HAW HAMBURG

BACHELOR THESIS

Squat Assessment using Pose Estimation

submitted on June 14 2024

Carolina Figueiredo

First Examiner: Prof. Dr. Stephan Pareigis
Second Examiner: Prof. Tim Tiedemann

HOCHSCHULE FÜR ANGEWANDTE WISSENSCHAFTEN HAMBURG
Department Informatik
Berliner Tor 7
20099 Hamburg

Abstract

This Thesis offers a new perspective on how to assess squat performance without the help of personal trainers. Media Pipe model allowed me to extract keypoints for various squat positions from pre-recorded videos. The data was then saved into a file with respective categorizations. Each file and a different parameter being valued. Squat data was categorized based on side (front, left, right), width (wide, narrow, neutral), range (low, medium, parallel, high) and stance (up, down). Thresholds were defined so that with the help of the Pose Estimation model keypoints were automatically categorized and saved.

The CSV files were then loaded into data frames to be used to train each category's model. With all the models trained, combining them in real-time feedback demonstration where all the categories were showcased and then saved as well into a csv file so that the set could be analyze.

Ultimately, it was decided that this last file was also a good base to try and classify a squat as correct or incorrect depending on the users preference. the study aims to provide a customizable framework for squat classification, accommodating various target ranges and widths, thereby offering flexibility based on different research parameters and individual goals. The resulting system increases the accuracy of squat performance feedback, highlighting the potential for automated, real-time assessment in fitness applications.

Keywords: Pose estimation, squat assessment, MediaPipe, scikit-learn, OpenCV, keypoints extraction

Contents

List of Figures

1 Introduction

1.1 Background and Motivation

1.1.1 Squat Assessment

Dynamic Squats are common core exercises used by athletes to enhance performance in sport. Being the reason, to grow muscle or to just have better mobility and enhancing overall physical performance. As a multi-joint exercise, it develops the largest and most powerful muscles of the body. The squatting movement is used to many everyday tasks, being considered one of the best exercises for improving quality of life. (Schoenfeld & J, 2010)

Benefits of a good squat performance are not exclusive to fitness enthusiast. They are often recommended in clinical environment due to they're designation as closed kinetic chain exercises. (ESCAMILLA et al., 2001) During Knee rehabilitation and in a controlled environment these exercises help to slowly recover the knee mobility.

When unsupervised, people can have difficulties assessing their own technique. This can lead to injuries, due to poor technique or inappropriate exercise prescription. Muscle and ligaments sprains, ruptured inter vertebral discs. Despite variations on how squat technique is instructed and executed to address specific performance goals, nearly all squat variations comprise a standard, basic, and fundamental blueprint that underlies the bio-mechanical technique that will support progressive physical attribute improvements and decrease the risk of training induced injuries.(Myer et al., 2014)

1.1.2 Pose Estimation

Pose estimation is a fundamental task in computer vision and artificial intelligence (AI) that involves detecting and tracking the position and orientation of human body parts in images or videos. (Odemakinde, 2024) Human pose estimation aims to locate the human body parts and build human body representation. This includes detecting, associating, and tracking semantic key points. It has been increasingly used in a wide range of application, including motion analysis, augmented reality, and virtual reality. (Zheng et al., 2023)

Detection of the human body keypoints is a challenging task. There are a lot of variables that can change the body's appearance, such as clothing, arbitrary occlusion, occlusions due to the viewing angle, and background contexts. To challenge the real-world variations the Pose estimation must be robust.

Current methods for squat analysis include wearable sensors and computer vision-based approaches. Wearable sensors, while accurate, can be intrusive and expensive. Computer vision-based methods, particularly those leveraging pose estimation models, offer a non-invasive and scalable solution. However, these methods still face challenges, such as handling occlusions and complex movements.

1.1.3 Motivation

As a dedicated fitness enthusiast, challenges with proper form are always in mind. Despite researching and seeking for gym trainers help, whenever alone one finds herself questioning her own skills. This uncertainty not only hindered my progress but also increased the risk of injury.

From a perspective of someone who often finds herself dreading to do any kind of squat exercises, mainly because of it's risks and the pain that sometimes comes from performing it incorrectly, it would only be logical that I would choose this exercise to start my research. Other exercises, such as arms or back exercises, don't cause as much confusion.

During my fitness journey, I noticed that many other individuals faced similar difficulties in assessing and correcting their squat techniques. This widespread issue highlighted the need for an accessible and accurate tool that could provide real-time feedback on squat form.

With this project I aim create a solution for people to get a accurate feedback, by combining advanced computational methods with practical fitness insights. This research not only addresses my personal interest but also contributes to the broader field of fitness technology by filling a significant gap in the current offerings.

1.2 Problem Statement

As previously mentioned, the accurate assessment of a squat performance is essential to improve training effectiveness and to minimize risks of injury. Many methods to evaluate squat rely on manual measurements, or subjective observations, being prone to inconsistency.

Despite the availability of various pose estimation models, there is a need for an efficient and accurate system that can assess squat form in real-time across different platforms. Existing

methods either lack the required accuracy or are too computationally intensive for real-time applications. This paper addresses the challenge by leveraging MediaPipe's pose estimation capabilities and machine learning to develop a robust squat assessment tool.

1.3 Objectives

The main goal for this project is to achieve an efficient program that would help analyzing in real-time the performance of a series of squats. Using MediaPipe as Human Pose Estimation model, track the body position for each frame and using researched parameters to label each position, depending on the perspective or the squat type. This thresholds can be used to automate the classification of the squats, achieving a wider data base than if it was manually, being less prone to subjective observation if pose estimation model detection does not fail. Afterwards trying to achieve a machine learning model that can use this keypoints and labels as data to detect the squat type using scikit-learn to train different models. Combining these models will allow me to develop a real-time feedback tool using OpenCV, providing immediate insights into the squat performance.

1.4 Contributions

This project presents significant contributions to the field of sports science and computer vision. Including having an automated squat classification system, threshold-based classification, real-time feedback and visualization, bridging research gaps.

1.4.1 Threshold-based Classification

As researches conclude that different type of squats are beneficial for different muscles or can be injury inducing depending on the persons motion, it is possible that some of the results from other projects can be too binding. The implementation of thresholds is flexible throughout different studies. With the increasingly search for the perfect squat for each individual this parameters give this tool adaptability characteristics. The implementation of specific thresholds for knee angles and hip-knee distances enabled accurate classification of squat positions and ranges, distinguishing between various squat depths and widths effectively. Introducing target ranges and widths allowed for customizable criteria to classify squats as good or bad, catering to varying research parameters and user preferences.

1.4.2 Automated Squat Classification System

Automated classification was achieved by combining various methods, it started by using MediaPipe model to extract keypoints, using Thresholds to automatically save keypoints to a table with label, scikit-learn served as a model training and it used the previous table as data, instead of trying to train a model based on footage.

1.4.3 Model Training

Various machine learning pipelines were evaluated to determine the best-performing models, ensuring high accuracy in squat classification.

1.4.4 Visualization

The implementation of real-time feedback was then performed to provide the user immediate insights, not only being possible to visualize each model individual and check their probabilities, but also check results for all models with the side limitations. With the Pose estimation model it can also show the skeleton perspective, the knee-angle and distances from important joints.

1.4.5 Contribution to Research

Researching for this paper I wasn't able to get a well rounded conclusion to each parameter, so for the squat assessment, this work provides a standardized approach that can be adopted and further refined by future studies.

2 Related work

2.1 Review of Existing Methods

2.1.1 Human Pose Estimation

- **Convolutional Pose Machines**(Wei et al., 2016): Pose Machines provide a sequential prediction framework for learning rich implicit spatial models.
- **Deep Learning-based Human Pose Estimation: A Survey**(Zheng et al., 2023): This survey article provides a comprehensive review of recent deep learning-based solutions for both 2D and 3D pose estimation via a systematic analysis and comparison of these solutions based on their input data and inference procedures
- **Building a Poor Body Posture Detection & Alert System Using MediaPipe Body Tracking**(Odemakinde, 2022): This blog post approaches Posture of people when they're seated and gives real-time feedback, using MediaPipe as a Pose Estimation model

2.1.2 Squat Assessment

- **Analysis of the Load on the Knee Joint and Vertebral Column with Changes in Squatting Depth and Weight Load**(Hartmann et al., 2013): The aim of this literature review is to assess whether squats with less knee flexion (half/quarter squats) are safer on the musculoskeletal system than deep squats
- **Biomechanics of Squatting Exercises**(Straub & Powers, 2024): Detailed analysis of muscle activation and joint loading during different squat depths, providing a basis for defining proper squat form and range.

2.2 Comparison

Some of the studies helped me find some guidance for the mainline of my work, what I think it lacked in all the studies and projects I saw was the lack of robustness and depending of one method alone. The main difference of my aproach is that I looked at all these have done and brought some of each, so that I would be able to achieve a more efficient way of Assessing Squat Exercises using Pose Estimation, and leveraging with other approaches.

3 State-of-The-Art

3.1 Human Pose Estimation

Pose estimation is a rapidly advancing field within computer vision, leveraging machine learning and deep learning techniques to predict the positions of key body joints from images or videos. This section reviews the current state-of-the-art methodologies and tools in pose estimation, highlighting their advancements, applications, and limitations.

3.1.1 Deep Learning-Based Approaches

- Convolutional Neural Networks
  - **OpenPose**: Developed by Carnegie Mellon University, OpenPose is a real-time multi-person system that jointly detects human body, hand, facial, and foot keypoints. It utilizes a bottom-up approach, first detecting all keypoints and then grouping them into individual poses.
  - **Hourglass Networks**: Introduced by Newell et al., hourglass networks are designed to capture information at various scales and repeatedly process and consolidate features, making them effective for keypoint localization.
  - **MediaPipe by Google**: This solution provides high-fidelity body pose tracking, detecting 33 landmarks on the human body. It uses a combination of machine learning models to predict keypoints from RGB images and is optimized for performance on various platforms, including mobile devices.

4 Methodology

4.1 Data Collection

Initial data was collected from an Iphone(`format='.mov'`), after some consideration and a lot of time processing this data, it was created a script (`record_video.ipynb`) that automatically recorded from the computers camera(`format='.avi'`), scaled and saved the videos to intended folder.

Videos were recorded in different perspectives in a local gym. However, for each video only one perspective was performed and saved to corresponding folder(`'_front'`,`'_left'`, `'_right'`, `'_new_videos'`).

For this paper I wasn't able to have other people helping me to record their squats, so there was only one subject recorded. More than 20 squats were recorded per squat type and side.

Due to some problems accessing files with certain names, both data and folders were all saved with '_' concatenated in the beginning.

To ensure that I can access all videos pathes to use them all at once or just by groups I saved all video paths in line with their folder and name to an excel (''). To update this list it is possible to run `save_data.ipynb`

Figure 4.1: Raw Image Front

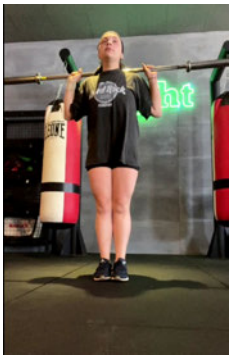

Figure 4.2: Raw Image Front Bar

4.2 Data Preprocessing

Videos recorded on phone's resolution were too big for this purpose, so before starting using it is necessary to adjust. This first videos were resized to 20%, all videos when used are always re-scaled to 75%, except if the user needs lower resolution

4.2.1 Media Pipe

MediaPipe is an open-source framework designed for creating and deploying machine-learning pipelines. It processes multimedia data such as images and video in real time. "Calculators", pre-built components that combined develops computer vision pipelines. It is considered versatile as it allows pipelines to be deployed across various platforms (Web, smartphones and small embedded systems). Its cross-platform capabilities help developers create immersive and responsive applications for any device. (Vina, 2024)

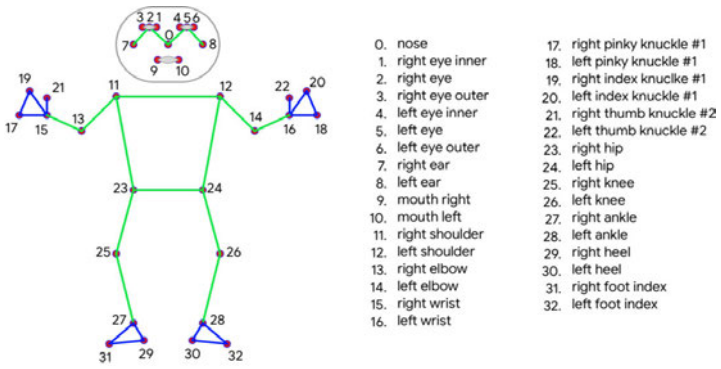

Figure 4.3: MediaPipe Pose Estimation Keypoints Model

4.2.2 Extraction and storage of keypoints

After processing all videos and updating the video library, keypoints are extracted and stored in CSV files ('_coords...' files). To initialize this:

```
landmarks = ['class']

for val in range(1, 33+1):
    landmarks += ['x{}'.format(val), 'y{}'.format(val), 'z{}'.format(val), 'v{}'.
    format(val)]
```

- `landmarks` is initialized as a list with the column header `'class'`.
- It iteratively adds columns for each keypoint coordinate (`'x1'`, `'y1'`, `'z1'`, `'v1'`, ..., `'x33'`, `'y33'`, `'z33'`, `'v33'`).

For the MediaPipe detection it was designed a class `detector()` where all it's features could be used. For example, to detect and draw the keypoints model, detector.Pose() was used.

The `poseDetector` class contains several methods for detecting and processing human poses in video frames. Below is a list of these important functions along with brief descriptions of their purposes.

- `def getPose(self, img, draw=True):`
  - Processes the input image to detect poses and optionally draws landmarks on the image.
- `def getPosition(self, img, draw=True):`
  - Extracts the positions of keypoints from the detected poses and optionally draws them on the image.
- `def getAngle(self, img, p1, p2, p3, draw=True):`
  - Calculates the angle between three keypoints using the cosine rule as follows:

$$\theta = \cos^{-1}\left(\frac{\mathbf{a}\cdot\mathbf{b}}{\|\mathbf{a}\|\|\mathbf{b}\|}\right)$$

- `def getDistanceKeypoints(self, img, p1, p2, x_flag=True, draw=True):`
  - Calculates the x or y distance between two keypoints.

4.3 Feature Extraction

Next step was to find a way of using some of the detector functions to automate the labeling of the squats. To achieve this, we identified specific parameters within the Pose Estimation Model that could detect the necessary keypoints.

4.3.1 Thresholds Determination

The first thing that came to my mind when thinking of a squat movement is the knee angle. To evaluate the knee angle, two vectors were needed, hip to knee and knee to ankle, with that we could access our `poseDetector` and use the `getAngle()` being 180° the most up position and 0° the lowest position.(ESCAMILLA et al., 2001; Hartmann et al., 2013; Schoenfeld & J, 2010)

Determining this range was a challenge but at the end I went with a more general approach to this. The thresholds for the knee angle were:

- **Low**: $140^\circ \leq \theta < 180^\circ$
- **Medium**: $90^\circ \leq \theta < 140^\circ$
- **Parallel**: $\theta = 70^\circ$
- **High (deep squat)**: $\theta < 70^\circ$

Another thing that can be difficult to determine when alone is the width in which the squat is being made. For this I compared the distances between the shoulders and ankles:

- **Narrow**: Shoulder distance > ankle distance.
- **Wide**: Shoulder distance < ankle distance.
- **Neutral**: Between narrow and wide.

For determining the up and down positions during squats, different thresholds are used based on the side(front, left, right):

1. **Right/Left (θ)**:
    - **Up**: $\theta \geq 140^\circ$
    - **Down**: $\theta < 140^\circ$
2. **Front (Coordinates of Hip and Knee)**:
    - Calculate the coordinates of hips and knees for both left and right sides.
    - Compute the average coordinates for hips (avg_hip_y) and knees (avg_knee_y).
    - Determine the maximum coordinate difference between hip and knee (max_hip_knee).
        - **Down**: avg_hip_y $\geq$ avg_knee_y
        - otherwise
            - **Down**: avg_hip_knee $\leq$ (max_hip_knee$\frac{\text{max_hip_knee}}{4}$)
            - **Up**: every other value

4.4 Data Annotation and Labeling

Combining the previous Methods we can achieve what we aimed for. That means using `poseDetector` and using the thresholds for data annotation and labeling.

All data originated in this project is saved between this folders:

- _csv
- _models
- _results

in a sub-folder for the persons name that is given at the begging of each script.

4.4.1 Side

First I began by using my pre-recorded data, organized in group folders for the `Side` class. Selecting the data saved in `video_library.csv` per folder, captured each video, detected every pose keypoint, as the folder identifies which side I was doing, it would automatically save in the `coords_side_folder.csv` the array with both keypoints and label. After doing this for all sides I appended the files in `coords_side_new.csv`.

I found that doing this for each label would be more productive, because sometimes I would see some poor MediaPipe estimation detection and I would have to do it all over again.

To avoid those poor estimations from MediaPipe I increased `min_detection_confidence` and `min_tracking_confidence` from 0.5 to 0.8 the results were actually satisfactory in the sense that the model was apparently more accurate.

4.4.2 Up/Down

`coords_counter_new.csv` This class determines if the body is in an Up or Down position, using this we will be able to count squats. Similarly to the previous class, it saves the keypoints and the label to CSV file. However this annotation method will already use the trained model of the previous class.

I started this method using only the angle to determine the stance. The results were not so good, because even though everything seemed to have been accurate, the angles for the squat in a frontal position were not really. Giving the fact that I had already trained the model to determine the side, I made the most of it.

Now, knowing this I could more accurately label this data, leveraging the model results with the thresholds previously mentioned.

For this method I used all the data available at once.

4.4.3 Range

`coords_range_right.csv` `coords_range_left.csv` `coords_range_new.csv` To establish the range of a squat I infer that in a frontal position one would not be able to reliably do it, because there is no exact form to know the angle the person is achieving in this position and I did not know how to label it using distance or coordinates for keypoints.

For this scenario, in the same way that I used the side model to label the UP/Down Class, I can confidently use it.

Merging the previous knowledge of the side, I can at first select only videos from `_right` or `_left` folders, Making use of the `video_library.csv` and finally use the previously deduced thresholds to annotate.

4.4.4 Width

`coords_width_new.csv` In line with the other methods I used MediaPipe to detect keypoints and used thresholds for the labelling.

However, in this case, the perspective from right or left cannot always detect the other side keypoints, so obtaining the distance between 2 opposite keypoints would not be accurate, for that reason this only label squat performed from the front.

4.4.5 Good or Bad Squat

This classification is obtained after all the models are trained. Combing `'_side00.pkl'`, `'_counter001.pkl'`, `'_range001.pkl'`, `'_width001.pkl'`.

In this method there is a little personalizing, user can choose the range or the width that they want to achieve, for range people can choose between deep or normal for it to work, reason being that low range is not a good range to achieve. As for the width is the same as the classes given, wide, neutral, narrow.

Starting by detecting the side, there is two options after this, or we assess the range or the width, to label squat as good or bad.

If the body is on the right or left side it only detects the angle and range. For this to evaluate the squat correctly it needs to check the lowest point achieved when the person goes down. When this requirement is met, the range and angle values do not change until it returns to up position. For squat to be label as a `good squat`, if `range target` is `normal`, the current range should be either `parallel` or `medium`, if `range target` is `deep` then current range should be `high`

For the front position it will detect distance of ankle, distance of shoulder and width. It detects in which width it was made. To achieve a good squat `width target` must be the same `width` of the current squat

When stance changes from a down position to an up position it changes all the parameters to start again. Both will always increase repetitions, however counter only increases when a good squat is achieved.

Before it changes the parameters will be all saved to:

```
squat_list = np.array(['class','side','angle', 'range', 'target_range', 'dist_ankle'
    , 'dist_shoulder', 'width', 'target_width','counter','repetitions'])
```

and then append it to a CSV file.

4.5 Scikit-Learn

Scikit-learn is a comprehensive machine learning library in Python, renowned for its user-friendly interface and extensive functionality. The sklearn library contains a lot of efficient tools for machine learning and statistical modeling including classification, regression, clustering and dimensionality reduction.

All the models were trained the same way except for `squat_classification.pkl`

I utilized a CSV file for data, partitioning it into training, validation, and test sets, employed scikit-learn and experimented with various pipelines to optimize performance. After evaluating different pipelines, I selected and trained the best-performing one.

In the squat classification model the only difference was that to properly evaluate the model I had to encode the string value labels, such as `'class','side', 'range', 'target_range', 'width', 'target_width'`, due to problems with using some tools.

4.6 OpenCV

OpenCV, Open Source Computer Vision Library, is a great tool for image processing and performing computer vision tasks. (Gupta, 2022) It is a widely used open-source software library an it offers a broad range of tools and algorithms for image and video processing, including object detection, tracking, motion estimation, and image enhancement.

It proved to be really useful for the real-time feedback. Being able to use this throughout the whole project was really amusing. When I started I was actually doing the labeling manually, and using this method I could actually do it efficiently. Applying the cv2.waitkey() function I was able to at first label the data manually, but afterwards I used it to reduce errors.

Sometimes when there was no body in the frame or only part of it, it would have trouble with the automate system, so I was still supervising and pressing

- `'s'`: when it was clear to start
- `'e'`: when there was an anomaly in the MediaPipe detection
- When I used the video library
  - `'c'`: to skip to the next video
- `'q'`: to quit

With this tool I was also able to draw the keypoints, the angles and the distances, and finally, the most important part was to have a real-time feedback with results from all the modules.

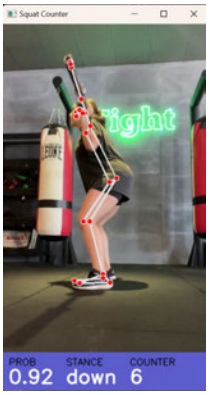

Figure 4.4: Model counter to the right

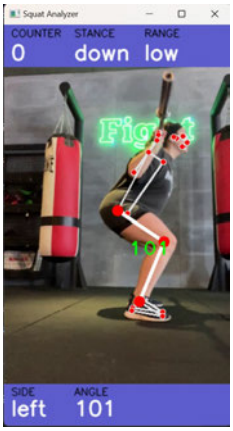

Figure 4.5: Model for Range

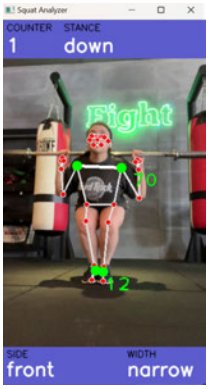

Figure 4.6: Overall Classification -> narrow width

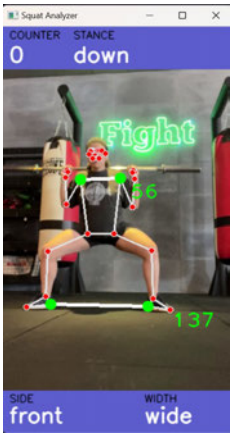

Figure 4.7: Overall Classification -> wide width

5 Experiment and Results

This chapter will describe how the models were trained and the results achieved. As the main focus of this Thesis was not to dive deep into Machine Learning I will show how I trained the models very briefly and share the results.

5.1 Training, Testing and Validation

I decided to use scikit-learn as it provides a simple and consistent interface to various machine learning algorithms. It is well integrated in the Python-ecosystem and allows for easy data manipulation, visualization and integration.

Using `'pandas'` library I loaded the data into a DataFrame from a CSV file. then separated the class that I wanted to predict, from the rest of the data.

Resorting to `train_test_split()` function I divided my data into three groups, training(size = 0.6), testing(size = 0.2) and validation(0.2).

I proceeded to test various pipelines:

```
pipelines = {
    'lr':pipeline_lr,
    'rc':pipeline_rc,
    'rf':pipeline_rf,
    'gb':pipeline_gb,
    'svc':pipeline_svc,
    'knn':pipeline_knn
}
pipe_dict = {'lr': 'Logistic Regression',
             'rc': 'Ridge Classifier',
             'rf':'Random Forest Classifier',
             'gb': 'Gradient Boosting Classifier',
             'svc':'Support Vector Machine',
             'knn':'k-Nearest Neighbors'}
```

Afterwards, the best pipeline was fitted, validated, tested.

5.2 Results

5.2.1 _counter001.pkl

- **best pipeline**: randomforestclassifier
- **Random Forest Classifier**: 0.9894406551908584
- **Validation Accuracy**: 0.9904659387974781
- **Testing Accuracy**: 0.9896970628940489
- **Precision**: 0.9896952208073676
- **Recall**: 0.9896970628940489
- **F1 Score**: 0.989695536062244

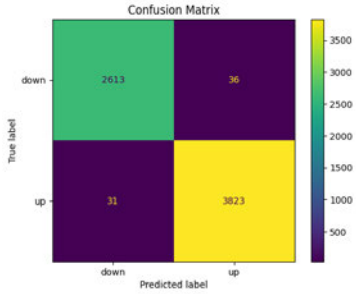

Figure 5.1: Confusion Matrix Counter

5.2.2 _range001.pkl

- **best pipeline**: randomforestclassifier
- **Random Forest Classifier**: 1.0
- **Validation Accuracy**: 0.9841909485430874
- **Testing Accuracy**: 0.98171109733416
- **Precision**: 0.9785110430258206
- **Recall**: 0.98171109733416
- **F1 Score**: 0.979833891672007
- **ROC AUC Score**: 0.99932727724582

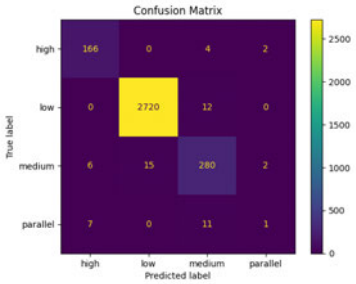

Figure 5.2: Confusion Matrix Range

5.2.3 _side001.pkl

- **best pipeline**: randomforestclassifier
- **Random Forest Classifier**: 0.983674375894249
- **Validation Accuracy**: 0.999532273152479
- **Testing Accuracy**: 1.0
- **Precision**: 1.0
- **Recall**: 1.0
- **F1 Score**: 1.0
- **ROC AUC Score**: 1.0

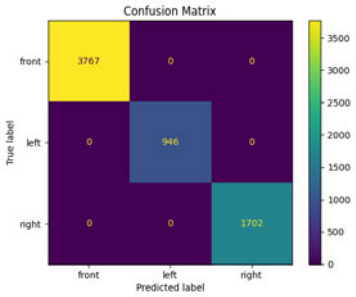

Figure 5.3: Confusion Matrix Side

5.2.4 _width001.pkl

- **best pipeline**: randomforestclassifier
- **Random Forest Classifier**: 0.9923636363636363

- **Validation Accuracy**: 0.9942732478865558
- **Testing Accuracy**: 0.9961821652577039
- **Precision**: 0.9961983699720006
- **Recall**: 0.9961821652577039
- **F1 Score**: 0.9961865617696007
- **ROC AUC Score**: 0.9997152635444112

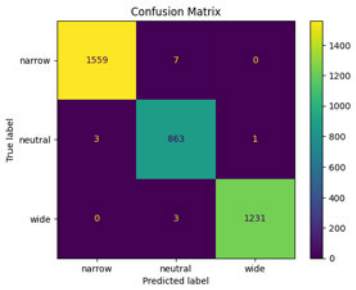

Figure 5.4: Confusion Matrix Width

5.2.5 _squat_classification_final.pkl

- **best pipeline**: k-nearestneighbors
- **k-Nearest Neighbors**: 0.9884241971620613
- **Validation Accuracy**: 0.9942196531791907
- **Testing Accuracy**: 0.976878612716763

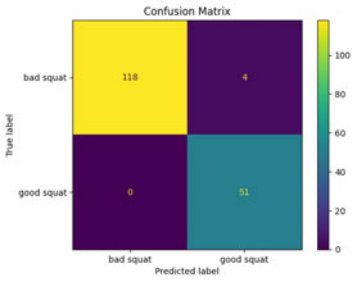

Figure 5.5: Confusion Matrix Squat Classification

5.2.6 Real-time feedback

As for the script to use OpenCV and real-time detection for immediate feedback, it actually performed really well, the Pose Estimation model was capable of detecting the keypoints and feeding the keypoints obtained really good results, most squats that I assessed were having great results. Not in the sense of performing a good or bad squat, but in actually classifying as such.

Here are some examples of the results:

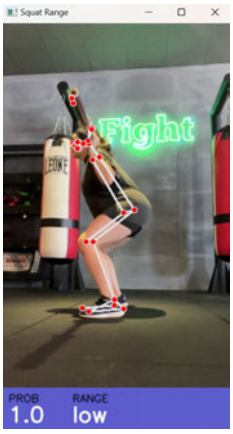

Figure 5.6: Model for Range Detection to the Right -> low

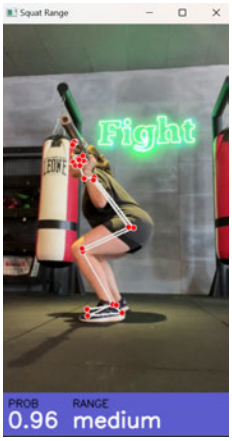

Figure 5.7: Model for Range Detection to the Right -> medium

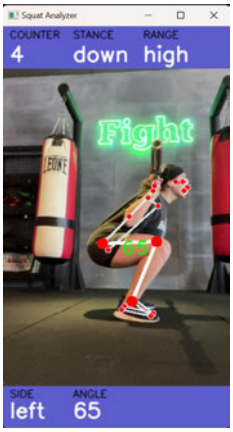

Figure 5.8: Overall Classification -> high range

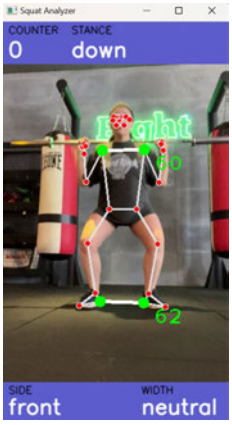

Figure 5.9: Overall Classification -> neutral width

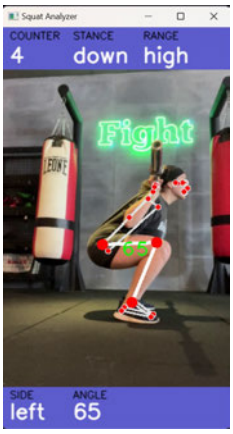

Figure 5.10: Overall Classification -> high range

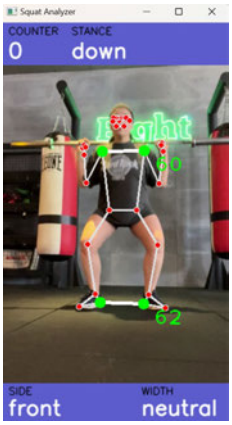

Figure 5.11: Overall Classification -> neutral width

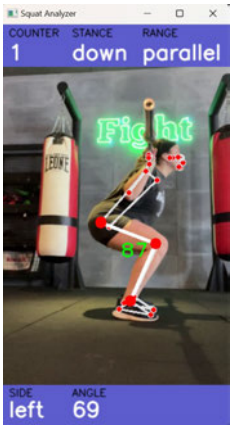

Figure 5.12: Overall Classification -> parallel range

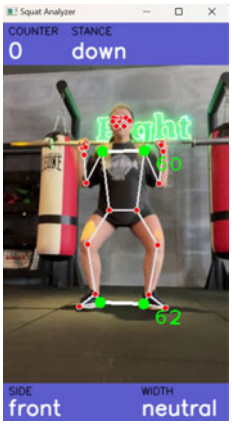

Figure 5.13: Overall Classification -> neutral width

6 Discussion

Gathered all the data and all the result is time now for the discussion. In this chapter I will talk about the results and how they could be more accurate or robust.

6.1 Media Pipe as Pose Estimation

Using MediaPipe for these project was a good decision. Even thought sometimes the model would identify keypoints that weren't visible or whenever the body gets out the picture the model would draw some random skeleton, for the purposes of this work and due to it's adaptability across platforms it actually served is purpose and did achieve great results.

This model was used only to get data and to check the results, however together with all the other methods it turned out to have the most contribution for the robustness of this work.

6.2 Squat Assessing

When I first thought of this work I was not expecting to be challenged with the wide range of research on this topic and the amount of contradictory views on squat performance. On a more basic level it is maybe easy to say which squat looks good or bad, however it as a lot more ro it. I was not able to use this model for the injury prevention part, but I have achieved good results approaching it in a more flexible way.

6.3 Data Problems

I think that for this work it also wouldn't change that much, because it is supposed to acquire some data from each person to then train it and get results for each individual. But the fact is that my model only as data from one person the coordinates are mainly the same for each position and so with the amount of data collected in the form of keypoints it got almost perfect score in every model. Maybe it would be different with other people's data. In my

opinion it would not change much because of the automation in data annotation it really increases the data available for training compared to a manual annotation.

7 Conclusion

This thesis presents a novel approach to assessing squat performance using pose estimation techniques and machine learning. It proves that is possible to use a Pose Estimation Model and Machine Learning tools to detect different squat types on the basis of video data.

There is a discrepancy between labels in the range confusion Matrix, that's because when performing a squat a lot more data is saved to the low range due to is proximity to up position. Parallel is also really rare to achieve, so is less frequent.

From the real-time feedback we can also conclude that all the models were well trained and can detect back the squat type. The models have big accuracy levels, because of the amount of data given to train the model.

I can say with confidence that this work left curious to approach this in a more deeper level, due to is efficiency I wish to improve this further and into new fitness exercises. Because it's obvious that Pose Estimation can achieve great things in the fitness industry.

I hope this approach can also improve the next generation of Pose Estimation, Machine Learning and Fitness Assessment.

Bibliography

ESCAMILLA, R. F., FLEISIG, G. S., ZHENG, N., LANDER, J. E., BARRENTINE, S. W., ANDREWS, J. R., BERGEMANN, B. W., MOORMAN, & MOORMAN, C. T. I. (2001). Effects of technique variations on knee biomechanics during the squat and leg press. *Medicine & Science in Sports & Exercise*, *33*(9), 1552–1566. https://doi.org/10.1097/00005768-200109000-00020

Gupta, A. (2022). *Some amazing applications of opencv library*. https://www.analyticsvidhya.com/blog/2021/09/some-amazing-applications-of-opencv-library/#:~:text=OpenCV%20is%20a%20great%20tool,landmark%20detection%2C%20and%20much%20more. (accessed: 30.05.2024).

Hartmann, H., Wirth, K., & Klusemann, M. (2013). Analysis of the load on the knee joint and vertebral column with changes in squatting depth and weight load. *Sports Medicine*, *43*(10), 993–1008. https://doi.org/10.1007/s40279-013-0073-6

Myer, G. D., Kushner, A. M., Brent, J. L., Schoenfeld, B. J., Hugentobler, J., Lloyd, R. S., Vermeil, A., Chu, D. A., Harbin, J., & McGill, S. M. (2014). The back squat: A proposed assessment of functional deficits and technical factors that limit performance. *Strength Cond J.*, *36*(6), 4–27. https://doi.org/10.1519/SSC.0000000000000103

Odemakinde, E. (2022). *Building a poor body posture detection alert system using mediapipe body tracking*. https://learnopencv.com/building-a-body-posture-analysis-system-using-mediapipe/ (accessed: 10.06.2024).

Odemakinde, E. (2024). *Human pose estimation with deep learning – ultimate overview in 2024*. https://viso.ai/deep-learning/pose-estimation-ultimate-overview/ (accessed: 12.05.2024).

Schoenfeld & J, B. (2010). Squatting kinematics and kinetics and their application to exercise performance. *Journal of Strength and Conditioning Research*, *24*(12), 3497–3506. https://doi.org/10.1519/JSC.0b013e3181bac2d7

Straub, R. K., & Powers, C. M. (2024). A biomechanical review of the squat exercise: Implications for clinical practice. *Sports Medicine*, *19*(4), 490–501. https://doi.org/10.26603/001c.94600

Vina, A. (2024). *What is mediapipe? a guide for beginners*. https://blog.roboflow.com/what-is-mediapipe/ (accessed: 20.05.2024).

Wei, S.-E., Ramakrishna, V., Kanade, T., & Sheikh, Y. (2016). Convolutional pose machines. https://doi.org/10.48550/arXiv.1602.00134

Zheng, C., Wu, W., Chen, C., Yang, T., Zhu, S., Shen, J., Kehtarnavaz, N., & Shah, M. (2023). Deep learning-based human pose estimation: A survey. *ACM Comput. Surv.*, *56*(1). https://doi.org/10.1145/3603618

Declaration of Independence

I hereby certify that I have completed this Bachelor's thesis with the title

Squat Assessment using Pose Estimation

independently and only with the specified aids. All passages that I have taken verbatim from the literature or from other sources such as websites have been clearly marked as quotations with the source indicated.

Hamburg, 12 June 2024