NVEO
Natural Volatiles & Essential Oils

Machine-Learning Technique For Camera-Based Monitoring And Evaluation Of Yoga Posture

Ramya HR[1] , Vikram Nag R C[2] , Pramod Pavan Krishna[3] , Disha Kulkarni[4]

[1]Assistant Professor, Dept. of Electronics and Telecommunication Engineering, Ramaiah Institute of Technology,Bangalore, India

[2,3,4] UG Students,Dept. of Electronics and Telecommunication Engineering, Ramaiah Institute of Technology Bangalore, India

ABSTRACT:

Human pose estimation is a long-standing issue in computer vision that has presented numerous difficulties in the past. Robotics, Video surveillance, biometrics, Augmented reality, assisted living, at-home health monitoring, and other industries benefit from analysing human actions. This project establishes the groundwork for developing a feedback-evaluation system by combining a machine learning-based model with deep learning methodologies to categorise yoga positions in both pre-recorded and real-time footage.

Our project is a browser based system divided into 3 parts - Creating the dataset, Training the model, Classification of pose for 3 Yoga asanas using the poseNet and neuralNetwork models which are present in the ml5.js library. PoseNet is a pre-trained model which detects 17 keypoints on the human body when input is provided through pre-recorded videos or in real-time using a webcam. These keypoints are stored in the form of x,y locations and used to train the overall system using the neuralNetwork model. The classification process implements both poseNet and neuralNetwork models simultaneously. The user's pose is compared to the expert's pose, and the angle difference between various body joints are determined.

Keywords: Human pose estimation, yoga, ml5.js, PoseNet, neuralNetwork, keypoints

1. INTRODUCTION

Yoga as a discipline has a plethora of advantages that benefit us both physically and emotionally. With ageing and accidents, individuals are more prone to musculoskeletal problems. To avoid this, some type of physical activity is required. Yoga, which is both a physical and spiritual practice, has garnered a lot of traction among medical researchers. Yoga has the power to treat various ailments totally without the use of any or many medications, as well as improve physical and mental wellness. Positive body image intervention, cardiac rehabilitation, mental disease, and other medical applications of yoga have created a large body of literature.Yoga consists of a variety of asanas, or physical static positions. Because yoga contains a complicated combination of postures, applying pose estimation to it is difficult. Moreover, when the asana requires horizontal body posture or both legs overlap each other, several state-of-the-art solutions fall short. Yoga can be a useful strategy for coping with the uncertainty and solitude of the lockdown, as well as maintaining physical health. To rebuild strength and balance during and after lockdown, a rising number of practitioners have turned to online yoga programmes. It demonstrates the increased significance of yoga in post-COVID-19 health. The Covid-19 issue was not only a physical one; it also had long-term effects on mental health, with many people suffering from psychological distress, despair, and anxiety as a result of the pandemic's imposed limits and losses. Many people dealing with such difficulties can benefit from yoga. As a result, a solid model that can assist popularise self-instructed yoga methods is required[1].

Yoga and its benefits:

Yoga's popularity has risen rapidly in recent years. Yoga has several benefits that have been shown by study, and celebrities and healthcare experts are already practicing and advocating it. While some deem yoga as just another fad and associate it with new age mysticism, others vouch for how amazing this form of exercise feels. What they don't realise is that what they think of as just another workout will benefit them in unexpected ways.

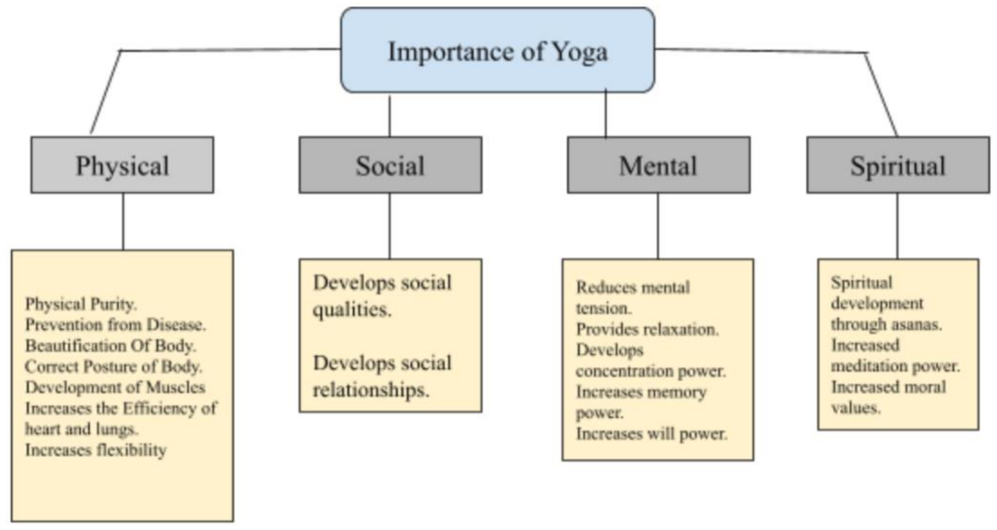

Figure 1.1 Importance of Yoga

2. METHODOLOGY:

To create a self-training system, yoga activity detection using a features-based approach[1] was used. It makes use of a Kinect to extract the person's body contour and create a body map. To obtain a descriptor for the human position, a star skeleton was employed for quick skeletonization. Delegate features, like a human skeleton, must be extracted in order to describe human postures. Various skeletonization strategies, such as thinning and distance transformation, have been documented in the literature. These methods, nonetheless, have a high computational cost and are susceptible to noise.

The end-to-end deep learning-based framework[2] removes the requirement for handcrafted features, allowing the model to be retrained with new data to include new asanas. It's used on the time-distributed CNN layer to find patterns between keypoints in a single frame, and on the LSTM to remember patterns observed in recent frames. The results make the system even more resilient by lowering the error due to false keypoint detection by using LSTM for prior frame memory and polling for denoising.

People in close proximity to each other, mutual occlusions, and limited visibility are all obstacles when calculating poses in highly populated areas[4]. A single-person pose estimator employing the ResNet50 network as the backbone is one of the methods used to optimize pose estimation for crowded photos. The technique is a top-down, two-stage strategy that first localises each individual before performing a single-person pose estimate for each forecast. The paper also introduces Occlusion Net and Occlusion Net Cross Branch, two occlusion detection networks. After two transposed convolutions, the Occlusion Net splits, allowing the prior layers to learn a joint representation. After one transposed convolution, the Occlusion Net Cross Branch divides. Per pose, the Occlusion Detection Networks generate two sets of heatmaps. One heatmap depicts keypoints that are visible, while the other depicts keypoints that are hidden.

2.1 Key Point Detection Method:
PoseNet is a deep learning structure similar to OpenPose that is used to identify human poses in photos or video sequences by identifying joint locations in the human body. These keypoints or joint regions are listed by "Part ID," which is a confidence score that ranges from 0.0 to 1.0, with 1.0 being the greatest. The execution of the PoseNet model varies depending on the gadget and yield step [14]. Because the PoseNet model is independent of the image size, it can predict current scenarios in the size of the original image, regardless of whether it has been downscaled.

2.2 Creating the dataset:
The best part about using pre-trained models is that a huge portion of the work is already done. There is no need to create, train and fine-tune our model. We just load it, and it is ready to be implemented.We will be using the poseNet model for pose estimation and neuralNetwork model for training and pose classification, both of which are present in the ml5.js library.Starting videoCapture on canvasIn order to implement majority of the operations, the p5 library provides a canvas on which we can perform actions such as display images/videos, draw shapes, print texts etc.,In our project we make use of such a canvas such that it covers the entire boundary of the live captured video, display the Keypoints, draw the Skeleton, print the identified Asana Label and it's confidence score.

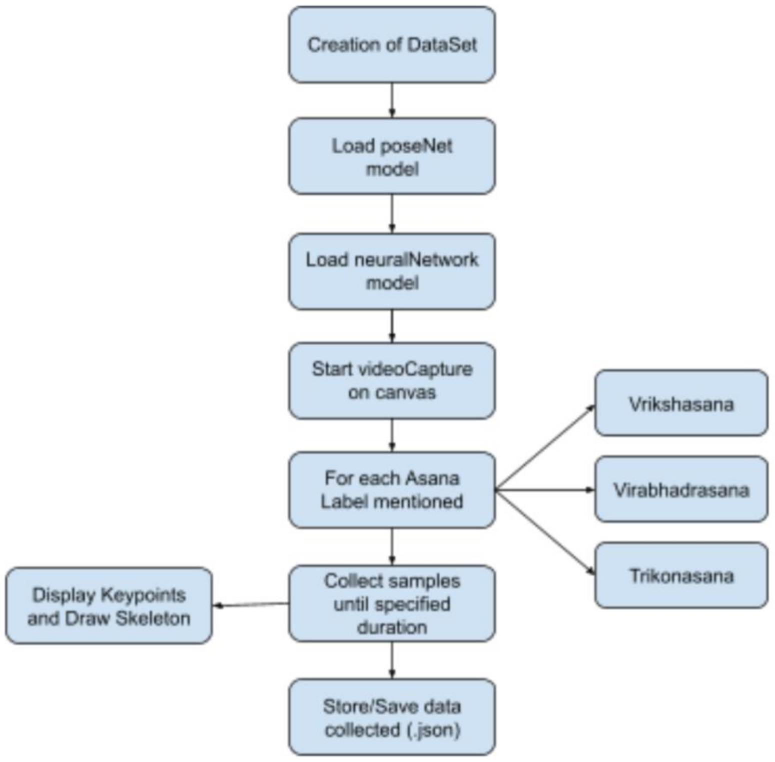
Creation of DataSet
Load poseNet model
Load neuralNetwork model
Start videoCapture on canvas
For each Asana Label mentioned
Vrikshasana
Virabhadrasana
Trikonasana
Collect samples until specified duration
Display Keypoints and Draw Skeleton
Store/Save data collected (.json)

Figure 2.2 Flowchart for Creating Dataset, Loading the pre-trained model.

2.3 Training the model:

The .json file comprising the location of keypoints under each label that is generated in the previous step is loaded into the training Model. Because the target variables or labels have a wide range of values or keypoints, high error gradient values may occur, causing weight values to fluctuate dramatically, making the learning process unstable. Thus In order to use neural network models, you must first normalize the data or scale the input and output variables.

The neuralNetwork takes the values inside the file under each Label and trains the model for a specified number of epochs. An epoch is a unit of time used to train a neural network with all of the training data for a single cycle. We use all of the data exactly once in an epoch. A forward pass and a backward pass are combined to make one pass: An epoch is made up of one or more batches in which we train the neural network using a portion of the dataset. In this project we train the model for 50 epochs to get the best possible minimum loss at the end. Once training is finished, the model generates 3 files - model, metadata and weights, which are essential for the next process that is classification.

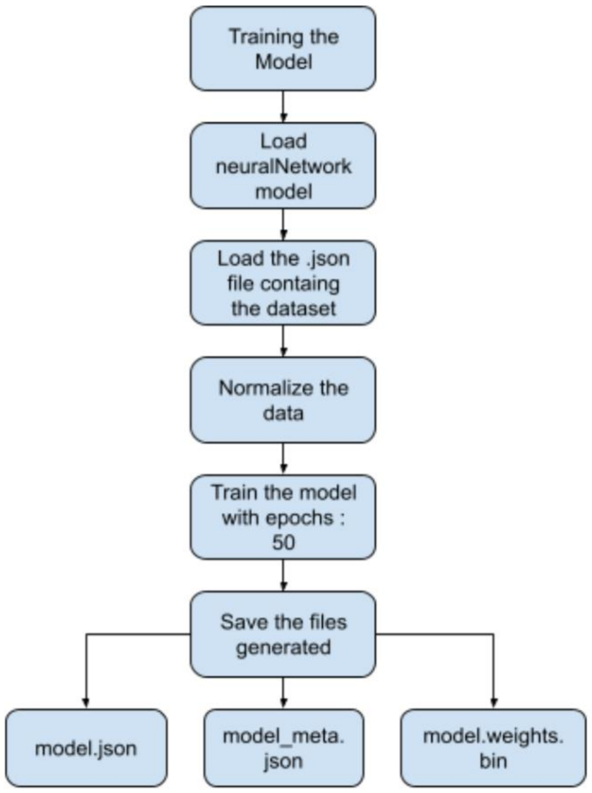

Figure 2.3 - Model Training

2.4 Classification of Yoga Pose:

For real-time classification of yoga poses, the model uses poseNet and neural Network models concurrently to determine the pose identified. All of the trained data which was downloaded is loaded into the model. Live video Capture input from the webcam is started and when a pose is detected, the poseNet model detects the keypoints and the skeleton which is drawn on the canvas displaying the webcam captured live video. When the person does a yoga pose, the neuralNetwork model identifies and classifies
the pose (in our project into one of the 3 poses - Vrikshasana, Virabhadrasana, Trikonasana). If a pose is not detected, the model waits until the callback function timeouts and upon detection of the pose, classification process is initiated. The identified pose is dynamically displayed along with it’s confidence score of identification which gives us the accuracy of the pose.

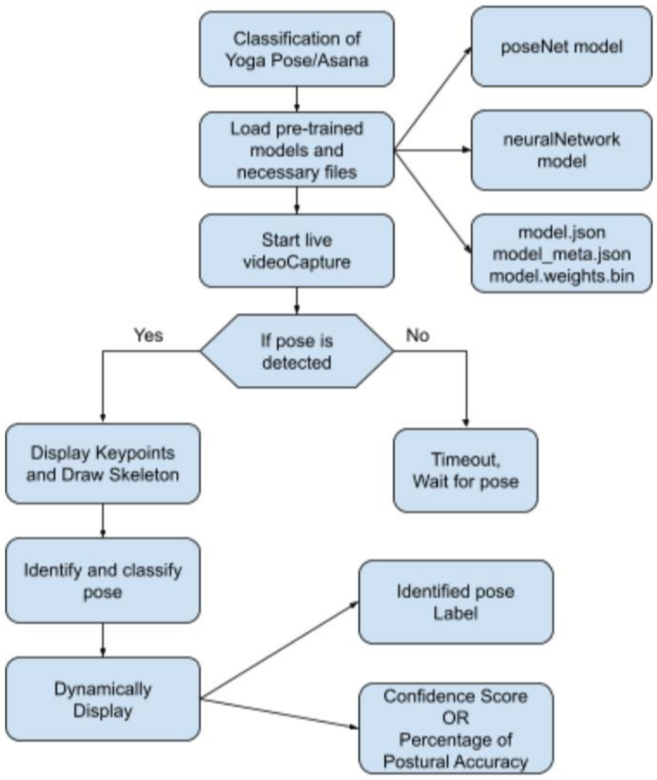
Classification of Yoga Pose/Asana
Load pre-trained models and necessary files
poseNet model
neuralNetwork model
model.json model_meta.json model.weights.bin
Start live videoCapture
If pose is detected
Yes
No
Display Keypoints and Draw Skeleton
Timeout, Wait for pose
Identify and classify pose
Dynamically Display
Identified pose Label
Confidence Score OR Percentage of Postural Accuracy

Figure 2.4 Classification of Yoga Asana

3. Workflow:

We start the training process by first setting up a canvas for the model to display the training performance once the training is complete. The neuralNetwork model is loaded into the variable “brain”, with the same parameter values as specified while creating the data. The .json file generated in the previous step, is loaded into the training model as a parameter to the loadData() - loads the data to neuralNetwork.data.data.raw and sets neuralNetwork.data.data.raw to the array specified in the "data" property of the incoming .json file. The other parameter is a user defined callback function, dataReady() under which normalization of the data is carried out. The normalizeData() function normalizes the data on a scale from 0 to 1. The data being normalized are part of the NeuralNetowrksData class which can be accessed in neuralNetowrks.data.data.raw. After normalization, the data is in proper format and is ready to be trained. The function train() uses data in the neuralNetowrks.data.training array to train the model. If an object of options is given, then optionsOrCallback will be an object where we can specify the batchSize and epochs. If a callback function is given here then this will be a callback that will be called when the training is finished. The parameters given for the train() function are, the epochs which is declared to be 50 and a callback function finished(). The training performance window is displayed on the canvas and shows the amount of loss reduction with each epoch until the model is trained for the specified number of epochs. Once training is complete, the callback function finished() is called and the model, metadata and weights files generated are saved onto the user’s computer using the save() function which downloads the model to a .json file and a model.weights.bin binary file in the downloads folder.

4. Results and Discussions:

The final output of our project is as shown in figure 4.1. The model detects the keypoints, draws the skeleton, identifies the yoga pose and displays the classified pose label and it’s confidence score (percentage of postural accuracy) simultaneously. Even if all keypoints are not detected, the model identifies the uniqueness of the pose and classifies the pose with good accuracy.

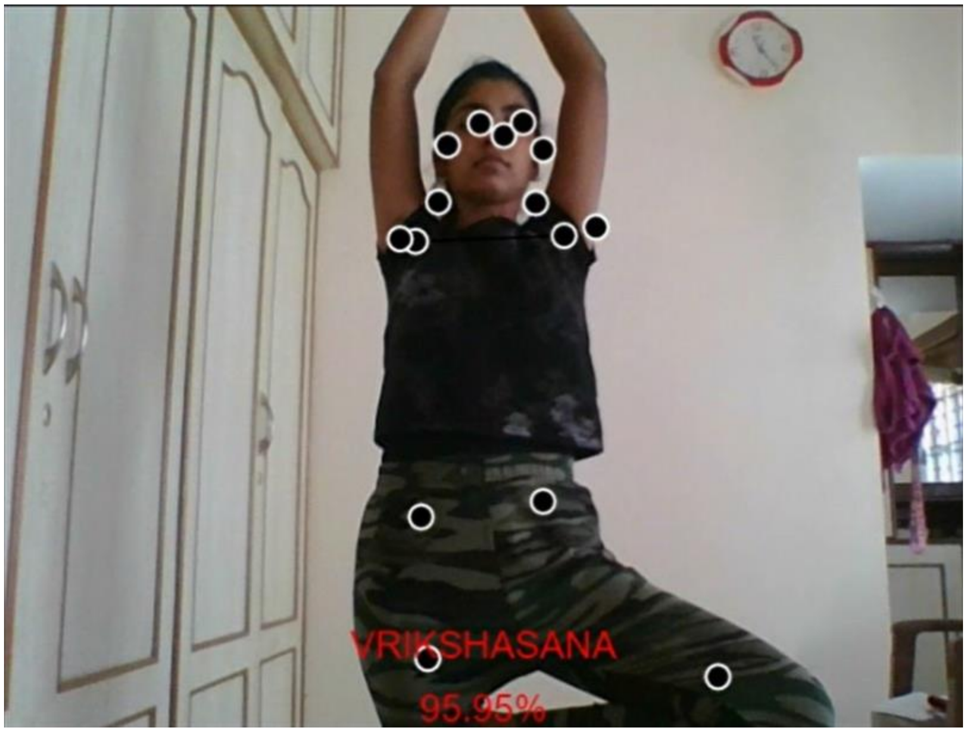

Figure 4.1 Final Output

5. Conclusion:

Three yoga asanas are currently classified using the proposed models. Because there are so many different yoga asanas, developing a pose estimate model for multipose implementation that works for all of them is a difficult task. More yoga positions performed by individuals not just in indoor but also outdoor settings can be added to the dataset. PoseNet pose estimation, which may not perform well in circumstances of many poses or overlap between body parts, is critical to the models' performance. This method can be deployed with a portable device for self-training and real-time forecasts. This project displays activity recognition in a real-world setting. Pose identification in jobs such as sports, surveillance, and healthcare can all benefit from a similar technique. Multi-person posture estimation is a whole different subject with a lot of room for investigation. There are several situations when a single person pose estimation would be insufficient; for example, pose estimation in crowded environments would require tracking and detecting the pose of each individual. Many considerations, such as background, lighting, overlapping figures, and so on, would make estimating multi-person poses even more difficult.

ACKNOWLEDGEMENTS Author thanks the research scholars for making this work successful.

REFERENCES

[1] Yoga posture recognition for self-training. Chen HT, He YZ, Hsu CC et al (2014)

[2] Yadav, S., Amitojdeep Singh, A. Gupta and J. L. Raheja. "Real-time Yoga recognition using deep learning." Neural Computing and Applications 31 (2019): 9349-9361.

[3] Xiao, Bin & Wu, Haiping & Wei, Yichen. (2018). Simple Baselines for Human Pose Estimation and Tracking. 10.1007/978-3-030-01231-1_29.

[4] T. Golda, T. Kalb, A. Schumann and J. Beyerer, "Human Pose Estimation for Real-World Crowded Scenarios," 2019 16th IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS), 2019, pp. 1-8, doi: 10.1109/AVSS.2019.8909823.

[5] ScholarWorks, Sjsu, Shruti Kothari and R. Chun. "Yoga Pose Classification Using Deep Learning." (2020).

[6] Jose, Josvin & Shailesh, S. (2021). Yoga Asana Identification: A Deep Learning Approach. IOP Conference Series: Materials Science and Engineering. 1110. 012002. 10.1088/1757-899X/1110/1/012002.

[7] Yoga posture recognition for self-training. Chen HT, He YZ, Hsu CC et al (2014)

8] Wang, Jin, L. Yu, K. Lai and Xuejie Zhang. "Dimensional Sentiment Analysis Using a Regional CNN-LSTM Model." ACL (2016).

[9] Cao, Zhe, Gines Hidalgo, Tomas Simon, Shih-En Wei and Yaser Sheikh. "OpenPose: Realtime Multi-Person 2D Pose Estimation Using Part Affinity Fields." IEEE Transactions on Pattern Analysis and Machine Intelligence 43 (2021): 172-186.

[10] Yadav, S., Amitojdeep Singh, A. Gupta and J. L. Raheja. "Real-time Yoga recognition using deep learning." Neural Computing and Applications 31 (2019): 9349-9361.

[11] Carreira, João, Pulkit Agrawal, Katerina Fragkiadaki and Jitendra Malik. "Human Pose Estimation with Iterative Error Feedback." 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2016): 4733-4742.

[12] Newell, Alejandro & Yang, Kaiyu & Deng, Jia. (2016). Stacked Hourglass Networks for Human Pose Estimation. 9912. 483-499. 10.1007/978-3-319-46484-8_29.

[13] Wei, Shih-En, V. Ramakrishna, T. Kanade and Yaser Sheikh. "Convolutional Pose Machines." 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2016): 4724-4732.

[14] Xiao, Bin & Wu, Haiping & Wei, Yichen. (2018). Simple Baselines for Human Pose Estimation and Tracking. 10.1007/978-3-030-01231-1_29.

[15] Fang, Hao-Shu & Xie, Shuqin & Tai, Yu-Wing & Lu, Cewu. (2017). RMPE: Regional Multi-person Pose Estimation. 2353-2362. 10.1109/ICCV.2017.256.

[16]T. Golda, T. Kalb, A. Schumann and J. Beyerer, "Human Pose Estimation for Real-World Crowded Scenarios," 2019 16th IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS), 2019, pp. 1-8, doi: 10.1109/AVSS.2019.8909823.