HIGH SCHOOL EDITION
Journal of Student Research

Volume 10 Issue 3 (2021)

Yoga pose perfection using Deep Learning: An Algorithm to Estimate the Error in Yogic Poses

Satyam Goyal[1] and Animesh Jain[#]

[1]American High School, Fremont, CA, USA
[#]Advisor

ABSTRACT

Even with lots of attention and work in the computer vision and artificial intelligence field, human body pose detection is still a daunting task. The application of human pose detection is wide-ranging from health monitoring to public security. This paper focuses on the application in yoga, an art that has been performed for over a millennium. In modern society yoga has become a common method of exercise and there-in arises a demand for instructions on how to do yoga properly. Doing certain yoga postures improperly may lead to injuries and fatigue and hence the presence of a trainer becomes important. As many people don't have the resources to have a yoga instructor or guide, artificial intelligence can act as a substitute and advise people on their poses. Currently, the research surrounding pose estimation for yoga mainly discusses the classification of yogic poses. In this work, we propose a method, using the Tensorflow MoveNet Thunder model, that allows real-time pose estimation to detect the error in a person's pose, thereby allowing them to correct it.

Introduction

Due to the growing level of awareness towards mental health, there has been a lot of research in mindfulness practices such as yoga. Yoga is a physical and spiritual art form that was first developed in southern Asia to allow users to transcend their consciousness and increase the understanding of themselves [1]. The yoga discipline uses a multitude of techniques from breathing to physical poses to achieve this [1]. Yoga helps achieve physical fitness along with mental wellness. The practice became more popular in western culture during the 1800s in Europe and eventually America. [1] As yoga became popular among academics its uses modernized and it has become popular among medical researchers due to its ability to cure diseases and improve a patient's health without the use of drugs [2]. Most recently, with the spread of Covid-19, yoga has become part of life to improve mental health and well-being and is able to resolve the issues caused by isolation and separation.

To obtain the benefits of yoga, it is important that yoga should be practiced in the correct postures and forms, just like any other exercise. Incorrect postures can cause unproductive results to the extent of pernicious effects. This creates the necessity for a Yoga instructor to administer the appropriate individual posture. The availability of Yoga instructors is very limited, and that makes it a costly affair as well. Overall cost and resource availability make it a difficult outreach for Yoga and hence creates the need for technology-based assistance, which calculates the accuracy of postures based on mathematical models.

Pose estimation and classification is a subject that has been extensively researched and therefore has made great advances in recent years. Current classification models detect the poses but don't account for the accuracy of a pose. An artificial intelligence-based application might be useful to identify yoga poses and provide personalized feedback to help individuals improve their poses [5]. In recent years, human pose estimation has benefited greatly from deep learning, and huge gains in performance have been achieved [5]. Deep learning approaches provide a more straightforward way of mapping the structure instead of having to deal with the dependencies between structures manually. We employ the Tensorflow MoveNet Thunder model for pose estimation in this work.

MoveNet is a pose estimation model that detects the pose with 17 key points in the human body through images or video sequences. These joint locations or key points are indexed by "Part ID" which is a confidence score whose value lies in the range of 0.0 and 1.0, with 1.0 being the greatest. The MoveNet model's performance varies depending on the device and output stride [14]. The MoveNet model is invariant to the size of the image, thus it can predict the pose position accurately.

We propose an algorithmic way to estimate the error in a yogic posture using angle calculations between joints. We analyze the results for three yogic postures, (i) Downward Facing Dog pose (Adho Mukha Svanasana), beneficial in getting rid of headache and backache; (ii) Warrior pose (Virabhadrasana), beneficial in improving emotional and mental stability, also strengthens the lower and middle back; (iii) Plank Pose (Kumbhakasana), beneficial in strengthening the abdominal organs and increasing core strength. All these poses if done properly have immense physical, mental, and spiritual benefits, however, if practiced improperly for a long time may lead to harmful effects instead. Finding the error can help improve yoga posture and ultimately will help users in doing yoga properly.

In this work, we (i) train the Pose estimation algorithm using datasets of asanas to find the "perfect pose", (ii) propose an algorithmic method to calculate the error between perfect pose and the current pose, and (iii) show the results of the model in real-time pose correction. Conclusions end the paper.

Methods

To determine the quantitative error in a person's pose, a pose must need to be defined in a numerical sense so that a calculation can be done to find a numerical error. For a yoga pose, the most important information is the angle between certain body parts [18]. Pose Estimation provides information about a person's pose in coordinates and then uses them to draw lines on devices used for the detection. Coordinates can be used to form vectors and then be used to find corresponding angles between adjacent vectors. Hence, we use an array of angles as means of describing one's pose. To find the error, a basis or "correct pose" needs to be used to compare against the user's pose. As there is now a way to determine poses in a numerical sense, two arrays can be used to define the current pose and the "perfect" pose. After establishing the poses, a calculation can be done to find the overall resulting error of the current pose in comparison to the selected "perfect" pose.

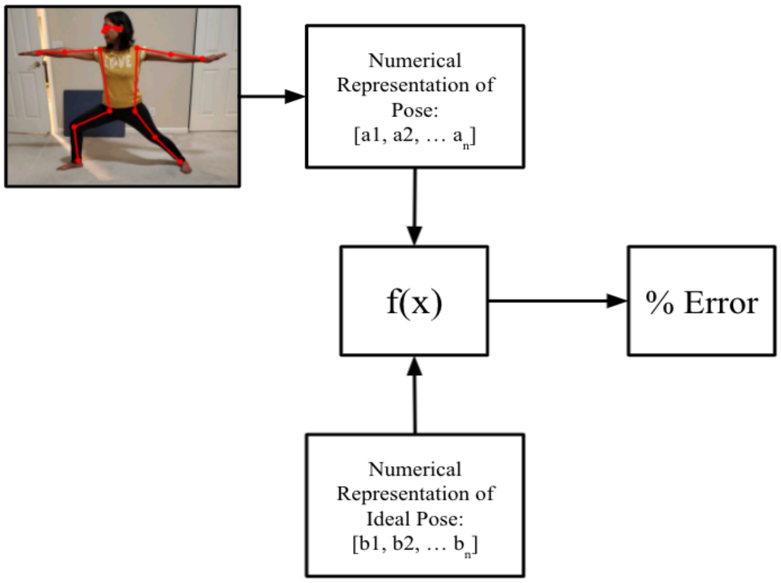

Numerical Representation of Pose: [a1, a2, … a_n]
f(x)
% Error
Numerical Representation of Ideal Pose: [b1, b2, … b_n]

Figure 1. A schematic of error estimation method.

This paper first examines how data was selected to be used for the pose comparison and how the perfect data was gathered. Then, the algorithm used to find the actual error is discussed, and finally, the implications and applications of this algorithm are observed.

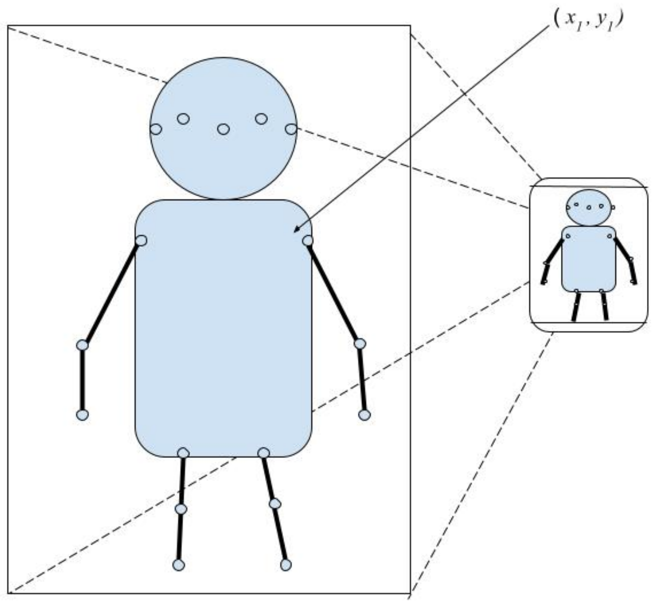


Figure 2. Key points of the MoveNet Thunder model. The illustration depicts the points that the MoveNet Thunder model is able to predict. The mobile device on the right captures the position of each joint.

MoveNet Thunder can give coordinates of key points for 17 distinct joints on the body as described in Figure 2. The coordinates are relative to the device's screen dimensions. Pairing adjacent joints create line segments that are displayed in the output of the model. The coordinates information can be used in a slightly different manner to create vectors for each segment. This is done by subtracting x-coordinates and y-coordinates of adjacent joints.

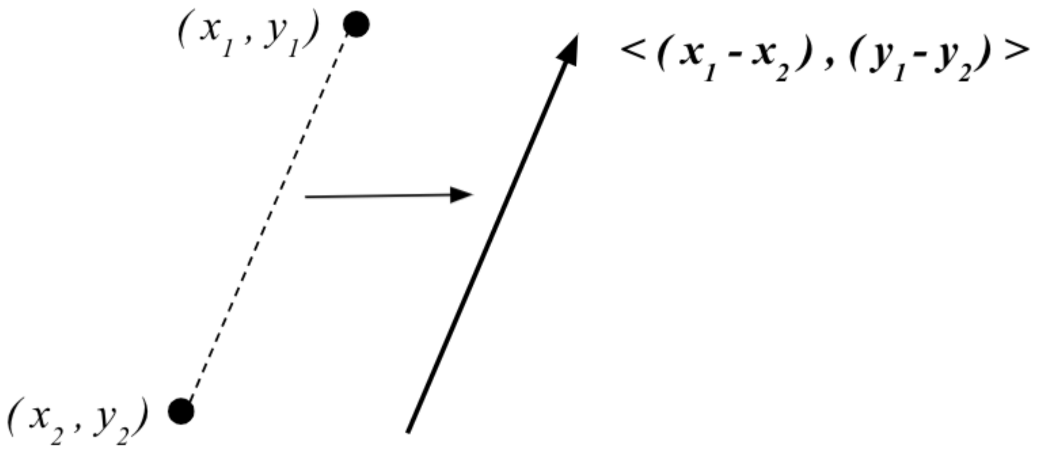
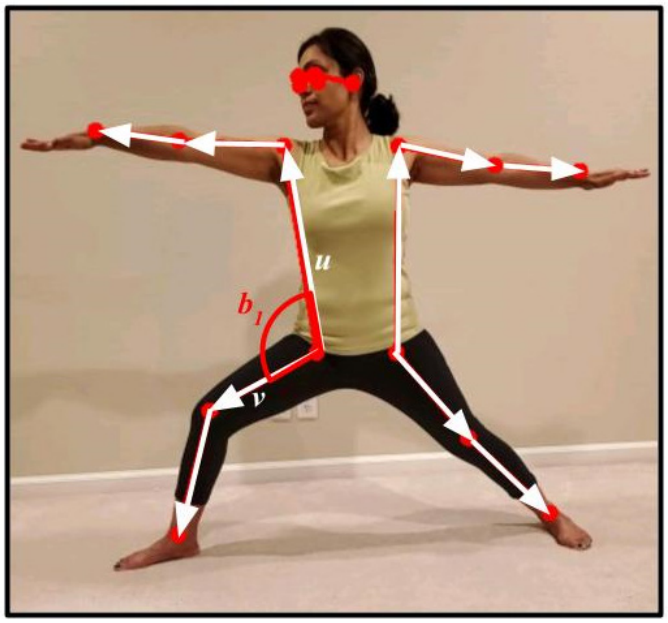


Figure 3. A graphical representation of how two coordinates, in this case, the location of two points on the body, can be converted into a vector.

A primary reason this conversion is done, as shown in Figure 3, is to convert two pairs of numbers into one, thereby decreasing the complexity of the task. Once all paired points have been converted into vectors, they can be used to find angles between adjacent vectors, for example, the angle between vectors from the knee to ankle and the hip to the knee. The cosine formula can be used for this:

$$\cos b = \frac{(\underline{u} \cdot \underline{v})}{(\,||\underline{u}||\;||\underline{v}||\,)}$$

Vectors *u* and *v* represent adjacent vectors.



With angles for all adjacent vectors, as shown in Figure 4, there are now eight angles that can be stored in an array form. Another way to consider this is a vector with eight components. In this paper we represent the angles we gather as b_1 to b_n. Angles we gather for the ideal pose we use a_1 to a_n.

To find data on creating an ideal pose a Kaggle dataset [25] was used as a model. As much research has been done in the past using this data set it was appropriate to use it to train the model. There is now a vector that can neatly represent a person's pose and with a basis of comparison from the Kaggle dataset, an algebraic calculation can be done to find the error.

There are many approaches that can be taken regarding the error estimation for a pose. Initially, the method of finding the Euclidean distance between the two vectors was used. The Euclidean distance – or some other form of vector distance, such as the Minkowski Distance – was an ideal as a mathematically accurate way to determine the difference in eight-dimensional vectors. Additionally, weights can be applied to certain angles based on their relevance to a particular pose.

$$d = \sqrt{k_1(a_1 - b_1) + k_2(a_2 - b_2) + \ldots + k_i(a_i - b_i)}$$

In the equation, k_i would be a constant and be defined as a component of the vector k. The problem surrounding this equation is that the distance or "error" has no bounds and as a result has no quantifiable measurement. Simply, stating "the distance is ten" is not meaningful unless given mathematical context. A percentage of error would be ideal as a user could understand the level of correctness in their pose. To scale the output of this equation into a percentage would require the output to be put into a function that would have a range between 0 and 1, and therefore result in a percentage value. As the discovery of such a function, that would act as a scale for the values, has yet to be researched this approach was dropped.

In replacement, a new method was adopted that used a simpler approach and directly calculated a percent error.

Equation (1)

$$\frac{\sum_{i=0}^{n} \frac{|a_i - b_i|}{a_i}}{n} \cdot 100$$

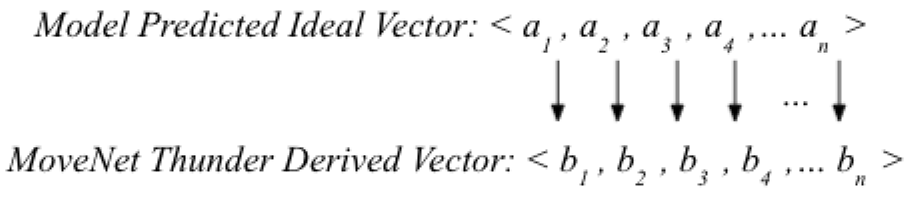

Model Predicted Ideal Vector: < a1 , a2 , a3 , a4 ,… an >
…
MoveNet Thunder Derived Vector: < b1 , b2 , b3 , b4 ,… bn >

Figure 5. Vectors that will be used for error derivation. The illustration shows the two vectors that form. The top one results from the data collected in Figure 5 and Table 1 and the other is formed from the method described in Figure 2, 3, and 4. Each component in vector *a* will be compared to vector *b* for the total error.

Figure 5 illustrates the method. It shows *a* is the vector determined from the Kaggle dataset, our "perfect" pose, and *b* is the vector determined from Pose Estimation's results. Using equation (1), the error can be estimated. The error values can help users to modify the pose in real-time to reach the perfect pose.

Results

In this section, we present and discuss the results of the above-mentioned methods. To obtain the angles between different joints in a perfect pose, we train the model using an image dataset. The angles acquired for one pose (Downward Facing Dog pose) are shown in Figure 5. The same process for finding the angles in a perfect pose was used for the other two poses. The averaged values of angles for all three poses are shown in Table 1.

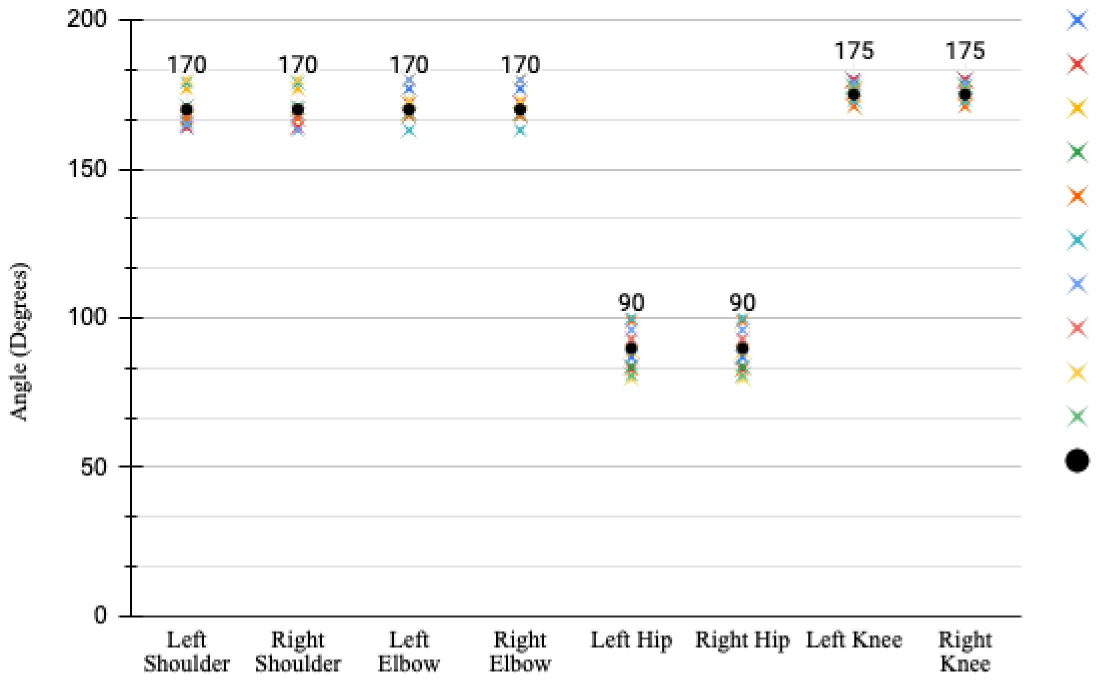


Figure 6. Data for Downward Facing Dog pose (Adho Mukha Svanasana). The x-axis in the graph is the 8 joints that the algorithm calculates as defined in Figure 4 and the y-axis is the resulting angle in degrees. The average value is denoted as a black circle for each joint data series. This graph displays the results for Adho Mukha Svanasana.

Table 1. A total of 240 data points were collected and then averaged. 80 of the data points are displayed in Figure 5 along with their averages.

(i) Downward Facing Dog pose (Adho Mukha Svanasana)

Joint:	Left Shoulder	Right Shoulder	Left Elbow	Right Elbow	Left Hip	Right Hip	Left Knee	Right Knee
Angle:	170º	170º	170º	170º	90º	90º	175º	175º

(ii) Warrior pose (Virabhadrasana)

Joint:	Left Shoulder	Right Shoulder	Left Elbow	Right Elbow	Left Hip	Right Hip	Left Knee	Right Knee
Angle:	85º	90º	170º	170º	135º	100º	170º	120º

(iii) Plank pose (Kumbhakasana)

Joint:	Left Shoulder	Right Shoulder	Left Elbow	Right Elbow	Left Hip	Right Hip	Left Knee	Right Knee
Angle:	80º	80º	170º	170º	170º	180º	175º	175º

After obtaining the angles of a perfect pose, the error in a real-time pose is calculated using equation 1 with $n = 8$. Initial testing was done using this algorithm to compare the average data from the Kaggle dataset and individual images. The algorithm reported an average of 15% error. As a result, this error was concluded due to slight variations in the pose of everyone, however, still didn't contribute to the pose being incorrect. The user data shown below is adjusted so that only error values of greater than 15% are displayed and included in the overall pose error.

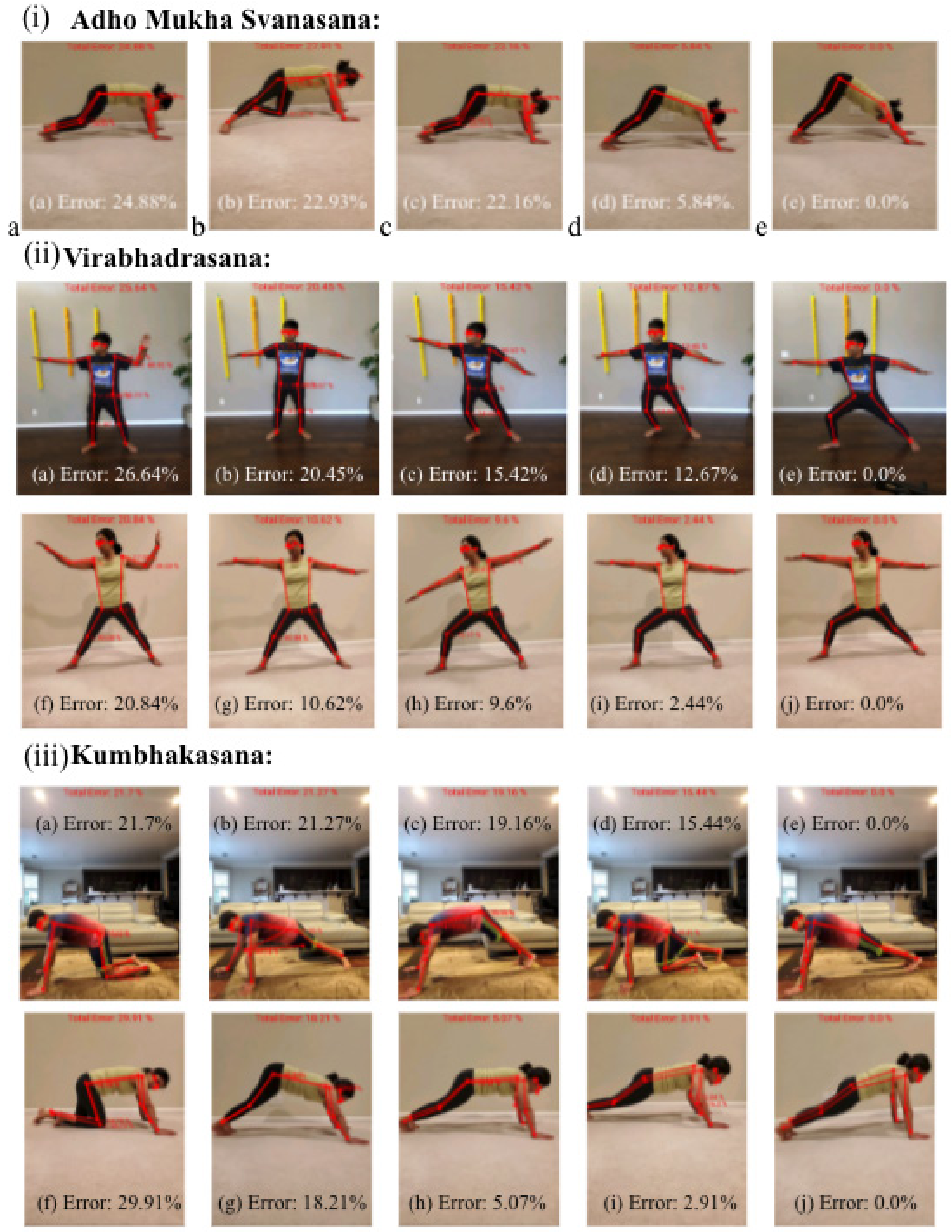


Figure 7. Results of model testing. Some tests of the algorithm are displayed in Figure 7. Each image has the total error of the pose in the text at the top, this error is defined by Equation 1. Additionally, for individual joints, the error is marked next to the joint. The tests are organized by poses and decreasing error from left to right.

Figure 7 shows the results of the method proposed for real-time error estimation for the three yoga postures. Figure 7 (i. (a-e)) shows the person entering in the posture and correcting the posture based on the error obtained while practicing the Downward Facing Dog pose (Adho mukha svanasana). Similarly, Figure 7 (ii & iii) shows the results for the Warrior pose and Plank pose. It can be observed that the error estimation by the method proposed can help get the maximum benefit of any exercise including yoga by practicing every posture perfectly. It can also help a practitioner gauge the progress and move towards a healthier mind and body.

Discussion

By using a mathematical algorithm on top of Google's Pose Estimation model, MoveNet Thunder, an accurate understanding of the error in a person's pose can be determined and, in the future, it will allow users to correct their posture based on the information the algorithm can present. As shown in Figure 8, as subjects can view the results of the algorithm, they improve their posture and make corrections as needed until they can correct their posture.

Conclusion

Human poses are seemingly complex, still emerging applications such as yoga posture detection have made the implementation of the technology imperative. An artificial intelligence-based resource that allows people to do yoga and other activities properly can help improve the popularity and benefits of these practices. In this work, we propose a model for error estimation in a yoga posture using Google's Pose Estimation model, MoveNet Thunder. The model was trained to find a perfect pose using an image dataset. The perfect pose can be compared with the pose of the practitioner in real-time, thereby helping them correct the pose until a perfect pose is reached. The method proposed in this work can help the development of applications and other resources for making yoga and its benefits accessible to everyone.

Acknowledgment

We would like to thank my mentor, Pranay Pogde, and my advisor, Animesh Jain. Without their technical guidance and feedback on the paper, this research would not have been possible.

References

[1] Douglass, Laura. "How did we get here? A history of yoga in America, 1800-1970." International Journal of Yoga Therapy 17.1 (2007): 35-42

[2] S. Patil, A. Pawar, and A. Peshave, "Yoga tutor: visualization and analysis using SURF algorithm",Proc. IEEE Control Syst. Graduate Research Colloq.,pp. 43-46, 2011.

[3] L. Sigal. "Human pose estimation", Ency. of Comput. Vision, Springer 2011.

[4] S. Yadav, A. Singh, A. Gupta, and J. Raheja, "Real-time yoga recognition using deep learning", Neural Comput. and Appl., May 2019. [Online]. Available: https://doi.org/10.1007/s00521-019-04232-7

[5] S. Kothari, "Yoga Pose Classification Using Deep Learning" https://scholarworks.sjsu.edu/etd_projects/932/

[6] U. Rafi, B. Leibe, J.Gall, and I. Kostrikov, “An efficient convolutional network for human pose estimation”, British Mach. Vision Conf., 2016. [4] S. Haque, A. Rabby, M. Laboni, N. Neehal, and S. Hossain, “ExNET: deep neural network for exercise pose detection”, Recent Trends in Image Process. and Pattern Recog., 2019.

[7] M. Islam, H. Mahmud, F. Ashraf, I. Hossain and M. Hasan, "Yoga posture recognition by detecting human joint points in real time using microsoft kinect", IEEE Region 10 Humanit. Tech. Conf., pp. 668-67, 2017.

[8] S. Patil, A. Pawar, and A. Peshave, “Yoga tutor: visualization and analysis using SURF algorithm”,Proc. IEEE Control Syst. Graduate Research Colloq.,pp. 43-46, 2011.

[9] W. Gong, X. Zhang, J. Gonzàlez, A. Sobral, T. Bouwmans, C. Tu, and H. Zahzah, “Human pose estimation from monocular images: a comprehensive survey”, Sensors, Basel, Switzerland, vol. 16, 2016.

[10] G. Ning, P. Liu, X. Fan and C. Zhan, “A top-down approach to articulated human pose estimation and tracking”, ECCV Workshops, 2018.

[11] A. Gupta, T. Chen, F. Chen, and D. Kimber, “Systems and methods for human body pose estimation”, U.S. patent, 7,925,081 B2, 2011.

[12] H. Sidenbladh, M. Black, and D. Fleet, “Stochastic tracking of 3D human figures using 2D image motion”, Proc 6th European Conf. Computer Vision, 2000.

[13] A. Agarwal and B. Triggs, “3D human pose from silhouettes by relevance vector regression”, Intl Conf. on Computer Vision & Pattern Recogn.pp.882–888, 2004.

[14] M. Li, Z. Zhou, J. Li and X. Liu, “Bottom-up pose estimation of multiple person with bounding box constraint”, 24th Intl. Conf. Pattern Recogn.,2018.

[15] Z. Cao, T. Simon, S. Wei, and Y. Sheikh, “OpenPose: realtime multi-person 2D pose estimation using part affinity fields”, Proc. 30th IEEE Conf. Computer Vision and Pattern Recogn,2017.

[16] A. Kendall, M. Grimes, R. Cipolla, “PoseNet: a convolutional network for real-time 6- DOF camera relocalization”, IEEE Intl. Conf. Computer Vision, 2015.

[17] S. Kreiss, L. Bertoni, and A. Alahi, “PifPaf: composite fields for human pose estimation”, IEEE Conf. Computer Vision and Pattern Recogn, 2019. [16] P. Dar, “AI guardman – a machine learning application that uses pose estimation to detect shoplifters”. [Online]. Available: https://www.analyticsvidhya.com/blog/2018/06/ai-guardman-machine-learningapplication-estimates-poses-detect-shoplifters/

[18] D. Mehta, O. Sotnychenko, F. Mueller and W. Xu, “XNect: real-time multi-person 3D human pose estimation with a single RGB camera”, ECCV, 2019.

[19] A. Lai, B. Reddy and B. Vlijmen, “Yog.ai: deep learning for yoga”. [Online]. Available: http://cs230.stanford.edu/projects_winter_2019/reports/15813480.pdf

[20] M. Dantone, J. Gall, C. Leistner, “Human pose estimation using body parts dependent joint regressors”, Proc. IEEE Conf. Computer Vision Pattern Recogn., 2013.

[21] S. Haque, A. Rabby, M. Laboni, N. Neehal, and S. Hossain, “ExNET: deep neural network for exercise pose detection”, Recent Trends in Image Process. and Pattern Recog., 2019

[22] J. Carreira, P. Agrawal, K. Fragkiadaki and J. Malik, "Human Pose Estimation with Iterative Error Feedback," 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016, pp. 4733-4742, doi: 10.1109/CVPR.2016.512.

[23] Y. Agrawal, Y. Shah and A. Sharma, "Implementation of Machine Learning Technique for Identification of Yoga Poses," 2020 IEEE 9th International Conference on Communication Systems and Network Technologies (CSNT), 2020, pp. 40-43, doi: 10.1109/CSNT48778.2020.9115758.
https://ieeexplore.ieee.org/abstract/document/9115758

[24] S. Liaqat, K. Dashtipour, K. Arshad, K. Assaleh and N. Ramzan, "A Hybrid Posture Detection Framework: Integrating Machine Learning and Deep Neural Networks," in IEEE Sensors Journal, vol. 21, no. 7, pp. 9515-9522, 1 April1, 2021, doi: 10.1109/JSEN.2021.3055898.
https://ieeexplore.ieee.org/abstract/document/9343347

[25] Niharika Pandit (2020) Yoga Poses Dataset (Version 1) [Data file] Retrieved from https://www.kaggle.com/niharika41298/yoga-poses-dataset