Perfecting Poses: A Machine Learning-Powered Yoga Assistant for Real-Time Alignment and Progress Tracking

STADS code: BIBAPRO1PE

Emil Braagaard Tjørnelund - emtj@itu.dk

Laurits Brok Petersen - labp@itu.dk

Supervisor: Fabricio Batista Narcizo - fabn@itu.dk

May 14, 2024

Abstract

This thesis explores the development of a web application that enhances at-home yoga practice by providing real-time feedback on pose alignment using machine learning. As yoga gains popularity globally, many practitioners utilize digital streaming for guidance, yet lack real-time, personalized feedback on their poses, increasing the risk of injury and reducing practice efficacy. Our project addresses this issue by implementing pose landmark detection and classification technologies to guide users toward correct alignment and provide accuracy scores for each pose. We evaluate the effectiveness of our application against traditional instructional videos through a comparative user study. The findings suggest that integrating technology into yoga practice could substantially improve pose accuracy and safety, providing valuable insights for enhancing home-based yoga routines.

Contents

Chapter 1

Introduction

Yoga, originating in ancient India thousands of years ago [4], is a practice that harmonizes the body, mind, and spirit. It includes physical postures, breath control, meditation, and ethical disciplines. The practice is renowned for offering a comprehensive wellness experience, attracting individuals globally for various reasons[1]. Yoga has seen a significant rise in popularity, with the number of practitioners in the US increasing from 20.4 million in 2012 to 36.7 million in 2016—a 79% increase in just four years[2].

A key component of practicing yoga involves executing a variety of poses that can be challenging for beginners. Many yoga poses require both strength and flexibility for correct performance. Despite the difficulty, proper alignment during these poses is important. Alignment is a crucial aspect of yoga, and David Keil, a renowned yoga instructor and anatomy expert, has significantly advanced our understanding of it. With a background in Kinesiology [20] and as a Neuromuscular Therapist [8], he possesses a deep knowledge of the musculoskeletal system. While various yoga styles approach alignment differently, it remains essential for safety, preventing overexertion, and reducing the risk of injuries. Proper alignment enhances the yoga experience by improving balance, deepening stretches, and engaging muscles more efficiently [18][3].

Although many yoga classes offer professional guidance, 85% of people practicing yoga do so at home because it is more convenient, cost-effective, and private for them. Additionally, 79% of home practitioners use streaming services to educate themselves on yoga routines[4]. However, practicing yoga this way does not provide feedback on pose correctness, leading to potential misalignment.

This research project addresses the pressing need for improved alignment in yoga practice by developing a web application that can provide real-time feedback during yoga sessions. This application will integrate machine learning techniques to perform pose landmark detection on the subject, guiding yoga enthusiasts toward correct alignment and posture. Our research will explore various machine learning models for pose landmark detection to identify the one best suited for yoga practice. To further enhance the usability of our application, we will also implement pose classification, enabling the application to identify which yoga pose the subject is attempting to perform. Finally, we will test how our application performs compared to instructional videos and evaluate if pose detection and classification can be a helpful tool for practicing yoga pose alignment.

[1] 50 Blissful Yoga Statistics for 2024, J.R. Yoginidra 2024

[2] 2016 Yoga in America Study Conducted by Yoga Journal and Yoga Alliance Reveals Growth and Benefits of the Practice, Yoga Alliance 2016

[3] Yoga Alignment: Does It Really Matter?, Yoga Alliance 2023

[4] The Global Yoga Survey 2021, DoYou 2021

1.1 Objectives

These are the objectives of the project:

Objective 1: Determine the most accurate pose detection model for landmark detection in Yoga poses.

Objective 2: Train a pose classification model capable of accurately predicting four selected yoga poses across various environments.

Objective 3: Develop a web application integrating pose detection and classification to assist yoga practitioners in achieving better pose alignment and providing an accuracy score for each pose.

Objective 4: Conduct a user split test consisting of two groups: Group A utilizing instructional videos and Group B utilizing our developed web application to assess which method produces the most precise pose alignment.

1.2 Research Question

The research question that this thesis aims to answer is: *"Can pose detection and classification be utilized effectively to provide learning-based feedback to improve the accuracy of yoga practitioners' poses compared to traditional video-based instructions?"*

1.3 Hypothesis

This bachelor project hypothesizes that integrating real-time pose detection and classification for guided alignment in yoga practice will result in more accurate poses compared to instructional yoga videos, enhancing the accuracy of yoga practice.

1.4 Outline

The report's structure is as follows: Chapter 2 introduces work related to this paper. Chapter 3 will establish a ground truth for correctly performed yoga poses and provide background on some deep learning concepts used for pose detection and classification. Chapter 4 will compare different pose detection models and evaluate which is most accurate for yoga poses. Chapter 5 will explain how our chosen model, BlazePose, functions under the hood when predicting poses. Chapter 6 documents the training and testing of an accurate yoga pose classification model. Chapter 7 showcases the features built into our application and explains how it works. Chapter 8 documents the test phase of the program and evaluates the data collection to see how our application compares to traditional home yoga practice using instructional videos. Chapter 9 suggests future improvements for our application based on the input given in previous chapters. Lastly, Chapter 10 will present the conclusion of this thesis, where we answer our thesis hypothesis.

Chapter 2

Related Work

Pose estimation is a field of its own in machine learning. More and more companies and organizations are investing time and money in advancing the field. Human Pose Estimation involves detecting and tracking specific body parts, mostly body joints, called pose landmarks. These key points can then be connected by lines, creating a skeleton representation of the human body [13][23].

After the COVID-19 pandemic shocked the world in late 2019, more and more fitness coaches transitioned their practice to an online environment [16]. This rise in the popularity of the online fitness industry has resulted in many projects aiming to evolve the technologies involved in these practices. Several projects have aimed to introduce machine learning through Pose Estimation into users' daily workout routines [7]. The paper *"AI Tool as a Fitness Trainer Using Human Pose Estimation"*s discusses the challenge of validating exercise posture that users face when performing exercises at home and suggests using Pose Estimation to enable a program to analyze the angles of the user's exercise to check if it is being performed correctly [11].

Like other branches of personal fitness, yoga is increasingly being practiced at home. With this shift, more individuals are turning to online instructional videos to guide them through the complexities of various yoga poses. While these videos are valuable for demonstrating techniques, they fall short of offering personalized feedback. This limitation can hinder a practitioner's ability to correct their posture or fully grasp the nuances of each pose.

Recognizing this issue, numerous studies have explored the application of pose estimation and classification technology in yoga.

The study *Yoga Pose Classification using Angle Heuristic Approach* [10] used pose detection to calculate the angles between landmarks, which could then be used to predict the yoga pose the user was performing. They found that it was quite inaccurate on some poses and suggested using a deep learning network would be more accurate.

In the study *"AI Enhanced Yoga Pose Detection And Alignment Using Deep Learning"* [12], they developed an application that shows when each angle in a pose is performed correctly by marking it green by using TensorFlow MoveNet for pose detection.

Another study called *"Real-time Yoga Pose Classification and Correction: YogaAI"* [22] developed a similar application but also integrated real-time audio and textual feedback to guide the user into better alignment.

While many studies have developed similar technology, we have not found any that have a testing phase and compare it to traditional ways of performing yoga.

Chapter 3

Background

This chapter focuses on selecting four beginner yoga poses for our application, which we will focus on throughout the thesis. Through analysis of correct execution, we establish the ground truth for performing each pose accurately. Additionally, we provide a background explanation of how CNN networks work under the hood to apply pose detection, and we discuss the parameters involved in training a model for pose classification.

3.1 Selecting supported poses for our application

There are hundreds of yoga poses, ranging from beginner to advanced difficulty levels[1]. Since the target group for our application comprises beginners, we have decided to include four distinct beginner poses in our project, as shown in Figure 3.1: Tree Pose, Warrior II, Four-Limbed Staff, and Downward Facing Dog. These poses are among the most commonly practiced in yoga and offer a range of positions from standing upright to lying down.

Figure 3.1: Tree Pose, Warrior II, Four-Limbed Staff, and Downward Facing Dog

[1]Yoga Pose Directory - Ranging from beginner to advanced

3.2 Definition of a correct yoga pose

Before defining how to analyze a yoga pose based on its execution, defining what constitutes a correctly performed yoga pose is crucial. To this end, we will define a correct angle for each joint connection in each pose. For example, when performing the Warrior II pose, it's essential that the angle at the right hip is 90 degrees. To obtain the correct angles for all four mentioned poses, we sourced instructional videos from a YouTube channel that specializes in yoga[2]. We extracted images demonstrating each pose from these videos and then conducted a pose detection analysis to identify all landmarks. Subsequently, we calculated the angles between landmarks. For instance, we calculated the angle at the right knee by drawing a line from the right knee to the right hip and another from the right knee to the right foot. Using these vectors, we calculated the angle, see Figure 3.2 for the final angle analysis on Warrior II.

Figure 3.2: A pose detection and angle analysis done on Warrior II pose.

We completed an angle analysis for all four poses, providing a precise definition of a correct yoga pose. This definition will be instrumental in developing our application (angles for all four yoga poses can be found in Appendix A.1).

[2] Alo Moves - Online Yoga & Fitness Videos - YouTube Channel

3.3 CNN - Convolutional Neural Networks

Pose detection leverages deep learning to identify landmarks on a subject in a photo. Most modern pose detection models utilize convolutional neural networks (CNN), which consist of an input layer, a series of hidden layers, and an output layer (see a visual representation of a CNN model in Figure 3.3.

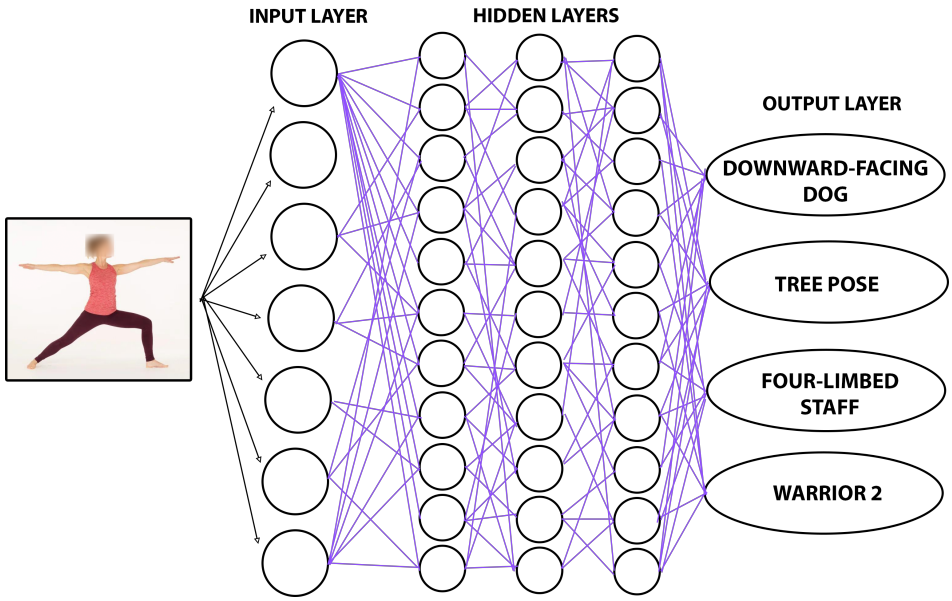


Figure 3.3: A visual representation of a convolutional neural network configured for a pose classifier with four classes.

A feature that classifies the pose a user is attempting to perform is essential for our application. When training a CNN on a yoga image, the input layer would consist of all the information we have on the image, such as the value for each pixel divided into red, green, and blue channels. The output layer recognizes the poses that the model is trained to identify. Within the hidden layers, a sequence of layers processes the input images. Some of the most common layers include filter and pooling layers.

3.3.1 Filter Layers

Filter layers [17] extract features from our input images that may be useful for classification. A filter is a small matrix of weights applied across the image's x and y axes. Common filters, such as edge filters, highlight edges within an image. In Figure 3.4, we apply four edge filters to an input image, demonstrating different edge detections. Each edge filter constitutes its layer.

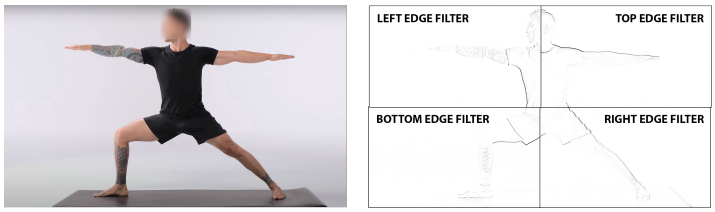


Figure 3.4: An example of Warrior 2 pose with four different edge filters applied to each section to showcase difference.

3.3.2 Pooling Layers

Pooling layers [15] reduce the input image's resolution to decrease computational demands while attempting to preserve as much information as possible. This process might halve the resolution by selecting the highest pixel value in each image quadrant. The example below shows how pooling is

applied (Figure 3.5). This method benefits model training by reducing memory requirements and emphasizing features over spatial resolution. For landmark classification, this allows for effective feature recognition while retaining the ability to pinpoint precise locations in the image after down-sampling.

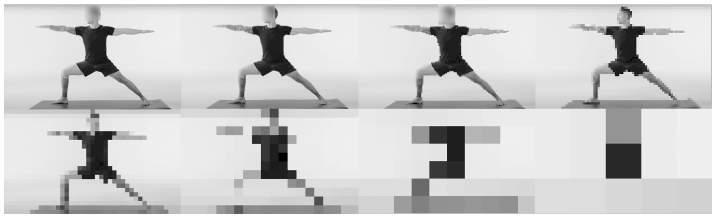


Figure 3.5: An example of Warrior 2 pose with a sequence of pooling layers applied.

3.3.3 Training the model

After defining the hidden layers, the model undergoes training. Each image is assigned a score for each possible classification, determined by the network's weighted connections. Initially, the model cannot identify the pose from the input layer. However, annotating the image as a Warrior 2 image adjusts the network's weights between nodes to maximize the score for that pose. This iterative process, applied across a vast dataset of images, trains the model to recognize patterns associated with each pose [14].

The advantage of using a CNN is its ability to automatically discover the optimal combination of features without manual specification, streamlining the image classification process. It is unknown what specific features the model trains itself to look for when classifying images, but it does not really matter as long as it is accurate.

3.4 Understanding and Analyzing Post-Training Data

When training any neural network, it is essential to gain some insights into how the training went to determine if it was successful or could be optimized. Even more importantly, it is crucial to understand which parameters should be optimized to create more efficient training and produce a higher-quality model. Several quantities can be monitored during a training cycle, giving us different pointers to improve the training process. Some of these quantities are the following plots:

- Loss pr epoch graph
- Accuracy pr epoch graph
- Confusion Matrix

3.4.1 Loss Per Epoch Graph

The loss per epoch graph depicts the loss function. This plot shows the evolution of the model's loss after each training epoch, and the shape of this graph can tell us a lot about the model's training; most importantly, it gives a very clear indication as to whether the learning rate is too high or too low [19].

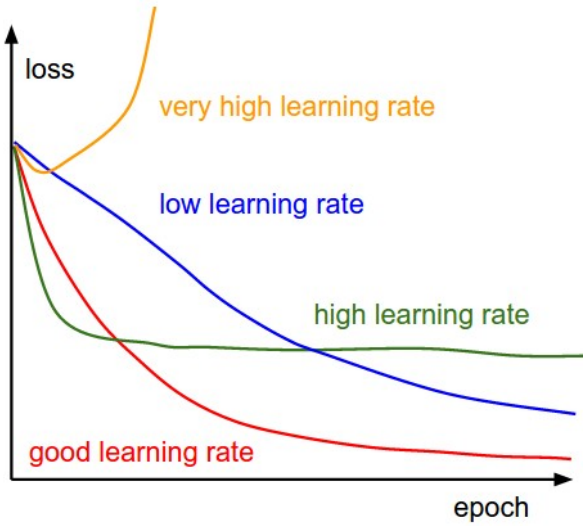


Figure 3.6: Example of different loss plots

In Figure 3.6, several different training cycles are depicted, each showing how the learning rate can affect a model's training. A learning rate that is too high will result in the loss decreasing rapidly in the beginning but will eventually reach a plateau, where the loss becomes linear. This is because the weights in the model are changed too drastically, resulting in the training being unable to fine-tune the weights effectively. A low learning rate will make the loss decay less rapidly and will, therefore, require a higher number of epochs because the model can fine-tune the weights with the less drastic changes, which will result in the loss being able to come closer to zero. A good learning rate will be a mix of high and low learning rates with a more rapid development at the beginning of the training but still on a level where the weight is being changed less chaotically [9].

3.4.2 Accuracy Per Epoch Graph

Another useful plot is the accuracy per epoch graph; this graph shows how the model's accuracy on the training and test data evolves through each epoch. The difference between the training and validation accuracy plots can give an indication of whether the model is overfitting or underfitting.

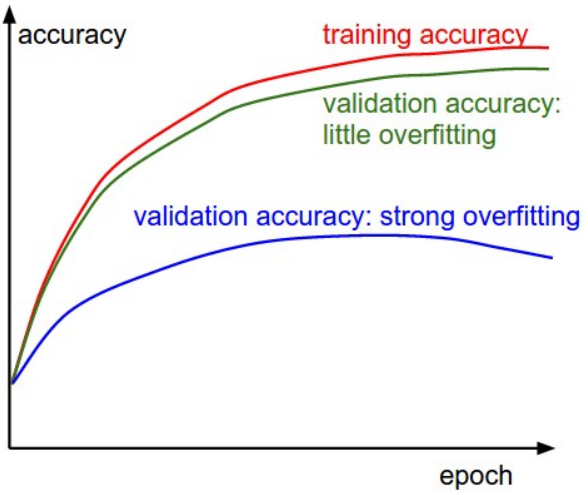


Figure 3.7: Example of different accuracy plots

Figure 3.7 shows different examples of accuracy evolving through epochs. The important thing to note in this kind of graph is how high the accuracy is on both the training and validation data. However, one should check the gap between the two because this gap indicates whether the model is overfitting

or underfitting. High accuracy on training data but lower accuracy on validation data suggests slight overfitting, and the larger the gap, the stronger the indication of overfitting. It is also possible for the validation accuracy to start dropping; in such cases, several techniques can be employed to handle this issue. Although no example of underfitting is shown, such a graph would typically show higher accuracy on validation data than on training data. Generally, when a model shows signs of underfitting, it may be because it has not been trained through enough epochs, or the model is not large enough, meaning its number of input features should be increased. When the model shows overfitting, adjustments could include reducing training time, decreasing the number of feature inputs, and more [9].

3.4.3 Confusion Matrix

The confusion matrix is a popular tool used to understand why and when models fail at image classification tasks. It is applicable to both binary and multi-class classification problems. As the name suggests, the matrix illustrates instances when the model confuses one class with another in the dataset, thereby providing insights into why and when classification errors occur—insights that an accuracy-per-epoch graph may not offer. The matrix is straightforward to interpret. Figure 3.8 presents a simple example of a confusion matrix. Here, the row names reflect the classes predicted by the model, and the column names indicate the ground truth classes. The entries in the matrix reveal the count of whether the prediction aligns with the ground truth or which class was incorrectly predicted, as exemplified by 'TP' (true positive) for correct predictions and 'FP' (false positive) for incorrect predictions where a negative is confused with a positive class. Although the figure only shows a binary class confusion matrix, it can easily be expanded to a multi-class matrix by adding rows and columns for additional classes.

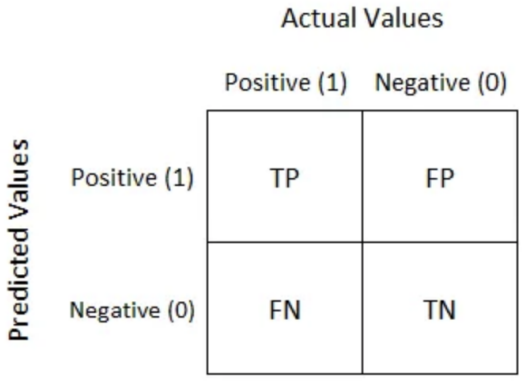


Figure 3.8: Simple example of a confusion matrix

While the confusion matrix builds upon the accuracy graph by helping us understand why the model fails, it does not provide additional information about the training itself that the accuracy graph does not already reveal. Therefore, it does not directly suggest which training parameters might need adjustment. However, it can offer insights into potential improvements in the dataset. For example, if 'Positive' is frequently confused with 'Negative,' and the dataset contains more positive data than negative, this could indicate an imbalance in the dataset. Similarly, if the model is trained to detect classes that are very similar to each other, it may confuse these classes. In such cases, the dataset might need to be expanded or refined to better differentiate between these closely related classes [1].

Chapter 4

Selection of the Optimal Pose Detection Model for Yoga Pose Landmark Detection

In this chapter, our objective is to explore the application of pose detection models for analyzing physical postures, focusing on the process from data input to pose analysis. With numerous pose detection models available, each varying in performance, accuracy, and ease of implementation, our objective is to find the most prominent ones to pick the most suitable model for our project. We will explore the strengths and limitations of these models, paying particular attention to their efficacy in interpreting yoga poses. By identifying potential challenges specific to yoga, we hope to gain insights into the optimal application of pose detection technology in this context.

4.1 Pose Detection Models

Pose detection technology has emerged as a pivotal tool in computer vision, enabling computers to identify and track human body postures and movements in images and videos. At its core, pose detection involves the identification of key body joints (landmarks), thereby enabling the model to detect the human pose [13]. Various domains increasingly utilize pose detection in the upcoming meta-verse industry to create digital models of oneself and in various physical fitness applications to enhance exercise techniques and precision.

There are many different pose detection methods, each with pros and cons. We will focus on three popular models MediaPipe, PoseNet, and MoveNet.

4.1.1 MediaPipe

The MediaPipe library features several machine-learning models, including a pose detection model known as Pose Landmark. This model can detect 33 key points, and unlike other models, it also includes points for hands and feet. The model is available in three versions: Lite, Full, and Heavy. The Lite version is designed to run on low-powered hardware. In contrast, the Heavy version offers the most precision but requires more computational power.

4.1.2 PoseNet

PoseNet is another creation by the Google Creative Lab team and TensorFlow.js. However, the continued development of the model relies on the developer community, as it is an open-source model. This model uses TensorFlow to detect 17 landmarks on the human body. Its efficient architecture allows it to run on low-powered devices such as mobile phones, making it the default model of the Core ML library for Apple development[1]. PoseNet has two different models called PoseNetSlow and PoseNetFast, one version with precision in mind and the other with performance.

[1]Detecting human body poses in an image, Official Apple Developer Documentation on Core ML

4.1.3 MoveNet

MoveNet is an upgraded version of PoseNet. According to TensorFlow, it performs better than PoseNet, especially in videos with quick movements like boxing[2]. Like PoseNet, MoveNet comes in two versions: one that prioritizes speed, and another that prioritizes accuracy.

4.2 Ground Truth Dataset

A ground truth dataset is essential for comparative analysis to accurately evaluate each of our selected pose detection models. This dataset must consist of images with accurately annotated landmarks to serve as a benchmark. The *"MPII Human Pose Dataset"* [3] is ideal, offering 25,000 images with meticulously annotated landmarks across various activities. The subset of 283 yoga images within this dataset fascinates our project.

We excluded images featuring multiple individuals to tailor the dataset to our focus on single-person pose detection. This refinement resulted in 238 yoga images suitable for our benchmarking needs (see Appendix A.2 for example images). Upon closer inspection of this subset, we discovered that some images lacked tracked values for specific landmarks, which could skew our analysis. To maintain the integrity of our benchmarking process, we decided to exclude these images from our dataset. This decision further refined our dataset to 212 yoga images. These images incorporate various conditions, from professionally lit, high-resolution scenes to lower-resolution images captured in poorly lit environments with potential obstructions. This diversity ensures a comprehensive evaluation of the pose detection models under varying conditions.

4.3 Testing Framework

Precision is of the greatest importance in our project as we assess the alignment of a user's yoga pose with the ideal execution. While the performance capabilities of various models—measured in the number of images processed per second—are notable, they hold less significance for our application due to the gradual nature and static endpoints of yoga poses.

The Percentage of Correct key points (PCK) is crucial in evaluating pose detection models[3]. This metric gauges the accuracy of predicted landmarks by determining whether they fall within a predefined proximity to their true positions, with the threshold adjusted according to the size of the subject's body. For our analysis, we use the PCKh@0.5 standard, where we deem a landmark prediction accurate if it deviates from the actual point by less than or equal to 50% of the head segment length.

Crucially, our ability to apply the PCKh@0.5 metric accurately stems from the detailed annotations in our ground truth dataset, particularly the precisely tracked positions of the upper neck and the top of the head for each subject. This data allows us to calculate the head length for each individual in the dataset, providing a personalized scale against which to measure the accuracy of landmark predictions. This approach ensures a nuanced and tailored assessment of model precision in capturing the intricate details of yoga poses.

4.4 Model Evaluation

To assess the performance of each pose detection model, we devised a systematic approach involving the creation of a script that processes each image in our dataset. This script applies pose detection to extract the coordinates of each landmark, recording them in a format consistent with the MPII Dataset [3]. This process resulted in seven distinct spreadsheets corresponding to each version of the models under review.

[2]Pose estimation, Tensorflow Official Documentation
[3]Percentage of correct keypoints (pck) - OECD

Subsequently, we developed another script designed to calculate the deviation of each detected landmark from its ground-truth counterpart, employing the Euclidean distance formula for precision[4]. This step produced additional spreadsheets, illustrating the discrepancies between the detected landmarks and their true positions.

The core of our evaluation lies in determining the accuracy of these landmarks relative to the PCKh@0.5 metric, which considers a landmark accurately detected if its distance from the true position is within half the length of the subject's head. To achieve this, our final script reviews each spreadsheet, calculating the head length for each subject to establish the PCKh@0.5 threshold. It then evaluates whether each landmark falls within this threshold, marking each as `true' (within the threshold) or `false' (outside the threshold). Each spreadsheet, now enhanced with this binary evaluation for each landmark, also includes a column indicating whether all landmarks in an image meet the accuracy threshold, signifying a 'perfectly tracked' pose.

These enriched spreadsheets (one for each model variant) allow our analysis to proceed. They allow us to dissect each model's precision in detail, revealing which model achieves the highest fidelity in tracking yoga poses across our dataset.

4.5 Analysis and Results

4.5.1 PCKh comparisons

In our comparison of MediaPipe, PoseNet, and MoveNet, significant variance in accuracy emerged, particularly evident in the PCKh@0.5 scores. PoseNet's fast variant demonstrated the least precision, achieving a perfectly tracked pose in only 1.4% of the dataset. In contrast, MediaPipe's fast version outperformed PoseNet, tracking 56.6% of poses perfectly. MoveNet also surpassed PoseNet, with its fastest variant registering 37.3% of poses as perfectly tracked. Notably, the most accurate performance was observed in MediaPipe's slowest version, which successfully tracked 80.2% of the images accurately (Figure 4.1).

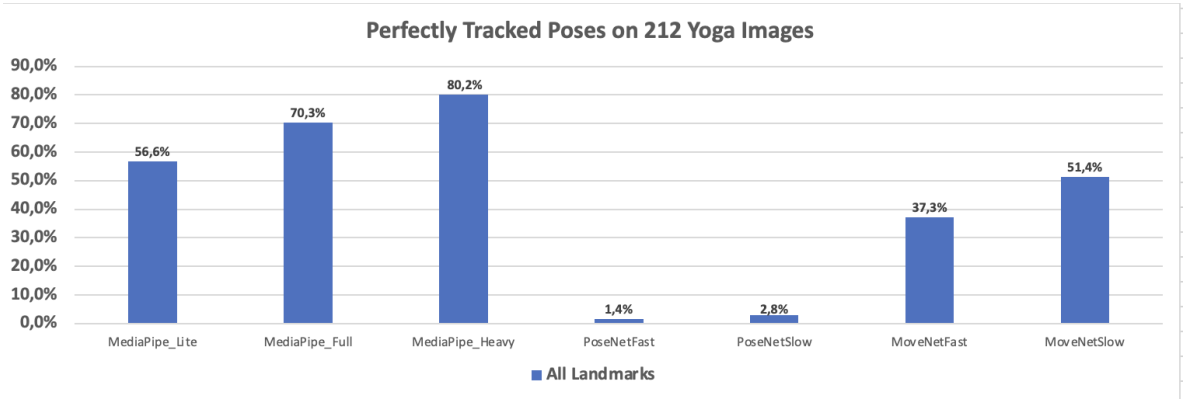


Figure 4.1: Comparison of perfectly tracked poses between each model

[4]Euclidean Distance Calculator - OMNI

4.5.2 Examples of pose detections failing

Figure 4.2 illustrates the instances where pose detection can fail significantly. For example, in Figure 4.2a, the detection model seems to misconstrue the subject's posture, mistaking a handstand for a standing position, which leads to the arms being incorrectly identified as legs. Figure 4.2b shows a nearly accurate detection at first glance. Still, a detailed look reveals a notable mistake: the model confuses the right leg with the left. These examples highlight the importance of choosing an exact pose detection model to avoid such inaccuracies, which could otherwise negatively impact the user experience.

(a) Pose detection failing on a handstand

(b) Pose detection mixing up left and right foot

Figure 4.2: Example of pose detections with MediaPipe Heavy failing

4.6 Conclusion

This chapter aimed to identify our application's most suitable pose detection model, prioritizing model precision. After thorough testing, it's evident that MediaPipe's models significantly outperform those of both PoseNet and MoveNet. Even MediaPipe's lightest model surpasses the performance of the heaviest models from PoseNet and MoveNet. Hence, opting for a MediaPipe model is undoubtedly the optimal choice. However, we must consider which version of the model to utilize. A comparison between the lite and full versions reveals a notable increase in precision. Similarly, transitioning from the full version to the heavy version yields further improvements in precision. Yet, it's crucial to note that with each step towards a heavier version, there's a corresponding dip in the model's performance. Accuracy will be a more important factor than performance for our application since yoga practice consists of a slow movement into a static pose. From this movement forward, we will use the MediaPipe Heavy version in our application.

Chapter 5

Understanding Pose Detection with BlazePose

The BlazePose model processes an input image to output coordinates for each body landmark. This pose prediction involves two main phases: detection and tracking [21].

During the detection phase, the model establishes a spatial baseline for the analyzed human. This step is crucial to avoid searching for joints in image regions where a human is absent. The initial goal is to locate the body's center point, typically around the mid-torso, providing a rough estimation of the person's position in the image. As seen in Figure 5.1, you can visualize it by drawing a circle with a radius from the torso center to the bottom of the feet—akin to the concept of the Vitruvian Man—the model estimates the body's positioning, scale, and orientation based on the torso-to-head line. The BlazePose team found that the person's face was the most vital link between the torso and its position.

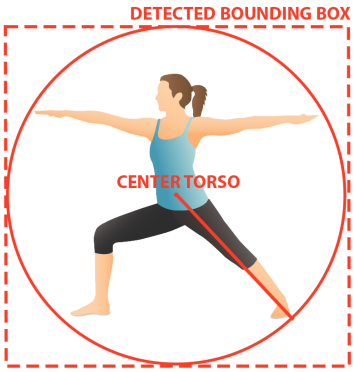


Figure 5.1: Pose detector finding a bounding box around a person based on the Vitruvian man.

Once the detection phase is complete, the model transitions to the tracking phase. Here, a cropped image is analyzed using a model trained by the BlazePose team with a dataset of 85,000 images depicting various human poses, each manually annotated. The model generates a heat map for each landmark, offering a general location estimate. This is refined through regression and offset adjustments to pinpoint each landmark's precise coordinates [21]. Inspired by the Stacked Hourglass Network architecture [2], this method employs an efficient pose estimation technique. It involves down-sampling the image to reduce resolution and increase feature map depth. The image is then up-sampled to get a better spatial overview of each recognized landmark, re-utilizing information from the down-sampling phase while filtering out irrelevant data. This iterative down-sampling and up-sampling process ensures detailed landmark recognition while maintaining low memory usage.

Chapter 6

Training a Yoga Pose Classification Model

Before we can assess the quality of a pose, we must establish a method to detect what pose the user is attempting. This task entails training a pose classification model capable of identifying the pose done in an image. To accomplish this, we've utilized Teachable Machine by Google Creative Lab[1]. Teachable Machine provides a user-friendly web application for training a pose classification model, enabling us to furnish custom training data for each classification the model should identify.

6.1 Collecting the initial training data

Before training our pose classification model, we must gather the training data for the selected poses as shown in Figure 6.1. Given the absence of existing datasets, we found an automated Python scraper to collect images[2]. This tool collects the initial datasets from online sources via Google Images. It operates by running a thread for each yoga pose, searching for their names on Google Images, and downloading the first 300 images it encounters. Following the initial data collection, we needed to normalize the dataset by excluding all images that did not meet the following criteria: correct pose, good quality, and only one person in the picture.

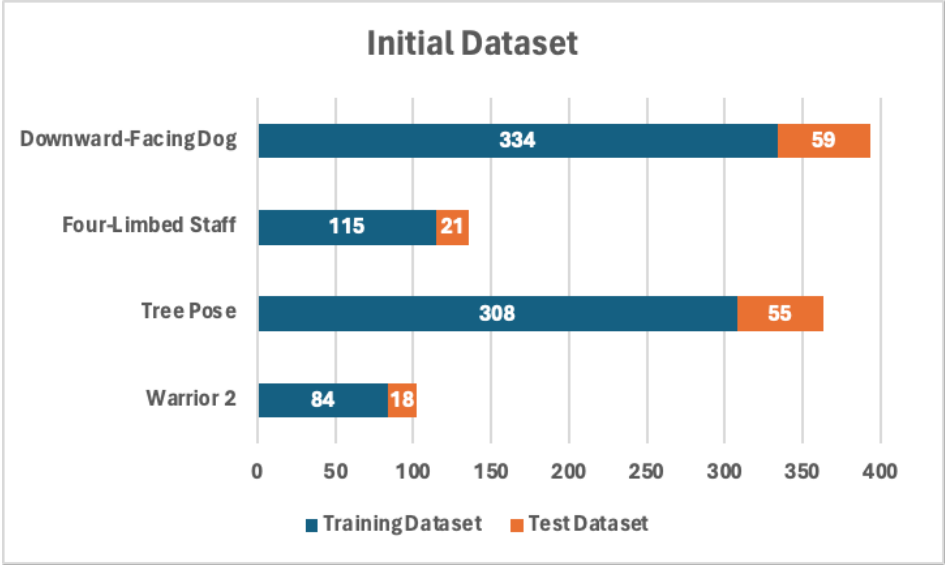


Figure 6.1: The image count for each pose in the initial training dataset

[1]Teachable Machine, Google Creative
[2]Google Image Scraper, Jayden Oh Yicong

6.2 Teachable Machine

Google Creative's Teachable Machine offers a user-friendly tool for training models to recognize diverse poses. This model actively employs the landmark output generated by a pose detection model to predict the corresponding poses. The pose classification model comprises two submodels, the first calculates a pose embedding, also referred to as a feature vector, from the provided landmarks. Subsequently, the final submodel processes these pose embeddings through a trained neural network, generating predicted labels for the poses depicted in the images.

Training a model to detect desired poses consists of three parts. Firstly, users upload the training data to the application via their website. Then, the test data must be labeled with the ground truth pose depicted in each image. The application proceeds to feed all images through a pose detection model to identify all relevant landmarks, thereby generating the necessary training data for the new model. A notable feature of the application is its automatic splitting of the dataset into training and test data, with 85% of the samples allocated for training and the remaining 15% reserved for testing the model. Notably, the model is not trained on the testing dataset, ensuring its independence from the training data. This separation enables the evaluation of the model's performance on unseen data, facilitating an understanding of its generalization capabilities.

Before training commences, users must define the general settings for training, which consists of the following parameters:

- **Epochs**: This parameter determines how often the model is trained on the dataset. The default value is 50, implying that the model is trained on each image 50 times. Increasing this number typically enhances the precision of the model at the expense of longer training times.
- **Batch Size**: This parameter dictates how many batches the training data will be divided into. While not directly impacting the quality of the final model, it affects the efficiency of the training process.
- **Learning Rate**: This parameter regulates the magnitude by which the weights are adjusted throughout the model's neural network. Adjusting this parameter can significantly influence the model's training dynamics and convergence speed. A low learning rate lets the model make more fine adjustments on weights, whereas a high learning rate makes more significant weight changes.

The pose classification model can be trained when the training data has been uploaded, and the settings are set. After the training is completed, the application also provides data that describes how the training progressed and information on the new model's performance on both the test and training data.

6.3 Initial model training

Our initial training round was done using the tool's default settings. This was done to get a feel for what we needed to tweak, if any, to produce the best-performing model. After the initial training, we found that the model was more accurate on the poses Downward-Facing Dog and Tree Pose (see Figure 6.2). These poses were also the ones with the most images in our unbalanced dataset, indicating a need to balance the dataset, by adding more images for the other poses.

Accuracy per class

CLASS	ACCURACY	# SAMPLES
Downward-Facing Do...	0.98	59
Four-Limbed Staff	0.90	21
Tree Pose	0.96	55
Warrior 2	0.94	18

Figure 6.2: accuracy per class after first training run

We analyzed how the model's loss and accuracy changed after each training epoch to gain deeper insights into what needed to be adjusted to enhance our model's performance. When examining the evolution of accuracy during training on both the test and training data, the model exhibited high performance from the onset of training, achieving accuracy levels above 80% on both the test and training dataset from the first epoch. The model's accuracy showed significant improvement in the initial stages of training. After the first ten epochs, both accuracies converge to a relatively stable value, indicating that further increasing the number of epochs would not likely yield substantial performance gains with this learning rate and training dataset (see Figure 6.3). The close alignment between the training and test accuracy suggests that the model does not significantly overfit the training data. This suggests that the model's complexity is appropriate for predicting the poses it was trained on.

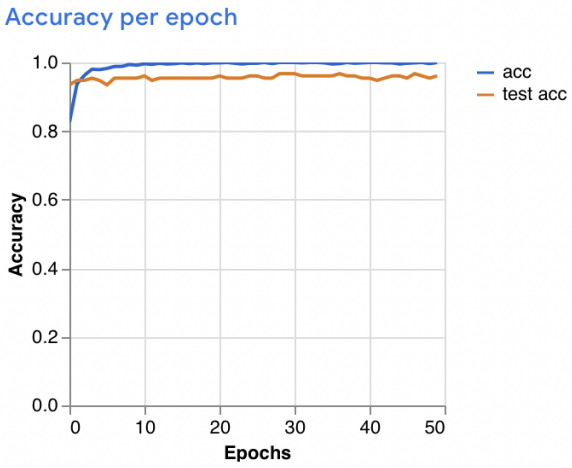


Figure 6.3: Accuracy per epoch on the test dataset after the first training run.

When analyzing how the Loss changed after each epoch of the training, we get some of the same indications that the model is a good fit. The loss decreases significantly in the initial training stage until it stabilizes after the first ten epochs. There are some significant fluctuations in the test loss, but this is to be expected when the model is exposed to unseen data. However, the significant fluctuation still indicates some improvements to be made likely due to unexpected noise in the dataset. The loss on the test data remains significantly higher than the training loss, indicating that the training was conducted with a high learning rate, which could also cause fluctuations (see Figure 6.4).

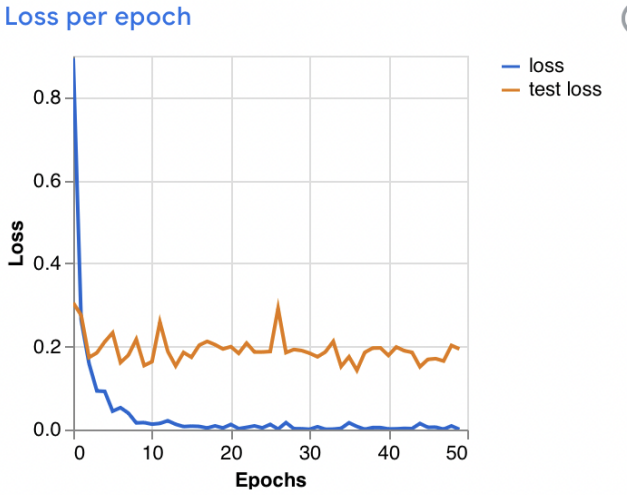


Figure 6.4: Loss per epoch after the first training run.

To understand the reasons behind prediction errors in the model, we must analyze the confusion matrix generated from the training data (see Figure 6.5). This matrix reveals misclassifications between similar poses, such as mistaking the Downward-Facing Dog for the Four-Limbed Staff or confusing the Tree Pose with the Warrior 2. Such errors are expected due to the resemblance between these poses.

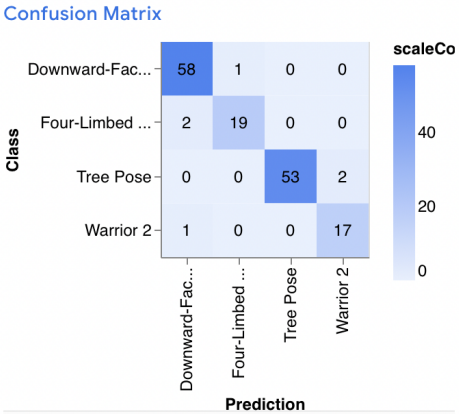


Figure 6.5: confusion matrix on the test dataset after the first training run.

However, an anomaly exists within this pattern, where the Warrior 2 pose is misidentified as the Downward-Facing Dog. We hypothesize that this outlier may stem from a flawed test case. Our observation suggests that the model favors predicting the Downward-Facing Dog, even when presented with a blank image or an image not depicting any target poses (refer to Figure 6.6). This could also be explained simply by the fact that models from Teachable Machines tend to favor the first declared class. This bias could contribute to misclassifying the Warrior 2 pose as the Downward-Facing Dog.

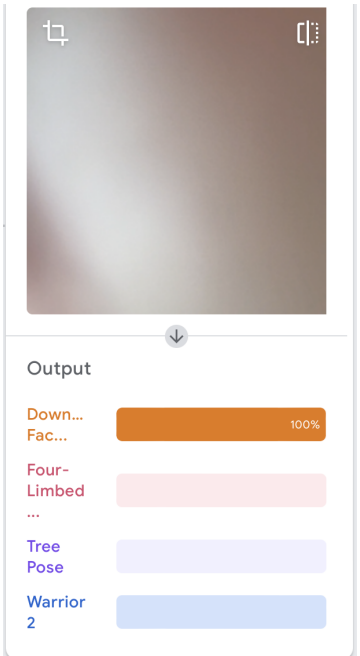


Figure 6.6: Image of model predicting downward facing dog on black input

6.4 Improving training setup

Our initial training round analysis revealed several indications that our model could be enhanced. The most prominent indication was the need to expand our training dataset to include more images of Warrior 2 and Four-Limbed Staff poses. We expanded the database by following the same practices used during the collection of the initial dataset. This resulted in a more comprehensive set of images for each pose, now consisting of over 340 images each (see Figure 6.7).

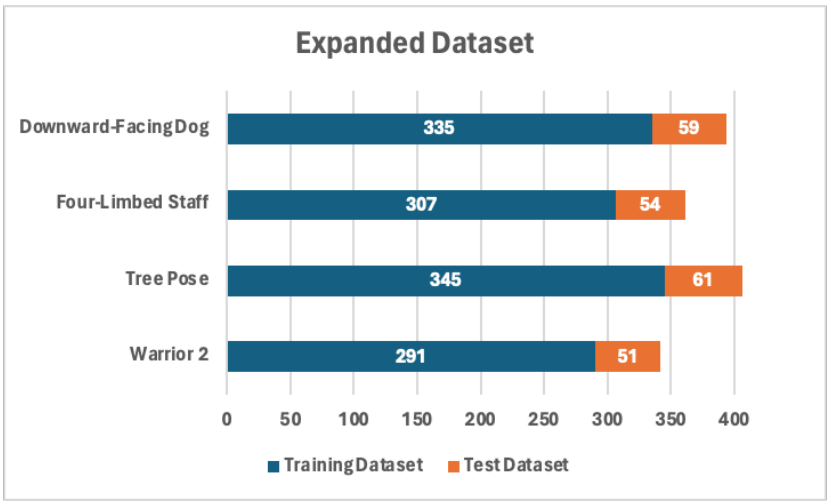


Figure 6.7: The image count for each pose in the expanded training dataset

As the loss per epoch graph from the initial training round (see Figure 6.4) indicated that there might be some unexpected noise in our dataset, we also chose to make extra quality control of the dataset. Many images, especially the pose Four-Limbed Staff, were cut unintendedly. The cause for this is that the Teachable Machine tool crops images to make them square. This made the pose no longer visible on the image (see Figure 6.8). To fix this, we created a Python script that iterates over all images for each pose and added a colored padding to the images to make them square (see Figure 6.8).

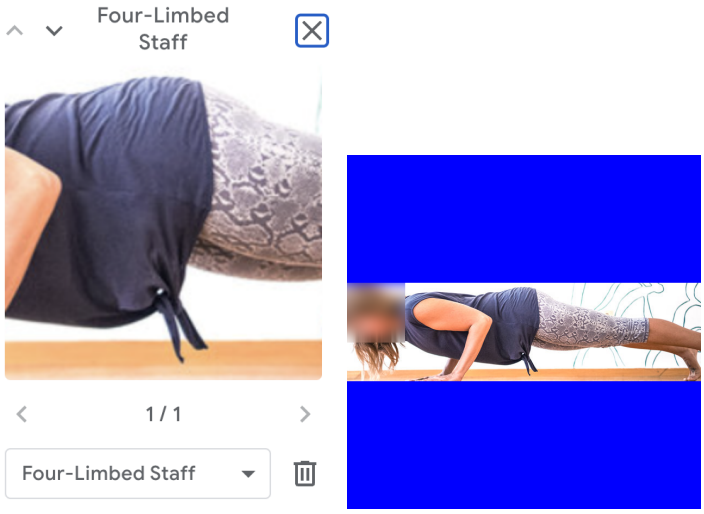


Figure 6.8: Case where images get unintentionally cropped and how it was fixed

We increased the next model's accuracy by making these changes and running a new training round while removing the most significant spikes in the loss per epoch graph. However, as we still saw a large gap between the test and training loss and some significant spikes in the test loss, we wanted to try to tweak the learning rate. The tweaking of the learning rate was an iterative process until we found

the most optimal setting, considering that a lower learning rate naturally requires a larger number of epochs. We found an optimal learning rate of 1e-6, which also needed more epochs, and we set it to 200. This learning rate removed all significant spikes on the loss per epoch graph, leading to a higher accuracy scoring higher than 97% on all poses (see Figure 6.9).

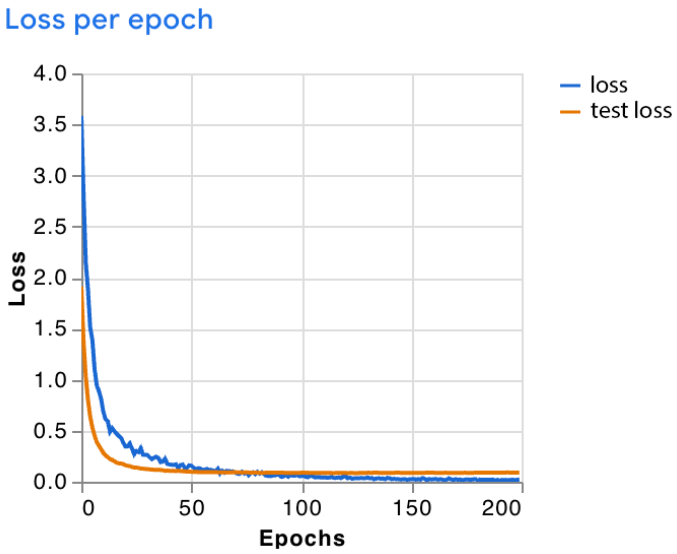


Figure 6.9: Training running with a 1e-6 learning rate in 200 epochs

6.5 Live Testing

It is crucial to recognize that the accuracy metrics obtained post-training reflect the model's performance on the specific dataset and indicate potential real-world application effectiveness. Ultimately, the efficiency of a neural network-based model is closely tied to the quality of the data it has been trained on. So, to check if the model also performs as expected in a real-world example, we created a testing video with videos of all four poses to check if the model can predict the correct pose and how it performs in different use cases. With this test, we found that for the pose Four-Limbed Staff, the model performs differently when the pose is done to the right side than if the pose was to the left (see Figure 6.10).

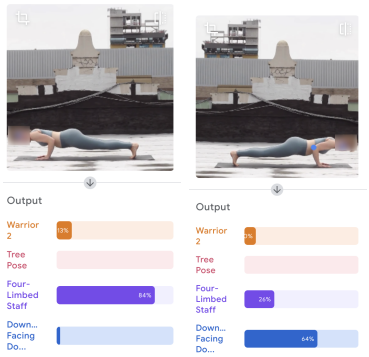


Figure 6.10: Different prediction for left and right Four-Limbed Staff poses

This discrepancy is unintentional because all of our poses can be performed on both the left and right sides, but the error stems from the dataset containing a larger number of images for one of the sides. To counter the unbalanced dataset, we implemented a new Python script to flip all images horizontally, ensuring a balanced dataset and effectively doubling the dataset for all poses. Before

rerunning the training, we also chose to add extra epochs, extending the training to 300 epochs. The new balanced dataset and the extended number of epochs produced a new model with slightly higher accuracy and lower loss (see Figure 6.11).

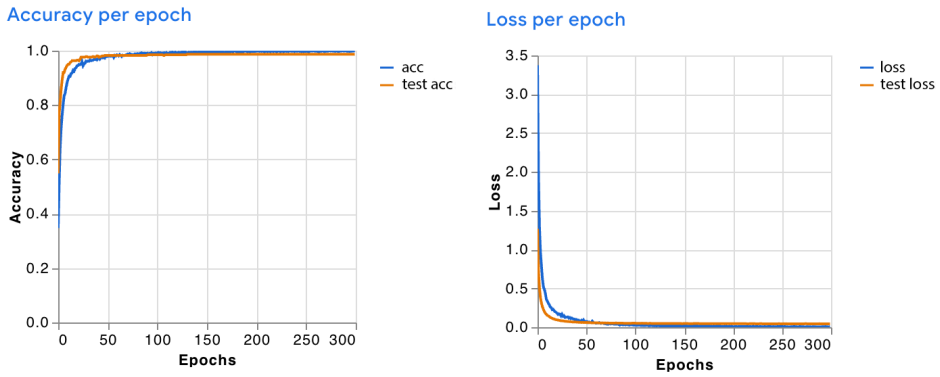


Figure 6.11: The accuracy and loss graph for a dataset with flipped images and 300 epoch

The final training also included flipped images, and we observed the optimized model performing on a Four-Limbed Staff in both directions (see Figure 6.12).

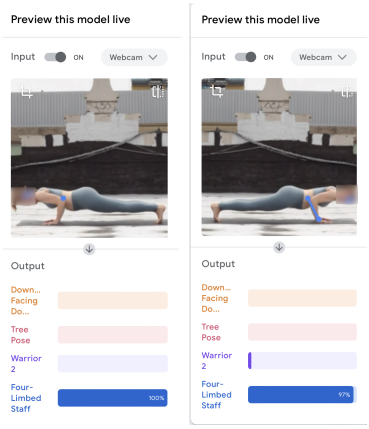


Figure 6.12: Optimized model, predicting the same for left and right Four-Limbed Staff poses

6.6 Conclusion

We set out to train a pose classification model by assembling our dataset through web scraping, ensuring a comprehensive collection of necessary poses. Leveraging the Teachable Machine tool by Google Creative Lab, we benefited from an intuitive training environment that facilitated quick and straightforward model training and provided critical insights into the training process, aiding our model's enhancement. However, a notable limitation of utilizing such a tool was the reduced ability to fine-tune the training process. Specifically, we could not explore various adaptive learning rate techniques, which adjust the learning rate dynamically during training, offering potentially optimized training dynamics. Despite this, the produced model exceeded our requirements for our project's goal of analyzing yoga poses through landmark angles.

Chapter 7

Showcasing the Application

We decided to implement the application as a web app using React [6]. This decision was made because MediaPipe's version of BlazePose, which is faster than TensorFlow's version, only runs on WebAssembly, and WebAssembly can only be executed in a browser. Therefore, to leverage MediaPipe's efficiency, we opted for a responsive web application.

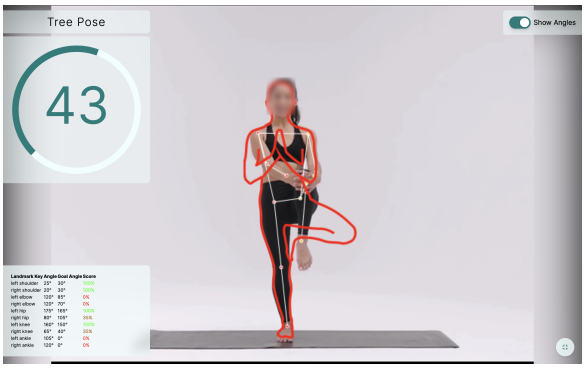
7.1 Integrating MediaPipe BlazePose

TensorFlow has built a simple demo application in pure JavaScript, allowing toggling between MoveNet, PoseNet, and BlazePose (both MediaPipe and TensorFlow versions). This demo takes images from the webcam, performs pose detection on each frame, and overlays a skeleton on the webcam feed (see Figure 7.1). As our focus was solely on MediaPipe BlazePose, we removed all unnecessary components and other model supports and converted the project to TypeScript for easier maintenance during development.

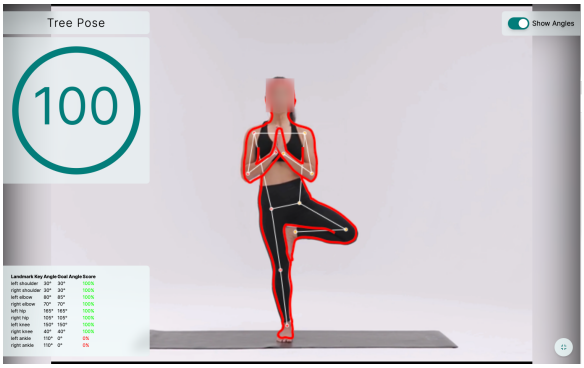
Figure 7.1: Initial setup of our app with pose detection implemented

7.2 Score Evaluation

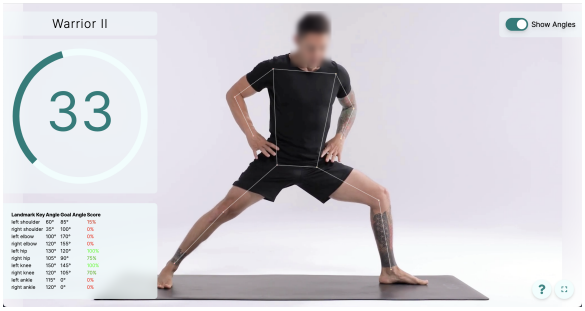
As described in the background chapter, we defined a map of correct angles between landmarks for each pose. After pose detection on a frame, we calculate the angle between landmarks on the subject and compare it, percentage-wise, to the predefined correct angle. We obtain an overall accuracy percentage for the performed pose by averaging all combined scores. Recognizing the difficulty of achieving 100% accuracy in all angles, we set an offset threshold of 15 degrees per landmark. Figure 7.2 shows the score evaluation clearly indicates how the pose is performed.



(a) Tree pose evaluation on 43% score



(b) Tree pose evaluation on 100% score



(c) Warrior 2 pose evaluation on 33% score

(d) Warrior 2 pose evaluation on 100% score

Figure 7.2: Showcase of the score evaluation on different poses

Beneath the evaluation score, we have added a debugging window that can be toggled and displays all calculated angles. This can show in greater detail which angles are incorrect and affecting the score negatively. This menu has also been very helpful when analyzing our test participants later in Chapter 8.
We excluded specific landmarks from calculations on all poses for more consistent results. For example, in the Tree Pose, we omitted the angle landmarks from the calculation because it's very difficult to track the right foot precisely when facing the camera directly.

7.3 Pose classification

To ensure the user fits within the webcam's view, they typically need to stand 2-3 meters from the device. We implemented pose classification to eliminate the need for screen interaction for each new pose, allowing the app to predict the desired pose automatically.

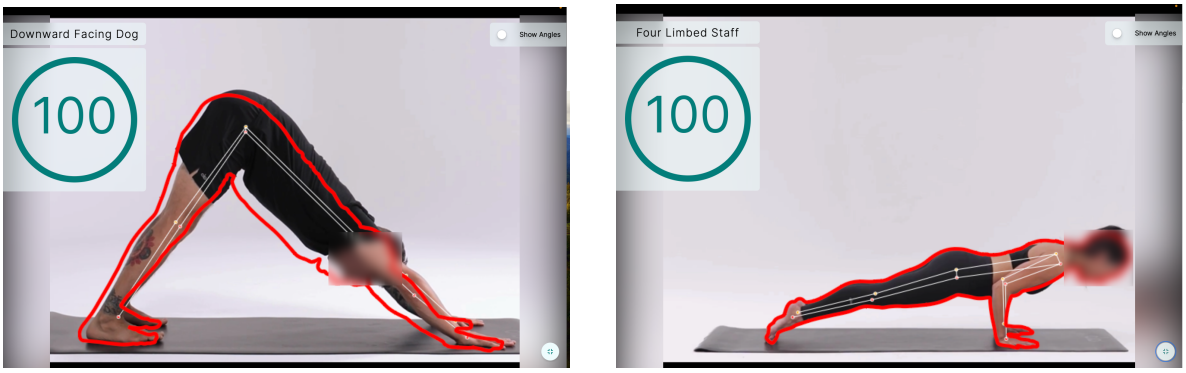
After training our model on the Teachable Machine platform, we converted the provided JavaScript code to TypeScript and integrated it into our application. Each frame is processed for pose detection and classification by our pose classifier component, which returns the identified pose. We only accept a pose classification if the evaluation score is above 35%. This threshold adds a layer of verification, ensuring the subject is attempting the classified pose. We also require the same pose to be returned five times consecutively to prevent erratic switching between poses due to incorrect frame detection.

7.4 Pose Overlay

By now, our application can provide an accurate score reflecting how precisely a user is performing a pose. However, it would be quite challenging for a beginner to improve their technique solely based on the score, as there is no guidance on achieving a better score.

To address this issue, we have introduced a feature that overlays an SVG image, representing the correct pose, on top of the user's image. We generated outlines from each reference image used to calculate the correct landmark angles and converted them into SVG format. This ensures the overlays can scale to any resolution without losing quality.

Since we already track the user's landmarks, we can anchor the SVG overlay by using the coordinates of either a foot or the head to adjust its position on the x and y axes, ensuring it aligns with the user's pose. Additionally, with the nose coordinate, we can estimate the user's height in the image. This measurement allows us to scale the SVG overlay accordingly. Figure 7.3 shows the red line represents the pose done correctly.



(a) Downward Facing Dog SVG overlay

(b) Four-Limbed Staff SVG overlay

Figure 7.3: SVG overlay on the Downward-Facing Dog- & Four-Limbed Staff pose.

7.5 Conclusion

After assembling all the components, we created a functional web application that leverages modern pose detection and classification models to guide yoga practitioners toward better pose alignment. The application provides instant feedback during movement and gives a precise score on how close the alignment is to the correctly performed pose. The complete code for the application and a live demo can be found in Appendix A.6.

Chapter 8

Evaluating our Yoga Application: App versus Video Instruction

This chapter presents a comparative analysis of our yoga application against traditional video instruction through a structured test involving 12 participants. We will examine the effectiveness of each method in aiding accurate yoga poses and gather participant feedback to refine our application.

8.1 Designing Our Test

We have created a split test in which all participants will attempt to perform the four poses we have implemented in our application. As we discovered in Chapter 1, most people practice yoga at home using video streaming. Therefore, half of the participants will practice these poses using a short video demonstration by a yoga instructor, while the other half will use our yoga application. The videos chosen for the test are the same ones from which we extracted the angles for a correct yoga pose in Chapter 3, ensuring we can precisely compare the two test groups. We will record each participant while performing the poses, which will later be analyzed with pose detection to see how closely they match the intended poses, similar to what is done in our application. After each test, we will conduct a short survey to understand how our application might benefit or detract from the yoga experience and to identify potential improvements. A full description of our test phase can be seen in Appendix A.3.

We conducted our test phase with 12 participants. Although this number is insufficient to confidently determine whether our application is a better alternative to practicing yoga via video streaming, we hope it will provide us with indicators of how the application can be used for yoga and what improvements are necessary to enhance the experience.

8.2 Test Results

In this section, we will begin by evaluating all the test footage recorded from each participant and compare Group A with Group B to see how well our app performs relative to using video instructions. Afterward, we will review all participants' surveys to determine what works well in our application and what can be improved.

8.2.1 Evaluating our footage from all test participants

During our test phase, we gathered significant information on how our yoga application enhances the experience and where it falls short. After inputting each video recorded by each participant into our application, we reviewed the footage and noted all instances when each participant received the highest score. We analyzed the overall dataset after compiling all the data into a spreadsheet.

Our test results appear positive, showing that Group B, which practiced yoga using our application, received an equal or higher score on all four poses performed (see Appendix A.4). Figure 8.1 shows the average score on all yoga poses for Group A was $75\% \pm 6\%$, whereas Group B had an average score

of 85% ± 7%, representing a 14.86% increase. The largest improvements were seen in the Warrior 2 pose (48.33% increase) and the Tree pose (12.94% increase).

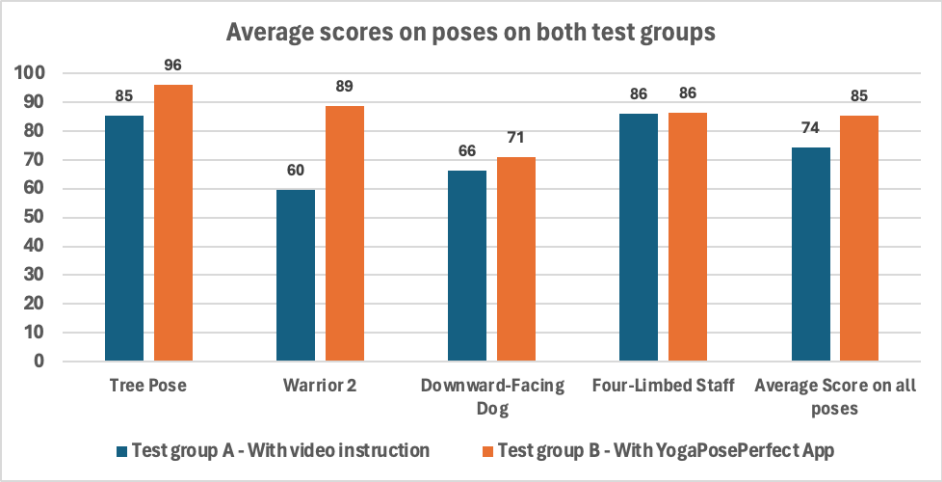


Figure 8.1: Compared average scores on poses from both test groups

In the Tree Pose, as shown in Figure 8.2, the average participant from Group A, who used video instruction, did not lift their right leg high enough, resulting in a 35% higher angle degree at the right knee, whereas the average participant from Group B, who used the app, exhibited only a 13% higher angle degree at the right knee. It's also evident from the graph that participants in Group A are flaring their left elbows too far out from the body compared to those in Group B, as concluded from the left shoulder angle. This may be because individuals are flaring the left elbow to balance the right knee flaring out to the opposite side. However, when using our app, it is clear that you should shift your hip to the left to balance the body instead of adjusting your elbow.

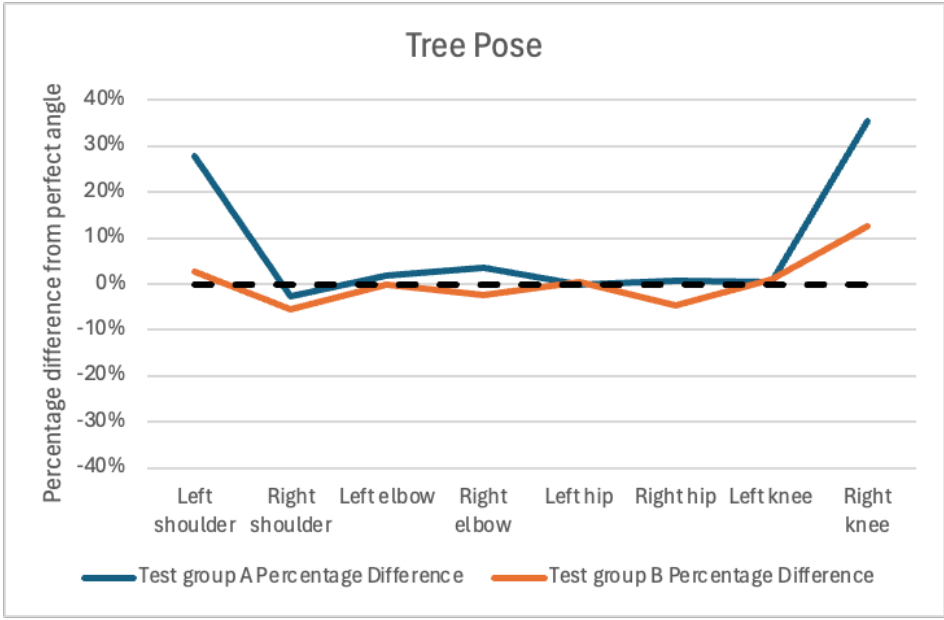


Figure 8.2: Average percentage difference in each angle when performing the Tree Pose, comparing Group A using video instructions to Group B using our application. The expected percentage difference in all angles is 0%. Data is derived from Figure A.4 in the appendix.

In the Warrior 2 pose, as shown in Figure 8.3, the average test subject from Group A, who used video instruction, did not spread their legs out far enough, which contributes to a 24% higher angle degree at the right hip, whereas the average test subject from Group B, who used the app, has only a 4% higher angle degree at the right hip. It's also visible on the graph that the average test subject in Group A isn't raising their arms high enough compared to Group B, as indicated by the shoulder angle.

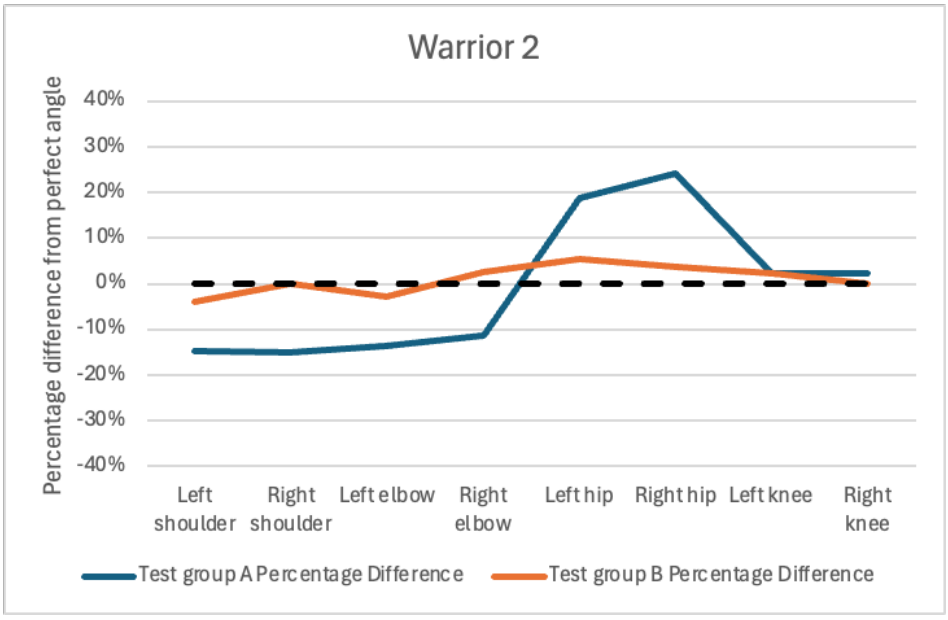


Figure 8.3: Average percentage difference in each angle when performing the Warrior 2 pose, comparing Group A using video instructions to Group B using our application. The expected percentage difference in all angles is 0%. Data is derived from Figure A.5 from the appendix.

Figure 8.4 shows that the average test subjects from Group A and Group B scored similarly on all angles performing the Downward-Facing Dog and Four-Limbed Staff pose.

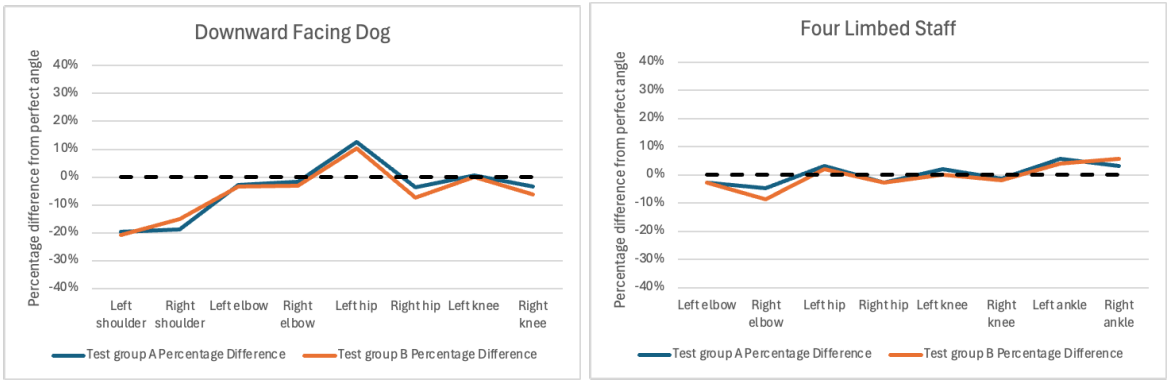


Figure 8.4: Average percentage difference in each angle when performing Downward-Facing Dog and Four-Limbed Staff poses, comparing Group A using video instructions to Group B using our application. The expected percentage difference in all angles is 0%. Data is derived from Figure A.7 from the appendix.

8.2.2 Bootstrapping The Test Results

Because we ran our test on a relatively small sample group for both the videos and our app, we need a way to gain confidence in the results before we can use them to test our hypotheses. A common way to do this is by using the technique called bootstrapping. This practice involves four simple steps [5]:

1. **Make a bootstrapped dataset** - Do this by randomly selecting n number of test subjects from the test group, where n is equal to the number of participants (test subjects can be selected multiple times for the same bootstrapped dataset).

2. **Calculate the needed value for the new bootstrapped dataset** - We will calculate the bootstrapped dataset's mean value.

3. **Keep track of the calculated value** - This entails keeping track of the calculated mean frequency, which we will use after bootstrapping.

4. **Repeat steps 1-3 any number of times** - We can create any number of bootstrapped datasets. To make the next test more precise, a larger number is better than a low number.

We did the following four steps for the scores of all four poses and the average score. We created a new Python script that utilized the package called Scipy[1]. This script helped us generate 9999 bootstrapped datasets to calculate their mean value and keep track of this value for all cases. Then we utilized the Python package called Matplotlib[2] to help create histograms of the bootstrapped distributions mean value (see Figure 8.5).

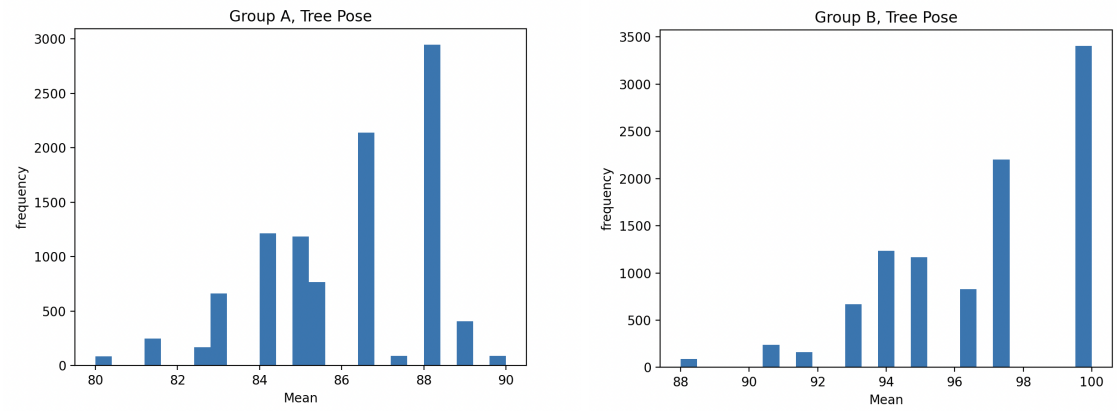


Figure 8.5: Bootstrapped distribution of the mean score of Tree Pose for both test groups (The other histograms can be found in the Appendix A.5)

[1]Scipy Homepage used to bootstrap the test data
[2]Matplotlib Homepage used to create a histogram for each bootstrapped distribution of the mean value

These histograms can then be used to calculate a 95% confidence interval for the mean score of all poses and the average score[5]. This is done by removing the outermost 2.5% of the mean frequency for each case, leaving an interval where we can conclude with 95% confidence that the mean value would fall between in the case we continued testing on more subjects. For the Tree pose, this interval for group A would be between 81.5 and 89.0, and for group B, the interval is 90.5 to 100. Using these confidence levels, we can plot them into our Average scores for both test groups diagrams (see Figure 8.6).

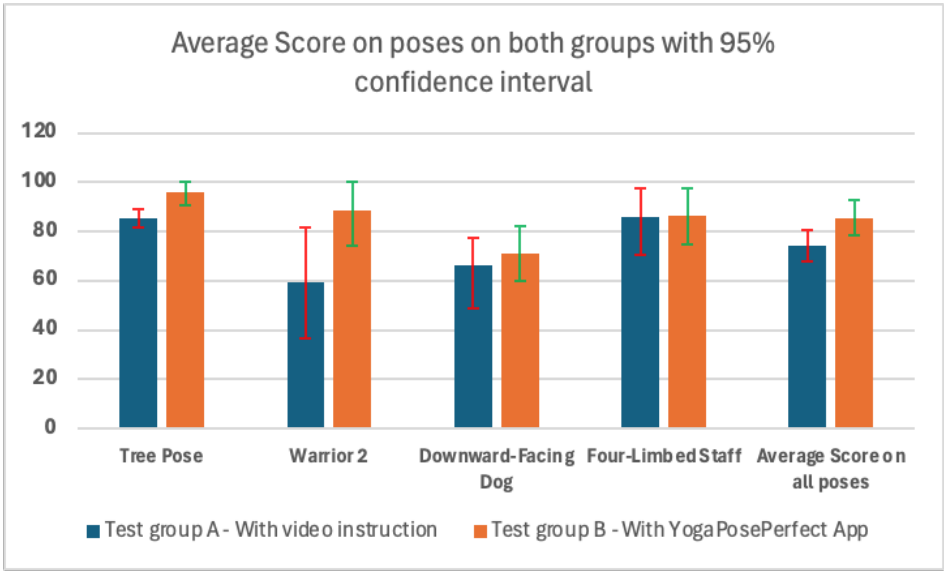


Figure 8.6: Compared average scores on poses from both test groups with 95% confidence intervals (Table for this Bar Chart can be found in Appendix A.13 and Appendix A.14)

In the bar chart shown in Figure 8.6, we can conclude that the 95% confidence interval for Tree Pose is higher for group B than for group A. Therefore, we can conclude with confidence that the app performs better than the video. As group B's confidence interval overlaps group A's confidence interval, we can neither disprove nor prove that the app performance is better than the video for the poses and Warrior II, Downward-Facing Dog, and Four-Limbed Staff.

8.2.3 Evaluating Test Surveys

Each test subject was asked a series of questions to gather more qualitative data on what aspects of our application worked well and what could be improved.

Positive Feedback on the Application

The score analyzer feature received the most positive feedback from Group B, as they felt it clearly indicated whether they were performing each yoga pose correctly or incorrectly. In contrast, Group A participants who performed yoga with video instructions mentioned that they were unsure if they were executing the poses correctly. Additionally, participants from Group A noted the lack of guidance on improving their poses, which was not mentioned by Group B.
Several participants from Group A mentioned that they found it difficult to follow the video instructions because they were very fast-paced in English. This appeared to be an advantage for our application, which provides purely visual guidance rather than spoken instructions.
Participants from Group A also found it challenging to keep up with the video instructions due to their pace. In contrast, Group B appreciated that they could perform each step of the yoga movements at their preferred speed.
Lastly, multiple participants from Group B found it satisfying to perform the poses, due to the visual feedback from the score evaluation that goes up whenever the participant approached the correct pose.

Negative Feedback on the Application

While participants from Group B, who performed yoga using our application, agreed that it worked excellently for the Tree Pose and Warrior II poses, most also agreed that it was less effective for the Downward-Facing Dog and Four-Limbed Staff pose. This issue arises because participants must turn sideways to the camera while the poses require looking down. This orientation makes it difficult to verify whether you are nearing a 100% pose score, as the score decreases as soon as you look at the camera. Specifically, your arms also block your view of the camera for the Downward-Facing Dog. During the test, multiple Group B members requested to see video instructions for each pose before performing them with our application. Since this test was designed purely to benchmark our application against the common method of performing yoga today, we did not offer video instructions to Group B. However, we acknowledge that combining both methods might yield even better results. Other feedback we received from a few participants in Group B was that they missed having a voice to guide them in perfecting each pose. Sometimes, the visual indication of a pose overlay did not communicate clearly enough what needed to be changed to achieve a higher score. While testing each participant from Group B, our application occasionally had difficulty identifying the participant's pose, indicating that our model could be improved further. Lastly, some participants found seeing all the information on the screen difficult. This is challenging because participants usually have to stand 2-3 meters from the screen to properly fit into the frame for poses such as the Tree Pose and Four-Limbed Staff, where the body is fully stretched.

8.3 Conclusion

The data gathered from our test phase indicates that our application functions as intended, as the average score was higher for participants using our application to perform yoga. It is important to remember that we tested only 12 participants due to limited time. For this reason, we cannot be certain that our application is definitively a better alternative than performing yoga with video instructions. Still, by calculating the 95% confidence interval for the mean score and all poses, we can with some confidence, conclude that the app is better for the poses Tree Pose. We are quite satisfied with the results, considering this is only the first iteration of testing. Additionally, we received valuable feedback on aspects of our application that can be significantly improved, which we will address in Chapter 9.

Chapter 9

Future Improvements

After analyzing the results of the initial testing phase and gathering feedback from our test subjects, we have identified several areas for improvement that could enhance the user experience of our application. This chapter will outline the improvements we propose for the application.

9.1 Enhancing Pose Analysis

We observed issues with the pose analysis algorithm during testing with our subjects. Throughout the test duration, we identified several ways subjects could manipulate the application to achieve artificially high scores by performing actions not intended for certain poses. For instance, while attempting Four-Limbed Staff, laying completely flat on the floor could result in a score higher than 70 since most angles resemble those expected in the pose when the body is straight (See Figure 9.1). However, this contradicts the intended behavior of our application, as Four-Limbed Staff requires only the feet and hands to touch the floor. To address this issue of incorrect pose analysis, we propose introducing a weight variable for each recorded angle of every pose. This weight would indicate the relative importance of each angle to the overall score, ensuring that crucial angles carry more weight in the scoring process. For example, the angle of the elbow would have a high weight for Four-Limbed Staff, thus contributing significantly to the final score.

Figure 9.1: A score of 70 for laying down instead of Four-Limbed Staff

Another observation regarding the pose analysis algorithm was that a seemingly better-executed pose, where the entire body remains within the outline of the perfect pose, would produce the same score as a pose where most of the body was outside the perfect pose outline. This discrepancy was particularly noticeable in the tree pose, where subjects leaning to one side still received a high score without fully being within the outline. To address this issue, we propose introducing additional angles for each pose. For instance, in the case of the Tree pose, an additional angle could be added between the landmarks for the shoulder, hip, and foot to prevent the user from leaning to the side. Incorporating more angles like this for each pose mitigates this problem and provides a more detailed score, as it includes values from a broader range of angles.

9.2 Refining the Pose Classification Model

In an earlier chapter, we discussed the model's tendency to predict Downward Facing Dog as a default, even when the frame was completely black. We believe a small adjustment could be made to the model's training by introducing a neutral class as the first. This neutral class would force our model to default to it when it fails to recognize any of the other four poses. During the post-analysis of the test group using videos, we found that our pose classification model lacked precision in some cases. For example, our model couldn't classify the correct pose in the specific case when analyzing some of the subjects' poses. In some cases, we could make our model predict the correct pose by moving our subject around the frame.

In Figure 9.2, it is shown that a subject doing the Warrior 2 pose in the center of the image, our model's classification of the pose is correct. Still, by moving the subject to the top right of the frame, the model classifies the pose as a Downward-Facing Dog, and by moving the subject to the bottom right of the frame, the prediction is instead of Four-Limbed staff. This is a small mistake as our subjects will usually be in the center of the frame, but nonetheless, it shows that our model, in some cases, is inaccurate. To improve our model, we believe that extending our training dataset with each image duplicated with the pose in different places of the frame and running the training again could improve the model's accuracy in these cases. Another solution we think would greatly help our model's precision would be extending our training dataset to include more images of each pose. When training any Machine Learning model, the rule is always 'data is king'[1], and running with a larger dataset of high quality would greatly improve the model.

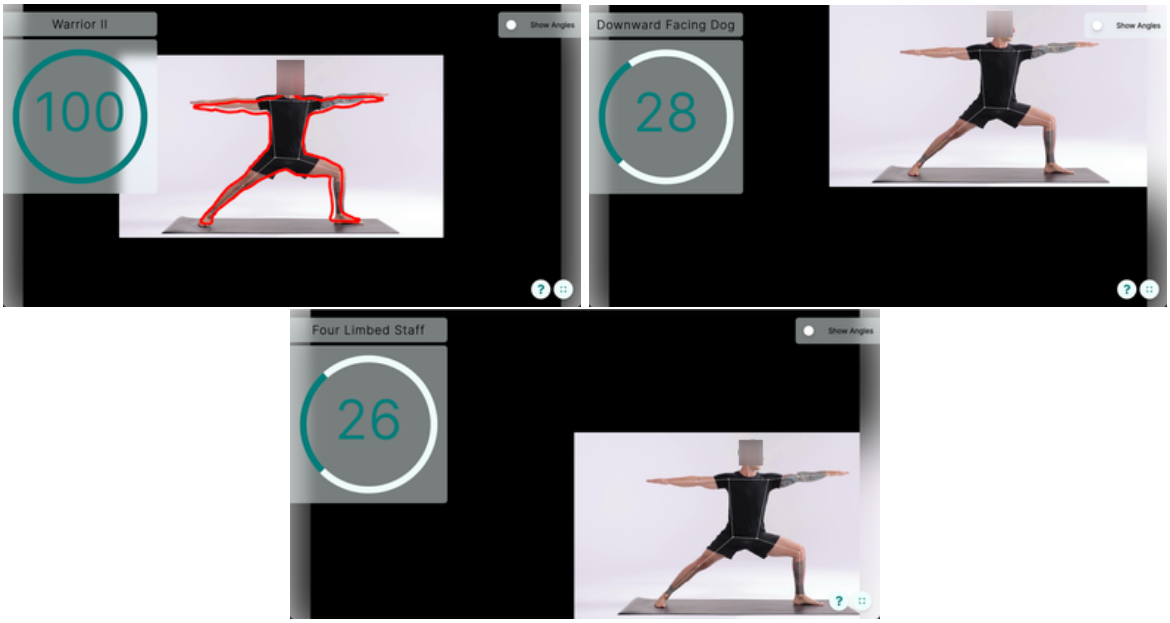


Figure 9.2: Examples of the same pose being classified differently based on frame position

9.3 Optimizing Pose Guidance

After observing our test subjects using the video guides to perform the chosen poses, we identified several key aspects that worked well in the videos. The most significant aspect was the guidance provided by the instructors throughout the video, which was corroborated by feedback from our test users. The step-by-step instructions offered by the yoga coaches effectively guided users into each pose. In contrast, the test group using our app had to interpret the end-state pose without explicit guidance. This structured approach to pose execution inspired us to consider adding a step-by-step guide as a new feature to our application. Our concept involves breaking down each pose into multiple steps

[1]Data is King: The Role of Data Capture and Integrity in Embracing AI (Anghel, Alexandra)

to facilitate user guidance. Figure 9.3 illustrates three distinct steps for the Tree Pose. We propose integrating these steps into the app by performing pose analysis on the initial step and allowing users to proceed to the next step once they have achieved a certain threshold in the previous step. However, implementing this idea would necessitate an extension of our pose classification model to incorporate the initial step. This presents a significant challenge as the steps may be similar, posing a risk of confusion for the model, thus requiring a substantial training dataset. Additional enhancements to the application could include extending the pose analysis to score the initial steps. We believe this extension would be a pivotal feature, enhancing the user experience and shifting the focus of the application from analysis to learning. Furthermore, incorporating voice instructions for the step-by-step poses could give users essential guidance points for each pose.

Figure 9.3: Tree pose split into three individual steps

9.4 Conclusion

After taking a step back and reviewing our test phase to brainstorm ideas for improving our application, we identified several potential improvements. Some improvements are essential for fulfilling the program's initial objectives, while others would enhance the overall user experience. It's important to note that the impact of these improvements will vary; some may have a more significant effect than others, and conversely, some may require more development effort.

Chapter 10

Conclusion

In this thesis, we explored integrating a web application using real-time pose detection and classification to enhance yoga practice at home. Our findings substantiate the initial hypothesis that such technology can improve pose alignment and accuracy compared to traditional video instructions, offering real-time feedback that aids in correctly executing yoga poses.

Firstly, we conducted several tests to determine the most precise pose detection model for landmark detection in yoga poses. We identified BlazePose Heavy as the most accurate model and successfully implemented it into our application, ensuring a seamless user interface without any performance issues.

Secondly, we focused on training a pose classification model capable of accurately predicting four selected yoga poses across various environments. Through multiple iterations and learning processes, we achieved a classification model that performed efficiently in diverse settings.

Our third objective was to develop a web application that integrates pose detection and classification to aid yoga practitioners in achieving better pose alignment. From our test data, we observed an average 15% increase in pose accuracy when using our application compared to instructional videos. To confirm or refute our hypothesis confidently, we applied bootstrapping to calculate the 95% confidence intervals. After doing so, we can confidently state that our application improves alignment for the Tree Pose. While there are indications that Warrior II and Downward-Facing Dog pose also showed better results when using our application, we cannot conclusively say that these poses will achieve better alignment with our tool.

Overall, our application improves alignment compared to instructional videos for some poses, while others are on par. Therefore, we can prove our hypothesis that our application generally provides better alignment results.

As our web application shows considerable potential for enhancing home yoga practice, there are opportunities for improvement based on insights gathered from observations during the test phase and participant feedback. Continued iteration and refinement will be crucial to optimize the user experience further and address remaining challenges.

References

[1] Feras A. Batarseh Ajay Kulkarni Deri Chong. *Foundations of data imbalance and solutions for a data democracy*. https://t.ly/7ciTa. Accessed: 2024-04-29. 2020.

[2] Jia Deng Alejandro Newell Kaiyu Yang. *Stacked Hourglass Networks for Human Pose Estimation*. https://arxiv.org/abs/1603.06937. 2016.

[3] Mykhaylo Andriluka et al. "2D Human Pose Estimation: New Benchmark and State of the Art Analysis". In: *2014 IEEE Conference on Computer Vision and Pattern Recognition*. 2014, pp. 3686–3693. DOI: 10.1109/CVPR.2014.471.

[4] Ishvar V Basavaraddi. "Yoga: Its origin, history and development". In: *Ministry of External Affairs of Government of India* (2015). eprint: http://www.redtwigyoga.com/uploads/1/2/1/9/12195443/yoga__its_origin_history_and_development.pdf.

[5] Albert Y. Kim Chester Ismay. *Statistical Inference via Data Science*. https://moderndive.com/8-confidence-intervals.html. 2024.

[6] Roy Derks. *React Projects: Build advanced cross-platform projects with React and React Native to become a professional developer*. 2022.

[7] Grandel Dsouza, Deepak Maurya, and Anoop Patel. "Smart gym trainer using Human pose estimation". In: *2020 IEEE International Conference for Innovation in Technology (INOCON)*. 2020, pp. 1–4. DOI: 10.1109/INOCON50539.2020.9298212.

[8] Jocelyn Granger. "Neuromuscular therapy manual". In: (2020). URL: https://books.google.dk/books?hl=da&lr=&id=inTtDwAAQBAJ&oi=fnd&pg=PP1&dq=Neuromuscular+therapy+manual&ots=ij6pbX2Hel&sig=TbPYH9buxJydGbjRyQywRa7Dgw8&redir_esc=y#v=onepage&q=Neuromuscular%20therapy%20manual&f=false.

[9] Andrej Karpathy. *CS231n Convolutional Neural Networks for Visual Recognition: Loss Functions*. https://cs231n.github.io/neural-networks-3/#loss. Accessed: 2024-04-08. 2020.

[10] Dev Kotak et al. "Yoga Pose Classification using Angle Heuristic Approach". In: *2022 4th International Conference on Inventive Research in Computing Applications (ICIRCA)*. 2022, pp. 1709–1714. DOI: 10.1109/ICIRCA54612.2022.9985555.

[11] Ajay L et al. "AI Tool as a Fitness Trainer Using Human Pose Estimation". In: *2023 International Conference on Network, Multimedia and Information Technology (NMITCON)*. 2023, pp. 1–12. DOI: 10.1109/NMITCON58196.2023.10276320.

[12] V. Lavanya et al. "AI Enhanced Yoga Pose Detection And Alignment Using Deep Learning". In: *2024 3rd International Conference for Innovation in Technology (INOCON)*. 2024, pp. 1–6. DOI: 10.1109/INOCON60754.2024.10512108.

[13] Tewodros Legesse Munea et al. "The Progress of Human Pose Estimation: A Survey and Taxonomy of Models Applied in 2D Human Pose Estimation". In: *IEEE Access* 8 (2020), pp. 133330–133348. DOI: 10.1109/ACCESS.2020.3010248.

[14] Keiron O'Shea and Ryan Nash. *An Introduction to Convolutional Neural Networks*. https://arxiv.org/abs/1511.08458. 2015.

[15] Takato Otsuzuki et al. "Meta-learning of Pooling Layers for Character Recognition". eng. In: *Document Analysis and Recognition – ICDAR 2021*. Lecture Notes in Computer Science. Cham: Springer International Publishing, pp. 188–203. ISBN: 9783030863333.

[16] Jonathan Passmore et al. “The impact of COVID-19 on coaching practice: results from a global coach survey”. In: *Coaching: An International Journal of Theory, Research and Practice* 16.2 (2023), pp. 173–189. URL: https://www.tandfonline.com/doi/full/10.1080/17521882.2022.2161923.

[17] R. Priyadharsini, T. Sree Sharmila, and V. Rajendran. “An Efficient Edge Detection Technique Using Filtering and Morphological Operations for Underwater Acoustic Images”. In: *Proceedings of the Second International Conference on Information and Communication Technology for Competitive Strategies*. ICTCS ’16. Udaipur, India: Association for Computing Machinery, 2016. ISBN: 9781450339629. DOI: 10.1145/2905055.2905168. URL: https://doi-org.ep.ituproxy.kb.dk/10.1145/2905055.2905168.

[18] Arun Kumar Rajendran and Sibi Chakkaravarthy Sethuraman. “A Survey on Yogic Posture Recognition”. In: *IEEE Access* 11 (2023), pp. 11183–11223. DOI: 10.1109/ACCESS.2023.3240769.

[19] Safyzan Salim et al. “Learning Rate Optimization for Enhanced Hand Gesture Recognition using Google Teachable Machine”. In: *2023 IEEE 13th International Conference on Control System, Computing and Engineering (ICCSCE)*. 2023, pp. 332–337. DOI: 10.1109/ICCSCE58721.2023.10237148.

[20] Gregg Twietmeyer. “What is Kinesiology? Historical and Philosophical Insights”. In: *Quest* 64.1 (2012), pp. 4–23. DOI: 10.1080/00336297.2012.653268. eprint: https://doi.org/10.1080/00336297.2012.653268.

[21] Karthik Raveendran - Tyler Zhu - Fan Zhang - Matthias Grundmann Valentin Bazarevsky Ivan Grishchenko. *BlazePose: On-device Real-time Body Pose tracking*. https://arxiv.org/abs/2006.10204. 2020.

[22] Prathmesh Waghmode et al. “Real-time Yoga Pose Classification and Correction: YogaAI”. In: *2023 4th International Conference for Emerging Technology (INCET)*. 2023, pp. 1–6. DOI: 10.1109/INCET57972.2023.10170599.

[23] Ce Zheng et al. *Deep Learning-Based Human Pose Estimation: A Survey*. 2023. arXiv: 2012.13392 [cs.CV].

Appendix A

Appendix

A.1 Correct angles for yoga poses

Landmarks	Downward-Facing Dog	Four-Limbed Staff	Tree pose	Warrior 2
Left knee	165°	165°	150°	145°
Right knee	170°	170°	40°	105°
Left hip	80°	165°	165°	120°
Right hip	90°	175°	105°	90°
Left elbow	150°	125°	85°	170°
Right elbow	140°	125°	70°	155°
Left shoulder	165°	30°	30°	85°
Right shoulder	165°	30°	30°	100°
Left ankle	95°	105°	110°	115°
Right ankle	95°	105°	110°	115°

A.2 MPII Human Pose Dataset - Snippet of dataset

MPII Human Pose Dataset [1]

Figure A.1: Snippet of images that occur in the dataset with people performing yoga

[1] [3]

A.3 Test Phase

A group of people is divided evenly into two groups:
Group A: Practices Yoga using a video demonstration by an instructor. Group B: Practices Yoga using our application

The test subject must perform four poses with 2 minutes for each pose.

Group A test instructions
Show the test subject an instruction video of the pose they should perform.
The person tries to perform it using the application, which is recorded for later analysis. This is done for each pose.

Group B test instructions
Give the test subject a video of an instructor explaining how to do a pose. The person tries to do the pose while being recorded for later analysis.

Survey after test:
1. How confident are you that you performed the poses correctly (1-5), and why?
2. Group A: Would you prefer to follow a video instead? Why? Group B: Would you prefer an app instead? Why?
3. Do you feel you received enough information to perform the poses correctly? If not, what was missing?
4. Did you find it difficult to perform the poses, and why?

Analysis
Each video is analyzed using our pose detection and angle calculator to verify how close each subject was to the actual pose.
In the end, we can compare group A to group B and verify whether the people receiving a better score used the app.

A.4 Test results

Test Subjects	Tree Pose	Warrior 2	Downward-Facing Dog	Four-Limbed Staff	Average Score on all poses
Test Subject #1	88	65	80	95	82
Test Subject #2	83	73	43	85	71
Test Subject #3	90	90	75	63	79
Test Subject #4	85	58	55	95	73
Test Subject #5	88	43	73	100	76
Test Subject #6	80	30	73	78	65
Average on all test subjects	**85**	**60**	**66**	**86**	**74**

Figure A.2: Average pose score from each test subject in each pose from test group A - With video instructions

Test Subjects	Tree Pose	Warrior 2	Downward-Facing Dog	Four-Limbed Staff	Average Score on all poses
Test Subject #1	95	80	55	80	78
Test Subject #2	100	100	65	70	84
Test Subject #3	88	68	70	90	79
Test Subject #4	100	100	70	100	93
Test Subject #5	100	90	75	83	87
Test Subject #6	93	95	90	95	93
Average on all test subjects	**96**	**89**	**71**	**86**	**85**

Figure A.3: Average pose score from each test subject in each pose from test group B - With YogaPosePerfect app

Tree Pose	Average Angle - Test group A - With video instruction	Average angle - Test group B - With YogaPosePerfect App	Test group A Percentage Difference	Test group B Percentage Difference
Left shoulder	38	31	128	103
Right shoulder	29	28	97	94
Left elbow	87	85	102	100
Right elbow	73	68	104	98
Left hip	165	166	100	101
Right hip	106	100	101	95
Left knee	151	152	101	101
Right knee	54	45	135	113

Figure A.4: Tree Pose: Average angle on each landmark on all poses on test groups A and B. Table also shows the difference to the correct angle (correct is 100)

Warrior 2	Average Angle - Test group A - With video instruction	Average angle - Test group B - With YogaPosePerfect App	Test group A Percentage Difference	Test group B Percentage Difference
Left shoulder	73	82	85	96
Right shoulder	85	100	85	100
Left elbow	147	165	86	97
Right elbow	138	159	89	103
Left hip	143	127	119	106
Right hip	112	93	124	104
Left knee	148	148	102	102
Right knee	108	105	102	100

Figure A.5: Warrior 2 Pose: Average angle on each landmark on all poses on test groups A and B. Table also shows the difference to the correct angle (correct is 100)

Downward Facing Dog	Average Angle - Test group A - With video instruction	Average angle - Test group B - With YogaPosePerfect App	Test group A Percentage Difference	Test group B Percentage Difference
Left shoulder	133	131	80	79
Right shoulder	134	140	81	85
Left elbow	146	145	97	97
Right elbow	138	136	98	97
Left hip	90	88	113	110
Right hip	87	83	96	93
Left knee	166	165	101	100
Right knee	164	159	97	94

Figure A.6: Downward Facing Dog Pose: Average angle on each landmark on all poses on test groups A and B. Table also shows the difference to the correct angle (correct is 100)

Four Limbed Staff	Average Angle - Test group A - With video instruction	Average angle - Test group B - With YogaPosePerfect App	Test group A Percentage Difference	Test group B Percentage Difference
Left elbow	122	122	97	97
Right elbow	119	114	95	91
Left hip	170	168	103	102
Right hip	170	170	97	97
Left knee	168	165	102	100
Right knee	168	167	99	98
Left ankle	111	109	106	104
Right ankle	108	111	103	106

Figure A.7: Four Limbed Staff Pose: Average angle on each landmark on all poses on test groups A and B. Table also shows the difference to the correct angle (correct is 100)

A.5 Bootstrapped Histograms and Table with Confidence intervals

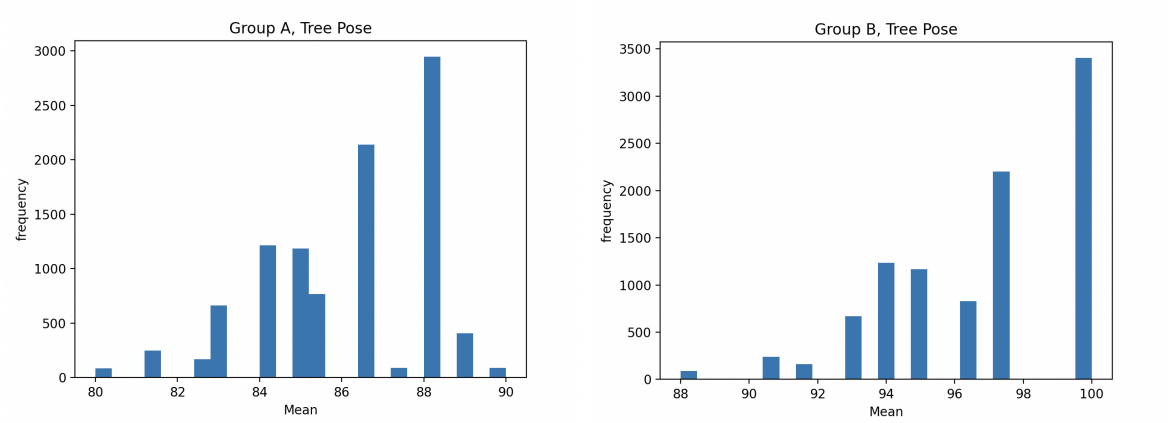


Figure A.8: Bootstrapped histograms for groups A and B on Tree Pose, with 9999 bootstrapped datasets

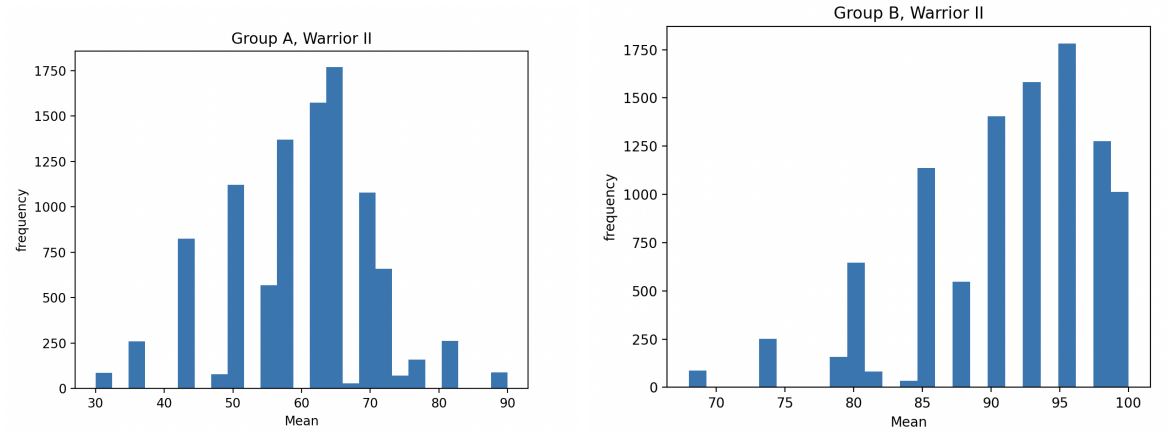


Figure A.9: Bootstrapped histograms for groups A and B on Warrior II, with 9999 bootstrapped datasets

Test Subjects A Tree Pose Warrior 2 Downward-Facing Dog Four-Limbed Staff Average

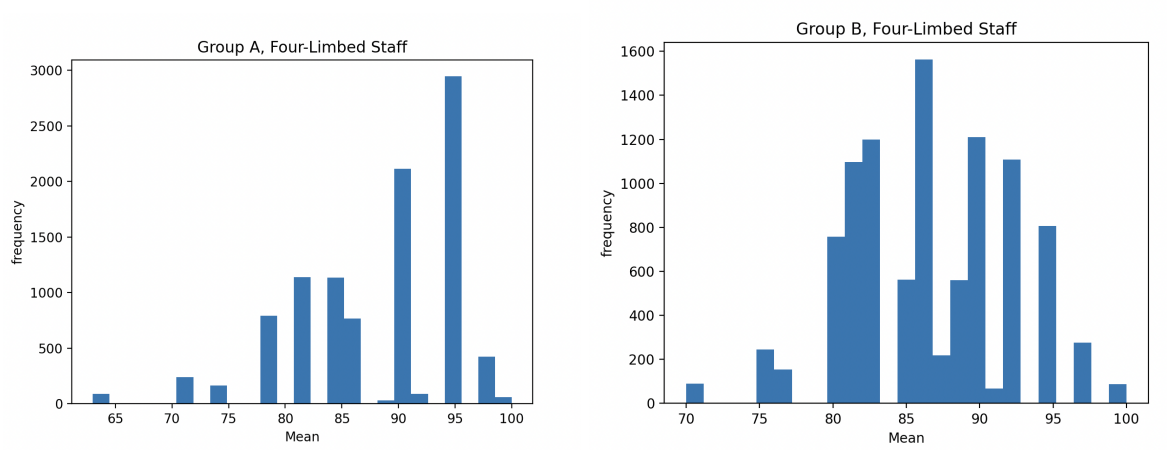


Figure A.10: Bootstrapped histograms for groups A and B on Four-Limped Staff, with 9999 bootstrapped datasets

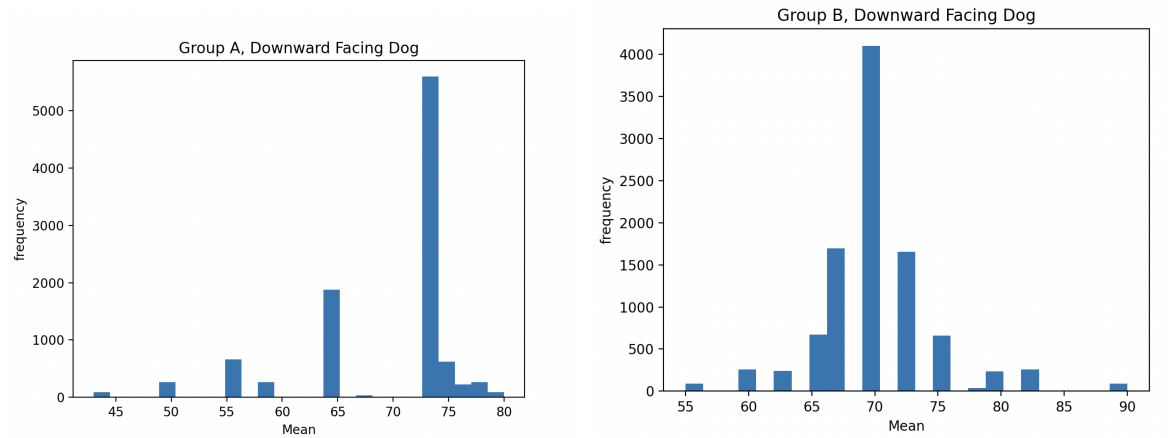


Figure A.11: Bootstrapped histograms for groups A and B on Downward Facing Dog, with 9999 bootstrapped datasets

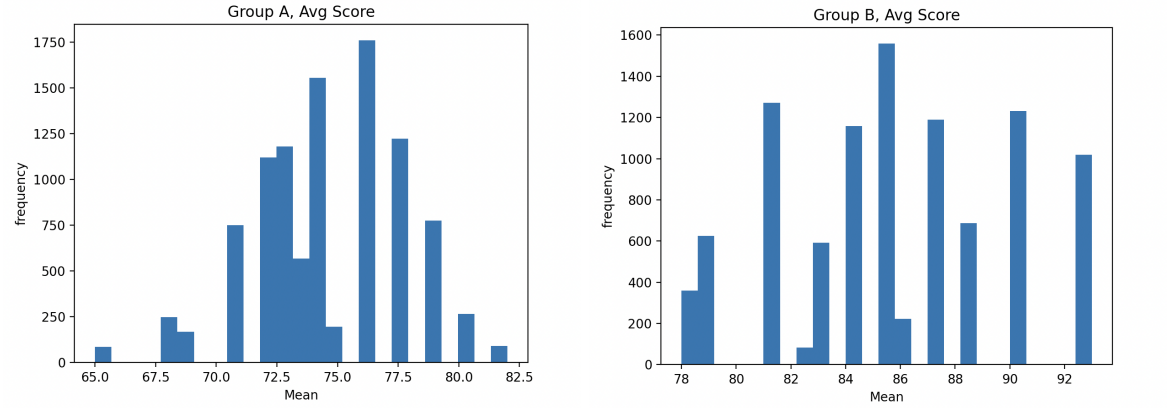


Figure A.12: Bootstrapped histograms for groups A and B on their average score, with 9999 bootstrapped datasets

Test Subjects A	Tree Pose	Warrior 2	Downward-Facing Dog	Four-Limbed Staff	Average Score on all poses
Test Subject #1	88	65	80	95	82
Test Subject #2	83	73	43	85	71
Test Subject #3	90	90	75	63	79
Test Subject #4	85	58	55	95	73
Test Subject #5	88	43	73	100	76
Test Subject #6	80	30	73	78	65
Average on all test subjects	**85**	**60**	**66**	**86**	**74**
95% Confidence Lower	***81,50***	***36,50***	***49,00***	***70,50***	***68,00***
95% Confidence Higher	***89,00***	***81,50***	***77,50***	***97,50***	***80,50***

Figure A.13: Table for all scores and average scores for all poses in test group A, including confidence intervals

Test Subjects B	Tree Pose	Warrior 2	Downward-Facing Dog	Four-Limbed Staff	Average Score on all poses
Test Subject #1	95	80	55	80	78
Test Subject #2	100	100	65	70	84
Test Subject #3	88	68	70	90	79
Test Subject #4	100	100	70	100	93
Test Subject #5	100	90	75	83	87
Test Subject #6	93	95	90	95	93
Average on all test subjects	**96**	**89**	**71**	**86**	**85**
Confidence Lower	***90,50***	***74,00***	***60,00***	***75,00***	***78,50***
Confidence Higher	***100,00***	***100,00***	***82,00***	***97,50***	***93,00***

Figure A.14: Table for all scores and average scores for all poses in test group B, including confidence intervals

A.6 Code & demo for the application

- The code for the application can be found in the following public GitHub repository.
- The application is hosted on Vercel.