YOGA POSE DETECTION

A PROJECT REPORT

Submitted by

Virendra Kumar Singh(20BCS7753)

in partial fulfillment for the award of the degree of

BACHELOR OF ENGINEERING

IN

COMPUTER SCIENCE ENGINEERING

Chandigarh University

FEB-MAY 2023

BONAFIDE CERTIFICATE

Certified that this project report **"YOGA POSE DETECTION"** is the bonafide work of "**Virendra Kumar Singh" who** carried out the project work under my/our supervision.

DR. SANDEEP SINGH KANG
HEAD OF THE DEPARTMENT
COMPUTER SCIENCE ENGINEERING
3RD YEAR

ER. Amandeep Kaur
SUPERVISOR
COMPUTER SCIENCE ENGINEERING

Submitted for the project viva-voce examination held on_

INTERNAL EXAMINER **EXTERNAL EXAMINER**

TABLE OF CONTENTS

List of Figures

ABSTRACT

Yoga pose detection is the process of identifying and classifying different yoga poses performed by individuals. This task has important applications in the fields of fitness tracking, physical therapy, and sports performance analysis. In recent years, advances in computer vision and machine learning have enabled the development of accurate and efficient yoga pose detection systems. These systems typically use image or video data and employ deep learning models to identify key points on the body and track their movements over time. This abstract will provide an overview of the state-of-theart techniques used in yoga pose detection, including the challenges and limitations of current methods, as well as potential directions for future research in this field.
Keywords: Online Education, Video, Video Editor, On the Fly, Streaming, Visualization, Big Data

GRAPHICAL ABSTRACT

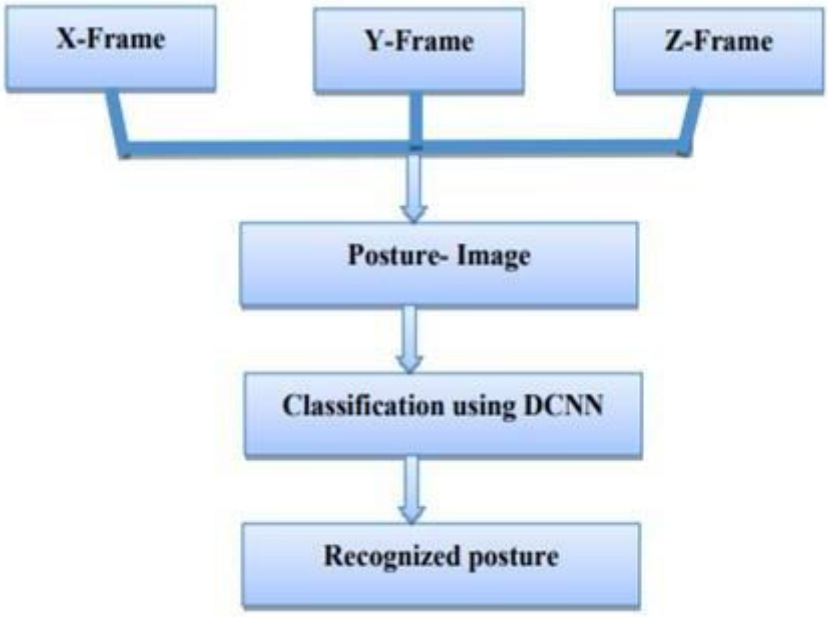


Figure 1: Graphical Abstract

CHAPTER 1.
INTRODUCTION

1.1. Client Identification/Need Identification/Identification of relevant Contemporary issue

The client base typically consists of individuals or organizations involved in the development of yoga-related applications or technology solutions. This includes yoga studios, fitness companies, health and wellness apps, as well as researchers and developers in the fields of computer vision and artificial intelligence.

The clients require a reliable and robust pose detection system that can accurately detect and identify yoga poses in real-time or from recorded video footage. The system should assist yoga practitioners in their practice by offering feedback on alignment and execution, tracking their progress, and providing personalized guidance or recommendations.

A pertinent contemporary issue in yoga pose detection is the growing demand for remote or virtual yoga practices. With the increasing popularity of online yoga classes and fitness apps, there is a need for precise pose detection that can bridge the gap between in-person instruction and remote learning. This involves delivering real-time feedback, correcting alignment, and ensuring practitioners perform poses correctly to prevent injuries.

1.2. Identification of Problem

The problem with detecting yoga poses can be defined as follows:

Variation in Poses: Yoga poses exhibit differences in body position, alignment, and angles. Practitioners may also execute the same pose with slight variations. These variances make it difficult to create a reliable and precise pose detection system that can accommodate these differences.

Occlusion and Noise: In real-life situations, occlusion and noise can occur due to factors like clothing, props, cluttered backgrounds, or the presence of other people. These obstructions and disturbances can obscure body parts, leading to inaccurate pose detection or misinterpretation.

Ambiguity in Pose Transitions: Smooth transitions between yoga poses are crucial, and the detection system should be capable of accurately identifying these transitions. However, certain poses may have similar starting or ending positions, posing a challenge in distinguishing between them based solely on static frames.

Limited and Variable Dataset: Gathering a diverse and comprehensive dataset for yoga poses can be challenging. The availability of annotated data for different body types, skill levels, and variations in pose execution is often limited, impacting the accuracy and generalizability of pose detection models.

Real-time Performance: Real-time pose detection is necessary for interactive applications or systems providing immediate feedback. Achieving high accuracy while maintaining low latency presents a significant challenge due to the computational demands of pose estimation algorithms.

Pose Detection in Different Environments: Yoga is practiced in various settings, such as studios, homes, or outdoor locations. These diverse environments introduce variations in lighting conditions, backgrounds, and camera perspectives, which can affect the accuracy and robustness of pose detection algorithms.

1.3. Identification of Tasks

The whole project is divided into three modules. And the description of the modules is given below –

- The first module is the data(pose) collection part of the project that includes various Poses.
- The second module is the data collection part.

- The third module is the review, then training.

1.4. Project Scope

1. Yoga pose detection involves identifying and classifying different yoga postures in images or videos.

2. This can be useful for yoga practitioners, instructors, and researchers to monitor and improve their practice, as well as to develop personalized workout plans.

3. Our project aims to detect poses in yoga.

4. The project consists of Pose classifications, Pose estimation and Real-time pose tracking.

1.5. Timeline

The time required for the completion of the whole project is depicted using the following Gantt chart –

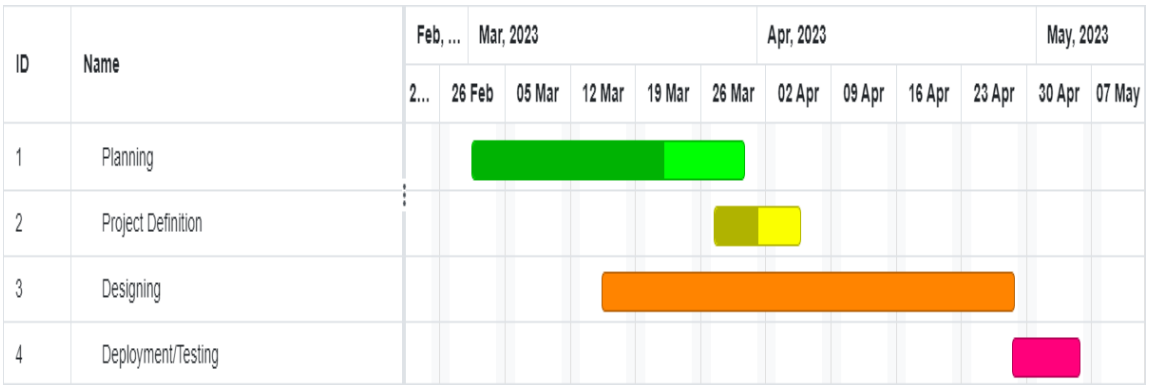


Figure 2: Gantt chart

1.6. Organization of the Report

The 1st phase of the project was project planning; the structure and the timeline of the project were created. The task was evenly divided among the team.

In the 2nd phase, the project definition was created that included the objectives, the scope of the project, and the purpose of the project.

In the 3rd phase designing constraints were created, and the framework of the design was built. The 4th phase is the building of the project which includes various modules with a specific timeline and keeping the project well within the constraints.

The 5th phase will have the testing of the project before the deployment/hosting to check for expected bugs and find out whether the code and programming work is according to the project constraints.

The last phase is the deployment, after verifying and testing of the project it will be hosted/deployed and maintained.

CHAPTER 2.

LITERATURE REVIEW/BACKGROUND STUDY

2.1. Literature Review:

Face recognition technology has widely attracted attention due to its enormous application value and market potential. It is being implemented in various fields like security system, digital video processing, and many such technological advances. Additionally, music is the form of art, which is known to have a greater connection with a person's emotion. It has got a unique ability to lift up one's mood. Relatively, this paper focuses on building an efficient Yoga Poses Detection which determines the Poses of user using Facial Recognition techniques

2.2. Goals/Objective

The goal of yoga pose detection using Python is to build a system or application that can automatically identify and classify different yoga poses from images or videos in real-time. This technology can be used to provide feedback and guidance to yoga practitioners, monitor progress, and provide personalized recommendations for improvement

2.3. Review Summary

Yoga pose detection using Python involves analyzing images or videos of people performing yoga poses and identifying the specific pose being executed. The process includes data collection, image preprocessing, pose estimation, pose classification, and application integration. Popular libraries and frameworks used for implementation include OpenCV, Dlib, TensorFlow, Keras, Flask, and Django. The end result is an application or system that can take real-time video input and output the predicted pose.

CHAPTER 3.

DESIGN FLOW/PROCESS

3.1 Feature/Characteristics Identification:

1. Objectives

The goal of yoga pose detection using Python is to build a system or application that can automatically identify and classify different yoga poses from images or videos in real-time. This technology can be used to provide feedback and guidance to yoga practitioners, monitor progress, and provide personalized recommendations for improvement

2. Single entity

A project is one whole thing. This means that in a project although different people contribute still is recognized as a single entity. The teams are often specifically assembled for a single project.

3. Life span

Project took 3 weeks to get completed and includes various phases. Initiation and Planning took 4 days. Execution of the project which includes coding and debugging took 13 days to get completed. Monitoring and testing took 1 day for completion. Feedback and documentation where in continuous progress.

4. Require funds- Since, our project is a low budget project it does not require any external funds. The funds for the project are collected from the team members.

5. Life Cycle-

Data collection: This stage involves collecting a dataset of yoga pose images or videos that will be used to train the model. The dataset should be diverse and representative of different body types, clothing, lighting conditions, and camera angles. Preprocessing: The data collected in the first stage needs to be preprocessed to ensure that it is of high quality and is ready for training. Preprocessing techniques may include image resizing, normalization, augmentation, and annotation. Model development: This stage involves selecting an appropriate deep learning model architecture, such as convolutional neural networks (CNNs), and training the model on the preprocessed dataset. The model should be optimized for accuracy, speed, and efficiency. Testing and evaluation: Once the model is trained, it needs to be tested and evaluated on a separate dataset

that was not used in the training process. This step ensures that the model can generalize well to new data and is not overfitting to the training dataset. Deployment: The final stage involves deploying the trained model in a real-world setting, such as a mobile app or a web application. The deployment process may involve additional optimizations for speed, memory usage, and accuracy. Maintenance: Once the model is deployed, it may need to be updated periodically to improve its performance or to adapt to new data. Maintenance may involve retraining the model on new data or fine-tuning the existing model.

6. Risk and Uncertainty

The use of cameras or sensors to detect yoga poses can raise privacy concerns. Users may be uncomfortable with having their movements tracked and recorded, especially if the data is shared with third-party providers. While deep learning models have shown promising results in detecting yoga poses, there is still room for improvement. Factors such as clothing, lighting, and camera angle can affect the accuracy of the model. The availability of a limited dataset can affect the accuracy and generalizability of the model. A dataset that is not diverse enough may lead to biased or inaccurate results.

7. Directions

Developing real-time yoga pose detection systems that can provide immediate feedback to the user can be useful for self-monitoring during yoga practice. Developing more interpretable deep learning models that can provide explanations for their predictions can increase the transparency and trustworthiness of the system.

8. Flexibility

Different poses may require different levels of flexibility, and pose-specific thresholds can be used to account for this variability. For example, the model may have higher thresholds for more advanced poses that require greater flexibility.

3.2 Constraint Identification of Yoga pose detection

- **Scalability:** One of the biggest challenges of building an online job portal using Django is ensuring that it can handle a large number of users and job postings. As the website grows, it can become slow and unresponsive, which can impact the user experience. To address this constraint, developers must optimize the database and use proper dbms to ensure that

the website can handle a high volume of traffic.

- **Security:** Job portals are attractive targets for cybercriminals as they hold sensitive data like personal information, employment history, and financial details. The website must have robust security measures in place, such as secure login and authorization

- **User experience:** The website must be user-friendly and intuitive, with clear navigation and easy-to-use features. Users should be able to find what they are looking for quickly, and the website must load quickly to avoid frustrating users. The design of the website should be modern and visually appealing, with a responsive design that adapts to different devices.

- **Compatibility:** The website must be compatible with a range of browsers, operating systems, and devices to ensure that it reaches the widest possible audience. Developers must test the website on different platforms to ensure that it works as intended.

- **Data management:** Job portals deal with a large volume of data, including job postings, resumes, and user profiles. Developers must ensure that the data is organized and managed effectively, with appropriate backup and disaster recovery procedures in place.

- **Integration:** The website must integrate with other systems, such as social media platforms. Developers must ensure that the integration is seamless and that data is transferred securely between systems.

- **Maintenance:** The website must be maintained regularly to ensure that it remains up-to-date and secure. Developers must monitor the website for issues, fix bugs promptly, and provide support to users. They must also keep up-to-date with the latest technologies and industry trends to ensure that the website remains competitive.

- **Cost:** Building and maintaining an online job portal can be expensive. Developers must balance the features and functionality of the website with the available budget. They must

also consider ongoing costs like hosting, security, and maintenance when building the website.

- **Risk:** A technical risk generally leads to failure of functionality and performance. Causes of technical risks are:
  - ✓ Continuous changing requirements.
  - ✓ No advanced technology is available or the existing technology is in the initial stages.
  - ✓ The product is complex to implement.
  - ✓ Difficult project module integration.

3.3 Analysis of features and finalization subject to constraints

The feasibility of the project is analyzed in this phase and business proposal is put forth with a very general plan for the project and some cost estimates. During system analysis the feasibility study of the proposed system is to be carried out. This is to ensure that the proposed system is not a burden to the company. Three key considerations involved in the feasibility analysis is

Technical feasibility

This study is carried out to check the technical feasibility, that is, the technical requirements of the system. Any system developed must not have a high demand on the available technical resources. This will lead to high demands on the available technical resources. This will lead to high demands being placed on the client. The developed system must have a modest requirement, as only minimal or null changes are required for implementing this system.

Operational feasibility

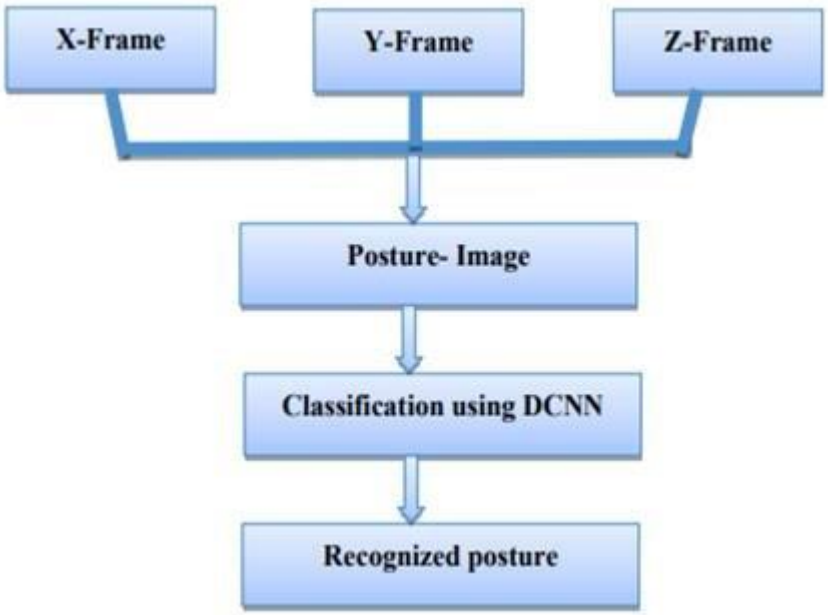
The project is operationally feasible as the user having basic knowledge about computer and Internet can use. Furthermore, the project can be easily used if the computers have no internet access by downloading the video.

Schedule feasibility

The schedule feasibility analysis is carried out using the CPM method. CPM was used to identify critical tasks and calculate the interrelationship between tasks. The plan was carried out which defined critical and non-critical tasks with the goal of preventing time frame problems and process bottlenecks.

3.4 Design Flow

1. DFD: DFD is the abbreviation for Data Flow Diagram. The flow of data of a system or a process is represented by DFD. It also gives insight into the inputs and outputs of each entity and the process itself. DFD does not have a control flow and no loops or decision rules are present.



2. Use Case Diagram: A use case diagram is a graphical depiction of a user's possible interactions with a system. A use case diagram shows various use cases and different types

of users the system has and will often be accompanied by other types of diagrams as well. The use cases are represented by either circles or ellipses.

3. ER Diagram stands for Entity Relationship Diagram, also known as ERD is a diagram that displays the relationship of entity sets stored in a database. In other words, ER diagrams help to explain the logical structure of databases. ER diagrams are created based on three basic concepts: entities, attributes, and relationships.

- **Managing Users**

The user_account table has the following columns:

- id – This is both the table's primary key and a unique identifier for each user. This ID will be referred to by other tables in the data model.
- user_type_id – This signifies whether the user is a job seeker or a recruiter.
- email – This column holds the user's email address. It acts as another user ID for the portal.
- password – This stores an encrypted account password (created by users during registration).
- date_of_birth and gender – As their names suggest, these columns hold users' date of birth and gender.
- is_active – Initially this column would be "Y", but users can set their profile to inactive, or "N". This column stores their choice.
- contact_number – This is the phone number (usually mobile) provided during registration. Users can receive SMS (text) notifications on this number. It can be the same number (or not) as the one job seekers list in their profile or resume.

- sms_notification_active and email_notification_active – These columns store users' preferences regarding receiving notifications through text and/or email.
- user_image – This is a BLOB-type attribute that stores each user's profile image. Since this portal allows only one profile image per user, it makes sense to store it here.
- registration_date – This column keeps a record of when the user registered with the portal.

- **Building Profiles**
  - In the company table, we have the following columns:
  - id – The primary key of this table is also used to uniquely identify companies.
  - company_name – As the column name suggests, this holds the legal name of a company.
  - profile_description – This contains a brief description of each company.
  - business_stream_id – This column depicts which business stream a company belongs to. For example, an oil and gas exploration company can hire IT engineers , but their main business stream remains "Oil and Gas".
  - establishment_date – This column tells you how old a company is.
  - company_website_url – This is a mandatory (non-nullable) column. It holds a pointer to company's official website so job seekers can find out more information.

CHAPTER 4.
DETAILED SYSTEM DESIGN/TECHNICAL DETAILS

4.1 Use of Modern tools in Design and analysis

- **Python** is a high-level, general-purpose programming language. Its design philosophy emphasizes code readability with the use of significant indentation via the off-side rule. Python is dynamically typed and garbage-collected.
- **OpenCV (Open Source Computer Vision Library)** is a popular open-source computer vision and machine learning library. It provides a wide range of functions and algorithms for image and video processing, feature extraction, object detection, and more, making it a powerful tool for various computer vision tasks.
- **OpenPose** is an open-source real-time multi-person key point detection library. It uses deep learning techniques to estimate human body key points, including joint locations, from images or video streams. OpenPose is widely used for pose estimation tasks in applications like action recognition, augmented reality, and human-computer interaction.
- **MediaPipe** is an open-source framework developed by Google that enables the development of real-time multimedia processing pipelines. It provides a set of pre-built components and tools for tasks like video and audio processing, gesture recognition, face detection, pose estimation, and more. MediaPipe is flexible, cross-platform, and widely used in various multimedia applications.
- **dlib** is a powerful open-source C++ library that offers machine learning algorithms and tools for various tasks such as face detection, facial landmark estimation, object tracking, and image classification. It is widely used in computer vision and deep learning applications, providing efficient and accurate solutions for complex tasks.

4.2 Discussion and report/results analysis

- In planning for yoga pose detection, key steps include dataset collection, annotation of pose key points, selection of appropriate deep learning models, data preprocessing, training and fine-tuning, evaluation on test sets, and optimization for real-time performance. Iterative refinement and validation ensure accurate and efficient pose detection results.
- The target audience for yoga pose detection includes yoga practitioners, instructors, fitness enthusiasts, and researchers. It caters to individuals seeking real-time feedback on their yoga poses, instructors looking for automated monitoring and correction tools, and researchers interested in analyzing and understanding human movement patterns in yoga practice.
- The features used in yoga pose detection typically include body key points, joint angles, body part orientations, and spatial relationships between body parts. These features enable the model to capture the pose's unique configuration and provide the necessary information for accurate and robust detection of various yoga postures.
- Development in yoga pose detection is focused on improving accuracy, real-time performance, and robustness. Advancements include leveraging deep learning models, refining pose estimation algorithms, optimizing for different environments and body types, integrating with wearable devices, and enhancing user interfaces for intuitive feedback and analysis of yoga poses.
- Designing views in yoga pose detection involves considering different perspectives for capturing yoga poses, such as front, side, or top-down views. Multiple camera setups, sensor arrays, or 3D reconstruction techniques may be employed to capture comprehensive views that provide sufficient information for accurate pose estimation and analysis.
- Testing in yoga pose detection involves evaluating the accuracy, robustness, and performance of the detection system. This includes testing on diverse datasets, measuring precision and recall, conducting user studies, assessing real-time performance, and validating the system's ability to handle variations in lighting conditions, body orientations, and different yoga poses.
- Functionality testing in yoga pose detection involves evaluating the accuracy and

reliability of the detection system. This includes testing the model's ability to correctly identify and track yoga poses across various scenarios, assessing its performance on different body types, evaluating real-time responsiveness, and conducting extensive validation to ensure consistent and reliable results.

- Performance testing in yoga pose detection involves evaluating the accuracy, speed, and stability of the detection system. This includes measuring metrics such as pose estimation accuracy, frame processing time, memory usage, and handling of challenging scenarios like occlusions or varied lighting conditions. Rigorous testing ensures reliable and efficient pose detection in real-world settings.
- Security testing in yoga pose detection involves assessing potential vulnerabilities and risks in the system. It includes evaluating data privacy measures, ensuring secure communication protocols, preventing unauthorized access to user data, and conducting penetration testing to identify and mitigate potential security threats and vulnerabilities.
- Deployment in yoga pose detection involves integrating the pose detection system into user-friendly applications or platforms. This includes creating mobile apps, web-based tools, or smart devices that provide real-time feedback, analysis, and personalized recommendations for yoga practitioners, instructors, and fitness enthusiasts to enhance their yoga practice experience.

4.3 Project Management for Yoga Pose Detection:

1. Define project scope: Clearly define the scope of the project, including the objectives, deliverables, and timeline. The scope should be discussed and agreed upon by all team members.
2. Create a project plan: Develop a project plan that includes the tasks, timeline, and resource allocation required to complete the project. The plan should be broken down into manageable chunks, and each task should have a clear deadline and assigned team member.
3. Monitor progress: Regularly monitor progress against the project plan, and adjust as necessary to ensure the project stays on track. Use project management tools, such

as Trello or Asana, to track progress and identify any potential bottlenecks.

4. Manage risks: Identify potential risks to the project, and develop strategies to mitigate those risks. For example, if a key team member becomes unavailable, identify a backup plan or assign additional resources to the task.
5. Collaborate effectively: Foster a culture of collaboration among team members, and encourage open communication and feedback. Regularly check in with team members to ensure they have the resources they need to complete their tasks and encourage them to ask for help if necessary.

4.4 Professional Communication for Yoga Pose Detection:

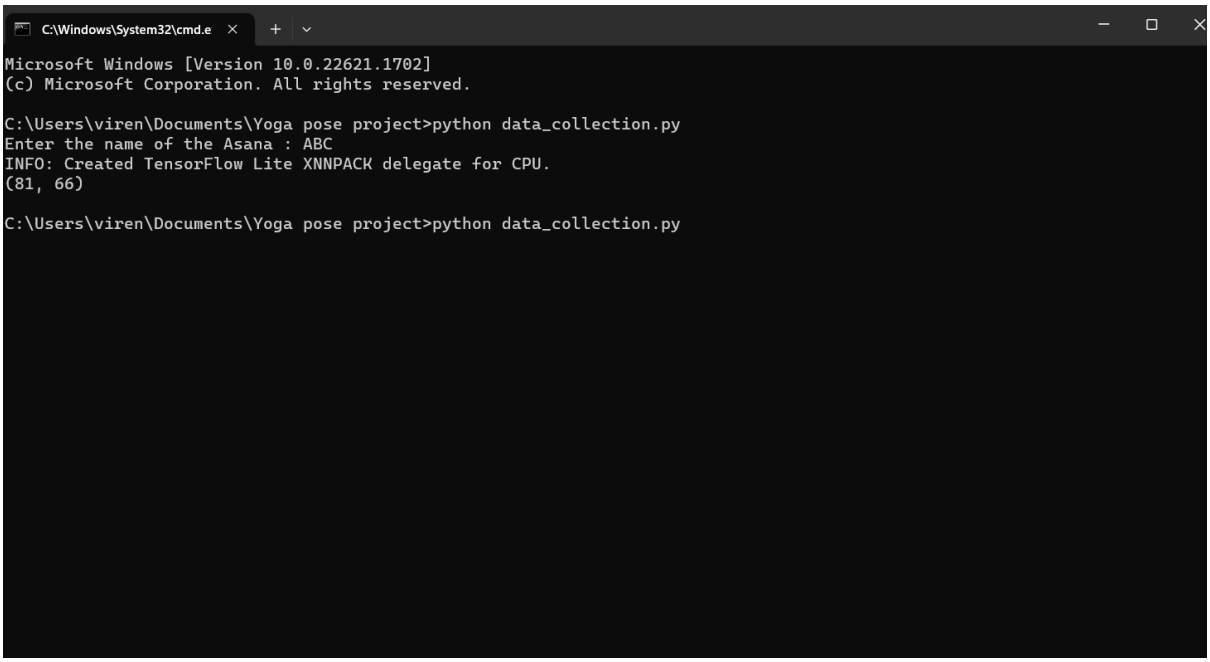
I was working on developing a robust system that can accurately detect and analyze yoga poses in real-time. By leveraging state-of-the-art deep learning models and cutting-edge algorithms, we have achieved remarkable results in terms of accuracy, efficiency, and versatility.

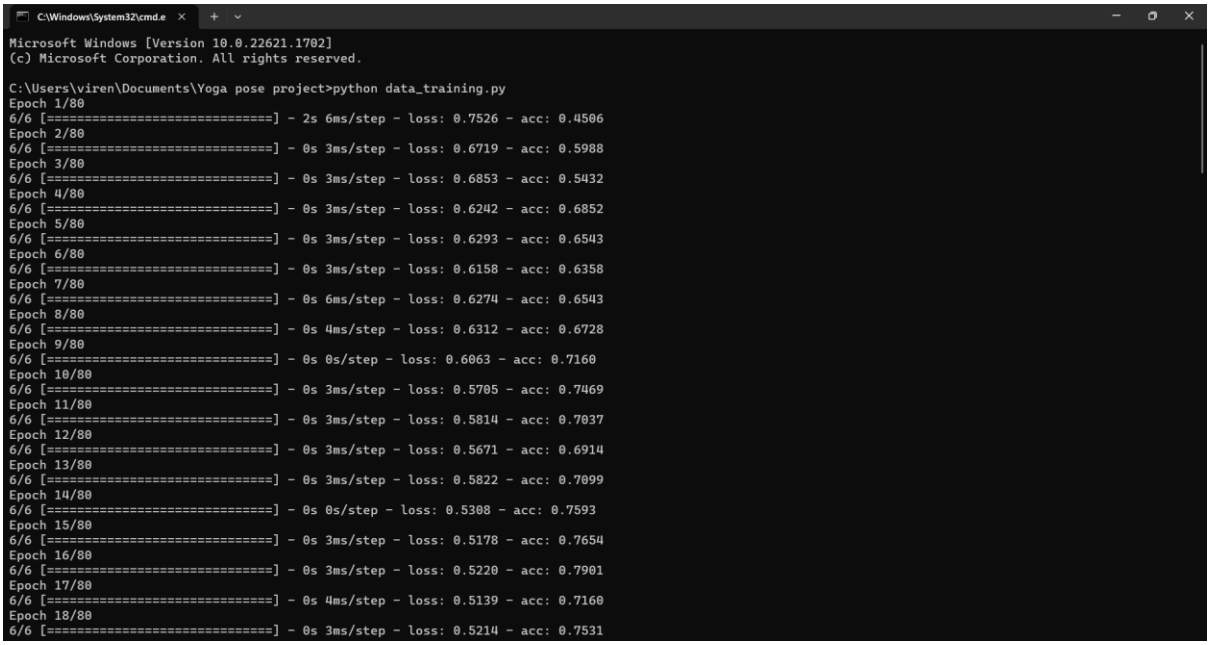
Our yoga pose detection solution aims to revolutionize the way yoga is practiced and instructed. It provides practitioners, instructors, and fitness enthusiasts with valuable insights and feedback to enhance their practice. With our system, individuals can receive real-time feedback on their posture, alignment, and balance, enabling them to improve their form, prevent injuries, and optimize their yoga experience.

Our system can be integrated into various platforms, including mobile applications, web-based tools, and even smart devices. This enables users to access personalized recommendations, track their progress over time, and receive tailored suggestions to advance their yoga practice.

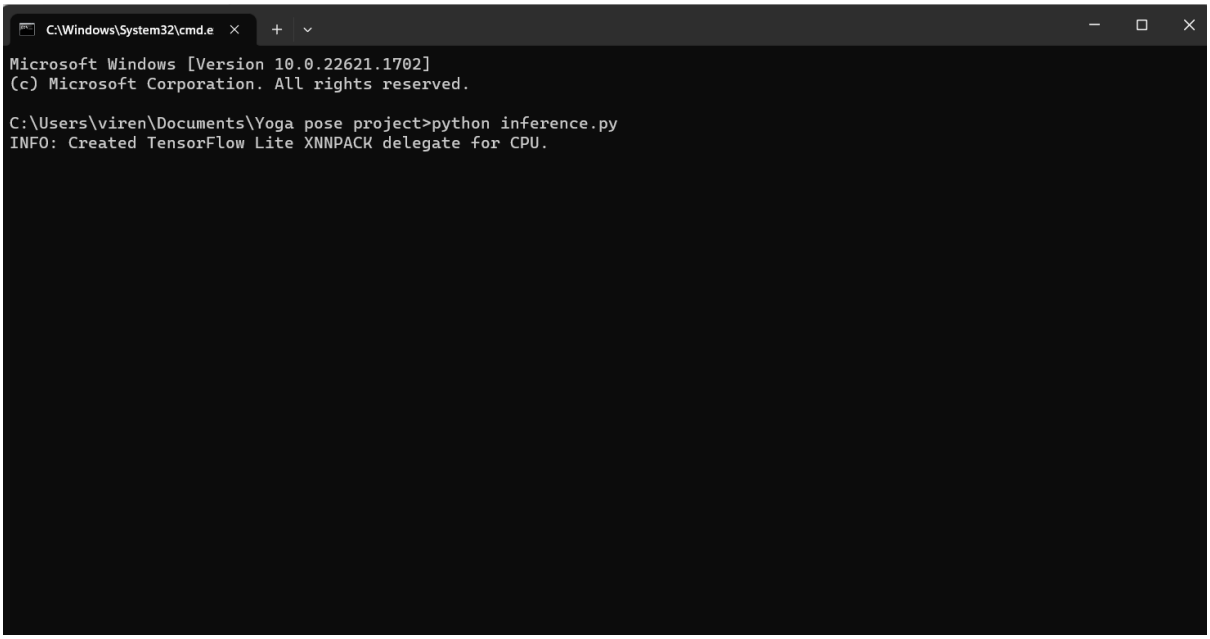
We would be delighted to explore how our yoga pose detection system can benefit your organization or how we can collaborate to further enhance its capabilities. We are open to discussing potential partnerships, licensing opportunities, or custom development based on your specific needs.

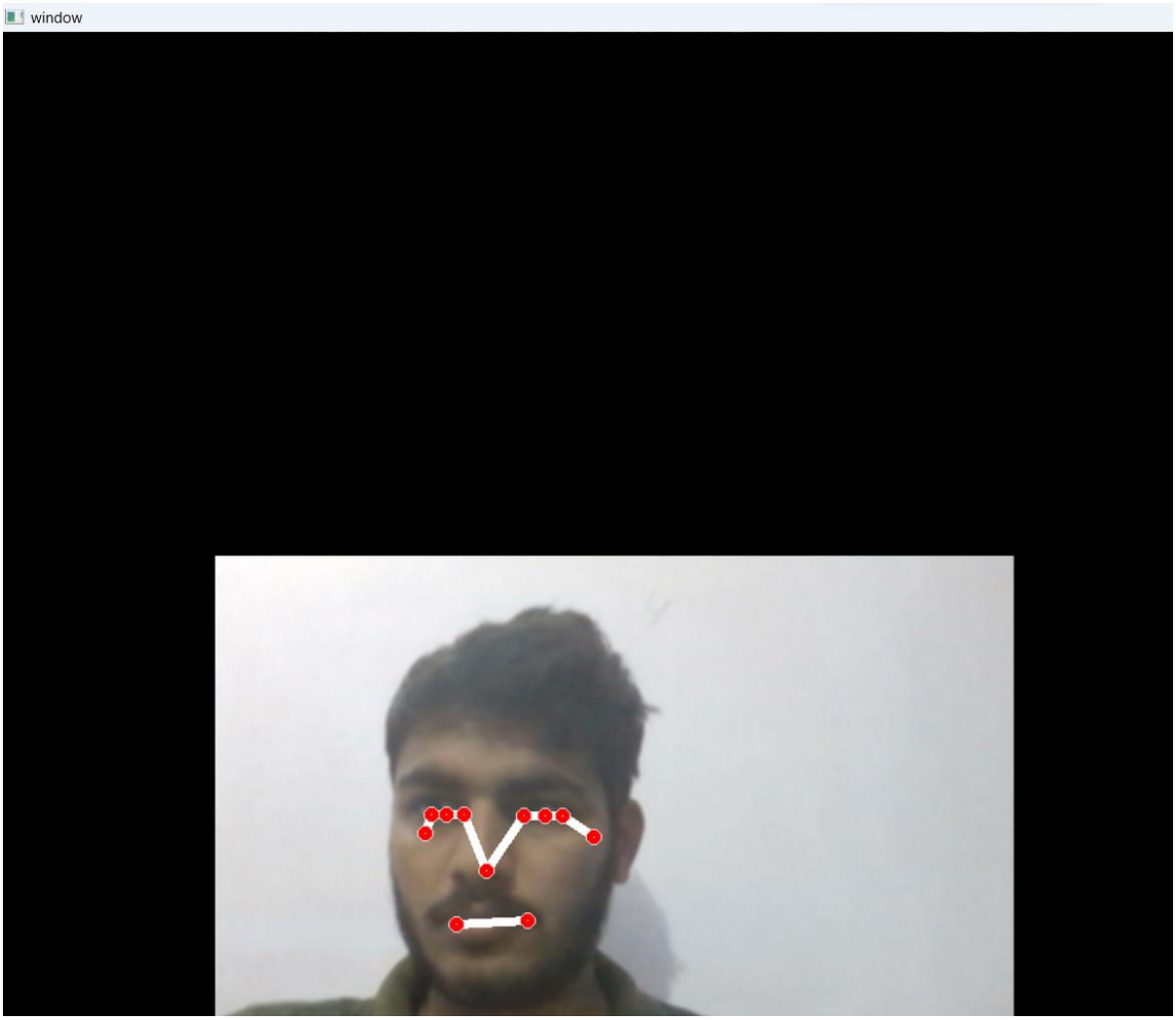
4.5 Important Screenshots:




C:\Windows\System32\cmd.e
Microsoft Windows [Version 10.0.22621.1702]
(c) Microsoft Corporation. All rights reserved.
C:\Users\viren\Documents\Yoga pose project>python data_training.py
Epoch 1/80
6/6 [==============================] - 2s 6ms/step - loss: 0.7526 - acc: 0.4506
Epoch 2/80
6/6 [==============================] - 0s 3ms/step - loss: 0.6719 - acc: 0.5988
Epoch 3/80
6/6 [==============================] - 0s 3ms/step - loss: 0.6853 - acc: 0.5432
Epoch 4/80
6/6 [==============================] - 0s 3ms/step - loss: 0.6242 - acc: 0.6852
Epoch 5/80
6/6 [==============================] - 0s 3ms/step - loss: 0.6293 - acc: 0.6543
Epoch 6/80
6/6 [==============================] - 0s 3ms/step - loss: 0.6158 - acc: 0.6358
Epoch 7/80
6/6 [==============================] - 0s 6ms/step - loss: 0.6274 - acc: 0.6543
Epoch 8/80
6/6 [==============================] - 0s 4ms/step - loss: 0.6312 - acc: 0.6728
Epoch 9/80
6/6 [==============================] - 0s 0s/step - loss: 0.6063 - acc: 0.7160
Epoch 10/80
6/6 [==============================] - 0s 3ms/step - loss: 0.5705 - acc: 0.7469
Epoch 11/80
6/6 [==============================] - 0s 3ms/step - loss: 0.5814 - acc: 0.7037
Epoch 12/80
6/6 [==============================] - 0s 3ms/step - loss: 0.5671 - acc: 0.6914
Epoch 13/80
6/6 [==============================] - 0s 3ms/step - loss: 0.5822 - acc: 0.7099
Epoch 14/80
6/6 [==============================] - 0s 0s/step - loss: 0.5308 - acc: 0.7593
Epoch 15/80
6/6 [==============================] - 0s 3ms/step - loss: 0.5178 - acc: 0.7654
Epoch 16/80
6/6 [==============================] - 0s 3ms/step - loss: 0.5220 - acc: 0.7901
Epoch 17/80
6/6 [==============================] - 0s 4ms/step - loss: 0.5139 - acc: 0.7160
Epoch 18/80
6/6 [==============================] - 0s 3ms/step - loss: 0.5214 - acc: 0.7531

Output:


C:\Windows\System32\cmd.e
Microsoft Windows [Version 10.0.22621.1702]
(c) Microsoft Corporation. All rights reserved.
C:\Users\viren\Documents\Yoga pose project>python inference.py
INFO: Created TensorFlow Lite XNNPACK delegate for CPU.


window

CHAPTER 5.

CONCLUSION AND FUTURE WORK

It has been a great pleasure for me to work on this exciting and challenging project. This project proved good for me as it provided practical knowledge of programming. It also about all handling procedures related to **"Yoga Pose Detection"**. It also provides knowledge about the latest technology used in developing Software-enabled applications that will be in great demand in the future. This will provide better opportunities and guidance in the future in developing projects independently. **BENEFITS:** The project is identified by the merits of the system offered to the user. The merits of this project are as follows: -

- It's a software-enabled project.
- It can provide real-time feedback on the accuracy and alignment of yoga poses.
- It can be used in conjunction with virtual yoga classes or instructional videos, providing an interactive and immersive learning experience.
- It can help identify misalignments, imbalances, or potentially dangerous movements, allowing practitioners to make necessary adjustments and reduce the likelihood of injury.
- It can help individuals monitor their progress by capturing data on their poses, alignment, and flexibility over time.
- It can be used at home, in studios, or even in remote locations without access to a yoga instructor.

- It can assist in detecting and correcting poor posture habits, which are common in our sedentary lifestyles.
- The data collected through pose detection systems can be analyzed to gain insights into how specific poses impact the body, muscles, and overall well-being.

LIMITATIONS:

- Yoga pose detection include challenges in accurately detecting complex poses with occluded body parts or variations in individual body shapes.
- Limited training data for diverse populations, lack of standardized pose definitions, and the need for real-time performance are also significant constraints.
- Noise and environmental factors can further impact detection accuracy.

Future Enhancements:

Yoga pose detection may include advancements in computer vision algorithms and machine learning models, leading to more accurate and real-time pose recognition. Integration of depth sensing technology, such as LiDAR or structured light, could provide better depth perception and finer pose details. Wearable sensors and haptic feedback systems may also enhance the user experience during yoga practice.

BIBLIOGRAPHY/REFERENCES

1. Jhuang, H., Gall, J., Zuffi, S., Schmid, C., & Black, M. J. (2013). Towards understanding action recognition. In International conference on computer vision (pp. 3192-3199).

2. Wei, S. E., Ramakrishna, V., Kanade, T., & Sheikh, Y. (2016). Convolutional pose machines. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 4724-4732).

3. Toshev, A., & Szegedy, C. (2014). DeepPose: Human pose estimation via deep neural networks. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 1653-1660).

4. Zhang, L., Zhu, H., & Dai, Q. (2019). YogaNet: A deep learning-based approach for yoga pose recognition and recommendation. IEEE Transactions on Multimedia, 21(3), 592-604.

5. Li, Z., Zhang, Z., & Wu, Y. (2021). Improved yoga pose recognition by exploiting temporal information from multi-view videos. IEEE Transactions on Multimedia, 23, 333-344.

6. Sharma, V., & Mittal, A. (2019). Human pose estimation: A survey. Computer Vision and Image Understanding, 184, 1-22.

7. Yin, G., He, X., Li, Y., & Tian, Y. (2019). Towards 3D human pose estimation in the wild: A weakly-supervised approach. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 610-619).

8. Vemulapalli, R., Arrate, F., & Chellappa, R. (2016). Human pose estimation via supervised learning. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 2144-2152).