional spiking neural network classification result for the gesture extension.

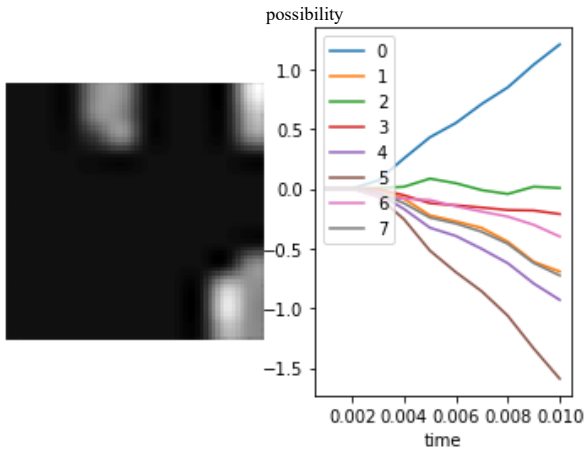

Figure 6.4: Convolutional spiking neural network classification result for the gesture palm up.

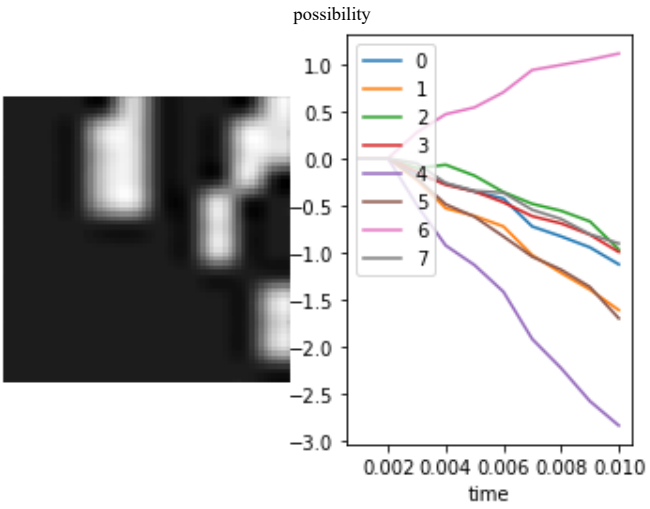

Figure 6.5: Convolutional spiking neural network classification result for gesture extension.

recognize the test image at around 0.0022s. Moreover, unlike the previous example, some lines appeared to rise initially, for example, green line No.2. These two-phenomenon showed that the network had some difficulty to decide this image's category i.e., the input test image caused the network to become confused in classifying this class. However, as the simulation progressed, the blue line eventually achieved 100% output while the other lines went down.

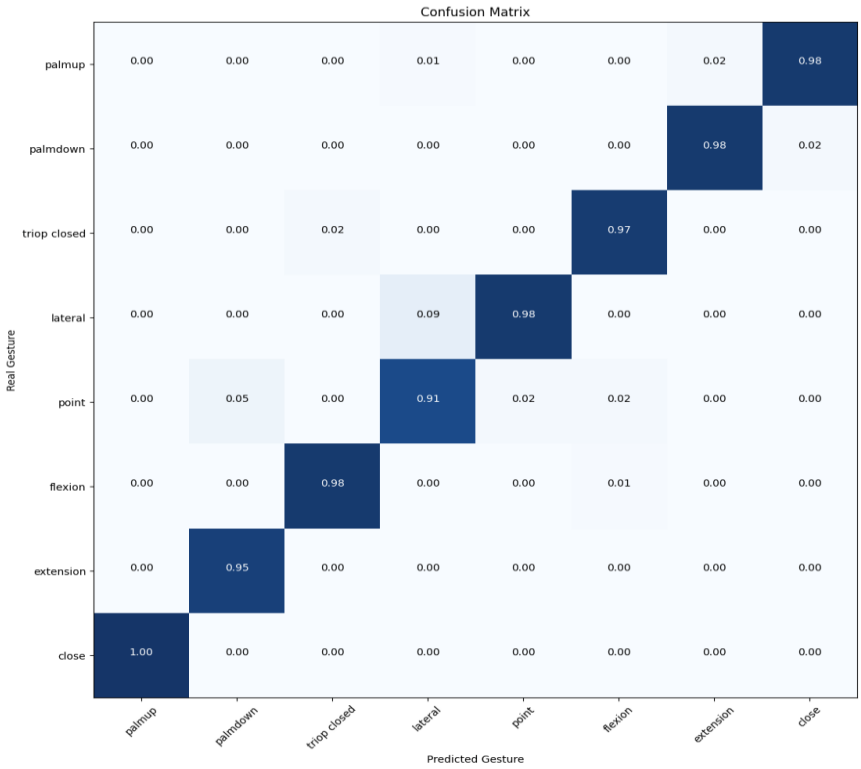

Figure 6.6: Confusion matrix for Strathclyde dataset via SSA and CSNN.

The image shown in figure 6.5 was another test image selected, and which represents the gesture extension. Though the curves of each line are different compared with the example shown in figure 6.3, the network still made the prediction that this image represents the gesture extension.

In Strathclyde dataset, the spiking convolutional neural network reached 97.56% testing accuracy after 10 epochs of training process. The confusion matrix is shown in figure 6.6.

6.3.3 Experimental Results for CapgMyo Dataset

After finishing the experiments on our self-collected sEMG dataset, the CapgMyo dataset was introduced to the frequency map approach. Figure 6.7 and figure 6.8 present examples of some of the test results.

In figure 6.7, the grayscale image also indicates the input test image (this time gesture fingers flexed) while the numbers shown on the ledger and the response diagram still representing the possibility range. The colour lines with numbers now represent a different set of 8 hand gestures (blue line, {0}, represents gesture thumb up; orange line, {1}, indicates gesture extension of index and middle, flexion of the others; green line, {2}, represents gesture flexion of ring and little finger, extension of the others; red line, {3}, indicates gesture thumb opposing; purple line, {4}, represents gesture abduction of all fingers; brown line, {5}, indicates gesture fingers flexed; pink line, {6}, represents

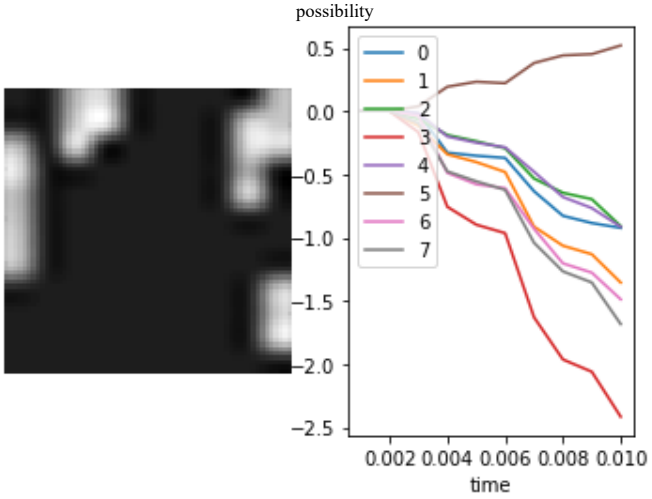

Figure 6.7: Convolutional spiking neural network classification result for gesture fingers flexed.

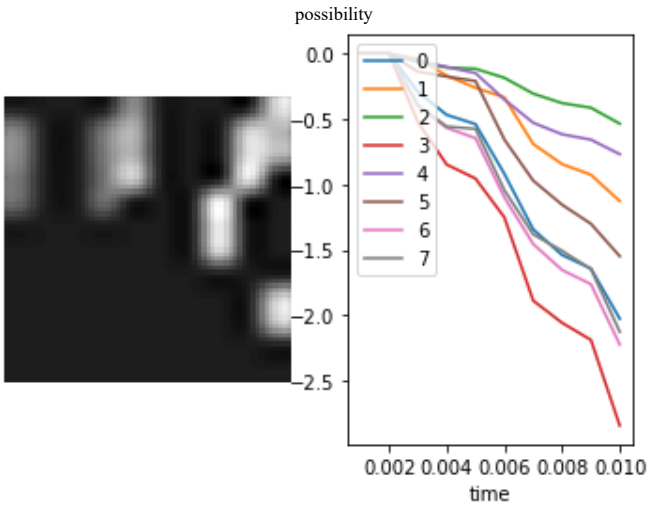

Figure 6.8: Convolutional spiking neural network classification result for gesture abduction.

gesture pointing index and grey line, {7}, indicates gesture adduction of extended fingers).

As it is clearly shown in figure 6.7, the simulation time setting was also 0.01s. However, the network classification process started at around 0.0018s which a bit earlier compared with the recognition process for the self-collected dataset. This faster reaction of the network indicates that the applied 8 hand gestures' features from the CapgMyo dataset are easier to detect through the common sensors when compared to the previous dataset. The reason for this to happen might be the common sensors defined for CapgMyo contains more significant values than for our self-collected dataset. Thus, the network classifies them quicker. Furthermore, the brown line which is the right prediction from the network just reached the 50% possibility point when the simulation process stopped. The main reason for this is the limited number of spike images the training dataset, only 112 frequency maps. The small amount of the training images also caused the network to appear confused when facing some classification tasks. For example, presented in figure 6.8, all the lines dropped by the end of the simulation while the input test image represents the gesture abduction. The two related lines (green line and purple line) mean that the network found it hard to distinguish between gestures flexion extension and abduction for this test image. However, after 5 epochs training, the recognition accuracy for the CapgMyo dataset still achieved 93.85% accuracy based on our novel frequency map methodology.

Again, a confusion matrix is also generated, as shown in figure 6.9.

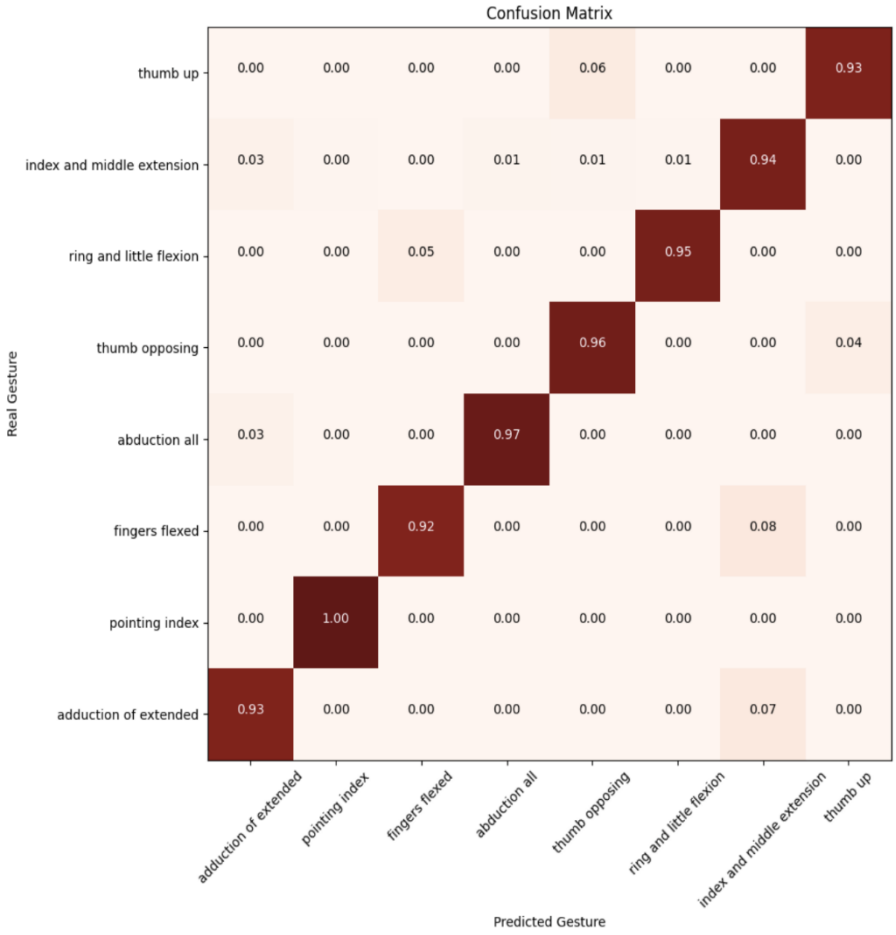

Figure 6.9: Confusion matrix for CapgMyo dataset via SSA and CSNN.

6.3.4 Comparison and Discussion

After finishing the experiments for both datasets, some previous research outputs were chosen for efficiency comparison of the new approach. A self-comparison was also included. The compared results are listed in table 6.1 and table 6.2.

Table 6.1 presents the self-comparison results. With SSA applied, the testing accuracy for both datasets increased about 2% which indicates the SSA method has the ability for further extracting features on frequency maps. Furthermore, the enhanced accuracy achieved through SSA, as compared to methods that rely on filtering and FFT, demonstrates its ability to precisely identify and eliminate noise while retaining valuable information during sEMG signal processing. When comparing with the previous common energy density based heatmap methodology, the new frequency approach shows around 1% accuracy decreasing on Strathclyde dataset and about 4% accuracy

drop on CapgMyo dataset. However, only 10 training epochs are required for the frequency maps approach while 15 required for the heatmap methodology (based on self-collected sEMG dataset) [250].

Table 6.1: Testing results comparation between our own research works

Dataset	Pre-processing	Methodology	Classifier	Testing accuracy
Strathclyde	**Filtering, SSA and FFT**	**FDM**	**CSNN**	**97.56%**
CapgMyo	**Filtering, SSA and FFT**	**FDM**	**CSNN**	**93.85%**
Strathclyde	Filtering and FFT	FDM	CSNN	95.38%
CapgMyo	Filtering and FFT	FDM	CSNN	91.00%
Strathclyde	Filtering	Common heatmaps	CSNN	98.76%
CapgMyo	Filtering	Common heatmaps	CSNN	98.21%

These testing losses might be caused by the feature differences of the sEMG signals between the frequency domain and time domain i.e., after conversion, some of the

apparent time domain features no longer exist while some tiny time domain interference becomes more noticeable for the network. Also, as a pre-set threshold is required when transfer normal images into spike images (mention in section 5.3), any lower value features would be ignored. Thus, the convolutional spiking neural network neglected part of the features. Because of these inevitable feature distortions, the LIF input layer can be further carefully adjusted to minimize their impact.

The comparison of our frequency map approach with previous studies, including those utilizing transfer learning and multilinear singular value decomposition (MLSVD), is summarized in Table 6.2. Our frequency map method achieves the second-highest accuracy of 93.85% on the CapgMyo dataset, surpassed only by the 94.7 ± 3% accuracy attained by Zhipeng Yu et al. [261] through a transfer learning-based convolutional neural network (CNN) methodology, which benefited from a major voted post-processing method during training. Kang Wang et al. [262] occupy the third spot with 86.62 ± 5.68% accuracy, achieved using a Multi-Source Integration based transfer learning method. Zhipeng Yu et al. [261] also rank fourth with 78.7 ± 5% accuracy, through a different transfer learning approach. The discrepancy between these two-transfer learning-based results arises from variations in training strategies. Sibasankar Padhy et al. [263] applying MLSVD, achieve the fifth-highest recognition accuracy of approximately 77.6%. MLSVD, akin to singular spectrum analysis (SSA), seeks to decompose high-order signals into lower-order components, facilitating efficient and meaningful analysis. While MLSVD shares similarities with our approach, the results in Table 6.2 demonstrate that our proposed frequency density map (FDM) methodologies outperform MLSVD. This performance gap could also be influenced by the classifiers used, as we applied a convolutional spiking neural network (CSNN) while Sibasankar Padhy et al. utilized support vector machines (SVM).

Generally, the combination of proposed FDM methodology and CSNN overcomes most of the state of art methodologies. In addition, all the results shown in table 6.2 are obtained via CapgMyo dataset.

Table 6.2: Testing results comparation between previous hand gestures recognition methodologies

Methodology	Gesture number	Participant number	Results
FDM	**8**	**10**	**93.85%**
Multilinear singular value decomposition ([263])	8	10	77.6%
Transfer learning ([261])	3	10	78.7 ± 5%
Transfer learning ([261])	3	10	94.7 ± 3%
CNN with spatial attention modules ([259])	8	10	96.06%
Multi-Source Integration based Transfer Learning ([262])	8	10	86.62 ± 5.68%

6.4 Conclusion

Using sEMG signals for hand gesture recognition has been possible for many years. However, problems like time-delayed reactions and wrongly evaluated response had prevented further development of this technology. In this thesis, we have presented a novel sEMG frequency map model combined convolutional spiking neural network methodology to address such issues.

The method has been evaluated on both our high-density Strathclyde dataset and the CapgMyo open-source dataset. Acquired hand gesture recognition performance has been compared with several proposed previous methods which use various machine learning and deep learning approaches. According to the comparison results, our new approach has been shown to be highly effective and leads to similar or even better results of over 93% with very significant reduction in pre-processing and network training time. In total, the generation time of the frequency maps of all 23 subjects took 5 minutes (about 13 seconds for each subject) and the network training process required another 5 minutes. Therefore, a potential use for real time applications could be achievable through the frequency map approach. Additionally, in real-time prosthesis control, faster processing is always advantageous, so even small reductions in computation time become significant [239][264][265]. Moreover, the use of the common sensors proves that high quality hand gesture recognition can be achieved through a limited number of sensors. Thus, the processing time of the real time operation system can be further minimized utilizing only commonly used sensors.

In addition, the implementation of the SSA has demonstrated great potential in sEMG frequency domain feature extraction as it generated better results for the same dataset when applying on our convolutional spiking neuron network.

The optimal algorithm choice for a practical application typically hinges on a range of factors, including hardware capabilities and the desired balance between precision and computational efficiency. Spiking Neural Networks (SNNs) offer distinct advantages in modelling biological neural processes, particularly when dealing with signals like sEMG. Because SNNs employ spikes rather than floating-point numbers for information storage, they often demand significantly fewer computational operations and less memory

compared to other neural network types. One potential drawback of using SNNs lies in the pre-processing step, involving the conversion of float values into spikes.

However, as demonstrated in sections 6.3.2 and 6.3.3, our novel sEMG frequency map model effectively mitigates this concern. It notably reduces processing time while maintaining nearly identical classification results. Consequently, our methodology presented in this chapter holds the promise of enhancing current real-time hand prostheses.

Moreover, an enhanced version of the SSA technique is available and detailed in Appendix I. However, owing to time constraints, this upgraded SSA technique remained unused in the scope of this research. The paper in Appendix I suggests that the improved SSA method features a considerably simplified calculation process. Consequently, there is the potential for further reduction in processing time if the improved SSA method is applied, all without compromising accuracy. As results in this chapter demonstrate that the SSA method effectively reduces processing time without any compromises in accuracy.

Chapter 7

Conclusion and Future Work

7.1 Conclusion

This thesis presents several novel deep learning and signal processing algorithms which aim at addressing the sEMG based hand gesture recognition tasks for prosthetic system by using signals for both able bodied people and amputees. All the novel approaches were based on two different sEMG datasets, Strathclyde dataset and CapgMyo dataset. The performance of each dataset was evaluated and compared with some existing hand gesture recognition methodologies. The main focusing of this thesis is accomplishing procedures to surpass difficulties on hand gesture recognition through different deep learning and signal processing techniques. Additionally, the common active sEMG sensors for a certain gesture for the two types of participants are also identified in this thesis. These different techniques include the heatmap algorithm that transforms the raw sEMG signals into heatmaps and a VGG16 based proposed CNN structure. The common heatmap approach which reduces the required sEMG channels while retains the testing accuracy almost unchanged and a novel CSNN architecture. The FDM (frequency density map) technique that obtains sEMG signal features from frequency domain and a novel CSNN architecture.

Chapter 2 of this thesis reviews the fundamentals of the sEMG technique, the reason why this technology was invented (section 2.2.1). These fundamentals contain methods which are designed to detect the sEMG signal (section 2.2.2), the methodologies that are commonly applied for sEMG signal processing (section 2.2.3) and several previous research works which used sEMG for the hand gesture recognition (section 2.2.4). In addition, the introduction of the development history of the hand prosthetics are also included in this chapter (section 2.3), from the very simple designs to some up to data versions. Moreover, the hybrid technology, which combines sEMG techniques and hand prosthetics are described (section 2.4). Through these reviews, the understanding of basic concepts, together with primary collecting and processing methods of the sEMG signals are improved. Also, the understanding of the ways how the hand prosthetics operate, and their structures is deeply increased after this chapter. The applied dataset, Strathclyde

sEMG dataset for 13 amputees and 9 able bodied participants with the 128-channel acquisition device (section 2.5.1) and CapgMyo dataset for 23 able-bodied participants are also presented (section 2.5.2). After chapter 2, additionally, the drawbacks of the conventional sEMG based hand prosthetics such as defective classification results and unavoidable time delayed response were fully outlined, the importance of creating an AI prosthetic is clearly evident.

Chapter 3 provides another literature review about current popular deep learning techniques and artificial neural networks. Nowadays, researchers usually separate neural networks to three generations, the very first generation, ANN, is initially introduced. This part of the review helps to understand the components and operations that constitute the whole ANN network, such as neurons (section 3.2.1), multi-layer perceptron (section 3.2.2), activation functions (section 3.2.3), loss function optimization (section 3.2.4) and back propagation (section 3.2.5). Though these components were designed when the AI concept was still at an early stage, they are still essential to all network generations. Then, each branch of the second-generation networks is presented separately. For the CNNs, its unique layers, convolutional layer and pooling layer (section 3.3.1 and section 3.3.2), together with the commonly used fully connected layer (section 3.3.3) are first introduced, followed by their training methods (section 3.3.4), which are given in section 3.3. With the help of these CNN components, the available hand gesture recognition approach through figures is unlocked and could be implemented. In section 3.3.5, it is also demonstrated by, the CNNs applied previously in hand gesture recognition, that there are several difficulties for current deep learning-based hand gesture recognition applications for both accuracy and reaction time. Although not used in this thesis, the important materials of RNN like recurrent layer structure (section 3.4.1), network training method (section 3.4.2) and variations (section 3.4.3) are also reviewed, which gives a further understanding of the networks operation process and provides extra details about the layers and training process and led to an enhanced understanding of deep learning techniques. Finally, the third-generation network, SNN, is reviewed (section 3.5), including different training methods such as supervised learning (section 3.8.2), unsupervised bio-inspired learning (section 3.8.3) and learning by ANN conventions (section 3.8.4). This part of the review reveals the great possibility of achieving sEMG based hand gesture recognition through SNNs due to the naturally characteristic of the sEMG signals, essentially, the sEMG signals are the reflections of

the electric signals that transfer inside bio-neurons, which are spikes. Moreover, the review of the SNN based hand gesture recognition (section 3.9) is also contained in this chapter. This discovery helps to understand that, in the case of the SNN based hand gesture recognition, there is not a wide attempt to use sEMG signals, neither the formed spike images. Since the properties for sEMG and spikes are so similar, designing a method that is the concatenating of these two techniques seems quite essential, possible and has a high chance of expending both industry (prosthetics) and artificial intelligent (research) fields.

In chapter 4, the detailed operation process on generating intersected heatmaps is presented (section 4.2). By converting the original signals to heatmaps, the signal strength level became more evident when compared with the format of the raw sEMG signal, which makes it more convenient to identify the active channels among the sensors. Further, according to the formed heatmaps, not all the 128 channels are active when performing a certain gesture. In fact, the major information is contained in only a few sensors. Thus, the intersection algorithm (table 4.1) is created to filter out sensors with significant values i.e., finding common channels for different gestures. With the designed improved CNN architecture (section 4.3), the hand gesture recognition accuracy reached 98.96% through 35 sensors (figure 4.7). However, the diagram demonstrated in figure 4.6 indicates that for a certain number of sensors (0-35), the recognition accuracy increases with the rising of the total applied amount of the sensors. Once above a certain threshold (35 sensors for current situation), the increasing tendency of the recognition result. Finally, the recognition results which obtained through intersected heatmap algorithm and novel CNN architecture are presented (section 4.3). The comparison between the existing hand gesture recognition methods and intersected heatmap-CNN approach shown in table 4.2 and 4.3 (section 4.4) denotes that the intersected heatmap algorithm outperforms most of the current sEMG based hand gesture recognition methods for classifying gestures performed by able-bodied people and amputees. Moreover, the classification errors occurred because of the unequally spread of the active sensors may be minimized through the use of the discovered common channels/sensors technique for the datasets we have. The common channels/sensors also reveal that the total amount of channels/sensors for hand gesture classification task can be further reduced as some of the sensors contribute more than one gesture, which is presented by the research work shown in chapters 5 and 6.

The identification of the common channels for the self-collected sEMG dataset presented in chapter 4 had inspired and helped to further develop the research work shown in chapter 5. The common channels/sensors denote the ability of achieving high quality recognition performance with reduced required sensor numbers. The essential operation time for present sEMG based hand gesture recognition is quite large because it still requires lots of sensors to identify different gestures. In addition, current sEMG based hand gesture recognitions are mostly designed for individuals as the sEMG signals can be different for even the same gesture because of the differences of the muscle strength. Furthermore, according to the reviews in chapters 2 and 3, there were only a few hand gesture recognition research attempts involving amputees, nor to mention classification of hand gestures for amputees with different injury level. The common heatmap technique could be considered as the solution to all listed problems, as it is designed for finding common sensors for different subjects. Besides, the reduced sensor numbers also bring a large decrease of the necessary processing time, which gives the device equipped with this technique a considerable advantage when accomplishing real time tasks. According to the discovered common channels/sensors (table 5.1), the common heatmap algorithm is created (section 5.2). Through the proposed CSNN architecture (section 5.3), the hand gesture recognition achieves an average 98.4% recognition accuracy (table 5.3). comparing to the conventional deep learning networks, our CSNN architecture not only further minimize the required training time, but also reduces the parameters for training through using spike images generated by the LIF coding. Experiment results comparation (table 5.4) indicates that, the CSNN and common heatmap algorithm outperforms most of the former research works for both the processing time and recognition accuracy. However, as the prior results are not all obtained through time domain features (magnitude), some of them are acquired from frequency domain features. Thus, an experiment that aims at exploring the possibility of applying frequency domain features on hand gesture recognition is set up and eventually results in the creation of the research work shown in chapter 6, which combines the FDM with CSNN based hand gesture recognition.

For chapters 4 and 5, it can be concluded that the heatmap algorithm is an efficient approach when dealing with sEMG based hand gesture recognition tasks. Through converting raw sEMG signals into figures, the information contained inside each channels/sensors could be presented in a clearer form. Moreover, it is possible to achieve

good recognition performance with a limited number of channels/sensors for both amputees and able-bodied people.

Chapter 6 includes hand gesture classification works based on the frequency features of sEMG signals. As many review papers in chapters 2 and 3 mentioned, frequency-based approaches such as wavelet transform can be applied for sEMG signals, even though the achieved recognition accuracy listed on these papers are not satisfactory. After comparing the obtained sEMG signal graphs for both frequency domain and time domain, the reason can be concluded as to why the noise has greater influence in the time domain which results in the covering of the features. Thus, the SSA (section 2.5.3) was used for advance denoising. According to the results presented in figure 2.15 and 2.16, the features in frequency domain are much more evident. With the SSA and FDM algorithm (section 6.2), the sEMG signal classification based on frequency domain features shows improved accuracy when comparing with previous works under the same conditions, which is proved by the results comparison in table 6.2 (section 6.3.4). In addition, table 6.2 also denotes that the common channels/gestures are valuable for both time domain and frequency domain. In other words, the features included in sEMG signals can be used by many domains and the gesture recognition can be achieved by either one of them. With an average 93% accuracy, the proposed CSNN (section 5.3) architecture requires much lower processing time. As the features are in spike forms, the total energy consumption of the whole recognition system is reduced. Thus, it is reasonable to consider its performance on real time applications while less energy and faster response time is required. In other words, the spike approach is applicable to real time application while conservation of power is always in demand. However, according to the comparison shown in table 6.1 (section 6.3.4), current performance of the hand gesture recognition which based on SSA-FDM algorithm is poorer than previous ones based on heatmaps and common heatmaps. Though, the losses (4.59%) might be considered tiny, it indicates that there is still potential to improve the sEMG frequency features-based hand gesture recognitions, both for signal processing and network architecture enhancement. The fast SSA included in appendix 1 might provide an answer.

The results obtained for hand gesture recognition using data from both able-bodied people and amputees in this thesis denote that, efficient techniques with better recognition accuracies have been developed which requires less sEMG channels/sensors

for training, is suitable for both amputees and healthy people and reduces training time and effort. The unique and proposed methods are able to classify hand gestures based on features from time domain or frequency domain. These methods help to substantially reduce problems such as slow movement, electrical interference compromised reliability, defective classification results, time delayed responses, etc. for current hand gesture recognition and prosthetic field, which may bring the development of the sEMG based prosthetic to the next level. Furthermore, each of the pre-processing methods is applied individually. In cases where power consumption has exceptionally stringent requirements, it may be advisable to leverage the SNN-based methodologies. Nevertheless, this doesn't preclude their complementary use. For instance, features extracted from conventional heatmaps, and frequency density maps can be combined within a single network for classification purposes. However, there remain additional avenues for enhancing the methodologies outlined in this thesis, which are discussed in the following section.

In general, the pre-processing techniques and the Spiking Neural Network (SNN) used in the sEMG-based hand recognition system presented in this thesis offer advantages over conventional neural network-based sEMG hand prostheses in several key aspects:

Firstly, the system closely emulates the processing found in biological systems, making it well-suited for applications where replicating human neural behaviour is of paramount importance.

Secondly, the system excels in managing temporal data, making it an ideal choice for tasks that involve real-time or time-series data.

Thirdly, the system has the potential to achieve efficient processing with reduced computational demands, which can be highly advantageous.

Fourthly, thanks to the event-driven nature of SNNs and reduced sensor input requirements, the system can be more energy-efficient. This may lead to extended battery life in portable devices.

These advantages collectively position the system as a promising and innovative solution for sEMG-based hand recognition, offering enhanced performance and efficiency over traditional neural network-based approaches.

7.2 Future Work

There are many addition aspects of the works presented in this thesis that can be further explored and achieved:

- For chapters 4, 5 and 6, the features contained inside sEMG signals are transformed to maps and they produced high accuracy. It is reasonable to assume that applying the raw sEMG signals directly could achieve the same result since that the features are inherent in the signals. Besides, the raw sEMG signals can be considered as spikes, which naturally match the SNN architecture. Thus, designing an sEMG based hand gesture recognition algorithm which avoids the transformation of the raw signal could be possible.
- For the works presented in chapter 4, the common sensors identification result indicated that some of the common sensors are active for multiple gestures. Which, in other words, means there is still possibility for these sensors to cause classification errors. By designing a proposed algorithm which removes these multi-active sensors, the performance of the recognition accuracy could be further improved or maintained at the same scale with fewer inputs. Additionally, the training time and efforts for the whole system is also reduced as the input parameters are decreased.
- In chapter 6, the performance of the frequency domain features of the sEMG signals are evaluated. Considering that the previous excellent recognition results obtained based on time domain features, an algorithm which combines features from both two domains might produce a vital boost of the network performance. This can be an algorithm which separate into two parts with each of them handle one domain features and forms all the features in one map/figure in the end.
- The research works presented in chapters 4, 5 and 6 require the pre-processing of the raw signals, e.g., use of filters. Based on the author's knowledge, it is possible to embed these filters into one neural network which provides a similar effect [266][267]. Compared to conventional filters, these filter networks show an

improved performance as their dynamic parameters allow them to change their behaviours to best match the type of signal that is being filtered. Furthermore, the network architectures shown in these three chapters can be further optimized through approaches such as: replacing the activation functions, using different LIF coding methods, changing the size of the kernels for convolution operation, etc. These would further reduce the training time and effort without affecting network performance.

- The work presented in chapter 4 hold the potential for real-time implementation on platforms like Raspberry Pi [268] to drives the hand prosthetic. Each of the works in chapters 5 and 6 can potentially be implemented in real time on a neuromorphic chip also to drives the hand prosthetic. Such a real-time system was not available for this research due to time limitations.

Appendix A

Novel Two-Dimensional Singular Spectrum Analysis (FSSA)

The difference between the SSA and FSSA is the SVD process. The FSSA is based on the common embedding process applied to each pixel before the SVD, which are similar transformation matrices. In other words, a single common matrix can be applied for the all the pixels when doing SVD in FSSA. Besides, this unique transformation matrix is based on a special SVD which can be used on a representative signal from the whole data set to be transformed [101].

Appendix B

Python code for proposed CNN.

```
################################################
# Libraries
################################################

from __future__ import division, absolute_import
import numpy as np
import tflearn
import matplotlib.pyplot as plt
import matplotlib.image as mpimg
from tflearn.layers.core import input_data, dropout, fully_connected
from tflearn.layers.conv import conv_2d, max_pool_2d
from tflearn.layers.estimator import regression
#from keras.models import Sequential
import glob

seed = 7
np.random.seed(seed)

SAVE_DIRECTORY = "D:/project/spectrogram/model"
SAVE_MODEL_FILENAME = '/gesture_recognition_model.tfl'
################################################
# Loading the training data
################################################

images=np.load('gesture_spectrogram_data.npy')

labels=np.load('gesture_label_data.npy')


num_example=images.shape[0]
```

```
ratio=0.8

s=np.int(num_example*ratio)


images=images.reshape([-1, 64, 64, 1])


label=labels.reshape([-1, 8])


################################################
# CNN architecture
################################################

class DNN_gesture_recogniton:

  def Convnet(self):

    print('[+] Building CNN')
    self.network = input_data(shape = [None, 64, 64, 1])
    self.network = conv_2d(self.network, 32, 3, activation = 'relu')
    self.network = conv_2d(self.network, 32, 3, activation = 'relu')
    self.network = conv_2d(self.network, 32, 3, activation = 'relu')

    self.network = max_pool_2d(self.network, 2, strides = 2)

    self.network = conv_2d(self.network, 64, 3, activation = 'relu')
    self.network = conv_2d(self.network, 64, 3, activation = 'relu')
    self.network = conv_2d(self.network, 64, 3, activation = 'relu')

    self.network = max_pool_2d(self.network, 2, strides = 2)

    self.network = conv_2d(self.network, 64, 3, activation = 'relu')
    self.network = conv_2d(self.network, 64, 3, activation = 'relu')
    self.network = conv_2d(self.network, 64, 3, activation = 'relu')

    self.network = max_pool_2d(self.network, 2, strides = 2)


    self.network = fully_connected(self.network, 512, activation = 'relu')
    self.network = dropout(self.network, 0.5)
    self.network = fully_connected(self.network, 512, activation = 'relu')
    self.network = dropout(self.network, 0.5)
    self.network = fully_connected(self.network, 128, activation = 'relu')
    self.network = dropout(self.network, 0.5)

    self.network = fully_connected(self.network, 8, activation = 'softmax')
    self.network = regression(self.network,optimizer = 'adam',loss =
'categorical_crossentropy',learning_rate=0.001) # Gradient descent
optimizer and loss function
    self.model = tflearn.DNN(self.network,checkpoint_path
='D:/project/spectrogram/chekpoint/gesture_recognition_model.tfl' ,
max_checkpoints = 1,tensorboard_verbose = 2)

    #self.Load_Model()


########################   Training parameters
#############################
```

```
  def Training(self):
    X_train=images[:s]
    X_test=images[s:]
    y_train=labels[:s]
    y_test=labels[s:]
    self.Convnet()

    # Training
    print('[+] Training network')
    self.model.fit(
      X_train,  y_train,
      validation_set = (X_test, y_test),
      n_epoch = 10,  # One epoch stands for one forward pass and one
backward pass of all the training examples. It is the number of times the
algorithm sees the entire data set.
      batch_size = 32, # The Batch size is the number of training samples
your training will use (in one forward and one backward pass) in order to
make one update to the model parameters.
      shuffle = True,
      show_metric = True,
      snapshot_step = 200,
      snapshot_epoch = True,
      run_id = 'gesture_recognition'
    )

#########################      Predict image
############################

  def Prediction(self, image):
    if image is None:
      return None
    image = image.reshape([-1, 64, 64, 1])
    return self.model.predict(image)

#########################        Save model
############################

  def Save_Model(self):

self.model.save('D:/project/spectrogram/model/together/gesture_recognition_
model.tfl')
    print('[+] Model trained and saved at ' + SAVE_MODEL_FILENAME)

#########################  Load pretrained model
############################

  def Load_Model(self):
    self.model.load('D:/project/spectrogram/model/together/35
98.96%/gesture_recognition_model.tfl')

###############################################
# Running scripts on the terminal
###############################################
network = DNN_gesture_recogniton()
network.Training()
network.Save_Model()
#network.Load_Model()

test = []
for img_path in glob.glob('D:/project/spectrogram/123/35_all/test/*.jpg'):
    test.append(mpimg.imread(img_path))
```

```
#for i in range(len(test)):
#result = network.Prediction(test[3])
#print("Prediction: %s" % str(result[0]))
```

Appendix C

Python code for proposed CSNN

```
# -*- coding: utf-8 -*-
"""
Created on Sun Jan 17 16:15:42 2021

@author: 慧轩真厉害
"""

import nengo
import tensorflow as tf
import numpy as np
import matplotlib.pyplot as plt
import nengo_dl
##load data
image=np.load('gesture_spectrogram_data.npy')
label=np.load('gesture_label_data.npy')
label = label.astype(np.int64)
num_example=image.shape[0]
ratio=0.8
s=np.int(num_example*ratio)

#img = np.zeros((4096,461040))
#img = np.zeros((4096,2720))
img = np.zeros((4096,38394))
#img = np.zeros((4096,952))
#for i in range(461040):
#for i in range(2720):
#for i in range(952):
for i in range(38394):
    [width,height]=[image.shape[1],image.shape[2]]
    b = image[i].reshape(width*height);
    im = img[:,i]
    img[:,i] = np.add(im,b)

data_img = np.transpose(img)

train_images = [data_img[:s]]
train_labels = [label[:s]]
test_images = [data_img[s:]]
test_labels = [label[s:]]
with nengo.Network(seed=0) as net:
    # set some default parameters for the neurons that will make
    # the training progress more smoothly
    net.config[nengo.Ensemble].max_rates = nengo.dists.Choice([100])
    net.config[nengo.Ensemble].intercepts = nengo.dists.Choice([0])
    net.config[nengo.Connection].synapse = None
```

```
    neuron_type = nengo.LIF(amplitude=0.01)

    # this is an optimization to improve the training speed,
    # since we won't require stateful behaviour in this example
    nengo_dl.configure_settings(stateful=False)

    # the input node that will be used to feed in input images
    inp = nengo.Node(np.zeros(28 * 28))

    # add the first convolutional layer
    x = nengo_dl.Layer(tf.keras.layers.Conv2D(filters=32, kernel_size=3))(
        inp, shape_in=(28, 28, 1)
    )
    x = nengo_dl.Layer(neuron_type)(x)

    # add the second convolutional layer
    x = nengo_dl.Layer(tf.keras.layers.Conv2D(filters=64, strides=2,
kernel_size=3))(
        x, shape_in=(26, 26, 32)
    )
    x = nengo_dl.Layer(neuron_type)(x)

    # add the third convolutional layer
    x = nengo_dl.Layer(tf.keras.layers.Conv2D(filters=128, strides=2,
kernel_size=3))(
        x, shape_in=(12, 12, 64)
    )
    x = nengo_dl.Layer(neuron_type)(x)

    # x = nengo_dl.Layer(tf.keras.layers.Conv2D(filters=256, strides=2,
kernel_size=3))(
    #     x, shape_in=(12, 12, 128)
    # )
    # x = nengo_dl.Layer(neuron_type)(x)
    # linear readout
    out = nengo_dl.Layer(tf.keras.layers.Dense(units=8))(x)

    # we'll create two different output probes, one with a filter
    # (for when we're simulating the network over time and
    # accumulating spikes), and one without (for when we're
    # training the network using a rate-based approximation)
    out_p = nengo.Probe(out, label="out_p")
    out_p_filt = nengo.Probe(out, synapse=0.1, label="out_p_filt")

minibatch_size = 200
sim = nengo_dl.Simulator(net, minibatch_size=minibatch_size)

# add single timestep to training data
train_images = train_images[:, None, :]
train_labels = train_labels[:, None, None]

# when testing our network with spiking neurons we will need to run it
# over time, so we repeat the input/target data for a number of
# timesteps.
n_steps = 30
test_images = np.tile(test_images[:, None, :], (1, n_steps, 1))
test_labels = np.tile(test_labels[:, None, None], (1, n_steps, 1))

def classification_accuracy(y_true, y_pred):
    return tf.metrics.sparse_categorical_accuracy(y_true[:, -1], y_pred[:,
-1])
```

```
# note that we use `out_p_filt` when testing (to reduce the spike noise)
sim.compile(loss={out_p_filt: classification_accuracy})
print(
    "Accuracy before training:",
    sim.evaluate(test_images, {out_p_filt: test_labels},
verbose=0)["loss"],
)

do_training = True
if do_training:
    # run training
    sim.compile(
        optimizer=tf.optimizers.RMSprop(0.001),
        loss={out_p:
tf.losses.SparseCategoricalCrossentropy(from_logits=True)},
    )
    sim.fit(train_images, {out_p: train_labels}, epochs=10)

    # save the parameters to file
    sim.save_params("./gestures_params")

sim.compile(loss={out_p_filt: classification_accuracy})
print(
    "Accuracy after training:",
    sim.evaluate(test_images, {out_p_filt: test_labels},
verbose=0)["loss"],
)

data = sim.predict(test_images[:minibatch_size])

for i in range(5):
    plt.figure(figsize=(8, 4))
    plt.subplot(1, 2, 1)
    plt.imshow(test_images[i, 0].reshape((28, 28)), cmap="gray")
    plt.axis("off")

    plt.subplot(1, 2, 2)
    plt.plot(tf.nn.softmax(data[out_p_filt][i]))
    plt.legend([str(i) for i in range(10)], loc="upper left")
    plt.xlabel("timesteps")
    plt.ylabel("probability")
    plt.tight_layout()

sim.close()
```

Reference

[1] Y. Lecun, Y. Bengio, and G. Hinton, “Deep learning,” 2015. doi: 10.1038/nature14539.

[2] Y. Taigman, M. Yang, M. Ranzato, and L. Wolf, “DeepFace: Closing the gap to human-level performance in face verification,” in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2014. doi: 10.1109/CVPR.2014.220.

[3] G. Chen, T. X. Han, Z. He, R. Kays, and T. Forrester, “Deep convolutional neural network based species recognition for wild animal monitoring,” in *2014 IEEE International Conference on Image Processing, ICIP 2014*, 2014. doi: 10.1109/ICIP.2014.7025172.

[4] M. Ahmad, I. Ahmed, F. A. Khan, F. Qayum, and H. Aljuaid, “Convolutional neural network–based person tracking using overhead views,” *Int. J. Distrib. Sens. Networks*, vol. 16, no. 6, 2020, doi: 10.1177/1550147720934738.

[5] M. F. Owings and L. J. Kozak, “Ambulatory and inpatient procedures in the United States, 1996,” *Vital Heal. Stat. Ser. 13 Data Heal. Resour. Util.*, vol. 13, no. 139, 1998.

[6] E. Colleoni, S. Moccia, X. Du, E. De Momi, and D. Stoyanov, “Deep Learning Based Robotic Tool Detection and Articulation Estimation with Spatio-Temporal Layers,” *IEEE Robot. Autom. Lett.*, vol. 4, no. 3, 2019, doi: 10.1109/LRA.2019.2917163.

[7] T. Kounalakis, G. A. Triantafyllidis, and L. Nalpantidis, “Deep learning-based visual recognition of rumex for robotic precision farming,” *Comput. Electron. Agric.*, vol. 165, 2019, doi: 10.1016/j.compag.2019.104973.

[8] D. Quillen, E. Jang, O. Nachum, C. Finn, J. Ibarz, and S. Levine, “Deep reinforcement learning for vision-based robotic grasping: A simulated comparative evaluation of off-policy methods,” in *Proceedings - IEEE International Conference on Robotics and Automation*, 2018. doi: 10.1109/ICRA.2018.8461039.

[9] K. Chhaya, A. Khanzode, and R. D. Sarode, “Advantages and Disadvantages of Artificial Intelligence and Machine Learning: A Literature Review,” *Int. J. Libr. Inf. Sci.*, vol. 9, no. 1, 2020.

[10] V. Nasir and F. Sassani, “A review on deep learning in machining and tool monitoring: methods, opportunities, and challenges,” 2021. doi: 10.1007/s00170-021-07325-7.

[11] J. C. Huang, K. M. Ko, M. H. Shu, and B. M. Hsu, “Application and comparison of several machine learning algorithms and their integration models in regression problems,” *Neural Comput. Appl.*, vol. 32, no. 10, 2020, doi: 10.1007/s00521-019-04644-5.

[12] V. V. Gavrishchaka, Z. Yang, X. (Rebecca) Miao, and O. Senyukova, “Advantages of hybrid deep learning frameworks in applications with limited data,” *Int. J. Mach. Learn. Comput.*, vol. 8, no. 6, 2018, doi: 10.18178/ijmlc.2018.8.6.744.

[13] S. Ding, N. Zhang, X. Xu, L. Guo, and J. Zhang, “Deep Extreme Learning Machine and Its Application in EEG Classification,” *Math. Probl. Eng.*, vol. 2015, 2015, doi: 10.1155/2015/129021.

[14] G. Li *et al.*, “Hand gesture recognition based on convolution neural network,” *Cluster Comput.*, vol. 22, 2019, doi: 10.1007/s10586-017-1435-x.

[15] M. A. Abu, S. Rosleesham, M. Z. Suboh, M. S. M. Yid, Z. Kornain, and N. F. Jamaluddin, “Classification of EMG signal for multiple hand gestures based on neural network,” *Indones. J. Electr. Eng. Comput. Sci.*, vol. 17, no. 1, 2019, doi: 10.11591/ijeecs.v17.i1.pp256-263.

[16] R. H. Chowdhury, M. B. I. Reaz, M. A. Bin Mohd Ali, A. A. A. Bakar, K. Chellappan, and T. G. Chang, “Surface electromyography signal processing and classification techniques,” 2013. doi: 10.3390/s130912431.

[17] A. Craik, Y. He, and J. L. Contreras-Vidal, “Deep learning for electroencephalogram (EEG) classification tasks: A review,” 2019. doi: 10.1088/1741-2552/ab0ab5.

[18] N. Massó, F. Rey, D. Romero, G. Gual, L. Costa, and A. Germán, “Surface electromyography applications,” *Apunt. Med. l’esport*, vol. 45, no. 166, 2010, doi: 10.1016/j.apunts.2010.02.005.

[19] G. M. Kim, J. E. Powell, S. A. Lacey, J. A. Butkus, and D. G. Smith, “Current and emerging prostheses for partial hand amputation: A narrative review,” 2023. doi: 10.1002/pmrj.12764.

[20] S. A. Raurale, “Acquisition of EMG signals to recognize multiple Hand Gestures for Prosthesis Robotic Hand-A Review,” *Int. J. Curr. Eng. Technol.*, vol. 4, no. 1, 2014.

[21] M. Sinke, A. Chadwell, and G. Smit, “State of the art of prosthesis simulators for the upper limb: A narrative review,” 2022. doi: 10.1016/j.rehab.2022.101635.

[22] W. Maass, “Networks of spiking neurons: The third generation of neural network models,” *Neural Networks*, vol. 10, no. 9, 1997, doi: 10.1016/S0893-6080(97)00011-7.

[23] P. A. Merolla *et al.*, “A million spiking-neuron integrated circuit with a scalable communication network and interface,” *Science (80-.).*, vol. 345, no. 6197, 2014, doi: 10.1126/science.1254642.

[24] F. Akopyan *et al.*, “TrueNorth: Design and Tool Flow of a 65 mW 1 Million Neuron Programmable Neurosynaptic Chip,” *IEEE Trans. Comput. Des. Integr. Circuits Syst.*, vol. 34, no. 10, 2015, doi: 10.1109/TCAD.2015.2474396.

[25] M. Davies *et al.*, “Loihi: A Neuromorphic Manycore Processor with On-Chip Learning,” *IEEE Micro*, vol. 38, no. 1, 2018, doi: 10.1109/MM.2018.112130359.

[26] D. Ielmini and S. Ambrogio, “Emerging neuromorphic devices,” 2020. doi: 10.1088/1361-6528/ab554b.

[27] C. J. De Luca, “The use of surface electromyography in biomechanics,” in *Journal of Applied Biomechanics*, 1997. doi: 10.1123/jab.13.2.135.

[28] M. B. I. Reaz, M. S. Hussain, and F. Mohd-Yasin, “Techniques of EMG signal analysis: Detection, processing, classification and applications,” *Biol. Proced. Online*, vol. 8, no. 1, pp. 11–35, 2006, doi: 10.1251/bpo115.

[29] R. F. M. Kleissen, J. H. Buurke, J. Harlaar, and G. Zilvold, “Electromyography in the biomechanical analysis of human movement and its clinical application,” 1998. doi: 10.1016/S0966-6362(98)00025-3.

[30] B. Rodríguez-Tapia, I. Soto, D. M. Marínez, and N. C. Arballo, “Myoelectric Interfaces and Related Applications: Current State of EMG Signal Processing-A Systematic Review,” 2020. doi: 10.1109/ACCESS.2019.2963881.

[31] “Cram’s Introduction to Surface Electromyography, 2nd Edition,” *Med. Sci. Sport. Exerc.*, vol. 43, no. 7, p. 1378, 2011, doi: 10.1249/01.mss.0000399576.80711.7d.

[32] H. J. Hermens, B. Freriks, C. Disselhorst-Klug, and G. Rau, “Development of recommendations for SEMG sensors and sensor placement procedures,” *J. Electromyogr. Kinesiol.*, vol. 10, no. 5, 2000, doi: 10.1016/S1050-6411(00)00027-4.

[33] C. J. Heckman and R. M. Enoka, “Physiology of the motor neuron and the motor unit,” *Handb. Clin. Neurophysiol.*, vol. 4, no. C, 2004, doi: 10.1016/S1567-4231(04)04006-7.

[34] L. Al-Qusairi and J. Laporte, “T-tubule biogenesis and triad formation in skeletal muscle and implication in human diseases,” 2011. doi: 10.1186/2044-5040-1-26.

[35] S. Micera and G. Vannozzi, “Improving detection of muscle activation intervals,” *IEEE Eng Med Biol Mag.*, vol. 20, no. 6, 2001.

[36] A. J. Thexton, “A randomisation method for discriminating between signal and noise in recordings of rhythmic electromyographic activity,” *J. Neurosci. Methods*, vol. 66, no. 2, 1996, doi: 10.1016/0165-0270(96)00004-0.

[37] P. Bonato, T. D’Alessio, and M. Knaflitz, “A statistical method for the measurement of muscle activation intervals from surface myoelectric signal during gait,” *IEEE Trans. Biomed. Eng.*, vol. 45, no. 3, 1998, doi: 10.1109/10.661154.

[38] L. Xu and A. Adler, “An improved method for muscle activation detection during gait,” in *Canadian Conference on Electrical and Computer Engineering*, 2004. doi: 10.1109/ccece.2004.1345029.

[39] A. Phinyomark, A. Nuidod, P. Phukpattaranont, and C. Limsakul, “Feature extraction and reduction of wavelet transform coefficients for EMG pattern classification,” *Elektron. ir Elektrotechnika*, vol. 122, no. 6, 2012, doi: 10.5755/j01.eee.122.6.1816.

[40] N. M. Kakoty, A. Saikia, and S. M. Hazarika, “Exploring a family of wavelet transforms for EMG-based grasp recognition,” *Signal, Image Video Process.*, vol. 9, no. 3, 2015, doi: 10.1007/s11760-013-0477-7.

[41] M. R. Guglielminotti P, “Effect of electrode location on surface myoelectric signal variables: a simulation study,” *9th Int. Congr. ISEK*, 1992.

[42] F. Laterza and G. Olmo, “Analysis of EMG signals by means of the matched wavelet transform,” *Electron. Lett.*, vol. 33, no. 5, 1997, doi: 10.1049/el:19970250.

[43] A. R. Ismail and S. S. Asfour, “Continuous wavelet transform application to EMG signals during human gait,” in *Conference Record of the Asilomar Conference on Signals, Systems and Computers*, 1998. doi: 10.1109/acssc.1998.750880.

[44] C. S. Pattichis and M. S. Pattichis, “Time-scale analysis of motor unit action potentials,” *IEEE Trans. Biomed. Eng.*, vol. 46, no. 11, 1999, doi: 10.1109/10.797992.

[45] D. K. Kumar, N. D. Pah, and A. Bradley, “Wavelet Analysis of Surface Electromyography to Determine Muscle Fatigue,” *IEEE Trans. Neural Syst. Rehabil. Eng.*, vol. 11, no. 4, 2003, doi: 10.1109/TNSRE.2003.819901.

[46] C. L, *Time-frequency analysis*. Englewood Cliffs, Prentice-Hall; New Jersey,USA, 1995.

[47] A. L. Ricamato, R. G. Absher, M. T. Moffroid, and J. P. Tranowski, “A time-frequency approach to evaluate electromyographic recordings,” in *Proceedings - IEEE Symposium on Computer-Based Medical Systems*, 1992. doi: 10.1109/CBMS.1992.245010.

[48] G. R. Pereira, L. F. de Oliveira, and J. Nadal, “Reducing cross terms effects in the Choi-Williams transform of mioelectric signals,” *Comput. Methods Programs Biomed.*, vol. 111, no. 3, 2013, doi: 10.1016/j.cmpb.2013.06.004.

[49] D. Graupe and W. K. Cline, “Functional Separation Of Emg Signals Via Arma Identification Methods For Prosthesis Control Purposes,” *IEEE Trans. Syst. Man Cybern.*, vol. SMC-5, no. 2, 1975, doi: 10.1109/TSMC.1975.5408479.

[50] S. MH, “Stochastic Model of Myoelectric Signals for Movement Pattern Recognition in Upper Limb Prostheses,” University of California at Los Angeles, 1980.

[51] C. H. Bernatos L, Crago P, “A discrete-time model of electricity stimulated muscle.,” *IEEE Trans Biomed Eng*, pp. 33:829-838, 1986.

[52] A. T. Moser and D. Graupe, “Identification Of Nonstationary Models With Application To Myoelectric Signals For Controlling Electrical Stimulation Of Paraplegics,” *IEEE Trans. Acoust.*, vol. 37, no. 5, 1989, doi: 10.1109/29.17563.

[53] L. Bi, A. Feleke, and C. Guan, “A review on EMG-based motor intention prediction of continuous human upper limb motion for human-robot collaboration,” 2019. doi: 10.1016/j.bspc.2019.02.011.

[54] P. Wellig and G. S. Moschytz, “Analysis of wavelet features for myoelectric signal classification,” *Proc. IEEE Int. Conf. Electron. Circuits, Syst.*, vol. 3, 1998, doi: 10.1109/icecs.1998.813946.

[55] B. Boashash and P. O’Shea, “A Methodology for Detection and Classification of Some Underwater Acoustic Signals Using Time-Frequency Analysis Techniques,” *IEEE Trans. Acoust.*, vol. 38, no. 11, 1990, doi: 10.1109/29.103085.

[56] C. I. Christodoulou and C. S. Pattichis, “New technique for the classification and decomposition of EMG signals,” in *IEEE International Conference on Neural Networks - Conference Proceedings*, 1995. doi: 10.1109/icnn.1995.487720.

[57] F. H. Y. Chan, Y. S. Yang, F. K. Lam, Y. T. Zhang, and P. A. Parker, “Fuzzy EMG classification for prosthesis control,” *IEEE Trans. Rehabil. Eng.*, vol. 8, no. 3, 2000, doi: 10.1109/86.867872.

[58] G. Cheron, J. P. Draye, M. Bourgeios, and G. Libert, “A dynamic neural network identification of electromyography and arm trajectory relationship during complex movements,” *IEEE Trans. Biomed. Eng.*, vol. 43, no. 5, 1996, doi: 10.1109/10.488803.

[59] C. L. Gooch *et al.*, “Motor unit number estimation: A technology and literature review,” 2014. doi: 10.1002/mus.24442.

[60] A. J. McComas, P. R. Fawcett, M. J. Campbell, and R. E. Sica, “Electrophysiological estimation of the number of motor units within a human muscle.,” *J. Neurol. Neurosurg. Psychiatry*, vol. 34, no. 2, 1971, doi: 10.1136/jnnp.34.2.121.

[61] Z. Xu and S. Xiao, “Estimation of motor unit firing statistics from surface EMG,” in *Annual International Conference of the IEEE Engineering in Medicine and Biology - Proceedings*, 1998. doi: 10.1109/iembs.1998.745114.

[62] P. Zhou and W. Z. Rymer, “Motor unit action potential number estimation in the surface electromyogram: wavelet matching method and its performance boundary,” in *International IEEE/EMBS Conference on Neural Engineering, NER*, 2003. doi: 10.1109/CNE.2003.1196829.

[63] L. A. Major and K. E. Jones, “Simulations of motor unit number estimation techniques,” *J. Neural Eng.*, vol. 2, no. 2, 2005, doi: 10.1088/1741-2560/2/2/003.

[64] D. Graupe, J. Magnussen, and A. A. Beex, “A Microprocessor System for Multifunctional Control of Upper-Limb Prostheses via Myoelectric Signal Identification,” *IEEE Trans. Automat. Contr.*, vol. 23, no. 4, 1978, doi: 10.1109/TAC.1978.1101783.

[65] C. J. Yen, W. Y. Chung, K. P. Lin, C. L. Tsai, S. H. Lee, and T. S. Chen, “Analog integrated circuit design for the wireless bio-signal transmission system,” in *AP-ASIC 1999 - 1st IEEE Asia Pacific Conference on ASICs*, 1999. doi: 10.1109/APASIC.1999.824099.

[66] J. Torresen, “Two-step incremental evolution of a prosthetic hand controller based on digital logic gates,” in *Lecture Notes in Computer Science (including subseries Lecture*

Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), 2001. doi: 10.1007/3-540-45443-8_1.

[67] B. Mitchell and L. Whited, *Anatomy, Shoulder and Upper Limb, Forearm Muscles*. 2019.

[68] R. French, "THE NATURAL HISTORY OF PLINY," in *Ancient Natural History*, 2020. doi: 10.4324/9780203974162-12.

[69] V. Putti, "Historical prostheses," *J. Hand Surg. Am.*, vol. 30, no. 3, 2005, doi: 10.1016/j.jhsb.2005.01.001.

[70] K. J. Zuo and J. L. Olson, "The evolution of functional hand replacement: From iron prostheses to hand transplantation," *Can. J. Plast. Surg.*, vol. 22, no. 1, pp. 44–51, 2014, doi: 10.1177/229255031402200111.

[71] R. Meier, *History of arm amputation, prosthetic restoration, and arm amputation rehabilitation*. New York: Demos Medical, 2004.

[72] W. WM, "Alms and Legs in France," *Macmillan's Magazine*, London, pp. 438–440, 1879.

[73] F. Sauerbruch, *Die willkürlich bewegbare künstliche Hand*. 1916. doi: 10.1007/978-3-642-64935-6.

[74] R. P. Petri and E. Aguila, "The military upper extremity amputee," 2002. doi: 10.1016/S1047-9651(03)00070-6.

[75] K. Ostlie, I. M. Lesjø, R. J. Franklin, B. Garfelt, O. H. Skjeldal, and P. Magnus, "Prosthesis use in adult acquired major upper-limb amputees: Patterns of wear, prosthetic skills and the actual use of prostheses in activities of daily life," *Disabil. Rehabil. Assist. Technol.*, vol. 7, no. 6, 2012, doi: 10.3109/17483107.2011.653296.

[76] J. D. Brown, T. S. Kunz, D. Gardner, M. K. Shelley, A. J. Davis, and R. B. Gillespie, "An Empirical Evaluation of Force Feedback in Body-Powered Prostheses," *IEEE Trans. Neural Syst. Rehabil. Eng.*, vol. 25, no. 3, 2017, doi: 10.1109/TNSRE.2016.2554061.

[77] D. S. Childress, “Historical Aspects of Powered Limb Prostheses,” *J. Prosthetics Orthot.*, vol. 9, no. 1, 1985.

[78] R. N. Scott, “Myoelectric Control of Prostheses: A brief History,” in *Proceedings of the 1992 MyoElectric Controls/Powered Prosthetics Symposium Fredericton*, 1992.

[79] E. D. Sherman and A. Lippay, “A Russian Bioelectric-Controlled Prosthesis.,” *Can. Med. Assoc. J.*, vol. 92, no. 5, 1965.

[80] C. Behrend, W. Reizner, J. A. Marchessault, and W. C. Hammert, “Update on advances in upper extremity prosthetics,” 2011. doi: 10.1016/j.jhsa.2011.07.024.

[81] A. Fleming, N. Stafford, S. Huang, X. Hu, D. P. Ferris, and H. H. Huang, “Myoelectric control of robotic lower limb prostheses: A review of electromyography interfaces, control paradigms, challenges and future directions,” 2021. doi: 10.1088/1741-2552/ac1176.

[82] R. N. Scott and P. A. Parker, “Myoelectric prostheses: State of the art,” *J. Med. Eng. Technol.*, vol. 12, no. 4, 1988, doi: 10.3109/03091908809030173.

[83] Ottobock, “BeBionic Hand,” *Ottobock UK*, 2013.

[84] R. G. E. Clement, K. E. Bugler, and C. W. Oliver, “Bionic prosthetic hands: A review of present technology and future aspirations,” 2011. doi: 10.1016/j.surge.2011.06.001.

[85] L. Haverkate, G. Smit, and Di. H. Plettenburg, “Assessment of body-powered upper limb prostheses by able-bodied subjects, using the Box and Blocks Test and the Nine-Hole Peg Test,” *Prosthet. Orthot. Int.*, vol. 40, no. 1, 2016, doi: 10.1177/0309364614554030.

[86] P. Parker, K. Englehart, and B. Hudgins, “Myoelectric signal processing for control of powered limb prostheses,” *J. Electromyogr. Kinesiol.*, vol. 16, no. 6, 2006, doi: 10.1016/j.jelekin.2006.08.006.

[87] A. E. Schultz and T. A. Kuiken, “Neural Interfaces for Control of Upper Limb Prostheses: The State of the Art and Future Possibilities,” *PM R*, vol. 3, no. 1, 2011, doi: 10.1016/j.pmrj.2010.06.016.

[88] K. E. Yu *et al.*, “Clinical evaluation of the revolutionizing prosthetics modular prosthetic limb system for upper extremity amputees,” *Sci. Rep.*, vol. 11, no. 1, 2021, doi: 10.1038/s41598-020-79581-8.

[89] WorkSafeBC Evidence-Based Practice Group and Dr. Craig W. Martin, “Upper Limb Prostheses A Review of the Literature With a Focus on Myoelectric Hands,” *Clin. Serv. – Work. Empl. Serv.*, no. February, 2011.

[90] C. M. Light, P. H. Chappell, B. Hudgins, and K. Engelhart, “Intelligent multifunction myoelectric control of hand prostheses,” 2002. doi: 10.1080/03091900210142459.

[91] R. Brånemark, L. O. Öhrnell, P. Nilsson, and P. Thomsen, “Biomechanical characterization of osseointegration during healing: An experimental in vivo study in the rat,” *Biomaterials*, vol. 18, no. 14, 1997, doi: 10.1016/S0142-9612(97)00018-5.

[92] S. Jönsson, K. Caine-Winterberger, and R. Branemark, “Osseointegration amputation prostheses on the upper limbs: Methods, prosthetics and rehabilitation,” *Prosthet. Orthot. Int.*, vol. 35, no. 2, 2011, doi: 10.1177/0309364611409003.

[93] R. Menon, G. Di Caterina, H. Lakany, L. Petropoulakis, B. A. Conway, and J. J. Soraghan, “Study on Interaction between Temporal and Spatial Information in Classification of EMG Signals for Myoelectric Prostheses,” *IEEE Trans. Neural Syst. Rehabil. Eng.*, vol. 25, no. 10, 2017, doi: 10.1109/TNSRE.2017.2687761.

[94] A. Van Boxtel, “Optimal signal bandwidth for the recording of surface EMG activity of facial, jaw, oral, and neck muscles,” *Psychophysiology*, vol. 38, no. 1, 2001, doi: 10.1017/S004857720199016X.

[95] O. Bioelectronica, “Adhesive Matrix ELSCH064R3S Pin Out,” [Online].

[96] N. J. Jarque-Bou, J. L. Sancho-Bru, and M. Vergara, “A systematic review of EMG applications for the characterization of forearm and hand muscle activity during activities of daily living: Results, challenges, and open issues,” 2021. doi: 10.3390/s21093035.

[97] W. Geng, Y. Du, W. Jin, W. Wei, Y. Hu, and J. Li, “Gesture recognition by instantaneous surface EMG images,” *Sci. Rep.*, vol. 6, 2016, doi: 10.1038/srep36571.

[98] D. H. Schoellhamer, “Singular spectrum analysis for time series with missing data,” *Geophys. Res. Lett.*, vol. 28, no. 16, 2001, doi: 10.1029/2000GL012698.

[99] H. Hassani, “Singular Spectrum Analysis: Methodology and Comparison,” *J. Data Sci.*, vol. 5, no. 2, 2007, doi: 10.6339/JDS.2007.05(2).396.

[100] T. Qiao *et al.*, “Effective denoising and classification of hyperspectral images using curvelet transform and singular spectrum analysis,” *IEEE Trans. Geosci. Remote Sens.*, vol. 55, no. 1, 2017, doi: 10.1109/TGRS.2016.2598065.

[101] J. Zabalza, J. Ren, Z. Wang, H. Zhao, J. Wang, and S. Marshall, “Fast Implementation of Singular Spectrum Analysis for Effective Feature Extraction in Hyperspectral Imaging,” *IEEE J. Sel. Top. Appl. Earth Obs. Remote Sens.*, vol. 8, no. 6, 2015, doi: 10.1109/JSTARS.2014.2375932.

[102] A. M. Andrew, “Spiking Neuron Models: Single Neurons, Populations, Plasticity,” *Kybernetes*, vol. 32, no. 7/8, 2003, doi: 10.1108/k.2003.06732gae.003.

[103] B. L. Yoon, “Artificial neural network technology,” *ACM SIGSMALL/PC Notes*, vol. 15, no. 3, 1989, doi: 10.1145/74657.74658.

[104] F. Attneave, M. B., and D. O. Hebb, “The Organization of Behavior; A Neuropsychological Theory,” *Am. J. Psychol.*, vol. 63, no. 4, 1950, doi: 10.2307/1418888.

[105] Y. chen Wu and J. wen Feng, “Development and Application of Artificial Neural Network,” *Wirel. Pers. Commun.*, vol. 102, no. 2, 2018, doi: 10.1007/s11277-017-5224-x.

[106] S. Dong, J., & Hu, “The progress and prospects of neural network research,” *Inf. Control*, pp. 360–368, 1997.

[107] A. B. Magoun, “A nonrandom walk down memory lane with bernard widrow [scanning our past],” *Proc. IEEE*, vol. 102, no. 10, 2014, doi: 10.1109/JPROC.2014.2351193.

[108] G. Modi, S. Kumar, and B. Singh, “Improved Widrow-Hoff Based Adaptive Control of Multiobjective PV-DSTATCOM System,” *IEEE Trans. Ind. Appl.*, vol. 56, no. 2, 2020, doi: 10.1109/TIA.2019.2960732.

[109] M. Minsky and S. Papert, “Perceptron: an introduction to computational geometry,” *MIT Press. Cambridge, Expand. Ed.*, vol. 19, 1969.

[110] B. McMullin, “John von Neumann and the evolutionary growth of complexity: Looking backward, looking forward. . .,” *Artif. Life*, vol. 6, no. 4, 2000, doi: 10.1162/106454600300103674.

[111] M. G. Anderson D, “Artificial Neural Networks Technology,” *Kaman Sci. Corp.*, vol. 258, no. 6, 1992.

[112] J. J. Hopfield, “Neural networks and physical systems with emergent collective computational abilities.,” *Proc. Natl. Acad. Sci. U. S. A.*, vol. 79, no. 8, 1982, doi: 10.1073/pnas.79.8.2554.

[113] D. L. Reilly, L. N. Cooper, and C. Elbaum, “A neural model for category learning,” *Biol. Cybern.*, vol. 45, no. 1, 1982, doi: 10.1007/BF00387211.

[114] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, “Learning Internal Representations by Error Propagation,” in *Readings in Cognitive Science: A Perspective from Psychology and Artificial Intelligence*, 2013. doi: 10.1016/B978-1-4832-1446-7.50035-2.

[115] O. I. Abiodun, A. Jantan, A. E. Omolara, K. V. Dada, N. A. E. Mohamed, and H. Arshad, “State-of-the-art in artificial neural network applications: A survey,” 2018. doi: 10.1016/j.heliyon.2018.e00938.

[116] L. Zagrean, “Principles of Neural Science,” *Acta Endocrinol.*, vol. 10, no. 3, 2014, doi: 10.4183/aeb.2014.529.

[117] A. Intelligence, “Fundamentals of Neural Networks Artificial Intelligence Fundamentals of Neural Networks Artificial Intelligence,” *Fundam. Neural Networks AI Course Lect. 37 – 38, notes, slides*, 2010.

[118] M. W. Gardner and S. R. Dorling, “Artificial neural networks (the multilayer perceptron) - a review of applications in the atmospheric sciences,” *Atmos. Environ.*, vol. 32, no. 14–15, 1998, doi: 10.1016/S1352-2310(97)00447-0.

[119] N. A. Mat Isa and W. M. F. W. Mamat, “Clustered-Hybrid Multilayer Perceptron network for pattern recognition application,” *Appl. Soft Comput. J.*, vol. 11, no. 1, 2011, doi: 10.1016/j.asoc.2010.04.017.

[120] U. Orhan, M. Hekim, and M. Ozer, “EEG signals classification using the K-means clustering and a multilayer perceptron neural network model,” *Expert Syst. Appl.*, vol. 38, no. 10, 2011, doi: 10.1016/j.eswa.2011.04.149.

[121] D. Bhattacharjee, D. K. Basu, M. Nasipuri, and M. Kundu, “Human face recognition using fuzzy multilayer perceptron,” *Soft Comput.*, vol. 14, no. 6, 2010, doi: 10.1007/s00500-009-0426-0.

[122] F. A. A. M. N. Soares, E. L. Flôres, C. D. Cabacinha, G. A. Carrijo, and A. C. P. Veiga, “Recursive diameter prediction and volume calculation of eucalyptus trees using Multilayer Perceptron Networks,” *Comput. Electron. Agric.*, vol. 78, no. 1, 2011, doi: 10.1016/j.compag.2011.05.008.

[123] L. Noriega, “Multilayer perceptron tutorial,” *Sch. Comput. Staff. Univ.*, 2005.

[124] T. L. Fine, “Feedforward Neural Network Methodology,” *IEEE Trans. Neural Networks*, vol. 12, no. 3, 2001, doi: 10.1109/TNN.2001.925573.

[125] I. Goodfellow, “Deep Feedforward Networks,” *Deep Learn. B.*, no. 1, 2015.

[126] “Searching for activation functions,” in *6th International Conference on Learning Representations, ICLR 2018 - Workshop Track Proceedings*, 2018.

[127] S. Sharma, S. Sharma, and A. Athaiya, “ACTIVATION FUNCTIONS IN NEURAL NETWORKS,” *Int. J. Eng. Appl. Sci. Technol.*, vol. 04, no. 12, 2020, doi: 10.33564/ijeast.2020.v04i12.054.

[128] A. Zemmari and J. Benois-Pineau, “Optimization methods,” in *SpringerBriefs in Computer Science*, 2020. doi: 10.1007/978-3-030-34376-7_4.

[129] S. ichi Amari, “Backpropagation and stochastic gradient descent method,” *Neurocomputing*, vol. 5, no. 4–5, 1993, doi: 10.1016/0925-2312(93)90006-O.

[130] A. Galántai, “The theory of Newton’s method,” *J. Comput. Appl. Math.*, vol. 124, no. 1–2, 2000, doi: 10.1016/S0377-0427(00)00435-0.

[131] K. Schleicher, “The conjugate gradient method,” *Lead. Edge*, vol. 37, no. 4, 2018, doi: 10.1190/tle37040296.1.

[132] R. Schoenberg, “Optimization with the Quasi-Newton Method,” *Unpubl. Manuscr.*, 2001.

[133] S. Sapna, “Backpropagation Learning Algorithm Based on Levenberg Marquardt Algorithm,” 2012. doi: 10.5121/csit.2012.2438.

[134] W. C. Thacker, “The role of the Hessian matrix in fitting models to measurements,” *J. Geophys. Res.*, vol. 94, no. C5, 1989, doi: 10.1029/jc094ic05p06177.

[135] R. Fletcher, “Function minimization by conjugate gradients,” *Comput. J.*, vol. 7, no. 2, 1964, doi: 10.1093/comjnl/7.2.149.

[136] E. Polak and G. Ribiere, “Note sur la convergence de méthodes de directions conjuguées,” *Rev. française d’informatique Rech. opérationnelle. Série rouge*, vol. 3, no. 16, 1969, doi: 10.1051/m2an/196903r100351.

[137] Z. Lin, J. Maris, L. Hermans, J. Vandewalle, and J. D. Z. Chen, “Comparison of gradient descent and conjugate gradient learning algorithms for classification of electrogastrogram,” in *Annual International Conference of the IEEE Engineering in Medicine and Biology - Proceedings*, 1995. doi: 10.1109/iembs.1995.575378.

[138] M. Cilimkovic, “Neural Networks and Back Propagation Algorithm,” *Fett.Tu-Sofia.Bg*, no. July, 2015.

[139] S. Lawrence, C. L. Giles, A. C. Tsoi, and A. D. Back, “Face recognition: A convolutional neural-network approach,” *IEEE Trans. Neural Networks*, vol. 8, no. 1, 1997, doi: 10.1109/72.554195.

[140] S. Albawi, T. A. Mohammed, and S. Al-Zawi, “Understanding of a convolutional neural network,” in *Proceedings of 2017 International Conference on Engineering and Technology, ICET 2017*, 2018. doi: 10.1109/ICEngTechnol.2017.8308186.

[141] J. Nagi *et al.*, “Max-pooling convolutional neural networks for vision-based hand gesture recognition,” in *2011 IEEE International Conference on Signal and Image Processing Applications, ICSIPA 2011*, 2011. doi: 10.1109/ICSIPA.2011.6144164.

[142] D. Yu, H. Wang, P. Chen, and Z. Wei, “Mixed pooling for convolutional neural networks,” in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 2014. doi: 10.1007/978-3-319-11740-9_34.

[143] E. M. Johansson, F. U. Dowla, and D. M. Goodman, “BACKPROPAGATION LEARNING FOR MULTILAYER FEED-FORWARD NEURAL NETWORKS USING THE CONJUGATE GRADIENT METHOD,” *Int. J. Neural Syst.*, vol. 02, no. 04, 1991, doi: 10.1142/s0129065791000261.

[144] M. HafizurRahman and J. Afrin, “Hand Gesture Recognition using Multiclass Support Vector Machine,” *Int. J. Comput. Appl.*, vol. 74, no. 1, 2013, doi: 10.5120/12852-9367.

[145] F. A. Mufarroha and F. Utaminingrum, “Hand gesture recognition using adaptive network based fuzzy inference system and K-nearest neighbor,” *Int. J. Technol.*, vol. 8, no. 3, 2017, doi: 10.14716/ijtech.v8i3.3146.

[146] J. Shukla and A. Dwivedi, “A method for hand gesture recognition,” in *Proceedings - 2014 4th International Conference on Communication Systems and Network Technologies, CSNT 2014*, 2014. doi: 10.1109/CSNT.2014.189.

[147] H. J. Kim, J. S. Lee, and J. H. Park, “Dynamic hand gesture recognition using a CNN model with 3D receptive fields,” *2008 IEEE Int. Conf. Neural Networks Signal Process. ICNNSP*, pp. 14–19, 2008, doi: 10.1109/ICNNSP.2008.4590300.

[148] P. Molchanov, S. Gupta, K. Kim, and J. Kautz, “Hand gesture recognition with 3D convolutional neural networks,” in *IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops*, 2015. doi: 10.1109/CVPRW.2015.7301342.

[149] R. F. Pinto, C. D. B. Borges, A. M. A. Almeida, and I. C. Paula, “Static Hand Gesture Recognition Based on Convolutional Neural Networks,” *J. Electr. Comput. Eng.*, vol. 2019, 2019, doi: 10.1155/2019/4167890.

[150] H. Chen, R. Tong, M. Chen, Y. Fang, and H. Liu, “A Hybrid Cnn-SVM Classifier for Hand Gesture Recognition with Surface Emg Signals,” in *Proceedings - International Conference on Machine Learning and Cybernetics*, 2018. doi: 10.1109/ICMLC.2018.8526976.

[151] Y. Wu, B. Zheng, and Y. Zhao, “Dynamic Gesture Recognition Based on LSTM-CNN,” in *Proceedings 2018 Chinese Automation Congress, CAC 2018*, 2018. doi: 10.1109/CAC.2018.8623035.

[152] M. B. Priatama, L. Novamizanti, S. Aulia, and E. B. Candrasari, “Hand gesture recognition using discrete wavelet transform and convolutional neural network,” *Bull. Electr. Eng. Informatics*, vol. 9, no. 3, 2020, doi: 10.11591/eei.v9i3.1977.

[153] N. S. Velandia, R. J. Moreno, and A. Rubiano, “CNN Architectures for Hand Gesture Recognition using EMG Signals Throw Wavelet Feature Extraction,” *ARPN J. Eng. Appl. Sci.*, vol. 14, no. 11, 2019, doi: 10.36478/JEASCI.2019.3528.3537.

[154] V. Shanmuganathan, H. R. Yesudhas, M. S. Khan, M. Khari, and A. H. Gandomi, “R-CNN and wavelet feature extraction for hand gesture recognition with EMG signals,” *Neural Comput. Appl.*, vol. 32, no. 21, 2020, doi: 10.1007/s00521-020-05349-w.

[155] J. Wan *et al.*, “A Critical Review of Recurrent Neural Networks for Sequence Learning arXiv : 1506 . 00019v2 [cs . LG] 29 Jun 2015,” *Int. J. Comput. Vis.*, vol. 106, no. March 2013, 2015.

[156] L. R. Medsker and L. C. Jain, “Recurrent Neural Networks Design and Applications,” *J. Chem. Inf. Model.*, vol. 53, no. 9, 2013.

[157] P. J. Werbos, “Backpropagation Through Time: What It Does and How to Do It,” *Proc. IEEE*, vol. 78, no. 10, 1990, doi: 10.1109/5.58337.

[158] S. M. T. Müller B., Reinhardt J., “BTT: Back-Propagation Through Time,” *Neural Networks*, pp. 296–302, 1995, doi: https://doi.org/10.1007/978-3-642-57760-4_28.

[159] M. Schuster and K. K. Paliwal, “Bidirectional recurrent neural networks,” *IEEE Trans. Signal Process.*, vol. 45, no. 11, 1997, doi: 10.1109/78.650093.

[160] T.-Y. Yen and W. Wolf, “Previous Work,” *Hardware-Software Co-Synthesis Distrib. Embed. Syst.*, vol. 9, no. 8, pp. 13–39, 1996, doi: 10.1007/978-1-4757-5388-2_2.

[161] Y. Yu, X. Si, C. Hu, and J. Zhang, “A review of recurrent neural networks: Lstm cells and network architectures,” 2019. doi: 10.1162/neco_a_01199.

[162] F. A. Gers, J. Schmidhuber, and F. Cummins, “Learning to forget: Continual prediction with LSTM,” *Neural Comput.*, vol. 12, no. 10, 2000, doi: 10.1162/089976600300015015.

[163] S. Hochreiter and J. Schmidhuber, “LSTM can solve hard long time lag problems,” in *Advances in Neural Information Processing Systems*, 1997.

[164] M. Pfeiffer and T. Pfeil, “Deep Learning With Spiking Neurons: Opportunities and Challenges,” *Front. Neurosci.*, vol. 12, 2018, doi: 10.3389/fnins.2018.00774.

[165] A. Tavanaei, M. Ghodrati, S. R. Kheradpisheh, T. Masquelier, and A. Maida, “Deep learning in spiking neural networks,” 2019. doi: 10.1016/j.neunet.2018.12.002.

[166] F. Ponulak and A. Kasiński, “Introduction to spiking neural networks: Information processing, learning and applications,” *Acta Neurobiol. Exp. (Wars).*, vol. 71, no. 4, 2011.

[167] A. Grüning and S. M. Bohte, “Spiking neural networks: Principles and challenges,” in *22nd European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning, ESANN 2014 - Proceedings*, 2014.

[168] A. L. Hodgkin and A. F. Huxley, “A quantitative description of membrane current and its application to conduction and excitation in nerve,” *J. Physiol.*, vol. 117, no. 4, 1952, doi: 10.1113/jphysiol.1952.sp004764.

[169] W. M. Kistler, W. Gerstner, and J. L. Van Hemmen, “Reduction of the Hodgkin-Huxley Equations to a Single-Variable Threshold Model,” *Neural Comput.*, vol. 9, no. 5, 1997, doi: 10.1162/neco.1997.9.5.1015.

[170] A. Delorme, J. Gautrais, R. Van Rullen, and S. Thorpe, “SpikeNET: A simulator for modeling large networks of integrate and fire neurons,” *Neurocomputing*, vol. 26–27, 1999, doi: 10.1016/S0925-2312(99)00095-8.

[171] R. Jolivet, T. J. Lewis, and W. Gerstner, “The spike response model: A framework to predict neuronal spike trains,” *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, vol. 2714, 2003, doi: 10.1007/3-540-44989-2_101.

[172] E. M. Izhikevich, “Simple model of spiking neurons,” 2003. doi: 10.1109/TNN.2003.820440.

[173] P. König, A. K. Engel, and W. Singer, “Integrator or coincidence detector? The role of the cortical neuron revisited,” *Trends Neurosci.*, vol. 19, no. 4, 1996, doi: 10.1016/S0166-2236(96)80019-1.

[174] M. Häusser, “The hodgkin-huxley theory of the action potential,” *Nat. Neurosci.*, vol. 3, no. 11s, 2000, doi: 10.1038/81426.

[175] L. J. Colwell and M. P. Brenner, “Action potential initiation in the Hodgkin-Huxley model,” *PLoS Comput. Biol.*, vol. 5, no. 1, 2009, doi: 10.1371/journal.pcbi.1000265.

[176] J. Ma and J. Tang, “A review for dynamics in neuron and neuronal network,” 2017. doi: 10.1007/s11071-017-3565-3.

[177] M. J. Skocik and L. N. Long, “On the capabilities and computational costs of neuron models,” *IEEE Trans. Neural Networks Learn. Syst.*, vol. 25, no. 8, 2014, doi: 10.1109/TNNLS.2013.2294016.

[178] A. A. Abusnaina and R. Abdullah, “Spiking Neuron Models: A Review,” *Int. J. Digit. Content Technol. its Appl.*, vol. 8, no. 3, 2014.

[179] E. M. Izhikevich, “Which model to use for cortical spiking neurons?,” *IEEE Trans. Neural Networks*, vol. 15, no. 5, 2004, doi: 10.1109/TNN.2004.832719.

[180] J. Vreeken, “Spiking neural networks , an introduction,” *Computing*, vol. 7, no. 3, 2002.

[181] D. Sterratt, B. Graham, A. Gillies, and D. Willshaw, *Principles of computational modelling in neuroscience*. 2011. doi: 10.1017/CBO9780511975899.

[182] E. Nordlie, T. Tetzlaff, and G. T. Einevoll, “Rate dynamics of leaky integrate-and-fire neurons with strong synapses,” *Front. Comput. Neurosci.*, vol. 4, 2010, doi: 10.3389/fncom.2010.00149.

[183] A. N. Burkitt, “A review of the integrate-and-fire neuron model: I. Homogeneous synaptic input,” *Biol. Cybern.*, vol. 95, no. 1, 2006, doi: 10.1007/s00422-006-0068-6.

[184] J. Feng, “Is the integrate-and-fire model good enough? - A review,” 2001. doi: 10.1016/S0893-6080(01)00074-0.

[185] G. D. Smith, C. L. Cox, S. M. Sherman, and J. Rinzel, “Fourier analysis of sinusoidally driven thalamocortical relay neurons and a minimal integrate-and-fire-or-burst model,” *J. Neurophysiol.*, vol. 83, no. 1, 2000, doi: 10.1152/jn.2000.83.1.588.

[186] E. M. Izhikevich, “Resonate-and-fire neurons,” *Neural Networks*, vol. 14, no. 6–7, 2001, doi: 10.1016/S0893-6080(01)00078-8.

[187] P. E. Latham, B. J. Richmond, P. G. Nelson, and S. Nirenberg, “Intrinsic dynamics in neuronal networks. I. Theory,” *J. Neurophysiol.*, vol. 83, no. 2, 2000, doi: 10.1152/jn.2000.83.2.808.

[188] M. S. de Queiroz, R. C. de Berrêdo, and A. de Pádua Braga, “Reinforcement learning of a simple control task using the spike response model,” *Neurocomputing*, vol. 70, no. 1–3, 2006, doi: 10.1016/j.neucom.2006.07.002.

[189] Q. Yu, H. Tang, K. C. Tan, and H. Yu, “A brain-inspired spiking neural network model with temporal encoding and learning,” *Neurocomputing*, vol. 138, 2014, doi: 10.1016/j.neucom.2013.06.052.

[190] Thorpe, “Thorpe, S. J. (1990). Spike arrival times: A highly efficient coding scheme for neural networks. In R. Eckmiller, G. Hartmann & G. Hauske (Eds.), Parallel processing in neural systems and computers (pp. 91-94): North-Holland Elsevier.,” *Parallel Process. neural Syst. Comput.*, no. January 1990, pp. 91–94, 1990.

[191] A. Pouget, P. Dayan, and R. Zemel, “Information processing with population codes,” *Nat. Rev. Neurosci.*, vol. 1, no. 2, 2000, doi: 10.1038/35039062.

[192] D. S. Jeong, “Tutorial: Neuromorphic spiking neural networks for temporal learning,” *J. Appl. Phys.*, vol. 124, no. 15, 2018, doi: 10.1063/1.5042243.

[193] E. D. Adrian and Y. Zotterman, “The impulses produced by sensory nerve-endings: Part II. The response of a Single End-Organ,” *J. Physiol.*, vol. 61, no. 2, 1926, doi: 10.1113/jphysiol.1926.sp002281.

[194] L. F. Abbott, “Theoretical Neuroscience Rising,” 2008. doi: 10.1016/j.neuron.2008.10.019.

[195] D. A. Butts *et al.*, “Temporal precision in the neural code and the timescales of natural vision,” *Nature*, vol. 449, no. 7158, 2007, doi: 10.1038/nature06105.

[196] and D. W. W. Bialek, F. Rieke, R. de Ruyter van Steveninck, “Science 252,” p. 1854, 1991.

[197] L. Kostal, P. Lansky, and J. P. Rospars, “Neuronal coding and spiking randomness,” 2007. doi: 10.1111/j.1460-9568.2007.05880.x.

[198] F. Rieke, D. Warland, R. De Ruyter Van Steveninck, and W. Bialek, “Spikes: Exploring the Neural Code,” 1997.

[199] S. Wu, S. I. Amari, and H. Nakahara, “Population Coding and Decoding in a Neural Field: A Computational Study,” *Neural Comput.*, vol. 14, no. 5, 2002, doi: 10.1162/089976602753633367.

[200] T. J. Gawne and B. J. Richmond, “How independent are the messages carried by adjacent inferior temporal cortical neurons,” *J. Neurosci.*, vol. 13, no. 7, 1993, doi: 10.1523/jneurosci.13-07-02758.1993.

[201] J. L. Puchalla, E. Schneidman, R. A. Harris, and M. J. Berry, “Redundancy in the population code of the retina,” *Neuron*, vol. 46, no. 3, 2005, doi: 10.1016/j.neuron.2005.03.026.

[202] N. S. Narayanan, E. Y. Kimchi, and M. Laubach, “Redundancy and synergy of neuronal ensembles in motor cortex,” *J. Neurosci.*, vol. 25, no. 17, 2005, doi: 10.1523/JNEUROSCI.4697-04.2005.

[203] E. Schneidman, M. J. Berry, R. Segev, and W. Bialek, “Weak pairwise correlations imply strongly correlated network states in a neural population,” *Nature*, vol. 440, no. 7087, 2006, doi: 10.1038/nature04701.

[204] P. Földiák, “Forming sparse representations by local anti-Hebbian learning,” *Biol. Cybern.*, vol. 64, no. 2, 1990, doi: 10.1007/BF02331346.

[205] N. Gupta and M. Stopfer, “A temporal channel for information in sparse sensory coding,” *Curr. Biol.*, vol. 24, no. 19, 2014, doi: 10.1016/j.cub.2014.08.021.

[206] S. R. Kheradpisheh, M. Ganjtabesh, and T. Masquelier, “Bio-inspired unsupervised learning of visual features leads to robust invariant object recognition,” *Neurocomputing*, vol. 205, 2016, doi: 10.1016/j.neucom.2016.04.029.

[207] B. Zhao, R. Ding, S. Chen, B. Linares-Barranco, and H. Tang, “Feedforward Categorization on AER Motion Events Using Cortex-Like Features in a Spiking Neural Network,” *IEEE Trans. Neural Networks Learn. Syst.*, vol. 26, no. 9, 2015, doi: 10.1109/TNNLS.2014.2362542.

[208] S. K. Esser *et al.*, “Convolutional networks for fast, energy-efficient neuromorphic computing,” *Proc. Natl. Acad. Sci. U. S. A.*, vol. 113, no. 41, 2016, doi: 10.1073/pnas.1604850113.

[209] E. Rueckert, D. Kappel, D. Tanneberg, D. Pecevski, and J. Peters, “Recurrent Spiking Networks Solve Planning Tasks,” *Sci. Rep.*, vol. 6, 2016, doi: 10.1038/srep21142.

[210] R. Pyle and R. Rosenbaum, “Spatiotemporal Dynamics and Reliable Computations in Recurrent Spiking Neural Networks,” *Phys. Rev. Lett.*, vol. 118, no. 1, 2017, doi: 10.1103/PhysRevLett.118.018103.

[211] W. Zhang and P. Li, “Spike-train level backpropagation for training deep recurrent spiking neural networks,” in *Advances in Neural Information Processing Systems*, 2019.

[212] N. Caporale and Y. Dan, “Spike timing-dependent plasticity: A Hebbian learning rule,” 2008. doi: 10.1146/annurev.neuro.31.060407.125639.

[213] H. Markram, W. Gerstner, and P. J. Sjöström, “A history of spike-timing-dependent plasticity,” 2011. doi: 10.3389/fnsyn.2011.00004.

[214] T. Masquelier, R. Guyonneau, and S. J. Thorpe, “Competitive STDP-based spike pattern learning,” 2009. doi: 10.1162/neco.2008.06-08-804.

[215] A. Coates, H. Lee, and A. Y. Ng, “An analysis of single-layer networks in unsupervised feature learning,” in *Journal of Machine Learning Research*, 2011.

[216] S. R. Kheradpisheh, M. Ganjtabesh, S. J. Thorpe, and T. Masquelier, “STDP-based spiking deep convolutional neural networks for object recognition,” *Neural Networks*, vol. 99, 2018, doi: 10.1016/j.neunet.2017.12.005.

[217] S. M. Bohte, J. N. Kok, and H. La Poutré, “Error-backpropagation in temporally encoded networks of spiking neurons,” 2002. doi: 10.1016/S0925-2312(01)00658-0.

[218] J. H. Lee, T. Delbruck, and M. Pfeiffer, “Training deep spiking neural networks using backpropagation,” *Front. Neurosci.*, vol. 10, no. NOV, 2016, doi: 10.3389/fnins.2016.00508.

[219] H. Mostafa, “Supervised learning based on temporal coding in spiking neural networks,” *IEEE Trans. Neural Networks Learn. Syst.*, vol. 29, no. 7, 2018, doi: 10.1109/TNNLS.2017.2726060.

[220] F. Ponulak, “Supervised learning in spiking neural networks with ReSuMe method,” *Int. Jt. Conf. Neural Networks*, vol. 3, no. 2, 2017.

[221] A. Kasiński and F. Ponulak, “Comparison of supervised learning methods for spike time coding in spiking neural networks,” 2006.

[222] A. Mohemmed, S. Schliebs, S. Matsuda, and N. Kasabov, “Span: Spike pattern association neuron for learning spatio-temporal spike patterns,” *Int. J. Neural Syst.*, vol. 22, no. 4, 2012, doi: 10.1142/S0129065712500128.

[223] S. B. Shrestha and G. Orchard, “Slayer: Spike layer error reassignment in time,” in *Advances in Neural Information Processing Systems*, 2018.

[224] G. E. Hinton, N. Srivastava, and K. Swersky, “Lecture 6a- overview of mini-batch gradient descent,” *COURSERA Neural Networks Mach. Learn.*, 2012.

[225] Y. Dan and M. M. Poo, “Spike timing-dependent plasticity: From synapse to perception,” 2006. doi: 10.1152/physrev.00030.2005.

[226] D. A. Nguyen, X. T. Tran, and F. Iacopi, “A review of algorithms and hardware implementations for spiking neural networks,” 2021. doi: 10.3390/jlpea11020023.

[227] G. G. Turrigiano and S. B. Nelson, “Homeostatic plasticity in the developing nervous system,” 2004. doi: 10.1038/nrn1327.

[228] A. Artola, S. Bröcher, and W. Singer, “Different voltage-dependent thresholds for inducing long-term depression and long-term potentiation in slices of rat visual cortex,” *Nature*, vol. 347, no. 6288, 1990, doi: 10.1038/347069a0.

[229] C. Clopath, L. Büsing, E. Vasilaki, and W. Gerstner, “Connectivity reflects coding: A model of voltage-based spike-timing-dependent-plasticity with homeostasis,” *Nat. Preced.*, 2009, doi: 10.1038/npre.2009.3362.1.

[230] E. Vasilaki and M. Giugliano, “Emergence of connectivity motifs in networks of model neurons with short- and long-term plastic synapses,” *PLoS One*, vol. 9, no. 1, 2014, doi: 10.1371/journal.pone.0084626.

[231] J. Kim, H. Kim, S. Huh, J. Lee, and K. Choi, “Deep neural networks with weighted spikes,” *Neurocomputing*, vol. 311, 2018, doi: 10.1016/j.neucom.2018.05.087.

[232] P. U. Diehl, D. Neil, J. Binas, M. Cook, S. C. Liu, and M. Pfeiffer, “Fast-classifying, high-accuracy spiking deep networks through weight and threshold balancing,” in *Proceedings of the International Joint Conference on Neural Networks*, 2015. doi: 10.1109/IJCNN.2015.7280696.

[233] A. Sengupta, Y. Ye, R. Wang, C. Liu, and K. Roy, “Going Deeper in Spiking Neural Networks: VGG and Residual Architectures,” *Front. Neurosci.*, vol. 13, 2019, doi: 10.3389/fnins.2019.00095.

[234] J. A. Pérez-Carrasco *et al.*, “Mapping from frame-driven to frame-free event-driven vision systems by low-rate rate coding and coincidence processing - Application to feedforward convnets,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 35, no. 11, 2013, doi: 10.1109/TPAMI.2013.71.

[235] J. Wu, E. Yılmaz, M. Zhang, H. Li, and K. C. Tan, “Deep Spiking Neural Networks for Large Vocabulary Automatic Speech Recognition,” *Front. Neurosci.*, vol. 14, 2020, doi: 10.3389/fnins.2020.00199.

[236] T. Iakymchuk, A. Rosado-Muñoz, J. F. Guerrero-Martínez, M. Bataller-Mompeán, and J. V. Francés-Víllora, “Simplified spiking neural network architecture and STDP learning algorithm applied to image classification,” *Eurasip J. Image Video Process.*, vol. 2015, no. 1, 2015, doi: 10.1186/s13640-015-0059-4.

[237] A. K. Mukhopadhyay, I. Chakrabarti, and M. Sharad, “Classification of Hand Movements by Surface Myoelectric Signal Using Artificial-Spiking Neural Network Model,” in *Proceedings of IEEE Sensors*, 2018. doi: 10.1109/ICSENS.2018.8589757.

[238] H. Chen, G. Ma, P. Wang, and X. Wang, “A Bio-Impedance Analysis Method Based on Human Hand Anatomy for Hand Gesture Recognition,” *IEEE Trans. Instrum. Meas.*, vol. 70, 2021, doi: 10.1109/TIM.2021.3112775.

[239] A. Amir *et al.*, “A low power, fully event-based gesture recognition system,” in *Proceedings - 30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017*, 2017. doi: 10.1109/CVPR.2017.781.

[240] E. Ceolini *et al.*, “Hand-Gesture Recognition Based on EMG and Event-Based Camera Sensor Fusion: A Benchmark in Neuromorphic Computing,” *Front. Neurosci.*, vol. 14, 2020, doi: 10.3389/fnins.2020.00637.

[241] S. B. Furber, F. Galluppi, S. Temple, and L. A. Plana, “The SpiNNaker project,” *Proc. IEEE*, vol. 102, no. 5, 2014, doi: 10.1109/JPROC.2014.2304638.

[242] K. Simonyan and A. Zisserman, “Very deep convolutional networks for large-scale image recognition,” in *3rd International Conference on Learning Representations, ICLR 2015 - Conference Track Proceedings*, 2015.

[243] L. J. Ba and B. Frey, “Adaptive dropout for training deep neural networks,” in *Advances in Neural Information Processing Systems*, 2013.

[244] D. P. Kingma and J. L. Ba, “Adam: A method for stochastic optimization,” in *3rd International Conference on Learning Representations, ICLR 2015 - Conference Track Proceedings*, 2015.

[245] X. Zhai, B. Jelfs, R. H. M. Chan, and C. Tin, “Self-recalibrating surface EMG pattern recognition for neuroprosthesis control based on convolutional neural network,” *Front. Neurosci.*, vol. 11, no. JUL, 2017, doi: 10.3389/fnins.2017.00379.

[246] M. R. Ahsan, M. I. Ibrahimy, and O. O. Khalifa, “Electromygraphy (EMG) signal based hand gesture recognition using artificial neural network (ANN),” in *2011 4th International Conference on Mechatronics: Integrated Engineering for Industrial and Societal Development, ICOM’11 - Conference Proceedings*, 2011. doi: 10.1109/ICOM.2011.5937135.

[247] U. Sahin and F. Sahin, “Pattern recognition with surface EMG signal based wavelet transformation,” in *Conference Proceedings - IEEE International Conference on Systems, Man and Cybernetics*, 2012. doi: 10.1109/ICSMC.2012.6377717.

[248] X. Chen and Z. J. Wang, “Pattern recognition of number gestures based on a wireless surface EMG system,” *Biomed. Signal Process. Control*, vol. 8, no. 2, 2013, doi: 10.1016/j.bspc.2012.08.005.

[249] U. Côté-Allard, C. L. Fall, A. Campeau-Lecoursy, C. Gosseliny, F. Laviolettez, and B. Gosselin, “Transfer learning for sEMG hand gestures recognition using convolutional neural networks,” in *2017 IEEE International Conference on Systems, Man, and Cybernetics, SMC 2017*, 2017. doi: 10.1109/SMC.2017.8122854.

[250] W. Ke, Y. Xing, G. Di Caterina, L. Petropoulakis, and J. Soraghan, “Intersected EMG Heatmaps and Deep Learning Based Gesture Recognition,” in *ACM International Conference Proceeding Series*, 2020. doi: 10.1145/3383972.3383982.

[251] W. Ke, Y. Xing, G. DI Caterina, L. Petropoulakis, and J. Soraghan, “Deep Convolutional Spiking Neural Network Based Hand Gesture Recognition,” in *Proceedings of the International Joint Conference on Neural Networks*, 2020. doi: 10.1109/IJCNN48605.2020.9207040.

[252] A. Krizhevsky, I. Sutskever, and G. E. Hinton, “ImageNet classification with deep convolutional neural networks,” *Commun. ACM*, vol. 60, no. 6, 2017, doi: 10.1145/3065386.

[253] E. O. Neftci, H. Mostafa, and F. Zenke, “Surrogate Gradient Learning in Spiking Neural Networks: Bringing the Power of Gradient-based optimization to spiking neural networks,” *IEEE Signal Process. Mag.*, vol. 36, no. 6, 2019, doi: 10.1109/MSP.2019.2931595.

[254] F. Ertam, “Data classification with deep learning using tensorflow,” in *2nd International Conference on Computer Science and Engineering, UBMK 2017*, 2017. doi: 10.1109/UBMK.2017.8093521.

[255] Y. Du *et al.*, “Semi-supervised learning for surface EMG-based gesture recognition,” in *IJCAI International Joint Conference on Artificial Intelligence*, 2017. doi: 10.24963/ijcai.2017/225.

[256] A. Sun, X. Chen, M. Xu, X. Zhang, and X. Chen, “Feasibility study on the application of a spiking neural network in myoelectric control systems,” *Front. Neurosci.*, vol. 17, 2023, doi: 10.3389/FNINS.2023.1174760.

[257] J. Chen, S. Bi, G. Zhang, and G. Cao, “High-density surface emg-based gesture recognition using a 3d convolutional neural network,” *Sensors (Switzerland)*, vol. 20, no. 4, 2020, doi: 10.3390/s20041201.

[258] A. Chahid, R. Khushaba, A. Al-Jumaily, and T. M. Laleg-Kirati, “A Position Weight Matrix Feature Extraction Algorithm Improves Hand Gesture Recognition,” in *Proceedings of the Annual International Conference of the IEEE Engineering in Medicine and Biology Society, EMBS*, 2020. doi: 10.1109/EMBC44109.2020.9176097.

[259] S. Hao, R. Wang, Y. Wang, and Y. Li, “A spatial attention based convolutional neural network for gesture recognition with HD-sEMG signals,” in *2020 IEEE International Conference on E-Health Networking, Application and Services, HEALTHCOM 2020*, 2021. doi: 10.1109/HEALTHCOM49281.2021.9399004.

[260] Y. Li, Q. Zhang, N. Zeng, J. Chen, and Q. Zhang, “Discrete Hand Motion Intention Decoding Based on Transient Myoelectric Signals,” *IEEE Access*, vol. 7, 2019, doi: 10.1109/ACCESS.2019.2923455.

[261] Z. Yu, J. Zhao, Y. Wang, L. He, and S. Wang, “Surface emg-based instantaneous hand gesture recognition using convolutional neural network with the transfer learning method,” *Sensors*, vol. 21, no. 7, 2021, doi: 10.3390/s21072540.

[262] K. Wang, Y. Chen, Y. Zhang, X. Yang, and C. Hu, “Multi-Source Integration based Transfer Learning Method for Cross-User sEMG Gesture Recognition,” in *Proceedings of the International Joint Conference on Neural Networks*, 2022. doi: 10.1109/IJCNN55064.2022.9892711.

[263] S. Padhy, “A Tensor-Based Approach Using Multilinear SVD for Hand Gesture Recognition from sEMG Signals,” *IEEE Sens. J.*, vol. 21, no. 5, 2021, doi: 10.1109/JSEN.2020.3042540.

[264] A. De Silva, M. V. Perera, K. Wickramasinghe, A. M. Naim, T. Dulantha Lalitharatne, and S. L. Kappel, “Real-Time Hand Gesture Recognition Using Temporal Muscle Activation Maps of Multi-Channel Semg Signals,” *ICASSP, IEEE Int. Conf. Acoust. Speech Signal Process. - Proc.*, vol. 2020-May, pp. 1299–1303, May 2020, doi: 10.1109/ICASSP40776.2020.9054227.

[265] N. Leroux, J. Finkbeiner, and E. Neftci, “Online Transformers with Spiking Neurons for Fast Prosthetic Hand Control,” *BioCAS 2023 - 2023 IEEE Biomed. Circuits Syst. Conf. Conf. Proc.*, 2023, doi: 10.1109/BIOCAS58349.2023.10388996.

[266] H. L. Najafi, D. W. Moses, C. H. Hustig, and J. Kinne, “Neural network based dynamic reconstruction filter for digital audio signals,” *Int. Conf. Knowledge-Based Intell. Electron. Syst. Proceedings, KES*, vol. 2, no. May, pp. 642–647, 1997, doi: 10.1109/kes.1997.619448.

[267] A. Uncini, “Audio signal processing by neural networks,” *Neurocomputing*, vol. 55, no. 3–4, pp. 593–625, 2003, doi: 10.1016/S0925-2312(03)00395-3.

[268] S. Vigneshwaran, M. ShifaFathima, V. VijaySagar, and R. Sree Arshika, “Hand Gesture Recognition and Voice Conversion System for Dump People,” in *2019 5th International Conference on Advanced Computing and Communication Systems, ICACCS 2019*, 2019. doi: 10.1109/ICACCS.2019.8728538.