Spring 5-20-2020

Yoga Pose Classification Using Deep Learning

Shruti Kothari

Yoga Pose Classification Using Deep Learning

A Thesis

Presented to

Dr. Robert Chun

The Faculty of the Department of Computer Science

San José State University

In Partial Fulfillment

Of the Requirements for the Class

CS298

By

Shruti Kothari

May 2020

The Designated Thesis Committee is pending approval on the Thesis Titled

Yoga Pose Classification Using Deep Learning

By

Shruti Kothari

APPROVED FOR THE DEPARTMENT OF COMPUTER SCIENCE

SAN JOSE STATE UNIVERSITY

May 2020

Dr. Robert Chun, Department of Computer Science

Dr. Katerina Potika, Department of Computer Science

Mr. Susmit Gaikwad, eDriving

ABSTRACT

Human pose estimation is a deep-rooted problem in computer vision that has exposed many challenges in the past. Analyzing human activities is beneficial in many fields like video-surveillance, biometrics, assisted living, at-home health monitoring etc. With our fast-paced lives these days, people usually prefer exercising at home but feel the need of an instructor to evaluate their exercise form. As these resources are not always available, human pose recognition can be used to build a self-instruction exercise system that allows people to learn and practice exercises correctly by themselves. This project lays the foundation for building such a system by discussing various machine learning and deep learning approaches to accurately classify yoga poses on prerecorded videos and also in real-time. The project also discusses various pose estimation and keypoint detection methods in detail and explains different deep learning models used for pose classification.

Keywords – Human pose estimation, yoga, openpose, machine learning, deep learning

TABLE OF CONTENTS

I. INTRODUCTION

Human pose estimation is a challenging problem in the discipline of computer vision. It deals with localization of human joints in an image or video to form a skeletal representation. To automatically detect a person's pose in an image is a difficult task as it depends on a number of aspects such as scale and resolution of the image, illumination variation, background clutter, clothing variations, surroundings, and interaction of humans with the surroundings [1]. An application of pose estimation which has attracted many researchers in this field is exercise and fitness. One form of exercise with intricate postures is yoga which is an age-old exercise that started in India but is now famous worldwide because of its many spiritual, physical and mental benefits [2].

The problem with yoga however is that, just like any other exercise, it is of utmost importance to practice it correctly as any incorrect posture during a yoga session can be unproductive and possibly detrimental. This leads to the necessity of having an instructor to supervise the session and correct the individual's posture. Since not all users have access or resources to an instructor, an artificial intelligence-based application might be used to identify yoga poses and provide personalized feedback to help individuals improve their form [2].

In recent years, human pose estimation has benefited greatly from deep learning and huge gains in performance have been achieved [3]. Deep learning approaches provide a more straightforward way of mapping the structure instead of having to deal with the dependencies between structures manually. [4] used deep learning to identify 5 exercise poses: pull up, swiss ball hamstring curl, push up, cycling and walking. However, using this method for yoga poses is a relatively newer application [2].

This project focuses on exploring the different approaches for yoga pose classification and seeks to attain insight into the following: What is pose estimation? What is deep learning? How can deep learning be applied to yoga pose classification in real-time? This project uses references from conference proceedings, published papers, technical reports and journals. Fig. 1 gives a graphical overview of topics this paper covers. The first section of the project talks about the history and importance of yoga. The second section talks about pose estimation and explains different types of pose estimation methods in detail and goes one level deeper to explain discriminative methods – learning based (deep learning) and exemplar. Different pose extraction methods are then discussed along with deep learning based models - Convolutional Neural Networks (CNNs) and Recurrent Neural Networks (RNNs).

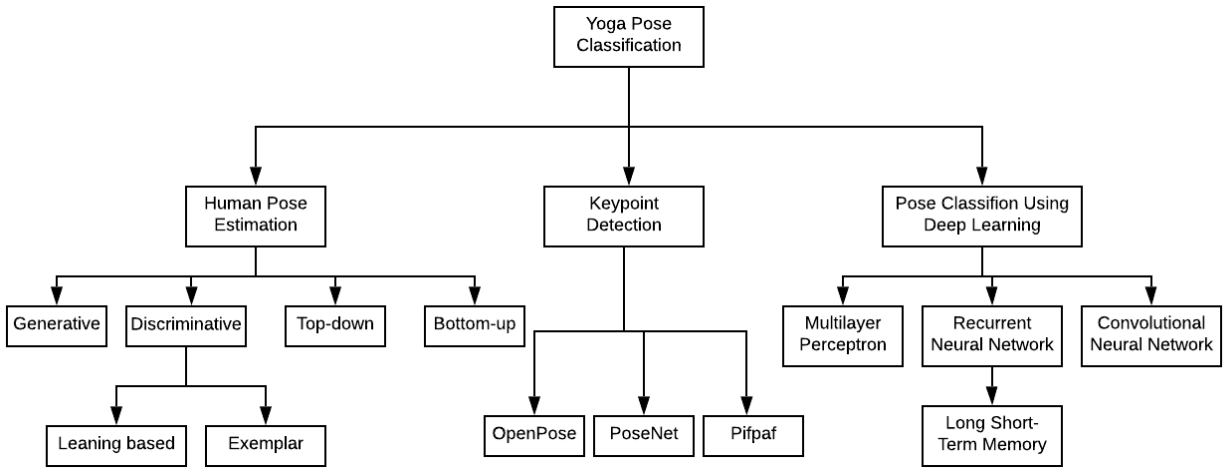


Fig. 1. Conceptual outline of topics

II. HISTORY OF YOGA

Humans are prone to musculoskeletal disorders with aging and accidents [5]. In order to prevent this some, form of physical exercise is needed. Yoga, which is a physical and spiritual exercise, has gained tremendous significance in the community of medical researchers. Yoga has the ability to completely cure diseases without any medicines and improve physical and mental health [6]. A vast body of literature on the medical applications of yoga has been generated which includes positive body image intervention, cardiac rehabilitation, mental illness etc. [6]. Yoga comprises of various asanas which represent physical static postures. The application of pose estimation for yoga is challenging as it involves complex configuration of postures. Furthermore, some state-of-the-art methods fail to perform well when the asana involves horizontal body posture or when both the legs overlap each other. Hence, the need to develop a robust model which can help popularize self-instructed yoga systems arises.

III. HUMAN POSE ESTIMATION

Human posture recognition has made huge advancements in the past years. It has evolved from 2D to 3D pose estimation and from single person to multi person pose estimation. [16] uses pose estimation to build a machine learning application that helps detect shoplifters whereas [17] uses a single RGB camera to capture 3D poses of multiple people in real-time. Human pose estimation algorithms can be widely organized in two ways. Algorithms prototyping estimation of human poses as a geometric calculation are classified as generative methods while algorithms modelling human pose estimation as an image processing problem are classified as discriminative methods [7]. Another way of classifying these algorithms is based on their method of working. Algorithms starting from a higher-level generalization and moving down are called top-down methods, whereas algorithms that start with pixels and move upwards are called bottom-up methods [8].

A. GENERATIVE

Generative procedures provide a technique to predict the features from a given pose hypothesis. They start with initializing the posture of the human body and project it to the image plane. Adjustments are made to make the projected image and current image observations compliant. Generative based approaches offer easy generalization due to less constraint of a training pose dataset [9]. However, due to the high dimensional projection space search, this method is not considered computationally feasible, and is thus slower as compared to discriminative methods. [10] describes a generative Bayesian method to track 3D segmented human body figures in videos. This is a probabilistic method which consists of a generative model for image appearance, an initial probability distribution over joint angles and pose that represents

movement of humans and a robust likelihood function. Even though the method is able to track humans in unknown complicated backgrounds, it faces the risk of eventually losing track of the object.

B. DISCRIMINATIVE

Contrary to generative methods, discriminative methods start with the evidence of the image and learn a technique to model the relationship between the human poses and evidence on the basis of training data. Model testing in discriminative methods is a lot faster as opposed to generative methods due to the search in a constrained space as opposed to a high dimensional feature space [7]. [11] explores a discriminative based learning method to obtain 3D human pose from silhouettes. This approach does not require a body model explicitly nor any prior labelled parts of the body in the image. It restores the pose using non-linear regression based on the shape descriptor vectors fetched automatically from silhouettes of images. It uses Relevance Vector Machine (RVM) regressors and damped least squares for regression [11]. The method, though increasing the accuracy by three times, is not accurate enough, as there are some instances of incorrect poses and results showing significant temporal jitter. Discriminative methods are further categorized into learning methods and exemplar methods [7].

1. LEARNING BASED – DEEP LEARNING:

One important learning-based method is deep learning which is built upon Artificial Neural Networks (ANNs). ANN is analogous to the human brain where the units in an ANN represent the neurons in the human brain, and weights represent the strength of connection between neurons. Deep learning provides an end-to-end architecture that allows automatic learning of key information from images. One popular deep learning model which has been widely used for pose

estimation is Convolutional Neural Network (CNN) which will be discussed later. [20] have contributed to the research by using CNNs and stacked auto-encoder algorithms (SAE) for identifying yoga poses and Indian classical dance forms. However, their performance evaluation is done only on images and not on videos.

2. EXEMPLAR METHODS:

In exemplar methods, pose estimation is based on a unique set of poses with their equivalent representations [7]. Classification algorithms such as random forests and randomized trees are robust and fast enough to manage this. Random forest is made up of various randomized decision trees and is hence called an ensemble classifier. It consists of non-terminal nodes which have a decision function to predict the similarities in images. Fig. 8 gives an example. [19] used an improved version of random forests which comprised of two levels of random forests. The first layer of the tree acted as a discriminative classifier to classify body parts and passed the classification results to the second layer which predicted joint locations in the body. Another approach similar to random forests is Hough forests [7] which consists of decision forest combinations, where the terminal node in every tree is either a regression or classification node. Enhanced Hough trees have a parts objective ("PARTS") which is an optimized objective based on discrete information gain.

C. TOP- DOWN:

Most studies refer to generative methods as top-down methods [7]. The top-down approach obtains keypoints by using a module to detect human subjects to which a pose estimator can be applied. The primary advantage of top-down methods is their ability to break down the task into

multiple smaller tasks. These smaller tasks include detecting the object followed by pose estimation. For this, the detector should be powerful enough to detect hard or comparatively smaller objects so that the performance of the pose estimator is improved. [8] uses a top down approach to articulate human pose estimation and tracking. Their top down approach comprises of three components – a human candidate detector, a single person pose estimator and a human pose tracker [8]. A general object detector is chosen to identify human candidates after which a cascaded pyramid tracker is used to recognize the equivalent human pose and lastly, a flow-based pose tracker is used to assign a distinct and temporally consistent id to each human candidate for multi pose tracking. Even though this research develops a modular system to human pose estimation and tracking, they achieve an average precision of only 69.4 for pose estimation and 68.9 for pose tracking, leaving a lot of scope for improvement.

D. BOTTOM – UP:

This approach involves detection of human keypoints from potential subjects and organizing them into human limbs using several data association mechanisms. The cost of computation in this method does not depend on the number of human subjects in the images. Thus, it provides a very good compensation between cost and accuracy. Some surveys refer to bottom-up methods as discriminative methods [7]. [12] proposes a novel approach that combines traditional bottom-up and top-down methods for multi-person pose estimation. The feed forwarding to the network is done in a bottom-up fashion while parsing of poses along with bounding box constraints is performed in a top-down fashion. The features from the image are extracted using a residual network which is trained to learn the confidence map of the joints and

relationships of connectivity between the joints. This residual network is nothing but ResNet50, which is a deep neural network architecture. Although this research does not focus on classifying the poses, it shows how deep learning architectures like ResNet50 can be used in order to make human pose estimation accurate.

IV. KEYPOINT DETECTION METHODS

A. OPENPOSE:

OpenPose is a multi-person real-time keypoint detection which brought a revolution in the field of pose estimation. It was invented in Carnegie Mellon University (CMU) by the Perceptual Computing Lab[13]. It uses CNN based architecture to identify facial, hand and foot keypoints of a human body from single images. OpenPose helps identify human body joints using an RGB camera. OpenPose keypoints include eyes, ears, neck, nose, elbows, shoulders, knees, wrists, ankles and hips. It presents the results obtained by processing inputs from a camera in real-time or pre-recorded videos or static images as 18 simple keypoints. Because of this, it finds its use in a variety of applications ranging from sports, surveillance, activity detection to yoga pose recognition. The work proposed in [18] uses OpenPose for initial keypoint identification followed by CNN for classification of yoga poses. However, they achieve an accuracy of only 78% which could be due to the limited dataset they used or architecture and hyperparameter tuning of their CNN model.

The first stage in OpenPose is detecting keypoints of every person in the image which is followed by assigning parts to each distinct individual. Fig. 2 depicts the architecture of the OpenPose model [13]. OpenPose network starts with extraction of features from the image using the initial layers (VGG -19 as shown in Fig. 2). These features are then passed to two convolutional layer branches which run in parallel. A prediction of 18 confidence maps, which represents specific parts of the human body, is made by the first branch. On the other hand, 38 Part Affinity Fields (PAF) which denote the association degree between parts is predicted by the second branch. More stages are used to make refinement to the predictions made from the previous branch. Bipartite

graphs are formed between different parts using part confidence maps. The links which are weaker in these graphs are removed using the PAF values. With these steps, human skeletons are estimated for every person in the frame or image.

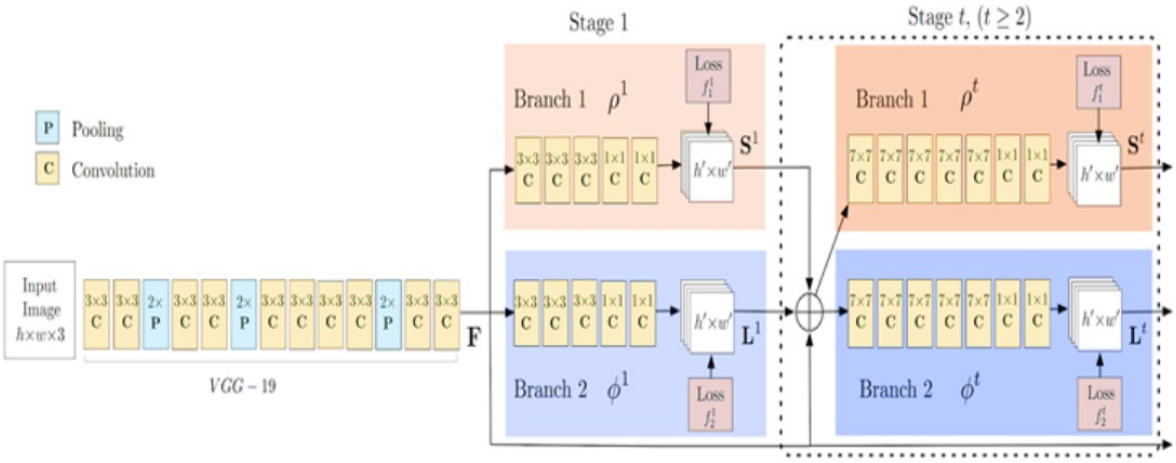


Fig. 2. OpenPose architecture

B. POSENET:

PoseNet is another deep learning framework similar to OpenPose which is used for identification of human poses in images or video sequences by identifying joint locations in a human body. These joint locations or keypoints are indexed by "Part ID" which is a confidence score whose value lies in the range of 0.0 and 1.0 with 1.0 being the greatest. The PoseNet model's performance varies depending on the device and output stride [14]. The PoseNet model is invariant to the size of the image, thus it can predict pose positions in the scale of the actual image irrespective of whether the image has been downscaled.

In PoseNet, the softmax layer is replaced by a sequence of fully connected layers. A high-level architecture of PoseNet is shown in Fig. 3 [31]. The first component in the architecture is an encoder which is responsible for generating the encoding vector v, a 1024-dimensional vector that is an encoded representation of the features of the input image. The second component is the localizer which generates vector u which denotes localization features. The last component is a regressor which consists of two connected layers that are used to regress the final pose.

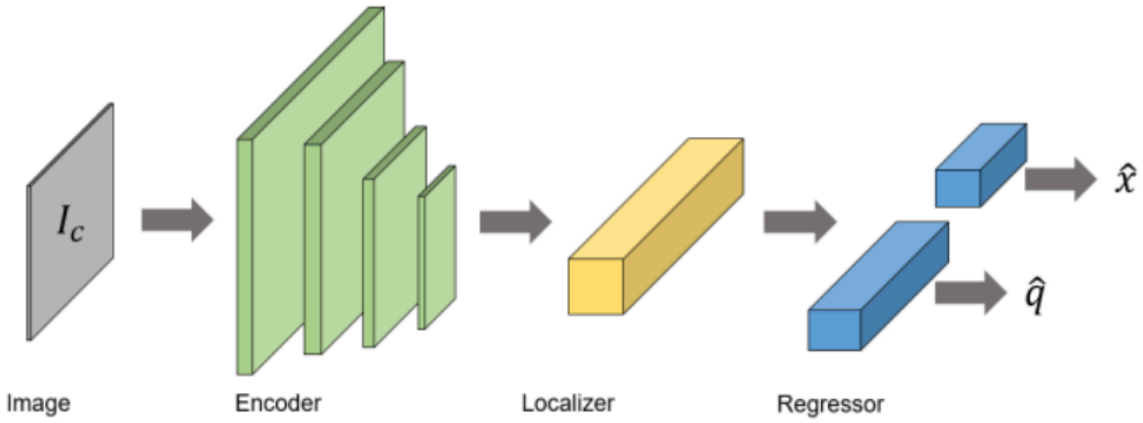


Fig. 3. System architecture of PoseNet

C. PIFPAF:

PifPaf is a new method based on the bottom-up approach for 2D multi-person human pose estimation. It uses a Part Intensity Field (PIF) for body part localization and a Part Association Field (PAF) for association of body parts to form full human poses [15]. The model beats other methods in terms of a lower resolution and better performance in overcrowded places mainly due to the following: (a) fine information encoded in a newer composite field PAF, (b) the selection of Laplace loss that integrates an opinion of uncertainty. The model architecture rests upon a completely convolutional box-free design [15].

Fig. 4 represents the architecture of PifPaf [15]. The input image is of size (H, W). It has the RGB channels which is shown by 'x3'. The encoder is based on neural networks, and it generates the PIF field with 17 x 5 channels and PAF field with 19×7 channels. '//2' represents an operation with strides of 2. The PIF and PAF fields are converted by the decoder into pose coordinates which have 17 joints each. Every joint is a 2D representation and has X and Y coordinates along with a confidence score.

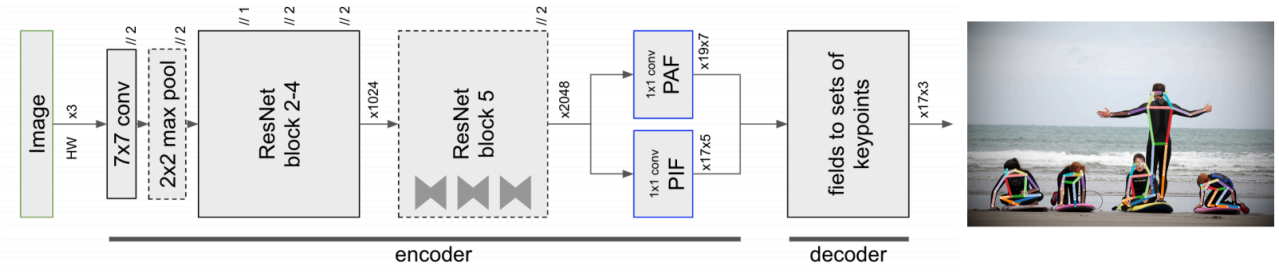


Fig. 4. Pifpaf architecture

This model architecture is a ResNet based network. The confidence, exact location and joint size is predicted by one of the head networks which is called PIF, and the associations between different parts are predicted by the other head network which is PAF. The method is hence called PifPaf.

V. POSE CLASSIFICATION USING DEEP LEARNING

Deep learning is widely used for image classification tasks wherein the model takes input in the form of images and outputs a prediction. Deep learning algorithms use neural networks to determine the connection between the input and output. For pose estimation problems, the image with pose of individuals is taken as input and the deep learning model tries to learn correctly the different poses so as to accurately classify them. As one can guess, this could be a computationally expensive task if the number of images is large. Also, as we want accurate results we would not want to compromise on the quality of the images as that could affect the features extracted by the model. Thus, in this project we propose using OpenPose (a pretrained model) to extract keypoints of the human joint locations from the images and then training the deep learning model on these keypoints. Below are some basic deep learning models used for classification problems.

A. MULTILAYER PERCEPTRON (MLP)

MLP is a classical neural network that has one input and one output layer. The intermediate layers between the input and output layer are known as hidden layers. There can be one or more hidden layers. MLPs form a fully connected network as every node in one layer has a connection to every node in another layer. A fully connected network is a foundation for deep learning. MLP is popular for supervised classification where the input data is assigned a label or class. [21] uses MLP for human pose classification by extracting keypoints from low resolution images using Kinect sensor.

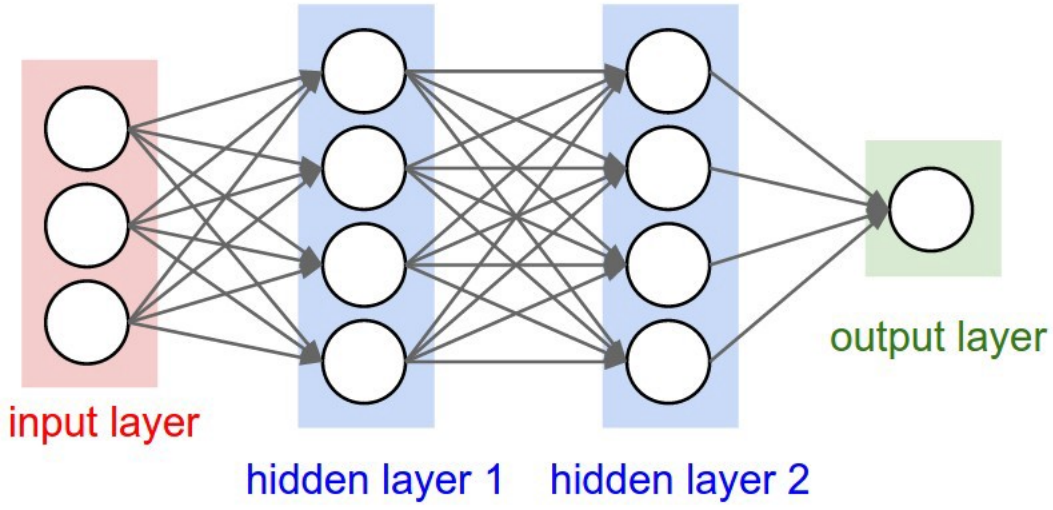


Fig. 5. Schematic diagram of multilayer perceptron

B. RECURRENT NEURAL NETWORK (RNN)

RNNs are neural network architectures that are used for sequence prediction problems. Sequence prediction problems can be one to many, many to one, or many to many. In RNNs, the previous data of a neuron is preserved which helps in handling the sequential data. As a result, the context is preserved, and output is generated taking into account the previously learned data. RNNs are most commonly used for natural language processing (NLP) problems where the input is naturally modeled in sequences.

However, in activity recognition or pose classification tasks too, there is a dependency between the previously performed action and the next action. In case of yoga as well, the context or information of initial or intermediary poses is important in predicting the final pose. Yoga can thus be thought of as a sequence of poses. This makes RNNs a suitable choice for yoga pose

classification as sequential estimation of joint locations can be able to better capture the dependency between joint locations. For the same reason, [22] use RNN for human pose estimation.

The problem with RNNs however is that they are unable to preserve long term dependencies. Sometimes, recent information is sufficient to conduct the current task, but there are cases when the gap between the relevant information and the present task becomes too large. In such cases RNNs fail as they are unable to connect this information. In the context of yoga, if the intermediary steps in a yoga asana are too many, RNNs find it difficult to keep track of the initial steps which are needed in order to predict the current task. This problem is called as the long term dependency problem.

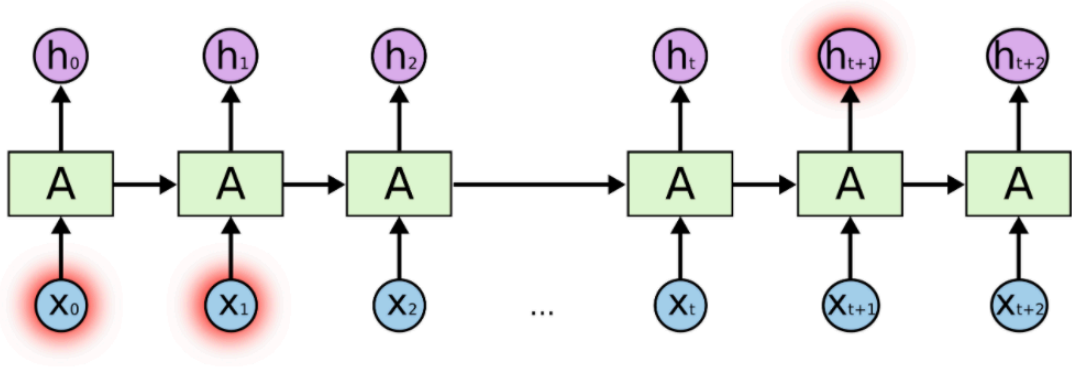


Fig. 6. Long-term dependency in RNN

1. LONG SHORT-TERM MEMORY (LSTM):

In order to deal with the above long-term dependency problem, a special type of RNN exists which is called LSTM. An LSTM is a famous RNN that can easily remember information or data for long enough periods of time which is its default behavior. The key idea which makes this possible is cell state. A cell state allows unchanged information flow. It can be thought of as a conveyor belt. LSTMs can add and eliminate data from the cell state using regulatory structures known as gates. These special gates allow for optionally letting the information through. LSTM uses three gates, in particular input, update and forget. An LSTM can thus selectively forget or remember the learnings. As LSTMs allow for longer retention of the input state in the network, they can efficiently process long sequences and give good results. [23] talks about how LSTM can be used with CNN for human activity recognition to achieve high accuracy.

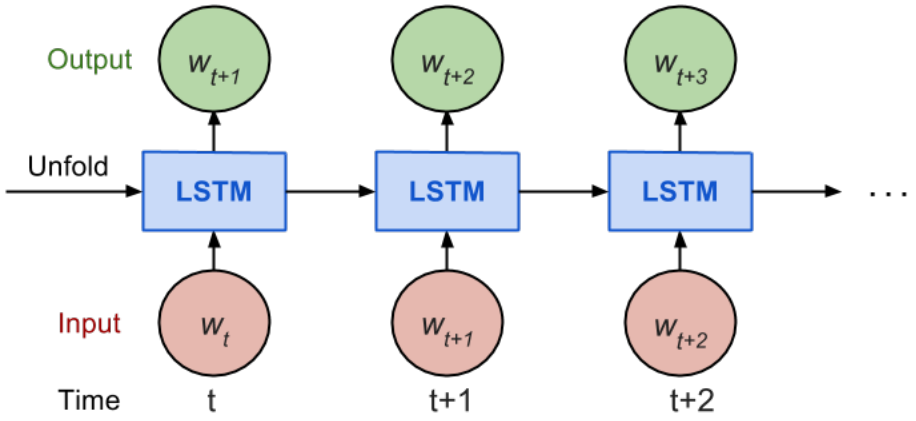


Fig. 7. LSTM architecture

C. CONVOLUTIONAL NEURAL NETWORK (CNN)

CNN is a type of neural network which is widely used in the computer vision domain. It has proved to be highly effective such that it has become the go-to method for most image data. CNNs consist of a minimum of one convolutional layer which is the first layer and is responsible for feature extraction from the image. CNNs perform feature extraction using convolutional filters on the input and analyzing some parts of the input at a given time before sending the output to the subsequent layer. The convolutional layer, through the use of convolutional filters, generates what is called a feature map. With the help of a pooling layer, the dimensionality is reduced, which reduces the training time and prevents overfitting. The most common pooling layer used is max pooling, which takes the maximum value in the pooling window.

CNNs show a great promise in pose classification tasks, thus making it a highly desirable choice. They can be trained on keypoints of joint locations of the human skeleton or can be trained directly on the images. [4] used CNN to detect human poses from 2D human exercise images and achieved an accuracy of 83%. On the other hand, [18] used CNN on OpenPose keypoints to classify yoga poses and achieved an accuracy of 78%. Although, the accuracy is not exactly comparable as the dataset along with the CNN architecture and exercises being classified are different, [18] shows how using CNNs on OpenPose keypoints is worth exploring.

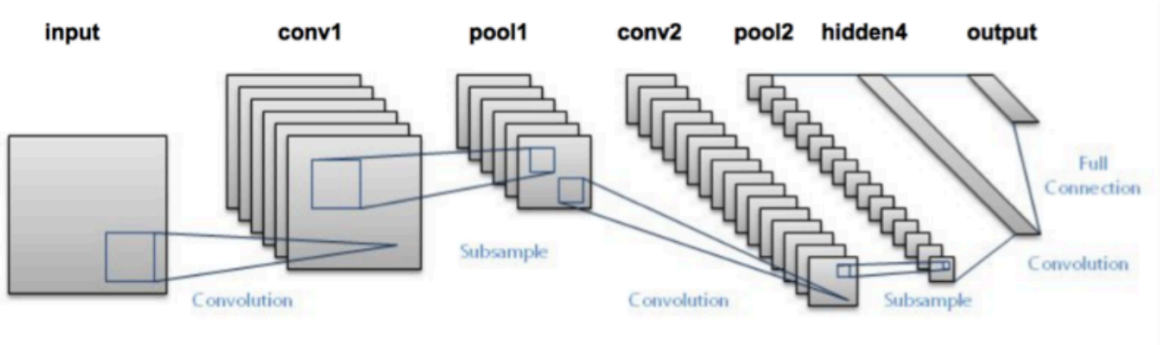


Fig. 8. CNN architecture layers

In the case of keypoints, CNN extracts features from 2D coordinates of the OpenPose keypoints using the same convolutional filter technique explained above. Based on the filter size, the convolutional filter slides to the next set of input. After the convolution, an activation function Rectified Linear Unit (ReLU) is generally applied to add nonlinearity in the CNN, as the real-world data is mostly nonlinear and the convolution operation by itself is linear. Tanh and sigmoid are other activation functions, but ReLU is mostly used because of its better performance.

VI. SUMMARY OF CURRENT STATE OF THE START

A lot of work has been done in the past in building systems that are automated or semiautomated which help to analyze exercise and sports activities such as swimming [24], basketball [25] etc. Patil et al. [26], proposed a system for identifying yoga posture differences between an expert and a practitioner using speeded up robust features (SURF) which uses information of image contours. However, describing and comparing the postures almost by using only the contour information is not sufficient.

A system for yoga training has been proposed by Luo et al. [27] which consists of inertial measurement units (IMUs) and tactors. But this can be uncomfortable to the user and at the same time affect the natural yoga pose. [28] presented a system for yoga pose detection for six poses using Adaboost classifier and Kinect sensors and achieved an accuracy of 94.8%. However, they have used a depth sensor based camera that may not be always accessible to users. Another system for yoga pose correction using Kinect has been presented by [29] which takes into account three yoga poses, warrior III, downward dog and tree pose. However, their results are not very impressive, and their accuracy score is only 82.84%. The traditional method of skeletonization has now been replaced by deep learning-based methods.

Deep learning is a promising domain where a lot of research is being done, enabling us to analyze tremendous data in a scalable manner. As compared to traditional machine learning models where feature extraction and engineering is a must, deep learning eliminates the necessity to do so by understanding complex patterns in the data and extracting features on its own.

VII. HYPOTHESIS

A deep learning model for classifying yoga poses can be built where the initial keypoint extraction of the human joint locations is done using OpenPose. The model can incorporate feature extraction capabilities of CNN along with context retention abilities of LSTM to effectively classify yoga poses in prerecorded videos and also in real time [2]. This model can be thought of as a hybrid model.

We also plan to experiment with basic CNN networks and compare the performance with the hybrid model. Kinect sensors could be a way to perform human pose estimation, but it accounts for additional equipment and specialized hardware and the performance is not always good in different surroundings. Machine learning models, although not widely used for human pose estimation, will be explored for comparison with the deep learning models.

The evaluation of the yoga pose classification system will be done by using classification scores, confusion matrix and evaluations by people. The system will predict the yoga pose sequence being performed by the user in real-time and we can examine if the prediction made by the system is correct. The results will also be compared to existing methods.

VIII. EVALUATION METRICS

A. Classification Score:

Classification score refers to what we usually mean by accuracy of the model. It can be described as the proportion of number of predictions that were correct to the total input samples.

$$Accuracy = \frac{Number\ of\ Correct\ predictions}{Total\ number\ of\ predictions\ made}$$

In case of multiclass classification, this metric gives good results when the number of samples in each class are almost the same.

B. Confusion matrix

Confusion matrix represents a matrix which explains the accuracy of the model completely. There are four important terms when it comes to measuring the performance of a model.

- True Positive: Predicted value and the actual output are both 1.
- True Negative: Predicted value and the actual output are both 0.
- False Positive: Predicted value is 1 but the actual output is 0.
- False Negative: Predicted value is 0 but the actual output is 1.

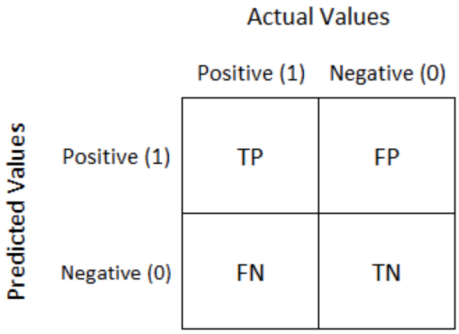


Fig. 9. Sample confusion matrix

Fig. 9. shows a basic confusion matrix for binary classification. The diagonal values represent the samples that are correctly classified and thus, we always want the diagonal of the matrix to contain the maximum value. In case of a multiclass classification, each class represents one row and column of the matrix.

C. Model accuracy and model loss curve:

These curves are also referred to as learning curves and are mostly used for models that learn incrementally over time, for example, neural networks. They represent the evaluation on the training and validation data which gives us an idea of how well the model is learning and how well is it generalizing. The model loss curve represents a minimizing score (loss), which means that a lower score results in better model performance. The model accuracy curve represents a maximizing score (accuracy), which means that a higher score denotes better performance of the model. A good fitting model loss curve is one in which the training and validation loss decrease and reach a point of stability and have minimal gap between the final loss values. On the other hand, a good fitting model accuracy curve is one in which the training and validation accuracy increase and become stable and there is a minimum gap between the final accuracy values.

IX. DATASET

The dataset used for this project is a part of the Open Source collection and is publicly available [30]. This dataset has been created by [2]. It consists of videos of 6 yoga poses performed by 15 different individuals (5 females and 10 males). The 6 yoga poses namely are – Bhujangasana (Cobra pose), Padmasana (Lotus pose), Shavasana (Corpse pose), Tadasana (Mountain pose), Trikonasana (Triangle pose) and Vrikshasana (Tree pose). The total number of videos is 88 with a duration of 1 hour 6 minutes and 5 seconds. The rate at which the videos have been recorded is 30 FPS (frames per second). All the videos have been recorded in an indoor environment at a distance of 4 meters from the camera. All individuals have performed yoga poses with variations so as to help build a dataset which can be used to build a robust yoga pose recognition system. The average length of all videos is about 45 to 60 seconds. Fig. 10 represents the number of videos for each yoga pose performed by the number of individuals [2].

Sr No.	Yoga pose	No. of videos
1	Bhujangasana	16
2	Padmasana	14
3	Shavasana	15
4	Tadasana	15
5	Trikonasana	13
6	Vrikshasana	15
	Total videos	88

Fig. 10. Dataset details

The frames from videos of individuals performing different yoga poses is shown in Fig. 11. Videos with different subjects have been used for the train, test and validation sets.

Fig. 11. Dataset frames

X. DATA PREPROCESSING

The first step in preprocessing the data is extracting keypoints of poses in video frames using the OpenPose library. For recorded videos, pose extraction is done offline whereas for real-time, it is done online wherein keypoints identified from the inputs to the camera are supplied to the model. OpenPose is run on each frame of the video and the corresponding output of each frame is stored in JSON format. This JSON data includes the locations of body parts of each person identified in the video frame. Default setting of OpenPose has been used for extracting pose keypoints for ideal performance. Fig. 12 depicts the 18 keypoint positions captured by OpenPose [2].

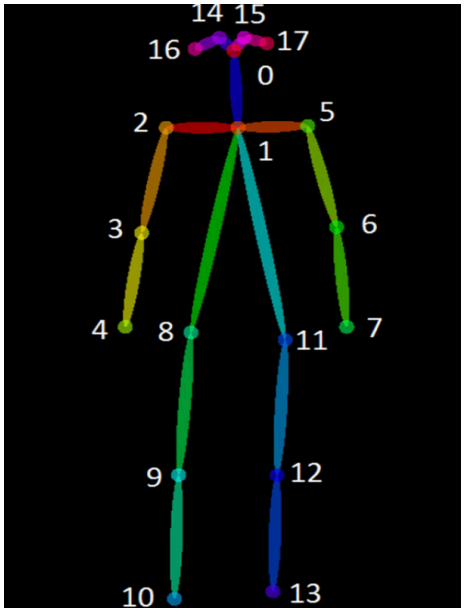


Fig. 12. Keypoints identified by OpenPose

The JSON data is fetched and stored in numpy arrays in sequences of 45 frames which is about 1.5 seconds of the video [2]. 60% of the dataset has been used for training, 20% for testing and 20% for validation. The training data has 7989 sequences of 45 frames, each containing the

2D coordinates of the 18 keypoints captured by OpenPose. The validation data consists of 2224 such sequences and the test data contains 2598 sequences.

The number of frames varied from 60,20,20 split at the video level. This was because of the difference in duration of videos. Fig. 13 shows the keypoints detected by OpenPose on all 6 yoga poses [2].

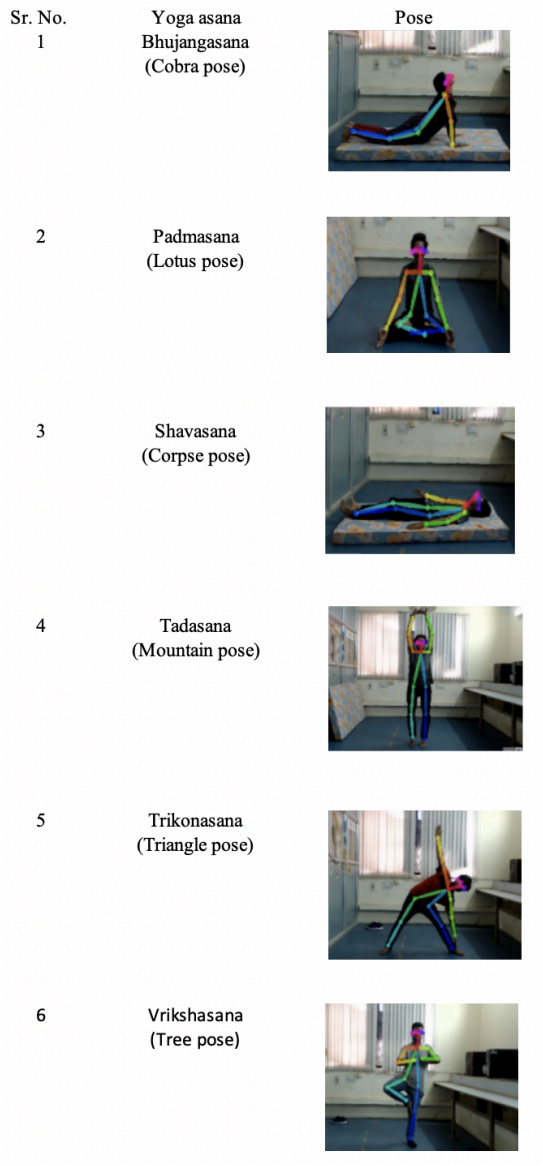


Fig. 13. OpenPose on different poses

XI. MODEL PERFORMANCE AND RESULTS

Training Setup:

The models are built using Python libraries such as TensorFlow - Keras (Theano backend), OpenPose, NumPy, Scikit Learn on a system with NVIDIA Tesla 1080 GPU having 4 GB memory.

1. Support Vector Machine (SVM)

SVM is a supervised machine learning model that is inherently a two-class classifier. However, as most problems involve multiple classes, a multiclass SVM is often used. A multiclass SVM forms multiple two class classifiers and differentiates the classifiers on the basis of the distinct label vs. the rest (one-vs- rest or one-vs-all) or between each pair of classes (one-vs-one). SVM performs the classification by creating a hyperplane in such a way that separation between classes is as wide as possible.

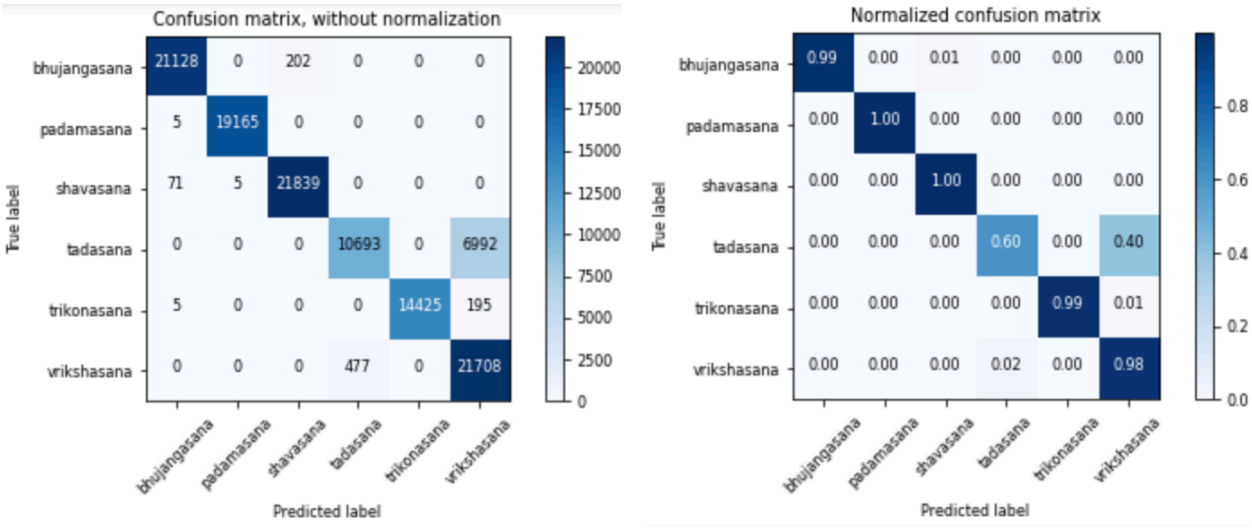
A default SVM has been trained on the training data with the radial basis function (rbf) kernel. Rbf is the default and most popular kernel which is a gaussian radial basis function. It provides more flexibility as compared to other kernels, linear and polynomial. The value of the soft margin parameter C is 1 and the decision function is one-vs-rest. The keypoints captured using OpenPose are used as features to SVM. These 18 keypoints are represented by X and Y coordinates which makes the total number of features as 36 (18 * 2). The data is reshaped to make the number of samples equal.

A. Results:

Train accuracy: 0.9953

Validation accuracy: 0.9762

Test accuracy: 0.9319



B. Analysis

The training accuracy of the model is pretty high at 0.99. There is a slight decrease in the validation and test accuracy, but the results are still good. We can see in the confusion matrix that most classes are classified correctly except for tadasana (mountain pose). Out of 17,685 frames for tadasana, 6992 have been misclassified as vrikshasana (tree pose) and similarly there is some incorrect classification for vrikshasana. This could be because of the similarity in the poses as both of them require a standing position and also the initial pose formation is similar.

2. Convolutional Neural Network

A one dimensional, one-layer CNN with 16 filters of size 3 x 3 is trained on the OpenPose keypoints. The input shape is 18 x 2 which signifies the 18 keypoints having X and Y coordinates. Batch normalization is applied to the output of the CNN layer so that the model converges faster. We also have a dropout layer that prevents overfitting by randomly dropping some fraction of the weights. The activation function used is Rectified Linear Unit (ReLU) which is applied for feature extraction on keypoints of each frame. The final output is flattened before being passed to the dense layer with softmax activation and 6 units where every unit represents the likelihood of a yoga pose in cross entropy terms for all 6 classes. The model architecture summary is shown in Fig. 14.

```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv1d_1 (Conv1D)            (None, 18, 16)            112
_________________________________________________________________
batch_normalization_1 (Batch (None, 18, 16)            64
_________________________________________________________________
dropout_1 (Dropout)          (None, 18, 16)            0
_________________________________________________________________
flatten_1 (Flatten)          (None, 288)               0
_________________________________________________________________
dense_1 (Dense)              (None, 6)                 1734
=================================================================
Total params: 1,910
Trainable params: 1,878
Non-trainable params: 32
_________________________________________________________________
None
```

Fig. 14. Model layers

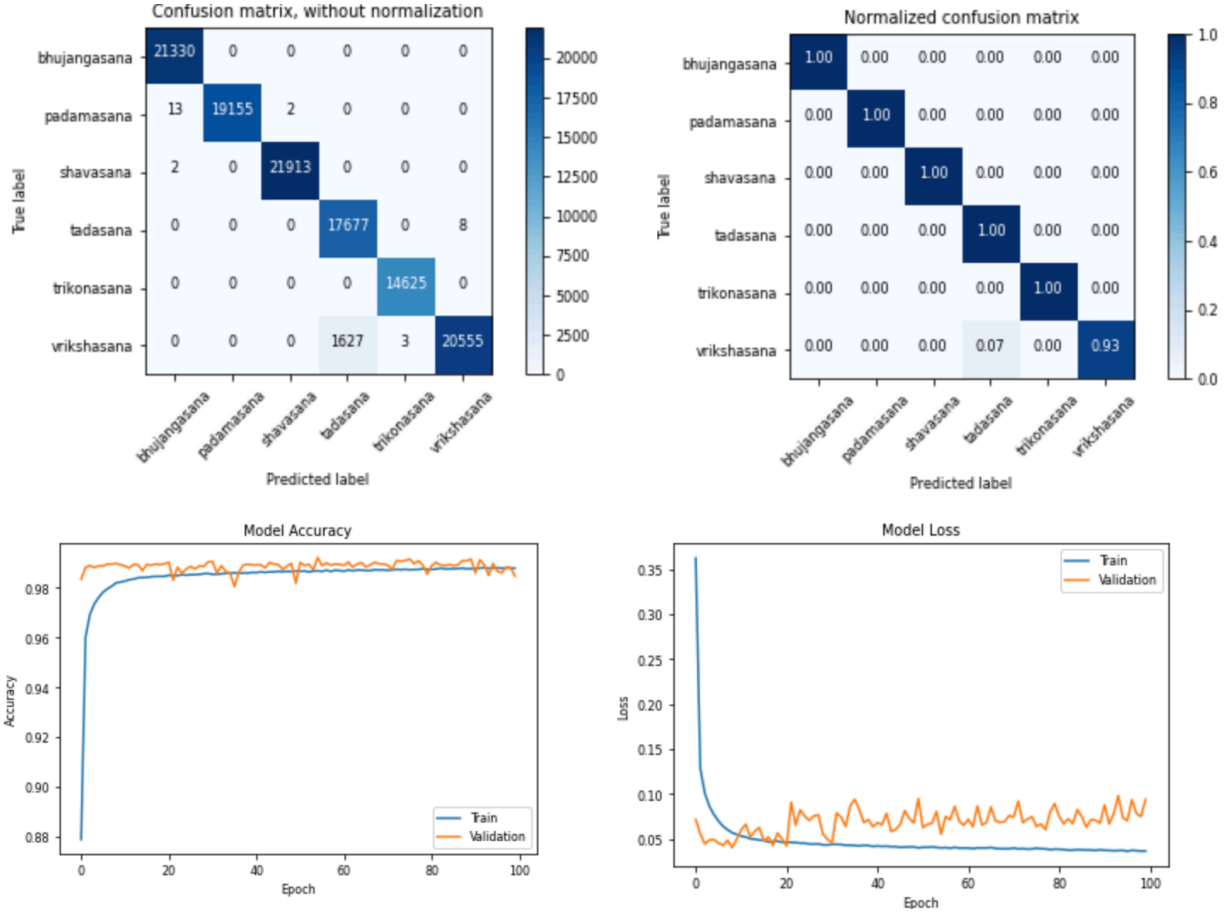
The loss function used for compiling the model is categorical cross - entropy which is also called softmax loss. This is used as it allows measuring the performance of the output of the densely connected layer with softmax activation. This loss function is used for multi class classification, and as we have multiple yoga pose classes, it makes sense to use categorical cross-entropy. Finally, to manage the learning rate, adam optimizer with an initial learning rate of 0.0001 is used. The total number of epochs for which the model is trained is 100.

A. Results:

Train accuracy: 0.9878

Validation accuracy: 0.9921

Test accuracy: 0.9858



B. Analysis

The training, validation and test accuracy of the model are almost the same, approximately 0.99. The confusion matrix further shows that the model does an excellent job of classifying all samples correctly, except for some samples in vrikshasana which are misclassified as tadasana, leading to 93% accuracy for vrikshasana. When compared to SVM, there are fewer misclassifications in CNN. However, the model loss curve above shows an increase in the validation loss, and a decrease in the training loss which shows that there is some overfitting.

3. Convolutional Neural Network + Long Short Term Memory

A deep learning model, CNN and LSTM is used [2]. CNN is used for analyzing frames and identifying patterns, while LSTM makes predictions on the temporal data. The CNN layer extracts features from the keypoints and passes it to the LSTM cells which examine variations in the attributes over different frames. The shape of the CNN input is 45 x 18 x 2 which represents 45 frames, 18 keypoints in each frame and each keypoint having 2 coordinates: X and Y. The convolution layers are time distributed which sends the output from CNN over 45 frames (1.5 seconds) of data to the LSTM as a sequence. The time distributed layer is beneficial for actions with movements and is hence used. The CNN output is flattened into a one dimensional vector and given as input to the LSTM layer which has 20 units, each unit having a forget bias of 0.5. The temporal changes in the attributes detected by the CNN are identified by LSTM which helps leverage the sequential nature of the input video data, thus treating the entire yoga pose beginning from its inception to pose changes and release as a complete activity. The rest of the CNN architecture is the same as model 2. A diagrammatic representation of the model is shown in Fig. 15 [2].

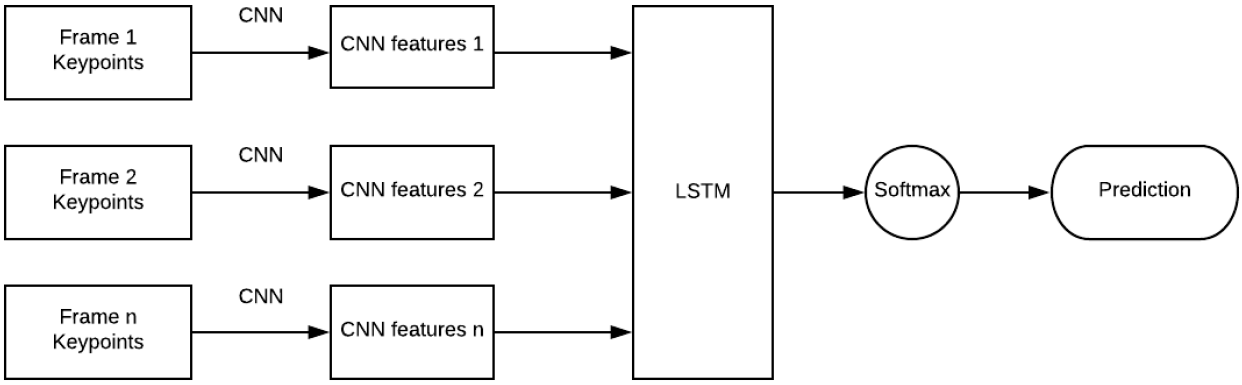


Fig. 15. Model architecture diagram

The model architecture summary is shown in Fig. 16.

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
time_distributed_1 (TimeDist (None, 45, 18, 16)        112
_________________________________________________________________
time_distributed_2 (TimeDist (None, 45, 18, 16)        64
_________________________________________________________________
time_distributed_3 (TimeDist (None, 45, 18, 16)        0
_________________________________________________________________
batch_normalization_2 (Batch (None, 45, 18, 16)        64
_________________________________________________________________
time_distributed_4 (TimeDist (None, 45, 288)           0
_________________________________________________________________
lstm_1 (LSTM)                (None, 45, 20)            24720
_________________________________________________________________
time_distributed_5 (TimeDist (None, 45, 6)             126
=================================================================
Total params: 25,086
Trainable params: 25,022
Non-trainable params: 64
_________________________________________________________________
```

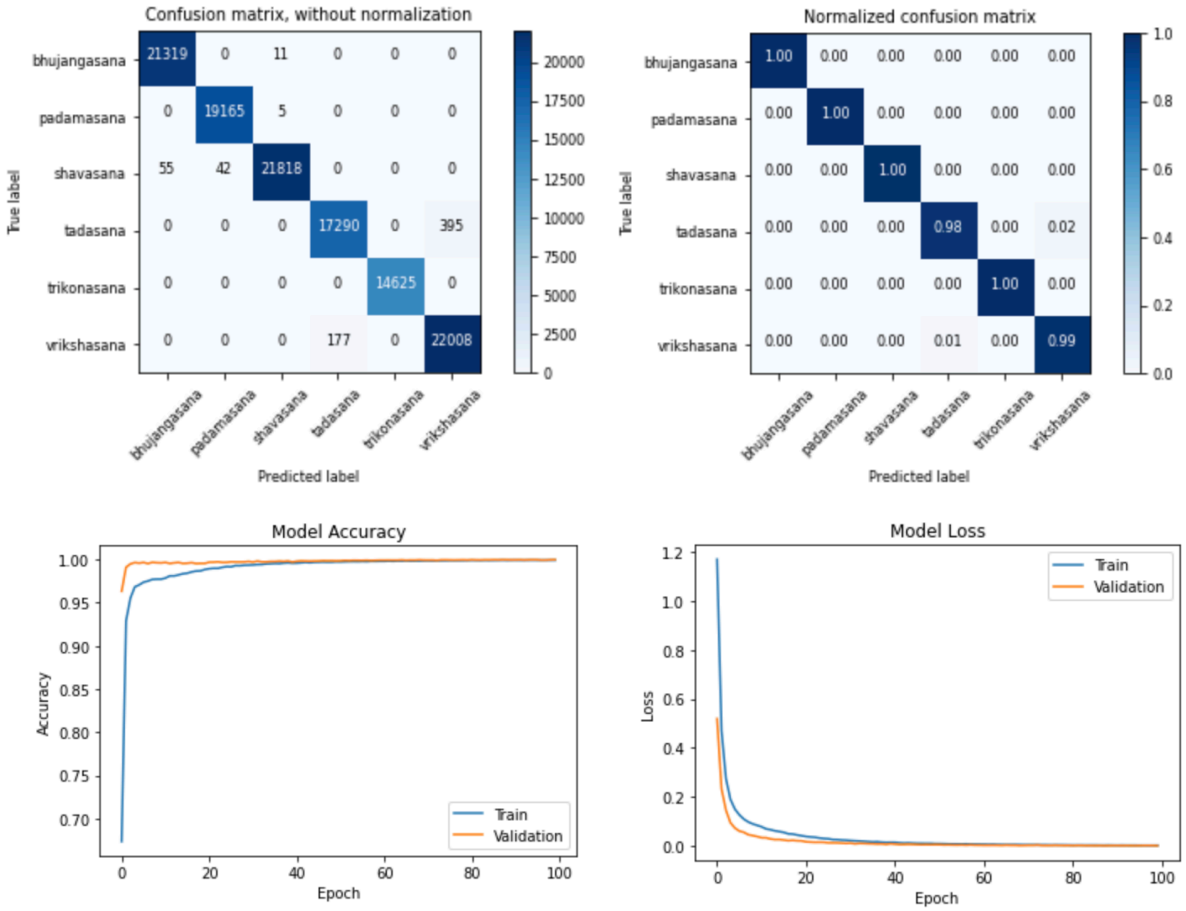
Fig. 16. Model (CNN + LSTM) layers

A. Results:

Train accuracy: 0.9992

Validation accuracy: 0.9987

Test accuracy: 0.9938



B. Analysis

The classification scores are almost close to 1 thereby showing perfect classification for all classes. The diagonal in the normalized confusion matrix is 1.0 for all classes except 0.99 for vrikshasana. The number of misclassifications for vrikshasana is only 177 which is much less as compared to the previous two models. Also, the model accuracy and model loss curve show a good fit with no fluctuations.

XII. CONCLUSION

Human pose estimation has been studied extensively over the past years. As compared to other computer vision problems, human pose estimation is different as it has to localize and assemble human body parts on the basis of an already defined structure of the human body. Application of pose estimation in fitness and sports can help prevent injuries and improve the performance of people's workout. [3] suggests, yoga self-instruction systems carry the potential to make yoga popular along with making sure it is performed in the right manner. Deep learning methods are promising because of the vast research being done in this field. The use of hybrid CNN and LSTM model on OpenPose data is seen to be highly effective and classifies all the 6 yoga poses perfectly. A basic CNN and SVM also perform well beyond our expectations. Performance of SVM proves that ML algorithms can also be used for pose estimation or activity recognition problems. Also, SVM is much lighter and less complex when compared to a neural network and requires less training time.

XIII. FUTURE WORK

The proposed models currently classify only 6 yoga asanas. There are a number of yoga asanas, and hence creating a pose estimation model that can be successful for all the asanas is a challenging problem. The dataset can be expanded my adding more yoga poses performed by individuals not only in indoor setting but also outdoor. The performance of the models depends upon the quality of OpenPose pose estimation which may not perform well in cases of overlap between people or overlap between body parts. A portable device for self-training and real-time predictions can be implemented for this system. This work demonstrates activity recognition for practical applications. An approach comparable to this can be utilized for pose recognition in tasks such as sports, surveillance, healthcare etc. Multi-person pose estimation is a whole new problem in itself and has a lot of scope for research. There are a lot of scenarios where single person pose estimation would not suffice, for example pose estimation in crowded scenarios would have multiple persons which will involve tracking and identifying pose of each individual. A lot of factors such as background, lighting, overlapping figures etc. which have been discussed earlier in this survey would further make multi-person pose estimation challenging.

REFERENCES

[1] L. Sigal. “Human pose estimation”, *Ency. of Comput. Vision, Springer* 2011.

[2] S. Yadav, A. Singh, A. Gupta, and J. Raheja, “Real-time yoga recognition using deep learning”, *Neural Comput. and Appl.*, May 2019. [Online]. Available: https://doi.org/10.1007/s00521-019-04232-7

[3] U. Rafi, B. Leibe, J.Gall, and I. Kostrikov, “An efficient convolutional network for human pose estimation”, *British Mach. Vision Conf.,* 2016.

[4] S. Haque, A. Rabby, M. Laboni, N. Neehal, and S. Hossain, “ExNET: deep neural network for exercise pose detection”, *Recent Trends in Image Process. and Pattern Recog*., 2019.

[5] M. Islam, H. Mahmud, F. Ashraf, I. Hossain and M. Hasan, "Yoga posture recognition by detecting human joint points in real time using microsoft kinect", *IEEE Region 10 Humanit*. Tech. Conf., pp. 668-67, 2017.

[6] S. Patil, A. Pawar, and A. Peshave, “Yoga tutor: visualization and analysis using SURF algorithm”,Proc. *IEEE Control Syst. Graduate Research Colloq*.,pp. 43-46, 2011.

[7] W. Gong, X. Zhang, J. Gonzàlez, A. Sobral, T. Bouwmans, C. Tu, and H. Zahzah, “Human pose estimation from monocular images: a comprehensive survey”, *Sensors*, Basel, Switzerland, vol. 16, 2016.

[8] G. Ning, P. Liu, X. Fan and C. Zhan, “A top-down approach to articulated human pose estimation and tracking”, *ECCV Workshops*, 2018.

[9] A. Gupta, T. Chen, F. Chen, and D. Kimber, “Systems and methods for human body pose estimation”, *U.S. patent*, 7,925,081 B2, 2011.

[10] H. Sidenbladh, M. Black, and D. Fleet, “Stochastic tracking of 3D human figures using 2D image motion”, *Proc 6th European Conf. Computer Vision*, 2000.

[11] A. Agarwal and B. Triggs, “3D human pose from silhouettes by relevance vector regression”, *Intl Conf. on Computer Vision & Pattern Recogn*.pp.882–888, 2004.

[12] M. Li, Z. Zhou, J. Li and X. Liu, “Bottom-up pose estimation of multiple person with bounding box constraint”, *24th Intl. Conf. Pattern Recogn*.,2018.

[13] Z. Cao, T. Simon, S. Wei, and Y. Sheikh, “OpenPose: realtime multi-person 2D pose estimation using part affinity fields”, *Proc. 30th IEEE Conf. Computer Vision and Pattern Recogn*,2017.

[14] A. Kendall, M. Grimes, R. Cipolla, “PoseNet: a convolutional network for real-time 6-DOF camera relocalization”, *IEEE Intl. Conf. Computer Vision*, 2015.

[15] S. Kreiss, L. Bertoni, and A. Alahi, “PifPaf: composite fields for human pose estimation”, *IEEE Conf. Computer Vision and Pattern Recogn*, 2019.

[16] P. Dar, “AI guardman – a machine learning application that uses pose estimation to detect shoplifters”. [Online]. Available: https://www.analyticsvidhya.com/blog/2018/06/ai-guardman-machine-learning-application-estimates-poses-detect-shoplifters/

[17] D. Mehta, O. Sotnychenko, F. Mueller and W. Xu, “XNect: real-time multi-person 3D human pose estimation with a single RGB camera”, *ECCV*, 2019.

[18] A. Lai, B. Reddy and B. Vlijmen, “Yog.ai: deep learning for yoga”. [Online]. Available: http://cs230.stanford.edu/projects_winter_2019/reports/15813480.pdf

[19] M. Dantone, J. Gall, C. Leistner, “Human pose estimation using body parts dependent joint regressors”, *Proc. IEEE Conf. Computer Vision Pattern Recogn.*, 2013.

[20] A. Mohanty, A. Ahmed, T. Goswami, "Robust pose recognition using deep learning", *Adv. in Intelligent Syst. and Comput,* Singapore, pp 93-105, 2017.

[21] P. Szczuko, "Deep neural networks for human pose estimation from a very low resolution depth image*", Multimedia Tools and Appl*, 2019.

[22] M. Chen, M. Low, "Recurrent human pose estimation", [Online]. Available: https://web.stanford.edu/class/cs231a/prev_projects_2016/final%20(1).pdf

[23] K. Pothanaicker, "Human action recognition using CNN and LSTM-RNN with attention model", *Intl Journal of Innovative Tech. and Exploring Eng,* 2019.

[24] N. Nordsborg, H. Espinosa, "Estimating energy expenditure during front crawl swimming using accelerometrics", *Procedia Eng*., 2014.

[25] P. Pai, L. Changliao, K. Lin, "Analyzing basketball games by support vector machines with decision tree model", *Neural Comput. Appl.,* 2017.

[26] S. Patil, A. Pawar, A. Peshave, "Yoga tutor: visualization and analysis using SURF algorithm", *Proc. IEEE Control Syst. Grad. Research Colloquium,* 2011.

[27] W. Wu, W. Yin, F. Guo, "Learning and self-instruction expert system for yoga", *Proc. Intl. Work Intelligent Syst. Appl,* 2010.

[28] E. Trejo, P. Yuan, "Recognition of yoga poses through an interactive system with kinect device", *Intl. Conf. Robotics and Automation Science,* 2018.

[29] H. Chen, Y. He, C. Chou, "Computer assisted self-training system for sports exercise using kinetics", *IEEE Intl. Conf. Multimedia and Expo Work,* 2013.

[30] Dataset [Online]. Available: https://archive.org/details/YogaVidCollected.

[31] Y. Shavit, R. Ferens, "Introduction to camera pose estimation with deep learning", [Online]. Available: https://arxiv.org/pdf/1907.05272.pdf.