Neural Computing and Applications
https://doi.org/10.1007/s00521-019-04232-7

ORIGINAL ARTICLE

Real-time Yoga recognition using deep learning

Santosh Kumar Yadav[1] · Amitojdeep Singh[2] · Abhishek Gupta[2] · Jagdish Lal Raheja[1]

Received: 24 September 2018 / Accepted: 9 May 2019

Abstract

An approach to accurately recognize various Yoga asanas using deep learning algorithms has been presented in this work. A dataset of six Yoga asanas (i.e. Bhujangasana, Padmasana, Shavasana, Tadasana, Trikonasana, and Vrikshasana) has been created using 15 individuals (ten males and five females) with a normal RGB webcam and is made publicly available. A hybrid deep learning model is proposed using convolutional neural network (CNN) and long short-term memory (LSTM) for Yoga recognition on real-time videos, where CNN layer is used to extract features from keypoints of each frame obtained from OpenPose and is followed by LSTM to give temporal predictions. To the best of our knowledge, this is the first study using an end-to-end deep learning pipeline to detect Yoga from videos. The system achieves a test accuracy of 99.04% on single frames and 99.38% accuracy after polling of predictions on 45 frames of the videos. Using a model with temporal data leverages the information from previous frames to give an accurate and robust result. We have also tested the system in real time for a different set of 12 persons (five males and seven females) and achieved 98.92% accuracy. Experimental results provide a qualitative assessment of the method as well as a comparison to the state-of-the-art.

Keywords Activity recognition · OpenPose · Posture analysis · Sports training · Yoga

1 Introduction

Human activity recognition is a well-established computer vision problem that has imposed several challenges over the years [1]. It is the problem of locating keypoints and the posture of a human body from the sensor data. Activity recognition is useful in many domains including biometrics, video-surveillance, human–computer interaction, assisted living, sports arbitration, in-home health monitoring, etc. [2–4]. The health status of an individual can be evaluated and predicted with the help of monitoring and recognizing their activities [5]. Yoga posture recognition is a relatively newer application.

Yoga is an ancient science that originated in India. According to the Bhagavad Gita, it is the remover of misery and destroyer of pain. Recently, Yoga is getting popular across the globe due to its physical, mental, and spiritual benefits. In 2014, the General Assembly of United Nations has declared 21st June as the 'International Day of Yoga' [6]. Over the last decade, Yoga is getting increasing importance in the medical research community, and numerous literature has been proposed for various medical applications including cardiac rehabilitation [6], positive body image intervention [7, 8], mental illnesses [9], etc. Without the use of medicines, Yoga can completely cure many diseases [10]. Yoga exercises boost physical health as well as help to cleanse the body, mind, and soul [11]. It comprises of many asanas and each of them denotes the static physical postures [12]. Yoga learning and self-instruction systems have the ability to popularize and spread Yoga while ensuring that it is practiced correctly [13, 14].

Computer-assisted self-training systems for sports and exercises can improve the performance of participants and prevent injuries [15]. Many work in the literature have proposed automated and semi-automated systems for analysing the sports and exercise activities such as soccer player ranking [16], swimming [17], tennis strokes [18],

✉ Santosh Kumar Yadav
santoshyadav@ceeri.res.in

1 Cyber Physical System, CSIR – Central Electronics Engineering Research Institute, Pilani 333031, India

2 Department of Computer Science, Birla Institute of Technology and Science (BITS), Pilani 333031, India

badminton [19], rugby [20], basketball [21, 22], vertical high jump [23], hurdles racing [24], etc.

To detect the difference in postures between a practitioner and an expert, Patil et al. [10] have proposed a ‘Yoga Tutor’ project using speeded up robust features (SURF). However, to compare and describe the postures approximately by using only the contour information is not enough. Luo et al. [25] have proposed Yoga training system with an interface suit comprising 16 inertial measurement units (IMUs) and six tactors, which is obtrusive to the user and can affect the user to perform asana in a natural manner. Wu et al. [26] proposed an image and text-based expert system for Yoga; however, they have not analysed the practitioner’s posture.

Chen et al. [11] introduced Yoga activity recognition using features-based approach to design a self-training system. It uses a Kinect for extracting user’s body contour and capturing the body map. A star skeleton was used for rapid skeletonization to obtain a descriptor for the human pose. The work in [12] provides computer-assisted self-training system for posture rectification using Kinect. It has taken three postures in consideration, i.e. tree, warrior III, and downward facing dog. However, the overall accuracy is very low at only 82.84%. In [27], a Yoga detection system is proposed for six asanas using Kinect and Adaboost classification with 94.78% accuracy score. However, they are using depth sensor-based camera which generally may not be available to the users.

Mohanty et al. [28] have applied image recognition techniques for Indian classical dance and Yoga postures identification from images using convolutional neural network (CNN) and stacked autoencoder (SAE) algorithms. However, they have evaluated their performance on still images only and not on videos. Chen et al. [15] proposed a Yoga self-training system to assist in rectifying postures while performing Yoga using a Kinect depth camera for 12 different asanas. However, it is using manual feature extraction and making separate models for each asana.

Delegate features, like a human skeleton, are compulsory to extract for describing the human postures. There are various skeletonization techniques in the literature, such as thinning and distance transformation. However, these approaches have a high computational cost and are sensitive to noise [11].

The conventional skeletonization approach has been replaced by deep learning-based methods since the advent of DeepPose by Toshev et al. [29]. DeepPose leads the shift toward deep network-based approaches from classical ones. It uses deep neural network-based regressors to directly regress on coordinates of joints. It anticipates the activity of a person and predicts the location of hidden body parts as well. However, their approach suffers from the localization problem.

OpenPose [30] is a real-time multi-person system presented by Perceptual Computing Lab of Carnegie Mellon University (CMU) to jointly detect a human body, hand, facial, and foot keypoints on single images. It is a major revolution in the field of pose recognition and provides the human body joint locations using convolutional neural networks (CNNs)-based architecture [31]. CNNs are most widely used among deep learning architectures for vision applications. Traditional machine learning algorithms use handcrafted features, while CNNs learn some representational features automatically [32].

Using the OpenPose, the location of human body joints can be obtained from an RGB camera (Fig. 1). The keypoints obtained using OpenPose include ears, eyes, nose, neck, shoulders, elbows, wrists, knees, hips, and ankles as shown in Table 1. It can process inputs from a real-time camera, recorded video, static images, IP camera, etc., and present the results as 18 simple keypoints. This makes it suitable for a wide range of applications including surveillance, sports, activity detection, and Yoga pose recognition. We have used OpenPose in our system’s pipeline to leverage its robustness and flexibility for keypoint extraction.

Recurrent neural networks (RNNs) are effective for sequential information and already explored in many areas, including speech recognition and natural language processing [33, 34]. An activity can be considered as a sequence of actions so RNNs can be used for this sequential data [4]. Among various RNN architectures, long short-term memory networks (LSTMs) are the most popular as it can store information for the extended duration [35]. Hybrid of CNN and LSTM has been recently

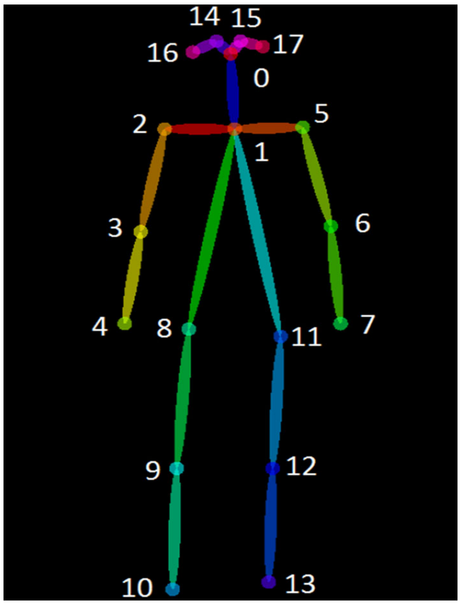


Fig. 1 Keypoints detected using OpenPose [30]

Table 1 Keypoints used

No.	Keypoint	No.	Keypoint
0	Nose	9	Right knee
1	Neck	10	Right foot
2	Right shoulder	11	Left hip
3	Right elbow	12	Left knee
4	Right wrist	13	Left foot
5	Left shoulder	14	Right eye
6	Left elbow	15	Left eye
7	Left wrist	16	Right ear
8	Right hip	17	Left ear

used for tasks like sentiment analysis [36], text classification [37], cardiac diagnosis [38], face anti-spoofing [39], and skeleton-based action recognition [40]. In the proposed method, we have used the hybrid model of CNN and LSTM, where the CNN is used for spatial features extraction and the LSTM is used for the temporal information processing.

The proposed system recognizes the very common six asanas in real time as well as from recorded videos. These asanas include Bhujangasana, Padmasana, Shavasana, Tadasana, Trikonasana, and Vrikshasana. We have also included one more class as ‘No Asana’ to identify the postures if it does not belong to any of the existing asanas. Table 2 describes the detail of each Yoga postures.

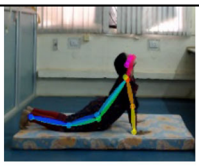
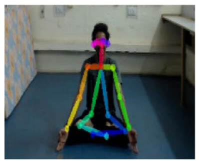
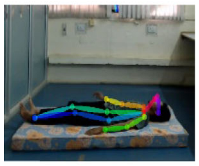
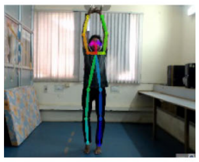
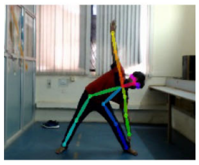
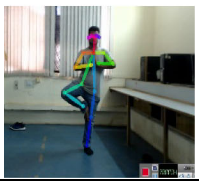
In subsequent sections, data collection, methodology, and results are discussed followed by the concluding remarks. Section 2 presents the data collection strategy and in Sect. 3, the proposed methodology is described, followed by the position of joints, the process of pose extraction, model description, and training and system information. The training outcomes and results are discussed in Sect. 4 and it shows the potential of our technique for real-world applications. Section 5 presents the discussion, where we have compared our results with some existing papers available for Yoga identification. Finally, in Sect. 6, the concluding remarks and scope for future applications are discussed.

2 Data collection

There is no publicly accessible dataset available for Yoga identification, so we have collected dataset with 15 individuals (10 males and 5 females) doing each of the six asanas. However, it can be extended to include a wider range of asanas.

Yoga videos are collected using HD 1080p (where p stands for progressive scan) Logitech web camera on a system with NVIDIA TITAN X GPU and Intel Xeon

Table 2 Asanas and corresponding keypoints

Sr. No.	Asana Name	Posture
1	Bhujangasana (Cobra Pose)	
2	Padmasana (Lotus Pose)	
3	Shavasana (Corpse Pose)	
4	Tadasana (Mountain Pose)	
5	Trikonasana (Triangle Pose)	
6	Vrikshasana (Tree Pose)	

processor with 32 GB RAM. Different persons have participated in data collection and in real-time prediction. Most of the asanas are performed at 4–5 metres distance in front of the camera. The users have performed each asana with all possible variations.

All the videos are collected for more than 45 s in an indoor environment with a frame rate of 30 frames per second. The total length of 88 videos for training is 1 h 6 min and 5 s at 30 frames per second, that is, a total of about 111,750 frames. We have also made the collected dataset publicly available at https://archive.org/details/YogaVidCollected. Table 3 describes the training dataset with the number of persons and number of videos for each asana. After the system is trained, the testing has been done on 18 different videos of all asanas (total length of ~12 min) and finally the system is also tested in real time (further details are in results section).

Table 3 Training dataset details

S. no.	Asana name	No. of persons	No of videos
1	Bhujangasana	15	16
2	Padmasana	14	14
3	Shavasana	15	15
4	Tadasana	15	15
5	Trikonasana	13	13
6	Vrikshasana	15	15
	Total number of videos		88

3 Proposed methodology

Our approach aims to automatically recognize the user's Yoga asanas from real time and recorded videos. The method can be decomposed into four main steps. First, data collection is performed which can either be a real-time process running in parallel with detection or can be previously recorded videos. Second, OpenPose is used to identify the joint locations using Part Confidence Maps and Part Affinity Fields followed by bipartite matching and parsing. The detected keypoints are passed to our model where CNN finds patterns and LSTM analyses their change over time. Finally, the model and training method of framewise prediction and polling approach on 45 frames (1.5 s) of output are discussed.

3.1 Pose extraction

This is the first step of our pipeline and the OpenPose library is utilized for it. In the case of recorded videos, this step takes place offline, whereas for real-time predictions, it takes place online using input from the camera to supply keypoints to the proposed model. OpenPose is an open-source library for multi-person keypoint detection, which detects the human body, hand, and facial keypoints jointly [30]. The positions of 18 keypoints tracked by the OpenPose, i.e. ears, eyes, nose, neck, shoulders, hips, knees, ankles, elbows, and wrists are displayed in Fig. 1.

The output corresponding to each frame of a video is obtained in JSON format which contains each body part locations for every person detected in the image. The pose extraction was performed at the default resolution of OpenPose network for optimal performance. The system operated at around 3 FPS at these settings. Figure 2 illustrates the proposed system architecture where OpenPose is used for keypoint extraction followed by the CNN and LSTM model to predict the user's asanas.

We used videos with distinct subjects for training, test, and validation sets with a 60:20:20 split at the video level.

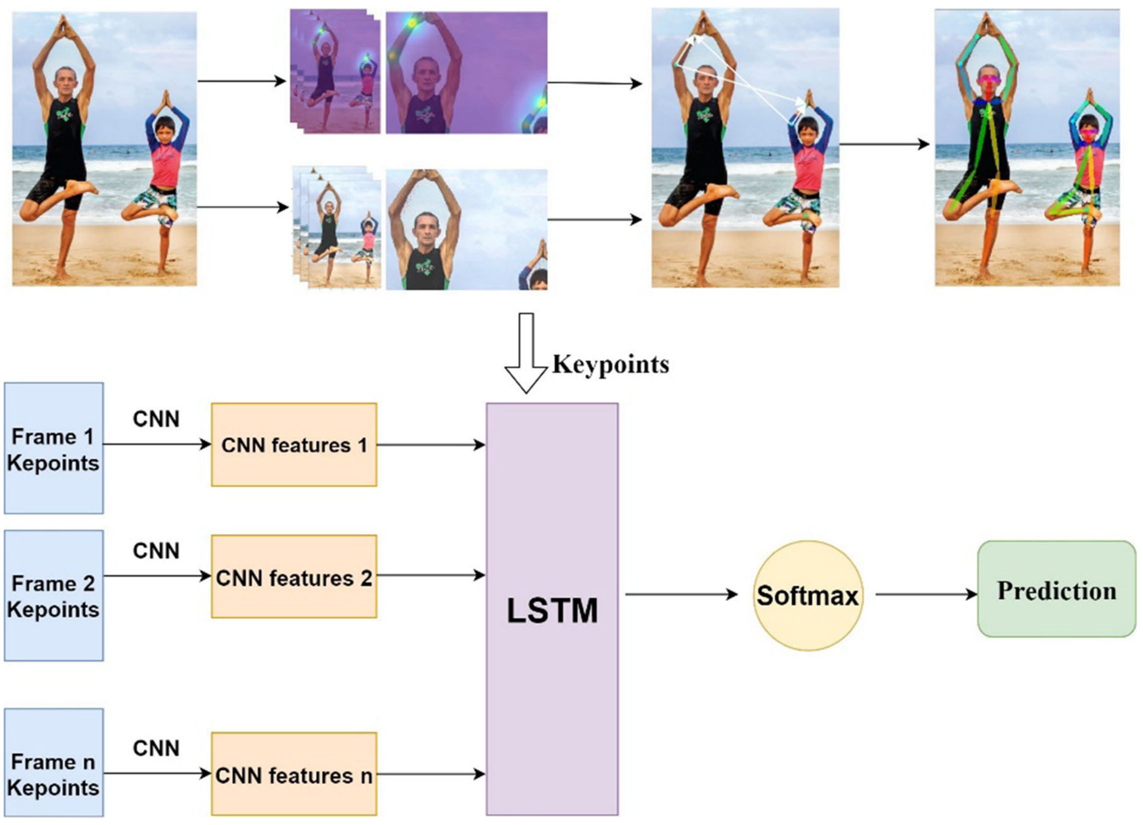


Fig. 2 System architecture: OpenPose followed by CNN and LSTM model

After the preprocessing, we obtain around 8000, 2500, and 2300 frames for training testing and validation cases, respectively. This deviated from 60:20:20 at the video level due to the variation in length of the videos.

3.2 Model

The deep learning model used here is a combination of CNN and LSTM (Fig. 2). CNN is commonly used for pattern recognition problems and LSTM is used for time-series tasks. In our work, the time-distributed CNN layer is used to extract features from the 2-D coordinates of the keypoints obtained in the previous step. The LSTM layer analyses the change in these features over the frames, and the probability of each Yoga in a frame is given by the Softmax layer. Thresholding is performed on this value to detect frames where the user is not performing Yoga and the effect of polling on 45 frames has been studied.

The model has been programmed using Keras Sequential API in Python. The input instance has a shape of $45 \times 18 \times 2$ which denotes the 45 sequential frames with 18 keypoints having X and Y coordinates each. Time-distributed CNN layer with 16 filters of size 3×3 having ReLU activation is applied to keypoints of each frame for feature extraction. CNNs have a strong ability to extract spatial features which are scale and rotation invariant. The CNN layer can extract spatial features like relative distance and angles between the various keypoints in a frame. Batch normalization is applied to the CNN output for faster convergence. This is followed by a dropout layer which randomly drops a fraction of the weights preventing overfitting.

The output from CNN, applied on each of the 45 frames, is then flattened and passed to LSTM layer with 20 units and unit forget bias of 0.5. LSTM is used to identify temporal changes in the features extracted by the CNN layer. This leverages the sequential nature of video input, and the entire Yoga starting from its formation to holding and release is treated as an activity.

The output of the LSTM layer corresponding to each frame is passed to a time-distributed fully connected layer with Softmax activation and six outputs. Each of these six outputs provides the probability of the corresponding Yoga in terms of cross-entropy. Thresholding is applied to this output to detect when the user is not performing Yoga. Though the model uses LSTM for capturing temporal relationship, the results are provided for each frame in the sequence and then polled for the entire sequence of 45 frames. This is further elaborated in the results section.

3.3 Training

Our task is to recognize the user's asanas with proper accuracy in real time. First, keypoint features are extracted

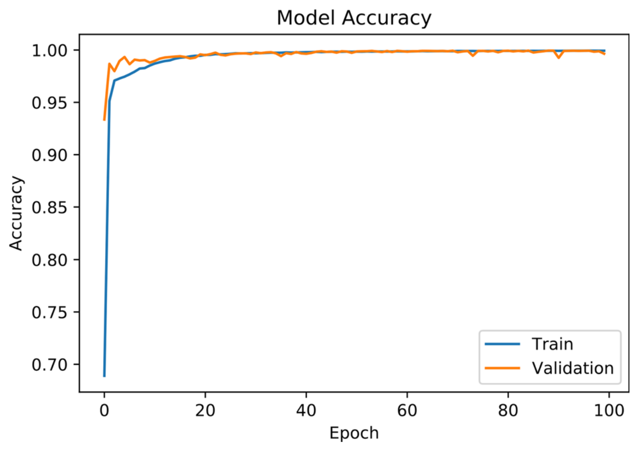


Fig. 3 Model accuracy for framewise approach over the epochs

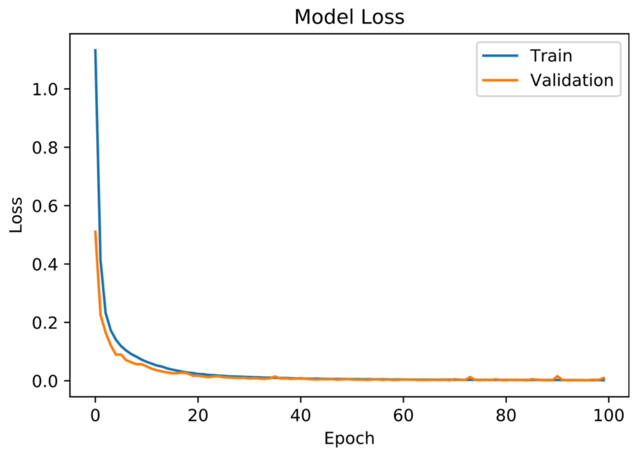


Fig. 4 Model loss over the epochs for framewise approach

using OpenPose and recorded the joint location values in the JSON file, and then CNN and LSTM models are applied for the prediction of asanas. Due to the combination of both, we get the best set of features filtered by CNN and long-term data dependencies established using LSTM.

The model is compiled using Keras with Theano backend. The categorical cross-entropy loss function is used because of its suitability to measure the performance of the fully connected layer's output with Softmax activation. Adam optimizer with an initial learning rate of 0.0001, a beta of 0.9 and no decay are used to control the learning rate.

The model has been trained for 100 epochs on a system with Intel i7 6700HQ processor, 16GB RAM, and Nvidia GTX 960M GPU. The training takes around 22 s per epoch which is relatively quick due to the simple inputs and compact design.

Figures 3 and 4 show the change in accuracy and loss function, respectively, over the course of training. Initially, the training and validation accuracies increase rapidly with validation accuracy staying above the training accuracy indicating a good generalization. Later, the growth is gradual, and convergence occurs after 20 epochs. The accuracy and loss approach to their asymptotic values after 40 epochs with minor noise in between. The weights of the best fitting model with highest validation accuracy are preserved for further testing. Both, training as well as validation loss have decreased uniformly and converged indicating a well-fitting model.

4 Experimental results

This section contains three main steps. The first step presents frame-by-frame results using the CNN and LSTM model on the videos recorded from the webcam. In the second step, polling is done for 45 frames and predicted the results on the recorded videos. After polling the 45 frames, it gives more accuracy as well as stable results. The third step presents the real-time prediction results using the webcam for a different set of peoples with all possible variations using thresholding method. If the score for particular asana is less than the threshold value, the model predicts as ‘No Asana.’

4.1 Frame-by-frame results

After training for 100 epochs, the system achieves 99.34% accuracy on training data and 99.41% accuracy on validation data. The system obtains a test accuracy of 99.04% for each frame. Model accuracy is plotted in Fig. 3 and model loss is in Fig. 4. Figures 5 and 6 present the confusion matrix and normalized confusion matrix, respectively, for frame-by-frame predictions using our model.

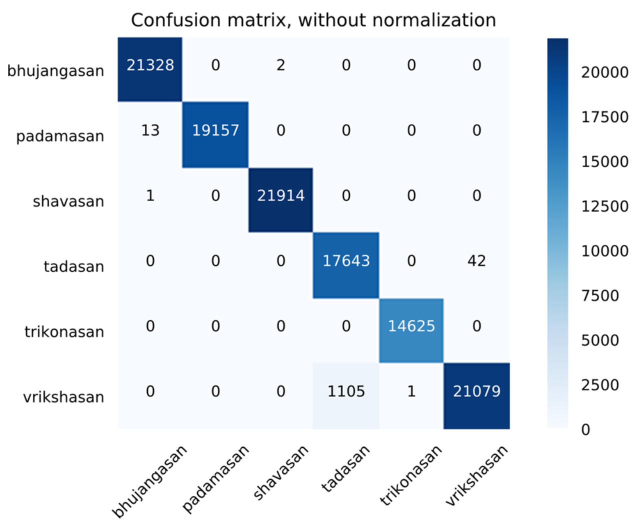


Fig. 5 Framewise confusion matrix with predicted labels on *X*-axis and true labels on the *Y*-axis

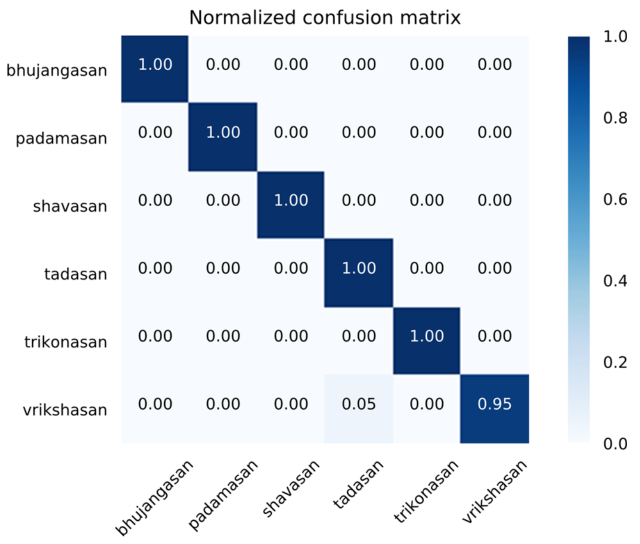


Fig. 6 Framewise normalized confusion matrix with predicted labels on *X*-axis and true labels on the *Y*-axis

The high density at the diagonal shows that most of the activities were predicted correctly.

The performance is well above and close to perfect in most of the asanas apart from Vrikshasana for test cases. Out of 21,079 frames for Vrikshasana, 1105 were classified as Tadasana leading to 95% accuracy for Vrikshasana. Likewise, there is some misclassification of Tadasana as Vrikshasana. The reason for this anomaly could be that both the asanas are performed while standing and their initial stages of formation are very similar.

The model gives fluctuation in the prediction of asanas in the real time using frame-by-frame approach. When the model is run in real time, its overall accuracy drops to 60%. Transition error occurs when asana is being formed or relieved. So, for dealing with these issues, polling is done for 45 frames to avoid the unstable results of the transition period. It corresponds to a time of 1.5 s since our dataset is recorded at 30 frames per seconds (fps). Using polling, the formation or relieving of an asana is also identified correctly as they are part of the activity of that asana.

4.2 Results after polling 45 frames

The mode of predictions for every frame out of a series of 45 frames corresponding to 1.5 s of video is taken. The asana that occurs for the maximum number of times out of 45 frames that is the predicted asana. In case ‘No Asana’ occurs more than any other asana among the 45 values, then ‘No Asana’ is shown. The system obtains a test accuracy of 99.38% with polling. There is an improvement in accuracy from 99.04 to 99.38% as the noisy predictions from some frames of a sequence are outnumbered by correct predictions in the sequence. The accuracy on

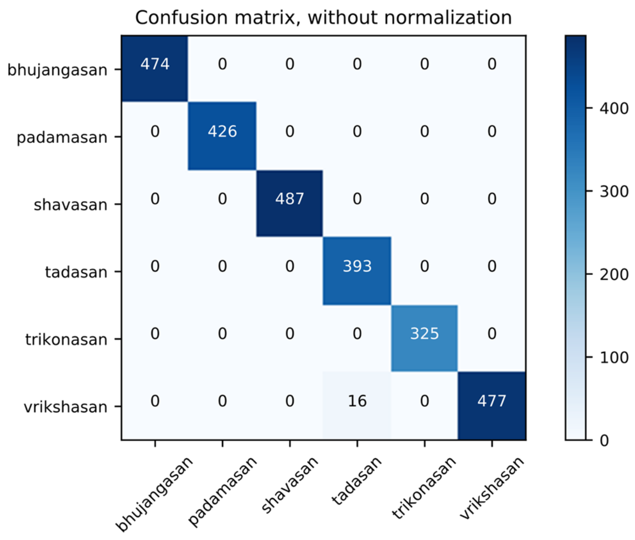


Fig. 7 Confusion matrix with predicted labels on *X*-axis and true labels on the *Y*-axis

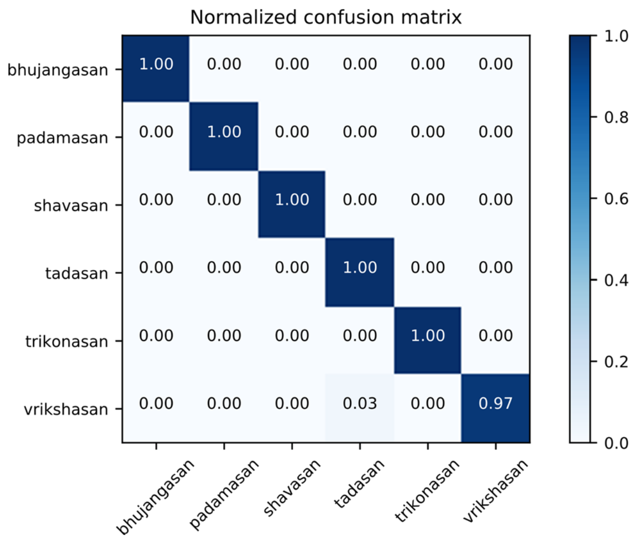


Fig. 8 Normalized confusion matrix with predicted labels on *X*-axis and true labels on the *Y*-axis

Vrikshasana, which is confused with Tadasana, goes up to 97% from 95% in the previous case. Figures 7 and 8 show the confusion matrix and normalized confusion matrix after performing polling on predictions of a sequence of 45 frames. This improvement in results shows that using a model with temporal data and applying polling to give the prediction on a sequence lead the better accuracy.

4.3 Real-time prediction using thresholding

Predictions are made in real time using 1080p HD Logitech webcam for 12 persons (5 males and 7 females) on a system with Intel Xeon processor with 32 GB RAM and Nvidia Titan X GPU. Different persons have participated in data collection and real-time prediction. All the participants have done asanas at a distance of 4–5 m from the camera in an indoor environment.

Thresholding on Softmax output was applied to determine whether the user was performing one of the asanas or not. A threshold value of 0.90 was found to be optimal as lower threshold value caused false positives in 'No Asana' case, while higher value caused true negatives in case asana was being performed. OpenPose performed at around 5.6 fps on our system and it takes around 8 s to generate 45 input frames for our model. So, the prediction generated at any time is for the previous 8 s. For faster but less accurate results, net resolution parameter of OpenPose can be altered by the user.

The results for real time are described in Table 4 and confusion matrices are presented in Figs. 9 and 10. The equation below describes the accuracy calculation method.

$$\text{Accuracy} = \frac{\text{Correct cases}}{\text{Total cases}} * 100$$

Table 4 Real-time results

S. no.	Activity	Total cases	Correct cases	Accuracy (%)
1	Bhujangasana	135	132	97.78
2	Padmasana	154	154	100.00
3	Shavasana	146	144	98.63
4	Tadasana	138	138	100.00
5	Trikonasana	178	178	100.00
6	Vrikshasana	178	173	97.19
	Total	929	919	98.92

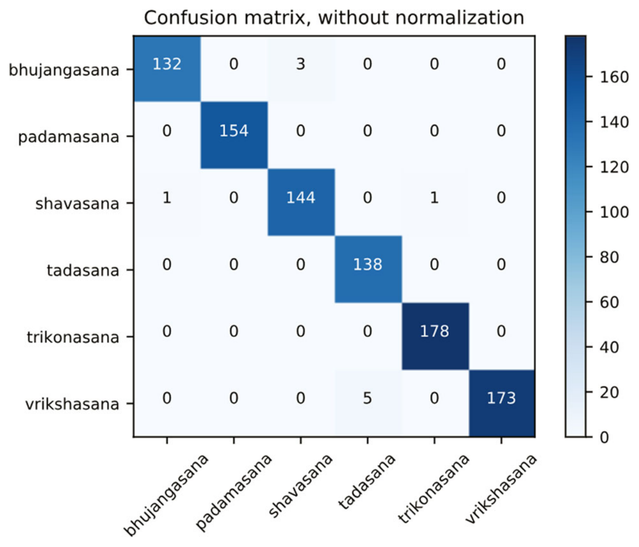


Fig. 9 Confusion matrix for real-time predictions with predicted labels on *X*-axis and true labels on the *Y*-axis

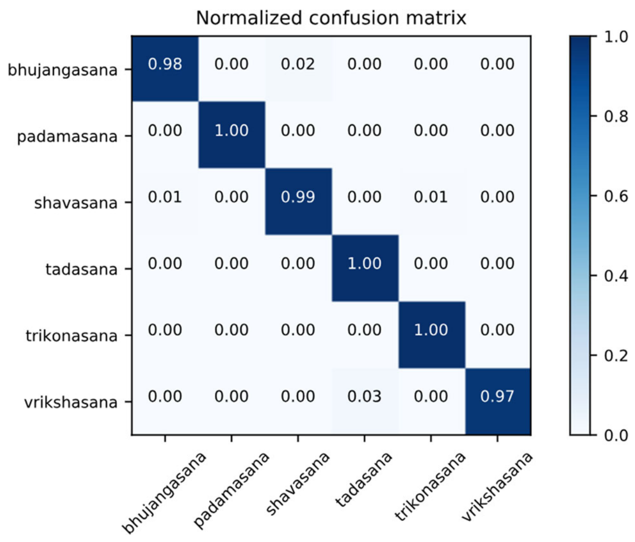


Fig. 10 Normalized confusion matrix for real-time predictions with predicted labels on *X*-axis and true labels on the *Y*-axis

The system achieves an overall accuracy of 98.92% with an accuracy above 97% for all the asanas individually. These results are coherent with those obtained for the test set, indicating a promising result in real-time scenarios.

5 Discussion

Due to inaccessibility to the same dataset, the accuracies can't be compared directly with the literature. Previously, similar experiments were performed using Kinect and Star Skeleton [11] resulting in an accuracy of 99.33%. Our model has a comparable if not better accuracy of 99.38%. However, the asanas used in their work was totally different from each other in appearance, while our asanas include significant similarities between them. Still, our model was able to give better accuracy than their model. Table 5 summarized the accuracy of our model for framewise as well as polling of 45 frames. The results are found to be consistent and polling has improved the accuracy. Figure 11 shows the prediction of the asanas in real time and Fig. 12 shows the prediction of asanas on recorded videos.

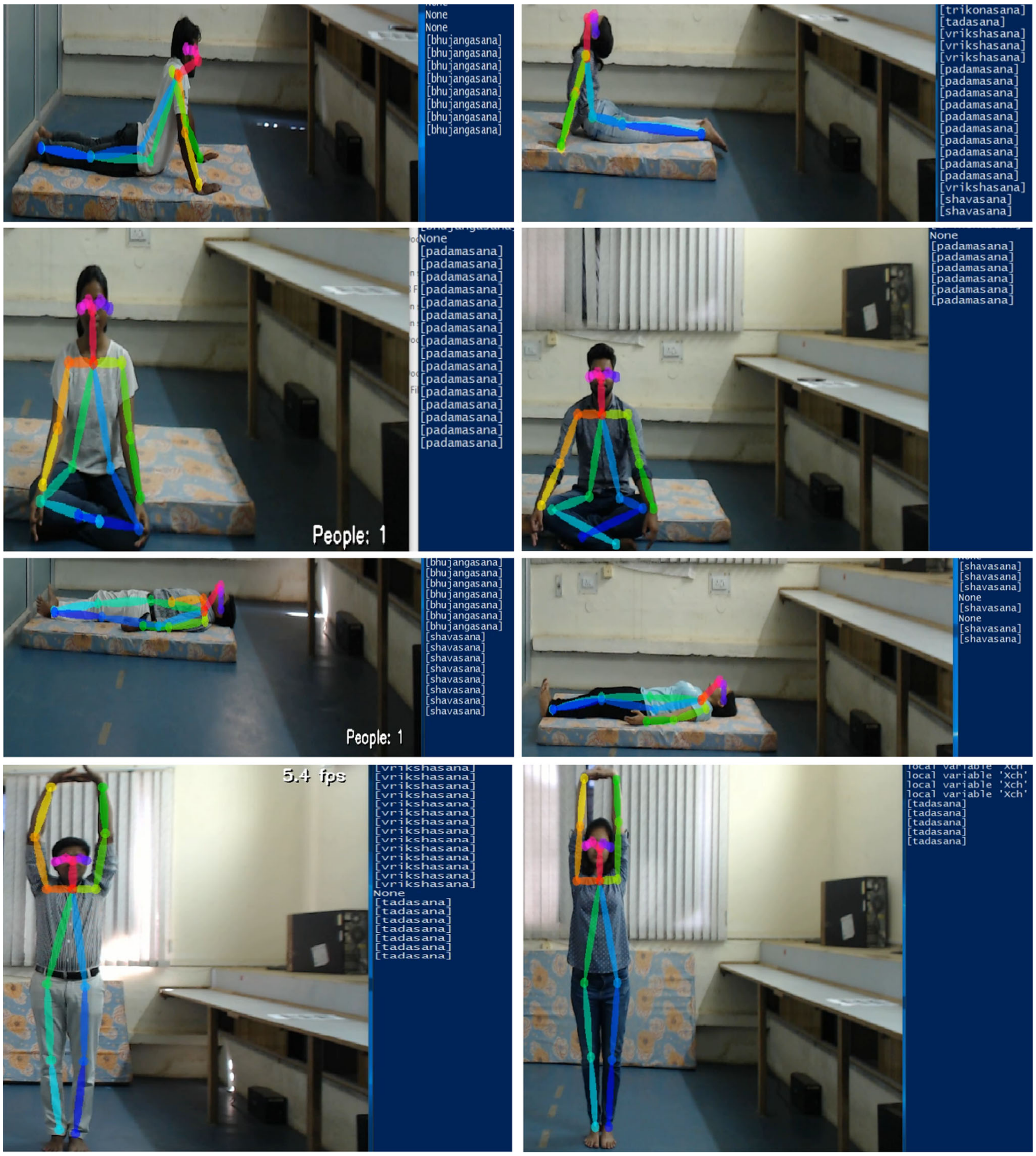
Table 5 Results summary

S. no.	Name	Type	Accuracy (%)
1	Training	Framewise	99.34
2	Validation	Framewise	99.41
3	Test	Framewise	99.04
4	Test	Polled	99.38
5	Real time	Polled	98.92

The work in [12] provides a self-training system for posture rectification using the Kinect sensor. It takes three postures in consideration, i.e. tree, downward facing dog, and warrior III, while we have taken six asanas into consideration and used normal RGB webcam. In their data collection, five persons have performed each asana five times, while in our case, 15 different peoples have performed all six asanas. In their method, even all three asanas were totally different from each other in appearance, it obtains an overall accuracy of 82.84% in feature axis extraction. Recently, Chen et al. [15] used body contour, skeleton, dominant axes, and feature points for accuracy of 76.22–99.87% for various poses and views using the Kinect sensor. However, they used manual feature extraction and made a separate model for each asana, which is time-consuming to implement and require a new set of features to be handcrafted every time to add new poses. In our approach, incorporating new classes of Yoga is convenient as we can add one neuron in the last dense layer and retrain the model on the new dataset. Our model gives 99.38% which is also superior to their model. However, our system is constrained by the quality of pose recognition of OpenPose which can fail in cases of overlapping parts shared by two persons and false positives on statues and animals.

6 Conclusions

In this paper, we proposed a Yoga identification system using a traditional RGB camera. The dataset is collected using HD 1080p Logitech webcam for 15 individuals (ten males and five females) and made publicly available. OpenPose is used to capture the user and detect keypoints. The end-to-end deep learning-based framework eliminates the need for making handcrafted features allowing for the addition of new asanas by just retraining the model with new data. We applied the time-distributed CNN layer to detect patterns between keypoints in a single frame and the LSTM to memorize the patterns found in the recent frames. Using LSTM for the memory of previous frames and polling for denoising, the results make the system even more robust by minimizing the error due to false keypoint detection. Since the frames of a Yoga video are sequential, the pattern in the last few frames must be taken into



```
None
None
[bhujangasana]
[bhujangasana]
[bhujangasana]
[bhujangasana]
[bhujangasana]
[bhujangasana]
[bhujangasana]
[bhujangasana]
```

```
[trikonasana]
[tadasana]
[vrikshasana]
[vrikshasana]
[vrikshasana]
[padamasana]
[padamasana]
[padamasana]
[padamasana]
[padamasana]
[padamasana]
[padamasana]
[padamasana]
[padamasana]
[padamasana]
[vrikshasana]
[shavasana]
[shavasana]
```

```
None
[padamasana]
[padamasana]
[padamasana]
[padamasana]
[padamasana]
[padamasana]
[padamasana]
[padamasana]
[padamasana]
[padamasana]
[padamasana]
[padamasana]
[padamasana]
[padamasana]
[padamasana]
[padamasana]
```

```
None
[padamasana]
[padamasana]
[padamasana]
[padamasana]
[padamasana]
[padamasana]
```

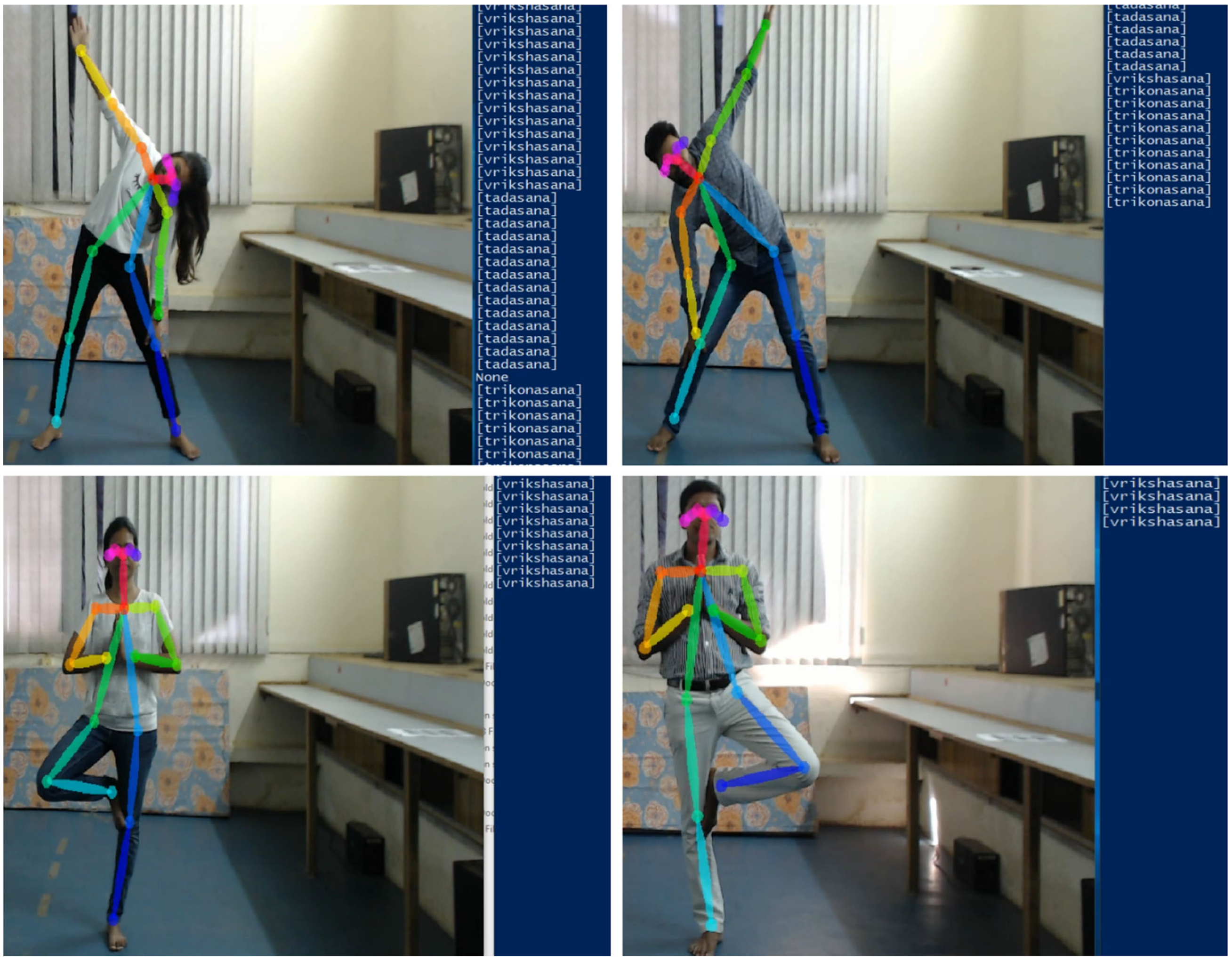
```
[bhujangasana]
[bhujangasana]
[bhujangasana]
[bhujangasana]
[bhujangasana]
[bhujangasana]
[bhujangasana]
[shavasana]
[shavasana]
[shavasana]
[shavasana]
[shavasana]
[shavasana]
[shavasana]
[shavasana]
```

```
[shavasana]
[shavasana]
[shavasana]
None
[shavasana]
None
[shavasana]
[shavasana]
```

```
[vrikshasana]
[vrikshasana]
[vrikshasana]
[vrikshasana]
[vrikshasana]
[vrikshasana]
[vrikshasana]
[vrikshasana]
[vrikshasana]
[vrikshasana]
[vrikshasana]
[vrikshasana]
[vrikshasana]
None
[tadasana]
[tadasana]
[tadasana]
[tadasana]
[tadasana]
[tadasana]
[tadasana]
```

```
local variable 'xch'
local variable 'xch'
local variable 'xch'
[tadasana]
[tadasana]
[tadasana]
[tadasana]
[tadasana]
```

Fig. 11 Predictions of asanas in real time (top to bottom row): Bhujangasana (column 2 has the wrong prediction), Padmasana, Shavasana, Trikonasana, and Vrikshasana



```
[vrikshasana]
[vrikshasana]
[vrikshasana]
[vrikshasana]
[vrikshasana]
[vrikshasana]
[vrikshasana]
[vrikshasana]
[vrikshasana]
[vrikshasana]
[vrikshasana]
[vrikshasana]
[vrikshasana]
[vrikshasana]
[tadasana]
[tadasana]
[tadasana]
[tadasana]
[tadasana]
[tadasana]
[tadasana]
[tadasana]
[tadasana]
[tadasana]
[tadasana]
[tadasana]
[tadasana]
[tadasana]
None
[trikonasana]
[trikonasana]
[trikonasana]
[trikonasana]
[trikonasana]
[trikonasana]
```

```
[tadasana]
[tadasana]
[tadasana]
[tadasana]
[tadasana]
[vrikshasana]
[trikonasana]
[trikonasana]
[trikonasana]
[trikonasana]
[trikonasana]
[trikonasana]
[trikonasana]
[trikonasana]
[trikonasana]
[trikonasana]
```

```
[vrikshasana]
[vrikshasana]
[vrikshasana]
[vrikshasana]
[vrikshasana]
[vrikshasana]
[vrikshasana]
[vrikshasana]
[vrikshasana]
```

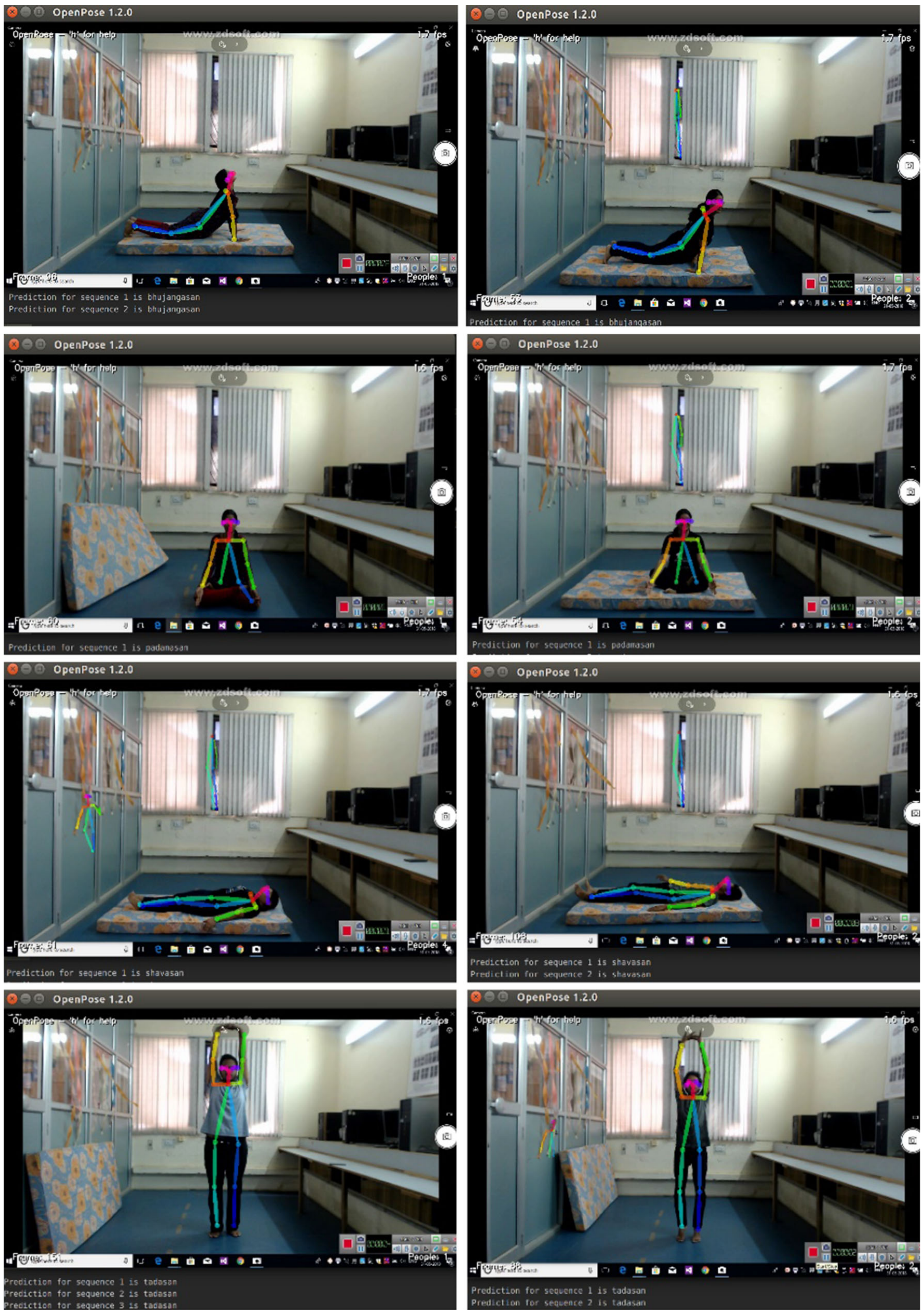
```
[vrikshasana]
[vrikshasana]
[vrikshasana]
[vrikshasana]
```

Fig. 11 continued

account especially during formation and release of the asanas. The system outputs ‘No Asana’ when the Softmax value for the majority of predictions in a sequence is below the threshold value.

The approach of using CNN and LSTM on pose data obtained from OpenPose for Yoga posture detection has been found to be highly effective. The system recognizes the six asanas on recorded videos as well as in real time for 12 persons (five males and seven females). Different persons have been used for data collection and real-time testing. The system successfully detects Yoga poses in a video with 99.04% accuracy for framewise and 99.38% accuracy after polling of 45 frames. The system achieved 98.92% accuracy in real time for a set of 12 different people showing its ability to perform well with new subjects and conditions.

It must be noted that our approach eradicates the need for Kinect or any other specialized hardware for Yoga posture identification and can be implemented on input from a regular RGB camera. In future work, more asanas and a larger dataset comprising of both image and videos can be included. Also, the system can be implemented on a portable device for real-time predictions and self-training. This work serves as a demonstration of activity recognition



```
Prediction for sequence 1 is bhujangasan
Prediction for sequence 2 is bhujangasan
```

```
Prediction for sequence 1 is bhujangasan
```

```
Prediction for sequence 1 is padamasan
```

```
Prediction for sequence 1 is padamasan
```

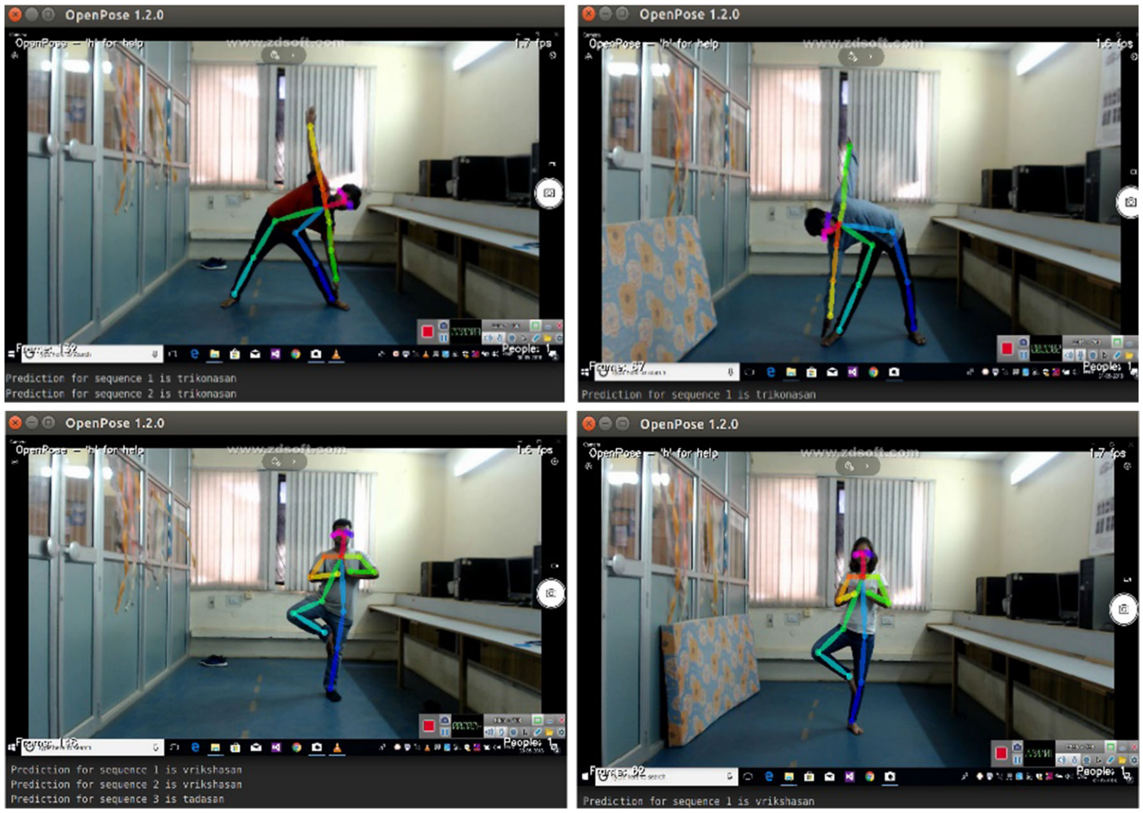
```
Prediction for sequence 1 is shavasan
```

```
Prediction for sequence 1 is shavasan
Prediction for sequence 2 is shavasan
```

```
Prediction for sequence 1 is tadasan
Prediction for sequence 2 is tadasan
Prediction for sequence 3 is tadasan
```

```
Prediction for sequence 1 is tadasan
Prediction for sequence 2 is tadasan
```

Fig. 12 Predictions of asanas on recorded videos (top to bottom row): Bhujangasana, Padmasana, Shavasana, Tadasana (column 2 has a wrongly predicted sequence), Trikonasana, and Vrikshasana (Column 1 has a wrongly predicted sequence)



```
Prediction for sequence 1 is trikonasan
Prediction for sequence 2 is trikonasan
```

```
Prediction for sequence 1 is trikonasan
```

```
Prediction for sequence 1 is vrikshasan
Prediction for sequence 2 is vrikshasan
Prediction for sequence 3 is tadasan
```

```
Prediction for sequence 1 is vrikshasan
```

Fig. 12 continued

systems for realistic applications. A similar approach can be used for posture recognition in various tasks like surveillance, sports, healthcare, image classification, etc.

Acknowledgement The work is carried out at CSIR-CEERI, Pilani and authors would like to thank Director, CSIR-CEERI, Pilani for providing the infrastructure and technical support and also, we appreciate the assistance provided by CSIR, India.

References

1. Gao Z, Zhang H, Liu AA et al (2016) Human action recognition on depth dataset. Neural Comput Appl 27:2047–2054. https://doi.org/10.1007/s00521-015-2002-0
2. Poppe R (2010) A survey on vision-based human action recognition. Image Vis Comput 28:976–990. https://doi.org/10.1016/j.imavis.2009.11.014
3. Weinland D, Ronfard R, Boyer E (2011) A survey of vision-based methods for action representation, segmentation and recognition. Comput Vis Image Underst 115:224–241. https://doi.org/10.1016/j.cviu.2010.10.002
4. Ladjailia A, Bouchrika I, Merouani HF et al (2019) Human activity recognition via optical flow: decomposing activities into basic actions. Neural Comput Appl. https://doi.org/10.1007/s00521-018-3951-x
5. Suto J (2018) Comparison of offline and real-time human activity recognition results using machine learning techniques. Neural Comput Appl. https://doi.org/10.1007/s00521-018-3437-x
6. Guddeti RR, Dang G, Williams MA, Alla VM (2018) Role of Yoga in cardiac disease and rehabilitation. J Cardiopulm Rehabil Prev. https://doi.org/10.1097/hcr.0000000000000372
7. Neumark-Sztainer D, Watts AW, Rydell S (2018) Yoga and body image: how do young adults practicing yoga describe its impact on their body image? Body Image 27:156–168. https://doi.org/10.1016/j.bodyim.2018.09.001
8. Halliwell E, Dawson K, Burkey S (2019) A randomized experimental evaluation of a yoga-based body image intervention. Body Image 28:119–127. https://doi.org/10.1016/j.bodyim.2018.12.005
9. Sathyanarayanan G, Vengadavaradan A, Bharadwaj B (2019) Role of yoga and mindfulness in severe mental illnesses: a narrative review. Int J Yoga 12:3–28. https://doi.org/10.4103/ijoy.IJOY_65_1
10. Patil S, Pawar A, Peshave A et al (2011) Yoga tutor: visualization and analysis using SURF algorithm. In: Proceedings of 2011 IEEE control system graduate research colloquium, ICSGRC 2011, pp 43–46
11. 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
12. 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
13. Schure MB, Christopher J, Christopher S (2008) Mind–body medicine and the art of self-care: teaching mindfulness to counseling students through yoga, meditation, and qigong. J Couns Dev. https://doi.org/10.1002/j.1556-6678.2008.tb00625.x

14. Lim S-A, Cheong K-J (2015) Regular Yoga practice improves antioxidant status, immune function, and stress hormone releases in young healthy people: a randomized, double-blind, controlled pilot study. J Altern Complement Med 1:1. https://doi.org/10.1089/acm.2014.0044
15. Chen HT, He YZ, Hsu CC (2018) Computer-assisted yoga training system. Multimed Tools Appl 77:23969–23991. https://doi.org/10.1007/s11042-018-5721-2
16. Maanijou R, Mirroshandel SA (2019) Introducing an expert system for prediction of soccer player ranking using ensemble learning. Neural Comput Appl. https://doi.org/10.1007/s00521-019-04036-9
17. Nordsborg NB, Espinosa HG, Thiel DV (2014) Estimating energy expenditure during front crawl swimming using accelerometers. Procedia Eng 72:132–137. https://doi.org/10.1016/j.proeng.2014.06.024
18. Connaghan D, Kelly P, O'Connor NE et al (2011) Multi-sensor classification of tennis strokes. Proc IEEE Sens. https://doi.org/10.1109/icsens.2011.6127084
19. Shan CZ, Su E, Ming L (2015) Investigation of upper limb movement during badminton smash. In: 2015 10th Asian Control conference, pp 1–6. https://doi.org/10.1109/ascc.2015.7244605
20. Waldron M, Twist C, Highton J et al (2011) Movement and physiological match demands of elite rugby league using portable global positioning systems. J Sports Sci 29:1223–1230. https://doi.org/10.1080/02640414.2011.587445
21. Pai PF, ChangLiao LH, Lin KP (2017) Analyzing basketball games by a support vector machines with decision tree model. Neural Comput Appl 28:4159–4167. https://doi.org/10.1007/s00521-016-2321-9
22. Bai L, Efstratiou C, Ang CS (2016) WeSport: utilising wrist-band sensing to detect player activities in basketball games. In: 2016 IEEE international conference on pervasive computing and communication workshops, PerCom workshops 2016. IEEE, pp 1–6
23. Yahya U, Arosha Senanayake SMN, Naim AG (2018) A database-driven neural computing framework for classification of vertical jump patterns of healthy female netballers using 3D kinematics—EMG features. Neural Comput Appl. https://doi.org/10.1007/s00521-018-3653-4
24. Przednowek K, Wiktorowicz K, Krzeszowski T, Iskra J (2018) A web-oriented expert system for planning hurdles race training programmes. Neural Comput Appl 1:1–17. https://doi.org/10.1007/s00521-018-3559-1
25. Luo Z, Yang W, Ding ZQ et al (2011) "Left arm up!" interactive Yoga training in virtual environment. In: 2011 IEEE virtual real conference, pp 261–262. https://doi.org/10.1109/vr.2011.5759498
26. Wu W, Yin W, Guo F (2010) Learning and self-instruction expert system for Yoga. In: Proceedings of 2010 2nd International Work Intelligent System Application: ISA, pp 2–5. https://doi.org/10.1109/iwisa.2010.5473592
27. Trejo EW, Yuan P (2018) Recognition of Yoga poses through an interactive system with kinect device. In: 2018 2nd international conference robotics and automation science: ICRAS, pp 12–17. https://doi.org/10.1109/icras.2018.8443267
28. Mohanty A, Ahmed A, Goswami T et al (2017) Robust pose recognition using deep learning. In: Raman B, Kumar S, Roy PP, Sen D (eds) Advances in intelligent systems and computing. Springer, Singapore, pp 93–105. https://doi.org/10.1007/978-981-10-2107-7_9
29. Toshev A, Szegedy C (2013) DeepPose: human pose estimation via deep neural networks. https://doi.org/10.1109/cvpr.2014.214
30. Cao Z, Simon T, Wei SE, Sheikh Y (2017) Realtime multi-person 2D pose estimation using part affinity fields. In: Proceedings of 30th IEEE conference computer vision and pattern recognition, CVPR 2017, 2017 January, pp 1302–1310. https://doi.org/10.1109/cvpr.2017.143
31. Qiao S, Wang Y, Li J (2017) Real time human gesture grading based on OpenPose. International Congress Image Signal Process
32. Luo G, Sun G, Wang K et al (2016) A novel left ventricular volumes prediction method based on deep learning network in cardiac MRI, pp 1604–1610. https://doi.org/10.22489/cinc.2016.028-224
33. Kiros R, Zhu Y, Salakhutdinov R et al (2015) Skip-thought vectors, pp 1–11. https://doi.org/10.1017/cbo9781107415324.004
34. Li J, Luong M-T, Jurafsky D (2015) A hierarchical neural autoencoder for paragraphs and documents. https://doi.org/10.3115/v1/p15-1107
35. Grushin A, Monner DD, Reggia JA, Mishra A (2013) Robust human action recognition via long short-term memory. In: International joint conference on neural networks (IJCNN), pp 1–8
36. Wang J, Yu L-C, Lai KR, Zhang X (2016) Dimensional sentiment analysis using a regional CNN-LSTM model, pp 225–230. https://doi.org/10.18653/v1/p16-2037
37. Zhou C, Sun C, Liu Z, Lau FCM (2015) A C-LSTM neural network for text classification. https://doi.org/10.1212/01.wnl.0000296829.66406.14
38. Oh SL, Ng EYK, Tan RS, Acharya UR (2018) Automated diagnosis of arrhythmia using combination of CNN and LSTM techniques with variable length heart beats. Comput Biol Med 102:278–287. https://doi.org/10.1016/j.compbiomed.2018.06.002
39. Xu Z, Li S, Deng W (2016) Learning temporal features using LSTM-CNN architecture for face anti-spoofing. In: Proceedings of 3rd IAPR Asian conference pattern recognition, ACPR 2015, pp 141–145
40. Wang X, Gao L, Song J, Shen H (2017) Beyond frame-level CNN: saliency-Aware 3-D CNN with LSTM for video action recognition. IEEE Signal Process Lett 24:510–514. https://doi.org/10.1109/LSP.2016.2611485