

▼ 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, 224, 3))

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()

block_14_depthwise_BN (BatchNormal (None, 7, 7, 960)      3840      block_14_depthwis...
block_14_depthwise_relu (ReLU) (None, 7, 7, 960)          0          block_14_depthwis...
block_14_project (Conv2D) (None, 7, 7, 160)        153600      block_14_depthwis...
block_14_project_BN (BatchNormal (None, 7, 7, 160)      640      block_14_project[...
block_14_add (Add) (None, 7, 7, 160)          0          block_13_project...
block_14_project...
block_15_expand (Conv2D) (None, 7, 7, 960)        153600      block_14_add[0][e...
```

block_15_expand_BN (BatchNormal (None, 7, 7, 960)	3840	block_15_expand[0]
block_15_expand_relu (ReLU) (None, 7, 7, 960)	0	block_15_expand_B
block_15_depthwise (DepthwiseCo (None, 7, 7, 960)	8640	block_15_expand_r
block_15_depthwise_BN (BatchNor (None, 7, 7, 960)	3840	block_15_depthwis
block_15_depthwise_relu (ReLU) (None, 7, 7, 960)	0	block_15_depthwis
block_15_project (Conv2D) (None, 7, 7, 160)	153600	block_15_depthwis
block_15_project_BN (BatchNorma (None, 7, 7, 160)	640	block_15_project[
block_15_add (Add) (None, 7, 7, 160)	0	block_14_add[0][0]
block_15_project_		block_15_project_
block_16_expand (Conv2D) (None, 7, 7, 960)	153600	block_15_add[0][e
block_16_expand_BN (BatchNormal (None, 7, 7, 960)	3840	block_16_expand[0]
block_16_expand_relu (ReLU) (None, 7, 7, 960)	0	block_16_expand_B
block_16_depthwise (DepthwiseCo (None, 7, 7, 960)	8640	block_16_expand_r
block_16_depthwise_BN (BatchNor (None, 7, 7, 960)	3840	block_16_depthwis
block_16_depthwise_relu (ReLU) (None, 7, 7, 960)	0	block_16_depthwis
block_16_project (Conv2D) (None, 7, 7, 320)	307200	block_16_depthwis
block_16_project_BN (BatchNorma (None, 7, 7, 320)	1280	block_16_project[
Conv_1 (Conv2D) (None, 7, 7, 1280)	409600	block_16_project_
Conv_1_bn (BatchNormalization) (None, 7, 7, 1280)	5120	Conv_1[0][0]
out_relu (ReLU) (None, 7, 7, 1280)	0	Conv_1_bn[0][0]
global_average_pooling2d_3 (Glo (None, 1280)	0	out_relu[0][0]
dense_6 (Dense) (None, 128)	163968	global_average_pc
dense_7 (Dense) (None, 2)	258	dense_6[0][0]
<hr/>		

INPUT DATASET

```
directory_dataset = "/content/drive/MyDrive/FaceRecog/new_dataset/dataset10"
category_dataset = ["edi", "unknown"]

data = []
labels = []

for category in category_dataset:
    path = os.path.join(directory_dataset, category)
    for file in os.listdir(path):
        if file.endswith(".jpg"):
            img = Image.open(os.path.join(path, file))
            data.append(np.array(img))
            if category == "edi":
                labels.append(0)
            else:
                labels.append(1)
```

```

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 = 4

(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/result10",
                            monitor="val_loss",
                            mode="min")

```

```

        monitor='val_loss',
        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 542ms/step - loss: 0.7268 - accuracy: 0.32

Epoch 00001: val_loss improved from inf to 0.66520, saving model to /content/drive/M
Epoch 2/10
4/4 [=====] - 1s 252ms/step - loss: 0.6001 - accuracy: 0.88

Epoch 00002: val_loss improved from 0.66520 to 0.62797, saving model to /content/dri
Epoch 3/10
4/4 [=====] - 1s 246ms/step - loss: 0.5067 - accuracy: 0.93

Epoch 00003: val_loss improved from 0.62797 to 0.59246, saving model to /content/dri
Epoch 4/10
4/4 [=====] - 1s 258ms/step - loss: 0.4823 - accuracy: 0.84

Epoch 00004: val_loss improved from 0.59246 to 0.56041, saving model to /content/dri
Epoch 5/10
4/4 [=====] - 1s 257ms/step - loss: 0.3793 - accuracy: 0.95

Epoch 00005: val_loss improved from 0.56041 to 0.53252, saving model to /content/dri
Epoch 6/10
4/4 [=====] - 1s 252ms/step - loss: 0.3896 - accuracy: 0.97

```

```

Epoch 00006: val_loss improved from 0.53252 to 0.51375, saving model to /content/dri
Epoch 7/10
4/4 [=====] - 1s 255ms/step - loss: 0.3448 - accuracy: 0.88

Epoch 00007: val_loss improved from 0.51375 to 0.49374, saving model to /content/dri
Epoch 8/10
4/4 [=====] - 1s 253ms/step - loss: 0.2932 - accuracy: 1.00

Epoch 00008: val_loss improved from 0.49374 to 0.47373, saving model to /content/dri
Epoch 9/10
4/4 [=====] - 1s 261ms/step - loss: 0.2619 - accuracy: 1.00

Epoch 00009: val_loss improved from 0.47373 to 0.45501, saving model to /content/dri
Epoch 10/10
4/4 [=====] - 1s 255ms/step - loss: 0.2410 - accuracy: 1.00

Epoch 00010: val_loss improved from 0.45501 to 0.44159, 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	0.67	1.00	0.80	2
unknown	1.00	0.50	0.67	2
accuracy			0.75	4
macro avg	0.83	0.75	0.73	4
weighted avg	0.83	0.75	0.73	4

▼ 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/result10/result10.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/result10/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)

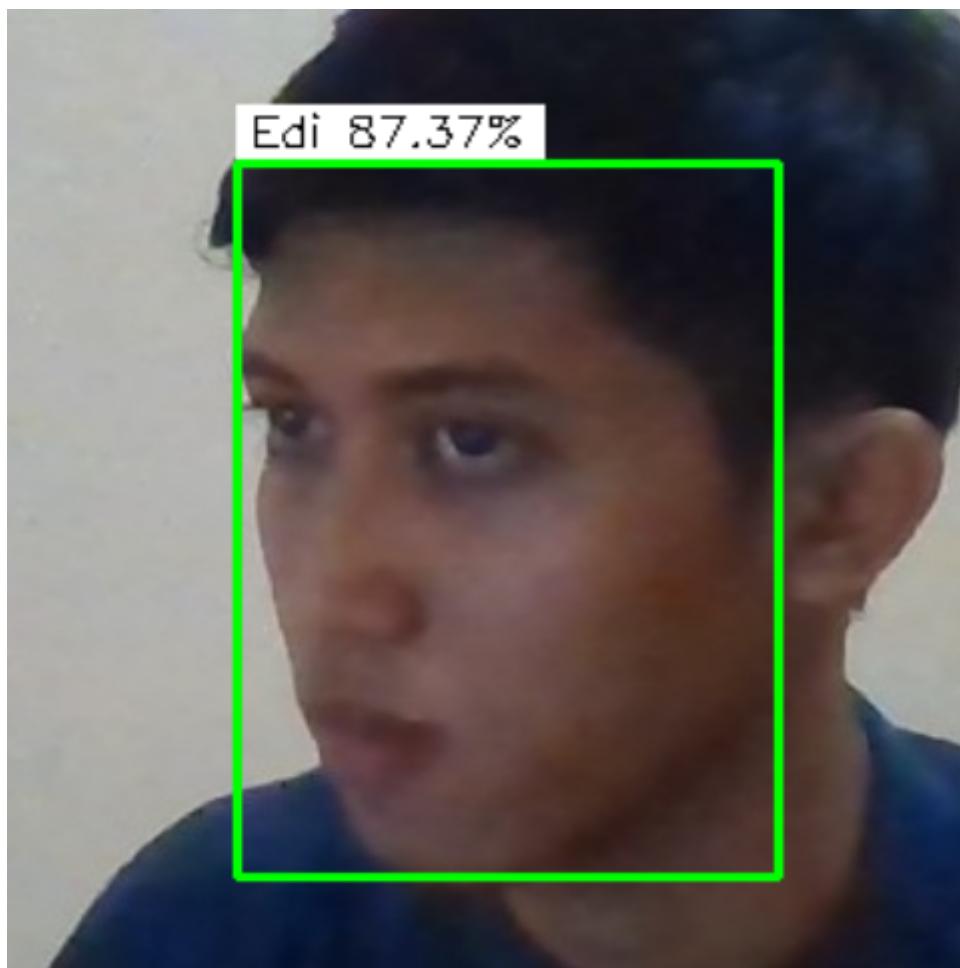
            # facial = pred[str(preds)]

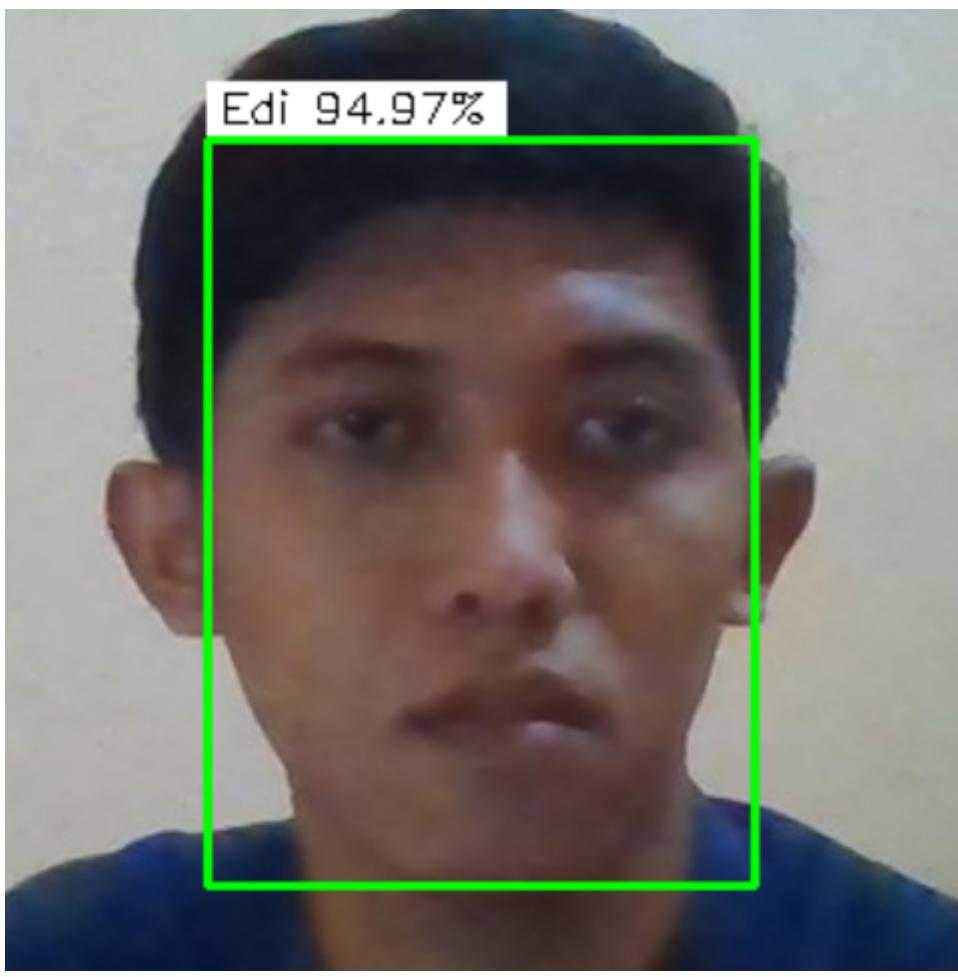
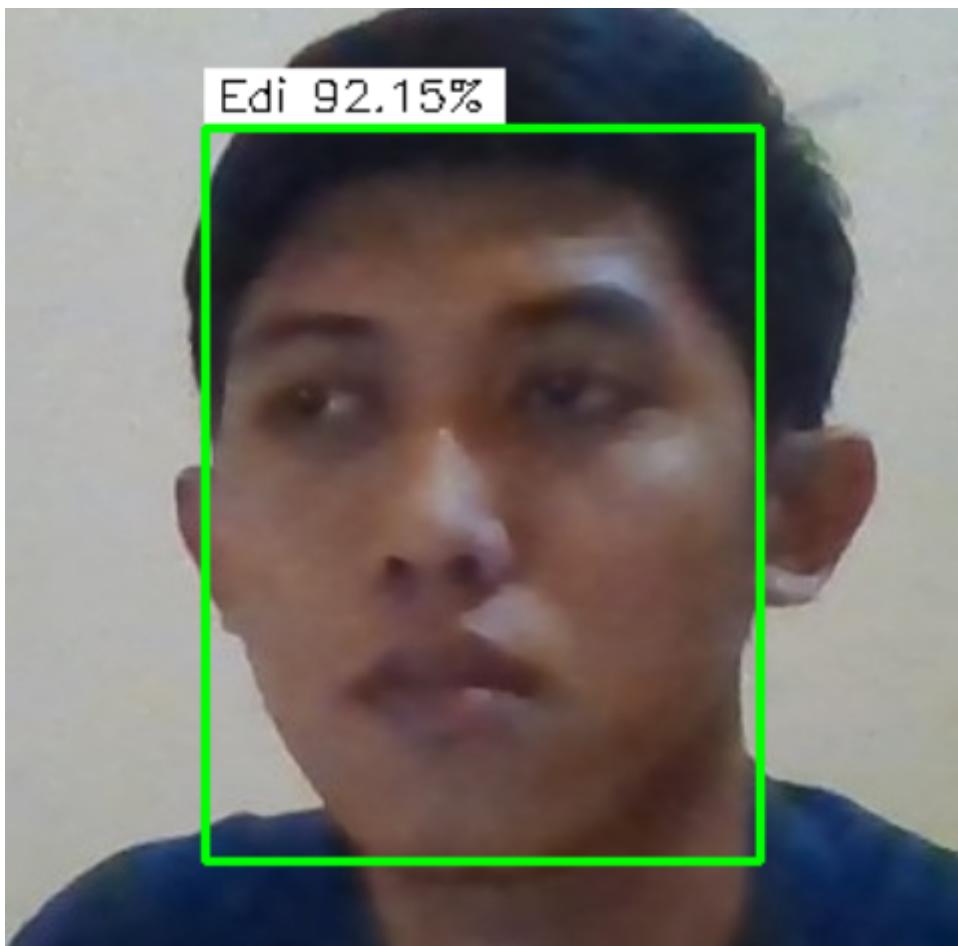
            if preds == 0:
                label = "Edi"
                color = (0, 255, 0)
            if preds == 1:
                label = "Unknown"
                color = (0, 0, 255)

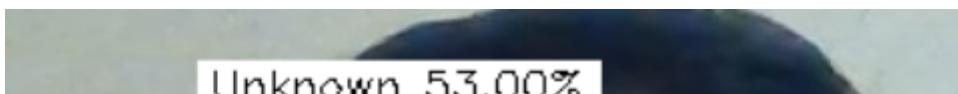
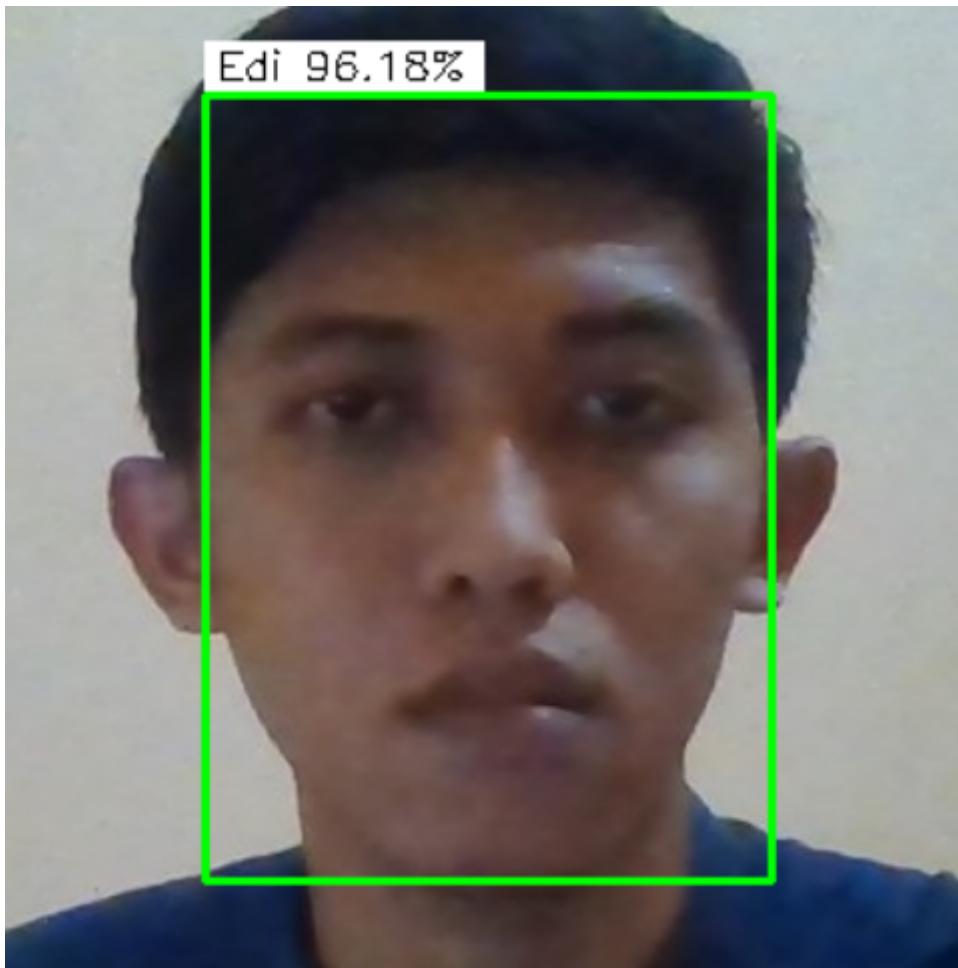
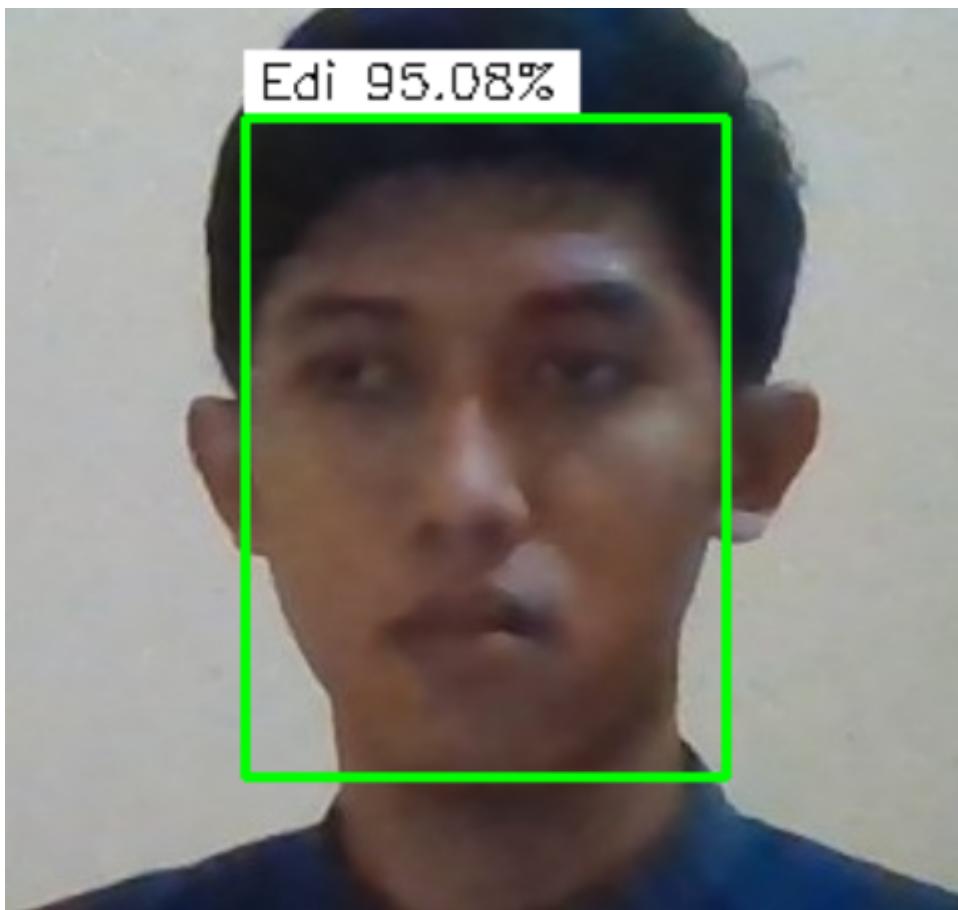
            label_full = label+" "+label_percent
            label_size = cv2.getTextSize(label_full, cv2.FONT_HERSHEY_SIMPLEX, 0.5, 1)
            cv2.rectangle(input_image, (startX, startY), (endX, endY), color, 2)
            cv2.rectangle(input_image, (startX, startY - 20), ((startX + label_size[0][0]) + 10,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

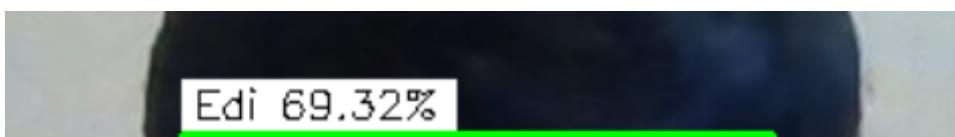
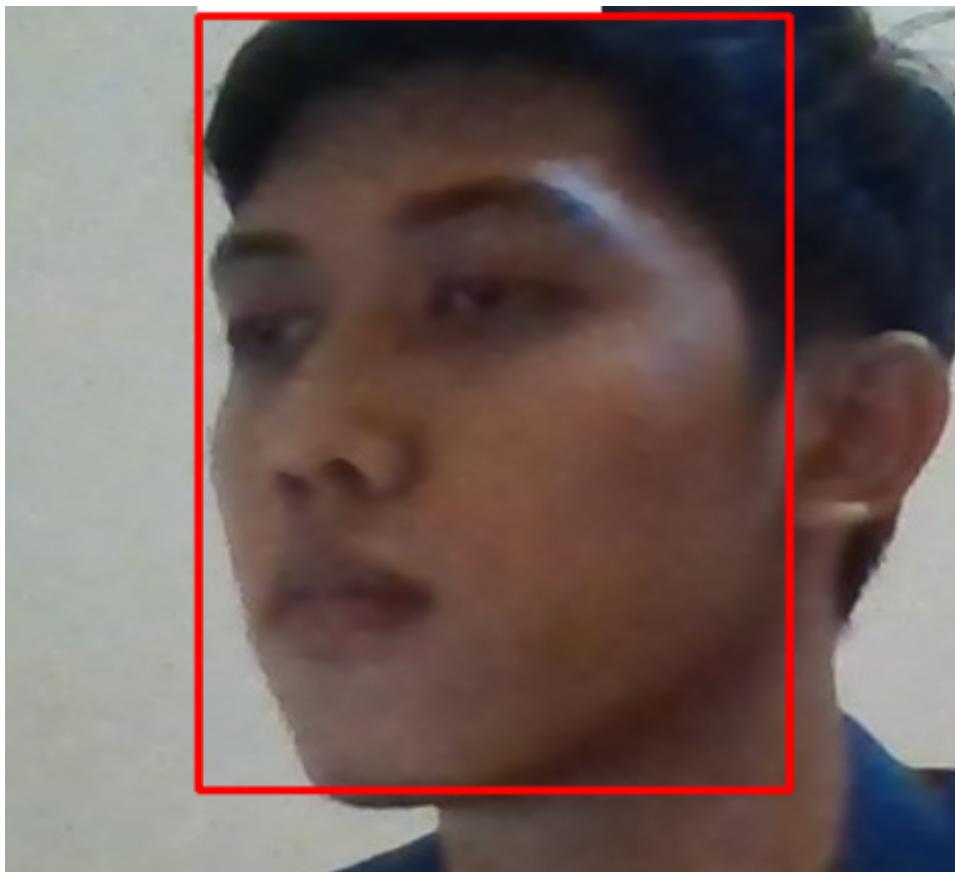
            image_resize = cv2.resize(input_image, (480, 480))
```

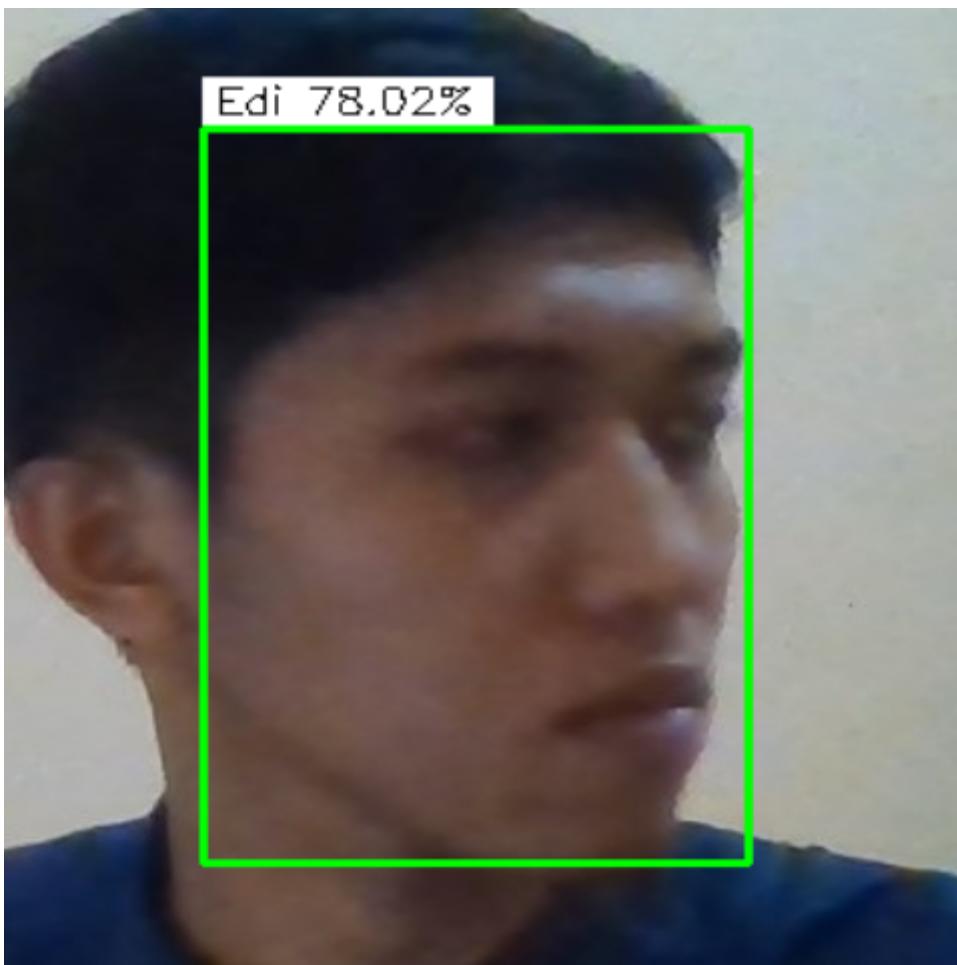
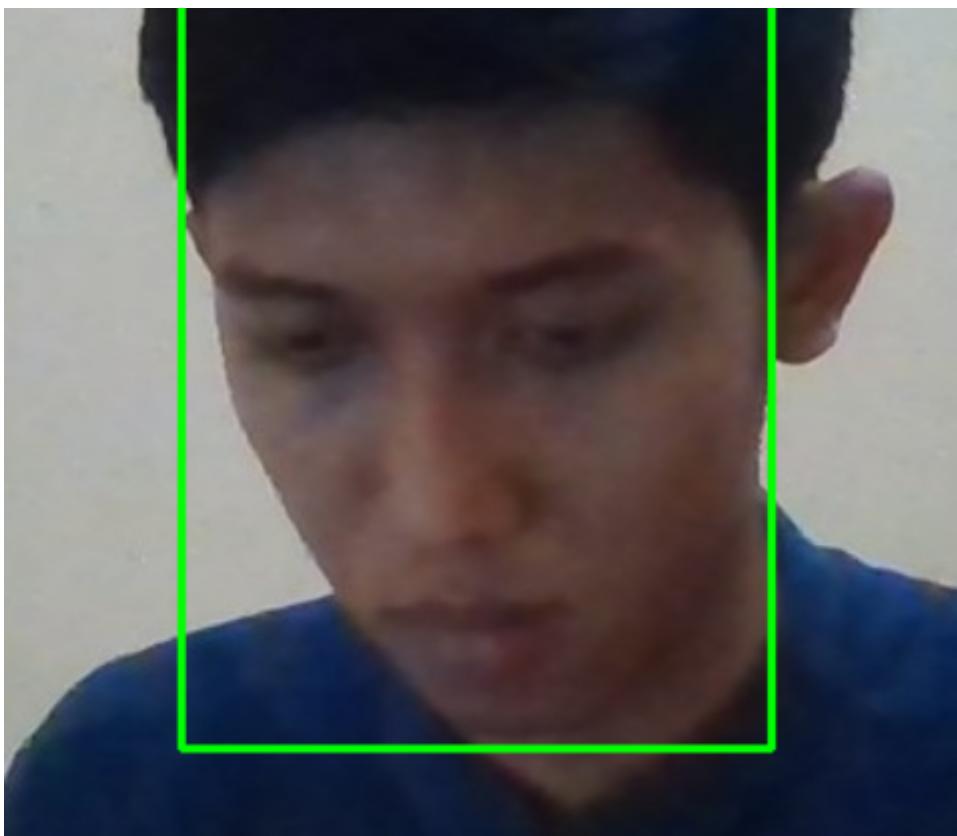
```
# print(preds)
cv2.imshow(image_resize)
cv2.imwrite("/content/drive/MyDrive/FaceRecog/new_dataset/result/result10/test"+str(court
time.sleep(1)
print('\n')
```

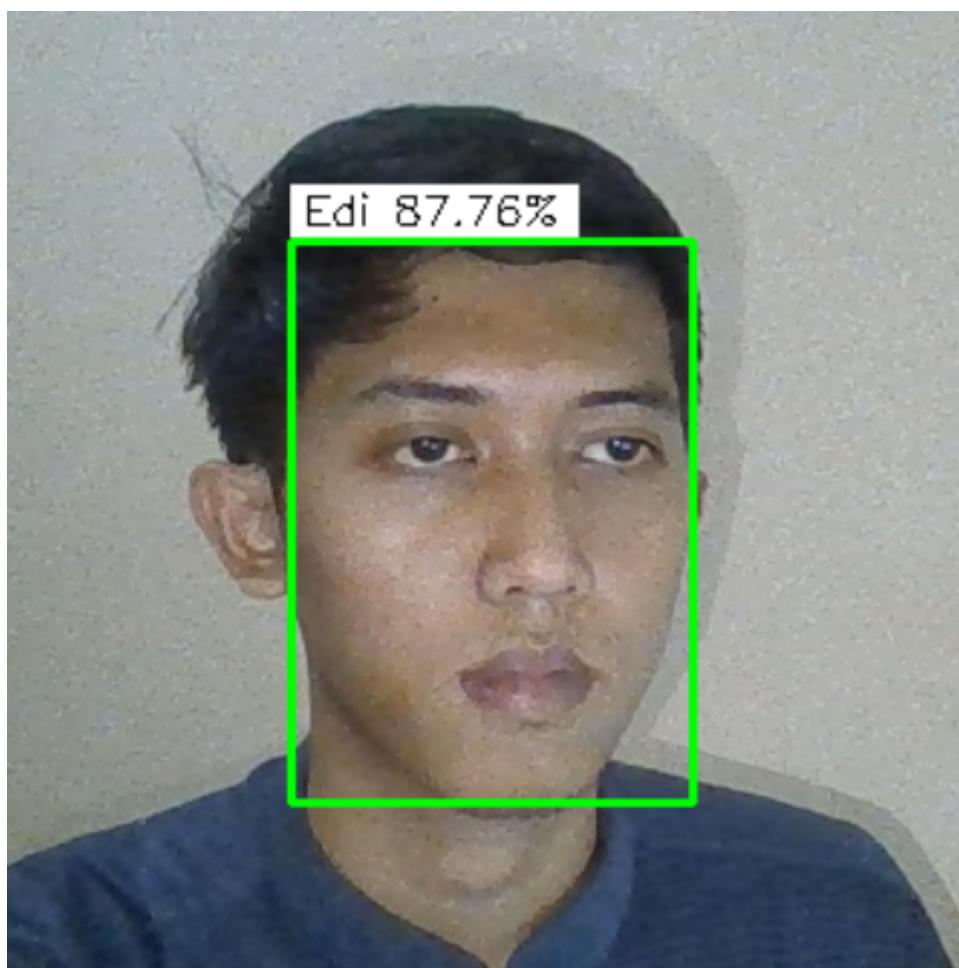
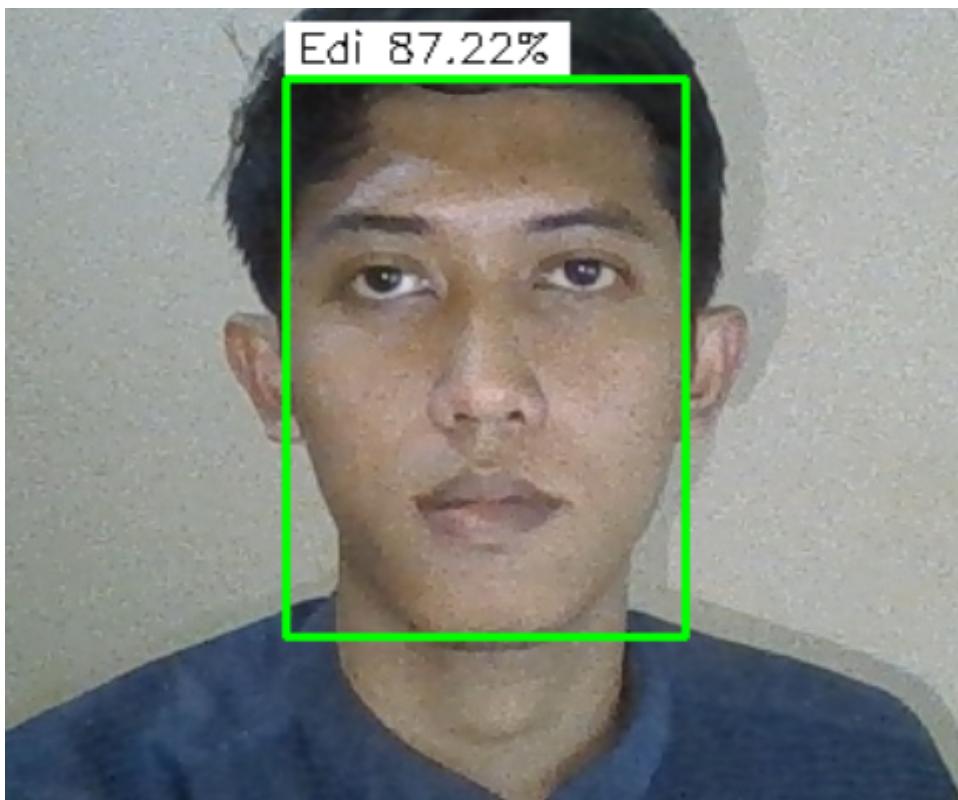


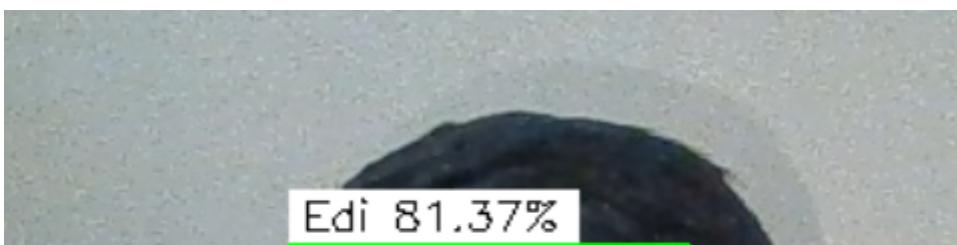
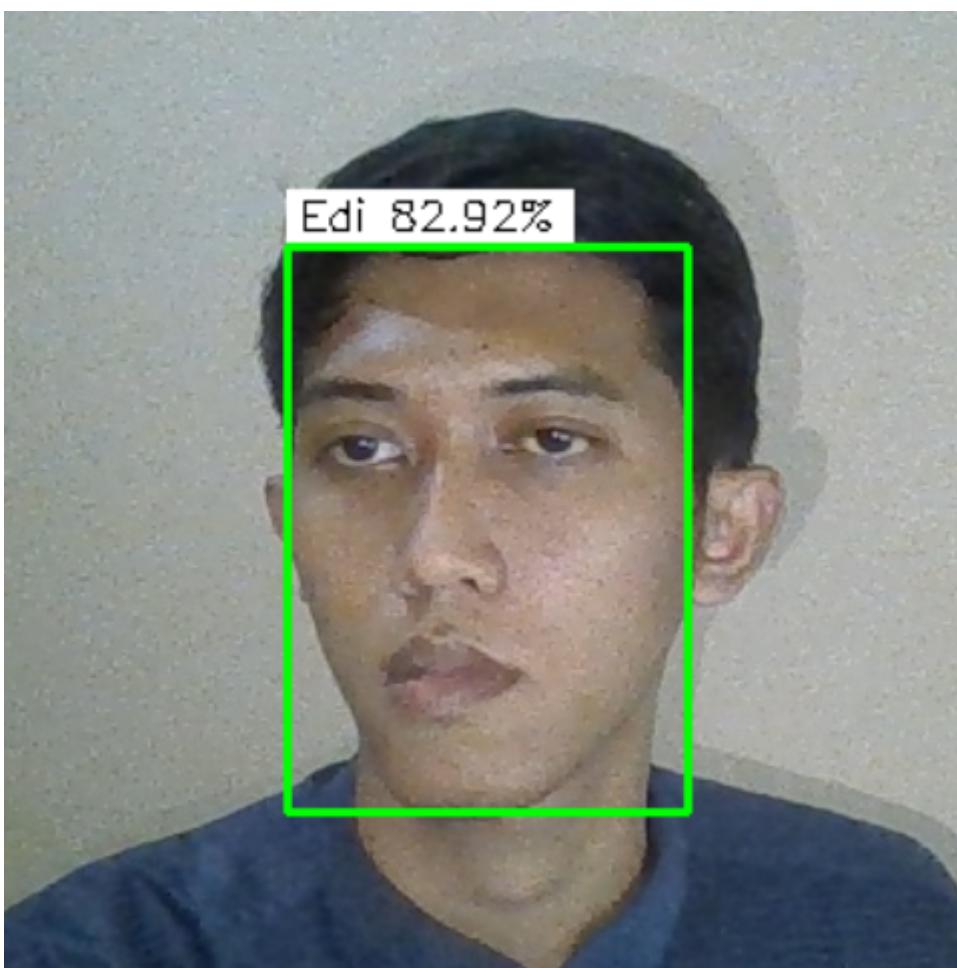
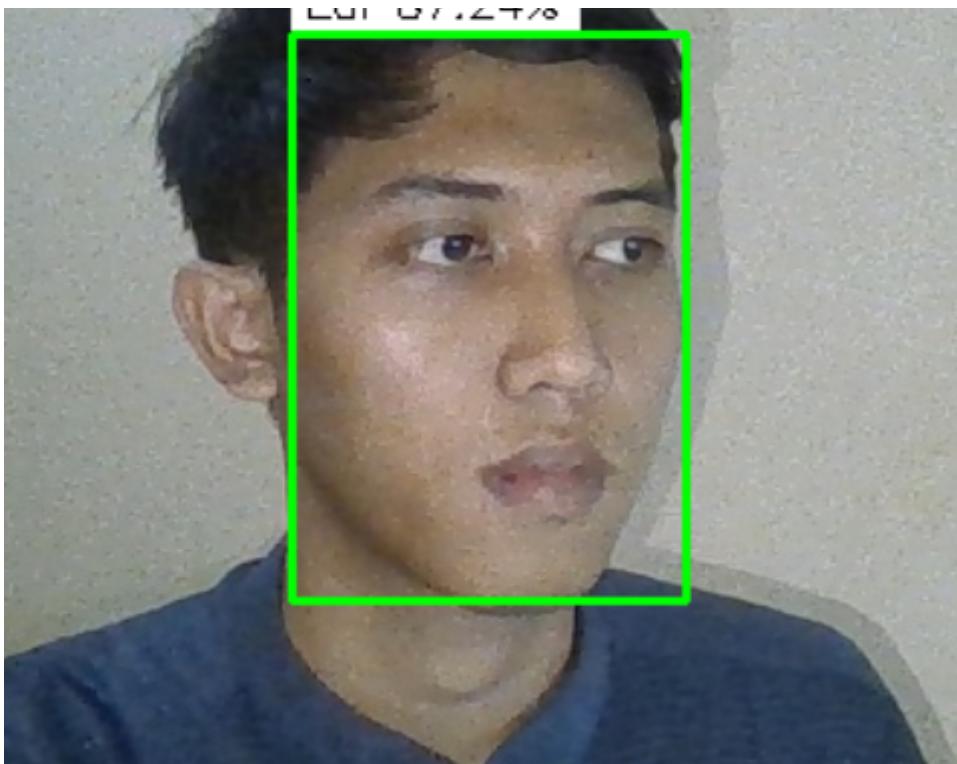


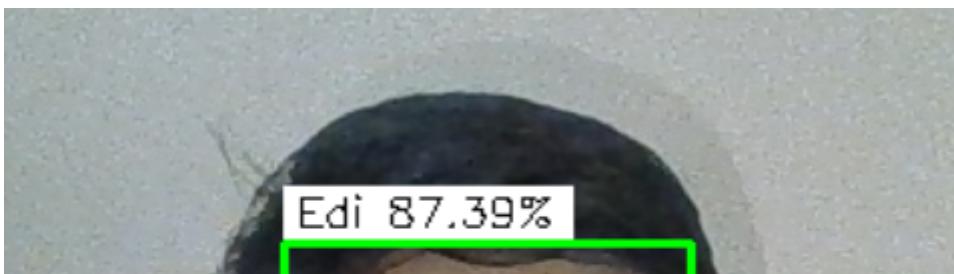
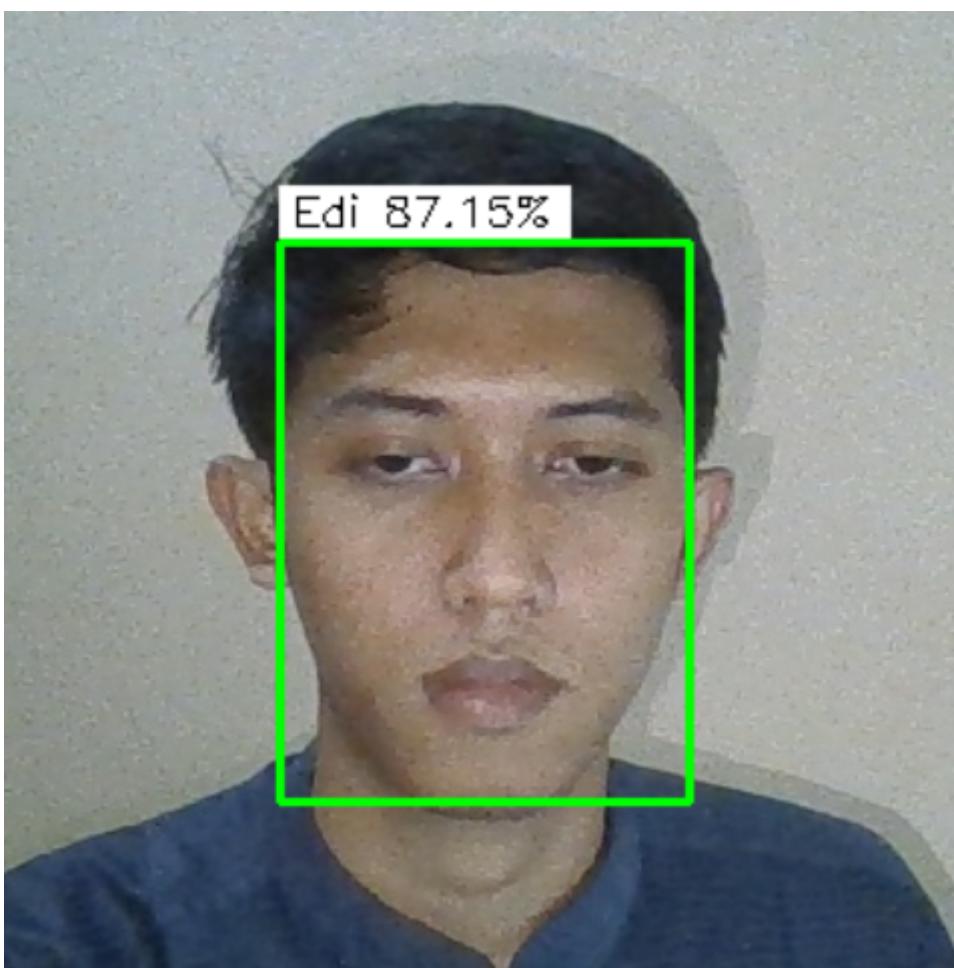
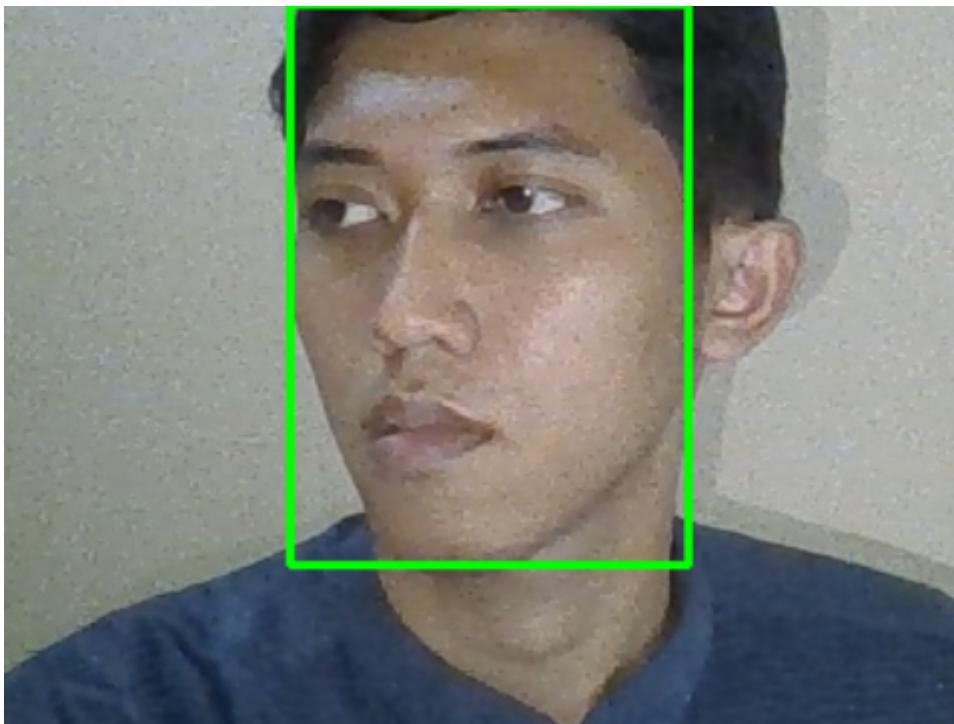


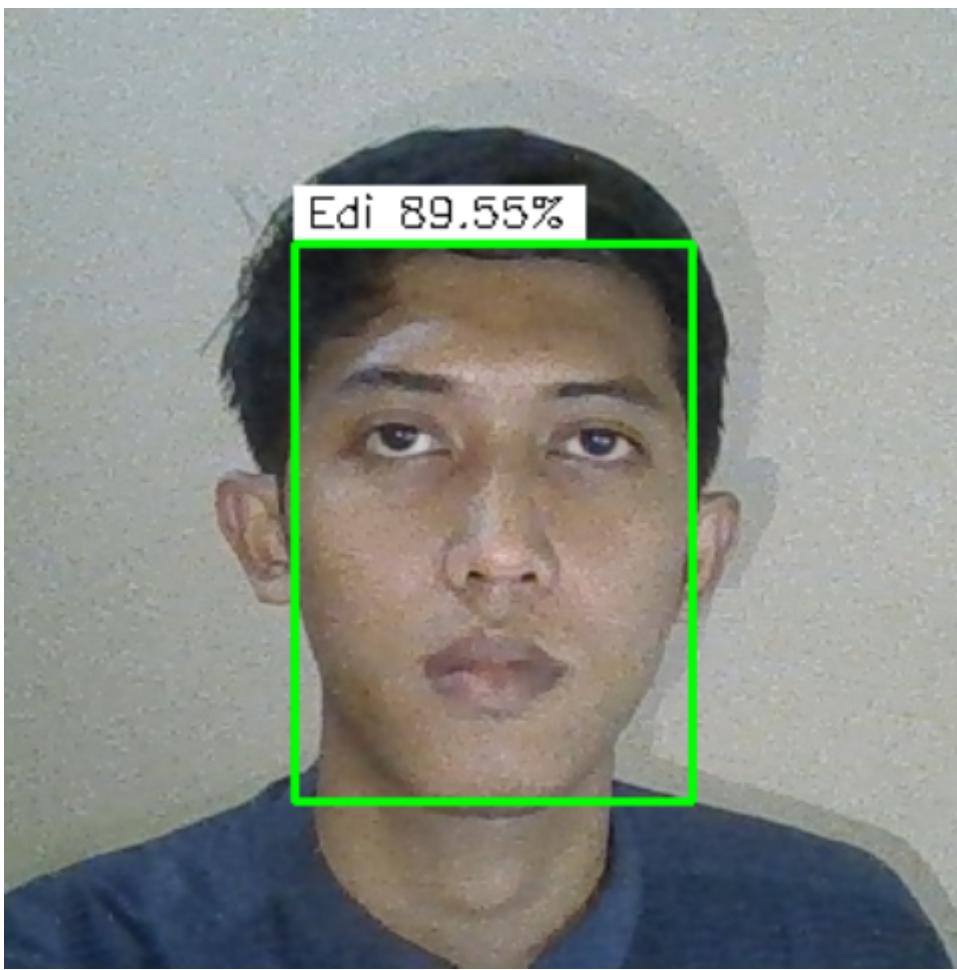
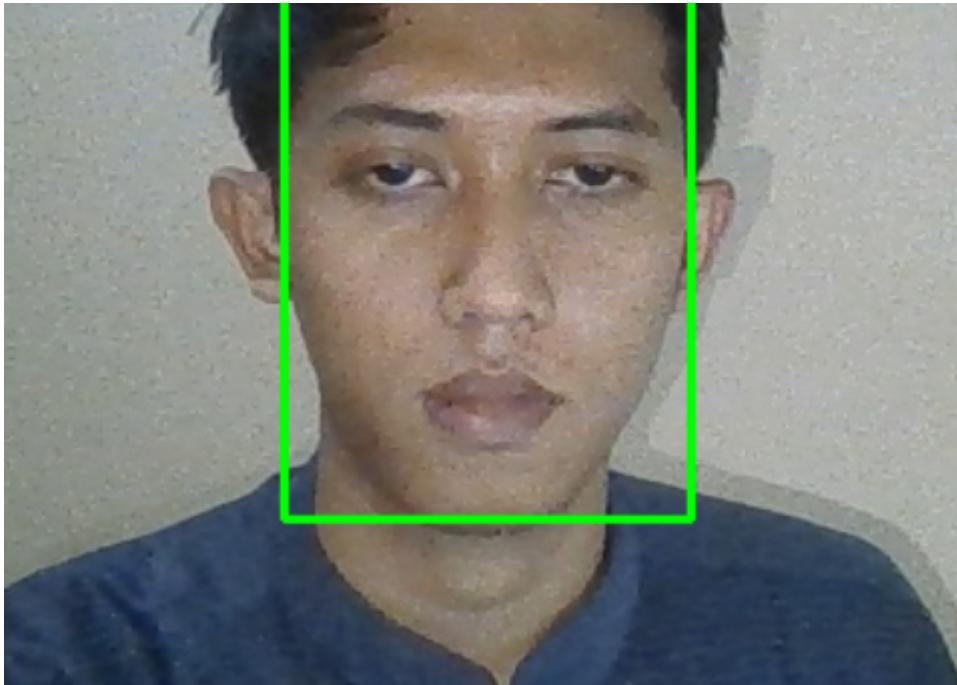












```
from tensorflow.keras.applications.mobilenet_v2 import preprocess_input  
from tensorflow.keras.preprocessing.image import img_to_array
```

https://colab.research.google.com/drive/1aPJDxWeV_M6S2l2GnrGZ7pH7UNEDS-cb#scrollTo=3x3DbRHqGFAu&printMode=true

```
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/result10/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_SIMPLEX, 0.5, color, 1)

image_resize = cv2.resize(input_image, (480, 480))
# print(preds)
cv2.imshow(image_resize)
cv2.imwrite("/content/drive/MyDrive/FaceRecog/new_dataset/result/result10/test"+str(court
time.sleep(1)
print('\n')
```

