Learning Proper Pose: Three Techniques for 3D Pose Correction

Alec Flowers (321786), Alexander Glavackij (322968), Nina Mogan Mainusch (323160)
CS-503 Final Project Report

***Abstract*—Understanding the multi-modal, high-dimensional distribution of body positions in 3D space is an important question. In many human activities, such as sports, dance, and physical therapy, there are correct and safe ways to orient one's body and to carry out movement patterns. In this paper, we develop a pipeline that can recognize poses, learn a distribution of correct poses and then identify and correct anomalies (incorrect poses). We developed three methodologies to identify, and correct anomalies learned from the data. One can imagine far further reaching applications: identifying unhealthy movement patterns in patients, correcting robot movements, and providing feedback for at-home followers of online fitness training. The code is publicly available at https://gitlab.com/aglavac/cs-503-visual-intelligence.**

I. Introduction

In recent years, much research has been done on pose estimation. State of the art pose estimators are generally able to classify and track landmarks in pictures and videos over multiple frames [1], [2]. On the contrary, only some researchers have taken it further and have tried to add semantic meaning to estimated poses. What makes a pose a proper pose? What are the differences between good and bad poses? Technique and correct poses are essential to many sports, e.g., yoga, golf, or weightlifting. Previous work, such as the work done by [3] and [4], defines hard-coded features (e.g. elbow angle) to identify incorrect poses. However, these approaches require humans to define these features manually.

Our main contribution is a system that automatically detects features of an incorrect pose and offers corrections to transform incorrect poses into correct ones. There are distinct challenges involved in this:

- The system learns which features render a pose incorrect. It uses a data-driven automated learning approach that considers differences in body size, positioning, and pose variations when extracting features.
- The system indicates which aspects of the poses need to be corrected and how.

We focus on Yoga poses as a first application since they follow well-defined rules and are usually executed in a controlled environment. However, one can imagine far further applications, such as identifying unhealthy movement patterns in patients or correcting robot movements.

To develop our proposed system, we pursued the following steps. First, we created a dataset consisting of 669 images showing three different yoga poses: Downward Dog (DD), Warrior I (W1), Warrior II (W2). Our system takes these images as inputs and runs a publicly available pose estimation to determine the key points. Afterward, it classifies the pose as one of the poses mentioned above and determines whether the pose is executed correctly or not. In case of an incorrect pose, we can employ three different approaches to correct the pose, which we explain in more detail in Chapter III.

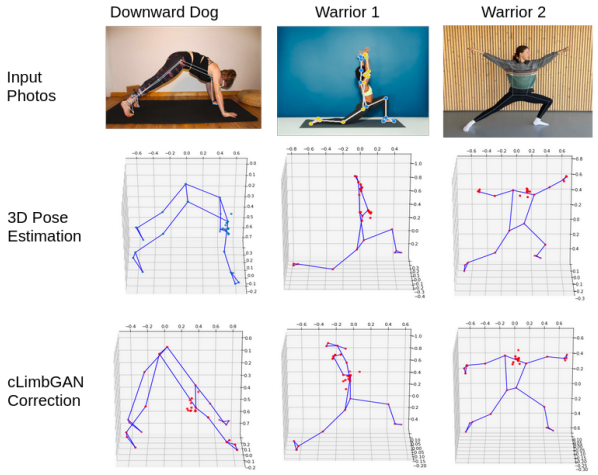


Figure 1: Our cLimbGAN approach corrected three incorrectly performed yoga poses. For the Downward Dog, the person's back and arms were not aligned; for Warrior 1, her knee was touching the ground, and for Warrior 2, her arms were too high and her front knee too far forward. All these errors are corrected in the pose generated by the cLimbGAN.

Our system can correctly classify a yoga pose and its correctness with test accuracies of 95% and 83% respectively. While struggling with minor difficulties, it corrects major pose errors reliably, cf. Figure 1.

II. Related Work

There exist prior work on pose correction, which focuses on rule-based corrections. In [3], [4], and [5] the authors design a pose correction system using predefined rules, such as body angles between landmarks, to classify poses as correct or improper. However, their approaches rely on humans to define these features.

Image transfer methods, like [6], [7], [8] and [9] take source images and poses and transform them into a target pose. The authors use poses obtained from a source with off-the-shelf human pose detectors, such as OpenPose, as an

intermediate representation. Generally, a generative neural network generates a target image based on the source pose estimation and the amateur target image. These approaches can transform poses, but they lack the understanding of correct poses.

In [10], the authors correct the execution of gym exercises using a predictive technique. While a subject is performing an exercise, they use a predictor trained on correctly performed movements to predict the correct next position. This prediction is then compared to the observed movement, and corrections are given. While resembling our approach in being data-driven, we do not rely on multiple frames of the same person and pose; instead, our system works with a single input image.

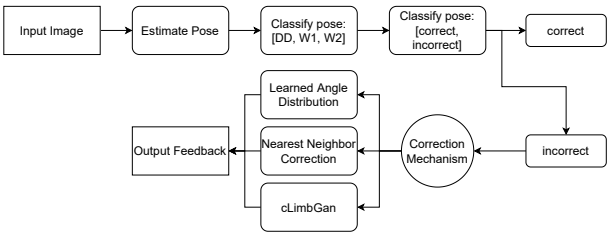


Figure 2: The pose correction pipeline implemented by our system.

III. Method

The pipeline we developed, cf. Figure 2, consists of four distinct stages: Pose Estimation, Pose Classification, Pose Correction, and Pose Feedback. To develop the pipeline, we also had to collect training data. This chapter is structured as follows: First, we present our data collection methods. Afterward, we present the five stages of the pipeline in order.

A. Data Collection

In order to develop our pipeline, we created a dataset of 669 images of three yoga poses. One of our reasons to focus on yoga poses as a first application was the availability of a dataset. According to the creators, it contains 1175 images for the three poses we focused on. However, when downloading the images, we ran into problems: out of these 1175 image links, 438 were broken or unavailable anymore. A large part of the remaining images was poor quality: icons, drawings, or even stick figures instead of an actual human performing the pose. After cleaning the dataset, we were left with 414 usable images. We decided to scrape Google images for additional pictures to increase the number of samples. We scraped the images appearing under the following keywords: "warrior 1 yoga", "warrior 2 yoga", and "downward dog yoga". Afterward, we manually labeled the images to yield our final dataset, cf. Table I.

B. Pose Estimation

Our focus laying on correcting a pose, we chose to use an existing pose estimation library over implementing the pose estimation ourselves. The chosen library should be accessible, as accurate as possible, and, most importantly, do 3D pose estimation. The estimated poses need to be in 3D, since to correct poses, we are interested in angles in 3D, and depending on the perspective, 2D angles can change even though the pose does not. There exist several open-source pose estimation libraries, such as OpenPose [11] and PifPaf [12]. However, most pose estimators do not provide 3D poses. In the end, we chose MediaPipe, a state-of-the-art pose estimator developed by Google [13], which works on the majority of modern devices and provides 33 3D landmarks, cf. Figure 3.

	Train set		Validation set	
	Good	Bad	Good	Bad
Downward Dog	135	62	33	15
Warrior 1	69	73	17	17
Warrior 2	119	73	30	17

Table I: Number of samples in the training and validation sets we created.

C. Pose Classification

Our pipeline requires classifying the pose type and pose quality for the subsequent stages. A natural, data-driven choice to perform this classification is a neural network. We trained two MLP classifiers for these two tasks, using the 33 landmarks returned by MediaPipe as input to both classifiers. Table II shows the architecture of both neural nets, which is described in more detail in the Appendix.

Layer	Output Shape
Input	33 * 3
Flatten	99
Dense	64, 32
ReLU	64, 32
Batch Norm.	64, 32
Dense	3

(a) Pose type classifier

Layer	Output Shape
Input	33 * 3
Flatten	99
Dense	64, 32, 16
Leaky ReLU	64, 32, 16
Batch Norma.	64, 32, 16
Dense	1

(b) Pose quality classifier

Table II: Network architecture of the pose type and pose quality classifiers. The notation 64, 32, 16 indicates 3 layers of the given dimension arranged sequentially.

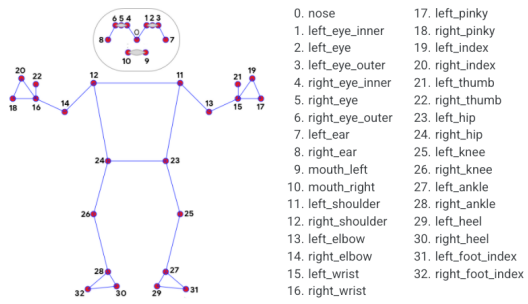
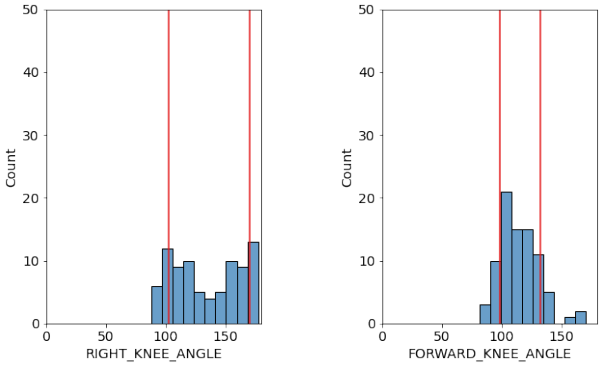


Figure 3: The 33 landmarks provided by MediaPipe [13].



(a) before foot forward correction (b) after correction

Figure 4: Warrior 1 right knee angle distributions of the correct data set before and after the foot forward correction were applied with a 70% CI (red vertical lines). The right knee angle is defined by the key points Right Hip, Right Knee, and Right Ankle.

D. Pose Correction

1) Learned Angle Distribution: As a first approach, we explicitly learn the distribution of a set of angles from the correct poses dataset. Our assumption here is that the body angles of correct poses fall into a specific range. An incorrect pose can be corrected by finding body angles that fall outside that range. We define a set of important body angles between the MediaPipe keypoints, see Appendix VII: Figure IV, and calculate the angle between the two $3D$ vectors formed by three keypoints k_1, k_2, and k_3:

$$\mathbf{v_1} = k_1 - k_2$$
$$\mathbf{v_2} = k_3 - k_2$$

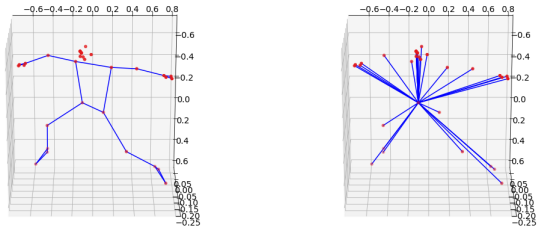
$$x = \|\mathbf{v_1} \times \mathbf{v_2}\|_2^2 = \|\mathbf{v_1}\| \cdot \|\mathbf{v_2}\| \cdot \sin\theta \cdot \|n\|_2^{2*} \quad (1)$$
$$y = \mathbf{v_1} \cdot \mathbf{v_2} = \|\mathbf{v_1}\| \cdot \|\mathbf{v_2}\| \cdot \cos\theta \quad (2)$$

$$\angle \mathbf{v_1 v_2} = |\arctan\frac{x}{y}| = |\arctan\frac{\|\mathbf{v_1}\| \cdot \|\mathbf{v_2}\| \cdot \sin\theta}{\|\mathbf{v_1}\| \cdot \|\mathbf{v_2}\| \cdot \cos\theta}|$$
$$= |\arctan\frac{\sin\theta}{\cos\theta}| = |\theta|. \quad (3)$$

This yields angles ranging between 0° and 180°. We apply the simplifying assumption that most of our bodies' angles operate between 0° and 180° degrees. We then define confidence intervals (CIs) which determine the acceptable range of body angles. Figure 4a shows the angle distribution of the right knee for correct poses and the corresponding CI.

With the distinction between left and right, we would see a multi-modal distribution in certain angles, as shown in Figure 4a. The chirality of the W1 and W2 poses causes the multi-modality: body angles change between poses with the left and poses with the right foot forward, even though they have the same pose. To remove the chirality, we calculate which foot is the forward foot. Instead of making a distinction between right and left, we use front and back. We calculate the front- and backside of the pose by forming a plane with the key points Left Hip, Right Hip, and Right Shoulder. This plane cuts the body into two halves. We then determine the front- and backside of the pose by assigning each keypoint to one half. As can be seen in Fig 4b this fixes the bi-modal distribution.

*n is a unit vector



(a) Stick Figure Representation (b) Vector Representation

Figure 5: We convert the stick figure pose representation into a vector representation to calculate the transformation from incorrect to correct pose.

The *Learned Angle Distribution* approach relies heavily on feature engineering: e.g., which body angles to define and calculate, what confidence interval to use, and how to calculate the forward body side. To generalize this approach to a new domain would be troublesome and require re-engineering to solve new domain-specific problems. The following correction mechanism we designed to improve upon this first attempt.

2) Nearest Neighbor 3D Pose: Here we take an incorrect pose, find the closest pose among the correct poses, i.e., nearest neighbor, and use it to correct to incorrect pose. To find the closest pose, we calculate the sum of the euclidean distance between all key points and select the image which minimizes this distance d:

$$\mathcal{G} = \text{set of images in good pose dataset}$$
$$i = \text{input image keypoints}$$
$$d = \underset{g \in \mathcal{G}}{\text{argmin}} \sum_{k \in keypoints} \|i_k - g_k\|_2^2$$

We expect the nearest correct neighboring pose to have characteristics similar to the incorrect pose, i.e., they are facing the same way and have the same foot forward. As the number of training images increases, the space of correct poses becomes denser; thus, finding a neighbor that exhibits these characteristics becomes easier.

We transfer the incorrect pose into a correct pose by applying the body angles from the correct pose to the incorrect pose. Applying only body angles personalizes the correction to the incorrect pose since the proportions (limb lengths) are not changed. We do this transformation in the

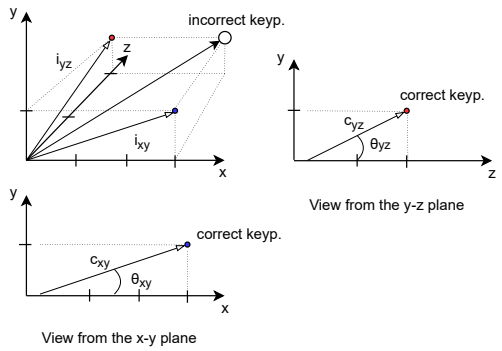


Figure 6: Illustration of the vector and angle calculations. θ_{xy} and θ_{yz} refer to the "correct" body angles. i_{x-y}, i_{y-z} refer to the projections of the incorrect keypoint's vector, c_{x-y} and c_{y-z} to the projections of the correct keypoint's vector.

following way: First, we normalize the right hip of the incorrect pose to be at point $(0, 0, 0)$. Every other keypoint can be represented as a vector emanating from the right hip. This vector representation of a pose is an alternative representation to the stick figure representation returned by pose estimators, cf. Fig 5. The vector representation simplifies the manipulation of the incorrect pose.

We do the following steps to correct a key point of an incorrect pose. First, we project the keypoint's vector into the x-y plane, and in the y-z plane, i_{x-y} and i_{y-z}. We do the same to the vector of the same key point in the nearest neighbor and get c_{x-y} and c_{y-z}. Then, we calculate the "correct" body angles θ_{xy} and θ_{yz} between c_{x-y}, c_{y-z} and the x-y- and y-z planes. With i_{x-y}, i_{y-z}, θ_{xy} and θ_{yz} we can recompute the location of the keypoint:

$$x = \|i_{x-y}\| * \cos\theta_{xy}$$
$$y = \|i_{x-y}\| * \sin\theta_{xy}$$
$$z = \|i_{y-z}\| * \sin\theta_{yz}.$$

The resulting vector has the same length as the incorrect vector, but the angles are taken from the nearest neighbor vector. While this simplification adjusts the angles appropriately, it does not keep the limb lengths constant.

3) cLimbGAN: Generative Adversarial Networks (GANs) employ the powerful technique of adversarial training, where a generative model is learning the data distribution in order to generate data indistinguishable from real data [14]. In the case of a multi-modal setting, we condition the generator on additional information such as class labels, thereby directing the data generation process [15]. We employ a conditional instead of manufacturing the correction mechanism and manually maneuvering angles in $3D$ space. We condition the GAN on specific aspects of incorrect poses and generate corresponding correct samples from its latent distribution. We train the GAN exclusively on images from the correct pose domain to learn the latent distribution for correct poses.

Our conditional GAN $cLimbGAN$ receives a noise vector, the pose type of the incorrect pose, and the limb lengths of the incorrect pose. It generates a correct pose of the same type and with the same limb lengths, i.e., proportions, cf. Figure 7. Further information regarding the network architecture can be found in the Appendix.

IV. Results

A. Pose Classification

The classifier successfully manages to differentiate between the different poses, even if they appear to be similar to each other like W1 and W2. It achieves a training accuracy of 99% and a validation accuracy of 93%. The confusion matrices for the classifier results can be found in the Appendix in Figure 8 and Figure 9.

The pose quality classifier achieves a training accuracy of 92% and a validation accuracy of 78%. It manages to classify a good pose as good with high success, but it struggles to detect a bad pose as such (cf. Figure 10 and Figure 11 in the Appendix for details). Visually inspecting the misclassified images, we find that this is mainly due to noisy pose estimations.

B. Pose Correction

One commonality to all our approaches is that they are based on the same 3D keypoints obtained from the pose estimation. When the pose estimation was noisy or malfunctioning, our techniques' quality suffered. All three techniques are relatively successful in correcting significant problems in poses but struggle to deal with more minor pose corrections.

1) Learned Angle Distribution: This approach captures major problems with poses; however, it is susceptible to noisy pose estimation for feet and hands. If an angle falls outside of a CI, this approach outputs the incorrect angle's difference to the CI. Set to cover 70% of the good datapoints, the CI averaged over all the angles covers for DD 18.4 ± 4.8 degrees, W1 23.7 ± 5.7 degrees, and W2 16.9 ± 5.1 degrees. These are wide ranges and can be attributed to the noise from the pose estimation. Looking at the DD correction from Fig 7 the text output does capture the significant problems with the pose. It identifies that the armpit angles need to be more straight and that the hip angle needs to be smaller. However, for W1 cf. Figure 7 middle column, the most significant problem it finds is the backward wrist angle which is coming from an inaccuracy in the wrist keypoint estimation.

2) Nearest Neighbor 3D Pose: This approach works well if it finds a neighboring correct pose similar to the incorrect pose. The results of this method are output in the form of 3D keypoints, which are plotted with limb connections on a 3D grid. In Fig 7 we can see that the nearest neighbor is comparable to the cLimbGAN in correcting the errors that the incorrect poses are showing. However, noise from the pose estimation appears in both the incorrect pose and

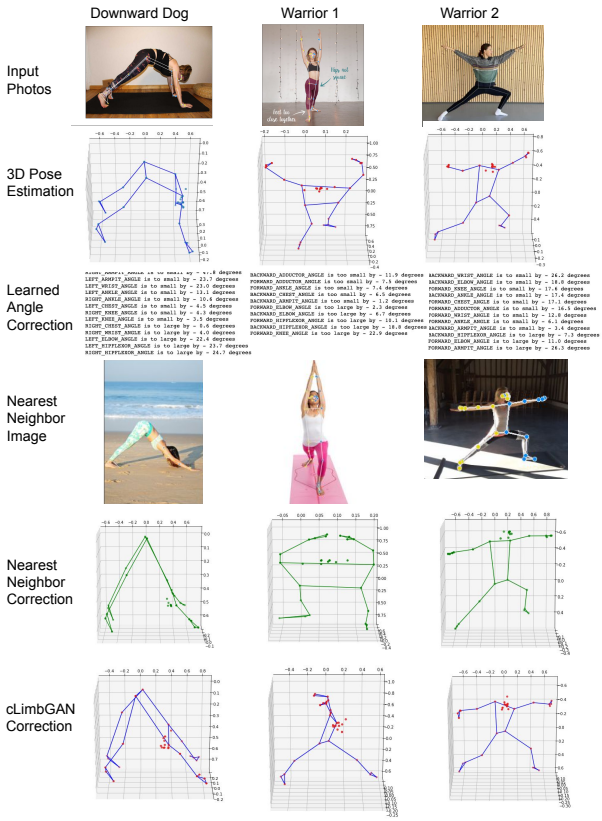


Figure 7: The output of all 3 of our correction mechanisms on an incorrect image for each class DD, W1 and W2.

nearest neighbor pose. When combined, these errors can amplify. This method fails when the pose to correct is different from images in the correct dataset. For example, in Fig 7 this approach turn the person's legs by 180° because it did not find a neighboring pose with the correct foot forward.

3) cLimbGAN: The $cLimbGAN$ for DD and W2 has learned the important features of each good pose and corrects the users accordingly, cf. Figure 7. The cLimbGAN will output a correction with the same foot forward and facing the same direction as the user. The only information given to the model is the limb lengths, but that is enough for it to infer these characteristics. However, the fingers and the feet of the generated image often look distorted. This comes from the pose estimator, which struggles to estimate these keypoints in the training dataset accurately. Comparable to the nearest neighbor correction, when an input image has characteristics not represented in the training dataset, the generated image has some semblance of form but is not helpful for correction. For example, in Figure 7 for W1, the input image is facing forward, and the cLimbGAN output does not represent a W1 pose. With more training data and more accurate pose estimation, these problems can be ameliorated.

V. Conclusion

Our contribution is implementing an entire pipeline ranging from crafting a dataset to correcting a detected incorrect pose in three different data-driven ways.

All of our approaches rely on the quality of the pose estimates of the correct dataset. The labels of the correct and incorrect poses were determined by the authors, who have a combined yoga experience of 15 years. As we use the correct dataset to build a distribution of what determines a correct pose, we recognize that we are building this distribution based on what we decided was a correct pose. The inherent bias that we may have added through this labeling process can be corrected or adjusted by enlisting the help of a panel of yoga teachers who could refine the datasets. More importantly, these methodologies can be applied to any pose dataset given labeling of correct and incorrect.

The size of our dataset and the mediocre quality of the 3D pose estimates were two limitations. The first can be remedied by scraping or producing more data. The second can be tackled by ensembling $3D$ pose estimators or trying new state-of-the-art methods.

A possible extension is to expand our system to video data. Regarding the $cLimbGAN$, we imagine not only providing the corrected stick figure to the user but synthesizing the user into the corrected pose as in [9].

VI. Individual Contributions

Alec labeled initial data, worked on the learned angle distribution with Alex, and built the nearest neighbor 3D pose correction.

Alex worked on the learned angle distribution with Alec, implemented our correct or not MLP classifier and worked with Nina on the cLimbGAN.

Nina built our data scraping tool, implemented our pose type MLP classifier, and worked with Alex on the cLimbGAN.

All authors took an active and equal part in brainstorming ideas, and crafting the presentation and this report.

References

[1] A. Toshev and C. Szegedy, "Deeppose: Human pose estimation via deep neural networks," pp. 1653–1660, 2014.

[2] A. Newell, K. Yang, and J. Deng, "Stacked hourglass networks for human pose estimation," *CoRR*, vol. abs/1603.06937, 2016. [Online]. Available: http://arxiv.org/abs/1603.06937

[3] C. Hua-Tsung, H. Yu-Zhen, and H. Chun-Chieh, "Computer-assisted yoga training system," *Multimed Tools Appl*, vol. 77, no. 4, pp. 23 969–23 991, 2018.

[4] A. Chaudhari, O. Dalvi, O. Ramade, and D. Ambawade, "Yog-guru: Real-time yoga pose correction system using deep learning methods," in *2021 International Conference on Communication information and Computing Technology (ICCICT)*, 2021, pp. 1–6.

[5] S. Chen and R. R. Yang, "Pose trainer: Correcting exercise posture using pose estimation," *CoRR*, vol. abs/2006.11718, 2020. [Online]. Available: https://arxiv.org/abs/2006.11718

[6] C. Chan, S. Ginosar, T. Zhou, and A. A. Efros, "Everybody dance now," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019, pp. 5933–5942.

[7] C. Lassner, G. Pons-Moll, and P. V. Gehler, "A generative model of people in clothing," *CoRR*, vol. abs/1705.04098, 2017. [Online]. Available: http://arxiv.org/abs/1705.04098

[8] L. Ma, X. Jia, Q. Sun, B. Schiele, T. Tuytelaars, and L. V. Gool, "Pose guided person image generation," *CoRR*, vol. abs/1705.09368, 2017. [Online]. Available: http://arxiv.org/abs/1705.09368

[9] G. Balakrishnan, A. Zhao, A. V. Dalca, F. Durand, and J. V. Guttag, "Synthesizing images of humans in unseen poses," *CoRR*, vol. abs/1804.07739, 2018. [Online]. Available: http://arxiv.org/abs/1804.07739

[10] V. et al., "3d pose based motion correction for physical exercises," Master's thesis, EPFL, 2021.

[11] Z. Cao, G. Hidalgo, T. Simon, S. Wei, and Y. Sheikh, "Openpose: Realtime multi-person 2d pose estimation using part affinity fields," *CoRR*, vol. abs/1812.08008, 2018. [Online]. Available: http://arxiv.org/abs/1812.08008

[12] S. Kreiss, L. Bertoni, and A. Alahi, "Pifpaf: Composite fields for human pose estimation," *CoRR*, vol. abs/1903.06593, 2019. [Online]. Available: http://arxiv.org/abs/1903.06593

[13] C. Lugaresi, J. Tang, H. Nash, C. McClanahan, E. Uboweja, M. Hays, F. Zhang, C. Chang, M. G. Yong, J. Lee, W. Chang, W. Hua, M. Georg, and M. Grundmann, "Mediapipe: A framework for building perception pipelines," *CoRR*, vol. abs/1906.08172, 2019. [Online]. Available: http://arxiv.org/abs/1906.08172

[14] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in *Advances in neural information processing systems*, 2014, pp. 2672–2680.

[15] M. Mirza and S. Osindero, "Conditional generative adversarial nets," *CoRR*, vol. abs/1411.1784, 2014. [Online]. Available: http://arxiv.org/abs/1411.1784

VII. Appendix

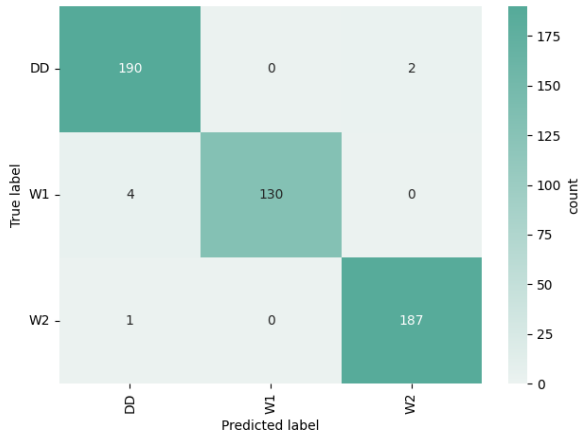


Figure 8: The confusion matrix for the training results of the pose type classifier. ”DD” stands for Downward Dog, ”W1” for Warrior 1 and ”W2” for Warrior 2.

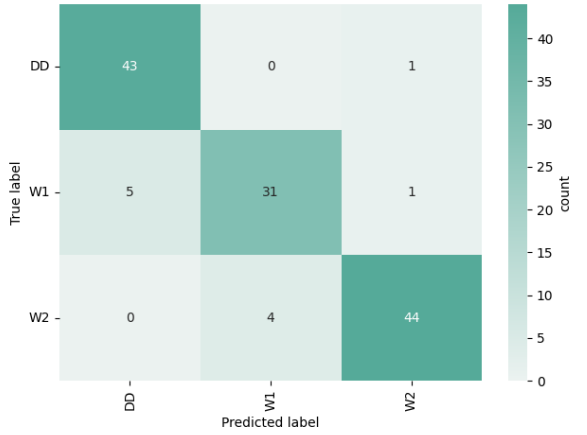


Figure 9: The confusion matrix for the validation results of the pose type classifier. ”DD” stands for Downward Dog, ”W1” for Warrior 1 and ”W2” for Warrior 2.

A. Network architectures

1) Pose classifiers: Our pose type classifier is a three layer MLP employing batch normalisation after each layer with the ReLU activation function to classify the pose, cf. Table IIa. Additionally, we used the Adam optimizer with initial learning rate 0.0002, a cross entropy loss function, a batch size of 64 and we trained the model for 800 epochs. Regarding our pose quality classifier, a three layer MLP was not able to fit the training data sufficiently, which is why we extended it to five layers. We are again using batch norm and a leaky ReLU activation function, cf. Table II. This network also gets trained with the Adam optimizer

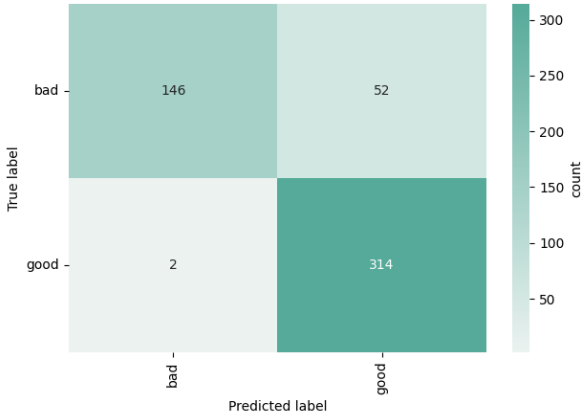


Figure 10: The confusion matrix for the training results of the pose quality classifier.

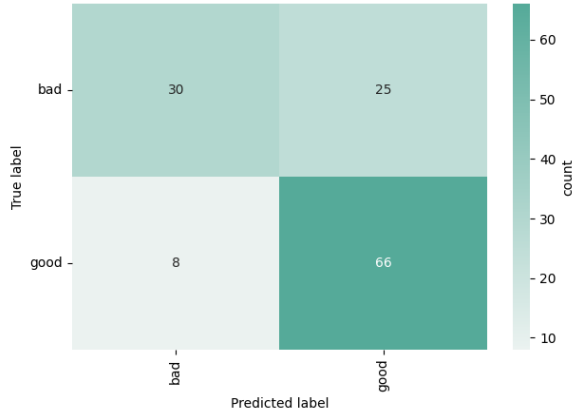


Figure 11: The confusion matrix for the validation results of the pose quality classifier.

with initial learning rate 0.0002, a binary cross entropy loss function, batch size 64 and 1000 epochs.

2) cLimbGAN: In order to verify that the GAN meets our expectations, we gradually increased the difficulty of the learning task. First, we trained a GAN for just one pose. Upon observing that it manages to learn the correct mapping to the good pose domain, we conditioned it on the one-hot encoded class label, where it again succeeded. Our final architecture is the $cLimbGAN$, which at correction time receives a noise prior with dimensionality 100, the user's pose and limb lengths and conditioned on them extracts a good pose from the latent distribution. The noise, the encoded pose and limb lengths get mapped by a linear layer to a 99 dimensional space and then concatenated before being passed through the network. The $cLimbGAN$ consists of two four layer MLPs, where they both employ the leaky ReLU activation function and the generator uses batch norm on top of that, cf. Table III. The model was trained using the Adam optimizer, an initial learning rate of 0.0002 and

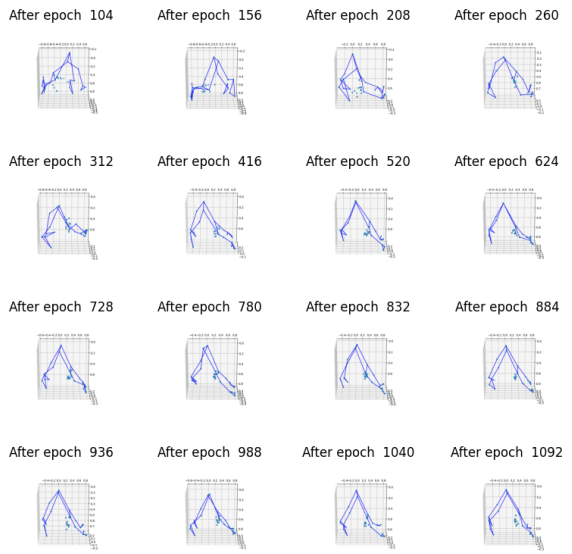


Figure 12: Generated images drawn during training of the $cLimbGAN$ from the learned latent pose distribution.

batches of size 64.

To investigate the convergence behavior of the GAN, we visually inspected the images it generates during training, cf. Figure 12. We can confirm that the GAN captures the principal characteristics of a good pose. To examine whether the GAN preserves the limb lengths of the user when generating the corrected pose, we track the euclidean distance of the generated to the original limb lengths during training of one pose. The results can be seen in Figure **??** and we can confirm that our $cLimbGAN$ does learn to preserve the limb lengths of the user. Table III shows the architectures of the $cLimbGAN$ generator and discriminator.

Layer	Output Shape
Input	33 * 3 * 3
Dense	512, 256, 128
Batch Norm	512, 256, 128
Leaky ReLU	512, 256, 128
Dense	33*3
Tanh	33 * 3

(a) cLimbGAN generator

Layer	Output Shape
Input	33 * 3 * 3
Dense	1024, 512, 256
Leaky ReLU	1024, 512, 256
Dense	1
Sigmoid	1

(b) cLimbGAN discriminator

Table III: Network architecture of the cLimbGAN generator and discriminator. The notation 128, 256, 512 or 1024, 512, 256 indicates 3 layers of the given dimension arranged sequentially.

	Keypoint 1	**Keypoint 2**	**Keypoint 3**
LEFT_ELBOW_ANGLE	left should	left elbow	left wrist
RIGHT_ELBOW_ANGLE	right should	right elbow	right wrist
LEFT_ARMPIT_ANGLE	left elbow	left should	left hip
RIGHT_ARMPIT_ANGLE	right elbow	right should	right hip
LEFT_CHEST_ANGLE	left elbow	left should	right should
RIGHT_CHEST_ANGLE	right elbow	right should	left should
LEFT_WRIST_ANGLE	left index	left wrist	left elbow
RIGHT_WRIST_ANGLE	right index	right wrist	right elbow
LEFT_KNEE_ANGLE	left hip	left knee	left ankle
RIGHT_KNEE_ANGLE	right hip	right knee	right ankle
LEFT_HIPFLEXOR_ANGLE	left should	left hip	left knee
RIGHT_HIPFLEXOR_ANGLE	right should	right hip	right knee
LEFT_ADDUCTOR_ANGLE	right hip	left hip	left knee
RIGHT_ADDUCTOR_ANGLE	left hip	right hip	right knee
LEFT_ANKLE_ANGLE	left knee	left ankle	left foot index
RIGHT_ANKLE_ANGLE	right knee	right ankle	right foot index

Table IV: 16 angles defined by the authors seen as important in yoga pose correction. should.=Shoulder