Martial Arts Kick Detector

Coursework 2 Final Report

Sidney Hughes
Student ID: 20266264

Computer Science
University of Nottingham
United Kingdom

May 2024

Contents

1. *Summary*

The main purpose of the ubicomp system is to be able to detect different kicks performed on a heavy bag. The system attaches the the lower part of the leg and uses an accelerometer and gyro to detect movement. As a fully developed system users would be able to wear a small sensor on each of their legs whilst doing a workout, automatically recording all of their kicking movements to track their training.

2. *Background and Motivation*

Gyms often have heavy bags in the gym which can be used for training; training on these consists of throwing different combinations of punches and kicks, it makes for an excellent cardio workout. Unlike other gym equipment in which you can record for example, the weight and number of reps and sets on machines and free weights, to record a workout on a heavy bag you might just time how long you are doing it for. This ubicomp product would allow you to accurately record all of the different movements that you are making so that you can track the workout and see just how many different kicks or even punches you have thrown in the duration of the workout. It could also record the amount of time you are active or inactive during the workout. Being able to track these things in detail could be very useful for someone working out as you could set targets for how many of a certain kick you would like to throw, you could see if you are doing a certain movement too often etc.

3. *Related Work*

3.1 Review of similar products

3.1.1 Fitness trackers

Fitness trackers normally come in the form of a watch, some examples of these would be the Apple watch or the Fitbit. Capabilities of these watches vary, looking in more depth at the Fitbit Charge 6[2] the main measurements that it uses to track your fitness are monitoring heart rate and distance travelled. It uses this data to track exercises such as cycling, running, swimming etc. the tracking however is quite restricted to the amount of calories you have burned and how far you have travelled. The ubicomp product I am developing would be a much more specific type of fitness tracker detecting specific movements, meaning its use cases are far fewer than a general fitness tracker but it will be able to give you an in depth analysis of excercising with a heavybag.

3.1.2 Impact wrap

Impact Wrap[3] is a home training product that is attached to a heavy bag. The product uses sensors to measure impacts on the bag, it can measure different strengths, frequencies and counts of impacts. The product seems to be moderately popular with over ten thousand downloads on the google play store and a high review score of *4.8* stars. This shows that there is certainly a market for this type of product. My ubicomp product is targeted at the same group however it offers slightly different functionality, instead of tracking impacts it tracks the type of movement used and could track the frequency and count of those movements.

4. *Design*

4.1 Overview

The ubicomp device would be worn at the end of a limb such as on the wrist for the arms or on the calf/ankle for legs.
The positioning of the device is important as firstly, you want it to be in a position that would not obstruct your movement and that would be comfortable. Secondly positioning towards the end of your limbs would mean that they are experiencing a larger range of motion and therefore may collect better data.

4.2 Use cases

- Training in a gym
- Training at home
- Analysing which kicks you are doing
- Counting the number of kicks
- Monitoring your workout
- Analysing trends in what you are doing
- Could set up combinations and then track if you are hitting those combinations

4.3 Technical Design

- Using the Raspberry PI with Grove PI - Attaching an Accelerometer and Gyro - Sensor data is collected and processed - Passed to a ML algorithm that predicts the current movement

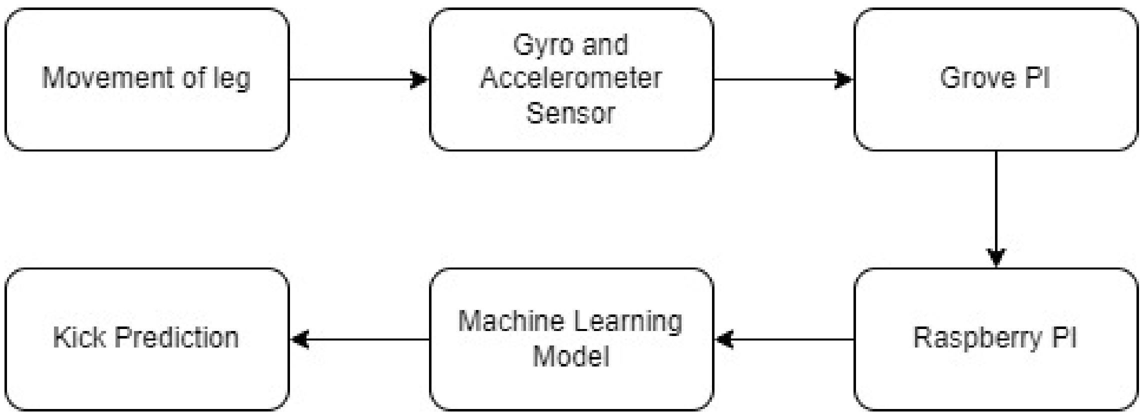


Figure 4.1: System Diagram for Kick Detector

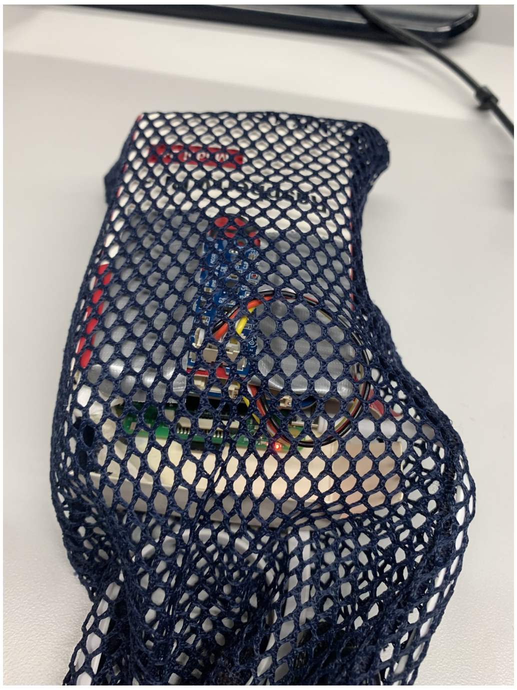
5. *Implementation*

5.1 Prototype

- Prototype uses the Raspberry PI with Grove PI and has the accelerometer and gyro sensor attatched to a ic2 pin - Construction of the prototype was putting the grove pi into a box with the sensor taped ontop of the box, this was so that the sensor was not touching the board as this caused interference and stopped it from working - the box and a portable charger were placed into a bag and strapped to the upper part of my right calf at the back - due to the device being to large to have at the lower part

(a) $y = x$

(b) $y = 3\sin x$



(c) $y = 5/x$

(d) $y = 5/x$

Figure 5.1: Four simple graphs

5.2 Data collection

The data I collected was of three different types of kicks, a front kick, a mid roundhouse kick and a low roundhouse kick.
Insert diagrams The kicks were performed on a heavy bag.

To collect the data I tried two different methods.

5.2.1 Method 1

Data collection file *kickSensorDataCollection.py*
For this method, kicks were performed one at a time, after a key press I would perform a kick this would be recorded as a 5 second window and written to a csv file.
I recorded 25 samples of each kick.
Data recorded:
- **Accelerometer** - x, y, z, magnitude
- **Gyro** - x, y, z

5.2.2 Method 2

Data collection file *kickSensorDataCollectionII.py*
For this method, kicks were performed consecutively, with a script of kicks to perform I would perform each kick and record the time after each kick.
I recorded four times, each time performing 5 of each kick.
Data recorded:
- **Accelerometer** - x, y, z, magnitude
- **Gyro** - x, y, z, magnitude
- **Time** - (current time - start time)

5.3 Machine Learning Model

5.3.1 Method 1

Preprocessing

Libararies used - *pandas*, *numpy*
To process the data for input to the model I converted the rows into new columns so that each kick would be one row within a dataframe and I appended the kick label in the final column.
I normalized the data using z-normalization.

Model

Libraries used - *tensorflow*, *sklearn*
Model Structure:

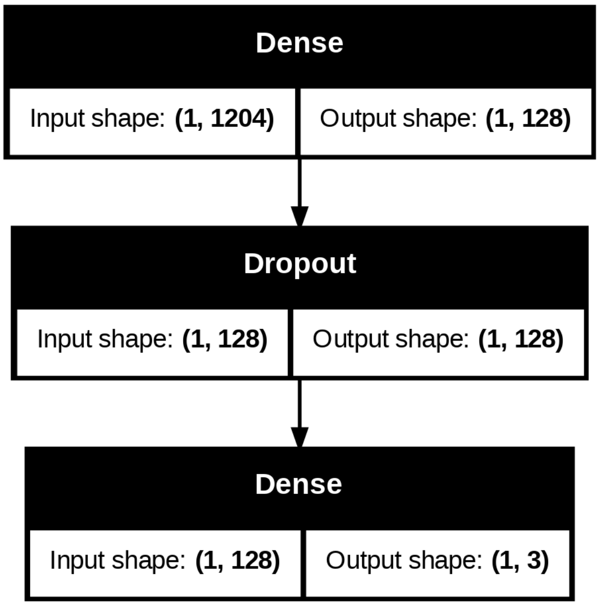


Figure 5.2: Model 1 Structure

Hyperparameters:

Neurons in hidden layer	128
Dropout	0.5
Optimizer	Adam
Epochs	10

5.3.2 Method 2

Preprocessing

Libararies used - *pandas*, *numpy*
For all data collected I created labels by using the time stamps I recorded and setting the rows three seconds before and after to the kick performed at the time stamp.
To process the data for input to the model, three of the continuous recordings were concatenated together to be used as the training/validation set. Z-normalization is applied to the features. The data is converted to a windowed format as it is time series data, the window size used is 300 and it is using a step size of 1, the code to roll data into windows was given by *Joe Marshall*.
The training data is then split into training and validation with a validation split of 0.2.

Model

Libraries used - *tensorflow*, *scikeras*, *sklearn* **Model Structure:**

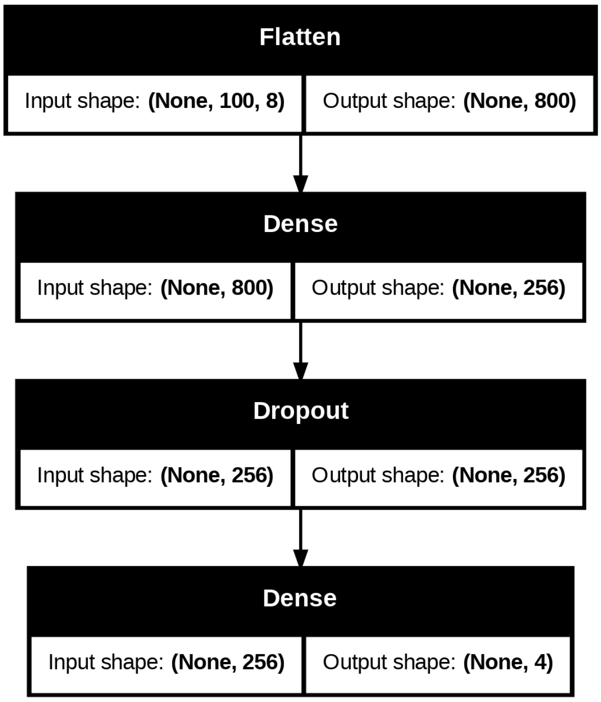


Figure 5.3: Model 2 Structure

Hyperparameter tuning: To perform hyperparameter tuning I followed a tutorial by Jason Brownlee[1], some code snippets are the same or similar to the ones used here.
When training for hyperparameter tuning I trained using only the validation to reduce the time needed.

Hyperparameter	Optimal value
Optimization algorithm	Nadam
Learning rate	0.0001
Beta 1	0.9
Beta 2	0.999
Network weight initialization	glorot uniform
Activation functions	relu
Dropout	0.1
Model weight constraint	4.0
Neurons in hidden layer	64
Epochs	50
Batch size	25

6. *Testing*

6.1 ML Model Performance

Model evaluation performance metrics:
- Accuracy score
- F1 score
- Confusion matrix

6.1.1 Model 1 testing

Model 1 training and testing results:
To train and test I used *sklearn StratifiedKFold* with 5 folds to get an accurate view of model performance.
Results
Model Accuracy

Mean Accuracy	0.8029
Mean Loss	0.6027

F1 Scores

Front Kick	0.9846
Low Roundhouse	0.6118
Mid Roundhouse	0.7628

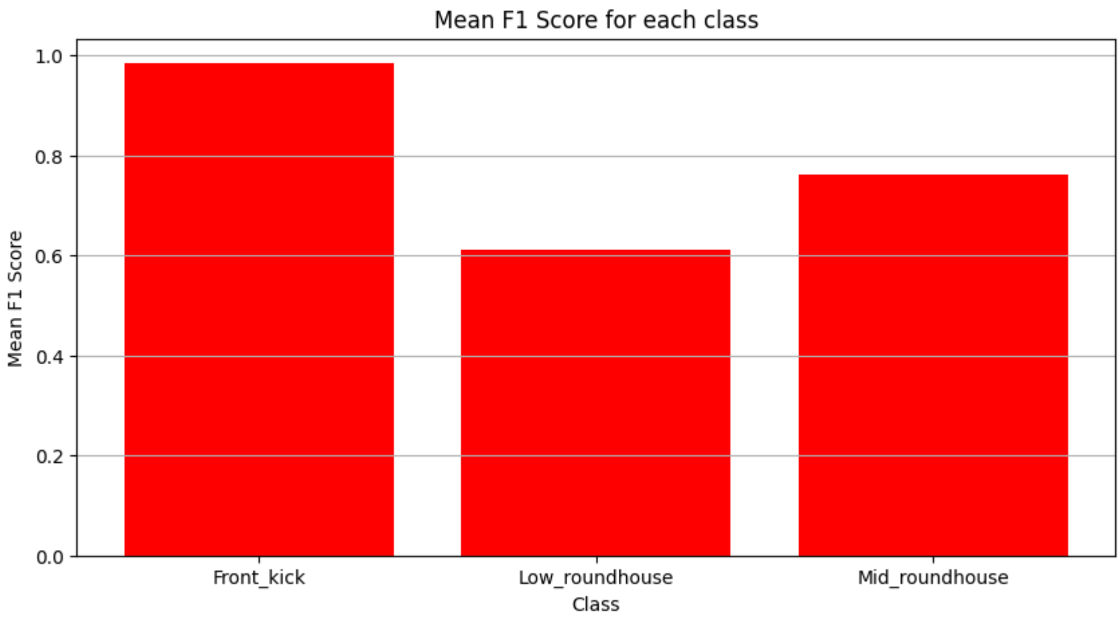


Figure 6.1: Model 1 F1 Scores

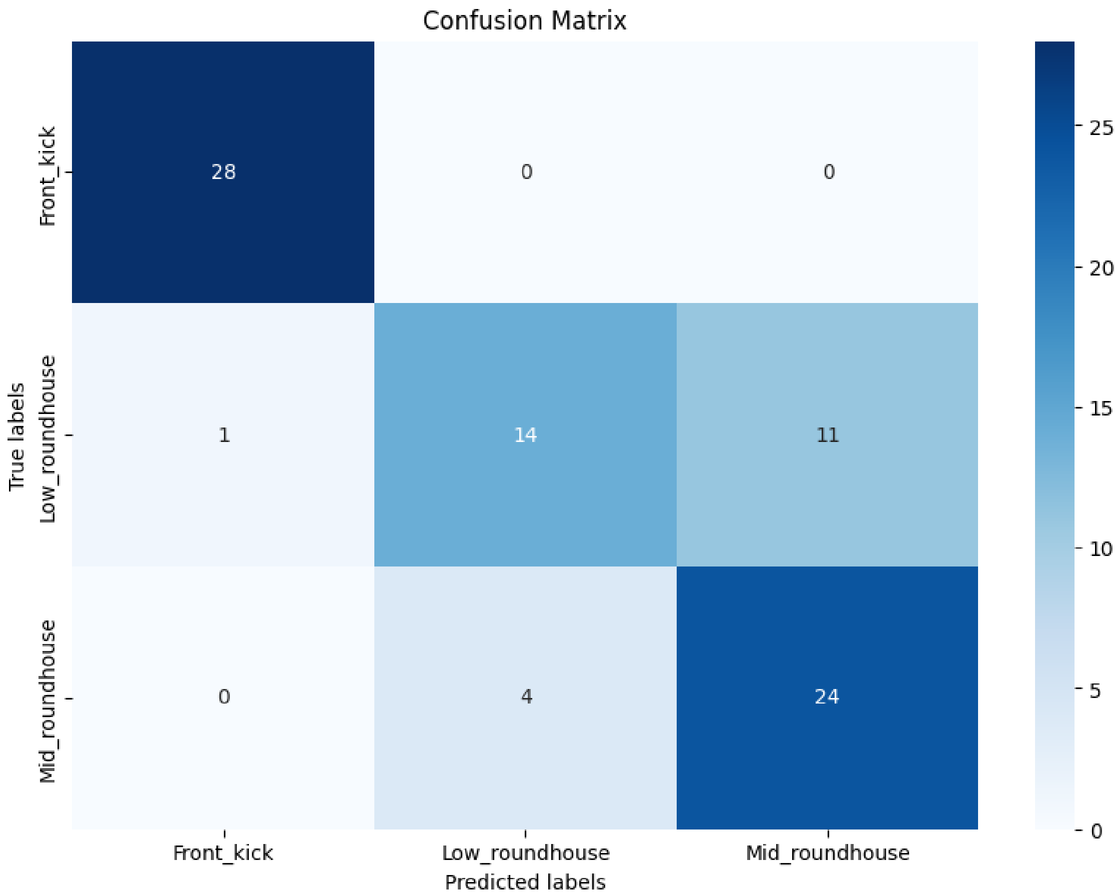


Figure 6.2: Model 1 Confusion Matrix

6.2 Model 2 testing

Training model 2 I did not use K fold cross validation as I did not want it to affect the data in a bad way by mixing up the order.

Results

Model Accuracy

Accuracy	0.6225
Loss	1.6244

F1 Scores

No Kick	0.7549
Front Kick	0.7108
Low Roundhouse	0.5356
Mid Roundhouse	0.2266

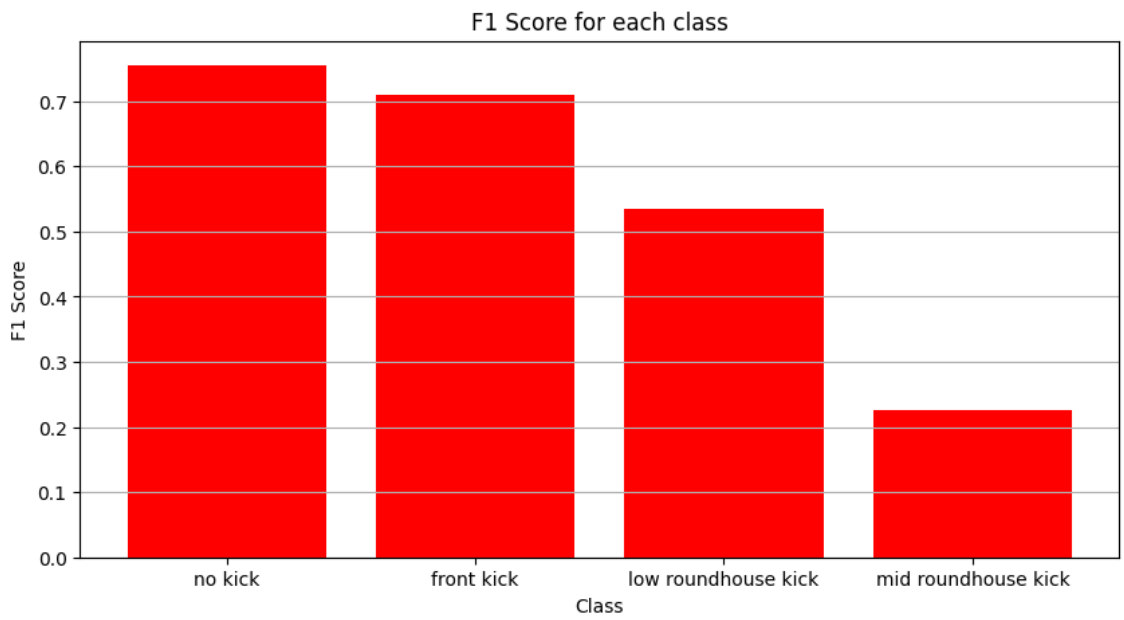


Figure 6.3: Model 2 F1 Scores

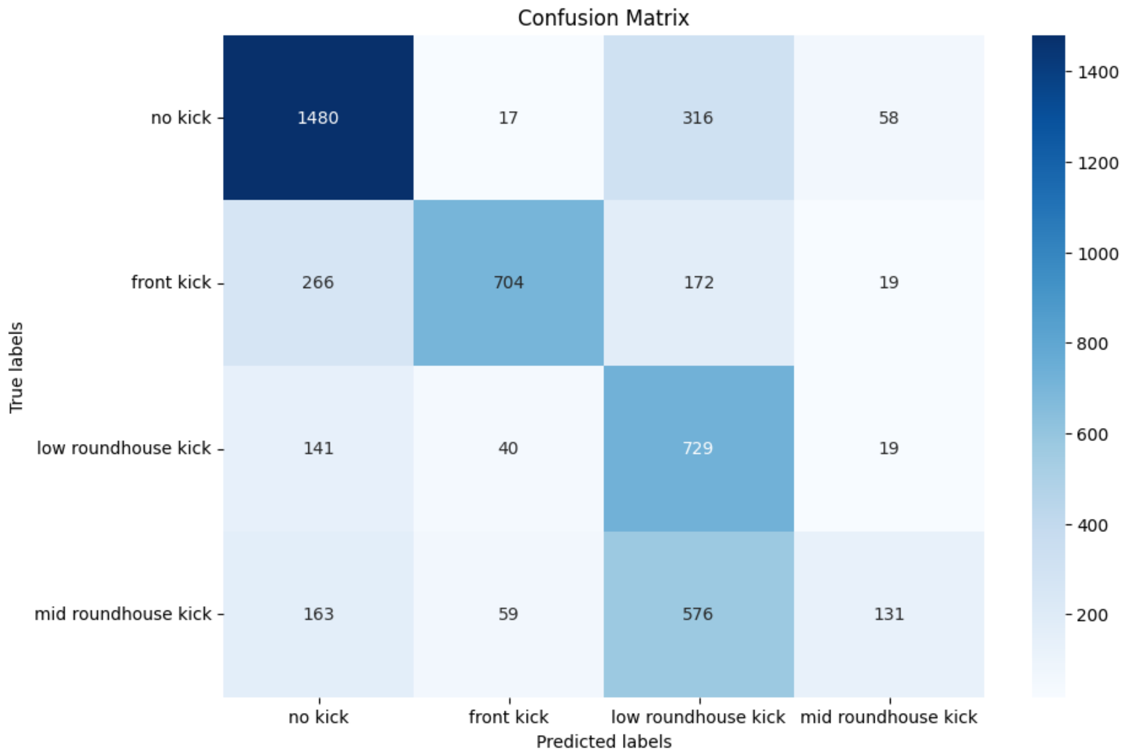


Figure 6.4: Model 2 Confusion Matrix

6.3 Summary of Model Performance

Model 1 performance is quite good with a high overall accuracy of 0.8029, the F1 score for front kick is also excellent with a score of 0.9846. The other two kicks scored much lower F1 scores with 0.6118 and 0.7628. The reason behind these results being lower can be seen in the confusion matrix, as is shown *42%* of the low roundhouse is predicted as being the mid roundhouse and *14%* of the mid roundhouse is predicted as low roundhouse. The cause of this is likely due to the similarity of the kick the model had a hard time differentiating between the two.

Model 2 performance is quite poor overall with an accuracy of 0.6225, the F1 scores for *No Kick* and *Front kick* are decent with scores of 0.7549 and 0.7108 respectively. The reason behind these scores being higher than the other two is likely similar to the reason stated for model 1. The lower overall accuracy of this model is likely due to there being more noise in the dataset as it was recorded continuously there are likely more unintentional movements that are being recorded and wrongly labelled and predicted.

Overall Model 1 has a much better performance, but it is important to note that the difference in ways of recording is playing a large role in the drop in performance for model 2. Recording continuous data may be producing worse results for the machine learning model although the continuous data may be more effective when using with the prototype as it allows for a more fluid experience.

6.4 Testing Models with the Prototype

6.4.1 Model 2

When testing model 2 with the prototype I conducted a series of 15 kicks continuously and used tflite to predict the current kick being performed. The kicks were five of each kick in order, so five front, then five mid and five low.
Results from testing:

```
no kick
no kick
no kick
low roundhouse kick
no kick
front kick
no kick
front kick
no kick
mid roundhouse kick
low roundhouse kick
no kick
mid roundhouse kick
no kick
mid roundhouse kick
no kick
mid roundhouse kick
no kick
front kick
no kick
low roundhouse kick
no kick
no kick
no kick
front kick
no kick
no kick
no kick
```

Figure 6.5: Model 2 Testing Result

6.4.2 Summary

The model did have some correct predictions, predicting two of the front kicks and most of the mid roundhouse kicks. The results however were disappointing with many false predictions.

7. *Critical Reflection*

To conclude the prototype did show some promise with the ability to predict some kicks. The design of the prototype itself was very simple however it did do the job effectively. More experimentation could be done with the positioning of the device as this could change the performance. Creating a smaller device for future use and testing would also be better for the ubicomp product as it is quite cumbersome.
Some testing was done with the first model however due to it not being continuous it meant that you would have to time the kicks to be exactly within the recording window. Having to kick within a specific time frame did not really fit within what I wanted the prototype to be able to do and so the continuous data model, although worse performing is the model I would progress with. One major weakness that contributed to poor performance was a lack of data to train the model on and also a lack of variety of data. If more data could be collected and data from different movements as well such as other types of kicks and also general movements such as walking this could significantly improve the prototype performance resulting in less false predictions. Another idea that could be tested is using sensors on both legs, although kicking is mostly performed using one leg, different kicks can involve different movements from the other leg which could help to distinguish between kicks.
To conclude after testing with both continuous and non continuous data, continuous is the better solution to realising the product. Increasing dataset size with the same kicks and also introducing other movements as well as adding an additional sensor to another leg are the best ways to move forward with the product.

Bibliography

[1] Jason Brownlee. *How to Grid Search Hyperparameters for Deep Learning Models in Python with Keras*. URL: https://machinelearningmastery.com/grid-search-hyperparameters-deep-learning-models-python-keras/.

[2] Fitbit website. *Features of Fitbit Charge 6*. URL: https://www.fitbit.com/global/us/products/trackers/charge6#:~:text=Keep%20up%20with%20your%20body,in%20target%20heart%20rate%20zones.&text=Want%20to%20know%20more%3F,body%20is%20responding%20to%20stress..

[3] Impact Wrap website. *Impact Wrap Homepage*. URL: https://impactwrap.com/for-home/.

8. *Appendix A*

8.1 Setup Prototype

The prototype just consists of the Accelerometer and Gyro sensor attached to an IC2 port. Once attached, place in a box as shown in the images within Implementation. Place box and a portable charger in a bag and tape to the upper part of your right leg calf.
Running the prototype.
Use the file *Kick_Detector_Inference.py*; you will have to install tflite. Once running it constantly reads the data and outputs a prediction after every 300 samples.