



BSCS-S22-004

03-134191-035 SYED SHAHAN TAHIR

03-134191-055 NAVEED AKRAM

# **Human Activity Recognition Using Deep learning**

In partial fulfilment of the requirements for the degree of  
**Bachelor of Science in Computer Science**

Supervisor: Dr. Iram Noreen

Department of Computer Sciences  
Bahria University, Lahore Campus

January 2023



# Certificate



We accept the work contained in the report titled  
“Human Activity Recognition Using Deep learning”

written by

SYED SHAHAN TAHIR

NAVEED AKRAM

as a confirmation to the required standard for the partial fulfilment of the degree of  
Bachelor of Science in Computer Science.

Approved by:

Supervisor: Dr. Iram Noreen

\_\_\_\_\_  
(Signature)

Jan 10, 2023

**DECLARATION**

We hereby declare that this project report is based on our original work except for citations and quotations which have been duly acknowledged. We also declare that it has not been previously and concurrently submitted for any other degree or award at Bahria University or other institutions.

Enrolment	Name	Signature
03-134191-035	SYED SHAHAN TAHIR	
03-134191-055	NAVEED AKRAM	

Date : Jan 10, 2023

Specially dedicated to  
my parents and teachers  
(SYED SHAHAN TAHIR)  
my parents and teachers  
(NAVEED AKRAM)

## **ACKNOWLEDGEMENTS**

We would like to thank everyone who had contributed to the successful completion of this project. We would like to express our gratitude to my research supervisor, Dr. Iram Noreen for his invaluable advice, guidance, and his enormous patience throughout the development of the research.

In addition, we would also like to express my gratitude to our loving parent and friends who had helped and given me encouragement.

**SYED SHAHAN TAHIR**  
**NAVEED AKRAM**

## Human Activity Recognition Using Deep learning

### ABSTRACT

Majority of computer vision tasks, such as Human Activity Recognition (HAR), are closely tied to security, virtual reality, video surveillance, and home monitoring applications. This sets a new trend and turning point in the HAR system development cycle. The problem is that most applications based on HAR available are low in accuracy due to which the percentage of faulty recognition of human activities and false alerts is quite high.

For the development of the system, we have selected UCF-SPORTS dataset which contains 9118 images. Dataset consists of images having different activities, which are classify according to the requirement of the system as their names. This dataset is used to train the model. Multiple Transfer Learning models are used for the development of this system. Design of the model is based on Xception, Inception, and MobileNet along with stack of different layers to improve the accuracy. Method of hyper-parameter tuning is used to analyse the behaviour of the discussed model. Model is tested and trained for different range of values of Batch size, learning rate and epochs. Performance of the model is validated through 5-fold cross validation. Experiments and analysis show that model detects activities with almost 97% cross validation accuracy.

Human Activity Recognition System (HARS) is developed as a mobile application for ease of use. In this system trained model is imported at the backend of the application which is used to detect human activities. System detects activities by

extracting frames from live feeds through camera. Moreover, system also notifies the app user by alerts if there's any suspicious activity to take safety measures as early as possible. In this age where computers can now perceive and analyse their environment with high accuracy, the detection system use latest and optimized model which specializes in activity detection. . Application will be able to recognize and classify several group of activities. This application is developed using Kivy a python framework and ML/AI kit.



## TABLE OF CONTENTS

<b>DECLARATION</b>	<b>ii</b>
<b>ACKNOWLEDGEMENTS</b>	<b>iv</b>
<b>ABSTRACT</b>	<b>v</b>
<b>TABLE OF CONTENTS</b>	<b>vii</b>
<b>LIST OF TABLES</b>	<b>xi</b>
<b>LIST OF FIGURES</b>	<b>xii</b>
<b>LIST OF SYMBOLS / ABBREVIATIONS</b>	<b>1</b>

## CHAPTERS

<b>1</b>	<b>INTRODUCTION</b>	<b>2</b>
	1.1 Background	2
	1.2 Problem Statements	3
	1.3 Aims and Objectives	4
	1.4 Scope of Project	4
	1.5 Dataset	4
	1.6 Transfer Learning Models	6
	1.6.1 Inception	6
	1.6.2 MobileNet V2	10
	1.6.3 Xception	14
	1.7 Final Deliverable of the Project and Beneficiaries	15
<b>2</b>	<b>SOFTWARE REQUIREMENT SPECIFICATION</b>	<b>16</b>
	2.1 Introduction	16
	2.2 User Classes and Characteristics	16

2.3	Operating Environment	17
2.4	Development System	17
	2.4.1 Design and Implementation Constraints	18
2.5	Assumptions and Dependencies	18
2.6	External Interface Requirements	18
	2.6.1 User Interface	18
2.7	Software Interfaces	19
2.8	System Use Cases	20
	2.8.1 Sign Up (U1)	21
	2.8.2 Login (U2)	22
	2.8.3 View Result (U3)	23
	2.8.4 View Live Feed (U4)	25
	2.8.5 Upload (U5)	26
2.9	Other Non-functional Requirements	28
	2.9.1 Performance Requirements	28
	2.9.2 Safety Requirements	28
	2.9.3 Security Requirements	28
	2.9.4 Software Quality Attributes	29
<b>3</b>	<b>DESIGN AND METHODOLOGY</b>	<b>30</b>
3.1	Phase 1: Project Initiation	32
3.2	Phase 2: Benchmark Dataset Collection and Pre-processing	32
3.3	Phase 3: Model Design and Model Evaluation	33
	3.3.1 Model Design	33
	3.3.2 Model Verification and Validation	34
	3.3.3 Training Dataset	34
	3.3.4 Testing Data	35
	3.3.5 Validation	35
3.4	Phase 4: Application (Proof of concept)	36
3.5	Workflow of Application	36
3.6	Sequence Diagrams	37
	3.6.1 Sign Up	38
	3.6.2 Login	39

3.6.3	View Result	40
3.6.4	Upload	41
3.6.5	Live Feed	42
<b>4</b>	<b>EXPERIMENT AND IMPLEMENTATION</b>	<b>43</b>
4.1	Experimental Setup	43
4.1.1	Google Colab	43
4.1.2	Hyper Parameter Tuning	45
4.2	List of Experiment on Models	46
4.2.1	Experiment on Inception	47
4.2.2	Experiment on MobileNet	49
4.2.3	Experiment on Xception	51
4.2.4	Comparison of Models	53
4.3	Languages	54
4.3.1	Python	54
4.3.2	Framework	54
4.4	Tools	54
4.4.1	Vs Code	54
4.4.2	Emulator	55
<b>5</b>	<b>RESULTS AND DISCUSSIONS</b>	<b>56</b>
5.1	Result of Application	56
5.1.1	Introduction	56
5.1.2	Log In and Sign Up	58
5.1.3	Main Screen	60
5.1.4	Camera Screen	62
5.1.5	Upload Screen	63
5.2	Result of Model	64
<b>6</b>	<b>CONCLUSION AND RECOMMENDATIONS</b>	<b>67</b>
6.1	Project Achievements	67
6.2	Future Work	67
6.3	Implementation Issues and Challenges	68

6.4 Conclusion

**REFERENCES**

**LIST OF TABLES**

<b>TABLE</b>	<b>TITLE</b>	<b>PAGE</b>
Table 1.1:	Summary of characteristics of UCF Sports	5
Table 2.1:	Software Interface	19
Table 2.2:	Sign Up	21
Table 2.3:	Login	22
Table 2.4:	View Results	23
Table 2.5:	View Live Feed	25
Table 2.6:	Upload	26
Table 4.1:	Hyperparameters of Model	45
Table 4.2:	Experiments on Inception	47
Table 4.3:	Experiments on MobileNet	49
Table 4.4:	Experiments on Xception	51
Table 4.5:	Comparison of Models	53

## LIST OF FIGURES

<b>FIGURE</b>	<b>TITLE</b>	<b>PAGE</b>
	<b>Figure 1.1: Sample images of UCF Sports Dataset</b>	5
	<b>Figure 1.2: Smaller Convolution [8]</b>	7
	<b>Figure 1.3: Asymmetric convolutions [8]</b>	8
	<b>Figure 1.4: Auxiliary classifier [8]</b>	8
	<b>Figure 1.5: Grid size reduction [8]</b>	9
	<b>Figure 1.6: Final architecture [7]</b>	9
	<b>Figure 1.7: Convolutional Blocks[10]</b>	10
	<b>Figure 1.8: Architecture [10]</b>	11
	<b>Figure 1.9: Depthwise Separable Convolution [11]</b>	11
	<b>Figure 1.10: Pointwise Convolution [11]</b>	13
	<b>Figure 1.11: Convolution Layers [11]</b>	13
	<b>Figure 1.12: MobileNet vs CNN [11]</b>	14
	<b>Figure 2.1: User Use Case Diagram</b>	20
	<b>Figure 2.2: Signup Use Case Diagram</b>	22
	<b>Figure 2.3: Login Use Case Diagram</b>	23
	<b>Figure 2.4: View Result Use Case Diagram</b>	24
	<b>Figure 2.5: View Live Feed Use Case Diagram</b>	26
	<b>Figure 2.6: Upload Use Case Diagram</b>	27
	<b>Figure 3.1: Methodology Diagram</b>	31

<b>Figure 3.2: Work Flow Diagram</b>	37
<b>Figure 3.3: Signup sequence diagram</b>	38
<b>Figure 3.4: Login sequence diagram</b>	39
<b>Figure 3.5: View Result sequence diagram</b>	40
<b>Figure 3.6: Upload sequence diagram</b>	41
<b>Figure 3.7: View Live Feed sequence diagram</b>	42
<b>Figure 4.1: Confusion Matrix of Inception</b>	47
<b>Figure 4.2: Accuracy Loss Graph of Inception</b>	48
<b>Figure 4.3: Confusion Matrix of MobileNet</b>	49
<b>Figure 4.4: Accuracy Loss Graph of MobileNet</b>	50
<b>Figure 4.5: Confusion Matrix of Xception</b>	51
<b>Figure 4.6: Accuracy Loss Graph of Xception</b>	52
<b>Figure 5.1: Welcome Screen</b>	57
<b>Figure 5.2: Sign up screen</b>	58
<b>Figure 5.3: Log in Screen</b>	59
<b>Figure 5.4: Main screen</b>	61
<b>Figure 5.5: Camera Screen</b>	62
<b>Figure 5.6: Upload Screen</b>	63
<b>Figure 5.7: Uploading Image</b>	64
<b>Figure 5.8: Result after uploading image</b>	65
<b>Figure 5.9: Result from live feed</b>	66

**LIST OF SYMBOLS / ABBREVIATIONS**

HAR	Human Activity Recognition
HARS	Human Activity Recognition System
ML	Machine Learning
AI	Artificial Intelligence
CNN	Convolutional Neural Network
DNN	Deep Neural Network
FDD	Feature Driven Development



## CHAPTER 1

### INTRODUCTION

#### 1.1 Background

Human Activity Recognition (HAR) is an extensive discipline of study concerned with figuring out the particular motion or movement of a person primarily based on sensor data [1]. HAR is widely used in a variety of applications such as intelligent video surveillance systems, suspicious activity detection, and in the medical field. Movements are standard activities frequently performed, which includes walking, standing, and sitting [1]. We need an Application or Program which could recognize human activities and offer assistance in our everyday Lives, however on the same time, ensure that it does not record videos which could violate our privacy. Sensor data can be remotely recorded, consisting of video, radar, or different wireless methods. Alternately, data can be recorded directly at the subject by a custom hardware or by smart phones which have accelerometers and gyroscopes. There are not any apparent or direct methods to relate the recorded sensor information to precise human activities and every subject may also carry out an activity with great variation [1].

Our intent is to record sensor data and corresponding activities for particular subjects, fit a model from this data, and generalize the model to categorise the activity of recent unseen subjects from their sensor data. Many researchers have contributed innovative algorithms and approaches in the area of human action recognition system and have conducted experiments on individual data sets by considering accuracy and computation. Moreover, this field requires high accuracy with less computational complexity. The existing techniques are inadequate in accuracy due to assumptions

regarding clothing style, view angle and environment. Hence, the main objective of this is to develop an efficient multi-view based human action recognition system using deep learning.

Following are some deep learning approaches that are reported in literature for human activity recognition.

- 2-Dimensional Convolution Neural Network (2D-CNN) [2]
- Long Short Term Memory (LSTM) [3]
- 3-Dimensional Convolution Neural Network (3D-CNN) [4]
- Gaussian Mixture Modelling (GMM) [5]

We are using Kivy [6] for the development of HAR App. Kivy is a free and open-source Cross Development Platform with a natural user interface distributed under MIT License. Kivy uses Python as its official programming language.

## **1.2 Problem Statements**

Multimodal object perception and action description using multimodal methods of Human activities recognition (HAR) are detected either by traditional complex machine learning approaches involving manual engineering or by automated deep learning approaches which use high computing resources to perform well. A lightweight deep learning-assisted approach is desirable to solve the challenges of multiple variations in viewpoint for significant features in HAR. A lightweight deep learning-assisted framework for activity recognition.

### **1.3 Aims and Objectives**

The objectives of the thesis are shown as following:

- To develop an app that will recognize human activities in real time functionalities.
- To make an app that will alert users for suspicious activities.
- To provide users better surveillance

### **1.4 Scope of Project**

Primary Objective is to develop an efficient human action recognition system using deep learning. The scope of the project is two folds. At backend level, it is a deep learning based trained model to achieve HAR with feasible performance measure. At frontend level, it will provide a cross platform smart application to provide HAR services remotely to wide range of users.

### **1.5 Dataset**

We have used UCF Sports Dataset [19]. UCF Sports dataset consists of a set of actions collected from various sports which are typically featured on broadcast television channels such as the BBC and ESPN. The video sequences were obtained from a wide range of stock footage websites including BBC Motion gallery and Getty Images.

The dataset includes a total of 150 sequences with the resolution of 720 x 480. The collection represents a natural pool of actions featured in a wide range of scenes and viewpoints.

The dataset includes the following 10 actions.

- Diving (14 videos)
- Golf Swing (18 videos)
- Kicking (20 videos)
- Lifting (6 videos)
- Riding Horse (12 videos)
- Running (13 videos)
- Skateboarding (12 videos)
- Swing-Bench (20 videos)
- Swing-Side (13 videos)
- Walking (22 videos)

**Table 1.1: Summary of characteristics of UCF Sports [19]**

Actions	10	Total Duration	958 fps
Clips	150	Frame Rate	10 fps
Mean Clip Length		Resolution	720 x 480
Min Clip Length		Max no of clips per class	22
Max Clip Length		Min no of clips per class	6



**Figure 1.1: Sample images of UCF Sports Dataset [19]**

We have converted videos to frames in order to train our model which has 9118 images in total.

## 1.6 Transfer Learning Models

Following are the external interface requirements of this project:

### 1.6.1 Inception

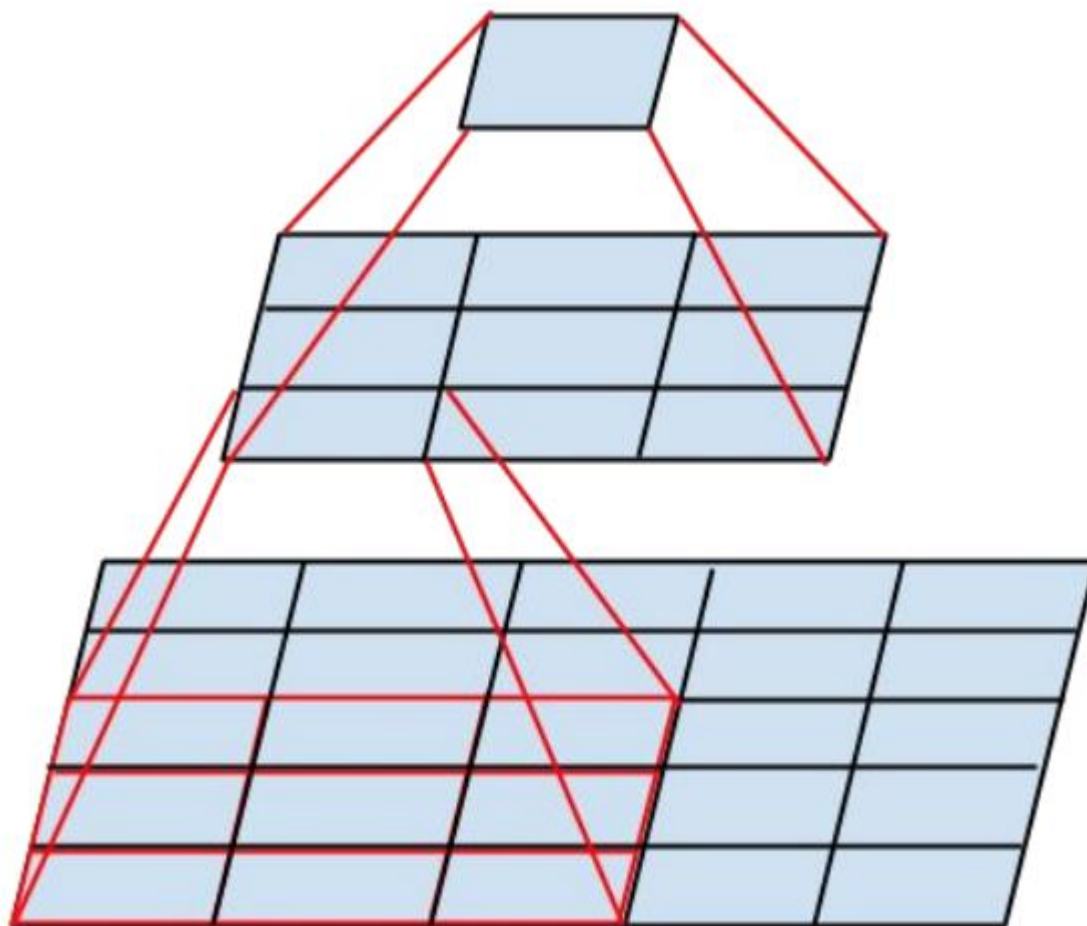
The Inception V3 [7] is a deep learning model used for image classification based on Convolutional Neural Networks. The inception V3 is a 3rd version of Inception model. It uses techniques like factorizing larger convolutions to smaller convolutions (say a 5X5 convolution is factorized into two 3X3 convolutions) and asymmetric factorizations (example: factorizing a 3X3 filter into a 1X3 and 3X1 filter). These factorizations are done with the aim of reducing the number of parameters being used at every inception module. The main focus of inception v3 is to use less computational power by changing the architectures of previous versions

In comparison to other models, Inception is superior in relationship of the number of parameters generated by the network and computational cost. While changing the Inception model care must be taken to avoid the computational loss. Thus, the adaptation of an Inception network for different use cases turns out to be a problem due to the uncertainty of the new network's efficiency. Several techniques are used for the adaption of the network. The techniques include factorized convolutions, regularization, dimension reduction, and parallelized computations.

The architecture of an Inception v3 network is built gradually, step-by-step, as explained below:

**1. Factorized Convolutions:** this helps to reduce the computational efficiency as it reduces the number of parameters involved in a network. It also keeps a check on the network efficiency.

**2. Smaller convolutions:** replacing bigger convolutions with smaller convolutions definitely leads to faster training. Say a  $5 \times 5$  filter has 25 parameters; two  $3 \times 3$  filters replacing a  $5 \times 5$  convolution has only 18 ( $3 \times 3 + 3 \times 3$ ) parameters instead.



**Figure 1.2: Smaller Convolution [8]**

In the middle we see a  $3 \times 3$  convolution, and below a fully-connected layer. Since both  $3 \times 3$  convolutions can share weights among themselves, the number of computations can be reduced.

**3. Asymmetric convolutions:** A  $3 \times 3$  convolution could be replaced by a  $1 \times 3$  convolution followed by a  $3 \times 1$  convolution. If a  $3 \times 3$  convolution is replaced by a  $2 \times 2$  convolution, the number of parameters would be slightly higher than the asymmetric convolution proposed.

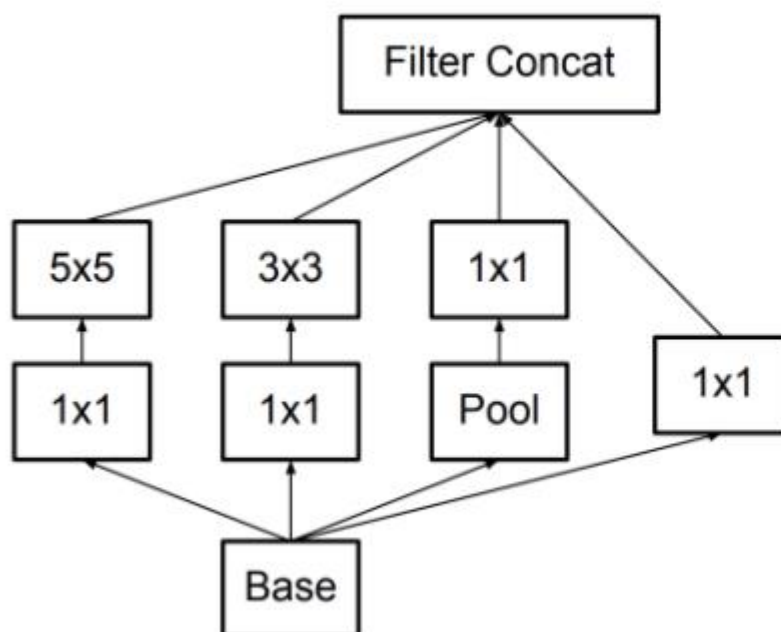


Figure 1.3: Asymmetric convolutions [8]

**4. Auxiliary classifier:** an auxiliary classifier is a small CNN inserted between layers during training, and the loss incurred is added to the main network loss. In Inception v3 an auxiliary classifier acts as a regularizer.

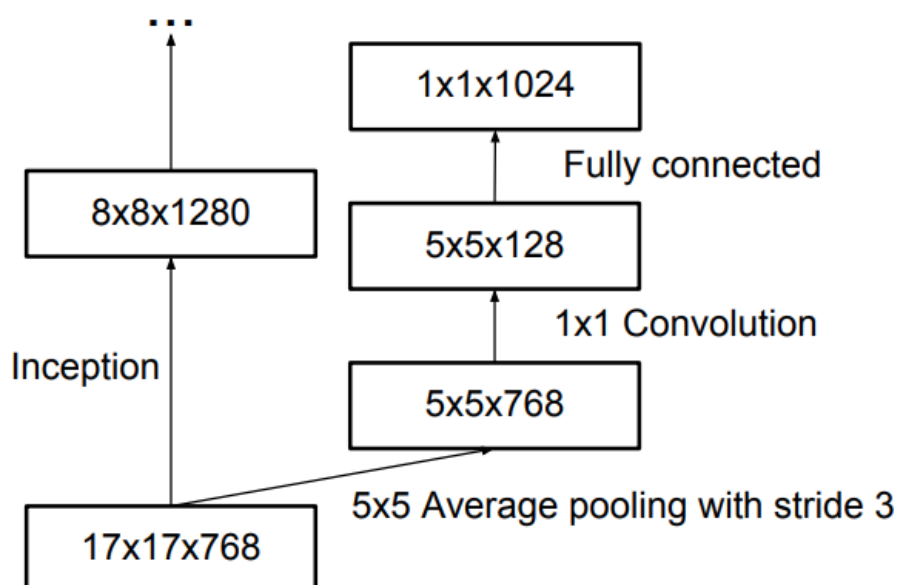
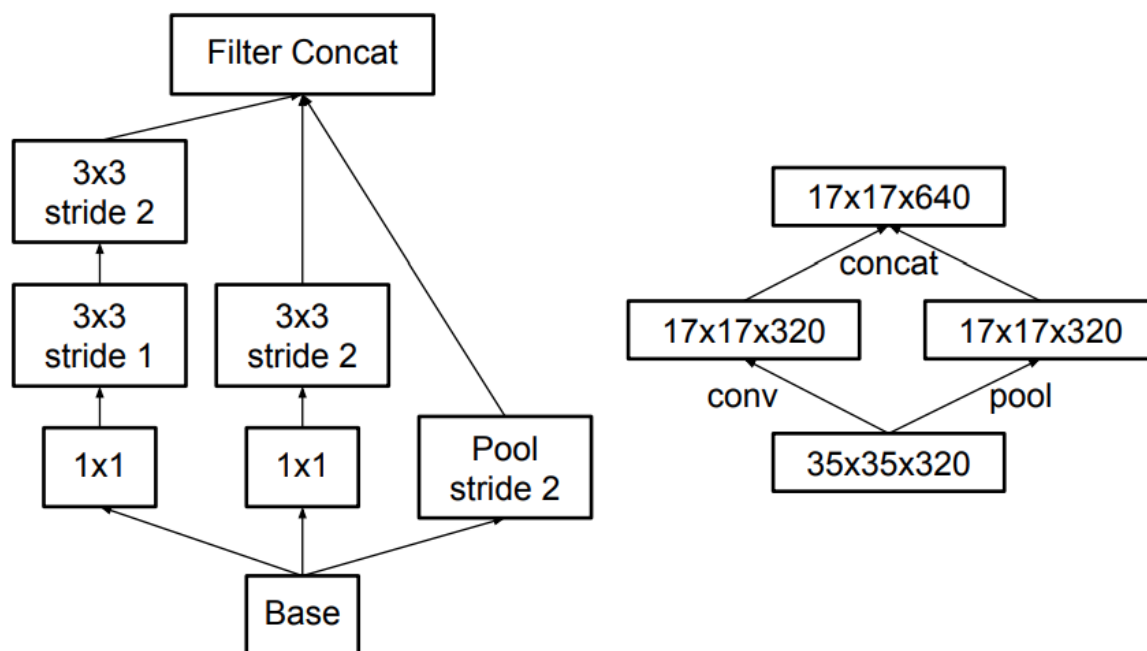


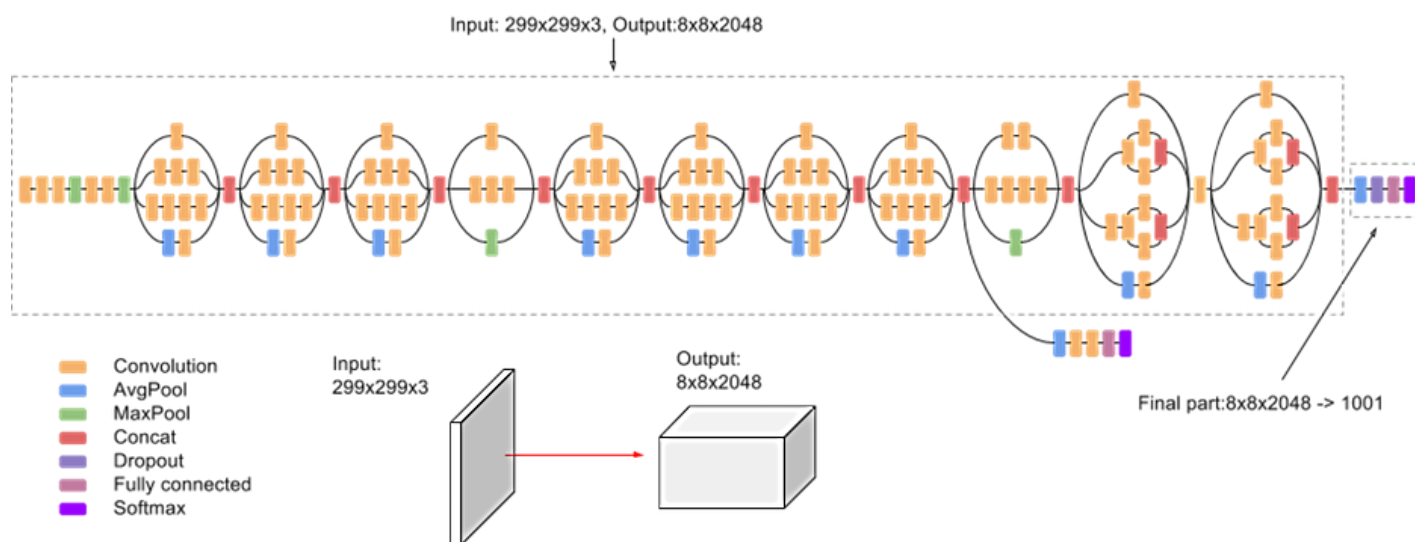
Figure 1.4: Auxiliary classifier [8]

**5. Grid size reduction:** Grid size reduction is usually done by pooling operations.



**Figure 1.5: Grid size reduction [8]**

All the above concepts are consolidated into the final architecture.

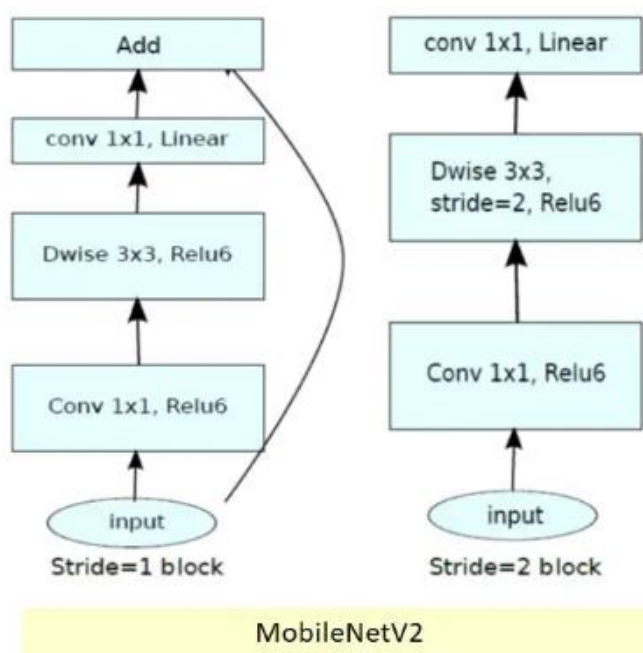


**Figure 1.6: Final architecture [7]**



### 1.6.2 MobileNet V2

MobileNet V2 [9] is a lightweight deep neural network which uses Depthwise separable convolutions. Regular convolutions have more parameters which are significantly reduced in MobileNet. MobileNet gives excellent starting point for training classifiers that are insanely small and insanely fast.



**Figure 1.7: Convolutional Blocks[10]**

Input	Operator	$t$	$c$	$n$	$s$
$224^2 \times 3$	conv2d	-	32	1	2
$112^2 \times 32$	bottleneck	1	16	1	1
$112^2 \times 16$	bottleneck	6	24	2	2
$56^2 \times 24$	bottleneck	6	32	3	2
$28^2 \times 32$	bottleneck	6	64	4	2
$14^2 \times 64$	bottleneck	6	96	3	1
$14^2 \times 96$	bottleneck	6	160	3	2
$7^2 \times 160$	bottleneck	6	320	1	1
$7^2 \times 320$	conv2d 1x1	-	1280	1	1
$7^2 \times 1280$	avgpool 7x7	-	-	1	-
$1 \times 1 \times 1280$	conv2d 1x1	-	k	-	-

**Figure 1.8: Architecture [10]**

The Following are the layers of model:

Depthwise Separable Convolution is convolution originated from the idea that a filter's depth and spatial dimension can be separated- thus, the name separable. Let us take the example of Sobel filter, used in image processing to detect edges.

-1	0	+1
-2	0	+2
-1	0	+1

Gx

+1	+2	+1
0	0	0
-1	-2	-1

Gy

**Figure 1.9: Depthwise Separable Convolution [11]**

You can separate the height and width dimensions of these filters. Gx filter can be viewed as a matrix product of  $\begin{bmatrix} 1 & 2 & 1 \end{bmatrix}$  transpose with  $\begin{bmatrix} -1 & 0 & 1 \end{bmatrix}$ . We notice that the filter had disguised itself. It shows it had nine parameters, but it has 6. This has been possible because of the separation of its height and width dimensions. The same idea applied to separate depth dimension from horizontal (width\*height) gives us depth-wise separable convolution where we perform depth-wise convolution. After that, we use a  $1 \times 1$  filter to cover the depth dimension. To produce one channel, we need  $3 \times 3 \times 3$  parameters to perform depth-wise convolution and  $1 \times 3$  parameters to perform further convolution in-depth dimension. But If we need three output channels, we only need  $3 \times 3$  depth filter, giving us a total of 36 ( $= 27 + 9$ ) parameters while for the same no. of output channels in regular convolution, we need  $3 \times 3 \times 3$  filters giving us a total of 81 parameters.

Depthwise separable convolution is a depthwise convolution followed by a pointwise convolution which is shown in the figure 1.9:

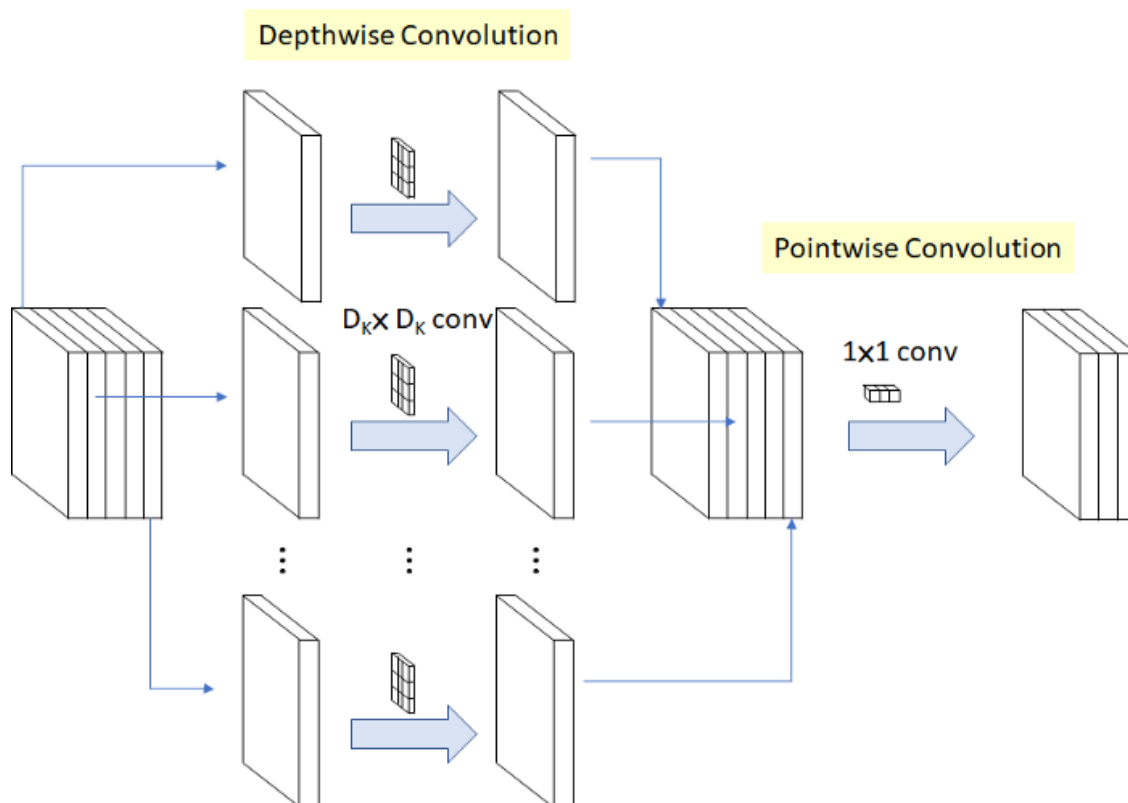


Figure 1.10: Pointwise Convolution [11]

1. **Depthwise convolution** is the channel-wise  $D_K \times D_K$  spatial convolution.
2. **Pointwise convolution** is the  $1 \times 1$  convolution to change the dimension.
3. **Depthwise convolution.**

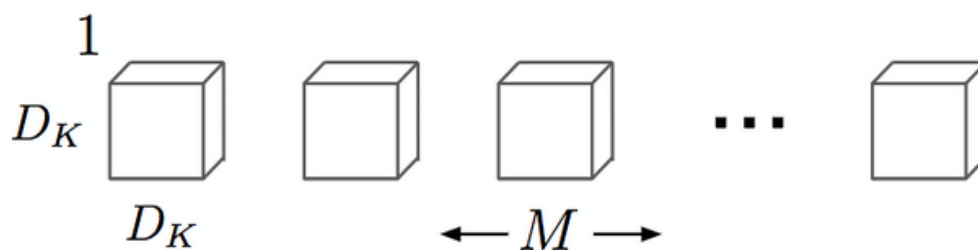
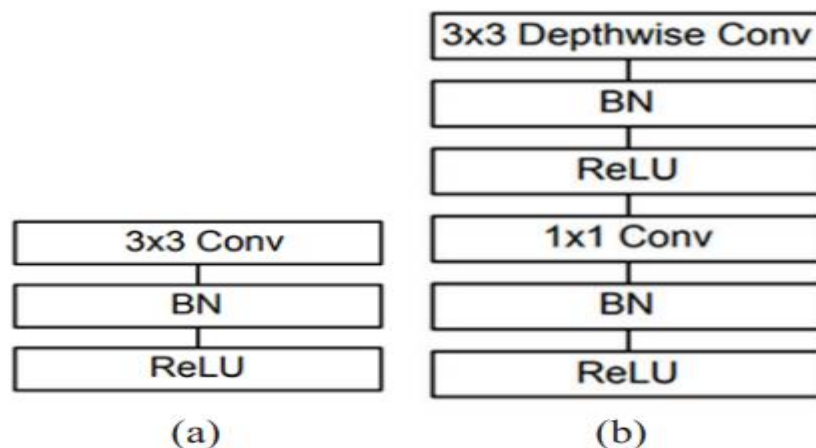


Figure 1.11: Convolution Layers [11]

The main difference between MobileNet architecture and a traditional CNN instead of a single  $3 \times 3$  convolution layer followed by the batch norm and ReLU. Mobile

Nets split the convolution into a 3x3 depth-wise conv and a 1x1 pointwise conv, as shown in the figure 1.11:



**Figure 1.12: MobileNet vs CNN [11]**

### 1.6.3 Xception

Xception [12] is a deep convolutional neural network architecture that involves Depthwise Separable Convolutions. It was developed by Google researchers. It is an interpretation of Inception modules in convolutional neural networks as being an intermediate step in-between regular convolution and the depthwise separable convolution operation. Xception stands for “extreme inception”, it takes the principles of Inception to an extreme. In Inception, 1x1 convolutions were used to compress the original input, and from each of those input spaces we used different type of filters on each of the depth space. Xception just reverses this step. Instead, it first applies the filters on each of the depth map and then finally compresses the input space using 1X1 convolution by applying it across the depth. This method is almost identical to a depthwise separable convolution. There is one more difference between Inception and Xception. The presence or absence of a non-linearity after the first operation. In Inception model, both operations are followed by a ReLU non-linearity, however Xception doesn't introduce any non-linearity.[13]

An inception network is a deep neural network (DNN) with a design that consists of repeating modules referred to as inception modules. In general, each layer of DNN is considered to extract some feature, then stacking these layers one above each other is not a great idea. Deep networks are prone to overfitting, and chaining multiple convolutional operations together increases the cost to train the network. Another issue is as each layer type extracts a different kind of information, how do we know which transformation (kernels) provides the most useful information to the DNN.

An Inception module computes multiple different transformations over the same input and then finally combining all the output which lets the model decide what features to take and by how much. There is one problem. It is still computationally inefficient because of convolutions. These convolutions not only happens spatially, but also across the depth. So, for each additional filter, we have to perform convolution over the input depth to calculate just a single output map, and because of this, the depth becomes a huge bottleneck in the DNN. The depth can be reduced by doing 1X1 convolution across the depth. This convolution looks across multiple channel's spatial information and compress it down to a lower dimension. For example, using 30 1x1 filters, input of size 128x128x100 (with 100 feature maps) can be compressed down to 128x128x30. Due to this reduction, the researchers of Inception module were able to concatenate different layer transformations in parallel, resulting in DNN that was wide and deep. The images below shows difference between Inception module without 1X1 filters and one with 1X1 filters.

## **1.7 Final Deliverable of the Project and Beneficiaries**

The final deliverables of this project will be an integrated system comprising of hardware and software. The beneficiaries of this system are people from the area where this system is installed.

## CHAPTER 2

### SOFTWARE REQUIREMENT SPECIFICATION

#### 2.1 Introduction

HAR App is Machine Learning based detection system which uses concepts and algorithms of machine learning and computer vision. It will be used to detect activities and make detection more reliable. Machine learning is a method of data analysis that automates analytical model building. It is a branch of artificial intelligence based on the idea that systems can learn from data, identify patterns and make decisions with minimal human intervention [4].

SRS stands for software requirement specification. This document will describe the functional and non-functional requirement for the system. The limitation that the application will have and reasons for the limitation. The system main features and blueprint of the system. The system requirements of HAR App are discussed below.

#### 2.2 User Classes and Characteristics

This software is created for two main user classes:

**User** – A user will be able to perform the following activities:

- Sign in

- Sign-out
- Login
- Forgot Password
- View Live Stream
- See Result

### **2.3 Operating Environment**

Mobile device with the following (minimum) specifications:

- 2 GB RAM
- 1 GB free storage
- Internet
- Android 6.0 marshmallow

### **2.4 Development System**

- Visual Studio Code [14]
- Kivy [6]
- Google Colab [15]
- Kaggle [16]
- Adobe XD [17]
- Adobe Illustrator [18]



### **2.4.1 Design and Implementation Constraints**

HAR app is a mobile application, and the front end of the Application is designed using Abode XD and Kivymd. For model training Google Colab and Kaggle is used. App will be developed using FDD which divides complex structures into subparts. This model recurrence last two phases i.e., Design by feature and Develop by Feature until all features will complete. Time is a major constrain as trained model is integrated in the app and to predict activity time is required. Model accuracy and dataset is also a constraint for the application.

### **2.5 Assumptions and Dependencies**

We are assuming that the user already has the knowledge required to operate this application. Camera is a must dependency for this software. Without these services' application won't be able to run.

### **2.6 External Interface Requirements**

Following are the external interface requirements of this project:

#### **2.6.1 User Interface**

- Full-screen application
- Splash screen
- UI/UX includes the following components:

- Image viewer
- Buttons
- Text Views
- Google defined UI constraint standards
- Material design theme

## 2.7 Software Interfaces

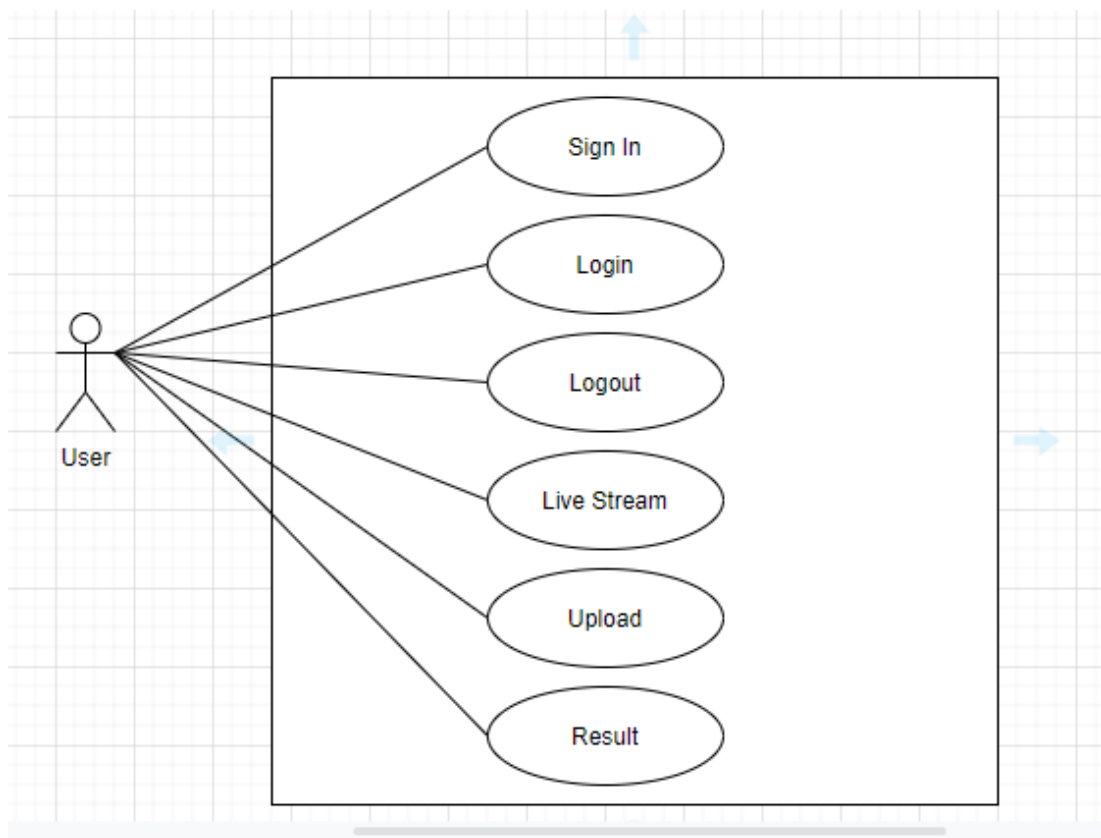
Following are the required software interfaces:

**Table 2.1: Software Interface**

<b>Android Version</b>	6.0+	This app will run on mobiles that are running Android version 6.0+
<b>Model Training</b>	Google Colab, Kaggle	Google Colab and Kaggle are used to train the models used.
<b>Tools</b>	Visual Studio Code	Visual Studio Code is the development platform that'll be used to develop this application using the Kivy framework.

## 2.8 System Use Cases

The following are the use cases for 'HAR App'. In these use cases User will be the actor that will interact with the presented system.



**Figure 2.1: User Use Case Diagram**

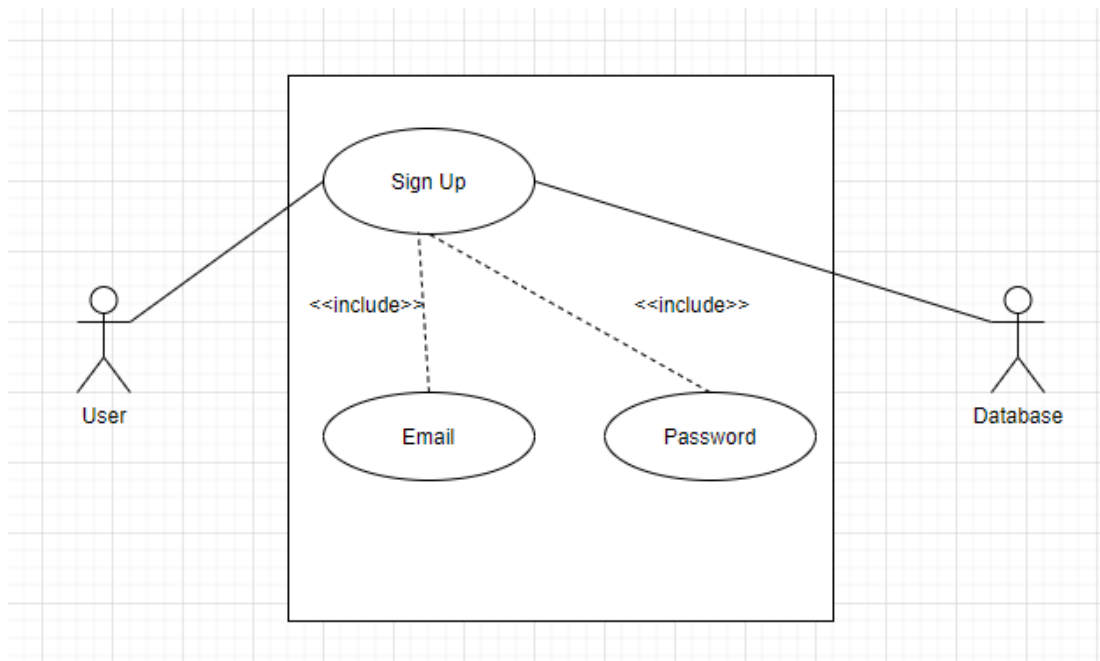
### 2.8.1 Sign Up (U1)

It would be used to register new admin in the system.

**Table 2.2: Sign Up**

	Name	Signup
1.	<b>Use-Case ID</b>	U1
2.	<b>Objective</b>	User will sign up in the system
3.	<b>Priority</b>	Low
4.	<b>Source</b>	Naveed Akram(Developer)
5.	<b>Actors</b>	User
6.	<b>Flow of Events</b>	<ul style="list-style-type: none"> <li>• Run the Software Application</li> <li>• Enter Email &amp; Password</li> <li>• Press Sign-up button</li> </ul>
6.1	<b>Basic Flow</b>	Information is entered and new user is created and can now login.
6.2	<b>Alternate Flow(s)</b>	No Alternative Flows
6.3	<b>Exception Flow(s)</b>	Information should be correctly added while signing up.
7.	<b>Includes</b>	No other Use case
8.	<b>Preconditions</b>	The user must be Signed-In.
9.	<b>Postconditions</b>	CRUD operations are performed.
10.	<b>Notes/Issues</b>	None.

If the user is using the system for the first time he will need to sign up in the application. For that purpose, he/she will enter his detail for sign up. The detail will then be sent to server side and it will be saved there. So, the admin could login again whenever he wants to. If some type of error occurs, error message will be displayed. Graphical representation of the use case is as show in Fig 2.1 Sign Up:



**Figure 2.2: Signup Use Case Diagram**

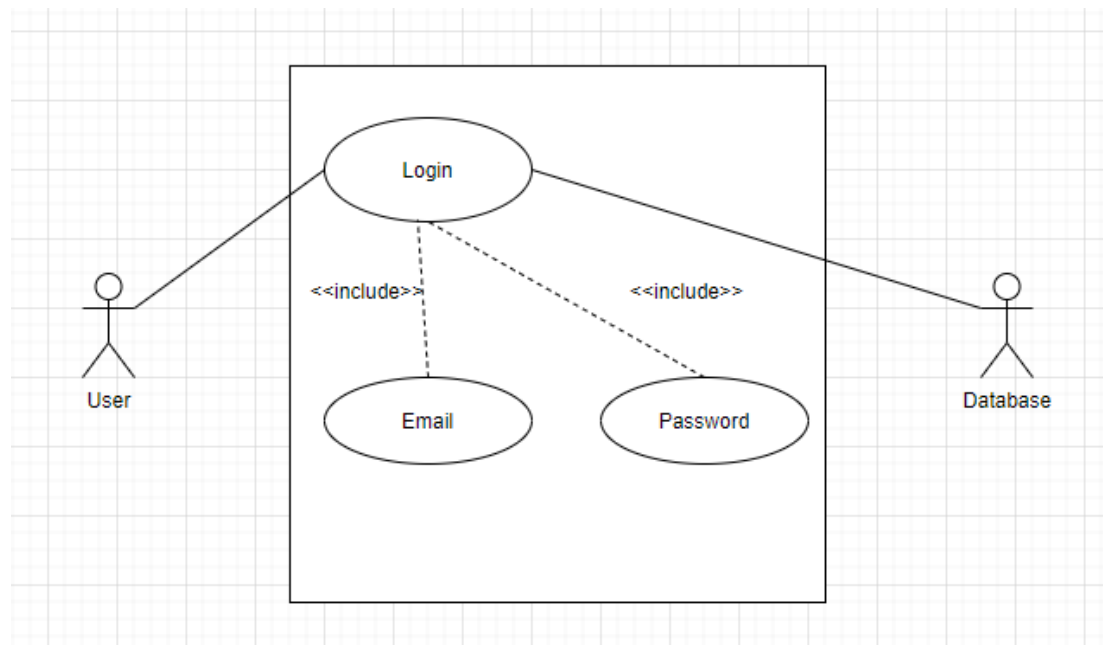
### 2.8.2 Login (U2)

**Table 2.3: Login**

	Name	Login
1.	<b>Use-Case ID</b>	U2
2.	<b>Objective</b>	Admin can login into system
3.	<b>Priority</b>	Low
4.	<b>Source</b>	Naveed Akram(Developer)
5.	<b>Actors</b>	User
6.	<b>Flow of Events</b>	<ul style="list-style-type: none"> <li>• Run the Software Application</li> <li>• Click the Login Option</li> <li>• Enter Details</li> <li>• Press Login Button</li> </ul>
6.1	<b>Basic Flow</b>	Information is entered and user can login.
6.2	<b>Alternate Flow(s)</b>	No Alternative Flows
6.3	<b>Exception Flow(s)</b>	User should enter correct information
7.	<b>Includes</b>	No other Use case

8.	<b>Preconditions</b>	Software must be installed
9.	<b>Postconditions</b>	User has logged in. Home screen will appear after it
10.	<b>Notes/Issues</b>	None.

If the user wants to access and use the system. For that purpose, he/she will enter his detail for login. The detail will then be sent to server side and it will be validated there. If details are correct admin will log into the system. If some type of error occurs, error message will be displayed.



**Figure 2.3: Login Use Case Diagram**

### 2.8.3 View Result (U3)

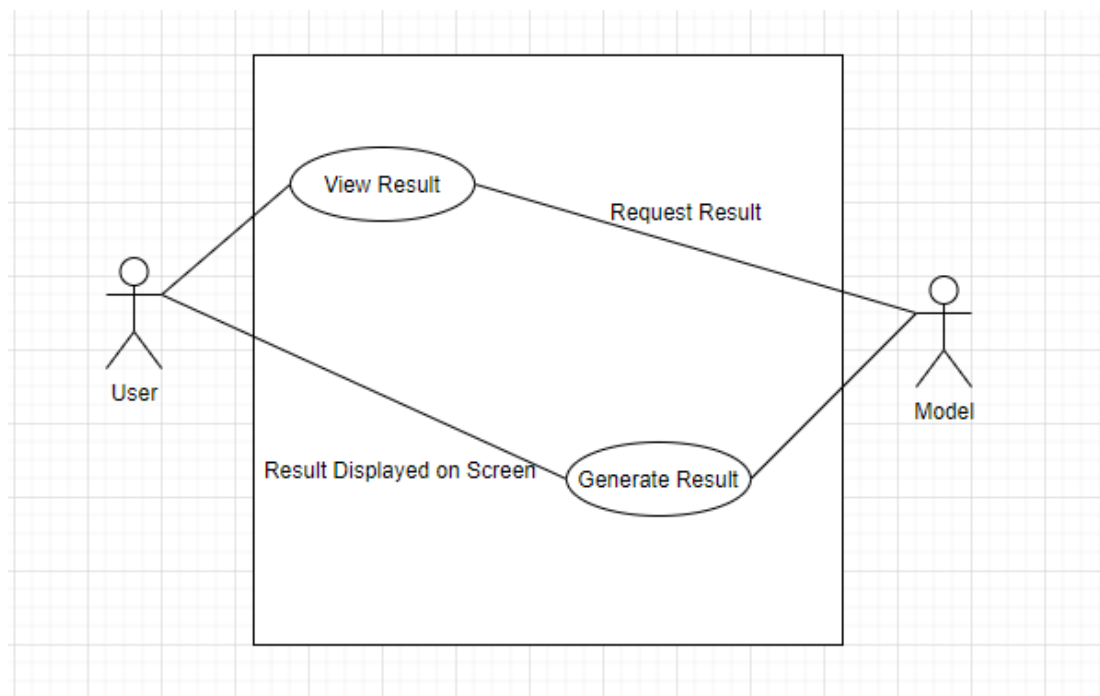
**Table 2.4: View Results**

	<b>Name</b>	<b>View Result</b>
1.	<b>Use-Case ID</b>	U3
2.	<b>Objective</b>	View Result
3.	<b>Priority</b>	Low
4.	<b>Source</b>	Naveed Akram(Developer)

5.	<b>Actors</b>	User
6.	<b>Flow of Events</b>	<ul style="list-style-type: none"> <li>• After successful login in system</li> <li>• Click on capture</li> </ul>
6.1	<b>Basic Flow</b>	Frame will be captures and result will be displayed on the screen
6.2	<b>Alternate Flow(s)</b>	No Alternative Flows
6.3	<b>Exception Flow(s)</b>	Admin should enter correct information
7.	<b>Includes</b>	U1
8.	<b>Preconditions</b>	System must be installed first and Admin must Login into the system.
9.	<b>Postconditions</b>	Result has been generated. Result will be displayed on Screen.
10.	<b>Notes/Issues</b>	None.

If the user wants to generate the result of recent activity. He/she will have to login into the system and then press Generate report button. If some type of error occurs, error message will be displayed.

Graphical representation of the use case is shown in figure 2.4:



**Figure 2.4: View Result Use Case Diagram**

### 2.8.4 View Live Feed (U4)

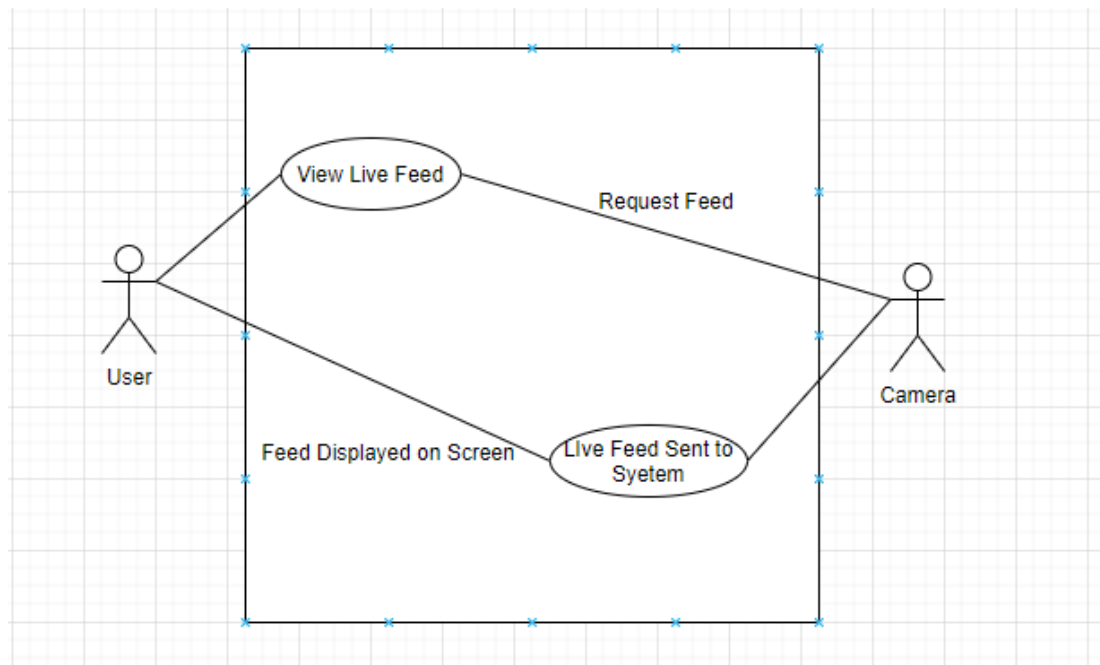
It would be used to view live stream of camera that is connected to the system.

**Table 2.5: View Live Feed**

	<b>Name</b>	View Live Feed
1.	<b>Use-Case ID</b>	U4
2.	<b>Objective</b>	View Live Feed
3.	<b>Priority</b>	Low
4.	<b>Source</b>	Naveed Akram(Developer)
5.	<b>Actors</b>	User
6.	<b>Flow of Events</b>	<ul style="list-style-type: none"> <li>• After successful login in system</li> <li>• Click on Play Button</li> </ul>
6.1	<b>Basic Flow</b>	After login into the system, default page that is opened is view live stream. Press Play there to view the live stream.
6.2	<b>Alternate Flow(s)</b>	No Alternative Flows
6.3	<b>Exception Flow(s)</b>	Information should be correctly added while signing up.
7.	<b>Includes</b>	U2
8.	<b>Preconditions</b>	User
9.	<b>Postconditions</b>	CRUD operations are performed.
10.	<b>Notes/Issues</b>	None.

If the user wants to view the live stream of the camera to see what is happening around. He/she will have to login into the system and then press Start button. If some type of error occurs, error message will be displayed.





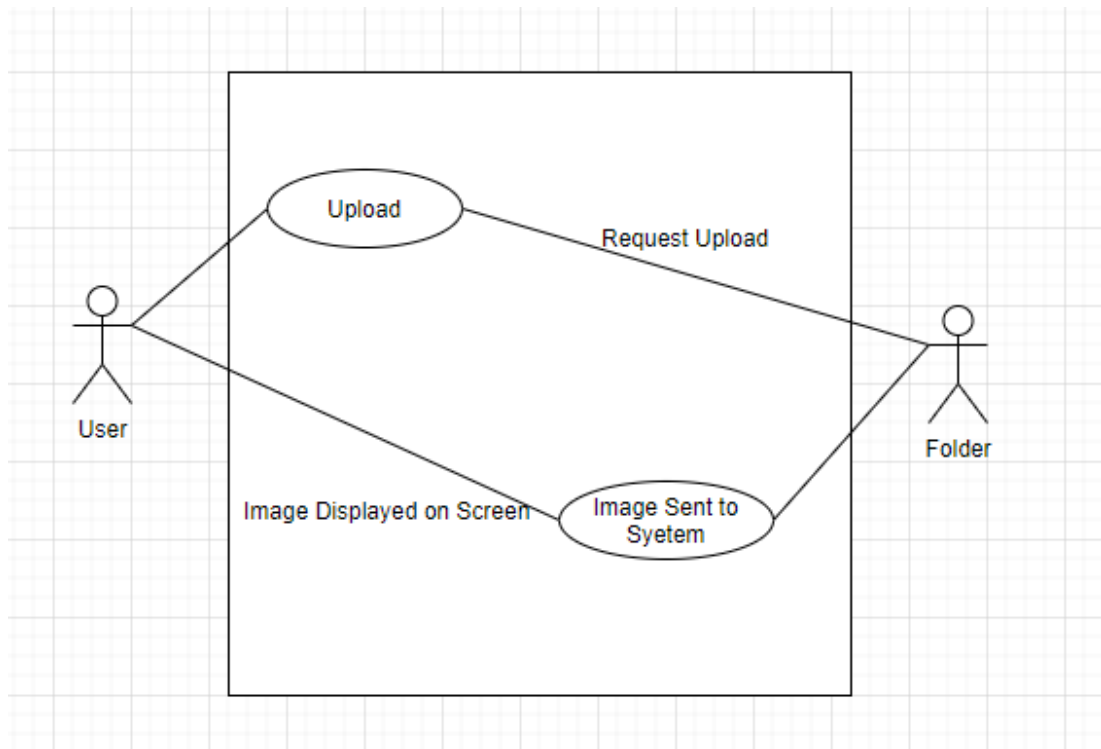
**Figure 2.5: View Live Feed Use Case Diagram**

### 2.8.5 Upload (U5)

**Table 2.6: Upload**

	Name	Signup
1.	<b>Use-Case ID</b>	U1
2.	<b>Objective</b>	User will sign up in the system
3.	<b>Priority</b>	Low
4.	<b>Source</b>	Naveed Akram(Developer)
5.	<b>Actors</b>	User
6.	<b>Flow of Events</b>	<ul style="list-style-type: none"> <li>• Run the Software Application</li> <li>• Enter Email &amp; Password</li> <li>• Press Sign-up button</li> </ul>
6.1	<b>Basic Flow</b>	Information is entered and new user is created and can now login.
6.2	<b>Alternate Flow(s)</b>	No Alternative Flows
6.3	<b>Exception Flow(s)</b>	Information should be correctly added while signing up.
7.	<b>Includes</b>	No other Use case

8.	<b>Preconditions</b>	System must be installed first and User must Login into the system.
9.	<b>Postconditions</b>	Live Feed has been displayed on Screen.
10.	<b>Notes/Issues</b>	None.



**Figure 2.6: Upload Use Case Diagram**

## **2.9 Other Non-functional Requirements**

Non-functional requirements of this project are:

### **2.9.1 Performance Requirements**

Performance of system depends upon the speed of its response. If malicious activity is present, system should immediately notify the situation to the owner. The speed of its response time should be very fast. Error rate of the software should be close to negligible.

### **2.9.2 Safety Requirements**

To provide the users with the best application experience we will bring time to time updates in our application to prevent any bugs and try to fix those bugs and errors. As the system comprises of both hardware and software. Safety measures are different for both of them. As it will have a camera, so protection of that camera against things that can damage it is essential. Camera should be installed in damp and dry place. Software should only be deployed when it is tested and verified.

### **2.9.3 Security Requirements**

System can be used inside or outside of the building and video data of all the happenings and activities are passed to system which is only used for detecting activities. Frames are being recorded that should not be used for any other reason keeping user activity private.

#### 2.9.4 Software Quality Attributes

To ensure the better quality of system the camera should be of good quality. Quality of camera affects performance of the detection. Besides this the system should only be deployed when performance is verified and is ensured.

Software quality attributes of this project are:

- **Availability:** The application will be available for the user 24/7.
- **Flexibility:** The application would be flexible for any type of user.
- **Usability:** The application should be user-friendly for the user. The users easily understand how to use the application.
- **Testability:** The application should be easy to test at each level and find the bugs/defect at each level of development and remove the defects easily.
- **Reusability:** The application is divided into different modules of coding. These modules can be used across the application.
- **Maintainability:** The application will be easy to maintain and removing bugs/errors and upgrade the application features, functionalities from time to time.

## CHAPTER 3

### DESIGN AND METHODOLOGY

In recent years activity detection system has been adopted due to the rapid development in the field of machine learning. There are many machine learning techniques and algorithms available which can detect hundreds of thousands of human activities. These techniques include neural networks, ID3 algorithm, k-nearest algorithm and many more. HARS has the capability to solve many problems. To detect Human Activity Dataset consists of Human activity images is required. To meet the requirements, system has been sub-divided into four phases. It is observed that life is full of problems but in a positive manner it helps us to identify the solutions which makes our life easier. So, first phase for the system is to identify the need for the system. Secondly, as system is using machine learning techniques for object detection dataset must be generated or collected. So, second phase of the system include collection of datasets and pre-processing of the dataset to make it useable for system to detect the object. It is very important to plan a design for the model which is used to detect human activity. After designing the model, verification and validation of the model is necessary to minimise the chances of failure this is the third phase of the system. Final phase of the system to run the application along with proof of concepts. Summary of the phases can be seen in the figure 3.1

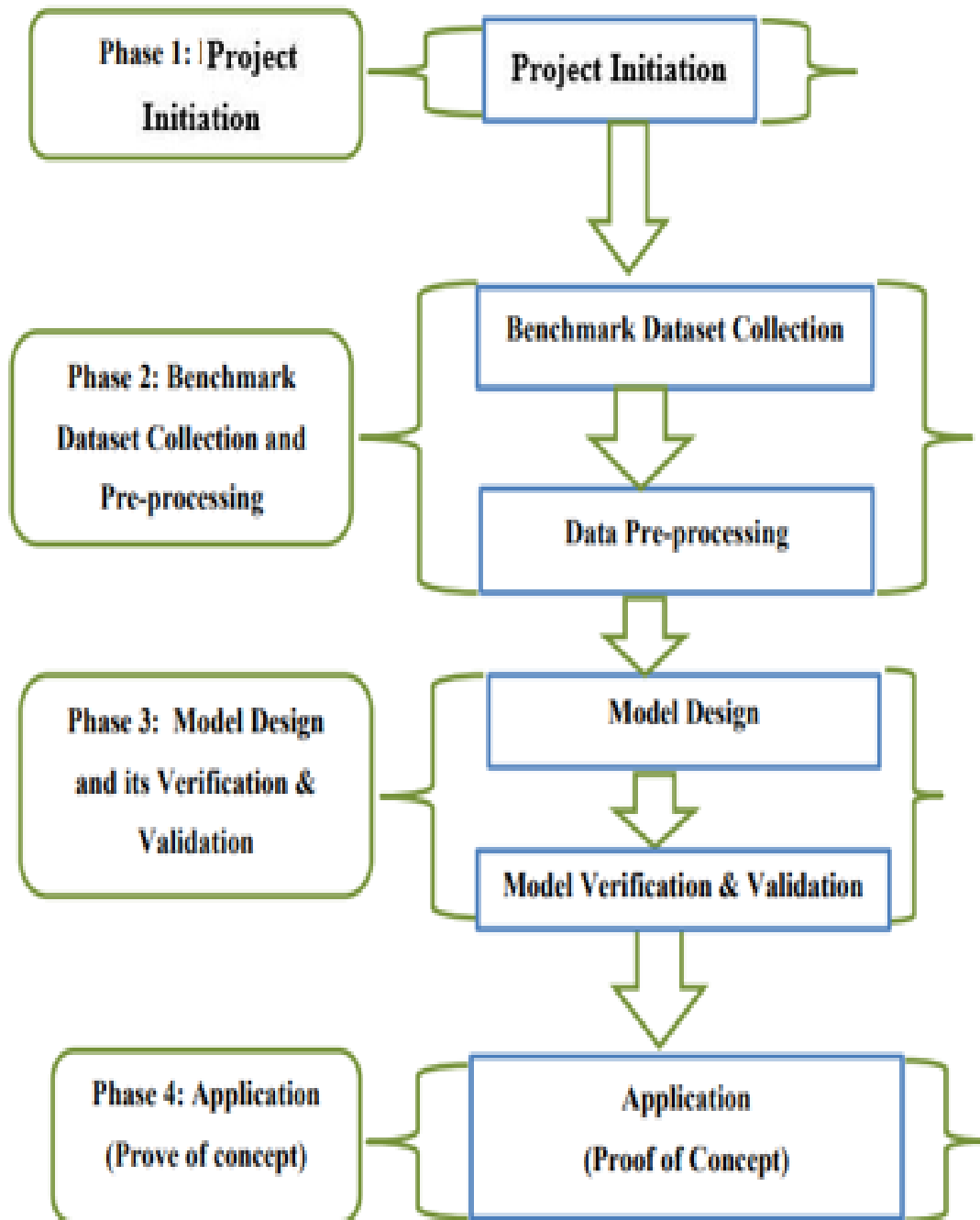


Figure 3.1: Methodology Diagram

### **3.1 Phase 1: Project Initiation**

The first phase of Methodology is Project Initiation. In which Problem statement highlights identification of problem. It states the problem our system is trying to solve. Multimodal object perception and action description using multimodal methods of HAR are detected either by traditional complex machine learning approaches involving manual engineering or by automated deep learning approaches which use high computing resources to perform well. A lightweight deep learning-assisted approach is desirable to solve the challenges of multiple variations in viewpoint for significant features in HAR. A lightweight deep learning-assisted framework for activity recognition.

### **3.2 Phase 2: Benchmark Dataset Collection and Pre-processing**

Second phase of project is about Data Collection and it's Pre-processing. Data is essential for every kind of system. Computer and digital technologies computes, calculates and operate on the basis of data. In this digital age many jobs are automated to achieve fast and accurate results or outputs. Machine learning and deep learning algorithms and techniques are used to automate many jobs or tasks such as detection of objects from images or real time video to save time and money by providing required results with fast speed. These algorithms and techniques need handsome amount of data to learn information. So, it can be used to solve the problem. These algorithms and techniques require specific dataset to solve the specific problem. That is why, data set of human activities is needed for the HARS. The second phase has two sub phases i.e., data collection and data pre-processing. For this project UCF-Sports Dataset [19] is used so that model could be trained on it.

### 3.3 Phase 3: Model Design and Model Evaluation

The third phase has two sub phases i.e., Model Design and its Verification and Validation. For this thesis images UCF Sports Dataset [19] were used so that model could be trained on it.

#### 3.3.1 Model Design

Neural Network which is used for model design is MobileNet (a pre trained convolutional neural network). MobileNetV2, there are two types of blocks. One is residual block with stride of 1. Another one is block with stride of 2 for downsizing.

- There are 3 layers for both types of blocks.
- This time, the first layer is  $1 \times 1$  convolution with ReLU6.
- The second layer is the depthwise convolution.
- The third layer is another  $1 \times 1$  convolution but without any non-linearity. It is claimed that if ReLU is used again, the deep networks only have the power of a linear classifier on the non-zero volume part of the output domain

As MobileNet is used as functional layer in this model input size required for the model which is (224, 224, 3). 224 is length and width. 3 is image type. RGB is the colour format for the images.

To design the model Python programming language 3.9, *Keras*, *Tensor flow API*, *Google Colab* has been used. Model structure has been written in python programming language. Keras is a neural network library developed in python while TensorFlow is an open-source library. Keras offer simple high-level APIs. It reduces the cognitive stress of users by following best practices to build and train neural models. TensorFlow offers both high-level and low-level APIs. TensorFlow is flexible as its eager execution allows for immediate iteration along with intuitive debugging. Both frameworks thus provide high-level APIs for building and training models with ease. Model design needs good computation sources. Google Colab is platform which provide free limited sources to perform machine learning operations, model design and many more. These sources include GPU, TPU, python Jupyter notebook environment and cloud storage in the form of google drive.



Keras is used to import the MobileNet model along with other layers to make the model more effective. There are two methods to implement convolutional neural network architecture using Keras.

- Sequential
- Functional

Functional API of Keras allows you to define a model when layers are linked to more than one layers (layers are previous layers and next layers) in the model. The functional API is able handle models with non-linear topology, shared layers, and multiple inputs or outputs. Sequential API is easy to implement as it allows to build a stack of layers (building model layer by layer). Sequential basically groups a linear stack of layers into *tf.keras.model*. Sequential also provide training and inference features on the model. Model type is sequential as it is fulfilling the need to design a model for the development of the system

A neural network model is defined by its layers as it defines the topology of the model. Number of layers varies for according to the requirement of the model. The model which is used in the development of the system consist of seven layers. There are 4 types of the layers involved in the building model architecture:

1. Functional Layer
2. Pooling Layer
3. Dense Layer
4. Dropout layer

### **3.3.2 Model Verification and Validation**

Model Verification and Validation is sub phase in third phase of methodology. In it training, testing and validation of dataset is done

### **3.3.3 Training Dataset**

UCF SPORTS Dataset [19] is used so that model can learn required information from the dataset for this purpose, dataset is split into training and testing. To perform training 80% of the data is used to train the model.

The training technique is often carried out by employing mini-batch gradient descent (based on back-propagation) with momentum to optimise the multinomial logistic regression goal. Three arguments (weights, include top, input shape) were passed to the constructor.

MobileNet has a feature called Fast Feature Extraction. *ImageDataGenerator* instances are used to extract photos as NumPy arrays, as well as their labels. The predict technique of the model is used to extract the features from these photos. A validation accuracy of roughly 95% is achieved here, which is significantly better than a small model trained from scratch. However, despite utilising a high rate of dropout, the plots show that overfitting occurs nearly immediately. This is due to the fact that this method does not employ data augmentation, which is critical for avoiding overfitting.

MobileNet has a huge number of parameters: 3.2 Million. On top of that, there are 300 million parameters in the classifier. Before building and training the model, it's critical to freeze the convolutional basis. The weights of a layer or set of layers cannot be frozen during training. If this is not done, the convolutional base's previously learnt representations will be updated during training, which is undesirable. Because the Dense layers on top are randomly initialised, very significant weight modifications would propagate across the network, effectively destroying previously learnt representations. A network can be frozen in Keras by setting the trainable attribute to False.

### **3.3.4 Testing Data**

Testing is very important for the model because it gives you detail about the performance of the model so far. Basically, model is evaluated with testing dataset. When training of the model is completed the other 20% of the data is used to test the model. For testing accuracy is used as a metric to test the trained model.

### **3.3.5 Validation**

Cross-validation is a technique for evaluating and testing the performance of a machine learning model (or accuracy). It entails conserving a portion of a dataset for which the model has not been trained. The model is then evaluated on this sample to see how well it works. Cross-validation is a technique for preventing overfitting in a model, especially when the quantity of data supplied is restricted. To predict the outcomes, our model was verified using

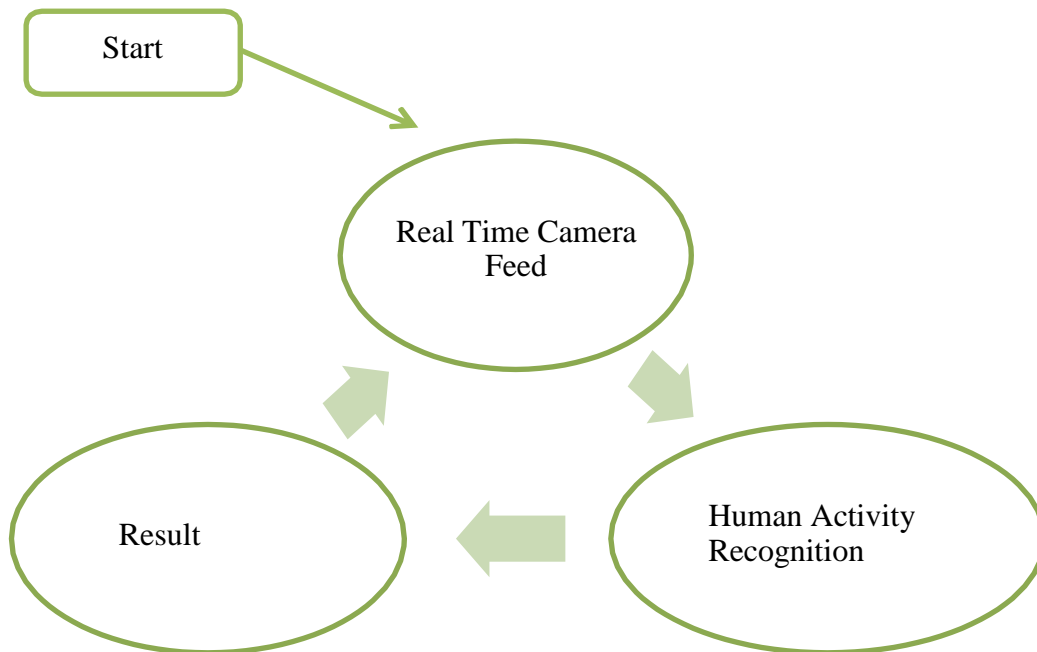
k-fold cross validation, where  $k = 5$ . Using 5-fold cross validation, it was discovered that the model produces positive results

### **3.4 Phase 4: Application (Proof of concept)**

Interface for Admin is developed using Python Language. Most of the Graphic representation is coded using Kivy framework of Python. Sign up, Login, View Live Feed, and Result is coded to perform the desired functionalities. Sign Up and Login uses CSV File to verify the authentic user. Model that has been exported is integrated in the application. Images are fed to model as input and activity is detected on them as output. System sends notification in case of detection.

### **3.5 Workflow of Application**

Human Activities are recognised by system when it receives a frame from live feed from a camera and passes that frame to model that has been trained to perform detection. Notification will popup when malicious activity is detected. The figure 3.2 describes the overall workflow of recognition system.



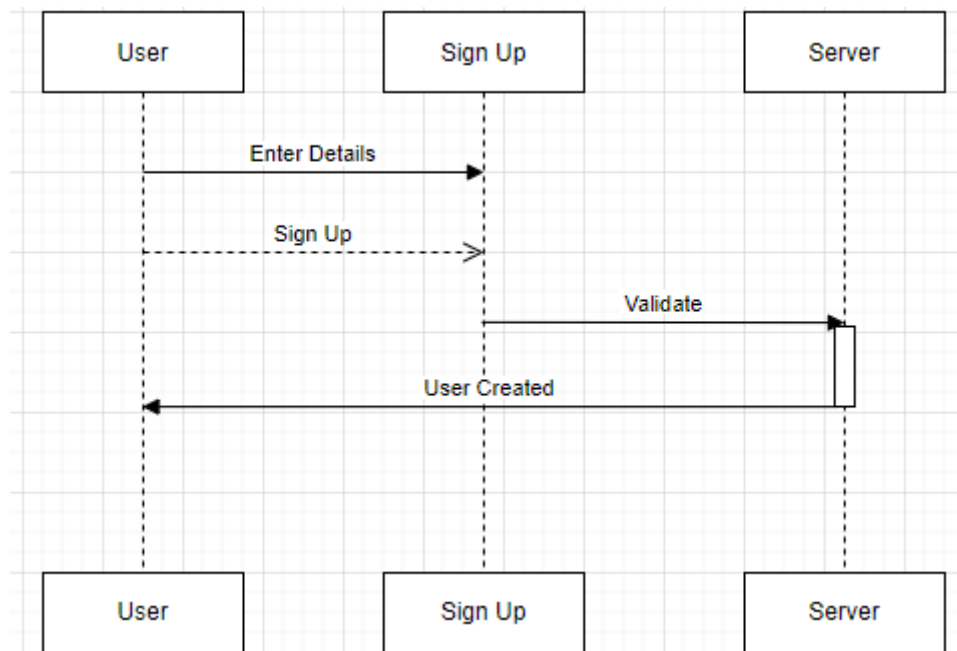
**Figure 3.2: Work Flow Diagram**

### **3.6 Sequence Diagrams**

The actions that can be performed by user are listed above. Here is a detailed version of it that how can user perform those operations on system. This is done by Sequence diagrams. Sequence diagrams of system are:

### 3.6.1 Sign Up

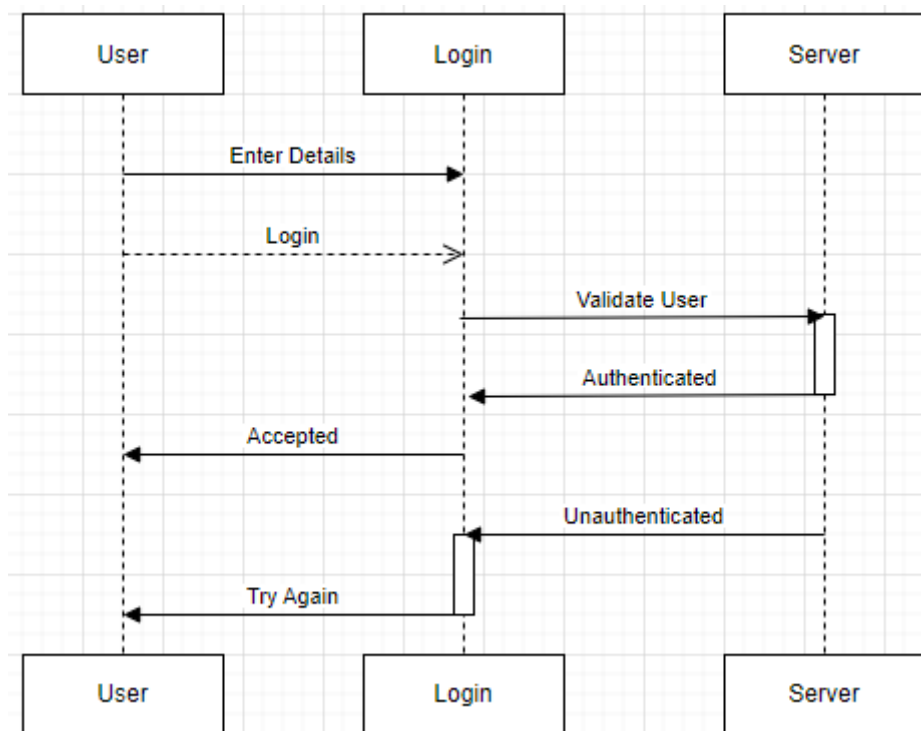
If a new user is to be entered into the system, they must enter the signup details in on the signup screen and press signup button, the system would validate the details entered and then sends the data to the server side and user created notification is generated and shown to the invigilator, if wrong details are entered, the system would then generate an error message as shown in figure 3.3:



**Figure 3.3: Signup sequence diagram**

### 3.6.2 Login

If user is to login into the system, they must enter their login details in on the login screen and press login button, the system would validate the details entered and then sends the data to the server side and user authenticated notification is generated and shown to the invigilator, if wrong details are entered, the system would then generate an error message.



**Figure 3.4: Login sequence diagram**

### 3.6.3 View Result

If the user wants to look into details of all the recent Activity, he clicks the View Capture button after Signing in. Result will be displayed on the screen after image being processed by the model. It is shown in figure 3.5

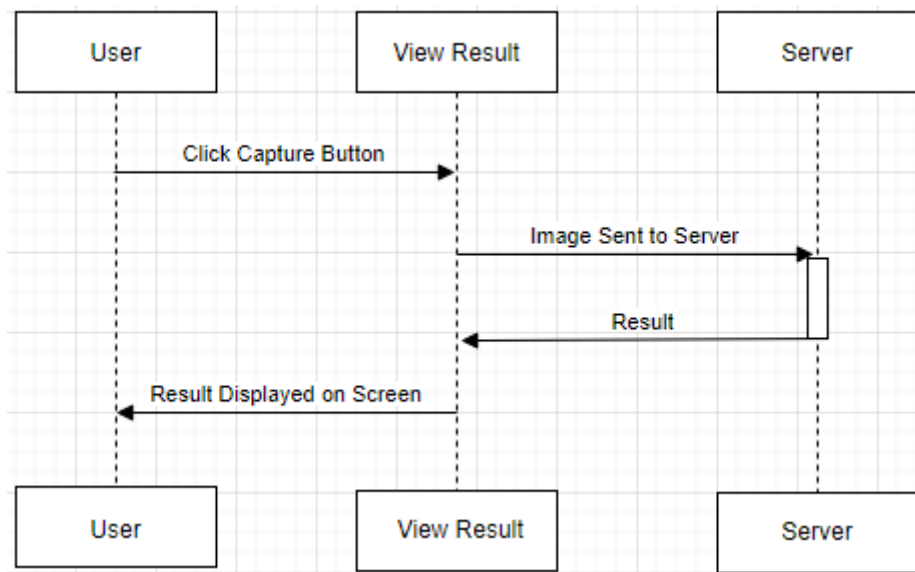
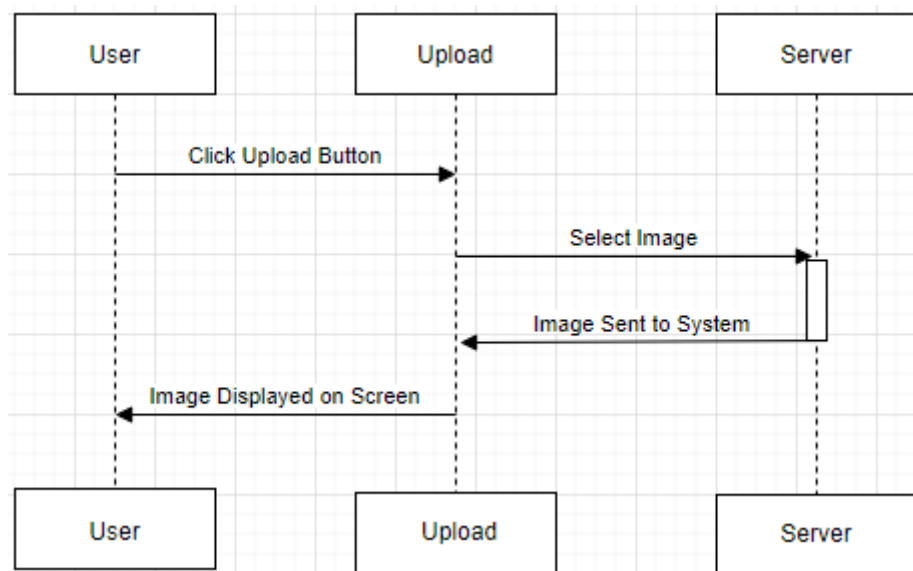


Figure 3.5: View Result sequence diagram

### 3.6.4 Upload

If the user wants to find the activity of a specific picture or frame, he clicks on upload button on the main screen, after that popup will appear on screen where he can upload a picture, after selecting the picture result will be shown recognizing the activity in the picture. It is shown in Figure 3.6

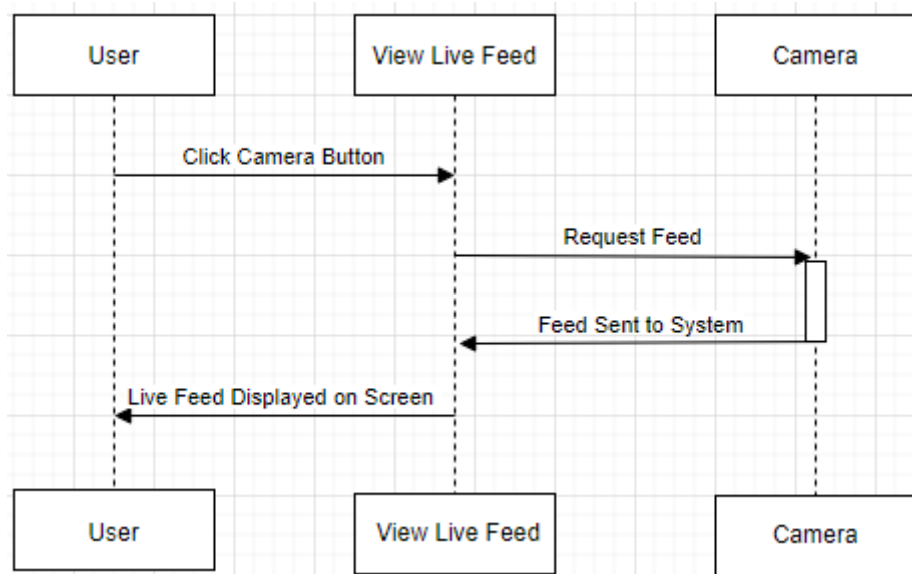


**Figure 3.6: Upload sequence diagram**



### 3.6.5 Live Feed

If the user wants to look into live feed from camera, he just needs to clicks the Play button to View Live Feed after Signing in. Live Feed will be displayed on Screen. It is shown in Figure 3.7



**Figure 3.7: View Live Feed sequence diagram**

## CHAPTER 4

### EXPERIMENT AND IMPLEMENTATION

In this section experiments are discussed which are performed on dataset. Dataset is used to train the neural network model (Inception, Xception and MobileNet) using machine learning and deep learning techniques. Experiments for the system was very vital as it helps us to find optimal conditions where neural network model (Inception, Xception, MobileNet) have shown highest accuracy.

#### 4.1 Experimental Setup

Google Colab is used to do experiments on our selected neural network (Inception, Xception and MobileNet). Python language is used to write code for experimentation. Gmail authentication is required for uploading the data on google Colab for training. After uploading dataset, Keras, TensorFlow, OpenCV, matplotlib, sklearn are the libraries that are imported to train, test and validate the model by performing experiments on the dataset using hyper-parameter tuning.

##### 4.1.1 Google Colab

Google Collaboratory (Colab) is a free Jupyter notebook environment running completely in the cloud. Amazing thing about Colab is that it does not require setup, In Colab files that you have created are termed as notebooks. Colab gives its users a collaboration feature in which your created notebooks can be simultaneously edited

by your team members same as you edit documents in Google Docs. Main and the greatest feature of Colab is that it supports almost all popular machine learning libraries without the need to download extensions. These libraries are easily loaded in your Colab notebook

### **Developer Features**

As a developer, you can perform the tasks mentioned below using Colab and Kaggle

- Write code in Python and execute it
- Notebooks can be uploaded, created and shared
- Notebook can be saved in Google Drive
- Notebook from GitHub can be imported and published
- Import datasets in Notebook
- Integrate PyTorch, TensorFlow, Keras, OpenCV
- Free Cloud storage service with free GPU and TPU

Gmail account is must to use Google Collaboratory. Otherwise, most of the Google Collaboratory features would not work.

### **Why Colab?**

Reasons to choose Google Colab to do training and testing on neural network model (MobileNet) are stated below:

- Pre-installed Libraries
- Free Cloud storage
- Collaboration with teammates
- Free GPU and TPU for machine learning or deep learning

### 4.1.2 Hyper Parameter Tuning

Hyper-parameter tuning is performed to find the best values of the hyper-parameter to design the model with the best accuracy. Hyper-parameters that are used for fine tuning a neural network model are Activation function, Optimizer, Learning rate, Batch Size and Epochs. Epochs are set to 100 for every batch size and learning rate. Batch sizes 8, 12, 16 and 20, 32, 64 are used in Experimentation along with learning rate of 0.001, 0.0001 and 0.00025 for model. Activation Function used was SoftMax and Optimizer used in model was Adam. Activation Function and Optimizer used remains constant throughout process of fine tuning. Details of hyperparameters can be seen in Table 4.1

**Table 4.1: Hyperparameters of Model**

<b>Hyperparameters</b>				
<b>Activation Function</b>	<b>Optimizer</b>	<b>Batch Size</b>	<b>Learning Rate</b>	<b>Epochs</b>
SoftMax	Adam	16,20,32,64	0.001,0.0001,0.00025	100

Hyper-parameters which are considered for the tuning the model (Inception, Xception, MobileNet) are Epochs, Batch size, and Learning rate. These hyper-parameters are fine-tuned on model to get insight on performance of model. Values of hyper-parameters on which model performed best are recorded and will be used for application later. Epochs, Batch Size and Learning Rate are defined in sections below

#### **Epoch**

One epoch means to train the neural network using all the training data in one cycle. In an epoch, all of the data is used exactly once. It consists of two passes a forward pass and a backward pass which together counted as one pass. Epoch sizes depends upon the size of dataset thus varies accordingly. Different epoch sizes are used to find desired better results. Epoch size of 100 is used in the development of the system.

### **Batch Size**

Number of samples processed prior to the update in model is batch size. The size of a batch must be greater than 1 and less than the number of samples in the training dataset. Batch sizes of 8, 12, 16 and 20, 32, 64 are used in experimentation.

### **Learning Rate**

Learning rate is also a hyper parameter used to fine tune model. It controls how much change is shown by the model in response to the estimated error each time the weights of the model are updated. Learning rate is thought to be the most important hyper-parameter when configuring your neural network. It is important to know and evaluate the effects of the learning rate on the performance of the model to build an awareness about the changing aspects of the learning rate on the behavior of model. Learning rate of 0.001, 0.0001 and 0.00025 are used for experimentation.

## **4.2 List of Experiment on Models**

Following are the list of done to obtain best accuracy for the model. Description about experiments related where experiments were done, images involved, platform used, problem which is handled by experiment, and number of epochs used are described below:

Location: Home

Dataset: UCFSPORTS [19]

Total Images: 9118

Equipment: Laptop

Platform: Google Colaboratory

Problem: Multiclass

Epochs: 100 for each 64 batch size and 0.0001 learning rate

### 4.2.1 Experiment on Inception

Table 4.2: Experiments on Inception

Hyper-parameters			Model Evaluation	
Image Size	Batch size	Learning rate	Accuracy	Validation Accuracy
75x75	64	0.001	0.975	0.999
90x90	64	0.001	0.856	0.758
120x120	64	0.001	0.996	0.988
150x150	64	0.001	0.993	0.77
180x180	64	0.001	0.993	0.612

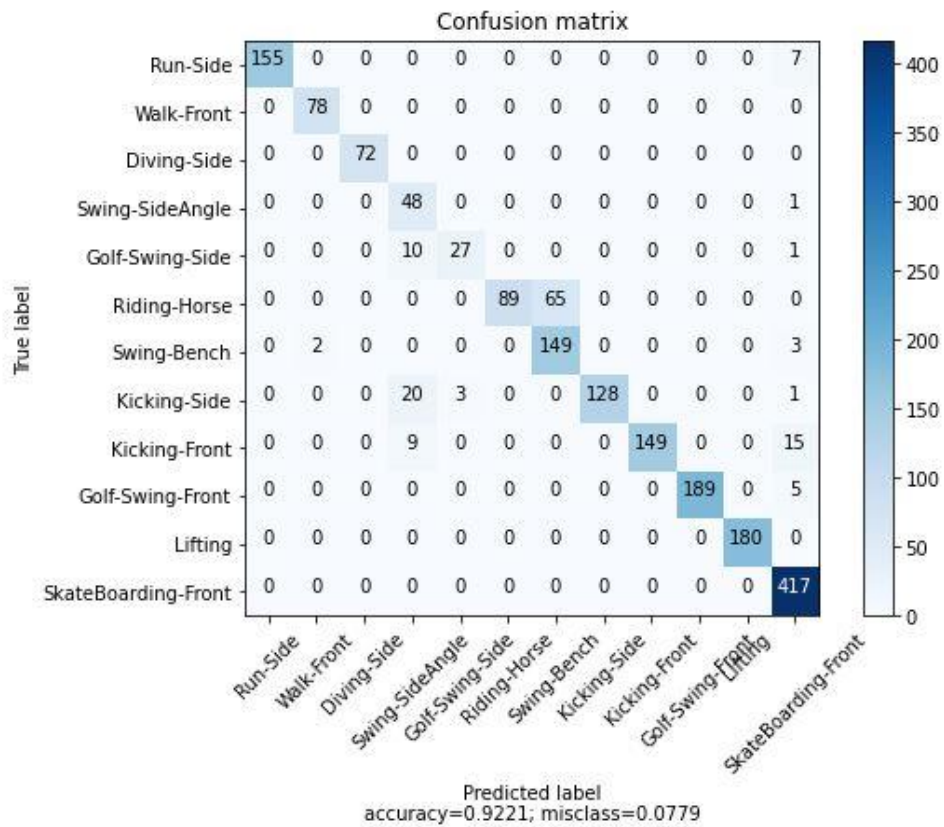
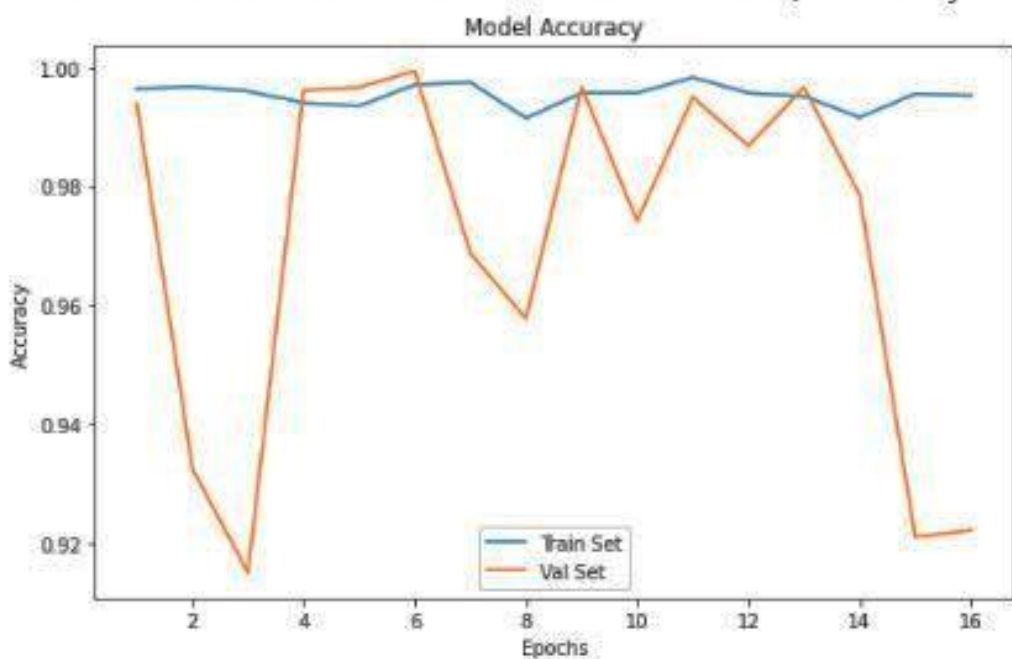


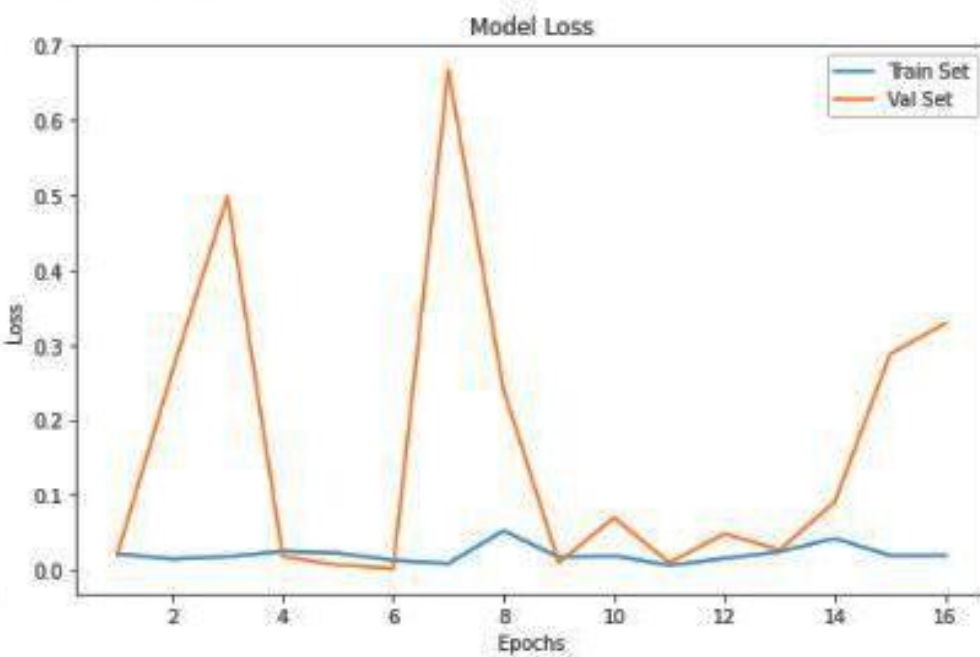
Figure 4.1: Confusion Matrix of Inception

Score for fold 5: loss of 0.3291342258453369; accuracy of



(a)

92.21064448356628%



(b)

Figure 4.2: Accuracy Loss Graph of Inception

## 4.2.2 Experiment on MobileNet

Table 4.3: Experiments on MobileNet

Hyper-parameters			Model Evaluation	
Image Size	Batch size	Learning rate	Accuracy	Validation Accuracy
75x75	64	0.001	0.991	0.971
90x90	64	0.001	0.995	0.940
120x120	64	0.001	0.996	0.806
150x150	64	0.001	0.992	0.656
180x180	64	0.001	0.993	0.210

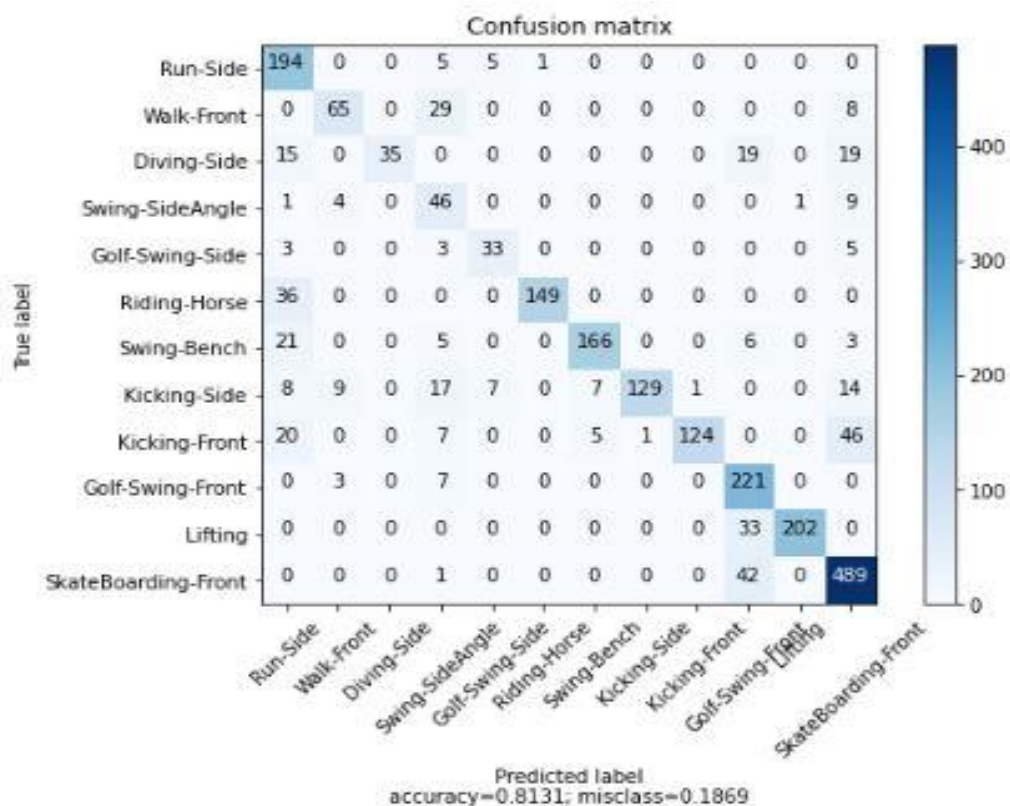
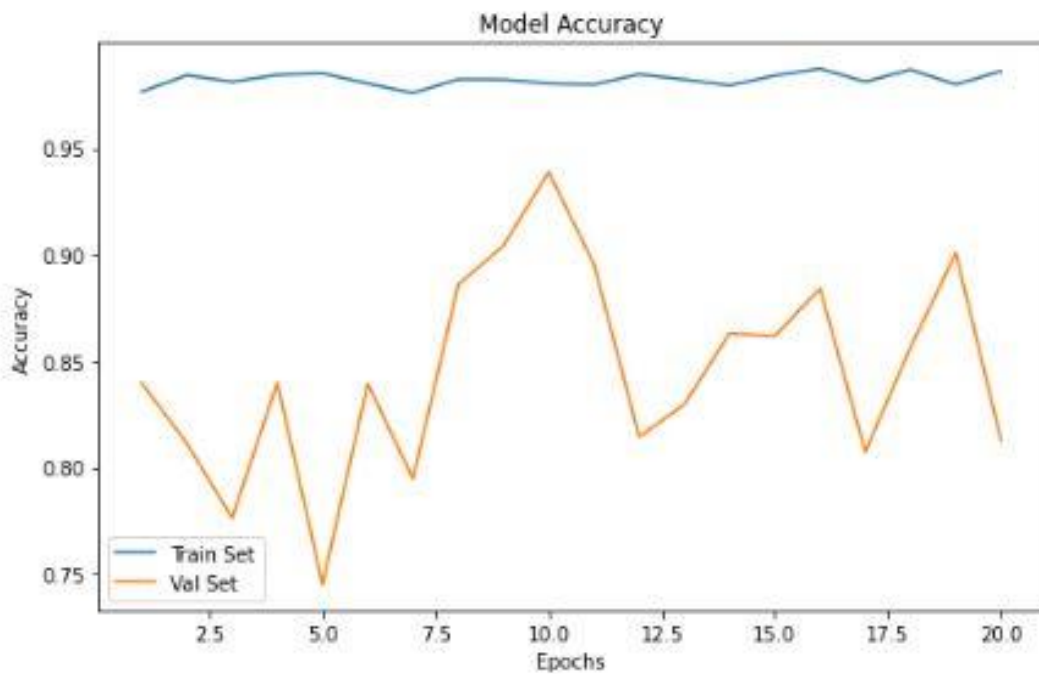
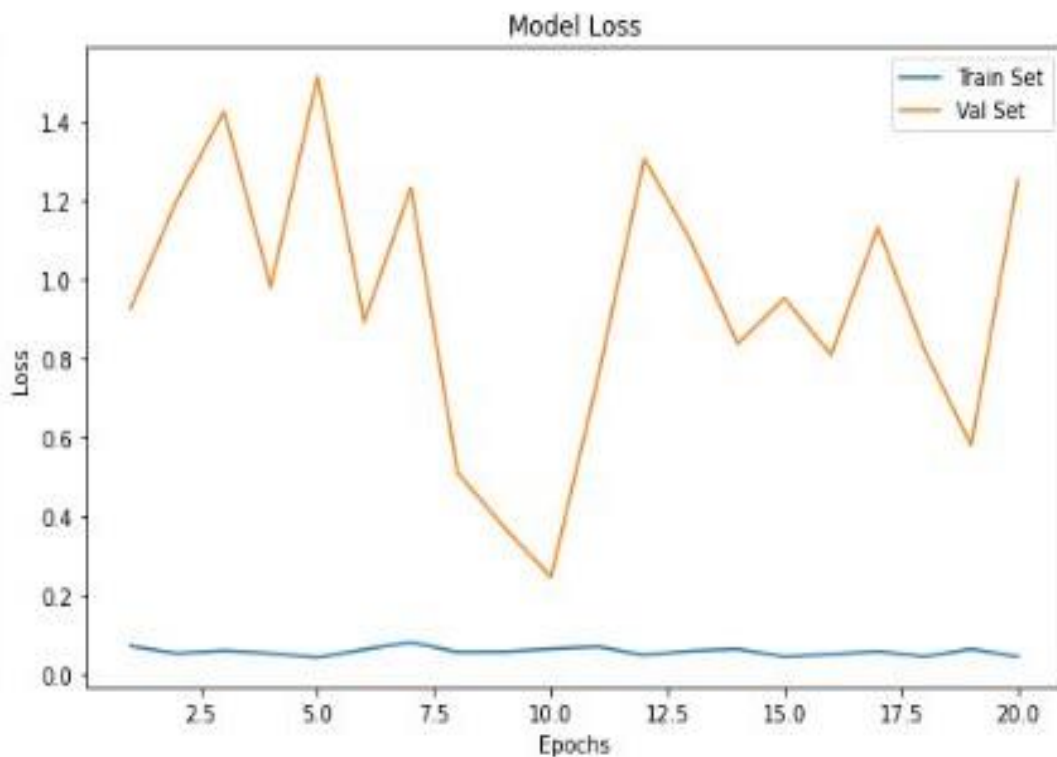


Figure 4.3: Confusion Matrix of MobileNet





(a)



(b)

**Figure 4.4: Accuracy Loss Graph of MobileNet**

### 4.2.3 Experiment on Xception

Table 4.4: Experiments on Xception

Hyper-parameters			Model Evaluation	
Image Size	Batch size	Learning rate	Accuracy	Validation Accuracy
75x75	64	0.001	0.992	0.997
90x90	64	0.001	0.989	0.966
120x120	64	0.001	0.994	0.989
150x150	64	0.001	0.984	0.891
180x180	64	0.001	0.981	0.715

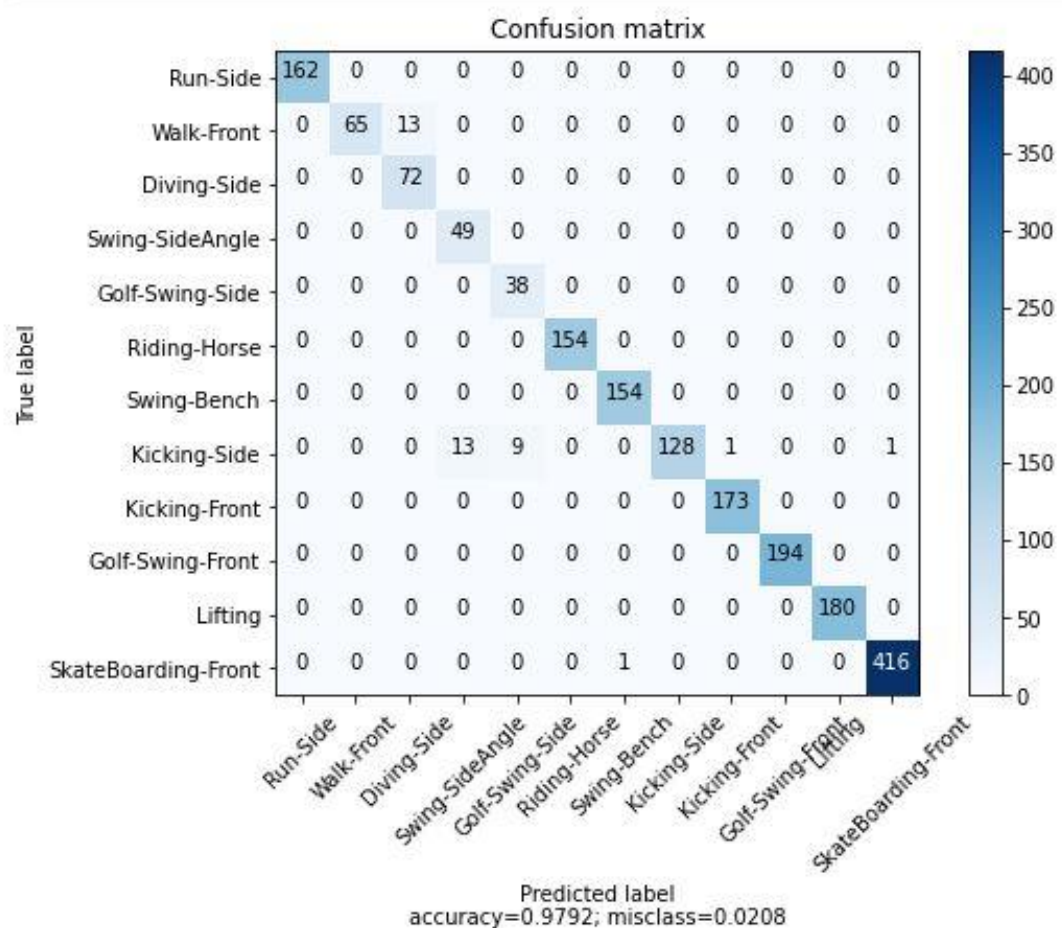
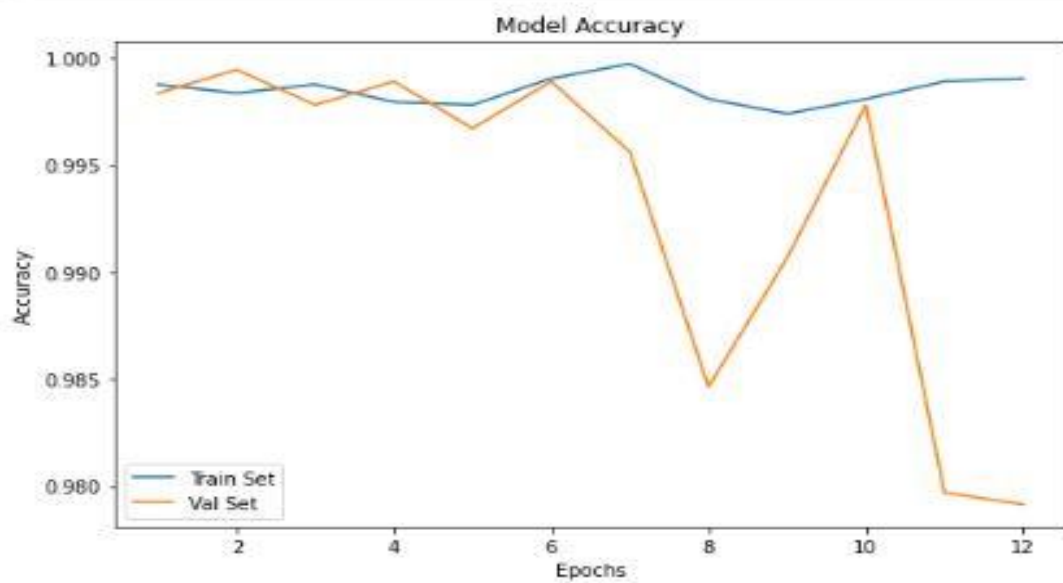
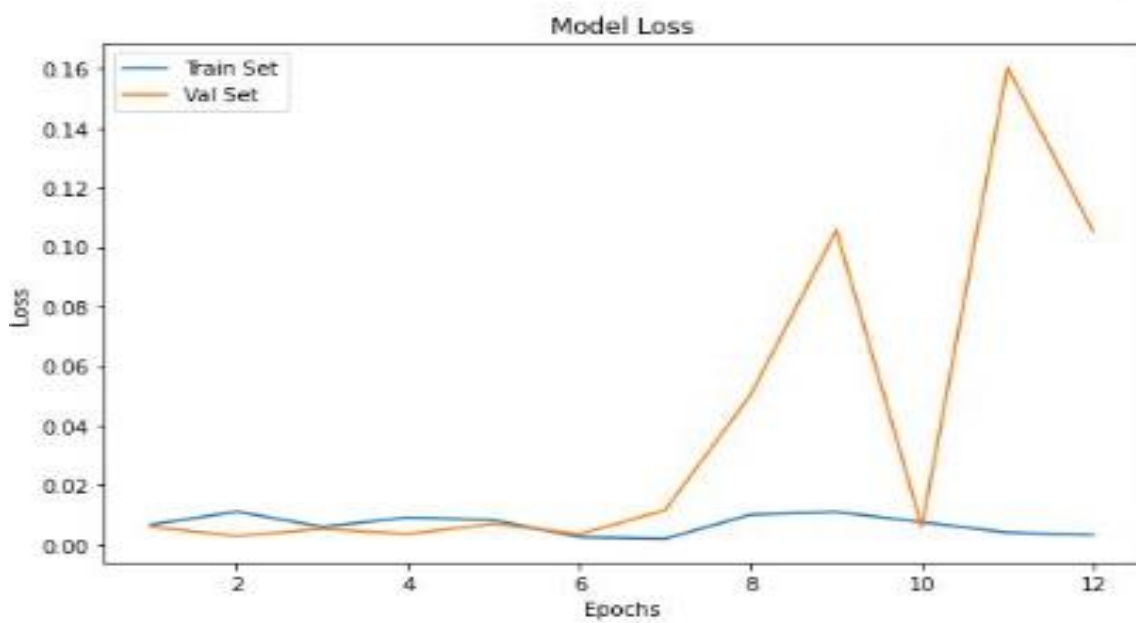


Figure 4.5: Confusion Matrix of Xception



(a)



(b)

**Figure 4.6: Accuracy Loss Graph of Xception**

## 4.2.4 Comparison of Models

Table 4.5: Comparison of Models

Sr. No	Model name	Validation method	Classes	Epochs	Dimensions	Training Accuracy	Validation Accuracy	Training Loss	Validation Loss
1	Inception	Train/Test /Split	3	50	150x150	99.36%	77%	0.0324	1.4501
					180x180	99.36%	61.27%	0.0234	3.9486
		KFold	13	100	90x90	85.65%	75.88	0.4398	0.8381
					120x120	99.66%	98.82%	0.0212	0.0476
				100 x 5 folds	75x75	99.37%	99.96%	0.0181	0.105
					90x90	99.74%	92.21%	0.0195	0.3291
2	Xception	Train/Test /Split	3	50	150x150	99.27%	89.12%	0.0353	0.7589
					180x180	99.37%	71.54%	0.0280	2.3106
		KFold	13	100	90x90	99.56%	96.67%	0.0256	0.0256
					120x120	99.68%	98.99%	0.0212	0.0474
				100 x 5 folds	75x75	99.96%	99.78%	0.0022	0.0091
					90x90	99.90%	97.92%	0.0033	0.1053
3	MobileNet	Train/Test /Split	3	50	150x150	98.44%	65.61%	0.0717	2.5221
					180x180	98.16%	21%	0.0822	23.2987
		KFold	13	100	90x90	98.99%	94.04%	0.0488	0.3927
					120x120	99.46%	80.66%	0.0216	2.9432
				100 x 5 folds	75x75	99.84%	97.15%	0.0126	0.1753
					90x90	99.04%	82.67%	0.0397	0.9486

## **4.3 Languages**

### **4.3.1 Python**

Python is a general-purpose high-level programming language. Its design philosophy emphasizes code readability with the use of significant indentation. Python is dynamically-typed and garbage-collected. It supports multiple programming paradigms, including object-oriented, functional and structured programming. It has comprehensive standard libraries.

### **4.3.2 Framework**

Kivy [19] is an Open-source, cross-platform framework for developing a native mobile application. Kivy allows us to build a mobile application for Android and IOS with a single code-based and programming language called Python. The Kivy framework contains Natural User Interface.

## **4.4 Tools**

### **4.4.1 Vs Code**

Microsoft Visual Studio Code is a standalone source code editor that runs on multiple operating systems. It is used to develop computer programs, as well as websites, web apps, web services, and mobile apps. It supports many programming languages and a set of features that differs per language. VS Code provides initial support including syntax highlighting etc.

#### **4.4.2 Emulator**

The emulator provides simulations of android devices on your computer which helps us to test our application on a variety of devices without needing to have each physical device. It came along with a predefined configuration for multiple android-based Phones, Tablets. Testing in an emulator is way faster than physical devices.

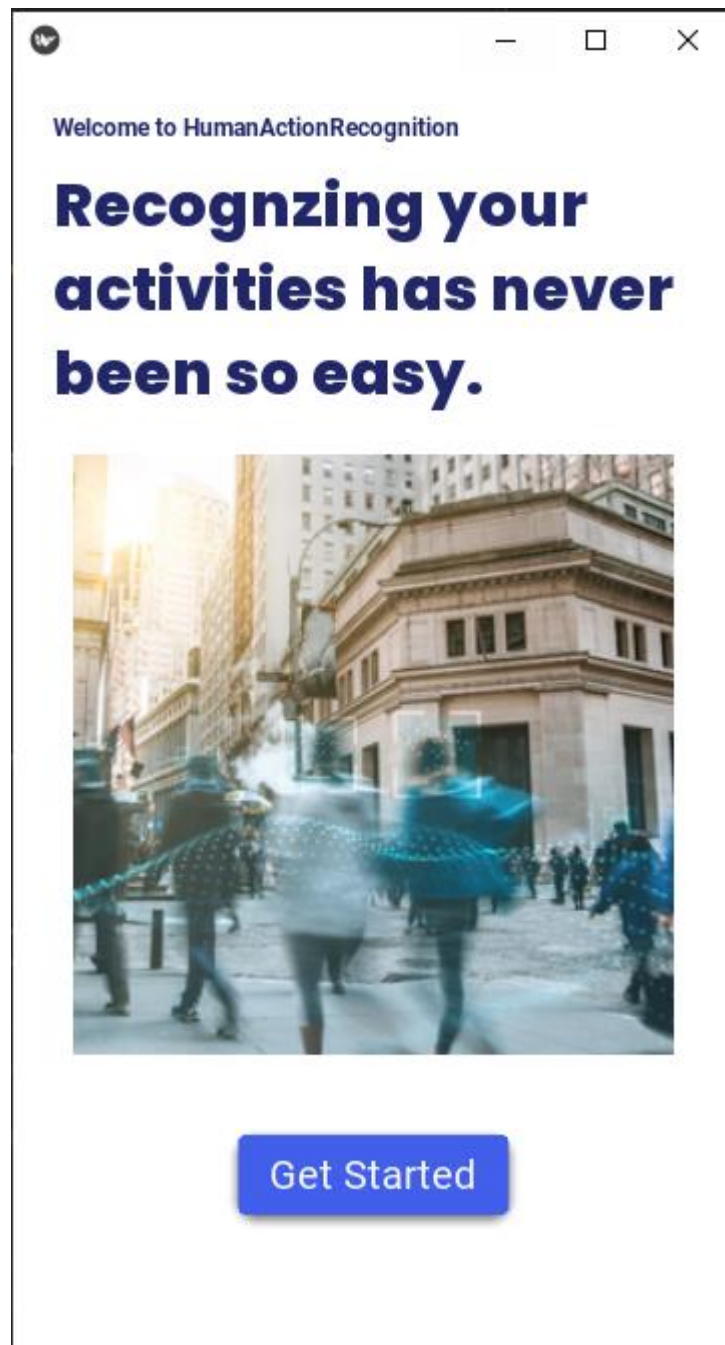
## **CHAPTER 5**

### **RESULTS AND DISCUSSIONS**

#### **5.1 Result of Application**

##### **5.1.1 Introduction**

This is the introduction of the application which tells the main features of the HAR app and this will show only once at the user downloads the application and runs it for the first time.

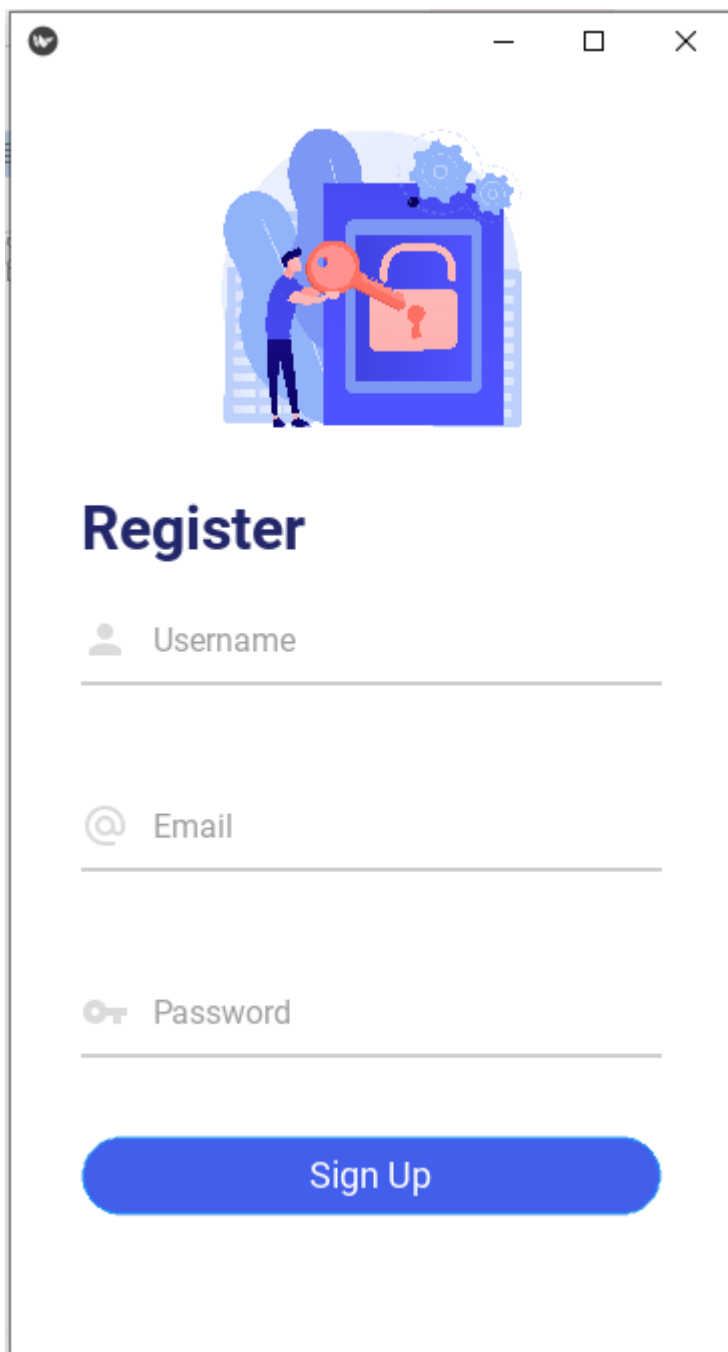


**Figure 5.1: Welcome Screen**



### 5.1.2 Log In and Sign Up

The Log in and sign up is simple as other mobile applications with the additional features of email verification which will support the confidentiality triad.



**Register**

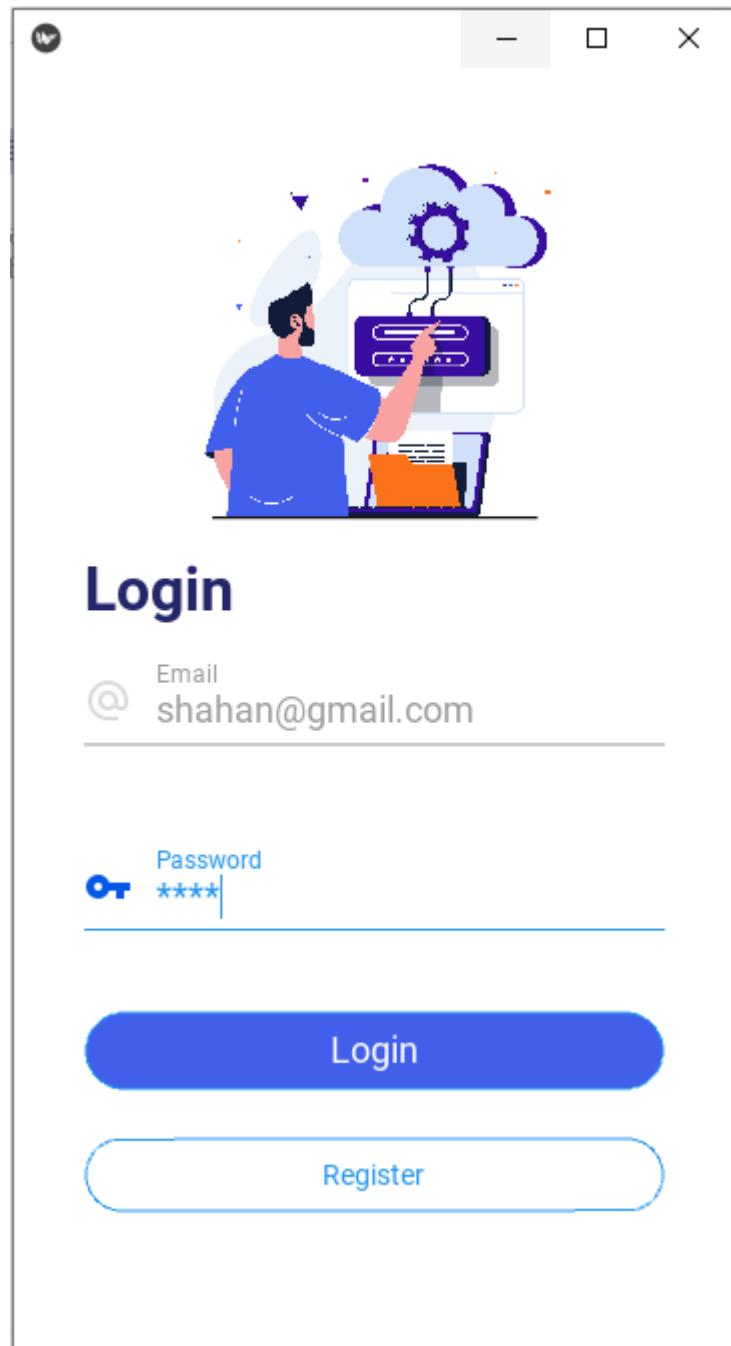
Username

Email

Password

Sign Up

**Figure 5.2: Sign up screen**

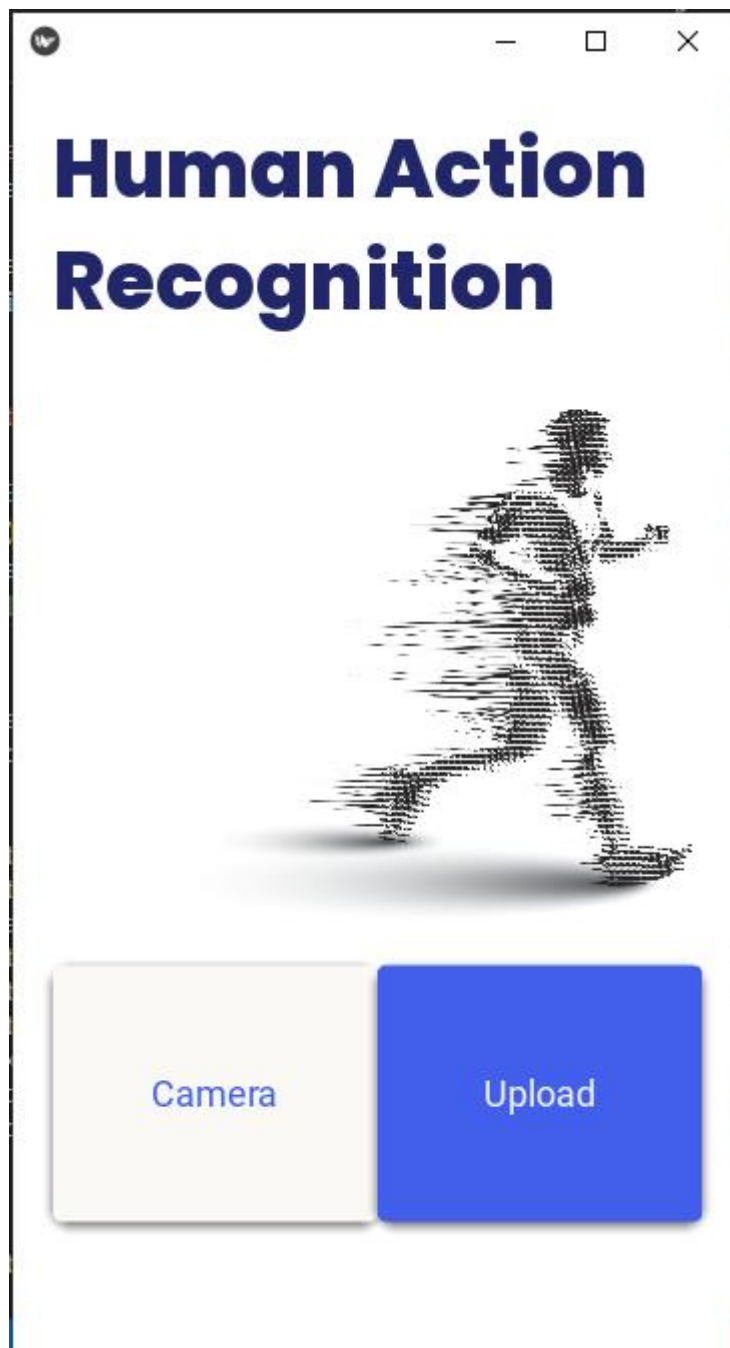


**Figure 5.3: Log in Screen**

### **5.1.3 Main Screen**

After Sign in as User, they will be directed towards this screen which has the following feature:

- Camera
- Upload



**Figure 5.4: Main screen**

### 5.1.4 Camera Screen

By clicking on the Camera button, camera screen will appear where we have live feed which will appear by using camera and capture button by which picture will get feuded to the model and result will be displayed after processing on the screen.



Figure 5.5: Camera Screen

### 5.1.5 Upload Screen

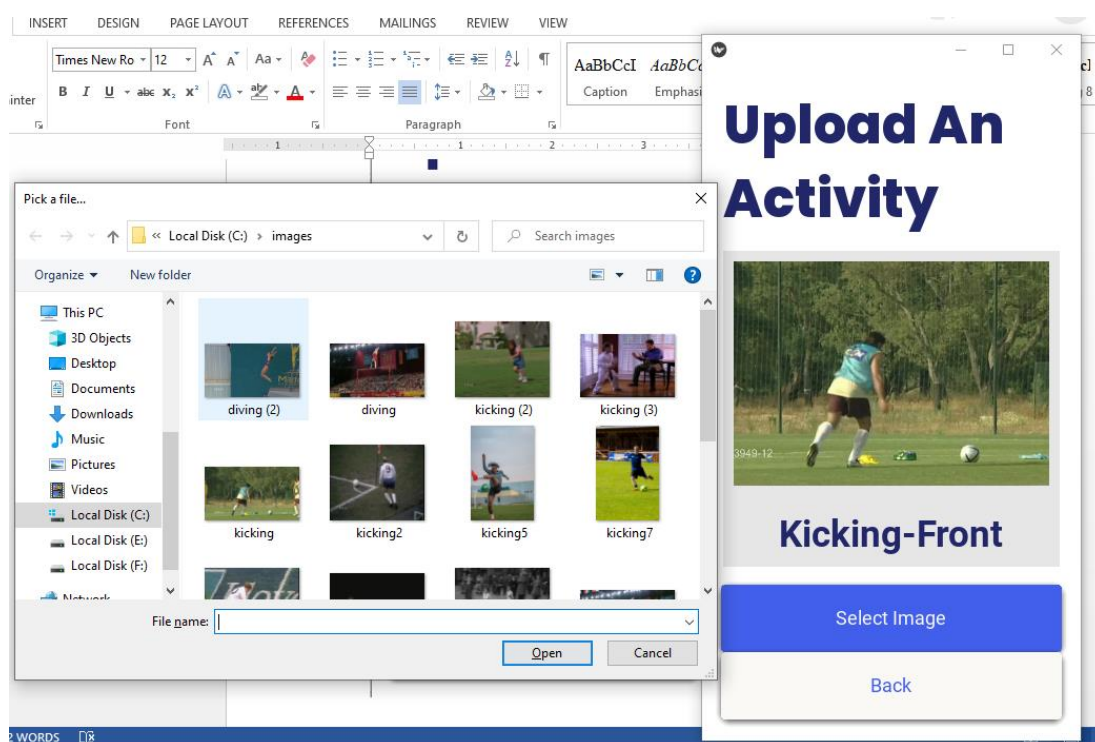
User can upload a picture by clicking on the upload button. Popup will appear where user can select the folder and the specific picture.



Figure 5.6: Upload Screen

## 5.2 Result of Model

Figure 5.7 show that when user clicks on select image button popup will appear by which use can select the image and after selection the model will display the result with the image on the screen.



**Figure 5.7: Uploading Image**

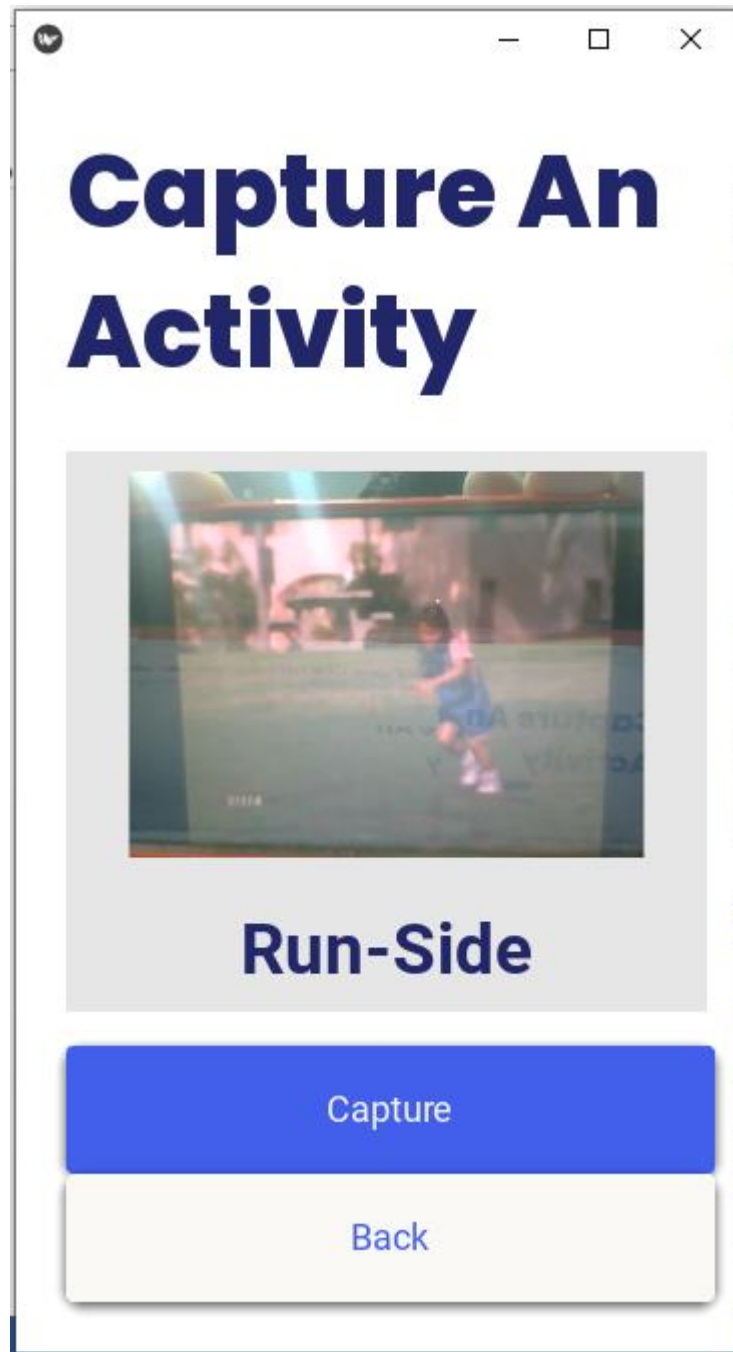
Figure 5.8 show the result with the image selected by user and the result of selected the image on the screen.



**Figure 5.8: Result after uploading image**



Figure 5.9 show the live feed and the result with the image captured when captured button is pressed by user on the screen.



**Figure 5.9: Result from live feed**

## CHAPTER 6

### CONCLUSION AND RECOMMENDATIONS

#### 6.1 Project Achievements

Form the verification of plan and after testing the application, the actual output of the HAR App is getting good result as manage to reach all the expected output and fulfilled the project objectives. In the end, HAR App manages to help the user by providing them a platform that will recognize human activities from live feed and alert user on suspicious activities. Furthermore, HAR App counters the factor which requires individual's physical presence in front of a screen to check the activities. By HAR App users will get an alert without the need of a constant watch.

#### 6.2 Future Work

Some improvements can be made in the future HARS is a detection system which uses machine learning procedure for activity detection. As it is designed and train to learn human activities. These kinds of system perform their operation based on dataset. In future, to enhance HARS as a recognition system a dataset will be generated in near future. Dataset will contain new images with better quality that will be produced by using tools and methods required to generate better quality benchmark dataset. Model will be trained on that dataset to make it useable for commercial purpose as well. Current system only works with a single camera. In the

future HARS will be enhanced by making it work on multiple Camera so that detection could be done at multiple places at the same time.

### **6.3 Implementation Issues and Challenges**

In this project, we encounter several challenges throughout the project life cycle. First and foremost, the selection of dataset and model is a major issue. There are number of datasets available on Human Activities, most of them are decade old and resolution is low and the size of dataset is not big enough to train model properly. Selecting the best transfer learning model is also a big challenge as there are many models available and each one has its own pros and cons. Training of the model is the biggest challenge we have encountered as model training requires lots of resources as dataset and pre trained models are themselves quite heavy and in order to train them again more resources are required and personal PCs cannot bear that much of a load. Online platforms such as Google Colab and Kaggle are better but gives limited resources so in order to train the model we have to change the structure of the model to work within the resources available.

On the other hand, development tools and environment need to be determined before the project can proceed. There are many much-integrated development environments such as Visual Studio Code, PyCharm, and Jupyter Notebook which can be used for the development of HAR App. IDEs like PyCharm require extra plugins to integrate with the system while developing a mobile application. From Jupyter studio and Visual Studio Code, we use both as both are the best tools moreover, they have a vast amount of developer community which helps native developers in the hour of need.

## **6.4 Conclusion**

In this modern era, most organizations are using the latest technologies and software but hopefully, through this project, we solve this long-awaited upgrade, by successfully meeting our aims and objectives. We have developed a cross-platform app that can be for both iOS and Android user which provide a light wait model integrated in a mobile app to detect human activities. We have created an app that recognizes human activities using deep learning model. This application also makes Surveillance operation easily automated.

## REFERENCES

- [1] “Deep Learning Models for Human Activity Recognition.” <https://machinelearningmastery.com/deep-learning-models-for-human-activity-recognition/> (accessed Mar. 15, 2022).
- [2] M. Gholamrezai and S. M. Taghi Almodarresi, “Human Activity Recognition Using 2D Convolutional Neural Networks,” *ICEE 2019 - 27th Iran. Conf. Electr. Eng.*, pp. 1682–1686, Apr. 2019, doi: 10.1109/IRANIANCEE.2019.8786578.
- [3] C. Feng, M. Sun, M. Dabbaghjamanesh, Y. Liu, and J. Zhang, “Advanced machine learning applications to modern power systems,” *New Technol. Power Syst. Oper. Anal.*, pp. 209–257, 2021, doi: 10.1016/B978-0-12-820168-8.00007-9.
- [4] R. Vrskova, R. Hudec, P. Kamencay, and P. Sykora, “Human Activity Classification Using the 3DCNN Architecture,” *Appl. Sci.*, vol. 12, no. 2, pp. 1–17, 2022, doi: 10.3390/app12020931.
- [5] N. Jaouedi, N. Boujnah, and M. S. Bouhleb, “A new hybrid deep learning model for human action recognition,” *J. King Saud Univ. - Comput. Inf. Sci.*, vol. 32, no. 4, pp. 447–453, May 2020, doi: 10.1016/J.JKSUCI.2019.09.004.
- [6] “Kivy: Cross-platform Python Framework for GUI apps Development.” <https://kivy.org/> (accessed Dec. 15, 2022).
- [7] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, “Rethinking the Inception Architecture for Computer Vision,” *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, vol. 2016-December, pp. 2818–2826, Dec. 2015, doi: 10.48550/arxiv.1512.00567.
- [8] “Inception V3 Model Architecture.” <https://iq.opengenus.org/inception-v3-model-architecture/> (accessed Dec. 15, 2022).
- [9] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L. C. Chen, “MobileNetV2: Inverted Residuals and Linear Bottlenecks,” *Proc. IEEE*

- Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, pp. 4510–4520, Jan. 2018, doi: 10.48550/arxiv.1801.04381.
- [10] “Review: MobileNetV2 — Light Weight Model (Image Classification) | by Sik-Ho Tsang | Towards Data Science.” <https://towardsdatascience.com/review-mobilenetv2-light-weight-model-image-classification-8febb490e61c> (accessed Dec. 15, 2022).
- [11] “Understanding Depthwise Separable Convolutions and the efficiency of MobileNets | by Arjun Sarkar | Towards Data Science.” <https://towardsdatascience.com/understanding-depthwise-separable-convolutions-and-the-efficiency-of-mobilenets-6de3d6b62503> (accessed Dec. 15, 2022).
- [12] F. Chollet, “Xception: Deep Learning with Depthwise Separable Convolutions,” *Proc. - 30th IEEE Conf. Comput. Vis. Pattern Recognition, CVPR 2017*, vol. 2017-January, pp. 1800–1807, Oct. 2016, doi: 10.48550/arxiv.1610.02357.
- [13] “Review: Xception — With Depthwise Separable Convolution, Better Than Inception-v3 (Image Classification) | by Sik-Ho Tsang | Towards Data Science.” <https://towardsdatascience.com/review-xception-with-depthwise-separable-convolution-better-than-inception-v3-image-dc967dd42568> (accessed Dec. 15, 2022).
- [14] “Visual Studio Code - Code Editing. Redefined.” <https://code.visualstudio.com/> (accessed Dec. 15, 2022).
- [15] “Welcome To Colaboratory - Colaboratory.” <https://colab.research.google.com/> (accessed Dec. 15, 2022).
- [16] “Kaggle: Your Home for Data Science.” <https://www.kaggle.com/> (accessed Dec. 15, 2022).
- [17] “Adobe Creative Cloud | Details and products | Adobe.” <https://www.adobe.com/creativecloud.html> (accessed Dec. 15, 2022).
- [18] “Industry-leading vector graphics software | Adobe Illustrator.” <https://www.adobe.com/products/illustrator.html> (accessed Dec. 15, 2022).
- [19] “CRCV | Center for Research in Computer Vision at the University of Central Florida.” [https://www.crcv.ucf.edu/data/UCF\\_Sports\\_Action.php](https://www.crcv.ucf.edu/data/UCF_Sports_Action.php) (accessed Dec. 15, 2022).
- [20] “CRCV | Center for Research in Computer Vision at the University of Central Florida.” <https://www.crcv.ucf.edu/data/UCF101.php> (accessed Dec. 15, 2022).

- [21] “Kinetics | DeepMind.” <https://deepmind.com/research/open-source/kinetics> (accessed Dec. 15, 2022).
- [22] “Serre Lab » HMDB: a large human motion database.” <https://serre-lab.clps.brown.edu/resource/hmdb-a-large-human-motion-database/> (accessed Dec. 15, 2022).
- [23] “Activity Net.” <http://activity-net.org/> (accessed Dec. 15, 2022).
- [24] “THUMOS Challenge 2014.” <http://cvc.ucf.edu/THUMOS14/> (accessed Dec. 15, 2022).
- [25] “GitHub - tejaskhot/KTH-Dataset: Experimenting with the KTH human activity recognition dataset from <http://www.nada.kth.se/cvap/actions/>.” <https://github.com/tejaskhot/KTH-Dataset> (accessed Dec. 15, 2022).
- [26] “Sports-1M Dataset | DeepAI.” <https://deepai.org/dataset/sports-1m> (accessed Dec. 15, 2022).
- [27] “YouTube-8M: A Large and Diverse Labeled Video Dataset for Video Understanding Research.” <https://research.google.com/youtube8m/> (accessed Dec. 15, 2022).
- [28] “UTKinect-Action3D Dataset.” <http://cvrc.ece.utexas.edu/KinectDatasets/HOJ3D.html> (accessed Dec. 15, 2022).
- [29] “COVE - Computer Vision Exchange.” <https://cove.thecvf.com/datasets/37> (accessed Dec. 15, 2022).
- [30] H. Zhao, A. Torralba, L. Torresani, and Z. Yan, “HACS: Human action clips and segments dataset for recognition and temporal localization,” *Proc. IEEE Int. Conf. Comput. Vis.*, vol. 2019-October, pp. 8667–8677, Oct. 2019, doi: 10.1109/ICCV.2019.00876.
- [31] “Coin.” <https://coin-dataset.github.io/> (accessed Mar. 16, 2022).
- [32] C. Liu, Y. Hu, Y. Li, S. Song, and J. Liu, “PKU-MMD: A Large Scale Benchmark for Continuous Multi-Modal Human Action Understanding,” Mar. 2017, Accessed: Mar. 16, 2022. [Online]. Available: <http://arxiv.org/abs/1703.07475>
- [33] “UT-Interaction Benchmark (Human Interaction Recognition) | Papers With Code.” <https://paperswithcode.com/sota/human-interaction-recognition-on-ut-1> (accessed Dec. 15, 2022).
- [34] “InceptionV3.” <https://keras.io/api/applications/inceptionv3/> (accessed Dec. 15,

- 2022).
- [35] “Xception.” <https://keras.io/api/applications/xception/> (accessed Dec. 15, 2022).
  - [36] “VGG Very Deep Convolutional Networks (VGGNet) - What you need to know - viso.ai.” <https://viso.ai/deep-learning/vgg-very-deep-convolutional-networks/> (accessed Dec. 15, 2022).
  - [37] “[1512.03385] Deep Residual Learning for Image Recognition.” <https://arxiv.org/abs/1512.03385> (accessed Dec. 15, 2022).
  - [38] “[1704.04861] MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications.” <https://arxiv.org/abs/1704.04861> (accessed Dec. 15, 2022).
  - [39] “[1608.06993] Densely Connected Convolutional Networks.” <https://arxiv.org/abs/1608.06993> (accessed Dec. 15, 2022).
  - [40] “CNN Starter - NasNet Mobile ( 0.9709 LB) | Kaggle.” <https://www.kaggle.com/code/CVxTz/cnn-starter-nasnet-mobile-0-9709-lb/notebook> (accessed Dec. 15, 2022).
  - [41] “Image classification via fine-tuning with EfficientNet.” [https://keras.io/examples/vision/image\\_classification\\_efficientnet\\_fine\\_tuning/](https://keras.io/examples/vision/image_classification_efficientnet_fine_tuning/) (accessed Dec. 15, 2022).



ORIGINALITY REPORT

---

16%

SIMILARITY INDEX

12%

INTERNET SOURCES

11%

PUBLICATIONS

%

STUDENT PAPERS

---

PRIMARY SOURCES

---

1	<a href="https://machinelearningmastery.com">machinelearningmastery.com</a> Internet Source	1%
2	<a href="https://maelfabien.github.io">maelfabien.github.io</a> Internet Source	1%
3	<a href="https://mc.ai">mc.ai</a> Internet Source	1%
4	<a href="https://huggingface.co">huggingface.co</a> Internet Source	1%
5	<a href="https://future.wsb.net.pl">future.wsb.net.pl</a> Internet Source	1%
6	Yogesh Kumar, Surbhi Gupta. "Deep Transfer Learning Approaches to Predict Glaucoma, Cataract, Choroidal Neovascularization, Diabetic Macular Edema, DRUSEN and Healthy Eyes: An Experimental Review", Archives of Computational Methods in Engineering, 2022 Publication	1%
7	Yogesh Kumar, Surbhi Gupta, Williamjeet Singh. "A novel deep transfer learning models	1%

for recognition of birds sounds in different environment", Soft Computing, 2022

Publication

---

8	<a href="http://ijircce.com">ijircce.com</a> Internet Source	1%
9	<a href="http://www.ijres.org">www.ijres.org</a> Internet Source	1%
10	<a href="http://cad-journal.net">cad-journal.net</a> Internet Source	1%
11	Suet-Peng Yong, Yoon-Chow Yeong. "Human Object Detection in Forest with Deep Learning based on Drone's Vision", 2018 4th International Conference on Computer and Information Sciences (ICCOINS), 2018 Publication	<1%
12	<a href="http://www.ijert.org">www.ijert.org</a> Internet Source	<1%
13	Andrea Barucci, Costanza Cucci, Massimiliano Franci, Marco Loschiavo, Fabrizio Argenti. "A Deep Learning Approach to Ancient Egyptian Hieroglyphs Classification", IEEE Access, 2021 Publication	<1%
14	Willian M. Freire, Aline M. M. M. Amaral, Yandre M. G. Costa. "Gemstone classification using ConvNet with transfer learning and fine-tuning", 2022 29th International Conference	<1%

# on Systems, Signals and Image Processing (IWSSIP), 2022

Publication

15

"Innovative Data Communication Technologies and Application", Springer Science and Business Media LLC, 2022

Publication

<1 %

16

[analyticsindiamag.com](http://analyticsindiamag.com)

Internet Source

<1 %

17

Hamza Ahmed Shad, Quazi Ashikur Rahman, Nashita Binte Asad, Atif Zawad Bakshi et al. "Exploring Alzheimer's Disease Prediction with XAI in various Neural Network Models", TENCON 2021 - 2021 IEEE Region 10 Conference (TENCON), 2021

Publication

<1 %

18

[tanthiamhuat.files.wordpress.com](http://tanthiamhuat.files.wordpress.com)

Internet Source

<1 %

19

[knowledgecommons.lakeheadu.ca](http://knowledgecommons.lakeheadu.ca)

Internet Source

<1 %

20

[osuva.uwasa.fi](http://osuva.uwasa.fi)

Internet Source

<1 %

21

Revanth Shankar Muthuselvam, Ramón Moreno, Mario Guemes, Miguel Del Río Cristobal et al. "Chapter 37 Deep Learning Based Baynat Foam Classification

<1 %

forHeadliners Manufacturing", Springer  
Science and Business Media LLC, 2023  
Publication

---

22 [ijsrcseit.com](https://www.ijsrcseit.com) <1 %  
Internet Source

---

23 Mohamed Afify, Mohamed Loey, Ahmed  
Elsawy. "A Robust Intelligent System for  
Detecting Tomato Crop Diseases Using Deep  
Learning", International Journal of Software  
Science and Computational Intelligence, 2022  
Publication

---

24 [ndvsu.org](https://www.ndvsu.org) <1 %  
Internet Source

---

25 "CIGOS 2021, Emerging Technologies and  
Applications for Green Infrastructure",  
Springer Science and Business Media LLC,  
2022  
Publication

---

26 [www.linuxlinks.com](https://www.linuxlinks.com) <1 %  
Internet Source

---

27 Roberto Pierdicca, Flavio Tonetto, Marco  
Mameli, Riccardo Rosati, Primo Zingaretti.  
"Chapter 13 Can AI Replace Conventional  
Markerless Tracking? A Comparative  
Performance Study for Mobile Augmented  
Reality Based on Artificial Intelligence",

# Springer Science and Business Media LLC, 2022

Publication

28

"Intelligent Computing", Springer Science and  
Business Media LLC, 2022

Publication

<1 %

29

[www.spadontologico.com.br](http://www.spadontologico.com.br)

Internet Source

<1 %

30

[hackernoon.com](http://hackernoon.com)

Internet Source

<1 %

31

[tsukuba.repo.nii.ac.jp](http://tsukuba.repo.nii.ac.jp)

Internet Source

<1 %

32

[visualstudio.microsoft.com](http://visualstudio.microsoft.com)

Internet Source

<1 %

33

[www.coursehero.com](http://www.coursehero.com)

Internet Source

<1 %

34

[www.tandfonline.com](http://www.tandfonline.com)

Internet Source

<1 %

35

"Innovations in Smart Cities Applications  
Volume 5", Springer Science and Business  
Media LLC, 2022

Publication

<1 %

36

[link.springer.com](http://link.springer.com)

Internet Source

<1 %

37

[web.archive.org](http://web.archive.org)

Internet Source

<1 %

38

"Image and Graphics", Springer Science and Business Media LLC, 2017

Publication

<1 %

39

Masoumeh Izadi, Aiden Chia, Bernard Cheng, Shangjing Wu. "NetClips: A Framework for Video Analytics in Sports Broadcast", 2018 IEEE International Conference on Big Data (Big Data), 2018

Publication

<1 %

40

Shamshad Ansari. "Building Computer Vision Applications Using Artificial Neural Networks", Springer Science and Business Media LLC, 2020

Publication

<1 %

41

Sujigarasharma K., Rathi R., Visvanathan P., Kanchana R.. "chapter 9 Emotion-Based Human-Computer Interaction", IGI Global, 2022

Publication

<1 %

42

[scholarcommons.usf.edu](https://scholarcommons.usf.edu)

Internet Source

<1 %

43

"Intelligent Systems in Medicine and Health", Springer Science and Business Media LLC, 2022

Publication

<1 %

44	Jesús M. Almendros-Jiménez, Luis Iribarne. "Chapter 12 Describing Use Cases with Activity Charts", Springer Science and Business Media LLC, 2005 Publication	<1 %
45	Parvathi R., Pattabiraman V.. "chapter 12 A Similarity-Based Object Classification Using Deep Neural Networks", IGI Global, 2019 Publication	<1 %
46	ebin.pub Internet Source	<1 %
47	"Big Data Analytics and Machine Intelligence in Biomedical and Health Informatics", Wiley, 2022 Publication	<1 %
48	"Artificial Intelligence and Blockchain for Future Cybersecurity Applications", Springer Science and Business Media LLC, 2021 Publication	<1 %
49	"Innovation in Information Systems and Technologies to Support Learning Research", Springer Science and Business Media LLC, 2020 Publication	<1 %
50	Sara Dilshad, Nikhil Singh, M. Atif, Atif Hanif, Nafeesah Yaqub, W.A. Farooq, Hijaz Ahmad, Yu-ming Chu, Muhammad Tamoor Masood.	<1 %

"Automated image classification of chest x-rays of covid-19 using deep transfer learning",  
Results in Physics, 2021

Publication

---

---