University of Southampton
Faculty of Physical Sciences and Engineering
Electronics and Computer Science

A Performance Comparison of Various Deep Learning Techniques on the Problem of Automatic Human Activity Recognition Via Wearable Sensors

by

Gregory Walsh

September 2018

Supervisor: Dr Kate Farrahi
Second Examiner: Professor Paul Lewin

A dissertation submitted in partial fulfilment of the degree of
MSc Data Science

Declaration of Authorship

- I have read and understood the ECS Academic Integrity information and the Universitys Academic Integrity Guidance for Students.
- I am aware that failure to act in accordance with the Regulations Governing Academic Integrity may lead to the imposition of penalties which, for the most serious cases, may include termination of programme.
- I consent to the University copying and distributing any or all of my work in any form and using third parties (who may be based outside the EU/EEA) to verify whether my work contains plagiarised material, and for quality assurance purposes.
- I have acknowledged all sources, and identified any content taken from elsewhere.
- I have used Python programming language and the open-source numpy, pandas, matplotlib, xlrd, scikit-learn, scipy, and pytorch python packages. Other than these, I have not used any resources, produced by anyone else.
- I did all the work myself, or with my allocated group, and have not helped anyone else.
- The material in the report is genuine, and I have included all my code and results data.
- I have not submitted any part of this work for another assessment.
- My work used secondary human activity recognition sensor data from anonymous human subjects (PAMAP2 dataset [1]). I received ethics approval from the ECS School Ethics Committee via the ERGOII system (Submission ID: 41273).

University of Southampton

Abstract

Faculty of Physical Sciences and Engineering
Electronics and Computer Science

MSc Data Science

by Gregory Walsh

While there is a growing body of research on the topic of human activity recognition (HAR) using deep learning techniques, as it stands, many of the results presented to date are based on biased sampling methods. As part of this project, the performances of four different deep learning architectures, on a HAR via wearable sensors dataset, were compared using an unbiased leave-one-subject-out cross-validation (LOSOCV) process. Little evidence was found to support the notion that any particular architecture dominates on performance. Therefore prior claims of the superiority of one method or another, made on the evaluation of data from a single test-subject and in the context of HAR datasets with small numbers of subjects, are disputed. Strong evidence was found, however, to suggest that Multi-layer perceptron (MLP) classifiers perform less well on average than other deep learning architectures.

This project also addresses the issue of an almost complete absence of research describing the effects that preprocessing data standardisation methods can have on model performance on HAR via wearable sensor problems. It is shown that by standardising input data on a per subject basis, rather than at the population level, one can considerably improve the generalisation capabilities across a range of architectures. Furthermore, a novel standardisation method is presented which significantly increases the performance of recurrent neural networks on atypical test subjects in comparison to min-max scaling.

Acknowledgements

I would like to thank Dr Kate Farrahi, my project supervisor, for all the helpful advice and encouragement she has given me throughout the course of this project, and Professor Paul Lewin for his time spent discussing this project. I would also like to thank Enrique Marquez for his patience and guidance on how to use the Iridis 5 supercomputer. Finally, I would like to thank all my lecturers, from whom I have learnt so much over the last year, thereby making it possible for me to tackle this project.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Human activity recognition (HAR) can be thought of as the process of automatically assigning labels, such as “running”, “sitting” or “falling”, using software, to samples of sensor data which capture some aspect of the physical movements and poses of an individual. The potential uses for HAR are broad. In the domain of healthcare, it has the potential to assist in various forms of rehabilitation [4, 5], and improve patient outcomes for vulnerable adults who are prone to falling by improving automatic fall detection [6]. With respect to public safety and security, HAR could be used to identify suspicious activities associated with criminal intent and alert security services who could then intercept, thereby possibly reducing crime [7]. Furthermore, HAR has already found numerous commercial applications by enabling novel modes of human-computer interaction with video game consoles and consumer virtual reality systems [8, 9].

1.1 Why Perform HAR Research?

In the recent literature, the main focus for researchers working on HAR problems has been to evaluate the performance capabilities of new deep learning techniques. Given the previously mentioned examples to which automatic HAR has already been applied, or in the future could be applied, it is not difficult to imagine the benefits of improved performance. More effective automatic fall detection systems, for example, would improve health outcomes for vulnerable people, since fewer falls would be missed, whilst at the same time reducing the costs associated with responding to false alarms. Improved detection rates would be preferable in safety and security applications for similar reasons. With respect to human-computer interaction, reducing the frequency of missed or incorrectly interpreted gestures could significantly improve the user experience associated with such systems.

Furthermore, it is possible that new effective data processing techniques or architectures, discovered during the course of HAR research, may be applicable to similar kinds of signal classification problems, such as speech-to-text.

1.2 Research Objectives

The principal aim of this project was to determine which deep learning architectures are most effective at performing HAR tasks by carrying out a rigorous performance comparison using a leave-one-subject-out cross-validation (LOSOCV) process, an approach not yet seen in the HAR deep learning literature. The selected architectures, which represent the top performing deep learning methods in the recent HAR via wearable sensors literature [3, 10–12], include multilayer perceptrons (MLPs), convolutional neural networks (CNNs), recurrent neural networks (RNNs), and hybrid convolutional-recurrent neural networks (CRNNs). With respect to the RNNs and the recurrent component of the CRNNs, two types of gated recurrent network unit are investigated, long short-term memory (LSTM) units and, not yet widely found in the HAR literature, gated recurrent units (GRUs).

In addition to investigating the performance of various deep learning architectures, this project has the aim of evaluating how different input data standardisation techniques, including novel techniques not previously seen in the HAR literature, influence the performance of HAR classifiers based on the previously mentioned deep learning architectures. All experiments were performed using the PAMAP2 dataset which contains the labelled sensor data from nine subjects [1].

Finally, this project was carried out in a way that would meet the author's personal goal of developing a more thorough understanding of deep learning methods, and experience in implementing and training them. Therefore, rather than reusing any code made available by previous researchers, all data processing, reporting, and construction of deep learning models was performed by the author himself.

1.3 Dissertation Structure

The remainder of this dissertation is structured in the typical format of a research report. Background information on HAR research and on deep learning techniques, models and best practices is presented in Chapter 2. In Chapter 3, experimental methods are described, alongside justifications for experimental design decisions. Following this, the results of the experiments, and discussion of those results is given in Chapter 4. Finally, in Chapter 5, a summary of the approach and findings are presented, along with potential opportunities for future research.

Chapter 2

Background Theory

This chapter serves to provide the reader with background information on HAR, and how the problem is defined. It also includes a thorough review of the implementation and performance of a range of HAR methods, as found in the literature, and highlights issues with current research.

2.1 Defining the HAR Problem

Given a time series of HAR sensor data (or multiple time series if multiple sensors have been used), in which instances of various activities are present, and in which each data point is labelled with the particular activity the subject was performing at that time, the most general formulation of the HAR problem is to generate a sequential list of correct activity predictions indexed to the time series. From this perspective, HAR is, in its purest form, a sequence-to-sequence problem [13]. In theory, a sequence-to-sequence model would be capable of continuously making predictions as new data becomes available. Other sequence-to-sequence problems include speech recognition, whereby audio data is converted into a sequence of words, and stock price prediction where a time series of historical prices is used to predict a sequence of future prices.

2.1.1 The Relaxed HAR problem

Unfortunately, sequence-to-sequence problems present an issue for researchers hoping to apply many of the popular and effective machine learning classification methods, since many of them require inputs of a fixed length, rather than a stream of data. Such methods are instead suited to classifying individual, independent samples of fixed dimensionality [14]. To make the problem compatible with these methods, the problem must first be reimagined as a simple multiclassification task. Indeed, almost all HAR researchers make such a modification [3, 10–12, 15–19].

To do so, typically the time series is first divided into separate windows of equal length, here on referred to as *samples* [13]. The class is then normally determined for each sample by either finding the data point label which occurs most frequently, or by selecting the label of the last data point in the sample. Having generated a set of labelled samples, all of equal length, the researchers are then free to apply classification methods that demand fixed sized inputs. The downside of such an approach is the loss of useful information that occurs when these samples are treated (as they are in most cases) as statistically independent observations. To illustrate the point, consider the fictional ordered sequence of samples $\{running, running, walking, walking, unknown, walking, walking\}$. Before even analysing the raw data, one can make a reasonable guess as to the label of the *unknown* activity, due to the surrounding information. Were the list not ordered like this, and instead the samples were presented in random order (as they are in the majority of the experiments described in the literature), it would be more difficult, and we would have no choice but to base our decision solely on the *unknown* sample's data.

With respect to how the samples are generated, a range of sample lengths have been experimented with in the literature, from as little as 0.08 seconds [20], up to 30 seconds [21]. Short sample lengths increase the computation required to train a model, since more samples are generated, however they bring the problem closer in spirit to the stricter sequence-to-sequence version, in a manner similar to how a higher frequency digital audio recording has more fidelity to its analogue source. Samples which are too short however, may fail to capture sufficient information, especially if the activity being performed is of a considerably longer length [22]. Having generated a collection of samples, they can then be divided into training and testing datasets, which can then be used to train and evaluate the performance of a classification method. Should the number of samples be too small to perform training effectively, the amount generated during the data extraction process can be increased by systematically overlapping the sample windows by some fixed proportion of their length.

Theoretically, the overlap between samples could be so extreme as to ensure that a prediction is made at every time step. This could be done either as the data is captured (real-time processing) or after all the data has been collected. However, given that the size of the time steps between capture events (typically 0.033-0.01s given sensor refresh rates of 33-100Hz [1, 17, 23]) is several orders of magnitude shorter than the duration of common human activities (which will typically last several if not many seconds), this may prove surplus to practical requirements [13].

More recently, with the development of faster hardware and algorithms capable of more effectively tackling sequence-to-sequence problems, specifically RNNs, this relaxing of the HAR problem as just described is not necessary. That said, researchers using RNNs continue to make the simplification for the purposes of comparing their results with those previously published [12], though not in every case, for example Hammerla et al. present results for a recurrent model trained in a sequential manner, one data point at a time, without extracting fixed length windows [3].

For the remainder of this document, the reader may assume any further mentions to the "HAR problem" refer to the relaxed version unless stated otherwise.

2.2 Evaluating HAR Models

A reasonable way to compare the performance of a set of machine learning methods is to perform k-fold cross-validation [24]. First, the dataset is randomly split into k partitions with an equal number of samples in each partition. The first of these partitions is then designated as the test set, and the other partitions are collectively used for training models. The process is then repeated with the second partition and so on until all partitions have been used as a test set. Test set results for each model can then be averaged over the k partitions and compared. This method of cross-validation will give a good indication of performance on new unseen data, so long as the training dataset evenly and thoroughly samples the latent distribution from which it is drawn, i.e. it is unbiased.

Unfortunately most HAR datasets do not thoroughly sample the latent distribution from which they are drawn (all people in the world) as they are typically generated using only a few human subjects, likely due to cost and time restrictions. This then causes a problem when used in conjunction with a k-fold cross-validation process to perform model comparison. To see this, imagine a k-fold cross validated model trained on the data from only a few people. With an appropriate model, the classification performance might be quite good for new data from these subjects, after all it was trained using their personal data. However, if data from a new person (who may be different physically or have a different level of mobility from the training subjects) is classified by this model, one would expect the accuracy of predictions to be lower. A fit young man tends to walk in quite a different manner to that of his elderly grandfather, and therefore the pair will produce quite different sensor readings for the same activity label. The likely decrease in performance would be a consequence of the tendency for models which are trained on a biased sets of data to overfit through the formulation of spurious and idiosyncratic decision rules [25].

Regrettably, in recent literature on HAR, especially that concerning the performance of deep learning techniques, several studies ignore this issue entirely, by both training and testing on data from all subjects [3, 10–12]. This limits the practical value of the research, because in order to maintain similar predictive capabilities, the models would need to be retrained each time a new person is added to the cohort. However, this is impractical for most applications due to the time and costs associated with the additional data collection, labelling, and computation that would be required for every new user.

Other times, machine learning methods are compared by training on a subset of the subjects (normally all but one), and then evaluating on the remaining subject [3, 12]. Whilst this is an improvement over the former approach, any claims made on this basis can only be said to hold for the particular division of subjects used in the experiments, rather than generally. An unscrupulous researcher could cherry-pick a particular test subject on which their approach performs well in order to make their research appear more impressive. By failing to perform a more rigorous method of analysis, such claims warrant further investigation.

As an remedy to this issue, different models can be compared using a LOSOCV process instead. As with k-fold validation, models are repeatedly evaluated on different partitions. In contrast to k-fold validation however, in a LOSOCV process, the data is partitioned by subject rather than randomly. This means machine learning methods are evaluated as thoroughly as possible given a limited number of subjects. A conceptually similar problem domain to HAR, to which LOSOCV approaches have been applied, is the unbiased evaluation of classification models trained on magnetic resonance image (MRI) data [26].

2.3 HAR Datasets

To collect data for HAR research, one or more human subjects are typically asked to perform a sequence of activities, during which sensors (e.g. cameras, wearable accelerometers, heart-rate monitors) are used to continuously capture information about their movements, poses, and occasionally their physiological condition [1, 7, 8, 17, 23, 27, 28]. During this data collection phase, the times at which one activity ends and another begins are accurately recorded. This activity log is then combined with the time series sensor data, with the result that each data point in the time series is then associated with a particular activity. If multiple sensors are used to capture information, then the data as a whole can be considered as a set of time series (one for each sensor channel) with time steps of a length that is inversely proportional to the frequency of the sensor updates.

2.3.1 Sensors used in HAR

The approaches to HAR data collection can be broadly divided into two paradigms depending on the type of sensors used. In the first, remote sensing devices such as video cameras and/or depth sensors are used to collect data on an individual's movements [7, 8, 27]. Since the sensors are fixed in position, the region of detection is limited to the range of the sensors, however, because the sensing is performed remotely, any individual within range can be monitored without the need for additional personal equipment. Therefore, this method is most suited to applications in which surveying individuals whilst they are in particular area, such as a bank lobby, is more important that continuous tracking of an individual regardless of their location.

In contrast, movement data can instead be collected using sensors worn on the person's body. This enables the data to be captured regardless of an individual's location, assuming that the sensors use some kind of portable memory. Such wearable devices often include sensors for physiological factors such as skin temperature and heart-rate, but most importantly, triaxial sensors such as accelerometers, gyroscopes, and magnetometers to measure the rate of change in velocity, rotational acceleration, and the strength of magnetic fields respectively [1, 17, 23, 28]. These triaxial sensors enable researchers to capture the inherently three dimensional characteristics of human movement and pose. Table 2.1 gives a breakdown of sensor modalities for a variety of popular HAR datasets.

2.3.2 Review of Datasets

Many HAR datasets have been published by the HAR research community and industry, and are available for public use. To select the one most suited to the experimental aims of this research, it was important to gather pertinent information on them so they could be easily compared. As a baseline, all datasets which do not contain data collected from wearable samples (for example video datasets) were immediately discounted from use. Furthermore, only datasets with time-stamps associated with sensor readings were considered.

The collection and labelling of human activity sensor data is a relatively manually intensive task, in comparison to say collecting and labelling images. Perhaps for this reason, most datasets contain activity data for a limited number of subjects, typically fewer than 10 (see Table 2.1). With regards to the types of activities captured, the range is quite broad. Several include household and sporting activities, but others have a much more specific focus, such as the Daphnet dataset [17], which was collected for the purpose of training machines to detect movements which are likely to instigate gait-freezing in people with from Parkinson's disease.

Table 2.1: Comparison of Popular HAR Datasets

Dataset	Recognition Purpose	# Classes	Worn Sensors	Environ. Sensors	Remote Sensors	# Subjects
OPPORTUNITY [23]	Household activities	13, 17, 15	Yes	Yes	No	4
Skoda Mini Checkpoint [28]	Workshop activities	10	Yes	No	No	1
Daphnet [17]	Detection of gait freezing	Binary	Yes	No	No	10
PAMAP2 [1]	Sporting/household activities	12	Yes	No	No	9
BodyAttack [29]	Fitness activities	6	Yes	No	No	1
UT-Interaction [30]	Human-human interactions	6	No	No	Yes	Several
Cornell Activity Dataset 120 [31]	Complex household activities	10, 10	No	No	Yes	4
HAR Using Smartphones [32]	Basic movement activities	6	Mobile Phone	No	No	30

2.4 Earlier Approaches to Performing HAR

Prior to the emergence of deep learning as a practical approach to solving HAR problems, a variety of machine learning classification methods have been evaluated across a range of datasets. The most significant of which, according to Lara and Labrador in their 2013 survey on HAR methods, are decision trees, ensemble learners, Bayesian methods, instance-based learning, and support vector machines (SVMs) [13]. Such methods have been applied to many other problems. Prior to the relatively recent application of deep learning techniques to HAR, SVM and instance-based methods (such as k-nearest neighbour) have been shown to be the most effective classifiers across multiple datasets [1, 33, 34].

2.4.1 Feature Engineering and its Issues

However, such methods cannot routinely be used with raw signal data, at least not in an effective manner (even when considering only the relaxed HAR problem - see section 2.1.1). To understand why, consider two observations of sensor data, one of which has been manually labelled as “walking”, and the other is unlabelled. We may wish to manually compare the “walking” observation with the unlabelled one with the goal of predicting whether or not the second observation was also a “walking” observation.

To do so, we could plot the two signals as a one-dimensional time series and visually compare them. In the case that both observations are “walking” observations, even if the signals are slightly out of phase or stretched relative to one another, we, as humans, are still able to identify common characteristic patterns in both observations, as seen in Figure 2.1.

We can do this due to the time ordered nature of the signal which provides additional visual information when we unconsciously process the values of each data point in the series relative to its neighbours, rather than independently. To see this is the case, imagine if instead, the data points for each observation were plotted in a random order. As a consequence there would be a significant loss of information thereby making the comparison more difficult. Unfortunately machine learning methods such as SVMs interpret their inputs in exactly this way, treating input variables as independent [14], meaning they are unable to account for the transpositions and scaling effects seen across observations [13].

As a workaround to this limitation, significant research has been carried out to find useful metrics, derived from the raw signal data, which are suitable for the purpose of

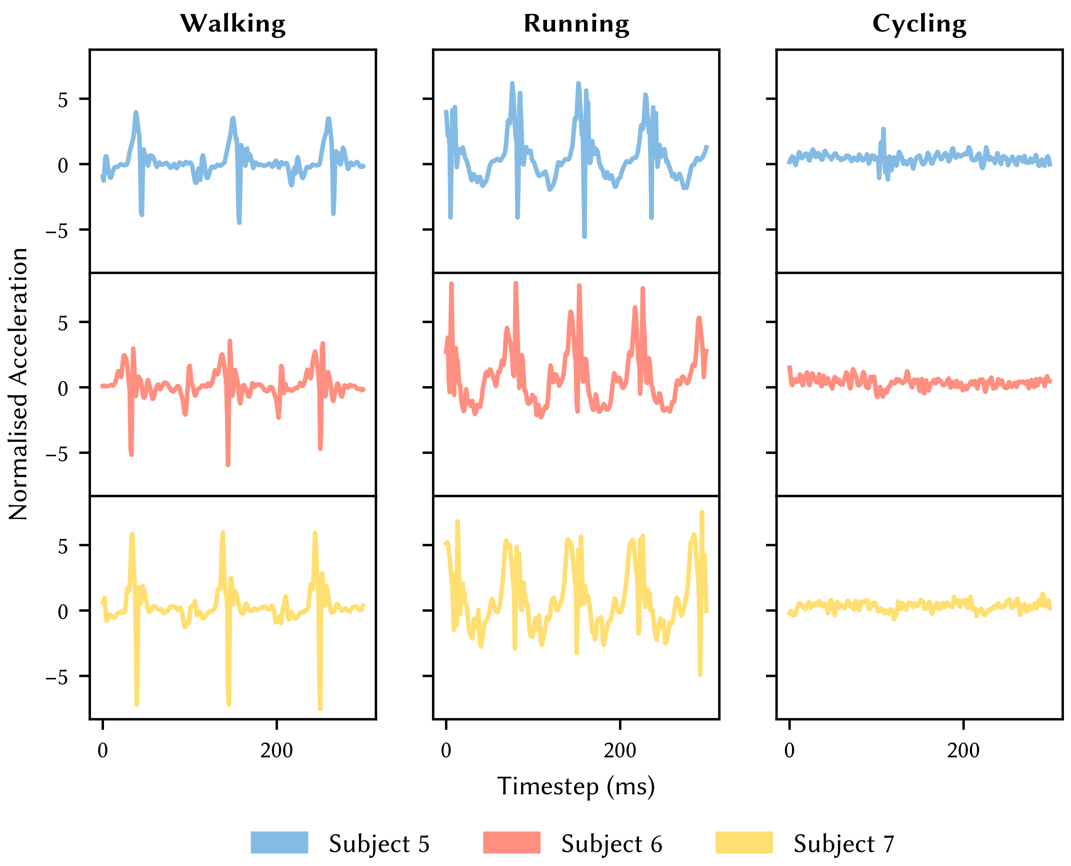


FIGURE 2.1: Samples of PAMAP2 ankle acceleration time series data (single channel shown) captured from three different subjects for a set of three different activities. Features could be generated from these samples by calculating statistics such as the mean or standard deviation over the individual sensor readings.

comparing observations in a process called feature engineering. Such engineered features vary from simple statistical measures such as mean and variance, to more complex features such as estimating the total energy associated with a particular accelerometer signal [35]. Transforming the signal from the time domain to the frequency domain is another approach to feature engineering [36], and is based on the assumption that sensor signals can be neatly decomposed into frequency bands. As a consequence, the domain expertise and resources required to develop and validate new and more discriminative features make such an approach a bottle-neck to HAR research when using the traditional classification algorithms mentioned earlier. Deep learning approaches to HAR mitigate this issue to a large extent, as discussed in Section 2.5, and thereby have opened up HAR research to a broader demographic.

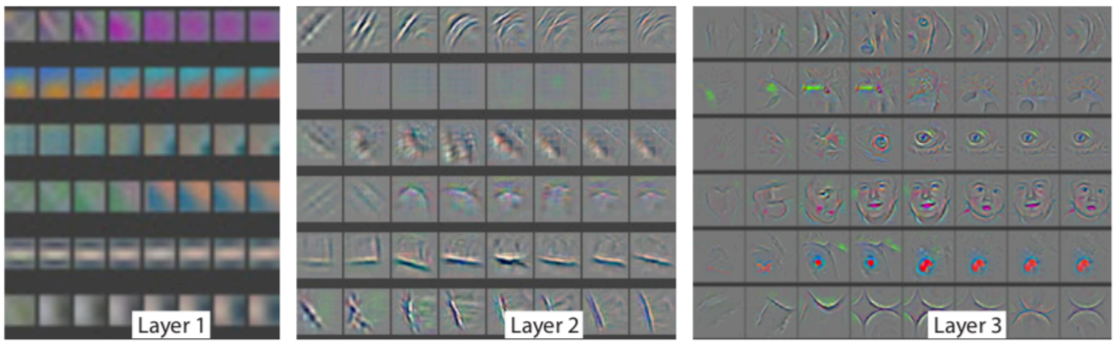


FIGURE 2.2: Visualisation of activation responses for different layers in a CNN, with several feature maps shown per layer. Reproduced in a cropped format from "Visualizing and Understanding Convolutional Networks" Lecun & Zeiler 2015 [2]

2.5 Deep Learning Approaches to HAR

Deep learning is a loosely defined term relating to the study and application of artificial neural networks (ANNs) comprised of multiple layers. In the broader literature, deep learning is typically considered from an architectural perspective and/or a conceptual-extraction one [37]. From the first viewpoint, deep learning models are sequential multilayer models, capable of extracting features in either a supervised or unsupervised setting, wherein each layer applies non-linear transformations to its inputs [38–40]. From the conceptual perspective, features (concepts) learned by the models are considered to be organised in a hierarchical fashion, where basic concepts learned in earlier layers are combined in later layers into more complex concepts [39–42]. This perspective is intuitively illustrated in Figure 2.2, where activations in the first layer are caused by simple gradients in the input, through to the activations in the third layer where neurons respond to more complex compound concepts, such as faces.

In contrast to algorithms such as SVMs, instance-based methods, and other non-deep learning approaches which require domain expertise to design useful features on which models can be trained, some deep learning models are innately capable of learning how to extract robust features, in other words, automatic feature extraction [2, 43]. Indeed, on a variety of HAR datasets, deep learning methods have been shown to outperform traditional classification algorithms paired with manual feature engineering [3, 10–12]. This is perhaps as a consequence of their ability to explore the space of possible features more broadly (and therefore more effectively) than can be practically achieved by an individual researcher or research group.

Given the performance benefits of using deep learning models over traditional approaches, coupled with the proliferation of hardware sufficiently powerful enough to

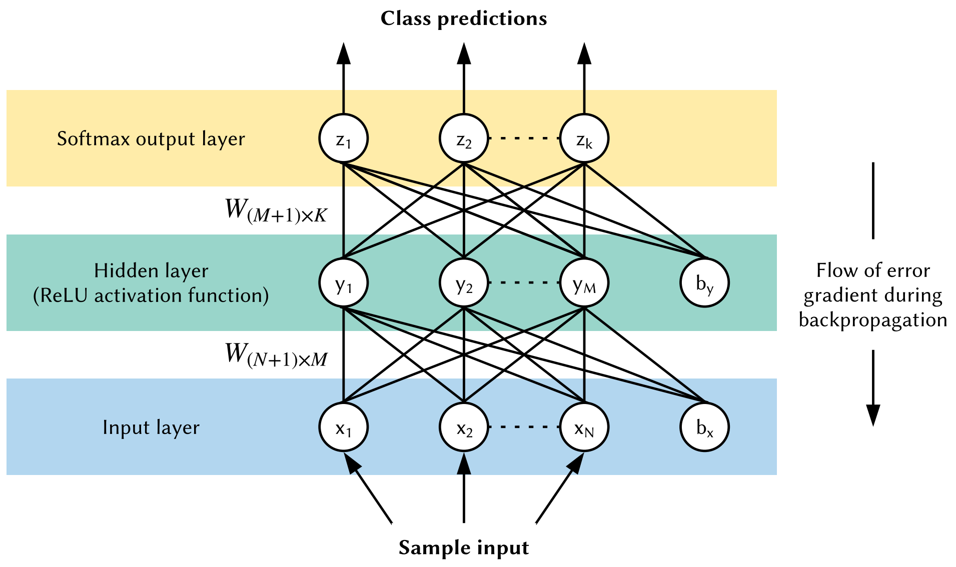


FIGURE 2.3: Schematic of a classification MLP with a single hidden layer. Elements b_x and b_y indicate biases for the hidden and output layers respectively. $W_{(N+1)\times M}$ and $W_{(M+1)\times k}$ represent weights encoded in matrix format.

train deep learning models in reasonable time, the evaluation of deep learning techniques has become much more prevalent in the recent HAR literature. The rest of this section is devoted to exploring the application of deep learning methods to the HAR via wearable sensors problem.

2.5.1 Multilayer Perceptrons

Theoretically speaking, an MLP with one hidden layer is a universal approximator, meaning it can approximate any smooth function acting on a bounded vector subspace to arbitrary precision, given a sufficient number of nodes in the hidden layer [44, p. 230]. They are a type of ANN, composed of an input layer where data is fed into the model, at least one hidden layer (each composed of several neurons/nodes) and an output layer from which useful information (such as the predicted class probabilities) is emitted given an input. Figure 2.3 shows the structure of such a network. Each node in the hidden layer and sometimes the output layer (depending on the model's purpose, e.g. classification), takes a linear combination the outputs of the previous layer's nodes (using predefined coefficients known as weights) and then applies a non-linear activation function to it, outputting the transformed value for use by the next layer or for use as a prediction as appropriate.

MLPs are, from an architectural perspective, one of simplest architectures in deep learning, and therefore one might expect that given their capability for automatic feature extraction one might expect to find multiple instances of results for them in the HAR literature. Whilst there are several examples from the previous decade or so [15, 16, 18, 19], in the last few years, only a single paper was found which presents results for MLPs, in which it functions solely as a performance comparison for more complex deep learning architectures [3].

In the older HAR papers, MLPs are occasionally evaluated on datasets with manually engineered features [15, 16, 18], even though MLPs have the capability to extract features automatically, possibly as a force of habit from other contemporary HAR research in which this approach was the norm at the time. In the recent paper by Hammerla et al., which also includes evaluation of MLPs, feature engineering was not performed [3]. Instead the readings from each time series sample are concatenated into a vector, and fed directly into the network. Even without feature engineering, the MLP models in their experiments perform quite competitively with the other deep learning techniques, even outperforming them in some instances. On this basis, there is motivation in further exploring the potential of MLPs for HAR problems, especially given the potential insights a cross-validation process might provide.

With respect to training and hyperparameter selection (hyperparameters being the parameters which define a model and its training process, but which are not modified during training, as for example the weights are), for HAR models in particular, Hammerla et al. recommend that a thorough search of the hyperparameter space be performed [3]. Their findings suggest that performance was particularly sensitive with respect to these properties. A breakdown of the type of parameter is also given which indicates that both learning hyperparameters (e.g. learning rate, weight decay) and architectural hyperparameters (e.g. number of layers in the network, number of neurons per layer) can play a significant role in performance. However, the relative importance of these two groups of hyperparameters were not equal for all of the datasets they tested, with learning parameters most influential on the PAMAP2 dataset, and architectural parameters more important when using the OPPORTUNITY dataset.

2.5.2 Convolutional Neural Networks

Like MLPs, CNNs are a layer-based ANN. They are designed to automatically extract useful features from n-dimensional raw signals, such as sensor time series data, images, and 3D MRI scans. What makes them effective at doing so, is their invariance to limited amounts of scaling and translations in their inputs [2].

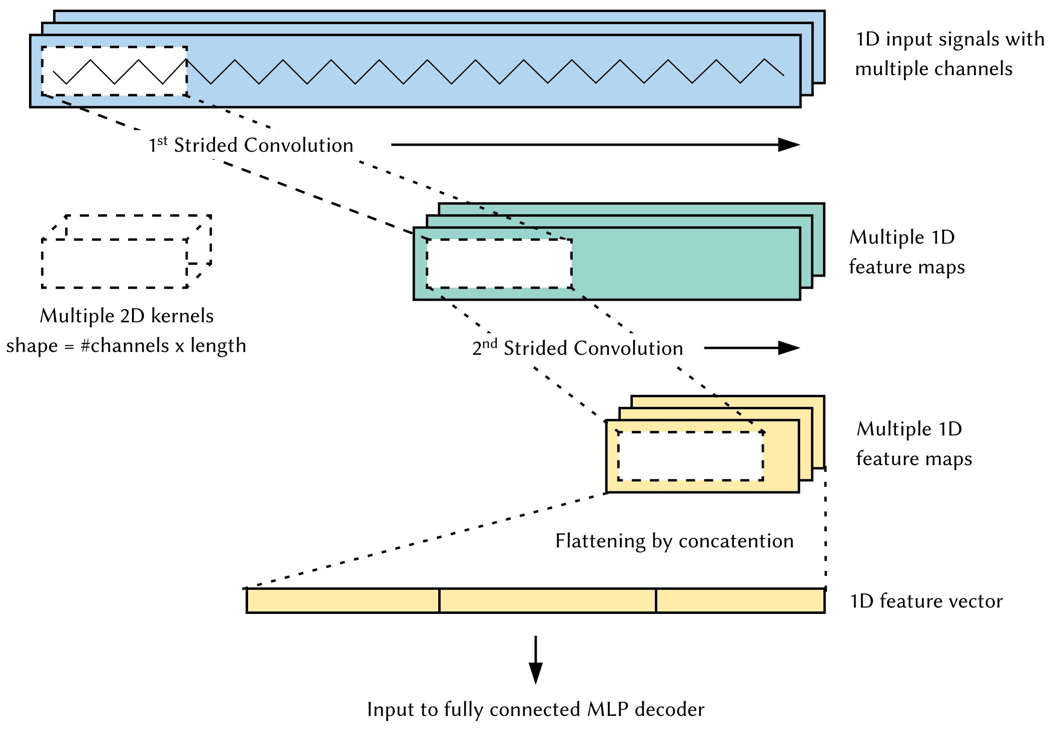

1D input signals with multiple channels
1st Strided Convolution
Multiple 2D kernels shape = #channels x length
Multiple 1D feature maps
2nd Strided Convolution
Multiple 1D feature maps
Flattening by concatention
1D feature vector
Input to fully connected MLP decoder

FIGURE 2.4: Schematic of an example multichannel 1D CNN, with two stages of convolution, representative of the architecture used in the project (ReLU activation not shown). In this example, a convolutional stride length of two is used, rather than pooling to reduce the size of the feature maps.

Figure 2.4 illustrates how information passes through a CNN with two layers of convolution. When an input is fed into a network, one or more kernels are convolved with the signal. The result of each convolution operation is a new feature map, which has dimensionality equal to the space/time dimensionality of the input to that layer. Optionally, the feature maps can be reduced in size by performing a pooling operation, or by using a stride length (the distance the kernel moves across the signal at each step of the convolution) of greater than one. Reducing the size of the feature maps with each subsequent layer can improve the generalisation performance of the network [45, 46]. Following the convolutional layers, either a linear output layer, or a shallow MLP can then be used to make the predictions.

Numerous recent papers on HAR examine the performance capabilities of CNNs [3, 10–12, 47–49]. In the domain of HAR via wearable sensors, the majority of researchers apply 1D CNNs to multichannel sensor data in a similar approach to that described in this section [3, 11, 12, 48, 49]. Within this group, variations on convolutional implementation exist, where some use a different set of kernels per sensor channel [11, 12, 48], while others imply the use of shared kernels [3, 49]. No comparison of these approaches is given.

Ronao et al. give an extensive analysis of the performance effects of various hyper-parameters [49]. They show that kernel length is an key parameter and that kernels should be of a length which corresponds to at least 0.18s of activity. This is empirically backed up by similar kernel lengths found in other HAR CNN research [3, 12]. Although most researchers add a pooling operation after each convolution, Ronao et al. found that pooling width does not significantly affect performance. This may indicate that the periodicity of signals tends to be similar enough between subjects that very little invariance to scaling is required. On a related note, recent more general research on CNNs indicates that using kernel strides greater than one (which produces smaller feature maps) works as well as pooling approaches for several problems, and is faster to compute (since there are half or fewer operations per layer) [46].

Additionally, Ronao et al. show that increasing the number of convolutional layers can significantly boost performance, with performance peaking at three layers on the HAR Using Smartphones dataset [49]. Further evidence for using several layers of convolutions can be found in an article published by Ordóñez et al., in which networks with four convolutional layers are shown to have the best performance on the OPPORTUNITY and Skoda Mini Checkpoint Datasets, though they do not present results for networks with more than four convolutional layers [12]. This phenomenon is to be expected, as deeper networks are capable of extracting more complex features that can then be used to differentiate between similar classes [2].

2.5.3 LSTM and GRU Recurrent Neural Networks

RNNs are designed with one-dimensional signal processing in mind and can be used in sequence-to-sequence regression and classification problems. In brief, they work by feeding the last output from a layer back into that layer along with the next external input at each time step. However, because these recurrent inputs are multiplied by a set of weights, which can be less or greater than one, with each time step, this leads to the issue of exploding and vanishing gradients during training [50]. These extreme gradients then cause numerical stability issues on fixed precision hardware, which limits effective training.

LSTMs are an evolution of the simple RNN, and are designed in part to solve the vanishing/exploding gradient problem [51]. Figure 2.5 illustrates a single layer for a common LSTM variant. The most important features are the persistent internal state $\mathbf{c}_t$ which stores information across multiple time steps, and the gates (sub-networks) which control the flow of information through the layer. As with simple RNNs, the previous outputs $\mathbf{y}_{t-1}$ from time $t-1$ are fed back into the network along with the next

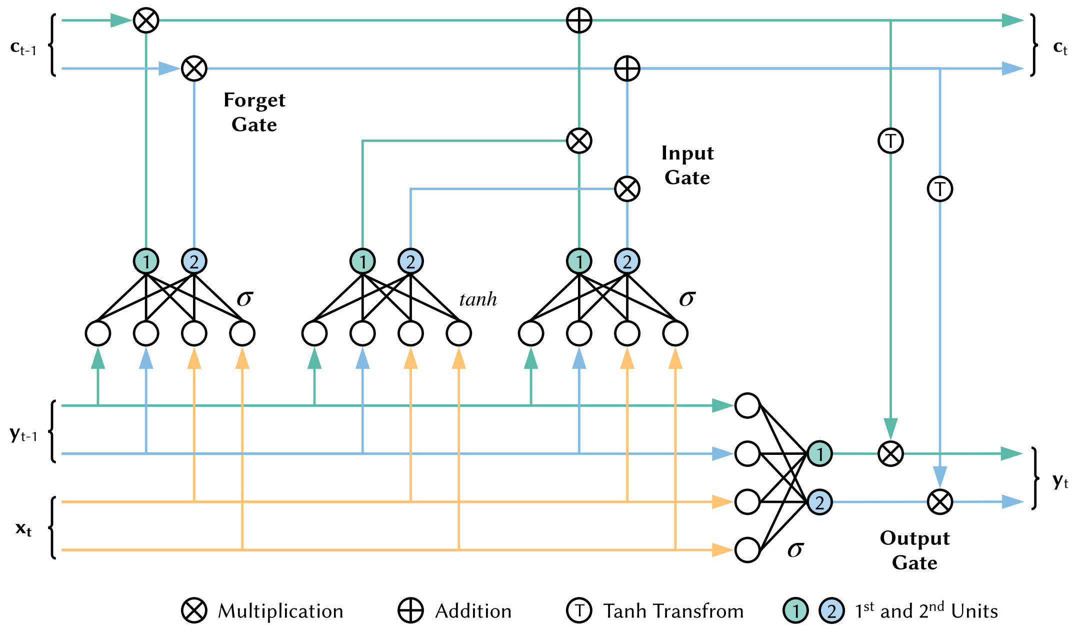


FIGURE 2.5: Schematic of a LSTM layer, simplified to show only two inputs and two internal units. Activation functions of internal sub-networks are shown adjacent.

input $\mathbf{x}_t$ at each time step. The forget gate allows the internal state to flow to later parts of the network unchanged, when necessary, preventing vanishing gradient problems. The input gate prevents the internal state from exploding by using an addition rather than multiplication operation on the internal state.

Because of the complexity of LSTMs (four fully connected sub-networks per layer), training can be slow. GRU networks are a simpler variant on the LSTM, which have only three sub-networks and no internal state as shown in Figure 2.6, which makes them faster to train. This is achieved by dropping the internal state $\mathbf{c}_t$ and using a single sub-network to perform the roles of the LSTM's input and output gate sub-networks. Updates to the recurrent inputs are then performed by a new gate, called the update gate, in two stages. The first stage controls how much of the previous output $\mathbf{h}_{t-1}$ will be output again (like the LSTM forget gate), and the second determines how much new information will included in the output (as with the LSTM input gate).

Given that LSTMs have an internal state, one might expect that LSTM networks may be better suited to learning long-term dependencies in comparison to GRUs, and therefore may perform better. However, empirical evaluations of these RNN variants suggest there is little performance difference [52].

Interestingly, despite the slower training times, only results for LSTMs are presented in the most recent research on HAR using RNNs [3, 12]. Hammerla et al., evaluate the performance of three different pure LSTM-RNNs designs [3]. These include two

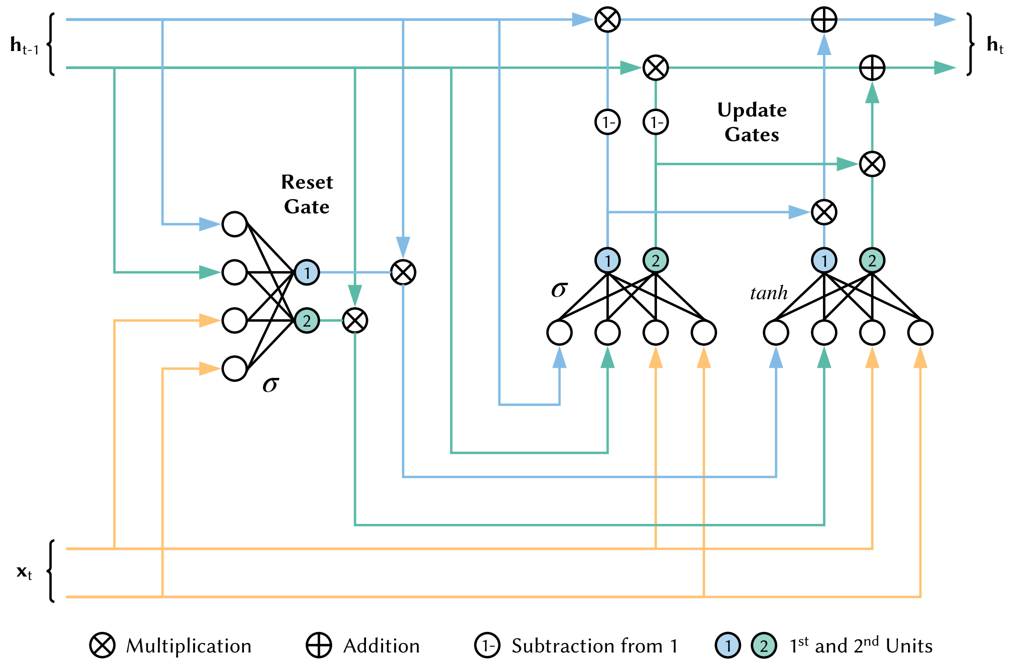


FIGURE 2.6: Schematic of a GRU layer, simplified to show only two inputs and two internal units. Activation functions of internal sub-networks are shown adjacent.

variants on typical multi-layer RNNs to which inputs are fed in sequentially, either one by one, or using a sliding window, and a single layer bi-directional RNN, made up of a pair of parallel LSTM sub-networks to which samples are input forwards and backwards respectively. The theorised benefit of a bi-directional network is that it may be able to improve class predictions by making use of future relevant information. For example, when performing automatic handwriting recognition, if the last few letters of a word are known with high certainty, this information can be used to narrow down what the earlier letters are likely to be.

Using these designs, Hammerla et al. achieve higher peak test scores than those previously published by other researchers, on two out of three datasets. However, peak test scores (wherein models are sorted by test performance, not validation performance) are a poor way to measure the expected performance one might achieve in a real-word scenario, since one cannot select a classifier based on its performance on unseen data.

Ordóñez et al. also present impressive results for a recurrent network, however, their version is augmented with convolutional layers before the double layered LSTM recurrent component (such a design is referred to here using the CRNN initialism) [12]. This CRNN outperforms their baseline CNN, but whether this is a consequence of the hybrid nature of the design, or simply because of the recurrent component, is not clear. It would have been useful for them to include an LSTM-RNN without convolutional layers as a second baseline for comparison.

2.6 Data Preprocessing and Standardisation Methods in HAR

Although manual feature engineering is no longer an essential aspect of solving HAR problems when using deep learning techniques, data preprocessing methods (such as the normalisation/standardisation of input data) can have a significant effect on training speed and model performance [53, p. 16-19]. Even so, little has been demonstrated in the HAR literature regarding the best data preprocessing methods for deep learning methods, meaning there is a significant area for potential research in this domain. Indeed, in the recent HAR literature only Ronao et al. share details on the standardisation applied, which was a population level normalisation wherein for each sensor the mean is subtracted from all readings, and all readings are then divided by the standard deviation [49]. This issue is compounded by the fact that the best data standardisation method may change depending on the deep learning architecture selected, a consequence of the interactions between activation functions, weights initialisation schemes and input distributions [53–55].

Furthermore, in HAR datasets, there are often significant differences in the intensities of sensor signals across different subjects, even when performing the same activity (see Figure 2.1. If nothing is done to correct for these inter-subject differences, it is likely to lead to overfitting during training, with poor performance on test subjects, especially if the test subject's signals have quite different intensities to those of the subjects used in training. Therefore, the evaluation of preprocessing methods, which are designed to reduce these inter-subject intensity differences, could lead to improved model performance across physically different subjects as a result of better generalisation.

Chapter 3

Methodology

3.1 Software Engineering and Hardware

The Python 3 programming language was selected for software development because of the range of open-source packages which facilitate data processing routines, and because of the availability of the PyTorch framework[1], a Python implementation of the Torch deep learning framework. Pytorch makes debugging quite straightforward, since the tensors which encode the inputs and outputs of layers can be easily inspected due to its use of dynamic computation graphs. Furthermore, it enables researchers to easily train and evaluate models on GPUs by interfacing with the CUDA API[2], which can decrease training time by over a factor of 10 in contrast to training on CPU [56].

With respect to hardware, the author was fortunate enough to be granted access to the university's supercomputer, specifically the Iridis 5 Lyceum GPU computing cluster. Students with access are granted the use of four Nvidia 1080TI GPUs, supported by 28 CPU cores and 128GB of memory. Use of this supercomputer enabled the aims of this project to be met, since without the ability to parallelise the experimental processing (thereby allowing for far more models to be trained and evaluated than would have been possible otherwise), a LOSOCV approach would have been unfeasible because of the need to repeat each set of experiments per subject. Figure 3.1 illustrates the combination of software and hardware used in the project.

[1] See `https://pytorch.org` for details of Pytorch.
[2] See `https://developer.nvidia.com/cuda-zone` for details of the CUDA API.

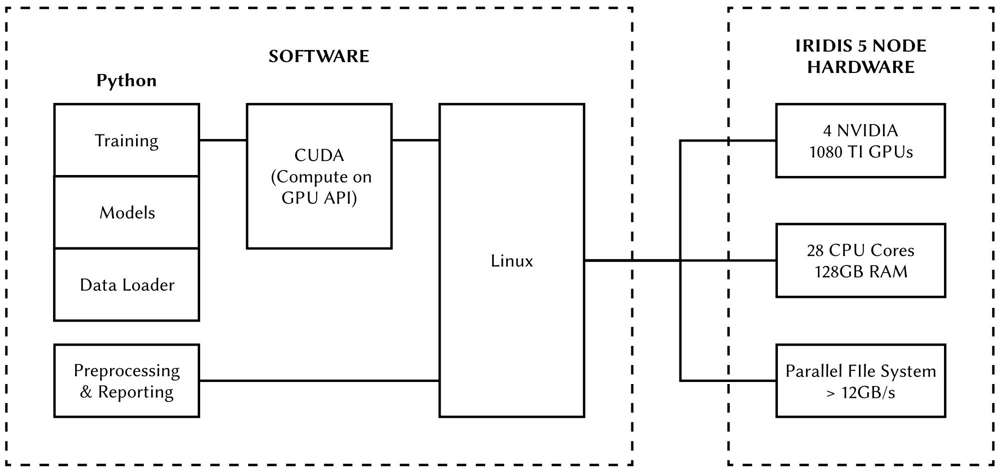


FIGURE 3.1: Software and hardware setup of the project. Hardware shown is for a single Iridis 5 "Lyceum" GPU compute node.

3.2 The Dataset

The PAMAP2 dataset [1] was selected for the purposes of this project for several reasons. It consists of data collected from nine subjects which makes the application of a LOSOCV process robust yet practical. This is in contrast to the more popular OPPORTUNITY dataset which contains data for only four subjects. Additionally, supplementary information about each subject was made available, which could (and did) prove useful during the analysis phase. It contains data captured across 12 different typical human activities, in comparison to other datasets with more specialised focus, such as the Daphnet dataset. Finally, the PAMAP2 dataset has recently been used in an evaluation of several deep learning architectures by Hammerla et al. [3], meaning their results could serve as a yardstick against which the results of this project could be measured.

During the data collection phase, subjects wore three inertial measurement units (IMUs), each hosting a variety of sensors, on the chest, dominant wrist, and dominant ankle. Of the nine subjects, one performed only a single activity for a short time and was therefore excluded from these experiments. The sensors used included accelerometers, gyroscopes, magnetometers, and orientation sensors. Furthermore, temperature, heart rate, and time stamp data were also recorded. Associated with each record in the dataset are subject and activity IDs.

Since Hammerla et al. are the most recent group to publish deep learning results using the PAMAP2 dataset, by and large, their approach to data extraction was replicated for the purposes of fair comparison [3], and they themselves replicate the approach of Reiss and Stricker who are the original authors of the dataset [1]. Therefore only the accelerometer sensors were used in making predictions and a matching sample length of 5.12 seconds was selected with a one-second overlap between adjacent samples.

3.3 Running Experiments

The main aims of the project were to carry out a performance comparison for four different deep learning architectures on HAR data using a LOSOCV approach and to evaluate the effects of a variety of data standardisation methods on the performance of these architectures. The decision to use a LOSOCV approach was made to determine if variations in performance across architectures and standardisation methods were consistent across all subjects.

To collect the required performance data, 250 differently parameterised models were trained and evaluated for each combination of architecture (see Section 3.4), standardisation method (see Section 3.8, and test subject), resulting in a total of 22,000 models trained and evaluated. Data from the remaining subjects was then used for training. However, before running all combinations of architectures, standardisation methods and test subjects, an initial evaluation phase was performed to determine if the models being trained were of acceptable performance. To do so, peak model performance for each combination of architecture and standardisation method was compared with the appropriate result presented by Hammerla et al. [3]. In line with Hammerla et al., subject 6 was used as the test set, and subject 5 as the validation (early stopping) set. On the basis of the results collected, the subject min-max scaling standardisation method for RNNs was dropped. Furthermore, only GRU RNNs were trained as part of the LOSOCV process, for practical reasons, since they achieved similar peak performance in comparison to LSTM units but were much faster to train. This allowed for a larger sample of models across all combinations of architecture and standardisation methods to be trained, given the finite computational resources available.

The architectural and learning hyperparameters for each new model were generated using the random search method. Random search, rather than grid search, was selected for practical reasons since evidence shows random search is likely to generate superior models given a fixed amount of computation [57]. Details of how the individual models were trained and evaluated can be found in Section 3.6.

Table 3.1: Architectural Hyperparameters

Hyperparameter	MLP	CNN	RNN	CRNN
Activation Func.	ReLU	ReLU	GRU, LSTM	ReLU[1]
Hidden Layers	1-3	1-4 + 1-3	1-2	1-4 + 1-2
Units per Layer[2]	64-1024	64-1024[3]	64-512	64-512 [4]
Feature Maps	N/A	16-256	N/A	16-256
Stride	N/A	1-3	N/A	1-3
Kernel Length	N/A	3-15	N/A	3-15
Batch Norm	Yes, No	Yes, No	No	Yes, No + No
Dropout	0, 0.25, 0.5, 0.9	0, 0.25, 0.5, 0.9 [3]	0, 0.25, 0.5, 0.9[5]	0, 0.25, 0.5, 0.9
Weights Init.	Kaiming	Kaiming	Xavier	Kaiming + Xavier

[1] Recurrent decoder uses either LSTM or GRU
[2] Increments in powers of two
[3] In the fully connected layers
[4] In the recurrent decoder
[5] When $>$ 1 layer, otherwise no dropout

3.4 Implementing the Deep Learning Architectures

In this section, details of how the different deep learning architectures were implemented is given. All models used a linear output layer with the logarithmic softmax function to make predictions, rather than plain softmax, due to its improved numerical stability and computational efficiency. Table 3.1 gives a summary of the architectural hyperparameter space explored by random search during the model generation and training process.

3.4.1 Multilayer Perceptrons

The MLP classifiers were implemented with between one and three hidden layers, using the ReLU activation function in all layers. Corresponding to the ReLU activation function, the weights were initialised using the Kaiming He scheme [55] to accelerate model convergence. For any given classifier, the number of units per hidden layer, which ranged from 64-1024, was kept constant. Since each sample consisted of nine different channels of accelerometer data, each of which contained 170 variables, the input vectors of each channel were concatenated into a single input vector containing 1,530 variables. Dropout was used, to various degrees, as a regularisation technique on the networks. In 50% of cases, batch normalisation was applied after the activation function in the hidden layers.

The MLP class was coded in such a way that it could also be easily used as a decoder in the CNN architecture.

3.4.2 Convolutional Neural Networks

The CNN classifiers consisted of a convolutional feature extraction component and a fully connected (i.e. MLP) decoder. The convolutional sections of classifiers were composed of between one and four convolutional layers, each with between 16 and 256 kernels. The ReLU activation function was applied in each layer. For any given model, the same number of kernels were used per convolutional layer. Two-dimensional kernels of odd length (3-15) and depth corresponding to the number of input channels/feature maps were used. As with the MLPs, weights were initialised using the Kaiming He scheme [55]. Pooling methods were not used to reduce feature size, instead kernel strides greater than one were tested since this approach is computationally far less expensive and produces similar results [46]. Dropout was also not applied in the convolutional layers either. Zero padding was applied to feature maps where appropriate, and weight sharing was used along the depth dimension of the kernel.

The fully connected decoder components of the models were parameterised in the same way as the MLP models, inputs to which were generated by concatenating the final set of feature maps from the convolutional component. Batch normalisation was applied to the encoder and decoder components with 50% probability.

Input samples were structured as two-dimensional tensors, corresponding to sample length and number of channels. The class of the convolutional component of the CNN was written in such a way that it could be reused in the CRNN models.

3.4.3 Recurrent Neural Networks

Two variations of gated recurrent neural networks were created, with LSTM and GRU units respectively. Inputs to these networks were organised as two-dimensional tensors, corresponding to sample length and number of channels. Networks were defined in a uni-directional manner, meaning that predictions are made by inputting the data points one at a time from earliest to latest. Since RNNs create an output for every time series input, only the final output of the model was taken as the prediction. The networks were created with between one and three hidden layers and an equal number of units per layer (between 64 and 512). In networks with more than one hidden layer, dropouts of varying intensity were used to regularise the networks. To initialise the networks, the Xavier Glorot weight initialisation scheme was selected [54]. Like the MLPs class, the RNN class was coded in a way that it could be used as a decoder in the CRNN models.

The CRNNs were generated by combining the convolutional component of the CNN models, with an RNN decoder. To keep things equivalent between architectures, the hyperparameter space explored for models in this architecture was the combination of the hyperparameter spaces of the CNN feature extraction component and the RNN architecture.

3.5 Measuring Performance

If the balance of classes within a dataset is even, accuracy (number of correct predictions divided by total predictions made) can be a reliable measure of a classifier's effectiveness. However, if there are significant class imbalances (as there are with the PAMAP2 dataset) accuracy can be a misleading measure of performance. To see this, imagine a hypothetical situation in which a dataset contains patient information and whether or not they have tested positive for a particular disease. It is our goal to build a classifier that can effectively predict if a new patient is likely to have this disease. Furthermore, imagine that the disease only affects 1% of patients. If we were to apply a classifier which *always predicts the negative class*, it will achieve an accuracy of 99% whilst failing to ever identify a single patient with the disease. In this instance, a metric which balances the precision and recall of the classifier will be more useful, where recall is given by

$$\text{recall} = \frac{\text{true positives}}{\text{true positives} + \text{false negatives}} \tag{3.1}$$

and precision is given by

$$\text{precision} = \frac{\text{true positives}}{\text{true positives} + \text{false positives}}. \tag{3.2}$$

F_1 score, the harmonic mean of precision and recall, is one such metric, and is defined as

$$F_1 = 2 \times \frac{\text{precision} \times \text{recall}}{\text{precision} + \text{recall}}. \tag{3.3}$$

In the event that there are multiple classes in a dataset (as with PAMAP2), the mean F_1 score across all classes (F_m) can be used to give an average estimation of a classifier's performance. Variations on this metric are common in the HAR literature [3, 11, 12, 58], but this particular version is used by Hammerla et al. in their evaluation of models on the PAMAP2 dataset. On this basis, it is used as the stopping metric for model training and in the performance comparisons given in Chapter 4.

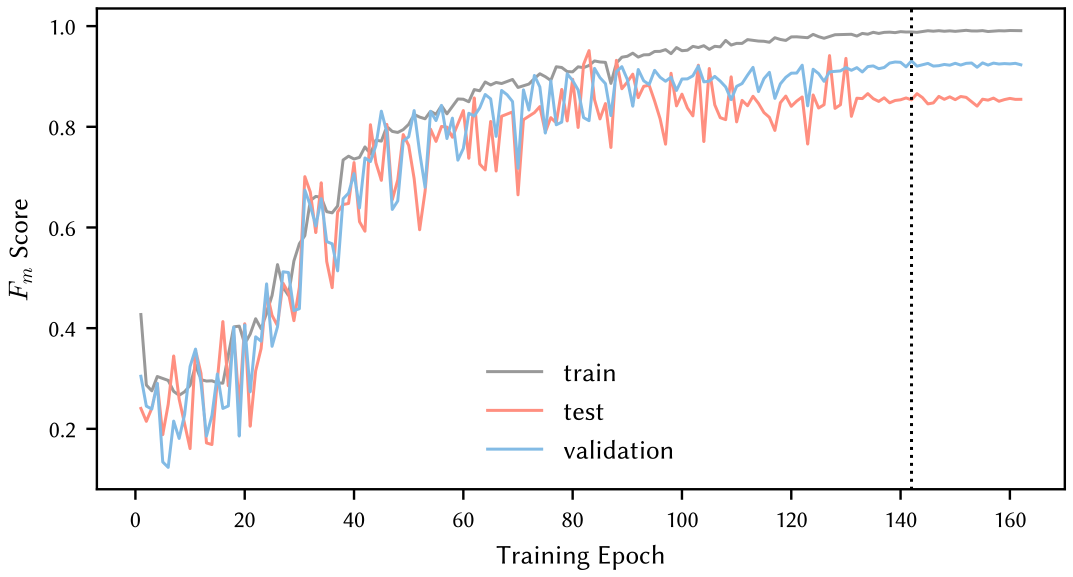


FIGURE 3.2: Training curve for a CRNN model with mean F_1 score metric (F_m). The test and validation set curves are quite noisy due to the relatively large initial learning rate (0.1). The dotted line indicates where the F_m metric peaks for the validation set and the point at which the test and validation F_m scores are captured.

3.6 Model Training

To get the highest expected performance on observations from the test subject, training must be stopped before the model overfits on the training data. Since the test subject's data must be treated as if it were unseen (to get a reasonable estimate of the model's performance on new data) it cannot be used to determine when to stop training. Instead, a proxy for the test set is required. This proxy is often called a validation or early stopping set (not to be confused with the term “validation” in LOSOCV, which refers to a validation process across subjects, not to the validation of individual models). Such validation sets were created by extracting all the data of a single subject from the training set. The subject used for validation was selected based on their ID being immediately previous to that of the test subject, looping round to 8 when subject 1's data was used for testing. This ensures that all subjects were used for validation, just as all subjects were used for testing.

The data from the remaining six subjects was then used for training. During training, if after 20 epochs a model showed no improvement in F_m score on the validation set, the training process was halted. Figure 3.2 shows this procedure in action. It can be seen that F_m score on the validation set peaked at epoch 142, and training then terminated at epoch 162 after no further improvement. Notice that epoch 142 does not correspond with the maximum F_m score of the test set, in line with treating it as unseen data. Models were trained for a minimum of 20 epochs and a maximum of 300.

As part of the process of hyperparameter exploration, for each model generated, learning hyperparameters were sampled from the values shown in Table 3.1. Mini-batch stochastic gradient descent was used to train the models, and the Nesterov momentum method was applied for parameterisations with non-zero momentum [59]. Weight decay was explored as a regularisation technique [60]. Half of the models were trained using a balanced sampler in an effort to improve the average performance over all classes in harmony with using the F_m score as the performance metric.

To promote faster learning in the initial training phase and slower learning later to fine-tune performance, the learning rate l was modified at the i^{th} epoch using the following annealing approach:

$$l_{i+1} = \frac{l_i}{1 + (i \times 10^{-2})} \tag{3.4}$$

.

TABLE 3.2: Learning Hyperparameters

Hyperparameter	Values
Momentum	0, 0.5, 0.9, 0.99
Learning Rate	0.1, 0.01, 0.001, 0.0001
Weight Decay	0, 0.01, 0.0001
Minibatch Size	16, 32, 64, 128
Balanced Sampling	Yes, No

3.7 Preprocessing the Data

In this section, the various stages of data preprocessing are described. Figure 3.3 illustrates the main stages involved in the process.

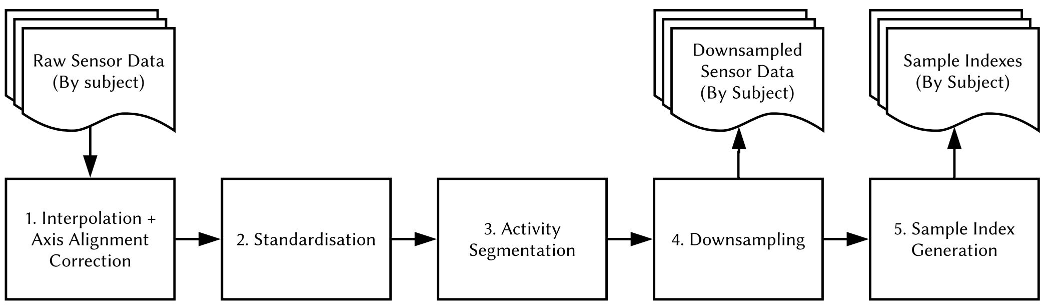

Raw Sensor Data (By subject)
Downsampled Sensor Data (By Subject)
Sample Indexes (By Subject)
1. Interpolation + Axis Alignment Correction
2. Standardisation
3. Activity Segmentation
4. Downsampling
5. Sample Index Generation

FIGURE 3.3: Flow chart indicating steps taken in the data preprocessing pipeline.

3.7.1 Interpolating Missing Data

The dataset suffers from a small quantity of missing information. Analysis showed that the longest contiguous set of records with missing data was 78 records, equivalent to less than one second's worth of activity. It was also noted that data loss was typically restricted to a single IMU unit, rather than all sensor data. For this reason, as an alternative to dropping records with missing data, linear interpolation was used to estimate the values. For a sequence of k missing items $(x_i)_{i=n}^{n+k}$, the value for the j^{th} missing value was computed using

$$x_j = x_i + j\frac{x_{n+k} - x_n}{k}. \tag{3.5}$$

3.7.2 Standardising Inputs

In line with the second aim of this project, that is, to evaluate the performance of a variety of data standardisation methods for different architectures, a variety of input data standardisation methods were applied and evaluated. See Section 3.8 for details on the particular standardisation methods which were tested.

3.7.3 Activity Segmentation and Signal Downsampling

The order, duration, and number of instances of activities varied between subjects. Therefore, before generating the sample indexes (i.e. the locations of the sample in the data stream), the continuous data-stream for each subject needed to be first partitioned into activity instances. The first and last ten seconds of each instance was discarded, in line with work by Reiss and Stricker [58] (Hamerla et al. do not mention this step [3]). Strictly replicating the approaches of both Reiss and Stricker and Hammerla et al, the data was then decimated to 33Hz.

3.7.4 Sample Index Generation

For models trained on GPUs, the speed at which data can be transferred from system memory to GPU memory during training can cause performance bottlenecks. Therefore where possible, all data ought to be moved into GPU memory before training. Given the memory requirements of the models (particularly the multilayered recurrent networks), fitting the entire dataset onto the GPU required a small amount of ingenuity. When samples were extracted one by one and stored as a collection (a typical approach), this incurred an additional 22% memory cost due to the overlap between samples, which resulted in frequent "out of memory" errors, causing training to fail. To rectify this, sample position indexes were calculated and a custom PyTorch data loader class was written for data retrieval.

3.8 Details of the Standardisation Methods

When standardising the dataset, two aspects of standardisation were considered. Firstly, standardisation of the data to ensure compatibility with the models, since expected performance and training speed are be greatly influenced by the interactions which occur between different weighting schemes, activation functions and the distribution of the inputs [44, p. 567][53, p. 16-19]. Secondly, standardisation for the purposes of reducing inter-subject differences in sensor reading distributions, with the goal of improved model generalisation. In this section, standardisation approaches designed to accomplish both aspects are referred to as "subject-level" methods, whilst approaches that do not factor into account inter-subject differences are referred to as "population-level" methods.

3.8.1 Standardisation Methods for MLPs, CNNs and CRNNs

LeCun et al. advise that input standardisation should be performed in a way that ensures outputs from the first hidden layer will have roughly unit variance [53, p. 16-19]. For the MLPs, CNNs, and convolutional layers of the CRNNs, given that ReLU was selected as the activation function, and the Kaiming He method was selected for the weights initialisation scheme, the standardisation methods were designed is such a way as to ensure a roughly normal distribution over each variable. In other words, given an $m \times n$ design matrix $\mathbf{X}$, where the i^{th} row X_{i*} is the data of the i^{th} sample, and the j^{th} column X_{*j} corresponds to the j^{th} input to the network, then the values of X_{*j} are approximately normally distributed for $1 \leq j \leq n$.

For MLPs, CNNs and CRNNs networks, three standardisation methods were evaluated, including one novel method. The first of which is well known method and is often simply referred to as normalisation [53]. It is a population-level method and is referred to in this document as *population-normalisation* to distinguish it from the other methods. It was performed by first calculating the mean sample vector $\bar{\mathbf{x}}$ and standard deviations for each variable encoded as the vector σ. Scaled values $\hat{X}_{i*}$ were then calculated as follows:

$$\hat{X}_{i*} = (X_{i*} - \bar{\mathbf{x}}) \oslash \sigma \tag{3.6}$$

where $\oslash$ is the element-wise Hadamard division operator. Data from the test subject was specifically excluded to prevent any information from the test dataset becoming available to the model during training, in order to simulate the realistic scenario in which activity

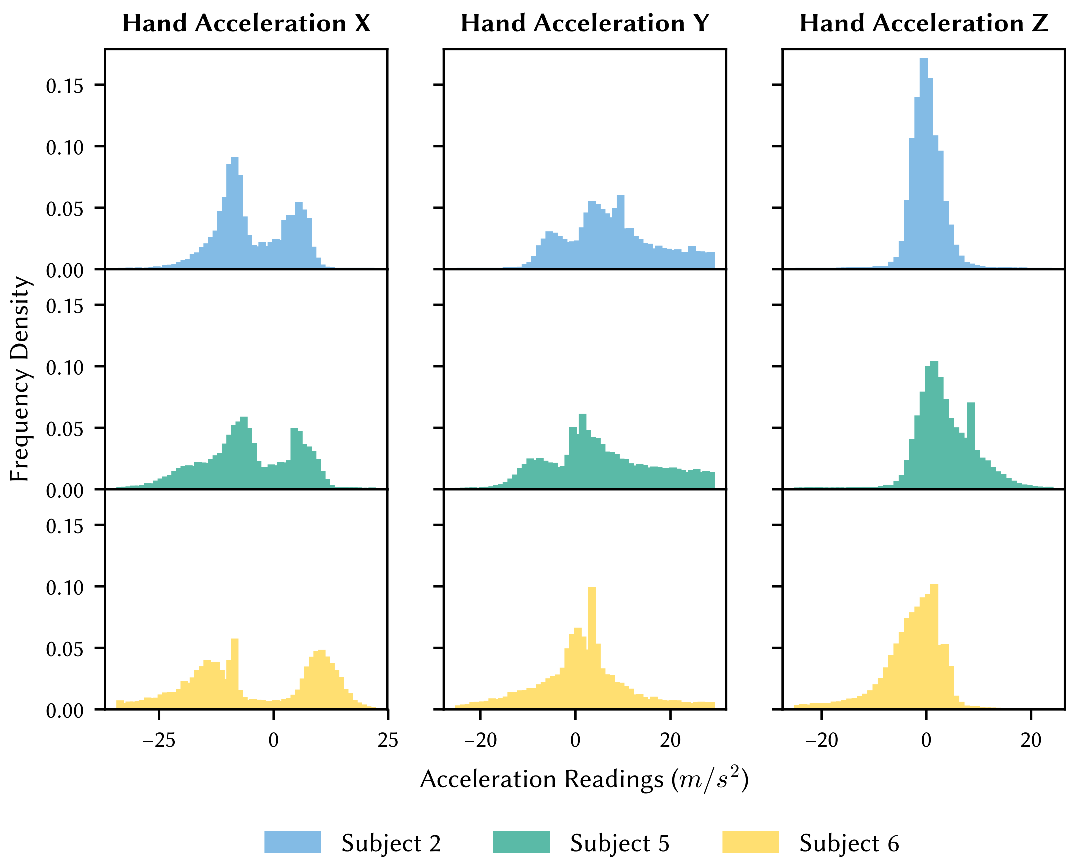


FIGURE 3.4: Frequency density histograms showing distributions of triaxial hand acceleration values captured from three subjects performing the running activity.

from new subjects would need to be classified after the model has already been trained. Failure to do so implies that a new model would be trained each time a new subject (or user in a commercial sense) joined, which would be extremely impractical if many users did so.

The second standardisation method, referred to as *subject-normalisation*, was performed in a similar manner to population-normalisation, but on a per subject basis, rather than for the entire population at once, thereby making it a subject-level method. As with the population-normalisation method, this ensures that no information about test subjects leaks to the model during the training process.

The novel third method, inspired by the Mann-Whitney U test[3] and also a per subject-level method, aims to reduce the idiosyncrasies in signal amplitudes between subjects. Inspecting Figure 3.4, we see that even when performing the same activity, distributions from the same sensor channel differ significantly between subjects. Furthermore, the distributions are clearly non-normal, differing by more than position and scale

[3]The Mann-Whitney U test is a non-parametric method for determining if the probability that one distribution is likely to generate larger values on average than another.

alone. Therefore affine transformations, as applied by the subject-normalisation and population-normalisation methods, cannot not fully align these distributions. The observed variation likely results from a combination of differences in body shape, weight, gait and mobility.

In an effort to better align the distributions, the data was first partitioned by user and sensor. For each partition, the values were then ranked using dense ranking to avoid gaps in the rank values. The ranks were then divided by the maximum rank, to get a set of values which lie on the half closed interval $(0, 1]$. Values very close to 0 or 1 were then clipped to prevent extreme values at the next step, in which the inverse of the cumulative density function for the normal distribution was used to map the data on the interval $(0, 1]$ to an ordered sequence of normally distributed values with mean zero and unit variance. This had the effect of aligning the values in each ranking percentile, including the median, from the training set to the same ranking percentile in the test set. This method is referred to in this project as *subject-normalisation-via-ranking*.

The subject-normalisation-via-ranking method also aims to reduce the effects of the extreme values in the dataset (not shown in 3.4), which may cause problems during training and testing. This is done in a convenient way for the practitioner since no knowledge of the distribution of the dataset is required, and there is no need to resort to extreme measures such as clipping the data beyond certain limits. The notable downside to this approach is the loss of information that is a consequence of throwing away the true values and using ranks in their stead (as also occurs in the Mann-Whitney U test).

3.8.2 Standardisation Methods for RNNs

Unlike the ReLU activation function used in the previously described MLP and CNN models, LSTM and GRU RNNs use a combination of sigmoidal and hyperbolic tangent (tanh) activations. These activation functions saturate (meaning that the gradient is close to 0) for inputs with absolute values much greater than 1. Hence, to prevent the saturation of the neurons in the first layer of the LSTM and GRU networks, each variable's values were mapped to the small closed interval $[-1, 1]$, as this corresponds to the majority of the region in which the gradient of the sigmoid function (as used in the gates) is greatest.

For recurrent networks, three standardisation methods were evaluated, including one novel method, following a similar pattern to those described in Section 3.8.1. Firstly, a common population-level min-max scaling method, referred to here as *population-min-max-scaling*, was performed as follows. For each input variable X_{*j}, scaled values $\hat{X}_{*j}$

were calculated as follows:

$$\hat{X}_{*j} = \frac{X_{*j} - \min(X_{*j})}{\max(X_{*j}) - \min(X_{*j})}. \tag{3.7}$$

In a similar vein to the population normalisation method described in Section 3.8.1, only training data was used in the calculations of $\min(X_{*j})$ and $\max(X_{*j})$ to prevent test information leaking into the model during training.

The second method for RNNs, is similar to the population-min-max-scaling method, but was applied on a subject by subject basis in an effort to better align the subjects' distributions and thereby improve the generalisation of the networks. This method is referred to as *subject-min-max-scaling* and is a subject-level method.

The novel third standardisation method for RNNs, is a subject-level method which follows the same process, and is motivated by the same argument, as that of the subject-normalisation-via-ranking method described in Section 3.8.1. The only difference being that the final step using the inverse CDF was not performed and instead the values of variables, currently on the interval $(0, 1]$, were mapped to the interval $(-1, 1]$ by multiplying by two and subtracting one. Like the subject-normalisation-via-ranking method, this method is also convenient to apply as one can deal with extreme values with minimal knowledge of the distributions. This method is referred to here as *subject-scaling-via-ranking*.

Chapter 4

Results & Discussion

In this chapter, the results of the experiments described in Chapter 3 are presented and interpreted. First, the results of the exploratory stage of experimentation are presented. Following this, results from the full LOSOCV process are presented, and an analysis and comparison of the performance of the different architectures and standardisation methods are provided.

4.1 Analysis of Initial Experiments

Table 4.1 shows the results from the initial peak performance evaluation phase, before the LOSOCV process was applied to all combinations of architectures and standardisation methods. For all combinations except two (LSTM and GRU RNNs combined with the subject-min-max-scaling method), peak test F_m scores exceed those presented by Hammerla et al. [3]. In all cases, Hammerla et. al. trained a greater number of machines than were trained for this project, therefore the higher peak scores seen here are not a result of this project having a larger sample size from which to draw results. During the experimental phase of the project, these intermediate results indicated that the architectures and data extraction processes had been implemented in an effective manner and that continuing on to a full LOSOCV using this code base was a reasonable progression.

These results also highlighted that the subject-min-max-scaling method for RNNs led to relatively poor test performance, which further testing on another test subject confirmed to be the case. One possible explanation is that the variation in minimum and maximum values across subjects was significant enough to increase the discrepancies between subjects' distributions, rather than decrease them, given that population-min-max-scaling

Table 4.1: Peak test F_m scores for models tested on subject 6, for comparison with reported results of Hammerla et al. [3]. Highest results per architecture are shown in bold. Early stopping performed using subject 5 in all cases.

Architecture	**Standardisation Method**	**Peak F_m**
MLP	Subject-normalisation-via-ranking	0.918
	Subject-normalisation	**0.934**
	Population-normalisation	0.915
	[Hammerla et al. 2016]	0.904
CNN	Subject-normalisation-via-ranking	0.963
	Subject-normalisation	**0.972**
	Population-normalisation	0.967
	[Hammerla et al. 2016]	0.937
CRNN	Subject-normalisation-via-ranking	0.954
	Subject-normalisation	**0.967**
	Population-normalisation	0.956
	[Hammerla et al. 2016]	N/A
RNN - LSTM	Subject-scaling-via-ranking	**0.953**
	Subject-min-max-scaling	0.538
	Population-min-max-scaling	0.938
	[Hammerla et al. 2016]	0.929
RNN - GRU	Subject-scaling-via-ranking	**0.949**
	Subject-min-max-scaling	0.534
	Population-min-max-scaling	0.936
	[Hammerla et al. 2016]	N/A

(in which every subject's data is transformed in exactly the same way) was far more effective. Given the poor performance, this method was not included as part of the full LOSOCV process.

Furthermore, the results indicated that the peak performance of RNNs with GRU units and LSTMs are very similar in this context. Given that the training times of LSTM RNN machines was considerably longer (between a factor of ten and forty) than those of the MLPs and CNNs and around double that of the GRU RNNs, the decision was made to perform the LOSOCV using only the GRU variety of RNN, which enabled more models for each combination of architecture and standardisation method to be trained, given the finite computational resources. This decision was taken in an effort to generate more accurate estimates of expected performance.

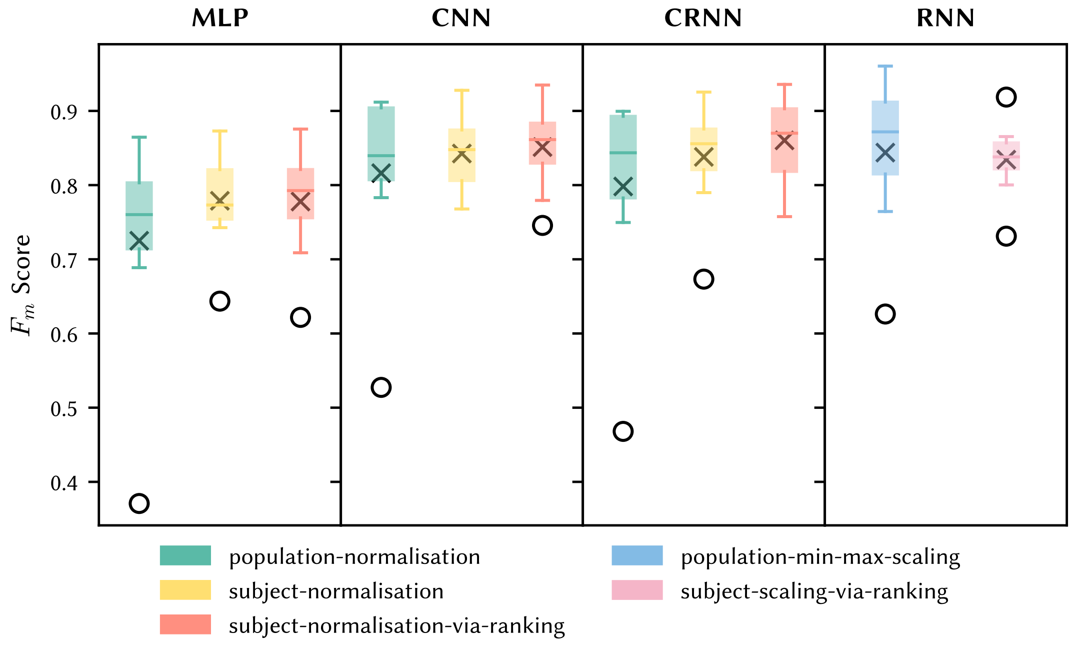
Table 4.2: LOSOCV results. For each combination of architecture, standardisation method, and test subject, the expected test F_m score (calculated using the top 10 performing machines ranked by validation F_m score) is shown. The best average performance across all test subjects is shown in bold.

Model	**Std Meth.**	**Expected Test F_m Score by Subject**								**Avg.**
		1	2	3	4	5	6	7	8	
MLP	SubNorm	0.758	0.808	0.644	0.743	0.873	0.86	0.78	0.767	0.779
	PopNorm	0.789	0.371	0.722	0.689	0.865	0.844	0.762	0.758	0.725
	SubNormVR	0.772	0.788	0.709	0.622	0.848	0.876	0.797	0.812	0.778
CNN	PopNorm	0.783	0.527	0.838	0.815	0.902	0.910	0.912	0.842	0.816
	SubNorm	0.826	0.869	0.768	0.777	0.886	0.928	0.87	0.816	0.843
	SubNormVR	0.846	0.849	0.746	0.779	0.905	0.935	0.874	0.876	0.851
CRNN	PopNorm	0.806	0.468	0.75	0.794	0.892	0.899	0.895	0.881	0.798
	SubNorm	0.848	0.863	0.673	0.790	0.908	0.925	0.865	0.831	0.838
	SubNormVR	0.846	0.821	0.758	0.809	0.921	0.936	0.894	0.897	**0.860**
RNN	PopMMS	0.849	0.626	0.832	0.764	0.912	0.895	0.960	0.912	0.844
	SubSVR	0.834	0.829	0.800	0.731	0.919	0.865	0.854	0.842	0.834

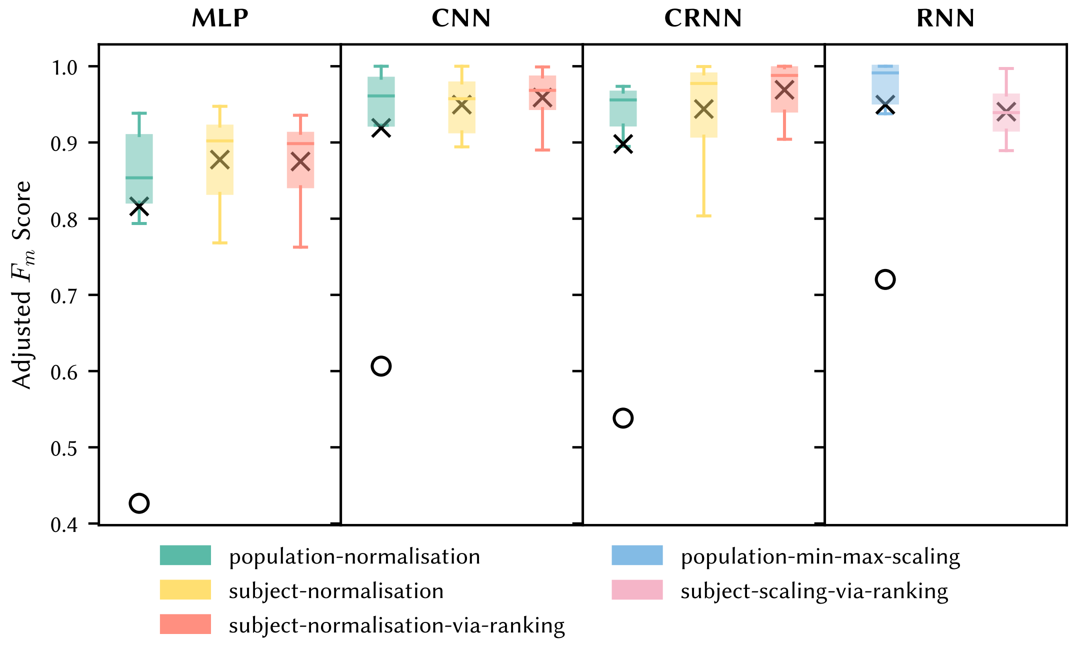
PopNorm: population-normalisation
SubNorm: subject-normalisation
SubNormVR: subject-normalisation-via-ranking (novel method)
PopMMS: population-min-max-scaling
SubSVR: subject-scaling-via-ranking (novel method)

4.2 Analysis of the Leave-One-Subject-Out Cross-Validation Results

The analysis in this section is presented from the perspective of someone who wishes to determine which combinations of architecture and standardisation methods will have the highest expected performance on an unseen test subject. As a comparison of methods, it is general in nature, in contrast to, for example, the comparison of individual instances of trained classifiers (machines) or even comparisons of particular hyperparameterisations. In alignment with this perspective, the performance metrics presented in Table 4.2, and those used in the creation of Figures 4.1(a) and 4.1(b), represent, for each combination of architecture and standardisation method, averages over the machines with the highest expected performance.



(a) Raw



(b) Adjusted

FIGURE 4.1: Distributions of LOSOCV results by architecture and standardisation method. Circles and crosses represent outliers and means respectively. Sub-figure (a) shows data taken from Table 4.2. Sub-figure (b) shows data which has been adjusted on a per subject basis in an effort to make the LOSOCV results more interpretable. The adjustment was performed by dividing all data points by the expected test F_m score of the highest performing combination of architecture and standardisation method for that subject.

4.2.1 Top Performing Architecture and Standardisation Method Combinations

Table 4.2 presents the distillation of the results from all 22,000 LOSOCV experiments. It shows, for each combination of architecture, standardisation method, and test subject, the average test F_m score for the top 10 performing machines, ranked by validation F_m score, belonging to that combination. Therefore, these values are not indicative of the peak test F_m scores each combination might be capable of producing, but are an estimate of the expected test F_m score one might get if one selects machines based on their performance on the validation set, as is typical for real-world machine learning applications. Results were averaged over the top 10 machines per combination because it was observed that significant variance in test F_m scores occurs for machines with very similar validation F_m scores (an average standard deviation of 0.054 F_m score corresponding to 6.6%).

From Table 4.2, we see that the CRNN architecture using the novel subject-normalisation-via-ranking method has the highest expected test F_m score when averaged over all test subjects at 0.860. In second place is the CNN architecture, also using the novel subject-normalisation-via-ranking method, with an expected test F_m score of 0.851. Estimations of expected results alone are, however, not sufficient to claim that one combination is superior to another. Confidence in a claim can sometimes be increased or decreased through the application of an appropriate statistical test for significance [61]. However, in the domain of machine learning such an approach is often unfeasible because of the huge amount of computational resources that would be required to generate sufficiently large sets of observations, especially considering that individual observations (results from a trained machine) can take minutes, hours or even days to generate.

As an alternative, one can look for consistency in a method's performance either across different problems or across different test sets, as one does when performing k-fold cross-validation. To evaluate the consistency with which the standardisation methods and architectures performed, the results for each test subject from Table 4.2 were visualised using a box plot. From this figure, we can ascertain two noteworthy pieces of information. Firstly, that population-level standardisation methods have more extreme, low scoring outliers than the subject level methods across all architectures. This indicates that population level standardisation methods appear to perform far worse on some subjects than others. This phenomenon is discussed in detail in Section 4.2.2. Secondly, that even after discounting the outliers, there is quite a range of performance within individual combinations of architecture and standardisation method. This is almost certainly because samples drawn from different test subjects are not identically distributed, i.e. not drawn from the same latent distribution.

Put another way, each subject has idiosyncrasies in their movement and posture which differentiate their samples in some way from other subjects. The larger these differences are between a test subject's sensor data distribution and those of the training group (see Figure 3.4), the worse one would expect the performance to be, particularly if no steps are taken to reduce the differences on a per subject basis (as is the case with the population-level methods).

In an effort to account for this variation in performance across test subjects, and thereby make comparisons across combinations of architecture and standardisation method simpler, a set of adjusted scores were calculated. This adjustment was performed per test subject, whereby the original values (as shown in Table 4.2) were divided by the maximum value that was achieved on that test subject (across all combinations). These adjusted scores are shown in Figure 4.1(b). From this figure, we can see that the variance in performance per combination is reduced as intended, but not to the extent to which comparing combination becomes completely trivial. For this LOSOCV process, it would seem that the noise introduced by the differences between subjects is too large to compensate for given the small difference in performance across combinations of architecture and standardisation method.

Reviewing Figure 4.1(b), it is apparent that the MLP combinations perform worse than those of the other architectures, which is in line with the results presented by Hammerla et al. [3]. This is likely because CNNs and RNNs have an invariance to translations and scaling in their inputs which MLPs do not have. There is, however, no clear top performer given the overlap of the inter-quartile ranges of the CNN, CRNN, and RNN combinations.

4.2.2 Comparison of Standardisation Methods

Although the top performing combination of architecture and standardisation method cannot be identified with a high degree of certainty, it is possible to infer some probable properties of the various standardisation methods, by reviewing their performance across architectures. For example, with regard to the MLP, CNN and CRNN architectures, the average performance across all test subjects when using the population-normalisation method is consistently lower in comparison to when the novel subject-normalisation-via-ranking method is used (-6.8%, -4.1%, and -7.2% respectively). Reviewing the results per test subject, we see that for most subjects, the scores are similar across the different combinations, however, the scores on test subject 2 are far lower for the population-normalised combinations, consistently so by an average of 44.6%.

This atypical result may be explained by the physical differences between subject 2 and the rest of the group. According to the supplementary information provided by with the PAMAP2 dataset [1], she is the only female in the group, she weighs 3.25 kg less than the average male subject and is 11.75 cm shorter. It is, on this basis, conceivable that the sensor readings measuring her movements are in some ways different to those of the other subjects (indeed it can be seen that this is the case from Figure 3.4). It then follows that if no efforts have been taken to correct for such differences (as is the case when a population-level standardisation method, rather than a subject-level method, is applied), that the training and validation (early stopping) data sets are likely to be less representative of this test set, thereby leading to poorer performance on this subject.

On this basis, it appears that the subject level standardisation methods, including the novel standardisation methods (subject-scaling-via-ranking and subject-normalisation-via-ranking) are, regardless of the architecture, useful for training models which will be used to classify data of subjects who are dissimilar to the training set. In the case of the MLP, CNN, and CRNN architectures it appears such methods improve performance by an average of 5.7% over the population level methods, with the novel subject-normalisation-via-ranking method having perhaps a slight edge for CNN and CRNN architectures. It would seem therefore that any detrimental loss of useful information which occurs as part of the subject-normalisation-via-ranking method, is balanced by the benefits of more effectively aligned distributions across subjects.

However, this is not to say that using such methods will always lead to a higher expected performance. In contrast to the other architectures, when the subject-scaling-via-ranking method is used with RNNs it appears to perform on average very slightly worse than the population min-max scaling method (-1%), however it does so with the benefit of avoiding extreme outliers (see Table 4.2 which shows an improvement in expected F_m score for subject 2 from 0.629 to 0.829, a change of 32%). This may be a consequence of the slight saturation of the GRU's activation functions in the input layer since each channel values are distributed uniformly from [-1, 1], which does not occur when using the population-min-max-scaling method.

Further evidence that subject-level methods led to better generalisation can be found in Figure 4.2, which shows the distributions of test F_m scores for all 22,000 trained and evaluated machines, grouped by architecture and standardisation method. For all architectures, it is apparent that distributions are skewed towards higher performance for the subject-level standardisation methods in comparison to the population-level standardisation methods, especially with respect to the RNN architecture.

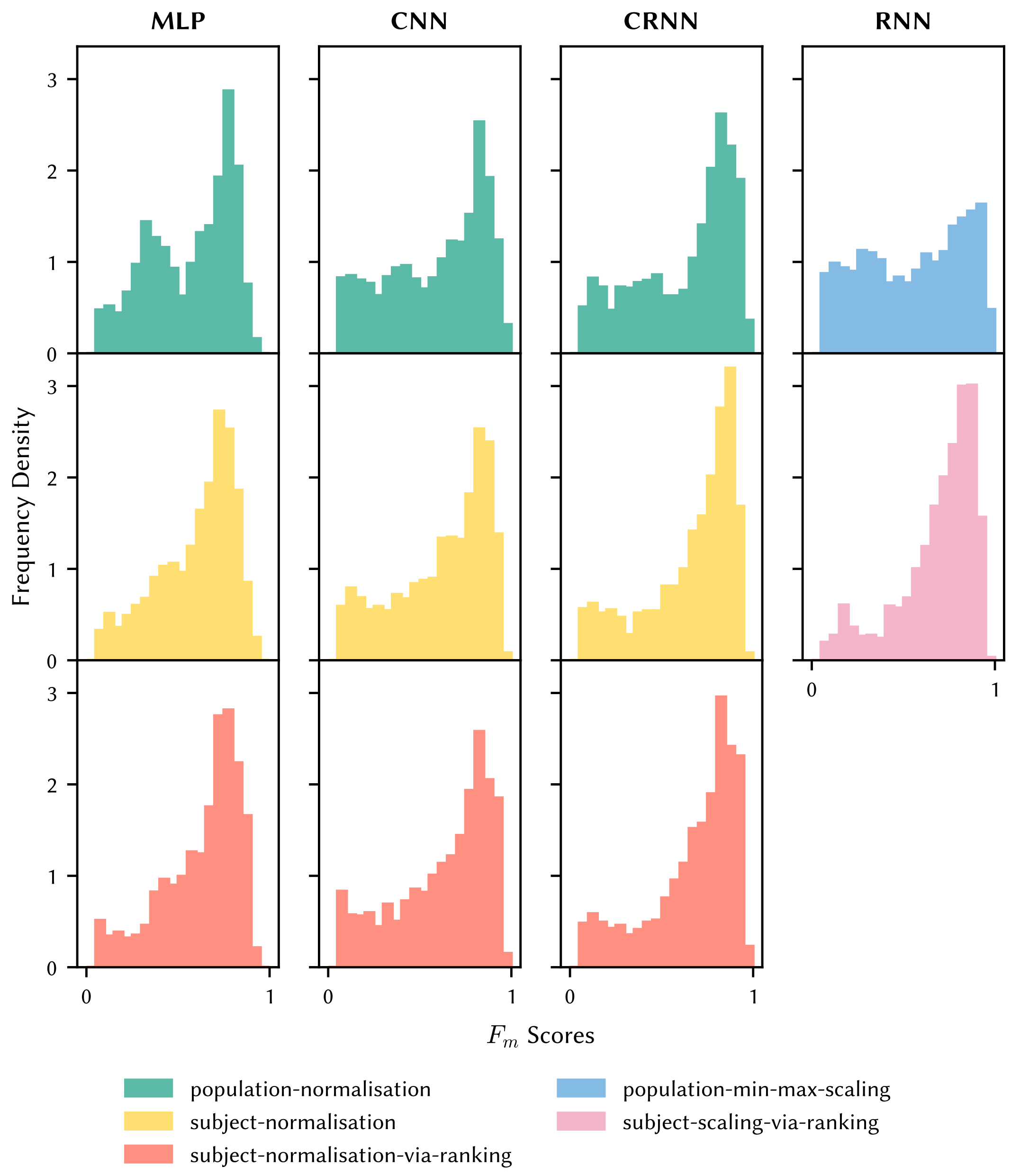
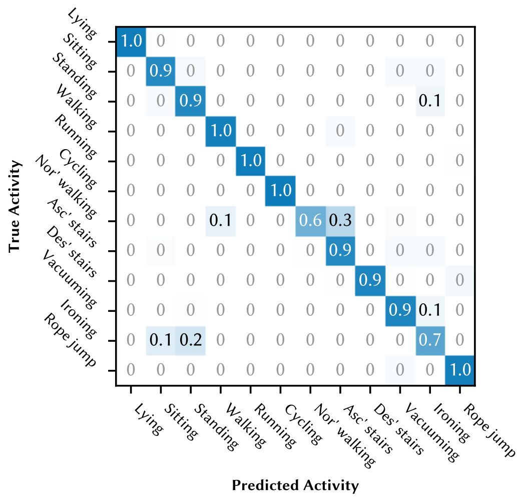
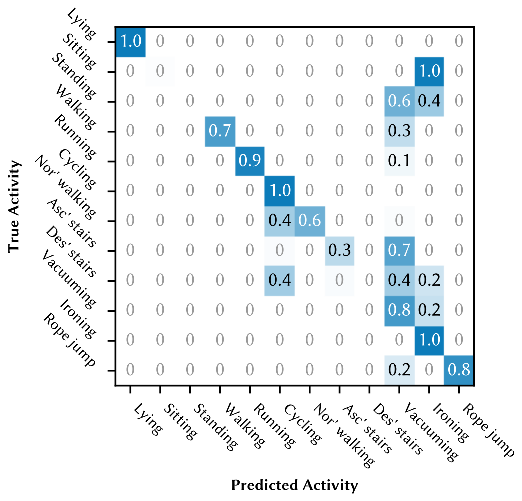


FIGURE 4.2: Frequency density distributions for F_m scores across all test subjects after training for all architectures and standardisation methods. 2000 250 per test subject) machines trained and evaluated for each combination. Machines which failed to perform any better than chance (23% of all models) are excluded for the sake of clarity.



(a) subject-normalisation-via-ranking



(b) population-normalisation

FIGURE 4.3: Confusion matrices for two different CRNN models trained on data standardised using two different methods and with subject 2 as the test set. Values on the diagonal running from upper left to lower right are equivalent to accuracy for the activity shown on the left axis. Values below 0.05 greyed out for clarity. Results for the subject-normalisation method were very similar to those shown in (b).

4.2.3 Analysis of Classification mistakes

Evaluating the kinds of mistakes machines make can provide insights into how they might be improved. From Figure 4.3(a), we see that almost all of the activities performed by subject 2 are correctly classified when the novel subject-normalisation-via-ranking method is used, with only a few mistakes occurring as a result of misclassifying Nordic walking as walking or ascending stairs, and ironing with standing or sitting. When the population-normalisation method is used, the accuracy decreases (consistent with the results in Table 4.2), but the effects are not evenly distributed across all activities as one might expect. Instead, some activities are classified almost perfectly, whilst other are misclassified entirely. For example, all of the sitting and standing samples are classified as ironing and vacuuming.

Interestingly, the reverse is not the case, i.e. ironing and vacuuming activities are never incorrectly classified as walking or sitting. This indicates that these misclassifications are not because subject 2 tends to generate similar readings when she is sitting, standing, ironing or vacuuming, otherwise vacuuming and ironing would occasionally be classified as sitting or standing. Instead, it's likely that the misclassifications arise as a result of similarities between the sensor readings generated when the men perform household chores and when subject 2 sits and stands. One possible explanation for this is that the male subjects were less energetic in their efforts.

Chapter 5

Conclusion

5.1 Summary of Research Objectives and Key Findings

Having reviewed the literature on HAR with deep learning methods, it was found that the results presented therein suffered from one of two weaknesses. The first of these is where training and testing had been evaluated on the same subjects, which therefore makes any results irrelevant for typical practical applications wherein data from new unseen subjects will need to be classified. The second is where results have been presented for only a single test subject, with no evidence that the performance is consistent across a range of unseen subjects.

In light of this, the first aim of this project was to determine if indeed any particular deep learning architecture consistently outperformed the other architectures (as is often alluded to in the literature [3, 12, 48]) by applying a rigorous LOSOCV process. The results of training and evaluating 22,000 machines on the PAMAP2 dataset, across a range of architectures and standardisation methods, indicate that CNNs, CRNNs and RNNs are all capable of achieving very similar good results. Indeed it was found that the highest performing combinations, CNNs and CRNNs paired with the novel subject-normalisation-via-ranking method, and GRU-RNNs paired with the population-min-max-standardisation method, achieved subject-averaged expected test F_m scores of 0.860, 0.851 and 0.844. However, there is no clear overall top performing combination.

On the basis of these results, practical considerations (such as training time versus available computing resources) ought to take precedence over potentially non-significant performance differences. Given a fixed amount of computation, far more CNNs can be trained than CRNNs or RNNs, due to their relative simplicity, meaning a large population of machines from which top performers can be picked. Therefore, the author would recommend that for the simplified HAR problem (see Section 2.1.1), that the CNN architecture is preferable. Similarly, MLPs may be useful to create as a baseline because of their short training time, but they would not be recommended for serious applications given that their performance was found to be significantly lower.

Another issue also became apparent from the literature review. It was clear that little to no research has been published concerning the performance effects that different data standardisation methods have on deep learning models. On this basis, a second aim for the project was defined, with the goal of determining these effects. It was found that, in contrast to population-level data standardisation methods, subject-level standardisation methods resulted in improved peak test F_m scores for MLPs, CNNs, GRU-RNNs, and CRNNs, and a higher expected test F_m score (averaged over all subjects) for MLPs, CNNs, and CRNNs. In addition, standardising input data on a per subject basis has the advantage of significantly improving the performance on subjects who are quite physically different from the training group (e.g. with subject 2) across all architectures, with the greatest improvements coming from the application of the novel standardisation methods (subject-scaling-via-ranking for RNNs, and subject-normalisation-via-ranking for CNNs, and CRNNs).

Because the data normalisation/standardisation via ranking methods clearly improved model generalisation across all architectures, and since there are indications that for CNNs and CRNNs they may even outperform the more commonplace subject-normalisation method, it follows that practitioners of HAR via deep learning may benefit from evaluating the effects of these novel methods as part of their model optimisation process.

5.1.1 Limitations of this Work

A LOSOCV approach was carried out so architectures and standardisation methods could be compared in a more rigorous way than had been done in prior research, however, there are still several improvements which could be made. Since the evidence for the claims presented is derived from experiments performed on a single dataset, there is a risk that the results would not generalise to other HAR datasets. After all, it is possible that the PAMAP2 dataset is in some way unusual when compared to other HAR datasets. Had the already significant amount of computational resources available not been exhausted by the large number of experiments which were required to generate the results presented here, it would have been useful to perform the analysis on other datasets.

It is also possible that the results given here do not reflect the true performance potential of the different architectures. This is because the complexity of machines was limited during the hyperparameter search (design) stage, and so only a relatively small number of designs (out of the infinite number of possible designs) could actually be tested. The limitations on design space were imposed because only a finite number of models could be trained given the finite computational resources, and since there are many more poor model designs than good ones, it was necessary to restrict the design space to something similar to what has already been shown in earlier research to produce reasonable models. If even more computational resources had been available, a more diverse design space could be tested which would improve the credibility of the results further.

5.2 Future Research

Given the early promising results which were seen with the subject-level standardisation methods, further research in this area may prove fruitful. The subject-normalisation and subject-normalisation-via-ranking methods each have, in addition to the shared benefit of improving model generalisation, their own potential advantages and disadvantages. The subject-normalisation method preserves all sensor information, since it applies only affine transformations, while subject-normalisation-via-ranking aligns subjects' distributions more closely (based on the relative intensity of the sensor readings).

If a standardisation method could be found which has better subject distribution alignment properties than subject-normalisation and a lower information loss than is incurred by subject-normalisation-via-ranking, it may lead to better performance than either method. One potential approach would involve using an unsupervised learning process to first identify common landmarks, across subjects, in the distributions of the values for each sensor (for example the peaks and troughs shown in Figure 3.4), and then, based on these landmarks, warp the data for each subject so as to align the these landmarks across subjects. After the warping procedure, application of the simple population-normalisation method should then be sufficient to appropriately scale and translate the data in preparation for training.

Bibliography

[1] Attila Reiss and Didier Stricker. Introducing a new benchmarked dataset for activity monitoring. In *Proceedings - International Symposium on Wearable Computers, ISWC*, pages 108–109, 2012. ISBN 9780769546971. doi: 10.1109/ISWC.2012.13. URL `http://www.pamap.org/demo.html`.

[2] Yann Lecun and Matthew Zeiler. Visualizing and Understanding Convolutional Networks. In *European conference on computer vision*, pages 1–11. Springer, 2015. ISBN 978-3-319-10589-5. doi: 10.1007/978-3-319-10590-1_53.

[3] Nils Y Hammerla, Shane Halloran, and Thomas Ploetz. Deep, Convolutional, and Recurrent Models for Human Activity Recognition using Wearables. In *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence*, pages 1533–1540. AAAI Press, 2016. ISBN 978-1-57735-770-4. URL `https://www.ijcai.org/Proceedings/16/Papers/220.pdf`.

[4] Yao Jen Chang, Shu Fang Chen, and Jun Da Huang. A Kinect-based system for physical rehabilitation: A pilot study for young adults with motor disabilities. *Research in Developmental Disabilities*, 32(6):2566–2570, 2011. ISSN 08914222. doi: 10.1016/j.ridd.2011.07.002.

[5] Oresti Banos, Miguel Damas, Hector Pomares, Alberto Prieto, and Ignacio Rojas. Daily living activity recognition based on statistical feature quality group selection. *Expert Systems with Applications*, 39(9): 8013–8021, 2012. ISSN 09574174. doi: 10.1016/j.eswa.2012.01.164.

[6] Vo Quang Viet, Gueesang Lee, and Deokjai Choi. Fall detection based on movement and smart phone technology. In *2012 IEEE RIVF International Conference on Computing and Communication Technologies, Research, Innovation, and Vision for the Future, RIVF 2012*, pages 1–4, 2012. ISBN 9781467303088. doi: 10.1109/rivf.2012.6169847.

[7] M. S. Ryoo. Human activity prediction: Early recognition of ongoing activities from streaming videos. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1036–1043. IEEE, nov 2011. ISBN 9781457711015. doi: 10.1109/ICCV.2011.6126349. URL `http://ieeexplore.ieee.org/document/6126349/`.

[8] W. Zeng and Z. Zhang. Microsoft Kinect Sensor and Its Effect. *Ieee Multimedia*, 19(2):4–10, 2012. ISSN 1070986X. doi: 10.1109/MMUL.2012.24. URL `http://ieeexplore.ieee.org/xpl/articleDetails.jsp?arnumber=6190806`.

[9] K Martin Sagayam and D. Jude Hemanth. Hand posture and gesture recognition techniques for virtual reality applications: a survey. *Virtual Reality*, 21(2):91–107, 2017. ISSN 14349957. doi: 10.1007/s10055-016-0301-0. URL `https://link.springer.com/content/pdf/10.1007{%}2Fs10055-016-0301-0.pdf`.

[10] W Jiang and Z Yin. Human activity recognition using wearable sensors by deep convolutional neural networks. In *MM 2015 - Proceedings of the 2015 ACM Multimedia Conference*, pages 1307–1310, 2015. ISBN 9781450334594. doi: 10.1145/2733373.2806333.

[11] Jian Bo Yang, Minh Nhut Nguyen, Phyo Phyo San, Xiao Li Li, and Krishnaswamy Shonali. Deep Convolutional Neural Networks On Multichannel Time Series For Human Activity Recognition. In *Proceedings of the 24th International Conference on Artificial Intelligence*, pages 3995–4001. AAAI Press, 2015. ISBN 9781577357384. URL `http://www.ijcai.org/Proceedings/15/Papers/561.pdf`.

[12] Francisco Javier Ordóñez and Daniel Roggen. Deep convolutional and LSTM recurrent neural networks for multimodal wearable activity recognition. *Sensors (Switzerland)*, 16(1):115–149, jan 2016. ISSN 14248220. doi: 10.3390/s16010115. URL `http://www.mdpi.com/1424-8220/16/1/115`.

[13] Oscar D Lara and Miguel A Labrador. A Survey on Human Activity Recognition using Wearable Sensors. *IEEE Communications Surveys & Tutorials*, 15(3):1192–1209, 2013. ISSN 1553-877X. doi: 10.1109/SURV. 2012.110112.00192. URL `http://ieeexplore.ieee.org/document/6365160/`.

[14] Christopher J.C. Burges. A Tutorial on Support Vector Machines for Pattern Recognition. *Data Mining and Knowledge Discovery*, 2(2):121–167, 1998. ISSN 13845810. doi: 10.1023/A:1009715923555.

[15] Jani Mäntyjärvi, Johan Himberg, and Tapio Seppänen. Recognizing human motion with multiple acceleration sensors. In *Systems, Man, and Cybernetics, 2001 IEEE International Conference on*, volume 2, pages 747–752 vol.2, 2001. ISBN 0-7803-7087-2. doi: 10.1109/ICSMC.2001.973004.

[16] Jennifer R Kwapisz, Gary M Weiss, and Samuel A Moore. Activity Recognition using Cell Phone Accelerometers. *SIGKDD Explor. Newsl.*, 12(2):74–82, 2010. doi: 10.1145/1964897.1964918.

[17] Marc Bächlin, Meir Plotnik, Daniel Roggen, Inbal Maidan, Jeffrey M Hausdorff, Nir Giladi, and Gerhard Tröster. Wearable assistant for Parkinsons disease patients with the freezing of gait symptom. *IEEE Transactions on Information Technology in Biomedicine*, 14(2):436–446, 2010. ISSN 10897771. doi: 10. 1109/TITB.2009.2036165.

[18] Susanna Pirttikangas, Kaori Fujinami, and Tatsuo Nakajima. Feature selection and activity recognition from wearable sensors. In *International Symposium on Ubiquitious Computing Systems*, pages 516–527. Springer Berlin Heidelberg, 2006. doi: 10.1007/11890348_39.

[19] Norbert Gy, Ákos Fábián, and Gergely Hományi. An Activity Recognition System For Mobile Phones. *Mobile Netw Appl*, 14(1):82–91, 2009. doi: 10.1007/s11036-008-0112-y.

[20] Martin Berchtold, Matthias Budde, Hedda R. Schmidtke, and Michael Beigl. An Extensible Modular Recognition Concept That Makes Activity Recognition Practical. In R{\"u}diger Dillmann, J{\"u}rgen Beyerer, Uwe D. Hanebeck, and Tanja Schultz, editors, *Lecture Notes in Artificial Intelligence Subseries of Lecture Notes in Computer Science LNAI Series Editors*, pages 400–409. Springer Berlin Heidelberg, 2010. ISBN 9783642335082. URL `https://link.springer.com/content/pdf/10.1007{%}2F978-3-642-16111-7.pdf`.

[21] Emmanuel Munguia Tapia, Stephen S Intille, William Haskell, K. W. J. Larson, Abby King, and Robert Friedman. Real-Time Recognition of Physical Activities and their Intensitiies Using Wireless Accelerometers and a Heart Monitor. In *11th IEEE International Symposium on Wearable Computers*, pages 37–40, 2007. ISBN 1424414539. doi: 10.1109/ISWC.2007.4373774.

[22] Andreas Bulling, Ulf Blanke, and Bernt Schiele. A tutorial on human activity recognition using body-worn inertial sensors. *ACM Computing Surveys (CSUR)*, 46(3):33:1–33:33, jan 2014. ISSN 03600300. doi: http://dx.doi.org/10.1145/2499621. URL `http://dl.acm.org/citation.cfm?doid=2578702.2499621`.

[23] Ricardo Chavarriaga, Hesam Sagha, Alberto Calatroni, Sundara Tejaswi Digumarti, Gerhard Tröster, José Del R. Millán, and Daniel Roggen. The Opportunity challenge: A benchmark database for on-body sensor-based activity recognition. *Pattern Recognition Letters*, 34(15):2033–2042, 2013. ISSN 01678655. doi: 10.1016/j.patrec.2012.12.014. URL `http://dx.doi.org/10.1016/j.patrec.2012.12.014`.

[24] Ron Kohavi. A Study of Cross-Validation and Bootstrap for Accuracy Estimation and Model Selection. *International Joint Conference on Artificial Intelligence*, 14(2):1137–1143, 1995. ISSN 10450823. doi: 10.1067/mod.2000.109031.

[25] Gavin C Cawley and Nicola L C Talbot. On Over-fitting in Model Selection and Subsequent Selection Bias in Performance Evaluation. *Journal of Machine Learning Research*, 11(Jul):20792107, 2010. ISSN 1532-4435. URL `http://www.jmlr.org/papers/v11/cawley10a.html`.

[26] Michael Esterman, Benjamin J Tamber-Rosenau, Yu Chin Chiu, and Steven Yantis. Avoiding non-independence in fMRI data analysis: Leave one subject out. *NeuroImage*, 50(2):572–576, 2010. ISSN 10538119. doi: 10.1016/j.neuroimage.2009.10.092.

[27] Bingbing Ni, Gang Wang, and Pierre Moulin. RGBD-HuDaAct: A Color-Depth Video Database for Human Daily Activity Recognition BT - Consumer Depth Cameras for Computer Vision. In *2011 IEEE International Conference on Computer Vision Workshops*, pages 1147–1153, 2011. ISBN 978-1-4471-4639-1. doi: 10.1109/ICCVW.2011.6130379.

[28] Piero Zappi, Thomas Stiefmeier, Elisabetta Farella, Daniel Roggen, Luca Benini, and Gerhard Tröster. Activity recognition from on-body sensors by classifier fusion: Sensor scalability and robustness. In *Proceedings of the 2007 International Conference on Intelligent Sensors, Sensor Networks and Information Processing, ISSNIP*, pages 281–286, 2007. ISBN 1424415020. doi: 10.1109/ISSNIP.2007.4496857.

[29] Kilian Förster, Daniel Roggen, and Gerhard Tröster. Unsupervised classifier self-calibration through repeated context occurences: Is there robustness against sensor displacement to gain? In *Proceedings - International Symposium on Wearable Computers, ISWC*, pages 77–84, 2009. ISBN 9780769537795. doi: 10.1109/ISWC.2009.12.

[30] M. S. Ryoo and J. K. Aggarwal. UT-Interaction Dataset, ICPR contest on Semantic Description of Human Activities, 2010. URL `http://cvrc.ece.utexas.edu/SDHA2010/Human{%}5C{_}Interaction.html`.

[31] Jaeyong Sung, Colin Ponce, Bart Selman, and Ashutosh Saxena. Human Activity Detection from RGBD Images. In *Proceedings of the 16th AAAI Conference on Plan, Activity, and Intent Recognition*, pages 47–55. AAAI Press, 2011. ISBN 9781577355328. doi: citeulike-article-id:9510173. URL `http://dl.acm.org/citation.cfm?id=2908772.2908779`.

[32] Davide Anguita, Alessandro Ghio, Luca Oneto, Xavier Parra, and Jorge L Reyes-Ortiz. A Public Domain Dataset for Human Activity Recognition Using Smartphones. In *European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning*, pages 24–26, Bruges, 2013. ISBN 9782874190810.

[33] Hong Cao, Minh Nhut Nguyen, Clifton Phua, Shonali Krishnaswamy, and Xiao-Li Li. An integrated framework for human activity recognition. In *Proceedings of the 2012 ACM Conference on Ubiquitous Computing - UbiComp '12*, pages 621–622, 2012. ISBN 9781450312240. doi: 10.1145/2370216.2370334.

[34] H. Sagha, S. T. Digumarti, J. del R. Millán, R. Chavarriaga, A. Calatroni, G. Tröster, and D. Roggen. Benchmarking classification techniques using the Opportunity human activity dataset. In *Systems, Man, and Cybernetics (SMC), 2011 IEEE International Conference on*, pages 36–40, 2011. ISBN 978-1-4577-0653-0.

[35] Thomas Plötz, Ny Hammerla, and Agata Rozga. Automatic assessment of problem behavior in individuals with developmental disabilities. In *Proceedings of the 2012 ACM Conference on Ubiquitous Computing*, pages 391–400, 2012. ISBN 9781450312240. doi: 10.1145/2370216.2370276.

[36] Zhenyu He and Lianwen Jin. Activity recognition from acceleration data based on discrete consine transform and SVM. In *Conference Proceedings - IEEE International Conference on Systems, Man and Cybernetics*, pages 5041–5044, 2009. ISBN 9781424427949. doi: 10.1109/ICSMC.2009.5346042.

[37] Li Deng and Dong Yu. Deep Learning: Methods and Applications. *Foundations and Trends® in Signal Processing*, 7(3-4):197–387, 2014. ISSN 1932-8346. doi: 10.1561/2000000039. URL `http://nowpublishers.com/articles/foundations-and-trends-in-signal-processing/SIG-039`.

[38] Itamar Arel, Derek Rose, and Thomas Karnowski. Deep machine learning-A new frontier in artificial intelligence research. *IEEE Computational Intelligence Magazine*, 5(4):13–18, 2010. ISSN 1556603X. doi: 10.1109/MCI.2010.938364. URL `https://www.researchgate.net/publication/224183837`.

[39] Jürgen Schmidhuber. Deep learning in neural networks: An overview. *Neural Networks*, 61:85–117, 2015. doi: 10.1016/j.neunet.2014.09.003. URL `http://dx.doi.org/10.1016/j.neunet.2014.09.003`.

[40] Li Deng. A tutorial survey of architectures, algorithms, and applications for deep learning. *APSIPA Transactions on Signal and Information Processing*, 3:e2, 2014. doi: 10.1017/atsip.2013.9.

[41] Dumitru Erhan, Yoshua Bengio, Aaron Courville, Pierre-Antoine Manzagol@umontreal Ca, Pascal Vincent@umontreal Ca, and Bengio@google Com. Why Does Unsupervised Pre-training Help Deep Learning? Pierre-Antoine Manzagol Pascal Vincent Samy Bengio. *Journal of Machine Learning Research*, 11(Feb): 625–660, 2010. URL `http://www.jmlr.org/papers/volume11/erhan10a/erhan10a.pdf`.

[42] Xavier Glorot, Antoine Bordes, and Yoshua Bengio. Domain Adaptation for Large-Scale Sentiment Classification: A Deep Learning Approach. In *Proceedings of the 28th international conference on machine learning*, pages 513–520, 2011. ISBN 978-1-4503-0619-5. URL `http://www.icml-2011.org/papers/342{_}icmlpaper.pdf`.

[43] Jianchang Mao and Anil K Jain. Artificial neural networks for feature extraction and multivariate data projection. *IEEE Transactions on Neural Networks*, 6(2):296–317, 1995. ISSN 10459227. doi: 10.1109/72.363467.

[44] Christopher M. Bishop. *Pattern recognition and machine learning.* Springer-Verlag, New York, 2006. ISBN 9780387310732.

[45] Dominik Scherer, Andreas Müller, and Sven Behnke. Evaluation of pooling operations in convolutional architectures for object recognition. *Artificial Neural Networks–ICANN 2010*, 6354:92–101, 2010. ISSN 03029743. doi: 10.1007/978-3-642-15825-4_10.

[46] Jost Tobias Springenberg, Alexey Dosovitskiy, Thomas Brox, and Martin Riedmiller. Striving for Simplicity: The All Convolutional Net. *arXiv preprint arXiv:1412.6806 (822 citations)*, 2014. ISSN 02548704. doi: 10.1163/_q3_SIM_00374. URL `https://arxiv.org/pdf/1412.6806.pdf`.

[47] Shuiwang Ji, Ming Yang, and Kai Yu. 3D Convolutional Neural Networks for Human Action Recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(1):221–231, 2013. ISSN 1939-3539. doi: 10.1109/TPAMI.2012.59.

[48] Ming Zeng, Le T Nguyen, Bo Yu, Ole J Mengshoel, Jiang Zhu, Pang Wu, and Joy Zhang. Convolutional Neural Networks for Human Activity Recognition using Mobile Sensors. In *Proceedings of the 6th International Conference on Mobile Computing, Applications and Services*, pages 197–205, 2014. ISBN 978-1-63190-024-2. doi: 10.4108/icst.mobicase.2014.257786.

[49] Charissa Ann Ronao and Sung-Bae Cho. Human activity recognition with smartphone sensors using deep learning neural networks. *Expert Systems with Applications*, 59:235–244, 2016. ISSN 09574174. doi: 10.1016/j.eswa.2016.04.032. URL `http://www.sciencedirect.com/science/article/pii/S0957417416302056`.

[50] Yoshua Bengio, Patrice Simard, and Paolo Frasconi. Learning Long-Term Dependencies with Gradient Descent is Difficult. *IEEE Transactions on Neural Networks*, 5(2):157–166, 1994. ISSN 19410093. doi: 10.1109/72.279181.

[51] Sepp Hochreiter and Jürgen Schmidhuber. Long Short-Term Memory. *Neural Computation*, 9(8):1735–1780, 1997. ISSN 08997667. doi: 10.1162/neco.1997.9.8.1735. URL `https://www.mitpressjournals.org/doi/pdf/10.1162/neco.1997.9.8.1735`.

[52] Junyoung Chung, Caglar Gulcehre, Kyunghyun Cho, and Yoshua Bengio. Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling. *arXiv preprint arXiv:1412.3555 (1568 citations)*, 2014. ISSN 2161-4393. doi: 10.1109/IJCNN.2015.7280624. URL `https://arxiv.org/pdf/1412.3555.pdf`.

[53] Yann A. LeCun, Léon Bottou, Genevieve B. Orr, and Klaus-Robert Müller. Efficient BackProp. In Gr{\'e}goire Montavon, Genevi{\`e}ve B. Orr, and Klaus-Robert M{\"u}ller, editors, *Neural Networks: Tricks of the Trade: Second Edition*, pages 9–48. Springer, Berlin, Heidelberg, Berlin, Heidelberg, 2012. doi: 10.1007/978-3-642-35289-8_3. URL `http://link.springer.com/10.1007/978-3-642-35289-8{_}3`.

[54] Xavier Glorot, Antoine Bordes, and Yoshua Bengio. Deep Sparse Rectifier Neural Networks. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, pages 315–323, 2011. URL `http://proceedings.mlr.press/v15/glorot11a`.

[55] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE International Conference on Computer Vision*, volume 2015 Inter, pages 1026–1034, 2015. ISBN 9781467383912. doi: 10.1109/ICCV.2015.123.

[56] Shaohuai Shi, Qiang Wang, Pengfei Xu, and Xiaowen Chu. Benchmarking State-of-the-Art Deep Learning Software Tools. In *Proceedings - 2016 7th International Conference on Cloud Computing and Big Data, CCBD 2016*, pages 99–104, 2016. ISBN 9781509035557. doi: 10.1109/CCBD.2016.029.

[57] James Bergstra, James Bergstra@umontreal Ca, and Yoshua Bengio@umontreal Ca. Random Search for Hyper-Parameter Optimization. *Journal of Machine Learning Research*, 13:281–305, 2012. URL `http://www.jmlr.org/papers/v13/bergstra12a.html`.

[58] Attila Reiss and Didier Stricker. Creating and Benchmarking a New Dataset for Physical Activity Monitoring. In *Proceedings of the 5th International Conference on PErvasive Technologies Related to Assistive Environments - PETRA '12*, pages 40:1–40:8, New York, 2012. ACM. ISBN 9781450313001. doi: 10.1145/2413097.2413148.

[59] Ilya Sutskever, James Martens, George Dahl, and Geoffrey Hinton. On the importance of initialization and momentum in deep learning. In *Proceedings of the 30th International Conference on International Conference on Machine Learning - Volume 28*, number ICML'13, pages III–1139—-III–1147, Atlanta, 2013. JMLR.org. ISBN 9781479903566. doi: 10.1109/ICASSP.2013.6639346. URL `http://proceedings.mlr.press/v28/sutskever13.pdf`.

[60] A. Krogh and John A Hertz. A Simple Weight Decay Can Improve Generalization. In *Advances in Neural Information Processing Systems*, volume 4, pages 950–957, 1992. ISBN 1558602224. URL `http://papers.nips.cc/paper/563-a-simple-weight-decay-can-improve-generalization.pdf`.

[61] Thomas G Dietterich. Approximate statistical tests for comparing supervised classification learning algorithms. *Neural Computation*, 10(7):1895–1923, 1998. ISSN 1530-888X (Electronic) 0899-7667 (Linking). doi: 10.1162/089976698300017197. URL `https://www.mitpressjournals.org/doi/pdf/10.1162/089976698300017197`.