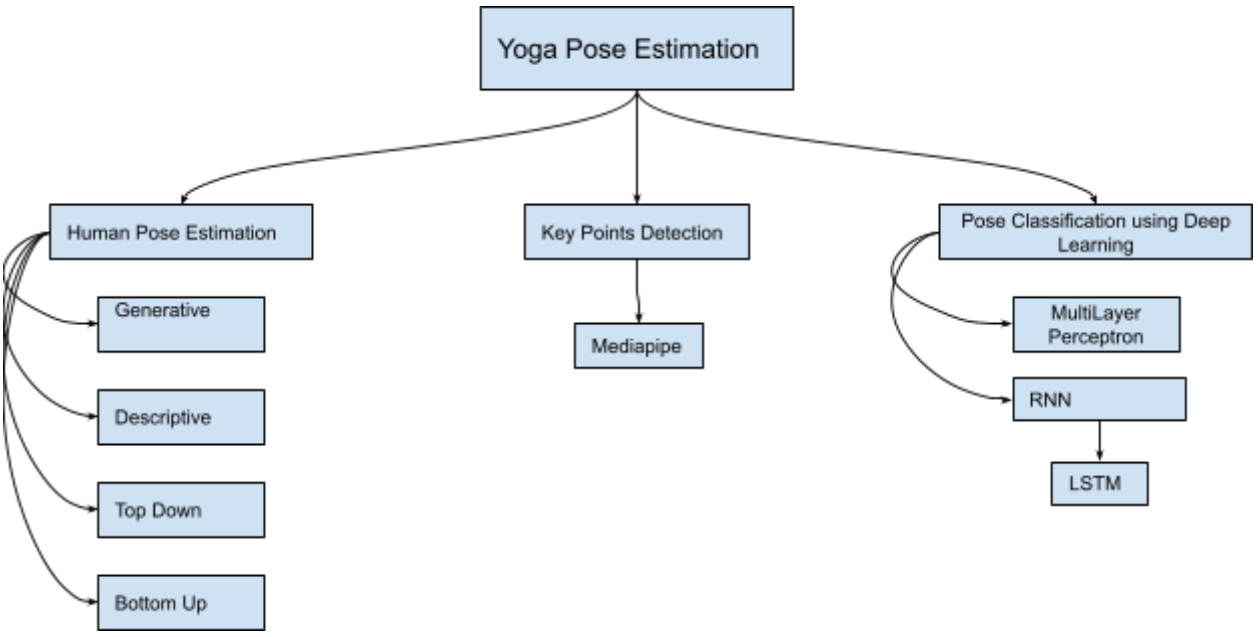
Content

Abstract

This report presents a method for reliably recognizing various Yoga asanas using deep learning techniques. At (https://archive.org/details/YogaVidCollected), a dataset of six Yoga asanas with 88 videos in total (Bhujangasana, Padmasana, Tadasana, Trikonasana, and Vrikshasana) is freely available, we have used this following dataset for our project. For Yoga recognition on real-time videos and captured videos, a deep learning model using MLP Classifier and long short-term memory (LSTM) will be proposed. This research aims to learn more about the various approaches to yoga pose classification and gain insight into the following: What is pose estimation, and how does it work? What is deep learning, and how does it work? How can deep learning be used to classify yoga poses in real time? The developed model will then be proposed for IoT devices [1]. References from conference proceedings, published papers, technical reports, and journals are used in this research.



Introduction

Almost all vision-based registration systems used markers until the early 2000s. Then, in a matter of weeks, a slew of markerless techniques appeared. Finally, in the late 2000s, Simultaneous Localization and Mapping (SLAM) emerged, which, as a follow-up to structure from motion techniques (often employed in off-line compositing for the film industry), permits the removal of a scene model. Although vision-based registration remains a challenging topic, mature solutions may

now be offered to users. In the meantime, various open source software libraries and software development toolkits have been launched, providing developers with simple alternatives. As a result, rapid prototyping of AR systems is possible. Pose estimation is an issue that originated in photogrammetry and is referred to as space resection. "Pose estimation is determining the location and orientation of the camera given a collection of correspondences between 3D features and their projections in the pictures plane," according to a simple definition. People have been living in the information era in recent years. There are several ways to access the Internet in everyday life, including computers, smartphones, smart TVs, tablets, and some smart cars, all of which are ready to make our lives easier and more pleasant. The new gadgets can run any programmes in a much better and more efficient manner for doing various activities like turning on or off the device, or sending alarms via built-in or external sensors. Smart objects can be used to describe these objects. Similarly, smart systems can be offered via the Internet of Things (IoT), which allows for the creation of either a large or small network in order to obtain collective intelligence through the processing of object knowledge.

The Internet of Things (IoT) can be used to provide a variety of services that benefit inhabitants. In other words, smart homes may be created by employing the Internet of Things (IoT), which has the capability of controlling and automating specific aspects of our homes such as lighting, doors, refrigerators, etc. Using Computer vision face detection, recognition, as well as body detection can be provided and are extremely fascinating applications for the Internet of Things. Estimating human pose is a difficult challenge in computer vision. Fitness is an application which has gathered a lot of interest from many researchers for detection of pose. Yoga is a form of exercise with sophisticated postures that originated in India. The issue with yoga postures, like any other exercise, is that it must be done correctly, as any incorrect posture can be dangerous. As a result, an instructor must monitor the practice and help the student improve his or her posture. Because not everyone has access to or the financial means to employ a yoga instructor, an artificial intelligence-based app could help people improve their technique by identifying yoga postures and providing personalised feedback.

No.	**KeyPoint**	**No.**	**KeyPoint**	**No.**	**KeyPoint**
0	nose	11	leftShoulder	22	rightThumb

1	left_eye_inner	12	rightShoulder	23	leftHip
2	left_eye	13	leftElbow	24	rightHip
3	left_eye_outer	14	rightElbow	25	leftKnee
4	right_eye_inner	15	leftWrist	26	rightKnee
5	right_eye	16	rightWrist	27	leftAnkle
6	right_eye_outer	17	leftPinky	28	rightAnkle
7	left_ear	18	rightPinky	29	leftHeel
8	right_ear	19	leftIndex	30	rightHeel
9	mouth_left	20	rightIndex	31	left_foot_index
10	mouth_right	21	leftThumb	32	right_food_index

Pose Landmarks in MediaPipe

It is useful to process data with *Recurrent neural networks (RNNs)*. Sequential data have been studied in a variety of fields. Since an activity is a series of activities, recurrent neural networks can be used to process sequential data. Because long-term memory networks (LSTMs) can store information for a long time, they are the most popular RNN architecture. The method suggested will identify the four most common asanas in real time as well as from video recordings. The asanas are Padmasana, Tadasana, Trikonasana, and Vrikshasana. We've also added a new class

Problem Statement

Objective of our research is to develop an algorithm which can extract key points or landmarks of the body that is built upon OpenPose. Through the correlation of the visualisation points a body map is created which shall predict the particular pose being performed by a person.

Literature Review

In [2] a method for estimating a series of human positions from videos is proposed. Compared to the widely used graph optimization techniques, they formulated this challenge as a two staged and tree based problem. It was a challenge to find a solution that maximized the spatial limitations and maintain the temporal consistency between frames. They proposed two essential ideas. The first notion was Abstraction, in which they introduced a new concept, abstract body parts, unlike the

usual tree representation. Combining symmetrical parts of the body conceptually. Thus, they avoided using simple cycles in the formulation by taking advantage of the symmetry of human body parts. In order to maintain temporal consistency, the second notion was to use Association. As each abstract body part was treated independently, the graph did not contain simple cycles.

In [3] the goal was to find out that in sports like golf, the player's posture and his or her motions are formed that can have a significant impact on the game's performance. Various studies and services have attempted to examine golf swings with human poses which are relatively tough to detect through the naked eye. A marker-less 3D human pose estimation technology was used to estimate human posture accurately. They presented an algorithm for estimating human posture that uses different types of random forests to improve results for sports analysis. To determine the ultimate position of anatomic joints, random verification forests assess and optimise the votes after using random regression forests to locate joints in the athlete's anatomy. As a result of experimental results, the suggested algorithm improves not only the performance but also the performance of the proposed algorithm.

In [4] patient's joint detection and posture provide numeric data that is clinically quite informative. Wearable devices have been used in several studies to collect data, but have also posed problems as a result of prolonged wear of physical sensors. For patients who cannot wear these devices due to chances of injuries at the wrists, they can have the provision of video-based joint tracking. PatientPose which is a modification of Caffe-Heatmap for pose estimation in a clinical environment. Apart from the existing pose estimation framework, it includes three elements that enable more accurate tracking of posture:

1) First step is to account for the changes in scene lighting found in hospitals;

2) Convolutional neural network models on each patient to capture the wide range of postures.

3) A Kalman filter

In [5] a single network method for body pose is proposed. The approach has consistent output. In both speed and accuracy, their approach significantly improves on the only known whole-body pose estimation method. Challenges included failure of global cases where a portion of the present person was outside of the picture boundaries.

In [6] a proposal of a new method for pose estimation for activities such as self-driving cars and delivery robots is mentioned. To form full human poses, PIFPAF is used to locate body parts and a Part Association Field (PAF) is used to integrate them. The method they use improves performance at low resolution

1) This method field PAF encodes specific information and

2) The use of a Laplace loss for regressions incorporating uncertainty. A fully convolutional, singleshot, box-free architecture is used.

As a result, they perform well with the existing method COCO keypoint task.

In [8] a method to correctly identify the various Yoga poses making use of deep learning models is mentioned. A dataset of six Yoga poses has been created and implemented using 15 people. The proposed model uses a deep learning technique that combines convolutional neural networks with long short-term memory and this is done so that yoga asanas can be correctly recognised from real-time videos.

Proposed Methodology

Our method seeks to recognise the user's Yoga asanas in real-time and recorded video automatically. The procedure can be broken down into three key steps. The first step is to collect data, which can be a real-time operation running in parallel with detection or previously recorded films. Second, MediaPipe is used to determine where the joints are located. Our model receives the keypoints and uses LSTM to uncover patterns and analyse their change over time.

Workflow

- Pose extraction using Mediapipe

  This is the first step in building our yoga pose model. After the dataset is collected from the publicly available source. We use the Mediapipe library to extract the landmarks from the videos. Each landmark consists of four values. The first three values are x, y, and z coordinates, and the last value is visible, which is used to define the pose landmark. Ne

  xt, we have created a function that will iterate over a folder of videos, extract the landmark points. After that, we add a label column and save it in a CSV file. The frame per second(fps) is set automatically to find the optimal performance. This process is repeated for all four folders one by one. So, in the end, we have four CSV files for each asana which contain all the four points for each landmark specified by a class label. Then all the four CSV files are combined into a single file. There are 134 columns in which there is an unnamed column created while doing this operation, so we drop that column. The class labels are given in categorical format, so with the help of the label encoder function, each

asana gets a numerical value. Then the feature and the target variables were defined after that dataset was split into training and testing in the ratio of 80:20.

Snapshots of Dataset:

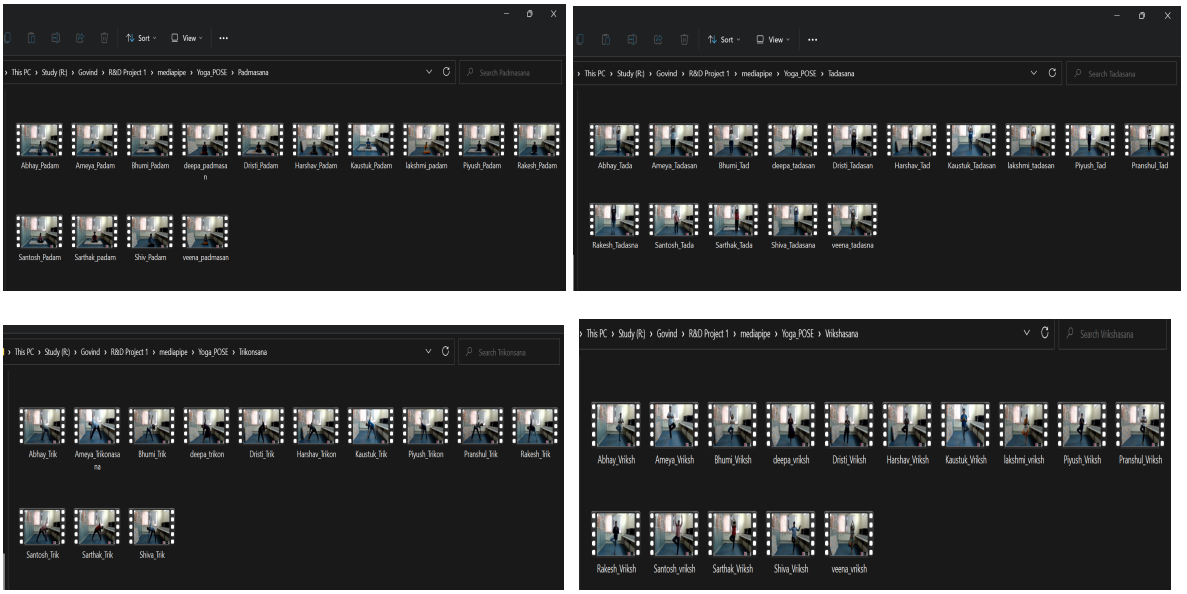


Figure - Snapshot of Datasets

- Model

The first model was LSTM; it takes input as no of features in training and 128 no of the layer. After that, one LSTM and one dense layer are applied are 128,32 with a dropout of 0.2 to prevent the overfitting. The last dense contains four as the number of classes, and the SoftMax activation function was used because it is a multi-class classification problem. Then the model is compiling, the LSTM model is run for the 20 epochs and then predictions on the test set. At last, the joblib library was used to save the model, and then we made the prediction on the real-time video or recorded videos.

To construct the MLP classifier Sklearn library is used. The inputs that were given to the MLP classifier were hidden layer set to 150×100×50 where the 150, 100 are the numbers of the hidden layer and 50 is the number of nodes, the number of epochs is set to 100, relu is used for the activation function for the hidden layer, and the weight optimization algorithm which is solver is set to Adam.

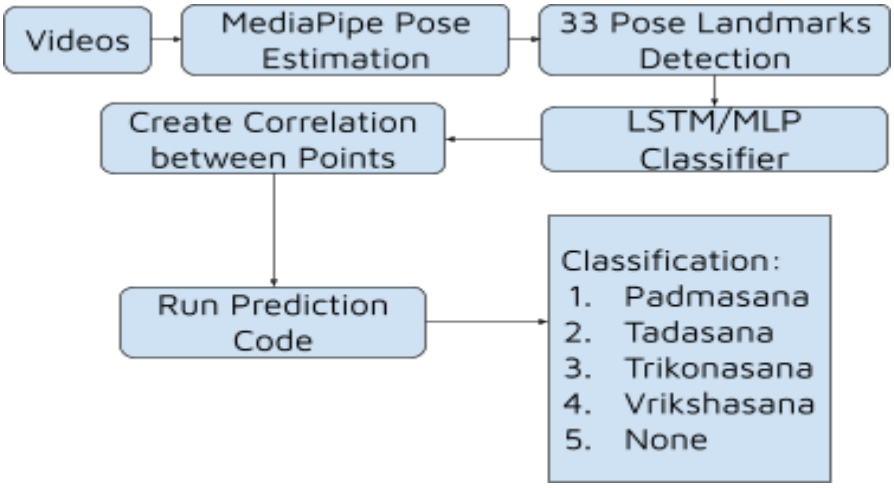


Figure 2- Workflow Diagram

- Training

Once the LSTM network was made after that the model is compiled using TensorFlow and keras as backend. The loss that we have used was sparse categorical cross-entropy, the optimizer is set according to Adam having a learning rate of 1e-3 and having decay of 1e-5. The deep learning model was trained on i5 Intel core processor, and 8 GB Ram. It takes around 3 to 5 hours to do all the tasks from feature extraction to model training. The loss curve for the MLP classifier and the LSTM is given below.

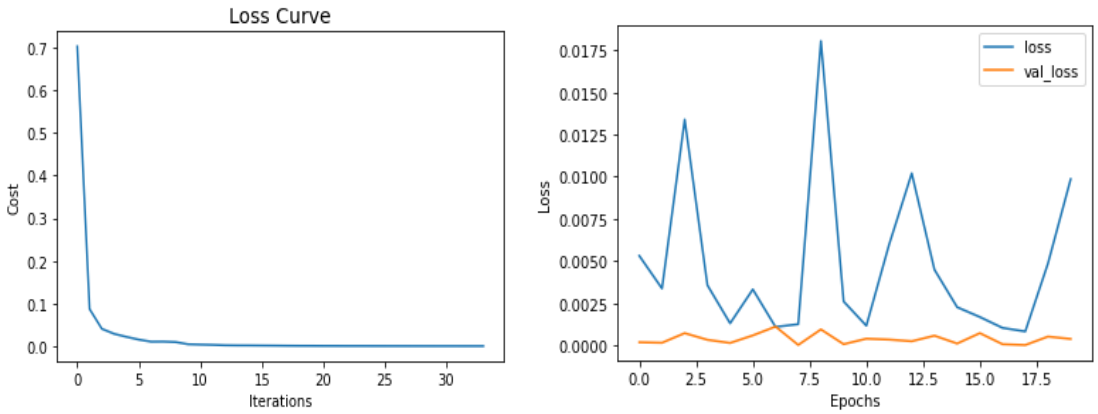


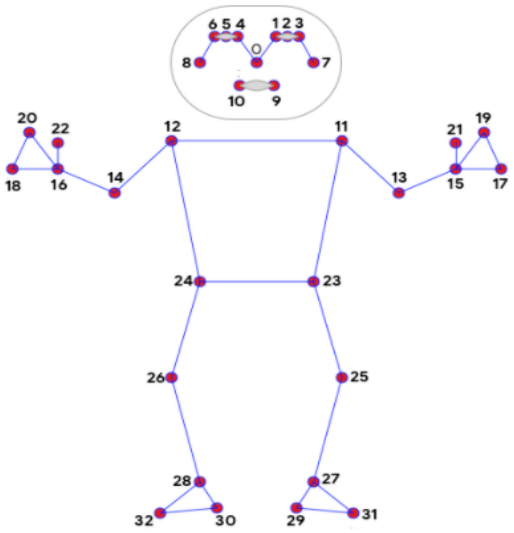
Figure 3- Loss plot for MLP and LSTM model with Mediapipe respectively

If we look at the plot of MLP classifier after every iteration the cost is reduce and when it crosses around 30 iteration it is almost is in in between 0.1 to 0 and for the LSTM model the training loss is following the zigzag pattern whereas the validation was decreasing when the epoch count increasing. So, we can conclude that in both the graphs the loss is decreasing which indicates a better model. But Comparatively MLP Classifier has an edge over LSTM.

Technology

- Pose extraction:

  This is the first stage of our project, and it employs the MediPipe library. This stage occurs offline in the case of recorded movies and online in the case of real-time forecasts, with the proposed model being fed posture landmarks via camera input. MediaPipe is an open platform multi-person feature point detection library that detects feature points on the body, hand, and face all at once. The ears, eyes, nose, neck, shoulders, knee, and other pose landmarks are tracked by the MediaPipe. Each landmark position for each person detected in the image/frame is included in the csv output for each frame of a movie. The pose extraction was done at the MediaPipe for the finest results.



0. nose
1. left_eye_inner
2. left_eye
3. left_eye_outer
4. right_eye_inner
5. right_eye
6. right_eye_outer
7. left_ear
8. right_ear
9. mouth_left
10. mouth_right
11. left_shoulder
12. right_shoulder
13. left_elbow
14. right_elbow
15. left_wrist
16. right_wrist
17. left_pinky
18. right_pinky
19. left_index
20. right_index
21. left_thumb
22. right_thumb
23. left_hip
24. right_hip
25. left_knee
26. right_knee
27. left_ankle
28. right_ankle
29. left_heel
30. right_heel
31. left_foot_index
32. right_foot_index

Figure 4- Source: Source: https://google.github.io/mediapipe/solutions/pose.html

- Model**.**

  LSTM is the Deep Learning model utilised in this case. For time series jobs, LSTM is used. We'll go over how to create a Long Short-Term Memory network model (LSTM) for the yoga position estimation dataset in this part. LSTM network models are recurrent neural network models that can remember and learn extended sequences of input data. They're designed to deal with data that consists of 200 to 400 time steps in lengthy sequences. They could be a suitable match for this issue. The model learns to extract characteristics from observation sequences and transfer internal features to different activity groups.

  MLPs are neural networks that only have one input and output layer. It's possible to have one or more hidden layers. Each node in a layer is connected to every other node in another, resulting in a network that is entirely connected. A fully linked network is the cornerstone for deep learning. MLP is commonly used for supervised classification, which entails assigning a label or class to input data.

- Python Tools.

  1) Pandas:

  A fast and efficient Python library for data manipulation and indexing. pandas is a tool used for reading and writing data in different formats, for example, CSV, text-file, etc. In our project we have used pandas for performing read/write operations on our dataset and performing EDA

  2) Scikit-learn:

  Scikit-learn is a python library which provides dozens pre-build machine learning algorithms like logistic regression, linear regression, random forest etc. In our project, we have implemented four built-in machine-learning algorithms to classify the poses.

  3)OpenCV: OpenCV is an open-source library that provides a common infrastructure for performing computer-vision tasks along with machine learning and is capable of working alongside numerous algorithms and frameworks. In our case we are using OpenCV with MediaPipe for pose estimation, pose detection and extracting the 3D coordinates of the pose.

Result

1. **MLP Classifier:**

   Training Accuracy - 0.9997

   Validation Accuracy - 1

   Test Accuracy - 0.9927

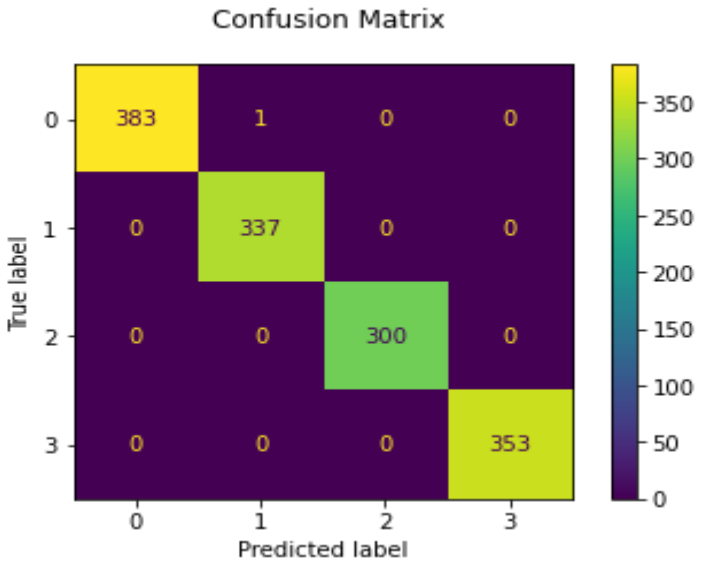


Figure - Confusion Matrix for MLP Classifier

The training accuracy of the MLP Classifier model implemented is at 0.99. Further slight decrease has been seen in Validation and Test Accuracy. But still the results for this model are good. Confusion matrix shown above for MLP Classifier shows that Label 1(Tadasana), Label 2(Trikonasana) and Label 3(Vrikshasna) have been classified correctly except Label 0(Padmasana). Out of 383 frames for Padmasana, just 1 have been misclassified as Tadasana.

2. **LSTM:**

Training Accuracy - 0.997

Validation Accuracy - 1

Test Accuracy - 0.986

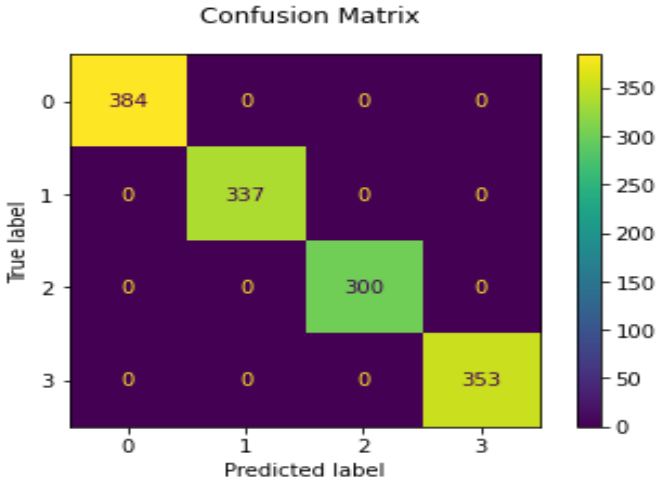


Figure - Confusion Matrix for LSTM Model

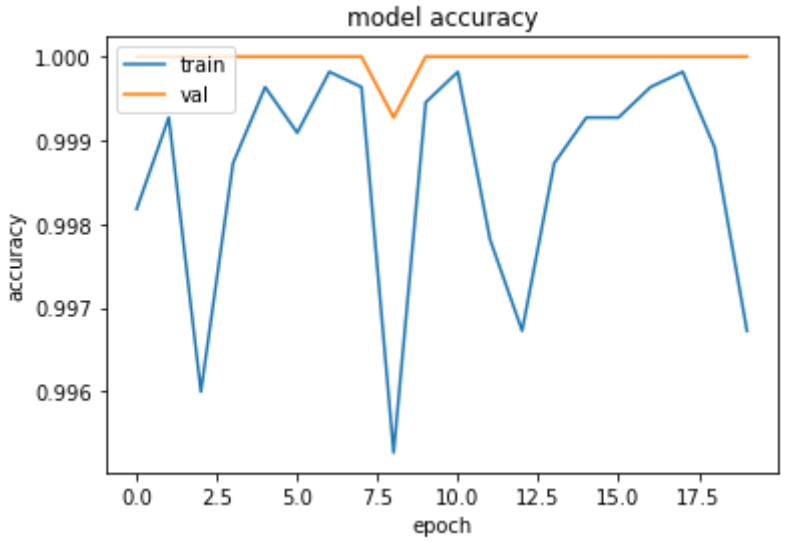


Figure - Model Accuracy for LSTM

The training, validation and test accuracy of LSTM are nearly corresponding 0.99. The confusion matrix also claims that models do a fantastic job of classifying all yogas correctly. When compared to MLP Classifier. However the loss curve for models during training that has been discussed in Methodology shows an increase in training loss and decrease in validation loss and also the accuracy model shows huge fluctuations in validation accuracy and training accuracy. Hence model is underfit as compared to MLP Classifier.

Below are some snapshots from the prediction that has been done by our system:

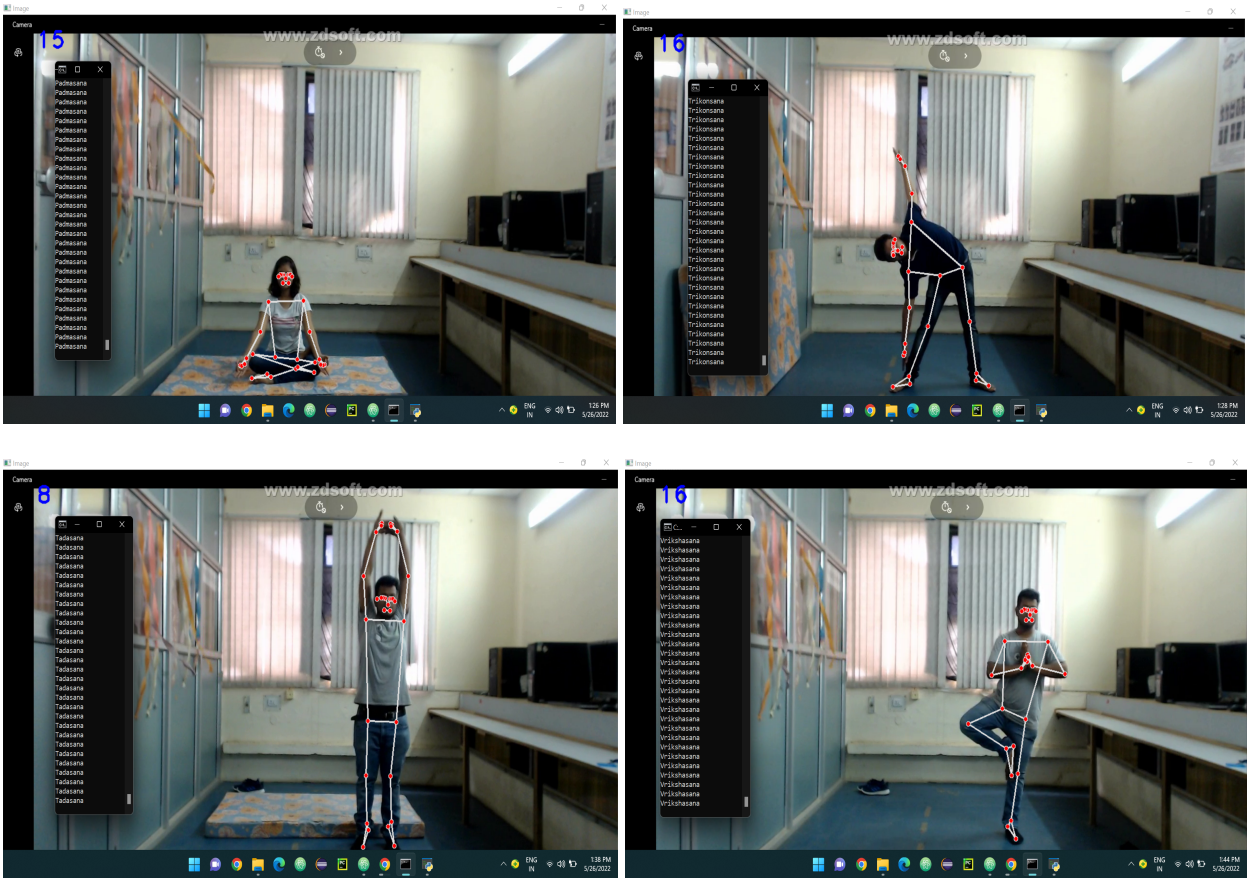
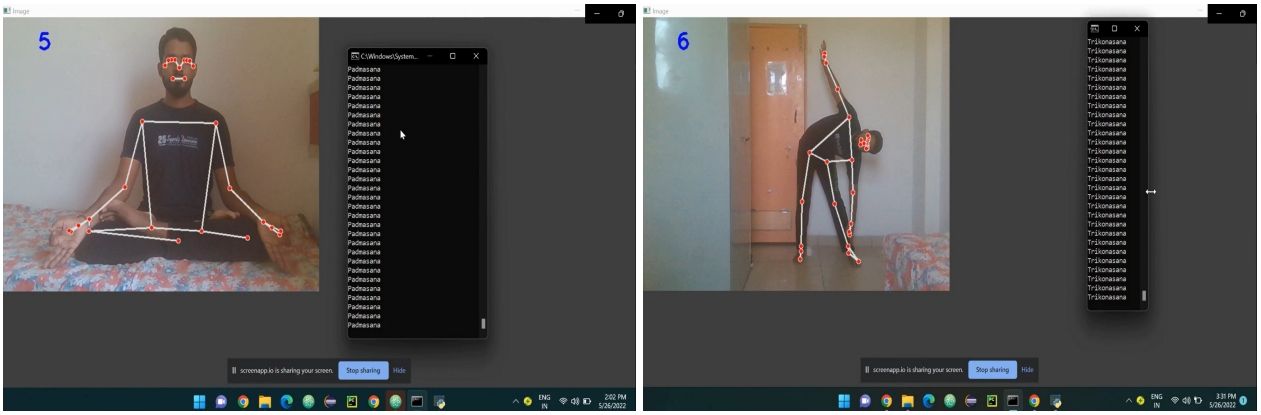


Figure - Predictions of asanas on recorded videos (top to bottom row): Padmasana,Trikonasana, Tadasana, Vrikshasana



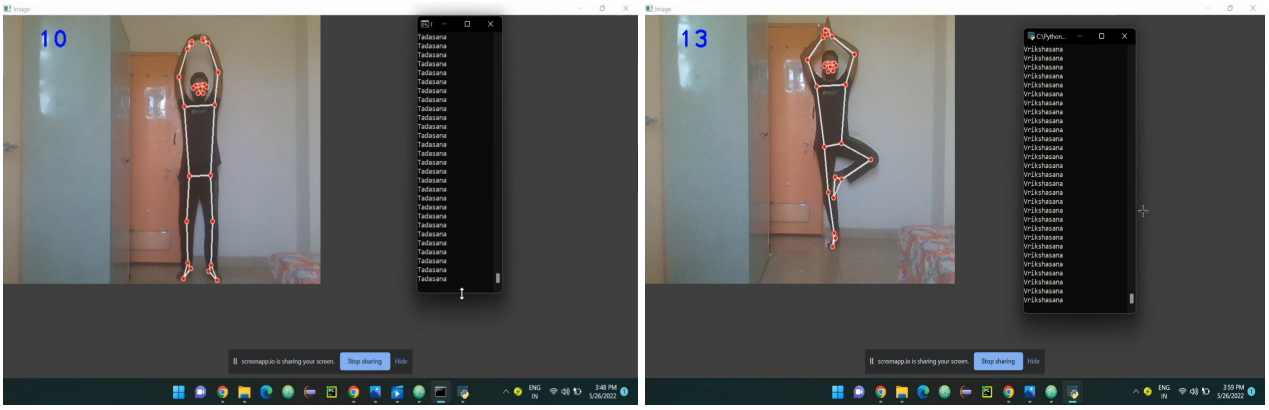
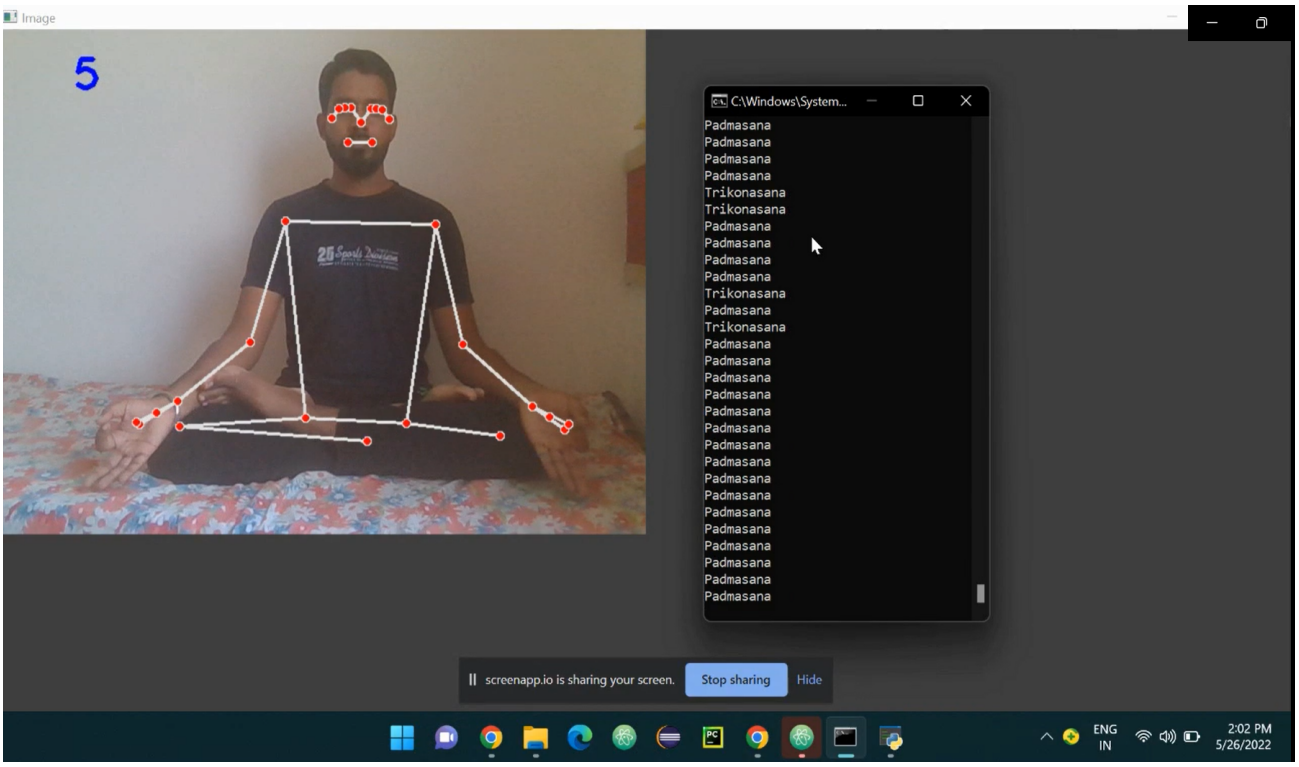


Figure - Predictions of asanas realtime (top to bottom row): Padmasana, Trikonasana, Tadasana, Vrikshasana



In the frame shown above the asana that is performed is Padmasana but the model shows Trikonasana at some points when there would have been some slight change in postures. Since, the role of landmarks location comes into play while prediction so the coordinates of different body parts helps the model to detect the asana there can be error while prediction because the coordinate read by the model may vary and correlation between different body location may give a chance of some other asana as a Prediction. Here Padmasana is getting predicted as Trikonasana This could be because of slight similarity in the position of elbow's coordinate since the coordinates extracted by mediapipe for the elbows in Padmasana and Trikonasana have slightly same correlation because the position of elbow in both correlates by near body part shoulder and in both cases the elbow is bended and make some angle with the coordinates of shoulder.

Comparison with other existing work and Analysis

Due to different dataset used by publishers the accuracies for the model can't be compared directly with the literature. In one of the literature that we reviewed the same data set was used. So a similar project was done with openpose and the model implemented was a combination of LSTM and CNN[8] where the accuracy of 99.38% has been achieved. Another Literature that has similar work using Kinetic and Star Skeleton [10] achieved accuracy of 99.33% but asanas in their dataset have a lot of difference in each other's appearance, whereas the asanas selected by us have some similarities between them. Experiments in [11] have used the Kinetic Sensor. This work gives a self training model for posture rectification here also the correlation between asanas had a lot of difference there system reached the accuracy of 82.84%, in feature axis extraction.

In our dataset 4 asanas have been performed by 15 different people and we have used 2 different models whereas accuracy of LSTM is better than MLP Classifier but looking at the possibilities of increasing in epochs with increase in no. of asanas in future MLP Classifier fits suitably as compared to LSTM in our work.

Overall, we can conclude from the comparison that the combination of LSTM+CNN would be the best model where CNN would be used for identifying and looking over patterns, and LSTM would make predictions.

Conclusion

The purpose of this paper is to propose a method for automatically detecting yoga poses in real-time or the recorded videos. There were five parts to the task. The first component of the study involved collecting publicly available data. The dataset can be found at https://archive.org/details/YogaVid. To detect all landmark points in the video frame by frame, we used Mediapipe in the second step after downloading the dataset. A CSV file was created for all four folders, which contained asanas such as Padmasana, Tadasana, Trikonasana, and Vrikshasana. This process was repeated for all four folders. To make the prediction, we used the Multilayer Perceptron (MLP) classifier and Long Short-Term Memory (LSTM) to train the model. Since the accuracy can be achieved better by different models, a new model with a hybrid of CNN and LSTM can be trained and a dataset consisting of different asanas with a variety of positions with not much correlation between them.

Future Scope

Implementing LSTM+CNN hybrid model for gesture and micropostures recognition. A large dataset will be taken with different backgrounds having a multiple of people in frame to produce precise and accurate results and minimise error occurring during predictions. This will be integrated with IOT for smart homes and this will be implemented through mobile applications to make it interactive with users. Applications of IOT which will be considered are smart AC and smart switches.

References

1. Qian, Junpeng & Cheng, Xiaogang & Yang, Bin & Li, Zhe & Ren, Junchi & Olofsson, Thomas & Li, Haibo. (2020). Vision-Based Contactless Pose Estimation for Human Thermal Discomfort. Atmosphere. 11. 376. 10.3390/atmos11040376.

2. Eric Marchand, Hideaki Uchiyama, Fabien Spindler. Pose Estimation for Augmented Reality: A Hands-On Survey. IEEE Transactions on Visualization and Computer Graphics, Institute of Electrical and Electronics Engineers, 2016, 22 (12), pp.2633 - 2651. <10.1109/TVCG.2015.2513408>

3. Zhang, Dong & Shah, Mubarak. (2016). A Framework for Human Pose Estimation in Videos.

4. Park, Soonchan & Chang, Ju & Jeong, Hyuk & Lee, Jae-Ho & Park, Ji-Young. (2017). Accurate and Efficient 3D Human Pose Estimation Algorithm Using Single Depth Images for Pose Analysis in Golf. 105-113. 10.1109/CVPRW.2017.19.

5. K. Chen et al., "Patient-Specific Pose Estimation in Clinical Environments," in IEEE Journal of Translational Engineering in Health and Medicine, vol. 6, pp. 1-11, 2018, Art no. 2101111, doi: 10.1109/JTEHM.2018.2875464.

6. Martinez, Gines Hidalgo. ‘OpenPose: Whole-Body Pose Estimation’. Carnegie Mellon University, 2019.

7. Kreiss, Sven, Lorenzo Bertoni, και Alexandre Alahi. ‘PifPaf: Composite Fields for Human Pose Estimation’. arXiv, 2019. https://doi.org/10.48550/ARXIV.1903.06593.

8. Yadav, Santosh & Singh, Amitoj Deep & Gupta, Abhishek & Raheja, Jagdish. (2019). Real-time Yoga recognition using deep learning. Neural Computing and Applications. 31. https://link.springer.com/article/10.1007/s00521-019. 10.1007/s00521-019-04232-7.

9. Nakano N, Sakura T, Ueda K, et al. Evaluation of 3D Markerless Motion Capture Accuracy Using OpenPose With Multiple Video Cameras. Front Sports Act Living. 2020;2:50. Published 2020 May 27. doi:10.3389/fspor.2020.00050

10. Chen HT, He YZ, Hsu CC et al (2014) Yoga posture recognition for self-training. In: Lecture notes in computer science (including subseries lecture notes in artificial intelligence and lecture notes in bioinformatics), pp 496–505

11. Chen HT, He YZ, Chou CL et al (2013) Computer-assisted self training system for sports exercise using kinects. In: Electronic proceedings of 2013 IEEE international conference multimedia and expo work ICMEW 2013 3–6. https://doi.org/10.1109/icmew.2013.6618307

Source Code:

https://github.com/Govind1212/Yoga-Pose-Estimation-using-Mediapipe