**Laporan Final Project**

**Pengenalan Pola**

**Real Time**

**Object Pattern Detection**



**Oleh :**

Kholilul Rachman N.M (17081010055)

Sandy Rizkyando (17081010065)

**DOSEN PENGAMPU :**

## Dr. Basuki Rahmat, S.Si, M.T.

## 

**PROGRAM STUDI INFORMATIKA**

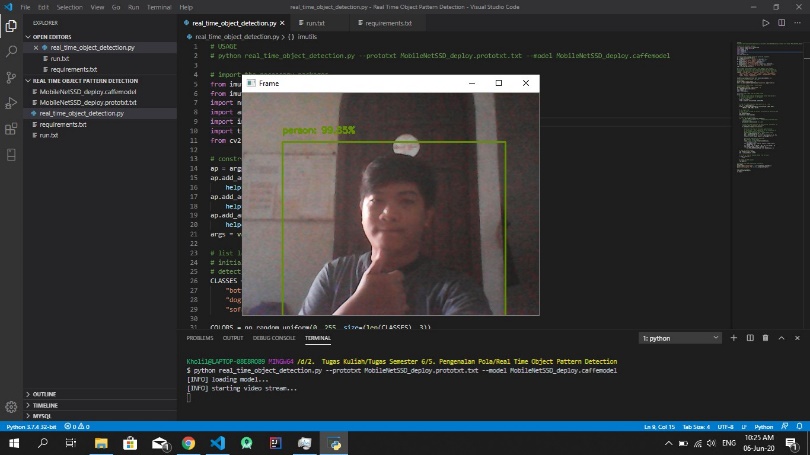
**FAKULTAS ILMU KOMPUTER**

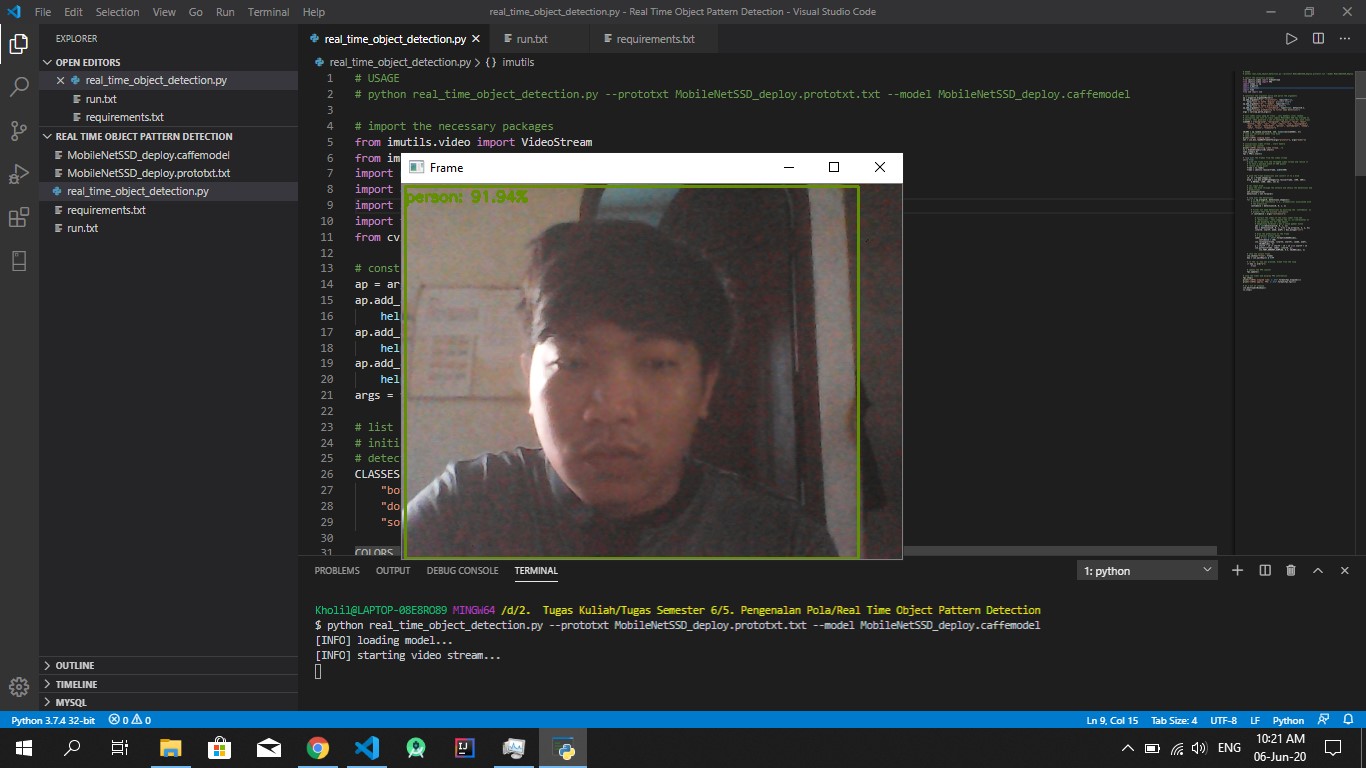
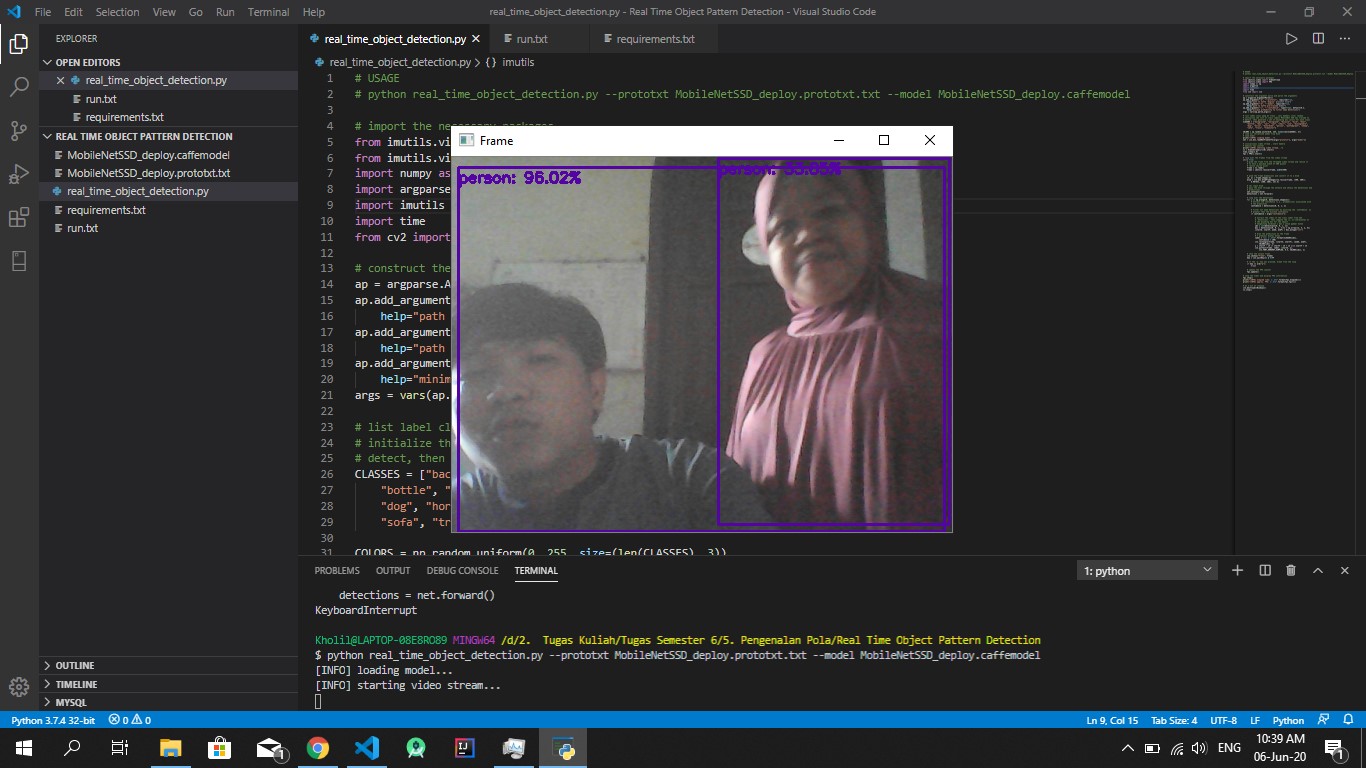
**UNIVERSITAS PEMBANGUNAN NASIONAL “VETERAN”**

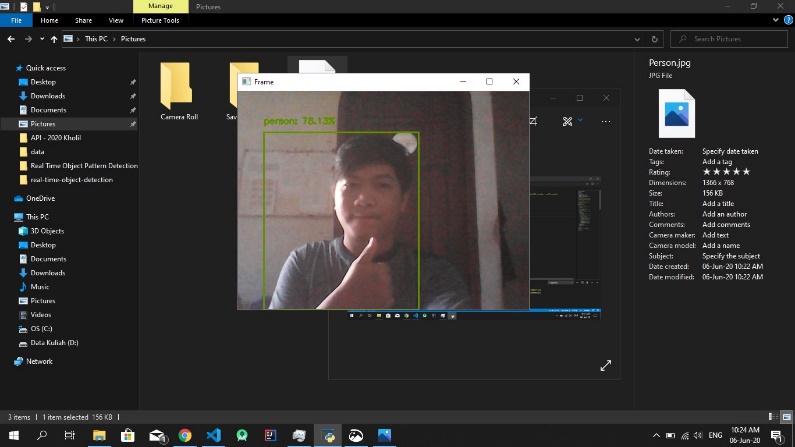
**JAWA TIMUR**

**2020**

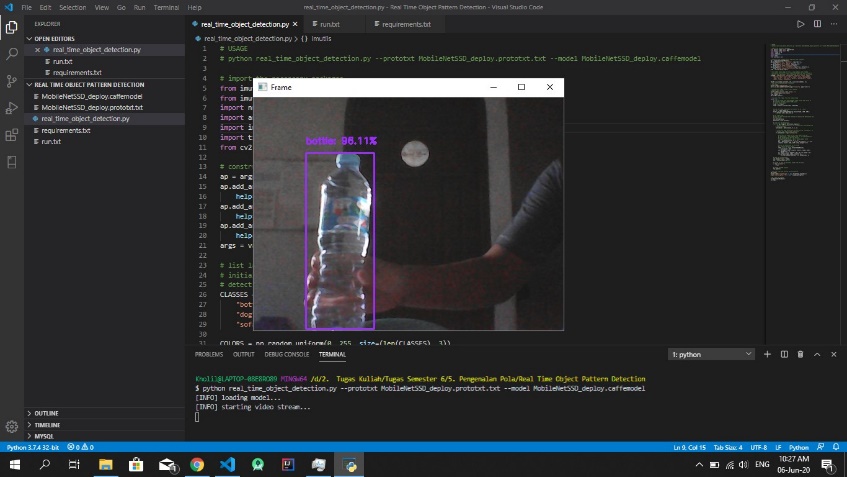
1. **Capture Demo Program**

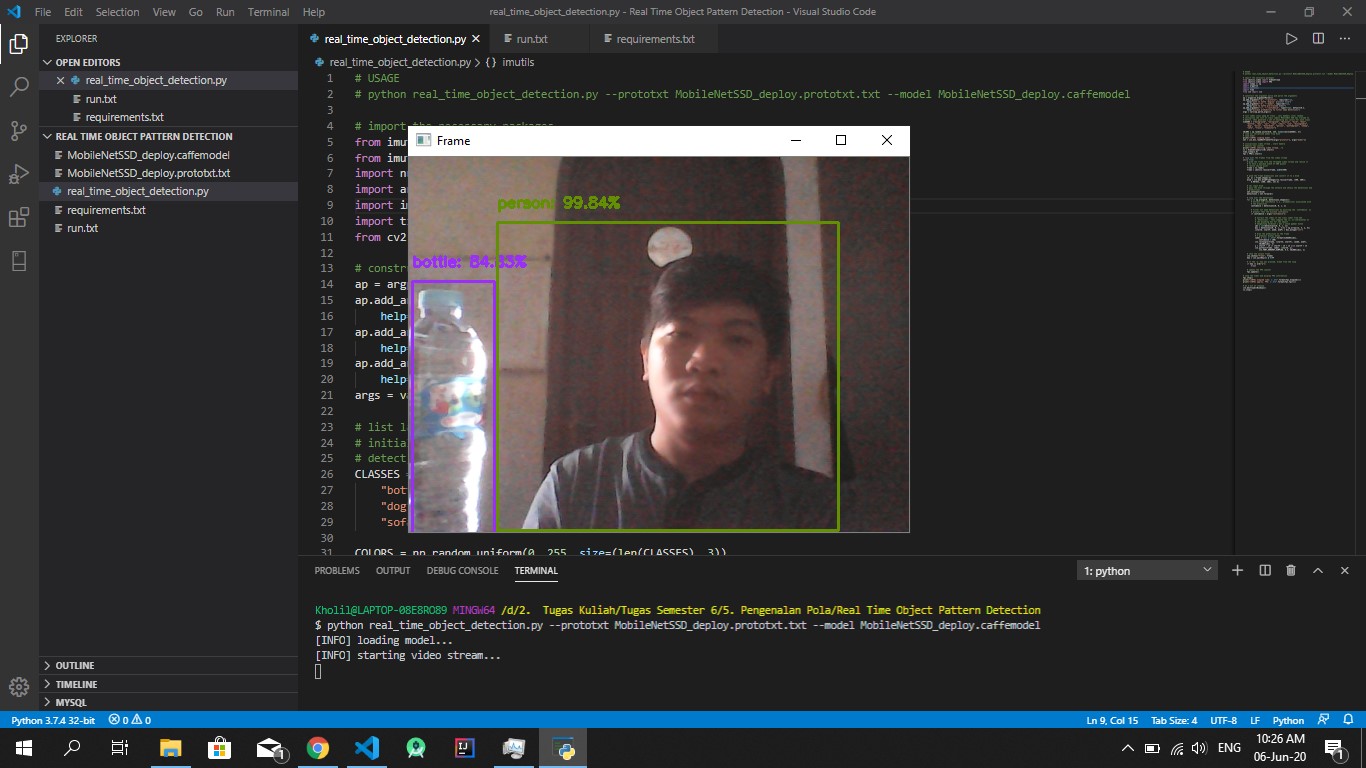




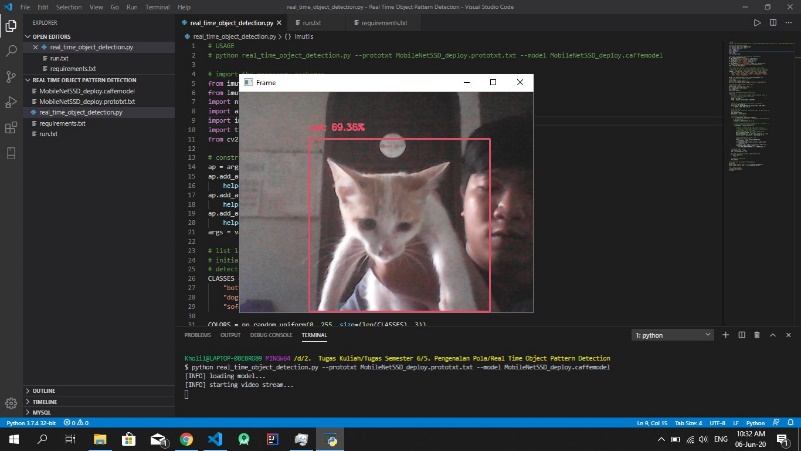
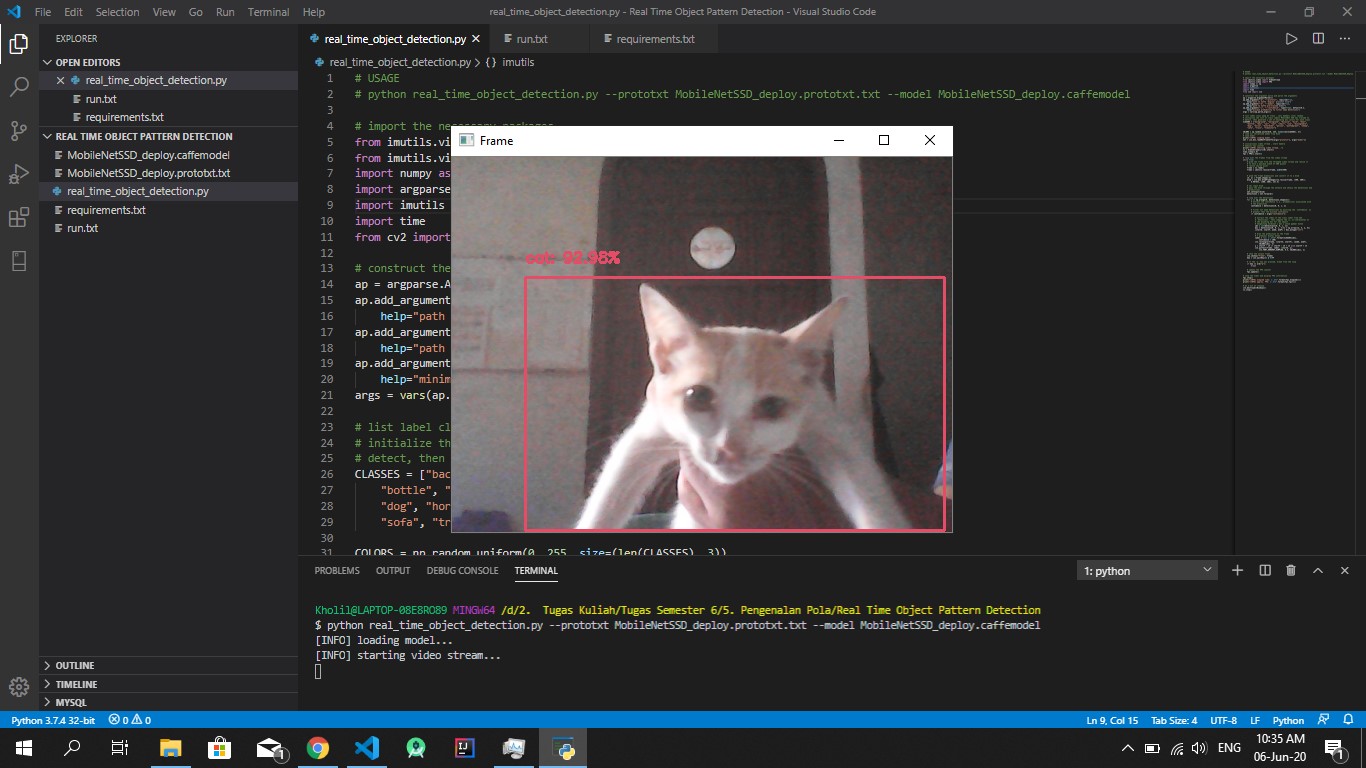


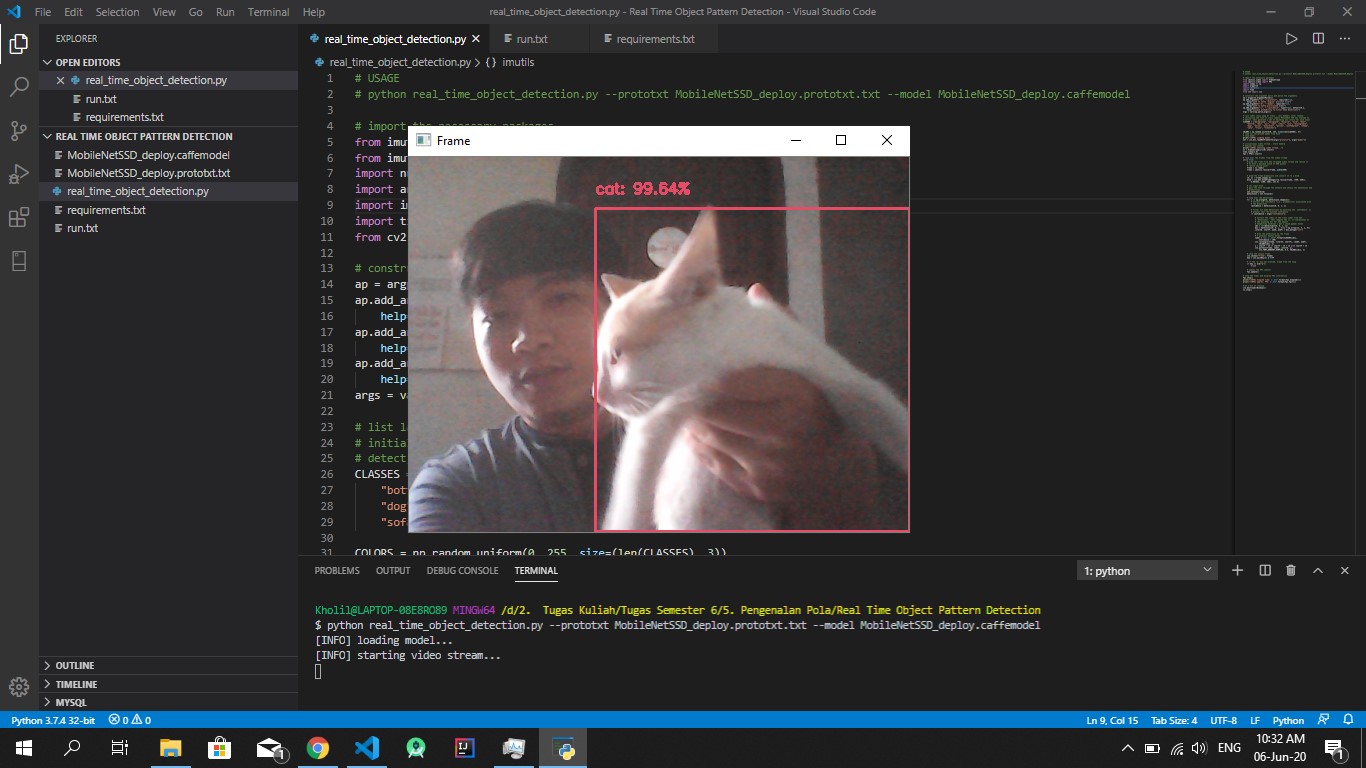
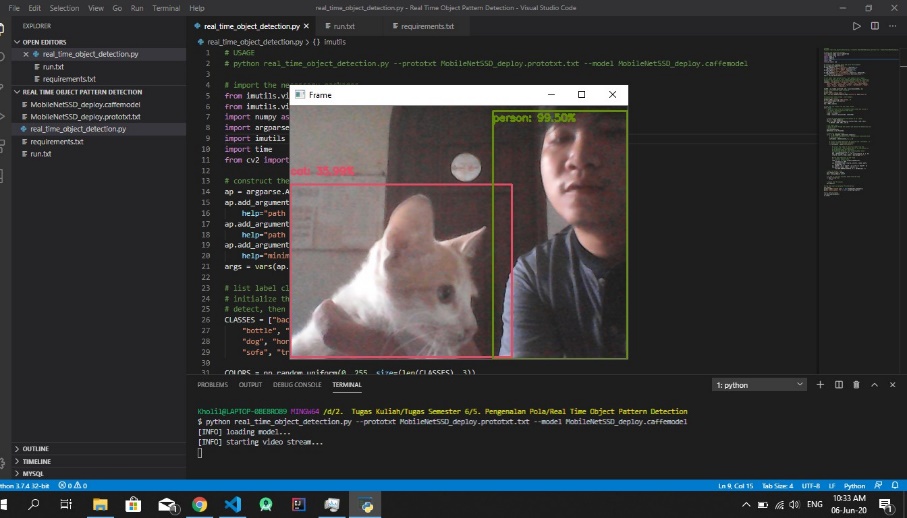
Gambar 1. Akurasi “Person”

****

****

Gambar 2. Akurasi “Bottle”

****

****

Gambar 3. Akurasi “Kucing”

1. **Penjelasan Kode Program**
2. **Kode dari file real\_time\_object\_detection.py**

|  |
| --- |
| # Penjelasan Kode Tiap baris  # Untuk Run Program = python real\_time\_object\_detection.py --prototxt MobileNetSSD\_deploy.prototxt.txt --model MobileNetSSD\_deploy.caffemodel  # import library  from imutils.video import VideoStream  from imutils.video import FPS  import numpy as np  import argparse  import imutils  import time  from cv2 import cv2  # bangun argumen parse dan parsing argumen parameter  ap = argparse.ArgumentParser()  ap.add\_argument("-p", "--prototxt", required=True,      help="path to Caffe 'deploy' prototxt file")  ap.add\_argument("-m", "--model", required=True,      help="path to Caffe pre-trained model")  ap.add\_argument("-c", "--confidence", type=float, default=0.2,      help="minimum probability to filter weak detections")  args = vars(ap.parse\_args())  # list label class yang di train , lalu memberi color random  # initialize dari daftar label kelas SSD Mobile Net dilatih untuk deteksi  # dari deteksi itu dihasilkan seperangkat warna kotak pembatas untuk setiap kelas  CLASSES = ["background", "aeroplane", "bicycle", "bird", "boat",      "bottle", "bus", "car", "cat", "chair", "cow", "diningtable",      "dog", "horse", "motorbike", "person", "pottedplant", "sheep",      "sofa", "train", "tvmonitor"]  COLORS = np.random.uniform(0, 255, size=(len(CLASSES), 3))  # load our serialized model from disk  # load model  print("[INFO] loading model...")  net = cv2.dnn.readNetFromCaffe(args["prototxt"], args["model"])  # inisialisasi video stream , start kamera  # memulai fps counter  print("[INFO] starting video stream...")  vs = VideoStream(src=0).start()  time.sleep(2.0)  fps = FPS().start()  # loop over the frames from the video stream  while True:      # grab the frame from the threaded video stream and resize it      # to have a maximum width of 400 pixels      # resize ukuran pixel      frame = vs.read()      frame = imutils.resize(frame, width=500)      # grab the frame dimensions and convert it to a blob      (h, w) = frame.shape[:2]      blob = cv2.dnn.blobFromImage(cv2.resize(frame, (300, 300)),          0.007843, (300, 300), 127.5)      # set input blob      # pass the blob through the network and obtain the detections and      # predictions      net.setInput(blob)      detections = net.forward()      # loop over the detections      for i in np.arange(0, detections.shape[2]):          # extract the confidence (i.e., probability) associated with          # the prediction          confidence = detections[0, 0, i, 2]          # filter out weak detections by ensuring the `confidence` is          # greater than the minimum confidence          if confidence > args["confidence"]:              # extract the index of the class label from the              # `detections`, then compute the (x, y)-coordinates of              # the bounding box for the object              # menghitung kordinat x dan y untuk gambar kotak              idx = int(detections[0, 0, i, 1])              box = detections[0, 0, i, 3:7] \* np.array([w, h, w, h])              (startX, startY, endX, endY) = box.astype("int")              # draw the prediction on the frame              # % prediksi diatas kotak              label = "{}: {:.2f}%".format(CLASSES[idx],                  confidence \* 100)              cv2.rectangle(frame, (startX, startY), (endX, endY),                  COLORS[idx], 2)              y = startY - 15 if startY - 15 > 15 else startY + 15              cv2.putText(frame, label, (startX, y),                  cv2.FONT\_HERSHEY\_SIMPLEX, 0.5, COLORS[idx], 2)      # show the output frame      cv2.imshow("Frame", frame)      key = cv2.waitKey(1) & 0xFF      # if the `q` key was pressed, break from the loop      if key == ord("q"):          break      # update the FPS counter      fps.update()  # stop the timer and display FPS information  fps.stop()  print("[INFO] elapsed time: {:.2f}".format(fps.elapsed()))  print("[INFO] approx. FPS: {:.2f}".format(fps.fps()))  # do a bit of cleanup  cv2.destroyAllWindows()  vs.stop() |

1. **Keterangan penjelasan MobileNetSSD\_deploy.caffemodel**

**Mobile Net caffemodel** ini merupakansalah satu arsitektur convolutional neural network (CNN) yang dapat digunakan untuk mengatasi kebutuhan akan computing resource berlebih.

Seperti namanya, Mobile, para peneliti dari Google membuat arsitektur CNN yang dapat digunakan untuk ponsel.

Perbedaan mendasar antara arsitektur MobileNet dan arsitektur CNN pada umumnya adalah penggunaan lapisan atau layer konvolusi dengan ketebalan filter yang sesuai dengan ketebalan dari input image. MobileNet membagi konvolusi menjadi depthwise convolution dan pointwise convolution

1. **Keterangan penjelasan MobileNetSSD\_deploy.prototext.txt**

**Mobile Net caffemodel berekstensi txt (tezt)** ini merupakan **export** data-data dari .caffemodel yang berbentuk text, sehingga dapat dilihat oleh pengguna yang ingin mengembangkan kode tersebut.

1. **KODE SUMBER :**

# Kode-kode sumber final project kami dapat di download di repo github.com kami di :

<https://github.com/kholilboy/FP_Pattern-Recognition>