Pose Estimation to Provide Personalized Posture Feedback During Home Exercises

Lyna Kim
Stanford University
lynakim@stanford.edu

Mehul Arora
Stanford University
aroram8@stanford.edu

Abstract

Accessibility to home exercise has become critical to individual physical fitness, particularly in the days of restricted physical activity during the COVID-19 pandemic. However, the repeated strenuous movements characteristic of workouts can be detrimental and cause injury if performed in the wrong posture. Thus, we introduce a method of safely monitoring and correcting workout postures by utilizing a single image input to output printed warnings and form improvement recommendations when the user in frame significantly deviates from the correct posture. We are able to leverage a CNN-based workout classification, an open-source pose estimation network, and a heuristic body geometry analysis component of our model to reliably provide users with useful form feedback on four different exercises: the deadlift, the overhead press, the bench press, and the squat.

1. Introduction

The recent COVID-19 pandemic has forced millions of people to quarantine at home, leaving them unable to leave for regular outdoor exercise or gym visits. After it became clear in early 2020 that COVID-19 was here to stay, people began to find indoor, quarantine-friendly alternatives to their regular exercise methods. As a result, home training and various home-based fitness trends such as Chloe Ting have taken off, their popularity and appeal spanning a diverse audience due to their broad applicability for general fitness during a period of severely restricted physical activity.

However, few people know of the importance of proper posture and form during these exercises, and thus the rise in home training opened the way to an increased wave of injuries from inexperienced workout beginners[1]. The repeated stress by working out with the wrong form and straining the wrong muscle groups led to injuries and chronic pain in many such individuals. Unfortunately, the most obvious solution, which is to receive constant monitoring from a trained personal trainer to guide posture, is unaffordable and impractical for most individuals. Not only this, in the light of social distancing protocols due to COVID-19, it can even be impossible to receive that level of personal attention.

Thus, we propose an application that will allow users to film themselves as they perform a physical exercise and receive feedback on their posture. The input will be a picture of the user in position to begin the exercise, and the user visible output will be feedback on their posture - for example, ”Good job! You are in the correct posture.” or ”Make sure to tense your core to prevent your shoulders from slumping.” The program will identify mistakes in their posture and provide the user with targeted feedback to help them correct their posture and form, with the ultimate goal of helping maximize the effectiveness of every workout and save the user from injuries. In the long run and scaled, this application also has the potential to be deployed in gyms and other training facilities to make working out a more effective and less injury-prone experience for everyone.

2. Related Work

Pose estimation as a computer vision problem attempts to infer the pose of an object in an image, which for humans is usually done by identifying anatomical keypoints on a person (such as right elbow, left knee, etc.). Human pose estimation, however, introduces an additional challenge, as our body points can move relative to each other depending on our pose (e.g. we can decide to raise our arm, which may move the relative position of our right elbow from beneath our right shoulder to above the right shoulder). This is more difficult than rigid pose estimation, which assumes the keypoints of an object are the same distance apart regardless of orientation (e.g. the corners of a brick).

The initial work on human pose estimation was done by Toshev et al. [14], who used a seven layer deep neural net regression to find the location of each joint. Carreira et al. [5] pioneered top-down feedback for pose estimation, which iteratively detects each instance of an object (body part) and estimates the keypoints within each region. Newell et al. [11] builds on this research with a stacked hourglass neural network architecture that repeatedly uses bottom-up and top-down processing, where bottom-up first finds the keypoints then draws the skeleton connecting the joints. The key datasets for human pose estimation are MPI, LSP, and FLIC, and each of these papers in turn produced state-of-the-art results on these.

The field developed into pose estimation for multiple people in a single frame, which was revolutionized by Pishchulin et al. [13] with DeepCut, using an adapted Fast R-CNN to first identify the number of people in the image and a modified VGG for each individual's pose estimation. Xiao et al.[15] build on this work to track individuals across multiple frames using deconvolutional layers added to a ResNet architecture. OpenPose by Cao et al.[4] created a realtime, open-source pose estimator that uses nonparametric part affinity fields to combine all functionalities described above in a lightweight system that can be run on a web browser.

On the body kinematics side, Zell et al. [16] uses body geometry, such as the angle between different joint pairs, to track human motion throughout a motion. Complementary to this work is Bonnet et al. [2], which identifies the optimal body geometry during a squat. We lean on similar works for different exercise types to establish the baseline ”good posture” for each exercise, then compare against the keypoints of the pose estimation output, based on which we provide users with feedback.

3. Methods

3.1. Pipeline Overview

We begin by describing the overall model pipeline, from user input to feedback output, as outlined in Figure 1.

The precise goal of our project is a model that, when provided with a picture of an individual engaged in a workout, will be able to perform two key tasks: one, it must be able to classify the kind of workout or exercise that the user in frame is engaging in; and two, it must be able to perform pose estimation to extract body geometry and compare it against the proper body geometry of that exercise, identifying and outputting any significant deviation as a warning to the user.

Proceeding through the pipeline step by step to model a run process from beginning to end, we begin with user input, an image of their workout position. For this input, our model is able to accommodate a wide variety of image types. In particular, assuming users are using their phones to film their exercise posture, the most common file types (.jpg, jpeg, png) are all able to be processed to return the expected output to the user without problem. Once the user

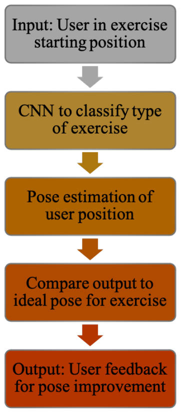
Input: User in exercise starting position
CNN to classify type of exercise
Pose estimation of user position
Compare output to ideal pose for exercise
Output: User feedback for pose improvement

Figure 1. Model pipeline describing model output process. Beginning from user input of exercise image, the model classifies the exercise, estimates the user's pose, and compares it against the optimal posture for the exercise to provide personalized feedback.

has uploaded a picture, we have constructed a CNN to take the user input image and classify the image as one of four supported exercises: deadlift, bench press, overhead press, and weighted squat. After this step, we find a pose estimation for the user in the frame, and compare the pose estimate to the empirically defined ideal body geometry of the exercise that was predicted for the user. Based on this comparison, the user is provided with feedback on their form for the exercise, with recommendations and warnings if the user is exceeding safe kinesiological ranges of motion.

3.2. Dataset

For our initial dataset, we begin by web scraping images for each category of our compatible exercise types: deadlift, bench press, overhead press, and weighted squat. For this initial dataset, we first run our images through a preprocessing pipeline, then an augmentation pipeline. As a note, for each exercise, we were careful to ensure that the images we initially collected were taken from a variety of angles - for example, a front facing view and a side facing view of a squat - to ensure that no matter the picture angle of our user input, our model would be robust and able to correctly classify the exercise.

3.2.1 Data Preprocessing

To preprocess the images, we resize the images to a constant dimension of $256x256$, maintaining the aspect ratio setting the larger dimension of width and height to 256, resizing the other dimension to scale, and adding padding to either end of the smaller dimension to make the final image square. We then change the images to a tensor and normalize, after which we pass the data to the augmentation step.

3.2.2 Data Augmentation

We then separately augment our dataset in four different ways: a random rotation clockwise or counterclockwise up to 15 degrees, with a probability of 0.5, to account for picture angle; horizontal flip, with a probability of 0.5, to account for perspective relative to the camera; greyscale, with a probability of 0.2, to account for extreme lighting differences or technological restrictions; and color jitter, with a probability of 0.8, to account for differences in hue, brightness, saturation, and contrast of user input images. Each of these transforms were selected because of their applicability to covering variance in user input image quality. After the augmentation process, we verified that there was an even split between categories before moving on to training, and made a random $90 : 10$ train-test split.

3.3. Exercise Classification

We construct the CNN intended to take the user input and predict what the kind of exercise the user is engaging in is. Figure 2 below depicts the architecture of our CNN. Similar to AlexNet[9], our model broadly consists of convolutional layers followed by pooling, with some added normalization to prevent overfitting.

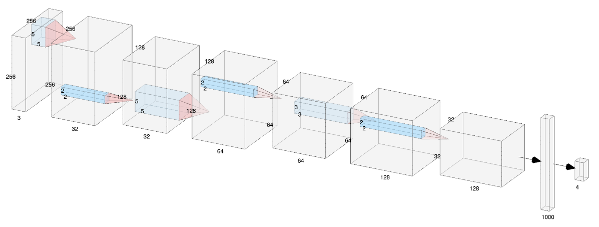

Figure 2. CNN model architecture used to predict user input's exercise type. Consists of three blocks, with each block consisting of a convolution, batch normalization, leaky ReLU, Dropout, and max pooling layer. Ends with two fully connected layers.

Specifically, the model takes in a classic $3x256x256$ RGB image with 3 channels, which is produced from user input through our data processing pipeline, and outputs a probability distribution over the four possible exercise types, where the exercise with the largest probability is the model prediction. We trained for 20 epochs with a batch size of 4, using the standard Adam optimizer. The performance and results of this model are discussed in Section 4.

3.4. Pose Estimation

3.4.1 Estimator Model Selection

For the third part of our pipeline as depicted in Figure 1, we must take the user image as input and output a pose estimation. There are several simplifications we can make in the context of the literature described in Section 2. First, because we assume that the user will be the only one in frame, both due to the independent home training problem this model is intended to address and the confusion that may be caused by providing simultaneous feedback to multiple users in the same frame, we are able to work with either single or multi-person pose estimator models. Second, since we are dealing with image inputs from the user rather than a time series input, we are able to simplify the problem of tracking movement across frames and only consider the pose of the user in the frame of the given picture.

Given the intentionally lightweight demands of our pose estimation needs to improve ease of use, we decide to proceed with a modified version of the pre-trained OpenPose[4] model, for the pose estimation part of our pipeline. This is because OpenPose is one of the easiest pose estimators for users, particularly because of their native GUI, and we aimed to ensure that our product was user friendly and scalable to relatively inexperienced, new users quickly. Most other pose estimators we considered, from Section 2, either required too much computational power or were difficult for the user to install without significant computing background. Thus, we proceeded with the best option available.

3.4.2 OpenPose Specifications

The OpenPose architecture, as depicted in Figure 3, consists of two connected stages, iteratively predicting a set of confidence maps S of body part locations within the image and a set of vector fields L of part affinity fields (PAFs). PAFs encode unstructured pairwise relationships between body parts for an undetermined number of people and can produce pairwise scores without an additional training step, which is a major advanceemnt of the OpenPose architecture. The predictions are refined over successive stages for $t \in \{1, \dots, T\}$, with intermediate supervision between each stage.

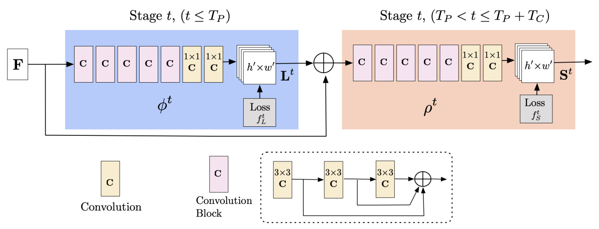

Figure 3. Multi-stage CNN architecture used in OpenPose. The blue stage predicts PAFs L^t, while the red stage predicts confidence maps S^t for each body part.

Tracing through the OpenPose pipeline in more detail, a user input is used to generate a set of feature maps F, which is input to the first stage to produce a set of PAFs. This feature map is used to produce a set of predicted PAFs, $L^1 = \phi^1(F)$, using the stage 1 CNN ϕ for inference. At each successive stage t up to the total number of PAF stages T_P, the predictions from the previous stage and the original image features are concatenated to produce the PAFs for stage t.

$$L^t = \phi^t(F, L^{t-1}), \forall 2 \le t \le T_P \tag{1}$$

After T_P iterations, this is repeated for T_C confidence map predictions, with each stage defined as the following.

$$S^{T_P} = \rho^t(F, L^{T_P}, \forall t = T_P \tag{2}$$

$$S^t = \rho^t(F, L^{T_P}, S^{t-1}), \forall T_P < t \le T_P + T_C \tag{3}$$

Throughout the training process, the overall objective is defined as the summed L_2 loss between the estimated predictions and the groundtruth PAFs and confidence maps as described in Equation 4, with the intermediate supervision replenishing the gradient.

$$f = \sum_{t=1}^{T_P} f_L^t + \sum_{t=T_P+1}^{T_P+T_C} f_S^t \tag{4}$$

Using this pre-trained model on an input image by users, OpenPose outputs a list containing the coordinate predictions of all keypoint locations, which include 18 different body features such as the neck, elbows, knees, or hips, which we then use in the next part of our pipeline to evaluate our user's pose.

3.5. Optimal Pose Comparison

3.5.1 Optimal Body Geometry

Deadlift: The correct posture for the deadlift involves the chest protruding outward, the spine in neutral position (not arched) and the stance at shoulder width. [12]

Overhead press: The correct posture for the overhead press involves the chest being slightly bent backwards, the forearm close to the chest and the stance at shoulder width. [10]

Bench press: The correct posture for the bench press involves the arms being opened up at about a 45 degree angle relative to the torso and the legs being behind the knees to allow the user to push easily.

Squat: The correct posture for a squat involves standing straight with your stance at shoulder width. [3]

3.5.2 Identifying Key Pose Elements

We use OpenPose to get the keypoints of joints in the input image. Using these keypoints, we can calculate angles by considering the image to be a 2D plane and the keypoints to be coordinates on this plane. The values of interest can be calculated as below.

Stance: We check for differences in the user's leg width and shoulder width in Deadlift, Overhead press and Squat. We do this by using the left and right ankle keypoints for leg width and left and right shoulder keypoints for shoulder width and calculating the Euclidian distance between the two.

Deadlift: In order to consider the arch in a user's back, we calculate the angle between the vector joining the torso and neck, and the vector joining the neck and the eyes. In order to find the vector joining the torso and the neck, we calculate the vector from the neck keypoint to both the left and right hip keypoints, and average the two.

Overhead press: In order to make sure that the user's forearm is not too far from their shoulder, we consider the angle between forearm and bicep. To do so, we get the vector between the shoulder and elbow keypoints, and calculate its angle with the vector between the elbow and wrist keypoints.

Bench press: Here, we need to make sure that the and between the user's upper arm and torso is not too large. We do this by calculating the vector between the user's elbow and shoulder keypoints and calculating the vector between the user's shoulder keypoint and hip keypoint, which is a good estimator for the torso vector. We then take the angle between the two, and make sure it is not too high. Secondly, we make sure that the angle between the user's leg and thigh is shallow, i.e. below 80 degrees. We do this by calculating the angle between the vectors connecting the ankle and knee, and knee and hip.

4. Results

4.1. CNN Exercise Classification

Our CNN reached a max accuracy on the training images of 86%. The final accuracy per class outlined below in Table 1.

Exercise	Accuracy
Deadlift	1.0
Overhead Press	0.71
Bench Press	1.0
Squat	0.92

Table 1. Final test set classification accuracy, by exercise class.

The first key aspect to note from the accuracy is that there is most likely not enough data. Note that the model is achieving a perfect accuracy for two of the four classes, while underperforming on the other two classes, indicating that there should be a larger dataset for both the training and test sets, since it may be that the model is not learning the key features of each exercise class. The other possibility is that our model is overfitting, despite the regularization of batch normalization and dropout, as well as the data augmentation, we built into our model.

In order to improve the interpretability of our results, we referenced Stanford's CS 231N Assignment 5 code to produce a heatmap for the first convolutional filter of our CNN, which is a stack of 32 $5x5$ filters, as shown in Figure 4. However, although the first CONV layer weights should be most interpretable since there is a direct relationship to the raw pixel data, we unfortunately find that there is no clear pattern discernable in the weights. Due to the noisiness in the pattern, it may be necessary to add more regularization. The other viable option, which is to train the model for longer, is not viable because the loss curve (not pictured) for our model bottoms out by the time our model finishes training.

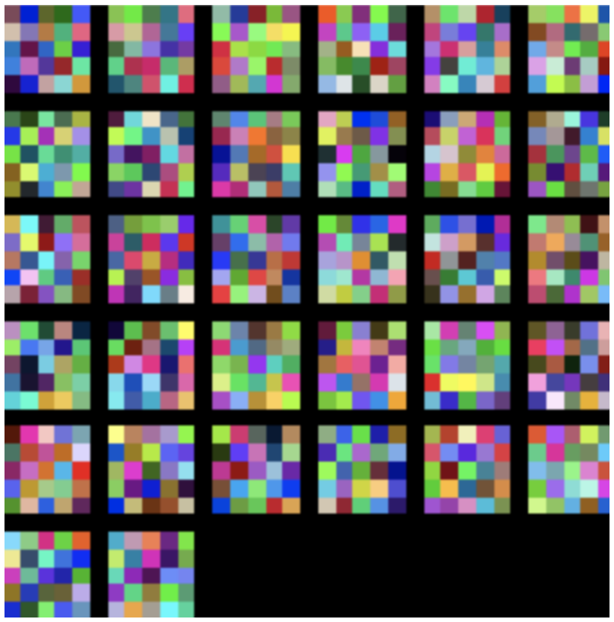
Figure 4. Visualization of the first convolution layer in our model, which is a 32 stack of 5x5 filters.

4.2. Pose Estimator & Body Geometry Parsing

Running the OpenPose architecture produced a vector of body joint coordinates, which we then visualized, pulling from the OpenCV repository [7] to individually draw line and ellipses in between each joint, and produced a skeleton as seen in Figure 5

We produce a similar skeleton image for every sample user input image, and print the feedback statement for the user based on our body geometry analysis described above. See specific output results for each exercise below.

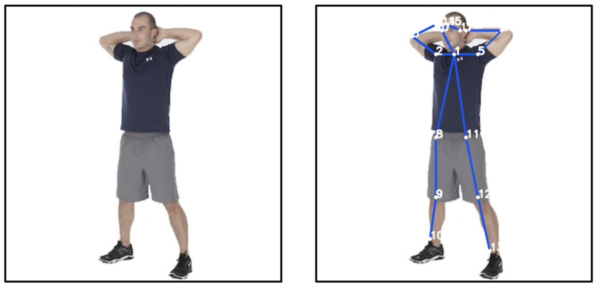
Figure 5. Visualization of OpenPose pose estimation output, with input image (left) and the overlaid skeleton image (right). In the skeleton, each white dot represents a predefined body part, and the blue lines represent connections between joints.

4.3. User Outputs by Exercise

4.3.1 Bench Press

If the user exercise is classified as bench press, we do one of three things:

1. Tell the user to tuck their elbows in if the angle between their upper arm and torso is greater than 60 degrees.
2. Tell the user push their feet back if their the angle between their feet leg and thigh is above 80 degrees.
3. Encourage them for performing the exercise correctly!

4.3.2 Overhead Press

If the user exercise is classified as bench press, we do one of three things:

1. Tell the user to fix their stance if the stance the difference between leg and shoulder width exceeds a threshold of 0.5.
2. Tell the user pull their forearm closer to their chest if they are too far away, i.e. the angle between them is more than 25 degrees.
3. Encourage them for performing the exercise correctly!

4.3.3 Deadlift

If the user exercise is classified as bench press, we do one of three things:

1. Tell the user to fix their stance if the stance the difference between leg and shoulder width exceeds a threshold of 0.5.
2. Tell the protrude their chest outwards if the angle in their back is less than 165 degrees.
3. Encourage them for performing the exercise correctly!

4.3.4 Squat

If the user exercise is classified as bench press, we do one of three things:

1. Tell the user to fix their stance if the stance the difference between leg and shoulder width exceeds a thresholdof 0.5.

2. Encourage them for performing the exercise correctly!

5. Discussion

5.1. Project Contributions

The primary contribution of this paper can be divided into two parts. First, it builds on traditional pose estimators, such as OpenPose, which was used in our work, by creating a model that utilizes the pose estimation architecture that has been developed. Fundamentally, we are incorporating prior work on pose estimation into a user product that can be leveraged to real use cases that support user health. While this project is not, and is not intended to be, a novel form of pose estimation, it is unique in its use of pose estimation capabilities to create human value.

Second, while there has been a small number of other attempts to use pose estimation posture in the context of workouts, none have the capacity of our model to both classify the exercise and provide feedback. For example, Pose Trainer [6] takes in a workout video as user input, but also requires the user to specify what exercise is being performed. This poses a significant challenge to ultimate usability that we overcome with our model - for example, consider a case where the user is uploading a large number of pictures in a folder and iteratively running all images through the model. It would be cumbersome for the user to do this on any scale beyond a handful of images if they had to manually enter the workout exercise type with each model run, as they must in Pose Trainer. Our model is able to allow increased ease of use through our added capabilities.

5.2. Common Classification Mistakes

While an upside of our model is that it is able to offer support for a number of different exercises, one of the more common mistakes our model appears to make is confusing squats with overhead presses, as seen in the confusion matrix below (Figure **??**).

Our qualitative explanation is that it is most likely because the starting positions for both exercises can look similar. Both overhead presses and squats involve the user in a standing position, and both can also involve supporting weights near the body. In order to control for this error more carefully in the future, it may be important to gather more data to help the model learn the difference or create more of a differentiation between dataset images by class. However, in terms of model expectation, it is also important to note that unless the user is using a proper 45lb bar, it can difficult for even for a human to predict whether the user is engaging in an overhead press or squat based on the starting position. Thus, it must also be understood that although the classification may fail for these two categories, the model will still be able to fulfill its function of ensuring the user is beginning both exercises in the proper form, as the correct starting posture will be identical for the two exercises.

5.3. Body Geometry Perspective

One shortcoming of the pose estimation part of our model pipeline is the dependence on perspective - it is unable to take perspective into account when estimating the user's pose. For example, consider the two pose estimation skeletons drawn for the bench press images in Figure 6. To a human viewer, it is evident by inspection that both users are in the correct form and have the same angle between their upper arm and torso. However, our model outputs a confirmation of good form to the first user(top) while giving a warning to the second user (bottom) to tuck in their elbows.

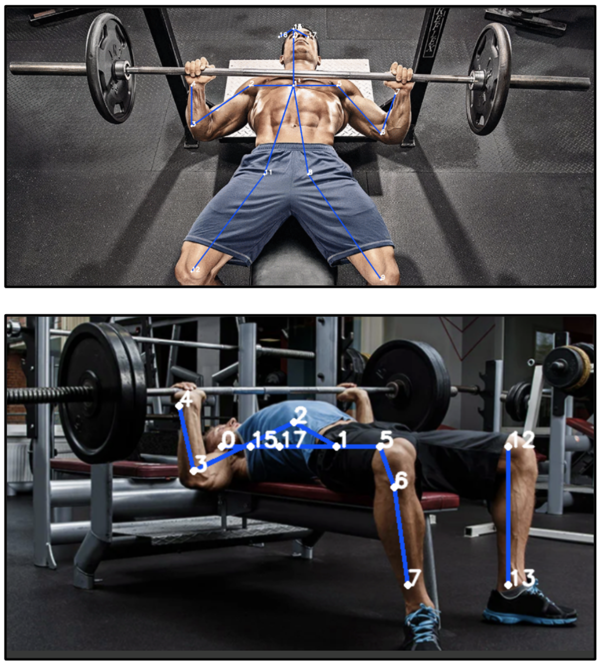

Figure 6. Pose estimation skeletons for bench press images from two different perspectives. Note that the angle between their upper arm and torso, though it appears equal for the two images in the eyes of a human viewer, is significantly different in the raw skeleton.

This inability to account for perspective is because OpenPose, our pose estimator, is a 2D pose estimation model, and thus does not factor in perspective depth when constructing the pose estimation. This is an aspect of the model that, while it is producing the expected behavior, is not producing the correct output for users. Possible solutions will be described in Section 5.4.

5.4. Future Work

We recommend three improvements to our model for future work.

One, as described in Section 5.3, out model is unable to account for the perspective of the image when providing feedback to users, as the pose estimation model we utilized, OpenPose, is a 2D pose estimator. Thus, future work should center around incorporating a 3D pose estimator to gauge user body geometry, and establish a new baseline for 3D pose estimation joint angles. This will allow the model to provide more diversified, higher quality feedback to users and avoid errors stemming from lack of depth perception.

Two, our current model requires the user to take an image of themselves and run it through our model pipeline, which will then print a terminal message to provide the user with form feedback. However, this comes at a clear time delay, while the goal should be to provide the user with feedback in real time. Thus, we recommend attempting to supplement to the pipeline so that the images can be input and run through the model in real time, perhaps by linking the model input it to a phone or laptop camera, and give audio output feedback to the user by reading the feedback messages, currently printed, out loud. This would improve the overall usability of the model.

Finally, our model is able to provide user feedback for four different exercises. In the future, the model should be extended to be able to classify and provide feedback on an increased number of exercises, as well as provide a more diverse range of feedback per exercise. This will expand the use cases of our model.

6. Conclusion

In this project, we generate a pipeline for a model intended to provide form feedback to users exercising at home without proper form guidance to prevent injury and maximize the effect of each workout. We have constructed a model that is able to take in a user workout image as input and provide form feedback as an output. The model is composed of two primary parts: the workout classification and the pose estimation. For the workout classification, the model uses a CNN to predict which of four supported exercises - the deadlift, overhead press, bench press, or squat - the user is engaging in. Using that prediction, the pose estimation portion of the model generates a pose estimate for the user, which is parsed for relevant body geometry and compared against the ideal form for the predicted workout type. Based on this comparison, the model prints form feedback to the user (e.g. informing the user that feet should be shoulder width apart for a squat). The future direction of this project should focus on improving ease of use and reliability of feedback for users.

7. Contributions

Both members contributed equally to the dataset development. For the CNN classification model, Mehul took lead in the initial construction, and Lyna took lead in the visualization and interpretation. For pose estimation, Lyna took lead in getting the OpenPose-based pose estimation model running, while Mehul took lead on parsing pose estimation to identify relevant body geometry. Our code can be found on a github repository [8] as an IPython Notebook.

References

[1] Bonnie Berkowitz. Running into trouble: Eager pandemic exercisers rack up injuries. *Washington Post*, May 2020. 1

[2] Vincent Bonnet, Claudia Mazzà, Philippe Fraisse, and Aurelio Cappozzo. Real-time estimate of body kinematics during a planar squat task using a single inertial measurement unit. *IEEE Transactions on Biomedical Engineering*, 60(7):1920–1926, 2013. 2

[3] Vincent Bonnet, Claudia Mazzà, Philippe Fraisse, and Aurelio Cappozzo. Real-time estimate of body kinematics during a planar squat task using a single inertial measurement unit. *IEEE Transactions on Biomedical Engineering*, 60(7):1920–1926, 2013. 4

[4] Zhe Cao, Gines Hidalgo, Tomas Simon, Shih-En Wei, and Yaser Sheikh. Openpose: Realtime multi-person 2d pose estimation using part affinity fields, 2019. 2, 3

[5] Joao Carreira, Pulkit Agrawal, Katerina Fragkiadaki, and Jitendra Malik. Human pose estimation with iterative error feedback, 2016. 1

[6] Steven Chen and Richard R. Yang. Pose trainer: Correcting exercise posture using pose estimation, 2020. 6

[7] Inc. GitHub. Pose estimation. `https://github.com/legolas123/cv-tricks.com/tree/master/OpenCV/Pose_Estimation`, 2019. 5

[8] Inc. GitHub. Stanford-cs231n-project. `https://github.com/mehularora8/Stanford-CS231N-Project`, 2021. 7

[9] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. Imagenet classification with deep convolutional neural networks. In F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 25*, pages 1097–1105. Curran Associates, Inc., 2012. 3

[10] CSCS USAW1; Mike Jonathan PhD CSCS*D NSCA-CPT*D USAW2 Kroell, Jordan BS. 2017. 4

[11] Alejandro Newell, Kaiyu Yang, and Jia Deng. Stacked hourglass networks for human pose estimation, 2016. 2

[12] CSCS D*; Waller Michael A. CSCS-D; NSCA-CPT* Piper, Timothy J. MS. *Strength and Conditioning Journal*. 4

[13] Leonid Pishchulin, Eldar Insafutdinov, Siyu Tang, Bjoern Andres, Mykhaylo Andriluka, Peter Gehler, and Bernt Schiele. Deepcut: Joint subset partition and labeling for multi person pose estimation, 2016. 2

[14] Alexander Toshev and Christian Szegedy. Deeppose: Human pose estimation via deep neural networks. *CoRR*, abs/1312.4659, 2013. 1

[15] Bin Xiao, Haiping Wu, and Yichen Wei. Simple baselines for human pose estimation and tracking. *CoRR*, abs/1804.06208, 2018. 2

[16] Petrissa Zell, Bastian Wandt, and Bodo Rosenhahn. Joint 3d human motion capture and physical analysis from monocular videos. In *2017 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pages 17–26, 2017. 2