

# YOG AI

**ENROLL. No. - 9920102018, 9920102019, 9920102052**

**NAME OF STUDENT - KOMAL KUMARI, SHIVAM MUKHERJEE, VAISHNAVI DUBEY**

**NAME OF SUPERVISOR - DR. ATUL KUMAR**



**NOVEMBER - 2023**

Submitted in partial fulfillment of the Degree of  
Bachelor of Technology

**DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING JAYPEE  
INSTITUTE OF INFORMATION TECHNOLOGY, NOIDA**

# **CERTIFICATE**

This is to certify that the major project report entitled, “**YOG AI**” submitted by **Komal Kumari, Shivam Mukherjee, Vaishnavi Dubey** in partial fulfillment of the requirements for the award of Bachelor of Technology Degree in Electronics and Communication Engineering of the Jaypee Institute of Information Technology, Noida is an authentic work carried out by them under my supervision and guidance. The matter embodied in this report is original and has not been submitted for the award of any other degree.

**Signature of Supervisor:**

**Name of the Supervisor : Dr. Atul Kumar**

**ECE Department,**

**JIIT, Sec - 128,**

**Noida - 201304**

**Dated: 30th Nov 2023**

# DECLARATION

We hereby declare that this written submission represents our own ideas in our own words and where others' ideas or words have been included, have been adequately cited and referenced the original sources. We also declare that we have adhered to all principles of academic honesty and integrity and have not misrepresented or fabricated or falsified any idea/data/fact/source in our submission.

**Place: JIIT, Sec-128, Noida-201304**

**Date: 30th Nov 2023**

**Name: Komal Kumari**

**Enrollment: 9920102018**

**Name: Shivam Mukherjee**

**Enrollment: 9920102019**

**Name: Vaishnavi Dubey**

**Enrollment: 9920102052**

## ACKNOWLEDGEMENT

The completion of any interdisciplinary project demands upon cooperation, coordination and combined efforts of several sources of knowledge. We are grateful to **Dr. Atul Kumar** for his willingness to give us valuable advice and direction whenever we approached him with any problem. We are thankful to him for providing us with immense guidance for this project. We would also like to thank our college authorities for giving us the opportunity to pursue our project in the field.

Finally, we would like to express our gratitude to all those people who have directly or indirectly helped us in the process and contributed toward this work.

# ABSTRACT

- A Yoga Tracker app can serve various purposes in today's context, providing users with a range of benefits. When integrating technologies like MoveNet and TensorFlow for pose detection and AI-driven features, the app can offer a more enhanced and personalized experience.
- The Yoga Tracker app serves as a powerful tool for individuals seeking a convenient and flexible way to practice yoga. Users can enjoy the freedom to engage in yoga sessions at any time, tailoring their practice to suit their preferences. The app offers a diverse range of yoga types, allowing users to explore and discover the style that resonates best with them. Beyond the physical benefits of yoga, the app promotes relaxation and overall fitness.
- The decision to invest in a Yoga Tracker app is substantiated by the surging popularity and high demand for yoga-related services. Google trends reflect a substantial increase in global searches for yoga apps, indicating a growing interest in practicing yoga through interactive online sessions. The potential for profitability in this venture is underscored by the remarkable statistics in the yoga industry:
  - Revenue Growth: In 2020, the yoga industry's revenue reached an impressive \$11.5 billion, with the global yoga market estimated to be valued at \$80 billion.
  - Massive User Base: The number of yoga practitioners worldwide has surpassed 300 million, reflecting the widespread adoption of yoga as a wellness practice.
- This data paints a compelling picture of the yoga industry's vibrancy and growth potential. With the increasing trend of individuals seeking online yoga experiences, a Yoga Tracker app stands as a lucrative opportunity to tap into this thriving market. The app not only aligns with the contemporary lifestyle of fitness enthusiasts but also addresses the evolving preferences of users who prioritize holistic well-being.

# TABLE OF CONTENTS

1. Certificate .....	ii
2. Declaration .....	iii
3. Acknowledgement .....	iv
4. Abstract .....	v
5. List of Tables and Figures .....	vii
6. Chapter 1: Introduction .....	1
1.1 Scope & Objectives .....	1
1.2 Problem Statement .....	2
1.3 Solution .....	3
7. CHAPTER 2: Literature Survey .....	4
2.1 Research Paper 1 .....	4
2.2 Research Paper 2 .....	4
2.3 Research Paper 3 .....	5
2.4 Research Paper 4 .....	5
8. CHAPTER 3: Methodology & Background Study.....	7
9. CHAPTER 4 : Experimental Analysis .....	12
10. CONCLUSION .....	16
11. FUTURE SCOPE .....	17
12. REFERENCES .....	18
13. APPENDICES .....	19

# LIST OF TABLES AND FIGURES

1. Fig 3.1 (Neural Network and Adding a dropout layer to the neural network) .....	9
2. Fig 3.2 (Dropout used to prevent overfitting with activation function as softmax).....	10
3. Table 3.1 (Difference between other Algorithms).....	11
4. Fig 4.1 (Entering the application).....	12
5. Fig 4.2 (Pose tracker app working testing and can be tested with different poses).....	13
6. Fig 4.3 (Some other pose detection and results).....	13
7. Fig 4.4 (MoveNet Algorithm Process ).....	14
8. Fig 4.5 (After model training and validation it was tested with the images of similar asana the accuracy achieved was 99.7% algorithm).....	15

# CHAPTER - 1

## INTRODUCTION

Welcome to the **YOG AI** project, where technology and ancient wisdom converge to revolutionize the way we approach yoga practice. In the era of interconnectedness and self-care, our project stands as a beacon of innovation, designed to enhance the yogic journey for enthusiasts of all levels.

### 1.1 SCOPE & OBJECTIVES

- 1. Accurate Pose Tracking with CNN:** Develop a mobile app that employs Convolutional Neural Networks (CNN) to accurately recognize and classify users' yoga poses in real time using data from the device's camera.
- 2. User-Friendly Interface:** Design an intuitive user interface that allows users to easily initiate pose tracking and view real-time feedback on their yoga alignment and form.
- 3. Real-Time Feedback with k-NN:** Implement k-Nearest Neighbors (k-NN) algorithm to provide instant feedback on users' pose alignment, comparing their poses with a database of correct poses.
- 4. Pose Dataset Integration:** Integrate a diverse dataset of annotated yoga pose images into the CNN model, ensuring it's trained to recognize a wide range of poses accurately.
- 5. Personalized Pose Suggestions:** Develop an algorithm that combines CNN's output with user-specific flexibility data to suggest personalized pose adjustments using k-NN's pattern recognition capabilities.
- 6. Pose Library with Visual References:** Create a comprehensive pose library with textual descriptions and visual references. Users can compare their poses to these references for self-assessment.
- 7. Educational Content Integration:** Include educational content on yoga philosophy, breathing techniques, and wellness to enhance users' understanding of the practice.



**8. Calorie Tracking Integration:** Integrate a calorie tracking feature that allows users to input their activity and receive estimates of calories burned during their yoga practice.

## **1.2 PROBLEM STATEMENT**

Design and develop a Yoga Posture Tracker App that utilizes machine learning algorithms, including Convolutional Neural Networks (CNN) and k-Nearest Neighbors (k-NN), to accurately track and analyze users' yoga postures. The app aims to provide immediate visual feedback to users, assisting them in achieving correct alignment and posture during their yoga practice. In addition to posture tracking, the app will incorporate calorie tracking features to provide a comprehensive health and wellness solution.

### **Key Objectives:**

- Pose Recognition
- Real-Time Feedback
- Intuitive user Interface
- Personalized Guidance
- Progress Tracking
- Accuracy and Performance
- Algorithm Integration
- Calorie Tracking Integration

### **Benefits:**

- Injury Prevention
- Health and Wellness
- User Engagement
- Education for beginners
- Convenience

## **1.3 SOLUTION**

Our vision is simple yet profound - to create a seamless fusion of traditional yoga principles and state -of-the-art technology.

- The YOG AI aims to provide practitioners with a comprehensive platform that empowers them to elevate their practice, cultivate mindfulness, and achieve holistic well-being.
- The YOG AI Project aims to bridge the gap between the ancient art of yoga and the demands of contemporary life, making this transformative practice more accessible, engaging, and effective.
- Our goal is to create a transformative platform that enhances the yoga experience, empowers practitioners, and fosters a deeper connection between mind, body, and spirit.

## **CHAPTER - 2**

### **LITERATURE SURVEY**

#### **O. Tarek, O. Magdy and A. Atia, "Yoga Trainer for Beginners Via Machine Learning," [1]**

- Highlights the growing popularity of remote Yoga practice due to technological advancements and increased demand for professional Yoga instructors.
- Proposes a system using machine learning techniques, featuring an Artificial Neural Network (ANN) model and human pose tracking. The goal is to classify Yoga Hatha movements, detect incorrect poses, and offer real-time feedback to enhance the learning experience.
- Reports a testing accuracy of 82.2% for the proposed model. Additionally, the system successfully reduces the average practice time by 6.4 seconds, based on tests conducted with 20 participants of diverse body features.
- Our Take: Propose upgrading to MoveNet and incorporating CNN for accuracy over 95%. Prioritize user experience, efficiency, and versatility. Emphasize rigorous testing, validation, and a feedback mechanism for improvement.

#### **J. Sunney, M. Jilani, P. Pathak and P. Styne, "A Real-time Machine Learning Framework for Smart Home-based Yoga Teaching System," [2]**

- Proposed a real-time machine learning framework combining pose estimation, classification, and feedback.
- Dataset includes popular yoga poses: downdog, tree, goddess, plank, and warrior.
- BlazePose used for pose estimation, transforming image data into 3D landmark points.
- Evaluated machine learning classifiers; XGBoost demonstrated superiority with 95.14% accuracy, 8 ms latency, and 513 KB size.
- XGBoost output used for real-time feedback to correct yoga poses.
- Framework has potential for integration into mobile applications, enabling unsupervised yoga practice at home.

- Our take: Will be using MoveNet, a powerful model excels in suppressing noise and outliers, making it suitable for quick motions, with less latency of 7.1 ms to achieve target and suitable for projects with not much redundant factors.

**A. Chaudhari, O. Dalvi, O. Ramade and D. Ambawade, "Yog-Guru: Real-Time Yoga Pose Correction System Using Deep Learning Methods," [3]**

- A dataset comprising five yoga poses (Natarajasana, Trikonasana, Vrikshasana, Virabhadrasana 1 & 2, and Utkatasana) is compiled from Internet images and contributions by individuals involved in system development.
- The system employs a deep learning model, utilizing Convolutional Neural Networks (CNN) for yoga pose identification and a human joints localization model for error identification.
- Achieves a commendable classification accuracy of 95% for accurate pose identification.
- After gathering user pose information, the system provides feedback for posture improvement or correction.
- Overall, the system contributes to promoting correct yoga practices, reducing the risk of injuries, and enhancing users' understanding of specific yoga poses.

**Valentin Bazarewsky, Ivan Grishchenko, Karthik Raveendran, "BlazePose: On-device Real-time Body Pose tracking,"[4]**

- The paper presents a comprehensive study about pose detection based on the use of blaze pose. BlazePose, a lightweight convolutional neural network architecture for human pose estimation that is tailored for real-time inference on mobile devices
- The following uses 33 key points over a body of a single individual and takes in 30 frames per second to give an accurate description. Calculates distance from the torso to various body parts and calculates change according to it.
- It focuses on detecting the bounding box of a relatively rigid body part like the human face or torso.
- This face detector predicts additional person specific alignment parameters: the middle point between the person's hips, the size of the circle circumscribing the

whole person, and incline (the angle between the lines connecting the two mid-shoulder and mid-hip points).

- In addition, it also used the coco technology with 17 key points and did detection based on it.
- As an evaluation metric, we use the Percent of Correct Points with 20% tolerance (PCK@0.2) (where we assume the point to be detected correctly if the 2D Euclidean error is smaller than 20% of the corresponding person's torso size).
- It obtained an average PCK@0.2 of 97.2. We trained two models with different capacities: BlazePose Full (6.9 MFlop, 3.5M Params) and BlazePose Lite (2.7 MFlop, 1.3M Params).

## CHAPTER - 3

### METHODOLOGY & BACKGROUND STUDY

#### 3.1 Background Study

- **Challenges Addressed:** Incorrect postures during yoga practice can lead to discomfort and injuries. This app addresses the challenge of ensuring users perform yoga poses correctly and safely, thus promoting injury prevention.
- **Health and Wellness Focus:** With the growing emphasis on health and wellness, people are increasingly turning to activities like yoga for physical and mental well-being. An app that helps users accurately track their yoga postures can promote healthy habits and encourage regular practice.
- **Personalized Guidance:** The app can provide users with personalized guidance by analyzing their posture and suggesting adjustments in real-time. This helps users achieve correct alignment, reducing the risk of injuries and enhancing the effectiveness of their yoga practice.
- **Remote Learning:** Many yoga studios and fitness centers have shifted to offering virtual classes. A posture tracking app can complement these virtual classes by providing students with visual feedback, making remote learning more interactive and engaging.
- **Motivation and Progress Tracking:** The app can track users' progress over time, showing improvements in posture and flexibility. This feature can motivate users to continue practicing and set achievable fitness goals.
- **Accessible Learning:** The app can serve as a learning tool for beginners who are new to yoga. Visual feedback through real-time posture tracking can help beginners understand how to perform each pose correctly.

- **User Engagement:** Gamification elements can be integrated, allowing users to earn points or rewards for maintaining correct postures. This gamified approach can increase user engagement and make the fitness journey more enjoyable. For training a neural network to detect poses from a dataset of pose images.

## 3.2 Methodology

### 3.2.1 Dataset Preparation:

- Collect a dataset containing around 50 images for each pose you want to recognize [5].
- Annotate the images to identify key points or landmarks associated with each pose. For example, you might identify 17 key points for each pose.

### 3.2.2 Feature Vector Extraction:

- Extract features from each annotated pose image to create a feature vector. Techniques like keypoint detection or skeletonization can be used.
- Save the keypoint coordinates (17 coordinates per pose) in a CSV file, where each row corresponds to a different pose image.

### 3.2.3 Data Normalization:

- Move all keypoint coordinates so that the origin becomes the center of the image. This involves adjusting the coordinates relative to the center of the image.

### 3.2.4 Data Transformation:

Convert the CSV file into a one-dimensional tensor suitable for training your neural network. This may involve reading the CSV file and reshaping the data into the desired format.

### 3.2.5 Neural Network Training:

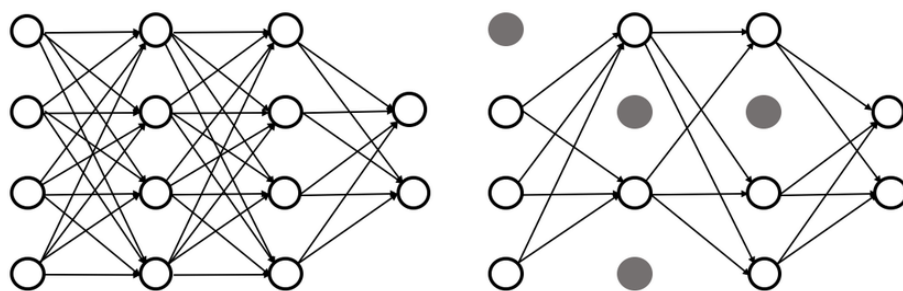
- **Architecture Design:** Design a neural network architecture suitable for pose detection. This may include convolutional layers for image processing and dense layers for classification.
- **Input Normalization:** Normalize the input data before training the neural network. Standardization or scaling can be used. Train the neural network using

the pre-processed dataset. Use appropriate loss functions, optimizers, and metrics for your pose detection task.

- Evaluate the model on a separate validation set to assess its performance and make adjustments if necessary.
- **Inference with Centered Coordinates:** During pose detection, ensure that the input coordinates are centered around the origin to allow the model to generalize well across different positions in the image. Adjust the detected coordinates based on the center of the image to obtain accurate pose predictions.
- **Additional Considerations:**
  - Consider applying data augmentation techniques to increase the diversity of your training dataset and improve the model's robustness.
  - Experiment with different hyperparameters to fine-tune the model's performance.
  - Apply regularization techniques, such as dropout, to prevent overfitting during training.

### 3.2.6 DROPOUT LAYER

Applying dropout is a regularization technique commonly used in neural networks to prevent overfitting. Overfitting occurs when a model learns the training data too well, including its noise and outliers, which can lead to poor generalization to new, unseen data. Dropout helps address this issue by randomly dropping (setting to zero) a proportion of neurons during training. All of which can be seen in fig 3.1(a)



**Fig 3.1 (a) Neural Network (b) Adding a dropout layer to the neural network**

- **Random Dropout:**
  - A dropout layer is typically added after fully connected layers in a neural network architecture. Similar pattern has been shown in fig 3.1(b)



- During training, each neuron (or node) in the dropout layer has a probability  $p$  of being "dropped out" or set to zero. This probability is a hyperparameter and is usually set between 0.2 and 0.5.
- During each forward pass of training, different neurons are randomly dropped out, introducing variability and preventing the network from relying too much on any particular subset of neurons.
- **Inference without Dropout:**
  - During inference or testing, the dropout layer is usually deactivated, and all neurons are used. This is done to make predictions without introducing randomness.
- **Regularization Effect:**
  - Dropout acts as a form of regularization. By randomly dropping out neurons, the network is forced to learn more robust features, reducing the risk of overfitting.

```
inputs = tf.keras.Input(shape=(34))
layer = keras.layers.Dense(128, activation=tf.nn.relu6)(inputs)
layer = keras.layers.Dropout(0.5)(layer)
layer = keras.layers.Dense(64, activation=tf.nn.relu6)(layer)
layer = keras.layers.Dropout(0.5)(layer)
outputs = keras.layers.Dense(len(class_names), activation="softmax")(layer)
```

**Fig 3.2 Dropout used to prevent overfitting with activation function as softmax**

**MoveNet** is designed with a focus on real-time performance, and it offers variants like "Lightning" that prioritize low latency for applications where quick response times are crucial. The Lightning variant of MoveNet is optimized for speed, making it suitable for scenarios such as fitness tracking, sports analysis, and real-time pose estimation in mobile and web applications. Some of the libraries used are listed in the figure 3.2.

While PoseNet and OpenPose are also capable of real-time performance, MoveNet has been specifically engineered to achieve low-latency inferences, making it potentially better suited for applications where minimizing processing time is a critical factor described in Table 3.1.

**Table 3.1 Difference between other Algorithms**

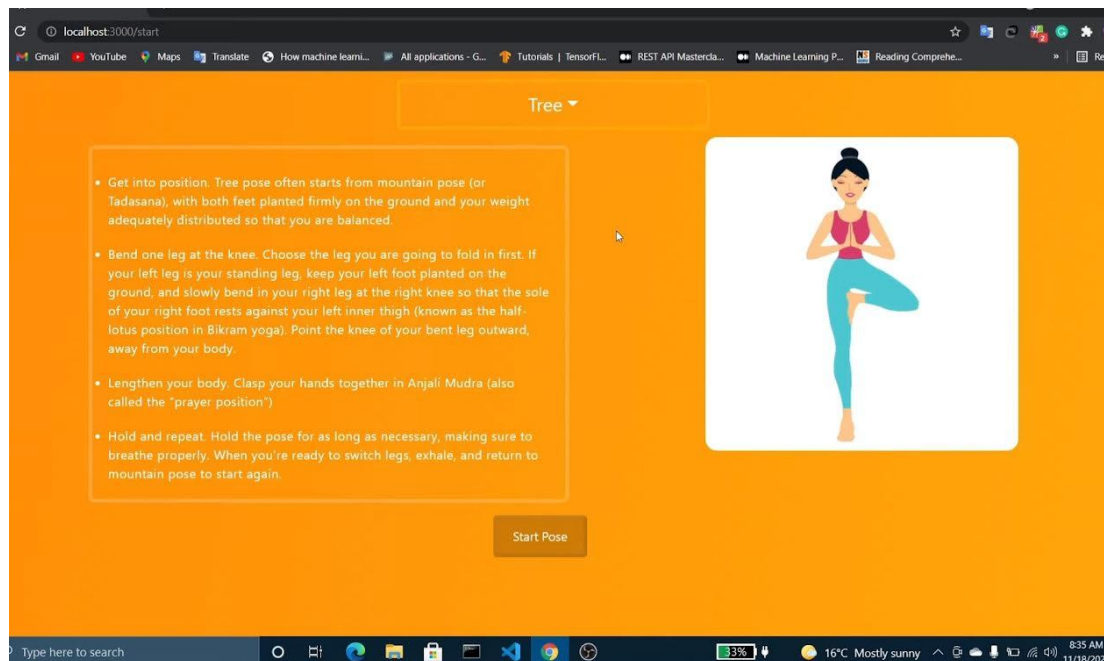
<b>Aspect</b>	<b>MoveNet</b>	<b>PoseNet</b>	<b>OpenPose</b>
<b>Key Point Detection</b>	17 Key Points	17 Key Points	Multi-keypoint detection (up to 135)
<b>Model Architecture</b>	Lightweight Mobile-Friendly Models	MobileNet architecture	Complex multi-stage architecture
<b>Real-Time Performance</b>	Yes, optimized for real-time inference	Yes, optimized for real-time inference	Yes, real-time performance
<b>Accuracy vs. Speed</b>	variants for speed (Lightning) and accuracy (Thunder)	Single model with a balance of accuracy and speed	Customizable for different trade-offs
<b>Deployment Platform</b>	TensorFlow, TensorFlow.js	TensorFlow.js, Web Browsers	C++, Python, and various bindings
<b>Use Case Focus</b>	Fitness, Sports, Health	General Pose Estimation, Web Apps	Multi-person Pose Estimation, Research

# CHAPTER - 4

## EXPERIMENTAL ANALYSIS

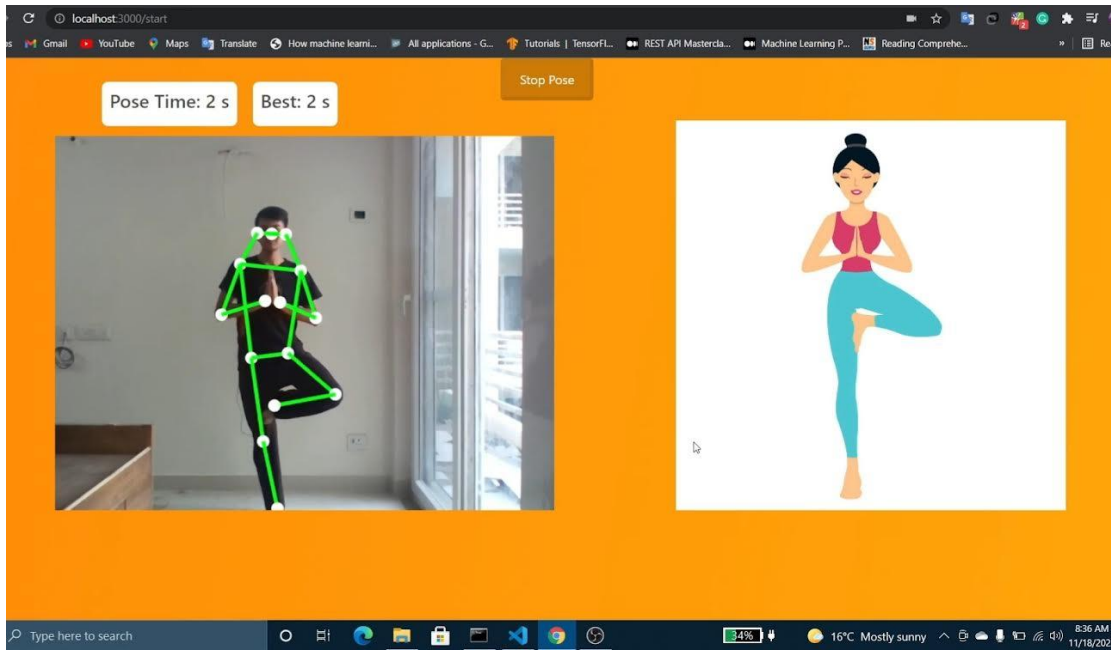
The homepage of Yog AI is built with React, a technology that helps save time for developers by allowing them to reuse components. This means they don't have to write the same code multiple times, making the development process more efficient.

Yog AI goes a step further by using convolutional neural network methods to create a model that can detect and correct your posture in real-time. This means as you use the app, it gives you feedback and guidance on how to improve your posture. What makes Yog AI user-friendly is its simple interface, making it easy for anyone to use.



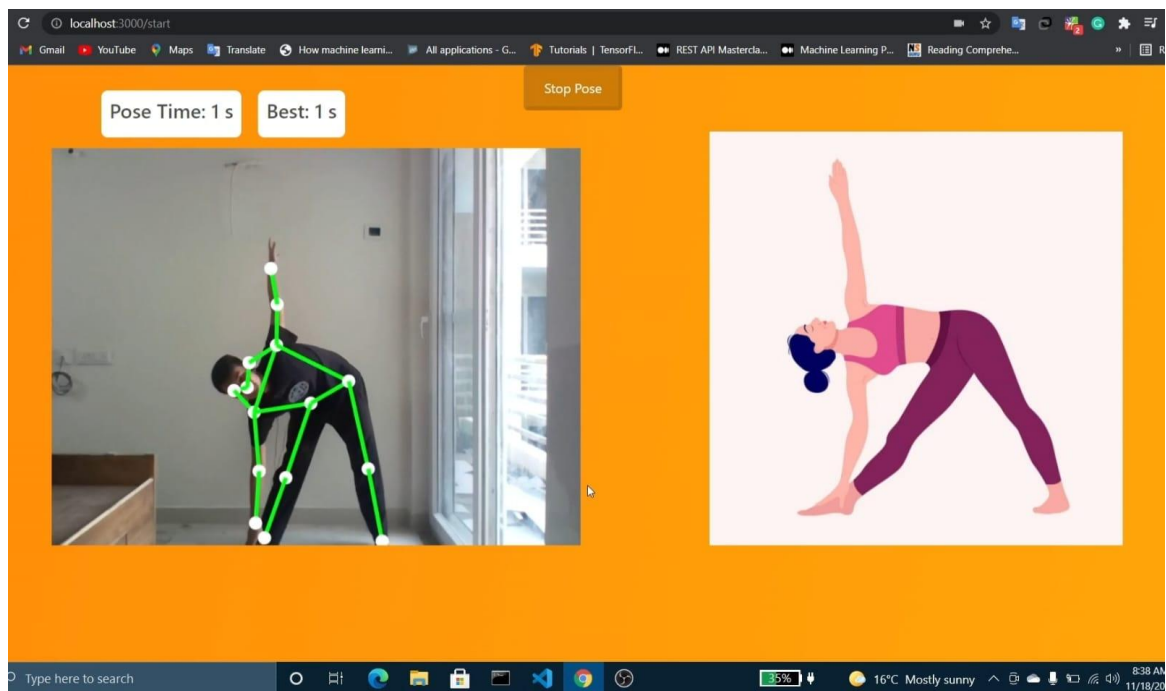
**Fig 4.1** Entering the application.

- Description about the pose and the correct way to implement it. Similar information will be portrayed for each pose before a person starts to perform it.
- In Fig 4.1 we have presented a clear representation of how to inculcate the correct body stances before starting .
- Start pose will lead to the start of the video where further evaluation of the person implementing the pose will be presented.



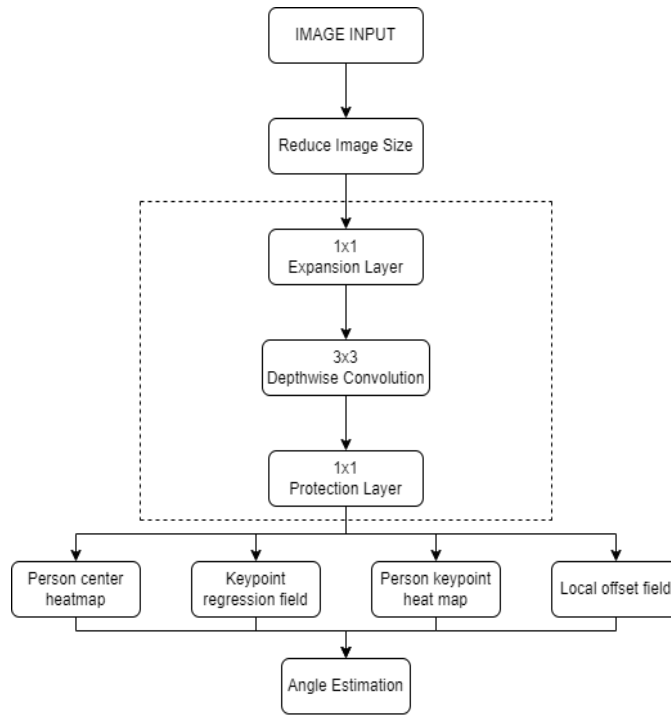
**Fig 4.2 Pose tracker app working testing and can be tested with different poses**

- In Fig 4.2 and Fig 4.3 the description is about the video representation of how the pose will be detected in the application



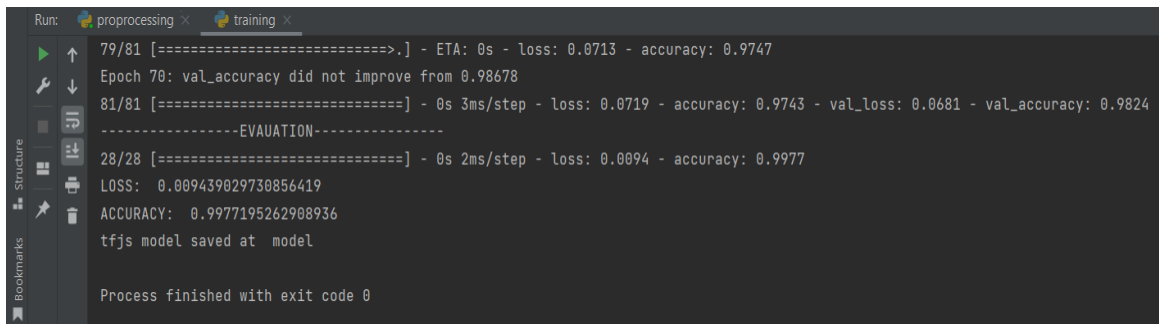
**Fig 4.3 Some other pose detection and results.**

- After correctly dimensioning it, we will get a dotted pattern which will indicate that the pose will be correct or not based by the color.



**Fig 4.4 MoveNet Algorithm Process [6]**

- As shown above in Fig 4.4 the initial step, individual video frames are extracted from the recorded audio-video interleave (AVI) file. Subsequently, these images undergo resizing to 640 x 480 pixels before being input into the TensorFlow MoveNet model.
- Utilizing the MobileNetV2 feature extractor within the MoveNet model, a high-resolution and semantically rich feature map is generated. The prediction heads then produce various outputs, including a person center heatmap (derived from estimating a person's geometric center), a keypoint regression field (predicting the full set of keypoints for a person, used for grouping key points into instances), a person keypoint heatmap (indicating the predicted location of all keypoints), and a 2D per-keypoint offset field (predicting local offsets from each output feature map).



```
Run: proprocessing x training x
79/81 [=====] - ETA: 0s - loss: 0.0713 - accuracy: 0.9747
Epoch 70: val_accuracy did not improve from 0.98678
81/81 [=====] - 0s 3ms/step - loss: 0.0719 - accuracy: 0.9743 - val_loss: 0.0681 - val_accuracy: 0.9824
-----EVALUATION-----
28/28 [=====] - 0s 2ms/step - loss: 0.0094 - accuracy: 0.9977
LOSS: 0.009439029730856419
ACCURACY: 0.9977195262908936
tfjs model saved at model

Process finished with exit code 0
```

**Fig 4.5 After model training and validation it was tested with the images of similar asana the accuracy achieved was 99.7% algorithm**

- This process ensures the efficient processing of video frames and extraction of essential information from the MoveNet model, contributing to the overall functionality of the system. The accuracy of the algorithm with the dataset was 99.7% as shown in Fig 4.5.

## CONCLUSION

The development and implementation of Yog AI, a posture tracking app, marks a significant stride in leveraging technology to enhance personal health and wellness. Through its innovative use of AI algorithms and motion tracking, Yog AI offers users a comprehensive tool to improve their posture and overall well-being.

In conclusion, Yog AI stands as a testament to the fusion of technology and health, providing users with real-time feedback and guidance to correct their posture effectively. Its user-friendly interface and accessibility make it a valuable asset for individuals seeking to maintain better posture habits in their daily lives.

The successful integration of Yog AI into the realm of health and wellness technology signifies not just a mere innovation, but a step towards empowering individuals to take control of their physical well-being. By providing personalized feedback and tailored recommendations, Yog AI not only addresses posture correction but also fosters a deeper understanding of one's body mechanics and the importance of maintaining good posture.

The ongoing advancements in technology, coupled with Yog AI's continuous improvement and adaptation, could see it becoming an integral part of various sectors, including healthcare, fitness, and ergonomics. Its ability to adapt to diverse user needs and contexts positions it as a versatile tool for a wide range of demographics, from office workers seeking ergonomic support to athletes aiming for optimal performance.

# FUTURE SCOPE

The future scope for a project like Yog AI, a posture tracking app, is vast and promising, considering its current capabilities and potential for further development. Here are several avenues that could expand its scope and impact:

**Enhanced Features:** The app could integrate additional functionalities such as personalized exercise routines, meditation sessions, or even augmented reality features for real-time posture correction guidance.

**Expanded User Base:** Beyond individual users, Yog AI could cater to broader audiences, including corporate settings for workplace ergonomics, rehabilitation centers for physical therapy, or educational institutions to promote healthy posture habits among students.

**Healthcare Integration:** Collaborations with healthcare providers could facilitate the app's integration into telemedicine platforms, enabling remote posture assessment and guidance for patients. It could also contribute to preventive healthcare measures, potentially reducing the incidence of musculoskeletal disorders.

**Data Analytics and Insights:** The accumulation of user data over time could lead to valuable insights for researchers, healthcare professionals, and fitness experts. Analyzing this data could help in understanding trends, identifying risk factors, and devising targeted interventions for posture-related issues.

**Global Accessibility:** Localization and multilingual support could make Yog AI accessible to a broader global audience. Tailoring the app to different cultural norms and lifestyles could significantly increase its adoption worldwide.

**Partnerships and Collaborations:** Collaborating with posture experts, physiotherapists, fitness influencers, or ergonomic specialists could add credibility, expertise, and varied perspectives to the app's offerings.

**Continuous Improvement:** Regular updates and improvements based on user feedback and technological advancements are crucial to maintaining relevance and staying ahead in the rapidly evolving health-tech landscape.



## REFERENCES

- [1] O. Tarek, O. Magdy and A. Atia, "Yoga Trainer for Beginners Via Machine Learning," *2021 9th International Japan-Africa Conference on Electronics, Communications, and Computations (JAC-ECC)*, Alexandria, Egypt, 2021, pp. 75-78.
- [2] J. Sunney, M. Jilani, P. Pathak and P. Stynes, "A Real-time Machine Learning Framework for Smart Home-based Yoga Teaching System," *2023 7th International Conference on Machine Vision and Information Technology (CMVIT)*, Xiamen, China, 2023, pp. 107-114.
- [3] A. Chaudhari, O. Dalvi, O. Ramade and D. Ambawade, "Yog-Guru: Real-Time Yoga Pose Correction System Using Deep Learning Methods," *2021 International Conference on Communication information and Computing Technology*, Mumbai, India, 2021, pp. 1-6.
- [4] V. Bazarewsky, I. Grishchenko, K. Raveendran, T. Zhu, F. Zhang, and M. Grundmann, "*BlazePose: On-device Real-time Body Pose Tracking*," Google Research, 1600 Amphitheatre Pkwy, Mountain View, CA 94043, USA, 17 Jun 2020.
- [5] R. Pal *et al.*, "Effect of Maha Mrityunjaya HYMN Recitation on Human Brain for the Analysis of Single EEG Channel C4-A1 Using Machine Learning Classifiers on Yoga Practitioner," *2020 17th International Computer Conference on Wavelet Active Media Technology and Information Processing*, Chengdu, China, 2020, pp. 89-92.
- [6] 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)*, Gwalior, India, 2020, pp. 40-43.

# APPENDICES

## Training image dataset code in python using the activation function softmax and dropout layer

```
import csv

import pandas as pd

from tensorflow import keras

from sklearn.model_selection import train_test_split

from data import BodyPart

import tensorflow as tf

import tensorflow as tf.js

tfjs_model_dir = 'model'


def load_csv(csv_path):

    df = pd.read_csv(csv_path)

    df.drop(['filename'], axis=1, inplace=True)

    classes = df.pop('class_name').unique()

    y = df.pop('class_no')


    X = df.astype('float64')

    y = keras.utils.to_categorical(y)

    return X, y, classes
```

```

def get_center_point(landmarks, left_bodypart, right_bodypart):

    left = tf.gather(landmarks, left_bodypart.value, axis=1)

    right = tf.gather(landmarks, right_bodypart.value, axis=1)

    center = left * 0.5 + right * 0.5

    return center


def get_pose_size(landmarks, torso_size_multiplier=2.5):

    hips_center = get_center_point(landmarks, BodyPart.LEFT_HIP,
BodyPart.RIGHT_HIP)

    shoulders_center = get_center_point(landmarks, BodyPart.LEFT_SHOULDER,
BodyPart.RIGHT_SHOULDER)

    torso_size = tf.linalg.norm(shoulders_center - hips_center)

    pose_center_new = get_center_point(landmarks, BodyPart.LEFT_HIP,
BodyPart.RIGHT_HIP)

    pose_center_new = tf.expand_dims(pose_center_new, axis=1)

    pose_center_new = tf.broadcast_to(pose_center_new, [tf.size(landmarks) // (17*2),
17, 2])

    d = tf.gather(landmarks - pose_center_new, 0, axis=0,
name="dist_to_pose_center")

    max_dist = tf.reduce_max(tf.linalg.norm(d, axis=0))

    pose_size = tf.maximum(torso_size * torso_size_multiplier, max_dist)

    return pose_size


def normalize_pose_landmarks(landmarks):

```

```

    pose_center = get_center_point(landmarks, BodyPart.LEFT_HIP,
BodyPart.RIGHT_HIP)

    pose_center = tf.expand_dims(pose_center, axis=1)

    pose_center = tf.broadcast_to(pose_center, [tf.size(landmarks) // (17*2), 17, 2])

    landmarks = landmarks - pose_center

    pose_size = get_pose_size(landmarks)

    landmarks /= pose_size

    return landmarks

```

```

def landmarks_to_embedding(landmarks_and_scores):

    reshaped_inputs = keras.layers.Reshape((17, 3))(landmarks_and_scores)

    landmarks = normalize_pose_landmarks(reshaped_inputs[:, :, :2])

    embedding = keras.layers.Flatten()(landmarks)

    return embedding

```

```

def preprocess_data(X_train):

    processed_X_train = []

    for i in range(X_train.shape[0]):

        embedding =
landmarks_to_embedding(tf.reshape(tf.convert_to_tensor(X_train.iloc[i]), (1, 51)))

        processed_X_train.append(tf.reshape(embedding, (34)))

    return tf.convert_to_tensor(processed_X_train)

```

```
X, y, class_names = load_csv('train_data.csv')
```

```
X_train, X_val, y_train, y_val = train_test_split(X, y, test_size=0.15)
```

```
X_test, y_test, _ = load_csv('test_data.csv')
```

```
processed_X_train = preprocess_data(X_train)
```

```
processed_X_val = preprocess_data(X_val)
```

```
processed_X_test = preprocess_data(X_test)
```

```
inputs = tf.keras.Input(shape=(34))
```

```
layer = keras.layers.Dense(128, activation=tf.nn.relu6)(inputs)
```

```
layer = keras.layers.Dropout(0.5)(layer)
```

```
layer = keras.layers.Dense(64, activation=tf.nn.relu6)(layer)
```

```
layer = keras.layers.Dropout(0.5)(layer)
```

```
outputs = keras.layers.Dense(len(class_names), activation="softmax")(layer)
```

```
model = keras.Model(inputs, outputs)
```

```
model.compile(
```

```
    optimizer='adam',
```

```
    loss='categorical_crossentropy',
```

```
    metrics=['accuracy']
```

)

```
checkpoint_path = "weights.best.hdf5"
```

```
checkpoint = keras.callbacks.ModelCheckpoint(checkpoint_path,  
monitor='val_accuracy', verbose=1, save_best_only=True, mode='max')
```

```
earlystopping = keras.callbacks.EarlyStopping(monitor='val_accuracy',  
patience=20)
```

```
print('-----TRAINING-----')
```

```
history = model.fit(processed_X_train, y_train, epochs=200, batch_size=16,  
validation_data=(processed_X_val, y_val), callbacks=[checkpoint, earlystopping])
```

```
print('-----EVALUATION-----')
```

```
loss, accuracy = model.evaluate(processed_X_test, y_test)
```

```
print('LOSS: ', loss)
```

```
print("ACCURACY: ", accuracy)
```

```
tfjs.converters.save_keras_model(model, tfjs_model_dir)
```

```
print('tfjs model saved at ', tfjs_model_dir)
```

For preprocessing the data taken to eliminate the corrupted one and making the landmarks to be stored in csv file

```
import tensorflow as tf
```

```
import numpy as np
```

```
import pandas as pd
```

```
import os
```

```
from movenet import Movement
```

```
import wget
```

```
import csv
```

```
import tqdm
```

```
from data import BodyPart
```

```
if ('movenet_thunder.tflite' not in os.listdir()):
```

```
wget.download('https://tfhub.dev/google/lite-model/movenet/singlepose/thunder/tflite/float16/4?lite-format=tflite', 'movenet_thunder.tflite')
```

```
movement = Movement('movenet_thunder')
```

```
def detect(input_tensor, inference_count=3):
```

```
    movenet.detect(input_tensor.numpy(), reset_crop_region=True)
```

```
    for _ in range(inference_count - 1):
```

```
        detection = movenet.detect(input_tensor.numpy(), reset_crop_region=False)
```

```
    return detection
```

```
class Preprocessor(object):
```

```
    def __init__(self, images_in_folder, csvs_out_path):
```

```

self._images_in_folder = images_in_folder

self._csvs_out_path = csvs_out_path

self._csvs_out_folder_per_class = 'csv_per_pose'

self._message = []

if (self._csvs_out_folder_per_class not in os.listdir()):

    os.makedirs(self._csvs_out_folder_per_class)

self._pose_class_names = sorted([n for n in os.listdir(images_in_folder)])

def process(self, detection_threshold=0.1):

    for pose_class_name in self._pose_class_names:

        images_in_folder = os.path.join(self._images_in_folder, pose_class_name)

        csv_out_path = os.path.join(self._csvs_out_folder_per_class, pose_class_name
+ '.csv')

        with open(csv_out_path, 'w') as csv_out_file:

            csv_out_writer = csv.writer(csv_out_file, delimiter=',',
quoting=csv.QUOTE_MINIMAL)

            image_names = sorted([n for n in os.listdir(images_in_folder)])

            valid_image_count = 0

```



```

for image_name in tqdm.tqdm(image_names):

    image_path = os.path.join(images_in_folder, image_name)

    try:

        image = tf.io.read_file(image_path)

        image = tf.io.decode_jpeg(image)

    except:

        self._message.append('Skipped' + image_path + ' Invalid image')

        continue

    if image.shape[2] != 3:

        self._message.append('Skipped' + image_path + ' Image is not in
RGB')

        continue

    person = detect(image)

    min_landmark_score = min([keypoint.score for keypoint in
person.keypoints])

    should_keep_image = min_landmark_score >= detection_threshold

    if not should_keep_image:

```

```
self._message.append('Skipped' + image_path + 'Keypoints score are  
below than threshold')
```

```
continue
```

```
valid_image_count += 1
```

```
pose_landmarks = np.array(  
    [[keypoint.coordinate.x, keypoint.coordinate.y, keypoint.score]  
     for keypoint in person.keypoints],  
    dtype=np.float32)
```

```
coord = pose_landmarks.flatten().astype(np.str).tolist()
```

```
csv_out_writer.writerow([image_name] + coord)
```

```
print(self._message)
```

```
all_landmarks_df = self.all_landmarks_as_dataframe()
```

```
all_landmarks_df.to_csv(self._csvs_out_path, index=False)
```

```
def class_names(self):
```

```
    return self.pose_class_names
```

```
def all_landmarks_as_dataframe(self):
```

```
    total_df = None
```

```
    for class_index, class_name in enumerate(self._pose_class_names):
```

```

        csv_out_path = os.path.join(self._csvs_out_folder_per_class, class_name +
'.csv')

        per_class_df = pd.read_csv(csv_out_path, header=None)

        per_class_df['class_no'] = [class_index] * len(per_class_df)

        per_class_df['class_name'] = [class_name] * len(per_class_df)

        per_class_df[per_class_df.columns[0]] = class_name + '/' +
per_class_df[per_class_df.columns[0]]

        if total_df is None:

            total_df = per_class_df

        else:

            total_df = pd.concat([total_df, per_class_df], axis=0)

        list_name = [[bodypart.name + '_x', bodypart.name + '_y', bodypart.name +
'_score'] for bodypart in BodyPart]

        header_name = []

        for columns_name in list_name:

            header_name += columns_name

        header_name = ['filename'] + header_name

        header_map = {total_df.columns[i]: header_name[i] for i in
range(len(header_name))}

        total_df.rename(header_map, axis=1, inplace=True)

        return total_df

```

```
# preprocess training data
```

```
images_in_folder = os.path.join('yoga_poses', 'train')
```

```
csvs_out_path = 'train_data.csv'
```

```
train_preprocessor = Preprocessor(images_in_folder, csvs_out_path)
```

```
train_preprocessor.process()
```

```
# preprocessing testing data
```

```
images_in_folder = os.path.join('yoga_poses', 'test')
```

```
csvs_out_path = 'test_data.csv'
```

```
test_preprocessor = Preprocessor(images_in_folder, csvs_out_path)
```

```
test_preprocessor.process()
```