A computer-vision based training coach for computerised physical training

UNIVERSITY OF
LINCOLN

George Davies
27421138

27421138@lincoln.ac.uk

School of Computer Science

College of Science

University of Lincoln

Submitted in partial fulfilment of the requirements for the
Degree of MSc Robotics and Autonomous Systems

Supervisor Dr. Christos Franzidis

August 2024

Acknowledgements

Firstly, I want to thank Dr Christos Frantzidis for coming up with this project's idea and providing me with valuable guidance and feedback. I also want to thank the AgriFoRwArdS CDT for funding my studies this academic year and providing me with experiences I would not have otherwise had.

Abstract

The following report deals with the development of a computer vision-based training coach for which the goal was to assist a person in physical training through real-time feedback and pose estimation. The system, developed on the Mediapipe framework from Google, provides very high-accuracy detection of human poses and rich feedback regarding form during exercise to improve performance while minimising chances of injury. This has a full-fledged GUI at the front-end and a back-end database that will store user data and history of their workouts, and a mood-tracking feature to study the psychological impact of exercises. In controlled settings, accuracy was high; however, poor lighting created problems. Users also felt the GUI was easy to operate and visual feedback is helpful, although it needs better instructions and the text must be clearer. Unlike wearable fitness trackers, more informative feedback regarding form can be provided by the computer vision-based system, an addition to quite a bit of the available current technologies in fitness. Future work includes addition of more types of exercises, more advanced machine learning models, and making the system robust. The project has practical implications for both computer vision and physical training, providing a foundation for further research and development in these fields.

Code: Github Page

Table of Contents

List of Figures

Chapter 1

Introduction

In many stories of science-fiction, there are robots that are intelligent, borderline human in the way they talk, the way they walk, and the way they interact with their environments. Famous examples of this are C-3PO from Star Wars, Sunny from I-Robot, and Wall-E for the eponymous Disney animation. These kinds of robots seem to be creatures that won't exist any time soon, and they probably won't, but with every advancement in the field of robotics and autonomous systems, we step closer to that reality.

One of the most key human senses, which provides a way for a robot to effectively communicate with and understand the real world, is sight. As with human vision, which is essential in interpreting and acting upon the surrounding world, so seeing confers a whole range of possibilities upon a machine. Computer vision is an area of robotics, dedicated to providing machines with the sense of sight. The computing of visual information, identification of objects, and consequent decision-making in respect to the environment bring us closer to the intelligent, interactive robots that have long been envisioned in science fiction.

1.1 Background

Human Pose Estimation (HPE) research started when computer vision gained popularity in the late 1960s and early 1970s. Scientists first focused on basic problems like as shape analysis, object recognition, and visual understanding. As computer vision developed, HPE became a stand-alone area of study (Lynn, 2023). Historically, HPE was frequently described probabilistically to account for likely inference

ambiguities. Since deep learning has been more widely used, the focus has switched to end-to-end trainable models because of their ability to extract intricate patterns and postures from data. Traditionally, computer vision systems have assessed an object's or person's posture by geometric calculations and feature-based techniques. But, the biggest developments in HPE came with the advent of deep neural networks, convolutional neural networks, and computer vision. The field has advanced considerably in spite of these challenges, and more recent techniques that make use of properly designed neural networks may provide amazing results in challenging scenarios involving a large number of, perhaps veiled, interacting individuals (M. Liu and Yuan, 2018). Now that these detections have the necessary technology and are sufficiently precise, they may be employed for commercial purposes. It also offers a wealth of new application potential and signifies a major change in HPE's overall direction.

The HPE innovations unlock the potential for using state-of-the-art computer vision technologies within physical training and, hence, change the way athletes and fitness enthusiasts everywhere can improve their performance.

The use of technology in physical training has transformed the way athletes and fitness enthusiasts approach their routines in recent years. Although traditional coaching techniques have long been successful in directing athletes, they can be improved by utilising the accuracy and instantaneous feedback that contemporary technology offers. The creation of computer-vision based training coaches, which use cutting-edge algorithms and machine learning to assess and improve physical performance, has been made possible by this gap. (Debnath et al., 2022)

The automatic extraction, analysis, and comprehension of meaningful information from a single image or a set of images is the focus of the computer vision area of artificial intelligence. Computer vision systems can precisely track and evaluate motions made during physical exercise, providing instantaneous feedback and coaching in real time. This technology helps prevent accidents in addition to assisting with form and technique modification by identifying improper actions. (Borisov et al., 2023)

Thanks to developments in wearable technology and high-definition cameras, com-

puter vision based training systems are now easy to utilise. These gadgets can record precise movement data, which is evaluated by astute algorithms to produce valuable results. Thus, training regimens that are especially designed for athletes and adjust to their unique demands and development may be beneficial.

Furthermore, the COVID-19 epidemic has sped up the transition in a number of industries, including sports and fitness, towards remote and digital solutions. The need for online coaching and training resources has increased as a result of lockdowns and social distancing. As a workable alternative, computer-vision based training instructors let customers to continue their exercise regimens under the supervision of professionals from the comfort of their own homes. It is important to note that this is not intended as a replacement for professional coaches, rather as an aid for them to have better effectivness.

1.2 Area of Research

HPE is an area of research within computer vision that aims to teach robots how to make sense of the human form and the motions it is capable of performing. It involves the identification and classification of the joints in the human body, capturing a set of coordinates for each joint, known as a key point, that can describe the pose of a person. HPE has a wide set of uses in many fields: In games, with motion capture technologies reliant on HPE, it allows developers to encode more realistic and fluid character movements. In healthcare, healthcare providers can monitor a patient's movements and detect any abnormalities. Augmented reality, allows the user to interact with the digital content in more natural and intuitive ways with gestures. And finally, the use-case that is the primary focus of this project, is sports training. HPE can be used to analyse a user's performance, identify areas for improvement, and develop personalised training programs based on the physical level of the user. For example, HPE could be used to analyse a runner's form, e.g. How straight is their back? What part of the foot are they landing on? Are they leaning more to one side or landing more heavily on one foot?..., and provide feedback on how to

improve their technique. HPE can be used to collect data about any exercises where the movement of the body is vital to its effectiveness.

1.3 Motivation

Over the last few decades, the percentage of people who are either overweight or obese has steadily increased. This is a concerning trend as being obese increases the risk of many other health conditions such as type 2 diabetes, coronary heart disease, some types of cancer, and stroke (NHS, 2023). As seen in Figure 1.1, over 60% of the UK population is considered overweight or obese.

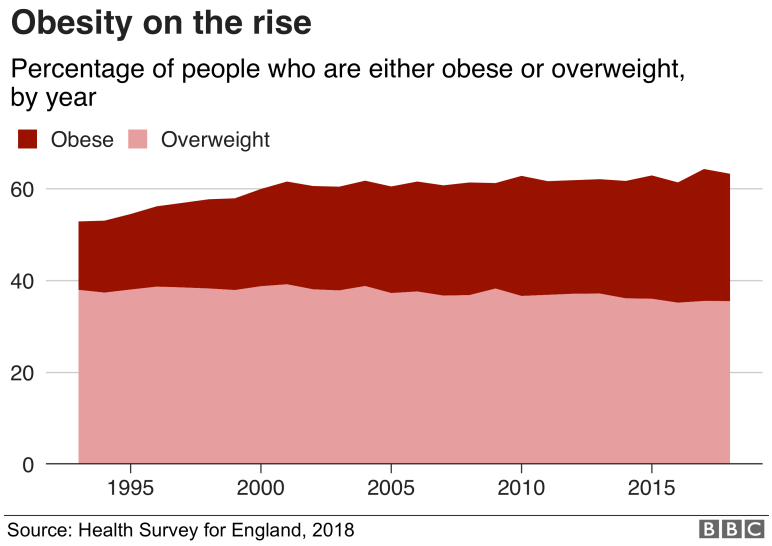

Figure 1.1: Percentage of people who are either obese or overweight, by year (Source: Health Survey for England, 2018)

To help solve that issue, it is important to encourage people to workout, but going to the gym can be daunting, especially when you don't know how to perform exercises properly. Creating an AI personal trainer will give people the knowledge and confidence to exercise more at home or the gym. Furthermore, exercise has many benefits other than weight loss, it reduces the risk of the issues mentioned above and has been shown to improve mental health (NHS, 2021).

1.4 Problem Statement

It is difficult to obtain real-time, individualised input for physical exercise, especially when it is done remotely. Exercise enhances both mental and physical health, but without the right supervision, people run the danger of getting hurt and don't obtain the full benefits of working out. The necessity for remote training solutions has been made evident by the COVID-19 outbreak increasing the popularity of working out at home instead of at the gym. Accurate, real-time feedback can be given by a computer-vision based training coach, which can enhance training, reduce the risk of injury, and promote mental health.

1.5 Scope of the Project

This project aims to create a program that will act as a personal trainer and that will track the mood of the user to determine the effect of physical exercise on acute mood. This means using a computer vision human pose estimator to find the landmarks on the user's body to determine their shape while asking them about their mood before and after an exercise session. This project will not use any deep learning methods to train any models for the recommendation system and repetition counter as no exercise data will be collect due to time constraints. Simple mathematical methods will be applied instead. The only data that will be collected will be user feedback solely for the purpose of evaluating the performance of the program.

1.6 Structure of the Thesis

The structure of the thesis will proceed as follows. First I will perform a literature review of related work about the different uses of Computer Vision methods, this will include reviewing object detection, object segmentation, human pose estimation, and motion capture. I will also quickly review papers that explore the link between physical exercise and its effect on mood and mental health.

Based on the findings from the literature review, I will lay out my aims and objectives for this project with the goal of having them be novel and have little previous

research. These aims and objectives will be Specific, Measurable, Assignable, Realistic and Time-related as per the SMART principles.

Following that, I will describe the methodology of the project, outlining the research design, the architecture of the system, the selection of the algorithm, the development environment including the tools and software used and the version control, the ethical considerations of the project, and an analysis of the risks that will potentially be faced.

Subsequently, I will detail how I implemented the program, first by specifying the system requirements (both software and hardware), then I'll move on to recounting of the creation of the application by describing the integration of the model and the real-time processing.

After the creation of the application, this is the creation of the Graphical User Interface (GUI) going through the development of both the frontend and the backend. The final section necessary for the detailing of the implementation is the testing and debugging with unit testing and integration testing.

For the penultimate chapter, I will evaluate the performance in results and discussion. In this chapter, I will compare this project with existing solution, I will ponder the user feedback from the usability testing, I will go over the limitations of this project, and finally, provide my interpretation of the results.

And Finally, to conclude, I will summarise the findings, describe the contributions to the field, explain the practical implications, make recommendations for future work, and provide my final thoughts on the entire project.

Chapter 2

Literature Review

Computer Vision is a subfield of Artificial Intelligence that has for goal to give machines the sense of vision, or in other words, the ability to perceive and recognise objects. Within Computer Vision there are many more subfields, such as object detection, object segmentation, human pose estimation, motion detection, etc. These are the methods I will be looking at in this review.

2.1 Object Detection

The ability to identify and locate objects in an image or video is one of the fundamental objectives of computer vision. To improve the accuracy and efficiency of object detection, many methods have been developed over time. Histogram of gradients (HOG) and Haar cascades are examples of conventional methods. The Haar cascades, developed by Paul Viola and Michael Jones, use a series of fundamental classifiers to identify objects based on features such as edges and lines that are derived from integral images. Their capacity to recognise faces has earned them particular notoriety (Viola and Jones, 2001). HOG descriptors, which are helpful for recognising pedestrians and other things with different forms, were developed by Navneet Dalal and Bill Triggs. They function by first dividing an image into small cells, then calculating the gradient orientation histogram for each of those cells, and finally normalising the output (Dalal and Triggs, 2005).

Deep learning algorithms have considerably improved the detection of objects. Region-

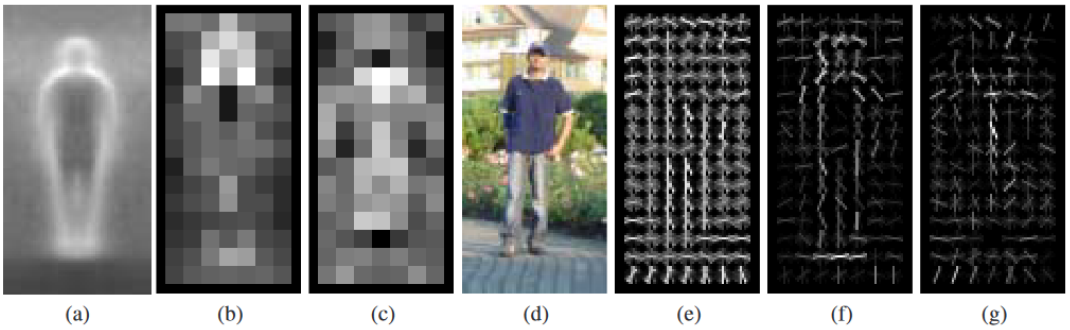

Figure 2.1: Navneet Dalal and Bill Triggs' HOG detectors cue mainly on silhouette contours (especially the head, shoulders and feet). The most active blocks are centred on the image background just outside the contour. (a) The average gradient image over the training examples. (b) Each "pixel" shows the maximum positive SVM weight in the block centred on the pixel. (c) Likewise for the negative SVM weights. (d) A test image. (e) It's computed R-HOG descriptor. (f,g) The R-HOG descriptor weighted by respectively the positive and the negative SVM weights. (Dalal and Triggs, 2005)

Based Convolutional Neural Networks, or R-CNN, by Ross Girshick et al., resorted to a two-step process of generating region proposals and classifying each proposal using a CNN. While this gave better accuracy in detection, it added a large computational cost (R. Girshick et al., 2014). Fast R-CNN by Ross Girshick does this with even better performance by integrating the recommendation of regions and classification into a single network. There will be one RoI pooling layer to pool the features of each proposal (R. Girshick, 2015). Faster R-CNN by Shaoqing Ren et al. incorporates region proposal networks, or RPNs for short, that share some convolutional layers with the detection network to provide near cost-free region proposals, thus speeding up the detection (Ren et al., 2015).

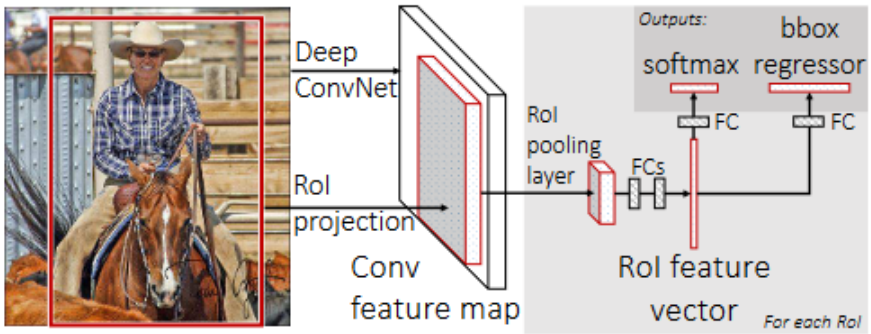

Figure 2.2: Fast R-CNN architecture.(R. Girshick, 2015)

Single Shot MultiBox Detector (SSD) by Wei Liu et al. bypass region proposal completely and use a single network to predict object classes and bounding boxes directly from feature maps at all scales. It goes in direct balance with speed and accuracy for the real-time application (W. Liu et al., 2016). Along this line, Tsung-Yi Lin et al. form the above class imbalance problem in the dense object detection task to the newly introduced function Focal Loss in RetinaNet. This is done by down-weighting the loss of the well-explained model instances to give the most focus to the hard-to-classify instances (Lin et al., 2017).

These methods present the evolution of object detection techniques from traditional ones, based on handcrafted features, to the modern deep learning-based framework with the power of Convolutional Neural Networks. This particular feature is strong in one visual function or another, and choice in using a method can often come down to the specifics of the application requirements, such as speed, accuracy, and computational resources.

2.2 Methods of Segmentation

Image segmentation is considered one of the most prominent tasks done in the field of computer vision. The process can be defined as partitioning a given image into a number of segments or regions, where every part represents some meaningful portion of the original image. Several schemes have been developed through the years to achieve perfect and efficient segmentation. The classical ones include thresholding, edge detection, and region-based segmentation. Thresholding: This type of simple segmentation simply takes a greyscale image and the threshold values to convert the image into a binary image. All pixels that have intensity values larger than the threshold are considered as foreground, and those with lower intensity values are classified under background. The technique of this kind basically works for those images that have huge differences in the values of intensity between objects and their background (Otsu et al., 1975). Edge detection methods, exemplified by the classic one—Canny edge detector—are designed to identify object boundaries in an image

by the detection of intensity discontinuities. They are used in applications where it can segment objects having well-defined edges. One of the most used was that developed by John Canny, known as the Canny edge detector, for its ability to detect edges with low error rates and good localisation (Canny, 1986). Methods for region-based image segmentation group pixels together with certain similar properties. The major methods are region growing and region splitting and merging. Region growing does the segmentation from user-specified starting points by picking neighboring pixels that meet the specified criteria. Region splitting and merging divide an image into local regions and then combine them into groups using measures of similarity (Adams and Bischof, 1994).

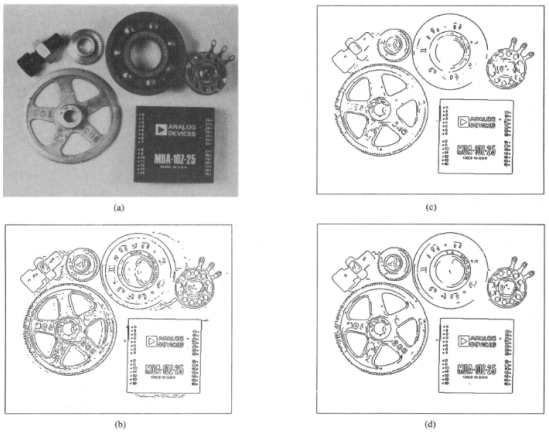

Figure 2.3: (a) Parts image, 576 by 454 pixels. (b) Image thesholded at T,. (c) Image thresholded at 2 T,. (d) Image thresholded with hysteresis using both the thresholds in (a) and (b). (Canny, 1986)

Deep learning techniques have significantly enhanced image segmentation procedures. Fully Convolutional Networks are one kind of neural network designed explicitly for pixel-wise prediction tasks by Jonathan Long et al. In general, an FCN is an extension of the traditional CNNs; it replaces fully connected layers with convolutional layers so that the network outputs a segmentation map of the same size as the input image. It increased the accuracy of semantic segmentation to a large extent (Long, Shelhamer and Darrell, 2015). Probably the most popular biomedical image segmentation architecture is the U-Net introduced by Olaf Ronneberger et al. The U-Net represents a network with an encoder-decoder architecture that provides

high-resolution features from the encoder, which are concatenated with upsampled features from the decoder through skip connections. This design enables retaining spatial information and improves segmentation accuracy (Ronneberger, Fischer and Brox, 2015). Mask R-CNN, introduced by Kaiming He et al., extends Faster R-CNN with an added branch for predicting a segmentation mask on each Region of Interest (RoI). While Faster R-CNN just localises the object, the method can do instance segmentation—in that each object instance is segmented on its part. Widespread adoption has taken place due to the accuracy and versatility of Mask R-CNN (He et al., 2017). The DeepLab was developed by Liang-Chieh Chen et al. and uses atrous or dilated convolutions to stride contextual information at multiple scales without losing resolution. DeepLab also integrates Conditional Random Fields that help refine segmentation borders, enabling it to effectively segment complex-shaped and scale-varying objects (L.-C. Chen et al., 2017).

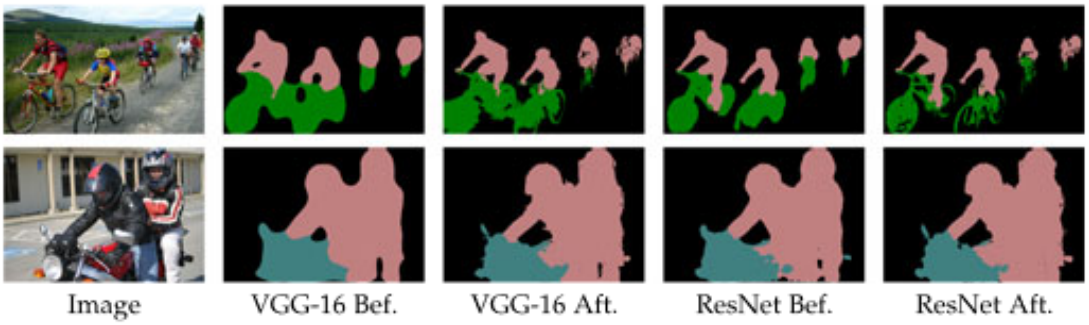

Figure 2.4: DeepLab results based on VGG-16 net or ResNet-101 before and after CRF. The CRF is critical for accurate prediction along object boundaries with VGG-16, whereas ResNet-101 has acceptable performance even before CRF.(L.-C. Chen et al., 2017)

The methods include traditional heuristic approaches to image segmentation and more advanced and deep learning-based frameworks that are empowered by neural networks. Each of the methods will have a share in strong and weak points and be more suitable for certain applications than others depending on the accuracy required, computational efficiency, and the type of complexity involved in the segmentation.

2.3 Human Pose Estimation (HPE)

Human pose estimation is a process for detecting and tracking the location of key body joints in images or videos; it is thus of paramount importance in the field of computer vision. This has been applied to motion capture, activity recognition, human–computer interaction, and other areas. Two traditional solutions are pictorial structures and deformable part models. Pictorial structures model a human body as a collection of rigid parts connected by flexible joints. Each part is modeled using a template, and configuration of parts is optimised to fit the observed image. This framework, thanks to Felzenszwalb and Huttenlocher has been quite successful in detecting human poses in static images (Felzenszwalb and Huttenlocher, 2005). Deformable Part Models generalise pictorial structures to allow for deformation of parts relative to each other. Felzenszwalb et al.'s method represents the object class by a mixture of templates, each capturing appearance and pose variations. DPMs have been used in many applications involving object detection and human pose estimation (Felzenszwalb, R. B. Girshick et al., 2009).

Figure 2.5: Top: Multi-person pose estimation. Body parts belonging to the same person are linked. Bottom left: Part Affinity Fields (PAFs) corresponding to the limb connecting right elbow and right wrist. The color encodes orientation. Bottom right: A zoomed in view of the predicted PAFs. At each pixel in the field, a 2D vector encodes the position and orientation of the limbs. (Cao et al., 2017)

Deep learning-based methods have greatly advanced human pose estimation. Employing a sequence of convolutional networks to iteratively predict the locations of body joints, the Convolutional Pose Machines method has already shown very accurate results. In each stage of this network, it is expected to refine the prediction

in the previous stage, hence improving the accuracy in pose estimation iteratively (Wei et al., 2016). The Stacked Hourglass Networks by Newell et al. uses symmetric encoder-decoder architecture capturing features at many scales. Such a network is made up of many hourglass modules stacked together, so that it is feasible to perform bottom-up and top-down processing repetitively. This algorithmic approach has achieved a top performance in human pose estimation (Newell, Yang and Deng, 2016). OpenPose, developed by Cao et al., is a real-time multi-person pose estimation framework. It uses a two-branch CNN to predict part affinity fields (PAFs) and keypoint locations simultaneously. PAFs encode the spatial relationships between body parts, enabling the network to associate detected keypoints with individual people in the image (Cao et al., 2017). PoseNet by Kendall et al. comprises a deep learning framework for estimating human poses from images. This approach uses a CNN to directly predict the 2D coordinates of body joints from an input image and can be end-to-end trained to solve a wide variety of pose estimation tasks. PoseNet has been identified to work quite well in terms of accuracy and to be very robust in very challenging conditions (Kendall, Grimes and Cipolla, 2015).

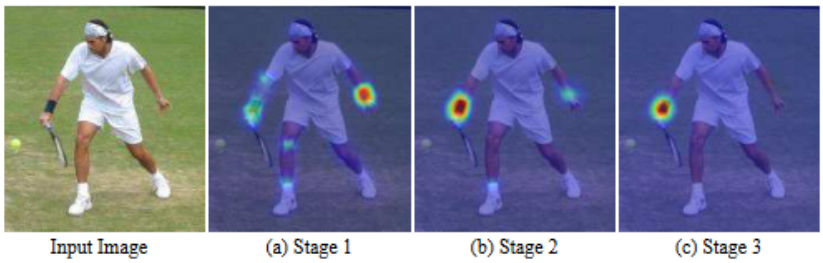

Figure 2.6: Here Wei et al. show the increasingly refined estimates for the location of the right elbow in each stage of the sequence. (a) Predicting from local evidence often causes confusion. (b) Multi-part context helps resolve ambiguity. (c) Additional iterations help converge to a certain solution. (Wei et al., 2016)

These methods reflect the journey that began with traditional approaches, which relied on handcrafted models for human-pose estimation, to deep-learning-based advanced frameworks exploiting the power of convolutional neural networks. Each approach has its strengths and weaknesses, and very often, the choice of method depends on the kind of application at hand, that is, accuracy, computational efficiency, or the complexity of poses estimated.

2.4 Motion Tracking

One of the big modules of computer vision, motion tracking involves the tracking of a human or any other entity from one frame to another in a video sequence. Major applications include surveillance, sports analysis, and augmented reality. Traditional approaches include optical flow, the Kalman filter, and mean shift. Most of the approaches in use for calculating the motion of an object via optical flow generally make use of the apparent motion of the brightness patterns between consecutive frames. Probably the most famous among these, which assumes constant flow in some small neighbourhood and finds the motion using least squares fitting, is the method proposed by Bruce D. Lucas and Takeo Kanade (Lucas and Kanade, 1981). Another popular method is based on the Horn-Schunck algorithm with the smoothness constraint of the flow field in order to cope with noise and discontinuities (Q. Chen and Koltun, 2016). Basically, the Kalman filter is a principle of state estimation for a dynamic system from noisy observations by a recursive algorithm. Since it finds applications in prediction regarding position and velocity for moving objects, it is normally used in motion tracking. Prediction and update are basically the two working steps of the Kalman filter, and the same have been designed efficiently to handle problems regarding real-time tracking (Kalman, 1960). Mean shift is a non-parametric feature-space analysis technique applied for finding the maxima of a density function. Applied to motion tracking, it shifts a window iteratively to the area with the highest density of features and uses it to track an object. Thus, this approach is robust against appearance variations of the object and partial occlusions (Comaniciu, Ramesh and Meer, 2000).

Deep learning methods have already shown a high improvement in the state-of-the-art performance in motion tracking. RNNs, and especially the LSTM network, conduct motion tracking through modelling temporal dependencies in sequential data. They hence learn how to predict the future positions of objects from their previous trajectories and hence can be applied to track complicated nonlinear motions (Hochreiter and Schmidhuber, 1997). Koch et al. have proposed Siamese networks

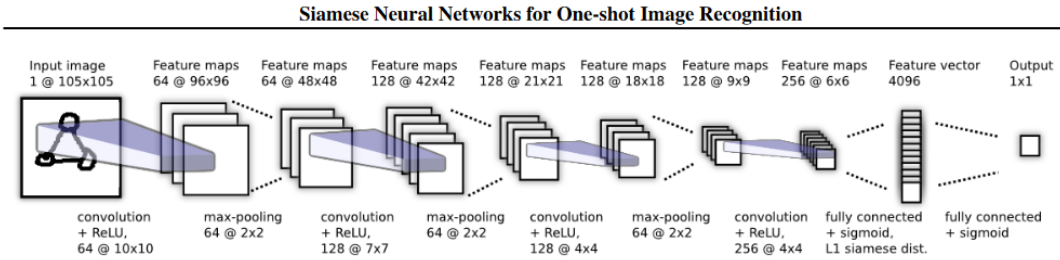

Figure 2.7: Best convolutional architecture selected for verification task. Siamese twin is not depicted, but joins immediately after the 4096 unit fully-connected layer where the L1 component-wise distance between vectors is computed (Koch, Zemel, Salakhutdinov et al., 2015)

for application in one-shot learning problems like object tracking. It consists of two identical sub-networks processing two different inputs and returning the similarity score as output. In the case of motion tracking, the Siamese Networks could be trained for the differentiation of target from background and offer robustness under challenging conditions (Koch, Zemel, Salakhutdinov et al., 2015). DeepSORT hybridises the SORT algorithm with a deep appearance descriptor. This approach uses a CNN for extracting features from detected objects and a Kalman filter, which provides motion prediction. By combining both appearance and motion information, this significantly improves not only accuracy but also resilience in tracking (Wojke, Bewley and Paulus, 2017). Huang et al. proposed a deep framework called TrackNet for sports tracking. It uses a CNN to predict the heatmap in every frame on the location of the target and further refines these predictions with post-processing. On the other hand, TrackNet was able to track fast-moving targets such as the badminton shuttlecock and tennis ball (Huang et al., 2019).

These methods very clearly show how far the progress of motion-tracking techniques has moved from more traditional, math-model-based methods to advanced, deep learning-based frameworks that exploit the power of neural networks. But each has its strengths and weaknesses, and often, the choice of the method depends on the specific requirements of the application.

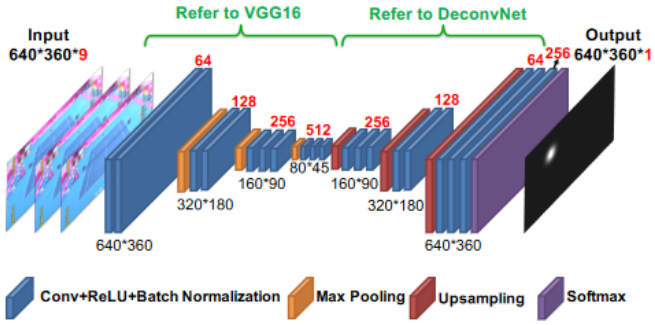

Figure 2.8: The architecture of the proposed TrackNet (Huang et al., 2019)

2.5 The Cutting Edge of HPE

There has been huge progress in HPE, with sophisticated models like YOLO by Ultralytics and Google's MediaPipe. While both of these frameworks offer very powerful detection and tracking tools for human pose, they differ in approach and capabilities.

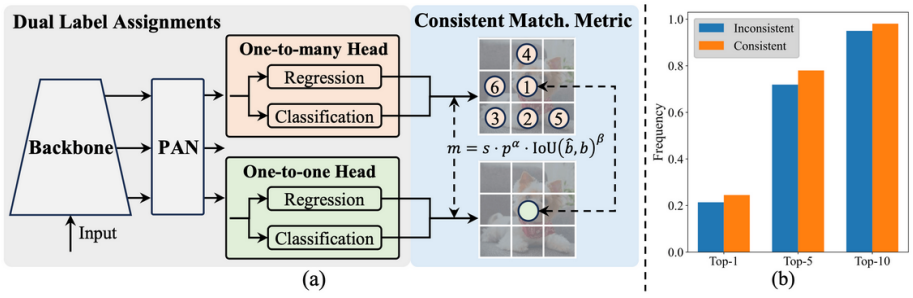

Figure 2.9: (a) Consistent dual adddignments for NMS-free training. (b) Frequency of one-to-one assignments in Top-1/5/10 of one-to-many results for YOLOv8-S which employs α_{o2m}=0.5 and β_{o2m}=6 by default. (https://docs.ultralytics.com/models/yolov10/)

2.5.1 YOLO by Ultralytics

YOLO stands for 'You Only Look Once,' referring to a family of real-time object detection models that have evolved significantly since their appearance. The newest version, YOLOv10, will raise the power from its predecessors to even more performance and versatility. Joseph Redmon et al. developed the first model for YOLO. It introduced a single unified architecture processing full images in pass-through,

very fast (Redmon, Divvala et al., 2016). Batch Normalisation, Anchor Boxes, and Dimension Clusters improve YOLOv2 (Redmon and Farhadi, 2017). It was succeeded by YOLOv3, which again pushed model performance by an enhanced and more lightweight backbone network with multiple anchors (Redmon and Farhadi, 2018). Then, mosaic data augmentation and a new anchor-free detection head were introduced in YOLOv4 (Bochkovskiy, C.-Y. Wang and Liao, 2020). Then, YOLOv5 from Ultralytics had a couple of interesting features, one of which was hyperparameter optimisation with integrated experiment tracking (Jocher, 2020). Thereafter, Li et al. published YOLOv6, which was optimised for autonomous delivery robots (C. Li et al., 2022). More enhancements were added to YOLOv7, and extra tasks like human pose estimation on COCO keypoints (C.-Y. Wang, Bochkovskiy and Liao, 2023). YOLOv8 is a highly versatile and efficient approach, ranging from detection to segmentation with human pose estimation and tracking, and classification (Jocher, Chaurasia and Qiu, 2023). Later, YOLOv9 proposed the new programmable gradient information concept along with a new lightweight network architecture, GELAN, to further hugely improve the use of parameters and performance on MS COCO (C.-Y. Wang, Yeh and Liao, 2024). Finally, to this end, YOLOv10 went ahead in efficiency by innovating consistent dual assignments for NMS-free training, with holistic efficiency-accuracy-driven model design strategies to achieve SOTA performance and efficiency across various model scales (A. Wang et al., 2024).

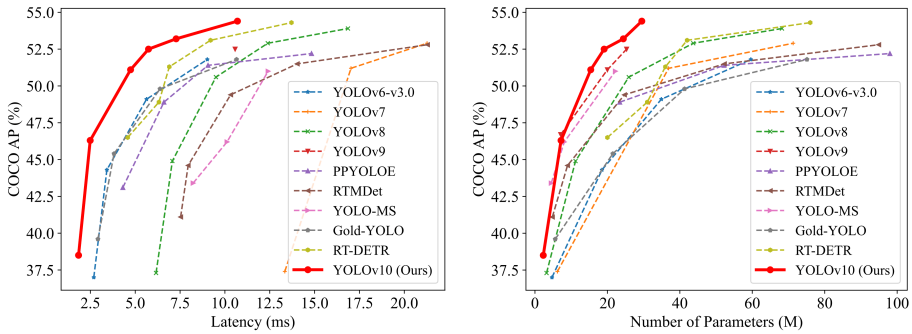

Figure 2.10: Comparision of various models including the YOLO models v6 to v10 (https://docs.ultralytics.com/models/yolov10/)

2.5.2 Mediapipe by Google

MediaPipe is an open-source framework developed by Google that enables cross-platform, customisable ML across live and streaming media. It is designed to be highly flexible and to be easily integrated into a range of applications. MediaPipe provides the most complete solution for pose estimation, which is realised through a two-stage detector: first, ROI detection containing the person, then keypoint prediction inside. This methodology assists in accurate estimation of pose in real-time (Lugaresi et al., 2019). MediaPipe is very flexible; it has developed models and pipelines that can be easily customised for a developer's requirements. It supports a great variety of platforms, including Android, iOS, web, and desktop, hence making it quite accessible for various applications (Google, 2023). MediaPipe has been used in a host of different applications, from fitness tracking to augmented reality. On the other hand, it is a popular choice among many developers who want to implement pose estimation and several other computer vision tasks because of the capability to handle a myriad of modalities, like video or audio, and the ease of integration (Lugaresi et al., 2019). Recent improvements to MediaPipe include hand-gesture recognition by improving the framework for 3D keypoint estimation more accurately and adding support for neural network-based gesture classifiers (Sung et al., 2021).

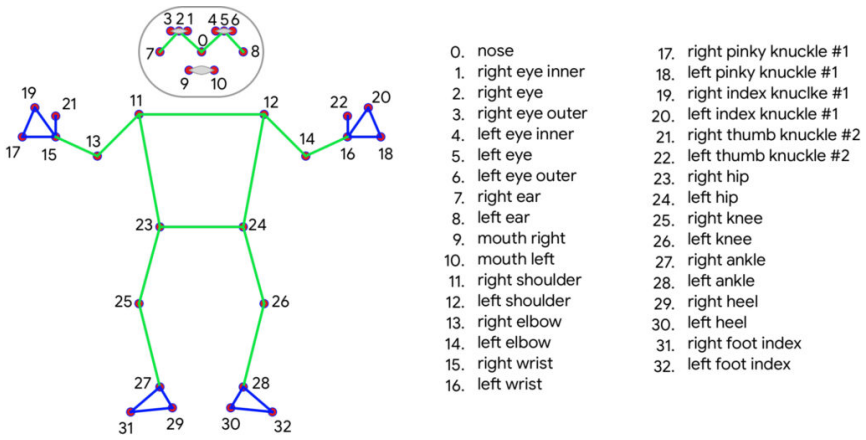

Figure 2.11: The body keypoints used by mediapipe

2.5.3 Comparison

Let's compare these two solutions:

1. **Speed and Performance**

   YOLO:

   - **Real-time Processing**: Models in YOLO are very fast and process natural images in real time. This makes them very appropriate to any application including immediate feedback, for example, live video analysis and autonomous systems.
   - **Unified Architecture**: YOLO's single-pass architecture allows it to detect objects and poses in one go, significantly reducing latency compared to multi-stage detectors.

   MediaPipe:

   - **Two-Stage Detection**: MediaPipe uses a two-stage approach: first, it detects the ROI and then predicts keypoints within the ROI. Compared with YOLO's single pass, this is slightly slower but often more accurate for pose estimation.
   - **Optimised for Mobile and Web**: MediaPipe is designed to run efficiently on various platforms, including mobile devices and web browsers, ensuring smooth performance even on less powerful hardware.

2. **Flexibility and Customisation**

   YOLO:

   - **Versatility**: In addition to position estimation, YOLO models—especially the most recent iterations, such as YOLOv8 and YOLOv10—support a variety of tasks, such as object detection, segmentation, and tracking. Because of its adaptability, YOLO is an effective tool for extensive AI vision applications.
   - **Customisation**: YOLO allows for extensive customisation through hyperparameter tuning and integration with various frameworks, enabling developers to optimise the model for specific use cases.

   MediaPipe:

- **Modular Design**: MediaPipe's modular architecture makes pipeline customisation and extensibility very convenient for developers to make precisely whatever application may be needed—be it a fitness tracker or an augmented reality application.
- **Cross-Platform Support**: MediaPipe's ability to run on Android, iOS, web, and desktop platforms makes it highly adaptable for different deployment environments.

3. **Accuracy and Robustness**

   YOLO:

   - **High Accuracy**: Accuracies of YOLO models have increased with each successive edition. For example, YOLOv10 introduces state-of-the-art techniques, such as consistent dual assignments and holistic efficiency-accuracy-driven design, to achieve state-of-the-art performance.
   - **Robustness**: YOLO's robust architecture and extensive training on diverse datasets ensure reliable performance across various scenarios, from crowded scenes to low-light conditions.

   MediaPipe:

   - **Precision**: The two-stage strategy of detection is very often accurate in MediaPipe pose estimation; therefore, it's applicable in various domains, from medical diagnosis to sports analysis.
   - **Multi-Modal Capabilities**: MediaPipe's support for multiple modalities (e.g., video, audio) enhances its robustness in handling complex tasks that require integrating different types of data.

4. **Ease of Integration**

   YOLO:

   - **Integration with AI Frameworks**: YOLO models can be easily integrated with popular AI frameworks like TensorFlow, PyTorch, and ONNX, facilitating seamless deployment in various environments.

- **Community and Support**: The extensive community support and comprehensive documentation available for YOLO models make it easier for developers to implement and troubleshoot.

MediaPipe:

- **Ease of Use**: MediaPipe also provides out-of-the-box solutions and configurable pipelines, making integration easy. Also, with its intuitive API and broad reach of documentation, it empowers developers who have minimal experience in computer vision.
- **Google Ecosystem**: As part of the Google ecosystem, MediaPipe benefits from continuous updates and improvements, ensuring compatibility with the latest technologies and platforms.

5. **Use Cases**

YOLO:

- **Real-Time Applications**: Ideal for applications requiring real-time processing, such as autonomous driving, surveillance, and live sports analysis.
- **Comprehensive Vision Tasks**: Suitable for projects that need a versatile model capable of handling multiple vision tasks simultaneously.

MediaPipe:

- **Custom Solutions**: Perfect for applications that require tailored solutions, such as fitness apps, augmented reality, and interactive media.
- **Cross-Platform Deployment**: Best suited for projects that need to run on various platforms, including mobile devices and web browsers.

In summary, while both YOLO and MediaPipe provide very powerful tools for tasks in the estimation of human pose, the former comes short in flexibility, ease of integration, and cross-platform support. It is perfect for those applications that require bespoke solutions with seamless deployment across all platforms, thanks to its modular design and capability for handling multiple modalities. Regarding the project in this paper, MediaPipe seems the better choice.

2.6 The Effect of Exercise on Mood

It has been known for decades that one's mental health can be improved by incorporating exercise into their lifestyle. The papers titled "The relation of physical activity and exercise to mental health." (Taylor, Sallis and Needle, 1985) and "Psychological effects of habitual aerobic exercise: A critical review" (Hughes, 1984) were early studies on this phenomenon and to what extent physical activity affects mental health. By reviewing controlled experiments on the effects of habitual aerobic exercise on mood, personality, and cognition, Tayor et al. found that the strongest evidence suggests that exercise will most likely alleviate symptoms of depression, alcoholism, anxiety, and schizophrenia, as well as improve self-image and social skills.

This more recent review titled "Physical exercise and mental health: The routes of a reciprocal relation" (Fossati et al., 2021) looks at the physical-mental link in the other direction, looking to find whether this relationship is reciprocal.

Finally, the study "Effects of a bout of exercise on mood in people with depression with and without physical pain" (Caviness et al., 2023) investigates the immediate effect of aerobic exercise on a community sample of 147 participants with above-average depressive symptoms by recording their mood before and after the activity. They found a statistically significant change between pre and post-workout mood. The question for this paper is: does this mood change still happen if the exercise is not aerobic, such as resistance training and callisthenics?

2.7 Summary

Object detection, segmentation, human pose estimation, and motion tracking have all benefited from corresponding development from traditional methods up to modern deep learning frameworks. Classic techniques in the line of HOG, Haar cascades, thresholding, edge detection, and Kalman filters did much to lay the ground. Deep learning has reshaped these with R-CNN, U-Net, Convolutional Pose Machines, and LSTM networks, attaining more accuracy and efficiency. State-of-the-art solutions,

such as YOLO and MediaPipe, are real-time, versatile, and cross-platform, really stretching the bounds of what could be done within computer vision.

2.7.1 Comparison of the Current Solutions

There are solutions to computer vision problems in terms of speed, accuracy, and suitability of applications. Most of the traditional methods are normally simple, less computationally intensive, and poor in terms of accuracy and robustness against deep approaches. Deep learning methods, like Faster R-CNN for detection and U-Net for segmentation, will rather be at the high end in terms of accuracy but at an expensive computational cost. On the other hand, YOLO models would give a compromise in speed and accuracy that is of relevance in real-time applications. MediaPipe offers flexibility and ease of integration across platforms on the other side.

2.7.2 Research Gaps

While the achievements that have been made so far in this area are very inspiring, there are still some gaps in the research. The first gaps, even though this is an area which is improving, are efficient models that can run on low-power devices without a loss of accuracy. Handling of occlusions and light variation is another challenge that is currently a goal for researchers. Better generalisation across different datasets and environments, and more robust methods for real-time multi-object tracking and pose estimation in crowded scenes, are required. Addressing these gaps will be highly instrumental in taking further applications of computer vision forward.

2.8 Aims & Objectives

Based on the previous research described in the literature review above, I can know express my aims and objectives for this project. Here are the SMART goals I have set for this endeavor:

1. **Specific**: The goal of creating an AI fitness trainer application that will set

the groundwork for studying the effect of exercise on mood. It involves these tasks:

- **User Detection**: Using video input to detect the presence of the user ensures the application can start analysing only when a user is present.
- **Keypoint Identification**: Identifying keypoints on the user's body is crucial for understanding body posture and movement.
- **Pose Analysis**: Analysing the poses created by the connections between keypoints allows the application to understand the user's form and make necessary inferences about occluded keypoints.
- **Form Recommendations**: Providing recommendations to improve the user's form help in minimising injury risks and maximising exercise effectiveness.
- **Mood Tracking**: Collecting the mood of the user before and after the exercise session.
- **Potential Add-on**: Developing a smartphone application enhances accessibility and user convenience.

2. **Measurable**: The success of the application can be qualitatively measured through:
   - **Detection Accuracy**: The ability to accurately detect the human shape and keypoints. This is quantitatively measurable through metrics such as balanced accuracy, f1 score, precision, recall...
   - **User Feedback**: Collecting user feedback on the effectiveness of the recommendations provided by the application. This is a more qualitative measure as the users' feedback will take the form of reviews. The users can be asked to give scores for usability and perceived effectivness.
3. **Achievable**: This project is achievable due to several factors:
   - **Experience**: I have previous experience with image processing using Python providing a solid foundation for developing this application. As

during my MComp degree, I worked on a project that detected and tracked seabirds on timelapse images using YOLOv5, tracking the birds from egg to adult stage.

- **Existing Technologies**: By leveraging existing libraries and frameworks for computer vision (e.g. OpenCV, MediaPipe) can accelerate development.
- **Incremental Development**: By breaking down the project into smaller, manageable tasks ensures a steady progress and allows for iterative improvements.

4. **Relevant**: The project is highly relevant for several reasons:
   - **Health and Fitness**: With the growing emphasis on physical and mental health and fitness, an AI personal trainer can provide valuable assistance to users looking to improve their exercise routines.
   - **Injury Prevention**: Proper form is crucial in preventing injuries during exercise, and this application can play a significant role in educating users.
   - **Technological Advancement**: This project contributes to advancements in AI and computer vision, showcasing practical applications of these technologies. This also creates a framework on which future research on the effect exercise as on mental health.
5. **Timebound**: The project has a clear deadline of August 29th, 2024, providing a two-month timeframe. This is sufficient for:
   - **Initial Development**: Completing the core functionalities of user detection, keypoint identification, and pose analysis.
   - **Testing and Refinement**: Conducting thorough testing and refining the application based on feedback.

Chapter 3

Methodology

3.1 Research Design

The design of the project follows a qualitative research approach that focuses majorly on user feedback as a way to assess whether or not the effectiveness of the computer-vision-based training coach is at its best. This was done so as to derive specific information touching on the user experience and satisfaction derived from the system by its users.

3.1.1 Qualitative Methods

The qualitative methods provided subjective data regarding the user's experience and satisfaction. In this case, the system users were subjected to a semi-structured interview so as to gather feedback on usability, intuitiveness, and overall user experience, The data was important in the identification of weak point and thus assuring the usability and meeting of the needs expected of the system.

3.1.2 Interviews

It included semi-structured, in-depth interviews taken with a set of subjects, both novice and experienced users, On its part, it allowed for open-ended responses and gave rich qualitative data about the experiences of the participants.

3.1.3 Rational for Qualitative Approach

The qualitative approach was chosen for eliciting in-depth understanding of the user experience. This could be done quantitatively, which would probably be more objective in regard to data about performance of the system, but it would miss fine details about user satisfaction and usability. Interviews were used to gather qualitative data that contributed useful information for the improvement of the system.

3.1.4 Algorithm Validation

While the focus of this study is on qualitative assessment, it should not go unmentioned that many research studies have already been done regarding the quantitative testing and validation of the Mediapipe algorithm used for human pose detection. This has established the ground for the accuracy and reliability of the algorithm previously to work in different contexts, thus forming a basis for application in this project. This allows for focusing on a review of the user experience, leaving out the technical accuracy matters at the core because of the use of an algorithm with a good performance history.

3.2 System Architecture

Basically, the system architecture of the computer-vision based training coach has three parts: the applications, the GUI, and the database. The modules are very key to attaining the objectives of the study, for they play a role in the correct detection of human pose, user-friendly interaction, and efficiency in managing data.

3.2.1 Applications

The applications incorporate core system functionality, human pose detection, and infer relevant information like joint angles and repetition counts. Each exercise type (e.g. bicep curls) has its own application script. This component is implemented using advanced computer vision algorithms and machine learning models.

Human Pose Detection

Keypoint detection of the human body is done through the framework MediaPipe, providing strong, real-time human body landmark tracking. It captures video input from a camera, processing frames for the detection of key points on the human body and tracking these points to analyse movements. The application will look only for those keypoints relevant for the individual exercise to optimise the speed of the detections.

Making the Inferences

These inferences are made by processing the detected key points toward calculating joint angles, counting repetitions, and assessing the correctness of exercises performed by the user. It means applying geometric and kinematic principles against detected key points for meaningful metric derivations. The results formed the basis for the user to receive feedback through the GUI.

3.2.2 Graphical User Interface (GUI)

This GUI component intuitively interacts with the system in a very user-friendly way. It provides real-time feedback, visualises the detected pose, and shows information about the performance of the user.

Design and Layout

The GUI is designed to include both the usability and aesthetic appeal of the system. It will display live video feed, overlay of detected key points, performance metrics, and instruction prompts, among other relevant components. All this has been laid out in a manner that clearly tells a user what is happening and how to interact with a system.

User Interaction

The GUI provides the user with the ability to start and stop a session, view performance metrics, and obtain real-time feedback. Interactive components include

buttons, sliders, and textboxes. Feedback provided by the GUI can be used by the users in self-correction of movement to perform better.

3.2.3 Database

This component is responsible for storing all information about the users, sessions, and workouts. The database component organises this data in a way that it is secure and can be retrieved with much ease for analysis and reporting.

Data Storage

There are three tables: users, sessions, and workouts. The workouts table is related to the sessions table through a foreign key on a many-to-one basis. Correspondingly, the sessions table is also connected to the users table via a foreign key on a many-to-one basis. This relational structure will ensure efficient organisation and retrieval of data.

Data Management

Data Management provides integrity, security, and privacy of data stored. The users' passwords are encrypted in order to protect sensitive information from hackers. The information is placed in a relational database management system. This offers robust integrity and consistency to the data.

3.2.4 System Integration

The interplay between these three major components of the system is such that it seamlessly provides a cohesive and efficient training experience. Applications digitise and process video data, GUI provisions make real-time feedback and interaction, and database provisions make arrangements for managing and storing all relevant data. This integration will assure that the system works smoothly and effectively to attain the objectives of the project.

3.3 Algorithm selection

The final algorithm to be deployed would have to be driven by considerations of accuracy, real-time performance, and the possibility of future scaling on mobile platforms. With such factors in careful consideration, Mediapipe was selected over YOLO for human pose estimation due to its appropriateness to the task at hand and related objectives and development plans.

3.3.1 Mediapipe

Mediapipe is a framework developed by Google that makes available robust and real-time body landmark tracking. The reason it was picked is that it could capture intricate movements in fine detail, necessary in analysing exercises. Due to its high performance in real-time and ease of integration with other tools, Mediapipe was chosen as the best for this project.

Advantages of Mediapipe

Mediapipe offers several advantages that align with the project's goals:

- **Real-Time Performance**: Mediapipe is optimised for real-time performance, ensuring that the system can provide immediate feedback to users during their training sessions.
- **Accuracy**: The framework has been extensively tested and validated, demonstrating high accuracy in detecting and tracking human body landmarks.
- **Cross-Platform Compatibility**: Mediapipe is designed to be cross-platform, making it possible to develop applications that can run on various devices, including smartphones. This aligns with the project's future goal of creating a mobile version of the training coach.
- **Ease of Integration**: Mediapipe can be easily integrated with other libraries and tools, such as OpenCV, facilitating the development process.

3.3.2 YOLO (You Only Look Once)

YOLO is a state-of-the-art object detection algorithm known for its speed and accuracy. While YOLO is highly effective for object detection tasks, it was not chosen for this project due to a couple reasons:

- **Specialisation**: YOLO is primarily designed for object detection rather than detailed human pose estimation. While it can detect human figures, it does not provide the same level of detail in tracking body landmarks as Mediapipe.
- **Resource Requirements**: YOLO's larger models' computational requirements are higher compared to Mediapipe, making it less suitable for real-time applications on resource-constrained devices such as smartphones. YOLO's smaller models are not as accurate as would be desired for this project.

3.3.3 Rationale for Algorithm Selection

Certain requirements in the project, considering the goals of further scaling, conditioned the choice of Mediapipe instead of YOLO. The fact that Mediapipe provides accurate human pose detection in real time, with cross-platform compatibility, made it just the right solution for this project. By using Mediapipe, the system will be able to produce high-quality performance both on desktop and mobile platforms, hence wide applicability and accessibility to users.

3.3.4 Future Scalability

Another major reason for using Mediapipe was that this would allow easy scaling in the future. A sister program of similar functionality is to be created running on smartphones. This will, in turn, provide users with a portable and convenient training coach. Mediapipe's efficient performance and compatibility with mobile platforms make it fit for this future development. The future-facing attribute within this approach means that, consequently, the evolution and expansion of the project are guaranteed to reach a broader audience, offer greater flexibility in how users engage with the training coach.

3.4 Development Environment

3.4.1 Software and Tools

Python was selected as the primary language of programming because of its variety of libraries and availability. Also, its flexibility and wide community support made it suitable for developing computer vision algorithms and for gluing a lot of different parts of the system together. The IDE I decided to use for this project was Pycharm as this project is relatively large and Pycharm provides many quality of life features that are not necessarily available in the other code editors I usually use such as VSCode and neovim. Pycharm also has database management tools that allow for easy implementation and testing.

Libraries

Several libraries were utilised to implement the system:

- **tkinter**: This library was used for designing the GUI.
- **OpenCV**: OpenCV was used for image and video processing tasks. Its comprehensive set of tools and functions made it easy to preprocess and analyse video data.
- **Mediapipe**: Mediapipe was used for body tracking. Its pre-built models and easy integration with other libraries made it a valuable tool for this project.
- **psycopg2**: This library provides a link between python and postgresql for easy database manipulation from python.

3.4.2 Version Control

For the version control I used git. Git is a distributed version control system developed to deal with small to very large projects. It tracks every single modification done in the code. Changes can be reverted easily to their previous form in case something goes wrong. By pushing your code to a remote repository (e.g. Github), you're basically saving your work and it can be accessed from anywhere, allowing you

to use/test the code on multiple machines. You can create branches where you can work on new features or experiments without touching the main codebase. When satisfied with the changes, you can merge into the main branch. Commit messages and logs document what changed and why, which is incredibly useful to understand how your project evolved.

3.5 Ethical Considerations

Ethical considerations were paramount in this project, ensuring that the research was conducted responsibly and ethically.

There are a few considerations that are necessary to make when dealing with any AI program. The first is about bias and fairness, it is important to ensure that the program doesn't unintentionally discriminate against certain groups. Another consideration is about privacy and data protection, if any data is collected, it must be in compliance with privacy regulations such as GDPR. In the case of this program, it must ensure that it does not put the user in any danger whatsoever, this includes not advising the user to perform dangerous movements that could lead to injury. The final consideration is about the impact this program may have on employment, as this program may disrupt the personal trainer job.

3.6 Risk Analysis

Risk analysis was conducted to identify potential risks associated with the project and to develop strategies for mitigating these risks.

- **Technical Risks**:
  - **Model Accuracy**: The vision model performance inside the computer may not be up to standard, thus giving incorrect exercise recommendations. Mitigation strategies then include thorough testing and continuous model evaluation, with incorporation of user feedback in order to improve the model. This is unlikely to happen as the mediapipe model has been

extensively tested, but the impact would be high as the program is dependant on its accuracy.

  - **System Reliability**: Technical failures can project from software bugs or hardware malfunction and affect the running of any system. This risk can be minimised by rigorous testing programs and backup systems. This is a likely issue as I am the sole developper, and the level of imapct would depend on the severity of the bug.

- **Data Privacy Risks**:

  - **Data Breaches**: There is the risk that unauthorised access to user data may result in privacy breaching. In a bid to protect against this, very strong encryption, secure data storage, and frequent security auditing ought to be implemented. This is unlikely for now as there is minimal data stored and it is stored locally, and the impact would be low aswell for now as there is no centralised server.

  - **Data Misuse**: Unless properly managed, collected data can be misused. This requires that data governance policies are clearly outlined and access controls be strict to avoid its misuse. For the same reasons as for Data Breaches, this is low likelihood and low impact for now.

- **User Risks**:

  - **Physical Harm**: Wrong exercise recommendations may lead to physical harm. Therefore, it is important that the system ensures safe and relevant exercises in view of the level of health conditions and fitness of the users. It may incorporate disclaimers and encourage users to consult with a professional in healthcare. The impact this may have could be extremely severe, therefore it is of paramount importance that the likelihood is reduced to a minimum.

  - **User Compliance**: Compliance by the users themselves in performing the recommended exercises may not be proper. Therefore, the effectiveness of the training program could be reduced. Clear instructions, visual

aids, and regular feedback can improve user compliance. A lack of user compliance would have a strong impact on the accuracy of the model and recommendations, so as with the risk of physical harm, the mitigation strategies must be followed to reduce the likelihood.

- **Ethical Risks**:
  - **Bias and Fairness**: It can also be the case that there are biases inherent in the pre-trained model itself and hence give unfair recommendations. Regular evaluation with respect to bias and the respective refinement can be very instrumental in making the model ensure fairness. As with the Model Accuracy, Google has performed the necessary steps to reduce this risk, the impact would be high as the model wouldn't work well on certain groups but the likelihood is low.
  - **Transparency**: The system may also be liable to suffer from a problem of mistrust if it is less than transparent in its recommendations. That is a risk, which can be countered by clear explanations and maintaining transparency in its operation. Mistrust in the program would discourage people from using it, therefore, it would have a high impact. The likelihood of this happening would be moderate.
- **Operational Risks**:
  - **Scalability**: In case of increase in number of users, there can easily be overload in the system. Making the system scalable at the first instance and eventually moving to a cloud solution will help to deal with the bigger loads. It's worth noting that moving to the cloud would increase the risk to the data integrity. The measures required to protect personal information in the cloud are much more strict. This has a moderate chance of happening, but would have a high impact.
  - **Resource Allocation**: The project may be impacted by low levels of resources, such as time and budget. The entailed risks can be reduced by well-managing the project and ensuring the most important tasks are

prioritised. This also has a moderate chance of happening, but would have a high impact.

Chapter 4

Implementation

4.1 System Requirements

4.1.1 Software Requirements

This project is designed to function on all major operating systems. The bulk of the project was programmed in a Ubuntu 22.04 environment but it has been tested on an Apple macbook and a windows laptop. To run the python scripts, there are a few libraries that are required:

- mediapipe
- openCV
- numpy
- pillow
- tkinter (normally included in python by default but not on macOS)
- psycopg2

The user data is stored locally using postgresql for the database management. This requires postgresql to be installed on the machine.

4.1.2 Hardware Requirements

The goal of this project was to lay the foundation for a software program that could run on a smartphone or other portable device, hence the software has relatively low

hardware requirements. A camera that can parse a video signal is essential. The machines used to develop the program are:

- A Linux PC with an RTX 4070 GPU and i5-13400k CPU
- A Windows laptop with a GTX 1070 GPU and i7... CPU
- A Macbook Air M1 2020

Even though these machines are fairly powerful, I expect the program to be able to run on far less powerful systems.

4.2 App Creation

The apps where created using Google's Mediapipe framework. Several criteria: the application must be able to detect the user's pose, the application must show the user the relevant joints and connections to the exercise, the application must be able to calculate the relevant angles within the user's pose, the application must be able to count the number of repetitions of the exercise the user has made, and the application must be able to be integrated into a python created GUI.

```
def make_detections(app, frame):
    img = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)
    img = cv2.flip(img, 1)
    img.flags.writeable = False
    results = app.pose.process(img)
    img.flags.writeable = True
    img = cv2.cvtColor(img, cv2.COLOR_RGB2BGR)

    return results, img
```

Figure 4.1: Code Snippet of the make_detections method in src/workouts/utils.py.

The implementation of mediapipe detections can be seen in Figure 4.1. This snippet shows the raw camera input being preprocessed for the mediapipe pose processing, first the colour format is converted from BGR (openCV format) to RGB (mediapipe format), then flipped horizontally, and finally the frame's flags are set to be immut-

able. Once the processing of the frame is done, the flags are set to be once again mutable and the colour format is returned to BGR.

The angle of the relevant joints to the exercise are calculated using the function seen in Figure 4.2. This function takes in three 2D points a, b, and c; and returns the angle between ab and cb in degrees.

```
def calculate_angle(a, b, c):
    a = np.array(a)
    b = np.array(b)
    c = np.array(c)

    radians = np.arctan2(c[1] - b[1], c[0] - b[0]) - np.arctan2(a[1] - b[1], a[0] - b[0])
    angle = np.abs(radians * 180.0 / np.pi)

    if angle > 180.0:
        angle = 360 - angle

    return round(angle, 2)
```

Figure 4.2: Code Snippet of the calculate_angle method in src/workouts/utils.py.

Next is the repetition counting functionality which is shown in Figure 4.3. This function is made to be generalise for it to be usable for every exercise type. It takes in as parameters the application that is using it, the current angle at the joint, the side (left or right), the low threshold the angle needs to reach for the repetition to count, the high threshold the angle needs to reach for the repetition to count, and the stage at which the muscle is at tension (or the stage that is harder to reach due to the mass of the weight). First it recovers the current count and the current rep stage, then it detects whether there has been a change in the rep stage. The rep stage will have changed from up to down if the current stage is up and the angle is above the high threshold and it will have changed from down to up if the current stage is down and the angle is below the low threshold. Then is updates the counter for the correct side.

Finally the culmination of these feature can be seen in Figure 4.4

```python
def rep_counter(app, angle, side, min_angle, max_angle, tension_stage="up"):
    if side != "left" and side != "right":
        raise ValueError("Side must be 'left' or 'right'")

    # Get the current count
    count = app.rep_count_l.get() if side == "left" else app.rep_count_r.get()
    rep_stage = app.rep_stage_l if side == "left" else app.rep_stage_r

    if angle > max_angle and rep_stage == "up":
        rep_stage = "down"
        count += 1 if tension_stage == 'down' else 0  # Increment the count
        print(f'down {count}')
    elif angle < min_angle and rep_stage == "down":
        rep_stage = "up"
        count += 1 if tension_stage == 'up' else 0  # Increment the count
        print(f'up {count}')

    # Update the IntVar with the new count
    if side == "left":
        app.rep_count_l.set(count)
        app.rep_stage_l = rep_stage
    else:
        app.rep_count_r.set(count)
        app.rep_stage_r = rep_stage
```

Figure 4.3: Code Snippet of the rep_counter method in src/workouts/utils.py

Figure 4.4: Screenshots from the development of the bicep curl app. It shows mediapipe's keypoint detections on the arms and displays the angle between the shoulder-elbow segment and the wrist-elbow segment.

4.3 GUI Creation

4.3.1 Frontend Development

The frontend was developed using the tkinter library that is prepackaged into python. The GUI is organised like a tree system where each page is a node within the tree. To get to a page you must travel the GUI tree from the root node (home page) through the branches that lead to the desired page. Each branch can be traversed in both directions and the program can be exited a any moment.
Here is a list of some of the pages:

- **home.py**: the root node containing links to register, login, and login as guest. A screen capture of the home page can be seen below in Figure 4.5

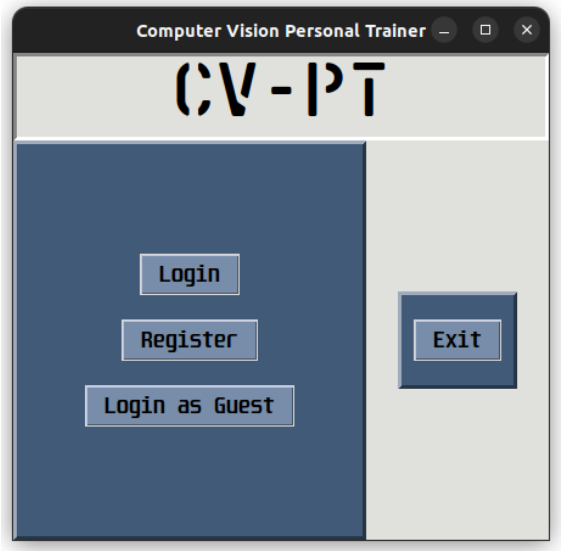

Figure 4.5: Screen capture of the home page. (4.3.1)

- **register.py**: contains a registration form that will save the users username and password to the database. If successful, it will send the user to menu.py.
- **login.py**: contains a login form that checks if the username exists and the (hashed) password matches. If successful, it will send the user to menu.py. Screen captures of the login and register pages can be seen below in Figure 4.6

Figure 4.6: Screen captures of the register (left 4.3.1) and login (right 4.3.1) pages.

- **menu.py**: contains a grid of buttons that link to each general muscle group/exercise type (arms, back, cardio, chest, legs, shoulders). If the user is logged in, there is also a button that links to the user's workout history. Screen captures of the menu pages with and without the history button can be seen below in Figure 4.7

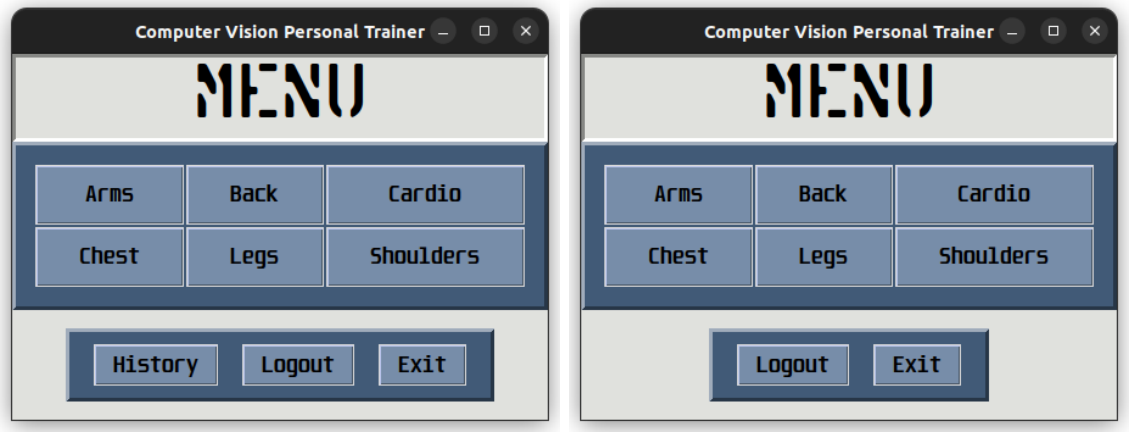

Figure 4.7: Screen captures of the menu page with (right) and without (left) a user logged in. (4.3.1)

- **arms.py**: contains a list of buttons linking to specific exercises (bicep curls and tricep pushdowns). The other child nodes of menu.py are the same where they are a list of the exercises relevant to that muscle group. A screen capture of the arms page can be seen below in Figure 4.8

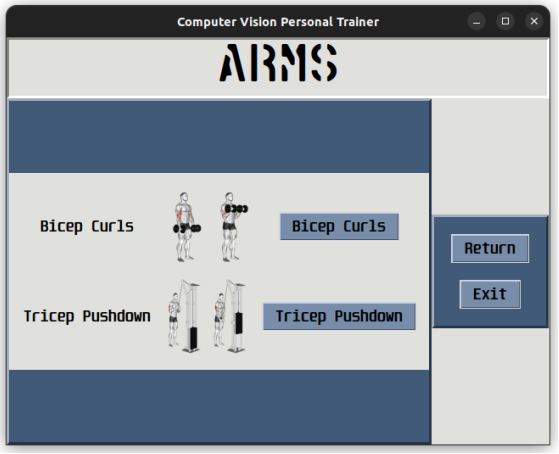

Figure 4.8: Screen capture of the arms page. (4.3.1)

- **bicep_curls.py**: These exercise pages are arranged to have a camera feed on the left half of the window and parameters and information on the right half. The camera feed will show the user what they are doing and act like the mirror in the gym, useful for visualising their form. Aswell as displaying the mediapipe skeleton on top of the joints that are relevant to the specific exercise. The right half has input areas where the user can enter the weight they are using, the amount of rest time they wish to have between sets, and depending on the exercise, which side of the body they are currently training. There is also a textbox that will display information at the top centre of the right half. A screen capture of the bicep curls app can be seen below in Figure 4.9

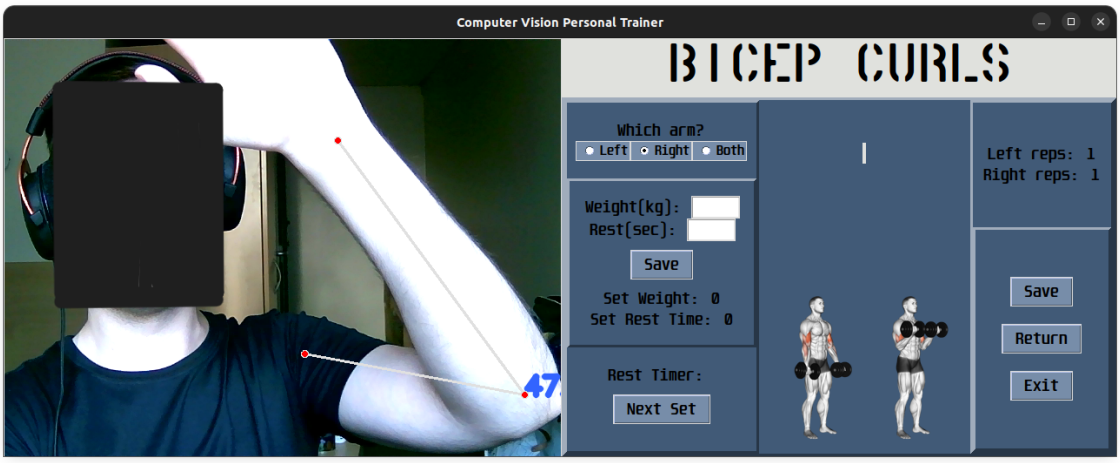

Figure 4.9: Screen capture of the bicep curls page. (4.3.1)

Every time a user logs in or out, a popup appears asking them to rate their current mood on a scale of 1 to 10. This popup can be seen in Figure 4.10

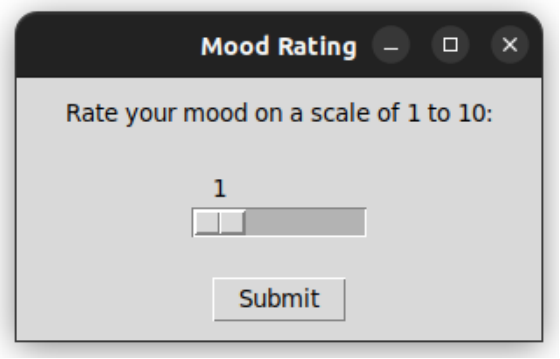

Figure 4.10: Screen capture of the mood query popup. (4.3.1)

4.3.2 Backend Development

For the implementation of the backend I used the database management system called PostgreSQL. PostgreSQL is a relational database system written in the C programming initially released in 1996 that is free and open source. It functions also on Windows, MacOS, and Linux. As python language is also written in C, it has a strong library for database manipulation called pyscopg2.

Database tables

The schema contains three tables:

1. **users**: (see Figure 4.13) the users table contains five columns:
    - id (uuid, unique, primary key, not-null): unique identifier.
    - username (varchar, unique, not-null): unique username.
    - password (varchar, not-null): hashed password.
    - height (int): height of the user in centimeters, optional.
    - weight (int): weight of the user in kilograms, optional.

    Each row refers to an individual user.

```sql
CREATE TABLE users (
    id uuid PRIMARY KEY NOT NULL UNIQUE,
    username VARCHAR(255) NOT NULL UNIQUE,
    password VARCHAR(255) NOT NULL,
    height INT,
    weight INT
);
```

Figure 4.11: Code Snippet of the SQL to create the users table

2. **sessions**: (see Figure 4.12)the sessions table contains seven columns:
    - id (uuid, unique, primary key, not-null): unique identifier.
    - username (varchar, foreign key, not-null): references the username of the user in the session.
    - created_at (timestamp, not-null): timestamp at which the session started.
    - duration (interval): time interval between created_at and the end of the session. Can be null because the row is created before the end of the session.
    - volume (int): the total weight moved by the user during the session.
    - start_mood (int): the mood of the user at the start of the session on a scale of 1 to 10.
    - end_mood (int): the mood of the user at the end of the session on a scale of 1 to 10.

    Each row refers to an individual session.

3. **workouts**: (see Figure **??**) the workouts table contains eight columns:
    - id (uuid, unique, primary key, not-null): unique identifier.
    - session_id (uuid, foreign key, not-null): references the id of the session the workout belongs to.

```
CREATE TABLE sessions (
    id uuid PRIMARY KEY NOT NULL UNIQUE,
    username VARCHAR(255) NOT NULL REFERENCES users(username),
    created_at TIMESTAMP NOT NULL DEFAULT NOW(),
    duration INTERVAL,
    volume INT,
    start_mood INT,
    end_mood INT
);
```

Figure 4.12: Code Snippet of the SQL to create the users table

- name (varchar, not-null): the name of the excerce being performed by the user (e.g. bicep curls)
- created_at (timestamp, not-null): timestamp at which the workout started.
- duration (interval): time interval between created_at and the end of the workout. Can be null because the row is created before the end of the workout.
- reps (int): number of repetitions the user performed during the workout.
- max_weight (int): the weight in kilograms lifted during the heaviest set during the workout.
- volume (int): the sum of the number of reps times the weight of the set for each set performed during the workout.

Each row refers to an individual workout.

Another functionality that is provided by postgresql is the ability to create functions, also known as routines, to preform multiple SQL queries and actions.

Register functionality

The register system (see Figure 4.14 functions as follows: first there is a equality check between the password and the confirm password entries inputted by the user.

```
CREATE TABLE workouts (
    id uuid PRIMARY KEY NOT NULL UNIQUE,
    session_id uuid NOT NULL REFERENCES sessions(id),
    name VARCHAR(255) NOT NULL,
    created_at TIMESTAMP NOT NULL DEFAULT NOW(),
    duration INTERVAL,
    reps INT,
    max_weight INT,
    volume INT
);
```

Figure 4.13: Code Snippet of the SQL to create the workouts table

If they match, then a connection to the database is made with DBConnection (see Figure 4.16).

Then, the password is hashed using the sha256 hashing function, and the users table is scanned for the username. If the username already exists, the process is stopped and the user is notified though a messagebox that their chosen username is already taken. Else, if the username is not taken, a new user is added to the users table with a universally unique identifier (uuid), the username, and the hashed password. The user is then automatically logged in (the login process is described next).

Login functionality

The login system (see Figure 4.15) functions as follows: first a connection to the database is formed using DBConnection (see Figure 4.16).

Following that, the password entered by the user is hashed using the same hashing algorithm used for the encryption of the passwords and the database is searched for a user who has both matching username and hashed password. If a user is found, a new session is created, with a uuid which acts as the session token. This token is both stored in the database and also a newly created text file that only exists if a user is logged in.

Finally, the menu GUI page is opened. If there are no matching users, a messagebox appears notifying the user that either the username or password is invalid and the connection to the database is closed. Which one of the username and password being

```python
def register_user(self):
    if self.password_entry.get() == self.c_password_entry.get():
        from src.db.db_connection import DBConnection
        db = DBConnection()
        conn = db.connect()
        cursor = conn.cursor()

        username = self.username_entry.get()
        password = self.password_entry.get()

        # Hash the password
        hashed_password = hashlib.sha256(password.encode()).hexdigest()

        # check if username already exists
        query = "SELECT * from cv_pt.public.check_user(%s)"
        cursor.execute(query, (username,))
        if cursor.fetchone()[0]:
            messagebox.showerror("Error", "Username already exists")
            db.close()
            return

        user_id = str(uuid.uuid4())
        query = "SELECT * from cv_pt.public.create_user(%s, %s, %s)"
        cursor.execute(query, (user_id, username, hashed_password))
        conn.commit()
        messagebox.showinfo("Success", "User registered successfully")

        from GUI.workouts.utils import login, get_mood

        mood = get_mood()
        self.destroy()
        login(username, password, mood)

    else:
        messagebox.showerror("Error", "Passwords do not match")
```

Figure 4.14: Code Snippet of the register function

```python
def login(username, password, mood_rating):
    from src.db.db_connection import DBConnection
    db = DBConnection()
    conn = db.connect()
    cursor = conn.cursor()

    # Hash the entered password
    hashed_password = hashlib.sha256(password.encode()).hexdigest()

    # Retrieve the stored hashed password
    query = "SELECT password FROM cv_pt.public.users WHERE username = %s"
    cursor.execute(query, (username,))
    result = cursor.fetchone()

    if result and result[0] == hashed_password:
        # Generate session token
        session_token = str(uuid.uuid4())
        query = "SELECT * FROM cv_pt.public.start_session(%s, %s, %s)"
        cursor.execute(query, (session_token, username, mood_rating))
        conn.commit()

        # Store session token
        session_token_path = os.path.join(os.path.dirname(__file__), '../../src/db/session_token.txt')
        with open(session_token_path, "w") as f:
            f.write(session_token)
        db.close()

        # open menu window
        from GUI.menu import MenuGUI
        MenuGUI()
    else:
        messagebox.showerror("Error", "Invalid username or password")
        db.close()
```

Figure 4.15: Code Snippet of the login function

```python
class DBConnection:

    conn = None

    def connect(self):
        try:
            self.conn = psycopg2.connect(
                dbname="[redacted]",
                user="[redacted]",
                password="[redacted]",
                host="[redacted]",
                port="[redacted]"
            )
            print("Connected to database")
            return self.conn
        except psycopg2.Error as e:
            messagebox.showerror("Error", f"An error occurred: {e}")
            return None

    def close(self):
        self.conn.cursor().close()
        self.conn.close()
```

Figure 4.16: Code Snippet of the DBConnection Class

invalid is not specified for security reasons, because if there is an indication that the username exists, it would be easier to brute force a login.

Figure 4.16 shows how to connect to a postgresql database using the psycopg2 library in python.

4.4 Testing and Debugging

4.4.1 Integration Testing

Testing whether the integration of the application into the GUI was fairly simple. All that needed to be done was to run the program and see if it displays. The first issue encountered was how to convert the default popup window openCV produces into a tkinter frame. This was solved by having the application be initialised as a tkinter frame by wrapping it in a class that inherits from ttk.Frame and running $super().\_\_init\_\_(parent)$ where *parent* is a parameter of the application.

The second issue was that the application would not open if an application had previously been opened during the program's uptime. This was caused by the application not closing properly when the window was closed. This was solved by creating a close method in the application class that destroys the application (see Figure 4.17) and calling it when closing the window.

```
def close(self):
    self.cap.release()
    self.destroy()
    cv2.destroyAllWindows()
```

Figure 4.17: Code Snippet of the close method in src/workouts/arms/bicep_curls.py

4.4.2 System Testing

For the testing of the backend system, no major bugs where encountered as I have had previous experience with setting up a postgresql database in the past and had knowledge on how to avoid issues. I must also say that the integration of database management in PyCharm facilitated the task significatly.

Chapter 5

Results & Discussion

5.1 Performance Evaluation

As there is no ground truth to compare the detections to, the evaluation of the detections will have to be qualitative rather than quantitative. That being said, the detections seemed to struggle when the lighting was too unfavorable such as when in a very sunny environment, and they also struggled to detect the user for the first time when their entire body wasn't visible, once a detection was made the users where able to get closer to the camera hiding parts of their bodies out of frame. The detections function both with the user facing the camera and with the user looking perpendicularly to the camera's line of sight. The accuracy of the angle calculation does seem to be lower when the user is facing the camera directly as the angle calculations do not currently use the z-axis (depth).

When all the relevant joints were in frame, the repetition counting was very accurate when testing, but when one or more of the joints went out of frame, a large number of false positives were counted. This can be solved by instructing the user to step further away from the camera or change the angle of the camera so that all the joints are in frame.

The performance of the database management system is sufficiently capable for the current scale and scope of the program, this will need to be revisited if the program reaches a much larger userbase.

The performances of the GUI and user experience are described in section 5.3

5.2 Comparison with Existing Solutions

While some virtual coaching solutions already exist, such as the Apple Watch and Samsung Galaxy Watch, it would be interesting to compare existing virtual coaching against computer vision-based systems. Both of the wearables do great jobs at health and fitness tracking, from heart rate monitoring and ECG, through monitoring blood oxygen levels to sleep tracking. They offer real-time feedback during workouts, automatic workout detection, integration with third-party fitness apps, making them quite versatile tools for various types of exercises. These watches, however, are limited as they are unable to detect the exact position and pose of the user, making them less efficient at detecting poor form. They provide no visual feedback at all during the exercise; only auditory and haptic feedback. Moreover, they can only automatically detect exercises that involve arm movement. This is quite opposite to computer vision-based systems, which work best for movement analysis and real-time feedback on the form and technique of exercises. Such information cannot be provided with regular wrist-based wearables. While smartwatches may be portable and convenient, computer vision-based systems can be much more accurate and richer in the movement analysis, hence highly effective for detailed coaching and corrections in form. The primary feature that the watches are capable of that visual methods can't is heart rate monitoring of vital signs including the measuring of blood oxygen, body temperature, and sleeping patterns. In theory the wearables are well placed to judge arm movements, but due to the fact that they are only worn one a single wrist, the user is constrained to bilateral exercises or to move the wearable to the other wrist to perform unilateral sets. The watch being positioned on one side of the user also allows the user to have poor form on the side that is not tracked. One advantage that wearables have over computer vision based methods is that it is not prone to poor camera placement and angles which would limit the effectiveness of the detections and quality of the recommendations.

5.3 User Feedback and Usability Testing

The login and register are reported as easy to use and simple to understand, they are familiar because they are similar to the login/register commonly found on websites. The history page received very positive feedback, with the user stating that it provides a quick summary that will be good for future workouts to know at what weight to start at, avoiding having to do sets at weights that are potentially too low to be effective or so high that they risk injury. They also stated that they appreciated the two list format. There was also very positive comments on the instant visual feedback provided by the camera and the visible skeleton of the body parts in motion during the exercise.

The main complaints came from the individual workout pages. There was confusion and annoyance with the rest timer, as the desired rest time would have to be re-entered after every set. The ease of understanding of the GUI was lacking, with no clear distinction between active time and rest time. The users expressed a desire for clearer instructions from the chatbox as at the time the only instruction was to set the rest timer if it had not already been done. The legibility at a distance of the text both on the camera side and the controls side was said to be poor, in particular the repetition counter at the top right corner.

Many recommendations were suggested and have been taken note of for improvement, such as moving the repetition counter to be on top of the camera feed and in larger font, adding more descriptive instructions to the chatbox, and having a "start set" button to make the application easier to understand.

5.4 Limitations of the Project

Due to the very short time-frame in which this project needed to be planned, designed, programmed, and written up, the scope of this project was very restricted. As mentioned in previous sections of this report, there are many possible improvements that I would have liked to be able to implement but was not able due to lack of time. This includes having functionality for more than the two arm exercises that

were implemented, having a more functional chatbox that would give feedback to the users specifically about their form, and not having the time to train, nor to obtain user feedback from more users. I also lacked the ethical approval for data collection for a deep learning model for repetition counting and form recommendations, which is something that would have greatly improved this project. As I am a single student developer and not a team of professional software engineers, this project was never going to result in a completed program with all the functionality required for public use, this is why this project acts more as a proof of concept and a foundation on which more research can based on.

Chapter 6

Conclusion

6.1 Summary of Project

This Project sought to create a computer-vision based training coach for computerised physical training, by creating a computer program capable of making recommendations on physical exercises and form to its users based on camera input where the users' body and movements are detectable. Implementing this project involved the creation of an application that is capable of the aforementioned tasks and embedding it into a graphical user interface that is easy to navigate and provides the necessary information to the user. A simple backend was attached allowing for a login system with workout history and mood tracking.

6.2 Contributions to the Field

This project contributed to the field of both computer science and more specifically human pose estimation but also to the fields of sports psychology and physical training. By creating this program, many research avenues have been facilitated and opened up in terms of studying the effect of exercise on mood and mental health and on how computer vision, HPE, and robotics in general can help in this sector.

6.3 Practical Implications

This project acts as a baseline from which further research can be make. Allowing for experts and researchers in the field of sports and physical health to utilise computer

vision technologies to perform studies on not only the impact of exercise on mood and mental health but also many more questions. This program, if improved and iterated upon, could have profound positive implications on public health in terms of getting the general population to exercise more and improve our understanding of certain workouts and how they are performed optimally.

6.4 Recommendations for Future Work

There is still much more that can be added to this program. More types of exercises, better recommendation system, using deep learning methods to train a recommendation model, and much more in terms of the application. Another recommendation for future work would be to port the program as a smartphone application, this will obviously require work on server side implementations and allowing models to either be much smaller while keeping their accuracy or be able to run on cloud computing.

6.5 Final Thoughts

Overall, even though I would have liked to have been able to implement more functionality into my application, I believe that this have laid a good foundation for future research.

References

Adams, Rolf and Leanne Bischof (1994). 'Seeded region growing'. In: *IEEE Transactions on pattern analysis and machine intelligence* 16.6, pp. 641–647 (cit. on p. 10).

Bochkovskiy, Alexey, Chien-Yao Wang and Hong-Yuan Mark Liao (2020). *YOLOv4: Optimal Speed and Accuracy of Object Detection*. arXiv: 2004.10934 [cs.CV]. URL: https://arxiv.org/abs/2004.10934 (cit. on p. 17).

Borisov, VV et al. (2023). 'Application of Computer Vision Technologies to Reduce Injuries in the Athletes' Training'. In: *International Conference on Intelligent Information Technologies for Industry*. Springer, pp. 137–145 (cit. on p. 2).

Canny, John (1986). 'A computational approach to edge detection'. In: *IEEE Transactions on pattern analysis and machine intelligence* 6, pp. 679–698 (cit. on pp. vi, 10).

Cao, Zhe et al. (2017). 'Realtime multi-person 2d pose estimation using part affinity fields'. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 7291–7299 (cit. on pp. vi, 12, 13).

Caviness, Celeste M et al. (2023). 'Effects of a bout of exercise on mood in people with depression with and without physical pain'. In: *Psychology, health & medicine* 28.4, pp. 1068–1075 (cit. on p. 22).

Chen, Liang-Chieh et al. (2017). 'Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs'. In: *IEEE transactions on pattern analysis and machine intelligence* 40.4, pp. 834–848 (cit. on pp. vi, 11).

Chen, Qifeng and Vladlen Koltun (2016). 'Full flow: Optical flow estimation by global optimization over regular grids'. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 4706–4714 (cit. on p. 14).

Comaniciu, Dorin, Visvanathan Ramesh and Peter Meer (2000). 'Real-time tracking of non-rigid objects using mean shift'. In: *Proceedings IEEE Conference on Computer Vision and Pattern Recognition. CVPR 2000 (Cat. No. PR00662)*. Vol. 2. IEEE, pp. 142–149 (cit. on p. 14).

Dalal, Navneet and Bill Triggs (2005). 'Histograms of oriented gradients for human detection'. In: *2005 IEEE computer society conference on computer vision and pattern recognition (CVPR'05)*. Vol. 1. Ieee, pp. 886–893 (cit. on pp. vi, 7, 8).

Debnath, Bappaditya et al. (2022). 'A review of computer vision-based approaches for physical rehabilitation and assessment'. In: *Multimedia Systems* 28.1, pp. 209–239 (cit. on p. 2).

Felzenszwalb, Pedro F, Ross B Girshick et al. (2009). 'Object detection with discriminatively trained part-based models'. In: *IEEE transactions on pattern analysis and machine intelligence* 32.9, pp. 1627–1645 (cit. on p. 12).

Felzenszwalb, Pedro F and Daniel P Huttenlocher (2005). 'Pictorial structures for object recognition'. In: *International journal of computer vision* 61, pp. 55–79 (cit. on p. 12).

Fossati, Chiara et al. (2021). 'Physical exercise and mental health: The routes of a reciprocal relation'. In: *International journal of environmental research and public health* 18.23, p. 12364 (cit. on p. 22).

Girshick, Ross (2015). 'Fast r-cnn'. In: *Proceedings of the IEEE international conference on computer vision*, pp. 1440–1448 (cit. on pp. vi, 8).

Girshick, Ross et al. (2014). 'Rich feature hierarchies for accurate object detection and semantic segmentation'. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 580–587 (cit. on p. 8).

Google (2023). *MediaPipe Solutions Guide*. URL: https://ai.google.dev/edge/mediapipe/solutions/guide (cit. on p. 18).

He, Kaiming et al. (2017). 'Mask r-cnn'. In: *Proceedings of the IEEE international conference on computer vision*, pp. 2961–2969 (cit. on p. 11).

Hochreiter, Sepp and Jürgen Schmidhuber (1997). 'Long short-term memory'. In: *Neural computation* 9.8, pp. 1735–1780 (cit. on p. 14).

Huang, Yu-Chuan et al. (2019). 'Tracknet: A deep learning network for tracking high-speed and tiny objects in sports applications'. In: *2019 16th IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS)*. IEEE, pp. 1–8 (cit. on pp. vii, 15, 16).

Hughes, John R (1984). 'Psychological effects of habitual aerobic exercise: A critical review'. In: *Preventive medicine* 13.1, pp. 66–78 (cit. on p. 22).

Jocher, Glenn (29th May 2020). *YOLOv5 by Ultralytics*. Version 7.0. DOI: 10.5281/zenodo.3908559. URL: https://github.com/ultralytics/yolov5 (cit. on p. 17).

Jocher, Glenn, Ayush Chaurasia and Jing Qiu (10th Jan. 2023). *Ultralytics YOLO*. Version 8.0.0. URL: https://ultralytics.com (cit. on p. 17).

Kalman, Rudolph Emil (1960). 'A new approach to linear filtering and prediction problems'. In: *The American Society of Mechanical Engineers* (cit. on p. 14).

Kendall, Alex, Matthew Grimes and Roberto Cipolla (2015). 'Posenet: A convolutional network for real-time 6-dof camera relocalization'. In: *Proceedings of the IEEE international conference on computer vision*, pp. 2938–2946 (cit. on p. 13).

Koch, Gregory, Richard Zemel, Ruslan Salakhutdinov et al. (2015). 'Siamese neural networks for one-shot image recognition'. In: *ICML deep learning workshop*. Vol. 2. 1. Lille, pp. 1–30 (cit. on pp. vii, 15).

Li, Chuyi et al. (2022). 'YOLOv6: A single-stage object detection framework for industrial applications'. In: *arXiv preprint arXiv:2209.02976* (cit. on p. 17).

Lin, Tsung-Yi et al. (2017). 'Focal loss for dense object detection'. In: *Proceedings of the IEEE international conference on computer vision*, pp. 2980–2988 (cit. on p. 9).

Liu, Mengyuan and Junsong Yuan (2018). 'Recognizing human actions as the evolution of pose estimation maps'. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1159–1168 (cit. on p. 2).

Liu, Wei et al. (2016). 'Ssd: Single shot multibox detector'. In: *Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11–14, 2016, Proceedings, Part I 14*. Springer, pp. 21–37 (cit. on p. 9).

Long, Jonathan, Evan Shelhamer and Trevor Darrell (2015). 'Fully convolutional networks for semantic segmentation'. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 3431–3440 (cit. on p. 10).

Lucas, Bruce D and Takeo Kanade (1981). 'An iterative image registration technique with an application to stereo vision'. In: *IJCAI'81: 7th international joint conference on Artificial intelligence*. Vol. 2, pp. 674–679 (cit. on p. 14).

Lugaresi, Camillo et al. (2019). 'Mediapipe: A framework for building perception pipelines'. In: *arXiv preprint arXiv:1906.08172* (cit. on p. 18).

Lynn, Trevor (July 2023). *Pose Estimation Algorithms: History and Evolution.* Roboflow Blog. URL: https://blog.roboflow.com/pose-estimation-algorithms-history/ (cit. on p. 1).

Newell, Alejandro, Kaiyu Yang and Jia Deng (2016). *Stacked Hourglass Networks for Human Pose Estimation*. arXiv: 1603.06937 [cs.CV]. URL: https://arxiv.org/abs/1603.06937 (cit. on p. 13).

NHS (2021). *Benefits of Exercise*. URL: https://www.nhs.uk/live-well/exercise/exercise-health-benefits/ (cit. on p. 4).

– (2023). *Obesity*. URL: https://www.nhs.uk/conditions/obesity/ (cit. on p. 4).

Otsu, Nobuyuki et al. (1975). 'A threshold selection method from gray-level histograms'. In: *Automatica* 11.285-296, pp. 23–27 (cit. on p. 9).

Redmon, Joseph, Santosh Divvala et al. (2016). 'You only look once: Unified, real-time object detection'. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 779–788 (cit. on p. 17).

Redmon, Joseph and Ali Farhadi (2017). 'YOLO9000: better, faster, stronger'. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 7263–7271 (cit. on p. 17).

– (2018). 'Yolov3: An incremental improvement'. In: *arXiv preprint arXiv:1804.02767* (cit. on p. 17).

Ren, Shaoqing et al. (2015). 'Faster r-cnn: Towards real-time object detection with region proposal networks'. In: *Advances in neural information processing systems* 28 (cit. on p. 8).

Ronneberger, Olaf, Philipp Fischer and Thomas Brox (2015). 'U-net: Convolutional networks for biomedical image segmentation'. In: *Medical image computing and computer-assisted intervention–MICCAI 2015: 18th international conference, Munich, Germany, October 5-9, 2015, proceedings, part III 18*. Springer, pp. 234–241 (cit. on p. 11).

Sung, George et al. (2021). 'On-device real-time hand gesture recognition'. In: *arXiv preprint arXiv:2111.00038* (cit. on p. 18).

Taylor, C Barr, James F Sallis and Richard Needle (1985). 'The relation of physical activity and exercise to mental health.' In: *Public health reports* 100.2, p. 195 (cit. on p. 22).

Viola, Paul and Michael Jones (2001). 'Rapid object detection using a boosted cascade of simple features'. In: *Proceedings of the 2001 IEEE computer society conference on computer vision and pattern recognition. CVPR 2001*. Vol. 1. Ieee, pp. I–I (cit. on p. 7).

Wang, Ao et al. (2024). 'Yolov10: Real-time end-to-end object detection'. In: *arXiv preprint arXiv:2405.14458* (cit. on p. 17).

Wang, Chien-Yao, Alexey Bochkovskiy and Hong-Yuan Mark Liao (2023). 'YOLOv7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors'. In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 7464–7475 (cit. on p. 17).

Wang, Chien-Yao, I-Hau Yeh and Hong-Yuan Mark Liao (2024). 'Yolov9: Learning what you want to learn using programmable gradient information'. In: *arXiv preprint arXiv:2402.13616* (cit. on p. 17).

Wei, Shih-En et al. (2016). 'Convolutional pose machines'. In: *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pp. 4724–4732 (cit. on pp. vi, 13).

Wojke, Nicolai, Alex Bewley and Dietrich Paulus (2017). 'Simple online and realtime tracking with a deep association metric'. In: *2017 IEEE international conference on image processing (ICIP)*. IEEE, pp. 3645–3649 (cit. on p. 15).