1. 깃허브에서 repository를 클론합니다.

!git clone https://github.com/gachonyws/face-mask-detector.git

2.디렉토리 이동 후 작업 시작

cd face-mask-detector/

### **1. 모델 로드**

from tensorflow.keras.applications.mobilenet\_v2 import preprocess\_input

from tensorflow.keras.models import load\_model

import numpy as np

import cv2

import os

import matplotlib.pyplot as plt

# 사진 속에서 얼굴을 탐지하는 face\_detector 모델

facenet = cv2.dnn.readNet('face\_detector/deploy.prototxt','face\_detector/res10\_300x300\_ssd\_iter\_140000.caffemodel')

# 얼굴인식 후 마스크 착용 여부를 확인하는 모델

model = load\_model('mask\_detector.model')

-> 클론한 github에서 face\_detector 모델을 로드해서 데이터를 적용한다.

### **2. 이미지 로드**

img = cv2.imread('/content/face-mask-detector/examples/example\_01.png')

h,w = img.shape[:2]

plt.figure(figsize=(16,10))

plt.imshow(img[:,:,::-1]) # BGR -> RGB 변환

->마스크 착용 여부를 확인할 이미지를 로드한다. 로드된 이미지의 크기 16, 10으로 조정한 뒤 그 이미지를 출력한다.

### **3. Preprocessing for Face Detection**

blob = cv2.dnn.blobFromImage(img, scalefactor=1.0, size=(300, 300), mean=

(104.0, 177.0, 123.0))

facenet.setInput(blob) # 모델에 들어가는 input

detections = facenet.forward() # 결과를 inference

->카메라로 얻은 이미지 배열을 일괄 추론해서 이미지 전처리 과정을 통해 detections라는 배열에 저장한다.

### **4. Detect Faces**

faces = [ ]

# 사진속 얼굴 개수가 여러 개 있을 수 있으니 반복문 사용

for i in range(detections.shape[2]):

confidence = detections[0,0,i,2]

if confidence < 0.5:

continue

else:

x1 = int(detections[0,0,i,3] \* w)

y1 = int(detections[0,0,i,4] \* h)

x2 = int(detections[0,0,i,5] \* w)

y2 = int(detections[0,0,i,6] \* h)

face = img[y1:y2, x1:x2]

faces.append(face)

->전처리 과정을 통해 저장한 detections를 confidence에 넣어, 0.5보다 크면 좌표들을 가져와 img 배열에 넣고 얼굴로 인식하여 출력한다.

**5. Detect Masks from faces**

plt.figure(figsize=(10,5))

for i, face in enumerate(faces):

plt.subplot(1, len(faces), i+1)

plt.imshow(face[:, :, ::-1])

plt.figure(figsize=(10,5))

for i, face in enumerate(faces):

face\_input = cv2.cvtColor(face, cv2.COLOR\_BGR2RGB)

face\_input = cv2.resize(face\_input, dsize=(224, 224))

face\_input = preprocess\_input(face\_input)

face\_input = np.expand\_dims(face\_input, axis=0)

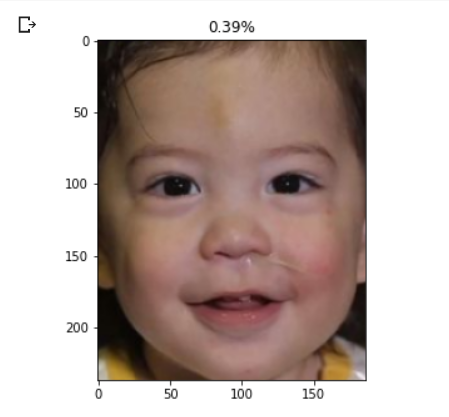
(mask, nomask) = model.predict(face\_input)[0]

plt.subplot(1, len(faces), i+1)

plt.imshow(face[:, :, ::-1])

plt.title('%.2f%%' % (mask \* 100))

-> 검출한 얼굴 수 만큼 반복문을 사용하여 (224,224)로 사이즈를 조정하고, 모델을 적용시킨 뒤 마스크를 착용하였는지에 대한 여부를 확인하고 퍼센트로 보여준다.



### **6. 이어서 동영상 재생**

from tensorflow.keras.applications.mobilenet\_v2 import preprocess\_input

from tensorflow.keras.models import load\_model

import numpy as np

import cv2

import matplotlib.pyplot as plt

import os

cap = cv2.VideoCapture('examples/03.mp4') -> 비디오 캡쳐 객체 생성

ret, img = cap.read()

fourcc = cv2.VideoWriter\_fourcc('m', 'p', '4', 'v')

-> cv2.VideoWriter 클래스를 이용하여 일련의 프레임을 동영상 파일로 저장할 수 있다.

일련의 프레임은 모두 크기와 데이터 타입이 같아야 한다. 또한 Fourcc (4-문자 코드, four character code)를 지정해줘야 한다.Fourcc는 동영상 파일의 코덱, 압축 방식, 색상, 픽셀 포맷 등을 정의하는 정수 값이다.

out = cv2.VideoWriter('./output.mp4', fourcc, cap.get(cv2.CAP\_PROP\_FPS), (img.shape[1], img.shape[0]))

-> cv2.VideoWriter(비디오 파일 이름, fourcc, 초당 프레임 수, frameSize) 객체를 생성하여, out에 영상을 저장한다.

while cap.isOpened():

ret, img = cap.read()

if not ret:

break

-> cap이 열린 동안 반복하고, 아니면 반복문을 종료한다.

h, w = img.shape[:2]

blob = cv2.dnn.blobFromImage(img, scalefactor=1., size=(300, 300), mean=(104., 177., 123.))

-> 이미지 전처리

facenet.setInput(blob)

dets = facenet.forward()

->blob 네트워크를 통해 예측을 가져온다

result\_img = img.copy()

for i in range(dets.shape[2]):

confidence = dets[0, 0, i, 2]

if confidence < 0.5:

continue

x1 = int(dets[0, 0, i, 3] \* w)

y1 = int(dets[0, 0, i, 4] \* h)

x2 = int(dets[0, 0, i, 5] \* w)

y2 = int(dets[0, 0, i, 6] \* h)

face = img[y1:y2, x1:x2]

face\_input = cv2.resize(face, dsize=(224, 224))

face\_input = cv2.cvtColor(face\_input, cv2.COLOR\_BGR2RGB)

face\_input = preprocess\_input(face\_input)

face\_input = np.expand\_dims(face\_input, axis=0)

mask, nomask = model.predict(face\_input).squeeze()

if mask > nomask:

color = (0, 255, 0)

label = 'Mask %d%%' % (mask \* 100)

-> 만약 mask를 착용했을 확률이 더 크다면, 색을 ‘초록’으로 color에 저장하고 Mask 착용의 퍼센트를 label에 저장한다.

else:

color = (0, 0, 255)

label = 'No Mask %d%%' % (nomask \* 100)

->mask를 착용했을 확률이 미착용했을 확률보다 작다면, 색을 ‘빨강’으로 color에 저장하고 mask 착용의 퍼센트를 label에 저장한다.

cv2.rectangle(result\_img, pt1=(x1, y1), pt2=(x2, y2), thickness=2, color=color, lineType=cv2.LINE\_AA)

-> 결과 이미지를 넣어, if문을 통해 얻은 color값을 가지고 두께가 2인 사각형을 출력한다.

cv2.putText(result\_img, text=label, org=(x1, y1 - 10), fontFace=cv2.FONT\_HERSHEY\_SIMPLEX, fontScale=0.8, color=color, thickness=2, lineType=cv2.LINE\_AA)

->결과 이미지를 넣고, if문을 통해 얻은 label값을 0.8 크기의 텍스트로 출력한다.

out.write(result\_img)

->저장된 결과를 출력하여 보여준다.

#cv2\_imshow(result\_img) # colab 환경에서 출력 때문에 계속 Busy 상태라 ignore

if cv2.waitKey(1) == ord('q'):

break

-> ‘q’를 입력하면, 실행을 종료한다.

out.release()

cap.release()

-> 할당 해제

from tensorflow.keras.applications.mobilenet\_v2 import preprocess\_input

from tensorflow.keras.models import load\_model

import numpy as np

import cv2

import winsound

# facenet : 얼굴을 찾는 모델

facenet = cv2.dnn.readNet('models/deploy.prototxt', 'models/res10\_300x300\_ssd\_iter\_140000.caffemodel')

# model : 마스크 검출 모델

model = load\_model('models/mask\_detector.model')

# 동영상 파일 읽기

# cap = cv2.VideoCapture('imgs/01.mp4')

# 실시간 웹캠 읽기

cap = cv2.VideoCapture(0)

i = 0

while cap.isOpened():

ret, img = cap.read()

if not ret:

break

# 이미지의 높이와 너비 추출

h, w = img.shape[:2]

# 이미지 전처리

# ref. https://www.pyimagesearch.com/2017/11/06/deep-learning-opencvs-blobfromimage-works/

blob = cv2.dnn.blobFromImage(img, scalefactor=1., size=(300, 300), mean=(104., 177., 123.))

# facenet의 input으로 blob을 설정

facenet.setInput(blob)

# facenet 결과 추론, 얼굴 추출 결과가 dets의 저장

dets = facenet.forward()

# 한 프레임 내의 여러 얼굴들을 받음

result\_img = img.copy()

# 마스크를 찾용했는지 확인

for i in range(dets.shape[2]):

# 검출한 결과가 신뢰도

confidence = dets[0, 0, i, 2]

# 신뢰도를 0.5로 임계치 지정

if confidence < 0.5:

continue

# 바운딩 박스를 구함

x1 = int(dets[0, 0, i, 3] \* w)

y1 = int(dets[0, 0, i, 4] \* h)

x2 = int(dets[0, 0, i, 5] \* w)

y2 = int(dets[0, 0, i, 6] \* h)

# 원본 이미지에서 얼굴영역 추출

face = img[y1:y2, x1:x2]

# 추출한 얼굴영역을 전처리

face\_input = cv2.resize(face, dsize=(224, 224))

face\_input = cv2.cvtColor(face\_input, cv2.COLOR\_BGR2RGB)

face\_input = preprocess\_input(face\_input)

face\_input = np.expand\_dims(face\_input, axis=0)

# 마스크 검출 모델로 결과값 return

mask, nomask = model.predict(face\_input).squeeze()

# 마스크를 꼈는지 안겼는지에 따라 라벨링해줌

if mask > nomask:

color = (0, 255, 0)

label = 'Mask %d%%' % (mask \* 100)

else:

color = (0, 0, 255)

label = 'No Mask %d%%' % (nomask \* 100)

frequency = 2500 # Set Frequency To 2500 Hertz

duration = 1000 # Set Duration To 1000 ms == 1 second

winsound.Beep(frequency, duration)

# 화면에 얼굴부분과 마스크 유무를 출력해해줌

cv2.rectangle(result\_img, pt1=(x1, y1), pt2=(x2, y2), thickness=2, color=color, lineType=cv2.LINE\_AA)

cv2.putText(result\_img, text=label, org=(x1, y1 - 10), fontFace=cv2.FONT\_HERSHEY\_SIMPLEX, fontScale=0.8,

color=color, thickness=2, lineType=cv2.LINE\_AA)

cv2.imshow('img',result\_img)

# q를 누르면 종료

if cv2.waitKey(1) & 0xFF == ord('q'):

break

|  |  |
| --- | --- |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |