마스크 판별 신호등

-코로나 시대의 딥러닝 활용-

송은이

01 서론

주제 선정 이유, 프로그램 활용 방안

02 코드 분석

사용 데이터와 모델 분석 ; CNN + 비디오 얼굴 인식

03 시뮬레이션

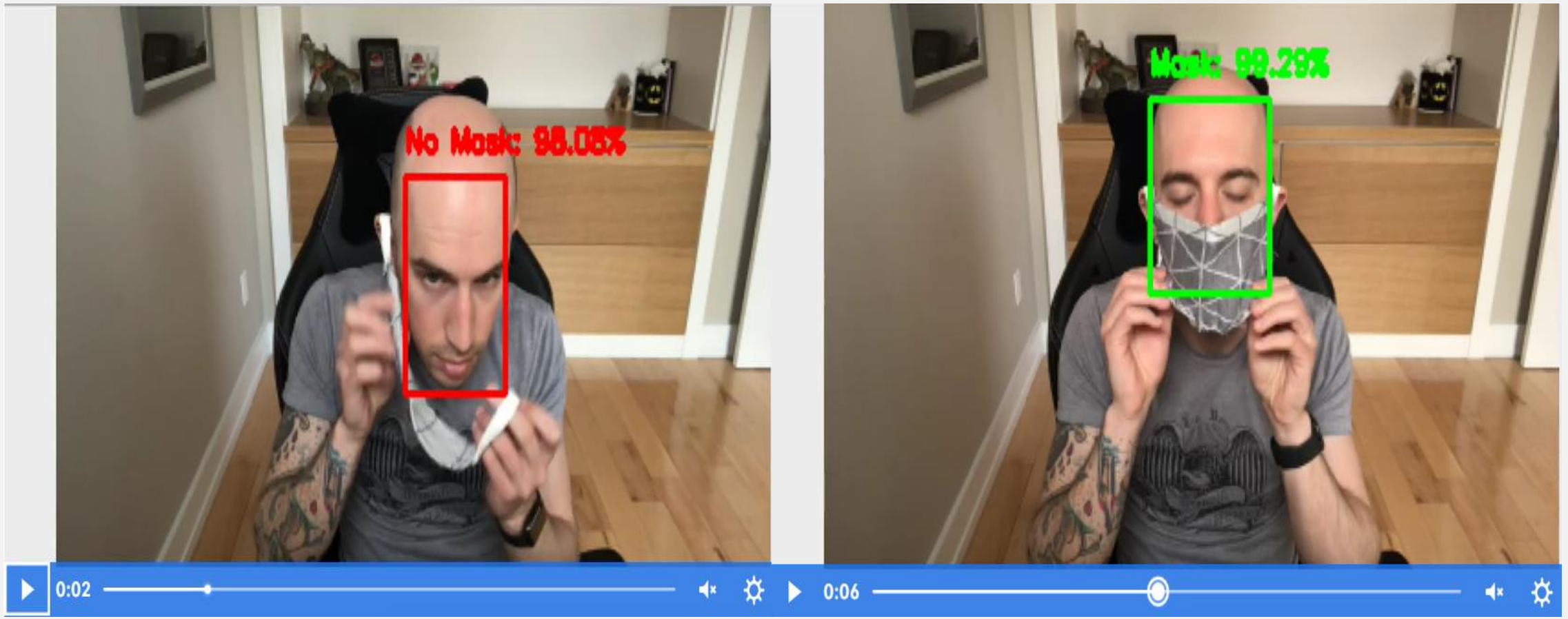
실제 동영상으로 모델 시뮬레이션

01 서론

주제 선정 이유

https://terms.naver.com/entry.nhn?docId=6155237&cid=51617&categoryId=67174

모델의 활용 예시와 효과



https://www.pyimagesearch.com/2020/05/04/covid-19-face-mask-detector-with-opencv-keras-tensorflow-and-deep-learning/

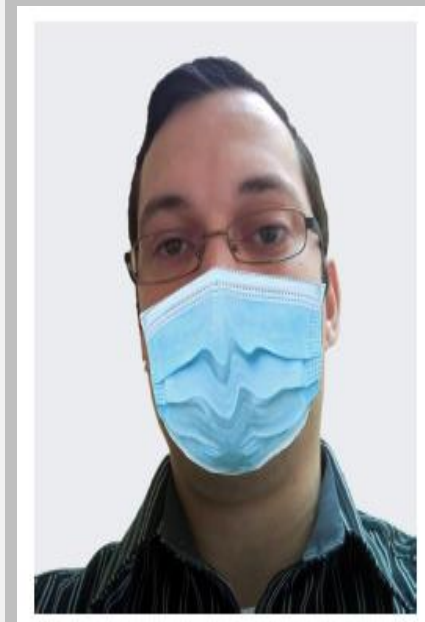
모델의 활용 예시와 효과

- 유치원, 초등학교 대상 '올바른 마스크 사용법' 수업 자료로 활용

- 공항, 카페 등 인구 밀집 공간에서 마스크 미 착용자 색출 및 주의 조치에 이용

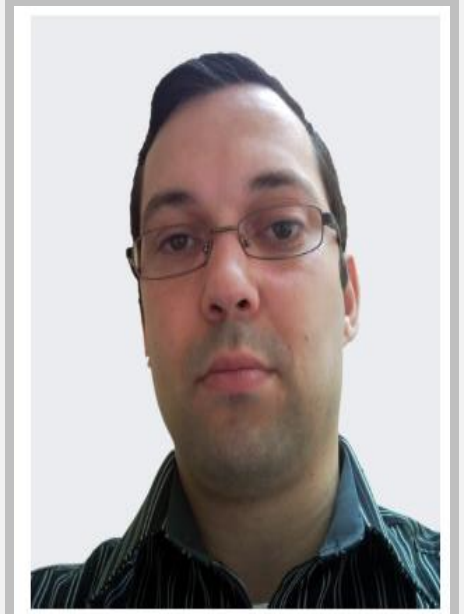
활용 데이터

데이터 출처 : https://arxiv.org/abs/2008.08016



#
1916

#
1893



#
1897

02 코드 분석

얼굴 사진에서 마스크 착용 여부 3 분류하는 코드

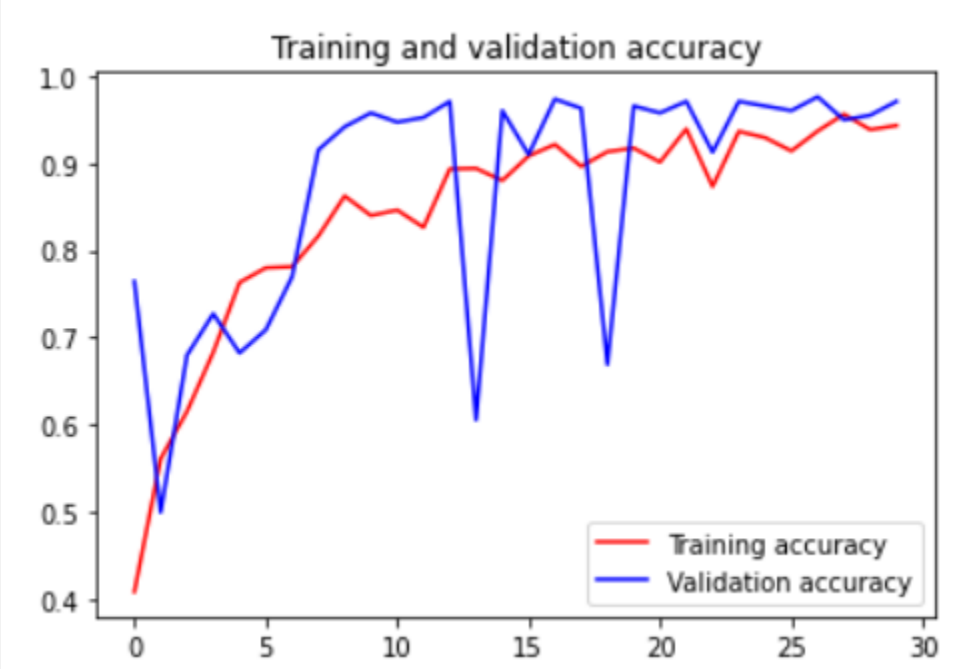
기존 사진을 여러 방법으로 변형하여 데이터 다양화

```
training_datagen = ImageDataGenerator(
      rescale = 1./255,
      rotation_range=40,
      width_shift_range=0.2,
      height_shift_range=0.2,
      shear_range=0.2,
      zoom_range=0.2,
      horizontal_flip=True,
      fill_mode='nearest')

validation_datagen = ImageDataGenerator(rescale = 1./255)

train_generator = training_datagen.flow_from_directory(
  train_dir,
  target_size=(150,150),
  class_mode='categorical',
  batch_size=126
)

validation_generator = validation_datagen.flow_from_directory(
  val_dir,
  target_size=(150,150),
  class_mode='categorical',
  batch_size=126
)
```

마스크 3분류 모델 핵심 파트

```
model = tf.keras.models.Sequential([
    # Note the input shape is the desired size of the image 150x150 with 3 bytes color
    # This is the first convolution
    tf.keras.layers.Conv2D(64, (3,3), activation='relu', input_shape=(150, 150, 3)),
    tf.keras.layers.MaxPooling2D(2, 2),
    # The second convolution
    tf.keras.layers.Conv2D(64, (3,3), activation='relu'),
    tf.keras.layers.MaxPooling2D(2,2),
    # The third convolution
    tf.keras.layers.Conv2D(128, (3,3), activation='relu'),
    tf.keras.layers.MaxPooling2D(2,2),
    # The fourth convolution
    tf.keras.layers.Conv2D(128, (3,3), activation='relu'),
    tf.keras.layers.MaxPooling2D(2,2),
    # Flatten the results to feed into a DNN
    tf.keras.layers.Flatten(),
    tf.keras.layers.Dropout(0.2),
    # 512 neuron hidden layer
    tf.keras.layers.Dense(512, activation='relu'),
    tf.keras.layers.Dense(3, activation='softmax')
])

model.summary()

model.compile(loss = 'categorical_crossentropy', optimizer='rmsprop', metrics=['accuracy'])

history = model.fit(train_generator, epochs=30, steps_per_epoch=20,
                    validation_data = validation_generator, verbose = 1, validation_steps=3)
```

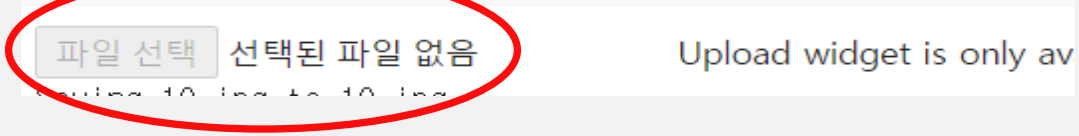
CNN 분석 결과



Training Loss : 0.1283 Accuracy : 95%
Validation Loss : 0.0722 Accuracy: 97%

CNN 분석 결과

```
import numpy as np
from google.colab import files
from keras.preprocessing import image

uploaded = files.upload()

for fn in uploaded.keys():

  # predicting images
  path = fn
  img = image.load_img(path, target_size=(150, 150))
  x = image.img_to_array(img)
  x = np.expand_dims(x, axis=0)
  # 순서 : [green,red,yellow]

  images = np.vstack([x])
  classes = model.predict(images, batch_size=10)
  classes_1 = classes.tolist()
  classes_2 = classes_1[0]
  light_color = light(classes_2)
  print(fn," is ",light_color)
```



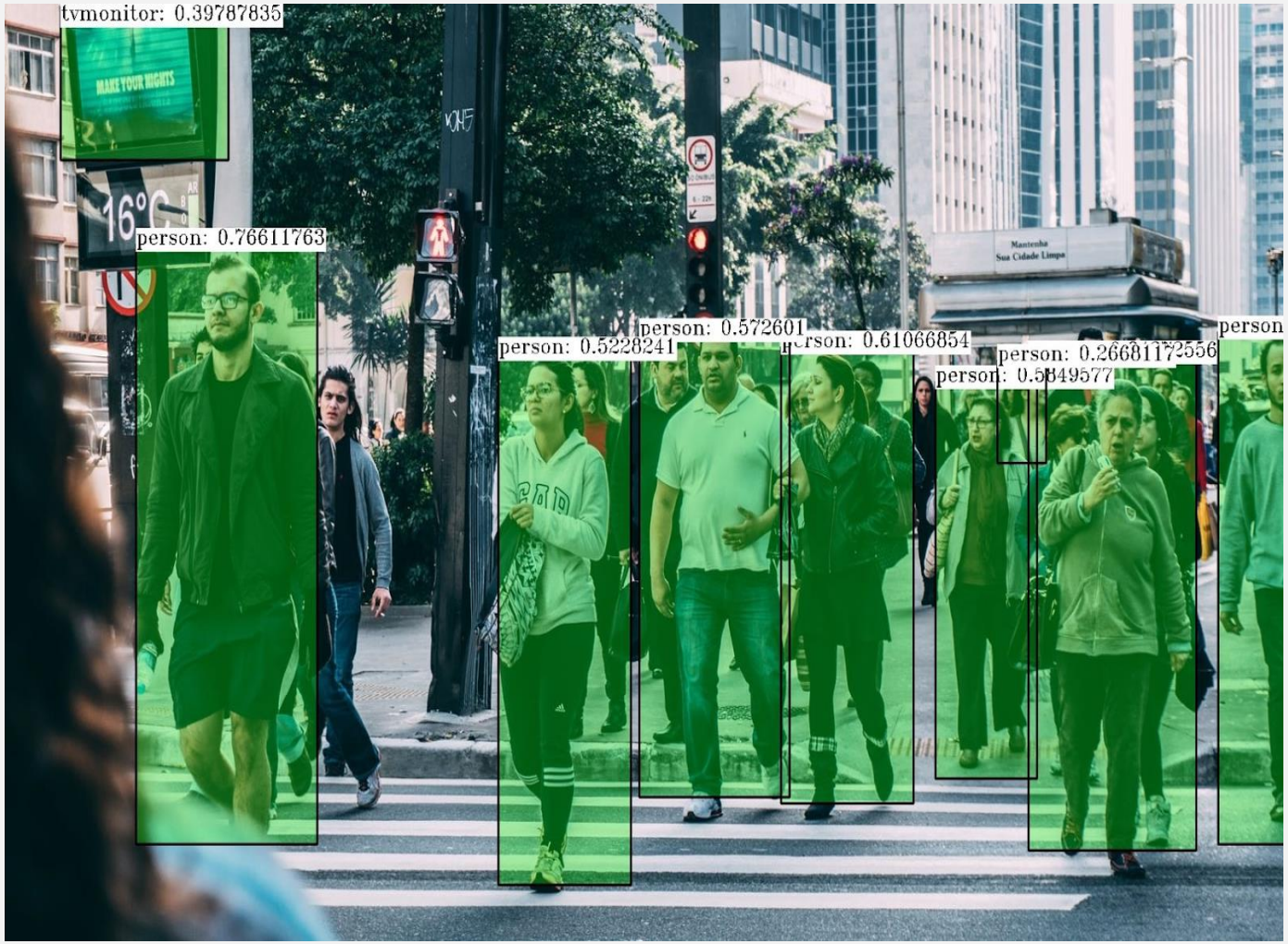
- 학습되지 않은 이미지 파일을 업로드해서 모델에 적용한 결과,

  - 초록과 노랑에 해당하는 경우 5% 정도의 오류가 발생

  - 빨강에 해당하는 경우는 15% 정도의 오류가 발생

- 본인이 생각하는 원인은 빨강 데이터(마스크 미착용)는 다른 이미지와 달리 해상도 크기가 비교적 많이 작았기 때문으로 생각됨

비디오 얼굴 인식 - openCV



[OpenCV 사용]

- 오픈 소스 컴퓨터 비전 라이브러리 중 하나로 크로스플랫폼과 실시간 이미지 프로세싱에 중점을 둔 라이브러리
- 영상처리 관련 가장 유명한 라이브러리

사진 얼굴 인식

```
uploaded = files.upload()

img = cv2.imread(list(uploaded.keys())[0])

# load face cascade and eye cascade
face_cascade = cv2.CascadeClassifier(utils.get_haarcascade_path('haarcascade_frontalface_default.xml'))

gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)

faces = face_cascade.detectMultiScale(gray, 1.3, 3)

rr=0
gg=0
yy=0
rr_faces=[]
gg_faces=[]
yy_faces=[]
```

```
for i, (x, y, w, h) in enumerate(faces):

    cropped = img[y - int(h/4):y + h + int(h/4), x - int(w/4):x + w + int(w/4)]
    face_img= cv2.cvtColor(cropped, cv2.COLOR_BGR2RGB)
    face_img=cv2.resize(face_img,(150,150))

    # plt.imshow(face_img)
    # plt.show()

    x = image.img_to_array(face_img)
    x = np.expand_dims(x, axis=0)
    images = np.vstack([x])
    classes = model.predict(images, batch_size=10)
    classes_1 = classes.tolist()
    classes_2 = classes_1[0]
    if classes_2.index(max(classes_2))==0:
      color="green"
    elif classes_2.index(max(classes_2))==1:
      color="red"
    elif classes_2.index(max(classes_2))==2:
      color="yellow"

    print(color)

    if color=="red":
      rr+=1
      rr_faces.append(faces[i])
    elif color=="green":
      gg+=1
      gg_faces.append(faces[i])
    elif color=="yellow":
      yy+=1
      yy_faces.append(faces[i])

print("There are total ",len(faces), "people!")
print("There are ",rr, "red people!")
print("There are ",yy, "yellow people!")
print("There are ",gg, "green people!")
```

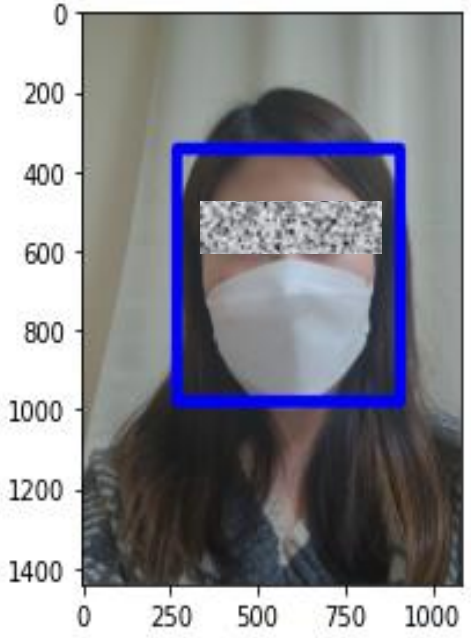
```
for (x, y, w, h) in gg_faces:
    cv2.rectangle(img, (x, y), (x+w, y+h), (255,0,0), 10)

for (x, y, w, h) in yy_faces:
    cv2.rectangle(img, (x, y), (x+w, y+h), (0,155,255), 10)

for (x, y, w, h) in rr_faces:
    cv2.rectangle(img, (x, y), (x+w, y+h), (0, 0, 255), 10)


img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
plt.figure()
plt.imshow(img)
plt.show()

# (255,0,0) 파랑
# (0,255,0) 초록
# (0,0,255) 빨강
# (0,155,255) 노랑
```

실제 사진을 넣고 결과 확인

파일 선택 mask.jpg

- **mask.jpg**(image/jpeg) - 520752 bytes, last

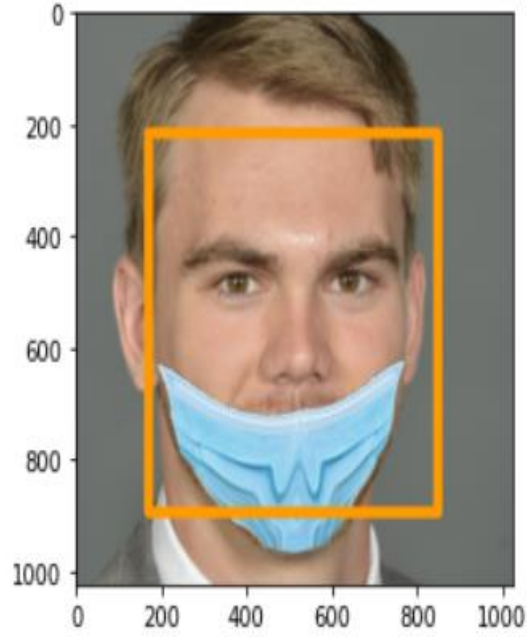
```
Saving mask.jpg to mask (2).jpg
green
There are total  1 people!
There are  0 red people!
There are  0 yellow people!
There are  1 green people!
```



파일 선택 00825_Mask...outh_Chin.jpg

- **00825_Mask_Mouth_Chin.jpg**(image/jpeg) -

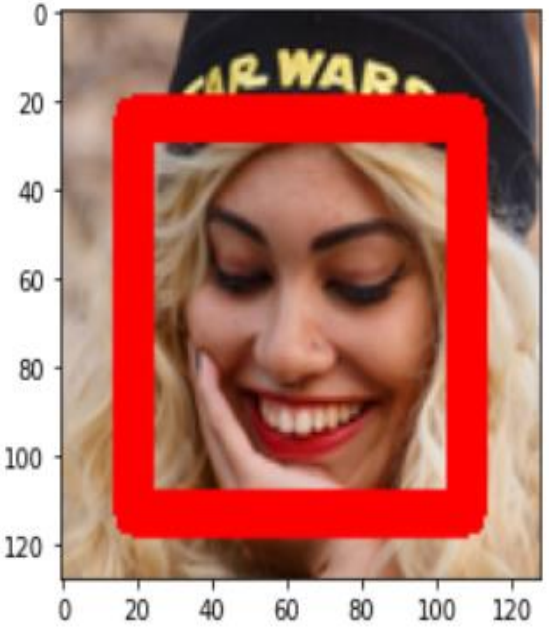
```
Saving 00825_Mask_Mouth_Chin.jpg to 00825_M
yellow
There are total  1 people!
There are  0 red people!
There are  1 yellow people!
There are  0 green people!
```



파일 선택 10090.png

- **10090.png**(image/png) - 33567 bytes, last r

```
Saving 10090.png to 10090 (1).png
red
There are total  1 people!
There are  1 red people!
There are  0 yellow people!
There are  0 green people!
```



감사합니다

https://github.com/EUN2I