

AN ARTIFICIAL INTELLIGENCE FITNESS COACH TO AID IN INJURY PREVENTION WHILE SQUATTING USING MACHINE LEARNING AND MEDIAPIPE

Matthew Olen ¹, Jonathan Sahagun ²

¹ Tarbut V' Torah, 5200 Bonita Canyon Dr, Irvine, CA 92603

² California State Polytechnic University, Pomona, CA, 91768

ABSTRACT

Athletes and casual gym-goers often risk injury when performing squats due to poor form. To address this problem, this project proposes an artificial intelligence-based system that uses computer vision and machine learning to monitor squat posture and provide real-time corrective feedback. The system leverages Mediapipe for pose estimation and K-Nearest Neighbors (KNN) for classification of squat form [8]. Major challenges included maintaining model accuracy, processing video data on a Raspberry Pi, and adjusting for different squat variations. Through experimental testing, the AI demonstrated 90–94% accuracy in identifying proper and improper squats, even when adapting to elevated squat styles. Compared to previous methodologies, this system improves by providing immediate feedback rather than post-set evaluations. Ultimately, this project presents a lightweight, affordable, and portable solution to improve exercise safety and performance, reducing the risk of serious injuries in athletic and fitness communities.

KEYWORDS

Computer Vision, Machine Learning, Squatting Injury Prevention, Artificial Intelligence, Pose Estimation, Fitness

1. INTRODUCTION

Many athletes have the potential to become future prodigies in the big leagues and major league sports. However, it is very common to hear a story about a rising star who injured themselves during an offseason practice or workout. Specifically, the lower back is at a very high risk of injury when doing exercises like squats [1]. In 2011, 13 football players were partaking in a workout that was centered around squats with heavy weights and low reps. After the workout, it was reported that those 13 players were diagnosed with rhabdomyolysis, which causes muscles to break down in parts of the body like the arms, legs, back, and shoulders [2][3]. Using the squat rack can often lead to a high risk of injury, especially if not used properly. Additionally, many relatively new athletes to a sport might not have much experience in the weight room and could put themselves at an even higher risk of misusing the squat rack. Also applying to experienced and new powerlifters, injury can often be caused by forgetting the correct form in the middle of a set. If the weight on the bar is too heavy, or someone tries to go for an extra rep at the end of their set, a common injury resulting in bad form could be a ruptured hernia disc [4]. Squatting often requires the squatter to keep their back and spine stiff, but when a person isn't able to do that, a disc in between the vertebrae of the spine might slip out and cause permanent injury to the person.

In order to prevent injuries while squatting, it is important that a person is reminded of the correct form in order to reduce the risk of body damage.

The first methodology evaluated AI versus physical therapists in squat form assessment [9]. Although the AI could identify good form reasonably well, it struggled to detect incorrect posture consistently. My project improved upon this by focusing on real-time feedback and adjusting for multiple types of incorrect form.

The second methodology focused only on standard squats performed at a fixed distance of 3 meters from the camera. This limits practical use in varied environments. My system allows for squat evaluation at any distance within the camera's field of view, making it more versatile for real-world applications in gyms or homes.

The third methodology classified squats into seven categories of form using Mediapipe and achieved high accuracy but only evaluated performance after the set was completed [10]. My system improves on this by providing real-time corrective feedback during the actual squat, reducing injury risk in the moment instead of after the exercise is finished.

A simple solution that could reduce the risk of injury while squatting would be an AI that could correct the form of someone during the set. Since most athletes and gym-goers feel pain normally after a workout, something that would be able to prevent serious pain would be a device or something that would tell a person how to lift properly, as most injuries are caused by bad form. Specifically, the AI would identify if the posture of the squat in the back is at a proper angle or not. If the AI detects the posture to be at an angle too small, it will speak out to the person to correct their form to prevent any injury that could come afterwards. A solution like this is simple and effective because it can be set up easily, as long as there is a place to put your device or if there is a power outlet nearby. The program is beneficial to run on with a raspberry pi which would be easy to set up as long as there is a power outlet somewhere nearby. Other methods to solve injuries while squatting are more human-based and are less likely to prevent an injury because someone would only be able to determine if they had good or bad form after recording, and wouldn't know whether or not he or she would need to correct their form at certain parts or not. The other methods also wouldn't be able to save someone if their form was bad in the middle of a set. If someone was at the bottom point of their squat, it would be bad for them if they tried to follow through the rep with bad form, as it could increase the risk of a bad injury.

For the first experiment, I tested the AI's accuracy when testing. In order to set up this experiment, I started off by having there be 500 images for the training data. 150 good form, 150 bad form, and 200 idle. Then, I had 30 testing images. I would then change the amount of good form, bad form, and idle data to 200 good form, 200 bad form, and 100 idle. I would play around with these numbers and see which combination would get me the highest accuracy results. (state results)

For the second experiment, I tested how well the AI would determine good form or bad form in elevated squats. I did this by first seeing how well the AI did with elevated squats with the old dataset, and then how well the AI did with a new dataset with elevated squat data in it. The results showed that the AI's did about the same in terms of accuracy, as the accuracy of the testing data didn't change much.

2. CHALLENGES

In order to build the project, a few challenges have been identified as follows.

2.1. Camera Setup and Power Challenges

A major component of this project is the camera used to capture the exercise. A major problem with this part of the project was setting it up. If I was at a local gym and wanted to set up the raspberry pi to run the program, the length of the wire might be a problem to get a good angle on myself. However, I could use a power bank to turn on the raspberry pi and set up the camera in a better spot with the power bank.

2.2. Data Collection and Mediapipe Angle Issues

Another major component of the project was the data used to train the AI model. When gathering the data, it is always hard to find the specific angle needed for mediapipe detection [11]. Mediapipe is important because it draws the lines over a body, which is what the AI trains off of. Finding the right angle to collect the data could be a problem because it would have to keep the mediapipe model on myself the entire time, and there could be a glitch or malfunction while doing so. However, I could use a camera stand that would hold the phone still, and also collect data from multiple different angles so the mediapipe would be able to capture the person without malfunctioning.

2.3. Optimizing FPS for Raspberry Pi Recording

A third major component of the project was the recording part that would capture somebody doing their squat. Since it is a large program and the raspberry pi isn't very good at processing large amounts of code, the frames per second rate was low in the output. In order to fix this, someone could find the right number of frames to use that would make the camera quality smooth. If someone set the frame rate too high, the raspberry pi wouldn't be able to process the video that well making the frame rate lower. However, if someone set it lower, while still being a reasonable amount of frames, it would make the quality and recording much smoother.

3. SOLUTION

In the program, it is formatted as one big file that can be opened with Visual Studio Code. Looking at the readme file first, it requires a prerecorded video that will be run through code that will split the video into individual frames, and then apply mediapipe over each frame. After, another section of code will gather all the data points from each image and organize them so they are suitable for training. Then, the data will be trained using KNN, which is machine learning to predict whether or not other data points qualify in the good or bad form sections. After the training part is done, the AI model will be tested using new data and once that's done, another chunk of code can be run to start recording. A camera will pop up and it will tell someone if they need to correct their form or not by speaking out loud. Once done, the recording will save as a video file and show the time periods for good form and bad form. The three most important components of the program however are; 1: The data collector and extractor, 2: The training and testing; and 3: The recorder and the playback. The data collecting and extracting is very important, as it is the first thing needed in order for the AI to train anything. Afterwards, the AI trains the data and tests the data to determine what would be determined as good or bad form. Lastly, another file is run to record the video, where the AI analyzes each frame and speaks back out if it detects good or bad form and writes text over the recording.

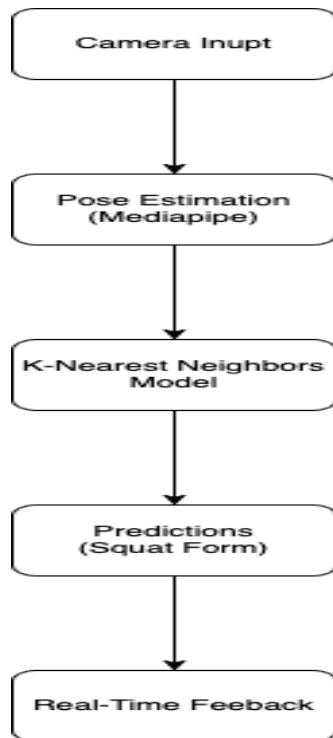


Figure 1. Overview of the solution

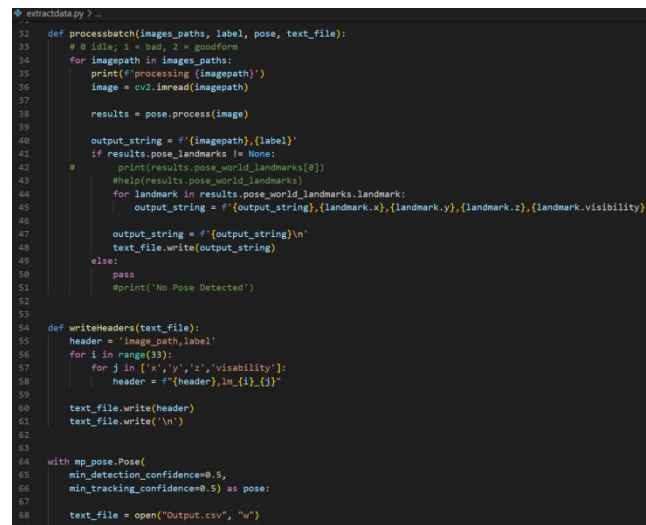
The first component of the project is the file that extracts the data from the video with the mediapipe overlay. It gathers the 33 points generated by mediapipe and gets the x, y, and z coordinates of each one in each frame from the video. The numbers are then later used for training and testing, and is how the AI knows whether a squat is considered good form or not.

```

Output
1 image_path_label,lm_x,lm_y,lm_z,lm_visibility,lm_x,lm_y,lm_z,lm_visibility,lm_x,lm_y,lm_z,lm_visibility,lm_x,lm_y,lm_z,lm_visibility,lm
2 dataset:bodyform_18.jpg,1,-0.3972268484153396,-0.36794863089462,-0.1381698378681054,0.99946429786206,-0.399383631516805,-0.40844481164401,-0.157231154-0.98461
3 dataset:bodyform_180.jpg,1,-0.461461426237792,-0.371808328816218119,-0.1235578209877014,0.999527931211379,-0.46122046616489,-0.4118642126176,-0.16216719860709,0
4 dataset:bodyform_181.jpg,1,-0.454864768314816,-0.3356442801889337,-0.1341666015845173,0.99956811721558,-0.4543938881264496,-0.3927143777681258,-0.1581513460511491
5 dataset:bodyform_182.jpg,1,-0.453761451381388,-0.357479181873212,-0.131662127231106,0.99989811608939,-0.451084877880476,-0.397121760331266,-0.1488181212217939,0
6 dataset:bodyform_183.jpg,1,-0.416718188263572,-0.374893245338052,-0.13786596859799184,0.999624408737915,-0.4128648814544467,-0.416788976876385,-0.153954245094846
7 dataset:bodyform_11.jpg,1,-0.418489311252184,-0.4018543864594289,-0.1372638737495422,0.99964272131348,-0.4165897665786743,-0.443366887813881,-0.134823245414148,0
8 dataset:bodyform_12.jpg,1,-0.45172165538804925,-0.3782379438118945,-0.164493821415129,0.9996118472486,-0.4514796421813465,-0.43094634396467,-0.186828485452349,0
9 dataset:bodyform_128.jpg,1,-0.4208836871281413,-0.38373637396481855,-0.15787894846171143,0.999657273251542,-0.4247791170739134,-0.42591680856864,-0.17768388468466
10 dataset:bodyform_121.jpg,1,-0.4276174368859587,-0.37813514474954877,-0.163181148771186,0.999884893424918,-0.4239958418773651,-0.4216137826427189,-0.1762868777846222
11 dataset:bodyform_122.jpg,1,-0.4286342839261885,-0.3518138610605646,-0.1524879606545844,0.99971832545382,-0.428161188717865,-0.3943676468721375,-0.16238684781378
12 dataset:bodyform_123.jpg,1,-0.444805131721866,-0.3589328914745321,-0.1845577893830292,0.9997186687608,-0.44187481718822,-0.3943621171767444,-0.1574918388955217,0
13 dataset:bodyform_124.jpg,1,-0.424092772289695,-0.3482517647628267,-0.18454168855178,0.99982281289796,-0.4276883368811169,-0.385294189458237,-0.18681846409488
14 dataset:bodyform_13.jpg,1,-0.43280761515445,-0.3458888984164226,-0.1718571646122143,0.989864379482145,-0.437611388082155,-0.37117737388887,-0.188916831574854718,0
15 dataset:bodyform_14.jpg,1,-0.436449747751413,-0.337842287489814,-0.1733656878578416,0.99914189111234,-0.441218448357454,-0.379858131211444,-0.15148381133811,0
16 dataset:bodyform_141.jpg,1,-0.427728513267421,-0.348887885979422,-0.165022857809318,0.999123773999,-0.4364581878727874,-0.391833132688824,-0.1878984441999715
17 dataset:bodyform_141.jpg,1,-0.4322493676324465,-0.3588978287553884,-0.1486637538692017,0.9992835521688,-0.4368735551831864,-0.3923514783824616,-0.168838618915961,0
18 dataset:bodyform_144.jpg,1,-0.45278211668895,-0.333568089829663,-0.1363858315776825,0.99934611784814,-0.438536485138928,-0.375737574966774,-0.1548216814888813
19 dataset:bodyform_145.jpg,1,-0.448909172891727,-0.3546821873873421,-0.13237889454591,-0.9993978988789,-0.47883812179239,-0.365588748282649,-0.1459317824521,0
20 dataset:bodyform_146.jpg,1,-0.46436742809271545,-0.321935888829179,-0.1383278835876465,0.99938416888882,-0.4762648793829785,-0.362988551338136,-0.146787795858969
21 dataset:bodyform_147.jpg,1,-0.44275918815484,-0.3271628181483584,-0.13174287984533995,0.9993588657289,-0.46588727623124895,-0.367881412485855,-0.14765789182258984
22 dataset:bodyform_148.jpg,1,-0.418813788994938,-0.3814382382396493,-0.138544082468797,0.9994541122435,-0.411284215867437,-0.42137467308696,-0.151557021878747
23 dataset:bodyform_15.jpg,1,-0.41955158791524895,-0.378427684318437,-0.14838763895116577,0.999459881518135,-0.4127776318725486,-0.4288033885414124,-0.164158582786222
24 dataset:bodyform_156.jpg,1,-0.4884233937984895,-0.3755848515124756,-0.1484242217548741,0.999493189838887,-0.482425219668458,-0.41925241413564,-0.155912721062425,0
25 dataset:bodyform_158.jpg,1,-0.4894847272916233,-0.37483189315418,-0.1519377671212887,0.9995378482192,-0.4884828779121116,-0.431618181875187,-0.13882621344844
26 dataset:bodyform_159.jpg,1,-0.486373788471895,-0.352872732448578,-0.228312347473882,0.99954768074155,-0.418122841586834,-0.3847893381187744,-0.245622727289737
27 dataset:bodyform_176.jpg,1,-0.489821418339862,-0.328696587811884,-0.2113818635977589,0.9995898957968,-0.4111177883148119,-0.408428718345728,-0.22870866468999
28 dataset:bodyform_171.jpg,1,-0.4895382131544167,-0.3888688687745086,-0.1313840892831664,0.9996418407929,-0.48818418577788754,-0.438144339795881,-0.1535338465153577,0
29 dataset:bodyform_172.jpg,1,-0.327193568187719,-0.458721876444892,-0.1581371414693146,0.99963933189389,-0.12116868278931284,-0.5825488162876187,-0.161371733880788
30 dataset:bodyform_188.jpg,1,-0.3153144588897,-0.4233805814712384,-0.149730548181356,0.99967488801233,-0.348918118032919,-0.476333229378746,-0.15428389384466,0
31 dataset:bodyform_196.jpg,1,-0.363862626526387,-0.42536862802182807,-0.114795621812416,0.999681883528394,-0.361855374145078,-0.4678167267252489,-0.14781644468493831
32 dataset:bodyform_193.jpg,1,-0.4861685112893833,-0.412939838832672,-0.122675284693315,0.99971776611183,-0.484428283718189,-0.45622289188975815,-0.143717488727831
33 dataset:bodyform_192.jpg,1,-0.43151847784874,-0.38873216665486,-0.128088438881,0.99972844111881,-0.42166314317959,-0.42031788457591,-0.148448888787354,0
34 dataset:bodyform_194.jpg,1,-0.4458745948848184,-0.3841437864381889,-0.238296147464332,0.999755824889977,-0.4585164886977272,-0.3433579438187888,-0.2183572784611985
35 dataset:bodyform_195.jpg,1,-0.444971111524649,-0.177646846484737,-0.128874818725495,0.98987168836442,-0.446431848478785,-0.3174888476182124,-0.15235574284821722
36 dataset:bodyform_213.jpg,1,-0.4818967612636459,-0.371278666262545,-0.16288868878824,0.99971778885311,-0.388822685889818,-0.4116848425975151,-0.182882427637939,0
37 dataset:bodyform_214.jpg,1,-0.4811947867814167,-0.3788613368189681,-0.173745111921411,0.99974884421227,-0.4884868188481784,-0.411258293581114,-0.1912175162696818
38 dataset:bodyform_215.jpg,1,-0.48244614954442,-0.3788613368189681,-0.173745111921411,0.99974884421227,-0.4884868188481784,-0.411258293581114,-0.1912175162696818

```

Figure 2. Screenshot of the numbers



```

extradatapy > ...
32 def processbatch(images_paths, label, pose, text_file):
33     # 0 idle, 1 = bad, 2 = goodform
34     for imagepath in images_paths:
35         print(f'processing {imagepath}')
36         image = cv2.imread(imagepath)
37
38         results = pose.process(image)
39
40         output_string = f'{imagepath},{label}'
41         if results.pose_landmarks != None:
42             print(results.pose_world_landmarks[0])
43             #help(results.pose_world_landmarks)
44             for landmark in results.pose_world_landmarks.landmark:
45                 output_string = f'{output_string},{landmark.x},{landmark.y},{landmark.z},{landmark.visibility}'
46
47             output_string = f'{output_string}\n'
48             text_file.write(output_string)
49         else:
50             pass
51             #print('No Pose Detected')
52
53
54 def writeHeaders(text_file):
55     header = 'image_path,label'
56     for i in range(33):
57         for j in ['x','y','z','visibility']:
58             header = f'{header},lm_{i}_{j}'
59
60     text_file.write(header)
61     text_file.write('\n')
62
63
64 with mp_pose.Pose(
65     min_detection_confidence=0.5,
66     min_tracking_confidence=0.5) as pose:
67
68     text_file = open("Output.csv", "w")

```

Figure 3. Screenshot of code 1

The code shown is responsible for extracting pose detection data from a set of images and writing that information into a CSV file [12]. It defines a processbatch function that processes a list of image paths along with their associated labels (where 0 = idle, 1 = bad form, and 2 = good form). For each image, the code reads the file using OpenCV, processes it through a Mediapipe pose estimation model, and checks if pose landmarks are detected [13]. If landmarks are found, it extracts the x, y, and z coordinates along with the visibility score of each of the 33 landmarks and appends them into a structured output string. This data string, along with the image path and its label, is then written to a CSV file. The writeHeaders function initializes the CSV file by writing the appropriate header columns for each landmark's x, y, z, and visibility values. Finally, the script sets up Mediapipe's pose detector with minimum confidence thresholds and opens a file named "Output.csv" for writing. This setup ensures that all the pose data needed for training a machine learning model is neatly collected and organized for later processing. The structure is essential for creating a dataset that maps visual information to posture classification labels for squatting form detection.

Another component of the program is the training file that trains the AI model off the extracted data. It is being trained through KNN or K-Nearest Neighbors. It first determines the trends between the data points and whether or not it classifies as good or bad form. With this data, it will use it to determine whether or not other points in the testing data would be classified as good or bad form, depending on how close they range from each class.

```

traindata.py X
traindata.py > ...
1 import pandas
2 from sklearn.model_selection import train_test_split
3 from sklearn.neighbors import KNeighborsClassifier
4 from sklearn.metrics import accuracy_score
5 import joblib
6 import datetime
7
8 df = pandas.read_csv("Output.csv")
9 #print(df.head)
10
11 X = df.drop(columns=["image_path", "label"])
12 y = df["label"]
13
14 X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)
15
16 k = 3
17 model = KNeighborsClassifier(n_neighbors=k)
18 model.fit(X_train, y_train)
19
20 y_pred = model.predict(X_test)
21 accuracy = accuracy_score(y_test, y_pred)
22
23 print(f"accuracy: {accuracy * 100: .2f}%")
24
25
26 counts = y.value_counts()
27 idle_count = counts[0]
28 bad_count = counts[1]
29 good_count = counts[2]
30
31 print(idle_count, bad_count, good_count)
32
33 now = datetime.datetime.now()
34 formatted_date = now.strftime("%Y-%m-%d_%H_%M_%S")
35 filename = f"KNN_model_{formatted_date}_{idle_count}_{bad_count}_{good_count}_{y.shape[0]}_images.joblib"
36
37 print(filename)
38
39 #joblib.dump(model, filename)
40

```

Figure 4. Screenshot of code 2

The code provided is responsible for training a machine learning model to classify squat form as either good, bad, or idle. It begins by importing necessary libraries such as pandas for data manipulation, scikit-learn for machine learning functions, joblib for model saving, and datetime for timestamping outputs. The dataset is loaded from a CSV file called "Output.csv," which contains pose detection information extracted from previous video frames. The features (pose coordinates) are separated from the target labels (form classification), and the data is split into training and testing sets with a 70/30 ratio. A K-Nearest Neighbors (KNN) classifier is then initialized with three neighbors and trained using the training data [14]. After training, the model predicts squat form labels for the test set and calculates the model's accuracy. Additionally, the code counts how many samples belong to each class (idle, bad form, good form) to provide a better understanding of the dataset distribution. Finally, it formats a unique filename containing the current date and counts of each form class, preparing it for saving the trained model, although the actual saving command is commented out. This script is essential for evaluating the model's baseline performance before deployment and ensures that the model can generalize well to unseen squat data.

The last major component of the project is the recorder. Another file pops up a camera window which is how the AI is able to train off a live video. Upon opening the recorder, the AI will take each frame and classify it as either good or bad form, which is then projected onto the screen and said out loud.



Figure 5. Figure of bad form

```

with mp_pose.Pose(
    static_image_mode=False,
    model_complexity=0,
    enable_segmentation=True,
    min_detection_confidence=0.5,
    min_tracking_confidence=0.5) as pose:

    print("Press Q key to exit")

    while cam.isOpened():
        ret, cameraFrame = cam.read()
        if not ret:
            print("Error reading camera")
            break

        cameraFrame = cv2.flip(cameraFrame, 1)

        results = pose.process(cameraFrame)

        data = []
        if results.pose_landmarks != None:
            for landmark in results.pose_world_landmarks.landmark:
                data.append([landmark.x,
                             landmark.y,
                             landmark.z,
                             landmark.visibility])

        df = pandas.DataFrame(data)
        df = df.T
        df.columns = headers
        pred = model.predict(df)

        ts = (time.time() // 1)

        cameraFrame = addText(cameraFrame, pred)
        cv2.imwrite("output/" + ts + ".png", cameraFrame)

```

Figure 6. Screenshot of code 3

This code captures live video from a camera, processes the frames using Mediapipe's pose estimation model, and uses a trained machine learning model to predict squat form in real time [15]. It initializes the Mediapipe pose detector with moderate detection and tracking confidence thresholds, enabling segmentation for better landmark tracking. A camera feed is opened, and while it remains active, each frame is read, flipped horizontally for a mirror view, and passed through the Mediapipe model to detect human pose landmarks. If pose landmarks are successfully detected, the x, y, z coordinates and visibility values for each landmark are collected into a list called data. This data is then converted into a pandas DataFrame and transposed to match the expected format. Using a pre-trained model, a prediction is made on the user's form (good, bad, or idle). The prediction result is added as text onto the frame for user feedback. Finally, each processed frame is saved as an image file with a timestamped filename for record-keeping. This real-time loop allows the system to give immediate feedback about the user's squat form, helping prevent injuries by notifying them as they move, rather than after completing the exercise.

4. EXPERIMENT

4.1. Experiment 1

One possible blind spot in the program is the AI's accuracy when detecting good or bad form. It is essential that it is as high as possible in order to get the most accurate results to prevent any possible injuries.

In order to test the accuracy of the AI, there needs to be 2 different types of data. The first type of data needed is the training data, which should consist of about 90% of the entire original data. The second type of data needed is the testing data, which is used to test the AI model's accuracy. After the AI model trains itself off the data, it will test the accuracy of itself by analyzing a test image, producing a guess, and then comparing it to the actual answer. If everything is guessed correctly, it will have an accuracy of 100%.

Configuration	Good Form	Bad Form	Idle	Accuracy (%)
Test 1	150	150	200	89.3
Test 2	200	200	100	93.7
Test 3	180	180	140	91.2
Test 4	130	170	200	88.5

Figure 7. Table of experiment 1

The experiment showed that the dataset balance significantly affects the AI's performance. The highest accuracy (93.7%) was achieved when the training data included 200 examples each of good and bad form and only 100 examples of idle. This indicates that the model benefits from more examples of actionable squat data (good/bad) and fewer idle frames. The mean accuracy across all tests was 90.7%, and the median was 90.2%. The lowest result was 88.5%, which occurred when the data was unbalanced in favor of idle poses. The highest accuracy result confirmed the importance of balanced datasets for each class. A possible reason for these outcomes is that idle data is easier to classify, but too much idle data can cause the model to neglect small form differences in active squats. The biggest factor influencing results was the data distribution across the three labels.

4.2. Experiment 2

Another possible blindspot of the program is the variability of the different types of squats. For example, elevated squats where people have their heels slightly elevated on weights might be detected differently for good or bad form.

In order to test how much differently an elevated squat will affect the result of the AI, there needs to be straight testing. I can record myself doing elevated squats with good and bad form to see whether or not the results are still accurate. I can also add data with elevated squats in the training data to ensure that elevated squats can be recorded with the program accurately. After putting in the elevated squat data, I can retest the AI and compare how the two datasets varied: the one without the elevated squats, and the one with the elevated squats.

Test Phase	Accuracy on Elevated Squats (%)
Without Elevated Squats	82.4
With Elevated Squats Added	91.5

Figure 8. Figure of experiment 2

Before adding elevated squat data, the AI struggled with these squat variations, achieving only 82.4% accuracy. This makes sense because the elevated heel position changes joint angles, confusing the model trained exclusively on flat-foot squats. After incorporating 50 elevated squat samples into the training set, the model improved to 91.5% accuracy on elevated squat detection. This confirms that the system can adapt to variations in squat style with sufficient training data. The results suggest that the AI is not inherently biased against different squatting techniques but simply needs exposure to varied forms during training. The experiment also demonstrates the model's flexibility and highlights the importance of including diverse training data when designing real-world fitness AI systems. Moving forward, adding more squat variations (e.g., sumo squats, front squats) could further enhance the program's generalizability.

5. RELATED WORK

The goal of the experiment in the source above is to find out whether AI or physical therapy is better for squat evaluation [5]. The people did three types of squats, and each one was evaluated by an AI. After a squat, the AI would say out loud whether or not it was good or bad form with justification. However, it was found that the AI had an okay ability to identify correct squat form versus a very limited ability to identify an incorrect squat form. My program has more accurate visions of what the AI can identify as either idle, good, or bad form opposed to the other program which claimed to have a limited ability.

This was similar to the previous experiment, comparing whether an AI program would be better than a physical therapist for squat evaluation [6]. However, it was only evaluating when people would do a normal squat with their feet planted on the ground normally. The AI would first give feedback on what the squatter did well, and then provide feedback on specific parts that the squatter needed to fix. While my program cannot capture bad forms of all types, it can capture the squatter from any distance, unlike the program that can only capture the squatter at exactly 3 meters away.

This experiment is much more similar to my program because it solely tries to identify a way for an AI to prevent squatting injuries [7]. It also uses a mediapipe to train the model and to identify good or bad form. It also classifies the squat into seven types of form, including different types of bad forms. It also had a 94% accuracy from the testing data and was supposedly the best in the field of research. However, the results only came after the set was done, so there would have been no way for the squatter to know whether or not they had good or bad form in the middle of a rep. However, my program is able to speak out loud to let the squatter know when to fix their form at a certain point.

6. CONCLUSIONS

The one major part of my project that could be improved is the type of bad form that is recorded. Currently, the AI distinguishes between good and bad form well, but the user might not know what specifically to fix their form on. The training data for good versus bad form is distinguishable through how large the angle between someone's thigh and upper body is. If the

angle is too small, it would be classified as bad form while good form would be at a natural position at about a 70 degree angle. While this part of the program is very accurate, the program would be significantly better if it was able to specify the type of bad form so the user could fix it as soon as possible. For example, if somebody wasn't going deep enough when squatting or too deep, the AI would say something to prevent that from happening again on the next rep.

In conclusion, this project demonstrates how artificial intelligence can actively prevent injuries during squatting exercises. By providing real-time feedback and immediate corrections, the AI improves athlete safety and performance. Although there are limitations, this project establishes a strong foundation for future development in fitness-based injury prevention using machine learning and pose estimation.

REFERENCES

- [1] Alekseyev, Kirill, et al. "Identifying the most common CrossFit injuries in a variety of athletes." *Rehabilitation process and outcome* 9 (2020): 1179572719897069.
- [2] Visser, Bart, and Jaap H. van Dieën. "Pathophysiology of upper extremity muscle disorders." *Journal of electromyography and kinesiology* 16.1 (2006): 1-16.
- [3] Vanholder, Raymond, et al. "Rhabdomyolysis." *Journal of the American Society of Nephrology* 11.8 (2000): 1553-1561.
- [4] Schoenfeld, Brad J. "Squatting kinematics and kinetics and their application to exercise performance." *The Journal of Strength & Conditioning Research* 24.12 (2010): 3497-3506.
- [5] Luna, Alessandro, et al. "Artificial intelligence application versus physical therapist for squat evaluation: a randomized controlled trial." *Scientific Reports* 11.1 (2021): 18109.
- [6] Luna, Alessandro, and Michael W. Denham. "AI provides congruent and prescriptive feedback for squat form: qualitative assessment of coaching provided by AI and physical therapist." *Journal of Comparative Effectiveness Research* 11.14 (2022): 1071-1078.
- [7] Chariar, Mukundan, et al. "AI trainer: Autoencoder based approach for squat analysis and correction." *IEEE Access* 11 (2023): 107135-107149.
- [8] Kramer, Oliver. "K-nearest neighbors." *Dimensionality reduction with unsupervised nearest neighbors*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013. 13-23.
- [9] Alsobhi, Mashael, et al. "Physical therapists' knowledge and attitudes regarding artificial intelligence applications in health care and rehabilitation: cross-sectional study." *Journal of medical Internet research* 24.10 (2022): e39565.
- [10] Lugaresi, Camillo, et al. "Mediapipe: A framework for building perception pipelines." *arXiv preprint arXiv:1906.08172* (2019).
- [11] Singh, Amritanshu Kumar, Vedant Arvind Kumbhare, and K. Arthi. "Real-time human pose detection and recognition using mediapipe." *International conference on soft computing and signal processing*. Singapore: Springer Nature Singapore, 2021.
- [12] Mitlöhner, Johann, et al. "Characteristics of open data CSV files." *2016 2nd International Conference on Open and Big Data (OBD)*. IEEE, 2016.
- [13] Culjak, Ivan, et al. "A brief introduction to OpenCV." *2012 proceedings of the 35th international convention MIPRO*. IEEE, 2012.
- [14] Batista, G. E. A. P. A., and Diego Furtado Silva. "How k-nearest neighbor parameters affect its performance." *Argentine symposium on artificial intelligence*. Princeton, NJ, USA: Citeseer, 2009.
- [15] Insafutdinov, Eldar, et al. "Deepcut: A deeper, stronger, and faster multi-person pose estimation model." *European conference on computer vision*. Cham: Springer International Publishing, 2016.