Imperial College London

ME4 Individual Project

Imperial College London

Department of Mechanical Engineering

Biomechanical Monitoring and Machine Learning for Human Motion Analysis in Cricket

Final Report

Author:

Daniel Anderton (CID: 01505697)

Supervisor:

Dr. Ravi Vaidyanathan

Date: 9th June, 2022

Word Count: 11858

Word Limit: 12000

Abstract

Evaluating the motion of an athlete is of great importance as it can result in the enhancement of performance during matches and training sessions. Additionally, the potential of developing an inexpensive, accessible and quality analysis tool has substantial consequences for the amateur cricket community worldwide. The goal of this project was to develop a sensor based framework to classify cricket batting shots, along with other performance metrics in real-time. Using gyroscope and accelerometer sensors, a gaussian SVM algorithm was optimised using Scikit learn to classify 5 broad batting strokes; defensive, drive, cut, pull and sweep, using a variety of players and a total of 538 shots collected prior to the data analysis. This classification algorithm has a overall accuracy of 86.8% (93.5% for detection and 92.8% for classification).

A real time application was created using the iPhone / Apple Watch ecosystem to classify each shot and display performance statistics of bat speed, shot quality and bat angles to the user in real time. The application classification was carried out initially using a neural network (NN) algorithm with the optimisation done by Apple using their Activity Classifier designed for sensory motion data. The gaussian SVM algorithm was then also integrated into the app, giving two different real time algorithms to compare. The real time algorithms both had the same detection process and were tested using another 550 shots and 140 shots respectively. The detection accuracy was 90.0% with the NN resulting in an overall accuracy of 68.6% (76.2% for classification) and the gaussian SVM with an overall accuracy of 79.1% (87.9% for classification). This application is now able to provide the basis for objective skill assessment, focusing on personal improvements for different players. Adding more classes to the algorithm, generalising for a wider range of players and inferring performance improvements from the data collected are all areas that will improve the classification and develop this project further.

Acknowledgements

I would like to express my appreciation to Dr. Ravi Vaidyanathan for his support and guidance throughout this project.

I would also like to thank Nathan Steadman for his general advice on all aspects of the project, particularly in assistance to resolve programming issues.

Lastly I would like to thank my friends and family, who have always shown their support throughout my project and my degree.

Table of Contents

1 Introduction

With the recent explosion of data analytics in the professional level of many sporting contexts, the real time statistics of a professional cricket match are now able to run advanced predictive models to calculate anything from a side's win percentage at a particular stage in a game, to the number of expected wickets given the positioning of the deliveries bowled in the match [1]. These models however, are using a large amount of input data from a variety of sources, making them both computationally and economically expensive for the amateur game.

Whilst this makes the game more appealing for the spectator, it does little to assist the 250 million amateur players that play the game worldwide, [2]. This project will aim to use inertial sensors from an inertial measurement unit (IMU) within an Apple watch, to discover the potential of a quality analysis tool for the amateur cricketer, which would have substantial benefits for the entire cricket community. The tool will focus on evaluating statistics on an individual basis to provide a base for objective skill assessment, acting as a interactive coaching tool. Each cricket shot will be classified using variuos mahcine learning algorithms, with several other performance metrics evaluated.

The motivation behind the project stems from a passion for the game of cricket, as well as coaching experience within the sport, gaining first hand experiences for the need for such a device. This project has continued the work at the biomechatronics laboratory on sports wearable technology - adapting the existing work on tennis motion tracking [3], and applying it to work for a batsmen in cricket.

1.1 Objectives

The objectives for the project are summarised below.

1. Use and expand on the knowledge of kinematics of cricket batting strokes.
2. Collect sensor data and apply Machine Learning and Deep Learning algorithms for stroke detection and classification.
3. Evaluate how to use this data to calculate other performance measures such as bat speed, shot quality and bat angles.
4. Expand the system's functionality to make it work in real time on a smart watch and mobile application.
5. Expand the app's capabilities and user interface (UI) such that users can simulate a match scenario real-time during a training session, as well as access their statistics.

1.2 Report Structure and Methodology

As highlighted in the project objectives, the initial goal of the project was to prove that the concept of a coaching tool, using solely sensory motion data, was possible by devising a thorough framework, determining what could / could not be adapted from previous work.

The project then entailed going back to each stage of the devised framework and converting the method into working in real time. The structure of this report summarises each of the stages for this framework and discusses the post processed analysis and the real time analysis in conjunction with one another. The report is therefore not structured in chronological order.

The report is structured with Section 2 explaining the basic kinematics of cricket motions, fundamental classification techniques and relevant previous work done. Section 3 summarises the data collection methods - an important phrasing introduced in this section is the difference between *Phase 1 data* and *Phase 2 data*.

- *Phase 1 data*: Data collected using third party apps and no real time analysis
- *Phase 2 data*: Data collected using an app developed for the project, allowing for with real time analysis

Sections 4-6, Detection & Segmentation, Classification and Performance Metrics, describe the methodology for each and subsequently present the results. Finally, Sections 7 and 8 discuss the findings, summarise future work required and ultimately conclude the project.

2 Background and Literature Study

This section will briefly explain the game of cricket and introduce the kinematics of various cricket shots. Additionally, the two machine learning algorithms used will be introduced and a summary of the findings from similar research will be outlined.

2.1 Inertial Measurement Units

The core of this project involves the use of an Inertial Measurement Unit (IMU). The IMU used for the project is the one embedded in an Apple Watch, with Figure 1 showing the orientation of the 9-axis IMU within the watch. A 9-axis IMU is composed of a 3-axis accelerometer, a 3-axis gyroscope and a 3-axis magnetometer, although the magnetometer will be ignored as it is not useful for this project.

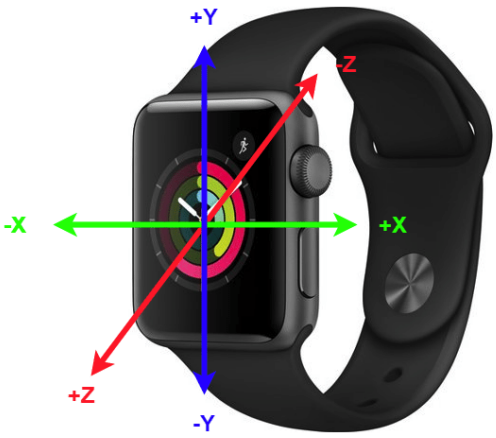

Figure 1: *Apple Watch Inertial Sensors [4]*

This IMU was used due to the easy integration into a real time app, as discussed in Section 2.5. An IMU with a ±8G range could saturate the data as the inertial measurements are expected to exceed 8G [3]. The ±16G range of the inertial sensors within the Apple Watch was therefore another reason why the method of data collection was chosen. Lastly, the orientation shown in Figure 1 means that for left handed players, the axis points towards the fingers, whereas for right handed players, it points up towards the elbow.

2.2 The Game of Cricket

Cricket is a team sport, with two teams of 11 players, involving a bat, with the objective to score more runs than the opposing team [5]. A batters role in a game of cricket is to score runs whilst simultaneously keeping their wicket by not getting out. Depending on the delivery type from the bowler, where the ball pitches, and the placement of the fielders, a batsmen has to decide the type of shot to play.

Evaluating relevant performance metrics from the inertial sensor data is key to assisting a coach infer actionable improvements for the batter to improve technique. A batters technique is comprised of numerous aspects and these suggested improvements can be inferred referring to the batters stance and the movement of their wrists, head & body.

2.2.1 Types of Cricket Shots

There are a large number of cricket shots. Traditional shots such as drives, cuts, pulls, hooks, or sweeps are the core foundation of a batters technique, however, the recent development of the T20 format has brought about new shots entirely such as the switch hit and reverse sweep [6]. There are also many factors considered when selecting a shot such as the angle of approach and the ball speed [7]. Studying the kinematics of various shots will aid in selecting the easiest shot types to classify and hence provide the basis of the classification.

There is also a large overlap in both the area of the pitch that the different shots are aimed, and in the definitions of what motion classifies a particular shot. Due to this it was decided to choose an initial 5 shots that have the largest variance between them; Drive, Cut, Pull, Defensive and Sweep. The first three have been chosen based on bat rotation, the defensive stroke due to the lower accelerometer spikes on impact and the sweep shot to include two classes with similar swing paths (sweep & pull), that are hit to similar parts of the field.

Figure 2 shows the part of the pitch that each of the chosen classes will typically go to - with the exception of a defensive stroke that moves very little from the point of impact. The diagrams are displayed for a right handed batsmen.

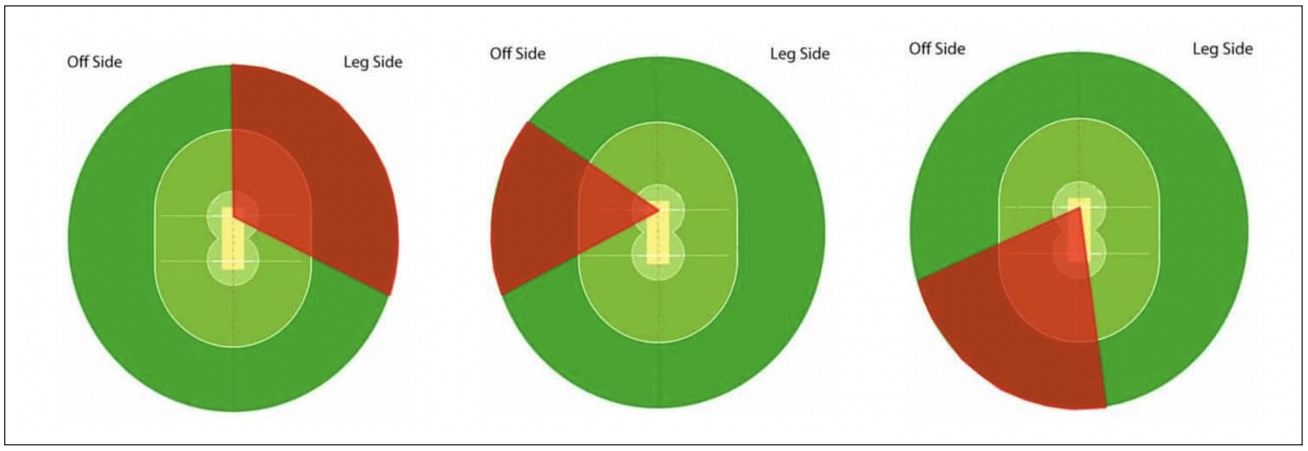

Figure 2: *Direction of a sweep / pull (left), a cut (center) and a drive (right) [6]*

2.2.2 Kinematics of Cricket Shots

A cricket shot can categorised into the following stages [8];

1. *Stance and Initiation*
2. *Back-Lift*
3. *Stride and downswing*
4. *Impact*
5. *Follow-through*

These general characteristics are valid for all cricket shots. During a typical shot, the angle of the bat relative to its forward motion is quite important, not just to maximise the contact point but also to direct the ball. Figure 2 shows the two sides of the ground, the ″off side″ and the ″leg side″. For example, when playing a drive shot on the ″off side″, the bat face will be angled (opened) towards this side through impact, to help guide the ball to exactly where the batter wants. Conversely, a ″leg side″ drive sees the bat face closed on impact for the same reason. As the bat face is opened or closed through impact, the component of the gyroscopic motion along the axis in the direction of the edges of the bat will vary [7], making shots such as a cut and pull shot, distinguishable on this basis. Cricket bat speeds, on average, are also lower than other racket sports such as tennis [9]. This is due to the more precise speed-accuracy trade off, where ′successful′ batting may necessitate sacrificing speed to enhance accuracy - for example in a defensive shot. This greater speed range will translate to a lower accelerometer spike that needs to be detected.

2.3 Machine Learning Classifiers

A classifier is the term given to a function or algorithm that implements classification [10]. Classifiers can also be split into supervised / unsupervised Learning. Supervised learning is when the class labels are known whereas unsupervised learning does not know this information and finds similarities within the dataset [11]. This is a supervised learning problem as the shots we want to classify are known. Two main types of supervised learning algorithms were used during the project - support vector machines (SVMs) and neural networks (NNs) - an explanation of the principles of each is described below.

2.3.1 Support Vector Machines

SVMs are based on mathematical functions and can analyse data for both classification and regression problems. The training data set provided to the SVM is separated by a hyperplane. The hyperplane is bounded in terms of the margin, which is the minimum distance

between a data point and the decision surface. A maximum margin hyperplane is chosen that splits the classes where the distance between support vectors and the hyperplane is maximised, as this minimises the risk of mis-classification, shown in Figure 3. SVMs are also less prone to over-fitting and they perform well for non-linearly separated data [11].

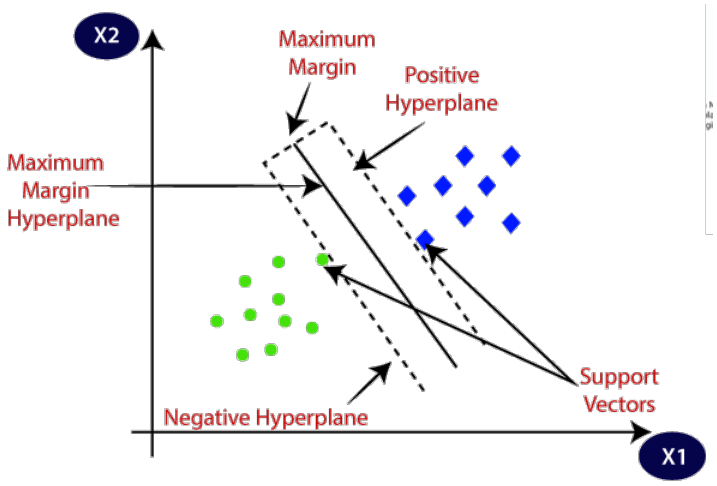

Figure 3: *SVM Hyperplane [12]*

2.3.2 Neural Networks

A feed forward neural network is a model that consists of an input layer, one or more hidden layers and an output layer. Within this, each node has an associated weight, $\Theta = \theta_1, \theta_2, ..., \theta_n$, and activation function, $F = f_1, f_2, ..., f_m$. If the output of the node is above its threshold value, it becomes activated. Figure 4 shows an example of a neural network with 2 hidden layers.

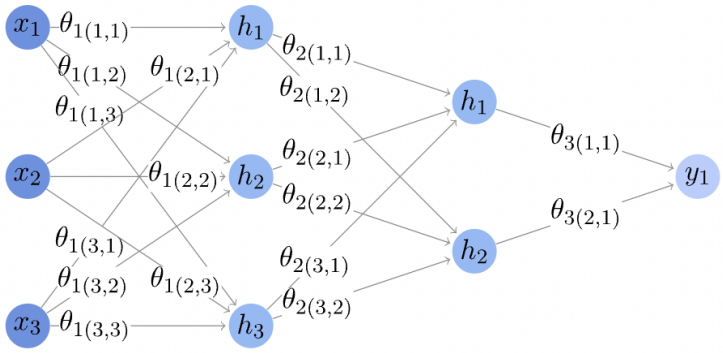

Figure 4: *Schematic of a Neural Network with 2 hidden layers [13]*

The weights will affect the forward propagation that goes through the network. However, they can then be changed during back propagation — where the neural network is now learning [14]. This process is iterative and is conducted on every piece of the training data.

2.4 Motion Tracking within Cricket

This section will summarise literature on motion tracking within cricket to help assist the project's direction. The location of the sensors for this project is the wrist of the dominant hand. Other commercially available products in the 'Smart Batting' space have chosen to place their sensors at different locations, as shown in Figure 5. The wrist was selected as a favourable position as it does not modify the player's equipment, something that was concluded to be an inconvenience.

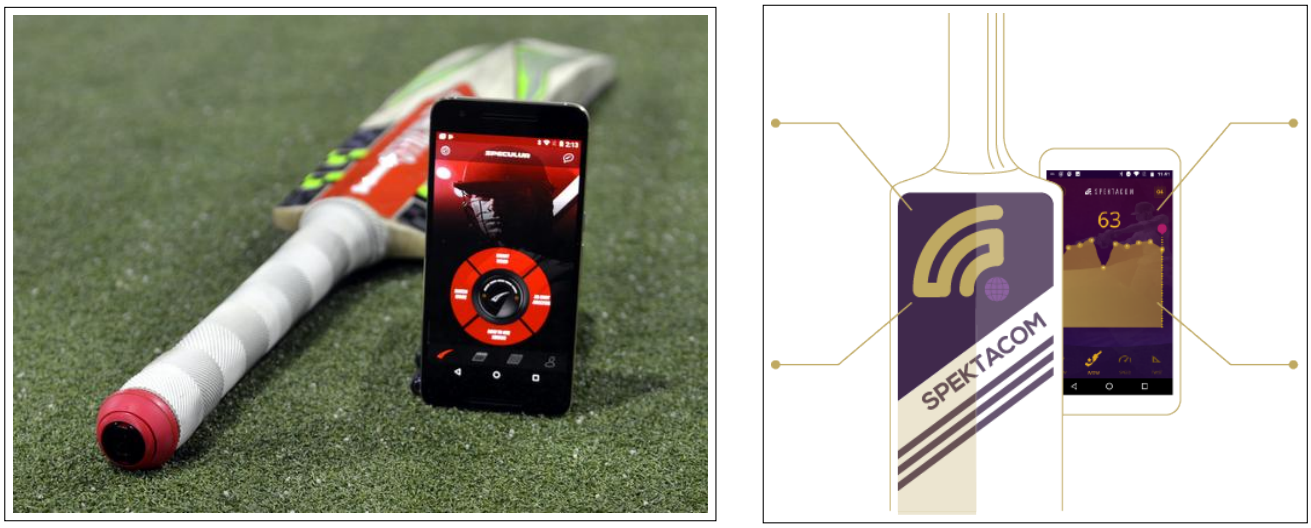

Figure 5: *Sensor located of the bat handle (left) [15] and on the bat itself (right) [16]*

Current algorithms used in literature for detailed motion analysis for a batsmen in cricket involve using inertial sensors in 4 different locations (one on each limb), to evaluate a hierarchical shot analysis system for 5 different shots [17]. A linear kernel SVM has also been used to classify a cut shot, a pull shot and a drive shot [18]. The authors in [19] used a CNN to identify 6 classes, a sweep shot, a cover drive, a straight drive, the pull shot and two obscure shots, a helicopter shot and the scoop shot, although this CNN was done via image recognition.

2.5 Software

2.5.1 App Design

It was decided to use Xcode and Swift to build the app as the Apple Watch could be used for data collection, and the same sensors would be used for consistency. Figure 6 shows the application architecture for an Apple Watch App with a paired iPhone App. This architecture also only works for an iPhone as the Watch cannot connect to an iPad. The app structure allows the sensors to be accessed through the WatchOS App, and then for the iOS App to be situated with the coach, displaying useful statistics as the shots are hit.

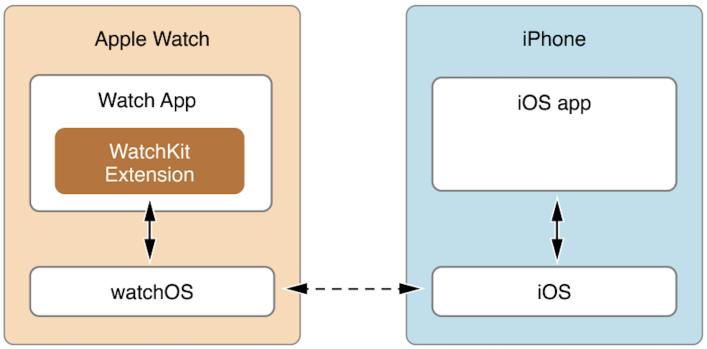

Figure 6: *Basic Application architecture [20]*

Using the WatchKit framework in Figure 6, the watch app can then rely on the paired iOS app to perform the computationally expensive shot classification. This is beneficial as the watch app needs to process data at 80Hz, whereas the phone app can run the classification in between shots whilst the watch continues to collect data. The Watch Connectivity Framework is used to provided this two way communication, where the data is exchanged via Bluetooth if the phone is within reach, or via Wi-Fi if available and bluetooth is not [21].

2.5.2 Post-processing analysis

Previous work has used MATLAB to analyse the data using its ”Classification learner toolbox” to train and test the various algorithms [3]. However, it was decided to carry out the same analysis for this project in Python, because of personal expertise, its wide use for machine learning research [11] and it is a more valuable language to develop. Additionally, Core ML, Apple's machine learning framework, enables the creation of models made outside of Apple's ecosystem, such as inside TensorFlow or SciKit Learn, using a set of command line tools [22]. Therefore, using Python allows an algorithm designed using one of the aforementioned machine libraries, can subsequently be implemented into Xcode as it is an open-source model from one of the compatible machine learning libraries.

3 Data Collection

An Apple Watch has been the source of data collection for this project. As described in Section 1.2, there have been two stages of data collection for the project, 'Phase 1 Data' collected using a third party app, for subsequent post processing analysis, and 'Phase 2 Data' using an app programmed for the project to allow for real time analysis. Both apps however record the same information which is as follows;

1. *General Logging Info*
2. *User Acceleration Data (X,Y,Z)*
3. *User Rotation Data (X,Y,Z)*
4. *User Gravity Vector (X,Y,Z)*

3.1 Phase 1 Data

'SensorLog' [23] was used to collect data that was configured to record the above features at a sampling rate of 80Hz, outputting the logged data as a *.csv* file for each session. Figure 7 shows the set up used for the majority of the data collection. An isolated setting was created using a bowling machine to ensure to deliveries for a given class were all landing in the correct area, to ensure a clean initial dataset. Varying the ball and speed of the machine, the data collected became more generalised, eventually removing the machine entirely and using data from actual bowlers with a hard cricket ball.

Figure 7: *Data Collection Set-Up*

Shots were hit consisting of only one shot type per $.csv$ so that the bowling machine did not have to be moved frequently. This was repeated for all five classes, and four different players, of both left and right handed players, to ensure no player or orientation bias. The raw data numbers for each shot collected can be seen in Table 1.

Table 1: Phase 1 Data Collection

Class	Player 1 (LH)	Player 2 (LH)	Player 3 (RH)	Player 4 (RH)	Total
Defensive	30	29	31	26	116
Drive	25	26	26	28	105
Cut	22	25	27	25	99
Pull	25	23	28	26	102
Sweep	30	27	28	32	117
Total	132	130	140	136	**538**

3.2 Phase 2 Data

The previous method of data collection, SensorLog, only has the ability of producing one *.csv* file at the end of a session and so cannot be used data collection and analysis could not occur simultaneously using this application.

Therefore, a new app was built to collect data in the same way but with the advantage of being able to add in real time analysis. The accelerometer and gyroscope sensors within the Watch, those used by SensorLog, can be accessed via Apple's Core Motion framework [24]. Core Motion reports motion - and environment-related data from the onboard hardware of WatchOS devices and this framework allows use in an app [25]. Additionally, the user will very rarely interact with the UI whilst batting, which means that the app needs to be able to access the sensor readings when the screen goes off and the app moves into the background. This can be done by implementing the Apple's Health Kit framework [26]. Any data collected using this app was called 'Phase 2 Data' as distinguishing between the data collection methods is important when evaluating the project's results. The raw data collected is shown by Table 2.

Table 2: Phase 2 Data Collection

Class	Player 1 (LH)	Player 5 (RH)	Player 6 (RH)	Player 7 (LH)	Total
Defensive	52	43	44	39	178
Drive	46	25	33	29	133
Cut	36	32	28	31	127
Pull	41	34	26	29	130
Sweep	28	32	38	24	122
Total	203	166	169	152	**690**

Overall, 1228 shots were collected across 7 different players that were all experienced, high performing cricketers.

4 Shot Detection and Segmentation

4.1 Shot Detection

The shot detection routine for the two phases is largely similar, with the slight difference being that for phase 1, multiple peaks are found within a finite *.csv* file, which means it is straight forward to use data after a peak to evaluate a shot. In phase 2, the routine has to occur whilst the data is continuing to be processed.

4.1.1 Detection Routine

When the ball impacts the bat, there is a spike in acceleration and rotational velocity, therefore detecting these peaks can provide the temporal location of each shot. To detect this impact, the magnitude of acceleration across the three axes is taken, given by Equation 1. The 80Hz signals are smoothed using Equation 2 to ensure only one peak for each shot. This moving average filter has a window size, n, of 20 to remove short-term fluctuations.

$$|a| = \sqrt{a_x^2 + a_y^2 + a_z^2} \tag{1}$$

$$|\overline{a}| = \frac{1}{n}\sum_{i=1}^{n} |a|_i \tag{2}$$

External noise the sensors will detect is peaks due to bat taps on the ground, swinging & missing and running. A detection routine must be able to distinguish between a shot and this external noise. The detection conditions that were configured to achieve this are displayed in Equations 3-5. Equation 3 sets a threshold value for the acceleration magnitude $(3.2G)$ - which was determined via trial & error, and graphically viewing the signal. Equation 4 then sets the minimum time between two peaks to 3s where $t_{|\hat{\overline{a}}|,i}$ and $t_{|\hat{\overline{a}}|,i+1}$ are the temporal locations of consecutive peaks. This was evaluated using the time between shots on the fastest setting of the bowling machine. Without a machine, this value is usually larger.

$$|\overline{a}|_i > 3.2G \tag{3}$$

$$t_{|\hat{\overline{a}}|,i} - t_{|\hat{\overline{a}}|,i+1} > 3s \tag{4}$$

Lastly, the peak prominence was set to 1G, shown by Equation 5 & 6, again determined via trial & error and graphically viewing the signal. $|\overline{a}|_i$ is the acceleration magnitude of the peak, and $|\overline{a}|_{i+1}$ is the acceleration magnitude of the next point.

$$|\overline{a}|_i - |\overline{a}|_{i+1} > 1G \tag{5}$$

$$|\overline{a}|_i - |\overline{a}|_{i-1} > 1G \tag{6}$$

4.1.2 Testing and Evaluation

Figure 8 shows the detected strokes for the 23 pull strokes of player 2. Table 3 then shows the overall detection accuracy for both post-processed and real time detection, accounting for both shots that were not detected and false positives.

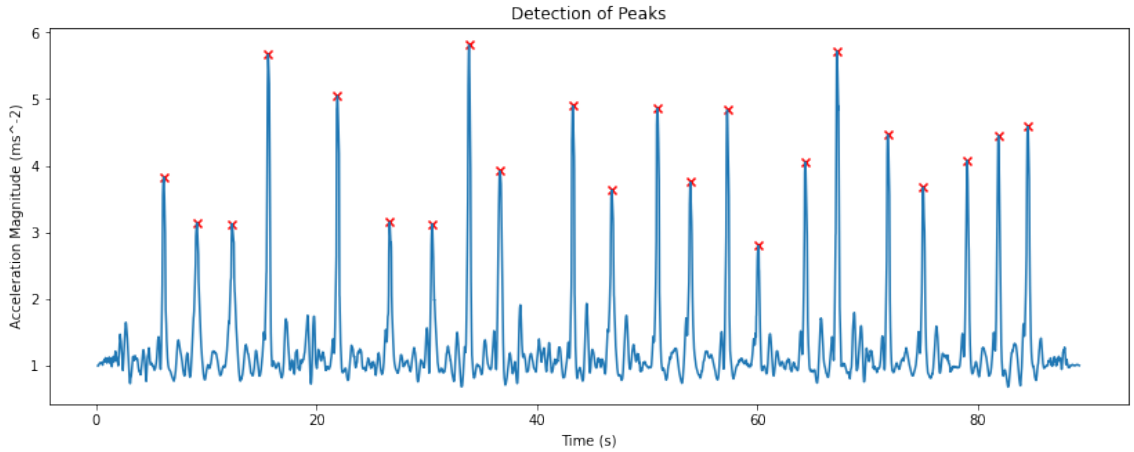

Figure 8: *A graph showing shot detection*

Table 3: Shot Detection Accuracy

Software	Correctly Detected	False Positives (Bat Tap)	False Positives (Running)	False Positives (Swing & Miss)	Accuracy
Python	520 / 538	6	2	9	93.5%
Xcode	679 / 690	16	19	23	90%

Table 3 shows that the detection routine performed well at detecting real shots, however, the majority of the error came from noise resulting in a false positive.

The lower accuracy for the real time detection can be down to the fact that this data was collected in a real training session where batsmen were not constrained to just hitting one shot repeatedly with a bowling machine. This meant that there were far more missed shots and the bat tapped false positives were higher.

Detection performance can be improved by looking at an example signal of a false positive due to running, shown in Figure 9. It is clear that the running signal has numerous peaks above the threshold given in Equation 3, within a small time window.

The minimum distance defined in Equation 4 will limit the running signal shown to just 1 false positive for every 3 seconds - however does not remove it entirely. Equation 7 shows the adjustment added to the end of the detection routine to reduce the error due to running. The integral is estimated using the trapezium rule with the real shot having a lower area under curve due to the batsmen being very still just before the shot.

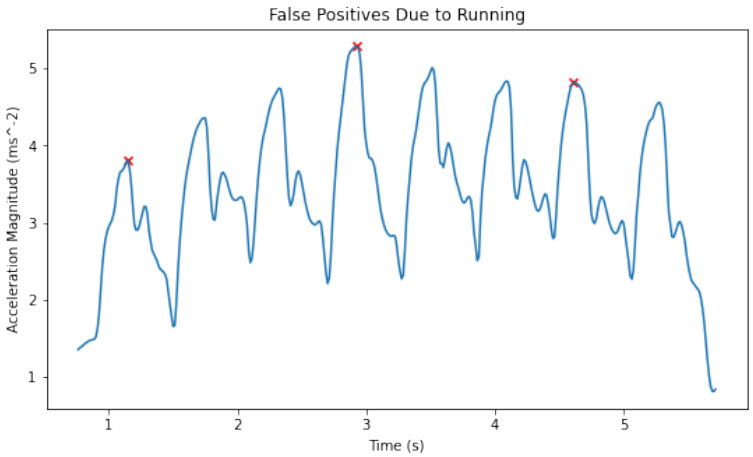

Figure 9: *Acceleration Magnitude of running with a cricket bat*

$$\int_{t=0}^{t=1.5s} y(x)dx < 0.5 \tag{7}$$

This equation however, requires bounds to integrate between and so the section below on segmentation must occur prior to Equation 7.

4.2 Segmentation and Alignment of Signals

The stroke signals can be extracted from the data. The approximate time for a shot was chosen as 1.5s, based on inspection of video analysis during data collection and literature [9]. The 1.5s signal, for each of the 6 sensors, is therefore extracted by taking the temporal location of the peak to be the middle, and extracting 0.75s either side.

However the peaks / troughs for the individual sensors may not be aligned due to different timings of one shot to the next. We can therefore re-align the shots more accurately before evaluating shot features for the classification algorithm. However, the 'model' signal for each to be aligned to is unknown. To determine the template signals, a pairwise cross correlation-based averaging procedure was used [27].

All signals for a given class are split into pairs and the cross correlation is evaluated between them - which analyses the pairs of time series data relative to one another and at what point the best match occurs. For every shift in time, the dot product between the two signals is performed - described by Equation (8) where m is the lag and x_1 and x_2 are the two signals. The pairs of signals are then realigned and the average is determined, a process that continues until a template for each class has been evaluated.

$$(x_1 * x_2)[m] = \sum_{n=-\infty}^{\infty} \overline{x_1[n]} x_2[n+m] \tag{8}$$

Up until this point, the player's hand orientation (left or right handed) has not been of im-

portance, however as discussed in Section 2.1, the different orientations on the watch will result in different template signals for the two orientations. Creating two templates could have been implemented, however for the template shown in Figure 10, the x-axis values of the right handed data were flipped so that the pairwise cross correlation procedure could construct accurate templates based on all data. This method of flipping the x-axis occurs for all right handed shots so one classification algorithm can be used.

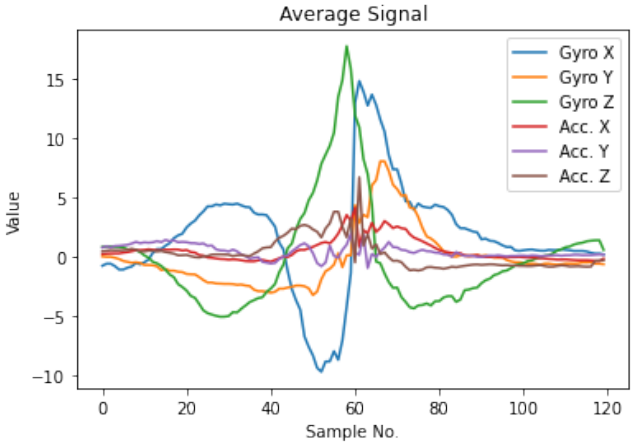

Figure 10: *Cut Model Output*

Lastly, we compare every shot to this template and re-align all signals. Figures 11 & 12 show how this process aligns the signals for the sweep shot of the left handed Player 1. One disadvantage to this method is the number of samples used to evaluate the template must be a power of 2, so the templates shown were only trained on 64 shots for each class. Another disadvantage to this method is that this method is computationally expensive. This is not an issue for the post processed data, however in real time, the total run time for all steps needs to be less than the time between shots, needing concise and efficient algorithms. For this reason, the templates evaluated from this section were saved and imported into Xcode so that every shot simply just needed to be segmented and re-aligned.

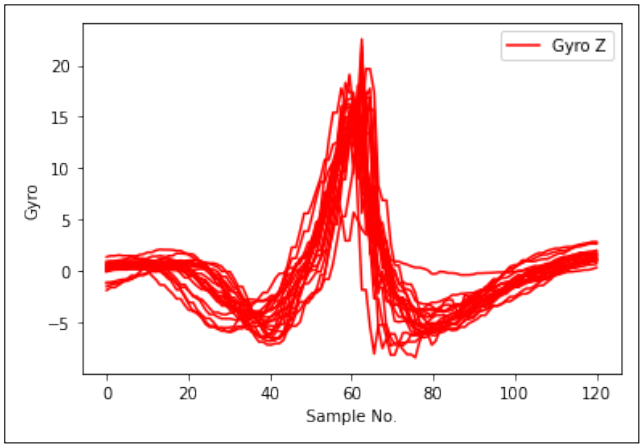

Figure 11: *Segmented signals*

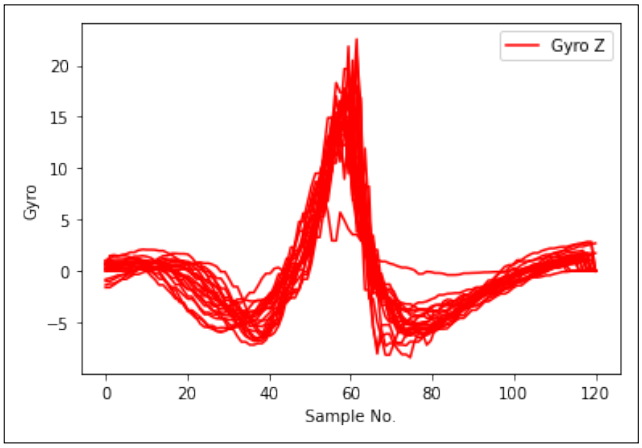

Figure 12: *Segmented & Aligned Signals*

4.3 WatchOS App

The purpose of the Apple Watch App is to implement the detection and segmentation stages, as described, before passing the segmenting signals to the phone for the heavy analysis. Using the framework described in Section 2.5, a Watch OS App was developed with Figure 13 displaying a flowchart of the watch app, showing how the detection and segmentation have been integrated into the app. The headings at the top of the flowchart reference the $.swift$ file at which the stages are held within the app's workflow.

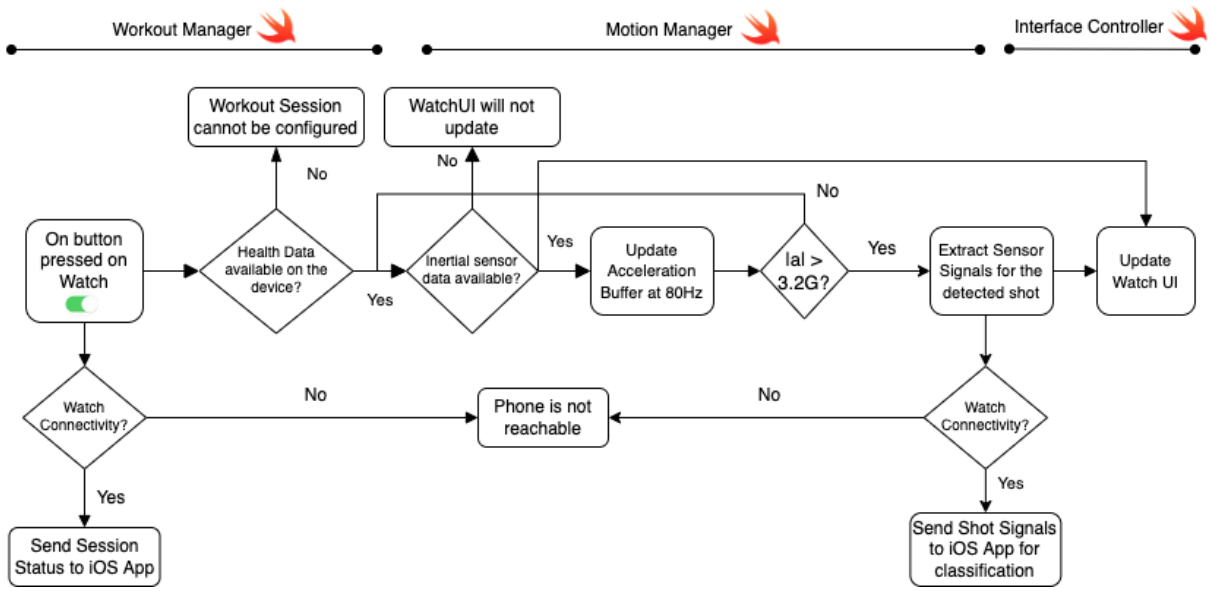

Figure 13: *WatchOS App Flowchart*

4.3.1 Watch App User Interface

Figure 14 shows the app's UI (left), and the app display in the watch library (right). The UI contains a start/stop toggle, a shot count and a button to save the session and reset back to 0. The sensor readings are also being printed onto the screen, which was implemented for debugging, but in principle the 80Hz refresh rate is too fast useful for the user.

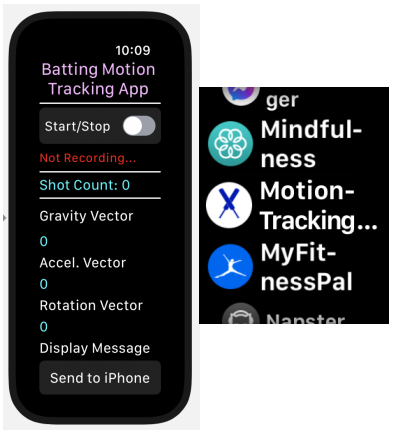

Figure 14: *Watch User Interface (UI)*

5 Classification

5.1 Shot Characteristics

This section will look further into detail around the kinematics of the different classes and how they might be distinguishable from one another.

5.1.1 Defensive Shot

Figure 15 shows the two stages of a defensive cricket shot. The initial stance (left) is assumed to be the same for the analysis of this section, with (right) showing the impact point of a typical defensive shot.

Figure 15: *Defensive Shot Stages*

Figure 16 shows the averaged sensor signals for the defensive shot. It can be seen that the magnitude of all values are low in comparison which is as expected.

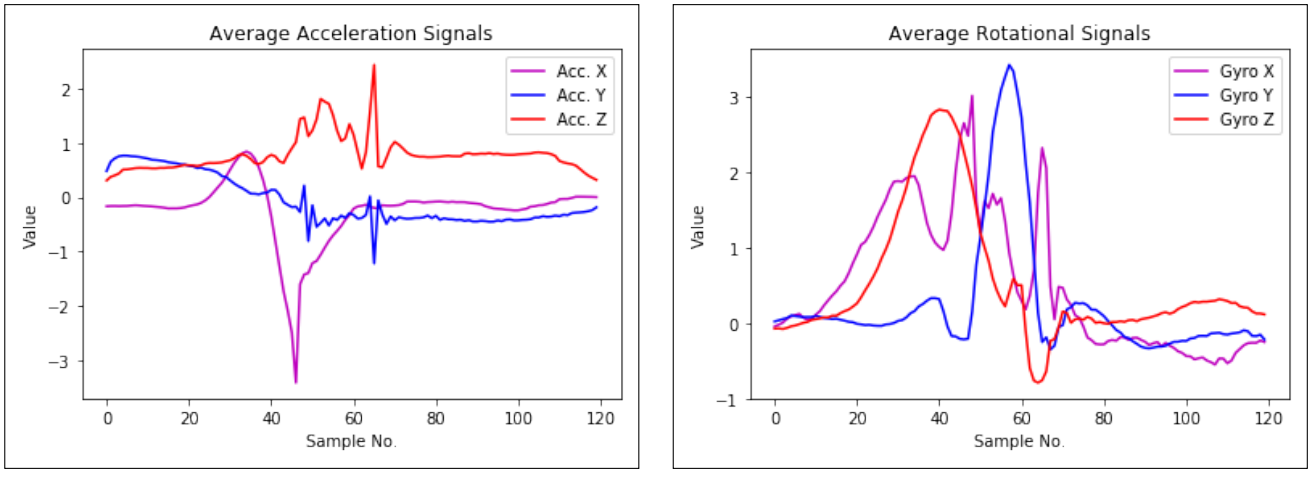

Figure 16: *Averaged Defensive Shot Signals*

5.1.2 Drive Shot

The drive is very similar to the front foot defensive, but hit with more power as it is a scoring shot. This is displayed in Figure 18 where the general shape of the average acceleration is the same, but with a larger peak and trough for the z and x axis respectively.

Figure 17: *Drive Shot Stages*

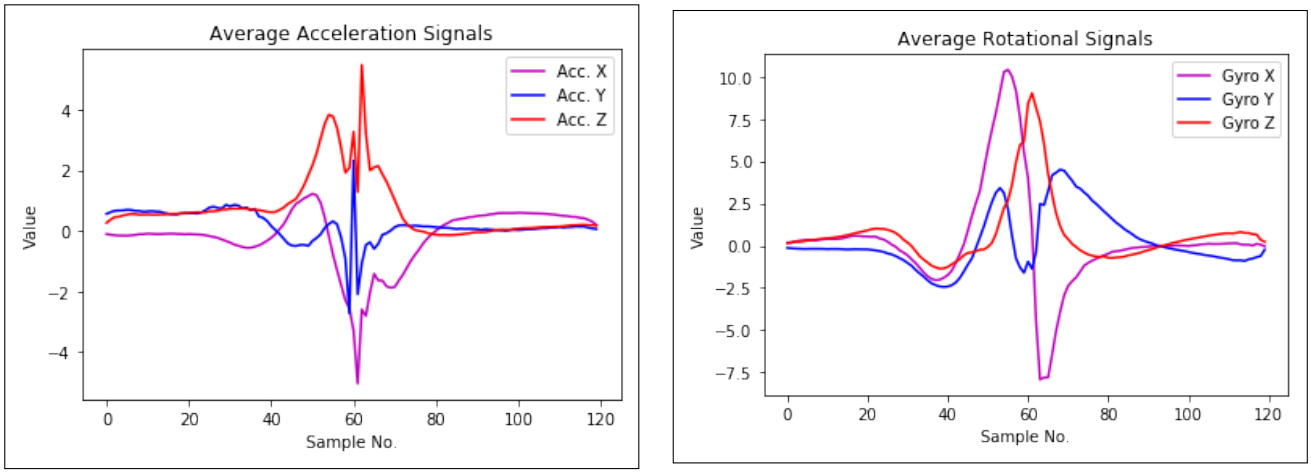

Figure 18: *Averaged Drive Shot Signals*

5.1.3 Cut Shot

A key feature of the cut shot, in going from the initial stance, Figure 19 (left), to the solid base to play to play the stroke, Figure 19 (right) is to rotate the wrist anti-clockwise around the axis. To stay in control of the shot, the wrist is then rotated clockwise through impact to keep the ball on the ground. This can be seen in Figure 20 where the x-axis rotation follows a sine wave form, indicating these two rotation stages.

Figure 19: *Cut Shot Stages*

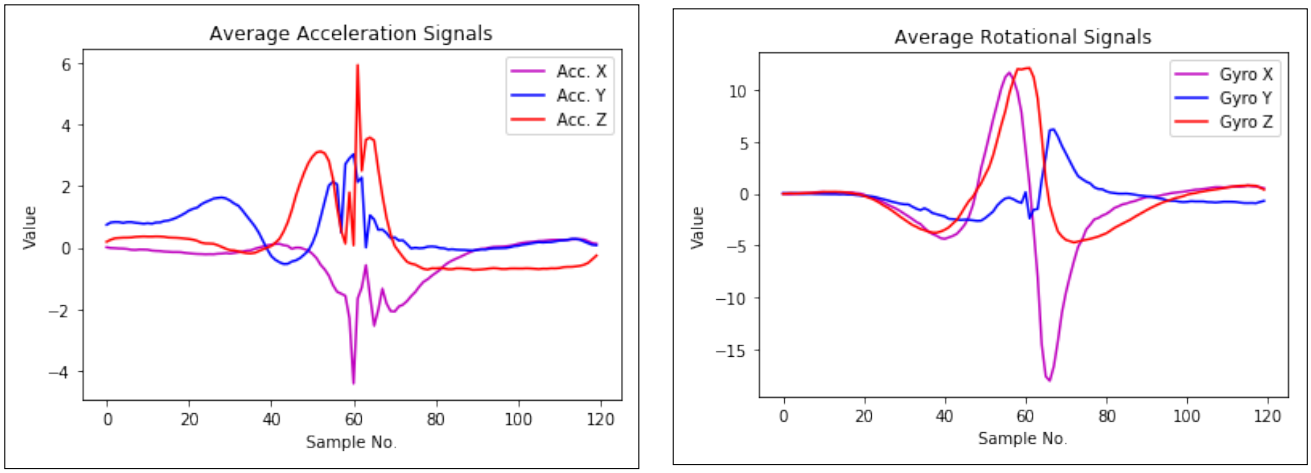

Figure 20: *Averaged Cut Shot Signals*

5.1.4 Pull Shot

The pull shot also follows the same x-axis rotation profile as with the cut shot. The wrists are rotated anti-clockwise to get the width of the bat from parallel to the floor to perpendicular, and then rotated clockwise through impact, just 180° further round, as shown in Figure 21. An experienced player also uses the flicking of their wrists to generate more power through impact, which translates to the spike in the z-axis rotation, shown in Figure 22.

Figure 21: *Pull Shot Stages*

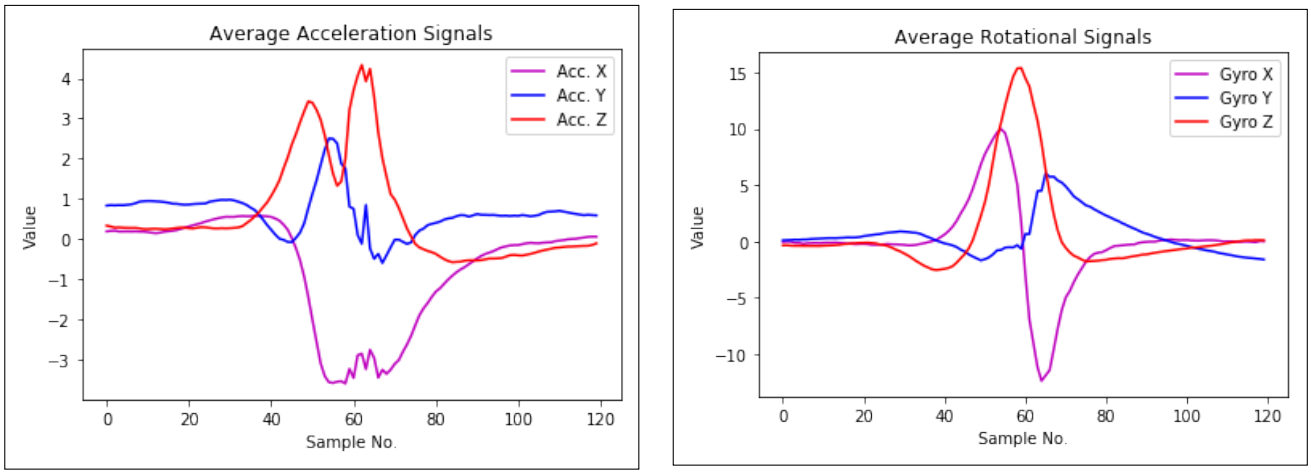

Figure 22: *Averaged Pull Shot Signals*

5.1.5 Sweep Shot

The final class is the sweep shot, shown in Figure 23. As discussed, the pull and sweep shots have a similar shot path and so the signals follow the same general shape, as shown in Figure 24.

Figure 23: *Sweep Shot Stages*

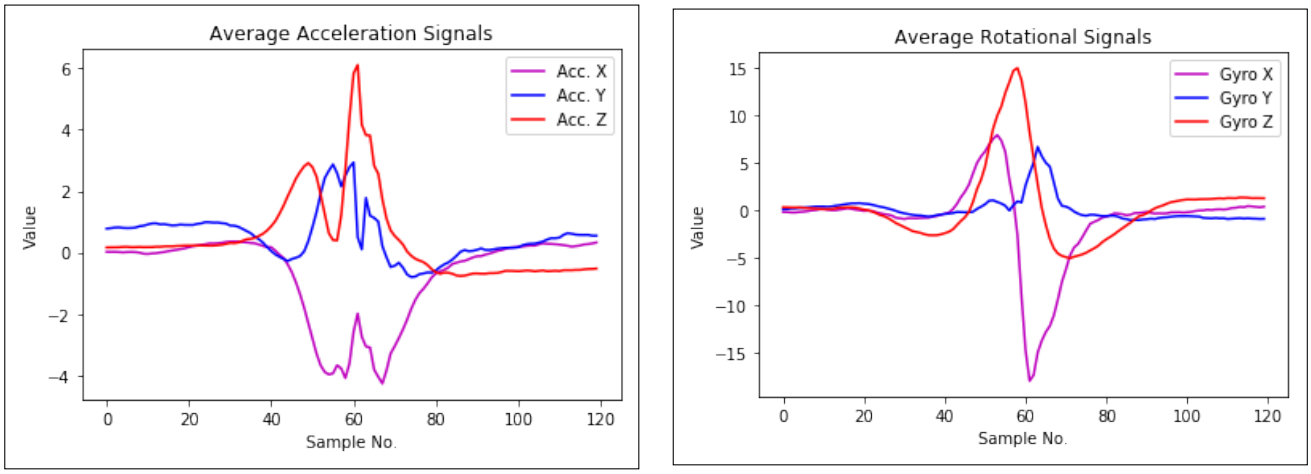

Figure 24: *Averaged Sweep Shot Signals*

5.2 Feature Selection

The last step before classifying was to evaluate the features to input into the algorithm. It was evaluated to initially use 5 features (mean, std, integral, max, min) for each of the 6 gyroscopic and accelerometer axis - giving a total of 30 features for each shot. The features were also normalised prior to entering the algorithm to make comparison between features easier, using Equation 9.

$$z = \frac{x - \mu}{\sigma} \tag{9}$$

where z is the normalised value, x is the original feature vector, μ is the mean of the feature vector and σ is its standard deviation. This normalisation results in each feature having zero mean and unit variance so that all features have equal influence. This method is widely used in machine learning, especially in SVMs [28].

5.2.1 Dimensionality Reduction

Using the 30 features listed above, Principal Component Analysis (PCA) was then undertaken to reduce the dimensional space of the data to speed up the algorithm and increase efficiency. This results in a lower-dimensional projection that preserves the majority of the variance, by zeroing out the smaller principal components [29]. The results found that 99% of the variance was in the first 6 principal components, which is shown in Figure 25.

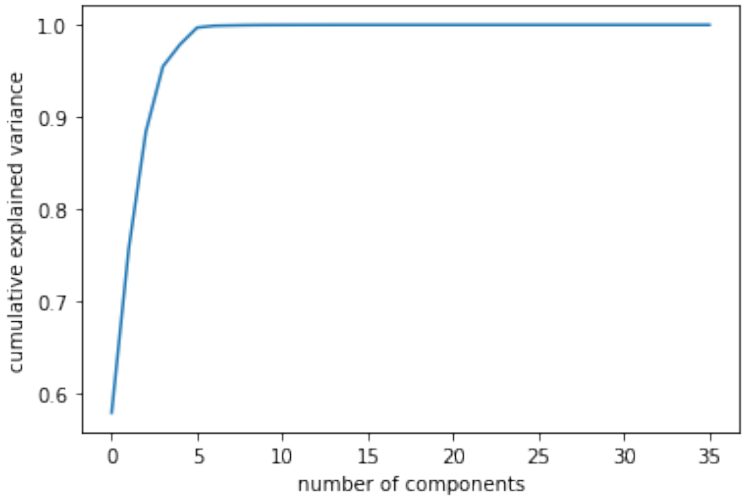

Figure 25: *The cumulative variance explained with number of principal components*

5.3 Clusters

In order to gain a visual understanding of the features selected, certain features were plotted against each other to give data clusters. The clusters show how the feature values

vary for the different classes. Figure 26 shows an example feature plot of the maximum rotational value in the z direction against integral under the rotation in x curve, for all the shots from the phase 1 data. It can be seen that the the cross batted shots, cut, pull and sweep, have a higher Z rotation at impact, which confirms the free body diagram analysis from Section 5.1. The denser the cluster, the smaller the variance and as a result it will be easier for the algorithm to classify shots of that class.

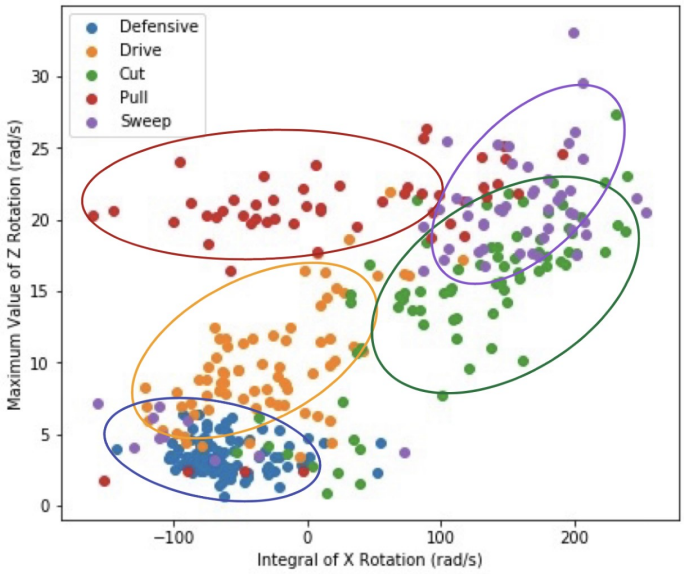

Figure 26: *Example Cluster from feature selection*

Figure 27 then shows a cluster plot of the first two principal components, after PCA had been carried out, with 75% of the variance being characterised by these two components.

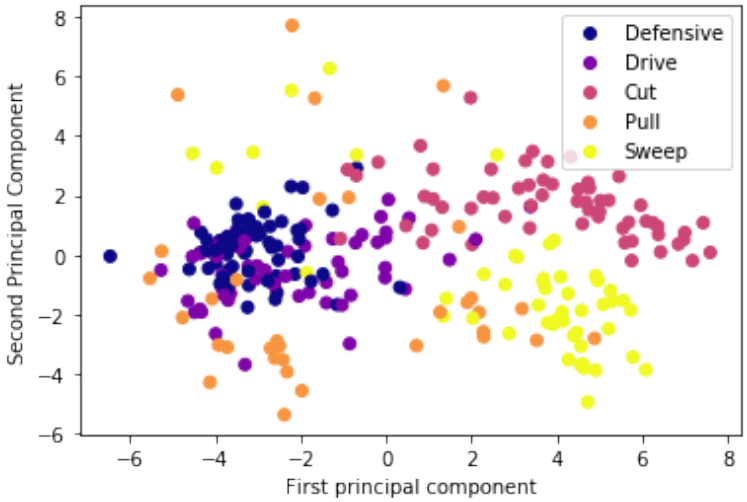

Figure 27: *Example Cluster after PCA*

5.4 Algorithm Selection

During the project, three different algorithms were used for shot classification.

1. Algorithm 1 - Gaussian SVM using SciKit Learn (Python)
2. Algorithm 2 - NN through Create ML (Xcode)
3. Algorithm 3 - Gaussian SVM through Core ML Converters (Xcode)

Algorithm 1 was optimised using the phase 1, post-processed data, making use of Python's machine learning library, SciKit learn. Algorithms 2 & 3 were then used for classification in real time, using the two different ways to train and embed a Core ML, machine learning algorithm into an iOS App [30]. It was decided to implement both methods to evaluate if there was any difference in performance, which is discussed below.

5.4.1 Algorithm 1 - Gaussian SVM using SciKit Learn

The supervised learning algorithms explained in Section 2.3 were experimented with and optimised using Python's Scikit Learn and Keras machine learning libraries. Table 4 shows the summary of the best training data outcome for each of the algorithms types.

Table 4: Algorithm 1 - Training Data Accuracy

Type	Accuracy
SVM	92.5%
NN	89.9%
KNN	91.2%

The conclusion was that SVMs were the superior algorithm for this application. This algorithm was optimised using a gaussian kernel and reducing the sensitivity to individual support vectors by allowing some data points to lie outside the margin region. This is achieved by lowering the value of the soft margin classifier, C, and hence makes the classification less over-fit [12]. Figure 28 shows the algorithm accuracy for various C values.

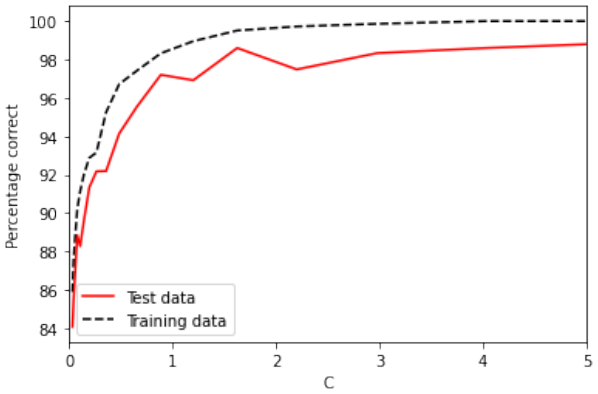

Figure 28: *Accuracy for Various SVM Models*

Figure 28 was evaluated using 4-fold cross validation to obtain a more accurate plot. This uses 75% of the data as the training set and 25% as the test set - repeating the process 4 times so that all data is used to test. Minimising the difference between training and testing accuracy gave a final accuracy of 96.7% for the training data, for a soft margin classifier of 0.48.

To show where the mis-classification was occurring, a confusion matrix was plotted, shown in Figure 29. It can be seen that the main error occurs due to mis-classifying the pull and sweep shots. This can be explained by the similar nature of these shots - as described in Section 5.1.

Removing the alignment section using pairwise cross correlation, due to its high computational expense reduced the training accuracy to 94.6%. Lastly, the training accuracy of an SVM trained just on the first 6 principal components was 92.16%, with the confusion matrix for this showing in Figure 30.

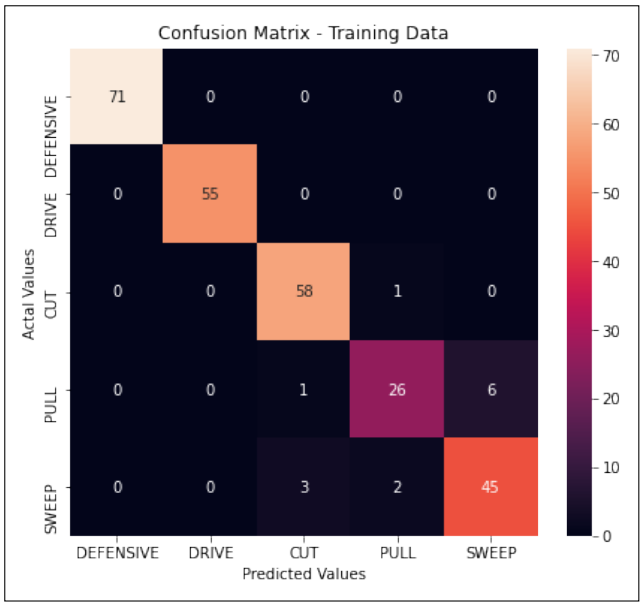

Figure 29: *Confusion Matrix (Raw Training Data)*

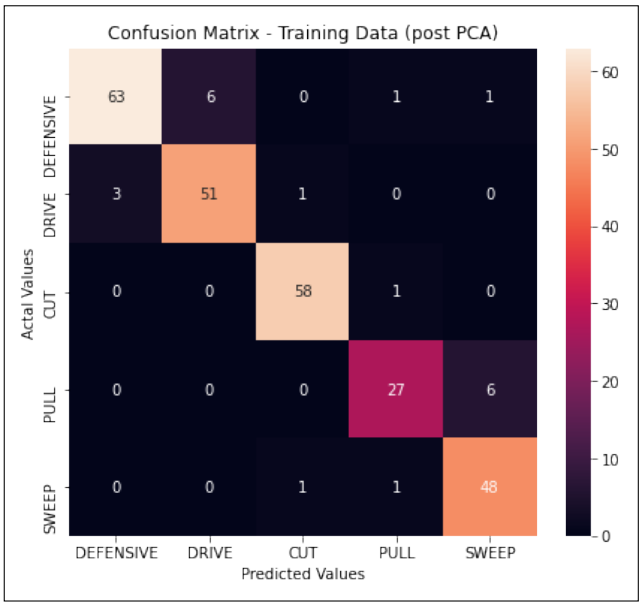

Figure 30: *Confusion Matrix (PCA Training Data)*

5.4.2 Algorithm 2 - NN through Create ML

The first of the Xcode machine learning algorithms uses Create ML, a standalone app, that enables the use of previously collected data to train, test and build a CoreML model without writing code, using models already prepared by Apple [31]. The big advantage of this method is the ease of deployment. Once the machine learning model is working, integrating it into the application code is straightforward. Although, it does mean that Algorithm 1 could not be re-used. Figure 31 shows a flowchart indicating the process of building an algorithm in create ML. Within the platform, there is an activity classification toolkit, specially designed for inputting sensor signals to produce a classification algorithm.

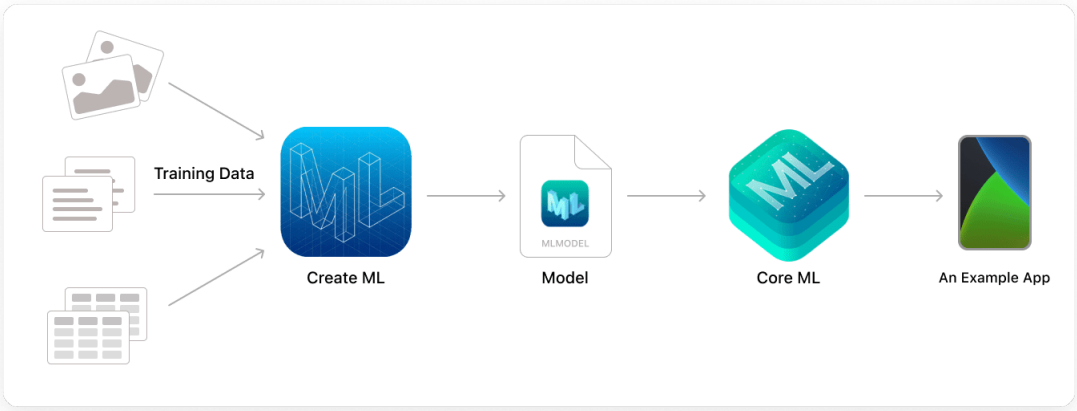

Figure 31: *Flowchart indicating the process of building an algorithm in create ML [31]*

The algorithm that the activity classification toolkit uses is a Neural Network (NN) [32]. Figure 32 shows the training and validation accuracy of the NN was 90.3% and 100% respectively - after 30 iterations. The training data for the algorithm was the segmented shot signals from SensorLog, from Section 4.2.

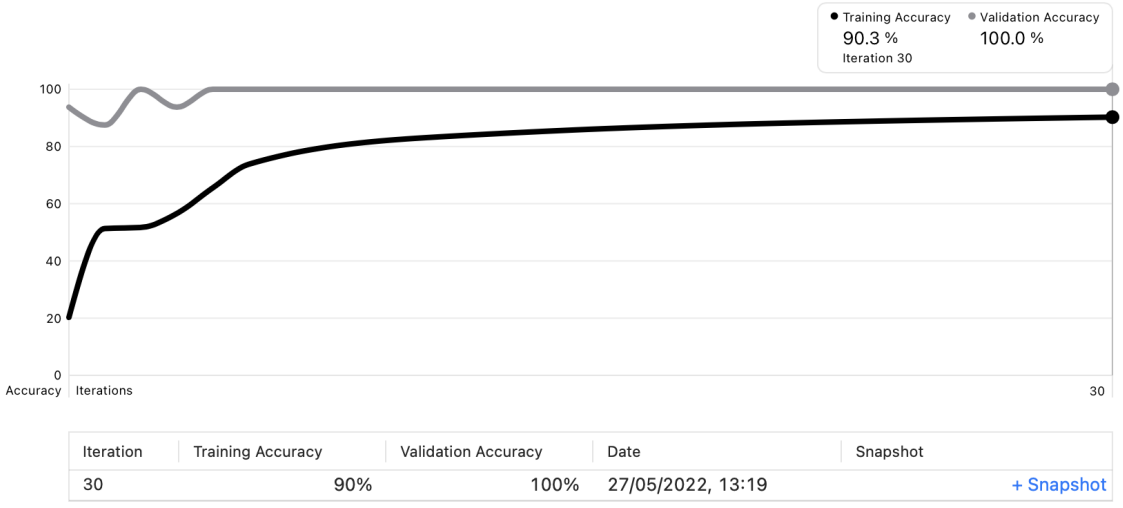

Figure 32: *Training and Validation Accuracy of the NN*

The two main differences in comparison to Section 4 is that a Neural Network is being used as opposed to an Support Vector Machine, and that the algorithm is inputting the raw 1.5s signals as oppose to a subset of features. The latter is because the activity classifier is programmed to input the whole signal, and the former down to the inability to be able to tweak the algorithm to optimize the model [30]. However, the low implementation time sped up the process to having a initial functional real time application. The activity classifier creates a deep learning model capable of detecting temporal features in sensor data, lending itself well to the task of activity classification. Figure 33 shows the layers of the NN, briefly described in Section 2.3.

Layer Distribution

Layer	Count
Slice	2
Concat	2
InnerProduct	2
ActivationReLU	2
Reshape	1
Convolution	1
BatchNorm	1
UniDirectionalLSTM	1
Softmax	1

Figure 33: *NN Layers*

5.4.3 Algorithm 3 - Gaussian SVM through Core ML Converters

Core ML also allows the conversion or creation of models made outside of Apple's ecosystem, such as inside TensorFlow or SciKit Learn. Open-sourced models from the aforementioned machine learning libraries can be converted to Core ML using Core ML Converters [33]. The advantage of this method is that the exact same gaussian SVM algorithm as in Section 4 can be implemented into Xcode.

Once the app was fully functional with Algorithm 2 being used for classification, the last stage was to implement the gaussian SVM that was evaluated to be the best performing algorithm for the use case. This requires the model to be saved as a $.h5$ file and then converted to a $.mlmodel$. The integration within Xcode is the same as for Algorithm 2. The algorithm was also trained using the phase 1 data and so the training accuracy was the same as Algorithm 1, 96.7%.

5.5 iOS App

5.5.1 Workflow

Where the Watch OS App was responsible for detection and segmentation, as discussed in Section 4, the iOS App is responsible for classification, using the $.mlmodel$ algorithms described above. Figure 34 shows a flowchart of the phone app, showing how the classification is carried out and subsequently how the User Interface (UI) is populated.

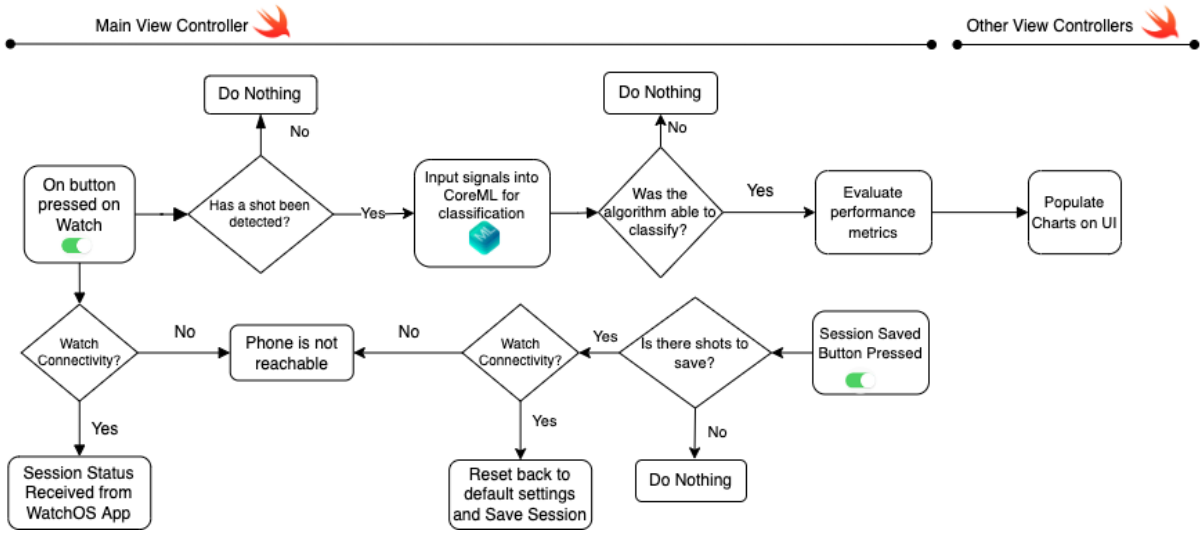

Figure 34: *iOS App Flowchart*

5.5.2 iOS App User Interface

Unlike the watch UI that just required the user to be able to turn on/off data collection, the iOS App requires both the classification integration in its back end, as mapped out in Figure 34, but also a UI in its front end that is engaging for the user. Once the classification integration was reliably working, other functionality was built to achieve this. The app was designed with 4 tabs, splitting up and modulating the information the user is analysing.

1. Home Tab
2. Statistics Tab
3. Raw Shots Tab
4. Session History Tab

5.5.2.1 Home Screen

The most important tab in the app is the Home screen due to the fact that all communication from the watch arrives at the $.swift$ file connected to this screen. This screen has several features, shown by Figure 35 (left & center). Similarly to the Watch, there is a shot

count and save session button. The two main features of the tab are the interactive and animated graphs, a pie chart showing the breakdown of shots in the session, and a line chart displaying the sensor signals of the most recent shot, with the UI buttons allowing the user to view the signal for any of the 6 sensors.

5.5.2.2 Statistics Screen

Figure 35 (right) shows the Statistics Screen display. As shown in Figure 34, once the class has been identified, the performance metrics of the shot, bat speed, shot quality, back lift angle and impact angle, are evaluated and displayed in the form of a radar chart. The 'Select Shot' button can be used to view these metrics for any particular shot. More information on the methods behind these metrics are in Section 6.

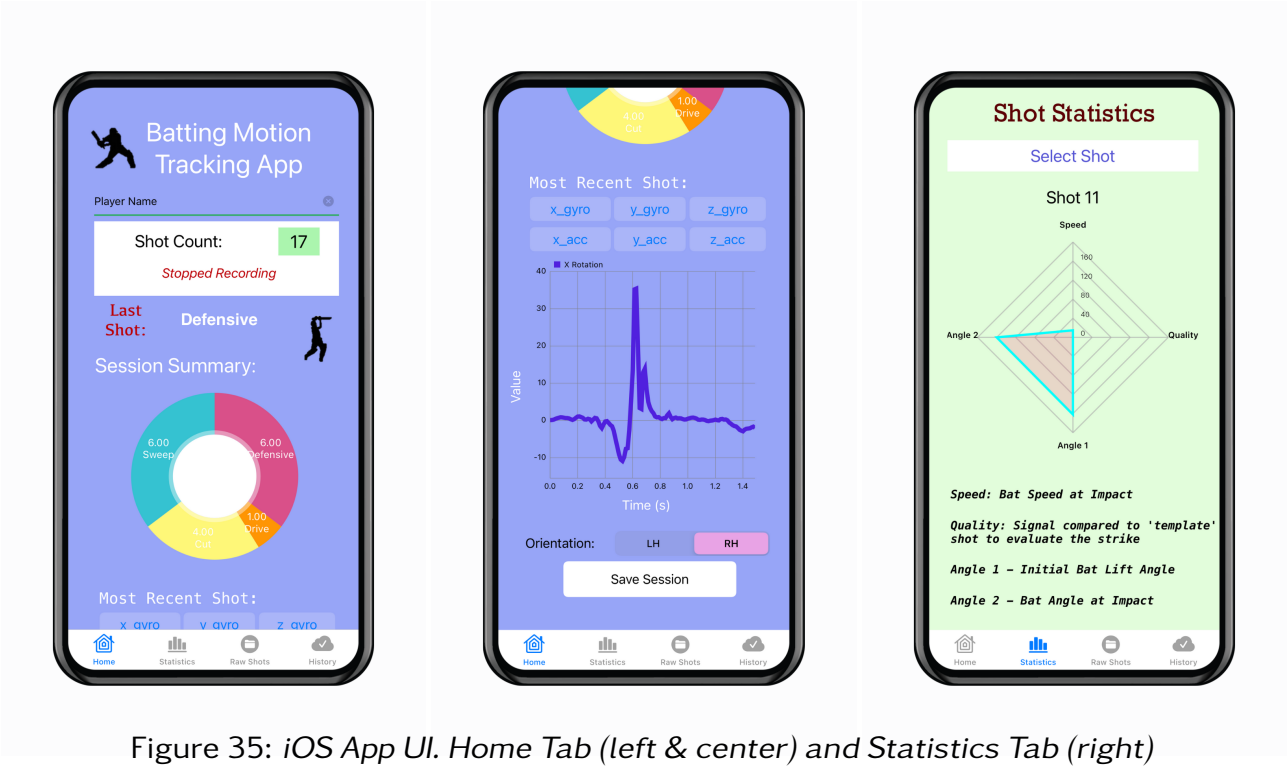

Figure 35: *iOS App UI. Home Tab (left & center) and Statistics Tab (right)*

5.5.2.3 Raw Shots Screen

Figure 36 shows the Raw Shots Screen display. This tab initially displays a table view of all the shots in the session. Once a shot is clicked on, a graph of the sensor signals for that shot is displayed. Additionally, this screen allows for the player / coach to annotate the shot with the real class, as shown at the bottom of the screen. This allows the app to both compute a value of the algorithm accuracy, as evaluated in the following testing below. Although, more importantly, indicating the actual class allows the app to seamlessly retrain the classification algorithm to become more accurate with time. This final point is not currently implemented, however updating the *.mlmodel* with the new training data is available to program via the coreML framework.

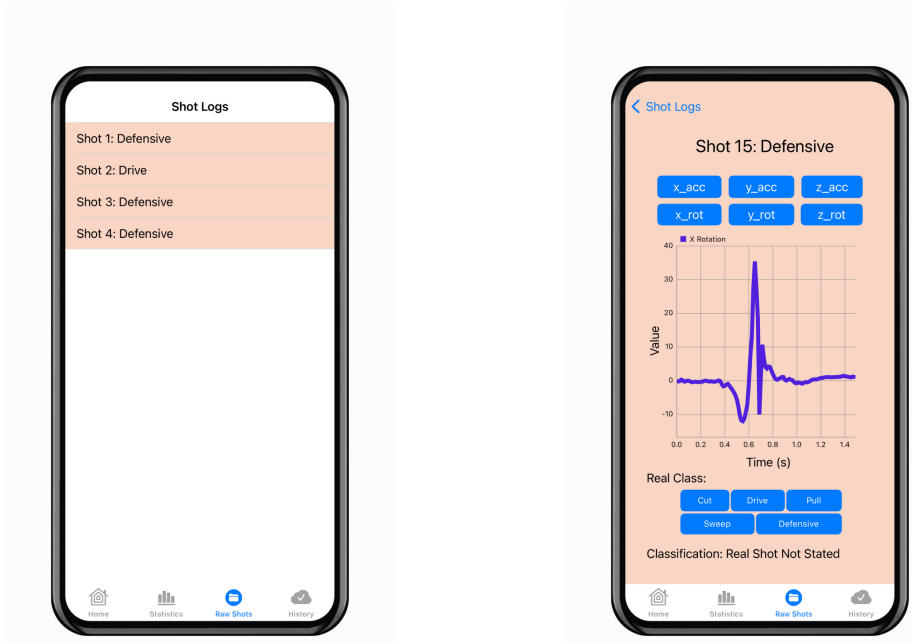

Figure 36: *Raw Shots Screen*

5.5.2.4 Session History Screen

Once the save session button is pressed, a new row in the History tab's table view is populated, shown in Figure 37. If clicked on, a session summary is displayed. This summary screen collects the name and orientation of the player, the start and end time, and the algorithm accuracy. This accuracy will only be accurate if the coach has annotated all shots. Lastly, the pie chart from the Home Tab is saved as a *.png* image and displayed.

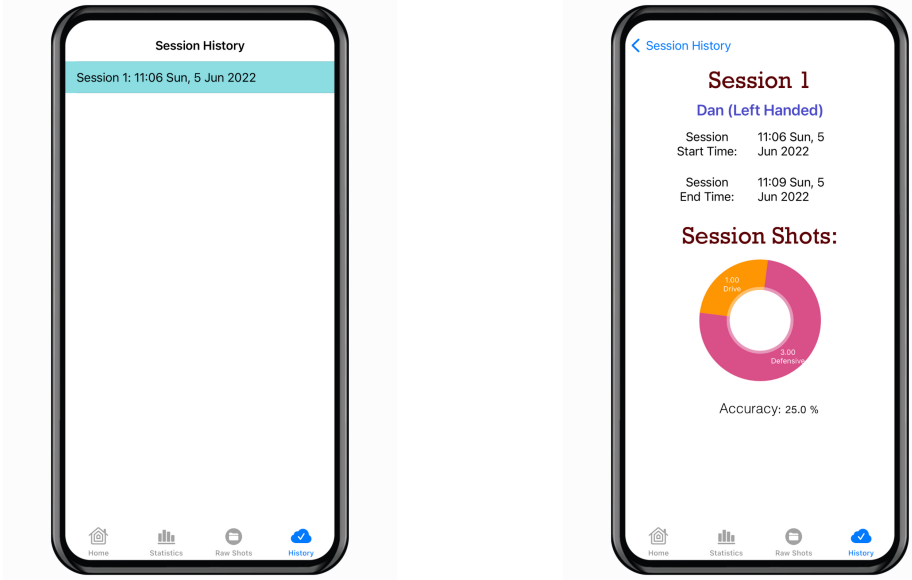

Figure 37: *Session History Screen*

5.6 Testing and Evaluation

The algorithm summaries above highlight the training data accuracy's of the three algorithms. Testing the algorithms can then be split into two stages. Firstly, testing the performance can be done via the 25% percent of data held back from the algorithm on each iteration of the K-Fold Cross Validation. All algorithms were trained and initially tested using this method on the phase 1 data and so the performance can be easily compared. Secondly and more importantly, for algorithms 2 and 3, the phase 2 data collected in real time will test the classifications accuracy and speed.

Table 5 shows a top level summary of the accuracy's of the 3 algorithms, for both training and testing data. For algorithms 2 & 3, this includes both stages of testing.

Table 5: Classification Accuracy (Testing Data)

Algorithm	No. of Shots	Training Accuracy	Testing Accuracy
1	538	96.7%	92.8%
2	550	90.3%	76.2%
3	140	96.7%	87.9%

Figures 38 shows the confusion matrix for the testing data for Algorithm 1. Both real time algorithm's run time was suffcient without the use of PCA and so the adjusted performance due to this has been omitted.

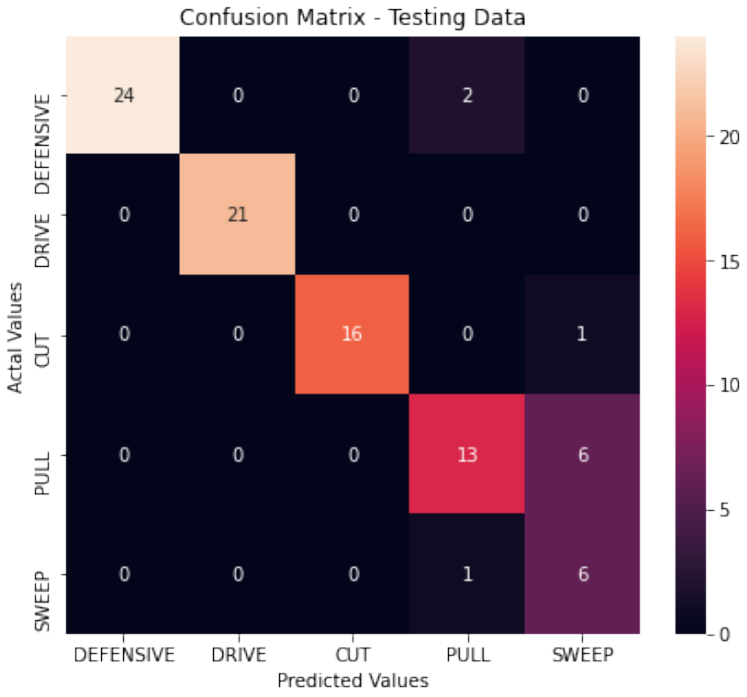

Figure 38: *Confusion Matrix (Testing Data)*

Figures 39 & 40 then show the confusion matrices for algorithm's 2 and 3. The performance of algorithm 2 was the worse of the two real time algorithms used. The cut shot and defensive shot were easily detected, however the main areas of mis classification were similarly the pull and sweep shot, but also drive shots being classified as a defensive shot.

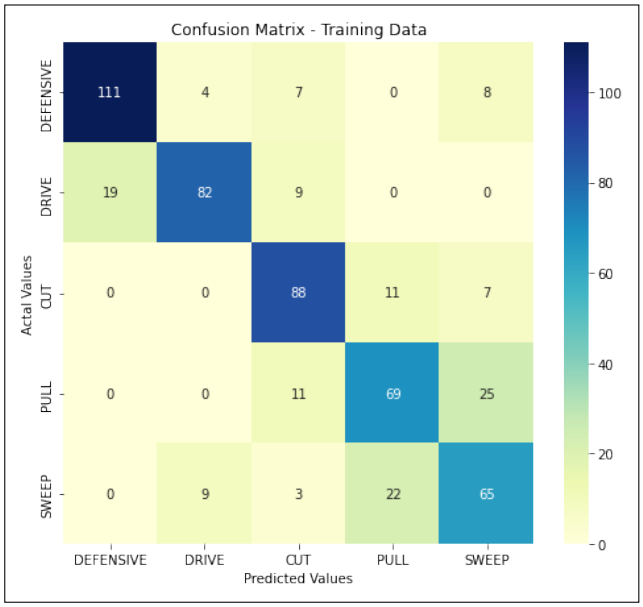

Figure 39: *Confusion Matrix (Testing Data - Algorithm 2)*

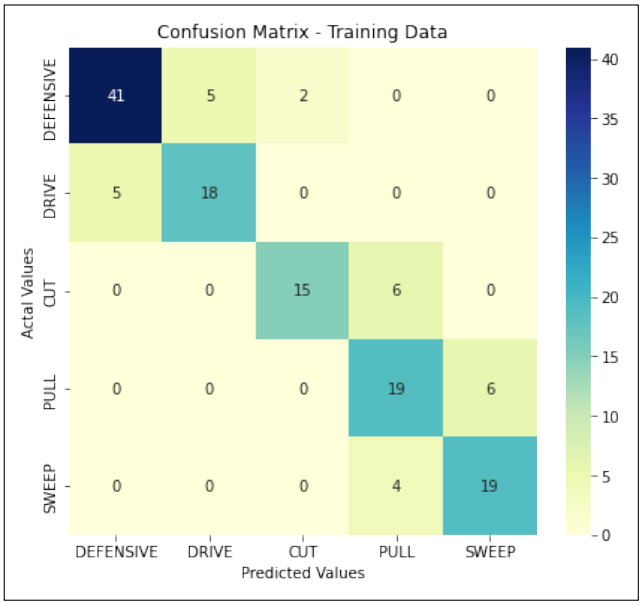

Figure 40: *Confusion Matrix (Testing Data - Algorithm 3)*

The drop in performance for the confusion matrices above highlights the algorithm's inability to generalise. This algorithm, as described in Section 5.4, was trained on sensor data on SensorLog, and tested using sensor data from the Xcode App. Whilst these are using the same sensors, any difference in the way the data is processed would give rise to an error. Additionally, it highlights the advantage of training the algorithm on features rather than the whole signal. The latter point is further highlighted by Figure 40. Whilst, due to time constraints, the testing on this algorithm was limited, it was also tested using data processed by the app, and so these errors would continue to show.

6 Performance Metrics

The sensor data collected, and the classification output, can both help evaluate various metrics of each shot. The performance parameters focused on this section are shot quality, hand orientation, bat angles and bat speed. These parameters will provide the basis for objective skill assessment, using data from the inertial sensors to assist the coach in giving actionable improvements to players. Hand orientation is the only metric that is currently being calculated prior to the classification, with the others using the classification output to improve the accuracy of the result.

6.1 Shot Quality

One metric that has been developed is the ability to give a quantitative measure of shot quality. The error metric used to evaluate the shot quality was the L_2 (least squares) error [11] that calculates the smallest sum of squares of errors, between each signal and the template signal for that class. This is shown by Equation 10, where x_i is the sensor value at a given timestamp and y_i denotes the template signal value at the same timestamp, n is 120 due to the 80Hz sample rate and the 1.5s length of each shot.

$$E_{2,shot} = \sum_{i=1}^{n} (x_1 - y_i)^2 \tag{10}$$

The least squares error was used over the L_2 norm as the errors will be normalised to a user friendly scale of 0-10 and so the process is unaffected by the square root and the factor of m. Using the $.csv$ files obtained from SensorLog, the least squares error was obtained for each of the 3 accelerometer and gyroscope signals in each shot. These values were then normalised between 0 and 10, with 10 being the 'perfect' shot and 0 being a bad shot. This normalisation process was done using Equation 11 where $E_{2,min}$ and $E_{2,min}$ are the minimum and maximum least squares error of the given sensor, of all the shots.

$$Quality = \frac{E_{2,shot} - E_{2,min}}{E_{2,max} - E_{2,min}} \cdot 10 \tag{11}$$

Averaging the 6 normalised scores for each shot, the average shot quality, and standard deviation, was computed for each class, shown in Table 6. The standard deviation was evaluated to be able to compare the testing data to previously calculated bounds.

For the subsequent data collected real time, this equation was modified into a conditional equation, shown in Equation 12. This conditional statement compares the error of each signal to the current maximum and minimum errors, and if a shot is worse, or better, than any shot in the app's history, that error becomes that new comparison value, with the quality

being 0 or 10. This conditional equation allows the normalisation to get better and more accurate with time.

Table 6: Shot Quality Averages

Class	Average Shot Quality (/10)	Standard Deviation (/10)
Defensive	6.03	2.01
Drive	6.70	2.61
Cut	6.97	2.76
Pull	7.27	2.61
Sweep	6.99	2.59

$$Quality = \begin{cases} 0, & \text{if } E_{2,shot} > E_{2,max}, \\ \dfrac{E_{2,shot} - E_{2,min}}{E_{2,max} - E_{2,min}} \cdot 10, & \text{if } E_{2,min} < E_{2,shot} < E_{2,max}, \\ 10, & E_{2,shot} < E_{2,min}. \end{cases} \tag{12}$$

One barrier to this is ensuring the templates are purely for the highest quality of shots. It is expected that a novice player will hit more inconsistent shots and generate a higher error. It can be seen that the higher the least squares error, the higher the variance from the template, and the lower the shot quality. The same method could also be evaluated for the difference between shot features and template features. Figure 41 shows the difference between an experienced player and a novice player when hitting a pull shot. It can be seen that the novice player snaps their wrists too late (peak is after the template) indicating they have swung too late. Looking into the difference in these signals has the potential to coach these more novice players.

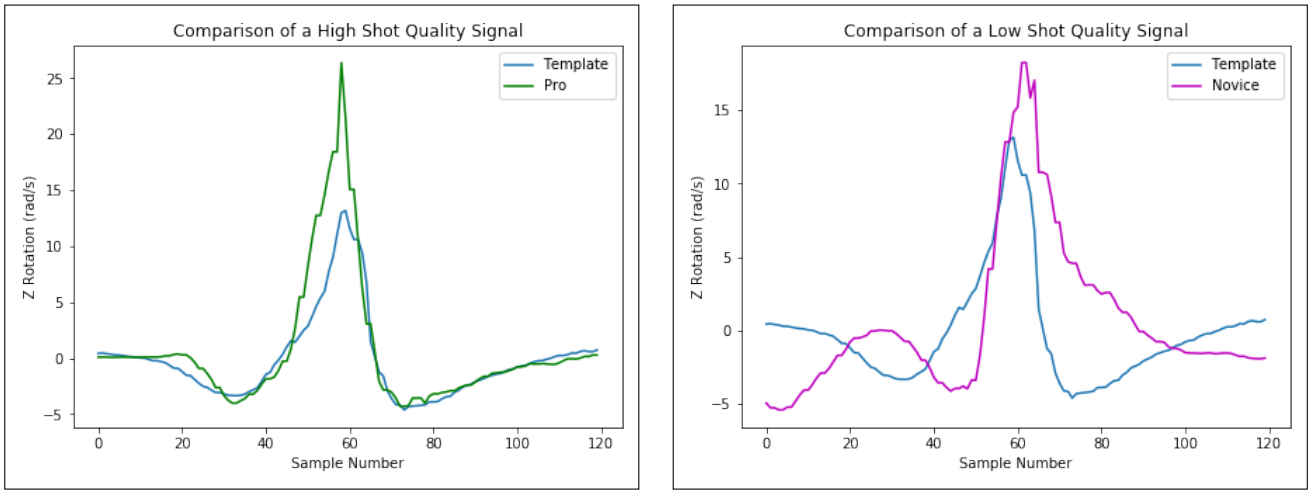

Figure 41: *Comparison of shot quality for experienced (left) and novice (right) players*

6.2 Hand Orientation

As mentioned in Section X, distinguishing between left handed and right handed players is crucial in being able to achieve a high classification accuracy. One method is to have the user manually set their hand orientation on the application user interface. However we can also use the free body diagrams in Figure 42 to infer this information automatically.

Figure 42: *Comparison of the back lift for left handed and right handed players [34]*

Figure 42 (left) shows that the back lift gravity vector is $(+ve, +ve, +ve)$ for a left handed batsmen, and figure 42 (right) shows $(-ve, +ve, +ve)$ for a right handed batsmen. When extending this analysis to all 10 players that assisted in the data collection, it was clear that the sign of the z-axis at back lift was very dependent on the player's technique so could not be of use. Additionally, the sign of the y-axis was also positive for for all players of both orientations. However it was evaluated that the sign of the x-axis at back lift was consistent between orientation and $+ve$ for a left handed batsmen and $-ve$ for a right handed batsmen. Once a shot is detected, the signs at the start of the signal could then be evaluated before running the classification algorithm. Table 7 summarises this for techniques with varying back lift angles, with 0° being when the bat is perpendicular to the floor.

Table 7: Stance signs for Hand Orientations

Gravity x Vector	Left Handed	Right Handed
x (45° back lift)	$+ve$	$-ve$
x (90° back lift)	$+ve$	$-ve$
x (135° back lift)	$+ve$	$-ve$

6.3 Bat Angles

Another input that is of importance is the orientation of the bat which enables us to determine bat angles which vary for different shots. This can be determined again using the position of the gravity vector [18]. Two main bat lift angles were determined. Firstly, the back lift angle, shown in Figure 42 above, which can be evaluated the same way for all shots. Secondly, the bat angle at impact will be evaluated - which is evaluated dependent on the shot class. We will also use the output from this section for the following section on bat speed, so it is important this calculation is situated between these two things.

6.3.1 Back Lift Angle

A higher back lift allows a batter to have a longer downswing angle and can be used to generate more power. The initial back lift angles of all 10 players that assisted in the data collection ranged from 47° to 151°, which is inclusive all nearly all techniques. These ′real′ values were obtained via video data to be able to compare to the values achieved by only the sensor signal. The initial back lift was detected by the y values of the gravity vector. As the angle is increased from 47° to 151°, the y component of the gravity vector will get weaker - moving from 1.0 to 0. Using the y value at the start of the 1.5s signal, the back initial bat lift angle can be inferred.

6.3.2 Impact Bat Angle

The impact angle is important in determining whether the ball will go up in the air or stay along the ground. A similar analysis to the back lift angle was undertaken, for each class using the free body diagram at impact for each player, and empirically matching the results to consistencies in the data. Table 8 shows the range of the impact angles for each class, the gravity vector range used to determine the impact angle and lastly what the reference plane is.

- Reference Frame 1 - The length of the bat is perpendicular the ground.
- Reference Frame 2 - The width of the bat perpendicular to the ground.

An example of the gravity vectors being used to evaluate the impact angle is shown in Figure 43. The value of the z component of the gravity vector at impact can be seen to be 0.63 - which equates to an average impact angle to be 74.9° for the cut shot. A similar analysis was undertaken for every shot to obtain the values in Table 8

Table 8: Impact Angle Values

Class	Gravity Vector Range	Impact Angle Range (°)	Reference Plane
Defensive	y component: 0.9-0.8	-10 to 45	1
Drive	y component: 0.9-0.8	-45 to 45	1
Cut	z component: 0.5-1.0	45 to 160	2
Pull	z component: 0.5-1.0	45 to 160	2
Sweep	z component: 0.5-1.0	45 to 160	2

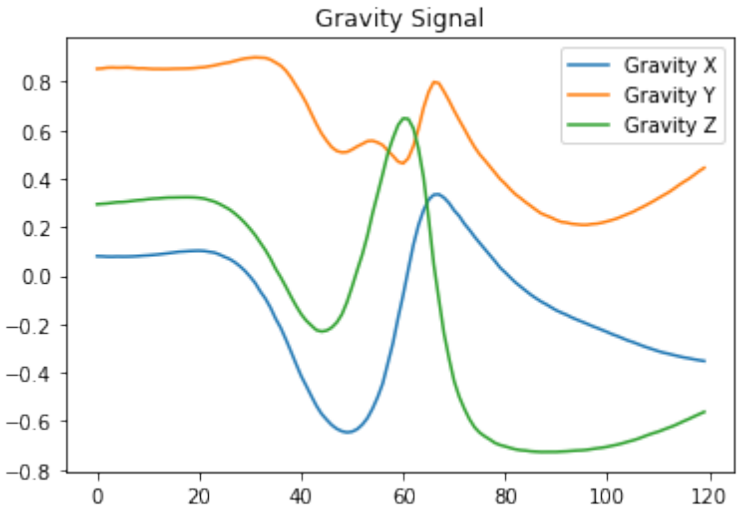

Figure 43: *Gravity Vector during a Cut Shot*

6.4 Bat Speed

Bat Speed is a crucial metric for skill assessment, with it being a key differences between average and elite players. The faster one can swing, the more time they have to obtain information about the delivery. Another reason for wanting to measure bat speed is that an increased bat speed will produce higher ball speeds, resulting in more runs for the batsmen. The bat speed was calculated as a measure of the forward velocity with which the bat impacts the ball, using Equation 13, where ω is the bat's angular velocity and R is the distance between the shoulder and the impact point.

$$V = R * \omega \tag{13}$$

Equation 14 gives the expression of the angular velocity, ω. The x-component has been omitted as it does not have an impact on the rotational speed of the bat.

$$\omega = \sqrt{g_y^2 + g_z^2} \tag{14}$$

The radius, R, can then be evaluated using visual experimental data and trigonometry [3].

The bat and arm are assumed to be rigid bodies, fixed about the shoulder as a reference point. The bat is also assumed to rotate with the same angular velocity as the wrist and the arm. Figure 44 shows a drive shot at impact and the subsequent free body diagram of the frame in shown in Figure 45.

Figure 44: *Drive at Impact [35]*

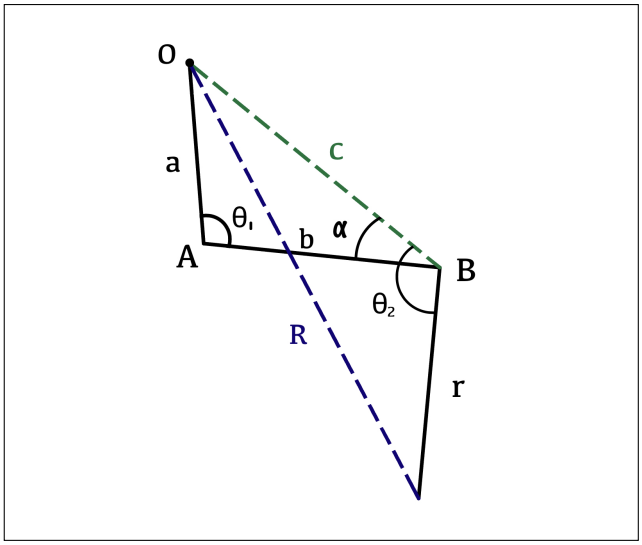

Figure 45: *Free body diagram of the drive*

θ_1 is the angle between the bicep and forearm, θ_2 is the angle between the wrist and bat. Using approximations for the bicep length, a, the forearm length, b, and the distance from the wrist to the impact point, r, the radius for the example in Figure 44 was calculated as 0.76m. Equations 15 - 17 shows the sine and cosine rules applied to the free body diagram to obtain this value.

$$c^2 = a^2 + b^2 - 2ab\cos\theta_1 \tag{15}$$

$$\alpha = \arctan\left(\frac{a\sin\theta_1}{b}\right) \tag{16}$$

$$R^2 = r^2 + c^2 - 2rc\cos\left(\theta_2 - \alpha\right) \tag{17}$$

However, ignoring the subtle changes that will occur in the impact point distance, r, the two angles, θ_1 and θ_2, will change significantly both within a shot but also between each shot. The angles changes within a shot (the initial angle to the impact angle, described in Section 5.2.2) have been neglected, with the bat speed simply being calculated using the radius and angular velocity at impact. Changes in θ_1 and θ_2 for different classes cannot be ignored however, with Table 9 showing the range of computed angles at impact for each shot class, with the subsequent radius value range, with the values being consistent for both orientations.

Table 9: Bat Speed Calculation Values (using $a = 0.25m$, $b = 0.3m$ & $r = 0.6m$)

Class	Angle between Bicep & Forearm, $\theta_1(^\circ)$	Angle between Wrist & Bat, $\theta_2(^\circ)$	Distance between impact & shoulder, $R(m)$
Defensive	110	90	0.769
Drive	110	90	0.769
Cut	90	180	0.814
Pull	135	180	0.807
Sweep	180	135	0.792

These ranges have been evaluated using visual experimental data, however using the bat lift angle at impact obtained using the method described the previous section, values for θ_1 and θ_2 can be inferred obtaining an accurate value for the radius for each shot. Using this method, bat speeds for the raw data set were evaluated, with Table 10 showing the average values. The standard deviation of these speeds was also evaluated using Equation 18.

$$\sigma_\omega = \sqrt{(\sigma^2)_{(g_y)} + (\sigma^2)_{(g_z)}} \tag{18}$$

Table 10: Bat Speed Numbers

Class	Linear Bat Speed (m/s)	Standard Deviation (m/s)
Defensive	4.21	2.14
Drive	9.11	4.24
Cut	13.22	6.74
Pull	17.39	6.21
Sweep	16.63	7.68

Table 10 shows the defensive shot to have the lowest value of bat speed, which was as expected since this is not a shot played with the objective of scoring runs. The cross bat strokes also had a higher linear bat speed than the straight bat drive, with the pull shot being the highest on average. This is due to the pull shot having an easier shot path to generate speed.

6.5 Testing and Evaluation

To test each of the performance metrics, the values obtained from the phase 2 data in real time, displayed on the radar chart were compared against the standard deviation bounds from the above performance metric sections above. Table 11 shows the percentage of this data that fell between these bounds.

Table 11: Performance Metric Evaluation

Metric	Percentage Accuracy
Bat Speed	96.2%
Shot Quality	74%%
Back Lift Angle	98.3%%
Impact Angle	86.1%

The metric with the highest accuracy was the bat speed - with shot quality being the most inaccurate. This suggests the comparison templates for the shot quality method are not truly representative of a 10 / 10 shot. Additionally, the impact angle's accuracy is lower due to the overall simplification of determining the impact angle based on one axis of the gravity vector. In reality, more parameters would have to be used for ovtain a higher accuracy.

7 Discussion and Findings

This section will be split into two main sections - discussing how the classification accuracy could be improved and then also how the performance metrics could be more accurate. Lastly, the future progression of the project will be discussed.

Based on the accuracy results from Section 5, the gaussian SVM appears to be the best algorithm for the use case. Table 12 shows the interrupt times between a shot being detected and a classification being given to a shot. Despite the worse performance, the neural network in Algorithm 2 has a lower value due to the an SVM's slowness during prediction, with comparison to a neural network. However both values are well under the between the 3 second limit for two consecutive shots, and so the SVM is still the preferable choice. However, limited testing was done and so thoroughly evaluating this third algorithm is needed to confirm this. There is also a small inaccuracy in the watch detecting a shot but not transferring any signal data to the phone. Algorithm 1 has not been included as the run time is not important with it not needing to meet the time limits.

Table 12: Classification Run Times

Algorithm	Overall Time
Algorithm 2	0.94s
Algorithm 3	1.32s

Whilst the right handed signals are being altered, this still gives a left handed bias and so a more vigorous method needs to be implemented - or by having two separate classification algorithms entirely. It is also still important to continually collect raw data - especially focusing on a greater range of abilities. Removing the false positives due to swinging and missing, which will occur more frequently for less experienced players, has also not been implemented, which would further improve the classification accuracy. One assumption is that all players will wear the watch on their dominant hand. This was assumed due to the fact that switching hands would result in the watch positioned on the wrist closest to the bowler which would significantly increase the chance of breaking the device.

Using the coach annotations implemented already in the iOS App UI, the algorithm accuracy could be improved by continually retraining the $.mlmodel$. Additionally, the coreML algorithms are currently set up to output a class every time - based on the highest probability weighting. An additional step could be such that the app displays a 6^{th} label of "N/A" if the highest probability from the SVM was below a certain threshold. Lastly, similarly to the adjustments required for the two hand orientations, inputting that into the algorithm as an 'annotation', other basic biometric info could be collected from the player, such as age and

height, as this could help improve algorithm accuracy by using these annotations to adjust the training weightings.

Of course, there are far more than 5 different cricket shots, and the classification accuracy's given in Section 5 are for training sessions when only these shots were hit. Therefore to increase the generalisation of the app to be used in any training session, more classes would need to be added. One example of increasing the number of classes would be to split the drive shot into three sub class - off drive, straight drive and on drive. Using these improvements, the app could be developed further to achieve the final project objective, an app that replicates a game during a training session and has the ability to provide improvements for players based off the signals.

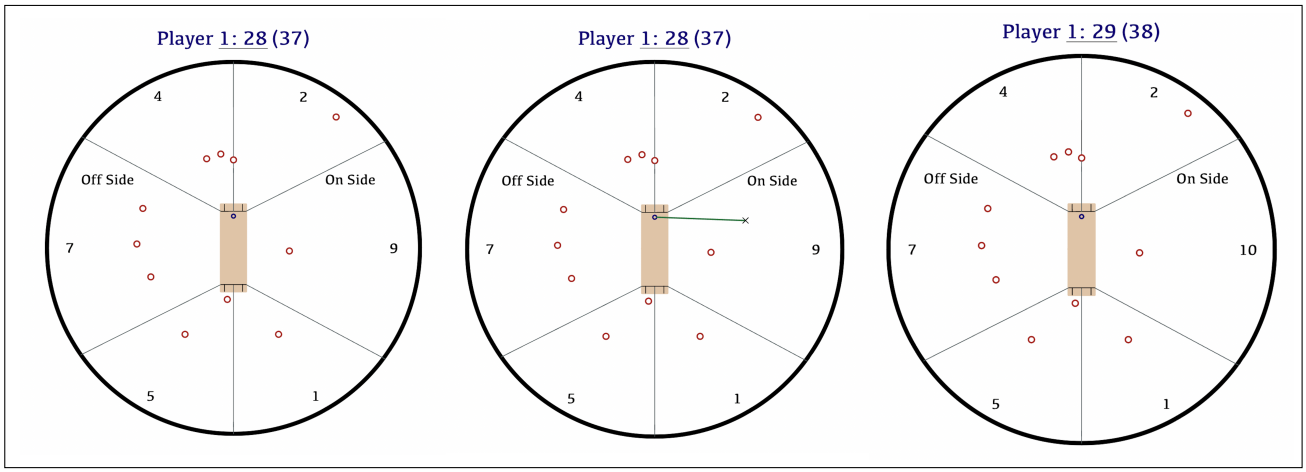

Figure 46: *Wagon-wheel Example for Gamification*

Gamification methods during sports coaching are also very effective methods for engaging players, retaining individuals towards improved athletic performance and simulating match scenarios in a training session. Improving on and implementing the metrics discussed would allow the app to introduce a gamification feature, with an example shown in Figure 46. Thr schematic shows a flowchart of how the UI would update if a shot was detected and projected. The 2D pitch has an interactive field that the coach can set (red circles), and then when a shot is detected, a projectile for each shot is mapped onto the pitch and a number of runs is evaluated based off the x, y co-ordinates of the closest fielders.

8 Conclusion

The goal of the project was to develop a sensor based framework to classify cricket batting shots, along with other performance metrics in real-time, using gyroscope and accelerometer sensors - so that a player can assess their own performance.

The current state of the coaching tool is such that a live iOS App and Watch OS App have been developed that can monitor this inertial sensor data in real time. The app is able to detect the shot, segment the 1.5s signals, evaluate features from the signal, run the signals through a classification algorithm and evaluate performance parameters all with an average prediction time of 0.94 seconds. The classification and performance parameters evaluated, bat speed, shot quality and bat angles, can now provide the base for objective skill assessment for the app to be used as a coaching tool.

The performance of the classification of the post-processed data was very accurate, with an overall accuracy of 86.8% (93.5% for detection and 92.8% for classification). The two classification algorithms used within the app, the neural network (NN) algorithm and also the gaussian SVM algorithm, saw lower detection accuracy's of 68.6% (90% for detection and 76.2% for classification) for the NN and 79.1% (90% for detection and 87.9% for classification) for the gaussian SVM.

The future work for the project involves improving the classification accuracy, removing false positives from the detection, increasing the algorithm's generalisation for beginners and implementing and a greater number of shots into the classification. Although further work remains to be done to get to a live app for general user consumption, cricket shot classification using sensory motion data can provide a strong foundation for objective skill assessment.

9 References

[1] CricViz. *USING DATA TO IMPROVE DECISION-MAKING AND ENHANCE STORY TELLING.* 2022. URL: https://www.cricviz.com/about/.

[2] World Population Review. *Most Popular Sport by Country 2022*. 2022. URL: https://worldpopulationreview.com/country-rankings/most-popular-sport-by-country.

[3] Disha Panchal & Ravi Vaidyanathan. *MOTION TRACKING DEVICE FOR TENNIS*. 2018. URL: https://www.theplayerstribune.com/en-us/articles-us-open-tennis.

[4] Abeer Mostafa et al. Multi-sensor gait analysis for gender recognition. SciTePress, 2020, pp. 629–636. ISBN: 9789897584428. DOI: 10.5220/0009792006290636.

[5] P Yadav & I S Pal. KINEMATIC ANALYSIS OF CUT SHOT, HOOK SHOT AND SQUARE CUT SHOT IN CRICKET. *Vidyabharati International Interdisciplinary Research Journal* 12 (2 2021). ISSN: 2319-4979. URL: www.viirj.org.

[6] Luke. *What Are The Different Types Of Cricket Shots?* 2021. URL: https://cricketershub.com/types-of-cricket-shots/.

[7] A. Busch & D. A. James. Analysis of cricket shots using inertial sensors. 2008, pp. 317–322. ISBN: 9780415456951. DOI: 10.1201/9781439828427.ch45.

[8] Ajay Krishno Sarker. Bat Swing Analysis in Cricket (2014). DOI: 10.25904/1912/476. URL: http://hdl.handle.net/10072/367481.

[9] Vishnu Sarpeshkar & David L. Mann. *Biomechanics and visual-motor control: How it has, is, and will be used to reveal the secrets of hitting a cricket ball*. 2011. DOI: 10.1080/14763141.2011.629207.

[10] Sidath Asiri. Machine Learning Classifiers (2018). URL: https://towardsdatascience.com/machine-learning-classifiers-a5cc4e1b0623.

[11] Peter Huthwaite. *Machine Learning (2021-22)*. 2021.

[12] Anshul Saini. *Support Vector Machine(SVM): A Complete guide for beginners*. 2021. URL: https://www.analyticsvidhya.com/blog/2021/10/support-vector-machinessvm-a-complete-guide-for-beginners/.

[13] Gabriele De Luca. *Advantages and Disadvantages of Neural Networks Against SVMs*. 2021. URL: https://www.baeldung.com/cs/ml-ann-vs-svm.

[14] Luis Bermudez. Overview of Neural Networks (2017). URL: https://medium.com/machinevision/overview-of-neural-networks-b86ce02ea3d1.

[15] SmartCricket Team. *BATSENSE: THE GAME CHANGER*. 2021. URL: https://smartcricket.com/.

[16] Sooryanarayanan Balasubramanian. *How Cricket Bat Sensor is Set to Complement Video Analysis in Coaching*. 2021. URL: https://spektacom.com/blog/how-cricket-bat-sensor-is-set-to-complement-video-analysis-in-coaching/.

[17] Aftab Khan, James Nicholson & Thomas Plötz. Activity Recognition for Quality Assessment of Batting Shots in Cricket using a Hierarchical Representation. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies* 1 (3 Sept. 2017), pp. 1–31. ISSN: 2474-9567. DOI: 10.1145/3130927. URL: https://dl.acm.org/doi/pdf/10.1145/3130927.

[18] Ashish Sharma et al. CommBox: Utilizing sensors for real-time cricket shot identification and commentary generation. Institute of Electrical and Electronics Engineers Inc., June 2017, pp. 427–428. ISBN: 9781509042500. DOI: 10.1109/COMSNETS.2017.7945426. URL: https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=7945426.

[19] Ammarah Farooq et al. *Deep CNN based Data-driven Recognition of Cricket Batting Shots*. IEEE, 2018. ISBN: 9781538654606.

[20] Brandon Chester. *he Apple Watch Series 2 Review: Building Towards Maturity*. 2016. URL: https://www.anandtech.com/show/10896/the-apple-watch-series-2-review/6.

[21] Apple. About Bluetooth, Wi-Fi and cellular on your Apple Watch (2022). URL: https://support.apple.com/en-gb/HT204562#:~:text=Your%5C%20Apple%5C%20Watch%5C%20uses%5C%20Bluetooth,Apple%5C%20Watch%5C%20uses%5C%20Wi%5C%2DFi..

[22] Meraj Molla. *Train your own ML model using Scikit and use in iOS app with CoreML (and probably with Augmented Reality)*. 2016. URL: https://itnext.io/train-your-own-ml-model-using-scikit-and-use-in-ios-app-with-coreml-and-probably-with-augmented-99928a3757ad.

[23] Bernd Thomas. SensorLog v4.0 (2021). URL: http://sensorlog.berndthomas.net/.

[24] Apple. Core Motion (2022). URL: https://developer.apple.com/documentation/coremotion.

[25] Eric Hsiao. *Introduction to Apple WatchKit with Core Motion — Tracking Jumping Jacks*. 2018. URL: https://heartbeat.comet.ml/introduction-to-apple-watchkit-with-core-motion-tracking-jumping-jacks-259ee80d1210.

[26] Apple. HealthKit (2022). URL: https://developer.apple.com/documentation/healthkit.

[27] Ravi Vaidyanathan et al. Tongue-movement communication and control concept for hands-free human-machine interfaces. *IEEE Transactions on Systems, Man, and Cybernetics Part A:Systems and Humans* 37 (4 July 2007), pp. 533–546. ISSN: 10834427. DOI: 10.1109/TSMCA.2007.897919.

[28] Aylin Tokuç. Why Feature Scaling in SVM? (2021). URL: https://www.baeldung.com/cs/svm-feature-scaling.

[29] Jason Brownlee. Principal Component Analysis for Dimensionality Reduction in Python (2020). URL: https://machinelearningmastery.com/principal-components-analysis-for-dimensionality-reduction-in-python/.

[30] Jonny Evans. Why Apple's Create ML matters to your enterprise (2018). URL: https://www.computerworld.com/article/3293446/why-apple-s-create-ml-matters-to-your-enterprise.html#:~:text=The%5C%20difference%5C%20between%5C%20the%5C%20two,such%5C%20as%5C%20inside%5C%20TensorFlow)%5C%20aboard..

[31] Maxim Skorynin. *Object Detection with Create ML*. 2019. URL: https://evilmartians.com/chronicles/object-detection-with-create-ml-images-and-dataset?hmsr=joyk.com&utm_source=joyk.com&utm_medium=referral.

[32] Tyler Hutcherson. Activity Classification for watchOS: Part 2 (2019). URL: https://medium.com/@tyler.hutcherson/activity-classification-for-watchos-part-2-1011ee5e75d5.

[33] Gerardo Lopez Falcón. *Creating an IOS app with Core ML from scratch!* 2017. URL: https://towardsdatascience.com/creating-an-ios-app-with-core-ml-from-scratch-b9e13e8af9cb.

[34] Luke. *What Is A Backlift In Cricket? – Full Explanation*. 2021.

[35] Ben Horne. 'Smash Factor': New technology to unlock secrets of batting (2019). URL: https://www.foxsports.com.au/cricket/smash-factor-new-technology-to-unlock-secrets-of-batting/news-story/45aff1a5d99797fa5d0a50a5407c22bc.

10 Appendix A - Video Demonstration of App

A video of the app being used in a training session can be viewed here. In the 10 shot video shown, the app displayed on screen detects 7/10 shots correctly.

11 Appendix B - GitHub Repositories

Source Control was used extensively throughout the project to be able to track changes and create verison releases. The GitHub repositories for both the Python post-processing code and the Xcode project can be found by clicking the links below.

- Python Repository
- Xcode Repository

The GitHub *README.md* files also provides a detailed but simple breakdown of each, to ensure the project is well documented so an individual that views the project without this report to hand, can still understand the functionality and workflow of the code.