AI ARENA

Enhanced Safety

AI survelliance system wiiht HAR, weapon and fire detection

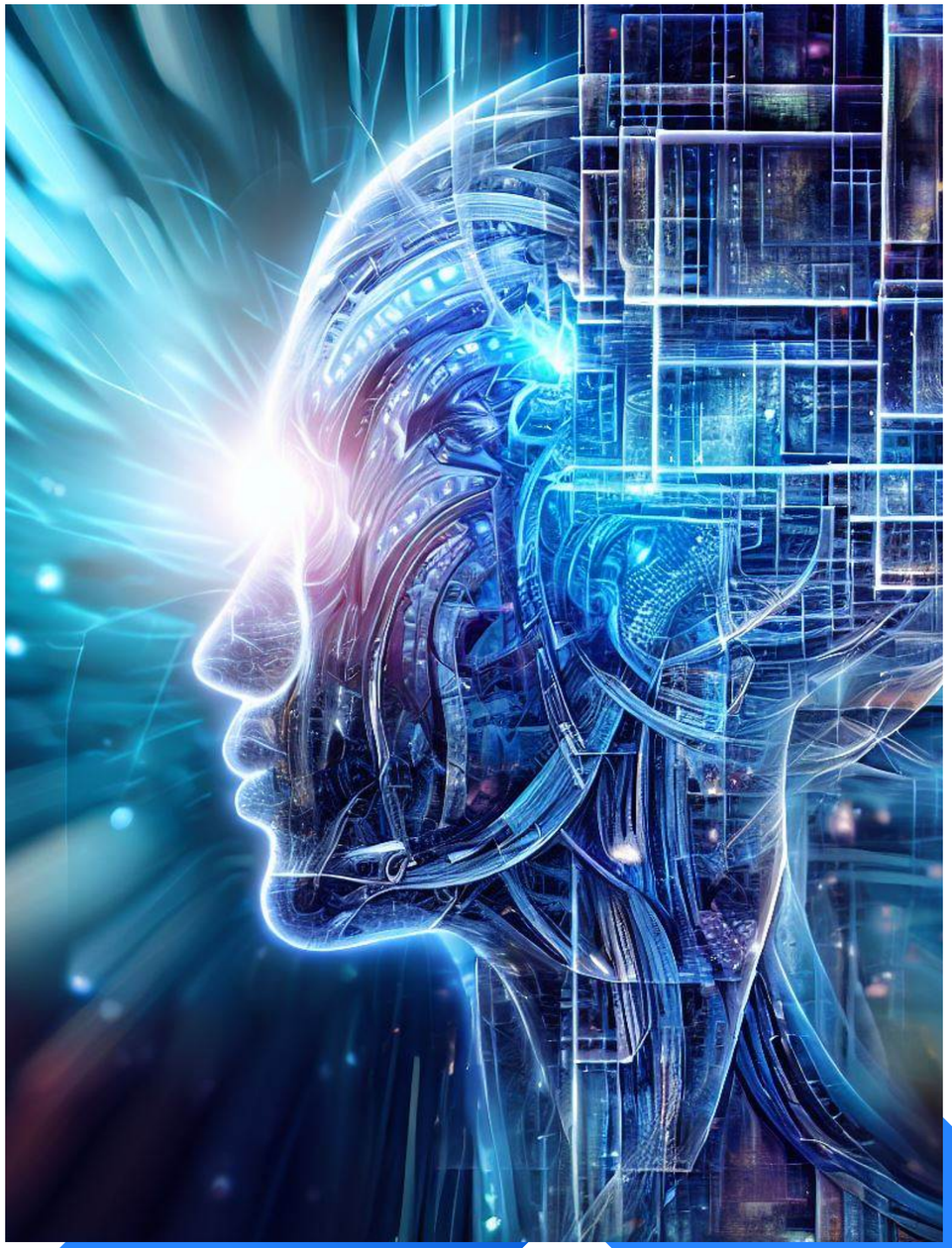
COMPANY GROUP

Table of Content

INTRODUCTION

AI Surveillance and Safety System

With increasing human population density. It's becoming increasingly difficult to ensure security of everyone manually. So I tried to develop a Surveillance system that requires least human interaction. It can track various human activities and emotions and can also alert if there is a fire or some kind of weapon or activity in the frame. It can be used tfor fast action in case of emergecy.



Concepts

Convolutional Neural Network

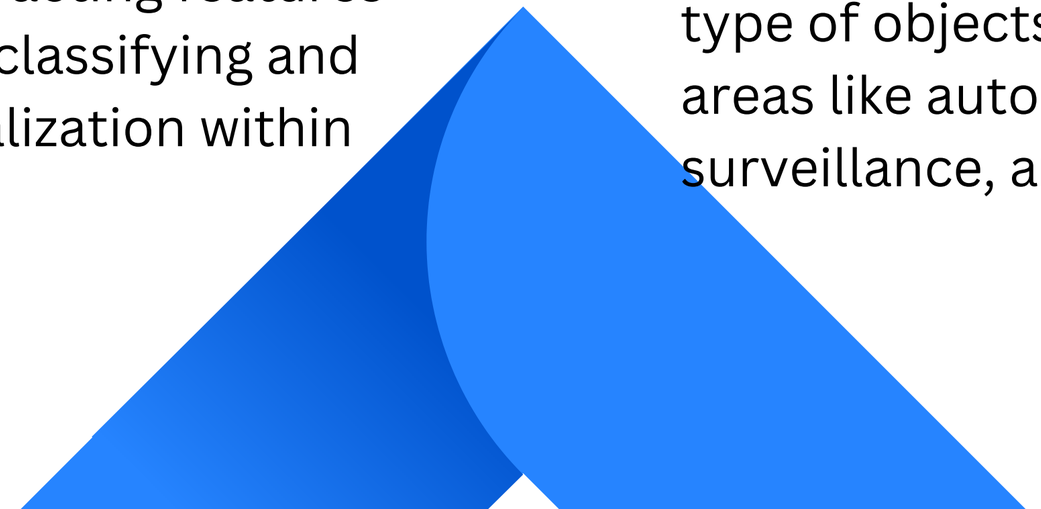
It is a type of deep learning model designed for processing and analyzing visual data. It utilizes convolutional layers to extract meaningful features from images and is commonly used in applications like image classification, object detection, and facial recognition.

Region-based Convolutional Neural Network

RCNN stands for Region-based Convolutional Neural Network. It is a popular deep learning model used for object detection in computer vision tasks. RCNN operates by proposing regions of interest, extracting features using a CNN, and then classifying and refining the object localization within those regions.

Object recognition,

Object recognition, a computer vision task, involves identifying and classifying objects in images or videos. It employs deep learning models like CNNs to extract features and make predictions about the presence and type of objects. It finds applications in areas like autonomous vehicles, surveillance, and augmented reality.



Impact

Cheap surviellance

Smart surviellance removes the need of human supervision which eliminates a lot of running expense for a firm

Faster response and disaster prevention

With AI detection we can detect any suspicious activity or an hazard faster as it removes the tendency of careless errors that might be caused by humans

Timeline

5/7/23-
8/7/23

Researching about the tech stack required

9/7/23-
11/7/23

Finding and analysing datasets required for training models

12/7/23-
14/7/23

Writing and debugging code and testing various models

15/7/23

Final reviewing and edititng

Code Workflow

```
pip install opencv
pip install numpy
pip install keras
pip install tensorflow
pip install pandas
pip install fer
import numpy as np
import argparse
import imutils #makes basix image processing resizing etc a lot easier
import sys
import cv2
import os
from fer import FER
from PIL import Image
from keras.models import Sequential
from keras.layers.core import Dense, Dropout, Activation, Flatten
from keras.layers.convolutional import Convolution2D, MaxPooling2D
from keras.utils import np_utils
import tensorflow as tf
from keras.utils.image_utils import  img_to_array
from keras.applications import xception
```

Importing and including modules

First we include and download all the required modules and libraries for the code

Code Workflow

Importing all the required models

FWe import all the required models in the main file

```
act_lst=open("Actions.txt").read().strip().split("\n")
#getting labels for all possible activities


# loading the kinetic model
md = cv2.dnn.readNet("resnet-34_kinetics.onnx")
face_haar_cascade = cv2.CascadeClassifier(cv2.data.haarcascades + 'haarcascade_frontalface_default.xml')
emotion=FER(mtcnn=True)
weapon=tf.keras.models.load_model("weapon_detection.h5")
# accuracy above 90%
fire=tf.keras.models.load_model("fd.h5")
# accuracy near 90%
print(type(fire))
```

```
def                                  ):
    image_hsv = cv2.cvtColor(image, cv2.COLOR_BGR2HSV)

    lower_hsv = np.array([0,0,250])
    upper_hsv = np.array([250,255,255])

    mask = cv2.inRange(image_hsv, lower_hsv, upper_hsv)
    kernel = cv2.getStructuringElement(cv2.MORPH_ELLIPSE, (11,11))
    mask = cv2.morphologyEx(mask, cv2.MORPH_CLOSE, kernel)

    return mask

#image segmentation function
def segment_image(image):
    mask = create_mask_for_plant(image)
    output = cv2.bitwise_and(image, image, mask = mask)
    return output/255

#sharpen the image
def sharpen_image(image):
    image_blurred = cv2.GaussianBlur(image, (0, 0), 3)
    image_sharp = cv2.addWeighted(image, 1.5, image_blurred, -0.5, 0)
    return image_sharp

def prepro(im):
    img=im.copy()
    img=cv2.resize(img,(255,255))
    img=img_to_array(img)
```

Code Workflow

Defining preprocessing functions

Making preprocessing functions
for every model requirement

Code Workflow

Live video Input and processing

We take input using opencv and process images at required frame rate and predict them on each model and show the output

```
cnt=0
cap=cv2.VideoCapture(0)
to_process=[]
original=[]
label="None"
while True:

    #now since we are video processing we would need continuous set of images instead of single frame by frame images
    we will need to convert it to blob

    # lets take 16 frames at a time to make blob

    bl,img=cap.read()
    img=cv2.flip(img,1)
   # cv2.imshow('wnd',img)
    if bl==False:
        print("Unable to get input. Exiting code")
        cv2.destroyAllWindows()
        break


    x=imutils.resize(img,400)
    to_process.append(x)
    if(len(to_process)==16):
        blob=cv2.dnn.blobFromImages(to_process,1.0,(112,112), (114.7748, 107.7354, 99.4750),swapRB=True, crop=True)
        blob = np.transpose(blob, (1, 0, 2, 3))
        blob = np.expand_dims(blob, axis=0)
```

Features

Resnet-34_kinetic model

This model is a classification model trained on CNN that can classify various human activities

```
# loading the kinetic model
md = cv2.dnn.readNet("resnet-34_kinetics.onnx")
face_haar_cascade = cv2.CascadeClassifier(cv2.data.haarcascades +
'haarcascade_frontalface_default.xml')
emotion=FER(mtcnn=True)
weapon=tf.keras.models.load_model("weapon_detection.h5")
# accuracy above 90%
fire=tf.keras.models.load_model("fd.h5")
# accuracy near 90%
print(type(fire))
```

Features

face Haar Cascade

It is a predefined opencv object detection model that detects and return's coordinates of faces

```
# loading the kinetic model
md = cv2.dnn.readNet("resnet-34_kinetics.onnx")
face_haar_cascade = cv2.CascadeClassifier(cv2.data.haarcascades +
'haarcascade_frontalface_default.xml')
emotion=FER(mtcnn=True)
weapon=tf.keras.models.load_model("weapon_detection.h5")
# accuracy above 90%
fire=tf.keras.models.load_model("fd.h5")
# accuracy near 90%
print(type(fire))
```

Features

Emotion Detection

We use FER library which is a classification model used to calssify various emotions

```
# loading the kinetic model
md = cv2.dnn.readNet("resnet-34_kinetics.onnx")
face_haar_cascade = cv2.CascadeClassifier(cv2.data.haarcascades +
'haarcascade_frontalface_default.xml')
emotion=FER(mtcnn=True)
weapon=tf.keras.models.load_model("weapon_detection.h5")
# accuracy above 90%
fire=tf.keras.models.load_model("fd.h5")
# accuracy near 90%
print(type(fire))
```

Features

Weapon Detection

It is an object detection model build on rcnn used to detect knifes.

```
# loading the kinetic model
md = cv2.dnn.readNet("resnet-34_kinetics.onnx")
face_haar_cascade = cv2.CascadeClassifier(cv2.data.haarcascades +
'haarcascade_frontalface_default.xml')
emotion=FER(mtcnn=True)
weapon=tf.keras.models.load_model("weapon_detection.h5")
# accuracy above 90%
fire=tf.keras.models.load_model("fd.h5")
# accuracy near 90%
print(type(fire))
```

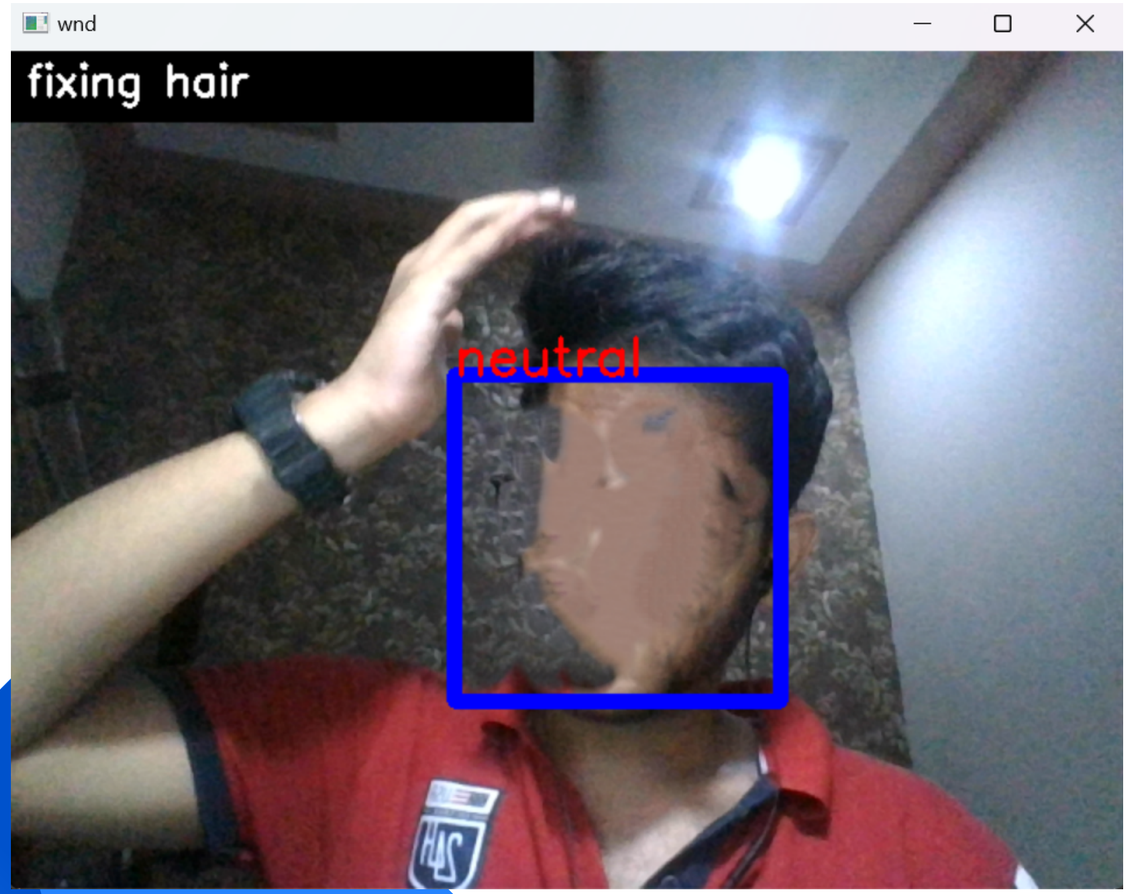
Features

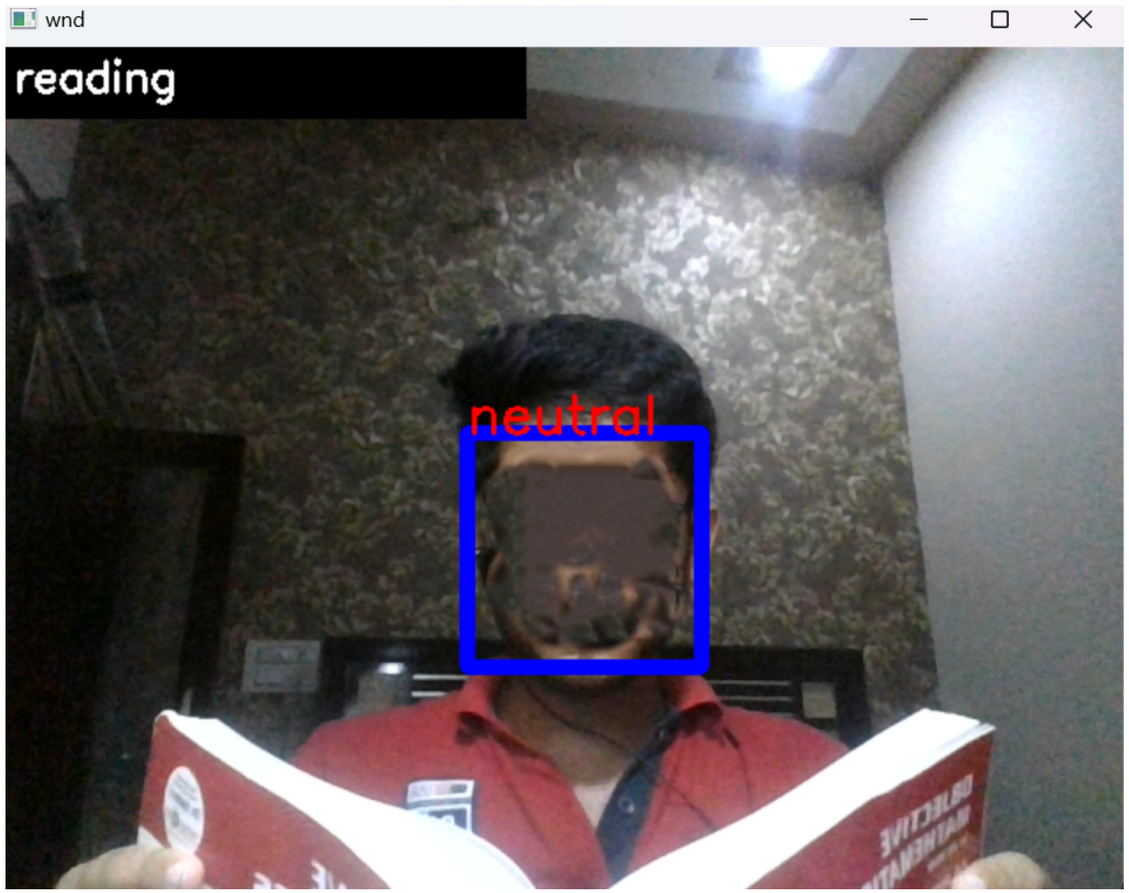
Fire Detection

It is an classification model build on cnn that can classify whether there is fire in image or not

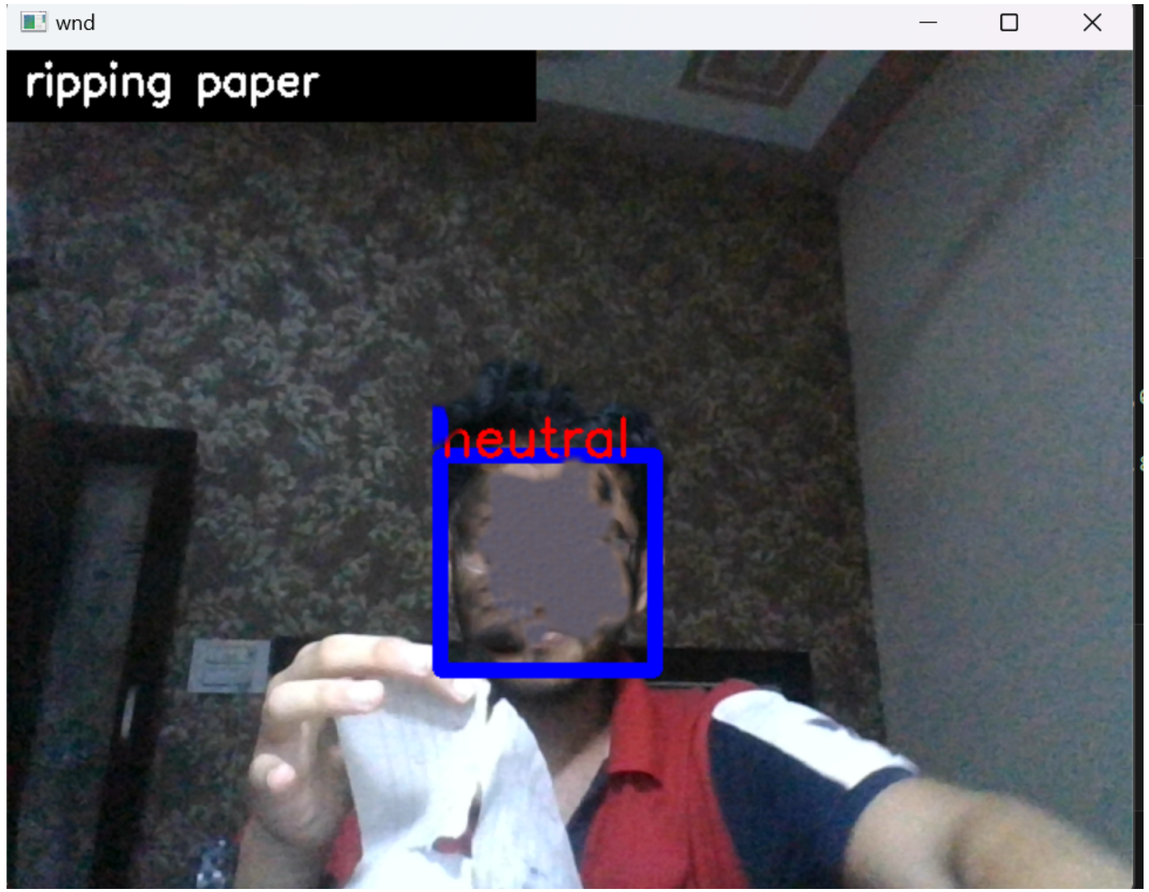
```
# loading the kinetic model
md = cv2.dnn.readNet("resnet-34_kinetics.onnx")
face_haar_cascade = cv2.CascadeClassifier(cv2.data.haarcascades +
'haarcascade_frontalface_default.xml')
emotion=FER(mtcnn=True)
weapon=tf.keras.models.load_model("weapon_detection.h5")
# accuracy above 90%
fire=tf.keras.models.load_model("fd.h5")
# accuracy near 90%
print(type(fire))
```

Output

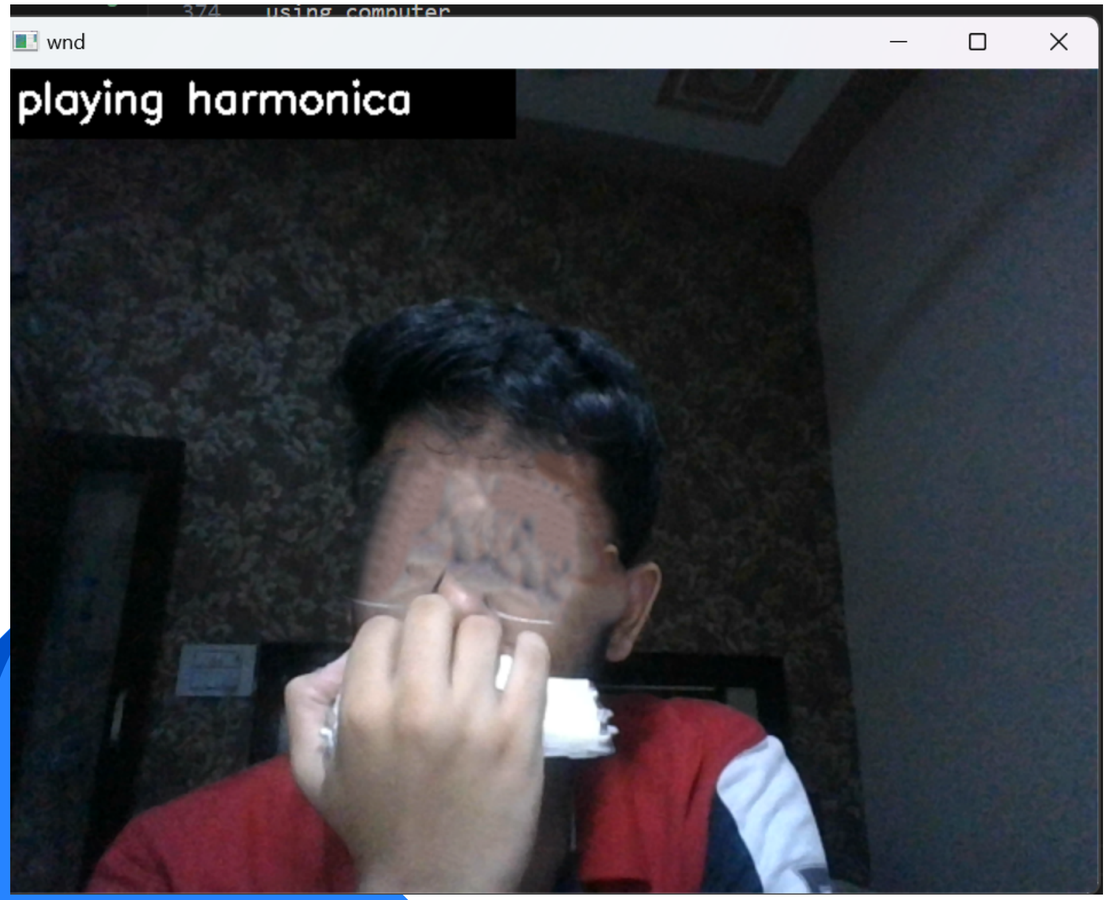


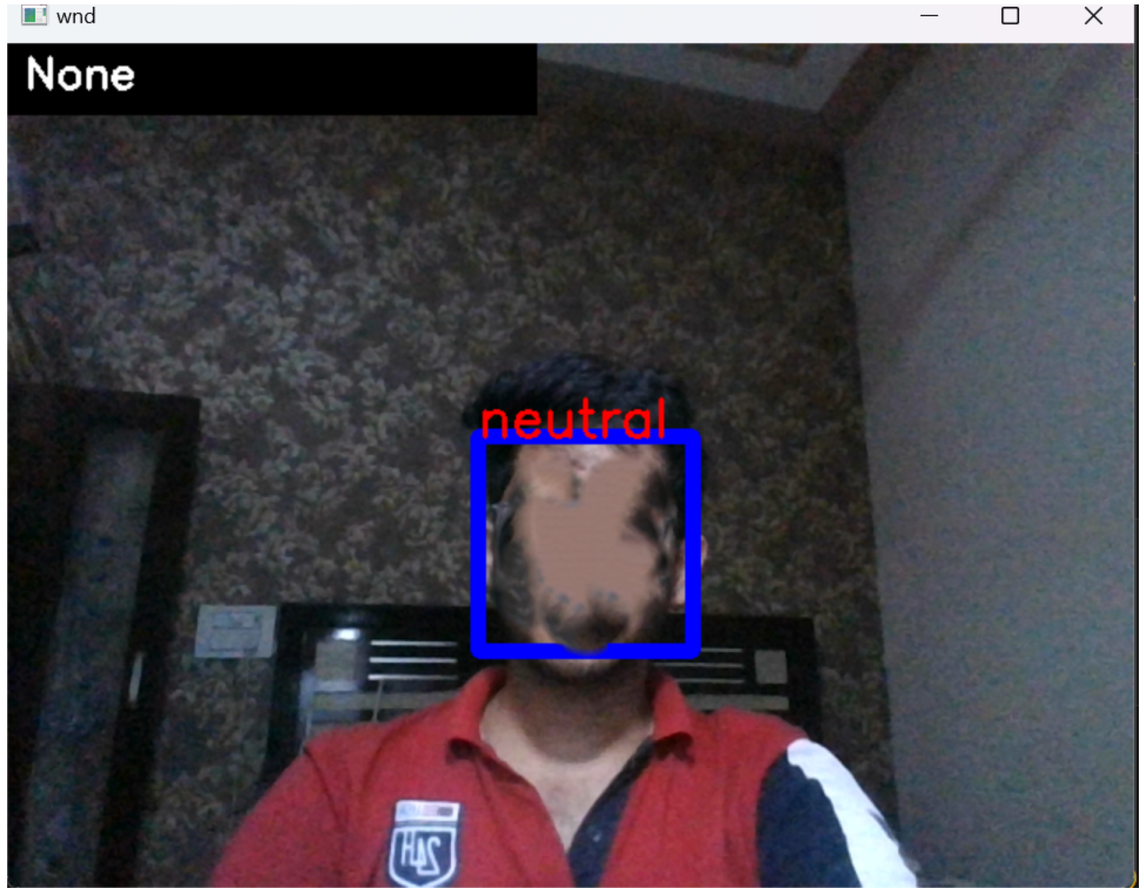


Output



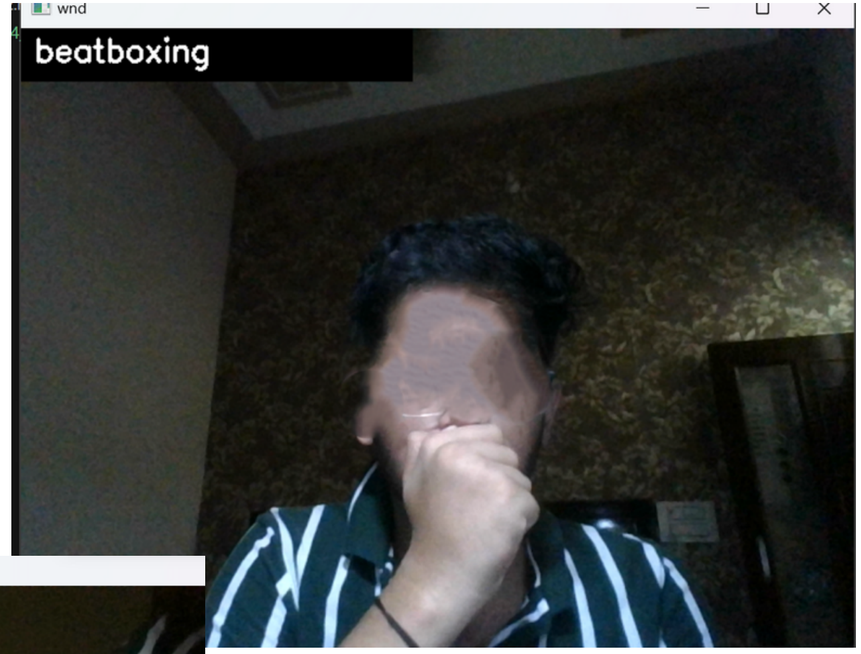
Output

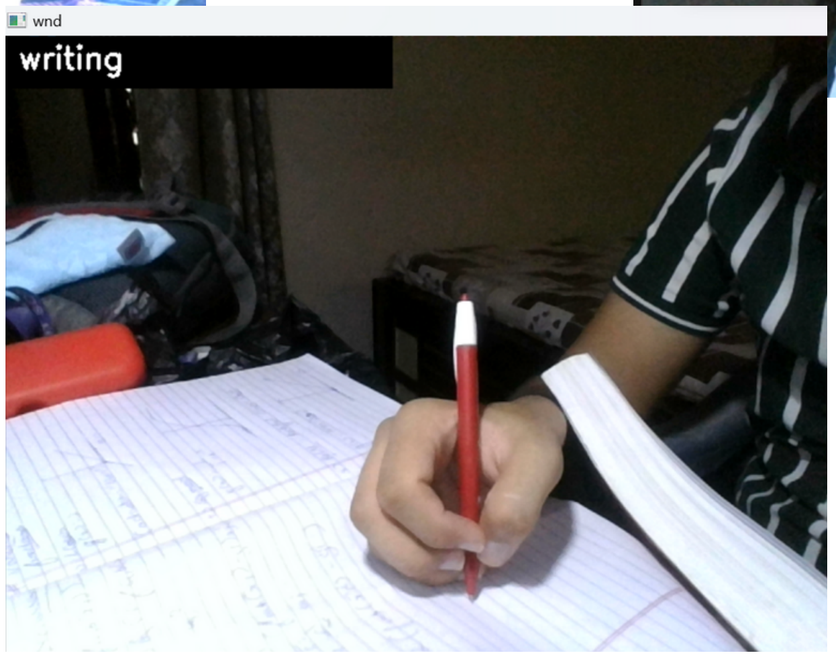




Output

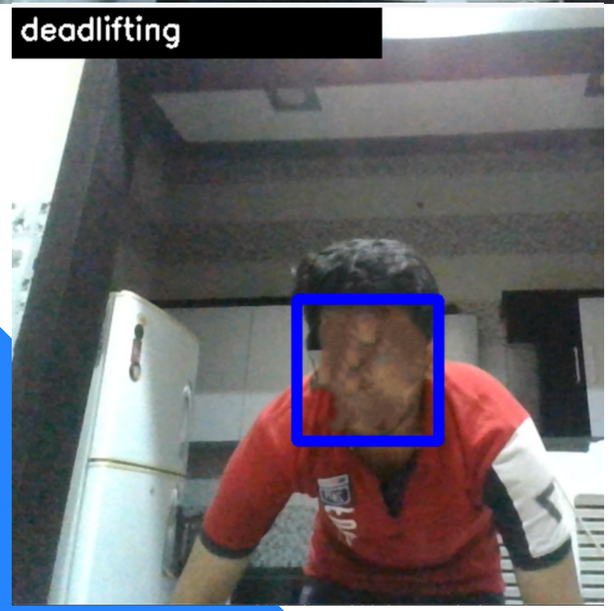

using remote controller (not gaming)

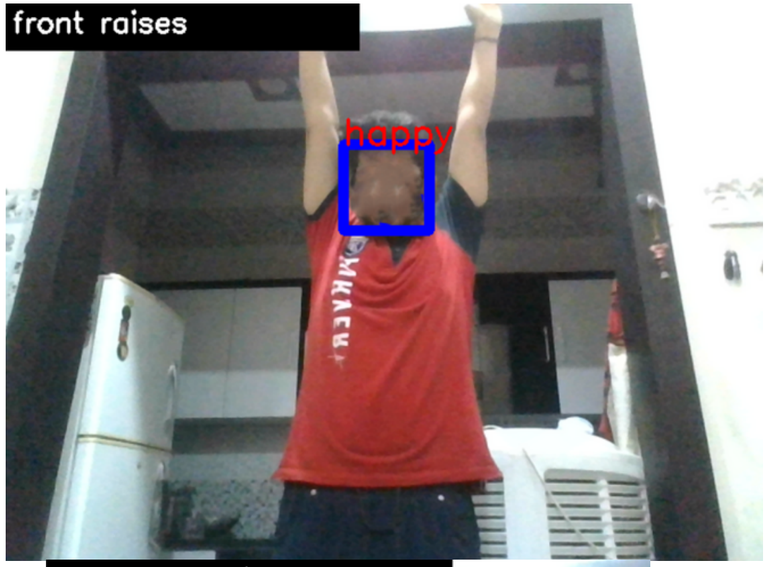

wnd
beatboxing


wnd
writing

Output

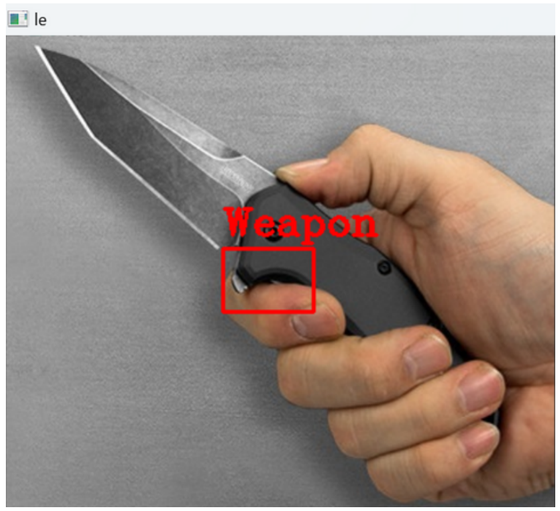

exercising arm
neutral

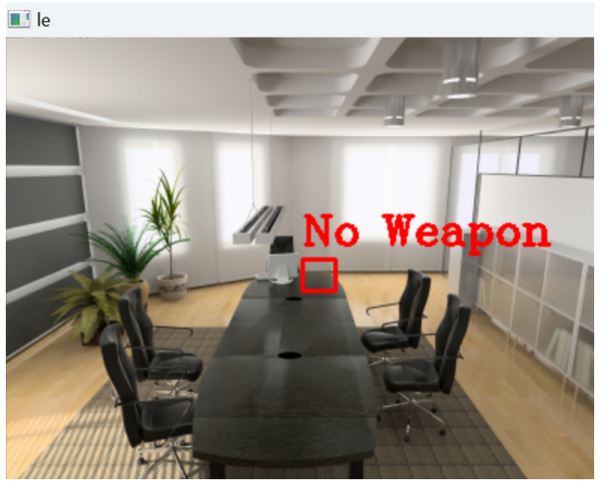

deadlifting

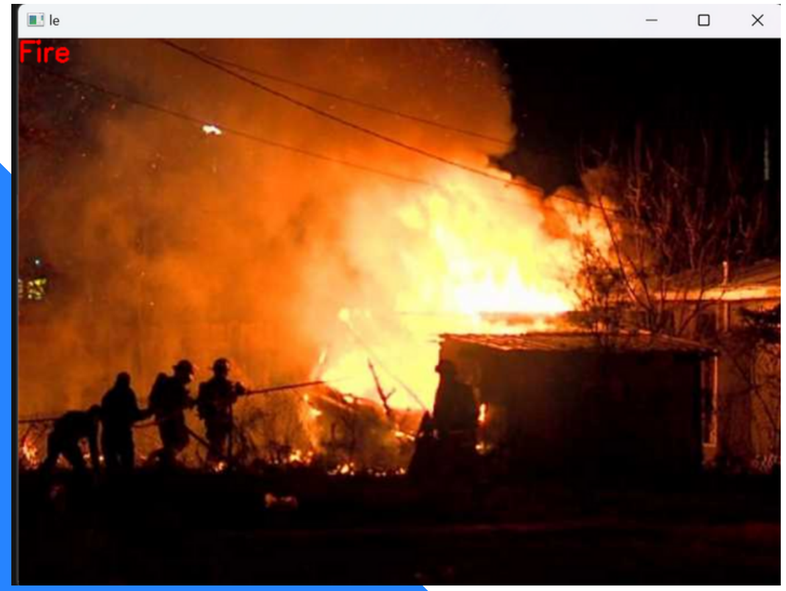

front raises
happy

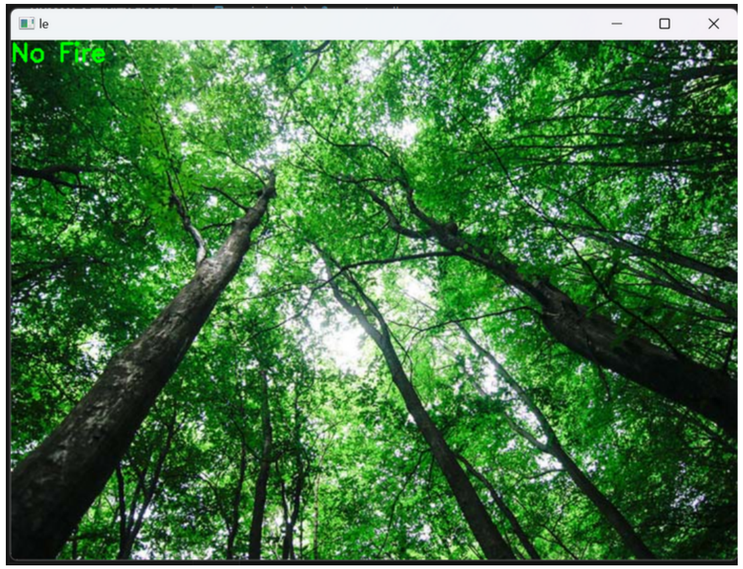

arm wrestling

Output









Thank You