POSTURE ESTIMATION FOR YOGA ASANAS

[1]Shaunak Ketkar, [2]Swaroop Udgaonkar, [3]Manasi Karpe, [4]Swapnil Behere, [5]Vandana Rupnar
[1,2,3,4,5]MMCOE,
[1]shaunakketkar@gmail.com , [2]swaroopudgaonkar@gmail.com , [3]karpe.manasi@gmail.com,
4swapnilbehere10@gmail.com, [5]vandanarupnar@mmcoe.edu.in

Abstract— Posture is a way in which a human holds his/her body. People prefer to exercise at home for fitness, however, the wrong way of exercising makes muscle damage. The main objective is to assess the input yoga pose with the standard yoga pose and provide a quantitative assessment about the same. In this, a 2-phased algorithm is used which has dectectron-2 and temporal dilated convolutions. With this, we also provide a self-supervised approach for training the data.

Posture estimation, as well as correction, can hence help users to learn, and practice exercises themselves correctly. The project compares various machine learning algorithms to identify exercises of yoga on a dataset of pre-recorded videos and in real-time videos. If the posture is incorrect the system also recommends a trainer to help the user in maintaining the posture. We record a dataset of over 88 exercise videos of correct and incorrect forms, based on personal training guidelines, and build geometric-heuristic and machine learning algorithms for evaluation and support of any Windows or Linux computer with a GPU.

Index Terms— Deep learning, Open Pose, Dilated convolutions, Detectron-2.

INTRODUCTION:

Exercise is an integral part of a person's daily life. It has advantages for not only the physical health of an individual but also the mental health. Yoga is a physical exercise consisting of varied postures. One needs to keep his/her pose straight. Wrong posture can lead to an individual's muscles becoming stiff and cause many problems to the human body. Doing the yoga asanas, the wrong way led to acute pain as well as chronic standing problems. As a result, keeping a good posture while exercising is very important. Posture estimation is a technique that tracks the movements of a human. Human pose estimation has been studied for over a decade. However, recent development in deep learning has helped in increasing the performance of posture estimation. Creating a model with high accuracy for all the yoga asanas is a complex challenge. The dataset used for training the model plays an important role in the performance of the model. After estimating, an image or a video of correct yoga asana can be demonstrated to the user. For users with a bad pose, a yoga instructor can also be recommended based on the location. The following sections talk more about pose estimation and explain its types in detail. The use of Convolutional Neural Networks (CNNs) and their importance in increasing the accuracy is also discussed.

METHODOLOGY AND ARCHITECTURE:

i. Human Posture Estimation:

Posture estimation has made advancements in recent years from 2D to 3D, from one user estimation to multi-user estimation (Hossain & Little, 2018) (Martinez, Hossain, Romero, & Little, 2017). In multi-user estimation, the location of users is unknown which makes it more difficult than single posture estimation. The issue can be resolved by two different approaches namely the top-down and bottom-up approaches.

A. Top-down Approach: It is a simple approach that detects the person in the video first and then estimates the posture of the individual.

B. Bottom-up Approach: It detects all the key points in the image/video and then associates them with distinct individuals.

Calculations are needed for posture assessment and are divided into two categories namely generative strategies and discriminative strategies.

Generative strategy starts with establishing the posture of the human body and visualizing it in the picture plane. They have very less requirements of a present dataset. It has a high dimensional projection space search and hence the technique is not considered plausible.

Discriminative-based strategy starts with the proof of the picture. Model testing is quicker than compared with generative strategy due to the hunt in obliged space.

ii. Detection Methods for Key-Points in Posture Estimation:

I. OPENPOSE: Using CNN to identify key points of a human body, Open Pose incorporates ears, eyes, elbows, shoulders, and knees utilizing an RGB camera. The project utilizes Open Pose for keypoint extraction and is followed by CNN for yoga asanas. The network extracts the highlights of the picture using its underlying layers and passes them to two convolutional layer branches.

II. POSENET: It is used for the ID of postures in picture and video successions. The model can predict key points in the size of original pictures regardless of whether the picture is downscaled or not.

iii. Deep Learning for Pose Classification:

The following models are used for the classification of posture:

A. Recurrent Neural Networks (RNNs): RNNs are a type of neural network where the output of the previous step is fed as an input to the current step. Any exercise is considered a sequence of postures. There is a reliance on the joint areas in yoga asanas. RNNs can make an appropriate decision for posture estimation while doing yoga asanas. However, if the steps in yoga asanas are too much, it is hard to monitor for RNNs and they end up in the long-haul reliance issue.

Long Short-Term Memory (LSTM): To handle the long-haul reliance issue, LSTM, an RNN is used which can recall data. It can hence overlook or recollect the learnings. There are three entryways in LSTM, namely, information, refresh, and overlook. It can hence consider long maintenance of the information state, LSTMs give accurate outcomes.

B. Convolutional Neural Networks (CNNs): The most popular neural network in the computer vision domain, CNNs have proved to be highly effective for the most image-related data. Convolutional layers of CNNs are responsible for feature extraction on the input and analyzing some parts of the input and then sending the output to the next layer. Feature map is generated from the convolutional layer with the help of convolutional filters. In the end with the help of the pooling layer, dimensionality is reduced to prevent overfitting. CNNs are trained on key points of human skeleton or images. CNN with Open Pose has achieved an accuracy of 78% for posture estimation. Accuracy can be improved with a better dataset. Activation functions are applied to CNNs to add non-linearity as the convolutional operation is linear (Newell, Yang, & Deng, 2016).

iv. Pose Estimator:

The pose estimator will have 2 phases:

1. Video to 2d key-points detection
2. 2d key-point to 3d key-points

Phase 1 comprises dectectron-2 which is an implementation of keypoint_rcnn_R_101_FPN_3x. CNN works on a region proposal mechanism; this is coupled with the traditional sliding window algorithm. In the original sliding window algorithm, the window slides through the whole image which causes the time complexity to increase, to tackle this issue, the region proposal mechanism was introduced, in this mechanism, a segmentation algorithm is used to propose a region for the window to slide. If we use the convolutional approach of this strategy, then we end up with an alternate version of this algorithm which is often called ‘faster rcnn’.

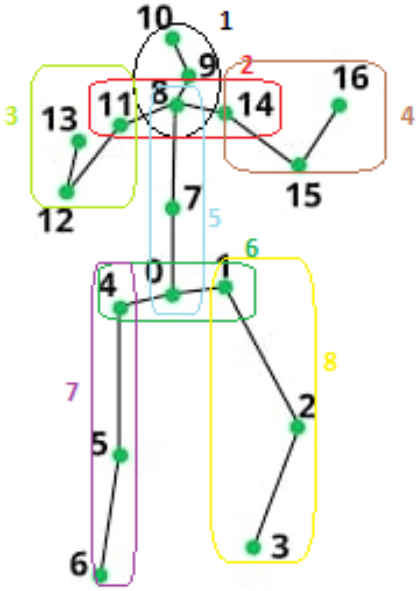
The next phase uses dilated temporal convolutions, this approach takes 2d key points

as input and uses the 1d convolution operator on them, this 1d convolution is a dilated one, which not only maintains the accuracy but also boosts the efficiency. This also leads to a significant reduction in several parameters while also utilizing a high receptive field. Moreover, since these are convolutions, the whole process becomes highly parallelizable, which isn't the case with RNN and LSTM.

Back Projection is a self-supervised methodology, which is used when we don't have enough data for training, in back-projection first we take the 2d key points, and using them we estimate the 3d key points. Then these 3d key points are projected back into 2d, and then we compare the original and reconstructed 2d key points. Thus, we get a formulation of the training methodology which is commonly used during the self-supervised training of autoencoders.
Finally, we pass the 3D key points through a Siamese network and get the embedding for the pose. This Siamese network is trained on triplet loss. We have tried to keep this network as disentangled as possible to regain interpretability though we feel that might have reduced the performance.

Experimental Setup:
We have used the Yoga videos dataset which contains 88 videos containing Padmasana, Bhujangasana, Trikonasana, Tadasana, and Vrikshana. For 2D key-point estimation we have used the Detectron2 implementation of keypoint_Rcnn_R_50_FPN_3x. Then we used the Dilated Convolutional Architecture for extracting 3D key points. We took those 3D key points and rearranged them by grouping certain parts of the human body like the head, shoulders, right arm, left arm, etc.

The key points may get duplicated but that is accounted for in the architecture. Thus the (17,3) shaped key points are transformed into (24,3). Other Pytorch-specific adjustments in shape are made like adding the batch size and channel's dimension which is 1,1 in our case. So the final shape should be (1,1,24,3). We pass this to our 5-Layered Siamese Network. The first layer has a kernel size of (3,3), and a stride of (3,1) so that we can extract the features from the grouped body parts mentioned earlier and the number of out_channels is 8. This gives us a vector of shape (1,8,8,1). The next four layers have kernel size of (1,1) with numbers of out_channels as 16, 8, 4, 1 respectively. Then we arranged 50 pairs of anchors, positive and negative samples, and trained the network for 12 epochs. This was trained with Triplet Margin Loss with a margin of 2.2. The optimization is done by Adam optimizer with a learning rate of 0.02



RELATED WORK:
Human movement recognition has been used in various applications, there has been a lot of development in single image recognition, Pavlakos's research [4] attempts to find the 3d pose from a single RGB image. Their approach relatively reduced the error by 30%. Even though there has been a quick development in the technology, there are still some errors, this paper offers insight and a direction for the existing problems in 3d pose, Martinez's work, [2] offers a simple approach to this problem which can be used as a baseline. They use a simple feedforward network on 2d keypoints. They find that even though 2d pose estimators have matured a lot they still are the main problem that hinders the estimation of 3d pose. There has been a lot of work on semi-supervised training, unlabeled recordings have been used for pre-training, but these recordings are rarely available. Contrary to the stacked hourglass method, Pavllo's work suggests using CPN as a more robust method. [5] It uses dilated temporal convolutions over a 2D keypoint. Dilated

convolutions have a higher receptive field and learn rich deterministic mappings. This is a fully convolutional model which makes it easier to parallelize over time and hence reduces training time. They also proposed a semi-supervised training method to improve accuracy in settings where the availability of labeled 3D ground-truth pose data is limited.

The pose estimation used is OpenPose which is a combination of CNN and LSTM. Various keypoint detection methods like PoseNet and OpenPose are CNN-based and are used to detect keypoints.

Hossain's work focuses on LSTM

[1] used the LSTM seq-to-seq model for handling temporal dependency. They found that their model improves the best-reported result on the Human3.6M dataset by approximately 12.2%. They also observed decent results even if the 2d pose detector failed.

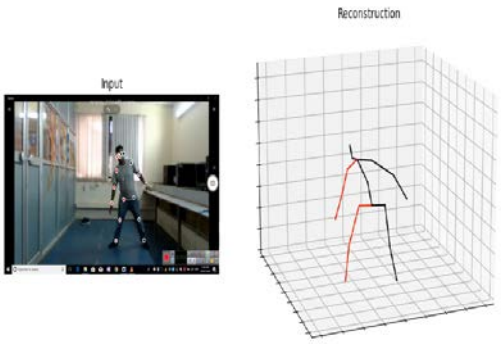
Information is captured at every scale, in a stacked hourglass model, it provides a pixel-wise prediction,

As multiple hourglasses are stacked together, bottom-up, top-down inference allows reevaluation of all the features of the entire image.

[3] The hourglass architecture forces the network to learn features at every layer resulting in rich

encodings of images. Thus, they witnessed over 2% average accuracy improvement over all joints and 4-5% on the difficult ones.

RESULTS:



The input video and the 2d reconstructed video.

We proposed a posture estimation system based on machine learning algorithms. We utilized a fully convolutional model on dilated temporal convolutions to estimate the user's posture while performing yoga asanas. Our system improves the estimated result. The method worked with the videos making it practical to use in scenarios like exercising. We believe the system advances the state-of-the-art in video-based Pose Estimation and will be used in a better way for real-time applications too.

Report Generation.

The Siamese network embeds the 3D key points into an 8-Dimensional Space. These 8 Dimensions roughly correspond to the features extracted from different parts of the human body. We found out that the embedding effectively represents the pose of the person. We reduced the dimensionality from 8 to 2 for visualizing the embedding. We can see in the below figure (Fig. 1) that there is a clear distinction between asanas as each asana is marked by a separate color. Of course, the embedding depends on the local/global minima achieved on the loss function and thus may be vastly different from before. But the separation between the classes will remain similar. This shows that the Siamese network was optimized correctly which is also evident through the training loss graph (Fig. 2).

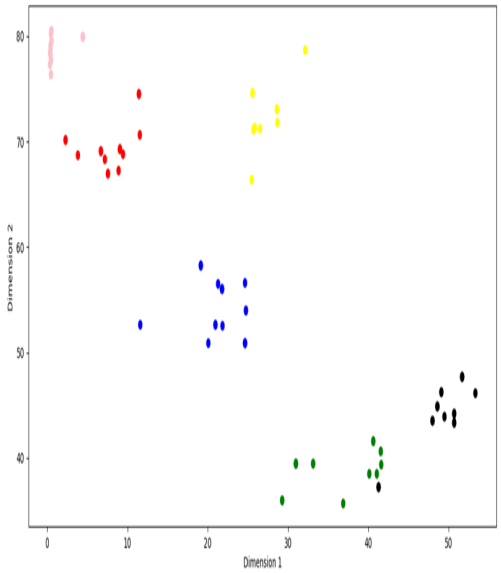


Fig. 1: Project Embeddings

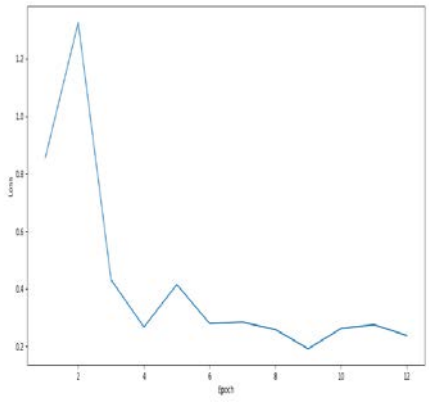


Fig. 2: Training Loss

Then we evaluated this model on different metrics like Precision, Recall, and F1 score. We did this by finding the mean distance between the anchor and positive samples and anchor and negative samples and then took 5 equally spaced points between these means as a threshold to decide whether a sample should be considered positive or negative.

	Precision	Recall	f1	thresholds
0	0.8913043478	0.82	0.8541666667	17.9981823
1	0.8823529412	0.9	0.8910891089	22.82213545
2	0.8363636364	0.92	8.8761904762	27.6460886
3	0.8214285714	0.92	0.8679245283	32.47004175
4	0.71875	0.92	0.8070175439	37.2939949

Fig. 3: Metric Results

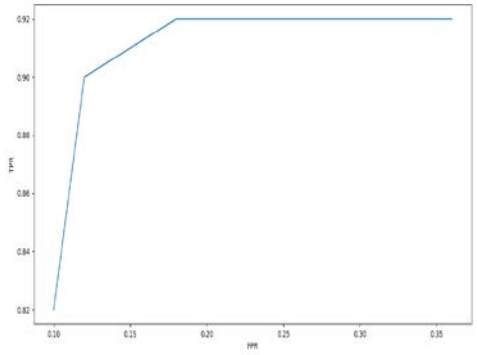


Fig. 4: ROC Curve

Now considering computationally we can safely say that our network is highly feasible even on a CPU. The computational summary of the inference of our model is shown below (Fig. 6).

As for the training, results don't change much. We have profiled the results for 1 epoch and all of these computational results are for CPU. This shows that this network can be easily trained and used on a CPU. We haven't included the data loading in the profiler and hence there is no trace of Data-loader in the graphs. In the graph (Fig. 7) we can see that most of the time is taken by the convolution operation. There aren't many convolutional layers but if the network does become in-feasible then a GPU will be enough as convolutions are inherently parallelizable and take up most of the time in this case.

Execution Summary

Category	Time Duration (us)	Percentage (%)
Average Step Time	6,310	100
DataLoader	0	0
CPU Exec	5,325	84.4
Other	984	15.6

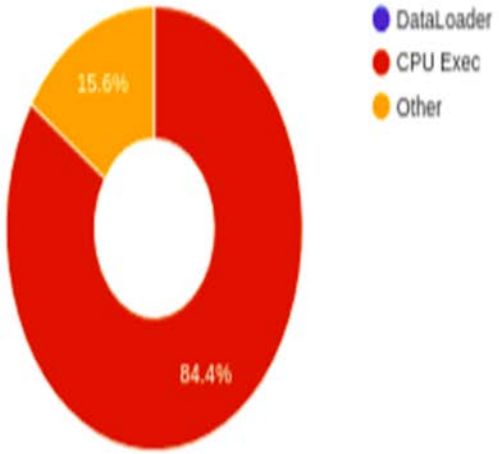


Fig. 5: Train Execution Summary

Name	Self CPU%	Self CPU	CPU Total %	CPU Total	CPU time avg	CPU Mem	Self CPU Mem	# of Cells
model_inference	29.32	280us	94.55	903us	903us	-4 b	-2.68 kb	1
aten::conv2d	2.72	26us	57.28	547us	189.4us	1.16 kb	0 b	5
aten::convolution	6.49	62us	54.55	521us	104.2us	1.16 kb	0 b	5
atten::_convolution	5.34	51us	48.06	459us	91.8us	1.16kb	0b	5
atten::thn_conv2d	1.47	14us	41.68	398us	79.6us	1.16kb	0b	5
atten::_slow_conv2d_forward	29.53	282us	40.21	384us	76.8us	1.16kb	-288b	5
aten::leaky_relu	7.23	69us	7.23	69us	13.8us	1.16kb	1.16kb	5
aten::zeros	3.25	31us	5.45	52us	52us	4b	0b	1
aten::empty	4.08	39us	4.08	39us	4.875us	660b	600b	8
aten::resize_	3.04	29us	3.04	29us	5.8us	1.16kb	1.16kb	5
Self CPU Time Total: 955us								

Fig. 6: CPU Memory Summary

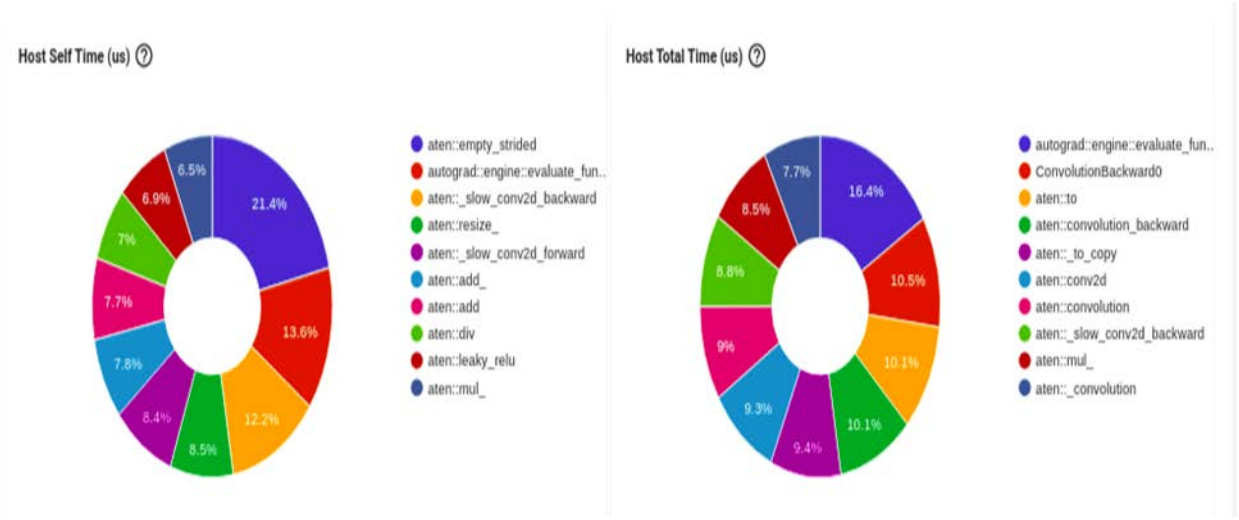


Fig. 7: CPU Breakdown

The graph below shows memory consumption during the training. There we can observe steady allocation and deallocation. The real time consumer is the 2D key-point estimation and 3D key-point extraction which may take up to 10-15 minutes depending on the size of the video.

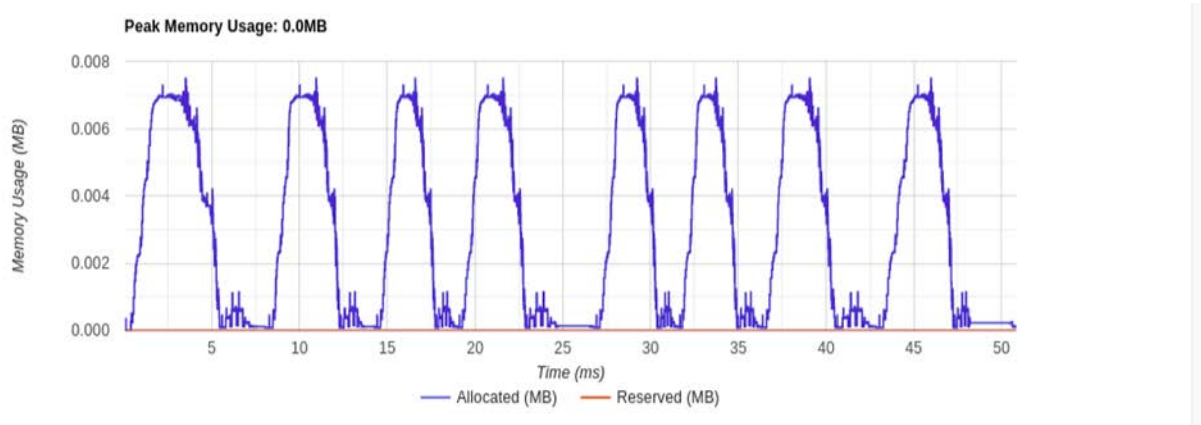


Fig. 7: Memory Usage

CONCLUSION & FUTURE SCOPE:

Human pose estimation has been studied extensively over the past years. As compared to other computer vision problems, the human pose estimation is different as it must localize and assemble human body parts based on an already defined structure of the human body. The application of pose estimation in fitness and sports can help prevent injuries and improve the performance of people's workouts. We suggest yoga self-instruction systems carry the potential to make yoga popular by making sure it is performed in the right manner. Deep learning methods are promising because of the vast research being done in this field. The use of a model on given data is seen to be highly effective and classifies all the yoga poses perfectly.

The proposed models currently classify yoga asanas. There are several yoga asanas, and hence creating a pose estimation model that can be successful for all the asanas is a challenging problem. The dataset can be expanded by adding

more yoga poses performed by individuals not only in indoor settings but also outdoor. A portable device for self-training and real-time predictions can be implemented for this system. This work demonstrates activity recognition for practical applications. An approach comparable to this can be utilized for pose recognition in tasks such as sports, surveillance, health care, etc. Multi-person pose estimation is a whole new problem and has a lot of scope for research. There are a lot of scenarios where single person pose estimation would not suffice, for example, pose estimation in scenarios would have multiple persons which will involve tracking and identifying the pose of everyone. A lot of factors such as background, lighting, overlapping figures, etc. The future scope of the project includes estimating posture for real-time applications considering all the light factors. A better understandable report with the real-time tutor is a scope to the existing project.

I. REFERENCES

[1] Hossain, M. &. (2018). Exploiting temporal information for 3d human pose estimation. *Proceedings of the European Conference on Computer Vision (EECV)*, 68-84.

[2] J. Hossain, R. R. (2017). A simple yet effective baseline for 3d human pose estimation. *IEEE*, (pp. 2640-2649).

[3] Newell, A. Y. (2016). Stacked hourglass networks for human pose estimation. *European Conference on Computer Vision*, (pp. 483-499).

[4] Pavlakos, G. Z. (2017). Coarse-to-fine volumetric prediction for single-image 3D human pose. *IEEE conference on computer vision and pattern recognition*, (pp. 7025-7034).

[5] Pavllo, D. F. (2019). 3D human pose estimation in video with temporal evolutions and semi-supervised training. *Proceedings of the IEEE/CVF conference on Computer Vision and Pattern Recognition*, (pp. 7753-7762).

[6] Sun, J. J.-C. (2020). View-invariant probablistic embedding for human pose. *European Conference on Computer Vision*, (pp. 53-70).