**\*\*Program Overview:\*\***

This project aims to identify and label yoga poses from video frames, intending to extract a yoga flow sequence automatically from user input video. It involves gathering images of various yoga poses, organizing them into pose labeled folders, extracting physical landmarks using MediaPipe, and training machine learning models with the labeled landmark data. Despite the project's unique approach to the aim, the models achieved a high level of accuracy in training but performed poorly in execution leading to its conclusion. Despite the projects lack-luster conclusion, it still provided many valuable insights into computer vision applications, pose recognition challenges and machine learning model development.

**\*\*Program Description:\*\***

This program is designed to analyze a yoga pose dataset and predict the type of poses from frames extracted from a given video. The program consists of several components:

1. \*\*Video Frames Extraction\*\*: Downloads a video from a YouTube URL using PyTube and extracts frames from the video at regular intervals using OpenCV.

2. \*\*Pose Detection and Landmark Extraction\*\*: Uses MediaPipe Pose API to detect the pose of a person in each frame and extract landmark coordinates (such as nose, eyes, ears, etc.).

3. \*\*Landmark Processing and Feature Engineering\*\*: Processes landmark coordinates to create a set of engineered features, such as angle calculations between different body parts.

4. \*\*Machine Learning Modeling\*\*: Uses pre-trained Machine Learning models (Keras, Random Forest, and Logistic Regression) to predict the type of pose from the extracted features.

5. \*\*Pose Prediction and Output\*\*: Predicts the type of pose for each frame and outputs the predicted pose as a label.

6. \*\*Image Annotation\*\*: Annotates each frame with the predicted pose label and saves the annotated images in designated model folder.

7. \*\* 3D Data Visualizations\*\*: Not relevant to the machine learning program but still valuable is a Plotly script for visualizing and interacting with the landmark data. This script will interact with the CSV file used for training the model and will show landmarks plotted in 3D. Visualizing this data is what initially inspired the engineered features.

**\*\*Program Usage:\*\***

To run the program, simply provide the YouTube video URL, and the program will extract frames, detect poses, and run the landmark coordinates through the pre-trained models provided to predict the types of poses. The program will then label the predictions on the images and save them in model’s folder (Keras, Random Forest, or Logistic Regression). Note throughout the code there are several variables that will require adjustment based on the your specific directories. Also not that this program uses pytube and may require adjustments based on Youtubes ever changing api.

**\*\*Program Outputs:\*\***

The program produces several outputs:

1. Directory of frames extracted from the designated video.

2. Directory of frames with the wireframes overlaid on images

3. Directory containing the annotated images with predicted pose labels for each model.

4. CSV file containing the landmark coordinates and engineered features used for prediction.

**\*\*Program Requirements:\*\***

1. Python 3.x

2. OpenCV

3. MediaPipe Pose API

4. PyTube

5. scikit-learn (for Random Forest and Logistic Regression models)

6. Keras (for Keras model)

7. Numpy

8. Pandas

9. Joblib

10. matplotlib (for ploting)

**\*\*Program Structure:\*\***

main/

│

├── data/

│ ├── ‘yoga\_program\_ML\_training\_data.csv’ #data extracted from images

│ └── data\_for\_3D\_visualization\_and\_plotly\_script

│

├── models/

│ ├── notebooks #ipynb & py files used in training

│ ├── saved\_models #models that performed well

│

│

├── ulitilies/

│ ├── scripts for running program

│

├── results/

├── results from model training

├── results from testing on unseen data

**\*\*Program Limitations:\*\***

1. The program assumes that the input video is in a supported format (e.g., MP4).

2. The program may require adjustments to the frame extraction interval and landmark detection settings to optimize performance.

3. The program may not work well with videos containing multiple persons or irregular poses.

4. The program may require additional models to be trained or updated to improve accuracy.

5. Pytube is a notoriously fussy, so you may have to work with some of these files to get it to work (I encountered issues and manually modified both innertube and cipher files from pytubes packages)

**\*\*Program Future Development:\*\***

1. Improve model accuracy by collecting additional data and fine-tuning the models.

2. Create pipeline for training on ‘unknown’ poses.

3. Goal is to be able to extract complete yoga flow.

4. Integrate with other yoga-related tools and applications.