

Advancing Sports Coaching Through Real-time Machine Learning and Computer Vision: A Comprehensive Analysis

ROSS USHER

COMPUTER SCIENCE (BSC)

SWANSEA UNIVERSITY | Bay Campus

Acknowledgments

The completion of this dissertation was made possible through the support and guidance of various individuals and institutions, to whom the author extends gratitude. Firstly, sincere appreciation is owed to Daniele Cafolla, whose expertise, encouragement, and insightful feedback were instrumental in shaping the direction and quality of this research. Special thanks are also due to the faculty and staff of Swansea University, whose resources and facilities facilitated the smooth progress of this study. Additionally, the author acknowledges the participants whose involvement and cooperation were essential for data collection. Moreover, heartfelt appreciation is extended to friends and family for their unwavering support and understanding throughout this academic journey. Their encouragement and patience have been invaluable. Furthermore, the author expresses profound gratitude to Talisa, whose unwavering support, love, and understanding have been a constant source of strength and motivation, making it possible to navigate through the challenges of academic life and achieve this milestone. Lastly, the author acknowledges the broader academic community for their contributions to the field, which have served as a foundation for this research endeavour.

Abstract

In the realm of sports coaching and performance analysis, the integration of machine learning and computer vision technologies has emerged as a transformative force. This dissertation presents a comprehensive exploration of a real-time sports coaching system designed to provide instant feedback on athlete movements. Leveraging the Mediapipe pose landmark model and a custom-trained classifier, the system analyses live video streams to detect and classify various punch types in real-time. Through extensive testing and analysis, the performance and implications of the system are thoroughly examined. Results demonstrate promising capabilities in providing real-time feedback, particularly in well-lit environments. However, challenges arise during difficult tasks such as tasks performed in challenging conditions and tasks that require a high level of precision such as identifying flexion and extension angles. Thus, highlighting areas for future refinement and exploration. The portability of the system, running on a laptop with a webcam, breaks barriers compared to existing systems requiring expensive and fixed equipment. This study contributes to advancing sports technology by critically evaluating system performance and implications, paving the way for future innovations in sports coaching and training methodologies.

Table of Contents

Acknowledgments	1
Abstract	1
Introduction.....	4
Project Definition.....	4
Aims and Objectives	4
1. Feasibility Assessment of MediaPipe Framework for Boxing Pose Detection: Investigate the feasibility of utilizing the MediaPipe framework as a foundational tool for real-time pose detection in boxing training scenarios.	5
2. Exploration of Pose Analysis Capabilities: Explore the capabilities and limitations of MediaPipe framework in providing actionable insights and feedback on boxing poses without the need for extensive algorithm development.	5
3. Evaluation of Real-Time Feedback Mechanisms: Evaluate the effectiveness and user experience of real-time feedback mechanisms implemented within the system, utilizing MediaPipe framework for pose detection.	5
4. Performance Comparison with Traditional Methods: Compare the performance and usability of the proposed system utilizing MediaPipe framework against traditional training methods in the context of boxing pose detection and feedback.	6
5. Validation and Proof-of-Concept Demonstration: Validate the potential utility and viability of the proposed system for enhancing traditional boxing training methods through empirical testing and proof-of-concept demonstrations.....	6
Structure of Document	6
Literature Review	7
Introduction to Real-Time Pose Detection Technology	7
Review of Existing Research on Pose Detection Technology in Sports	8
Identification of Research Gaps	9
Methodology.....	11
Live Video Capture	11
Detecting the Pose Landmarks in each Frame using Mediapipe.....	11
Analysing the Landmark Data.....	12
Punch classification using machine learning.....	12
Punch analysis using algorithms.....	14
Testing the Systems.....	15
Testing the speed of the systems	15
Testing the accuracy of the systems	16
General Information regarding the tests	21
Testing the traditional methods.....	21
Gathering qualitative data.....	23

Results	23
Testing the speed of the system components	23
Testing in a light environment	24
Testing in a dark environment	25
Testing at short range	26
Testing at long range	27
Analysis of Trends	28
Concluding Remarks.....	28
Testing the accuracy of the system components	29
Review of the Blazepose model output	29
Testing the classifier	32
Testing the algorithmic analysis.....	34
Testing human coaches	41
Feedback from human coaches about the technology	43
Discussion	44
Conclusion	45
Bibliography.....	47
Appendix	50
Code snippet for preprocessing the training data.	50
Code Snippet for training the CNN.	51
Code snippet for the “main system”.	54
Code snippet for the algorithmic analysis.	60
Transcript of the discussion with the coaches.	63

Introduction

In today's global society, sports hold a prominent position, captivating audiences worldwide and generating substantial revenue streams (*Revenue of the Big Five Soccer Leagues in Europe from 2012/13 to 2021/22, with a Forecast to 2023/24*, by League, 2024; *Statistics About The Most Popular Sports In The World* • Gitnux, 2024). However, behind all the glory displayed to the public, lies a fiercely competitive landscape, where athletes make tremendous sacrifices to achieve greatness (Cunningham, 2022).

Nowhere is this competitive spirit more evident than in the sport of boxing, where athletes grapple not only with opponents in the ring but also with the physical and mental toll of their chosen profession. For many boxers, the pursuit of success comes at a cost, with the rigours of training and competition taking a toll on their bodies and minds. It is from these challenges that the inspiration for this project arises – a burning desire to revolutionize traditional boxing training, to empower athletes to reach new heights of performance while safeguarding their physical well-being. The stories of athletes like Tony Jeffries, an Olympic medallist whose journey embodies the triumphs and tribulations of the sport, serve as reminders of the sacrifices made in the pursuit of greatness (*Timeline of A Typical Boxer's Life & Career* - by Tony Jeffries, n.d.).

To achieve the goal of enhancing traditional boxing training, this research project turns to the cutting-edge technology of real-time pose detection. While previous attempts have been made to leverage similar techniques (*BHOUT - The First Boxing Bag with a Brain*, n.d.), our approach represents a unique approach, with a focus on refining and optimizing the application of pose detection technology in the context of boxing. As of now, little research exists that explores the methodologies and techniques presented in this project, underscoring the significance of the efforts in both the realms of computer science and in the world of sports.

Project Definition

At its core, the project seeks to harness the power of real-time pose detection technology to provide boxers with immediate feedback on their technique, thereby aiding them in achieving peak performance while mitigating the risks of injury. While the original vision for this project was to develop a fully-fledged application that can assist boxers with their training, the complexity of such a project and the lack of research led the aims and objectives of the project to be adjusted. These adjustments will be highlighted later on but it is important to note this change in order to define what the project stands to be now. While the core foundation of the project is the same, the project is now focussed on the research of the technology and providing proofs that the technology can be used to achieve the original goal of making an application that can revolutionise traditional boxing training.

Aims and Objectives

To translate the vision into action, specific aims and objectives have been outlined that will guide our project towards its goal. From integrating MediaPipe framework for pose detection to offering real-time feedback and guidance to users, each objective is designed to contribute to the overarching mission of enhancing safety and performance in boxing.

The revised aims and objectives can be seen below. These aims and objectives have been developed in line with the new goal of providing evidence not only that the media pipe

framework can be used to provide real-time feedback during boxing training, but also that it can provide enhancements to traditional boxing methods. I personally believe that the achievement of these aims and objectives can still be classed as a success in regard to the motivation behind the project and will have a major impact on the world of computer science and sports.

The new aims and objectives are:

1. Feasibility Assessment of MediaPipe Framework for Boxing Pose

Detection: Investigate the feasibility of utilizing the MediaPipe framework as a foundational tool for real-time pose detection in boxing training scenarios.

1.1 Evaluate the compatibility of MediaPipe's pose detection algorithms with boxing-specific movements and poses, considering factors such as accuracy, robustness, and adaptability.

1.2 Assess the technical requirements and constraints of integrating MediaPipe framework into existing boxing training setups, including hardware compatibility and computational resources.

1.3 Determine the potential advantages and limitations of using MediaPipe framework as a standalone solution for pose detection in boxing, compared to alternative technologies or methods.

2. Exploration of Pose Analysis Capabilities: Explore the capabilities and limitations of MediaPipe framework in providing actionable insights and feedback on boxing poses without the need for extensive algorithm development.

2.1 Investigate the extent to which pose landmark data extracted from MediaPipe framework can be utilized to assess the correctness, precision, and effectiveness of boxing techniques.

2.2 Explore methods for leveraging MediaPipe's pose detection outputs to provide users with informative feedback on their boxing poses.

2.3 Identify potential challenges and opportunities in translating pose detection data into meaningful feedback and guidance for boxing athletes, considering factors such as interpretability and user acceptance.

3. Evaluation of Real-Time Feedback Mechanisms: Evaluate the effectiveness and user experience of real-time feedback mechanisms implemented within the system, utilizing MediaPipe framework for pose detection.

3.1 Solicit feedback from boxing athletes and trainers to gauge the perceived value and practicality of real-time feedback mechanisms in enhancing training outcomes and performance.

4. Performance Comparison with Traditional Methods: Compare the performance and usability of the proposed system utilizing MediaPipe framework against traditional training methods in the context of boxing pose detection and feedback.

4.1 Conduct comparative evaluations to assess the speed, accuracy, and reliability of pose detection results obtained using MediaPipe framework versus manual assessment by experienced trainers.

4.2 Analyse the overall user satisfaction and perceived benefits of the proposed system compared to traditional training methods, with a focus on practicality, accessibility, and scalability.

5. Validation and Proof-of-Concept Demonstration: Validate the potential utility and viability of the proposed system for enhancing traditional boxing training methods through empirical testing and proof-of-concept demonstrations.

5.1 Design and execute validation experiments to assess the system's performance and effectiveness in real-world boxing training environments.

5.2 Analyse the outcomes and insights gathered from validation experiments to draw conclusions regarding the feasibility and efficacy of using MediaPipe framework for boxing pose detection and feedback, informing future research and development efforts in the field.

Structure of Document

The dissertation is structured into distinct sections that systematically explore and evaluate the feasibility and effectiveness of utilizing MediaPipe framework for real-time pose detection and feedback in the context of boxing training. Following this Introduction, the Literature Review section delves into existing literature on pose detection technology, sports science, and boxing training methodologies, providing a comprehensive background for the research. The Methodology section details the approach taken in the study, including the implementation of the MediaPipe framework, data collection processes, algorithm development, and testing protocols. Subsequently, the Results section presents the findings of the empirical analysis, showcasing the performance and effectiveness of the proposed system. The Discussion section contextualizes the results within the broader literature and theoretical framework, interpreting their implications for boxing training and pose detection technology. Finally, the Conclusion summarizes the key findings of the dissertation, reflects on the research process, and suggests potential avenues for future research and development in the field.

Literature Review

Introduction to Real-Time Pose Detection Technology

Pose detection technology plays a pivotal role in modern computer vision applications, offering real-time analysis of human body poses in images and videos. As described by (Walia, 2022) pose detection involves the identification and classification of key body parts and joint positions to allow for the visualization of human poses. This technology has many uses across a variety of fields, including sports training, where it can provide valuable insights into athletes' movements and techniques. Within sports training, particularly in disciplines like boxing, the integration of pose detection technology can offer numerous advantages in enhancing training methodologies and assessing performance.

There are numerous ways to approach pose detection. Upon conducting research, two main approaches have been identified: Inertial Measurement Units (Ahmad et al., 2013) and Computer Vision. For this project, computer vision is obviously a greater focus, however, it is important to look at the results of IMU technology to draw comparisons across a variety of approaches. With regards to computer vision approaches, Object Classification and Key Landmark Identification were the two main approaches found during research. These approaches will be spoken about later on in the document but mentioning them here is necessary to provide the context for information in this section. The context being that the choice of approach for this project is Key Landmark Identification. To achieve this the Google Mediapipe framework will be used (*MediaPipe | Google for Developers*, n.d.).

When considering the modelling of the human body for pose detection, various approaches exist, including kinematic, planar, and volumetric models, as outlined by (Gong et al., 2016). For this project, the kinematic model proves most suitable, considering its effectiveness in representing limb movements and joint interactions. The kinematic model conceptualizes the human body as a hierarchical structure of limbs and joints, akin to a tree structure, wherein the movement of each limb is dependent on its parent limb's motion. This hierarchical representation facilitates the interpretation of body poses and movements, forming the basis for pose detection algorithms.

In recent years, pose detection technology has found practical applications in fitness and sports training, exemplified by products like Tempo, an AI-powered home gym membership (*Award-Winning AI-Powered Home Gym Membership*, n.d.). Tempo leverages real-time pose detection to provide users with immediate feedback on their workouts, offering insights into form and range of motion. These applications will be reviewed further on in this document.

Google's MediaPipe framework emerges as a strong platform for integrating pose detection technology into sports training systems (*MediaPipe | Google for Developers*, n.d.). Developed as an open-source project, MediaPipe offers a versatile toolkit for processing sensory data, including visual inputs (Lugaresi et al., 2019). At its core, MediaPipe employs machine learning algorithms to identify key points on the human body, known as landmarks, facilitating real-time pose estimation and tracking. The Pose Landmark Detection task within MediaPipe enables the detection of 33 key points on the body, forming the basis for kinematic modelling and pose analysis (Bazarevsky et al., 2020).

Implementing MediaPipe for pose detection entails configuring the system to process input images or live video streams, utilizing pretrained models such as BlazePose. The framework

provides flexibility in adjusting confidence thresholds for landmark detection and offers various running modes, including real-time processing (*MediaPipe Solutions Guide*, n.d.). Additionally, MediaPipe's compatibility with popular libraries like OpenCV (*Home*, n.d.) facilitates seamless integration into existing software ecosystems, further enhancing its usability and scalability.

In the context of boxing training, the application of pose detection technology offers unique opportunities for assessing and refining athletes' techniques. By capturing and analysing real-time movement data, coaches and athletes can gain valuable insights into posture, alignment, and movement dynamics, enabling targeted interventions and performance optimizations. However, the effectiveness of pose detection technology in boxing must be evaluated against traditional coaching methods, considering factors such as the coach-athlete relationship and the ability to provide personalized feedback during training and competitions.

To assess the feasibility and efficacy of integrating pose detection technology into boxing training, a comprehensive analysis of its strengths and weaknesses is essential. Drawing from existing literature and first-hand experiences of athletes and coaches, this section aims to provide insights into the potential benefits and limitations of leveraging pose detection technology in sports training contexts. Through empirical investigations and comparative studies, the viability of using MediaPipe and similar frameworks for real-time pose detection in boxing can be evaluated, paving the way for enhanced training methodologies and performance assessment strategies.

Review of Existing Research on Pose Detection Technology in Sports

Before conducting any new research on the use of Pose Detection technology in the sport of Boxing, reviewing the research that has already been done was paramount. Not only was it important to look at how the technology has been used in Boxing, it was also important to review the current techniques used with the world of sports as a whole. In this section, a couple of existing products on the market will be reviewed, as well as some academic research that has been conducted on the use of Pose Detection technology within the world sports.

A research paper looking at the use of Inertial Measurement Units for classifying Boxing punches generated some interesting results (Hanada et al., 2022). They achieved a detection accuracy of 98.8% with classification accuracies ranging from 98.9% to 91.1%. Not only that, but they were also able to classify punches with an upper limit of a tenth of a second. This is a remarkable result and really highlights the power of IMU technology. When looking at the results for this project, it is important to keep these figures in mind as they set a very strong benchmark for the accuracy and speed of a system that aims to detect and analyse poses in real time.

Another research paper that was studied is a paper looking at the classification of Boxing punches using Object Classification (Stefański et al., 2023). As mentioned previously, this is a method of using computer vision in order to identify and classify objects. In the case of this study, the objects represented the type of punch being thrown. This is a fascinating approach to the problem but is one that is very similar to the approach taken in this project, thus, the results achieved by these researchers is extremely relevant. The results that were achieved showed a classification accuracy metric of 94%, 84% and 81% of the F1 score (*Understanding and Applying F1 Score: AI Evaluation Essentials with Hands-On Coding Example*, n.d.). Comparing these scores to the paper looking at IMU technology, it can be seen that the accuracy scores range from a 4.8% decrease to an upper bound of a 17.8%. This is quite a large range and whilst the better of the scores performs very well compared to IMU, the not-so-good scores are quite a

way off the accuracy of IMU. Improving on these results within this project is paramount to driving forward the research into computer vision techniques.

As this project centres around the use of the Mediapipe framework (*MediaPipe Solutions Guide*, n.d.), reviewing a paper that uses the technology provided a fundamental basis for the project and was invaluable for setting the levels that this project needed to exceed. The paper (Pauzi et al., 2021) looks at using the BlazePose model (Bazarevsky et al., 2020) in order to detect and analyse key landmarks taken from a video of the human body. A great similarity with this project is that the system focuses on videos containing high-speed movements. A lot of the lessons learned in the project from that paper with regards to high-speed poses played a crucial role in the smooth implementation of this project. One of the key aspects that the paper highlights is the analysis of the velocity and angles of joints. For this project, having that foundation in place was important as it allows for a large amount of further analysis and so being able to build on the foundation that this paper provided was a great aid to this project. As done with the previous approaches, the results presented in the paper were carefully reviewed. The paper showed that the system that had been implemented came within 10% of their own IMU experiments. While the results of the IMU experiments were not shown, having seen the power of IMU technology and after closely scrutinising their experimental methods and the IMU technology they used, it can be said that achieving within 10% of that accuracy using computer vision demonstrates that the use of Mediapipe can draw very good detection results.

In conclusion, the review of existing research on pose detection technology in sports provides valuable insights into the advancements of utilizing such technology, particularly in the context of boxing. Studies employing Inertial Measurement Units (IMU) showcased impressive detection accuracy and speed, setting a high benchmark for real-time pose analysis. Similarly, research utilizing object classification methods highlighted the potential of computer vision techniques, albeit with varying degrees of accuracy compared to IMU technology. Furthermore, investigations into frameworks like Mediapipe and models such as BlazePose offer foundational knowledge crucial for enhancing the precision and efficiency of pose detection systems, especially in capturing high-speed movements. Overall, these findings underscore the importance of building upon existing research to further advance the capabilities of pose detection technology in sports, including boxing.

Identification of Research Gaps

In order for this project to make a significant impact, it is important to identify the gaps in the existing research of the technology. While it is important to look at existing research from academics, it is also crucial to look at the work done in industry. This section will look at both of these and explain where exactly there is a gap for this project.

One of the most relevant papers to this project, which was reviewed earlier in this document, was a paper (Pauzi et al., 2021) that looks at using the Mediapipe framework for analysing poses during high-speed movements in videos. While this paper does have a lot of useful findings, there are many gaps in this research that need to be filled. One of the main gaps being that the study was conducted on pre-processed videos. The use of pre-processed videos does highlight the power of computer vision for pose detection, however, it fails to prove the use of such software in real-time systems. Where this project provides further insights into the technology is through research of the technology in real-time environments. Testing the technology in such environment is crucial to assessing its suitability for industry and is research that could not been found to exist.

In order to identify gaps in the technology used in industry, two upcoming and very competitive systems in this section of industry have been analysed.

The first system, Tempo (*Award-Winning AI-Powered Home Gym Membership*, n.d.), is a pose detection technology that focuses on the activity of weight lifting. While no technical documentation could be found on this system, there is information on their website that explains the use of coloured weights to aid the visualisation system. It is assumed that the system will use some sort of object classification technology in order to determine the position of the weights. A technology similar to that described in the paper reviewed previously (Stefański et al., 2023). One of the main gaps associated with this type of system is the requirement of additional equipment. There are many downsides to requiring additional equipment. Firstly, the equipment is not very easy to transport, thus, limiting the ability of people to use the technology whilst away from home. In a study conducted by Hilton hotels (Winter, 2023), it can be seen that 33% of respondents said that they prioritise exercising whilst on holiday. With figures like this, it's clear that a portable solution would have great significance in industry. Another issue faced by the requirement of additional equipment is the price that comes with the equipment. For the most basic package that Tempo offers, "Tempo Core", there is a cost of \$713 (USD) for the package. While this may be worth it for a lot of people, for some people the cost makes it completely inaccessible. As we established earlier, many of the target audience for this project (Boxers) go through a lot of struggles. One of the key areas pointed out in the story of Tony Jeffries (*Timeline of A Typical Boxer's Life & Career - by Tony Jeffries*, n.d.) that was mentioned previously is the lack of money that many boxers have throughout their career, especially at the beginning. This is a major gap that needs to be exploited by this project.

The second system that was reviewed is an AI-powered punch bag that has very similar goals as this project, to help athletes improve their punching technique. The system, BHOUT (*BHOUT - The First Boxing Bag with a Brain*, n.d.), uses a combination of both computer vision and sensors in order to track the athlete's punching technique and fully analyse the punches being thrown. Similarly to the last system, no technical documentation could be found for the system, however, the website does offer quite a lot of information about the technology. Some of the key

points that are highlighted on the website is the use of landmark identification to track the body. This is a very similar approach to that of this project, except, the system also uses sensors to aid the technology. This is very intriguing and was the spark for a lot of questions about why the sensors are needed to aid the system. The system (as seen in Figure 1) is one complete unit that contains the punch bag, the camera set up and the sensors.

What is very apparent from looking at the system is that it is by no means easily transported. As discussed, this is a major gap in the current technology available that this project aims to exploit. Also similar to the last system, the equipment that is required costs \$1875 (USD) for the

Figure 1
BHOUT Equipment Set-Up



Note. From Bhout. (n.d). Setup of Equipment [Photograph]. Bhout.
<https://www.bhout.com>

most basic kit. This is an extraordinary amount for the average athlete to pay and contributes to the need for a low-cost solution.

In conclusion, there are many technologies in industry as well as in academic research that have very similar approaches to that of this project. Where this project differs from technology in industry is through the sole use of computer vision with no extra equipment necessary. Additionally, where this project differs from academic research is using real-time pose detection and analysis. Providing evidence that this technology is sufficient for its purpose will allow for very low-cost and extremely portable systems to be developed that will make the technology accessible by almost anyone. There is a clear gap for this project in the existing research and a success for this project will allow for major developments of the technology.

Methodology

To support the veracity of this project, this section will provide a full account of the research. There are many aspects to the research that make up the full project, therefore, this section will tackle each aspect individually to aid the understanding of what was done and how it was achieved. Before delving into the different sections, some general information about the methodologies is important to note. The project was completed using the Python programming language. There are many reasons for this, for example, the vast number of libraries available to developers that can be used for machine learning projects, specifically the Google Mediapipe library that is being tested has great support within Python.

Live Video Capture

The first major section to talk about is the live video capture system. The main focus on the project is the suitability of Mediapipe for real-time pose detection. As such, a live video capture is needed to be able to test this. For this project, the OpenCV library was used in order to capture the live video from a webcam connected to the host's device. The webcam being used had no specific requirements as the importance of this research is to investigate the availability of such a system to an audience with limited resources. However, for this project, the webcam used had a resolution of 720p and a frame rate of 30 fps.

Detecting the Pose Landmarks in each Frame using Mediapipe

The second major part of the project was creating a small system that can turn the live video capture into a set of landmarks that can be analysed. This is where Mediapipe comes in. The Mediapipe task that was used for this project is a pose landmark detection task that Google provides to developers. The model is a variant of the BlazePose model that was described previously in this document. While a new model could be produced for use in this project, it would be unnecessary to do so as the accuracy of the results provided by this model would be extremely difficult to beat. As seen in the previous research, the provided Mediapipe model performs very well on videos containing high-speed movements when compared to IMU technology. This system works by taking each frame from the live video capture and using the provided model to generate a set of 33 landmarks on the body in the frame. For each frame, the 33 landmarks are processed in real-time by the next system that will be discussed.

Analysing the Landmark Data

Once the live video capture was set up and the detection of pose landmarks was implemented, the two systems were used by what can be classed as the “main” system. This system utilises the different systems created during the project to produce a system that can capture, detect, and analyse the pose landmarks in real-time. The system is broken into three main components, the capture which has already been described, the detection of landmarks which again has already been described, and finally, the analysis of the landmarks. As mentioned previously, the aims of this project have changed since the initial document that was produced. As the aim has changed to testing the suitability of Mediapipe for use in real-time systems as opposed to producing a fully-fledged product, the functionality of the produced system also had to change. This significantly changes the deliverables of the analysis system. For this project, the analysis section was made up of two systems, one of the systems is a bespoke machine learning model that has been trained to detect the type of punch that is being thrown by the user. The other system consists of algorithms that are used to calculate and verify several characteristics of the punch. In the following subsections, a detailed breakdown of the systems that make up the Analysis component of the “main” system will be provided.

Punch classification using machine learning.

Creating the model used to detect the punches was quite a long process. This subsection will break down the steps that were taken to develop the model (assuming of the reader some basic knowledge of training ML models).

Capturing the data used for training and testing the model.

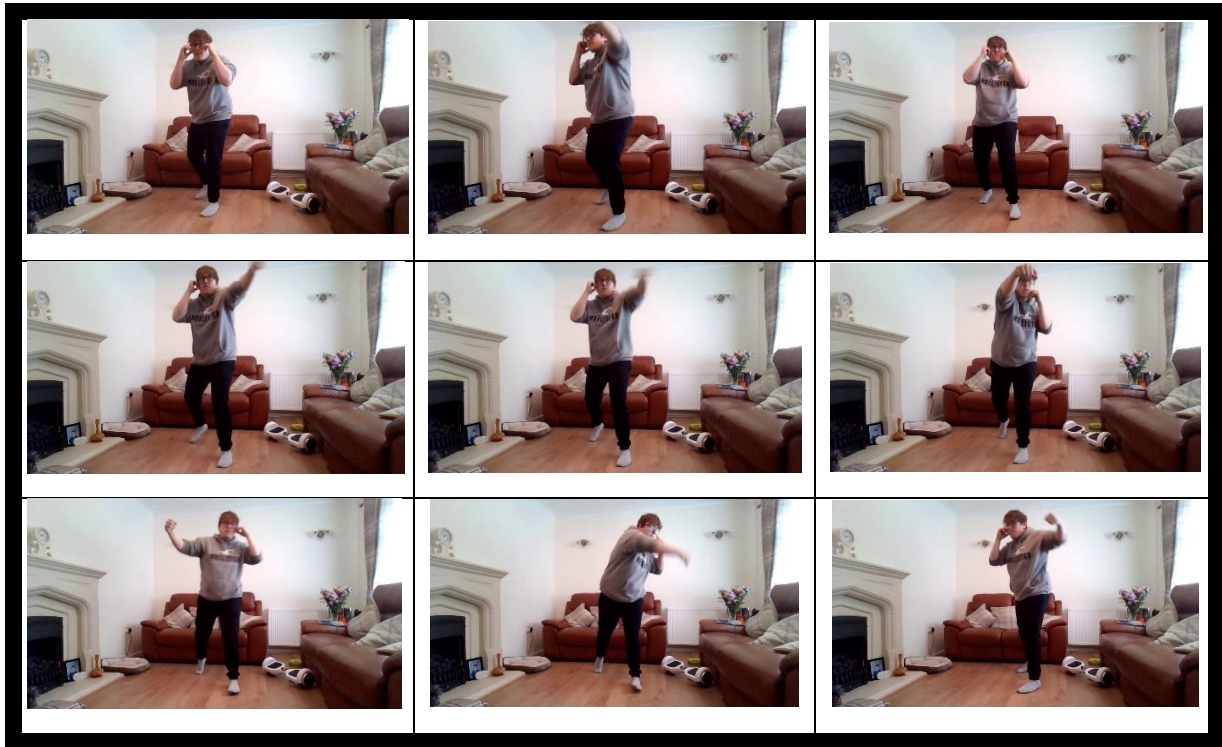
To capture the data used for training the model, a video was filmed of a participant performing 50 movements. The video was filmed using a laptop webcam in a well-lit room. The video contained 10 repetitions of each of the 5 movements that the model was trained to detect. These include a Jab, Hook, Straight, Upper Cut and finally No Punch. The video was saved as an MP4 file in a local folder ready for preprocessing.

Preprocessing the data used for training and testing the model.

The preprocessing of the data before training/testing was straightforward for this project. The first step was to crop the video to remove unnecessary footage that was captured. This saved a lot of time when labelling the data. The next step was to break down the video into individual frames and save the frames as .jpg images inside a folder. From the video that was recorded, 2549 frames were extracted. A few examples of the captured frames can be seen in figure 2 below.

Figure 2

Examples of the captured training data.



Using a python script that was developed, the folder was iterated over. For each file, the frame was displayed, and a label was manually inputted into the system. This method was chosen due to a lack of publicly available datasets and as one might expect, the process of labelling the data took an extremely long time to complete. This was a major set-back to the completion of the project that was overlooked initially. Once the data had been labelled, the images were then analysed by the BlazePose model, this generated 33 landmarks for each image and stored the set of landmarks along with its label inside an array. Once all images had been processed, the data was then split into training data and testing data at a ratio of 80%:20% respectively. Once the training and testing data had been established, the data was then transformed into a NumPy array to allow for use as training data and testing data. In total, 4 NumPy arrays were created (X_train.npy, y_train.npy, X_test.npy, y_test.npy) which held the landmark data and the label data for the training and testing data (X = Landmarks, y = Labels). These NumPy arrays were saved inside a folder ready to be used to train and test the model.

Training and testing the model.

Once the arrays containing the training and testing data were created, the model could then be trained. The model that was trained is a Convolutional Neural Network. A Convolutional Neural Network (CNN) is the ideal choice for classifying punch types based on landmark data extracted using MediaPipe due to its ability to effectively capture spatial relationships and patterns in the input data. CNNs are specifically designed for tasks involving spatial data, such as images or sequences, making them well-suited for processing the coordinates of body landmarks (Yamashita et al., 2018). By leveraging convolutional layers to extract features from the landmark data and pooling layers to down sample and retain important information, CNNs can

learn hierarchical representations that are essential for accurate classification. Additionally, CNNs inherently handle the translation invariance of the data, making them robust to variations in the positioning of the landmarks across different instances. Overall, a CNN offers a powerful and efficient solution for classifying punch types. The process begins by loading the training and testing data, which consists of arrays representing the coordinates of 33 landmarks in 3D space. The data is reshaped to match the input shape expected by the CNN model. Next, the unique punch-type labels are encoded into a numerical format using a Label Encoder from the sklearn library and converted into one-hot encoding (Pedregosa et al., 2011). The CNN model architecture is then defined, comprising convolutional layers with max pooling for feature extraction, followed by fully connected layers for classification. To combat overfitting, a dropout layer is also included as well as a learning rate scheduler. The model is compiled with the Adam optimizer and categorical cross-entropy loss function. It is then trained on the training data, with validation performed on the test data. The data is trained over 125 epochs with a batch size of 16. After training was complete, the model's performance is evaluated on the test set, and the accuracy is displayed. The training accuracy and loss curves are also displayed. Finally, the trained model is saved for future use. For reference, a code snippet of this process can be found in the appendix of this document.

Using the trained classifier

Within the system, the classification of each frame plays a crucial role in generating a single prediction regarding the type of punch being executed. As the webcam captures successive frames, the system processes the landmarks extracted from each frame individually. These landmarks are then fed into the trained classifier, which evaluates them and assigns a probability distribution across the different punch classes. By aggregating the classifications from multiple frames over a short time window, the system infers the predominant punch type being performed. This temporal aggregation enhances the robustness and accuracy of the prediction, compensating for any inconsistencies or noise present in individual frames. Ultimately, this approach enables the system to provide a reliable and timely assessment of the ongoing punch activity, empowering coaches and athletes with actionable insights for optimizing training techniques and refining performance. A code snippet of this is available in the appendix of this document.

Punch analysis using algorithms.

The second part of the analysis system uses a traditional algorithmic approach to determine some correctness aspects of the punch. While the system does not verify the correctness of all aspects of the punch, certain characteristics have been targeted for different movements to find out how well the approach holds up in the environment established within this project.

The first to be implemented was a basic algorithm that calculates the speed of a Straight punch or Jab punch. The algorithm takes the position of the punching hand at the start of the punch and the position of the hand at the point of maximum extension. The duration between these points is then calculated by using the frame rate and the number of frames that occurred over that time. With this information, the speed of the punch is then calculated. While this algorithm is very basic, it is a crucial part of the research as it will help to evaluate the capability of Mediapipe to accurately measure distance within a 3D space.

The next algorithm that was implemented checks some other correctness aspects of the jab and the straight punch. Specifically, it ensures that the trajectory of the punch follows a straight

line. This is done by calculating the equation of the line between the origin of the punch and the point of maximum extension. The perpendicular distance from each of the other points is calculated and this is used to determine if the punch followed a straight trajectory. As highlighted previously when looking at correct form in Boxing, it is important for technique that a jab or a straight punch follows a straight trajectory to the target. Therefore, this algorithm serves two purposes for the research, not only does it begin to develop fundamental algorithms for determining the correctness of a punch, but it also serves to test the practicality of using the approach within a real-time system.

The next two algorithms that were implemented are focussed on the Guard pose. The first of the two aims at checking if the back heel is raised when in the guard position. This is done by identifying the back heel and the back foot and comparing the heights of the two landmarks. This is something that was discovered in the Literature Review as crucial for having good technique. The second of the two aims at checking that the hands are positioned near the head when guarding. This is done by calculating the distance of each hand to its corresponding ear using the generated landmarks. Again, this was justified in the Literature Review. Both algorithms have a vital role in the research as they allowed the precision of Mediapipe to be tested where landmarks are extremely close together. It also allowed for some testing to be done on the depth perception of Mediapipe.

The final algorithm that was implemented focussed on the flexion and extension angles of the body parts. Due to the nature of this project laying within the realm of Boxing, a selection of body parts has been chosen that are common indicators of correctness of Boxing poses. The algorithm calculates the angles for the left arm, the right arm, the left leg, and the right leg. Being able to accurately calculate these angles is a crucial aspect of the proposed system.

For all these algorithms, the code snippets are available in the Appendix of this document.

Testing the Systems

Now that an understanding of the different systems in the project have been put in place, it is important to talk about how the systems have been used to produce results. This section will explain the process that was taken to test the approach of using landmark detection with the Mediapipe library to analyse Boxing movements in real-time. The section will cover all significant tests that were conducted on the systems.

Testing the speed of the systems

A crucial characteristic of any real-time system is of course its processing speed. Therefore, it was necessary to test the speed of the systems that have been developed during this project to evaluate the feasibility of such as system.

The three main components that were included in the speed test were; the Mediapipe detection, the punch classifier, and the algorithmic analysis. Three components were tested in real-time. For each frame that was captured in real-time, the Mediapipe system would detect the landmarks and pass this onto the classifier and the algorithmic analyser. These systems would each then produce an output. The processing time for each of the three components was measured individually. This was done by capturing the system time before and after a component was active. The difference in these times was calculated as the length of time taken to process the frame. The results for each frame were stored and upon completion of the real-time test, a report was generated.

The generated reports contained three graphs representing the processing times of each component for each frame in the test. The mean, minimum and maximum processing times were calculated for each of the components. The reports were analysed in order to identify any patterns or trends in the data or anything that may be of interest such as an anomalous result. The mean, min and max processing times were also calculated for the system as a whole and these values were used to determine the feasibility of the system for real-time use with regards to the processing speed. To determine this, a threshold of 0.016 seconds was used, meaning that if the system has a whole can be said fairly to process frames in a time less than the threshold then it would be feasible as a real-time system. This threshold was decided based upon the desire in the video game sector to have “ultra realistic visual quality” running at 60 fps (Andreev, 2010). This seems to be an accepted standard across the video game industry where it is well known that high quality graphics and ultra-high frame rates are what set leading video game developers apart from the rest. Hence, the threshold of 0.016 seconds per frame (60 fps) has been chosen.

One consideration that was made when developing the tests for the systems is that the goal of the tests is to evaluate the feasibility of the Mediapipe approach for a real-time Boxing application. Therefore, the system should be tested in a variety of environments that replicate the possible environments that an application of this type would be used in. For that reason, four tests were conducted on the systems which cover the four main environments that an application like this would be used in.

The first environment that the test was conducted in was in a well-lit environment. This environment would simulate an athlete using the system outside whilst training during the day or training at night in a well-lit gym. This test is crucial to provide an initial benchmark for the system and for evaluating the speed of the system in a highly likely environment that a potential application may encounter.

The second environment that the system was tested in was a poorly lit environment. This environment simulates an athlete training outside at night or in a poorly lit gym. This is an obvious test to complete, especially after testing in a well-lit environment. Being able to compare these results will provide great analysis of the systems performance under different lighting conditions.

A vital aspect of any sort of vision is the ability to perceive depth and how depth affects how one perceives the distances between objects, as well as the size of objects. This is especially crucial for the analysis that needs to be done in a computer vision project like this one. For this reason, the other two environment that the system was tested in was an environment where the participant of the test was at close range to the webcam, followed by an environment where the participant was at long range to the webcam. This will allow for the opportunity to evaluate the ability of a potential application to remain efficient across a variety of short range or long-range environments.

Testing the accuracy of the systems

Another vital characteristic of any real-time system is its accuracy. Therefore, it was necessary to test the accuracy of the systems that have been developed during this project to evaluate the viability of this approach to a real-time pose detection and analysis system.

Mediapipe Detection System

As discovered in the Literature Review of this document, the Blazepose model has immense accuracy when processing videos. As such, the accuracy of the model does not need to be retested. Only the speed of the system needed to be tested as there was no current research found that measures its speed in real-time systems.

It is important to consider that the results of the Mediapipe detection system directly affect the results of the classifier and the algorithmic analysis. As such, achieving great accuracy from the classifier and the algorithmic analysis supports the claim of achieving great accuracy from the Mediapipe detection system.

Punch Classifier

As mentioned, the classifier model was trained using a training/test split of 80%:20%. Once trained, the test data was used to check the accuracy of the model. This is an excellent representation of the accuracy of the model. To further test the model, a series of tests were performed in real-time. These tests aim to evaluate the accuracy of the Mediapipe detection system and the punch classifier as a combined system. The accuracy of a single system cannot be identified through these tests; however, testing the accuracy of the combined system provides significant evidence to support or oppose the use of Mediapipe within a real-time boxing application.

Similar to the speed tests, a variety of environments were used to test the system. The tests that were completed can be seen in Figure 3 below.

Figure 3

Description of Accuracy Tests for the Classifier.

<i>Test Number</i>	<i>Environment</i>	<i>Expected Class</i>	<i>Test Number</i>	<i>Environment</i>	<i>Expected Class</i>
1	Well-lit	Jab	21	Close Range	Jab
2	Well-lit	Jab	22	Close Range	Jab
3	Well-lit	Straight	23	Close Range	Straight
4	Well-lit	Straight	24	Close Range	Straight
5	Well-lit	Hook	25	Close Range	Hook
6	Well-lit	Hook	26	Close Range	Hook
7	Well-lit	Upper Cut	27	Close Range	Upper Cut
8	Well-lit	Upper Cut	28	Close Range	Upper Cut
9	Well-lit	No Punch	29	Close Range	No Punch
10	Well-lit	No Punch	30	Close Range	No Punch
11	Dark	Jab	31	Long Range	Jab
12	Dark	Jab	32	Long Range	Jab

13	Dark	Straight	33	Long Range	Straight
14	Dark	Straight	34	Long Range	Straight
15	Dark	Hook	35	Long Range	Hook
16	Dark	Hook	36	Long Range	Hook
17	Dark	Upper Cut	37	Long Range	Upper Cut
18	Dark	Upper Cut	38	Long Range	Upper Cut
19	Dark	No Punch	39	Long Range	No Punch
20	Dark	No Punch	40	Long Range	No Punch

The test suite seen in Figure 3 contains the Test Number, Environment and Expected Class. The environment refers to the environment that the test was conducted under. The expected class refers to the type of movement that can be seen in the test and therefore, the class that is expected to be chosen for the test. The results of these accuracy tests will be evaluated later in this document.

Algorithmic Analysis

Testing the accuracy of the Algorithmic Analysis was done using a similar methodology as the Classifier tests; however, the expected result will be the feedback that the system should produce for the user about the movement.

As the algorithms are fixed decisions, only one test needs to be completed for each of the expected feedback as these tests are more focused on proving the validity of the algorithms as well as testing the combined system of the Mediapipe detection system with the algorithmic analysis.

The tests that were completed can be seen in Figure 4 below.

Figure 4:

Description of Accuracy Tests for some of the Algorithmic Analysis.

Test Number	Environment	Movement	Expected Feedback
1	Well-lit	Jab	“The punch was thrown at a speed of 0.33 m/s” “The punch was thrown in a straight trajectory”
2	Well-lit	Jab	“The punch was thrown at a speed of 0.33 m/s” “The punch was not thrown in a straight trajectory”
3	Well-lit	Straight	“The punch was thrown at a speed of 0.33 m/s” “The punch was thrown In a straight trajectory”
4	Well-lit	Straight	“The punch was thrown at a speed of 0.33 m/s”

			"The punch was not thrown in a straight trajectory"
5	Well-lit	No Punch	"Back heel is raised"
6	Well-lit	No Punch	"Please raise your back heel!"
7	Dark	Jab	"The punch was thrown at a speed of 0.33 m/s" "The punch was thrown in a straight trajectory"
8	Dark	Jab	"The punch was thrown at a speed of 0.33 m/s" "The punch was not thrown in a straight trajectory"
9	Dark	Straight	"The punch was thrown at a speed of 0.33 m/s" "The punch was thrown In a straight trajectory"
10	Dark	Straight	"The punch was thrown at a speed of 0.33 m/s" "The punch was not thrown in a straight trajectory"
11	Dark	No Punch	"Back heel is raised"
12	Dark	No Punch	"Please raise your back heel!"
13	Close Range	Jab	"The punch was thrown at a speed of 0.33 m/s" "The punch was thrown in a straight trajectory"
14	Close Range	Jab	"The punch was thrown at a speed of 0.33 m/s" "The punch was not thrown in a straight trajectory"
15	Close Range	Straight	"The punch was thrown at a speed of 0.33 m/s" "The punch was thrown In a straight trajectory"
16	Close Range	Straight	"The punch was thrown at a speed of 0.33 m/s" "The punch was not thrown in a straight trajectory"
17	Close Range	No Punch	"Back heel is raised"
18	Close Range	No Punch	"Please raise your back heel!"
19	Long Range	Jab	"The punch was thrown at a speed of 0.33 m/s" "The punch was thrown in a straight trajectory"
20	Long Range	Jab	"The punch was thrown at a speed of 0.33 m/s"

			"The punch was not thrown in a straight trajectory"
21	Long Range	Straight	"The punch was thrown at a speed of {SPEED} units/second" "The punch was thrown In a straight trajectory"
22	Long Range	Straight	"The punch was thrown at a speed of {SPEED} units/second" "The punch was not thrown in a straight trajectory"
23	Long Range	No Punch	"Back heel is raised"
24	Long Range	No Punch	"Please raise your back heel!"

In Figure 4, many of the expected feedback contain feedback about the speed of the punch. To calculate the expected speed, the punches were thrown at a distance of 1m from a target. A 1m ruler was laid along the floor to guide the punch, this can be seen in figure 5 below. The punch was thrown at a slow pace to have full control over the punch. The duration of the punch from the origin to the target was timed at 3 seconds and only punches that were correctly timed at 3 seconds were used to gather results. In the case that the duration of the punch was not 3 seconds, the movement was repeated. Using this information, we can calculate that the expected speed of the punch is 0.33 m/s.

Figure 4

1m ruler that was used as a guide when testing the punch speed.



To test the final algorithm that was implemented, (the flexion and extension angles algorithm), a test suite was created to test the algorithm at different angles. For this test, the participant was asked to perform a predetermined stationary pose and the results of the algorithm were captured and compared to the expected results. The test suite that was used can be seen in figure 5 below.

Figure 5

Description of Accuracy Tests for the Flexion and Extension Angle Analysis.

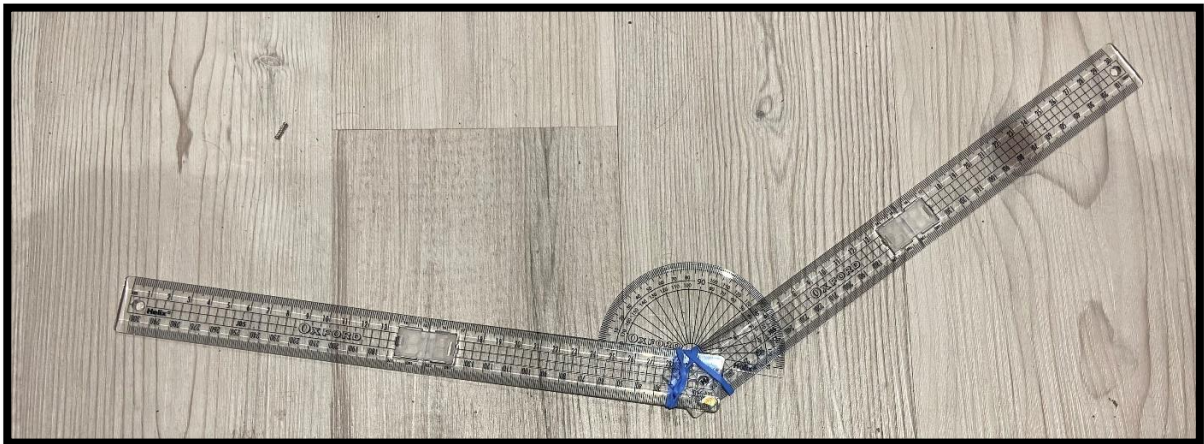
<i>Test Number</i>	<i>Target Joint</i>	<i>Expected Result</i>
1	Right Knee	180
2	Right Knee	135

3	Right Knee	90
4	Left Knee	180
5	Left Knee	135
6	Left Knee	90
7	Right Elbow	180
8	Right Elbow	135
9	Right Elbow	90
10	Left Elbow	180
11	Left Elbow	135
12	Left Elbow	90

To ensure fairness in the tests, an apparatus was set up to guide and measure the angle of the participant's joints. The apparatus was placed against the participant to confirm that the actual angle matched the expected outcome. This can be seen in figure 6 below.

Figure 6:

Apparatus used to measure the angle of a participant's joints.



General Information regarding the tests

All of the tests described in this section were completed on 27th February 2024. The tests were conducted in the different environments described; however, each environment was created inside a house located at the postcode CF38 2NY.

Testing the traditional methods

Since this project aims to improve traditional coaching methods, it is vital to compare the developed system against a human coach. To do this, three human coaches were tested. The tests given to the user were similar to the tests that were used to test the Classifier. This allowed for a direct comparison between the Human and the System. Of course, to allow the Human to

take the test, there had to be some interface implemented to allow the Human to interact with the tests. The methods of testing the Humans can be seen in the following section.

The set of tests to be completed by the humans aimed to simulate the tests done by the classifier. The test allowed the coach's ability to identify characteristics within movements. For these tests, the human was shown a series of images that show an athlete during a point in one of the 5 movements that the system is trained to classify. The tests that were given to the Human are the still image versions of the tests that were given to the Classifier; therefore, the test suite seen in Figure 3 can also be used as the test suite in this case. For each image that the Human was shown, they were asked to select the class that they would assign to the user. They were asked to do this as quickly as they could accurately do so. Just like before, the system time was taken as soon as the image was shown to the Human and the system time was taken as soon as they selected a class. The speed of the response could then be measured.

Testing in this way not only allowed data to be gathered for the speed of the human, it also allowed accuracy to be measured as well. After the test was completed, a report was automatically generated which contained the minimum, maximum and mean response times. The percentage of the correct classifications was also calculated and given in the report. This data could then be evaluated against the data received from testing the classifier system.

Examples of the testing application can be seen in Figures 7 & 8 below.

Figure 7

Start page of the human coach testing application.

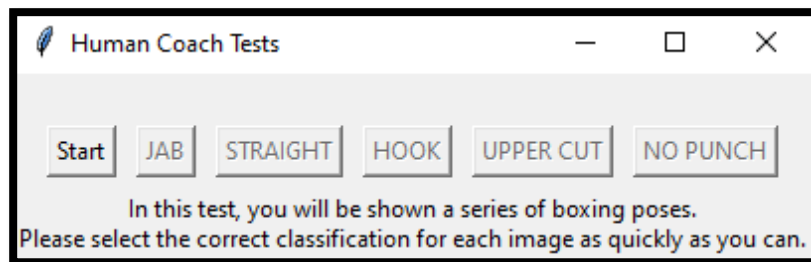
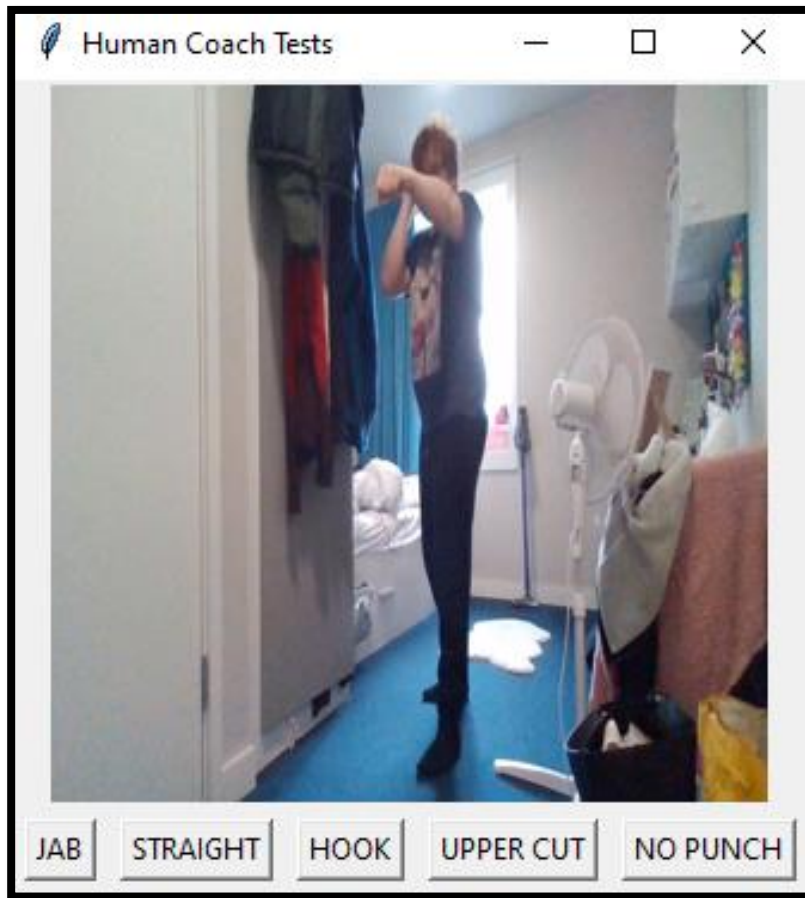


Figure 8

Test page of the human coach testing application.



Gathering qualitative data

Once the human coaches had completed their test, they were then shown the system that had been developed containing the three components. They were asked to interact with the system for a short while and their feedback on the system was recorded. It was important not to have a script or set of questions when asking for their feedback. This is because a unique perspective on the system is what was being looked for in gathering this data and prompting the coaches with questions may result in feedback that is limited to the question that was presented. Instead, more of a discussion-based approach was taken where the coaches were able to steer the conversation and ask questions that they felt were necessary to answer to determine the potential of the technology. A transcript of the conversation was kept and will be evaluated later in this document.

Results

Testing the speed of the system components

As mentioned previously, there were two main characteristics of the project that were tested. The first characteristic was the speed of the different systems that have been developed. To fully evaluate the performance of the systems, tests were performed in a variety of environments. The results have been broken down into three components, the Mediapipe detection system,

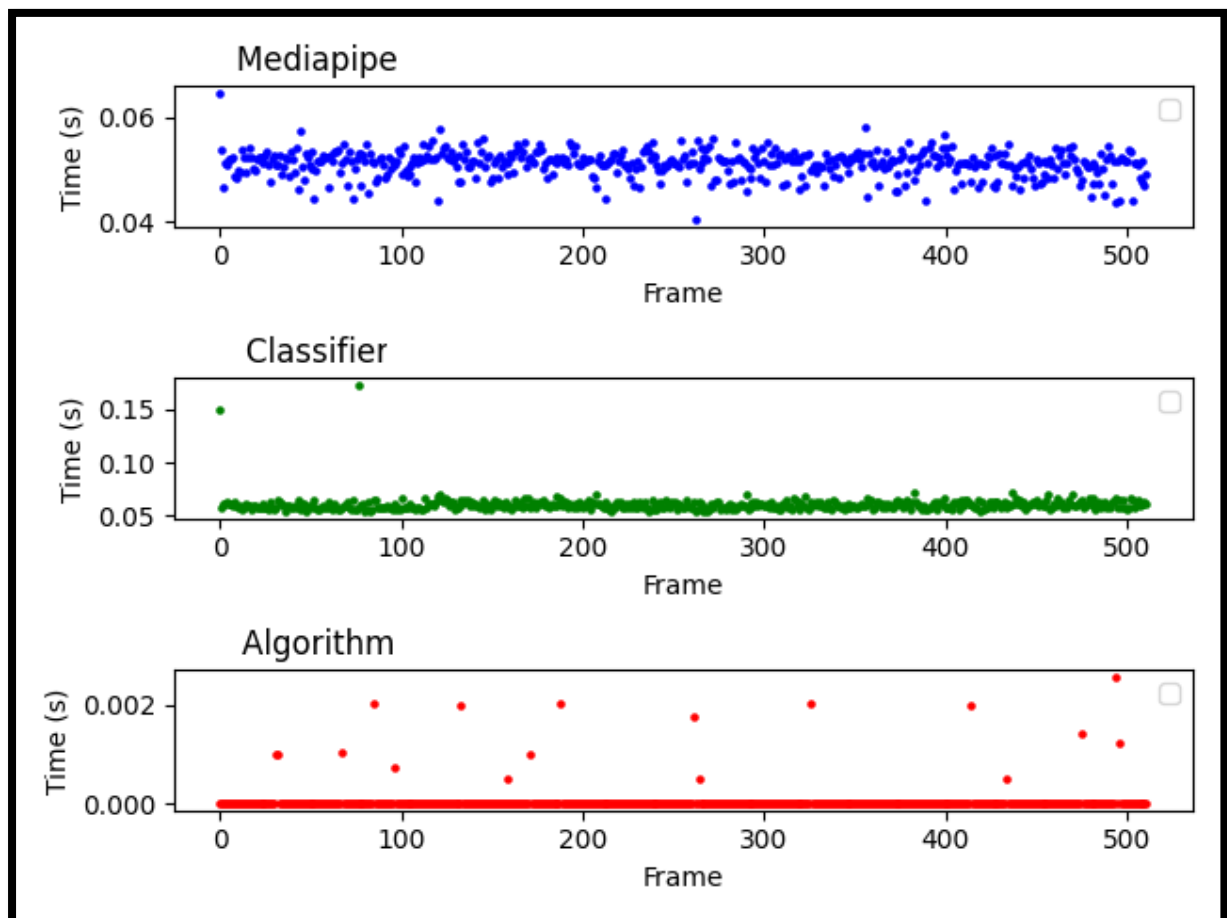
the classifier and the algorithmic analysis. For each system, the amount of time taken for the system to complete its task(s) has been measured for each frame. This has been plotted and the mean, min and max times have been calculated.

Testing in a light environment

The initial tests were conducted in a well-lit environment, similar to typical training spaces or competition arenas. The results, as depicted in the figures below, showcased the processing speed of each component measured over more than 500 frames.

Figure 9

Results of the speed test in a well-lit environment.



Mediapipe Detection System Results

The Mediapipe detection system demonstrated remarkable speed in detecting and tracking the athlete's pose. The graph illustrates the processing times for each frame, with minimal variation observed across the test duration. For example, the mean processing time was calculated at 0.0512 seconds. With a narrow range of only 0.0244 seconds, it is clear that the speed of the model is very consistent. This indicates efficient performance of the detection system in a well-lit environment.

Classification Results

The classifier exhibited rapid processing times at identifying punch types. The graph in Figure 9 depicts the speed test results for the classification system. The mean processing time for the classifier was calculated at 0.0605 seconds, with the range of the minimum and maximum

processing times calculated at 0.1194 seconds. This suggests that the classifier maintained stable performance throughout the test and provided classifications very quickly.

Algorithmic Analysis Results

The algorithmic analysis component showcased exceptional speed and accuracy in analysing detected landmarks. The graph seen in Figure 9 illustrates the processing times for algorithmic analysis, demonstrating consistent performance across different frames. Notably, the minimum processing time achieved was near-instantaneous, indicating the algorithm's efficiency in providing an analysis of the athlete's form. Additionally, the maximum processing time measured was 0.0026 seconds, highlighting the algorithm's ability to handle more complex frames swiftly.

The system as a whole

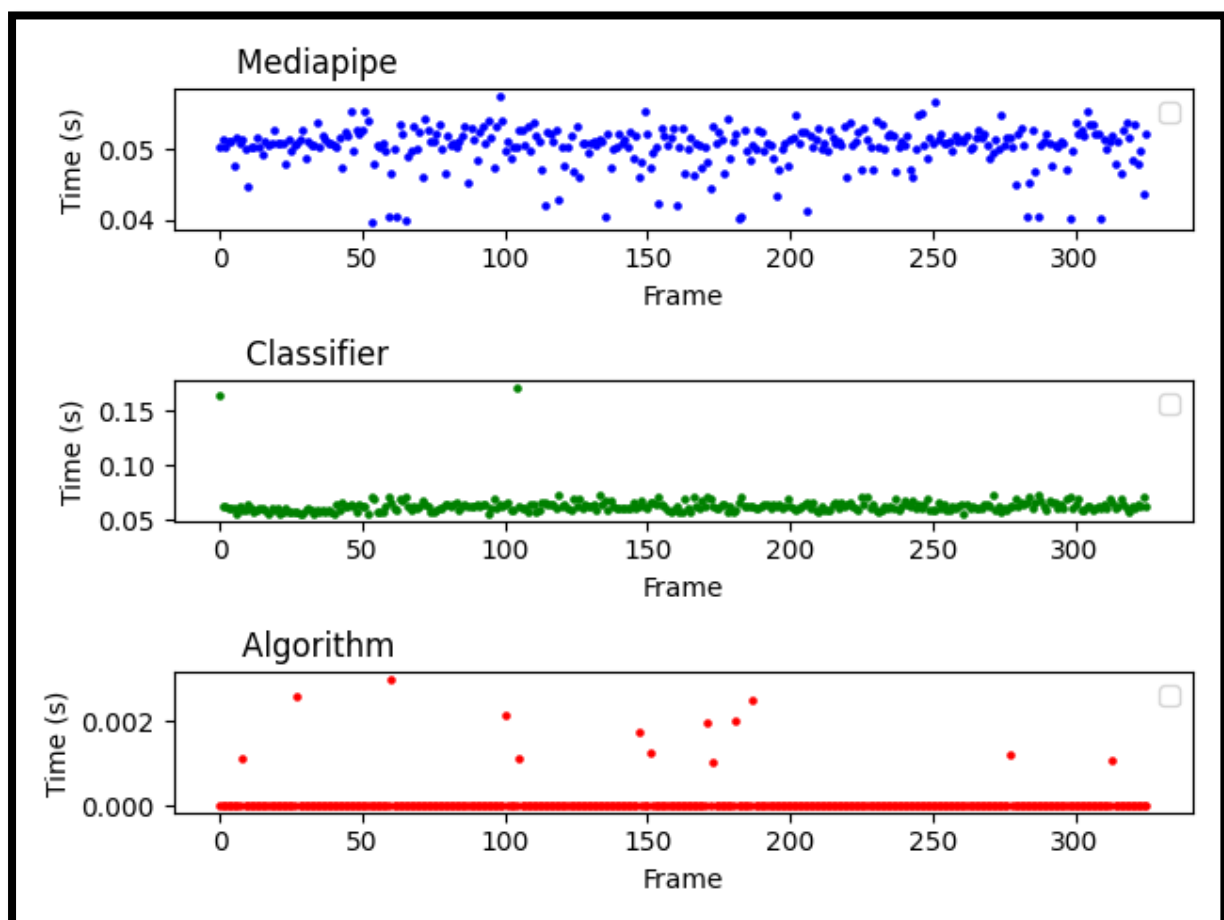
The average processing time was calculated for the three components combined: a value of 0.1143 seconds was obtained. This is greater than the defined threshold.

Testing in a dark environment

Even in challenging low-light conditions, the system maintained its efficiency, as shown in Figure 10 below. The Mediapipe detection system, classifier, and algorithmic analysis continued to perform reliably, with processing times remaining consistent despite the adverse lighting conditions.

Figure 10

Results of the speed test in a poorly lit environment.



The performance of the system in a dark environment remained consistent with that in a well-lit environment. For example, the Mediapipe detection system achieved minimum processing times of 0.0395 seconds and maximum processing times of 0.0576 seconds, demonstrating comparable efficiency under different lighting conditions. Similarly, the classifier and algorithmic analysis components exhibited stable processing times (with ranges of 0.1194 seconds and 0.0026 seconds respectively) indicating resilience to changes in lighting conditions.

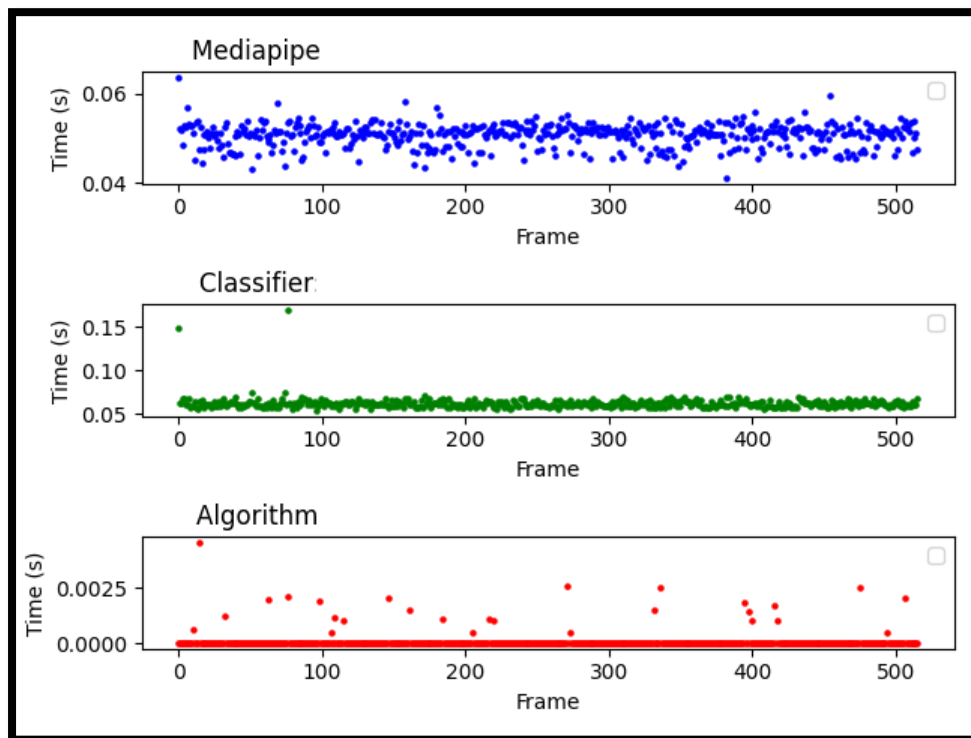
The processing time for the three components combined was also very similar to that of the well-lit test results and also did not meet the defined threshold.

Testing at short range

As part of the speed tests, the three systems were tested with the participant positioned 1 meter away from the webcam. Figure 11 presents the results of these tests that lasted for over 500 frames.

Figure 11

System performance at Short Distance



Mediapipe Detection System Results

The Mediapipe detection system demonstrated impressive efficiency in capturing and monitoring the athlete's pose. The data plotted in the graph seen in Figure 11 reveals consistent processing times for each frame, indicating a reliable performance throughout the test. Notably, the average processing time stood at 0.0507 seconds, with a narrow range of only 0.0228 seconds. These results underscore the system's promptness at analysing poses at close range.

Classification Results

The classifier exhibited rapid processing times in recognizing various punch types. As depicted in Figure 11, the speed test outcomes for the classification system showcased quick

classification processes. The mean processing time was calculated at 0.0618 seconds, with the range between the minimum and maximum processing times at 0.1168 seconds. This suggests consistent and efficient performance throughout the test.

Algorithmic Analysis Results

The algorithmic analysis component displayed remarkable speed and precision in examining detected landmarks. Figure 11 visually represents the processing times for algorithmic analysis, demonstrating rapid performance across different frames. Notably, the minimum processing time achieved was nearly instantaneous, indicating the algorithm's speed at analysing the athlete's form. Furthermore, the maximum processing time recorded was 0.0045 seconds, highlighting the algorithm's capability to handle more complex frames expeditiously.

The system as a whole

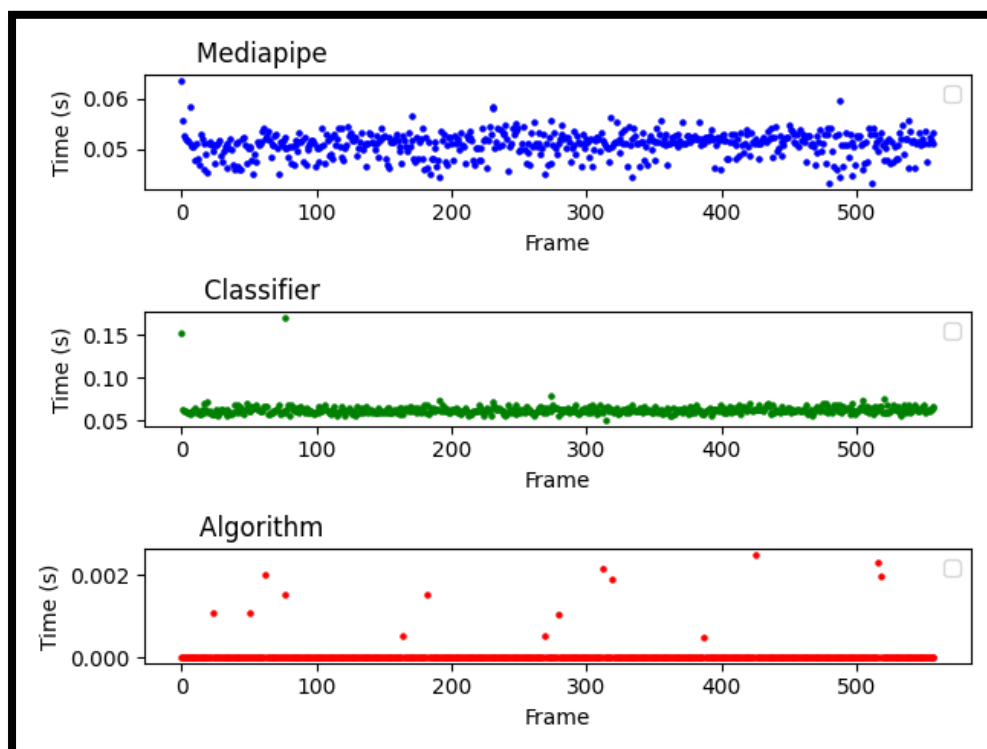
The average processing time was calculated for the three components combined: a value of 0.117 seconds was obtained. This is much greater than the defined threshold.

Testing at long range

In an environment where the participant is at long range, the system maintained its efficiency, as shown in Figure 12 below. All 3 components of the system continued to perform with a consistent processing speed.

Figure 12

Results of the speed test at long range.



The performance of the system at long range remained consistent with the results of the short-range test. For example, the Mediapipe detection system achieved minimum processing times of 0.0430 seconds and maximum processing times of 0.0638 seconds, demonstrating comparable efficiency at different ranges. Similarly, the classifier and algorithmic analysis components exhibited settled processing times (with ranges of 0.1209 seconds and 0.0025

seconds respectively) indicating resilience to changes in the range of the distance of the user to the webcam.

The processing time for the three components combined was also very similar to that of the short-range test results and also did not meet the defined threshold.

Analysis of Trends

While each of the test that were completed revealed information about the individual environment, there are also some trends within the data that need to be evaluated.

Mediapipe Detection System Results

The first trend that was identified is within the data generated from the Mediapipe detection tests. Being a machine learning model, one would expect the model to require more processing time in environments where landmarks cannot be as easily identified, however, it is clear from the data that this is not the case. The speed of processing remains very consistent across all the tested environments. The results of the speed tests, combined with the immense accuracy of the model that was discovered in the literature review, creates a solid argument for the use of the Mediapipe framework within the real-time Boxing application presented in this document.

Classification Results

The second trend that was identified is within the data generated from the Classifier tests. What can be noticed in the graphs for each of the tests is an anomalous result at the beginning of the test and another at around 100 frames into the test. This is a completely unexpected result and is extremely difficult to identify the cause of this due to the repetition of this anonymous result over all the tests. The possible causes of this can be narrowed down to either an issue with the webcam that's causing the frames to be distorted, an issue with the Mediaipe detection system producing distorted landmarks, or an issue with the landmark classifier that's causing the large processing time for those two frames. It may also be the case that there are multiple causes, or it may be the case that there is another possible cause that hasn't been identified however it is likely that the cause is one of the three possibilities identified above.

Algorithmic Analysis Results

The final trend that was identified across all the tests was in the data generated by the algorithmic analysis tests. What can be seen across these tests is most data points tending towards zero, with only a small portion of the data points being calculated at more than a millisecond. The reason for this is due to the structure of the algorithms. While every frame receives some level of analysis, most of the frames will only go through at most a few IF statements which leads to the large number of instantaneous results. While these results still have some importance, the results from the frames that received a high level of analysis have a much greater significance for this research. For this reason, the most significant data point for the algorithmic analysis would be the maximum value achieved during these tests. Looking at these values, the algorithmic analysis still processes frames rapidly and would be a viable option for a real-time system with regards to the speed of the system.

Concluding Remarks

In conclusion, the comprehensive testing of the system components, including the Mediapipe detection system, classifier, and algorithmic analysis, has returned valuable insights into their performance across various environments. The results clearly demonstrate the efficiency of the system in real-time monitoring of athlete movements, particularly in the context of boxing;

however, none of the systems managed to achieve a processing time that fell within the defined threshold.

The consistent speed exhibited by the Mediapipe detection system, irrespective of lighting conditions or distance from the webcam, underscores its suitability for pose detection in

dynamic scenarios. This aligns seamlessly with the primary objective of the project, which aimed to develop a real-time boxing application capable of analysing and providing feedback on athlete performance.

While anomalies observed in the classifier tests warrant further investigation, the overall rapid processing times and stable performance highlight its potential in swiftly identifying punch types. Addressing the anomalies could further enhance the classifier's robustness, ensuring consistent and accurate performance in all scenarios.

Furthermore, the algorithmic analysis component has proven its ability to rapidly analyse detected landmarks, with all frames being processed within a few milliseconds. This aligns with the thesis's emphasis on leveraging algorithmic efficiency to provide timely feedback to athletes, aiding in form correction and performance optimization during training and competitions.

The identified trends within the data not only validate the efficacy of the developed system but also offer insights for future enhancements and research directions. By leveraging the strengths of the Mediapipe framework and refining the classifier's performance, the real-time boxing application can continue to evolve as a valuable tool for athletes and coaches alike.

In essence, the results of these tests confirm the speed of the system is feasible for real-world Boxing applications.

Testing the accuracy of the system components

Review of the BlazePose model output

As mentioned previously, the accuracy of the BlazePose model has already been evaluated in the literature review. However, to ease the visualisation of the models output, the captured frames have been overlayed with the landmark data that was extracted by the model. Examples of the output can be seen in figures 13, 14, 15, 16 & 17 below.

Figure 13

Example of BlazePose model output.

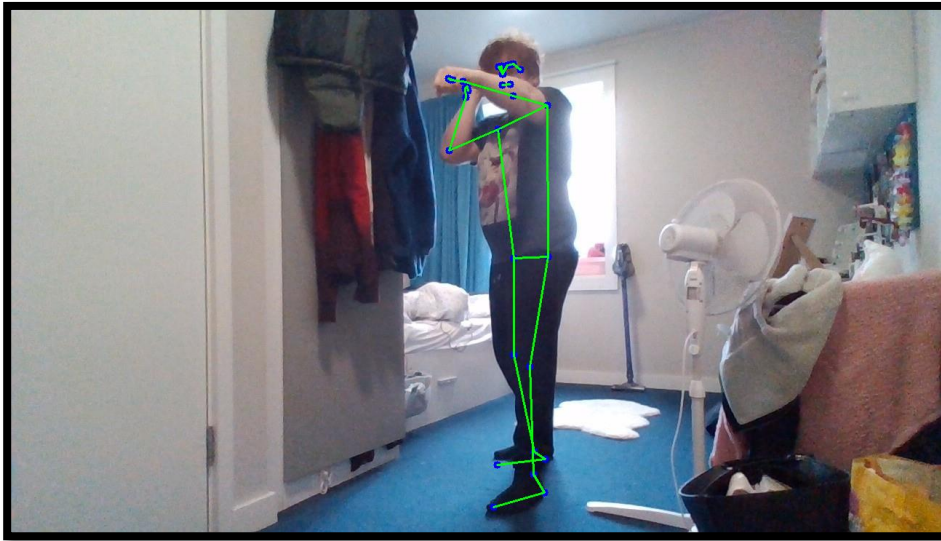


Figure 14

Example of BlazePose model output.

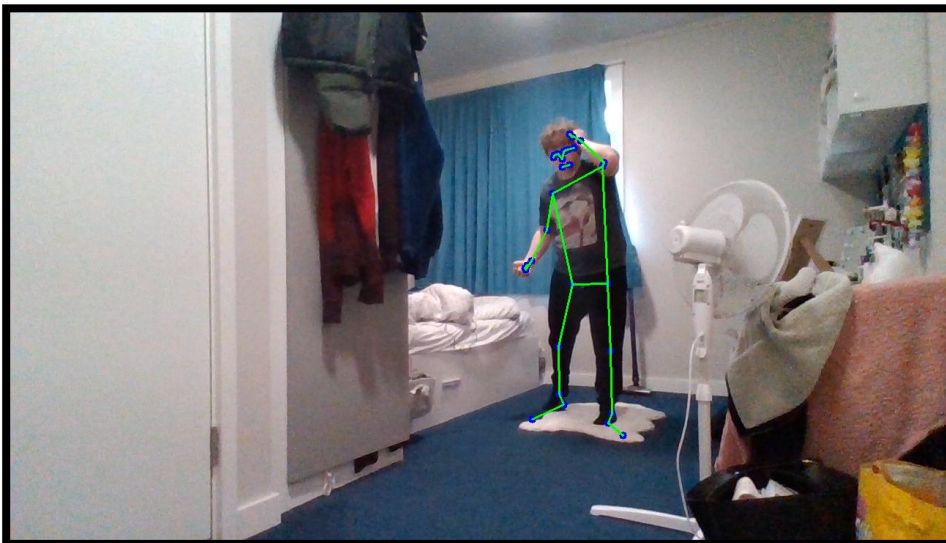


Figure 15

Example of Blazepose model output.

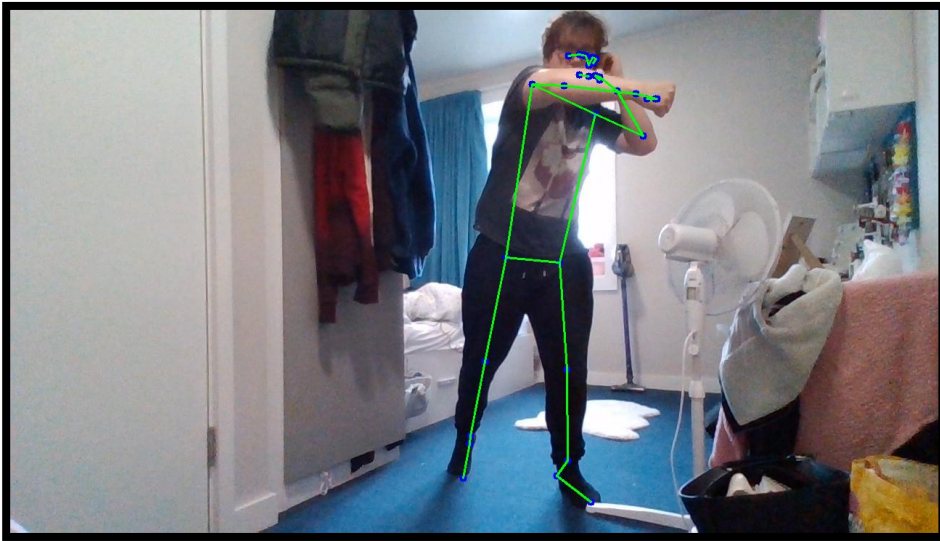


Figure 16

Example of Blazepose model output.

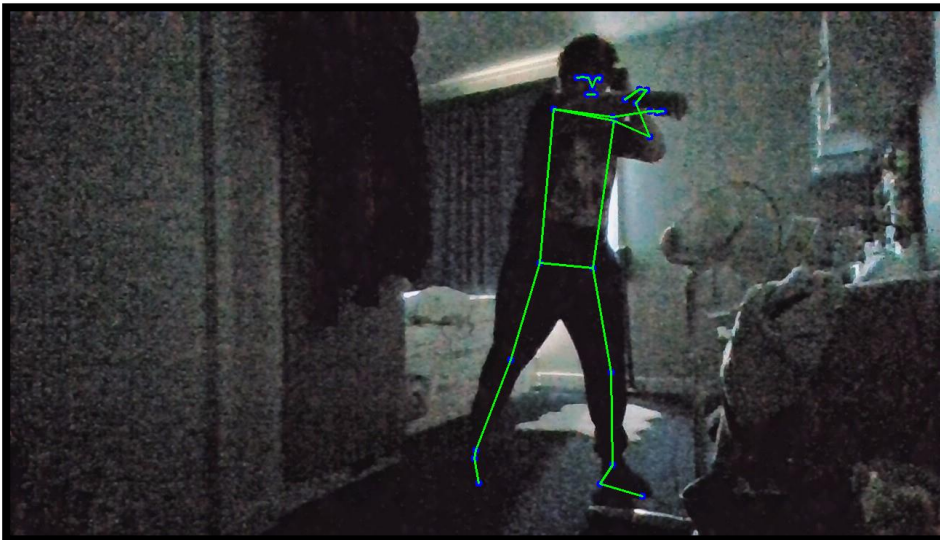
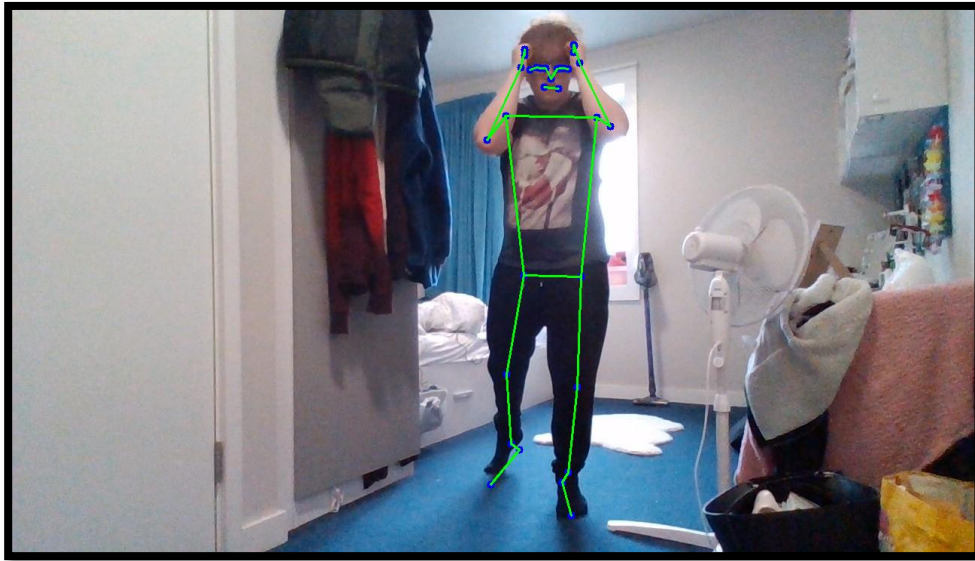


Figure 17

Example of Blazepose model output.



What can be seen from the figures above is the great accuracy of the model. What can also be noticed is that there are some landmarks that the model has got wrong, especially where landmarks overlap; however, for the most part the model is highly accurate.

Testing the classifier

Testing during the training process

Once the model had been trained successfully, the remaining 20% of the data that was kept back for testing was then given to the model. The model received an accuracy rating of 92.35% when tested using this data. This is an excellent result and paired with the rapid processing speeds demonstrated by the classifier, this result provides a solid argument for the use of the machine learning approach within a real-time Boxing application.

Testing using the test suite

The tests outlined by the test suite in Figure 3 yielded some interesting results. The results of these tests can be seen in Figure 18 below.

Figure 18

Results of the Classification accuracy test.

Test Number	Environment	Expected Class	Result	Test Number	Environment	Expected Class	Result
1	Well-lit	Jab	Jab	21	Close Range	Jab	Jab
2	Well-lit	Jab	Jab	22	Close Range	Jab	Jab
3	Well-lit	Straight	Straight	23	Close Range	Straight	Straight
4	Well-lit	Straight	Straight	24	Close Range	Straight	Straight
5	Well-lit	Hook	Hook	25	Close Range	Hook	Hook
6	Well-lit	Hook	Hook	26	Close Range	Hook	Hook
7	Well-lit	Upper Cut	Hook	27	Close Range	Upper Cut	Upper Cut

8	Well-lit	Upper Cut	Upper Cut	28	Close Range	Upper Cut	Upper Cut
9	Well-lit	No Punch	No Punch	29	Close Range	No Punch	No Punch
10	Well-lit	No Punch	No Punch	30	Close Range	No Punch	No Punch
11	Dark	Jab	Jab	31	Long Range	Jab	Jab
12	Dark	Jab	Jab	32	Long Range	Jab	Jab
13	Dark	Straight	Straight	33	Long Range	Straight	Jab
14	Dark	Straight	Straight	34	Long Range	Straight	Straight
15	Dark	Hook	Jab	35	Long Range	Hook	Hook
16	Dark	Hook	No Punch	36	Long Range	Hook	Hook
17	Dark	Upper Cut	No Punch	37	Long Range	Upper Cut	No Punch
18	Dark	Upper Cut	Upper Cut	38	Long Range	Upper Cut	Upper Cut
19	Dark	No Punch	No Punch	39	Long Range	No Punch	No Punch
20	Dark	No Punch	Jab	40	Long Range	No Punch	Jab

From these results, an accuracy of 80% can be calculated for the classifier. This result is significantly less than the 92.35% that was achieved during the training of the model. To investigate this, the accuracy of the classifier over each of the environments can be calculated further.

Looking at the well-lit environment, an accuracy of 90% was achieved. This result is a lot closer to the 92.35% that one would expect. Since the model only failed to correctly classify a single movement, it may be the case that more test cases within this environment would result in a greater accuracy. Overall, this result is very promising and shows great potential for use within a real-time Boxing application.

Looking at the dark environment, an accuracy of 60% was achieved. This result is very poor and is nowhere near the 92.35% that was achieved during training. Once again, the accuracy achieved may increase with more test cases, however, a highly accurate model would be expected to perform better even with a small amount of test cases. While one would expect the accuracy of the model to be reduced in a dark environment, a decrease of around 30-35% is larger than expected and is something that would need to be improved to be feasible for use within a real-time Boxing application.

Looking at the close-range environment, an accuracy of 100% was achieved. Obviously, this is an amazing result; however, it cannot be said that the model is 100% accurate in this environment because this result was achieved over only 10 test cases. For a test suite containing many more test cases, the accuracy result may be less. Nevertheless, the accuracy of the model can still be said to be strong and is probably close to the 92.35% mark that was achieved during training.

The final environment to look at is the long-range environment. Within this environment, an accuracy of 70% was achieved. This is not a bad result, but it is not a result that would be classed as feasible for a real-time Boxing application. Once again, with more test cases the

accuracy may improve but even with a small amount of test cases, a good model would still perform well.

Testing the algorithmic analysis

Figure 19

Results of the accuracy test for the algorithmic analysis.

Test Number	<i>Expected Feedback</i>	<i>Actual Feedback</i>
1	“The punch was thrown at a speed of 0.33 m/s” “The punch was thrown in a straight trajectory”	“The punch was thrown at a speed of 0.36 m/s” “The punch was thrown in a straight trajectory”
2	“The punch was thrown at a speed of 0.33 m/s” “The punch was not thrown in a straight trajectory”	“The punch was thrown at a speed of 0.28 m/s” “The punch was not thrown in a straight trajectory”
3	“The punch was thrown at a speed of 0.33 m/s” “The punch was thrown In a straight trajectory”	“The punch was thrown at a speed of 0.3 m/s” “The punch was thrown In a straight trajectory”
4	“The punch was thrown at a speed of 0.33 m/s” “The punch was not thrown in a straight trajectory”	“The punch was thrown at a speed of 0.36 m/s” “The punch was not thrown in a straight trajectory”
5	“Back heel is raised”	“Back heel is raised”
6	“Please raise your back heel!”	“Please raise your back heel!”
7	“The punch was thrown at a speed of 0.33 m/s” “The punch was thrown in a straight trajectory”	“The punch was thrown at a speed of 0.39 m/s” “The punch was thrown in a straight trajectory”
8	“The punch was thrown at a speed of 0.33 m/s” “The punch was not thrown in a straight trajectory”	“The punch was thrown at a speed of 0.33 m/s” “The punch was thrown in a straight trajectory”
9	“The punch was thrown at a speed of 0.33 m/s” “The punch was thrown In a straight trajectory”	“The punch was thrown at a speed of 0.31 m/s” “The punch was thrown in a straight trajectory”
10	“The punch was thrown at a speed of 0.33 m/s” “The punch was not thrown in a straight trajectory”	“The punch was thrown at a speed of 0.39 m/s” “The punch was not thrown in a straight trajectory”
11	“Back heel is raised”	“Please raise your back heel!”
12	“Please raise your back heel!”	“Please raise your back heel!”
13	“The punch was thrown at a speed of 0.33 m/s” “The punch was thrown in a straight trajectory”	“The punch was thrown at a speed of 0.29 m/s” “The punch was thrown in a straight trajectory”
14	“The punch was thrown at a speed of 0.33 m/s” “The punch was not thrown in a straight trajectory”	“The punch was thrown at a speed of 1.02 m/s” “The punch was not thrown in a straight trajectory”

15	“The punch was thrown at a speed of 0.33 m/s” “The punch was thrown In a straight trajectory”	“The punch was thrown at a speed of 0.34 m/s” “The punch was thrown In a straight trajectory”
16	“The punch was thrown at a speed of 0.33 m/s” “The punch was not thrown in a straight trajectory”	“The punch was thrown at a speed of 0.34 m/s” “The punch was thrown In a straight trajectory”
17	“Back heel is raised”	“Back heel is raised”
18	“Please raise your back heel!”	“Please raise your back heel!”
19	“The punch was thrown at a speed of 0.33 m/s” “The punch was thrown in a straight trajectory”	“Please raise your back heel!”
20	“The punch was thrown at a speed of 0.33 m/s” “The punch was not thrown in a straight trajectory”	“The punch was thrown at a speed of 0.31 m/s” “The punch was not thrown in a straight trajectory”
21	“The punch was thrown at a speed of 0.33 m/s” “The punch was thrown In a straight trajectory”	“The punch was thrown at a speed of 0.37 m/s” “The punch was not thrown in a straight trajectory”
22	“The punch was thrown at a speed of 0.33 m/s” “The punch was not thrown in a straight trajectory”	“The punch was thrown at a speed of 0.29 m/s” “The punch was thrown In a straight trajectory”
23	“Back heel is raised”	“Please raise your back heel!”
24	“Please raise your back heel!”	“Please raise your back heel!”

There are two main aspects of the results that need to be looked at. There is the speed aspect and then there is the correctness aspect.

Correctness Aspect

With regards to the correctness aspect of the results. The overall accuracy achieved can be calculated at 66.66%. This is not a great result. Once again, evaluating the environments individually will allow for a better understanding of the results.

Looking at the well-lit environment, an accuracy score of 100% has been achieved. This is expected as one would assume that the landmarks being produced by Mediapipe are very accurate under these conditions based on the research that was completed on the Blazepose model.

Looking at the dark environment, an accuracy of 66.66% was achieved. As mentioned previously, one would expect the accuracy of the system to decrease under dark conditions but maybe not this low. This is a very poor result.

Looking at the close-range environment, an accuracy of 83.33% was achieved. This is a good result and is one that would be expected under these conditions.

Finally, the long-range environment achieved an accuracy of 33.33% which again is expected under these conditions.

Breaking down the results into the individual environments revealed some fascinating trends. Since the algorithms are fixed, the only factor affecting the result is the landmarks that the BlazePose model produced. In environments that ease the difficulty of vision, the model produces more accurate results. In environments where computer vision is hard, the model produces far less accurate results which as a result is causing a drastic loss of accuracy throughout the system. This is a significant point of concern.

Speed Aspect

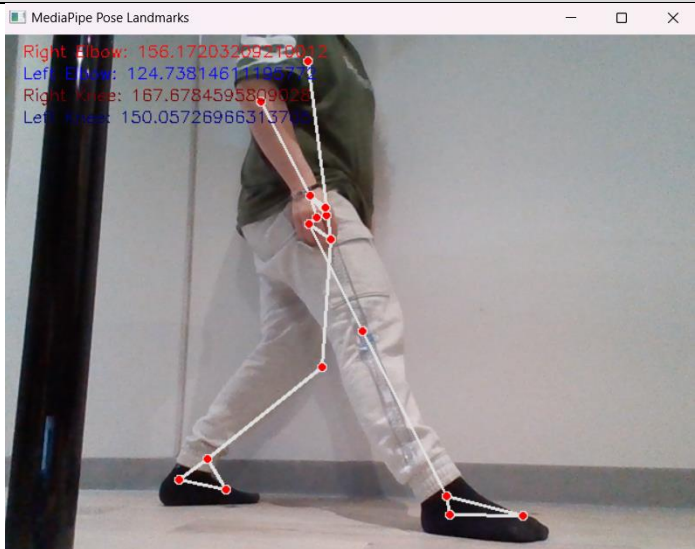
Looking at the speed aspect of the result, to determine the accuracy, for each reading, the difference between the actual value and the expected value was calculated as a percentage of the expected value. The mean value of these was then calculated. The mean percentage difference achieved by the system was 24.42%. This is a quite a large difference to achieve. What can be notice in the results is one of the results is extremely large compared to the rest. While taking this result into consideration is something that is open to debate, the calculated mean percentage difference if this result is ignored comes out at 10.21%. Both results are very large percentage errors to have within the system and shows that there is still a lot of work that needs to be done to improve the technology.



Flexion and Extension Angle Aspect



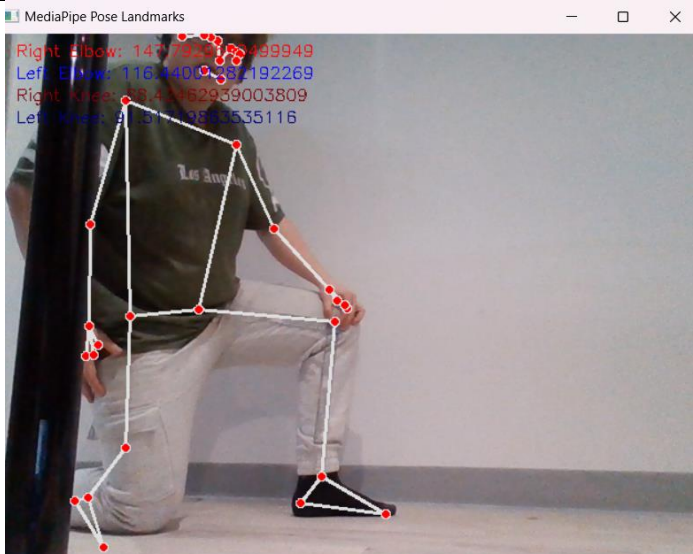
The results from the flexion and extension angle tests revealed some interesting information about the accuracy of the BlazePose model. For the elbow joints, a mean error of 27.495% was calculated (a mean error of 35.34% for the right elbow and a mean error of 19.59% for the left elbow). For the knee joints, a mean error of 8.215% was calculated (a mean error of 7.22% for the right knee and a mean error of 9.21% for the left knee). This large difference in error was completely unexpected and while the results for the knee joints demonstrates a level of accuracy suitable for a real-time application like the one proposed, the results for the elbow joints highlights that this technology has a lot of room for improvement. The full test results can be seen in figure 20 below.

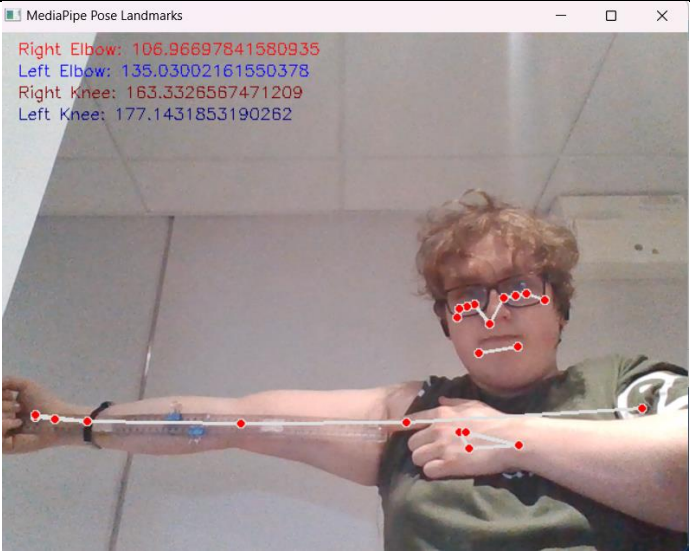
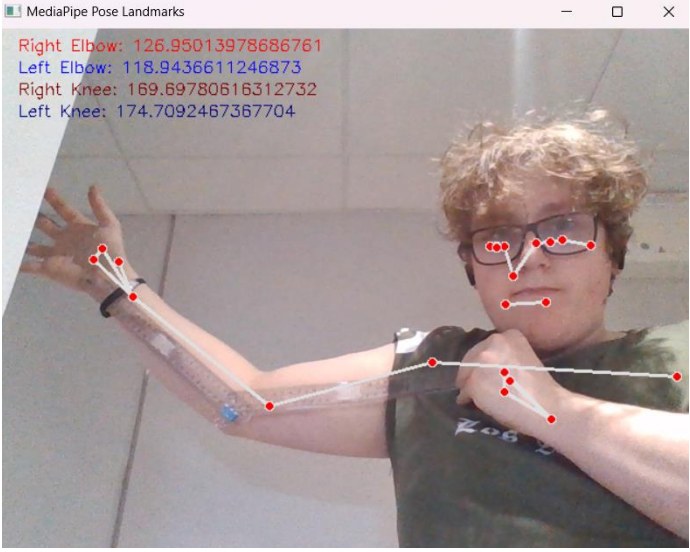
Figure 20

Results of the flexion and extension angle tests.


Test Number	Expected Result	Actual Result
1	180	 <p>MediaPipe Pose Landmarks</p> <p>Right Elbow: 156.17203208212012 Left Elbow: 124.73814611195733 Right Knee: 167.6784595809023 Left Knee: 150.05726966313705</p>

		<p>Error = 6.84%</p>
2	135	<div> <div>MediaPipe Pose Landmarks</div> <div> <p>Right Elbow: 153.92339028930878</p> <p>Left Elbow: 108.40452275871307</p> <p>Right Knee: 153.51687321004992</p> <p>Left Knee: 168.11508029175318</p> </div>  </div> <p>Error = 13.72%</p>
3	90	<div> <div>MediaPipe Pose Landmarks</div> <div> <p>Right Elbow: 115.29377190889595</p> <p>Left Elbow: 149.00801243148663</p> <p>Right Knee: 90.99865778663799</p> <p>Left Knee: 65.90862429803717</p> </div>  </div> <p>Error = 1.11%</p>

4	180	 <p>Right Elbow: 127.24710919591902 Left Elbow: 175.26499124374953 Right Knee: 168.71114064143822 Left Knee: 170.03361969370715</p> <p>Error = 5.54%</p>
5	135	 <p>Right Elbow: 91.70052086745667 Left Elbow: 133.76009138596777 Right Knee: 129.0280353061205 Left Knee: 107.46712592076125</p> <p>Error = 20.39%</p>
6	90	 <p>Right Elbow: 147.793275499949 Left Elbow: 116.44094382192269 Right Knee: 107.43462939003809 Left Knee: 91.51743863535116</p>

		<p>Error = 1.69%</p>
7	180	<div> <div>MediaPipe Pose Landmarks</div> <div> <div>Right Elbow: 106.96697841580935</div> <div>Left Elbow: 135.03002161550378</div> <div>Right Knee: 163.3326567471209</div> <div>Left Knee: 177.1431853190262</div> </div>  <p>Error = 40.57%</p> </div>
8	135	<div> <div>MediaPipe Pose Landmarks</div> <div> <div>Right Elbow: 126.95013978686761</div> <div>Left Elbow: 118.9436611246873</div> <div>Right Knee: 169.69780616312732</div> <div>Left Knee: 174.7092467367704</div> </div>  <p>Error = 5.96%</p> </div>

9	90	<div data-bbox="678 190 1380 734"> <p>MediaPipe Pose Landmarks</p> <p>Right Elbow: 143.54570855484948 Left Elbow: 109.70095040231317 Right Knee: 158.2645482959015 Left Knee: 175.34116179604374</p> </div> <p>Error = 59.5%</p>
10	180	<div data-bbox="678 817 1380 1361"> <p>MediaPipe Pose Landmarks</p> <p>Right Elbow: 138.73619377090725 Left Elbow: 144.8244742391682 Right Knee: 174.89260805317394 Left Knee: 174.0947867671199</p> </div> <p>Error = 19.54%</p>
11	135	<div data-bbox="678 1444 1380 1989"> <p>MediaPipe Pose Landmarks</p> <p>Right Elbow: 119.6164141055506 Left Elbow: 132.75524734971896 Right Knee: 165.12548341210933 Left Knee: 174.82028208746826</p> </div>

		Error = 1.67%
12	90	

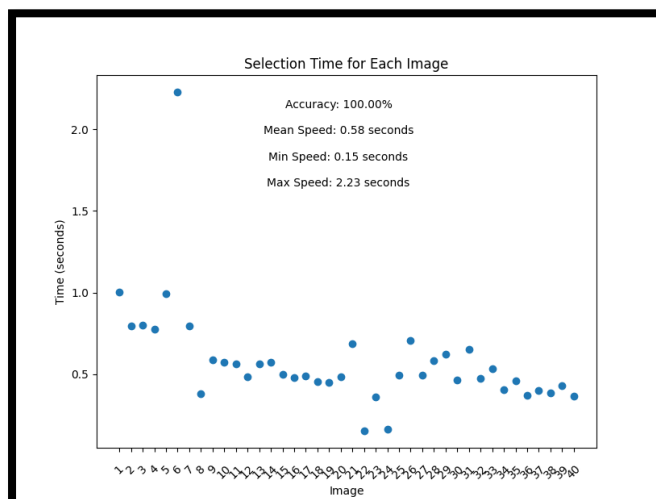
Testing human coaches

As mentioned previously, the tests for the human coaches followed the structure of test suites seen in Figure 3 and Figure 4. The first test was aimed at testing the human coach's ability to classify punches. The second test was aimed at testing the human coach's ability to identify characteristics of correctness within movements. In this section, the results of the human coach tests will be evaluated and compared to the results of the system tests.

Human Coach 1

Figure 21

Results of the classification test for human coach 1.

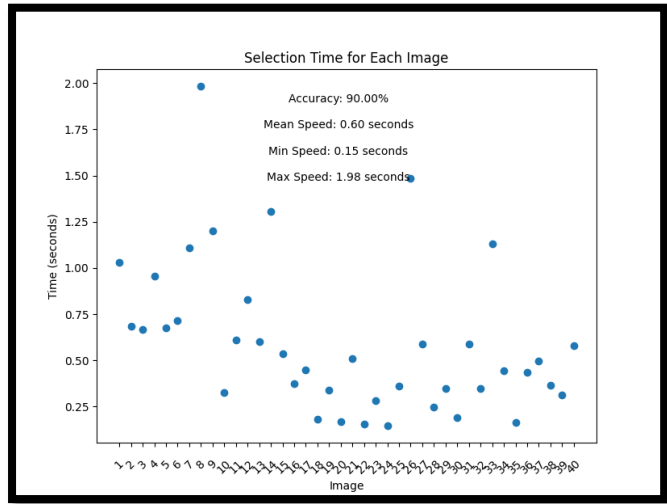


As seen in Figure 21, the results of the classification test revealed some interesting results. The first thing worth noting is the accuracy of 100% that was achieved. This is expected from a human coach and solidifies the need for the proposed system to achieve immense accuracy levels to keep up with human coaches. While the speed of the classifications has some inconsistencies, what can be noticed is a very slight downward trend. This shows some level of

learning from the coach as the test went on. Achieving a mean classification time of 0.58 seconds, this human coach is significantly slower than the system at classifying the movements. This result was expected.

Human Coach 2

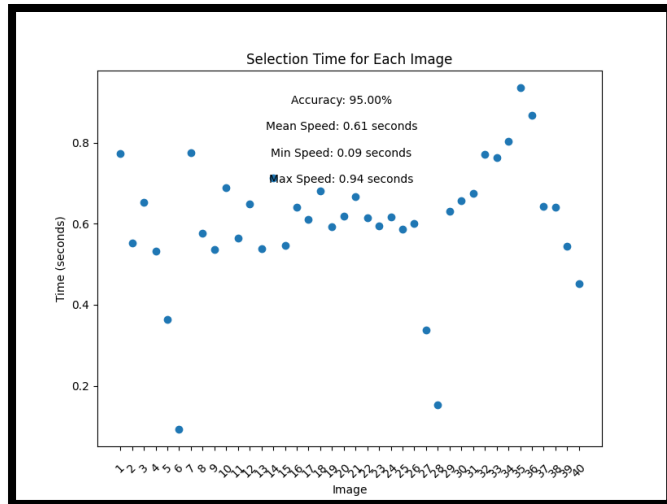
Figure 22
Results of the classification test for human coach 2.



The results seen in figure 22 are follow similar patterns to the classification results of human coach 1. While the accuracy score of 90% is not as high as the previous coach, the result still provides a strong argument for the need of the proposed system to be performing with near perfect accuracy. The speed results for this coach are far more inconsistent than the previous coach; however, the results still show the downwards trend that the results of the previous coach showed with the classification speeds being far quicker at the end of the test than at the beginning. Once again, as expected, the classification speed of this coach is much slower than the developed system.

Human Coach 3

Figure 23
Results of the classification test for human coach 3.



The results seen in figure 23 show some interesting patterns. The accuracy rating of 95% is consistent with the other two coaches. The result further strengthens the vitalness of the proposed system to perform with accuracies as close to perfect as possible to compete with traditional human coaching. Although the mean speed of this coach is very consistent with the other two coaches, the graph in Figure 23 shows a pattern that has not been seen before. There are two main trends that can be identified. Firstly, there are a couple of points where rapid processing speeds can be seen. This is interesting to see and may have some correlation with the 5% accuracy loss. The second trend that can be identified which is very interesting is that the coach becomes slower as the test goes on. This is different from the other two coaches whose results show some form of learning as the test goes on that allows for quicker classifications which one would expect during a test like this. Nevertheless, the mean speed of the human coach still does not come close to the mean processing times of the developed system.

Final Remarks

What can be taken from the human coach tests is that the visual accuracy of a human coach is extremely hard to beat with an average score of 95% across all coaches. Achieving a score of 94.51% during training and an average score of 80% during a full test suit, the developed system does not meet the accuracy standards that would improve traditional coaching. While this does not directly affect the Mediapipe framework, it does show that there is a lot of work needed within the computer vision and machine learning field to bring the accuracy of the technology up to par.

As for the speeds displayed by the coaches, the average speed was 0.596 seconds. Comparing this to the developed system that achieved average speeds of around 0.11 seconds, it is fair to say that the coaches certainly lose to the technology in this realm. Processing speed is a major aspect of any real-time feedback system so improvements of around 5 times compared to traditional coaching is a significant improvement.

Feedback from human coaches about the technology

The discussion with the coaches yielded some very useful feedback. The feedback that was taken out the conversation can be seen in Figure 24 below. A full copy of the transcript can be found in the Appendix of this document.

Figure 24

Feedback taken from the discussion with the coaches.

<i>Pros</i>	<i>Cons</i>
Real-time feedback: The system offers real-time feedback on the boxer's form, allowing for immediate adjustments and corrections. This can significantly enhance training effectiveness and accelerate skill development.	Risk of overreliance: There is a concern among the coaches about the risk of boxers becoming over-reliant on the technology. Over-reliance could potentially lead to a dependence on the system rather than developing essential instincts and intuition crucial in boxing.
Technological advancement: Introducing pose detection technology into boxing training demonstrates a commitment to leveraging innovation for the benefit of athletes. It	Accuracy issues: The coaches acknowledge the inherent limitations of pose detection systems, such as accuracy issues. False readings or inaccuracies in the data could result in incorrect

showcases an openness to exploring new methods to improve performance.	feedback and potentially harm the boxer's progress if not addressed promptly.
Objective analysis: By analysing detected poses, the system provides objective insights into the boxer's technique. This objective analysis can help identify areas for improvement that may not be evident through subjective observation alone.	Accessibility: There is a mention of potential disparities in access to the technology among different gyms or trainers. Not all facilities may have access to such advanced systems, which could create inequalities in training opportunities and development.
Supplemental tool: The coaches recognize the system as a useful supplement to traditional coaching methods. It is seen as a tool that can enhance training without completely replacing the role of coaches or traditional training techniques.	Balancing technology and tradition: While the system offers technological advancements, there is a need to strike a balance between integrating technology into training and preserving the fundamentals of the sport. It's essential not to lose sight of the traditional coaching methods and the instinctual aspects of boxing.

Discussion

The findings presented in this study offer a comprehensive examination of the performance of the developed system across various testing conditions. Through rigorous testing in both well-lit and challenging environments, valuable insights into the system's capabilities and limitations were gained.

The data reveals a promising performance from the system in well-lit environments, where it demonstrated remarkable speed and accuracy in detecting and analysing athlete movements. These results align closely with the theoretical framework, affirming the efficacy of the system's design and implementation in optimal conditions. Such findings reinforce the potential of integrating advanced technology, such as machine learning and computer vision, into sports coaching and training paradigms.

However, the compatibility of the system with the theoretical framework was tested when subjected to challenging environments, such as low-light conditions and long-range distances. Here, unexpected outcomes emerged, revealing a significant discrepancy in accuracy compared to well-lit scenarios. The system also fell short when faced with high-precision tasks, such as identifying flexion and extension angles, where the system achieved high levels of accuracy for some joints but for others the system really struggled with producing accurate results. This discrepancy challenges conventional assumptions and underscores the need for further exploration and refinement in real-world applications.

The unexpected outcomes observed prompt a reconsideration of our approach and expectations regarding sports technology. While the theoretical framework provided a solid foundation for system development, the practical implementation highlighted the complexities and nuances inherent in real-world settings. This shift in perspective calls for a more nuanced

understanding of the interplay between technology, environmental factors, and user context in sports coaching and training.

Transparency in reporting these unexpected outcomes is crucial for establishing credibility and fostering a culture of continuous improvement. By openly acknowledging the limitations and challenges encountered during the research process, we contribute to the collective knowledge base and pave the way for future advancements in the field.

The developed system's portability signals a pivotal moment in sports coaching technology, challenging conventional paradigms and inspiring a shift towards more flexible and accessible approaches. Its ability to operate on commonplace devices like laptops with webcams challenges the reliance on costly and stationary equipment prevalent in current research. This shift towards portability not only democratizes access to advanced coaching tools but also spurs innovation in research and development. Researchers are compelled to explore the potential of portable technologies in sports coaching, driving the creation of more inclusive and adaptable solutions. This trend encourages a departure from the status quo and fosters a mindset that prioritizes versatility and accessibility in the design and implementation of coaching technologies. Consequently, future research is likely to concentrate on refining and broadening the capabilities of portable systems, potentially revolutionizing the landscape of sports training and coaching.

Reflecting on the research process, it's evident that while significant strides have been made in developing the system, there remain areas for exploration and refinement. The journey from theory to application has been enlightening, revealing both the potential and complexities of integrating advanced technology into sports coaching and training.

In conclusion, the discussion chapter serves as a platform for synthesizing the findings, examining their implications, and reflecting on the research journey. By critically evaluating the data, considering their compatibility with the theoretical framework, and embracing unexpected outcomes, we contribute to a deeper understanding of sports technology and pave the way for future innovation and improvement.

Conclusion

In this dissertation, the researcher embarked on a comprehensive exploration of the performance and implications of a real-time sports coaching system integrating machine learning and computer vision technologies. Through extensive testing and analysis, valuable insights were gained into the system's capabilities, limitations, and potential implications for sports coaching and training.

The research revealed promising results in the system's ability to provide real-time feedback on athlete movements, particularly in well-lit environments. The Mediapipe detection system, classifier, and algorithmic analysis components demonstrated remarkable speed and accuracy in certain areas, highlighting the potential of advanced technology in enhancing sports performance analysis. The system's portability breaks the boundaries of all known systems in the industry and marks a significant advancement in current research.

However, challenges encountered in testing under challenging environments, such as low-light conditions and long-range distances, underscore the need for further refinement and exploration. Furthermore, the systems accuracy fell off when faced with more challenging tasks

requiring a high level of precision such as identifying flexion and extension angles. The discrepancies in accuracy observed in these conditions call for a nuanced understanding of the interplay between technology, environmental factors, and user context in real-world applications.

The conclusion of this dissertation reaffirms the significance of ongoing research and development in sports technology. While the findings contribute to a deeper understanding of the system's capabilities and limitations, they also serve as a springboard for future innovation and improvement. As the study concludes, it's essential to acknowledge the inherent limitations in any research endeavour and lay the groundwork for future advancements in the field of sports technology.

Bibliography

- Ahmad, N., Ghazilla, R. A. R., Khairi, N. M., & Kasi, V. (2013). Reviews on Various Inertial Measurement Unit (IMU) Sensor Applications. *International Journal of Signal Processing Systems*, 256–262. <https://doi.org/10.12720/ijsp.1.2.256-262>
- Andreev, D. (2010). Real-time frame rate up-conversion for video games: Or how to get from 30 to 60 fps for 'free'. *ACM SIGGRAPH 2010 Talks*, 1. <https://doi.org/10.1145/1837026.1837047>
- Award-Winning AI-powered Home Gym Membership*. (n.d.). Tempo. Retrieved 27 October 2023, from <https://tempo.fit>
- Bazarevsky, V., Grishchenko, I., Raveendran, K., Zhu, T., Zhang, F., & Grundmann, M. (2020). *BlazePose: On-device Real-time Body Pose tracking* (arXiv:2006.10204). arXiv. <http://arxiv.org/abs/2006.10204>
- BHOUT - The first boxing bag with a brain*. (n.d.). Retrieved 14 March 2024, from <https://www.bhout.com/>
- Cunningham, S. (2022, January 7). *97% of Premier League academy players never play a minute in top flight, new analysis reveals*. Inews.Co.Uk. <https://inews.co.uk/sport/football/premier-league-academy-players-figures-appearances-numbers-1387302>
- Gong, W., Zhang, X., González, J., Sobral, A., Bouwmans, T., Tu, C., & Zahzah, E. (2016). Human Pose Estimation from Monocular Images: A Comprehensive Survey. *Sensors*, 16(12), Article 12. <https://doi.org/10.3390/s16121966>
- Hanada, Y., Hossain, T., Yokokubo, A., & Lopez, G. (2022). BoxerSense: Punch Detection and Classification Using IMUs. In M. A. R. Ahad, S. Inoue, D. Roggen, & K. Fujinami (Eds.), *Sensor- and Video-Based Activity and Behavior Computing* (pp. 95–114). Springer Nature. https://doi.org/10.1007/978-981-19-0361-8_6

Home. (n.d.). OpenCV. Retrieved 27 October 2023, from <https://opencv.org/>

Lugaresi, C., Tang, J., Nash, H., McClanahan, C., Uboweja, E., Hays, M., Zhang, F., Chang, C.-L.,

Yong, M. G., Lee, J., Chang, W.-T., Hua, W., Georg, M., & Grundmann, M. (2019).

MediaPipe: A Framework for Building Perception Pipelines (arXiv:1906.08172). arXiv.

<http://arxiv.org/abs/1906.08172>

MediaPipe | Google for Developers. (n.d.). Retrieved 27 October 2023, from

<https://developers.google.com/mediapipe>

MediaPipe Solutions guide. (n.d.). Google for Developers. Retrieved 27 October 2023, from

<https://developers.google.com/mediapipe/solutions/guide>

Pauzi, A. S. B., Mohd Nazri, F. B., Sani, S., Bataineh, A. M., Hisyam, M. N., Jaafar, M. H., Ab

Wahab, M. N., & Mohamed, A. S. A. (2021). Movement Estimation Using Mediapipe

BlazePose. In H. Badioze Zaman, A. F. Smeaton, T. K. Shih, S. Velastin, T. Terutoshi, B. N.

Jørgensen, H. Aris, & N. Ibrahim (Eds.), *Advances in Visual Informatics* (pp. 562–571).

Springer International Publishing. https://doi.org/10.1007/978-3-030-90235-3_49

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M.,

Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D.,

Brucher, M., Perrot, M., & Duchesnay, É. (2011). Scikit-learn: Machine Learning in

Python. *Journal of Machine Learning Research*, 12(85), 2825–2830.

Revenue of the Big Five soccer leagues in Europe from 2012/13 to 2021/22, with a forecast to

2023/24, by league. (2024). Statista. [https://www.statista.com/statistics/261218/big-](https://www.statista.com/statistics/261218/big-five-european-soccer-leagues-revenue/)

[five-european-soccer-leagues-revenue/](https://www.statista.com/statistics/261218/big-five-european-soccer-leagues-revenue/)

Statistics About The Most Popular Sports In The World • Gitnux. (2024, February 7).

<https://gitnux.org/most-popular-sports-in-the-world/>

Stefański, P., Jach, T., & Kozak, J. (2023). Classification of Punches in Olympic Boxing Using

Static RGB Cameras. In N. T. Nguyen, J. Botzheim, L. Gulyás, M. Núñez, J. Treur, G.

Vossen, & A. Kozierekiewicz (Eds.), *Computational Collective Intelligence* (pp. 540–551).

Springer Nature Switzerland. https://doi.org/10.1007/978-3-031-41456-5_41

Timeline of A Typical Boxer's Life & Career—By Tony Jeffries. (n.d.). Retrieved 13 March 2024,

from <https://www.boxnburnacademy.com/blog/timeline-boxer-tony-jeffries-career>

Understanding and Applying F1 Score: AI Evaluation Essentials with Hands-On Coding Example.

(n.d.). Retrieved 18 March 2024, from <https://arize.com/blog-course/f1-score/>

Walia, M. S. (2022, January 11). A Comprehensive Guide on Human Pose Estimation. *Analytics*

Vidhya. [https://www.analyticsvidhya.com/blog/2022/01/a-comprehensive-guide-on-](https://www.analyticsvidhya.com/blog/2022/01/a-comprehensive-guide-on-human-pose-estimation/)

[human-pose-estimation/](https://www.analyticsvidhya.com/blog/2022/01/a-comprehensive-guide-on-human-pose-estimation/)

Winter, M. (2023, April 5). *A third of Brits rank exercising highly on their list of holiday activities.*

The Mirror. [https://www.mirror.co.uk/travel/travel-holidays-exercise-fitness-workout-](https://www.mirror.co.uk/travel/travel-holidays-exercise-fitness-workout-29629539)

[29629539](https://www.mirror.co.uk/travel/travel-holidays-exercise-fitness-workout-29629539)

Yamashita, R., Nishio, M., Do, R. K. G., & Togashi, K. (2018). Convolutional neural networks: An

overview and application in radiology. *Insights into Imaging*, 9(4), 611–629.

<https://doi.org/10.1007/s13244-018-0639-9>

Appendix

Code snippet for preprocessing the training data.

```
import Video_Splitter
import Frame_Labeler
import Landmark_Extractor
import random
import numpy as np
import os

video_directory = "./"
video_filename = "Boxing Punches Custom"
video_extension = ".mp4"

output_folder = video_filename + "_split_frames"

train_test_ratio = 0.8

# Splitting the Video

video_path = video_directory + video_filename + video_extension

splitter = Video_Splitter.VideoSplitter(video_path, output_folder)
splitter.split_video_into_frames()

# Labeling the Frames
folder_path = output_folder

possible_labels = ["NO PUNCH", "JAB", "UPPER CUT", "HOOK", "STRAIGHT"]

labeler = Frame_Labeler.FrameLabeler(folder_path, possible_labels)
label_dict = labeler.label_images()

landmark_extractor = Landmark_Extractor.LandmarkExtractor()
landmarks = landmark_extractor.extract_landmarks(folder_path, label_dict)

landmark_label_dict = {}
for key, value in landmarks.items():
    landmark_label_dict[tuple(map(tuple, value))] = label_dict[key]

# Split the data into training and testing data (80% Split)
X_train = []
y_train = []

X_test = []
y_test = []
```

```

keys = list(landmark_label_dict.keys())
random.shuffle(keys)

split_index = round(len(keys) * train_test_ratio)

train_keys = keys[:split_index]
test_keys = keys[split_index:]

train_data = {key: landmark_label_dict[key] for key in train_keys}
test_data = {key: landmark_label_dict[key] for key in test_keys}

for key, value in train_data.items():
    X_train.append(key)
    y_train.append(value)

for key, value in test_data.items():
    X_test.append(key)
    y_test.append(value)

# Create the numpy arrays
np_X_train = np.array(X_train)
np_y_train = np.array(y_train)
np_X_test = np.array(X_test)
np_y_test = np.array(y_test)

# Save the numpy arrays
np_folder_path = "./numpy_arrays"
if not os.path.exists(np_folder_path):
    os.makedirs(np_folder_path)
    print("Folder created successfully at:", np_folder_path)
else:
    print("Folder already exists at:", np_folder_path)

np.save(np_folder_path + "/" + "X_train.npy", np_X_train)
np.save(np_folder_path + "/" + "y_train.npy", np_y_train)
np.save(np_folder_path + "/" + "X_test.npy", np_X_test)
np.save(np_folder_path + "/" + "y_test.npy", np_y_test)

print("Files saved successfully!")

```

Code Snippet for training the CNN.

```

import tensorflow as tf
from tensorflow.keras import layers, models, optimizers
import numpy as np
from sklearn.preprocessing import LabelEncoder
from tensorflow.keras.utils import to_categorical
import matplotlib.pyplot as plt

```



```

# Load data
X_train = np.load('./numpy_arrays/X_train.npy')
y_train = np.load('./numpy_arrays/y_train.npy')
X_test = np.load('./numpy_arrays/X_test.npy')
y_test = np.load('./numpy_arrays/y_test.npy')

# Reshape data to match the input shape of the model
X_train = X_train.reshape(X_train.shape[0], 33, 3, 1)
X_test = X_test.reshape(X_test.shape[0], 33, 3, 1)

# Define the list of unique labels
unique_labels = ["NO PUNCH", "JAB", "UPPER CUT", "HOOK", "STRAIGHT"]

# Initialize the LabelEncoder
label_encoder = LabelEncoder()

# Fit the LabelEncoder to your unique labels
label_encoder.fit(unique_labels)

# Transform your original string labels to numerical labels
y_train = label_encoder.transform(y_train)
y_test = label_encoder.transform(y_test)

# Convert labels to one-hot encoding
num_classes = 5
y_train_encoded = to_categorical(y_train, num_classes=num_classes)
y_test_encoded = to_categorical(y_test, num_classes=num_classes)

# Define the CNN model
def create_cnn_model(input_shape, num_classes):
    model = models.Sequential([
        layers.Conv2D(128, (3, 3), activation='relu', input_shape=input_shape,
padding='same'),
        layers.MaxPooling2D((2, 2)),
        layers.Conv2D(256, (3, 3), activation='relu', padding='same'),
        layers.MaxPooling2D((1, 1)),
        layers.Flatten(),
        layers.Dense(128, activation='relu'),
        layers.Dropout(0.75),
        layers.Dense(256, activation='relu'),
        layers.Dense(num_classes, activation='softmax')
    ])

    return model

# Example input shape
input_shape = X_train.shape[1:]

```

```

# Create the model
model = create_cnn_model(input_shape, num_classes)

# Define the learning rate schedule
def lr_schedule(epoch):
    lr = 0.001

    if epoch > 75:
        lr *= 0.25
    elif epoch > 50:
        lr *= 0.5
    return lr

# Define the learning rate scheduler callback
lr_scheduler = tf.keras.callbacks.LearningRateScheduler(lr_schedule)

# Compile the model
model.compile(optimizer='adam',
              loss='categorical_crossentropy',
              metrics=['accuracy'])

# Train the model with the learning rate scheduler callback
history = model.fit(X_train, y_train_encoded, epochs=125, batch_size=16, validation_data=(X_test, y_test_encoded), callbacks=[lr_scheduler])

# Evaluate the model
test_loss, test_accuracy = model.evaluate(X_test, y_test_encoded)
print("Test accuracy:", test_accuracy)

# Plot accuracy
plt.subplot(1, 2, 1)
plt.plot(history.history['accuracy'], label='Training Accuracy')
plt.plot(history.history['val_accuracy'], label='Validation Accuracy')
plt.xlabel('Epoch')
plt.ylabel('Accuracy')
plt.title('Training and Validation Accuracy')
plt.legend()

# Plot loss
plt.subplot(1, 2, 2)
plt.plot(history.history['loss'], label='Training Loss')
plt.plot(history.history['val_loss'], label='Validation Loss')
plt.xlabel('Epoch')
plt.ylabel('Loss')
plt.title('Training and Validation Loss')
plt.legend()

```

```
plt.show()

# Save the model
model.save("./punch_classifier.keras")
```

Code snippet for the “main system”.

```
import mediapipe as mp
import numpy as np
import cv2
import Punch_Classifier
import time
import Punch_Analyser
import copy
import matplotlib.pyplot as plt # Import Matplotlib for plotting

model_path = './pose_landmarker_heavy.task'

mp_pose = mp.solutions.pose

cap = cv2.VideoCapture(0)

unique_labels = ["NO PUNCH", "JAB", "UPPER CUT", "HOOK", "STRAIGHT"]

# TESTING: List of times to process frames using mediapipe
mediapipe_timings = []

# TESTING: List of times to process frames using the classifier
classifier_timings = []

# TESTING: List of times to process frames using the algorithms
algorithm_timings = []

# Holds the landmarks that trace the punching hand during a movement
punch_trace = [None, []] # = ["Punching_Hand: LEFT or RIGHT", [(Landmark, Pre-
dicted_Class)]]

# Function to process the result from the pose landmark model
def process_result(result):
    landmarks = []
    for landmark in result.pose_landmarks.landmark:
        landmarks.append([landmark.x, landmark.y, landmark.z])

    landmarks = tuple(map(tuple, landmarks))
    landmarks = np.array(landmarks)

    landmarks = landmarks.reshape(1, 33, 3, 1)
```

```

    return landmarks

# Function to get the class of the punch trace
def get_trace_class():
    counter_NO_PUNCH = 0
    counter_JAB = 0
    counter_STRAIGHT = 0
    counter_HOOK = 0
    counter_UPPER_CUT = 0

    for _, classification in punch_trace[1]:
        match classification:
            case "NO PUNCH":
                counter_NO_PUNCH += 1
            case "JAB":
                counter_JAB += 1
            case "STRAIGHT":
                counter_STRAIGHT += 1
            case "HOOK":
                counter_HOOK += 1
            case "UPPER CUT":
                counter_UPPER_CUT += 1

    highest_variable = max(
        [
            ("NO PUNCH", counter_NO_PUNCH),
            ("JAB", counter_JAB),
            ("STRAIGHT", counter_STRAIGHT),
            ("HOOK", counter_HOOK),
            ("UPPER CUT", counter_UPPER_CUT)
        ], key=lambda x: x[1])

    trace_class = highest_variable[0]

    return trace_class

# Function to analyze the punch trace
def analyse_trace():
    trace_class = get_trace_class()

    if trace_class == "JAB" or trace_class == "STRAIGHT":
        global punch_trace
        trace = copy.deepcopy(punch_trace)

        speed = Punch_Analyser.check_punch_speed(trace[1], 30)
        trajectory = Punch_Analyser.check_trajectory(trace[1])

        print("The Punch: " + trace_class)

```

```

        print(f"The punch was thrown at a speed of {speed} units/second")

    if trajectory:
        print("The punch was thrown in a straight trajectory")
    else:
        print("The punch was not thrown in a straight trajectory")

punch_trace = [None, []]

# Initialize MediaPipe PoseLandmark model
with mp_pose.Pose(model_path) as pose_landmarker:
    # Assume you have your OpenCV capture object here named 'cap'
    while cap.isOpened():
        success, image = cap.read()
        if not success:
            print("Ignoring empty camera frame.")
            continue

        # TESTING: Get the time at which a frame is captured
        capture_time = time.time()

        # Convert the BGR image to RGB and process it with MediaPipe PoseLand-
mark model
        image_rgb = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)
        results = pose_landmarker.process(image_rgb)

        # Process Landmarks
        if results.pose_landmarks:
            # TESTING: Get the time that the frame has been processed by medi-
pipe
            mediapipe_process_time = time.time()
            mediapipe_timings.append(mediapipe_process_time - capture_time)

            processed_results = process_result(results)

            # TESTING: Get the time before processing with the classifier
            before_classifier_time = time.time()

            prediction = Punch_Classifier.classify_landmarks(processed_re-
sults)

            predicted_class_index = np.argmax(prediction)
            predicted_class = unique_labels[predicted_class_index]

            # TESTING: Get the time after a prediction has been made by the
classifier
            after_classifier_time = time.time()

```

```

        classifier_timings.append(after_classifier_time - before_classifier_time)

        processed_results = processed_results[0]

        # TESTING: Get the time before the algorithmic analysis
        before_algorithm_time = time.time()

        left_ear = processed_results[7]
        right_ear = processed_results[8]
        left_hand = processed_results[19]
        right_hand = processed_results[20]
        left_foot = processed_results[31]
        right_foot = processed_results[32]
        left_heel = processed_results[29]
        right_heel = processed_results[30]
        right_elbow = processed_results[14]
        left_elbow = processed_results[13]
        right_shoulder = processed_results[12]
        left_shoulder = processed_results[11]
        right_knee = processed_results[26]
        left_knee = processed_results[25]
        right_hip = processed_results[24]
        left_hip = processed_results[23]

        if (Punch_Analyser.is_hand_by_head(left_ear, left_hand) and
Punch_Analyser.is_hand_by_head(right_ear, right_hand)):
            is_heel_raised = Punch_Analyser.is_back_heel_raised(right_heel, right_foot)
            if is_heel_raised:
                print("Back heel is raised")
            else:
                print("Raise your back heel!")

            if (punch_trace[0] != None):
                analyse_trace()

        elif (Punch_Analyser.is_hand_by_head(left_ear, left_hand) and not
Punch_Analyser.is_hand_by_head(right_ear, right_hand)):
            if (punch_trace[0] == "RIGHT"):
                punch_trace[1].append((right_hand, predicted_class))
            elif (punch_trace[0] == None):
                punch_trace[0] = "RIGHT"
                punch_trace[1].append((right_hand, predicted_class))
            else:
                analyse_trace()
                punch_trace[0] = "RIGHT"
                punch_trace[1].append((right_hand, predicted_class))

```



```

        elif (Punch_Analyser.is_hand_by_head(right_ear, right_hand) and
not Punch_Analyser.is_hand_by_head(left_ear, left_hand)):
            if (punch_trace[0] == "LEFT"):
                punch_trace[1].append((left_hand, predicted_class))
            elif (punch_trace[0] == None):
                punch_trace[0] = "LEFT"
                punch_trace[1].append((left_hand, predicted_class))
            else:
                analyse_trace()
                punch_trace[0] = "LEFT"
                punch_trace[1].append((left_hand, predicted_class))

        right_elbow_angle_text = "Right Elbow: " + str(Punch_Ana-
lyser.get_joint_angle(right_hand, right_elbow, right_shoulder))
        left_elbow_angle_text = "Left Elbow: " + str(Punch_Ana-
lyser.get_joint_angle(left_hand, left_elbow, left_shoulder))
        right_knee_angle_text = "Right Knee: " + str(Punch_Ana-
lyser.get_joint_angle(right_foot, right_knee, right_hip))
        left_knee_angle_text = "Left Knee: " + str(Punch_Ana-
lyser.get_joint_angle(left_foot, left_knee, left_hip))

        # Get the time after the algorithmic analysis has been completed
        after_algorithm_time = time.time()
        algorithm_timings.append(after_algorithm_time - before_algo-
rithm_time)

        # Draw landmarks on the image (optional)
        annotated_image = image.copy()
        mp.solutions.drawing_utils.draw_landmarks(annotated_image, re-
sults.pose_landmarks, mp_pose.POSE_CONNECTIONS)
        cv2.putText(annotated_image, right_elbow_angle_text, (20, 20),
cv2.FONT_HERSHEY_SIMPLEX, 0.5, (0, 0, 255), 1)
        cv2.putText(annotated_image, left_elbow_angle_text, (20, 40),
cv2.FONT_HERSHEY_SIMPLEX, 0.5, (255, 0, 0), 1)
        cv2.putText(annotated_image, right_knee_angle_text, (20, 60),
cv2.FONT_HERSHEY_SIMPLEX, 0.5, (0, 0, 127), 1)
        cv2.putText(annotated_image, left_knee_angle_text, (20, 80),
cv2.FONT_HERSHEY_SIMPLEX, 0.5, (127, 0, 0), 1)
        cv2.imshow('MediaPipe Pose Landmarks', annotated_image)

        # Break the loop when 'q' is pressed
        if cv2.waitKey(1) & 0xFF == ord('q'):
            # Plot the timings using Matplotlib

            mediapipe_mean = np.mean(mediapipe_timings)
            mediapipe_max = np.max(mediapipe_timings)
            mediapipe_min = np.min(mediapipe_timings)

```

```

        classifier_mean = np.mean(classifier_timings)
        classifier_max = np.max(classifier_timings)
        classifier_min = np.min(classifier_timings)

        algorithm_mean = np.mean(algorithm_timings)
        algorithm_max = np.max(algorithm_timings)
        algorithm_min = np.min(algorithm_timings)

        plt.subplot(3, 1, 1)
        plt.scatter(range(len(mediapipe_timings)), mediapipe_timings,
color='blue', s=5)
        plt.title(f"Mediapipe: Mean = {mediapipe_mean:.4f}, Max = {medi-
apipe_max:.4f}, Min = {mediapipe_min:.4f}")
        plt.xlabel('Frame')
        plt.ylabel('Time (s)')
        plt.legend()

        plt.subplot(3, 1, 2)
        plt.scatter(range(len(classifier_timings)), classifier_timings,
color='green', s=5)
        plt.title(f"Classifier: Mean = {classifier_mean:.4f}, Max = {clas-
sifier_max:.4f}, Min = {classifier_min:.4f}")
        plt.xlabel('Frame')
        plt.ylabel('Time (s)')
        plt.legend()

        plt.subplot(3, 1, 3)
        plt.scatter(range(len(algorithm_timings)), algorithm_timings,
color='red', s=5)
        plt.title(f"Algorithm: Mean = {algorithm_mean:.4f}, Max = {algo-
rithm_max:.4f}, Min = {algorithm_min:.4f}")
        plt.xlabel('Frame')
        plt.ylabel('Time (s)')
        plt.legend()

        plt.tight_layout()
        plt.show()

        print("Mediapipe: " + str(sum(mediapipe_timings) / len(medi-
apipe_timings)))
        print("Classification: " + str(sum(classifier_timings) / len(clas-
sifier_timings)))
        print("Algorithm: " + str(sum(algorithm_timings) / len(algo-
rithm_timings)))

        # Reset the timings
        mediapipe_timings = []
        classifier_timings = []

```

```

        algorithm_timings = []

        break

# Release the capture
cap.release()
cv2.destroyAllWindows()

```

Code snippet for the algorithmic analysis.

```

import math
import numpy as np

def get_distance_between(landmark1, landmark2):
    x1 = landmark1[0]
    y1 = landmark1[1]
    z1 = landmark1[2]

    x2 = landmark2[0]
    y2 = landmark2[1]
    z2 = landmark2[2]

    dist = math.sqrt((x2 - x1)**2 + (y2 - y1)**2 + (z2 - z1)**2)
    return dist

def check_punch_speed(trace, fps):

    number_of_frames = len(trace)

    time = number_of_frames / fps

    distance = get_distance_between(trace[number_of_frames - 1][0],
trace[0][0])

    speed = distance / time

    return speed

def is_back_heel_raised(heel_landmark, foot_landmark):
    if (foot_landmark[2] > heel_landmark[2]):
        return True
    else:
        return False

def is_hand_by_head(head_landmark, hand_landmark):
    # Can be changed depending on the required sensitivity
    head_hand_threshold = 0.3

```

```

dist = get_distance_between(head_landmark, hand_landmark)
if (dist < head_hand_threshold):
    return True
else:
    return False

def check_trajectory(trace):
    # Can be changed depending on the required sensitivity
    tolerance = 0.1

    # Extracting the landmarks
    x1 = trace[0][0][0]
    y1 = trace[0][0][1]
    z1 = trace[0][0][2]

    x2 = trace[-1][0][0]
    y2 = trace[-1][0][1]
    z2 = trace[-1][0][2]

    # Calculate the direction vector of the line
    v = np.array([x2 - x1, y2 - y1, z2 - z1])

    # Check the distance of each landmark from the line
    for landmark in trace:
        xP = landmark[0][0]
        yP = landmark[0][1]
        zP = landmark[0][2]

        # Calculate the vector from P1 to P
        P1P = np.array([xP - x1, yP - y1, zP - z1])

        P1P = P1P.reshape(-1)
        v = v.reshape(-1)

        # Calculate the projection of P1P onto v
        projection = np.dot(P1P, v) / np.dot(v, v) * v

        # Calculate the distance between P and the point on the line
        distance = np.linalg.norm(P1P - projection)

        if (distance > tolerance):
            return False

    return True

def get_joint_angle(adj_joint1, target_joint, adj_joint2):

    adj1_X = adj_joint1[0][0]

```

```
adj1_Y = adj_joint1[1][0]
adj1_Z = adj_joint1[2][0]

adj2_X = adj_joint2[0][0]
adj2_Y = adj_joint2[1][0]
adj2_Z = adj_joint2[2][0]

targ_X = target_joint[0][0]
targ_Y = target_joint[1][0]
targ_Z = target_joint[2][0]

adj_joint1 = (adj1_X, adj1_Y, adj1_Z)
adj_joint2 = (adj2_X, adj2_Y, adj2_Z)
target_joint = (targ_X, targ_Y, targ_Z)

# Define vectors representing the two lines
vector1 = np.array(target_joint) - np.array(adj_joint1)
vector2 = np.array(target_joint) - np.array(adj_joint2)

# Calculate dot product and magnitudes
dot_product = np.dot(vector1, vector2)
magnitude_vector1 = np.linalg.norm(vector1)
magnitude_vector2 = np.linalg.norm(vector2)

# Calculate cosine of the angle between the lines
cos_angle = dot_product / (magnitude_vector1 * magnitude_vector2)

# Convert cosine to angle in radians
angle_in_radians = np.arccos(cos_angle)

# Convert radians to degrees
angle_in_degrees = np.degrees(angle_in_radians)

return angle_in_degrees
```

Transcript of the discussion with the coaches.

[Interviewer]: Thank you all for your participation. I'd like to take a minute to discuss with you a new technology that is being tested for use within Boxing training. The technology makes use of pose detection to track the athlete's body in real-time. The detected poses can then be analysed to provide feedback to the athlete. If you would like to have a look, I have a system that I have developed that I would love for you to try out.

*** Interacting with the developed system ***

[Interviewer]: Thank you for taking time to interact with the system. Would it be possible to gather your thoughts?

[Coach 1]: Of course!

[Coach 2]: Ok.

[Coach 3]: Sure, go ahead!

[Interviewer]: Great! I guess I'll start by asking: what are your initial thoughts on the technology?

[Coach 1]: Well, it's certainly an interesting concept. Anything that can enhance training and technique in boxing is worth exploring. The technology is allowing boxers to get real-time feedback on their form which is invaluable for improvement.

[Interviewer]: That sounds promising. Coach 2, do you see any potential drawbacks to this system?

[Coach 2]: Absolutely. While it's great for refining technique, there's a risk of overreliance on technology. Boxing is as much about instinct and feel as it is about form. Relying too heavily on a detection system could detract from that.

[Coach 1]: I agree with Coach 2. Plus, there's the issue of accuracy. These systems aren't infallible. A false reading could lead to incorrect adjustments and potentially harm a boxer's progress.

[Interviewer]: Interesting points. Coach 3, what do you think could be done to mitigate these concerns?

[Coach 3]: Well, I think it's about finding a balance. The system can be a useful tool, but it shouldn't replace traditional coaching methods entirely. It's about using it as a supplement rather than a crutch. And of course, regular calibration and testing to ensure accuracy is crucial.

[Coach 2]: Exactly. It's about integrating technology into training without losing sight of the fundamentals of the sport.

[Interviewer]: Thank you all so much for your time. It's been a pleasure sharing this with you and gathering your thoughts on the technology. Thank you!

[Coach 1]: No worries! It's nice to see new technology coming into the sport.

[Coach 2]: Yes, maybe one day you can show us the finished product. We may even get a discount if we're lucky! [LOTS OF LAUGHING]

[Interviewer]: I hope that is the case. I can't promise you any discounts just yet though. Well, thank you once again for your time. I'm sure your very busy so I'll let you get back to your coaching.

[Coach 1]: In a bit.

[Coach 2]: ta-ra.

[Coach 3]: ta-ra.

[Interviewer]: ta-ra.

