Workout Assistant Using Computer Vision

Dominic Quintero
Computer Science
UC Davis
daqquintero@ucdavis.edu

Billy Ouattara
Computer Science and Engineering
UC Davis
btouattara@ucdavis.edu

Jose Gavidia
Computer Science and Engineering
UC Davis
jgavidiapaz@ucdavis.edu

Abstract

This paper presents a basic workout assistant designed to classify and evaluate different types of exercises and the correctness of users' posture. Our approach uses machine learning models to provide feedback, ensuring that users perform exercises with the proper form to maximize effectiveness and minimize the risk of injury. We use a neural network that uses the key points of individuals in an image to determine the type of exercise, movement, and correctness. Our model can differentiate between the following exercises: squats, pushups, and deadlift positions but can be expanded to any exercise type. It can predict flexion or extension movements. Lastly, it can assess exercise correctness to provide feedback on the user's posture.

Our results show that the application significantly enhances workout quality and safety, making it a valuable tool for fitness enthusiasts and professionals. Future work could focus on the range of detectable exercises, refining the correctness model for even greater precision, and providing a visual overlay that the user can follow.

I. Introduction

Proper exercise form is crucial to minimizing the risk of injury. However, many individuals either don't have access to personal trainers or friends who can provide accurate feedback, or they are too shy to seek guidance. The pandemic has further highlighted this gap, underscoring the need for an accessible, automated solution that can assist users in exercising correctly to prevent serious injury. Recent advancements in machine learning and computer vision present an opportunity to develop such solutions.

II. Methodology

A. Data Collection / Processing

The first datasets used, “Workout Fitness Video” [1] and “Pushup” [2], were obtained from Kaggle, and fulfill three important preconditions for our models:

1. A label for the type of workout being performed
2. Clear, full-body view of the exercise (minimal or no obstructions if applicable)
3. A single person in the view performing the exercise.

Correctness at this stage doesn't matter, but the larger the dataset demonstrating the possible range of motion (ROM) during an exercise, the better the result. With these preconditions filled, we can use multiple approaches for data processing. When we process an image, we also process the horizontally flipped version to double the size of our dataset.

The second dataset used, the Kinetics Dataset [3], has 600 human action classes. The annotations corresponding to the pushups, squats, and deadlift classes were extracted from a CSV file. These annotations include the name of the exercise, the IDs for YouTube videos where the exercise is performed, and the starting and ending times. The library ‘yt-dlp’ was used to download the videos, and ‘ffmpeg’ was used to trim the video and extract frames. Furthermore, we cleaned the frames using MediaPipe by checking the visibility of the necessary key points. Finally, we created an annotation file corresponding to our cleaned frames. The file includes the image path, exercise type, key points, movement type, and correctness. The key points were extracted using MediaPipe. We wrote functions to determine movement type and correctness based on joint angles.

Our datasets focus on pushups, squats, and deadlifts but the methods described in the next sections can theoretically be applied and expanded upon to any exercise.

B. Naive Classification and Correctness

Since exercise is a full-body activity, we can accurately map an exercise's ROM to a single label using key points, where each key point is a tuple (x, y, z, visibility). Initially, we attempted to use angles of the important key points involved in a specific exercise for classification and correctness.

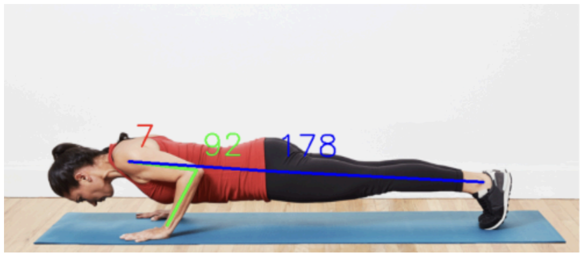

Figure 1. Naive Pushup Approach Using Angles

With fine-tuning to a specific position and orientation this approach worked. However, we found it impractical due to its heavy reliance on the user's positioning and estimations required for describing a proper ROM. Anything outside the expected position and orientation would be falsely labeled in both classification and correctness. This highlighted the need for a more generalized approach. We identified the need for two models: a classifier and a correctness model.

C. Exercise Classification

The classifier still uses the key points described in the naive implementation section. We first created the classifier to guide our efforts towards correctness.

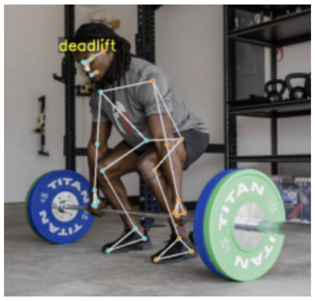

Figure 2. Classifier Visualization

We trained the classifier to recognize the exercise being performed given the key points. A key point is described as a tuple (x, y, z, visibility). We trained a CNN using key points and a label corresponding to the data folder. During training, our model achieved an accuracy of approximately 90%. When tested with validation data, our model produced accurate results as seen in the figure above, suggesting we're on the right track. We think that we achieved surprisingly high accuracy because we removed image noise factors, enforced our data preconditions, and only trained off the ROM of each exercise.

However, we would like to have classification and correctness prediction capabilities within the same model. We will expand the capabilities of the classifier in section E of the paper.

D. Exercise Correctness Prediction Based on Images

One of our approaches to predict the correctness of exercises was to build a convolutional neural network that would classify an input image as correct or incorrect. Our dataset consists of exercises like squats, deadlifts, and push-ups. Each exercise type has about 4500 images depicting both proper and improper posture. To ease the processing of the data, each image is converted to greyscale, resized to 64x64, and normalized before being processed.

The model processes a 64x64 grayscale image as input, feeding the image data into a 2D convolutional layer. We apply max pooling on the output to extract the important features and repeat the process on the second 2D convolutional layer. We use the remaining four fully connected layers to reduce the dimensions and integrate the features extracted by the convolutional layers. The last layer is a single node that outputs whether the exercise was done correctly or not. Since we are predicting binary values, we used the binary cross-entropy loss function.

Using this neural network and our data, we achieved a loss convergence of 0.0364. However, when evaluating the model on the test set, we observed an accuracy of approximately 65%. After careful analysis, we concluded that the model behaved that way because there was too much variety in the exercises. We had a total of three combined exercises, with their correctness described. It could be that, due to the variety of images, the model had a hard time predicting the correctness. This result led to our third and final approach: Multi-variable classification using key points.

E. Multi-variable Classification using Key Points

For this approach, we decided to use key points because:

1. Processing key points is less computationally expensive than processing images.
2. Key points capture only the essential information needed for our task while removing irrelevant information.
3. The model becomes less variant to the specific visual features of the images.

Given that our data has been preprocessed and annotated with image path, exercise type, key points, movement type, and correctness, we decided to maximize our predictions and have three output heads: Exercise head, movement head, and correctness head.

The model is a simple neural network (NN) with two fully connected layers. The first layer maps the input features to a 128-dimensional space. The input features correspond to the values of the coordinates of every key point. There are 33 key points, each described as a tuple (x, y, z) coordinate (the visibility parameter was removed), resulting in 99 elements. The second layer has 64 dimensions. We also used ReLU to introduce non-linearity.

We achieved 88.8% exercise classification accuracy, 78.3% movement classification accuracy, and 83.5% correctness classification accuracy. These results are satisfactory given the simplicity of our model and the variety of data used (about 14,000 total images, containing diverse scenarios and settings).

III. Conclusion

The use of key point estimation libraries such as MediaPipe with neural networks has shown effectiveness in exercise classification and assessment tasks. Using key points instead of images to train NN increases the training performance and accuracy of the model. With larger datasets and improved data preprocessing, we believe we can achieve greater accuracy and provide more value to people adopting an exercise routine. Some areas of improvement include demonstrating our methods with a wider range of exercises and exploring the feasibility of developing a comprehensive classification and correctness model on a larger scale. We believe this is plausible, although it was not explored in this study.

IV. Contributions

By Methodology Section:

Dominic Quintero: A, B, C

Billy Ouattara: A, B, D

Jose Gavidia: A, B, E

V. References

Datasets

[1] H. Abdillah, “Workout Fitness Video,” Kaggle. Available: https://www.kaggle.com/datasets/hasyimabdillah/workoutfitness-video

[2] M. A. Salama, “Pushup,” Kaggle. Available: https://www.kaggle.com/datasets/mohamadashrafsalama/pushup

[3] C. Kay, J. Carreira, K. Simonyan, and A. Zisserman, "The Kinetics Human Action Video Dataset," Papers with Code. Available:https://paperswithcode.com/dataset/kinetics. [Online]. Available: https://github.com/cvdfoundation/kinetics-dataset.

Figures

[1] “The Push-Up Exercise,” Verywell Fit. Available: https://www.verywellfit.com/the-push-up-exercise-3120574

[2] “Deadlift vs Romanian Deadlift,” Titan Fitness. Available: https://www.titan.fitness/blog/deadlift-vs-romanian-deadlift.html