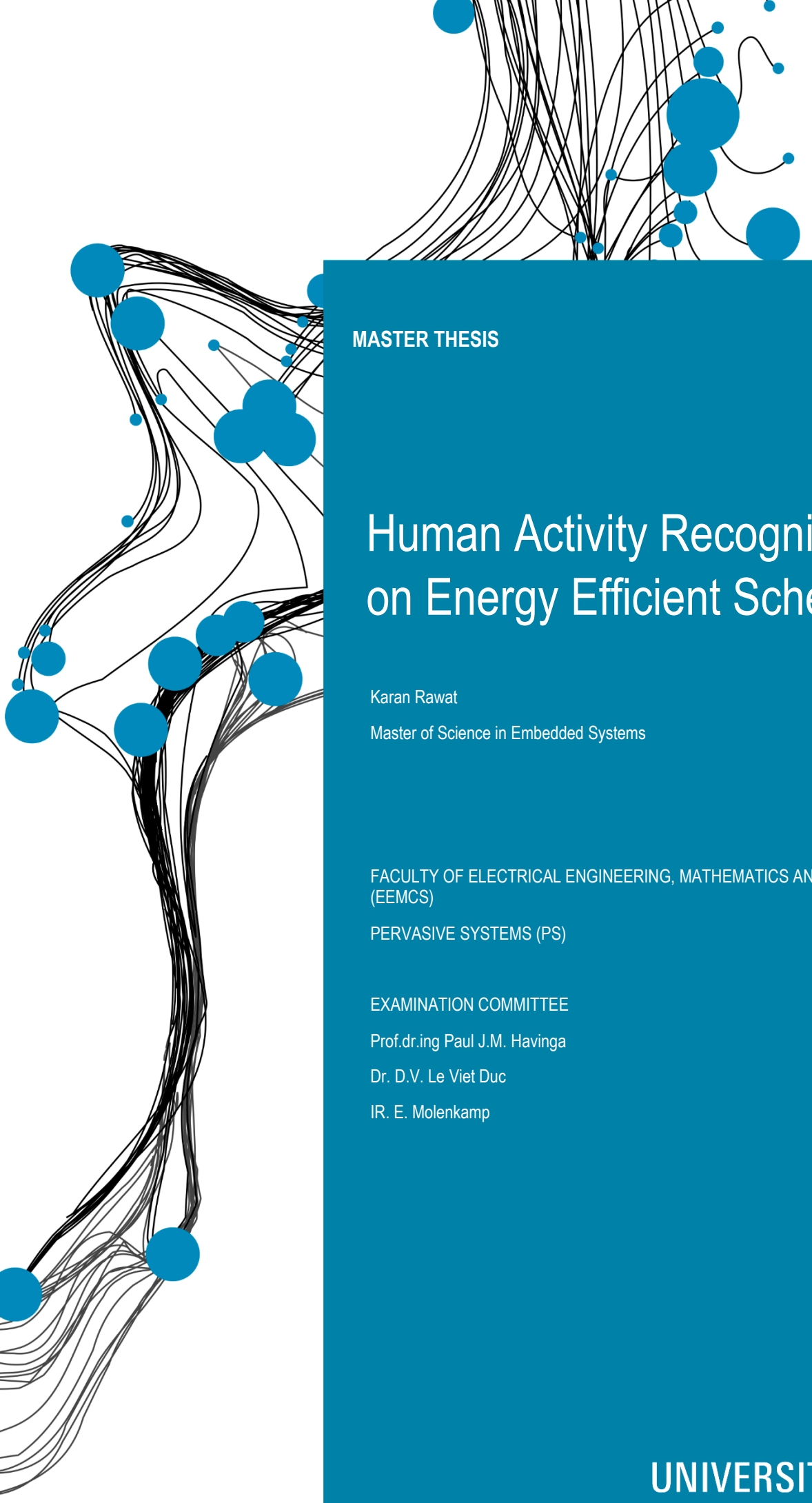

MASTER THESIS

Human Activity Recognition based on Energy Efficient Schemes

Karan Rawat

Master of Science in Embedded Systems

FACULTY OF ELECTRICAL ENGINEERING, MATHEMATICS AND COMPUTER SCIENCE (EEMCS)

PERVASIVE SYSTEMS (PS)

EXAMINATION COMMITTEE

Prof.dr.ing Paul J.M. Havinga

Dr. D.V. Le Viet Duc

IR. E. Molenkamp

UNIVERSITY OF TWENTE.

ACKNOWLEDGEMENTS

I would like to express my heartfelt gratitude to my supervisors Dr. P. J. M. Havinga and Dr. D. V. Le Viet Duc for their encouragement and more importantly their patience with this project. I would like to thank Mr. Molenkamp for agreeing to be a part of my committee.

Finally I owe everything to my parents without whom I would not be here.

LIST OF ABBREVIATIONS

Abbreviation	Description
BN	Bayesian Network
CNN	Convolutional Neural Network
HAR	Human Activity Recognition
IBL	Instance Based Learning
KNN	K-Nearest Neighbour
LSTM	Long Short-Term Memory
MDL	Minimum Description Length
NB	Naïve Bayes
PCA	Principle Component Analysis
PSD	Power Spectral Density
RF	Random Forest
RNN	Recurrent Neural Network
SVM	Support Vector Machine

LIST OF FIGURES

LIST OF TABLES

CONTENTS

ABSTRACT

Human activity recognition (HAR) has been an active area of research for decades. While traditional sensor-based activity recognition methods have demonstrated high recognition accuracy, they suffer from a significant overhead in terms of energy and computation, especially for resource-constrained devices. To address that, this thesis employs a multi-faceted approach to arrive at an optimized system where the design involves optimization of energy consumption through number of sensors, computation through minimal set of features and reduced classifier size and cost through a nominal hardware platform.

The performance of the design is evaluated against other baseline models in terms of classification accuracy and power consumption on two publicly available datasets. The experimental results on the Raspberry Pi Zero board show that the approach reduced the size of the network by 60% and maintained an accuracy of 96% while consuming 1.07J. These results validate the viability of the system on resource-constrained devices, thereby making it affordable to low-income groups.

1. INTRODUCTION

Context-aware ubiquitous computing has seen a meteoric growth with the ever-decreasing cost and size of microprocessors and sensors. Advances in the last decade on the development of embedded electronics and computer systems has improved the field of human activity recognition within context-aware systems. This has impacted fields like health monitoring and management, security and surveillance, behaviour recognition and monitoring, and ambient assisted living, among other things. There is a need for low-cost HAR designs that minimise data processing and resource consumption while accurately classifying the performed activity, to ensure the availability of activity recognition on low power systems.

1.1 Motivation

In literature, many activity recognition systems rely on data from multiple sensors and implement complex feature extraction and recognition algorithms. While this ultimately increases the accuracy of the system, it also increases the costs associated with energy consumption and complexity. There is a need for the deployment of HAR systems on cheap but resource-constrained embedded devices if these technologies are to be made universally accessible across various income groups.

This thesis proposes a system which reduces the obtrusiveness and hardware cost by using the data from a single accelerometer. The use of a single sensor reduces the computation cost as well. Combined with the computationally inexpensive features calculated from the data, this makes it possible to implement the system on a low-cost embedded platform, without making huge trade-offs with the recognition accuracy.

1.2 Research Questions

This thesis aims to improve the energy efficiency of human activity recognition while maintaining a good predictive accuracy over the existing benchmark research. The design and optimisation of a human activity recognition model, for deployment on an embedded device, requires addressing the following research questions:

Question 1: Which machine learning method can be used for the classifier?
An extensive literature survey will assess the merits and demerits of various machine learning algorithms and score them accordingly. Evaluating and comparing each of these models can help in selecting the right approach for performing activity recognition.

Question 2: Which factors affect the energy efficiency of sensor-based activity recognition?
During the literature survey, the different factors that have a significant impact on energy consumption are studied. Factors that are independent of the selected classifier

algorithm, as well as the dependent factors, are considered.

Question 3: How can the energy efficiency of the system be increased without a significant drop in its classification performance?
Experiments on the independent factors is conducted to find the suitable techniques that can help in the construction of an energy-efficient HAR system while maintaining acceptable accuracy levels.

Question 4: Which parameters are vital to the optimisation of the classifier, and what are the optimal values of the parameters?
Exhaustive experiments will be done with different values for the classifier parameters to establish an optimal configuration for the classifier that further improves its accuracy. The accuracy vs energy efficiency trade-off will be studied, prioritising energy efficiency.

The result would be an energy-efficient human activity recognition system that is computationally light, is deployed on an embedded platform and whose accuracy is at par with other HAR systems in the literature.

1.3 Outline

The remainder of this thesis will start with a discussion of the related work in Chapter 2. The details of the datasets used in the research are discussed in Chapter 3. Chapter 4 presents the implemented HAR system. Chapter 5 explains the conducted experiments and discusses the results. Chapter 6 concludes the thesis and provides scope for possible future work.

2. RELATED WORK

The fields of context-aware and ubiquitous computing have been at the forefront of research in human activity recognition using on-body sensors. With the extensive accessibility of smartphones, research has focused on data captured by the sensors embedded in them. Among these, the accelerometer sensor has been very popular due to its low energy requirement and good performance in capturing human motion.

This chapter briefly overviews previous work done in the field of human activity recognition, and is structured as follows: Section 2.1 discusses the different sensor placements studied in the literature, Section 2.2 presents how researchers have collected and processed data in the literature, Section 2.3 explains the process of feature selection and extraction, and Section 2.4 examines the choice of classifiers made in the literature.

2.1 Sensor location and obtrusiveness

In literature, many on-body sensor locations like the waist, wrist, thigh, ankle etc. have been discussed. [1] concluded that while accelerometers placed on the hip, wrist, upper arm, ankle and thigh performed well, the performance only dropped slightly with the uses of just two biaxial accelerometers on the thigh and the wrist respectively. [2] concluded that models aware of the locations of the sensors on the subject's body performed better than the models that were independent of the location information. Recent deep learning architectures learn representations from raw data from multiple on-body sensors [3][4]. The obtrusiveness of on-body sensors plays an essential part in the success of a HAR system. Configurations like [5], [6], [1], and [7] are invasive and uncomfortable and unsuitable for HAR. Not only does the obtrusiveness of a system reduce by minimising the number of sensors, but it also reduces energy consumption and improves processing (fewer data to be processed). In the end, the number and location of on-body sensors are governed by user acceptance and resultant accuracy.

2.2 Data collection and processing

The first step of a HAR system is the collection of raw data from on-body sensors. This raw data is sampled for every sensor, and the resulting multivariate time series requires

synchronisation. This raw data can also be corrupted by various sources resulting in artefacts. The pre-processing step synchronises the multimodal sensor data, resulting in a time series. Signal processing algorithms are used to filter out the artefacts while ensuring the preservation of relevant information about the activities of interest. Techniques like calibration, unit conversion, normalisation, resampling, synchronisation, or signal-level fusion[8] can be used for pre-processing of accelerometer and gyroscope signals.

The segments of the pre-processed data that are most likely to contain activity information are identified in the data segmentation stage. Since human activities are fluent, it is not easy to segment a continuous sensor data stream. Three window-based data segmentation techniques exist in the literature, activity-defined window, event-defined window, and sliding window[9]. The sliding window approach is the most used technique because of its simplicity and better performance. The window length is a key factor for HAR systems since it directly determines the performance of the system. It is subject to a trade-off between precision and the computational load on the system; smaller windows can increase precision, but the increased number of samples would increase the computation load on the system. The optimal window size is not clear beforehand. Different window lengths have been used in the literature: 0.08s [10], 1s [11], 1.5s [12], 3s [13], 5s [14], 7s [15], 12s [16], or up to 30s [5]. Of course, this decision is influenced by the activities to be recognised and the attributes that are being measured. The generated samples are passed through a low-pass filter to remove high-frequency noise [17]. Furthermore, a Butterworth filter is used to separate the low-frequency gravitational acceleration from the higher frequency body acceleration [18].

2.3 Feature selection and extraction

In this stage, the segmented data streams are reduced into discriminatory activity features. The total number of features that are extracted from the data makes up the feature space. Ideally, features belonging to the same activity are clustered together, and those of different activities are clearly separated in the feature space. The recognition performance of the system depends on how clearly the feature groups are separated in the feature space. In literature, time-domain features such as mean and standard deviation have been used to distinguish between postures [19], and static and dynamic activities, respectively. In the case of multi-axial accelerometers, the correlation between different axes has been used to distinguish movement along different axes [20]. Frequency domain features like energy and entropy can help measure the periodicity and complexity of activities, respectively [21].

Not all features in the processed dataset are essential for recognition, and feature ranking and feature reduction techniques are useful to reduce computations and build simpler learning models. The complexity of the system can be reduced for the training and classification stages by using only a subset of the feature space. Additionally, this would prevent overfitting, and the system would generalise better for unseen data. As pointed out in [22] for a given set of training examples, increasing the number of features beyond the optimal subset of features among a set of N possible features will decrease performance. In literature, methods such as the Principal Component Analysis (PCA) and the Minimum Description Length (MDL) [23] have been used for feature reduction. [24] proposed a feature reduction system that reduced the number of features from 1170 to 30 by performing PCA.

2.4 Choice of classifier

In the context of human activity recognition, the training set consists of feature vectors extracted from time windows.

a) Supervised learning
   Supervised learning takes training data which is labelled with the class names

(e.g., running, sitting), and its known outputs, to develop predictive models.

i. *Decision trees:* Decision trees sort instances based on feature values using the Iterative Dichotomiser 3(ID3) algorithm. In a decision tree, every node represents a feature, and each branch represents a value that the node can assume in the hierarchical model of a decision tree. Each branch from the root to a leaf node is a classification rule [25]. Post-pruning techniques are used to avoid over-fitting decision trees.

ii. *Bayesian* methods: The *Bayesian Network* (BN) [26] classifier and *Naive Bayes* (NB) [27] is the principal exponents of this family of classifiers. They use graphical models to calculate probability relationships between features. Topology construction of the directed acyclic graphs used for these methods is an issue since it is necessary to assume independence between feature nodes. These assumption does not hold in many cases. Bayesian Networks are not suitable for datasets with many features [28].

iii. *Instance-based learning* (IBL): These methods use a distance function to compare an instance with similar instances in the training set [23]. Since each new instance is evaluated against every instance in the training set, this makes these methods computationally very expensive.

iv. *Support Vector Machines:* SVMs [29] use kernel functions to find the optimal hyperplane that partitions the given dataset with the maximum margin. The versatility of choosing the kernel function makes SVMs useful for higher dimensions. They are also memory efficient since the support vectors used for the decision function are just a subset of the feature points.

v. Convolutional Networks: CNN's are analogous to the neurons in the human brain. They incorporate local connections, parameter sharing, pooling and multi-layers. CNN's were the first hierarchical deep learning architectures [11]. CNN's have been used as an automatic feature extractors [11], [30]. In terms of HAR, CNNs can extract localised features and can retain pace and frequency information from extracted features because of their scale-invariance to changes in pace and frequency.

vi. Recurrent Neural Network: Recurrent networks are suitable for exploiting temporal dependencies in movement data. RNN based LSTM (long short term memory) [31] was designed to avoid the conventional RNNs drawbacks with long-term dependencies. Each LSTM cell tracks its internal state and can learn to change it based on current and historical state values. LSTMs can thus retain information over many time-steps [32]. LSTMs can access the long-range context of sequential data, and its accuracy increases with an increase in available data. This also makes RNNs more susceptible to changes in the input data.

b) Semi-supervised learning
Some systems use partially unlabelled data to employ a semi-supervised learning approach. The efforts put into collection and precision of datasets has made supervised learning more prominent in literature, with only a few following semi-supervised techniques. Semi-supervised techniques are useful when the annotation of data is difficult due to the high granularity of the activity [33]–[37]. Semi-supervised methods do not follow a general approach.

2.5 Network Pruning

Pruning of neural networks has been shown to be effective at reducing network complexity and improving generalisation [38]. The idea of pruning is to explore the redundancy in the model parameters and try to remove the redundant, non-informative and uncritical parameters which are not crucial for model performance.

a) Weight quantization
Network compression using quantization works by reducing the number of bits required to represent weights and biases in a network. [39] showed that a significant improvement in speed can be achieved with minimal loss of accuracy if weights are represented using 8-bit quantization. The number of floating point operations and the memory usage was significantly reduced by using 16-bit fixed-point representation in [40].

b) Network pruning
Network pruning has been used to address the complexity of neural networks as well the over-fitting issue while training them. To remove the redundancy in networks, [41] proposed a data-free pruning method. [42] used a deep compression method to remove the redundant connections. Furthermore, the weights were quantized and then encoded using Huffman coding. In [43], low-cost hash functions were used to group the weights of a HashedNets model. [44] proposed a simple regularization method to perform quantization and pruning in a simple (re-) training procedure.

3. DATASETS

In this work two public HAR datasets, the WISDM dataset and the UCI HAPT dataset, are used for the training and classification purposes. These datasets were selected because,

- The location of the smartphones on the subject's body were less obtrusive [1]
- Both the datasets lack rotational activities, which results in minimal overlap between the frequency domains of the gravitational and movement-related components of acceleration [45] and makes their separation using simple frequency-based filters easier.

3.1 WISDM Dataset

The Wireless Sensor Data Mining (WISDM) Lab's dataset [46] [47] contains sensor signal recordings of 36 subjects performing six activities- WALKING, WALKING UPSTAIRS, WALKING DOWNSTAIRS, SITTING, STANDING and JOGGING. The raw data consists of 1,098,207 data samples collected from the embedded tri-axial accelerometer of the subjects' Android smartphones, which were kept in their front pockets. The sampling frequency was 20Hz. A total of 915 minutes of recording data is collected. The corresponding proportion of each activity is shown in figure 1.

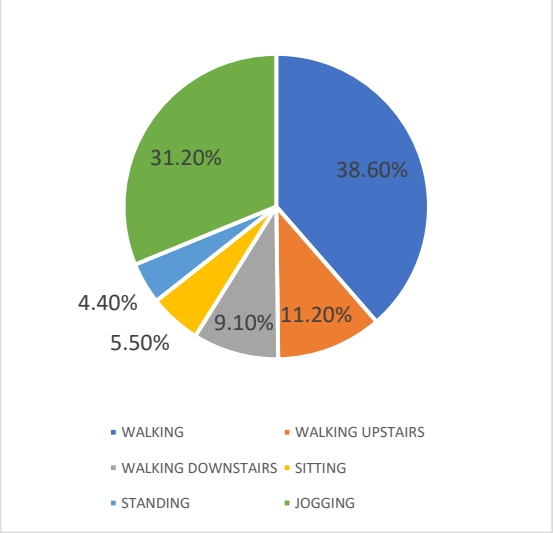

Figure 1: Distribution of the different activities in the WISDM dataset

3.2 UCI HAPT Dataset

The University of California Irvine's Human Activities and Postural Transitions (UCI-HAPT) dataset [48] is the updated version of the more popular UCI-HAR dataset [17]. It provides the original raw accelerometer and gyroscope signals collected from a waist-mounted Samsung Galaxy S II phone. The dataset consists of signal recordings of 30 subjects between the ages of 19 and 48 performing three static activities- STANDING, SITTING and LYING; and three dynamic activities- WALKING, WALKING DOWNSTAIRS and WALKING UPSTAIRS; and six postural transitions- STAND_TO_SIT, SIT_TO_STAND, SIT_TO_LIE, LIE_TO_SIT, STAND_TO_LIE and LIE_TO_STAND. The data is collected at a sampling frequency of 50Hz. The corresponding proportion of each activity is shown in figure 2.

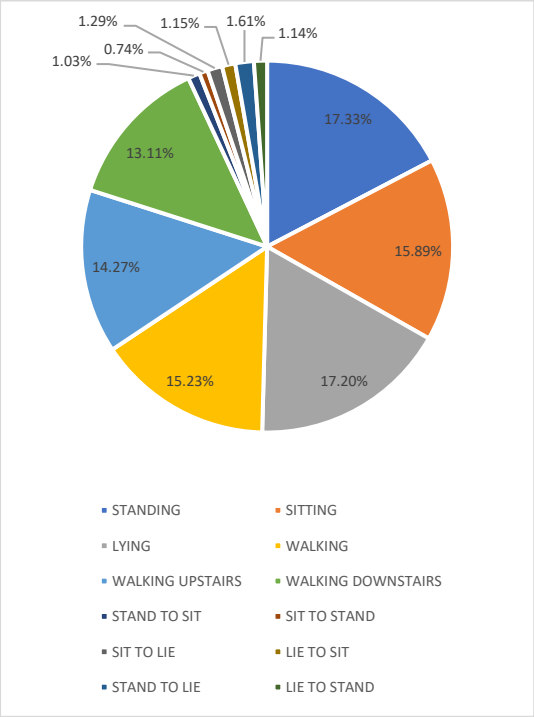

Figure 2: Distribution of the different activities in the UCI-HAPT dataset

4. HUMAN ACTION RECOGNITION

The aim of this research was to develop a lightweight HAR model which uses data from embedded accelerometers. The general structure of HAR systems is similar to other pattern recognition systems, as shown in figure 3. The input to the training phase is a time series stream of sensor data for the measured attributes. This data is pre-processed to denoise them and filter out any artifacts. Relevant features are extracted by splitting the time series data into time windows. In the training phase, these features are used by the learning methods to train an activity recognition model. In the testing phase, such a pre-trained classifier and new extracted features are used to calculate a predicted activity label. The implemented HAR process is discussed in the following subsections.

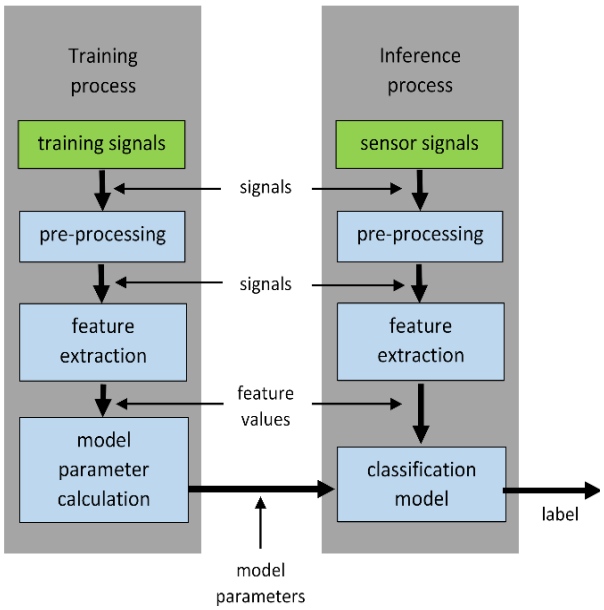

Figure 3: General data flow for training and testing HAR systems based on wearable sensors[49]

4.1 Pre-processing data

The raw data was pre-processed by applying a rolling mean low-pass filter with a window size of 3 [30] to smoothen the signal and remove high frequency noise, as shown in figure 4. Furthermore, the gravitational

component of the acceleration data was separated from the body motion component by using a low-pass Butterworth filter.

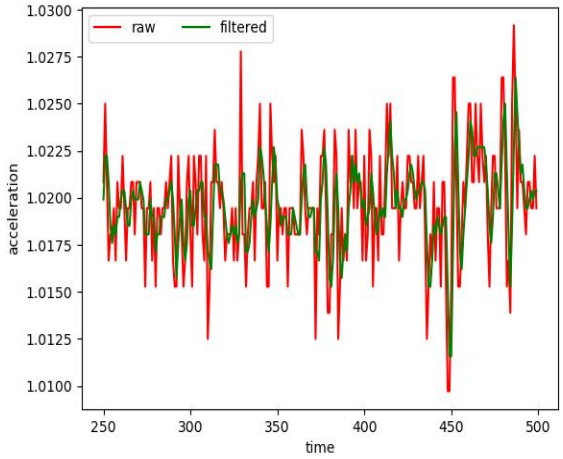

Figure 4: Effect of the low-pass filter

After the data was de-noised, it was sampled in fixed-width sliding windows. The percentage of overlap was 50% for both datasets. For the WISDM dataset, 100 readings/window were used, whereas for the HAPT dataset, 150 readings/window were used.

4.2 Feature extraction and Normalisation

Selecting of the proper features from raw data is an important factor in the overall performance of a HAR system. Since the data is collected from triaxial accelerometers, features were extracted from each axis. The extracted features are Mean, Maximum, Minimum, Standard Deviation, Skew, Median, Kurtosis, Energy (normalized sum of squared amplitude) and Weighted Mean Frequency (calculated by PSD). Furthermore, the correlation between each pair of axes is used. This amounts to each sample being a (9*3 + 3) 30-dimensional vector.

To make sure that the gradient descents of different features can converge quickly, we transform all the features to a common scale. Using the min-max normalisation method, all the feature values are normalised to the 0-1 scale.

4.3 Dimensionality reduction

Higher number of features or dimensions are harder to visualise and fail geometric intuition, and are computationally expensive to use. Dimensionality reduction is the process of projecting data to a lower dimensional subspace, such that redundant features are removed, and multi-collinearity taken care of. Ideally, the set of reduced features are a combination of the input variables and represent the same information, but in a lower dimensional subspace. The total (cumulative) variance is the sum of the individual variances of the principal components. As is evident from figure 5, more than 90% of the cumulative variance can be explained by the first 10 principal components. Principal Component Analysis (PCA) is used to reduce the dimensionality of the feature set. The key idea is to decompose the input into principle components, which are the linear combinations of the input features.

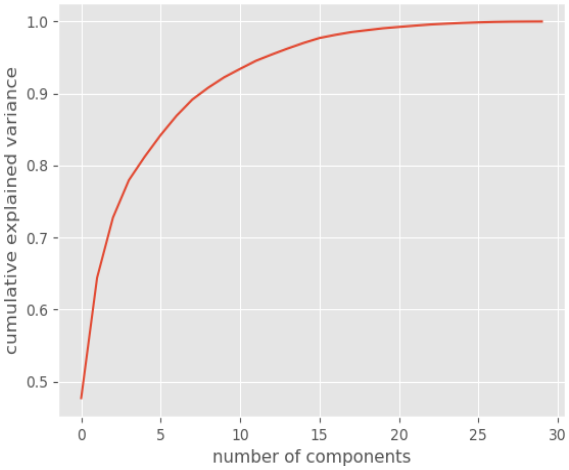

Figure 5: Cumulative explained variance in the HAPT dataset

4.4 CNN architecture

The model developed for our experiment uses a sliding window size of size equal to the number of activity labels of the dataset for the convolution with a ReLU activation.

Following the convolutional layer is a max-pooling layer, and then a flatten layer to convert the output of max-pooling function into a single vector. This is followed by a dense layer with a ReLU activation function, and a final dense layer with a 50% dropout rate and a softmax layer. The basic Keras [50] generated architectural view of the network is shown in figure 6.
The ReLU activation function was used for its computational efficiency and decreased run time. Furthermore, ReLU avoids vanishing gradients over multiple layers. The softmax function was used in the output layer because it models probabilistic distributions well and is a good fit for 1 of N classification.
The Adam optimiser was used because of its low memory requirements and computational efficiency; with MSE as the loss function. The epoch size was set to 50, learning rate to 0.0004 and batch size to 32 samples.

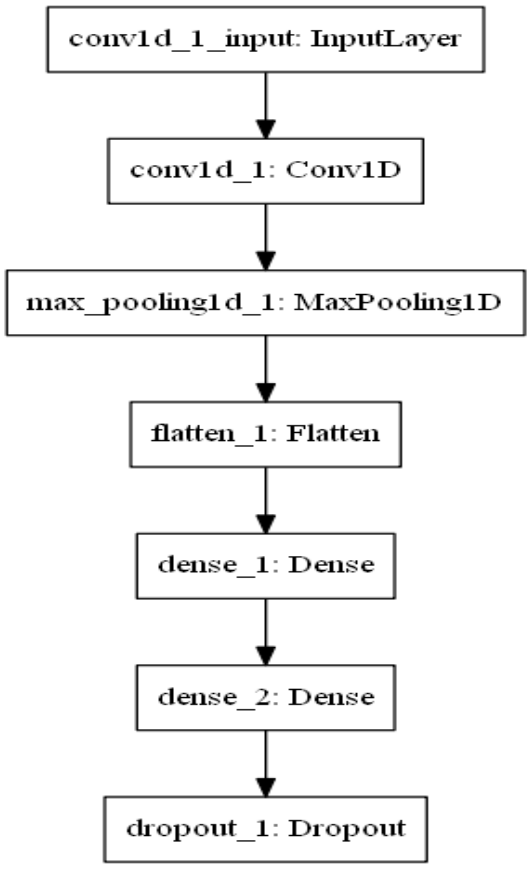

Figure 6: CNN architecture for HAR

4.5 Network Pruning

Once a CNN is fully trained, weights are assigned to the neurons with respect to their importance in the network, with the least active nodes being assigned values closer to zero. To reduce the size of the network, and speed-up inference, neurons with lower weights are removed.

```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv1d_1 (Conv1D)            (None, 25, 100)           700
_________________________________________________________________
max_pooling1d_1 (MaxPooling1 (None, 12, 100)           0
_________________________________________________________________
flatten_1 (Flatten)          (None, 1200)              0
_________________________________________________________________
dense_1 (Dense)              (None, 6)                 7206
_________________________________________________________________
dense_2 (Dense)              (None, 6)                 42
_________________________________________________________________
dropout_1 (Dropout)          (None, 6)                 0
=================================================================
Total params: 7,948
Trainable params: 7,948
Non-trainable params: 0
_________________________________________________________________
```

Figure 7: Number of parameters in a non-pruned network

As can be seen from figure 7, the first dense layer contains the majority of the parameters, and it is this layer that was pruned. This was done by sorting the neurons according to their absolute values and then removing a certain percentage (pruning factor) of them. Once the neurons are removed and the network retrained, the pruned model has a significantly lower number of neurons, which reduces its memory footprint and computational requirement. The effect on accuracy isn't significant up to a certain pruning factor (0.5 in our case), after which higher weight neurons start getting removed and the accuracy plummets. Figure 8 shows an example of pruning.

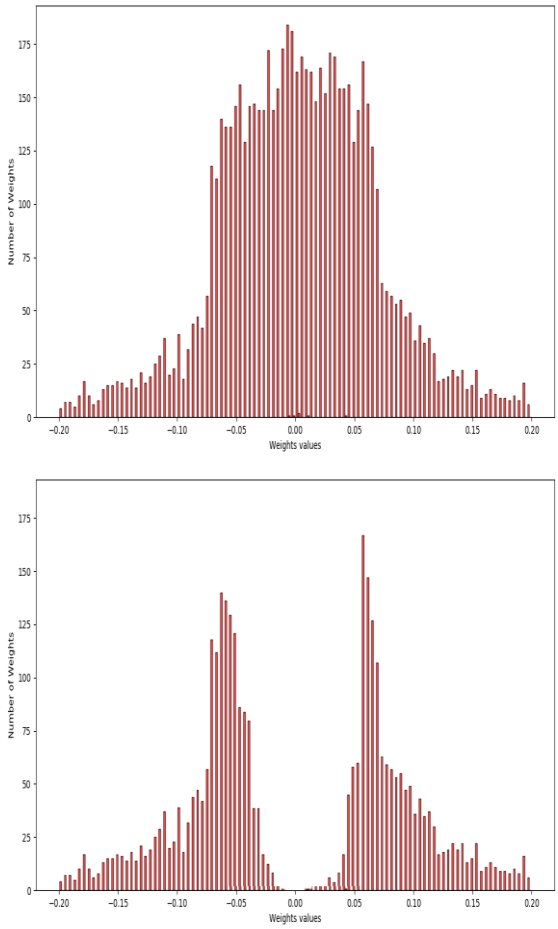

Figure 8: Distribution of weights before pruning (top) and after pruning (down) the dense layer. Pruning removes neurons with weight values close to 0.

5. EXPERIMENTS AND RESULTS

The code for this thesis was written in Python 3.x. This pipeline extensively uses open source machine learning libraries Scikit-Learn [51] and Numpy [52]. The deep learning models were developed using Keras [50] with TensorFlow [53] as backend. TensorFlow was used as the backend because it is well established in the deep learning community. Another advantage of TensorFlow is the format of the saved models. The TensorFlow documentation [54] describes SavedModel as "a language-neutral, recoverable, hermetic serialization format that enables higher-level systems and tools to produce, consume, and transform TensorFlow models". The SavedModel format enables the saved models to be transferred to any device and perform inferences without the need of retraining or compatibility issues (provides TensorFlow is supported by the device). This enabled the experiments to be conducted on a PC and the Raspberry Pi Zero board, with the training done on the PC.

5.1 PC

The PC used is a Dell Inspiron 15 5559 with a 6th Generation Intel Core i5-6200U 2.30GHz Processor. The laptop is not equipped with a dedicated GPU and hence GPU accelerated libraries were not used.

5.1.1 Classification Accuracy and Sampling Frequency

The impact of different sampling frequencies on the classification accuracy was investigated. As shown in figure 9, higher sampling frequencies give better classification accuracies, with a 3% loss in classification accuracy between the original sampling frequency of the HAPT dataset, 50Hz, and the lowest studied sampling frequency of 20Hz. Since the window size is the same while studying all the sampling frequencies, the reduction in accuracy can be attributed to two reasons,

- Some of the characteristics of a signal captured at a higher sampling rate are missed at a low sampling rate. Less characteristics results in a lower accuracy for the classifier. As the sampling rate grows, we can observe a general trend of improving classification performance.
- The original sampling rate for the HAPT dataset was 50Hz and the lower sampling rates were obtained by reducing the number of samples. Thus, the reduction in sampling rate directly resulted in a lower number of samples for the classifiers to be trained against, thereby affecting its accuracy.

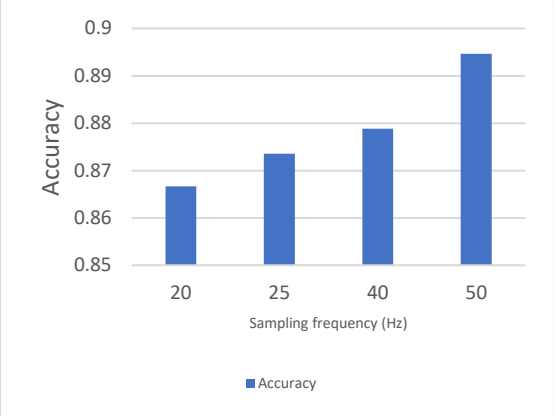

Figure 9: Effect of Sampling Frequency on Accuracy (HAPT)

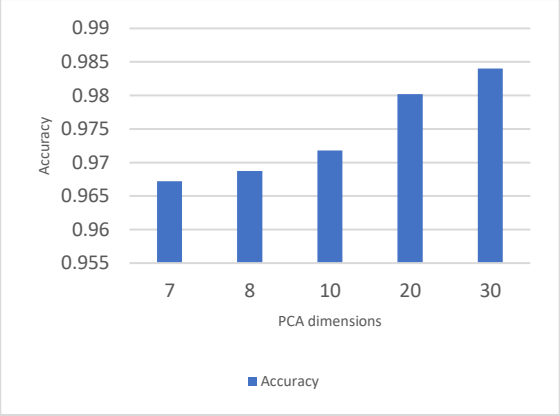

Figure 10: Effect of Dimensionality on Accuracy (WISDM)

5.1.2 Classification Accuracy and Feature Vector Dimensionality

Figures 10 and 11 show how the accuracy changes with the feature vector dimensionality. While accuracy is higher when the number of features is more, the difference is not a lot. For both datasets, about 2% loss in accuracy is observed between the maximum and minimum values of the feature dimensions. Intuitively, reducing the number of features by a factor of 4 (or 2 in case of HAPT) should adversely affect the accuracy, but only a low drop-off in accuracy is observed. As seen in figure 5, most of the cumulative variance is explained by the first few principal components (dimensions). This means that a high classification accuracy can be achieved by using the principal components, and increasing the components beyond a certain point does not increase the accuracy of the classifier by a lot.

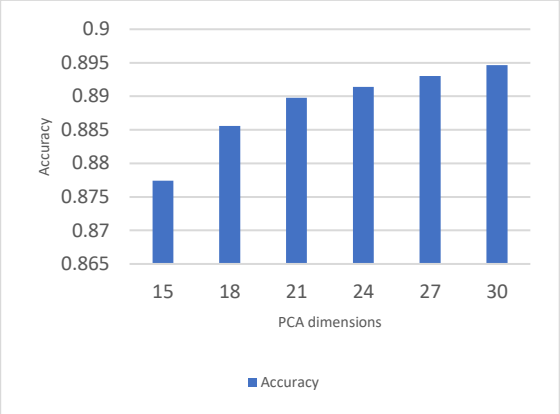

Figure 11: Effect of Dimensionality on Accuracy (50Hz, HAPT)

5.1.3 Classification Accuracy and Pruning

Figures 12 and 13 show the trend for the accuracy to drop as the percentage of pruned weights increases. As the pruning factor increases, the higher weight neurons start to get affected by the pruning procedure and are removed from the network. This is why the reduction in accuracy is less initially, and then takes a plunge as more and more higher weight neurons start to get affected.

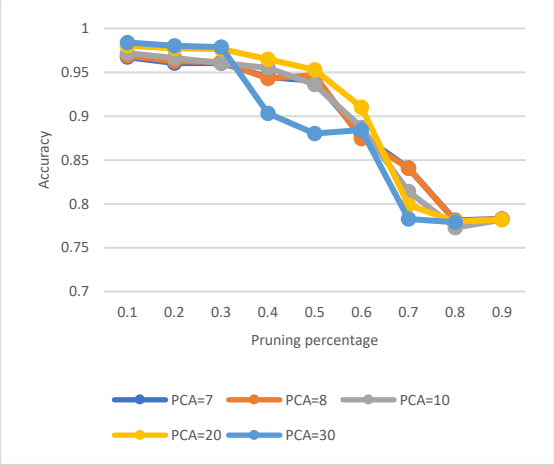

Figure 12: Effect of Pruning on Accuracy (WISDM)

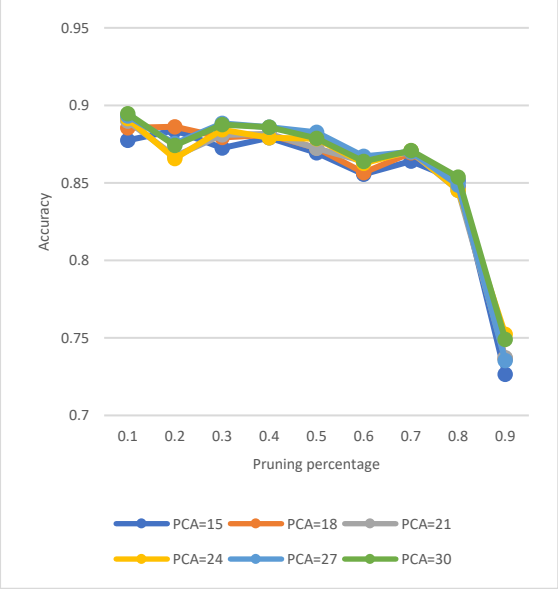

Figure 13: Effect of Pruning on Accuracy (50Hz, HAPT)

5.1.4 Performance Comparison with other Classifiers

The performance of the pruned CNN classifiers is compared with those of standard machine learning classifiers trained on the same datasets. Accuracy is chosen as the performance metric as size and energy consumption are not important factors when comparing classifier performances on a PC. The accuracy of our pruned CNNs are tabulated against other classifiers in tables 1 and 2.

As can been seen from tables 1 and 2, the pruned CNN has a slightly lower classification accuracy (difference of 2%) as compared to the other classifiers. Since the difference in accuracy is not huge, it can be concluded that the CNN does not face a huge disadvantage in terms of accuracy against the other classifiers.

Table 1: Comparison of accuracy among classifiers (WISDM)

Sampling rate	PCA dims	SVM	KNN	RF	CNN (pruned)
20	7	0.971037	0.9847561	0.989329	0.960366
20	8	0.965701	0.9809451	0.990854	0.96189
20	10	0.980183	0.9832317	0.994665	0.960366
20	20	0.988567	0.9847561	0.991616	0.976372
20	30	0.990091	0.9847561	0.992378	0.980183

Table 2: Comparison of accuracy among classifiers (HAPT)

Sampling rate	PCA dims	SVM	KNN	RF	CNN (pruned)
20	15	0.889431	0.890244	0.895122	0.869106
20	18	0.901626	0.886992	0.902439	0.877236
20	21	0.903252	0.889431	0.901626	0.878862
20	24	0.902439	0.891057	0.898374	0.87561
20	27	0.903252	0.891057	0.897561	0.878049
20	30	0.90813	0.891057	0.888618	0.881301

5.2 Raspberry Pi Zero

The Raspberry Pi Zero is a single board computer with a single core 1GHz Broadcom BCM2835 CPU. At 65mm x 30mm x 5mm, it is smaller than bank cards, and weighs just nine grams. Its small dimensions, low cost and low input voltage requirement of just 5V make it ideal for embedded applications. Since the classifiers are trained on the PC, the inference trends generated by the Pi are the same. Unlike the PC, the energy and size considerations of inference tasks are important factors with the Pi.

5.1.5 Energy Consumption and Sampling Frequency

As observed in figure 14, the accuracy of the classifier increases with an increase in sampling frequency, but so does the energy consumption of the inference task. With a constant window size, the number of features calculated per window are higher for higher sampling rates. This results in more energy being consumed for inference, since more features need to be processed.

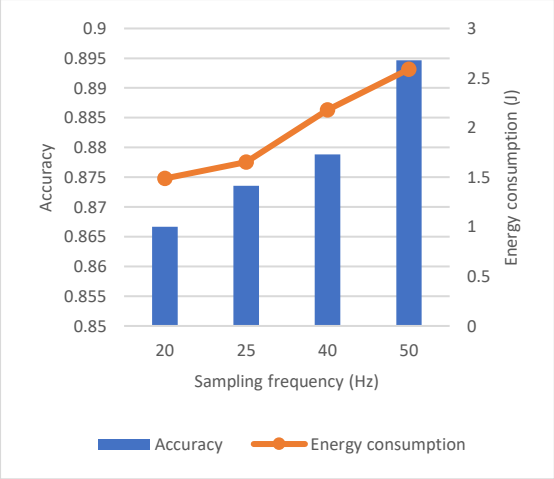

Figure 14: Effect of Sampling Frequency on Accuracy and Energy consumption on Raspberry Pi Zero (HAPT)

5.1.6 Energy Consumption and Feature Vector Dimensionality

Figures 15 and 16 show the impact of feature vector dimensionality on the accuracy and energy consumption of the CNN on the Pi Zero. The drop in accuracy between the maximum and minimum PCA values is about 2% for both datasets. This is an insignificant change when compared to the drop in energy consumption (1J per inference task).

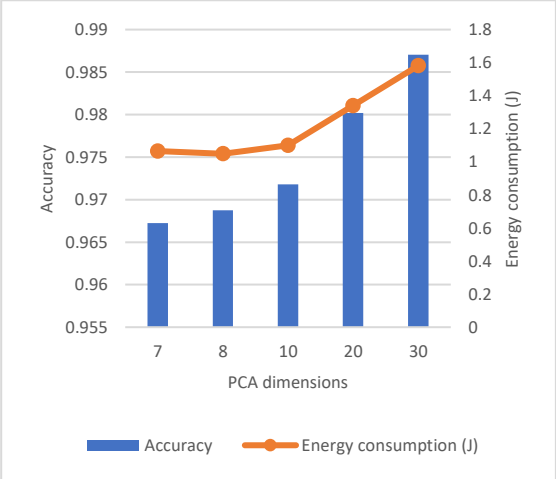

Figure 15: Effect of Dimensionality on Accuracy and Energy consumption on Raspberry Pi Zero (WISDM)

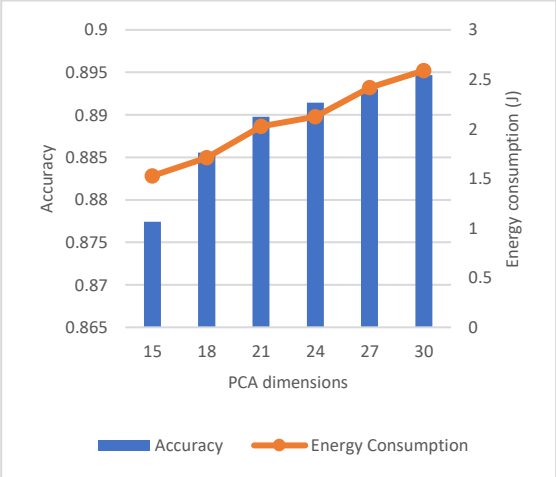

Figure 16: Effect of Dimensionality on Accuracy and Energy consumption (50Hz, HAPT)

By design, the kernel size of the CNN is dependent on the PCA dimensions. As the number of PCA dimensions are increased, the size of the kernel increases, which increases the number of parameters in the dense layer of the CNN. This increase in the number of parameters increases the energy consumption of the network.

5.1.7 Energy Consumption and Pruning

As can be observed in figures 17-20, the energy consumption for inference tasks reduces as the level of pruning increases. This is because pruning reduces the number of connections and neurons in the classifier network, which in turn reduces the number of parameters in the network. As the number of parameters reduces, the energy consumption of the neural network reduces as well. As can be observed, the energy consumption of the classifiers for HAPT dataset are more than those for the WISDM dataset. A major reason for this is the kernel size used for the classifiers, with the kernel size for the HAPT classifiers being twice as big as that for WISDM classifiers (due to twice as many activities to classify in the HAPT dataset). Since the kernel size directly affects the number of parameters of the CNN, the classifiers trained for the HAPT dataset inherently have more parameters.

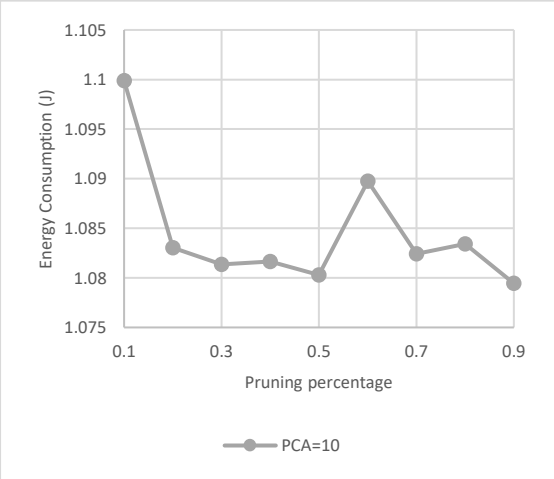

Figure 17: Effect of Pruning on Energy consumption with PCA=10 (WISDM)

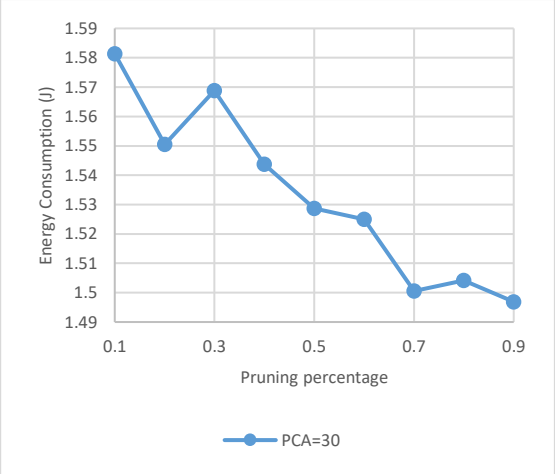

Figure 18: Effect of Pruning on Energy consumption with PCA=30 (WISDM)

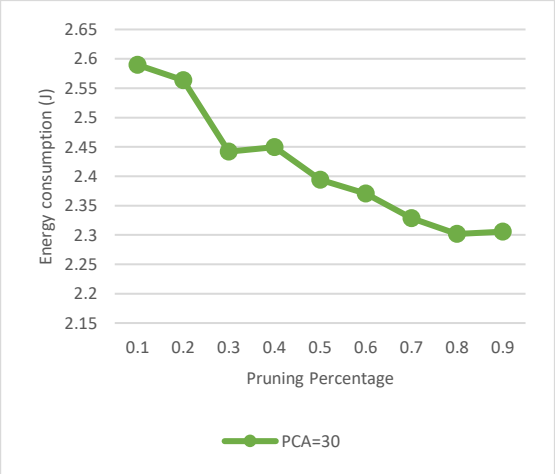

Figure 19: Effect of Pruning on Energy consumption with PCA=30 (50Hz, HAPT)

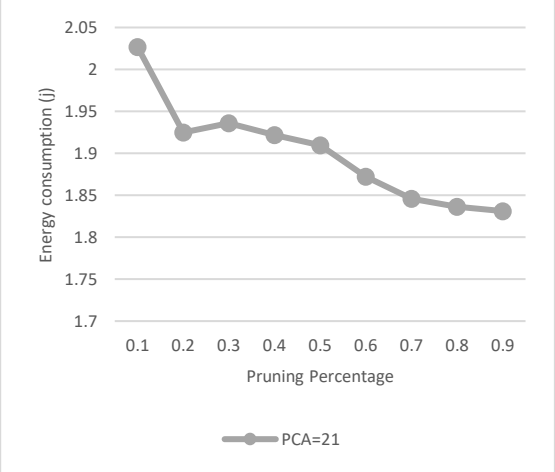

Figure 20: Effect of Pruning on Energy consumption with PCA=21 (50Hz, HAPT)

Table 3 shows the reduction in network parameters that is achieved by pruning the network.

Table 3: Impact of Pruning on the number of network parameters (50Hz, HAPT)

PCA dims	Pruning percentage	Number of CNN parameters
30	0.1	11355
30	0.2	10442
30	0.3	9529
30	0.4	8616
30	0.5	6790
30	0.6	5877
30	0.7	4964
30	0.8	4051
30	0.9	3138

5.1.8 Performance Comparison with other Classifiers

The energy consumption of our pruned CNNs are tabulated against other machine learning classifiers in tables 4 and 5. Tables 4 and 5 show the energy consumption of the classifiers.

Table 4: Comparison of Energy Consumption (J) per Inference (WISDM)

Sampling rate	PCA dims	SVM	KNN	RF	CNN (pruned)
20	7	1.368156 254	1.303820 527	1.365370 75	1.030004 95
20	8	1.472176 766	1.404722 536	1.365616 543	1.076557 3
20	10	1.625521 636	1.554691 029	1.365879 849	1.081347 15
20	20	2.304341 793	2.256927 311	1.366537 651	1.326814 45
20	30	3.593727 446	3.514954 674	1.364904 997	1.568844 55

Table 5: Comparison of Energy Consumption (J) per Inference (HAPT)

Sampling rate	PCA dims	SVM	KNN	RF	CNN (pruned)
20	15	0.680 95691 2	1.89531 8878	1.14027 524	0.890196 7
20	18	0.753 40666 8	2.27600 8093	1.08253 4659	0.940637 509
20	21	0.840 8041	2.31952 1618	1.09094 8713	1.008791 827
20	24	0.943 18097 8	2.42989 5055	1.10897 7079	1.062097 203
20	27	1.002 30406 5	2.57216 2366	1.14833 3013	1.156004 598
20	30	1.046 51093 5	2.76559 6962	1.16860 4136	1.218192 226
25	15	0.971 33072 6	2.45501 4789	1.26878 6764	0.949696 099
25	18	1.117 26529 6	2.74753 4037	1.21966 4645	1.014809 765
25	21	1.239 19701 6	3.02247 3562	1.25844 0363	1.117897 683
25	24	1.327 26346 3	3.18354 5649	1.24894 9409	1.209725 888
25	27	1.421 31291 6	3.54773 8552	1.30835 005	1.254905 105
25	30	1.574 42238 3	3.78321 6143	1.30580 8663	1.556107 202
40	15	2.450 97435 7	4.40506 748	1.77570 5075	1.422598 1
40	18	2.832 33258 7	4.94700 4211	1.71642 1437	1.512317 944
40	21	2.933 40227 6	5.57633 8995	1.76013 0692	1.801396 346
40	24	3.225 18447 6	5.99359 4027	1.81368 0053	1.826097 81
40	27	3.441 66961 9	6.49336 4096	1.88311 0023	2.079557 896
40	30	3.672 43884 8	6.81560 0479	1.91659 0655	2.123178 506
50	15	4.002 44103 7	5.82868 9325	2.08901 8321	1.524766 541
50	18	4.352 52140 8	7.28109 5839	2.02712 6658	1.674859 941
50	21	4.499 14792 8	7.96698 5822	2.07020 9515	1.936128 449
50	24	4.871 04774 7	8.45628 2413	2.13155 5521	2.084341 574
50	27	5.260 31389 2	9.46176 0461	2.20840 5554	2.364618 027
50	30	5.649 68123 4	10.1301 5069	2.25668 9584	2.442194 259

As can be seen from tables 4 and 5, the energy consumption of the pruned CNN is much better than that of the other classifiers. In case of the results of the HAPT dataset, the pruned CNN and Random Forest classifiers are better for higher sampling rates and SVM is slightly better than the CNN for the lower sampling rates.

In tables 6 and 7, the size of the classifiers is compared.

Table 6: Comparison of size in KBs among classifiers (WISDM)

Sampling rate	PCA dims	SVM	KNN	RF	CNN (pruned)
20	7	22	223	1232	10
20	8	22	249	1523	10
20	10	24	301	1659	12
20	20	26	560	1945	21
20	30	29	819	2027	31

Table 7: Comparison of size in KBs among classifiers (HAPT)

Sampling rate	PCA dims	SVM	KNN	RF	CNN (pruned)
20	15	82	372	5601	18
20	18	90	442	5939	21
20	21	100	512	6239	28
20	24	108	583	6672	32
20	27	117	653	6869	39
20	30	125	723	7042	42

6. CONCLUSION AND FUTURE WORK

This thesis addressed the energy and computation overhead in HAR systems by optimizing workflow by using single accelerometer data and selecting minimal and computationally inexpensive features. The trade-off between accuracy and power consumption was studied and an energy efficient 1D CNN was designed and implemented on a Raspberry Pi Zero. By reducing the number of features from 30 to 7, and by pruning the network by a factor of 0.3, the accuracy was sacrificed by 2% for an energy reduction of 34.8%. Furthermore, the size of the network reduced by 67%.

Table 8: Trade-off between accuracy, energy, and size for the CNN

Parameter	Model	
	Final CNN	Initial CNN
Accuracy	0.960365854	0.987042683
Energy Consumption (J)	1.03000495	1.5813746
Size (kb)	10	31

Compared to the best performing baseline model, the Random Forest classifier, our final CNN implementation had a classification accuracy that was 2% less, but it consumed 24% less energy per inference. Compared to the small size of the SVM, the final CNN classifier was 54% smaller. The selection of the Pi Zero further reduced the cost of the system, thereby making it affordable for lower income user groups.

Table 9: Performance comparison between classifiers

Parameter	Model			
	SVM	KNN	RF	Final CNN
Accuracy	0.971037	0.984756	0.989329	0.960366
Energy Consumption (J)	1.368156	1.303821	1.365371	1.03000495
Size (kb)	22	223	1232	10

The implemented model is an offline model, i.e. the data used for training and testing the classifiers does not change. A future improvement can be an online implementation on the Pi Zero, wherein the classifier continuously learns, and the model weights are continuously updated based on new observations. Furthermore, TensorFlow was used as the Keras backend. The TensorFlow team is slowly rolling out the much-optimised version of TensorFlow, the TensorFlow Lite, for low power embedded devices. With the TensorFlow Lite support for the Pi Zero in the nearby future, the CNN models would be even more optimised, and the inference tasks would be more computationally and energy efficient.

BIBLIOGRAPHY

[1] L. Bao and S. S. Intille, "Activity Recognition From User-Annotated Acceleration Data," in *Pervasive Computing; Springer: Heidelberg/Berlin, Germany*, 2004, pp. 1–17.

[2] T. Sztyler and H. Stuckenschmidt, "On-body localization of wearable devices: An investigation of position-aware activity recognition," in *Proceedings of the 2016 IEEE International Conference on Pervasive Computing and Communications (PerCom), Sydney, Australia*, 2016, pp. 1–9.

[3] F. J. Ordóñez and D. Roggen, "Deep convolutional and lstm recurrent neural networks for multimodal wearable activity recognition," *Sensors*, vol. 16, p. 115, 2016.

[4] S. Ha and S. Choi, "Convolutional neural networks for human activity recognition using multiple accelerometer and gyroscope sensors," in *Proceedings of the 2016 International Joint Conference on Neural Networks (IJCNN), Vancouver, BC, Canada*, 2016, pp. 381–388.

[5] E. M. Tapia *et al.*, "Real-time recognition of physical activities and their intensities using wireless accelerometers and a heart rate monitor," in *Proceedings - International Symposium on Wearable Computers, ISWC*, 2007, pp. 37–40.

[6] J. Pärkkä, M. Ermes, P. Korpipää, J. Mäntyjärvi, J. Peltola, and I. Korhonen, "Activity classification using realistic data from wearable sensors," *IEEE Trans. Inf. Technol. Biomed.*, vol. 10, no. 1, pp. 119–128, 2006.

[7] M. Ermes, J. Parkka, and L. Cluitmans, "Advancing from offline to online activity recognition with wearable sensors," in *Engineering in Medicine and Biology Society. 30th Annual International Conference of the IEEE*, 2008, pp. 4451–4454.

[8] D. Figo, P. Diniz, D. Ferreira, and J. Cardoso, "Preprocessing techniques for context recognition from accelerometer data," *Pers. Ubiquitous Comput.*, vol. 14, no. 7, pp. 645–662, 2010.

[9] O. Banos, J. Galvez, M. Damas, H. Pomares, and I. Rojas, "Window size impact in human activity recognition," *Sensors*, vol. 14, no. 4, pp. 6474–6499, 2014.

[10] M. Berchtold, M. Budde, H. R. Schmidtke, and M. Beigl, "An extensible modular recognition concept that makes activity recognition practical," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 2010, vol. 6359 LNAI, pp. 400–409.

[11] S. Reddy, M. Mun, J. Burke, D. Estrin, M. Hansen, and M. Srivastava, "Using mobile phones to determine transportation modes," *ACM Trans. Sens. Networks*, vol. 6, no. 2, pp. 1–27, 2010.

[12] J. Cheng, O. Amft, and P. Lukowicz, "Active capacitive sensing: Exploring a new wearable sensing modality for activity recognition," *Pervasive Comput.*, vol. 6030, no. May 2014, pp. 319–336, 2010.

[13] D. Minnen, T. Westeyn, D. Ashbrook, P. Presti, and T. Starner, "Recognizing soldier activities in the field," *IFMBE Proc.*, vol. 13, pp. 236–241, 2007.

[14] K. Altun and B. Barshan, "Human activity recognition using inertial/magnetic sensor units," *Hum. Behav. Understanding, Lect. Notes Comput. Sci.*, pp. 38–51, 2010.

[15] D. McGlynn and M. G. Madden, "An ensemble dynamic time warping classifier with application to activity recognition," in *Research and Development in Intelligent Systems XXVII, London:Springer*, 2011, pp. 339–352.

[16] Ó. D. Lara, A. J. Prez, M. A. Labrador, and J. D. Posada, “Centinela: A human activity recognition system based on acceleration and vital sign data,” *Pervasive Mob. Comput.*, vol. 8, no. 5, pp. 717–729, Oct. 2012.

[17] D. Anguita, A. Ghio, L. Oneto, X. Parra, and J. L. Reyes-Ortiz, “A public domain dataset for human activity recognition using smartphones,” *ESANN 2013 proceedings, 21st Eur. Symp. Artif. Neural Networks, Comput. Intell. Mach. Learn.*, no. April, pp. 437–442, 2013.

[18] F. R. Allen, E. Ambikairajah, N. H. Lovell, and B. G. Celler, “Classification of a known sequence of motions and postures from accelerometry data using adapted Gaussian mixture models,” *Physiol. Meas.*, vol. 27, no. 10, pp. 935–951, 2006.

[19] P. H. Veltink, H. B. J. Bussmann, W. De Vries, W. L. J. Martens, and R. C. Van Lummel, “Detection of static and dynamic activities using uniaxial accelerometers,” *IEEE Trans. Rehabil. Eng.*, vol. 4, no. 4, pp. 375–385, 1996.

[20] T. Brezmes, J.-L. Gorricho, and J. Cotrina, “Activity Recognition from Accelerometer Data on a Mobile Phone,” in *Distributed Computing, Artificial Intelligence, Bioinformatics, Soft Computing, and Ambient Assisted Living*, 2009, pp. 796–799.

[21] K. Laasonen, M. Raento, and H. Toivonen, “Adaptive On-Device Location Recognition,” in *Pervasive Computing*, 2004, pp. 287–304.

[22] U. Maurer, A. Smailagic, D. P. Siewiorek, and M. Deisher, “Activity recognition and monitoring using multiple sensors on different body positions,” in *Proc. International Workshop on Wearable and Implantable Body Sensor Networks*, 2006, pp. 113–116.

[23] I. H. Witten and E. Frank, *Data Mining Practical Machine Learning Tools and Techniques, Elsevier*. 2005.

[24] K. Altun, B. Barshan, and O. Tunçel, “Comparative study on classifying human activities with miniature inertial and magnetic sensors,” *Pattern Recognit.*, vol. 43, no. 10, pp. 3605–3620, 2010.

[25] J. Quinlan, “C4.5: programs for machine learning, Morgan Kaufmann Publishers,” 1993.

[26] P. Antal, “Construction of a classifier with prior domain knowledge formalised as bayesian network,” in *Proc. 24th Annual Conference of the IEEE Industrial Electronics Society*, 1998, pp. 2527–2531.

[27] H. Zhang, “The Optimality of Naive Bayes,” in *FLAIRS Conference*, 2004.

[28] J. Cheng, R. Greiner, J. Kelly, D. Bell, and W. Liu, “Learning Bayesian networks from data: An information-theory based approach,” *Artif. Intell.*, vol. 137, pp. 43–90, 2002.

[29] C. Cortes and V. Vapnik, “Support Vector Networks,” *Mach. Learn.*, vol. 20, pp. 273–297, 1995.

[30] A. M. Khan, Y. K. Lee, S. Y. Lee, and T. S. Kim, “A triaxial accelerometer-based physical-activity recognition via augmented-signal features and a hierarchical recognizer,” *IEEE Trans. Inf. Technol. Biomed.*, vol. 14, no. 5, pp. 1166–1172, 2010.

[31] J. Penttil, J. Peltola, and T. Seppnen, “A speech/music discriminator based audio browser with a degree of certanity measure,” in *Proc. International Workshop Information Retrieval*, 2001, pp. 125–131.

[32] S. Hochreiter and J. Schmidhuber, “Long Short-term Memory,” *Neural Comput.*, vol. 9, no. 8, pp. 1735–1780, 1997.

[33] M. Stikic, D. Larlus, and B. Schiele, “Multi-graph based semisupervised learning for activity recognition,” in *International Symposium on Wearable Computers*, 2009, pp. 85–92.

[34] M. Stikic, D. Larlus, S. Ebert, and B.

Schiele, “Weakly supervised recognition of daily life activities with wearable sensors,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 33, no. 12, pp. 2521–2537, 2011.

[35] A. Ali, R. King, and G. Z. Yang, “Semi-supervised segmentation for activity recognition with multiple eigenspaces,” in *International Summer School and Symposium on Medical Devices and Biosensors*, 2008, pp. 314–317.

[36] D. Guan, W. Yuan, Y.-K. Lee, A. Gavrilov, and S. Lee, “Activity recognition based on semi-supervised learning,” in *13th IEEE International Conference on Embedded and Real-Time Computing Systems and Applications*, 2007, pp. 469–475.

[37] T. Huynh and B. Schiele, “Towards less supervision in activity recognition from wearable sensors,” in *10th IEEE International Symposium on Wearable Computers*, 2006, pp. 3–10.

[38] Y. Gong, L. Liu, M. Yang, and L. D. Bourdev, “Compressing deep convolutional networks using vector quantization,” *CORR*, vol. abs/1412.6, 2014.

[39] V. Vanhoucke, A. Senior, and M. Z. Mao, “Improving the speed of neural networks on cpus,” *Deep Learn. Unsupervised Featur. Learn. Work. NIPS*, 2011.

[40] S. Gupta, A. Agrawal, and P. Gopalakrishnan, K. Narayanan, “Deep learning with limited numerical precision,” in *Proceedings of the 32nd International Conference on International Conference on Machine Learning*, 2015, pp. 1737–1746.

[41] S. Srinivas and R. V. Babu, “Data-free parameter pruning for deep neural networks,” in *in Proceedings of the British Machine Vision Conference 2015, BMVC 2015, Swansea, UK, September 7-10*, 2015, pp. 31.1–31.12.

[42] S. Han, H. Mao, and W. J. Dally, “Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding,” in *International Conference on Learning Representations (ICLR)*, 2016.

[43] W. Chen, J. Wilson, S. Tyree, K. Q. Weinberger, and Y. Chen, “Compressing neural networks with the hashing trick,” in *JMLR Workshop and Conference Proceedings*, 2015.

[44] K. Ullrich, E. Meeds, and M. Welling, “Soft weight-sharing for neural network compression,” *CORR*, vol. abs/1702.0, 2017.

[45] V. T. van Hees *et al.*, “Separating Movement and Gravity Components in an Acceleration Signal and Implications for the Assessment of Human Daily Physical Activity,” *PLoS One*, vol. 8, no. 4, pp. 1–10, 2013.

[46] G. M. Weiss and J. W. Lockhart, “The impact of personalization on smartphone-based activity recognition,” *AAAI Work. - Tech. Rep.*, vol. WS-12-05, pp. 98–104, 2012.

[47] J. R. Kwapisz, G. M. Weiss, and S. A. Moore, “Activity Recognition using Cell Phone Accelerometers,” 2010.

[48] J. L. Reyes-Ortiz, L. Oneto, A. Samà, X. Parra, and D. Anguita, “Transition-Aware Human Activity Recognition Using Smartphones,” *Neurocomputing*, vol. 171, pp. 754–767, 2016.

[49] J. Morales and D. Akopian, “Physical activity recognition by smartphones, a survey,” *Biocybern. Biomed. Eng.*, vol. 37, no. 3, pp. 388–400, 2017.

[50] F. Chollet and others, “Keras.” GitHub, 2015.

[51] G. Varoquaux, L. Buitinck, G. Louppe, O. Grisel, F. Pedregosa, and A. Mueller, “Scikit-learn,” *GetMobile Mob. Comput. Commun.*, vol. 19, no. 1, pp. 29–33, 2015.

[52] S. Van Der Walt, S. C. Colbert, and G. Varoquaux, “The NumPy array: A structure for efficient numerical computation,” *Comput. Sci. Eng.*, vol. 13, no. 2, pp. 22–30, 2011.

[53] M. Abadi *et al.*, “TensorFlow: Large-

Scale Machine Learning on Heterogeneous Distributed Systems," 2016.

[54] "Using the SavedModel format." [Online]. Available: https://www.tensorflow.org/guide/saved_model.