FPT UNIVERSITY

FPT University

AI-Based Solution for Exercise Posture Correction

Ngo Quoc Bao, To Van Duc, Pham Thien Nhat

Supervisor: M.S.E. Le Dinh Huynh

December 22, 2022

Project Specification

The authors confirm contribution to the thesis as follows:

Ngo Quoc Bao:

- Software design.
- Web and Android implementation.
- Deployment.
- Validation and reproducibility.

To Van Duc:

- Posture correction research.
- Methodology.
- Evaluation.
- Interpretation and analysis.
- System implementation.
- Administration.

Pham Thien Nhat:

- Pose estimation research.
- Data collection.
- Evaluation.
- Interpretation and analysis.
- System implementation.

All authors took part in the conceptualization and the writing process of the thesis.

Abstract

In recent years, interest in all things related to physical wellness has been rising rapidly, especially after the COVID-19 pandemic – which brought about social distancing and many infrastructure shutdowns. Normally, people would go to the gym for wellness improvement, but during the pandemic lockdown, they can only do so at home with limited equipment and almost no direct guidance from an expert. However, recent achievements in human pose estimation have shown significant progress in the field, opening the possibility of a system that can utilize the real-time efficiency potential to offer posture correction. In this thesis, we present an human pose estimation-based exercise posture correction system leveraging a technique called batch processing and apply it to three weightlifting exercises. The system is able to correct practitioners' posture in a real-time manner. The thesis work also includes web and Android applications to demonstrate the practical aspect of the system.

Keywords — human pose estimation, exercise posture correction, real-time, application.

Contents

1 Introduction

Performing physical exercises regularly is vital to one's overall health and well-being, especially during the era in which most people live sedentary lives. There are different types of exercises with different aims. For example, cardiovascular exercises which include running, cycling, jumping jack, etc. are shown to lower blood pressure, improve sleep and mood, and are beneficial for your heart [1]. Weightlifting and calisthenics help increase strength and build muscles [2]. Overall, some forms of exercise are highly recommended to be included in one's weekly schedule.

However, while exercising can be beneficial, there are always potential injury risks associated with them. For example, weightlifting exercises are centered around body movement which involves moving increasingly heavy weights for hypertrophy, which is an increase and growth of muscle cells. Lots of pressure is put not only on the practitioner's muscles, but also on joints, and these exercises can induce systemic stress and fatigue as well. Statistics [3] show that 4.5% of men, and 0.6% of women participating in weight training experience injury within 1 year. The risk factors associated with these injuries include older age and excessive training time [3]. The spine, shoulders, and knees are the most susceptible to these kinds of exercises [4]. Thus, it is important to perform them in the correct form. Many exercises are mostly safe, but only if the form is correct.

Due to the potential danger of performing exercises wrong, beginners and intermediates alike usually seek professional guidance from personal trainers (PTs). PTs help clients to perform exercises correctly, recommending a personal exercise routine or diet. It has been shown that personal trainers can help positively impact clients' attitudes toward physical activities [5]. However, many people either can not afford the service or find it not worth buying given their interest and the usual price range for PT services. Also, most personal trainers are often associated with certain gyms, meaning users have to go to the gym and get guidance from PTs at that gym. This can be inconvenient for people training at home. Thankfully, there are plenty of online video tutorials available showing how to perform exercises correctly. However, many people still perform exercises in the wrong form while knowing the right way to do

them, mostly because they are not aware of both the major and minor details of their form during exercises.

There are systems making use of the advance in artificial intelligence and deep learning to help solve the problem. One approach is to make use of human pose estimation models. One of the firsts of its kind was dated back before 2001 that requires wearing a suit with markers, and nowadays markerless, deep learning-based systems are rapidly growing in accuracy and popularity [6]. These models can be used to estimate the skeletal posture of people. Afterward, it is possible to give guidelines or corrections based on which.

In this thesis, we tackle the specific task of correcting postures of exercises and seek to create a real-time, human pose estimation-based system that can help correct wrong exercise postures. The system provides real-time feedback as the users perform exercises, which helps them immediately correct their posture. We also aim to demonstrate its practicality by building complementary web and Android applications.

2 Related Work

2.1 Human Pose Estimation

Human pose estimation (HPE) is a general problem in computer vision that involves detecting the position and orientation of a person, primarily by estimating the configuration of human body parts. This task is challenging due to a wide variety of poses, numerous degrees of freedom, costs, complexity, and occlusions. However, the resulting geometric and motion information of the human body can be utilized in many applications (*e.g.* , CGI, Virtual Reality, healthcare,...). Thus, more techniques are being actively developed, and their capabilities are being pushed further.

Traditionally, one of the most widely used techniques for the task is Motion Capture (Mo-cap) [7]. It requires the target to wear a suit with the attachments of markers, fixtures, or sensors (usually reflective or LED-lit) near each joint, which then will be tracked by one or multiple motion camera(s), or sensors [8], [9]. While the technique is still widely used in studios, it is not approachable by everyone, mainly due to the costly equipment and complex setup process. In our case, this would be against one of our aims - ease of use - we want to bring an exercise posture correction solution suitable and usable for everyone.

There exist several computer vision techniques that try to overcome the problems of Mo-cap by limiting it to only the 2D plane. Andriluka *et al*. proposed a generic approach based on the pictorial structures framework [10], which is a powerful and efficient model for part constellations inference. Yi Yang and Deva Ramanan used a flexible model of part mixtures [11] to capture contextual co-occurrence relations between parts. However, they all face the general problem of poor generalization and accuracy, mainly due to depth ambiguity and the inability to capture hidden or unclear limbs. As research and development in deep learning started to take off, especially with the emergence of the convolutional neural network (CNN). It has sparked a breakthrough in HPE by improving performance significantly and overcoming past problems of classical approaches.

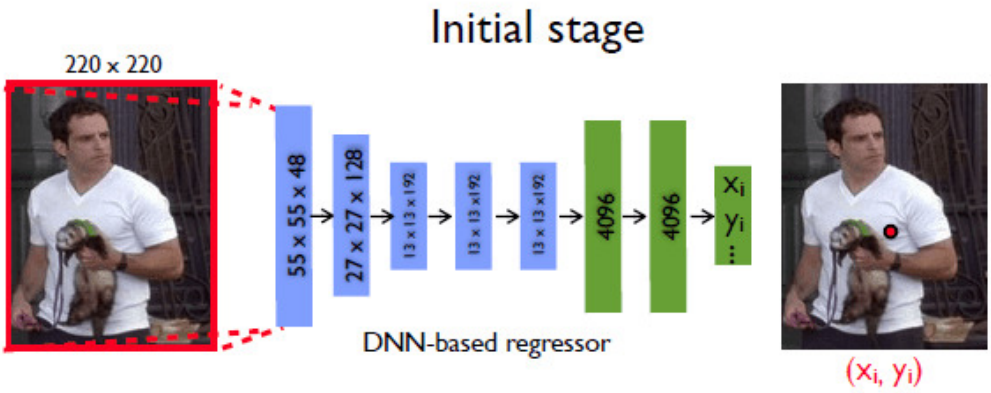


Figure 1: DeepPose uses AlexNet as its CNN architecture [12].

DeepPose [12] was one of the first deep learning-based approaches to HPE, which re-defined the problem as a CNN-based regression problem towards body joints by using AlexNet [13] as the backbone as shown in Figure 1. While it has shown promising performance and arguably holistic reasoning, it has also brought forth new challenges: occlusion (*e.g.* in a crowded area) and computational efficiency as the model would struggle with multi-human cases, which need increased computational power. With its impressive performance, DeepPose began to shift the research paradigm of HPE from classic approaches to deep learning, especially CNNs.

Figure 2: Two common approaches to human pose estimation, adapted from [14]. (a) Top-down approaches give pose estimation for each detected human. (b) Bottom-up approaches give joint estimation directly for every body part, which will then be grouped up and assembled into human poses.

Deep-learning approaches to human pose estimation consist of two main approaches: top-down and bottom-up. As shown in Figure 2, the top-down approach divides the

task into two sub-tasks: human detection and pose estimation of each human. The bottom-up approach is also composed of two sub-tasks: estimating the human body parts and assembling them into individual poses. It is also noteworthy that methods to extract keypoints would begin to shift from regression to body joint heat map estimation as research in segmentation took off.

Computation-wise, bottom-up approaches are usually faster when performing HPE on multi-person cases, due to not having to perform pose estimation for each individual separately. They also excel when handling complex poses and crowded areas, but suffer a lot from false alarms, usually misidentifying objects in the environment as parts. Top-down approaches, on the other hand, suffer less from false alarms and ambiguity and thus can be more robust. When used for single-person HPE, top-down approaches usually perform better than bottom-up counterparts. In our case, as multi-person HPE is not needed, we can focus more on top-down approaches for performance gain, which is crucial for real-time inference.

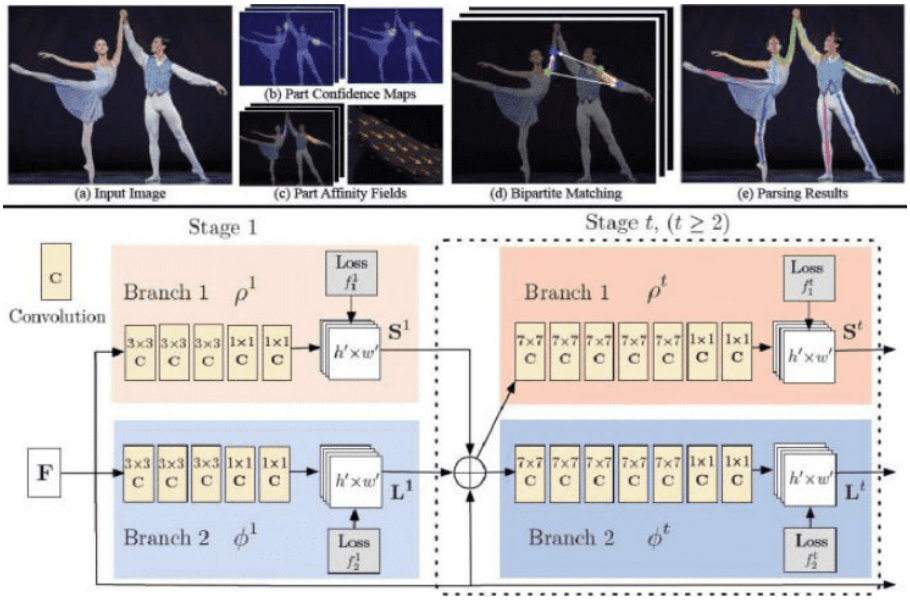


Figure 3: Proposed architecture of OpenPose [15].

OpenPose [15] is a well-known method following a bottom-up approach, which uses a CNN as its main architecture. It consists of a backbone (the authors used VGG-19 in the paper) to extract patterns, which is then fed into two separate branches of convolutional networks as shown in Figure 3. The first network would produce a confidence map (sometimes also referred to as a heat map) of each body part, while the second network would produce a Part Affinity Field that represents a degree of association between parts. The model can scale very well and have good performance and accuracy even when facing multi-human cases.

Mask R-CNN [16] is usually used for instance segmentation, however, it can also be

utilized for HPE tasks [17]. By simply extending and modifying the basic architecture, the human pose can be estimated from both keypoints heat map and human detection output. Note that while the method somewhat resembles a top-down approach, it is not entirely top-down, since human detection and keypoints of body parts detection are performed parallelly.

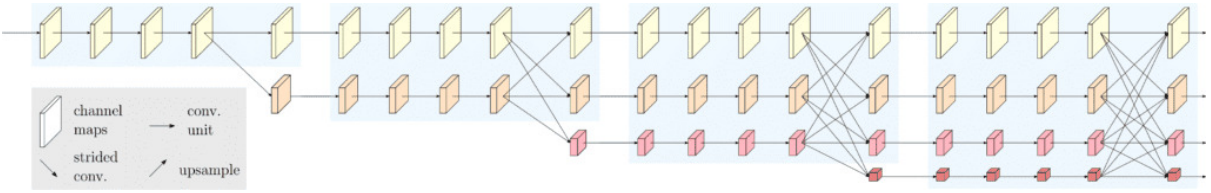


Figure 4: Proposed architecture of HRNet [18].

HRNet [18] is a more CNN-oriented top-down approach, but imagined differently: The architecture would connect high-to-low resolution convolution streams in parallel rather than in series as shown in Figure 4. They claimed that the architecture can maintain high-resolution representations that are not only semantically strong but also spatially precise. By applying the said process to a heat map estimation framework, the authors claimed to achieve superior results compared to prior methods when benchmarked on COCO 17-keypoints dataset. There also exists a slim and less computationally complex version of it, called Lite-HRNet [19], which applied a more efficient version of ShuffleBlock [20] to the backbone of HRNet. It achieved a very good accuracy-complexity tradeoff, effectively making real-time mobile HPE more viable - which lines up with our interests.

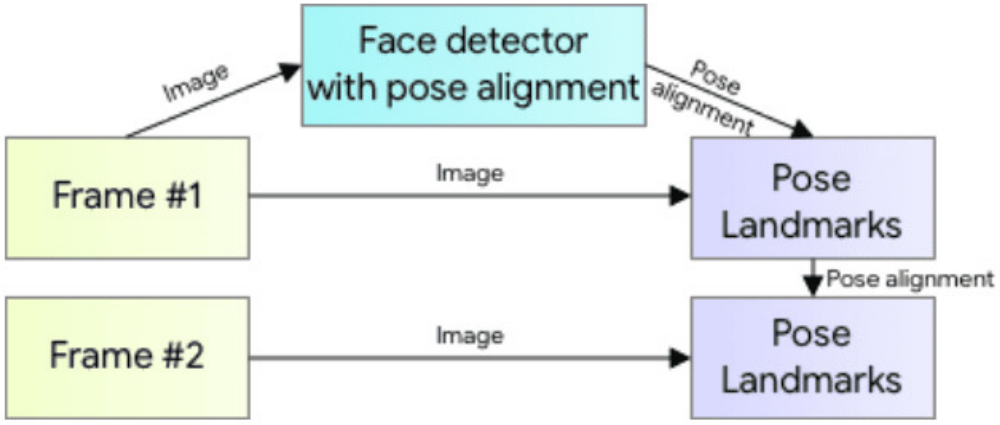


Figure 5: The inference pipeline proposed in BlazePose [21].

Bazarevsky *et al*. introduced a lightweight convolutional neural network architecture for human pose estimation called BlazePose [21]. It follows the top-down approach and is specifically made for mobile devices. It utilizes a detector-tracker inference

pipeline as shown in Figure 5, where the detector only needs to run until a person is detected, which then the tracker setup would take over.

Contrary to the common approach, which is to produce heat maps for each key point; they used a regression-based approach, a much less computationally demanding technique. As shown in Figure 6, their network uses an encoder-decoder architecture, in which two encoders would be utilized: one to predict heat maps for all joints, one to regress into all joints coordinates. Thus, by discarding the heat maps encoder during inference, they can have the heat map supervise the lightweight embedding for coordinates regression while training, while effectively reducing the computational cost afterward.

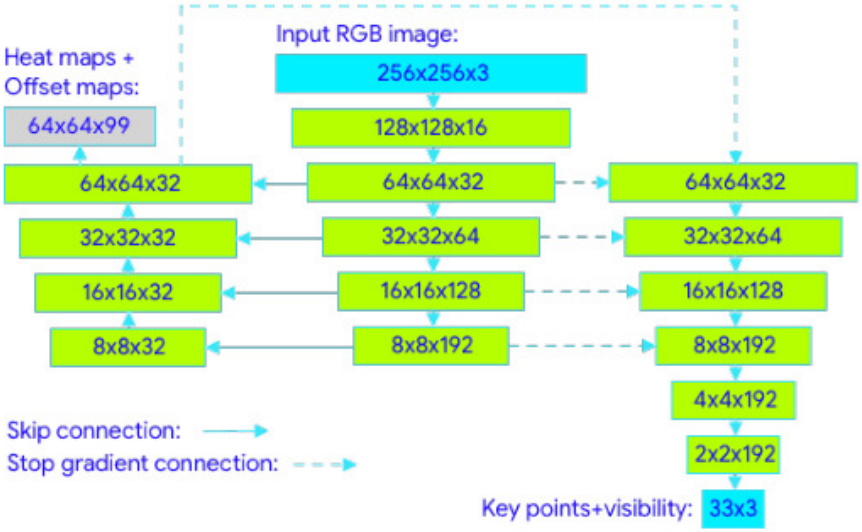


Figure 6: The proposed Pose estimation network of BlazePose [21]. Note that the heat map and offset loss (on the left) are only used during training to supervise and will be disabled later. During inference, the model would utilize only the keypoints from regression (on the right).

In the BlazePose paper, the authors claimed to have achieved a better accuracy-complexity tradeoff compared to OpenPose. Their lightweight architecture has allowed for real-time capabilities, combined with their device compatibility, making BlazePose favorable for mobile deployment. This aligned much with our interests.

2.2 Pose Estimation-Based Exercise Posture Correction

Our interest focuses on exercise posture correction methods using human pose estimation. The research status on exercise posture correction using this approach is not extensive. These systems are composed of two components: keypoints extraction and pose evaluation. The first component is meant to extract posture information, *i.e.* joint coordinates from the users. Thanks to the advance of human pose estimation research, nowadays it may only require running the input images through a pose estimation model. Model selection, however, depends on the situation, in the sense that highly accurate models are usually computationally expensive, and vice versa. The second component makes use of some methods to evaluate if the user's posture is correct, and gives adjustment guidelines accordingly. Usually, the methods and techniques used in the second component distinguish one system from another.

Steven Chen and Richard R. Yang's pipeline [22] uses OpenPose [15] for the first stage of the system. The output coordinates are two-dimensional, *i.e.* represented by a pair of numbers (x, y) corresponding to their location on the frame. The confidence level of each keypoint is tracked to see if the keypoint is being obscured. All length measurements between points are normalized using torso length to account for different possible distances from the user to the camera. For asymmetrical exercises, *e.g.* any one-arm lift, perspective ambiguity is resolved by measuring which parts of the body are more visible throughout all frames. Body vectors are calculated for all frames using these keypoints to construct a `Pose` object. Poses constructed across all frames are used to create a `PoseSeries` object. For evaluation, the authors use two approaches: geometric evaluation and machine learning evaluation. The geometric evaluation method uses relevant joint angles to judge if the posture is correct or not. Machine learning evaluation first requires collecting examples of correct and wrong series of poses. Then, for each example, dynamic time warp (DTW) is applied to the user's pose series to calculate an optimal matching between the two sequences. A nearest-neighbor classifier is used to classify the user's performance into right or wrong categories, where the distance metric is the DTW cost between two sequences. For machine learning evaluation, due to the lack of benchmark data for this particular problem, the authors recorded several videos for four exercises and measured precision, recall, and F1 scores for each exercise.

Geometric evaluation, while may require laborious work of identifying and implementing heuristics of interest, can be used to provide personal feedback to the user on how to correct their posture. Machine learning evaluation, on the other hand, is a data-driven approach. However, the approach using DTW and nearest-neighbor classifiers can only predict binary labels (right or wrong) without telling the user what

must be fixed in case the user performs the exercise wrong. The authors' pose estimation model of choice, OpenPose, while robust, is currently not available on mobile operating systems, though some workarounds can get it to run on mobile devices. The application also requires the user to record and trim the video of them performing the exercise themselves before analysis. This limits the portability of the application and is inconvenient for the user.

On this note, Ohri *et al*. [23] showed how to create a real-time, on-device system for pose estimation and correction. Testing procedures led them to choose BlazePose [21] as it was suitable for their use cases. For evaluation, the authors first recognized two categories of exercises: low-intensity and high-intensity. Low-intensity exercises involve little or no movement, *e.g*. plank and yoga poses. High-intensity exercises are those with faster movements. The authors realized that for low-intensity exercises, correcting the user's pose based on predefined angles, which is a way of exploiting geometric features of the exercise, is most suitable. On the other hand, the authors used DTW for high-intensity exercises. Instead of raising errors every frame, if an error persists after a long enough period, the error will be raised. This is also to avoid random errors caused by the pose estimation model's estimation.

Once again, DTW matching/distance calculation requires two complete sequences, thus the user will have to finish the exercise to get feedback from the application, *i.e*. the application can not correct user postures on-the-fly.

3 Methodology

On a high-level overview, our system consists of two main stages: Pose estimation and pose evaluation. The first stage consists of running the frames through a pose estimation model, then performing post-processing measures to partially negate model error. For the second stage, we batch adjacent frames, then use batch statistics to calculate a *state* of the exercise, and apply geometric evaluation guidelines to give live, real-time recommendations to the user. Figure 7 summarizes our system.

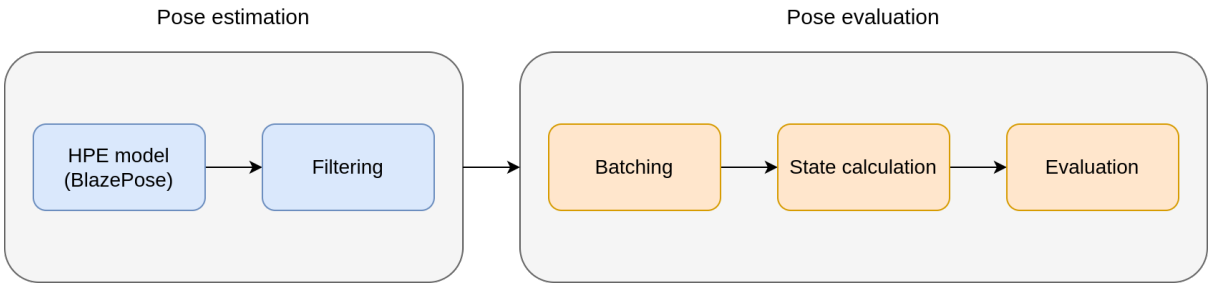


Figure 7: System overview.

3.1 Pose Estimation Method

3.1.1 Pose Estimation Model Selection

For the pose estimation model, as we want with state-of-the-art models at speed and accuracy, we narrowed our model selection to three candidates: OpenPose [15], Lite HRNet [19], and BlazePose [21]. While OpenPose is robust and can give more appropriate keypoints inferences from videos of movements, it lacks real-time performance capabilities and requires much more hardware resources than Lite HRNet or BlazePose. There are lightweight alternatives such as lightweight OpenPose [24] with a cost of accuracy. On the application side, the model lacks real-time support, and users would have to record videos before they can be evaluated.

Lite HRNet is more real-time capable than OpenPose and strikes a good balance between accuracy and performance than its predecessor, HRNet. It can achieve accept-

able performance in real-time, but its heatmap output requires post-processing for exact key point coordinates.

BlazePose is a lightweight CNN architecture designed specifically for mobile purposes. Due to the detector-tracker inference pipeline nature, the model can only perform single-person pose estimation and may be erroneous when two people or more are in a frame, resulting in worse performance and key points jumping. However, in our scenario, we only need to perform HPE for one person, thus this is acceptable. On the Google AR dataset, BlazePose performs slightly worse than OpenPose [21]. While experimenting, we also saw that BlazePose performs approximately six times faster than OpenPose, and almost three times faster than Lite HRNet. Furthermore, BlazePose is shipped in Google's Mediapipe, which provides optimized, easy-to-use APIs for multiple platforms and programming languages.

Overall, we decided to choose BlazePose as our human pose estimation model of choice due to the aforementioned performance capabilities and compatibility.

3.1.2 Filtering

BlazePose's accuracy is among the best for its speed, however, there are cases where the model can not identify the pose correctly. The main categories of error include:

- Most detected keypoints are wrong and clump up into a cloud-shaped figure of random keypoints.
- Several keypoints are missing although the corresponding part of the body is visible on camera.

To partially remedy this problem, we employ a preliminary pose-checking procedure: If the detected pose roughly matches a *normal* pose, we register it as a valid pose and move the pose to the next step. This step directly helps with the first problem. For the second problem, if the missing keypoints are not relevant to the evaluation of the pose correctness, they should be ignored. Thus, this similarity-checking procedure only applies to a certain set of keypoints of interest. Accumulating frames during a short period, which we later define as a *batch*, if the majority of frames contain invalid poses, an error should be raised to the user.

3.2 Pose Evaluation Method

3.2.1 Batching

For the next part of the system, we propose grouping every N adjacent frames into batches and then further processing in batch units based on the batch statistics. Prior methods include simple geometric evaluation that processes the input stream frame-by-frame can give evaluations and corrections in a real-time manner as the user performs the exercise. As pointed out by Ohri *et al*. [23], this is most suitable for *low-intensity* exercises. On the other hand, methods such as DTW are robust to exercises with lots of movement in which static rules may not fully cover the evaluation/correction procedure due to the variety of requirements for each part of the exercise. Our proposal of processing on batch units has the advantage of both worlds: we can give live feedback to the user (per batch) while being very flexible on the evaluation methods. Combined with determining a *state* variable for each batch, different geometric evaluation rule sets can be applied depending on the state or state transition. DTW reference sequences can be defined based on the exercise state, then can be applied for each batch.

Selecting N - the number of frames in a batch - is a heuristic based on several observations. A small N of 5 or less may result in too little information within a batch, which is not much more useful than just a frame. A large N of 30 or more, which typically transfers to more than one second of batch duration for 30 FPS cameras, results in too much information within a batch, that is, the batch sequence may contain multiple exercise stages. This may tamper with the state determination procedure and evaluation. We discuss more on the selection of N in Appendix A.

3.2.2 State Calculation

Batching frames together provides the basis for the flexibility of our system. Determining the state for each batch and applying evaluation methods based on state information enable this flexibility. In layman's terms, we define *state* as the answer to the question: *What part of the exercise is the user performing?*. For example, the shoulder press includes two main motions: lifting the weight and bringing the weight down. There can be two main states, namely up and down. A transition between these two states can be identified within extra states: top and bottom, corresponding to the location of the weight.

Determining the state of a batch, at the most basic level, can be done by calculating relevant joint angles. Furthermore, a batch contains information within a short period t. Other measures such as joint displacement, velocity, acceleration, etc. can be uti-

lized to calculate batch state. For example, measuring elbow or wrist displacement in a batch on the vertical axis can help determine the direction of the motion being performed. Concretely, let us denote $P_i = (x_i, y_i), 1 \leq i \leq M$ as the coordinates of M keypoints constructing a human pose P and $P^k, 1 \leq k \leq N$ as the poses within a batch of N frames. In general, the displacement of an arbitrary keypoint i within a batch is:

$$d_i = P_k^N - P_k^1$$

This is effective and useful in many state calculation situations, despite its simplicity.

It is possible to calculate the state of a batch differently. For example, we may treat batches as short sequences, create reference state sequences and apply DTW and a nearest-neighbor classifier to calculate the state. This approach comes with higher computational cost.

3.2.3 Evaluation

In our system, pose evaluation and correction are done using predefined sets of geometric guidelines for each exercise, depending on the batch state. For example, in the shoulder press, a correct movement includes keeping the forearm parallel to the body at all states. While the weight is at the bottom position, the elbow should be at least past the shoulder line. These can all be represented geometrically.

There are many benefits to using this approach. While it may require laborious work to implement custom sets of rules, it fits very well with the aim of the system. We are interested in not only telling the user if they are performing the exercise right or wrong but also giving very specific guidelines to correct the user's posture if something is wrong. For each exercise, there are mandatory requirements as well as some degree of flexibility on certain details. We should try to encourage the user to conform to the non-negotiable requirements and be less strict with others. We evaluate and correct key elements and key joint angles of exercises while allowing flexibility on some other less relevant factors.

Furthermore, the set of tools needed so far in the system would allow us to naturally extend this approach. Many parts of the system already rely on joint angle calculation, keypoint distance calculation, etc.

4 Implementation

We report implementation details of our work on the software side and the central, AI system side. The source code for our AI and software implementation is available at `https://github.com/tokudayo/aifa`.

4.1 Software

To demonstrate the application of our system, we have designed and implemented a web application and an Android application with the central component being our system. The software is designed with the following specifications:

- Windows 11
- Chrome 106.0.5249.119
- NodeJs 16.17.1
- Gstreamer 1.21.1
- React 18.2.0
- Android 12
- Android SDK 33
- WSL 2
- Python 3.10
- Visual Studio 2022
- Docker 20.10.17

We compose separate backend, web frontend, and AI containers in Docker. We use NodeJs' Buffer API to transfer images from the web client and the mobile app to the server. Other data transfers like pose estimation inference results, AI system feedback, etc. are transferred via JSON.

We provide illustrative diagrams in Figure 8, 9, 10, 11 on architectural details of our software implementation.

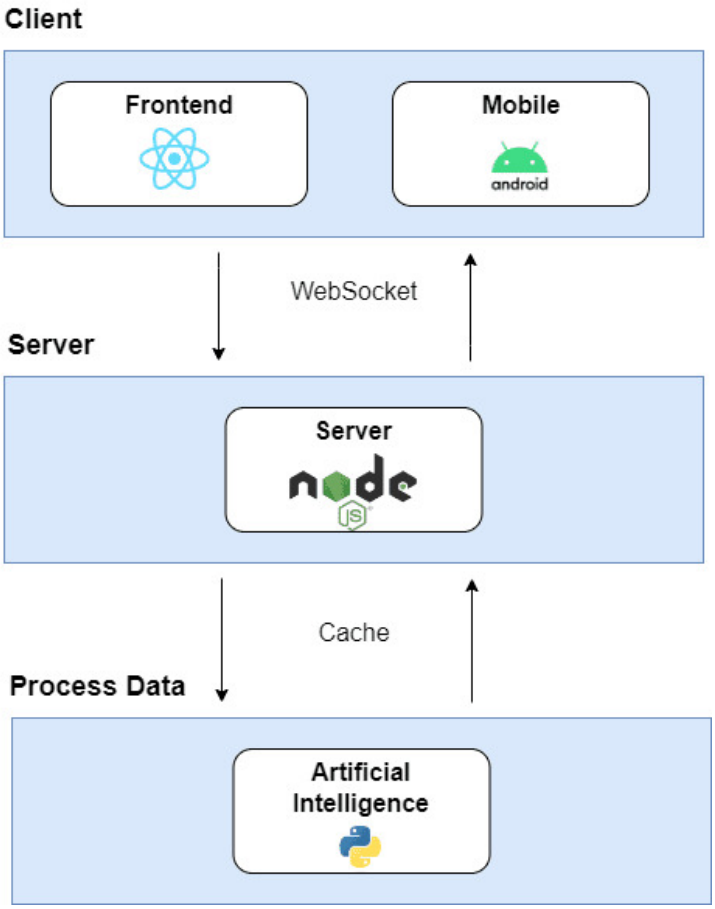


Figure 8: Overall architecture of our software.

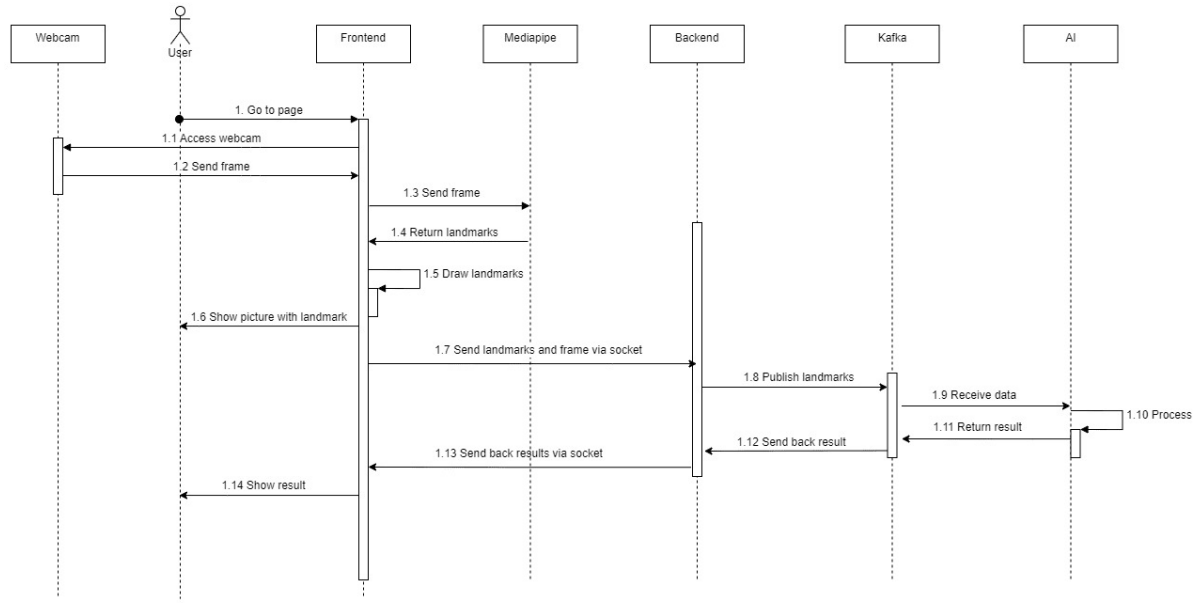


Figure 9: Architecture for webcam input on our web application.

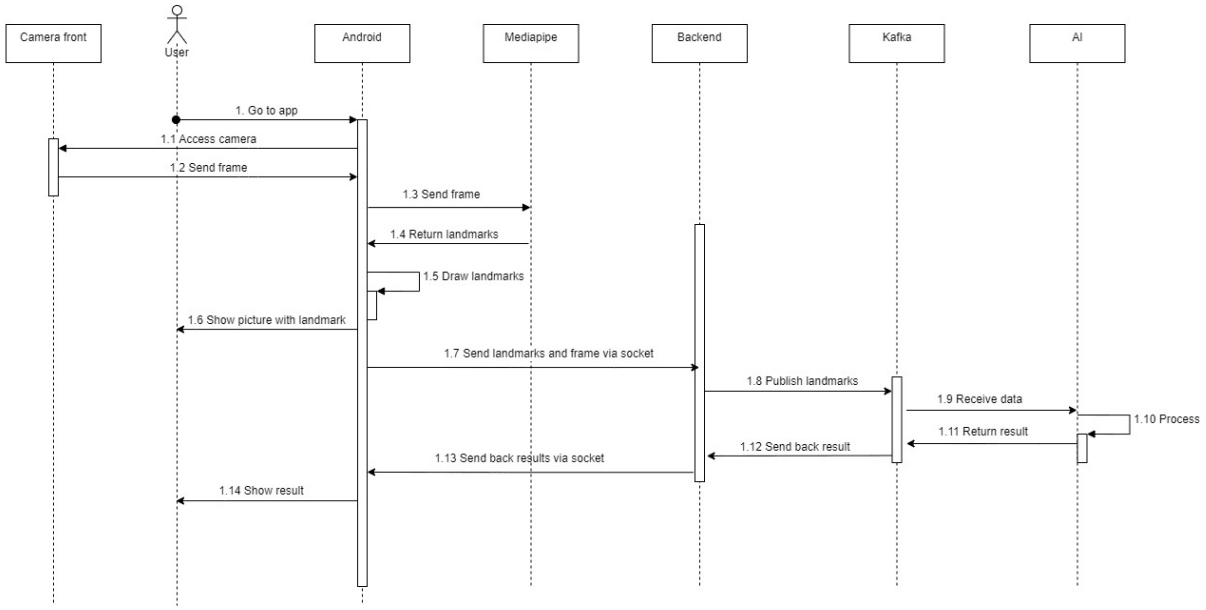


Figure 10: Architecture for mobile camera input on our Android application.

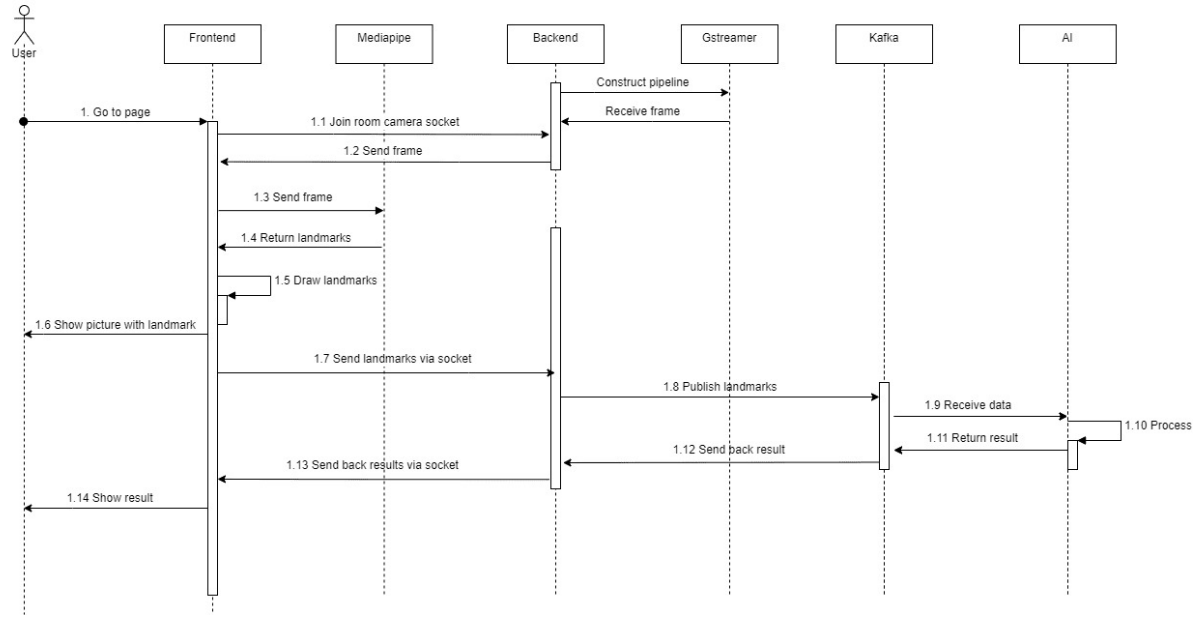


Figure 11: Architecture for some other sources of input (HTTP, RTSP streams, etc.), handled via Gstreamer.

4.2 AI System

We utilize Google's Mediapipe framework [25] that provides APIs for Android, JS, and more, which aligns with our interests. BlazePose [21] is a part of Mediapipe's ML solutions. The focus of the framework is ML solutions on CPUs, which are less expensive than GPUs, although GPU utilization is supported.

Our central AI system is implemented in Python and comes in form of a package. Most functions and utilities are built using Numpy. Normal installation would install basic dependencies for the system, as well as Mediapipe Python API for pose estimation and OpenCV for video streaming and processing. Lite installation excludes video processing and pose estimation capabilities.

For local testing, experiments, or building simple applications, we recommend using the full version, while for more sophisticated applications, we recommend using the lite version. Streaming and pose estimation should be handled by the backend of the application. This is for the best performance of the application.

In our applications, rather than sending the video frames to our AI system and letting the system handles everything, the backend is responsible for pose estimation, using Mediapipe API for either JavaScript or Android, depending on the platform. The extracted keypoints, the exercise the user is performing, and the resolution of the frames are sent to our Python AI system. This reduces data transfer and improves the latency of the application.

For each video frame, once we get the coordinates from the pose estimation model, we extract 18 out of 33 keypoints that represent the skeletal frame of the human body. These 18 keypoints are used to construct a `Pose` object.

To account for model errors, preliminary checking in the filtering step is performed by comparing the pose with a normal, expected pose. Concretely, we define a *normal pose* for each exercise as the pose in which the user would perform the exercise, excluding all keypoints that would be variable during the exercise. For many exercises, this would mean an upright pose. This detail requires implementing a `Pose.similarity` function. We measure the similarity between two poses by comparing joint angles between the two poses. For each joint, we calculate the absolute difference between the two angles in each pose. We take the maximum value across all locations and rescale it between 0 and 1. We keep poses with a similarity score to reference larger than 0.8.

After we register a valid pose, we extend a `PoseArray` object with the latest valid pose. `PoseArray`'s internal data holds a $M \times 18 \times 4$ array containing data of all valid poses from the start of a rep. We keep accumulating frames until the number of poses exceeds a threshold, at which point we discard the oldest poses to make place for new ones.

When a new, full batch containing N frames is obtained, we extract the batch into a smaller `PoseArray` object of size $N \times 18 \times 4$. This represents the batch based on which we try to calculate a state and make an evaluation.

All the steps described above are implemented in the `Exercise` class. The remaining steps are calculating the state of the exercise based on the latest batch, making an evaluation, and giving corrections if needed based on custom geometric evaluation rules. These details are different for every exercise. In our design, to define a new exercise, one needs to subclass the `Exercise` class and implement `Exericse.state` and `Exercise.evaluation`. We provide some utility functions and classes that can help implement new exercises, such as utilities in the `Pose`, `PoseArray`, `Vector`, and `VectorArray` classes, as geometric evaluation may use lots of vector maths. As we further implement detailed systems for three exercises discussed in Section 5, we discover some more reusable components and we add them to the package as well.

5 Experiment

5.1 Detailed Implementation for Three Exercises

We first further implement details in the state calculation and evaluation phase for three exercises: shoulder press, deadlift, and cross-body hammer curl and perform quantitative evaluation on these exercises. These are weightlifting exercises where the wrong form potentially results in joint injury and places tension on the wrong muscle group. This section shows details of the implementation for each chosen exercise based on our general framework, which includes our approach to state calculation and evaluation for that particular exercise.

There are several common components within all three exercises. We require a frontal view of the practitioner in all three exercises. We also keep track of an *upright* axis of the body. We define this as the line connecting either the left or right side of the shoulders and hips. Also, due to the varying distances the user may stay relative to the camera, we would have to measure movement significance by a unit of length relative to the body size. With the camera perspective requirements, we can use the length of either the line mentioned above or the line connecting the hips or shoulders, as they should be fully visible and stay mostly constant throughout the exercise. We chose the first option, although it may lead to a disadvantage that when the user tilts backward excessively, the line length can change and it also means the user is performing the exercise wrong.

Now, we define *significant movements within a batch* having displacement d as those which $d > rh; 0 \leq r$, where h is the torso length, and r is a hyperparameter denoting how sensitive we are to picking up movements. For all three exercises, we found some r around 0.1 to be practically sound, as the nature of what a *movement* is stays the same across exercises. Smaller r may cause model noises to be mistaken for movement, while large r are not useful in picking up movement because it requires excessive movement from the user in a short period. For the shoulder press, we experiment with r by measuring the accuracy of state detection with varying r. We selected three recorded videos of people performing at least one repetition of shoulder press and

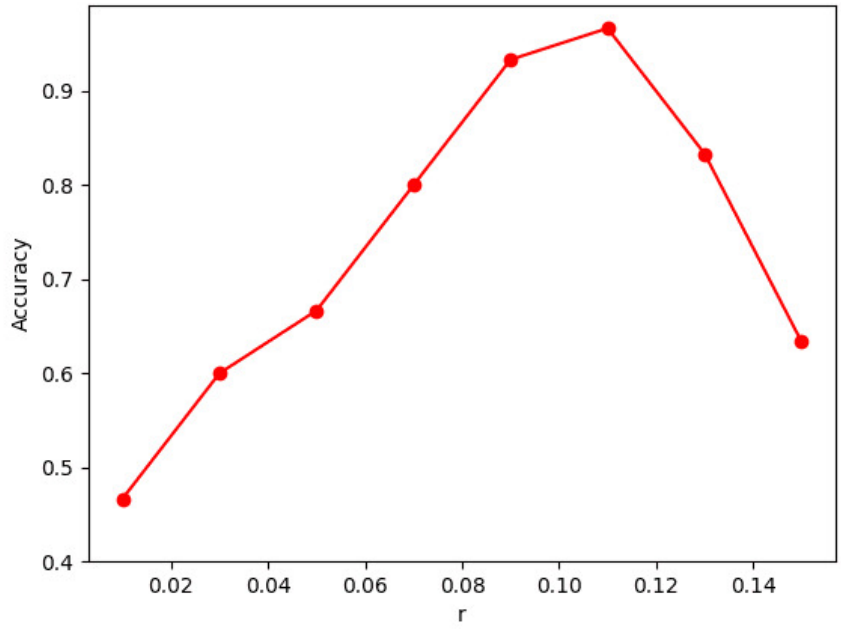


Figure 12: Effect of r on determining the batch state in our implementation of the shoulder press.

manually inspecting and labeling the desired state outputs after dividing the video into batches. Figure 12 demonstrates the accuracy of state classification with respect to r:

r also relates to the choice of N, the number of frames in a batch, which we discuss further in Appendix A. This is because a larger time period should generally allow a greater amount of movement.

5.1.1 Shoulder Press

The shoulder press is a weightlifting exercise that aims to strengthen the shoulder and upper back muscles. It can be performed in a seated position or a standing position, with either a barbell or a pair of dumbbells. In its starting position, the practitioner holds the weights slightly above shoulder height with their forearms parallel to the body, and the elbows slightly tucked forward for a wider range of motion. This exercise's movement involves pushing the weights upward until the arms are almost straight, while still keeping the forearm in the same parallel line to the body as fast as possible, then slowly bringing the weights down to the starting position in a controlled manner [26], [27].

(a) Acceptable form.

(b) Correct form.

Figure 13: Resting position of the arms; (a) acceptable and (b) correct.

State Calculation

During the exercise, it is clear that the arms are the main *moving* part. Based on this observation, for this exercise, we perform state calculation by tracking wrists' and elbows' movements. Concretely, we measure the average velocity and the overall displacement within a batch of these keypoints. If the movement is *significant* in the vertical direction relative to the body, it tells us whether the user is lifting the weights or bringing them down. A transition between these two *UP* and *DOWN* state tells us that the weights have reached the top or bottom positions. There are also cases where the movement is not significant enough. Typically, this can mean two things: the user is either struggling to make further progress lifting the weight or resting between reps. This *STATIC* state can also tell us more about the actual state of the exercise, for example, if the measured states are *UP* to *STATIC* to *UP*, it means the user was just struggling to lift the weights, however, if they are *UP* to *STATIC* to *DOWN*, or some similar forms, it means there is a movement transition in the exercise.

Evaluation

For this exercise, we measure the line connecting the two shoulders and the two hips to make sure the body stays straight all the time, as tilting the body can induce injuries while performing the shoulder press. The two lines should stay relatively parallel during the exercise duration. We allow a 10-degree deviation from the parallel position.

Also, while performing the shoulder press, the forearm should stay parallel to the body at all times, or tilted inward just slightly. Tilting the forearms outward not only makes the exercise harder but also introduces great injury risks. We allow a 15-degree deviation inward and a 5-degree deviation outward.

Furthermore, when the weights reach the top or the bottom, we check for arm positioning. According to most guidelines, it is recommended to lift the weights to the top. This means a close-to-parallel arm position. Thus, we check if the user extends their arms fully by measuring the forearm angle with respect to the x-axis at this position. We expect a 90-degree angle with a 30-degree deviation. Not meeting these requirements means either the user should try harder for the full muscle-building effect or the weights is too heavy for their current strength/stamina and that the user should stop the set or switch to lighter weights. At the bottom position, the elbows should go all the way to the lower chest level. However, it is also partially acceptable that the elbows only reach shoulder level, *i.e.* the arms are perpendicular to the body, although this can induce shoulder joint stress and is limiting the full range of shoulder motion. This is visually demonstrated in Figure 13. We accept the user's performance if the y-coordinates of the elbows are shoulder level, while still recommending them to tuck the elbows forward and go lower if the angle of the arms with respect to the body is larger than 45 degrees.

5.1.2 Deadlift

The deadlift is a powerlifting exercise that involves lifting a barbell to hip level while keeping the body perpendicular to the ground. It is referred to as a compound exercise, which strengthens multiple muscle groups at the same time, mainly in the uppers and lowers back, glutes, and hamstrings muscles. The deadlift is usually divided into two main styles: sumo deadlift and conventional deadlift [28], [29]. We will be focusing on the more popularly used style - the conventional deadlift style. To perform a deadlift, one usually follows the five-step setup [30], as shown below:

- Get into position: stand so that the feet are under the bar while keeping them not too wide or close - just comfortable enough to jump. Body straight, arms relaxed.
- Grip the bar at a position just outside of the legs.
- Bend the knees so that the shins touch the bar, and the knees touch the arms.
- Pump the chest outward so that the back can be flat.
- Drag the bar up the shins and thighs.

New performers usually run into common issues such as not keeping the legs or arms

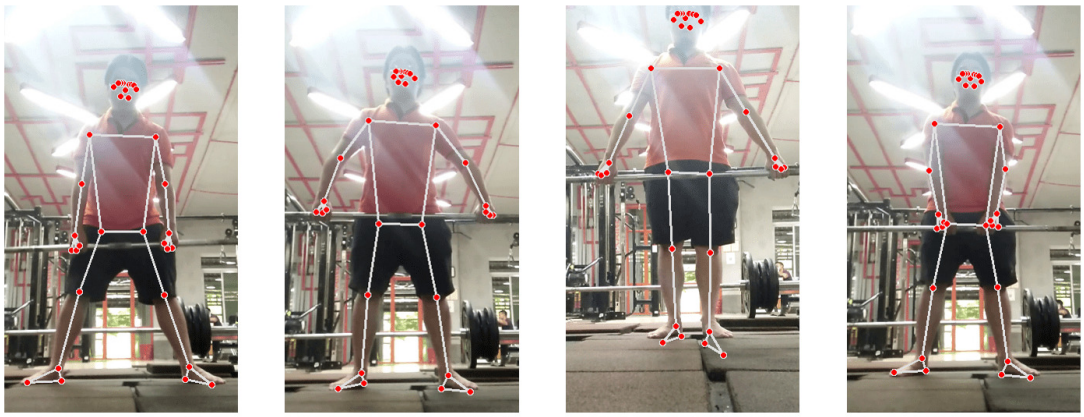


Figure 14: Common mistakes when doing a deadlift. (a) Stance too wide. (b) Grip too wide, arms are not kept straight. (c) Stance too narrow and grip too wide - note that this is still correct for the Romanian deadlift style. (d) Grip too narrow.

in the correct position, which can drastically hinder the performance of the exercise. If one was to lift in an uncomfortable and painful position often, it might even affect the performer's health. Another issue is dropping the hip and not keeping the back straight when dragging the bar up, which can negatively affect the efficiency of the exercises: the performer has to lift the barbell in a more difficult path. It can also injure the performer's back as more pressure is put on the back when it is already in a bad position [31]. One should also keep their body as stationary and straight as possible during the exercise so as to avoid potential accidents where the barbell can drag them down with its weight and momentum.

State Calculation

To help users correct their errors, we track their performance in the frontal view with the full body - from feet to head - in the camera. We made sure that the body is always straight and stationary. The main parts of the body in motion are the arms, which directly lift the body up. Thus, we calculate the state of exercise by measuring the y-displacement of the two wrists in a batch: if it surpasses the threshold, the movement is significant and counts as a state change.

Evaluation

To make sure the performers get in their correct stance - they can not be too wide nor too close - we calculated the stance width by measuring the distance between the two ankles as shown in Figure 15. Because the distance from the user to the camera

Figure 15: Ensuring correctness of deadlift stance and grip through width comparison with the hip width. Yellow, blue, and green lines represent the hip, grip, and stance width respectively. Adapted from [32].

can vary, a fixed length would not be suitable, and thus we used the hip width to dynamically constrain the stance width. It can not be greater than one and a half the width of the hip, while still greater than half the hip so that they are not too close. The same procedure was also used to ensure the correctness of the arm position: the grip can not be lower than half the width of the hip, nor greater than one and a half the hip of the performer.

During the lift-up process, we make sure the arms stay as straight as possible to ensure the performers are not lifting incorrectly, *e.g.* bending the elbows. We define two vectors in particular: the shoulder-elbow vector, and the shoulder-wrist vector as shown

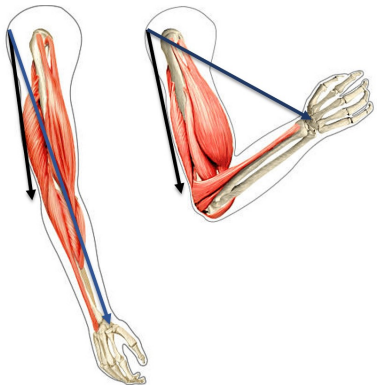


Figure 16: Determining the straightness of the arms through the angle between the shoulder-wrist vector and the shoulder-elbow vector. (a) The angle is small, thus the arm is straight. (b) The angle is big, so the arm is not straight.

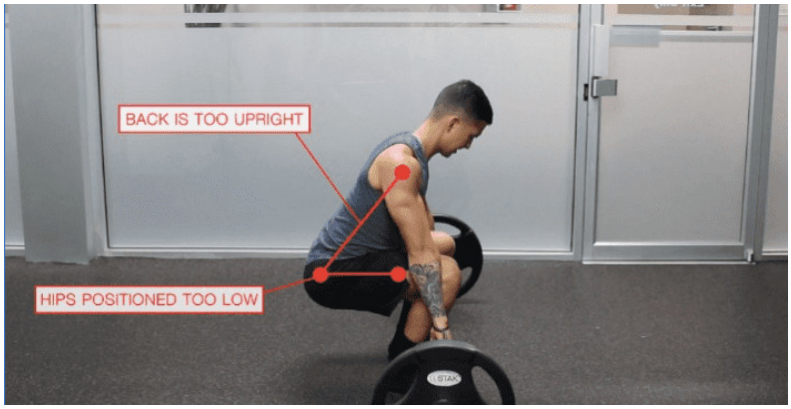


Figure 17: Common mistakes with hip positioning [28]. Performers should avoid dropping the hip too low to avoid overstraining the lumbar spinal structures.

in Figure 16. By allowing arms angles less than 10 degrees, we can ensure the arms are always straight.

Another mistake that performers often made during the deadlift is to drop their hip too low when lifting or dropping as shown in Figure 17, which can impact their performance and health severely [31]. The ideal hip position is just low enough for the user to comfortably grip the barbell, without putting the user in the squat stance [29]. It must also not be too high, as it can put too much strain on the user's back [31]. In our case, with the limitation of the 2D plane, we utilized the y position of the hips. By making sure that it does not pass the knees, we can detect if the user dropped their hip or not.

5.1.3 Cross-body Hammer Curl

The cross-body hammer curl is a dumbbell exercise that targets the biceps, brachialis, and forearm muscles where one lifts the weights across the torso. To perform the exercise, one first stands up straight with a dumbbell in each hand. While keeping the palms facing the body and without twisting the arm, alternatively curl the dumbbell of each arm up towards the opposite shoulder. At the end of the positive motion, the top of the dumbbell should be close to or touch the shoulder. After holding for about a second, slowly lower the dumbbell along the same path [33].

Common mistakes while performing this exercise may include not bringing the weights up high enough, resulting in less muscle tension and contraction. While it feels easier, this induces less muscle-building effect. This limited range of motion can also be a sign of inadequate strength while lifting, the best response to which is to either stop the set or switch to lower weights. Another mistake is to use excessive momentum from the shoulders and shoulders movement to help with the lifting. This shifts the tension away from the biceps, which is the main target of hammer curl, and can cause

(a) Incorrect form.

(b) Correct form.

Figure 18: Two examples of a hammer curl; (a) incorrect and (b) correct.

shoulder injuries, as the shoulders rotator cuff and joint are very susceptible to injury. The overall body frame should be stationary and solid during the exercise.

State Calculation

To help users perform the exercise correctly, we first require a frontal view of the body while performing the exercise. We identify the parts of the body in motion during the exercise to be both forearms and arms of the exercise. Due to the asymmetrical nature of the exercise, for state calculation, we first need to identify which arm is performing the lift. To achieve this, we measure and compare the displacements of both arms within a batch. The arm with more significant movement during the batch is likely to be the one performing the exercise. If the y-displacement of this arm exceeds a threshold, we register it as moving up, or down. During this exercise, both arms cannot perform the lift at the same time, thus, for example, if we identify the left arm to be lifting the weights, we may associate the batch with the state *L_UP*. This tells us that the left arm is moving up, and the right arm must stay still. There are, of course, cases where both arms are stationary in a batch. This does not mean neither arm is performing the exercise. Again, we take previous states into account to correctly realize what part of the exercise the user is currently at.

Evaluation

Similar to the shoulder press, we measure the lines from both hips to shoulders and constantly check to make sure the body stays straight all the time. Within each batch, we monitor shoulder movement to prevent the use of excessive shoulder strength and momentum to lift the weights. Concretely, we measure the shoulders tilting in the lifting direction, using their displacement and angle change as heuristics. During the exercise, both arms should stay relatively parallel to the body. We allow a 10-degree deviation from the parallel position. The forearms perform the main movement of the exercise, are crucial for state calculation, and need to follow several principles to be classified as correct. At the end of the lifting, *positive* motion, the weights should be in front of the upper chest. This means the angle between the forearm and the arm should be around 45 degrees. Much higher than which means an inadequate range of motion, while much lower than which means the user is performing the lift excessively without additional muscle-building effects. Figure 18 demonstrates this visually.

During the *negative* motion, it is against the guidelines to drop the weights too fast. It is recommended that the practitioner lowers the weights gradually over one to two seconds. If we measure significant downward motion within less than one second, which corresponds to three batches in our implementation, we would raise a warning to the user. When the weights reach the bottom position, the arms should be fully extended, without any tension other than carrying the weights. In this natural position, the arms should be parallel to the body, thus we perform a check here and give recommendations to the user if needed.

5.2 Experimental Setup

5.2.1 Hardware and Software

We evaluate the performance of the system and benchmark its speed on both high-medium-end and low-medium-end machines with the following specifications:

Specification	**System A**	**System B**
OS	Ubuntu 22.04	Windows 10
CPU	Ryzen 7 4800H	Intel i7-8565U
RAM	16GB	8GB
Python version	3.9.10	3.8.5

Table 1: System specifications.

5.2.2 Data

We provide testing results on the three exercises we implemented based on our system. Due to the lack of benchmark data for this specific area of research, for each exercise, we first try to find online videos that satisfy our requirements (camera setup, view angle, etc.). Furthermore, we record several videos of people performing the exercise. We are interested in knowing if wrong exercise forms are corrected and if correct forms are constantly approved throughout the exercise duration. Each repetition is counted as one example. The ground truth label for each example is obtained based on whether they conform to various public, online guidelines. The dataset is available on request.

5.2.3 Metrics

We define *wrong posture* as the positive class and *correct posture* as the negative class. We measure the system's performance using three metrics: accuracy, precision, and recall. The accuracy of the system is defined as:

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}$$

Where TP is the number of true positives, TN is the number of true negatives, FP is the number of false positives, and FN is the number of false negatives.

The precision measures the percentage of correct wrong form predictions over total wrong form predictions and is mathematically represented as:

$$\text{Precision} = \frac{TP}{TP + FP}$$

Finally, the recall is defined as:

$$\text{Recall} = \frac{TP}{TP + FN}$$

5.3 Results

Table 2 shows measured relevant metrics for these three exercises.

Exercise	Examples	Accuracy (%)	Precision	Recall
Shoulder press	22	95.5	0.91	1
Deadlift	42	90.4	0.95	0.86
Hammer curl	20	95	1	0.9

Table 2: Accuracy, precision, and recall of our implemented system on three exercises.

Qualitatively, for the shoulder press, common errors are always detected by our system. The pose estimation model also makes fewer mistakes that are detrimental to the performance of the overall system, likely due to the facts that all body parts are visible and do not obscure each other during the exercise, and that the weights do not obscure any parts of the body during the exercise.

For the deadlift, during our evaluation of the system's performance, the system showed good capabilities, being able to correctly identify most of the common errors: bad stance, incorrect grip, hand placement, and hip dropping.

The system agrees with cases where the hammer curl is performed correctly and picks up on common errors: not bringing the weights up fully to the chest, not bringing the weights down to a relaxed position, bent arms, and excessive body tilting. However, backward tilting of the shoulders is hard to detect, partly due to the pose estimation model limitations and errors. Theoretically, if the pose estimation model gives perfect estimation and tracking of a key shoulder part, the minor length contraction of the hip-shoulder line when tilting backward can be a pick-up sign. However, this is impractical, at least as of now. Another problem is for this particular exercise, when the weights are lifted close to the chest, part of the weights might obscure the other stationary hand. This causes model estimation error and is very common when recording from angles other than the frontal view, thus it is more important for this exercise that the users record themselves facing the camera directly.

The key to extending our system for custom exercises is to identify the characteristics of different stages of an exercise. For the three exercises we implemented in detail, the stages are characterized by the motions of the lift. If done correctly, the state of the exercise can be calculated with high accuracy. Evaluation guidelines can be adjusted accordingly. This enables detailed, fine-grained evaluation of the practitioner's pose,

as demonstrated in our results. The system shows high accuracy, precision, and recall in identifying erroneous poses. Our approach also has several advantages over existing methods. For example, what the cross-body hammer curl differs from the other two exercises we implemented based on our framework is that this exercise is asymmetrical in the sense that only one arm can perform the exercise at a time, thus there is a need to first detect which arm is lifting the weights. Implementing this is natural in our framework, whereas methods such as DTW may give suboptimal sequence alignment if the user lifts the weights with only one arm, or in general lifting in an order different from the reference sequence, and not being able to correct the posture in a real-time. Also, given roughly the same poses in different snapshots during the exercise, our method can differentiate them through batch characteristics such as the direction of movement, velocity, stages, etc. A useful application of this is to differentiate between upward and downward motion while lifting. Many weightlifting exercises require lifting the weights as fast as possible while lowering the weights gradually. Our approach can detect them and give guidelines based on the state of the exercise, whereas methods that only take the information of a single frame into account can not.

Performance-wise, it is important to acknowledge that as with most posture correction systems using human pose estimation models, the speed and resource consumption bottleneck lies within the pose estimation model. We chose BlazePose as the pose estimator due to its speed and good speed-accuracy tradeoffs. This plays an important role in enabling the real-time capabilities of our system. The components of the system that follow are also designed with speed in mind. Overall, the system achieves 35 FPS at 720×504 on machine A and 25 FPS at 720×504 on machine B. As the pose estimation model use can be changed, when leaving out the model cost, the rest of the system in our implementation runs at 1350 FPS on system A and 600 FPS on system B.

6 Conclusion

This thesis presents a human pose estimation-based system for real-time exercise posture correction and implement in details three weightlifting exercises: shoulder press, deadlift, and cross-body hammer curl. The approach processes input frames in batch units and is able to give live corrections to users.

This thesis also demonstrate the practicality of the work by implementing the system in Python, as well as building applications for web users and Android devices based . The applications take input from the device camera or webcam and can give feedback and guidance to the user on-the-fly.

There are limitations in this thesis that could be addressed in future work. The system relying on 2D estimations of body keypoints introduces several limitations: (1) the system requires camera perspectives that minimize the impact when lacking keypoints depth information; (2) this also means that the system is only suitable for exercises with few movements depthwise. Another limitation is while developing the system and implementing the three exercises, we have not fully explored the potential techniques that help filter out erroneous poses or calculate the batch state. Finally, we report the results in Section 5 on our data due to the lack of benchmark data in research, thus these results must be interpreted with caution.

References

[1] Matthew A. Nystoriak and Aruni Bhatnagar. "Cardiovascular Effects and Benefits of Exercise". In: *Frontiers in Cardiovascular Medicine* 5 (Sept. 2018). DOI: 10.3389/fcvm.2018.00135. URL: https://doi.org/10.3389/fcvm.2018.00135.

[2] Wayne L. Westcott. "Resistance Training is Medicine". In: *Current Sports Medicine Reports* 11.4 (2012), pp. 209–216. DOI: 10.1249/jsr.0b013e31825dabb8. URL: https://doi.org/10.1249/jsr.0b013e31825dabb8.

[3] Tyson Grier, Raina D. Brooks, Zack Solomon, et al. "Injury Risk Factors Associated With Weight Training". In: *Journal of Strength and Conditioning Research* 36.2 (Aug. 2020), e24–e30. DOI: 10.1519/jsc.0000000000003791. URL: https://doi.org/10.1519/jsc.0000000000003791.

[4] Ulrika Aasa, Ivar Svartholm, Fredrik Andersson, et al. "Injuries among weightlifters and powerlifters: a systematic review". In: *British Journal of Sports Medicine* 51.4 (2017), pp. 211–219. ISSN: 0306-3674. DOI: 10.1136/bjsports-2016-096037. eprint: https://bjsm.bmj.com/content/51/4/211.full.pdf. URL: https://bjsm.bmj.com/content/51/4/211.

[5] Steven R McClaran. "The effectiveness of personal training on changing attitudes towards physical activity". en. In: *J Sports Sci Med* 2.1 (Mar. 2003), pp. 10–14.

[6] Haoming Chen, Runyang Feng, Sifan Wu, et al. *2D Human Pose Estimation: A Survey*. 2022. DOI: 10.48550/ARXIV.2204.07370. URL: https://arxiv.org/abs/2204.07370.

[7] Shubham Sharma, Shubhankar Verma, Mohit Kumar, et al. "Use of Motion Capture in 3D Animation: Motion Capture Systems, Challenges, and Recent Trends". In: *2019 International Conference on Machine Learning, Big Data, Cloud and Parallel Computing (COMITCon)*. Feb. 2019, pp. 289–294. DOI: 10.1109/COMITCon.2019.8862448.

[8] Matteo Menolotto, Dimitrios-Sokratis Komaris, Salvatore Tedesco, et al. "Motion Capture Technology in Industrial Applications: A Systematic Review". In: *Sensors* 20.19 (2020). ISSN: 1424-8220. DOI: 10.3390/s20195687. URL: https://www.mdpi.com/1424-8220/20/19/5687.

[9] Lars Mündermann, Stefano Corazza, and Thomas P. Andriacchi. "The evolution of methods for the capture of human movement leading to markerless motion capture for biomechanical applications". In: *Journal of NeuroEngineering and Rehabilitation* 3.1 (Mar. 2006), p. 6. ISSN: 1743-0003. DOI: 10.1186/1743-0003-3-6. URL: https://doi.org/10.1186/1743-0003-3-6.

[10] Mykhaylo Andriluka, Stefan Roth, and Bernt Schiele. "Pictorial structures revisited: People detection and articulated pose estimation". In: *2009 IEEE Conference on Computer Vision and Pattern Recognition*. June 2009, pp. 1014–1021. DOI: 10.1109/CVPR.2009.5206754.

[11] Yi Yang and Deva Ramanan. "Articulated pose estimation with flexible mixtures-of-parts". In: *CVPR 2011*. June 2011, pp. 1385–1392. DOI: 10.1109/CVPR.2011.5995741.

[12] Alexander Toshev and Christian Szegedy. "DeepPose: Human Pose Estimation via Deep Neural Networks". In: *2014 IEEE Conference on Computer Vision and Pattern Recognition*. June 2014, pp. 1653–1660. DOI: 10.1109/CVPR.2014.214.

[13] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. "ImageNet Classification with Deep Convolutional Neural Networks". In: *Proceedings of the 25th International Conference on Neural Information Processing Systems - Volume 1*. NIPS'12. Lake Tahoe, Nevada: Curran Associates Inc., 2012, pp. 1097–1105.

[14] Ce Zheng, Wenhan Wu, Chen Chen, et al. *Deep Learning-Based Human Pose Estimation: A Survey*. 2020. DOI: 10.48550/ARXIV.2012.13392. URL: https://arxiv.org/abs/2012.13392.

[15] Zhe Cao, Gines Hidalgo, Tomas Simon, et al. *OpenPose: Realtime Multi-Person 2D Pose Estimation using Part Affinity Fields*. 2018. DOI: 10.48550/ARXIV.1812.08008. URL: https://arxiv.org/abs/1812.08008.

[16] Kaiming He, Georgia Gkioxari, Piotr Dollár, et al. *Mask R-CNN*. 2017. DOI: 10.48550/ARXIV.1703.06870. URL: https://arxiv.org/abs/1703.06870.

[17] Meng'An Shi, Huimin Cai, and Yang Gao. "Optimization of Human Pose Detection Based on Mask RCNN". In: *2021 2nd International Symposium on Computer Engineering and Intelligent Communications (ISCEIC)*. Aug. 2021, pp. 273–277. DOI: 10.1109/ISCEIC53685.2021.00064.

[18] Jingdong Wang, Ke Sun, Tianheng Cheng, et al. *Deep High-Resolution Representation Learning for Visual Recognition*. 2019. DOI: 10.48550/ARXIV.1908.07919. URL: https://arxiv.org/abs/1908.07919.

[19] Changqian Yu, Bin Xiao, Changxin Gao, et al. *Lite-HRNet: A Lightweight High-Resolution Network*. 2021. DOI: 10.48550/ARXIV.2104.06403. URL: https://arxiv.org/abs/2104.06403.

[20] Sudhakar Kumawat, Gagan Kanojia, and Shanmuganathan Raman. *ShuffleBlock: Shuffle to Regularize Deep Convolutional Neural Networks*. 2021. DOI: 10.48550/ARXIV.2106.09358. URL: https://arxiv.org/abs/2106.09358.

[21] Valentin Bazarevsky, Ivan Grishchenko, Karthik Raveendran, et al. *BlazePose: On-device Real-time Body Pose tracking*. 2020. DOI: 10.48550/ARXIV.2006.10204. URL: https://arxiv.org/abs/2006.10204.

[22] Steven Chen and Richard R. Yang. *Pose Trainer: Correcting Exercise Posture using Pose Estimation*. DOI: 10.48550/ARXIV.2006.11718. URL: https://arxiv.org/abs/2006.11718.

[23] Ashish Ohri, Shashank Agrawal, and Garima S. Chaudhary. "On-device Real-time Pose Estimation & Correction". In: *International Journal of Advances in Engineering and Management* 3 (2021). DOI: 10.35629/5252-030716911696. URL: https://ijaem.net/issue_dcp/On%20device%20Realtime%20Pose%20Estimation%20and%20Correction.pdf.

[24] Daniil Osokin. *Real-time 2D Multi-Person Pose Estimation on CPU: Lightweight OpenPose*. 2018. DOI: 10.48550/ARXIV.1811.12004. URL: https://arxiv.org/abs/1811.12004.

[25] Camillo Lugaresi, Jiuqiang Tang, Hadon Nash, et al. *MediaPipe: A Framework for Building Perception Pipelines*. 2019. DOI: 10.48550/ARXIV.1906.08172. URL: https://arxiv.org/abs/1906.08172.

[26] bodybuilding.com. *Seated dumbbell shoulder press*. URL: https://www.bodybuilding.com/exercises/dumbbell-shoulder-press.

[27] Nick Harris-Fry. *How To Do A Dumbbell Shoulder Press*. URL: https://www.coachmag.co.uk/shoulder-exercises/6134/how-to-do-a-dumbbell-shoulder-press.

[28] Jeremy Ethier. *Deadlifts: Which Type is BEST For You?* 2019. URL: https://builtwithscience.com/fitness-tips/types-of-deadlifts/.

[29] Jason M Cholewa, Ozan Atalag, Anastasia Zinchenko, et al. "Anthropometrical Determinants of Deadlift Variant Performance". en. In: *J Sports Sci Med* 18.3 (Aug. 2019), pp. 448–453.

[30] digitalbarbell.com. *FIVE STEP DEADLIFT SETUP*. URL: https://www.digitalbarbell.com/blog/2018/7/23/five-step-deadlift-setup.

[31] Victor Bengtsson, Lars Berglund, and Ulrika Aasa. "Narrative review of injuries in powerlifting with special reference to their association to the squat, bench press and deadlift". en. In: *BMJ Open Sport Exerc. Med.* 4.1 (July 2018), e000382.

[32] Joel Snape. *How To Deadlift: Your Expert Guide To The King Of Lifts*. URL: https://www.coachmag.co.uk/barbell-exercises/3725/how-to-deadlift.

[33] bodybuilding.com. *Cross-body hammer curl*. URL: https://www.bodybuilding.com/exercises/cross-body-hammer-curl.

Appendix

A On the Number of Frames in a Batch N

In this section, we provide the rationale for choosing the number of frames within a batch, or the batch size N. We do not provide a concrete, mathematically-proven method for selecting the best N, that is, it is mostly a heuristic. We recognize two important factors playing a role in determining N, that is model capability and the nature of human movement.

Model capability changes how we would select N. In practice, real-time models like BlazePose have taken the trade for speed in the speed-accuracy tradeoff [21]. That is to say, even in cases where the model correctly identifies body keypoints, there are noises while it tracks the keypoints across multiple frames. To see why this may be a problem, note that the definition of a keypoint is not absolute. For example, for a single image, a valid, correct keypoint detection for the left shoulder can be anywhere within a small region of the left shoulder rather than an exact point. Across multiple frames, subsequent keypoints should still point to the left shoulder, but might not be at the same previous spot on the body. In movement tracking within a batch, if the batch size is too small concerning some function of the model capability, the movement may be definite but not significant enough to overcome the noise in the model. There are other factors that affect the model performance like camera quality, lighting, etc. that can in turn indirectly affect our choice of N. To sum up, a very accurate and consistent human pose estimation model with less noise would allow the batch size to be smaller.

The other factor, the nature of human movement, plays a role in the sense that for state calculation, we would want to pick up characteristics traits, including movement traits, like pulling the weights up or extending the arms to the side. They compose of short periods of time where the movement is *singular*. We aim to make sure relevant movements in a batch are singular so that it is easy and computationally cheap to detect. Thus, if the batch size is large, the batch may be comprised of multiple singular movements that should correspond to multiple stages. As we can handle the unavoid-

able case where the batch contains the transition between two states, the restriction should now correspond to no more than a singular movement or a state transition. On the other hand, if the batch size is too small, there is not enough conclusive evidence to be certain that parts of the body are moving in a definite direction. In the scenario where the estimation model accuracy is perfect, the lower bound for a good batch size would still be decided by this factor. Overall, we want neither a batch size too large that may comprise of multiple movements/phases of the exercise nor a batch size too small that there is not enough information to reasonably calculate the batch state if it relies on movement detection.

In cases where we do not care about sequential information of the body pose at all, the batch size can be set to one, which reduces to the single-frame method. This is suitable for static, low-intensity exercises like yoga poses as pointed out in [23].

We optimize the hyperparameter N by first reasoning that we want to select the minimum N that satisfies the conditions mentioned above. We want N to be as small as possible because processing shorter time frames means we can give immediate feedback to the user, and it should naturally satisfy the upper bound constraint. We now need to select N so that the movements are significant within that time frame and are significant enough that model noises do not dampen it down so that it is undetectable. Heuristically, we found N corresponds to time $t \in [0.3, 0.4]$ to be practically sound. Thus, for 30 FPS cameras, we would select $N = 10$, and for 60 FPS cameras, we choose $N = 20$, and so on.

B Implemented Applications

B.1 Web Application

Our web application can be accessed via `https://aifa.one/`. Once entering, one will be first prompted to log in. The login screen is as shown in Figure 19. Users can proceed to enter their registered email and password to log in or register following the guidelines displayed under the login button.

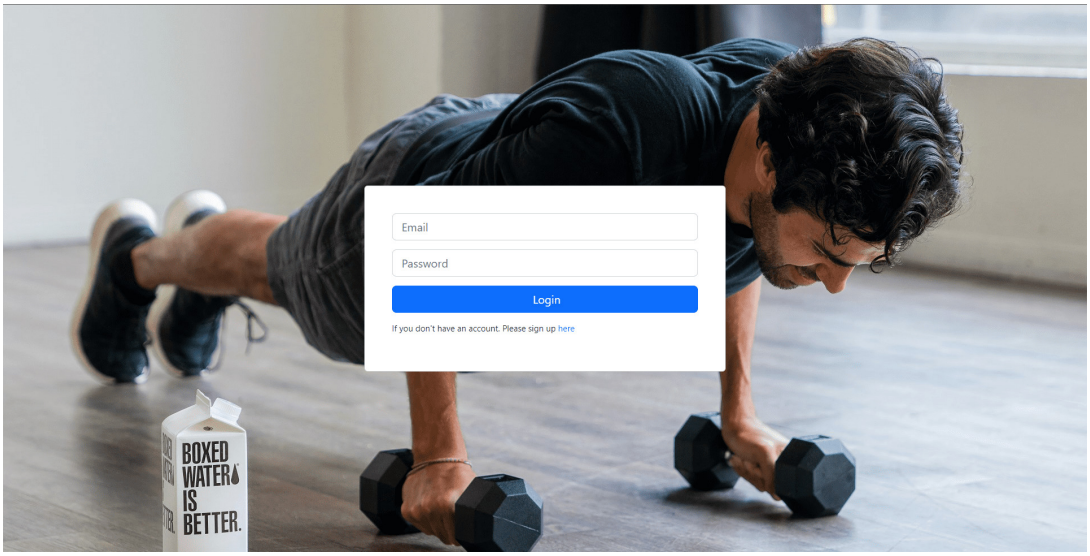


Figure 19: The login screen of our web application.

After logging in, the user will be directed to a page where they can choose one of the three exercises we have implemented using our system. In the drop-down menu shown in Figure 20, the user can pick from a range of three exercises: shoulder press, deadlift, and hammer curl. After choosing one, they can start a session through the *Start streaming* button.

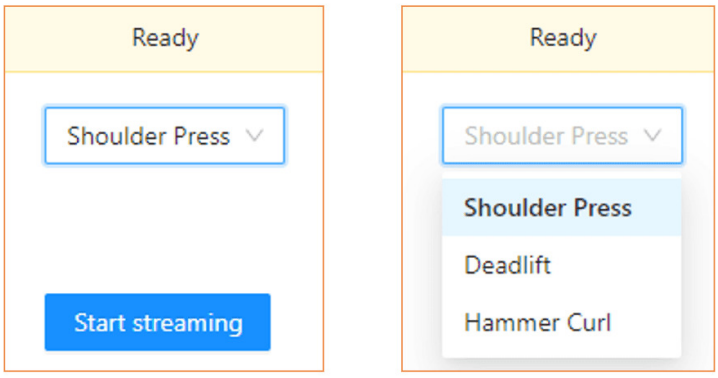


Figure 20: The drop-down menu where the user can choose an exercise to perform and get corrections.

The web application will use either the built-in webcam of the user's device or a connected webcam of the device as the input video stream source. After initializing a session, the user can start performing the exercise. An example of the GUI is shown in Figure 21.

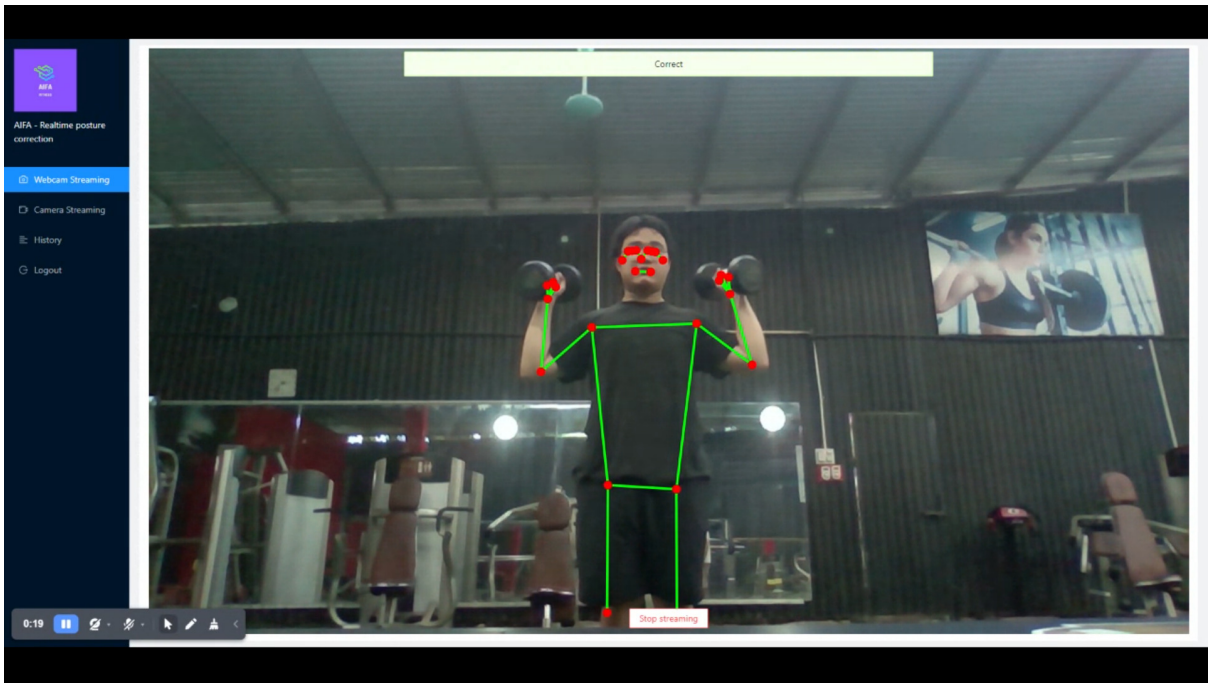


Figure 21: The interface during a session.

If the system detects wrong exercise forms during the session, it will be shown in the status box.

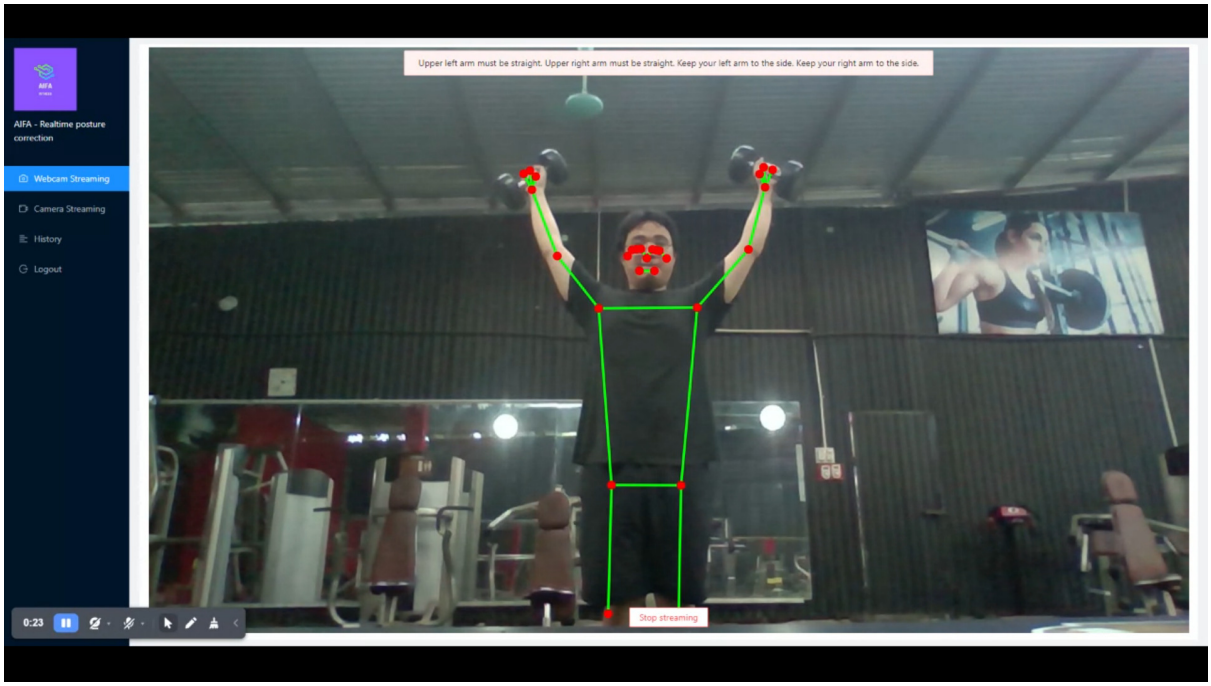


Figure 22: The interface when an error is detected.

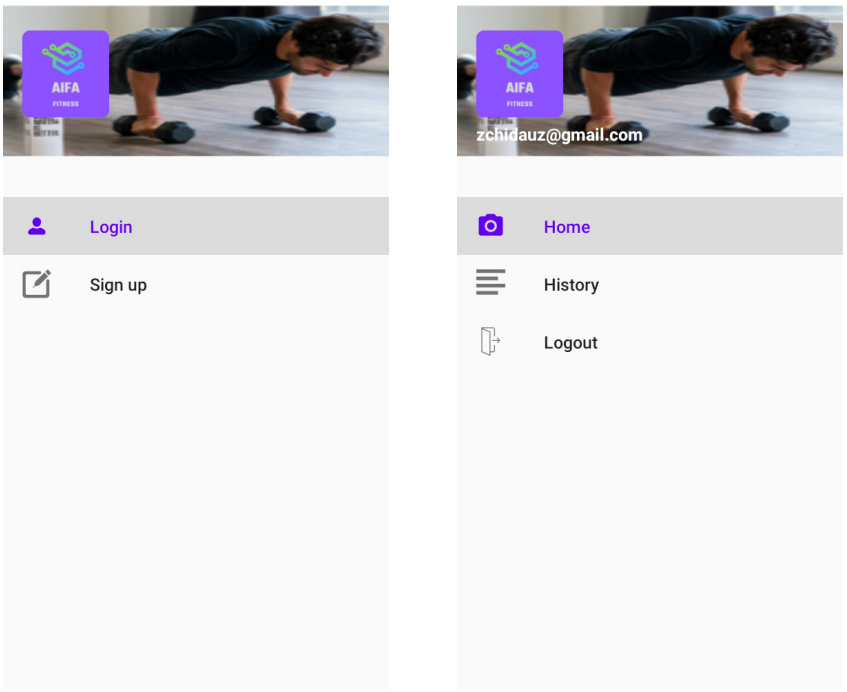
An example is shown in Figure 22. While performing the shoulder press, the user failed to keep the arms straight. The system detected the problem and gave instructions accordingly in the status box.

To end a session, the user needs to click the *Stop streaming* button which will return the user to the homepage. Now they can change the exercise to continue practicing.

After a workout, the user can exit by hitting the *Logout* button on the left panel.

B.2 Android

After installing the Android package, launching the application for the first time will move us to the login screen. Users can proceed to enter their registered email and password to login, or chose to register a new account through the *Sign up* option in the side menu as shown in Figure 23a.



(a) Options before logging in. (b) Options after logging in.

After logging in, the user will be directed to the main screen and can access the features of the application as shown Figure 23b. In the drop-down menu shown in Fig 24, users can choose from three exercises: shoulder press, deadlift, and hammer curl.

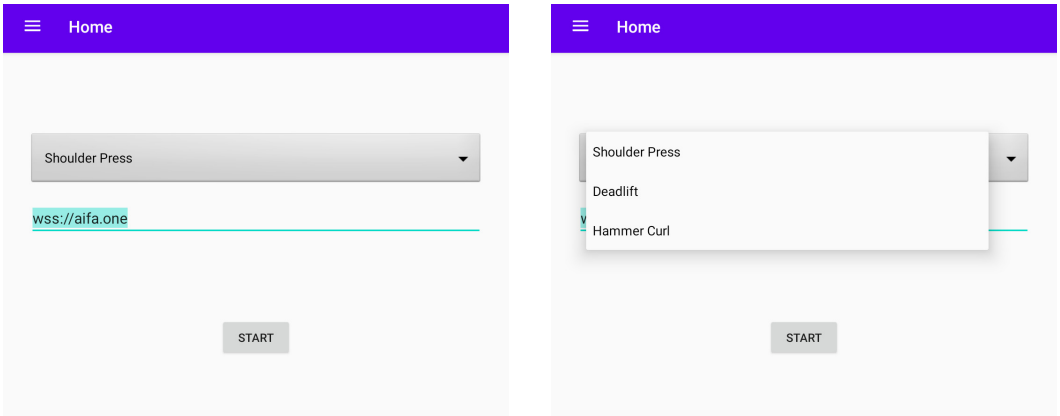
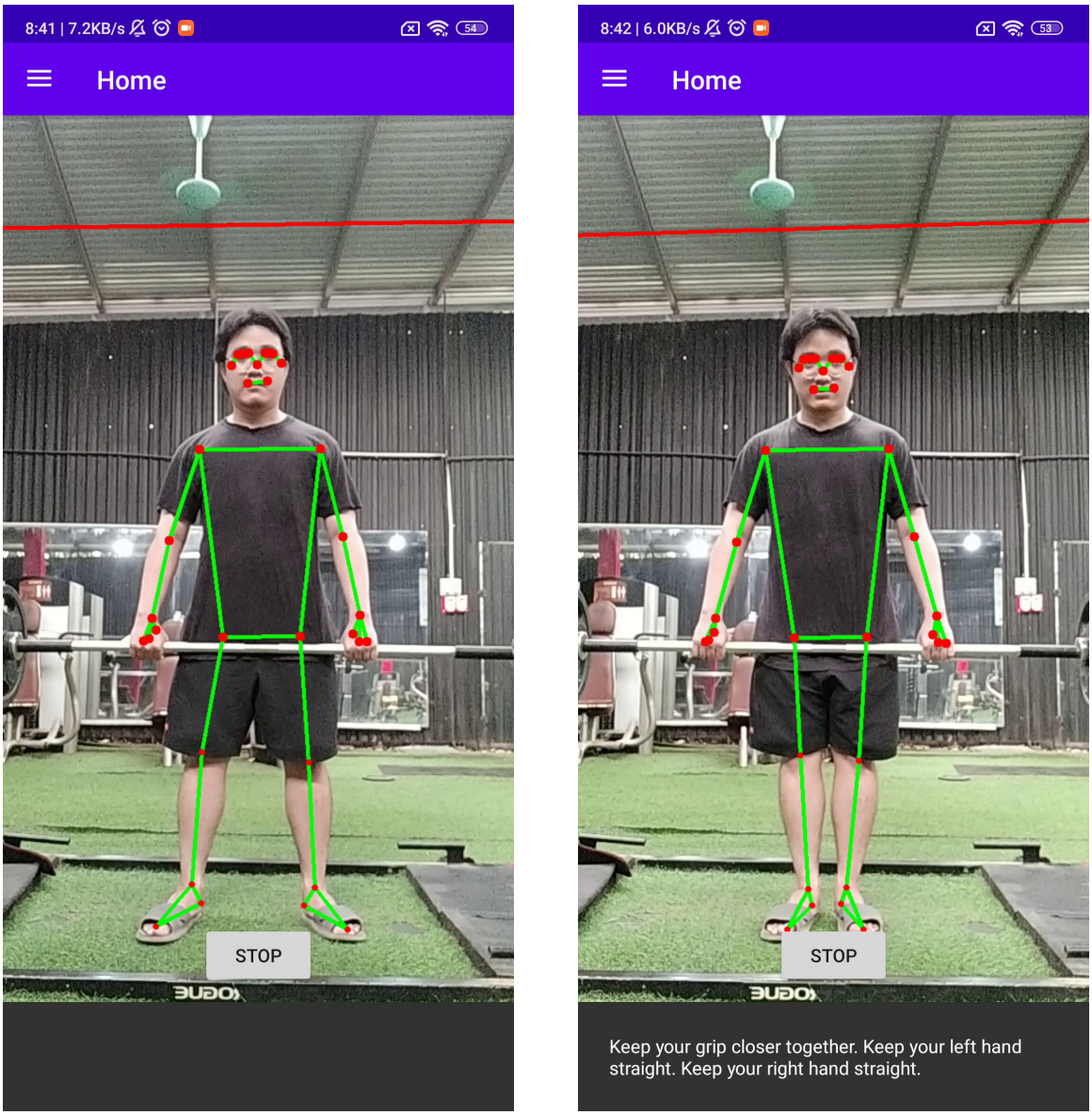


Figure 24: The drop-down menu where the user can choose an exercise to perform and get corrections.

With an exercise chosen, users can proceed to initialize a session by hitting the *Start* button. The application will use the front camera as the main input source.

An example of the GUI can be seen in Figure 25. During this specific shoulder press session, the user failed to keep their arms straight. The system detected the problem and gave the user instructions accordingly, as can be seen in 25b.



(a) GUI when no error is detected.

(b) GUI when an error is detected.

Figure 25: The GUI during a session.

To end a session, users need to click the *Stop* button, which will return them to the main screen. Now they can change the exercise to continue practicing.

If the user wish to log out, they need to hit the *Logout* button on the side panel, which will return them to the login screen.