Machine Learning for Hand Gesture Classification from Surface Electromyography Signals

A thesis submitted to the University of Sheffield for the degree of Doctor of Philosophy

Adam Hartwell

Department of Automatic Control and Systems Engineering

July 2019

For the spirit of discovery

Abstract

Classifying hand gestures from Surface Electromyography (sEMG) is a process which has applications in human-machine interaction, rehabilitation and prosthetic control. Reduction in the cost and increase in the availability of necessary hardware over recent years has made sEMG a more viable solution for hand gesture classification. The research challenge is the development of processes to robustly and accurately predict the current gesture based on incoming sEMG data.

This thesis presents a set of methods, techniques and designs that improve upon evaluation of, and performance on, the classification problem as a whole. These are brought together to set a new baseline for the potential classification.

Evaluation is improved by careful choice of metrics and design of cross-validation techniques that account for data bias caused by common experimental techniques. A landmark study is re-evaluated with these improved techniques, and it is shown that data augmentation can be used to significantly improve upon the performance using conventional classification methods.

A novel neural network architecture and supporting improvements are presented that further improve performance and is refined such that the network can achieve similar performance with many fewer parameters than competing designs. Supporting techniques such as subject adaptation and smoothing algorithms are then explored to improve overall performance and also provide more nuanced trade-offs with various aspects of performance, such as incurred latency and prediction smoothness.

A new study is presented which compares the performance potential of medical grade electrodes and a low-cost commercial alternative showing that for a modest-sized gesture set, they can compete. The data is also used to explore data labelling in experimental design and to evaluate the numerous aspects of performance that must be traded off.

Acknowledgements

I'd first like to express my thanks to my supervisor Dr. Sean Anderson. His guidance has been invaluable in improving not just my research but also improving my own skill set, particularly with regards to writing. His enthusiasm was critical for helping me achieve this culmination of my work and helped motivate me during times where I struggled.

I also want to thank my second supervisor Professor Visakan Kadirkamanathan for his useful advice, suggestions and insight on the varied topics and problems I've consulted with him about. I'd also like to thank him for cajoling me into taking a more active part in our research community and giving me the opportunity to do so.

I would like to thank my partner Dawn for her love and support and always knowing how to put a smile on my face when I needed it.

I'd like to thank the numerous other researchers and colleagues who gave me the opportunity to discuss interesting problems and gave me new insight or perspective. I'm also thankful for the opportunities given to me by same to teach and present which have been both fulfilling and a perfect opportunity to solidify my own knowledge base.

My friends, I thank for keeping me sane and reminding me to keep things in balance.

Lastly, I'd like to thank my parents Stephen and Ewa for their love, patience and understanding as well as their support, willingness to debate me on my ideas and instilling in me a desire to learn.

Contents

Abbreviations **xi**

Nomenclature **xiv**

1 Introduction **1**
1.1 Background 1
1.2 Aims and Objectives 3
1.3 Thesis Structure 4
1.4 Associated Publications 5

2 Literature Review **7**
2.1 Electromyography 7
2.1.1 Biological Basis 7
2.1.2 Characteristics of sEMG 9
2.1.3 Signal Processing Issues 9
2.1.4 Summarised History of EMG 10
2.1.5 Electrode Types 10
2.1.6 Skin Preparation and Electrode Placement 11
2.1.7 Applications of sEMG 12
2.1.8 Myo Armband 13
2.2 Machine Learning Overview 14
2.2.1 Feature Engineering 14
2.3 Neural Networks and Deep Learning 15
2.3.1 No Free Lunch Theorems 16
2.3.2 The Neuron Model 16
2.3.3 Training and Backpropagation 17
2.3.4 Activation Functions 23
2.3.5 Network Architectural Choices 28
2.4 Hand Movement Classification 34
2.4.1 Deep Learning Approaches 37

2.4.2 Key Issues 38

3 Methods **40**
3.1 Statistics for Comparison of Techniques 40
3.1.1 Comparison of Two Techniques 41
3.1.2 Comparison of Multiple Techniques 41
3.2 Machine Learning Algorithms 43
3.2.1 Support Vector Machines 43
3.2.2 K Nearest Neighbours 46
3.2.3 Hidden Markov Models 46
3.3 Evaluation Metrics 49
3.3.1 Binary Case 49
3.3.2 Multiclass Case 50
3.4 Data Labelling 52

4 Robust Feature-Based Classification **55**
4.1 Introduction 55
4.2 Data Analysis 57
4.3 Robust Preprocessing and Evaluation 61
4.3.1 Windowing 61
4.3.2 Metrics 65
4.3.3 Validation 67
4.4 Benchmark Details 69
4.4.1 Preprocessing 69
4.4.2 Features 70
4.4.3 Classifiers 71
4.4.4 Validation and Metrics 72
4.4.5 Meta-Validation 73
4.4.6 Data Resampling and Augmentation Techniques 74
4.4.7 Primary Benchmark Variants 75
4.4.8 Person-specific Movement Set Selection 76
4.5 Results and Discussion 79
4.5.1 Benchmark Results 79
4.5.2 Metric Comparison 79
4.5.3 Benchmark Performance Breakdown 81
4.5.4 Feature Differences 84
4.5.5 Comparison of Window Lengths 86
4.5.6 Classifier Trends 87
4.5.7 Meta-Validation Results 89

4.5.8 Movement Sub-selection Results . . . 90
4.6 Conclusion . . . 93

5 Deep Neural Networks for Person-Specific Classification 94
5.1 Introduction . . . 94
5.2 Methodology . . . 95
5.2.1 Baseline SVM-RBF . . . 97
5.2.2 Baseline CNN . . . 99
5.2.3 Temporal-to-Spatial Network . . . 101
5.2.4 Additional Design Choices . . . 103
5.2.5 Adam Algorithm . . . 105
5.2.6 Comparison to Contemporary Networks . . . 106
5.2.7 Filter Visualisation . . . 107
5.3 Results and Discussion . . . 108
5.3.1 Effect of Repetition Number . . . 110
5.3.2 Distribution of Classification Performance . . . 111
5.3.3 Filter Visualisations . . . 112
5.4 Conclusion . . . 113

6 Compact Deep Neural Networks with Comparison of Electrodes 119
6.1 Introduction . . . 119
6.2 Methods . . . 120
6.2.1 Experiment Overview . . . 120
6.2.2 Experiment Protocols . . . 122
6.2.3 Experiment Software . . . 126
6.2.4 Experiment Extensions . . . 127
6.2.5 Electrode Comparison . . . 127
6.2.6 Data Preprocessing . . . 128
6.2.7 Performance Baselines . . . 129
6.2.8 Compact Deep Neural Network . . . 130
6.2.9 Hardware Performance Comparison . . . 131
6.3 Results and Discussion . . . 132
6.3.1 Key Findings . . . 132
6.3.2 Performance on Hardware . . . 135
6.3.3 GLR vs Hold vs Expert Labelling . . . 136
6.4 Conclusion . . . 139

7 Online Classification **141**
7.1 Introduction . . . 141
7.2 Methods . . . 142
7.2.1 Improving Performance with Transfer Learning . . . 142
7.2.2 Motivation for Smoothing Predictions . . . 145
7.2.3 Latch Algorithm . . . 148
7.2.4 Majority Voting Algorithm . . . 148
7.2.5 The MSPRT Algorithm . . . 149
7.2.6 Smoothing with Hidden Markov Models . . . 149
7.2.7 Viterbi Algorithm . . . 150
7.3 Results and Discussion . . . 151
7.3.1 Subject Adaptation . . . 151
7.3.2 Prediction Smoothing . . . 153
7.4 Integration of Techniques . . . 159
7.5 Conclusion . . . 160

8 Closing Remarks **163**
8.1 Summary and Conclusions . . . 163
8.2 Future Research Avenues . . . 165

Bibliography **167**

List of Figures

2.1 Illustration of the formation of an EMG signal 8
2.2 Ideal Bipolar Electrode Placement 12
2.3 The Myo Armband 13
2.4 Biological Neuron and Neural Network Neuron 17
2.5 Visualisation of Chain Rule 19
2.6 Comparison of Momentum and Nesterov Momentum 21
2.7 Dropout Layer Visualisation 24
2.8 Logistic Sigmoid Activation 25
2.9 Tanh Activation 26
2.10 Rectified Linear Unit Activation 27
2.11 Basic Neural Network 28
2.12 Visualisation of Convolutional Layer 31
2.13 Visualisation of Stride 32
2.14 Pooling Layer Visualisation 34
2.15 LSTM and GRU Visualisation 35

3.1 SVM Mapping and Decision Boundary 45
3.2 HMM Visualisation as Bayesian Network 47
3.3 Hidden Markov Model 48

4.1 Illustration of Data Imbalance 59
4.2 PCA Visualisation 60
4.3 t-SNE Visualisations 62
4.4 Windowing Visualisation 63
4.5 Visualisation of Windowed sEMG 64
4.6 Histogram of Accuracies from NinaPro Study 66
4.7 Example Comparison of Metrics 83
4.8 Gesture Sub-selection Comparison 91

5.1 Visualisation of X_ω 98

5.2 TtS Network Diagram . . . 101
5.3 Geng et al. Performance Per Class . . . 109
5.4 TtS Performance by Repetition Number . . . 115
5.5 Normalised Position in Movement Against Performance, NinaPro Study . . . 116
5.6 Normalised Position in Movement Against Performance . . . 116
5.7 Visualisation of Temporal Convolution Weights . . . 117
5.8 Visualisation of Temporal Convolution Maximum Activation Inputs 117
5.9 Visualisation of Spatial Reduction Maximum Activation Inputs . . . 118

6.1 Jetson TX2 . . . 120
6.2 Electrode Positioning . . . 123
6.3 Set of Gestures . . . 123
6.4 Normalised Position in Movement Against Performance . . . 126
6.5 Myo v Delsys Per Class Performance . . . 133
6.6 Myo v Delsys Per Subject Performance . . . 133
6.7 Myo v Delsys Performance Subject 1 . . . 134
6.8 Myo v Delsys Performance Subject 2 . . . 135
6.9 Comparison of Gesture Labelling Methods . . . 138

7.1 Classification Flicker . . . 145
7.2 Latency Definitions . . . 146
7.3 Pareto Frontier of Smoothing Algorithms (Onset) . . . 158
7.4 Pareto Frontier of Smoothing Algorithms (Tail) . . . 159
7.5 Comparison of Smoothing Algorithms on Compact TtS . . . 161

List of Tables

3.1 Binary Confusion Matrix 49
3.2 Multiclass Confusion Matrix 51

4.1 Table of Features 70
4.2 Benchmark Training and Test Repetitions 72
4.3 100ms Benchmark Results 80
4.4 200ms Benchmark Results 81
4.5 400ms Benchmark Results 82
4.6 Average Rank Against Set Size 92

5.1 Training and Test Sets 96
5.2 Baseline CNN DB1 99
5.3 Baseline CNN DB2 99
5.4 Baseline Adapted CNN DB2 99
5.5 TtS DB1 102
5.6 TtS DB2 102
5.7 Adapted TtS DB2 102
5.8 Deep Learning Results Database 1 108
5.9 Deep Learning Results Database 2 108

6.1 Study Training, Validation and Test Repetitions 129
6.2 Compact TtS (Myo Data) 131
6.3 Compact TtS (Delsys Data) 131
6.4 Network Performances and Run Time on Jetson TX2 136
6.5 Comparison of Labelling Techniques 137

7.1 Updated TtS 143
7.2 Study Training, Validation and Test Repetitions 143
7.3 Subject Adaptation Results 151
7.4 Subject Adaptation Results When Trained On Individual Repetitions 152

7.5 Supplemental Smoothing Results (Expert Trained, Expert Tested) . . 154
7.6 Supplemental Smoothing Results (GLR Trained, Expert Tested) . . . 154
7.7 Supplemental Smoothing Results (Expert Trained, GLR Tested) . . . 155
7.8 Supplemental Smoothing Results (GLR Trained, GLR Tested) 155
7.9 Cross-Validated Smoothing Results 157

Abbreviations

ANOVA Analysis Of Variance. 42

API Applications Programming Interface. 14

AUC Area Under Curve. 50

CNN Convolutional Neural Network. 30, 32, 33, 99, 113

DSP Digital Signal Processing. 61

DT Decision Tree. 71, 87, 88

DWT Discrete Wavelet Transform. 70

ECG Electrocardiography. 7, 10

ECoG Electrocorticography. 1, 2

EEG Electroencephalography. 1, 103

EM Expectation Maximisation. 149

EMG Electromyography. 1, 7, 9, 10, 14, 23, 34, 36, 37, 53, 58, 61, 63, 65, 67, 68, 69, 70, 76, 101, 156, 166

FFT Fast Fourier Transform. 37

FN False Negative. 49, 50, 51, 65

FP False Positive. 49, 50, 51, 65

GLR Generalised Likelihood Ratio. 53, 57, 61, 120, 125, 127, 137, 138, 139, 140, 147, 153, 154, 155, 156, 157, 159

GPU Graphics Processing Unit. 18, 96, 107, 120

GRU Gated Recurrent Unit. 34, 35

HIST Histogram. 70, 84, 85, 86

HMM Hidden Markov Model. 46, 47, 48, 149, 150, 158

IMU Inertial Measurement Unit. 14

JSON Javascript Object Notation. 124

KL Kullback-Leibler. 78, 92

KNN K-Nearest Neighbours. 46, 71, 87, 88

L-BFGS Limited-Memory Broyden-Fletcher-Goldfarb-Shanno. 22

LDA Linear Discriminant Analysis. 36, 71, 84, 85, 87, 88

LSTM Long Short-Term Memory. 34, 35, 104

MAE Mean Absolute Error. 19, 20, 74, 89, 90

MAV Mean Absolute Value. 70, 77, 85, 86, 88, 92, 98, 103, 112, 113

mDWT Mariginal Discrete Wavelet Transform. 70, 73, 74, 77, 83, 84, 85, 86, 88, 89, 98, 112, 129

ML Machine Learning. 16

MLP Multilayer Perceptron. 16, 28, 71

MSE Mean Squared Error. 19, 20

MSPRT Multi-Hypothesis Sequential Probability Ratio Test. 149, 157, 158

NinaPro Non-Invasive Adaptive Hand Prosthetics. 35, 36, 37, 38, 57, 65, 66, 95, 106, 111, 125, 131, 156

PCA Principal Component Analysis. 59, 61

RBF Radial Basis Function. 46

ReLU Rectified Linear Unit. 26, 27, 28

RF Random Forest. 36

RMS Root Mean Squared. 57, 69

ROC Receiver Operating Characteristic. 50, 52

sEMG Surface Electromyography. i, 1, 2, 3, 4, 7, 8, 9, 10, 11, 12, 13, 14, 15, 35, 36, 37, 43, 52, 55, 57, 64, 89, 93, 94, 98, 99, 100, 103, 104, 106, 112, 137, 139, 140, 141, 145, 151, 159, 160, 162, 163, 164, 165, 166

SGD Stochastic Gradient Descent. 20, 21

SMOTE Synthetic Minority Oversampling Technique. 74, 75, 81, 82, 83, 84, 85, 89, 100, 132, 159, 160

SVM Support Vector Machine. vii, 36, 42, 43, 44, 113, 116

SVM-L Support Vector Machine - Linear Kernel. 71, 87, 88

SVM-RBF Support Vector Machine - Radial Basis Function Kernel. 45, 71, 73, 74, 83, 87, 88, 89, 97, 98, 108, 129, 132

t-SNE t-Stochastic Neighbour Embedding. 61, 107

TN True Negative. 49, 50, 51, 65

TP True Positive. 49, 50, 51, 65

TtS Temporal-to-Spatial. 101, 102, 108, 109, 110, 111, 112, 113, 115, 116, 117, 118, 119, 120, 126, 129, 130, 131, 132, 133, 134, 135, 136, 137, 139, 140, 143, 144, 159, 160, 161

VAR Variance. 70, 85, 86

WL Waveform Length. 70, 85, 86, 98

Nomenclature

A list of the variables and notation used in this thesis is defined below. The definitions and conventions set here will be observed throughout unless otherwise stated.

$[a, b]$	Closed interval between a and b
$\hat{x}$	Estimate of x
$\in$	Member of operator
$\vee$	Logical "OR" operator
$\mathbb{R}^D$	Real numbers of dimensions D
$\mathcal{L}$	Loss
μ	Mean
σ	Standard deviation
b	Bias term
$P(A)$	Probability of event A
w	Weight term
X	A set of data points
x	A data point
Y	A set of outputs
y	An output

Chapter 1

Introduction

1.1 Background

Surface Electromyography (sEMG) records electrical signals from changes in muscle membrane potential with non-invasive electrodes placed on the skin [1]. These signals have applications in prosthetic control [2, 3], rehabilitation [4] and as a natural control interface for human-machine interaction [5]. Hand gesture classification plays an important role in both human-machine interaction and prosthetic control as a way of determining user intention. Therefore the accurate classification of hand gestures is a vital step towards realising these applications.

This work focuses on framing and solving the hand gesture classification problem with an emphasis on practical considerations. The classification problem is tackled by improving upon current data preparation and evaluation methods, applying novel deep learning methods, exploring methods for subject adaptation and gathering data to support hardware choices.

Gesture-based interaction or control has applications in many areas. This includes entertainment via games consoles such as the Wii [6], Kinect [7] and PlayStation Move [6], control of dashboard utilities in cars [8, 9], operating surgical equipment [10], aiding in rehabilitation [11] and augmenting working environments [12] amongst others.

Electroencephalography (EEG) and Electrocorticography (ECoG) are alternative methods of extracting user intention from the brain directly via electrodes. These methods, however, are not currently practical for routine clinical or personal use [13]. When using EEG or ECoG user intention needs to be extracted from among a multitude of other signals [14]. Meanwhile, Electromyography (EMG) allows focussing on specific groups of muscles as they are manipulated via the nervous system, which improves intention extraction relative to EEG and

ECoG [13].

An alternate way of recognising gestures is by using a camera which has the benefit of being generalisable to tracking any part of the human body but can require markers to be worn for precise tracking [15]. Camera-based approaches suffer from occlusion problems and may be sensitive to environmental issues such as changes in lighting, obstructions or varying distance from the camera. Gloves with flex sensors have also been trialled [16] which alleviate many of the environmental issues of cameras but require substantial hardware adjustment for different users. These two approaches cannot be used with amputees.

In general, the major benefits of sEMG over other methods is that it is not invasive, adaptation to individuals can generally be performed in software, it does not suffer from normal changes in the environment, and it can be used on amputees. Having access to user intention via muscle contraction also allows remapping of muscle contraction and intention for prosthetic control [17], which makes sEMG a versatile data source.

This thesis focuses on the problem of classifying hand gestures from sEMG with a general view towards application as a human-machine interface but presents ideas relevant to any application. It aims to improve the overall performance of gesture classification in as generalisable way as possible.

Studies on hand gesture classification with sEMG generally focus on smaller sets of movements, e.g. 4-7 [18–28] or 9-12 movements [29–35]. Most studies, e.g. [20, 23, 27, 30, 34–40], by necessity, employ *rest*-movement-*rest* cycles as part of their experiment design and label their data as such which often causes significant data imbalance. Typically accuracy is then used to evaluate these experiments [25, 27, 37, 41–43] or "accuracy" / "classification accuracy" is stated without an equation [18, 20, 23, 26, 28, 29, 35, 37–40, 44, 45] which can lead to uncorrected bias in results.

The key challenges are the design of standard robust evaluation methods for ensuring representative performance reporting and comparability between studies, improving the overall performance of classification techniques, tailoring solutions to individual subjects and addressing issues that arise when classifying online.

Therefore the first issue tackled is the improvements that can be made to evaluation methods. It is shown that common experiment design paradigms lead to biased data sets and thus biased results. The macro-average accuracy is presented as a viable alternative metric that avoids the skew the typical accuracy metric introduces. It is combined with proper cross-validation (a technique that is missing in many studies [22–25, 30, 37, 39, 46–49]) and a new stratification technique that

produces a result more representative of the performance that would be found when evaluating all possible folds.

These evaluation techniques are then applied to a reproduction of a landmark open-data study with many hand gestures [37], 52, rather than the typical ≤ 12, to demonstrate the technique's necessity and that effective classification can be achieved with this higher number of gestures. The study is further improved upon via data augmentation techniques which allow a new performance baseline to be established.

In order to further improve overall performance, a novel neural network architecture is then designed. It is shown that this architecture outperforms both the baseline and other neural network approaches to the problem [39, 47]. The network architecture is later refined such that it can produce a similarly high performance as the original while requiring an order of magnitude fewer parameters. The new compact design is evaluated on an embedded system, the Jetson TX2, to demonstrate the run time improvement it produces.

Supporting methods including gesture subselection, subject adaptation, and post-classification smoothing are explored as other viable ways to improve performance and to show the large decision space associated with defining performance outside of only an accuracy measure.

These techniques together present a reliable, robust way of improving overall classification performance and allowing researchers to make informed decisions regarding the trade-offs involved for any particular application. A new study is also presented that compares medical grade electrodes with a low-cost commercial alternative illustrating that the techniques can be used achieve similar performance levels with both helping to make the necessary hardware for classification more accessible.

1.2 Aims and Objectives

This thesis aims to improve sEMG-based gesture classification. Accordingly, the objectives are as follows:

- Reproduce and benchmark existing approaches for gesture classification using conventional classifiers. Improve evaluation and cross-validation to lay the groundwork for robust benchmarking in this thesis and future studies
- Investigate how the number of gestures classified affects performance to establish a realistic number of gestures on which high performance can be achieved and evaluate algorithms for person-specific gesture subselection

- Reproduce, benchmark and extend the design of the current state of the art in deep neural networks for EMG-based gesture classification using state of the art techniques along with robust evaluation and validation
- Extend the current state of the art by designing novel deep neural networks suited to embedded systems that exploit compression techniques and domain knowledge
- Compare new generation low-cost consumer surface EMG electrodes to expensive medical grade electrodes to establish the viability of low-cost electrodes
- Extend the state-of-the-art in online EMG classification by augmenting deep neural network classifiers with smoothing methods to prevent class 'jitter' within the classification signal

1.3 Thesis Structure

This thesis is structured around the central idea of improving sEMG gesture classification:

- Chapters 2 and 3 introduce the theories, techniques, and background this work is built upon. Particular attention is given to machine learning techniques and neural networks. These chapters are intended as a reference that supports the later chapters.
- Chapter 4 proposes a set of fundamental improvements to the classification process that are necessary to ensure that results are reproducible and representative. A landmark study is repeated and shown to have significantly biased results, data resampling and augmentation methods are then applied to significantly improve performance over the original. Lastly, a novel evaluation of gesture sub-selection techniques is presented, which demonstrates the possibility for further performance improvement via patient-specific adaptation outside of the classifiers themselves.
- Chapter 5 introduces a novel neural network architecture that encodes domain knowledge into its design, leading it to significantly outperform contemporary designs. The network is evaluated across two data sets and against the top performing contemporary designs to demonstrate its improvement.

- Chapter 6 introduces a new study that compares high- and low-cost electrodes finding that the low-cost electrodes have the potential to produce a similar or better performance on at least some variants of the classification problem. A new compact version of the network design is also introduced that greatly reduces the number of parameters necessary for similar levels of classification performance. The new study's data is then used to evaluate the run time impact of the parameter reduction, demonstrating real-world run time reduction, which is vital for actual usage. Lastly, different labelling methods are explored to improve upon future experiment designs.
- Chapter 7 explores subject adaptation and online classification signal smoothing in order to further improve resultant classification performance. Subject adaptation results show that it is possible to fine-tune a network trained on other subjects to a new individual by freezing the feature extractor layers in a way that improves performance over training directly on a new subject. A variety of techniques are explored for signal smoothing, and it is shown that most lie on the Pareto frontier between smoothness and added latency, although different techniques have additional benefits and drawbacks.
- Chapter 8 concludes the thesis. The development of improved classification methods and techniques is summarised, and potential future avenues for research are presented.

1.4 Associated Publications

The following papers have resulted from work on this PhD:

- Person-Specific Gesture Set Selection for Optimised Movement Classification from EMG Signals

  - A. Hartwell, V. Kadirkamanathan, and S.R. Anderson. Person-Specific Gesture Set Selection for Optimised Movement Classification from EMG Signals. In *Proceedings of the 38th Annual International Conference of the IEEE Engineering in Medicine and Biology Society*, pages 880–883, 2016

- Compact Deep Neural Networks for Computationally Efficient Gesture Classification From Electromyography Signals

  - A. Hartwell, V. Kadirkamanathan, and S.R. Anderson. Compact Deep Neural Networks for Computationally Efficient Gesture Classification From Electromyography Signals. In *Proceedings of the 7th IEEE RAS/EMBS International Conference on Biomedical Robotics and Biomechatronics*, pages 891–896, 2018

Further papers are under review and being prepared for publication at the time of submission.

The set of tools and utilities developed for this work will be made freely available on GitHub [52], and specific libraries are available via Python's package manager. The associated scripts and data will be available alongside this thesis via the University of Sheffield's White Rose system.

The following papers resulted from collaborations with fellow researchers:

- Multi-Compartmentalisation in the MAPK Signalling Pathway Contributes to the Emergence of Oscillatory Behaviour and to Ultrasensitivity

  - A. Shuaib, A. Hartwell, E. Kiss-Toth, and M. Holcombe. Multi-Compartmentalisation in the MAPK Signalling Pathway Contributes to the Emergence of Oscillatory Behaviour and to Ultrasensitivity. *PLOS ONE*, 11(5):e0156139, 2016

- Top-Down Bottom-Up Visual Saliency for Mobile Robots Using Deep Neural Networks and Task-Independent Feature Maps

  - U. Jaramillo-Avila, A. Hartwell, and S. Anderson. Top-Down Bottom-Up Visual Saliency for Mobile Robots Using Deep Neural Networks and Task-Independent Feature Maps. In *Proceedings of the 19th Annual Conference: Towards Autonomous Robotic Systems*, volume 10965, page 489. 2018

Chapter 2

Literature Review

This chapter covers the necessary background information and relevant literature used to inform later chapters. The key topics covered here are the basis and characteristics of Electromyography (EMG) signals, the machine learning techniques used in this work along with a particular focus on modern neural networks and the current academic landscape of EMG based interaction and control.

2.1 Electromyography

2.1.1 Biological Basis

Electromyography (EMG) is the recording of electrical activity produced by muscle fibres when they contract [55]. Muscles themselves are formed of bundles of muscle fibres which are a form of long tubular cell. The EMG signal is made up of a superposition of the signals from each muscle cell modified by a person's physiology.

In a biomedical context, the recorded EMG waveform is known as an electromyogram and is closely related to other bioelectrical signals such as Electrocardiography (ECG). ECG deals specifically with the signals from heart muscles.

The electrical source for the readings is the change in muscle membrane potential which causes muscle contraction and the associated movement of potassium and calcium ions [1]. Therefore a useful interpretation of the resultant electrical activity is an indirect measurement of movement intention modified by the local physiological and anatomical properties of a subject at the location(s) of interest. The reading is sometimes referred to as the myoelectric signal.

There are two main methods of acquisition for EMG signals: invasive (also referred to as intramuscular) and non-invasive. The non-invasive method is the focus of this thesis and is often referred to as Surface Electromyography (sEMG).

The non-invasive method is also the generally preferred method where possible since it is relatively free of discomfort and presents a much lower risk of infection [56, 57].

Acquisition of sEMG is generally achieved by placing electrodes on the surface of the skin to record the electrical activity. This is in contrast to invasive methods which require placement of electrodes under the skin. Invasive methods allow placement of electrodes closer to specific muscles of interest but are less practical for general usage. Invasive methods have also been shown to produce higher inter-subject variability as well a lower repeatability over extended periods of time [58].

An illustration of the EMG signal in relation to a muscle cell membrane potential change is shown in figure 2.1. The figure shows a rough approximation of how the movement of Sodium and Potassium ions across cell membranes (via active transport) produces the observed electrical potential changes as might be viewed by an electrode.

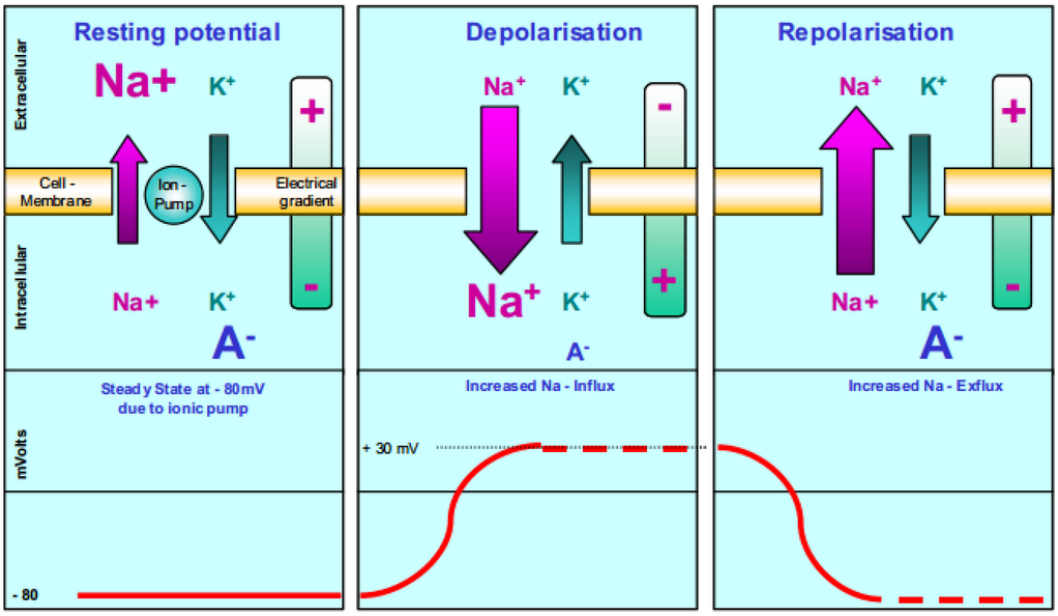

Figure 2.1: Illustration of how an EMG signal is formed by the depolarization and repolarization of the muscle membrane in skeletal muscles. The model is simplified to highlight the important phases [59].

The resting potential of the muscle fibre membrane is ∼-80/-90mV [1, 59] relative to the outside of the cell when not contracted. This difference is maintained by ion pumps, as shown in Figure 2.1, which provide the active transport. Then when alpha-motor anterior horn cell is activated either by reflex or via the central nervous system, the excitation is conducted along the motor nerve leading to an

electrical potential being formed at the motor endplates. This causes the diffusion characteristics of the muscle fibre membrane to change, leading to sodium ion inflow. The membrane, therefore, becomes depolarized, which is quickly reversed via the ion pumps leading to repolarization [59].

2.1.2 Characteristics of sEMG

Figure 2.1 presents a simplified view of the EMG signal itself, which is also more complicated in practice. The key characteristics of sEMG signals are:

- Amplitudes in the μV to low mV range depending on muscle type and condition [60, 61]
- The resting potential is typically $\sim$-80/-90mV [1, 59]
- Two distinct phases: transient followed by a steady state[62]
  - During contraction, voltages may be either positive or negative [55]
- Bandwidth for most significant activity in the range 5-500Hz although other bandwidths such as 20-450Hz are used depending on area and application of interest [59, 63–65]
  - The entire usable range is 0-500Hz [66, 67]
  - The most significant spectrum is 50-150Hz [66]
- May be modelled as a non-stationary stochastic process [55, 66]
- Composite of all the muscle fibre potentials underneath or near each electrode [55]

2.1.3 Signal Processing Issues

From a signal processing standpoint, the major issues that must be considered are:

- Motion artefacts caused by movement of electrode relative to the skin (typically 0-20Hz) [55, 66]
- Quasi-randomness of motor unit firing [55, 66]
- Electrode design (particularly self-interference from circuitry design) [37, 55, 66]
- Electrode placement determines which muscle fibres will be targeted and precise anatomy varies significantly between individuals [68]

- Cross-talk from nearby muscles [56]
- Muscle fatigue alters the action potentials of muscles [68, 69]
- Interference from Electrocardiography (ECG) signals [59] (when near the heart)
- Interference from nearby equipment or power supplies (generally 50-60Hz)

2.1.4 Summarised History of EMG

The term EMG was first used by French scientist Étienne-Jules Mare in 1890 however work on what would now be called EMG is recorded as far back as 1666 when experiments were performed on the muscles of electric ray fish[70]. The beginnings of clinical usage began in 1922 when it was shown that the electrical signals from muscles could be recorded and analysed [71]. The first use of sEMG was in 1966 to monitor the activity of speech muscles [72]. This work was expanded upon in 1983 when a clinical method was introduced that demonstrated sEMG could be used to assess the neuromuscular contribution to pain states [73].

Since 1983 EMG and sEMG have mostly been used in clinical and research settings. Primary uses include robot control [74], identification of neuromuscular disorders [75], rehabilitation [76] and prosthetic control [77].

In late 2014 Thalmic Labs shipped an sEMG device called the Myo Armband [78] which provided a low cost, user-friendly way of gathering and using sEMG data in real time. The low cost of the device means that the electrode (and thus signal) quality is not comparable to medically certified devices. The commercialisation of an sEMG device, however, moved sEMG toward wider spread usage.

2.1.5 Electrode Types

Invasive EMG (intramuscular) data acquisition typically makes use of needle electrodes that use a pointed tip as a detection surface or fine wire electrodes that utilise small diameter wires typically made of a non-oxidising alloy [79]. Application of these kinds of electrodes often requires strict certification and supervision to ensure no harm is done to the subject.

There are two types of electrode typically used when gathering sEMG data: Gelled electrodes and Dry electrodes [80].

Gelled electrodes use an electrolytic gel as an interface between skin and electrode, which acts to minimise electrical noise. In this setup, a redox reaction occurs at the metal junction of the electrodes producing the observed signal.

Gelled electrodes are also manufactured in single-use and reusable variants with the single-use variant being most common. The main drawback of gelled electrodes is the greater need for special skin preparation, such as hair removal and cleaning as well as the need to apply the gel to the subject, which can make usage cumbersome.

Dry electrodes forgo the gel interface and place the metal contact directly on the skin. These electrodes often contain multiple detecting surfaces and integrated pre-amplification/filtering circuitry to improve signal quality. Due to the desirability of local circuitry, dry electrodes are usually heavier than a gel counterpart. This added weight can lead to issues with keeping the electrodes affixed to the subject and in a stable relationship with the muscles. Engineering for platform stability is, therefore, seen as vital.

Furthermore, sEMG sensors may be classified as active or passive. Passive electrodes do not use any filtering/amplification circuitry prior to signal output while active sensors do; therefore, most dry electrodes fall under the active category.

The most common material for the metallic contact of electrodes is silver-silver chloride (Ag-AgCl) which in 2002 was estimated to be used in 80% of sEMG applications [80]. Ag-AgCl is popular because of its relatively low impedance and low half-cell potential [81]. It is considered "the gold standard" material for sEMG electrodes at the time of writing [82].

In general intramuscular electrodes provide better quality and better-targeted recordings since they can be placed closer to specific muscles (including muscles deeper within the body) thus reducing cross-talk and improving discrimination. However, for human-machine interaction purposes, it is highly undesirable to require invasive procedures; therefore, surface electrodes are much better suited to the task.

2.1.6 Skin Preparation and Electrode Placement

Proper skin preparation improves the quality of the sEMG signal received when using any type of surface electrode. Specifically, skin preparation aims to provide impedance matching between skin and electrode. This maximises power transfer between the two by minimising signal reflection. Ideally, all hair and other dead skin cells, as well as moisture, would be removed from the electrode locations. The typical advice is to use an abrasive gel to reduce the dry layer of skin and then clean with alcohol to eliminate moisture [79].

Most surface electrodes utilise two detecting surfaces in a bipolar configuration. In this configuration, the detecting surfaces should be $1-2$cm apart [66]. The ideal placement for this kind of electrode involves adjusting the electrode's longi-

tudinal axis (that which intersects both detection surfaces) parallel to the length of the muscle fibres. The electrode should also be between the motor unit and tendon insertion of the muscle of interest, empirically placing detection surfaces on the belly of the muscle has proven to produce good results; likely due to this being the location of highest muscle fibre density [66]. Figure 2.2 shows this ideal placement; as can be seen in the figure this placement maximises intersection with muscle fibres, which results in an improved signal due to the electrical superposition of the resultant signals.

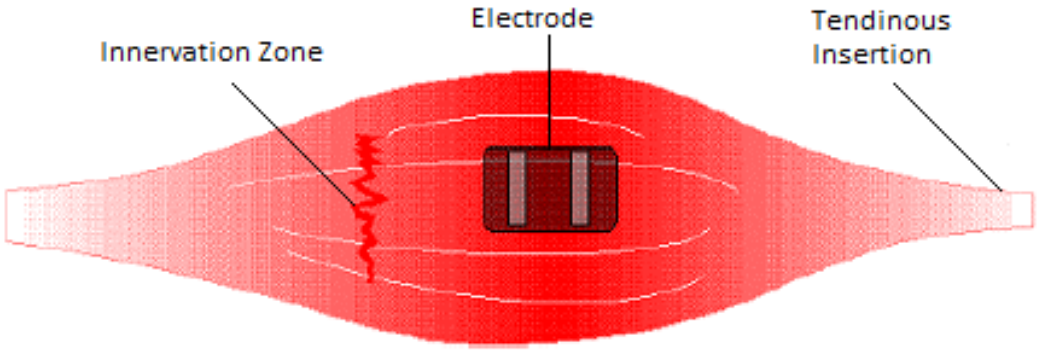

Figure 2.2: Ideal positioning for a bipolar electrode maximising intersection with muscle fibres [66].

2.1.7 Applications of sEMG

Key sEMG applications are:

Human-Computer Interaction, which has the potential to provide a more natural interface for controlling computers and devices. For users with a physical disability, sEMG also has the potential to improve the interaction experience considerably.

Physiotherapy and Rehabilitation, typically an electromyogram can be recorded to evaluate the activity of skeletal muscles that a doctor may analyse to determine whether a patient's muscles are working correctly. Beyond this basic human-in-the-loop use, however, research is being conducted to automate this detection [83] and provide feedback not just to doctors making an assessment but also as part of neuro-rehabilitation where stimulation can be targeted to malfunctioning muscle tissue [84, 85].

Prosthetic Control is a growing area for sEMG [86]. It has the benefit over neural

interfaces that it does not cause neural scarring. Additionally, through techniques such as targeted muscle re-innervation, it is possible for new nerve clusters and muscles to be grown specifically for the control of a prosthetic [17].

Robotic Control, as it is possible to map EMG signals to control of humanoid robots, such as robotic arms, in a natural way. Interaction with robots in this more natural way is important to consider as it reduces the training time for operators. In exoskeleton robotics being able to control a device without the need for a traditional interface has been shown to improve the utility of a system as well as user comfort [87].

Other uses for EMG being explored currently are: facial expression recognition for use in psychological studies and speech recognition without audio data [88, 89], new diagnostic tests for diseases [90], translation of sign language in real-time [91], and design of fall prevention mechanisms for lower-limb amputees [92].

2.1.8 Myo Armband

The Myo Armband [78] (pictured in Figure 2.3) is of particular relevance to this work as its attractive cost (£150) relative to other sEMG detection systems, e.g. Delsys Trigno Wireless System ($\sim$ £15000) [93], and usability makes it ideal for widespread usage. Simultaneously the quality constraints imposed by such a low price make utilising the data a challenging problem.

Figure 2.3: The Myo Armband, a low-cost commercial sEMG device [78].

The Myo uses eight stainless steel electrodes in contrast to the more typical Ag-AgCl electrodes. The electrodes are active and make use of custom amplification

circuitry before outputting data via Bluetooth which can be read via an Applications Programming Interface (API). The Myo samples sEMG at 200Hz, giving it a bandlimit for perfect reconstruction of $\leq$ 100Hz [94]. This relatively low bandlimit would appear to limit its utility as the interesting range of EMG is generally held as 50-150Hz, and the useful range can extend up to 500Hz [66]. The following chapters, however, will demonstrate that despite its limitations, the Myo can be a valuable tool for sEMG classification.

The Myo also contains a nine axes Inertial Measurement Unit (IMU) which is made up of a gyroscope, accelerometer, and magnetometer, each of which has three axes. This data is sampled at 50Hz and is transmitted over Bluetooth and made available via the API.

Many other EMG acquisition systems exist, but the convenience and cost reduction provided by the Myo significantly reduce the barrier to entry while also helping broaden the appeal of sEMG based interaction. Evidence of this can be seen in the diverse research directions taken by recent work on sEMG utilising the device [95–98].

For these reasons, considerable effort is devoted to utilising the Myo Armband.

2.2 Machine Learning Overview

Let the definition of "Machine Learning" in this thesis' context be:

"The field of study that gives computers the ability to learn without being explicitly programmed."

This definition is attributed to Arthur Samuel (1959); it allows explicit encoding of the role machine learning plays in this work, which is to serve as the method for prediction. Related fields include statistics and data mining. This work often draws on machine learning literature while using statistical and analytical techniques to inform choices in how to choose and design learners to make predictions.

This work focuses on the problem of classifying hand gestures from sEMG within the framework of "End-to-End" learning, that is taking in raw data and designing a process that predicts the current hand gesture.

2.2.1 Feature Engineering

The first step in the "End-to-End" pipeline is typically the design and extraction of useful features. The idea being to represent the underlying data in the most informative way possible. This can aid interpretation of the data as well as reduce the informational, computational, and complexity load on any predictive algorithm used, which typically also improves the algorithm's performance.

Feature extraction is closely related to dimensionality reduction as features that lead to high performance often remove redundant and collinear information in the data. Similarly, useful features often incorporate an aspect of dimensionality reduction as this is an effective way of aiding algorithms in learning high-performance prediction.

The major downsides to feature design are the need for hand-crafting of the representation (often requiring domain-specific knowledge), the additional load incurred by extraction and the potentially complex trade-offs and interactions with prediction algorithms involved in the selection of individual features or combinations thereof.

The issue with features is well highlighted in sEMG classification by the large number of proposed features and the difficulty in attempts to compare them in the literature. The reason it is difficult is the diversity in experimental procedures and capture devices used, which reduces a researcher's ability to make quantitative comparisons [68, 99, 100]. This is in addition to the standard issues of feature-classifier interaction and information loss from dimension reduction. Lack of data availability from many studies potentially further exacerbates these issues. In contrast, the computer vision community has made fast progress on the basis of shared data sets for the benchmarking of algorithms and competitions such as Imagenet [101].

2.3 Neural Networks and Deep Learning

A large component of this work focuses on leveraging the predictive power of neural networks. The first and foremost reason to use neural networks is their state-of-the-art performance on a diverse range of problems [102]. Given the difficulties shown by previous work [37, 46] in reaching high levels of performance, neural network's are, therefore, a natural research avenue. In the context of sEMG hand movement classification, the other key advantage is the ability to form a complete end-to-end classification solution by bypassing the difficulties posed by feature extraction described in Section 2.2.1. Finally, the recent emergence of low-cost, efficient hardware designed for running neural networks, e.g. the Jetson TX2 [103], enhances their appeal for situations where a general purpose computer is unavailable or undesirable.

The main disadvantages are the fact that the resultant model is typically a black box making interpretation of the model difficult without additional measures. Training also requires potentially large amounts of data and significant computation time.

In the $\sim$ 7 years since Alexnet [104] the phrase "deep learning" has grown to encompass the majority of neural network based endeavours. There is no precise, universally agreed definition of "deep learning" however here it shall be defined as:

"Any neural network with an architecture more complex than a one hidden layer Multilayer Perceptron (MLP)"

Before exploring deep learning, however, it is necessary to ensure grounding in the fundamentals of neural networks. This section covers the fundamentals of neural network design and implementation in terms of the maths involved, architecture choices and hyper-parameter choices as well as summarising the history of the field.

2.3.1 No Free Lunch Theorems

The "no free lunch" theorems for machine learning and optimisation were derived by David Wolpert in the 1990s and have proved contentious within the machine learning community [105, 106]. The salient point of Wolpert's work was that two optimisation algorithms will perform the same when their performance is averaged across all possible problems. The insight being that finding a technique, method, algorithm or set thereof that can be applied in any circumstance is not possible, therefore making informed choices in solution design is necessary. The theorems are tempered by the fact that real problems are not randomly selected with uniform distribution from the set of all possible problems which leads to algorithms such as cross-validation performing better, on average, in practical problem-solving contexts [107].

2.3.2 The Neuron Model

A modern neural network is similar to most other ML algorithms in that it allows fitting a prediction function to a given set of input and outputs, provided sufficient quantity and quality of data is available.

Neural networks have a basis in the biology of the brain, which makes a useful starting point for the topic as well as providing intuitions into basic neural network operation.

Figure 2.4 shows a biological neuron and its counterpart model as used in neural networks. Both form the basic computational unit in their respective system although note that the mathematical model used is a *gross* simplification of real neuron interaction, meaning direct comparisons to real brain action are not useful. Neural networks are biologically inspired as opposed to an attempt to model real

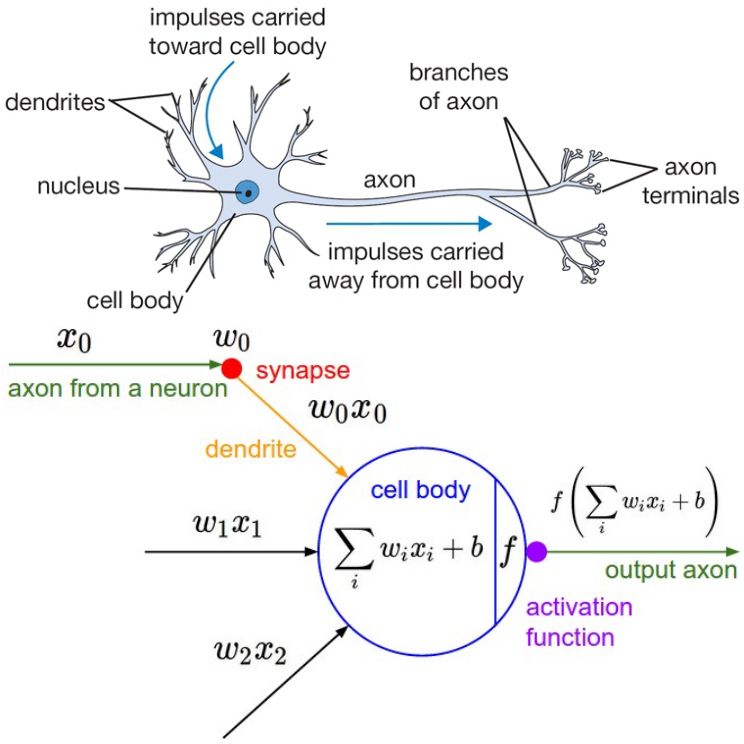

Figure 2.4: Drawing of an abstract biological neuron (top) and the mathematical neuron model used in neural networks (bottom) [108]. The analogy between the two is highlighted by the coloured text.

brain activity.

The model in Figure 2.4 is not immutable. For instance, it is possible to delay the application of the activation function or allow a neuron to reference its past outputs, however, unless otherwise specified, it is assumed the following equation describes a neuron's output:

$$y(x) = f\Big(\sum_i (w_i x_i) + b\Big) \tag{2.1}$$

where f is the activation function, w_i is a weight associated with the connection i, x_i is the input on connection i and b is a bias term.

2.3.3 Training and Backpropagation

Once an architecture has been selected it must then be "taught" via training on a given set of data. The dominant algorithm for accomplishing this training is the backpropagation algorithm.

Backpropagation itself is a special case of automatic differentiation with reverse accumulation and is used in conjunction with an optimisation algorithm and loss

function to adjust the weights of neurons in the network to minimise the loss function. The basis of backpropagation is to apply the chain rule through all possible paths through the network utilising dynamic programming to minimise the computation incurred by the vast number of possible paths [109, 110].

After a forward pass through the network, a loss value (error) is calculated for each output based on the selected loss function. The optimisation function defines how weights should be changed to reduce the error and backpropagation allows the error to be propagated backwards from the output to the input which makes it possible to update every weight in the network. This is why neural backpropagation is a supervised algorithm; it is necessary to have a target to compare against in order to compute the error.

Rather than passing a single input through the network and computing the error, it is typical to use a batch. A batch consists of storing the error of multiple forward passes and using the mean (or another function) of those errors to compute the update to be backpropagated. Batching reduces the computational overhead of training as modern implementations, particularly when using GPUs, make the cost of additional forward passes, once the model is loaded, small. Batches also smooth out training by reducing the likelihood of substantial changes in direction between updates. Batch size then becomes a hyper-parameter for training which can be adjusted to improve training times but must be monitored as overly large batches can harm performance [104].

Batches also prevent all weight updates becoming strictly positive or negative in a particular update pass as can be caused by all the inputs becoming positive. This issue is most relevant when using non-zero mean activations, such as the sigmoid function (see section 2.3.4).

Using batch sizes that are powers of 2 may also improve computation time both for training and testing due to the alignment of virtual and physical processors in GPUs. However, this has not been explicitly studied and thus is better used as a guideline.

Once the error for a batch has been computed the backpropagation algorithm computation of the appropriate partial derivative of the loss to flow back to a neuron so it can be updated. Algorithmically for a given weight in a layer l:

$$\frac{\delta \mathcal{L}}{\delta w^{(l)}} = \sum_{i}^{N_{l+1}} \left(\frac{\delta \mathcal{L}}{\delta w_i^{(l+1)}} \cdot \frac{\delta w_i^{(l+1)}}{\delta w^{(l)}} \right) \tag{2.2}$$

where $\frac{\delta \mathcal{L}}{\delta w^{(l)}}$ is the partial derivative of the loss $\mathcal{L}$ with respect to the weight to be updated $w^{(l)}$ of layer l. N_{l+1} is the number of weights in the layer $l+1$ and

i indexes the weights in the layer $l + 1$. This method can be applied recursively from the network output to reach any given weight with intermediates stored and reused to reduce computation. Figure 2.5 illustrates this visually with respect to the neuron model.

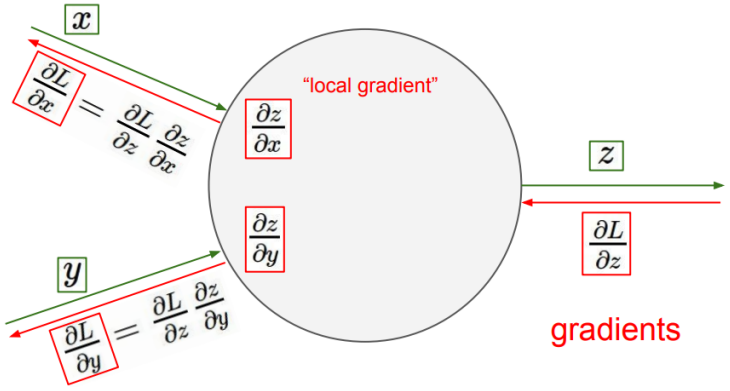

Figure 2.5: Backpropagation through a single neuron via application of change rule, L is an incoming loss or error signal ([108], edited).

Loss Functions

Correction selection or design of loss of function is essential to ensure training converges to a useful model. Where there are multiple objectives, it is generally necessary to hand design the loss function to account for the differences between the types of predictions and weightings of each objective. Particular attention must be paid to the relative scaling of objectives as if one objective has a broader range it can dominate training, therefore, each objective must be scaled to the same range or weighted appropriately for the problem. There are several common loss functions that can be used on the majority of problems and make useful bases for designing custom loss functions.

Mean Squared Error (MSE) is defined as

$$\mathcal{L} = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2 \tag{2.3}$$

where $\mathcal{L}$ is the loss, N is the number of output values, y_i is the true output and $\hat{y}_i$ is the predicted output. When targets are all in the same range computation is often saved by omitting the divide by N, the function is then called the *L2* Loss.

Mean Absolute Error (MAE) is similar to MSE but takes the absolute value in-

stead of the square.

$$\mathcal{L} = \frac{1}{N} \sum_{i=1}^{N} |y_i - \hat{y}_i| \tag{2.4}$$

Similar to MSE the division by N can be dropped in some cases, in which case it is called the *L1* Loss.

MSE or L2 loss is generally less robust than MAE or L1 loss since L2 squares the error making it much more sensitive to outliers than L1. L1, however, can suffer from instability in the solution making L2 generally a better solution when outliers can be accounted for. The instability in L1 loss is caused by the constant magnitude of its gradient, which can cause inappropriately large updates when close to minima.

Cross-entropy or *log loss* is useful when using probabilistic predictions for class labels. It is defined as:

$$\mathcal{L} = -\sum_{i=1}^{N} \sum_{k=1}^{K} y_i^{(k)} \log \left(\hat{y}_i{}^{(k)} \right) \tag{2.5}$$

where $y_i^{(k)}$ is the true probability of class k for sample i and $\hat{y}_i^{(k)}$ is the predicted probability of class k for the sample i. The cross-entropy is generally used with an output layer that uses the softmax activation function.

The primary benefit of the cross-entropy loss is that it accelerates learning by making the weight update rate proportional to the error in the output instead of the gradient of the error [111].

This setup is canonical with using a softmax activation function (see Section 2.3.5) in the output layer. When it is used there is a probabilistic interpretation of the output.

Optimisation Functions

Once the loss has been computed for a particular example, batch or mini-batch, an optimisation function is required to determine how to use that loss to update the weights of the neurons in the network. The Stochastic Gradient Descent (SGD) method with momentum is the typical starting point:

$$v_t = \gamma v_{t-1} - \eta \nabla f(w_{t-1}) \tag{2.6}$$

$$w_t = w_{t-1} + v_t \tag{2.7}$$

where v_t is a velocity term initialised at 0 and recomputed at each update step, γ is the momentum term, v_{t-1} is the previous velocity, η is the learning rate and

$\nabla f(w_{t-1})$ is the gradient of the error signal at the weight w_{t-1}. w_t is the new weight. The error term is computed from the cost function and backpropagated to the neuron being updated. The momentum term γ is a value in the range $[0, 1]$ which adjusts the attenuation of the update velocity, making it act more similarly to a friction coefficient than momentum despite the naming.

The gradient term is often augmented to become Nesterov (accelerated) momentum due to reliable, practical results and better convergence guarantees for convex functions [112]. Adding Nesterov momentum is done by modifying the weight update to the following three-step process:

$$w_{ahead} = w_{t-1} + \gamma v_{t-1} \tag{2.8}$$

$$v_t = \gamma v_{t-1} - \eta \nabla f(w_{ahead}) \tag{2.9}$$

$$w_t = w_{t-1} + v_t \tag{2.10}$$

The idea is to evaluate the gradient update with the weight that would be produced by the momentum update step alone, a "lookahead" on the gradient update portion of the equation. Figure 2.6 visualises this method.

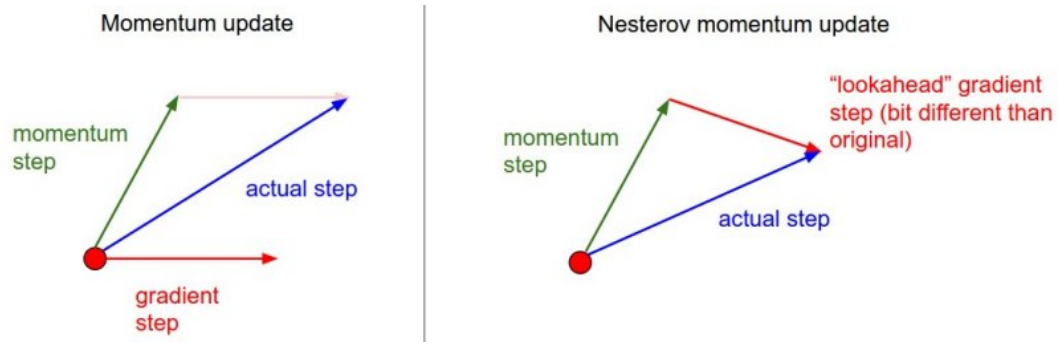

Figure 2.6: Comparison of momentum and Nesterov momentum in terms of update rule [108].

There are many different optimisation functions that improve upon SGD, typically by allowing for varying learning rate η (making selection of the value less critical), speeding up convergence or introducing optimisations for different sets of problems. In this work, the Adam [113] algorithm is used as the choice of optimisation function; it is covered in greater detail in Chapter 5, where it is first used.

Methods based on the Newton method [114] take the form

$$w_t = w_{t-1} - [Hf(w_{t-1})]^{-1} \nabla f(w_{t-1}) \tag{2.11}$$

where $Hf(w_{t-1})$ is the Hessian matrix (square matrix of second-order partial derivatives) and $\nabla f(w_{t-1})$ is the gradient vector. These methods have the potential to improve convergence while reducing the necessary hyper-parameters (note no learning rate). In deep learning applications, however, explicit computation and inversion of the Hessian is often infeasible due to its $\mathcal{O}(n^2)$ complexity in terms of the number of network parameters.

Quasi-Newton methods such as the Limited-Memory Broyden-Fletcher-Goldfarb-Shanno (L-BFGS) [115] attempt to bypass this issue by computing an approximate of the inverse Hessian however often have other limitations e.g. L-BFGS must be computed over the whole training set rather than batches. These limitations, the added complexity and lack of scaling to larger problems has meant these methods are not widely used.

Normalisation

An essential step in the training process is normalisation. While not strictly necessary, normalisation can provide a substantial speed up in convergence time and help guard against the finding of poor quality minima. Normalisation achieves this by ensuring that all features/inputs to the network have the same range preventing the learning rate effectively being adjusted proportionally to a feature's relative range. This process helps ensure the resultant loss/error surface is conducive to learning.

One standard method is to normalise all data to mean 0 and variance 1, based on the mean and variance calculated from the training data set:

$$x_i = \frac{x_i - \mu}{\sigma^2} \tag{2.12}$$

for all input data x_i in the data being worked on. μ is the mean value for each feature in x_i calculated from the training data only and σ is the standard deviation also only calculated from training data.

An alternative approach is batch normalisation. Batch normalisation [116] normalises each output of a layer in a similar way to Equation 2.12 for each training batch. Utilising batch normalisation has the additional benefits of providing regularisation and reducing internal covariate shift.

Internal covariate shift is the change in the distribution of network activations

due to changes in the network parameters caused by training. It can be viewed as the coupling between outputs of earlier layers and later ones. Batch normalisation reduces this coupling by making the activation distribution more consistent.

Batch normalisation also adds regularisation to a network. The regularisation comes from the fact that the normalisation computes mean and variance from each batch, which effectively adds noise to the process.

Regularisation

Good regularisation is vital to prevent overfitting of a network to its training data, which will result in poor generalisability.

There are many possible ways to regularise a network. Popular choices are global L1 and L2 weight decay which add a term to weight updates of the form

$$-\lambda|w| \tag{2.13}$$

in the L1 case and similarly in the L2 case:

$$-\lambda w^2 \tag{2.14}$$

where λ is a hyperparameter. L1 decay allows weights to decay to near zero values which allows for sparse representations analogous to feature selection within the network. L2 effectively punishes outliers usually leading to "diffuse" representations where all inputs are used by some small amount.

Dropout [117] is also a powerful technique to combat overfitting. It functions by (during training only) randomly removing or nullifying connections at each update step with the probability of a connection being dropped as a hyper-parameter. Dropout is shown in Figure 2.7.

Regularisation can also be domain-specific. For instance, when working with images, it is typical to rotate and crop them so that a network learns a representation that is invariant of such transformations. These are likely to be irrelevant to the desired output but might cause bias depending on the data set available. In EMG classification filtering techniques to remove mains noise or known interference perform a similar job preventing a network learning to classify based on signals known to be decoupled from the classes under investigation.

2.3.4 Activation Functions

An activation function (Figure 2.4, purple) is the non-linear portion of the neuron model, which allows fitting to any function. Appropriate selection of activation

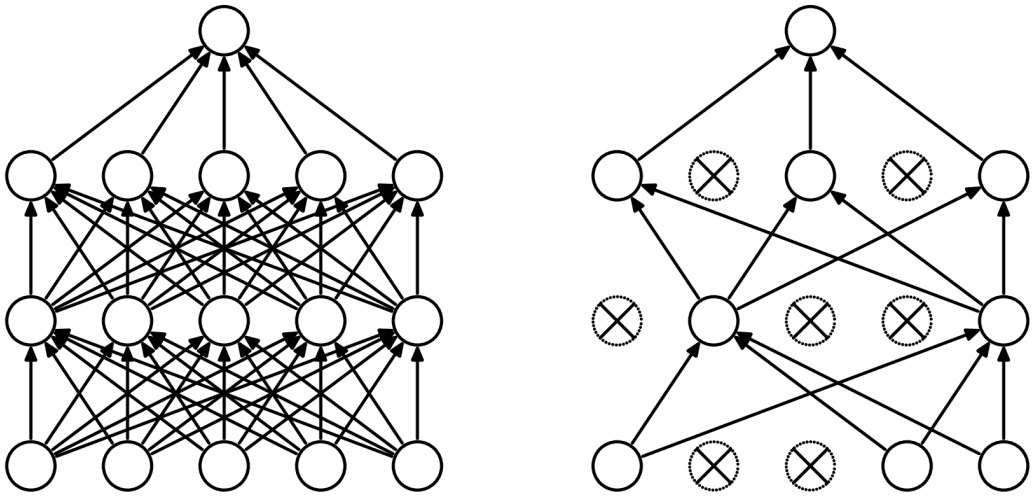

Figure 2.7: Visualisation of Dropout's effect on network architecture during training. Left is a standard neural network architecture; right is the same architecture after applying dropout [117].

function is critical to achieving the best results in practice [118].

Logistic Sigmoid

A logistic sigmoid is defined as:

$$f(x) = \frac{1}{1 + e^{-x}} \tag{2.15}$$

which effectively compresses any real-valued input into the range 0 to 1. Figure 2.8 shows this squashing effect as well as the function's derivative.

The logistic sigmoid function is often referred to as, "sigmoid function" or, "sigmoid" in the neural network literature.

The choice to use the logistic sigmoid function, historically, was based on the biological analogy for neurons. It can be viewed as an interpretation of the firing rate real neurons exhibit, i.e. when the inputs multiplied by the weights reach a threshold, the real neuron will "fire" producing a voltage spike on its output. The logistic sigmoid represents this with an output of 0 indicating no firing and an output of 1 denoting firing at the maximum possible frequency.

In the computational model, however, the logistic sigmoid poses a significant issue. This can be seen in Figure 2.8; the gradient/derivative is close to zero for any high magnitude input. During training, this causes almost no error to flow back along the path resulting in little to no weight update and consequently, poor learning. This issue is known as the "Vanishing Gradient" problem and is covered

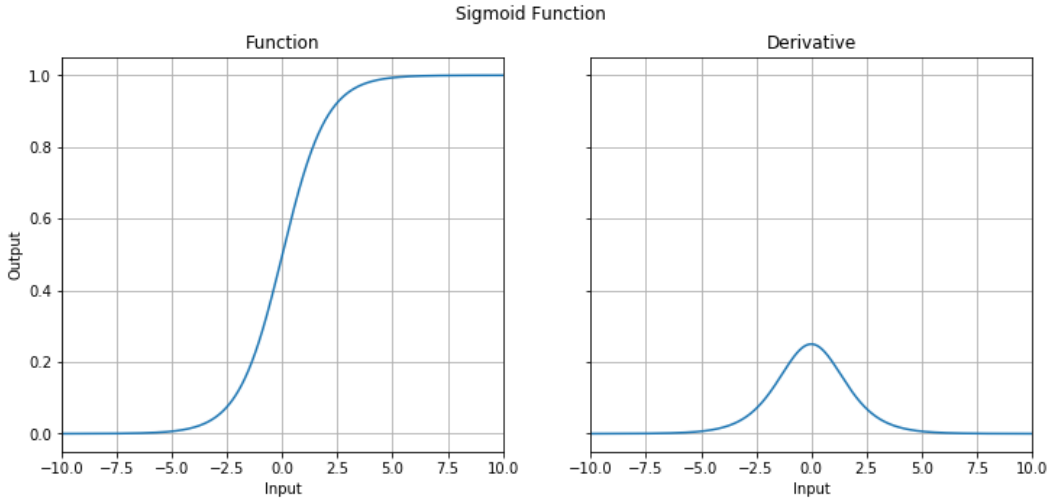

Figure 2.8: Logistic sigmoid activation function and its derivative.

in Section 2.3.3.

Another issue presented by the logistic sigmoid function is that its output is not centred at 0. If all inputs to a neuron (i.e. in the next layer) are $x > 0$ then the gradients on all the weights will all be strictly positive or strictly negative which further hinders the backpropagation algorithm. Specifically, it may be desirable to increase the value of one weight while reducing the value of another, which is not possible. This is mostly mitigated by the use of batches in training (since taking an average of the gradient allows positive and negative updates) this, however, must be accounted for when using any non-zero centred activation function.

Mathematically this issue can be shown by the following equations:

$$z = \sum_i (w_i x_i) + b \tag{2.16}$$

$$\frac{\mathrm{d}z}{\mathrm{d}w_i} = x_i \tag{2.17}$$

$$\frac{\mathrm{d}L}{\mathrm{d}w_i} = \frac{\mathrm{d}E}{\mathrm{d}z}\frac{\mathrm{d}z}{\mathrm{d}w_i} = \frac{\mathrm{d}L}{\mathrm{d}z}x_i \tag{2.18}$$

where L is the incoming loss (alternatively, the error signal). Therefore, unless $\frac{\mathrm{d}E}{\mathrm{d}z} = 0$ which would preclude training, then if all $x > 0$ all the gradients will have the same sign.

Consequently, because of these two issues, the logistic sigmoid function is now rarely used.

Tanh

The *tanh* function is shown in Figure 2.9 and is another type of sigmoid function. For neural network purposes, it can be viewed as a scaled version of the logistic sigmoid function translated to centre on 0 fixing one of the issues of the logistic sigmoid function. It still suffers from the vanishing gradient problem, however.

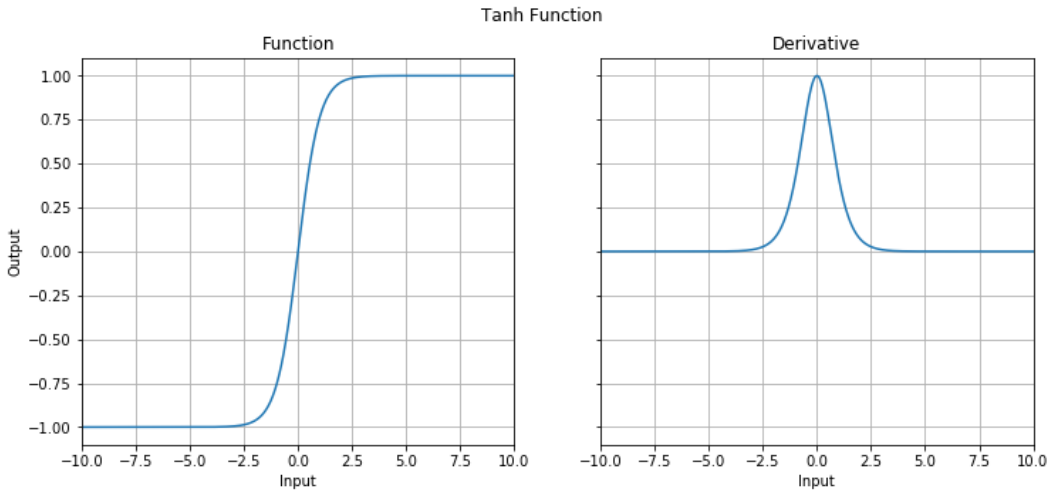

Figure 2.9: Tanh activation function and its derivative.

Note that the *tanh* activation was taken from trigonometry as a replacement rather than as an evolution of the logistic sigmoid activation.

Rectified Linear Unit

Figure 2.10 shows the Rectified Linear Unit (ReLU) [119]. The ReLU is defined as $max(x, 0)$ or:

$$y(x) = \begin{cases} x, & x > 0 \\ 0, & x < 0 \end{cases} \tag{2.19}$$

The gradient at 0 is technically undefined; however, it is generally set to 0 to avoid computational issues.

This design introduces the necessary non-linearity for fitting any function while alleviating the vanish gradient problem by producing a gradient that is either 1 or 0. These attributes make the design very useful for training networks with many parameters.

Though a formal study does not exist, it is reasonable to assume that the ReLU and its extensions are the most popular choice of activation function for most applications, based on the prevalence of their usage in the literature e.g. [39, 113, 120–130]. The combination of alleviating the vanishing gradient problem while

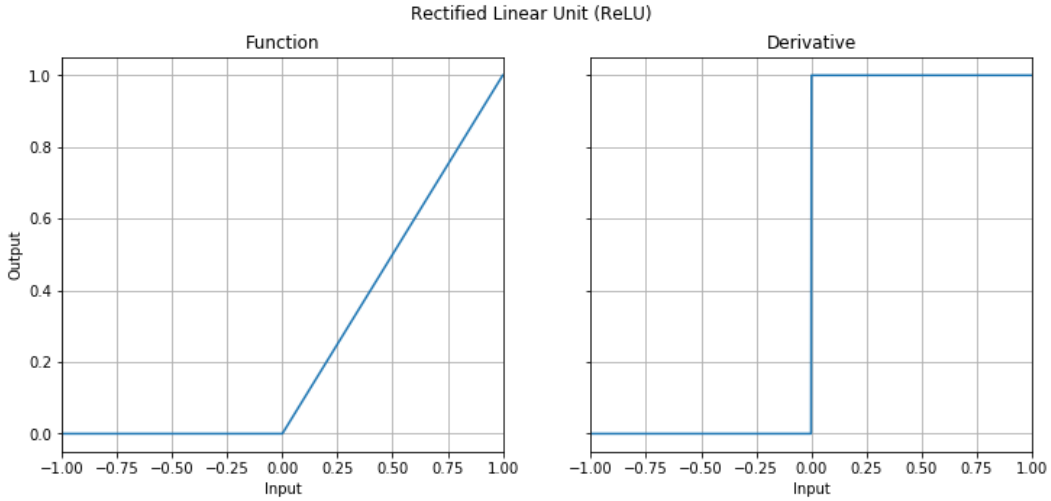

Figure 2.10: Rectified Linear Unit activation and its derivative. The derivative shows the useful property of either being 0 or 1 and no other values.

being simple to compute and also improving the convergence rate of gradient descent [104] make the ReLU an attractive option.

The main issue with the ReLU is that of "dying" [131]. An improper update (too large and in the "wrong" direction) can cause a ReLU to never activate again on any data point in the dataset. This means that, during backpropagation, the gradient of the error with respect to the ReLU is always 0. Therefore the weights do not update, leading to the ReLU's "death". Dying can be mitigated by careful selection of learning rate; however, it remains an issue that must always be monitored when using the ReLU.

Rectified Linear Unit Variants

Several variants on the ReLU have been proposed that deal with the issue of "dying". Particularly popular is the Leaky Rectified Linear Unit (LReLU) which allows a small proportion of the input through below 0:

$$y(x) = \begin{cases} x, & x > 0 \\ \alpha x, & x \leq 0 \end{cases} \tag{2.20}$$

where α is small value often between 0.1 and 0.01 although this is generally a global hyper-parameter for the network in question. This is equivalent to $max(x, \alpha x)$.

Other variants include the Parametric Rectified Linear Unit (PReLU) [132] which makes the α value of the LReLU a learnable parameter and the Exponential Linear Unit (ELU) which aim to make the mean activation close to zero to improve

training times [131].

An alternative neuron design known as Maxout [133] generalises the idea of ReLUs to encompass the whole neuron by performing the computation:

$$y(x) = max(w_1 x + b_1,\ w_2 x + b_2) \tag{2.21}$$

This keeps the benefits of the ReLU but avoids the "dying" issue at the cost of doubling the number of parameters to learn in each neuron.

2.3.5 Network Architectural Choices

The Multilayer Perceptron (MLP) is a feed-forward network formed of sequential layers where each neuron in the layer L_{i-1} is connected to each neuron in the layer L_i. This design is known as a dense layer. Figure 2.11 shows this design with two hidden layers, although any non-zero number of hidden layers would qualify as an MLP. The term "hidden layer" is used to indicate that the layer's outputs are not visible under normal operation and as such in most neural networks every layer except the input and output are considered to be "hidden layers".

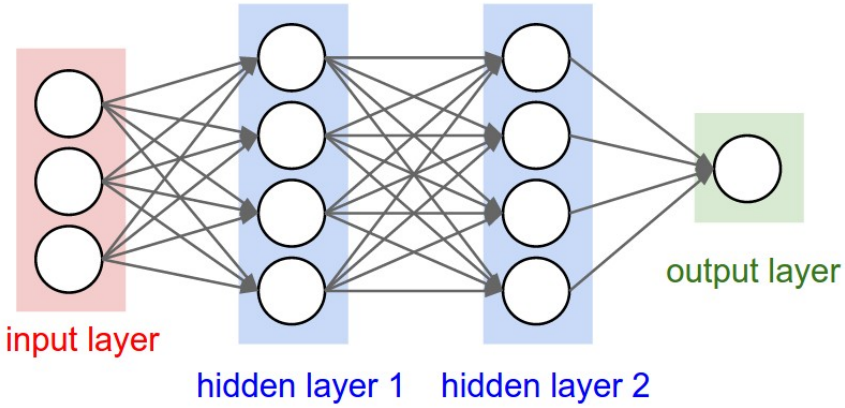

Figure 2.11: A basic neural network model; this architecture is also known as an Multilayer Perceptron (MLP) [108].

Forward Pass

At run time, information in the network flows forward hence the name "forward pass". Exact programmatic implementation can vary depending on the hardware and network architecture in use; however, an appropriate model for most cases is that each layer is updated in turn using the neuron equation:

$$y(x) = f\Big(\sum_i (w_i x_i) + b\Big) \tag{2.22}$$

where the neuron output $y(x)$ in a layer becomes one of the neuron inputs x_i for the next layer.

In the architecture shown in Figure 2.11 this leads to a matrix of the inputs being read at the input layer, a matrix of outputs being computed for hidden layer 1, then a matrix of outputs for hidden layer 2 and finally the output being computed in the output layer. This output could be either classification or regression depending on how the output is treated.

Classification Output

When a classification is desired, a typical approach is to make the activation function of the final layer the softmax function [134].

In Figure 2.11, the output layer only has a single neuron which leads to a special case of softmax, making it equivalent to logistic regression in the output layer. The general (multinomial) form of the softmax function is:

$$y(z)_j = \frac{\exp(z_j)}{\sum_{n=1}^{N} \exp(z_n)} \tag{2.23}$$

for z being the pre-activation function output of a neuron where N is the number of neurons and j is the output class $j = 1, \ldots, N$.

The softmax function normalises the output such that all outputs sum to one, making it suited to probabilistic interpretation. If trained using cross-entropy loss (Section 2.3.3) the output can be interpreted as performing maximum likelihood estimation because the negative log-likelihood of the correct class is minimised.

Despite this probabilistic interpretation of the output it cannot be treated as a strict probability, i.e. if the network outputs 0.8 for a class this is *not* equivalent to assigning a probability of 80% that the input represents that particular class.

Regression Output

If a regression on the output is desired, the most straightforward approach is to make the activation function of the output layer the identity function. Alternatively, a restriction may be imposed by using a different function depending on the application, e.g. constraining the output to non-negative numbers. If used, the alternate function must be chosen carefully to avoid issues with gradients and to ensure that it works with the chosen loss function (Section 2.3.3).

Dense Layers

Dense or fully connected layers connect each neuron to every input; this is the type of layer shown in Figure 2.11. Formally each neuron in the layer L_i is connected to each neuron in the layer L_{i-1} such that:

$$z^{(l,k)} = f\left(\sum_{m=1}^{K_{l-1}} \left(w^{(l,k,m)} z^{(l-1,m)}\right) + b^{(l,k)}\right) \tag{2.24}$$

where $z^{(l,k)}$ is the output of neuron k in layer l, $h_a(.)$ is the activation function of the layer, $w^{(l,m)}$ is the weight associated with input $z^{(l-1,m)}$ (output from the previous layer $l-1$). m indexes these inputs, of which there are a total of K_{l-1} and lastly $b^{(l,k)}$ is a bias term.

Superscript indexing is used for consistency and to avoid space issues caused by the numerous indexes necessary for more complicated layers.

Convolutional Layers

Convolutional layers are similar to dense layers; however, instead of global connectivity to the previous layer, each neuron is only connected locally to a small number of neurons in the previous layer at any given time. The connections are then moved, as part of the forward pass, so that each neuron's local connection pattern is swept through the previous layer [104, 135]. This process encodes the explicit assumption that local information is useful for solving the problem.

A network that makes use of convolutional layers is generally known as a Convolutional Neural Network (CNN). CNNs have seen usage in many domains, often improving the state-of-the-art. The idea for convolutional layers came from the study of the visual cortex where it was noted that individual neurons in the visual cortex respond to stimuli in small regions of the visual field.

Convolutional layers solve the scaling issues inherent in dense layers by enforcing local connectivity, which dramatically reduces the number of parameters compared to a dense layer. For example, a small 400×400 pixel RGB image would need $400 \times 400 \times 3 + 1 = 480001$ weights per neuron in a dense layer. This quickly makes the network unmanageably large, leading to issues with computation, necessary data and overfitting. A convolutional layer, in contrast, can accept the same size input with parameter numbers orders of magnitude lower. For instance, a 5×5 convolution with 64 filters on an image with 3 colour channels uses $(5 \times 5 \times 3 + 1) \times 64 = 4864$ weights no matter the height or width.

In modern convolutional networks, many convolutional layers are stacked on top of each other. Each layer is composed of k learnable filters where each is

connected to a small volume of the previous layer (and the original image in the input layer). This connected volume, known as a **receptive field**, is then swept through the filter's input volume to produce a map of features at the filter output. This process allows the lower level layers to detect edges, the ones above to detect groups of edges and the higher level layers to identify motifs and complex patterns. Intuitively a network such as this encodes the idea that local features are important, that they are equally relevant anywhere in the input, and that there is a hierarchy present.

The neuron model thus remains the same, the only difference being which inputs are connected. The number of parameters for each filter is, therefore, calculable as

$$n = F_w \times F_h \times x_d + 1 \tag{2.25}$$

where n is the number of parameters, F_w is the filter width, F_h is the filter height, x_d is the depth of the input and the additional 1 comes from the bias term.

Consider images as 3D volumes where the depth is the colour channels. Each convolutional filter is considered to have dimensions equal to the number of dimensions it moves through, i.e. for an image, a 2D convolutional filter generally sweeps a volume of some small width and height through the image connecting fully depth-wise. Figure 2.12 shows an example on a 32×32 image with 3 colour channels.

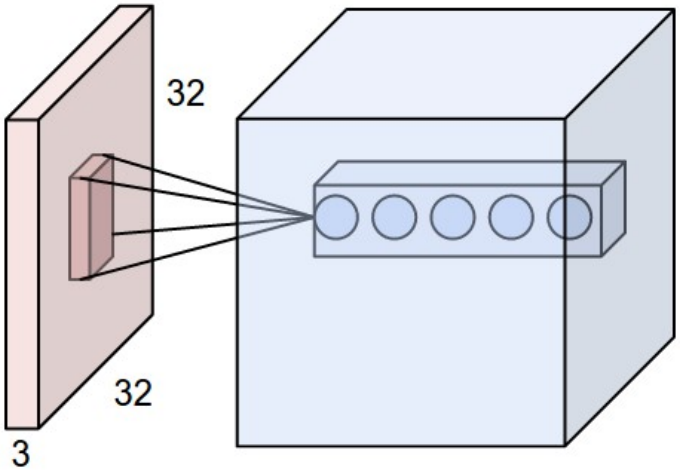

Figure 2.12: Example of connections in a convolutional layer with 5 filters showing the volume each filter "sees" in the input image, which is a $32 \times 32 \times 3$ volume [108].

The output dimensions of the convolutional layer are defined by the input dimensions and three hyper-parameters: the number of filters used, the stride size and the padding.

The number of filters used directly determines the layer output depth as feature maps are stacked.

Stride determines how far the filter is moved through the input volume at each step. If the stride is 1 then the filter is moved one pixel at each step. The stride is set for each dimension to be moved through and often makes the output smaller in those dimensions.

Lastly, zero-padding may be added around the input of each layer to control the dimensionality of the output volume.

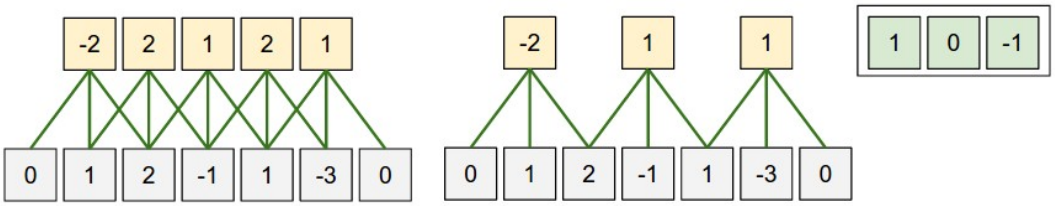

Figure 2.13: Illustration of stride for a 1D convolutional filter. The filter is shown in the top right and has length 3 with no bias term. The example on the left uses a stride of 1, and the example on the right shows the same input with a stride of 2 showing the difference in calculation and output volume [108].

Figure 2.13 shows a numerical example of the filter update for a 1D convolution with differing stride sizes. It can be seen that a padding of zeros can be added to edges of the input to change the resultant output size. Care must be taken in selecting appropriate filter sizes, and padding parameters or the edges of the input may be missed due to the input size not being exactly divisible by the filter size in a particular dimension.

In this thesis, 2D convolutions are primarily used for their flexibility, although, in several cases, they equate to 1D convolutions evaluated in a 2D manner. The equation for 2D convolutions is

$$z_{r,c}^{(l,k)} = f\left(\left(\sum_{m=1}^{K_{l-1}}\sum_{i=0}^{R_l-1}\sum_{j=0}^{C_l-1} w_{i,j}^{(l,k,m)} z_{\tilde{r}+i,\tilde{c}+j}^{(l-1,m)}\right) + b^{(l,k)}\right) \tag{2.26}$$

where $z_{r,c}^{(l,k)}$ is the neuron output at location (r,c) in the current layer l, for $r = 1,\ldots,r_l$, $c = 1,\ldots,c_l$. $R_l \times C_l$ is the convolution filter size, the convolution filter indexed by k, for $k = 1,\ldots,K_l$, is composed of the adjustable CNN weights $w_{i,j}^{(l,k,m)}$. m indexes the filters in the previous layer, $b^{(l,k)}$ is a bias term and $h_a(.)$ is the activation function of the current layer. The terms $\tilde{r}$ and $\tilde{c}$ index locations in the previous layer $l-1$ which for odd numbers begins at $\tilde{r} = r - \lfloor R_l/2 \rfloor$ and $\tilde{c} = c - \lfloor C_l/2 \rfloor$ and the offset must be chosen by the designer for even numbers. The 1D case is a particular case of this equation where the R_l or C_l summations

are removed. For higher dimensional convolutions additional summations must be added for each additional dimension.

Stride is handled by increasing the step size used when selecting r and c locations to compute then concatenating the results while maintaining relative positions to form the new output matrix (see Figure 2.13)

Typically in a CNN, the last convolutional layer is flattened and fed into a dense layer to produce the final output although this is not strictly necessary and can be replaced with a 1×1 convolutional layer with a number of filters k equal to the number of labels to make a fully convolutional network.

Since the neuron model has not been altered in constructing a convolutional layer, dense layers and convolutional layers are fully interchangeable. That is, it is possible to convert any convolutional layer to a fully connected layer that performs the same computation and vice versa.

Pooling Layers

Pooling layers can be used between convolutional layers in CNNs. They perform downsampling by reducing the non-depth dimensions of their input volume. Downsampling allows a reduction in the number of parameters in the network, reducing computational requirements and helping guard against overfitting. The downside to pooling layers is that, as with any downsampling, information is lost in the process which can impact performance.

Pooling layers follow a similar architecture to convolutional layers. They use a filter size such as 2×2 and apply a function at each depth level sweeping across the input volume with a defined stride. The difference is that the function generally does not have learnable parameters like a convolutional filter.

Taking the *max* value of the filter region is a typical approach which produces good results in practice; however, the filter function is a hyperparameter. Other functions that are often used are *mean*, $L1$ and $L2$ *norms*. The different functions allow for control over sparsity and to encode assumptions about the importance of information in the network. Figure 2.14 shows an implementation of a max pooling layer and how it reduces the output dimensions as well as its effect on images.

Recurrent Layers

Layers with recurrent connections allow information to be retained between forward passes of a network and thus allow for dynamic temporal behaviour. This also makes them excellent candidates for handling sequences and dealing with

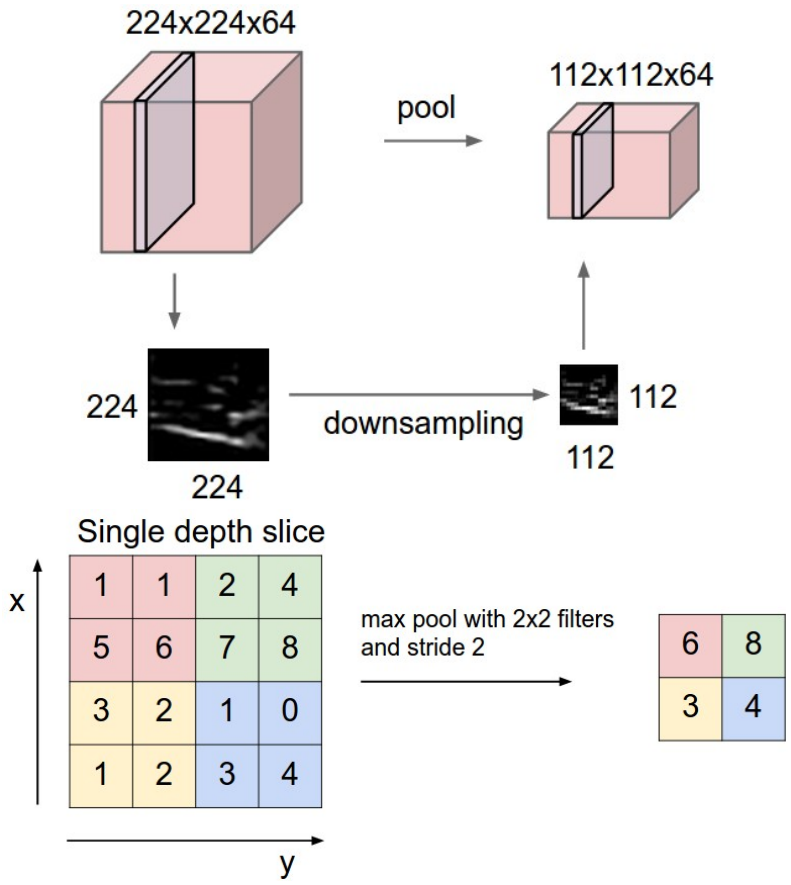

Figure 2.14: Illustration of the spatial reduction caused by the pooling layer and its effect on an image (top). The bottom example is of a *max* pooling layer with stride size (2, 2) operating on a $4 \times 4 \times 1$ input volume [108].

variable length inputs such as in speech recognition [136].

Modern variants often make use of the Long Short-Term Memory (LSTM) [137] units or the Gated Recurrent Unit (GRU) [138] which help alleviate the vanishing/-exploding gradient problems common when using simple recurrent units [139]. The designs are shown in Figure 2.15. Combating vanishing/exploding gradient is achieved via the various switches within each unit that control information and learning flow (via additional learnable parameters) such that long term dependencies can be captured and information explicitly "forgotten" rather than gradually removed via learning updates.

2.4 Hand Movement Classification

One of the key obstacles to the usage of EMG in many cases is the difficulty and expense of large scale data acquisition. The necessary hardware is expensive,

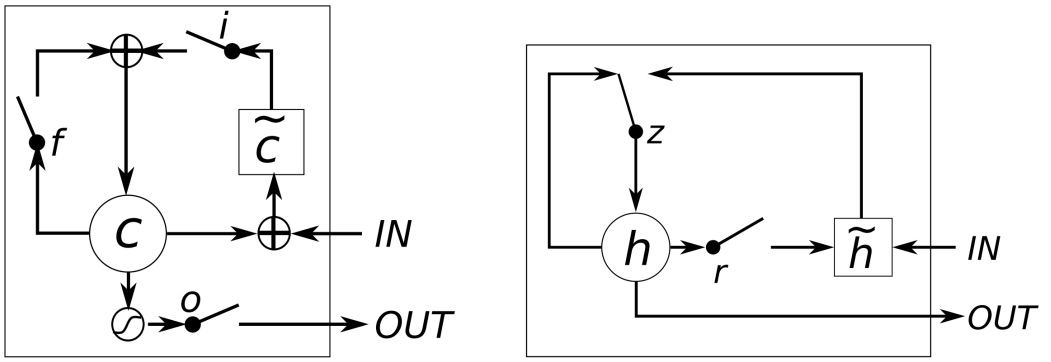

Figure 2.15: Illustration of an Long Short-Term Memory (LSTM) unit (left) and Gated Recurrent Unit (GRU) (right). On the left i is the input gate, f is the forget gate, o is the output gate, c is the memory cell, and $\widetilde{c}$ is the new memory cell contents. On the right r is the reset gate, z is the update gate, and h and $\widetilde{h}$ are the activation and candidate activation, respectively [139].

although the Myo Armband has helped in this regard (Section 2.1.8), experiments may require hours to acquire reasonable amounts of data which can cause fatigue or attention issues and acquisition of ground truth requires special attention and methods.

Until recently, this hurdle to data collection stifled research progress due to a culture of closed-source databases [21, 27, 29, 33, 35, 43, 140–144] requiring any potential researcher to undertake significant initial effort to acquire the data necessary for an investigation. This goes against the lessons learnt form other fields e.g. image recognition where the consensus is that making data publicly available facilitates progress [145].

In an effort to overcome this significant hurdle the Non-Invasive Adaptive Hand Prosthetics (NinaPro) Project [37, 46, 146] (now also the Megane Pro Project [147]) have produced several databases of sEMG data and various other sensor data on both healthy subjects and trans-radial amputees. Along with the data, they have also published studies looking to benchmark classification performance on and explore the data [37, 48, 148]. More recently, they have also applied deep neural networks to the problem [47], although they did not find a design that improved upon other techniques.

This work dedicates a significant amount of effort to expanding upon the results presented by the NinaPro project improving upon classification performance through algorithmic choices as well as data processing and validation techniques. The NinaPro data sets were chosen not just because they are freely available but because they present data from high-quality electrodes used on a relatively large number of participants with significantly more movements tested than most other

studies. This makes the data ideal for exploring what is possible with sEMG classification.

Aside from the issue of closed-source data, there are several other vital issues in the field that require addressing. Most important is that of agreeing on the appropriate metrics and validation procedures as these vary significantly study to study making results even on the same data largely incomparable without reimplementation of past methods. Relatedly, as with data, the code used is often not published or not published in a functional state which exacerbates the problem of repeatability and comparison to existing studies.

Another issue in designing new studies and comparing with old studies is the definition of ground truth and decisions on what is to be classified. One of the bigger issues being whether *rest* should be classified, some studies choose to omit it from classification [21, 24, 26, 29, 42–44] while others include it [19, 20, 23, 37, 39, 48] which can make a difference in the results reported as well as being essential to practical applications. These topics will also be extensively covered in later chapters.

In general, the field focusses on the classification of relatively small numbers of movements, e.g. 4-7 [18–28] or 9-15 movements [29–35, 149] in comparison to the $\sim 40 - 50$ in the NinaPro data sets. Expanding the number of movements that can be classified is vital for the improvement of myoelectric interfaces as it increases the potential interactions available to users.

Research has been conducted on the fusion of EMG signals with other data streams, e.g. [150–153]. The aim being to improve performance by incorporating additional information. Results from these studies are generally positive, indicating that sensor fusion is a viable way to improve performance in some settings. Addition of extra sensors, however, increases the cost of the necessary hardware and the complexity of associated algorithms. These concerns may be minor but must be balanced with expected performance improvement for a given application.

The full spectrum of machine learning algorithms has been applied to EMG including everything from simple Linear Discriminant Analysis (LDA) classifiers [23, 37, 42] to Support Vector Machine (SVM) classifiers [19, 37, 154, 155], neural networks [19, 37, 156], mixture of experts models [157] and many others. Notably, SVM and Random Forest (RF) classifiers have put up consistently good results in the past.

Significant effort has also been put into feature engineering to improve the performance of learning algorithms [55, 62, 99, 100, 158]. Thus far, however, no single feature or set of features has been found that is optimal in all scenarios

likely due to the fundamental complexity of the received signals. Therefore it is common to use combinations of features as input to attempt to achieve a more broadly applicable representation of the underlying information which has shown good results in practice [37].

2.4.1 Deep Learning Approaches

Newer work has started to make use of deep learning [28, 39, 47, 150, 159, 160], which allows side-stepping of the feature design problem by offering a complete end-to-end learning solution with the attractive property of tailoring the effective feature extraction to each subject.

The work by Atzori *et al.* [47] demonstrated that a convolutional neural network could be used to achieve performance comparable to, but not better than, other classification approaches on the NinaPro databases. Their architecture appeared to struggle on the higher frequency data from database 2 losing performance relative to database 1 while algorithms such as random forests lost little to no performance. They conclude that more complex architectures may be necessary to improve performance over the baseline they set.

Another convolutional neural network solution is presented by Geng *et al.* [39]. They demonstrate high performance on some data sets using a solution that only views a single instance of EMG data rather than a window (which is more typical). The network appears to work particularly well when a large number of electrodes are available. Their network is evaluated on NinaPro databases 1 and 2 but only on the full gesture set for database 1. On database 1 it improves upon the accuracy of Atzori *et al.'s* results [47] demonstrating that it is possible to outperform other classification methods using a convolutional neural network.

Further Hu *et al.* [150] recently extended Geng *et al.'s* [39] network design by adding an attention mechanism [161] before the classification output. This allowed the network to accept windowed input and further improved the accuracy on all the data sets tested relative to the original. Additional feature engineering was also added that improved accuracy by preprocessing windows using the Fast Fourier Transform (FFT) and extracting feature known to be useful in the sEMG domain.

The work by Atzori *et al.* [47] and Geng *et al.* [39] is compared to directly in later chapters. Hu *et al.'s* work [150] is not due to the timing of its release. However, Hu *et al.* [150] shares a majority of its authors with Geng *et al.* [39] so it is reasonable to assume supporting methods are similar between the two since differences are not specified. This is relevant because the evaluation of accuracy in these works potentially leads to biased results. Issues surrounding result bias are explored in Chapters 4 and 5.

Another recent piece of work by Côté-Allard *et al.* [28] has explored the applications of transfer learning using deep neural networks. This is of particular interest since the amount of data on any single subject is often relatively small since it is unrealistic to gather thousands of examples from an individual subject. The idea is to take a corpus of data from other subjects and a smaller amount of data from a new target subject and use the combination of the two sets to improve overall performance; a concept which is explored further in Chapter 7.

2.4.2 Key Issues

There are three key issues that naturally arise from the literature:

The first issue is the representativeness and comparability of results. Different studies can vary to a large degree. Differences include, but are not limited to: numbers of subjects, electrodes used, electrode placement, gestures or movements used, data capture techniques used, and validation techniques used. This variability between studies often makes direct comparison not useful, because the actual problems being tackled vary substantially.

The way to tackle this underlying issue is by the sharing of data sets and concerted effort applied to improvement in methods on freely available data.

Even when working on a particular data set, there are still critical issues that must be addressed. Experimental methods for gathering data on hand movements typically involve *rest*-movement-*rest* cycles. This approach almost invariably leads to significant class imbalance due to *rest* being repeated more often than other movements. Class imbalances are also further amplified by different movement durations between non-*rest* classes. Therefore, unless measures are taken to correct the class imbalance, metrics such as accuracy create misleading results and classifiers end up over-specialising to a subset of the classes under investigation.

Further, many studies, e.g. [22–25, 30, 37, 39, 46–49], only make use of a single split of their data. It will be shown herein, that lack of cross-validation further exacerbates representativeness issues as different splits can bias results.

This issue of representativeness and comparability motivates the investigation and development of improved validation methodologies. Further, the re-evaluation of results in the light of improved methods is critical to understanding the true limits of the field.

The second issue is the lack of high performance on large numbers of movements. Since most studies focus on fewer than 15 movements, e.g. [18–35, 149], the NinaPro data sets (40+ movements) offer a valuable resource that allows the community to explore how to improve performance in situations where more movements are required. The results on these data sets, e.g. [37, 48], indicate there is

not yet a classification solution that performs sufficiently well in this context, even when the utilised metrics are biased towards higher performance.

The lack of a sufficiently high-performance solution motivates the exploration of neural networks as a way of improving performance, particularly in the context of many movements. Newer research work also shows the potential benefit of such methods [39, 47, 150]. Further, this indicates that the exploration of supporting methods, such as adaptation to individual subjects, are worthwhile research avenues.

The final issue is the integration of classification solutions into an online context and the practical concerns it produces. Many studies, e.g. [28, 35, 37, 39, 47, 48, 150, 157, 158], demonstrate results only in an offline context. This means that, generally, little emphasis is placed on reduction of the computational overhead of classification. Similarly the offline context does not take into account the practical issues around implementation on embedded hardware, or how performance may differ in different usage contexts. Therefore, the optimisation of classifier design for embedded hardware and the study of practical classification concerns is motivated.

Chapter 3

Methods

This chapter captures methods used to support the work in this thesis that were not discussed in the previous chapter.

3.1 Statistics for Comparison of Techniques

A common problem, in this work, is the need to compare different techniques over multiple subjects controlling for the inherent variability between subjects. In all the data sets tested herein, subjects show a high degree of variation necessitating training of algorithms on each individual. Therefore each subject is effectively an independent data set for the purposes of algorithm evaluation. In the general case, it is often desirable to test whether a technique produces significantly better performance relative to other techniques when evaluated over multiple independent data sets.

In this work, multiple classifiers are often compared across a number of subjects, however, the same methods are applied to comparing other independent variables such as data resampling techniques.

The methodology of Demšar [162] is adopted here to solve this problem as it is an established approach that minimises the assumptions that need to be made about the data. The methodology requires that two or more techniques have been run on a set of data sets and evaluated using a suitable measure. No assumption is necessary about the sampling scheme; however, the results must provide a reliable measure of each technique's performance on each data set. Herein multiple resampling of each data set is performed, typically using cross-validation, to ensure this reliability. This process leads to N independent, matched measures (one for each data set) for each of the k techniques under review. The source of variance is, therefore, the differences in performance over the data sets.

3.1.1 Comparison of Two Techniques

In the special case where the number of techniques $k = 2$, the Wilcoxon signed-rank test is used [163]. The test is non-parametric and ranks the difference in performance between the two techniques for each data set, ignores the signs, and compares the ranks for positive and negative differences.

The positive rank sum is denoted R^+ and calculates the sum of ranks on which the second algorithm outperforms the first:

$$R^+ = \sum_{d_i>0} rank(d_i) + \frac{1}{2} \sum_{d_i=0} rank(d_i) \tag{3.1}$$

where d_i is the difference between the performance scores of the two techniques on data set i.

The negative rank sum is denoted R^- and calculates the sum of ranks on which the first algorithm outperforms the second:

$$R^- = \sum_{d_i<0} rank(d_i) + \frac{1}{2} \sum_{d_i=0} rank(d_i) \tag{3.2}$$

When the techniques perform the same ($d_i = 0$) they are split evenly between the sums and one is ignored if there is an odd number of such occurrences.

The test statistic is calculated as:

$$z = \frac{min(R^+, R^-) - \frac{1}{4}N(N+1)}{\sqrt{\frac{1}{24}N(N+1)(2N+1)}} \tag{3.3}$$

When N is high this statistic is approximately normally distributed, at lower N a table of exact critical values may be consulted [164].

Where the Wilcoxon signed-rank test is used in this work, it is preferred to the paired t-test due to the relatively low sample size (in terms of data sets). The t-test requires the differences between the two tested techniques to be normally distributed for samples sizes of $<\sim 30$; an assumption for which there is no provision for, given the complex relationship between experimental setups, collected data and the techniques under investigation. The Wilcoxon signed-rank test is also more robust to outliers [162].

3.1.2 Comparison of Multiple Techniques

It is possible to perform tests such as the Wilcoxon signed-rank test on all pairs of techniques under consideration on a specific problem. However, when many such

tests are needed the likelihood of incorrectly rejecting at least one null hypothesis (that two techniques are not significantly different) increases. Formally, multiple hypothesis testing of this nature requires intervention to control the family-wise error.

Repeated-measure Analysis Of Variance (ANOVA) [165] is a standard statistical method for testing the differences between multiple matched sample means. However, it is not appropriate in this thesis' context for two reasons. The first is that it assumes the samples are drawn from normal distributions. This assumption *may* be valid when comparing strictly across subjects (with no other modifications) however when other modifications are present, this assumption is unlikely to hold true. Second, ANOVA assumes sphericity, i.e. that the variances of the differences between all combinations of matched samples are the same. An assumption that cannot be taken for granted due to the diversity of techniques that are investigated in any given analysis herein. For example, it cannot be assumed that the variance of the performance differences between a neural network and an SVM is the same as the variance of the differences between two completely different neural network designs.

The Friedman test [166] is instead adopted to test whether there is a significant difference in the performance of different techniques. The Friedman test is a non-parametric equivalent of the repeated-measures ANOVA. The test ranks each technique for each data set separately, with 1 being the best rank and N being the worst, and computes the average rank for each technique R_j. The null hypothesis states that all techniques are equivalent, and therefore, their average ranks should be the same.

The base Friedman statistic is computed as:

$$\chi_F^2 = \frac{12N}{k(k+1)} \left[\sum_j R_j^2 - \frac{k(k+1)^2}{4} \right] \tag{3.4}$$

Iman and Davenport [167] have improved this statistic by removing undesirable conservativeness leading to the following test statistic used here:

$$F_F = \frac{(N-1)\chi_F^2}{N(k-1) - \chi_F^2} \tag{3.5}$$

which follows the F-distribution with $k-1$ and $(k-1)(N-1)$ degrees of freedom. If this statistic allows for rejection of the null hypothesis at the given significance level, then it is possible to proceed with post-hoc testing to determine whether a technique significantly outperforms the others under consideration.

Following Demšar's recommendation [162], the post-hoc Holm step-down procedure [168] is adopted here.

The Holm step-down procedure requires that the following test statistic be computed:

$$z = \frac{R_i - R_j}{\sqrt{\frac{k(k+1)}{6N}}} \tag{3.6}$$

for each technique j being compared to the control technique i. The statistic is normally distributed and therefore, can be mapped to a corresponding p value.

The p values generated from the above must then be ordered from most significant to least significant such that $p_1 \leq p_2 \leq .. \leq p_{k-1}$.

Each p_i value is then compared, in turn, to a modified significance level:

$$\frac{\alpha}{k-i} \tag{3.7}$$

where α is the desired significance level (here always 5%) and i is the index of the compared p value. If the p value is less than the modified significance level, then the corresponding null hypothesis is rejected, and the next comparison can take place. If any null hypothesis cannot be rejected then all remaining null-hypotheses are retained as well.

3.2 Machine Learning Algorithms

3.2.1 Support Vector Machines

The literature has shown Support Vector Machine (SVM) to be good at sEMG hand movement classification tasks [26, 37, 48]. This work uses SVMs as a baseline alongside a comparison to previous work.

The SVM is a supervised learning algorithm that may be used for classification, regression or outlier detection (a special case of classification) [169] although in this work it is only used for classification.

An SVM constructs an optimal hyperplane (or a set thereof) based on some decision criteria in high dimensional space that, for classification purposes, allows splitting the input data into

$$A \vee B \tag{3.8}$$

or

$$A \vee \bar{A} \tag{3.9}$$

where A and B are classes and $\bar{A}$ represents not belonging to the class A. A high

dimensional space is used because the assumption is made that if the set were linearly separable in the original space, then a simpler algorithm would be used [170].

The hyperplane is defined as:

$$f(x) = \beta_0 + \beta^T x \tag{3.10}$$

where x is the support set (closest training examples to the plane), β is the weight vector and β_0 is the bias. There are, therefore, an infinite number of representations of the hyperplane so by convention the representation satisfying

$$|\beta_0 + \beta^T x| = 1 \tag{3.11}$$

is used.

For the purposes of this work, it cannot be assumed that the data is linearly separable; therefore, it is necessary to use a soft-margin to design the hyperplane. A soft-margin is a trade-off between training error and margin. A soft-margin is implemented by using the hinge loss function [170]:

$$max(0,\ 1 - y_i(\beta x_i - \beta_0)) \tag{3.12}$$

where y_i is a class label which takes the value 1 or -1. This function returns 0 if x_i is on the correct side of the hyperplane and is proportional to the distance from the hyperplane otherwise.

The classifier may then be trained by minimising

$$\left(\frac{1}{N}\sum_{i=1}^{N} max(0,\ 1 - y_i(\beta x_i - \beta_0))\right) + \lambda||\beta||^2 \tag{3.13}$$

where λ is a hyperparameter that determines the trade-off between increasing the margin and ensuring points lie on the correct side of the hyperplane.

An SVM in this form is, therefore, strictly a binary classifier. There are two main ways to apply SVMs to multiclass classification. The first (and more scalable) approach is to treat the problem as a set of "one vs rest" problems, training an SVM to distinguish between one class and all the others then running all SVMs at test time and selecting the class with the largest margin or confidence. The alternative is "one vs one" i.e. to consider all pairs of classes and use a voting system to determine the final classification [170].

A kernel function is used to reduce the computational load of the algorithm by allowing dot products in the high dimensional space to be computed in terms

of the original lower dimensional space.

Assume there exists a mapping:

$$\varphi : \mathbb{R}^n \rightarrow \mathbb{R}^m \tag{3.14}$$

where $\mathbb{R}^n$ is the original input space and $\mathbb{R}^m$ is the higher dimensional space the hyperplane will be designed in. Therefore for some x_1 and x_2 in $\mathbb{R}^n$ the kernel k computes:

$$k(x_1, x_2) = \varphi(x_1) \cdot \varphi(x_2) \tag{3.15}$$

without needing to know φ or to compute in the higher dimensional space (providing an appropriate kernel is selected).

The vectors that define the hyperplane can then be determined as linear combinations of points in the input data X. Figure 3.1 demonstrates this mapping function and the differences between the decision boundary made by the hyperplane in low dimensional space (left) and high dimensional space (right).

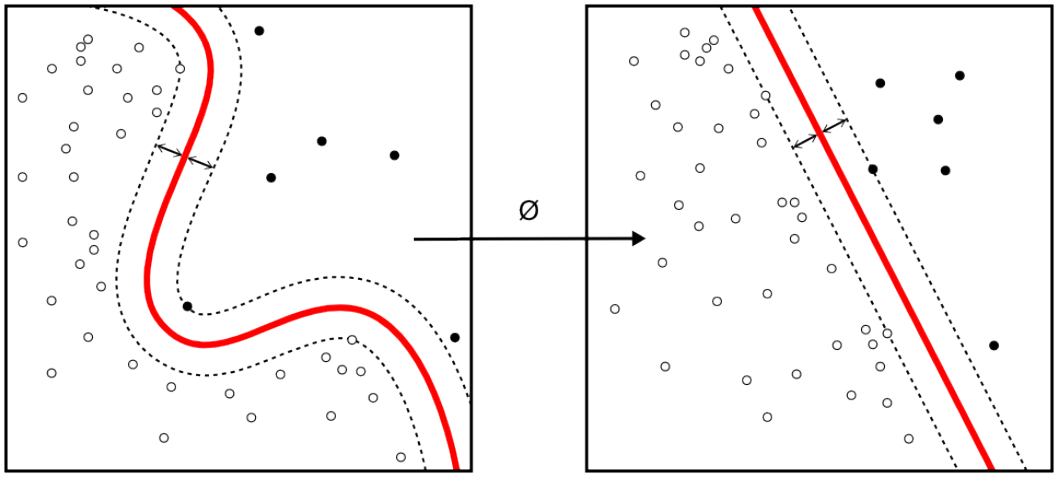

Figure 3.1: Visualisation of kernel mapping ($\oslash = \varphi$) from low dimensional input space to high dimensional space where the hyperplane(s) are designed [171].

During training, the best hyperplane(s) are selected according to a loss function. The Hinge Loss function is typically used for classification problems which allows the training data to be used to select the hyperplane(s) with the maximum margin to the two classes.

Once the hyperplane(s) have been learnt a new input's proximity to the hyperplane(s) can be determined for prediction of class, confidence or margin which may then be used as part of a multiclass scheme or on its own.

The Support Vector Machine - Radial Basis Function Kernel (SVM-RBF) is a

variant of particular note in this work that uses a Radial Basis Function (RBF) as its kernel. The RBF is Gaussian function expressed as

$$exp(-\frac{||x - v_i||^2}{2\sigma^2}) \tag{3.16}$$

where x is the input (vector), v_i is the ith support vector and σ is the standard deviation of the kernel.

3.2.2 K Nearest Neighbours

K-Nearest Neighbours (KNN) is a one of the simpler machine learning algorithms; it is used in this work as a classifier, although it can also be applied to regression problems. It is non-parametric and defers all computation until test time. These properties often makes it a useful algorithm to use as a benchmark but can present speed and memory issues at test time depending on the problem.

In the classification setting the algorithm outputs a class label based on a majority vote of the K closest training examples where each example votes for its class label. The hyperparameters are the value of K and the metric used to measure distance for determining the closest neighbours. A common distance measure used is the Euclidean distance, which can be calculated as:

$$f(x_1, x_2) = \sqrt{\sum_{i=1}^{D} (x_{1,i} - x_{2,i})^2} \tag{3.17}$$

where $f(x_1, x_2)$ is the distance between data points x_1 and x_2 while D is the dimensionality of the data, $x_{1,i}$ indicates the dimension i of the data point 1. Other commonly used distance measures are the Minkowski and Mahalanobis distances. A naive implementation, therefore, must search all other data points in the training set to determine the closest neighbours. Typically, however, data structures such as k-d trees are used to reduce the necessary computation by reducing the number of comparisons required.

It is also possible to introduce weighting on the distance metrics, either a-priori or with gradient learning methods.

3.2.3 Hidden Markov Models

The Hidden Markov Model (HMM) [172] is a statistical model based on Markov chains [173]. This work focuses only on first-order Markov chains, which are used

as a solution for online smoothing. HMM's make the assumption that:

$$P(S_t|S_{t-1}, ..., S_1) = P(S_t|S_{t-1}) \tag{3.18}$$

That is, the conditional probability distribution of the state S at the time t depends only on the state at the time $t-1$ and thus is independent of all prior states.

The HMM extends the Markov chain by introducing the idea of hidden states and observable states (also known as emissions). A Markov chain underlies the model but the states (S_t) may not be directly observed and instead must be inferred from observations. This makes the HMM a useful tool for representing probability distributions over a sequence of observations.

The joint distribution of a sequence of states and observations can be factored as

$$P(S_1, ..., S_T,\ Y_1,\ ..., Y_T) = P(S_1)P(Y_1|S_1)\prod_{t=2}^{T}(P(S_t|S_{t-1})P(Y_t|S_t))) \tag{3.19}$$

where S is the state, Y is the observation and T is the length of the sequence. This factorisation allows graphical visualisation of the conditional independence relations in the HMM, as shown in Figure 3.2.

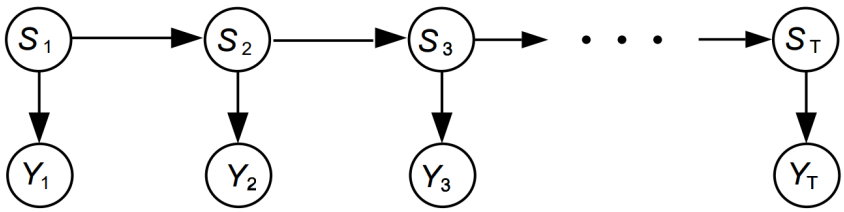

Figure 3.2: Visualisation of conditional independence of relations in a HMM, structure is the same as a Bayesian network.

Figure 3.3 shows an example of a HMM with two hidden states and three observable emissions for illustration purposes.

Given a set of K discrete states, a transition matrix A can be used where $A \in \mathbb{R}^{K\times K}$ with each element representing the state transition probability for the current time step t given the state at $t-1$:

$$A_{rc} = P(S_{t,\,c} = 1|S_{t-1,\,r} = 1) \tag{3.20}$$

where r is the row index (the state at $t-1$) and c is the column index (the state at

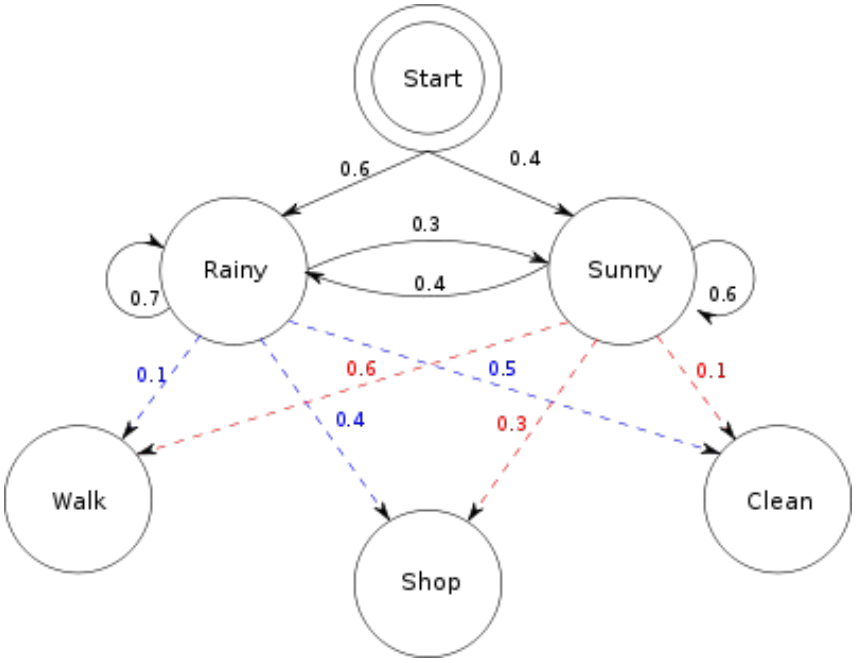

Figure 3.3: Example of a HMM. The "Start" state shows the initial prior probabilities $P(S_1)$ of the hidden states, solid edges indicate state transition probabilities, dashed edges show emission probabilities, and circles are states and emissions [174].

t) of A. A also satisfies the following:

$$\sum_{c=1}^{K} A_{rc} = 1, \quad for\ all\ r \tag{3.21}$$

$$0 \leq A_{rc} \leq 1 \tag{3.22}$$

The conditional distribution is therefore given by:

$$P(S_t|S_{t-1}, A) = \prod_{r=1}^{K}\prod_{c=1}^{K} A_{rc}^{(S_{t,\ c}, S_{t-1,\ r})} \tag{3.23}$$

The emissions matrix B defines the probability of each of possible observation (of which there are N) for each of the K states. It, therefore, takes the form $B \in \mathbb{R}^{K\times N}$ and satisfies:

$$\sum_{c=1}^{K} B_{rc} = 1, \quad for\ all\ r \tag{3.24}$$

$$0 \leq B_{rc} \leq 1 \tag{3.25}$$

Finally, the joint probability can, therefore, be restated to encompass all the

necessary parameters and formulated in a more useful way:

$$P(S_1, ..., S_T,\ Y_1,\ ..., Y_T|A,\ B,\ P(S_1)) = P(S_1)\left(\prod_{t=2}^{T} P(S_t|S_{t-1},\ A)\right)\prod_{t=1}^{T} P(Y_t|S_t,\ B) \tag{3.26}$$

3.3 Evaluation Metrics

3.3.1 Binary Case

When evaluating binary classification problems, a confusion matrix can be used, as shown in Table 3.1. This layout presents a convenient and intuitive way to reason about how the classifier performs from different perspectives.

	Positive Label	**Negative Label**
Predicted Positive	True Positive (TP)	False Positive (FP)
Predicted Negative	False Negative (FN)	True Negative (TN)

Table 3.1: Confusion matrix for binary classification.

The meaning of each entry in the table can, therefore, be formalised in several ways. Here the definitions used are the same as Sokolova and Lapalme [175]:

- True Positive (TP) is the number of correctly recognised positive class examples
- True Negative (TN) is the number of correctly recognised negative class examples
- False Positive (FP) is the number of negative class examples incorrectly classified as the positive class
- False Negative (FN) is the number of positive class examples incorrectly classified as the negative class

Common metrics may, therefore, be described in terms of the Table 3.1:
Accuracy is the most commonly used metric and is defined as:

$$Accuracy = \frac{Correct\ Classifications}{Total\ Examples} = \frac{\mathrm{TP} + \mathrm{TN}}{\mathrm{TP} + \mathrm{FN} + \mathrm{FP} + \mathrm{TN}} \tag{3.27}$$

which is used to represent the overall effectiveness of the classifier but does not account for imbalance in the numbers of examples belonging to each class or, the cost of False Negatives and False Positives.

Precision calculates how often the classifier is correct when it predicts the positive class and is, therefore, a measure of agreement between the data labels and the positive predictions of the classifier:

$$Precision = \frac{Correct\ Positive\ Classifications}{Positive\ Predictions} = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FP}} \tag{3.28}$$

Recall or **Sensitivity** measures how effectively the classifier identifies positive labels :

$$Recall = \frac{Correct\ Positive\ Classifications}{Positive\ Labels} = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FN}} \tag{3.29}$$

F-score computes a weighted harmonic mean of precision and recall to determine the relationship between the data's positive labels and the predicted positive labels:

$$F - score = (1 + \beta^2)\frac{Precision \cdot Recall}{(\beta^2 \cdot precision) + Recall} = \frac{(1 + \beta^2)\mathrm{TP}}{(1 + \beta^2)\mathrm{TP} + \beta^2\mathrm{FN} + \mathrm{FP}} \tag{3.30}$$

where Recall is assigned β times more importance than Precision.

Specificity measures how effectively the classifier recognises negative labels:

$$Specificity = \frac{Correct\ Negative\ Classifications}{Negative\ Labels} = \frac{\mathrm{TN}}{\mathrm{FP} + \mathrm{TN}} \tag{3.31}$$

Receiver Operating Characteristic (ROC) is a graphical representation of the trade-off between Sensitivity and Specificity, showing how changing the settings of a classifier impacts the two characteristics. The **Area Under Curve (AUC)** of this graph can be calculated to provide a threshold and scale invariant measure of performance on the range $[0, 1]$ with 1 being perfect classification and 0 being complete misclassification. The AUC is a measure of the average sensitivity for all possible specificity values [176].

The AUC is also directly related to the Mann-Whitney U Test statistic [177] by the following equation:

$$AUC = \frac{U}{N_p * N_n} \tag{3.32}$$

where U is the Mann-Whitney U Test statistic, N_p is the number of positive class examples and N_n is the number of negative class examples.

3.3.2 Multiclass Case

In the multiclass case, each class may be evaluated using its own confusion matrix calculated in a one-vs-all manner or the definitions of TP, TN, FP and FN may be updated to account for multiple classes. The major difference in the multiclass

case is that the idea of negative labels is less precise since a negative of one class is the positive of another.

Therefore in the multiclass case the definitions of TP, TN, FP and FN must be updated to consider a specific class rather than a positive and negative:

- TP_i is the number of correctly recognised class examples of class i
- FP_i is the number of times the classifier predicts class i and is incorrect
- FN_i is the number of times the label is class i but the classifier prediction is incorrect
- TN_i is no longer a useful definition in the multiclass case because the idea of a negative class is no longer used and so TN_i is omitted from consideration

These definitions capture the fact that the multiclass case requires an extension of metric calculations to the rows and columns of the confusion matrix. Table 3.2 shows how these definitions function in a three-class classification problem.

		Actual Class		
Predicted Class		**A**	**B**	**C**
	A	TP_A	FN_B/FP_A	FN_C/FP_A
	B	FN_A/FP_B	TP_B	FN_C/FP_B
	C	FN_A/FP_C	FN_B/FP_C	TP_C

Table 3.2: Confusion matrix for three class classification. In general, rows contain are either TP or FP for the predicted class and columns are TP or FN for the actual class.

Metrics in the multiclass case may, therefore, be calculated as either a micro-average or macro-average. The macro-average is the average of the measure across classes while the micro-average obtains cumulative totals across all classes before calculation. The macro-average, therefore, treats all classes equally while the micro-average weights towards classes with more examples.

Micro-average Accuracy may, therefore, be defined as:

$$Accuracy_{mi} = \frac{Total\ Correct\ Classifications}{Total\ Examples} = \frac{\sum_{i=1}^{C} \mathrm{TP}_i}{\sum_{i=1}^{C} (\mathrm{TP}_i + \mathrm{FN}_i)} \tag{3.33}$$

where C is the number of classes. Note TN and FP are omitted since they are equivalent to the TP and FN of other classes.

Macro-average Accuracy is defined as:

$$Accuracy_{ma} = \frac{1}{C} \sum_{i=1}^{C} \frac{\mathrm{TP}_i}{\mathrm{TP}_i + \mathrm{FN}_i} \tag{3.34}$$

which removes the relative class example frequency weighting from the equation.

Micro-average Recall is, therefore, the same as Micro-average Accuracy:

$$Recall_{mi} = \frac{\sum_{i=1}^{C} \mathrm{TP}_i}{\sum_{i=1}^{C} (\mathrm{TP}_i + \mathrm{FN}_i)} \tag{3.35}$$

Macro-average Recall, similarly, is the same as Macro-average Accuracy:

$$Recall_{ma} = \frac{1}{C} \sum_{i=1}^{C} \frac{\mathrm{TP}_i}{\mathrm{TP}_i + \mathrm{FN}_i} \tag{3.36}$$

Micro-average Precision is defined as:

$$Precision_{mi} = \frac{\sum_{i=1}^{C} \mathrm{TP}_i}{\sum_{i=1}^{C} (\mathrm{TP}_i + \mathrm{FP}_i)} \tag{3.37}$$

Macro-average Precision is defined as:

$$Precision_{ma} = \frac{1}{C} \sum_{i=1}^{C} \frac{\mathrm{TP}_i}{\mathrm{TP}_i + \mathrm{FP}_i} \tag{3.38}$$

Micro-average F-score and **Macro-average F-score** retain the same definition in terms of Precision and Recall (Equation 3.30) except with the Precision and Recall calculated with the appropriate averaging technique.

Specificity and Receiver Operating Characteristic (ROC) related measures, in general, are considered not well developed for the multiclass case [175, 178] and therefore are not considered here.

3.4 Data Labelling

When gathering sEMG data, it is typically necessary to label where, in the data stream, a subject is performing each movement/gesture, i.e. to delimit each gesture. Delimitation can be achieved by having an expert manually label data with an appropriate aid such as a video of the experiment with time stamps; however, this is human time intensive so automated methods are typically employed.

Open loop methods, such as flagging X seconds after a subject has been asked to assume a gesture, are a standard solution. Open loop methods, however, are not as accurate in their delimitation compared to methods that incorporate additional information although open loop methods can significantly speed up the labelling process relative to other techniques.

An alternative method sometimes used for delimitation is the maximisation of

the Generalised Likelihood Ratio (GLR) [179] to produce likely delimitations for *rest*-movement-*rest* cycles. This method generates a boundary closer to what an expert would label but potentially captures unintentional movement or undesired transitory effects.

The GLR method, as applied to EMG [37], is described in the following section.

First, the EMG is whitened [180]. Then it is assumed that the distribution of signals during movement and when in the rest state are Gaussian but with different parameters from each other, which will be determined as part of the process. These parameters are:

- $\theta_0 = (\mu_0,\ \sigma_0^2)$ is the rest distribution with mean μ_0 and standard deviation σ_0
- $\theta_1 = (\mu_1,\ \sigma_1^2)$ is the movement distribution with mean μ_1 and standard deviation σ_1

The assumption is made that the movement's variance is higher than the *rest* state and that, since the data is whitened, the means are 0.

The corresponding probability density function for some point x in either Gaussian distribution is:

$$p(\theta, x) = \frac{1}{\sqrt{2\pi\sigma}} \exp\left[-\frac{(x-\mu)^2}{2\sigma^2}\right], -\infty < x < \infty \tag{3.39}$$

Two hypotheses can then be tested for a proposed start and end of the movement within the segment:

- H_0 that there is no movement: all data is rest (i.e. data is drawn from the same distribution)
- H_1 that a rest-movement-rest sequence occurs (i.e. data is drawn from the distributions in sequence)

Correspondingly the likelihoods for a time period of N samples are:

$$H_0 \Rightarrow L(\theta_0) = p(\theta_0, x_{[0,\ N]}) \tag{3.40}$$

$$H_1 \Rightarrow L([\theta_0, \theta_1, \theta_0]) = p(\theta_0, x_{[0,\ r_1-1]})p(\theta_1, x_{[r_1,\ r_2]})p(\theta_0, x_{[r_2+1,\ N]}) \tag{3.41}$$

Where r_1 and r_2 are the start and end of the movement respectively.The likelihood ratio can then be expressed as:

$$\frac{L([\theta_0, \theta_1, \theta_0])}{L(\theta_0)} = \frac{p(\theta_0, x_{[0,\ r_1-1]})p(\theta_0, x_{[r_1,\ r_2]})p(\theta_0, x_{[r_2+1,\ N]})}{p(\theta_0, x_{[0,\ N]})} \tag{3.42}$$

This can be simplified by ignoring locations not marked as movement because these will evaluate to 1, therefore the objective function is:

$$\max \left[\prod_{k=r_1}^{r_2} \frac{p(\theta_1, x_t)}{p(\theta_0, x_t)} \right], \; 0 < r_1 < N, \;\; r_1 \leqslant r_2 < N \tag{3.43}$$

The distributions θ_0 and θ_1 may then be estimated from the data:

$$\hat{\mu_0} = \frac{1}{(r_1 - 1) + (N - r_2 - 1)} \left(\sum_{k=1}^{r_1 - 1} x_k + \sum_{k=r_2+1}^{N} x_k \right) \tag{3.44}$$

$$\hat{\sigma_0} = \frac{1}{(r_1 - 1) + (N - r_2 - 1)} \left(\sum_{k=1}^{r_1 - 1} (x_k - \hat{\mu})^2 + \sum_{k=r_2+1}^{N} (x_k - \hat{\mu})^2 \right) \tag{3.45}$$

$$\hat{\mu_1} = \frac{1}{N} \sum_{k=r_1}^{r_2} x_k \tag{3.46}$$

$$\hat{\sigma_1^2} = \frac{1}{N} \sum_{k=r_1}^{r_2} (x_k - \hat{\mu})^2 \tag{3.47}$$

Lastly by utilising the log-likelihood ratio the computation required can be reduced and potential for error due to precision mitigated:

$$\ln \prod_{k=r_1}^{r_2} \frac{p(\theta_1, x_t)}{p(\theta_0, x_t)} = \frac{1}{2} \Big[N_m \ln \frac{\sigma_0}{\sigma_1} + \frac{1}{\sigma_0^2} \sum_{k=r_1}^{r_2} (x_t - \mu_0)^2 - \frac{1}{\sigma_1^2} \sum_{k=r_1}^{r_2} (x_t - \mu_1)^2 \Big] \tag{3.48}$$

$$N_m = r_2 - r_1 + 1 \tag{3.49}$$

An exhaustive search may then be used to find the points that produce the highest likelihood with further simplification because θ_1 is estimated from the data:

$$\ln \prod_{k=r_1}^{r_2} \frac{p(\theta_1, x_t)}{p(\theta_0, x_t)} = \Big[N_m \ln \frac{\sigma_0}{\sigma_1} + \frac{1}{\sigma_0^2} \sum_{k=r_1}^{r_2} (x_t - \mu_0)^2 - N_m \Big] \tag{3.50}$$

In practice assumptions may be made on the *rest* distribution, priors introduced on the likelihood of any one example belonging to a particular class and minimum movement duration added to reduce the search space and refine the predictions.

Chapter 4

Robust Feature-Based Classification

4.1 Introduction

The focus of this chapter is improving the fundamental building blocks of sEMG based classification. The key novel contributions are an analysis of and set of solutions for, a host of recurrent problems in the field that frequently negatively impact how representative results are of expected application-time performance. Another contribution is that, with the solutions to representativeness issues in place, appropriate data resampling techniques can be used to produce state-of-the-art performance with feature-based classification methods. Finally, a novel evaluation of methods for choosing movements to classify from a superset is presented, which allows person-specific adaptation of the movement set in order to optimise performance for an individual.

For any advance in a field to be meaningful, the methods it is built on must be robust and reproducible, or the contribution cannot be appropriately evaluated. There are several recurrent issues in the sEMG classification field that are tackled here to lay the groundwork for performance improvements in this and future chapters.

While not directly tackled by the work in this chapter, one of the most fundamental issues is access to the data used for a study. Many studies [18, 21, 27, 29, 33, 35, 43, 140–144, 181] do not make their data available, which means their work cannot be reproduced. Here a well known open-source dataset [182] is used so that the work can be reproduced by other researchers. Another vital issue is the usage of smaller numbers of movements in studies, e.g. 4-7 [18–28] or 9-12 movements [29–35] which limits the utility of any resultant classifier since many

applications benefit from having access to more movements even if only to ensure performance is optimised for the particular setup or individual. Here 53 movements are classified, which improves the potential of the resultant classifiers.

In terms of contributions to representativeness, this chapter focuses on the evaluation of classifiers. The first issue is making an appropriate choice of metric; most studies, e.g. [20, 23, 27, 30, 34–36, 38–40, 182], by necessity, employ *rest*-movement-*rest* cycles as part of their experiment design and label their data as such. This leads to the *rest* class being an anomaly in that it is explicitly treated differently to all other movement classes and, unless special measures are introduced, ends up with significantly more data on it than other classes. Even between movements, there can be significant variation in the numbers of examples due to the subject not taking the same time to perform each movement. Together this leads to data imbalance in most data sets, which can skew measures and classifier training.

Many studies also use accuracy [25, 27, 37, 41–43] or state the use of "accuracy" / "classification accuracy" without an appropriate equation [18, 20, 23, 26, 28, 29, 35, 37–40, 44, 45]. When there is data imbalance in these studies, accuracy leads to a result biased towards the over-represented classes. Therefore any results are likely to not be reflective of application-time performance since accuracy effectively imposes the prior that the class distribution at test time is the same as at application-time which is unlikely to be the case due to the structured nature of experiments.

Therefore the macro-average accuracy is presented and evaluated in this chapter as a more representative alternative to accuracy. It assumes that the application-time distribution is unknown and therefore, weights each class equally for the evaluation of classifiers better representing expected performance. Note that some studies [21, 24, 26, 29, 42–44] exclude *rest* from classification however since the majority of applications desire knowledge of when a subject is not performing a movement it is assumed here that it is important to be able to classify *rest*. This can make a significant difference to the performance profiles of a study and therefore studies that use *rest* are generally incomparable with ones that do not.

In the same vein, many studies [22–25, 30, 37, 39, 45–49] do not cross-validate their results. Cross-validation is known to be a useful technique for reporting representative results [183, 184] and is shown in this chapter to be applicable to performance evaluation in this context. Further, since random fold selection is not compatible with data processing techniques, such as windowing, commonly used in the field; a stratification technique is presented and validated which ensures performance results are representative of the result found if trialling the entire

population of possible folds.

Using these improved validation techniques, the original, well cited, benchmark [37] on this data set is re-evaluated and shown to produce significantly lower performance compared to the original. This re-evaluation provides new insight into the data, and other data sets like it, which lead to the application of enhanced data preprocessing in the form of data resampling techniques which improved performance by over 10% absolute macro-average accuracy compared to the original. These techniques are fully benchmarked to present a new baseline. The trends in the new baseline are discussed to highlight important differences between it and the original.

Finally, a novel evaluation of movement sub-selection techniques (now published in IEEE EMBC [50]) is presented, which demonstrates the possibility for adapting the movements used in an application to a specific subject. This allows quantification of the trade-off between the number of movements classified and the resultant performance as well as demonstrating that some selection methods provide significantly better trade-off ratios than others.

4.2 Data Analysis

NinaPro database 1 [37, 146, 182] is used as a testing ground since the data is open-source and widely used in the literature, making it an ideal case study. At the time of writing the NinaPro databases [182] are the most extensive open-source data sets available for researching sEMG based hand movement recognition or tracking, particularly with regards to the number of different hand movements captured.

The database contains data from 10 Otto Bock MyoBock 13E200 sEMG sensors attached to 27 healthy subjects performing 10 repetitions of 52 different hand movements. The *resting* state returned to between each movement is treated as an additional movement, however, making an effective total of 53 for the context of classification. The database is divided into subjects and sub-divided by experiments where each experiment contains the raw sEMG signals, labels for repetition and movement ID along with *a posteriori* refined version of the labels obtained via use of the Generalised Likelihood Ratio (GLR) test [37]. Some experiments also include data from a CyberGlove that estimates joint angles although these are not used. The data is already synchronised via timestamps collected at experiment time.

The Otto Bock MyoBock 13E200 electrodes sample at 100Hz but also perform RMS filtering which downshifts the spectral properties of the signal to retain infor-

mation from higher frequencies [37] despite the relatively low sampling frequency.

In this chapter, the data D takes the form:

$$D = \left\{ \left(X^{(1)}, Y^{(1)}, R^{(1)} \right), \ldots, \left(X^{(N)}, Y^{(N)}, R^{(N)} \right) \right\} \tag{4.1}$$

where X is a single sample of raw emg data which with an associated movement ID Y and a repetition ID R. Sequential $X^{(i)}$'s are sequential temporally except at the boundaries between exercises.

The prediction problem is then framed as taking in ω_l contiguous X samples and predicting the movement ID Y of the most recent X. Intuitively: predicting the current hand gesture a subject is performing given access to the current sample and a finite amount of past data. Therefore *rest*, or at least non-performance of a movement of interest, must be considered as a separate class.

Since the data is labelled, the first thing it is necessary to establish is the balance of data between different classes and repetitions.

In terms of classes, the data is highly imbalanced. The *rest* class makes up over 50% of the data on each subject due to the experimental procedure using *rest*-movement-*rest* cycles. On top of this, between the non-*rest* movements, there is also a high degree of variability in the number of examples which also varies across subjects; some movements can have over twice as much data on them as others. The variability is caused by the experiment calling for following along with a video with a standardised runtime but with variance in length of the actual recorded movement content coupled with the natural differences between each subject. Plotting the distribution of classes, as in Figure 4.1, highlights the significance of the problem.

This imbalance in the number of examples means the performance metric will be critical to ensuring any results present a meaningful view of classifier performance.

Repetition labels (and therefore, the amount of data per repetition) vary to a lesser degree but still up to 10 seconds over the average of $\sim$ 180 seconds per repetition per subject.

The other issue with the repetition labels is that all the *rest* data is labelled as a separate repetition 0, which can cause reproducibility issues since it is generally desirable to split the data by the repetition number.

In order to obtain further insight into potential issues, data visualisation is an invaluable tool. The EMG signals in the database have 10 non-time dimensions (the number of channels) but as it is more typical to take a windowed segment (see Section 4.3.1) the actual input space to a classifier is typically an order of magnitude higher.

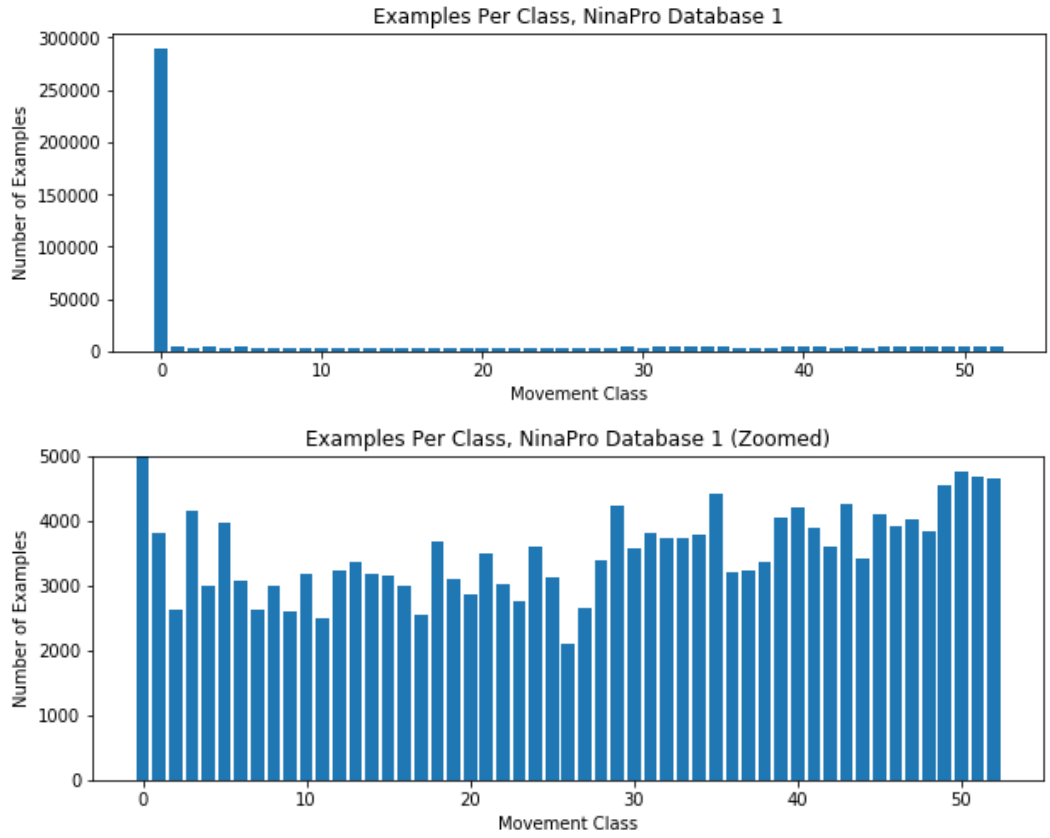

Figure 4.1: Number of examples of each class in NinaPro database 1 for subject 1. The *rest* class (class 0) dominates the examples. All subjects show similar trends.

Principal Component Analysis (PCA) allows visualisation of high dimensional spaces by calculating a transform that changes a data set's variables into a set of linearly uncorrelated variables; vectors orthogonal to one another. The vectors, or "principal components", are calculated such that the first captures the highest possible variance within the data and each successive vector also maximises the variance captured while being orthogonal to all previously computed vectors.

To visualise the data 200ms (20 sample) windows were taken with the movement label being the most recent label within the window, and the first two principal components for all windows were plotted. This process captures 72% of the variance in the data and is shown for Subject 1 in Figure 4.2 representing a typical case.

While PCA does not capture the full variance Figure 4.2 still shows that only a few movements stand out to any degree based on the raw data. Further, while it is expected that the *rest* state/class is relatively low energy compared to other movements, the Figure shows that, for this data set, there is no clear divide that can

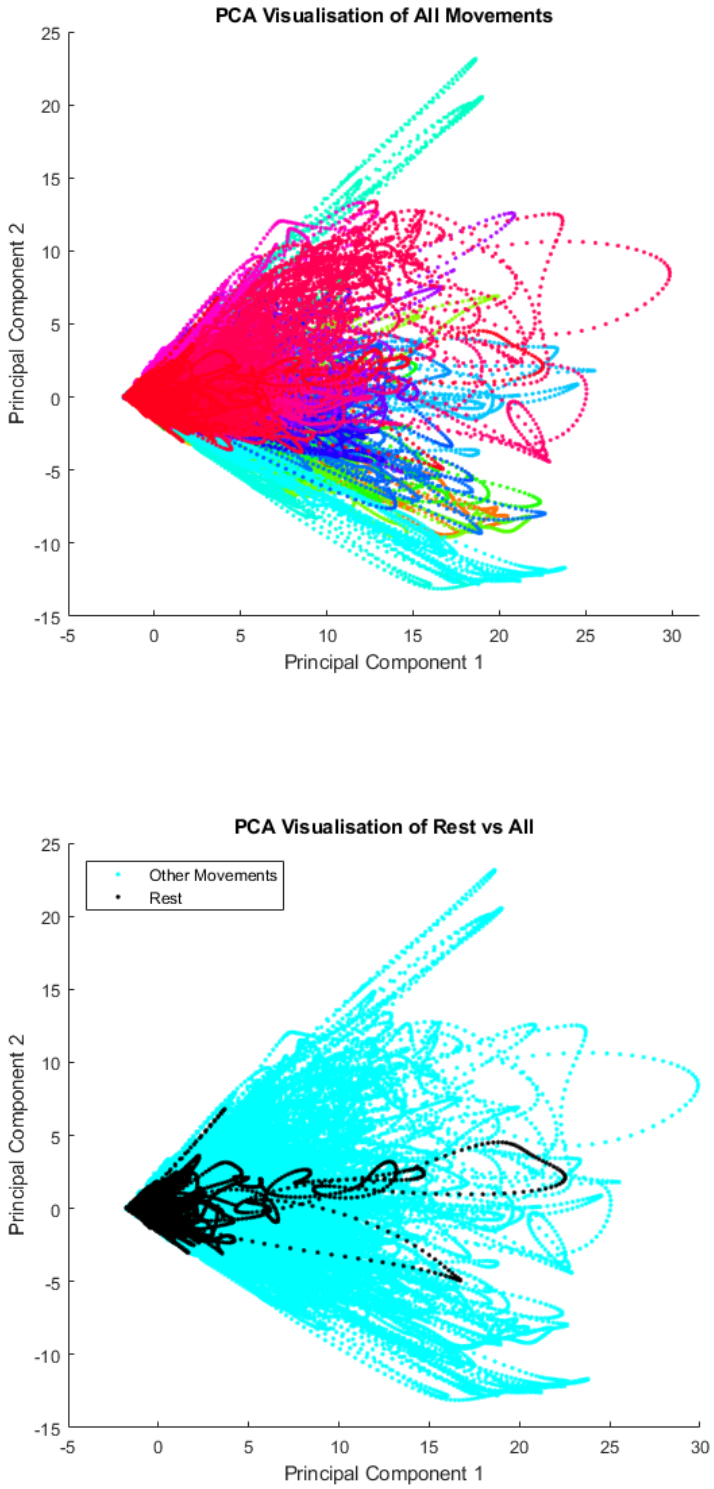

Figure 4.2: PCA Visualisation of 72% of the variance of the 53 movements in the database. The top graph shows movements in different colours, demonstrating a large amount of overlap between movements. The bottom graph shows the same visualisation with *rest* plotted against all other movement classes.

be made. This lack of a clear divide may be caused by the Generalised Likelihood Ratio (GLR) technique used to delimit *rest* and movements combined with subjects not being fully relaxed between some movements.

Another visualisation technique is t-Stochastic Neighbour Embedding (t-SNE) [185, 186], which achieves dimension reduction through iterative stochastic non-uniform transformations of the data set that attempt to preserve the local information at each step. t-SNE uses a "perplexity" term which trades off keeping global information vs local information. The same windowed data as used for the PCA is used under different perplexity values to produce a human intelligible visualisation of the high dimensional topology in Figure 4.3. Since t-SNE performs non-uniform transformations on the data, the axes of the resultant graph have no meaning hence their exclusion. Instead, the graph captures only local relations making precise position lose meaning and distances only useful to show the separation of points. A guide to the interpretation of t-SNE plots can be found in [186].

Figure 4.3 reinforces what was seen in Figure 4.2, showing that *rest* (at least from the raw data) is very similar to the other movement classes. The t-SNE visualisation also shows some movements being fragmented, suggesting distinct stages in the movement or fanned out along a curve suggesting some sort of progression over the movement similar to the strong curves seen in the PCA.

Together these two visualisations inform the reader that the labelling process is imprecise; movement classes with a particular label may not belong to that movement or the movement boundaries may not be as clear cut as the labelling process implies. The Figures 4.2 and 4.3 demonstrate this in that rather than labels representing neat groups they share space with other labels and even within a particular label, there can be a high degree of variance and potential sub-groupings.

4.3 Robust Preprocessing and Evaluation

4.3.1 Windowing

Windowing is a Digital Signal Processing (DSP) technique that can be used to feed past information into classifiers that do not support recursive updates. The typical method used for EMG is to take a segment of data with length ω_l samples and slide it through contiguous segments of the emg data with a step size of ω_{inc}. This is equivalent to using a rectangular window function in other DSP applications, which expresses the expectation that all of the data is equally relevant. This also allows explicit calculation of features or for the classifier to do so automatically without introducing bias. Figure 4.4 illustrates the windowing process.

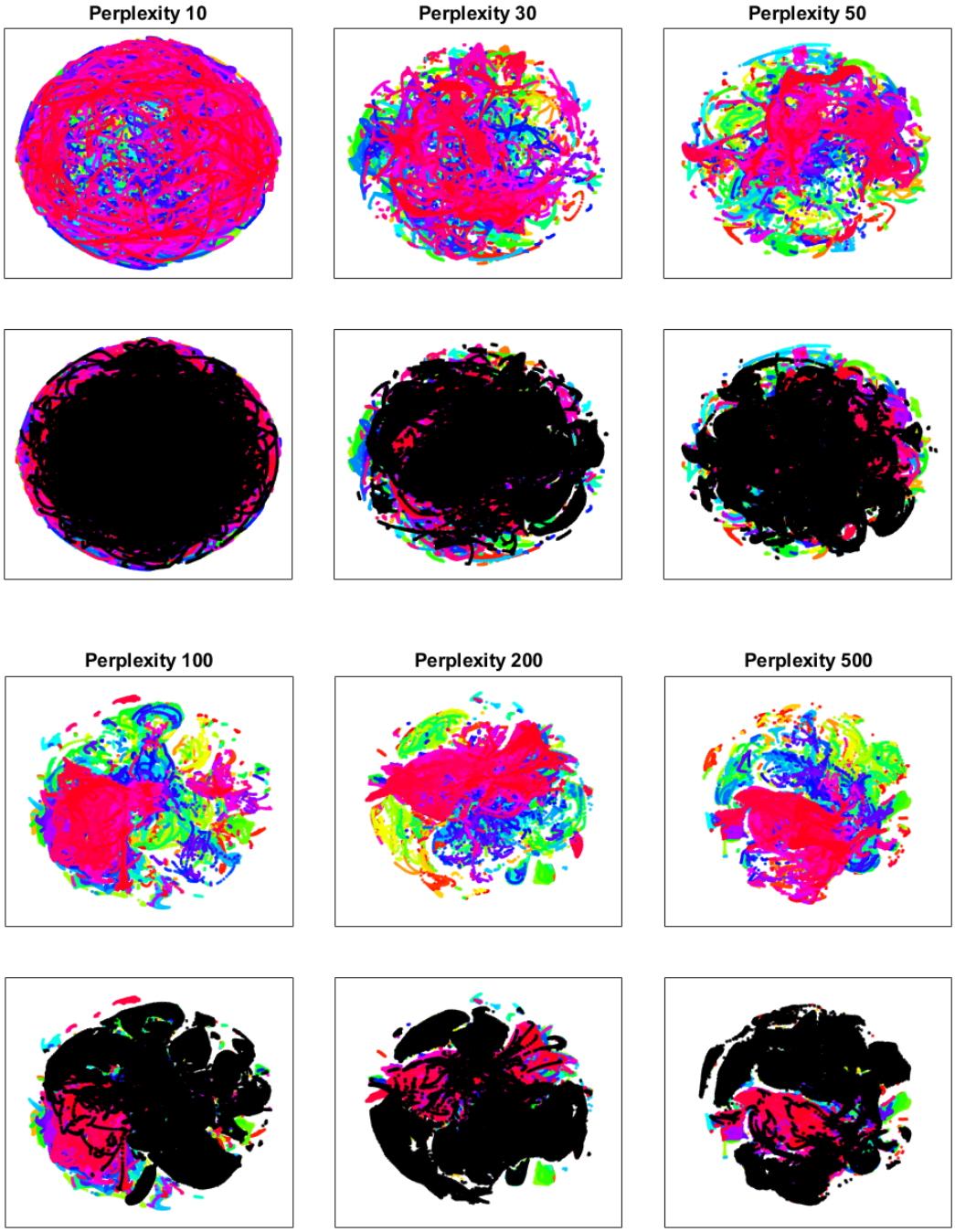

Figure 4.3: Visualisations created using t-SNE with different perplexities. Plots on the first row underneath titles are plotted without *rest* while plots on the lower rows have *rest* plotted in black on top of all other classes. It can be seen that *rest* is similar to most other movements and that some movements are fragmented or follow curves suggesting distinct stages or continuous progression, respectively. Axis labels are removed due to the discontinuity of space in the visualisation.

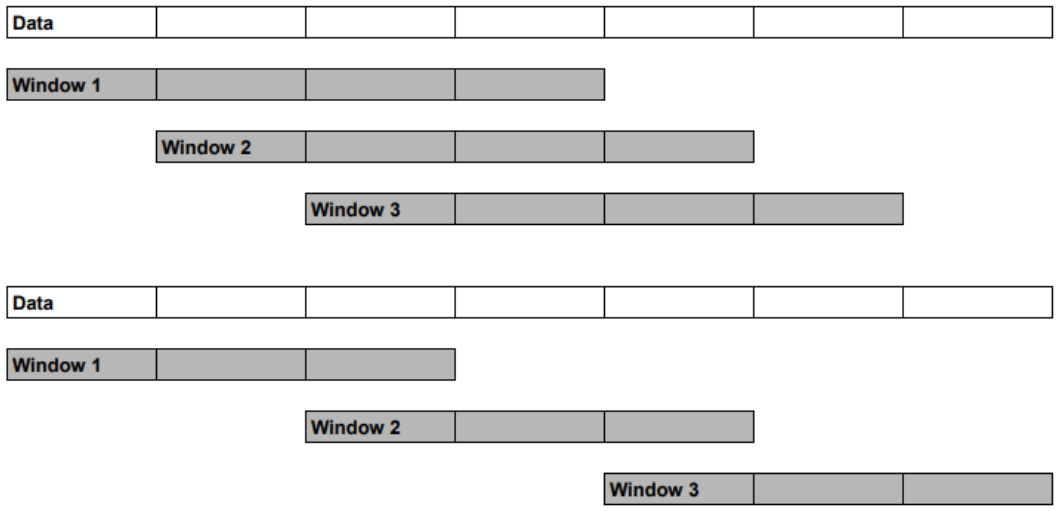

Figure 4.4: Illustration of windowing for two examples; window length ω_l 4 samples and increment of 1 sample (upper) and window length 3 samples and increment of 2 samples (lower).

A common misconception is that the window length is directly proportional to the observed latency in classification. The two are related, but the relationship is highly dependent on other parameters such as the classifier being used and how the window is labelled. In this study, and most EMG classification studies, the window label is taken from the most recent time step, which means a perfect classifier would induce zero latency due to the windowing process. In practice, however, there is likely to be a drop in performance when classifying windows that only include a small percentage of data on a particular movement. This will generally lead to a higher observed latency for longer windows; however, the exact trade-off will be highly dependent on the other parameters of the study. As a corollary, a short window length does not necessarily imply a short latency as performance may be degraded during the onset and at the end of movements when the signal is different to the remainder of the movement.

Therefore, while useful as a rough guideline for expected latency, it is not possible to directly relate the window length and latency without explicit testing. Greater window lengths also tend to improve performance (to a point) and so in environments where latency is important, the actual latency will need to be confirmed experimentally.

When using windowing, it is also necessary to take extra care when splitting up data such as for training and testing. If the window increment is lower than the window length, then nearby windows will share data. This can be seen in Figure 4.4 where all windows share samples with adjacent windows. This means that

the normally recommended usage of random splits for training and testing sets is improper and will lead to a biased result since the assumption of set independence is violated.

Windowing also causes an imbalance in the use of the underlying data since the data at the start and end of a continuous segment is used less than the remainder. The magnitude of the effect is proportional to the length of contiguous segments. This has implications for normalisation and potentially other areas such as data balance.

Here the highlighted issues are dealt with via methodological changes covered in later sections. For the benchmarks, a sliding window with three different window lengths was used: 100ms, 200ms and 400ms. The window increment was kept constant at 10ms. It is, therefore, useful to update the data definition:

$$D = \left\{ \left(X_\omega^{(1)}, Y^{(1)}, R^{(1)} \right), \ldots, \left(X_\omega^{(N)}, Y^{(N)}, R^{(N)} \right) \right\} \tag{4.2}$$

where X_ω is an ω_l by C matrix of sEMG data, ω_l is the window length and C is the number of channels/electrodes for data in the database (10). Y is the movement ID and R is the repetition ID each associated with the most recent time step in the window based on the a-posteriori refined movement labels.

This structure is convenient since it makes the first axis of X_ω a sequence of chronological sEMG observations and the second an index to a particular channel. Figure 4.5 presents a graphical representation to aid intuition.

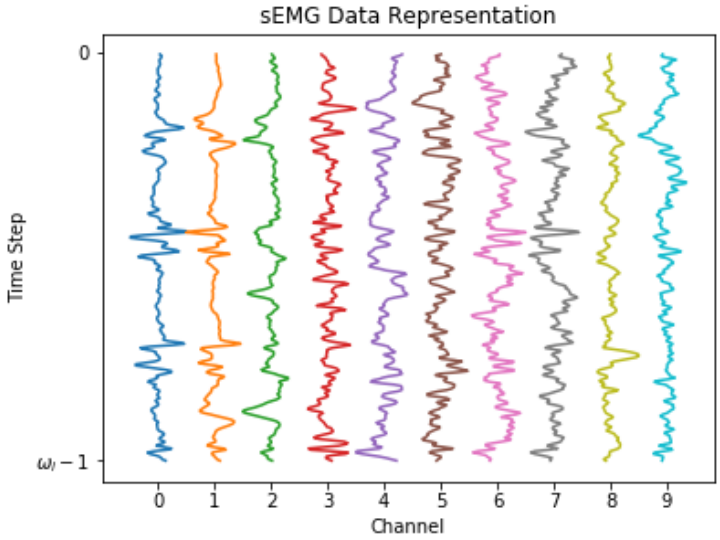

Figure 4.5: Visualisation of a raw sEMG window before feature extraction.

4.3.2 Metrics

Choice of a representative metric or metrics to review performance is potentially the most important choice in any study as an inappropriate selection will invalidate any findings. This is a particularly relevant issue in many EMG studies including the one conducted here (see Section 4.2 and Figure 4.1) because the typical experiment involves *rest*-movement-*rest* cycles and it is normally desirable to classify the *rest*. This is coupled with the fact that often movements take different amounts of time to perform, all of which leads to data imbalance.

The standard metric "accuracy" or sometimes "classification accuracy" for the M class case is defined as:

$$acc = \frac{\sum_{i=1}^{M} \mathrm{TP}_i}{\sum_{i=1}^{M} (\mathrm{TP}_i + \mathrm{FN}_i)} \tag{4.3}$$

where TP_i is the number of correctly predicted examples of class i, FN_i is the number of incorrectly predicted examples of class i and M is the number of classes. See chapter 3.3 for a discussion of multiclass True Positive (TP), True Negative (TN), False Negative (FN) and False Positive (FP).

Many EMG studies explicitly use "accuracy" as a convenient way to represent overall performance [25, 37, 41–43] or state the use of "accuracy" / "classification accuracy" without an appropriate equation [18, 20, 23, 26, 28, 29, 40, 44, 45]. This is problematic because, as evidenced by the equation, "accuracy" does not take into account class balance, which effectively weights each class' contribution by its relative frequency in the test data. This problem with "accuracy" is pointed out as being of particular relevance to myoelectric pattern recognition problems by Ortiz-Catalan *et al.* [187].

On NinaPro database 1 using "accuracy" equates to *rest* contributing to the final accuracy value more than all other classes combined and to weighting longer movements more than shorter ones.

This issue is present in the original benchmark [37] as evidenced by the histogram taken from the original paper shown in Figure 4.6. Specifically, the "accuracy" which is reported in the paper is $\sim 76\%$ despite only 1 of the 53 movements achieving an accuracy greater than 70%.

A more representative measure can be used when it is explicitly assumed that all classes are of equal importance to the final measure (Section 4.3.2 discusses the validity of this assumption). The equal importance of classes can be enforced by taking the mean of the per class accuracy, this is also known as the "macro-average accuracy" and is defined by the following for the M class case:

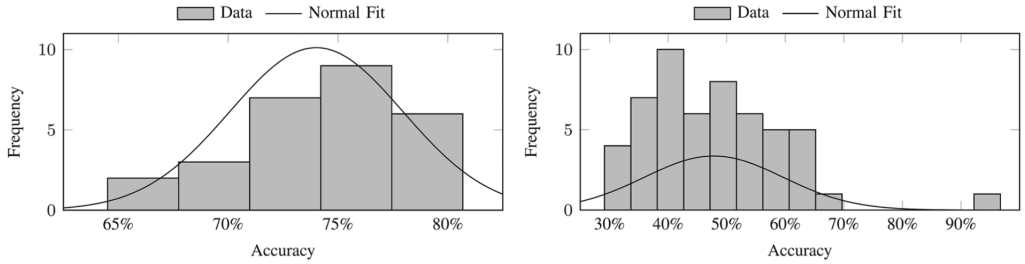

Figure 4.6: Histograms of average accuracies over subjects (left) and movement classes (right) taken from the original benchmark on NinaPro database 1 [37]. The outlier movement class on the right is the *rest* class showing how it is significantly better classified than all other classes.

$$acc_{ma} = \frac{1}{M} \sum_{i=1}^{M} \frac{\mathrm{TP}_i}{\mathrm{TP}_i + \mathrm{FN}_i} \tag{4.4}$$

where TP_i is the number of correctly predicted examples of class i, FN_i is the number of incorrectly predicted examples of class i and M is the number of classes. This equation is equivalent to a macro-average of Recall [188] due to consideration of multiple classes. The term "macro-average accuracy" is used here since intuitively the quantity captures the same ideal as accuracy, which is the expected number of correct predictions on a new set of data with minimal prior assumptions.

For the sake of distinction the original cross-class accuracy metric described in Equation 4.3 shall henceforth be referred to as "micro-average accuracy".

In this work, the macro-average accuracy will be the default metric because it provides a single number representing the ability of a classifier to classify incoming windows of data without any assumptions about the distribution of classes in that new data. Having a single number is also helpful for comparing performance between methods. Note that in the multiple class case, per class accuracy is also equivalent to per class Recall.

The macro-average accuracy is calculated for each subject, and the mean is then taken across subjects to get a final value.

The mean rank is also presented, calculated by treating each subject as a matched sample, which allows controlling for the natural variability between subjects which biases the standard deviation. The mean rank is also necessary for computing whether a performance improvement is statistically significant.

On Equal Class Weighting

One of the assumptions made here is that all classes should be weighted equally for performance validation. This assumption is used because it encodes the fact that the expected distribution of new data is unknown.

There is an argument that for some applications, and particularly for the *rest* class, that there may be a useful prior making it desirable to achieve high performance on one class at the expense of others however if this is desired it can be implemented using a principled approach such as incorporating a weight into the metric:

$$acc_{ma} = \frac{1}{M} \sum_{i=1}^{M} \gamma_i \left(\frac{\mathrm{TP}_i}{\mathrm{TP}_i + \mathrm{FN}_i} \right) \tag{4.5}$$

where γ_i is the weighting for class i. This allows for specific tuning of the metric to the task, unlike the micro-average accuracy which imposes a frequency based weighting which is often not representative of expected performance in contexts outside of a particular known test set.

In any study, it is incorrect to assume the distribution of classes outside the scope of the work in question, and therefore, an unbiased metric is the only defensible solution without additional information.

4.3.3 Validation

Stratified K-fold cross-validation and its variants [183] are essential techniques for ensuring results are reliable [184]. This is also an area of major deficiency with many EMG studies which only use single data splits [22–25, 30, 37, 39, 46–49]; therefore, this thesis provides a more thorough approach to validation to ensure the integrity of results.

Validation is even more important when used in studies, such as this one, that use overlapping windows because the data cannot be randomly divided into independent sets. This is because nearby windows share underlying data-points as described in Section 4.3.1. A common approach to solve this issue is to record a repetition number during data collection and then split the sets based on data from different repetitions. This has a high potential to cause bias since not all repetitions are equal.

Empirically it was found that in this and other data sets the first repetition of each movement, particularly, is of lower quality leading to lower reported performance than when testing with other repetitions. The exact degree of the effect varies between data sets and classification methods but consistently appears. It may be posited that this is likely due to a combination of factors including but not

limited to inexperience with performing the movement and inexperience with the experimental technique.

Later repetitions sometimes also show a performance drop relative to others, including in this data. For small numbers of repetitions, as here, the literature suggests this is unlikely to be caused by muscle fatigue [141] but may be caused by preemption of the experiment or inattention causing non-optimal replication of the movement.

One way to counter these effects is to run preliminary tests to find outlying repetitions and remove them from the data however this may harm the utility of a study depending on how it is done since these effects may be representative of the expected use case. Data is also often at a premium in EMG studies, making any wastage an expensive proposition.

This thesis proposes a method for Stratified K-fold cross-validation that accounts for the differences in repetition quality. Instead of removing repetitions or choosing K and randomly selecting repetitions for each fold, K should be selected to be an integer multiple of the number of repetitions in the data set then folds should be randomly generated under the condition that each repetition is equally represented in the test set. Secondly, it is proposed that when classifying the *rest* class in experiments making use of *rest*-movement-*rest* cycles, all the data either preceding or proceeding a movement be labelled with the repetition number of the movement.

Together these two methods ensure correct stratification at both the class and repetition level, which helps ensure that, when combined with an appropriate metric, results are robust, generalisable, and repeatable.

Forman *et al.* [184] show that how metrics are computed across validations folds can also bias results. Therefore their recommended methodology is adopted here, which is to calculate metrics as if all test folds formed a single test set. The macro-average accuracy is, therefore, modified:

$$acc^*_{ma} = \frac{1}{M}\sum_{i=1}^{M}\frac{\sum_{j=1}^{K}\mathrm{TP}_{i,j}}{\sum_{j=1}^{K}\mathrm{TP}_{i,j}+\sum_{j=1}^{K}\mathrm{FN}_{i,j}} \tag{4.6}$$

where K is the number of cross-validation folds, $TP_{i,j}$ is the correct predictions of movement class i for validation fold j, $FN_{i,j}$ is the number of incorrectly predicted examples of movement class i for validation fold j and M is the number of classes.

4.4 Benchmark Details

An updated set of benchmarks was designed that addressed shortcomings in previous studies as well as incorporating improvements in data augmentation during training to improve performance.

For the sake of comparability, the methods of Atzori *et al.* [37] are mirrored except for where improvements are made to fix the highlighted issues.

4.4.1 Preprocessing

Before being passed to the classifiers, the raw EMG stream was preprocessed following these steps:

1. Each channel was passed through a 5Hz, second-order, zero phase, low-pass Butterworth filter
    - This is justified because the Otto Bock electrodes used, perform Root Mean Squared (RMS) filtering which shifts the frequency spectrum of the resultant signal down to 0-5 Hz [37] from the generally held relevant range for EMG of 20-500 Hz [63, 66]
2. The signals were split into a continuous stream of windows:
    - Window length: 100/200/400 ms (10/20/40 samples)
    - Increment: 10 ms (1 sample)
3. Each window was labelled as belonging to the movement at the most recent sample within the window
4. Features were extracted from each channel of each window separately
    - The features are described in Table 4.1
    - Therefore each window is a data point of format $c \times n$ where c is the number of channels (10) and n is the size of the specific feature per channel
5. Data was split 50/50 for training/testing by repetition number
6. The training set was regularly downsampled by a factor of 10 to reduce the computational load which effectively acts as an increase in the window increment by the same factor

4.4.2 Features

Table 4.1 breaks down the features benchmarked. They were chosen as a subset of the original benchmark based on their performance in other studies [55, 100, 189]. The features are extracted over a sample window of length ω_l, where x_t is a single data sample from a single channel and each feature is extracted identically for each of the 10 sEMG channels.

The histogram feature was calculated using equally spaced bins based on signal value since the ranges/method was unspecified in the original work [37]. The maximum and minimum value were calculated across the entire data set for consistency and assumed to be limits of the hardware based on qualitative analysis of the signals.

The Mariginal Discrete Wavelet Transform (mDWT) is a reduction of the Discrete Wavelet Transform (DWT) that has been shown to work well for EMG problems [26, 190]. The mother wavelet chosen for the DWT was "sym4" based on the previous study [37].

Feature	Description
Mean Absolute Value (MAV)	$x_{mav} = \frac{1}{\omega_l}\sum_{t=1}^{\omega_l} \lvert x_t \rvert$
Variance (VAR)	$x_{var} = \frac{1}{\omega_l}\sum_{t=1}^{\omega_l} (x_t - \bar{x})^2$
Waveform Length (WL) [68]	$x_{wl} = \sum_{t=1}^{\omega_l - 1} \lvert x_t - x_{t+1} \rvert$
Histogram (HIST)	$\mathbf{x}_{hist} = \text{hist}(x_{1:\omega_l}, B)$ where $\mathbf{x}_{hist} \in \mathbb{R}^B$, B is the number of bins, and B=10 bins were used here, spaced equally between 0.0024 and 4.8351.
Mariginal Discrete Wavelet Transform (mDWT) [26, 190]	$\mathbf{x}_{mdwt} = \sum_{\tau=0}^{\omega_l/2^s - 1} \lvert \sum_{t=1}^{\omega_l} x_t \psi_{l,\tau}(t) \rvert$ where $\mathbf{x}_{dwt} \in \mathbb{R}^S$, $\psi_{l,\tau}(t) = 2^{-\frac{s}{2}} \psi(2^{-l}t - \tau)$, for $s = 1...S$, where S is the maximum level of decomposition (3 levels were used) ψ is the mother wavelet (sym4) l is a translation τ is a dilation

Table 4.1: Table of features where ω_l is the window length and x_t is a sample of EMG data at a time t for a single channel. All features are calculated on each channel individually.

Testing combinations of features was considered as it often improves performance [191]; however, was omitted here to reduce computational load and study complexity. In practical applications, it is likely to be desirable to combine several of the better performing features to further improve performance if there is

sufficient processing power and allowable latency to handle the additional computation.

4.4.3 Classifiers

The following five classifiers were used based on the original benchmark [37]. The MLP classifier is omitted since it is not fully specified and later chapters utilise neural network based approaches to the problem that are superior to the MLP.

- K-Nearest Neighbours (KNN)
  - K chosen to be 10 [this gave a comparable accuracy to the original [37] from a candidate pool of $K = 1 - 50$]
  - Euclidean distance [chosen for its comparable accuracy to the previous work]
- Linear Discriminant Analysis (LDA)
- Support Vector Machine - Linear Kernel (SVM-L)
  - Multiclass method: One vs One
  - Sequential Minimal Optimisation
  - Heuristic (subsample) kernel scaling
- Support Vector Machine - Radial Basis Function Kernel (SVM-RBF)
  - Multiclass method: One vs All
  - Sequential Minimal Optimisation
  - Box Constraint of 1 [for comparable accuracy and reasonable training time]
  - Heuristic (subsample) kernel scaling
- Decision Tree (DT)
  - Maximum number of decision splits: 150
  - Gini's Diversity Index used as split criterion
  - Prior probabilities based on class frequencies
  - No surrogates
- Each classifier was trained on each feature for each subject

4.4.4 Validation and Metrics

The validation procedure outlined in Section 4.3.3 was used, which improved upon the original single split method and used the macro-average accuracy to account for the class imbalances.

The metric used was the mean macro-average accuracy (Equation 4.6) across subjects presented as a decimal on the range $[0,\ 1]$ with 1 representing a perfect score.

Table 4.2 shows the randomly selected stratified set of folds for the training and test sets. Other sets of folds are possible which meet the balancing requirements; however, as is shown in Section 4.5.7 folds generated in this stratified way are representative of the true mean across all possible folds.

Split	Training	Testing
1	[1, 2, 3, 7, 10]	[4, 5, 6, 8, 9]
2	[1, 2, 4, 6, 9]	[3, 5, 7, 8, 10]
3	[1, 2, 7, 9, 10]	[3, 4, 5, 6, 8]
4	[1, 3, 4, 5, 7]	[2, 6, 8, 9, 10]
5	[1, 3, 5, 7, 10]	[2, 4, 6, 8, 9]
6	[2, 4, 5, 6, 8]	[1, 3, 7, 9, 10]
7	[2, 4, 6, 8, 10]	[1, 3, 5, 7, 9]
8	[3, 5, 6, 8, 9]	[1, 2, 4, 7, 10]
9	[3, 5, 8, 9, 10]	[1, 2, 4, 6, 7]
10	[4, 6, 7, 8, 9]	[1, 2, 3, 5, 10]

Table 4.2: Validation folds of training and test sets for the benchmarks. Numbers are the repetition label used in that fold; each repetition appears a total of 5 times in the folds of each set, ensuring proper stratification.

The set is built by first setting the number of repetitions that will be in the test set (N_{test}) in each fold then calculating the number of necessary folds to allow for the stratification:

$$N_{folds} = N_{reps} * N_{mult} \tag{4.7}$$

where N_{folds} is the total number of folds needed, N_{reps} is the number of different repetition labels in the data and N_{mult} is a positive integer multiplier that can be used when more folds are desired.

A list of all possible folds can then be generated and an initial fold chosen at random via computational pseudo-random number generation. A new fold is then added randomly, and the set is checked to ensure the maximum number of

examples of a particular repetition in the test portion of the fold is $\leq N_{test}$. If there are $> N_{test}$ examples of a specific repetition in the test portion, then the last added fold is removed, and another random fold is selected. Folds are added until there are N_{folds} in the set. If 10 folds are rejected for inclusion in a row, then the process is reset and begins from randomly selecting an initial fold.

Other ways of determining an appropriate stratification are possible; however, the above process is sufficiently fast and random on modern hardware.

4.4.5 Meta-Validation

Meta-validation was used to ensure that the validation methodology used was representative. This was achieved by performing an exhaustive validation across all 252 possible folds on all subjects with the SVM-RBF using the mDWT feature and window length set to 200ms. The SVM-RBF was chosen because it performed best out of the pool of benchmark classifiers.

There were 252 possible validation folds because there were 10 repetitions, and the data split was 50/50 training/testing with the order of repetitions in a fold unimportant. This meant that the problem took the form n choose k where n was 10 and k was 5:

$$\frac{n!}{k!(n-k)!} = \frac{10!}{5!5!} = 252 \tag{4.8}$$

The number of unique sets of 10 folds chosen from the 252 possible can also be calculated using the same equation with n set to 252 and k set to 10:

$$\frac{n!}{k!(n-k)!} = \frac{252!}{10!242!} = 2.38 \times 10^{17} \tag{4.9}$$

which indicates that complete exploration of the space is non-trivial.

The main goal was to demonstrate that the mean of the 10 folds selected via the previously proposed stratification method (see Table 4.2) was representative of the mean of all possible folds. These will be referred to as the stratified folds. The secondary aims were to show the potential for bias from randomly selected folds and that the fold selection technique worked in general rather than only on the one specific instance.

These goals were achieved by finding the true mean across all folds and the mean estimated by the stratified folds for each subject. A Monte-Carlo simulation was also run on each subject to estimate the range of values for the mean produced by randomly selected folds. A Monte-Carlo simulation was used because, as per Equation 4.9, trialling each combination would be computationally infeasible. The

Monte-Carlo simulation tested 10,000,000 random combinations of 10 folds on each subject.

The MAE was calculated between the true mean for each subject and the estimated mean given by the stratified folds. In order to quantify the potential error from randomly selected folds, the mean (across subjects) standard deviation of the Monte-Carlo results was calculated, and the mean maximum potential deviation was calculated as:

$$\delta = \frac{1}{N}\sum_{i=1}^{N} max(|\mu_i - \hat{\mu}_{min}|, |\mu_i - \hat{\mu}_{max}|) \tag{4.10}$$

where δ is the maximum potential deviation, N is the number of subjects (27), μ_i is the true mean for subject i and $\hat{\mu}_{min}$ and $\hat{\mu}_{max}$ are the minimum and maximum estimated subject means respectively, as found by the Monte-Carlo simulation. This allowed quantification of the potential MAE that may result from random selection.

4.4.6 Data Resampling and Augmentation Techniques

The imbalance in the dataset presented an avenue for improvements of classifier performance over the original. The SVM-RBF with mDWT was used to test out several data resampling strategies based on the literature from other domains [192–196].

Three different categories of resampling technique were trialled:

Downsampling was tested in several variations. Randomly downsampling all classes so that each had the same number of examples as the least represented class was tested as well as downsampling only the *rest* class until it had $0.5/1/2/3$ times the number of examples of the next most represented class.

Naive upsampling randomly copies data in the training set. Upsampling all non-*rest* classes to twice the highest number of non-*rest* examples was tested.

Synthetic Minority Oversampling Technique (SMOTE) [197] is a more principled approach to upsampling that generates new data points by taking points in the original feature data, randomly selecting one of the K nearest neighbours and "pulling" the original point towards that neighbour by a random factor on the range $0 - 1$. This creates new synthetic data interpolated on the original data. The authors of SMOTE and results from high dimensional data testing [195] also recommend downsampling of the majority class in combination with the SMOTE algorithm. Therefore for this test SMOTE (with $K = 25$) was used to upsample non-*rest* classes by a factor of 2 while simultaneously randomly downsampling *rest* to have a number of examples equal to twice the highest number of examples

of a non-*rest* class prior to upsampling.

Algorithm 4.1 shows the pseudo-code for the algorithm as used here, which operates on each class individually.

Algorithm 4.1 Pseudo code for the SMOTE algorithm. Here N_a was set to 1 to double the number of examples of each class, $K = 25$ and each class was computed separately.

Input: X : Matrix of examples of a class, N_a : Amount of SMOTE (integer), K : Number of nearest neighbours
Result: $N_s \times N_a$ synthetic samples

N_s = Number of examples in X
N_{dim} = Number of dimensions of each example in X
for $i = 1$ `to` N_s **do**
    `// Compute` K `nearest neighbours for sample` X_i
    for $j = 1$ `to` N_a **do**
        `// Randomly select one of the nearest neighbours:` k
        `// Initialise new synthetic` X_{syn}
        for $d = 1$ `to` N_{dim} **do**
            $dif = X_i[d] - k[d]$
            $gap = random([0, 1])$
            $X_{syn}[d] = X_i[d] + gap * dif$
        end
    end
end

4.4.7 Primary Benchmark Variants

The final benchmark was divided into three variants. Each followed the preprocessing and validation procedures outlined above across all feature, classifier and window length combinations but with the following differences in data augmentation:

- The "Baseline" featured no additional data augmentation
- The "*Rest* Downsampled" variant randomly downsampled *rest* data to have as many examples as the next most represented class
- The "2x SMOTE" variant used SMOTE to generate twice as many examples of each non-*rest* class and then randomly downsampled *rest* data to have as many examples as the next most represented class

These variations were chosen based on empirical testing of the aforementioned

pool of data augmentation techniques and computational constraints, which limited the amount of additional data it was feasible to evaluate.

4.4.8 Person-specific Movement Set Selection

The number of movements/gestures used in EMG studies varies greatly from 3 [44] to 53 as benchmarked here. The majority of studies assess less than 10, e.g. [19, 20, 22, 27–29, 33, 35] and in most cases, the number of movements is determined ahead of experiment time without any specific reasoning as to why exactly that number of movements is investigated. The general trend in these studies is, expectedly, that more movements leads to lower overall performance; however, the trade-off has not been explored. Selecting the "best" movements for an application is therefore also handled ahead of time in these cases which generally means selection of movements from the literature which potentially leads to lost performance since the selected movements may not be ideal for the particular subject or electrode setup.

Intuitively, movement performance is unlikely to be equal, and performance on any particular movement is intrinsically tied to both the subject performing the movement and the EMG recording setup. This intuition is backed up by the results demonstrated later in this and future chapters. Therefore it may be hypothesised that if there were some leeway in which movements could be used for a given application then, without significant exploratory work, it would be unlikely that for a given subject the best performing movements would be selected for the experiment.

Gathering EMG data from a subject includes an overhead to get the subject ready for testing as well as the time taken gathering data. Therefore if it is assumed that there is some leeway in terms of which movements can be used in the final application and the highest performance set of movements from a superset can be determined. Then, augmenting experimental procedures with additional movements becomes an efficient way of improving performance since the movements classified at test time may be tailored to an individual.

In most human-machine interface and some prosthetic control problems [17], the assumption of lee-way in terms of movements that must be classified holds however whether the highest performance set of movements from a superset can be found efficiently has not been explored. Therefore an experiment was designed using the NinaPro data to determine whether this was possible and to explore what the performance trend looks like when more movements are classified.

The search space for testing combinations of movements that could form a subset of a particular size is generally too big for brute force calculation. Therefore,

the experiment consisted of testing four different algorithms for sub-selection on the same process and classifiers described above. The MAV feature and a window length of 400ms were used to reduce the computational overhead of the experiment rather than running on the full combination of possibilities.

Even in this setup computation time was a major limiting factor of the study since evaluation at each different number of movements, from 2-53, amplified the computation needed by a factor of ~ 30, this was why the MAV feature was used over the mDWT since it is a more compact representation. The properly stratified cross-validation also had to be dropped to make the experiment feasible, therefore, making the absolute performance reported less robust while still allowing a proper review of the performance trends. Repetitions [1, 3, 5, 7, 9] were used for training and [2, 4, 6, 8, 10] for testing chosen to match with original benchmarks on the database [37].

The four approaches for movement sub-selection are outlined below.

Grouped Addition Baseline

The Grouped Addition approach formed the baseline for the experiment and involved adding to the set of movements classified in the order they are presented in the database [37]. Effectively this amounts to adding groups of related movements sequentially, starting with flexions and extensions of the fingers then moving to isometric and isotonic hand configurations moving on to wrist movements then various functional movements and grasps.

Ideally, the baseline would have been random selection but without the computational power to trial a reasonable number of random selections this alternative approach presented a way of finding a baseline in a principled manner. Adding the movements sequentially meant that this baseline likely represents performance towards the lower end of the performance spectrum given that similar movements are added after each other.

Maximisation of Minimum Distance Between Movement Means

First the mean MAV for each channel of each gesture may be calculated as

$$\bar{x}_m^{(c)} = \frac{1}{N_m} \sum_{i=1}^{N_m} x_m^{(i,c)} \tag{4.11}$$

where N_m is the number of examples of movement m and $x_m^{i,c}$ is the MAV of example i of movement m, channel c. This allows the Euclidean distance to be

calculated between two movements as

$$d_{m1,\ m2} = \sqrt{\sum_{i=1}^{C} (\bar{x}_{m1}^{(c)} - \bar{x}_{m2}^{(c)})^2} \tag{4.12}$$

where C is the number of channels. Then the distances between each movement in the current set $\mathbb{S}$ and the set of remaining movements may be calculated. The minimum distance between each movement not in $\mathbb{S}$ and the movements in $\mathbb{S}$ can then be found, and the movement with the maximum, minimum distance from all $\mathbb{S}$ is selected to be added to $\mathbb{S}$:

$$d_{min}^{(m)} = min(d_{s,m}), s \in \mathbb{S} \tag{4.13}$$

where $\mathbb{M}$ represents the set of all movements in the experiment and calculated for all $m \in \mathbb{M}, m \not\subseteq \mathbb{S}$.

$$max(d_{min}^{(m)}), m \in \mathbb{M},\ for\ m \not\supseteq S \tag{4.14}$$

Since the set $\mathbb{S}$ begins with the *rest* class contained within it, this approach iteratively builds up a set of movements by selecting the most dissimilar movement at each update step based on the Euclidean distance.

Maximisation of Minimum Symmetric Kullback-Leibler Divergence Between Movements

This approach has the same aim as the previous one, aiming to iteratively build up the set of movements but based on the Kullback-Leibler (KL) divergence [198] rather than the Euclidean distance.

For this metric it was assumed that the feature space could be represented by a multivariate normal distribution and therefore that each movement could be defined by a set of c means μ and a $c \times c$ covariance matrix Σ which allows calculation of the KL divergence:

$$d_{kl}(\mathcal{N}_\infty||\mathcal{N}_\in) = \frac{1}{2}\left(tr(\Sigma_2^{-1}\Sigma_1) + (\mu_2 - \mu_1)\right) \tag{4.15}$$

where N_1 and N_2 represent the movement distributions and $tr()$ is the trace operation. Due to the KL divergence being non-symmetric, the final comparison is performed on the value:

$$d_{kl}^{(1,2)} = d_{kl}(\mathcal{N}_\infty||\mathcal{N}_\in) + d_{kl}(\mathcal{N}_\in||\mathcal{N}_\infty) \tag{4.16}$$

which is used to calculate the most divergent movement from the current set:

$$d_{min}^{(m)} = min(d_{kl}^{(s,m)}), s \in \mathbb{S},\ for\ m \in \mathbb{M}, m \not\subseteq \mathbb{S} \tag{4.17}$$

$$max(d_{min}^{(m)}), m \in \mathbb{M},\ for\ m \not\supseteq S \tag{4.18}$$

Superset Performance Selection

Superset performance selection works on a subject-specific and classifier-specific basis. First, the classifier of interest is trained on the subject of interest with all possible movements. The performance of each class may then be evaluated, and movements iteratively added to the set to be used by selecting those movements which perform best on the complete classification problem.

$$max(acc_m), for\ m \not\supseteq S \tag{4.19}$$

4.5 Results and Discussion

4.5.1 Benchmark Results

Tables 4.3, 4.4 and 4.5 show the results, split up by window length, across all classifiers, features and variants.

The overarching conclusions that may be drawn from the results are that longer window lengths tend to improve performance, choice of classifier and feature combination is critical, and data resampling methods can make a significant difference to the final performance.

4.5.2 Metric Comparison

The "Baseline" results highlight the significant effect of using macro-average accuracy when compared to the original study [37] with the general performance level being greatly reduced, and the differences between features becoming more prominent. Figure 4.7 shows how the macro-average accuracy produces a much more representative value for performance under the assumption that the test time class distribution is unknown. This assumption is used throughout this work since the goal is to solve the most general form of the classification problem. In specific circumstances it may be desirable to weight the accuracy metric, but this is inappropriate without a strong prior on the expected class distribution or a use-case requirement of particular classes being more important than others.

Window Length: 100ms

		Benchmark					
		Baseline		*Rest* Downsampled		2x SMOTE	
Classifier	**Feature**	μ_{subj}	(μ_{rank})	μ_{subj}	(μ_{rank})	μ_{subj}	(μ_{rank})
KNN	MAV	0.428	(4.4)	0.508	(3.9)	0.539	(4.0)
	VAR	0.098	(22.0)	0.141	(22.7)	0.151	(22.7)
	WL	0.160	(16.0)	0.220	(17.6)	0.231	(17.5)
	HIST	0.383	(6.9)	0.412	(9.2)	0.437	(8.9)
	mDWT	0.431	(3.4)	0.509	(3.5)	0.540	(3.3)
LDA	MAV	0.274	(12.0)	0.347	(11.7)	0.367	(11.6)
	VAR	0.049	(24.2)	0.073	(25.0)	0.116	(24.9)
	WL	0.106	(20.8)	0.166	(20.7)	0.195	(20.0)
	HIST	0.338	(9.6)	0.368	(10.7)	0.374	(10.7)
	mDWT	0.291	(10.8)	0.364	(10.3)	0.368	(10.6)
SVM-RBF	MAV	0.454	(2.2)	0.534	(2.0)	0.555	(2.0)
	VAR	0.042	(24.8)	0.128	(24.0)	0.143	(24.0)
	WL	0.133	(18.3)	0.225	(16.9)	0.237	(16.6)
	HIST	0.419	(4.3)	0.451	(6.9)	0.472	(6.9)
	mDWT	**0.467**	**(1.0)**	**0.549**	**(1.0)**	**0.575**	**(1.0)**
SVM-L	MAV	0.378	(7.7)	0.491	(5.8)	0.515	(5.2)
	VAR	0.099	(21.7)	0.192	(19.0)	0.211	(18.7)
	WL	0.146	(17.1)	0.253	(15.2)	0.266	(14.8)
	HIST	0.358	(8.7)	0.431	(8.0)	0.450	(8.1)
	mDWT	0.379	(7.1)	0.492	(5.0)	0.514	(5.6)
DT	MAV	0.211	(13.5)	0.283	(13.4)	0.285	(13.7)
	VAR	0.111	(20.7)	0.153	(21.9)	0.165	(22.3)
	WL	0.123	(18.9)	0.166	(20.5)	0.178	(20.9)
	HIST	0.183	(15.3)	0.237	(16.1)	0.236	(16.8)
	mDWT	0.210	(13.7)	0.281	(13.9)	0.283	(14.0)

Table 4.3: 100ms window benchmark results across all features, classifiers and variants. For each benchmark μ_{subj} is the inter-subject mean macro-average accuracy for a classifier-feature combination. The bracketed value μ_{rank} is the inter-subject mean rank (range 1-25) of the classifier-feature combination for a particular benchmark. A rank of 1 indicates top performance on each subject for a specific benchmark.

The cause of the over-reporting by the micro-average accuracy seen in Figure 4.7, 79% accuracy reported despite only 2 classes performing higher, is the class imbalance inherent in the data (see Section 4.2). In conjunction with the micro-average this leads to an effective weighting of the performance proportional to the ratios between classes. On this data, that means *rest* performance makes up the majority of the reported performance which is what causes the over-reporting. Specifically, the inter-subject mean contribution of the *rest* class to the final micro-average accuracy is 58.1%.

The macro-average accuracy (dotted line) reduces this weighting to 1.9%, making it equal to each other class, which leads to a metric which better represents the assumptions about the use-case and better fits the classic intuition on what ac-

Window Length: 200ms

		Benchmark					
		Baseline		*Rest* Downsampled		2x SMOTE	
Classifier	Feature	μ_{subj}	(μ_{rank})	μ_{subj}	(μ_{rank})	μ_{subj}	(μ_{rank})
KNN	MAV	0.444	(4.9)	0.528	(4.6)	0.552	(4.2)
	VAR	0.122	(21.8)	0.176	(22.4)	0.188	(22.6)
	WL	0.223	(14.2)	0.294	(16.0)	0.306	(15.6)
	HIST	0.425	(6.1)	0.458	(8.7)	0.475	(8.6)
	mDWT	0.458	(3.5)	0.540	(3.2)	0.567	(2.7)
LDA	MAV	0.285	(12.1)	0.360	(12.0)	0.376	(12.0)
	VAR	0.055	(24.6)	0.083	(25.0)	0.131	(25.0)
	WL	0.139	(20.1)	0.212	(20.3)	0.241	(19.4)
	HIST	0.362	(9.5)	0.396	(10.4)	0.398	(10.5)
	mDWT	0.301	(11.0)	0.376	(10.9)	0.384	(10.8)
SVM-RBF	MAV	0.472	(2.6)	0.558	(2.0)	0.569	(2.4)
	VAR	0.057	(24.4)	0.164	(23.8)	0.181	(23.3)
	WL	0.217	(15.1)	0.311	(14.2)	0.318	(14.2)
	HIST	0.460	(3.5)	0.500	(6.3)	0.511	(6.6)
	mDWT	**0.502**	**(1.0)**	**0.588**	**(1.0)**	**0.606**	**(1.0)**
SVM-L	MAV	0.391	(8.3)	0.514	(6.0)	0.529	(6.1)
	VAR	0.116	(22.2)	0.231	(19.1)	0.252	(18.3)
	WL	0.187	(17.1)	0.317	(13.6)	0.327	(13.3)
	HIST	0.386	(8.5)	0.466	(8.3)	0.478	(8.4)
	mDWT	0.398	(7.2)	0.521	(4.8)	0.536	(4.9)
DT	MAV	0.214	(14.9)	0.292	(15.9)	0.288	(16.4)
	VAR	0.129	(21.4)	0.178	(22.6)	0.187	(22.9)
	WL	0.148	(19.4)	0.201	(20.7)	0.209	(21.0)
	HIST	0.190	(17.1)	0.251	(18.1)	0.245	(19.0)
	mDWT	0.216	(14.5)	0.294	(15.1)	0.290	(15.8)

Table 4.4: 200ms window benchmark results across all features, classifiers and variants. For each benchmark μ_{subj} is the inter-subject mean macro-average accuracy for a classifier-feature combination. The bracketed value μ_{rank} is the inter-subject mean rank (range 1-25) of the classifier-feature combination for a particular benchmark. A rank of 1 indicates top performance on each subject for a specific benchmark.

curacy represents. Figure 4.7 shows an example on a single subject. However the same result is present across all subjects, window lengths, classifiers and features which justifies the metric choice.

4.5.3 Benchmark Performance Breakdown

The SMOTE algorithm augmented benchmark significantly outperformed the other two benchmarks at each window length. The subject performance of each classifier-feature combination on the different benchmarks can be treated as matched samples i.e. each row of Tables 4.3, 4.4 and 4.5 can be treated as 3 sets of 27 matched samples, one for each subject. This equates to comparing 3 algorithms (the benchmarks) across $25 \times 27 = 675$ matched samples. Therefore the Friedman test

Window Length: 400ms

		Baseline		Benchmark *Rest* Downsampled		2x SMOTE	
Classifier	**Feature**	μ_{subj}	(μ_{rank})	μ_{subj}	(μ_{rank})	μ_{subj}	(μ_{rank})
KNN	MAV	0.458	(5.6)	0.544	(5.3)	0.576	(4.8)
	VAR	0.170	(20.2)	0.239	(21.7)	0.258	(21.3)
	WL	0.299	(13.2)	0.374	(14.0)	0.392	(13.9)
	HIST	0.468	(5.1)	0.502	(8.2)	0.526	(8.0)
	mDWT	0.494	(3.1)	0.576	(2.4)	0.612	(2.0)
LDA	MAV	0.302	(13.1)	0.377	(13.7)	0.395	(13.4)
	VAR	0.071	(24.9)	0.104	(25.0)	0.160	(25.0)
	WL	0.188	(18.9)	0.272	(19.0)	0.301	(17.9)
	HIST	0.389	(9.3)	0.429	(10.4)	0.436	(10.5)
	mDWT	0.324	(11.5)	0.400	(11.9)	0.413	(11.9)
SVM-RBF	MAV	0.485	(3.7)	0.573	(2.8)	0.592	(3.2)
	VAR	0.099	(24.1)	0.234	(22.3)	0.254	(21.6)
	WL	0.314	(12.1)	0.400	(12.0)	0.410	(12.2)
	HIST	0.502	(2.6)	0.546	(5.0)	0.561	(5.7)
	mDWT	**0.537**	**(1.1)**	**0.625**	**(1.0)**	**0.651**	**(1.0)**
SVM-L	MAV	0.401	(9.1)	0.528	(6.8)	0.552	(6.7)
	VAR	0.148	(22.0)	0.300	(17.2)	0.324	(16.5)
	WL	0.236	(16.0)	0.388	(13.0)	0.403	(13.0)
	HIST	0.411	(8.2)	0.497	(8.5)	0.517	(8.7)
	mDWT	0.419	(7.4)	0.543	(5.0)	0.570	(4.9)
DT	MAV	0.217	(17.0)	0.299	(17.7)	0.301	(18.4)
	VAR	0.151	(22.1)	0.210	(23.7)	0.220	(23.9)
	WL	0.173	(20.3)	0.236	(21.9)	0.244	(22.2)
	HIST	0.199	(18.3)	0.266	(19.8)	0.268	(20.6)
	mDWT	0.221	(16.1)	0.303	(16.6)	0.305	(17.6)

Table 4.5: 400ms window benchmark results across all features, classifiers and variants. For each benchmark μ_{subj} is the inter-subject mean macro-average accuracy for a classifier-feature combination. The bracketed value μ_{rank} is the inter-subject mean rank (range 1-25) of the classifier-feature combination for a particular benchmark. A rank of 1 indicates top performance on each subject for a specific benchmark.

followed by post-hoc Holm procedure could be used to determine whether the SMOTE augmented benchmark was significantly better than the other two.

The Friedman test indicated the benchmarks have different performances at $p < 2.2 \times 10^{-308}$ on all window lengths; therefore, the post-hoc Holm procedure could be used. Note that 2.2×10^{-308} is the smallest value that can be represented in a standard double floating-point number [199]. On all window lengths the SMOTE augmented benchmark was found to be significantly better than the other two with the final Holm procedure p values:

- $p < 1.2 \times 10^{-57}$ for window length 100ms
- $p < 2.9 \times 10^{-46}$ for window length 200ms

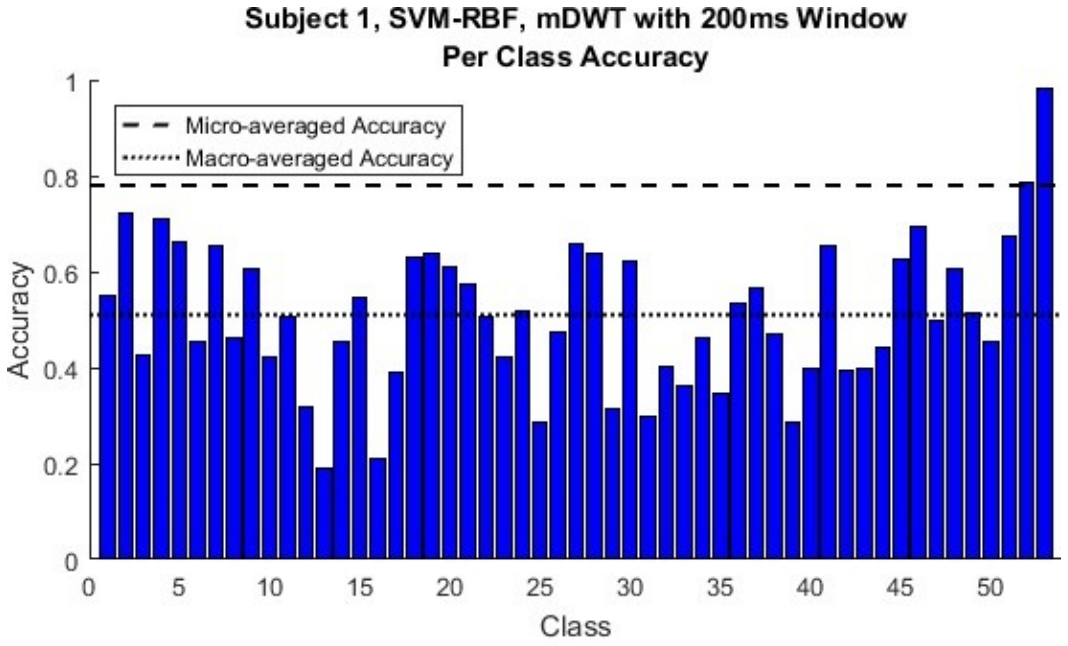

Figure 4.7: Accuracy per class demonstrating the over-reporting of performance caused by using micro-averaged accuracy as a performance metric when compared to the macro-average accuracy. The above example is taken from the Baseline benchmark using the SVM-RBF with mDWT and 200ms window on Subject 1. However, the highlighted differences are consistent across subjects classifiers and features.

- $p < 1.6 \times 10^{-60}$ for window length 400ms

Concretely the SMOTE augmented benchmark provided top performance on the following number of subject-classifier-feature combinations:

- 632 of 675 for window length 100ms
- 601 of 675 for window length 200ms
- 639 of 675 for window length 400ms

SMOTE increased computational load relative to the downsampling benchmark, however, produced a significant improvement compared to the other benchmarks. SMOTE's improved performance over just the *rest* downsampling approach made it the best algorithm for achieving high performance in this context justifying the increase in computation.

SMOTE's success implies that the classifiers would benefit from additional data for training purposes which lead to the hypothesis that reducing the original downsampling of the training data in the preprocessing step would also produce similar results when combined with *rest* downsampling. Empirical testing

revealed this to be the case. Reduction of preprocessing step downsampling to maintain a similar number of training examples as used in the SMOTE benchmark produced a similar performance to the SMOTE benchmark. This is potentially due to the interaction between the windowing process and SMOTE producing synthetic examples that are similar to nearby windows, particularly at longer window lengths when more information is shared.

Further complete removal of the preprocessing step downsampling was trialled, which produced a $1-2\%$ improvement over the SMOTE/reduced downsampling approaches but increased the computational time necessary by a factor of ~ 12 on the available hardware. This made it impractical to perform the fully cross-validated benchmark due to time constraints.

4.5.4 Feature Differences

The mDWT feature significantly outperformed the other features across all window lengths. Statistical comparison was computed by treating each benchmark-classifier-subject combination as a set of 5 matched samples with the independent variable being the feature. This lead to a total of $3 \times 5 \times 27 = 405$ matched samples (benchmarks $\times$ classifiers $\times$ subjects).

The Friedman test indicated at $p < 2.2 \times 10^{-308}$ that the features produced different performances, at each window length, which allowed for post-hoc Holm procedure testing. The following p values were obtained on the final step of the Holm procedure, confirming that the mDWT was the best feature of those tested:

- $p < 3.6 \times 10^{-7}$ for window length 100ms
- $p < 1.2 \times 10^{-17}$ for window length 200ms
- $p < 4.2 \times 10^{-22}$ for window length 400ms

Concretely the mDWT provided the highest performance for the following number of benchmark-classifier-subject combinations:

- 252 of 405 for window length 100ms
- 306 of 405 for window length 200ms
- 305 of 405 for window length 400ms

The majority of non-top performances came from combinations including the LDA classifier which performs better with the HIST feature.

The average ranks computed for Friedman and Holm procedure testing were:

- Window length 100ms
  - mDWT: 1.39
  - MAV: 1.96
  - HIST: 2.65
  - WL: 4.00
  - VAR: 5.00
- Window length 200ms
  - mDWT: 1.25
  - MAV: 2.20
  - HIST: 2.56
  - WL: 4.00
  - VAR: 5.00
- Window length 400ms
  - mDWT: 1.26
  - MAV: 2.33
  - HIST: 2.44
  - WL: 3.97
  - VAR: 5.00

The MAV performed very well, given its simplicity as a single value for each channel, with the second best average rank. This implies that for many movements knowing that muscles near the electrode are activating and approximately to what extent is sufficient for narrowing down the possible movement.

The HIST feature generally performed worse than the MAV with the notable exception of when it was combined with the LDA classifier. From the ranks in Tables 4.3, 4.4 and 4.5 it is clear that the HIST feature also gains the least from adding *rest* downsampling. This can be seen in its consistent rise in rank when going from the baseline to the *rest* downsampled benchmark which is not as prominent or consistent when going from the *rest* downsampled to the SMOTE benchmark. This implies it is useful for defining *rest* compared to the other features.

The MAV feature's overall performance decreases as window length increases. The HIST tends to outperform it more often. This is due to the greater fidelity afforded by the HIST, meaning it can retain more information.

The WL performed poorly. It almost never outperformed the mDWT, MAV or HIST for any benchmark-classifier-subject combination at any window length. This differs from the original study [37], which found it had performance rivalling the aforementioned other features. This is due to this study's usage of macro-average accuracy. When evaluated under micro-average accuracy, as in the original, the WL's performance improves dramatically implying it useful for differentiating movements and *rest* but less useful at distinguishing between movements.

The VAR is worst of all the features tested. It never outperformed any other feature for any benchmark-classifier-subject combination at any window length. This indicates it does not capture much meaningful information compared to the other features.

4.5.5 Comparison of Window Lengths

The 400ms window length significantly improves performance over the 100ms and 200ms window lengths. Statistical comparison was computed by treating each benchmark-feature-classifier-subject combination as a set of 3 matched samples with the independent variable being the window length. This lead to a total of $3 \times 5 \times 5 \times 27 = 2025$ matched samples (benchmarks $\times$ features $\times$ classifiers $\times$ subjects).

The Friedman test indicated at $p < 2.2 \times 10^{-308}$ that the window lengths produced different performances justifying the use of the post-hoc Holm procedure.

The Holm procedure found that, at $p < 1.2 \times 10^{-214}$, the 400ms window length improved performance. The average rank for the different window lengths was:

- 400ms: 1.02, top performance on 1999/2025 matched samples
- 200ms: 2.00, top performance on 20/2025 matched samples
- 100ms: 2.98, top performance on 6/2025 matched samples

This result confirms the intuition that a longer window length is better for classification since more useful information is contained as the window size increases. The result also implies that a longer window length than those tested here may further improve performance. Using a longer window may also increase the latency between a subject performing a movement and it being recognised, however, therefore this trade-off would have to be assessed based upon the context of the classifier's usage.

4.5.6 Classifier Trends

The SVM-RBF performed significantly better than all other classifiers across all window lengths. Statistical comparison was computed by treating each benchmark-feature-subject combination as a set of 5 matched samples with the independent variable being the classifier. This lead to a total of $3 \times 5 \times 27 = 405$ matched samples (benchmarks $\times$ features $\times$ subjects).

On the window length 100ms results the Friedman indicated at $p < 6.4 \times 10^{-260}$ that the features produced different performances. Longer window lengths lead to lower p values. Therefore the usage of the post-hoc Holm procedure was valid. The following p values were obtained on the final step of the Holm procedure, confirming that the SVM-RBF was the best classifier:

- $p < 1.5 \times 10^{-2}$ for window length 100ms
- $p < 1.6 \times 10^{-6}$ for window length 200ms
- $p < 1.2 \times 10^{-10}$ for window length 400ms

Concretely the SVM-RBF provided the highest performance for the following number of benchmark-feature-subject combinations:

- 242 of 405 for window length 100ms
- 263 of 405 for window length 200ms
- 306 of 405 for window length 400ms

The average ranks computed for Friedman and Holm procedure testing were:

- Window length 100ms
  - SVM-RBF: 1.96
  - SVM-L: 2.22
  - KNN: 2.33
  - LDA: 4.21
  - DT: 4.28
- Window length 200ms
  - SVM-RBF: 1.73
  - KNN: 2.27

  - SVM-L: 2.36
  - LDA: 4.22
  - DT: 4.41

- Window length 400ms
  - SVM-RBF: 1.47
  - KNN: 2.19
  - SVM-L: 2.50
  - LDA: 4.20
  - DT: 4.63

These results for the SVM-RBF agree with the original [37] finding it to perform best overall and improve with increasing window length compared to the other tested classifiers. Contrary to the original, however, it was found that the SVM-L performed similarly to the KNN particularly as the window length was decreased.

The SVM-RBF with mDWT performs best of the 25 classifier-feature pairs tested on all benchmarks and at all window lengths. It achieved the top performance on almost every subject as evidenced by all its ranks being ≤ 1.1 in Tables 4.3, 4.4 and 4.5. However, it is not possible, at the 5% significance level, to determine that the SVM-RBF with mDWT is significantly better than the KNN with mDWT or SVM-RBF with MAV at any particular window length. This was due to the need to make 24 pairwise comparisons between the classifiers with only $3 * 27 = 81$ (benchmarks $\times$ subjects) matched samples per window length.

Combining window lengths into the statistical analysis allowed it to be conclusively shown that SVM-RBF with mDWT is the best performing classifier-feature pair. The comparison takes the form of comparing the 25 classifiers over matched samples from $3 * 27 * 3 = 243$ benchmark-subject-length combinations. The Friedman test confirmed the difference in performance at $p < 2.2 \times 10^{-308}$ and the Holm procedure showed significantly better performance at $p < 2.2 \times 10^{-2}$. The SVM-RBF with mDWT achieved the highest performance on 239/243 combinations.

As evidenced in Figure 4.7 there is a large amount of variation between classes performance for a given classifier-feature combination. This is present across all classifiers and features and indicates that certain classes are more easily differentiable or closely related to others. There is also high variation between subjects, e.g. for the SVM-RBF classifier using the mDWT the inter-subject standard deviation was 0.059. Correspondingly, different movements are often more easily

classifiable for different subjects. This leads to the idea that it can be beneficial for applications to sub-select movements for particular subjects and in so doing tailor classification to the subject, something which is already done to some extent by training on each subject individually. This premise is explored further in Chapter 7 as a way to reduce inter-subject variation as well as to improve overall performance in situations where it is unnecessary to use all 53 movements.

In general, the only class to drop in performance in the non-Baseline benchmarks relative to the original study [37] was the *rest* class. This is a predictable outcome of correcting the class imbalance in the data. However, it is still important to note as often it may be desirable to improve performance on the *rest* class, particularly in terms of false negatives. How this can and should be handled is application specific, particularly with regards to expected class distributions, however, so is not dealt with in these benchmarks.

The majority of the conclusions drawn from this study are the same as the original [37]. However, the absolute performance numbers are lower due to the re-calibrations in terms of metric choice and handling of the *rest* class. These changes to the underlying assumptions of the study, compared to the original, are necessary to make the results useful for practical application, therefore, the results presented here represent a new baseline for performance in the hand movement classification field.

The best overall combination, in terms of raw performance, was the SVM-RBF with 400ms window length utilising the mDWT feature and the SMOTE based data augmentation which achieved a macro-average accuracy of 65.1%. This sets a new conservative benchmark for performance on the classification problem. Results, both from this study and the wider literature [191], show that making more data available to the classifier and utilisation of feature combinations could push this performance even higher. Similarly, this benchmark tackled simultaneous classification of many movements which may not be necessary for a given application so despite this performance being relatively low in absolute terms these results are encouraging for practical application of sEMG based movement classification.

4.5.7 Meta-Validation Results

In order to ensure the validation approach was representative, the meta-validation strategy described in Section 4.4.5 was implemented.

The stratified folds were found to have a MAE of 0.0022 (relative to the true mean calculated over all possible folds) while the Monte-Carlo simulation of the random folds was found to have a mean maximum potential deviation of 0.0425

and a mean standard deviation of 0.0071. The stratified folds tended slightly towards an overestimate of the true mean with 19 of the 27 subjects being overestimated. When tested on 10 other sets of different, random stratified folds the MAE varied by ± 0.0001.

These numbers show that the stratified approach to selecting cross-validation folds is superior to random fold selection because it gives a closer estimate of the true mean across all folds than the majority of other selections. Specifically, analysis of the z-score indicates that the stratified approach produces, on average, an error better than 76% of random selections. This validates the choice of validation methods used here, although it is worth to note the small positive bias in results it produces.

4.5.8 Movement Sub-selection Results

The results of the comparison between movement subset selection algorithms is shown in Figure 4.8.

The best performing was the Superset Performance Selection algorithm, which consistently outperformed the other algorithms across all classifiers and all movement set sizes. It was possible to confirm that the algorithm was significantly better by assessing each combination of classifier, movement set size, and subject as a set of 4 matched samples, one for each algorithm being tested. Both the Friedman test and the Holm procedure returned $p < 2.2 \times 10^{-308}$.

The average ranks and number of combinations performed best on by each algorithm is summarised below:

- Superset Performance: 1.15, top performance on 6144/6885 matched samples
- KL Divergence: 2.50, top performance on 310/6885 matched samples
- Mean Distance: 2.80, top performance on 317/6885 matched samples
- Baseline: 3.55, top performance on 114/6885 matched samples

The reason the Superset Performance Selection algorithm outperforms the others is that it incorporates extra information. It tailors the movement set to the best for each subject-classifier combination. This contrasts with the other algorithms that only have access to information on the subject.

Table 4.6 shows how the average rank of the selection algorithms at small movement set sizes.

The Mean Distance algorithm competes with Superset Performance algorithm at very small movement set sizes but quickly falls off. This potentially indicates

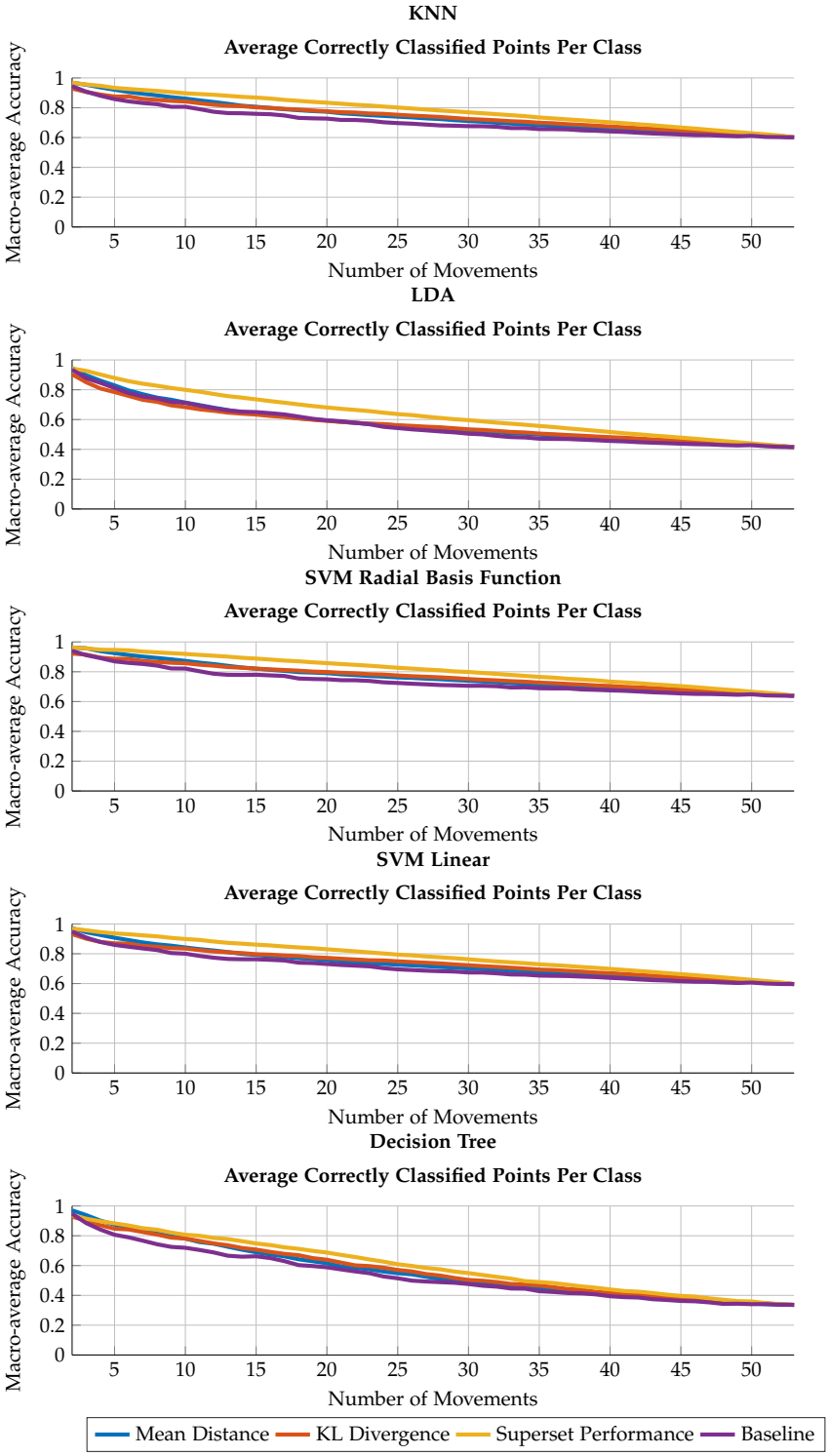

Figure 4.8: Performance of classifiers using different size subsets of the 53 movements in the database. Different algorithms are used to select the subset with the Superset Performance Selection algorithm consistently providing higher performance on all classifiers at different set sizes.

	Average Rank			
Set Size	**Superset Performance**	**Mean Distance**	**KL Divergence**	**Baseline**
2	2.25	2.30	2.46	2.99
3	1.83	2.02	3.00	3.15
4	1.55	2.09	3.16	3.20
5	1.39	2.23	3.05	3.33

Table 4.6: Average rank of the movement set selection algorithms for small set sizes. Rank was computed based on the 135 classifier-subject pairs that exist for a given movement set size.

that, at small set sizes, the most important factor is how dissimilar the movements in the set are. At larger sizes, the extra information about the classification algorithm added by the Superset Performance algorithm becomes more important. This leads to its decreasing average rank as the set size is increased.

The KL Divergence performs worse than the Mean Distance algorithm at smaller set sizes but improves over it at larger ones. This is evidenced by the Mean Distance algorithm's performance in Table 4.6 contrasted with its overall average rank of 2.50. Given it was designed to perform a function similar to the Mean Distance algorithm, this means that the mean MAV may not present a suitably normal distribution on each channel for comparison.

Therefore it would be reasonable to assume that these two algorithms are likely to vary more in utility with different features than the Superset Performance Selection algorithm. This highlights another important benefit of Superset Performance Selection, which is that it is feature-agnostic.

Finally, it can be seen in Figure 4.8 that all these algorithms make the most impact at lower set sizes since there is most dissimilarity in sets at those sizes. These algorithms also beat the baseline Grouped Addition algorithm demonstrating their practical utility. Overall, therefore, the Superset Performance Selection algorithm is the most versatile and useful algorithm since it provides clear performance improvements of up to 10% over the baseline and is mostly invariant to changes in the features used.

This work demonstrates that there is utility in over-provisioning experiments with more movements than is necessary for a given application in order to improve performance in cases where specific movements are not required, such as in human-computer interaction. This performance improvement can come at minimal overhead at experiment time and removes the need to try to design a set of movements that is the best for a population, instead, allowing tailoring of systems to individuals.

4.6 Conclusion

The methodological improvements to evaluation presented in this chapter alleviate several common issues found in sEMG-based classification studies. Therefore these represent a more robust set of techniques for evaluating performance relative to many previous studies which are critical to ensuring representative repeatable results.

In particular, the need for a precise definition of the performance metric is discussed and the major effect of inappropriate choice demonstrated. The macro-average accuracy is presented as a more informative default performance measure since it applies a uniform a priori probability to the relative importance of classifying each class rather than the frequency based probability standard accuracy imposes, which is unlikely to be valid in real-world application.

A new stratification technique for cross-validation is also presented, which removes the potential for bias that random selection causes when the number of folds, K, is low. The technique is meta-validated (Section 4.4.5) to show it is representative of the performance that would be found across all folds and provides a tangible improvement over random selection.

In light of these issues and improvements, a major study in sEMG classification [37] is re-evaluated with these more realistic techniques, and the overall performance is shown to be lower than originally reported. This is of importance to the field since the original study is well cited and its data and methodology used in several other research ventures potentially leading to misleading conclusions.

The original study is also extended with several data resampling techniques which improve performance by over 10% in terms of absolute macro-average accuracy. These represent a new improved and more robust benchmark for the potential performance that can be achieved on this data set as well as presenting a general-purpose methodology for sEMG hand movement classification.

Parallel to the benchmark, a novel evaluation of movement sub-selection techniques is presented. The evaluation found that when fewer movements are needed than are available, it is possible to tailor movement selection to a particular subject to improve performance relative to naive selection. The best technique was Superset Performance Selection, which consistently improved performance over the other tested techniques. This evaluation demonstrated that adaptation to each subject is a useful avenue for improving the performance and utility of hand gesture classification techniques.

Chapter 5

Deep Neural Networks for Person-Specific Classification

5.1 Introduction

Deep neural networks have been shown to be a powerful tool for a variety of machine learning problems [102, 104, 181]. In the last chapter, feature-based classification methods were explored and improved upon to present more robust and representative performance results. In this chapter, the methodological improvements are built upon by applying deep neural networks in the place of the feature based classifiers and feature extraction steps. This leads to significant performance improvement over the original baseline setting a new state-of-the-art for sEMG gesture classification.

Using deep neural networks also has a distinct advantage over feature based techniques in that it allows person-specific adaptation at the feature level. The literature for sEMG gesture classification relies heavily on high-performance feature extraction [99, 100, 200] with the best features varying depending on the particular setup and subjects involved. Deep neural networks, on the other hand, allow tailoring of this step to each individual, automating a potentially complex and time-consuming part of the classification process.

The major novelty of this chapter is the design and evaluation of a novel convolutional neural network architecture that encodes domain knowledge to produce best-in-class performance. Specifically, the architecture limits early convolutions to ensure that features are extracted from individual channels first and that only high-level features may be inter-channel. This design is based on observation of the sEMG feature literature, which almost invariably makes the same assumption during feature extraction. The network is evaluated against several contemporary

deep neural network approaches [39, 47, 201], across two different data sets to demonstrate its utility.

An analysis of the resultant performance distribution and visualisation of the features the network extracts is also presented. It is shown that there is a consistent negative performance bias that results from the use of the first repetition as well as other non-uniform trends that can affect the overall results. This presents further evidence that the stratified cross-validation technique proposed in the last chapter is vital for repeatability. Further, the substantial change in performance caused by inappropriate accuracy measures is demonstrated again on the new data set to show it is not an issue limited to the original data. In addition, it is shown that the neural network presented here improves the curve associated with performance against normalised position within a movement. This is shown to have implications for experimental design to be explored in future chapters. Finally, visualisation and review of the features the network extracts demonstrate a number of other interesting issues that have an impact on the design of the network itself and supporting software.

5.2 Methodology

The same evaluation methodology as described in Chapter 4 is used: cross-validating across a balanced set of folds delineated by repetition number, avoiding data sharing between folds, using macro-average accuracy as the performance metric and Forman *et al.*'s inter-fold evaluation [184]:

$$acc^*_{ma} = \frac{1}{M}\sum_{i=1}^{M}\frac{\sum_{j=1}^{K}\mathrm{TP}_{i,j}}{\sum_{j=1}^{K}\mathrm{TP}_{i,j}+\sum_{j=1}^{K}\mathrm{FN}_{i,j}} \tag{5.1}$$

NinaPro databases 1 and 2 were used to trial the methodology. Database 1 is explored and analysed in Section 4.2. Database 2 [202] is collected using a similar experimental technique to Database 1 but uses a different set of electrodes and covers 40 subjects performing 40 hand movements plus *rest* along with 9 finger force exercises. The force exercises are omitted here since they are intended for force regression problems rather than classification. The electrodes used are a Delsys Trigno system with 12 channels, a 2kHz sampling frequency and no signal rectification [202]. Compared to database 1, therefore, the signals in database 2 are of higher fidelity. Database 2 suffers from a similar level of class imbalance as database 1 due to the experimental methodology being similar.

On both databases, a window length of 150ms and an increment of 10ms was used for comparability with contemporary work [39, 47, 201] and recommenda-

tions made by Farrell and Weir [203]. For any given application, this length could be varied to match the desired latency-performance trade-off (Section 4.3.1); However, it was more important to maintain comparability with other works for developing this benchmark. 150ms corresponds to 15 sample windows with a 1 sample increment on database 1 and a 300 sample window with 20 sample increments on database 2. Windows are still labelled by the most recent label within their data view, and repetitions are labelled in the same way as in Chapter 4.

In this set of experiments, the training set was not downsampled because of the significant reduction ($\sim$ 30x) in the number of classifiers that required training and the ability to effectively utilise Graphics Processing Unit (GPU) computation for training the neural networks. The zero-phase low-pass filtering was also omitted since true zero-phase filtering is not possible in an online context, which is the ultimate goal for the classification process. No feature extraction was performed.

Stratified K-fold cross-validation was used with the stratification based on repetition numbers as discussed in Section 4.3.3. The specific splits/folds used are shown in Table 5.1, the folds were generated randomly with the first split being seeded on both databases since this is the single split examined in the papers benchmarked against [39, 47, 201]. Similarly, the data was only split into training and testing without a separate validation set to maintain this comparability.

	Database 1		**Database 2**	
Split	**Training**	**Testing**	**Training**	**Testing**
1	[1, 3, 4, 6, 8, 9, 10]	[2, 5, 7]	[1, 3, 4, 6]	[2, 5]
2	[1, 2, 3, 5, 7, 9, 10]	[4, 6, 8]	[1, 4, 5, 6]	[2, 3]
3	[1, 2, 4, 6, 7, 8, 10]	[3, 5, 9]	[1, 2, 3, 5]	[4, 6]
4	[1, 2, 5, 6, 8, 9, 10]	[3, 4, 7]	[1, 2, 4, 6]	[3, 5]
5	[2, 3, 4, 6, 8, 9, 10]	[1, 5, 7]	[2, 3, 4, 5]	[1, 6]
6	[1, 2, 3, 4, 5, 7, 9]	[6, 8, 10]	[2, 3, 5, 6]	[1, 4]
7	[3, 5, 6, 7, 8, 9, 10]	[1, 2, 4]		
8	[1, 2, 4, 5, 7, 8, 9]	[3, 6, 10]		
9	[3, 4, 5, 6, 7, 8, 10]	[1, 2, 9]		
10	[1, 2, 3, 4, 5, 6, 7]	[8, 9, 10]		

Table 5.1: Training and test set splits for validation. Each number refers to a movement repetition (for 10 total repetitions in database 1 and 6 repetitions in database 2).

All data was independently normalised to zero-mean and standard deviation one, using training set data only for each validation fold.

The preprocessing steps were therefore reduced to the following (unless oth-

erwise noted):

1. Overlapped windowing:
    - Window length: 150ms
        - 15 samples on database 1
        - 300 samples on database 2
    - Window increment: 10 ms
        - 1 sample on database 1
        - 20 samples on database 2
2. Each window labelled as belonging to the movement at the most recent sample within the window
3. Data was split so as to maintain comparability with the previous studies [39, 47, 201]
    - $\frac{7}{10}$ for training and $\frac{3}{10}$ for testing on database 1
    - $\frac{2}{3}$ for training and $\frac{1}{3}$ for testing on database 2
4. Data normalised to zero-mean and standard deviation based on statistics calculated from the training data set

Architecture development and empirical testing used multiple randomised splits on randomised cross-sections of the subjects to enable rapid prototyping.

Since feature extraction is not performed for the neural networks X_ω is the actual input, which means each network solves the problem:

$$y = f(X_\omega, \phi) \tag{5.2}$$

where f is the neural network forward pass and ϕ are the parameters to be learnt.

The shape and intuition for X_ω is shown again in Figure 5.1 as it is critical for understanding the network design choices made later in this chapter which operate along the axes separately.

5.2.1 Baseline SVM-RBF

The SVM-RBF classifier from Chapter 4 was reimplemented here as an additional check against the changes in the methodology and to provide a baseline for performance.

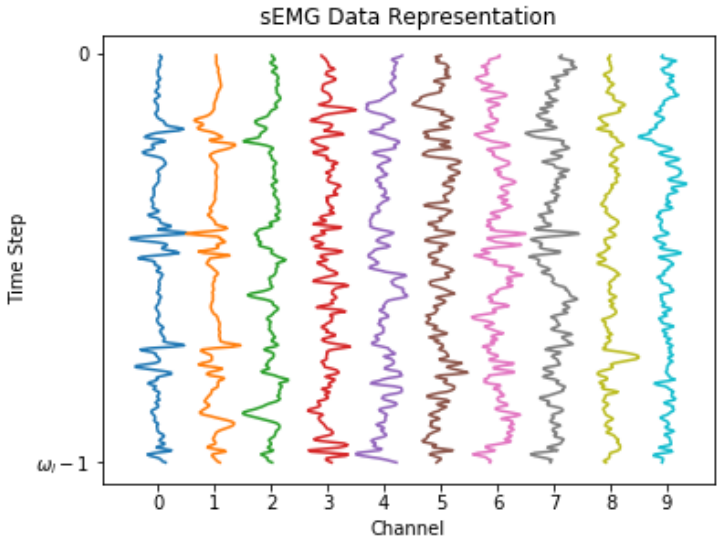

Figure 5.1: Visualisation of the raw sEMG window X_ω from database 1 as presented to the neural networks. On database 2 there are two additional channels, and the data is denser over the same period due to the increase in sampling frequency.

The MAV, WL and mDWT features were combined to make the input (using the same definitions as in Section 2.2.1) for the SVM-RBF normalised after extraction using training set data. Class weighting based on training set class frequency was also used.

Therefore the SVM-RBF solves a slightly different form of the problem:

$$Y = f(g(X_\omega, \theta), \phi) \tag{5.3}$$

where f is the SVM-RBF maximum margin computation ϕ is the set of support vectors, g is the feature extraction process and θ is the parameters of the extraction process.

The change in window size from the original benchmarks as well the alterations to the preprocessing, specifically the removal of the zero-phase filter, also make it essential to reimplement the SVM-RBF. Class weighting was used over the data augmentation strategies to maximise the amount of data for training and to make the training set for classifiers uniform throughout the methodology. The combination of features is used to help improve the overall performance of the classifier [191].

5.2.2 Baseline CNN

In order to test whether the performance improvement was due to architecture design or just the use of deep learning as a classification method a CNN architecture based on a "standard" network architecture, as used in many other domains[102], was devised as a litmus test. This style of design is also the approach seen in other work on sEMG classification with deep learning [28, 45, 47].

Table 5.2 shows the design for database 1 and Tables 5.2 and 5.4 show the designs for databases.

Layer Type	Output Size	# Filters	Filter Size	Stride	Padding	Activation	# Parameters
Input EMG	15x10x1						
Gaussian Noise	15x10x1					$\sigma = 0.001$	
Conv	15x10x128	128	3x3	1x1	Same	LReLU($\alpha = 0.1$)	1,280
Conv	15x10x64	64	5x3	1x1	Same	LReLU($\alpha = 0.1$)	122,944
Conv	15x10x32	32	5x3	1x1	Same	LReLU($\alpha = 0.1$)	30,752
Dropout	4,800					rate = 0.5	
Dense	128	128				LReLU($\alpha = 0.1$)	614,528
Dropout	128					rate = 0.5	
Dense	53	53				Softmax	6,837
						Total	776,341

Table 5.2: Baseline CNN architecture for database 1. Channels last format.

Layer Type	Output Size	# Filters	Filter Size	Stride	Padding	Activation	# Parameters
Input EMG	300x12x1						
Gaussian Noise	300x12x1					$\sigma = 0.001$	
Conv	15x12x128	128	3x3	1x1	Same	LReLU($\alpha = 0.1$)	1,280
Conv	15x12x64	64	5x3	1x1	Same	LReLU($\alpha = 0.1$)	122,944
Conv	15x12x32	32	5x3	1x1	Same	LReLU($\alpha = 0.1$)	30,752
Dropout	4,800					rate = 0.5	
Dense	128	128				LReLU($\alpha = 0.1$)	14,745,728
Dropout	128					rate = 0.5	
Dense	53	53				Softmax	5,289
						Total	14,905,993

Table 5.3: Baseline CNN architecture for database 2. Channels last format.

Layer Type	Output Size	# Filters	Filter Size	Stride	Padding	Activation	# Parameters
Input EMG	300x12x1						
Gaussian Noise	300x12x1					$\sigma = 0.001$	
Conv	15x12x128	128	60x3	20x1	Same	LReLU($\alpha = 0.1$)	23,168
Conv	15x12x64	64	5x3	1x1	Same	LReLU($\alpha = 0.1$)	122,944
Conv	15x12x32	32	5x3	1x1	Same	LReLU($\alpha = 0.1$)	30,752
Dropout	4,800					rate = 0.5	
Dense	128	128				LReLU($\alpha = 0.1$)	737,408
Dropout	128					rate = 0.5	
Dense	53	53				Softmax	5,289
						Total	919,561

Table 5.4: Baseline Adapted CNN architecture for database 2. Channels last format.

While these networks were based on a typical template: stacked, increasing size convolutions leading to dropout and dense layers, several notable deviations

are implemented that were discovered to be useful and improved upon the performance relative to uninformed designs.

The first is the use of additive Gaussian noise at the input, which is applied as a form of regularisation during training. This was based on the use of the same by Atzori *et al.* [47] in their architecture. Gaussian noise is a natural fit here to help guard against overfitting given the real-valued nature of the sEMG signals although the standard deviation was kept small at 0.001 based on empirical testing and the erratic results demonstrated by higher values in Atzori *et al.* [47].

Gaussian noise fills a similar role to SMOTE as used in the previous chapter due to the multiple training passes performed in the neural network training by presenting "new" variations during each pass. This provides a form of regularisation to the network helping prevent overfitting to individual data samples.

The second is the use of non-symmetric kernels in the convolutional layers. Typically square kernels are used since there is no a priori reason to assume the shape of useful kernels; therefore, squares are used to minimise assumptions. For this problem, it is known that the input (x_ω) will always be rectangular with a longer first (temporal) axis and that, based on the way features are extracted, the temporal axis is likely to have useful information embedded in it. This fact is taken advantage of by extending the filter kernel shape in the second and third convolutional layers which widens the receptive field of each layer along the temporal axis. This led to a small boost in the overall performance of networks using this rectangular filter kernel compared to square counterparts using either of the rectangular filter's dimensions as its width.

The non-symmetric kernel technique is extended in the second database 2 design (Table 5.4) based on performance improvement and parameter reduction results from the novel network. The first layer's convolutional kernel is expanded along the temporal axis by a factor of 20 as is the stride in the same axis. This measure was designed as an approach to reducing the number of parameters in the original design, from Table 5.3, to reduce the computational load caused by using an order of magnitude more parameters than the network used on database 1 (Table 5.2). This increase in the number of parameters is caused partially by the increase in the number of channels but overwhelmingly by the 20x increase in data points in the window due to the 2kHz sampling frequency in database 2 compared to the 100Hz used in database 1. The expanded kernel, therefore, represents the first convolutional layer viewing the same time period with the same increment in both databases.

5.2.3 Temporal-to-Spatial Network

The major novel design is designated a Temporal-to-Spatial (TtS) network because of the explicit enforcement of separation of the data in the early layers of the network forcing learning of temporal features before later combining them across channels (spatially). Intuitively this is based on the EMG feature extraction literature which nigh invariably extracts features on a per channel basis [55, 68, 99, 100, 200, 204] which provides a strong indication that extracting features from channels individually is a useful technique. This is directly contrary to the methodology employed by the contemporary work benchmarked against [39, 47, 201] which enforces early spatial-only features and more typical architectures like the networks described in Section 5.2.2.

Figure 5.2 shows a visualisation of the general TtS network and Tables 5.5, 5.6, 5.7 shower a finer grained breakdown of the architectures used on the different databases.

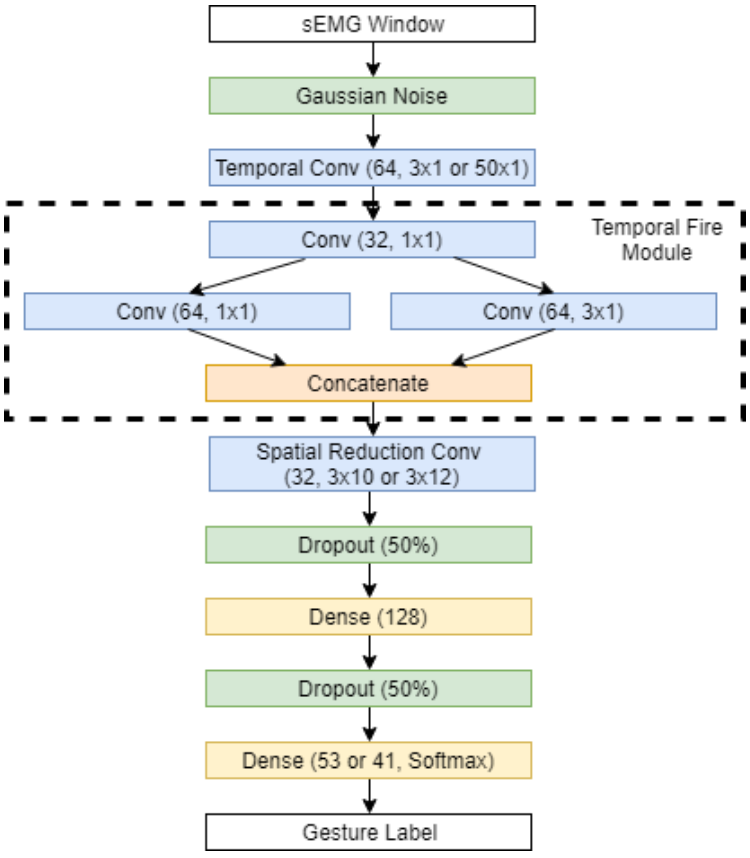

Figure 5.2: Visualisation of the Temporal-to-Spatial (TtS) network. Brackets show the number of filters followed by filter size. The "or" statements in filter sizes show the different sizes for database 1 and database 2, respectively.

Layer Type	Output Size	# Filters	Filter Size	Stride	Padding	Activation	# Parameters
Input EMG	15x10x1						
Gaussian Noise	15x10x1					$\alpha = 0.001$	
Conv (Temporal)	15x10x64	64	3x1	1x1	Same	LReLU ($\alpha = 0.1$)	256
Temporal Fire	15x10x128	(32, 64, 64)	(1x1, 1x1, 3x1)	1x1	Same	LReLU ($\alpha = 0.1$)	10,400
Conv (Spatial)	15x10x32	32	3x10	1x1	Same	LReLU ($\alpha = 0.1$)	122,912
Dropout	4,800					rate = 0.5	
Dense	128	128				LReLU ($\alpha = 0.1$)	614,528
Dropout	128					rate = 0.5	
Dense	53	53				Softmax	6,837
						Total	754,933

Table 5.5: TtS architecture for database 1. Channels last format.

Layer Type	Output Size	# Filters	Filter Size	Stride	Padding	Activation	# Parameters
Input EMG	300x12x1						
Gaussian Noise	300x12x1					$\alpha = 0.001$	
Conv (Temporal)	12x12x64	64	3x1	1x1	Same	LReLU ($\alpha = 0.1$)	256
Temporal Fire	12x12x128	(32, 64, 64)	(1x1, 1x1, 3x1)	1x1	Same	LReLU ($\alpha = 0.1$)	10,400
Conv (Spatial)	12x12x32	32	3x12	1x1	Same	LReLU ($\alpha = 0.1$)	147,488
Dropout	4,608					rate = 0.5	
Dense	128	128				LReLU ($\alpha = 0.1$)	14,745,728
Dropout	128					rate = 0.5	
Dense	41	41				Softmax	5,289
						Total	14,909,161

Table 5.6: Unadapted TtS architecture for database 2. Uses the same design as for database 1 with the numbers of parameters updated to account for the larger input size. Channels last format.

The key points of the architecture are the use of Gaussian noise as regularisation, the first Temporal Convolution, the Temporal Fire Module and the Spatial Reduction Convolution before the final dense classification portion of the network.

The same Gaussian noise parameters are used as in the baseline CNN, i.e. zero mean with standard deviation 0.001, based on empirical testing of a pool of candidates and the work by Atzori *et al.* [47].

The Temporal Convolution acts as the first layer of enforcement of temporal feature learning by restricting the kernel to viewing only a single channel at a time. On database 2 two different variants were possible, the first, naive imple-

Layer Type	Output Size	# Filters	Filter Size	Stride	Padding	Activation	# Parameters
Input EMG	300x12x1						
Gaussian Noise	300x12x1					$\alpha = 0.001$	
Conv (Temporal)	12x12x64	64	50x1	25x1	Same	LReLU ($\alpha = 0.1$)	3,264
Temporal Fire	12x12x128	(32, 64, 64)	(1x1, 1x1, 3x1)	1x1	Same	LReLU ($\alpha = 0.1$)	10,400
Conv (Spatial)	12x12x32	32	3x12	1x1	Same	LReLU ($\alpha = 0.1$)	147,488
Dropout	4,608					rate = 0.5	
Dense	128	128				LReLU ($\alpha = 0.1$)	589,952
Dropout	128					rate = 0.5	
Dense	41	41				Softmax	5,289
						Total	756,393

Table 5.7: TtS architecture for database 2 with adaption for higher sampling frequency. The key difference is the large filter size and large step size in the temporal dimension of the first convolutional layer. Channels last format.

mentation, with the 3x1 kernel (Table 5.6) as designed for use on database 1 and the second with a 50x1 kernel and a stride of 25x1 (Table 5.7). The exact value of this latter set of hyperparameters was based on testing; however, the principle was to account for the much higher (20x) sampling frequency in database 2. This adaptation for higher sampling frequencies vastly reduces the necessary number of parameters while also improving performance (as can be seen by comparing the tables) making it a powerful technique for adapting networks to different electrodes while controlling the computational complexity. Both the adapted and unadapted versions are including in testing for comparison.

The Temporal Fire Module is a co-opt and adaptation of the Fire module that Iandola *et al.* used to reduce the number of parameters necessary for AlexNet level accuracy on Imagenet by 50x as part of their SqueezeNet architecture [205]. The Fire module, which learns useful inter-filter features, was taken and an additional temporal feature constraint was added by replacing the 3x3 kernel in the original with a 3x1 kernel.

Early temporal convolutions were inspired by work on deep learning for EEG where good performance was found utilising temporal convolutions [206]. Here the ideas are expanded and adapted for use with sEMG. Some of the latest research in the sEMG classification field has also shown promising results with temporal convolutions [45].

Lastly, the Spatial Reduction Convolution layer uses a kernel that acts on all channels enforcing the spatial portion of the Temporal-to-Spatial architecture. Intuitively this layer is able to extract cross-channel features which have a strong prior on being the higher level of useful feature, e.g. the MAV per channel across a window can be well classified. Therefore this expectation that useful top-level features will be combinations of low-level temporal ones is explicitly enforced. This allows extending of the receptive field of the layer, which allows it to build a representation based on a larger amount of the original data. It was found that this single kernel worked better than adding additional layers in other places to increase the receptive field by the same amount.

5.2.4 Additional Design Choices

There are six major design choices for the new neural networks not covered explicitly up to this point.

The first is the choice to train classifiers on each subject individually rather than training a single classifier or set thereof to work on all subjects. This approach is adopted throughout this work up to this point and very common in the wider literature. The reason for this is purely practical, training a classifier across

subjects performs significantly worse than training to an individual likely due to the differences in the biophysical properties of the forearm between subjects as well as other concerns such as relative electrode placement. In Chapter 7 methods of making classifiers more useful across subjects are explored.

The second is not making use of ensembles. It is well known that using ensembles can improve performance; however, this increases the computational complexity, the method complexity and introduces numerous additional hyper-parameters. Ensembles were omitted to reduce the complexity of the benchmark so as to ensure the results were as clear cut and comparable as possible to other studies and presented the building blocks for more complicated classifier arrangements.

The third is the usage of convolutional neural networks rather than recurrent networks. A plethora of recurrent architectures were tested including Long Short-Term Memory (LSTM) networks [207], attentional interfaces and other variants of recurrent architectures [161]; however, they were found to be more complicated and perform worse than convolutional solutions. This finding is also implied in the literature by the fact that other neural network solutions to the classification problem are not recurrent. It is not clear why recurrent networks generally appear to perform worse than convolution networks given the problem's temporal element. However, based on the evidence, convolutional architectures were chosen as the appropriate solution.

The fourth point is the lack of pooling layers in any of the designs. Pooling layers act as downsampling within a network generally averaging or taking a maximum activation. In larger networks, these layers can be vital to managing the numbers of parameters and are a staple of convolutional network design. Similar to the work of Springenberg *et al.* [208] this design principle was re-evaluated in the sEMG context, and it was found that removing pooling, even without introducing alternative downsampling, e.g. via stride increase, consistently improved performance. This was generally, although not universally, true on all networks, including those with nearly 15 million parameters implying it may not only be a function of the generally small network size. A potential implication, therefore, is that the information extracted over different time periods is not directly generalised to larger time periods, which is the expected behaviour, given that the underlying signal is non-stationary.

Fifthly, training was achieved using a set number of epochs (10) for each model as in each case this was found suitable for convergence, optimisation was performed using the Adam algorithm with default parameters. Using a set number of epochs like this, as opposed to a dedicated validation data set, makes under

or overfitting more likely, which is why the number of epochs is conservative to avoid overfitting. The set number of epochs was used to ensure the data split was the same as in previous studies for greater comparability.

Finally, since it is known that the data is heavily imbalanced and that this is detrimental to the desired classification, class weighting was implemented during training to ensure the networks did not end up heavily biased towards the *rest* class. Several weighting strategies were trialled; however, the following lead to the best results:

$$w_i = 1 + log_2(\frac{n_{max}}{n_i}) \tag{5.4}$$

where w_i is the weight for class i, n_i is the number examples of class i and n_{max} is the maximum number of examples of any class. This leads to a weighting scheme where examples of the most represented class (*rest*) have a weighting of 1 while other classes have their associated error signals amplified but to a much smaller degree than a frequency based approach would, which helps to guard against amplification of noise during training and from particular samples dominating a batch update.

5.2.5 Adam Algorithm

The Adam algorithm [113] is one of the most popular general-purpose optimisation algorithms at the time of writing and is the algorithm used to train all the networks used in this and future chapters. Adam takes its name from ADAptive Moment estimation, which is key to its performance. The reason for its popularity is its similar or better performance to contemporary algorithms coupled with a significant decrease in convergence time for many applications without the need for parameter tuning in most circumstances.

The key points of the Adam algorithm are its usage of a smoothed gradient vector based on a decaying average of past gradients and past gradient squares (first and second moments) which allows, effectively, individual adaptive learning rates for each parameter in the network. The algorithm is included as pseudocode in Algorithm 5.1.

The bias correction steps in the last few lines are not strictly necessary and are not implemented in some popular frameworks such as Keras [209]. The correction terms exist because the algorithm computes moving averages and, therefore, the first few steps will be significantly biased towards the initialisation of m_0 and v_0 which may cause inappropriate updates in some circumstances.

The hyperparameters α, β_1 and β_2 typically perform well with the default values of $\alpha = 0.001$, $\beta_1 = 0.9$ and $\beta_2 = 0.999$ [113] but can be adjusted to fit the

Algorithm 5.1 The Adam algorithm [113] which is popular due to its improvements in convergence time on many problems while having matching or better performance. Note ϵ is a small value to prevent division by 0 issues, typically set to 10^{-8}.

Input: α : Step size, β_1 : Exponential decay rate for 1^{st} moment estimate, β_2 : Exponential decay rate for 2^{nd} moment estimate, $f(\theta)$: Cost function with parameters θ, θ_0 : Initial parameters
Result: θ_t : Updated parameters

$m_0 = 0$, // 1^{st} moment estimate
$v_0 = 0$, // 2^{nd} moment estimate
$t = 0$, // Current time step
while θ_t *not converged* **do**
  $t = t + 1$
  $g_t = \nabla_\theta f_t(\theta_{t-1})$ (// gradient vector for current time step
  $m_t = \beta_1 m_{t-1} + (1 - \beta_1) g_t$ // biased first moment estimation
  $v_t = \beta_2 v_{t-1} + (1 - \beta_2) g_t^2$ // biased second moment estimation
  $\hat{m}_t = m_t / (1 - \beta_1^t) g_t$ // initialisation bias correction
  $\hat{v}_t = v_t / (1 - \beta_2^t) g_t$ // initialisation bias correction
  $\theta_t = \theta_{t-1} - \alpha \hat{m}_t / (\sqrt{\hat{v}_t} + \epsilon)$
end

particular needs of a problem if necessary. When changes are made, they are typically to reduce β_1 to reduce the effect of the first moment although no benefit was found in doing this in the problems explored here.

5.2.6 Comparison to Contemporary Networks

Two contemporary network architectures were reimplemented that both claimed to have best in class performance for the classification problem [39, 47]. Both made use of NinaPro databases 1 and 2 in their testing, which provided an ideal testing ground for comparison.

Atzori *et al.*'s network [47] was reimplemented based on their description and communication with the authors about some of the implicit choices made in the architecture, such as the padding on convolutional layers. Their network uses the same data format as the new networks implemented here, so no special considerations were necessary for comparability.

Geng *et al.* [39, 201] forgo direct windowing in favour of an "instantaneous" approach where majority voting on all samples in a window is used to classify windows. Therefore data is presented in training as x, y pairs where x is only a single sample of sEMG and y the associated label rather than presenting a full window. At test time the same windows were used as for the other classifiers but

evaluated each individual sample in each window and took the unweighted majority vote as the predicted label. Geng *et al.* provided an implementation of their network [38] which was fixed to work with current generation hardware however this code was not compatible with the testing methodology. Therefore, the network was reimplemented directly using the code provided to inform implicit design choices in the architecture.

Implementations were validated by testing them under the original methodology and verifying that the results were similar to the original studies. Exact replication of results was not possible due to stochasticity in the training process. In the reimplementation, random number seeds were set to help reduce this issue. However, even then it is not entirely possible due to stochastic processes present in GPU processing software such as cuDNN [210] which are often used in training.

5.2.7 Filter Visualisation

Visualisation techniques can be useful to aid in understanding how a neural network functions and what sort of features it extracts as well as in providing insight into the data as a whole. There are several ways to help visualise a neural network including but not limited to weight visualisation, generating inputs that maximise the activation of a particular neuron, t-SNE and occlusion mapping [108]. The t-SNE algorithm, however, is less useful here than in the previous chapter or image recognition since clustering of similar examples does not lend itself well to visualisation.

Weight visualisation is the straightforward plotting of weights to see what sort of pattern will fit them best. This technique is most useful in the earliest layers since these deal with the input directly and becomes less useful later on due to the abstract representation later network layers work with. In images, this visualisation generally takes the form of a 2D image plot where edge, shape and colour detectors become visible in the pixel pattern.

Maximising activation of a neuron either by finding examples in the data set or by using gradient descent to optimise an input for the neuron is also a useful technique for finding biases that may be present in the network and determining prime examples of classes of phenomena.

Lastly, occlusion mapping involves running the test procedure with sections of all the input data removed. Here this means setting the signal to zero amplitude over different segments and determining how this affects performance as an indicator of which segments of the input the network focuses on at test time.

5.3 Results and Discussion

Tables 5.8 and 5.9 show the performance of the different networks on database 1 and 2, respectively. Results are ordered in terms of ascending macro-average accuracy. The micro-average accuracy is also presented to demonstrate the disparity of results under the two metrics.

Database 1				
Classifier	**Macro Accuracy**		**Micro Accuracy**	
	$\mu_{subj}(\%)$	μ_{rank}	$\mu_{subj}(\%)$	μ_{rank}
Atzori *et al.* [47]	51.4	4.9	71.0	5.0
Geng *et al.* [39, 201]	59.0	3.6	79.5	1.1
SVM-RBF	60.4	3.1	77.8	2.5
Baseline CNN	65.0	2.3	77.1	3.7
TtS	66.6	1.2	77.5	2.7
TtS*	69.3	/	78.0	/

Table 5.8: Summary of classifier performances on database 1. The value μ_{subj} is the inter-subject mean accuracy calculated using either the micro-average or macro-average for a given classifier, μ_{rank} is the inter-subject mean rank (range 1-5). *Repetition 1 removed from test set without retraining, not included in ranking.

Database 2				
Classifier	**Macro Accuracy**		**Micro Accuracy**	
	$\mu_{subj}(\%)$	μ_{rank}	$\mu_{subj}(\%)$	μ_{rank}
Geng *et al.* [39, 201]	24.5	7.0	58.2	6.8
Atzori *et al.* [47]	50.2	6.0	61.0	6.0
Adapted Baseline CNN	57.0	4.9	64.8	5.1
SVM-RBF	60.5	3.8	71.2	1.4
Unadapted TtS	63.1	2.7	69.3	2.5
Unadapted Baseline CNN	63.2	2.5	68.6	3.5
TtS	67.8	1.0	69.5	2.6
TtS*	70.6	/	70.9	/

Table 5.9: Summary of classifier performances on database 2. The value μ_{subj} is the inter-subject mean accuracy calculated using either the micro-average or macro-average for a given classifier, μ_{rank} is the inter-subject mean rank (range 1-7). *Repetition 1 removed from test set without retraining, not included in ranking.

On both databases, the TtS network significantly outperforms all others tested in terms of macro-average accuracy. The micro-average is included for comparison against the comparison networks [39, 47, 201] and to show that it again masks the performance of virtually all but *rest* class as demonstrated by Figure 5.3 which is not desirable for a benchmark unless there is a strong prior on the test time class distribution.

The statistical significance of the TtS network's performance was confirmed using the Friedman test and post-hoc Holm procedure.

On database 1, the Friedman test determined that the classifiers performed differently at $p < 5.8 \times 10^{-33}$. The Holm procedure then showed that the TtS outperformed the others at $p < 1.3 \times 10^{-2}$. The TtS performed best on 22 of the 27 subjects in the database.

On database 2, the Friedman test determined that the classifiers performed differently at $p < 4.4 \times 10^{-152}$. The Holm procedure then showed that the TtS outperformed the others at $p < 1.3 \times 10^{-3}$. The TtS performed best on all 40 subjects.

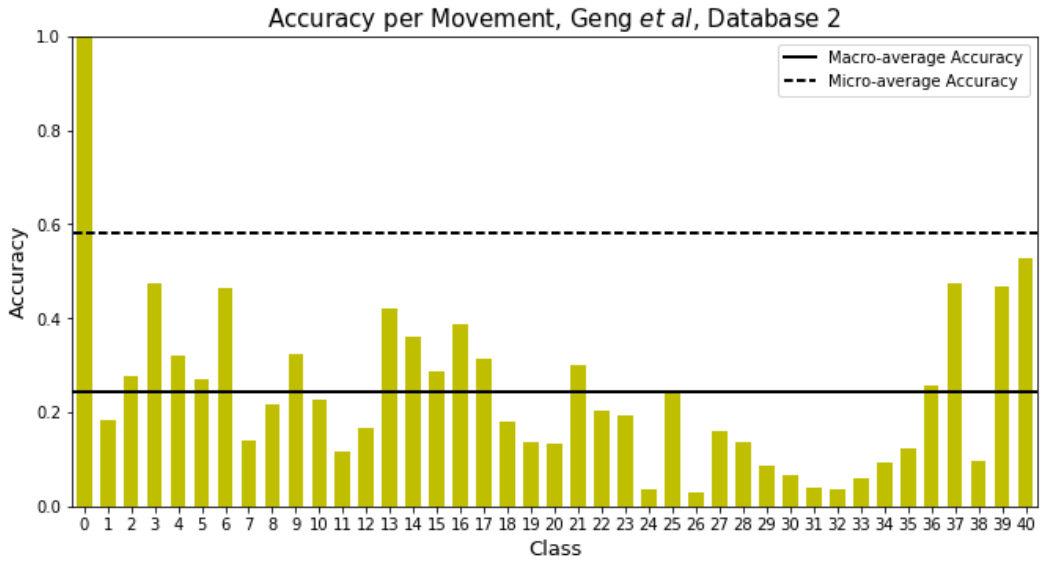

Figure 5.3: Accuracy of individual movement classes compared to the overall micro-average and macro-average accuracy for Geng *et al.'s* classifier [39]. The over-reporting of the micro-average is demonstrated again under the assumption that the test time class distribution is unknown.

Figure 5.3 demonstrates the underlying cause for Geng *et al.'s* significantly reduced performance on database 2. The instantaneous classification method employed by the network naturally tends to high *rest* classification rates due to *rest* generally being a lower energy state. The Figure shows this in the near 100% *rest* classification rate, but this comes at the expense of the other classes being misclassified as *rest* and is amplified as the number of examples in each window increases which is what causes this much lower macro-average accuracy on database 2.

Geng *et al.* present an alternate formulation of the classification problm they describe as trial based classification; classification by majority vote over all labels a priori known to be part of a single repetition of each movement. In this setup,

their network achieves 87% macro-average accuracy on database 1 and 20.1% on database 2. The TtS network achieves 92.9% on database 1 and 95.0% on database 2. This trial based classification, however, is not a practical measure since it requires the movement to be complete before classification can take place. On this data, this is equivalent to introducing a prior on movement duration and a latency of the order of seconds which is much higher than the generally accepted $\sim 200ms$ maximum acceptable control latency [203] as well as requiring prior knowledge on when a movement begins and ends which is generally unavailable.

All classifiers exhibit a degree of inter-subject variability. This can be quantified in terms of the standard deviation of the macro-average accuracy between subjects. On database 1 this is $4.8 - 5.7\%$ and on database 2 this is $5.6 - 6.3\%$. This shows that there is a high degree of variation between subjects.

5.3.1 Effect of Repetition Number

It was discovered that, on average, the first repetition of each movement that each subject performed had a significantly lower classification rate compared to later repetitions. This result is illustrated in Figure 5.4 on the TtS network, although this effect is present in all classifiers.

It may be posited that this is likely a, difficult to avoid, artefact of the experimental procedure and sEMG data gathering in general; the first, or first few times a subject performs a movement or after having performed another movement there is a higher likelihood of error. This is backed up by the observation from Figure 5.4 that the first repetition is significantly worse than the others. To a lesser degree the later repetitions on database 1 also show a decline in performance. This is unlikely to be from muscle fatigue (due to the limited number of repetitions [141]) but may be caused by preemption of the video stimuli or inattention causing non-optimal replication of the movement.

Regardless the underlying cause, it may be beneficial for studies to omit the first repetitions from their data corpus when considering these databases and to analyse other data sets for similar effects. However this must be traded off with the loss of valuable data so it is important to decide this on a case-by-case basis. Tables 5.8 and 5.9 include the performance of the TtS with repetition 1 removed from testing to demonstrate the effect; other networks see a similar increase in performance. Removing different repetitions has a much smaller effect on performance ($< 1\%$).

Figure 5.4 shows that the reported effect of the repetition is significantly reduced under the micro-average which implies the effect is likely not present or significantly reduced in the *rest* class. This was substantiated by review of the per-

formance of *rest* across repetitions which did not show the bias based on repetition number.

This is an issue not frequently addressed in the literature that, as is clear from Tables 5.8 and 5.9, can make a significant difference to the final performance and therefore is worthy of additional consideration in studies going forward.

5.3.2 Distribution of Classification Performance

In one of the original studies on the NinaPro databases [37], it is noted that average classification performance is highest in the middle of each movement and degrades significantly towards the start and end of each movement. The graph from the original study is shown in Figure 5.5.

In this work, a similar effect was found, shown in Figure 5.6, however, while the curve is similar here, the minimum performance level is higher and the performance ramp during movement onset is significantly steeper. The difference in the ramp is due to the class weighting implemented here, which reduces the bias towards the *rest* class that the data naturally exerts on the training process. The difference in the minimum performance is due to the increase in performance for non-*rest* movements when using the TtS network rather than the original study's classification solution.

The original study asserts that the incorrect classifications in these regions are best characterised as delays to the correct classification. This assertion holds true as long as the misclassifications are of the *rest* class since *rest* precedes and follows each movement and if *rest* is also well classified. The assertion fails, however, if the misclassifications are of other classes or if *rest* is not well classified throughout. Review of the confusion matrix in the early and late movement regions indicated that while classes were often misclassified as *rest*, they were also likely to be misclassified as other movement classes, meaning that these regions cannot be assumed to just be an added latency.

Therefore, at least in the case of the new networks presented here, it is better to characterise these misclassifications as a subset of the broader performance problem that may have solutions in the form of alterations to labelling methods and experiment design as well as in classifier design. The ideal being to design a system that is able to make consistent hard cutoffs between when a movement class starts and ends. This line of research is explored in the next two chapters, particularly with regards to the labelling methods chosen in experiment design.

5.3.3 Filter Visualisations

This section focuses on visualising data from the TtS network on database 2 due to the network's significantly better performance and the data's high sampling frequency leading to more interpretable visualisations. The first validation fold of subject 1 is used to generate the examples.

Figure 5.7 visualises the weights from the first temporal convolution layer (50 weights) where the network acts on the raw incoming sEMG and so is most interpretable. Here the channel-wise, temporal nature of the signals is taken advantage of to plot the weights as though they were a signal representing the general waveform being searched for by each filter. Figure 5.8 connects with Figure 5.7 by showing the corresponding input waveform snippet that maximises the neuron activation for the same filters. Due to the interaction with striding this snippet is the same length as the stride (25 samples) and produces a maximal activation over that time period which then maximises the activation across the whole window when repeated except for minor variation in the first and last segment. Variation in these segments is due to them being evaluated only once as the filter strides through the sample.

Looking at Figure 5.7 it is possible to see that the filters do not look for a clean waveform but instead fit a noisy variant which is shadowed by the input segment that maximises each neuron's output in Figure 5.8. This implies that it may be worthwhile to explore stronger regularisation options to attempt to reduce the noise seen in the filters.

Figure 5.9 shows a variety of the waveforms that produce maximal activations for the filters in the spatial reduction layer. In contrast to the inputs producing maximal outputs for the first temporal layer, the waveforms here show patterns that span the whole window length. This demonstrates that despite the repeating nature necessary for maximal activation in earlier layers, the later layers are able to distinguish waveforms with more complex, time-varying shapes.

Noise is still a relevant factor in these higher level representations, as can be seen via inspection, further reinforcing that stronger regularisation may be of use. Despite the noise, however, some of the filters show a clear amplitude bias preferring non-zero mean amplitude across the window, which is similar to the MAV feature which has shown good performance on the problem. The MAV is particularly interesting since it requires minimal computation to compute compared to features like the mDWT. The fact that the network utilises a similar feature at higher levels of representation suggests that augmentation of the input vector with the MAV may be a useful line towards improvement that keeps the benefits of the end-to-end learning system while potentially promoting better performance.

Searching the data for examples that created high neuron activations yielded what were, effectively, adversarial examples. A few of the data windows contained large voltage spikes and little else that often led to high neuron output despite not being representative of any particular class. This echoes a fundamental problem with deep neural networks which is that, since the effective classification function is not smooth throughout the input space, deviations from the input space encoded by the training dataset can cause unexpected results [211]. It is therefore interesting to note that, at least for this data set, spikes in the input pattern may cause particular problems. This is relevant because such spikes may be caused by electrode slippage during normal usage, meaning for practical applications precautions may be necessary to ensure adequate performance.

5.4 Conclusion

A novel neural network architecture (TtS network) was designed and evaluated in this chapter. It was shown to significantly outperform SVM and CNN baselines as well as other contemporary networks. The architecture can also be simply adjusted to account for varying sampling frequencies; the first convolution filter may be adjusted to cover an equivalent time period. This was demonstrated to improve performance, and makes the network much easier to reuse as only one filter shape needs adjustment rather than the entire architecture.

The TtS design also produced an improvement in the performance curve plotted against normalised movement position implying it generally reached a correct classification quicker than other networks. Analysis of the confusion matrix for this region, however, showed that the misclassification pattern was not solely confusion with *rest* which implies that, contrary to previous assumptions [37], this is not an added latency but rather a manifestation of the overall performance.

Visualisation of the TtS network shows that the filters look for somewhat noisy waveforms, which implies that stronger regularisation may lead to improved performance since the filters may be more likely to find less noisy waveforms to match. It is shown that the higher level features often have non-zero mean amplitudes similar to how the MAV extracts information and that the network is potentially sensitive to high amplitude voltage spikes, which may result from electrode slippage. This has implications for supporting software design since, if it is known that slippage may essentially create adversarial examples, software development can smooth or omit these to prevent errors.

The analysis also showed additional evidence for the need for the methodological improvements described in the previous chapter. Specifically, it was shown that

repetitions are unequal in terms of resultant performance and therefore that there was a need for the stratified cross-validation technique and that the micro-average accuracy leads to unrepresentative performance.

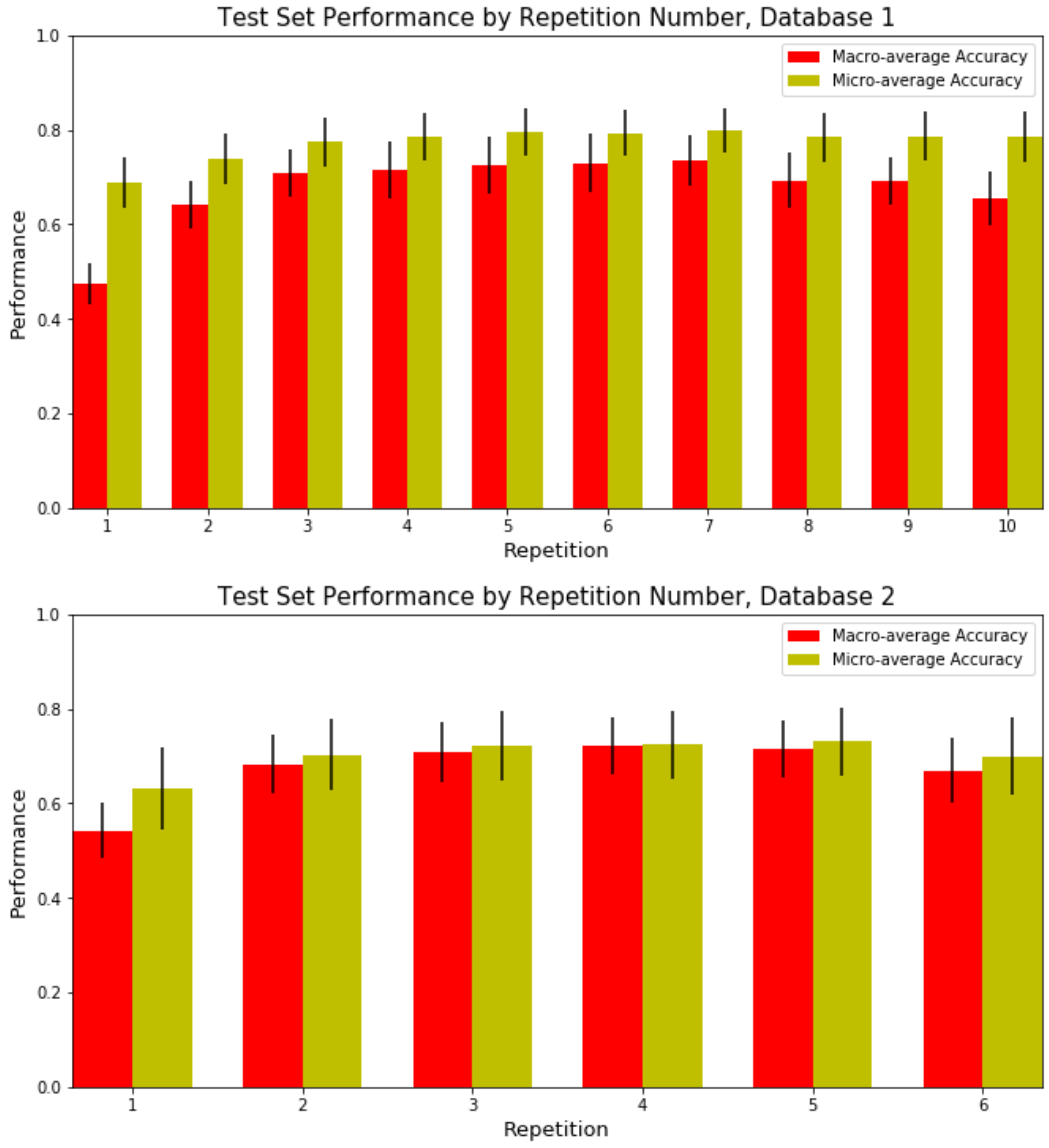

Figure 5.4: Performance by repetition number of the TtS network on the two databases (mean across subjects) demonstrating a distinct performance reduction in the first repetition for both databases. Lines indicate 1 standard deviation.

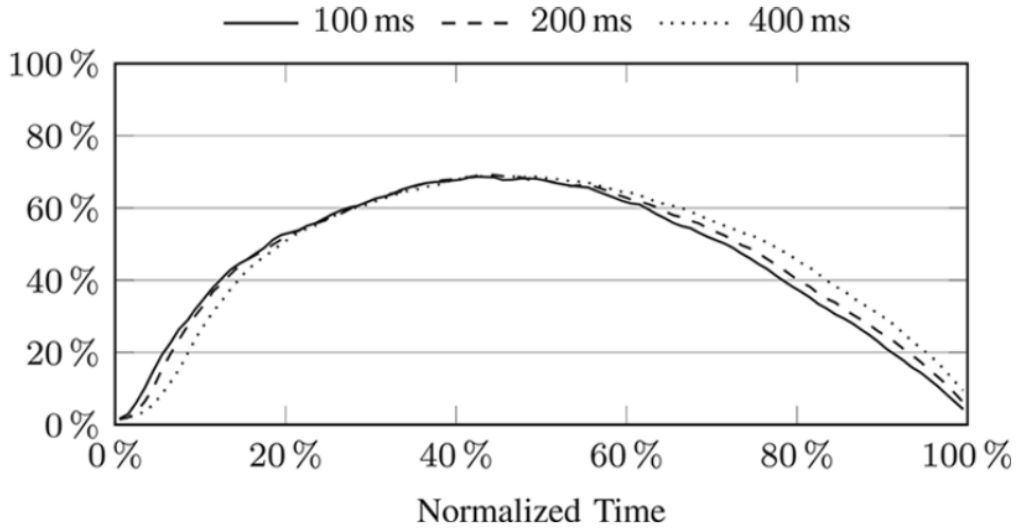

Figure 5.5: The average performance of the SVM classifier given the normalised time-position within movement for different window lengths [37]. Drop off in performance is notable as the distance from the centre increases.

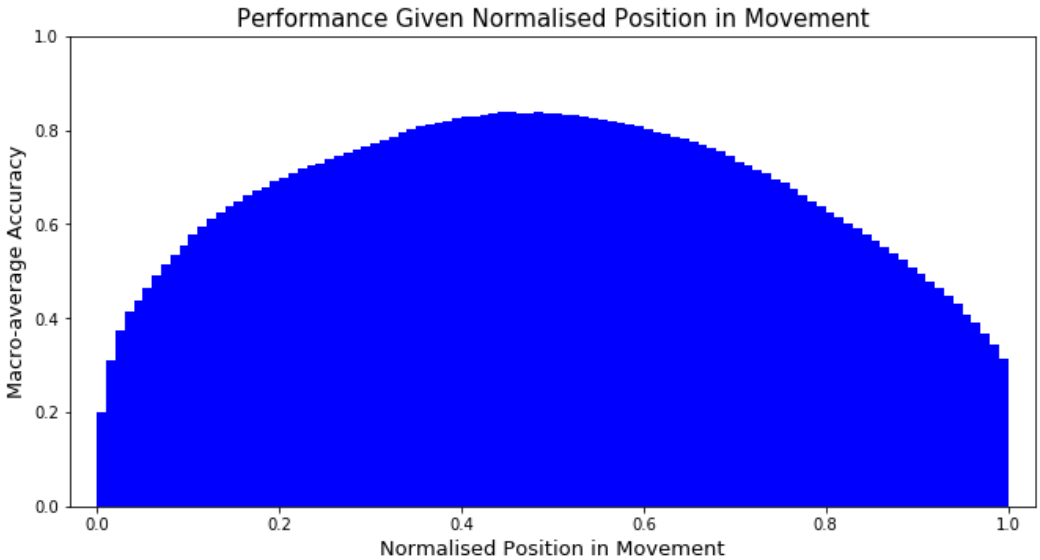

Figure 5.6: Performance of the TtS network given normalised time position within the movement showing similar overall trend as in Atzori *et al.* [37] but improved overall performance and faster onset detection.

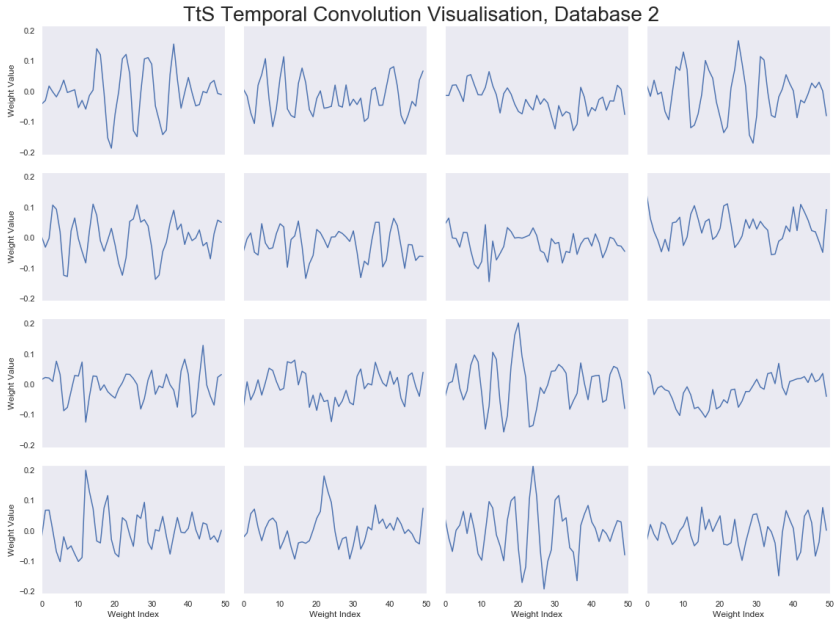

Figure 5.7: Visualisation of the weights of the first 16 filters in the first temporal convolutional layer of the TtS network on database 2. The visualisation shows that there is potentially noise in the weights as well as the underlying signals.

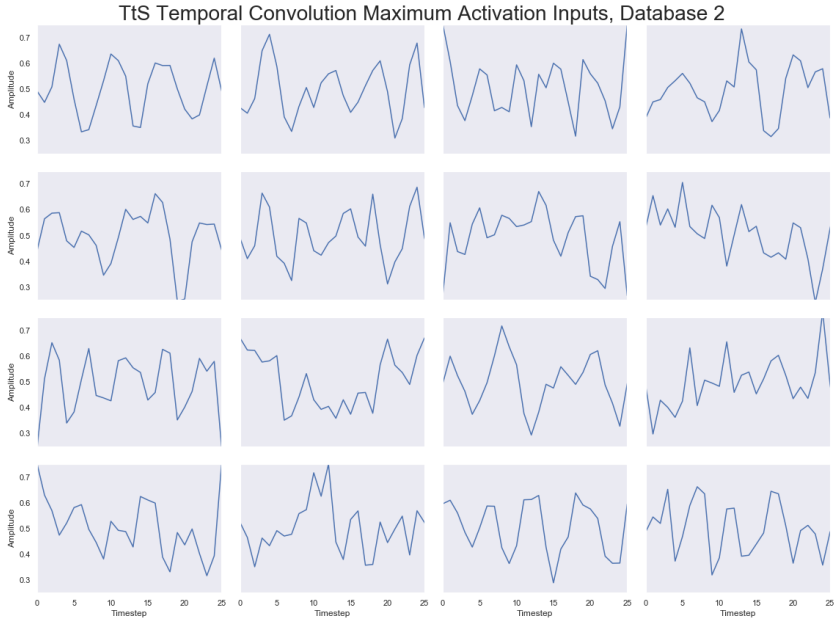

Figure 5.8: Visualisation of the repeated input waveform that creates the maximum activation for the first 16 filters in the first temporal convolutional layer of the TtS network on database 2.

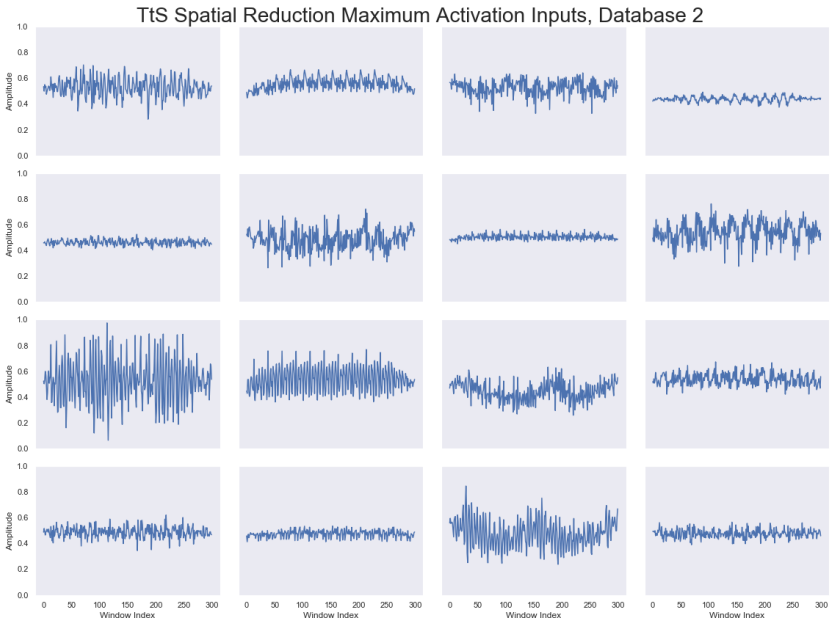

Figure 5.9: Visualisation of the window that creates the maximum activation for the first 16 filters of the spatial reduction layer of the TtS network on database 2.

Chapter 6

Compact Deep Neural Networks with Comparison of Electrodes

6.1 Introduction

This chapter presents a new study that aims to simultaneously compare a set of low-cost commercial electrodes with an expensive medical grade alternative. Specifically, the Myo Armband [78] is used to gather data simultaneously with the Delsys Trigno Wireless System [93]. Usually data would be captured from one set of electrodes at a time; however, this can lead to biases since the subject will be familiar with the experiment if multiple tests are conducted with different electrodes.

This study eschews running the same experiment multiple times with different electrodes so that it is possible to evaluate performance from exactly the same movements. This means that the experimental bias is largely limited to the placement of electrodes, which allows comparison of electrodes in a different light to other studies. The study mainly aimed to provide useful data for evaluation of the Myo Armband relative to a medical grade alternative due to its low-cost and easy availability.

A benchmark is presented using both feature-based classification and the TtS network to demonstrate the relative performance of the two data sets and to compare and contrast performance trends.

Another novelty is a refinement of the TtS network design which vastly reduces the number of parameters necessary for similar levels of performance as the original [51]. This extension is evaluated on a piece of embedded hardware (the Jetson TX2 [103]) to demonstrate the importance of compact neural network designs for applications where processing power is limited. It is shown that in terms of pro-

cessing time, the Compact TtS achieves significantly faster classification making it more suited to computation-constrained environments.

Finally, the study is used as a vessel to explore how labelling methods affect classifier performance and how a commonly used method, the GLR, generally leads to labelling different from what an expert would define as the ground truth which suggests it is sufficient for classifier training but potentially not good enough for evaluation.

6.2 Methods

In order to capture the full process from study to classification, the study was designed and conducted first, and then the TtS network was adapted and reviewed in the context of the new data. From there the network was optimised to minimise the number of parameters and then tested on the Jetson TX2 [103], a small embedded device with GPU capabilities, to evaluate the improvements to classification time in a real-world setting. An image of the device is shown in Figure 6.1

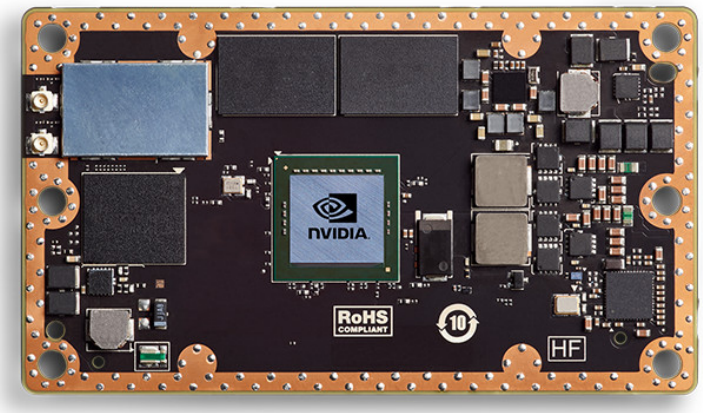

Figure 6.1: Image of the Jetson TX2 the embedded device used in this chapter [212]. The device measures 50mm by 90mm.

6.2.1 Experiment Overview

The main aim of the experiment was to contribute a useful addition to the data available to the research community while also providing a testbed for further study. Since the resources available made performing a large or complex study untenable, it was decided to focus on comparing the low-cost, consumer-grade Myo Armband [78] with a much more expensive (at least 90x more expensive) medical grade electrode set, the Delsys Trigno wireless system [93]. While gath-

ering new data on the two systems is useful in its own right, other concurrent studies tackled evaluation of the Myo's relative performance [35], therefore, in order for the experiment to provide a unique perspective, data was captured from both systems simultaneously.

The capture of data simultaneously avoids common issues including subjects performing movements differently between experimental runs, differences in fatigue and attention levels as well as subject familiarisation with experimental technique. Simultaneous capture does introduce new issues, most importantly, the fact that since all the electrodes are surface electrodes, they cannot share the same physical space. This introduces a new set of biases however also presents an opportunity to explore the relative performance from a different perspective which is vital for growth in the field particular as the Myo Armband has gained popularity in the research community [98, 213–219].

The experiment and all related procedures were approved by the University of Sheffield Ethics Committee.

The key points of the study data are:

- 14 Gestures + *rest*
  - Each gesture is stationary (held)
  - *Rest* is treated as an additional gesture
- 6 Repetitions of each gesture
  - Held for 10s each
  - Strongly delimited
- 10 Subjects
  - At least 1 dataset for each
  - Various supplementary data on some subjects
  - All healthy
- Myo Armband
  - 8 channels of sEMG
  - 3 channels of accelerometer data
- Delsys Trigno
  - 5 channels of sEMG
  - 5 channels of accelerometer data

The main limitations of the study are the number of participants and repetitions as ideally hundreds of subjects would be studied performing as many repetitions as possible however resource constraints made this infeasible. Despite this, the study size in both terms is still quite comparable to other studies [19, 20, 29, 33, 220–222]. Having data on 10 subjects ensured a reasonable variety of data to examine, and 6 repetitions meant the whole experimental procedure could be kept to under an hour which was useful for ensuring continuous subject attention and avoiding issues with muscle fatigue. From previous chapters, it is also clear that the amount of data generated from similar length procedures is sufficient for training of classifiers although, as more data is generally a good thing, more supplementary data was gathered where possible.

Figure 6.2 shows how the electrodes were arranged on the forearm so that the Delsys system could target muscles without getting in the way of the Myo Armband. The Delsys electrode positions were selected to target both specific muscles and general areas of interest. Electrode E1 is placed just behind the wrist along the Abductor Pollicis Longus muscle. E2 is placed similarly behind the wrist along the Flexor Digitorum Superficialis. E3 again is set behind the wrist along the Extensor Carpi Ulnaris. E4 is placed further up the forearm, but in front of the Myo Armband along the Flexor Carpi Radialis. Lastly, electrode E5 is placed inline with E4 on the forearm but along the Flexor Carpi Ulnaris.

The set of gestures used was selected from a large pool of candidates based on the hand taxonomy/robotics literature [223–226] and easily recognisable gestures such as the "thumbs up". Selection criteria were based on preliminary trials conducted on subjects 1, 2 and 10. These trials involved gathering data on each of these different gestures and quantitatively comparing the performance achieved by the Neural Network and Support Vector Machine classifiers on offline data with various combinations of these gestures, as well as a qualitative comparison of the classification potential in an online context using the Myo Armband. The set of gestures selected is shown in Figure 6.3.

6.2.2 Experiment Protocols

The experiment was divided into three distinct steps:

- Consent and clinical data gathering
- Subject training
- Gesture data gathering

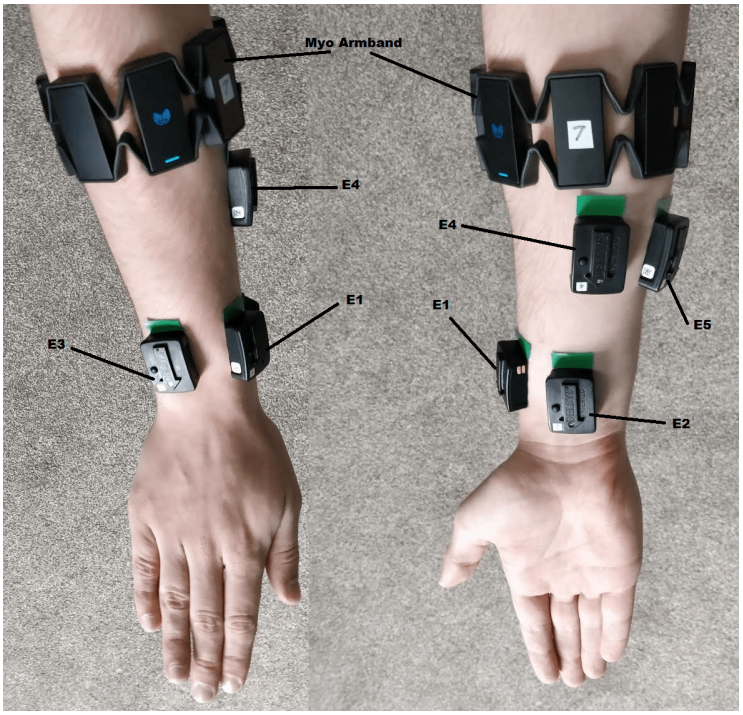

Figure 6.2: Position of electrodes on the forearm. Electrodes denoted "E" are the Delsys system blocks while the Myo Armband is shown separately. Subject 1 is used as a demonstration model.

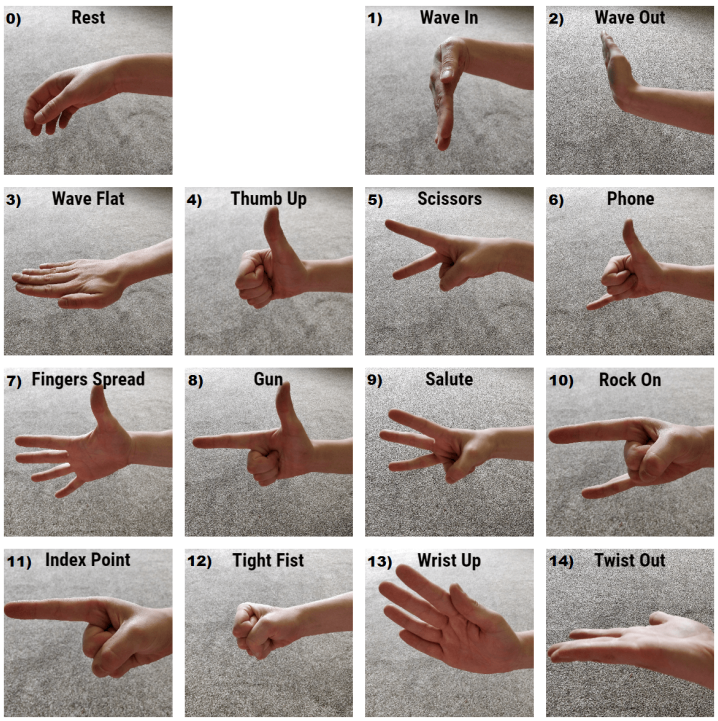

Figure 6.3: The 14 gestures (+ *rest*) included in the study and their associated labels.

Consent and clinical data was taken via a form presented to the subject before the beginning of the experiment. The subject was asked to fill out a form that ensured they understood how the experiment would proceed and how their data would be handled. The form also recorded their name, height, weight, age, gender, handedness and the date of the experiment. In order to anonymise the data, a unique number was recorded for each subject then the name was removed before the rest of the clinical data and unique ID was transcribed to Javascript Object Notation (JSON) format to be included with the dataset. This allowed a subject's data to be identified later via the unique number in the case they wished to withdraw their data from the study.

The training step was performed to ensure that each subject was familiar with the gestures and that each performed the gestures in as similar a way as possible. Each subject was provided with a copy of Figure 6.3 for reference. The Figure shows each movement used in the experiment as well as the order they appeared. The experimental supervisor then ran through each gesture, in turn, instructing the subject in how it should be performed. The subject was shown a demonstration of how the testing software would act and asked to demonstrate each gesture to ensure they could reliably replicate them. This helped prevent errors during the data gathering step.

Gesture data gathering began with affixing the two sets of electrodes to the subject's dominant hand as per Figure 6.2 so that data can be captured on both sets simultaneously. The Myo Armband was slipped on first with no special attachments, and then the Delsys electrodes were placed using the adhesive patches recommended by the manufacturer.

For the Myo Armband, the subject was asked to make a "Thumb Up" gesture with the forearm parallel to the ground and thumb pointing directly upward. The Myo Armband was then placed $\frac{2}{3}$ of the way up the forearm (measured from the lower electrode edge) with the main electrode block directly on top, status LED closest to the wrist, band perpendicular to forearm. In the case where $\frac{2}{3}$ was too low to keep the Myo on it was moved up until it was secure.

Once the electrodes were secured, the testing software was run which guided the subject through all 6 repetitions of each of the 15 movements. Each gesture (including explicitly *rest* as a separate gesture) was held for 10 seconds with a 3-second pause to a neutral position (not necessarily *rest*) afterwards. The gestures were performed in the order shown in Figure 6.3; then the process was repeated 6 times.

For each gesture the detailed acquisition process was:

- The subject has their hand at *rest*/neutral

- A prompt was displayed instructing subject to assume a gesture
- 3s delay (data not labelled as belonging to movement)
- 10s capture (this data was labelled as belonging to the gesture)
- Prompt displayed signalling to return to *rest*/neutral
- 3s delay
- Repeat for next gesture in sequence

A stationary hold of each gesture was chosen over a movement into and out of a gesture to avoid the issue of inaccurate ground truth caused by movement [35]. When a subject performs a dynamic movement, as opposed to a hold, the classification algorithm must account for the movement into, hold of the gesture and movement out of the gesture back to *rest* or into another movement. During the movement into and out of a gesture hold, the class is ill-defined since the hand is in a dynamic transition between distinct classes and may contain elements associated with either, as well as additional artefacts caused by making the transition. Therefore these transitory periods are ill-suited to classification and thus labelled in this study as belonging to none of the classes of interest. This philosophy is extended to the neutral posture that subjects are asked to adopt between other gestures, labelling it not as *rest* but as "not of interest". This is done because the exact start and end points of the posture cannot be guaranteed, and testing found that subjects often moved during these periods, which could make the label boundaries erroneous.

Labelling over a hold allows strong delimitation of the gesture boundary. The 3-second delay after prompting the subject to assume the gesture was found to be sufficient to ensure that the subject was already holding the gesture when the labelling began. This allows for a high degree of confidence that the data is representative of the gesture. Similarly, since the data stops being labelled as a gesture at the same time as the prompt appears, the end boundary can also be assigned a high degree of confidence.

These methods contrast to most other approaches such as the NinaPro databases, where transitory movement is labelled the same as the held position for a particular gesture. An offline algorithm (the GLR) is then required to estimate the position of the boundary between a gesture and *rest*. This produces uncertainty in the label at the boundaries as well as introducing the assumption that all data between gestures is *rest*.

Figure 5.6 shows the classification performance from the previous chapter as a function of the normalised position within a movement. This sort of performance

curve is the expected result given transitory movements which involve moving from *rest* to a gesture and then back over a labelled period. Therefore the hypothesis is that by labelling a hold, a much flatter performance curve would be observed leading to sharper classification at the boundaries, reducing misclassification rates and providing more consistent performance.

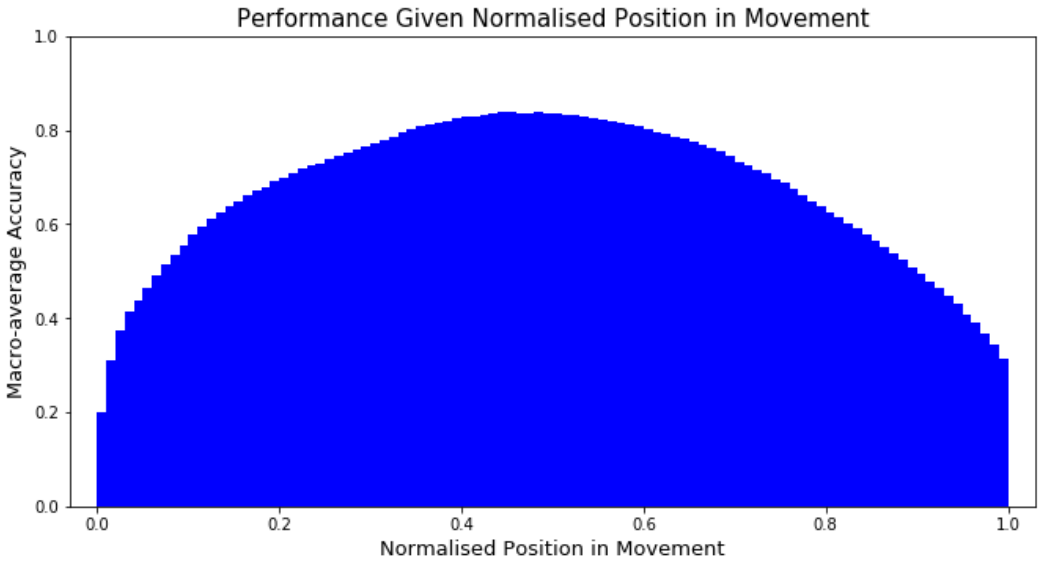

Figure 6.4: Performance of the TtS network given the normalised position within movement.

The treating of *rest* as a separate gesture, rather than the class found between other gestures of interest, helps to ensure consistency since it receives the same delimitation as other gestures. This reduces the likelihood of errant movement during capture or residual movement from previous gestures. An additional benefit is that, since this methodology means all gestures are treated equally, the data does not end up imbalanced.

6.2.3 Experiment Software

A Python script was developed that handles the experiment proper as well as the necessary post-experiment data synchronisation via high-resolution timestamps. Data was acquired from the Delsys electrodes using proprietary software and exported manually for synchronisation; however, for the Myo Armband, a set of tools were developed to facilitate this and future studies.

The toolset for the Myo includes data gathering utilities to capture data from all the sensors on board the device as well as bindings to connect to MATLAB and Python in real time. High-resolution timestamps are also provided that can be

used to synchronise with other sources. This software has been made available to the community and successfully used on numerous other research projects [52].

6.2.4 Experiment Extensions

After the initial study, an extension to the experiment was designed that integrated video into the data gathering so that movement labels could be labelled by a human expert. This was particularly useful for comparing the performance of algorithms such as the GLR and hold delimitation, as used here, to what a human would classify as the start and end of each movement. Video data also makes identification of deviations in the experiment possible for the purposes of detecting unreliable data.

This extended experiment was performed twice on subject 1 and will be made available as supplemental data. An additional tool was created that allowed expert labelling and resynchronisation with the data streams.

In order to test for differences in placement additional supplemental data was also collected on subjects 1 and 2 which was the same as the original experiment except with 8 Delsys electrodes placed in the same position that the Myo Armband electrodes normally occupy.

6.2.5 Electrode Comparison

The two electrode sets used were the Myo Armband [78] and the Delsys Trigno Wireless System [93]. A quantitative and qualitative comparison of the devices is given below.

Cost, the Myo Armband retails for roughly £150 at the time of writing while the Delsys Trigno Wireless system is sold for ∼£15,000 (for 8 electrodes, available only via quotation) [93]. Notably, the Trigno System also requires expensive additions for continued use including specially designed single-use sticky fixers for placement of electrodes although it would be possible to use alternate means of attaching the electrodes.

Sampling Frequency, the Delsys electrodes are rated for a 2kHz sample rate, although, in practice, a sample rate of ∼1930Hz was observed. The Myo Armband has a sampling frequency 200Hz. This is one of the major differences between the devices.

Number of Electrodes, the Myo Armband is made up of 8 electrodes fixed into a band which cannot be split up. It is possible to use multiple Myos simultaneously. The Delsys Trigno system uses electrodes which are self-contained and affixed to a subject using specially designed sticky fixers. Using a single base station, 16 of

such units may be used simultaneously. Additional base stations allow more units to be used.

Electrode Resolution, the Myo quantises incoming sEMG data to 8 bits while the Delsys Trigno uses a 16-bit conversion giving it a much higher resolution.

Electrode Quality comparison is difficult as the technical specification for the Myo electrodes is not available, although it is likely a reasonable assumption that the Trigno electrodes are of a higher quality.

Filtering, the Trigno electrodes filter to 20Hz ($\pm$5Hz) - 450Hz ($\pm$50Hz) while the Myo Armband uses built-in notch filters at 50Hz and 60Hz although the raw data without this filter is available.

Ease of Use is an important, although highly subjective, aspect for the usage of any wearable device. For an everyday application such as control of a computer or robot, the Myo Armband is convenient as its arm-band allows quick, simple, repeatable placement of electrodes whereas the Trigno requires the use of new sticky fixtures each time and electrodes cannot be moved around easily. The fixed nature of the Myo Armband does, however, significantly hinder its ability to target individual muscles.

The Myo Armband also has the advantage of using Bluetooth as a communication protocol allowing for easy integration with mobile devices and applications whereas the Trigno requires a bulky base station and separate power source to communicate with its sensors.

6.2.6 Data Preprocessing

After each experimental run was concluded timestamps were used to synchronise labelling of gestures and repetitions within the data. Windowing was performed using a sliding window of 150ms with an increment of 5ms. This translates to window lengths of 30 samples and increments of 1 sample on the Myo data and window lengths of 300 samples and increments of 10 samples on the Delsys data. All the data is then standardised to $N(0, 1)$ prior to classification using statistics calculated from the training data.

In keeping with previous chapters, performance is measured using the macro-average accuracy and cross-validated using the balanced stratification approach. In this study, since there was no need for concessions to improve comparability, the data was divided into three sets: training, validation and testing using repetition number. The validation set was used for early stopping purposes during training as well as for exploratory design purposes. The cross-validation folds are shown in Table 6.1.

Split	Training	Validation	Testing
1	[1, 5, 6]	[4]	[2, 3]
2	[1, 3, 6]	[2]	[4, 5]
3	[1, 2, 4]	[3]	[5, 6]
4	[1, 2, 5]	[6]	[3, 4]
5	[2, 4, 5]	[1]	[3, 6]
6	[1, 3, 5]	[6]	[2, 4]
7	[3, 5, 6]	[4]	[1, 2]
8	[4, 5, 6]	[2]	[1, 3]
9	[2, 4, 6]	[3]	[1, 5]
10	[1, 2, 3]	[5]	[4, 6]
11	[3, 4, 6]	[1]	[2, 5]
12	[2, 3, 4]	[5]	[1, 6]

Table 6.1: Cross-validation folds for training, validation and test sets for this study. Numbers are the repetition label used in that fold; each repetition appears a total of 4 times in the test folds to ensure proper stratification.

6.2.7 Performance Baselines

In order to establish a performance baseline and compare the utility of the data sets, an SVM-RBF with mDWT as its feature was trained along with the adaptive variant of the TtS described in the previous chapter (Section 5.2.3). The SVM-RBF was chosen due to its top performance from the range of standard classification techniques in the previous benchmarks. The TtS represented the top performer overall and was used to determine what level of performance was possible from the data.

Early stopping was implemented during all neural network training in this chapter with stopping criteria designed to improve generalisation performance at the cost of potentially extending training time [227]. The idea is to keep training the model until the loss $\mathcal{L}$ (calculated on the validation set) does not improve by at least a minimum value δ_{min} for p training epochs and then return the model that produced the best loss value. The algorithm is illustrated in Algorithm 6.1.

In practice δ_{min} was set to 0.005 and the patience p was set to 5.

Support Vector Machines do not utilise early stopping; therefore, when training the SVM-RBF, the validation set is discarded to preserve the integrity of the other sets.

Algorithm 6.1 Early stopping criteria based on evaluation of techniques by Bottou [227]. Algorithm trades off extended training time for improvements to generalisation.

Input: p : Patience, δ_{min} : Minimum improvement
D_{val} : Validation data
Result: m_{final} : Trained model

m `// Initialised model`
$m_{final} = m$
$L_{prev} = inf\ i = 0$
while $i < p$ **do**
  `// Train network` m `for 1 epoch`
  $i = i + 1$
  $\mathcal{L} = m(D_{val})$ `// Evaluate` m `with validation data`
  if $\mathcal{L} < L_{prev}$ **then**
    $m_{final} = m$
  end
  if $\mathcal{L} + \delta_{min} < L_{prev}$ **then**
    $i = 0$
    $L_{prev} = \mathcal{L}$
  end
end

6.2.8 Compact Deep Neural Network

In addition to evaluating the TtS networked design in the previous chapter, the design was iterated on with a focus to creating a new, refined network architecture that reduced the total number of parameters for a minimum trade-off in performance. The motivation for this focus was the finding that running the networks on lower-cost systems or embedded devices led to delays of the order of tens of milliseconds which is a relatively long time when the ideal delay for control is $< 200ms$ [203].

Tables 6.2 and 6.3 show the Compact TtS designs for the Myo and Delsys data, respectively. It was determined that it was possible to use the sampling frequency adaptation approach described in the previous chapter to adapt for the two data sources and that several modifications could be made within the architecture to limit the number of parameters.

The first related change was the dramatic reduction in the number of filters used in the Spatial Reduction layer and limitation of its data view. This produced a significant bottleneck effect, similar to AutoEncoders [228], compressing the effective feature space while still retaining most of the important information. This allowed the penultimate dropout and dense layers to be removed from the ar-

Layer Type	Output Size	# Filters	Filter Size	Stride	Padding	Activation	# Parameters
Input EMG	30x8x1						
Guassian Noise	30x8x1					$\alpha = 0.001$	
Conv (Temporal)	10x8x16	16	3x1	3x1	Same	LReLU ($\alpha = 0.1$)	64
Temporal Fire	10x8x64	(16, 32, 32)	(1x1, 1x1, 3x1)	1x1	Same	LReLU ($\alpha = 0.1$)	2,384
Spatial Reduction	10x8x2	2	1x8	1x1	Same	LReLU ($\alpha = 0.1$)	1,026
Dropout	160					rate = 0.5	
Dense	15	15				Softmax	2,415
						Total	5,889

Table 6.2: Compact TtS architecture that dramatically reduces the number of parameters relative to the original. Myo data variant, channels last format.

Layer Type	Output Size	# Filters	Filter Size	Stride	Padding	Activation	# Parameters
Input EMG	300x5x1						
Guassian Noise	300x5x1					$\alpha = 0.001$	
Conv (Temporal)	12x5x16	16	50x1	25x1	Same	LReLU ($\alpha = 0.1$)	816
Temporal Fire	12x5x64	(16, 32, 32)	(1x1, 1x1, 3x1)	1x1	Same	LReLU ($\alpha = 0.1$)	2,384
Spatial Reduction	10x5x2	2	1x5	1x1	Same	LReLU ($\alpha = 0.1$)	642
Dropout	120					rate = 0.5	
Dense	15	15				Softmax	1,815
						Total	5,657

Table 6.3: Compact TtS architecture that dramatically reduces the number of parameters relative to the original. Delsys data variant, channels last format.

chitecture entirely since the representation was now simpler and less tied to the interaction between different filters. Unlike when a similar approach was tested on the NinaPro data, it was found that, in combination with the compression, removal of these layers had a minimal impact on performance. This was due to the reduction in the number of movements being classified and the reduction in the number of electrode channels which allowed simpler feature extraction to be effective.

The number of filters in each layer (except in the Spatial Reduction and Classification layers) was then adjusted via coarse grid search. The search was limited to powers of 2 as a way of reducing the search space as well as to minimise the chance of overfitting to the particular data under evaluation.

6.2.9 Hardware Performance Comparison

The Jetson TX2 [103] was used to evaluate relative run-time performance; the Jetson was chosen because it is an embedded device that incorporates graphics processing hardware. The timing was performed using Python's "timeit" package reporting the lowest value from 20 trials, each of which took the mean run-time of 1000 predictions with the model pre-loaded into memory. This produced a soft lower bound on the computation time.

Since there is a large number of embedded device and software implementation pairs, the results of this part of the study are intended to demonstrate,

conceptually, the point that a more compact network does provide a tangible improvement in classification time as opposed to producing hard numbers on what that performance improvement may be for any specific application.

6.3 Results and Discussion

6.3.1 Key Findings

On the Myo data the TtS network achieved a mean macro-average accuracy of 85.1%, the variance between subjects was 6.2%. The SVM-RBF achieved a mean macro-average accuracy of 70.5% with slightly higher variance between subjects of 6.9%. The Wilcoxon signed-rank test was used to confirm the significance of this performance difference.

The Wilcoxon signed-rank test was performed using the 10 subjects as 10 matched samples to compared the TtS network and SVM-RBF. The TtS network outperformed the SVM-RBF on each subject, which leads to the conclusion that the TtS network significantly outperforms the SVM-RBF at $p < 5.1 \times 10^{-3}$. The minimum performance improvement on any specific subject was 9.6%.

On the Delsys data, the TtS network achieved a mean macro-average accuracy of 82.7%, with a variance between subjects of 7.1%. The SVM-RBF achieved a mean macro-average accuracy of 67.9% with, again, the relatively higher variance between subjects of 9.5%.

The Wilcoxon signed-rank test was used on this data as well. Similarly, the TtS network outperformed the SVM-RBF on each subject, which lead to the same conclusion; the TtS network significantly outperforms the SVM-RBF at $p < 5.1 \times 10^{-3}$. The minimum performance improvement on any specific subject was 8.7%.

The performance trend between the TtS network and SVM-RBF remains the same as in previous benchmarks with the TtS network significantly outperforming the SVM-RBF. As seen in Chapter 4 it may be possible to improve the SVM-RBF's performance via the use of SMOTE; however, this would still not bridge the performance difference. Therefore the remainder of this analysis focuses on the comparison of the neural network performances.

Figure 6.5 shows the mean performance per class for the TtS network. It can be seen that while the Myo data generally leads to higher performance, the improvement is not uniform with different gesture classes performing better on different data sets. In order to investigate whether this was a direct result of the electrode placement, a supplementary data set was obtained on subjects 1 and 2 using 8 Delsys electrodes in the positions the Myo Armband electrodes would occupy.

Figures 6.7 and 6.8 show the results with the supplementary data, while Figure 6.6 shows the original study broken down per subject for reference.

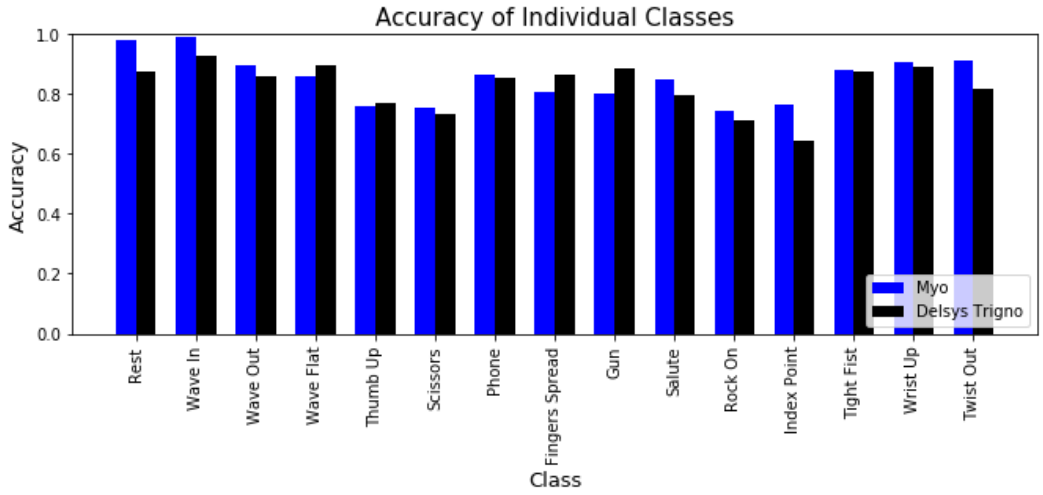

Figure 6.5: Performance of the TtS network on the Myo and Delsys data broken down per class (mean taken across subjects).

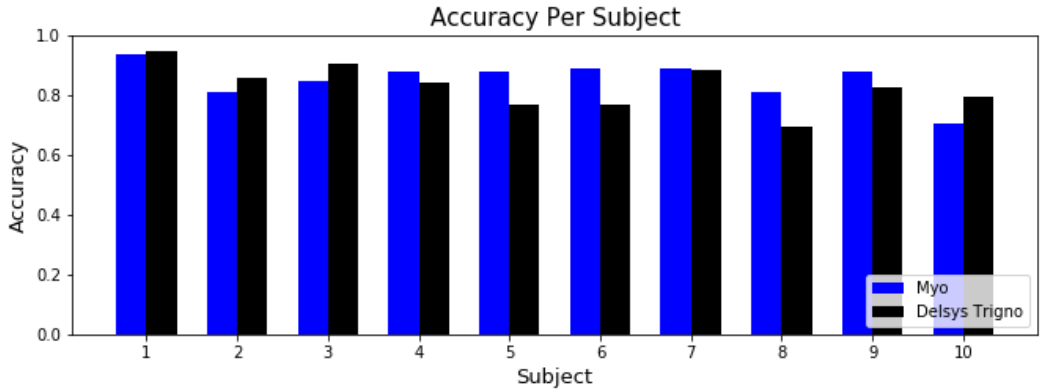

Figure 6.6: Performance of the TtS network on the Myo and Delsys data broken down per subject.

Overall subject 1 saw a drop in the performance of the Delsys system when going from the original setup to the supplementary set. Specifically 94.6% to 89.2% while subject 2 saw a minor increase in performance from 85.8% to 86.5%. Both subjects originally showed better performance on the Delsys electrodes. This implies that, rather than electrode placement being the sole factor, it is necessary to consider placement in the context of each subject's physiology, and its interaction with the electrodes and gestures under investigation. Further, as seen in Figure 6.6, there is no clear-cut better setup lending further credence to the theory. This idea, that for real-world usage, greater weight needs to be lent to the personalisation

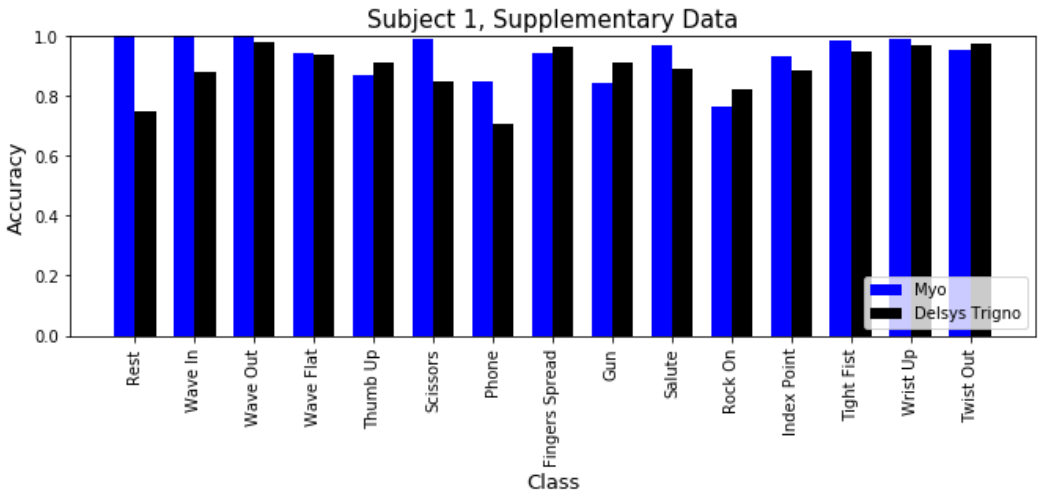

Figure 6.7: Performance of the TtS network on the Myo data and supplementary Delsys data with electrodes placed in the same position as the Myo Armband. Subject 1.

of devices and control mechanisms has also been a key discussion recently in the Bio-medical community [229, 230].

Subject-wise performance indicates that subject 1 is classified better than all other subjects on all data sets. This is likely due to the subject's familiarity with the experimental procedure as well as the experimental design. This makes the subject a useful test case rather than an anomaly since they represent a trained operator for whom the setup has some personalisation, therefore, making them an exemplar closer to the upper bound of the possible performance.

The class-wise performance breakdown demonstrates variability on two fronts: between classes and between data sets. The variation between classes is expected given the usage of different muscle combinations, but the difference between best and worst performers indicates that it may be possible to sub-select from a pool of gestures for a given application in order to trade off number of gestures with performance. This is explored in Chapter 7.

The data set differences in class performance do not show a clear trend towards certain types of gestures which is reinforced by the supplementary data differences and the idea that the evidence for the wider context is an important factor. Importantly, however, both in the per class and both supplementary results, the Delsys system performs worse than the Myo Armband at classifying the *rest* class. It is not apparent why this is the case but, by elimination, it can be inferred that the cause is some quality of the Delsys electrodes or their interaction with other system elements. Potential causes include the higher sensitivity of the electrodes and artefacts caused by the increased input resolution. However,

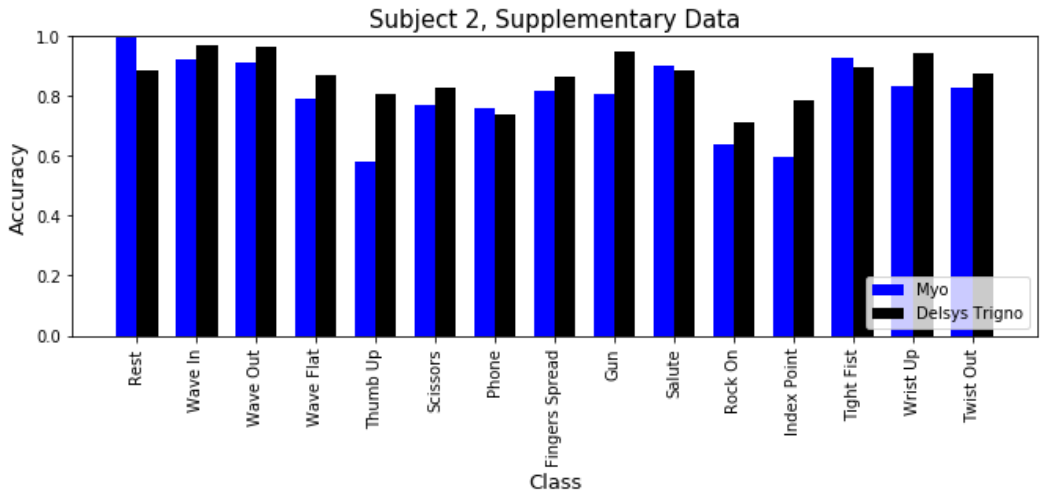

Figure 6.8: Performance of the TtS network on the Myo data and supplementary Delsys data with electrodes placed in the same position as the Myo Armband. Subject 2.

determining the true cause would require further investigation.

Regardless of the cause of the Delsys system's lower performance on the *rest* class, it is interesting that the Myo performs consistently well on it, achieving over 98% accuracy on the majority of subjects (see Figure 6.5). This shows that, in this setup with the Myo Armband, it is unnecessary to bias performance metrics, weight classification or to acquire extra data to ensure high performance on the *rest* class. This is a useful result for the many use-cases that desire consistent performance on the *rest* class.

Overall this study shows that the Myo Armband is competitive with the Delsys system, while costing a factor of ~ 100 less, making it of particular interest to use cases where cost is a factor. Therefore for gesture sets of this size or smaller, the Myo Armband is a viable, low-cost alternative to medical grade systems. Recently, similar promising results using the Myo Armband have been demonstrated on transradial amputees further demonstrating its potential [231].

6.3.2 Performance on Hardware

The Compact version of the TtS network was trialled on the Jetson TX2 and a modern graphics card (NVIDIA GTX 1080 Ti) to demonstrate the utility of parameter reduction in a real-world context. The results are shown in Table 6.4 which includes a comparison with the networks designed by Atzori et al. [47] and Geng et al. [39] covered in the previous chapter with Atzori et al.'s [47] strides updated to allow evaluation on the new data shape.

The Compact TtS achieves a lower macro-average accuracy than the standard

Myo Data				
	Params	Acc.	1080 Ti	TX2
Compact TtS	5,889	84.2%	1.68ms	7.89ms
Atzori et al. [47]	97,883	81.7%	1.69ms	13.17ms
Geng et al. [39]	644,435	44.1%	3.19ms	22.26ms

Delsys Data				
	Params	Acc.	1080 Ti	TX2
Compact TtS	5,657	80.3%	1.74ms	8.07ms
Atzori et al. [47]	99,308	65.4%	1.66ms	15.36ms
Geng et al. [39]	546,131	26.4%	3.21ms	20.14ms

Table 6.4: Comparison of the number of parameters, cross-subject mean macro accuracy and run-times for different neural networks on the Jetson TX2 and NVIDIA 1080 Ti.

TtS network on both data sets, which is the main trade-off for the reduction in the total number of parameters. The ratio of performance reduction to parameter reduction is excellent; however, as the number of parameters was reduced by a factor of 100 for $< 3\%$ performance drops on both data sets. The Compact TtS also still outperforms the contemporary networks tested while using many fewer parameters [39, 47]. The other networks follow the same performance trend, as observed in the previous chapter.

Since the data shape is different in this data than for their original design, the network by Atzori et al. [47] required alteration to its strides to be trained on this data. Therefore the performance result is not for an identical network. However, a brute force search of a pool of potential stride changes was trialled and the best performance design used, which led to the performance reported in Table 6.4.

The run-time results demonstrate the stochasticity present in non-real-time operating systems. However, this still demonstrates that parameter reduction has tangible benefits on low-power hardware such as the Jetson. On the higher performance computer, utilising the 1080Ti, the benefit is more marginal since other factors dominate the run-time below some number of parameters, although there is still an improvement over the much larger network used by Geng *et al*.

The overall run-time reduction of the Compact TtS is at least 40% (see Table 6.4) compared to the other networks which makes the network a useful design for many applications where response time to input is critical, such as device control or prosthetic application.

6.3.3 GLR vs Hold vs Expert Labelling

The experiment also provided an opportunity to qualitatively evaluate the differences between the methods for labelling of where gestures begin and end. Figure

6.9 shows supplementary data gathered on subject 1 and labelled expertly through the use of accompanying video data. The expert label denoted where the subject reached a hold position and the frame before they started moving out of it. The video was synchronised with the original data via timestamps attached to both the video and sEMG data frames taken from the same source.

Figure 6.9 shows that the GLR captures a longer duration, including the movement into and out of each gesture, however it fails to delimit only the held gesture. The expert labelling clips a small amount off the start and end compared to the GLR, avoiding transitory movements which was desirable for this study. Compared to the hold markers from the experiment, the expert labelling leads to more usable data per subject but may also include more edge effects as seen at the start of the gesture example in Figure 6.9.

Ideally, expert labelling would be used on all subjects; however, the process is time intensive and so does not scale well to larger studies. It may also not be feasible in many applications, e.g. commercial or medical devices where the human resources to label the data are either expensive, unavailable, or slow relative to the desired time frame for device usage. The GLR trades the human resources issue for increased computational cost since an exhaustive search is necessary. Even with aggressive limiting of the search space, the computational overhead can be high, which may be limiting although generally less of an issue compared to the needs of expert labelling. Hold-based labelling requires no overhead and therefore is attractive as a labelling option. However, it was necessary to explore whether it was sufficiently representative as despite being used in the literature [34, 35], no study had proven its utility in a machine learning context.

The TtS network was retrained and tested with the same cross-validation procedure on this supplementary data with different labelling strategies to evaluate the resultant differences. Table 6.5 compiles the results.

		Test		
		Expert	Hold	GLR
Train	Expert	90.8%	91.1%	86.3%
	Hold	87.7%	90.3%	83.2%
	GLR	89.2%	88.6%	85.7%

Table 6.5: Comparison of performance for the different labelling procedures applied to the training and test sets for the supplementary experiment on subject 1.

The results lend credence to the study premise that using "hold" labels is an effective proxy for expected performance as can be seen by the fact that the "hold" and "expert" diagonal entries are similar. Intuitively this makes sense because both attempt to capture the held gesture although the expert labels capture a

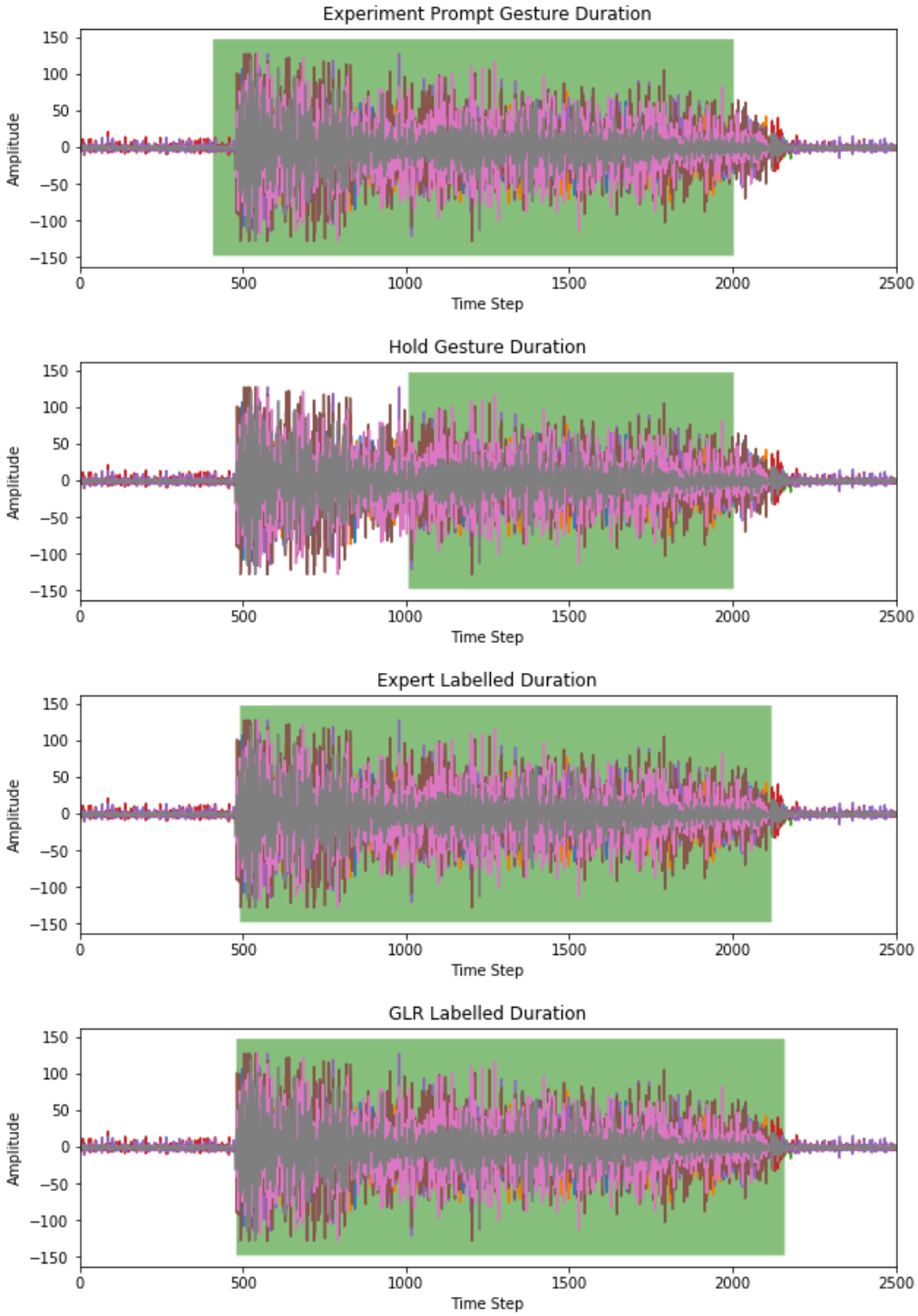

Figure 6.9: Comparison of movement duration labelled via different techniques showing GLR capturing a longer time period than the expert labelled hold position.

more complete picture. This is evidenced in the "expert"-trained, "hold"-tested entry which performs slightly better than all other entries and the fact the "hold"-trained, "expert"-tested entry leads to lower performance.

If it is assumed that the expert labelled tests are closest to expected real-world conditions, the first column indicates that, as would be expected, expert labels would be ideal followed by GLR and then "hold" for actually training a useful classifier.

The GLR trained and tested TtS network performed worse than the other diagonal results. This is caused by its capture of the transitory movements into and out of each gesture compared to the other labelling methods. Since these periods are relatively short compared to the held gesture duration, they have a limited impact on the classifier itself, but since the transitions present different signals to the held gestures, this reduces overall test-time performance. This is evidenced by the fact that the "expert"-trained, "GLR"-tested network performs slightly better than with the training and test labels swapped.

Given the above, it is reasonable to assume that for the sake of performance it will often be desirable to assume transitory movements should not be classified and to instead classify only when a subject actually holds a gesture. This is especially true when it cannot be assumed a subject will always start from *rest* or a neutral hand position, which will further enhance the differences between held gestures and the movements into and out of them. Under this assumption, it is clear that it would be ideal to have an expert label all data in future experiments; however, as previously noted, this adds extra human resources overhead. If this is not possible, the GLR has been found to be a reasonable substitute for training purposes although it is likely to skew test-time results downwards in the absence of expertly labelled data that better reflects the intended usage to test on.

The "hold" labels perform worst but give a good indicator of classifier performance potential, and their lack of overhead may be an advantage in some situations, particularly if it is possible to recoup performance via other methods.

Since these results are based on the data from a single subject they should be regarded as preliminary, however, they represent an exciting set of options for future experiments that have previously not been explored in the literature.

6.4 Conclusion

A new comparative sEMG classification study was detailed in this chapter. It found that data from the Myo Armband was capable of producing similar or better levels of classification performance than the Delsys Trigno Wireless System.

This is an important finding since the Myo Armband is a factor of ~ 100 cheaper than the Delsys Trigno as well as being considerably easier to set up and use. This potentially makes the barrier to entry for sEMG based gesture classification much lower. Recent work has also echoed the finding that the Myo Armband has high classification potential in transradial amputees [231], further demonstrating that it is a viable alternative to other systems.

The TtS network, introduced in the last chapter, was also streamlined here to form a new design: the Compact TtS, which uses many fewer parameters than the original variants. It is shown that the Compact TtS achieves slightly lower performance than the TtS however it can be used to predict much faster on real hardware. The Jetson TX2 [103] was used as an example of a low processing power environment, and the Compact TtS's forward pass runtime was evaluated relative to other contemporary networks demonstrating that it could predict gestures much faster [51].

Finally, labelling methods were reviewed, and it was shown that, while "hold" labels produce a reasonable performance estimate, current labelling methods potentially underestimate overall performance or lead to classifiers that underperform in practice due to differences from what an expert would label as the ground truth. Of particular interest is that the GLR labelling method can be used to train a reasonably good classifier but underestimates performance overall in the absence of expertly labelled data to evaluate on. Therefore, for future studies, it is shown that expert labelling is necessary to get the most accurate performance picture and that there is still more research to be done in the area of data labelling.

Chapter 7

Online Classification

7.1 Introduction

Previous chapters have focused on determining what level of performance is possible when classifying hand gestures from sEMG data. Throughout, additional improvements have been made and constraints added to move the problem away from offline classification towards the sort of solutions that could be practically deployed in real-world situations. This chapter brings together those improvements and adds several more to present an end-to-end solution for classifying sEMG in an online context.

Chapter 4 set up the fundamentals by codifying how to evaluate the problem; that the minimum number of assumptions on the test-time class distribution must be used to avoid biases caused by data imbalance. The macro-average accuracy was used to enforce this explicitly in the performance metric. It also introduced a robust method for stratified cross-validation so that results from offline analysis would be representative of expected performance, and a baseline was designed with feature-based classification algorithms to give an expected performance floor. Finally, Chapter 4 showed how, if an application has leeway in the possible gestures to be classified, then additional gestures can be tested during data gathering to enhance final performance by tailoring the gestures used to each individual.

Chapter 5 raised the performance potential by introducing a novel neural network architecture that improved overall performance. It also removed the zero-phase filtering in the preprocessing to better represent real applications, since zero-phase filters are not possible online. The analysis also indicated a consistent negative performance bias when testing on the first repetition of each movement performed by all subjects which presents a compelling edge case that may warrant additional investigation or removal from testing depending on the application.

Chapter 6 introduced a new study on a myoelectric control device to act as a testbed for further enhancements and demonstrated that the neural networks allowed a set of 15 gestures to be classified with over 90% macro-average accuracy on a trained individual. Further, a new compact variant of the network was introduced that significantly reduced the number of parameters necessary for a similar level of performance. This reduced the classification time on an embedded system by $> 30\%$ and thus demonstrated that it would be possible to run this mode of classification even in contexts with limited computation.

This chapter tackles the online problem directly by combining all these previous enhancements and building upon them. First, support methods were explored to determine whether data from other subjects could be used to improve performance without adding complexity. Specifically, the concept of subject adaptation was investigated which seeks to use data from a corpus of other subjects to improve the performance of a specific subject by integrating the corpus into the training process. This process is known as transfer learning and pushed performance higher without the need for extra data gathering. These performance improvements, via subject adaptation methods, have also been corroborated by recent work by Côté-Allard *et al.* [28].

Second, in the online case, despite high gesture accuracy, jitter/instability is seen in the classification signal; the classification output often flickers between multiple gestures despite a subject holding a particular gesture. Multiple methods for achieving smooth classification outputs are presented and evaluated finding that most algorithms tested lie on a Pareto frontier in terms of smoothness and latency trade-offs, so a nuanced evaluation depending on the precise context is necessary to select the best for an application. However, it is also noted that an elegant majority voting algorithm can serve as a good choice, improving the smoothness of the classification signal for minimal added latency.

Both investigations make use of the Myo data from the previous chapter.

7.2 Methods

7.2.1 Improving Performance with Transfer Learning

Most applications of hand gesture recognition benefit from reducing the amount of training needed to get usable levels of accuracy since training forms a barrier to normal operation. Improvements to the overall performance without the need for additional data gathering are also useful since classification accuracy has room to improve on all classes. Therefore a sequence of experiments was designed to examine how to utilise data recorded from other subjects to either improve overall

performance or reduce the data needed from a single subject to produce a given level of performance.

The ideal situation would be to require no data on a new subject and still classify with high performance. One way to achieve this is to train a network on other subjects and test it on a new subject. The fact that this sort of design has not been explored in the literature implies its difficulty; however, here an implementation is included as a demonstration of the lower performance band on the problem to inform future researchers.

The implementation consisted of an updated TtS network (that kept the same performance as the original but with reduced parameters) tested on one target subject while having been trained on all other subjects (the source subjects) in a leave-one-out fashion combined with the evaluation procedures outlined in earlier chapters. The updated network is shown in Table 7.1. Cross-validation folds for this implementation and the other transfer learning experiments are shown in Table 7.2.

Layer Type	Output Size	# Filters	Filter Size	Stride	Padding	Activation	# Parameters
Input EMG	30x8x1						
Conv (Temporal)	10x8x16	16	3x1	3x1	Same	LReLU ($\alpha = 0.1$)	64
Temporal Fire	10x8x64	(16, 32, 32)	(1x1, 1x1, 3x1)	1x1	Same	LReLU ($\alpha = 0.1$)	2,384
Conv (Spatial)	10x8x16	16	3x12	1x1	Same	LReLU ($\alpha = 0.1$)	8,208
Dropout	1280					rate = 0.5	
Dense	15	15				Softmax	19,215
						Total	29,871

Table 7.1: Updated TtS that achieves performance the same as the original while incorporating parameter reducing improvements from Chapter 6. Channels last format.

Split	Training	Validation	Testing
1	[1, 4, 6]	[5]	[2, 3]
2	[1, 3, 6]	[2]	[4, 5]
3	[1, 2, 4]	[3]	[5, 6]
4	[2, 5, 6]	[1]	[3, 4]
5	[3, 4, 5]	[6]	[1, 2]
6	[2, 3, 5]	[4]	[1, 6]

Table 7.2: Cross-validation folds for training, validation and test sets for this study. Numbers are the repetition label used in that fold; each repetition appears a total of 2 times in the test folds to ensure proper stratification.

Two fine-tuning methods were also explored. These involved pretraining the TtS network on the source subject then freezing some of the layers and training on the target subject. This is similar to how many computer vision problems utilise

image-net trained networks [232], freeze the lower layers and retrain them for a more niche task. The two variants differed in which layers were frozen; the first froze all but the final dense layer essentially forcing identical feature extraction while allowing the classification to be tailored to each individual. The second was the opposite, freezing only the final dense layer to provide a contrast. This is not normally done in other fields since the feature representation training is generally considered the problematic part [232].

An augmented TtS network was also introduced, which allowed simultaneous training on source and target. This was achieved by creating two TtS networks, one for the source subjects and one for the target subject that shared weights in the Temporal Fire module and Convolution (Spatial) layers (see Table 7.1). This creates a shared latent space with the aim being to get the benefits of feature extractor pretraining without sacrificing the ability to adapt to the quirks of a specific subject. This was the best performing design of a multitude that were built on a similar theme but sharing different sections. The training was performed using an equal amount of data from the target subject and source subjects, which was presented to the network simultaneously.

In summary, the following setups were fully tested:

- TtS trained with no data on the target subject
- TtS pretrained on source subjects then trained for the target subjects with feature layers frozen
- TtS pretrained on source subjects then trained for the target subjects with classification layers frozen
- An augmented TtS design that simultaneously trains on the source subjects and target subject
- Variants of the previous three using only partial data on each target subject

When evaluating with partial data, each cross-validation fold was split into 3 sub-folds where the classifier was trained on only a single training repetition of the three in the fold.

Several other problem definitions were explored, including training with unlabelled target data or target data being labelled using different methods to source data. However, these did not lead to any interesting results. Similarly, a variety of other architectures were explored based on interesting results from style transfer [233], language translation learning [234] and other domain adaptation task

[235, 236]. None of these other designs, however, produced consistently high performance on this problem so were omitted from testing, the implications of this, however, are discussed further in Section 7.3.

7.2.2 Motivation for Smoothing Predictions

All the classifiers explored thus far are instantaneous, i.e. they take in a single window of sEMG and produce a classification label with no memory capacity to augment the prediction based on previous inputs. Recurrent neural networks were trialled in various configurations; however, no design achieved results of a similar performance to the convolutional neural networks discussed in previous chapters.

An issue of instantaneous classification is that, even with a high macro-average accuracy, when the classification is treated as a signal the signal can be unsteady manifesting as a flicker between different output labels at successive time steps. Figure 7.1 illustrates this on the test repetitions for a single fold on the expertly labelled supplemental data from subject 1.

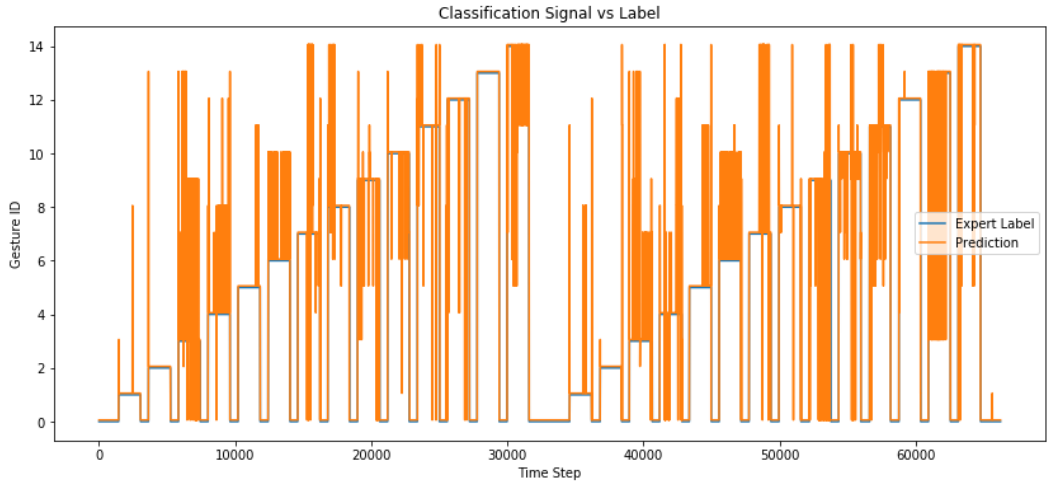

Figure 7.1: Demonstration of the flicker of the classification signal at test time. While overall accuracy remains high, the graph shows that the classification signal flickers between gestures rather than producing a consistent output which is likely to cause issues if used directly for many applications. The data used was the expertly labelled supplemental data on Subject 1. Note that the *rest* and neutral/unclassified labels have both been labelled 0 since the classifier is not trained to output an unclassified label.

The rapid flickering between one class and another manifests on the graph as solid blocks, although the amplitude thereof has no particular meaning since the y-axis is categorical. While not always present for each gesture it is clear that the flicker is a consistent issue and is likely to negatively affect application

performance if not corrected because a potentially large amount of unintended inputs are produced.

Therefore to improve the quality of the classification signal and to increase the utility of classifiers, some form of smoothing is required. Several solutions are possible and were tested to determine their utility; these are outlined below. Each algorithm used several windows worth of data and therefore acted as a super-windowing technique.

Due to the macro-average accuracy not being sufficient to capture the complete picture, three additional metrics were used to compare performance: onset latency, tail latency and mean deviations.

Onset latency was defined for all non-*rest* gestures as the time from the first ground truth label of the gesture to the first correct classification. Similarly, tail latency was defined as the time from the first ground truth label of *rest* after each gesture to the first correct classification of same. This is illustrated in Figure 7.2 and presented with and without preemptive results, i.e. without 0 latency results included since these are highly likely to be artefacts.

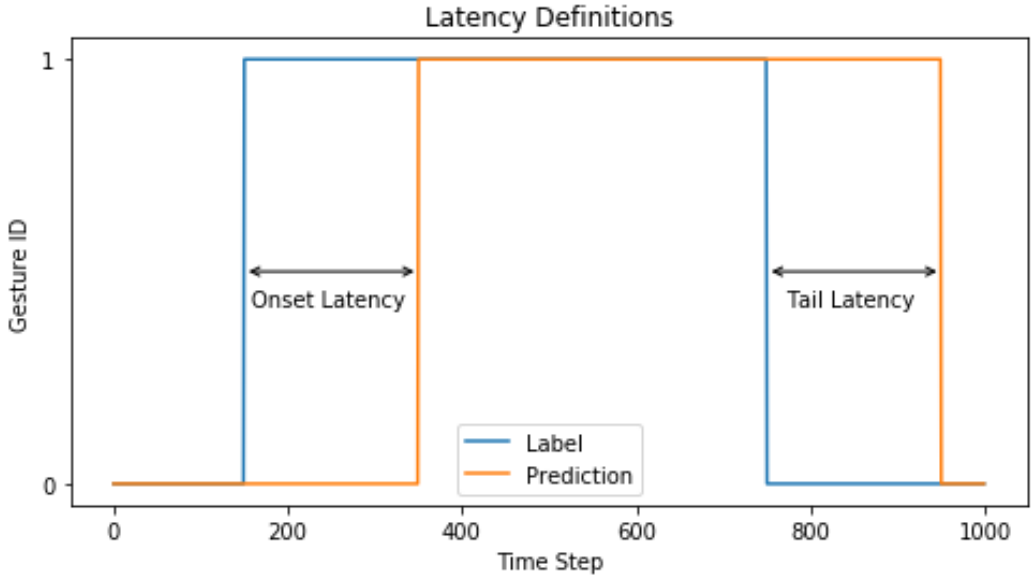

Figure 7.2: Onset and tail latency definitions.

The final additional measure was the Mean Deviations algorithm, which calculates the mean number of deviations per gesture. Deviations were measured as changes from a correct label to an incorrect one to contrast with the macro-average accuracy and latency metrics by representing an expected flicker rate for a given gesture irrespective of latencies or time spent misclassified. The algorithm is detailed in Algorithm 7.1.

Algorithm 7.1 Mean Deviations algorithm that calculates the mean number of deviations from the true label per gesture.

Input: Y : Ground truth labels, $\hat{Y}$: Predicted labels
Result: N_d : Mean deviations per gesture

$count = 0$
for *each non-rest gesture* $i \in 1 : N$ **do**
  // N gestures in data being evaluated
  for *each data point* $j \in 1 : T_i - 1$ **do**
    // T_i time steps in current gesture
    if $\hat{y}_j == y_j$ *and* $\hat{y}_{j+1} != \hat{y}_j$ **then**
      $count = count + 1$
    end
  end
end
$N_d = \frac{count}{N}$

The point of the mean deviations algorithm is to capture the relative smoothness of the classification signal in a way that taking accuracy measures does not.

The smoothing algorithms were evaluated based on the supplementary data with expert labelling as well as the main study data. In order to correctly test the smoothing algorithms, it was necessary to test on contiguous data rather than sliced data as previously used. Therefore when splitting by repetition, two seconds before the start of the first gesture with that repetition until two seconds after the final one was used as a single block. This necessitated precise labelling that correlated as closely as possible to the subject's actual movements, i.e. GLR or expert labelled data rather than hold labelled. It also necessitated a combination of the explicit *rest* class with the unclassified portions of the experiment to make a neutral posture class in order to make the transitions meaningful because the classifier does not output an "unclassified" class and augmenting the classifier with an "unclassified" class caused significantly degraded performance.

Therefore the GLR algorithm was run on each subject in the study to produce usable labels; however, this often led to short *rest* regions between gestures. This was due to the slow return to the *rest* state between trials as observed on several subjects. This caused significant deviations in onset and tail latencies. This variant otherwise used the same stratified cross-validation and preprocessing methods as in the original study evaluation (see Chapter 6). The effects and issues around the GLR are discussed later in the chapter.

On the supplementary data, since both labelling methods are available, all combinations of training and testing with the GLR and expert labels are evaluated

to compare and contrast the various performance trade-offs. This variant also used the same stratified cross-validation and preprocessing methods as in the original study evaluation.

Both experiments use the same validation folds as for the subject adaptation (described in Table 7.2).

7.2.3 Latch Algorithm

Latching is a simple, baseline, smoothing solution. An initial state y_0 is selected, typically the *rest*/neutral posture, and a stream length l_{stream} chosen based on acceptable latencies and testing with a given classifier. The algorithm functions by maintaining the current state unless the last l_{stream} predictions all predict a different state. Algorithm 7.2 describes the algorithm for completeness.

Algorithm 7.2 Latch algorithm for taking a stream of predictions and producing a smoother stream.

Input: $\hat{y}$: Stream of predicted labels, y_0 : Initial state, l_{stream} : Length of latch stream
Result: $\hat{\hat{y}}$ Stream of smoothed labels

$\hat{\hat{y}} = y_0$
Initialise Y `// Array of length` l_{stream} `for storing past predictions`
while *smoothing* **do**
  Get new $\hat{y}$
  Right shift Y
  $Y[0] = \hat{y}$
  if $Y[...] == Y[0]$ **then**
    $\hat{\hat{y}} = Y[0]$
  end
end

The only hyperparameter for the latch algorithm was the stream length l_{stream}; the values $[2, 5, 10, 20, 40]$ were trialled to demonstrate and contrast performance, latency and smoothness trade-offs.

7.2.4 Majority Voting Algorithm

Another algorithm for smoothing is majority voting. That is voting over the last l_{stream} windows and outputting the gesture with the most votes. Similar to the latch algorithm l_{stream} is the only hyperparameter and was trialled at the values $[5, 11, 21, 41]$.

7.2.5 The MSPRT Algorithm

The Multi-Hypothesis Sequential Probability Ratio Test (MSPRT) [237] is an algorithm that is used to simulate biological decision-making, such as found in the basal ganglia [238]. It effectively provides a recursive update for a set of evidence λ of a particular hypothesis, outputting a result when evidence crosses a given threshold:

$$\lambda_i = g^* \lambda_i - \log \sum_{n=1}^{N} \exp \lambda_n \tag{7.1}$$

$$y(\lambda) = \begin{cases} argmax(\lambda), & max(\lambda) \geq thres \\ y_{def}, & max(\lambda) < thres \end{cases} \tag{7.2}$$

where λ_i is evidence for a hypothesis i, g^* is a hyperparameter and there are N hypotheses. Effectively g^* represents a forgetting factor or how long evidence persists and $\log \sum_{n=1}^{N} \exp \lambda_n$ is a continuous approximation of the max function. The final outputted class is y, $thres$ is the threshold hyperparameter and y_{def} is a chosen default output for when there is not enough evidence for any particular class.

The default output y_{def} was set as the *rest* class and the other hyperparameters $thres$ and g^* were determined via grid search.

7.2.6 Smoothing with Hidden Markov Models

The Hidden Markov Model (HMM) is a method that has been shown to be useful in speech recognition [172] and useful as an addition to deep learning techniques [136]. In order to apply it as a smoothing algorithm, it was assumed the classifier outputs are the observations and that the true state was the associated labelled gesture as per the data.

In the same way as the Majority Vote and Latch algorithms, the HMM was also given a set length l_{stream} stream of data that includes the current classifier output and $l_{stream} - 1$ previous outputs. The Viterbi algorithm (see next section) was then used to predict the most likely sequence of states that led to the observed classifier outputs. Instead of predicting at the most recent time step, however, the prediction at the middle of the stream was used to facilitate smoothing. This introduced a more direct trade-off with latency because the algorithm actively predicted the true classification label at a previous time step.

The parameters for HMMs are often learnt using Expectation Maximisation (EM) [239] although this work forgoes direct parameter learning of the HMM due

to lack of a suitable training set. Instead, grid search was used to determine the emission and transmission matrices. The initial probability of the states was set uniformly so as to not make any assumptions about the expected classification.

HMMs have been successfully applied as an addition to the classification process in hand gesture recognition [3] in order to increase overall accuracy. The methods explored by Rossi *et al.* [3] require additional labelling, however, and so here the HMM is used only for smoothing rather than to switch between classifiers.

7.2.7 Viterbi Algorithm

The Viterbi algorithm [240] was used to calculate the most likely sequence of states given a sequence of observations. Algorithm 7.3 illustrates the process, which is essentially to iterate forward through a set of observations determining the probability of state transitions and then backtrace to find the most likely sequence of transitions from the endpoint probabilities.

Algorithm 7.3

Input: $P(S_1)$: Prior state probabilities, A : Transition matrix, B : Emission matrix, Y : Sequence of observations $\{Y_1, ..., Y_T\}$
Result: S : The most likely sequence of states $\{S_1, ..., S_T\}$

$\delta[1...K,\ 1] = P(S_1)$
$\psi[1...K,\ 1] = 0$
for *each observation* $i \in 2, ..T$ **do**
  for *each possible state* $j \in 1, ..K$ **do**
    $\delta[j,\ i] = max(\delta[1...K,\ i-1] \cdot A[1...K,\ j] \cdot B[j,\ Y[i]])$
    $\psi[j,\ i] = argmax(\delta[1...K,\ i-1] \cdot A[1...K,\ j] \cdot B[j,\ Y[i]])$
  end
end
$S[T] = argmax(\delta[1...K,\ T])$
for $i \in T, ..2$ **do**
  $S[i-1] = argmax(\psi[S[i],\ i])$
end

Similar to the vanishing gradient problem in deep learning, the Viterbi algorithm can run into numerical issues due to the potentially large number of multiplications of numbers ≤ 1. In practice, this issue is mitigated by computing the logarithms of the probabilities.

The Viterbi algorithm performs best on predicting states early in a sequence since these have more information associated with them.

7.3 Results and Discussion

7.3.1 Subject Adaptation

Table 7.3 summarises the performance of the adaptation algorithms over the 10 subjects in the study.

Algorithm	μ_{subj} (%)	μ_{rank}
No Adaptation	42.4	5.00
Baseline	84.6	3.80
Freeze Classification	86.2	1.60
Freeze Features	86.4	1.40
Dual Train	84.8	3.20

Table 7.3: Performance of subject adaptation algorithms on study data in terms of macro-average accuracy and rank. Freezing classification or feature layers are shown to improve over baseline method. The value μ_{subj} is the inter-subject mean macro-average accuracy, μ_{rank} is the inter-subject mean rank (range 1-5).

No adaptation, i.e. not using any data on the target subject, performed the worst. While this is the expected result, it still reinforces the need for subject-specific adaptation of classification in order to achieve high performance. The baseline performs in line with results from the previous chapter.

The two freezing algorithms significantly outperform the other algorithms at the 5% significance level, although neither is significantly better than the other. This result was calculated using the Friedman test followed by post-hoc Holm procedure. This demonstrates that, while it is still necessary to adapt for each subject, appropriate usage of data from other subjects can be used to boost performance. The frozen feature extractor algorithm, in particular, indicates that a common set of features is usable across subjects as would be expected given the extensive sEMG feature literature. The freeze algorithms also show a slight reduction in the inter-subject variance, from $\sim 7\%$ down to $\sim 6\%$, which is another desirable improvement over the baseline.

The dual training algorithm attempts to achieve a similar goal as the feature freezing algorithm; however, it uses a shared latent space within the classifier rather than pre-learning the feature extractor. The fact that it performed essentially the same as the baseline, however, indicates that the stronger assumption made by the feature freezing algorithm was more useful. Specifically, the feature freezing algorithm can be interpreted as applying the assumption:

"The most useful sEMG features are the same across all subjects"

It is useful to have some experimental evidence that this assumption is correct. The sEMG feature literature by necessity is aimed to design features that were

useful across subjects whereas being an end-to-end learning system deep neural networks present the opportunity to maximise customisation to a particular subject. Therefore knowing that it may be useful to limit customisation to some degree and to make use of data from other subjects is vitally important.

This result is similar to results from image classification [241, 242], where it has been shown that starting with a network pretrained on a similar problem and fine tuning some of its layers to a new problem results in better performance than only using data from the new problem, this is shown to be particularly true when the amount of data available is small.

The most recent studies have applied transfer learning in a way similar to the dual train methodology demonstrated here [28], and other studies have pre-initialised their networks on other subject data (a strategy found to be ineffective here). This work is the first to demonstrate that fine-tuning a pretrained network is a viable method for improving performance on a particular subject.

Table 7.4 shows the results of training the different adaptation algorithms with only a single repetition rather than the set of three for each fold.

	Macro-average Accuracy (%)				
Repetition Number	**Baseline**	**Freeze Classification**	**Freeze Features**	**Dual Train**	**Average Rank**
1	71.0	74.2	73.9	72.0	4.88
2	75.6	78.6	78.5	77.2	4.35
3	76.4	78.5	78.5	76.9	3.75
4	77.2	78.9	79.1	78.3	3.42
5	74.5	76.8	76.8	75.6	2.58
6	78.1	80.3	80.6	79.1	2.02

Table 7.4: Performance of subject adaptation methods when only trained on one repetition of the target subject's data. Repetition 1 is shown to produce a less useful classifier while the performance trends between the adaptation methods remain the same. Average rank was calculated over adaptation-subject pairs, i.e. over 40 matched samples.

The results show that repetition 1 generally leads to a poorer classifier, which provides further evidence for the conclusions from previous chapters that repetition 1 is generally a lower quality data set. Previously it was shown that repetition 1 had poorer performance in testing compared to other repetitions; however, here the other side is shown in that the classifier trained from repetition 1 is also lower quality. Statistically, it is possible to conclude that repetition 1 is significantly worse than the other repetitions except for repetition 5 at the 5% significance level using the Friedman test and post-hoc Holm procedure.

Unlike in previous chapters, it was found that there was a dip in performance for repetition 5. It is not clear exactly why this was the case; however, it potentially

stems directly from the experimental setup. Several subjects indicated their boredom around this point of the experiment, which may have caused a decline in gesture reproduction. Moving into the final repetition may then have reinvigorated their concentration on the experiment producing the observed performance improvement in repetition 6 relative to repetition 5. This has implications for future experiment design, suggesting that adding additional measures to keep subjects on task and produce a more consistent level of concentration may be useful ways of improving the quality of gesture reproduction.

The general trends between the performance of the adaptation algorithms remain the same as when all the data is available (Table 7.3) with both freezing algorithms outperforming the others. The one exception is the Dual Train algorithm, which provides a larger improvement over the baseline than when more data is available. This demonstrates that enforcing the shared latent space is beneficial when data on a subject is limited.

Overall, this subject adaptation experiment has shown for the first time that fine-tuning a network trained to be generalisable on other subjects is a useful way of enhancing performance without needing to design a new architecture. Further confirmation of the generally poor performance of the first repetition has been shown, and avenues for improving experiment design in the future have been presented.

7.3.2 Prediction Smoothing

Tables 7.5, 7.7, 7.6 and 7.8 show the results of the prediction smoothing algorithms on the supplemental study data for different labels in training and testing.

The general trend is similar across all variants; every smoothing algorithm improves the overall macro-average accuracy, reduces the Mean Deviations per movement but also increases both the onset and tail latencies.

Omitting preemptive results in the latency calculations has the most effect when the overall latency is lower. Once the latencies are higher, preemption is much less frequent. The tail latency is most affected by the removal of preemption, meaning that, in general, the classification is most likely to preemptively predict a return to the *rest* position as opposed to movement into a gesture. The tail latency is also affected more in the GLR trained and tested variant (Table 7.8) even at higher overall latencies.

The issues with the GLR in this context become apparent when comparing the tables. When GLR labels are used for training, and expert labels are used for testing (Table 7.6), the network's macro-average accuracy is slightly reduced along with the associated latencies indicating that for training purposes the two are not

Expert Trained, Expert Tested											
Algorithm	**Accuracy (%)**	**Mean Dev.**		**Onset (ms)**		**Tail (ms)**		**Onset* (ms)**		**Tail* (ms)**	
		μ	P_{90}	μ	P_{90}	μ	P_{90}	μ	P_{90}	μ	P_{90}
Baseline	94.4	21.7	60.3	17.5	55.0	13.7	55.0	27.3	65.0	32.4	85.0
Latch (2)	94.7	7.1	19.0	26.4	71.5	21.1	70.0	33.8	85.0	41.8	98.0
Latch (5)	95.0	3.1	9.0	47.8	118.0	39.2	115.0	48.9	122.0	51.0	120.0
Latch (10)	95.3	1.7	5.3	82.6	170.0	71.3	149.5	82.6	170.0	78.8	169.0
Latch (20)	95.6	0.7	2.0	152.6	241.5	132.8	245.0	152.6	241.5	133.6	245.0
Latch (40)	95.0	0.1	0.3	286.1	431.5	243.0	360.0	286.1	431.5	243.0	360.0
HMM (5)	94.8	7.3	19.3	29.9	76.5	21.8	71.5	36.9	90.0	37.8	95.0
HMM (11)	94.9	4.2	11.3	43.7	95.0	33.8	90.0	45.1	95.0	41.5	102.0
HMM (21)	95.1	2.5	7.0	70.3	126.5	62.6	130.0	70.3	126.5	69.6	135.0
HMM (41)	95.2	1.3	5.0	118.6	165.0	108.1	170.0	118.6	165.0	110.1	170.0
HMM (61)	95.3	0.7	3.0	167.7	210.0	152.8	205.0	167.7	210.0	152.8	205.0
Vote (5)	94.8	7.2	19.3	29.9	76.5	21.7	71.5	36.7	90.0	37.6	95.0
Vote (11)	94.9	4.1	11.3	44.6	95.0	33.8	90.0	45.1	95.0	41.5	102.0
Vote (21)	95.1	2.5	7.0	70.2	126.5	61.0	130.0	70.2	126.5	67.9	130.0
Vote (41)	95.2	1.3	5.0	120.3	166.5	108.1	170.0	120.3	166.5	110.1	170.0
MSPRT	95.7	1.6	5.3	90.7	140.0	57.5	95.0	90.7	140.0	61.9	95.0

Table 7.5: Smoothing algorithm performance on supplementary data, trained and tested on expert labelled data. The value μ is the mean and P_{90} is the 90th percentile. Accuracy is given in terms of macro-average. *Preemptive results omitted.

GLR Trained, Expert Tested											
Algorithm	**Accuracy (%)**	**Mean Dev.**		**Onset (ms)**		**Tail (ms)**		**Onset* (ms)**		**Tail* (ms)**	
		μ	P_{90}	μ	P_{90}	μ	P_{90}	μ	P_{90}	μ	P_{90}
Baseline	93.1	22.4	53.6	23.9	95.0	189.8	280.0	52.8	125.0	196.9	280.0
Latch (2)	93.4	7.4	19.0	29.6	101.5	198.4	285.0	64.5	142.0	202.0	285.0
Latch (5)	93.8	3.5	10.0	53.4	150.0	221.0	313.0	77.4	160.0	222.3	314.0
Latch (10)	94.1	1.9	5.0	84.2	180.0	264.9	371.5	90.6	182.5	266.5	372.0
Latch (20)	94.4	0.8	3.0	167.1	301.5	332.8	456.0	167.1	301.5	334.8	458.0
Latch (40)	93.5	0.2	1.0	310.2	495.0	443.8	589.5	310.2	495.0	443.8	589.5
HMM (5)	93.5	7.7	20.0	31.1	110.0	202.5	290.0	63.0	144.0	203.7	290.0
HMM (11)	93.7	4.5	12.0	40.0	131.5	220.5	315.0	64.7	153.5	223.2	315.0
HMM (21)	94.0	2.8	8.0	63.5	170.0	249.0	353.0	69.2	173.5	250.5	354.0
HMM (41)	94.3	1.5	5.0	108.6	210.0	295.7	391.5	108.6	210.0	297.5	392.0
HMM (61)	94.4	0.9	3.0	156.2	238.0	333.3	438.0	156.2	238.0	333.3	438.0
Vote (5)	93.5	7.5	19.0	31.2	110.0	202.4	290.0	63.2	144.0	203.6	290.0
Vote (11)	93.7	4.4	11.3	40.7	131.5	220.1	311.5	63.3	155.0	222.8	312.5
Vote (21)	94.0	2.8	8.0	63.1	170.0	246.8	349.5	68.9	173.5	248.2	351.0
Vote (41)	94.2	1.5	5.0	110.4	211.5	295.2	386.5	110.4	211.5	297.0	387.0
MSPRT	94.7	1.9	6.0	82.0	161.5	180.9	321.5	83.0	162.5	184.2	323.0

Table 7.6: Smoothing algorithm performance on supplementary data, trained on GLR labelled data and tested on expert labelled data. The value μ is the mean and P_{90} is the 90th percentile. Accuracy is given in terms of macro-average. *Preemptive results omitted.

Expert Trained, GLR Tested											
Algorithm	Accuracy (%)	Mean Dev.		Onset (ms)		Tail (ms)		Onset* (ms)		Tail* (ms)	
		μ	P_{90}	μ	P_{90}	μ	P_{90}	μ	P_{90}	μ	P_{90}
Baseline	89.8	22.2	61.0	197.4	735.0	2.0	0.0	197.4	735.0	36.7	81.0
Latch (2)	90.0	7.6	19.0	208.0	740.0	2.4	0.0	208.0	740.0	45.0	91.0
Latch (5)	90.5	3.8	9.0	231.0	768.0	3.4	0.0	231.0	768.0	47.1	101.5
Latch (10)	91.1	2.6	6.0	265.9	805.0	4.5	0.0	265.9	805.0	62.5	126.5
Latch (20)	92.1	1.5	3.0	335.9	859.5	12.6	45.0	335.9	859.5	65.9	139.5
Latch (40)	92.6	0.7	1.0	469.4	994.5	48.8	168.0	469.4	994.5	110.7	220.0
HMM (5)	90.1	7.9	20.0	212.2	745.0	2.5	0.0	212.2	745.0	42.5	95.0
HMM (11)	90.4	4.9	12.3	226.9	760.0	3.5	0.0	226.9	760.0	48.3	106.0
HMM (21)	90.8	3.3	8.0	253.6	785.0	4.6	0.0	253.6	785.0	65.0	117.5
HMM (41)	91.6	2.2	5.3	301.8	836.5	7.7	10.0	301.8	836.5	71.7	151.0
HMM (61)	92.2	1.5	3.3	350.9	886.5	11.5	38.0	350.9	886.5	68.8	155.5
Vote (5)	90.1	7.7	20.0	212.2	745.0	2.5	0.0	212.2	745.0	42.5	95.0
Vote (11)	90.4	4.9	12.0	227.8	760.0	3.5	0.0	227.8	760.0	48.3	106.0
Vote (21)	90.8	3.3	8.0	253.5	785.0	4.6	0.0	253.5	785.0	64.6	113.0
Vote (41)	91.5	2.2	5.3	303.5	841.5	7.7	10.0	303.5	841.5	71.7	151.0
MSPRT	91.6	2.5	6.0	274.0	811.5	3.6	0.0	274.0	811.5	60.0	107.0

Table 7.7: Smoothing algorithm performance on supplementary data, trained on expert labelled data and tested on GLR labelled data. The value μ is the mean and P_{90} is the 90th percentile. Accuracy is given in terms of macro-average. *Preemptive results omitted.

GLR Trained, GLR Tested											
Algorithm	Accuracy (%)	Mean Dev.		Onset (ms)		Tail (ms)		Onset* (ms)		Tail* (ms)	
		μ	P_{90}	μ	P_{90}	μ	P_{90}	μ	P_{90}	μ	P_{90}
Baseline	90.1	25.1	59.3	183.6	720.0	13.2	36.5	185.8	720.0	27.1	69.5
Latch (2)	90.3	8.6	20.0	197.2	730.0	17.1	45.0	198.4	730.0	32.4	76.0
Latch (5)	90.8	4.4	10.3	230.9	746.5	27.6	66.5	232.3	747.0	47.3	95.0
Latch (10)	91.3	2.6	6.0	266.7	771.5	47.6	116.5	266.7	771.5	76.2	135.0
Latch (20)	92.3	1.1	3.0	350.4	862.5	92.7	195.0	350.4	862.5	122.6	205.0
Latch (40)	91.9	0.3	1.0	493.3	1013.0	177.8	295.0	493.3	1013.0	204.6	300.0
HMM (5)	90.4	9.0	21.0	201.0	731.5	18.6	46.5	202.2	732.0	36.0	80.0
HMM (11)	90.7	5.5	12.3	217.4	745.0	27.0	61.5	218.7	745.0	48.8	95.0
HMM (21)	91.1	3.7	8.3	246.0	770.0	40.8	91.5	246.0	770.0	65.9	115.0
HMM (41)	91.8	2.1	5.3	291.8	821.5	67.4	145.0	291.8	821.5	94.3	155.5
HMM (61)	92.3	1.3	4.0	339.5	871.5	90.9	183.0	339.5	871.5	123.2	195.0
Vote (5)	90.4	8.8	20.0	201.8	731.5	18.6	46.5	203.0	732.0	36.0	80.0
Vote (11)	90.7	5.4	12.0	218.5	745.0	27.0	61.5	219.9	745.0	48.8	95.0
Vote (21)	91.1	3.6	8.0	245.7	770.0	39.5	91.5	245.7	770.0	65.7	115.0
Vote (41)	91.8	2.1	6.0	293.7	821.5	66.9	145.0	293.7	821.5	93.7	155.5
MSPRT	92.1	2.6	6.0	265.1	793.0	35.9	96.5	265.1	793.0	61.5	115.0

Table 7.8: Smoothing algorithm performance on supplementary data, trained and tested on GLR labelled data. The value μ is the mean and P_{90} is the 90th percentile. Accuracy is given in terms of macro-average. *Preemptive results omitted.

overly different. However, when training and testing on GLR labels (Table 7.8), the network performs worse across all measures. The variation in latency is also increased as the network is more likely to misclassify larger sections of data.

Since it may be reasonably assumed that the expert labels are more representative of the ground truth. It is, therefore, reasonable to assume that the GLR, while being an acceptable substitute for expert labels during training, is not a suitable method for evaluating actual expected performance. This is an important result since the GLR is often used for labelling purposes in the field, including its usage in previous chapters where the NinaPro databases were used since the "refined" movement labels presented in those databases are based on the GLR.

The GLR was initially developed to detect the onset of muscle contraction using EMG where it proved useful over threshold methods which were typical at the time [179]. This is subtly different from the classification problem where it is currently used. During classification, it is desirable to detect when a subject has reached a given gesture as opposed to detection of the onset of muscle contraction. Likewise, at the end of a gesture where the GLR is used to estimate the return to the *rest* state, the time step it predicts will be somewhat after the subject has stopped holding the gesture. This is why the GLR does not make a suitable labelling method for evaluating actual expected performance in this context.

These issues were even more apparent in the results from the cross-validation on the full study data. Table 7.9 presents the useful results of the full study. The latency measures are omitted because they continue the trend of the GLR trained and tested pilot (Table 7.8), i.e. they become excessively large and have a large spread. Further, a large number of preemptions were observed indicating the measures of latency were less useful.

The problems with the GLR are exacerbated here since some subjects moved slowly into and out of the *rest*/neutral state causing short regions to be labelled as *rest* when ideally the labelling of *rest* would occur as soon as the subject stopped performing a gesture even if there was a delay before actually coming completely to *rest*.

Table 7.9, however, still demonstrates that the same performance trends are present as in the supplemental data. Specifically, every smoothing algorithm improves the overall macro-average accuracy while reducing the mean deviations. Based on the other results, however, this improvement comes at the cost of increased latencies.

Having noted the issues with the GLR, the focus is now returned to Table 7.5 since it covers the expertly labelled and tested data which most accurately reflects real-world conditions. From this data, the Pareto Frontiers were plotted in Figures

Full Study, GLR Labels			
Algorithm	**Macro-average Accuracy (%)**	**Mean Deviations**	
		μ	P_{90}
Baseline	81.6	61.8	4.0
Latch (2)	81.9	20.3	2.0
Latch (5)	82.5	10.1	1.0
Latch (10)	83.4	5.8	0.0
Latch (20)	84.5	2.7	0.0
Latch (40)	85.4	0.6	0.0
HMM (5)	82.0	21.7	1.9
HMM (11)	82.3	13.1	1.0
HMM (21)	82.9	8.7	0.0
HMM (41)	83.8	5.2	0.0
HMM (61)	84.5	3.5	0.0
Vote (5)	82.0	20.9	1.0
Vote (11)	82.3	12.9	1.0
Vote (21)	82.8	8.6	0.0
Vote (41)	83.8	5.2	0.0
MSPRT	83.9	6.2	0.0

Table 7.9: Smoothing algorithm performance cross-validated on study data, GLR used for labelling. The value μ is the mean and P_{90} is the 90th percentile, both calculated across all subject-gesture-repetition combinations. Accuracy is given in terms of macro-average. *Preemptive results omitted.

7.3 and 7.4.

What the Pareto frontiers show is that the various smoothing algorithms are relatively similar when it comes to the trade-off between smoothness and latency; each algorithm lies either on or close to both frontiers. Only the latching algorithm has any clear downside compared to the others; it has worse 90th percentile expected latencies than the other algorithms (see Table 7.5). This poorer performance is caused by the fact that the algorithm sometimes causes particularly long delays. This is not unique and happens in all the algorithms; however, it happens more frequently with the latch. The reason for this is that the latch algorithms requires an uninterrupted stream of predictions for a specific gesture in order to change to it. Even a single different prediction causes a delay equal to the length of the latch, which is compounded by additional different predictions. The other algorithms are more tolerant of different predictions within their incoming prediction stream.

It is also important to note *how* the algorithms manipulate the predictions. A key example here is the MSPRT which outputs *rest* when there is no strong evidence for another gesture. This significantly changes the distribution of misclassifications which may or may not be beneficial, depending on the context, but here,

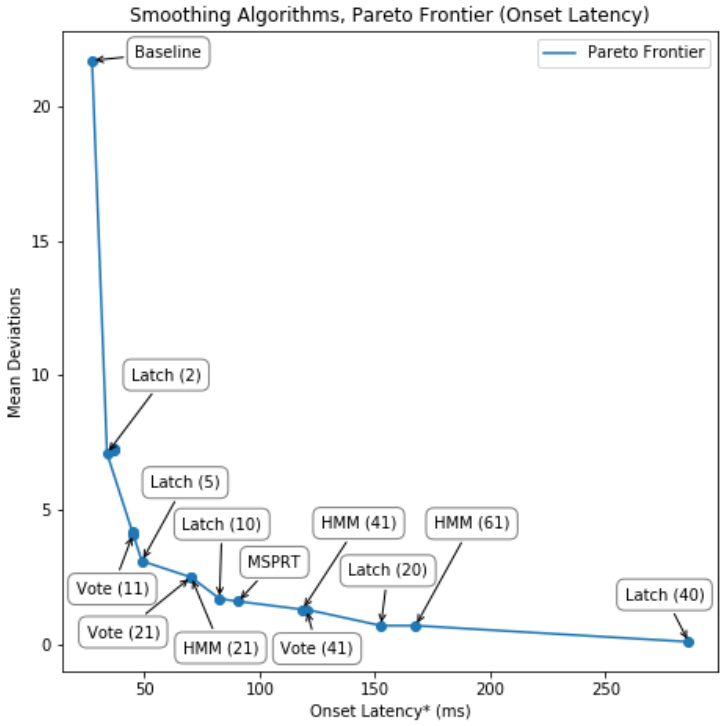

Figure 7.3: Pareto frontier of mean deviations against onset latency. Data from Table 7.5. All algorithms tested lie on or very close to the frontier, indicating that all have a high degree of Pareto efficiency.

as can be seen from Table 7.5, it improves upon the overall accuracy and tail latency compared to other algorithms. Similarly, the HMM could be modified to use a non-uniform prior on the initial state, biasing it towards certain gestures, which could be useful. The other algorithms have a much lower impact on prediction distribution.

Another key difference is that all the smoothing algorithms, because they operate over longer intervals than single point prediction, tend to change the misclassification distribution such that more of the misclassifications are due to latency rather than switching and holding an incorrect classification. In applications such as prosthetic control, this may be preferable since the cost of misclassification is likely to be higher than the cost of additional latency.

Therefore it can be surmised that, for the majority of cases, applying any of the smoothing algorithms with appropriately tuned hyperparameters will produce a reasonable trade-off. However, if maintaining the classification distribution is important, the MSPRT should not be used. Conversely the MSPRT or HMM should be used if it is desirable to bias the classification towards certain classes. Otherwise, majority voting makes a reasonable algorithm choice since it requires min-

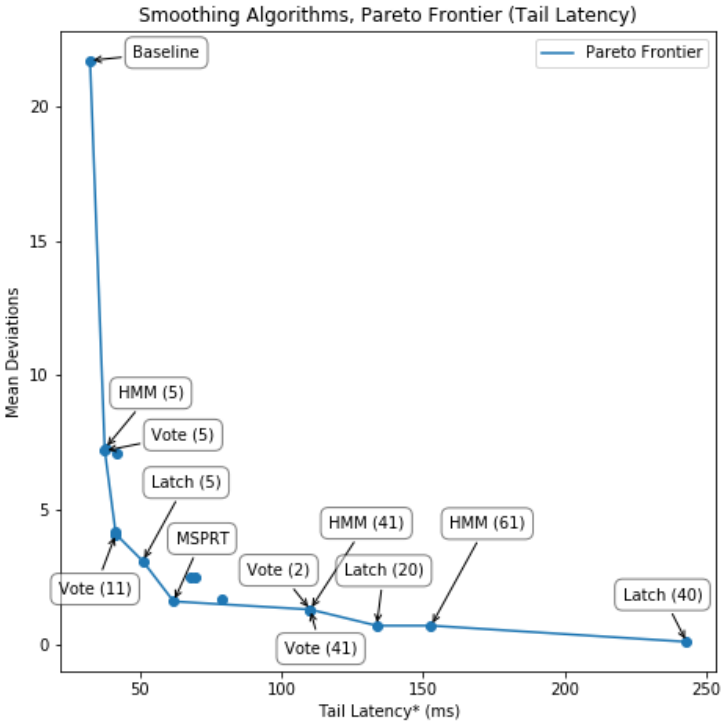

Figure 7.4: Pareto frontier of mean deviations against tail latency. Data from Table 7.5. Some algorithm variants lie behind the frontier; however, most still lie on the frontier itself, indicating that all have a high degree of Pareto efficiency.

imal hyperparameter tuning, specialist knowledge and computational overhead while providing a useful reduction in the mean deviations over no smoothing.

7.4 Integration of Techniques

The aim of this thesis is to present an improved set of methods for sEMG based hand gesture classification. In this section, the techniques and methods discussed in this and previous chapters are brought together to demonstrate both the performance possible and the improvements over other solutions.

The supplemental data with expert labels was used to best represent the ground truth, and the same cross-validation folds were used as in Table 7.2. The TtS and Compact TtS were tested making use of SMOTE, subject adaptation and smoothing algorithms to enhance performance further.

The TtS network performed best with feature freezing pretraining using the main study data labelled using the GLR and achieved a macro-average accuracy of 91.6% before smoothing. Combined with the latch 20 algorithm, this rose to 93.2% with an average onset latency of 188.6ms and mean deviations 1.0. In contrast, the

Compact TtS performed worse with the feature freezing pretraining likely due to its smaller number of parameters being less flexible in capturing an appropriate generalisation of features. Here SMOTE provided a performance enhancement over the subject adaptation, which is more useful on the non-compact version of the TtS. The Compact TtS achieved 90.3% macro-average accuracy rising to 92.0% with the latch 20 algorithm generating 1.0 mean deviations but with a higher average onset latency of 243.8ms.

The performance of the latching algorithm on the Compact TtS is shown in Figure 7.5 as the latch length increases to illustrate the changes to the classification signal.

Together the results show the need for a more nuanced approach throughout the classification process. The fact that subject adaptation does not work on the Compact TtS demonstrates the need for flexibility in training processes to account for both the data available and the network's structure. Similarly the harder to quantify differences in smoothing algorithms such as how they change the timing distributions of misclassifications and the overall distribution of misclassifications as well as the smoothness of the resultant classification signal are of differing importance in different applications but are still relevant and are largely obfuscated by standard measures.

The differences in latencies between the compact and non-compact networks further evidences the idea that the end-to-end classification process has a large decision space of complex inter-linked trade-offs and hence supports the need for nuanced evaluation with respect to particular applications.

7.5 Conclusion

Subject adaptation and prediction smoothing have been investigated in this chapter. Both of these investigations built on work from previous chapters to develop improvements to the classification process to make it more applicable to real-world scenarios.

Subject adaptation was used to improve the performance of the TtS network in situations where similar data was available from other subjects. Pretraining the network on data from other subjects, freezing the lower feature extraction layers then fine-tuning the classification layer to a new subject was shown to improve the overall macro-average accuracy by 1.8%. This mirrors results from the image classification domain [232] where using pretrained networks and fine tuning to problems is a well-known tactic. However, for sEMG hand classification, this is the first successful demonstration of the technique. While the overall performance

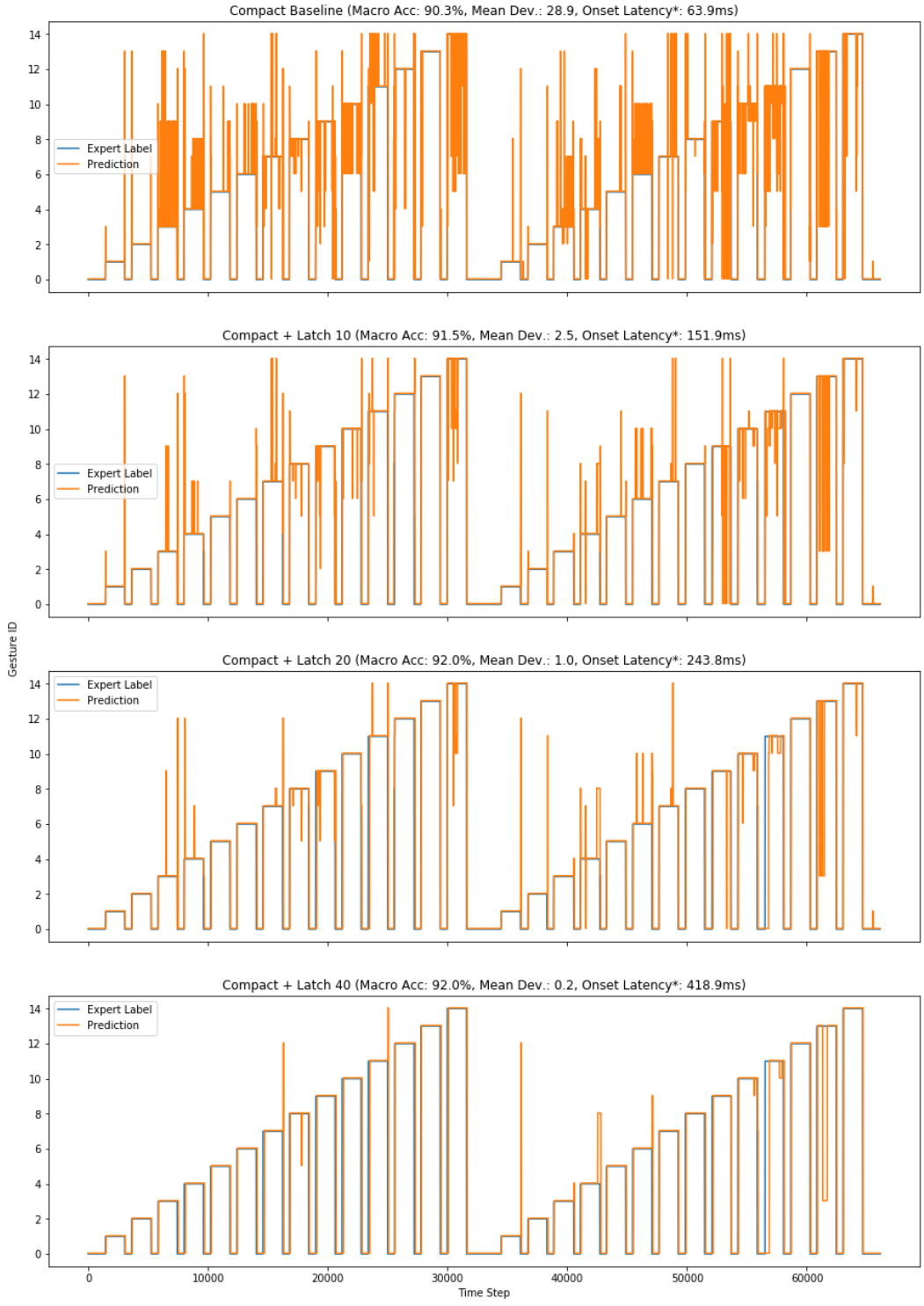

Figure 7.5: Classification signal against ground truth from the first cross-validation fold for the Compact TtS with different length latches showing the increase in smoothness and added latency. Metrics given are for the complete cross-validation.

improvement is relatively small from this method the fact that in any study with multiple subjects the improvement only costs minimal computation time and is potentially applicable to any neural network makes it particularly valuable. It was also demonstrated that it is not desirable to only train on other subjects and that the fine-tuning is a necessary step, even when the same gestures are being classified. This supports continuing research into person-specific adaptations.

The exploration of post-classification smoothing algorithms for better online classification showed that, for a small trade-off in latency, a significant reduction in the number of expected deviations per movement could be made through the use of a number of smoothing techniques. The investigation showed most of the algorithms had a similar trade-off between latency and mean deviations and that all also provided an additional boost to the overall macro-average accuracy. Therefore it was demonstrated that these algorithms make a useful addition to simply using the output from the classifier, particularly in applications that might be sensitive to the high-frequency flickering of the classification signal.

Both these methods present ways for researchers to improve performance on the classification problem in general purpose, practically implementable ways, which would make sEMG based interaction and control more viable in real applications.

The integration testing demonstrated that it is not possible to unequivocally produce a single best set of methods for any given setup. It did, however, demonstrate that it is possible to classify hand gestures with a high macro-average accuracy using a compact network that could be run on an embedded device while using smoothing algorithms to trade off smoothness, latency and other important characteristics to fit the needs of a particular problem.

Chapter 8

Closing Remarks

8.1 Summary and Conclusions

This thesis has centred on improving the performance, robustness and reproducibility of sEMG based gesture classification. The work has been largely exploratory in nature, with a wide variety of techniques and classifiers being evaluated to determine which are the most effective, and has been formulated as a direct response to the methodological issues highlighted in the field. The main contributions of this work can be split into three sections: improved evaluation, higher performance classification and better supporting methods.

The importance of appropriate evaluation was highlighted in Chapter 3. It was shown that the standard accuracy metric frequently used in the field leads to a class frequency bias in most experiments which is unlikely to be representative of the class frequency at application time or desired class weighting. Therefore the macro-average accuracy was proposed as an alternative that uses a uniform a priori probability on expected class distribution which leads to a more representative metric for the purposes of evaluation without a known application time class distribution. Further, cross-validation techniques were implemented which are vital for reliable, reproducible results [183, 184] but have often been neglected in the field. A novel stratification technique for cross-validation was also presented and validated that solves a common stratification problem associated with gesture classification.

This thesis contributed several classification performance improvements to the field that operate along different axes. The performance of feature-based classifiers was improved by data resampling and augmentation techniques which were demonstrated in the re-evaluation of a landmark study. A novel neural network design was presented, which significantly outperforms both other classification

methods and contemporary network designs. The network design was also modified to minimise the number of parameters necessary for a similar level of performance. This lead to tangible run time performance enhancement on embedded hardware, which is necessary for many real-world applications where computational power is limited.

Several supporting methodologies were also designed and evaluated, which have practical value to current and future researchers.

The effect of different labelling methodologies was explored and shown to have a potentially significant impact on the reported performance due to differences between the presented labels and the ground truth demonstrating an often overlooked area of experiment design that potentially deserves further investigation.

Gesture sub-selection was explored which demonstrated that superset performance is a useful metric for selecting the best gestures for a subject, when many are available, in order to maximise classification performance. This is useful for any situation where only a subset of all the gestures that there is data on need to be classified at any one time: a situation that can often be engineered if performance is critical and allows tailoring of the classification to an individual.

Fine-tuning, and more generally utilising data from other subjects, was shown to be a viable way of improving classifier performance. This is useful in most studies and applications since data from multiple subjects is generally available. This work also indicates that cross-subject learning is a useful endeavour for improving overall performance, a hypothesis which has recently been shown to have merit [45].

Online smoothing algorithms were explored in terms of the trade-offs they present, a topic which has not been explored within the current context of sEMG based gesture classification. It was shown that many algorithms lie on or near the Pareto front for trading off latencies and smoothness while all also provide a boost to the overall accuracy of the classification. This presents a potentially low computational cost way of improving classification performance regardless of the classifier in use and making informed trade-offs with the expected latency.

A novel study was also presented that explored the performance of the low-cost Myo Armband compared to a more expensive medical counterpart indicating that under the right conditions it was possible for it to produce data that lead to similar or better performance than its more expensive counterpart. This potentially represents the opportunity to drastically reduce the barrier to entry to the field.

Bringing everything together it has been shown that it is possible to achieve

high-performance classification on the sEMG hand classification problem while allowing for the nuanced trade-off between other characteristics of the classifier such as computation time, latencies and smoothness of signal.

In summary, the contributions of this thesis are a set of methods, techniques and designs to improve sEMG hand gesture classification in terms of raw classification performance, run-time computational cost, representative performance evaluation, experiment design, and performance trade-offs as well providing freely available data on low-cost data gathering equipment and code for reproducibility. Complete processes were also evaluated that demonstrated how these techniques could be combined to produce high-quality gesture classification.

8.2 Future Research Avenues

This work raises several potential avenues for future research work:

- The macro-average accuracy metric has been shown to be a useful metric since it applies a uniform a priori probability to the expected class distribution at application time. In specific applications, it, therefore, would be useful to have an approximate expected class distribution. Since the particular gestures involved may be different between studies, it would likely be most useful to determine expected distributions for *rest* vs other gestures which would allow informed trade-off choices and appropriate metric weighting.

- In a similar vein, determining the specific trade-offs with non-accuracy parameters for different applications such as the maximum possible additional latency from classification + smoothing and the degree to which smoothing is necessary. Along with weighing other concerns such as memory footprint and necessary computation would allow for informed design choices with respect to particular applications.

- Chapters 6 and 7 highlighted potential issues around common labelling processes used to label the ground truth of data sets. It would be interesting to explore this line of research with larger amounts of data based on determining what labelling methods can be used in what contexts. Similarly, exploration of alternative labelling techniques that can better approximate expert labelling would also be very useful for the field since using experts, which was shown to work best here, is human time-intensive.

- The fine-tuning results from Chapter 7 show that data from other subjects can be used to improve performance on a target subject. Exploring alternative architectures and methods to exploit this to its maximum potential is,

therefore, an exciting area and is starting to be explored by other researchers [45]. Mixed error-component modelling may also be a useful extension to this process to precisely the sources of improvement when a number of methods are combined.

- The fusion of other data sources with EMG appears to be another promising approach for improving overall performance in some contexts. Work done by McIntosh *et al.* [149] has shown that pressure sensing can be a useful augmentation to the data input and other work suggests accelerometer data can also be beneficial, e.g. Xu *et al.* [151].

- Further exploration of the utility of the Myo Armband or other low-cost sEMG devices. Reducing the cost of hardware by a factor of $\sim 100\times$ has the potential to vastly reduce the barriers to entry in the field, and recent work has shown promising results not just on healthy subjects but also transradial amputees [231]. Since this thesis was exploratory, a follow-up study would be useful to confirm the hypothesis that for classification of a reasonable number of gestures the Myo Armband can achieve performance at least as good as its more expensive counterparts.

- Approaching the problem as a regression rather than a classification may be beneficial for some applications where precise tracking of the hand is useful. This potentially increases the difficulty of the problem but allows for finer control of devices. Data for training purposes may also be more difficult to capture since joint angles must be captured. However, some open source data sets are available to support this research avenue, e.g. finger force-pattern exercises captured with synchronised sEMG and joint angles [37].

- Modelling the dynamics of the hand or arm is a potential avenue for improving what can be inferred from EMG signals. This may help support classification or allow inference of hand/arm position. The key challenge is designing the model to minimise the need for acquisition of the physical parameters of the arm. For instance, the bioimpedance of different volumes of the arm may be difficult to acquire but could make a large difference to the EMG signals received at a given electrode.

In general, the recommended future work is based on hypotheses testing to experimentally confirm trends observed in this work, building upon the classification methods to make maximum use of available data and exploring how techniques can be adapted to best suit specific applications.

Bibliography

[1] B.M. Nigg and W. Herzog. *Biomechanics of the Musculo-Skeletal System*. John Wiley & Sons, 2007.

[2] F. Tenore, A. Ramos, A. Fahmy, S. Acharya, R. Etienne-Cummings, and N.V. Thakor. Towards the control of individual fingers of a prosthetic hand using surface EMG signals. In *Proceedings of the 29th Annual International Conference of the IEEE Engineering in Medicine and Biology Society*, pages 6145–6148. 2007.

[3] M. Rossi, S. Benatti, E. Farella, and L. Benini. Hybrid EMG classifier based on HMM and SVM for hand gesture recognition in prosthetics. *Proceedings of the IEEE International Conference on Industrial Technology*, pages 1700–1705, 2015.

[4] R. Lipovský and H.A. Ferreira. Self hand-rehabilitation system based on wearable technology. In *Proceedings of the 3rd 2015 Workshop on ICTs for Improving Patients Rehabilitation Research Techniques*, pages 93–95, 2015.

[5] X. Zhang, X. Chen, W.h. Wang, J.h. Yang, V. Lantz, and K.q. Wang. Hand gesture recognition and virtual game control based on 3D accelerometer and EMG sensors. In *Proceedings of the 14th International Conference on Intelligent User Interfaces*, pages 401–406. 2009.

[6] L. Bai, M.G. Pepper, Y. Yana, S.K. Spurgeon, and M. Sakel. Application of low cost inertial sensors to human motion analysis. In *Proceedings of the International Conference on Instrumentation and Measurement Technology*, pages 1280–1285. 2012.

[7] Z. Zhang. Microsoft kinect sensor and its effect. *IEEE multimedia*, 19(2):4–10, 2012.

[8] M. Zobl, M. Geiger, K. Bengler, and M. Lang. A usability study on hand gesture controlled operation of in-car devices. In *Proceedings of the International Conference on Human-Computer Interaction*, 2001.

[9] M. Alpern and K. Minardo. Developing a car gesture interface for use as a secondary task. In *Extended Abstracts on Human Factors in Computing Systems*, pages 932–933. 2003.

[10] J. Pauchot, L. Di Tommaso, A. Lounis, M. Benassarou, P. Mathieu, D. Bernot, and S. Aubry. Leap Motion Gesture Control With Carestream Software in the Operating Room to Control Imaging: Installation Guide and Discussion. *Surgical Innovation*, 22(6):615–620, 2015.

[11] J.P. Wachs, M. Kölsch, H. Stern, and Y. Edan. Vision-based hand-gesture applications. *Communications of the ACM*, 54(2):60–71, 2011.

[12] J. Kela, P. Korpipää, J. Mäntyjärvi, S. Kallio, G. Savino, L. Jozzo, and D. Marca. Accelerometer-based gesture control for a design environment. *Personal and Ubiquitous Computing*, 10(5):285–299, 2006.

[13] S. Balasubramanian, E. Garcia-Cossio, N. Birbaumer, E. Burdet, and A. Ramos-Murguialday. Is EMG a Viable Alternative to BCI for Detecting Movement Intention in Severe Stroke? *IEEE Transactions on Biomedical Engineering*, 65(12):2790–2797, 2018.

[14] F. Lotte, M. Congedo, A. Lécuyer, F. Lamarche, and B. Arnaldi. A review of classification algorithms for EEG-based brain–computer interfaces. *Journal of Neural Engineering*, 4(2):R1, 2007.

[15] L. Kendrick, A. Bzostek, and V.J. Doerr. System and method for tracking positions of uniform marker geometries, U.S. Patent 9220573, dec 2015.

[16] P. Pławiak, T. Sośnicki, M. Niedźwiecki, Z. Tabor, and K. Rzecki. Hand body language gesture recognition based on signals from specialized glove and machine learning algorithms. *IEEE Transactions on Industrial Informatics*, 12(3):1104–1113, 2016.

[17] T.A. Kuiken, G. Li, B.A. Lock, R.D. Lipschutz, L.A. Miller, K.A. Stubblefield, and K.B. Englehart. Targeted muscle reinnervation for real-time myoelectric control of multifunction artificial arms. *Journal of American Medicine*, 301(6): 619–628, 2009.

[18] H. Jahani Fariman, S.A. Ahmad, M. Hamiruce Marhaban, M. Ali Jan Ghasab, and P.H. Chappell. Simple and computationally efficient movement classification approach for EMG-controlled prosthetic hand: ANFIS vs. artificial neural network. *Intelligent Automation and Soft Computing*, 21(4):559–573, 2015.

[19] C. Castellini and P. Van Der Smagt. Surface EMG in advanced hand prosthetics. *Biological Cybernetics*, 100(1):35–47, 2009.

[20] S. Shin, R. Tafreshi, and R. Langari. A performance comparison of hand motion EMG classification. In *Proceedings of the 2nd Middle East Conference on Biomedical Engineering*, pages 353–356. 2014.

[21] A. Balbinot and G. Favieiro. A neuro-fuzzy system for characterization of arm movements. *Sensors*, 13(2):2613–2630, 2013.

[22] M. Khezri and M. Jahed. A neuro-fuzzy inference system for sEMG-based identification of hand motion commands. *IEEE Transactions on Industrial Electronics*, 58(5):1952–1960, 2011.

[23] R.N. Khushaba, A. Al-Jumaily, and A. Al-Ani. Evolutionary fuzzy discriminant analysis feature projection technique in myoelectric control. *Pattern Recognition Letters*, 30(7):699–707, 2009.

[24] M. Khezri and M. Jahed. Real-time intelligent pattern recognition algorithm for surface EMG signals. *Biomedical Engineering Online*, 6(1):1, 2007.

[25] Y. Huang, K.B. Englehart, B. Hudgins, and A.D.C. Chan. A Gaussian mixture model based classification scheme for myoelectric control of powered upper limb prostheses. *IEEE Transactions on Biomedical Engineering*, 52(11): 1801–1811, 2005.

[26] M.F. Lucas, A. Gaufriau, S. Pascual, C. Doncarli, and D. Farina. Multichannel surface EMG classification using support vector machines and signal-based wavelet optimization. *Biomedical Signal Processing and Control*, 3 (2):169–174, 2008.

[27] M. Zia-ur Rehman, A. Waris, S. Gilani, M. Jochumsen, I. Niazi, M. Jamil, D. Farina, and E. Kamavuako. Multiday EMG-Based Classification of Hand Motions with Deep Learning Techniques. *Sensors*, 18(8):2497, 2018.

[28] U. Côté-Allard, C.L. Fall, A. Drouin, A. Campeau-Lecours, C. Gosselin, K. Glette, F. Laviolette, and B. Gosselin. Deep Learning for Electromyographic Hand Gesture Signal Classification by Leveraging Transfer Learning. *arXiv:1801.07756*, 2018.

[29] Z. Ju, G. Ouyang, M. Wilamowska-Korsak, and H. Liu. Surface EMG based hand manipulation identification via nonlinear feature extraction and classification. *IEEE Sensors Journal*, 13(9):3302–3311, 2013.

[30] R.N. Khushaba and A. Al-Jumaily. Fuzzy wavelet packet based feature extraction method for multifunction myoelectric control. *International Journal of Biological and Medical Sciences*, 2(3):186–194, 2007.

[31] F.V.G. Tenore, A. Ramos, A. Fahmy, S. Acharya, R. Etienne-Cummings, and N.V. Thakor. Decoding of individuated finger movements using surface electromyography. *IEEE Transactions on Biomedical Engineering*, 56(5):1427–1434, 2009.

[32] R. Zhou, X. Liu, and G. Li. Myoelectric signal feature performance in classifying motion classes in transradial amputees. In *Proceedings of the Congress of the International Society of Electrophysiology and Kinesiology*, pages 16–19, 2010.

[33] M.J. Ortiz-Catalan, R. Brånemark, and B. Håkansson. Biologically inspired algorithms applied to prosthetic control. In *BioMed 2012*, page 764, 2012.

[34] M. Zia-ur Rehman, S. Gilani, A. Waris, I. Niazi, G. Slabaugh, D. Farina, and E. Kamavuako. Stacked sparse autoencoders for EMG-based classification of hand motions: A comparative multi day analyses between surface and intramuscular EMG. *Applied Sciences*, 8(7):1126, 2018.

[35] I. Mendez, B.W. Hansen, C.M. Grabow, E.J.L. Smedegaard, N.B. Skogberg, X.J. Uth, A. Bruhn, B. Geng, and E.N. Kamavuako. Evaluation of the Myo armband for the classification of hand motions. In *Proceedings of the International Conference on Rehabilitation Robotics*, pages 1211–1214. 2017.

[36] M. Ortiz-Catalan, R. Brånemark, and B. Håkansson. BioPatRec: A modular research platform for the control of artificial limbs based on pattern recognition algorithms. *Source Code for Biology and Medicine*, 8(1):11, 2013.

[37] M. Atzori, A. Gijsberts, I. Kuzborskij, S. Elsig, A.G. Mittaz Hager, O. Deriaz, C. Castellini, H. Muller, and B. Caputo. Characterization of a Benchmark Database for Myoelectric Movement Classification. *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, 23(1):73–83, 2015.

[38] W. Geng, Y. Du, W. Jin, W. Wei, Y. Hu, and J. Li. Gesture Recognition by Instantaneous Surface EMG Images [code], 2016. URL `http://zju-capg.org/myo/`.

[39] W. Geng, Y. Du, W. Jin, W. Wei, Y. Hu, and J. Li. Gesture Recognition by Instantaneous Surface EMG Images. *Scientific Reports*, 6:36571, 2016.

[40] B. Karlik, M.O. Tokhi, and M. Alci. A fuzzy clustering neural network architecture for multifunction upper-limb prosthesis. *IEEE Transactions on Biomedical Engineering*, 50(11):1255–1261, 2003.

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

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

[43] O. Fukuda, T. Tsuji, M. Kaneko, and A. Otsuka. A human-assisting manipulator teleoperated by EMG signals and arm motions. *IEEE Transactions on Robotics and Automation*, 19(2):210–222, 2003.

[44] S. Micera, A.M. Sabatini, and P. Dario. On automatic identification of upper-limb movements using small-sized training sets of EMG signals. *Medical Engineering and Physics*, 22(8):527–533, 2000.

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

[46] S. Pizzolato, L. Tagliapietra, M. Cognolato, M. Reggiani, H. Müller, and M. Atzori. Comparison of six electromyography acquisition setups on hand movement classification tasks. *PLOS ONE*, 12(10):e0186132, 2017.

[47] M. Atzori, M. Cognolato, and H. Müller. Deep learning with convolutional neural networks applied to electromyography data: A resource for the classification of movements for prosthetic hands. *Frontiers in Neurorobotics*, 10 (SEP):1–10, 2016.

[48] A. Gijsberts, M. Atzori, C. Castellini, H. Müller, and B. Caputo. Movement Error Rate for Evaluation of Machine Learning Methods for sEMG-Based Hand Movement Classification. *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, 22(4):735–744, 2014.

[49] X. Zhang, Y. Yang, X. Xu, and M. Zhang. Wavelet based neuro-fuzzy classification for EMG control. In *Proceedings of the International Conference on Communications, Circuits and Systems and West Sino Expositions*, volume 2, pages 1087–1089. 2002.

[50] A. Hartwell, V. Kadirkamanathan, and S.R. Anderson. Person-Specific Gesture Set Selection for Optimised Movement Classification from EMG Signals. In *Proceedings of the 38th Annual International Conference of the IEEE Engineering in Medicine and Biology Society*, pages 880–883, 2016.

[51] A. Hartwell, V. Kadirkamanathan, and S.R. Anderson. Compact Deep Neural Networks for Computationally Efficient Gesture Classification From Electromyography Signals. In *Proceedings of the 7th IEEE RAS/EMBS International Conference on Biomedical Robotics and Biomechatronics*, pages 891–896, 2018.

[52] A. Hartwell. Machine Learning for Hand Gesture Classification from Surface Electromyography Signals [code], 2018. URL `https://github.com/Lif3line`.

[53] A. Shuaib, A. Hartwell, E. Kiss-Toth, and M. Holcombe. Multi-Compartmentalisation in the MAPK Signalling Pathway Contributes to the Emergence of Oscillatory Behaviour and to Ultrasensitivity. *PLOS ONE*, 11 (5):e0156139, 2016.

[54] U. Jaramillo-Avila, A. Hartwell, and S. Anderson. Top-Down Bottom-Up Visual Saliency for Mobile Robots Using Deep Neural Networks and Task-Independent Feature Maps. In *Proceedings of the 19th Annual Conference: Towards Autonomous Robotic Systems*, volume 10965, page 489. 2018.

[55] M.B.I. Reaz, M.S. Hussain, and F. Mohd-Yasin. Techniques of EMG signal analysis: detection, processing, classification and applications. *Biological Procedures Online*, 8(1):11–35, 2006.

[56] R.H. Chowdhury, M.B.I. Reaz, M.A.B.M. Ali, A.A.A. Bakar, K. Chellappan, and T.G. Chang. Surface electromyography signal processing and classification techniques. *Sensors*, 13(9):12431–12466, 2013.

[57] A.A.B.A. Nadzri, S.A. Ahmad, M.H. Marhaban, and H. Jaafar. Characterization of surface electromyography using time domain features for determining hand motion and stages of contraction. *Australasian Physical and Engineering Sciences in Medicine*, 37(1):133–137, 2014.

[58] A.W. Preece, H.S. Wimalaratna, J.L. Green, E. Churchill, and H.M. Morgan. Non-invasive quantitative EMG. *Electromyography and Clinical Neurophysiology*, 34(2):81–86, 1994.

[59] N. Nazmi, M.A. Abdul Rahman, S.I. Yamamoto, S.A. Ahmad, H. Zamzuri, and S.A. Mazlan. A Review of Classification Techniques of EMG Signals during Isotonic and Isometric Contractions. *Sensors*, 16(8):1304, 2016.

[60] J.V. Basmajian and C.J. De Luca. *Muscles Alive - The Functions Revealed by Electromyography*. Williams & Wilkins, Baltimore, MD, 1985.

[61] W.M.B.W. Daud, A.B. Yahya, C.S. Horng, M.F. Sulaima, and R. Sudirman. Features extraction of electromyography signals in time domain on biceps Brachii muscle. *International Journal of Modeling and Optimization*, 3(6):515, 2013.

[62] M. Asghari Oskoei and H. Hu. Myoelectric control systems-A survey. *Biomedical Signal Processing and Control*, 2(4):275–294, 2007.

[63] P.A. DeLuca, K.J. Bell, and R.B. Davis. Using surface electrodes for the evaluation of the rectus femoris, vastus medialis and vastus lateralis muscles in children with cerebral palsy. *Gait and Posture*, 5(3):211–216, 1997.

[64] A. Van Boxtel. Optimal signal bandwidth for the recording of surface EMG activity of facial, jaw, oral, and neck muscles. *Psychophysiology*, 38(01):22–34, 2001.

[65] R. Merletti and P. Di Torino. Standards for reporting EMG data. Technical report, 1999.

[66] C.J. De Luca. *Surface Electromyography: Detection and Recording*. 2012.

[67] D.R. Rogers and D.T. MacIsaac. A comparison of EMG-based muscle fatigue assessments during dynamic contractions. *Journal of Electromyography and Kinesiology*, 23(5):1004–1011, 2013.

[68] D. Tkach, H. Huang, and T.A. Kuiken. Study of stability of time-domain features for electromyographic pattern recognition. *Journal of Neuroengineering and Rehabilitation*, 7:21, 2010.

[69] C.J. De Luca. The use of surface electromyography in biomechanics. *Journal of Applied Biomechanics*, 13:135–163, 1997.

[70] B. Gick, I. Wilson, and D. Derrick. *Articulatory Phonetics*. Wiley-Blackwell, 2013.

[71] H.S. Gasser and J. Erlanger. A study of the action currents of nerve with the cathode ray oscillograph. *American Journal of Physiology–Legacy Content*, 62 (3):496–524, 1922.

[72] C.D. Hardyck, L.F. Petrinovich, and D.W. Ellsworth. Feedback of speech muscle activity during silent reading: Rapid extinction. *Science*, 154(3755): 1467–1468, 1966.

[73] J.R. Cram and J.C. Steger. EMG scanning in the diagnosis of chronic pain. *Biofeedback and Self-Regulation*, 8(2):229–241, 1983.

[74] P.K. Artemiadis and K.J. Kyriakopoulos. EMG-based control of a robot arm using low-dimensional embeddings. *IEEE Transactions on Robotics*, 26(2):393–398, 2010.

[75] P. Marshall and B. Murphy. The validity and reliability of surface EMG to assess the neuromuscular response of the abdominal muscles to rapid limb movement. *Journal of Electromyography and Kinesiology*, 13(5):477–489, 2003.

[76] O. Fukuda, T. Tsuji, A. Ohtsuka, and M. Kaneko. EMG-based human-robot interface for rehabilitation aid. In *Proceedings of the IEEE International Conference on Robotics and Automation*, volume 4, pages 3492–3497. 1998.

[77] D. Nishikawa, W. Yu, H. Yokoi, and Y. Kakazu. EMG prosthetic hand controller using real-time learning method. In *Proceedings of the IEEE Conference on Systems, Man, and Cybernetics*, volume 1, pages 153–158. 1999.

[78] Thalmic Labs. Myo Armband, 2015. URL `https://www.myo.com/`.

[79] M.Z. Jamal. Signal acquisition using surface EMG and circuit design considerations for robotic prosthesis. In *Computational Intelligence in Electromyography Analysis-A Perspective on Current Applications and Future Challenges*. InTech, 2012.

[80] S. Day. Important factors in surface EMG measurement. *Bortec Biomedical Ltd*, pages 1–17, 2002.

[81] S. Lee and J. Kruse. Biopotential electrode sensors in ECG/EEG/EMG systems. *Analog Devices*, 200:1–2, 2008.

[82] H.F. Posada-Quintero, R.T. Rood, K. Burnham, J. Pennace, and K.H. Chon. Assessment of carbon/salt/adhesive electrodes for surface electromyography measurements. *IEEE Journal of Translational Engineering in Health and Medicine*, 4:1–9, 2016.

[83] B. Cesqui, P. Tropea, S. Micera, and H.I. Krebs. EMG-based pattern recognition approach in post stroke robot-aided rehabilitation: a feasibility study. *Journal of Neuroengineering and Rehabilitation*, 10(1):1, 2013.

[84] X.L. Hu, R.K.Y. Tong, N.S.K. Ho, J.J. Xue, W. Rong, and L.S.W. Li. Wrist rehabilitation assisted by an electromyography-driven neuromuscular electrical stimulation robot after stroke. *Neurorehabilitation and Neural Repair*, 2014.

[85] N. Kang, J. Idica, B. Amitoj, and J. Cauraugh. Motor recovery patterns in arm muscles: coupled bilateral training and neuromuscular stimulation. *Journal of Neuroengineering and Rehabilitation*, 11(57), 2014.

[86] P. Geethanjali. Myoelectric control of prosthetic hands: state-of-the-art review. *Medical Devices (Auckland, N.Z.)*, 9, 2016.

[87] K. Kiguchi and Y. Hayashi. An EMG-based control for an upper-limb power-assist exoskeleton robot. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 42(4):1064–1071, 2012.

[88] C. Ho-Seung, C. Wond-Du, and I. Chang-Hwan. A real-time lip gesture recognition system using facial EMG. In *Proceedings of the 38th Annual International Conference of the IEEE Engineering in Medicine and Biology Society*, 2016.

[89] H. Manabe, A. Hiraiwa, and T. Sugimura. Unvoiced speech recognition using EMG-mime speech recognition. In *Extended Abstracts on Human Factors in Computing Systems*, pages 794–795. 2003.

[90] P. Kugler, C. Jaremenko, J. Schlachetzki, J. Winkler, J. Klucken, and B. Eskofier. Automatic recognition of Parkinson's disease using surface electromyography during standardized gait tests. In *Proceedings of the 35th Annual International Conference of the IEEE Engineering in Medicine and Biology Society*, pages 5781–5784. 2013.

[91] J. Wu, Z. Tian, L. Sun, L. Estevez, and R. Jafari. Real-time American sign language recognition using wrist-worn motion and surface EMG sensors. In *Proceedings of the 12th International Conference on Wearable and Implantable Body Sensor Networks*, pages 1–6. 2015.

[92] C. Shirota. *Towards Effective Fall Prevention Mechanisms in Lower-Limb Prostheses: Trip Recovery in Transfemoral Amputees*. PhD thesis, Northwestern University, 2015.

[93] Delsys. Delsys Trigno Wireless System, 2017. URL `http://www.delsys.com/products/wireless-emg/trigno-lab/`.

[94] A.J. Jerri. The Shannon sampling theorem-Its various extensions and applications: A tutorial review. *Proceedings of the IEEE*, 65(11):1565–1596, 1977.

[95] F. Kerber, P. Lessel, and A. Krüger. Same-side hand interactions with arm-placed devices using EMG. In *Proceedings of the 33rd Annual ACM Conference Extended Abstracts on Human Factors in Computing Systems*, pages 1367–1372. 2015.

[96] M. Sathiyanarayanan and T. Mulling. Map Navigation Using Hand Gesture Recognition: A Case Study Using MYO Connector on Apple Maps. *Procedia Computer Science*, 58:50–57, 2015.

[97] K. Nymoen, M.R. Haugen, and A.R. Jensenius. Mumyo–Evaluating and Exploring the MYO Armband for Musical Interaction. In *Proceedings of the International Conference on New Interfaces For Musical Expression*. 2015.

[98] K.S. Krishnan, A. Saha, S. Ramachandran, and S. Kumar. Recognition of human arm gestures using Myo armband for the game of hand cricket. In *Proceedings of the IEEE International Symposium on Robotics and Intelligent Sensors*, pages 389–394. 2017.

[99] A. Phinyomark, C. Limsakul, and P. Phukpattaranont. A Novel Feature Extraction for Robust EMG Pattern Recognition. *Journal of Computing*, 1(1): 71–80, 2009.

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

[101] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, and M. Bernstein. Imagenet large scale visual recognition challenge. *International Journal of Computer Vision*, 115(3):211–252, 2015.

[102] Y. LeCun, Y. Bengio, G. Hinton, L. Y., B. Y., and H. G. Deep learning. *Nature*, 521(7553):436–444, 2015.

[103] NVIDIA. Jetson TX2 Module [Embedded Device], 2018. URL `https://developer.nvidia.com/embedded/buy/jetson-tx2`.

[104] A. Krizhevsky, I. Sutskever, and G.E. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems*, pages 1097–1105, 2012.

[105] D. Whitley and J.P. Watson. Complexity theory and the no free lunch theorem. In *Search Methodologies*, pages 317–339. Springer, 2005.

[106] C. Giraud-Carrier and F. Provost. Toward a justification of meta-learning: Is the no free lunch theorem a show-stopper. In *Proceedings of the ICML-2005 Workshop on Meta-learning*, pages 12–19, 2005.

[107] T. Lattimore and M. Hutter. No free lunch versus Occam's razor in supervised learning. In *Algorithmic Probability and Friends. Bayesian Prediction and Artificial Intelligence*, pages 223–235. Springer, 2013.

[108] A. Karpathy, F.F. Li, and J. Johnson. CS231n: Convolutional neural networks for visual recognition, 2016, 2017. URL `http://cs231n.github.io`.

[109] D.P. Bertsekas, D.P. Bertsekas, D.P. Bertsekas, and D.P. Bertsekas. *Dynamic Programming and Optimal Control*. Athena scientific Belmont, MA, 2005.

[110] N.M. Nasrabadi. Pattern recognition and machine learning. *Journal of Electronic Imaging*, 16(4):49901, 2007.

[111] M.A. Nielsen. *Neural networks and deep learning*. Determination Press, 2015.

[112] Y.E. Nesterov. A method for solving the convex programming problem with convergence rate O(1/k^2). In *Dokl. Akad. Nauk SSSR*, volume 269, pages 543–547, 1983.

[113] D.P. Kingma and J.L. Ba. Adam: a Method for Stochastic Optimization. *International Conference on Learning Representations*, pages 1–15, 2015.

[114] D.P. Bertsekas. *Nonlinear Programming*. Athena scientific Belmont, 1999.

[115] D.C. Liu and J. Nocedal. On the limited memory BFGS method for large scale optimization. *Mathematical Programming*, 45(1-3):503–528, 1989.

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

[117] N. Srivastava, G.E. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(1):1929–1958, 2014.

[118] X. Glorot and Y. Bengio. Understanding the difficulty of training deep feedforward neural networks. *Proceedings of the 13th International Conference on Artificial Intelligence and Statistics*, 9:249–256, 2010.

[119] V. Nair and G.E. Hinton. Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th International Conference on Machine Learning*, pages 807–814, 2010.

[120] G.E. Dahl, T.N. Sainath, and G.E. Hinton. Improving deep neural networks for LVCSR using rectified linear units and dropout. *ICASSP, IEEE International Conference on Acoustics, Speech and Signal Processing - Proceedings*, (2010): 8609–8613, 2013.

[121] J. Kim, J.K. Lee, and K.M. Lee. Deeply-Recursive Convolutional Network for Image Super-Resolution. *arXiv:1511.04491*, 2015.

[122] P. Rajpurkar, C. Bourn, A.Y. Ng, P. Cs, S. Edu, A.N.G. Cs, and S. Edu. Cardiologist-Level Arrhythmia Detection with Convolutional Neural Networks. *arXiv:1707.01836v1*, 2017.

[123] C. Doersch. Tutorial on Variational Autoencoders. *arXiv:1606.05908*, pages 1–23, 2016.

[124] P. Isola, J.Y. Zhu, T. Zhou, and A.A. Efros. Image-to-Image Translation with Conditional Adversarial Networks. *arXiv:1611.07004*, 2016.

[125] K. He and J. Sun. Convolutional neural networks at constrained time cost. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 07-12-June:5353–5360, 2015.

[126] A. van den Oord, S. Dieleman, H. Zen, K. Simonyan, O. Vinyals, A. Graves, N. Kalchbrenner, A. Senior, and K. Kavukcuoglu. WaveNet: A Generative Model for Raw Audio. *arXiv:1609.03499*, pages 1–15, 2016.

[127] A.S. Razavian, H. Azizpour, J. Sullivan, and S. Carlsson. CNN features off-the-shelf: An astounding baseline for recognition. *IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops*, pages 512–519, 2014.

[128] G. Huang, Z. Liu, K.Q. Weinberger, and L. van der Maaten. Densely Connected Convolutional Networks. *arXiv:1608.06993*, 2016.

[129] K. He, X. Zhang, S. Ren, and J. Sun. Identity Mappings in Deep Residual Networks Importance of Identity Skip Connections Usage of Activation Function Analysis of Pre-activation Structure. *arXiv:1603.05027*, (1):1–15, 2016.

[130] A.L. Maas, A.Y. Hannun, and A.Y. Ng. Rectifier nonlinearities improve neural network acoustic models. In *Proc. ICML*, volume 30, 2013.

[131] D.A. Clevert, T. Unterthiner, and S. Hochreiter. Fast and accurate deep network learning by exponential linear units (ELUs). *arXiv:1511.07289*, 2015.

[132] K. He, X. Zhang, S. Ren, and J. Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1026–1034, 2015.

[133] I.J. Goodfellow, D. Warde-Farley, M. Mirza, A. Courville, and Y. Bengio. Maxout networks. *arXiv:1302.4389*, 2013.

[134] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus. Intriguing properties of neural networks. *arXiv:1312.6199*, 2013.

[135] Y. LeCun, B. Boser, J.S. Denker, D. Henderson, R.E. Howard, W. Hubbard, and L.D. Jackel. Backpropagation applied to handwritten zip code recognition. *Neural Computation*, 1(4):541–551, 1989.

[136] A. Graves, A.R. Mohamed, and G. Hinton. Speech recognition with deep recurrent neural networks. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 6645–6649. 2013.

[137] S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural Computation*, 9(8):1735–1780, 1997.

[138] K. Cho, B. Van Merriënboer, D. Bahdanau, and Y. Bengio. On the properties of neural machine translation: Encoder-decoder approaches. *arXiv:1409.1259*, 2014.

[139] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv:1412.3555*, 2014.

[140] C. Castellini, E. Gruppioni, A. Davalli, and G. Sandini. Fine detection of grasp force and posture by amputees via surface electromyography. *Journal of Physiology-Paris*, 103(3):255–262, 2009.

[141] T.R. Farrell. A comparison of the effects of electrode implantation and targeting on pattern classification accuracy for prosthesis control. *IEEE Transactions on Biomedical Engineering*, 55(9):2198–2211, 2008.

[142] B. Crawford, K. Miller, P. Shenoy, and R. Rao. Real-time classification of electromyographic signals for robotic control. In *AAAI*, volume 5, pages 523–528, 2005.

[143] E.D. Engeberg and S. Meek. Improved grasp force sensitivity for prosthetic hands through force-derivative feedback. *IEEE Transactions on Biomedical Engineering*, 55(2):817–821, 2008.

[144] E.D. Engeberg, S.G. Meek, and M.A. Minor. Hybrid force-velocity sliding mode control of a prosthetic hand. *IEEE Transactions on Biomedical Engineering*, 55(5):1572–1581, 2008.

[145] H. Müller, J. Kalpathy-Cramer, I. Eggel, S. Bedrick, S. Radhouani, B. Bakke, C.E. Kahn, and W. Hersh. Overview of the CLEF 2009 medical image retrieval track. In *Workshop of the Cross-Language Evaluation Forum for European Languages*, pages 72–84. 2009.

[146] M. Atzori, A. Gijsberts, S. Heynen, A.g.M. Hager, O. Deriaz, P. Van Der Smagt, C. Castellini, B. Caputo, and M. Henning. Building the NINAPRO Database : A Resource for the Biorobotics Community. *Biomedical Robotics and Biomechatronics*, pages 1258 – 1265, 2012.

[147] F. Giordaniello, M. Cognolato, M. Graziani, A. Gijsberts, V. Gregori, G. Saetta, A.G.M. Hager, C. Tiengo, F. Bassetto, and P. Brugger. Megane Pro: myo-electricity, visual and gaze tracking data acquisitions to improve hand prosthetics. In *Proceedings of the International Conference on Rehabilitation Robotics*, pages 1148–1153. 2017.

[148] I. Kuzborskij, A. Gijsberts, and B. Caputo. On the Challenge of Classifying 52 Hand Movements from Surface Electromyography. *Proceedings of the Annual International Conference of the IEEE Engineering in Medicine and Biology Society*, pages 4931–4937, 2012.

[149] J. McIntosh, C. McNeill, M. Fraser, F. Kerber, M. Löchtefeld, and A. Krüger. EMPress: Practical hand gesture classification with wrist-mounted EMG and pressure sensing. In *Proceedings of the 2016 Conference on Human Factors in Computing Systems*, pages 2332–2342. 2016.

[150] Y. Hu, Y. Wong, W. Wei, Y. Du, M. Kankanhalli, and W. Geng. A novel attention-based hybrid CNN-RNN architecture for sEMG-based gesture recognition. *PLOS ONE*, 13(10):e0206049, 2018.

[151] Y. Xu, C. Yang, P. Liang, L. Zhao, and Z. Li. Development of a hybrid motion capture method using MYO armband with application to teleoperation. In *Proceedings of the IEEE International Conference on Mechatronics and Automation*, pages 1179–1184. 2016.

[152] C. Xiang, Z. Xu, C. Xiang, and W. Kong-qiao. Hand gesture recognition and virtual game control based on 3D accelerometer and EMG sensors Hand Gesture Recognition and Virtual Game Control Based on 3D Accelerometer

and EMG Sensors. *Proceedings of the 14th International Conference on Intelligent User Interfaces*, (Feb), 2014.

[153] Y. Fang, N. Hettiarachchi, D. Zhou, and H. Liu. Multi-modal Sensing Techniques for Interfacing Hand Prostheses : a Review. *IEEE Sensors Journal*, 15 (11):6065–6076, 2015.

[154] M.F. Lucas, A. Gaufriau, S. Pascual, C. Doncarli, and D. Farina. Multi-channel surface EMG classification using support vector machines and signal-based wavelet optimization. *Biomedical Signal Processing and Control*, 3 (2):169–174, 2008.

[155] L.R. Quitadamo, F. Cavrini, L. Sbernini, F. Riillo, L. Bianchi, S. Seri, and G. Saggio. Support vector machines to detect physiological patterns for EEG and EMG-based human-computer interaction: a review. *Journal of Neural Engineering*, 14(1):11001, 2017.

[156] F. Duan, L. Dai, W. Chang, Z. Chen, C. Zhu, and W. Li. sEMG-based identification of hand motion commands using wavelet neural network combined with discrete wavelet transform. *IEEE Transactions on Industrial Electronics*, 63(3):1923–1934, 2016.

[157] T. Baldacchino, W.R. Jacobs, S.R. Anderson, K. Worden, and J. Rowson. Simultaneous Force Regression and Movement Classification of Fingers via Surface EMG within a Unified Bayesian Framework. *Frontiers in Bioengineering and Biotechnology*, 6:13, 2018.

[158] A. Phinyomark, P. Phukpattaranont, and C. Limsakul. Feature reduction and selection for EMG signal classification. *Expert Systems with Applications*, 39(8):7420–7431, 2012.

[159] 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. *Frontiers in Neuroscience*, 11:379, 2017.

[160] M. Zia-ur Rehman, S.O. Gilani, A. Waris, I.K. Niazi, and E.N. Kamavuako. A novel approach for classification of hand movements using surface EMG signals. In *Proceedings of the IEEE International Symposium on Signal Processing and Information Technology*, pages 265–269. 2017.

[161] C. Olah and S. Carter. Attention and augmented recurrent neural networks. *Distill*, 1(9):e1, 2016.

[162] J. Demšar. Statistical Comparisons of Classifiers over Multiple Data Sets. *Journal of Machine Learning Research*, 7:1–30, 2006.

[163] F. Wilcoxon. Individual comparisons by ranking methods. *Biometrics*, 1(6): 80–83, 1945.

[164] F. Wilcoxon, S.K. Katti, and R.A. Wilcox. Critical values and probability levels for the Wilcoxon rank sum test and the Wilcoxon signed rank test. *Selected Tables in Mathematical Statistics*, 1:171–259, 1970.

[165] R.A. Fisher. *Statistical methods and scientific inference*. Hafner Publishing Co., New York, 1956.

[166] M. Friedman. The use of ranks to avoid the assumption of normality implicit in the analysis of variance. *Journal of the American Statistical Association*, 32 (200):675–701, 1937.

[167] R.L. Iman and J.M. Davenport. Approximations of the critical region of the fbietkan statistic. *Communications in Statistics-Theory and Methods*, 9(6):571–595, 1980.

[168] S. Holm. A simple sequentially rejective multiple test procedure. *Scandinavian Journal of Statistics*, pages 65–70, 1979.

[169] M.A. Hearst, S.T. Dumais, E. Osuna, J. Platt, and B. Scholkopf. Support vector machines. *IEEE Intelligent Systems and their Applications*, 13(4):18–28, 1998.

[170] C.J.C. Burges. A tutorial on support vector machines for pattern recognition. *Data Mining and Knowledge Discovery*, 2(2):121–167, 1998.

[171] Wikipedia. Kernel Mapping Visualisation, 2018. URL `https://en.wikipedia.org/wiki/Support{_}vector{_}machine{%}5C{#}/media/File:Kernel{_}Machine.svg`.

[172] L.R. Rabiner. A tutorial on hidden Markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–286, 1989.

[173] W.R. Gilks, S. Richardson, and D. Spiegelhalter. *Markov Chain Monte Carlo in Practice*. CRC press, 1995.

[174] Wikipedia. Hidden Markov Model Graph Example, 2009. URL `https://commons.wikimedia.org/wiki/File:HMMGraph.svg`.

[175] M. Sokolova and G. Lapalme. A systematic analysis of performance measures for classification tasks. *Information Processing and Management*, 45(4): 427–437, 2009.

[176] C.E. Metz. Basic principles of ROC analysis. In *Seminars in Nuclear Medicine*, volume 8, pages 283–298. 1978.

[177] H.B. Mann and D.R. Whitney. On a test of whether one of two random variables is stochastically larger than the other. *The Annals of Mathematical Statistics*, pages 50–60, 1947.

[178] N. Lachiche and P.A. Flach. Improving accuracy and cost of two-class and multi-class probabilistic classifiers using ROC curves. In *Proceedings of the 20th International Conference on Machine Learning*, pages 416–423, 2003.

[179] S. Micera, A.M. Sabatini, and P. Dario. An algorithm for detecting the onset of muscle contraction by EMG signal processing. *Medical Engineering and Physics*, 20(3):211–215, 1998.

[180] A. Hyvärinen, J. Hurri, and P.O. Hoyer. Principal components and whitening. In *Natural Image Statistics*, pages 93–130. Springer, 2009.

[181] O. Faust, Y. Hagiwara, T.J. Hong, O.S. Lih, and U.R. Acharya. Deep learning for healthcare applications based on physiological signals: a review. *Computer Methods and Programs in Biomedicine*, 2018.

[182] NinaPro Project. Ninapro Database Website, 2015. URL `http://ninapro.hevs.ch/`.

[183] R. Kohavi. A Study of Cross-Validation and Bootstrap for Accuracy Estimation and Model Selection. *Proceedings of the International Joint Conference on Artificial Intelligence*, 14(12):1137–1143, 1995.

[184] G. Forman and M. Scholz. Apples-to-apples in cross-validation studies. *ACM SIGKDD Explorations Newsletter*, 12(1):49, 2010.

[185] L.J.P. Van Der Maaten and G.E. Hinton. Visualizing high-dimensional data using t-sne. *Journal of Machine Learning Research*, 9:2579–2605, 2008.

[186] M. Wattenberg, F. Viégas, and I. Johnson. How to use t-sne effectively. *Distill*, 1(10):e2, 2016.

[187] M. Ortiz-Catalan, F. Rouhani, R. Brånemark, and B. Håkansson. Offline accuracy: a potentially misleading metric in myoelectric pattern recognition

for prosthetic control. In *Proceedings of the 37th Annual International Conference of the IEEE Engineering in Medicine and Biology Society*, pages 1140–1143. 2015.

[188] D.M.W. Powers. Evaluation: from precision, recall and F-measure to ROC, informedness, markedness and correlation. *Journal of Machine Learning Technologies*, 2(1):37–63, 2011.

[189] M. Mode, E. Journal, and O.F. Scientific. EMG Signal Classification for Human Computer Interaction : A Review. *European Journal of Scientific Research*, 33(3):480–501, 2009.

[190] M.R. Canal. Comparison of wavelet and short time Fourier transform methods in the analysis of EMG signals. *Journal of Medical Systems*, 34(1):91–94, 2010.

[191] T. Joachims. Text categorization with support vector machines: Learning with many relevant features. In *Proceedings of the European Conference on Machine Learning*, pages 137–142. 1998.

[192] N. Japkowicz and S. Stephen. The class imbalance problem: A systematic study. *Intelligent Data Analysis*, 6(5):429–449, 2002.

[193] H. He and E.a. Garcia. Learning from imbalanced data. *IEEE Transactions on Knowledge and Data Engineering*, 21(9):1263–1284, 2009.

[194] V. Ganganwar. An overview of classification algorithms for imbalanced datasets. *International Journal of Emerging Technology and Advanced Engineering*, 2(4):42–47, 2012.

[195] R. Blagus and L. Lusa. SMOTE for high-dimensional class-imbalanced data. *BMC Bioinformatics*, 14(1):106, 2013.

[196] F. Provost. Machine learning from imbalanced data sets 101. In *Proceedings of the AAAI'2000 Workshop on Imbalanced Data Sets*, 2000.

[197] N.V. Chawla, K.W. Bowyer, L.O. Hall, and W.P. Kegelmeyer. SMOTE: Synthetic Minority Over-sampling Technique. *Journal of Artificial Intelligence Research*, 16:321–357, 2002.

[198] S. Kullback and R.A. Leibler. On information and sufficiency. *The Annals of Mathematical Statistics*, 22(1):79–86, 1951.

[199] IEEE Computer Society. Standards Committee and American National Standards Institute. IEEE standard for binary floating-point arithmetic (IEEE 754), 1985.

[200] Z. Arief, I.A. Sulistijono, and R.A. Ardiansyah. Comparison of five time series EMG features extractions using Myo Armband. In *Proceedings of the International Electronics Symposium*, pages 11–14. 2015.

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

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

[203] T.R. Farrell and R.F. Weir. The optimal controller delay for myoelectric prostheses. *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, 15 (1):111–118, 2007.

[204] E. Scheme and K. Englehart. On the robustness of EMG features for pattern recognition based myoelectric control; a multi-dataset comparison. In *Proceedings of the 36th Annual International Conference of the IEEE Engineering in Medicine and Biology Society*, pages 650–653. 2014.

[205] F.N. Iandola, M.W. Moskewicz, K. Ashraf, S. Han, W.J. Dally, and K. Keutzer. SqueezeNet: AlexNet-Level Accuracy with 50x Fewer Parameters and <1MB Model Size. *arXiv:1602.07360*, 2016.

[206] R.T. Schirrmeister, J.T. Springenberg, L.D.J. Fiederer, M. Glasstetter, K. Eggensperger, M. Tangermann, F. Hutter, W. Burgard, and T. Ball. Deep learning with convolutional neural networks for brain mapping and decoding of movement-related information from the human EEG. *arXiv:1703.05051*, 2017.

[207] H. Sak, A. Senior, and F. Beaufays. Long short-term memory recurrent neural network architectures for large scale acoustic modeling. In *Proceedings of the 15th Annual Conference of the International Speech Communication Association*, 2014.

[208] J.T. Springenberg, A. Dosovitskiy, T. Brox, and M. Riedmiller. Striving for simplicity: The all convolutional net. *arXiv:1412.6806*, 2014.

[209] F. Chollet and others. Keras, 2015. URL `https://github.com/fchollet/keras`.

[210] S. Chetlur, C. Woolley, P. Vandermersch, J. Cohen, J. Tran, B. Catanzaro, and E. Shelhamer. cuDNN: Efficient primitives for deep learning. *arXiv:1410.0759*, 2014.

[211] A. Kurakin, I. Goodfellow, and S. Bengio. Adversarial examples in the physical world. *arXiv: 1607.02533*, 2016.

[212] NVIDIA. Jetson TX2 [Image], 2018. URL `https://www.nvidia.com/en-gb/autonomous-machines/embedded-systems-dev-kits-modules/`.

[213] E. Decleir. Using Myo during surgery with ADORA Assistant, 2016. URL `https://tinyurl.com/y48h6nzt`.

[214] M. Sathiyanarayanan and S. Rajan. MYO Armband for physiotherapy healthcare : A case study using gesture recognition application. In *Proceedings of the 8th International Conference on Communication Systems and Networks*, pages 1–6, 2016.

[215] J.G. Abreu, J.M. Teixeira, L.S. Figueiredo, and V. Teichrieb. Evaluating Sign Language Recognition Using the Myo Armband. In *Proceedings of the 18th Symposium on Virtual and Augmented Reality*, pages 64–70. 2016.

[216] K. Nymoen, M.R. Haugen, and A.R. Jensenius. Mumyo-evaluating and exploring the myo armband for musical interaction. In *Proceedings of the International Conference on New Interfaces For Musical Expression*. 2015.

[217] A. Ganiev, H.S. Shin, and K.h. Lee. Study on Virtual Control of a Robotic Arm via a Myo Armband for the Self- Manipulation of a Hand Amputee. *International Journal of Applied Engineering Research*, 11(2):973–4562, 2016.

[218] J. Ding, R.Z. Lin, and Z.Y. Lin. Service robot system with integration of wearable Myo armband for specialized hand gesture human-computer interfaces for people with disabilities with mobility problems. *Computers and Electrical Engineering*, 2018.

[219] F. Gaetani, G.A. Zappatore, P. Visconti, and P. Primiceri. Design of an Arduino-based platform interfaced by Bluetooth low energy with Myo armband for controlling an under-actuated transradial prosthesis. In *Proceedings of the International Conference on IC Design and Technology*, pages 185–188. 2018.

[220] J. Ngeo, T. Tamei, and T. Shibata. Estimation of continuous multi-DOF finger joint kinematics from surface EMG using a multi-output Gaussian process.

In *Proceedings of the 36th Annual International Conference of the IEEE Engineering in Medicine and Biology Society*, pages 3537–3540. 2014.

[221] C. Sapsanis, G. Georgoulas, A. Tzes, and D. Lymberopoulos. Improving EMG based classification of basic hand movements using EMD. In *Proceedings of the 35th Annual International Conference of the IEEE Engineering in Medicine and Biology Society*, pages 5754–5757. 2013.

[222] F.C.P. Sebelius, B.N. Rosen, and G.N. Lundborg. Refined myoelectric control in below-elbow amputees using artificial neural networks and a data glove. *The Journal of Hand Surgery*, 30(4):780–789, 2005.

[223] T. Feix, R. Pawlik, H.B. Schmiedmayer, J. Romero, and D. Kragic. A comprehensive grasp taxonomy. In *Robotics, Science and Systems: Workshop on Understanding the Human Hand for Advancing Robotic Manipulation*, 2009.

[224] M.R. Cutkosky. On grasp choice, grasp models, and the design of hands for manufacturing tasks. *IEEE Transactions on Robotics and Automation*, 5(3): 269–279, 1989.

[225] N. Kamakura, M. Matsuo, H. Ishii, F. Mitsuboshi, and Y. Miura. Patterns of static prehension in normal hands. *American Journal of Occupational Therapy*, 34(7):437–445, 1980.

[226] S.J. Edwards, D.J. Buckland, and J.D. McCoy-Powlen. *Developmental and Functional Hand Grasps*. Slack, 2002.

[227] L. Bottou. Early Stopping - but when? *Neural Networks: Tricks of the Trade*, 7700(1):421–436, 2012.

[228] P. Vincent, H. Larochelle, Y. Bengio, and P.A. Manzagol. Extracting and composing robust features with denoising autoencoders. In *Proceedings of the 25th International Conference on Machine Learning*, pages 1096–1103. 2008.

[229] H. Herr. State of the Art Prosthetics, Plenary Talk. In *Proceedings of the 7th IEEE RAS/EMBS International Conference on Biomedical Robotics and Biomechatronics*. 2018.

[230] S. Collins. Rapid Prototyping of Prosthetic Devices, Plenary Talk. In *Proceedings of the 7th IEEE RAS/EMBS International Conference on Biomedical Robotics and Biomechatronics*. 2018.

[231] M. Cognolato, M. Atzori, D. Faccio, C. Tiengo, F. Bassetto, R. Gassert, and H. Muller. Hand Gesture Classification in Transradial Amputees Using the

Myo Armband Classifier. In *Proceedings of the 7th IEEE RAS/EMBS International Conference on Biomedical Robotics and Biomechatronics*, 2018.

[232] M. Oquab, L. Bottou, I. Laptev, and J. Sivic. Learning and transferring mid-level image representations using convolutional neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1717–1724, 2014.

[233] L.A. Gatys, A.S. Ecker, and M. Bethge. Image style transfer using convolutional neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2414–2423, 2016.

[234] M. Long, H. Zhu, J. Wang, and M.I. Jordan. Deep transfer learning with joint adaptation networks. *arXiv:1605.06636*, 2016.

[235] Y. Ganin and V. Lempitsky. Unsupervised domain adaptation by backpropagation. *arXiv:1409.7495*, 2014.

[236] E. Tzeng, J. Hoffman, K. Saenko, and T. Darrell. Adversarial discriminative domain adaptation. In *Computer Vision and Pattern Recognition*, volume 1, 2017.

[237] V.P. Draglia, A.G. Tartakovsky, and V.V. Veeravalli. Multihypothesis sequential probability ratio tests. I. Asymptotic optimality. *IEEE Transactions on Information Theory*, 45(7):2448–2461, 1999.

[238] L.F. Nunes and K. Gurney. Multi-alternative decision-making with non-stationary inputs. *Royal Society Open Science*, 3(8):160376, 2016.

[239] J.A. Bilmes. A gentle tutorial of the EM algorithm and its application to parameter estimation for Gaussian mixture and hidden Markov models. *International Computer Science Institute*, 4(510):126, 1998.

[240] G.D. Forney. The Viterbi Algorithm. *Proceedings of the IEEE*, 61(3):268–278, 1973.

[241] Z. Liu, P. Luo, X. Wang, and X. Tang. Deep learning face attributes in the wild. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 3730–3738, 2015.

[242] W. Ge and Y. Yu. Borrowing treasures from the wealthy: Deep transfer learning through selective joint fine-tuning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, volume 6, 2017.