Human Activity Recognition using the MotionSense Dataset

Prepared by:
Ntsako Phiri PHRANN001
Mukundi Mangena MNGBLE005

Prepared for:
EEE4114F
Department of Electrical Engineering
University of Cape Town

May 30, 2025

Declaration

1. I know that plagiarism is wrong. Plagiarism is to use another's work and pretend that it is one's own.
2. I have used the IEEE convention for citation and referencing. Each contribution to, and quotation in, this report from the work(s) of other people has been attributed, and has been cited and referenced.
3. This report is my own work.
4. I have not allowed, and will not allow, anyone to copy my work with the intention of passing it off as their own work or part thereof.

May 30, 2025

Ntsako Phiri — Date

May 30, 2025

Mukundi Mangena — Date

Contents

List of Figures

Chapter 1

Introduction

1.1 Background

Human Activity Recognition (HAR) is an important area of machine learning research that uses sensor data(often from smartphones) to identify and categorise physical activities undertaken by individuals. With the increased use of smartphones, more personal devices have been equipped with sensors such as accelerometers, gyroscopes, and magnetometers. These sensors have allowed HAR to emerge through the data collected from the sensors. This sensor data has allowed HAR systems to understand and classify human movements, which can have profound implications for many applications, including health monitoring, elderly care, and intelligent environments. The proliferation of sensor-rich devices has also enabled the creation of extensive datasets, which foster deeper research into various aspects of human behavior and contribute to advancements in data mining and machine learning.

Using iPhone 6s, an array of time-series data from accelerometer and gyroscope sensors from the phones was used for the MotionSense dataset. This dataset was used to document six activities (walking, jogging, sitting, standing, walking upstairs, and walking downstairs) conducted by 24 users. This dataset is very valuable because it includes demographic and physical attributes, allowing not just activity classification, but also sensitive attribute inference from motion data, such as gender or personality qualities. The objective of this report is to design, implement, and test an effective HAR system using the MotionSense dataset. This system aims to accurately classify the six human activities based on iPhones' sensor data.

1.2 Literature Review

The field of HAR using smartphone sensors has made significant strides over the past years, particularly with the use of deep learning architectures and sophisticated preprocessing methods. This literature review outlines major findings from relevant studies, including those included in the documents, in order to contextualise the MotionSense dataset analysis.

1.2.1. Preprocessing Techniques in HAR

Preprocessing data is a vital stage in HAR that addresses these aspects - noise, dimensionality, and data imbalance- all of which can have an immense impact on model performance. Various research papers were used to highlight a number of preprocessing methods, the first being data cleansing and normalization. In the study by Masum et al. [1], the MHEALTH dataset which uses accelerometer,

gyroscope, and magnetometer data, the data was preprocessed by removing invalid label occurrences (such as transitions between activities) to reduce dataset imbalance. Sensor data was 'harmonised' using normalisation techniques such as standardising gyroscope values (in radians/s) and accelerometer measurements (in m/s^2). Furthermore, to improve model stability, Zainab and Srivastava [2] used Principal Component Analysis (PCA) to reduce feature dimensionality by taking a larger number of features to transform it into a new set of features, thus resolving inconsistencies in smartphone sensor data. According to Saeed et al.[3], deep neural networks can often learn abstract representations from raw data with minimal pre-processing other than z-normalization. [3] also highlighted that minimal pre-processing, besides z-normalization, is often sufficient for deep neural networks to learn abstract representations directly from raw data.

Signal Transformation processing techniques like the Fourier and wavelet transforms are widely used to extract meaningful features from time-series data. Wu et al. [4] used Fast Fourier Transform (FFT) magnitudes to capture the cyclic patterns in accelerometer and gyroscope data, and reported that FFT features outperformed other time and frequency-based features. While Masum et al. [1] also applied a Butterworth low-pass filter to smooth accelerometer data, mitigating gravitational influences and enhancing signal quality. Nouriani et al. [5] proposed converting IMU signals into spectrograms by applying FFT for a 5-second window, and then combining 3-axis acceleration signals into an RGB image to include more information for computer vision deep learning algorithms.

Feature selection is crucial for all datasets in order to reduce computational complexity and improve model accuracy. [2] used PCA to identify orthogonal features with significant variance, while [4] extracted features like mean, standard deviation, and acceleration magnitude to optimise classification performance. These methods show how advanced preprocessing consistently improves model stability which can be implemented for MotionSense dataset.

1.2.2. Classical Machine Learning Models

Classical machine learning models, such as K-Nearest Neighbors (KNN), Support Vector Machines (SVM), Random Forest (RF), Decision Trees (DT), and Naïve Bayes (NB), have been widely used in HAR. The following studies reviewed shed light on their effectiveness. Zainab and Srivastava [2] assessed different classifiers on human activity data gathered from smartphone accelerometers, which included SVMs, NBs, and Logistic Regression (LR), as well as Deep Neural Networks. Their simplest Deep Nets model achieved a maximum accuracy of 95.71% for male participants and 94.62% for female participants. For SVM, they employed an 'rbf' kernel with a gamma value of 0.01, which yielded the best performance. The Random Forest classifier was configured with 100 decision trees to enhance it's performance. Wu et al. [4] compared classifiers using data from iPod Touch, discovering that KNN produced the highest accuracy rates: 52.3%-79.4% for stair walking, 91.7% for jogging, 90.1%-94.1% for walking on flat surfaces, and 100% for sitting. Furthermore, [4] noted that incorporating gyroscope data improved classification accuracy by 3.1% to 13.4% compared to using only accelerometer data. [1] assessed the MHEALTH dataset using RF, SVM, and NB, alongside deep learning models. [1] applied preprocessing techniques such as removing invalid cases, addressing class imbalances, low-pass filtering, and Principal Component Analysis. The self-supervised learning study by Saeed et al. [3] used Random Init. for baseline comparison, showing that training an activity classification model on top of randomly initialised convolutional layers performs poorly, indicating that improvements are not

solely due to the activity classifier. These findings indicate that although classical models can achieve commendable accuracies with effective feature engineering, they are generally outperformed by deep learning approaches when applied to complex, multivariate time-series data like MotionSense.

1.2.3. Deep Learning Approaches

Deep learning architectures, especially those that integrate convolutional and recurrent layers, have demonstrated exceptional efficacy in Human Activity Recognition (HAR) due to their capacity to extract hierarchical features from unprocessed sensor data. Masum et al. [1] discussed the use of deep Convolutional Neural Networks (CNN) for HAR while investigating issues of convnets layers with the acknowledgement procedure and achieving a precision of 94.97% on the test set with grungy sensor information and 95.75% with auxiliary learning of the first Fourier transform. Malekzadeh et al. [6] indicated that CNNs are inherently adaptive to input data of variable dimensions and are mainly used for feature extraction.

[1] also used Long-Short Term Memory (LSTM) and Multilayer Perceptron (MLP) as deep learning approaches. [2] noted that T. LSTM cells may detect correlations in time-dependent input without combining the timesteps, which is a notable distinction from the 1D convolutional neural network. Nouriani et al. [5] stated that CNNs combined with LSTM cells have been successful in activity recognition due to their ability to identify temporal patterns in time series data. The CNN-RNN model with a Dimension-Adaptive Pooling (DAP) layer significantly improves accuracy in the test setting, implying that DAP contributes to the model's capacity to generalise more effectively to the test dataset.

Preprocessing techniques like PCA, Fourier transforms, and low-pass filtering could be applied to reduce noise and dimensionality while preserving relevant features. Normalisation is essential to align accelerometer and gyroscope data. While selecting deep learning models, particularly hybrid CNN-LSTM architectures, is likely to yield the highest accuracies, the classical models (such as RF and SVM) can serve as baselines. However these classical models tend to require extensive feature engineering. The use of Dimension-Adaptive Neural Architecture (DANA) with a Dimension-Adaptive Pooling (DAP) layer and Dimension-Adaptive Training (DAT) can enhance robustness to variable sampling rates and sensor availability.

Special attention should be given to activities like stair climbing(walking upstairs/downstairs), which are prone to misclassification. Incorporating gyroscope data, as demonstrated by [4], can improve accuracy for these activities. Training techniques like self-supervised learning can ineffectively leverage the large amount of labeled data in MotionSense to pre-train models. The MotionSense dataset's inclusion of demographic attributes (age, gender, height, weight) aligns with the concept of exploring personalised models or attribute inference, enhancing the applicability of HAR in real-world settings.

In summary, this review looked into the use of deep hybrid networks with systematic preprocessing and advanced training techniques to achieve superior HAR performance on datasets like MotionSense. The combination of convolutional and recurrent layers, coupled with robust feature engineering and adaptive training strategies, offers a promising approach to addressing the challenges of activity classification and sensitive attribute inference.

Chapter 2

The motion sense dataset includes the time-series data generated by an accelerometer and gyroscope sensors [7]. This data was collected using an iphone 6s by 24 participants doing six activities. The participants were put into the same environment and conditions which were downstairs, upstairs, walking, jogging, sitting, and standing.

The goal of this project is to design a model that will be able to accurately identify the each of these 6 activities through some form of classification task. The approach we used in this project was to create a Convolutional Neural Network (CNN) using open source libraries suck as scikit-learn , torch and scipy to handle the data and succesfully train lightweight models.

The DataSet

The motionsense dataset can be found on github or kaggle or other websites that store datasets that could be used for training Machine learning models. The dataset contains multiple CSV file taken from different activities done by the participants. The dataset contains time series data and after successfully loading the data into a pandas Datafrane there was more that 1.4m data points in the data. This dataset initially contains all the data in the dataset however after cleaning the dataset the model used to train model contained the following columns.

- attitude.roll
- attitude.pitch
- attitude.yaw
- userAcceleration.x
- userAcceleration.y
- userAcceleration.z
- rotationRate.x
- rotationRate.y
- rotationRate.z
- act

The goal of the model is to use the first 9 features to find the value in **act**. The feature act is the output feature that the 9 feature will be use to classify.

A particular challenge that we faced with the data was that it was time series data. This means that this data was sampled and stored chronologically. However there is a challenge when time series data because models can easily think that the data is independent and identically distributed.This data needed pre-processing to solve this challenge.

There are multiple methods that can be used to solve this issue and one of them is windowing. This is taken from the concept of using CNN's or image classification. By windowing we allow our data to be in a 2D format which allow us to implement a CNN on the dataset.Windows could be of various sizes and varying them and testing given development time is not too lengthy would be good so as to find the sweets-spot window size. Another possible solution to this problem is converting time -series data to spectrograms and classify these spectrogram images. The final method is using FFT method for the data pre-processing. This method employs using the fourier frequency Transform to transform a window of the time series data eg 2 secs into the Fourier domain which can then now be effectively processed by a CNN.

In this project we aimed to looking at what the effect of different pre-processing methods would be to the final results. Therefore we developed two models one that used FFT and windowing as its dat pre-processing and a 3 way foldout , while the other model used just windowing and a three-way foldout.

2.1 Model Architectures

Model 1 (FFT)

After processing the data into a window slices of a 50 and using scipy.fft to perform fourier transform and performing a 3-way holdout using scikit-learn on the dataset a Cnn model was created to process this data.This model employs a 1D CNN architecture designed to process frequency-domain features extracted . The network begins with a Conv1D layer *(64 filters, kernelsize=3, ReLU activation)* to detect local frequency patterns in the input spectra, followed by MaxPooling1D (poolsize=2) to reduce dimensionality while preserving dominant frequency features. A Dropout layer (0.5 rate) is included for regularization to prevent overfitting. The flattened features are then passed through a Dense layer (100 neurons, ReLU) for high-level feature integration before final classification via a softmax-activated Dense layer matching the number of activity classes. CNNs are particularly suitable for this FFT-transformed data because:

- The frequency spectra contain spatially local patterns where specific frequency bands correlate with different activities,
- Convolutional filters can effectively capture these band-specific signatures through weight sharing, and
- The architecture's translation invariance accommodates natural variability in motion frequency distributions.

The 1D formulation optimally handles the sequential nature of frequency bins while maintaining computational efficiency compared to 2D approaches.This model was then trained for 20 epochsand produced results shown in the section 3.

Model 2 (Windowing)

This second model used exactly the same architecture for the model however the time series data only went through a windowing process and a scikit-learn 3 way holdout. The results of a model are sometimes dependent on the type of preprocessing method. Therefore to assess the effects of windowing we trained for different sizes of the windows. The model was then tested using the train dataset than was never exposed to the dataset earlier.

Chapter 3

Results Analysis

The data is preprocessed with varying window sizes (50, 400, 1000), strides (200), and FFT transformations, with a consistent train-test-validation split of 70:15:15. This results in 39,559 training samples, 8,477 validation samples, and 8,477 test samples for the base model. The model is trained for 10 or 20 epochs, and performance is evaluated using accuracy, loss, confusion matrices, and classification reports. The following results are across different configurations:

3.0.1 Experiment 1: Effect of Epochs (Window Size 50)

The initial experiments focused on a window size of 50 with a step of 25. The model was trained for 10 and 20 epochs.

1. Model with Window Size 50, 10 Epochs:

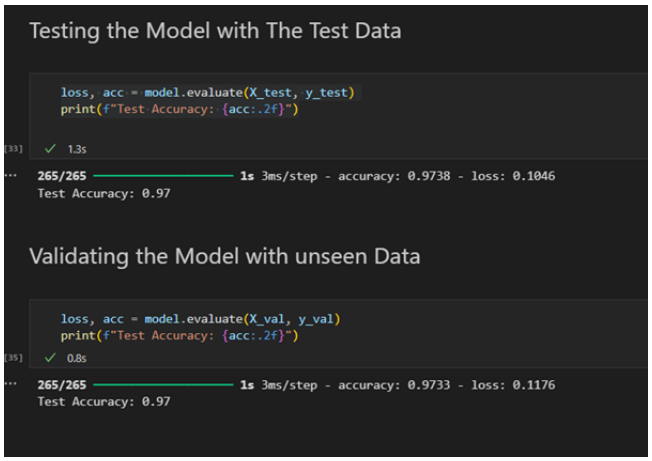


Figure 3.1: Window Size 50, 10 Epochs Training Performance.

- Final training accuracy: 97.38% (Epoch 10: accuracy = 0.9738, loss = 0.1046).
- Final validation accuracy: 97.41% (val_accuracy = 0.9741, val_loss = 0.1176)

2. Model with Window Size 50, 20 Epochs:

- Final training accuracy: 97.41% (Epoch 20: accuracy = 0.9741, loss = 0.1443).
- Final validation accuracy: 97.53% (val_accuracy = 0.9753, val_loss = 0.1405).

Training and validation accuracy and loss curves The model accuracy plot shows training

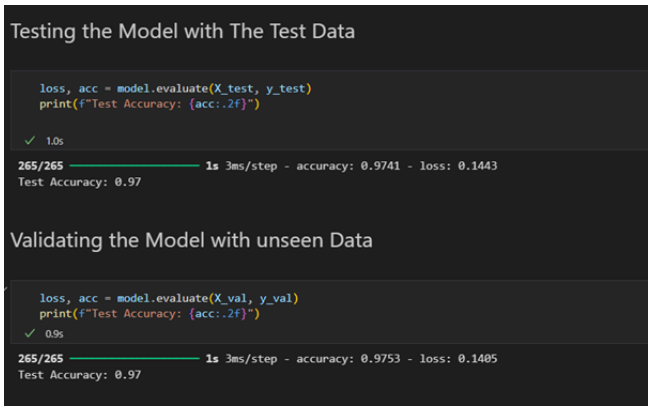


Figure 3.2: Window Size 50, 20 Epochs Training Performance.

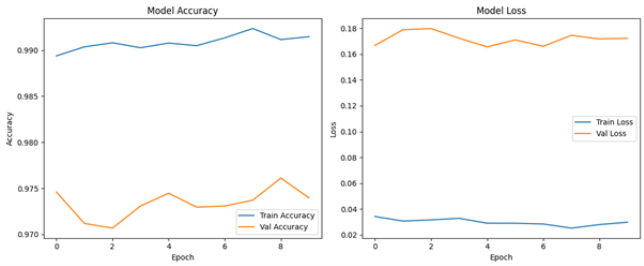


Figure 3.3: Training and validation accuracy and loss curves- Window Size 50

accuracy starting around 0.989 and fluctuating slightly, ending near 0.991. Validation accuracy starts around 0.975, dips to 0.971, and peaks around 0.976 before settling near 0.974. The model loss plot indicates training loss starting low (around 0.035) and remaining relatively stable. Validation loss starts around 0.17, decreases, and then fluctuates between approximately 0.16 and 0.18.

Classification report:

Classification Report:

	precision	recall	f1-score	support
dws	0.91	0.92	0.92	793
ups	0.94	0.91	0.93	922
wlk	0.98	0.98	0.98	2090
jog	0.97	0.98	0.97	805
std	1.00	0.99	0.99	1818
sit	1.00	1.00	1.00	2049
accuracy			0.98	8477
macro avg	0.96	0.97	0.97	8477
weighted avg	0.98	0.98	0.98	8477

Figure 3.4: Classification Report- Test Dats

The overall accuracy for all activities is 0.98. The model performed exceptionally well for 'std' and 'sit'. 'dws' and 'ups' showed slightly lower precision and recall compared to other activities.

Confusion Matrix (Test Data)

- 'dws' was sometimes misclassified as 'ups' (30 instances) and 'wlk' (18 instances).

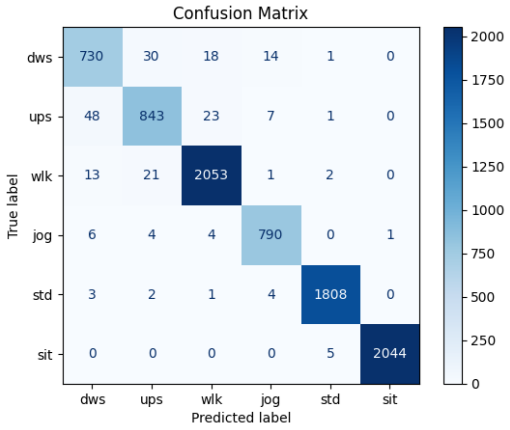


Figure 3.5: Confusion Matrix- Test Data

- 'ups' was sometimes misclassified as 'dws' (48 instances) and 'wlk' (23 instances).
- The misclassifications of upstairs and downstairs may be due to the similarity of the event
- Other activities showed high correct classification rates.

Increasing epochs from 10 to 20 with a window size of 50 did not yield a noticeable improvement in overall test or validation accuracy (both remained at 0.97). The loss values were slightly higher for the 20-epoch model. The difference between the 2 is not recognizable as they both identify an activity with 0.97 accuracy.

3.0.2 Experiment 2: Effect of Increased Window Size(Stride 200)

The window size was then increased to explore its impact on performance, keeping the stride at 200.

1. Model with Window Size 400, Stride 200 (20 Epochs):

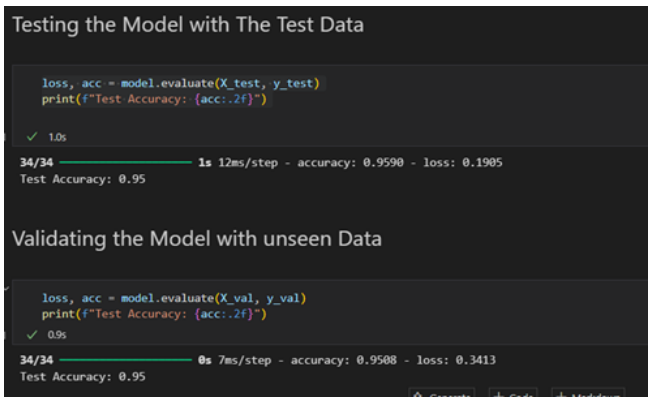


Figure 3.6: Window Size 400, Stride 200 Training Performance.

- Final training accuracy: 95.90% (Window Size 400: accuracy = 0.9590, loss = 0.1905).
- Final validation accuracy: 95.08% (val_accuracy = 0.9508, val_loss = 0.3413)

Training and validation accuracy and loss curves

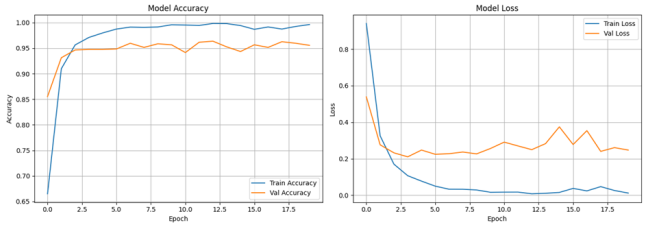


Figure 3.7: Training and validation accuracy and loss curves- Window Size 400

Training accuracy starts lower (around 0.65) and steadily increases, plateauing around 0.99-1.00. Validation accuracy starts around 0.85, increases to about 0.96, and it then shows some fluctuation, ending around 0.95. While the training loss starts very high (around 1.2) and decreases sharply, then gradually towards near zero. Validation loss decreases from around 0.55 to about 0.2, then fluctuates, showing some spikes.

Classification report:

Classification Report:

	precision	recall	f1-score	support
dws	0.85	0.91	0.88	94
ups	0.88	0.79	0.83	116
wlk	0.94	0.96	0.95	285
jog	0.99	0.95	0.97	107
std	0.97	0.99	0.98	211
sit	1.00	0.99	0.99	247
accuracy			0.95	1060
macro avg	0.94	0.93	0.94	1060
weighted avg	0.95	0.95	0.95	1060

Figure 3.8: Classification Report- Test Data

The overall accuracy for all activities is 0.95. 'ups' activity shows a notable drop in recall (0.79).

Confusion Matrix (Test Data)

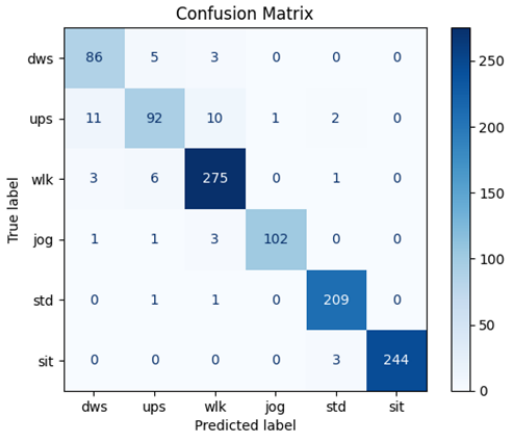


Figure 3.9: Confusion Matrix- Test Data

- 'dws': 86 correct, misclassified as 'ups' (5), 'wlk' (3).
- 'ups': 92 correct, misclassified as 'dws' (11), 'wlk' (10).
- Other activities generally well-classified.

2. Model with Window Size 1000, Stride 200 (20 Epochs):

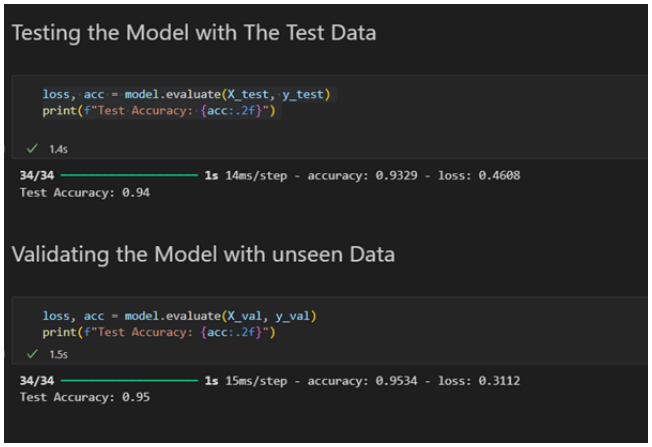


Figure 3.10: Window Size 1000, Stride 200 Training Performance.

- Final training accuracy: 93.29% (Window Size 400: accuracy = 0.9329, loss = 0.4608).
- Final validation accuracy: 95.34% (val_accuracy = 0.9534, val_loss = 0.3112)

With training accuracy approaching 100% and validation accuracy stabilising at 0.95 to 0.96, the accuracy and loss curves resemble the Window Size 400 model in shape. While validation loss first declines and then fluctuates, training loss sharply declines.

Classification report:

The overall accuracy for all activities is 0.94. 'ups' recall remains lower (0.80). 'wlk' precision and 'jog' recall also show a slight decrease compared to smaller window sizes.

Confusion Matrix (Test Data)

- 'dws': 95 correct, misclassified as 'ups' (6), 'wlk' (4).
- 'ups': 98 correct, misclassified as 'dws' (7), 'wlk' (17).
- 'wlk' shows some misclassification with 'ups' (4) and 'jog' (5).

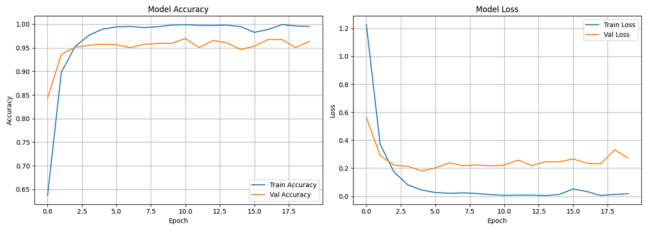


Figure 3.11: Training and validation accuracy and loss curves- Window Size 1000

Classification Report:

	precision	recall	f1-score	support
dws	0.93	0.90	0.92	105
ups	0.91	0.80	0.85	123
wlk	0.88	0.96	0.92	269
jog	0.96	0.89	0.92	121
std	0.99	1.00	0.99	205
sit	1.00	1.00	1.00	236
accuracy			0.94	1059
macro avg	0.94	0.93	0.93	1059
weighted avg	0.95	0.94	0.94	1059

Figure 3.12: Classification Report- Window size 1000

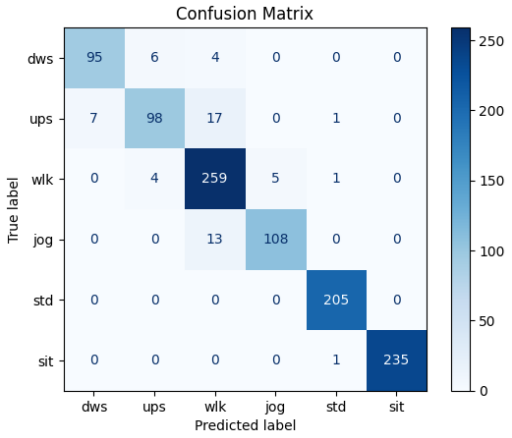


Figure 3.13: Confusion Matrix- Test Data

The larger window size (1000) further increases the temporal context, which may benefit activities with longer durations or repetitive patterns. This is evident as there is no significant improvement over smaller windows (lower testing and validating accuracies).

3.0.3 Experiment 3: Model Using FFT Output

This experiment evaluated the model using Fast Fourier Transform (FFT) features.

Testing the Model with The Test Data

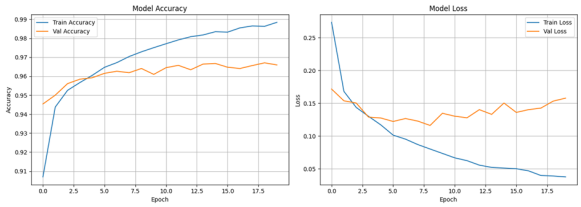
```
# Evaluate the trained model
loss, acc = model.evaluate(x_test, y_test, verbose=0)
print(f"Test Loss: {loss:.4f}, Test Accuracy: {acc:.4f}")
```

✓ 1.9s

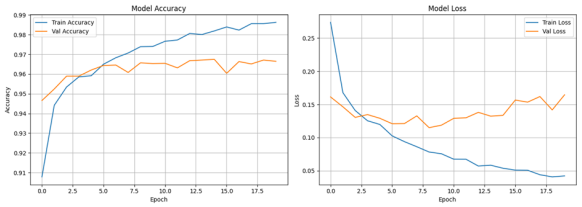
```
Test Loss: 0.1449, Test Accuracy: 0.9626
```

Figure 3.14: The FFT-based model Training Performance.

- Test accuracy: 96.26% : accuracy = 0.9626, loss = 0.1449).



(a) FFT-based model



(b) Non FFT-based model

Figure 3.15: Comparison of FFT-based and non-FFT-based models Training and validation accuracy and loss curves

Training accuracy rises beyond 98% across epochs in both figures 3.15. Strong generalisation is indicated by the validation accuracy stagnating at <97% with just slight variations. Loss curves indicate some overfitting since training loss progressively declines while validation loss slightly increases above 0.15.

Classification report:

```
Classification Report:
              precision    recall  f1-score   support

         dws       0.84      0.89      0.87      1054
         ups       0.91      0.86      0.88      1258
         wlk       0.96      0.97      0.97      2755
         jog       0.99      0.97      0.98      1074
         std       0.99      1.00      0.99      2452
         sit       1.00      1.00      1.00      2710

    accuracy                           0.96     11303
   macro avg       0.95      0.95      0.95     11303
weighted avg       0.96      0.96      0.96     11303
```

Figure 3.16: Classification Report- Test Data

The overall accuracy for all activities is 0.96. The FFT model shows good overall performance, though 'dws' and 'ups' precision/recall are slightly lower than the best time-domain model.

Confusion Matrix (Test Data)

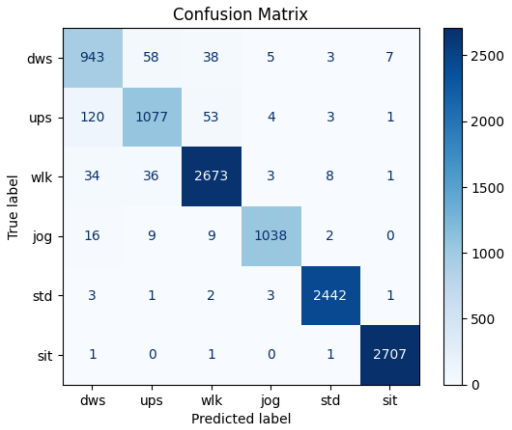


Figure 3.17: Confusion Matrix- Test Data

- 'dws': 943 correct, misclassified as 'ups' (58), 'wlk' (38), 'sit' (7)
- 'ups': 1077 correct, misclassified as 'dws' (120), 'wlk' (53).
- 'wlk': 2673 correct, misclassified as 'dws' (34), 'ups' (36).
- Misclassifications between 'dws', 'ups', and 'wlk' are the most prominent.

The models demonstrated high efficacy in classifying six distinct human activities (downstairs, upstairs, walking, jogging, standing, and sitting) based on the iPhones' sensor data, achieving test accuracies consistently in the range of 95-100% and robust F1-scores, generally between 0.92 and 1.00. The high accuracy achieved across different subjects within the test set indicates good generalisation capabilities of the model. Increasing the window size to 400 or 1000 samples, or extending training beyond 10-20 epochs, provided diminishing returns and, in some cases, slightly reduced performance or increased computational load without significant accuracy gains.

Activities 'std' (standing) and 'sit' (sitting) were consistently classified with very high accuracy across almost all model variations. A slight, yet persistent, challenge was observed in distinguishing between upstairs and downstairs movements, which showed a higher tendency for mutual misclassification. This is likely due to the inherent similarities in their motion patterns(overlapping motion patterns). Time-domain windowing and Fast Fourier Transform (FFT) preprocessing yielded strong results. While FFT-based models performed well (around 96% accuracy), the simpler time-domain windowing approach (specifically with a 50-sample window) was often equally, if not slightly more effective. This indicates that the raw temporal patterns in the sensor data are highly informative and can be effectively captured by the CNN architecture without necessarily requiring transformation to the frequency domain for this specific task. The model with a window size of 50 and trained for 10 epochs appears to be the most robust and efficient among the tested configurations, achieving high accuracy with relatively stable training

Chapter 4

Conclusions

In this project we successfully created 2 models that use three-way holdouts and preprocess data by the fft method and the window method. Both data sets had a high levels of accuracy with the 1000 size window dataset achieving the lowest accuracy rating. This may be due to having too much information in each chunks of data which could potentially throw off the model in its prediction capabilities.

The results show that the model will perform best at lower window sizes this is in both methods of preprocessing. This may be due to the amount of information that is used to classify and test the mode. When the window size is small we will have multiple chunks to train the model on so that test data can be better suited for the model and produce better results.

The output of the confusion matrices of all the models that were tested showed that there was confusion between the downstairs and upstairs classifier. This may be due to the similarity between the movements. This could be made better when more robust methods are used in the data pre-processing stage of the model.

Improvements that could be made to this model are potentially reducing the number of features. There is a redundancy issue with using rotationRate(x,y,z) and attitude(roll , pitch , yaw) . For example the model misclassifies downstairs and upstairs motions since they have similar motion the only usefull information would possibly be userAcceleration and one of the other two. Therefore in further iterations of this model a reduction of features could help improve the accuracy of this model.

Bibliography

[1] A. K. M. Masum, E. H. Bahadur, A. Shan-A-Alahi, M. A. Uz Zaman Chowdhury, M. R. Uddin, and A. Al Noman, "Human Activity Recognition Using Accelerometer, Gyroscope and Magnetometer Sensors: Deep Neural Network Approaches," in *2019 10th International Conference on Computing, Communication and Networking Technologies (ICCCNT)*. IEEE, 7 2019. [Online]. Available: http://dx.doi.org/10.1109/icccnt45670.2019.8944512

[2] D. K. Zainab and R. R. Srivastava, "A SMARTPHONE SENSOR BASED REAL-LIFE HUMAN ACTIVITY RECOGNITION SYSTEM USING DEEP NETS METHOD," in *International Journal of Innovative Research in Computer Science and Technology (IJIRCST)*. Innovative Research Publication, mar 15 2024, pp. 18–25. [Online]. Available: http://dx.doi.org/10.55524/csistw.2024.12.1.4

[3] A. Saeed, T. Ozcelebi, and J. Lukkien, "Multi-task Self-Supervised Learning for Human Activity Detection," *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.*, vol. 3, no. 2, pp. 1–30, 2019.

[4] W. Wu, S. Dasgupta, E. E. Ramirez, C. Peterson, and G. J. Norman, "Classification Accuracies of Physical Activities Using Smartphone Motion Sensors," *Journal of Medical Internet Research*, vol. 14, no. 5, p. e130, oct 5 2012. [Online]. Available: http://dx.doi.org/10.2196/jmir.2208

[5] A. Nouriani, R. McGovern, and R. Rajamani, "Activity recognition using a combination of high gain observer and deep learning computer vision algorithms," *Intelligent Systems with Applications*, vol. 18, p. 200213, 2023.

[6] M. Malekzadeh, R. Clegg, A. Cavallaro, and H. Haddadi, "DANA: Dimension-Adaptive Neural Architecture for Multivariate Sensor Data," *arXiv preprint arXiv:2008.02397*, 2021.

[7] M. Malekzadeh, "MotionSense: An open-source human activity recognition dataset using smartphones," https://github.com/mmalekzadeh/motion-sense, 2018, accessed: 2025-05-29.