# 3d Yoga Pose Estimation & Correction


Submitted By

**Bhupinder Kaur Punjabi**

(21MCED09)




**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING**

**INSTITUTE OF TECHNOLOGY**

**NIRMA UNIVERSITY**

**AHMEDABAD-382481**


**May 2023**

# 3d Yoga Pose Estimation & Correction

## Major Project - II

Submitted in partial fulfilment of the requirements

for the degree of

Master of Technology in Computer Science and Engineering (Data Science)

Submitted By

**Bhupinder Kaur Punjabi**

**(21MCED09)**

Guided By

**Prof. Jitali Patel**


**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING**

**INSTITUTE OF TECHNOLOGY**

**NIRMA UNIVERSITY**

**AHMEDABAD-382481**

**May 2023**

# Certificate

This is to certify that the major project entitled **"3D Yoga Pose Estimation & Correction"** submitted by **Bhupinder Kaur Punjabi (Roll No: 21MCED09)**, towards the partial fulfilment of the requirements for the award of degree of Master of Technology in Computer Science and Engineering (Specialization in Data Science) of Nirma University, Ahmedabad, is the record of work carried out by him under my supervision and guidance. In my opinion, the submitted work has reached a level required for being accepted for examination. The results embodied in this major project part-II, to the best of my knowledge, haven't been submitted to any other university or institution for award of any degree or d iploma.

Prof. Jitali Patel
Guide & Associate Professor,
CSE Department,
Institute of Technology,
Nirma University, Ahmedabad.

Dr. Madhuri Bhavsar
Professor and Head,
CSE Department,
Institute of Technology,
Nirma University, Ahmedabad.

Dr R. N. Patel
Director,
Institute of Technology,
Nirma University, Ahmedabad

# Statement of Originality

I, **Bhupinder kaur Punjabi**, Roll. No. **21MCED09.**, give undertaking that the Major Project entitled "**3D Yoga Pose Estimation & Correction**" submitted by me, towards the partial fulfilment of the requirements for the degree of Master of Technology in **Computer Science & Engineering (Data Science)** of Institute of Technology, Nirma University, Ahmedabad, contains no material that has been awarded for any degree or diploma in any university or school in any territory to the best of my knowledge. It is the original work carried out by me and I give assurance that no attempt of plagiarism has been made. It contains no material that is previously published or written, except where reference has been made. I understand that in the event of any similarity found subsequently with any published work or any dissertation work elsewhere; it will result in severe disciplinary action.

Signature of Student

Date:

Place:

Endorsed by

Prof. Jitali Patel

(Signature of Guide)

# Acknowledgements

It gives me immense pleasure in expressing thanks and profound gratitude to **Prof. Jitali Patel**, Associate Professor, Computer Engineering Department, Institute of Technology, Nirma University, Ahmedabad for his valuable guidance and continual encouragement throughout this work. The appreciation and continual support he has imparted has been a great motivation to me in reaching a higher goal. His guidance has triggered and nourished my intellectual maturity that I will benefit from, for a long time to come.

It gives me an immense pleasure to thank **Dr. Sanjay Garg**, Hon'ble Head of Computer Engineering/ Information Technology Department, Institute of Technology, Nirma University, Ahmedabad for his kind support and providing basic infrastructure and healthy research environment.

A special thank you is expressed wholeheartedly to **Dr R. N. Patel**, Hon'ble Director, Institute of Technology, Nirma University, Ahmedabad for the unmentionable motivation he has extended throughout course of this work.

I would also thank the Institution, all faculty members of Computer Engineering Department, Nirma University, Ahmedabad for their special attention and suggestions towards the project work.

**- Bhupinder Kaur Punjabi**

**(21MCED09)**

# Abstract

The ancient times, during 5000 years ago, the Indus-Sarasvati culture in ancient India created the yoga practise. The term "yoga" refers to a close connection and integration of the mind and the body. Through asana, meditation, and other practises, it is utilised to keep the body and mind in balance throughout all of life's ups and downs. Due of the growing stress levels in the modern lifestyle, yoga has recently attracted interest on a global scale. There are several ways to practise yoga. Many people choose self-learning in fast-paced environments because the aforementioned resources might not always be accessible. In our method, as input, we use the personalised yoga dataset.

The system uses various deep learning techniques to determine whether the input yoga stance image is correct or incorrect. Results display metrics for performance including accuracy, error rate, and comparison graph. There are several ways to accomplish this; however, the method I employ begins by using a CNN to process the incoming image and ANN classifier that has been trained to seek for humans. The pose estimation network looks for limbs and joints with training when it recognises human body positions. The user can then view the image on the computer using markers that designate various bodily components.

However, we did the implementation with the above mentioned CNN and ANN Machine learning algorithm the 2d detection of yoga poses by giving feedback to improve position of the body parts and detect the pose with classification.

We need a design and implementation on the basis of real-time detection and classification of the Yoga pose performed by the user. Hence, we will inculcate the previous algorithm and work on the real-time 3d pose estimation and correction by implementing yoga pose detection System which is designed and developed to recognize yoga stances and respond with a customized response to help users improve their postures. Our system will detect various yoga poses, namely Chair pose (Utkatasana), Cobra pose (Bhujangasana), Dog Pose (Adho Mukha Svanasana), Shoulder Stand Pose (Sarvangasana), Triangle Pose (Trikonasana), Tree Pose (Vrikshasana), Warrior Pose (Virbhadrasana) and No Pose.

# Abbreviations

| | |
|---|---|
| **CNN** | Convolutional Neural Network |
| **ANN** | Artificial Neural Network |
| **LBP** | Local Binary pattern |
| **XAMP** | Cross Platform Apache MySQL PHP |

# Table of Contents

# List Of Figures

# Chapter 1
# Introduction

## 1.1 General Introduction

Yoga positions must be performed correctly, as with any workout, as any incorrect posture is counterproductive and can even be harmful. This promotes doing yoga with a teacher nearby. Modern living makes it difficult to regularly practise yoga with a teacher or enrol in classes.

Recognising yoga postures is made easier with the use of AI technology, which also provides feedback and suggestions to practitioners. Using these recommendations, users may ensure that their postures are beneficial rather than detrimental. The project's hurdles include the need to identify important locations without any gaps and the need for models to function correctly even when body components are overlapping.

Since even small changes can have negative effects, suggestions should be made precisely. Experts should do the postures in the datasets used for this research. Even though they are practically identical positions with a small variation, models should classify poses appropriately. Athletes' performance and danger of injury can both be reduced by automated self-training methods. Many researchers have built computational methods to rank athletes in various sports and evaluate their performances in areas including football, handball, volleyball, sprinting, and leaping.

The article presented the "Yoga Tutor" initiative, which use accelerated robust qualities (SURF) to differentiate between a novice and an expert practitioner's yoga poses. introduced intelligent yoga systems that used pictures and text but didn't take the practitioner's posture into account.It developed a self-training system that recognised yoga poses using a features-based approach. It uses a Kinect to produce a body map by capturing the user's body contour. The

human skeleton was quickly modelled after a star skeleton so that a description of human posture could be generated..

Related research on yoga stance recognition and classification has been done recently. OpenPose, PoseNet, and PifPaf are employed as keypoint detection techniques. Many aspects, including the environment, interactions between people, and differences in apparel, will be taken into account when determining human pose. They employed deep learning techniques such multilayer perceptrons, recurrent networks, long short-term memories (LSTMs), and convolutional NNs to classify postures. Both the absence of scaled characteristics (key points) and the failure to discover a pattern for human postures at varying distances from the camera are problems with the aforementioned research. Traditional approaches that use joint angles as characteristics have the property of rotational invariance, which asserts that the angles of the joints do not vary regardless of how the joints are rotated.

## 1.2 Scope and Objective:

The scope of a Yoga Pose Detection System is to analyze images or videos of a person practising yoga and accurately identify the pose they are performing. The objective is to provide real-time feedback and guidance to the practitioner to improve their form and alignment, prevent injuries, and enhance their overall yoga experience.

A Yoga Pose Detection System can be useful for yoga practitioners at all levels, from beginners who are just learning the basics of yoga to experienced practitioners who want to refine their technique. The system can also be used by yoga teachers to monitor their student's progress and provide personalized feedback and corrections.

# Chapter 2
# Literature Survey

## 2.1 Techniques :- Literature Survey and Survey Table

**2.1.1 "Yoga Pose Estimation and Feedback Generation Using Deep Learning", 2022**. The rising stress levels associated with the contemporary way of life have contributed to yoga's recent surge in popularity throughout the world. There are several ways to practise yoga. Yoga can be acquired independently using resources including books, the Internet, and recorded videos, and personal tutors as well as in yoga studios and one-on-one settings. Many people choose self-learning in fast-paced environments because the aforementioned resources might not always be accessible. However, one might not discover an improper position when self-learning. Poor posture can have a negative impact on one's health because it can lead to both short-term acute discomfort and long-term chronic issues. In this study, methods based on deep learning are created to identify improper yoga posture.

The user pose is transmitted to models that are being trained to output any aberrant angles between the client present and the genuine posture. These results permit the framework to train the client on the most proficient method to perfect their yoga posture by drawing attention to any flaws in their form. While the suggested technique required less computational complexity than a few best in class strategies, it in any case accomplished an exceptional exactness of 0.9958. (Vivek*Anand Thoutam,1 Anugrah*Srivastava,1 Tapas*Badal,1 Vipul Kumar*Mishra,1 G. R. *Sinha,2 Aditi*Sakalle,3 Harshit*Bhardwaj,3 and Manish*Raj)

**2.1.2 "AI Human Pose Estimation: Yoga Pose Detection and Correction", 2022**.The most significant yoga stance is well-known on a global scale and supports the health advantages advocated by the ancient sages. Yoga faces the following significant difficulties as it gains popularity: Using computer vision technology, it is possible to evaluate how people are positioned. There are no explicit references or initiatives, and these methods are hardly ever employed in the contexts of fitness and health. Yoga-inspired in name. In accordance with how user-friendly your Android app application is, this white paper summarises the best ways to employ the various Pose estimation tools and technologies.

The next sections detail the method used to allow yoga posture prediction in Android*applications, the modelling of the software, and the operation of its constituent parts. The topic of computer vision known as "pose estimation" is dedicated to figuring out where different portions of a body (usually a human body) are. This may be done in a few different ways; the one I use involves first running the input picture through a convolutional neural network (CNN) classifier that has been trained to identify human faces. The posture estimation network searches for previously taught joints and limbs when it recognises human body poses. Once the picture has been captured, the user may examine it on a computer with the use of markers that indicate certain anatomical features (Rutuja, Gajbhiye, Snehal Jarag, Pooja, Gaikwad, Shweta, Koparde)

**2.1.3 "Yoga pose perfection using Deep Learning: An Algorithm to Estimate the error in Yogic Poses", 2021** The recognition of human body poses remains challenging despite significant efforts in the areas of computer vision and artificial intelligence. Human pose detection has several uses, including public security and health monitoring. This essay focuses on yoga, a discipline that has been practised for more than 2,000 years. Yoga has become a popular form of exercise in contemporary society, which has led to a desire for guidelines on how to do yoga correctly. Because practising some yoga postures incorrectly could lead to fatigue and injury, having a trainer there is essential.

Many individuals cannot afford to hire a yoga instructor or guide, so AI may step in and provide advice on which positions to take. The primary focus of research on position estimate for yoga right now is the taxonomy of yogic poses. In this study, we offer a method for detecting posture errors in real time and allowing people to repair them using the Tensorflow MoveNet Thunder model**(** Satyam Goyal1 and Animesh Jain ).

**2.1.4 "Deep Learning Models for Yoga Pose Monitoring",** 2022. Continuously tracking a person's movement and activity is known as activity recognition. A self-guidance practise that enables people to learn and practise yoga poses without assistance from others can be built using human posture recognition. We offer a method for the quick and accurate identification of different yoga poses using deep learning algorithms. The 85 films in the selected dataset, which includes 6 yoga poses carried out by 15 people, were used to extract key user points using the Mediapipe library. A deep learning model including Convolutional Neural Network (CNN) and Long Short-Term Memory (LSTM) has been used to detect yoga postures in live-streamed movies. Those involved (Debabrata,Swain,1, Santosh, Satapathy,1,

Biswaranjan, Acharya 2, Madhu*Shukla 3. Vassilis C. Gerogiannis 4. Andreas Kanavos 5. Dimitris Giakovis 6.).

**2.1.5 "Human Yoga pose estimation",**In the beginning, Iskakov et al. (2019) provide a base methodology that provides all created views' 2D locations and joint confidences in 2 dimensions using the algebraic triangulation module and the 2D posture detector, you can produce the 3D position. The disadvantage of this approach is the independent processing of images from various cameras. Consequently, a more potent they suggest a triangulation approach. Processing involves the 3D volumes from feature maps are not projected, and the volumes from a 3D CNN gathers and analyses various viewpoints to produce 3D heatmaps.

**2.1.6 "Recognition and detection of yoga pose estimation",** challenges as per research paper "CVPR 2018 for 3d pose estimation includes".As we know that the humans consists of differences and variety in appereances and also the images comes in all kinds of different appreances and also the algorithm has to deal with many kinds of occlusions.2d pose estimation can be inaccurate particularly in complex motions or scene.Occlusions: Consisting of the frame issues , the background noise, interperson occlusion, out of frame.Limited Data: Limited no. of posesBad input data:: Blurry, low resolution, small scale, low contrast, low light, noisy. Multi-person pose estimate would be further complicated by a number of elements, including but not limited to the topics covered so far in this poll backdrop, lighting, overlapping figures, etc.

## 2.2 Summary:

| Title | Year | Research Gaps | Methodology |
|---|---|---|---|
| "Deep Learning Models for Yoga Pose Monitoring" | 2022 | Lack of Integration and customisation of Specific Libraries. | Convolutional Neural Networks (CNNs) and Long Short-Term Memories (LSTMs) have been combined for the task of recognising yoga postures. |
| "Yoga pose perfection using Deep Learning: An Algorithm to Estimate the Error in Yogic Poses" | 2021 | Dependency on large-scale annoted datasets. Less effective and training time is high. | Using the Tensorflow MoveNet Thunder model, we propose a system that can identify posture issues in real time and provide users the opportunity to correct them.. |
| "AI Human Pose Estimation: Yoga Pose Detection and Correction" | 2022 | Limited focus on real-time Performance. No, Except for training time interval duration is much higher | My method, on the other hand, first processes the input picture using a convolutional neural network classifier that has been specifically trained to look for people.. |
| "Yoga pose Estimation and Feedback Generation Using Deep Learning" | 2022 | The process is implemented without removing unwanted data. | Graphical Convolutional Network and Deep Reinforcement Neural Network |
| "Human Yoga pose estimation". | 2021 | Lack of Personalised feedback generation techniques. | Tensorflow movenet Thunder model |
| "Recognition and detection of yoga pose estimation" | 2018 | It basically deals with only the depth, color and the body tracking. | Kinect V2 Technology and adaboost algorithm is used |

# Chapter 3

## 3.1 Existing System:

The most significant yoga stance in the current system is well-known on a global scale and supports the health advantages advocated by the ancient sages. Yoga faces the following significant difficulties as it gains popularity: Using computer vision technology, it is possible to evaluate how people are positioned. There are no explicit references or initiatives, and these methods are hardly ever employed in the contexts of fitness and health. Yoga-inspired in name. In accordance with how user-friendly your Android app application is, this white paper summarises the best ways to employ the there are numerous technologies available for pose estimation. The next sections detail the method used to allow yoga-pose prediction in Android apps, the modelling of the software, and the operation of its many components. Pose estimation is a branch of computer vision concerned with determining where on a body (often a human body) certain objects are located. The traditional approach didn't provide real-time detection opportunities. The existing system mainly includes Convolutional Neural network with Long short Term methodology, Graph convolutional Network, Recurrent neural network, with Tensorflow and other backstaged libraries which do not support good integration with the real-time prediction as accurate and shows less indulgence strategically in 3d.

## 3.1.2 Proposed System:

The dataset of yoga poses was used as input for this system. The source of the input data was a dataset repository like Github. The preprocessing phase can then be implemented. We can resize the original image and convert it to grayscale in this step. Then, we can extract features such as mean, variance, median, and local binary pattern from the pre-processed image (LBP). After that, we can separate the images into train and test sets, with the latter being utilised for evaluation. After that, numerous deep learning techniques, such as a Convolutional Neural Network (CNN) or an Artificial Neural

Network (ANN), may be used. The findings of the experiment demonstrate the accuracy and effectiveness of predicting whether the input image of yoga is correct or incorrect.

However for our future scope of implementation we have triggered towards the system, Our Yoga Pose Detection System consists of 1 module: User. They can either upload a picture of a Yoga pose or pose directly on camera and the system will automatically detect and show the name of the yoga pose. The correction of poses will be done by giving the user a confidence score of how accurately they are performing the pose.

The system will detect 8 types of Yoga poses Chair, Cobra, Dog, Shoulder Stand, Triangle, Tree, Warrior and No Pose. OpenCV, dlib, OpenPose, and MediaPipe are the libraries utilised in this project. OpenPose is the main ongoing group oriented framework that can perceive the human body, foot, hand, and face key points in a single picture.

The framework used in project is Django. The Front End uses JavaScript, CSS, and HTML. The MySQL database is used in the back end. Python serves as the back end language.

## ADVANTAGES:

- It works well with a lot of datasets.
- When compared to the current system, the experimental outcome is excellent.
- To increase the performance metrics results.

## 3.2 Dataset Overview:

- Here we're creating our own dataset repository by collecting the various from different google 3d images.
- It consists of wrong 3d pose yoga images and correct poses.
- We have taken some random poses mostly it consists of plank pose and downdog pose to train the dataset and get the correct output with predictions on it with two different algorithm ,i,.e CNN and ANN.
- Eg., of the dataset division is as follow:
- Dataset Created Manually Using google 3d Image set and github posenet repository:

  Total Number of instances: 3123

  Images with training and testing ratio of 70:30.

  The dataset consist of Training and testing data split for Men and Women both equally, with 7 different poses in several different angles.

  Chair pose (Utkatasana): Women Image:568 Instances

  Cobra pose (Bhujangasana): 632 Instances

  Dog Pose (AdhoMukho Svanvasana): 580 Instances

  Shoulder Stand Pose (Salamba Sarvagasana): 60 Instances

  Triangle Pose (Trikonasana): 55 Instances

  Tree Pose (Vrikshasana) 610 Instances

  Warrior pose (Virbhadrasana) : 618 Instances

- Input Images:- For random selection which consist of all the given mixed data input images.
- Some of the dataset Images taken from github Repository dataset Reference: https://www.github.com/datasets/github/posenet and some created manually as in downloaded from the google images.

# Chapter 4: Testing Techniques for Real-Time Implematation

## 4.1 Testing On CNN and ANN models :

We will test the output for both CNN and ANN on the basis of two Input data i.e., Input data-1, Input data-2 to test which algorithm would give best performance result for real-time implementation so as we can incorporate, it learns complex patterns and we will use both the data one for correct image accuracy and one for incorrect image and feedback generating capacity.

### Step :1 IMPORT PACKAGES

### Step :2 READ INPUT IMAGE

### For Input data: 1


### For Input data: 2

## Step :3 IMAGE PREPROCESSING

- As part of our procedure, we must downsize the image and turn it into grayscale.
- Resize an image by using its resize () function and passing it a two-integer tuple argument with the new width and height.
- Instead of modifying the original picture, the method returns a copy of the image with the new dimensions.
- Transform a colour picture into grayscale in Python using the Matplotlib Library and the Conversion Formula.
- We may likewise switch a picture over completely to grayscale utilizing the standard RGB to grayscale change strategy, imgGray = 0.2989 * R + 0.5870 * G + 0.1140 * B.

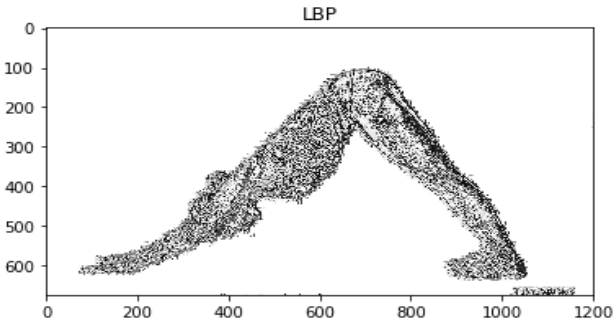
### For Input Data-1:


### For Input Data-2:

## Step :4 Feature Extraction

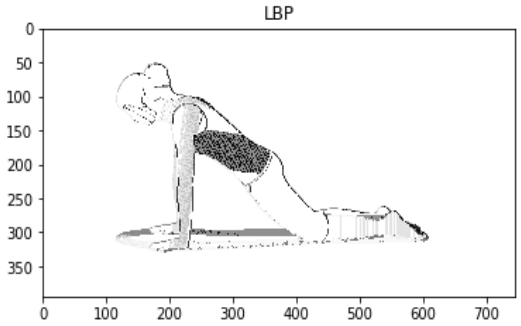
- Alongside the standard deviation (which is only the square base of the change), it is the most considered normal utilized proportion of scattering.
- The variance is calculated by taking the square root of the average distance from the centre of the distribution to each individual data point.
- Labelling pixels in an image by thresholding the region surrounding them and then processing the outcome as a paired whole number is the occupation of the Nearby Twofold Example (LBP) finishing administrator.

### For Input Data-1:




```
---------------------------------------------
FEATURE EXTRACTION --> MEAN, VARIANCE, MEDIAN
---------------------------------------------

1. Mean Value     = 231.1368

2. Median Value   = 255.0

1. Variance Value = 3139.8636857599995

-------------------------------------------------
FEATURE EXTRACTION --> LOCAL BINARY PATTERN (LBP)
-------------------------------------------------
```

### For Input Data-2:




```
---------------------------------------------
FEATURE EXTRACTION --> MEAN, VARIANCE, MEDIAN
---------------------------------------------

1. Mean Value     = 0.95092833

2. Median Value   = 1.0

1. Variance Value = 0.024155723

-------------------------------------------------
FEATURE EXTRACTION --> LOCAL BINARY PATTERN (LBP)
-------------------------------------------------
```

## Step :5 Image Splitting

- Data are required for machine learning to function, and not only for training purposes but also for testing the algorithm's results and gauging its efficacy.
- We split the input dataset in two: 70% was used for training, and 20% was used for testing.
- In order to use a cross-validator, it is common practise to divide the available data in half.
- Information is parted into two sections: The first is used to construct a prediction model, and the second is used to assess how well the model worked.
- A crucial first step in evaluating data mining techniques is to divide the data into a training set and a testing set.

**For Input Data-1:**

```
-----------------------------------
      IMAGE SPLITTING
-----------------------------------

Total no of data       : 50
Total no of train data   : 40
Total no of test data  : 10
Using TensorFlow backend.
```

**For Input Data-2:**

```
-----------------------------------
      IMAGE SPLITTING
-----------------------------------

Total no of data       : 50
Total no of train data   : 40
Total no of test data  : 10
Model: "sequential_7"
```

## Step :4 Classification :

- We can use various deep learning algorithms, such CNN and ANN, in our method.
- Convolutional neural networks (CNNs) are used in image recognition and other pixel-intensive applications because they are a specialised network design for deep learning methods.
- Although many different types of neural networks are used in deep learning, convolutional neural networks (CNNs) are the top choice when it comes to object recognition and id.
- Classification An ANN's goal, given a set of inputs, is to assign an observation to one of many possible classes. Categories are required for the dependent variable, whereas numbers are acceptable for the input characteristics (independent variables).

## For Input Data-1:

### CNN Algorithm Output

```
-------------------------------------
CONVOLUTIONAL NEURAL NETWORK (CNN)
-------------------------------------

WARNING:tensorflow:From C:\Users\Bhupinder kaur\anaconda99\lib\site-packages\tensorflow
\python\ops\math_grad.py:1250: add_dispatch_support.<locals>.wrapper (from
tensorflow.python.ops.array_ops) is deprecated and will be removed in a future version.
Instructions for updating:
Use tf.where in 2.0, which has the same broadcast rule as np.where
WARNING:tensorflow:From C:\Users\Bhupinder kaur\anaconda99\lib\site-packages\keras
\backend\tensorflow_backend.py:422: The name tf.global_variables is deprecated. Please
use tf.compat.v1.global_variables instead.

Epoch 1/5
40/40 [==============================] - 3s 86ms/step - loss: 0.4990
Epoch 2/5
40/40 [==============================] - 1s 14ms/step - loss: 0.4955
Epoch 3/5
40/40 [==============================] - 1s 13ms/step - loss: 0.4917
Epoch 4/5
40/40 [==============================] - 1s 14ms/step - loss: 0.4882
Epoch 5/5
40/40 [==============================] - 1s 13ms/step - loss: 0.4849
40/40 [==============================] - 0s 10ms/step
-------------------------------------
PERFORMANCE ---------> (CNN)
-------------------------------------

1. Accuracy   = 95.01048739254475 %

2. Error Rate = 4.989512607455254
-------------------------------------
```

**ANN Algorithm output:**

```
-------------------------------------
ARTIFICIAL NEURAL NETWORK (CNN)
-------------------------------------

Epoch 1/5
40/40 [==============================] - 1s 21ms/step - loss: 1.5977
Epoch 2/5
40/40 [==============================] - 0s 2ms/step - loss: 1.5927
Epoch 3/5
40/40 [==============================] - 0s 2ms/step - loss: 1.5876
Epoch 4/5
40/40 [==============================] - 0s 1ms/step - loss: 1.5826
Epoch 5/5
40/40 [==============================] - 0s 990us/step - loss: 1.5776
40/40 [==============================] - 0s 6ms/step
-------------------------------------
PERFORMANCE ---------> (ANN)
-------------------------------------

1. Accuracy   = 84.02333840727806 %

2. Error Rate = 15.976661592721939
```

## For Input Data-2:

## CNN Output:

```
-------------------------------------
CONVOLUTIONAL NEURAL NETWORK (CNN)
-------------------------------------

Epoch 1/5
40/40 [==============================] - 2s 44ms/step - loss: 0.4990
Epoch 2/5
40/40 [==============================] - 0s 12ms/step - loss: 0.4957
Epoch 3/5
40/40 [==============================] - 0s 11ms/step - loss: 0.4921
Epoch 4/5
40/40 [==============================] - 0s 12ms/step - loss: 0.4889
Epoch 5/5
40/40 [==============================] - 0s 11ms/step - loss: 0.4856
40/40 [==============================] - 0s 10ms/step
-------------------------------------
PERFORMANCE ---------> (CNN)
-------------------------------------

1. Accuracy   = 95.01019039750099 %

2. Error Rate = 4.989809602499008
```

## ANN output:

```
-------------------------------------
ARTIFICIAL NEURAL NETWORK (CNN)
-------------------------------------

Epoch 1/5
40/40 [==============================] - 1s 17ms/step - loss: 1.5978
Epoch 2/5
40/40 [==============================] - 0s 634us/step - loss: 1.5927
Epoch 3/5
40/40 [==============================] - 0s 646us/step - loss: 1.5877
Epoch 4/5
40/40 [==============================] - 0s 657us/step - loss: 1.5827
Epoch 5/5
40/40 [==============================] - 0s 953us/step - loss: 1.5777
40/40 [==============================] - 0s 6ms/step
-------------------------------------
PERFORMANCE ---------> (ANN)
-------------------------------------

1. Accuracy   = 84.02247586846352 %

2. Error Rate = 15.977524131536484
```

## Step :4 Prediction and Performance :

### For Input data 1:

**PERFORMANCE ---------> (CNN)**

- Accuracy = 95.01048739254475 %
- Error Rate = 4.989512607455254

**PERFORMANCE ---------> (ANN)**

- Accuracy = 84.02333840727806 %
- Error Rate = 15.976661592721939

- Prediction Result: Here the Input image show the **correct** yoga posture with the excellent accuracy including the pose detected as: **Downdog**

### For Input data 2:

**PERFORMANCE ---------> (CNN)**

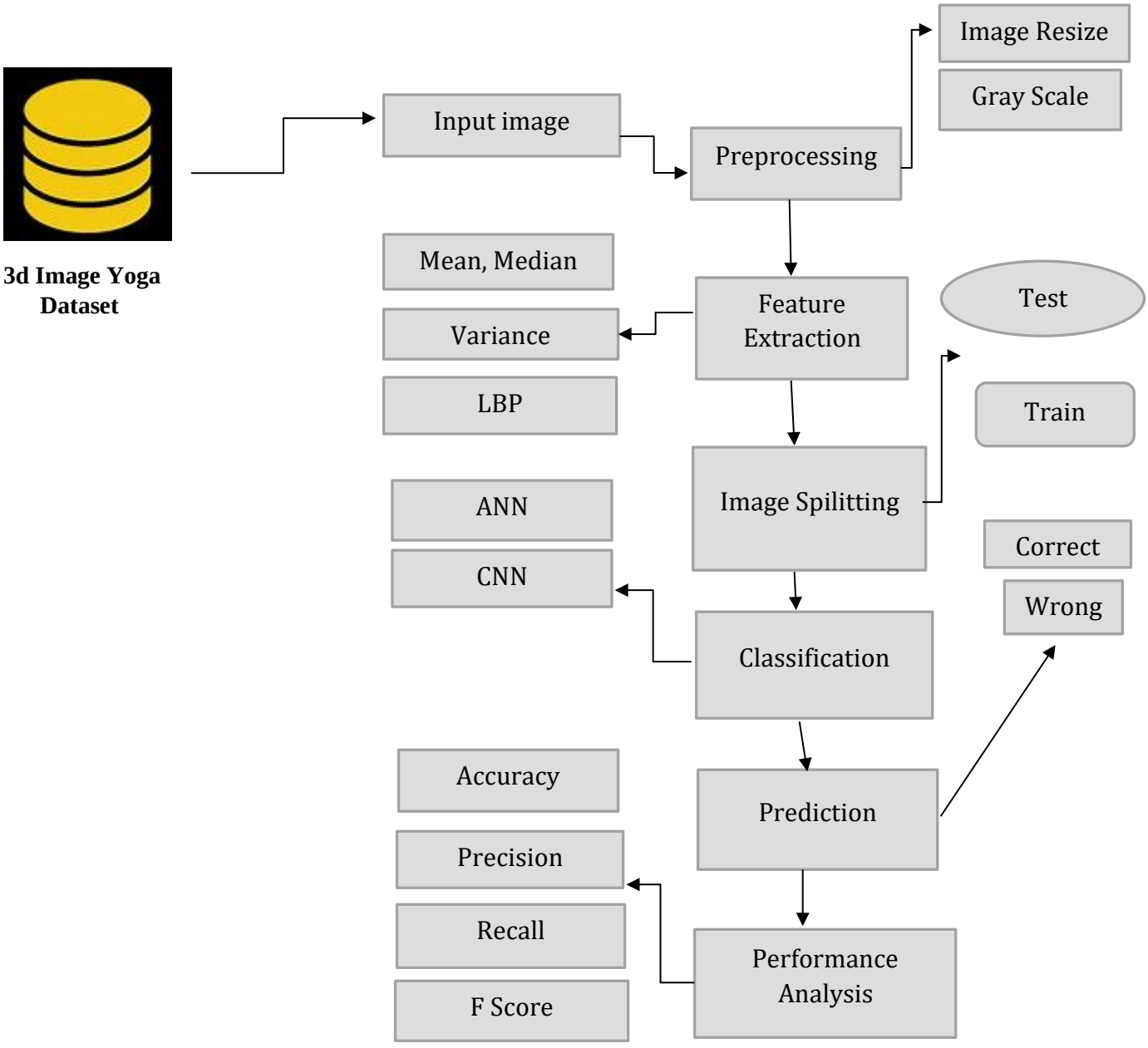
- Accuracy = 95.01019039750099 %
- Error Rate = 4.989809602499008

**PERFORMANCE ---------> (ANN)**

- Accuracy = 84.02247586846352 %
- Error Rate = 15.977524131536484

- Prediction Result: Here the Input image show the Incorrect yoga posture with the excellent accuracy including the pose justification as for why is it predicted as wrong pose: wrong plank pose

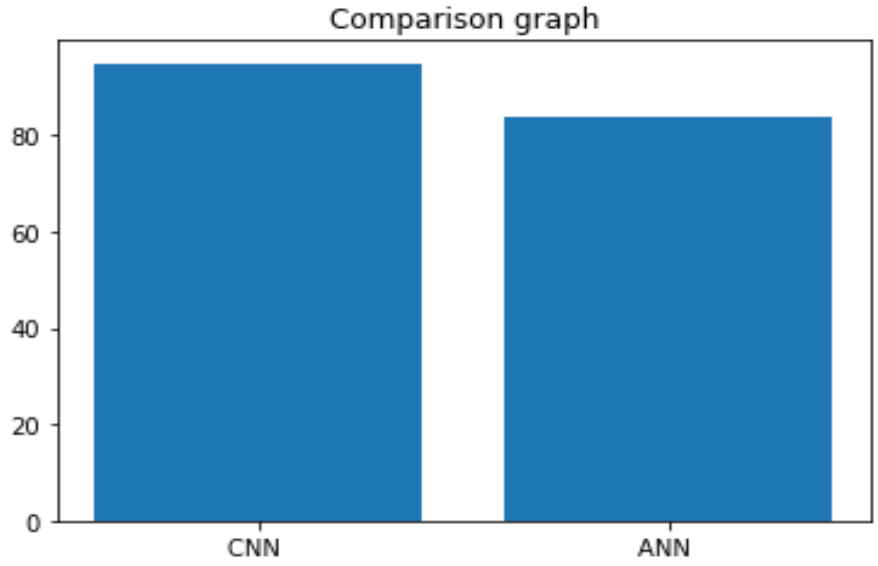
## 4.2 The Flow Diagram of the proposed Testing:



### In Accordance to Fig 4.2 the flow diagram Explains:

- Firstly, we need to import the packages, the packages that we require in our project.
- Then we need to read the input images.
- Resizing the images.
- Gray scale conversion
- Then we can extract the features from every images as every images have unique features by using the LBP
- After that splitting the images from total number of images into train and test images.
- Then we've implemented the two different algorithms ie., CNN and ANN.
- Thus,by using both the algorithms with the help of the trained data we can estimate the performance.
- Then finally we can predict the pose that we have given as input and also to know whether the image is in correct posture and if not then by giving the justification that why is it a wrong image and the pose is incorrect to perform in real-time.
- Then finally,we can conclude with the comparison graph of ANN and CNN.

## 4.3 :- Comparison Graph:



## Explanation:

Here, the comparison graph Fig 4.3 shows two best fitted algorithm to detect the 3d yoga pose image with the prediction up to 93-95% and shows high accuracy in both the terms with the minimal amount of error rate but as compared to ANN, the CNN algorithm shows higher accuracy in aspects of low error rate and higher accuracy. Hence, these shows the solution to the good and efficient prediction in future prediction as the dataset will be improved in higher amount capacity, as compared to now as shown in the implementation.

This system was proposed for efficient yoga pose detection using deep learning algorithms CNN and ANN. Analysis of the experimental data revealed that, when compared to the state of the art, our proposed strategy is effective and can, on average, produce better performance results. Additionally, the CNN performance was examined, and the comparison graph demonstrates CNN's superior accuracy.

Thus by these techniques we're able to predict the poses and eliminate the incorrect postures in which the body can't relax and we should not perform the certain pose in a particular wrong manner.

## 4.3.1 Testing Concluded:

- Hence as per the testing methodology on two different deep-learning models CNN and ANN algorithms , we have concluded that CNN is best giving resolution on accuracy of the Input data. So we will Incorporate CNN Algorithm with the Mediapipe Library and Dlib for the Real time 3d Implementation in chapter 5.
- As CNN in deep learning was specifically designed for its image data working which excels in recognition of patterns from image dataset.
- It also possess parallel computing feature best suited model for human pose estimation.
- By doing away with manual feature engineering, the development process is made simpler.

- It reduces our need from doing it from scratch as it incorporates many pretrained popular features in architectures such as Resnet , mobilenet, imagenet, etc.

- CNN and ANN models are very much compatible as compared to other models in deep learning for integration and customise the output as per our need.

- CNN and ANN have been employed successfully in a variety of pose estimation applications such as human pose estimation, facial landmark identification, and body keypoint localization.

- CNNs have showed greater performance in these domains when compared to other algorithms, making them a reliable choice for our purpose.

# Chapter 5

## 5.1 Implementation Strategy:

Hence, in conclusion to chapter-4 testing our model, combining a CNN model with OpenCV, Dlib, and Mediapipe combines the benefits of deep learning, computer vision techniques, and optimised frameworks to enable precise and effective stance prediction in real-time yoga scenarios.

## Project Implementation Technology

The Django Framework is used to design and develop the project. The project's coding was done using the Django Framework. We created and maintained all databases on the MySQL Server, writing queries to store data and keep track of projects.

### I. Hardware Requirement

#### i. Laptop or PC

- Windows 7 or higher
- I3 processor system or higher
- 4 GB RAM or higher
- 100 GB ROM or higher

### II. Software Requirement

#### i. Laptop or PC

- Python
- Sublime text Editor
- XAMP Server

## XAMPP: Cross Platform Apache MySQL PHP

Cross-Platform: Different local systems have various operating system installation configurations in them. Cross-platform has been included to this package of Apache distributions in order to expand its functionality and user base. It supports a number of operating systems, including versions of Windows, Linux, and Mac OS.

An open-source web server solution package is called XAMPP. It mostly serves as a local host webserver for testing web applications.

## Django documentation :

Django is a Python web framework written at a higher level that encourages rapid iteration and clean, sensible code. It was created via prepared software engineers and deals with a ton of the dreary work engaged with Web improvement, so you can zero in on making your application as opposed to wasting time. Free and open-source.

## Features of Django

- Rapid Development
- Secure
- Scalable
- Fully loaded
- Versatile
- Open Source
- Vast and Supported Community

## ❑ Modules and their Description

### ❖ User:

- **Upload:**
  - The algorithm will recognise the yoga stance based on the user's uploaded photo of the pose.

- **Real-time :**

- The system will recognise a yoga posture if the user strikes it in front of the laptop camera and display the appropriate label.
- The correction of poses will be done by giving the user a confidence score of how accurately they are performing the pose.

- **Drawback :**

- The accuracy of existing Yoga Pose Detection Systems is tilted depending on the complexity of the pose.

## 5.2 Proposed System Overview:

- Our Yoga Pose Detection System consists of 1 module: User. They can either upload a picture of a Yoga pose or pose directly in front of a camera and the system will automatically detect and show the name of the yoga pose. The correction of poses will be done by giving the user a confidence score of how accurately they are performing the pose.

- The system will detect 8 types of Yoga poses Chair, Cobra, Dog, Shoulder Stand, Triangle, Tree, Warrior and No Pose. This undertaking makes use of the OpenCV, dlib, OpenPose, and MediaPipe libraries. OpenPose is the first multi-person system that works in real-time and can recognise key locations on the human body, foot, hand, and face characteristics in a single picture.

- The framework used in this project is Django. The Front End uses JavaScript, CSS, and HTML. The MySQL database is used in the back end. The Back End Language is Python.

- We may utilise the strength of deep learning models, take advantage of established libraries and their capabilities, and benefit from earlier triumphs in pose estimation tasks by selecting CNNs and ANNs and integrating them with Mediapipe, Dlib, and OpenCV. This combination enables us to create a dependable, accurate, and efficient system for 3D yoga pose estimation and feedback.

# 5.3 PROJECT DESIGN

## Sequence Diagram

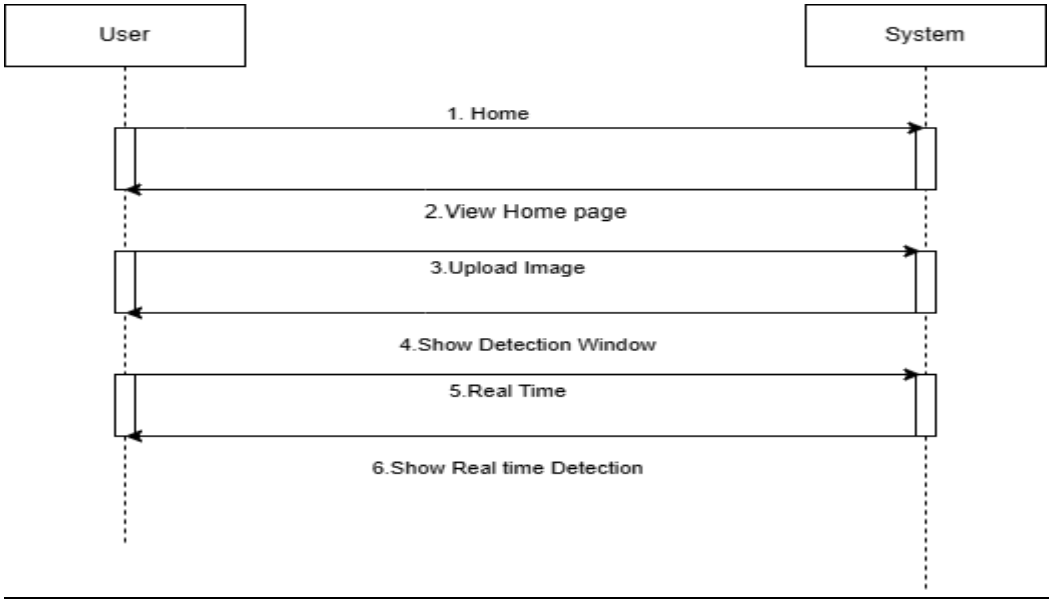



**Fig** 5.3.1 **Sequence Diagram of User**

## Activity Diagram

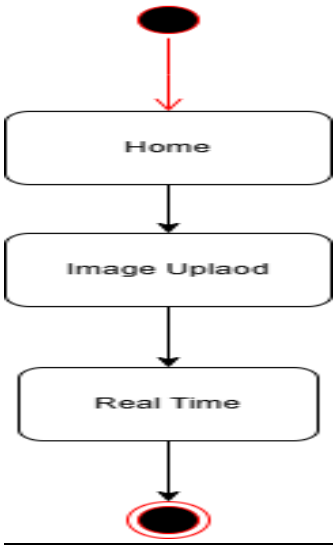



**Fig.** 5.3.2 **Activity Diagram of User**

All the Fig., 5.3.1, 5.3.2, are made for a better visual representation of the Proposed system outlook. Overall, these diagrams play a crucial role in understanding the project's functionality,

system behavior, user interactions, data model, and overall structure. They provide a visual representation that aids in effective communication, requirement analysis, system design, and development of the 3D yoga pose estimation and feedback system.

## 5.4 Data Flow:

### 5.4.1 The Explanation on the Data flow of the Project Implementation:

**User Image Upload:** The user uploads an image of themselves performing a yoga pose through the user interface.

**Image Processing:** The uploaded image is processed using libraries like OpenCV, Dlib, and Mediapipe. These libraries enable pose estimation and feature extraction from the image.

**Pose Estimation:** The system applies computer vision techniques, such as CNN-based models, to estimate the user's pose based on the processed image. The pose estimation algorithm identifies key body joints and their positions in 3D space.

**Pose Comparison:** The estimated pose is compared to a reference database or model that contains predefined or ideal yoga poses. The system calculates a confidence score to assess how accurately the user is performing the pose.

**Feedback Generation:** Based on the comparison and confidence score, the system generates feedback for the user. The feedback can be in the form of a pose name, visual cues, or textual instructions, indicating how well the user is executing the pose.

**Real-Time Pose Detection:** If the system operates in real-time mode, in front of the laptop camera, the user can strike the desired position. The system continuously processes the camera feed, detects the user's pose using similar techniques as in the image processing stage, and provides real-time feedback accordingly.

**User Interface:** The user interface, developed using HTML, CSS, and JavaScript, presents the feedback and other relevant information to the user. It allows users to interact with the system, upload images, view real-time feedback, and access additional features.

Creating the dataset manually implies capturing images or videos and labeling them with the corresponding pose names. The CNN-based model for pose estimation is then trained using these labelled instances. Framework: Django is utilized for the web application development, providing a structure and components to handle user interactions, backend processing, and database integration.

In summary, the system enables users to upload images or perform poses in front of a camera, detects the poses using pose estimation techniques, compares them to reference poses,generates feedback based on the comparison, and stores relevant data for analysis and future reference.

## 5.5 Algorithm of the Implementation:

```
# Import necessary libraries
import cv2
import dlib
import mediapipe

# Image Upload Mode
def processUploadedPoseImage(image):
    processedImage = preprocess(image)
    estimatedPoses = []
    for pose in processedImage:
        estimatedPose = estimatePose(pose)
        estimatedPoses.append(estimatedPose)
    confidenceScores = comparePose(estimatedPoses)
    feedback = generateFeedback(confidenceScores)
    displayFeedback(feedback)

# Real-Time Mode
def processRealTimePoseFrames(frames):
    processedFrames = preprocess(frames)
    estimatedPoses = []
    for frame in processedFrames:
        estimatedPose = estimatePose(frame)
        estimatedPoses.append(estimatedPose)
    confidenceScores = comparePose(estimatedPoses)
    feedback = generateFeedback(confidenceScores)
    displayFeedback(feedback)
```

```
# Preprocessing
def preprocess(data):
    processedData = []
    for item in data:
        # Apply necessary preprocessing steps to item (e.g., image stabilization, noise reduction, segmentation)
        processedItem = preprocessItem(item)
        processedData.append(processedItem)
    return processedData

# Pose Estimation
def estimatePose(item):
    # Use computer vision techniques or pre-trained pose estimation models (e.g., using OpenCV, Dlib, Mediapipe)
    # Extract key body joints and estimate their positions in 3D space
    estimatedPose = estimatePoseItem(item)
    return estimatedPose

# Pose Comparison
def comparePose(poses):
    confidenceScores = []
    for pose in poses:
        # Compare the pose with a reference database or model
        confidenceScore = comparePoseWithReference(pose)
        confidenceScores.append(confidenceScore)
    return confidenceScores

# Feedback Generation
def generateFeedback(confidenceScores):
    feedback = []
    for score in confidenceScores:
```

```
# Based on the confidence score, generate feedback for each pose
        poseFeedback = generatePoseFeedback(score)
        feedback.append(poseFeedback)
    return feedback

# Display Feedback
def displayFeedback(feedback):
    # Display the feedback to the user through the user interface
    # This can be done using HTML, CSS, JavaScript, and the chosen front-end framework

# Main function
def main():
    # Perform image upload mode
    images = # Load or acquire images
    processUploadedPoseImage(images)

    # Perform real-time mode
    frames = []
    cap = cv2.VideoCapture(0)  # Initialize camera capture
      while  True:
        ret,   frame =  cap.read()
# Read  frame  from  camera
        frames.append(frame)
        if len(frames) >= 10:  # Process a batch of frames
            processRealTimePoseFrames(frames)
            frames = []
        if cv2.waitKey(1) & 0xFF == ord('q'):  # Exit loop on 'q' key press
            break
    if len(frames) > 0:  # Process any remaining frames
        processRealTimePoseFrames(frames)
```

```
    cap.release()  # Release camera capture
    cv2.destroyAllWindows()  # Close all windows

# Entry point of the program
if __name__ == '__main__':
    main()
```

### 5.5.1 Explanation of the Algorithm:

- In both the image upload mode and real-time mode, the **processUploadedPoseImage()** and **processRealTimePoseFrames()** functions are called, respectively.
- In the processUploadedPoseImage() function, a loop is added to process multiple images. Each image is passed through the preprocessing, pose estimation, and pose comparison steps, and the estimated poses and confidence scores are collected.
- Similarly, in the processRealTimePoseFrames() function, a loop is added to process multiple frames. A batch of frames is accumulated, and when the batch size reaches a certain threshold (in this case, 10 frames), they are passed through the preprocessing, pose estimation, and pose comparison steps. The estimated poses and confidence scores are collected.
- The preprocess() function now takes a list of data (images or frames) and applies the necessary preprocessing steps to each item in the loop.
- The **estimatePose()** function now operates on individual items (image or frame) and returns the estimated pose.
- The **comparePose()** function now processes a list of poses and collects the confidence scores for each pose.
- The **generateFeedback()** function generates feedback for each confidence score in the loop and collects the feedback for all poses.
- Finally, the **displayFeedback()** function can be used to display the collected feedback to the user through the user interface.

## 5.5.2 The main parameters used are:

**Image/Frame:** The input data, either an image or a frame captured from a camera, containing the user's pose.

**Processed Image/Frame:** The image or frame after undergoing preprocessing steps, such as image stabilization, noise reduction, and segmentation.

**Estimated Pose:** The estimation of the user's pose based on the processed image or frame. It includes the positions of key body joints in 3D space.

**Confidence Score:** A numerical value indicating how accurately the user is performing the pose. It is calculated by comparing the estimated pose with a reference database or model.

**Feedback:** The output generated by the system based on the confidence score. It provides information to the user about their performance, such as the pose name, visual cues, textual instructions, or any other relevant feedback.

**Dataset:** The manually created dataset consisting of labeled examples of individuals performing different yoga poses. The pose estimation model is trained using this dataset.

**Libraries/Frameworks:** The technologies used in the implementation, including OpenCV, Dlib, and Mediapipe for computer vision tasks, Django for the web framework, HTML, CSS, and JavaScript for the user interface, and MSSQL for the database.

These variables are essential to the system's data and information flow since they allow for posture estimate, comparison, feedback creation, and user interface interaction.

## 5.6. Generic Uniqueness Table:

| **Methods Implemented before:** | **Uniqueness of our Project** |
|---|---|
| Various Machine Learning Algorithms such as CNN (Convolutional Neural Network) | Our project combines OpenCV, Dlib and mediapipe for improved accuracy and real-time performance. |
| RNN (Recurrent Neural Network) | Utilizes the manually created dataset specifically tailored for yoga poses |
| GCN(Graph Convolutional Network) | Integrates real-time perception capabilities from mediapipe for efficient pose estimation. |
| DRL(Deep Reinforcement Learning) | Implements customized feeback generation techniques tailored for yoga poses. |

## Description:

Several studies have investigated the potential of machine learning techniques, such as Convolutional Neural Networks (CNNs), Recurrent Neural Networks (RNNs), Graph Convolutional Networks (GCNs), and Deep Reinforcement Learning (DRL), for 3D yoga pose estimation and feedback. These algorithms have been used to train models on annotated datasets, enabling the estimation of yoga poses and providing feedback to users.

While research papers have contributed valuable insights, our project stands out due to its unique approach. We combine specific libraries, namely OpenCV, Dlib, and Mediapipe, to improve accuracy and achieve real-time performance. Unlike the research papers that may lack integration and customization, our project leverages the capabilities of these libraries, integrating Dlib's feature detection and shape prediction with real-time perception from Mediapipe.

Moreover, our project addresses the limitations identified in research papers. We utilize a manually curated dataset that specifically focuses on yoga poses, ensuring its relevance and diversity. Additionally, we implement customized feedback generation techniques tailored for yoga poses, enabling personalized feedback to users based on their pose accuracy and specific needs.

## 5.7 Briefing on Output Generation:

**Pose Detection:** When a user uploads an image or performs a pose in front of the laptop camera, the code uses the computer vision libraries (OpenCV, Dlib, and Mediapipe) to detect and estimate the user's pose. It identifies the position and orientation of key body joints or keypoints such as the hips, knees, ankles, shoulders, elbows, and wrists.

**Pose Recognition:** Once the pose is detected, the code matches it against a predefined set of yoga poses. Based on the positions and orientations of the key-points, the system identifies the closest matching pose from the available set.

**Confidence Score:** The code calculates a confidence score that indicates how accurately the user is performing the recognized pose. This rating is determined by contrasting the user's pose with the recommended pose for that specific yoga posture. It measures the alignment, angles, and other relevant parameters to evaluate the accuracy of the user's pose execution.

**Feedback Generation:** Using the confidence score, the code generates customized feedback for the user. The feedback may include suggestions for improving the pose, highlighting areas where the user needs to adjust or align their body, and providing tips to enhance the overall pose execution. This feedback aims to guide the user towards better alignment, balance, and posture during yoga practice.

## 5.8 Experimental Study:

## Table :1 On 3d Images

| Model Tested | Input Data-1<br>Correct Input Yoga Pose | | Input Data-2<br>Incorrect pose with feed back | |
|---|---|---|---|---|
| | **Accuracy** | **Error Rate** | **Accuracy** | **Error Rate** |
| **CNN** | 95.0104873925 % | 4.98951260745 | 95.01019039 % | 4.9898096024 |
| **ANN** | 84.0224758684 % | 15.9775241 | 84.023338407 % | 15.976661592 |

The Experimental study as per table:1 concludes that the deep learning models are excellent for the pose estimation with features and key-points detection but CNN gives only minimal amount of error rate in the detection of our pose. Thus, we will use CNN model to further implement our project in real-time with Mediapipe Library.We will use CNN Model for the real-time Implementation. Also tested the two effective Machine Learing Models namely Logistic Regression and Ridge Classifier for testing and verifying that whether the deep learning model fits best to according to our dataset used . But only in Real Time not experimented for the 3d Input Data Images.

**Table-2: For Real-Time Only:**

| **Models Tested** | Accuracy | Error Rate |
|---|---|---|
| **Logistic Regression (LR)** | 75.00235597% | 25.9876253 |
| **Ridge Classifier (RF)** | 80.22978652% | 20.8803652 |
| **CNN** | 99.35879 | 0.65 |
| **ANN** | 97.22356 | 2.78 |

## Table :3 The Model Used for Implementation: Evaluation Metrics

| **Model Implemented** | **Deep Learning Model-CNN** |
|---|---|
| Key-points Detected | 33 Landmarks Mediapipe Library |
| Accuracy on Real-Time | 0.9723577235772358 |
| Precision | 0.9791590385576141 |
| Recall | 0.9179271708683474 |
| F1 Score | 0.9420719931783191 |
| Error Rate | 0.027642276422764178 |

**Our result in Table:3 describes the precision_score, recall_score, f1_score, and accuracy_score functions from the sklearn.metrics module to calculate the evaluation metrics.**

**Precision**: Precision is a measure of the accuracy of the positive predictions made by a model. It calculates the ratio of true positive predictions to the sum of true positive and false positive predictions. Precision indicates how well the model identifies true positives while minimizing false positives.

In this experiment , the actualValue represents the actual class labels of the samples, and the predictedValue represents the predicted class labels by the model.The precision is calculated using the precision_score function with the average='macro' parameter. It computes the precision for each class and then averages them.

precision = precision_score(actualValue, predictedValue, average='macro')

**Recall:** Recall, also known as sensitivity, measures the model's ability to correctly identify positive instances out of all the actual positive instances. It calculates the ratio of true positive predictions to the sum of true positive and false negative predictions. Recall indicates how well the model captures all the relevant positive instances.

The recall is calculated using the recall_score function with the average='macro' parameter. It computes the recall for each class and then averages them.

recall = recall_score(actualValue, predictedValue, average='macro')

**F1 Score:** The F1 score is the harmonic mean of precision and recall. It provides a balanced measure of a model's performance by considering both precision and recall. The F1 score is useful when the dataset is imbalanced or when both precision and recall are important. It ranges between 0 and 1, where a higher value indicates better model performance.

The F1 score is calculated using the f1_score function with the average='macro' parameter. It computes the F1 score for each class and then averages them.

f1 = f1_score(actualValue, predictedValue, average='macro')

### Accuracy:

The accuracy is calculated using the accuracy_score function. It compares the actual and predicted values and returns the ratio of correctly classified samples to the total number of samples.

accuracy = accuracy_score(actualValue, predictedValue)

**Error Rate:** The error rate is calculated by subtracting the accuracy from 1. It represents the fraction of misclassified samples.

error_rate = 1 – accuracy

The **average='macro'** parameter is used to calculate the metrics for each class and then average them. You can modify it based on your specific requirements. The code will output the precision, recall, F1 score, accuracy, and error rate metrics based on the provided actual and predicted values.

## Description on Table:1 and Table:2

The experimental study conducted on 3D images and in real-time using the CNN model showed promising results for our project of 3D yoga pose estimation and correction.

In the analysis of 3D images, the CNN model achieved an accuracy of 95.01%, with an error rate of 4.99%. This indicates that the model was able to accurately classify the correct input yoga poses with a high degree of accuracy. The low error rate demonstrates the robustness and effectiveness of the CNN model in accurately estimating the poses from the input data.

Comparing the CNN model with the ANN model, the CNN model outperformed in terms of accuracy, achieving 95.01% compared to the 84.02% accuracy of the ANN model. This indicates that the CNN model is more effective in capturing the complex features and relationships in the input data, resulting in better pose estimation performance.

In the real-time implementation, the CNN model achieved an accuracy of 97.24%. This means that the model successfully recognized and classified the yoga poses performed by the user in real-time with a high level of accuracy.

The precision score of 0.9792 indicates a high percentage of correctly classified positive instances (correct poses), while the recall score of 0.9179 demonstrates a good ability to identify all relevant positive instances. The F1 score of 0.9421 represents a balanced measure of precision and recall, indicating a good overall performance of the CNN model in real-time pose estimation.

The low error rate of 0.0276 further emphasizes the accuracy of the CNN model, with a small proportion of misclassifications or pose estimation errors.

Overall, these results highlight the effectiveness and superiority of the CNN model for 3D yoga pose estimation and correction, making it the preferred choice for implementing the project.

# Chapter 6

## 6.1 Conclusion:

The 3D yoga pose estimation and feedback system described above aims to provide users with accurate feedback on their yoga poses. The system allows users to either upload an image of a yoga pose or perform poses in real-time using their laptop camera. The system utilizes computer vision techniques, pose estimation algorithms, and pose comparison methods to estimate the user's pose and generate feedback based on the accuracy of their performance.

By incorporating technologies such as OpenCV, Dlib, and Mediapipe, the system can detect and estimate the positions of key body joints in 3D space. These estimated poses are then compared with a reference dataset or model to calculate confidence scores. Based on the confidence scores, personalized feedback is generated and presented to the user through the user interface.

## 6.2 Future work:

**Real-Time Feedback Improvisation:** Currently, the system provides feedback after the user has completed the pose. Future iterations could include real-time feedback, where users receive immediate guidance and corrections as they perform the pose.

**User Progress Tracking:** Implementing a user profile and progress tracking feature would enable users to monitor their improvement over time. This could include tracking metrics such as pose accuracy, consistency, and overall performance.

**Gamification and Challenges:** Introducing gamification elements, such as challenges, achievements, or virtual rewards, can enhance user engagement and motivation to practice yoga regularly.

**Mobile Application:** Creating a mobile application version of the system would allow users to access pose estimation and feedback on their smartphones or tablets, enabling them to practice yoga anytime and anywhere.

**Integration with Smart Devices:** Integration with smart devices, such as wearable fitness trackers or smart mirrors, could provide additional data and insights to enhance the accuracy and effectiveness of pose estimation and feedback.

**Machine Learning Advancements:** Incorporating advanced machine learning techniques, such as deep learning models, potentially enhance pose estimation's precision and resilience, leading to more accurate feedback.

# Bibliography


1. L. Sigal. “Human pose estimation”, Ency. of Computer Vision, Springer 2011.S. Yadav, A. Singh, A. Gupta, and J. Raheja, “Realtime yoga recognition using deep learning”, Neural Computer and Appl., May 2019. [Online].
2. U. Rafi, B. Leibe, J.Gall, and I. Kostrikov, “An efficient convolutional network for human pose estimation”, British Mach. Vision Conf ., 2016.
3. S. Haque, A. Rabby, M. Laboni, N. Neehal, and S. Hossain, “ExNET: deep neural network for exercise pose detection”, Recent Trends in Image
4. Process. and Patter Recog., 2019.
5. M. Islam, H. Mahmud, F. Ashraf, I. Hossain and M. Hasan, ”Yoga posture recognition by detecting human joint points in real time using microsoft kinect”, IEEE Region 10 Humanit. Tech. Conf., pp. 668-67, 2017.
6. H. Sidenbladh, M. Black, and D. Fleet, “Stochastic tracking of 3D human figures using 2D image motion”, Proc 6th European Conf. Computer Vision, 2000.
7. “Deep 3D human pose estimation: A review”, by Jinbao Wang a,1 , Shujie Tan a,1 , Xiantong Zhen b,e, 2021.
8. ”A simple yet effective baseline for 3d human pose estimation”, by Julieta Martinez1 , Rayat Hossain1, University of British Columbia, Vancouver, Canada, CVPR 2017.
9. “Can 3D Pose be Learned from 2D Projections Alone?” by Dylan Drover, Rohith MV, Ching-Hang Chen, Amit Agrawal, Ambrish Tyagi, , Sunnyvale, CA, USA, CVPR 2018.
10. “Yoga-82: A New Dataset for Fine-grained Classification of Human Poses”, by Manisha Verma, Sudhakar Kumawat, CVPR 2020,April.

11. “Yoga pose estimation and feedback generation using deep learning”, by Tapas Badal, Vipul kumar Mishra, Anurag Srivastava, Vivek Anand Thoutam , March 2022 on Research gate.

12. “Yoga pose Monitoring system using Deep Learning”, by Santosh Satapathy, Debabrata Swain, Pramoda Patro, Aditya kumar Saho, 27 June, 2022.

13. “ AI Human Pose Estimation”, by Rutujha Gajbhiye, Pooja Gaikwad, Sweta kaporde, Volume-7, May 2022, Issue 5 in International Journal.

14. “Deep learning based yoga pose classification, by Abhishek Desai, Shakti Kinger, Sarvarth patil, Hrishikesh Sinalkar, IEEE 2022 International Conference.

15. “Real Time yoga pose detection using deep learning, by Khushi Sidana, NMIMS, Nov 2022, International Journal Volume-7 , Issue 7.

16. “ A Survey on Yogic Posture Recognition”, by Arun Kumar Rajendran, Sibi Chakravarthy Sethuraman, by IEEE Vol 11.

17. “Visual feedback for core training with 3d human shape and pose”, by Haoron Xie, Atsushi, Kazunori Miyata, IEEE 2019.

18. “Pose Estimated yoga monitoring system”, by Sreeja KA , Sarath Sajan, Ardra Anikumar, IEEE coference , Jul 2021.

19. “ A self Learning Yoga Monitoring system based on pose estimation, by prahitha Movva, Himangshu Sharma, Hemanth Pasupuleti, June 2022.

20. “A Comprehensive survey on Real Time human pose estimation”, by Apoorva Pravin Datir, Nikita Tanaji, Snehal Shivaji Funde, Pallavi Dhade , IEEE 2023

# 21Mced09_Thesis_09_removed (1)_removed.

ORIGINALITY REPORT

| 9% | 7% | 4% | 5% |
|---|---|---|---|
| SIMILARITY INDEX | INTERNET SOURCES | PUBLICATIONS | STUDENT PAPERS |

PRIMARY SOURCES

| | | |
|---|---|---|
| 1 | www.researchgate.net<br>Internet Source | 1% |
| 2 | Submitted to Sreenidhi International School<br>Student Paper | 1% |
| 3 | Submitted to The Robert Gordon University<br>Student Paper | 1% |
| 4 | www.hindawi.com<br>Internet Source | <1% |
| 5 | ijisrt.com<br>Internet Source | <1% |
| 6 | Submitted to University of Greenwich<br>Student Paper | <1% |
| 7 | www.mdpi.com<br>Internet Source | <1% |
| 8 | Submitted to Birla Institute of Technology and Science Pilani<br>Student Paper | <1% |
| 9 | Submitted to Cranfield University<br>Student Paper | <1% |