IMPORT LIBRARY

```
from tensorflow.keras.callbacks import ModelCheckpoint, EarlyStopping
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.applications import MobileNetV2
from tensorflow.keras.layers import GlobalAveragePooling2D, Dense
from tensorflow.keras.layers import Input
from tensorflow.keras.models import Model
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.applications.mobilenet_v2 import preprocess_input
from tensorflow.keras.preprocessing.image import img_to_array
from tensorflow.keras.preprocessing.image import load_img
from tensorflow.keras.utils import to_categorical
from sklearn.preprocessing import LabelBinarizer
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report
from imutils import paths
import matplotlib.pyplot as plt
import numpy as np
import os
```

INPUT MODEL MOBILENET

```
base_model = MobileNetV2(weights = "imagenet", include_top = False, input_shape = (224, 22

head_model = base_model.output
head_model = GlobalAveragePooling2D()(head_model)
head_model = Dense(128,activation='relu')(head_model)
head_model = Dense(2, activation='softmax')(head_model)

model_train = Model(inputs = base_model.input, outputs = head_model)

for layer in base_model.layers:
  layer.trainable = False

model_train.summary()
```

```
Model: "model_1"
__________________________________________________________________________________________
Layer (type)                    Output Shape         Param #     Connected to
==========================================================================================
input_2 (InputLayer)            [(None, 224, 224, 3) 0
__________________________________________________________________________________________
Conv1 (Conv2D)                  (None, 112, 112, 32) 864         input_2[0][0]
__________________________________________________________________________________________
bn_Conv1 (BatchNormalization)   (None, 112, 112, 32) 128         Conv1[0][0]
__________________________________________________________________________________________
Conv1_relu (ReLU)               (None, 112, 112, 32) 0           bn_Conv1[0][0]
__________________________________________________________________________________________
```

expanded_conv_depthwise (Depthw	(None, 112, 112, 32)	288	Conv1_relu[0][0]
expanded_conv_depthwise_BN (Bat	(None, 112, 112, 32)	128	expanded_conv_dep
expanded_conv_depthwise_relu (R	(None, 112, 112, 32)	0	expanded_conv_dep
expanded_conv_project (Conv2D)	(None, 112, 112, 16)	512	expanded_conv_dep
expanded_conv_project_BN (Batch	(None, 112, 112, 16)	64	expanded_conv_pro
block_1_expand (Conv2D)	(None, 112, 112, 96)	1536	expanded_conv_pro
block_1_expand_BN (BatchNormali	(None, 112, 112, 96)	384	block_1_expand[0]
block_1_expand_relu (ReLU)	(None, 112, 112, 96)	0	block_1_expand_BN
block_1_pad (ZeroPadding2D)	(None, 113, 113, 96)	0	block_1_expand_re
block_1_depthwise (DepthwiseCon	(None, 56, 56, 96)	864	block_1_pad[0][0]
block_1_depthwise_BN (BatchNorm	(None, 56, 56, 96)	384	block_1_depthwise
block_1_depthwise_relu (ReLU)	(None, 56, 56, 96)	0	block_1_depthwise
block_1_project (Conv2D)	(None, 56, 56, 24)	2304	block_1_depthwise
block_1_project_BN (BatchNormal	(None, 56, 56, 24)	96	block_1_project[0
block_2_expand (Conv2D)	(None, 56, 56, 144)	3456	block_1_project_B
block_2_expand_BN (BatchNormali	(None, 56, 56, 144)	576	block_2_expand[0]
block_2_expand_relu (ReLU)	(None, 56, 56, 144)	0	block_2_expand_BN
block_2_depthwise (DepthwiseCon	(None, 56, 56, 144)	1296	block_2_expand_re
block_2_depthwise_BN (BatchNorm	(None, 56, 56, 144)	576	block_2_depthwise
block_2_depthwise_relu (ReLU)	(None, 56, 56, 144)	0	block_2_depthwise
block_2_project (Conv2D)	(None, 56, 56, 24)	3456	block_2_depthwise
block_2_project_BN (BatchNormal	(None, 56, 56, 24)	96	block_2_project[0
block_2_add (Add)	(None, 56, 56, 24)	0	block_1_project_B block_2_project_B

INPUT DATASET

```
directory_dataset = "/content/drive/MyDrive/FaceRecog/new_dataset/dataset15"
category_dataset = ["edi", "unknown"]

data = []
labels = []

for category in category_dataset:
    path = os.path.join(directory_dataset, category)
```

```
    path = os.path.join(directory_dataset, category)
    for img in os.listdir(path):
      img_path = os.path.join(path, img)
      image = load_img(img_path, target_size=(224, 224))
      image = img_to_array(image)
      image = preprocess_input(image)

      data.append(image)
      labels.append(category)
```

LEARNING RATE, EPOCH, BATCH SIZE ETC

```
lb = LabelBinarizer()
labels = lb.fit_transform(labels)
labels = to_categorical(labels)

data = np.array(data, dtype="float32")
labels = np.array(labels)
# print(labels)


learning_rate_size = 1e-4
epoch_size = 10
batch_size_number = 6

(trainX, testX, trainY, testY) = train_test_split(data,
                                                  labels,
                                                  test_size=0.20,
                                                  stratify=labels,
                                                  random_state=42)
```

DATA AUGMENTATION

```
aug = ImageDataGenerator(rotation_range=20,
                         zoom_range=0.15,
                         width_shift_range=0.2,
                         height_shift_range=0.2,
                         shear_range=0.15,
                         horizontal_flip=True,
                         fill_mode="nearest")
```

CALLBACKS

```
checkpoint = ModelCheckpoint("/content/drive/MyDrive/FaceRecog/new_dataset/result/result15
                             monitor="val_loss",
                             mode="min"
```

```
                              mode= 'min',
                              save_best_only = True,
                              verbose=1)

earlystop = EarlyStopping(monitor = 'val_loss',
                          min_delta = 0,
                          patience = 3,
                          verbose = 1,
                          restore_best_weights = True)

# we put our call backs into a callback list
callbacks = [earlystop, checkpoint]
```

OPTIMIZER AND TRAINING

```
print("[INFO] compiling model...")
opt = Adam(lr=learning_rate_size, decay=learning_rate_size / epoch_size)
model_train.compile(loss="binary_crossentropy",
                    optimizer=opt,
                    metrics=["accuracy"])

# train the head of the network
print("[INFO] training head...")
H = model_train.fit(aug.flow(trainX, trainY, batch_size=batch_size_number),
                    steps_per_epoch=len(trainX) // batch_size_number,
                    validation_data=(testX, testY),
                    validation_steps=len(testX) // batch_size_number,
                    epochs=epoch_size,
                    callbacks = callbacks)
```

```
[INFO] compiling model...
[INFO] training head...
Epoch 1/10
4/4 [==============================] - 5s 645ms/step - loss: 0.7358 - accuracy: 0.57

Epoch 00001: val_loss improved from inf to 0.51346, saving model to /content/drive/M
Epoch 2/10
4/4 [==============================] - 1s 341ms/step - loss: 0.6304 - accuracy: 0.67

Epoch 00002: val_loss improved from 0.51346 to 0.45110, saving model to /content/dri
Epoch 3/10
4/4 [==============================] - 1s 335ms/step - loss: 0.5499 - accuracy: 0.82

Epoch 00003: val_loss improved from 0.45110 to 0.39951, saving model to /content/dri
Epoch 4/10
4/4 [==============================] - 1s 346ms/step - loss: 0.4901 - accuracy: 0.97

Epoch 00004: val_loss improved from 0.39951 to 0.34987, saving model to /content/dri
Epoch 5/10
4/4 [==============================] - 1s 336ms/step - loss: 0.3883 - accuracy: 0.97

Epoch 00005: val_loss improved from 0.34987 to 0.30860, saving model to /content/dri
Epoch 6/10
4/4 [==============================] - 1s 339ms/step - loss: 0.4667 - accuracy: 0.80
```

```
Epoch 00006: val_loss improved from 0.30860 to 0.27402, saving model to /content/dri
Epoch 7/10
4/4 [==============================] - 1s 335ms/step - loss: 0.3070 - accuracy: 1.00

Epoch 00007: val_loss improved from 0.27402 to 0.24323, saving model to /content/dri
Epoch 8/10
4/4 [==============================] - 1s 330ms/step - loss: 0.3034 - accuracy: 1.00

Epoch 00008: val_loss improved from 0.24323 to 0.21776, saving model to /content/dri
Epoch 9/10
4/4 [==============================] - 1s 337ms/step - loss: 0.2513 - accuracy: 1.00

Epoch 00009: val_loss improved from 0.21776 to 0.19593, saving model to /content/dri
Epoch 10/10
4/4 [==============================] - 1s 337ms/step - loss: 0.2729 - accuracy: 1.00

Epoch 00010: val_loss improved from 0.19593 to 0.17666, saving model to /content/dri
```

CLASIFICATION REPORT

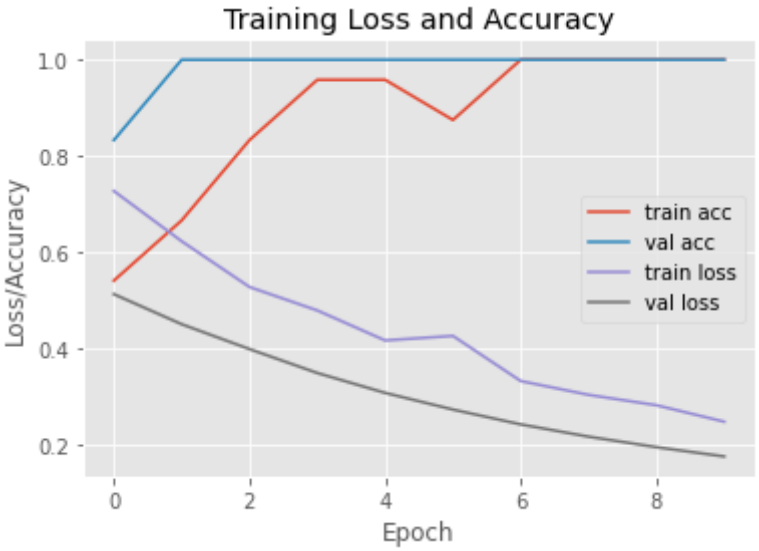
```
# make predictions on the testing set
print("[INFO] evaluating network...")
predIdxs = model_train.predict(testX, batch_size=batch_size_number)

# for each image in the testing set we need to find the index of the
# label with corresponding largest predicted probability
predIdxs = np.argmax(predIdxs, axis=1)

# show a nicely formatted classification report
print(classification_report(testY.argmax(axis=1), predIdxs,
  target_names=lb.classes_))
```

```
[INFO] evaluating network...
              precision    recall  f1-score   support

         edi       1.00      1.00      1.00         3
     unknown       1.00      1.00      1.00         3

    accuracy                           1.00         6
   macro avg       1.00      1.00      1.00         6
weighted avg       1.00      1.00      1.00         6
```

TABLE TRAINING RESULT

```
plt.style.use("ggplot")
plt.figure()

plt.plot(H.history['accuracy'] , label = 'train acc')
plt.plot(H.history['val_accuracy'] , label = 'val acc')
```

```
plt.plot(H.history['loss'] , label = 'train loss')
plt.plot(H.history['val_loss'] , label = 'val loss')

plt.title("Training Loss and Accuracy")
plt.xlabel("Epoch")
plt.ylabel("Loss/Accuracy")
plt.legend()
plt.savefig("/content/drive/MyDrive/FaceRecog/new_dataset/result/result15/result15.png")
```



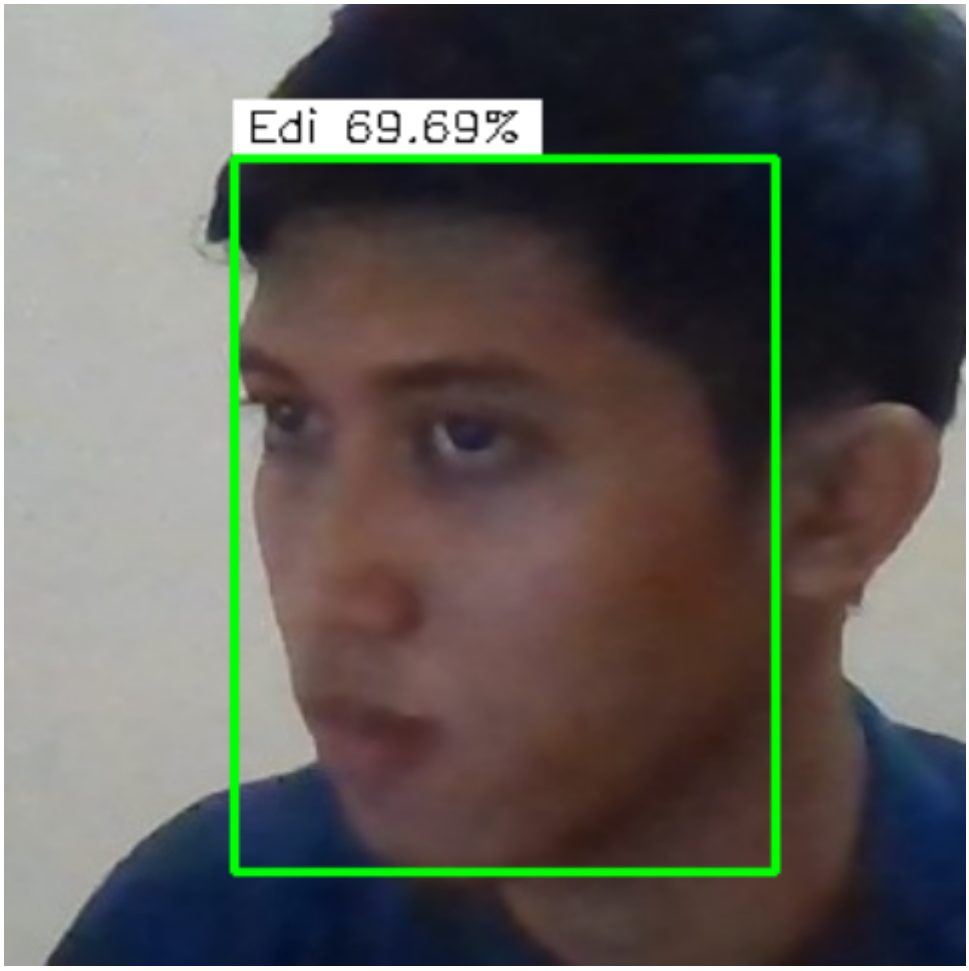
TESTING MODEL

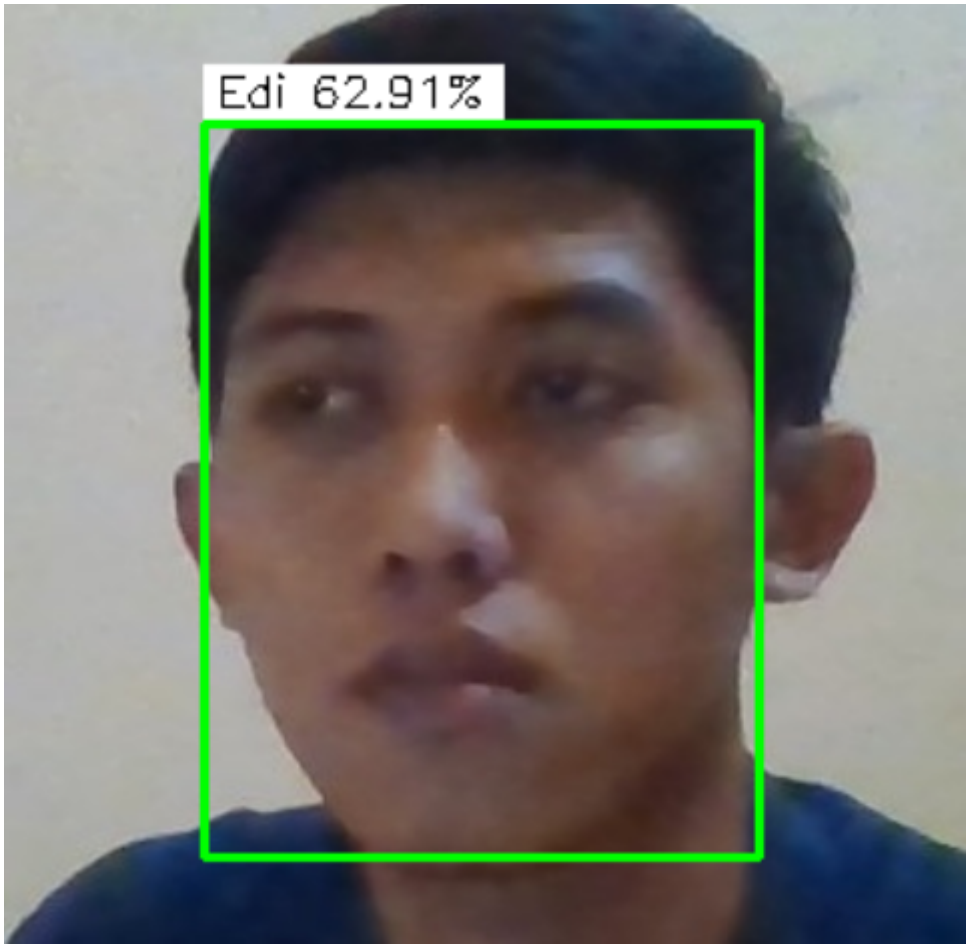
```
from tensorflow.keras.applications.mobilenet_v2 import preprocess_input
from tensorflow.keras.preprocessing.image import img_to_array
from tensorflow.keras.preprocessing import image
from tensorflow.keras.applications import imagenet_utils
from tensorflow.keras.models import load_model
from google.colab.patches import cv2_imshow
import numpy as np
import imutils
import time
import cv2
import os
from os import listdir
from os.path import isfile, join

prototxt_path = "/content/drive/MyDrive/FaceRecog/ssd_face.prototxt"
weights_path = "/content/drive/MyDrive/FaceRecog/ssd_face.caffemodel"

faceNet = cv2.dnn.readNet(prototxt_path, weights_path)

# load the face mask detector model from disk
model_recog = load_model('/content/drive/MyDrive/FaceRecog/new_dataset/result/result15/fac

path = "/content/drive/MyDrive/FaceRecog/new_dataset/test/edi/"
# path = "/content/drive/MyDrive/FaceRecog/new_dataset/test/unknown/"
```

```
    many_file_in_dict = len(os.listdir(path))
    image_file = os.listdir(path)
    # print(image_file)

    for count in range(many_file_in_dict):
      path_image = os.path.join(path+image_file[count])
      input_image = cv2.imread(path_image)

      # cv2_imshow(input_image)

      (h, w) = input_image.shape[:2]
      blob = cv2.dnn.blobFromImage(input_image, 1.0, (224, 224), (104.0, 177.0, 123.0))

      faceNet.setInput(blob)
      detections = faceNet.forward()

      # faces = []
      pred = {"0", "1"}

      for i in range(0, detections.shape[2]):
        confidence = detections[0, 0, i, 2]
        box = detections[0, 0, i, 3:7] * np.array([w, h, w, h])
        (startX, startY, endX, endY) = box.astype("int")

        if confidence > 0.5:
          (startX, startY) = (max(0, startX), max(0, startY))
          (endX, endY) = (min(w - 1, endX), min(h - 1, endY))

          face_detect = cv2.cvtColor(input_image, cv2.COLOR_BGR2RGB)
          face_detect = cv2.resize(face_detect, (224, 224))
          face_detect = img_to_array(face_detect)
          face_detect = preprocess_input(face_detect)
          face_detect = np.expand_dims(face_detect, axis=0)

          preds = model_recog.predict(face_detect)
          label_percent = str("{:.2f}%".format((max(max(preds)))*100))

          preds = np.argmax(preds)

          # facial = pred[str(preds)]

          if preds == 0:
            label = "Edi"
            color = (0, 255, 0)
          if preds == 1:
            label = "Unknown"
            color = (0, 0, 255)

          label_full = label+" "+label_percent
          label_size = cv2.getTextSize(label_full, cv2.FONT_HERSHEY_SIMPLEX, 0.5, 1)
          cv2.rectangle(input_image, (startX, startY), (endX, endY), color, 2)
          cv2.rectangle(input_image, (startX, startY - 20), ((startX + label_size[0][0]) + 10,
          cv2.putText(input_image, label_full, (startX + 4, startY - 6), cv2.FONT_HERSHEY_SIMF

      image_resize = cv2.resize(input_image, (480, 480))
```

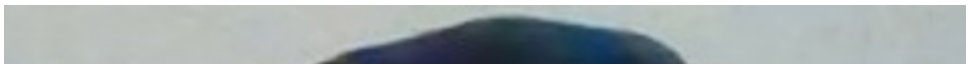
```
        # print(preds)
        cv2_imshow(image_resize)
        cv2.imwrite("/content/drive/MyDrive/FaceRecog/new_dataset/result/result15/test"+str(cour
        time.sleep(1)
        print('\n')
```

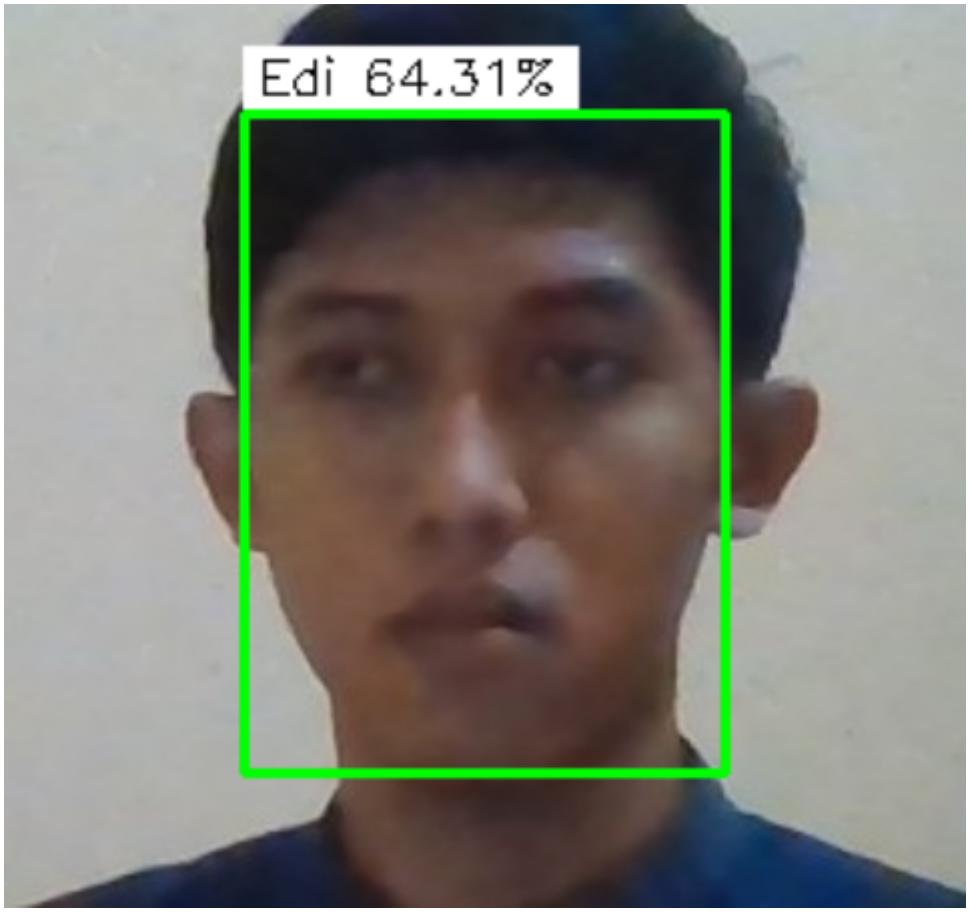

Unknown 64.17%

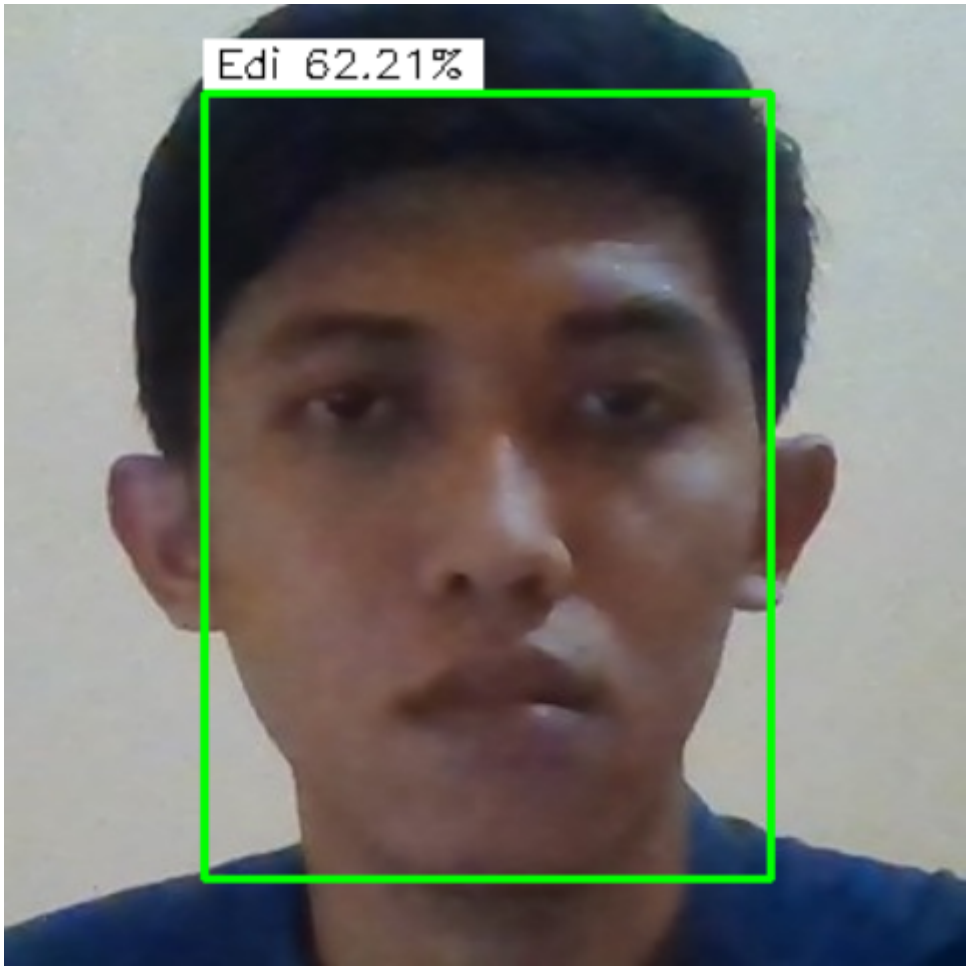

Edi 69.69%

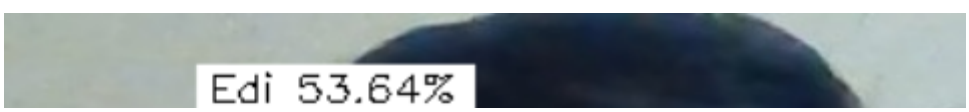

Edi 62.91%

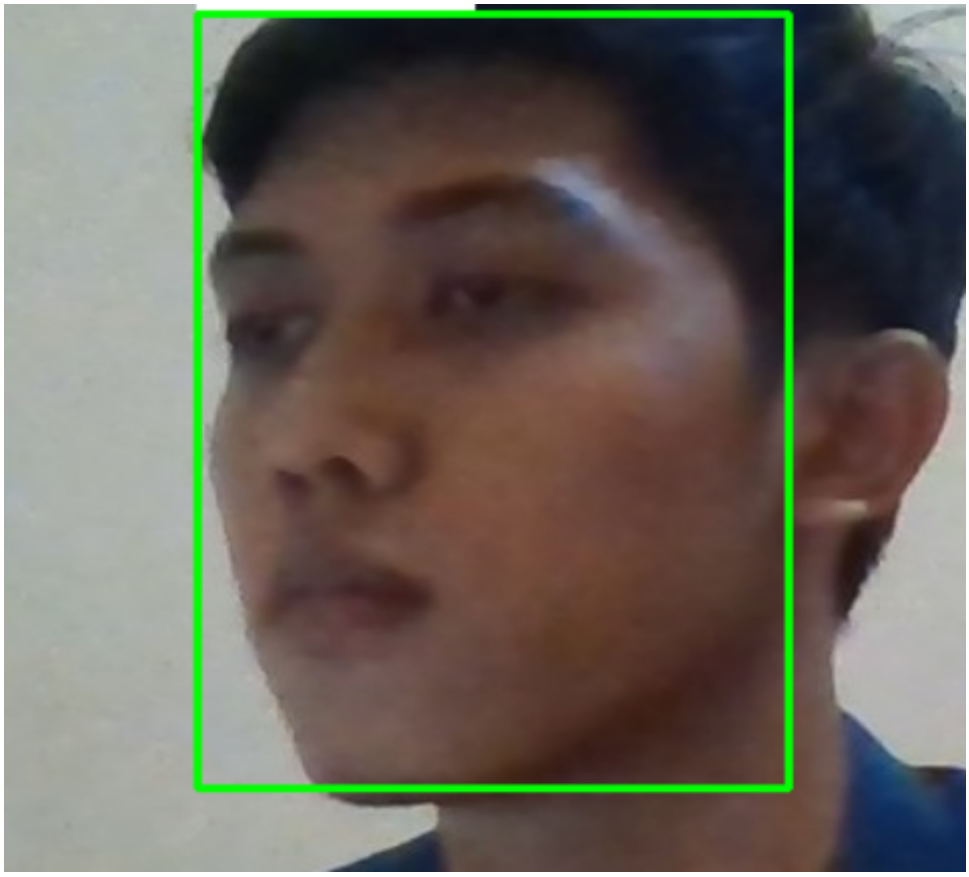

Unknown 51.71%

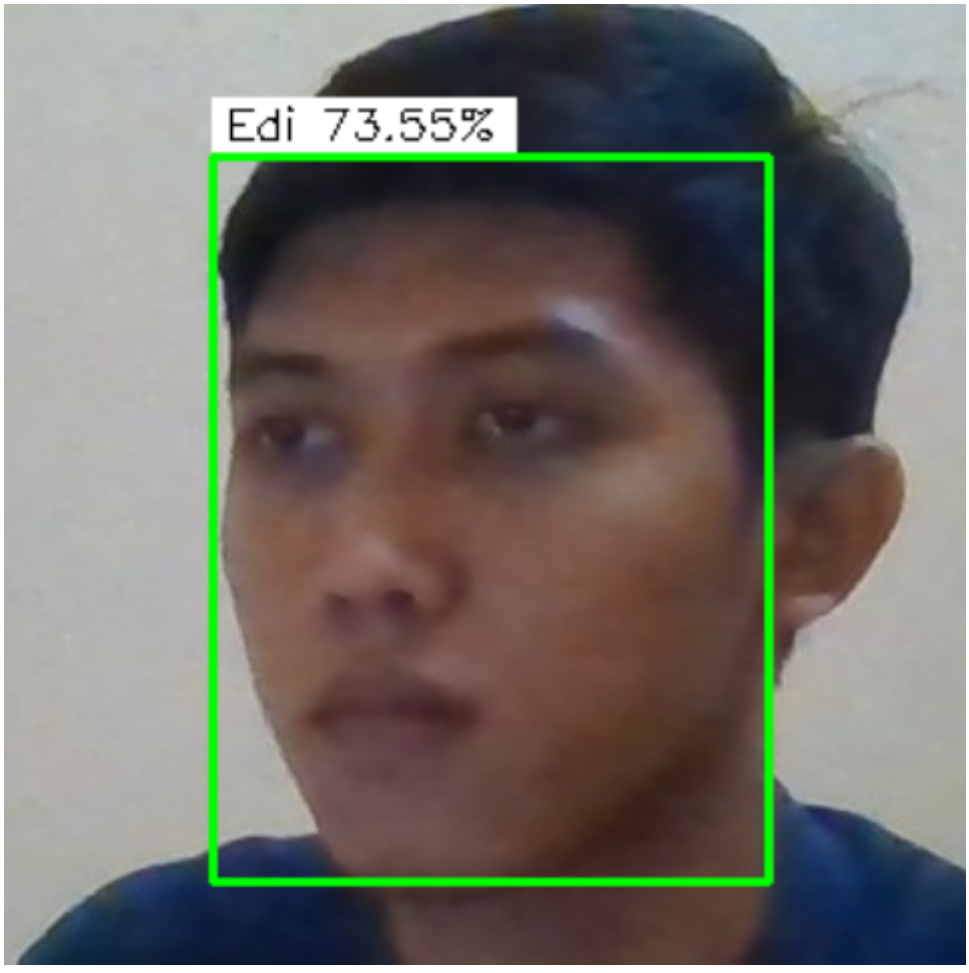


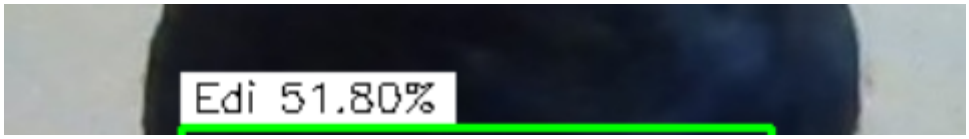

Edi 64.31%

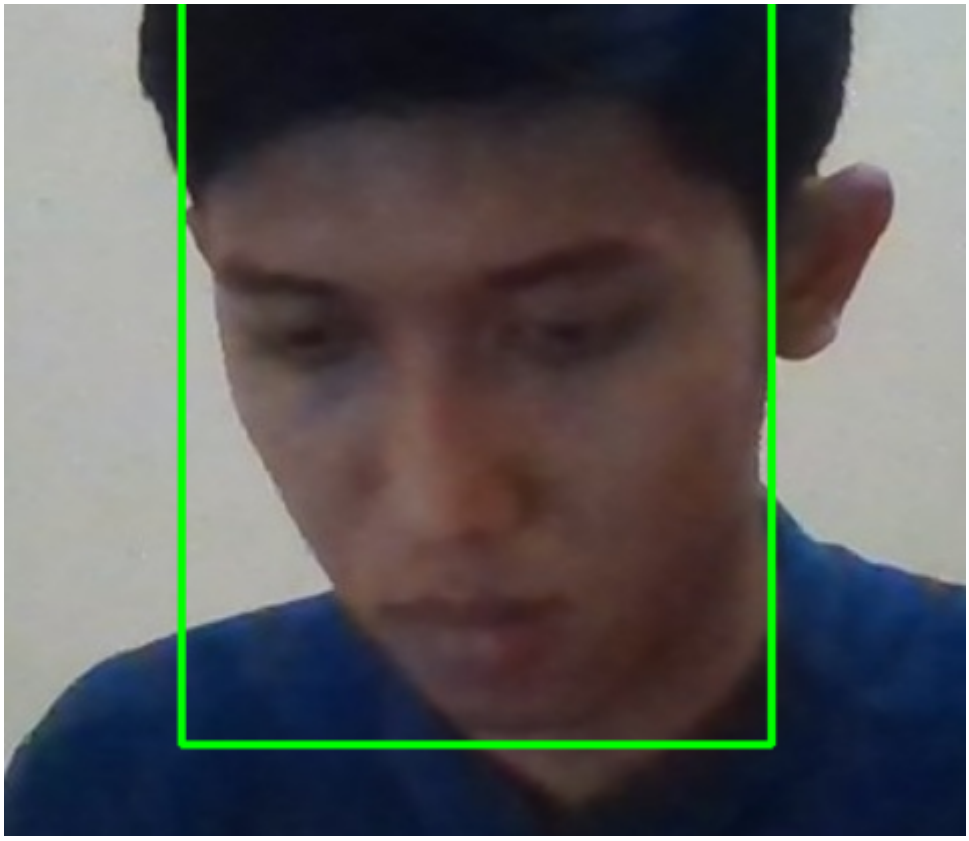

Edi 62.21%


Edi 53.64%

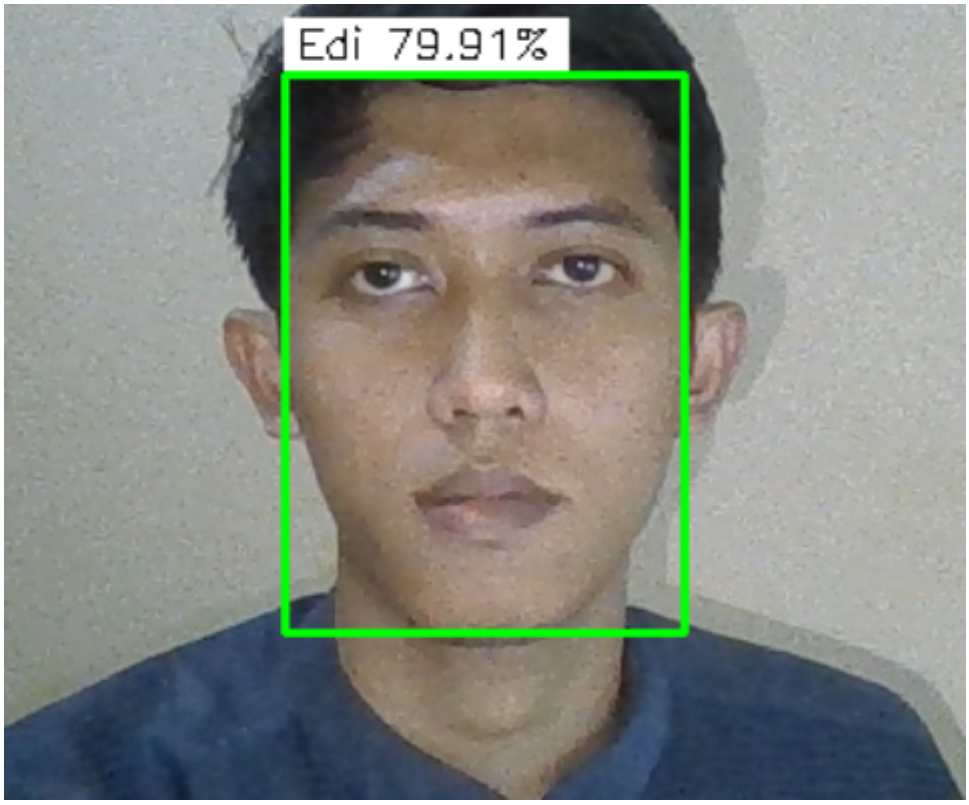


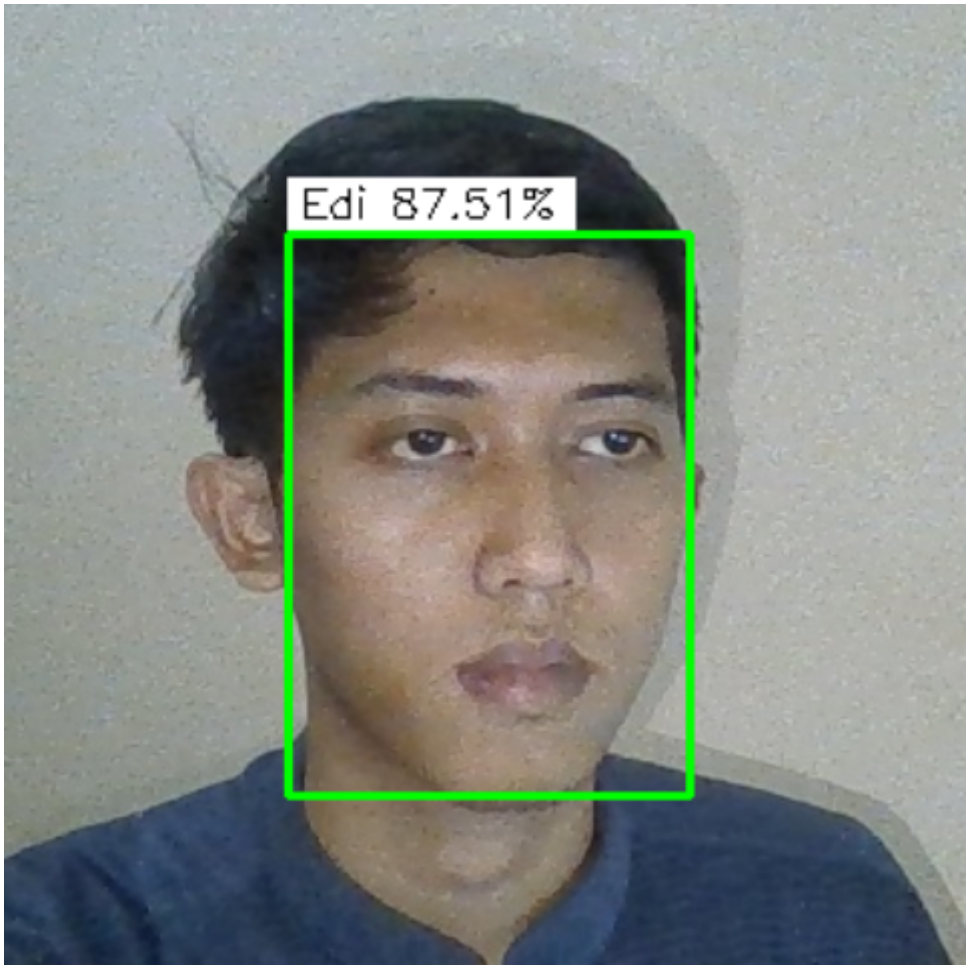

Edi 73.55%


Edi 51.80%

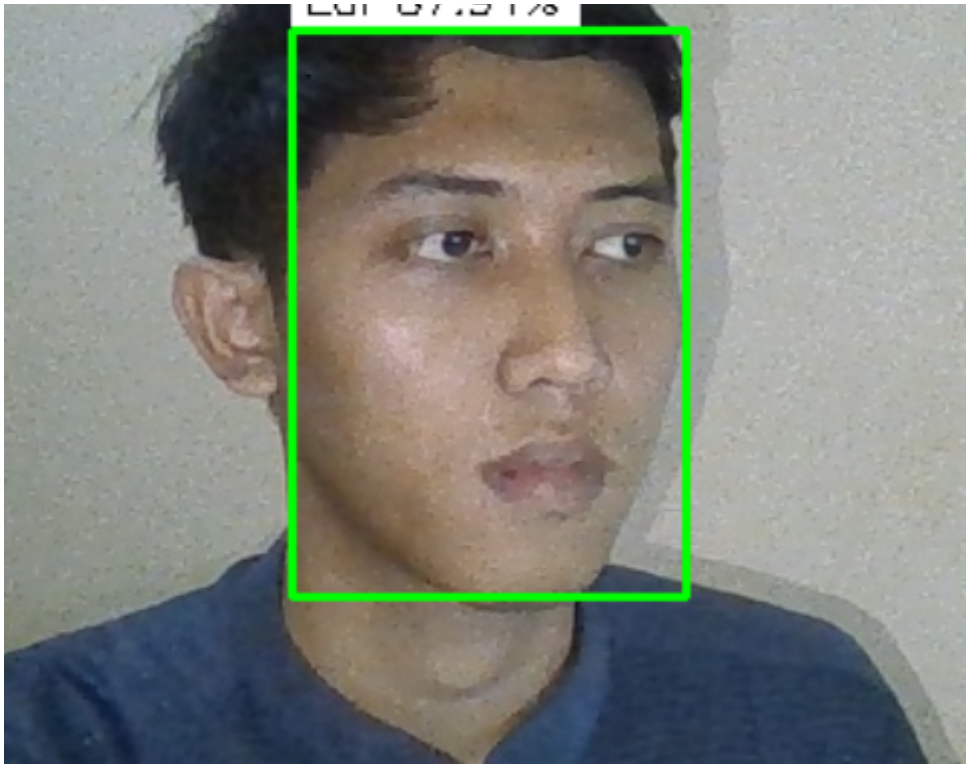


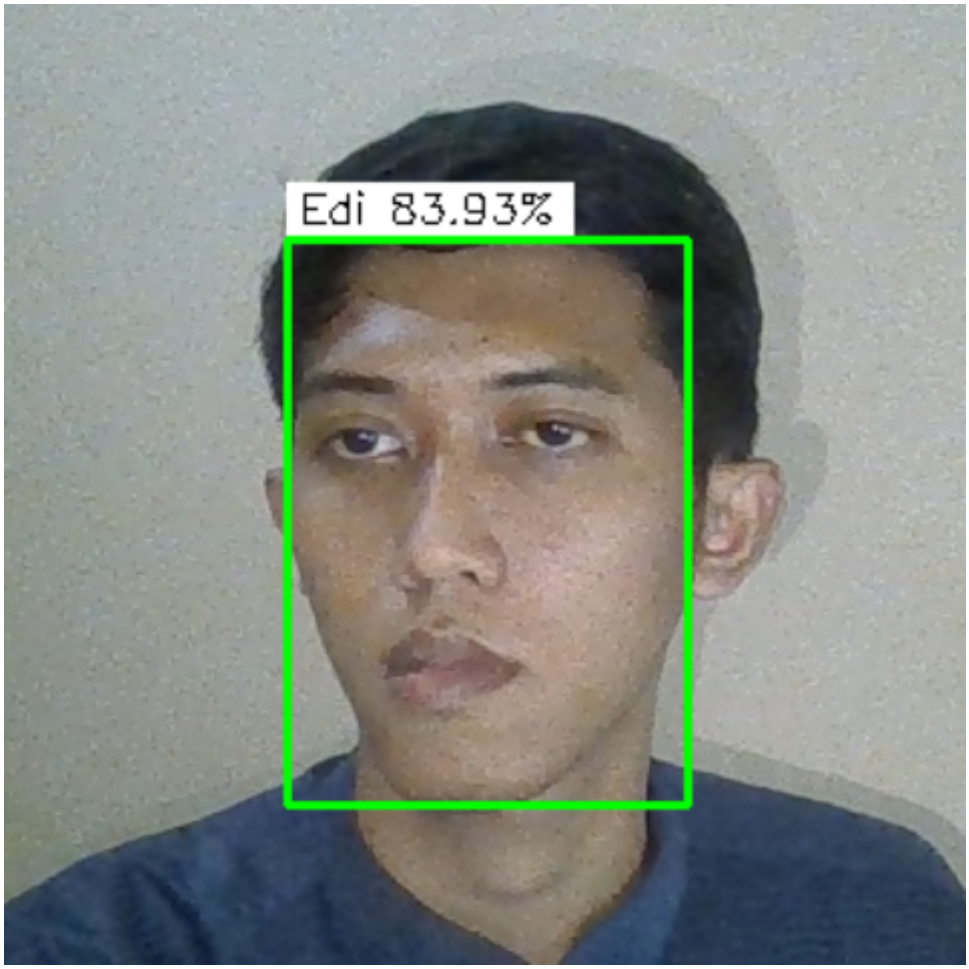

Edi 61.01%

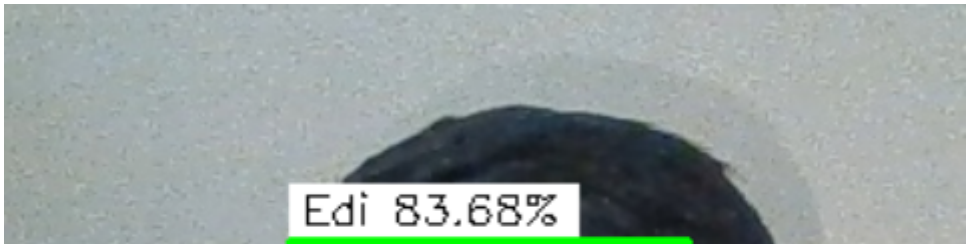

Edi 79.91%

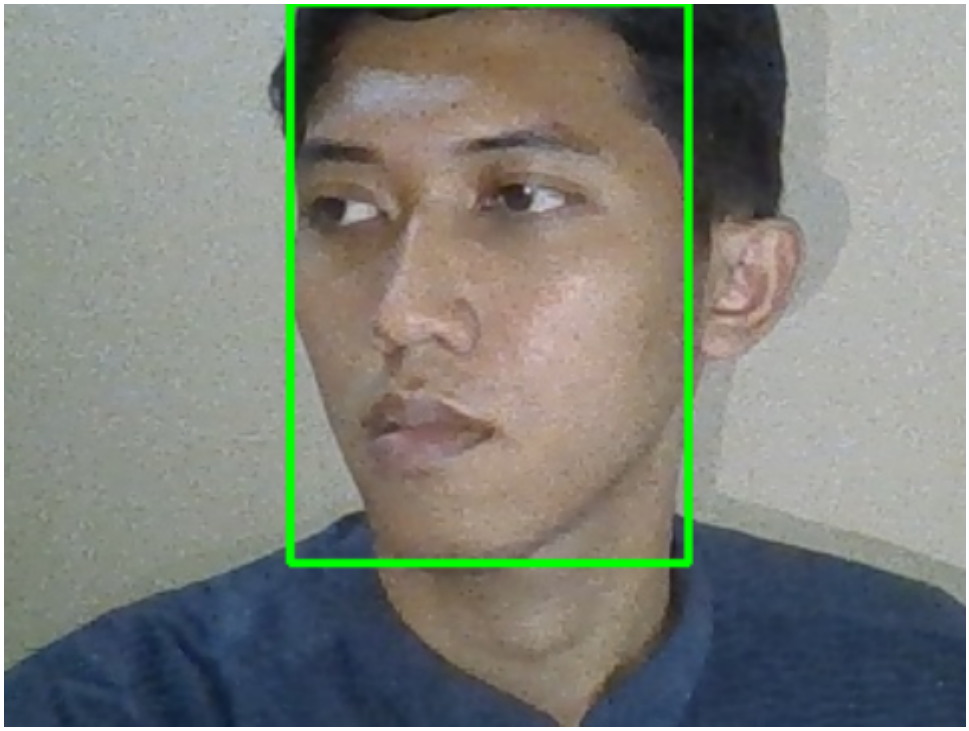

Edi 87.51%

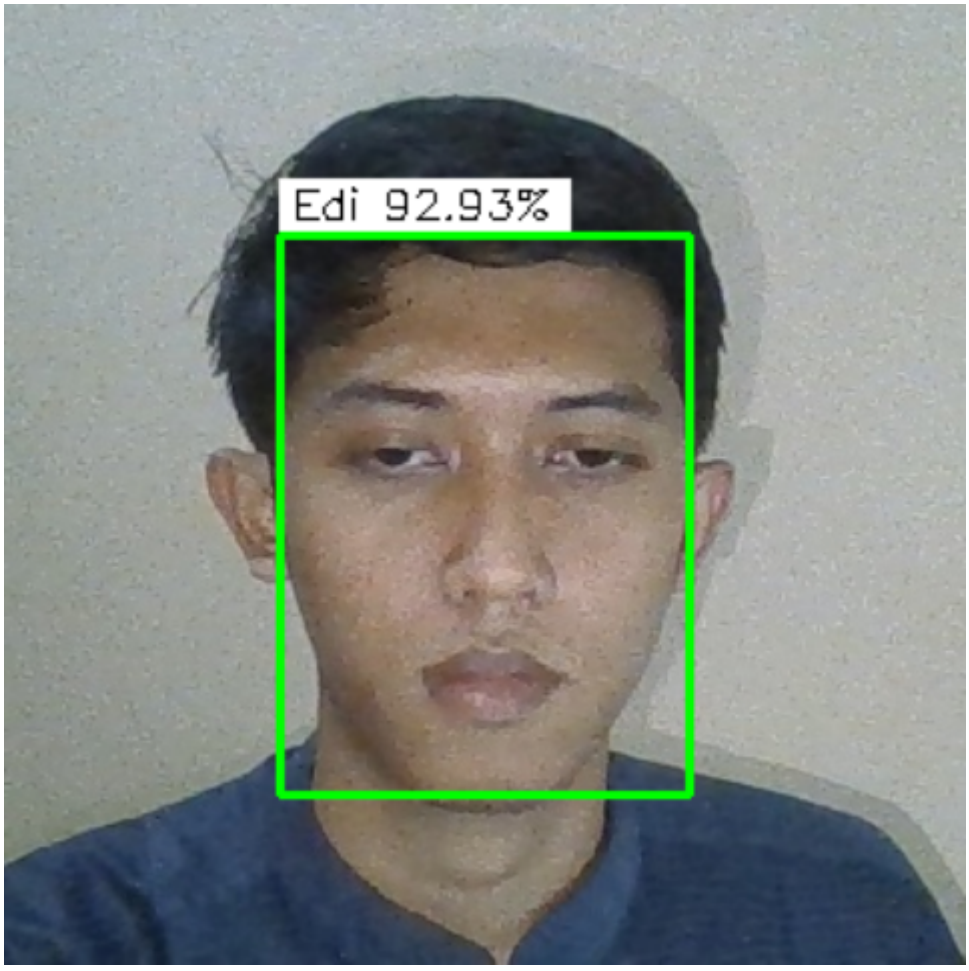

Edi 87.91%



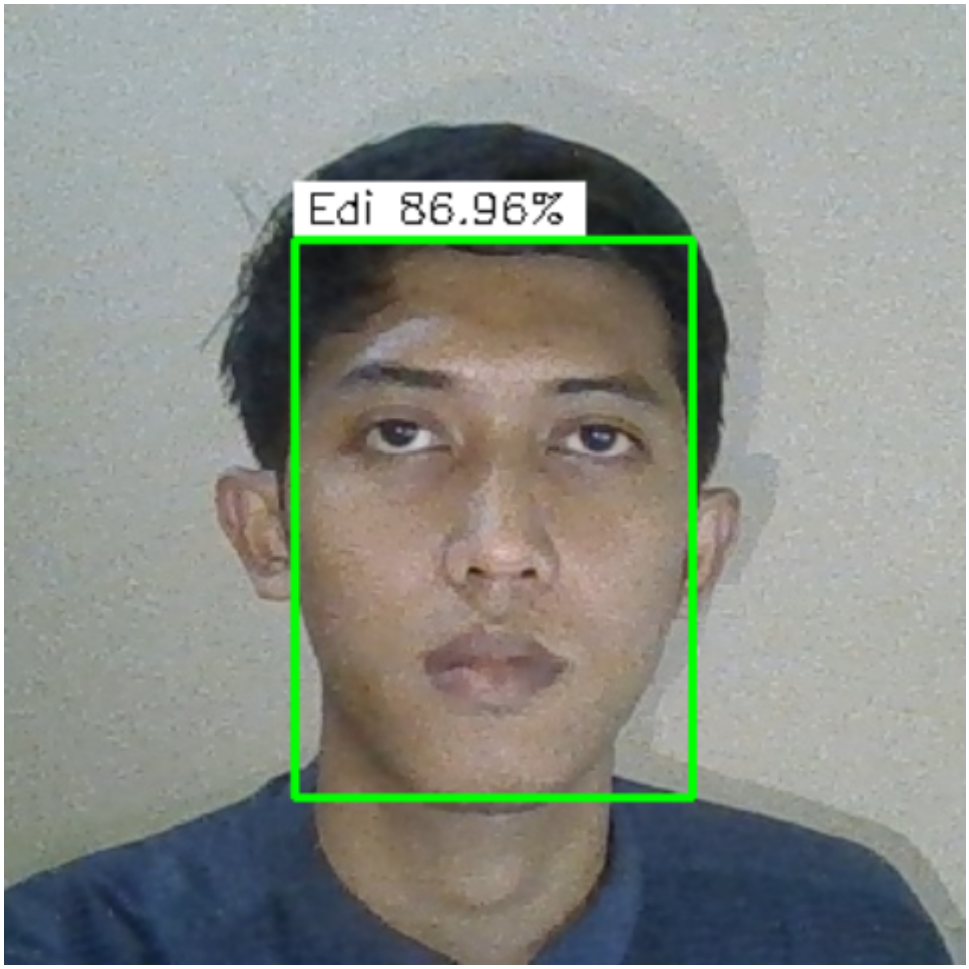

Edi 83.93%


Edi 83.68%




Edi 92.93%


Edi 90.38%

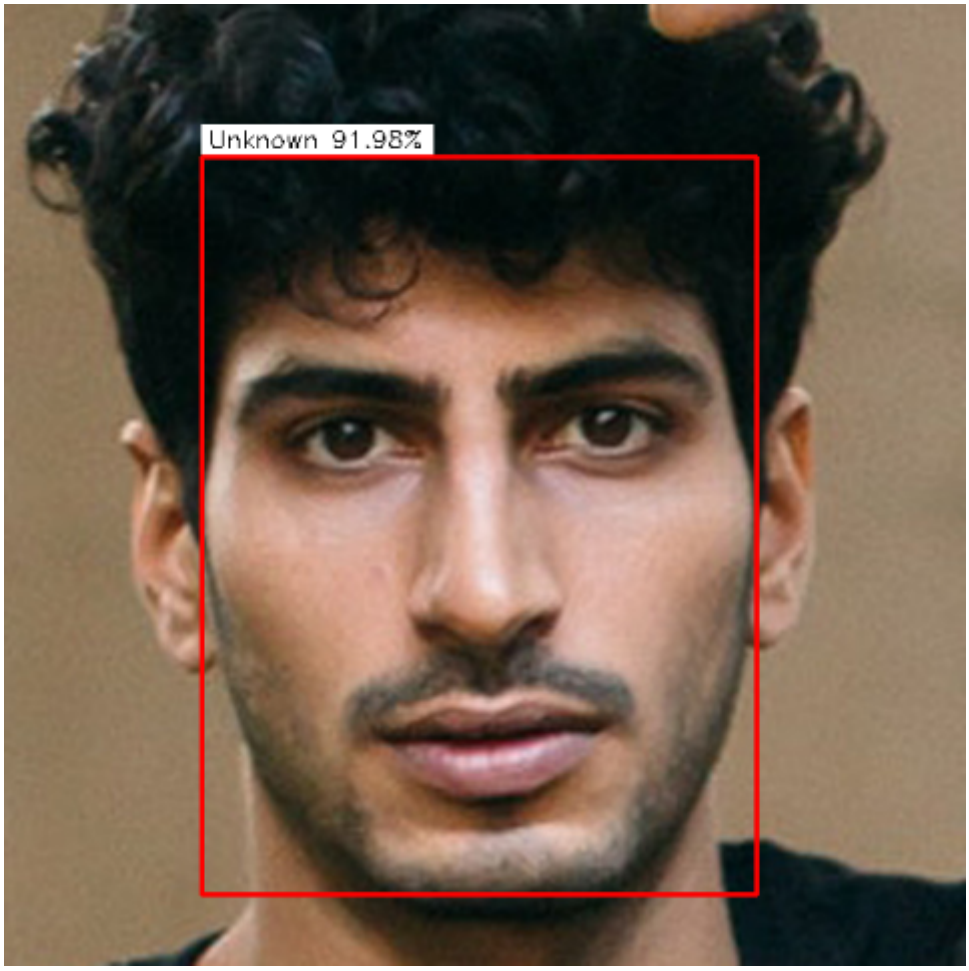


```
from tensorflow.keras.applications.mobilenet_v2 import preprocess_input
from tensorflow.keras.preprocessing.image import img_to_array
```

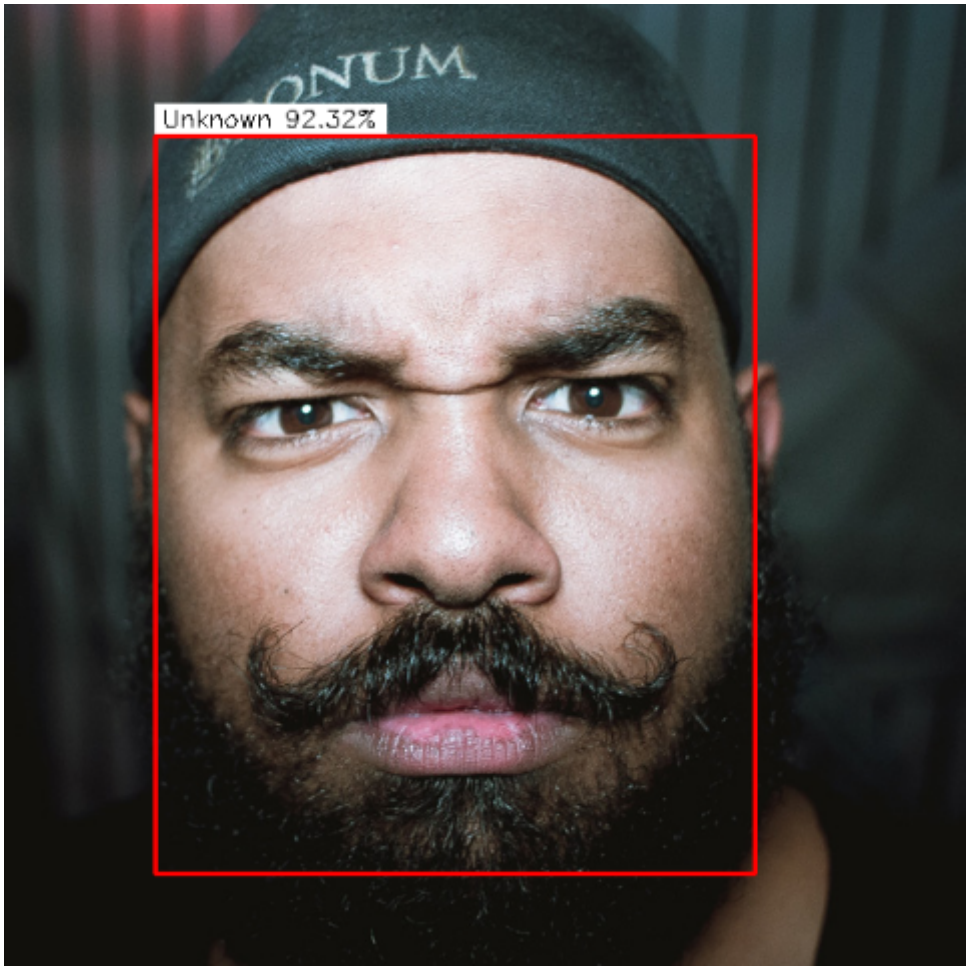
```
from tensorflow.keras.preprocessing import image
from tensorflow.keras.applications import imagenet_utils
from tensorflow.keras.models import load_model
from google.colab.patches import cv2_imshow
import numpy as np
import imutils
import time
import cv2
import os
from os import listdir
from os.path import isfile, join

prototxt_path = "/content/drive/MyDrive/FaceRecog/ssd_face.prototxt"
weights_path = "/content/drive/MyDrive/FaceRecog/ssd_face.caffemodel"

faceNet = cv2.dnn.readNet(prototxt_path, weights_path)

# load the face mask detector model from disk
model_recog = load_model('/content/drive/MyDrive/FaceRecog/new_dataset/result/result15/fac

# path = "/content/drive/MyDrive/FaceRecog/new_dataset/test/edi/"
path = "/content/drive/MyDrive/FaceRecog/new_dataset/test/unknown/"

many_file_in_dict = len(os.listdir(path))
image_file = os.listdir(path)
# print(image_file)

for count in range(many_file_in_dict):
  path_image = os.path.join(path+image_file[count])
  input_image = cv2.imread(path_image)

  # cv2_imshow(input_image)

  (h, w) = input_image.shape[:2]
  blob = cv2.dnn.blobFromImage(input_image, 1.0, (224, 224), (104.0, 177.0, 123.0))

  faceNet.setInput(blob)
  detections = faceNet.forward()

  # faces = []
  pred = {"0", "1"}

  for i in range(0, detections.shape[2]):
    confidence = detections[0, 0, i, 2]
    box = detections[0, 0, i, 3:7] * np.array([w, h, w, h])
    (startX, startY, endX, endY) = box.astype("int")

    if confidence > 0.5:
      (startX, startY) = (max(0, startX), max(0, startY))
      (endX, endY) = (min(w - 1, endX), min(h - 1, endY))

      face_detect = cv2.cvtColor(input_image, cv2.COLOR_BGR2RGB)
      face_detect = cv2.resize(face_detect, (224, 224))
      face_detect = img_to_array(face_detect)
      face_detect = preprocess_input(face_detect)
```

```
      face_detect = np.expand_dims(face_detect, axis=0)

      preds = model_recog.predict(face_detect)
      label_percent = str("{:.2f}%".format((max(max(preds)))*100))

      preds = np.argmax(preds)

      # facial = pred[str(preds)]

      if preds == 0:
        label = "Edi"
        color = (0, 255, 0)
      if preds == 1:
        label = "Unknown"
        color = (0, 0, 255)

      label_full = label+" "+label_percent
      label_size = cv2.getTextSize(label_full, cv2.FONT_HERSHEY_SIMPLEX, 0.5, 1)
      cv2.rectangle(input_image, (startX, startY), (endX, endY), color, 2)
      cv2.rectangle(input_image, (startX, startY - 20), ((startX + label_size[0][0]) + 10,
      cv2.putText(input_image, label_full, (startX + 4, startY - 6), cv2.FONT_HERSHEY_SIMF

  image_resize = cv2.resize(input_image, (480, 480))
  # print(preds)
  cv2_imshow(image_resize)
  cv2.imwrite("/content/drive/MyDrive/FaceRecog/new_dataset/result/result15/test"+str(cour
  time.sleep(1)
  print('\n')
```

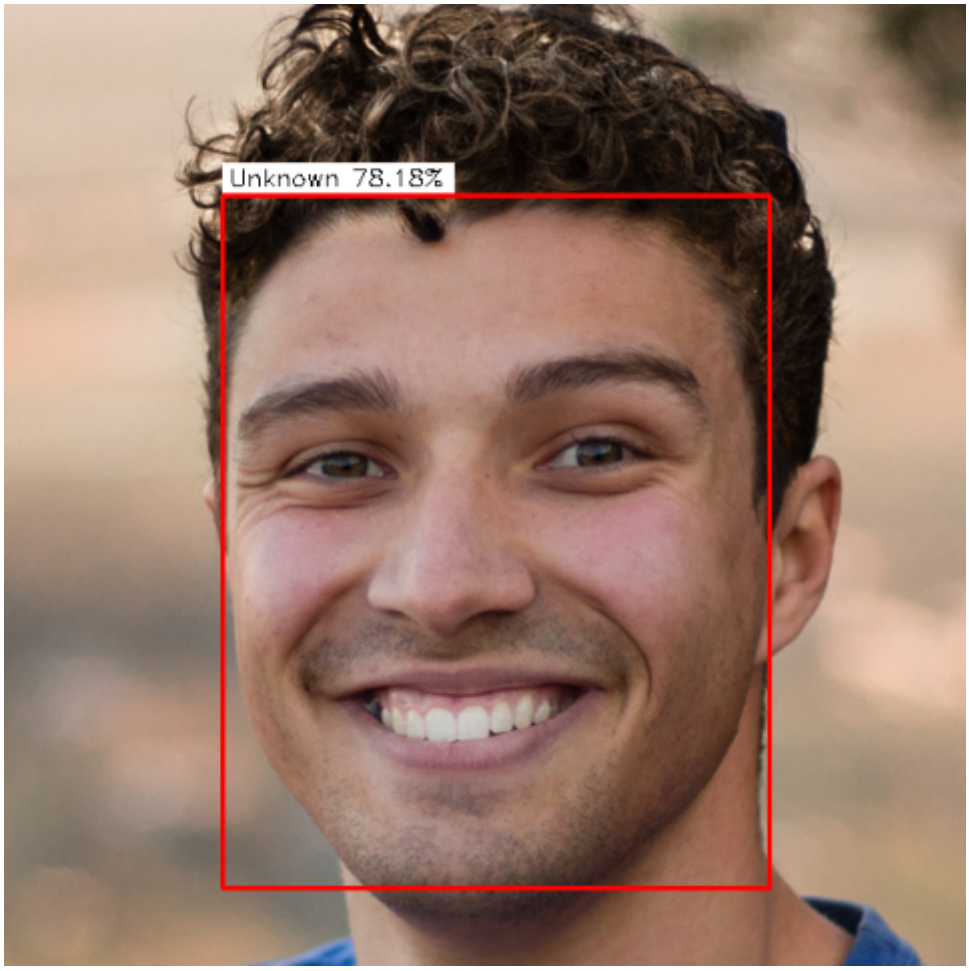

Unknown 76.51%

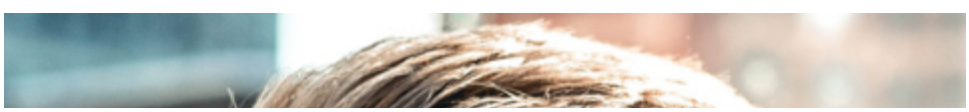

Unknown 91.98%

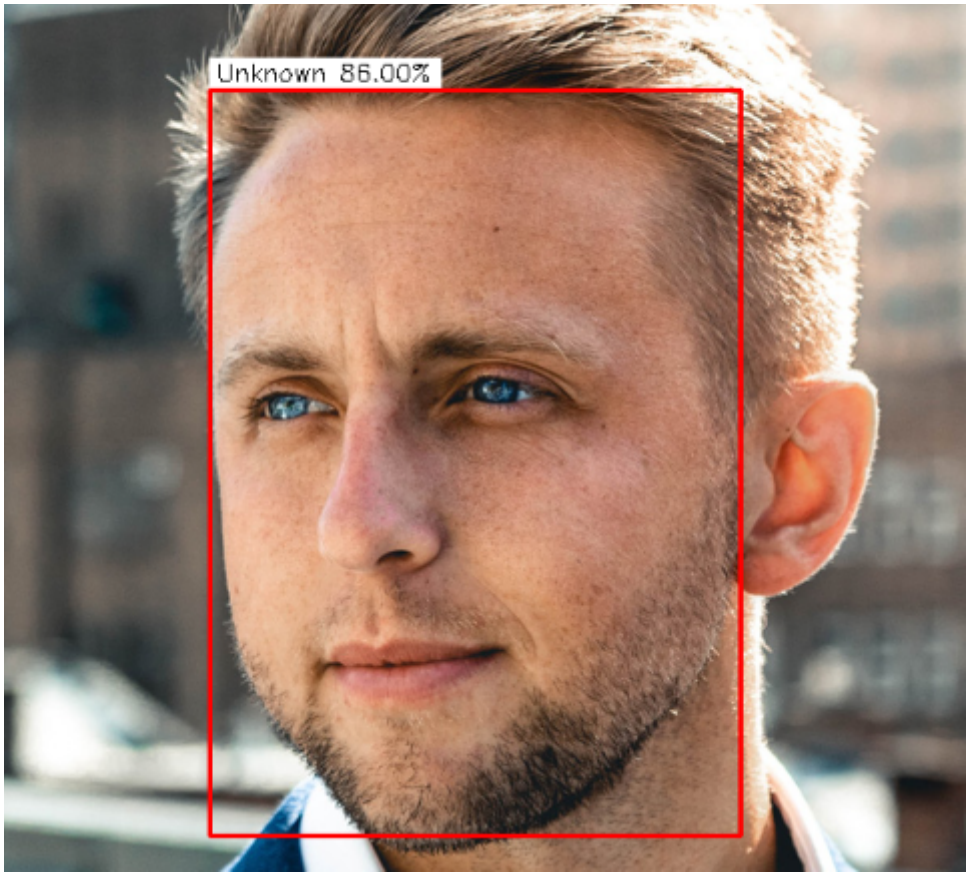

Unknown 90.09%

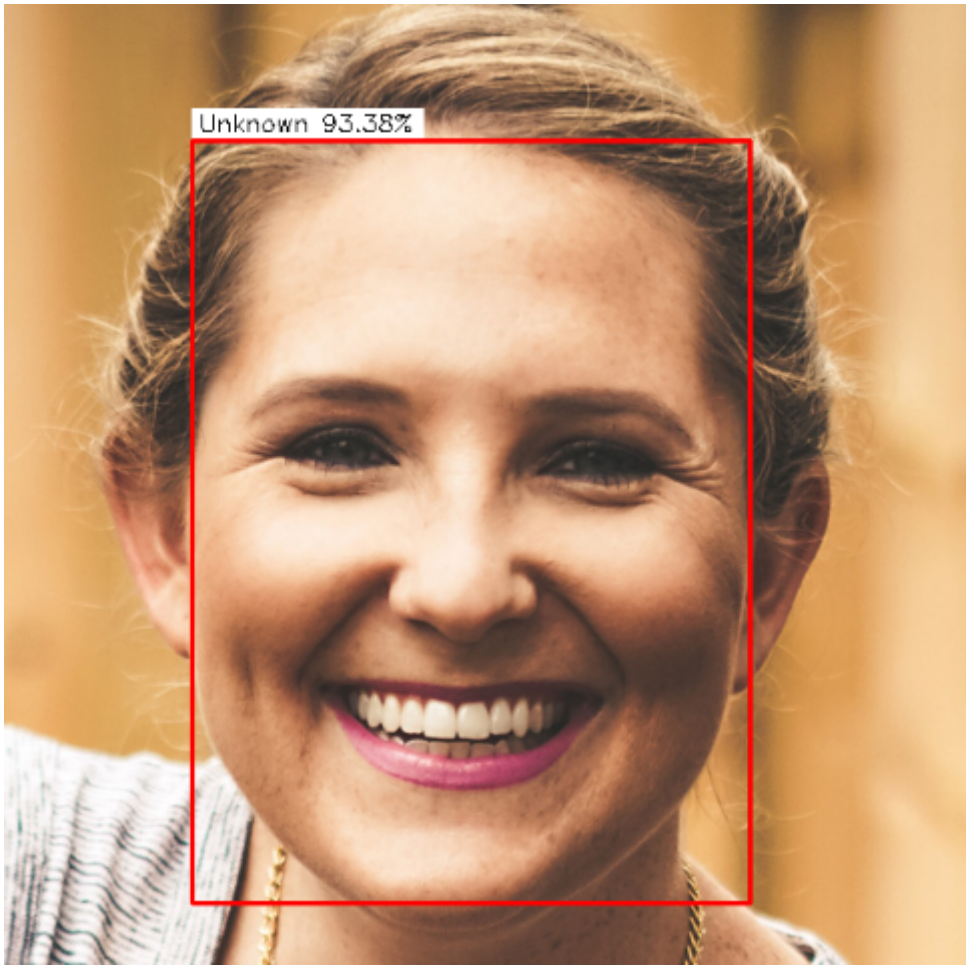

Unknown 92.32%


Unknown 90.72%

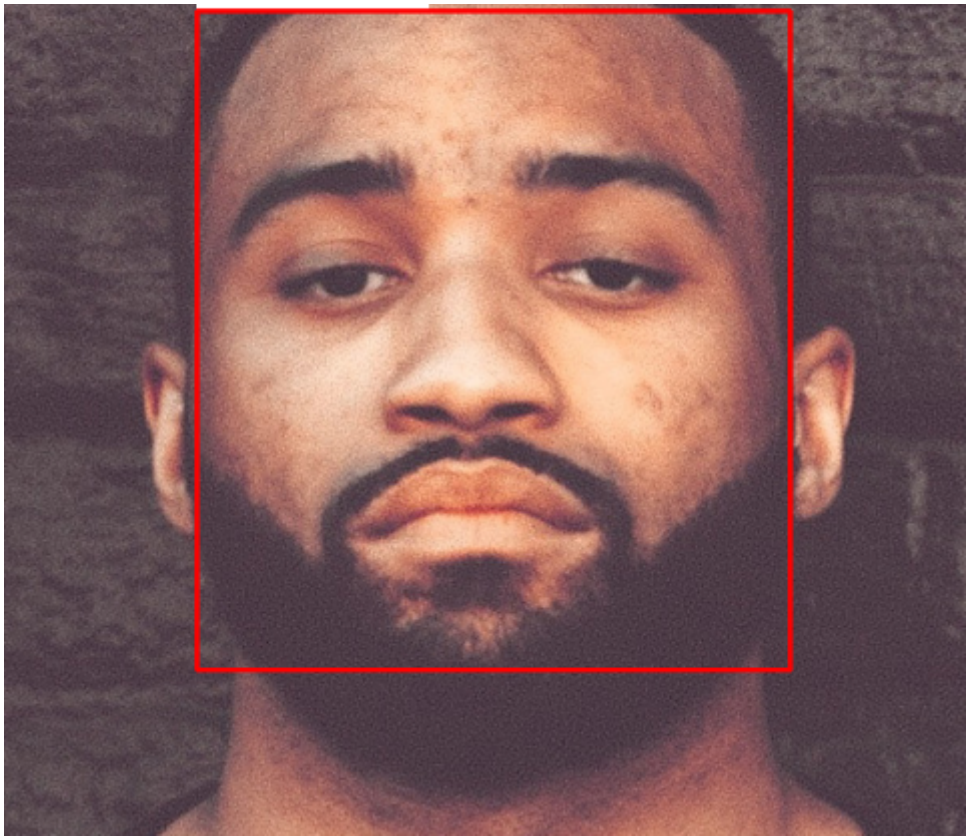

Unknown 78.18%

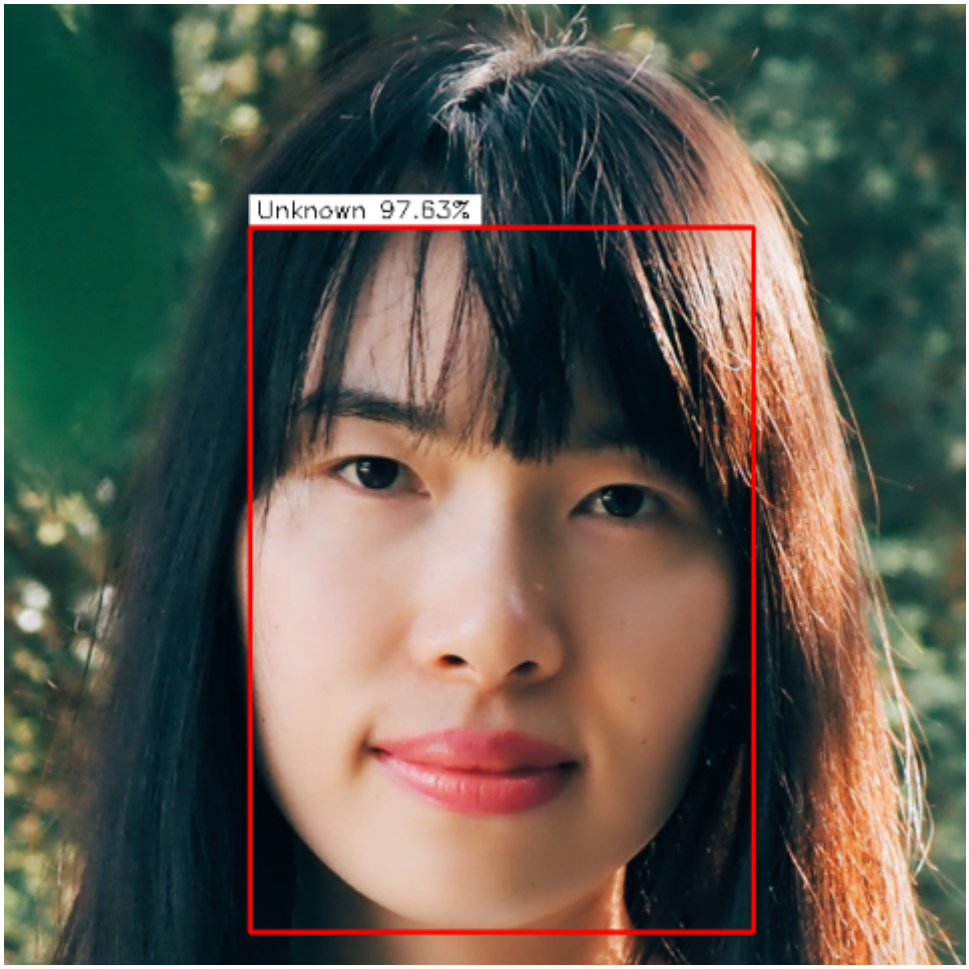



Unknown 86.00%

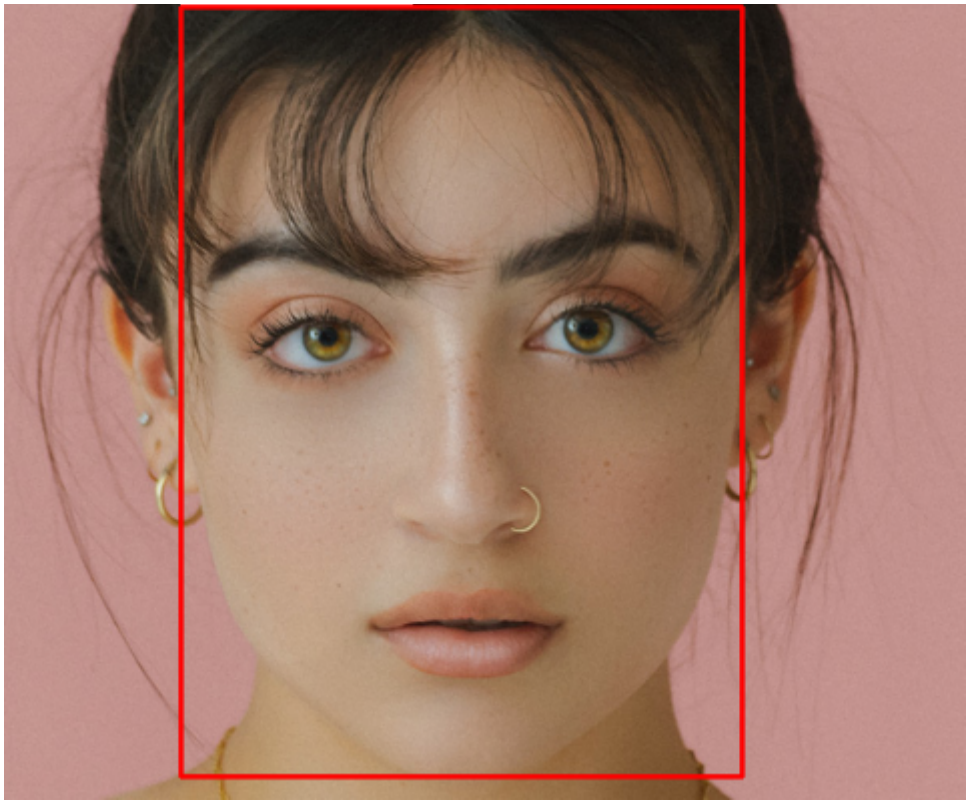

Unknown 93.38%

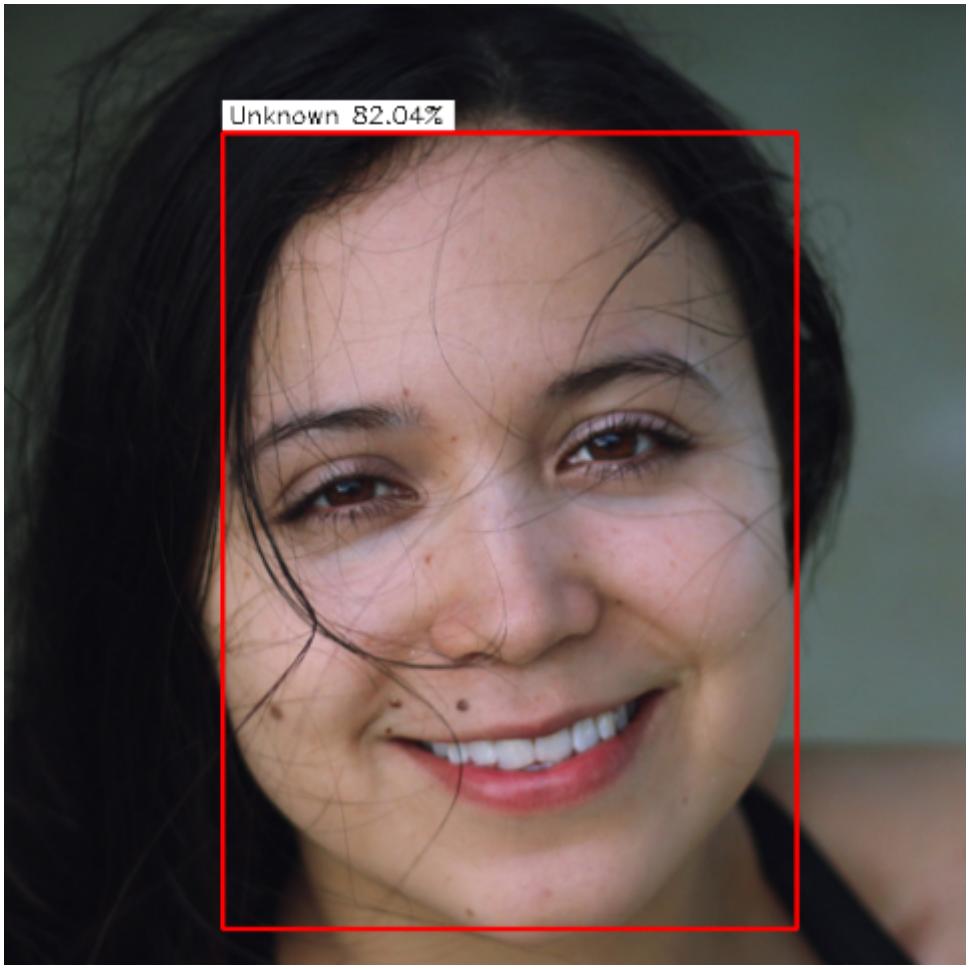

Unknown 65.48%

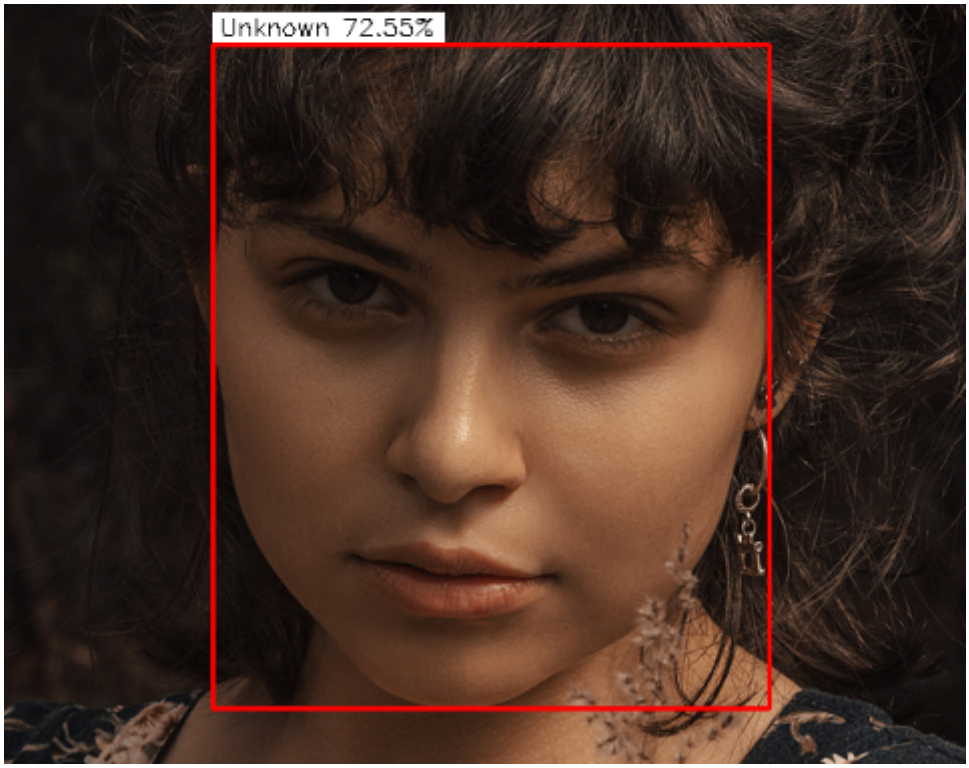


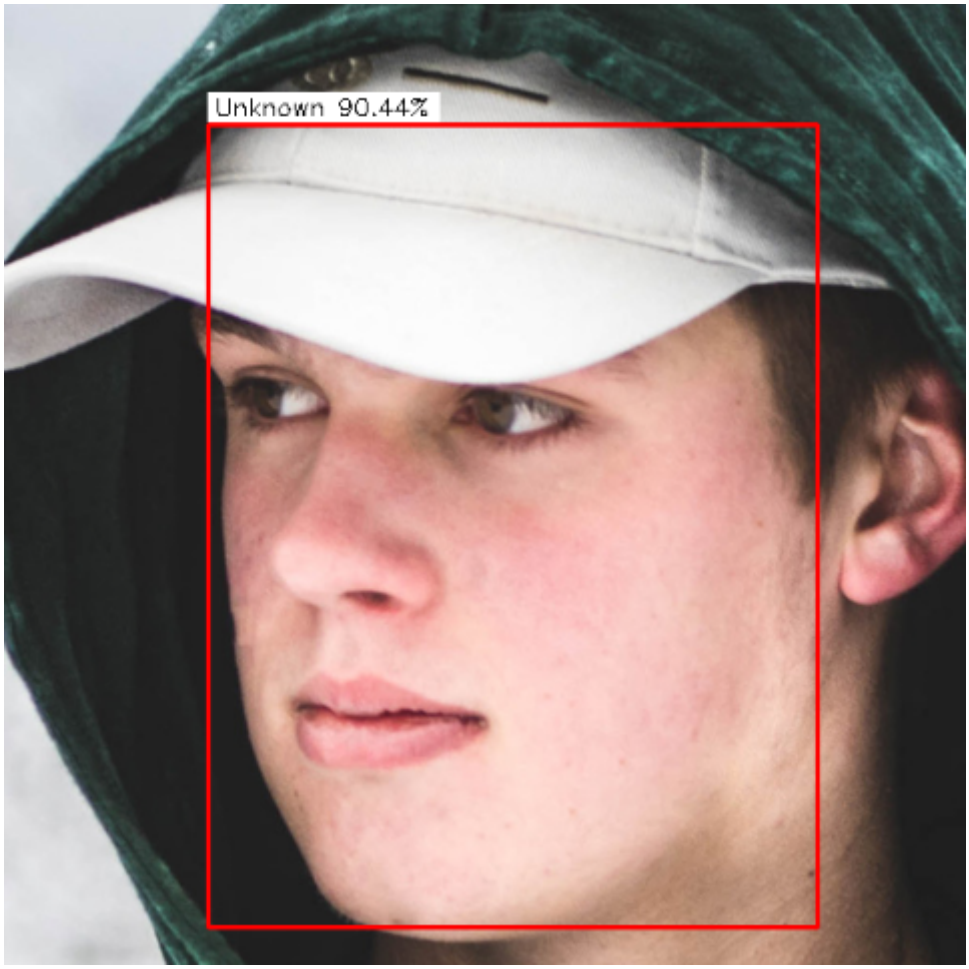

Unknown 97.63%


Unknown 78.51%




Unknown 82.04%

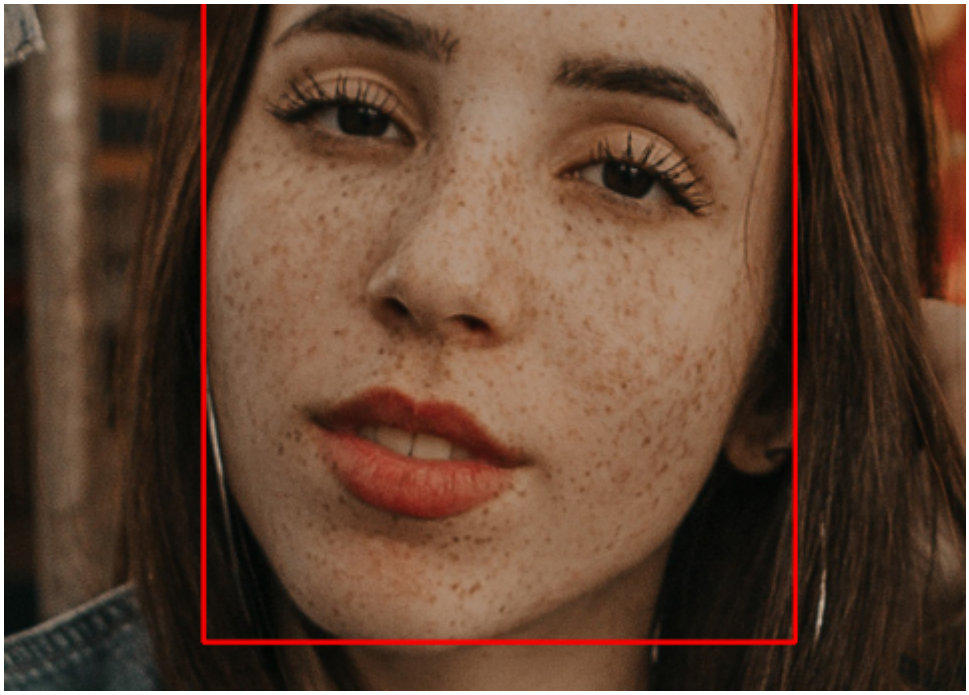

Unknown 72.55%

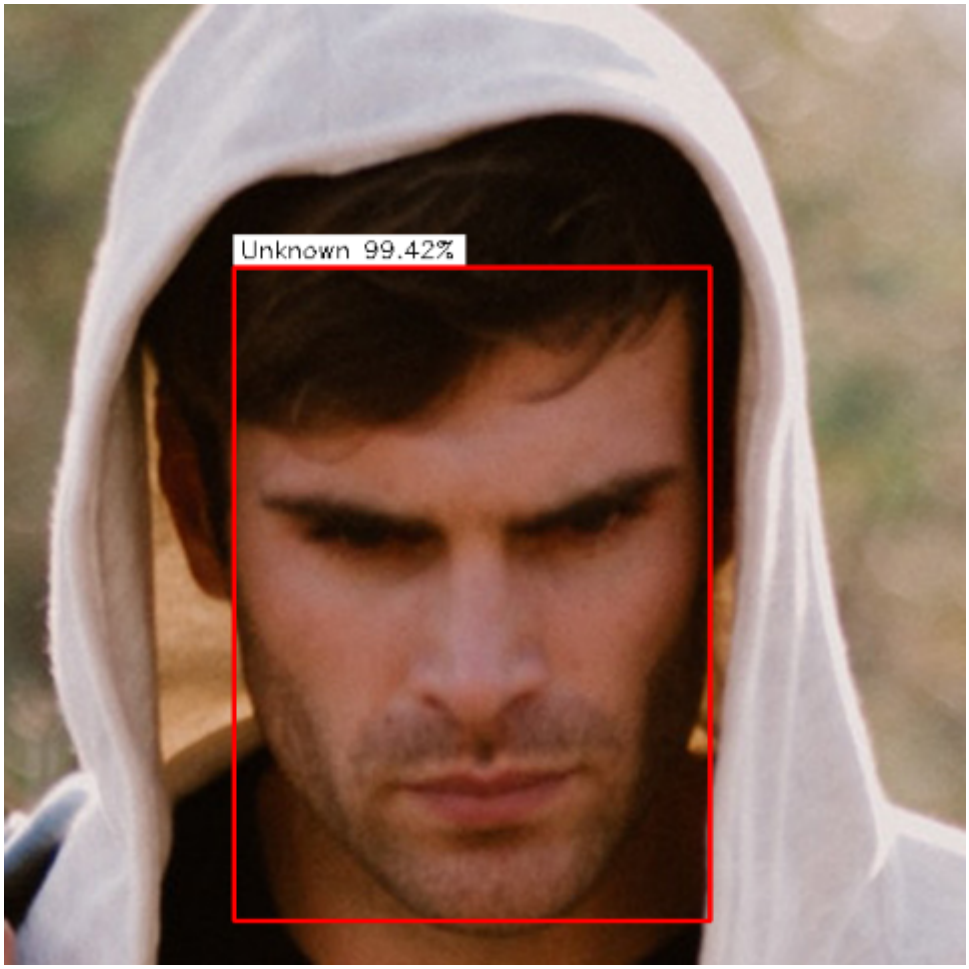

Unknown 90.44%


Unknown 66.56%

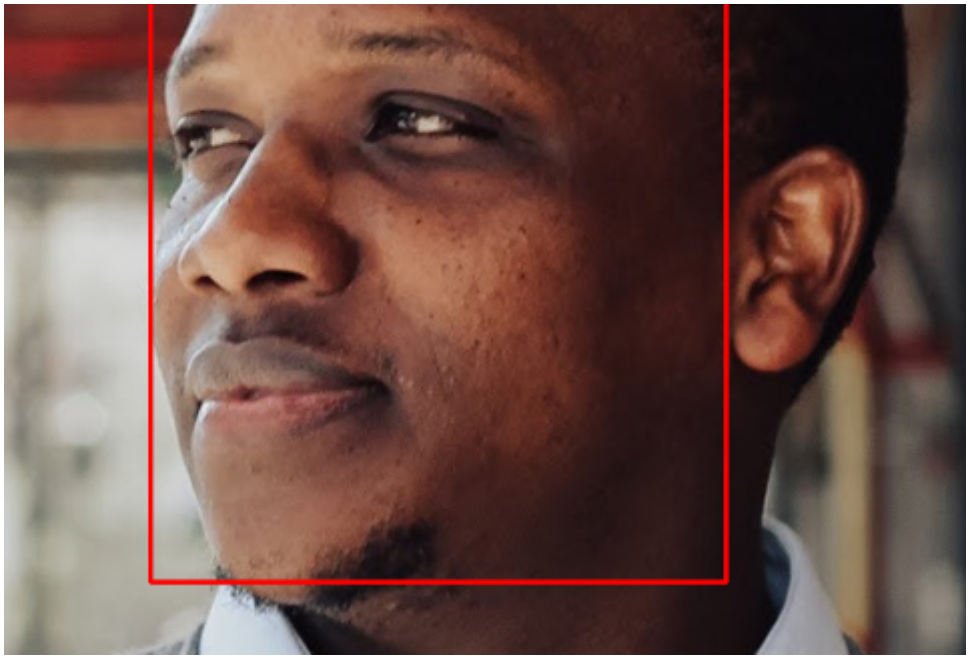

Unknown 95.93%

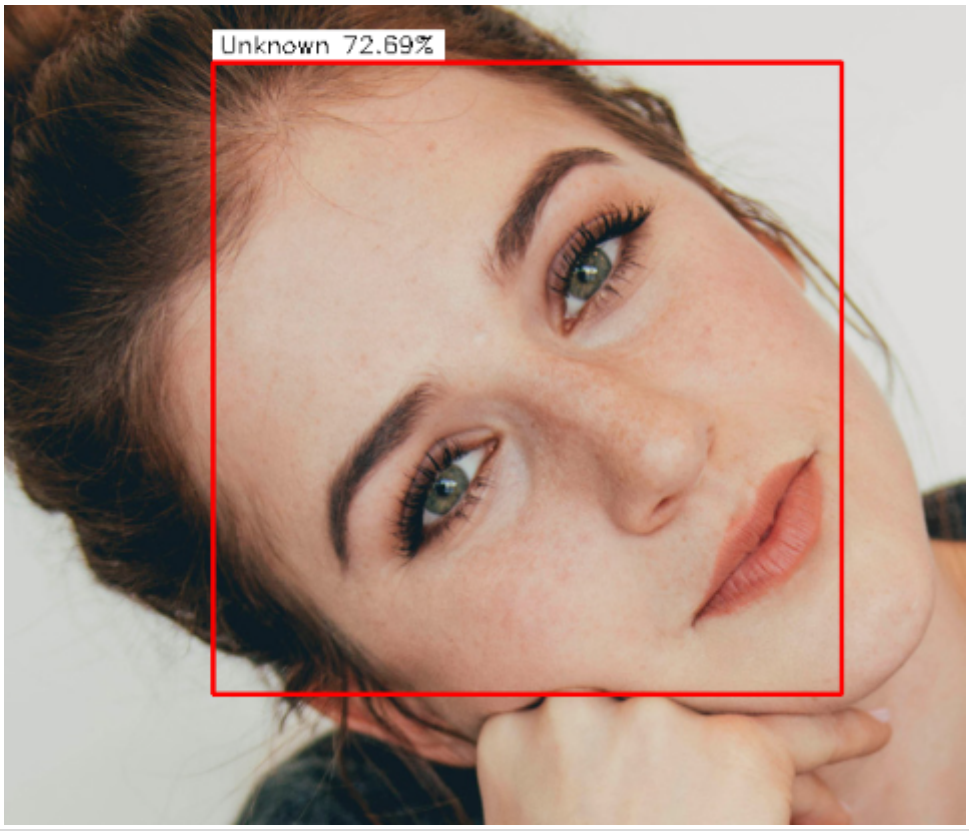

Unknown 93.10%




Unknown 99.42%


Unknown 56.00%




Unknown 72.69%