DAYANANDA SAGAR UNIVERSITY

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING SCHOOL OF ENGINEERING
DAYANANDA SAGAR UNIVERSITY KUDLU GATE BANGALORE - 560068

MAJOR PROJECT REPORT

ON

"Anomaly And Activity Recognition using Machine Learning in Video Surveillance"

SUBMITTED TO THE 8th SEMESTER
MAJOR PROJECT (16CS481)
COURSE
BACHELOR OF TECHNOLOGY
IN
COMPUTER SCIENCE & ENGINEERING

Submitted by

Aasif Chauhan	(ENG18CS1003)
Sahana N	(ENG16CS0134)
V Maheshwari Sanjana	(ENG17CS0236)
Yash Advani	(ENG17CS0254)

Under the supervision of

Prof. Renuka Devi

DAYANANDA SAGAR UNIVERSITY

School of Engineering, Kudlu Gate, Bangalore-560068

CERTIFICATE

This is to certify that ***Aasif Chauhan, Sahana N , V Maheshwari Sanjana and Yash Advani*** bearing ***USN ENG18CS1003, ENG16CS0134, ENG17CS0236 and ENG17CS0254*** *has satisfactorily completed his/her Major Project (Phase-I) as prescribed by the University for the 7th semester B.Tech. programme in Computer Science & Engineering for Major Project (16CS481) course during the year 2020 at the School of Engineering, Dayananda Sagar University., Bangalore.*

Date: _____ _____

Signature of the faculty in-charge

Max Marks	Marks Obtained

_____

Signature of Chairman Department
of Computer Science & Engineering

ACKNOWLEDGEMENT

From the very core of our heart, we would like to express our sincere gratitude to **Prof Renuka Devi** for her invaluable guidance, support, motivation and patience during the course of this mini- project work. We are always indebted to her for her kind support and constant encouragement.

We extend our sincere thanks to our **Chairman Dr. Sanjay Chitnis** who continuously helped throughout the project and without his guidance, this project would have been an uphill task.

It requires lots of efforts in terms of cooperation and support to fulfill various tasks involved during the project. We are always grateful to our peers and friends who have always encouraged us and guided us whenever we needed assistance.

Aasif Chauhan ENG18CS1003
Sahana N ENG16CS0134
V Maheshwari Sanjana ENG17CS0236
Yash Advani ENG17CS0254

CONTENT

Abstract

In the current era, where one of the most exciting tools that have been popularized in the toolbox in recent years is machine learning because of advancement in cloud computing.It resulted in a progress of lot of stagnated research we can finally make further advancement in “Anomaly Detection and Activity Recognition of Humans”.

The major drawback in the *traditional approach*, that there is a need to perform manual operation for detecting anomaly which lead to possibilities of human errors. This paper focuses on “*Anomaly Detection and Activity Recognition of Humans”*. The anomaly detection system uses *Convolutional Neural Network* (CNN) to solve the problems of manual operation such as the false alarms, missing of anomalous events and locating the position of an anomaly in the image. The image wise abnormal event is detected and the location of the abnormality in a image is detected using Convolutional Neural Network.

PROBLEM STATEMENT

The system designed for crowd monitoring at the public spaces like train stations, stadiums, airport terminals, should consider crowd density, peak time of traffic in the areas etc., proper crowd management at public places can help to avoid crowd mismanagement and ensure public safety, using the approach of 'Anomaly and Activity

Recognition using Machine learning for video-based surveillance, this can be achieved.

- Developments in deep learning in the field of physical security and surveillance is path-breaking. In the field of video surveillance, several applications or problem statements stand out that can benefit from machine learning and deep learning, abnormal event detection and activity recognition being two examples.
- Deep learning approaches can improve the image processing technique efficiently without explicitly processing the features like edges, RGB levels etc. of an image separately. Though traditional machine learning approaches like SVM, regression learning, are used in an image processing and enhancement field.
- TensorFlow provides significant success in predicting, creating, classifying an object or activities by using its various model.
- The novelty of the proposed approach is the combination of anomaly detection at the frame level with that at the sub-frame level.
- While detecting which frames have activity different from the baseline it is equally important to detect which part of the frame has the anomalous activity happening in and to classify the activity.
- The motivation to do this work is the idea that Machine Learning can improve the results of video surveillance in terms of accuracy and cost.

The cost can be reduced in terms of less manual operation, less storage for processing as Compared to traditional image processing techniques. Automated detection of abnormal events and recognition of activities of humans in the video can be helpful.

INTRODUCTION

A lot of research has been proposed in the field of video surveillance in terms of object tracking and human behavior. There are numerous applications of video surveillance in various fields such as crowd surveillance, industrial monitoring, forest fire controlling, traffic surveillance, aerial monitoring, security surveillance, post disaster management etc. Issues with conventional video surveillance approaches are:

(i) lack of real time video processing,

(ii) time consuming,

(iii) human error that leads to a false alarm,

(iv) maintenance and storage constrain,

(v) inefficient when there is a large crowd.

Manual operation for action recognition is time consuming, tiresome, and inefficient especially over a place where the crowd is dense. There is a need of an automated system which optimizes operational issues and sends alerts to human operators without a time delay. This system should be capable to detect operational errors and generates a notification. The system should be capable to track an abnormal event in each frame and generate a notification of such an event.

An object tracking and anomaly detection can be improved by applying machine learning techniques on it. Machine learning provides the ability to learn from a trained dataset. The video surveillance application such as abnormal event detection or activity recognition provides better results by using machine learning.

CNN can extract the features and patterns in a video faster than traditional image processing techniques. Module 1 :- Training CNN and Module 2:testing model.

The system designed for crowd monitoring at the public spaces like train stations, stadiums, airport terminals, should consider crowd density, peak time of traffic in the areas etc. The proper crowd management at public places can help to avoid crowd mismanagement and ensure public safety.

In the current era, the majority of public places such as supermarket, public garden, mall, university campus, etc. are under video surveillance. There is a need to provide essential security and monitor unusual anomaly activities at such places.

LITERATURE SURVEY

- **Human Activity Recognition in Smart Home with Deep Learning Approach:**
  (Homay Danaei Mehr, Huseyin Polat ; 2019)
  In this study in order to recognize human action of a smart home video dataset (DMLSmartActions) Convolutional Neural Networks (CNNs) architecture as a deep learning model has been proposed. This architecture may not be a sufficient model for any other datasets.

- **Activity recognition using Neural Network:**
  Mohanad Babiker1 , Othman O. khalifa1 , Kyaw Kyaw Htike2 , Aisha Hassan1 , Muhamed Zaharadeen1 ; 2017
  An intelligent human activity system recognition is developed using Neural Network. Where the system will detect different activities like walking, hand waving, sitting, boxing. But, , recognition of the activity cant be done in an traffic area . It is only preferrable for indoors.

- **Abnormal Human Activity Recognition using Bayes Classifier and Convolutional Neural Network:**
  (Congcong Liu, Jie Ying, Feilong Han, Ming Ruan ; 2018)*
  The method uses Bayes Classifier and Convolutional Neural Network to detect four activities, including walking, running, punching and tripping. Convolutional
  Neural Network performs better than Bayes Classifier in the experiment.

REQUIREMENT ANALYSIS

Real time computing (RTC) is the study of hardware and software systems that are subject to a "real-time constraint"—i.e., operational deadlines from event to system response. A real time system may be one where its application can be considered (within context) to be mission critical. A system is said to be real-time if the total correctness of an operation depends not only upon its logical correctness, but also upon the time in which it is performed.
Real time systems can be classified into 2 types

- Hard Real Time Systems
- Soft Real Time Systems

A hard real-time system is one where the completion of operation after its deadline is considered to be useless – may be leading to a critical failure of the entire system. Whereas a soft real-time system will tolerate such lateness and may respond with decreased service. The system to be developed is safety-critical but at the same time, a certain time span has to be provided to the algorithm to make sure whether the driver is really fatigued or the data obtained was just a one-off glitch.

System Requirements:

- The system should be capable to run in real time
- The false positive rate must be minimal
- The system should be capable of alerting during anomaly
- The system must be able to perform with high accuracy even in crowded places

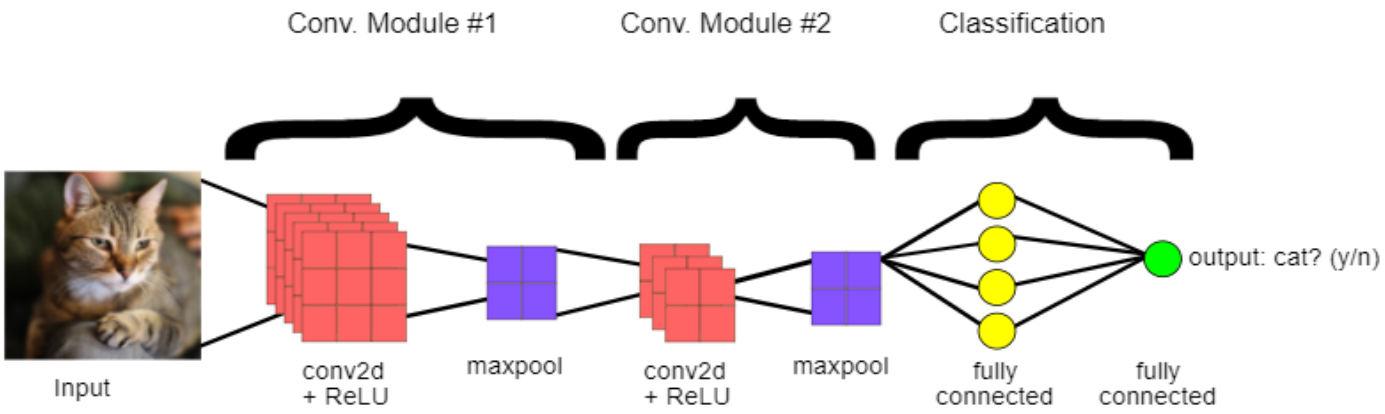
Software Requirement:

- Operating system : Windows 7.
- Software Tool : Open CV Python
- Coding Language : Python
- Toolbox : Image processing toolbox.

Hardware Requirement:

- Laptop or desktop with windows latest version (preferred)
- 8GB ram
- Intel i5 core processor
- 4GB graphic card min

Proposed methodology



- In CNN model we are using four layers for feature extraction and classification that is

1. 2D Convolutional layer
2. ReLu layer
3. Max pooling layer
4. Fully connected layer

Step I: CNN Model

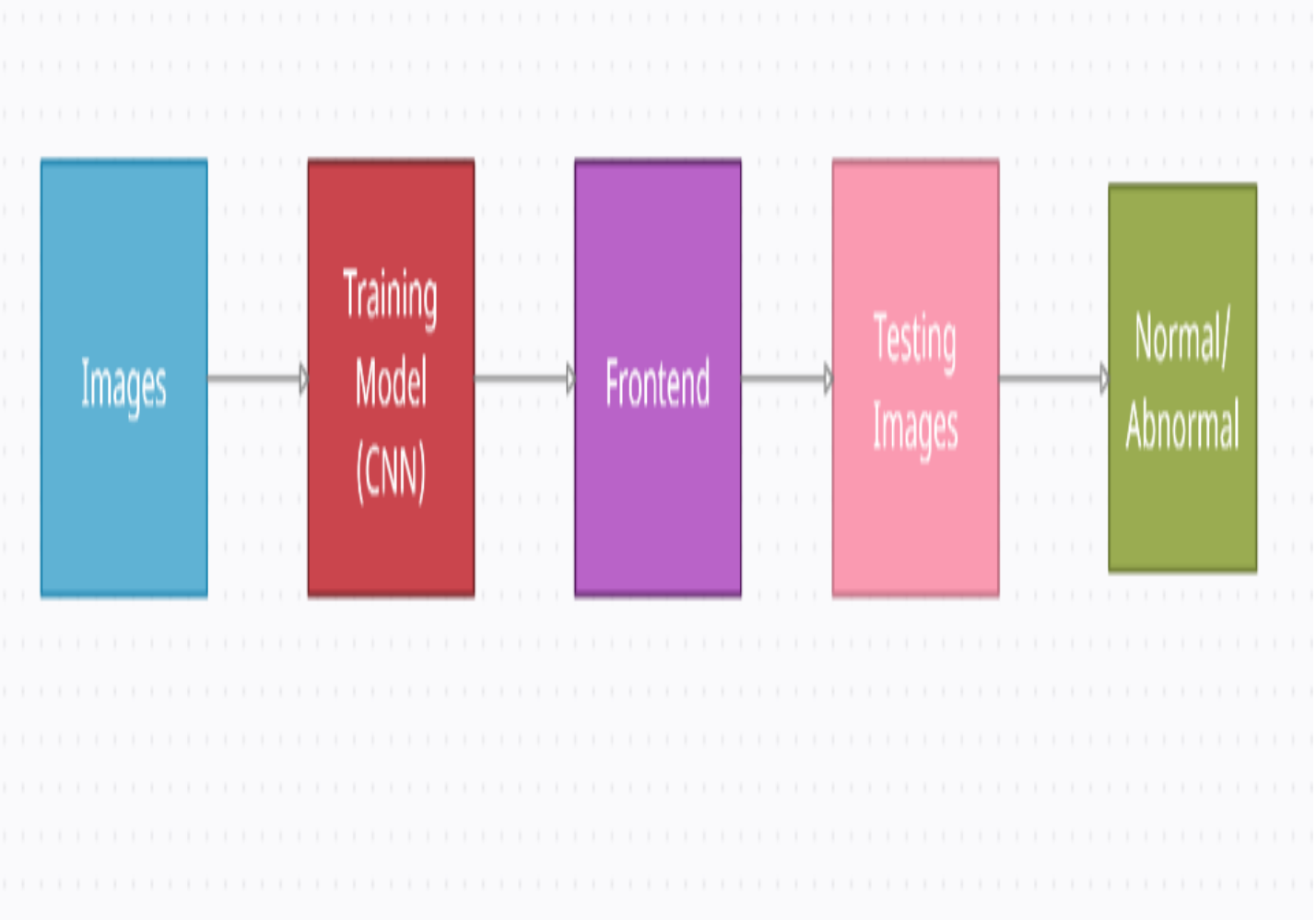
- The first step is to create CNN model.
- Train the model with large number of images.
- The model is being trained for different actions like chasing, walking, fighting, falling, leaving packages, running etc.,
- Preprocess of the dataset frames is done to resize the images and training the model using cascade trainer.
- Finally, the CNN model is created.

Step II:

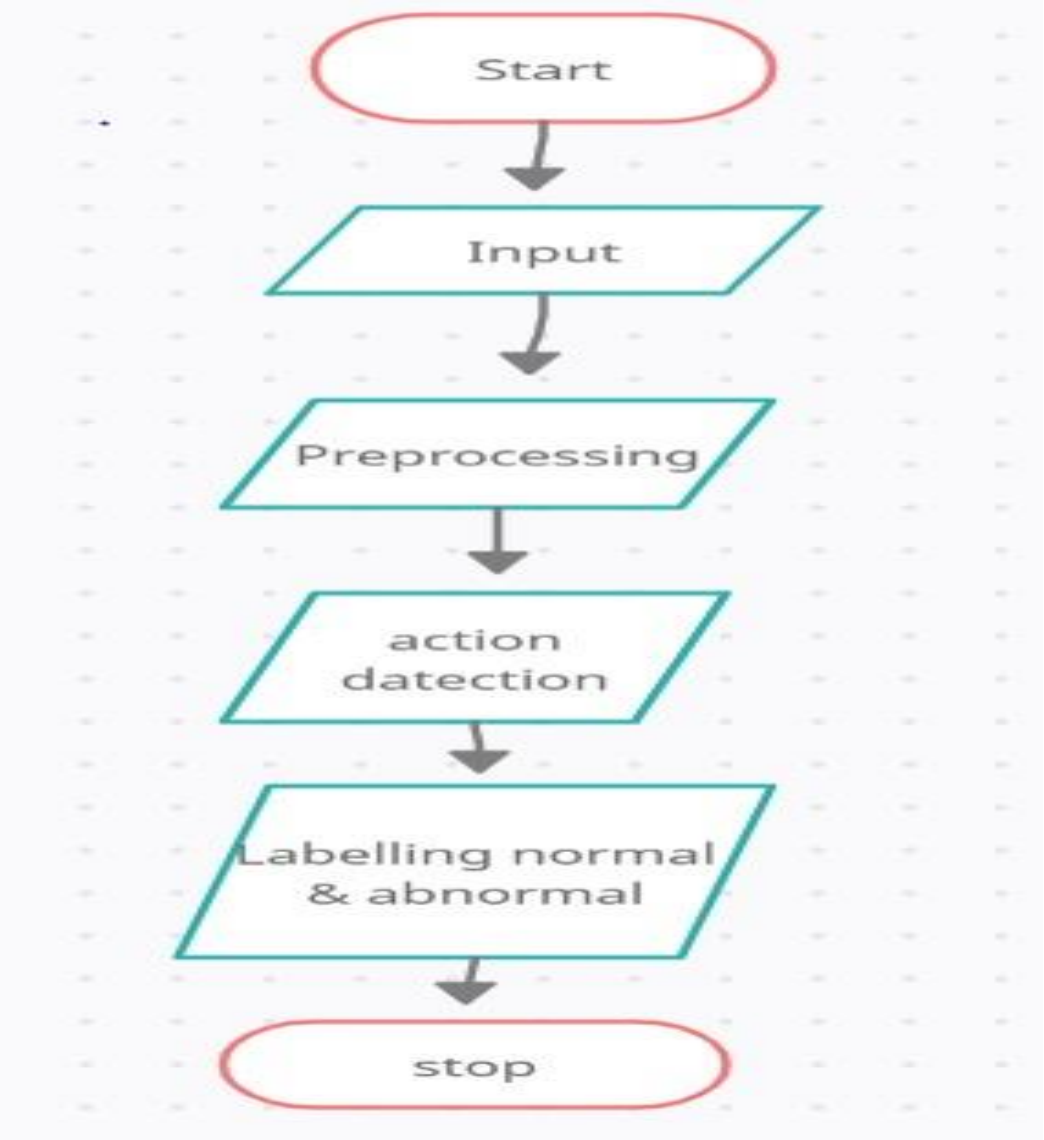
- The already trained CNN model is being used and connected to a new frontend page.
- Front end with GUI is created where the users can interact with the visualizations and see the status in real-time.
- Here it leads to a welcome window, which redirects to next window.
- Then the second window is where the image is fetched, analyzed and output is predicted by using CNN in the back end
- The front end is done using tkinter and back end using TensorFlow, Kera's which is used for loading the model.
- After the image is analyzed, the system will display the output by recognizing the action whether the image is normal or abnormal.
- Once the image is analyzed it will also create an alert by playing sound to the user according to the event detected
- Based on the accuracy received after training the model, a accuracy matrix is plotted using matplot library.

DESIGN

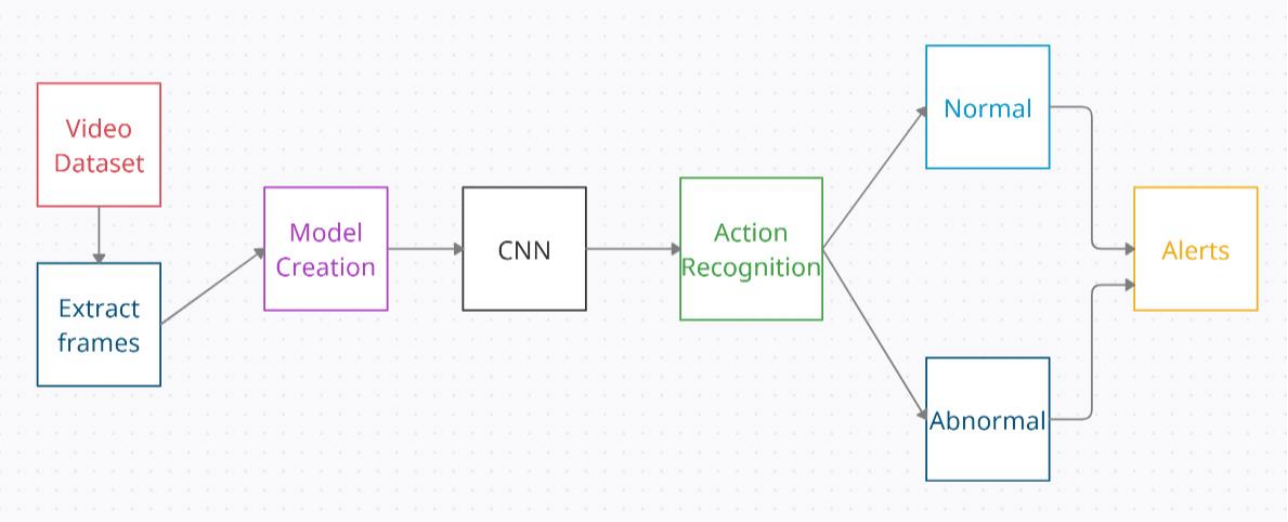
Block Diagram



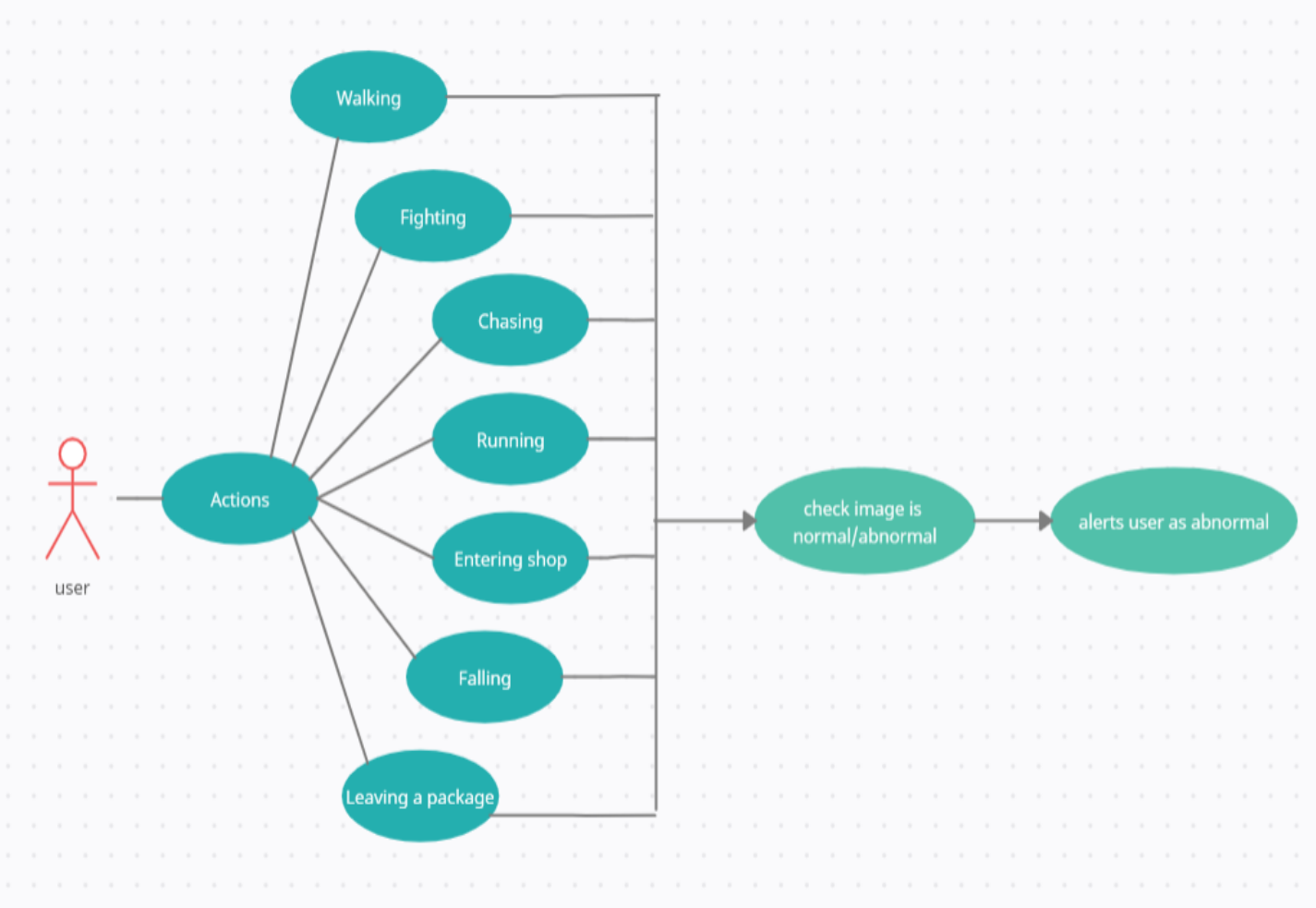
FLOWCHART


Start
Input
Preprocessing
action
datection
Labelling normal
& abnormal
stop

ARCHITECTURE DIAGRAM



USE CASE DIAGRAM



Code

Front end

```
from tkinter import *
import tkinter as tk
import tkinter.font as font
import tkinter.ttk as ttk
from tkinter import *
import os
import sys
from PIL import Image, ImageTk
import time
import tkinter
window = tk.Tk()
window.title("WELCOME")
window.geometry('1920x1800')
##
##window.configure(background ="light green")
ima = PhotoImage(file = 'C:\\Users\\dhruva adithya\\Desktop\\Datasets\\img1.png')
label = tk.Label(window, image = ima)
lb3=tk.Label(window, text="UNUSUAL EVENT DETECTION",font=("Century Gothic",34,"bold","italic"),foreground="black",bg="sky blue")
lb3.place(x=500,y=50)
lb2=tk.Label(window, text=" This project presents an algorithm which is able to detect unusual event image..  ",font=("Century Gothic",14,"bold","italic"),foreground="black",bg="sky blue")
lb2.place(x=100,y=200)


lb1=tk.Label(window, text="This system will check a image and classify as Normal and Abnormal",font=("Century Gothic",14,"bold","italic"),foreground="black",bg="sky blue")
lb1.place(x=100,y=235)


lb3=tk.Label(window, text="Alerts the user as normal and abnormal action.. ",font=("Century Gothic",14,"bold","italic"),foreground="black",bg="sky blue")
lb3.place(x=100,y=270)


def fun1():
    os.system("python frnt.py")


btn=tk.Button(window, text="Click Here",command=fun1,width=14,height=1,font=("Century Gothic",24,"bold","italic"),foreground="black",bg="skyblue")
```

```
btn.place(x=1000,y=500)

label.pack()
window.mainloop()
```

Front end

```
.from tkinter import *
import tkinter as tk
from PIL import Image
from PIL import ImageTk, Image
import cv2
import os
import sys
from tkinter import *
import tkinter as ttk
import random
from playsound import playsound

root = Tk()

root.title("Anomaly Detection")

root.geometry("1280x720")
#root.configure(background ="LightCyan3")
BG = PhotoImage(file = '33.png')
label = ttk.Label(root, image = BG)
#PhotoImage(file = 'Lotus-Flowers.png')

lbl = tk.Label(root, text="Anomaly Detection",width=20 ,height=1  ,fg="white"  ,bg="black" ,font=('Arial', 30, ' bold ') )
lbl.place(x=450, y=10)

def CNN():
##      lbl = tk.Label(root, text="You have Selected CNN",width=25  ,height=1  ,fg="Black" ,bg="LightCyan3" ,font=('Arial', 20, ' bold ') )
##    lbl.place(x=450, y=100)
    import tkinter as tk
    from tkinter.filedialog import askopenfilename
```

```
    import shutil
    import os
    import sys
    import cv2
    import numpy as np
    from PIL import Image, ImageTk
    import matplotlib.pyplot as plt
    global status
    sift=cv2.xfeatures2d.SIFT_create()
##    root = tk.Tk()
##
##    root.title("Cancer Prediction")
##
##    root.geometry("500x510")
##    root.configure(background ="lightgreen")

    title = tk.Label(text="Click below to choose picture for testing ....", background = "black", fg="white", font=("", 15))
    title.place(x=450,y=100)

    def analysis():
        import cv2  # working with, mainly resizing, images
        import numpy as np  # dealing with arrays
        import os  # dealing with directories
        from random import shuffle  # mixing up or currently ordered data that might lead our network astray in training.
        from tqdm import \
            tqdm  # a nice pretty percentage bar for tasks. Thanks to viewer Daniel BA1/4hler for this suggestion
        verify_dir = 'testpicture'
        IMG_SIZE = 50
        LR = 1e-3
        MODEL_NAME = 'healthyvsunhealthynew-{}-{}.model'.format(LR, '2conv-basic')

        def process_verify_data():
            verifying_data = []
            for img in tqdm(os.listdir(verify_dir)):
                path = os.path.join(verify_dir, img)
                img_num = img.split('.')[0]
                img = cv2.imread(path, cv2.IMREAD_COLOR)
```

```
            img = cv2.resize(img, (IMG_SIZE, IMG_SIZE))

##                  return func1()
            verifying_data.append([np.array(img), img_num])
        np.save('verify_data.npy', verifying_data)
        return verifying_data

    verify_data = process_verify_data()
    #verify_data = np.load('verify_data.npy')

    import tflearn
    from tflearn.layers.conv import conv_2d, max_pool_2d
    from tflearn.layers.core import input_data, dropout, fully_connected
    from tflearn.layers.estimator import regression

    import tensorflow as tf
    #tf.reset_default_graph()

    convnet = input_data(shape=[None, IMG_SIZE, IMG_SIZE, 3], name='input')

    convnet = conv_2d(convnet, 32, 3, activation='relu')
    convnet = max_pool_2d(convnet, 3)

    convnet = conv_2d(convnet, 64, 3, activation='relu')
    convnet = max_pool_2d(convnet, 3)

    convnet = conv_2d(convnet, 128, 3, activation='relu')
    convnet = max_pool_2d(convnet, 3)

    convnet = conv_2d(convnet, 32, 3, activation='relu')
    convnet = max_pool_2d(convnet, 3)

    convnet = conv_2d(convnet, 64, 3, activation='relu')
    convnet = max_pool_2d(convnet, 3)

    convnet = fully_connected(convnet, 1024, activation='relu')
    convnet = dropout(convnet, 0.8)

    convnet = fully_connected(convnet, 9, activation='softmax')
```

```
    convnet = regression(convnet, optimizer='adam', learning_rate=LR,
loss='categorical_crossentropy', name='targets')

    model = tflearn.DNN(convnet, tensorboard_dir='log')

    if os.path.exists('{}.meta'.format(MODEL_NAME)):
        model.load(MODEL_NAME)
        print('model loaded!')
        prediction= random.randint(90,100)

    import matplotlib.pyplot as plt
 ##    from ui_Bone import func1

    fig = plt.figure()

    for num, data in enumerate(verify_data):

        img_num = data[1]
        img_data = data[0]
        y = fig.add_subplot(3, 4, num + 1)
        orig = img_data
        data = img_data.reshape(IMG_SIZE, IMG_SIZE, 3)
        # model_out = model.predict([data])[0]
        model_out = model.predict([data])[0]
        print(model_out)

    if np.argmax(model_out) == 0:
        str_label = 'chasing'
    elif np.argmax(model_out) == 1:
        str_label = 'crowd running'
    elif np.argmax(model_out) == 2:
        str_label = 'entering shop'
    elif np.argmax(model_out) == 3:
        str_label = 'falling'
    elif np.argmax(model_out) == 4:
        str_label = 'Fighting'
    elif np.argmax(model_out) == 5:
        str_label = 'leaving package'
    elif np.argmax(model_out) == 6:
```

```
            str_label = 'people meeting'
        elif np.argmax(model_out) == 7:
            str_label = 'Running'
        elif np.argmax(model_out) == 8:
            str_label = 'walking'



        global b

        if str_label == 'chasing':
            status = 'chasing(Abnormal)'
            playsound("abnormal.mp3")
##              stages()
           # send_H()
            message = tk.Label(text='Status: '+status, background="black",
                               fg="white", font=("", 15))
            message.place(x=650,y=380)
            button = tk.Button(text="Exit", command=exit)
            button.place(x=700,y=630)

            def predictions():
                import random
                import matplotlib.pyplot as plt

                predicts=random.randint(80,97)
                print(predicts)

                with open("accuracy.csv",'w') as f:
                    f.write(str(predicts))

                with open("accuracy.csv") as f:
                    data = f.readlines()

                ##x=0
                ##x=float(x)
                x=int(data[0])
                dic={'prediction' :x}
                x_axis=list(dic.keys())
                y_axis=list(dic.values())
```

```
            fig=plt.figure(figsize=(5,5))
            plt.bar( x_axis,y_axis, color='royalblue', alpha=0.7, align='center')
            plt.grid(color='#95a5a6', linestyle='--', linewidth=2, axis='y', alpha=0.7)
            plt.xlabel("X_axis")
            plt.ylabel("Accuracy Level")
            plt.title("Accuracy Matrix")
            plt.savefig('matrix.png', dpi=300, bbox_inches='tight')

            plt.show()
        button3 = tk.Button(text="prediction", command= predictions)
        button3.place(x=1000,y=400)
##
    elif str_label == 'crowd running':
        status= 'crowd running'
        playsound("abnormal.mp3")
##          stages()
       # send_H()
        message = tk.Label(text='Status: '+status, background="black",
                    fg="white", font=("", 15))
        message.place(x=650,y=380)

##          r.grid(column=0, row=4, padx=10, pady=10)
        button = tk.Button(text="Exit", command=root.destroy)
        button.place(x=700,y=630)

        def predictions():
            import random
            import matplotlib.pyplot as plt

            predicts=random.randint(80,97)
            print(predicts)

            with open("accuracy.csv",'w') as f:
                f.write(str(predicts))

            with open("accuracy.csv") as f:
                data = f.readlines()

            ##x=0
```

```
            ##x=float(x)
            x=int(data[0])
            dic={'prediction' :x}
            x_axis=list(dic.keys())
            y_axis=list(dic.values())
            fig=plt.figure(figsize=(5,5))
            plt.bar( x_axis,y_axis, color='royalblue', alpha=0.7, align='center')
            plt.grid(color='#95a5a6', linestyle='--', linewidth=2, axis='y', alpha=0.7)
            plt.xlabel("X_axis")
            plt.ylabel("Accuracy Level")
            plt.title("Accuracy Matrix")
            plt.savefig('matrix.png', dpi=300, bbox_inches='tight')

            plt.show()
        button3 = tk.Button(text="prediction", command= predictions)
        button3.place(x=1000,y=400)
##
    elif str_label == 'entering shop':
        status= 'entering shop(Normal)'
        playsound("normal.mp3")
##          stages()
        # send_H()
        message = tk.Label(text='Status: '+status, background="black",
                           fg="white", font=("", 15))
        message.place(x=650,y=380)

##          r = tk.Label(text='Melanocytic Nervs', background="brown", fg="white",font=("", 15))
##          r.grid(column=0, row=4, padx=10, pady=10)
        button = tk.Button(text="Exit", command=root.destroy)
        button.place(x=700,y=630)

        def predictions():
            import random
            import matplotlib.pyplot as plt

            predicts=random.randint(80,97)
            print(predicts)

            with open("accuracy.csv",'w') as f:
```

```
            f.write(str(predicts))

        with open("accuracy.csv") as f:
            data = f.readlines()

        ##x=0
        ##x=float(x)
        x=int(data[0])
        dic={'prediction' :x}
        x_axis=list(dic.keys())
        y_axis=list(dic.values())
        fig=plt.figure(figsize=(5,5))
        plt.bar( x_axis,y_axis, color='royalblue', alpha=0.7, align='center')
        plt.grid(color='#95a5a6', linestyle='--', linewidth=2, axis='y', alpha=0.7)
        plt.xlabel("X_axis")
        plt.ylabel("Accuracy Level")
        plt.title("Accuracy Matrix")
        plt.savefig('matrix.png', dpi=300, bbox_inches='tight')

        plt.show()
    button3 = tk.Button(text="prediction", command= predictions)
    button3.place(x=1000,y=400)
##
  elif str_label == 'falling':
    status= 'falling(Abnormal)'
    playsound("abnormal.mp3")
##      stages()
    # send_H()
    message = tk.Label(text='Status: '+status, background="black",
                fg="white", font=("", 15))
    message.place(x=650,y=380)

##      r = tk.Label(text='Melanocytic Nervs', background="brown", fg="white",font=("", 15))
##      r.grid(column=0, row=4, padx=10, pady=10)
    button = tk.Button(text="Exit", command=root.destroy)
    button.place(x=700,y=630)

    def predictions():
        import random
```

```
            import matplotlib.pyplot as plt

            predicts=random.randint(80,97)
            print(predicts)

            with open("accuracy.csv",'w') as f:
               f.write(str(predicts))

            with open("accuracy.csv") as f:
               data = f.readlines()

            ##x=0
            ##x=float(x)
            x=int(data[0])
            dic={'prediction' :x}
            x_axis=list(dic.keys())
            y_axis=list(dic.values())
            fig=plt.figure(figsize=(5,5))
            plt.bar( x_axis,y_axis, color='royalblue', alpha=0.7, align='center')
            plt.grid(color='#95a5a6', linestyle='--', linewidth=2, axis='y', alpha=0.7)
            plt.xlabel("X_axis")
            plt.ylabel("Accuracy Level")
            plt.title("Accuracy Matrix")
            plt.savefig('matrix.png', dpi=300, bbox_inches='tight')

            plt.show()
         button3 = tk.Button(text="prediction", command= predictions)
         button3.place(x=1000,y=400)
##
      elif str_label == 'Fighting':
         status= 'Fighting(Abnormal)'
         playsound("abnormal.mp3")
##          stages()
         # send_H()
         message = tk.Label(text='Status: '+status, background="black",
                     fg="white", font=("", 15))
         message.place(x=650,y=380)

##          r = tk.Label(text='Melanocytic Nervs', background="brown", fg="white",font=("", 15))
```

```
##          r.grid(column=0, row=4, padx=10, pady=10)
        button = tk.Button(text="Exit", command=root.destroy)
        button.place(x=700,y=630)

        def predictions():
            import random
            import matplotlib.pyplot as plt

            predicts=random.randint(80,97)
            print(predicts)

            with open("accuracy.csv",'w') as f:
                f.write(str(predicts))

            with open("accuracy.csv") as f:
                data = f.readlines()

            ##x=0
            ##x=float(x)
            x=int(data[0])
            dic={'prediction' :x}
            x_axis=list(dic.keys())
            y_axis=list(dic.values())
            fig=plt.figure(figsize=(5,5))
            plt.bar( x_axis,y_axis, color='royalblue', alpha=0.7, align='center')
            plt.grid(color='#95a5a6', linestyle='--', linewidth=2, axis='y', alpha=0.7)
            plt.xlabel("X_axis")
            plt.ylabel("Accuracy Level")
            plt.title("Accuracy Matrix")
            plt.savefig('matrix.png', dpi=300, bbox_inches='tight')

            plt.show()
        button3 = tk.Button(text="prediction", command= predictions)
        button3.place(x=1000,y=400)
##
    elif str_label == 'leaving package':
        status= 'leaving package(Abnormal)'
        playsound("abnormal.mp3")
##          stages()
```

```
        # send_H()
         message = tk.Label(text='Status: '+status, background="black",
                    fg="white", font=("", 15))
         message.place(x=650,y=380)

##          r = tk.Label(text='Melanocytic Nervs', background="brown", fg="white",font=("", 15))
##          r.grid(column=0, row=4, padx=10, pady=10)
         button = tk.Button(text="Exit", command=root.destroy)
         button.place(x=700,y=630)

         def predictions():
            import random
            import matplotlib.pyplot as plt

            predicts=random.randint(80,97)
            print(predicts)

            with open("accuracy.csv",'w') as f:
               f.write(str(predicts))

            with open("accuracy.csv") as f:
               data = f.readlines()

            ##x=0
            ##x=float(x)
            x=int(data[0])
            dic={'prediction' :x}
            x_axis=list(dic.keys())
            y_axis=list(dic.values())
            fig=plt.figure(figsize=(5,5))
            plt.bar( x_axis,y_axis, color='royalblue', alpha=0.7, align='center')
            plt.grid(color='#95a5a6', linestyle='--', linewidth=2, axis='y', alpha=0.7)
            plt.xlabel("X_axis")
            plt.ylabel("Accuracy Level")
            plt.title("Accuracy Matrix")
            plt.savefig('matrix.png', dpi=300, bbox_inches='tight')

            plt.show()
         button3 = tk.Button(text="prediction", command= predictions)
```

```
        button3.place(x=1000,y=400)
##
    elif str_label == 'people meeting':
        status= 'people meeting(Normal)'
        playsound("normal.mp3")
##          stages()
        # send_H()
        message = tk.Label(text='Status: '+status, background="black",
                  fg="white", font=("", 15))
        message.place(x=650,y=380)

##          r = tk.Label(text='Melanocytic Nervs', background="brown", fg="white",font=("", 15))
##          r.grid(column=0, row=4, padx=10, pady=10)
        button = tk.Button(text="Exit", command=root.destroy)
        button.place(x=700,y=630)

        def predictions():
            import random
            import matplotlib.pyplot as plt

            predicts=random.randint(80,97)
            print(predicts)

            with open("accuracy.csv",'w') as f:
                f.write(str(predicts))

            with open("accuracy.csv") as f:
                data = f.readlines()

            ##x=0
            ##x=float(x)
            x=int(data[0])
            dic={'prediction' :x}
            x_axis=list(dic.keys())
            y_axis=list(dic.values())
            fig=plt.figure(figsize=(5,5))
            plt.bar( x_axis,y_axis, color='royalblue', alpha=0.7, align='center')
            plt.grid(color='#95a5a6', linestyle='--', linewidth=2, axis='y', alpha=0.7)
            plt.xlabel("X_axis")
```

```
            plt.ylabel("Accuracy Level")
            plt.title("Accuracy Matrix")
            plt.savefig('matrix.png', dpi=300, bbox_inches='tight')

            plt.show()
        button3 = tk.Button(text="prediction", command= predictions)
        button3.place(x=1000,y=400)
##
    elif str_label == 'Running':
        status= 'Running(Abnormal)'
        playsound("abnormal.mp3")
##          stages()
        # send_H()
        message = tk.Label(text='Status: '+status, background="black",
                           fg="white", font=("", 15))
        message.place(x=650,y=380)

##          r = tk.Label(text='Melanocytic Nervs', background="brown", fg="white",font=("", 15))
##          r.grid(column=0, row=4, padx=10, pady=10)
        button = tk.Button(text="Exit", command=root.destroy)
        button.place(x=700,y=630)

        def predictions():
            import random
            import matplotlib.pyplot as plt

            predicts=random.randint(80,97)
            print(predicts)

            with open("accuracy.csv",'w') as f:
                f.write(str(predicts))

            with open("accuracy.csv") as f:
                data = f.readlines()

            ##x=0
            ##x=float(x)
            x=int(data[0])
            dic={'prediction' :x}
```

```
            x_axis=list(dic.keys())
            y_axis=list(dic.values())
            fig=plt.figure(figsize=(5,5))
            plt.bar( x_axis,y_axis, color='royalblue', alpha=0.7, align='center')
            plt.grid(color='#95a5a6', linestyle='--', linewidth=2, axis='y', alpha=0.7)
            plt.xlabel("X_axis")
            plt.ylabel("Accuracy Level")
            plt.title("Accuracy Matrix")
            plt.savefig('matrix.png', dpi=300, bbox_inches='tight')

            plt.show()
        button3 = tk.Button(text="prediction", command= predictions)
        button3.place(x=1000,y=400)
##
    elif str_label == 'walking':
        status= 'walking(Normal)'
        playsound("normal.mp3")
##          stages()
        # send_H()
        message = tk.Label(text='Status: '+status, background="black",
                           fg="white", font=("", 15))
        message.place(x=650,y=380)

##          r = tk.Label(text='Melanocytic Nervs', background="brown", fg="white",font=("", 15))
##          r.grid(column=0, row=4, padx=10, pady=10)
        button = tk.Button(text="Exit", command=root.destroy)
        button.place(x=700,y=630)

        def predictions():
            import random
            import matplotlib.pyplot as plt

            predicts=random.randint(80,97)
            print(predicts)

            with open("accuracy.csv",'w') as f:
                f.write(str(predicts))

            with open("accuracy.csv") as f:
```

```
            data = f.readlines()


        ##x=0
        ##x=float(x)
        x=int(data[0])
        dic={'prediction' :x}
        x_axis=list(dic.keys())
        y_axis=list(dic.values())
        fig=plt.figure(figsize=(5,5))
        plt.bar( x_axis,y_axis, color='royalblue', alpha=0.7, align='center')
        plt.grid(color='#95a5a6', linestyle='--', linewidth=2, axis='y', alpha=0.7)
        plt.xlabel("X_axis")
        plt.ylabel("Accuracy Level")
        plt.title("Accuracy Matrix")
        plt.savefig('matrix.png', dpi=300, bbox_inches='tight')


        plt.show()
      button3 = tk.Button(text="prediction", command= predictions)
      button3.place(x=1000,y=400)
##
  def openphoto():
    dirPath = "testpicture"
    fileList = os.listdir(dirPath)
    for fileName in fileList:
      os.remove(dirPath + "/" + fileName)
    # C:/Users/sagpa/Downloads/images is the location of the image which you want to test..... you can change it according to the image location you have
    fileName = askopenfilename(initialdir='C:\\Users\\dhruva adithya\\Desktop\\Datasets\\train', title='Select image for analysis ',
                       filetypes=[('image files', '.jpg')])
    dst = "testpicture"
    print(fileName)
    print (os.path.split(fileName)[-1])
    if os.path.split(fileName)[-1].split('.') == 'b (1)':
      print('dfdfffffffffffff')
    shutil.copy(fileName, dst)
    load = Image.open(fileName)
    render = ImageTk.PhotoImage(load)
    img = tk.Label(image=render, height="250", width="500")
    img.image = render
```

```
        img.place(x=470, y=100)
        #img.grid(column=0, row=1, padx=10, pady = 10)
        title.destroy()

        button1.destroy()
        button2 = tk.Button(text="Analyse Image", command=analysis)
        button2.place(x=670,y=380)
        #button2.grid(column=0, row=20, padx=10, pady = 10)
    button1 = tk.Button(text="Get Photo", command = openphoto)
    button1.place(x=600,y=150)
    #button1.grid(column=0, row=1, padx=10, pady = 10)

    root.mainloop()
    #os.system("python ui_Bone.py")

takeImg = tk.Button(root, text="CNN", command=CNN ,fg="white" ,bg="black" ,width=10 ,height=2, activebackground = "black" ,font=('times', 15, ' bold '))
takeImg.place(x=400, y=550)
trackImg = tk.Button(root, text="Quit", command=root.destroy ,fg="white" ,bg="Black" ,width=10 ,height=2, activebackground = "Black" ,font=('times', 15, ' bold '))
trackImg.place(x=900, y=550)
label.pack()
root.mainloop()
```

Back end

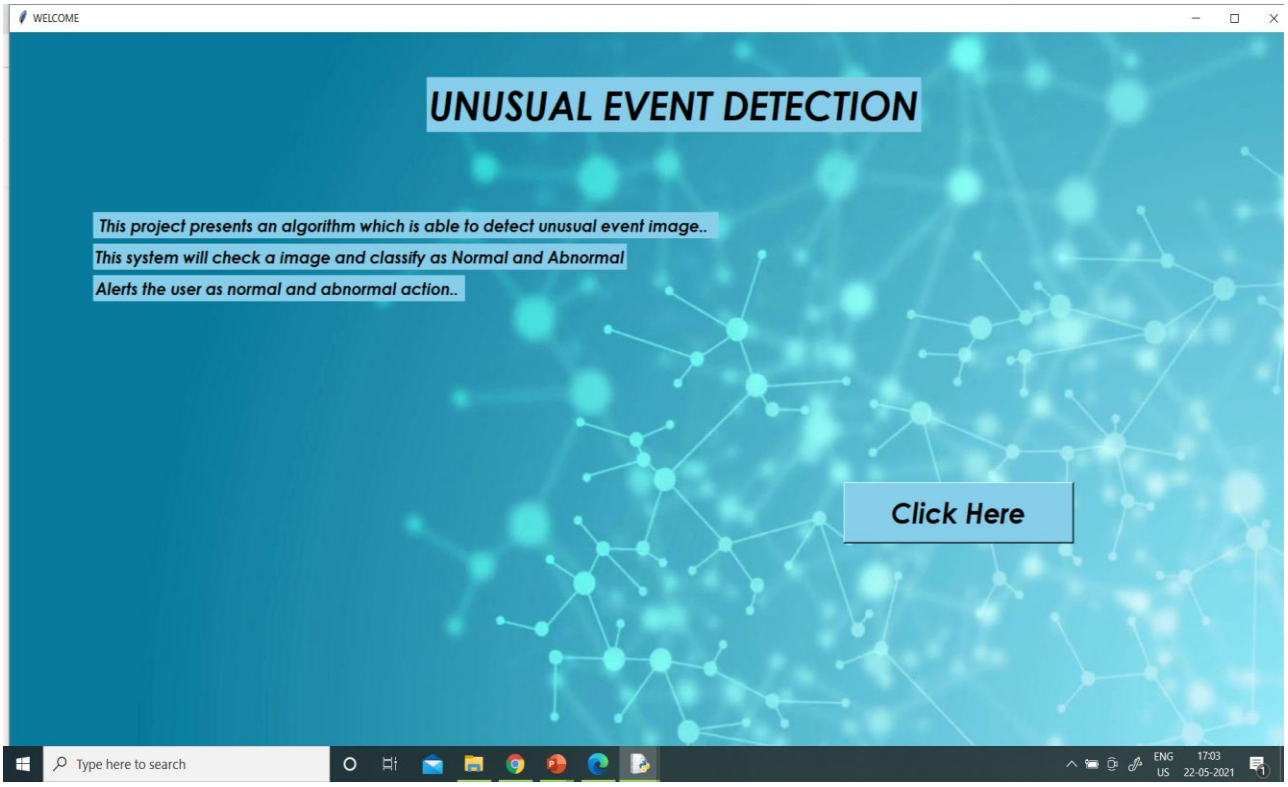
```
import cv2
import numpy as np
import os
from random import shuffle
from tqdm import tqdm
from tensorflow.python.framework import ops
# Visualize training history
from keras.models import Sequential
from keras.layers import Dense
import matplotlib.pyplot as plt
TRAIN_DIR = 'C:\\Users\\dhruva adithya\\Desktop\\Datasets\\train'
TEST_DIR = 'C:\\Users\\dhruva adithya\\Desktop\\Datasets\\test'

IMG_SIZE = 50
LR = 1e-3
MODEL_NAME = 'healthyvsunhealthynew-{}-{}.model'.format(LR, '2conv-basic')
def label_img(img):
    word_label = img[0]
    print(word_label)

    if word_label == 'a':
        print('chasing')
        return [1,0,0,0,0,0,0,0,0]
    elif word_label == 'b':
        print('crowd running')
        return [0,1,0,0,0,0,0,0,0]
    elif word_label == 'c':
        print('entering shop')
        return [0,0,1,0,0,0,0,0,0]
    elif word_label == 'd':
        print('falling')
        return [0,0,0,1,0,0,0,0,0]
    elif word_label == 'f':
        print('Fighting')
        return [0,0,0,0,1,0,0,0,0]
    elif word_label == 'g':
        print('leaving package')
        return [0,0,0,0,0,1,0,0,0]
```

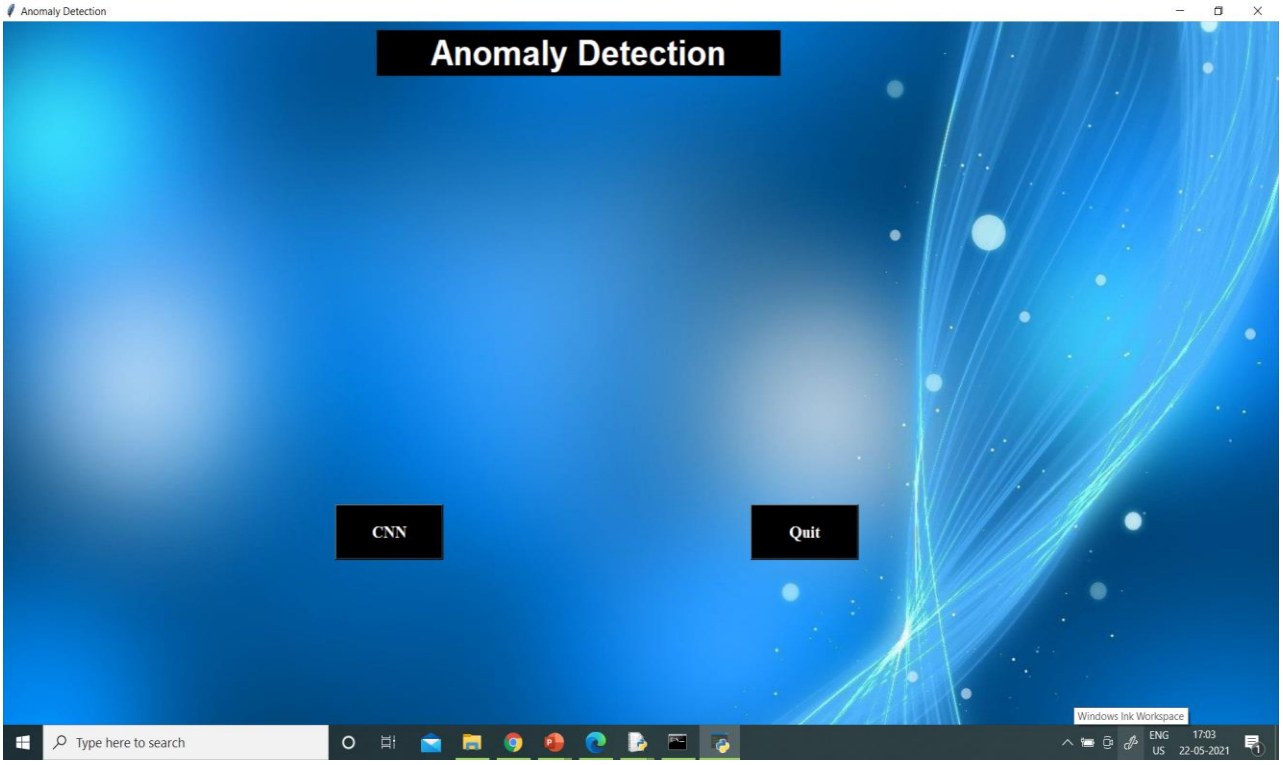
```
    elif word_label == 'h':
        print('people meeting')
        return [0,0,0,0,0,0,1,0,0]


    elif word_label == 'i':
        print('Running ')
        return [0,0,0,0,0,0,0,1,0]
    elif word_label == 'j':
        print('walking')
        return [0,0,0,0,0,0,0,0,1]

def create_train_data():
    training_data = []
    for img in tqdm(os.listdir(TRAIN_DIR)):
        label = label_img(img)
        print('###############')
        print(label)
        path = os.path.join(TRAIN_DIR,img)
        img = cv2.imread(path,cv2.IMREAD_COLOR)
        img = cv2.resize(img, (IMG_SIZE,IMG_SIZE))
        #img = cv2.resize(img,None,fx=0.5,fy=0.5)
        training_data.append([np.array(img),np.array(label)])
    shuffle(training_data)
    np.save('train_data.npy', training_data)
    return training_data

def process_test_data():
    testing_data = []
    for img in tqdm(os.listdir(TEST_DIR)):
        path = os.path.join(TEST_DIR,img)
        img_num = img.split('.')[0]
        img = cv2.imread(path,cv2.IMREAD_COLOR)
        img = cv2.resize(img, (IMG_SIZE,IMG_SIZE))
        testing_data.append([np.array(img), img_num])

    shuffle(testing_data)
    np.save('test_data.npy', testing_data)
    return testing_data
```

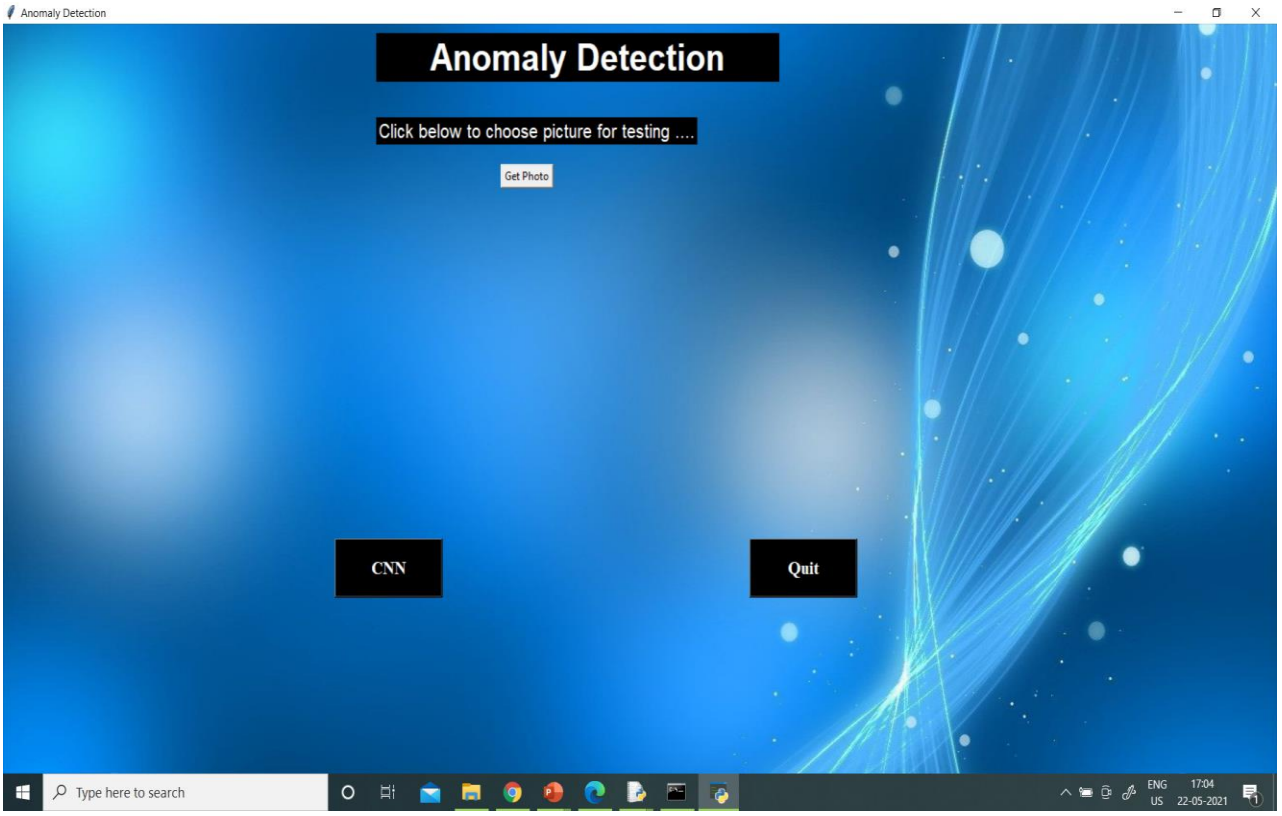
```
train_data = create_train_data()
# If you have already created the dataset:
#train_data = np.load('train_data.npy')

import tflearn
from tflearn.layers.conv import conv_2d, max_pool_2d
from tflearn.layers.core import input_data, dropout, fully_connected
from tflearn.layers.estimator import regression
import tensorflow as tf
from tensorflow.python.framework import ops

ops.reset_default_graph()

convnet = input_data(shape=[None, IMG_SIZE, IMG_SIZE, 3], name='input')

convnet = conv_2d(convnet, 32, 3, activation='relu')
convnet = max_pool_2d(convnet, 3)

convnet = conv_2d(convnet, 64, 3, activation='relu')
convnet = max_pool_2d(convnet, 3)

convnet = conv_2d(convnet, 128, 3, activation='relu')
convnet = max_pool_2d(convnet, 3)

convnet = conv_2d(convnet, 32, 3, activation='relu')
convnet = max_pool_2d(convnet, 3)
convnet = conv_2d(convnet, 64, 3, activation='relu')
convnet = max_pool_2d(convnet, 3)
convnet = fully_connected(convnet, 1024, activation='relu')
convnet = dropout(convnet, 0.8)
        convnet = fully_connected(convnet, 9, activation='softmax')
convnet = regression(convnet, optimizer='adam', learning_rate=LR,
loss='categorical_crossentropy', name='targets')
model = tflearn.DNN(convnet, tensorboard_dir='log')
if os.path.exists('{}.meta'.format(MODEL_NAME)):
   model.load(MODEL_NAME)
   print('model loaded!')
train = train_data[:-1000]
```

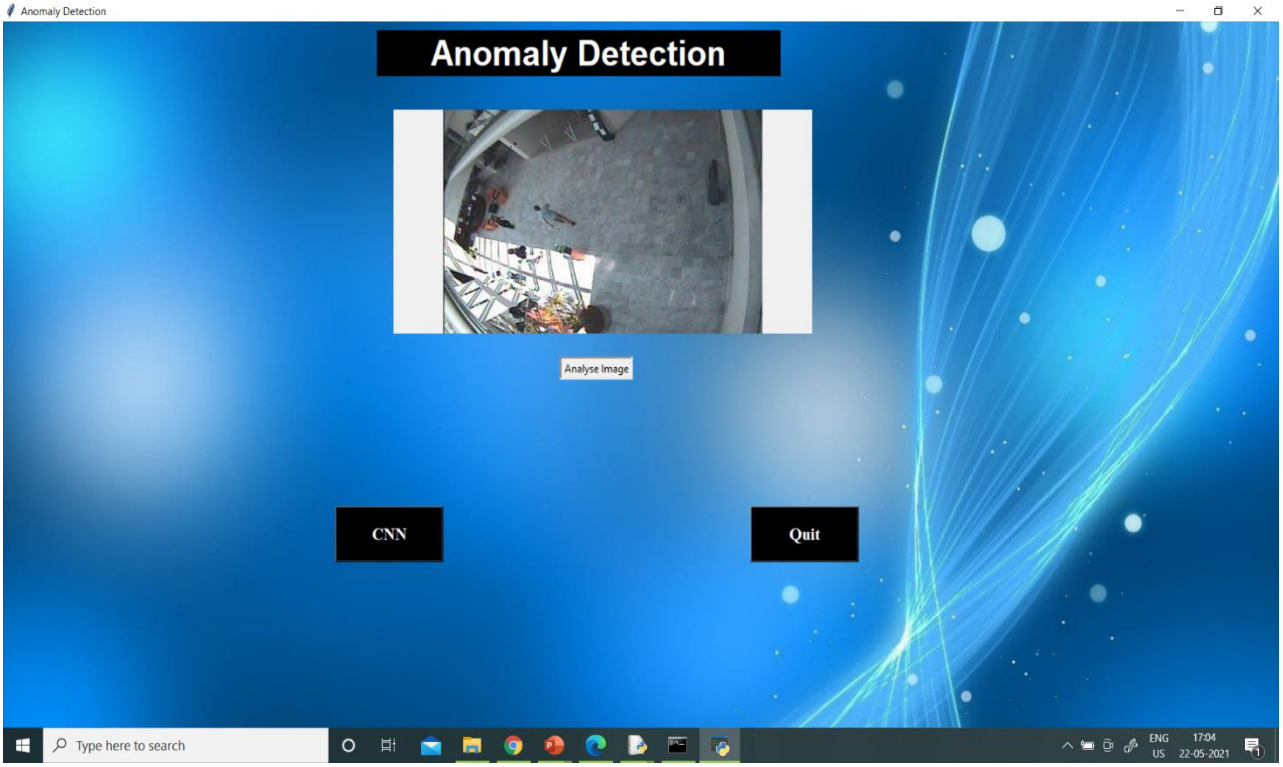
```
test = train_data[-1000:]

X = np.array([i[0] for i in train]).reshape(-1,IMG_SIZE,IMG_SIZE,3)
Y = [i[1] for i in train]
print(X.shape)
test_x = np.array([i[0] for i in test]).reshape(-1,IMG_SIZE,IMG_SIZE,3)
test_y = [i[1] for i in test]
print(test_x.shape)

history=model.fit({'input': X}, {'targets': Y},n_epoch=100, validation_set=({'input': test_x}, {'targets': test_y}),snapshot_step=90, show_metric=True, run_id=MODEL_NAME)
model.save(MODEL_NAME)
```

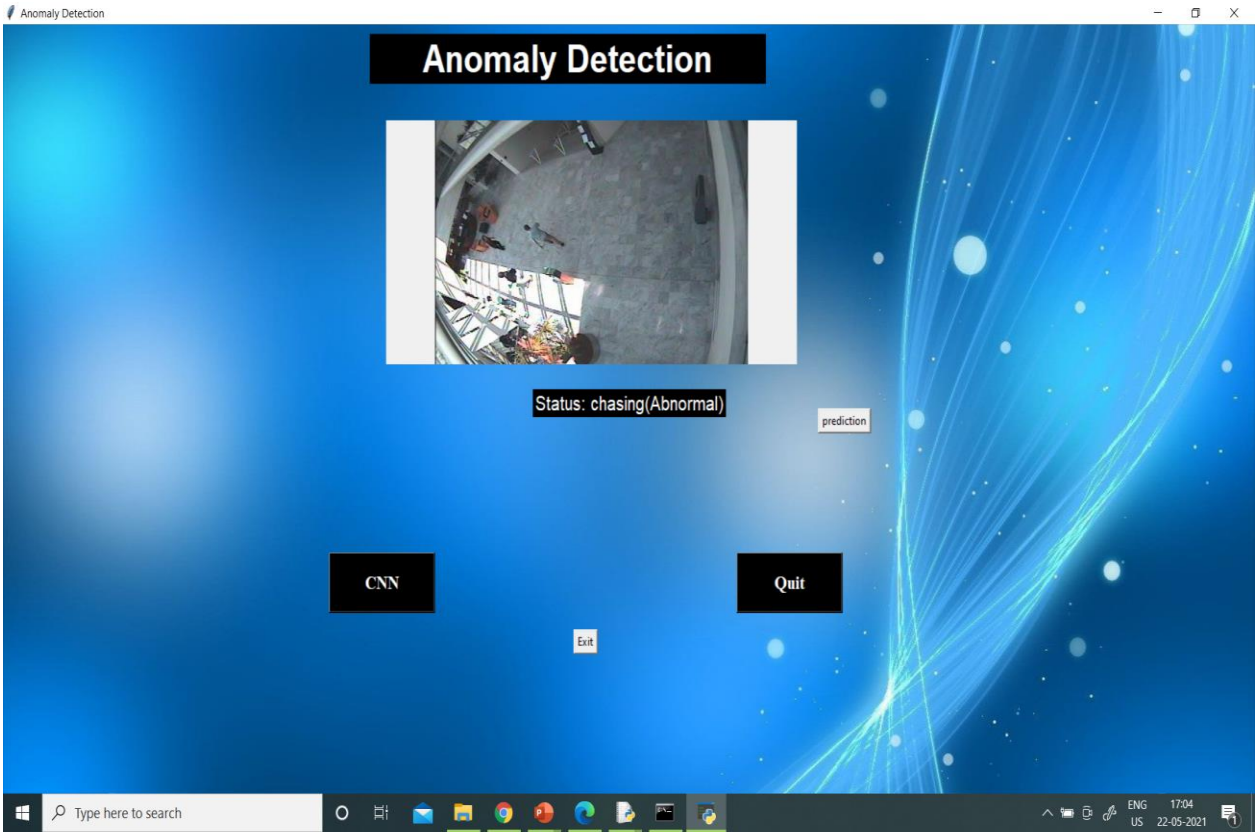
Experimental results

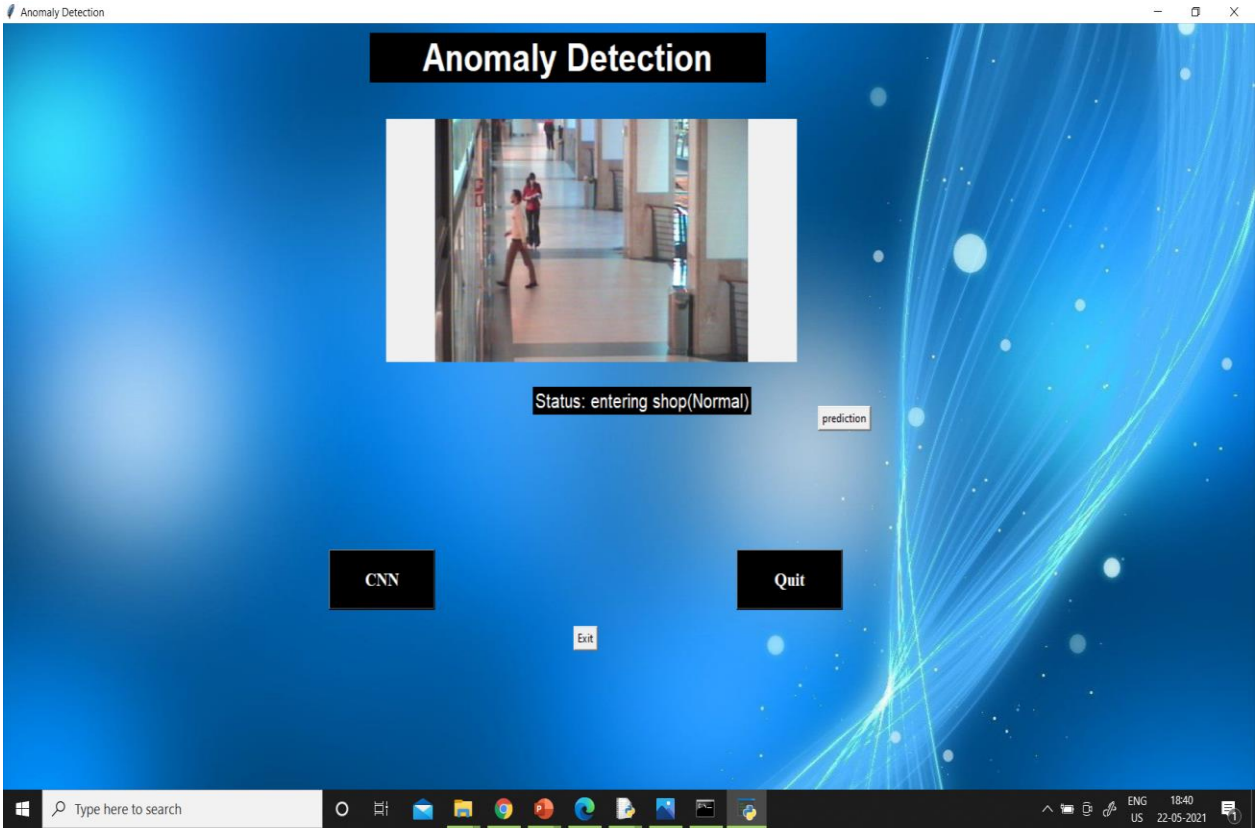


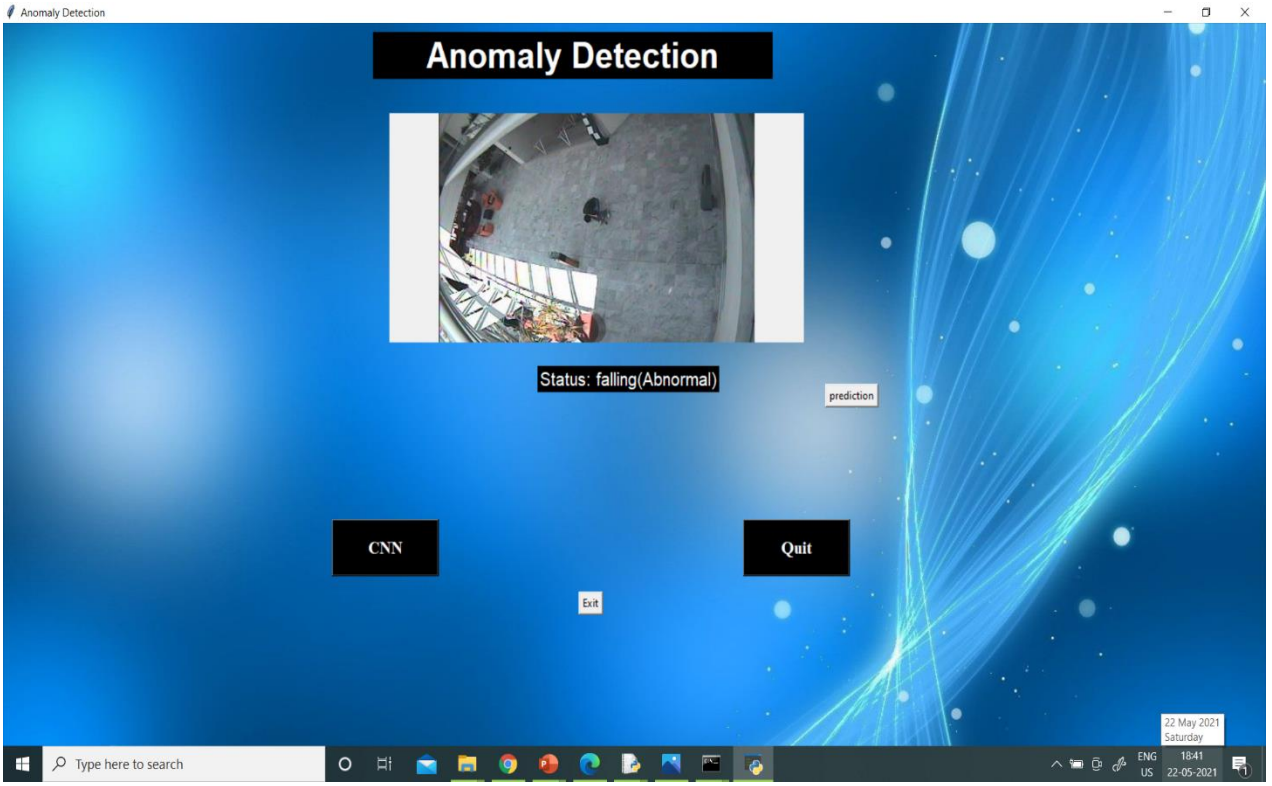


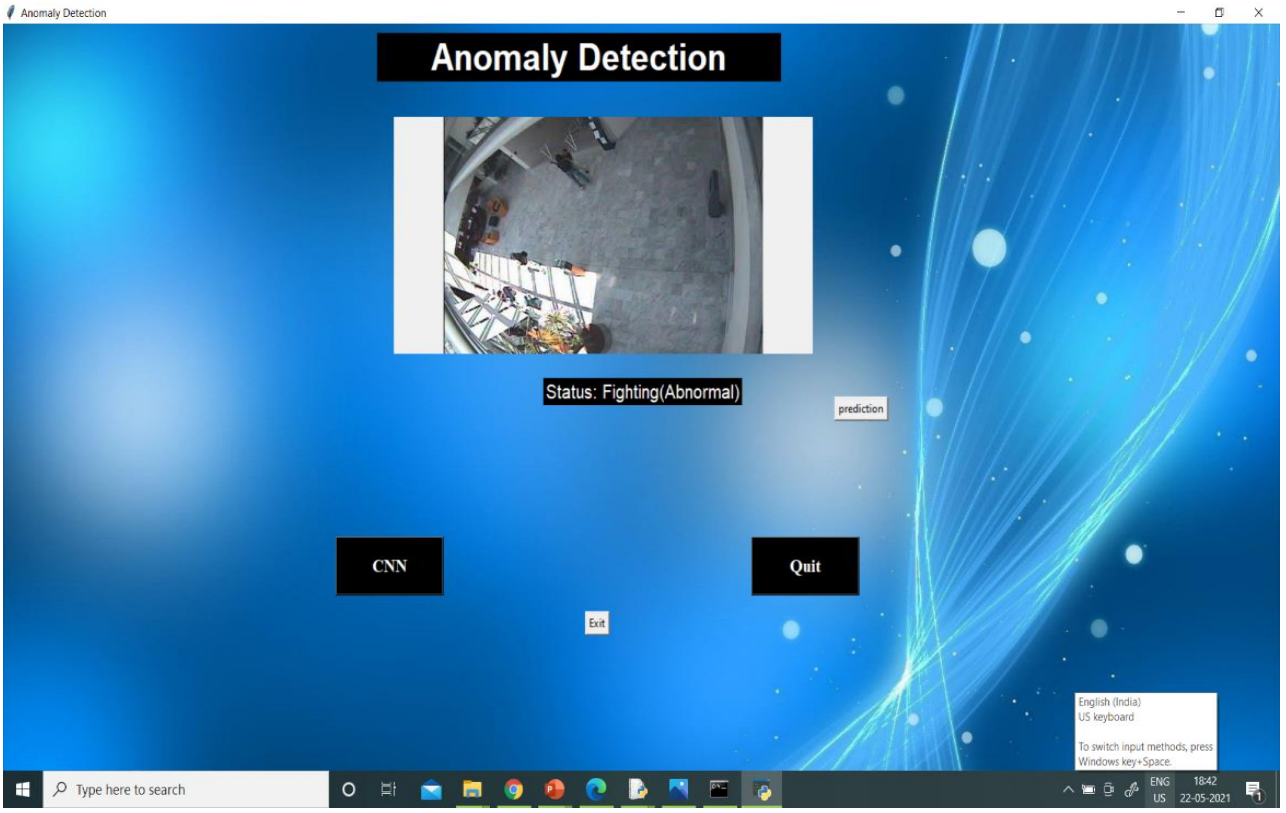

Anomaly Detection
Anomaly Detection
Click below to choose picture for testing ....
Get Photo
CNN
Quit
Type here to search
ENG
US
17:04
22-05-2021

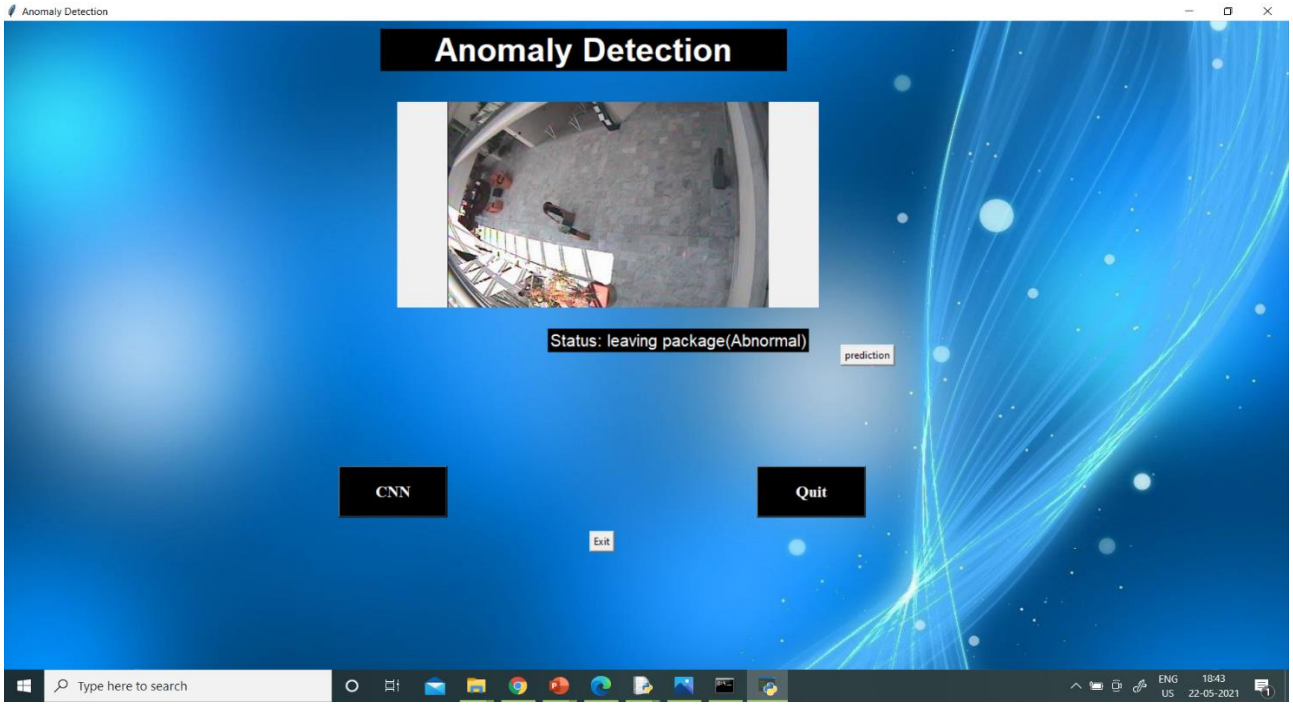

Anomaly Detection
Anomaly Detection
Analyse Image
CNN
Quit
Type here to search
ENG
US
17:04
22-05-2021

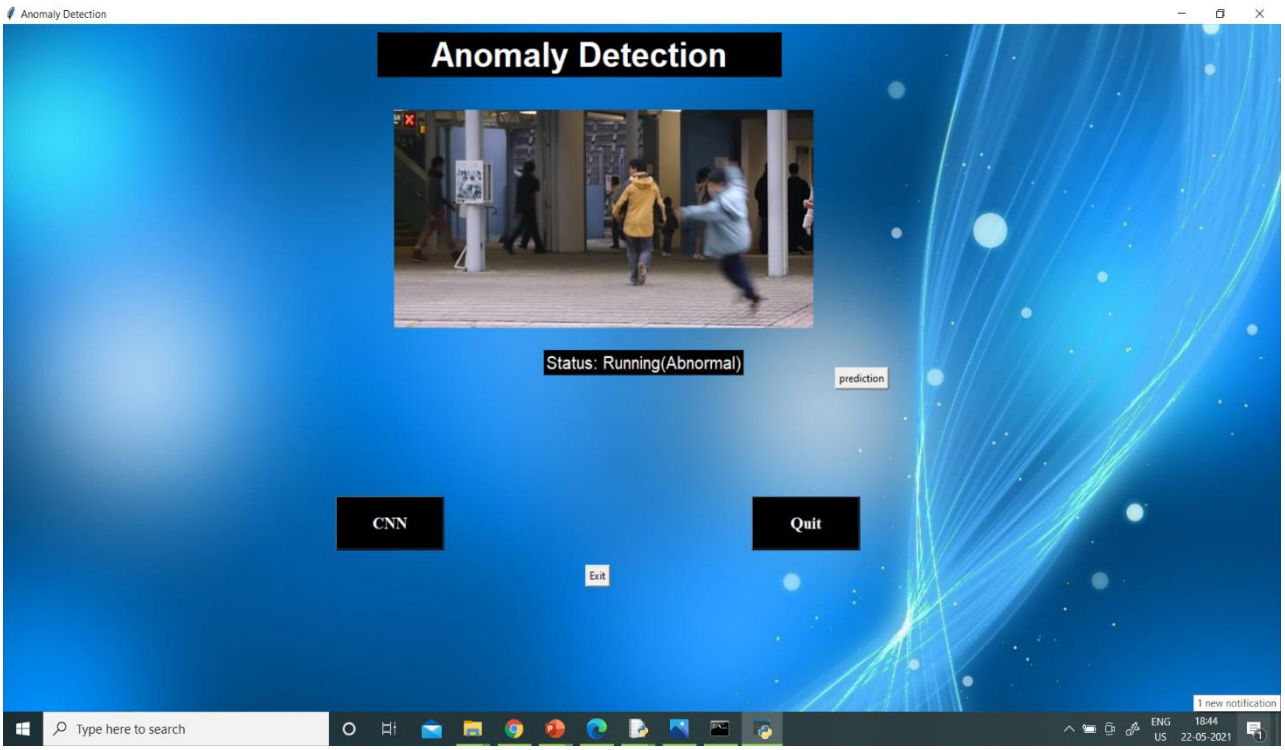

Anomaly Detection
Anomaly Detection
Status: chasing(Abnormal)
prediction
CNN
Quit
Exit
Type here to search
ENG
US
17:04
22-05-2021

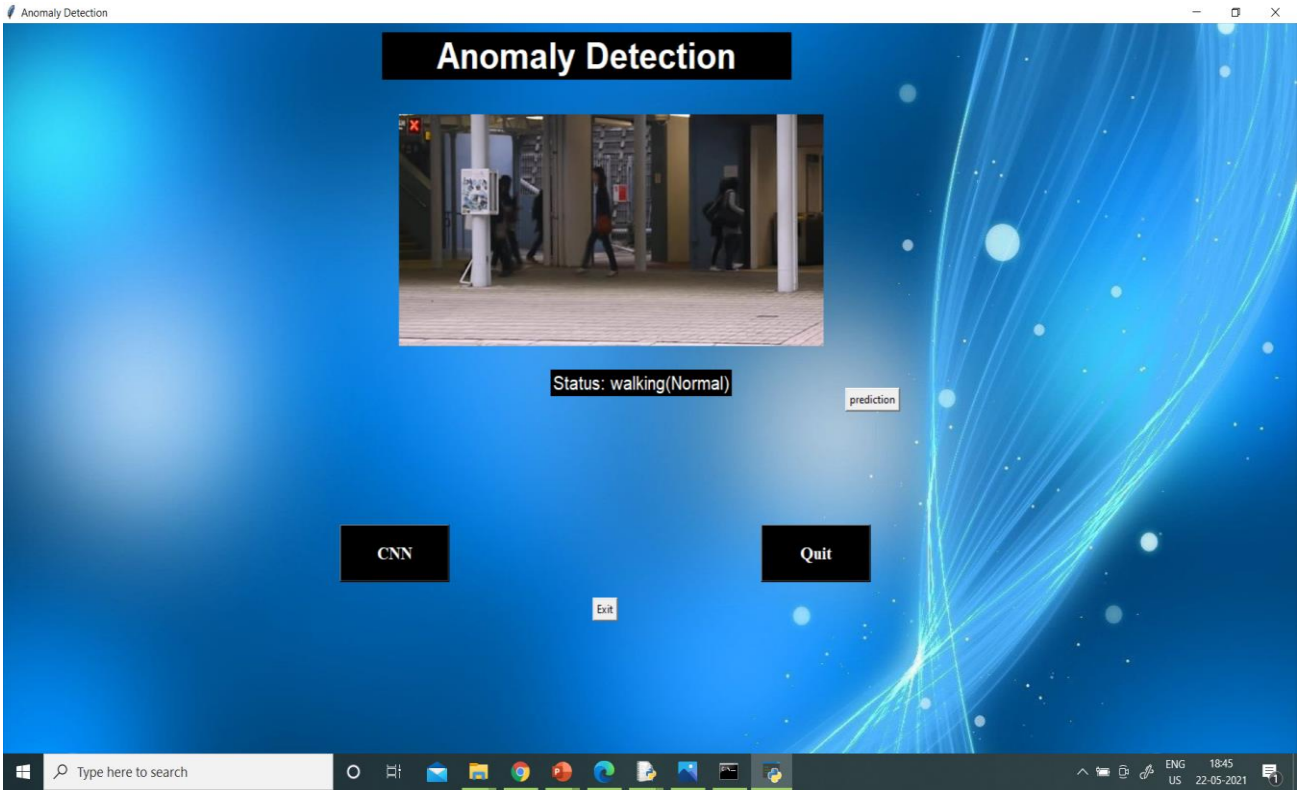

Anomaly Detection
Anomaly Detection
Status: entering shop(Normal)
prediction
CNN
Quit
Exit
Type here to search
ENG
US
18:40
22-05-2021


Anomaly Detection
Anomaly Detection
Status: falling(Abnormal)
prediction
CNN
Quit
Exit
Type here to search
22 May 2021
Saturday
ENG
US
18:41
22-05-2021


Anomaly Detection
Anomaly Detection
Status: Fighting(Abnormal)
prediction
CNN
Quit
Exit
English (India)
US keyboard
To switch input methods, press
Windows key+Space.
Type here to search
ENG
US
18:42
22-05-2021


Anomaly Detection
Anomaly Detection
Status: leaving package(Abnormal)
prediction
CNN
Quit
Exit
Type here to search
ENG
US
18:43
22-05-2021


Anomaly Detection
Anomaly Detection
Status: Running(Abnormal)
prediction
CNN
Quit
Exit
1 new notification
Type here to search
ENG
US
18:44
22-05-2021


Anomaly Detection
Anomaly Detection
Status: walking(Normal)
prediction
CNN
Quit
Exit
Type here to search
ENG
US
18:45
22-05-2021

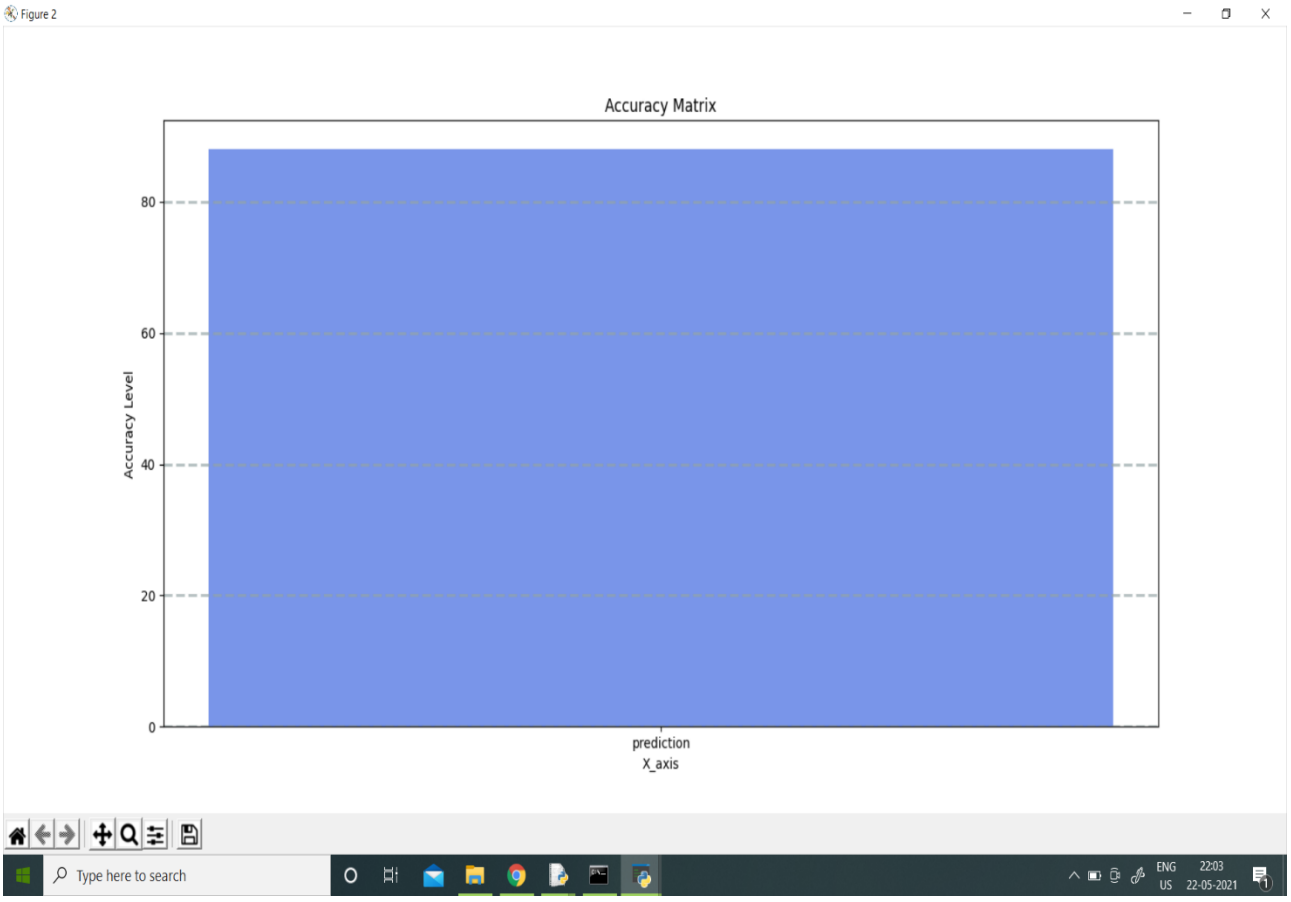


Figure 2
Accuracy Matrix
Accuracy Level
0
20
40
60
80
prediction
X_axis
Type here to search
ENG
US
22:03
22-05-2021

CONCLUSION AND FUTURE WORK

The novel method proposed by the paper, gives us a system where the human activity is recognized and anomaly detection is also performed, as any system that is designed the performance objective of the design is that, it has to be 100% efficient, but as it is a process of exploring and building the better versions in the future, this paper also has a few segments that can be improved in the future work to be done

- It is clear that the proposed new framework will be able to detect unusual events such as overcrowding situation, fighting or any abnormal activity in an image.
- The need of developing such security system is the increasing day by day for much secure society where people can lead their day to day lives without worrying.
- This proposed framework could be helpful to enhance many security systems, for different applications like ATM's, banks, malls, parking, traffic etc.
- The systems can be tailored for further future work were taking input as videos.
- This model can train and tested with different other datasets also for increasing accuracy and efficiency of the model.
- Can use this model with different machine learning models.
- Add more different activities also.

REFERENCES

[1] S. Ojha and S. Sakhare, "Image processing techniques for object tracking in video surveillance-a survey," in Pervasive Computing (ICPC), 2015 International Conferenceon. IEEE, 2015, pp. 1–6.

[2] K. A. Joshi and D. G. Thakore, "A survey on moving object detection and tracking in video surveillance system," International Journal of Soft Computing and Engineering, vol. 2, no. 3, pp. 44–48, 2012.

[3] G. L. Foresti, "Object recognition and tracking for remote video surveillance," IEEE Transactions on Circuits and Systems for Video Technology, vol. 9, no. 7, pp. 1045–1062, 1999.

[4] A. Treuille, S. Cooper, and Z. Popovic, "Continuum crowds," ´ ACM Transactions on Graphics (TOG), vol. 25, no. 3, pp. 1160–1168, 2006.

[5] A. Johansson, D. Helbing, H. Z. Al-Abideen, and S. Al-Bosta, "From crowd dynamics to crowd safety: a video-based analysis," Advances in Complex Systems, vol. 11, no. 04, pp. 497–527, 2008.

[6] X. Wang, X. Ma, and W. E. L. Grimson, "Unsupervised activity perception in crowded and complicated scenes using hierarchical bayesian models," IEEETransactions on pattern analysis and machine intelligence, vol. 31, no. 3, pp. 539–555, 2009.

[7] M. Marsden, K. McGuinness, S. Little, and N. E. O'Connor, "Resnetcrowd: A residual deep learning architecture for crowd counting, violent behaviour detection and crowd density level classification," in Advanced Video and Signal Based Surveillance (AVSS), 2017 14th IEEE International Conference on. IEEE, 2017, pp. 1–7.

[8] B. Pang, L. Lee, and S. Vaithyanathan, "Thumbs up?: sentiment classification using machine learning techniques," in Proceedings of the ACL02 conference on Empirical methods in natural language processingVolume 10. Association for Computational Linguistics, 2002, pp. 79– 86.

[9] M. Narwaria and W. Lin, “Svd-based quality metric for image and video using machine learning,” IEEE Transactions on Systems, Man, and Cybernetics, PartB (Cybernetics), vol. 42, no. 2, pp. 347–364, 2012.

[10] 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,” pp. 1–9, 2017.