Final Report II

PostureFit - FYP22078

An Intelligent Fitness Trainer Application

COMP4081 - Final Year Project

Supervisor:
Dr. Schnieders, Dirk

Author:
Lioe, Andrew Tian (3035666633)

Members:
Lioe, Andrew Tian (3035666633)
Tinmanee, Phatrapol (3035667508)
Abstract

It can easily be understood that a large share of the population are willing to perform exercise, or at least are considering to include an effective (with proper posture) exercise regimen into their lives, no matter their age group or wherever they are. However, this is difficult to do due to the inconvenience of time, location, resources and most importantly, the absence of personal guidance during exercise execution. Improper execution mechanics (posture) and execution past a movement’s range of motion (ROM) are two of the common reasons for injury. This report describes how our team was able to tackle the issues mentioned by developing an exercise mechanics analyzer or simply, posture analyzer, with novices lacking such movement experience as our main target user group. The development of this analyzer application based on the Apple iOS ecosystem also involved tasks comprising of stages of the Data Analysis process. This enabled us to synthesize a well-defined Machine Learning model for detection and analysis. We have completed a prototypical (limited) build of PostureFit featuring a guidance system made for the squat and deadlift movements solely for the meantime due to time constraints.
Acknowledgments

We would like to express our gratitude to Dr. Dirk Schnieders, our final year project advisor and a Senior Lecturer in the University of Hong Kong, for his patience and effortful contributions in assisting us with this project’s development process. His advice and suggestions in consultation sessions of which we did not immediately realize, were profoundly helpful for us. The flexibility he has given us with the project’s due dates and deliverables suited to our needs and capabilities, was also a significant aid to this project’s success, especially for me, Lioe Andrew Tian, requiring special considerations due to medical reasons. Without his full-hearted guidance, we would not be this accomplished with our work.

We are also grateful for the involvement of Dr. Luo Ping, the second examiner of this project and an Associate Professor in the University of Hong Kong, for his contributions in examining our work and presentations, during the first in which he has provided us with a clearer understanding of the project’s relation to similar work and realize how complex our project could be - we could not have realized what critical challenges lied in front of us at that time without him mentioning about these matters, especially within the short timespan of the project.

This endeavor would also not have been possible without Dr. James Fong, my technical English lecturer from the Centre of Applied English Studies in the University of Hong Kong. The ability to express and communicate our project and its goals effectively would not have been possible without the guidance and many hours expended by him in training us.

Thanks should also go to my fellow friends, Darren Liang Li Wong, a final year MBBS student in the University of Hong Kong, Audrey Natalie Widodjati, a third year BSc student in the University of Hong Kong and Gladys Kifa Uchov, a final year BSc student in the University of Hong Kong, for providing us with valuable videos (data) of themselves performing the said movements. Our model and hence, the significant aspect of this project could not have been made without their contributions.
# Table of contents

Abstract .............................................................................................................. ii  
Acknowledgments ........................................................................................ iii  
Table of contents .......................................................................................... iv  
List of figure(s) .............................................................................................. vi  
List of table(s) ............................................................................................... vii  
1. Background ............................................................................................... 1  
   1.1. Feature Scope and Shortcomings ......................................................... 2  
   1.2. Platform for Future Work .................................................................. 2  
   1.3. Similar Existing Solutions ................................................................ 3  
2. Objectives .................................................................................................. 3  
   2.1. Intuitive fitness application ............................................................... 3  
   2.2. Cheaper alternative ......................................................................... 4  
   2.3. Body posture analyzer ..................................................................... 4  
3. Methodology .............................................................................................. 5  
   3.1. Data Collection and Classification .................................................... 6  
   3.2. Data Cleaning .................................................................................. 9  
   3.3. Data Extraction and Feature Engineering ........................................ 11  
   3.4. Exploratory Data Analysis ................................................................ 13  
   3.5. Counter Model Development ............................................................ 13  
   3.6. Mobile Application Development .................................................... 14  
      3.6.2. Settings View ........................................................................... 16  
      3.6.3. Main View ............................................................................... 16  
      3.6.3.1. Overlay View ...................................................................... 19  
      3.6.3.3. Arrow Drawing .................................................................... 21  
      3.6.4. Voice Feedback ....................................................................... 24  
4. Testing ....................................................................................................... 25  
5. Limitations ................................................................................................. 26  
   5.1. Left-side Correction Only ................................................................. 26  
   5.2. Front-facing Camera Issues ............................................................... 26  
   5.3. Counter Borderline Uncertainty ......................................................... 27  
   5.4. Presence of Other Bodily Figures ...................................................... 27  
   5.5. Overlay Line Color .......................................................................... 27  
   5.6. Angle of Camera .............................................................................. 27  
6. Results ....................................................................................................... 28  
7. Future Improvements ............................................................................... 29  
   7.1. Data Collection Variation .................................................................. 29  
   7.2. Consideration for Both Body Sides .................................................... 29  
   7.3. Fix for Front-facing camera inversion issue ...................................... 30  
   7.4. Repetition Counting Fix at Boundaries ............................................. 30
7.5. Color coding of overlays ........................................... 30
7.6. Testing considerations ........................................... 30
8. Conclusion .............................................................. 31
References ................................................................. 33
Appendix ....................................................................... 35
List of figure(s)

Fig. 1 Screenshot of the folder of videos collected ___________________________ 6
Fig. 2 Folders for graphic classification ____________________________________ 7
Fig. 3 Folder for deadlift score classification _______________________________ 7
Fig. 4 Folder containing deadlift score 5 videos _____________________________ 8
Fig. 5 Folder for squat score classification ________________________________ 8
Fig. 6 Folder containing squat score 3 videos ______________________________ 8
Fig. 7 Additional under-running image sequence (video) ____________________ 9
Fig. 8 Additional over-running image sequence (video) _____________________ 9
Fig. 9 Cleaned and targeted movement image sequence (video) _______________ 9
Fig. 10 A video frame with other bodily figures in the background ___________ 10
Fig. 11 17 key-points identifiable with MoveNet __________________________ 11
Fig. 12 Code to input the relative file path ________________________________ 12
Fig. 13 Video frames to aid with identification of top or bottom stances _______ 12
Fig. 14 Code to mark video frames with 0/1 for stance identification __________ 13
Fig. 15 Self-designed PostureFit logo ____________________________________ 13
Fig. 16 Splash Screen ___________________________________________________ 14
Fig. 17 Home View _____________________________________________________ 14
Fig. 18 Settings View ___________________________________________________ 14
Fig. 19a Counter protocol and Class flow _________________________________ 16
Fig. 19b Correction protocol and Class flow _______________________________ 16
Fig. 20 Main View _____________________________________________________ 17
Fig. 21 Dot connections for line drawing _________________________________ 19
Fig. 22 Code to calculate knee and hip angle ______________________________ 20
Fig. 23 Body segment orientation for typical conventional lifter ____________ 21
Fig. 24a Upwards arrow w/coordinates __________________________________ 23
Fig. 24b Up-rightwards arrow w/coordinates ______________________________ 23
Fig. 25 Covered body part test for squat _________________________________ 24
Fig. 26 45º camera angle test for squat _________________________________ 24
Fig. 27 Hip angle test for deadlift ______________________________________ 24
Fig. 28a Squat movement test on a different body structure ___________________ 25
Fig. 28b Squat movement test on a different body structure ___________________ 25
Fig. 29 Main View in action ____________________________________________ 28
Fig. A Squat ________________________________________________________ 35
Fig. B Deadlift ________________________________________________________ 35
List of table(s)

Table. 1 Performance comparison among the three major HPE models_________________ 5
1. Background

Exercise is an essential activity for maintaining a healthy and well regulated body physiology such as a beneficial reduction in body weight, improvements in bone structure, increase in insulin sensitivity (for the prevention of diabetes), reducing all-cause mortality associated with cancer and even preventing mental illnesses. [1].

Conventionally on learning exercise movements, people get started by referring to a range of how-to YouTube videos, reading online text tutorials, starting their search with “How to do a squat properly?” or even “How to squat without hurting my back?”, with the possibility of even hiring their own personal trainer (PT) to help them achieve their goals - a less cumbersome, but more expensive way. This describes the current situation for those looking to take part in physical activity (PA), specifically strength training.

Opting for self-learning methods as described above - referring to videos and reading online text, is time-consuming and usually imprecisely understood by the viewers (hence, a reason why PTs exist). It can be said to be bounded by viewers’ perspective of understanding the movement concept, either that they have minimal experience in PA and body mechanics or simply are just unable to understand the videos/imagery in detailed entirety - very common in beginners lacking such knowledge and experience. Performing repetitive improper movements is unfortunately, the most common issue leading to acute injuries [2] which may even turn chronic, if not corrected early. This may also be the cause of mobility issues in performers, which requires an ultimately different form of routine to fix this particular root cause of pain/injury, before proceeding with their desired exercise movement. This is typically only able to be identified by a 3rd party [3]. Henceforth, the apparent panacea would be to have someone specialized in identifying the causes to observe them, i.e., to hire a personal trainer. However, this may put forward another issue against beginners - the hourly rate charged is expensive [4]. This is where the team believes that our mobile-based application, PostureFit, would fit into and solve a range of problems in strength training. We aim to provide an application that shall provide users a safe and inexpensive, yet effective environment to exercise in, while nurturing a fun and painless experience.
1.1. Feature Scope and Shortcomings

Currently, we have completed a limited prototypical version of the software product with features including: repetition counter and posture guidance arrows during movement execution for at least two body parts per movement (i.e., squat and deadlift) which is explained in later parts of this Final Report II with references to Final Report I. The solution is less than what we have expected to accomplish in our project plan. Nonetheless, we have the main aspects of our goals available and working satisfactorily to solve a range of problems with repetition counting and posture guidance - with the identification of at least two body parts/pivots requiring correction as mentioned previously.

We were aiming to identify body parts/pivots which need correction with the use of a Machine Learning (ML) model, display a highlighting overlay on the corresponding body part/s and display animated guidance elements on-screen, such as animated arrows and angles to illustrate correction direction clearly. However, this proved to be challenging and particularly time consuming, especially with the short timespan allotted to us for this final year project. We were also aiming for simpler user interaction in which the user would be able to start the software without requiring to explicitly select a movement before proceeding with the exercise. Though, this feature may easily be understood as a gimmick, and may not be as effective as intended. Details of implementation is elaborated in section 3.

1.2. Platform for Future Work

We believe that this prototypical build can continue to serve as a platform on which we can continue to develop solutions (the application’s features) and incorporate further free-weight exercise movements, of which we are certain that the number of parts for correction guidance within our user target group (i.e., novice users) are numerous.
1.3. Similar Existing Solutions

There exist a number of similar solutions currently available in the market for download in app marketplaces such as the University of Hong Kong Sport AI Laboratory’s RoboCoach [5] or invite-only schemes which is the most predominant in today’s market. They are seemingly not directly accessible to users through the mentioned app marketplaces such as Apple’s App Store and Google’s Play Store, these include IncludeHealth [6] Alfa AI [7] and VAY [8]. Although they depict body overlays, repetition counters, measuring systems and scoring systems, it is unclear whether or not they work similarly to our approach, which is, along with the implementation of overlays and a repetition counter, the provision of an execution guidance system for correction. This is due to a lack of information and demo within their website in promoting their proposed solution.

2. Objectives

A prototypical version of PostureFit has been implemented and deployed on the Apple iOS ecosystem with a Graphical User Interface (GUI) - the “screen” where users would look into to interact with the application. With this, PostureFit embodies two of the most popular and complex movements to perform, i.e., squats [9] and deadlifts [10] (see Appendix) only for the meantime. It provides users with posture guidance and correction when parameters exceed the normal and proper range. PostureFit shall provide and solve the following:

2.1. Intuitive fitness application

There hardly exist mobile-based fitness applications that guide users towards their exercise goals without requiring them to presume whether the posture is proper for the particular movement they are performing or not. Such applications are generally one-way, i.e., they only depict and animate movements for users to see, and do not offer distinct and clear instructions for proper movement execution. Although similar solutions for posture guidance and correction exist in the market, they are generally on an invite-only basis, involve high fees and are vague in their promotion.
2.2. Cheaper alternative

Hiring a personal trainer has been the norm for most people looking to improve their fitness journey. This is usually impractical for most people due to the costs associated with subscribing to a personal training plan. We agree that similar solutions exist in the market, such as Alfa AI [7], however most are notorious with their pricing scheme, in that they do not offer flexibility to users’ needs - they charge users for functions they may never use or simply do not want. Another example is IncludeHealth, which works by observing movements of body parts with a repetition counter as guidance during training [6]. This, however, as far as the description of their solution explains, does not provide feedback concerning the user’s execution technique correctness. PostureFit observes, analyzes and corrects the user’s form, along with counting the repetitions in every movement set. IncludeHealth is also still under the invite-only scheme of software distribution, meaning that it is not possible to compare the efficacy between the two. 93% of the population older than 10 years old in Hong Kong owning a smartphone in 2021 [11], meaning that this solution can be adopted with zero additional hardware cost for most.

2.3. Body posture analyzer

We have developed an intuitive and effective body posture analyzer, although limited in the meantime, which allows users to exercise with proper guidance wherever they wish to - from the comfort of their home to fitness environments. This has been implemented onto the convenience of a mobile-based application developed for the Apple iOS ecosystem. With the software’s current development status and acting as a platform for future work, we are still aiming to pave the way for user adoption in the near future. We believe that this could be done with greater ease due to the significant market share of iOS users worldwide.
3. **Methodology**

We are utilizing an open-source pose detection model called *MoveNet* [12], developed by Google and readily available on the TensorFlow [13] library. This Computer Vision model is specifically developed for pose detection and Human Pose Estimation (HPE) i.e., the general problem in Computer Vision and Machine Learning with the goal to observe, analyze and identify the position and key body parts of a person in the open environment within a camera frame [14]. We have selected Google’s *MoveNet* model instead of other major HPE models such as OpenPose and PoseNet, as it has the fastest processing speed in comparison to the other two, despite scoring the lowest in terms of accuracy [15]. We are aiming for faster performance with an acceptable limit to accuracy, since the users will be moving while such processing is done within the mobile device itself - a more complex model will expend a longer processing time and require more power. Hence, the other two are not desirable for a mobile device. *MoveNet* also offers two variants of the model, namely: Lightning and Thunder. Lightning has been developed for latency-critical applications, whereas Thunder has been developed for high-accuracy applications. Nonetheless, it is claimed that both variants run faster than real-time applications (i.e., at 30+ FPS). This is faster than the rate at which the users performing the movements in scope will be moving, hence we have adopted *MoveNet* Thunder for our project. The mentioning of ‘*MoveNet*’ from hereon refers to the Thunder version of *MoveNet*.

<table>
<thead>
<tr>
<th></th>
<th>MoveNet Thunder</th>
<th>MoveNet Lightning</th>
<th>OpenPose</th>
<th>PoseNet</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Accuracy (%)</strong></td>
<td>80.6</td>
<td>75.1</td>
<td>86.2</td>
<td>97.6</td>
</tr>
<tr>
<td><strong>Frame rate (FPS)</strong></td>
<td>30</td>
<td>30</td>
<td>±20</td>
<td>±10</td>
</tr>
</tbody>
</table>

Table. 1 Performance comparison among the three major HPE models
The development of PostureFit commenced with stages associated with the process of Data Analysis (DA) i.e., Data Collection, Data Cleaning, Data Extraction/Feature Engineering, Exploratory Data Analysis and Model Development, at first. The development of a ML model to detect top and bottom positions of the squat and deadlift followed successively. In parallel, an iOS-based application with a GUI has been developed as a basis to contain and serve as a platform to test early builds of ML models and to implement our solution. This is where potential users are able to utilize our solution to solve the identified issues.

### 3.1. Data Collection and Classification

![Screenshot of the folder of videos collected](image)

Data best suited for analysis in the next steps is in the form of graphics, i.e., still images and videos (image sequence) of different body stances associated with the squat and deadlift. I have collected such data from:

i. Online sources/databases (e.g. YouTube, Google Images, etc.).

ii. Recording videos/images of ourselves or participating subjects (i.e., our acknowledged friends) performing the selected movements.

iii. Asking participants (i.e., our acknowledged friends) to submit recordings of themselves performing the selected movements.
I had to classify the graphics in each movement to be squats or deadlifts based on descriptions and instructions as explained in journal articles in [9] and [10], respectively, for further processing in the later stages of Feature Engineering. This classification should not be confused with Data Cleaning (i.e., the process of removing anomalous data from the dataset), as we should process data from both categories (i.e., squats or deadlifts). This is crucial as we utilize the datasets for Model Development - refer to Final Report I section 2.6, 2.8 and 3 for elaborated details.

Fig. 2 Folders for graphic classification

Fig. 3 Folder for deadlift score classification
Due to limited time, there is only a limited range of data that I have collected within the project’s timespan, potentially affecting accuracy in the model developed in a later stage. Details are elaborated in section 7.1.
3.2. Data Cleaning

The graphical data collected from the previous Data Collection process has brought the need to remove those that are anomalous, which may significantly affect processing in later stages, in terms of rate and accuracy. For our project’s scope, I have cleaned such imagery data by viewing each video/image file-by-file. This may be: a) under-/over-running videos that recorded additional frames of the subject not performing the required action, such as preparing the equipment to perform the movement or not stopping the recording after the subject is done with the movement, respectively, b) removing videos with excessive background noise (fig. 10), i.e. if another bodily figure appears within frame (such as crowded backgrounds in the fitness environment/gym) and even c) submitted videos that recorded frames of movements that are out of scope (e.g., push-ups) from incorrect submissions - these are classified as neither movements and stored in the ‘Unclassified_other’ folder as shown in fig. 2. This stage may considerably affect later stages of DA.

To demonstrate a) in the previous paragraph, a screenshot of a video file capturing unwanted frames is shown below in fig. 7 and 8:

![Fig. 7 Additional under-running image sequence (video)](image)

![Fig. 8 Additional over-running image sequence (video)](image)

A cleaned version of the video file recording the targeted movement (deadlift here) only can be seen in fig. 9 below:

![Fig. 9 Cleaned and targeted movement image sequence (video)](image)

An approximate of 30% of the total timespan of the project’s development process has been expended on this cleaning process of removing unwanted data.
Fig. 10 A video frame with other bodily figures in the background
3.3. Data Extraction and Feature Engineering

There only exist color and pixel bits in the graphical data that I have collected, therefore deeming it as ‘raw’ data. This is the step where MoveNet has brought about its functionality. With its utilization, we were able to identify all 17 body key-points as per MoveNet’s capability, comprising:

1. Nose
2. Left eye
3. Right eye
4. Left ear
5. Right ear
6. Left shoulder
7. Right shoulder
8. Left elbow
9. Right elbow
10. Left wrist
11. Right wrist
12. Left hip
13. Right hip
14. Left knee
15. Right knee
16. Left ankle
17. Right ankle

Fig. 11 17 key-points identifiable with MoveNet
The classified and cleaned data that I have worked on in earlier stages, along with the MoveNet model has enabled us to extract countless 17 key-point data records, where each key-point is represented as an X- and Y-coordinate with a confidence score. This has been done with Python on the Jupyter Notebook - with a classified and cleaned video as input, processed with MoveNet and the identified 17 key-points extracted into the comma-separated values (CSV) file format, i.e. the spreadsheet file, for storage. However, manual processing was required for this part.

In order to be able to get input into the Python program, I had to manually a) input the relative file path in my computer system to each and every video deemed suitable (i.e., after classification and cleaning) for processing (fig. 12) as collected in section 3.1, and b) indicate whether it is a squat or deadlift movement with ‘0’ or ‘1’, respectively, and indicate the corresponding movement execution score, i.e. score 0 - 5 (fig. 12). However, the scoring data was not used for this project at the end due to time constraints. Manual processing was then done to, c) identify which frames of the particular video correspond to top and bottom stances of the movement with the aid of printed video frames (fig. 13). A range of video frames is then, d) marked with integers of either ‘0’ or ‘1’, for top and bottom stances, respectively (fig. 14).

```
# insert path containing extracted image and out.csv
dir_path = '/Users/at/IHKU/YEAR_4/COMP4081_YMP/extractedFrames_perVideo/Deadlift/Deadli
vid_num = 242

# path save output
out_path = '/Users/at/IHKU/YEAR_4/COMP4081_YMP/extractedFrames_perVideo/Deadlift/Deadli:

# type of movement
# 1 = squat
# 0 = deadlift
movement = 0

# score
score = 5
```

Fig. 12 Code to input the relative file path - for task a) and b) in the previous paragraph

![Fig. 12 Code to input the relative file path - for task a) and b) in the previous paragraph](image)

Fig. 13 Video frames to aid with identification of top or bottom stances - for task c) in the previous paragraph

![Fig. 13 Video frames to aid with identification of top or bottom stances - for task c) in the previous paragraph](image)
This process is done for both the squat and deadlift. It is referred to as *Data Transformation* - the data extracted is now well-shaped for model development. Details of this processing and corresponding implementation algorithms are elaborated in Final Report I Section 2.3.

### 3.4. Exploratory Data Analysis

Elaborated in Final Report I Section 2.4.

### 3.5. Counter Model Development

Elaborated in Final Report I Sections 2.6 - 2.8.
3.6. Mobile Application Development

In parallel with all the above mentioned processing steps, I have developed an iOS-based mobile application built from the ground-up with Apple’s Integrated Development Environment (IDE), Xcode. At the beginning of this project, we had no prior experience in Apple’s Swift (programming language), and had to learn simultaneously with the project’s development. This may commonly be known as the ‘learning-by-doing’ method of learning. It was difficult to comprehend its syntax and working principles at first, and like most programming languages, I was able to build up my experience and knowledge in Swift at a progressively fast rate.

This mobile application has been the platform for testing our back-end algorithms and ML models, and serves as the main representative of PostureFit - could continue if we develop PostureFit further and distribute in Apple’s App Store (i.e. app marketplace). It does not require a server as data and functions will only be processed locally in the device. Swift, however, is only able to run TensorFlow Lite, which is a toned-down
version of TensorFlow made for mobile devices. It acts as an interpreter for model inference such as MoveNet, which we are utilizing for this project. This also requires the installation of Cocoapods, which is an application level dependency manager for Swift as it runs on the Objective-C runtime. It provides a standard format for managing external libraries. After which we have the TensorFlow Lite module as a Swift API for MoveNet to infer on.

The mobile application consists of the **home view**, **settings view** and **main view**, where users are able to switch between the home view and settings view by tapping on the corresponding icons at the bottom:

### 3.6.1. Home View

A splash screen (fig. 16) is implemented as an overlay in this view as the first “screen” that the user will see upon launching the application, it is part of the first *structure* (programmatic; similar to that of a *class*) to be run in this software, i.e. within the Home View Swift file. It is implemented to convey to the users that the application is running and loading for necessary background initialization - to prevent users from considering the stock white screen (if the splash screen is not implemented) as an ‘error’ or ‘crash’ that has occurred to the application. A zoom-in animation of the PostureFit logo (fig. 15) into the base home view is also implemented to indicate that the initialization is complete and ready for input with user interaction.

The base home view is where users are able to locate the movements (i.e. squat and deadlift) as depicted in fig. 17, and select one to perform by tapping on the respective gray box. Selecting either of the movements will hyperlink the user into the Main View, explained in section 3.6.3.
3.6.2. Settings View

Here, users are able to concurrently adjust the two settings within scope of this project while guidance algorithms are running: a) enable/disable voice feedback for every increment in repetitions and b) enable/disable overlays onto the bodily figure in frame. However, the user is required to tap on ‘Apply’ to save changes made on the settings - a limitation.

3.6.3. Main View

Upon selection of the movement, the user will be hyperlinked into the same main view for both selections of squat and deadlift. Selections made only differ in the counter ML model and correction algorithms being called, e.g. selecting the squat movement will call the squat counter ML model and the corresponding correction algorithm designed solely for the squat. This is made possible by the Swift Protocol, defined as: ‘A blueprint of methods, properties, and other requirements that suit a particular task or piece of functionality. The protocol can then be adopted by a class, structure, or enumeration to provide an actual implementation of those requirements.’.

The implemented protocols are adopted by a) respective repetition counter classes utilizing CoreML-converted counter ML models for both the squat and deadlift movements, and similarly for the b) respective classes implementing the correction algorithms.

---

Fig. 19a Counter protocol and Class flow for a) in the previous paragraph

Fig. 19b Correction protocol and Class flow for b) in the previous paragraph
This view contains:

a) A repetition counter on the top right in a blurred box (visual effect view);

b) Dot and line overlays when MoveNet detects a bodily figure in frame, if the ‘overlays’ setting is enabled;

c) Arrow overlays on body points requiring correction as detected by the correction algorithm.

*details of overlay output is explained in section 3.6.3.1.
This view is implemented with the *Storyboard* approach of developing a GUI on iOS - on-screen elements are positioned via the drag-and-drop method, and distancing constraints applied for their proper positioning, these include:

a) Overlay view;  

b) Visual effects view containing,  
c) Counter dynamic label and,  
d) ‘Reps’ static label;  
e) ‘Flip Camera’ button.

The entirety of Main View and the above elements are linked to code with a *View Controller* Swift file, and as the name suggests, it controls corresponding actions (functions) associated to each element or group of elements with/without user input. In the Main View, it is where the objectives of repetition counting and correction guidance are implemented. An *AVCaptureSession* object, i.e. a class which activates the camera and starts buffering video (frames) input, is called for processing by *MoveNet* and our self-developed counter ML model and correction algorithms. This object (programmatic) is handled by another self-defined class responsible for camera input matters (i.e. the *CameraFeedManager* class). This view is also the frame in which the user can position themselves into to begin their exercise.

As the user performs either movement, repetition counting runs on the basis of change in top and bottom stances as detected by our self-developed ML model. The process of deciding when to increment the counter is elaborated in Final Report I section 2.9.2 and 3.3. The counter label is updated after every thread cycle of the counter function is completed, whether or not there is a change depends entirely if the said decision process increments the counter variable associated with the label.

The user may also flip the camera between the rear-facing camera and the front-facing camera by tapping on the ‘Flip Camera’ button, concurrently while the repetition and guidance algorithms are running. This is also handled by the *CameraFeedManager* class.
Details on how overlays (dots, lines and arrows) are implemented is elaborated below:

3.6.3.1. Overlay View

This is where the overlays are drawn on the user. Upon detection of the user’s body parts in frame by MoveNet from input received from the AVCaptureSession object, processing is done in the background for overlay drawing with a) dots on the 17 key-points representing user body parts and b) lines on limbs as defined by code with key-point to key-point connection (fig. 21). This can be enabled/disabled with a switch in the settings view.

```
static let lines = {
    (from: BodyPart.leftWrist, to: BodyPart.leftElbow),
    (from: BodyPart.leftElbow, to: BodyPart.leftShoulder),
    (from: BodyPart.leftShoulder, to: BodyPart.rightShoulder),
    (from: BodyPart.rightShoulder, to: BodyPart.rightElbow),
    (from: BodyPart.rightElbow, to: BodyPart.rightWrist),
    (from: BodyPart.leftShoulder, to: BodyPart.leftHip),
    (from: BodyPart.leftHip, to: BodyPart.rightHip),
    (from: BodyPart.rightHip, to: BodyPart.rightShoulder),
    (from: BodyPart.leftHip, to: BodyPart.leftKnee),
    (from: BodyPart.leftKnee, to: BodyPart.leftAnkle),
    (from: BodyPart.rightHip, to: BodyPart.rightKnee),
    (from: BodyPart.rightKnee, to: BodyPart.rightAnkle),
}
```

Fig. 21 Dot connections for line drawing

Alongside, execution guidance is also running in the background, where corresponding arrows are displayed when a body part passes a threshold level, as suggested by the aforementioned journal articles in [9] for the squat and [10] for the deadlift, indicating a need for correction of the overlayed body part towards the direction of the arrow. The threshold levels may lie along the X-axis, Y-axis or angles as calculated by mathematical equations. Details are explained in section 3.6.3.2.

3.6.3.2. Correction Guidance

Squat Thresholds:

1. The chest is recommended to be held upward and shoulder blades are retracted [9]. This is implemented with the nose’s X-axis value not being
too far beyond the left\(^1\) knee’s X-axis\(^2\) value, indicating that the user is leaning too much to the forwards.

2. At the apex of depth, the tops of thighs are recommended to be at least parallel to the ground [9]. This is implemented by checking if the left\(^1\) knee’s Y-axis\(^2\) value and left\(^1\) hip’s Y-axis\(^2\) value are equal. However, we have allowed a range in implementing this, in order to create more flexibility for the users.

**Deadlift Thresholds:**

Referring to the studied angles regarding the [conventional] deadlift [10] as depicted in fig 23, as normal acceptable angles. I have implemented correction guidance for:

1. The hip angle in which it should be less than \((60+30)^\circ\) at the bottom position (position i in fig 23) and more than \((130-30)^\circ\) approaching the top position (position ii in fig 23). Adjusting \(30^\circ\) to each found angle is to ensure flexibility to the user and potential errors in calculations. This is implemented by calculating the knee angle and hip angle with the equation and code:

```javascript
let slopeLeg = (leftKnee.y - leftAnkle.y) / (leftKnee.x - leftAnkle.x)
let slopeThigh = (leftHip.y - leftKnee.y) / (leftHip.x-leftKnee.x)
let slopeThorax = (leftShoulder.y - leftHip.y) / (leftShoulder.x-leftHip.x)

var angleKnee = atan( (slopeLeg - slopeThigh) )
var angleHip = atan( (slopeLeg - slopeThorax ) )
```

Fig. 22 Code to calculate knee and hip angle; leftKnee.y represents the Y-axis value of the leftKnee at a particular time, whereas leftKnee.x represents the X-axis value; atan: arctan

2. Stopping at the 180\(^\circ\) hip angle at the top position (position iii in fig 23), and not cross beyond. This is simply implemented by checking if the left ear X-axis\(^2\) value is not greater than the left hip X-axis\(^2\) value.
The left body parts are chosen due to the limitation that we have only considered correction with the left side of the body only. Calculations and other unknown factors need to be considered for the other side. Limitation elaborated in section 5.1.

\(^2\)X-axis value increases towards the right of the frame; Y-axis value increases towards the bottom of the frame.

3.6.3.3. Arrow Drawing

Overlaying an arrow on-screen is not as simple as pasting a transparent (like that of PNG files) arrow-shaped graphic directly. This required mathematical calculations transformed into a drawing of an arrow, as an extension of the UIBezierPath class in Swift. Extensions add new functionality to an existing class, structure, enumeration, or protocol type. This includes the ability to extend types for which we do not have access to the original source code (known as retroactive modeling). This consist of calculations for: the arrowhead’s width and length, and the arrow’s tail width. Fortunately, this has already been implemented on GitHub [16] by Rob Mayoff, made freely available for our use without license. We simply have to make a call to the
extension. However, as the call requires coordinates for the starting and ending positions of the arrow on-screen, I had to consider its implementation.

Since MoveNet already provides real-time coordinates to the 17 key-points of the body, this can simply be implemented by taking the associated body part’s coordinates identified to require correction guidance with arrows, as the center coordinate of the arrow. It is now possible to find the starting and ending coordinates of the arrows for the directions as I have considered below:

1. Left
2. Right
3. Up
4. Down
5. Up-Left
6. Up-Right
7. Down-Left
8. Down-Right
Below are figures to demonstrate adjustments for the arrow directions, upwards and up-rightwards:

**Upwards** (Fig. 24a): Taking (256, 178) as the XY-coordinate received from MoveNet, it can be understood that the start coordinate can simply add (+70 here; value depends on desired length of the arrow) to the Y-axis value of the received coordinate - (256+0, 178+70), and the end coordinate can simply subtract from the Y-axis value - (256+0, 178-70).

**Up-rightwards** (Fig. 24b): Taking (256, 178) as the XY-coordinate received from MoveNet, the start coordinate can simply add (+50 here to maintain the arrow length equal to the upwards direction arrow by Pythagoras’ theorem) to the Y-axis value of the received coordinate and subtract from the corresponding X-axis value - (256-50, 178+50), and the end coordinate can simply subtract to the Y-axis value and add to the corresponding X-axis value - (256+50, 178-50).
3.6.4. Voice Feedback

As the counter variable is incremented, an AVSpeechUtterance class is called to encapsulate the text for speech synthesis and parameters that affect the speech. Afterwards, the AVSpeechSynthesisVoice class is called for the voice used in speech synthesis. Altogether, a read-out of the repetition number after every repetition increment is output via the device’s loudspeaker.

This can be enabled/disabled with a switch in the settings view.
4. Testing

Although I did not have the abundance of time to test the software, I was able to test the main aspects of the features offered by PostureFit within current scope. Apart from testing the repetition counter in all, which works well with continuous movement, below are what I have found:

a) Tested myself with a barbell plate (the blue weight) to cover a part of myself still detected my body parts. However, when another bodily figure appears in-frame in the background with higher confidence scores than myself, identification shifted towards that background figure. Shown in figure 25.

b) Tested the camera angle at 45° from the direction I was facing still detected my body parts. However, I have noticed a moderate parallax error rate (i.e. differences in the length of limbs and XY-coordinates of body parts, as compared to a 90° angle) during testing, in which the arrow appears when it does not at a camera angle of 90°. Shown in figure 26.
c) Tested myself performing a deadlift movement improperly, testing **deadlift threshold 1** in section 3.6.3.2. It works as expected with the downwards arrow for correction guidance appearing as an overlay on my hip. Depicted in figure 27.

d) Tested on other subjects of different body structures to ensure an unbiased testing. These tests are done for **squat threshold 2** in section 3.6.3.2. It works as expected with the upwards arrow for correction guidance appearing as an overlay on the subjects’ hip. Depicted in figures 28a and 28b.

5. **Limitations**

5.1. **Left-side Correction Only**

Towards the end of this current project cycle, I have only realized that our correction implementation only considered for the left bodyside correction only. The reason behind this is mainly due to the threshold design as mentioned in section 3.6.3.2, when utilizing the X-axis for correction, as it increases towards the right. I did not have the time to fix this issue.

The issue with angles is still unclear, though I have found during debugging sessions with console printing, that the angles calculated are negative, i.e. rotating towards the opposite direction, if the camera faces the right bodyside.

5.2. **Front-facing Camera Issues**

Similarly, towards the end of this current project cycle, I have only realized that when utilizing the front-facing camera there are issues with the correction guidance. The reason is currently unclear. However, I hypothesize that it may also be due to the aforementioned correction algorithm design, involving the increasing X-axis value towards the right. This is highly likely because the camera orientation for the front-facing camera is inverted horizontally, for the intuitive mirror-like recording of the user. Otherwise, it would be confusing for the user when looking into the screen, to see that the recording moves in an opposite direction.
5.3. Counter Borderline Uncertainty

Whilst performing the two movements during testing, we noticed that at the borderline between identifying top and bottom position by the ML model, there is a sudden burst in the increment of the repetition counter. This indicates that there is also a rapid change between the detection of top and bottom by the ML model. However, this might not be an issue in practice, as the users would perform the movements continuously without pausing at the middle. This is identical for both movements. Indeed, it is a call to further improve the two ML models’ accuracy.

5.4. Presence of Other Bodily Figures

During testing in section 4 point a), I noticed that as another bodily figure (person) enters into the frame for guidance, there is a potential that MoveNet would detect the background figure as the main figure. This could be due to higher confidence scores of the background figure as calculated by MoveNet when part of my body is covered. This can be seen in figure 25.

5.5. Overlay Line Color

I did not manage to apply different colors to the dots, lines and arrows on time. Therefore, they all are currently red.

5.6. Angle of Camera

With testing in section 4 point b): There exists a moderate parallax error rate (i.e. differences in the length of limbs and XY-coordinates of body parts, as compared to a 90° angle) during testing, in which the arrow appears when it does not at a camera angle of 90°.
6. Results

Alongside testing, I was able to identify that most aspects and the end-goal of the project has been achieved. As per our approach to solving the issue of execution correction guidance with exercise movements of the squat and deadlift, the counter ML model and algorithms work as expected in counting the repetitions while the user is performing the exercise for both movements in scope of this project, though marginal errors still exist as mentioned in the limitations in section 5.3. This can be seen in figure 29, implemented as the counter label output for users to refer to. The option to enable/disable the voice feedback concurrently while the guidance algorithms are running, for repetition counting also works as expected. Moreover, the correction algorithm is also working as expected, again, even though limitations associated to left bodyside guidance only as elaborated in section 5.1 and inversion issues associated with the front-facing camera as elaborated in section 5.2. As it can be seen in figure 29, the correction guidance arrow can be seen to be overlayed on the hip.

Fig. 29 Main View in action

Dot overlay on left ear
Upward correction arrow overlay on left hip
Line overlay from left knee to left ankle
body part, indicating a need to bring the associated body part upwards as it is detected to be too low with regards to **squat threshold 2** in section 3.6.3.2. This is for the squat movement.

The option to enable/disable the overlays through the settings concurrently while the guidance algorithms are running, like the voice feedback for the counter, also works as expected, and can be seen to overlay identified body parts and corresponding limbs as shown in figure 29.

As mentioned as a limitation in section 5.5, currently all overlaying elements, i.e. the dots, lines and arrows, are in the red color due to issues and time constraints.

7. **Future Improvements**

7.1. **Data Collection Variation**

In the future, data collected should be more diverse and varying, as compared to the current dataset we have. This may be from the collection of more data from more subjects of different body structures and shapes and also gender, the level of experience in doing the movements in scope of the work and also the collection of recordings at different camera angles and sides of the body (i.e. left or right bodyside). This would ensure a wider range of data values to be collected for a more accurate repetition counter ML model, and potentially for the development of ML models for execution guidance correction. We have realized the potential of ‘decision trees’ to get values within range for thresholds of the movements in scope.

7.2. **Consideration for Both Body Sides**

Like data collection, we shall also consider the right bodyside of camera placement as well. This would add to the ease of use by the user, where the user may be able to place their device as appropriate to their comfort and surrounding environment.
7.3. Fix for Front-facing camera inversion issue
This is a crucial factor to be considered and fixed, as one of the main goals of PostureFit is to provide feedback to the user regarding their exercise execution. This would not be possible without looking into their device’s display if only the rear-facing camera is implemented, defeating its purpose. The feedback mentioned here are mainly the overlays and arrows, as the graphical aspect of feedback.

7.4. Repetition Counting Fix at Boundaries
As a limitation mentioned in section 5.3, it is necessary to consider a fix for the detection of top/bottom position, as the ML model is currently uncertain at the boundary region in between the two position. This would in turn serve as part of the fix for the repetition counter increment bursting issue, as mentioned in section 5.3. Details on how this could be improved is explained in Final Report I section 4.3.

7.5. Color coding of overlays
Color coding is also an important aspect to consider for graphical user feedback. With the use of colors, users would find it simpler to understand what is right or wrong, such as associating them with the colors green and red, respectively. I have understood the need to apply overlays with more conspicuous multi-tone colors, to indicate what each indicates intuitively, for future work.

7.6. Testing considerations
Indeed, testing shall be done with more scenarios and a well-defined plan. It should involve more subjects that could not be part of the training data in regards to the development of the ML model, or even not part of the remaining testing data. Using never seen data for testing would be best to identify strengths and weaknesses in the developed models, as they tend to not overfit into the development data (incl. training data). An over-simplistic definition of overfitting would be: the ML model created is unable to generalize to real-world scenarios and fits very tightly to the training/testing dataset.
8. Conclusion

PostureFit is a versatile and intuitive mobile application based on Apple’s iOS ecosystem, designed to be readily available to the general population. With its execution guidance features, it assists users in the types of exercise movements (i.e., squat and deadlift for now) they wish to perform with effective and proper form. This would in turn help to reduce or potentially eliminate pain entirely, that might otherwise be experienced if done improperly. Alongside, with 93% of the Hong Kong population older than 10 years old already owning a smartphone, distributing our solution to potential users would be simpler with the required hardware already with them. PostureFit shall also provide as a solution to the expensive training fees associated with hiring a personal trainer, at an average of HK$650 per hour. Conveniently on-hand and in-device, users can start exercising with guidance wherever they are and whenever. Our solution does not have network constraints, as processing is done locally in the device.

We have explored open-source pose estimation models available for use, namely: OpenPose, PoseNet and MoveNet. The decision to adopt MoveNet was due to its appropriate performance in speed and accuracy - 30+ frames per second analysis rate on a moving subject, with an average accuracy rate of 80.6% (Thunder variant). We are also aiming to be as energy-efficient as possible with the adoption of MoveNet, without sacrificing a great amount of accuracy. Following the adoption of a pose estimation model, we were able to perform stages involved with DA - Data Collection, Cleaning, Transformation, EDA and Model Development. This can be said to be the most crucial phase in the project in order to be able to yield a useful and precise ML model.

In addition, the utilization of a mobile device’s embedded cameras would be the quintessential in terms of convenience for the user - there is no requirement to purchase additional hardware to be able to perform proper exercise. The GUI that we have designed and developed shall serve as the platform in which the user can utilize PostureFit’s functionality and hence, the project’s main purpose.
We acknowledge that limitations exist, of which a major one is the positioning of the camera (device) from the user at angles away from 90º and its orientation (front-facing camera issue). We understand that it is of natural preference that users would like their mobile device to be placed right in-front of them, without the need to glance to their side periodically, straining their neck. However, due to physical limitations, this is not possible without placing the camera around the side of the user. We have also identified potential areas for future improvement such as the consideration for the choice of placement of the camera on either body sides. We believe that such improvements are crucial to the overall development of PostureFit and user convenience. Overall, although with minor delays, setbacks and difficulties that we have encountered along the way, we have indeed learned and experienced a lot beyond the technicals of Computer Science and programming, such as the aspects of teamwork. We believe that our Final Year Project, PostureFit, has been a success.
References


Appendix

Squats and deadlifts as depicted in Fig. A and Fig. B respectively, are very complex movements to perform, requiring gross motor skills (large muscle involvement) and a great amount of concentration, due to which inexperienced individuals performing either movements have a high chance of executing it improperly, if not guided by a professional. This is the main reason why we have chosen them as the first two types of exercise movements that PostureFit shall aid with.

The squat mainly targets the gluteus Maximus, minimus and medius (buttocks), quadriceps (front of the thigh) and hamstrings (back of the thigh) - a good number of large muscle groups to explain the reason behind the need of experience and good coordination in order to perform either movements properly. If a squat is done improperly, for example by leaning too much forward during execution, it will cause stress on the lower back instead of the target areas as mentioned before. This may also lead to lower back pain, knee pain and possibly herniated discs if done heavily and for a period of time.

The deadlift on the other hand, targets hamstrings, glutes, hip flexors, quads, core muscles, upper back muscles and lower back muscles - an even more complicated exercise movement in comparison to the squat. With similar reasons behind the need of experience and good coordination for proper execution, it is worth to reiterate that PostureFit is being developed to help beginners execute the movements properly and to prevent the aforementioned problems that might be encountered if done improperly.