Smart Internz

FINAL REPORT

Project Title: Yoga Pose Classification

Team Details:

Name	Application ID	Reg.No	VIT-Email
Yashi Goyal *(Team Leader)*	SPS_APL_20230530939	20BCE0436	yashi.goyal2020@vitstudent.ac.in
Aditi Prashar	SPS_APL_20230530502	20BCE0426	aditi.prashar2020@vitstudent.ac.in
Ishita Gupta	SPS_APL_20230530646	20BCE0446	ishita.gupta2020a@vitstudent.ac.in
Sneha Jayshri	SPS_APL_20230530885	20BCE2673	sneha.jayshri2020a@vitstudent.ac.in

Under the Guidance of:

Mentor: Prince

1 INTRODUCTION

1.1 Overview

In today's busy lifestyle, finding time and opportunities for outdoor exercise or running can be challenging for many individuals. However, practicing yoga indoors offers a convenient and accessible fitness solution with numerous benefits. Yet, for those who are not regular yoga practitioners, achieving proper form and posture in various yoga poses can be difficult.

The project "Yoga Pose Classification" aims to develop a system called Yoga-Tracker, which can classify various yoga poses from static images and real-time poses captured through a camera. It provides users with the ability to track their pose durations, offers guidance on proper pose techniques, and educates them about the benefits of each pose. The project also addresses the lack of a high-quality dataset by incorporating a yoga pose generation system to create synthetic poses for training the classification models.

1.2 Purpose

The purpose of this project is to provide individuals with a convenient and accessible solution for practicing yoga. It enables users to classify and track their yoga poses, receive real-time feedback and guidance, and monitor their progress over time. By offering educational resources and motivating users on their fitness journey, Yoga-Tracker aims to promote physical well-being and a healthier lifestyle.

By utilizing yoga pose detection technology, individuals can not only classify different poses accurately but also assess their own posture and alignment in comparison to the ideal standards. This feature provides valuable feedback, allowing users to make necessary adjustments and improve their yoga practice, ensuring they derive maximum benefits from each pose.

2 LITERATURE SURVEY

2.1 Existing problem

The existing problem is the lack of availability of a good quality dataset for developing a yoga pose classification system. This poses a challenge in training accurate models for recognizing yoga poses.

2.2 Proposed solution

The proposed solution is to build a yoga pose generation system that can create a large, diverse dataset of fake and real poses. This dataset will be used to train the classification models, enabling accurate classification of various yoga poses. Additionally, the system incorporates image processing and machine learning techniques to classify poses from both static images and real-time camera inputs.

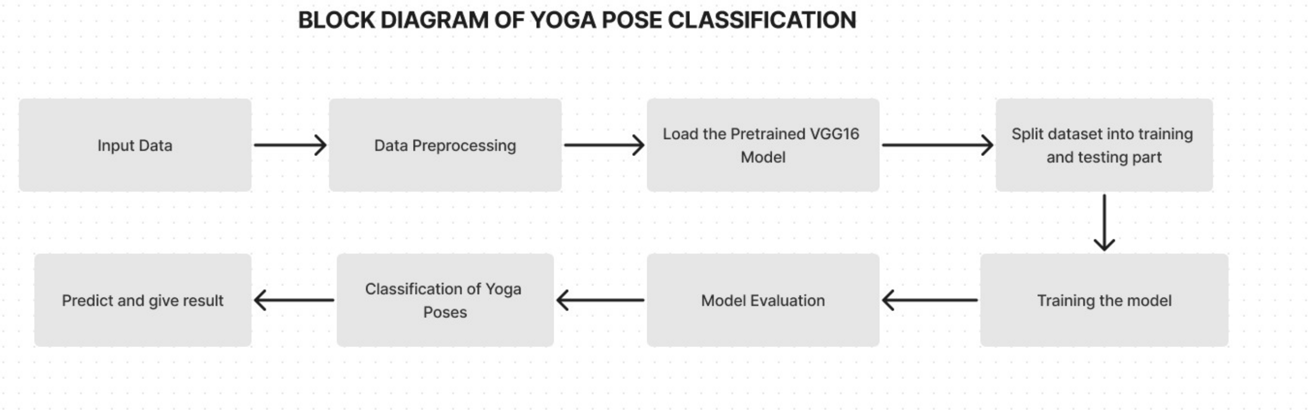
Title	Related Work	Drive to Present work	Gaps Identified
Yoga for posture correction/deformity treatment	Effect on posture alignments and their consequent changes have been discussed with respect to yoga practices. Body posture of university students was calculated on a weekly basis with their varying amount of yoga practice.	Yoga has a positive effect in benefiting the human spine curvature and enables a good posture. Yoga helps in enabling proper circulation and proper functioning of internal body parts leading to positive posture benefits.	If done incorrectly, these poses can harm the physical and physiological structures of the human body and can lead to unforeseen injuries and ailments.
Injuries/Harms Resulting from Incorrect Adjustments/Alignments Performed by Yoga Asana Practitioners	Different parts of the body that have been injured due to misalignment are discussed. Furthermore, different poses have been discussed, especially the ones that are considered risky and tough, as well as the methods in which they can avoid these injuries.	The present study discussed the basic solutions to avoid the above injuries with misalignment which includes expert supervision, warming up, and exercise to strengthen the weak parts of the body, starting with simple poses, and estimating the body's capacity.	Although several poses are discussed, the study does not elaborate on the right steps to do the pose which would help the practitioners identify where they go wrong
Yoga Pose Detection and Classification Using Deep Learning	This paper explains how Deep learning can be used in detecting the posture of a person in a video. In this method, different models like POSENET and OPENPOSE are used to detect the poses in the video given by the user and then compare them with the already	The method discussed in this paper can be easily and effectively used to detect incorrect poses and the efficiency of this method is also above 98%.	The only con in this paper is that the method could only be used to detect 6 postures for now.

	present videos of experts.		
Adversarial PoseNet: A Structure-aware Convolutional Network for Human Pose Estimation	This paper discusses the shortcomings of basic GAN-based D-CNN models that, due to bad image conditions or external props mimicking human-like features, render human posture and occultation's impossible. PoseNet is a multitask pose generator with two discriminator networks.	This paper gives an insight upon the need and construction of PoseNet as well as factors that lead to the improved posture detection and all three components that help in verifying the correctness of the prediction.	The paper doesn't touch upon the feasibility of implementation as well scalability and latency. The algorithm is only implemented on known datasets rather than real-life applications for testing.
Implementation of Machine Learning Technique for Identification of Yoga Poses	This paper uses tf-pose which is a human pose estimation algorithm provided by Google's Tensorflow to draw a skeleton of the human body on real-time bases. Angles are extracted from the various joints and fed as features to machine learning models. These angles are stored in a CSV file and used for classification.	The authors of this paper had a big dataset of about 5500 images which helps the model learn better. They also achieved an accuracy of 99.04% by using a Random Forest model.	The algorithm is only implemented on known datasets rather than real-life applications for testing. Collecting data manually of a huge volume requires a lot of effort, attention and precision. The image needs to be taken with a lot of care and effort so it is not a feasible way of making a dataset.
Yoga Posture Recognition By Detecting Human Joint Points In Real Time Using Microsoft Kinect	This paper uses Microsoft Kinect, which a motion sensor is made by Microsoft for its gaming console Xbox, to detect yoga poses. The Kinect has a skeletal tracking tool which can track 20 joints of the human body. The angles	Xbox is a very common console bought by families with children and hence Kinect is widely available and can be used easily. The model provides an accuracy of about 97%.	A small deviation of 2.5 degrees in the angle of the posture leads to no detection of the posture. This is hard to achieve for people learning yoga for the first time.

	between joints are calculated from the coordinates Kinect returns.		

3 THEORITICAL ANALYSIS

3.1 Block diagram


BLOCK DIAGRAM OF YOGA POSE CLASSIFICATION
Input Data
Data Preprocessing
Load the Pretrained VGG16 Model
Split dataset into training and testing part
Training the model
Model Evaluation
Classification of Yoga Poses
Predict and give result

The block diagram provides an overview of the project's components and their interconnections. It depicts the flow of information from input sources (static images or real-time camera) through the pose detection and classification algorithms, resulting in the output of classified yoga poses and pose duration tracking.

3.2 Hardware / Software designing

The hardware requirements for this project include a camera or webcam for capturing real-time poses.

The software requirements involve image processing libraries, machine learning frameworks, and a programming environment for implementing the pose detection and classification algorithms.

Dataset: https://www.kaggle.com/datasets/elysian01/yoga-pose-classification

Libraries used-

• *Keras- This is the main library used in this project. The model is entirely built on Keras including all the image augmentation, transfer learning, training, and testing.*

• *NumPy- Used for pixels manipulation*

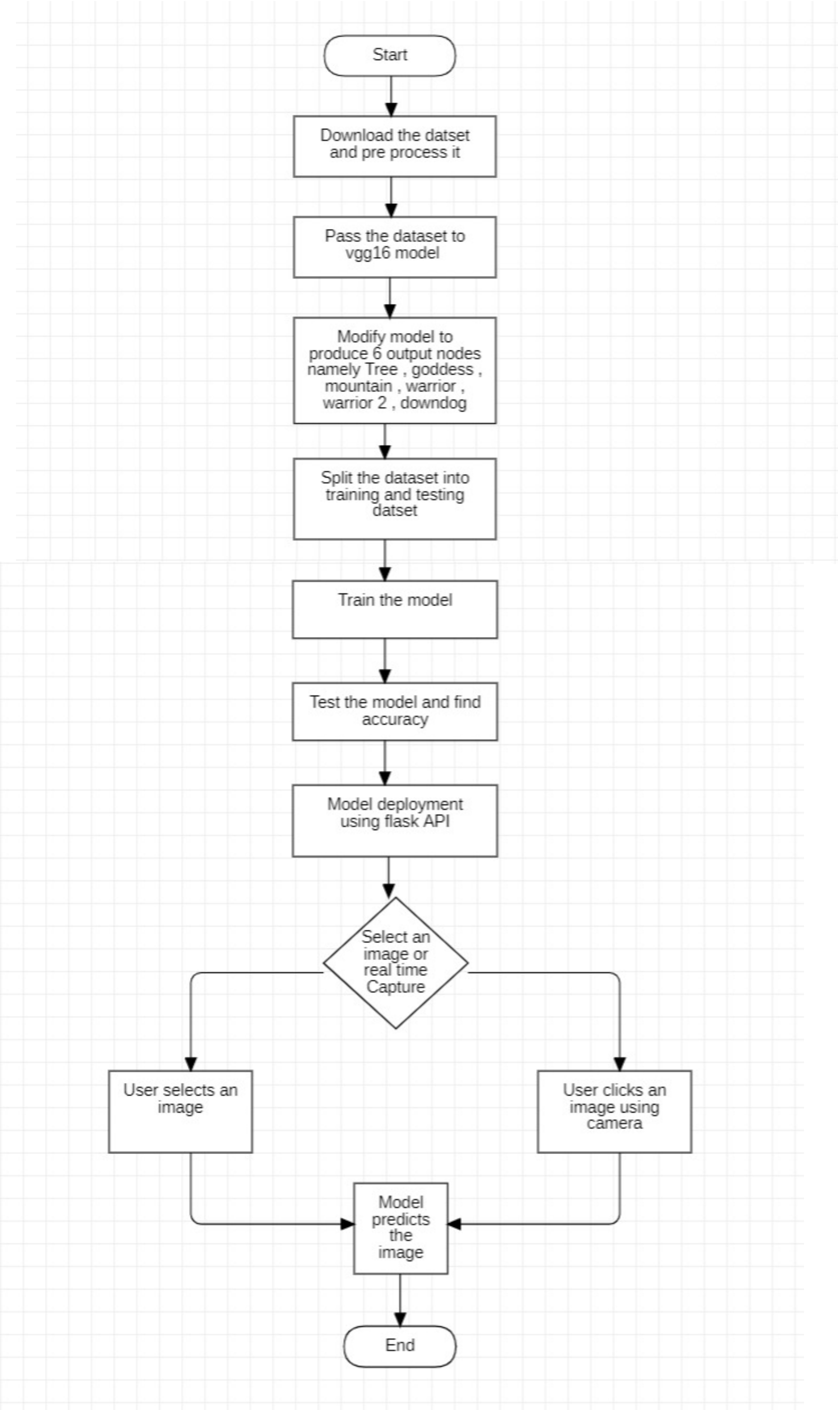
• *Matplotlib- To plot images and loss plots.*

• *Flask- Used for deploying our classification model.*

4 EXPERIMENTAL INVESTIGATIONS

Experimental investigations involve analysing and testing various approaches to pose detection, feature extraction, and classification. This includes evaluating different machine learning algorithms, fine-tuning model parameters, and optimizing performance to achieve accurate and efficient yoga pose classification.

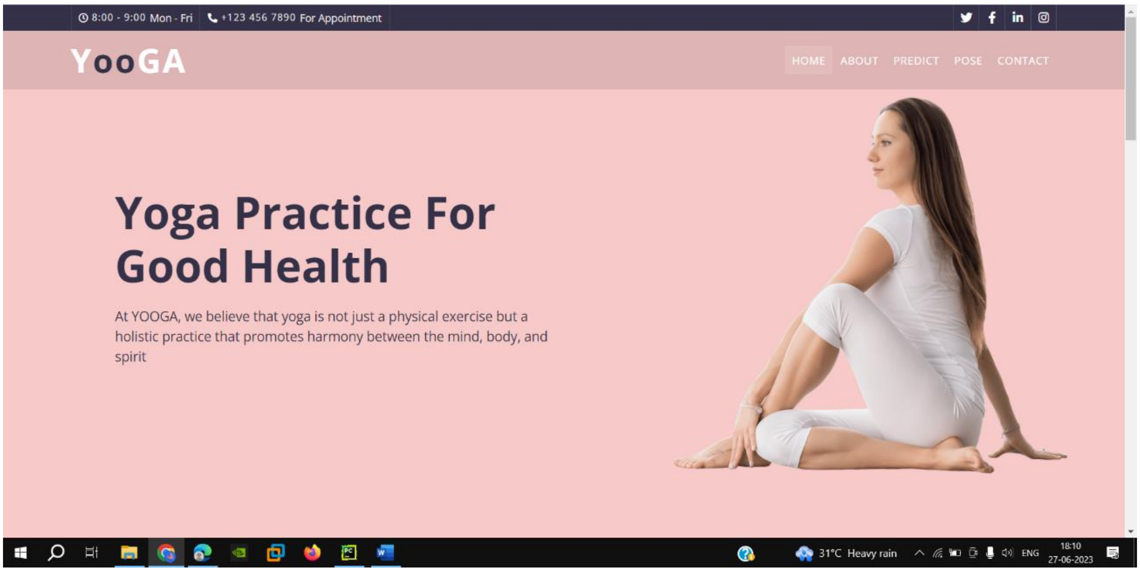
5 FLOWCHART


Start
Download the datset and pre process it
Pass the dataset to vgg16 model
Modify model to produce 6 output nodes namely Tree , goddess , mountain , warrior , warrior 2 , downdog
Split the dataset into training and testing datset
Train the model
Test the model and find accuracy
Model deployment using flask API
Select an image or real time Capture
User selects an image
User clicks an image using camera
Model predicts the image
End

The flowchart illustrates the control flow of the solution, showing the steps involved in pose detection, feature extraction, classification, and duration tracking. It provides a visual representation of the decision-making process and the logic behind the system's operation.

6 RESULT

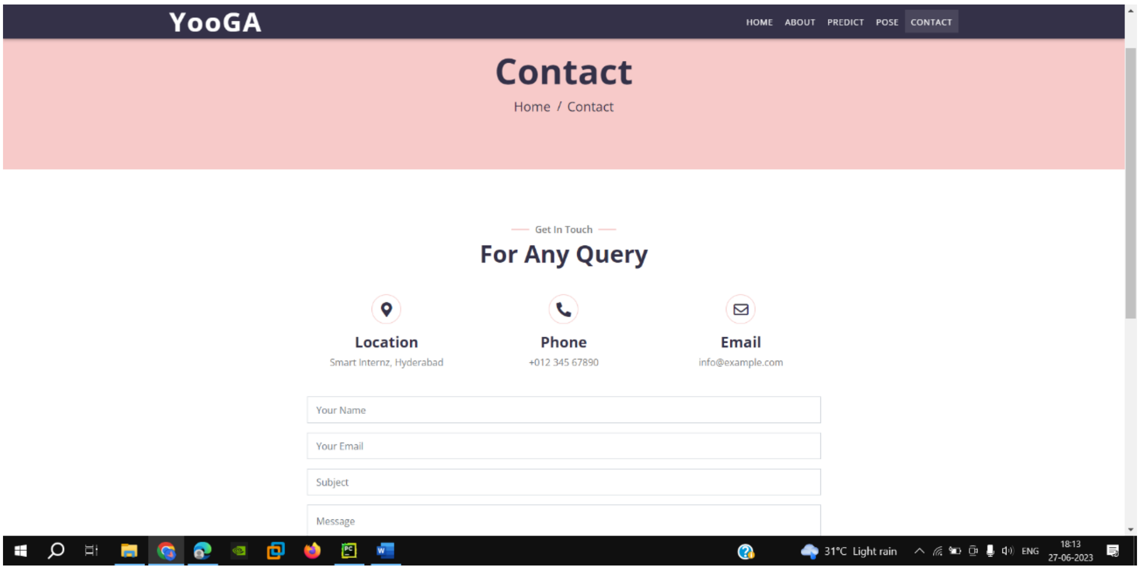
1. Landing Page



Yoga Practice For Good Health

At YOOGA, we believe that yoga is not just a physical exercise but a holistic practice that promotes harmony between the mind, body, and spirit

2. Contact Us Page



Contact

Home / Contact

Get In Touch

For Any Query

Location
Smart Internz, Hyderabad

Phone
+012 345 67890

Email
info@example.com

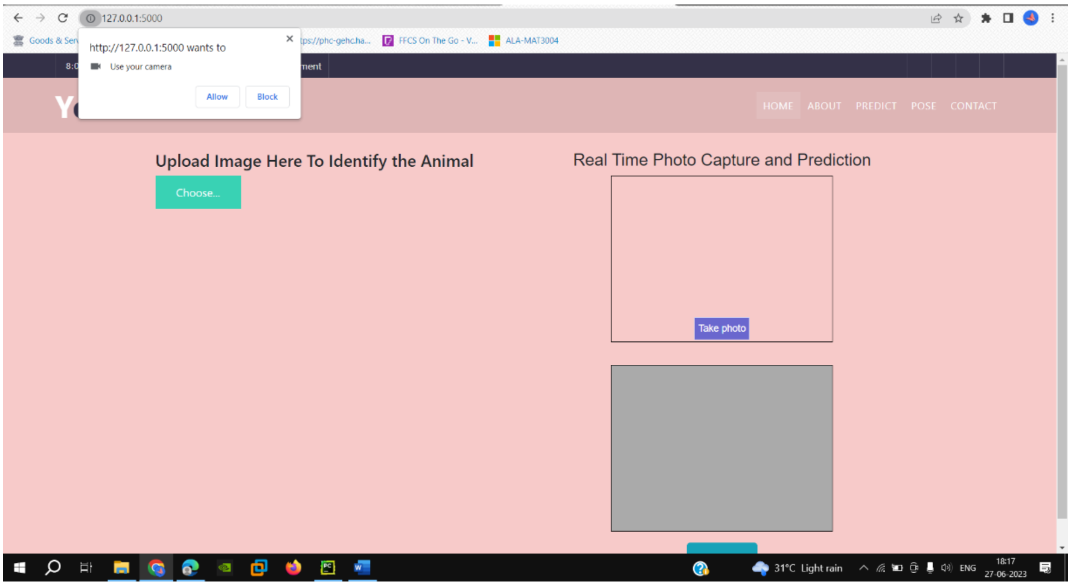
Your Name

Your Email

Subject

Message

3. Prediction by uploading an image

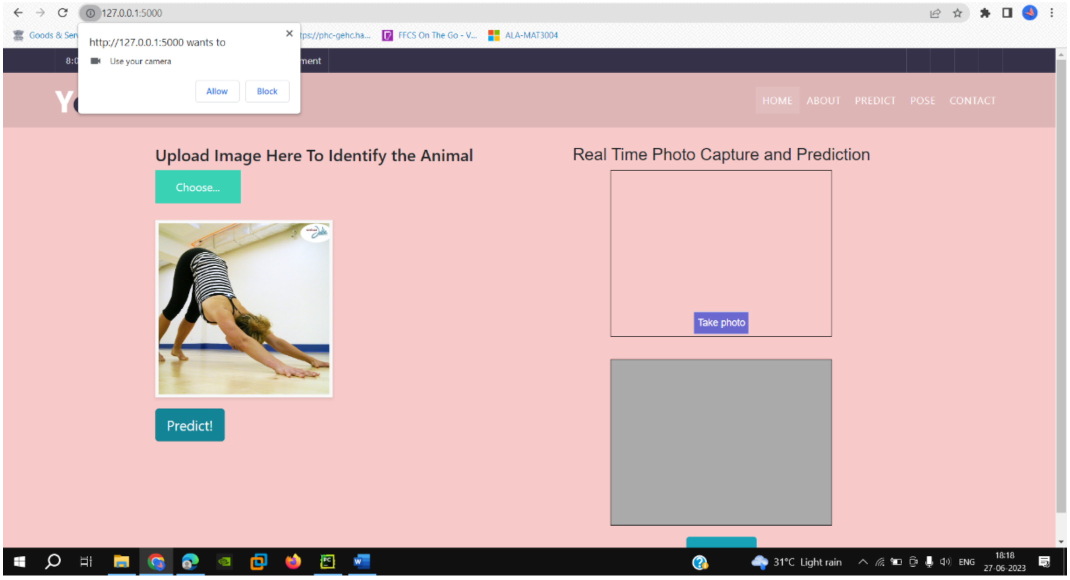


http://127.0.0.1:5000 wants to

Use your camera

Upload Image Here To Identify the Animal

Real Time Photo Capture and Prediction

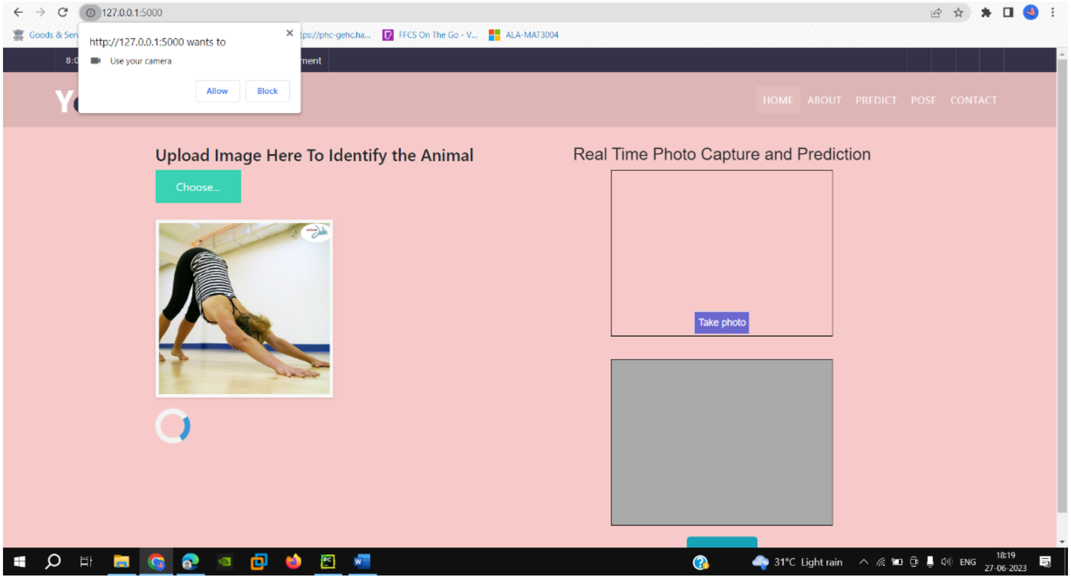


http://127.0.0.1:5000 wants to

Use your camera

Upload Image Here To Identify the Animal

Real Time Photo Capture and Prediction

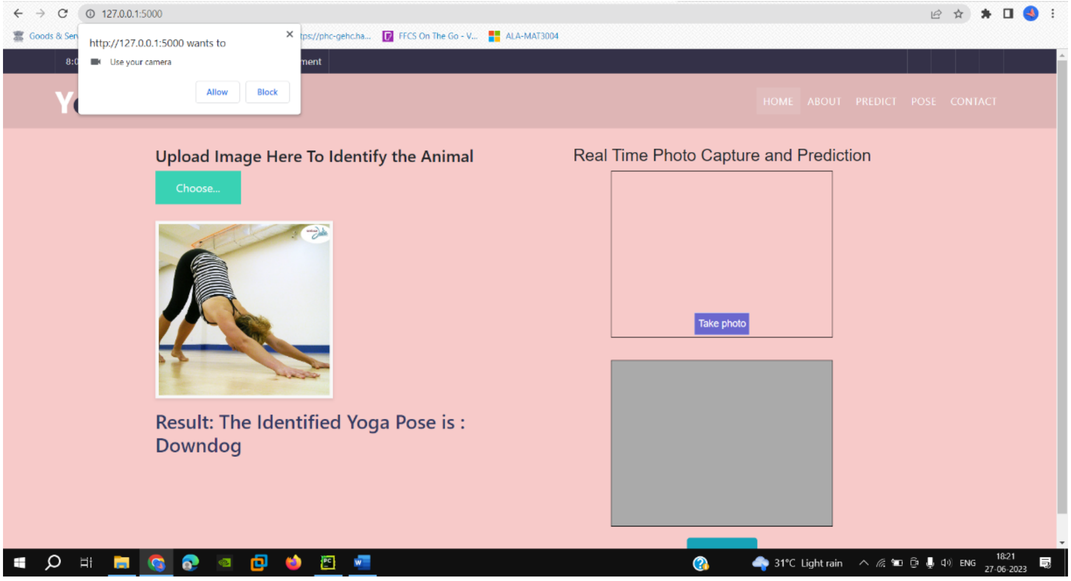


http://127.0.0.1:5000 wants to

Use your camera

Upload Image Here To Identify the Animal

Real Time Photo Capture and Prediction

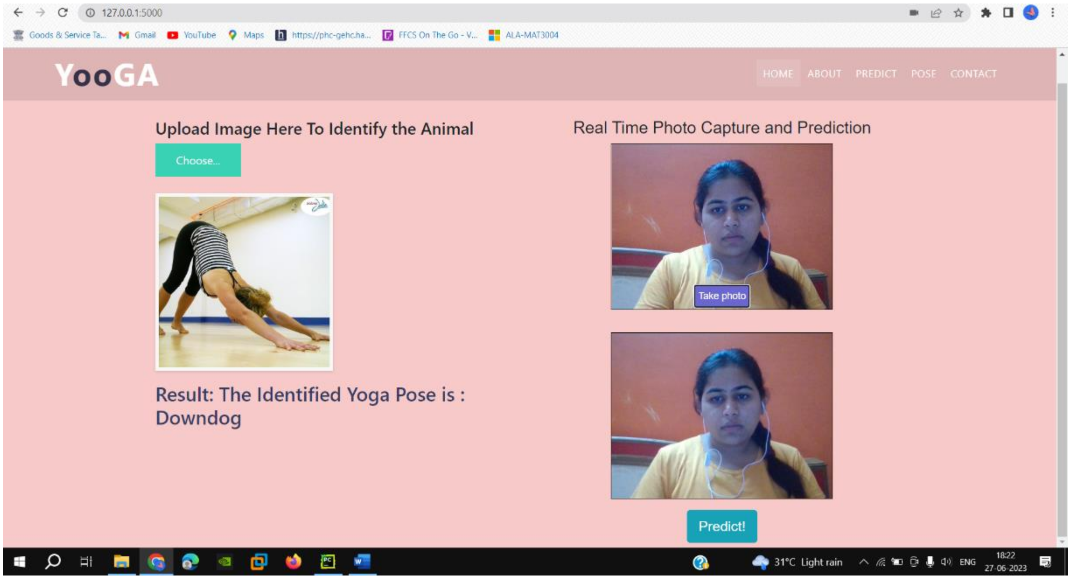


Upload Image Here To Identify the Animal

Real Time Photo Capture and Prediction

Result: The Identified Yoga Pose is : Downdog

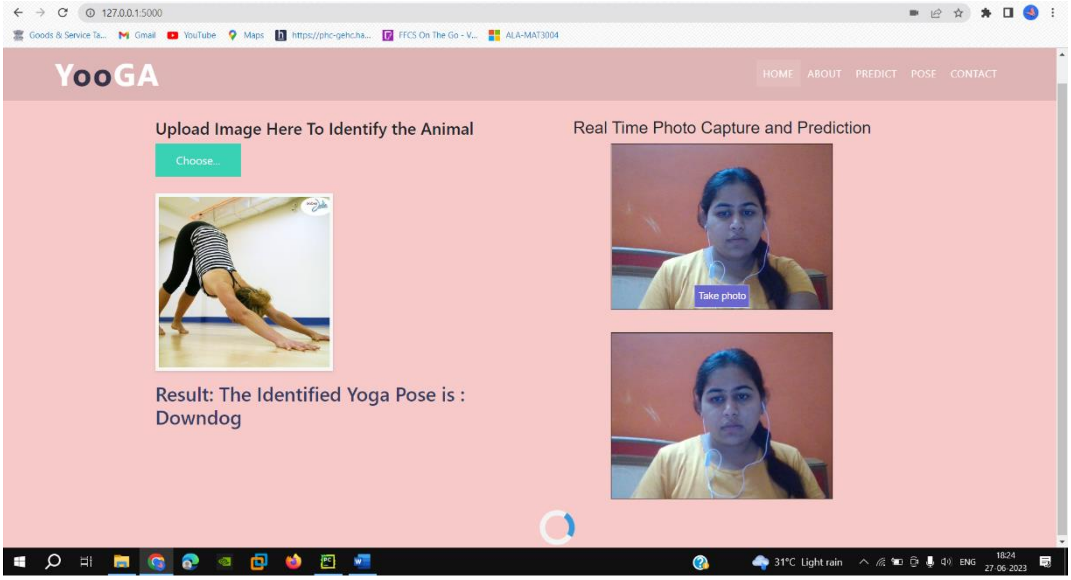
4. Real Time Image Capturing and Prediction



Upload Image Here To Identify the Animal

Real Time Photo Capture and Prediction

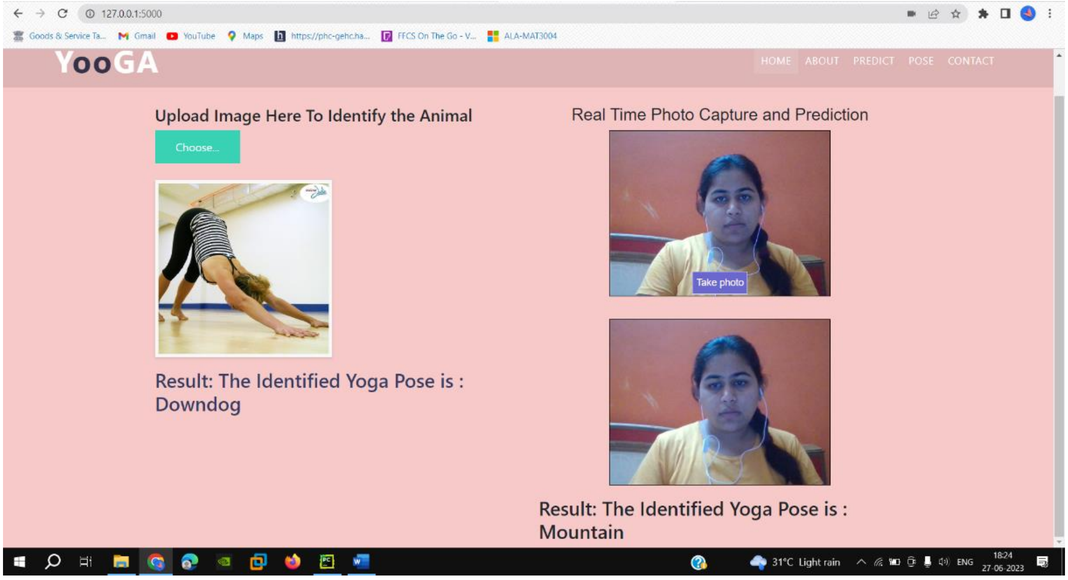
Result: The Identified Yoga Pose is : Downdog



Upload Image Here To Identify the Animal

Real Time Photo Capture and Prediction

Result: The Identified Yoga Pose is : Downdog



Upload Image Here To Identify the Animal

Result: The Identified Yoga Pose is : Downdog

Real Time Photo Capture and Prediction

Result: The Identified Yoga Pose is : Mountain

The result section presents the final findings and outputs of the project. It includes screenshots or visual representations of the Yoga-Tracker interface, demonstrating the accurate classification of yoga poses, real-time pose tracking provided to users.

7 ADVANTAGES & DISADVANTAGES

Advantages of the proposed solution may include:

- Convenient and accessible yoga practice at the user's own pace.
- Real-time feedback and guidance for correct pose execution.
- Progress tracking and motivation for users' fitness journeys.
- Educational resources on various yoga poses and their benefits.

Disadvantages may include:

- Dependency on camera or webcam for real-time pose recognition.
- Potential challenges in accurately detecting complex or subtle yoga poses.
- Limitations in the accuracy of pose duration tracking.

8 APPLICATIONS

The proposed solution has applications in various areas, including:

- Personal fitness and wellness tracking.
- Yoga studios and instructors for guiding and monitoring students.
- Fitness apps and platforms for integrating yoga practice features.
- Healthcare and rehabilitation centers for promoting therapeutic yoga.

9 CONCLUSION

The conclusion summarizes the entire work and findings of the project. It emphasizes the successful development of the Yoga-Tracker system, which provides users with accurate yoga pose classification, real-time guidance, pose duration tracking, and educational resources. The project demonstrates the potential of combining image processing, machine learning, and synthetic data generation to overcome challenges in yoga pose recognition.

10 FUTURE SCOPE

- Expanding the dataset and classification models to include more yoga poses.
- Incorporating advanced pose tracking techniques, such as skeleton tracking.
- Integrating the system with wearable devices for more accurate pose duration tracking.
- Developing mobile applications for on-the-go yoga practice and tracking.
- Adding scoring features to tell how well your yoga posture is Comparing the perfect pose with the user's pose will help the user to improve by correcting their posture
- Adding Style Transfer. The person will be able to add stylish backgrounds to their image.

11 BIBILOGRAPHY

- R. Muralidharan, S. J. Dinakaran, and S. Shree, "Real-time Pose Recognition and Tracking for Yoga Asanas," International Journal of Advanced Research in Computer Science, vol. 9, no. 5, pp. 641-646, 2018.

- D. P. Karunaweera and D. P. Athukorala, "Yoga Pose Classification using Convolutional Neural Networks," 16th International Conference on Advances in ICT for Emerging Regions (ICTer), pp. 140-146, 2016.

- Chen, X., Xie, H., Cheng, G., Poon, L.K., Leng, M. and Wang, F.L., 2020. Trends and features of the applications of natural language processing techniques for clinical trials text analysis. Applied Sciences, 10(6), p.2157.

- Hruthika, JChaitanya, P.K. and Shiva Chaithanya, G., Deep Learning Based Human Pose Estimation Using Opencv. International Journal of Innovations in Engineering Research and Technology, 7(12), pp.246-253.

- X. Zhao, Y. Wu, Y. Song, and Y. Lu, "Yoga Pose Classification Based on Transfer Learning and SVM," 14th International Symposium on Distributed Computing and Applications to Business, Engineering and Science (DCABES), pp. 1-4, 2015.

- Irani, G.S., 2020. Injuries/Harms Resulting from Incorrect Adjustments/Alignments Performed by Yoga Asana Practitioners. Archives of Pharmacy Practice, 1, p.38.

- Gupta, S. Ranjan, V. Sharma, K. Chaudhary, and A. Kaur, "Deep Learning for Yoga Pose Estimation," International Conference on Machine Learning, Image Processing, Network Technologies and Data Sciences, pp. 19-25, 2019.

Appendix: Yoga Pose Classification - Glossary of Terms

1. Accuracy:
A measure of how correctly the Yoga Pose Classification system identifies and classifies yoga poses. It quantifies the system's ability to produce accurate results, typically expressed as a percentage or a metric such as precision, recall, or F1 score.

2. Pose Detection:
The technique or algorithm used to detect and localize the human body or specific body parts within an image or video frame, which is essential for accurately recognizing yoga poses.

3. Feature Extraction:
The process of extracting relevant information or features from the detected poses, such as body joint locations or angles, to represent the pose in a meaningful and quantifiable manner.

4. Classification Model:
A machine learning model trained to assign a specific label or category to a given input. In the context of Yoga Pose Classification, the model is trained to recognize and classify different yoga poses based on the extracted features.

5. Pose Duration Tracking:
The functionality that tracks the length of time a user holds a specific yoga pose. It provides feedback and motivation to users by recording and visualizing their progress over time.

6. Synthetic Data Generation:
The process of creating artificial or synthetic data samples to supplement a limited or inadequate real-world dataset. In the context of Yoga Pose Classification, synthetic data generation can be used to augment the training data and improve the performance and diversity of the classification models.

7. Image Processing:
The domain of computer vision that involves analysing, manipulating, and interpreting images to extract meaningful information or enhance their quality. In Yoga Pose

Classification, image processing techniques may be employed for tasks such as pose detection and pre-processing.

9. Skeleton Tracking:
A technique that involves tracking the movement and position of the human skeletal structure to accurately estimate body poses. Skeleton tracking can be used as an alternative or complementary approach to pose detection for yoga pose classification.

10. User Interface (UI):
The graphical interface or system that allows users to interact with the Yoga Pose Classification application. It provides visual feedback, options for pose classification, and displays information such as pose duration and instructional resources.

APPENDIX

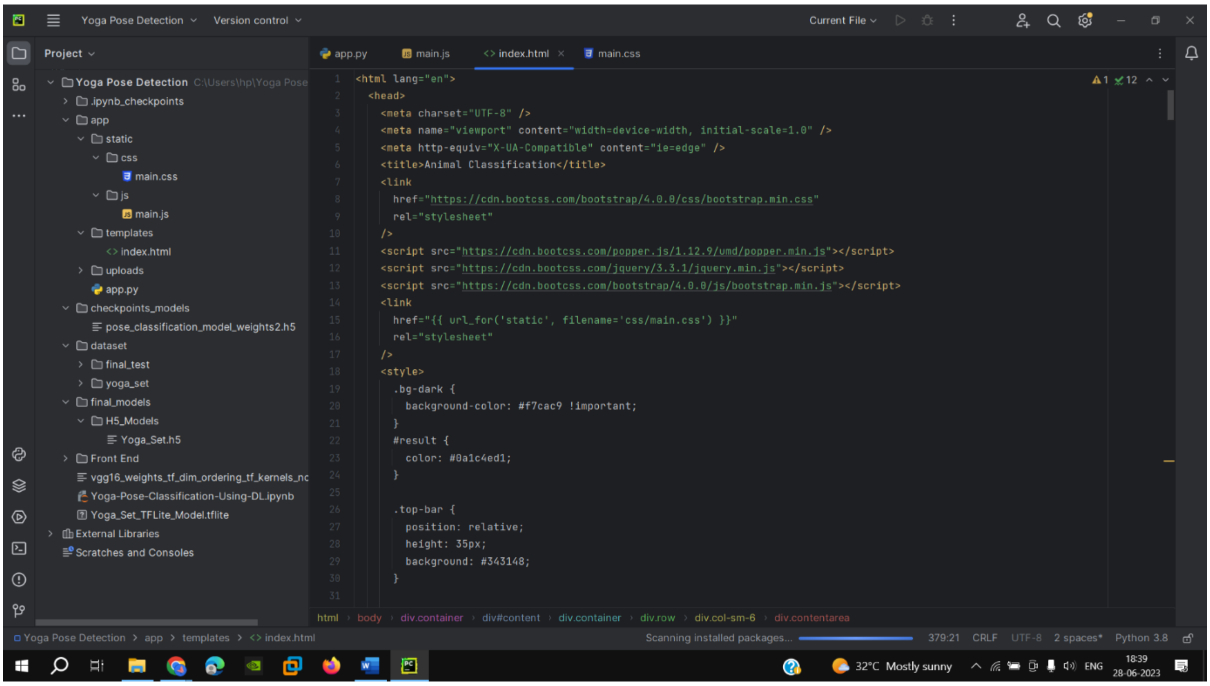
A. Source Code

- app.py

```
# !pip install --upgrade tensorflow
import numpy as np
import os
from tensorflow.keras.models import load_model
from tensorflow.keras.preprocessing import image
from flask import Flask, render_template, request

app = Flask(__name__)
model = load_model("../final_models/H5_Models/Yoga_Set.h5")


@app.route('/')
def index():
    return render_template("index.html")


@app.route('/predict', methods=['GET', 'POST'])
def upload():
    if request.method == 'POST':
        f = request.files['image']
        basepath = os.path.dirname(__file__)
        filepath = os.path.join(basepath, 'uploads', f.filename)
        f.save(filepath)
        img = image.load_img(filepath, target_size=(300, 300))
        x = image.img_to_array(img)
        x = np.expand_dims(x, axis=0)
        pred = np.argmax(model.predict(x), axis=1)
        index = ["Downdog", "Goddess", "Mountain", "Tree", "Warrior1",
"Warrior2"]
        text = "The Identified Yoga Pose is : " + str(index[pred[0]])
    return text
```

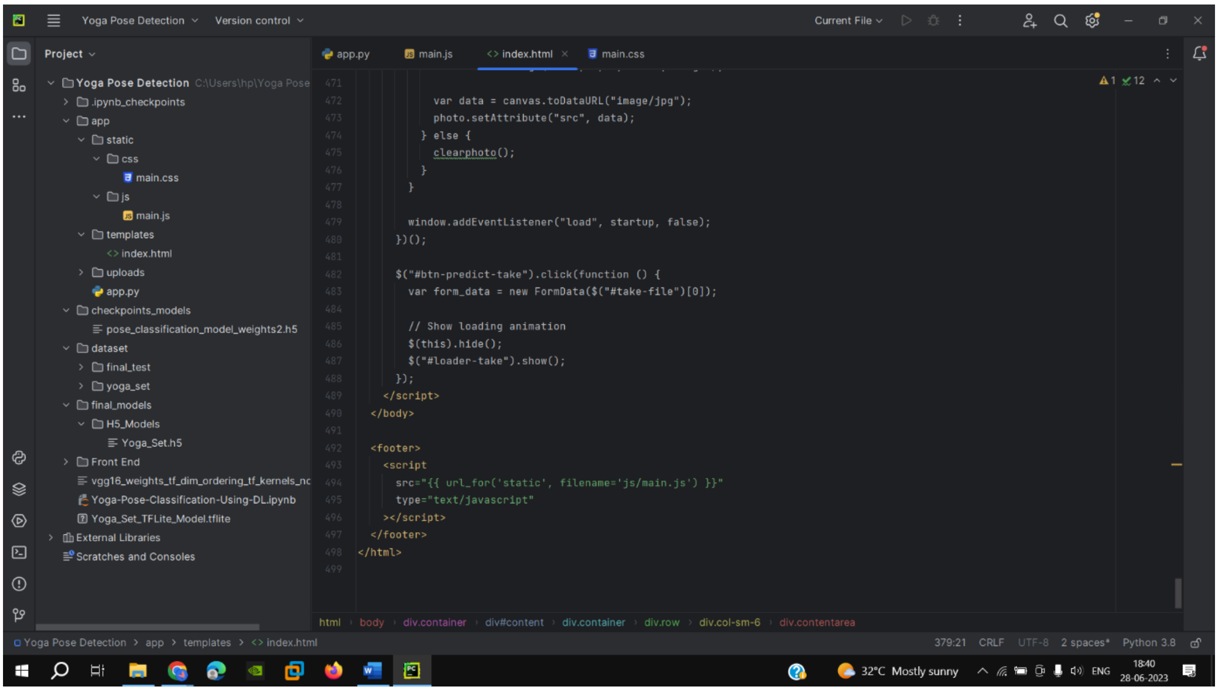
```
if __name__ == '__main__':
    app.run()
```

- index.html



```
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <meta http-equiv="X-UA-Compatible" content="ie=edge" />
    <title>Animal Classification</title>
    <link
      href="https://cdn.bootcss.com/bootstrap/4.0.0/css/bootstrap.min.css"
      rel="stylesheet"
    />
    <script src="https://cdn.bootcss.com/popper.js/1.12.9/umd/popper.min.js"></script>
    <script src="https://cdn.bootcss.com/jquery/3.3.1/jquery.min.js"></script>
    <script src="https://cdn.bootcss.com/bootstrap/4.0.0/js/bootstrap.min.js"></script>
    <link
      href="{{ url_for('static', filename='css/main.css') }}"
      rel="stylesheet"
    />
    <style>
      .bg-dark {
        background-color: #f7cac9 !important;
      }
      #result {
        color: #0a1c4ed1;
      }

      .top-bar {
        position: relative;
        height: 35px;
        background: #343148;
      }

```

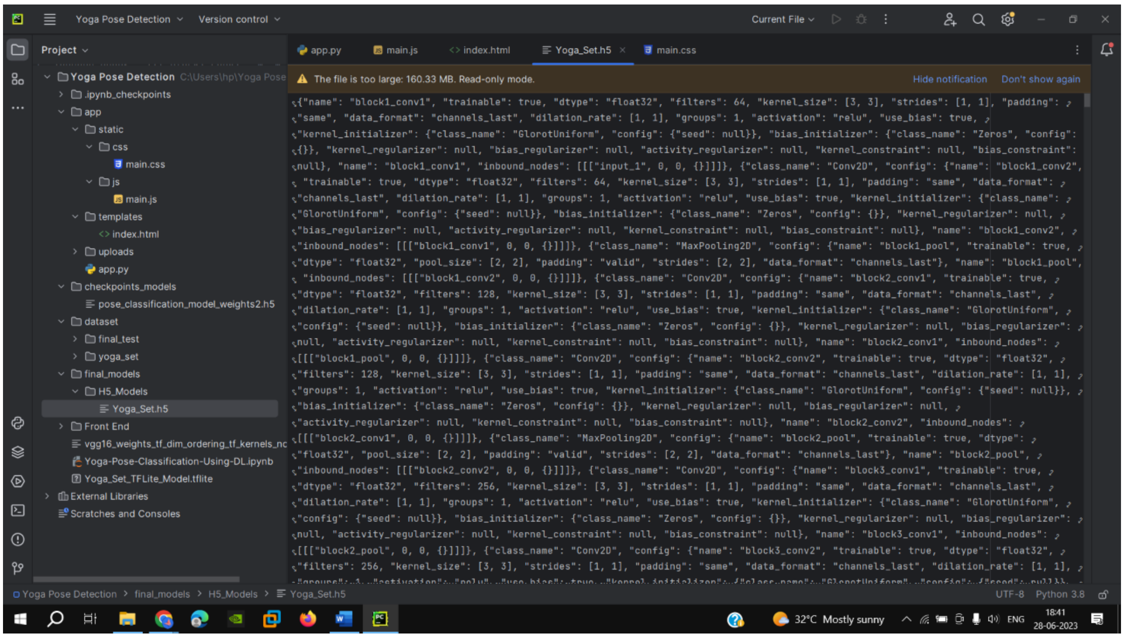
Yoga Pose Detection
Version control
Current File
Project
app.py
main.js
index.html
main.css
Yoga Pose Detection C:\Users\hp\Yoga Pose
.ipynb_checkpoints
app
static
css
main.css
js
main.js
templates
index.html
uploads
app.py
checkpoints_models
pose_classification_model_weights2.h5
dataset
final_test
yoga_set
final_models
H5_Models
Yoga_Set.h5
Front End
vgg16_weights_tf_dim_ordering_tf_kernels_nc
Yoga-Pose-Classification-Using-DL.ipynb
Yoga_Set_TFLite_Model.tflite
External Libraries
Scratches and Consoles
html › body › div.container › div#content › div.container › div.row › div.col-sm-6 › div.contentarea
Yoga Pose Detection › app › templates › index.html
Scanning installed packages...
379:21 CRLF UTF-8 2 spaces* Python 3.8
32°C Mostly sunny
ENG
18:39
28-06-2023



```
            var data = canvas.toDataURL("image/jpg");
            photo.setAttribute("src", data);
          } else {
            clearphoto();
          }
        }

        window.addEventListener("load", startup, false);
      })();

      $("#btn-predict-take").click(function () {
        var form_data = new FormData($("#take-file")[0]);

        // Show loading animation
        $(this).hide();
        $("#loader-take").show();
      });
    </script>
  </body>

  <footer>
    <script
      src="{{ url_for('static', filename='js/main.js') }}"
      type="text/javascript"
    ></script>
  </footer>
</html>

```

Yoga Pose Detection
Version control
Current File
Project
app.py
main.js
index.html
main.css
Yoga Pose Detection C:\Users\hp\Yoga Pose
.ipynb_checkpoints
app
static
css
main.css
js
main.js
templates
index.html
uploads
app.py
checkpoints_models
pose_classification_model_weights2.h5
dataset
final_test
yoga_set
final_models
H5_Models
Yoga_Set.h5
Front End
vgg16_weights_tf_dim_ordering_tf_kernels_nc
Yoga-Pose-Classification-Using-DL.ipynb
Yoga_Set_TFLite_Model.tflite
External Libraries
Scratches and Consoles
html › body › div.container › div#content › div.container › div.row › div.col-sm-6 › div.contentarea
Yoga Pose Detection › app › templates › index.html
379:21 CRLF UTF-8 2 spaces* Python 3.8
32°C Mostly sunny
ENG
18:40
28-06-2023

- Saved Model – Yoga_Set.h5



The file is too large: 160.33 MB. Read-only mode.

```
{"name": "block1_conv1", "trainable": true, "dtype": "float32", "filters": 64, "kernel_size": [3, 3], "strides": [1, 1], "padding":
"same", "data_format": "channels_last", "dilation_rate": [1, 1], "groups": 1, "activation": "relu", "use_bias": true,
"kernel_initializer": {"class_name": "GlorotUniform", "config": {"seed": null}}, "bias_initializer": {"class_name": "Zeros", "config":
{}}, "kernel_regularizer": null, "bias_regularizer": null, "activity_regularizer": null, "kernel_constraint": null, "bias_constraint":
null}, "name": "block1_conv1", "inbound_nodes": [[["input_1", 0, 0, {}]]]}, {"class_name": "Conv2D", "config": {"name": "block1_conv2",
 "trainable": true, "dtype": "float32", "filters": 64, "kernel_size": [3, 3], "strides": [1, 1], "padding": "same", "data_format":
"channels_last", "dilation_rate": [1, 1], "groups": 1, "activation": "relu", "use_bias": true, "kernel_initializer": {"class_name":
"GlorotUniform", "config": {"seed": null}}, "bias_initializer": {"class_name": "Zeros", "config": {}}, "kernel_regularizer": null,
"bias_regularizer": null, "activity_regularizer": null, "kernel_constraint": null, "bias_constraint": null}, "name": "block1_conv2",
"inbound_nodes": [[["block1_conv1", 0, 0, {}]]]}, {"class_name": "MaxPooling2D", "config": {"name": "block1_pool", "trainable": true,
"dtype": "float32", "pool_size": [2, 2], "padding": "valid", "strides": [2, 2], "data_format": "channels_last"}, "name": "block1_pool",
 "inbound_nodes": [[["block1_conv2", 0, 0, {}]]]}, {"class_name": "Conv2D", "config": {"name": "block2_conv1", "trainable": true,
"dtype": "float32", "filters": 128, "kernel_size": [3, 3], "strides": [1, 1], "padding": "same", "data_format": "channels_last",
"dilation_rate": [1, 1], "groups": 1, "activation": "relu", "use_bias": true, "kernel_initializer": {"class_name": "GlorotUniform",
"config": {"seed": null}}, "bias_initializer": {"class_name": "Zeros", "config": {}}, "kernel_regularizer": null, "bias_regularizer":
null, "activity_regularizer": null, "kernel_constraint": null, "bias_constraint": null}, "name": "block2_conv1", "inbound_nodes":
[[["block1_pool", 0, 0, {}]]]}, {"class_name": "Conv2D", "config": {"name": "block2_conv2", "trainable": true, "dtype": "float32",
"filters": 128, "kernel_size": [3, 3], "strides": [1, 1], "padding": "same", "data_format": "channels_last", "dilation_rate": [1, 1],
"groups": 1, "activation": "relu", "use_bias": true, "kernel_initializer": {"class_name": "GlorotUniform", "config": {"seed": null}},
"bias_initializer": {"class_name": "Zeros", "config": {}}, "kernel_regularizer": null, "bias_regularizer": null,
"activity_regularizer": null, "kernel_constraint": null, "bias_constraint": null}, "name": "block2_conv2", "inbound_nodes":
[[["block2_conv1", 0, 0, {}]]]}, {"class_name": "MaxPooling2D", "config": {"name": "block2_pool", "trainable": true, "dtype":
"float32", "pool_size": [2, 2], "padding": "valid", "strides": [2, 2], "data_format": "channels_last"}, "name": "block2_pool",
"inbound_nodes": [[["block2_conv2", 0, 0, {}]]]}, {"class_name": "Conv2D", "config": {"name": "block3_conv1", "trainable": true,
"dtype": "float32", "filters": 256, "kernel_size": [3, 3], "strides": [1, 1], "padding": "same", "data_format": "channels_last",
"dilation_rate": [1, 1], "groups": 1, "activation": "relu", "use_bias": true, "kernel_initializer": {"class_name": "GlorotUniform",
"config": {"seed": null}}, "bias_initializer": {"class_name": "Zeros", "config": {}}, "kernel_regularizer": null, "bias_regularizer":
null, "activity_regularizer": null, "kernel_constraint": null, "bias_constraint": null}, "name": "block3_conv1", "inbound_nodes":
[[["block2_pool", 0, 0, {}]]]}, {"class_name": "Conv2D", "config": {"name": "block3_conv2", "trainable": true, "dtype": "float32",
"filters": 256, "kernel_size": [3, 3], "strides": [1, 1], "padding": "same", "data_format": "channels_last", "dilation_rate": [1, 1],
"groups": 1, "activation": "relu", "use_bias": true, "kernel_initializer": {"class_name": "GlorotUniform", "config": {"seed": null}},
```

Thank You