MSc Data Science

20COP328

F033806

# Long Short-Term Memory Recurrent Neural Networks for Human Activity Recognition: A Comparative Study

by

**Arlen McKinnon**

Supervisor: Dr Axel Finke

Department of Computer Science

Loughborough University

August 2021

## ABSTRACT

Human activity recognition (HAR) – the process of using machine learning algorithms to build models which classify human movement data into a set of predefined activity categories – has become a field of intense study in recent years. Over the last decade, many researchers have reported achieving state-of-the-art performance by employing deep learning algorithms, such as long short-term memory recurrent neural networks (LSTMs), in their HAR models. However, there is a lack of standardisation of state-of-the-art performance in HAR research. Thus, we compared standardised versions of four commonly used LSTM architectures on their performance on two commonly used HAR datasets to more definitively determine which LSTM varieties show the most promise for HAR applications. We found that the CNN-LSTM architecture showed the most promise, reaching high levels of performance with minimal training time.

We also attempted to improve the performance of our models in several ways. Firstly, we tested the efficacy of using a two-stage classification approach which first divided the data into stationary or dynamic movement categories before classifying the data into a specific activity category. This classification approach was largely unsuccessful, as it increased training times whilst rarely improving classification performance compared to the models built with a standard one-stage classifier.

Secondly, two data manipulation techniques were applied to the datasets, the results from which were inconclusive. The first, oversampling, was not found to improve model performance, though we propose that it would be worthwhile to investigate using other, more advanced oversampling methods than the one we employed in our experiments. The second data manipulation technique, feature selection, improved performance in one dataset but not the other.

## TABLE OF CONTENTS

# 1 INTRODUCTION

The goal of human activity recognition (HAR) is to categorize data, obtained from a person performing different movements, gestures, or activities, into a set of predefined activity categories. HAR has been a field of interest for several decades, relying on data acquired from video cameras (Aggarwal & Cai, 1997) or sensors attached to the body (Mäntyjärvi et al., 2001). More recently, with the advent of smartphones, smartwatches, and other portable devices featuring motion sensors such as accelerometers and gyroscopes, sensor-based HAR has become an area of more intense study. Due to the pervasiveness of such portable devices, obtaining data for HAR research has never been cheaper or easier, and HAR is being used in an ever-widening array of applications. To date, HAR has proven useful in fields as diverse as healthcare (Mazilu et al., 2014), fitness coaching (Kranz et al., 2013), robotics (Piyathilaka & Kodagoda, 2015), and surveillance (Qin et al., 2016).

The sensor-based HAR process has four main stages. In the first stage, the data is pre-processed to convert it into a format which is suitable for analysis. The raw data is typically obtained as a long list of sensor readings taken at equal timesteps; usually, these readings are taken several times per second over a period between several minutes and an hour (Yang et al., 2015). An important part of the pre-processing stage is segmentation, where the data is converted from the long list of timesteps into smaller segments, usually a few seconds long. This allows the HAR algorithm to learn how to classify these small segments of data, enabling it to quickly recognise activities when used in real-world applications.

The second stage is the feature extraction stage, where it is decided which aspects of the data are most useful as features to be inputted into the HAR algorithm. Until recently, the feature extraction stage was performed manually, with researchers using hand-crafted mathematical transformations of the data – such as variance, mean, and amplitude – as features (e.g. Ben-Arie et al., 2002; Mäntyjärvi et al., 2001). However, such hand-crafted features rely heavily on the domain knowledge of researchers, work poorly for more complex activities, and do not generalise well (Wang et al., 2018).

The final two stages of HAR – training and classification – are performed using machine learning techniques, such as Bayesian networks (Liu et al., 2018), logistic regression (Farah et al., 2019), random forests (Nunes et al., 2017), or artificial neural networks (Murad & Pyun, 2017). In the training stage, the algorithm is shown many examples of segments of activity data, with each segment labelled as a certain activity category. The algorithm then learns to associate certain movement data with certain activities. After several thousands of examples, the algorithm is trained, and can then be used to classify new movement data in the classification stage.

Fortunately, with the rapid and intense innovation of deep learning techniques over the last decade, the feature extraction stage no longer needs to be performed manually. Deep learning algorithms – such as convolutional neural networks (CNNs) and long short-term memory recurrent neural networks (LSTMs) – are able to perform the feature extraction, training, and classification stages automatically, in an end-to-end fashion. This leads to more generalisable and accurate HAR models which require little domain knowledge to build (Ronao & Cho, 2016). Deep learning algorithms, and how they are used in HAR, will be discussed in more detail in sections 1.1 and 1.2, respectively.

## *1.1 DEEP LEARNING*

The term 'deep learning' refers to a subset of machine learning algorithms which use several layers of neural networks to extract information from data. Neural networks are a type of machine learning algorithm developed based on how the brain is thought to operate, with a network of interconnected neurons which pass information on from one layer to the next to learn and make decisions based on input data. The layers in a deep learning model are arranged hierarchically, with each layer extracting higher level information. For example, a lower layer may detect edges in an image, while higher layers may use the information passed on from the lower layers to recognise faces, animals, or objects. How much information is passed from one layer to the next depends on the weights between the neurons in each layer.

Deep learning models are stochastic in nature, meaning that there is a level of randomness and variability in the outputs produced by a model. Deep learning models are trained through backpropagation where, after each data instance is inputted to the algorithm to produce an output, the algorithm uses the error in the output to adjust the weights between neurons in the network so that the error is reduced. The weights between neurons in a model are initialised randomly, and so the same model trained on the same data may learn to give different weights between its neurons each time it is trained, thus producing a different output each time.

Currently, CNNs and LSTMs are two of the most popular deep learning algorithms for many different applications. CNNs are neural networks which process inputted data that has a grid-like shape, such as an image, using convolutional layers. Due to the shape of data inputted into CNNs, they are most often used for image processing (Perez-Munuzuri & Chua, 1993) and classification (Ren et al., 2017) purposes, though they have also been applied to HAR (Zeng et al., 2015).

Figure 1 shows a diagram of a simple CNN architecture. In the convolutional layers of a CNN, a filter is applied to the input data which moves across the input data step by step, extracting the relevant features found. This feature map is then passed onto a pooling layer, which reduces the size of the feature map whilst keeping the important information. This reduced feature map is then fed into the fully connected layers – 'fully connected' meaning each node in the layer is connected with every node in the next layer – which perform classification. A CNN algorithm can be made up of several of each of these layers; AlexNet (Krizhevsky et al., 2012), for example, had five convolutional layers, three pooling layers, and three fully connected layers.

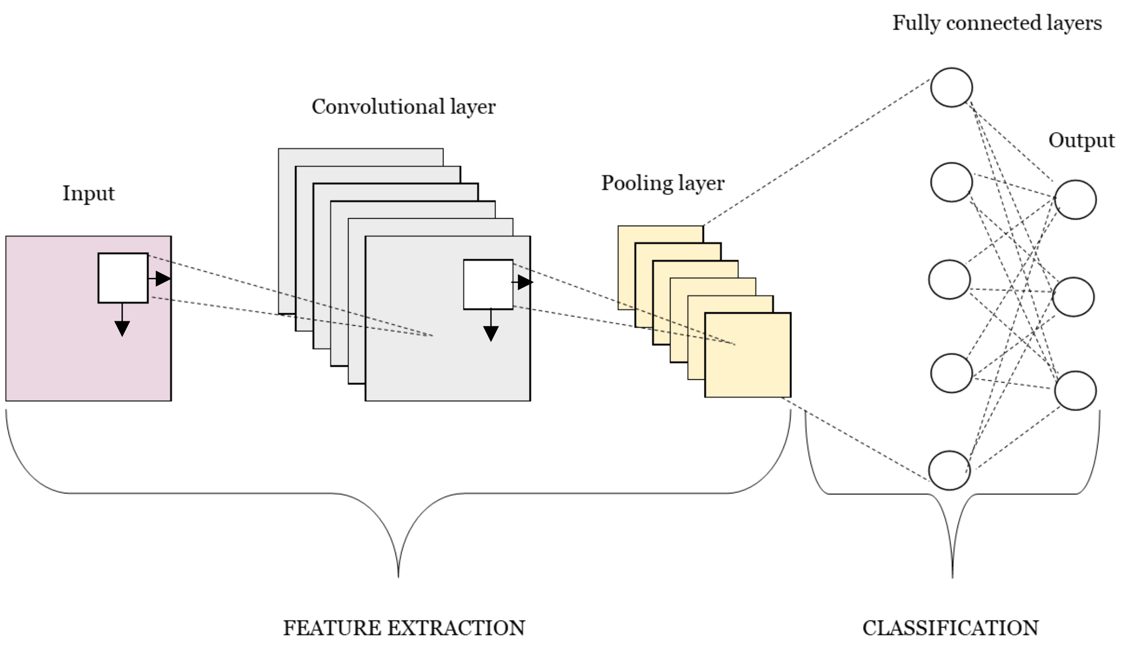


*Figure 1: Diagram of how a basic CNN performs classification. Adapted from Phung & Rhee (2019).*

LSTMs are a special type of recurrent neural network (RNN) capable of learning long-term dependencies, which means that they can store and remember important information from early on in a sequence of data that can then be used later on by the model. Due to their ability to learn such long-term dependencies, LSTMs are particularly useful for tasks involving long sequences of data, such as speech recognition (Graves et al., 2013), time-series prediction (Schmidhuber et al., 2005), and HAR (Chen et al., 2016).

In a traditional RNN, the network contains loops which allow information to be kept from one input to the next (see Figure 2). They do this by taking the outputted result from a previous input and feeding this output back to the network along with the next input, and so on. However, these loops do not allow information to be kept for very long – they are not capable of learning long-term dependencies.

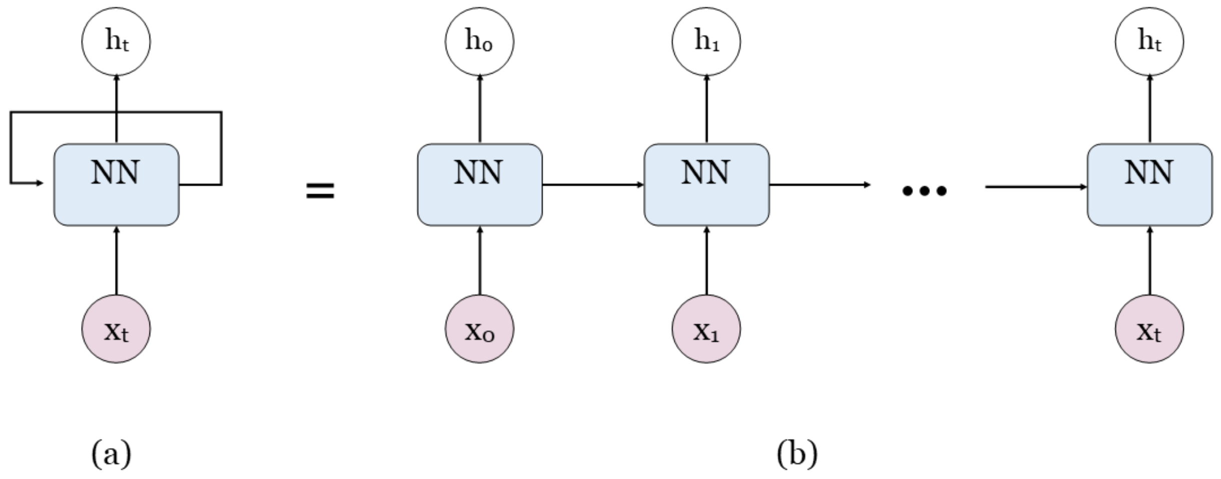


*Figure 2: Diagram of how the output (h) from an input (x) to a neural network layer (NN) is fed back to the NN as an input in an RNN. (a) shows the diagram with the loop, and (b) shows the loop unravelled, more clearly displaying how the output from input t-1 is fed into the network along with input t. Adapted from Olah (2015).*

LSTMs were developed by Hochreiter and Schmidhuber (1997) specifically to address this problem. LSTMs keep the loops found in RNNs, but instead of the loop just containing one neural network layer, LSTMs contain four, each with a specific purpose. These four neural network layers interact with each other to form gates, which control which information is kept and which is discarded (see Figure 3). Any information that is kept is stored in the cell state (the thick arrow in Figure 3). The cell state is the key component in an LSTM because information stored in the cell state changes very little from input to input, allowing relevant information to be stored in the network for much longer than in a traditional RNN.

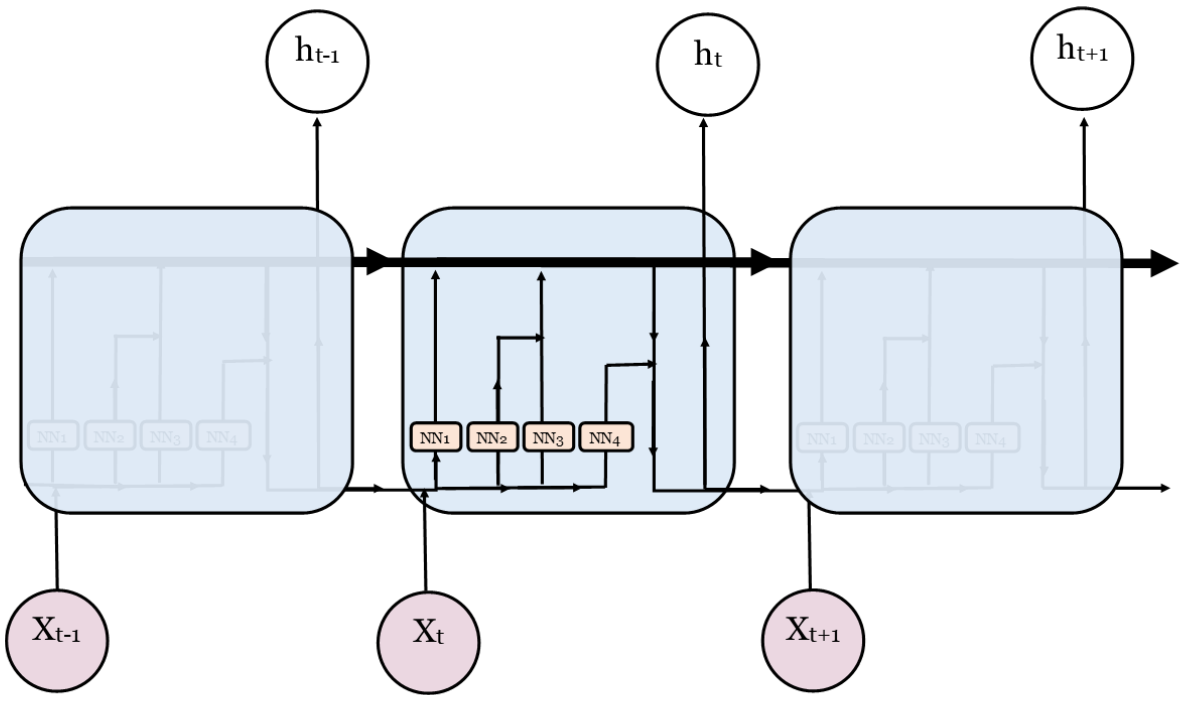


*Figure 3: Diagram of an LSTM layer, unravelled as in Figure 2b. The four neural network layers (NNs) in the LSTM interact to form gates, which decide what information is stored in the cell state (represented by the thicker horizontal arrow running through the top of the LSTM). The cell state changes little from one input (x) to another, meaning that information can be stored over long sequences of inputs. Adapted from Olah (2015).*

### *1.2 DEEP LEARNING AND HAR*

Much research in the last decade has focused on the use of deep learning algorithms in HAR. Earlier research focused on the use of multi-layer perceptrons (Dernbach et al., 2012; Kwapisz et al., 2011), relatively simple neural network architectures where each node in a layer is connected to every node in the next layer. However, more recent research has focused more intently on other architectures, such as CNNs and LSTMs. CNNs are well suited for HAR problems as, by their default nature, they are very good at extracting features from data, and are also capable of learning local dependencies – though not long-term dependencies. Numerous papers have been published which detail research into the use of CNNs in HAR and much success has been found, with many papers reporting higher classification performance than more traditional HAR methods (for a review, see; Wang et al., 2018)

LSTMs are another intuitive architecture for HAR purposes, and this is the area we focus on in this paper. LSTMs, unlike other architectures, are capable of learning long-term dependencies – a particularly useful capability in HAR problems. When a segment of movement data is inputted into the LSTM, the algorithm can remember relevant information from the start of the segment throughout, meaning that it considers the whole segment of data when classifying the segment into a certain activity category.

In HAR research, LSTMs have seen a similar level of success as CNNs in terms of classification performance. However, as noted by Chen and colleagues (2021), there is a distinct lack of standardisation of state-of-the-art performance in HAR research. Evaluation metrics, and experimental settings such as the time taken to train models, hardware used, and proportion of data allocated to training and testing datasets, vary rather a lot from one paper to another. This has led many researchers to claim that their models are achieving state-of-the-art performance (e.g. Jiang & Yin, 2015; Ronao & Cho, 2016; Zhu & Qiu, 2016), but there is no way of fairly comparing the performance of one researcher's model to another. This is an issue that we aimed to address in this current paper.

## *1.3 OUR CONTRIBUTIONS*

Our experiments, described in this paper, aimed to compare four common LSTM architectures that have been used for HAR purposes. Our purpose was, similarly to Greff and colleagues (2017), not to achieve state-of-the-art performance but to compare standardised versions of the architectures in practical terms – finding which architectures provide the best classification performance to computation time trade-off. It was hoped that the results from our experiments would provide a starting point for other researchers, which they could build upon to achieve state-of-the-art performance.

Deep learning based HAR models are often computationally expensive, and this is due to two main reasons. Firstly, the models require a large amount of data to be trained to an effective level. The way activities are performed varies form one person to another, and different devices may record these activities slightly differently, and so it is important that the HAR model is trained to a level where it can cope with such variability (Chen et al., 2021). Secondly, the most accurate HAR models (e.g. Ordóñez et al., 2016; Mekruksavanich & Jitpattanakul, 2021) often use deep architectures, which increases computation time. As our focus was on focus on finding the best model in practical terms, the architectures used in our models were relatively shallow.

Because many HAR applications are executed on devices such as smartphones and smartwatches – which use much less computing power than HAR models are typically trained and tested on – it was deemed important to find models that are both accurate and computationally lightweight enough to be used by such devices. Therefore, models were set a time limit of five minutes for training, after which point the training process ended – though most models converged to a certain accuracy well before this five-minute time limit. The architectures used in our experiments are detailed in section 2.

Two versions of each of the four architectures were tested in our experiments. First, the architectures were tested as standard, one-stage classification models. Second, the architectures were tested using a two-stage classification model, where the model first classified an activity as either stationary or dynamic, and then classified activities it classed as stationary into a higher-level stationary activity category, and vice-versa (see section 2.6 for a more detailed explanation of this approach). Such an approach has been found to increase HAR classification performance using CNN architectures (Cho & Yoon, 2018), and so we thought it would be worthwhile to explore weather this would have any effect on the performance of LSTM models in our experiments. This gave a total of eight models which were compared in our experiments.

As the purpose of our experiments was to compare the performance of different architectures in a standardised way, we chose to use two datasets which have been commonly used in HAR research over the last decade. The first dataset used was the University of California, Irvine (UCI) Smartphones dataset (Anguita et al., 2013), and the second was the wireless sensor data mining (WISDM) dataset (Weiss et al., 2019). Both datasets provided somewhat different challenges, which was intended to bring further validation to our results; if a model performed particularly well across both datasets, it could be said with a good degree of confidence that this was a good model for HAR purposes. The datasets are described in detail in section 3.

Data manipulation techniques were also applied to each dataset. The WISDM dataset had a class imbalance problem, and machine learning models trained on such datasets are often biased towards the most common class (Japkowicz, 2000). Thus, oversampling was applied to it to attempt to address this. A random forest classifier was also used on both datasets to select each dataset's most important features and reduce dimensionality, a technique which has been found to reduce computational cost and improve performance on HAR problems (Mohammed Hashim & Amutha, 2021). We wanted to see if we could replicate such improvements on our models. These data manipulation techniques are described in more detail in section 4.

In summary, our main contributions are as follows:

- We compared several common LSTM architectures in a standardised way, to determine more definitively which types of architecture perform best for HAR purposes.
- Rather than focusing on achieving state-of-the-art performance, we compared models in practical terms: which models gave the best training time to accuracy trade-off?

- We trained and tested models on two commonly used, publicly available datasets. Each dataset provided somewhat different challenges, allowing us to determine if the relative performance of each model was consistent across datasets.
- We applied data manipulation techniques to both datasets, to observe what effect this had on performance.

## *1.4 EXPERIMENTAL OVERVIEW*

This subsection was intended to give a brief overview of the experimental procedure we used in our study. More detail will be given in the following sections, but this subsection should help to give some context to the more detailed explanations which follow.

Two different Jupyter notebooks were created for the purposes of this study – one for each dataset. Models were built on each dataset separately, in their respective notebook; we were not comparing the results from one dataset to another as such, rather we chose to build the same models on two datasets to assess whether models that performed particularly well on one dataset would perform similarly well on another.

The experimental procedure was similar for both datasets. The datasets were first loaded, and then pre-processed into a format suitable to be inputted into the LSTM models. The datasets were also split into a training dataset and a testing dataset. Four LSTM architectures were used, each of which will be explained in detail in the following section. Each of the four architectures were built using both a standard one-stage classification approach and a two-stage "divide-and-conquer" (DAC) approach. These eight different LSTM models were trained and tested on both pre-processed datasets in the first phase of analysis.

In the second phase of analysis, oversampling was applied on the WISDM dataset, which originally had a class imbalance problem. The eight LSTM models were then trained and tested on this oversampled dataset.

In the third phase of analysis, a random forest classifier was applied to each dataset for the purpose of feature selection, resulting in both datasets having fewer features than they had originally. The eight LSTM models were then trained and tested on both of these datasets with reduced features.

Due to the stochastic nature of deep learning models, we repeated the training and testing process of each model ten times, to account for the variation in outputs produced each time by the same model. The mean classification performance of each model was compared visually using plots, as well as analytically using Welch's t-test (Welch, 1947) to determine whether any differences in model performance were significant. We found that the standard deviations in the mean performances of our models varied from model to model and so Welch's t-test was deemed an appropriate test, as it does not assume equal standard deviations between means.

All analysis undertaken as part of this overall experiment was accomplished using Python 3.9 in Jupyter notebooks. The Keras deep learning package was used to implement all models. Model training and testing was accomplished using a personal computer with an *AMD Ryzen 7* CPU and *NVIDIA GeForce GTX 1660 SUPER* GPU.

## 2 MODELS

This section describes the four types of LSTM architecture compared in our experiments, as well as explaining the two-stage DAC classification approach we tested in addition to the one-stage classification approach.

### *2.1 STANDARDISING MODELS FOR FAIR COMPARISON*

As the purpose of our experiments was to compare different models in a fair and standardised way, it was important that we implemented the models using consistent hyperparameters and similar degrees of complexity. For this reason, each neural network layer in every architecture we built contained 100 neurons. Categorical cross entropy was chosen as the loss function for all models as it is well-suited to multi-class classification tasks such as ours and is commonly used by other HAR researchers (e.g. Mutegeki & Han, 2020; Yao et al., 2017).

With our focus on finding the best models in practical terms in mind – models which achieved good accuracy but also trained quickly – we decided to use relatively basic architectures for each of the models. Thus, none of the architectures used were particularly deep, each containing close to the minimal number of layers needed to build an effective model. To reiterate, the purpose of our experiments was not to achieve state-of-the-art performance but to compare some common architectures in a fair and standardised way. The results of our comparisons could be used as a starting point in future research and built upon to then achieve state-of-the-art performance.

### *2.2 VANILLA LSTM*

The first architecture built for our experiments was a 'vanilla' LSTM – the standard LSTM architecture with no additional features (see Figure 4). Such models have seen some success in HAR research (Chen et al., 2016; Singh et al., 2017), though more recent research has largely moved towards more complex models, such as those detailed in following subsections of this section. The architecture we built here was rather basic, with an LSTM layer followed by a dropout layer to reduce overfitting, and then two fully connected layers to perform classification.

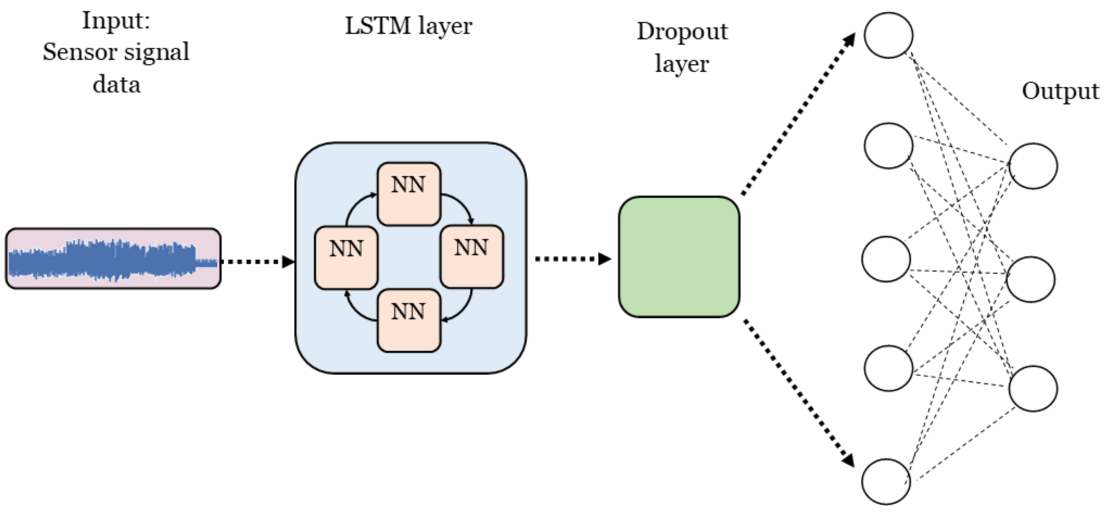


*Figure 4: Diagram of the vanilla LSTM architecture we used. The LSTM layer in this architecture contained four interacting fully connected neural network layers (NNs).*

### *2.3 CNN-LSTM*

The second architecture built for our experiments was a CNN-LSTM (see Figure 5). In a CNN-LSTM model, the feature extraction step is performed by convolutional layers, which then feed these extracted features to an LSTM layer for classification. CNN-LSTMs have been used to build models with very high accuracies in recent years (Mekruksavanich & Jitpattanakul, 2021; Mutegeki & Han, 2020); the paper which originally used this CNN-LSTM architecture (Sainath et al., 2015) also reported that the architecture outperformed a standard LSTM on a set of vocabulary tasks. The CNN-LSTM architecture built for our experiments made use of two convolutional layers for feature extraction, followed by a dropout layer, before the output from these layers was fed into an LSTM architecture with the same structure as the vanilla LSTM detailed above.

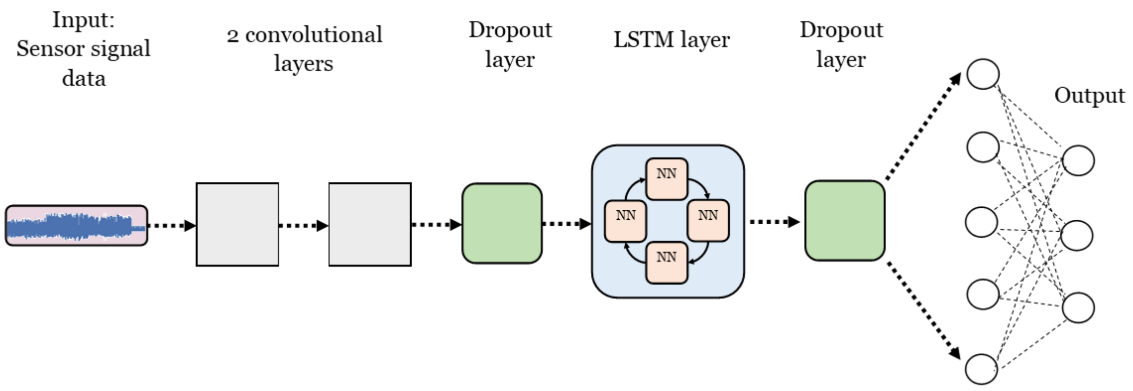


*Figure 5: Diagram of the CNN-LSTM architecture we used.*

## 2.4 *CONVLSTM*

A ConvLSTM was built as the third architecture to use in our experiments (see Figure 6). A ConvLSTM is a special type of LSTM where, instead of using fully connected neural network layers to form gates within the LSTM, convolutional layers are used. This type of architecture has also been applied to HAR problems to achieve very high accuracy scores in recent years (Li et al., 2020; Singh et al., 2021). The ConvLSTM architecture used in our experiments followed a very similar structure to our vanilla LSTM, with a ConvLSTM layer followed by a dropout layer and two fully connected layers.

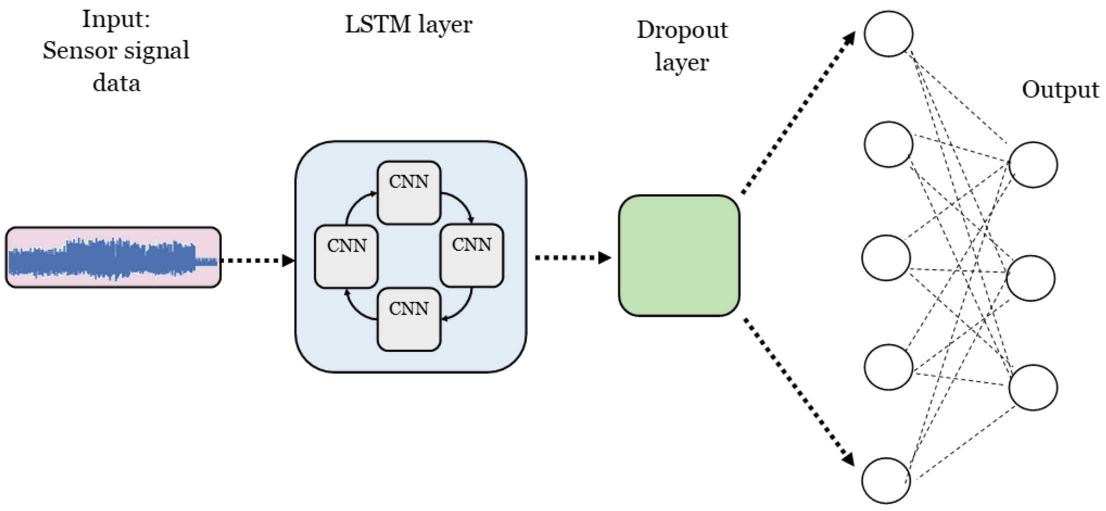


*Figure 6: Diagram of the ConvLSTM architecture used. The LSTM layer in this architecture contained four interacting convolutional neural network layers (CNNs).*

## 2.5 *BI-DIRECTIONAL LSTM*

The fourth architecture used in our experiments was a bi-directional LSTM (see Figure 7). A bi-directional LSTM differs from a normal LSTM because it runs in both the forwards and backwards directions (see Figure 8), in contrast to a normal LSTM which only runs in the forward direction. This means that bi-directional LSTMs can remember information from both the past and the future parts of a sequence, meaning that the model can understand context even better than standard LSTM models. Again, several papers have been published reporting state-of-the-art results achieved with bi-directional LSTM models (Pigou et al., 2018; Zhao et al., 2018). In our experiments, the bi-directional LSTM was also implemented similarly to the vanilla LSTM, with a bidirectional LSTM layer followed by a dropout layer and two fully connected layers.

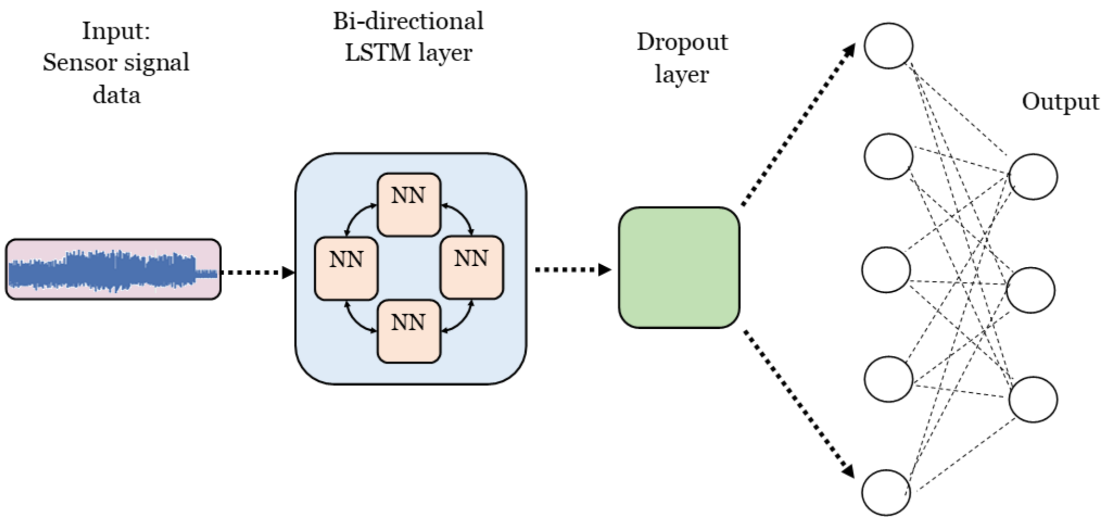


*Figure 7: Diagram of the bi-directional LSTM architecture used. The architecture was very similar to the vanilla LSTM architecture, but a bi-directional LSTM layer took the place of the standard LSTM layer. The bi-directional flow of information in the bidirectional LSTM layer is represented by the double-headed arrows in the layer.*

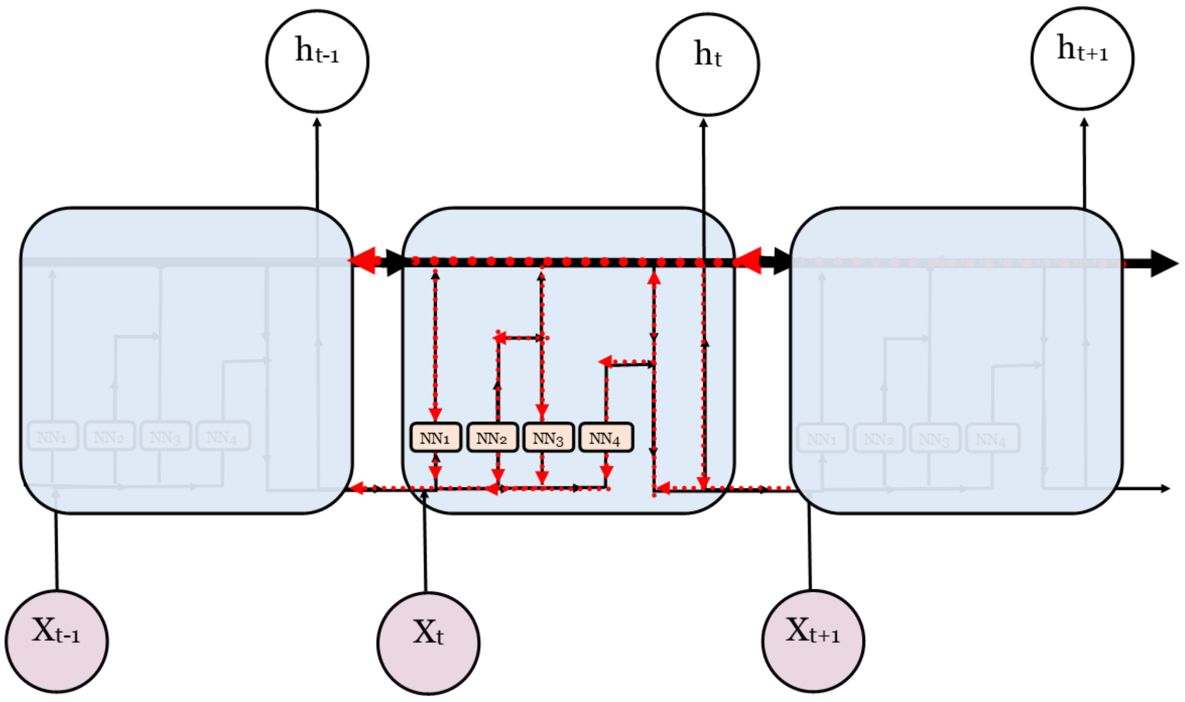


*Figure 8: Diagram of the bi-directional LSTM layer. The dotted red arrows show the information moving from front to back, whereas the solid black arrows show information moving from back to front. Adapted from Olah (2015).*

## *2.6 DIVIDE-AND-CONQUER MODELS*

The four architectures described above were used to build two types of models. First, we built standard one-stage classification models, and then we also built two-stage DAC models (see Figure 9). In the DAC models, the first stage in the model classified a segment as either belonging to a stationary class or a dynamic class. After classifying a segment into one of these two lower-level categories, the second stage of the model then classified the segment into a more specific, higher-level class. This type of approach has been found to improve HAR performance in CNN models (Cho & Yoon, 2018), as well as performing well in multi-layer perceptron models (Khan et al., 2010). We wanted to see if this type of model could lead to better performance in models employing LSTM architectures.

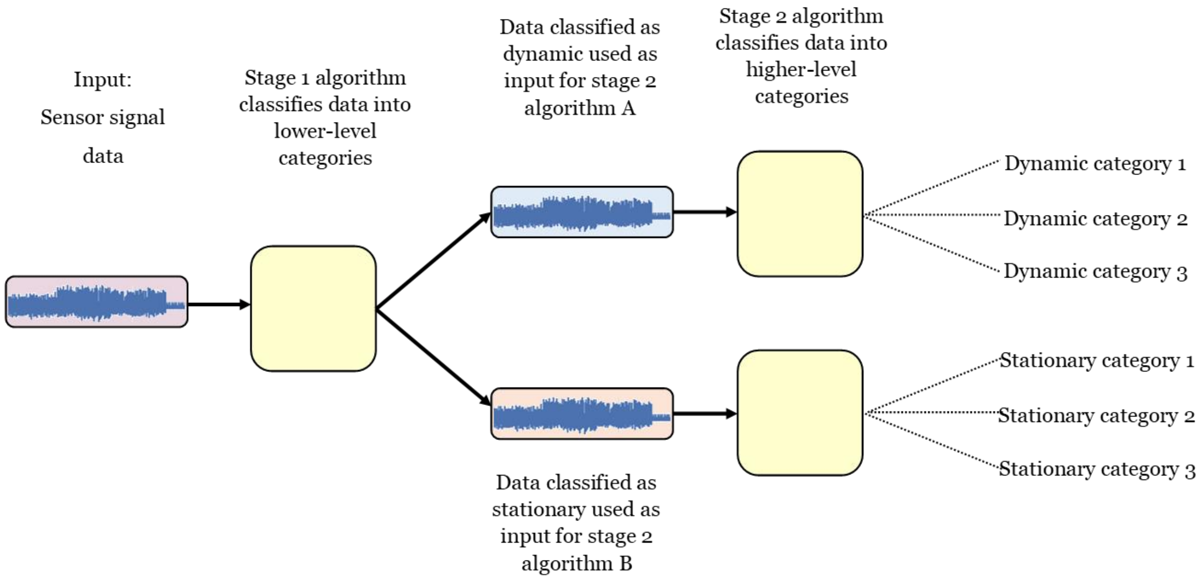

Input:
Sensor signal data
Stage 1 algorithm classifies data into lower-level categories
Data classified as dynamic used as input for stage 2 algorithm A
Stage 2 algorithm classifies data into higher-level categories
Dynamic category 1
Dynamic category 2
Dynamic category 3
Stationary category 1
Stationary category 2
Stationary category 3
Data classified as stationary used as input for stage 2 algorithm B

*Figure 9: Diagram of the divide-and-conquer approach. The algorithm used here, represented by the yellow square, was any one of the four LSTM architectures detailed above.*

One potential problem with using this approach in our experiments was that this type of model is more complex than a single stage classifier, which does go against what we were trying to achieve in our experiments. However, we allowed these DAC models the same amount of training time as the single stage models to offset this issue. The first stage classifier was allocated 30 seconds to run, and the two second stage classifiers were allocated two minutes and fifteen seconds each. Cho and Yoon (2018) noted that for a successful DAC model to be built, the first stage of the classifier must be very accurate when assigning segments to one of the two lower-level movement categories. In our experiments, this first-stage classifier did usually reach very high accuracy levels — often above 99% — within 30 seconds.

## 3 DATASETS

The two datasets we used to train and test our models on were selected due to them both being frequently used in HAR research, and the fact that they pose somewhat differing challenges to the process of building a successful HAR model. This section will describe both datasets in detail.

## *3.1 UCI SMARTPHONES DATASET*

The UCI Smartphones dataset contained data from a group of 30 participants who performed a set of six standard activities whilst wearing a Samsung Galaxy II smartphone attached to their waist. The smartphone recorded 3-axial (x, y, and z) linear acceleration using its accelerometer and 3-axial angular velocity using its gyroscope, at a constant rate of 50Hz, while the participants performed these activities. Video recordings were also taken of the participants while they performed the activities, allowing the researchers to accurately label the data captured by the smartphone. The six activities performed by participants were:

1. *Walking*
2. *Walking Upstairs*
3. *Walking Downstairs*
4. *Sitting*
5. *Standing*
6. *Laying*

The raw data from these experiments was not publicly available; however, two pre-processed versions of the data could be downloaded from the UCI Machine Learning Repository website. Both available versions of the dataset had been split into training and testing datasets with a 70:30 ratio and noise filters had been applied to the data. The data had also been segmented already, with the accelerometer and gyroscope sensor signals split into 2.56 second segments – corresponding to 128 readings per segment.

Feature engineering had been applied to the first version of the dataset, resulting in 561 hand-crafted features commonly used in HAR research. Although some researchers have used this version of the dataset (e.g. Zhao et al., 2018), in our research we aimed to compare models which were generalisable and applicable to a range of different datasets. Thus, it was preferable to build models which performed feature extraction themselves in an end-to-end fashion, rather than relying on hand-crafted features selected by experts for a specific dataset.

Fortunately, the second available version of the dataset contained only the recordings from the gyroscope and accelerometer; the accelerometer readings had been split into total acceleration and the estimated body acceleration, which was obtained by subtracting gravitational acceleration from total acceleration. This resulted in a total of nine features, made up of the three axes of gyroscopic, total acceleration, and body acceleration readings.

This version of the dataset allowed us to extract features from this sensor signal data, train models, and test models in an end-to-end fashion.

Before training any models on it, the dataset was explored to determine if there were any important characteristics to the dataset that we needed to be aware of. First, we visualised the class distribution in both the testing and training datasets, to see if any activities were recorded in drastically differing quantities to each other (Figure 10). The data did appear to be spread reasonably evenly between the six classes, in both the training and testing datasets, and the small variations seen between numbers of segments in each activity category were unlikely to cause any issues to the HAR models built.

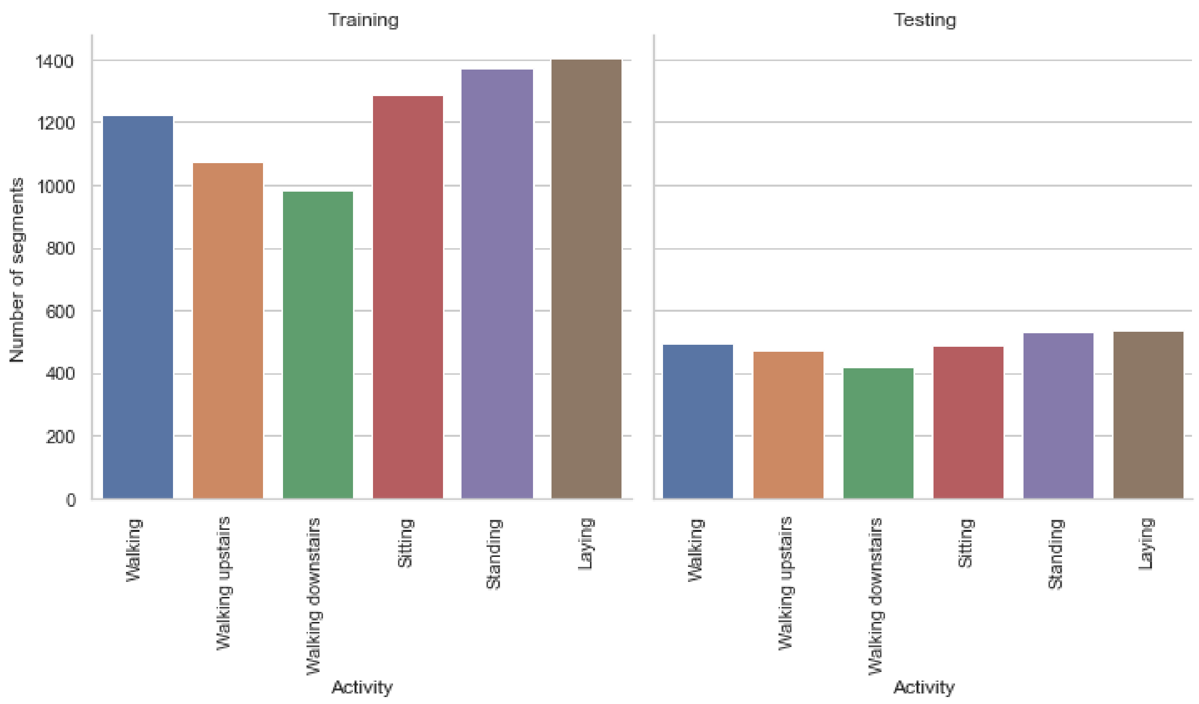


*Figure 10: Class distribution of the training and testing UCI Smartphones datasets.*

In Figure 11, we visualised the class distribution for each participant, where there also seemed to be a relatively even spread of segments in each activity category. To make the visualisation clearer, ten participants were selected at random from the dataset here; see Appendix A for a visualisation of the data distribution for all participants.

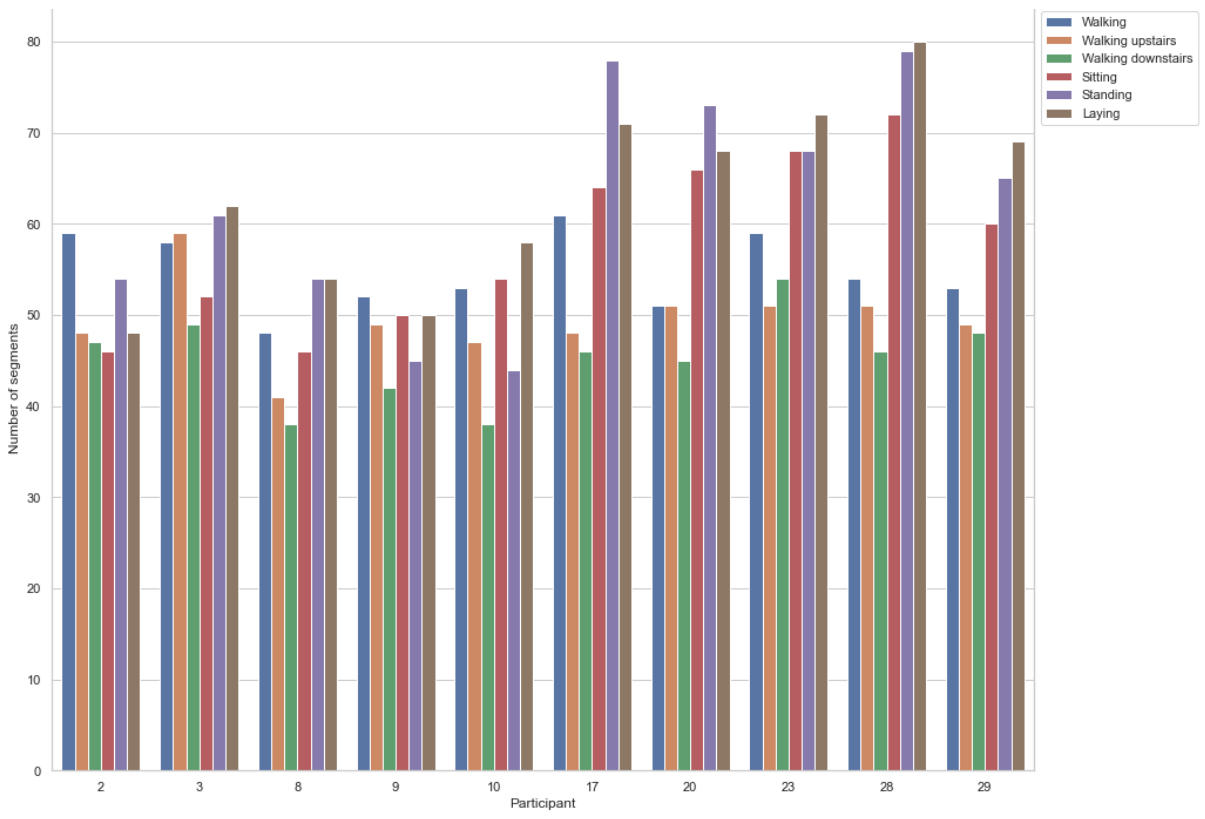


*Figure 11: Class distribution for ten randomly selected participants from the UCI Smartphones dataset.*

To visualise the sensor signal data, and how it related to each activity, the values recorded for each axis of each sensor were plotted for a single participant (see Figure 12). The activity at each corresponding point in time was plotted underneath the nine sensor signal plots, making it possible to what the sensor signal data for each activity looked like. For example, the transition from activity 6 (*laying*) to activity 1 (*walking*) shortly after the red vertical line drawn on the plot corresponded closely to a transition to much more vigorous activity in all sensors. This makes sense intuitively, giving us confidence in the quality of the dataset and confirming that we had loaded the dataset correctly.

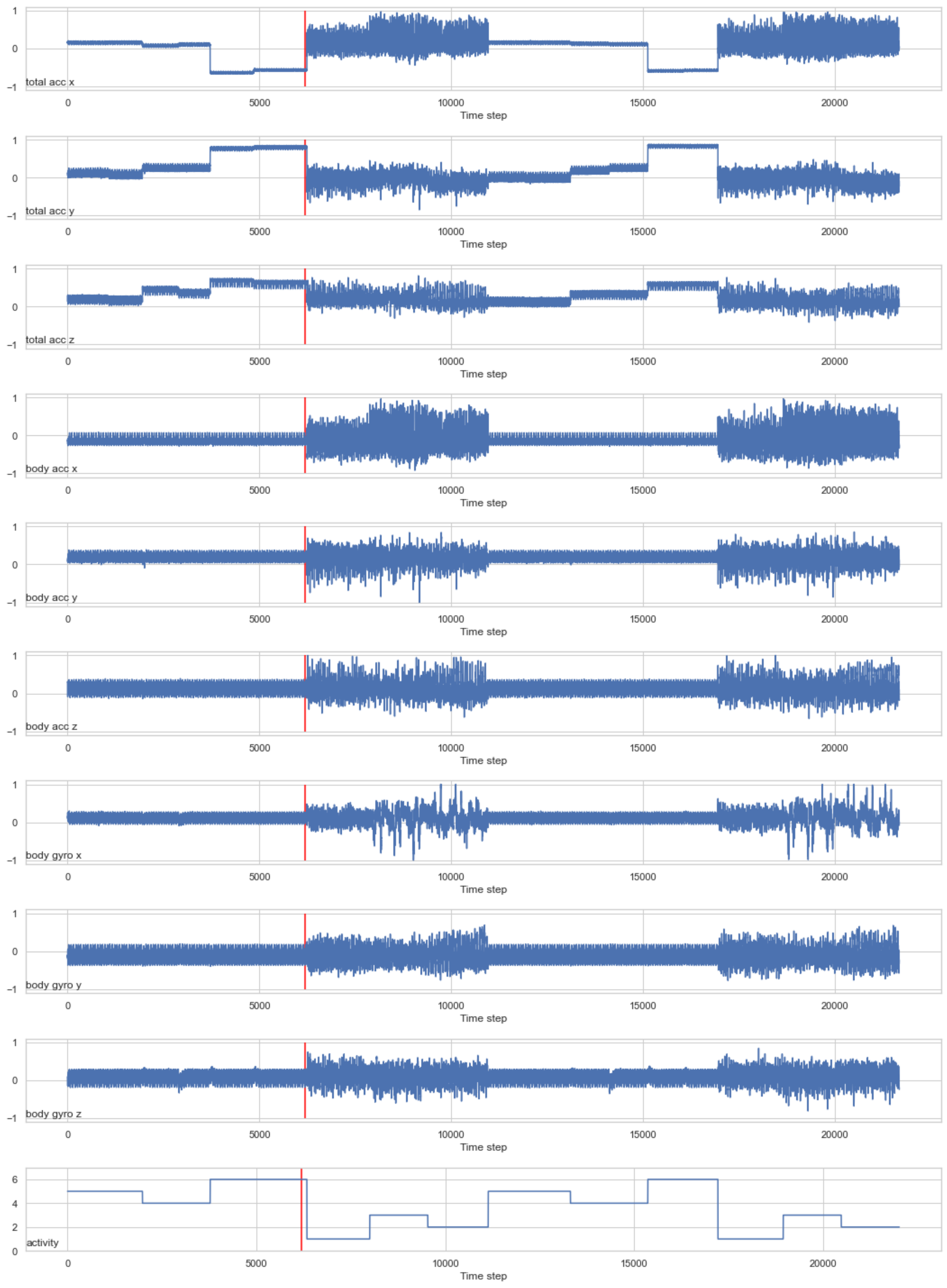


*Figure 12: Visualising the UCI Smartphones sensor data for a single participant. The bottom plot shows the activity being performed, and the red line is an arbitrary point chosen to show the transition from one activity to another.*

As the UCI smartphones dataset had already been segmented and split into a training and testing set, little more pre-processing needed to be done to the dataset before it could be used to train and test models. Still, the data needed to be normalised, as inputting non-normalised data into machine learning algorithms can cause the algorithms to interpret features with larger scale values as being more important, which degrades the quality of any model built. Normalisation can also speed up the training process in neural networks (Chen et al., 2016). For our experiment, the training dataset was used to build a scaler which scaled the values for all features between -1 and 1. This same scaler was then applied to the testing dataset.

The data was then reshaped into a shape suitable for input into the LSTM models – a three-dimensional array with the shape [*samples, timesteps, features*], where *samples* was the number of segments of signal data, *timesteps* was the number of signal readings per segment, and *features* represented each of the nine features in the dataset. After this, the training data was an array of shape [7352, 128, 9] and the testing data was an array of shape [2947, 128, 9].

## *3.2 WISDM DATASET*

The WISDM dataset contained data from 36 participants who performed a slightly different set of six standard activities whilst carrying an android smartphone in their front pocket. The smartphone recorded three-axial linear acceleration using its accelerometer at a constant rate of 20Hz while participants performed these activities. This contrasted with the UCI Smartphones dataset, which employed a gyroscope in addition to an accelerometer as well as recording data 2.5 times as often. The six activities performed by participants were:

1. *Walking downstairs*
2. *Jogging*
3. *Sitting*
4. *Standing*
5. *Walking upstairs*
6. *Walking*

Unlike the UCI Smartphones dataset, the raw data for the WISDM dataset was publicly available online. This raw data needed to be segmented and split into a test and training set before analysis. We decided to split the data into segments containing 42 timesteps; this was in the optimal range found by Ignatov (2016) when testing deep learning models with several different segment sizes in the WISDM dataset, and covered a similar amount of time (2.1 seconds) as the segments covered in the UCI Smartphones dataset (2.56 seconds). We also decided to use the same training and testing split as used by Ignatov, using the first 26 participants' data as a training dataset and the remaining ten participants' data as a testing dataset. This resulted in the data being split in a 71:29 ratio, also similar to the 70:30 split used in the UCI Smartphones dataset.

The rest of the pre-processing steps undertaken on the WISDM dataset were similar to those undertaken on the UCI Smartphones dataset. The training dataset was normalised between -1 and 1 using a scale built on the training dataset, and this scale was then applied to the testing dataset. The dataset was then reshaped into an array with the shape [*samples, timesteps, features*]. After this, the training data was an array of shape [37068, 42, 3] and the testing data was an array of shape [15224, 42, 3]. The WISDM dataset contained around five times as many samples as the UCI Smartphones dataset.

Unlike the UCI smartphones dataset, the WISDM dataset had a serious class imbalance problem (see Figure 13). *Walking* and *jogging* segments were recorded much more often than the other activities; *walking*, the most common class, outnumbered *standing*, the least common class, almost 9:1. This was likely to cause our models built on this dataset to be biased towards classifying segments into the *walking* or *jogging* categories, negatively affecting performance. In a later stage of analysis, oversampling was performed on the dataset to attempt to address this, which is explained in more detail in section 4.

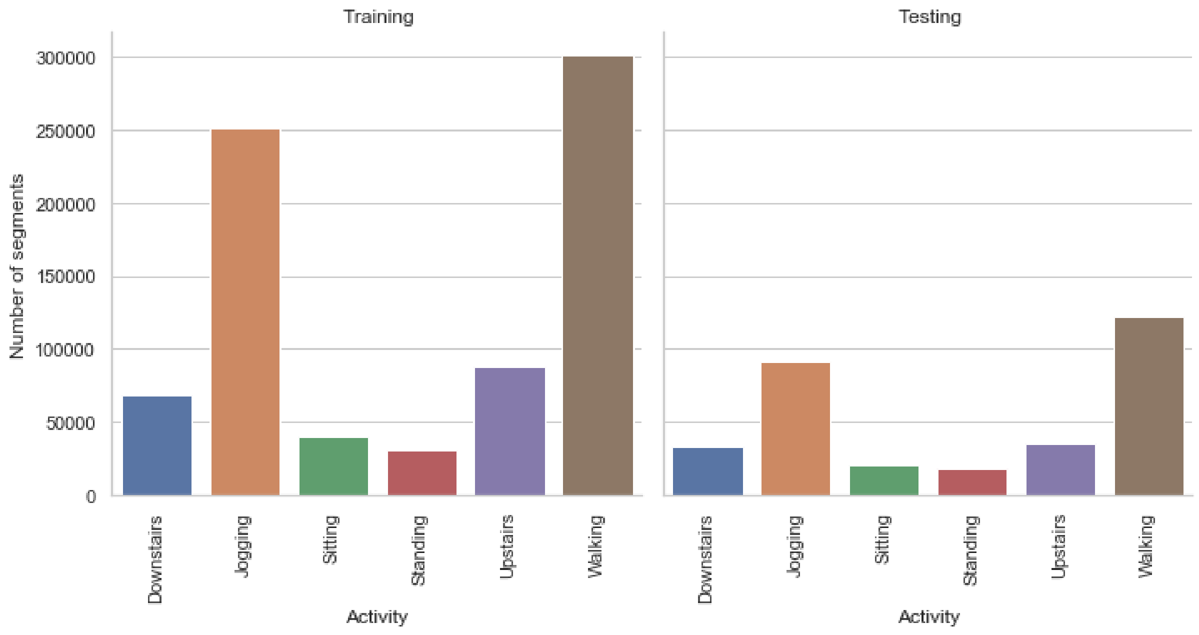


*Figure 13: Class distribution of the training and testing WISDM datasets. This visualisation clearly shows the class imbalance problem present in the dataset.*

Figure 14 visualises the class distribution for selected participants in the WISDM dataset. Similarly to the corresponding visualisation in the UCI Smartphones dataset, only ten random participants were shown for the sake of clarity; a visualisation of the data distribution for all participants can be found in Appendix B. Here, we can see that some participants did not record any data at all for some activities, and most participants recorded much more data for *jogging* and *walking* than for other activities. The fact that some participants recorded no data at all for some activities was also likely to be another source of error in our models – models should be trained on data from a wide range of different people to be able to cope with the variability in how different people perform different activities (Chen et al., 2021).

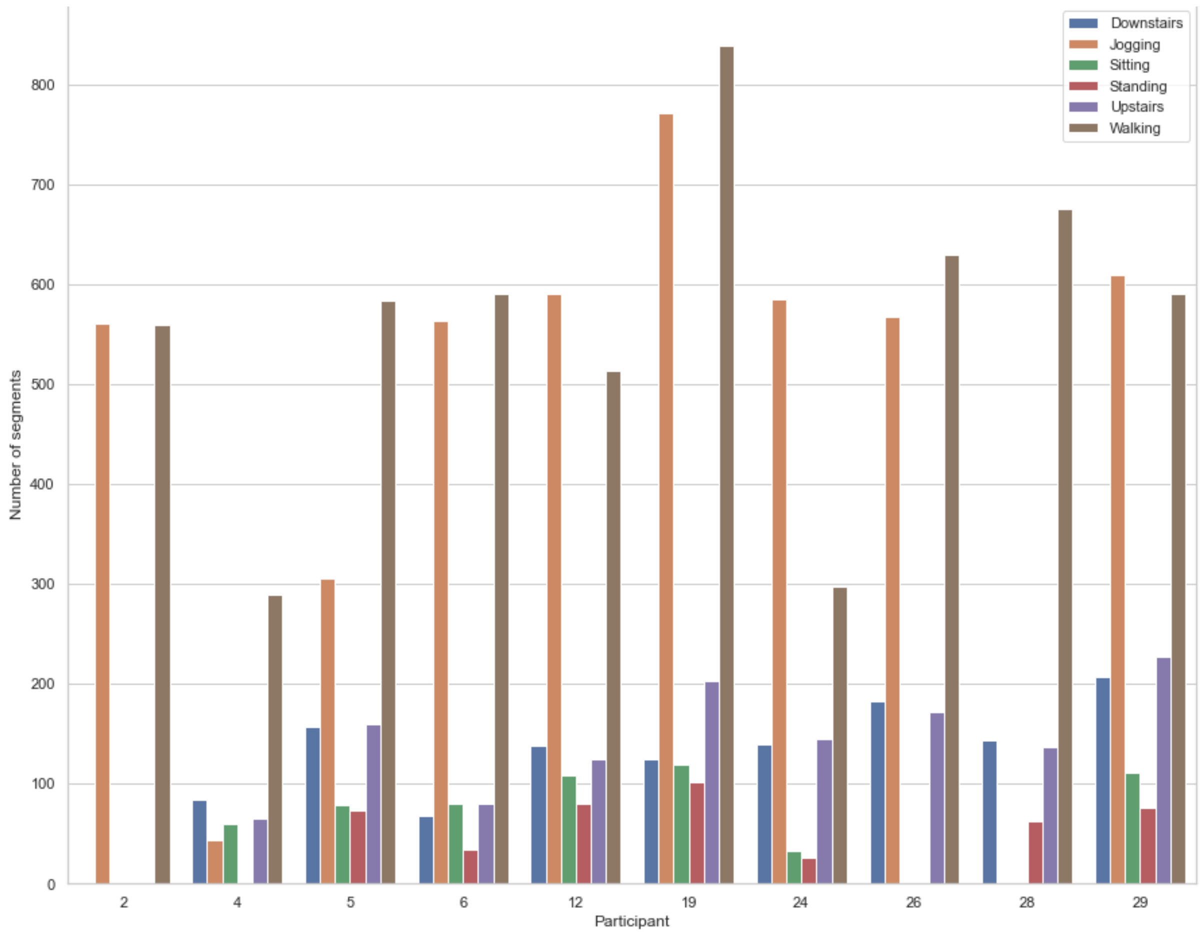


*Figure 14: Class distribution for ten randomly selected participants from the WISDM dataset.*

The sensor signal data and corresponding activities for one participant in the WISDM dataset was plotted in Figure 15. Once again, it is possible to see the transitions between different activities by looking at the plot. The data was strangely bunched up around 1 for much of the x and y data, particularly for the *jogging* activity. At first, we thought this may be due to some error in the pre-processing steps we applied to the dataset, such as the normalisation step. However, the original data had this bunching artefact as well (see Appendix C). This is possibly due to how the data was recorded — rather than attaching the smartphone to the participant to record the data, participants kept the device in their pocket. This meant that the device likely moved around in a much more jerky and unnatural way, causing the accelerometer to measure linear acceleration at consistently high values. This was another potential source of inaccuracies in our models trained on this dataset.

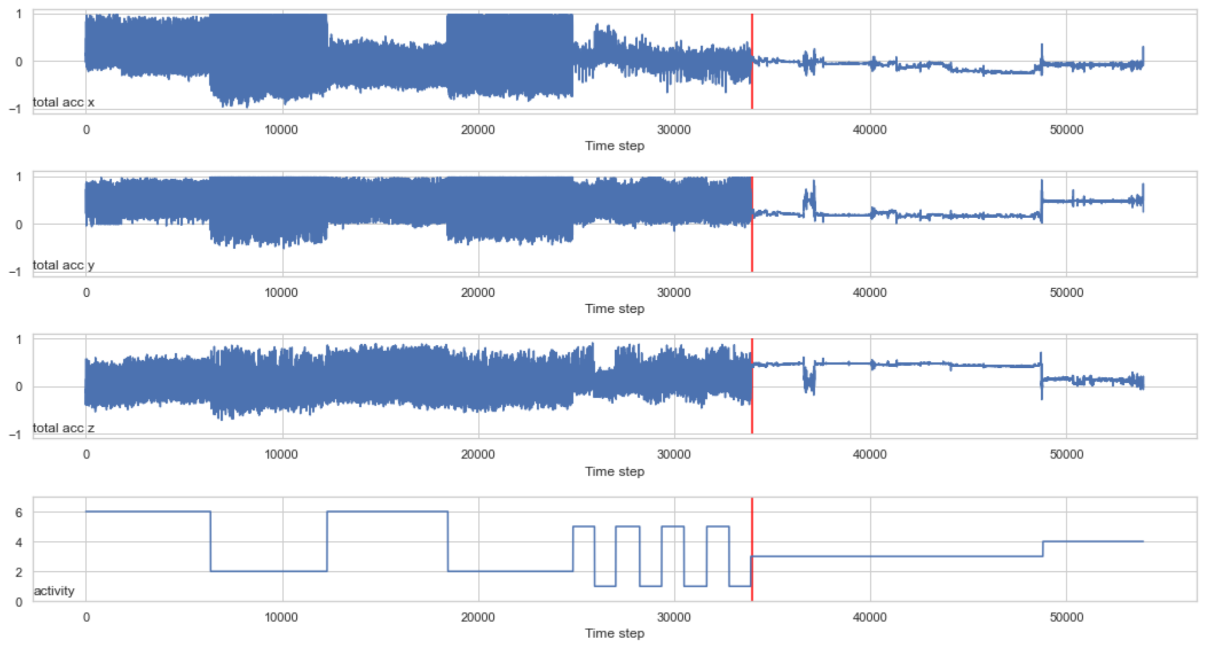


*Figure 15: Visualising the WISDM sensor data for a single participant. The bottom plot shows the activity being performed, and the red line is an arbitrary point chosen to show the transition from one activity (walking downstairs) to another (sitting).*

## 4 DATA MANIPULATION TECHNIQUES

### *4.1 OVERSAMPLING*

The WISDM dataset had a class imbalance problem, with samples from the most common class outnumbering samples from the least common class in a ratio of almost 9:1 in the training dataset. Machine learning models trained on such imbalanced datasets are often biased towards the most common class (Japkowicz, 2000), reducing the quality of the model, and so researchers have developed several ways of counteracting this.

Resampling methods are one such approach to mitigating the effects of imbalanced data. These methods can be divided into two main categories: undersampling, where samples in the most common classes are removed to bring the number of samples down to a more even spread between classes; and oversampling, where synthetic samples of the least common classes are added to the dataset to bring the number of samples up to a more even spread between classes.

Undersampling has some significant disadvantages when compared to oversampling. Firstly, deleting so many samples from a dataset can lead to valuable information being lost (Buda et al., 2018). Also, deep learning methods require large amounts of data to be trained effectively. In our case, the variation between how different people perform activities means that a large number of sensor signal segments, from a wide range of people, is required to train models effectively (Chen et al., 2021). Thus, we decided against employing undersampling in our experiments, choosing to oversample less-common classes instead. Oversampling is not without disadvantages, namely that it can cause models to become overfitted on training data, but we felt that it was still the preferable strategy for our specific application.

There are several ways in which oversampling can be implemented. In HAR research, Alani and colleagues (2020) used synthetic minority oversampling technique (Chawla et al., 2002), which creates artificial samples which lie in-between two real samples in the feature-space of the dataset. However, this method does not preserve the temporal structure of segments in HAR data, and so is not well-suited for HAR applications. Instead, we decided to use a more tailored approach, in which we made exact copies of segments (see Figure 16) in less common classes and added them to the dataset until the classes were sufficiently balanced (see Figure 17). The second stage of our experiments involved training and testing our eight LSTM models on this oversampled WISDM dataset.

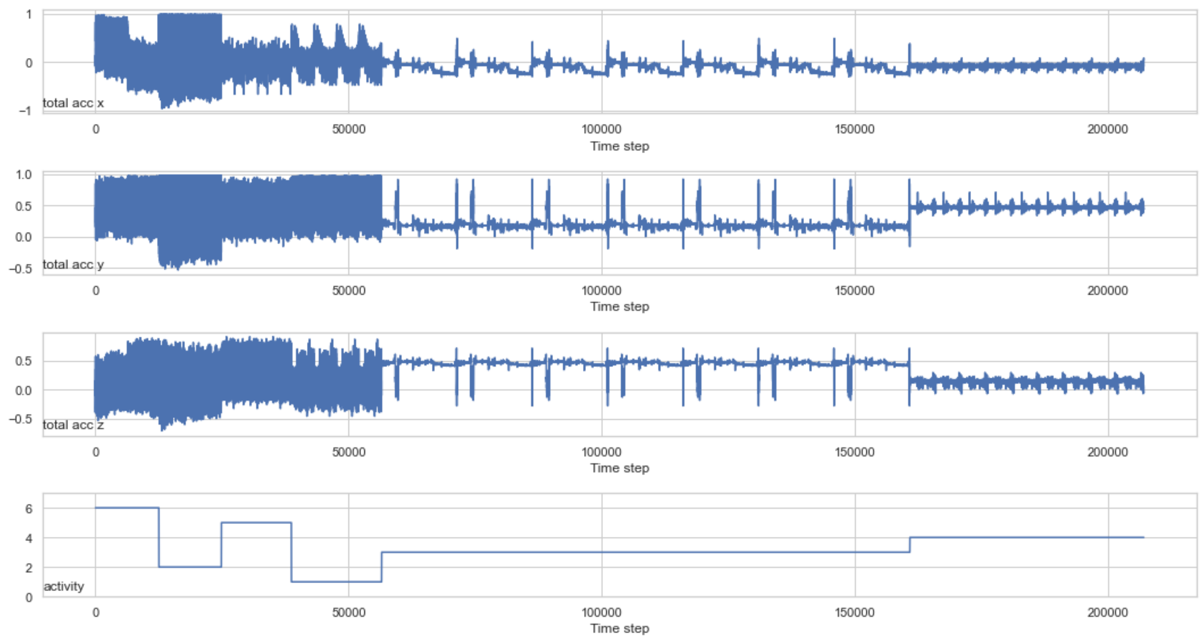


*Figure 16: Plot of the sensor signal data for one participant in the WISDM dataset after oversampling was applied.*

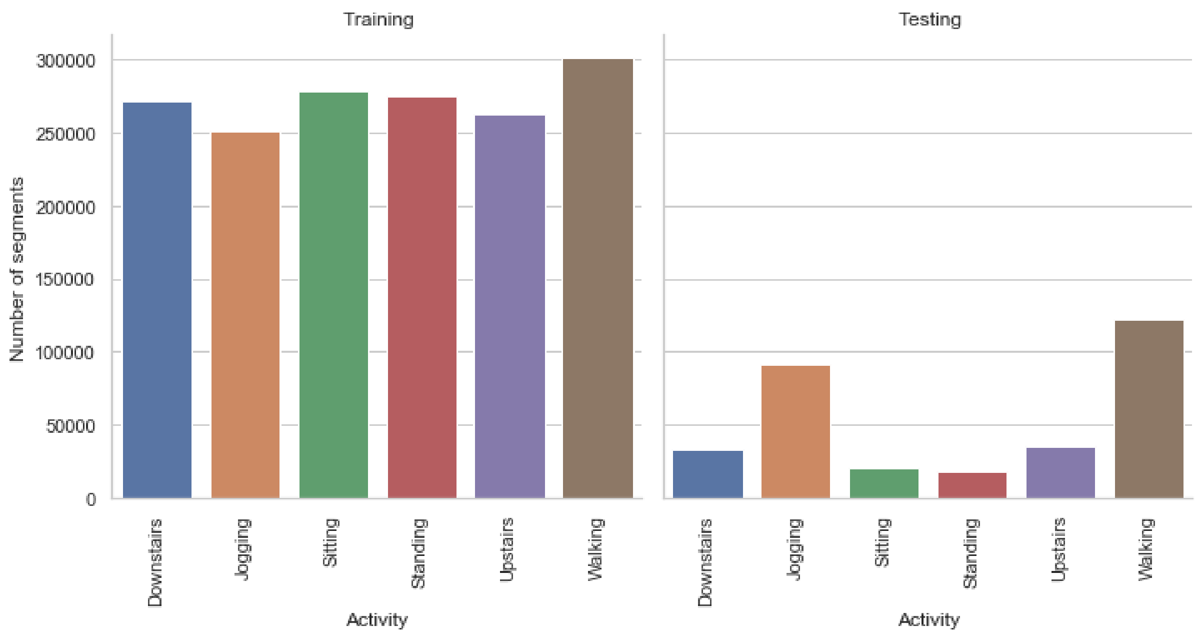


*Figure 17: Visualisation of the class distribution in the WISDM dataset after oversampling was applied. The training dataset was much more balanced, but the testing dataset remained unchanged.*

## *4.2 FEATURE SELECTION*

Feature selection can have two main benefits on machine learning models. Firstly, selecting the most important features and discarding less important ones reduces the number of variables a model has to process, reducing computational cost (Cai et al., 2018). This was of particular value to the purposes of our experiment – reducing the computational cost of our models would make them more lightweight and reduce training times, potentially making more effective models for HAR applications.

The second benefit feature selection can have on machine learning models is that it can improve classification performance (Cai et al., 2018; Mohammed Hashim & Amutha, 2021). This benefit is achieved through removing features which are not relevant to the class variable of the dataset. Thus, using feature selection in our models potentially provided the dual benefit of increasing the accuracy of our models as well as making them quicker to train. The third stage of our experiments was completed to determine if either of these benefits could be seen in our models.

A random forest classifier was used to select the most important features in both datasets (Genuer et al., 2010). Random forest is a flexible machine learning algorithm which can be used in a wide range of applications, and has been applied successfully for feature selection in HAR research (Uddin & Uddin, 2015). Random forest uses an ensemble of decision trees to select the most important features for predicting the class in a dataset, using a combination of the results from the ensemble of decision trees to improve confidence in the overall result.

When the random forest classier was applied to the UCI Smartphones dataset, it found that three of the nine sensor signal features in the dataset were sufficiently important to be included in building models, namely:

- *Total acceleration (X)*
- *Total acceleration (Y)*
- *Total acceleration (Z)*

This is interesting, as these are the three sensor signal features provided in the WISDM dataset, and suggest that the use of the gyroscope, and working out the bodily acceleration from the total acceleration by accounting for gravitational acceleration, were not necessary for this dataset. The third stage in our experiments tested whether this was true.

When the random forest classier was applied to the WISDM dataset, it found that only one sensor signal feature, *total acceleration (Y)* was important for predicting the class variable, *activity*. The third stage of our experiments also trained and tested models on this version of the WISDM dataset using this single predictor variable.

## 5 RESULTS: UCI SMARTPHONES DATASET

This section and the following section report the results from our experiments. This section specifically details the results from the UCI Smartphones dataset, first comparing the eight models built on the original dataset with each other (section 5.1), then comparing the eight models built on the dataset with reduced features with each other (section 5.2). Section 5.3 compares the results achieved on the original dataset with those achieved after feature selection. Section 5.4 compares the results achieved using the one-stage approach against those achieved using the DAC approach. All visualisations in these two sections, and all metrics reported, were comparing the models on their ability to classify movement data in each dataset's respective test dataset.

The metrics reported in this section and the next are the mean metrics for each model, calculated from the ten repeats each model was run for. We used two different metrics to compare the performance of our models. Like many other HAR researchers, we used classification accuracy as one of these metrics (for a review, see; Minh Dang et al., 2020). Accuracy is simply the total number of samples a model predicts correctly in a test dataset, divided by the total number of samples in a dataset. Accuracy is often a useful metric, but it does have some downfalls. For example, if 100 people were tested for a disease, and only five of those people did have the disease, a model which predicted all 100 people to not have the disease would achieve an accuracy score of 95%, which is reasonably high. However, such a model is actually rather useless, as it is not capable of detecting if somebody has a disease – a potentially fatal mistake. So, accuracy is not always a great metric to use, especially in examples where the dataset has imbalanced classes such as above.

The WISDM dataset also had a class imbalance problem, and it was important that we could measure whether the models we built could detect all activity categories in both datasets. Thus, we needed to use another metric than accuracy to compare our models – one which would take account of the models' ability to recognise each activity category, even when a category was less common. F1 score is often used as a metric for models built on imbalanced datasets for this reason (Murad & Pyun, 2017), and so we decided to use F1 score as our second metric to compare models against each other.

F1 score is a measure of the balance of *precision* and *recall*. Precision is defined as the number of correctly classified positive results, divided by the total number of results predicted as positive by a model:

$$Precision = \frac{True\ positives}{True\ positives + False\ positives}$$

Recall is defined as the number of correctly classified positive results, divided by the total number of actual positive results in the dataset:

$$Recall = \frac{True\ positives}{True\ positives + False\ negatives}$$

F1 score is calculated as:

$$F1\ score = \frac{Precision \times Recall}{Precision + Recall}$$

Using F1 score, rather than accuracy, is optimal for comparing models built on imbalanced datasets as it considers the precision and recall of each class equally. This means that a model which is biased towards the most common class in an imbalanced dataset may achieve high accuracy scores, but its F1 score will suffer.

We used Welch's t-tests used to compare models analytically, to check whether any differences in F1 score after the models had been trained for five minutes were significant. A significance level of 0.95 was adopted here, meaning that we could be 95% certain any differences found were legitimate.

Tables were used to display the metrics obtained from our experiments on each model. To make comparisons between the metrics of models easier, DAC models were displayed in italics. In tables showing side-by-side comparisons between models, significant differences were shown in bold.

We also compared the performance of models in this and the following section visually by plotting the mean test accuracies of each model over time. We achieved this by calculating the mean accuracy of the model at each epoch and the mean time elapsed at each epoch, over the ten repeats, and then plotting these mean values. Due to small differences in how long it took for the same model to be trained per epoch, this was an approximation; however, for the purposes of visually comparing the mean accuracy achieved by each model over time, we deemed this to be a sufficient approximation.

In the plots, for the sake of brevity, vanilla LSTM models were referred to as 'LSTM' and bi-directional LSTM models were referred to as 'Bi-LSTM'. Also, some plots contained bars around the lines plotted for each model. These bars represented one standard deviation in test accuracy at each point in time for each model.

## *5.1 ORIGINAL DATASET*

This subsection reports the results of our experiments on the original UCI Smartphones dataset – before feature selection. Table 1 displays the mean final accuracies and F1 scores after five minutes of training for the eight models trained and tested here, along with the respective standard deviations. The vanilla LSTM architecture, using the one-stage classification approach, appeared to achieve the highest mean F1 score of all the models tested on this dataset.

*Table 1: Mean accuracy and F1 score for each model built on the original UCI Smartphones dataset, with the standard deviation for each in brackets. Models which used the DAC approach are reported in italics. The table is sorted in descending order based on the F1 score of each model.*

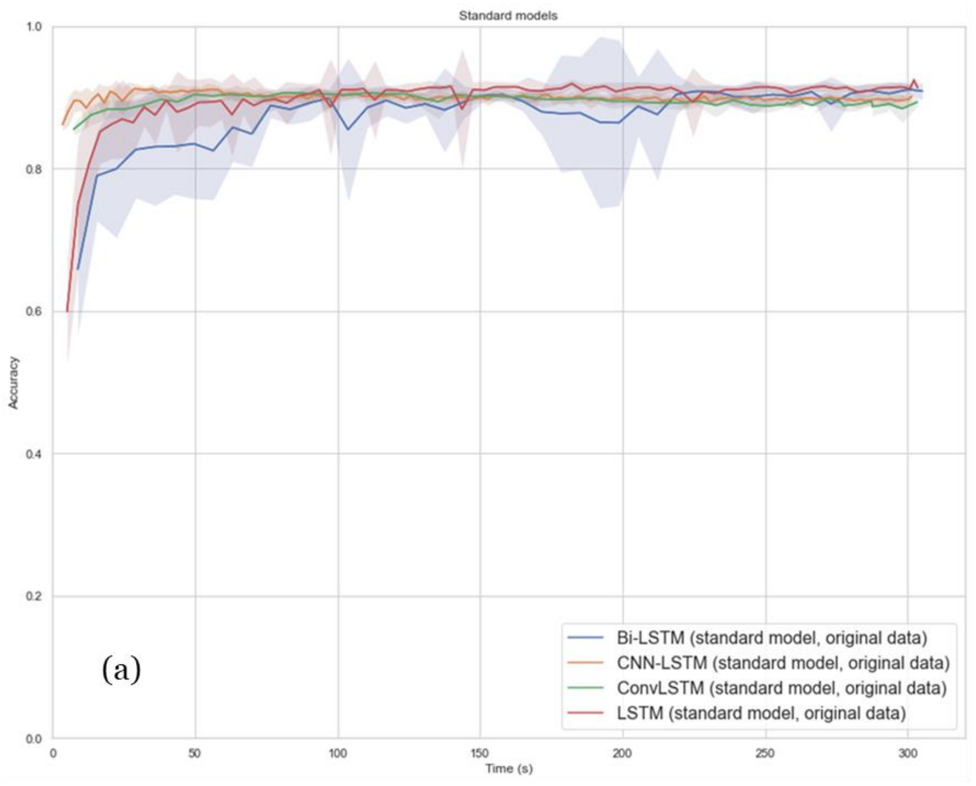
| Model | Mean test accuracy (standard deviation) | Mean test F1 score (standard deviation) |
|---|---|---|
| Vanilla LSTM (standard) | 91.46% (0.75%) | 0.9143 (0.0075) |
| Bi-directional LSTM (standard) | 91.03% (0.93%) | 0.9097 (0.0094) |
| *Vanilla LSTM (DAC)* | *90.48% (2.31%)* | *0.9034 (0.0265)* |
| *CNN-LSTM (DAC)* | *89.80% (0.65%)* | *0.8986 (0.0060)* |
| *ConvLSTM (DAC)* | *89.72% (0.75%)* | *0.8972 (0.0075)* |
| CNN-LSTM (standard) | 89.67% (0.55%) | 0.8965 (0.0054) |
| *Bi-directional LSTM (DAC)* | *89.54% (2.36%)* | *0.8953 (0.0240)* |
| ConvLSTM (standard) | 89.27% (0.58%) | 0.8922 (0.0057) |

However, the difference in performance – measured by F1 score – between the vanilla LSTM and some other models after five minutes was not significant (See Table 2). Thus, it can be concluded that the best performing models here were the vanilla LSTM and bi-directional LSTM using the standard approach, and the vanilla LSTM using the DAC approach.

*Table 2: Results of the t-tests performed to determine if differences between the "best" performing model (the vanilla LSTM using the one-stage classifier) and all other models built on this version of the dataset were significant. Significant differences are shown in bold.*

| Model | Welch's t-test result (Vanilla LSTM (standard) vs Model) | Significance (Vanilla LSTM (standard) vs Model) |
|---|---|---|
| **CNN-LSTM (standard)** | ***t* = 5.73** | ***p* < 0.001** |
| **ConvLSTM (standard)** | ***t* = 6.98** | ***p* < 0.001** |
| Bi-directional LSTM (standard) | *t* = 1.13 | *p* = 0.272 |
| Vanilla LSTM (DAC) | *t* = 1.19 | *p* = 0.261 |
| **CNN-LSTM (DAC)** | ***t* = 4.89** | ***p* < 0.001** |
| **ConvLSTM (DAC)** | ***t* = 4.81** | ***p* < 0.001** |
| **Bi-directional LSTM (DAC)** | ***t* = 2.27** | ***p* = 0.045** |

The tables above, however, only consider the final accuracy and F1 score reached by the models, and not the time taken by models to reach this level of performance. Therefore, we also visualised the mean classification accuracy of each model over time (see Figure 18). From this visualisation, it can be seen that the models employing the CNN-LSTM and ConvLSTM architectures trained extremely quickly, reaching high levels of accuracy after little time. The models employing a bi-directional LSTM took longest to train, taking around 80 seconds to reach similar accuracy levels to the other models when using the standard classification approach; and around 160 seconds when using the DAC classification approach. The visualisation also shows that models using the one-stage classification approach generally reached high accuracies faster than those using the DAC approach, which was to be expected given the extra steps involved.



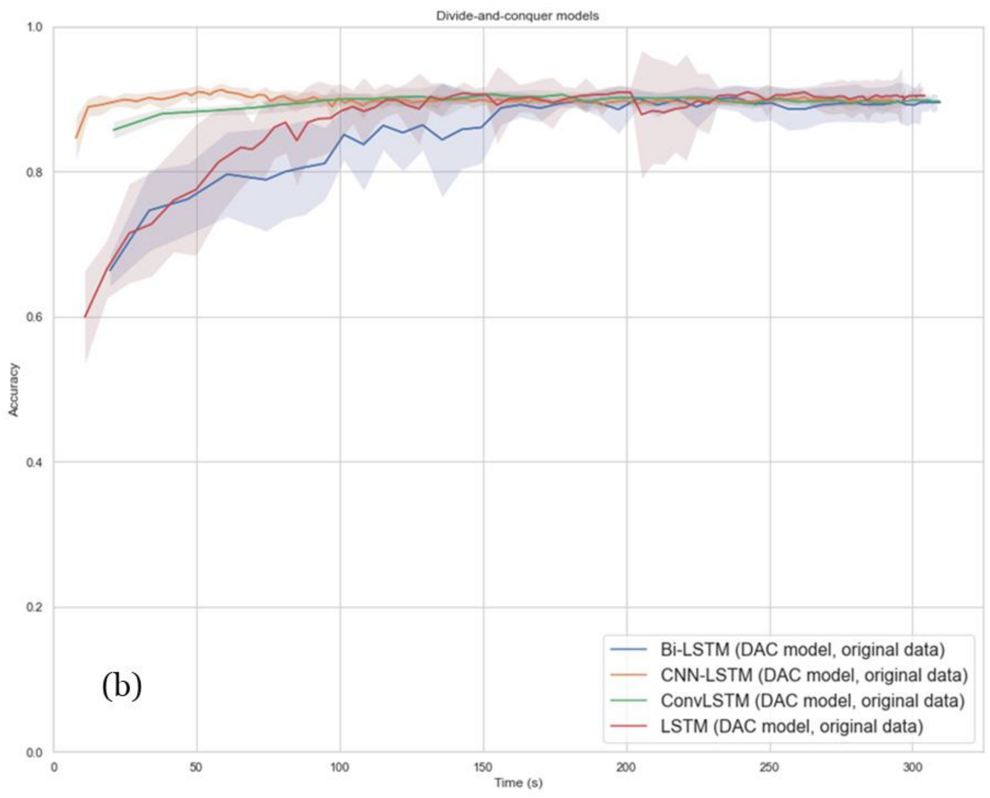


*Figure 18: Mean test accuracy vs time plot for models trained on the original UCI Smartphones dataset. (a) shows the mean test accuracies of the models using a one-stage approach, and (b) shows the mean test accuracies of the models using a DAC approach. The coloured bars around the lines represent one standard deviation in the test accuracy for each model at each point in time.*

It's also worth noting that Figure 18 shows that, for both approaches, the accuracy of all models was very similar after a certain point, long before the five-minute time limit was up. Though there were significant differences between the final accuracies of each model (see Table 1 and Table 2), researchers should weigh up whether these differences in classification performance are more important than the time taken for each model to reach such accuracies - where there was a more recognisable difference between the models. Also, the performance of the bi-directional LSTM and vanilla LSTM models was much more variable than that of the other two models. It could be said that the CNN-LSTM and conv-LSTM provided more reliable performance here.

## *5.2 DATASET AFTER FEATURE SELECTION*

This subsection reports the results of experiments on the dataset after feature selection had been applied on it, resulting in the dataset being reduced from nine features to three. Here, the DAC models all performed better than their one-stage counterparts in terms of their F1 score achieved after five minutes of training, with the CNN-LSTM appearing to score the highest (see Table 3).

*Table 3: Mean accuracy and F1 score for each model built on the UCI Smartphones dataset after feature selection.*

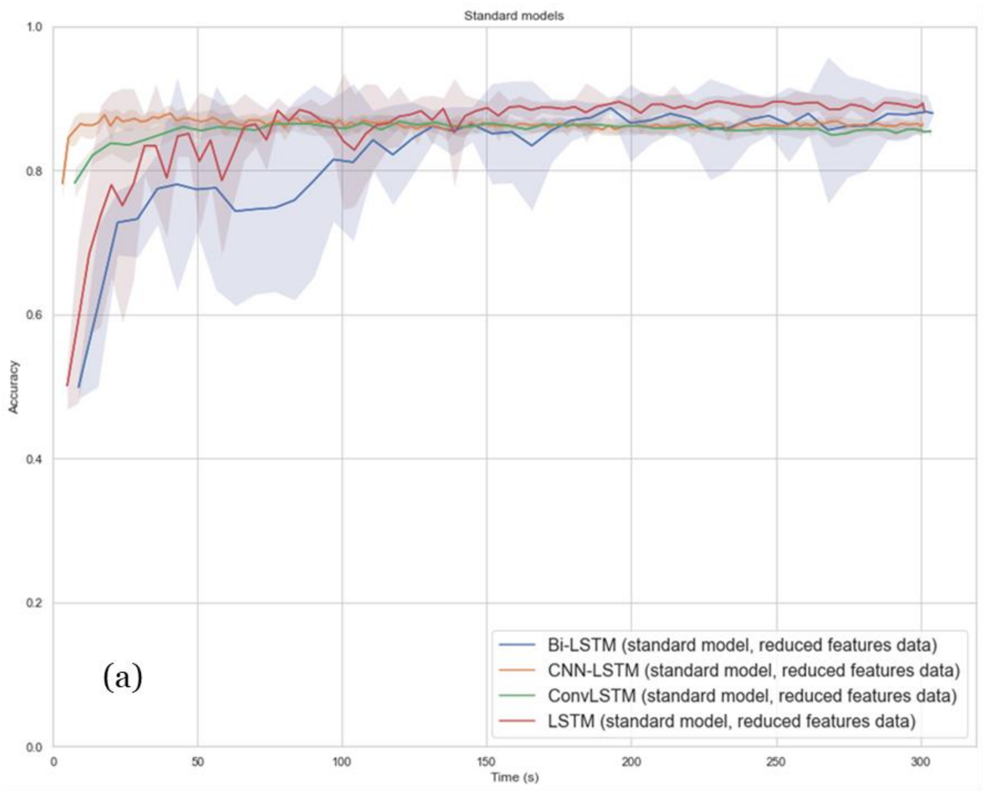
| Model | Mean test accuracy (standard deviation) | Mean test F1 score (standard deviation) |
|---|---|---|
| *CNN-LSTM (DAC)* | *90.14% (0.66%)* | *0.9019 (0.0063)* |
| *Vanilla LSTM (DAC)* | *89.90% (1.58%)* | *0.8987 (0.0161)* |
| *ConvLSTM (DAC)* | *89.79% (0.90%)* | *0.8980 (0.0090)* |
| *Bi-directional LSTM (DAC)* | *89.46% (1.79%)* | *0.8944 (0.0178)* |
| Vanilla LSTM (standard) | 88.20% (3.36%) | 0.8800 (0.0377) |
| Bi-directional LSTM (standard) | 87.96% (2.11%) | 0.8781 (0.0215) |
| CNN-LSTM (standard) | 86.42% (0.81%) | 0.8639 (0.0080) |
| ConvLSTM (standard) | 85.44% (0.73%) | 0.8543 (0.0073) |

However, the difference between the F1 score achieved after five minutes by the CNN-LSTM employing the DAC approach and that achieved by the other architectures using this approach was not significant (see Table 4).

*Table 4: Results of the t-tests performed to determine if differences between the "best" performing model (the CNN-LSTM model using the DAC approach) and all other models built on this version of the dataset were significant.*

| Model | Welch's t-test result (CNN-LSTM (DAC) vs Model) | Significance (CNN-LSTM (DAC) vs Model) |
|---|---|---|
| Vanilla LSTM (standard) | $t = 1.72$ | $p = 0.118$ |
| **CNN-LSTM (standard)** | $\boldsymbol{t = 11.2}$ | $\boldsymbol{p < 0.001}$ |
| **ConvLSTM (standard)** | $\boldsymbol{t = 4.77}$ | $\boldsymbol{p < 0.001}$ |
| **Bi-directional LSTM (standard)** | $\boldsymbol{t = 3.18}$ | $\boldsymbol{p = 0.009}$ |
| Vanilla LSTM (DAC) | $t =0.544$ | $p = 0.597$ |
| ConvLSTM (DAC) | $t = 1.06$ | $p = 0.303$ |
| Bi-directional LSTM (DAC) | $t = 1.18$ | $p = 0.261$ |

Again, we visualised the mean classification accuracy over time for each model built on the UCI smartphones dataset after feature selection (see Figure 19). The bi-directional LSTM and vanilla LSTM took longer to train than the other two models; this was especially true for models using the DAC approach. Again, the aforementioned models also varied a lot more than the other two models in terms of classification accuracy. One obvious difference between the training of the models built here, compared to those built on the original dataset, is that they generally took longer to train. The models built on each version of the dataset will be compared in the next subsection, but it can be seen in Figure 19b that some models appear to have only just been trained to a good accuracy level near the end of the five-minute time limit.



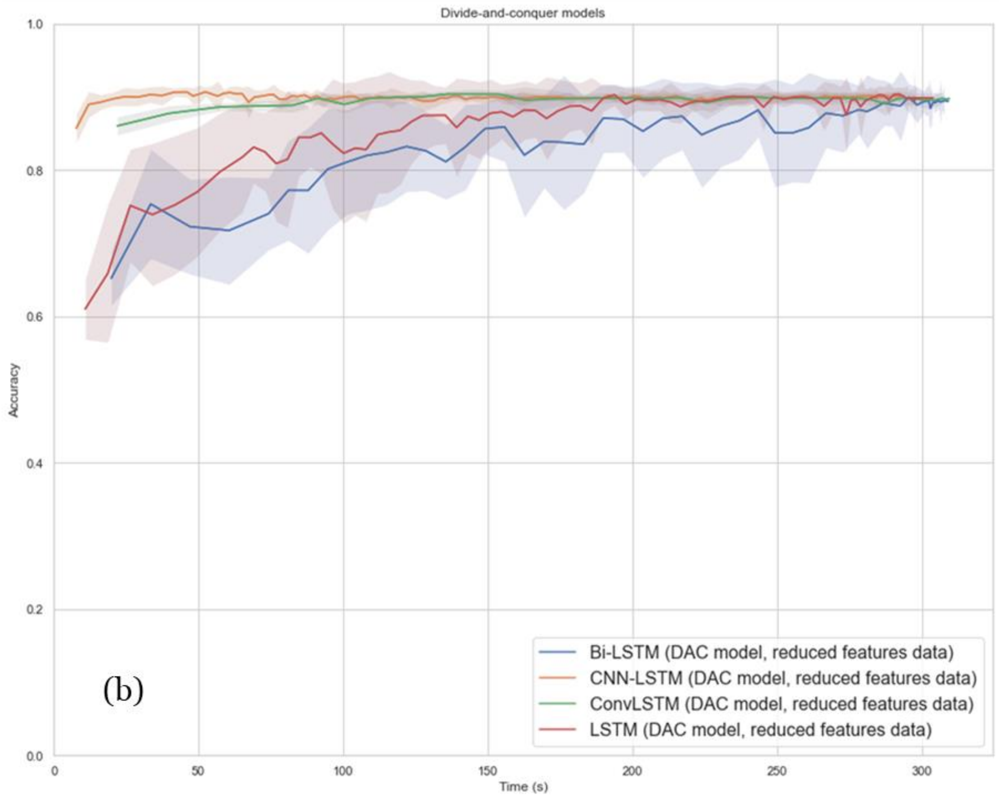


*Figure 19: Mean test accuracy vs time plot for models trained on the UCI Smartphones dataset after feature selection had been applied. Again, (a) shows the mean test accuracies of the models using a one-stage approach, and (b) shows the mean test accuracies of the models using a DAC approach.*

## *5.3 COMPARING ACROSS VERSIONS OF THE DATASET*

As we were also investigating whether feature selection had any effect on our HAR models, we decided to compare each model when built on the original dataset against the same models when built on the dataset after feature selection, side-by-side. In Table 5, we compared the F1 score achieved by each model, and tested whether there were any significant differences here. We found that, somewhat surprisingly, all models built using the standard classification approach performed significantly worse on the dataset after feature selection, whereas the DAC models all performed similarly well on both versions of the dataset.

*Table 5: The final F1 scores for each model, when built on the original dataset compared to when built on the dataset after feature selection. Also reported are the significance levels of the t-tests performed to determine if the performance of the models built on each version of the dataset were significantly different to each other.*

| | Mean test F1 score (standard deviation) | | |
|---|---|---|---|
| Model | Original dataset | After feature selection | t-test significance |
| **Vanilla LSTM (standard)** | **0.9143 (0.0075)** | **0.8800 (0.0377)** | ***p* = *0.024*** |
| **CNN-LSTM (standard)** | **0.8965 (0.0054)** | **0.8639 (0.0080)** | ***p* < *0.001*** |
| **ConvLSTM (standard)** | **0.8922 (0.0057)** | **0.8543 (0.0073)** | ***p* < *0.001*** |
| **Bi-directional LSTM (standard)** | **0.9097 (0.0094)** | **0.8781 (0.0215)** | ***p* = *0.002*** |
| Vanilla LSTM (DAC) | 0.9034 (0.0265) | 0.8987 (0.0161) | *p = 0.660* |
| CNN-LSTM (DAC) | 0.8986 (0.0060) | 0.9019 (0.0063) | *p = 0.271* |
| ConvLSTM (DAC) | 0.8972 (0.0075) | 0.8980 (0.0090) | *p = 0.847* |
| Bi-directional LSTM (DAC) | 0.8953 (0.0240) | 0.8944 (0.0178) | *p = 0.930* |

We also visualised the accuracy achieved by each model over time when built on the original dataset compared to when they were built on the dataset after feature selection (see Figure 20). Figure 20a shows that, as well as achieving lower performance, the models built on the reduced dataset using the standard classification approach also took longer to train. Figure 20b shows that the training process for the CNN-LSTM and Conv-LSTM was almost identical across the different versions of the dataset when using the DAC approach. The other two DAC models, however, did train slower on the dataset after feature selection.

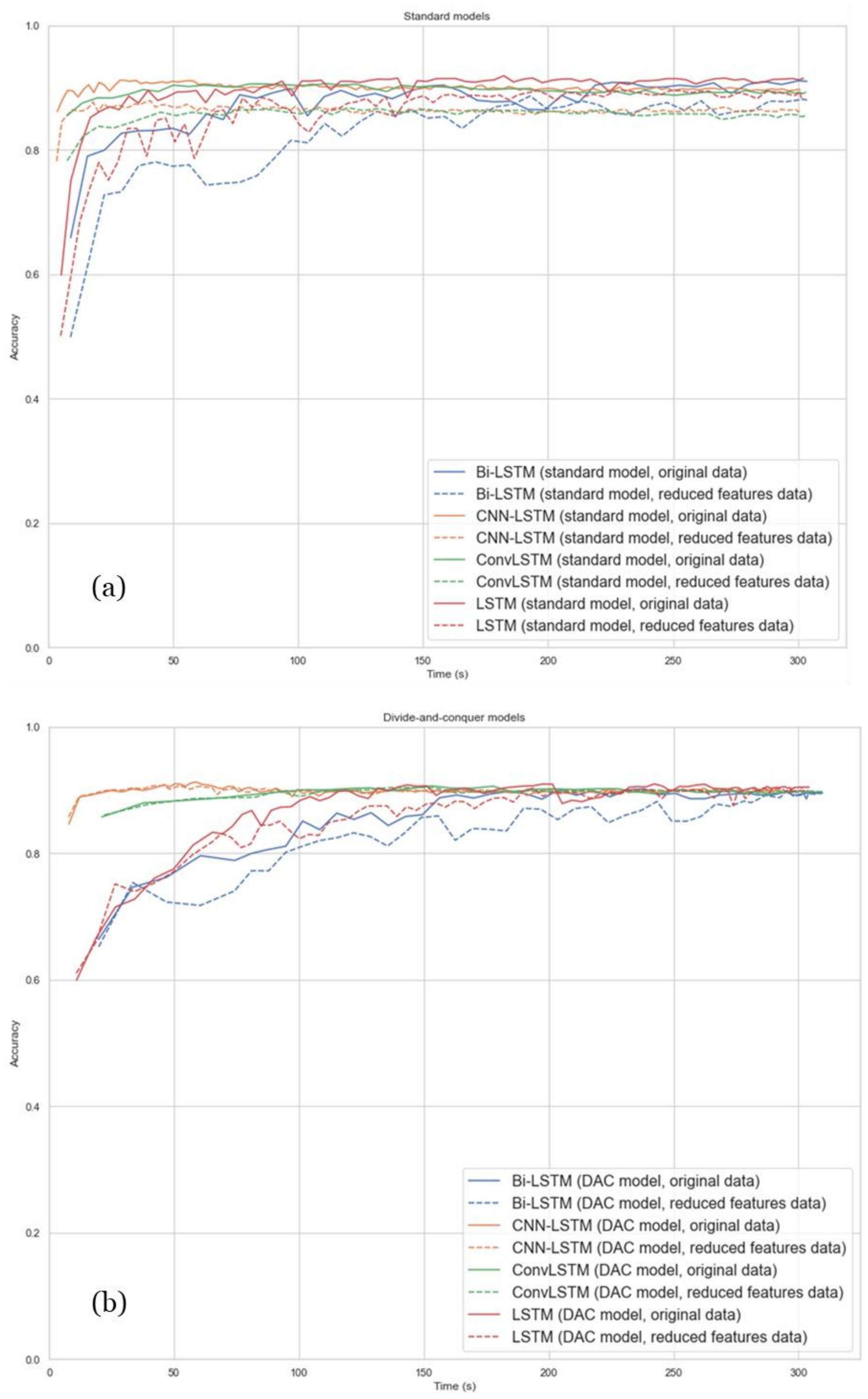


*Figure 20: Mean test accuracy vs time plot for models trained on each version of the UCI Smartphones dataset. Again, (a) shows the mean test accuracies of the models using a one-stage approach, and (b) shows the mean test accuracies of the models using a DAC approach. The models built on the dataset after feature selection are plotted using a dashed line.*

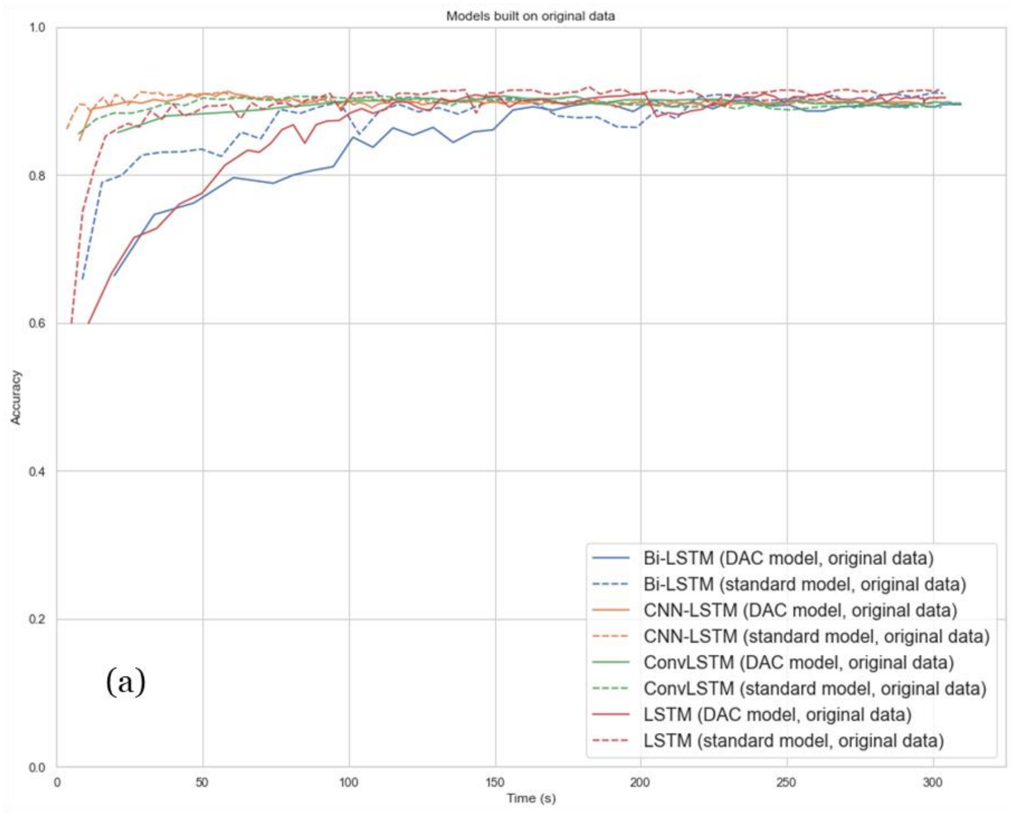
## 5.4 *COMPARING THE STANDARD APPROACH AGAINST THE DAC APPROACH*

As we also wanted to test whether the DAC approach had any effect on the performances of the models, we performed t-tests between the F1 score achieved after five minutes by each architecture when deployed using a one-stage approach and when deployed using a DAC approach (see Table 6). Most models did not perform significantly differently depending on what type of approach was used. However, the CNN-LSTM and ConvLSTM built on the reduced dataset did perform better when using a DAC approach, with an improvement of around 0.04 in F1 score. This is a reasonably large improvement, and will be discussed some more in the discussion section of this report.

*Table 6: Models employing a one-stage classification approach were compared to those using a DAC approach. Here, we also reported the significance level of the t-tests performed to see whether any differences between the performance with either approach was significant.*

| | Mean test F1 score (standard deviation) | | |
|---|---|---|---|
| Model | Standard models | DAC models | t-test significance |
| Vanilla LSTM (original dataset) | 0.9143 (0.0075) | 0.9034 (0.0265) | *p = 0.261* |
| CNN-LSTM (original dataset) | 0.8965 (0.0054) | 0.8986 (0.0060) | *p = 0.463* |
| ConvLSTM (original dataset) | 0.8922 (0.0057) | 0.8972 (0.0075) | *p = 0.134* |
| Bi-directional LSTM (original dataset) | 0.9097 (0.0094) | 0.8953 (0.0240) | *p = 0.120* |
| Vanilla LSTM (after feature selection) | 0.8800 (0.0377) | 0.8987 (0.0161) | *p = 0.195* |
| **CNN-LSTM (after feature selection)** | **0.8639 (0.0080)** | **0.9019 (0.0063)** | ***p < 0.001*** |
| **ConvLSTM (after feature selection)** | **0.8543 (0.0073)** | **0.8980 (0.0090)** | ***p < 0.001*** |
| Bi-directional LSTM (after feature selection) | 0.8781 (0.0215) | 0.8944 (0.0178) | *p = 0.097* |

We also visualised the accuracy over time for DAC models and standard models (see Figure 21). Figure 21a shows that, for models built on the original dataset, the one-stage classification models tended to train faster than the DAC models and reached slightly higher accuracies. However, on the dataset after feature selection, the CNN-LSTM and ConvLSTM models using DAC classification approach tended to reach higher accuracies than their one-stage counterparts early on and continued to do so until the five-minute time limit was reached (see Figure 21b). The other two architectures seemed to perform similarly well whether they used the one-stage approach or the DAC approach on this version of the dataset, both in terms of training time and accuracy.



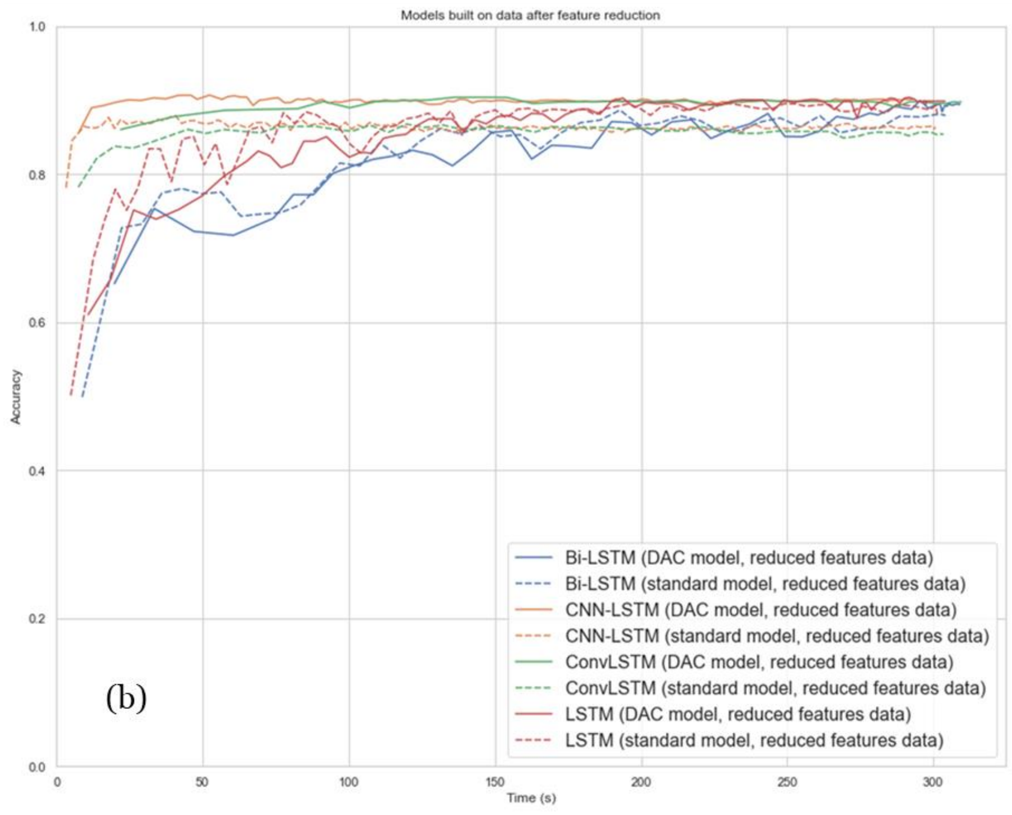


*Figure 21: Mean test accuracy vs time plot for models employing the one-stage approach and those employing the DAC approach. Here, the models built on the original dataset are shown in (a), and the models built on the dataset after feature selection are shown in (b). The models employing the standard classification approach are plotted with a dashed line.*

# 6 RESULTS: WISDM DATASET

This section details the results from the WISDM dataset, following a similar structure to the previous section. The eight models built on the original dataset are compared with each other first (section 6.1), then the eight models built on the dataset after oversampling are compared with each other (section 6.2), followed by the eight models built on the dataset after feature selection (section 6.3). Section 6.4 compares the results achieved on the three versions of the WISDM dataset with each other; and section 6.5 compares the results achieved using the one-stage classification approach against those achieved using the DAC approach. The same metrics and visualisations were used to compare the models built on the WISDM dataset as those used to compare models built on the UCI Smartphones dataset.

## *6.1 ORIGINAL DATASET*

This subsection reports the results of our experiments on the original WISDM dataset, before oversampling or feature selection was applied to it. Table 7 displays the mean final accuracy and F1 score achieved by each model after five minutes of training. It can be seen here that the models built on the original WISDM dataset performed considerably worse than the same models built on the UCI Smartphones dataset. However, this was somewhat expected given the problems present in the WISDM dataset (highlighted in section 2.2).

*Table 7: Mean accuracy and F1 score for each model built on the original WISDM dataset. The table is sorted in descending order based on the F1 score of each model.*

| Model | Mean test accuracy (standard deviation) | Mean test F1 score (standard deviation) |
|---|---|---|
| CNN-LSTM (standard) | 83.29% (0.77%) | 0.8340 (0.0065) |
| ConvLSTM (standard) | 82.24% (0.61%) | 0.8236 (0.0055) |
| Bi-directional LSTM (standard) | 81.99% (0.58%) | 0.8232 (0.0056) |
| *Bi-directional LSTM (DAC)* | *81.65% (1.31%)* | *0.8212 (0.0125)* |
| Vanilla LSTM (standard) | 81.76% (0.59%) | 0.8199 (0.0052) |
| *CNN-LSTM (DAC)* | *81.61% (0.58%)* | *0.8184 (0.0054)* |
| *Vanilla LSTM (DAC)* | *81.40% (1.05%)* | *0.8175 (0.0102)* |
| *ConvLSTM (DAC)* | *81.61% (0.43%)* | *0.8160 (0.0043)* |

Though the CNN-LSTM model built with the one-stage classification approach appeared to be the best performing model here, we performed t-tests between the accuracy scores achieved by this model and those achieved by all other models to see if this model was significantly better (see Table 8). Here, we found that the CNN-LSTM model using the standard classification approach did achieve the highest classification performance after five minutes, with a significantly higher F1 score than all other models tested on this dataset.

*Table 8: Results of the t-tests performed to determine if differences between the "best" performing model (the CNN-LSTM using the standard 1-stage classifier) and all other models built on this version of the dataset were significant.*

| Model | Welch's t-test result (CNN-LSTM (standard) vs Model) | Significance (CNN-LSTM (standard) vs Model) |
|---|---|---|
| **Vanilla LSTM (standard)** | ***t* = 5.07** | ***p* < 0.001** |
| **ConvLSTM (standard)** | ***t* = 3.65** | ***p* = 0.002** |
| **Bi-directional LSTM (standard)** | ***t* = 3.80** | ***p* = 0.001** |
| **Vanilla LSTM (DAC)** | ***t* = 4.10** | ***p* = 0.001** |
| **CNN-LSTM (DAC)** | ***t* = 5.51** | ***p* < 0.001** |
| **ConvLSTM (DAC)** | ***t* = 6.91** | ***p* < 0.001** |
| **Bi-directional LSTM (DAC)** | ***t* = 2.27** | ***p* = 0.017** |

To observe how the test accuracy of each model changed during the training process, we also visualised the mean classification accuracy of each model over time (see Figure 22). Similarly to models trained on the original UCI Smartphones dataset, the models all reached a good level of accuracy before the five-minute time limit was up. The CNN-LSTM trained fastest whether it used the standard or DAC classification approach, reaching high accuracies in little time. As the CNN-LSTM model built with a one-stage classification approach both trained the fastest and achieved the highest F1 score, it can be concluded that this was the best overall model built on this version of the dataset.

Figure 22b shows the accuracy against time plot of the models which used the DAC approach. Here, after around 180 seconds, the plot for all models appears to almost be a flat, horizontal line. This is due to an error on our behalf. The WISDM dataset contained only two stationary classes, both of which contained a relatively small number of samples. This meant that, in the second classification stage, the DAC models had much less stationary data to process than they did dynamic data. The second stage DAC classifiers were both given the same time limit (135 seconds) to run for, meaning that the second stage stationary classifier ran for many more epochs than the second stage dynamic classifier. When we built our models, we calculated the time elapsed as time elapsed per epoch – adding the time taken by each of the DAC classifiers at each epoch together to find this; therefore, the times calculated after a certain point only took the stationary classifier into account. This meant that at later points in time – as we calculated it – the dynamic classifier's accuracy stayed constant. The accuracy of DAC models was calculated as a weighted mean of the accuracy of the stationary classifier and the accuracy of the dynamic classifier and so, at later points in the plot where only the stationary classifier’s accuracy varied, only small changes can be seen. Unfortunately, the time elapsed was calculated within the function which ran model in our code, and we noticed this error too late to change it as re-running the models fully would take too long.

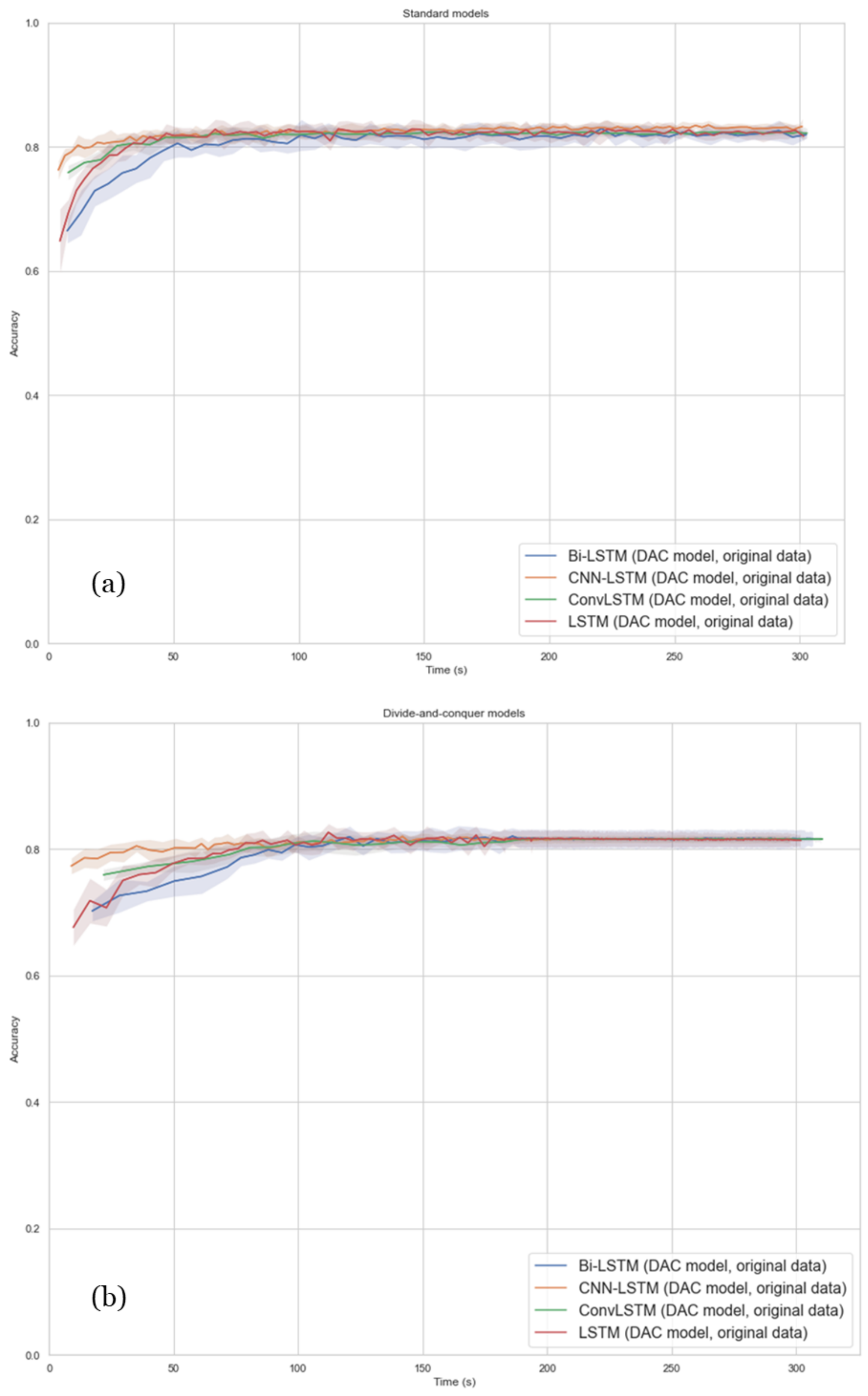


*Figure 22: Mean test accuracy vs time plot for models trained on the original WISDM dataset. (a) shows the mean test accuracies of the models using a standard 1-stage approach, (b) shows the mean test accuracies of the models using a DAC approach.*

## 6.2 *AFTER OVERSAMPLING*

This subsection reports the results of the experiments on the WISDM dataset after oversampling had been applied to it. This resulted in the size of the dataset increasing substantially, which we expected to cause longer training times. Once again, the CNN-LSTM using the standard classification approach appeared to achieve the highest mean F1 score after five minutes (see Table 9).

*Table 9: Mean accuracy and F1 score for each model built on the WISDM dataset after oversampling was applied. The table is sorted by F1 score in descending order.*

| Model | Mean test accuracy (standard deviation) | Mean test F1 score (standard deviation) |
|---|---|---|
| CNN-LSTM (standard) | 82.46% (1.00%) | 0.8289 (0.0079) |
| Vanilla LSTM (standard) | 82.07% (1.01%) | 0.8238 (0.0099) |
| ConvLSTM (standard) | 81.81% (0.76%) | 0.8214 (0.0064) |
| *Vanilla LSTM (DAC)* | *80.93% (1.13%)* | *0.8143 (0.0112)* |
| Bi-directional LSTM (standard) | 80.60% (1.14%) | 0.8135 (0.0095) |
| *Bi-directional LSTM (DAC)* | *80.24% (1.75%)* | *0.8111 (0.0163)* |
| *CNN-LSTM (DAC)* | *80.19% (1.36%)* | *0.8084 (0.0127)* |
| *ConvLSTM (DAC)* | *79.95% (1.48%)* | *0.8041 (0.0124)* |

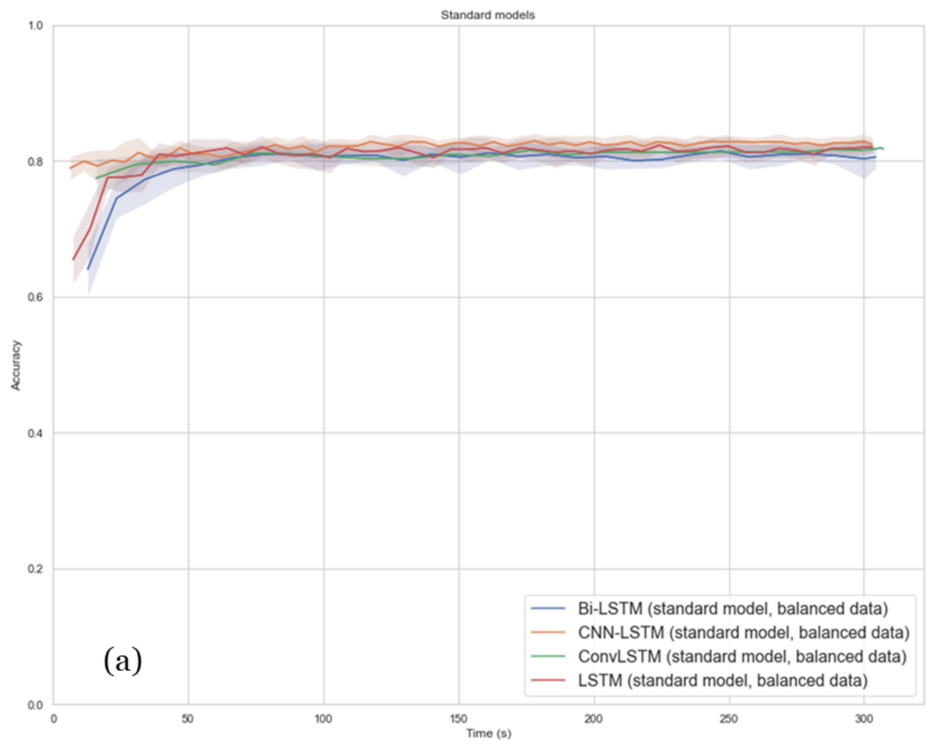
However, on this oversampled dataset, the CNN-LSTM using the one-stage classification approach did not achieve a significantly higher F1 score after five minutes than every other model, with three other models performing similarly here (see Table 10).

*Table 10: Results of the t-test performed to determine if differences between the "best" performing model (the CNN-LSTM model using the one-stage approach) and all other models built on this version of the dataset were significant.*

| Model | Welch's t-test result (CNN-LSTM (standard) vs Model) | Significance (CNN-LSTM (standard) vs Model) |
|---|---|---|
| Vanilla LSTM (standard) | $t = 1.21$ | $p = 0.243$ |
| ConvLSTM (standard) | $t = 2.22$ | $p = 0.398$ |
| **Bi-directional LSTM (standard)** | $\boldsymbol{t = 3.77}$ | $\boldsymbol{p = 0.015}$ |
| Vanilla LSTM (DAC) | $t = 3.21$ | $p = 0.055$ |
| **CNN-LSTM (DAC)** | $\boldsymbol{t = 4.13}$ | $\boldsymbol{p = 0.001}$ |
| **ConvLSTM (DAC)** | $\boldsymbol{t = 5.05}$ | $\boldsymbol{p < 0.001}$ |
| **Bi-directional LSTM (DAC)** | $\boldsymbol{t = 2.95}$ | $\boldsymbol{p = 0.011}$ |

The accuracy of these models over time was visualised again in Figure 23. The CNN-LSTM architectures trained the fastest once again, both in standard and DAC models. The bi-directional LSTM models trained the slowest, while the ConvLSTM and vanilla LSTM trained relatively similarly. Though the dataset was much larger after oversampling had been applied, all models still trained to a good accuracy level well before the five-minute time limit was up.

As the class imbalance problem had been addressed in this version of the dataset, the plotting error where the plot turns into a near-straight horizontal line for the DAC models was less drastic in Figure 23b. However, as the WISDM dataset still contained four dynamic activities compared to two stationary activities, this error was still present.



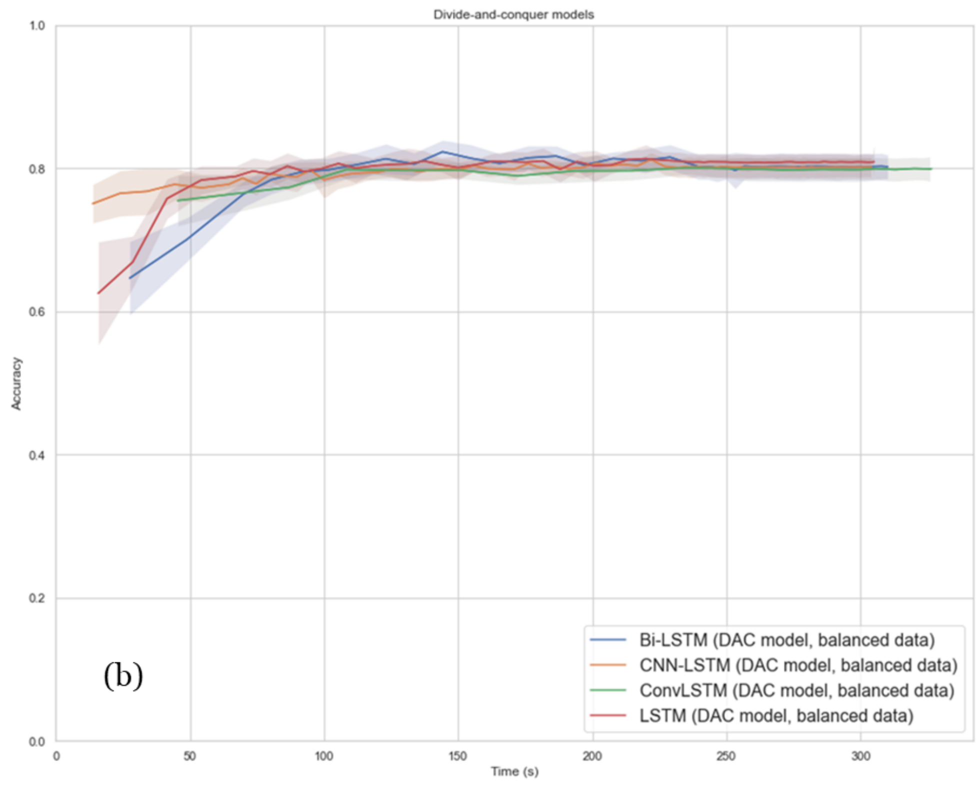


*Figure 23: Mean test accuracy vs time plot for models trained on the WISDM dataset after oversampling had been applied. Again, (a) shows the mean test accuracies of the models using a standard 1-stage approach, and (b) shows the mean test accuracies of the models using a DAC approach.*

## *6.3 AFTER FEATURE SELECTION*

This subsection reports the results of experiments on the dataset after feature selection had been applied on it, resulting in the dataset being reduced from three features to one. It appears that feature selection improved the performance of models on the WISDM dataset (see Table 11), though section 6.5 explores this in more detail to determine if any improvements found were significant. Table 11 displays the mean accuracies and F1 scores of all models built on this version of the dataset, where most models achieved similar performance after five minutes of training. Once again, the CNN-LSTM appeared to achieve the highest mean F1 score, with CNN-LSTM models using either the DAC or the standard classification approach both obtaining an F1 score of 0.8520. The CNN-LSTM model using the standard classification approach did achieve slightly higher mean accuracy though.

*Table 11: Mean accuracy and F1 score for each model built on the WISDM dataset after feature selection. The table is sorted by F1 score in descending order.*

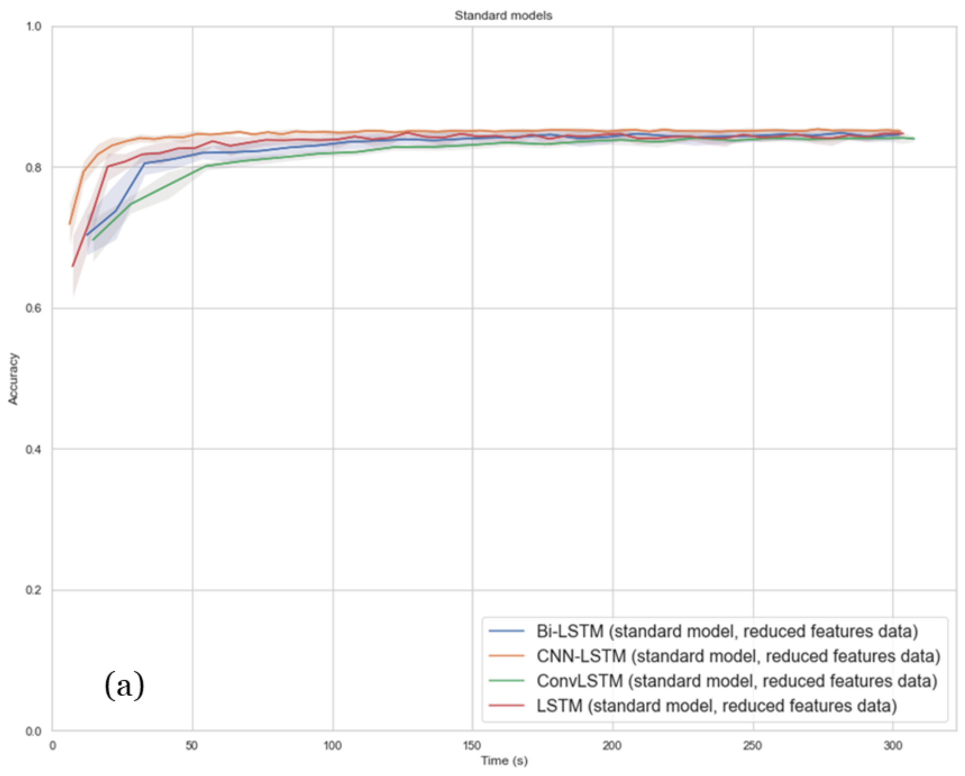
| Model | Mean test accuracy (standard deviation) | Mean test F1 score (standard deviation) |
|---|---|---|
| CNN-LSTM (standard) | 85.07% (0.49%) | 0.8520 (0.0041) |
| *CNN-LSTM (DAC)* | *85.05% (0.43%)* | *0.8520 (0.0037)* |
| Vanilla LSTM (standard) | 84.75% (0.59%) | 0.8512 (0.0049) |
| Bi-directional LSTM (standard) | 84.60% (0.63%) | 0.8506 (0.0059) |
| *Vanilla LSTM (DAC)* | *84.43% (0.42%)* | *0.8494 (0.0043)* |
| *Bi-directional LSTM (DAC)* | *84.10% (0.66%)* | *0.8469 (0.0057)* |
| ConvLSTM (standard) | 84.00% (0.48%) | 0.8456 (0.0041) |
| *ConvLSTM (DAC)* | *81.84% (0.37%)* | *0.8196 (0.0039)* |

Table 12 confirms the similarities between the average performances of each model after five minutes of training. Only the bi-directional LSTM using the DAC approach, and both versions of the ConvLSTM, achieved significantly lower F1 scores than the CNN-LSTM built with the standard classification approach.

*Table 12: Results of the t-test performed to determine if differences between the "best" performing model (the CNN-LSTM model using the one-stage approach) and all other models built on this version of the dataset were significant.*

| Model | Welch's t-test result (CNN-LSTM (standard) vs Model) | Significance (CNN-LSTM (standard) vs Model) |
|---|---|---|
| Vanilla LSTM (standard) | *t* = 0.399 | *p* = 0.695 |
| **ConvLSTM (standard)** | ***t* = 3.32** | ***p* = 0.004** |
| Bi-directional LSTM (standard) | *t* = 0.590 | *p* = 0.563 |
| Vanilla LSTM (DAC) | *t* = 1.30 | *p* = 0.211 |
| CNN-LSTM (DAC) | *t* = 0.00664 | *p* = 0.995 |
| **ConvLSTM (DAC)** | ***t* = 17.2** | ***p* < 0.001** |
| **Bi-directional LSTM (DAC)** | ***t* = 0.0423** | ***p* = 0.042** |

The accuracy over time of these models built on the WISDM dataset after feature selection was visualised for Figure 24. This quite clearly shows how similar the classification accuracy of most models was here after a few minutes of training. The clear worst performing model was the ConvLSTM using the DAC approach, which took longer to train and never reached the same accuracy levels as the other models. Once again, both CNN-LSTM models trained the fastest, reaching top accuracy scores in little time.



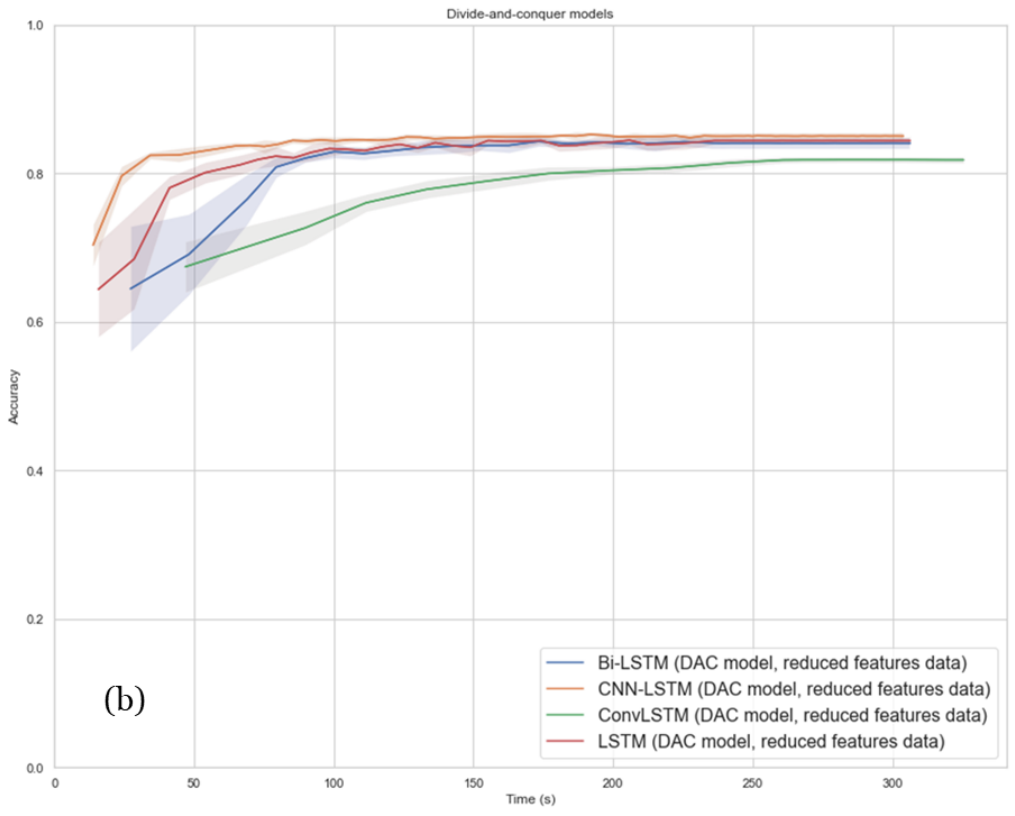


*Figure 24: Mean test accuracy vs time plot for models trained on the WISDM dataset after feature selection had been applied. Again, (a) shows the mean test accuracies of the models using a standard 1-stage approach, and (b) shows the mean test accuracies of the models using a DAC approach.*

## *6.4 COMPARING ACROSS VERSIONS OF THE DATASET*

This section explores whether either oversampling or feature selection led to any improvements in the HAR models built on the WISDM dataset. First, we compared the models built on the original dataset to those built on the dataset after oversampling in their classification performance after five minutes of training (see Table 13). The F1 scores achieved here were similar for most models, however three models performed worse when they were trained on the oversampled dataset. This was a surprising result, given that oversampling was intended to address the class imbalance problem and therefore improve the performance of the models. Possible explanations for this are presented in the discussion section of this report.

*Table 13: The final F1 scores for each model when built on the original dataset compared to when built on the dataset after oversampling was applied. The table also displays the significance level of the pairwise t-tests performed to determine if the performances of the models built on each version of the dataset were significantly different.*

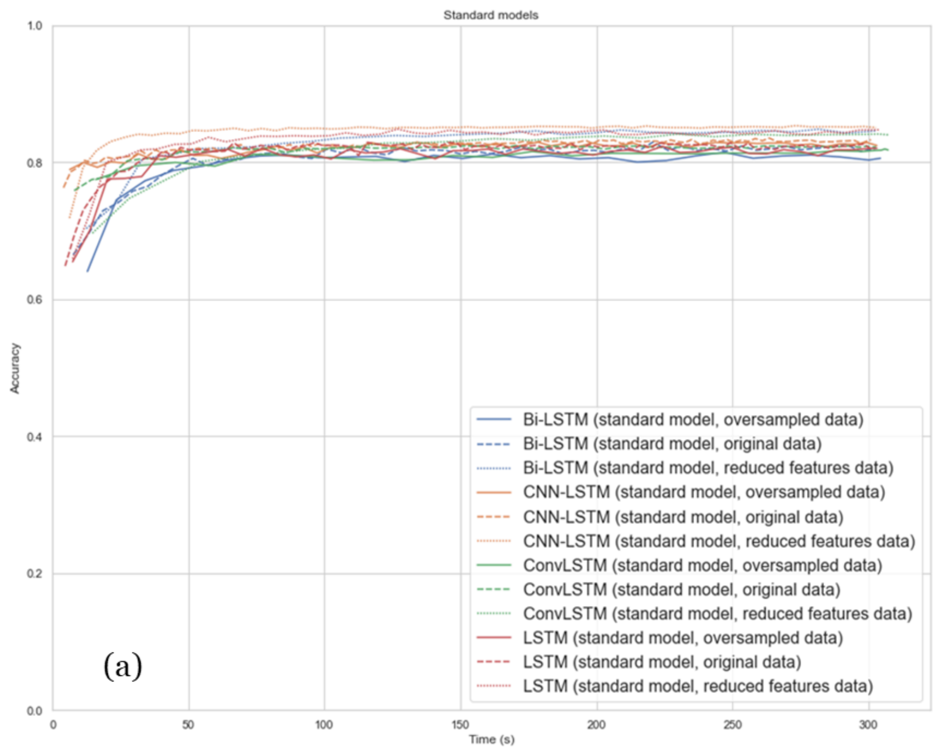
| | Mean test F1 score (standard deviation) | | |
|---|---|---|---|
| Model | Original dataset | After oversampling | t-test significance |
| Vanilla LSTM (standard) | 0.8199 (0.0052) | 0.8238 (0.0099) | *p = 0.309* |
| CNN-LSTM (standard) | 0.8340 (0.0065) | 0.8289 (0.0079) | *p = 0.155* |
| ConvLSTM (standard) | 0.8236 (0.0055) | 0.8214 (0.0064) | *p = 0.450* |
| **Bi-directional LSTM (standard)** | **0.8232 (0.0056)** | **0.8135 (0.0095)** | ***p = 0.018*** |
| Vanilla LSTM (DAC) | 0.8175 (0.0102) | 0.8143 (0.0112) | *p = 0.533* |
| **CNN-LSTM (DAC)** | **0.8184 (0.0054)** | **0.8084 (0.0127)** | ***p = 0.049*** |
| **ConvLSTM (DAC)** | **0.8160 (0.0043)** | **0.8041 (0.0124)** | ***p = 0.020*** |
| Bi-directional LSTM (DAC) | 0.8212 (0.0125) | 0.8111 (0.0163) | *p = 0.160* |

Next, we compared the models built on the original dataset to those built on the dataset after feature selection in their classification performance after five minutes of training (see Table 14). Here, we did find significant and reasonably substantial differences between the performances of most models. Every model, apart from the ConvLSTM model utilizing the DAC approach, achieved a significantly higher F1 score after feature selection had been applied to the dataset.

*Table 14: The final F1 scores for each model when built on the original dataset compared to when built on the dataset after feature selection. The table also displays the significance level of the pairwise t-tests performed to determine if the performances of the models built on each version of the dataset were significantly different.*

| Model | Mean test F1 score (standard deviation) | | t-test significance |
|---|---|---|---|
| | Original dataset | After feature selection | |
| **Vanilla LSTM (standard)** | **0.8199 (0.0052)** | **0.8512 (0.0049)** | ***p* < 0.001** |
| **CNN-LSTM (standard)** | **0.8340 (0.0065)** | **0.8520 (0.0041)** | ***p* < 0.001** |
| **ConvLSTM (standard)** | **0.8236 (0.0055)** | **0.8456 (0.0041)** | ***p* < 0.001** |
| **Bi-directional LSTM (standard)** | **0.8232 (0.0056)** | **0.8506 (0.0059)** | ***p* < 0.001** |
| **Vanilla LSTM (DAC)** | **0.8175 (0.0102)** | **0.8494 (0.0043)** | ***p* < 0.001** |
| **CNN-LSTM (DAC)** | **0.8184 (0.0054)** | **0.8520 (0.0037)** | ***p* < 0.001** |
| ConvLSTM (DAC) | 0.8160 (0.0043) | 0.8196 (0.0039) | *p = 0.083* |
| **Bi-directional LSTM (DAC)** | **0.8212 (0.0125)** | **0.8469 (0.0057)** | ***p* < 0.001** |

We also visualised the accuracy achieved by each model over time when built on the each of the three versions of the dataset, comparing models built on different versions in the same plot (see Figure 25). Figure 25 shows the benefit that most models received from feature selection, quickly achieving higher classification accuracy than the same models built on other versions of the dataset. Oversampling appeared to do little to the performance of the models other than slowing down the training process.



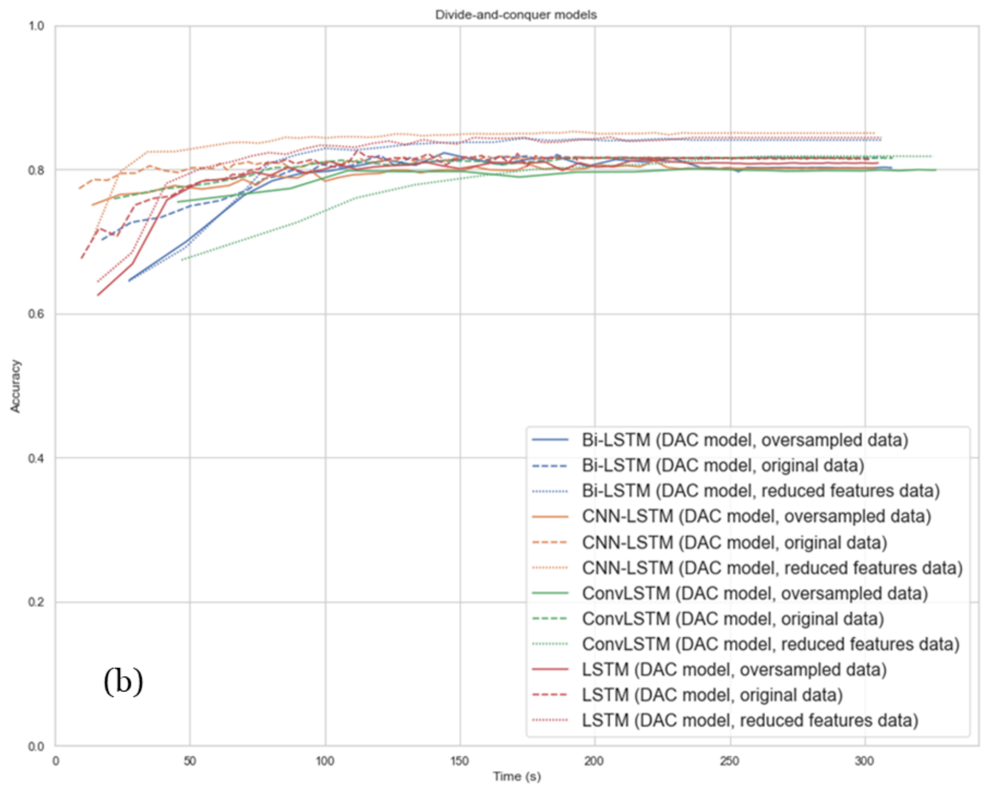


*Figure 25: Mean test accuracy vs time plot for models trained on each version of the UCI Smartphones dataset. Again, (a) shows the mean test accuracies of the models using a one-stage approach, and (b) shows the mean test accuracies of the models using a DAC approach. The models built on the original dataset are shown with a dashed line, the models built on the dataset after oversampling are shown with a solid line, and the models built on the dataset after feature selection are shown with a dotted line.*

## *6.5 COMPARING THE STANDARD APPROACH AGAINST THE DAC APPROACH*

To determine whether the DAC approach provided any benefit to models built on the WISDM dataset, we performed t-tests between the F1 score achieved after five minutes by each architecture when deployed using a one-stage approach and when deployed using a DAC approach (see Table 15). Here, we found that DAC approach provided no improvement over the one-stage classification approach for any of the models; some DAC models performed worse than their one-stage counterparts. This contrasts to the findings from the UCI Smartphones dataset, where DAC models performed better after feature selection.

*Table 15: Models employing a one-stage classification approach compared to those using a DAC approach. Also reported is the significance level of the t-tests performed to see whether any differences between the performance of models using either approach was significant.*

| | Mean test F1 score (standard deviation) | | |
|---|---|---|---|
| Model | Standard models | DAC models | t-test significance |
| Vanilla LSTM (original dataset) | 0.8199 (0.0052) | 0.8175 (0.0102) | *p = 0.543* |
| **CNN-LSTM (original dataset)** | **0.8340 (0.0065)** | **0.8184 (0.0054)** | ***p < 0.001*** |
| **ConvLSTM (original dataset)** | **0.8236 (0.0055)** | **0.8160 (0.0043)** | ***p = 0.005*** |
| Bi-directional LSTM (after oversampling) | 0.8232 (0.0056) | 0.8212 (0.0125) | *p = 0.674* |
| Vanilla LSTM (after oversampling) | 0.8238 (0.0099) | 0.8143 (0.0112) | *p = 0.072* |
| **CNN-LSTM (after oversampling)** | **0.8289 (0.0079)** | **0.8084 (0.0127)** | ***p = 0.001*** |
| **ConvLSTM (after oversampling)** | **0.8214 (0.0064)** | **0.8041 (0.0124)** | ***p = 0.003*** |
| Bi-directional LSTM (after oversampling) | 0.8135 (0.0095) | 0.8111 (0.0163) | *p = 0.717* |
| Vanilla LSTM (after feature selection) | 0.8512 (0.0049) | 0.8494 (0.0043) | *p = 0.438* |
| CNN-LSTM (after feature selection) | 0.8520 (0.0041) | 0.8520 (0.0037) | *p = 0.995* |
| **ConvLSTM (after feature selection)** | **0.8456 (0.0041)** | **0.8196 (0.0039)** | ***p < 0.001*** |
| Bi-directional LSTM (after feature selection) | 0.8506 (0.0059) | 0.8469 (0.0057) | *p = 0.188* |

We also visualised the training process of the standard models compared to the DAC models for each version of the dataset (see Figure 26). Here we can see that all one-stage models trained faster than their DAC counterparts, and usually reached slightly higher levels of accuracy as well - although differences in accuracy were often small.

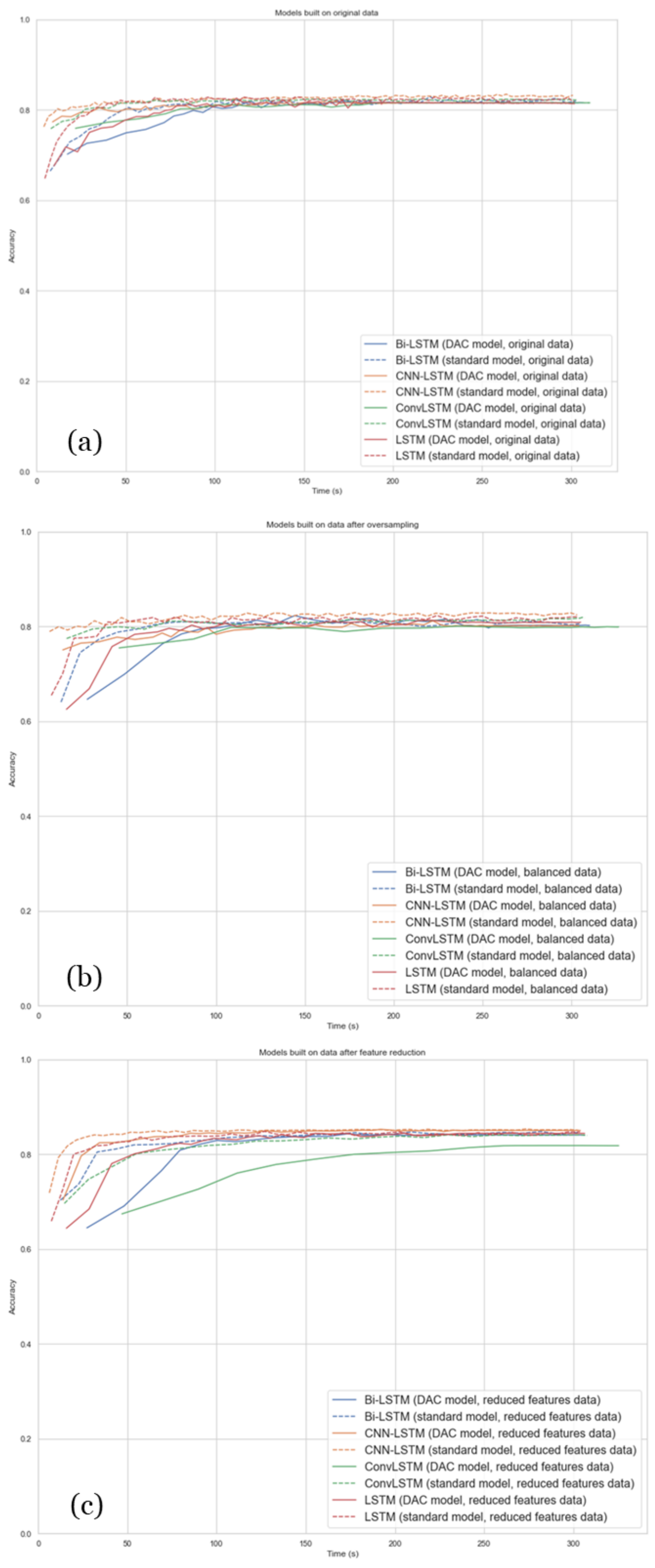


*Figure 26: Mean test accuracy vs time plot for models employing the one-stage approach and those employing the DAC approach. Here, the models built on the original dataset are shown in (a), models built on the dataset after oversampling are shown in (b), and the models built on the dataset after feature selection are shown in (c). The models employing the standard classification approach are plotted with a dashed line.*

# 7 DISCUSSION

This section discusses the results found in the previous two sections. Section 7.1 discusses the results found on the UCI Smartphones dataset, and section 7.2 discusses the results from the WISDM dataset. The overall findings from our various experiments are presented in section 7.3, discussing the best overall type of LSTM architecture to use in HAR applications, as well as whether the DAC approach or any of the data manipulation techniques we used are worthwhile for further investigation. Section 7.4 discusses the limitations we identified in our experiments reported in this paper, before we conclude in section 7.5.

## *7.1 UCI SMARTPHONES DATASET*

Overall, 16 models were built on the UCI Smartphones dataset. This comprised of four different LSTM architectures, built using both a one-stage classification approach and a DAC approach and on two versions of the dataset. The final test F1 scores achieved by each of these models after five minutes of training is summarised in Figure 27. The box plot shown in Figure 27a shows how similar most of these models performed after five minutes of training, while Figure 27b shows the box plot with a reduced x-axis range to see more clearly the small differences between the performances of models here.

Though most models performed relatively similarly after five minutes of training, the most obvious differences could be found with the one-stage models bult on the dataset after feature selection, where the models performed the worst. Generally, the DAC models performed equally well whether they were trained on the original dataset or the dataset after feature selection, whereas one-stage models performed worse after feature selection.

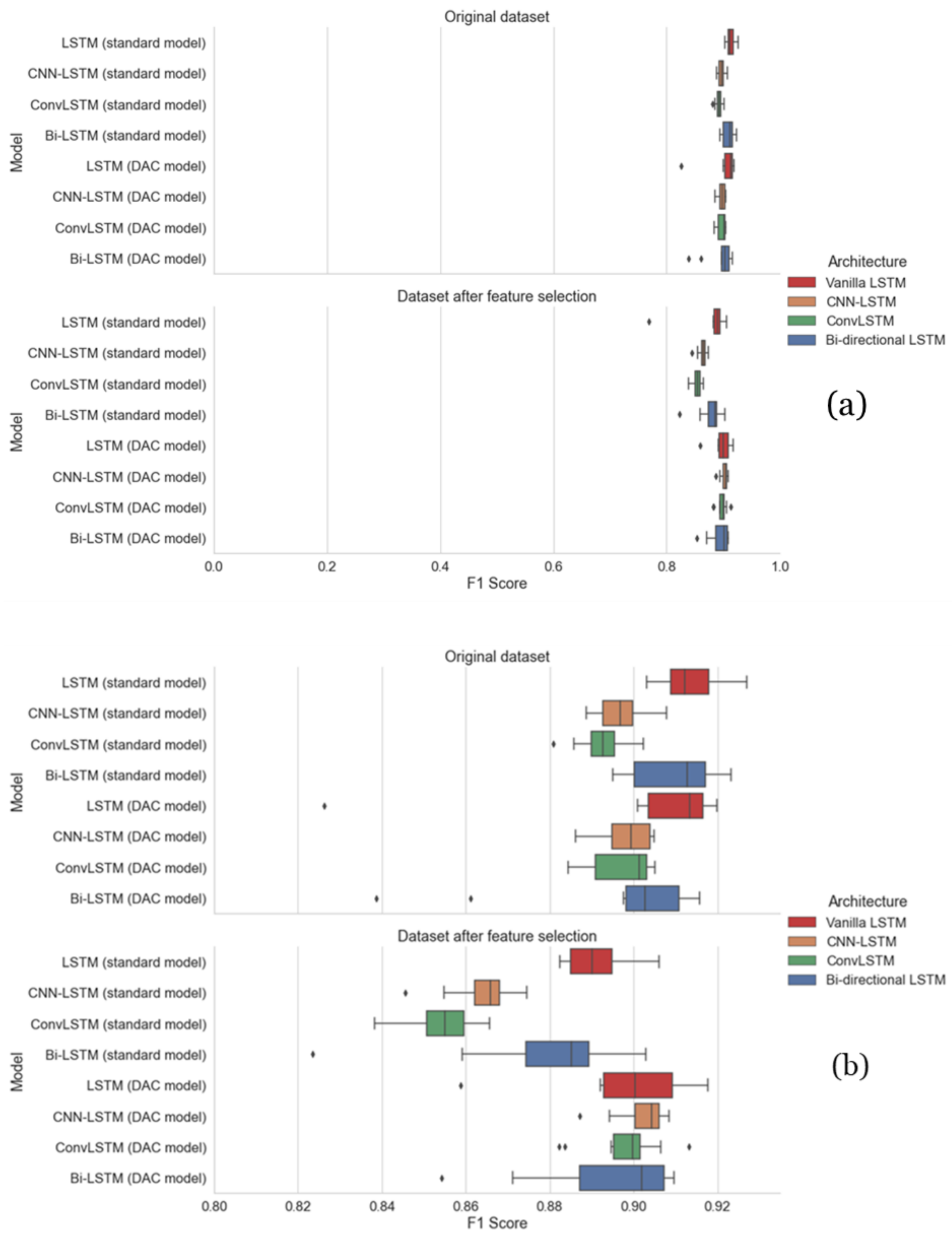


*Figure 27: Box plot of the final F1 scores achieved by all models built on the two versions of the UCI smartphones dataset. (b) displays the same data as (a), but with the x-axis starting at 0.8 to show the differences between the models more clearly.*

The mean final F1 scores achieved by the three best models built on this dataset were not found to be significantly different from each other (see section 5 for a full summary). One of the objectives of our experiments was to find models that both achieved high levels of classification performance and trained quickly. Therefore, it was important for us to look at how long each model took to train to a good level of performance in addition to looking at the final classification performance of the models.

The accuracy over time for the three best performing models found in section 5 is visualised in Figure 28. In the plot, we also added the CNN-LSTM using the DAC approach built on the dataset after feature selection. This is because the CNN-LSTM was generally the fastest architecture to train across experiments on both datasets, and this particular CNN-LSTM model achieved an F1 score only slightly lower than the three best performing models after five minutes. Comparing this CNN-LSTM model with the three best architectures allowed us to see how much benefit each of the best performing models provided over the fast-training CNN-LSTM in terms of the accuracy to training time trade-off. Looking at Figure 28, the LSTM model built on the original dataset with the one-stage classifier trained the fastest of the best models, and not much slower than the CNN-LSTM model plotted. Therefore, it can be concluded that both of these models provided similar levels of performance as both models trained quickly and reached similar levels of classification performance after five minutes.

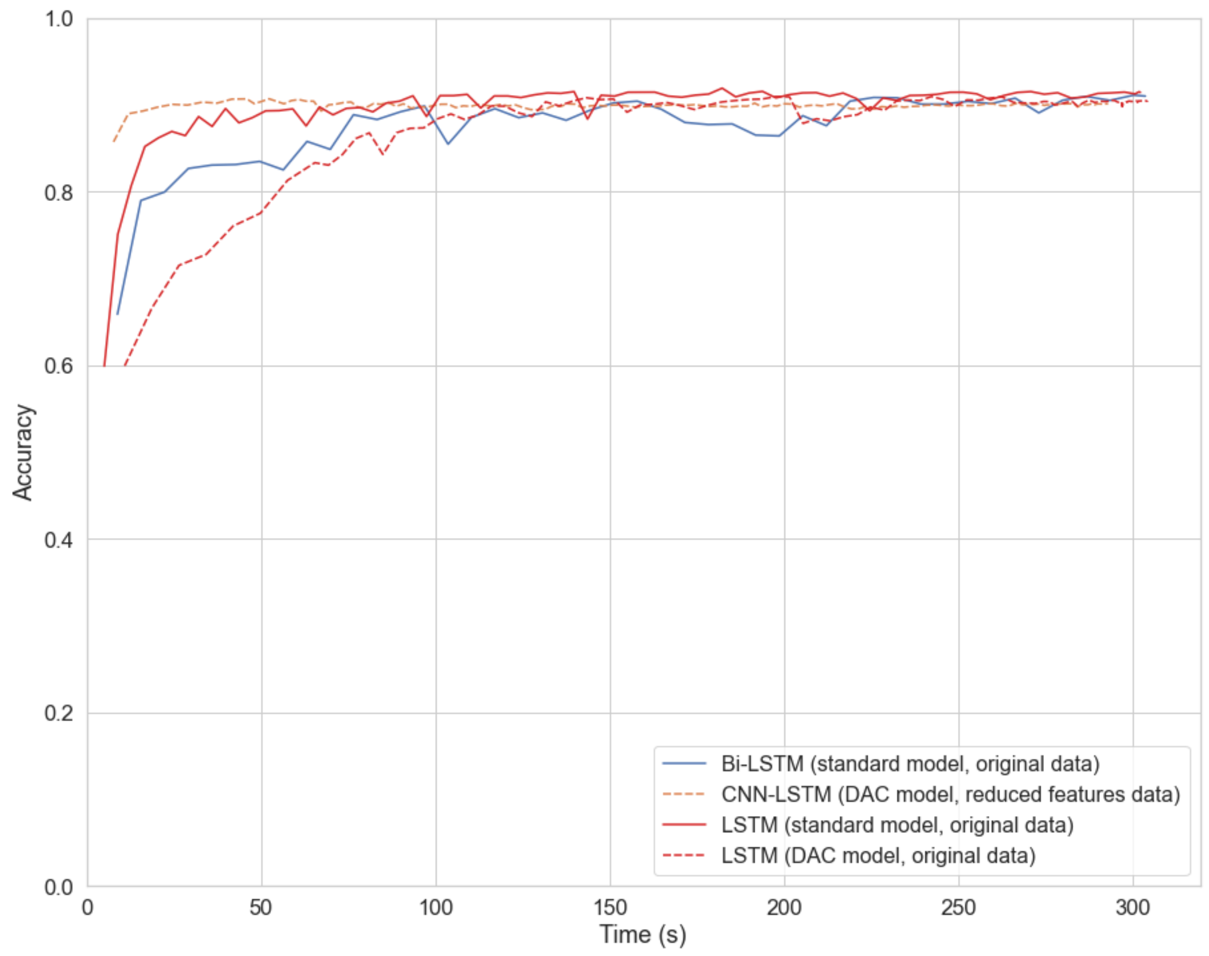


*Figure 28: Plot of test accuracy against time for some of the best performing models on the UCI Smartphones dataset. DAC models are plotted with a dashed line here.*

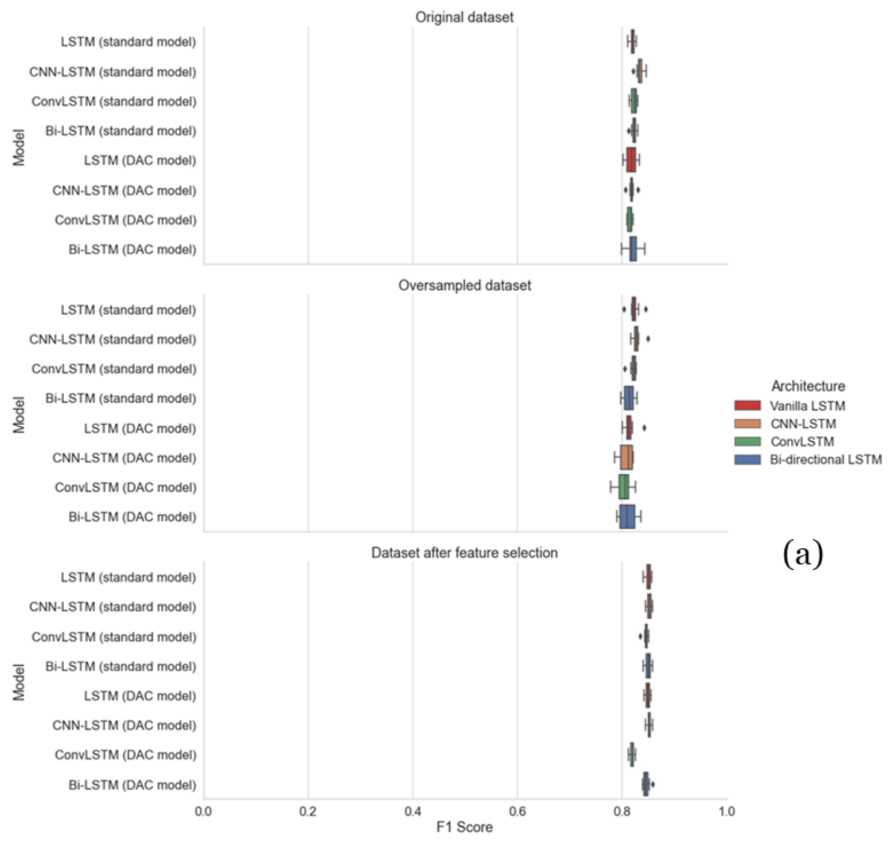
Thus, for the UCI smartphones dataset, the most promising architectures were found to be the vanilla LSTM and the CNN-LSTM. These architectures, after somewhat different pre-processing steps had been taken, provided the best training time to accuracy trade-off, and our findings suggest that future research into building HAR models on this dataset should focus their attention here. As we experimented on models using rather simple versions of these architectures, it is likely that models using more complex versions of these architectures could lead to better classification performance. However, researchers should be careful when building these more complex models not to sacrifice training speed too heavily in favour of reaching the best possible classification performance. As noted earlier in this report, successful HAR models used in real-life applications require a massive amount of data to be trained effectively, and are often deployed on devices with much less processing power than the personal computer used for our experiments (Jiang & Yin, 2015).

The question of whether either feature selection or the use of a DAC approach could improve performance in this dataset remains inconclusive. Neither method alone improved models; however, after feature selection had been applied to the dataset, the models employing the DAC approach did perform better than models that employed the one-stage classification approach. A possible explanation for this is that, after feature selection was applied, the models had less information to learn from and so were aided by splitting the problem into two more manageable chunks. It does appear, however, that more than just the three features selected by the random forest classifier were important for the training process of the HAR model.

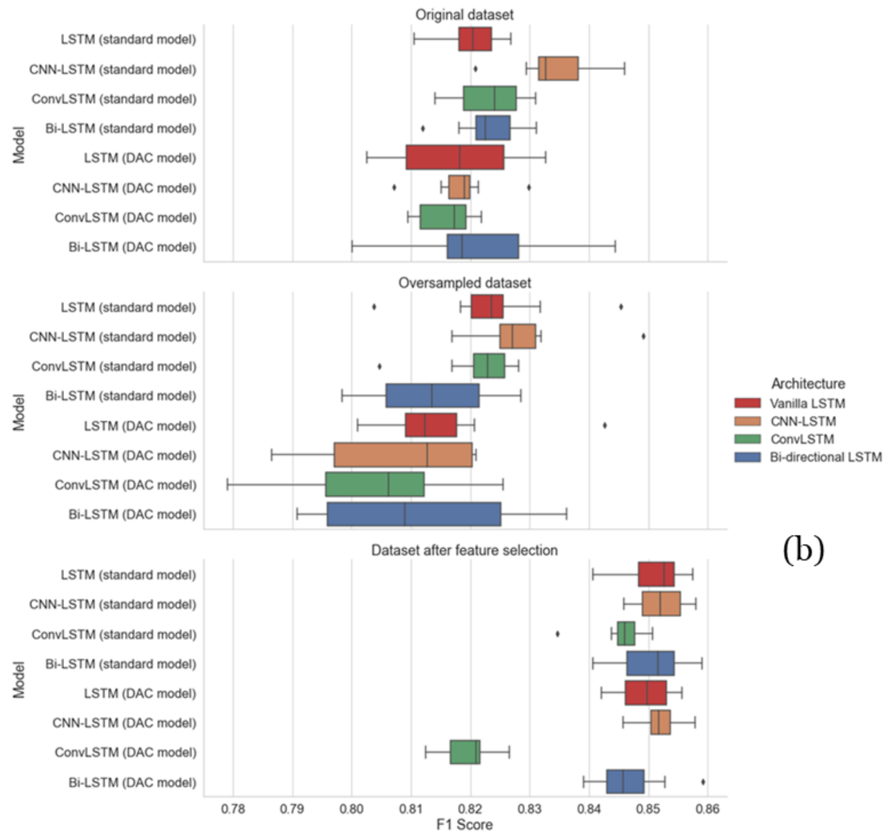
## *7.2 WISDM DATASET*

This section discusses the results found in our experiments on the WISDM dataset. Overall, 24 models were built on this dataset, comprising of four different architectures built using both a one-stage classification approach and a DAC approach on each of the three versions of the dataset. Using the WISDM dataset in addition to the UCI Smartphones dataset allowed us to determine whether the performance of each of our models was consistent across different datasets. The WISDM dataset also provided a slightly different challenge to the UCI Smartphones dataset in that it had a class imbalance problem which we predicted would cause problems for our HAR models.

The final test F1 scores achieved by each of these models is summarised in Figure 29. Figure 29a shows how similar these models generally performed once again, though even here it can be seen that the models usually performed best after feature selection. Figure 29b shows the differences between the models more clearly by decreasing the x-axis range. Here, we can see visually what was confirmed analytically in section 6 — that the CNN-LSTM model using the one-stage classifier was generally either the best model or one of the best models, in terms of F1 score achieved after five minutes, for each version of the dataset.



(a)



(b)

*Figure 29: Box plot of the final F1 scores achieved by all models built on the three versions of the WISDM dataset. (b) displays the same data as (a), but with a smaller axis range to show the differences between the models more clearly.*

As we aimed to address the class imbalance problem found in the WISDM dataset, we also produced normalised confusion matrices for the best performing model built on each version of the dataset. Figure 30 shows this confusion matrix for the CNN-LSTM model built with a one-stage classification approach on the original dataset. Here, we can see that *walking upstairs* and *walking downstairs* were the two classes which this model struggled the most to correctly classify. These two classes had much lower numbers of segments in the training data than the two most common classes, *jogging* and *walking*, and so this was to be expected. The two least common classes in the training dataset, *sitting* and *standing*, were actually detected rather well by the model; this is likely due to them both being stationary movement categories, associated with more drastically different sensor signal data than the four dynamic categories.

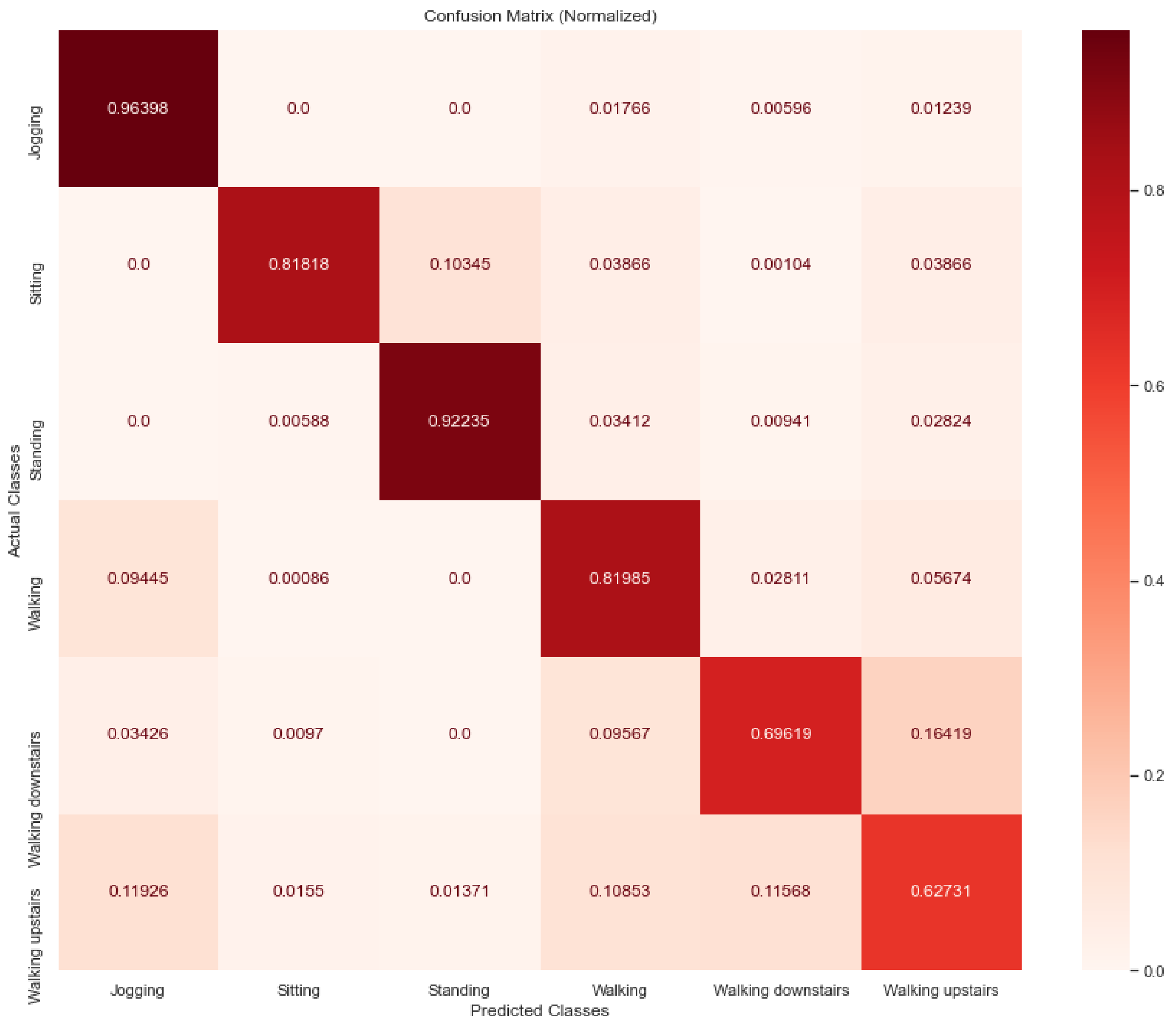
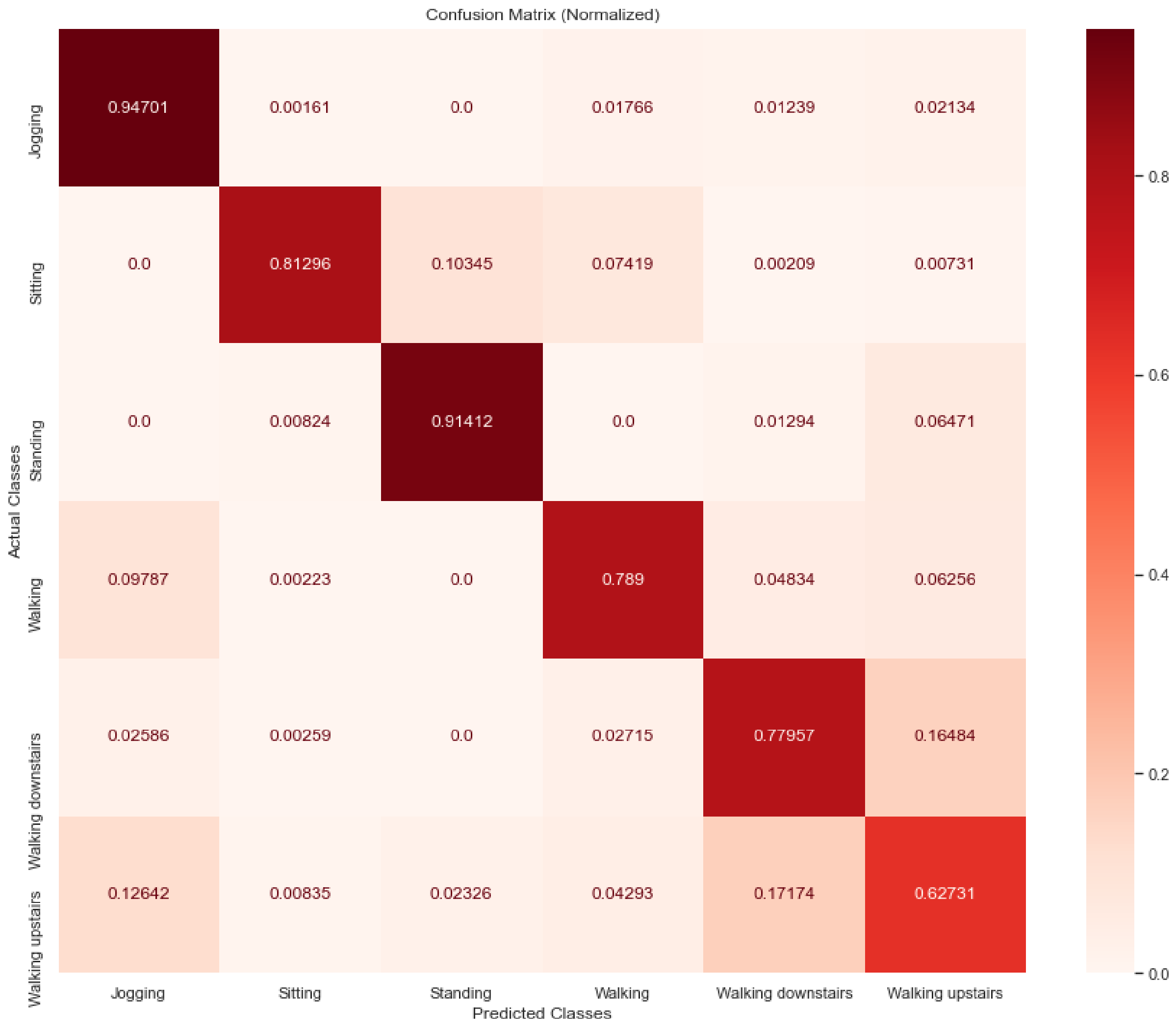


*Figure 30: Normalised confusion matrix of the classes predicted by the model built on the original WISDM dataset compared to the actual classes found in the test dataset.*

Figure 31 shows the confusion matrix produced to show how the best performing model on the oversampled version of the dataset — also the CNN-LSTM model with the one-stage classifier — performed for each activity category. This visualisation shows just how ineffective our oversampling method was on this dataset. The accuracy level for each class was slightly worse in almost all categories, though the accuracy for *walking downstairs* was improved somewhat. This improvement in classification accuracy for one of the less common activity categories is a positive sign but, overall, it must be said that our oversampling method was ineffective for this dataset. It is likely that, by creating many exact copies of segments in the training dataset, we caused models built on this version of the training dataset to be overfitted on it, making it less capable of correctly classifying new data. A more complex oversampling method, which creates similar but not exact copies of training segments, may fare better; future research could investigate using such a method.



| | Jogging | Sitting | Standing | Walking | Walking downstairs | Walking upstairs |
|---|---|---|---|---|---|---|
| Jogging | 0.94701 | 0.00161 | 0.0 | 0.01766 | 0.01239 | 0.02134 |
| Sitting | 0.0 | 0.81296 | 0.10345 | 0.07419 | 0.00209 | 0.00731 |
| Standing | 0.0 | 0.00824 | 0.91412 | 0.0 | 0.01294 | 0.06471 |
| Walking | 0.09787 | 0.00223 | 0.0 | 0.789 | 0.04834 | 0.06256 |
| Walking downstairs | 0.02586 | 0.00259 | 0.0 | 0.02715 | 0.77957 | 0.16484 |
| Walking upstairs | 0.12642 | 0.00835 | 0.02326 | 0.04293 | 0.17174 | 0.62731 |

*Figure 31: Normalised confusion matrix of the classes predicted by the model built on the oversampled WISDM dataset compared to the actual classes found in the test dataset.*

The normalised confusion matrix for the best model built on the WISDM dataset after feature selection — once again the CNN-LSTM with a one-stage classification approach — is displayed in Figure 32. Here, the accuracy for most classes increased compared to the models built on the original dataset, including for some of the less common classes. It is interesting to note that the accuracy for *sitting* was less than it was in the original dataset, with a substantial number of *sitting* segments being predicted as *laying*. The dataset only used one feature — *total acceleration (Y)* — after feature selection and it is likely that one or both of the other features in the original dataset were important for differentiating between these two stationary categories.

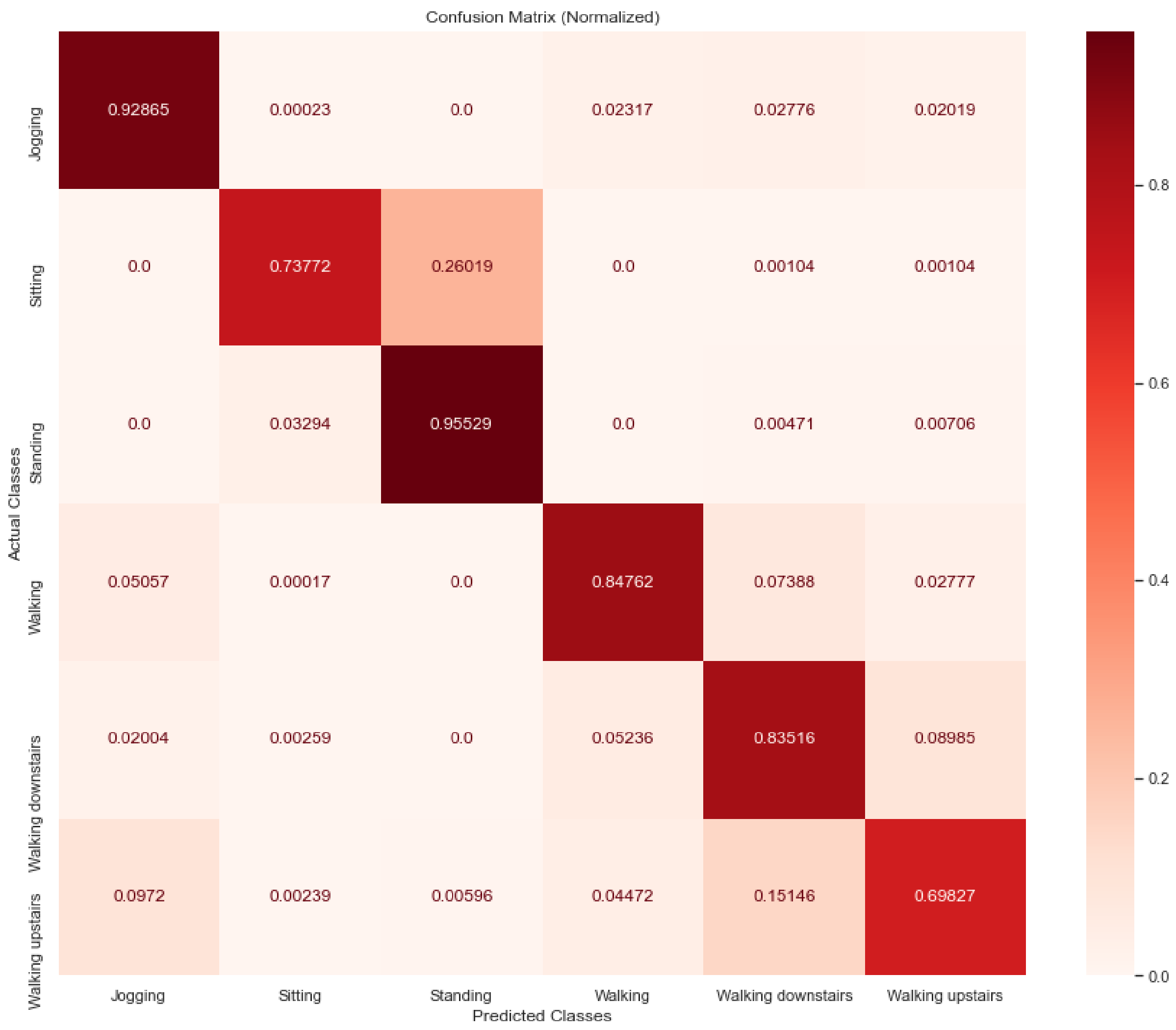


*Figure 32: Normalised confusion matrix of the classes predicted by the model built on the WISDM dataset after feature selection compared to the actual classes found in the test dataset.*

We visualised the accuracy over time for the five models which achieved the highest F1 score on this dataset in Figure 33. All these best performing models were built on the dataset after feature selection. Here, the CNN-LSTM model using a one-stage classifier was once again the fastest to train, reaching peak performance within one minute.

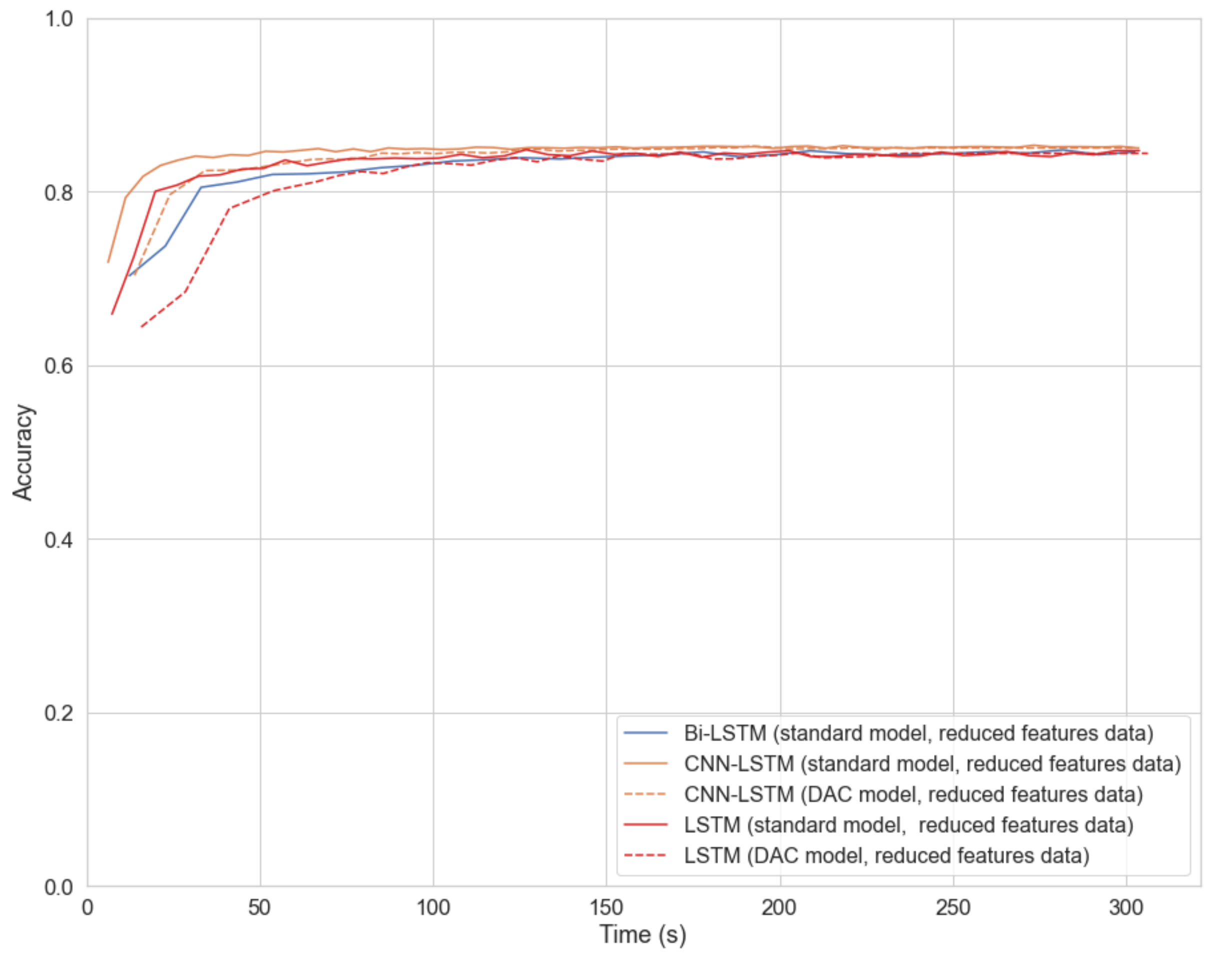


*Figure 33: Plot of test accuracy against time for some of the best performing models on the WISDM dataset. DAC models are plotted with a dashed line here.*

Overall, the results from our experiments on the WISDM dataset gave us clear insights to the best architecture for HAR applications on this dataset. The CNN-LSTM model using the one-stage classifier was either the best performing model or the joint-best performing model across different versions of the dataset, while also being the fastest to train. This makes it a great choice for HAR applications, and researchers building HAR models on this dataset should explore the possibility of further improving the classification performance of CNN-LSTM models built on this dataset whilst also maintaining low training speeds.

Neither the DAC approach nor our oversampling method led to any improvements in classification performance on models built on this dataset; in fact, these methods did little but slow down the training process of models in most cases. The DAC approach was likely less successful on this dataset as the dataset contained twice as many dynamic activity classes as stationary classes. This meant that the second stage classifier for the dynamic classes still had four classes to differentiate between, not many fewer than the six that the one-stage classifier had to differentiate between. This problem highlights the fact that, even though the DAC approach may improve performance on some models on some datasets, it is not applicable to all datasets.

Oversampling was likely unsuccessful in improving model performance here due to the oversampling method we used. By simply copying segments in less common classes, we likely caused models to be overfitted on the training dataset, causing them to be less applicable to new data. More complex oversampling techniques, such as those using generative adversarial network to generate similar samples in less common classes, have been successful in improving HAR models (Alharbi et al., 2020), and so future research should test whether such techniques could fare better on this dataset.

The use of feature selection did improve the performance of our models built on the WISDM dataset. Using only one feature, *total-acceleration (Y)*, we were able to achieve significantly higher F1 scores on the test dataset whilst also reducing computational cost. Feature selection should definitely be utilized in future research into building HAR models on this dataset.

## *7.3 OVERALL FINDINGS*

In each dataset used in our experiments, most architectures achieved relatively similar F1 scores after five minutes of training. The largest differences found were in how long it took models to reach such classification performance. The CNN-LSTM was found to be the fastest architecture to train across both datasets, whilst the bi-directional LSTM was the slowest. Our focus was to find which types of architectures were the best choice when building HAR models for practical applications, such as for fitness tracking purposes executed on devices with much less processing power than the models were trained and tested on here (Kranz et al., 2013). For HAR models to be trained to a useful level for such purposes, a large amount of training data is required, and so it is important that any models built are sufficiently lightweight in terms of computational cost. Therefore, the findings of our experiments recommend that the CNN-LSTM is the most promising architecture for future research, as it achieved amongst the highest levels of classification performance in both datasets after only a short period of training.

Looking at the plots we produced of the training process of CNN-LSTM models in our experiments, it can be seen that these models generally reach high levels of classification accuracy almost instantaneously. We propose that this is due to the convolutional layers present in the architecture (see Figure 5). These layers performed feature extraction, transforming the input data into a format which the LSTM layer in the architecture could learn from both quickly and effectively.

The DAC approach used in our experiments did not, generally, improve the performance of our models. The only models it did improve were the models built on the UCI dataset after feature selection; however, these DAC models were still not any better than the one-stage models built on the original version of the dataset. Therefore, we find it unlikely to be worthwhile using this DAC approach when building LSTM models for HAR purposes, as it increases computation time whilst rarely improving the quality of models — and is not applicable to all datasets.

Oversampling was not found to improve performance in the WISDM dataset, which had a class imbalance problem. As stated in the previous subsection, we propose that this is most likely due to the oversampling approach we adopted. It is very possible that more advanced oversampling methods which create synthetic segments similar to the segments found in the original dataset, rather than creating exact copies, could nonetheless lead to improved classification performance in future research.

Feature selection improved performance in the WISDM dataset but not the UCI Smartphones dataset. This could be caused by the random forest classifier adopting too harsh a selection process when choosing the most important features in the UCI Smartphones dataset, causing it to remove features which are important for the models' training process. From our findings, we would recommend that future researchers attempt to use feature selection when building HAR models, but make sure to first build models on the original dataset to compare whether feature selection leads to any improvement. It is also possible that other dimensionality reduction techniques, such as principal component analysis, may lead to better performance than simply reducing dimensionality by removing less important features.

## *7.4 LIMITATIONS AND FUTURE RESEARCH*

In this project, we aimed to compare a standard set of LSTM architectures in their suitability for HAR applications. By building models using similar hyperparameters and with similar levels of complexity, we aimed to determine more definitively which architectures show the most promise for such problems. However, it is possible that if hyperparameters were adjusted or models were built in a slightly different way, the architectures may perform better or worse than they did here. In future research it may be worthwhile to experiment with a range of different versions of each architecture, finding the optimum version of each LSTM architecture for an even fairer comparison. Even so, we believe that our experiments provide a good starting point for such future research.

Though we tested four commonly used LSTM architectures in our experiments, there are still many other machine learning methods which may provide even better performance than we found. As noted in section 1, CNN architectures have seen much success in HAR applications, and it would be worthwhile to compare CNN models to our LSTM models in future research.

As noted in section 6, the accuracy over time plots for the DAC models built on the WISDM dataset were rather inaccurate, due to an error in how we calculated the time elapsed in these models. Due to time limitations, we were not able to address this issue, though it is important to bear this in mind when looking at these visualisations.

## *7.5 CONCLUSION*

In these experiments, we aimed to compare four commonly used LSTM architectures in their ability to correctly classify activities in two commonly used HAR datasets. Rather than focusing on achieving state-of-the-art performance, we aimed to compare basic, standardised versions of the architectures to assess the merits of each architecture more fairly. Also, rather than focusing solely on the classification performance of models after they had been trained, we were interested in how long it took the models to reach such levels of performance. This is an important consideration for HAR applications; due to the amount of data needed to effectively train HAR models, and due to the limited computing power available on devices which run HAR applications, it is necessary that HAR models are built as lightweight as possible.

The most promising architecture found in our experiments was the CNN-LSTM, which reached high levels of performance in little time whilst also being amongst the best in terms of classification performance across datasets. Future research should use our findings to design CNN-LSTM models which achieve even better classification performance without sacrificing training speeds too heavily. It would also be worthwhile to test the CNN-LSTM architecture alongside standardised CNN models, in a similar manner to we did in the experiment, to see how the CNN-LSTM architecture compares in terms of training speed and classification performance for HAR applications.

In our experiments, we also tested two data manipulation techniques on the datasets, to observe whether this would lead to improved performance. The results of these tests were somewhat inconclusive. The oversampling technique applied to the WISDM dataset did not improve performance, though we believe that employing more advanced oversampling techniques could still prove beneficial; future research should investigate this. Feature selection did improve performance on the WISDM dataset, but not on the UCI Smartphones dataset. This also warrants further investigation; it is possible that different dimensionality reduction techniques may perform better on either or both datasets.

Overall, the findings detailed in this paper provide a good starting point for researchers interested in building HAR models for practical applications. Our findings should be built upon in future research, focusing on how to improve upon the classification performance of our models whilst keeping training times low.

## REFERENCES

Aggarwal, J. K., & Cai, Q. (1997). Human motion analysis: A review. *Proceedings of the IEEE Nonrigid and Articulated Motion Workshop*, 90–102. https://doi.org/10.1109/namw.1997.609859

Alani, A. A., Cosma, G., & Taherkhani, A. (2020). Classifying Imbalanced Multi-modal Sensor Data for Human Activity Recognition in a Smart Home using Deep Learning. *Proceedings of the International Joint Conference on Neural Networks, 1-8*. https://doi.org/10.1109/IJCNN48605.2020.9207697

Alharbi, F., Ouarbya, L., & Ward, J. A. (2020). Synthetic Sensor Data for Human Activity Recognition. *Proceedings of the International Joint Conference on Neural Networks*, 1–9. https://doi.org/10.1109/IJCNN48605.2020.9206624

Anguita, D., Ghio, A., Oneto, L., Parra, X., & Reyes-Ortiz, J. L. (2013). A Public Domain Dataset for Human Activity Recognition Using Smartphones. *Proceedings of the 21st European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning*, 437–442. https://doi.org/10.3390/s20082200

Ben-Arie, J., Wang, Z., Pandit, P., & Rajaram, S. (2002). Human activity recognition using multidimensional indexing. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *24*(8), 1091–1104. https://doi.org/10.1109/TPAMI.2002.1023805

Buda, M., Maki, A., & Mazurowski, M. A. (2018). A systematic study of the class imbalance problem in convolutional neural networks. *Neural Networks*, *106*, 249–259. https://doi.org/10.1016/j.neunet.2018.07.011

Cai, J., Luo, J., Wang, S., & Yang, S. (2018). Feature selection in machine learning: A new perspective. *Neurocomputing*, *300*, 70–79. https://doi.org/10.1016/j.neucom.2017.11.077

Chawla, N. V., Bowyer, K. W., Hall, L. O., & Kegelmeyer, W. P. (2002). SMOTE: Synthetic Minority Over-sampling Technique. *Journal of Artificial Intelligence Research*, *16*, 321–357.

Chen, K., Zhang, D., Yao, L., Guo, B., Yu, Z., & Liu, Y. (2021). Deep learning for sensor-based human activity recognition: Overview, challenges, and opportunities. *ACM Computing Surveys*, *54*(4), 1-40. https://doi.org/10.1145/3447744

Chen, X., Liu, X., Qian, Y., Gales, M. J. F., & Woodland, P. C. (2016). CUED-RNNLM - An open-source toolkit for efficient training and evaluation of recurrent neural network language models. P*roceedings of the 2016 IEEE International Conference on Acoustics,* *Speech and Signal Processing*, 6000–6004. https://doi.org/10.1109/ICASSP.2016.7472829

Chen, Y., Zhong, K., Zhang, J., Sun, Q., & Zhao, X. (2016). LSTM Networks for Mobile Human Activity Recognition. *Proceedings of the 2016 International Conference on Artificial Intelligence: Technologies and Applications*, 50–53. https://doi.org/10.2991/icaita-16.2016.13

Cho, H., & Yoon, S. M. (2018). Divide and conquer-based 1D CNN human activity recognition using test data sharpening. *Sensors (Switzerland)*, *18*(4), 1–24. https://doi.org/10.3390/s18041055

Dernbach, S., Das, B., Krishnan, N. C., Thomas, B. L., & Cook, D. J. (2012). Simple and complex activity recognition through smart phones. *Proceedings of the 8th International Conference on Intelligent Environments*, 214–221. https://doi.org/10.1109/IE.2012.39

Farah, J. D., Baddour, N., & Lemaire, E. D. (2019). Design, development, and evaluation of a local sensor-based gait phase recognition system using a logistic model decision tree for orthosis-control. *Journal of Neuroengineering and Rehabilitation*, *16(1)*, 1–11. https://doi.org/10.1186/s12984-019-0486-z

Genuer, R., Poggi, J. M., & Tuleau-Malot, C. (2010). Variable selection using random forests. *Pattern Recognition Letters*, *31*(14), 2225–2236. https://doi.org/10.1016/j.patrec.2010.03.014

Graves, A., Abdel-Rahman, M., & Hinton, G. (2013). Speech Recognition with Deep Recurrent Neural Networks, *Proceedings of the 2013 IEEE international conference on acoustics, speech and signal processing*, 6645-6649.

Greff, K., Srivastava, R. K., Koutnik, J., Steunebrink, B. R., & Schmidhuber, J. (2017). LSTM: A Search Space Odyssey. *IEEE Transactions on Neural Networks and Learning Systems*, *28*(10), 2222–2232. https://doi.org/10.1109/TNNLS.2016.2582924

Hochreiter, S., & Schmidhuber, J. (1997). Long Short-Term Memory. *Neural Computation*, *9*(8), 1735–1780. https://doi.org/10.1162/neco.1997.9.8.1735

Ignatov, A. (2016). Real-time human activity recognition from accelerometer data using Convolutional Neural Networks. *Applied Soft Computing Journal*, *62*, 915–922. https://doi.org/10.1016/j.asoc.2017.09.027

Japkowicz, N. (2000). The Class Imbalance Problem: Significance and Strategies. *Proceedings of the 2000 International Conference on Artificial Intelligence*, 111-117.

Jiang, W., & Yin, Z. (2015). Human activity recognition using wearable sensors by deep convolutional neural networks. *Proceedings of the 2015 ACM Multimedia Conference*, 1307–1310. https://doi.org/10.1145/2733373.2806333

Khan, A. M., Lee, Y. K., Lee, S. Y., & Kim, T. S. (2010). A triaxial accelerometer-based physical-activity recognition via augmented-signal features and a hierarchical recognizer. *IEEE Transactions on Information Technology in Biomedicine*, *14*(5), 1166–1172. https://doi.org/10.1109/TITB.2010.2051955

Kranz, M., Möller, A., Hammerla, N., Diewald, S., Plötz, T., Olivier, P., & Roalter, L. (2013). The mobile fitness coach: Towards individualized skill assessment using personalized mobile devices. *Pervasive and Mobile Computing*, *9*(2), 203–215. https://doi.org/10.1016/j.pmcj.2012.06.002

Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). ImageNet Classification with Deep Convolutional Neural Networks. *Advances in neural information processing systems*, *25*, 1097-1105.

Kwapisz, J. R., Weiss, G. M., & Moore, S. A. (2011). Activity recognition using cell phone accelerometers. *ACM SIGKDD Explorations Newsletter*, *12*(2), 74–82. https://doi.org/10.1145/1964897.1964918

Li, Z., Liu, Y., Guo, X., & Zhang, J. (2020). Multi-convLSTM neural network for sensor-based human activity recognition. *Journal of Physics: Conference Series*, *1682*(1), 1-11. https://doi.org/10.1088/1742-6596/1682/1/012062

Liu, L., Wang, S., Qiong, Q., Wen, J., & Rosenblum, D. S. (2018). Learning structures of interval-based Bayesian networks in probabilistic generative model for human complex activity recognition. *Pattern Recognition*, *81*, 545–561. https://doi.org/10.1016/j.patcog.2018.04.022

Mäntyjärvi, J., Himberg, J., & Seppänen, T. (2001). Recognizing human motion with multiple acceleration sensors. *Proceedings of the IEEE International Conference on Systems, Man and Cybernetics*, *2*, 747–752. https://doi.org/10.1109/ICSMC.2001.973004

Mazilu, S., Blanke, U., Hardegger, M., Tröster, G., Gazit, E., & Hausdorff, J. M. (2014). GaitAssist: A Daily-Life Support and Training System for Parkinson's Disease Patients with Freezing of Gait. *Proceedings of the ACM Conference on Human Factors in Computing Systems*, 2531–2540. https://doi.org/10.1145/2556288.2557278

Mekruksavanich, S., & Jitpattanakul, A. (2021). A Multichannel CNN-LSTM Network for

Daily Activity Recognition using Smartwatch Sensor Data. *Proceedings of the 2021 Joint 6th International Conference on Digital Arts, Media and Technology with 4th ECTI Northern Section Conference on Electrical, Electronics, Computer and Telecommunication Engineering,* 277–280. https://doi.org/10.1109/ECTIDAMTNCON51128.2021.9425769

Minh Dang, L., Min, K., Wang, H., Jalil Piran, M., Hee Lee, C., & Moon, H. (2020). Sensor-based and vision-based human activity recognition: A comprehensive survey. *Pattern Recognition*, *108, 1-24.* https://doi.org/10.1016/j.patcog.2020.107561

Mohammed Hashim, B. A., & Amutha, R. (2021). Human activity recognition based on smartphone using fast feature dimensionality reduction technique. *Journal of Ambient Intelligence and Humanized Computing, 12*(2), 2365–2374. https://doi.org/10.1007/s12652-020-02351-x

Murad, A., & Pyun, J. Y. (2017). Deep recurrent neural networks for human activity recognition. *Sensors (Switzerland)*, *17*(11), 1-17. https://doi.org/10.3390/s17112556

Mutegeki, R., & Han, D. S. (2020). A CNN-LSTM Approach to Human Activity Recognition. *Proceedings of the 2020 International Conference on Artificial Intelligence in Information and Communication*, 362–366. https://doi.org/10.1109/ICAIIC48513.2020.9065078

Nunes, U. M., Faria, D. R., & Peixoto, P. (2017). A human activity recognition framework using max-min features and key poses with differential evolution random forests classifier. *Pattern Recognition Letters*, *99*, 21–31. https://doi.org/10.1016/j.patrec.2017.05.004

Olah, C., 2015. *Understanding LSTM Networks*. [online] Colah's blog. Available at: <https://colah.github.io/posts/2015-08-Understanding-LSTMs/> [Accessed 17 August 2021].

Ordóñez, F. J., Roggen, D., Liu, Y., Xiao, W., Chao, H.-C., & Chu, P. (2016). Deep convolutional and LSTM recurrent neural networks for multimodal wearable activity recognition. *Sensors (Switzerland)*, *16*(1), 1-25. https://doi.org/10.3390/s16010115

Perez-Munuzuri, V., & Chua, L. O. (1993). Autowaves for Image Processing on a Two-Dimensional CNN Array of Excitable Nonlinear Circuits: Flat and Wrinkled Labyrinths. *IEEE Transactions on Circuits and Systems I: Fundamental Theory and Applications*, *40*(3), 174–181. https://doi.org/10.1109/81.222798

Phung, V. H., & Rhee, E. J. (2019). A High-accuracy model average ensemble of convolutional neural networks for classification of cloud image patches on small datasets. *Applied Sciences (Switzerland)*, *9*(21), 1-16. https://doi.org/10.3390/app9214500

Pigou, L., van den Oord, A., Dieleman, S., Van Herreweghe, M., & Dambre, J. (2018). Beyond Temporal Pooling: Recurrence and Temporal Convolutions for Gesture Recognition in Video. *International Journal of Computer Vision*, *126*(4), 430–439. https://doi.org/10.1007/s11263-016-0957-7

Piyathilaka, L., & Kodagoda, S. (2015). Human activity recognition for domestic robots. In *Field and Service Robotics* (395–408). Springer, Charm. https://doi.org/10.1007/978-3-319-07488-7

Qin, J., Liu, L., Zhang, Z., Wang, Y., & Shao, L. (2016). Compressive sequential learning for action similarity labeling. *IEEE Transactions on Image Processing*, *25*(2), 756–769. https://doi.org/10.1109/TIP.2015.2508600

Ren, S., He, K., Girshick, R., & Sun, J. (2017). Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *39*(6), 1137–1149. https://doi.org/10.1109/TPAMI.2016.2577031

Ronao, C. A., & Cho, S. B. (2016). Human activity recognition with smartphone sensors using deep learning neural networks. *Expert Systems with Applications*, *59*, 235–244. https://doi.org/10.1016/j.eswa.2016.04.032

Sainath, T. N., Vinyals, O., Senior, A., & Sak, H. (2015). Convolutional, Long Short-Term Memory, fully connected Deep Neural Networks. *Proceedings of the 2015 IEEE International Conference on Acoustics, Speech and Signal Processing*, 4580–4584. https://doi.org/10.1109/ICASSP.2015.7178838

Schmidhuber, J., Wierstra, D., & Gomez, F. (2005). Evolino: Hybrid neuroevolution/optimal linear search for sequence prediction. *Proceedings of the 19th International Joint Conference on Artificial Intelligence*, 853–858.

Singh, D., Merdivan, E., Psychoula, I., Kropf, J., Hanke, S., Geist, M., & Holzinger, A. (2017). Human activity recognition using recurrent neural networks. *Proceedings of the International Cross-Domain Conference for Machine Learning and Knowledge Extraction*, 267–274. https://doi.org/10.1007/978-3-319-66808-6_18

Singh, S. P., Sharma, M. K., Lay-Ekuakille, A., Gangwar, D., & Gupta, S. (2021). Deep ConvLSTM with Self-Attention for Human Activity Decoding Using Wearable Sensors. *IEEE Sensors Journal*, *21*(6), 8575–8582.

https://doi.org/10.1109/JSEN.2020.3045135

Uddin, M. T., & Uddin, M. A. (2015). A guided random forest based feature selection approach for activity recognition. *Proceedings of the 2nd International Conference on Electrical Engineering and Information and Communication Technology*, 1-6. https://doi.org/10.1109/ICEEICT.2015.7307376

Wang, J., Chen, Y., Hao, S., Peng, X., & Hu, L. (2018). Deep learning for sensor-based activity recognition: A survey. *Pattern Recognition Letters*, *119*, 3–11. https://doi.org/10.1016/j.patrec.2018.02.010

Weiss, G. M., Yoneda, K., & Hayajneh, T. (2019). Smartphone and Smartwatch-Based Biometrics Using Activities of Daily Living. *IEEE Access*, *7*, 133190–133202. https://doi.org/10.1109/ACCESS.2019.2940729

Welch, B. L. (1947). The generalisation of student's problems when several different population variances are involved. *Biometrika*, *34*(2), 28–35. https://doi.org/10.1093/biomet/34.1-2.28

Yang, J. B., Nguyen, M. N., San, P. P., Li, X. L., & Krishnaswamy, S. (2015). Deep convolutional neural networks on multichannel time series for human activity recognition. *Proceedings of the International Joint Conference on Artificial Intelligence*, 3995–4001.

Yao, S., Hu, S., Zhao, Y., Zhang, A., & Abdelzaher, T. (2017). DeepSense: A unified deep learning framework for time-series mobile sensing data processing. *Proceedings of the 26th International World Wide Web Conference,* 351–360. https://doi.org/10.1145/3038912.3052577

Zeng, M., Nguyen, L. T., Yu, B., Mengshoel, O. J., Zhu, J., Wu, P., & Zhang, J. (2015). Convolutional Neural Networks for human activity recognition using mobile sensors. *Proceedings of the 6th International Conference on Mobile Computing, Applications and Services*, 197–205. https://doi.org/10.4108/icst.mobicase.2014.257786

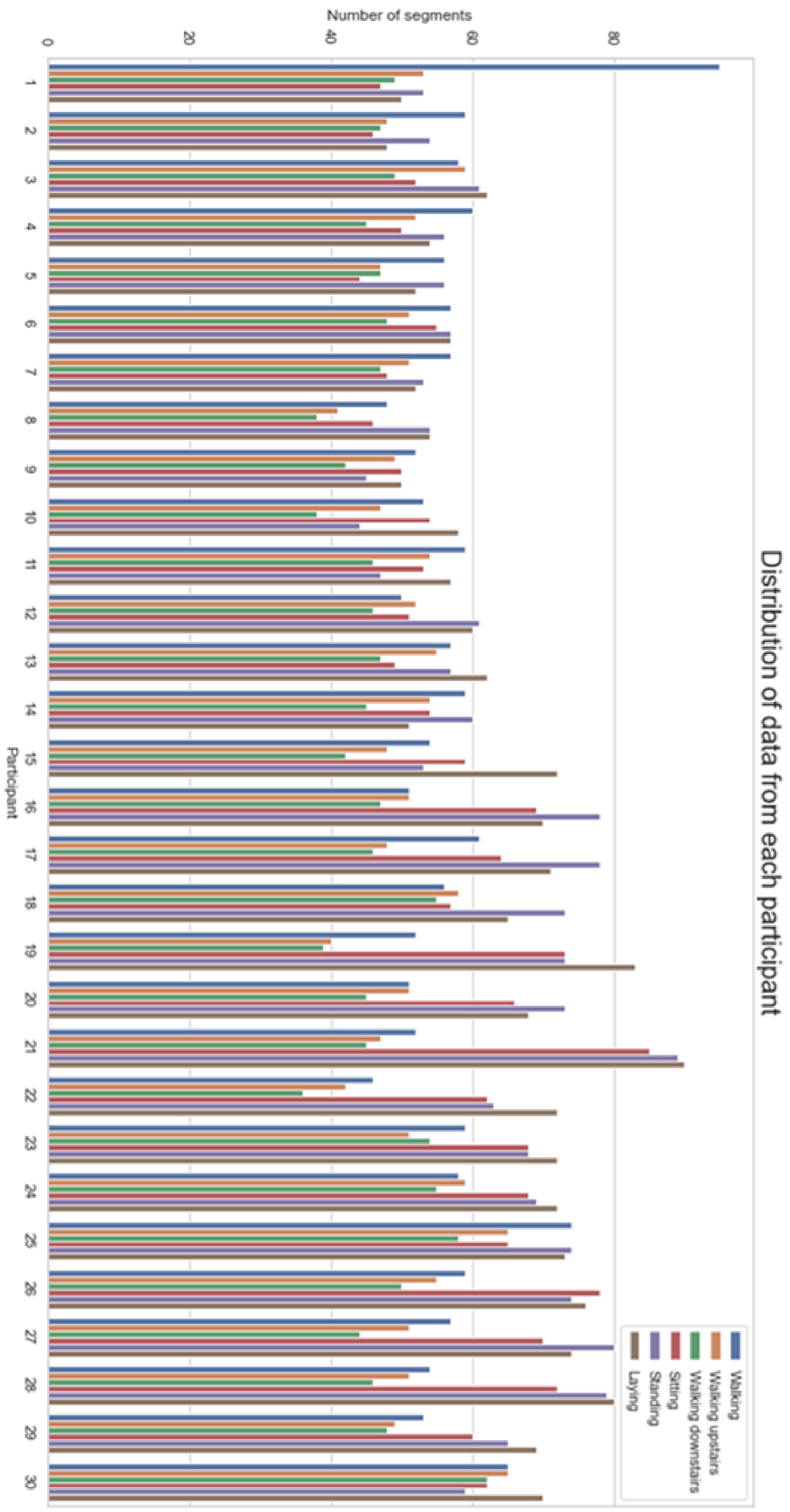
Zhao, Y., Yang, R., Chevalier, G., Xu, X., & Zhang, Z. (2018). Deep Residual Bidir-LSTM for Human Activity Recognition Using Wearable Sensors. *Mathematical Problems in Engineering, 2018,* 1-14. https://doi.org/10.1155/2018/7316954

Zhu, X., & Qiu, H. (2016). High accuracy human activity recognition based on sparse locality preserving projections. *PloS one*, *11*(11), 1-18.

## APPENDICES

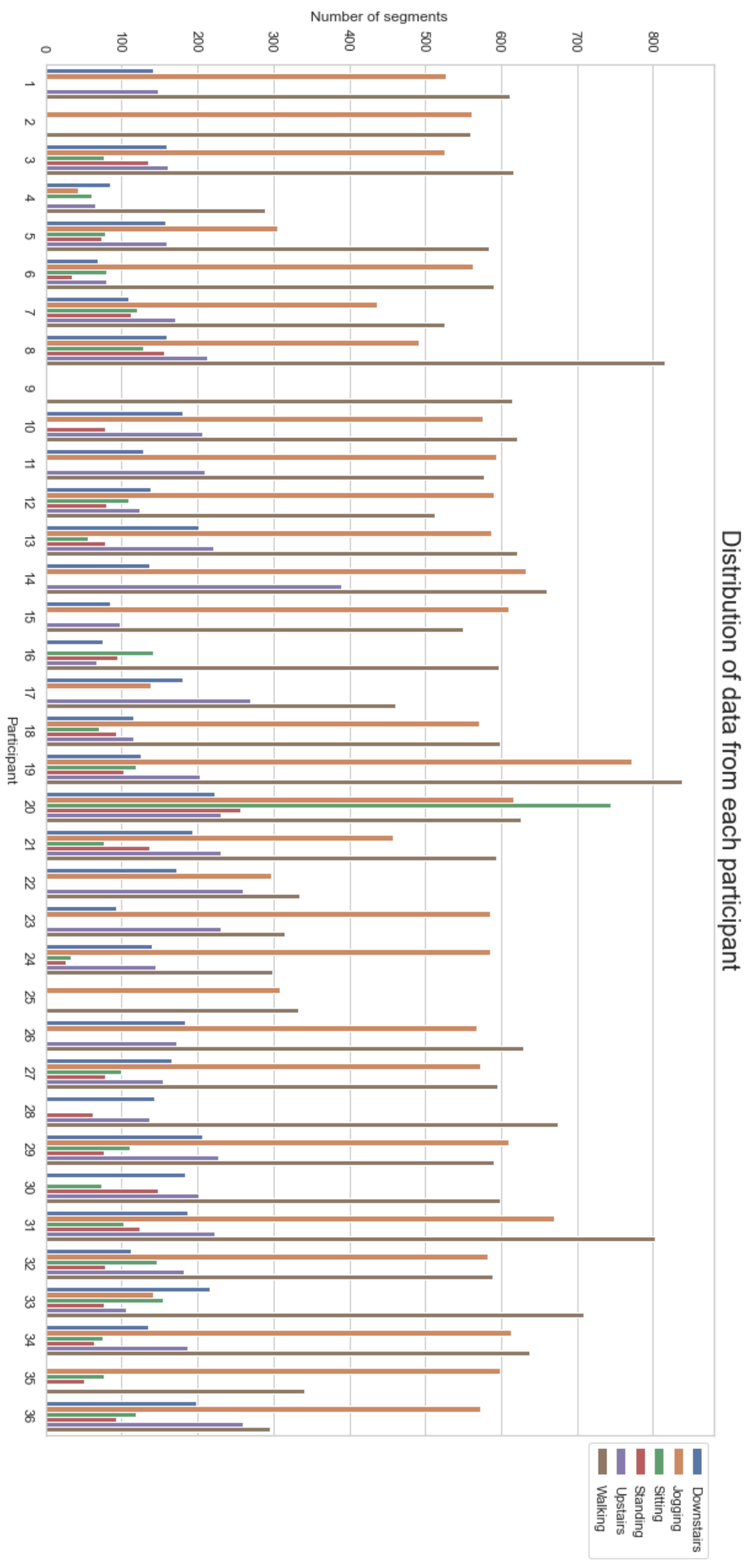
### APPENDIX A

Visualisation of the data obtained from all participants in the UCI Smartphones dataset.



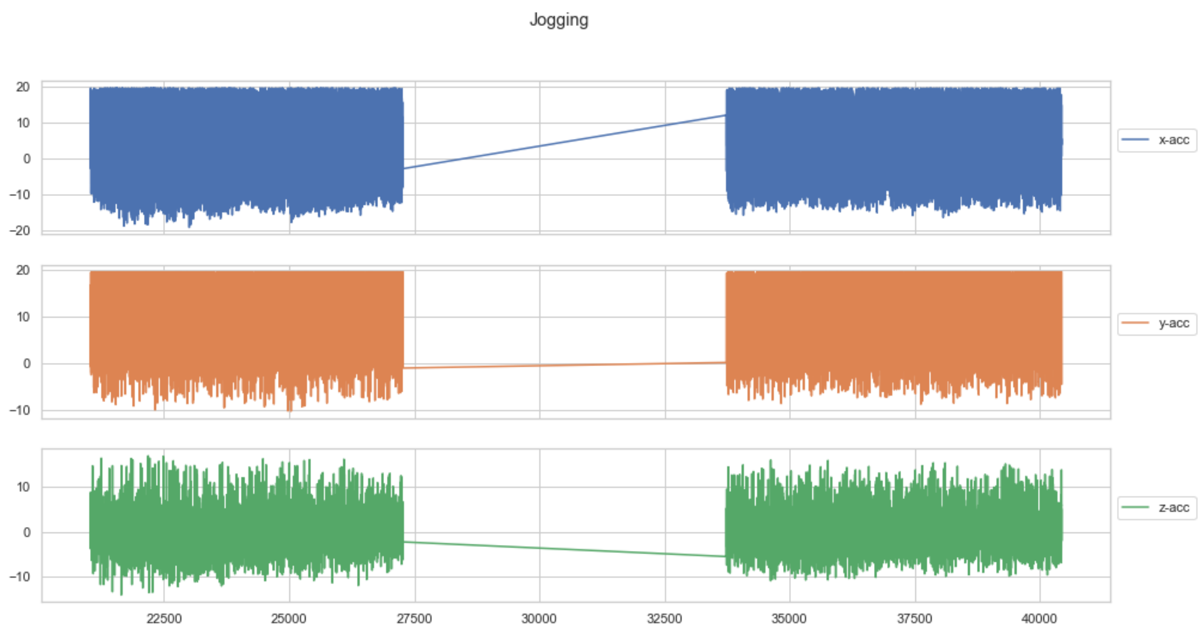
APPENDIX B

Visualisation of the data obtained from all participants in the WISDM dataset.



## APPENDIX C

Visualisation of activities in the WISDM dataset before any pre-processing had been applied. The activity shown in this visualisation, *jogging*, appeared to have been pre-processed incorrectly when looking at the activity plot in Figure 15, as the x- and y- acceleration data was bunched up around 1. However, the unprocessed data, visualised below, shows this same artefact.


Jogging
x-acc
y-acc
z-acc