Project Team

MohiT SHaRMA 224161018

NIKHIL THOSAR 224161019

MAYANK BANSAL 224161006

NEELMANI 224161019

HARSHIT SHARMA 224161016

**github link:** https://github.com/nikhilthosar/Yoga-Pose-Detection-IPML-Project-

**Course Instructor: Dr. Debang Raj Neog**

**Indian Institute of Technology Guwahati**

Yoga Pose Detection

Using Classical Machine Learning and Deep Learning

gi

**INDEX PAGE**

**Problem Statement:**

The purpose of this project is to explore the use of machine learning for yoga pose classification. Given an input image I in which some yoga pose is performed, the objective is to learn a classifier which can detect the pose performed in I. This task comes under the general human activity recognition problem.

Activity Recognition is the process of continuously monitoring a person’s activity and movement. Recognizing human activity through computer vision has been an area of prime interest among researchers over several years. 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. Yoga is a form of exercise which is helpful in reducing mental stress and keeping a person physically and mentally fit. It also improves concentration, focus, calmness, and blood circulation. Yoga originated in ancient India and has now been adopted by countries all over the world. 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.

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. Our motivation with this project is to be part of the bridge that covers this gap. We want to help the end user to correct their form and perform the Yoga poses correctly for maximal benefit. Our Multi-Class Classification model can help Yoga practitioners to get the details of Yoga Pose that they are doing and as the project evolves in the futures stages with Video Input, we can modify the model to provide Form Correcting cues.

**Dataset:** The dataset comprises of 2113 images, 1708 training images and 405 test images. It contains images of 6 different yoga poses, namely Downdog, Plank, Tree, Goddess, Warrior2 as illustrated in the figure below.

(a) (b) (c)

****

(d) (e)

Figure: Yoga Poses (a) Downdog, (b) Plank, (c) Goddess, (d) Tree, (e) Warrior2

|  |  |  |  |
| --- | --- | --- | --- |
| No. | Yoga Pose | Number of images in Training set | Number of images in Test set |
| 1 | Downdog |  |  |
| 2 | Plank |  |  |
| 3 | Goddess |  |  |
| 4 | Tree |  |  |
| 5 | Warrior2 |  |  |

**Related Works:**

Although a lot of research has been done in human activity recognition, but its application in detecting yoga posture is considered a relatively new and under-researched field. Here we mention two related works which worked on classifying yoga pose.

**1. Deep Learning Models for Yoga Pose Monitoring**

Authors: Debabrata Swain, Santosh Satapathy, Biswaranjan Acharya, Madhu Shukla, V Gerogiannis, A Kanavos, D Giakovis

In this work, authors have proposed an approach for the efficient detection and recognition of various yoga poses. Their dataset consists of 85 videos with 6 yoga postures performed by 15 participants. Their Proposed system takes video sequence frames as input in real time. The output would be the predicted yoga pose along with the corresponding feedback in terms of both angle and pose prediction. Their system consists of three main phases, namely, keypoint extraction, pose prediction, and pose correction. The keypoints of the users are extracted using the MediaPipe library in python. For pose prediction, a combination of Convolution Neural Network (CNN) and Long Short-Term Memory (LSTM) has been employed for yoga pose recognition through real-time monitored videos as a deep-learning model. Specifically, the CNN layer is used for the extraction of the features from the keypoints and the following LSTM layer understands the occurrence of sequence of frames for prediction.

Their final phase is pose correction where user is further given feedback for the correction of the pose and depicts also the similarity percentage (using cosine similarity) compared to the actual pose. For all the six yoga poses present in their dataset, important and critical angles are identified and rules are formulated for each pose. For each rule, a threshold is set, which constitutes the maximum deviation allowed for the user from the standard pose. If the user exceeds this threshold value, a corresponding feedback is given accordingly in the form of text and speech.

After running the model for 50 epochs, they obtained 99.53% test accuracy.

**2. Implementation of Machine Learning Technique for Identification of Yoga Poses**

Authors: Y. Agarwal, Y. Shah, and A. Sharma

The authors implemented Traditional Machine Learning Techniques. They implemented tf-pose-estimation, Decision Tree, Random Forest, Support Vector Machines, Naïve Bayes, Logistic Regression, K-Nearest Neighbours. They created their own dataset, named it Yogi Dataset. The dataset consisted of at least 5500 images of 10 different yoga poses. The tf-pose skeletal is often used to extract angle from human joints, that are then used as a feature in ML models. All machine learning models achieved an overall accuracy of 94.28 percent. They achieved the best accuracy with the Random Forest Classifier where they achieved an accuracy of 99.04%.​

**Methods:**

The main points in solving the problem are as follows:

1. Data Augmentation and Preprocessing
2. Keypoint extraction using MediaPipe
3. Pose Prediction

Now we will see all these steps in details.

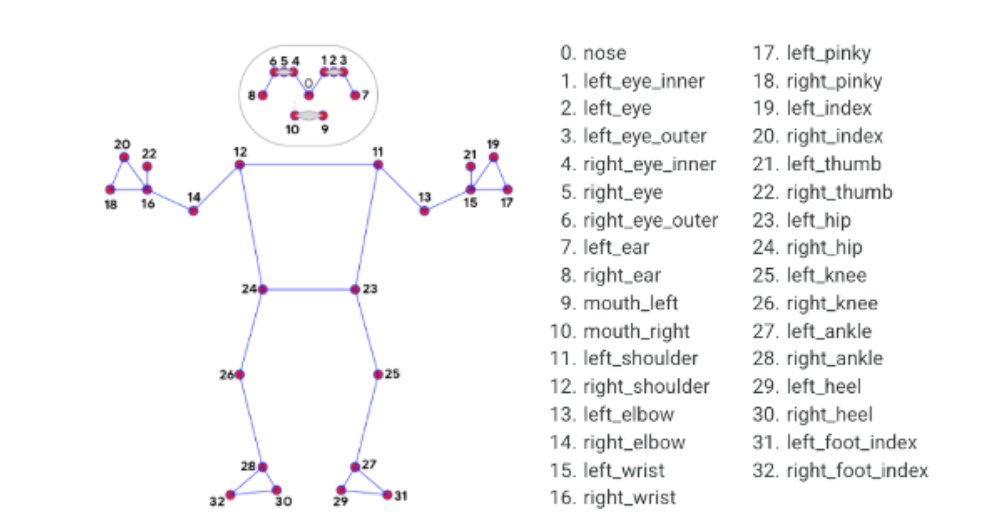
1. ***Data Augmentation and Preprocessing:***

Deep Learning techniques require a lot of data to train and perform better. Performance of deep learning models increase with increase in data. Insufficient data is one of the main challenges in deep learning. The dataset which we used here doesn’t contain that much images which can be used to train a deep learning model with better performance. Hence, we used data augmentation techniques to increase our training dataset size. The increase in dataset resulted in significant improvement in the deep learning model’s performance. Data augmentation is a set of techniques to artificially increase the amount of data by generating new data points from existing data. We used the following data augmentation techniques in our implementation, rotation, zooming, horizontal flip, and shearing. We did not use vertical flip as it is not suitable in our case as it can lead to different posture and hence can confuse the model while prediction.

The initial images varied in their shapes, sizes, and formats. However, in order to accommodate Convolutional Neural Networks (CNN), which require inputs of a consistent size, we have standardized the images by reshaping them to dimensions of 256\*256. Additionally, to simplify the input data, the images have been converted to grayscale.

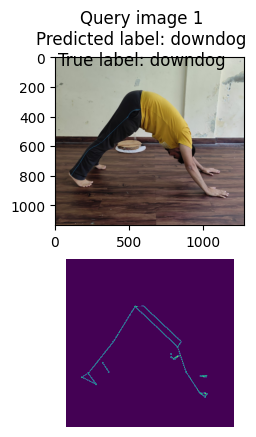
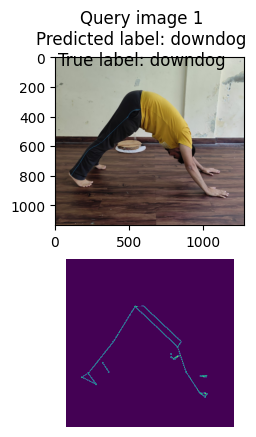
1. ***Keypoint Extraction using MediaPipe***

Key points are essential elements in a person's body that play a crucial role in creating a yoga pose. These points include the shoulders, elbows, wrists, knees, and more. To extract these key points, we utilized the MediaPipe library. Developed by Google, MediaPipe is a versatile cross-platform library that offers pre-built machine learning solutions specifically designed for computer vision tasks.

****The pose estimator in MediaPipe library uses a highly-optimized pre-trained Convolutional Neural Network for body pose tracking, it infers 33 3D landmarks on the whole body from RGB video frames/ images. MediaPipe library generates 3 coordinates (*X*, *Y* and *Z*), where *Z* indicates the depth of a 2D coordinate. The following figure represents the 33 keypoints provided by MediaPipe library.

**Source of the image :** <https://github.com/google/mediapipe/blob/master/docs/solutions/pose.md.>

In our methodology, for each image we extract the keypoints using MediaPipe and we draw those keypoints on an blank image, which is given as an input to model 1 and model 2, mentioned below. The following image shows the effect of drawing the keypoints on the blank image.



The motivation behind drawing the keypoints on an blank image is to discard noise from the original image and to concentrate only on the form of the human performing the pose. This technique improved model’s performance significantly. Previously when we drew the keypoints on the original image itself, due to a lot of noise in the images, the model performance was not satisfactory.

1. ***Pose Prediction***

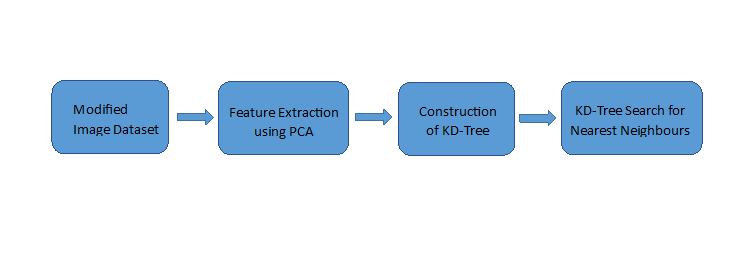
For pose prediction we used three models, one uses traditional Machine Learning Algorithms, and the other two are based on deep learning architecture.

**Model 1:** Using Classical ML Algorithms

Modified input image: We used the new modified images as input data where a skeletal figure corresponding to the pose landmarks and keypoints was drawn on a black background.

Principal Component Analysis (PCA) was implemented to reduce the dimensionality of the data while capturing the most significant variance in the dataset. By applying PCA, the original high-dimensional data image was transformed into a lower-dimensional representation while retaining as much information as possible. In this case, PCA was used to reduce the dimensionality of the data from 65536 to 200 principal components.

A KD-Tree was constructed from the reduced training set. KD-Tree is a data structure that partitions the feature space to enable efficient nearest neighbor searches. The leaf size parameter was set to 20, which determines the number of points required to form a leaf node in the KD-Tree.

KD-Tree Search for Nearest Neighbors: To classify a given test image, KD-Tree search was performed to find the four nearest neighbors in the training set. The distances and indices of the nearest neighbors were obtained from the KD-Tree search.

The trained model using the KD-Tree and PCA was evaluated on the test dataset. The accuracy of the model on the test dataset was measured and the obtained accuracy was 85%.

**Model 2:**

The selection of the specific CNN architecture was based on a careful evaluation of performance, complexity, and generalization ability for the yoga pose classification task without overfitting or underfitting the training data.

**Data Preparation and Preprocessing**

To prepare the input for training and testing, we employed a preprocessing step. For each original image, we utilized the ‘MediaPipe’ library to extract 33 key points representing the pose landmarks of the human figure. With these key points, we created a new image where a skeletal figure corresponding to the pose landmarks was drawn on a black background. This new image retained all the essential features of the original image.

Initially, we experimented with a CNN architecture consisting of three convolutional layers. However, after thorough evaluation, we decided to enhance the model's capacity by increasing the number of convolutional layers to five. This modification allowed the model to learn more complex and abstract features from the yoga pose images.

In the revised architecture, the number of filters gradually increased in the deeper layers, starting from 32 filters in the first layer and progressively growing to 64, 128, 256, and 512 filters in subsequent layers. This design choice was made to capture hierarchical features at different levels of abstraction, enabling the model to extract more intricate representations of the yoga poses.

**Model Architecture:**

(256,256,1) INPUT​

(254, 254, 32) Conv2D: 32 filters of size (3, 3)​

(127, 127, 32) MaxPooling2D: pool size (2, 2)​

(125, 125, 64) Conv2D: 64 filters of size (3, 3)​

(62, 62, 64) MaxPooling2D: pool size (2, 2)​

(60, 60, 128) Conv2D: 128 filters of size (3, 3)​

(30, 30, 128) MaxPooling2D: pool size (2, 2)​

(28, 28, 256) Conv2D: 256 filters of size (3, 3)​

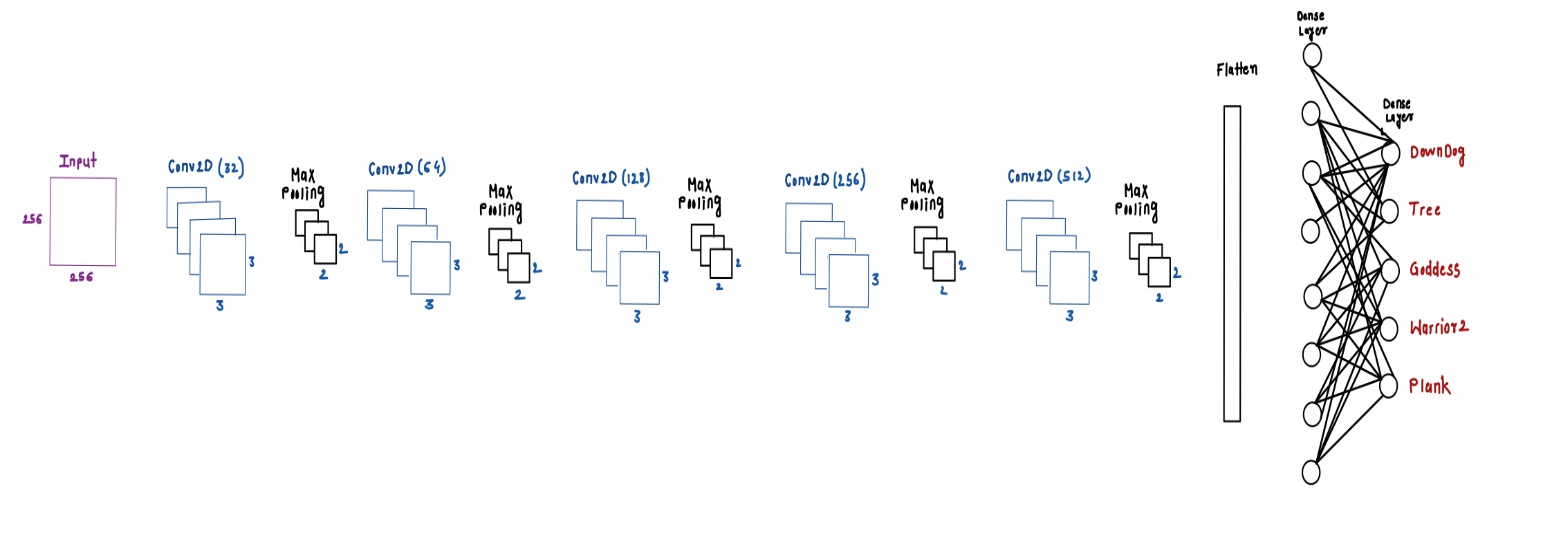
(14,14,256) MaxPooling2D:  pool size (2, 2)​

(12,12,512) Conv2D: 512 filters of size (3, 3)​

(6,6,512) MaxPooling2D:  pool size (2, 2)​

(None, 18432) Flatten layer​

(None, 128) Dense layer with 128 units​

****Dense layer with 5 classes (number of yoga pose categories),activation = softmax.

**Model Training Strategy**

We compiled the model with the Adam optimizer, which adapts the learning rate during training and used the sparse categorical cross-entropy as our loss function which is suitable for multi-class classification. Finally, we chose accuracy as the evaluation metric to monitor model performance.

We trained the model using the Training Data (Skeletal Images) and Labels for 20 epochs. We also made use of a validation split of 20% to evaluate the model's performance on unseen data during training. Finally, monitoring the training and validation loss and accuracy to assess model performance and identify overfitting.

We tracked the training and validation loss and accuracy during training. If the training loss continues to decrease while the validation loss starts to increase or plateau, it indicates overfitting. Similarly, if the training accuracy keeps improving while the validation accuracy stagnates or decreases, overfitting may be occurring. To avoid overfitting, we used Early Stopping technique in our model training.

**Model 3:**

For this model, we take ideas from the paper [1] mentioned in References.

The proposed model takes input the keypoints extracted from MediaPipe, for an image this is an array of shape (33,2), as there are 33 keypoints with their x and y coordinates. The keypoints extraction phase aims at detecting the location of important keypoint based on user’s position.

The first layer used in the model is a CNN layer with 16 Conv1D layers with window size 3, the activation function used in this layer is ReLU.

The second layer used is a Batch Normalization layer which can solve the problem of internal covariance shift. It also makes easier the flow of data through different layers.

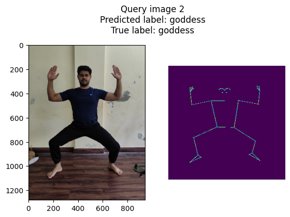
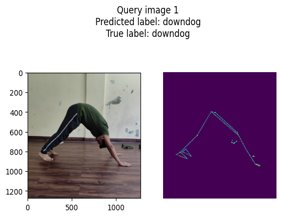
The next layer is dropout layer to prevent overfitting, the rate of drop out is 0.5. After that another Batch Normalization layer is used. The output from this layer is then flattened to get a one-dimensional array. The final layer is a linear layer which uses softmax as activation function, from which we predict the class label.

**Add model image here**

The loss function used is categorical cross-entropy. We trained the model for 50 Epochs. The optimizer is Adam Optimizer and the learning rate is 0.001.

In model 2 and model 3, after each epoch, we checked whether the loss has been decreased and became lesser than the least loss obtained. If it is lesser than the least loss, then the least loss is replaced by the current loss and the parameters of the current epoch are then saved. At the end, the parameters which gave us the least loss were loaded in the model.

**Experiment and Results:**

****

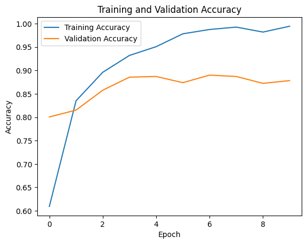
***Model Loss and Accuracy:***

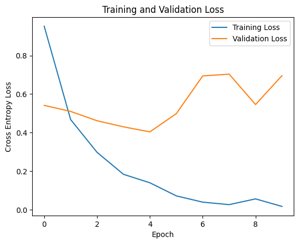
***Model II Accuracy Plot:***

We can see the accuracy increasing gradually in epochs for training and validation.

***Model II Loss Plot:***

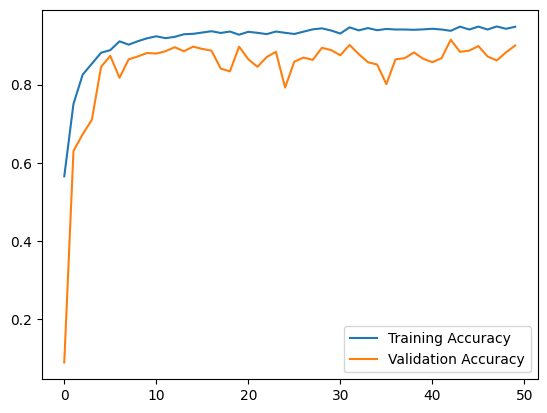
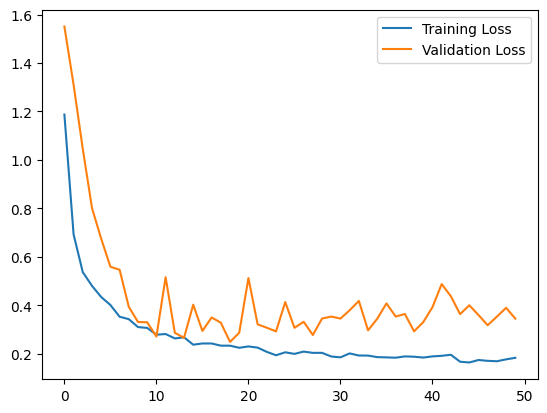
We can see the training loss decreasing exponentially, the validation loss goes down for three-four epochs and then validation loss goes up as model starts overfitting after certain number of epochs.

******

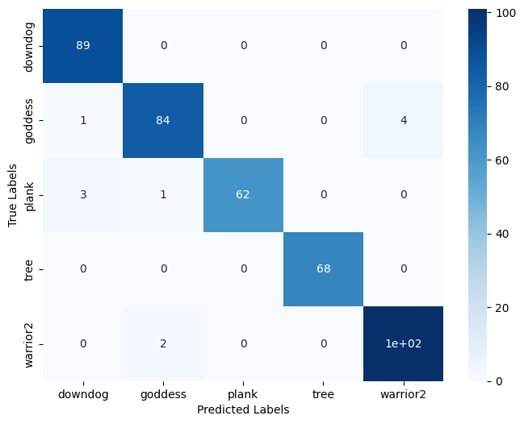
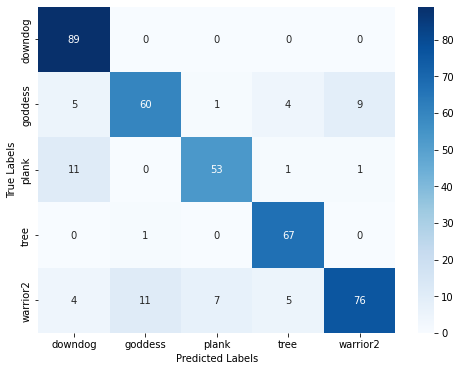
******

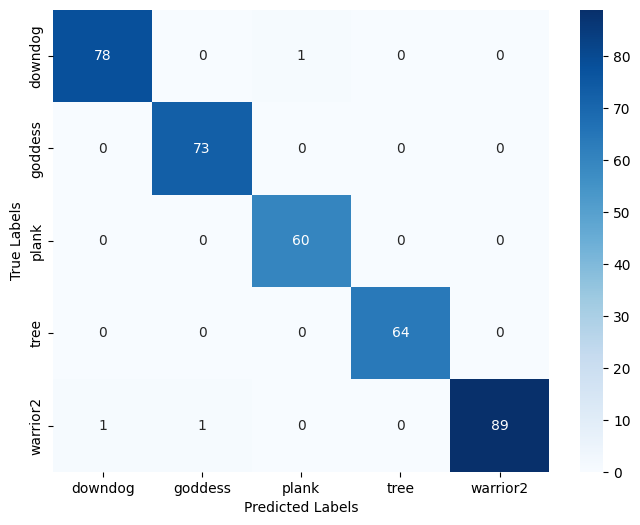
***Model III Accuracy Plot:***

***Model III Loss Plot:***

******

***Confusion Matrix:***

***Model I Confusion Matrix: Model II Confusion Matrix:***

***Model III Confusion Matrix:***

**Classification Report:**

We have used the following evaluation metrics:

**Precision:** Precision is the ratio of true positives (TP) to the sum of true positives and false positives (FP

**Precision = TP / (TP + FP)**

**Recall:** Recall, also known as sensitivity or true positive rate (TPR), is the ratio of true positives to the sum of true positives and false negatives (FN).

**Recall = TP / (TP + FN)**

**F1-score:** The F1-score is the harmonic mean of precision and recall. It provides a single metric that balances precision and recall.

**F1-score = 2 \* (Precision \* Recall) / (Precision + Recall)**

***Model 1 Report:***

|  |  |  |  |
| --- | --- | --- | --- |
| ***Pose*** | ***Precision*** | ***Recall*** | ***F1-Score*** |
| *Downdog* | 0.82 | 1 | 0.90 |
| *Plank* | 0.87 | 0.8 | 0.83 |
| *Goddess* | 0.83 | 0.76 | 0.79 |
| *Tree* | 0.87 | 0.99 | 0.92 |
| *Warrior2* | 0.88 | 0.74 | 0.80 |

***Model 2 Report:***

|  |  |  |  |
| --- | --- | --- | --- |
| ***Pose*** | ***Precision*** | ***Recall*** | ***F1-Score*** |
| Downdog | 0.96 | 1 | 0.98 |
| Plank | 1 | 0.94 | 0.97 |
| Goddess | 0.97 | 0.94 | 0.95 |
| Tree | 1 | 1 | 1 |
| Warrior2 | 0.96 | 0.98 | 0.97 |

***Model 3 Report:***

|  |  |  |  |
| --- | --- | --- | --- |
| ***Pose*** | ***Precision*** | ***Recall*** | ***F1-Score*** |
| Down dog | 0.99 | 0.99 | 0.99 |
| Plank | 0.98 | 1 | 0.99 |
| Goddess | 0.99 | 1 | 0.99 |
| Tree | 1 | 1 | 1 |
| Warrior2 | 1 | 0.98 | 0.99 |

***Accuracy and F1-Scores on the test data of the models:***

|  |  |  |
| --- | --- | --- |
| Model | Accuracy | Macro F1 |
| Model 1 | 0.82 |  |
| Model 2 | 0.97 | 0.97 |
| **Model 3** | ***0.99*** | ***0.99*** |

**Conclusion:**

In this project, we explored various machine learning and deep learning techniques to learn a classifier which can classify yoga pose performed in an Image. We used various data augmentation techniques which helped the models perform better and more accurately. We explored MediaPipe library to identify keypoints of the person, which served as the main features for our models. We used Traditional ML algorithms, like K-Nearest Neighbour, KD tree and deep learning models with CNN layers. Best accuracy achieved is 99% by Model 3, which is implemented from the Research Paper [1] mentioned in References.

The above models especially Model 3 can be generalized to accept video as inputs, and as done in [1] we can generalize the model so that it can also give feedback in real time.

**References:**

1. D. Swain and S. Satapathy et al., "Deep Learning Models for Yoga Pose Monitoring ​  
   ",Algorithms, vol. 15, issue 11, 2022. [Algorithms | Free Full-Text | Deep Learning Models for Yoga Pose Monitoring (mdpi.com)](https://www.mdpi.com/1999-4893/15/11/403)​
2. Y. Agrawal, Y. Shah and A. Sharma, "Implementation of Machine Learning Technique for Identification of Yoga Poses", *2020 IEEE 9th International Conference on Communication Systems and Network Technologies (CSNT)*, pp. 40-43, 2020.