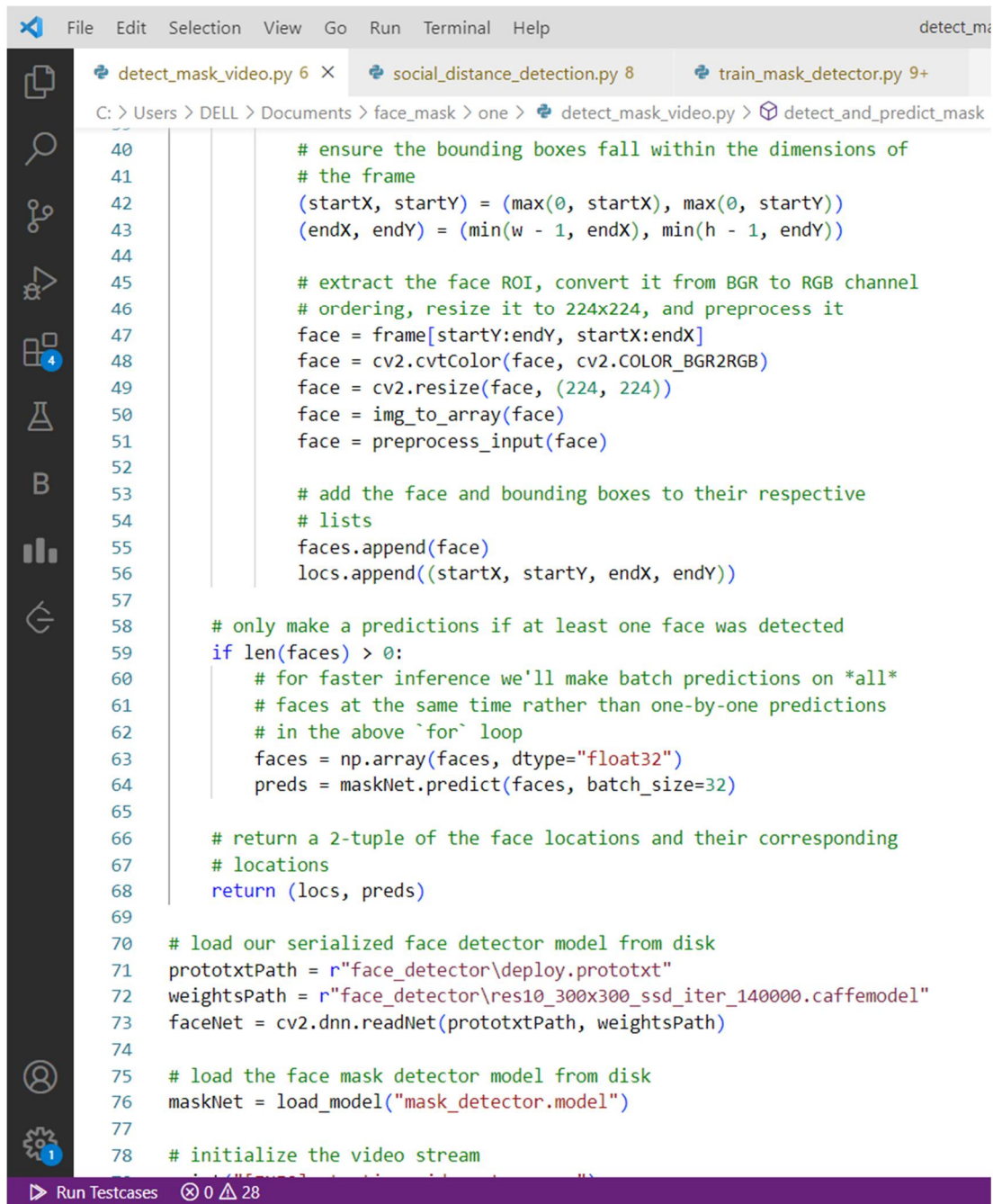


# Codes

- 200001006
- 200005040

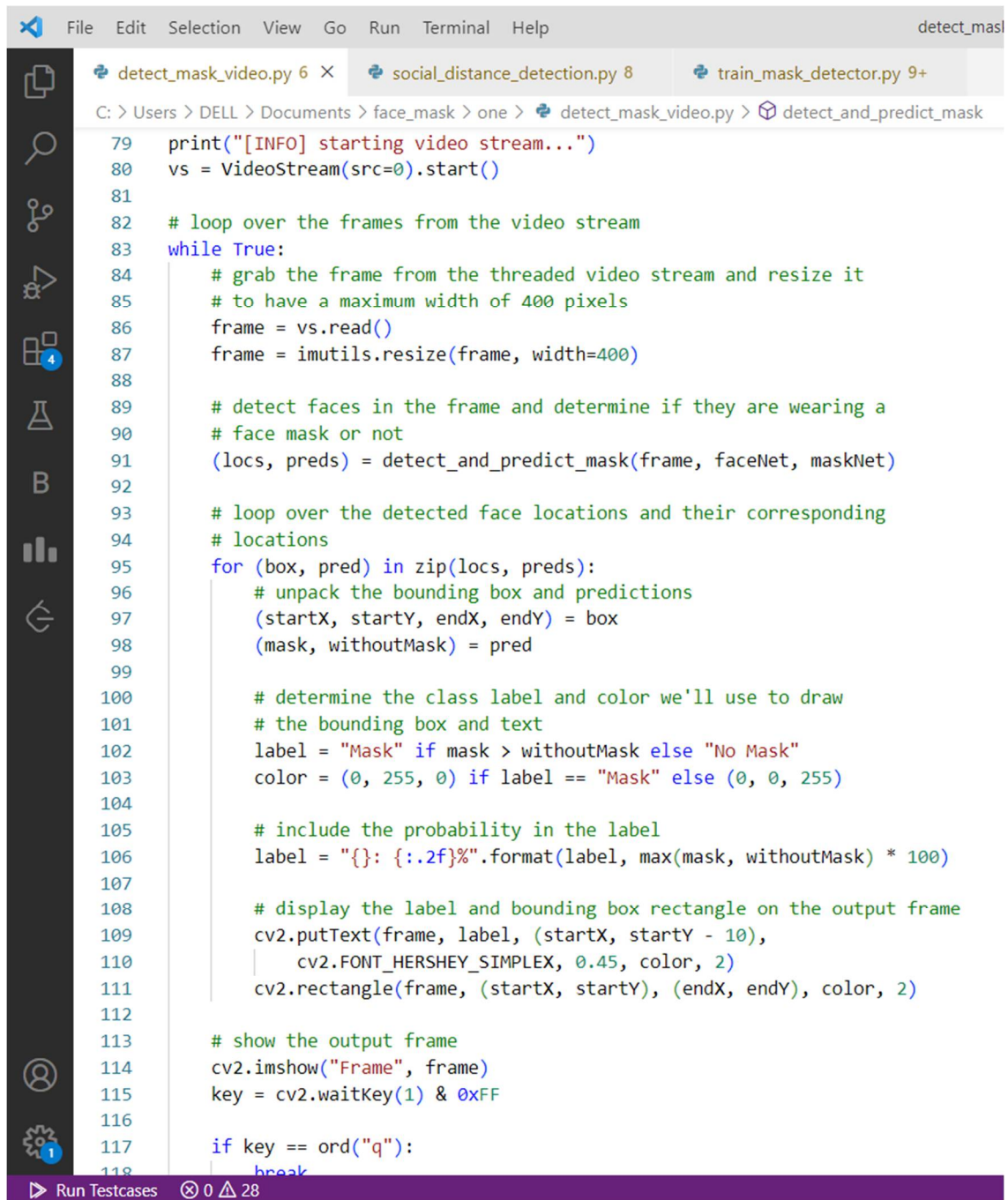
## Detect\_mask\_video.py Code-

```
detect_mask_video.py 6 × social_distance_detection.py 8 train_mask_detector.py 9+
C: > Users > DELL > Documents > face_mask > one > detect_mask_video.py > ...
1  # import the necessary packages
2  from tensorflow.keras.applications.mobilenet_v2 import preprocess_input
3  from tensorflow.keras.preprocessing.image import img_to_array
4  from tensorflow.keras.models import load_model
5  from imutils.video import VideoStream
6  import numpy as np
7  import imutils
8  import time
9  import cv2
10 import os
11
12 def detect_and_predict_mask(frame, faceNet, maskNet):
13     # grab the dimensions of the frame and then construct a blob from it
14     (h, w) = frame.shape[:2]
15     blob = cv2.dnn.blobFromImage(frame, 1.0, (224, 224), (104.0, 177.0, 123.0))
16
17     # pass the blob through the network and obtain the face detections
18     faceNet.setInput(blob) # setting input for facenet network
19     detections = faceNet.forward() # The detections list contains the predicted bounding boxes and confidence scores for each detected face
20     print(detections.shape)
21
22     # initialize our list of faces, their corresponding locations, and the list of predictions from our face mask network
23     faces = []
24     locs = []
25     preds = []
26
27     # loop over the detections
28     for i in range(0, detections.shape[2]):
29         # extract the confidence (i.e., probability) associated with the detection
30         confidence = detections[0, 0, i, 2]
31
32         # filter out weak detections by ensuring the confidence is
33         # greater than the minimum confidence
34         if confidence > 0.5:
35             # compute the (x, y)-coordinates of the bounding box for
36             # the object
37             box = detections[0, 0, i, 3:7] * np.array([w, h, w, h])
38             (startX, startY, endX, endY) = box.astype("int")
39
40             # ensure the bounding boxes fall within the dimensions of
```



The image shows a screenshot of an IDE with a dark theme. The top menu bar includes File, Edit, Selection, View, Go, Run, Terminal, and Help. The title bar on the right says 'detect\_m'. The editor has three tabs: 'detect\_mask\_video.py 6', 'social\_distance\_detection.py 8', and 'train\_mask\_detector.py 9+'. The active tab is 'detect\_mask\_video.py', and the file path is 'C: > Users > DELL > Documents > face\_mask > one > detect\_mask\_video.py > detect\_and\_predict\_mask'. The code is a Python script for face mask detection, with line numbers 40 to 78 visible. It includes comments in green and code in blue. The script processes a video frame to detect faces and masks, using OpenCV and a pre-trained model. The bottom status bar shows 'Run Testcases', '0', and '28'.

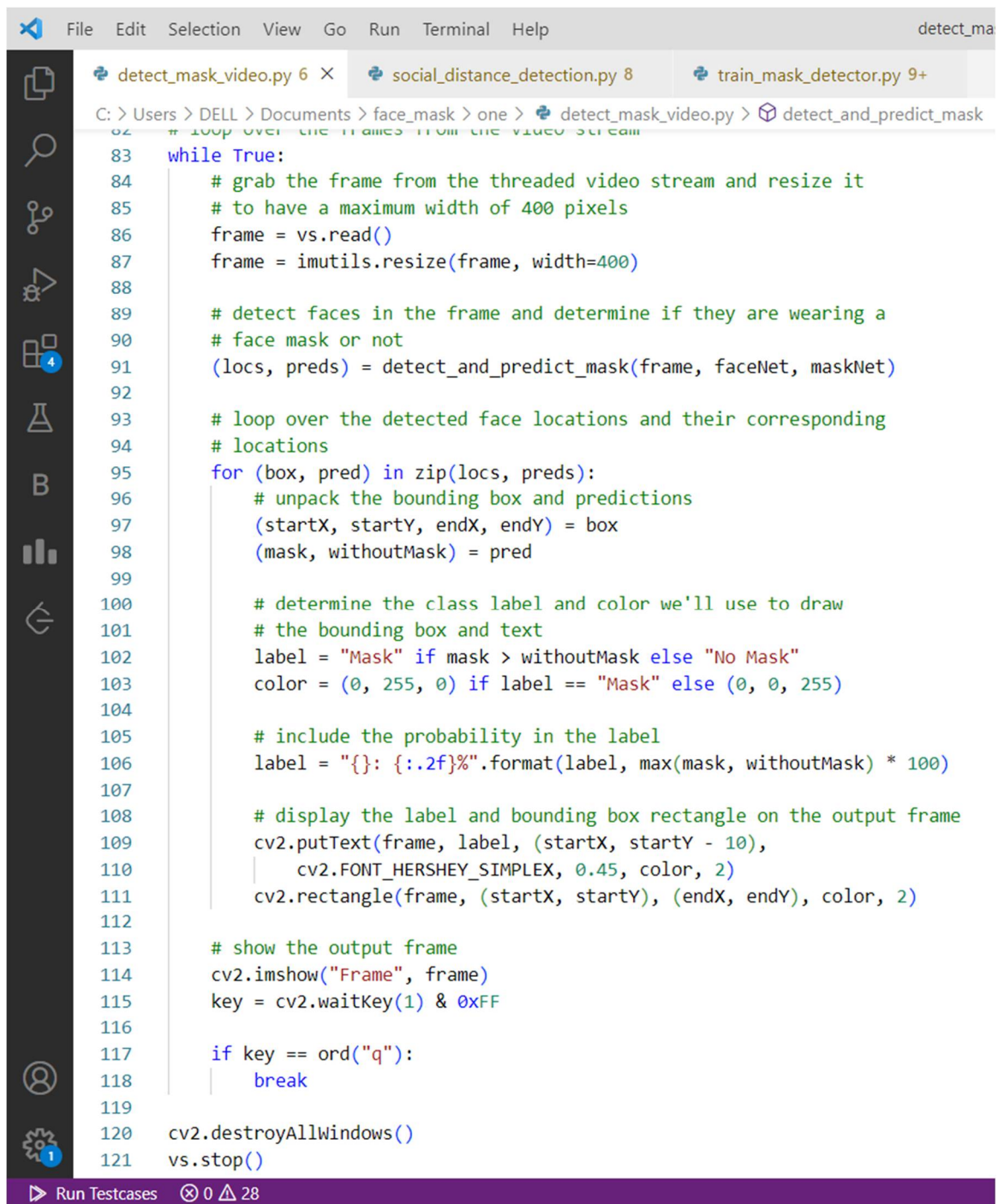
```
40     # ensure the bounding boxes fall within the dimensions of
41     # the frame
42     (startX, startY) = (max(0, startX), max(0, startY))
43     (endX, endY) = (min(w - 1, endX), min(h - 1, endY))
44
45     # extract the face ROI, convert it from BGR to RGB channel
46     # ordering, resize it to 224x224, and preprocess it
47     face = frame[startY:endY, startX:endX]
48     face = cv2.cvtColor(face, cv2.COLOR_BGR2RGB)
49     face = cv2.resize(face, (224, 224))
50     face = img_to_array(face)
51     face = preprocess_input(face)
52
53     # add the face and bounding boxes to their respective
54     # lists
55     faces.append(face)
56     locs.append((startX, startY, endX, endY))
57
58     # only make a predictions if at least one face was detected
59     if len(faces) > 0:
60         # for faster inference we'll make batch predictions on *all*
61         # faces at the same time rather than one-by-one predictions
62         # in the above `for` loop
63         faces = np.array(faces, dtype="float32")
64         preds = maskNet.predict(faces, batch_size=32)
65
66     # return a 2-tuple of the face locations and their corresponding
67     # locations
68     return (locs, preds)
69
70 # load our serialized face detector model from disk
71 prototxtPath = r"face_detector\deploy.prototxt"
72 weightsPath = r"face_detector\res10_300x300_ssd_iter_140000.caffemodel"
73 faceNet = cv2.dnn.readNet(prototxtPath, weightsPath)
74
75 # load the face mask detector model from disk
76 maskNet = load_model("mask_detector.model")
77
78 # initialize the video stream
```



The image shows a screenshot of an IDE with a dark theme. The top menu bar includes File, Edit, Selection, View, Go, Run, Terminal, and Help. The title bar on the right says 'detect\_mas'. The editor has three tabs: 'detect\_mask\_video.py 6', 'social\_distance\_detection.py 8', and 'train\_mask\_detector.py 9+'. The active tab is 'detect\_mask\_video.py', and the file path is 'C: > Users > DELL > Documents > face\_mask > one > detect\_mask\_video.py > detect\_and\_predict\_mask'. The code is a Python script for video mask detection, starting with a print statement and a while loop that processes video frames, detects faces, and displays the results. The status bar at the bottom shows 'Run Testcases', '0' errors, and '28' warnings.

```
79 print("[INFO] starting video stream...")
80 vs = VideoStream(src=0).start()
81
82 # loop over the frames from the video stream
83 while True:
84     # grab the frame from the threaded video stream and resize it
85     # to have a maximum width of 400 pixels
86     frame = vs.read()
87     frame = imutils.resize(frame, width=400)
88
89     # detect faces in the frame and determine if they are wearing a
90     # face mask or not
91     (locs, preds) = detect_and_predict_mask(frame, faceNet, maskNet)
92
93     # loop over the detected face locations and their corresponding
94     # locations
95     for (box, pred) in zip(locs, preds):
96         # unpack the bounding box and predictions
97         (startX, startY, endX, endY) = box
98         (mask, withoutMask) = pred
99
100        # determine the class label and color we'll use to draw
101        # the bounding box and text
102        label = "Mask" if mask > withoutMask else "No Mask"
103        color = (0, 255, 0) if label == "Mask" else (0, 0, 255)
104
105        # include the probability in the label
106        label = "{}: {:.2f}%".format(label, max(mask, withoutMask) * 100)
107
108        # display the label and bounding box rectangle on the output frame
109        cv2.putText(frame, label, (startX, startY - 10),
110                    cv2.FONT_HERSHEY_SIMPLEX, 0.45, color, 2)
111        cv2.rectangle(frame, (startX, startY), (endX, endY), color, 2)
112
113    # show the output frame
114    cv2.imshow("Frame", frame)
115    key = cv2.waitKey(1) & 0xFF
116
117    if key == ord("q"):
118        break
```

Run Testcases 0 28

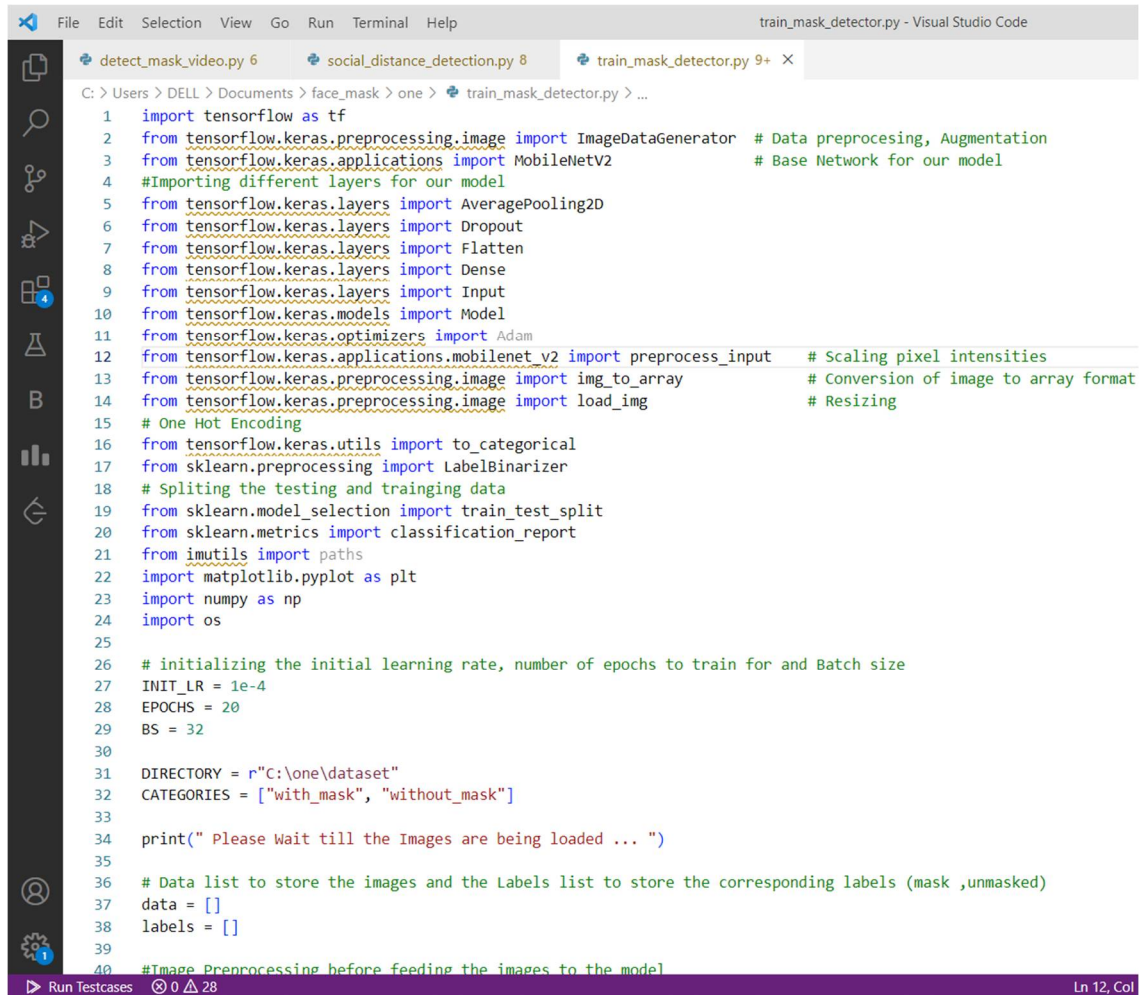


```
File Edit Selection View Go Run Terminal Help detect_ma
detect_mask_video.py 6 X social_distance_detection.py 8 train_mask_detector.py 9+
C:\Users\DELL\Documents\face_mask\one\detect_mask_video.py > detect_and_predict_mask
82 # loop over the frames from the video stream
83 while True:
84     # grab the frame from the threaded video stream and resize it
85     # to have a maximum width of 400 pixels
86     frame = vs.read()
87     frame = imutils.resize(frame, width=400)
88
89     # detect faces in the frame and determine if they are wearing a
90     # face mask or not
91     (locs, preds) = detect_and_predict_mask(frame, faceNet, maskNet)
92
93     # loop over the detected face locations and their corresponding
94     # locations
95     for (box, pred) in zip(locs, preds):
96         # unpack the bounding box and predictions
97         (startX, startY, endX, endY) = box
98         (mask, withoutMask) = pred
99
100        # determine the class label and color we'll use to draw
101        # the bounding box and text
102        label = "Mask" if mask > withoutMask else "No Mask"
103        color = (0, 255, 0) if label == "Mask" else (0, 0, 255)
104
105        # include the probability in the label
106        label = "{}: {:.2f}%".format(label, max(mask, withoutMask) * 100)
107
108        # display the label and bounding box rectangle on the output frame
109        cv2.putText(frame, label, (startX, startY - 10),
110                    cv2.FONT_HERSHEY_SIMPLEX, 0.45, color, 2)
111        cv2.rectangle(frame, (startX, startY), (endX, endY), color, 2)
112
113        # show the output frame
114        cv2.imshow("Frame", frame)
115        key = cv2.waitKey(1) & 0xFF
116
117        if key == ord("q"):
118            break
119
120    cv2.destroyAllWindows()
121    vs.stop()
```

Run Testcases 0 28



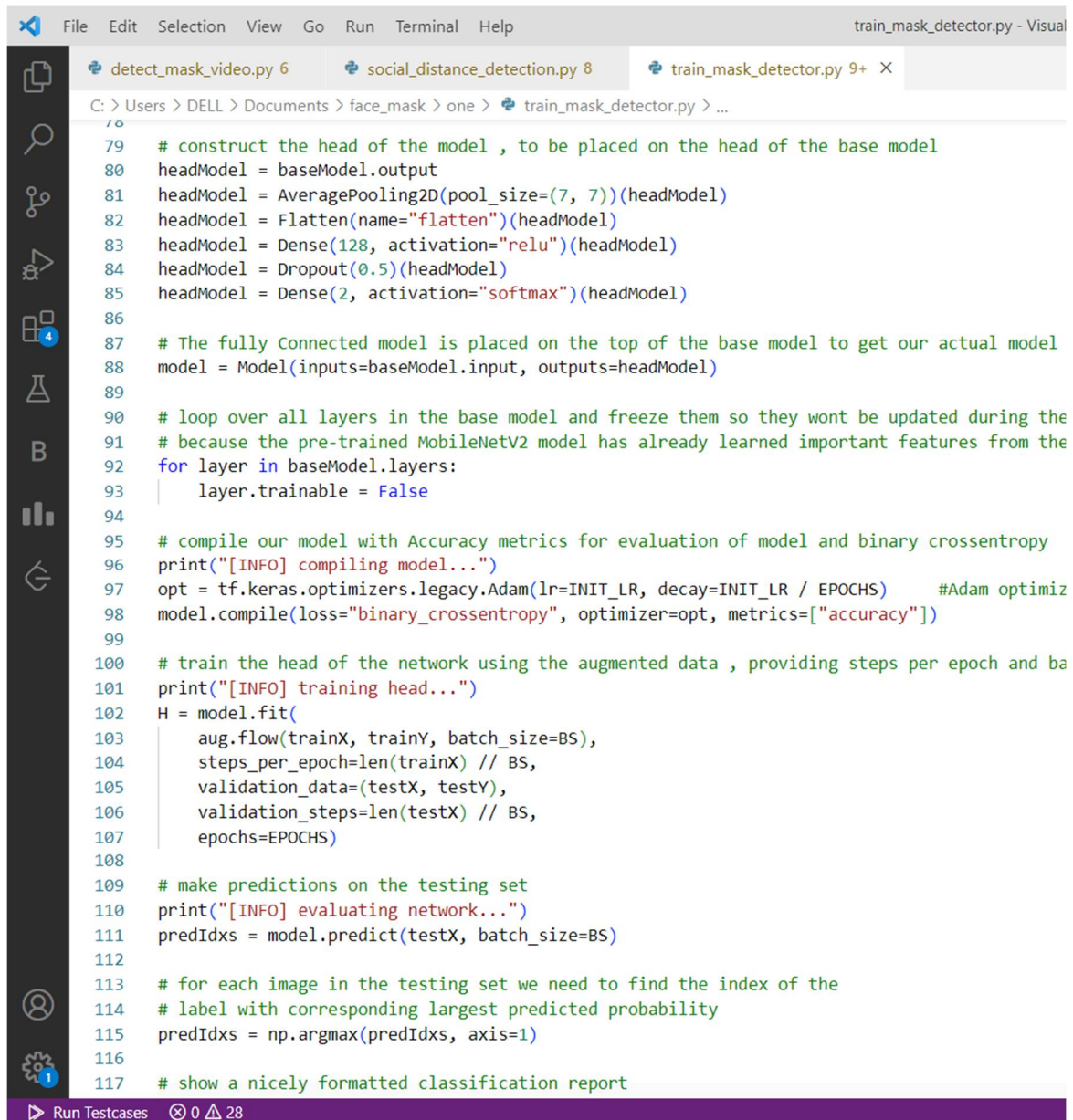
# Train\_mask\_detector.py Code-



```
1 import tensorflow as tf
2 from tensorflow.keras.preprocessing.image import ImageDataGenerator # Data preprocessing, Augmentation
3 from tensorflow.keras.applications import MobileNetV2 # Base Network for our model
4 #Importing different layers for our model
5 from tensorflow.keras.layers import AveragePooling2D
6 from tensorflow.keras.layers import Dropout
7 from tensorflow.keras.layers import Flatten
8 from tensorflow.keras.layers import Dense
9 from tensorflow.keras.layers import Input
10 from tensorflow.keras.models import Model
11 from tensorflow.keras.optimizers import Adam
12 from tensorflow.keras.applications.mobilenet_v2 import preprocess_input # Scaling pixel intensities
13 from tensorflow.keras.preprocessing.image import img_to_array # Conversion of image to array format
14 from tensorflow.keras.preprocessing.image import load_img # Resizing
15 # One Hot Encoding
16 from tensorflow.keras.utils import to_categorical
17 from sklearn.preprocessing import LabelBinarizer
18 # Splitting the testing and training data
19 from sklearn.model_selection import train_test_split
20 from sklearn.metrics import classification_report
21 from imutils import paths
22 import matplotlib.pyplot as plt
23 import numpy as np
24 import os
25
26 # initializing the initial learning rate, number of epochs to train for and Batch size
27 INIT_LR = 1e-4
28 EPOCHS = 20
29 BS = 32
30
31 DIRECTORY = r"C:\one\dataset"
32 CATEGORIES = ["with_mask", "without_mask"]
33
34 print(" Please Wait till the Images are being loaded ... ")
35
36 # Data list to store the images and the Labels list to store the corresponding labels (mask ,unmasked)
37 data = []
38 labels = []
39
40 #Image Preprocessing before feeding the images to the model
```

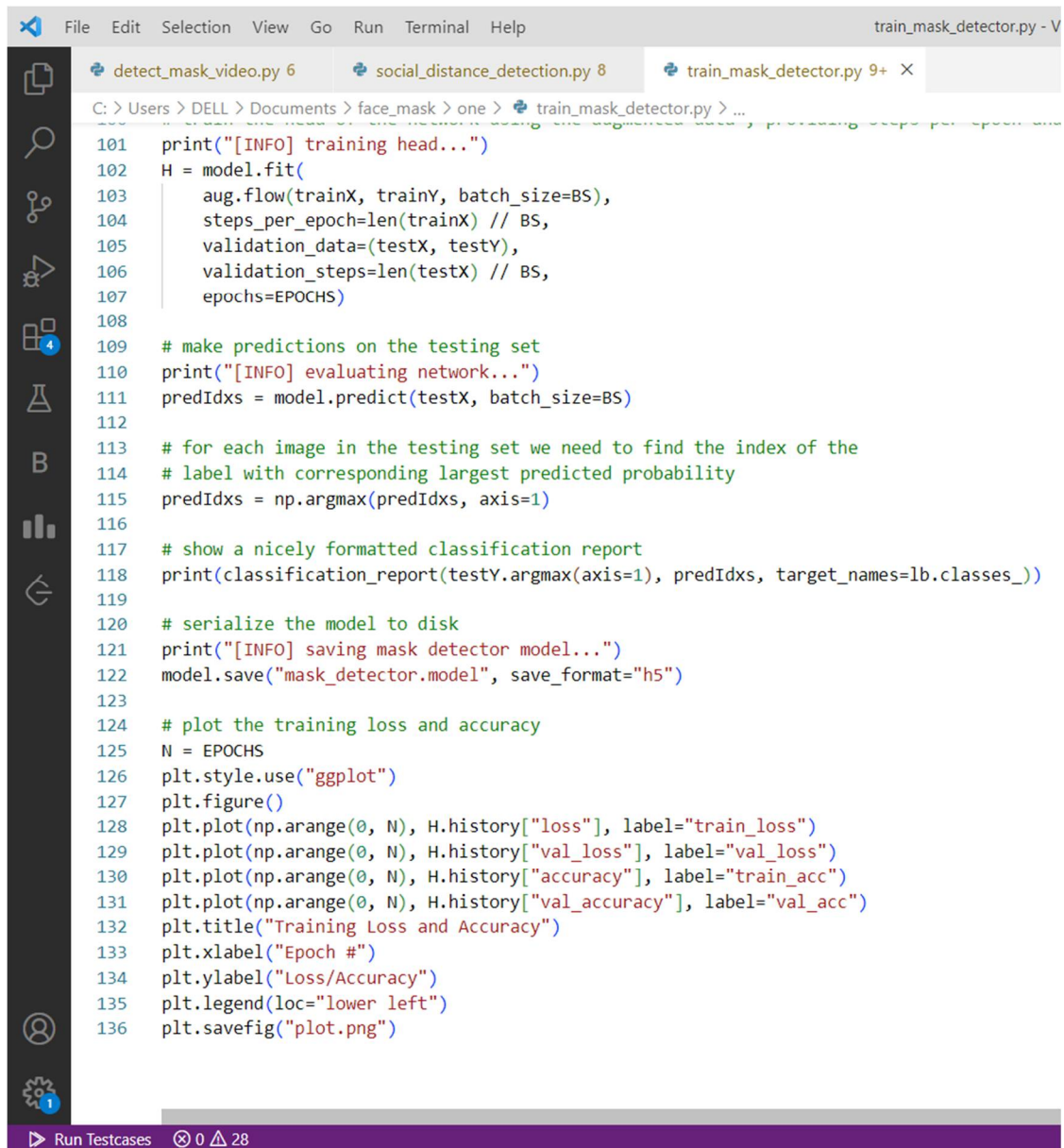
Run Testcases 0 28 Ln 12, Col

```
File Edit Selection View Go Run Terminal Help train_mask_detector.py - Visual Studio Code
detect_mask_video.py 6 social_distance_detection.py 8 train_mask_detector.py 9+ X
C: > Users > DELL > Documents > face_mask > one > train_mask_detector.py > ...
40 #Image Preprocessing before feeding the images to the model
41 for category in CATEGORIES:
42     path = os.path.join(DIRECTORY, category) # Creating full path to the directory of the images
43     for img in os.listdir(path):
44         img_path = os.path.join(path, img) # Resizing the Images to 224x224 pixels ensuring all images
45         image = load_img(img_path, target_size=(224, 224)) # Conversion to numpy array format
46         image = img_to_array(image) # Scaling the pixel intensities in the input image to range
47         image = preprocess_input(image)
48
49 data.append(image) # Appending the images to the Data List
50 labels.append(category) # Appending the corresponding label to the Labels List
51
52 # perform one-hot encoding on the labels
53 lb = LabelBinarizer() # converting categorical data to binar vectors
54 labels = lb.fit_transform(labels) # fits the label binarizer to labels and transforms the labels
55 labels = to_categorical(labels) # convert binary vectors to one hot encoded labels
56
57 # Numpy Array
58 data = np.array(data, dtype="float32")
59 labels = np.array(labels)
60
61 # Splitting the dataset into training and testing dataset
62 # Stratify ensures that each class in the target variable is proportionally represented in the training and testing sets.
63 (trainX, testX, trainY, testY) = train_test_split(data, labels, test_size=0.20, stratify=labels, random_state=42)
64
65 # construct the training image generator for data augmentation
66 aug = ImageDataGenerator(
67     rotation_range=20, # Rotation
68     zoom_range=0.15, # Magnification
69     width_shift_range=0.2, # Horizontal Shift
70     height_shift_range=0.2, # Vertical Shift
71     shear_range=0.15, # Shear
72     horizontal_flip=True, # Flip
73     fill_mode="nearest") # Fill empty pixels with the nearest pixels on shifting
74
75 # load the MobileNetV2 network, Excluding the fully connected layer to build our own binary classification
76
77 baseModel = MobileNetV2(weights="imagenet", include_top=False, input_tensor=Input(shape=(224, 224, 3)))
78
79 # construct the head of the model to be placed on the head of the base model
```



The image shows a Visual Studio Code editor window with the file `train_mask_detector.py` open. The editor has a dark theme and a sidebar on the left with icons for Explorer, Search, Source Control, Run and Debug, Extensions, Testing, and Settings. The top menu bar includes File, Edit, Selection, View, Go, Run, Terminal, and Help. The file explorer shows three files: `detect_mask_video.py 6`, `social_distance_detection.py 8`, and `train_mask_detector.py 9+ X`. The current file is `train_mask_detector.py`, located at `C:\Users\DELL\Documents\face_mask\one\train_mask_detector.py`. The code is a Python script for training a mask detector model. It starts with a comment on line 79: `# construct the head of the model , to be placed on the head of the base model`. The code then defines the head model by adding layers to `baseModel.output`: `AveragePooling2D` (line 80), `Flatten` (line 81), `Dense` (line 82), `Dropout` (line 83), and another `Dense` layer (line 84). The fully connected model is placed on top of the base model (line 85). The code then loops over all layers in the base model and freezes them (lines 86-90). The model is compiled with Accuracy metrics for evaluation of model and binary crossentropy (line 91). The model is trained using the augmented data, providing steps per epoch and batch size (lines 92-107). The code then makes predictions on the testing set (line 108) and evaluates the network (line 109). The predictions are used to find the index of the label with corresponding largest predicted probability (line 110). The code then shows a nicely formatted classification report (line 111).

```
79 # construct the head of the model , to be placed on the head of the base model
80 headModel = baseModel.output
81 headModel = AveragePooling2D(pool_size=(7, 7))(headModel)
82 headModel = Flatten(name="flatten")(headModel)
83 headModel = Dense(128, activation="relu")(headModel)
84 headModel = Dropout(0.5)(headModel)
85 headModel = Dense(2, activation="softmax")(headModel)
86
87 # The fully Connected model is placed on the top of the base model to get our actual model
88 model = Model(inputs=baseModel.input, outputs=headModel)
89
90 # loop over all layers in the base model and freeze them so they wont be updated during the
91 # because the pre-trained MobileNetV2 model has already learned important features from the
92 for layer in baseModel.layers:
93     layer.trainable = False
94
95 # compile our model with Accuracy metrics for evaluation of model and binary crossentropy
96 print("[INFO] compiling model...")
97 opt = tf.keras.optimizers.legacy.Adam(lr=INIT_LR, decay=INIT_LR / EPOCHS) #Adam optimiz
98 model.compile(loss="binary_crossentropy", optimizer=opt, metrics=["accuracy"])
99
100 # train the head of the network using the augmented data , providing steps per epoch and ba
101 print("[INFO] training head...")
102 H = model.fit(
103     aug.flow(trainX, trainY, batch_size=BS),
104     steps_per_epoch=len(trainX) // BS,
105     validation_data=(testX, testY),
106     validation_steps=len(testX) // BS,
107     epochs=EPOCHS)
108
109 # make predictions on the testing set
110 print("[INFO] evaluating network...")
111 predIdxs = model.predict(testX, batch_size=BS)
112
113 # for each image in the testing set we need to find the index of the
114 # label with corresponding largest predicted probability
115 predIdxs = np.argmax(predIdxs, axis=1)
116
117 # show a nicely formatted classification report
```

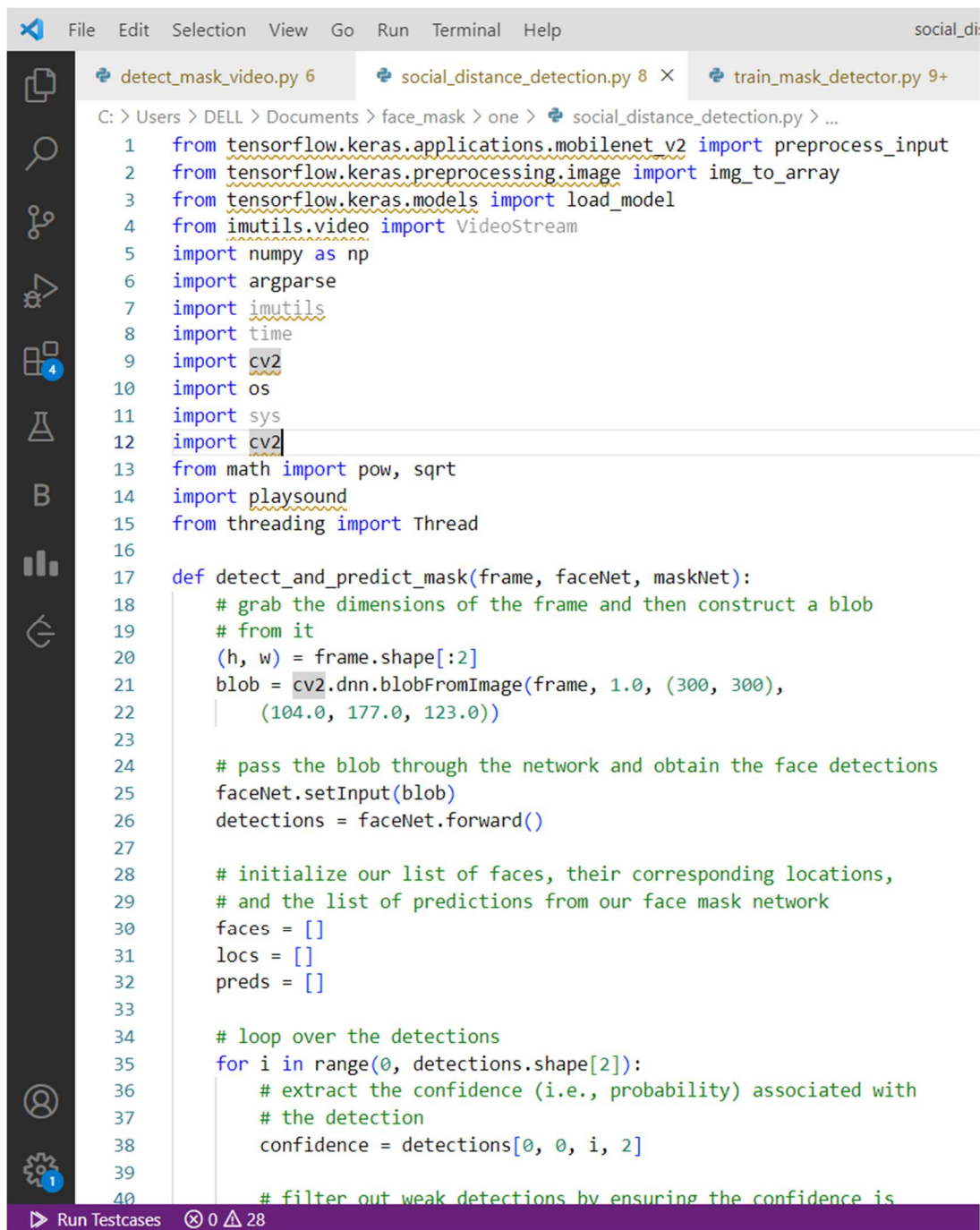


```
101 print("[INFO] training head...")
102 H = model.fit(
103     aug.flow(trainX, trainY, batch_size=BS),
104     steps_per_epoch=len(trainX) // BS,
105     validation_data=(testX, testY),
106     validation_steps=len(testX) // BS,
107     epochs=EPOCHS)
108
109 # make predictions on the testing set
110 print("[INFO] evaluating network...")
111 predIdxs = model.predict(testX, batch_size=BS)
112
113 # for each image in the testing set we need to find the index of the
114 # label with corresponding largest predicted probability
115 predIdxs = np.argmax(predIdxs, axis=1)
116
117 # show a nicely formatted classification report
118 print(classification_report(testY.argmax(axis=1), predIdxs, target_names=lb.classes_))
119
120 # serialize the model to disk
121 print("[INFO] saving mask detector model...")
122 model.save("mask_detector.model", save_format="h5")
123
124 # plot the training loss and accuracy
125 N = EPOCHS
126 plt.style.use("ggplot")
127 plt.figure()
128 plt.plot(np.arange(0, N), H.history["loss"], label="train_loss")
129 plt.plot(np.arange(0, N), H.history["val_loss"], label="val_loss")
130 plt.plot(np.arange(0, N), H.history["accuracy"], label="train_acc")
131 plt.plot(np.arange(0, N), H.history["val_accuracy"], label="val_acc")
132 plt.title("Training Loss and Accuracy")
133 plt.xlabel("Epoch #")
134 plt.ylabel("Loss/Accuracy")
135 plt.legend(loc="lower left")
136 plt.savefig("plot.png")
```

Run Testcases 0 / 28

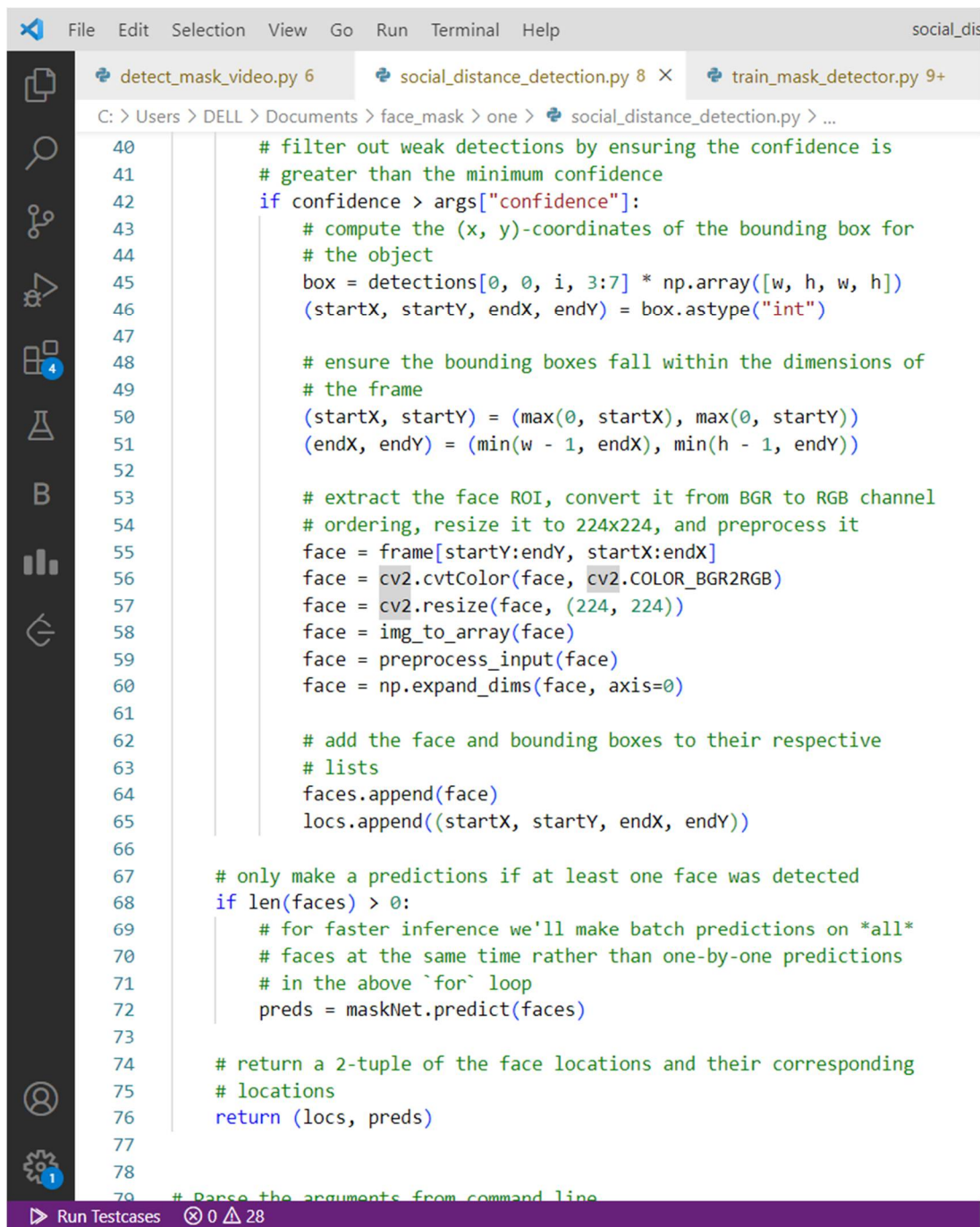


## Social\_mask\_distancing.py Code-



```
C: > Users > DELL > Documents > face_mask > one > social_distance_detection.py > ...
1  from tensorflow.keras.applications.mobilenet_v2 import preprocess_input
2  from tensorflow.keras.preprocessing.image import img_to_array
3  from tensorflow.keras.models import load_model
4  from imutils.video import VideoStream
5  import numpy as np
6  import argparse
7  import imutils
8  import time
9  import cv2
10 import os
11 import sys
12 import cv2
13 from math import pow, sqrt
14 import playsound
15 from threading import Thread
16
17 def detect_and_predict_mask(frame, faceNet, maskNet):
18     # grab the dimensions of the frame and then construct a blob
19     # from it
20     (h, w) = frame.shape[:2]
21     blob = cv2.dnn.blobFromImage(frame, 1.0, (300, 300),
22     |     (104.0, 177.0, 123.0))
23
24     # pass the blob through the network and obtain the face detections
25     faceNet.setInput(blob)
26     detections = faceNet.forward()
27
28     # initialize our list of faces, their corresponding locations,
29     # and the list of predictions from our face mask network
30     faces = []
31     locs = []
32     preds = []
33
34     # loop over the detections
35     for i in range(0, detections.shape[2]):
36         # extract the confidence (i.e., probability) associated with
37         # the detection
38         confidence = detections[0, 0, i, 2]
39
40         # filter out weak detections by ensuring the confidence is
```

Run Testcases 0 28



```
File Edit Selection View Go Run Terminal Help social_dis
detect_mask_video.py 6 social_distance_detection.py 8 X train_mask_detector.py 9+
C: > Users > DELL > Documents > face_mask > one > social_distance_detection.py > ...

40 # filter out weak detections by ensuring the confidence is
41 # greater than the minimum confidence
42 if confidence > args["confidence"]:
43     # compute the (x, y)-coordinates of the bounding box for
44     # the object
45     box = detections[0, 0, i, 3:7] * np.array([w, h, w, h])
46     (startX, startY, endX, endY) = box.astype("int")
47
48     # ensure the bounding boxes fall within the dimensions of
49     # the frame
50     (startX, startY) = (max(0, startX), max(0, startY))
51     (endX, endY) = (min(w - 1, endX), min(h - 1, endY))
52
53     # extract the face ROI, convert it from BGR to RGB channel
54     # ordering, resize it to 224x224, and preprocess it
55     face = frame[startY:endY, startX:endX]
56     face = cv2.cvtColor(face, cv2.COLOR_BGR2RGB)
57     face = cv2.resize(face, (224, 224))
58     face = img_to_array(face)
59     face = preprocess_input(face)
60     face = np.expand_dims(face, axis=0)
61
62     # add the face and bounding boxes to their respective
63     # lists
64     faces.append(face)
65     locs.append((startX, startY, endX, endY))
66
67 # only make a predictions if at least one face was detected
68 if len(faces) > 0:
69     # for faster inference we'll make batch predictions on *all*
70     # faces at the same time rather than one-by-one predictions
71     # in the above `for` loop
72     preds = maskNet.predict(faces)
73
74 # return a 2-tuple of the face locations and their corresponding
75 # locations
76 return (locs, preds)
77
78 # Parse the arguments from command line
79
```

Run Testcases 0 / 28

```
it Selection View Go Run Terminal Help social_distance_detection.py - Visual Studio Code
tect_mask_video.py 6 social_distance_detection.py 8 X train_mask_detector.py 9+
Jsers > DELL > Documents > face_mask > one > social_distance_detection.py > ...

# Parse the arguments from command line
arg = argparse.ArgumentParser(description='Social distance detection')

arg.add_argument("-a", "--alarm", type=str, default="", help="path alarm .WAV file")

arg.add_argument('-v', '--video', type = str, default = '', help = 'Video file path. If no path is given, video is captured using device.')

arg.add_argument('-m', '--model', required = True, help = "Path to the pretrained model.")

arg.add_argument('-p', '--prototxt', required = True, help = 'Prototxt of the model.')

arg.add_argument('-l', '--labels', required = True, help = 'Labels of the dataset.')

arg.add_argument('-c', '--confidence', type = float, default = 0.2, help='Set confidence for detecting objects')

arg.add_argument("-f", "--face", type=str, default="face_detector", help="path to face detector model directory")

arg.add_argument("-m1", "--model1", type=str, default="model/mask_detector.model", help="path to trained face mask detector model")

#arg.add_argument("-c", "--confidence", type=float, default=0.5, help="minimum probability to filter weak detections")

args = vars(arg.parse_args())

ALARM_ON = False

def sound_alarm(path):
    # play an alarm sound
    playsound.playsound(path)

labels = [line.strip() for line in open(args['labels'])]

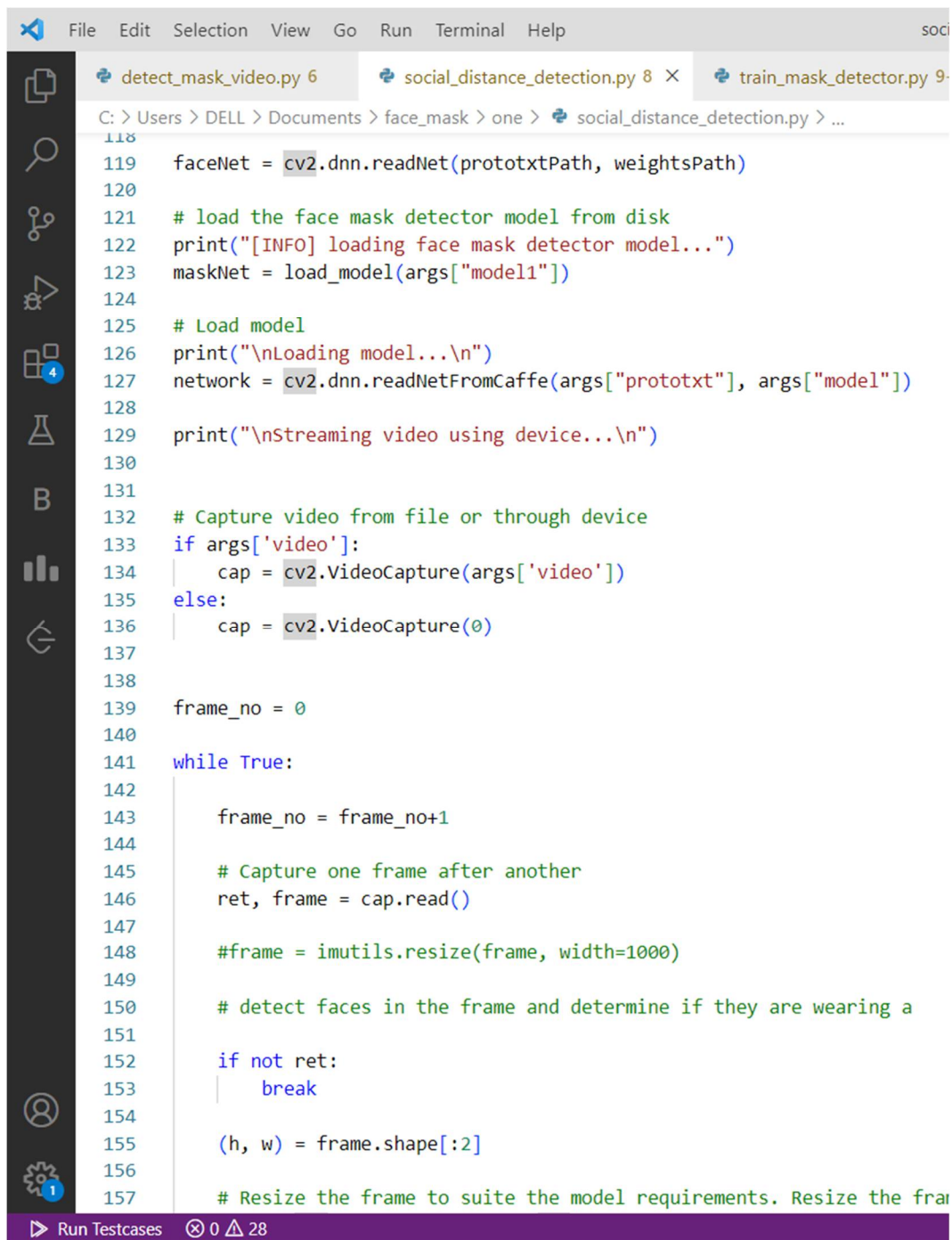
# Generate random bounding box bounding_box_color for each label
bounding_box_color = np.random.uniform(0, 255, size=(len(labels), 3))

print("[INFO] loading face detector model...")

prototxtPath = os.path.sep.join([args["face"], "deploy.prototxt"])

weightsPath = os.path.sep.join([args["face"], "res10_300x300_ssd_iter_140000.caffemodel"])
```

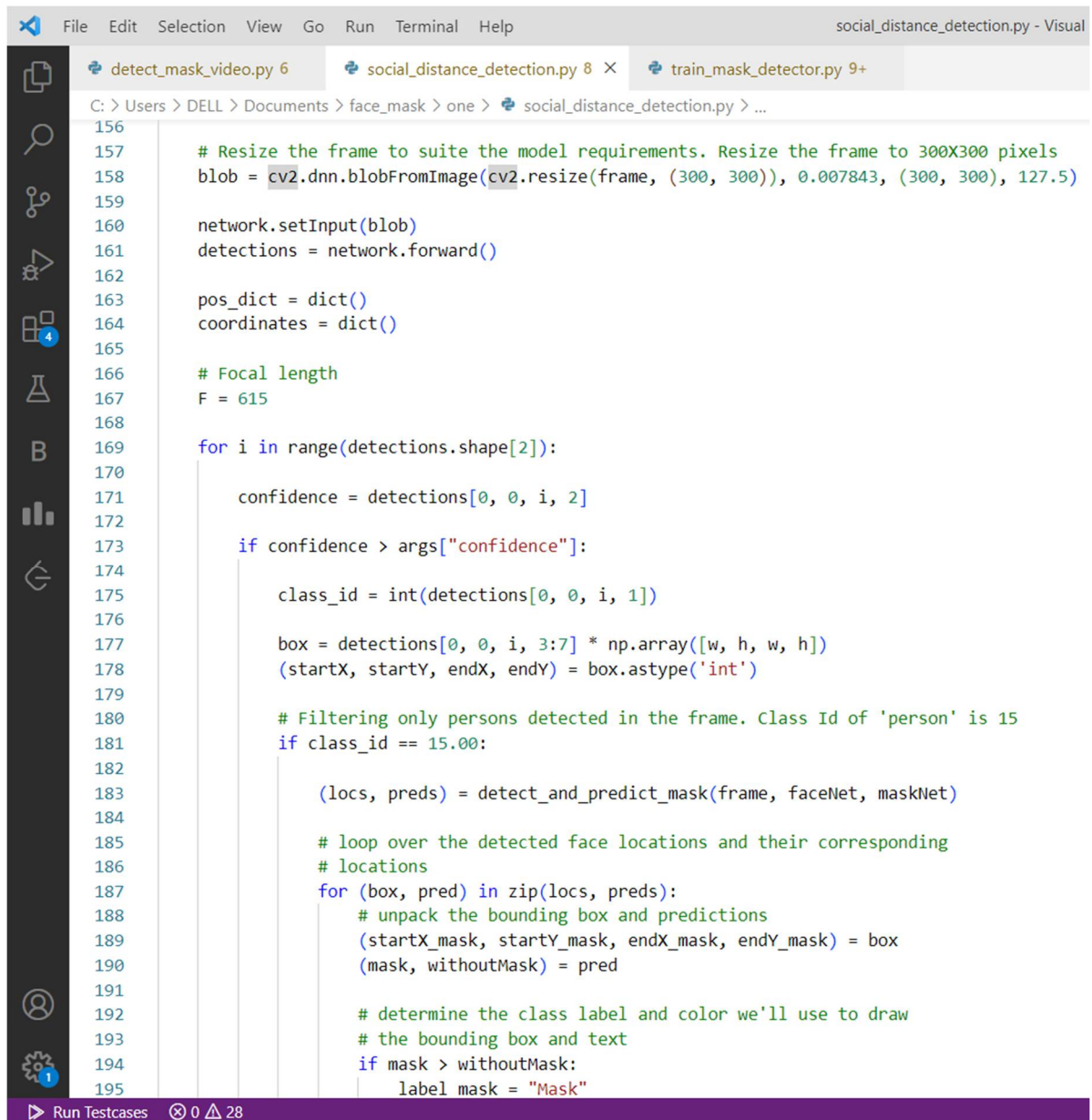
ses 0 28 Ln 12, Col 11 Spaces: 4 UTF-8 CRLF Py



The image shows a code editor window with three tabs: `detect_mask_video.py 6`, `social_distance_detection.py 8`, and `train_mask_detector.py 9`. The active tab is `social_distance_detection.py`, showing a Python script for social distance detection. The script includes comments in green and code in blue and black. The path in the top bar is `C:\> Users > DELL > Documents > face_mask > one > social_distance_detection.py > ...`. The status bar at the bottom shows `Run Testcases` and `0 28`.

```
118
119 faceNet = cv2.dnn.readNet(prototxtPath, weightsPath)
120
121 # load the face mask detector model from disk
122 print("[INFO] loading face mask detector model...")
123 maskNet = load_model(args["model1"])
124
125 # Load model
126 print("\nLoading model...\n")
127 network = cv2.dnn.readNetFromCaffe(args["prototxt"], args["model"])
128
129 print("\nStreaming video using device...\n")
130
131
132 # Capture video from file or through device
133 if args['video']:
134     cap = cv2.VideoCapture(args['video'])
135 else:
136     cap = cv2.VideoCapture(0)
137
138
139 frame_no = 0
140
141 while True:
142
143     frame_no = frame_no+1
144
145     # Capture one frame after another
146     ret, frame = cap.read()
147
148     #frame = imutils.resize(frame, width=1000)
149
150     # detect faces in the frame and determine if they are wearing a
151
152     if not ret:
153         break
154
155     (h, w) = frame.shape[:2]
156
157     # Resize the frame to suite the model requirements. Resize the fra
```





```
156
157 # Resize the frame to suite the model requirements. Resize the frame to 300X300 pixels
158 blob = cv2.dnn.blobFromImage(cv2.resize(frame, (300, 300)), 0.007843, (300, 300), 127.5)
159
160 network.setInput(blob)
161 detections = network.forward()
162
163 pos_dict = dict()
164 coordinates = dict()
165
166 # Focal length
167 F = 615
168
169 for i in range(detections.shape[2]):
170
171     confidence = detections[0, 0, i, 2]
172
173     if confidence > args["confidence"]:
174
175         class_id = int(detections[0, 0, i, 1])
176
177         box = detections[0, 0, i, 3:7] * np.array([w, h, w, h])
178         (startX, startY, endX, endY) = box.astype('int')
179
180         # Filtering only persons detected in the frame. Class Id of 'person' is 15
181         if class_id == 15.00:
182
183             (locs, preds) = detect_and_predict_mask(frame, faceNet, maskNet)
184
185             # loop over the detected face locations and their corresponding
186             # locations
187             for (box, pred) in zip(locs, preds):
188                 # unpack the bounding box and predictions
189                 (startX_mask, startY_mask, endX_mask, endY_mask) = box
190                 (mask, withoutMask) = pred
191
192                 # determine the class label and color we'll use to draw
193                 # the bounding box and text
194                 if mask > withoutMask:
195                     label mask = "Mask"
```

Run Testcases 0 / 28

```
File Edit Selection View Go Run Terminal Help social_distance_detection.py - Visual Studio Code
detect_mask_video.py 6 social_distance_detection.py 8 X train_mask_detector.py 9+
C: > Users > DELL > Documents > face_mask > one > social_distance_detection.py > ...

195     label_mask = "Mask"
196     color = (0, 255, 0)
197     ALARM_ON = False
198 else:
199     label_mask="No Mask"
200     color = (0, 0, 255)
201     if not ALARM_ON:
202         ALARM_ON = True
203         if args["alarm"] != "":
204             t = Thread(target=sound_alarm, args=(args["alarm"],))
205             t.daemon = True
206             t.start()
207
208 # Draw bounding box for the object
209 cv2.rectangle(frame, (startX, startY), (endX, endY), bounding_box_color[class_id], 2)
210
211 label = "{}: {:.2f}%".format(labels[class_id], confidence * 100)
212 label_mask = "{}: {:.2f}%".format(label_mask, max(mask, withoutMask) * 100)
213 print("{} {}".format(label, label_mask))
214 cv2.putText(frame, label, (startX_mask, startY_mask - 10), cv2.FONT_HERSHEY_SIMPLEX, 0.45, color, 2)
215 cv2.rectangle(frame, (startX_mask, startY_mask), (endX_mask, endY_mask), color, 2)
216
217
218 coordinates[i] = (startX, startY, endX, endY)
219
220 # Mid point of bounding box
221 x_mid = round((startX+endX)/2,4)
222 y_mid = round((startY+endY)/2,4)
223
224 height = round(endY-startY,4)
225
226 # Distance from camera based on triangle similarity
227 distance = (165 * F)/height
228 print("Distance(cm):{dist}\n".format(dist=distance))
229
230 # Mid-point of bounding boxes (in cm) based on triangle similarity technique
231 x_mid_cm = (x_mid * distance) / F
232 y_mid_cm = (y_mid * distance) / F
233 pos_dict[i] = (x_mid_cm,y_mid_cm,distance)
234
Run Testcases 0 28 Ln 12, Col 11
```

```
File Edit Selection View Go Run Terminal Help social_distance_detection.py - Visual Studio Code
detect_mask_video.py 6 social_distance_detection.py 8 X train_mask_detector.py 9+
C:\Users\DELL\Documents> face_mask > one > social_distance_detection.py > ...

235 # Distance between every object detected in a frame
236 close_objects = set()
237 for i in pos_dict.keys():
238     for j in pos_dict.keys():
239         if i < j:
240             dist = sqrt(pow(pos_dict[i][0]-pos_dict[j][0],2) + pow(pos_dict[i][1]-pos_dict[j][1],2) + pow(pos_dict[i][2]-pos_dict[j][2],2))
241
242             # Check if distance less than 2 metres or 200 centimetres
243             if dist < 200:
244                 close_objects.add(i)
245                 close_objects.add(j)
246
247 for i in pos_dict.keys():
248     if i in close_objects:
249         COLOR = (0,0,255)
250         if not ALARM_ON:
251             ALARM_ON = True
252             if args["alarm"] != "":
253                 t = Thread(target=sound_alarm, args=(args["alarm"],))
254                 t.daemon = True
255                 t.start()
256
257     else:
258         COLOR = (0,255,0)
259         ALARM_ON = False
260         (startX, startY, endX, endY) = coordinates[i]
261
262         cv2.rectangle(frame, (startX, startY), (endX, endY), COLOR, 2)
263         y = startY - 15 if startY - 15 > 15 else startY + 15
264         # Convert cms to feet
265         cv2.putText(frame, 'Depth: {i} ft'.format(i=round(pos_dict[i][2]/30.48,4)), (startX, y),
266                     cv2.FONT_HERSHEY_SIMPLEX, 0.5, COLOR, 2)
267
268 cv2.namedWindow('Frame', cv2.WINDOW_NORMAL)
269
270 # Show frame
271 cv2.imshow('Frame', frame)
272 cv2.resizeWindow('Frame', 800, 600)
273
```

Run Testcases 0 28 Ln 12, Col 11 Spaces: 4 UTF-8 CRLF Python