Yoga Pose Monitoring System using Deep Learning

Debabrata Swain
Pandit Deendayal Energy Unversity
Santosh Satapathy
Pandit Deendayal Energy Unversity
Pramoda Patro
Koneru Lakshmaiah Education Foundation
Aditya Kumar Sahu (adityasahu.cse@gmail.com)
Vignan's Foundation for Science, Technology & Research

Research Article

Keywords: Yoga Pose, Deep Learning, Asanas, CNN, LSTM, Media pipe, Pose prediction

Posted Date: June 27th, 2022

DOI: https://doi.org/10.21203/rs.3.rs-1774107/v1

Abstract

Activity recognition is the process of continuously monitoring a person's activity and movement. Human posture recognition can be utilized to assemble a self-guidance practice framework that permits individuals to learn and rehearse yoga postures accurately without help from anyone else. Using Deep learning algorithms, an approach has been taken to accurately detect and recognize various yoga poses. The chosen dataset consists of a total of 85 videos with 6 yoga postures performed by 15 participants. Initially, keypoints of the user are extracted using the Mediapipe library. A combination of convolutional neural network (CNN) and long short-term memory (LSTM) has been used for Yoga pose recognition through real-time monitored videos as a deep learning model. CNN layer is used for extraction of features from the keypoints and it is followed by LSTM that understands the occurrence of sequence of frames for predictions to be made. Then, the poses are classified as correct or incorrect. If a correct pose is identified, the system will give the user feedback through text/speech.

1. Introduction

Recognizing human activity through computer vision has been an area of prime interest among researchers over several years, owing to its far-reaching applicability [1]. Human activity recognition has the potential to impact various domains such as robotics, human-computer interaction, gaming, video surveillance, biometric verification, chaos detection, sports monitoring, and health tracking among many others. While multiple opportunities for implementing recognition systems have been explored throughout the years, its application in detecting yoga posture is a relatively new and under-researched field that holds the immense capacity to promote public health and welfare through the promotion of this ancient Indian practice, besides serving as an important milestone for researchers in the advancement in this sphere of activity recognition [2].

Yoga, an ancient discipline that originated in India and was once local to the nation, is now becoming popular worldwide on account of its various physical, mental and spiritual benefits [3]. The increasing significance of yoga in medicine can be attributed to its extraordinary healing effects in a variety of conditions affecting the human body such as respiratory issues, cardiac ailments, musculoskeletal problems, and many others [4, 5]. However, a certain gap exists between the new generation and their understanding and awareness of the boons of yoga, leading to a host of health issues associated with today's rapid lifestyle that can be easily curbed by the adoption of yoga as a part of daily routine. One of the major factors leading to the misconceptions surrounding yoga, eventually contributing to people's unwillingness to incorporate it into their lives, is the unavailability of proper guidance. The growing innovation in technology, however, brings up the possibility of addressing this inaccessibility to right tutoring through a real-time self-learning aid, capable of detecting various yoga postures by activity recognition tools, allowing it to serve as a convenient means of instruction, necessary to help popularize the form in the desired manner.

One of the key objects of research in the scientific domains of deep learning and computer vision is the human skill to identify and interpret another person's actions. Activity recognition refers to the vast topic of study that focuses on the detection of the activities and goals of one or more individuals from a sequence of observations of the individuals' actions and surroundings. Human activity recognition is the subset concerning the modeling of the human body to recognize a person's unique motion and behavior through data received from either sensors or vision-based techniques. The movements to be identified may include common activities such as sitting, standing, conversing, or walking, or may encompass highly specific and complicated gestures such as the vivid yoga postures, or specific athletic stances [6]. Accurate classification of human poses still poses a challenge to the scientific community, stimulating further work in this area. Across the various characterization methods prevalent, a two-fold approach to the categorization problem remains uniform: firstly, the recognition part, and secondly, the localization problem, which indicates the description of the action to be detected, and the identification of the portion of the series of real-time movements which contains the object of interest respectively [7]. The entire process generally consists of three stages of representation: the low-level core technology, the mid-level behavior detection, and the high-level implementation [8]. In the initial step of the first phase, object segmentation is conducted on each frame of the video to distinguish and isolate the target object from its environment. The second level involves the retrieval of the attributes of separated objects such as color, shape, outlines, and movements and their subsequent representation into features, which can be broadly classified into four main types: frequency transform, local identifiers, space-time statistics, and body modeling data. The event detection and classification techniques employed to detect various human poses and movements predicated on the features derived from the previous step comprise the third stage of the operation. It is the difficulties faced at each of these sub-levels in the wake of a multitude of factors such as poor illumination, changes in video quality, partial visibility, closeness, background disturbance, angle involved, etc. that make human activity recognition a pretty arduous task [9].

Intensive studies continue to be conducted in this field, with an aim to improve upon the existing results achieved. Methods based on deep learning have gained popularity lately due to their use of representation learning algorithms that can instinctively extract suitable features from the input data supplied by sensors without the need for human interference, and showcase unique underlying patterns [10]. In this work, we propose a deep-learning-based model to recognize and correct yoga poses in real-time, with the motive to address the lack of availability of proper training and to make such education more affordable and fitting for people, by attempting to exploit the opportunity of self at-home training through an application that can accurately differentiate between the poses practiced by expert trainers from those of beginners, resulting in reliable recommendations to learners concerning necessary corrections. With the use of CNN and LSTM, an intelligent yoga pose monitoring system has been devised that takes in the videos of sample subjects as an input, converts them into a sequence of key points, which are the coordinates of key object points on a person, performs the detection task, and finally, provides essential feedback to the user based on cosine similarity in the form of text/speech [6]. The forthcoming sections will elucidate the precise methods employed, and the results achieved, before

highlighting the scope for further improvement. Activity recognition is the problem of predicting the movement of a person, based on sensor data or video sequences. It is a challenging computer vision problem that requires a lot of attention and improvement [1]. Activity recognition has many real-life applications like user interface design, robot learning, surveillance, etc. Activity recognition faces many challenges due to differences in video quality, partial appearance, brightness, proximity, background noise, angle, and viewpoint, etc. Human activity recognition is of concern with yoga pose recognition [2].

Yoga is a form of exercise that is extremely helpful in reducing mental stress and keeping a person physically and mentally fit. Moreover, it improves concentration, focus, calmness, and blood circulation [3]. Yoga originated in ancient India and has now been adopted by countries all over the world. Yoke refers to *yuj* in Sanskrit, a physical device that was used to join cattle. These devices were enormous, brute, and solid. They were yoking long ago for war horses. Yoga was both the method and device to calm down the horses so that you could make them concentrate and perform well in the war[4]. Research shows that people who practice yoga daily have a positive mindset, an improved sense of energy to live a full life, and a good control over-breathing. The classic, *Yoga Sutra*, written by Patanjali first described yoga philosophy and its practices [5].

In this work, we propose a system that can recognize and correct yoga poses in real-time using various deep learning techniques. By using such a system, a whole yoga class atmosphere can be created at the user's home where the yoga pose is detected and corrected by the system automatically. Finding a yoga class that is affordable and one that aligns with one's schedule is difficult. This paper tries to fill that gap of on-demand yoga classes with feedback on the yoga routine in real-time. The main idea is to create a deep learning model that can correctly classify a user's yoga pose by training it on a dataset of yoga videos. For this, the videos need to be in correct format by converting them into a sequence of key points. Key points are the coordinates of key object points on a person [6]. After detection, further feedback is given to the user based on cosine similarity in the form of text/speech.

2. Literature Survey

Significant work has been done in this field of real-time human monitoring for a variety of applications. The proposed research aims to take the existing literature into account and contribute towards an enhancement in the currently achievable results in the field of computer vision for human pose estimation. An attempt has been to devise a Deep Learning based system to efficiently detect yoga poses and practically work as a substitute for a trainer, by giving accurate feedback to the user. Kothari et al. [2] in a comparative study between machine learning and deep learning techniques for yoga pose estimation carried out an experimental work involving both the technologies and analyzed the performance of the framework by using support vector machine, convolutional neural network, and Convolutional Neural Network along with Long Short Term Memory. The research concluded that the performance of hybrid CNN-LSTM model resulted in the least number of misclassifications. Kadbhane et al. [11] captured video information using Microsoft Kinect. Kinect has a skeletal pursuit tool that may acknowledge twenty joints of a person's body. They used the chosen ten joint points for calculation. They engineered the reference

structure for each yoga position by gathering info of the joint points from human posture. They calculated the cosine similarity of vectors by finding angles between all the vectors connecting any two joint points. If the calculated deviation is more than a set threshold value, the pose is classified as incorrect. The paper did not mention the overall accuracy obtained. S. Haque et al. [12] proposed a model named ExNet, which is a multilayer convolutional neural network (CNN). They used an image dataset containing 2000 images of human exercise poses divided into five classes labeled as push up, pull up, cycling, swiss ball hamstring curl, walking. The model used adam optimizer and automatic learning rate reduction method. After 50 epochs, ExNET got 82.68% accuracy on classifying 2D human exercise pose from the dataset. The model needs better hyper-parameter tuning because it faces the problem of overfitting. Agarwal et al. [13] worked on a dataset that contained 10 different yoga poses with around 5500 images. They proposed a system where the skeleton of the user is first detected using an algorithm called the tf-pose estimation. They then used and compared six different machine learning models: Decision Tree, Random Forest, Logistic Regression, Naive Bayes, SVM and KNN. They obtained the optimum performance with the Random Forest classifier with an accuracy of 99.04%. Anilkumar et al. [14] proposed a method in which, when a pose is made by the user in front of his camera, mediapipe library does the geometric analysis based on the frames obtained from the camera. Geometric analysis produces output based on angles between various joints. The calculated angles are then compared with the accurate angles that are saved in the database for the specific yoga pose. If the difference between these two angles is more than the set threshold, feedback is given to the user, either in the form of text or through an audio device [14]. Luvizon et al. [15] had proposed a system capable of detecting poses associated with four kinds of normal human activity, and identifying human action in both the second and third dimension [40]. The research was conducted on data derived from four different sources i.e. Human 3.6M, NTU, Penn action and MPII using the multitask convolutional neural network. Another model to correct posture while exercising was proposed by Chen et al. [16] who used a dataset containing more than 100 videos of both correctly and incorrectly performed exercises. Training of the model was done using OpenPose, and 18 keypoints were defined by the authors to accurately detect the differences in poses. The task was conducted on 4 different exercises such as front raise, standing shoulder press, bicep curl, and shoulder shrug. Zou et al. [17] developed a smart fitness trainer on the concepts of pose detection and estimation using deep learning methods. A regional multi-person pose estimation framework has been used for the work. AlphaPose has been used for identifying the keypoints, which are the human joints within every frame. The figure obtained on connecting them is then compared with the ideal poses, resulting in suggestions to the user on how to correctly practice the exercises. Yadav et al. [18] with a dataset comprising six asanas, i.e. Tadasana, Shavasana, Bhujangasana, Trikonasana, Padmasana and Trikonasana, managed to accurately design a real-time deep learning model using OpenPose to extract the keypoints from the videos supplied. While system was capable of an accuracy of 99.04% with the input taken framewise, detection on the basis of a survey of the preset 45 edges yielded a performance measure of 99.38%. The work was further tested on twelve individuals and proved to be highly effective with an accuracy of 98.92%. Thoutam et al. [19] too applied a deep learning technique for training the model to recognize yoga poses. After extracting features through Keras multi-purpose pose estimation, classification into one of the six poses: corpse pose, tree pose, mountain pose, triangle pose,

lotus pose and cobra pose, according to the angles between twelve keypoints was done using Multilayer Perceptron, resulting in an accuracy of 99.58%.

3. Dataset

The dataset comprises a total of 85 videos of HD 1080p quality, recorded by 15 different individuals. It contains videos of 6 different yoga poses: Trikonasana, Padmasana, Vrikshasana, Bhujangasana, Tadasana and Shavasana. The videos were recorded using an NVIDIA TITAN X GPU and Intel Xeon processor with 32 GB RAM with a frame rate of 30 frames per second. The users tried to perform different asanas with many variations.

S. no.	Asana name	No. of persons	No. of videos
1	Trikonasana	13	13
2	Padmasana	14	14
3	Vrikshasana	12	12
4	Bhujangasana	15	16
5	Tadasana	15	15
6	Shavasana	15	15

4. Proposed System

The proposed system takes video sequence frames as input in real time. The output would be the predicted yoga pose along with possible feedback for angle and pose correction. The system consists of three main phases: Keypoints extraction, Pose prediction and Pose correction. Keypoints extraction phase does the job of detecting and extracting location of important keypoints based on the user's position[20]. The pose prediction phase defines the model architecture and classifies if the pose is correct or not. The final phase is pose correction where the user is further given feedback for correction of pose and is also shown the similarity percentage to the actual pose. Figure 1 shows the proposed system architecture with all the above phases.

4.1. Keypoints extraction

The first step is to extract keypoints from all frames of the video and store it in JSON format. Keypoints consists of different points in a person that are vital in the formation of a yoga pose. Examples include shoulders, elbows, wrist, knees, etc. We used the library 'MediaPipe' for keypoints extraction. Mediapipe is a cross-platform library developed by Google that provides amazing ready-to-use ML solutions for computer vision tasks. It is a highly optimized pre-trained CNN model used for high-fidelity body pose tracking, inferring 33 3D landmarks and background segmentation masks on the whole body from RGB

video frames. Mediapipe generates 3 coordinates (X, Y and Z), where Z indicates the depth of a 2D coordinate [21]. Figure 2 shows all the 33 keypoints provided by MediaPipe. Figure 3 shows the result after extracting the keypoints using the MediaPipe library. After converting the videos to JSON format, we split it to train, validation and test dataset. The split ratio used is 60:20:20. Each test case consists of a sequence of 45 frames with an overlap size of 36 frames, containing the coordinates of all the 33 keypoints. The input shape of a single test case can be given as (45, 33, 2).

4.2. Pose prediction

The second step is to create a deep learning model that correctly classifies any real-time video to one of the six poses given in the dataset. Here a hybrid model is used which is a mixture of CNN and LSTM. In this work, CNN is used for feature extraction [22]. CNN is a multilayered ANN (artificial neural network) that is specifically created to work on images and is used for tasks like object recognition and image classification [23]. LSTM is useful for understanding the sequence of frames occurring in a particular yoga pose. LSTM is a type of RNN that is equipped for learning and recollecting extremely long-term dependencies over long successions of input data [24]. We used a TimeDistributed layer along with CNN which is particularly useful when working with video frames or time series data [25]. Finally, a Softmax layer which uses a normalised exponentiation function. Here, it is used to find the probability of each yoga pose. The pose with the highest probability is predicted as the output.

The first layer used is a CNN layer with 16 filters and a window size of 3. The activation function used in this layer is ReLU. The ReLU activation function is a piecewise linear function that gives an output 0 when the input provided is less than 0, else it gives output as the given input [26, 27]. Eq. (1) shows the activation function of ReLU.

$$\mathrm{ReLU}(\mathrm{x}) = \max(0, \mathrm{x})$$

1

where x.∈ R.

The second layer used is a BatchNormalization layer that solves a problem of internal covariate shift and makes the flow of data through different layers easier [28]. The next layer used is a dropout layer [29], which is a regularization technique for preventing overfitting at the rate of 0.5. The output obtained from this layer is then passed to a Flatten layer that converts the data to a one-dimensional array. The next layer used is an LSTM layer. The size of the layer is 20 units with forget bias set to true in order to return the output of every node. The output of LSTM generally generated through a series of Gated operations like: Forget gate, Input gate and output gate. The mathematical equations for an LSTM are shown in the equations 2, 3, 4, 5, 6, and 7. The state information of current cell and previous cell are stored in C. The sigmoid function output in the forget gate indicates about which information to retain or forget. The output mainly depends upon the previous state output vector $h_{r-1}$and current state input vector x_r. If there is difference then σ function does not allow the information to retain. The forget bias helps to give a shift to the output either in the direction of 1 (retain) or 0 (forget). The model outperforms with forget bias

value as 0.5. In the input gate the σ function decides which value to update and tanh function adds new values for the state. In the output layer the σ function determines the value to be selected as output h_r.

$l_r = \sigma (W_l . [h_{r-1}, x_r] + b_l)$ (2)

$m_r = \sigma(W_m . \left[h_{r-1}, x_r\right] + b_m)$ (3)

$$\check{C}_r = \tanh(W_c . \left[h_{r-1}, x_r\right] + b_c$$

4

$$C_r = l_r * C_{r-1} + i_r * \check{C}_r$$

5

$$o_r = \sigma(W_o \left[h_{r-1}, x_r\right] + b_o$$

6

$$h_r = o_r * \tanh\left(C_r\right)$$

7

Where l_r, m_r, o_r, h_r = output of forget gate, input gate, output gate and current state

W_f , W_m, W_o=weight of forget gate, input gate and output gate

x_r = current state input

b_l , b_m, b_o= bias of forget gate, input gate and output gate

m_r = output of input gate

$\check{C}_r$, C_r= cell state of current and previous

The final layer used is a Dense layer that uses Softmax [30] as the activation function which assigns probabilities of different poses based on the current given input.

The mathematical equation of Softmax activation function is shown in Eq. (8).

$$\sigma\left(\underset{\_}{z}\right)_i = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}}$$

8

where, σ= *softmax*, z = *input vector*, e^{z_i}= *standard exponential function for input vector, K = number of classes in the multi-class classifier*, e^{z_j}= standard exponential function for output vector.

The output obtained after this layer is then polled on 45 frames to get the final prediction. The optimizer used is an Adam optimizer [31] with a learning rate of 0.0001. The Adam optimizer helps the model to fast converge by the addition of momentum term and scaling term as shown in Eq. (9). Adam optimizer combines the idea of momentum and RMSprop optimizer and helps to avoid the exponential decay of learning rate issue.

$$\theta_{\mathrm{new}} = \theta_{\mathrm{old}} - \eta\, \hat{m} \oslash \sqrt{\hat{s} + \epsilon} \quad (9)$$

Where, θ_{new},θ_{old}= New and old weight value, η = learning rate, $\hat{m}$= momentum term, $\hat{s}$= scaling term

Є= smoothing term to avoid zero division error, ⊘= element wise division operation

The loss function used is categorical cross-entropy which is very popular for multi-class classification tasks [32]. Eq. (10) depicts the mathematical equation used in categorical cross-entropy loss function.

$$\mathrm{E_{CC}} = -\frac{1}{\mathrm{N}} \sum_{\mathrm{i}=1}^{\mathrm{N}} \sum_{\mathrm{c}=1}^{\mathrm{C}} \left(\mathrm{p_{ic}} \log\left(\mathrm{y_{ic}}\right)\right)$$

10

Where, E_{CC}= categorical cross-entropy, N = no. of pairs available in training set, C = no. of categories, p_{ic} = a binary indicator function that detects whether the ith training pattern belongs to cth category, y_{ic}= a predicted probability distribution for ith observation belonging to class c.

The metric used to gauge the performance of the model is Accurac*y* [33]. The model was trained for a total of 50 epochs. Initially, the growth was exponential, it became steady after a few epochs. After each epoch, it is checked whether the accuracy has improved and is better than the best accuracy obtained. If it is better than the best accuracy, the best accuracy is replaced by the current accuracy. All the parameters used in the model have been tuned perfectly using hyperparameter tuning in order to obtain the most optimized results [34, 35].

4.3. Pose correction

After the predicted pose is classified to be correct, with respect to the chosen pose, the user is given appropriate feedback and similarity percentage (using cosine similarity) is calculated to be shown to the user. For all the six yoga poses present in the dataset, critical angles have been identified and rules have been formulated for each pose. For each rule, a threshold is set, which is the maximum deviation allowed for the user from the standard pose. If the user exceeds this threshold value, feedback is given

accordingly in the form of text and speech. The angle between two keypoints can be found out by calculating the tangent inverse of the slope with positive X-axis. Eq. (11) shows the formula for finding the angle, given the two coordinates of the keypoints.

$$\Theta = \tan^{-1}\left(\frac{y_2 - y_1}{x_1 - x_2}\right)$$

11

Where, (x_1, y_1) and $(x_2, y_2)$are coordinates of 2 keypoints.

Feedback is initially obtained in the form of text which is then converted into speech using *Pyttsx3* library[36]. It is a text-to-speech converter and works offline as well.

Cosine similarity is also shown to the user which is a measure that compares two vectors by calculating the cosine of angles between them [37]. The mathematical formula of cosine similarity is given in Eq. (12).

$$\cos\theta = \frac{A \cdot B}{||A|| \cdot ||B||}$$

12

Where, *A* and *B* are two vectors in a multidimensional space.

In this work, cosine similarity is calculated between keypoints of the user's pose and the standard pose. This way, it shows the degree of closeness to the actual pose. Since the distance of different users can vary based on their position from the camera, all keypoints are first normalized to bring them on a similar scale.

5. Results

Prediction of yoga pose for a sequence of 45 frames is done using polling where mode of all yoga pose predictions is considered as the final prediction. After running the model for 50 epochs, the training accuracy obtained is about 99.49% whereas validation accuracy is about 99.70%. Figure 4 shows the graphical representation of model accuracy and model loss.

5.1. Confusion matrix

A confusion matrix is an important performance measure that is used for calculation of other analytics like recall, precision, specificity, sensitivity, etc. It is also used for classification problems to summarize prediction results [38]. It consists of four main values: true positive (TP), true negative (TN), false positive (FP) and false negative (FN) [39]. The test accuracy obtained is about 99.53%. Figure 5 shows the

confusion matrix between the predicted yoga asanas and the actual asanas. Darker the blue colour is on the diagonal, higher is the correlation between the predicted and true values. Since most of the diagonal values are close to 1, it indicates high correlation and corresponds to a good model.

Precision is a metric used to evaluate the performance of a model. It is defined as the fraction of samples classified correctly as positive from all the samples that are actually positive [40]. Its mathematical equation is given in Eq. (12). Precision obtained from the model is about 0.9866.

$$Precision = \frac{TP}{TP + FP}$$

12

Recall is yet another metric used to evaluate the performance of a model. It is defined as the fraction of samples classified correctly as positive from all the samples that are predicted positive [41]. Its mathematical equation is given in Eq. (13). Recall obtained from the model is about 0.9869.

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}}$$

13

5.2. ROC curve

ROC curve is a graphical representation that shows the performance of a model by plotting false positive rate against true positive rate[42]. A model is said to be ideal when area under the ROC curve tends to 1. Figure 6 shows the ROC curve of all the asanas. Area under the curve obtained is 0.99, which is exceptionally high.

5.3. Comparative analysis of activation functions

Activation functions are essential to any deep learning model in helping the artificially designed framework to autonomously learn complex patterns from the data supplied to it. Also known as the transfer function, it is generally located at the end, with an assigned task of carrying out a number of non-linear operations on the input, before finalizing it as the output and sending it to the successive layer of neurons. Among the various activation functions involved in this study, relu proved to be the most accurate followed by sigmoid, softmax and tanh, as can be seen in Fig. 7.

5.4. Comparative analysis of optimizers

Optimizers in a neural network play an important role in reducing the model loss and improving accuracy. These are the algorithms that are responsible for modifying certain aspects of the network, such as weights and learning rate, in order to provide optimal results. Figure 8 depicts the performance of varied optimizers deployed in the experiment.

5.5. Impact of number of CNN layers on model performance

The number of hidden layers in a CNN model can significantly affect its accuracy of classification. While an increased number of layers is generally attributed to an increased overall accuracy, it is not a universal observation and often depends on the complexity of the task involved. In the absence of a sufficiently large training set, adding additional layers can give rise to a highly large and complicated network that is prone to overfitting, and thereby reduces the accuracy obtained on the test data. Adding extra layers can also contribute to a greater execution time, as demonstrated through the line graph in Fig. 9. For this particular model, the best results were obtained on the use of a single convolutional layer.

5.6. Impact of number of CNN filters on model performance

A strong correlation can be seen between the number of CNN filters used, and the corresponding accuracy achieved in Fig. 10. Every layer of filter is responsible for extracting a certain set of features from the input data. As patterns continue to get more complicated, a growing number of filters are required for accurately extracting a number of different combinations of prominent features from the data received.

5.7. Impact of Learning Rate on model performance

Learning rate refers to the extent of adjustment allowed in the model in response to the error observed every time the model weights are altered. Deciding upon an optimal learning rate can be tricky, since neither of the extremes is desirable. While a lower learning rate may help build a more accurate model, it can result in an extremely slow training process. Similarly, too high a value can significantly reduce learning time, but at the cost of accuracy. Figure 11 shows the dependence of accuracy on learning rate.

5.8. Impact of Dropout Rate on model performance

At every stage of the training process, a certain number of neurons are dropped out from the layer. This fraction of neurons whose values are to be nulled, gives rise to the dropout rate. Dropout is necessary to prevent overfitting in neural networks, and is also used as a means of dealing with complex co-adaptions caused by neurons with similar connection weights during training. Figure 12 represents the ideal dropout rate obtained for this model.

5.9. Overall result

After combination of all the three phases, a reliable frontend was created for monitoring the user in real-time for pose prediction and correction. Figure 13 shows the overall view of the frontend and Fig. 14 shows the result after pose prediction and correction.

6. Conclusion

In this paper, we proposed an efficient system for real-time yoga monitoring. It first finds keypoints of the user using the MediaPipe library, where important coordinates are recorded and stored in JSON format. We then create a sequence of 45 frames from real-time, and pass it to the model. The model, using a mixture of CNN and LSTM, first finds the useful features using CNN and then observes the occurrence of sequence of frames using LSTM. The Softmax layer finds the probability of each yoga asana for the current frame sequence and outputs the highest probability asana. The output of each frame is then polled on 45 frames where mode is calculated and shown to the user as output. The system gives an excellent accuracy of 99.53% on the test dataset. If the pose is classified as correct, further feedback is given to the user based on a set threshold. The threshold is set in such a way that it does not make the system too strict and also ensures accurate pose and angles made by the user at the same time. Finally, the similarity percentage is shown to the user when compared with the standard pose.

Declarations

Competing interests: We declare we have no competing interests.

Funding: We declare this work is an independent work and no financial assistance has been received for the work.

Authors' contributions: The first and second authors have conceived the idea and executed it. Other authors have performed the analysis and language correction. Finally, all authors agreed on the paper for submission.

References

1. Vrigkas, M., Nikou, C., & Kakadiaris, I. A. (2015). A Review of Human Activity Recognition Methods. *Frontiers in Robotics and AI, 2*. doi:10.3389/frobt.2015.00028
2. Kothari, Shruti, "Yoga Pose Classification Using Deep Learning" (2020). Master's Projects. 932.DOI: https://doi.org/10.31979/etd.rkgu-pc9k
3. McClafferty, Hillary (2017). Medical Yoga Therapy. Children (Basel). 4(2). doi: 10.3390/children4020012
4. Newcombe, S. (2009). The Development of Modern Yoga: A Survey of the Field. Religion Compass, 3(6), 986–1002. doi:10.1111/j.1749-8171.2009.00171.x
5. Woodyard, C. (2011). Exploring the therapeutic effects of yoga and its ability to increase quality of life. International Journal of Yoga, *4*(2), 49. doi:10.4103/0973-6131.85485
6. Brownlee, Jason (2018). Deep learning models for human activity recognition. Machine Learning Mastery.
7. Vrigkas, M., Nikou, C., Kakadiaris, I. (2015). A review of human activity recognition methods. https://doi.org/10.3389/frobt.2015.00028

8. Shian-Ru Ke, Hoang Le Uyen Thuc, Yong-Jin Lee, Jenq-Neng Hwang, Jang-Hee Yoo, Kyoung-Ho Choi (2013). A review on video-based human activity recognition. Computers. 2. 88–131. doi: 10.3390/computers2020088
9. Alzahrani, M., Kammoun, S. (2016). Human Activity Recognition: Challenges and Process Stages. International Journal of Innovative Research in Computer and Communication Engineering. 4(5). 1111–1118.
10. Gupta, Saurabh (2021). Deep learning based human activity recognition (HAR) using wearable sensor data. International Journal of Information Management Data Insights. 1(2). doi: https://doi.org/10.1016/j.jjimei.2021.100046
11. Kadbhane,S., Datir,K., Jagdale,T., Dhongade,S., & Jagtap,G.(2021). Yoga Posture Recognition. International Journal of Advanced Research in Computer and Communication Engineering, 10(1), 143–147. doi:10.17148/IJARCCE.2021.10128
12. Haque, S., Rabby, A. S., Laboni, M. A., Neehal, N., & Hossain, S. A. (2019). ExNET: Deep Neural Network for Exercise Pose Detection. Communications in Computer and Information Science Recent Trends in Image Processing and Pattern Recognition, 186–193. doi:10.1007/978-981-13-9181-1_17
13. Agrawal, Y., Shah, Y., & Sharma, A. (2020). Implementation of machine learning technique for identification of yoga poses. In 2020 IEEE 9th International Conference on Communication Systems and Network Technologies (CSNT), 40–43. doi:10.1109/CSNT48778.2020.9115758
14. Anilkumar, A., K.t., A., Sajan, S., & K.a., S. (2021). Pose Estimated Yoga Monitoring System. *SSRN Electronic Journal*. doi:10.2139/ssrn.3882498
15. Luvizon, Diogo C., David Picard, and Hedi Tabia. "2d/3d pose estimation and action recognition using multitask deep learning." Proceedings of the IEEE conference on computer vision and pattern recognition. 2018.
16. Chen, Steven & Yang, Richard. (2018). Pose Trainer: Correcting Exercise Posture using Pose Estimation.
17. Zou, J., Li, B., Wang, L., Li, Y., Li, X., Lei, R., & Sun, S. (2019). Intelligent Fitness Trainer System Based on Human Pose Estimation. Signal and Information Processing, Networking and Computers, 593–599. doi:10.1007/978-981-13-7123-3_69
18. Yadav, Santosh & Singh, Amitojdeep & Gupta, Abhishek & Raheja, Jagdish. (2019). Real-time Yoga recognition using deep learning. Neural Computing and Applications. 31. https://link.springer.com/article/10.1007/s00521-019. 10.1007/s00521-019-04232-7.
19. Vivek Anand Thoutam, Anugrah Srivastava, Tapas Badal, Vipul Kumar Mishra, G. R. Sinha, Aditi Sakalle, Harshit Bhardwaj, Manish Raj, "Yoga Pose Estimation and Feedback Generation Using Deep Learning", Computational Intelligence and Neuroscience, vol. 2022, Article ID 4311350, 12 pages, 2022. https://doi.org/10.1155/2022/4311350
20. Josyula, R., & Ostadabbas, S. (2021). A Review on Human Pose Estimation. arXiv preprint arXiv:2110.06877.

21. Zhang, F., Bazarevsky, V., Vakunov, A., Tkachenka, A., Sung, G., Chang, C. L., & Grundmann, M. (2020). Mediapipe hands: On-device real-time hand tracking. arXiv preprint arXiv:2006.10214.
22. Jogin, M., Mohana, Madhulika, M. S., Divya, G. D., Meghana, R. K., & Apoorva, S. (2018). Feature Extraction using Convolution Neural Networks (CNN) and Deep Learning. 2018 3rd IEEE International Conference on Recent Trends in Electronics, Information & Communication Technology (RTEICT). doi:10.1109/rteict42901.2018.9012507
23. Al-Saffar, A. A., Tao, H., & Talab, M. A. (2017). Review of deep convolution neural network in image classification. *2017 International Conference on Radar, Antenna, Microwave, Electronics, and Telecommunications (ICRAMET)*. doi:10.1109/icramet.2017.8253139
24. Shiranthika, C., Premakumara, N., Chiu, H., Samani, H., Shyalika, C., & Yang, C. (2020). Human Activity Recognition Using CNN & LSTM. *2020 5th International Conference on Information Technology Research (ICITR)*. doi:10.1109/icitr51448.2020.9310792.
25. Ullah, A., Ahmad, J., Muhammad, K., Sajjad, M., & Baik, S. W. (2018). Action Recognition in Video Sequences using Deep Bi-Directional LSTM With CNN Features. IEEE Access, 6, 1155–1166.doi:10.1109/access.2017.2778011
26. Agarap, A. F. (2018). Deep learning using rectified linear units (relu). arXiv preprint arXiv:1803.08375.
27. Swain, Debabrata, Santosh Pani, and Debabala Swain. "Diagnosis of coronary artery disease using 1-D convolutional neural network." Int. J. Recent Technol. Eng.(IJRTE) 8.2 (2019).
28. Thakkar, V., Tewary, S., & Chakraborty, C. (2018). Batch Normalization in Convolutional Neural Networks — A comparative study with CIFAR-10 data. 2018 Fifth International Conference on Emerging Applications of Information Technology (EAIT). doi:10.1109/eait.2018.8470438
29. Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. (2014). Dropout: a simple way to prevent neural networks from overfitting. The journal of machine learning research, 15(1), 1929–1958.
30. Szandała, T. (2020). Review and Comparison of Commonly Used Activation Functions for Deep Neural Networks. Bio-inspired Neurocomputing Studies in Computational Intelligence, 203–224. doi:10.1007/978-981-15-5495-7_11.
31. Kingma, D. P., & Ba, J. (2014). Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980.
32. Rusiecki, A. (2019). Trimmed categorical cross-entropy for deep learning with label noise. Electronics Letters, *55*(6), 319–320. doi:10.1049/el.2018.7980
33. Fatourechi, M., Ward, R. K., Mason, S. G., Huggins, J., Schlögl, A., & Birch, G. E. (2008). Comparison of Evaluation Metrics in Classification Applications with Imbalanced Datasets. *2008 Seventh International Conference on Machine Learning and Applications*. doi:10.1109/icmla.2008.34
34. Diaz, G. I., Fokoue-Nkoutche, A., Nannicini, G., & Samulowitz, H. (2017). An effective algorithm for hyperparameter optimization of neural networks. IBM Journal of Research and Development, *61*(4/5). doi:10.1147/jrd.2017.2709578

35. Swain, D., Pani, S. K., & Swain, D. (2019). An efficient system for the prediction of coronary artery disease using dense neural network with hyper parameter tuning. International Journal of Innovative Technology and Exploring Engineering (IJITEE), *8*, 6S.
36. S., S., S., S., Srivatsa, P. N., A., U., & B., S. (2021). Developing a Graphical User Interface for an Artificial Intelligence-Based Voice Assistant. International Journal of Organizational and Collective Intelligence, *11*(3), 49–67. doi:10.4018/ijoci.2021070104
37. Lahitani, A. R., Permanasari, A. E., & Setiawan, N. A. (2016). Cosine similarity to determine similarity measure: Study case in online essay assessment. 2016 4th International Conference on Cyber and IT Service Management. doi:10.1109/citsm.2016.7577578
38. Hasnain, M., Pasha, M. F., Ghani, I., Imran, M., Alzahrani, M. Y., & Budiarto, R. (2020). Evaluating Trust Prediction and Confusion Matrix Measures for Web Services Ranking. IEEE Access, *8*, 90847–90861. doi:10.1109/access.2020.2994222
39. Swain, D., Ballal, P., Dolase, V., Dash, B., & Santhappan, J. (2020). An Efficient Heart Disease Prediction System Using Machine Learning. In *Machine Learning and Information Processing* (pp. 39–50). Springer, Singapore.
40. Buckland, M., & Gey, F. (1994). The relationship between recall and precision. Journal of the American society for information science, 45(1), 12–19.
41. Bradley, A. P. (1997). The use of the area under the ROC curve in the evaluation of machine learning algorithms. Pattern Recognition, *30*(7), 1145–1159. doi:10.1016/s0031-3203(96)00142-2
42. D. Swain, S. K. Pani and D. Swain, "A Metaphoric Investigation on Prediction of Heart Disease using Machine Learning," 2018 International Conference on Advanced Computation and Telecommunication (ICACAT), 2018, pp. 1–6, doi: 10.1109/ICACAT.2018.8933603.

Figures

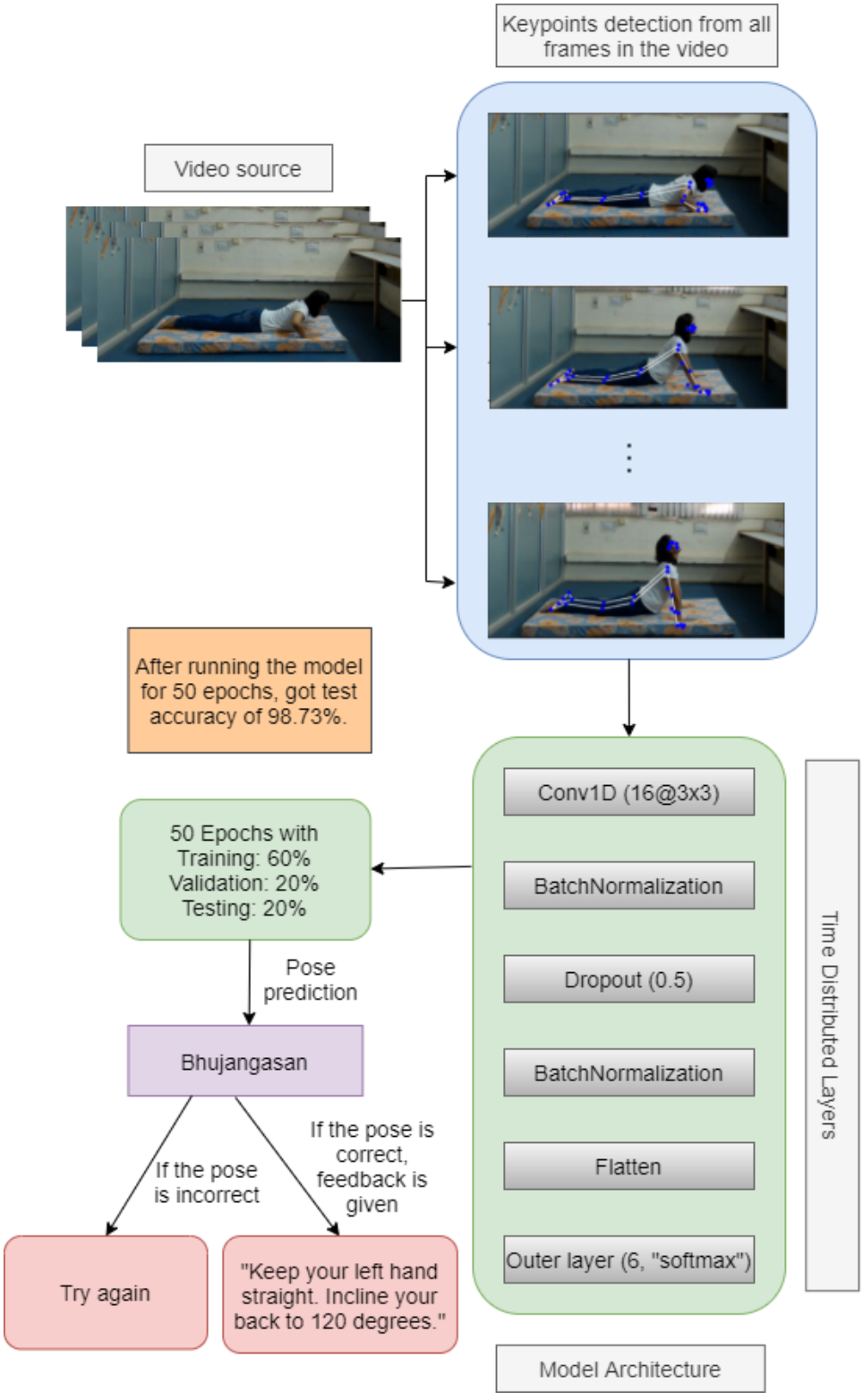
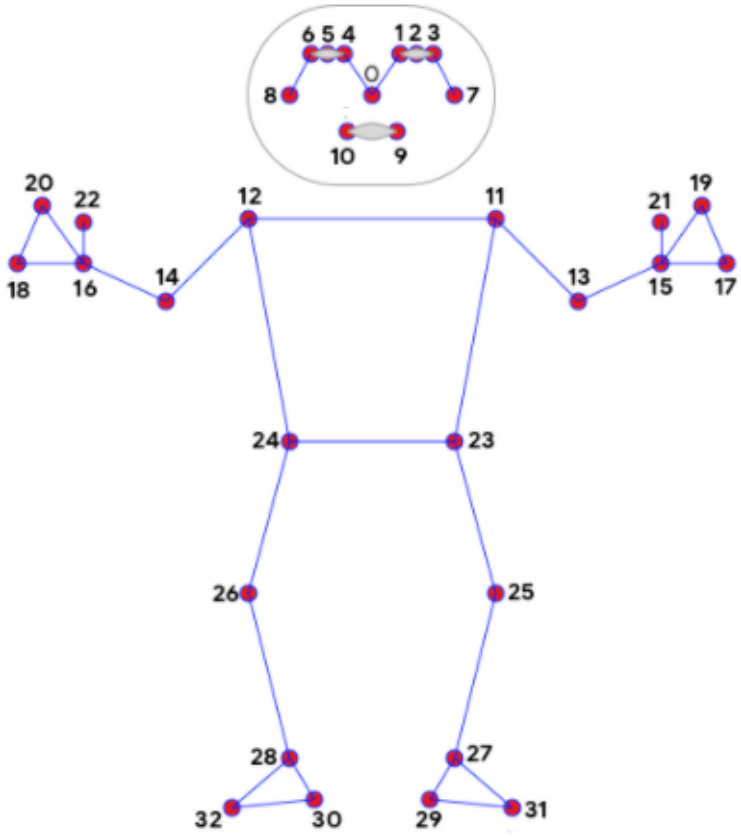


Figure 1

Flowchart of Proposed System



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

Figure 2

MediaPipe keypoints

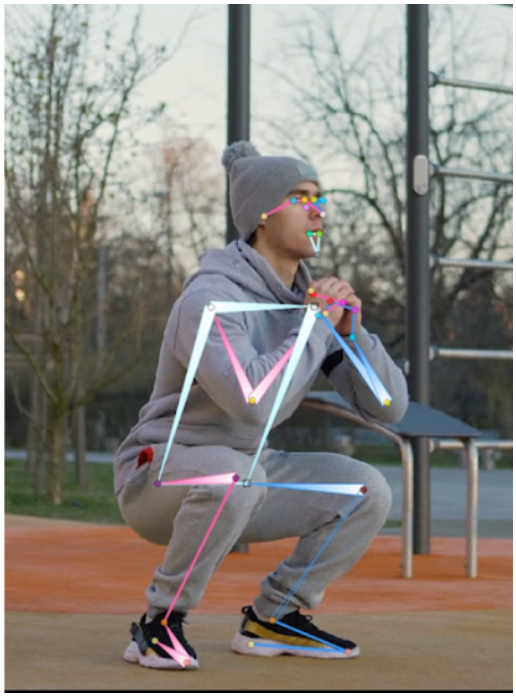


Figure 3

MediaPipe keypoints on actual humans

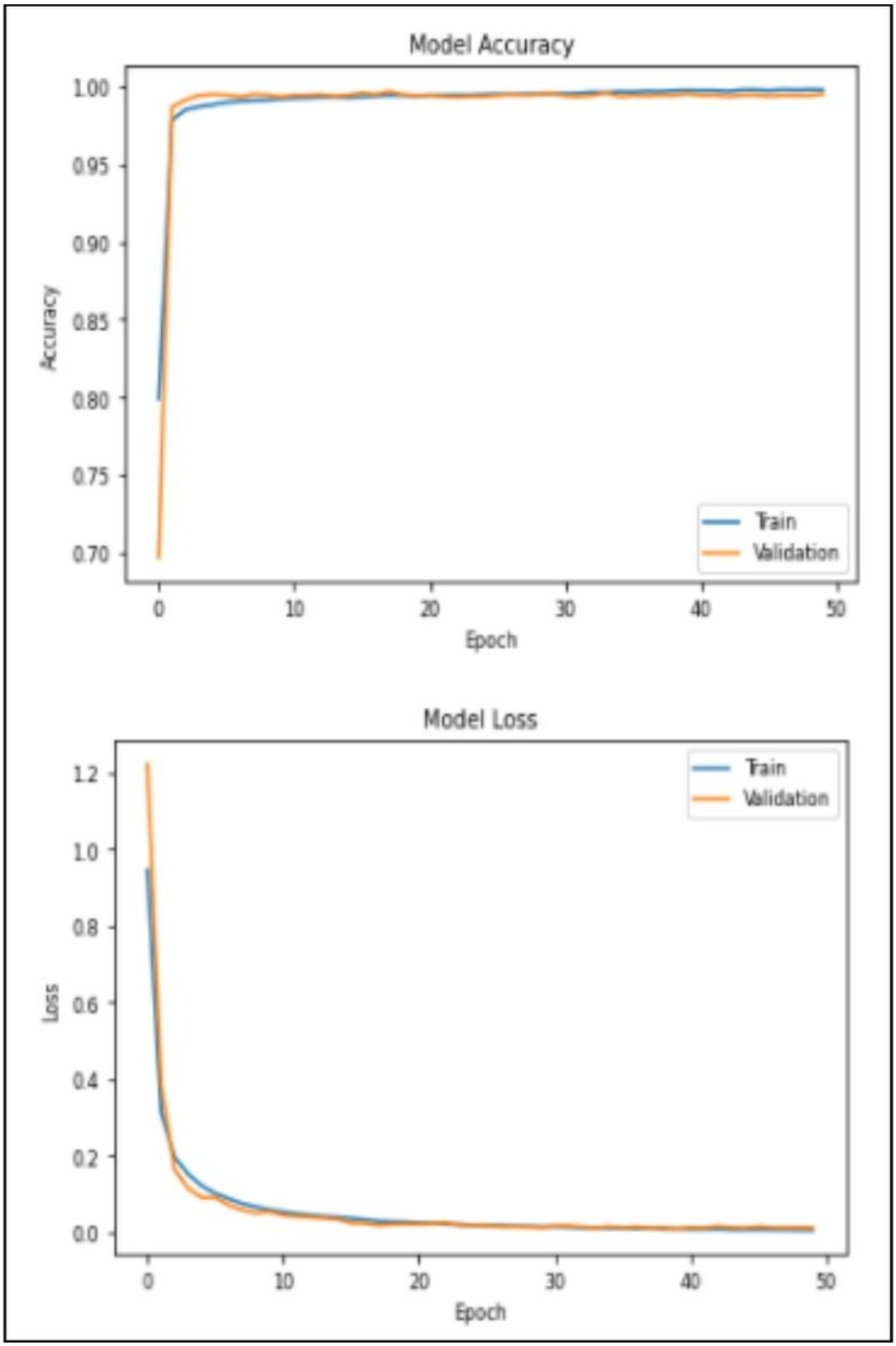


Figure 4

Model accuracy and model loss

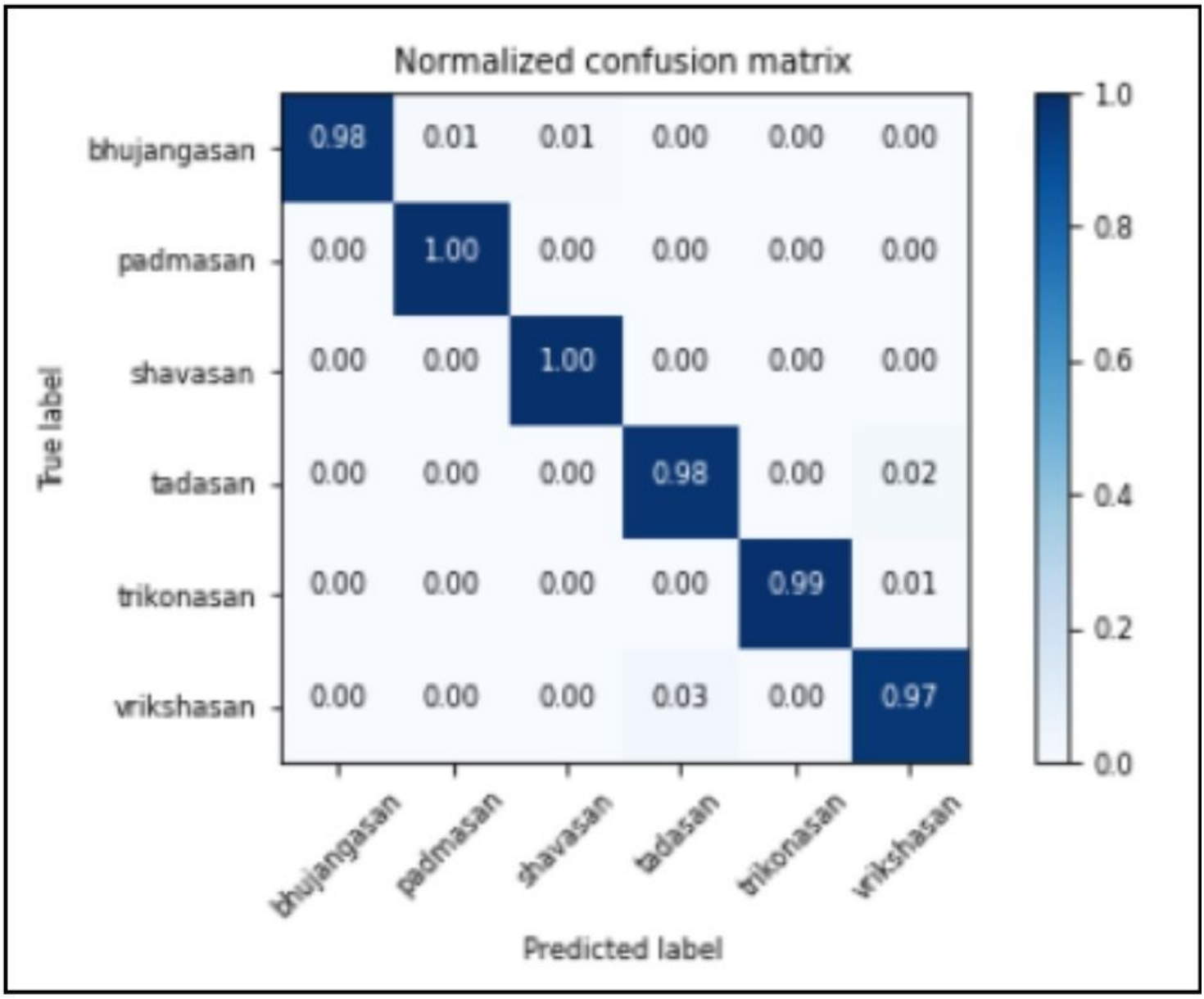


Figure 5

Confusion Matrix

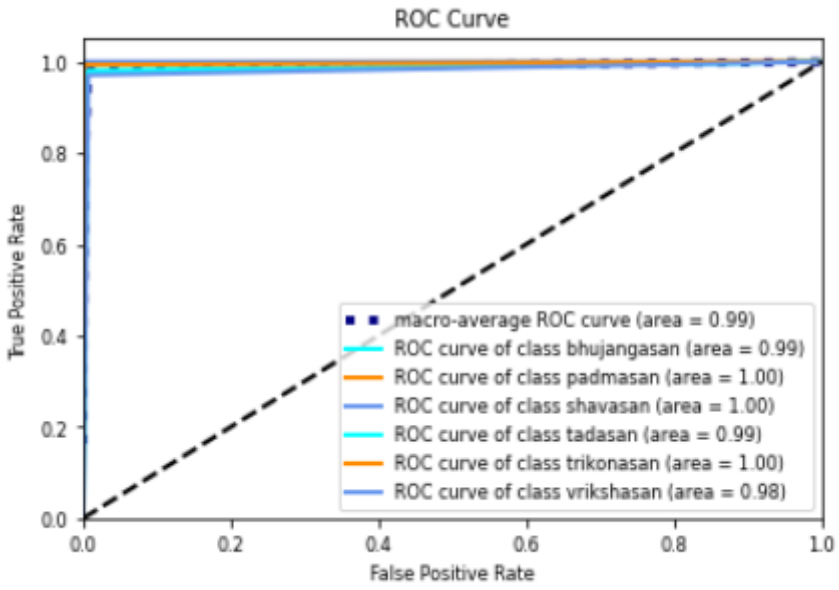


Figure 6

ROC curve

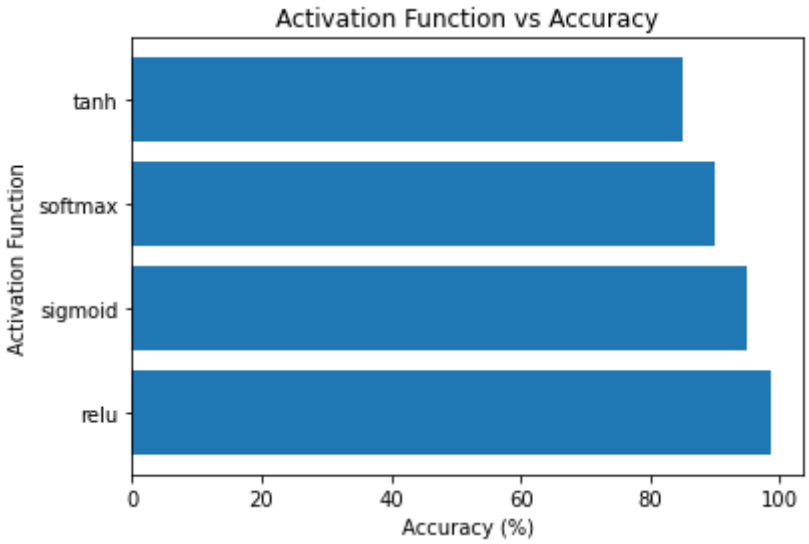


Figure 7

Activation Functions used and their corresponding accuracies

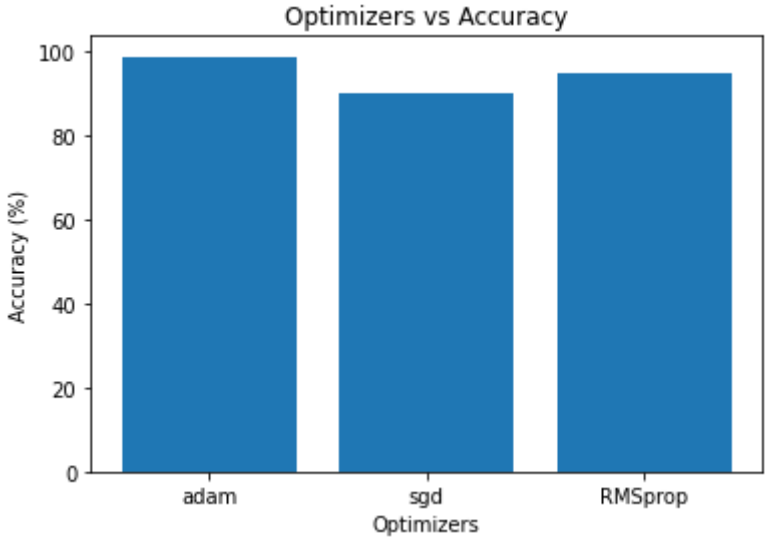


Figure 8

Optimizers used and their corresponding accuracies

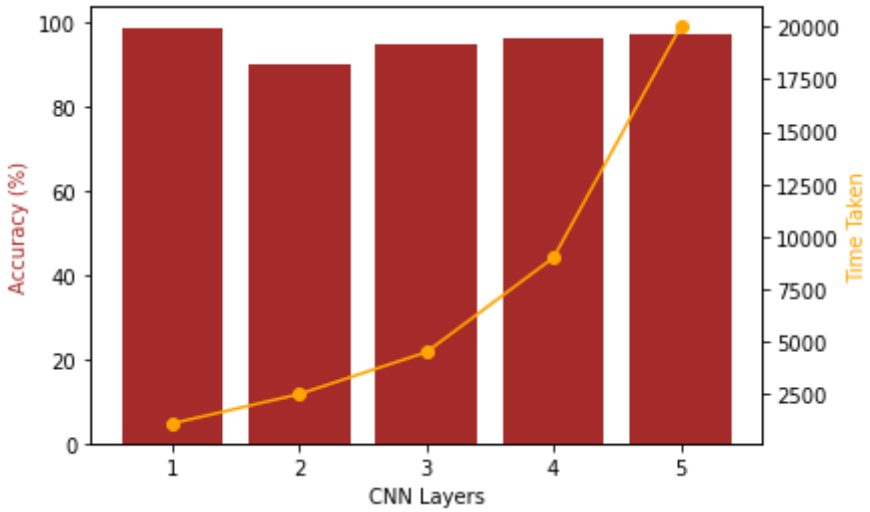


Figure 9

Accuracies associated with varied number of CNN layers

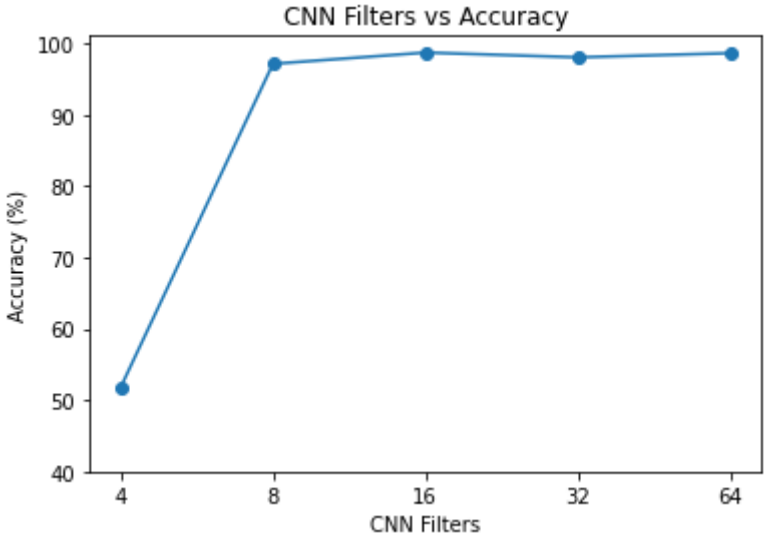


Figure 10

Accuracies associated with varied number of CNN filters

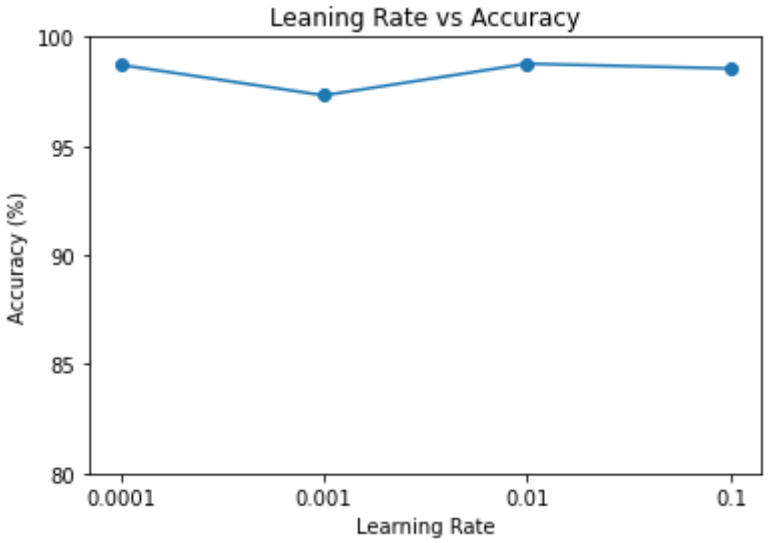


Figure 11

Impact of learning rate on accuracy

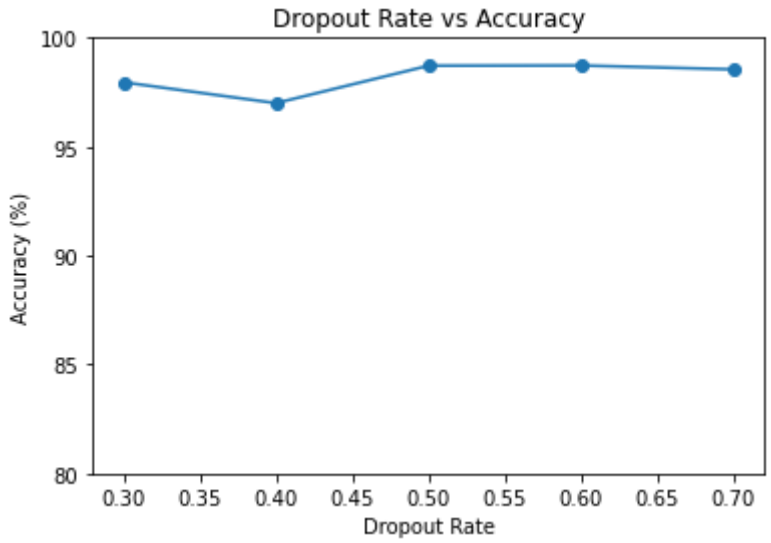


Figure 12

Impact of Dropout Rate on Accuracy

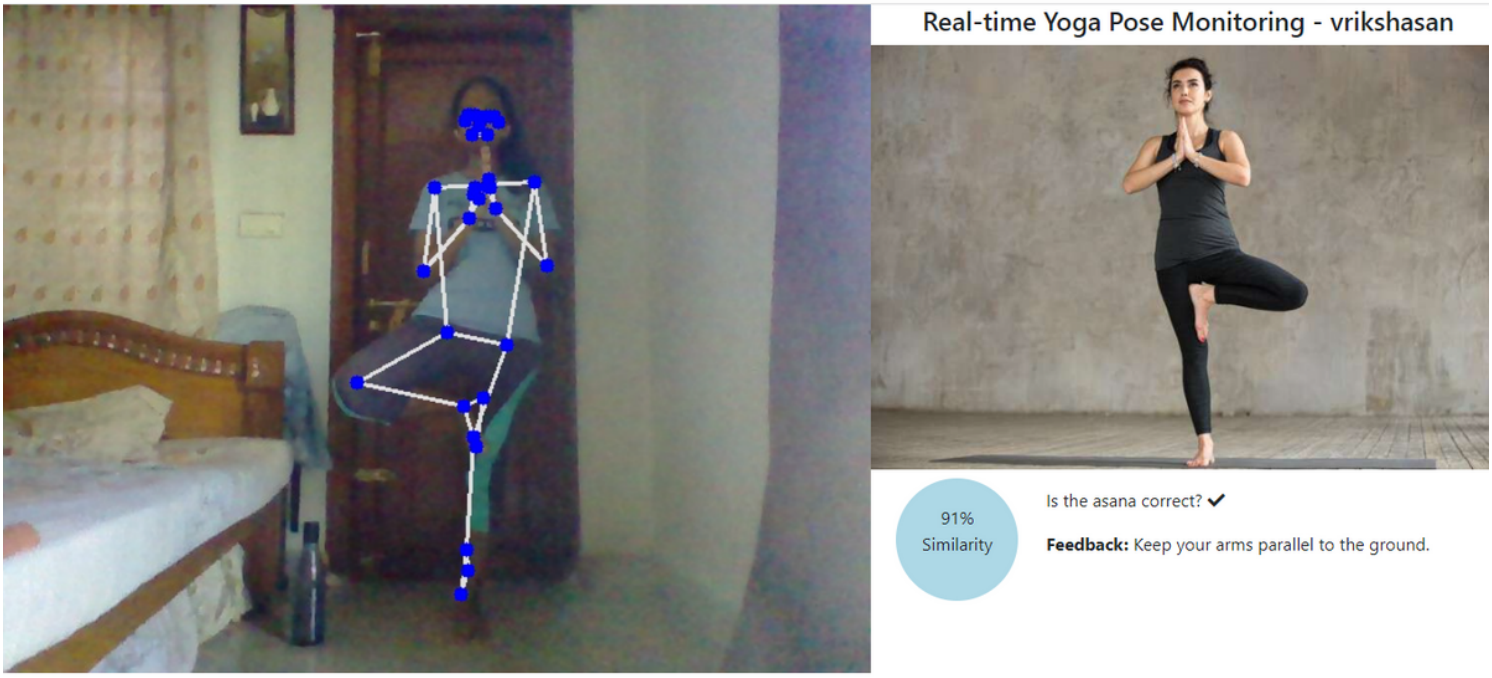


Figure 13

Frontend

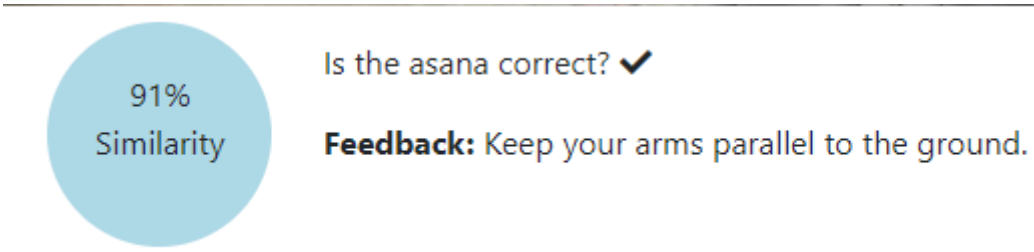


Figure 14

Prediction and correction