CJC 2401 Mid-Term Progress Report

Hui Wang Chi 1155159410

Yuen Ho 1155176855

Table of Contents

1. Abstract

This midterm report outlines all the work we have completed by the end of October 2024. Since the submission of the project planning report, we have been working in parallel, with each of us focusing on our respective tasks to maximize efficiency. Gordon has primarily handled the technical aspects, including coding and benchmarking, while Leo has focused on tasks such as data preparation and writing non-technical sections of the report.

In addition to detailing the progress we have made during this midterm period, we have also included sections from the planning report, such as the introduction and related work, and expanded on them to bring this report closer to its final form. Furthermore, we briefly discuss our ongoing work and outline our upcoming plans.

2. Introduction

2.1 Project Goal and Report Theme

Our project goal is to develop a model that offers real-time posture correction for yoga and gym exercises. First, we have compiled our own dataset of posture images specifically tailored for this project. Second, we have examined and compared the performance of various existing human pose estimation models regarding latency and accuracy.

In this report, we review the methods and processes undertaken up to mid-term assessment, focusing on the two major goals outlined above. After mid-term, we will employ the most effective model for extracting body keypoints, then develop a model to classify the actual yoga pose type and determine whether it is correct.

2.2 Current Focus of Project

While our primary goal is to develop a model for correcting yoga and gym exercises, due to time constraints, we are currently concentrating on yoga postures in the first semester. Gym exercises are the focus of the upcoming semester.

Yoga places significant emphasis on proper posture. Some yogis may assume that yoga poses are less likely to cause injuries because its poses often appear static, and soothing compared to faster, more dynamic sports like football or basketball. However, research indicates that improper yoga postures can also lead to pain [1]. The risk of injury and discomfort from yoga is comparable to that of other exercises.

For individuals practicing yoga without the guidance of a coach, identifying and correcting improper postures can be difficult. To address this issue, we aim to develop a posture correction model to assist these solo trainers in facilitating safe and convenient training sessions.

2.3 Problem Statement

Currently, there is a shortage of mature posture correction applications in the market. MoveNet, PoseNet, OpenPose, and BlazePose are the four mainstream human pose estimation models. While some research papers compare these individual models, none provide a comprehensive analysis of all four simultaneously in terms of their suitability for our specific case. Hence, this report introduces two research papers that compare these models, and discusses the limitations identified in these studies. Besides, it presents our own benchmarks, and the corresponding approaches applied to these four models.

Additionally, there is no existing dataset that fully meets the requirements of our project. Key challenges include the absence of negative samples (incorrect postures) and the high variance in activity poses, leading to insufficient data for certain postures. In a later section of this report, we review the progress made in dataset preparation and outline the issues we encountered during the process.

3. Related Work

3.1 Existing Market Product

Regarding pose correction in yoga, it is essential to ascertain whether any existing market products offer similar functionalities to what we aim to achieve, so that we can draw insights from the approaches used by current developers.

Considering this, we have identified several leading market products that deliver yoga lessons to individual users, including Peloton [2], Yoga Studio App [3], Alo Moves [4], DoYouYoga [5], Gaia [6], and Glo [7]. However, none of these products provide the service or feature for assessing and correcting users' poses.

For example, Peloton and Alo Moves provide pre-recorded lessons to users. The video lessons feature one instructor introducing the poses and demonstrating the correct postures. Although they offer guidance to help users avoid potential injuries, it is difficult for users to assess their posture correctness during exercises without real-time adjustments.

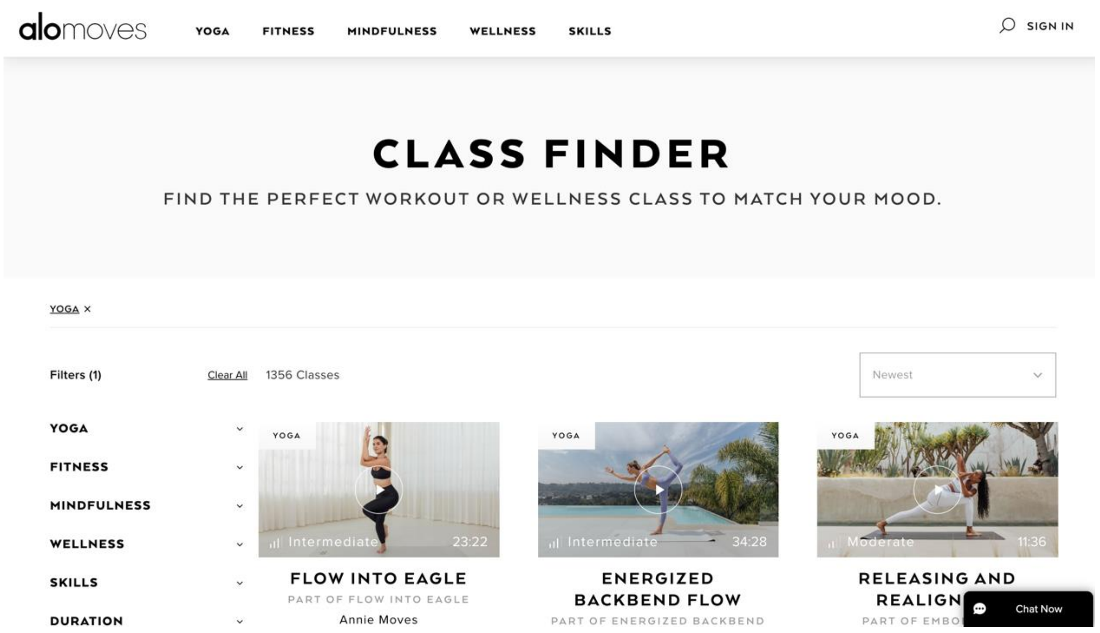
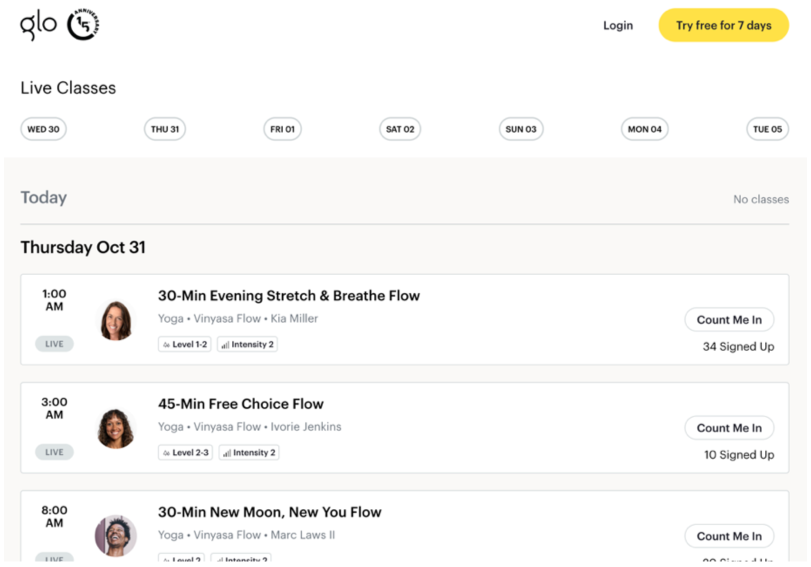


Figure 1: Interface of the Alo Moves website.

Some products offer live-streaming lessons instead of pre-recorded yoga lessons, such as Glo. These products and platforms provide real-time correction through coaching. However, this feature is not aligned with our project goal, because our focus is to give pose correction for the solo trainers who practice yoga without the guidance of a coach.



Live Classes

WED 30 | THU 31 | FRI 01 | SAT 02 | SUN 03 | MON 04 | TUE 05

Today — No classes

Thursday Oct 31

1:00 AM — LIVE
30-Min Evening Stretch & Breathe Flow
Yoga • Vinyasa Flow • Kia Miller
Level 1-2 | Intensity 2
Count Me In — 34 Signed Up

3:00 AM — LIVE
45-Min Free Choice Flow
Yoga • Vinyasa Flow • Ivorie Jenkins
Level 2-3 | Intensity 2
Count Me In — 10 Signed Up

8:00 AM
30-Min New Moon, New You Flow
Yoga • Vinyasa Flow • Marc Laws II
Count Me In

Figure 2: Live classes offered by Glo.

In conclusion, there are no existing products that align with the objectives of our product in the realm of yoga practices. Further research will be undertaken in the upcoming semester to explore any products, if they exist, that are focused on correcting gym postures.

3.2 Existing Comparison of Human Pose Estimation Models

3.2.1 Brief Introduction to Existing Papers

We have reviewed two existing research papers that focus on comparing mainstream human pose estimation models: "Comparative Analysis of OpenPose, PoseNet, and MoveNet Models for Pose Estimation in Mobile Devices" [8] and "Body Posture Detection and Comparison between OpenPose, MoveNet, and PoseNet" [9].

In the first paper, the study compares OpenPose, PoseNet, and MoveNet in terms of the time taken by each model to estimate 1,000 images and their respective accuracy. Images from the COCO and MPII datasets are used for the evaluation. The study concludes that while MoveNet is the fastest model, PoseNet achieves the highest average accuracy of 97.6%, while also being the second fastest. Although OpenPose ranks second in accuracy, its processing time is approximately 12 times slower than MoveNet.

In the second paper, the study also compares OpenPose, PoseNet, and MoveNet. Over 1,000 images from the COCO dataset were used to evaluate the accuracy and processing time for these three models per testing. The results are consistent with the previous paper: PoseNet achieves the highest average accuracy at 98.04%, MoveNet is the fastest model, and OpenPose remains the slowest, taking approximately 13.5 times longer than MoveNet.

3.2.2 Shortcomings of the Current Papers

Considering the shortcomings of these two papers, they share several limitations.

3.2.2.1 Omission of BlazePose in Benchmarking

First, both papers benchmark only OpenPose, MoveNet, and PoseNet, but omit BlazePose. Despite being published after 2022, and although BlazePose was released in 2020, it was not included in their comparisons. This is a significant omission, as BlazePose is also a well-known and advanced pose estimation model worth evaluating.

3.2.2.2 Insufficient Information for Judgment

Second, neither paper provides sufficient information for us to determine whether these pose estimation models can be effectively applied to our task. Since we aim to develop a real-time posture correction application, latency is a critical factor. Although the papers report processing times, they do not specify the hardware or environment used for benchmarking. This makes it difficult to assess whether the models are fast enough for our use case, as they might have been tested on highly advanced hardware setups that do not reflect typical deployment environments.

3.2.2.3 Potential Dataset Bias in Benchmarking Results

Third, the datasets used for benchmarking in these two papers may introduce bias into the results. Both papers use COCO or MPII datasets for benchmarking. While these are mainstream open-source datasets for human pose estimation tasks, it is difficult to rule out the possibility that the pre-trained weights of the models were fine-tuned using these datasets. Although we cannot find official documentation on the datasets used to train the pre-trained weights of the three models, it is widely inferred that PoseNet and MoveNet were trained with the COCO dataset, as both output 17 keypoints that match the labeled keypoints in COCO. Therefore, the results of these two papers may be biased, as they use these large public datasets for testing.

4. Benchmark on Existing Pose Detection Models

4.1 Briefing

As outlined in our project plan, we aim to use an existing pose detection model to process input images (video frames) and extract the body keypoints of the user as the first stage of our model. Among several studies on posture detection and comparison of pose estimation models [8, 9, 10, 11], we identified four mainstream, commonly used models: OpenPose, PoseNet, MoveNet, and BlazePose. Although there is existing research comparing some of these models [8, 9], it does not provide sufficient heuristics, as discussed in Section 3.2.2, to determine which model is best suited for our project. Therefore, we conducted a thorough and fair comparison of their performance in terms of latency, accuracy, and resource usage to select the most suitable model.

4.2 Detailed Approach

Our benchmarking process is straightforward. We are running the four different models on the same dataset and measuring their performance. Specifically, we evaluate:

- Latency: The time taken by the models to process the images.
- Accuracy: How well the models identify and infer keypoints.
- Memory Usage: A rough reference value of the total memory consumption after the inference process on the entire benchmark dataset.

For both latency and memory usage, we are running each model three times and taking the average to ensure more accurate results. The benchmarking is performed under the same conditions (e.g., platform, GPU, software version) to ensure fairness.

4.2.1 Dataset

We used a public dataset (hereinafter referred to as the base dataset) from Kaggle [12] that originally consisted of 1,171 images across six different yoga postures: Down Dog, Goddess, Plank, Side Plank, Tree, and Warrior. After revising the dataset, we removed duplicate images (based on SHA-256 hash values, using the `hashlib` library in Python) and excluded irrelevant images, such as cartoon-style illustrations, to ensure the dataset focuses on real-life scenarios. The resultant dataset for this benchmark now contains 926 images.

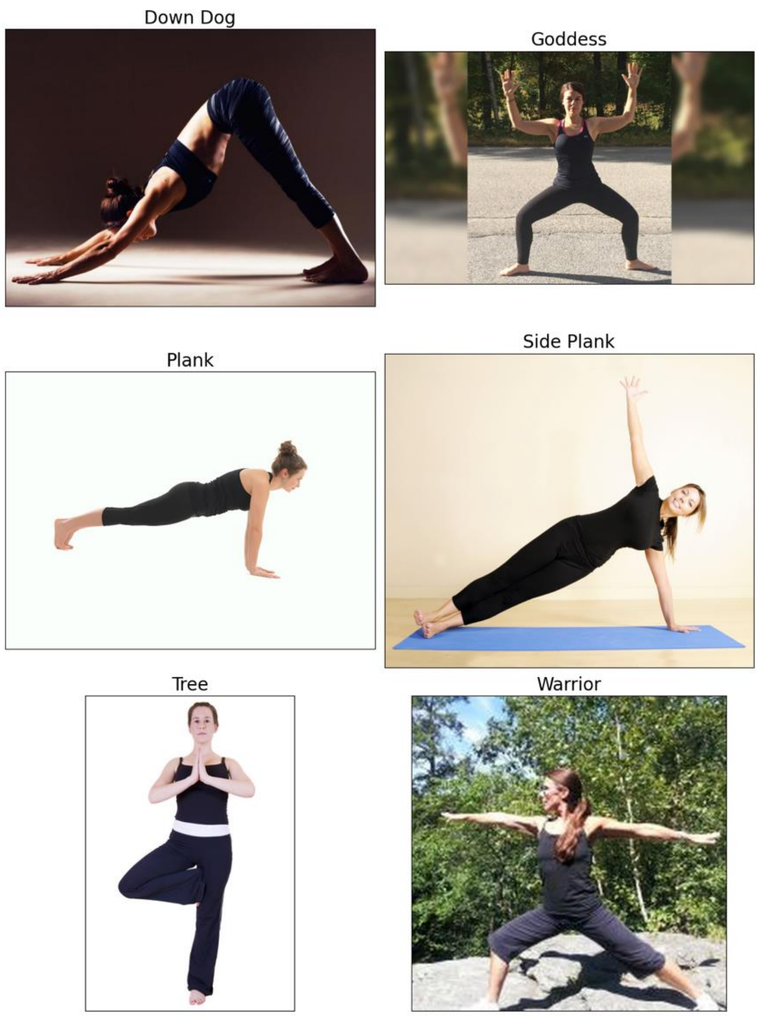


Figure 3: A sample from the dataset, showing one image for each of the six postures.

4.2.2 Evaluation of Model Latency

Latency is measured per image by dividing the total time taken by the number of input images. We only measured the time required for model inference, excluding other processes such as initialization. All four models processed the images one by one (not in batches) to ensure identical conditions for a fair comparison.

4.2.3 Evaluation of Accuracy

Since the dataset consists of raw images of yoga poses without pre-labeled keypoints, the most intuitive way to evaluate the models' accuracy is through human evaluation. We randomly sampled images of four different poses from the dataset and visually inspected the keypoints generated by the four models to assess their accuracy.

The number of keypoints output by different models may vary. Therefore, we evaluated the models using the widely adopted set of 17 COCO keypoints [13]. COCO, which stands for "Common Objects in Context," is a large-scale dataset designed for various computer vision tasks, including keypoint detection [14]. The official COCO keypoints represent the following body parts:

Nose	Left Eye	Right Eye	Left Ear	Right Ear	Left Shoulder
Right Shoulder	Left Elbow	Right Elbow	Left Wrist	Right Wrist	Left Hip
Right Hip	Left Knee	Right Knee	Left Ankle	Right Ankle	

For our accuracy evaluations, we focused exclusively on these 17 keypoints, as they are shared among the models we are comparing, ensuring a consistent benchmark.

When visualizing the keypoints on an image, we connect specific keypoint pairs to form a skeleton, which helps facilitate the assessment of keypoint accuracy. The connection rules remain consistent across all models in our evaluation and are based on references from the official TensorFlow site [15].

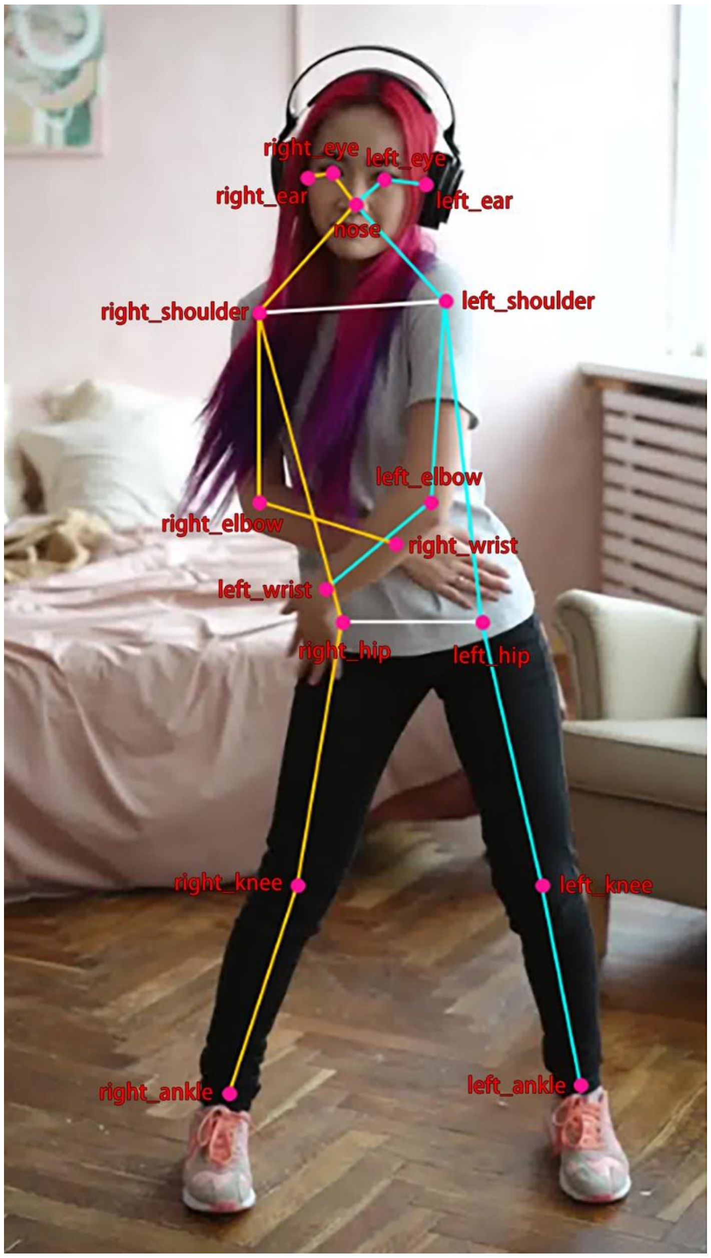


Figure 4: A sample image showing the 17 COCO keypoints and the paired connections used in this benchmark.

4.2.4 Evaluation of Memory Usage

Different models may be deployed using different frameworks. For instance, PoseNet uses TensorFlow LiteRT, while BlazeNet relies on the MediaPipe module. This variation complicates the definition and measurement of memory usage across models. To simplify and maintain consistency in our benchmarks, we measured peak memory usage immediately after processing the entire dataset with each model, using the code `resource.getrusage(resource.RUSAGE_SELF).ru_maxrss`, as referenced from StackOverflow [16]. This approach measures the usage of unified memory, where the memory is shared between the CPU and GPU [17]. It's important to note that this provides only a rough estimate, as memory usage can vary significantly based on factors such as changes in data processing.

4.3 Code Implementation

For the benchmarking code, we primarily referenced the sample codes provided for each model. It's important to note that the raw output formats of the four candidate models differ significantly, as shown below:

	OpenPose	PoseNet	MoveNet	BlazePose
Output Format	N/A	A heatmap along with offset vectors [18]	Relative coordinates of the input image padded into a square	Relative coordinates directly from the input image

As a result, we put effort into standardizing these outputs into a unified format: an array of shape "number of keypoints" × "keypoint dimensions".

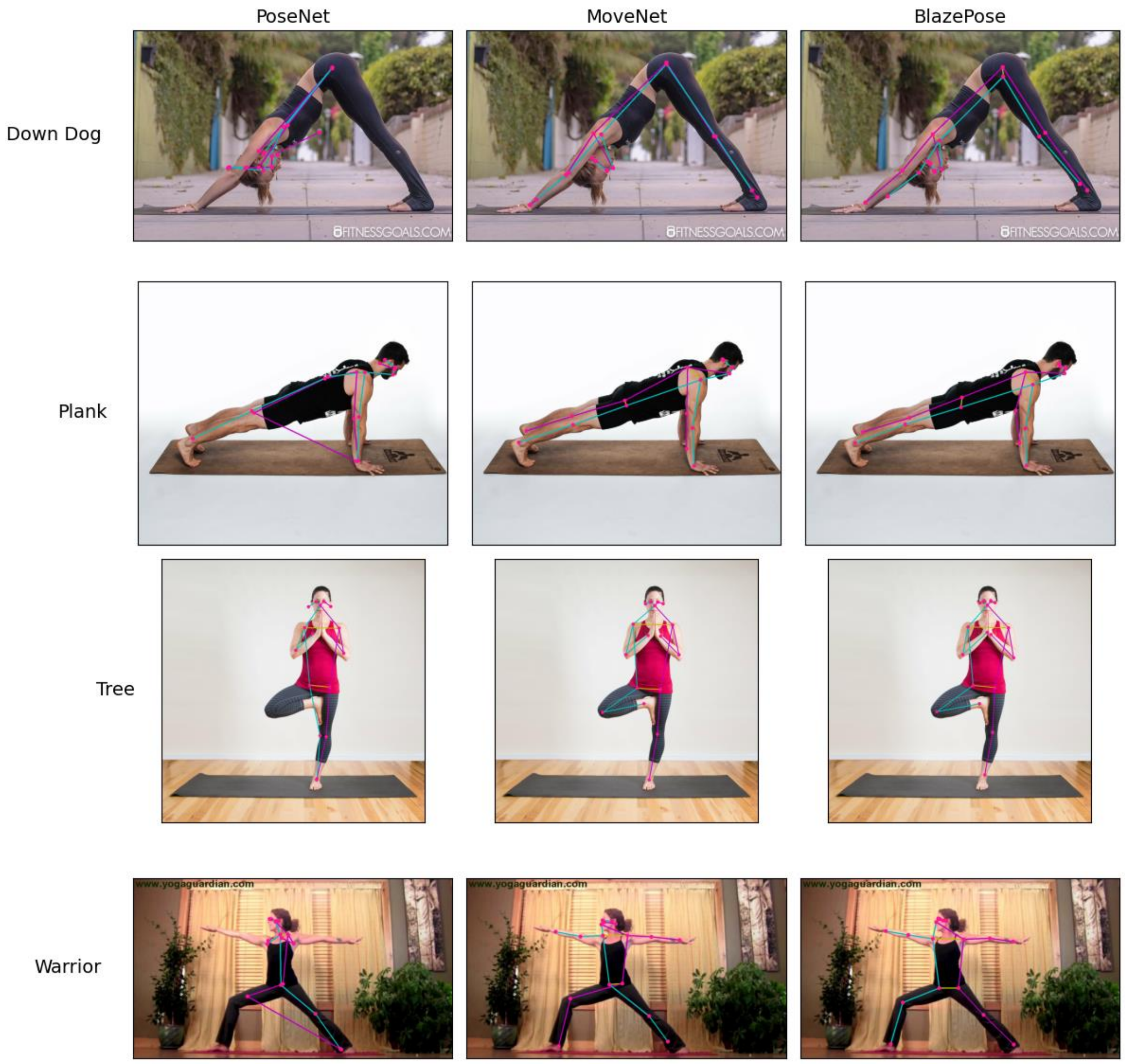
Additionally, we implemented a function called `save_image_with_prediction()` to plot the predicted keypoints and draw the skeleton for visualization, as shown in Figure 4. Although the documentation also provides a plotting function, we chose not to use it because its implementation, which relies on the Matplotlib backend, distorts the original image by adding borders or reducing resolution. Instead, we implemented our solution using OpenCV as the backend, which avoids these issues and preserves the image quality even after plotting the skeleton.

4.4 Benchmark Results

4.4.1 Result Table

	OpenPose	PoseNet	MoveNet	BlazePose
Year of First Release	2017	2017	2021	2020
Year of Latest Version	2020	2021	2023	2023
Number of Keypoints	135	17	17	33
Keypoint Coordinate Dimensions	x, y	x, y	x, y	x, y, z
Total Inference Time	N/A	33.8224 s	31.6186 s	24.5882 s
FPS (Frames per Second)	N/A	27.3783	29.2866	37.6603
Approximate Memory Usage (Unified Memory)	N/A	4.36 GiB	5.01 GiB	4.59 GiB

4.4.2 Result Samples



4.5 Model Results Analysis

4.5.1 OpenPose

OpenPose [19], developed by CMU, is claimed to be the first real-time, multi-person system capable of detecting human body, hand, facial, and foot keypoints (a total of 135 keypoints) in single images. It was first released in 2017, with the latest update in 2020.

In this benchmark, OpenPose did not run successfully. The issue was caused by the failure to download the pre-trained weights from the server using the setup command provided in the documentation. This may be due to the model being out of maintenance. Additionally, as a model last updated in 2020, OpenPose is not as up-to-date as the other three benchmarked models and lacks official support for Python versions newer than 3.7. Moreover, OpenPose is written in C++, rather than commonly used Python deep learning libraries like PyTorch or TensorFlow, making it more difficult to inspect or modify its structure. It also cannot be easily installed via pip, further complicating the setup.

Regarding the issue with downloading the weights, we found a related discussion on the OpenPose GitHub repository [20]. However, no update had been made at the time of our benchmark. As a workaround, we referred to two existing research studies [8, 9] (introduced in Section 3.2.1) that compared OpenPose with PoseNet and MoveNet. Both studies concluded that OpenPose offers no advantage in terms of accuracy or speed compared to the other two. Based on these findings, we reasonably concluded that OpenPose is not suitable for the purposes of our project.

4.5.2 PoseNet

PoseNet is a human pose estimation model developed by Google, first released in 2017, capable of detecting 17 COCO keypoints.

During the planning phase of our project, we were already aware that PoseNet is no longer actively maintained, as Google has developed a successor called MoveNet. According to Google's official statements [21], MoveNet outperforms PoseNet in both latency and accuracy across various datasets. Despite this, we still decided to simply benchmark PoseNet to gain hands-on experience with the environment setup and assess the actual performance difference between these two generations of pose estimation models from Google.

For this benchmark, PoseNet was run in CPU mode. We had already confirmed that our TensorFlow environment had GPU support enabled by using the official check command, as shown below:

```
>>> print("Num GPUs Available: ", len(tf.config.list_physical_devices('GPU')))
 Num GPUs Available:  1
```

Additionally, we successfully deployed other models in the same environment using the GPU. Therefore, the likely cause of PoseNet's failure to run on the GPU could be unsupported GPU operations in the model. This issue is further explained by the fact that TensorFlow LiteRT, which PoseNet's main version is based on, supports only a limited set of machine learning operations [22].

Although there may be a workaround to deploy PoseNet on the GPU, we decided not to invest too much time in pursuing this, as PoseNet is deprecated. Our primary objective was simply to familiarize ourselves with the environment setup. Thus, we proceeded with CPU mode, but the time performance was no longer suitable for direct comparison.

In terms of benchmark results, PoseNet showed significant accuracy degradation when the person's pose was not facing the camera, such as in the downward dog and plank poses. Despite running on the CPU, PoseNet achieved around 27.4 fps (926 images / 33.8 seconds), which is acceptable. The memory consumption after the inference process was 4.36 GiB.

4.5.3 MoveNet

As mentioned earlier, MoveNet is Google's successor to PoseNet, first released in 2021 [15]. It detects the same 17 COCO keypoints as PoseNet but remains one of Google's most advanced human pose estimation models, known for its ultra-fast performance and high accuracy.

Google offers several versions of MoveNet, varying in precision and model size to suit different platforms and user needs. For this benchmark, we selected the default version, MoveNet Lightning (FP32), which balances speed and accuracy.

MoveNet performed well in skeleton estimation across all random samples. The skeleton was accurately drawn regardless of posture, camera angle, or the direction the user was facing. In terms of latency, MoveNet achieved 29.3 fps (926 images / 31.6 seconds) with GPU utilization, and the total memory consumption after the inference process was 5 GiB.

4.5.4 BlazePose

BlazePose, also known as MediaPipe BlazePose, is another human pose estimation model developed by Google, first released in 2020. BlazePose is part of the MediaPipe Solutions framework and is used for pose landmark detection [23]. Unlike other models, BlazePose can detect not only the 17 COCO keypoints but also an additional 16, for a total of 33 keypoints, making it another state-of-the-art pose estimation model from Google.

BlazePose also estimates the z-coordinate (relative depth) of each keypoint, in addition to the 2D coordinates (x, y), allowing for more accurate joint angle estimation regardless of camera angle. However, for this benchmark, we only considered the x and y coordinates when comparing accuracy.

In the MediaPipe interface, model complexity is a customizable parameter that allows developers to choose between better accuracy or lower latency. For this benchmark, we selected the default setting, `model_complexity=1`.

BlazePose performed well in skeleton estimation, comparable to MoveNet. There were no significant issues, and the skeleton was consistently well-fitted to the user's body, regardless of posture. In terms of latency, BlazePose achieved 37.6 fps (926 images / 24.6 seconds) with GPU utilization, and the memory consumption after the inference process was 4.6 GiB.

4.6 The Chosen Model

4.6.1 Quick Summary

Based on the individual result analysis in Section 4.5, we concluded that BlazePose is the most suitable model among the four for our project.

4.6.2 Detailed Explanation

In terms of accuracy, as demonstrated by the sample estimations in Section 4.4, both MoveNet and BlazePose perform almost identically. Both models significantly outperform PoseNet, regardless of whether the background is simple or complex. Even for a human observer, it is difficult to easily judge which model, MoveNet or BlazePose, provides better estimations. However, upon closer inspection, BlazePose produces a more reasonable representation of the hip area in the Warrior pose output sample. So, we conclude that BlazePose performs the best in this regard.

When it comes to latency and memory usage, BlazePose offers lower latency and consumes less memory. We can confidently say that BlazePose outperforms MoveNet in terms of computational efficiency and resource usage, at least when both models are used with their default configurations.

In summary, BlazePose stands out as the better option overall, with advantages in accuracy, latency, and memory usage. Additionally, BlazePose provides more keypoints and an extra dimension (relative depth) compared to MoveNet. It is also an up-to-date model that continues to receive maintenance and updates. For these reasons, BlazePose is the model we have chosen for the next stages of our project.

5. Dataset preparation

5.1 Observation

As mentioned in section 4.2.1, we utilized a dataset sourced from Kaggle as our base dataset, which was used in the benchmarking process. Before October, out of 6 poses datasets within the base dataset, we selected Down Dog, Plank, Tree, and Warrior as our basic cases. These 4 poses had the highest number of images available for pose correction, making them the basic cases of our project. However, upon observing the data, we identified issues within the current dataset including:

1. Duplicate Images
2. Irrelevant Images
3. Misclassified Images
4. Potentially Suboptimal Image Selections for the Dataset

5.1.1 Duplicate Images

Although the base dataset has a diverse variety of images regarding each pose, we observed that some images are duplicated within the dataset. This leads to redundancy, consuming extra computational resources and storage. Moreover, it leads to a biased analysis of our model toward new data in future development.

5.1.2 Irrelevant Images

Our project aims to secure pose correction for adolescents and adults in real-time. Considering this, some unrelated images were observed in the dataset. These images include cartoon-style illustrations, images of children doing specific yoga poses, and incomplete poses. This issue might lead to reduced performance of the model if it learns from these irrelevant patterns and unrelated images.

5.1.3 Misclassified Images

First, we identified that the base dataset is not accurately categorized. For example, the Warrior dataset included 4 distinct poses instead of 1 specific pose, which are Warrior I, II, III and Peaceful Warrior.

Secondly, we observed data interchange between datasets, such as images of side planks present in the plank dataset, and vice versa.

5.1.4 Potentially Suboptimal Image Selections for the Dataset

First, upon observing Section 5.1.3, we concluded that some of our initially chosen poses might not be optimal choices for future development, considering the possible reduction in the number of images of these datasets after correct dataset categorization.

Second, a comparison of the benchmark images reveals that it is difficult to distinguish some poses' correctness, as the model might not be able to detect minor deviations through skeleton structures and joint angles. For example, yoga tutorials introduce common mistakes in tree poses, such as trainers not keeping their weight evenly distributed in their supporting foot and failing to engage their core muscles to hold the torso upright [24]. These are subtle pose inaccuracies that can be difficult to recognize.

5.2 Dataset Optimization

5.2.1 Enhancement of Dataset Size

To train the model with a larger image dataset, we explored four additional datasets from Kaggle [25, 26, 27, 28]. After comparing the dataset sizes, we selected Down Dog, Plank, Side Plank, and Warrior II as the poses for our latest dataset (whereas the original dataset included Down Dog, Plank, Tree, and Warrior).

This decision is based on the abundance of images available across the Kaggle datasets, and this solves the issue raised in Section 5.1.4. Below is a graph showing the number of images for each pose after combining the five datasets into one single dataset:

Datasets	Number of Images in Original Dataset	Number of Images after combining 5 Datasets
Down Dog	253	745
Plank	153	328

Side Plank	106	168
Warrior II	249	351

5.2.2 Enhancement of Dataset Correctness

5.2.2.1 Duplicate Removal

Similar to Section 4.2.1, we removed duplicate images using Python scripts by comparing the SHA256 values of the images in the dataset.

Additionally, we implemented the Duplicate File Finder by Nektony [29] to ensure all the duplicates are removed, which guarantees the reliability and completeness of our dataset to avoid suboptimal model performance. These approaches solve the issue mentioned in Section 5.1.1.

5.2.2.2 Dataset Categorization

Problems introduced in Sections 5.1.2 and 5.1.3 are solved upon dataset categorization. However, unlike the processes in other data preparation stages, we categorize the pose data by distinguishing the correctness manually. Therefore, further confirmation is required to ensure the accuracy of the dataset.

6. Work in Progress

6.1 Briefing

Once the benchmarking was completed and we selected BlazePose as the model for the first stage of our project (as mentioned in Section 4.6), we moved on to designing the model for the second stage.

The purpose of the second-stage model is to take the output from BlazePose as input—specifically a (33, 4) array containing the (x, y, z, confidence) values of 33 keypoints—and perform a classification task to determine the posture type and its correctness. In the following subsection, we will provide a more detailed overview of the second-stage model specifications.

6.2 Model Input Structure

As mentioned in Section 6.1, the model's input is a (33, 4) array for each image. In our current approach, the coordinate information of these 33 keypoints is normalized before being passed to the second-stage model. The x and y coordinates are normalized per image, with values ranging from [0,1].

However, we are still experimenting with whether to pass the entire BlazePose output to the second-stage model, as some keypoints, such as those indicating fine details like eye or mouth positions, may not be relevant for tasks related to yoga postures.

6.3 Model Output Structure

To support both posture classification and correctness evaluation, the output is designed as an array of shape (# of posture classes, 2). For each row, the first entry represents the score for a specific posture type, and the second entry represents the correctness. To estimate the posture type, we apply a *softmax* function and select the *argmax* from the first column. For correctness, we apply a sigmoid function to the second element of the corresponding row, where a value > 0.5 indicates a correct posture, and ≤ 0.5 indicates an incorrect one.

One advantage of this structure is its flexibility. If we want to classify incorrect postures into more specific categories, we can expand the output array's row dimension to support finer granularity.

6.4 Model Architecture

Given the nature of the input data, we believe that a Multi-Layer Perceptron (MLP) is sufficient for our task. Since the 33 keypoint coordinates can be treated as extracted features from the raw image, we do not need more complex architectures like CNNs (for local connectivity) or RNNs (for long-term dependencies).

6.5 Dataset

For our project, we plan to construct a custom Yoga dataset. It will include four yoga postures: Downward Dog, Plank, Side Plank, and Warrior II, each with correct and incorrect samples.

After integrating the related data into our dataset, data variety was a problem that was left to be tackled. Most of the data sourced from Kaggle consisted of correct/positive samples of the poses. However, negative samples of the poses are important for the dataset to train the model comprehensively and evaluate the correctness of users' poses in later stages.

Given the situation, we decided to collect images by extracting frames from specific videos. We searched YouTube for common mistakes or incorrect postures corresponding to each pose. Initially, we attempted to extract the relevant frames using the frame extraction feature in VLC media player but encountered technical issues and failed to convert videos to frames.

As a result, we are now using OpenCV to convert the YouTube videos into frames. If successful, this approach will enable us to expand the dataset with more negative samples, and we can apply a similar strategy to acquire additional positive samples in subsequent stages.

Due to time and resource constraints, we do not anticipate building a large-scale dataset. Thus, to further enhance the dataset size and variety, we will employ data augmentation techniques and random sample weighting to improve training performance with the limited data available.

6.6 Current Status

As mentioned in Section 1, dataset preparation is happening in parallel with model design and coding. At the time of designing the second-stage model, Leo is still working on dataset preparation, with only two postures available so far, including both positive and negative samples.

In the current two-posture classification setup, using a custom MLP model, we have achieved around 80% accuracy on the test set (unseen data). A more in-depth training experiment will be conducted once the full dataset is ready.

7. Upcoming Tasks

In the remaining time before the final report submission, we will focus on refining the second-stage model to achieve the best possible accuracy. For the Term 1 FYP presentation, we aim to present a model capable of classifying four different yoga postures and determining whether the pose is correct, given an image input. Support for real-time video frame input is planned for development in the coming semester.

8. References

[1] M. Campo, M. P. Shiyko, M. B. Kean, L. Roberts, and E. Pappas, "Musculoskeletal pain associated with recreational yoga participation: A prospective cohort study with 1-year follow-up," J. Bodywork Movement Ther., vol. 22, no. 2, pp. 418–423, Apr. 2018. [Online]. Available: https://doi.org/10.1016/j.jbmt.2017.05.022

[2] Peloton, "Guided yoga classes online | Peloton," [Online]. Available: https://www.onepeloton.com/classes/yoga

[3] Yoga Studio App, "#1 Yoga Studio App: Mind & Body," [Online]. Available: https://yogastudioapp.com/

[4] Alo Moves, "Alo Moves | Your At-Home Studio," [Online]. Available: https://www.alomoves.com/

[5] DOYOU, "DOYOU | Online Yoga, Fitness, and You," [Online]. Available: https://www.doyou.com/

[6] Gaia, "Streaming Online Yoga Videos | Gaia," [Online]. Available: https://www.gaia.com/yoga

[7] Glo, "Glo | Online yoga, meditation, & Pilates app for all levels," [Online]. Available: https://www.glo.com/

[8] B. Jo and S. Kim, "Comparative analysis of OpenPose, PoseNet, and MoveNet models for pose estimation in mobile devices," Traitement du Signal, vol. 39, no. 1, pp. 119, 2022. [Online]. Available: https://www.researchgate.net/publication/359476644

[9] P. Kaushik, B. P. Lohani, A. Thakur, A. Gupta, A. K. Khan, and A. Kumar, "Body posture detection and comparison between OpenPose, MoveNet and PoseNet," in 2023 6th International Conference on Contemporary Computing and Informatics (IC3I), vol. 6, pp. 234-238, IEEE, 2023. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/10397937/

[10] N. Kumar Reddy Boyalla, “Real-time Exercise Posture Correction Using Human Pose Detection Technique,” soar.suny.edu, Dec. 2021, Accessed: Sep. 23, 2023. [Online]. Available: https://soar.suny.edu/handle/20.500.12648/8622

[11] Y. Kwon and D. Kim, "Real-Time workout posture correction using OpenCV and MediaPipe," 한국정보기술학회논문지, vol. 20, no. 1, pp. 199-208, 2022. [Online]. Available: https://ki-it.com/xml/32020/32020.pdf

[12] L. Rajak, "Yoga Pose Dataset," Kaggle, 2023. [Online]. Available: https://www.kaggle.com/datasets/lakshmanarajak/yoga-dataset

[13] TensorFlow Team, "@tensorflow-models/pose-detection," npm, Version 0.0.3, [Online]. Available: https://www.npmjs.com/package/@tensorflow-models/pose-detection/v/0.0.3

[14] T. Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, "Microsoft COCO: Common Objects in Context," European Conference on Computer Vision (ECCV), Zurich, Switzerland, Sep. 2014. [Online]. Available: https://cocodataset.org/#home

[15] TensorFlow Hub, "MoveNet: Ultra fast and accurate pose detection model," TensorFlow, 2023. [Online]. Available: https://www.tensorflow.org/hub/tutorials/movenet

[16] Basj, "Total memory used by Python process?", Stack Overflow, Feb. 7, 2014. [Online]. Available: https://stackoverflow.com/a/7669482/15416614

[17] "Choosing a Resource Storage Mode for Apple GPUs," Apple Developer Documentation. [Online]. Available: https://developer.apple.com/documentation/metal/resource_fundamentals/choosing_a_resource_storage_mode_for_apple_gpus

[18] D. Oved, "Real-time human pose estimation in the browser with TensorFlow.js," The TensorFlow Blog, May 7, 2018. [Online]. Available: https://blog.tensorflow.org/2018/05/real-time-human-pose-estimation-in.html

[19] CMU Perceptual Computing Lab, "OpenPose: Real-time multi-person keypoint detection library for body, face, hands, and foot estimation," GitHub repository, 2024. [Online]. Available: https://github.com/CMU-Perceptual-Computing-Lab/openpose

[20] melodicoder, "Installing models through getBaseModels.bat (Windows 11, OpenPose Release v1.7.0)," GitHub, Sep. 30, 2024. [Online]. Available: https://github.com/CMU-Perceptual-Computing-Lab/openpose/issues/2316

[21] TensorFlow, "姿态预测," TensorFlow Lite 示例, Jan. 11, 2024. [Online]. Available: https://www.tensorflow.org/lite/examples/pose_estimation/overview?hl=zh-cn

[22] Google AI Edge, "GPU delegates for LiteRT," Google AI for Developers, Aug. 30, 2024. [Online]. Available: https://ai.google.dev/edge/litert/performance/gpu

[23] Google AI, "Pose Landmarker," Google MediaPipe Solutions, [Online]. Available: https://google.dev/edge/mediapipe/solutions/vision/pose_landmarker

[24] E. Millard, “A Step-By-Step Guide On How to Do Tree Pose (or One of Its Many Variations),” onepeloton, Jun. 28, 2024. [Online]. Available: https://www.onepeloton.com/blog/tree-pose/

[25] M. Tyagi, "Yoga Posture Dataset," Kaggle, 2024. [Online]. Available: https://www.kaggle.com/datasets/tr1gg3rtrash/yoga-posture-dataset

[26] U. Chowdhury, "Yoga Pose Classification," Kaggle, 2024. [Online]. Available: https://www.kaggle.com/datasets/ujjwalchowdhury/yoga-pose-classification

[27] S. Saxena, "Yoga Pose Image Classification Dataset," Kaggle, 2024. [Online]. Available: https://www.kaggle.com/datasets/shrutisaxena/yoga-pose-image-classification-dataset

[28] S. Khaorapapong and B. Latif, "Yoga Posture Dataset," Kaggle, 2024. [Online]. Available: https://www.kaggle.com/datasets/suradechk/yoga-posture-cleaned

[29] Nektony, "Duplicate File Finder for Mac - Free Download," [Online]. Available: https://nektony.com/duplicate-finder-free