DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Ghurdauri, Pauri Garhwal (U.K)

GOVIND BALLABH PANT INSTITUTE OF ENGINEERING & TECHNOLOGY

VIRTUAL YOGA TRAINER

By

Ayushi	**(180214)**
Bhajan Singh Kandari	**(180215)**
Himanshu Pandey	**(180224)**
Ritesh Kumar	**(192204)**

Under the guidance of

Dr. ANNAPURNA SINGH

(ASSOCIATE PROFESSOR)

(CSED)

VIRTUAL YOGA TRAINER

A Project report on

Submitted in partial fulfilment of the requirements for the award of the degree

BACHELOR OF TECHNOLOGY

in

COMPUTER SCIENCE AND ENGINEERING

By

Student Name	Roll no.
Ayushi	**180214**
Bhajan Kandari	**180215**
Himanshu Pandey	**180224**
Ritesh Kumar	**192204**

Under the guidance of

DR. ANNAPURNA SINGH

(ASSOCIATE PROFESSOR)

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

G.B PANT INSTITUTE OF ENGINEERING & TECHNOLOGY, PAURI, UTTRAKHAND

(2018-2022)

CANDIDATE'S DECLARATION

I/We hereby Certify that the project work entitled "**VIRTUAL YOGA TRAINER**" in partial fulfillment of the requirements for the award of the Degree of BACHELOR OF TECHNOLOGY in COMPUTER SCIENCE AND ENGINEERING with specialization in OPEN SOURCE AND OPEN STANDARDS and submitted to the Department of Computer Science and Engineering Pauri, Uttarakhand, is an authentic record of my/our work carried out during a period from **Feb, 2022** to **June, 2022** under the supervision of **Dr.Annapurna Singh, Associate Professor, Department of Computer Science and Engineering.**

The matter presented in this project has not been submitted by me/us for the award of any other degree of this or any other University.

Student Name	**Roll No.**
Ayushi	180214
Bhajan Singh Kandari	180215
Himanshu Pandey	180224
Ritesh Kumar	192204

This is to certify that the above statement made by the candidate is correct to the best of my knowledge.

Date:

Dr. Annapurna Singh
(Supervisor)

Prof. Vivek Kumar Tamta
Project Coordinator
Computer Science & Engineering
G.B Pant Institute of Engineering
& Technology,
Pauri (Uttarakhand)

Dr. H.S. Bhadauria
Head
Computer Science & Engineering
G.B Pant Institute of Engineering
& Technology,
Pauri (Uttarakhand)

CANDIDATE'S UNDERTAKING

We hereby declare that the work which is being presented in the report entitled "**VIRTUAL YOGA TRAINER**" is original and certify that the plagiarism has already being checked by us (plagiarism report attached). If in future and issue related to plagiarism is found, the whole responsibility will be of the candidate undersigned.

Date: …….

Ayushi(180214)

Bhajan Singh Kandari(180215)

Himanshu Pandey(180224)

Ritesh Kumar(192204)

ACKNOWLEDGEMENT

We would like to begin by thanking; **Dr. Annapurna Singh**, our supervisor, her support, encouragement and enthusiasm motivated us to believe in ourselves towards this project work. We thank her for all the energy and time she has spent for us, discussing everything from research to career choice, reviewing our project and guiding our work through the obstacles and setbacks. We thank her for holding our hand when even we did not believe in ourselves and made us able to stand. Her professional yet caring approach towards people, and her passion for living the life to the fullest has truly inspired us. We need her guidance throughout our life.

We would like to thank **Dr. H.S. Bhadauria** (Head of the Department, Computer Science and Engineering), for giving his important time for reviewing our work and suggesting us that how work will be done carefully.

Mr Vivek Kumar Tamta (Project Coordinator) for reviewing our work, and making it worth for final submission.

We would like to thank our entire classmate and friends at the college for playing a precious role in our life and inspiring us by their direct/indirect help and support, especially who were with us in our tough time

Finally, We want to thank our family for working so hard for us, for their love and for giving us all the happiness and opportunities that most people can only dream of. We could not have done this without them. Last but not the least, thanks to the Almighty God, without his wish nothing can be possible.

TABLE OF CONTENT

LIST OF FIGURES

LIST OF ABBREVAIATIONS

CNNs	Convolutional Neural Networks
RNNs	Recurrent Neural Networks
2D	2-Dimensional
3D	3-Dimensional
RGB	Red Green Blue
AI	Artificial Intelligence
ANN	Artificial Neural Networks
MLP	Multi Layered Perceptron
NLP	Normal Language Processing
HAR	Human Activity Recognition
OpenCV	Open Computer Vision
TF	Tensorflow
ML Model	Machine Learning Model
C3D	3-Dimensional Convolutional Neural Network
LSTM	Long Short-Term Memory
SVM	Support Vector Machine
KNN	K-Nearest Neighbour
DA	Data Augmentation
DL	Deep Learning

ABSTRACT

Yoga is an ancient science and discipline originated in India 5000 years ago. It is used to bring harmony to both body and mind with the help of asana, meditation and various other breathing techniques It bring peace to the mind. Due to increase of stress in the modern lifestyle, yoga has become popular throughout the world. There are various ways through which one can learn yoga. Yoga can be learnt by attending classes at a yoga centre or through home tutoring. It can also be self-learnt with the help of books and videos. Most people prefer self-learning but it is hard for them to find incorrect parts of their yoga poses by themselves. Using the system, the user can select the pose that he/she wishes to practice. He/she can then upload a photo of themselves doing the pose. The pose of the user is compared with the pose of the expert and difference in angles of various body joints is calculated. Based on this difference of angles feedback is provided to the user so that he/she can improve the pose.

CHAPTER-1

INTRODUCTION

1.1 GENERAL OVERVIEW

The innovations in technology and science are happening in a drastic phase, which makes human life is more and more hassle-free. Nowadays everybody is aware of its relevance in day to day life. As in every domain, the influence of computers and computer-powered technologies are well established in healthcare and related domains. Apart from the usual medical practices, other practices like Yoga, Zumba, martial arts, etc are also widely accepted among society as a way to achieve good health. Yoga is a set of practices sprouted in ancient India which deals with the wellness of physical, mental and spiritual condition of a man. Yoga has gained a big significance in the medical community. The benefits of yoga are improved health, mental strength, weight loss, etc. But Yoga must be practiced under the supervision of an experienced practitioner since incorrect or bad posture can lead to health problems such as ankle sprain, stiff neck incorrect or bad posture can lead to health problem , muscle pulls, etc. The yoga instructor must correct the individual postures during training. After proper training only, one can practice Yoga on their own. However, in today's situation, people are more comfortable in their homes and everyone changes almost everything to online mode. This situation demands a technology-driven Yoga practice. This can be done with the help of mobile applications or with virtual tutoring applications. In both cases, there are vast opportunities to explore computationally for making them more powerful, intelligent, and efficient. We put forward a classification model for recognizing and identifying a yoga asana from an image or from a frame of a video in this work. The classification model is developed using deep learning techniques backed with image processing and computer vision methods. The entire work classifying 10 classes of yoga asanas. The postures like Downward dog, Tree Plank, goddess & Warrior1 were analysed and recognized by the deep learning model and then used to identify the exact class of the yoga posture.

This project focuses on exploring the different approaches for yoga pose classification and seeks to attain insight into the following: What is pose estimation? What is deep learning? How can deep learning be applied to yoga pose classification in real-time? This project uses references from conference proceedings, published papers, technical reports and journals. Fig. 1 gives a graphical overview of topics this paper covers. The first section of the project talks about the history and importance of yoga. The second section talks about pose estimation and explains different types of pose estimation methods in detail and goes one level deeper to

explain discriminative methods – learning based (deep learning) and exemplar. Different pose extraction methods are then discussed along with deep learning based models - Convolutional Neural Networks (CNNs) and Recurrent Neural Networks (RNNs)

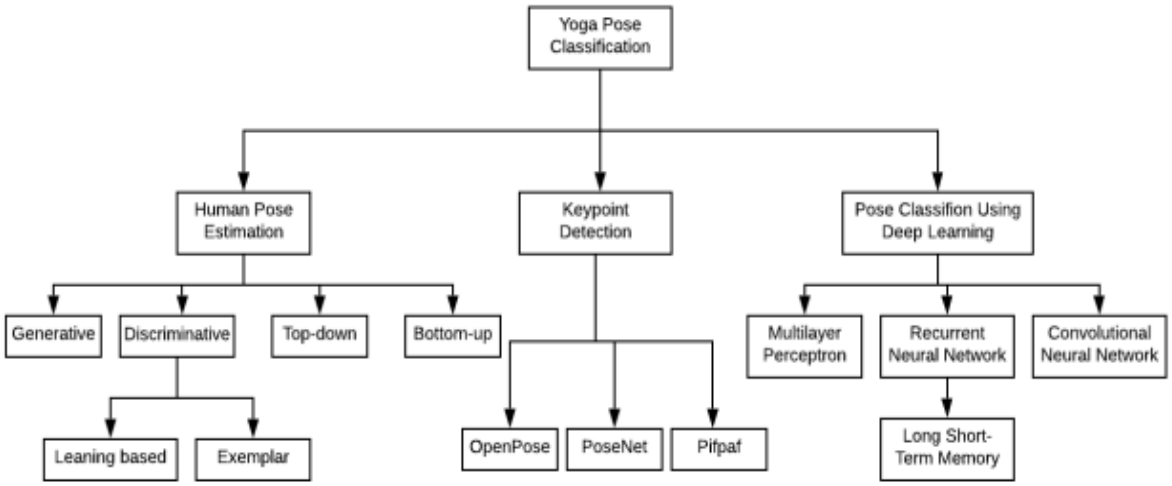


Fig. 1. Conceptual outline of topics

1.2 HISTORY

People are inclined to musculoskeletal problems with maturing and mishaps. So as to forestall this a few, type of actual exercise is required. Yoga, which is a physical and otherworldly work out, has increased colossal centrality in the network of clinical scientists. Yoga has the capacity to totally fix illnesses with no prescriptions and improve physical and mental wellbeing . A tremendous assemblage of writing on the clinical uses of yoga has been created which incorporates positive self-perception mediation, heart restoration, psychological sickness and so forth . Yoga contains different asanas which speak to actual static stances. The utilization of posture assessment for yoga is trying as it includes complex setup of stances. Moreover, some best in class strategies neglect to perform well when the asana includes even body act or then again when both the legs cover one another. Subsequently, the need to build up a strong model which can help advocate self-taught yoga frameworks emerges.

1.3 HUMAN POSE ESTIMATION

Human stance acknowledgment has made colossal headways in the previous years. It has advanced from 2D to 3D present assessment and from single individual to multi individual posture assessment. employments present assessment to fabricate an AI application that identifies shoplifters while utilizes a solitary RGB camera to catch 3D stances of numerous individuals continuously. Human posture assessment calculations can be broadly coordinated in two different ways. Calculations prototyping assessment of human stances as a mathematical estimation are named generative strategies while calculations demonstrating human posture assessment as a picture preparing issue are named discriminative strategies. Another method of grouping these calculations depends on their strategy for or king. Calculations beginning from a more significant level speculation and descending are called top-down strategies, while calculations that start with pixels and move upwards are rung base strategies.

1.3.1 GENERATIVE

Generative methods give a procedure to foresee the highlights from a given posture speculation. They start with instating the stance of the human body and venture it to the picture plane. Changes are made to make the extended picture and current picture perceptions consistent. Generative based methodologies offer simple speculation because of less requirement of a preparing present dataset . Nonetheless, because of the high dimensional projection space search, this technique isn't considered computationally plausible, and is in this manner more slow when contrasted with discriminative techniques. portrays a generative Bayesian technique to follow 3D fragmented human body figures in recordings. This is a probabilistic technique which comprises of a generative model for picture appearance, an underlying likelihood dissemination over joint points and represent that speaks to development of people and a strong probability work. Despite the fact that the strategy can follow people in obscure convoluted foundations, it faces the danger of in the long run forgetting about the object

1.3.2 DISCRIMINATIVE

In opposition to generative techniques, discriminative strategies start with the proof of the picture and get familiar with a strategy to demonstrate the connection between the human postures and proof on the premise of preparing information. Model testing in discriminative strategies is much quicker instead of generative strategies because of the hunt in an obliged space rather than a high dimensional include space. Investigates a

discriminative based learning technique to get 3D human posture from outlines. This methodology doesn't need a body model unequivocally nor any earlier marked portions of the body in the picture. It reestablishes the posture utilizing non-straight relapse dependent on the shape descriptor vectors brought naturally from outlines of pictures. It utilizes Relevance Vector Machine (RVM) regressors and damped least squares for relapse . The strategy, however expanding the precision by multiple times, isn't sufficiently exact, as there are a few examples of erroneous postures and results indicating critical fleeting jitter. Discriminative strategies are further ordered into learning techniques and model strategies

1.3.2.1 LEARNING BASED – DEEP LEARNING:

One significant learning-based technique is profound realizing which is based upon Artificial Neural Organizations (ANNs). ANN is similar to the human cerebrum where the units in an ANN speak to the neurons in the human mind, and loads speak to the quality of association between neurons. Profound learning gives a start to finish design that permits programmed learning of key data from pictures. One famous profound learning model which has been generally utilized for present assessment is Convolutional Neural Network (CNN) which will be talked about later. have added to the exploration by utilizing CNNs and stacked auto-encoder calculations (SAE) for distinguishing yoga stances and Indian old style move structures. In any case, their presentation assessment is done distinctly on pictures and not on recordings

1.3.2.2 EXEMPLAR METHODS:

In model techniques, present assessment depends on a special arrangement of postures with their equal portrayals. Characterization calculations, for example, arbitrary timberlands and randomized trees are strong and quick enough to deal with this. Irregular woods is comprised of different randomized choice trees and is henceforth called a group classifier. It comprises of non-terminal hubs which have a choice capacity to foresee the similitudes in pictures.

1.4 POSE CLASSIFICATION USING DEEP LEARNING

Profound learning is generally utilized for picture order undertakings wherein the model takes input as pictures and yields a forecast. Profound learning calculations utilize neural organizations to decide the association between the information and yield. For present assessment issues, the picture with posture of people is taken as information and the profound learning model attempts to adapt effectively the various postures in order to precisely group them. As one can figure, this could be a computationally costly assignment if the quantity of pictures is enormous. Additionally, as we need precise outcomes we would not need to settle on the nature of the pictures as that could influence the highlights removed by the model. Consequently, in this venture we propose utilizing OpenPose (a pretrained model) to separate keypoints of the human joint areas from the pictures and afterward preparing the profound learning model on these keypoints. The following are some essential profound learning models utilized for characterization issues.

1.4.1 MULTILAYER PERCEPTRON (MLP):

MLP is an old style neural organization that has one info and one yield layer. The transitional layers between the info and yield layer are known as concealed layers. There can be at least one shrouded layers. MLPs structure a completely associated network as each hub in one layer has an association to each hub in another layer. A completely associated network is an establishment for profound learning. MLP is well known for directed characterization where the information is relegated a mark or class. employments MLP for human posture order by separating keypoints from low goal pictures utilizing Kinect sensor.

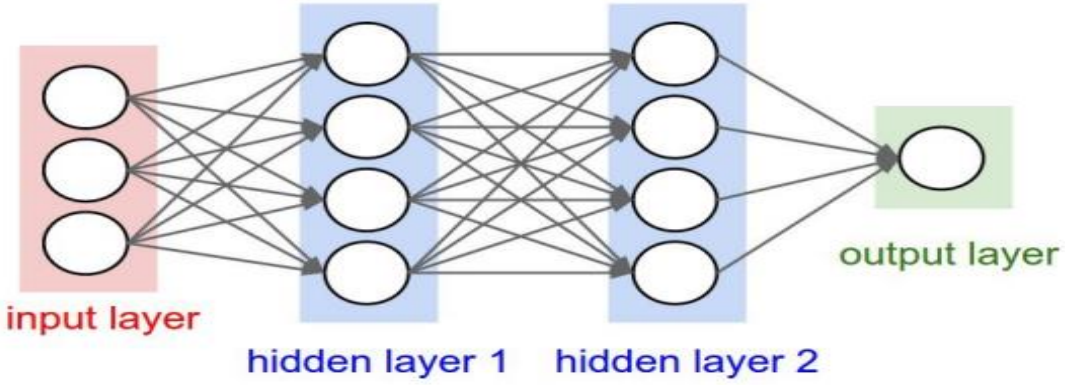


Fig.2. Schematic diagram of multilayer perceptron

1.4.2 REPETITIVE NEURAL NETWORK (RNN):

RNNs are neural organization designs that are utilized for grouping expectation issues. Grouping expectation issues can be one to many, numerous to one, or numerous to many. In RNNs, the past information of a neuron is saved which helps in dealing with the successive information. Thus, the setting is protected, and yield is created considering the recently learned information. RNNs are most regularly utilized for normal language handling (NLP) issues where the information is normally demonstrated in groupings.Be that as it may, in action acknowledgment or posture arrangement undertakings as well, there is a reliance between the recently performed activity and the following activity. In the event of yoga also, the unique situation or on the other hand data of introductory or go-between presents is significant in anticipating the last posture. Yoga can in this way be considered as a succession of postures.

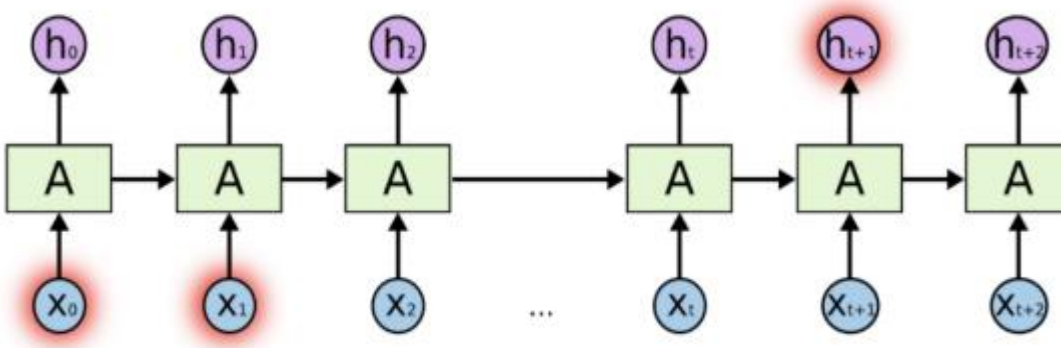


Fig. 3. Long-term dependency in RNN

1.5 LONG SHORT-TERM MEMORY (LSTM):

So as to manage the above long haul reliance issue, an extraordinary kind of RNN exists which is called LSTM. A LSTM is a renowned RNN that can undoubtedly recall data or then again information for a considerable length of time timeframes which is its default conduct. The key thought which makes this conceivable is cell state. A cell state permits unaltered data stream. It tends to be considered as a transport line. LSTMs can add and kill information from the cell state utilizing administrative structures known as entryways. These exceptional entryways consider alternatively letting the data through. LSTM utilizes three entryways, specifically information, refresh and overlook. A LSTM

can accordingly specifically overlook or recollect the learnings. As LSTMs take into account longer maintenance of the info state in the organization, they can proficiently deal with long successions and give great outcomes. discusses how LSTM can be utilized with CNN for human action acknowledgment to accomplish high exactness.

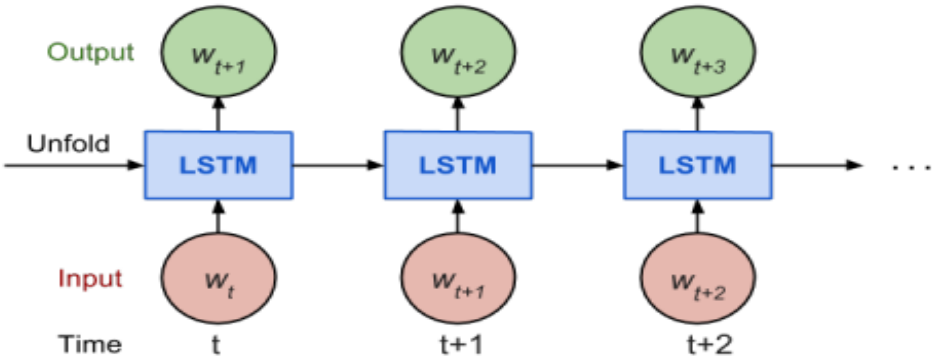


Fig. 4. LSTM architecture

1.6 CONVOLUTIONAL NEURAL NETWORK (CNN):

CNN is a type of neural network which is widely used in the computer vision domain. It has proved to be highly effective such that it has become the go-to method for most image data. CNNs consist of a minimum of one convolutional layer which is the first layer and is responsible for feature extraction from the image. CNNs perform feature extraction using convolutional filters on the input and analyzing some parts of the input at a given time before sending the output to the subsequent layer. The convolutional layer, through the use of convolutional filters, generates what is called a feature map. With the help of a pooling layer, the dimensionality is reduced, which reduces the training time and prevents overfitting. The most common pooling layer used is max pooling, which takes the maximum value in the pooling window. CNNs show a great promise in pose classification tasks, thus making it a highly desirable choice. They can be trained on key points of joint locations of the human skeleton or can be trained directly on the images. used CNN to detect human poses from 2D human exercise images and achieved an accuracy of 83%. On the other hand, used CNN on OpenPose key points to classify yoga poses and achieved an accuracy of 78%. Although the accuracy is not exactly comparable as the dataset along with the CNN architecture and exercises being classified are different, shows how using CNNs on OpenPose key points is worth exploring.

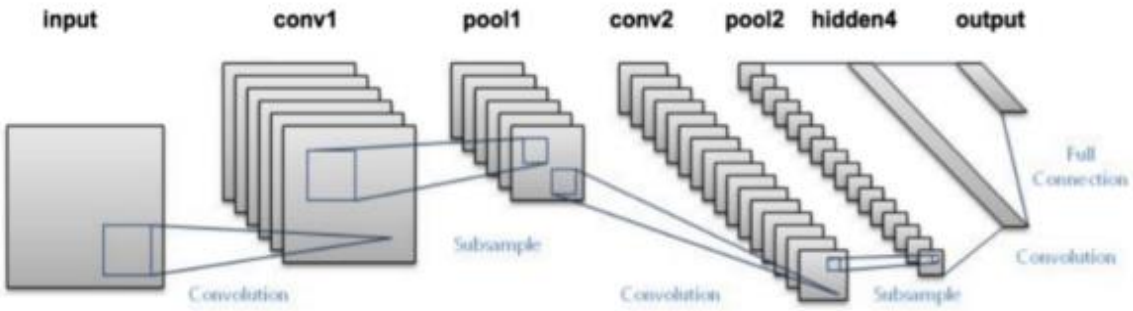


Fig. 5. CNN architecture layers

CHAPTER-2

LITERATURE SURVEY

AI strategies may maybe rely significantly upon heuristic human element extraction in everyday errands of location of social exercises. It is limited by human zone mindfulness. To talk this danger, creators have decided on a couple of techniques like profound learning methods. These strategies could consequently extricate explicit highlights during the reparing stage from crude sensor information, and afterward low-level fleeting qualities with significant level unique requests would be introduced. With regards to the freed application from profound learning approaches in fields like picture grouping, voice acknowledgment, preparing of regular language, and some others, it has been developing into a novel examination way in design location and to move it to an area of human action identification. It shows a couple of the current AI alongside the exactness of HAR. Here, we have referenced just the techniques that give the best exactness for the most extreme number of subjects. For instance, on the off chance that the subject tally is less, at that point the precision might be better.

2.1 RELATIVE WORKDONE

- Sruti Kothari explained a computational method using deep learning, particularly CNN, for classifying the Yoga postures from images. They have considered a dataset containing roughly 1000 images distributed over 6 classes for building the classification model. Nearly 85% accuracy was obtained for this work.
- Hua- Tsung Chen proposed a yoga posture recognition system that can recognize the yoga posture performed by the trainer. In the first step, he used a kinetic to capture the body map of the user and body contour extraction. Then, the star skeleton which is a fast skeletonization technique by connecting from the centroid to other joint parts is done as the next step. From this technique, accuracy of 96% is acquired.
- Edwin W.trejo introduced a model for pose correction in yoga. the user will receive real-time instruction about the pose correction made by an expert. for this, they used a recognition algorithm based on the AdaBoost algorithm to create a robust database for estimating 6 yoga poses. Finally, 92% accuracy is obtained.
- Yoga is a traditional exercise that can bring harmony and peace to both body and mind. But self-learning yoga without the help of an instructor is a hard task. But here a solution is proposed for this, a photo of themselves doing the pose is uploaded then it

compares to the pose of the expert and difference in angles of various body joints is calculated. An alternative computationally efficient approach for yoga pose recognition in the real world is presented. A 3D CNN architecture is designed to implement real time recognition of yoga poses, it is a modified version of C3D architecture. For computational efficiency computationally fully connected layers are pruned, and supplementary layers such as the batch normalization and average pooling were introduced.as a result an accuracy of 91.5% is obtained.

- A dataset consisting of 6 yoga asanas is created. For this work CNN which is used to extract features from key points and LSTM to give temporal prediction is used. As a result a accuracy of 99.04% is obtained.
- Chen et al. proposed a system for distinguishing a yoga act utilizing a Kinect camera. Those assembled an amount of 300 recordings of 12 yoga stances from 5 yoga pros with each present performed on five events. At first, the closer view part is fragmented from the cut, and the star skeleton is used and acquired an exactness of 99.33%. The creators in proposed a stance location system utilizing a quality Kinect camera with a goal of 640 X 480. They saw six postures performed by five volunteers. They separated 21000 casings from the Kinect camera by utilizing a foundation deduction technique with a 74% precision. Later the creators in proposed another posture discovery methodology utilizing Kinect camera. Also, Wang et al. proposed a position acknowledgment methodology using the Kinect camera. They isolated the human blueprint and utilized a learning vector quantization neural framework for five fundamental stances. The system accuracy was roughly 98 %. onetheless, these outcomes have high precision rates, and they are seen as security prominent.
- Yao et al. have proposed a human stance recognizable proof system using a disconnected RFID signal. This methodology sees the human postures subordinate over the assessment of RSSI signal models, which made while singular plays out the posture in a RFID name group and a RFID gathering mechanical assembly. An accuracy of 99 percent for 12 stances was achieved through the structure. In spite of its higher exactness rate, the RSSI signal is climate subordinate; likewise, the foundation and upkeep of this technique are modern; an immediate aftereffect of its various parts. In [35], the creators have developed a savvy petition tangle that sees four represents that are seen during supplication. The tangle has a couple power identifying resistive strips inside and perceives the region where the individual's body is crushing. The tangle

apparent the stances assembled from 30 individuals with 100% exactness. Regardless, such a procedure can't recognize the body parts if they are not pushed on the tangle. An earlier variation of this paper grasps comparative hardware, including three tomahawks (x, y, and z) hubs, for the security protected posture acknowledgment framework. Exploratory results achieved a high ordinary F1-score of roughly 98% in various blends of measurements. In any case, the outcomes rely upon only six postures assembled from four volunteers.

- In , the creators utilized an inclination histogram, and Fourier descriptors subject to centroid highlights are used. By then, Jain et al. used two classifiers, SVM also, K-NN, to see the activities of two open datasets. They utilized six inertial assessment units to construct the system. The creators used the Random Forest classifier to portray the exercises. Finally, an overall accuracy of 84.6% was refined. The creators in proposed a classifier by coordinating the profound CNN and LSTM for grouping hand movements of 5 developments with a F1 score of 0.93 and 0.95 for the two classifiers independently. Lin et al. presented another iterative CNN approach with autocorrelation pre-preparing, as opposed to ordinary little scope Doppler preprocessing, which can absolutely arrange seven activities or five subjects. Furthermore, this framework used an iterative profound learning structure to normally characterize and separate highlights.

CHAPTER-3

BRIEF DESCRIPTION OF THE PROJECT

This project aims to tackle a very common yet deep rooted problem in the field of Computer Vision, not only this with the help of Machine learning and deep learning we are also tackling the issue of hiring a trainer or risking ourselves by getting in contact with another person in this pandemic.

BASIC STEPS TO CREATE THE ML MODEL

• The dataset is downloaded from Kaggle which was created for this specific purpose.

• The data is then split into 3 parts Training Data, Validation Data and Test Data

• Training the model using a Convolutional Neural Network.

• Testing the model against the Test data.

• Using the model in real life by feeding live video from the webcam.

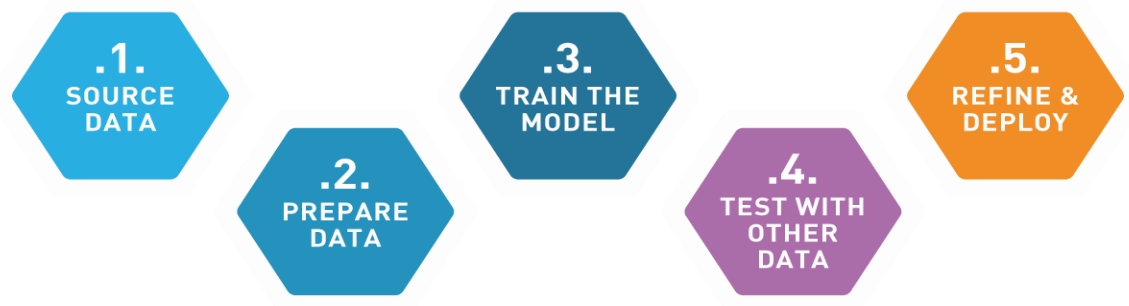


Fig.6. Basic Steps to create a model

3.1 PROBLEM STATEMENT

The problem in today's world is to maintain a healthy lifestyle and that too in the covid pandemic without any public gathering and following covid protocols strictly. The healthy lifestyle can be maintained by performing regular **'Yogic asanas'** so we will be developing a virtual method for practising yoga without the assistance of an instructor that is both cost-effective and prevents individuals from physically interacting in this pandemic.

3.2 SOLUTION TO THE PROBLEM

With the use of machine learning algorithms, we will be able to classify yoga positions. This technology will not only assist users in correcting yoga positions, but it will also benefit them in terms of comfort, economy, and mental well-being. We'll import modules and libraries, load

the dataset, pre-process it, then do some exploratory data analysis. After that, we'll normalise the data and utilise it to create training and testing datasets to train and test our model. We'll then train and test our model to see how well they perform in terms of accuracy, precision, and other factors.

The main benefits of the virtual yoga trainer will be:

- It will be cost Effective as only a laptop with a webcam will be needed.
- It will reduce the risk of getting in contact with other people by removing the need of a yoga instructor.
- Can be accessible anytime from anywhere.
- Covid protocols can be followed easily.
- Computer based application helps children learn faster.

3.3 TECHNOLOGIES USED

3.3.1 Python 3.8 (Language):

Python is termed as interpreted, object-oriented, high-level programming language with dynamic semantics. The high-level is made in data structures, in combination with dynamic typing and binding, which helps in making it very attractive for Rapid Application Development. Python is very well suited for machine learning applications
because of its large community of developers and the easy availability of libraries like Scikit-learn, OpenCV, Numpy, Tensorflow, Keras etc. It helps in scripting languages where the components are together. Python is termed as very simple, easy to learn and has simple syntax.

3.3.2 OpenCV (Python Library):

Python is termed as interpreted, object-oriented, high-level programming language with dynamic semantics. The high-level is made in data structures, in combination with dynamic

typing and binding, which helps in making it very attractive for Rapid Application Development. Python is very well suited for machine learning applications
because of its large community of developers and the easy availability of libraries like Scikit-learn, OpenCV, Numpy, Tensorflow, Keras etc. It helps in scripting languages where the components are together. Python is termed as very simple, easy to learn and has simple syntax.

3.3.3 NumPy(Python Library):

NumPy is the fundamental package for scientific computing in Python. It is a Python library that provides a multidimensional array object, various derived objects (such as masked arrays and matrices), and an assortment of routines for fast operations on arrays, including mathematical, logical, shape manipulation, sorting, selecting, I/O, discrete Fourier transforms, basic linear algebra, basic statistical operations, random simulation and much more.

3.3.4 Pandas(Python Library):

Pandas is a software library written for the Python programming language for data manipulation and analysis. In particular, it offers data structures and operations for manipulating numerical tables and time series. It is free software released under the three-clause BSD license. The name is derived from the term "panel data", an econometrics term for data sets that include observations over multiple time periods for the same individuals. Its name is a play on the phrase "Python data analysis" itself. Wes McKinney started building what would become pandas at AQR Capital while he was a researcher there from 2007 to 2010.

3.3.5 Scikit-Learn(Python Library):

Scikit-learn (formerly scikits.learn and also known as sklearn) is a free software machine learning library for the Python programming language. It features various classification, regression and clustering algorithms including support vector machines, random forests, gradient boosting, *k*-means and DBSCAN, and is designed to interoperate with the Python numerical and scientific libraries NumPy and SciPy.

3.3.6 Keras(Python Library):

Keras is an open-source software library that provides a Python interface for artificial neural networks. Keras acts as an interface for the TensorFlow library.

Keras contains numerous implementations of commonly used neural-network building blocks such as layers, objectives, activation functions, optimizers, and a host of tools to make working with image and text data easier to simplify the coding necessary for writing deep neural network code. The code is hosted on GitHub, and community support forums include the GitHub issues page, and a Slack channel

.

3.3.7 Tensorflow (Python Library):

TensorFlow is an open source machine learning framework for all developers. It is used for implementing machine learning and deep learning applications. To develop and research on fascinating ideas on artificial intelligence, The Google team created TensorFlow. TensorFlow is designed in Python programming language, hence it is considered an easy to understand Framework.

3.4 DATAFLOW DIAGRAM

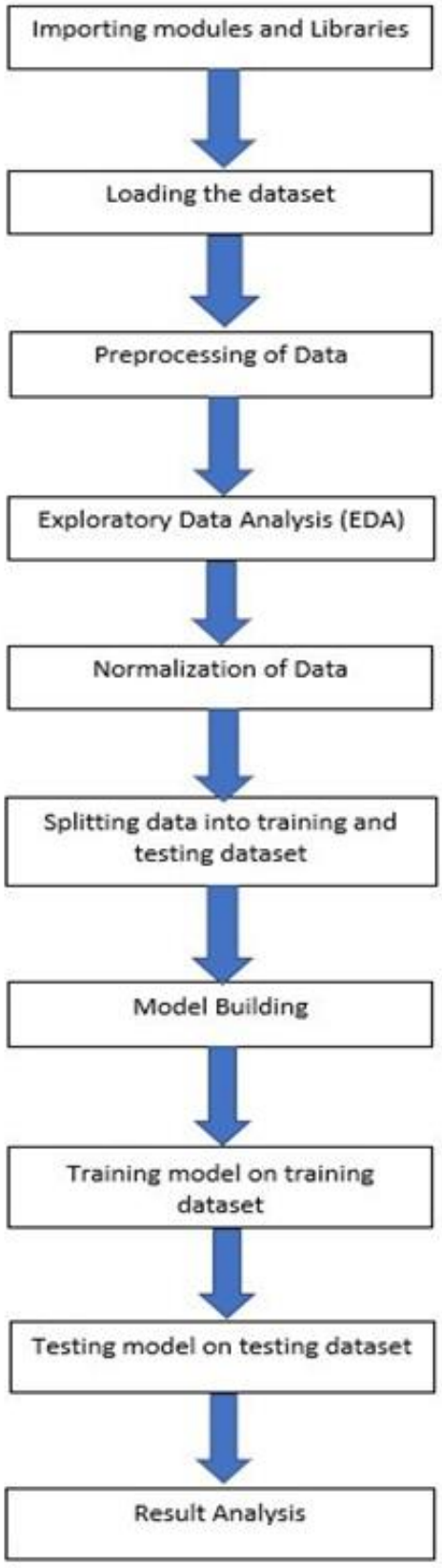


Fig. 7. Data Flow Diagram

CHAPTER-4

METHODS AND METHODOLOGY

4.1 DATASET DESCRIPTION

We have downloaded the dataset from Kaggle.

- **Following are the 5 yoga poses**
    - Downward Dog Pose (Adho Mukha Svanasana)
    - Goddess Pose(Utkata Konasana)
    - Plank
    - Tree Pose(Vrksasana)
    - Warrior II Pose(Virabhadrasana II)

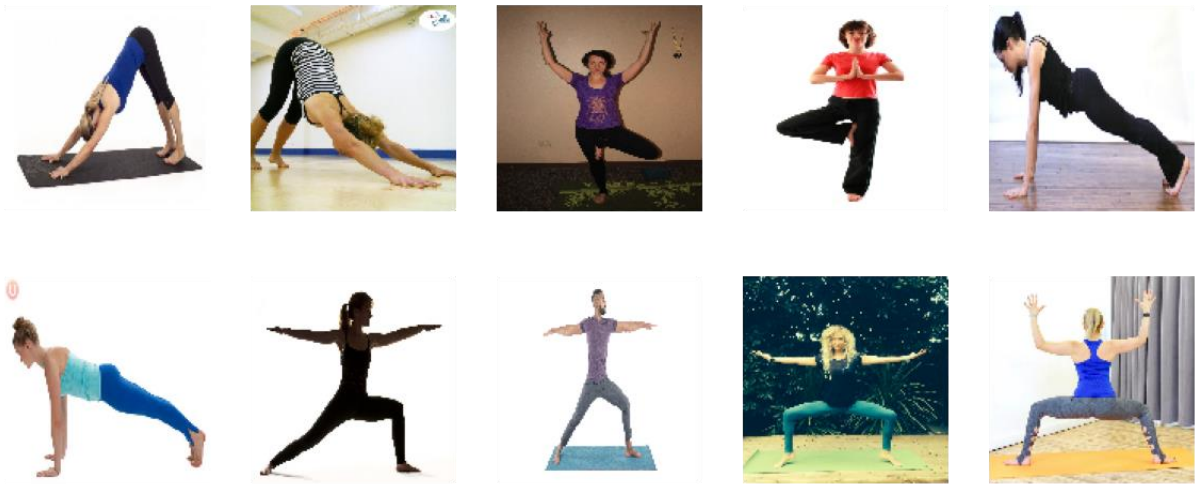


Fig.8. Yoga Poses

4.2 SPLITTING THE DATA INTO TRAIN AND TEST SETS

Why do we need to split the data into Training, Testing & Validation sets?

We don't want our model to over-learn from training data and perform poorly after being deployed in production. We need to have a mechanism to assess how well our model is generalizing. Hence, we need to separate your input data into training, validation, and testing subsets to prevent our model from overfitting and to evaluate our model effectively.

Why even prevent Overfitting in the first place?

Overfitting is a concept in data science, which occurs when a statistical model fits exactly against its training data. When this happens, the algorithm unfortunately cannot perform accurately against unseen data, defeating its purpose. Generalization of a model to new data is ultimately what allows us to use machine learning algorithms every day to make predictions and classify data.

- ***Training data:*** This type of data builds up the machine learning algorithm. The data scientist feeds the algorithm input data, which corresponds to an expected output. The model evaluates the data repeatedly to learn more about the data's behavior and then adjusts itself to serve its intended purpose. The actual dataset that we use to train the model (weights and biases in the case of a Neural Network). The model *sees* and *learns* from this data.

- ***Test data:*** After the model is built, testing data once again validates that it can make accurate predictions. If training and validation data include labels to monitor performance metrics of the model, the testing data should be unlabeled. Test data provides a final, real-world check of an unseen dataset to confirm that the ML algorithm was trained effectively. The Test dataset provides the gold standard used to evaluate the model. It is only used once a model is completely trained(using the train and validation sets). The test set is generally what is used to evaluate competing models (For example on many Kaggle competitions, the validation set is released initially along with the training set and the actual test set is only released when the competition is about to close, and it is the result of the the model on the Test set that decides the winner). Many a times the validation set is used as the test set, but it is not good practice. The test set is generally well curated. It contains carefully sampled data that spans the various classes that the model would face, when used in the real world**.**

- ***Validation data****:* During training, validation data infuses new data into the model that it hasn't evaluated before. Validation data provides the first test against unseen data, allowing data scientists to evaluate how well the model makes predictions based on the new data. Not all data scientists use validation data, but it can provide some helpful information to optimize hyperparameters, which influence how the model assesses data. The validation set is used to evaluate a given model, but this is for frequent evaluation.

We, as machine learning engineers, use this data to fine-tune the model hyperparameters. Hence the model occasionally *sees* this data, but never does it "*Learn*" from this. We use the validation set results, and update higher level hyperparameters. So the validation set affects a model, but only indirectly. The validation set is also known as the Dev set or the Development set. This makes sense since this dataset helps during the "development" stage of the model.

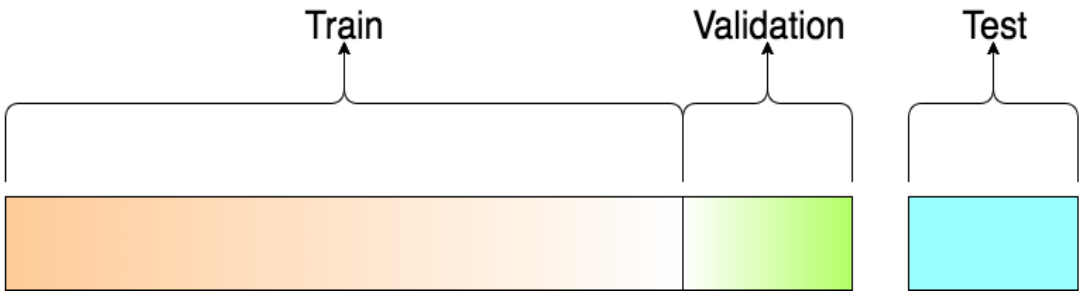
- 

Fig.9. A visualization of the splits

About the dataset split ratio

Now that you know what these datasets do, you might be looking for recommendations on how to split your dataset into Train, Validation and Test sets.

This mainly depends on 2 things. First, the total number of samples in your data and second, on the actual model you are training.

Some models need substantial data to train upon, so in this case you would optimize for the larger training sets. Models with very few hyperparameters will be easy to validate and tune, so you can probably reduce the size of your validation set, but if your model has many hyperparameters, you would want to have a large validation set as well(although you should also consider cross validation). Also, if you happen to have a model with no hyperparameters or ones that cannot be easily tuned, you probably don't need a validation set too!

All in all, like many other things in machine learning, the train-test-validation split ratio is also quite specific to your use case and it gets easier to make judge ment as you train and build more and more models.

Note on Cross Validation: Many a times, people first split their dataset into 2 — Train and Test. After this, they keep aside the Test set, and randomly choose X% of their Train dataset to be the actual **Train** set and the remaining (100-X)% to be the **Validation** set, where X is a fixed number(say 80%), the model is then iteratively trained and validated on these different sets. There are multiple ways to do this, and is commonly known as Cross Validation. Basically you use your training set to generate multiple splits of the Train and Validation sets. Cross validation avoids over fitting and is getting more and more popular, with K-fold Cross Validation being the most popular method of cross validation.

4.3 DATA AUGMENTATION

Data augmentation in data analysis are techniques used to increase the amount of data by adding slightly modified copies of already existing data or newly created synthetic data from existing data. It acts as a regularizer and helps reduce overfitting when training a machine learning model. It is closely related to oversampling in data analysis. If a dataset is very small, then a version augmented with rotation and mirroring etc. may still not be enough for a given problem. Another solution is the sourcing of entirely new, synthetic images through various techniques, for example the use of generative adversarial networks to create new synthetic images for data augmentation. Additionally, image recognition algorithms show improvement when transferring from images rendered in virtual environments to real-world data. In general, **DA** is frequently used when building a **DL** model. That is why throughout this article we will mostly talk about performing **Data Augmentation** with various **DL** frameworks. Still, you should keep in mind that you can augment the data for the **ML** problems as well.

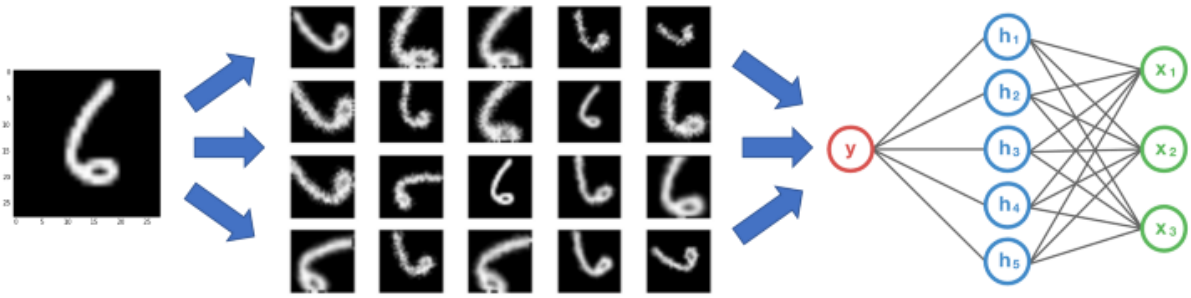


Fig.10. Data Augmentation in play

Oftentimes, when working on specific complex tasks such as classifying a weed from a crop, or identifying the novelty of a patient, it is very hard to get large amounts of data required to train the models. Though transfer learning techniques could be used to great effect, the challenges involved in making a pre-trained model to work for specific tasks are tough.

Another way to deal with the problem of limited data is to apply different transformations on the available data to synthesize new data. This approach of synthesizing new data from the available data is referred to as 'Data Augmentation'.

Data augmentation can be used to address both the requirements, the diversity of the training data, and the amount of data. Besides these two, augmented data can also be used to address the class imbalance problem in classification tasks.

4.3.1 Popular Augmentation Techniques

In this section, we present some basic but powerful augmentation techniques that are popularly used. Before we explore these techniques, for simplicity, let us make one assumption. The assumption is that, we don't need to consider what lies beyond the image's boundary. We'll use the below techniques such that our assumption is valid.

What would happen if we use a technique that forces us to guess what lies beyond an image's boundary? In this case, we need to interpolate some information. We'll discuss this in detail after we cover the types of augmentation.

For each of these techniques, we also specify the factor by which the size of your dataset would get increased (aka. Data Augmentation Factor)

4.3.1 flip: You can flip images horizontally and vertically. Some frameworks do not provide function for vertical flips. But, a vertical flip is equivalent to rotating an image by 180 degrees and then performing a horizontal flip. Below are examples for images that are flipped.

4.3.2 Rotation: One key thing to note about this operation is that image dimensions may not be preserved after rotation. If your image is a square, rotating it at right angles will preserve the image size. If it's a rectangle, rotating it by 180 degrees would preserve the size. Rotating the image by finer angles will also change the final image size. We'll see how we can deal with this issue in the next section. Below are examples of square images rotated at right angles.

4.3.3 Scale: The image can be scaled outward or inward. While scaling outward, the final image size will be larger than the original image size. Most image frameworks cut out a section from the new image, with size equal to the original image. We'll deal with scaling inward in the next section, as it reduces the image size, forcing us to make assumptions about what lies beyond the boundary. Below are examples or images being scaled.

4.3.4 Crop: Unlike scaling, we just randomly sample a section from the original image. We then resize this section to the original image size. This method is popularly known as random cropping. Below are examples of random cropping. If you look closely, you can notice the difference between this method and scaling.

4.3.5 Translation: Translation just involves moving the image along the X or Y direction (or both). In the following example, we assume that the image has a black background beyond its boundary, and are translated appropriately. This method of augmentation is very useful as most **objects** can be located at **almost anywhere** in the image. This **forces** your **convolutional neural network to look everywhere**.

4.3.6 Gaussian Noise: Over-fitting usually happens when your neural network tries to learn high frequency features (patterns that occur a lot) that may not be useful. Gaussian noise, which has zero mean, essentially has data points in all frequencies, effectively distorting the high frequency features. This also means that lower frequency components (usually, your intended data) are also distorted, but your neural network can learn to look past that. Adding just the right amount of noise can enhance the learning capability.

A toned down version of this is the salt and pepper noise, which presents itself as random black and white pixels spread through the image. This is similar to the effect produced by adding Gaussian noise to an image, but may have a lower information distortion level.

Fig.11. From the left, we have the original image, image with added Gaussian noise, image with added salt and pepper noise

4.4 Code Snippets

4.4.1 Loading the Training & Testing Dataset

```
train_dir = '../content/DATASET/TRAIN'

test_dir = '../content/DATASET/TEST'
```

4.4.2 Data Augmentation

```
train_datagen = ImageDataGenerator(width_shift_range=0.1,

horizontal_flip = True, rescale = 1/255, validation_split = 0.2)

test_datagen = ImageDataGenerator(rescale = 1/255, validation_split = 0.2)
```

4.4.3 Separating Training & Validation Data

```
train_generator = train_datagen.flow_from_directory(directory = train_dir, target_size = (224,224), color_mode = 'rgb', class_mode = 'categorical', batch_size = 16, subset = 'training')

validation_generator = test_datagen.flow_from_directory(directory = test_dir, target_size = (224,224), color_mode = 'rgb', class_mode = 'categorical', subset = 'validation')
```

4.4.4 Creating the Neural Network/CNN Model

```
model = tf.keras.models.Sequential([

tf.keras.layers.Conv2D(64, (3,3), activation='relu', padding = 'Same', input_shape = (320, 320, 3)),

tf.keras.layers.MaxPooling2D(2, 2),

tf.keras.layers.Dropout(0.25),

tf.keras.layers.Conv2D(128, (3,3), activation='relu',padding = 'Same'), tf.keras.layers.MaxPooling2D(2,2),

tf.keras.layers.Dropout(0.25),
```

```
tf.keras.layers.Conv2D(256, (3,3), activation='relu',padding = 'Same'),
tf.keras.layers.MaxPooling2D(2,2),

tf.keras.layers.Dropout(0.25),

tf.keras.layers.Flatten(),

tf.keras.layers.Dense(1024, activation='relu'),

tf.keras.layers.Dropout(0.5), tf.keras.layers.Dense(5, activation='softmax')])

optimizer = Adam(lr=0.001)

model.compile(loss='categorical_crossentropy', optimizer = optimizer,

metrics=['accuracy'])

epochs = 50

batch_size = 16

model.summary()
```

4.4.5 Training the ML model

```
history = model.fit(train_generator, epochs = epochs, validation_data = validation_generator, verbose=1)
```

4.4.6 Plots for Training & Validation Accuracy

```
fig , ax = plt.subplots(1,2)

train_acc = history.history['accuracy']

train_loss = history.history['loss']

fig.set_size_inches(12,4)

ax[0].plot(history.history['accuracy'])

ax[0].plot(history.history['val_accuracy'])
```

```
ax[0].set_title('Training Accuracy vs Validation Accuracy')
ax[0].set_ylabel('Accuracy')
ax[0].set_xlabel('Epoch')
ax[0].legend(['Train', 'Validation'], loc='upper left')
ax[1].plot(history.history['loss'])
ax[1].plot(history.history['val_loss'])
ax[1].set_title('Training Loss vs Validation Loss')
ax[1].set_ylabel('Loss')
ax[1].set_xlabel('Epoch')
ax[1].legend(['Train', 'Validation'], loc='upper left')
plt.show()
```

4.4.7 Saving the Trained Model

```
model.save('YogaNet_mode.h5')
```

4.4.8 Create a class to access the WebCam

```
import cv2
import time
class CameraUse():
    def UseCamera():
        TIMER = int(1)
        cap = cv2.VideoCapture(0)
        while True:
```

```
ret, img = cap.read()
cv2.imshow('a', img)
k = cv2.waitKey(10)
if True:
    prev = time.time()
    while TIMER >= 0:
        ret, img = cap.read()
        font = cv2.FONT_HERSHEY_SIMPLEX
        cv2.putText(img, str(TIMER),
                    (200, 250), font,
                    7, (0, 255, 255),
                    4, cv2.LINE_AA)
        cv2.imshow('a', img)
        cv2.waitKey(10)
        cur = time.time()
        if cur-prev >= 1:
            prev = cur
            TIMER = TIMER-1
    else:
        ret, img = cap.read()
        cv2.imshow('a', img)
```

```
                cv2.imwrite('camera.jpg', img)
                break
            elif k == 27:
                break
        cap.release()
        cv2.destroyAllWindows()
    UseCamera()
```

4.4.9 Driver Code to Execute the WebCam and Trained model

```
    from tensorflow.keras.models import load_model
    import cv2
    import numpy as np
    import collections
    import tensorflow as tf
    import CameraUse
    import time
    model=load_model("YogaNet_Model_1_1.h5")
    try:
        collectionsAbc = collections.abc
    except AttributeError:
        collectionsAbc = collections
```

```
dictn = {1:"Downdog", 2:"Goddess", 3:"Plank", 4:"Tree", 5:"Warrior 2" }

model.compile(loss='binary_crossentropy', optimizer='rmsprop', metrics=['accuracy'])


def predictPic(img):
  img = cv2.resize(img,(320,320))
  img = np.reshape(img,[1,320,320,3])
  pred = model.predict(img)
  listt = pred[0].tolist()
  for i in range(0, len(listt)):
    listt[i] = int(listt[i])
  return listt


img = cv2.imread('camera.jpg')
listt = predictPic(img)
for i in range(len(listt)):
  if listt[i]==1:
    abc = "Pose: " + str(dictn[i+1]) + " || Accuracy :" + str(percent)
cv2.imshow(abc, img)
cv2.waitKey()
print(abc)
```

CHAPTER-5

RESULTS

5.1 DATASET VISUALIZATION

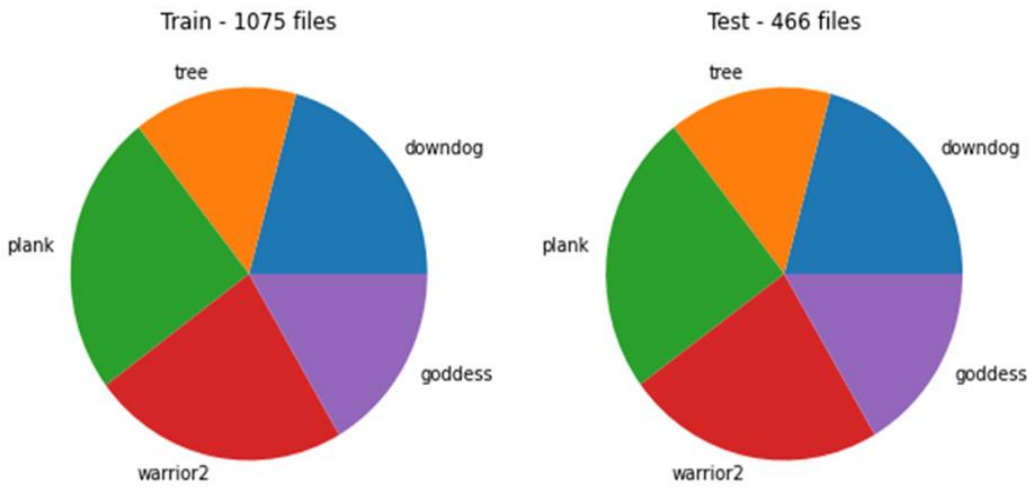


Fig. 12. Dataset Visualization

5.2 SUMMARY

```
model.summary()

Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 224, 224, 64)      1792

 max_pooling2d (MaxPooling2D  (None, 112, 112, 64)     0
 )

 dropout (Dropout)           (None, 112, 112, 64)      0

 conv2d_1 (Conv2D)           (None, 112, 112, 128)     73856

 max_pooling2d_1 (MaxPooling  (None, 56, 56, 128)      0
 2D)

 dropout_1 (Dropout)         (None, 56, 56, 128)       0

 conv2d_2 (Conv2D)           (None, 56, 56, 256)       295168

 max_pooling2d_2 (MaxPooling  (None, 28, 28, 256)      0
 2D)

 dropout_2 (Dropout)         (None, 28, 28, 256)       0

 flatten (Flatten)           (None, 200704)            0

 dense (Dense)               (None, 1024)              205521920

 dropout_3 (Dropout)         (None, 1024)              0

 dense_1 (Dense)             (None, 5)                 5125

=================================================================
Total params: 205,897,861
Trainable params: 205,897,861
Non-trainable params: 0
_________________________________________________________________
```

Fig. 13. Summary of the CNN model

```
[ ] history = model.fit(train_generator, epochs = epochs,validation_data = validation_generator)

Epoch 1/50
 5/55 [=>............................] - ETA: 26s - loss: 24.7623 - accuracy: 0.2125/usr/local/lib/python3.7/dist-packages/PIL/Image.py:960: UserWarning: Palette images with Transparency expressed
  "Palette images with Transparency expressed in bytes should be "
55/55 [==============================] - 50s 704ms/step - loss: 3.7388 - accuracy: 0.2656 - val_loss: 1.5899 - val_accuracy: 0.2826
Epoch 2/50
55/55 [==============================] - 34s 608ms/step - loss: 1.5507 - accuracy: 0.2852 - val_loss: 1.4315 - val_accuracy: 0.3370
Epoch 3/50
55/55 [==============================] - 33s 604ms/step - loss: 1.4473 - accuracy: 0.3811 - val_loss: 1.1987 - val_accuracy: 0.5543
Epoch 4/50
55/55 [==============================] - 33s 609ms/step - loss: 1.3731 - accuracy: 0.4388 - val_loss: 1.1517 - val_accuracy: 0.6522
Epoch 5/50
55/55 [==============================] - 33s 596ms/step - loss: 1.2850 - accuracy: 0.4873 - val_loss: 0.8619 - val_accuracy: 0.6957
Epoch 6/50
55/55 [==============================] - 33s 595ms/step - loss: 1.1603 - accuracy: 0.5485 - val_loss: 0.7695 - val_accuracy: 0.7174
Epoch 7/50
55/55 [==============================] - 33s 596ms/step - loss: 1.0702 - accuracy: 0.6109 - val_loss: 0.5265 - val_accuracy: 0.8696
Epoch 8/50
55/55 [==============================] - 33s 605ms/step - loss: 0.9279 - accuracy: 0.6709 - val_loss: 0.5725 - val_accuracy: 0.8043
Epoch 9/50
55/55 [==============================] - 33s 603ms/step - loss: 0.8572 - accuracy: 0.6998 - val_loss: 0.3858 - val_accuracy: 0.9022
Epoch 10/50
55/55 [==============================] - 33s 599ms/step - loss: 0.8181 - accuracy: 0.7125 - val_loss: 0.3532 - val_accuracy: 0.8804
Epoch 11/50
55/55 [==============================] - 33s 607ms/step - loss: 0.6787 - accuracy: 0.7517 - val_loss: 0.3725 - val_accuracy: 0.8587
Epoch 12/50
55/55 [==============================] - 33s 604ms/step - loss: 0.6603 - accuracy: 0.7702 - val_loss: 0.4082 - val_accuracy: 0.8696
Epoch 13/50
55/55 [==============================] - 33s 603ms/step - loss: 0.6200 - accuracy: 0.7806 - val_loss: 0.3056 - val_accuracy: 0.9130
Epoch 14/50
55/55 [==============================] - 33s 603ms/step - loss: 0.5389 - accuracy: 0.8037 - val_loss: 0.2795 - val_accuracy: 0.9239
Epoch 15/50
55/55 [==============================] - 33s 605ms/step - loss: 0.5138 - accuracy: 0.8314 - val_loss: 0.2765 - val_accuracy: 0.8913
Epoch 16/50
55/55 [==============================] - 33s 607ms/step - loss: 0.4564 - accuracy: 0.8406 - val_loss: 0.2425 - val_accuracy: 0.9022
Epoch 17/50
55/55 [==============================] - 33s 604ms/step - loss: 0.3763 - accuracy: 0.8661 - val_loss: 0.3072 - val_accuracy: 0.8696
Epoch 18/50
55/55 [==============================] - 33s 601ms/step - loss: 0.3866 - accuracy: 0.8649 - val_loss: 0.2611 - val_accuracy: 0.8913
Epoch 19/50
55/55 [==============================] - 34s 611ms/step - loss: 0.3820 - accuracy: 0.8718 - val_loss: 0.2679 - val_accuracy: 0.8696
Epoch 20/50
55/55 [==============================] - 34s 613ms/step - loss: 0.3196 - accuracy: 0.8949 - val_loss: 0.3526 - val_accuracy: 0.8804
Epoch 21/50
55/55 [==============================] - 34s 612ms/step - loss: 0.2982 - accuracy: 0.9007 - val_loss: 0.3533 - val_accuracy: 0.9022
Epoch 22/50
55/55 [==============================] - 33s 607ms/step - loss: 0.2895 - accuracy: 0.9065 - val_loss: 0.3605 - val_accuracy: 0.8696
Epoch 23/50
55/55 [==============================] - 34s 604ms/step - loss: 0.2153 - accuracy: 0.9296 - val_loss: 0.3028 - val_accuracy: 0.8913
Epoch 24/50
55/55 [==============================] - 33s 603ms/step - loss: 0.1801 - accuracy: 0.9319 - val_loss: 0.2855 - val_accuracy: 0.8913
Epoch 25/50
55/55 [==============================] - 34s 609ms/step - loss: 0.2000 - accuracy: 0.9296 - val_loss: 0.5228 - val_accuracy: 0.8478
Epoch 26/50
55/55 [==============================] - 34s 610ms/step - loss: 0.2321 - accuracy: 0.9365 - val_loss: 0.3807 - val_accuracy: 0.8804
Epoch 27/50
55/55 [==============================] - 33s 607ms/step - loss: 0.1951 - accuracy: 0.9319 - val_loss: 0.4332 - val_accuracy: 0.8587
Epoch 28/50
55/55 [==============================] - 33s 609ms/step - loss: 0.2436 - accuracy: 0.9296 - val_loss: 0.3775 - val_accuracy: 0.9130
```

Fig. 14. Model Training

5.3 ACCURACY

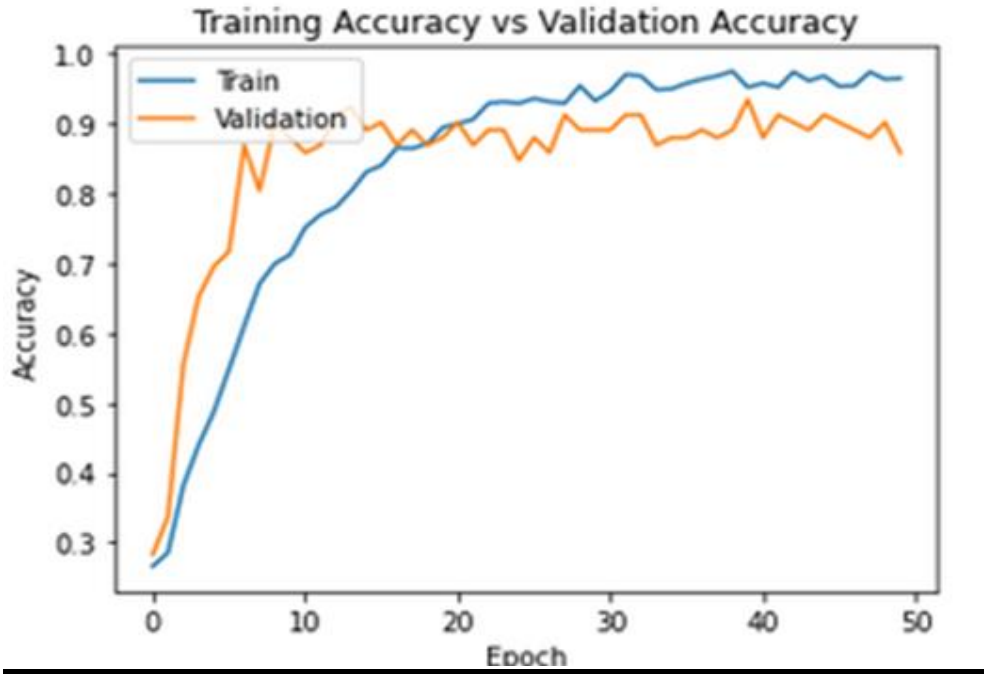


Fig. 15. Training V/S Validation Accuracy

Fig. 16. Training V/S Validation Loss

5.4 OUTPUT

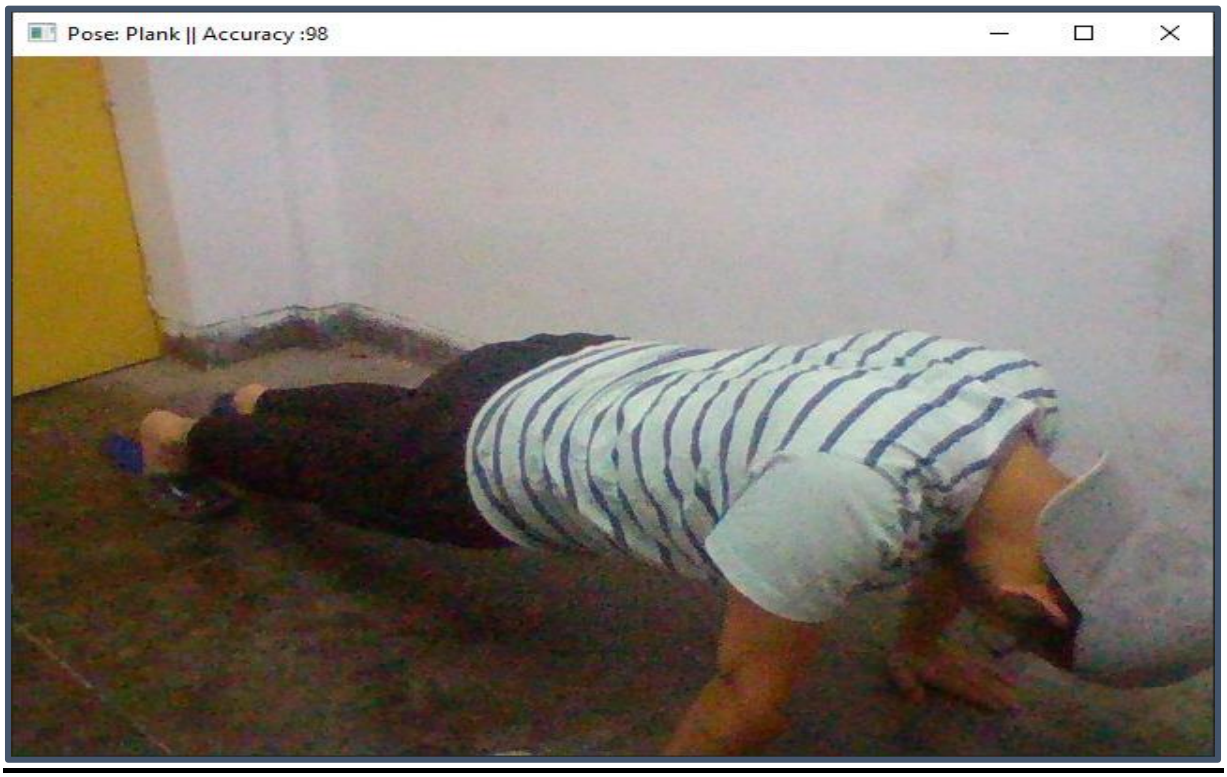


Fig. 17. Output of the Driver Code 1

Fig.18. Output of the Driver Code 2

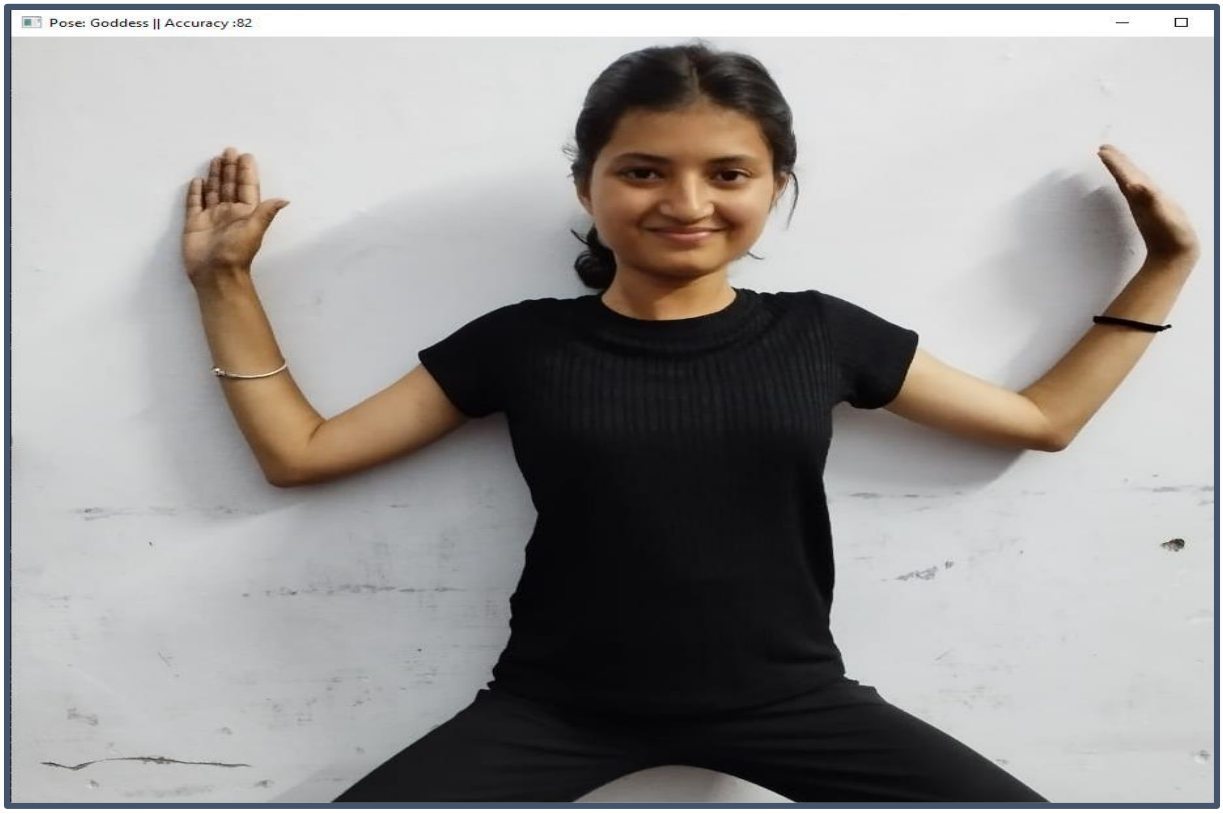


Fig.19. Output of the Driver Code 3

Fig.20. Output of the Driver Code 4

Fig.21. Output of the Driver Code 5

CHAPTER-6

CONCLUSION

Human posture assessment has been concentrated widely over the previous years. When contrasted with other PC vision issues, human posture assessment is distinctive as it needs to limit and amass human body parts based on an effectively characterized structure of the human body. Use of posture assessment in wellness and sports can help forestall wounds and improve the execution of individuals' exercise. [3] recommends, yoga selfguidance frameworks convey the potential to make yoga famous alongside ensuring it is acted in the correct way. The utilization of CNN is profoundly successful and arranges all the 5 yoga presents impeccably. A fundamental CNN perform well past our desires.

6.1 FUTURE WORK

The proposed models right now characterize just 6 yoga asanas. There are various yoga asanas, and subsequently making a posture assessment model that can be effective for all the asanas is a testing issue. The dataset can be extended my adding more yoga presents performed by people in indoor setting as well as open air. The exhibition of the models depends upon the nature of OpenPose present assessment which may not perform well in instances of cover between individuals or cover between body parts. A convenient gadget for self-preparing and constant forecasts can be executed for this framework. This work exhibits movement acknowledgment for reasonable applications. A methodology practically identical to this can be used for present acknowledgment in undertakings for example, sports, reconnaissance, medical services and so forth Multiindividual posture assessment is a totally different issue in itself and has a great deal of degree for research. There are a ton of situations where single individual posture assessment would not get the job done, for instance present assessment in jam-packed situations would have various people which will include following and distinguishing posture of every person. A great deal of factors, for example, foundation, lighting, covering figures and so on which have been talked about before in this overview would additionally make multi-individual posture assessment testing

REFERENCES

[1]. L. Sigal. “Human pose estimation”, Ency. of Comput. Vision, Springer 2011.

[2]. S. Yadav, A. Singh, A. Gupta, and J. Raheja, “Real-time yoga recognition using deep learning”, Neural Comput. and Appl., May 2019.Online].

[3]. U. Rafi, B. Leibe, J.Gall, and I. Kostrikov, “An efficient convolutional network for human pose estimation”, British Mach. Vision Conf., 2016.

[4]. S. Haque, A. Rabby, M. Laboni, N. Neehal, and S. Hossain, “ExNET: deep neural network for exercise pose detection”, Recent Trends in Image Process. and Pattern 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]. S. Patil, A. Pawar, and A. Peshave, “Yoga tutor: visualization and analysis using SURF algorithm”,Proc. IEEE Control Syst. Graduate Research Colloq.,pp. 43-46, 2011.

[7]. W. Gong, X. Zhang, J. Gonzàlez, A. Sobral, T. Bouwmans, C. Tu, and H. Zahzah, “Human pose estimation from monocular images: a comprehensive survey”, Sensors, Basel, Switzerland, vol. 16, 2016.

[8].. G. Ning, P. Liu, X. Fan and C. Zhan, “A topdown approach to articulated human pose estimation and tracking”, ECCV Workshops, 2018.

[9].. A. Gupta, T. Chen, F. Chen, and D. Kimber, “Systems and methods for human body pose estimation”, U.S. patent, 7,925,081 B2, 2011.

[10]. H. Sidenbladh, M. Black, and D. Fleet, “Stochastic tracking of 3D human figures using 2D image motion”, Proc 6th European Conf. Computer Vision, 2000.

[11]. A. Agarwal and B. Triggs, “3D human pose from silhouettes by relevance vector regression”, Intl Conf. on Computer Vision & Pattern Recogn.pp.882–888, 2004.

[12]. M. Li, Z. Zhou, J. Li and X. Liu, “Bottom-up pose estimation of multiple person with bounding box constraint”, 24th Intl. Conf. Pattern Recogn.,2018.

[13]. Z. Cao, T. Simon, S. Wei, and Y. Sheikh, “OpenPose: realtime multi-person 2D pose estimation using part affinity fields”, Proc. 30th IEEE Conf. Computer Vision and Pattern Recogn,2017.

[14]. A. Kendall, M. Grimes, R. Cipolla, “PoseNet: a convolutional network for real-time 6DOF camera relocalization”, IEEE Intl. Conf. Computer Vision, 2015.

[15]. S. Kreiss, L. Bertoni, and A. Alahi, “PifPaf: composite fields for human pose estimation”, IEEE Conf. Computer Vision and Pattern Recogn, 2019.

[16]. P. Dar, “AI guardman – a machine learning application that uses pose estimation to detect shoplifters”.Online]

[17]. D. Mehta, O. Sotnychenko, F. Mueller and W. Xu, “XNect: real-time multi-person 3D human pose estimation with a single RGB camera”, ECCV, 2019.

[18]. A. Lai, B. Reddy and B. Vlijmen, “Yog.ai: deep learning for yoga”.Online].

© Copyright GOVIND BALLABH PANT INSTITUTE OF ENGINEERING AND TECHNOLOGY,

PAURI GARHWAL

JUNE-2022

ALL RIGHTS RESERVED