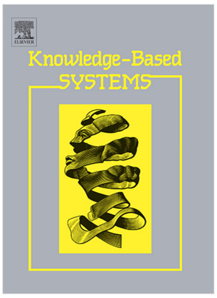


YogNet: A two-stream network for realtime multiperson yoga action recognition and posture correction

Santosh Kumar Yadav, Aayush Agarwal, Ashish Kumar,
Kamlesh Tiwari, Hari Mohan Pandey, Shaik Ali Akbar

PII: S0950-7051(22)00541-X
DOI: https://doi.org/10.1016/j.knosys.2022.109097
Reference: KNOSYS 109097

To appear in: *Knowledge-Based Systems*

Received date : 6 October 2021
Revised date : 15 May 2022
Accepted date : 17 May 2022

YogNet: A Two-Stream Network for Realtime Multiperson Yoga Action Recognition and Posture Correction

Santosh Kumar Yadav[a,b], Aayush Agarwal[c], Ashish Kumar[c], Kamlesh Tiwari[c],
Hari Mohan Pandey[d], Shaik Ali Akbar[a,b]

[a] *Academy of Scientific and Innovative Research (AcSIR), Ghaziabad, UP-201002, India*
[b] *Cyber Physical System, CSIR-Central Electronics Engineering Research Institute (CEERI), Pilani-333031, India*
[c] *Department of CSIS, Birla Institute of Technology and Science Pilani, Pilani Campus, Rajasthan-333031, India*
[d] *Department of Computing and informatics, Bournemouth University,United Kingdom*

Abstract

Yoga is a traditional Indian exercise. It specifies various body postures called *asanas*, practicing them is beneficial for the physical, mental, and spiritual well-being. To support the yoga practitioners, there is a need of an expert yoga *asanas* recognition system that can automatically analyze practitioner's postures and could provide suitable posture correction instructions. This paper proposes YogNet, a multi-person yoga expert system for 20 asanas using a two-stream deep spatiotemporal neural network architecture. The first stream utilizes a keypoint detection approach to detect the practitioner's pose, followed by the formation of bounding boxes across the subject. The model then applies time distributed convolutional neural networks (CNNs) to extract frame-wise postural features, followed by regularized long short-term memory (LSTM) networks to give temporal predictions. The second stream utilizes 3D-CNNs for spatiotemporal feature extraction from RGB videos. Finally, the scores of two streams are fused using multiple fusion techniques. A yoga asana recognition database *(YAR)* containing 1206 videos is collected using a single 2D web camera for 367 minutes with the help of 16 participants and contains four view variations *i.e.* front, back, left, and right sides. The proposed system is novel as this is the earliest two-stream deep learning-based system that can perform multi-person yoga asanas recognition and correction in realtime. Simulation result reveals that YogNet system achieved 77.29%, 89.29%, and 96.31% accuracies using pose stream, RGB stream, and via fusion of both streams, respectively. These results are impressive and sufficiently high for recommendation towards general adaption of the system.

Keywords: Action recognition, Computer vision, Posture correction, Yoga and exercise

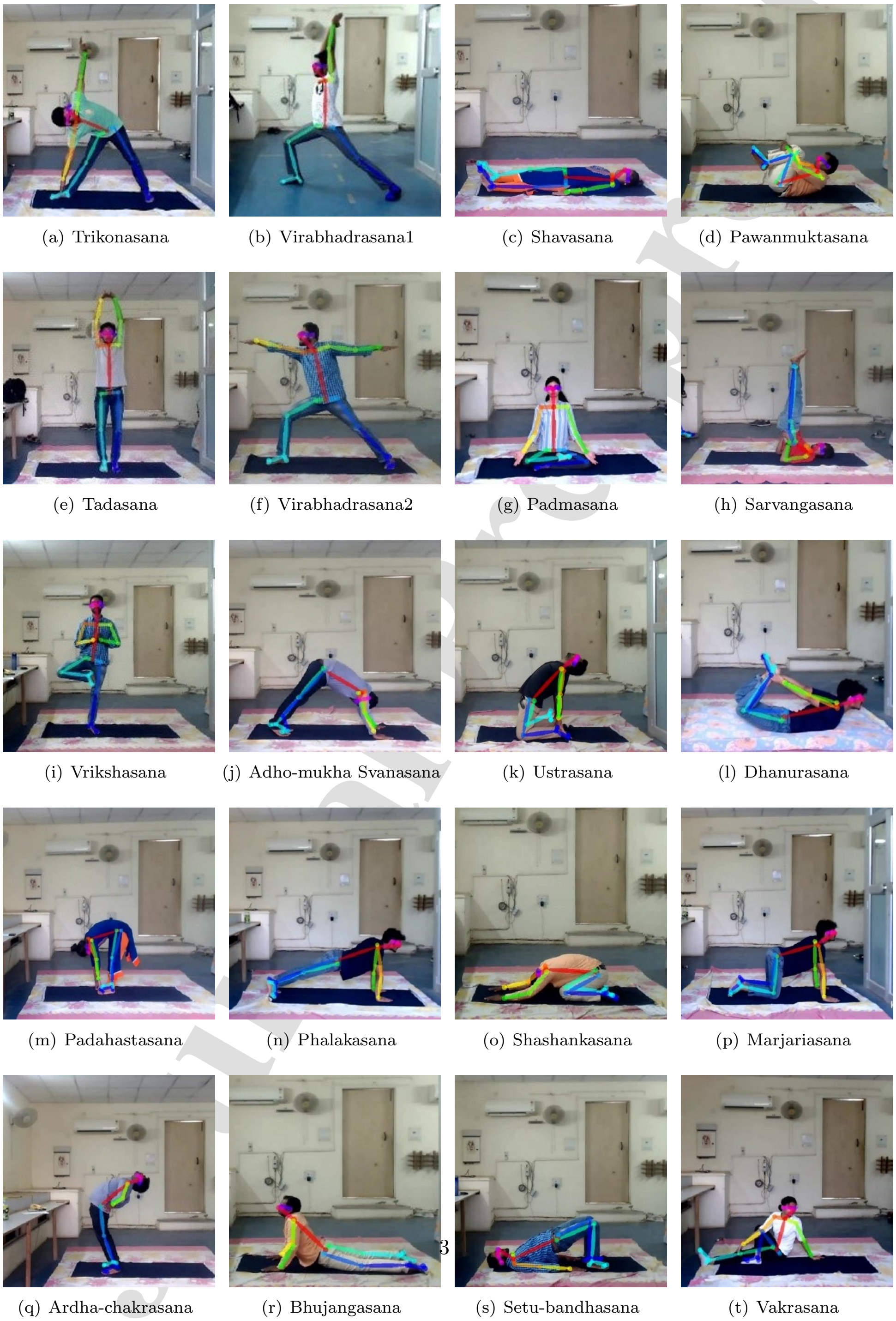
Email addresses: santosh.yadav@pilani.bits-pilani.ac.in (Santosh Kumar Yadav), f2016716@pilani.bits-pilani.ac.in (Aayush Agarwal), f2016636@pilani.bits-pilani.ac.in (Ashish Kumar), kamlesh.tiwari@pilani.bits-pilani.ac.in (Kamlesh Tiwari), hpandey@bournemouth.ac.uk (Hari Mohan Pandey), saakbar@ceeri.res.in (Shaik Ali Akbar)

1. Introduction

Yoga is an ancient Indian science that is in practice for long to unite the body, mind, and soul [1]. Fundamental aspects of yoga include asanas (physical postures), pranayama (breathing techniques), dhyana (meditation), mantras (chants), and sutras (wisdom teachings) [2]. The most common form of yoga in the western world is Hatha yoga, which is a combination of asanas (physical postures), pranayama (diaphragmatic breathing), and dhyana (meditation) [3]. It consists of multiple stretching exercises and prolonged physical poses, that elongate the key muscles and trigger the stretch receptors in joints, ligaments, and muscles [3, 4]. The regular practice of yoga asanas improves body alignment, agility, flexibility, muscular strength, endurance, relaxation, and overall fitness [4, 5].

Limited availability of yoga centers and high demand for proper instructors lead to a need for an expert yoga asanas recognition system that can analyze practitioner's postures and can provide valuable correction measures to the yoga performers on their asanas. To determine human postures, it is necessary to extract certain keypoints of the human skeleton [6, 7]. Popular skeletonization techniques, like thinning and distance transformation are sensitive to noise and have very high computational cost [8]. Recent pose estimation methods provide high-level human skeleton keypoints from a 2D image. These methods aim to localize body joint keypoints and/or finding the individual's body parts [9, 10]. Architectures following top-down approaches such as DeepPose [11], AlphaPose [12] and Mask R-CNN [13], first leverage a person detector on an image and then proceed with pose estimation for each person detected. However, there is no alternate for recovery in case of the failure of the person detector. Moreover, the computational cost and running-time increases with the increase in the number of people. In contrast, bottom-up approaches such as DeepCut [14], DeeperCut [15] and OpenPose [9, 16] first estimate the human body parts' locations in an image and then articulate a human pose on the basis of the parts detected earlier. Here, the prediction time is independent of the number of people present in an image frame.

Deep learning algorithms are getting popular for activity recognition due to their capability of automatic feature extraction. The convolutional neural networks (CNNs) are being preferred for spatial feature extraction while recurrent neural networks (RNNs) are being used for extracting sequential information present in the data [17]. The conventional RNN models suffer from the



(a) Trikonasana

(b) Virabhadrasana1

(c) Shavasana

(d) Pawanmuktasana

(e) Tadasana

(f) Virabhadrasana2

(g) Padmasana

(h) Sarvangasana

(i) Vrikshasana

(j) Adho-mukha Svanasana

(k) Ustrasana

(l) Dhanurasana

(m) Padahastasana

(n) Phalakasana

(o) Shashankasana

(p) Marjariasana

(q) Ardha-chakrasana

(r) Bhujangasana

(s) Setu-bandhasana

(t) Vakrasana

Figure 1: Twenty postures of yoga asanas considered in this paper.

problem of vanishing and exploding gradients [18]. To overcome these challenges and enable long-term dependencies, various RNN variants have been proposed such as long short-term memory (LSTM), gated recurrent unit (GRU), *etc.* [19]. Also, many works of literature utilized deep hybrid models (like ConvLSTM, Conv-SVM, *etc.*) for different activity recognition applications [20]. A few works have utilized 3D-CNNs for for extracting the spatiotemporal features from videos [21, 22, 23]. Karpathy *et al.* [22] proposed a 3D-CNN architecture for large-scale video-based sports action classification. Tran *et al.* [24] proposed an efficient 3D-CNN architecture named C3D. Varol *et al.* [23] stated that 3D-CNN's performance can be improved by expanding the temporal temporal lengths of input sequences for activity recognition.

Recognition of the yoga asanas falls within a general framework of computer vision called activity recognition. These systems aim to automatically determine human actions based on the data obtained from sensors [25, 26]. The applications of human activity recognition include video-surveillance, human-computer interaction, assisted-living, in-home health monitoring, *etc.* [27, 28, 29, 30]. Wang *et al.* [31] tried to address the issues with joint training of multi-stream model by proposing a new metric, overfitting-to-generalization ratio (OGR) to quantify the quality of training. They proposed a new method, Gradient Blending which generates weights to minimize OGR to improve the performance of multi-stream models. However, in the proposed YogNet model training of streams is done separately and fused later. Zhu *et al.* [32] proposed novel cross-layer attention and center attention layer. They fused multiple scales of different granularity based on their importance as determined by the attention framework, however, it still uses a single stream of RGB data. In contrast, our YogNet model is a 2 stream deep spatiotemporal method, which helps the model understand cross-orientation of various body parts. Wang *et al.* [33] proposed a framework to tackle egocentric action recognition problem, where every class is annotated by a combination of noun (object) and verb (action). The authors proposed three-stream network to classify the two parts, *i.e.* noun and verb. Yoga asana detection, when seen as an activity recognition problem, poses more challenge as it involves very complex body postures and twisting the body into multiple manners.

The proposed YogNet system consists of a two-stream network *i.e.* pose and RGB streams. First, the pose stream utilizes a keypoint detection technique followed by the formation of bounding boxes across the subject. The hybrid model of pose stream network consists of time-distributed CNNs and regularized LSTMs to train the yoga classifier, where, the CNN and LSTM are used for extracting postural and temporal features, respectively. Second, the RGB stream utilizes 3D-CNNs for spatiotemporal feature extraction from the raw RGB videos. It is a modified version of C3D [24] architecture of 3D-CNNs. Finally, the Softmax scores of the two streams are fused at the feature and decision levels for the final classification. The model is capable of realtime multi-person classification using a pseudo-tracking mechanism, specific to multi-person yoga pose detection and correction, wherein each yoga practitioner's keypoints are sorted based on their bounding boxes from left to right, as explained in Section 4.4. Further, posture correction techniques are applied, based on angles and distances between keypoints, to provide the correction measures in realtime, as described in Section 3.5. A schematic diagram of the proposed methodology is presented in Figure 2. The paper considers twenty asanas (as shown in Figure 1) and a 'No Activity' class, for actions that do not belong to any of the predefined classes, in YogNet.

In particular, the major contributions of this paper are highlighted as follows:

1. We propose a novel two-stream deep spatiotemporal architecture referred to as *YogNet*. The first stream incorporates time-distributed CNNs and regularized LSTMs, respectively, to perform postural feature extraction and to give temporal predictions. The second stream incorporates 3D-CNNs for extracting spatiotemporal features from RGB videos. The Softmax scores of the two streams are fused using feature and decision level fusion. The system is capable of recognizing multi-person yoga asanas in realtime and, can be used as a posture correction system. To the best of our knowledge, this is the earliest attempt using the two-stream network to recognize yoga asanas and provide multi-person posture corrections.
2. A new dataset is presented for yoga asanas recognition named as *YAR* dataset. This dataset has all features and enough complexity generally required for training of a system. We used this dataset during computer simulations to determine the effectiveness of YogNet in the area of yoga asana recognition and posture correction (described in Section 4.1).
3. Robustness of the YogNet is analyzed for different environments (with varying lighting conditions, places, *etc.*). Result reveals that the proposed system is robust in realistic scenarios with impressive accuracy. It is noted that YogNet can work in a multi-person setting for recognizing the asanas, even if participants practice different asanas at a time.
4. Extensive computer simulations are conducted using the *YAR* dataset containing 1206 videos. These videos have been collected from 16-participants using a 2D RGB web camera. Result reveals that YogNet model showed high value of accuracy using pose stream (77.29%) and

RGB (89.29%) stream. After fusion of both streams, the system was able to achieve even higher accuracy *i.e.* 96.31%. Aditionally, a comparison of the performance of Yadav *et al.* [34] (75.00%) and Jain *et al.* [35] (85.47%) on our proposed *YAR dataset* is presented (Table 9).

5. Finally, concluding remarks along with future research scope is presented.

The manuscript is organized as follows. Section 2 presents the literature review of yoga and posture recognition. Section 3 discusses the methodology of the proposed work, including data preprocessing, keypoint detection, yoga asanas recognition using pose stream and RGB streams, fusion of both streams, and posture correction models. Section 4 describes the data collection procedure and presents the experimental results and discussions. Finally, Section 5 presents the concluding remarks and future works.

2. Related Works

Few studies presented wearable sensors based yoga recognition systems, for example, Luo *et al.* [36] proposed a training system for yoga with an interface outfit containing 16 inertial measurement units (IMUs) and 6 tactors to detect motion features based on motion replication techniques. Wu *et al.* [37] proposed wearable sensors based yoga recognition and evaluation system for 18 static yoga postures. They collected a dataset from 11 participants, each wearing 11 IMUs on different parts of their body, to measure the yoga posture data with the quaternion format. Their model scored an accuracy of 95.39% by adopting a two-stage classifier mechanism, where the first one, backpropagation-artificial neural networks (BP-ANN) was used to divide yoga postures into different categories, while the second classifier, fuzzy C-means (FCM) was utilized to classify the postures in a category. However, wearing many sensors on different parts of the body is not convenient for the user in a practical scenario. Also, the system's obtrusiveness can affect the user's exercise.

There are some vision-based yoga recognition systems, for example, Rector *et al.* [38] developed an exergame, Eyes-Free Yoga, to teach six common postures to the blind and people with low-vision. They used a Kinect depth sensor for skeletal tracking and to give auditory feedback to the user. Patil *et al.* [39] proposed a 'yoga tutor' to detect the posture difference between a beginner and an expert yoga practitioner, using the speeded-up robust features (SURF) algorithm. However, capturing the unidirectional contour information was not enough for describing and comparing the poses precisely. Wu *et al.* [40] developed an expert system for yoga using problem-oriented analysis and decision expert system (POADES) [41], for providing training instructions using images and texts. However, there was no posture analysis conducted, so the user does not know if they are doing the asanas correctly.

Hsieh *et al.* [42] developed a yoga training system with the help of a web camera and image processing that can interact remotely using the internet. The system matched the distance transformation between the practitioner's silhouette and standard yoga posture to give an evaluation score. The authors collected the dataset with the help of six practitioners for 23 asanas and at least three videos were collected for each asana with overall 414 videos. Their system gives an evaluation score for pose recognition on a scale of 0 to 100. Around 86% difference between the computer and yoga teacher's evaluation scores lie within a range of -2.5 to 2.5.

Chen *et al.* [43] proposed a self-training system for yoga posture recognition and gave instruction for posture correction using visual instructions, to help the user in doing the asanas correctly. The contour information, skeletal features, and body coordinates were extracted using two Kinect depth sensors, one for front view and the other one for side view, by placing them perpendicular to each other. Chen *et al.* [8] proposed a yoga posture recognition system for self-training using a Kinect depth sensor camera. Their dataset contains 300 clips for twelve different yoga postures in which five yoga practitioners have performed each asana five times. The model first extracted the contour information and then applied the star skeleton method for posture representation and obtained an accuracy of 99.33%.

Likewise, Trejo *et al.* [44] and Pullen *et al.* [45] proposed yoga posture recognition using the Kinect depth sensor camera and AdaBoost algorithm for six asanas (accuracy >94.78%) and five asanas (accuracy >90%) respectively. Furthermore, Islam *et al.* [46] proposed a yoga recognition system using 15 different human body joint keypoints extracted from Kinect depth sensor images. Their dataset contains 45 videos, each of duration 5-8 seconds for three different asanas, namely Goddess squat, Warrior, and Reverse Warrior, performed by 5 different practitioners. Their model scored a test accuracy of 97%. Chen *et al.* [47] proposed a self-training system for yoga posture recognition to assist the practitioners in rectifying their postures if they do the asanas incorrectly. A Kinect depth sensor was used for recognizing twelve different asanas. However, their system makes separate models for each asana and calculates the features manually. Additionally, using a depth-sensor camera is not quite ubiquitous and may not be available in general homes.

Gochoo *et al.* [48] built an IoT-enabled yoga recognition system for 26 yoga postures using a deep CNN and three WSN nodes with an infrared sensor. Their system scored an average F1-score of 0.9854 and 0.9989 for one and three wireless sensor network (WSN) nodes, respectively. Maddala *et al.* [49] presented a 3D joint motion representation system for yoga postures using a joint angular distance map (JADM) encoding mechanism and a single-stream CNN model. They evaluated their model on a self-collected dataset using 9 cameras, 8 IR, and 1 RGB video for 42 yoga poses, and on two publicly available datasets (*i.e.* CMU [50] and HDM05 [51]). Their system scored cross-subject accuracies of 87.27% and 87.92%, and cross-view accuracies of 88.67% and 89.15%, on the CMU [50] and HDM05 [51] datasets, respectively. However, there were no posture correction instructions to the practitioner.

Recently, Verma *et al.* [52] proposed Yoga-82 dataset, which consists of 28.4k images for 82 classes of yoga *asanas*. They provided three-level hierarchy in their dataset, where top, middle, and class level hierarchies are 6, 20, and 82, respectively. They tested different variants of CNN architectures *i.e.* ResNet [53], DenseNet [54], MobileNet [55], and ResNext [56]. However, as yoga *asanas* consist of a sequence of multiple postures, recognizing *asanas* using a single image may lead to false results in a realistic scenario. In [34], a yoga recognition system was built to classify 6 different asanas. It used OpenPose [9] to generate 18 keypoints of the human body and passed them through a hybrid model of CNN and LSTM. The system scored a test accuracy of 99.38% after polling. However, the asanas were visually quite dissimilar to each other. Similarly, Yadav *et al.* [57] proposed activity recognition and fall detection system using pose estimation based classification network. However, detecting keypoints using pose estimation methods is hard for complex yoga asanas. Jain *et al.* [35] proposed a yoga pose recognition system using 3D-CNNs for 10 yoga *asanas*. They collected an in-house yoga dataset with the help of 27 individuals using smartphone cameras at 30 fps and 4K resolutions. However, they have not yet released their dataset to the public. Moreover, the above two systems [34, 35] were able to recognize single-person asanas only, and there were no feedback instructions for the corrective measures to the practitioner. Table 1 presents a summary of the literature study.

3. Proposed Methodology

This section describes the technical details of the proposed YogNet system. The system targets to recognize specific twenty asanas being performed by the practitioner in realtime with high accuracy. The system is capable of identifying multiple practitioners simultaneously performing dif-

Method	Description	Dataset Release?	Limitations
Rule based system, 2013, [43]	2 Kinect Sensors, 3 classes, 5 actors, 2 camera views, and 82.84% accuracy	No	Poor Performance
Rule based system with star skeleton features, 2014, [8]	1 Kinect Sensor, 12 classes, 5 actors, 2 camera views, and 99.33% accuracy	No	No posture correction instruction to the practitioner
Rule based feature calculation, 2017, [46]	1 Kinect Sensor, 3 classes, 5 actors, 1 camera view, and 97.00% accuracy	No	Classifying postures based on joint angles only
Rule based system with template star skeleton, 2018, [47]	1 Kinect Sensor, 12 classes, 5 actors, 2 camera views, and 94.30% accuracy	No	Sets different and complex rules for each posture
AdaBoost Algorithm, 2018, [45]	2 Kinect Sensors, 5 classes, 5 actors, 2 camera views, and 90.00% accuracy	No	No posture correction instruction to the practitioner
AdaBoost Algorithm, 2018, [44]	1 Kinect Sensor, 6 classes, 1 actor, 1 camera view, and 94.78% accuracy	No	The dataset consists 3 clips of a single practitioner only.
IoT based yoga recognition using Deep-CNN, 2018, [48]	3 WSN nodes with 1 IR sensor, 26 classes, 18 actors, 2 views, and 0.9989 f1-score	No	No posture correction instruction to the practitioner
Hybrid model using CNNs and LSTM, 2019, [34]	1 Web Camera, 6 classes, 15 actors, 1 camera view, and 99.38% accuracy	**Yes**	No posture correction instruction to the practitioner
Joint angular distance maps (JADMs) along with CNN, 2019, [49]	9 Cameras, 8 IR and 1 RGB Video, 42 classes, 10 actors, 10 camera views, and 88.25% accuracy	No	No posture correction instruction to the practitioner
BP-ANN and Fuzzy C-Means (FCM), 2019, [37]	11 wearable IMUs, 18 classes, 11 actors, and 95.39% accuracy	No	Highly obtrusive
3D-CNNs, 2020, [35]	Smartphone cameras, 10 classes, 27 actors, 1 camera view, and 91.15% accuracy	No	No posture correction instruction to the practitioner
Proposed YogNet (Ours)	1 RGB Camera, 20 classes, 16 actors, 4 camera views, and 96.31% accuracy	**Yes**	None

Table 1: Summary of the literature review and our proposed work.

ferent yoga asanas in a single go. At the same time, the system also recommends posture correction instructions to correct the asana poses if required.

The proposed approach has five key steps. In the first step, keypoint detection is performed using part affinity fields and bipartite matching. In the second step, the keypoints are fed to a hybrid model consisting of time-distributed CNN layers followed by the regularized LSTM layers. CNN layers capture the postural information and patterns whereas the LSTM layer captures the temporal information present in the data. In the third step, for the RGB stream, the RGB frames are preprocessed and passed to the 3D-CNNs for extracting the spatiotemporal features. In the fourth step, either the probability scores of the pose and RGB streams are fused using the decision level fusion, or spatiotemporal and postural features are fused together using feature-level fusion. Finally, the fifth step establishes the posture correction model. The final prediction and pose correction instructions are displayed to the respective practitioners in realtime. Figure 2 presents the schematic diagram of the proposed YogNet system. These steps are described in detail in further subsections.

3.1. Keypoint Detection and Data Generation

The videos of the yoga practitioners retrieved from the web camera are inputted to the pose detection model. The model uses two-branch multi-stage CNNs to process the images using a feedforward neural network for each stage. The first branch predicts the confidence maps for localizing human body parts in an image, and the second branch predicts part affinity fields to

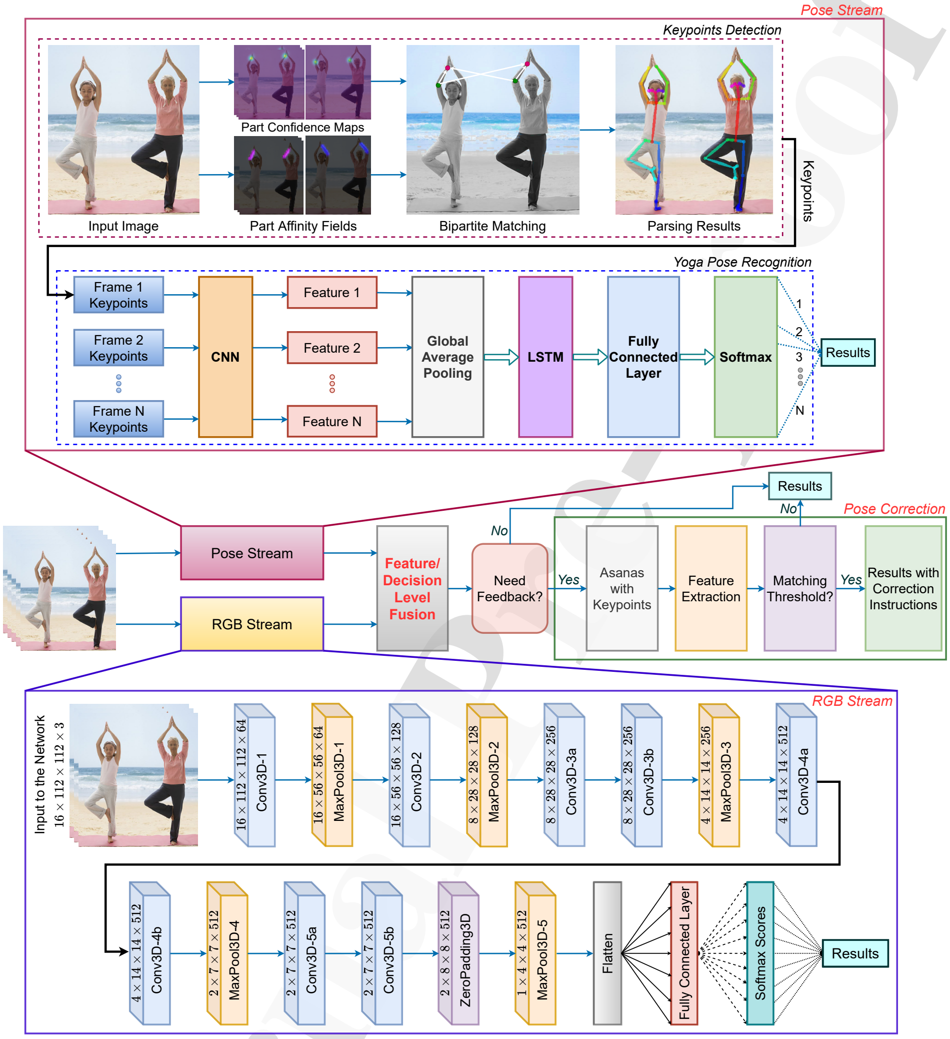


Figure 2: The schematic diagram of the YogNet system. The proposed approach of YogNet has four main components *i.e.* pose stream, RGB stream, fusion, and yoga pose correction. First, in the pose stream, human body keypoints are extracted using parts affinity fields and bipartite matching. The preprocessed data is fed to the deep hybrid (time-distributed CNNs and regularized LSTMs) neural networks model for yoga asanas recognition. Second, in the RGB stream, the preprocessed video frames are inputted to especially designed 3D-CNNs for spatiotemporal feature extraction. Third, the output of pose stream and RGB stream are fused using the feature and decision level fusion. Finally, in the fourth step, a pose correction technique is used for providing the correction measures to the yoga performer.

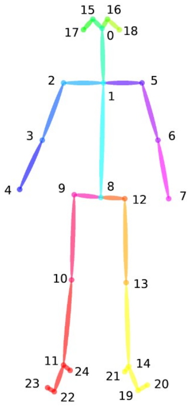


Figure 3: Twenty five keypoints detected [16]

No.	Keypoint
0	Nose
1	Neck
2	Right Shoulder
3	Right Elbow
4	Right Wrist
5	Left Shoulder
6	Left Elbow
7	Left Wrist
8	Mid Hip
9	Right Hip
10	Right Knee
11	Right Ankle
12	Left Hip
13	Left Knee
14	Left Ankle
15	Right Eye
16	Left Eye
17	Right Ear
18	Left Ear
19	Left Big Toe
20	Left Small Toe
21	Left Heel
22	Right Big Toe
23	Right Small Toe
24	Right Heel

Table 2: List of the keypoints.

associate the detected body parts with individuals in the image. Finally, the predictions of each stage and their corresponding image features are combined to output the 2D keypoints in the images [16]. Figure 3 presents the location of 25 keypoints, and Table 2 presents the mapping of the keypoint numbers to the joint locations.

For each frame of the practitioner's video, we extract the X and Y coordinates of joint locations. By this, we obtain X and Y coordinate vectors:

$$X = [x_1, x_2, x_3, ..., x_{25}] \tag{1}$$

$$Y = [y_1, y_2, y_3, ..., y_{25}] \tag{2}$$

These two vectors are then combined to form matrix $T = [(x_1, y_1), (x_2, y_2), (x_3, y_3), ...(x_{25}, y_{25})]$ to represent the coordinates for a frame. The matrices corresponding to each frame are then appended together in a window $(T_1, T_2, T_3, ..., T_N)$.

Further, for the generation of all batch data samples, generator functions are written which randomly selects a window comprising of 16 consecutive frames from each video (containing at least 300 frames). It is made sure that each video can only be selected once per epoch cycle, however, the order of videos and the window that each video provides, varies randomly in uniform distribution. In the case of pose stream, there may be several maligned frames, *i.e.* frames without

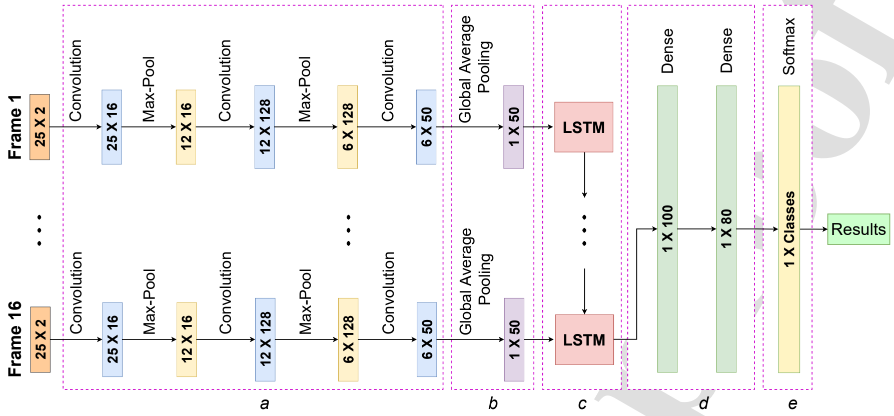


Figure 4: The architecture of the proposed network. This is an extended version of pose recognition block of *Figure 2*, where, a, b, c, d, and e represents the convolution layers, global average pooling, LSTM layer, fully connected layer, and Softmax layer, respectively.

any keypoints. In such cases, these frames are skipped and the generator function picks up the next good (non-maligned) frame until all 16 frames have been selected.

3.2. Yoga Asanas Recognition using Pose Stream

The proposed pose stream involves a combination of time-distributed CNNs and regularized LSTM layers. The CNN layer captures the joint location positions and salient postural features, while the LSTM layer analyses how the asanas are being performed over time and capture the temporal features for the asanas. These features are then passed through a fully connected layer and finally, the Softmax layer predicts the framewise probabilities of each yoga asana.

The model consists of 3 units of time-distributed CNN and max-pooling layers, with the CNN in the first unit comprising of 16 filters followed by the next two time-distributed CNN layers with 128 and 50 filters each, all having a kernel size of 3×3, and a ReLU activation function.

The output from the CNN layers is then flattened to one dimension (per frame in a training example) using a global average pooling layer, the output of which is passed to a regularized LSTM layer with 100-dimensional output and a unit forget bias of 0.2. LSTM layer helps in capturing the changes in the performance of various asanas over time such as the motion of keypoints and their temporal dependencies. The output of the regularized LSTM layer is then passed to a time-distributed fully connected layer with 80 hidden units having a dropout rate of 0.5 and then finally passed to the Softmax layer to predict the probabilities of each yoga asana. Adam optimizer is used for optimization with a custom learning rate policy. A threshold value is then chosen to get the prediction of the asanas performed by the practitioner. If the probability of an asana is above the threshold value (hyperparameter), the model correctly predicts that asana. However, in the case where probability scores are below the threshold, the system predicts 'No Activity'. Thus, there needs to be a minimal resemblance to the corresponding standard asana for the yoga performer to receive appropriate correction instructions. This is further elaborated in Section 4. Figure 4 presents the architecture of the proposed network.

3.3. Yoga Asanas Recognition using RGB Stream

The RGB stream of the proposed network utilizes a modified lightweight C3D [24] model as its backbone structure which uses 3D CNNs for extracting spatiotemporal features. The first two dimensions of 3D convolutions extract spatial features, while the third is responsible for temporal features. Each input frame of a video is resized to a size of 112×112. Further, a window comprising of 16 frames is extracted from a set of videos at random to form a single batch of input. A single batch consisting of 16 frames concatenated together is referred to as a window. A single video consists of at least 300 frames (10 seconds) as videos are shot at 30 FPS. However, only 16 consecutive frames are picked out of the collection per epoch at random. Thus a window has collective dimension of $16 \times 112 \times 112 \times 3$. A batch size of 32 windows is used for training. The model has a total of eight 3D-CNN layers and five max-pooling layers. 3D-CNN layer filters are of size $3 \times 3 \times 3$, with a stride of $1 \times 1 \times 1$. The pooling layers however, are of size $2 \times 2 \times 2$ with stride $2 \times 2 \times 2$ except for the first pooling layer which has a kernel size of $1 \times 2 \times 2$ and a stride of $1 \times 2 \times 2$. All the convolution layers have no regularization, only the fully connected layers are regularized with L2 and L1 regularization of 0.1 and 0.01, respectively. ReLU activation function is used for all the layers except for the classification layer. The SGD optimizer is used to train the model for 500 epochs with a learning rate of 0.001 for the first 400 epochs, 0.0001 for the last 100 epochs. The extracted features are then passed through fully connected layers with a dropout of 0.5 with softmax at the end for classification.

3.4. Fusion of Pose and RGB Streams

The fusion of the two streams helps in combining the postural and spatiotemporal features extracted in the corresponding streams. In this paper, two types of fusions have been explored *i.e.*

Decision level fusion and *Feature level fusion*.

Decision level fusion was performed by fusing the Softmax scores of the two streams *i.e.* pose stream and RGB stream using max, average and weighted-average fusion techniques. Since the decision level fusion does not require a separate classifier, it is faster than other fusion techniques [58]. The weighted average fusion (Equation 4) and max fusion (Equation 5) equations can be described as follows, with average fusion equation (Equation 3) derivable from (Equation 4), if $\forall j \; w_j = 1$, where j refers to the j^{th} stream.

$$c = \underset{i}{\operatorname{argmax}} \left(\frac{\sum_{j=1}^{n} p_j(\hat{c}_i|x_j)}{n} \right) \tag{3}$$

$$c = \underset{i}{\operatorname{argmax}} \left(\frac{\sum_{j=1}^{n} w_j \times p_j(\hat{c}_i|x_j)}{n} \right) \tag{4}$$

$$c = \underset{i}{\operatorname{argmax}} \left(\max_{j} p_j(\hat{c}_i|x_j) \right) \tag{5}$$

where, c denotes the predicted class, $p_j(\hat{c}_i|x_j)$ is the probability of class c_i on input x_j belonging to stream j, and w_j is weight associated with stream j in (Equation 4) such that, $\sum_{j=1}^{n} w_j = 1$, while n is the number of streams.

In the average fusion, the probability scores of each activity returned by the Softmax layer of each stream were averaged to give the final prediction. The maximum fusion technique takes the prediction from the stream, which was more confident in its prediction, while the weighted average fusion scales the Softmax scores according to the importance levels of both streams (using weights), before combining the softmax scores.

Additionally, feature level fusion was also performed on two streams, by concatenating the final layers of both streams to form a single column vector and then passing the features to a trainable network of fully connected layers. The architecture that performed best was composed of a dense neural network (DNN) of size 100 × 50 × 20.

3.5. Yoga Asanas Correction

Different joint angles and distance comparisons have been used for postural error detection. The thresholds for common postural errors for different asanas can be different (Table 3) because the skeletal structure of keypoints is formed using straight lines (not curved lines), whereas the human

Sr. No.	Asana	Side	Left Elbow (p_5, p_6, p_7)	Right Elbow (p_2, p_3, p_4)	Left Knee (p_{12}, p_{13}, p_{14})	Right Knee (p_9, p_{10}, p_{11})	Waist Angle ($s_{mid}, w_{mid}, l_{mid}$)
1	Adho-mukha -svanasana	Left	$< 160^\circ$ Straighten your left elbow		$< 160^\circ$ Straighten your knees		$> 100^\circ$ Move your hands and legs close to each other
2	Ardha-chakrasana	Left					$> 145^\circ$ bend your waist more
3	Dhanurasana	Right					$> 160^\circ$ Lift your chest more
4	Marjariasana	Right		$< 170^\circ$ Straighten your elbows		$> 100^\circ$ Bend knee more	
5	Padahastasana	Left	$< 160^\circ$ Straighten your elbow		$< 170^\circ$ Straighten your knees		$> 70^\circ$ Bend more
6	Phalakasana	Left	$< 160^\circ$ Straighten your elbow		$< 170^\circ$ Straighten your knees		$< 170^\circ$ Line hips with back and legs
7	Shashankasana	Right				$> 35^\circ$ Lower your hips	
8	Tadasana	Front	$< 150^\circ$ Straighten your left elbow	$< 150^\circ$ Straighten your right elbow			
9	Trikonasana	Left	$< 165^\circ$ Straighten your left elbow	$< 165^\circ$ Straighten your right elbow	$< 165^\circ$ Straighten your left knee	$< 165^\circ$ Straighten your right knee	
10	Virabhadrasana1	Left (left leg bend)	$< 160^\circ$ Straighten your left elbow		$> 120^\circ$ Bend left knee more	$< 160^\circ$ Straighten your right knee	
11	Virabhadrasana1	Left (right leg bend)	$< 160^\circ$ Straighten your left elbow		$< 160^\circ$ Straighten your left knee	$> 120^\circ$ Bend right knee more	
12	Virabhadrasana2	Left (left leg bend)	$< 160^\circ$ Straighten your left elbow	$< 160^\circ$ Straighten your right elbow	$> 120^\circ$ Bend left knee more	$< 160^\circ$ Straighten your right knee	
13	Virabhadrasana2	Left (right leg bend)	$< 160^\circ$ Straighten your left elbow	$< 160^\circ$ Straighten your right elbow	$< 160^\circ$ Straighten your left knee	$> 120^\circ$ Bend right knee more	

Table 3: Pose correction instructions for asanas and the joint angle threshold values.

body is not a structure of straight lines. So, the bend in a body part can not be fully mapped to angles among straight lines. However, a threshold can be established for a particular postural defect. The threshold values for a pose depends on the sensitivity between angles and the body curves.

Unlike all other angles, waist angle $(s_{mid}, w_{mid}, l_{mid})$ is dependent on both left and right sides of the body parts, and it is usually calculated for yoga asanas in left or right side views, so the keypoints like p_1, p_2, p_5 are expected to overlap in an ideal side-view, thus allowing us to take any one of these for angle calculation. However, in practical scenarios, people are not in an ideal side-view, so these keypoints may not overlap. We take the average of these keypoints, to account for variations in the coordinates.

If p_i denotes the i^{th} keypoint number (Figure 3 and Table 2), then the coordinates for waist angle can be calculated with the following equations:

$$S_{mid} = \frac{p_1 + p_2 + p_3}{3} \quad (6)$$

$$w_{mid} = \frac{p_9 + p_8 + p_{12}}{3} \quad (7)$$

$$l_{mid} = \frac{p_{10} + p_{13}}{2} \quad (8)$$

where s_{mid} represents midpoint at shoulder, w_{mids} represents midpoint at the waist and l_{mid} represents the midpoint between the left and right knee.

The thresholds of different angles and the corresponding instructions for asanas are mentioned in Table 3. The posture correction instructions for the Tadasana, Virabhadrasana1, Virabhadrasana2, and Vrikshasana also depends upon the following conditions:

1. *Tadasana-front:*
   If $d_{waist} < d_{feet}$: Put your feet together
   where, $d_{waist} = |x_{coord}(p_9) - x_{coord}(p_{12})|$ and $d_{feet} = |x_coord(p_{11}) - x_{coord}(p_{14})|$
2. *Virabhadrasana1-left-side:*
   $x_{coord}(p_{13}) > x_{coord}(p_{10})$: Right knee is bent
   $x_{coord}(p_{13}) < x_{coord}(p_{10})$: Left knee is bent
3. *Virabhadrasana2-left-side:*
   $x_{coord}(p_{13}) > x_{coord}(p_{10})$: Right knee is bent
   $x_{coord}(p_{13}) < x_{coord}(p_{10})$: Left knee is bent
4. *Vrikshasana-front:*
   - Right knee is bent: $y_{coord}(p_{24}) < y_{coord}(p_{21})$
     - If the right ankle is below the left knee *i.e.* $y_{coord}(p_{24}) > y_{coord}(p_{13})$: Bend the knee more
   - Left knee is bent: $y_{coord}(p_{24}) > y_{coord}(p_{21})$
     - If the left ankle is below the right knee *i.e.* $y_{coord}(p_{21}) > y_{coord}(p_{10})$: Bend the knee more

4. Experimental Results and Discussion

This section consists of four subsections. The first subsection describes the data collection procedure. The second subsection presents the experimental settings. The third subsection presents the evaluation results using the pose stream, RGB stream, and by fusing both streams. Finally, the fourth subsection presents the realtime pose detection and correction results for single and multiple practitioners.

4.1. Data Collection

The *YAR dataset* was collected for specific 20 asanas (Figure 1) with the help of 16 participants (11 males and 5 females). A single 2D RGB Logitech web camera (HD 1080p) was used in the data collection with a frame rate of 30 frames per second (fps) and a resolution of 1280 × 720. The camera was kept in a static position at a height of 5 feet above the ground and the asanas were performed at a distance of 3-8 meters from the camera. The age of participants varies from 22 years to 30 years. The age, gender, height, and weight details are mentioned in Table 4. All the asanas were performed under the guidance of an expert yoga teacher. However, most of the participants were not regular yoga practitioners. The specific twenty asanas include Trikonasana (triangle pose), Tadasana (mountain pose), Vrikshasana (tree pose), Padahastasana (standing forward bend), Ardha-chakrasana (half wheel pose), Virabhadrasana1 (warrior pose-1), Virabhadrasana2 (warrior pose-2), Adho-mukha svanasana (downward-facing dog), Phalakasana (plank pose), Bhujangasana (cobra pose), Shavasana (corpse pose), Padmasana (lotus pose), Ustrasana (camel pose), Shashankasana (rabbit pose), Setu-bandhasana (bridge pose), Pawanmuktasana (wind-relieving pose), Sarvangasana (shoulder stand), Dhanurasana (bow pose), Marjariasana (cat stretch pose), and Vakrasana (twisted pose) as shown in Figure 1.

Each practitioner performed all the asanas in front of the camera with a different view that includes front (F), back (B), left (L), and right (R) sides. The duration of each asana is for about one minute. Table 5 presents the *YAR dataset* details in terms of the specific time duration details, number of participants, number of videos, and the total duration for each asana. The variation in the number of participants is due to the fact that a few asanas are difficult to perform. The most difficult asanas experienced by some participants include Sarvangasana, Ustrasana, and Phalakasana. The total length of 1206 videos for training is 06 hours, 06 minutes, and 20 seconds. Table 6 presents the comparison of the recent video-based yoga action recognition datasets. The

User ID	Age (year)	Gender (M/F)	Height (cm)	Weight (kg)
UID-1	21	M	175.26	72
UID-2	21	M	185.42	79
UID-3	30	M	160.02	75
UID-4	24	F	160.02	65
UID-5	26	M	160.02	73
UID-6	20	F	144.78	59
UID-7	26	F	157.48	71
UID-8	22	F	162.56	83
UID-9	21	M	170.18	72
UID-10	28	M	189.15	87
UID-11	24	F	162.56	62
UID-12	21	M	187.96	82
UID-13	27	M	164.09	75
UID-14	26	M	160.02	70
UID-15	29	M	157.48	68
UID-16	22	M	167.64	66

Table 4: Anonimized demographic detail of the participants.

Sr. No.	Asana	Side wise Duration (in seconds)				No. of People	No. of Videos	Duration (in seconds)
		Left	Right	Front	Back			
1	Trikonasana	10	10	20	20	16	64	1107
2	Tadasana	10	10	30	10	16	64	1088
3	Vrikshasana	10	10	30	10	16	64	1087
4	Padahastasana	20	20	10	10	15	60	1012
5	Ardha-chakrasana	20	20	10	10	16	64	981
6	Virabhadrasana1	20	20	10	10	16	63	1054
7	Virabhadrasana2	20	20	10	10	16	65	1111
8	Adho-mukha svanasana	20	20	10	10	16	64	1133
9	Phalakasana	20	20	10	10	15	58	951
10	Bhujangasana	20	20	10	10	16	64	1110
11	Shavasana	20	20	10	10	16	63	1209
12	Padmasana	10	10	40	N/A	16	48	1184
13	Ustrasana	20	20	10	10	15	59	1074
14	Shashankasana	20	20	10	10	16	57	1104
15	Setu-bandhasana	20	20	10	10	15	59	1082
16	Pawanmuktasana	20	20	10	10	16	64	1166
17	Sarvangasana	20	20	10	10	12	45	890
18	Dhanurasana	20	20	10	10	16	62	1162
19	Marjariasana	20	20	10	10	16	64	1262
20	Vakrasana	20	20	10	10	16	55	1213
		Total number of videos and duration:					1206	21980

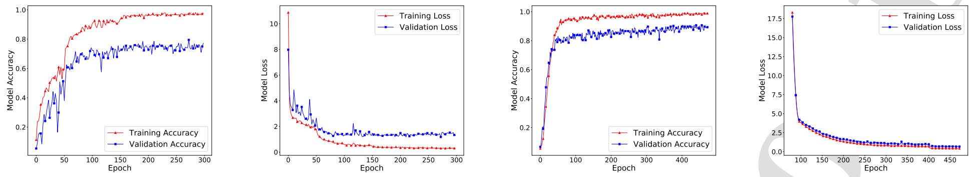
Table 5: Time duration, number of participants and number of videos for each asana.

Sr. No.	Reference	Year	Classes	Subjects	Videos	Frames	Viewpoints	Duration (In Seconds)	Is Public?
1	[34]	2019	6	15	88	111,750	1	3965	Yes
2	[35]	2020	10	27	261	68,190	1	2273	No
3	*YAR*	Proposed	20	16	1206	677,419	4	21980	Yes

Table 6: Comparison of the proposed *YAR* dataset with recent video based yoga action recognition datasets

dataset has been made publicly available to the research community and can be accessed using link: https://data.mendeley.com/datasets/k842kz6v4n/draft?a=f152b926-8be3-4c78-8f81-fe69ce636965.

The actors themselves are divided in 4:1 ratio (training and validation : test), such that 13 actors were seperated for the training and validation bucket while the remaining 3 actors were selected for the test bucket at random. It was made sure that all 3 actors belonging to the test bucket were able to perform all asanas. Validation and training data shared the same set of actors in their data pool, but were still disjoint on the exercise videos performed by those actors, from where the sample window of continuous frames were picked. It was made sure that the set of videos were divided approximately in 3:1:1 ratio of training, validation, and test, respectively. In case such



(a) Pose Stream Accuracy (b) Pose Stream Loss (c) RGB Stream Accuracy (d) RGB Stream Loss

Figure 5: Model accuracy and loss curves using Pose and RGB streams of YogNet.

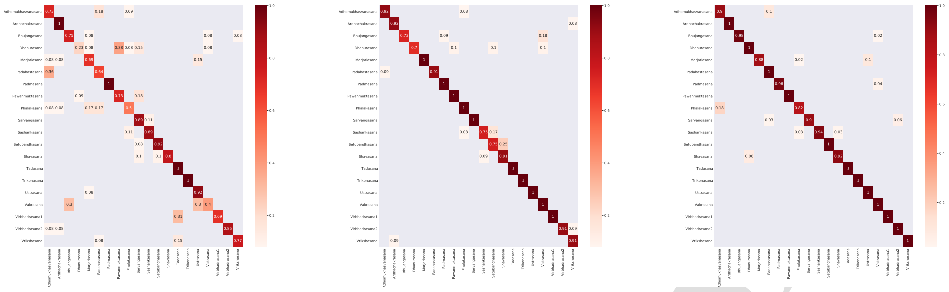
a division was not possible because of fewer actors in training and validation bucket, the count of validation set was made to sacrifice in favor of training set.

4.2. Experimental Settings

As pose estimation requires a high-performance computing device setup, a machine with an Intel-Xeon processor with 32 GB RAM and Nvidia Titan X GPU was used for this task. We used an open-source library for the keypoint detection called OpenPose [16]. The model has been compiled using Keras API with TensorFlow backend. The input to our pose stream is of size $16 \times 25 \times 2$, and $16 \times 112 \times 112 \times 3$ for RGB stream. The first dimension for both streams having value 16 represents the window size. In pose stream, 25 represents the number of joint locations and 2 represents the X and Y coordinates of each joint location, while in RGB stream the last 3 dimensions refer to an RGB image of size 112×112.

The streams were run for 300 epochs and 500 epochs, respectively, with a batch size of 32 where training took around 29 seconds per epoch and 22 seconds per epoch, respectively. The best scoring weights after every epoch, having the highest validation accuracy, were stored and later used for prediction. The model also uses a learning rate decay controller with a custom policy which decreases the learning rate when set epoch boundaries are achieved. Figure 5 presents the accuracy and loss curves of pose and RGB streams alongside epoch numbers.

From the accuracy curves in Figure 5(a) and 5(c), we can see that both train and validation accuracy increases with time and asymptotically reaches a steady-state value, with train accuracy above validation accuracy. From the graph, it can be seen that overfitting is minimal at 272 epochs and 421 epochs respectively, as the validation curve saturates and the difference between the training



(a) Pose Stream Heatmap (b) RGB Stream Heatmap (c) Fusion Heatmap

Figure 6: Confusion matrix for 20 asanas using pose stream, RGB stream, and fusion of both streams on the *YAR* dataset.

loss and the validation loss is minimum. Therefore, while training the model, early stopping is used as a regularization mechanism for consequently stopping the training at 272 epochs and 421 epochs, respectively. Figure 5(b) and 5(d) shows that training loss and the validation loss decreases with epoch numbers.

4.3. Model Evaluation

This subsection presents the model evaluation performance of the YogNet system using the pose stream, RGB stream, and by fusing both streams.

4.3.1. Pose Stream Results

The *YAR dataset* comprises yoga asanas performed from all four angles (*i.e.* front, back, left, and right). During the training phase, the dataset was divided into 20 classes, each class containing all four sides of the respective asanas. After training for 300 epochs with a batch size of 32, the model could only achieve training, validation, and test accuracies of 96.59%, 79.46%, and 77.23%, respectively. However, this model was unable to perform any better using this strategy on the dataset even after using multiple regularization techniques aggressively. The classification score using this strategy is presented in Table 7.

A few factors accounting for variance in the model are as follows:

1. *Presence of Inter-asana Similarity:* As the model contains all sides of asanas in each class, many of these asanas from different classes are visually non-differentiable from each other.

Sr.	Asana Name	Precision	Recall	F1-Score
1	Adho-mukha svanasana	0.8409	0.9024	0.8706
2	Ardha-chakrasana	1.0000	1.0000	1.0000
3	Bhujangasana	1.0000	0.9750	0.9873
4	Dhanurasana	0.9268	1.0000	0.9620
5	Marjariasana	1.0000	0.8750	0.9333
6	Padahastasana	0.8864	1.0000	0.9398
7	Padmasana	1.0000	0.9693	0.9811
8	Pawanmuktasana	1.0000	1.0000	1.0000
9	Phalakasana	0.9429	0.8250	0.8800
10	Sarvangasana	1.0000	1.0032	0.9492
11	Shashankasana	1.0000	0.9394	0.9688
12	Setu-bandhasana	1.0000	1.0000	1.0000
13	Shavasana	0.9722	0.9211	0.9459
14	Tadasana	1.0000	1.0000	1.0000
15	Trikonasana	1.0000	1.0000	1.0000
16	Ustrasana	0.8974	1.0000	0.9459
17	Vakrasana	0.9429	1.0000	0.9706
18	Virabhadrasana1	1.0000	1.0000	1.0000
19	Virabhadrasana2	0.9394	1.0000	0.9688
20	Vrikshasana	1.0000	1.0000	1.0000

Table 7: Precision, Recall and F1-Score after fusion of the two streams.

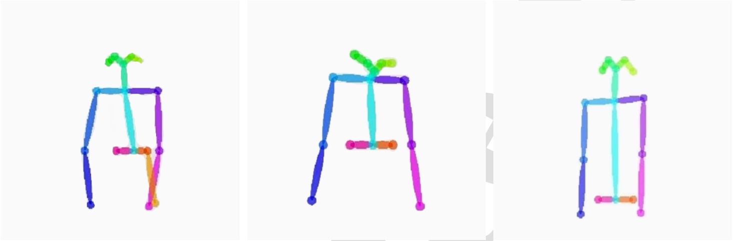


Figure 7: Body Keypoints from the front side for Marjariasana, Phlakasana, and Bhujangasana.

For example, when observed from the front-side or back-side Marjariasana, Phalakasana and Bhujangasana are nearly similar and thus non-differentiable for an average observer with basic knowledge of asanas, as shown in Figure 7. A similar case can be observed in Shashankasana and Shavasana or Dhanurasana and Pawanmuktasana, *etc.*

2. *Presence of Intra-asana Variability:* As the *YAR dataset* contains all sides of asanas in each class, most of these asanas are visually very dissimilar amongst themselves when observed from different sides, and sometimes even unidentifiable as a pose of that asana when observed from these sides as shown in Figure 8.

3. *Limitations due to Openpose:* OpenPose [16] can detect a practitioner when he/she is perfectly visible and is standing front-faced to an RGB camera, while it tends to present minor fluctuations when the practitioner is side-faced or in back-angle positions. However, it shows clear limitations when the practitioners are performing complex yoga asanas that require multiple twisting and turning sequences, or when the person is in an inverted upside-down pose.

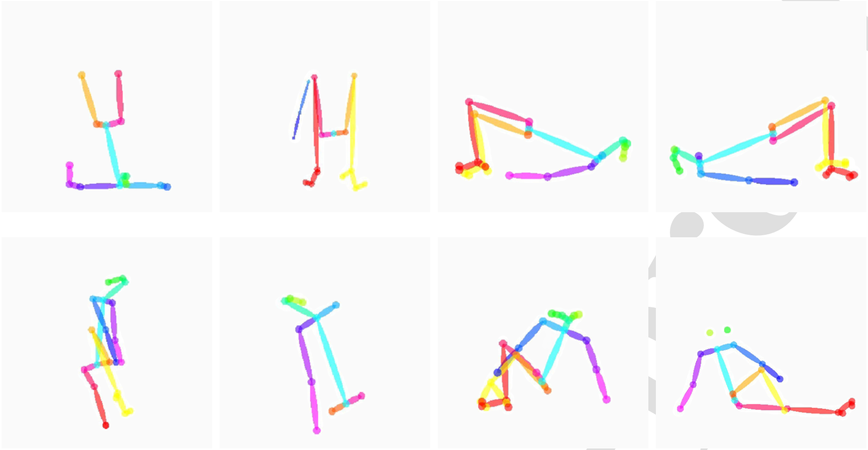


Figure 8: Body keypoint structures for Setu-bandhasana (front, back, left and right) and Vakrasana (front, back, left and right).

In such cases, a set of keypoints go undetected for several frames at once, which leads to a drop in the quality of the training dataset as in the case of Sarvangasana and Ustrasana as shown in Figure 9. Another kind of noise in this library is over-sensitivity towards person detection, wherein it detects objects like chair, shoes, *etc.* to be partial humans. A problem arises in this case when complex asanas are performed. It often detects these complex asanas with lower confidence on keypoints, where the average thresholding mechanism may not give proper results.

The presence of *Inter-asana similarity* and *Intra-asana variability* as well as limitations in keypoint detection by Openpose, are significant factors causing poor generalization of the YogNet model.

4.3.2. RGB Stream Results

The RGB stream achieved training, validation, and test accuracies of 98.72%, 91.51%, and 89.29%, respectively on *YAR dataset.* The models were trained for 400 epochs with each batch comprising of 32 window sets. The Figure 6 shows normalized confusion matrix on the test dataset of *YAR dataset*, wherein the model performs remarkably well on most asanas, with the exception of a few asanas such as Sashankasana, Setubandhasana, Dhanurasana, and Bhujangasana. The loss

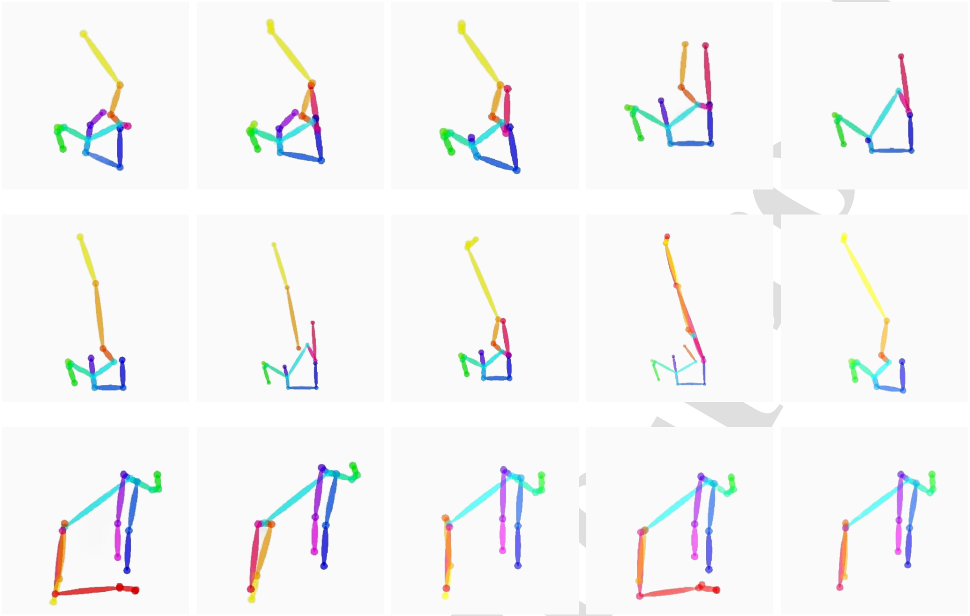


Figure 9: Body keypoint structures for Sarvangasana (first and second rows) and Ustrasana (third row) for ten and five continuous frames, respectively.

and accuracy curve of training the RGB stream can be seen in the Figure 5.

It generalizes well on *YAR dataset* with the exception of the above-mentioned asanas, as these asanas resemble each other in many angles and most of them share the same starting pose, for example - Shashankasana and Setubandhasana have similar structures, while one is face-up, other is face down. Similarly, Setubandhasana and Shavasana have exactly the same structure if not for the angle that the legs form with the ground. Similar comparisons can be drawn out for other asanas as well. However, pose stream clearly performs better on these asanas, as it is able to differentiate between them by the clear difference of angles (postural feature) between the corresponding keypoints that are present, and does not rely simply on spatial feature extraction by the convolutional network.

Sr.	Reference	Dataset Name	Their Result	YogNet Result
1.	[34]	Deep Yoga	99.38%	99.67%
2.	[35]	in-house Yoga	91.15%	93.29%
3.	[57]	ADLF	89.64%	93.33%
4.	Proposed	YAR	NA	96.31%

Table 8: Comparision results on different datasets using our proposed YogNet model.

4.3.3. Results After Fusing the Pose and RGB Streams

It can be seen from Subsections 4.3.1 and 4.3.2 that both the streams alone have some deficiencies that prevent them from being a total package for the prediction of yoga asanas. However, it can be seen that many of the features these two streams provide are complementary to each other. Decision level fusion and feature level fusion are both an effort to bring together the streams in an optimal manner to extract most of the different kinds of information available and perform high-quality classification of the same.

In the current paper, we have used 3 types of Decision level fusion techniques, namely Average, Weighted Average, and Max fusion, while we use DNNs for feature level fusion. In decision level fusion techniques *average fusion* achieved training, validation, and test accuracies of 94.46%, 92.41% and 92.03%, respectively, *weighted average fusion* achieved training, validation, and test accuracies of 97.58%, 94.01% and 95.30%, respectively, and *max fusion* achieved training, validation, and test accuracies of 96.59%, 92.57% and 93.31%, respectively, on *YAR dataset.* Out of three techniques weighted average fusion achieved highest f1 score and accuracy percentages of the lot, with $w_{pose\ stream} = 0.3$ and $w_{C3D} = 0.7$. In feature level fusion techniques, the DNN architecture that performed the best fusion was of size 100 × 50 × 20, which achieved training, validation, and test accuracies of 98.29%, 96.43%, and 96.31%, respectively on *YAR dataset.*

Table 8 presents the comparision results of our proposed YogNet model on different datasets i.e., Deep Yoga [34], in-house Yoga [35], ADLF dataset [57], and on our proposed YAR dataset. Our proposed YogNet model outperforms the existing results on these datasets. Additionally, a comparison of models of Yadav *et al.* [34] and Jain *et al.* [35] from YogNet are peresented in Table 9 using *YAR dataset.* It can be seen that the fusion model clearly outperforms both the models. Both feature level fusion and decision level fusion with weighted average technique performed similarly, with the latter being the most efficient of the two. However, the weights have to be hyper-tuned in the case of the latter method, making feature level fusion with DNNs the most optimal choice for classification. The final normalized confusion matrix of DNN (of size 100 × 50 × 20) is presented in

Ref.	Model Name	Training	Validation	Testing
Other	Yadav *et al.* [34]	97.60%	76.34%	75.00%
	Jain *et al.* [35]	98.58%	87.50%	**85.47%**
Ours	Pose Stream	96.59%	79.46%	77.23%
	RGB Stream	98.72%	91.51%	89.29%
	Average Fusion	94.46%	92.41%	92.03%
	Max Fusion	96.59%	92.57%	93.31%
	Weighted Average Fusion	97.58%	94.01%	95.30%
	Feature Level Fusion	**98.29%**	**96.43%**	**96.31%**

Table 9: Training, validation and testing accuracies using different strategies on *YAR dataset*. **Blue** denotes the previous method's best result. **Red** denotes the best result.

Figure 6, alongside data on class-wise precision, recall and f1-score in Table 7.

4.4. Realtime Prediction

Realtime predictions for yoga asanas uses a single RGB camera as an input to the model, wherein each captured frame is passed first through a keypoint detection library. The JSON file obtained from the keypoint detection library contains keypoints for all detected people, so a separate list of keypoints is created for every detected performer. These keypoints are collected for 16 consecutive frames (window size), which are then inputted to the pose stream. Bounding boxes across subjects are formed from the keypoints themselves using a set of equations in 9.

$$\begin{aligned} X_{top-left} &= \min_j(x_i) \\ Y_{top-left} &= \max_j(y_i) \\ X_{bottom-right} &= \max_j(x_i) \\ Y_{bottom-right} &= \min_j(y_i) \end{aligned} \tag{9}$$

where, $X_{top-left}$ and $Y_{top-left}$ denote the x and y coordinates of top-left point of the bounding box, while $X_{bottom-right}$ and $Y_{bottom-right}$ denote the x and y coordinates of bottom-right point of the bounding box. The bounding box is then constructed for all the persons using the above equations and then extracted and processed to a size of 112×112 before passing the frames into the RGB stream.

The prediction is made for all the frames using a thresholding mechanism wherein windows (*i.e.* set of 16 frames) having maximum class probability less than the threshold value are predicted as having class 'No Activity'. The threshold value is tuned for an optimal prediction and is taken to

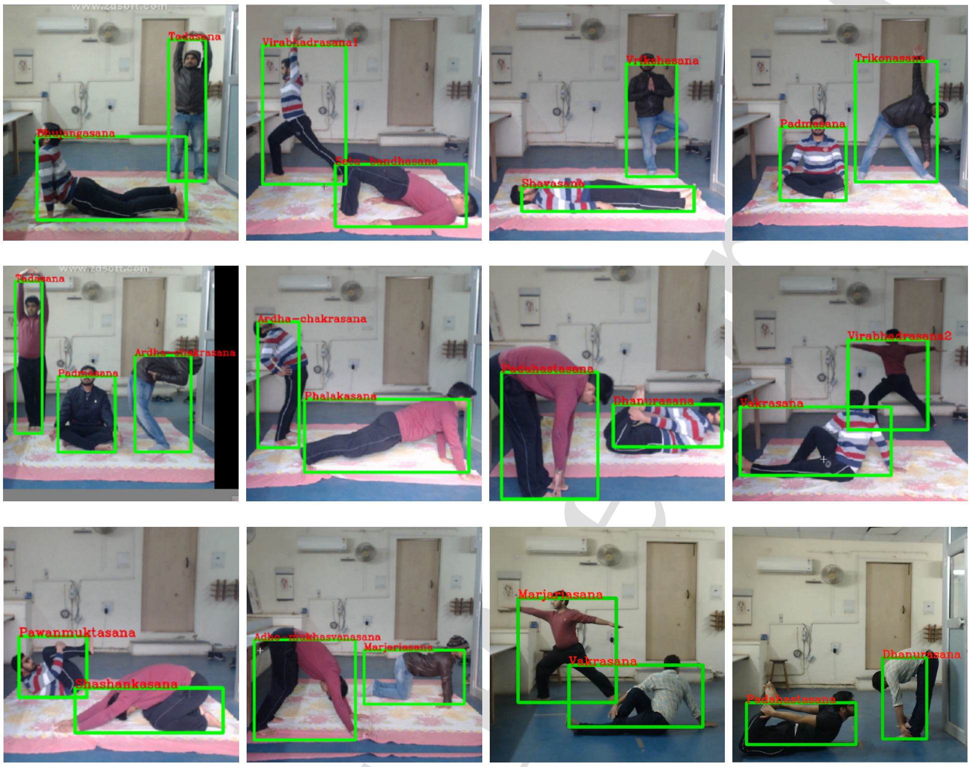


Figure 10: Realtime pose recognition of multiple practitioners. In the third row, the third and fourth columns show the cases where predictions are interchanged for practitioners. The demo videos of our realtime yoga action recognition using YogNet can be found on this link: https://data.mendeley.com/datasets/k842kz6v4n/draft?a=f152b926-8be3-4c78-8f81-fe69ce636965.

be 0.9 in our case.

The prediction results are displayed over the bounding box of corresponding yoga performers (Figure 10). The bounding boxes are created frame-wise as compared to predictions that are given over the whole window (16 frames). The top-left and bottom-right coordinates from the JSON files for a frame are used to form bounding boxes as mentioned in Equation 9.

A shortcoming of OpenPose [16] is that it does not provide any proper object-tracking mechanism within its bundle, due to which, in the JSON files generated by the key point detection library for each frame, a person at index ’0’ in one frame may not be at the same index ’0’ in the subsequent frames. For example, a person performing Tadasana alongside a Padmasana performer maybe sometimes wrongly labeled as performing Padmasana. Thus, keypoint extraction and collection of 16 frames for a practitioner based on just indexing in a multi-person environment may result in different yoga poses negatively affecting each other's predictions. In order to address this issue, a *pseudo-tracking mechanism* specific to the current problem is proposed, wherein each yoga practitioner's keypoints are sorted based on their bounding boxes from left to right. This tracking mechanism ensures that the collected 16 frame keypoints belong to the same practitioner and that the corresponding prediction is labeled to the correct person. If a practitioner does indeed change their order with another practitioner, all those whose orders have changed will have to wait for a maximum time equivalent for collecting 32 frames (16×2) for a correct prediction, once for the current session of frame collection from 16 frames to get over and secondly for collecting the new frames. The ordering of keypoints in JSON files, however, does not affect bounding box construction which is only dependent on keypoint coordinates. The bounding boxes are refreshed every frame, however, the labels are refreshed only at the end of 16 frames (*i.e* 1.875 times per second, assuming asana video is shot at 30 FPS).

Note that the RGB frame extraction process is done only after keypoint detection and bounding box formation from keypoints using Equation 9, and the order is decided based on the tracking mechanism. Thus the above method resolves the issue for both pose stream and RGB stream effectively.

Similar to pose prediction, pose correction has also been implemented for single as well as multiple practitioners. The pose correction includes twelve asanas which are complex to perform, while pose prediction was for twenty asanas. In realtime pose correction, the angles and distances between detected joints are used for suggesting corrective measures to the yoga performers, as mentioned in Section 3.5. The thresholding mechanism mentioned in the section 4.4 is used (Table 3) with attention to both single and multi-person pose correction in realtime, *i.e.* multiple pose correction recommendations can be given to multiple practitioners performing asanas side-by-side. The pose correction instructions are appended to the prediction and displayed on their respective bounding boxes. Figure 11 presents the realtime pose correction results for single and multiple people in different environments. The model was tested for three people simultaneously, in realtime, based on the number of people that can be accommodated in a single camera frame while performing the asanas. More people lead to more occlusion and hence, decrease the accuracy of our classifier.

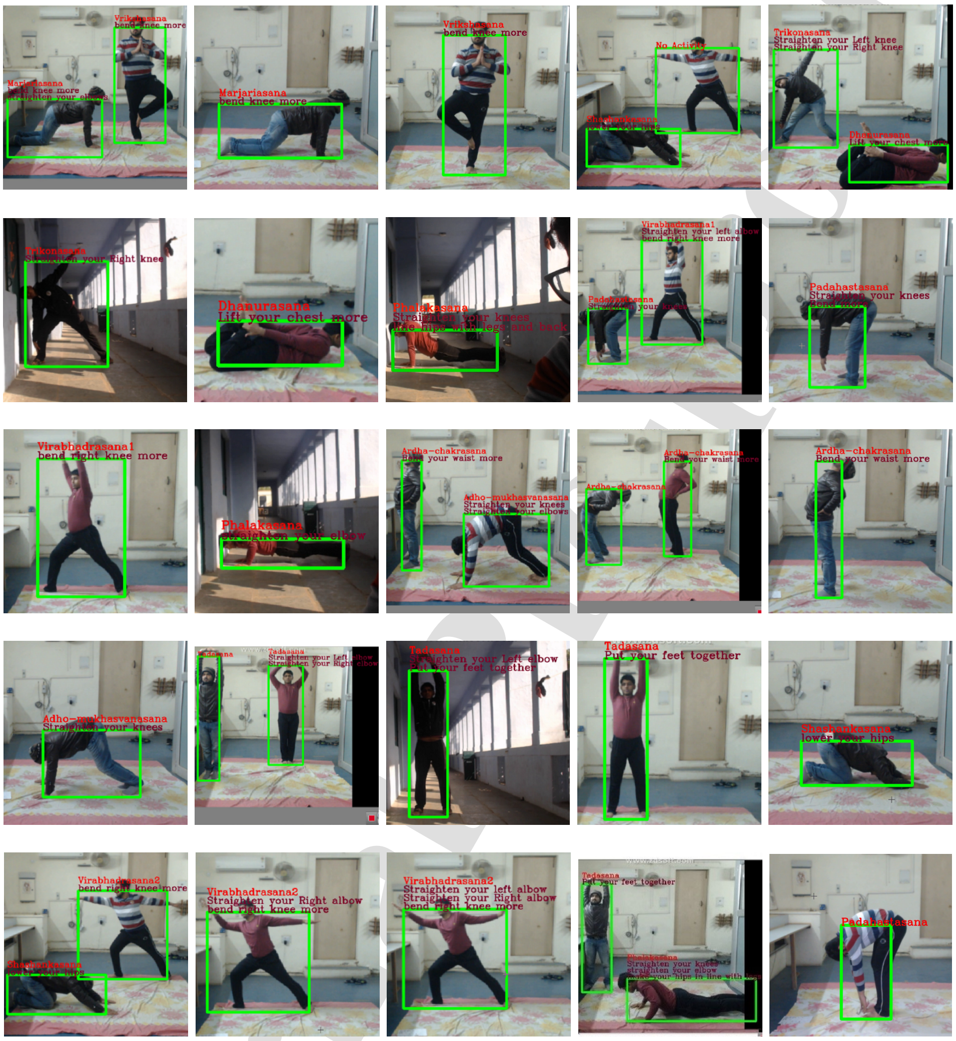


Figure 11: Realtime pose correction feedback to the single as well as multiple yoga practitioners in different environment doing similar and different asanas at a time. The system does not provide any feedback to the practitioner if they do the asana correctly (5^{th} row's 5^{th} column). The demo videos of our realtime yoga posture correction using YogNet can be found on this link: https://data.mendeley.com/datasets/k842kz6v4n/draft?a=f152b926-8be3-4c78-8f81-fe69ce636965.

Nevertheless, the method can be extended to any number of people.

5. Conclusion

This paper presented a realtime yoga expert system for yoga asanas recognition and posture correction using a 2D web camera. The system recognizes multiple practitioners' yoga asanas simultaneously. Also, it recommends posture correction measures to the practitioner if required. The system consists of a two-stream network. Pose stream network utilized a pose estimation method to detect keypoints, after which time-distributed CNNs have been employed for postural feature extraction followed by regularized LSTMs for temporal prediction. The second stream utilized 3D-CNNs for extracting the spatiotemporal features from raw RGB videos. Finally, feature level and decision level fusions of the two streams are performed to get final Softmax scores. The system was able to generalize well overall the 20 asanas with high accuracy. It must be noted that the system is able to recognize poses independent of posture correction instructions, both for single as well as multiple people. The multiple pose correction part of the proposed YogNet system comes into play only after the corresponding pose has been identified as it should bear a minimum resemblance (threshold) to the correct pose in order to receive corrections for that pose. Multiple pose correction works by calculating angles, distances, *etc.*, between keypoints and thresholding on those metrics. The system is able to provide multiple independent recommendations to all performing yoga practitioners simultaneously in realtime.

The work demonstrates an application of activity recognition. The same system can be applied to other similar applications such as exergaming, sports, surveillance, rehabilitation, and other healthcare applications. Since many yoga asanas involve very complex body postures and multiple body torsion and are sometimes unidentifiable as an activity of that asana by just visual means from several camera angles or pose detection algorithms, in future works, inertial sensors can be used alongside cameras for near-perfect recognition and correction.

Acknowledgements

Authors would like to thank anonymous reviewers and our parent organizations for extending their support for the betterment of the manuscript. Special thanks to Dr. J. L. Raheja (Chief Scientist, CSIR-CEERI, Pilani) and everyone who had participated in the data collection and appreciates the assistance provided by CSIR, India.

Funding Information

This research did not receive any specific grant from funding agencies in the public, commercial, or not-for-profit sectors.

Conflict of Interest

The authors declare that they have no conflict of interest.

References

[1] L. Ward, S. Stebbings, K. J. Sherman, D. Cherkin, G. D. Baxter, Establishing key components of yoga interventions for musculoskeletal conditions: a delphi survey, BMC complementary and alternative medicine 14 (1) (2014) 196.

[2] M. McCall, S. Thorne, A. Ward, C. Heneghan, Yoga in adult cancer: an exploratory, qualitative analysis of the patient experience, BMC complementary and alternative medicine 15 (1) (2015) 245.

[3] M. Van Puymbroeck, A. Walter, B. L. Hawkins, J. L. Sharp, K. Woschkolup, E. Urrea-Mendoza, F. Revilla, E. V. Adams, A. A. Schmid, Functional improvements in parkinson's disease following a randomized trial of yoga, Evidence-Based Complementary and Alternative Medicine 2018 (2018).

[4] V. Gaurav, Effects of hatha yoga training on the health related physical fitness, International Journal of Sports Science and Engineering 5 (03) (2011) 169–173.

[5] R. Lindquist, M. F. Tracy, M. Snyder, Complementary and alternative therapies in nursing, Springer Publishing Company, 2018.

[6] Y. Liu, H. Zhang, D. Xu, K. He, Graph transformer network with temporal kernel attention for skeleton-based action recognition, Knowledge-Based Systems (2022) 108146.

[7] S. K. Yadav, K. Tiwari, H. M. Pandey, S. A. Akbar, Skeleton-based human activity recognition using convlstm and guided feature learning, Soft Computing (2021) 1–14.

[8] H.-T. Chen, Y.-Z. He, C.-C. Hsu, C.-L. Chou, S.-Y. Lee, B.-S. P. Lin, Yoga posture recognition for self-training, in: International Conference on Multimedia Modeling, Springer, 2014, pp. 496–505.

[9] Z. Cao, T. Simon, S.-E. Wei, Y. Sheikh, Realtime multi-person 2d pose estimation using part affinity fields, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2017, pp. 7291–7299.

[10] S.-E. Wei, V. Ramakrishna, T. Kanade, Y. Sheikh, Convolutional pose machines, in: Proceedings of the IEEE conference on Computer Vision and Pattern Recognition, 2016, pp. 4724–4732.

[11] A. Toshev, C. Szegedy, Deeppose: Human pose estimation via deep neural networks, in: Proceedings of the IEEE conference on computer vision and pattern recognition, 2014, pp. 1653–1660.

[12] H.-S. Fang, S. Xie, Y.-W. Tai, C. Lu, Rmpe: Regional multi-person pose estimation, in: ICCV, 2017.

[13] K. He, G. Gkioxari, P. Dollár, R. Girshick, Mask r-cnn, in: Proceedings of the IEEE international conference on computer vision, 2017, pp. 2961–2969.

[14] L. Pishchulin, E. Insafutdinov, S. Tang, B. Andres, M. Andriluka, P. V. Gehler, B. Schiele, Deepcut: Joint subset partition and labeling for multi person pose estimation, in: Proceedings of the IEEE conference on computer vision and pattern recognition, 2016, pp. 4929–4937.

[15] E. Insafutdinov, L. Pishchulin, B. Andres, M. Andriluka, B. Schiele, Deepercut: A deeper, stronger, and faster multi-person pose estimation model, in: European Conference on Computer Vision, Springer, 2016, pp. 34–50.

[16] Z. Cao, G. Hidalgo, T. Simon, S.-E. Wei, Y. Sheikh, Openpose: realtime multi-person 2d pose estimation using part affinity fields, arXiv preprint arXiv:1812.08008 (2018).

[17] P. Wang, W. Li, C. Li, Y. Hou, Action recognition based on joint trajectory maps with convolutional neural networks, Knowledge-Based Systems 158 (2018) 43–53.

[18] G. Shen, Q. Tan, H. Zhang, P. Zeng, J. Xu, Deep learning with gated recurrent unit networks for financial sequence predictions, Procedia computer science 131 (2018) 895–903.

[19] I. D. Jordan, P. A. Sokol, I. M. Park, The expressive power of gated recurrent units as a continuous dynamical system (2018).

[20] J. Wang, Y. Chen, S. Hao, X. Peng, L. Hu, Deep learning for sensor-based activity recognition: A survey, Pattern Recognition Letters 119 (2019) 3–11.

[21] S. Ji, W. Xu, M. Yang, K. Yu, 3d convolutional neural networks for human action recognition, IEEE transactions on pattern analysis and machine intelligence 35 (1) (2012) 221–231.

[22] A. Karpathy, G. Toderici, S. Shetty, T. Leung, R. Sukthankar, L. Fei-Fei, Large-scale video classification with convolutional neural networks, in: Proceedings of the IEEE conference on Computer Vision and Pattern Recognition, 2014, pp. 1725–1732.

[23] G. Varol, I. Laptev, C. Schmid, Long-term temporal convolutions for action recognition, IEEE transactions on pattern analysis and machine intelligence 40 (6) (2017) 1510–1517.

[24] D. Tran, L. Bourdev, R. Fergus, L. Torresani, M. Paluri, Learning spatiotemporal features with 3d convolutional networks, in: Proceedings of the IEEE international conference on computer vision, 2015, pp. 4489–4497.

[25] A. Wijekoon, N. Wiratunga, S. Sani, K. Cooper, A knowledge-light approach to personalised and open-ended human activity recognition, Knowledge-based systems 192 (2020) 105651.

[26] X. Ji, Q. Zhao, J. Cheng, C. Ma, Exploiting spatio-temporal representation for 3d human action recognition from depth map sequences, Knowledge-Based Systems (2021) 107040.

[27] R. Poppe, A survey on vision-based human action recognition, Image and vision computing 28 (6) (2010) 976–990.

[28] D. Weinland, R. Ronfard, E. Boyer, A survey of vision-based methods for action representation, segmentation and recognition, Computer vision and image understanding 115 (2) (2011) 224–241.

[29] T. Özyer, D. S. Ak, R. Alhajj, Human action recognition approaches with video datasets—a survey, Knowledge-Based Systems 222 (2021) 106995.

[30] S. K. Yadav, K. Tiwari, H. M. Pandey, S. A. Akbar, A review of multimodal human activity recognition with special emphasis on classification, applications, challenges and future directions, Knowledge-Based Systems (2021) 106970.

[31] W. Wang, D. Tran, M. Feiszli, What makes training multi-modal classification networks hard?, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2020, pp. 12695–12705.

[32] L. Zhu, H. Fan, Y. Luo, M. Xu, Y. Yang, Temporal cross-layer correlation mining for action recognition, IEEE Transactions on Multimedia (2021).

[33] X. Wang, L. Zhu, Y. Wu, Y. Yang, Symbiotic attention for egocentric action recognition with object-centric alignment, IEEE Transactions on Pattern Analysis and Machine Intelligence (2020).

[34] S. K. Yadav, A. Singh, A. Gupta, J. L. Raheja, Real-time yoga recognition using deep learning, Neural Computing and Applications 31 (12) (2019) 9349–9361.

[35] S. Jain, A. Rustagi, S. Saurav, R. Saini, S. Singh, Three-dimensional cnn-inspired deep learning architecture for yoga pose recognition in the real-world environment, Neural Computing and Applications (2020) 1–15.

[36] Z. Luo, W. Yang, Z. Q. Ding, L. Liu, I.-M. Chen, S. H. Yeo, K. V. Ling, H. B.-L. Duh, "left arm up!" interactive yoga training in virtual environment, in: 2011 IEEE Virtual Reality Conference, IEEE, 2011, pp. 261–262.

[37] Z. Wu, J. Zhang, K. Chen, C. Fu, Yoga posture recognition and quantitative evaluation with wearable sensors based on two-stage classifier and prior bayesian network, Sensors 19 (23) (2019) 5129.

[38] K. Rector, C. L. Bennett, J. A. Kientz, Eyes-free yoga: an exergame using depth cameras for blind & low vision exercise, in: Proceedings of the 15th International ACM SIGACCESS Conference on Computers and Accessibility, 2013, pp. 1–8.

[39] S. Patil, A. Pawar, A. Peshave, A. N. Ansari, A. Navada, Yoga tutor visualization and analysis using surf algorithm, in: 2011 IEEE Control and System Graduate Research Colloquium, IEEE, 2011, pp. 43–46.

[40] W. Wu, W. Yin, F. Guo, Learning and self-instruction expert system for yoga, in: 2010 2nd International Workshop on Intelligent Systems and Applications, IEEE, 2010, pp. 1–4.

[41] W. Yin, P. Tu, X. Chen, H. Zhang, Problem oriented analysis and decision expert system with large capacity knowledge-base, in: 2008 3rd International Conference on Intelligent System and Knowledge Engineering, Vol. 1, IEEE, 2008, pp. 32–37.

[42] C.-C. Hsieh, B.-S. Wu, C.-C. Lee, A distance computer vision assisted yoga learning system, Journal of Computers 6 (11) (2011) 2382–2388.

[43] H.-T. Chen, Y.-Z. He, C.-L. Chou, S.-Y. Lee, B.-S. P. Lin, J.-Y. Yu, Computer-assisted self-training system for sports exercise using kinects, in: 2013 IEEE International Conference on Multimedia and Expo Workshops (ICMEW), IEEE, 2013, pp. 1–4.

[44] E. W. Trejo, P. Yuan, Recognition of yoga poses through an interactive system with kinect device, in: 2018 2nd International Conference on Robotics and Automation Sciences (ICRAS), IEEE, 2018, pp. 1–5.

[45] P. Pullen, W. Seffens, Machine learning gesture analysis of yoga for exergame development, IET Cyber-Physical Systems: Theory & Applications 3 (2) (2018) 106–110.

[46] M. U. Islam, H. Mahmud, F. B. Ashraf, I. Hossain, M. K. Hasan, Yoga posture recognition by detecting human joint points in real time using microsoft kinect, in: 2017 IEEE Region 10 Humanitarian Technology Conference (R10-HTC), IEEE, 2017, pp. 668–673.

[47] H.-T. Chen, Y.-Z. He, C.-C. Hsu, Computer-assisted yoga training system, Multimedia Tools and Applications 77 (18) (2018) 23969–23991.

[48] M. Gochoo, T.-H. Tan, S.-C. Huang, T. Batjargal, J.-W. Hsieh, F. S. Alnajjar, Y.-F. Chen, Novel iot-based privacy-preserving yoga posture recognition system using low-resolution infrared sensors and deep learning, IEEE Internet of Things Journal 6 (4) (2019) 7192–7200.

[49] T. K. K. Maddala, P. Kishore, K. K. Eepuri, A. K. Dande, Yoganet: 3-d yoga asana recognition using joint angular displacement maps with convnets, IEEE Transactions on Multimedia 21 (10) (2019) 2492–2503.

[50] Y. Ke, R. Sukthankar, M. Hebert, Event detection in crowded videos, in: 2007 IEEE 11th International Conference on Computer Vision, IEEE, 2007, pp. 1–8.

[51] M. Müller, T. Röder, M. Clausen, B. Eberhardt, B. Krüger, A. Weber, Documentation mocap database hdm05 (2007).

[52] M. Verma, S. Kumawat, Y. Nakashima, S. Raman, Yoga-82: a new dataset for fine-grained classification of human poses, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops, 2020, pp. 1038–1039.

[53] K. He, X. Zhang, S. Ren, J. Sun, Deep residual learning for image recognition, in: Proceedings of the IEEE conference on computer vision and pattern recognition, 2016, pp. 770–778.

[54] G. Huang, Z. Liu, L. Van Der Maaten, K. Q. Weinberger, Densely connected convolutional networks, in: Proceedings of the IEEE conference on computer vision and pattern recognition, 2017, pp. 4700–4708.

[55] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, H. Adam, Mobilenets: Efficient convolutional neural networks for mobile vision applications, arXiv preprint arXiv:1704.04861 (2017).

[56] S. Xie, R. Girshick, P. Dollár, Z. Tu, K. He, Aggregated residual transformations for deep neural networks, in: Proceedings of the IEEE conference on computer vision and pattern recognition, 2017, pp. 1492–1500.

[57] S. K. Yadav, A. Luthra, K. Tiwari, H. M. Pandey, S. A. Akbar, Arfdnet: An efficient activity recognition & fall detection system using latent feature pooling, Knowledge-Based Systems (2021) 107948.

[58] J. Kittler, M. Hatef, R. P. Duin, J. Matas, On combining classifiers, IEEE transactions on pattern analysis and machine intelligence 20 (3) (1998) 226–239.

Declaration of interests

☒ The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

☐The authors declare the following financial interests/personal relationships which may be considered as potential competing interests: