Correcting Fitness Poses with Monocular 3D Human Pose Estimation from a Single Image

Ben Alexander
Stanford University
balex@stanford.edu

Carlota Parés-Morlans
Stanford University
cpares@stanford.edu

Abstract

Exercise is very important for maintaining one's health, but training with poor form can easily lead to injuries. Getting personalized training can be expensive; in contrast, computer vision-based approaches hold a lot of promise for providing an effective, low-cost solution for personalized feedback, even for at-home workouts. Here we use 3D human pose estimation to provide users with low-cost, personalized, automated feedback about their joint positions and joint angles during various exercises. We re-implement a high-performing 3D pose estimation model, and provide some ablations and experiments with it. Given the low quality of the model's predictions on fitness poses when trained on Human3.6M [10], we perform fine-tuning on a custom version of InfiniteForm [28], a brand-new synthetic dataset specifically for 3D fitness poses. We find that by using a dataset that includes fitness poses, the model is able to give more accurate predictions, allowing us to provide better feedback to the user. Moreover, we create a downstream app that allows users to upload their own camera footage and get personalized analysis of their form for particular exercises. We believe this approach demonstrates that with the right model and data, potential solutions for providing automated, low-cost, personalized exercise feedback can be developed.

1. Introduction

In recent years, remote fitness and home gyms have seen tremendous growth. Amidst the COVID-19 pandemic, home workouts have become very popular and personal trainers have started to live-stream home-friendly fitness programs with a wide range of activities and intensities. Regular physical activity helps maintain physical and psychological health [14], but exercising with poor form can easily lead to injuries like muscle strains [30]. Individualized supervision by a personal trainer is often expensive, making it mostly accessible to the wealthy population. This situation has led to an increase in the demand for computer vision models that estimate human poses. These methods can provide users with automated feedback on their form during various exercises, even at home, and for a fraction of the price of a personal trainer.

The problem of monocular 3D human pose estimation, which is one of the most fundamental and challenging problems in computer vision [5], provides a great solution for automated fitness feedback. In general, 3D human pose estimation approaches aim to identify a person's body-joint locations in 3D from images or video sequences. Specifically, monocular 3D pose estimation seeks to obtain the posture of the human body from a monocular image or video. This technique is particularly interesting as it does not need complex multi-camera setups or wearable marker points [19], making it useful for a wide variety of applications such as action detection, human-computer interaction or sports motion analysis [5]. However, predicting the 3D locations of human joints (including depth) makes the problem of human 3D pose estimation quite difficult. Additionally, identifying a human 3D pose from a single RGB image adds more difficulties derived from occlusions and 2D-to-3D pose recovery ambiguities. Nevertheless, we believe this is the best approach for us because even though it is technically challenging, it provides a better user experience since it only requires footage from a single camera.

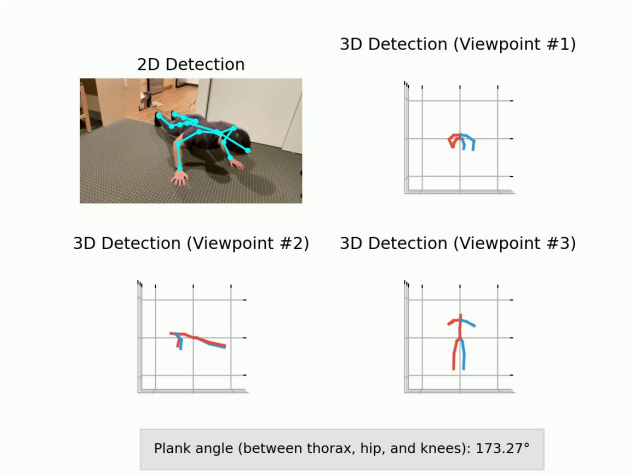


Figure 1. Example output from our system. In the 3D detections, the right side of the body is in red, and the left is in blue.

In this work, we use 3D joint keypoints to compute joint angles and provide them as feedback to the user. For example, in Figure 1, our goal is to correctly predict the positions of the joints in 3D, calculate the joint angles of interest (in this case, the angle between the thorax, hip, and knees to ensure that the back is mostly straight), and provide relevant feedback to the user. In Figure 1, we can see that the viewpoint is NOT a perfect side view, yet the pose estimation model still correctly identifies the 3D pose, and calculates an angle that appears to be quite accurate. Our full model pipeline includes both a 2D pose estimation model as well as a 3D one, for reasons we will explain later.

The main contributions of this paper are as follows:

- We compare two different 2D pose estimation models on a fitness pose dataset, ultimately choosing one due to superior performance. We then perform experiments and ablations with this 2D pose estimation model. These experiments are related to person-to-image ratio, test-time augmentation, and model size.
- We create a clean, stripped-down implementation of a popular 3D human pose estimation model, and publish the code. We also perform a few experiments with the model hyperparameters and attempt to reduce the size of the model.
- We train and evaluate our overall pipeline on the popular Human3.6M dataset, to provide a baseline.
- We request a custom fitness dataset from the company Infinity AI, which is a modified version of their fitness pose dataset InfiniteForm [28] that is more applicable to our particular use-case. We then fine-tune our baseline model on this dataset, and evaluate the results on three main fitness poses (pushups, squats, and overhead presses). This also involves performing some preprocessing in order to adapt the codebase to run on a new dataset (e.g., we need to perform some skeleton joint conversions).
- We produce video demos displaying the results of our app in real-time, on footage that we take of ourselves.
- Finally, we also create a web app (accessible in the browser on web or mobile) that allows users to upload an image of themselves and receive personalized visualizations and feedback about their form.

2. Related Work

2.1. 3D Human Pose Estimation

3D human pose estimation is a popular topic in the computer vision literature. To get a broad understanding of the current landscape, survey papers such as Zheng et al. [32] and Gamra et al. [7] are very helpful.

Many current methods for 3D human pose estimation follow the same 2-step process: first, they detect 2D joint positions (keypoints) using a 2D pose detector; second, they use another model to predict the 3D joint positions from the 2D keypoints. Martinez et al. [21] use a feed-forward network to predict 3D poses from 2D keypoints. Zhao et al. [31] use a Graph Convolutional Network to do the same task. Chen et al. [3] do so using nearest neighbors. Li et al. [16] do this with a mixture density network.

Some other approaches skip the 2D step, and estimate 3D poses directly from the image. For example, Li and Chan were one of the first to adopt this approach [17]. Sun et al. [25] do this as well, using integral regression. Tekin et al. [27] make use of an overcomplete auto-encoder that learns a high-dimensional latent pose representation of the body joints.

Other approaches explicitly incorporate kinematic constraints into their models, which encode prior knowledge about the human body to make the models more robust. This type of knowledge includes skeletal joint connections, joint rotation properties, and bone-length ratios. Zhou et al. [33] and Kundu et al. [15] are among the groups that take this approach.

There are also numerous methods for images with multiple people, such as OpenPose [2]. However, we focus only on single-person scenes, which is the typical scenario we expect from people seeking fitness advice. Moreover, as we will explain later, we follow the 2-step approach outlined above, since it is the method that best fits our requirements.

2.2. Fitness Applications

There are a number of prior works that attempt to provide automated feedback about exercise form. Many methods, such as Chen et al. [4] and Jeon et al. [11], only use 2D pose estimation, which requires that the user position themselves in very specific orientations with respect to the camera. In contrast, our method can work for all body orientations. Fieraru et al. [6] use 3D pose estimation, and also create an associated dataset called Fit3D, although it has not been released yet. Some other 3D approaches come from Xie et al. [29], who use SMPL models, and Jiang et al. [12]. There are also many 3D methods that rely on external sensor data. For example, Seuter et al. [24] relies on IMU data, and Jin et al. [13] requires a Microsoft Kinect device. Although this additional sensor data can help improve accuracy, it is not very practical for average gym-goers, so we choose to focus on camera-only approaches.

3. Technical Approach

As mentioned earlier, for 3D human pose estimation, it is common to follow a 2-step approach: 1. Detect 2D keypoints with a 2D pose estimation model, then 2. Use a 3D pose estimation model to estimate 3D positions from those 2D keypoints. We follow this 2-step approach, which is summarized in Figure 2.

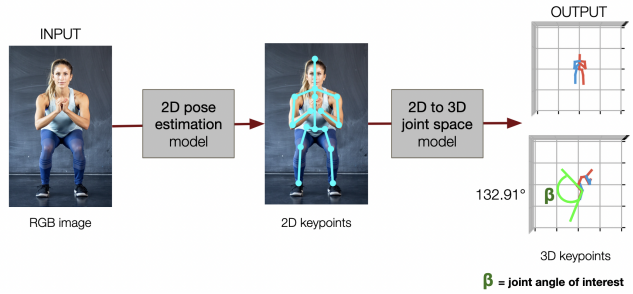


Figure 2. Summary of our overall approach. We take in an RGB image, detect 2D keypoints, use these to estimate 3D keypoints, and then extract the joint angle of interest.

3.1. 2D Pose Estimation Model

As a first step in creating a 3D pose estimation model, we need a 2D pose estimation model that identifies the positions of the joints in the image, in pixel space. We try two different 2D models: Stacked HourGlass [22] and EfficientPose IV [8]. Both models achieve strong performance on 2D pose estimation for a single person.

Stacked HourGlass uses a number of sequential "hourglass" modules, each of which downsamples to a lower resolution through the use of pooling, and then upsamples back to a higher resolution. This architecture allows the model to identify features at every scale, ranging from smaller details of the body to larger ones like the overall body shape.

EfficientPose IV makes use of EfficientNets [26] in order to produce an efficient and scalable solution for 3D human pose estimation. EfficientPose IV uses a multi-scale feature extractor, as well as efficient detection blocks (using the mobile inverted bottleneck convolutions that were originally developed for MobileNets [23]).

In our Experiments section, we examine and contrast both of these models to see how they compare. Ultimately, we choose Stacked HourGlass.

3.2. 3D Pose Estimation Model

For the 3D pose estimation model, we create a clean re-implementation of the approach by Martinez et al. [21]. Unlike the original implementation [20], which is in TensorFlow, we implement our version in PyTorch.

The main building block of the architecture is shown in Figure 3 below. Each block consists of a linear layer with 1024 hidden units, followed by BatchNorm, a ReLU activation, and Dropout with probability 0.5. This is repeated twice within each block. Note that the model also makes use of skip connections, as popularized by ResNets [9]. This building block is then repeated twice. At the end, there is a final linear layer (not shown in the diagram) which outputs the 3D keypoints.

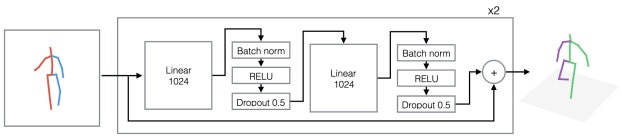


Figure 3. The main building block of the pose estimation model. The model takes in 2D keypoints on the left, passes them through a number of hidden layers, and outputs 3D keypoints on the right. (Image borrowed from original Martinez et al. paper).

Since we use Stacked HourGlass for our 2D detections, the input size to the 3D model is an array of size (16,2), since HourGlass outputs 16 joints, each of which has two 2D coordinates. The output size of the 3D model is (15,3), containing the 3D coordinates of 15 joints. We have 15 output joints instead of 16 because we predict *root-relative* coordinates, where the hip is considered to be fixed at (0,0,0). This allows the model to generalize better, since we only need to predict each joint's location *relative to the hip*, rather than having to predict the absolute location in 3D.

Following the advice of Martinez et al., we also constrain the weights of each layer to have a maximum norm of 1, which they say stabilizes training and improves generalization from the training to test set.

We use mean squared error (MSE) as our training loss metric. Thus, our training process learns the function f^* that minimizes the MSE over a dataset of N training examples, where $\mathbf{x}_i$ is the input 2D keypoints for image i and $\mathbf{y}_i$ is the ground-truth 3D keypoints for image i:

$$f^* = \min_f \frac{1}{N} \sum_{i=1}^{N} (f(\mathbf{x}_i) - \mathbf{y}_i)^2$$

We also experiment with SemGCN [31], another 3D human pose estimation model, by evaluating the official implementation on Human3.6M. Like the approach by Martinez et al., SemGCN also uses a two-step process, so we also use Stacked HourGlass as the 2D model before passing the keypoints to SemGCN. Unlike Martinez et al., SemGCN uses a Graph Convolutional Network (GCN) to predict the 3D keypoints. The GCN is a good approach for this, because the relationship between joints in the human body can be well-posed as a structured graph problem. We ultimately decide to re-implement the approach from Martinez et al. instead of SemGCN because it is faster, more streamlined, and architecturally simpler, characteristics that allow us to experiment more with it given our constrained resources. Additionally, on Human3.6M, we find that SemGCN achieves better performance than Martinez et al. by just 2 millimeters per joint (around 42mm vs. 44mm). This is a very small improvement, and we ultimately decide that it is not enough to justify the added complexity. However, SemGCN is still a very interesting approach that we would like to explore as future work.

3.3. Calculating joint angles

For our end-use application, we ultimately calculate joint angles to provide feedback to the user. Each angle is defined by three joints. For example, the squat angle is defined by the hip, knee, and ankle. We first calculate the two vectors that define the angle of interest; in this case, let $\mathbf{a} = \mathbf{p}_{hip} - \mathbf{p}_{knee}$ and $\mathbf{b} = \mathbf{p}_{ankle} - \mathbf{p}_{knee}$, where $\mathbf{p}_i$ is the 3D coordinates of joint i. We then calculate the joint angle β as follows:

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

and then convert from radians to degrees.

4. Experiments/Results

In this section, we state and discuss our experiments and results.

4.1. Datasets

In this project, we mainly use three 3D human pose estimation datasets. First, given its size and wide use, we train and evaluate the performance of our model with the Human3.6M dataset [10]. Secondly, given that we want to focus on fitness poses, we fine-tune the model using InfiniteForm [28]. This is a brand-new, synthetic dataset designed specifically for performing pose estimation on fitness-specific poses. Actually, due to some limitations of the original dataset for our use-case, we contacted the founders of the company that created InfiniteForm, and requested a customized dataset, which is what we use here. Finally, we test the system with videos of ourselves performing multiple fitness exercises. In Figure 4, we can observe an example for each of the mentioned datasets. We provide more information about each of these datasets below.

(a) Human3.6M (b) InfiniteForm (c) Personal

Figure 4. Sample images from three different datasets.

4.1.1 Human3.6M

The Human3.6M dataset [10] contains 3.6 million images from 11 different subjects (5 female, 6 male) performing 15 different scenarios. Human3.6M is one of the most popular 3D pose estimation benchmarks around. However, it is not specifically designed for fitness-related poses. We provide additional details about this dataset in Appendix A.

4.1.2 InfiniteForm (customized)

The original InfiniteForm dataset [28] contains $\sim 60K$ images with both single and multi-person scenes (up to 5 people) where each person is doing unique variations of 15 different fitness pose categories. We provide additional details about this dataset in Appendix A.

This dataset is brand-new, and consequently, no results have been published yet. In our original analysis of the dataset, we discovered the following:

- For more than 70% of the images, the person takes up less than 5% of the total image area.
- 9% of the images contain at least one joint that is not visible due to image cropping (e.g., the head of the person is outside the image).
- Many of the images contain multiple people, whereas we wish to focus on single-person images.

Given these findings, we decided to contact Infinity AI, the company that created InfiniteForm, and request a custom dataset from them that would be better-matched to our project. They kindly agreed, and sent us a new custom dataset of 4500 images. Each image contains a single person who is fairly large in the frame, and does not have any joints cropped out. The dataset contains three fitness poses: pushups, squats, and overhead presses. We use this custom dataset moving forward. A few example images are shown in Figure 5 below.

Figure 5. Example images from our custom version of InfiniteForm.

4.1.3 MPII

We also want to briefly mention the MPII dataset [1]. This dataset is for 2D human pose estimation, and although we do not use it directly, we do use it indirectly, since our pretrained Stacked HourGlass model was trained on MPII. MPII includes around 25K images containing over 40K people performing 410 human activities. We provide additional details about MPII in Appendix A.

4.2. Data Preprocessing

For this project, data preprocessing turned out to be much more complex than expected. Many of these issues

were due to our use of multiple datasets, which had different conventions for storing and manipulating the 3D data, forcing us to do various conversions. For example, to make our pipeline work on InfiniteForm as well as Human3.6M, we had to convert some of the skeletons (i.e., joint mappings) between the labeling style of COCO [18] (used by InfiniteForm) and the labeling style of Human3.6M, which are different from each other.

We also converted our 3D joint coordinates from the world frame to the camera frame. This is because it is unreasonable to expect the network to identify the 3D joint locations in an arbitrary world coordinate frame; a more reasonable choice of the world frame is to arbitrarily set it as the camera frame. This also causes the 2D-to-3D prediction step to be similar across different cameras, which prevents the model from overfitting to just a single world coordinate system. We do this by using the inverse of the camera extrinsics (rotation and translation) to convert from world to camera coordinates. The rotation matrix R contains the roll, pitch, and yaw of the camera frame with respect to the world frame, and the translation vector T is the translation vector between the two frames.

Lastly, we converted the 3D joint coordinates to be root-relative (relative to the hip, which is defined as (0,0,0)), for the reasons discussed earlier in this paper. We also normalized the images to have pixel values between [-1,1].

4.3. Evaluation metrics

To compare the different 2D pose estimation models, we use the Percentage of Correct Keypoints (PCK) metric as our performance indicator. Precisely, we compute PCK by considering a joint prediction as correct if the Euclidean distance between the predicted and the true joint is within a certain threshold. We set this threshold to be 50% of the width of the person's waist, in pixels, and call the resulting metric $PCK_{0.5}$.

To evaluate the performance of 3D human pose estimation, we can also use a 3D version of PCK; however, we choose instead to use Mean Per Joint Position Error (MPJPE), since MPJPE is the most widely used performance metric in existing literature. This measure calculates the Euclidean distance from the estimated 3D joints to the ground truth in millimeters, averaged over all joints in one image.

4.4. 2D Pose Estimation

4.4.1 Stacked HourGlass vs EfficientPose

To test which model is more suitable for our application, we downloaded pretrained versions of both models (pretrained on MPII), and evaluated their performance on 250 images from the original InfiniteForm dataset.

Furthermore, we tested two separate version of HourGlass, which have 2 and 8 stacked hourglass modules, respectively. In Table 1, we display the scores for the different models.

	PCK						
Model	*Shoulder*	*Elbow*	*Wrist*	*Hip*	*Knee*	*Ankle*	***Mean***
EP	0.17	0.16	0.19	0.16	0.15	0.16	0.16
SH2	0.32	0.25	0.32	0.29	0.26	0.28	0.29
SH8	0.38	0.33	0.39	0.39	0.36	0.35	0.37

Table 1. $PCK_{0.5}$ for Efficient Pose, Stacked HourGlass 2, and Stacked HourGlass 8

As we see in Table 1, Stacked HourGlass outperforms EfficientPose. Moreover, HourGlass with 8 stacked modules performs better than HourGlass with 2 stacked modules. Additionally, it is worth mentioning that the average inference time per image for HourGlass is 0.7 seconds, while for EfficientPose IV it is 8 seconds. However, we also tested EfficientPose RT Lite, which is the version for resource constrained applications, and the average prediction time was reduced to 0.4 seconds (although this also reduced the mean PCK score even lower, to 10%). The GPU used for all inferences was an NVIDIA Tesla T4.

Given these results, we believe that Efficient Pose IV might be overfitted to the MPII dataset, and fails to generalize to other datasets. In contrast, even though Stacked HourGlass was also trained on MPII, it generalizes much better to InfiniteForm. So, we chose Stacked HourGlass as our 2D Pose detection model.

Also note that the overall PCK metrics, even for Stacked HourGlass 8, are somewhat low. This is mainly due to the reasons we described earlier that led us to request a custom dataset from Infinity AI. On the new custom dataset, the average PCK is 0.513, which is noticeably better.

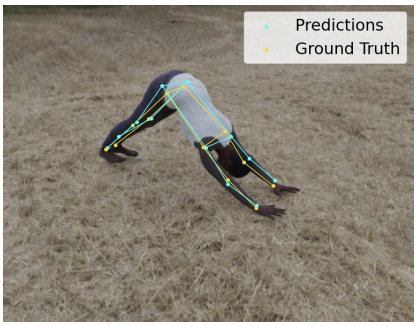


Figure 6. Example 2D Prediction on InfiniteForm, using Stacked HourGlass 8. Displays predicted pose (cyan) and ground truth pose (yellow).

4.4.2 Person/Image Ratio

Before requesting the custom InfiniteForm dataset, we observed that both Stacked HourGlass and EfficientPose perform better when the person is larger/more prominent in the image. Consequently, we further analyzed this observation

by calculating the ratio of the area of the person's bounding box to the total area of the image on 250 random images from InfiniteForm; the distribution is shown in Figure 7. Based on this histogram, we computed the average PCK for the following three bins: 0.05 to 0.075, 0.075 to 0.1, and 0.1 and up. Results are summarized in Table 2.

Model	*0.05 to 0.075*	*0.075 to 0.1*	*0.1 and up*
EP	0.098	0.09	0.285
SH2	0.167	0.247	0.443
SH8	0.206	0.301	0.589

Table 2. Model performance for different person to image ratios

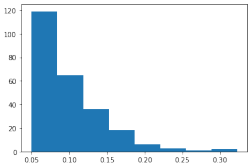


Figure 7. Person to image ratio distribution.

From Table 2, we can conclude that a larger person to image ratio improves performance for all three 2D prediction models. This observation contributed to our decision to contact Infinity AI to request a custom dataset; many of the people in the original InfiniteForm dataset were too small to work for our pipeline. With the custom dataset, which more closely mimics how close people would be to the camera in their exercise footage, we no longer suffer from this problem. The average person/image ratio in the new dataset is 0.195, which is much better for our purposes.

4.4.3 Test-Time Augmentation

Following a procedure mentioned in the original Newell et al. paper [22], we studied how test-time augmentation (by horizontally flipping each image to create a second view, and averaging the results) affects the performance on our dataset. The results of this study are summarized in Table 3.

	PCK						
Model	*Shoulder*	*Elbow*	*Wrist*	*Hip*	*Knee*	*Ankle*	***Mean***
SH2 (flip)	0.32	0.25	0.32	0.29	0.26	0.28	0.29
SH2 (no flip)	0.28	0.26	0.3	0.28	0.25	0.26	0.27
SH8 (flip)	0.38	0.33	0.39	0.39	0.36	0.35	0.37
SH8 (no flip)	0.36	0.31	0.35	0.37	0.35	0.37	0.35

Table 3. HourGlass Scores ($PCK_{0.5}$), with and without test-time augmentation.

As we can see in Table 3, test-time augmentation increases PCK score by around 2%.

4.5. 3D Pose Estimation

4.5.1 Baseline model trained on Human3.6M

We begin by training our model from scratch on Human3.6M. We train it for 200 epochs, with batch size 64, a learning rate of `1e-3`, and learning rate decay every 100000 training steps (with a gamma value of 0.96). We save the best checkpoints, according to MPJPE on the validation set, along the way.

When we evaluate the best checkpoint on the validation set, the final validation MPJPE is 43.46 mm.

4.5.2 Hyperparameter tuning

Next, we perform hyperparameter tuning to see if we can improve our training performance and speed on Human3.6M. Firstly, we double the batch size from 64 to 128, which leads to approximately a 33% speedup in training time.

Batch size	*MPJPE*
64	43.46 mm
128	43.70 mm

Table 4. Validation performance for different training batch sizes.

As we can see in Table 4, there is essentially no loss of performance with the higher batch size, so we choose a batch size of 128 moving forward.

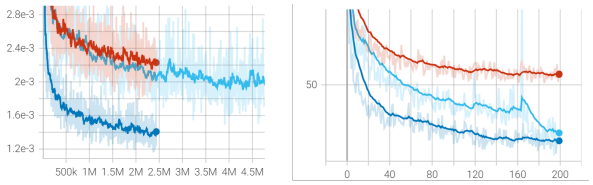
Next, we also attempt to reduce the size of the model by reducing the size of all the hidden layers from 1024 to 512.

Hidden layer size	*MPJPE*
1024	43.70 mm
512	49.81 mm

Table 5. Validation performance for different hidden layer sizes.

From the results in Table 5, we see that there is around a 6mm loss of accuracy associated with this change. This is a significant degradation in performance, so we choose to leave the hidden layer size at 1024.

The training and validation loss curves can be seen in Figure 8 below.



(a) Training Losses (b) Validation MPJPEs

Figure 8. Training and validation losses on Human3.6M over 200 epochs of training. Red = batch size 128, hidden layer size 512. Light blue = batch size 64, hidden layer size 1024. Dark blue = batch size 128, hidden layer size 1024.

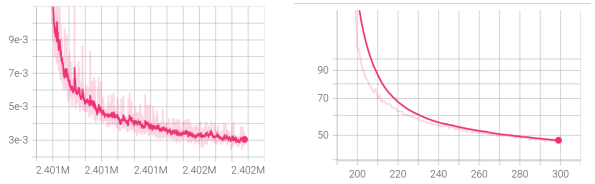
From Figure 8, it is interesting to observe that the training loss (MSE) for a larger batch size (dark blue) is lower

than with a smaller batch size (light blue), even though the validation loss (MPJPE) reaches almost the same value after 200 epochs. We believe that this difference in the training loss is not significant as it is around `0.6e-3`, and what really matters is the validation loss MPJPE value. It is worth mentioning that the curve for the model with a smaller batch size (light blue) runs twice the iterations given that it was run for the same number of epochs but with a batch size half the size of the other two models in the plot.

4.5.3 Fine-tuning on custom InfiniteForm

Next, we fine-tuned the model on the custom InfiniteForm dataset. Note that there are some differences in the 2D joints outputted by Stacked HourGlass vs. the 2D joints provided by InfiniteForm. Therefore, rather than training the 2D-to-3D model on the ground-truth 2D labels from InfiniteForm, we first passed the images through Stacked HourGlass and used the predicted detections as the 2D labels. We kept all images that had a PCK of at least 0.6 so that we wouldn't train on completely incorrect data. Since we trained on slightly imperfect 2D labels, it's likely that our results below would be even better if we had been able to train on the ground-truth directly. Given more time, we would request that feature to improve our results further.

We fine-tuned on InfiniteForm for 100 epochs. The training and validation loss curves are shown in Figure 9 below. The best model achieves a MPJPE value of 47.87mm, which is quite similar to the value we achieved on Human3.6M.



(a) Training Loss (b) Validation MPJPE

Figure 9. Training and validation losses on InfiniteForm over 100 epochs of fine-tuning.

We also broke down the performance across the three exercises in the dataset: pushups, squats, and overhead presses. The results are shown in Figure 6.

Exercise	# samples	*MPJPE*
overhead press	101	40.33 mm
pushup	56	61.13 mm
squat	94	48.07 mm
(all)	251	47.87 mm

Table 6. Validation performance, by exercise.

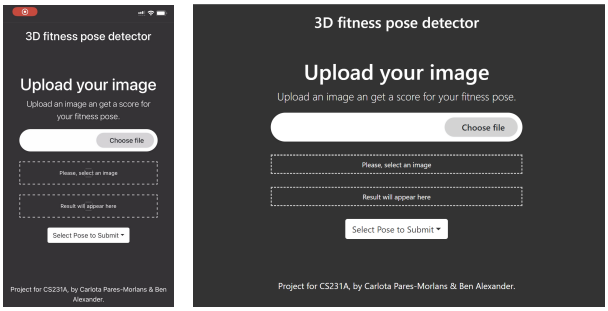
From the results presented in Table 6, it is interesting to note how the model mainly lags behind on pushups; we believe this is because pushups involve the person lying horizontally on the floor, whereas most of the training data (from the rest of InfiniteForm, and also Human3.6M) has people in an upright position. So, it makes sense that performance on pushups would be worse. This is another reminder that the training data can greatly impact the final model performance.

4.5.4 Personal dataset + demo video

Finally, we also evaluated the performance of our fine-tuned model on some of our own footage, and created some demos. Overall, the performance is quite good, although it is far from perfect, especially due to the fairly small size of the training dataset. Please see the demo video in the README of our GitHub repo at `https://github.com/carlotapares/CS231A_FinalProject`. In order to generate this demo, we recorded some footage of ourselves doing pushups and passed this through the model pipeline.

4.5.5 Web app

Finally, we also implemented a downstream web app that works in the browser, both on web and mobile (see Figure 10). The user uploads an image and chooses the exercise they are doing; we then pass this to the backend, and return visualizations of the predicted 3D pose and some feedback about the joint angle of interest. In the future, we would expand this to include video footage as well, similar to our demo above. Please contact us if you would like to test the app, and we will send you the URL since the server is currently turned off.



(a) Mobile Interface (b) Desktop Interface

Figure 10. Mobile and desktop app interfaces. Allows users to upload a photo of themselves and choose which exercise they are doing.

After uploading an image, the app returns feedback such as the one we can find in Figure 11. It is important to note that even though the person's position with respect to the

camera is not ideal for the angle of interest to compute, the pipeline still does a good job.

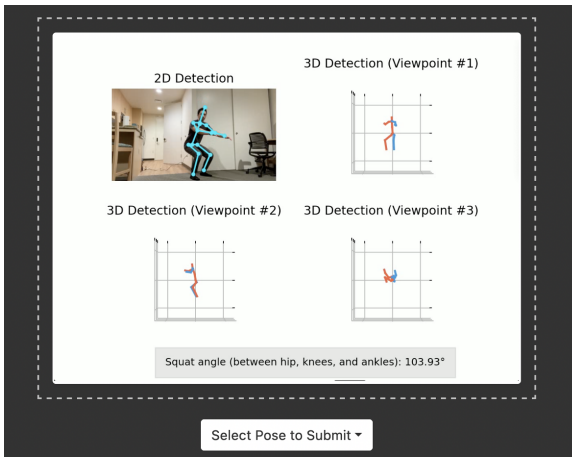


Figure 11. Output from app, showing the 2D detection, 3D detection from three viewpoints, and measurement of the joint angle of interest.

5. Conclusions and Future Work

In this paper, we explore 3D human pose estimation as a potential solution for providing automated, low-cost, personalized exercise feedback. We re-implement a high-performing 3D pose estimation model, train it on Human3.6M, and then fine-tune it on a custom version of a brand-new fitness pose dataset called InfiniteForm. We find that the model performs very well on this dataset, suggesting that this is indeed a promising direction to go in. Additionally, we produce a web app that allows users to upload photos and get personalized feedback, as an initial prototype of what our personal trainer app could eventually look like.

In the future, we would like to expand our model to more exercises by training on images of additional exercise poses. This also applies to the app; with more exercises, we would be able to program more options for the user to choose their joint angle of interest. We would also experiment with additional 2D-to-3D pose estimation models, to compare them to our current approach. We would like to also further evaluate the impact of training on synthetic vs. non-synthetic data, since InfiniteForm is a synthetic dataset and Human3.6M is non-synthetic.

Code Availability

Our code is available at the following repo: https://github.com/carlotapares/CS231A_FinalProject.

References

[1] M. Andriluka, L. Pishchulin, P. Gehler, and B. Schiele. 2d human pose estimation: New benchmark and state of the art analysis. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2014.

[2] Z. Cao, G. Hidalgo, T. Simon, S.-E. Wei, and Y. Sheikh. Openpose: realtime multi-person 2d pose estimation using part affinity fields. *IEEE transactions on pattern analysis and machine intelligence*, 43(1):172–186, 2019.

[3] C.-H. Chen and D. Ramanan. 3d human pose estimation= 2d pose estimation+ matching. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 7035–7043, 2017.

[4] S. Chen and R. R. Yang. Pose trainer: correcting exercise posture using pose estimation. *arXiv preprint arXiv:2006.11718*, 2020.

[5] Y. Chen, Y. Tian, and M. He. Monocular human pose estimation: A survey of deep learning-based methods. *Computer Vision and Image Understanding*, 192:102897, 2020.

[6] M. Fieraru, M. Zanfir, S. C. Pirlea, V. Olaru, and C. Sminchisescu. Aifit: Automatic 3d human-interpretable feedback models for fitness training. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9919–9928, 2021.

[7] M. B. Gamra and M. A. Akhloufi. A review of deep learning techniques for 2d and 3d human pose estimation. *Image and Vision Computing*, 114:104282, 2021.

[8] D. Groos, H. Ramampiaro, and E. A. Ihlen. Efficientpose: Scalable single-person pose estimation. *Applied intelligence*, 51(4):2518–2533, 2021.

[9] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[10] C. Ionescu, D. Papava, V. Olaru, and C. Sminchisescu. Human3.6m: Large scale datasets and predictive methods for 3d human sensing in natural environments. *IEEE transactions on pattern analysis and machine intelligence*, 36(7):1325–1339, 2013.

[11] H. Jeon, Y. Yoon, and D. Kim. Lightweight 2d human pose estimation for fitness coaching system. In *2021 36th International Technical Conference on Circuits/Systems, Computers and Communications (ITC-CSCC)*, pages 1–4. IEEE, 2021.

[12] Y. Jiang, C. Cao, X. Zhu, Y. Ma, and Q. Cao. Rgbd-based real-time 3d human pose estimation for fitness

assessment. In *2020 3rd World Conference on Mechanical Engineering and Intelligent Manufacturing (WCMEIM)*, pages 103–108. IEEE, 2020.

[13] X. Jin, Y. Yao, Q. Jiang, X. Huang, J. Zhang, X. Zhang, and K. Zhang. Virtual personal trainer via the kinect sensor. In *2015 IEEE 16th international conference on communication technology (ICCT)*, pages 460–463. IEEE, 2015.

[14] H. Kaur, T. Singh, Y. K. Arya, and S. Mittal. Physical fitness and exercise during the covid-19 pandemic: a qualitative enquiry. *Frontiers in psychology*, 11:2943, 2020.

[15] J. N. Kundu, S. Seth, M. Rahul, M. Rakesh, V. B. Radhakrishnan, and A. Chakraborty. Kinematic-structure-preserved representation for unsupervised 3d human pose estimation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 11312–11319, 2020.

[16] C. Li and G. H. Lee. Generating multiple hypotheses for 3d human pose estimation with mixture density network. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9887–9895, 2019.

[17] S. Li and A. B. Chan. 3d human pose estimation from monocular images with deep convolutional neural network. In *Asian Conference on Computer Vision*, pages 332–347. Springer, 2014.

[18] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick. Microsoft coco: Common objects in context. In *European conference on computer vision*, pages 740–755. Springer, 2014.

[19] W. Liu, Q. Bao, Y. Sun, and T. Mei. Recent advances in monocular 2d and 3d human pose estimation: A deep learning perspective. *arXiv preprint arXiv:2104.11536*, 2021.

[20] J. Martinez, R. Hossain, J. Romero, and J. J. Little. Github, Martinez et al.: una-dinosauria/3d-pose-baseline. `https://github.com/una-dinosauria/3d-pose-baseline/`. Accessed: 2022-03-02.

[21] J. Martinez, R. Hossain, J. Romero, and J. J. Little. A simple yet effective baseline for 3d human pose estimation. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2640–2649, 2017.

[22] A. Newell, K. Yang, and J. Deng. Stacked hourglass networks for human pose estimation. In *European conference on computer vision*, pages 483–499. Springer, 2016.

[23] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen. Mobilenetv2: Inverted residuals and linear bottlenecks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4510–4520, 2018.

[24] M. Seuter, L. Opitz, G. Bauer, and D. Hochmann. Live-feedback from the imus: Animated 3d visualization for everyday-exercising. In *Proceedings of the 2016 ACM International Joint Conference on Pervasive and Ubiquitous Computing: Adjunct*, pages 904–907, 2016.

[25] X. Sun, B. Xiao, F. Wei, S. Liang, and Y. Wei. Integral human pose regression. In *Proceedings of the European conference on computer vision (ECCV)*, pages 529–545, 2018.

[26] M. Tan and Q. Le. Efficientnet: Rethinking model scaling for convolutional neural networks. In *International conference on machine learning*, pages 6105–6114. PMLR, 2019.

[27] B. Tekin, I. Katircioglu, M. Salzmann, V. Lepetit, and P. Fua. Structured prediction of 3d human pose with deep neural networks. *arXiv preprint arXiv:1605.05180*, 2016.

[28] A. Weitz, L. Colucci, S. Primas, and B. Bent. Infiniteform: A synthetic, minimal bias dataset for fitness applications. *arXiv preprint arXiv:2110.01330*, 2021.

[29] H. Xie, A. Watatani, and K. Miyata. Visual feedback for core training with 3d human shape and pose. In *2019 Nicograph International (NicoInt)*, pages 49–56. IEEE, 2019.

[30] L. Yang, Y. Li, D. Zeng, and D. Wang. Human exercise posture analysis based on pose estimation. In *2021 IEEE 5th Advanced Information Technology, Electronic and Automation Control Conference (IAEAC)*, volume 5, pages 1715–1719. IEEE, 2021.

[31] L. Zhao, X. Peng, Y. Tian, M. Kapadia, and D. N. Metaxas. Semantic graph convolutional networks for 3d human pose regression. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3425–3435, 2019.

[32] C. Zheng, W. Wu, T. Yang, S. Zhu, C. Chen, R. Liu, J. Shen, N. Kehtarnavaz, and M. Shah. Deep learning-based human pose estimation: A survey. *arXiv preprint arXiv:2012.13392*, 2020.

[33] X. Zhou, X. Sun, W. Zhang, S. Liang, and Y. Wei. Deep kinematic pose regression. In *European Conference on Computer Vision*, pages 186–201. Springer, 2016.

A. Appendix

A.1. Additional dataset details

Here we provide more details about each dataset.

A.1.1 Human3.6M

The Human3.6M dataset [10] contains 3.6 million images from 11 different subjects (5 female, 6 male) performing 15 different scenarios (Directions, Discussion, Eating, Activities while seated, Greeting, Taking photo, Posing, Making purchases, Smoking, Waiting, Walking, Sitting on a chair, Talking on the phone, Walking dog, Walking together). For each of the images, they provide the joint positions in 3D space as well as their projected 2D position in the image plane. Specifically, the list of the 17 joints they provide is the following: Pelvis, RHip, RKnee, RAnkle, LHip, LKnee, LAnkle, Spine, Neck, Head, Site, LShoulder, LElbow, LWrist, RShoulder, RElbow, RWrist.

A.1.2 InfiniteForm

The original InfiniteForm dataset [28] contains $\sim 60K$ images with both single and multi-person scenes (up to 5 people) where each person is doing unique variations of 15 different fitness pose categories (boatpose, curls, downdog, hipbridge, jumpingjacks, legraise, lunge, plank, pushups, sideplank, situp, squats, tree, updog, warrior). For each of the images, they provide the 2D position of the joints and a depth map. Specifically, the list of the 17 joints they provide is the following: Nose, LEye, REye, LEar, REar, LShoulder, RShoulder, LElbow, RElbow, LWrist, RWrist, LHip, RHip, LKnee, RKnee, LAnkle, RAnkle.

A.1.3 MPII

The MPII dataset [1] is for 2D human pose estimation, and although we do not use it directly, we do use it indirectly, since our pretrained Stacked HourGlass model was trained on MPII. MPII includes around 25K images containing over 40K people performing 410 human activities. 2D annotations of the following 16 joints are provided: RAnkle, RKnee, RHip, LHip, LKnee, LAnkle, Pelvis, Spine, Neck, HeadTop, RWrist, RElbow, RShoulder, LShoulder, LElbow, LWrist.

A.2. Baseline vs Fine-tuned model results

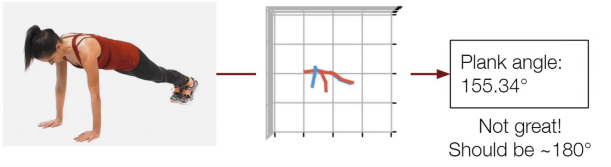


Figure 12. Baseline model plank prediction

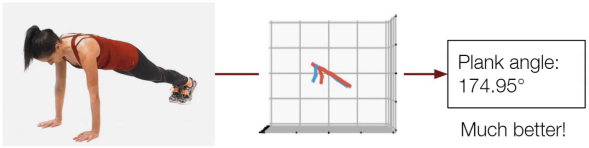


Figure 13. Fine-tuned model plank prediction