Yoga Posture Recognition Using the Deep Learning Process

By

Abidul Islam
18301144

Zarif Sadman
18301008

Shah Imran
18301149

Sazzadul Islam
18301162

A thesis submitted to the Department of Computer Science and Engineering in partial fulfillment of the requirements for the degree of B.Sc. in Computer Science

Department of Computer Science and Engineering
Brac University

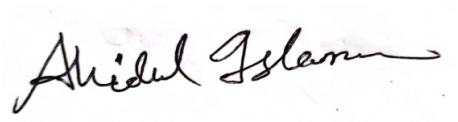
May 2022

Declaration

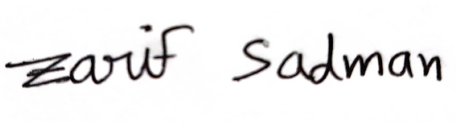
It is hereby declared that

1. The thesis submitted is our own original work while completing degree at Brac University.
2. The thesis does not contain material previously published or written by a third party, except where this is appropriately cited through full and accurate referencing.
3. The thesis does not contain material which has been accepted, or submitted, for any other degree or diploma at a university or other institution.
4. We have acknowledged all main sources of help.

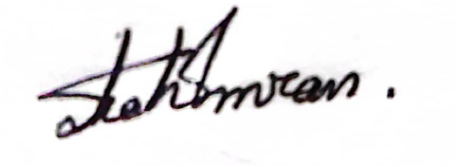
Student's Full Name & Signature:



Abidul Islam
18301144



Zarif Sadman
18301008



Shah Imran
18301149

Sazzadul Islam
18301162

Approval

The thesis titled "Yoga Posture Recognition Using the Deep Learning Process" submitted by

1. Abidul Islam (18301144)
2. Zarif Sadman (18301008)
3. Shah Imran (18301149)
4. Sazzadul Islam (18301162)

Of Spring, 2022 has been accepted as satisfactory in partial fulfillment of the requirement for the degree of B.Sc. in Computer Science on May 26, 2022.

Examining Committee:

Supervisor:

(Member)

Faisal Bin Ashraf
Lecturer, Department of Computer Science and Engineering
Brac University

Program Coordinator:
(Member)

Dr. Md. Golam Rabiul Alam
Associate Professor
Department of Computer Science and Engineering
Brac University

Head of Department:
(Chair)

Dr. Sadia Hamid Kazi
Chairperson & Associate Professor
Department of Computer Science and Engineering
Brac University

Abstract

Yoga is one of the best activities from home to preserve our physical condition in the present epidemic. Yoga, on the other hand, is all about performing the 82 Yoga Asanas correctly over the course of six classes. Regrettably, not everyone have the knowledge or can perform yoga accurately. So to do yoga poses correctly we will have to find a yoga instructor, but it can be very hard and expensive to find yoga instructors considering all possible general situation and status. Using Deep Learning(DL), picture categorization and various machine learning approaches, we attempted to build a system or a model that will operate as a self-instructor of Yoga for the user to classify different poses of yoga to distinguish accurate pose in our thesis. It will assist the user in performing Yoga correctly by recognizing errors in their Yoga Asanas. In a nutshell, this section will cover several posture estimation, key point detection, and pose categorization techniques. Moreover, we tried ensemble modeling as a booster to improve the pose prediction accuracy as much as possible.

Dedication

This thesis paper is dedicated to the people who are suffering from mental health issues, to all the drug addicts who want to live a healthy life and want to come back from the the dark path. Also we dedicate the paper to those people who want to practice yoga but can not afford the classes as they are quite expensive.

Acknowledgement

Firstly, Alhamdulillah for everything we have done so far in this thesis paper. Secondly, we are thankful to our supervisor, Mr. Faisal Bin Ashraf who guided us to the proper direction to fulfil the paper. And finally, thanks to the family and friends who supported us all the way of our life.

Table of Contents

Declaration i

Approval ii

Abstract iii

Dedication iv

Acknowledgment v

Table of Contents vi

List of Figures viii

List of Tables 1

1 Introduction 2
- 1.1 Human Activity Recognition & Yoga . . . 2
- 1.2 Problem Statement . . . 3
- 1.3 Research Objectives . . . 4

2 Previous Works 5

3 Data Collection 12

4 Data Pre-processing 17

5 Models & Classifiers 19
- 5.0.1 DenseNet201 . . . 19
- 5.0.2 Xception . . . 20
- 5.0.3 MobileNets . . . 21
- 5.0.4 ResNet50 & ResNet101 . . . 22
- 5.0.5 Ensemble modeling . . . 24
- 5.0.6 Random Forest Classifier . . . 25

6 Result Analysis 27
- 6.0.1 Result . . . 27
- 6.0.2 Analysis . . . 31

7 Discussion 32

8 Conclusion **34**

9 Future Plan **35**

Bibliography **37**

List of Figures

1.1 Yoga pose(Vrikshasana) 2
1.2 Upword Bow Pose 3
1.3 Upward Facing Two Foot Staff Pose 3

4.1 Pre-proccess 17

5.1 DenseNet201 19
5.2 5 layer Dense Block 20
5.3 Xception Work Flow 21
5.4 MobileNet 21
5.5 The standard convolutional filters in (a) are replaced by two layers: depthwise convolution in (b) and pointwise convolution in (c) to build a depthwise separable filter. 22
5.6 Different Layers of ResNet50 23
5.7 ResNet architecture 23
5.8 error rate of different keras models 24
5.9 bias-variance tradeoff 25
5.10 Random Forest diagram 26

6.1 Accuracy for 82 classes 27
6.2 Accuracy for 6 basic classes 27
6.3 Classification report 30

List of Tables

2.1 Paper review summary table: 11

3.1 Sample of poses we are working on: 12
3.2 Sample of poses we are working on: 13
3.3 Sample of poses we are working on: 14
3.4 Sample of poses we are working on: 15
3.5 Sample of poses we are working on: 16

5.1 Comparison of Xception: 20

Chapter 1

Introduction

1.1 Human Activity Recognition & Yoga

"Man is by natural a social animal"-said by, Aristotle the legendary Greek philosopher. As human are social being then human-to-human interaction and interpersonal relationship is quite regular in our world and to maintain this interaction and relationship human activity recognition plays an important role. This is because it provides information about the personality, identity and the mental or psychological state of a person. The ability of one human to recognize another human's activities is one of the primary subject of the study of computer vision and machine learning. Human activity recognition can locate few key points and can figure out the specific body posture that represents those key points through sensor data. By the use of this technology yoga postures also can be recognized.

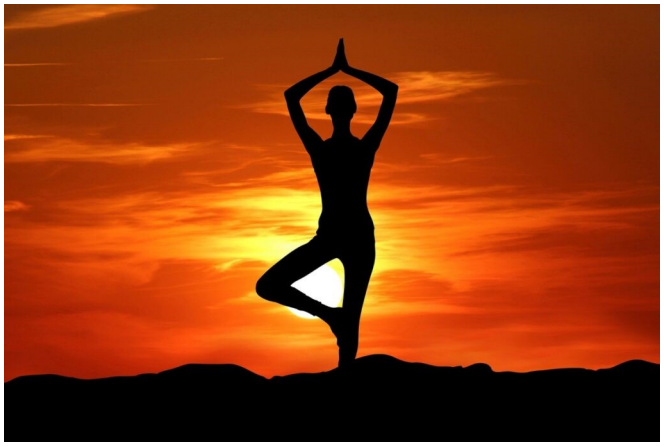


Figure 1.1: Yoga pose(Vrikshasana)

Yoga is basically a spiritual discipline which is based on tremendously tenuous science. Yoga concentrates on bringing the harmony between the human body and mind. Yoga is a collective exercise for human body and mind and therefore, it possesses multiple number of postures. The birth place of this yoga is India. As yoga is a type of exercise, this also being used as therapy which is gaining more popularity day by day and this is also being used as an alternative of modern medicines. According to multiple researches and studies, yoga can put a massive impact on both human body and mind. If yoga is performed correctly then, it is possible to use it as

treatment for stress related disorder such as hypertension, psychosomatic disorder, back pain, irritated bowel syndrome, diabetes, asthma and many others. Yoga can be a huge help for the victims of addiction(drug addiction) to return to their regular life as yoga improves a person's ability to self-control, as well as their willpower and helps them realize that they have a whole new reason to live life to the fullest. Yoga is also beneficial for the treatment of Parkinson disease, Dementia, Alzheimer etc. If the yoga is not performed correctly then it can cause serious damage to the physical health of the user so, it is mandatory to perform yoga asanas as correctly as possible.

1.2 Problem Statement

Yoga is a collective exercise for human body and mind has a significant influence on the human body and mind therefore, doing it correctly and on time is mandatory, as practicing it in the improper position can cause serious harm to a person's body. Furthermore, performing a Yoga stance for longer or shorter than the recommended period is damaging to one's health. However, we will identify not just a person's correct yoga position, but also how long that person must remain in that pose. Furthermore, several current systems employ various ways to recognize Yoga asanas, including the Convolutional Neural Networks (CNN) model. However, there is an overlap[21][17] issue that affects the system's accuracy. To address this problem, we present a Deep Learning model that combines modified Keras Applications (DenseNet201,Xception,MobileNet) with the usage of Ensemble modeling to determine difficult postures. For instance:

Figure 1.2: Upword Bow Pose

Figure 1.3: Upward Facing Two Foot Staff Pose

Both stances contain some comparable properties, as seen in figures 1.2 and 1.3, and there is a potential tendency of data overlapping while detecting the poses. We can circumvent some of the difficulties of the overlapping concerns stated previously by using the correct sensors in the right places.

1.3 Research Objectives

Research objectives are simply defined as a target or a goal of the project. There are few main objectives we have to maintain through the whole procedure of the project. The objectives are given below:

- Alerting the user how long he or she has to stay in the same position.
- Finding 6 Yoga classes with a total of 82 poses.
- Avoid overlapping concerns with comparable types of poses as much as possible.
- Bringing out better prediction output than traditional transfer learning by the usage of ensemble modeling.

Chapter 2

Previous Works

Following paper *"Yoganet: 3D Yoga Asana Recognition Using Joint Angular Displacement Maps with ConvNets(Convolutional Neural Networks)"*[12], advances with a goal to have an improvement in the accuracy in Yoga Asana Recognition and also optimize the training time. For this they come up with Joint Angular Displacement Map (JADM) using a single-stream deep CNN model. Their main focus and contribution was usage of JADMs, and it gives them optimize training times with increased accuracy compared to mainstream JDMs (Joint Distance Maps) and CNNs. Over the course of four weeks, they used a nine-camera mocap (Motion Capture) system to collect data on 42 different yoga poses performed by ten different people in ten different orientations. After calculating JADMs from 3D data and color-coding the JADMs, they next used this data to train and test the CNN. They've employed JADMs and RGB to color it (Red-Green-Blue). The data was then trained and tested using Python 3.6 and Keras and Tensorflow. They've used publically available datasets, such as CMU and HDM05, for 3D action datasets (Motion Database). Test subject 3's average recognition per sub-pose in the visual plane was used to verify CNN testing data's results. JDMs were tested on their dataset using a four-stream CNN architecture suggested by Li et al. JADMs had an average recognition rate of 90.01 percent, while JDMs had an average recognition rate of 53.68 percent. The rate climbed to 76.25 percent following the implementation of a four-stream CNN architecture. The matter of concern is, JDM net needed 267 epochs where JADM took only 98 epochs to reach their maximum performance. They have worked on ten angles of joints (+45, +60, +90, -45, -60, -90 horizontally and +20, +45, -20, -45 degrees) but there can be many angles to be considered to increase the accuracy.

The basic purpose of the paper *"Yoga Pose Classification Using Deep Learning"*[17], is to construct a deep learning model (Hybrid) for detecting yoga poses, with OpenPose being used for the first key-point extraction of human joint positions. The algorithm can efficiently classify yoga poses in both prerecorded films and in real time (Long Short Term Memory). Because RNNs (Recurrent Neural Networks) are unable to sustain long-term dependencies, they used LSTM (Long Short Term Memory) to reduce this problem. Prior to analysis, they processed the video data with OpenPose software, extracting and storing in JSON format the main points of posture from individual frames of video. From 60 to 20 to 20 frames, the video had a wide variation of frame counts. TensorFlow-Keras, OpenPose, NumPy, and Scikit Learn are among the Python packages used to create the models. The findings are

then analyzed using SVM, CNN, and lastly CNN+LSTM. This project's data set is part of the Open-Source collection and is freely accessible. It includes videos of 15 different people performing six distinct yoga poses (5 females and 10 males). There are 88 videos in all, each lasting 1 hour, 6 minutes, and 5 seconds. To summarize, they have shown that classifying all six yoga poses accurately using a hybrid CNN and LSTM model on OpenPose data is highly effective. SVM (Support Vector Machines) and CNN (Convolutional Neural Networks) also performed admirably. In this research, their model for CNN and SVM outperforms their expectations, with SVM being significantly more lighter and less sophisticated than a neural network, and it takes less time to train. The models' success is reliant on the accuracy of OpenPose posture estimate, which may not work well in situations when persons or body parts overlap. A portable gadget for self-training and real-time forecasting can also be included in this system. Although they just discussed six poses, there are many more to learn. Finally, their publication lists all of the approaches, making it harder to find the ones they used for this project.

"A hybrid posture detection framework: Integrating machine learning and deep neural networks"[21], this paper contains the uses of an advanced proposal of hybrid approach which involves both machine learning and deep learning at the same time. The main motive of the paper was to detect postures using deep learning and the machine learning both at the same time to make the system more efficient and accurate. According to the paper, using this hybrid approach the benefit was clearly visible through the accuracy and efficiency check. The paper assured that the hybrid approach can show really better performance than using machine learning(ML) or deep learning(DL) alone. The working procedure of the paper was quite organized and impressive. The paper mentioned that, DL method was used to reduce time in detecting the body postures and it was also being cleared that, because of the usage of DL methods, no feature engineering was required and in comparison to the ML methods, the performance was way better. To calculate the efficiency and the performance of the system, the ML methods were used. According to this paper, this hybrid approach is introduced to enhance the performance of a particular system. As it is mentioned before that, for accuracy checking ML methods were used and some of them are the prediction of ML classifiers such as Logistic Regression, Random Forest, K-Nearest Neighbors(KNN), Decision tree, Na'ive Bayes, Linear discriminant analysis, Support Vector Machine(SVM) and Quadratic discriminant analysis. The proposed DL methods or classifiers were used as the input of Convolutional Neural Networks(CNN) and Long-Short term Memory(LSTM) architecture. There was no specific and proper mention or reference of the used data set in the paper which was one of the drawbacks without solution or answer in the paper. Another major drawback of their proposed system was combining the features to predict the postures was time consuming. So, to handle this drawback, DL methods were being proposed in the paper and by using the proposed methods of hybrid approach, the paper showed the achievement of the 98% accuracy which is quite remarkable. To sum up, to get around the time constraint, implementing DL methods by contrast of traditional ML methods for feature extraction was a brilliant idea. Also, the usage of the hybrid approach helped the system to get a performance boost which was clearly visible through the accuracy of the whole process. It can not be helped but to mention that, the limited dataset for just two body

postures of sitting and standing was one major drawback of the proposed system in the paper. Therefore, the latent application of the proposed system could not be justified. The hybrid approach can be a revolutionary idea for implementing future developments containing higher efficiency with the help of proper data analysis.

"AI-Based Yoga Pose Estimation for Android Application"[16], the main motive of the mentioned paper was to lessen the struggle of the self learning process of the yoga poses. This paper mentioned that, there are a lot of various technologies out there which contains huge latent for predicting body postures. According to the paper, the best method can be distinguished by the usability of an android applications. Deep learning(DL) methods were used to support the proposed system in the paper to predict the yoga postures precisely. To do the mentioned process in a mobile application, PoseNet was implemented. Also, TensorFlow-Lite framework was used in order to implement that. According to the paper, the whole work procedure was divided in two major parts and one of those were pre-processing part and another one was real time application part of the system. The first or the preprocessing module or part had a motive of achieving the target values of the yoga poses which will be necessary to perform. The real-time or native component was created with the goal of dealing with apps that operate on an Android device without requiring any external intervention. The main goal of the pre-processing section was to determine the ideal or target values of each yoga position to serve as a measure to determine the accuracy of the actual pose, and this module helped with that, a single ideal image was collected to gain the key points and then locally store those key points in the machine. Later, OpenCv was used to predict the 17 essential key points by the usage of pre- trained model such as OpenPose and branch multi stage Convolutional Neural Networks(CNN). In the next phase, the key points were hard coded as the ideal or the target values. Inside the native or real time module, the android application provided built in Graphical User Interface (GUI) which takes the user video frame and then estimates the angle by the use of PoseNet. After that, it compare the calculation of the angle then displays the results. To quickly navigate though the app, navigation drawer was used for GUI. For both the target values and actual values, the angles were calculated by the use of the cosine law of triangles. The math module was used for the mathematical application in the android. The image view Layout was used for the purpose of displaying frames , the angles and correctness and skeletons. Any kind of reference or indication of data set or data set collection was mentioned in this paper. To be precise, the paper was well explained and detailed though there was no mention of the achieved accuracy of the whole process. In short, the mentioning of accuracy could be a justification of the proposed system to the readers or the reviewers to evaluate the value of the system properly. It is also necessary to mention that, this paper has its own salvation and downfall at the same time. For the positive side, the usage of DL methods for feature extraction to counter the time consumption problem in that portion and the usage of PoseNet to gain higher performance through the mobile application was an eye catching ideology to work on. They also mentioned about their future plan on implementing the progress tracking for the users to clarify their real time progress and keep on improving through time more efficiently. However, for the downside of the paper is the lack of information about the accuracy of the posture detection of the proposed system which actually hampered the massive potential of the whole

project.

In the paper called *"Real-time Multi-Person 2D Pose Estimation using Part Affinity Fields"*[6], they have come up with an effective and efficient method for multi-person pose detection. They used Part Affinity Fields (PAFs), which may be defined by a 2D vector field to encode the location, to accomplish the first bottom-up depiction of association scores.In addition, the first real-time system for multi-person 2D pose identification and limb orientation over the image domain. They have predicted the detection confidence maps with affinity field which encodes part-to-part association. After that, they have generated the ground truth confidence maps form annotated 2D key points to examine during the training. Then, they have used PAFs for part association and also performed Multi-Person Parsing with it. They have used two datasets to evaluate the method: (i) The MPII multi-person human dataset, (ii) the COCO 2016 key points challenge dataset. In the second dataset the gained 81.6% by using GT connection. Bottom-up gains 58.4% AP which can be increased overall 2.6% more with a single person CPM (Cost Per Mille). In this study, they have considered real time algorithm to detect the 2D pose in image for multiple people. They provide a non-parametric model of the key points association which usually encodes both the location and orientation of the human-limbs of the body and along with it, their learning parts detection and association system works simultaneously. Finally, they demonstrate that a Greedy Parsing Algorithm can produce high-quality body posture parses that stay efficient as the number of people in the image increases. They could use a 3D posture to implement the approach and verify that all failure instances are replicated completely.

The authors focused on Yoga Asana recognition in their study, *"Yoga Asana Identification: A Deep Learning Approach,"*[20] because Yoga has great medical value when done correctly. However, they used deep learning to perform the identification. They employed Transfer Learning, and the majority of the papers used CNN. They adopted the Transfer Learning approach in this paper because they didn't have a huge data collection. They employed a data set developed by Anastasia Marchenkova, in which 700 photographs of ten yoga asanas were dispersed among ten courses. When we compare this research to the others, We can see that they detected yoga asanas from still photographs using the models that are pre-defined but studying other papers we see that most of the researchers used real-time video input and then compared it with a trainers pose. However, depending on the approach utilized, the accuracy percentage ranged from 85 percent to 99.04 percent in those circumstances. The photos were reduced to 100*100 pixels, and 4608 features were retrieved, according to the report. The classifier model train set was utilized to partition the entire data set into train and test sets. The accuracy of the classifier was evaluated using the test set. At last, the authors achieved an accuracy of 85 percent, despite the fact that Hu moments (73.5%), Histogram of Oriented Gradients(HoG) (70.3%) and CNN (68.3%) were all significantly lower. However, the 85% accuracy is a bit is low considering their primary goal was health.

The authors of the paper *"Real-time Yoga recognition using deep learning."*[15] tried to detect the main 6 asanas correctly using Deep Learning(DL).And according to them, their study is the first ever study of detecting yoga pose from video input

using the end to end Deep Learning. Each frames key points was extracted from OpenPose by using CNN, then a Long short-term memory(LSTM) layer to provide predictions temporarily. The OpenPose tracks the positions of 18 critical spots on the body. They used the "Keras Sequential API" in the language python to program the deep learning model in this study. There is no publicly available data collection for Yoga identification, according to the author. They gathered data by having 15 people (10 men and 5 women) do each of the six asanas. 99.04 percent on each frames from videos, and 99.38 percent after pooling predictions on 45 frames from videos. It detects 98.92 percent accurately in real time. When using video input, they achieved more than 99 percent accuracy, and for real-time Yoga stance detection, they achieved 98.92 percent accuracy. This is the first study to use an end-to-end deep learning pipeline to detect Yoga from videos, according to the author. In this study, no specific data set is employed. They employed a webcam of logitech on a system with an GPU(NVIDIA X) and an processor(Intel Xeon) along with RAM of 32GB, which could be extremely pricey for certain people.

The authors intend to tackle the complex variability in yoga poses of a person using fine grained hierarchical classification of poses in their paper *"Yoga-82: A New Data Set for Fine-grained Classification of Human Poses"*[18]. Instead of detailed key points and skeletal annotations for human subjects, the author employed hierarchical labeling of human poses in this article. Using hierarchical labeling gives a number of advantages over traditional flat N-way categorization. Their experiments are split into two sections in this paper. To begin, they ran trials on the Yoga-82 data set. In the second section, three CNN architectures are described to utilize the class structure in the Yoga-82 data set for performance analysis utilizing hierarchical labels. This study makes use of the 'Yoga-82' large-scaled data set. There are over 28.4 thousands yoga position photos in this data set. These photos are divided into 82 classes, which are then subdivided into 20 sub-classes. These 20 sub-classes are then combined into six super-classes. The use of a hierarchical structure improves the system's performance. The accuracy of their third level classifier (82 classes) was increased from 74.91 percent to 79.35 percent. Because of additional learning supervision, the hierarchy information provided with the data set aids in improving performance. They want to incorporate explicit constraints between anticipated labels for different class levels in the future.

"Classification of yoga pose using machine learning techniques"[22], the paper mentioned here, proposed a system which will use machine learning modules and a large number of a huge dataset to perform training of different individual yoga poses of the users. It is mentioned in the paper that, four different machine learning model classifier were used to classify Sun Salutation yoga poses. The main algorithm that was used in the system was Pose estimation algorithm to execute the system. According to the paper, the data set that was used for the system contained a total of 12 different and individual yoga poses performed by the certified trainers. Every kind of necessary and mandatory preparation and precautions were being taken due to the collection of the large size of the dataset that was mentioned in the paper. The whole data set was divided into two parts such as train and test sets as it was a dataset of combined images of different yoga poses. According to the paper, the proposed system was able to pull out accuracy of almost 96% through the whole

process of the system which is an applaudable gain. There were few drawbacks in the proposed system one of them is the data collection procedure. Though, the data set was huge but webcam was used to collect image data which is not that much of an efficient tool to capture images. Because of that, the accuracy could have been hampered as the train and test set both contains low quality of data. The whole system process was tried to describe through the whole paper but there were few grammatical and structural error in the paper which can be a great problem for a reader or a reviewer to understand the proper proposal that was suggested in the paper. To sum up, the procedure could have been a little bit more clearer in the paper and if the data set collection was a little bit thought out then the proposed system could exceed the expectation and uphold the potential it was displaying through the paper.

Table 2.1: Paper review summary table:

Title of Paper	Year of Publication	Total Poses	Size of the Dataset	Accuracy Stats	Algorithm used	Strong Point	Weak Point
AI-Based Yoga Pose Estimation for Android Application	2020	Not Available	Not Available	Not Available	OpenPose architecture, with two-branch multistage Convolutional Neural Network.	1. Posenet to to implement on mobile 2. Deeplearning to reduce time complexity of feature extraction.	1. No progress tracking 2. No accuracy information
A hybrid posture detection framework: Integrating machine learning and deep neuralnetworks	2017	6 postures	Total 88 videos	Not Available	On open pose they used hybrid Convolutional Neural Network and Long Short-Term Memory model	Works exceptionally for CNN and SVM	1. Only 6 postures in total 2. Overlap issue
Classification of yoga pose using machine learning techniques	2020	12	Sun salutation yoga poses but size is unavailable.	96%	They used Pose estimation algorithm	Accuracy was high(96%)	Dataset was biased
Yoga Asana Identification: A Deep Learning Approach	2021	10	around 700 pictures.	85%	They used Transfer Learning	85% accuracy compared to CNN(68.3%), HoG(70.3%), and Hu Moment(73.5%).	1. Dataset was not large 2. 10 postures only
YogaNet: 3D Yoga Asana Recognition Using JointAngular Displacement Maps with ConvNets.	2019	42	HDM05 motion capture dataset	98.29%	JADM(Joint Angular Displacement Maps)	JADM which was better than using JDM.	Front view data rotation was in 10 different angles.
Real-time Yoga recognition using deep learning	2019	6	Not Available	99%	Openpose followed by CNN and LSTM model.	1. End to end deep learning pipeline to detect postures from video 2. High accuracy on real time posture detection	1. Expensive 2. Dataset is not mentioned.
Yoga-82: A New Dataset for Fine-grained Classification of Human Poses	2020	82	28400	Around 79.5%	The algorithm "DenseNet" is used.	A hierarchical classification was provided with the data set.	They could add superclasses in order to organize 82 different poses.
Yoga Pose Classification Using Deep Learning.	2020	6	88 videos	Training set accuracy: 99.92% Validation accuracy: 99.87% Test accuracy: 99.38%	Openpose	SVM, which is lighter and faster than CNN	1. Only 6 postures in total 2. Overlap issue
Realtime Multi-Person 2D Pose Estimation using Part Affinity Fields	2017	1160	1. Common Object in Context dataset 2. MPII human pose dataset	mean Average Precision = 81.6%	They used PAF (Part Affinity Fields).	Their algorithm gives high quality parses even though the number of people increases.	For some failure cases, 3D poses could be implemented.

Chapter 3

Data Collection

The data set we used is "Yoga-82" to collect our images of various yoga pose. In this data set we have total of 82 different classes each defining different individual pose. In these 82 files we have approximately 23945 different images of different poses. Images in these files are taken from different angles and these images contain different backgrounds such as both indoors and outdoors. Some of the sample poses of different sub and main classes are given below:

Table 3.1: Sample of poses we are working on:

Sample Pose	Pose Name	Class
	• Chair Pose or Utkatasana	• Class: Standing • Subclass: Straight
	• Intense Side Stretch Pose or Parsvottanasana	• Class: Standing • Subclass: Forward Bend

Table 3.2: Sample of poses we are working on:

Sample Pose	Pose Name	Class
	• Half Moon Pose or Ardha Chandrasana	• Class: Standing • Subclass: Side Bend
	• Warrior II Pose or Virabhadrasana II	• Class: Standing • Subclass: Others
	• Staff Pose or Dandasana	• Class: Sitting • Subclass: Normal 1(legs in front)
	• Half Lord of the Fishes Pose or Ardha Matsyendrasana	• Class: Sitting • Subclass: Normal 2(legs behind)
	• Wide Angle Seated Forward Bend pose or Upavistha Konasana	• Class: Sitting • Subclass: Split

Table 3.3: Sample of poses we are working on:

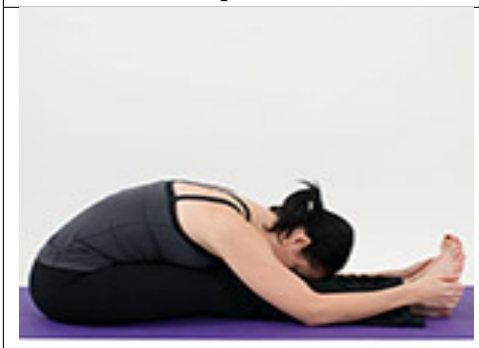
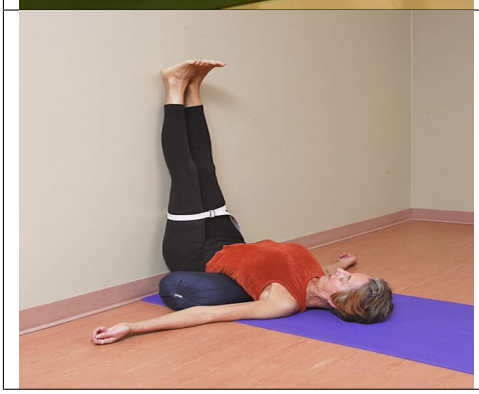
Sample Pose	Pose Name	Class
	• Seated Forward Bend pose or Paschimottanasana	• Class: Sitting • Subclass: Forward Bend
	• Heron Pose or Krounchasana	• Class: Sitting • Subclass: Twist
	• Scale Pose or Tolasana	• Class: Balancing • Subclass: Front
	• Eight Angle Pose or Astavakrasana	• Class: Balancing • Subclass: Side
	• Legs Up the Wall Pose or Viparita Karani	• Class: Inverted • Subclass: Legs Straight up

Table 3.4: Sample of poses we are working on:

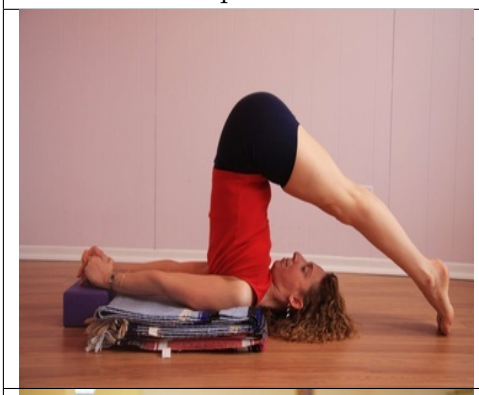
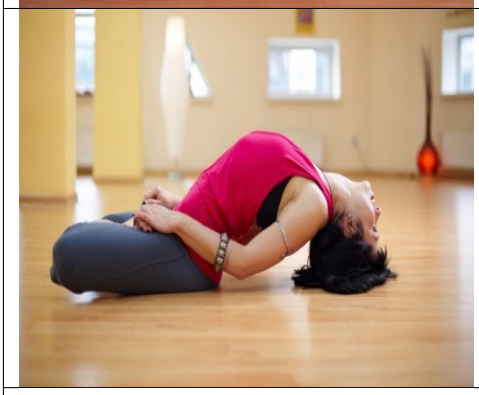
Sample Pose	Pose Name	Class
	• Plow Pose or Halasana	• Class: Inverted • Subclass: Legs Bend
	• Fish Pose or Matsyasana	• Class: Reclining • Subclass: Up Facing
	• Cobra Pose or Bhujangasana	• Class: Reclining • Subclass: Down Facing
	• Side Plank Pose or Vasisthasana	• Class: Reclining • Subclass: Side Facing
	• Plank Pose or Kumbhakasana	• Class: Reclining • Subclass: Plank Balance

Table 3.5: Sample of poses we are working on:

Sample Pose	Pose Name	Class
	• Upward Bow (Wheel) Pose- orUrdhvaDhanurasana	• Class: wheel • Subclass: Up Facing
	• Cat Cow Pose or Marjaryasana	• Class: wheel • Subclass: Down Facing
	• Boat Pose or Paripurna Navasana	• Class: wheel • Subclass: Others

Chapter 4

Data Pre-processing

As the data set was collected from different links for different poses, there were few invalid image files in those links. Therefore, to handle these null or invalid images we performed pre-processing on the data collection. To handle these null values we used glob(global), PIL(Python image Library) etc. to avoid the invalid images to create a fresh Collection of data set.

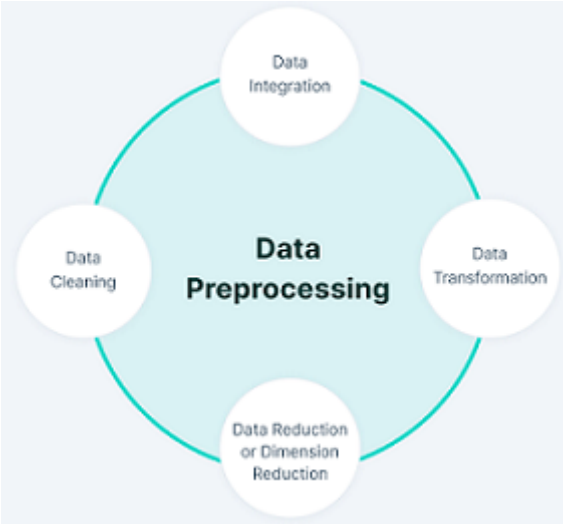


Figure 4.1: Pre-proccess

Moreover, we know that, pre-processing is a mandatory part for data analysis. To preprocess the data we used several keras library functions for different models which processes the data or the input images and make it ready according to the pre-trained model that is being used in the AI system. We have scaled them in (224, 224) format. We used this method for other keras pre-trained model for pre-processing the data or the images by simply changing the naming of the method according to the pre-trained model that is being used in the AI system.

If we see the size of our dataset then we can clearly clarify that our dataset is large and for gaining better performance large dataset can be a valid resource. The enormous volume of data needed to train deep learning methods like neural networks sets them apart from other machine learning techniques. To some extent, training on huge datasets alleviates the problem of excessive variation in neural networks. Training data overfitting is far less likely when a model is subjected to a larger number of training examples. Deep learning models are designed to find the most efficient route from input parameters to output ones. To do this, it needs to pick out the right features and look for patterns that aren't obvious in the data.

There can be three considerable advantages for using large datasets such as feature selection, reduction in noise and accuracy improvement. Selection of features is improved because of a wider range of features and more opportunities to recognize patterns. To tackle the problem of noise in data, reducing noise helps. More data means it's easier to uncover hidden patterns and cut through the clutter. Moreover, the presence of large training datasets can be really helpful for improving the overall performance of the proposed model or models.

As we used a large size of image dataset of different yoga poses, there were some null and garbage data present in the dataset. To eliminate these unusable data we excluded few types of images and images which are not compatible for loading. After these null data elimination process we gained total of 18488 images as our dataset. We also did proper labelling for different poses and categories to help our model to train more easily and also, to improve the prediction accuracy of our model.

Chapter 5

Models & Classifiers

We have researched popular models of Keras Application for image classification and based on our dataset and future goal, we made a short list of the models we are planning to modify and use in future.

5.0.1 DenseNet201

In 2016, Facebook AI research introduced the DenseNet[10] family in the paper called "Densely Connected Convolutional Networks"[9]. It is a family because there are numerous versions with varying depths, differing from 121 layers and 0.8 million parameters to 264 layers and 20.2 million parameters.

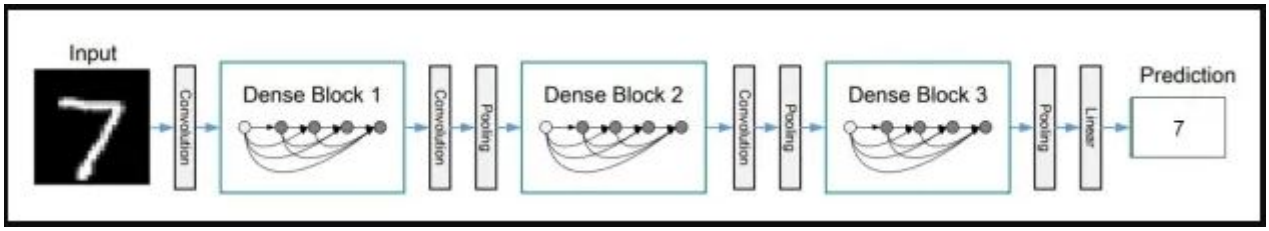


Figure 5.1: DenseNet201

Like all convolution methods DenseNet[19] also uses polling and ReLu activation function to work. The dense blocks in this architecture play the vital role.
The previous image shows a five-layer dense block. In these dense blocks, each layer receives as input all prior feature maps, which facilitates the training process by resolving the vanishing gradient issue. This vanishing gradient problem occurs in networks that are sufficiently deep that when the error is back-propagated into the network, it decreases at each step and finally becomes zero. Without being reduced too much these connections allow the error to be propagated further. In the same way, these links enable feature reuse and lower the amount of parameters because it reuses existing feature-maps information rather than creating new parameters. And therefore accessing the networks "collective knowledge" and reducing the chance of over fitting, due to this reductional total parameters. DenseNet can use the same number of layers as another state-of-the-art architecture ResNet but it will reduce the number of the parameters by around 5 times compared to ResNet. Dense201[19] has 201 layers and comparatively light in size(80 MB) and takes and each inference step takes around 127.24 ms(millisecond) for CPU and 6.67 ms for GPU.

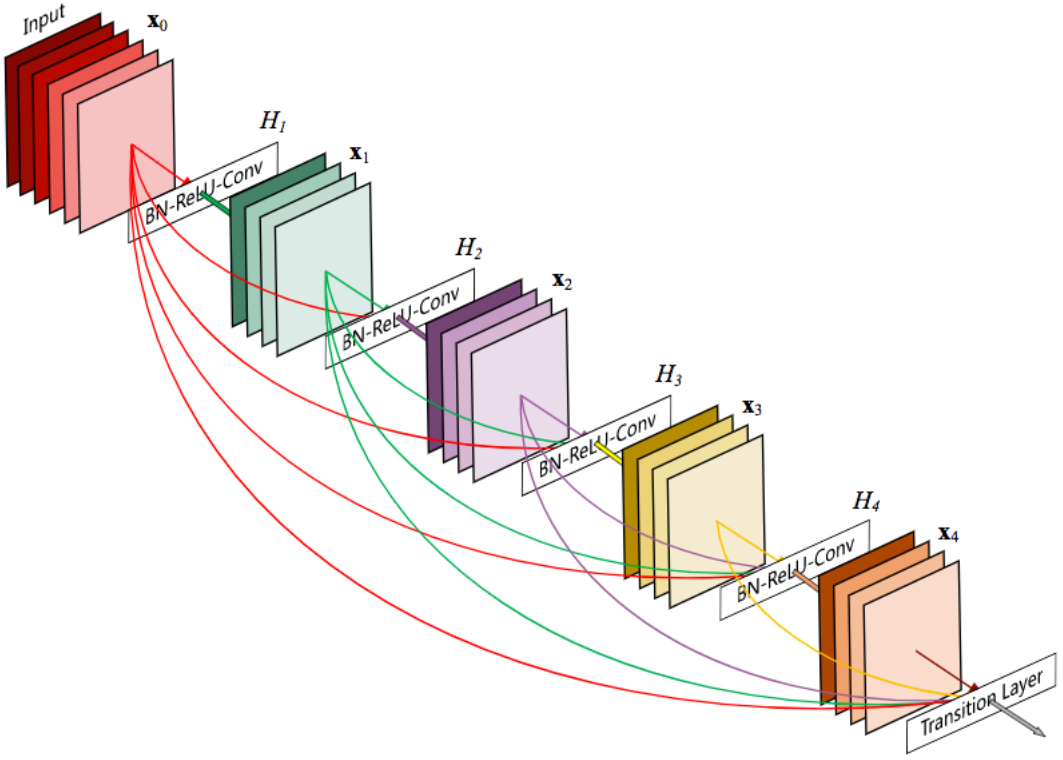


Figure 5.2: 5 layer Dense Block

5.0.2 Xception

Xception[7] is a novel deep convolutional neural network architecture (Arguments: weights, include_top, input_tensor, pooling, classifier_activation = "softmax", input_shape,classes=1000). It is highly inspired by Inception V3 and these two architectures takes almost same number of parameters (Xception = 22,855,952; Inception V3 = 23,626,728). Xception proposes more efficiency which performs just better than Inception V3 on the ImageNet data set and also performs much better on huge image classification data set (350 million images with 17,000 classes). It is a CNN architecture based on depth wise separable convolutional layers which contains 36 Convolutional layers.

Table 5.1: Comparison of Xception:

Models	Top-1 accuracy	Top-5 accuracy
VGG-16	0.715	0.901
ResNet-152	0.770	0.933
Inception V3	0.782	0.941
Xception	0.790	0.945

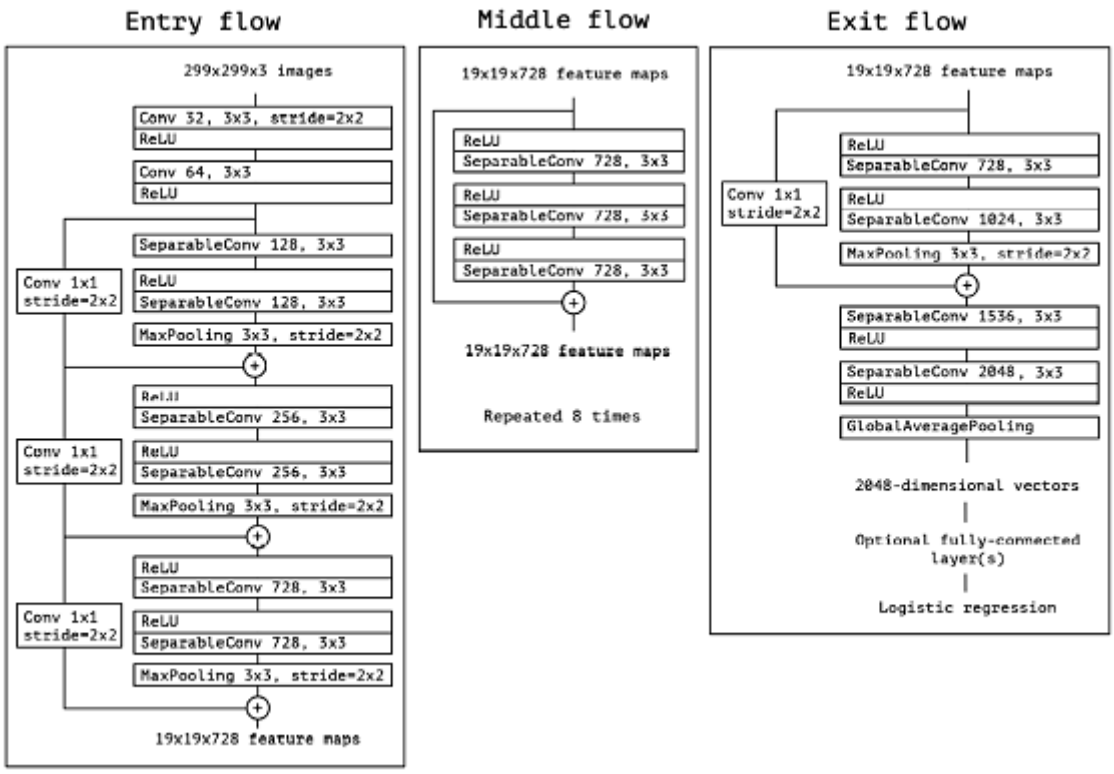


Figure 5.3: Xception Work Flow

5.0.3 MobileNets

MobileNets[8] is another architecture which is to create light weight deep neural networks by using depthwise separable convolutions. MobileNets[14] mainly focus on optimization and speed. This model applies to each input channel only a single filter. Depthwise convolutions and pointwise convolutions are the two layers of depthwise separable convolution. MobleNets use batchnorm and ReLU non linearities for both layers and also 3 x 3 depthwise separable convolutions which need between 8 to 9 times less computational complexity than standard convolutions but the accuracy reduction is very small.

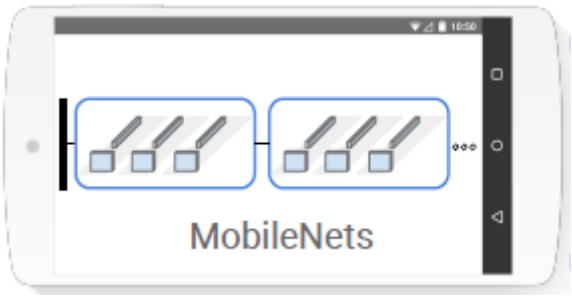


Figure 5.4: MobileNet

The MobileNet structure uses full convolution for the first layer and for the other layers it's built on depthwise separable convolutions but down sampling is handled with strided for all the layers. MobileNet has 28 layers which spends 95% of it's computational time in 1 x 1 convolutions which also has 75% of the parameters.

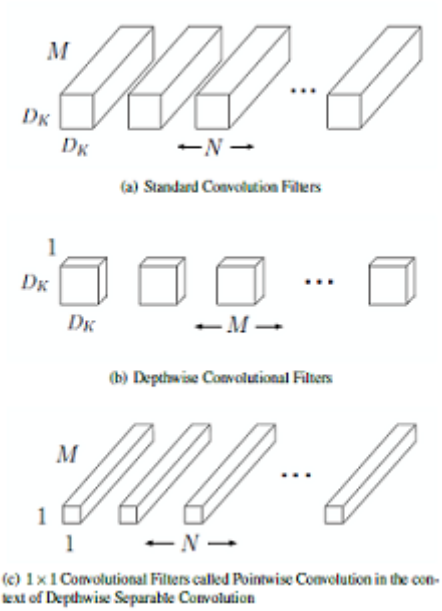


Figure 5.5: The standard convolutional filters in (a) are replaced by two layers: depthwise convolution in (b) and pointwise convolution in (c) to build a depthwise separable filter.

By removing 5 layers of separable filters with feature size 14x14x512, the structure can be shallower but making MobileNets thinner is 3% is better than making them shallower.
We have used Convolutional Neural Networks of depth 2, with pooling and ReLu activation function. In first depth we have 32 filters and in second depth we used 64 filters.

5.0.4 ResNet50 & ResNet101

ResNet50(Residual Neural Network)[13] is one kind of ResNet model. It has 1 Average Pool layer and 1 MaxPool layer of 48 Convolution layers. In addition, it also contains 3.8×10^9 Floating points operations. After winning the first prize by AlexNet at the LSVRC2012 classification contest in 2012, RestNet was the most fascinating thing that occurred for computer vision along with deep learning. By using the ResNet presented framework it is possible to train ultra deep neural networks with great performance.

layer name	output size	18-layer	34-layer	50-layer	101-layer	152-layer
conv1	112×112	7×7, 64, stride 2				
conv2_x	56×56	3×3 max pool, stride 2				
		$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$
conv3_x	28×28	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times8$
conv4_x	14×14	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix}\times6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times23$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times36$
conv5_x	7×7	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$
	1×1	average pool, 1000-d fc, softmax				
FLOPs		1.8×10^9	3.6×10^9	3.8×10^9	7.6×10^9	11.3×10^9

Figure 5.6: Different Layers of ResNet50

There is a tiny change in ResNet50 architecture as it can skip three layers and also added 1 × 1 convolution layers.

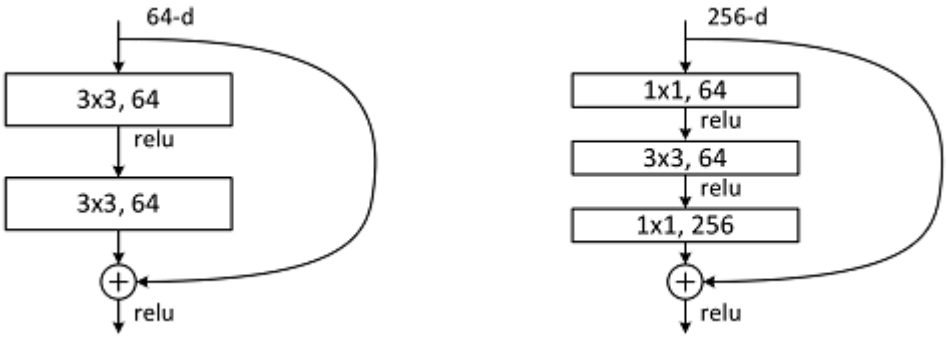


Figure 5.7: ResNet architecture

As we can see, ResNet50 has a convolution whose kernel's size is 7 × 7 and it has 64 different kernels with a stride of size 2 giving 1 layer each. Then there is a max pooling with a stride size of 2. After that, in the next convolution it has a 1 × 1, 64 kernel following a 3 × 3, 64 kernel and lastly a 1 × 1, 256 kernel. These 3 layers are repeated 3 times and give 9 layers. Next, we have a kernel of 1 × 1, 128 then 3 × 3, 128 and lastly 1 × 1, 512 and these steps are repeated 4 times and give 12 layers. After that layer, we have a kernel of 1 × 1, 256 and two more kernels with 3 × 3, 256 and 1 × 1, 1024 and repeated 6 times to give 18 layers. In addition, a kernel that is 1 by 1,512 followed by two more that are 3 by 3,512 and one that is 1 by 1, 2048 and then repeated three times results in nine layers. In the end, we finish it off with a completely connected layer that has a thousand nodes total. This layer has an average pool. In addition to this, a softmax function will provide one layer. By skipping count of activation functions and the max/avg pooling layers we have 1+9+12+18+9+1 = 50 layers Deep convolutional network. On the ImageNet validation set ResNet 50 model achieved a top-1 error rate of 20.47% and a top-5 error rate of 5.25%. This is for a single model of 50 layers. The comparison is given below :

method	top-1 err.	top-5 err.
VGG [41] (ILSVRC'14)	-	8.43†
GoogLeNet [44] (ILSVRC'14)	-	7.89
VGG [41] (v5)	24.4	7.1
PReLU-net [13]	21.59	5.71
BN-inception [16]	21.99	5.81
ResNet-34 B	21.84	5.71
ResNet-34 C	21.53	5.60
ResNet-50	20.74	5.25
ResNet-101	19.87	4.60
ResNet-152	**19.38**	**4.49**

Figure 5.8: error rate of different keras models

ResNet-101[11] is a CNN model which has 101 layers. An ImageNet-trained version of the network can be loaded as a pre-trained version. There are over 1000 object types that can be categorized by the pre-trained network. These categories can be of mouse, pencil, different animals or keyboards. Thus, the network gets the ability to represent images with a wide range of features. The network accepts images with a maximum resolution of 224 by 224 pixels.

5.0.5 Ensemble modeling

An ensemble model is a machine learning model that integrates multiple base models to provide generalized predictions by combining their predictive power[2]. Variables such as training data and algorithm/model architecture are unique to each base model. Due to its specialized tuning, each of these may catch only a few elements or learn from only a portion of the training data. Ensemble modeling gives us the ability to merge all of these models into a single superior model that is based on learning from most or all of the training datasets. Predictions that are averaged out eliminate the problem of local minima in individual models. Ensemble modeling has an unusual property in that there is no limit to the number of base models that can be used. Base models or "ensemble members" can range in number from two to ten. Ensemble modeling is usually done to improve the computational power of

models used for classification or prediction problems. This can be seen in a number of Kaggle contests where the winner simply combined several base models to make a single generalizable model with better accuracy. Ensemble modeling can also be used for a number of other purposes. With the help of its "ensemble members," an ensemble model is able to generate predictions with greater generalizability and even greater accuracy in the vast majority of cases. The primary goal of any machine learning challenge is to develop a model that has a low generalization error and a high degree of accuracy. An ensemble model can construct a model that learns

better from the training data by combining optimal aggregation combinations of base models or 'weak learners'.

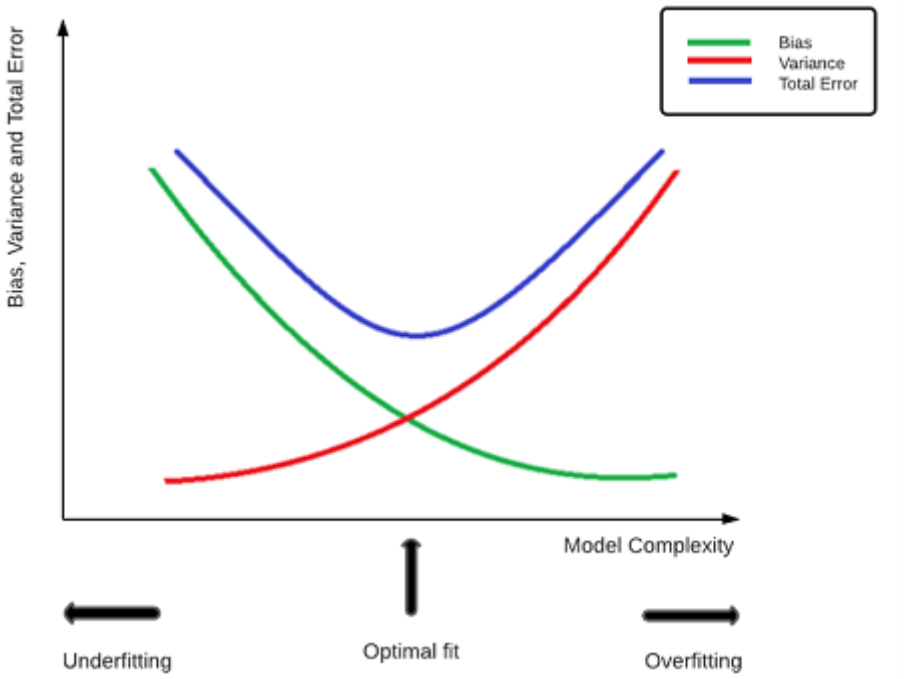


Figure 5.9: bias-variance tradeoff

Here is a visual illustration of the bias-variance tradeoff, which is a critical parameter for any supervised machine learning model. It's possible to understand a model's bias-variance tradeoff by looking at them as complementary aspects. A model's ability to adapt to new data is measured by its variation, while bias is the result of inaccurate assumptions about data. When one goes up, the other goes down, and vice versa. Models that have good bias-variance tradeoff values are more accurate when dealing with unobserved data.

Ensemble modeling is a better a choice for reduction in variance, bias and noise and also for better overall performance. For better learning from large datasets and improved feature selection Ensemble modeling is a better choice. Though it increases complexity but for greater performance sometime ensemble modeling is a better choice.

5.0.6 Random Forest Classifier

Random Forest is one of the most commonly utilized advanced techniques in Supervised learning algorithms for both regression and classification issues using non-linear or real-time data[4]. Random Forest, as the name suggests, is a forest made up of a variety of trees. There are several decision trees in a random forest to help enhance the accuracy of the dataset's predictions. To anticipate the final output, the random forest considers predictions from all trees and votes on which predictions are most likely to be correct. It is likely that certain trees in the random forest will correctly predict the dataset's class, while others may not, because it uses many trees to aggregate the classification of the dataset. However, when all the trees work together,

they forecast the proper result. A better Random forest classifier has the following two assumptions, such as In order for the classifier to make accurate predictions, the feature variable in the dataset should contain actual values. Also, Each tree's predictions must have a very low correlation coefficient.

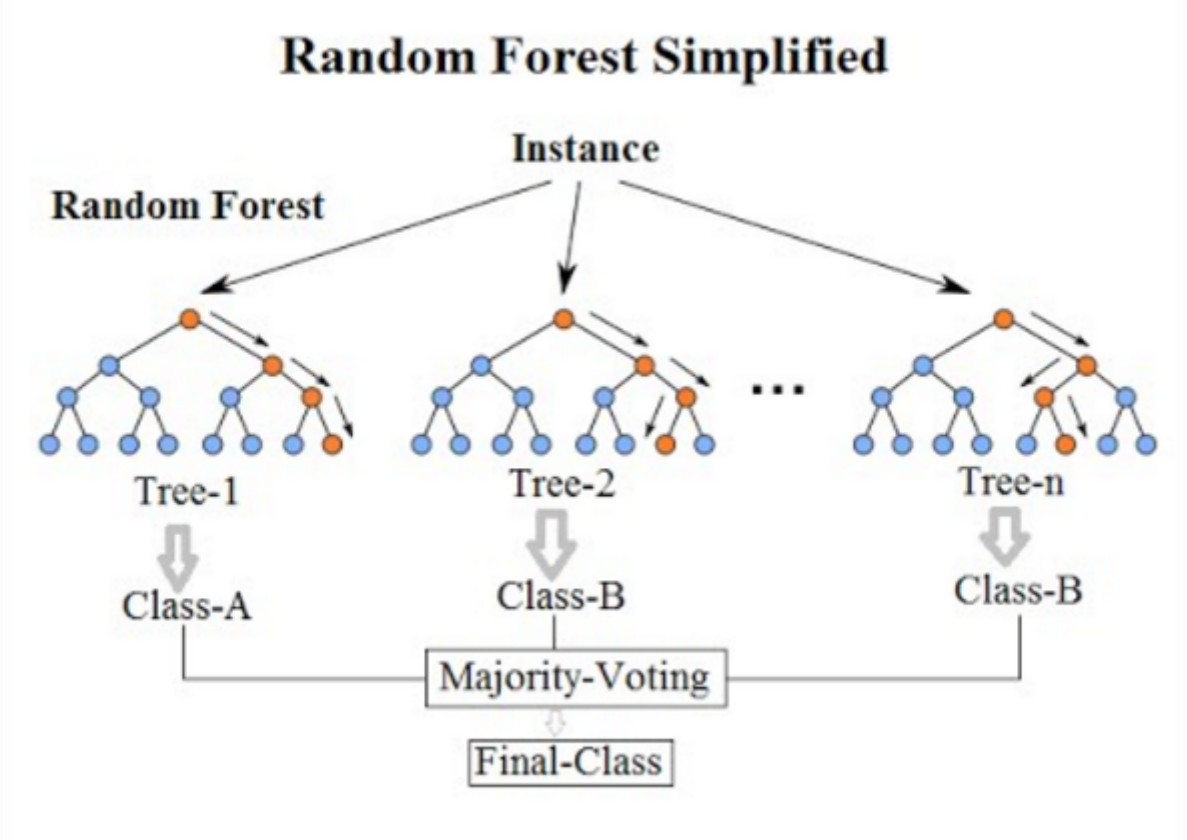


Figure 5.10: Random Forest diagram

To better understand how things work, consider the following processes and diagram:

- Step-1: Select random K data points from the training set.
- Step-2: Build the decision trees based on the specified data points in this step(Subsets).
- Step-3: Choose the number N for decision trees that you wish to build.
- Step-4: Continue with Steps 1 and 2 .
- Step-5: New data points should be assigned to the category that has the most votes in each decision tree.

Chapter 6

Result Analysis

6.0.1 Result

To testify the result analysis after the train and test for all the models for the 82 poses and 6 basic classes we attained different accuracy measurement from different models. For each model we attained Top-5 accuracy and Top-10 accuracy for predicting 82 classes along with the validation accuracy. Also, we attained Top-2 accuracy and Top-3 accuracy for using 6 basic classes along with the validation accuracy which are given below in the following table:

Accuracy table of Yoga-82(82 classes)			
Models	Test Accuracy(%)	Top-5 accuracy(%)	Top-10 accuracy(%)
MobileNet	75.83	91.16	95.14
MobileNetV2	70.19	88.93	93.64
ResNet50	74.44	92.80	96.26
ResNet101	**76.46**	92.03	95.69
Xception	74.52	93.24	96.43

Figure 6.1: Accuracy for 82 classes

Accuracy table of Yoga-82(6 basic classes)			
Models	Test Accuracy(%)	Top-2 accuracy(%)	Top-3 accuracy(%)
MobileNet	82.86	93.33	95.49
MobileNetV2	86.46	95.15	95.99
ResNet50	86.25	95.34	97.75
ResNet101	86.06	95.17	98.09
Xception	**88.52**	96.21	98.29

Figure 6.2: Accuracy for 6 basic classes

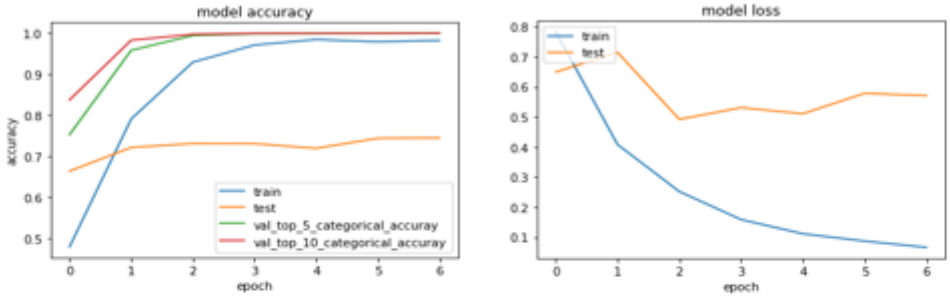


Figure 1: ResNet50 acc & loss (for 82 classes)

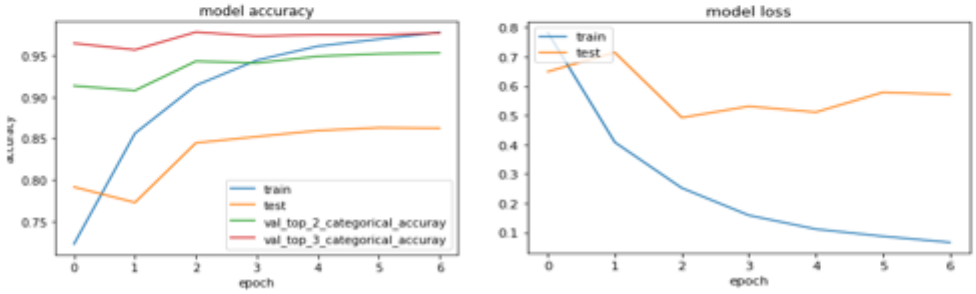


Figure 2: ResNet50 acc & loss (for 6 classes)

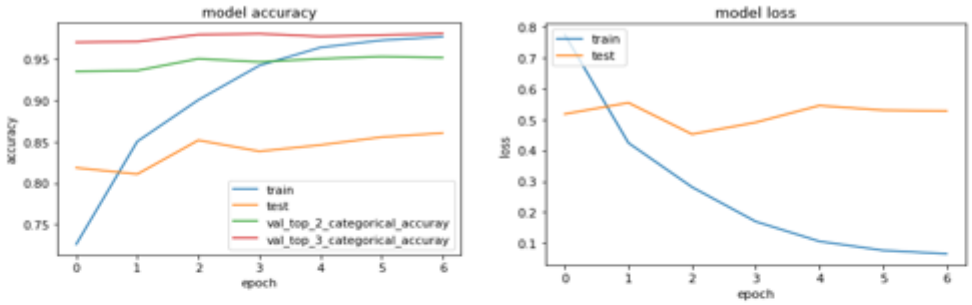


Figure 3: ResNet101 acc & loss (for 6 classes)

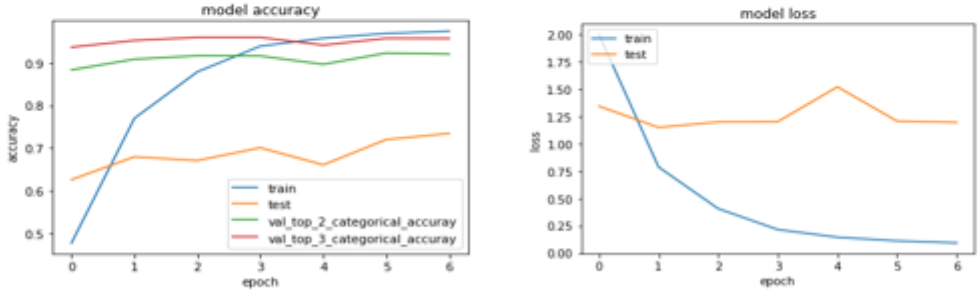


Figure 4: ResNet101 acc & loss (for 82 classes)

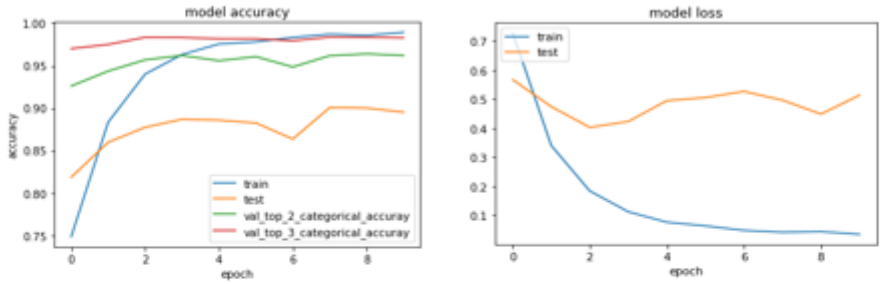


Figure 5: Xception acc & loss (for 6 classes)

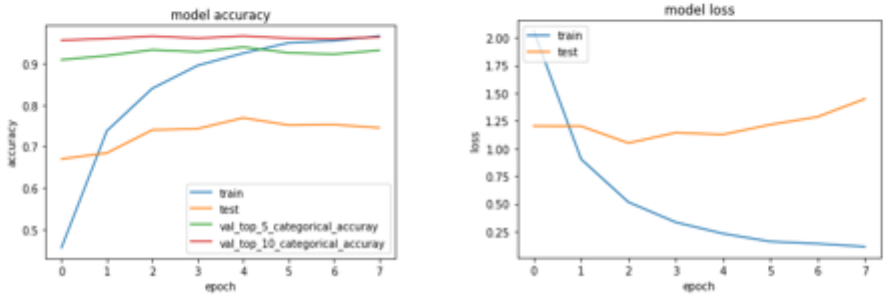


Figure 6: Xception acc & loss (for 82classes)

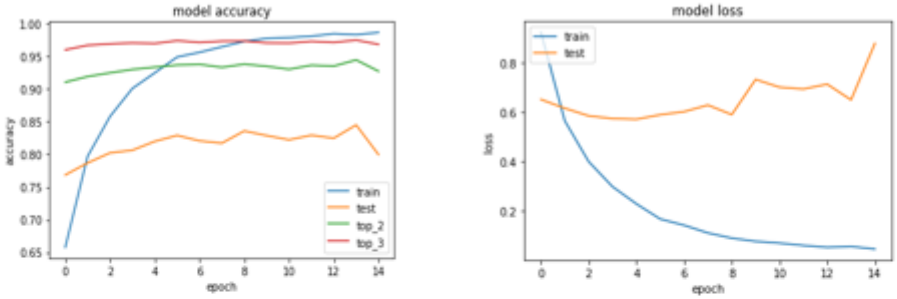


Figure 7: MobileNetV2 acc & loss (for 6 classes)

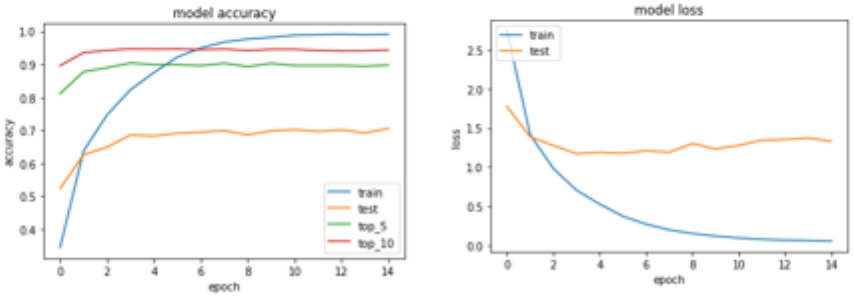


Figure 8: MobileNetV2 acc & loss (for 82 classes)

	precision	recall	f1-score	support
0	1.00	0.80	0.89	15
1	1.00	0.73	0.85	15
10	0.89	0.97	0.93	154
11	0.79	0.76	0.78	25
12	0.89	0.97	0.93	61
13	0.92	0.93	0.92	58
.	.	.	.	.
.	.	.	.	.
.	.	.	.	.
.	.	.	.	.
.	.	.	.	.
46	0.86	0.92	0.89	26
47	0.85	0.88	0.87	52
48	0.96	0.86	0.91	28
49	0.93	0.86	0.89	44
5	0.96	0.98	0.97	54
50	1.00	0.91	0.95	23
51	0.87	0.95	0.91	43
.	.	.	.	.
.	.	.	.	.
.	.	.	.	.
.	.	.	.	.
.	.	.	.	.
6	0.98	0.97	0.97	59
60	0.95	1.00	0.97	74
61	0.88	0.92	0.90	24
62	0.91	0.89	0.90	35
63	0.95	0.88	0.91	60
64	0.96	0.94	0.95	51
65	0.95	0.89	0.92	62
66	0.93	0.92	0.92	59
67	0.93	0.90	0.91	29
68	0.94	0.90	0.92	51
69	0.98	0.93	0.96	60
7	0.95	0.97	0.96	79
70	0.93	1.00	0.96	13
71	0.93	0.93	0.93	42
72	0.86	0.95	0.90	64
accuracy			0.93	3666
macro avg	0.93	0.91	0.92	3666
weighted avg	0.93	0.93	0.93	3666

Figure 6.3: Classification report

6.0.2 Analysis

In figure 6.1 and figure 6.2 test accuracy represents how much accurately the models can classify the yoga categories(6 super-class) and the yoga poses(82 classes) respectively. Top-N categorical accuracy means even if one of the N predictions matches the intended label, the classification is deemed accurate. We have predicted Top-5 and Top-10 categorical accuracy for the 82 classes and Top-2 and Top-3 categorical accuracy for the 6 super-classes.

In a nutshell, we know that if a model is over-fitting or not that can be known through loss function graph. For models, over-fitting[3] occurs when they become overly dependent on their training data and lose their ability to perform well on fresh data. To put it another way, the model takes in and incorporates the random fluctuations in the training data as new concepts. From the above loss function graphs of traditional transfer learning models, we can see that, we might have gone through slight over-fitting issues. The result of these over-fitting issues had impact on the accuracy of prediction. To solve this issue we used ensemble modeling to improve or boost the prediction accuracy of the yoga pose classification. To get the better result we chose the top result from figure 6.1 which is ResNet101 with 76.46% accuracy for classification of 82 classes and top result from figure 6.2 which is Xception with 88.52% accuracy for classification of 6 classes as two inputs for the ensemble modeling which is our third model. From the figure(6.3), we can see the classification report of the ensemble model where we have showed some of the test batches. From the precise column, we can see that how well each classes are predicted divided by all the times the model has predicted the class(rightly or wrongly). However, in recall column, the scores represent the ratio of number of correctly predicted members in a class divided by the number of total members. Then, the f1 score indicates, how good are precise and recall values. Finally, the support column represents the occurrences of individual classes in the data set.

Chapter 7

Discussion

At first, we worked on 82 different yoga poses then we divided these 82 poses into 6 different categories. We worked on different keras models such as MobileNetV1, MobileNetV2, ResNet50, ResNet101 & Xception to train and predict different yoga poses. We used all those models to classify 82 poses and also classify the 6 main categories such as Balancing, Inverted, Reclining, Sitting, Standing, Wheel. Designated pre-processing was used for each pre-trained keras models. Proper labelling and null data removal was also an important part of this image pre-processing . After pre-processing the data or the images we gained the total image of 18488 and splitted the data-set into 7:2:1 ratio of train, valid and test respectively . Afterward, we tried to perform the prediction and measure the accuracy of the pre-trained model for the given data-set. We tried to get Top-2, Top-3 accuracy reading for classifying categorical poses and Top-5 and Top-10 accuracy reading for classifying 82 yoga poses.

Our main goal of this paper was to build a model or a system which will predict yoga poses as accurately as possible and predict the class and the basic super class of a given pose. So, to satisfy our goal to improve the performance of our system to predict more accurately we introduced third model which mainly uses ensemble[1] modeling. We used the output of previous two models(first model for classifying 82 poses and the second one for 6 categories) as inputs for the third model to predict the pose more precisely and efficiently with its proper details of classification.

We used feature extraction from numpy array. All the predictions of the 82 poses and the 6 categories are used here as features. After we used one hot encoding on these features and we applied these modified features into random forest[5] classifier to achieve satisfactory prediction accuracy. Moreover, we used decision tree classifier which also brought satisfactory outcome. To achieve satisfactory result and accurate classification of a given pose we used two different preferred Keras models' prediction output each time as input for our ensemble modeling. For attaining better performance from ensemble modeling, we used best output from each transfer learning models like ResNet101's output for 82 classes and Xception for 6 basic classes as input for the third model to boost the performance.

While Jupyter Notebook's memory size can be increased, there are also additional memory-related issues that need to be taken into consideration. More powerful CPUs and RAM can be likely solutions, but they can be costly. Splitting the dataset into smaller subsets, which can then be loaded and used by the individual base models, is a cost-effective option. When working with large datasets, novices often make the mistake of only using a portion of the data. Because some of the data may be omitted, the model will not be as accurate as it could be. If we were to look at a dataset of brain scans from patients, we could create a classifier to detect mental disorders. The classifier uses the patterns in each image to determine whether or not the disorder is present. Because essential characteristics (patterns) may be lost when a section of the dataset is discarded, a decrease in accuracy may occur. As a result, ensemble modeling is the most cost-effective and efficient way for training neural networks with a huge dataset.

An improvement in the bias-variance trade off of the overall model is achieved by lowering the high variance value of the individual members using ensemble modeling of neural networks. Besides improving dependability and decreasing generalization error, it is possible that this method will also improve prediction accuracy.

Although, we got satisfactory result for applying the third model but because of the complexity of ensemble modeling and features we had to face few obstacles on the way. We had to face issues with different formats of images such as only JPG and PNG formats were usable for the model and other formats were not suitable for the models. That is why we had to eliminate unsuitable formats of images from the dataset to help the dataset for processing efficiently. Also in our dataset in a same class few images contained different angles of same pose and contained complex background which gave the model hard time to train on the images and imposed as a challenge for us. For executing the models, we used local device which uses AMD Ryzen™ 5 3600 as CPU and Nvidia GEFORCE GTX 1660 SUPER as GPU. Although, we tried to use the GPU for computation but the time duration for executing the models were significant. Even we faced all these challenges and issues, we were able to pull through all the way in the end.

Chapter 8

Conclusion

In this digital world, body posture detection is one of the most common topic in the computer vision field. Image classification under computer vision field is a worthy research topic to mention. There are a lot of AI systems that are performing outstanding tasks by using image classification. In this paper, we tried to build a model or a system using deep learning to classify different yoga poses through image data. The word 'yoga' came from Sanskrit, which means 'union'. It is called unions because it connects the soul with our body. The benefits of yoga are beyond description. Yoga mainly increases practitioners strength, flexibility, stability and balance. Beside improving these, the impact of yoga is far stretched. Yoga also helps with blood pressure reduction, controlling cholesterol level, cardiovascular health, maintenance of wight loss etc. As we know that yoga is necessary exercise for our daily life to maintain our synchronization between our body and mind , it is also a mandatory fact to perform poses accurately. To do that, we need to detect and learn about specific pose or the classes of yoga. Therefore, using deep learning method we introduced a system which will train and test on a given dataset containing a significant number of images of 82 different yoga poses which can be divided into 6 basic super classes and then it will predict the class and pose of any given yoga pose as accurately as possible.

We tried different keras pre-trained model such as MobileNet, ResNet, Xception to attain best performance of our system for predicting the classification of different yoga poses. We also introduced ensemble modeling to boost the performance of our proposed model to classify those images of yoga poses with as much accuracy as possible. We were successful enough to boost our result to 92.77% using ensemble modeling for using the prediction output of two Xception models for 82 classes and 6 basic super classes respectively. In brief, we used one of the top output or result of our pre-trained models and tried to boost the result using our proposed ensemble modeling for predicting yoga poses with classification.

Chapter 9

Future Plan

In future, we want to detect the Joint Angular Displacement Map(JADM) using OpenPose. We also intent to build an mobile application which will be able to detect the accuracy of the pose given image input. Also, the proposed application will be able to classify for any given yoga poses with better prediction accuracy. In our current system we have faced over-fitting and overlapping issues. We will handle the overlapping issues and also, using data augmentation in the future we hope that we will be able to minimize the over-fitting issues.

Bibliography

[1] R. Zwanzig, "Ensemble method in the theory of irreversibility," *The Journal of Chemical Physics*, vol. 33, no. 5, pp. 1338–1341, 1960.

[2] T. G. Dietterich, "Ensemble methods in machine learning," in *International workshop on multiple classifier systems*, Springer, 2000, pp. 1–15.

[3] D. M. Hawkins, "The problem of overfitting," *Journal of chemical information and computer sciences*, vol. 44, no. 1, pp. 1–12, 2004.

[4] V. F. Rodriguez-Galiano, B. Ghimire, J. Rogan, M. Chica-Olmo, and J. P. Rigol-Sanchez, "An assessment of the effectiveness of a random forest classifier for land-cover classification," *ISPRS journal of photogrammetry and remote sensing*, vol. 67, pp. 93–104, 2012.

[5] K. Fawagreh, M. M. Gaber, and E. Elyan, "Random forests: From early developments to recent advancements," *Systems Science & Control Engineering: An Open Access Journal*, vol. 2, no. 1, pp. 602–609, 2014.

[6] Z. Cao, T. Simon, S.-E. Wei, and Y. Sheikh, "Realtime multi-person 2d pose estimation using part affinity fields," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 7291–7299.

[7] F. Chollet, "Xception: Deep learning with depthwise separable convolutions," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 1251–1258.

[8] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam, "Mobilenets: Efficient convolutional neural networks for mobile vision applications," *arXiv preprint arXiv:1704.04861*, 2017.

[9] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger, "Densely connected convolutional networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 4700–4708.

[10] G. Huang, S. Liu, L. Van der Maaten, and K. Q. Weinberger, "Condensenet: An efficient densenet using learned group convolutions," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 2752–2761.

[11] P. Ghosal, L. Nandanwar, S. Kanchan, A. Bhadra, J. Chakraborty, and D. Nandi, "Brain tumor classification using resnet-101 based squeeze and excitation deep neural network," in *2019 Second International Conference on Advanced Computational and Communication Paradigms (ICACCP)*, IEEE, 2019, pp. 1–6.

[12] T. K. K. Maddala, P. Kishore, K. K. Eepuri, and A. K. Dande, "Yoganet: 3-d yoga asana recognition using joint angular displacement maps with convnets," *IEEE Transactions on Multimedia*, vol. 21, no. 10, pp. 2492–2503, 2019.

[13] A. S. B. Reddy and D. S. Juliet, "Transfer learning with resnet-50 for malaria cell-image classification," in *2019 International Conference on Communication and Signal Processing (ICCSP)*, IEEE, 2019, pp. 0945–0949.

[14] Q. Xiang, X. Wang, R. Li, G. Zhang, J. Lai, and Q. Hu, "Fruit image classification based on mobilenetv2 with transfer learning technique," in *Proceedings of the 3rd International Conference on Computer Science and Application Engineering*, 2019, pp. 1–7.

[15] S. K. Yadav, A. Singh, A. Gupta, and J. L. Raheja, "Real-time yoga recognition using deep learning," *Neural Computing and Applications*, vol. 31, no. 12, pp. 9349–9361, 2019.

[16] G. G. Chiddarwar, A. Ranjane, M. Chindhe, R. Deodhar, and P. Gangamwar, "Ai-based yoga pose estimation for android application," *Int J Inn Scien Res Tech*, vol. 5, pp. 1070–1073, 2020.

[17] S. Kothari, "Yoga pose classification using deep learning," 2020.

[18] M. Verma, S. Kumawat, Y. Nakashima, and S. Raman, "Yoga-82: A new dataset for fine-grained classification of human poses," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, 2020, pp. 1038–1039.

[19] A. Jaiswal, N. Gianchandani, D. Singh, V. Kumar, and M. Kaur, "Classification of the covid-19 infected patients using densenet201 based deep transfer learning," *Journal of Biomolecular Structure and Dynamics*, vol. 39, no. 15, pp. 5682–5689, 2021.

[20] J. Jose and S. Shailesh, "Yoga asana identification: A deep learning approach," in *IOP Conference Series: Materials Science and Engineering*, IOP Publishing, vol. 1110, 2021, p. 012 002.

[21] S. Liaqat, K. Dashtipour, K. Arshad, K. Assaleh, and N. Ramzan, "A hybrid posture detection framework: Integrating machine learning and deep neural networks," *IEEE Sensors Journal*, vol. 21, no. 7, pp. 9515–9522, 2021.

[22] J. Palanimeera and K. Ponmozhi, "Classification of yoga pose using machine learning techniques," *Materials Today: Proceedings*, vol. 37, pp. 2930–2933, 2021.