Free Throw Shot Video Analyser

Jadon Omo-Adesanya

Supervised by Matthias Englert

3rd Year of Study

University of Warwick, Department of Computer Science

2020-2021

Abstract

Athletes and trainers have an existing co-dependant relationship. This report follows from a project investigating how professional player training can be achieved independently, as well as how player performance can be refined through the use of technology. More specifically, the project entails the development of a web application conducting a video analysis of a user's free throw shot. The application is made to be utilised for the improvement of said user's ability to shoot free throws. This report includes discussions and reviews on the methods used to achieve this video analysis through each feature on the subsequent web application. Although the video analysis proved to be successful in improving users free throw shooting, we can conclude this is only to an extent. Beginners to the sport of basketball show much more improvement after use of the application, compared to advanced users that have several years of experience playing the game. This suggests that the more experienced the user, the less of an impact the application has in the improvement of their ability to shoot free throws. After review of the application, professional player training through the use of this technology fails to replicate or improve upon the effect of player training using a real trainer. To achieve the same affect, the application would have much more time, resources and effort dedicated towards it in order to compete with its biological counterpart.

Keywords List: Free Throw, Basketball, Training, Computer Vision, Neural Networks, Convolutional Neural Network, Django, Video Analysis, Machine learning, Python, Web Application

Contents

List of Figures

1 Introduction

The professionally played game of basketball has seen a drastic change to the sport since its inception in 1946. The bulk of these changes stem from rules and regulations, however, since sporting technology became inaugurated into the game it has seen a new kind of development. Athletes stay safer using improved equipment. Scouting and recruiting becomes more dependable with the potential of a player being more easily realised through the use of relevant metrics. Understanding the game itself using previous performance data and forecasting discloses a new level of play. Analytics has become a pivotal component in the sport of basketball in many different ways. This report is fixated on the training aspect of the game, and how we can use analytics for individual player development. The commercialisation of sport has caused organisations and individual athletes to have much greater incentives, making it increasingly common for professionals to put extra effort and hours into their training to gain an upper hand against their opposition. In the past, this was achieved using improved training techniques and mitigating injury. During the last decade, technology revolutionised sports training by assisting professionals in perfecting their mechanics and movements, tracking live performance, and further mitigating their injuries. Improving performance has transferred focus on how data is collected and utilised, both in training and organised games. AI and machine learning applications for sports organisations have become increasingly useful; providing insights, statistics, analytics, and catering to decision making.

During training, athletes have access to wearable sensors that convey real-time information to a trainer's tablet or can use GPS to accurately pinpoint motion. This can be incredibly beneficial in the athlete's development. However, these technologies belong to the organisation meaning they can only be used when training with the organisation's trainers and such. If an athlete decides to train by themselves or from home, which has become more common at present, they cannot use these technologies. The project aims to provide readily available technology to a niche of athletes that can be used at their luxury, essentially reducing the need

to physically be with an organisation's trainers or in their facilities by still having access to somewhat high-level equipment.

This report follows the completion of a project focused on developing an application that is used to help improve a user's performance in shooting free throw shots. The application centres around providing useful insights and a breakdown of a user's free throw shot using video analysis. The ultimate goal of the project is to help users improve their free throw shot percentage using the feedback and analysis generated by the application.

1.1 Background

Whether playing basketball as a professional athlete, in organised competition, or for leisure the free throw shot is an essential part of the game. Although the shot seems simple enough many people, including professionals, struggle to make them. During the 2019-2020 NBA regular season, Dwight Howard made a league-wise low 51.4% of his free throws [1]. As a team, the New York Knicks made only 69.4% of their free throws during the same season [2]. Comparing these percentages to the regular season's team average of 77.3%, we can assume even individuals competing on the highest level of play yearn to increase their free throw percentages. Although there are no statistics available for low/leisure level competitive competition, we can also assume the numbers are considerably lower than professionals, showing the desire to improve on this aspect of the game on all levels.

There are several reasons an athlete may wish to train independently. One of the most common is the desire to put in more hours of training outside of the hours of practice with the team. Athletes may also not have direct access to training facilities, especially during off-seasons whether it is due to frequent traveling or national holidays. Other than the desire to train independently, the past year has shown us that working independently may become a necessity when training with

others is not possible. During training athletes often shoot a prolific number of free throws. Keeping track of the number of shots missed and made becomes increasingly difficult with the number of shots taken, especially when the athlete is training by themselves. Many factors are involved in the shot mechanics of a free throw. For example, the maximum height of the ball, the follow-through, and the release angle. Adjusting these factors in an individual's shot is difficult, especially when the optimal shot mechanic values are unknown. This becomes more difficult when an athlete is training by themselves. The application intends to find a solution to these problems.

1.2 Motivational Material

Research supports the feasibility of the project as related work has previously been developed. Markowitz developed a system achieving video analysis of serves in the sport of tennis using an Auto ML vision model for object detection, as well as Google Cloud's video intelligence API for pose estimation [3]. This system is an adaptation of Akil's work, previously developing a similar system analysing penalty kicks in the sport of football. Both these systems can be adapted to give the desired results for this project; to achieve video analysis of a free throw shot. Although their work proves to be successful, a model is developed by hand as opposed to using an Auto ML vision model. This is so that more developmental control is available over each component in order to adhere to the requirements of this project and to accommodate change. Multiple other projects and systems [4-7] perform a type of analysis of the free throw shot using a variety of methods. These provide continued support toward the feasibility of this project, showing an analysis of the shot is more than possible.

Two applications further motivate the work of the project. The first, SwingVision, an iOS application providing real-time video analysis and coaching for tennis available on a monthly subscription. Swing Vision can be considered as a more

complete and advanced version of Markowitz's work. Not only does the application analyse tennis serves but provides automated shot tracking which includes the computation and feedback of stroke type, ball speed, shot placement, rally length, footwork, and posture as well as several other features. The second, HomeCourt, is an interactive basketball application also available on iOS. HomeCourt is essentially a basketball personal trainer, capturing user performance and providing guided feedback. Again, this is an application that is far more advanced than previously mentioned work as it gives instant feedback for real-time interactive workouts, however, this is expected considering an official partnership with the NBA. These related systems encourage the development of the project as it shows a free throw shot video analyser can be made as a simpler or similar form of existing work.

2 System Design

The final collective deliverable for this project is a web application equipped with the underlying functionality that aids the user in improving their free throw shooting. Although these functions and models may increase in complexity, the system should be relatively simple to use. The figure below shows a use case diagram representing how the system is intended to interact with users. As is recognisable, the system is limited to two or three main website functions due to be on the application. The first, 'Analyse Video', can be considered as the fundamental application function where users call the underlying functions and models to complete their video analysis. 'Show Feedback' follows the previous function where users can view the results of their video analysis. Lastly, 'Download PDF' is considered as an extension of 'Show Feedback' where users can keep a copy of their results.

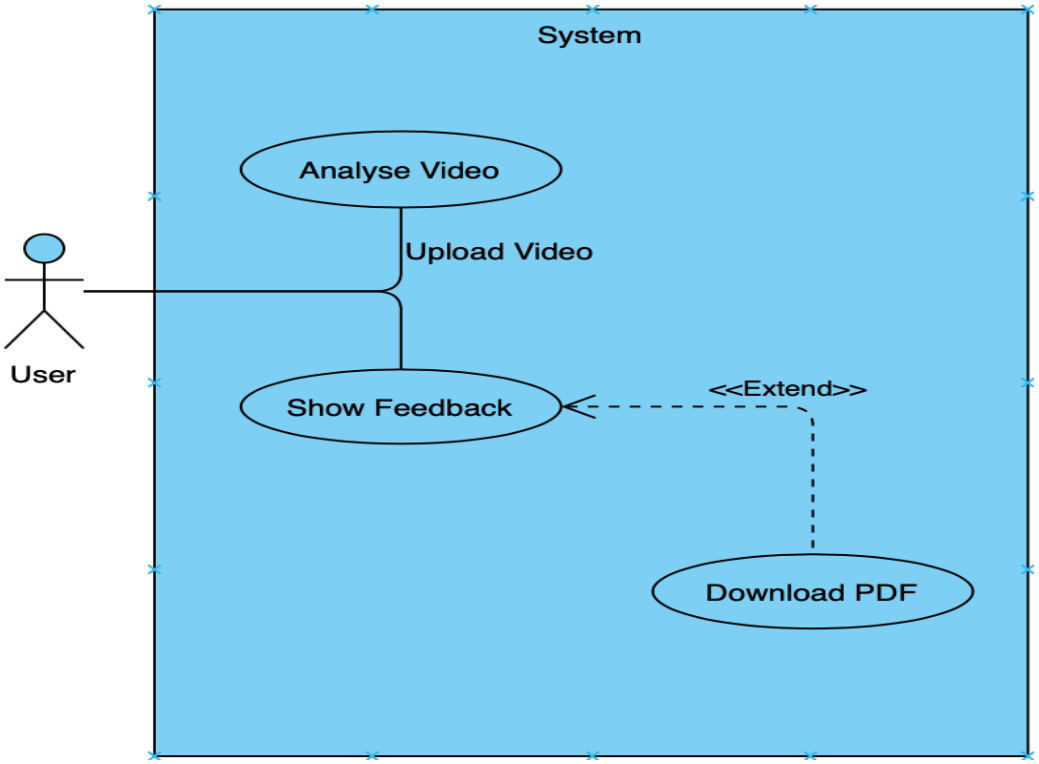


Figure 2.1: Use Case Diagram

The figure below illustrates a work breakdown structure detailing the activities to be completed through each deliverable in order to achieve a successful project. The project has four main deliverables whose completion will determine its overall success. The last deliverable is the combination of extension tasks whose completion will add value to the project but will not be considered heavily in the determination of the projects success.

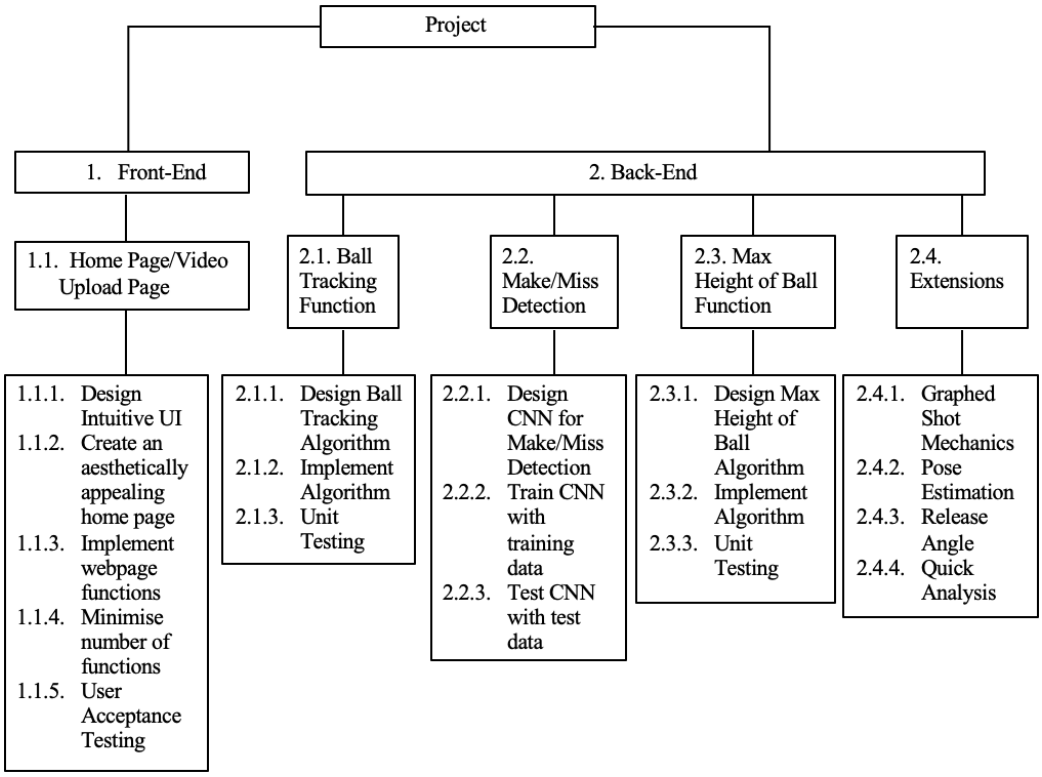


Figure 2.2: Work Breakdown Structure

2.1 Project Requirements

The overarching aim of the project is to establish a platform for athletes to train and improve their free throw shooting without the help of a real athletic trainer. Completing the objective of the project is induced by a web application that

allows users to upload videos of themselves shooting free throw shots followed by an analysis-type breakdown. Machine learning techniques are to be deployed which recognise when a shot has been missed or made while tracking the number of missed/made/taken shots. Additionally, the system should also provide some useful insights and statistics about the mechanics of the individuals shot.

These goals can be more easily managed by breaking them down into bite-sized requirements. Requirements are classified as functional or non-functional, and further classed as core or extension below. Core requirements need to be implemented successfully for the overall success of the project. Extension requirements are not absolute however will benefit the project and improve the overall solution. All non-functional objectives are also core requirements.

Functional:

Core:

C1 The user should be able to upload recorded videos as input files on the web application

C2 The system should be able to correctly track the basketball in videos

C3 The system should identify when a shot has been missed or made

C4 The number of made/missed/taken shots should be correctly tracked after the analysis of a video

C5 The system should provide a shot make percentage (percentage of made free throw shots)

C6 The maximum height of the ball during the shot should be given

C7 The user should be able to download a pdf file of a report of their shot analysis

Extension:

E1 The system should generate a type of graph of the user's shot mechanics

E2 A pose estimation of the user shooting the free throw should be recorded

E3 The system should calculate the user's release angle

E4 The system should generate a quick analysis

Non-Functional:

NF1 The web application should be made simplistic as there are not many complicated functions

NF2 The web application should be very easy to use

NF3 The web application should be able to be used intuitively so first time users can navigate through the web pages easily

NF4 The web application should have low latency

NF5 Users should find the web application aesthetically appealing

2.2 Project Management

We can construct a many-to-one mapping of functional requirements to the features to be developed and delivered. This allows us to manage the completion of each requirement more efficiently as through each feature, the appropriate functional requirements can be fixated on to ensure it is successfully delivered. Figure 2.3 illustrates this many-to-one mapping, listing functional requirements and their corresponding feature where they will be considered heavily in the feature's development. The non-functional requirements are not included in the mapping as they are all in relation to the web application, therefore during its development, all non-functional requirements will be emphasised.

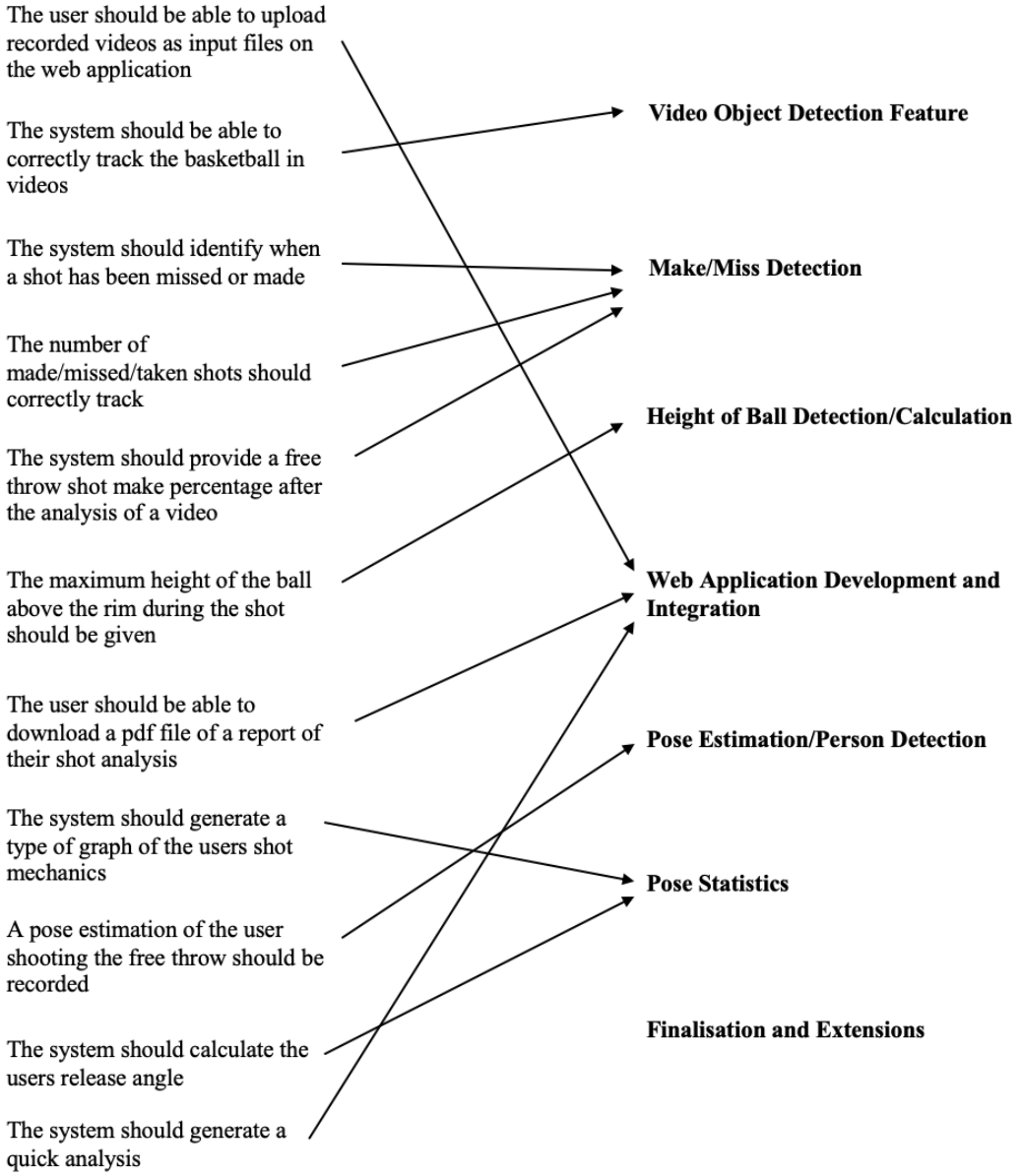
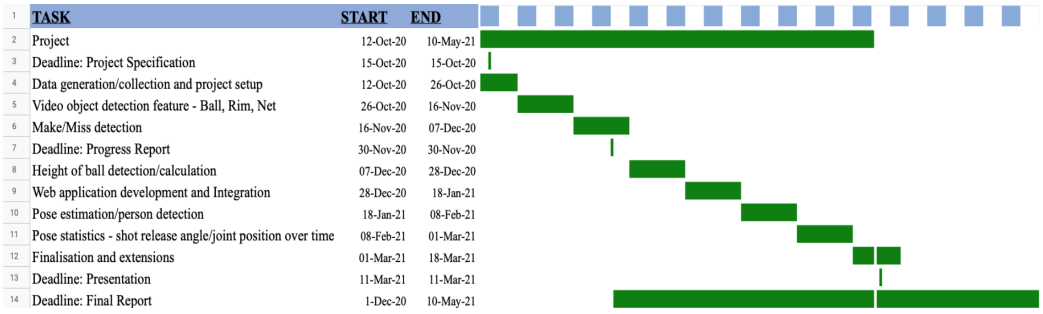


Figure 2.3: Requirements Mapping

Figure 2.4 below shows a Gantt chart detailing the feature or task to be completed at each time stage during the project's development. The chart also includes the

relevant documents to be delivered. The pose estimation and pose statistics features are listed as extension tasks in the project requirements section above, however in the Gantt chart they are listed independently from extensions. This is because, while considering the duration of the whole project, there is time remaining for the completion of two extension tasks, assuming each task is completed according to schedule. Pose estimation and pose statistics take the allocated time remaining as these are the desired extensions to be completed before the others.



	TASK	START	END
2	Project	12-Oct-20	10-May-21
3	Deadline: Project Specification	15-Oct-20	15-Oct-20
4	Data generation/collection and project setup	12-Oct-20	26-Oct-20
5	Video object detection feature - Ball, Rim, Net	26-Oct-20	16-Nov-20
6	Make/Miss detection	16-Nov-20	07-Dec-20
7	Deadline: Progress Report	30-Nov-20	30-Nov-20
8	Height of ball detection/calculation	07-Dec-20	28-Dec-20
9	Web application development and Integration	28-Dec-20	18-Jan-21
10	Pose estimation/person detection	18-Jan-21	08-Feb-21
11	Pose statistics - shot release angle/joint position over time	08-Feb-21	01-Mar-21
12	Finalisation and extensions	01-Mar-21	18-Mar-21
13	Deadline: Presentation	11-Mar-21	11-Mar-21
14	Deadline: Final Report	1-Dec-20	10-May-21

Figure 2.4: Gantt Chart

An agile method of development is most appropriate for this project to accommodate for adaptations and changes. Initial ideas and designs may be adjusted during development to favour the feasibility and completion of the project given the amount of time. Agile methods tend to favour projects with a fixed amount of time and a planned but estimated scope, also making it a convenient development strategy for this project. More specifically, an iterative Feature-Driven Development(FDD) methodology will be deployed with each iteration lasting 2-3 weeks, as apparent in the Gantt chart above.

Feature-Driven Development is customer-centric, and incremental, with the aim of delivering tangible software results often and efficiently. Like the name suggests, software development is organised around making progress on features or tasks where each feature is planned, designed, and built through each cycle. A new feature should be added to the system with each successful iteration cycle where

possible, giving the advantage of a rapid development process. Some features of the system are more complex and difficult than others therefore the iteration cycle length may increase with the complexity of a new feature. At the start of each cycle the development of each feature will be planned to be completed accordingly but may succumb to unavoidable and unforeseen changes.

2.3 Legal, Social, Ethical and Professional Issues

The project involves collecting data in the form of videos of participants shooting free throws. Due to legal reasons, only videos of myself in the recording of said videos of participants shooting free throws will be used. This is to avoid any legal implications and make the process of data collection easier. Participants other than myself are made aware that their recordings will no longer be used for the purpose of this project and are to be withdrawn and deleted. Development of the project is to be completed individually so there are no professional issues regarding collaboration.

There are no apparent social or ethical issues related to this project. All resources are free to use and open-source software. Friends and family will be included in some of the testing of the system however no issues should arise. This project exists for the purpose of a university third year project/dissertation therefore results and findings are not shared publicly, avoiding further issues. Other previous work used to assist in the development of this project are acknowledged and referenced.

3 Benchmark Performance Testing

To aid in the evaluation of the project, benchmark performance testing of anonymous participants are undertaken. Participants remained anonymous to protect the privacy of the volunteers agreeing to take part in the project, regardless of the fact that their data will not be publicly shared or processed in any way. The objective of these benchmark performance tests is to record a pre-application-use performance level in the shooting of a fixed number of free throws. This provides the first step in the investigation of whether a participants free throw shooting improves, after use of the application generated from this project. Following the completion and use of the application, the same participants will shoot the same number of free throws. Again, each participants performance level will be recorded and later compared to determine if there are any conclusive results regarding the application's significance in the improvement of a user's free throw shooting post-use. This directly stems from the objectives of the project as a whole. Participants are asked to return to the same facility and make an effort to not practice their free throw shooting in an attempt to keep the number of independent variables to a minimum.

The benchmark performance testing consists of each participant shooting 25 consecutive free throw shots. Another volunteer rebounds and passes the participant the ball following each shot so they do not move from the free throw line. The number of made shots are tracked and recorded along with the number of shots taken. This allows the calculation of a make percentage, calculated as $\frac{\#\,of\ made\ shots}{\#\,of\ shots\ taken\,(25)} \times 100$, which represents the respective participants performance level.

To provide a rational measure of ability, an experience level chart shown below in figure 3.1 is designed to assist in labelling participants by an appropriate ability level. The experience level chart is based on the number of years of experience a participant has collectively and actively accumulated in the sport of basketball. Before the gathering of results, each participant is asked how many collective years they have actively been playing basketball for. Although this may not be the most

accurate method of measuring the ability of participants, it will suffice for the purpose of this project as it gauges a good estimation. Labelling participants with ability levels is used to compare how the application affects users with different levels of experience. This lets us determine if the amount of experience of a user has any significance in the application's own ability to improve said user's free throw shooting.

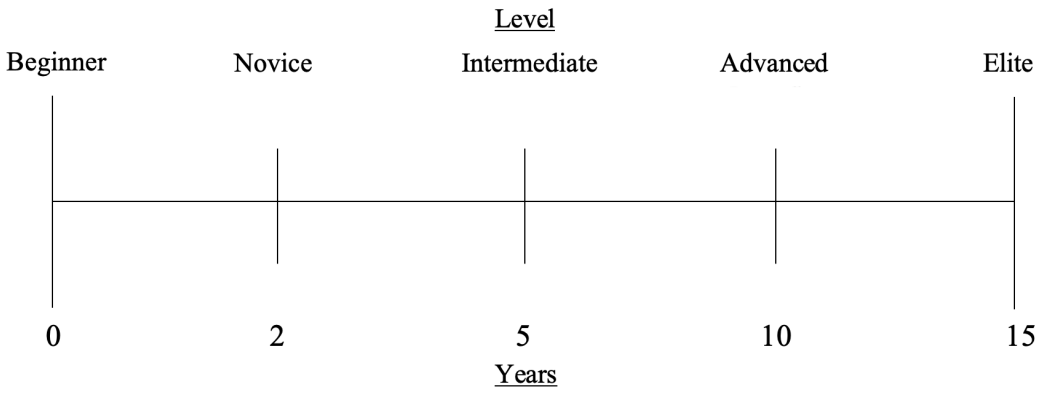


Figure 3.1: Experience Level Chart

It was intended to gather several more participants for benchmark performance testing and subsequently post-application-use performance testing. Unfortunately, due to the implications surrounding COVID-19 and the recent nation-wide lock-downs, it became increasingly difficult to gain access to basketball court facilities and to safely meet with others. Fortunately, upon the initial recording of data (discussed in section 4) two participants were readily available to take part in the benchmark performance testing. No further participants were able to take part in the testing due to the previously mentioned implications. The results are shown in the table below.

Participant	Ability Level	Performance
A	Advanced	72%
B	Beginner	40%

During the recording of results, the last five of the participant's twenty-five shots will be used as input for the application. This is so the application is not overloaded with an extensive video length as we do not yet know its capability. Five shots should be enough for the application to generate a useful enough analysis for participants to attempt to improve their shooting.

4 Data Generation

The data required to build or train machine learning and computer vision models are video samples of free throw shots being taken, both missed and made. This is quite specific data therefore there is no dataset that is publicly available on dataset finders like Kaggle. As a result, the data is required to be generated by hand. This task is the only section of the project which is to be completed physically, therefore there is not much technical aspect to it. The resources required to generate these data are:

- Basketball Court Facility
- Tripod Stand
- Camera
- Basketball

The basketball court is rented from a local sports centre for £22 and the tripod stand purchased from amazon for £15. The camera used is an iPhone 8 phone camera which was readily available, so no further expenses were required to purchase a camera. A molten basketball was also readily available, which did not further the cost. The total cost of data generation and collection is £37. This is essentially the total cost of the project as all expenses are predicted to be derived from data generation because of the required resources.

During the process of data generation, a side-on angle was used so the video samples capture a good view of the shooter, ball and basket. The basketball court facility is rented for one hour. During this time over 300 shots are taken and are all recorded. The type of shot is also varied to improve the data's diversity thus making the subsequent models more robust. These types of shots include high arcing shots, bank shots (where the ball hits the backboard before going into the net), and air-balls (where the ball misses and does not touch the rim). The air-ball is of particular interest as this shot can commonly be mistaken as a made shot rather than a bad miss, even by the eye. The ball may travel left or right of the rim

giving the illusion that the ball has gone straight through the net while observing from a side angle. In reality it has completely missed the target therefore should be classified as a miss by any subsequent model.

Again, due to the implications surrounding COVID-19 and the recent nation-wide lockdowns, only on this day were facilities available for booking before restrictions became enforced. It was planned to gather much more data at different facilities to improve data's diversity, however this was not possible and the project will have to progress and make do with the amount of data already gathered. This task was allocated two weeks for completion. The data generation process begun in the second week because it could only begin once the tripod stand was made available. The remaining time left for this task was dedicated to the project setup which involved creating a git repository, downloading relevant libraries, and file organisation.

5 Object Detection

Both machine learning and computer vision tools are considered for use in the development of this feature. It is finally decided that computer vision tools are best suited for development due to its ease of use and simplicity which helps adhere to the timeline of the project. Using machine learning to detect if the ball is in the image or not would be a more simple task as it could be achieved by utilising a neural network. Tracking and identifying the location of the ball, on the other hand, may prove to be much more difficult using machine learning compared to computer vision techniques. This gives rise to unforeseen problems which are best to avoid. The Gantt chart (figure 2.4) in section 2 states the object detection feature is for the ball, net, and rim. However, only detection for the ball is required to successfully complete the functional objectives, to track the ball, also stated in section 2. The net and rim are essentially detected in the next feature.

Initial research shows an article detailing a program which utilises the OpenCV computer vision library to develop a tennis ball tracker [8]. This is similar to the objectives of the object detection feature for this project but replacing a tennis ball with a basketball. The likeness between the objectives of both projects induces the articles use as reference. The key to this feature is detecting the ball using its colour. The ball used is an official molten game ball therefore the majority of people will have access to it. Most balls are also of a similar colour so using the colour of the ball is appropriate. The upper and lower bounds of this colour are defined in HSV colour space (Hue, Saturation, Value) as it allows the intensity of the colour to also become bounded without altering colour information unlike BGR colour space, which is the colour space of the original video. HSV is one of many colour spaces that make the distinction between colour and intensity, however implementation using HSV colour space is often easier as library functions for converting between BGR and HSV are more widely available. The object detection feature is intended to be used for video inputs. To make this possible the video must be decomposed into each of its constituent frames using a videoCapture object, making a set of images/frames. Each of these images have the same operations performed on them

in order to detect the ball through each frame of the video.

The image first needs to be blurred to remove the image's background noise. This is achieved by convolving the image with a low-pass filter kernel (see section 6.1 for more details on the convolution method). More specifically, blurring the image removes high frequency content like edges. In the case of the images in our dataset, the lines between each of the wood panels and each brick on the walls will be blurred as they appear frequently in the image. The *GaussianBlur()* OpenCV function is used to achieve this image blurring. In this method, an 11 x 11 Gaussian kernel is used along with a standard deviation (sigma), automatically calculated from the kernel size, to define the amount of blur. The sigma value determines the magnitude of Gaussian noise to be added to the original image which, in turn, removes the background noise of the image. Gaussian noise is essentially a statistical noise that has a probability density function equal to normal distribution. To gain a better understanding, Verma and Ali [9] consider each pixel in the blurred image to be the sum of the true pixel value in the original image and a random, gaussian distributed noise value.

Next, the image is converted from BGR colour space to HSV colour space. This is necessary as the bounds of the colour of the ball are defined in HSV colour space. A mask is created with lower and upper bound HSV values which are thought to represent the colour of the ball. The mask is essentially a segmentation of the colours of the image that lie within the threshold between our defined bounds and the colours of the image that do not. A pixel's value is set to 255 if it lies within the bounds producing a white pixel and is set to 0 if it does not, producing a black pixel.

The mask is then eroded through 2 iterations to remove small, irrelevant blobs that may be in the range of the specified HSV values, but are not large enough to be considered as the target object. Finally, a series of 20 dilations are applied to the mask to increase the size of the blob the ball makes. As is apparent in figure 5.1,

the ball is not entirely one colour therefore produces blobs separated by the pixels of colours that are not within the defined bounds. The series of dilations enlarges these separated blobs resulting in them merging as if the ball is originally a single colour.

To find the appropriate HSV lower and upper bound values, another short program is written that applies all the functions previously mentioned to the image, starting with initial HSV bound values that are estimations of the colour of the ball. These values are adjusted by hand until only the blob of the ball is shown in the resulting mask. The program allowed the appropriate HSV upper and lower bounds to be found and these values are used in the final implementation of the feature. This allows us to detect the ball in each frame of the video successfully. Figure 5.1 shows the initial frame. Figure 5.2 shows the final result of the program.

Figure 5.1: Frame From Video In Data Set

Figure 5.2: Figure 1 After Running Program

Tracking the ball and its path is also completed successfully using OpenCV. Similar to the previous program, the ball is detected using the previously discovered HSV values. The contours between white and black in the resulting mask are used to find the ball in the image with the white region representing the ball. The function minEnclosingCircle() defines the centre of the ball. After each centre is found in a frame, its pixel coordinate value is appended to the end of a queue that has a fixed size. Using the points in this queue, a line is drawn using the line() function that connects each point in the queue. This results in a contrail being drawn originating from the centre of the ball and essentially mapping the ball's trajectory. In this case, the length of the contrail is equal to the queue length. This is shown in Figure 5.3. Each frame with a contrail drawn is written to video using the videoWriter OpenCV object at 30 frames per second, creating a video with this contrail following the path of the ball. This trajectory mapping is utilised on the web application to aid in improving a user's free throw shooting which will be discussed later (see section 8).

Figure 5.3: Ball Tracker

The program performs well when tested against the videos from the dataset, tracking the ball and drawing a contrail from the centre of the ball 87% of the time, taking into account positions where the ball is bouncing or in the participants hands. This feature is completed shortly before the assigned amount of time allocated for this feature of three weeks. The remaining time is spent doing research and understanding the methods behind the next feature as it is predicted that this would be more difficult than the last.

6 Make/Miss Detection

6.1 The Convolutional Neural Network(CNN)

The solution fashioned for the make/miss detection feature is centred around the use of a convolutional neural network. To fully comprehend the solution it is recommended to first understand the concept behind the convolutional neural network.

CNNs are designed for image classification by adaptively learning spatial hierarchies of features therefore assumes the input into the network to be image data. The data is processed and classified into relevant labelled classes respectively from the training data. An input image is taken as an array of pixels with the form of a h x w x d matrix (with h = height, w = width, d = dimensions). For example, an image of 5 x 5 x 3 refers to a matrix of RGB as 3 refers to the three RGB values and at each pixel position there is a value for the intensity of each of the colour channels. An image of 5 x 5 x 1 refers to a matrix of grayscale as there is one colour channel representing only the intensity of light. Each input image is passed through a series of convolution layers with filters/kernals, pooling, and finally a fully connected layer at which point a final activation function is applied to classify the images. CNNs look for specific localised image features like horizontal and vertical edges in the image that are used later, deeper in the layers to find and extract much more complex patterns. This ultimately allows the recognition of complex structures in the image relevant to a specific class, which in turn assists in the recognition of said class.

The convolution layer is always the first layer in a CNN and is fundamentally based on applying the convolution mathematical operation to extract features from our input image. A set of filters consisting of matrices of k x k x d dimensions, typically either 3 x 3 x d or 5 x 5 x d, of weights or parameters slides across the pixels of the input image one unit at a time left-to-right and top-to-bottom. Element wise multiplications are computed using these filters on the input images

pixel values that are within the filters sliding window. These multiplications are summed for every window on the input image. This process is called convolution and can be simplified as feature map (F) = Input (I) * filter(K) where F is a matrix of the form (h – k + 1) x (w – k + 1) x d. Velickovic, P., et al. [10] illustrates a single convolution using a 3x3 filter, shown in figure 6.1 below. N filters in the convolution layer will result in the feature map containing N layers with each feature map representing the relation between the data and the relative filter. Activation functions including the ReLU and Tanh functions are commonly used as the last component of the convolution layer to increase the non-linearity in the layers output. The hyper-parameters (parameter values manually chosen to control the learning process) of a convolutional layer are the filter/kernel size (k), number of filters (f), stride length (s) to determine the number of units to slide the filter window across the input image by, and the padding (p) of the input which applies a row of values on the outer side of the input image so filters can be applied at every position of the input.

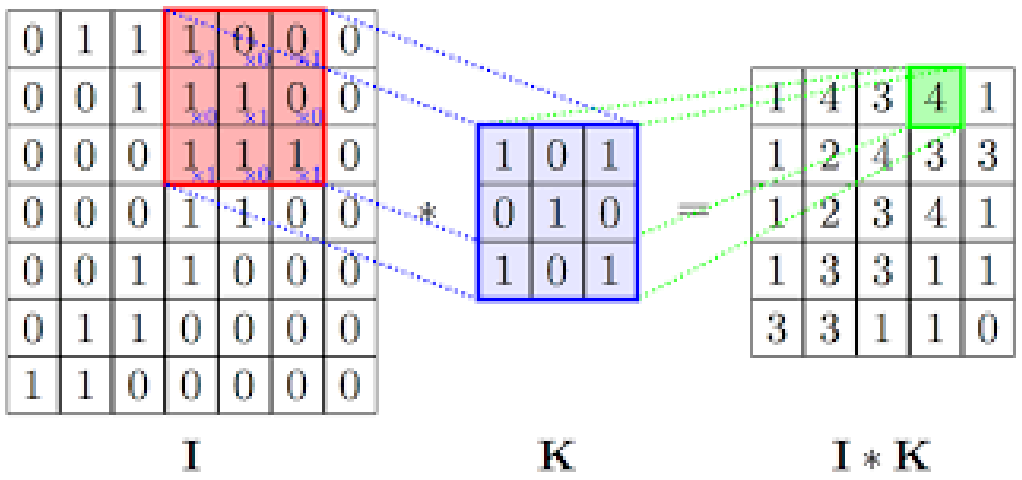


Figure 6.1: Convolution Example

Pooling layers usually follow from convolution layers. This layer performs a sub

sampling operation reducing the dimensionality of the feature maps while retaining important information. Subsequently, the number of parameters are reduced when the images are too large and therefore the required amount of computation is reduced. Pooling is performed in a similar way to convolution where a sliding window is utilised but under different operations. There are different types of pooling including max pooling where the largest value element is taken from the sliding window in the feature map, average pooling where the average value of all elements in the window is taken, and lastly sum pooling where simply the sum of all elements in the window is taken. For example, [11] depicts this concept using max pooling and a stride length of 2, given in Figure 6.2.

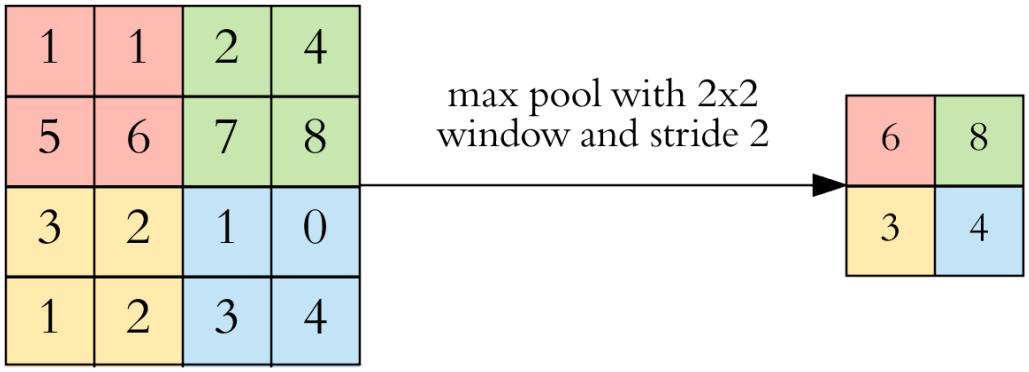


Figure 6.2: Max Pooling Example

The fully connected layer is essentially attached to the end of the previously mentioned network of convolution and pooling layers. The layer takes a flattened matrix vector as input. The neurons in this layer have full connectivity with the neurons in the preceding and succeeding layer like a standard feed forward neural network. The learned representations given by the convolution and pooling layers are used to produce an output. This output is a prediction of the class of the image using a final activation function like the sigmoid or softmax functions. Figure 6.3 [12, fig. 1] below shows an example of a full convolutional neural network including the fully connected layer for aircraft structural health monitoring.

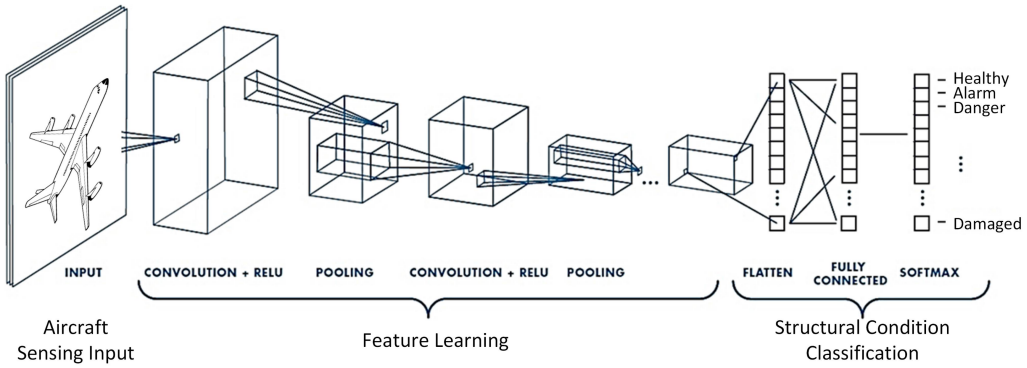


Figure 6.3: Full Convolutional Neural Network Example

6.2 Make/Miss Detection Solution

As the Gantt chart in the figure 2.4 shows, the next stage of development is the make/miss detection feature. The duration of this development cycle is three weeks, similar to the previous feature. A proportion of time allocated for the development of this feature in the second week of the cycle is dedicated to writing relevant documentation, namely a progress report. Naturally from the previous sub-section it is established that a CNN would be most suitable as a machine learning model. The fruition of this feature fundamentally revolves around a multi class classification problem to determine whether a free throw shot has been made or missed, using frames from videos as the input image data. CNNs are designed to map image data to an output variable by developing an internal representation of an image, while also reducing the number of parameters without reducing the quality of a model. They are effective in its design therefore is the pronounced model of choice. Tensorflow and Keras are libraries used to build the network as these libraries have clear documentation, are simple to learn and not particularly difficult to use.

The objective of this feature is to enable the detection of missed and made shots in videos however the convolution neural network does not take video data as

input, it takes image data. This means a method must be used to convert a video from its original form into frames or images. These frames are used as input into the network to produce predictions across each frame. The frames and their predictions must be combined in some way to give a final sense of continuous video classification. Converting video from its original form into frames is simple, as this has already been accomplished (see section 5) using the videoCapture object.

Karpathy, A., et al. [13] proposes three different strategies for combining frames and their subsequent predictions. The single frame strategy simply aggregates predictions across static single frames, which is suitable enough for the purpose of this project, therefore is the video classification strategy of choice. The free, open-source, interactive web tool, Jupyter Notebook, is used as an environment to develop the CNN. Jupyter Notebook is the preferred choice because of the familiarity with it due to previous and current projects also using the tool for development.

The data collected from the data generation process (see section 4) must be organised before it can be used as input for training the neural network. The video data is first converted into its constituent frames and a large sample of these frames are used as training data. To begin with, it was thought the classification task was a binary classification problem with the two classes corresponding to a make and a miss, as suggested by the name of the feature. Later, it was discovered that this was not the case as there is another important class that frames of video will need to be classified as, so all frames can fall into a distinct class. As mentioned previously, we are dealing with a multi class classification problem therefore these frames are examined, classified, and labelled as one of three possible classes. The first is the "none" class. The ball in the frames of this class are either in the participant's hands or possession, travelling towards the rim after just being shot by the participant, or is a loose ball (not in anyones possession) as the shot attempt has just been missed or made. Frames classified as the none class are labelled with the integer 0. An example of a frame of this class is shown in figure 6.4.

Figure 6.4: Example Frame From The "None" Class

Next, the “miss” class which is self-explanatory as a free throw shot will have been missed to be labelled as this class. The ball in the frames of this class will be in the close vicinity of the rim, as the aim is for the network to detect a miss as soon as the ball does not appear to have any chance of going in. Frames classified as the miss class are labelled with the integer 1. An example of a frame of this class is shown in figure 6.5.

Figure 6.5: Example Frame From The "Miss" Class

Finally, the “make” class which is also self-explanatory as free throw shots will have been made to be labelled as this class. The ball in the frames of this class will be in the basket and/or the net, and will be classified as the make class as soon as the ball does not appear to have any chance of missing; contrary to the miss class. Frames classified as the make class are labelled with the integer 2. An example of a frame of this class is shown in figure 6.6.

Figure 6.6: Example Frame From The "Make" Class

The majority of the frames in the dataset are of the none class. This is because there is a short period of time in the videos where frames can be classified as a miss or make, compared to the period of time where frames can be classified as the none class. The dataset reflects this. The miss and make class also have frames that are quite similar especially with misses that travel past the sides of the net giving the illusion that the ball may be in the basket. It is predicted that the network will find it most difficult to classify these types of frames correctly. To combat these issues and to reduce the general overfitting of the network, data augmentation is used to increase the size of the dataset. This is achieved by using popular data augmentation techniques including image rotations, width shifts, height shifts, image shearing, zooming and horizontal flipping. A technique that is disregarded is vertical flipping as although this may improve the model's ability to generalise, users of the ensuing web application will not give videos which are upside down. On the other hand, horizontal flipping has significance as users can provide videos where the basket is either on the left or right side of the frame. Figure 6.7 shows an original image from the dataset before any processing. Figure 6.8 shows three

different augmentations of the original image all of which are used for the training of the model. The original dataset consists of 133 frames of the make class, 195 frames of the miss class, and 2171 frames of the none class to make a total of 2499 frames in the dataset. Using the Keras *ImageDataGenerator()* and *flow()* functions several more frames are generated and appended to the original dataset in batches of size 32. Due to the prolific number of frames in the none class, most of the generated images are of the other two classes with 224 additional augmented frames for each of the two, and 96 additional augmented frames for the none class. This increases the size of the dataset to 3043. The number of frames could not surpass this due to the limited amount of memory available while training the model.

Figure 6.7: Pre-Augmentation Frame

Figure 6.8: Augmented Frames

The architecture of the CNN consists of two convolution layers, each with max pooling followed by two fully connected layers. Using shorthand notation, the full architecture of the network is *C(64, 3) – MP – C(64, 3) – MP – D(0.25) – FC(64) – D(0.2) – FC(3)*, where *C(f, k)* indicates a convolutional layer with f number of filters of spatial size k x k x d. *MP* denotes max pooling following a convolution layer with a sliding window of a fixed spatial size 2 x 2. *FC(n)* indicates a fully connected layer with n output neurons making up the dimensionality of the output space. Lastly, *D(x)* represents a dropout that randomly selects a proportion, x, of the neurons in the layer and sets their weights to zero for one iteration of training. Dropout is considered as a regularisation technique, that has the main advantage of preventing all the neurons in a layer from synchronously optimising their weights, decorrelating the weights from each other. This essentially helps prevent overfitting, giving the network the ability to generalise more and making it more robust. The final fully connected layer has three neurons in its output space as we have three classes for classifying our input data, as mentioned before. Equipped with a softmax activation function, given by the formula $\frac{e^{z_i}}{\Sigma_{j=1}^{K} e^{z_j}}$ where K is the number of classes, and Z is the pre-activation output value given to each class, the final layer in the network is able to give a multinomial probability distribution. The vector of numbers in the output space are converted to a vector of probabilities, where the probabilities of each value are proportional to the relative

scale of each value in the original vector. The softmax activation function is very commonly used in multi class classification, as you receive the probability of the input data belonging to each class. This makes it very easy to see which class the input is predicted to belong to as it has the greatest probability. The next closest predicted class can also be observed as the next largest probability, and so on. The convolution layers both have 64 filters in the convolution along with a kernel size of 3 x 3. Both also use the ReLU activation function, which is a piecewise linear function that directly outputs the input if the input is positive, otherwise, if the input is negative the function will output zero. The ReLU function is shown graphically in figure 6.9 below.

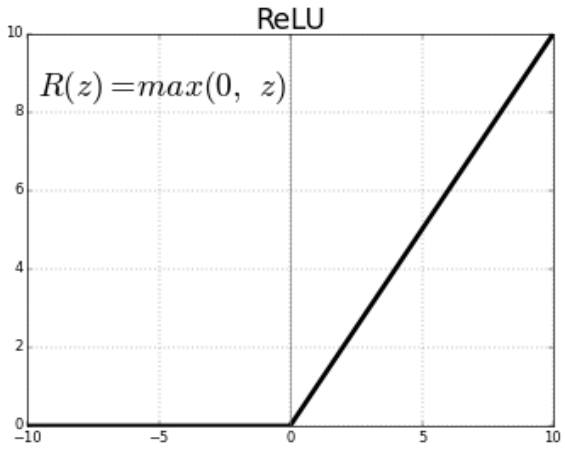


Figure 6.9: ReLU Activation Function

Each frame from the dataset is imported and labelled with their corresponding classes. The frames are then converted to grayscale and resized such as the examples seen in figure 6.8. This results in less processing for the CNN to undergo while training, and therefore reduces the computational expense compared to a larger sized RGB frame which uses three colour channels. The data is then shuffled before it is used for training, as it serves the purpose of reducing the variance of the model and to ensure the model remains generalised by in turn

reducing overfitting. With a total of 3043 frames in the dataset, the data is split into a training set and test set. The test set is composed of 10% of the total data. 10% is decided to be suitable as we do not have a vast amount of data to train our network with, therefore it is ideal that a large percentage of data remains a part of the training set. Before we begin training the model, each frame's corresponding class needs to be one hot encoded in preference to its current representation as integer values of {0}, {1}, and {2}. This is because, similar to other machine learning algorithms, our model cannot work with categorical data directly. One hot encoding is a representation of categorical variables as binary vectors, where all values in the vector are zero with the exception of the index of each integer value which is marked as 1. For example, the class {1} will be represented as {0, 1, 0}. This type of representation is only required in multi class classification models.

Using the Keras' sequential model, the network is trained through 15 epochs. This is where the entire dataset is passed forward, for forward propagation, and backwards, for back propagation, through the network once. The batch size defines the number of frames in the training data that will be worked through before updating the model's internal parameters, this is set as 32. In addition to shuffling the training data before training the model, the training data is also shuffled before each epoch giving the same effect of reducing variance and overfitting. A validation split is defined as a fraction of the training data to be used as validation data, more specifically 10% of the training data is used as this validation set. The model will not be trained on this validation data, as it will instead be used to evaluate loss and accuracy of unseen data at the end of each epoch.

The training data accuracy, that calculates how often predictions equate to correct labels, through each of the 15 epochs steadily increases from an initial accuracy of 0.8307 to a final accuracy of 0.9982. The high accuracy conveys the network's strong ability to class the training data with the correct labels. This is ideal as this data is frequently sort through to update the internal parameters of the model.

Using the categorical cross entropy loss function, also known as the softmax loss function, the loss steadily decreases from an initial value of 1.6837 to a final value of 0.0115. The final validation set loss and accuracy of the model on the training data is 0.0697 and 0.9867, respectfully, showing the model performs well with unseen data. The training data accuracy is consistently above 0.8 even through the first epoch. This is thought to be because of the large number of data belonging to the none class which is relatively easy to classify, compared to the other two classes because of its distinct frames. This means the high accuracies are not a perfect representation for the performance of the model. The model is also evaluated with unseen test dataset achieving an accuracy of 0.9839 upon completion of this evaluation. Comparing this test data accuracy with the training data accuracy, the model is considered to be structured well. A test accuracy that is far less than the training accuracy is indicative of overfitting of the model, as the model performs well with familiar data but performs poorly with unseen data as it fails to generalise. A test accuracy that is far greater than the training accuracy is indicative of underfitting, as the model may not be able to capture the underlaying trend of the data causing the test accuracy to be greater by chance, whereas this should not be the case. A test accuracy of 0.9839 is similar to the accuracy of the training set therefore a good indicator of the model's ability to capture the underlaying trend of the data. To gauge the model's ability to accurately classify data of each class, the table below shows accuracies corresponding to each of the three classes.

Class	Accuracy
None {0}	1.0000
Miss {1}	0.9130
Make {2}	1.0000

Evidently, from the values in the table the model has no apparent problem classifying frames of the none and make class as the accuracies are shown to be 1.000

or 100%, predicting all frames with correct labels. The model struggles more when attempting to classify frames of the miss class. This may be because of the underlaying data, where the miss class contains frames which are rather similar to the other two classes, especially the none class. Figure 6.10 below shows two frames, the first on the left-hand side is a frame of the none class. The second on the right-hand side is a frame of the miss class. As you can observe, the frames are very similar with only the ball moving distance of a few pixels to distinguish them. The accuracies corresponding to each of the three classes use frames from the test data to obtain them. The majority of the test data is also composed of frames of the none class as it is a proportion of the dataset. This means that although these accuracies are a good indicator of how the model is performing regarding each individual class, it may not be a great indicator of how it will perform with a real life sample of a sequence of frames from video.

Figure 6.10: Frames From Both The None Class and Miss Class

The *predict()* Keras' function produces an array containing the probabilities of a frame belonging to each of the classes. Each class is represented by each index of the array, as previously mentioned. For example, if the model predicts a frame to be of the make class, index 2 will have the greatest probability. To convert these arrays into an effective form, the index of the maximum value in the array is taken hence the predicted class. Adhering to the functional requirements of the project, the number of missed, made and total shots can be computed using the predictions of the model. This is achieved using make and miss counters. Through each

frame's prediction, if there is a prediction of the make class the make counter is incremented by one, and likewise if there is a prediction of the miss class with the miss counter. If any of the two counters are greater than the value 5, the number of made or missed shots are incremented by one respectively. This is because during a made or missed shot, a make or miss frame can be classified as such through 6 consecutive frames of the video at the least. Intuitively, this means if we receive 6 or more consecutive frames that are predicted as the make or miss class, we can conclude that we have a make or miss in the video. Once we have a make or miss, both counters are set to negative infinity to avoid multiple makes and misses being recorded, due to a sequence of 7 or more consecutive frames predicting a class from a single make or miss. This also prevents the recording of the opposing class due to incorrect predictions. If the none class is predicted, both counters are reset to zero which allows the recording of the next make or miss following the previous shot. This also resets the counters if a make or miss class is predicted incorrectly when the none class should have been predicted. The number of total shots taken is simply the addition of made and missed shots. This method is tested against a sample video where two shots are taken, and two shots are made. The combination of this method and the model's predictions correctly give the result of two made shots and two total shots taken. Due to the small amount of sample video to test the method and the model's predictions against, the videos collected from the participants involved in benchmark performance testing is used. Each of the two participants have five recorded shots. Participant A makes four shots and misses one. The model and method predict four made shots and two misses, incorrectly predicting one additional miss. Participant B makes two shots and misses three. The combination of the model and method predict two made shots and three misses, correctly predicting the number of missed and made shots. These results reflect the model's ability to classify each class, as the make class is predicted correctly without issue, however the model incorrectly predicts an additional missed shot for one of the sample videos. Despite this, the results show the model performs well.

For the short amount of remaining time allocated to this features development, time is spent modifying the CNN in an attempt to raise the test accuracy, giving better performance. This proves to be futile as the original CNN produces the best accuracies.

7 Height of Ball Detection

Alike the previous two features, the duration of this developmental cycle is of three weeks. This feature is successfully completed within the allocated time cohering with the scheduled timeline of the project. The objective is to ultimately detect or determine the balls maximum height during the arc of a free throw shot. By doing so, this assists in the improvement of a user's shooting ability by providing key shot mechanics information to the user. There are two contrasting methods considered for use as possible solutions for this feature of height of ball detection.

The first method utilises a reference object for distance calculation. Jungel, M., et al. [14] show several different methods for using reference objects for distance measurements, including using objects of known size (e.g. the dimensions of a ball), objects of known height, and objects of known outline on the ground (e.g. goals and field lines). All the above require the object to be relatively easily identifiable in an image or frame. For this feature and project, the most suitable method is to use a reference object of known height, as this is the variable we are most concerned with, measured in meters. Research shows an article using OpenCV to measure distance between objects [15], the program detailed in the article accomplishes goals similar to those related to the objective of this feature, therefore the article is used as reference. An image containing the reference object of known height must first be processed by converting the image to grayscale, and blurring it to remove background noise. Edge detection is performed on the image using OpenCV's canny edge detector, essentially giving distinct objects in the form of contours, along with dilations and erosions to close the gaps within objects. The contours in the image are sorted left to right, therefore the reference object is always the first entry in the sorted list if it is the left-most object in the image. Using the reference objects contour, we can compute the number of pixels per meter which is simply the number of pixels in the image that fit into a known distance of a meter, using the reference objects known height. The vertical pixel distance between the centre of the reference object and the centre of the target object, in this case the ball, can be computed using their y-coordinate pixel

locations derived from their respective contours. The final distance in meters can be calculated as this vertical pixel distance divided by the pixels per meter value. The actual distance of the ball from the ground is half of the known distance of the reference object, as we use the centre of the object to measure the distance from the ball, with the addition of the computed vertical distance from the ball. Vertical distances are computed through each frame of video, the distance that is the largest is taken as the maximum height of the ball above the ground during the free throw shot. It is possible to use this value as the maximum height of the ball as the greatest vertical distance will be found where the ball is highest in the air, as the reference object is always on the ground. This is the preferred method of choice as it produces a tangible figure for users to know the maximum height of the ball, however the initial video recordings do not include a reference object as this method is only discovered after the recording of data. In order to achieve height of ball detection using this method, the video samples will have to be generated again. Issues regarding the booking of a facility suitable for the re-recording of the video data continued to arise due to implications surrounding COVID-19 therefore alternative methods have to be considered.

The second method uses a similar sequence of techniques as the object detection feature (see section 5). Unlike using a reference object, only the ball is detected through first blurring the image, converting to image to HSV colour space, and applying a mask on the image with bounded HSV values thought to represent the colour of the ball. Followed by a series of erosions and dilations, the ball can be successfully detected by using the contours in the resulting mask. The ball is now represented as a white blob which can be seen in figure 5.2. The centre of this blob can be found using the minEnclosingCircle() function giving the (x, y) pixel coordinates of the ball through each frame of video. These centre points are evaluated as the series of operations are applied to each frame to produce these coordinate values. The frame corresponding to the pixel coordinates with the lowest y value, is taken as the frame that contains the maximum height of the ball during the shot, as y pixel coordinate values increase from top to bottom

in a frame. Using this method, we cannot directly produce a figure for users to analyse and evaluate their free throw shot with like the prior method. Instead, we can produce a frame which contains the maximum height of the ball. Although this may not be as useful as a direct figure, the frame provides a spatial and visual representation of the maximum height of the ball which some users may find more useful. The feature performs well during testing which verifies its functionality. Each test video contains a single free throw shot and the frame with the maximum ball height is known. The maximum height of the ball during the shot is correctly identified, with the correct frame produced with the ball at its peak through all 20 of the test videos. Due to the implication surrounding COVID-19 mentioned in the previous method, this method is used for the development of the feature which proves to be successful in detecting the maximum height of the ball during a user's shot. Originally planning to utilise the first method but finally realising this cannot be done, is an example of one of the advantages of using a FDD methodology, as it welcomes the change in plan.

8 Web Application Development and Integration

The high-level Python based, free and open-source web framework, Django, enables the fast development of sustainable and maintainable web applications, suited towards both the front and back end. With Django's compilation of tools like URL management and template languages, as well as its compatibility with Python, it is the most appropriate web framework to use for development also considering that its features are written in Python. The web page and its relevant stylings are achieved using HTML and CSS, however to aid in the aesthetic development of the website, the popular open source front end framework Bootstrap v4 is used. Bootstrap provides several HTML and CSS templates for user interface elements, allowing websites to be designed faster and easier. More specifically, elements from the modern business template including a navigation bar and carousel slide are used in the web application. Figure 8.1 shows a screenshot of the home page of the website showing both the navigation bar and carousel slide.

Figure 8.1: Screenshot Of Homepage

To abide by functional requirement C1, a file upload button is placed on the home page which makes use of the *<input>* form element with a file type. This allows users to select relevant input files from their local folders. The "analyse" button beside the file upload is of submit type therefore upon clicking the button, the POST method of the form is called and the action of loading the next "analysis" page is taken. A small script of JavaScript allows the analyse button to only be clickable once there is a file selected using the *addEventListener()* JavaScript function. The number of functions required to upload and analyse a video on the homepage are made to be minimal, to abide by the non-functional requirements of the project. The video upload and analyse buttons are the first functions the user will be presented with upon visiting the website, making the website intuitive and easy to use. Both buttons can be seen in Figure 8.1 above.

To ensure users upload acceptable videos for the analysis to be successful, a list of recommended requirements is placed on the homepage just below the analyse and video upload button. This can be seen in Figure 8.2. The first requirement is in reference to the object detection feature, where the ball is detected using its colour. Standard BGG molten balls are all of the same colour, the range of colours used to detect the ball should still be suitable for balls that are not standard BGG molten balls but are similar. The next requirement relates to the video data as all recordings are taken from a side-on angle from the shooter and the rim. The make/miss detection feature makes use of this angle to train the model, therefore videos that the model makes predictions on must make use of the same angle for better accuracy. The camera angle should be wide enough to capture the ball at all times to ensure the maximum height of the ball can be captured during the free throw shot. Users are encouraged to avoid wearing orange clothing because the ball is of an orange colour, so wearing orange may result in the object detection feature detecting the user's clothes instead of the ball producing less accurate results. The video data is recorded indoors and the model is trained on this data, therefore the videos the model makes predictions on must also be indoors for the best accuracy. Sunlight, background objects, and other uncontrollable environmental variables

will have an influence on the performance of the free throw shot analysis therefore is best to avoid. A camera stand should be used for stability to avoid the shaking, or moving of the camera used to record the video. The rim in the video data that the model is trained on has a net attached, so the rim in user's videos should also have a net attached as this may be important for predictions, especially of the make class. The last requirement states the optimal time between shots is 10 seconds as the contrail generated from the object detection feature gradually disappears after this period of time.

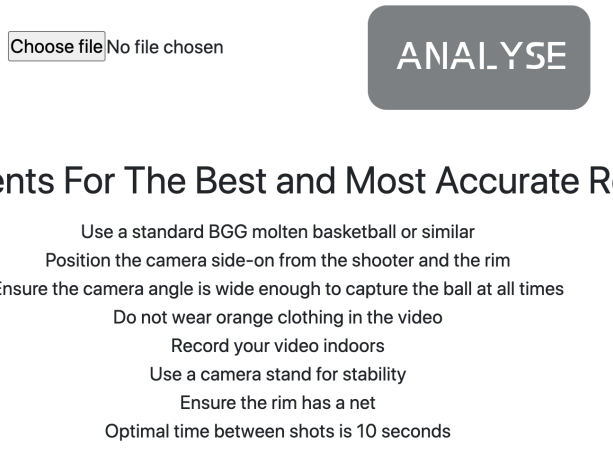


Figure 8.2: User Video Requirements on Homepage

A view function or view is a python function used by Django to receive web requests and return web responses. In the case of this web application, all views receive web requests and return a response in the form of the HTML contents of the web page that is requested. To integrate all features within the website, the analyse button requests an analysis view that contains all the function calls and code regarding the back-end features of the application, and essentially the analysis of the user's video. The button also submits the uploaded video file to be processed. The video is then saved into a media file for later use. The code for all subsequent features of the application is divided and organised into different files and independent modules. This achieves modularisation that allows the code to

be more maintainable and easily understood.

The object detection feature is written into its own file named balltracker.py and made into separate methods that still complete the same objective of the feature. The first method *createFrames()* converts and saves the video into its constituent frames that have the ball tracked and a contrail drawn as specified in section 5. The second method *frames_to_video()* convert these sequence of frames into video also mentioned in section 5. At this point, the user's input video is converted to a video with the ball tracked and a contrail constantly drawing the path of the ball originating from its centre. A visualisation of the trajectory of the ball during a user's free throw shot is created, but to aid in the analysis and improvement of the user's shot, an "arc analysis" helps the user to make more sense of the results of the object detection feature. Figure 8.3 below is a screenshot from the analysis page detailing the description given for the arc analysis process. From reading the description, improving free throw shooting through the arc analysis fundamentally involves the user comparing the trajectory of the ball through each of their shots. The goal is for the arc or contrail generated from any given free throw to follow the same or very similar path as the shots preceding and succeeding it. The more similar the paths, the more consistent the shot and therefore the better the shooter. The arc analysis finally states that if the arcs vary a lot the user should practice again, focusing on consistency. At this point the user is able to understand a key element in shooting free throw shots with a visualisation of their own shooting. The arc analysis helps the user in making their shooting as consistent as possible for the benefit of improvement. Each "Arc Video" is downloadable therefore a user can shoot, record and upload their videos for analysis any number of times to compare the improvement and consistency in each video. The final method from the balltracker.py module *createBlankFrames()* converts the video into its plain constituent frames, as opposed to frames that have the ball tracked and contrails drawn which become replaced. The next few features require blank frames for processing so this method is required. These frames are stored in a media folder.

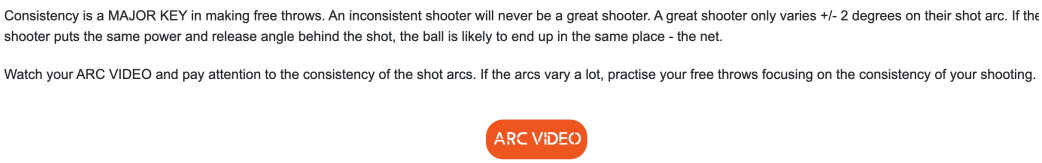


Figure 8.3: Arc Analysis

The make/miss detection feature trains a convolutional neural network that predicts frames to be of three different classes. The model is saved into a h5 file by simply calling the *save()* function and named *make_miss_detector.h5*. Only a single method *predict()* is required for the make/miss detection feature which is located in the makemiss.py module. The model must first be loaded from the h5 file using the Keras' *load_model()* function, which loads exactly the same model previously trained with exactly the same parameters. Each frame must first be processed in the same way frames are processed before being used for the training of the model. The model then produces raw predictions in the form of probabilities of each class, again in the same way following training, these raw predictions are converted to a more effective form by taking the index of the maximum value in the array hence the predicted class. As mentioned, counters are used to predict the number of made and missed shots with the total number of shots as an addition of the two. A made shot percentage is simply computed as $\frac{\#\,of\ made\ shots}{\#\,of\ total\ shots} \times 100$. The method returns an array of the form *[total shots, made shots, missed shots, percentage]*. Figure 8.4 below shows the results of the make/miss detection on the web application using the same sample video specified in section 6 of two made shots and two shots taken.

Figure 8.4: Display Of Make/Miss Detection On Webpage

The height of ball detection is written into a maxHeight.py module containing only a single function *max_height()* that is almost identical to the chosen method in section 7. Combining the make/miss detection feature with the maximum height of ball detection feature, we can produce frames that contain the maximum height of the ball during a shot through each make and miss. The *max_height()* method has three parameters to accommodate this integration with the make/miss detection feature. *Lower*, *upper*, and *make_or_miss*. In the *predict()* function from the makemiss.py module two variables, *upper* and *lower*, are used to determine the frames the *max_height()* function should be applied to. While the predictions are iterated through using counters to determine the number of made and missed shots, the upper variable represents the frame prediction we are currently on. The lower variable represents the frame prediction directly after a frame prediction where a make or miss had just previously been concluded, or the variable initially represents the starting frame. Once a make or miss is recorded the *max_height()* function is called with the relevant arguments. Applying this function to frames only between the *lower* and *upper* numbered frames, means the maximum height of the ball is found during a single shot which is either a make or miss. If a made shot is recorded the function is called with the *make_or_miss* variable specifying a make, and the maximum height frame is stored in a make file. Otherwise, the variable specifies a miss and the maximum height frame is stored in a miss file. This allows the user to view the maximum height of the ball when a shot is missed compared to when a shot is made. Using the same sample video with two made shots and two shots taken, we expect to see two frames of each shot containing the maximum height of the ball during both shots. A screenshot from the web application shots the results of the maximum height of ball detection in Figure 8.5.

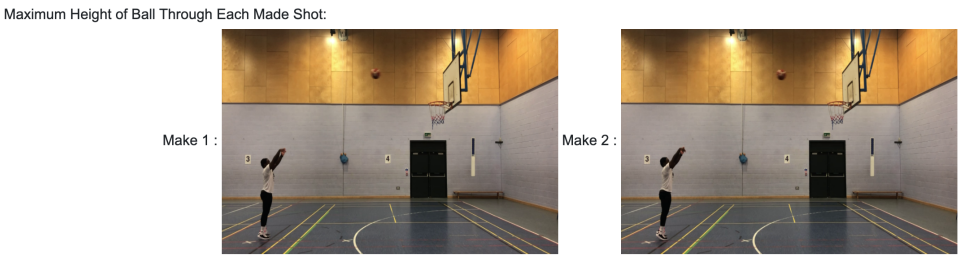


Figure 8.5: Display Of Maximum Height Detection On Webpage

Improving a user's free throw shooting using this feature entails the following description on the web application shown in Figure 8.6 below. It begins with a small discussion of optimum release angle to give the user a greater understanding of underlaying shot mechanics. Next, stating the maximum height of the ball should ideally be 3 feet above the rim and 8 feet from the shooter at the free throw line. Similar to the arc analysis, users are asked to compare the maximum height of their shot with the diagram shown below the description in Figure 8.6. The aim of this is for each maximum height ball placement through each made shot, to be as close as possible to ideal maximum height ball placement in the diagram. The description states to notice the consistency of the height of the ball during each made shot, as the ball placement through a made shot will more often be much more ideal compared to missed shots, hence the made shot.

Average shooters who shoots with an optimal arc of 45 degrees (give or take two degrees) will make about 11% more free throws than a shooter with a high arc of 53 degrees (68% vs. 57%). The same holds true going the other way. That same average 45-degree shooter makes 12% more free throws than a flat-arc 35-degree shooter (68% vs. 56%).

The maximum height of the ball should reach 3 FEET above the rim for an ideal shot arc. The ball should reach its maximum height slightly closer to the rim than the shooter - Approximately 8 feet from the shooter at the free throw line.

Compare the maximum height of your shot with the diagram below. Notice the consistency of the height of the ball during made shots! The maximum height of the ball should reach the top of the backboard.

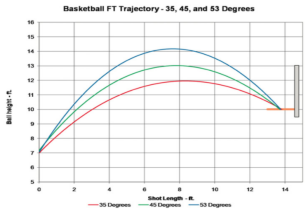


Figure 8.6: Display Of Description Before Maximum Height Detection On Webpage

Referring to the Gantt chart in figure 2.4, the web application and integration task is scheduled for development between the dates of 28th of December through to the 18th of January, for an unvarying development cycle of three weeks. This task did not commence until the week of the 18th of January due to an over embellished start to the new year and profuse personal circumstances, resulting in the development of the project being behind schedule. In addition, the development of the web application and integration of features proved to be one of the more challenging aspects of the project, because of the task of single-handedly building a system to satisfaction. The task had not been attempted before causing a learning curve with a gentle slope, as difficulty is encountered processing the underlying workings of Django. This stagnated the timely progression of the cycle, resulting in this feature/task being the last.

9 Evaluation

This evaluation determines the quality and success of the project, in addition to evaluating its features and the investigation of whether it can be used to improve a user's free throw shooting ability without the assistance of a personal trainer. The features/tasks up to and including the height of ball detection were all completed in a timely manner, adhering to the schedule of the project. The schedule breaks down during the development of the web application and integration, as mentioned. Although the main features of the application were completed, this meant other extensions tasks and improvements did not have much time left allocated towards it or were not attempted at all. Knowing this, the scheduling of the project somewhat contributes to its quality as with better time management and circumstances, more features could have been included in the application as well as the potential of improving upon completed features. These both will have unquestionably improved the project's ability to be successful in its investigation.

9.1 Evaluation Against Project Requirements

In an attempt to make the evaluation of this project more robust, a metric is designed to aid in the determination of how successful the project is with respect to the project's requirements. This metric will be referred to as the project's success score that quantifies the success of the project. The following formula defines the success score: $\Sigma_{i=1}^{C} 2(c_i) + \Sigma_{j=1}^{E}(e_j)$ where c denotes a core requirement taking the value of 1 if it is successfully completed and 0 otherwise, and with e denoting an extension requirement also taking the same values of 1 or 0 if successfully completed or not. C and E are the total number of core and extension requirements respectfully. The core requirements are weighted more heavily compared to the extension requirements as it is not necessary that extensions be completed for the overall success of the project, however it does contribute to the overall quality of the solution. Considering this, core requirements are weighted doubly. The project requirements are made up of 12 core and 4 extensions, as all non-functional requirements are regarded as core, thus the maximum possible success score of the

project is bounded at 28. Producing a success score of 13 or less will result in the project being considered as unsuccessful as many of the project's requirements, especially cores, have not been met as well as more than half of the maximum success score not being achieved.

The non-functional requirements (NF1- NF5) all consider the users experience with the web application. To gauge the number of non-functional requirements that have been successfully completed, the application undergoes usability testing. All users participating in usability testing are given the sample video mentioned in the previous section with instructions to complete the video analysis of the sample. Five 'yes or no' questions are asked to each user to with each question directly corresponding to a non-functional requirement. These questions and their corresponding requirement are listed below.

- NF1 - Is the application simplistic?
- NF2 - Is the application easy to use?
- NF3 - Is the application easy to navigate?
- NF4 - Does the application respond quickly to user action?
- NF5 - Is the website aesthetically appealing?

Users are also asked to give a score on a scale of 1-5, 1 representing a strong "no" and 5 representing a strong "yes" and so forth. The results of the usability tests are shown in the table below. Non-functional requirements that receive a score from their corresponding question that is 3.5 or greater can be considered as successful. From the table, it is observed that requirements NF1, NF2, NF3, and NF5 are successfully completed as their respective questions all receive an average score greater than 3.5. On the other hand, requirement NF4 received a very low average score of 1.4. This is unquestionably because of the amount of time it takes to load the analysis page following the upload of a sample video and click of the analyse

button. Typically, it takes just over five minutes for the analysis page containing the results from all the relevant features to load. This time becomes longer with very long input videos. In light of this, non-functional requirement NF4 “the web application should have low latency” is unsuccessful.

User	NF1	NF2	NF3	NF4	NF5
A	5	5	5	2	5
B	4	5	4	1	3
C	5	5	5	2	5
D	5	5	4	1	5
E	4	5	5	1	4
Average	4.6	5	4.6	1.4	4.4
Completed?	Yes	Yes	Yes	No	Yes

Core and extension functional requirements are also evaluated to determine whether each one has been successfully completed or not. This is accomplished through a combination of successful individual tests and on the basis of whether the requirement had been attempted. To provide a short hand representation of the process mentioned, the table below summarises the results of evaluating the success of each functional requirement. The table shows core requirement C7 as well as all extension requirements (E1-E4) were not successfully completed. The primary extension tasks supported by requirements E2 and E3 specified in the Gantt chart (figure 2.4) were not attempted, simply because the project had become behind schedule during the development of the web application. It was decided to allocate time and effort into the presentation and report of the project, rather than to attempt to complete features that were not necessary for the project’s success. Leaving the extension tasks in the Gantt chart incomplete undoubtedly leads to the incompletion of the subsequent extension tasks, supported by requirements E1 and E4 as these do not have priority over the other extensions. Core requirement C7 states “the user should be able to download a pdf file of a report of their shot analysis”. This sub-feature is initially viewed as an important task hence documented as a core requirement, however after near completion of the web application and careful

consideration it is decided that a downloadable pdf is no longer necessary. Reason for this is that a part of the analysis of a user's free throw shot, the arc analysis, is in video format which cannot be accessed via a pdf report. Users are able to re-upload recorded videos for analysis therefore can access their analysis as many times as they need to, given they have the same video. Acknowledging this, the core requirement, C7, will be considered as an extension requirement henceforth, as to accommodate a slight change in requirements but to still apply penalties to incomplete or unsuccessful requirements. This decreases the maximum possible success score from 28 to 27.

Requirement	C1	C2	C3	C4	C5	C6	C7	E1	E2	E3	E4
Completed?	Yes	Yes	Yes	Yes	Yes	Yes	No	No	No	No	No

Through ten successful and one unsuccessful completed core requirements, as well as five unsuccessful extension requirements the computed success score for this project is 20. As this is greater than the minimum score of 13, we can define this project as successful in relation to its project's requirements.

9.2 Object Detection Evaluation

For the most part, the object detection feature performs well in its ability to detect the ball through each frame of video and hence through video recordings themselves. Unit tests show the results of the ball and its centroid being successfully detected and found through 87% of test frames that contain the ball. The feature also provides the foundation for an aspect in improving user's free throw shooting through the arc analysis. Although this is the case, the feature is built on the premise that the ball in each users video is of a certain colour. The ball in the generated data is used as a basis, therefore the feature ultimately assumes the colour of every ball to be an orange-red. Despite the fact that most basketballs used in practice are of the same colour or very similar, which the feature also assumes, this is not the case for a proportion of users as basketballs can often come in a

variety of colours. This implies that users who possess a basketball of the colour black, blue, green, etc., immediately cannot use the application or will have to find another ball that is suitable which is impractical.

The arc analysis provides a good method for users to gain a greater understanding and improve upon their free throw shooting, however the "arc video" shows the user a contrail that remains in the image between shots before dissipating. Figure 9.1 shows a screenshot from an arc analysis sample video illustrating the excess contrail region below the rim and after the shot has been made or missed. Although it still allows the user to compare shot arcs of consecutive free throw shots, a more ideal solution would only allow the contrail to be drawn while the ball is in the air on the way to the basket. This would prevent excess contrail in the image that is present between the time after the ball hits the rim and right before the ball is shot again, producing a better quality solution. Accomplishing this involves incorporating the make/miss detection CNN model into this feature, along with another network that is trained to determine the start of a free throw shot. This way a contrail can be drawn if and only if the ball is in the process of being shot, so before the first network detects a make or miss but after the second network detects the start of a free throw shot. The method is much more extensive than the original therefore could only be accomplished with more time.

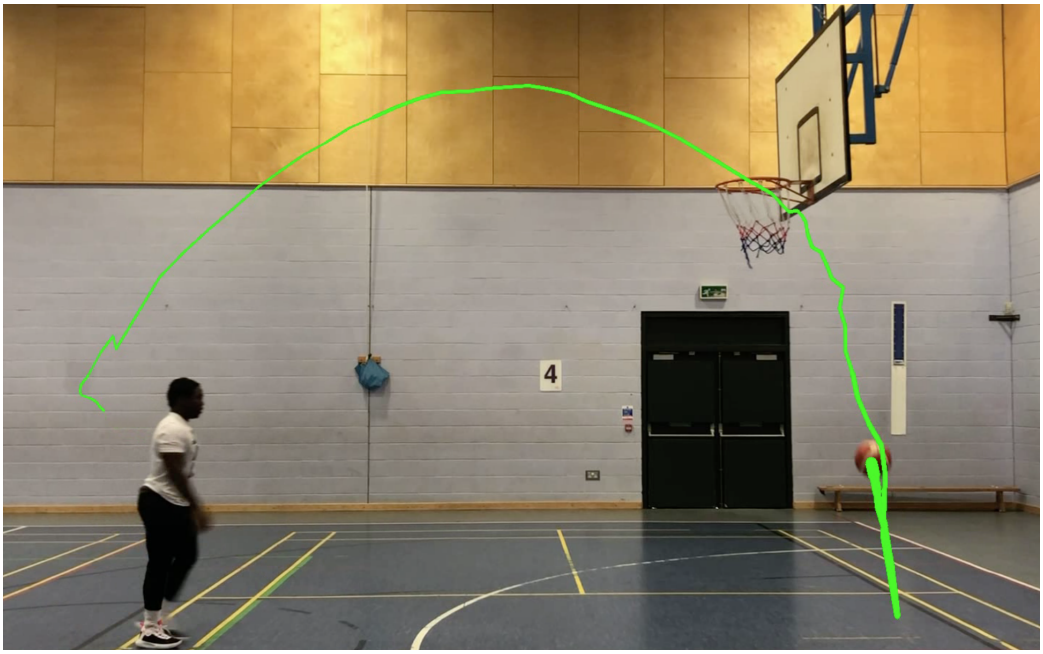


Figure 9.1: Screenshot Of Arc Video Showing Excess Contrail

A much more sound development method would be to use the motion of the ball instead of the colour, as this means any user with any ball is suitable to use the feature. There are also advantages and disadvantages to this method as the person shooting the ball in the video is subject to motion detection. The period of time from when the person has just released the ball to shoot, to when the ball has reached the basket will be the period of time of successful ball tracking using motion. More often than not, the person will not be moving much compared to the ball on the way to the basket. An alternative approach, used to detect and track tennis balls in Broadcast Tennis Video (BTV), proposed by Archana and Kalaisevi-Geetha [16] uses logical AND operations applied between the original image and a created background, alongside the performance of image difference. From this, ball candidates are detected by applying threshold values and dilating the image finally allowing the ball to be tracked. This presents a more complex solution, however proves to be accurate in identifying the ball and determining

the ball projection position. Tennis balls tend to move at very high speeds, much faster than that of a basketball when shooting a free throw, therefore the errors this method succumbs to due to the ball's small size and fast movement will not be the case when used for the purpose of this project, making it more ideal.

Machine learning methods were also considered but were eventually abandoned because of the simplicity of computer vision methods. Buric, M., et al. [17] compare the performance of two convolutional neural network based object detectors for the task of ball detection in non-staged, real world conditions. Again, with the resource of more time, the machine learning approach is preferred over the original computer vision approach. The first object detector uses the YOLO "You Only Look Once" method that has the aim of detecting the ball speedily by first slicing the input image into an SxS grid. For each cell, a probability distribution of an object class is simultaneously calculated to predict a corresponding confidence score with its bounding box for each sliced cell making the algorithm fast. The second uses the Mask R-CNN method which is more precise as it provides instance segmentation, giving more information as objects do not get localised with a bounding box. Instead, the output consists of exact pixel number and location of the desired object. Both methods can be used to capture the basketball through each frame in a video without the need to constrain the user to use a ball of a specific colour, therefore provides a much more practical solution.

Although the alternative methods mentioned provide a better quality feature, the original solution is still successful in capturing the objective of the feature. If this feature was to be deployed in the real world, with real world conditions and real users, the solution may prove to be unsuccessful as the colour of the ball is decided according to how it is perceived in the collected data. This means the feature is affected by the lighting in the facility, camera angle and other unknown variables. Considering this, with more time or undertaking further work the feature should be modified to use some of the said methods or other methods that do not make use of the colour of the ball.

9.3 Make/Miss Detection Evaluation

The performance of the make/miss detection is above expectation as the accuracy of the designed CNN proves to be quite high. The feature also performs well against the sample videos where it almost perfectly identifies the number of miss or made shots through each of the three. These results are all gathered from test videos that are similar to the videos used to train the network as they were recorded in the same facility, on the same day. As a result, this makes it easier for the network to produce accurate results as it is trained on similar and somewhat familiar data. This is the drawback of the make/miss detection feature. The dataset is not representative of the real world. In actuality, if the application is to be deployed, users will provide the application with input video that are of great variety and the training data does not reflect this. It is unfortunate that more data could not be generated and collected for reasons stated in section 4, as this would have made the dataset more representative of the real would therefore making the feature more practical. With more time or more ideal circumstances through the year and progression of this project, much more data would be generated in an attempt to more accurately represent the type of input that would be provided by users of the application.

Another aspect that may benefit the development and quality of this feature is to explore different models and strategies as only one is planned and developed. Comparing different models may lead to the realisation of a better method resulting in better performance although the original method already performs well. Fu, XB., et al. [18] present a camera-based basketball scoring detection (BSD) method with CNN based object detection and frame difference-based motion detection. This method entails implementing the YOLO model similar to Buric, M., et al. [17] to locate the position of the rim. Then, motion detection based on frame difference is used to detect whether there is any object motion in the area of the hoop to determine the basketball scoring condition. Although this method still utilises

a CNN the framed difference-based motion detection has potential of producing similar accuracies of the original solution, while making it easier to generalise to real world users as new, unfamiliar user video will not affect the motion of the ball. The use of a CNN may still encompass a large variety of training data being used to detect the position of the rim. This would be useful to investigate and compare with the existing solution with the possibility of a better quality feature.

9.4 Height Of Ball Detection Evaluation

Although the first method is preferred over the second, the solution for the height of ball detection feature is successful and suitable for the purpose of this project. With ideal circumstances, the video data would be re-recorded and the first method used, which enables users to know the actual height of the ball in feet when it is at its maximum position during the shot. This means users can compare each maximum ball height through each make or miss with ease by comparing the numbers given. Instead, users essentially undergo an eye test to determine if the ball is at the ideal maximum height given by the description on the application, at 3 feet above the rim or level with the top of the backboard. Through the utilised method users receive less detail, however are provided with a visualisation of the maximum height of the ball as a frame of the video. Some users may view this as more useful because in real life the user will not be able to determine what 3 feet above the rim is without measurement. Given the video data can be re-recorded, a better quality solution encompasses the utilisation of both methods where a reference object can be used to determine the maximum height of the ball in feet from the ground, as well as providing users with a frame of video where this maximum height is found during a shot. This gives the advantage of providing the users with both a visualisation and a more comparable height in feet, making the feature more practical than it is.

Combining this feature with the make/miss detection feature gives the benefit of

detecting the maximum height of the ball through each registered made or missed shot. This contributes greatly to the quality of the solution as providing users with a single frame of the maximum height of the ball through their entire shooting span, does not do much to aid in their understanding and improvement. This means to achieve a useful solution and complete functional requirement C6, this feature becomes dependent on the make/miss detection feature. Fortunately, the make/miss detection performs quite well resulting in this feature also producing expected results. Contrarily, this is a drawback of the height of ball detection as if the make/miss detection fails to produce accurate results, the user will not only be presented with false make or miss recordings, but also maximum heights of the ball that may not be during a free throw shot. Further work entails a review of the design of this feature to make it possible to detect the maximum height of the ball independently.

9.5 Post-Application-Use Performance Testing

Following the completion of the application and its underlaying functionality, the same two anonymous participants are asked to return to the same facility once it is available for booking. Each of the two, again, shoot twenty-five free throws each in an attempt to improve after their use of the application where videos of themselves are submitted shooting five free throws each. To reiterate, the goal of the project is to help users improve their free throw shot percentage using the feedback and analysis generated by the application. A prediction of the results is that both the advanced and beginner users will show improvement in their free throw shot percentage, however it is most likely that the beginner user will show a more drastic increase in their percentage compared to the advanced user. This is thought to be because of the beginner user's greater potential of improvement as the advanced user may already understand some of the insights and analysis given by the application. The results of the post-application-use performance tests is shown in the table below.

Participant	Ability Level	Pre-Use	Post-Use	Percentage Increase
Participant A	Advanced	72%	68%	-5.6%
Participant B	Beginner	40%	56%	40%

It is observed from the table that the improvement varies with each user. The advance user seems to have performed worse following the use of the application with a percentage increase of -5.6%, whereas the beginner user shows good improvement with a percentage increase of 40%. Similar to the initial prediction, the beginner users shows a large amount of improvement compared to that of the advanced user. Opposing the initial prediction, the advance user has not improved at all and has instead showed a slight decline in performance. The amount the advanced user has declined is minimal, corresponding to only one missed shot rather than a made shot. Due to this, the advanced user is considered to not have improved with more or less the same performance between pre and post application use.

We can consider the application and therefore the project to be somewhat successful in achieving its goal. The feedback and analysis generated by the application proves to be useful and effective in improving the beginner user's shooting. This may be because of the beginner users lack of experience in the sport of basketball resulting in a large amount of room for improvement. The analysis helps the user understand some of the underlaying mechanics behind free throw shooting and how to shoot better. That ultimately plays a part in the user making more of their shots. This is not the case for the advanced user. The analysis generated by the application may provide information that the advanced user is already aware of due to their years of experience, therefore proves to be ineffective and essentially results in the advanced user learning and developing near to nothing, hence no improvement in their free throw shooting. The application helps users to improve their free throw shooting through its use however not all users. Users who have several years of experience playing basketball may succumb to less improvement as these years increase. Users who are fairly new to the sport will experience a drastic increase

in their performance and can consider the application far more effective.

10 Conclusion

In this report, the development of a project focused on delivering an application that can be used to help improve performance in a user's free throw shooting, is documented through each of the application's features. The investigation of whether this application is significant in this improvement is conducted and conclusive results are found. Users with minimal experience and a lower ability level will find the application much more effective in the improvement of their free throw shooting, compared to users with vast experience in the sport. This is supported by the results of the beginner and advanced users improving their shooting by 40% and -5.6%, respectively. The amount of improvement can be considered to gradually decrease with the increase of a user's ability level. To support this claim more participants should be included in performance testing as only an advanced and beginner user were tested. Inclusion of both novice and intermediate users will provide a spectrum of results that can be further evaluated to determine the extent of the application's effectiveness. To cater to the advanced and, assumingly, elite users, the application could incorporate features which are specifically meant for users with several years of experience. The extension tasks, which had failed to be completed, could have contributed to a greater improvement seen in the advance user. Features like a computation of the user's release angle will not only produce a better quality application, but will also provide another useful insight into one of the most important aspects of a shot in basketball. Being that, the implementation of the primary extension tasks including this computation of a user's release angle, should prove to increase both the percentage increase of the advanced and beginner user. Although there are things the application could improve on, it is essentially successful in its goal of improving performance in shooting free throw shots. This is evidently from the results of the beginner user, but is only successful to an extent, as evident by the results of the advanced user.

The overarching aim of the project is to provide readily available technology to basketball players that can be used at their luxury, essentially reducing the significance of the co-dependent relationship between athletes and organisation's

trainers or facilities. Reviewing the finished product, the application does not seem powerful enough to replace or compete with a real trainer or high level equipment that organisations own. Consequently, the value of said relationship remains unchanged with real trainers being much preferred over our artificial trainer. There is much more flexibility and personalisation that comes with training with a real trainer. For the application to achieve a similar affect requires a more substantial solution. Given more time, this may be attainable however the effect of a real trainer remains unparalleled; post-completion of this project.

Further work entails a general shot analyser which extends the free throw shot analyser to anywhere on the basketball court. The current features of the free throw shot video analyser can be slightly modified and adapted to function for a shot analyser. This demands the re-recording of videos, so features are developed using data of shots taken from numerous places on the court, rather than just the free throw line. To improve the application's own ability to improve a user's shooting ability, interactive drills and mini-games could be incorporated into its design, giving more depth for the more experienced user. This combined with the saving and tracking of user progression will allow a greater sense of training, as the project somewhat fails to do so. The application provides insights into a user's shot to help understand the mechanics better but does not actively train the user through practice and instruction. Real-time free throw shot analysis can also be used to achieve this sense of training. With this, users receive instant feedback while still on the court where they can instantly make amendments and survey the results. Implementing this may involve mobile application development, perhaps using Swift for iOS devices, so a phone camera can be used. This is favoured over having the user record and upload their video via a web browser, then waiting for their analysis which is much more impractical.

References

[1] ESPN. NBA Player Stats 2019-20. [Online].
(URL https://www.espn.com/nba/stats/player/_/season/2020/seasontype/2/table/off
2020. (Accessed 30th December 2020).

[2] ESPN. NBA Team Regular Season Stats 2019-20. [Online].
(URL https://www.espn.com/nba/stats/team/_/season/2020/seasontype/2/table/offen
2020. (Accessed 30th December 2020).

[3] Markowitz, D. Can AI Make You a Better Athlete? Using Machine Learning to Analyze Tennis Serves and Penalty Kicks. [Online].
(URL https://daleonai.com/machine-learning-for-sports). 2020, July 16. (Accessed 10th October 2020)

[4] Przednowek, K., Krzeszowski, T., Przednowek, K., Lenik, P. A System for Analysing the Basketball Free Throw Trajectory Based on Particle Swarm Optimization. Applied Sciences [Internet]: MDPI AG, 2018, October 29. 8(11):2090.

[5] Rahma, AMS., Rahma, MAS., Rahma, MAS. Automated Analysis For Basketball Free Throw. 2015 IEEE Seventh International Conference on Intelligent Computing and Information Systems (ICICIS), 2015, pp. 447-453.

[6] Mullineaux, DR., Uhl, TL. Coordination-Variability And Kinematics Of Misses Versus Swishes Of Basketball Free Throws. Journal of Sports Sciences, 2010. 28(9), 1017–1024.

[7] Button, C., Macleod, M., Sanders, R., Coleman, S. Examining Movement Variability In The Basketball Free-Throw Action At Different Skill Levels. Research Quarterly for Exercise and Sport, 2003. 74(3), 257–269.

[8] Rosebrock, A. Ball Tracking with OpenCV. [Online].
(URL https://www.pyimagesearch.com/2015/09/14/ball-tracking-with-opencv/).

2015, September 14. (Accessed 30th October 2020)

[9] Verma, R. and Ali, J. A Comparative Study of Various Types of Image Noise and Efficient Noise Removal Techniques. International Journal of Advanced Research in Computer Science and Software Engineering, 3(10), 2013, pp. 618.

[10] Velickovic, P., et al. Graph attention networks. [Online]. (URL https://petar-v.com/GAT/). Proc. ICLR, 2018. (Accessed 20th April 2021)

[11] Dertat, A. Applied Deep Learning - Part 4: Convolutional Neural Networks. [Online]. (URL https://towardsdatascience.com/applied-deep-learning-part-4-convolu Towards Data Science, 2018, November 2017. Section 2.4. (Accessed 20th April 2021)

[12] Tabian, I., Fu, H., Khodaei, ZS. A Convolutional Neural Network for Impact Detection and Characterization of Complex Composite Structures. Sensors [Internet]: MDPI AG, 2019, November 12. 19(22):4933.

[13] Karpathy, A., et al. Large-scale Video Classification with Convolutional Neural Networks. In Proceedings of CVPR, 2014, pp. 1725–1732.

[14] Jüngel, M., Mellmann, H., Spranger M. Improving Vision-Based Distance Measurements Using Reference Objects. 2008. In: Visser U., Ribeiro F., Ohashi T., Dellaert F. (eds) RoboCup 2007: Robot Soccer World Cup XI. RoboCup 2007.

[15] Rosebrock, A. Measuring Distance Between Objects In An Image With OpenCV. [Online]. (URL https://www.pyimagesearch.com/2016/04/04/measuring-distance- 2016, April 4. (Accessed 2nd January 2020)

[16] Archana, M., Kalaiselvi Geetha, M. (2016) An Efficient Ball and Player Detection in Broadcast Tennis Video. In: Berretti S., Thampi S., Srivastava P. (eds) Intelligent Systems Technologies and Applications. Advances in Intelligent

Systems and Computing, vol 384. Springer, Cham.

[17] Buric, M., Pobar, M. and Ivasic-Kos, M. "Ball Detection Using Yolo and Mask R-CNN," 2018 International Conference on Computational Science and Computational Intelligence (CSCI), Las Vegas, NV, USA, 2018, pp. 319-323

[18] Fu, XB., Yue, SL. and Pan, DY. Camera-based Basketball Scoring Detection Using Convolutional Neural Network. Int. J. Autom. Comput. 18, 266–276 (2021).