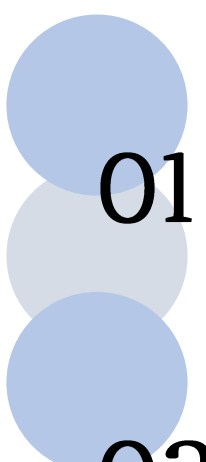


# Real-time Facial Analysis

Using webcam

윤희재 이수환 장예진

# 목차



01 주제

02 중간발표 review 및 변경 사항

03 Prior research

04 Data 소개 & preprocessing

05 Model (1),(2),(3)

06 결과 및 보완점

# 주제



웹 캠을 활용한  
실시간 얼굴 분석 진행



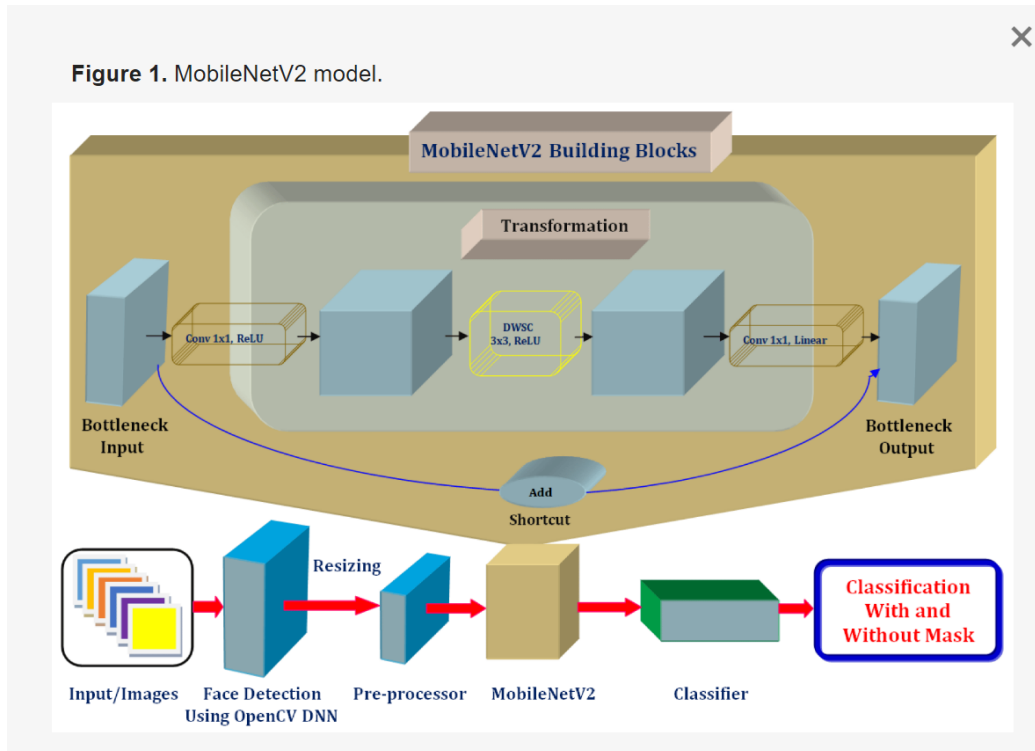
## 중간 발표 review 및 변경사항

웹 캠을 활용한  
실시간 얼굴 “감정인식”



웹 캠을 활용한  
실시간 얼굴 분석  
(성별, 나이, 감정인식)로 세분화 !

# Prior research(1)



## Face Mask Detection on Photo and Real-Time Video Images Using Caffe-MobileNetV2 Transfer Learning,2023

주제 : 사진 및 실시간 비디오 이미지에서 얼굴 마스크 감지

사용 모델 :

Caffe-MobileNetV2(CMNV2) :

MobileNetV2 모델에 대한 전이학습 + 미세 조정을 위한 단일 사진 multi-box 검출기로 Caffe 모델

MobileNetV2 구조 :

입력층 - 1x1 CNN 레이어, - 17개의 3x3 CNN 레이어, - 최대 풀링 평균레이어 - 분류 레이어

[깊이와 커널수, 필터수 조정하여 계산 시간을 단축]

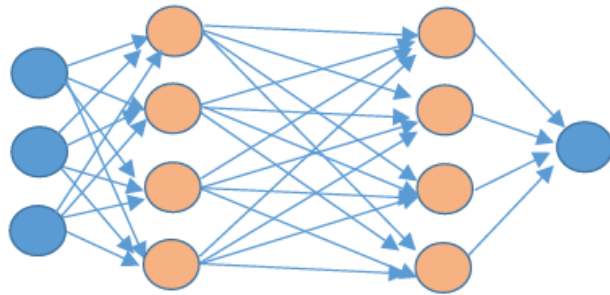
제안된 모델은 학습 데이터 세트의

패턴과 라벨을 기반으로 예측을 수행

예측 결과 99.64% 정확도

train data 1100개, test data 276개(마스크 착용 138개, 마스크 미착용 138개)

## Prior research(2)



Input layer Hidden layer1 Hidden layer 2 Output layer

FIGURE 4: An example of RNN architecture [29, 30].

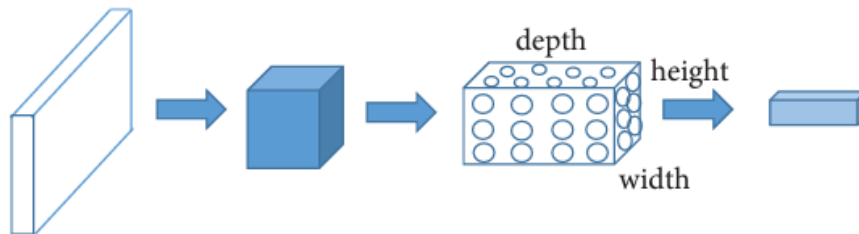


FIGURE 5: An example of CNN architecture [29, 30].

### Proposed Detection Face Model by Using Asian Data Set ,2021

Retina Face로 얼굴 감지 - MaskTheFace 프로그램을 사용하여 데이터셋을 생성 -MobileNetV2 모델로 분류

CNN을 활용하여 각 뉴런에 입력을 받음 - 모델 이미지의 특성을 encoding - 매개변수 수를 크게 줄임

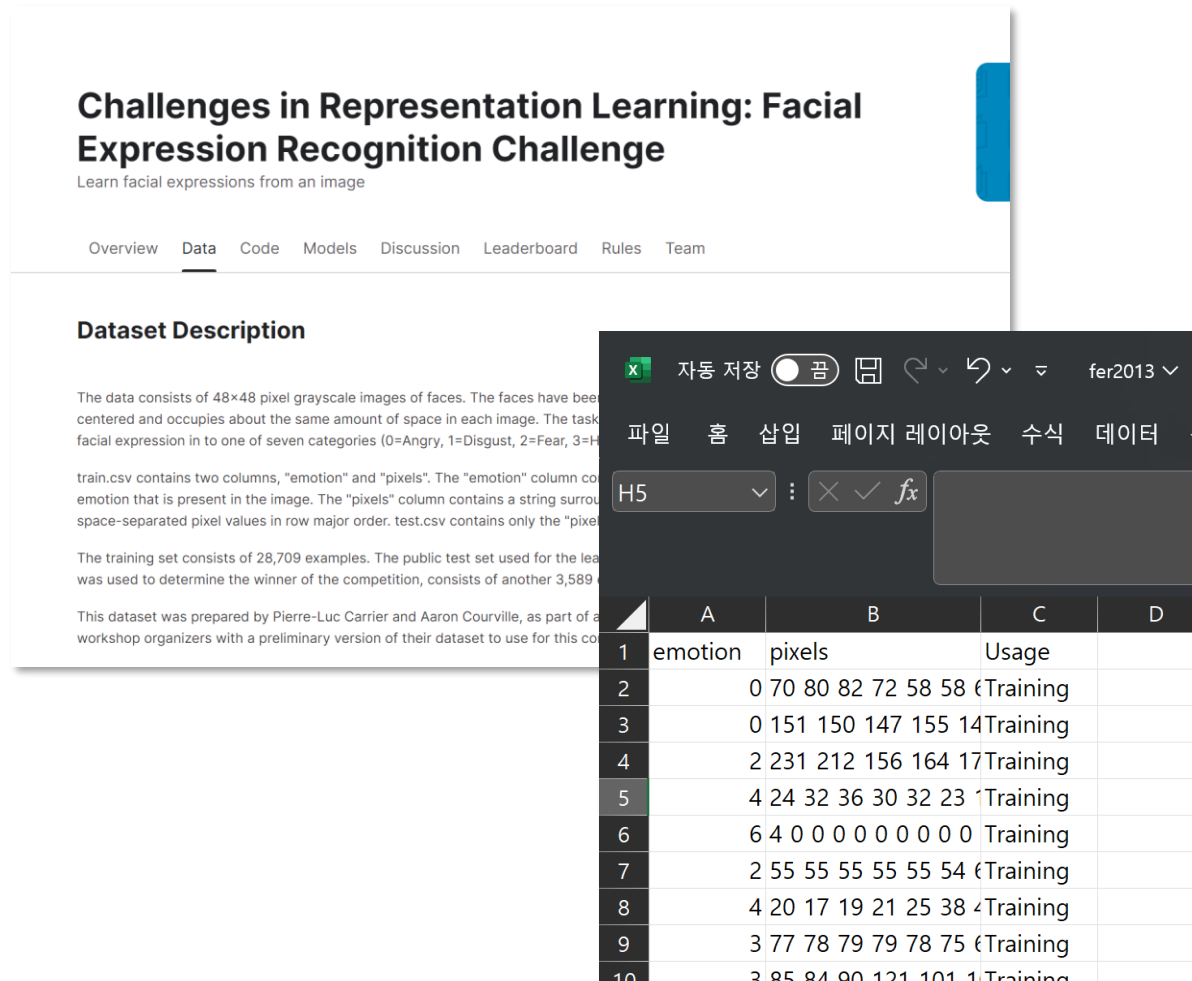
MobileNet(속도 증강)과 ResNet(정확도 증강) 목적

사진의 원본에 마스크를 적용하는 방식으로 데이터셋 설계  
하지만 특정 아시안 국가 사람 상대로 모델이 적합하지 않았음(베트남)  
-> 해결방법 : 직접 데이터셋 구현

결과 : 속도 높지 않았음 + 정확도 높았음 (최대 99.37%)  
한번에 많은 얼굴도 처리 가능

Train data : 8000개

# Dataset(1)



**Challenges in Representation Learning: Facial Expression Recognition Challenge**  
Learn facial expressions from an image

Overview **Data** Code Models Discussion Leaderboard Rules Team

### Dataset Description

The data consists of 48x48 pixel grayscale images of faces. The faces have been centered and occupies about the same amount of space in each image. The task is to recognize the facial expression in to one of seven categories (0=Angry, 1=Disgust, 2=Fear, 3=Happy, 4=Sad, 5=Surprise, 6=Neutral).

train.csv contains two columns, "emotion" and "pixels". The "emotion" column contains the emotion that is present in the image. The "pixels" column contains a string surrounded by double quotes containing space-separated pixel values in row major order. test.csv contains only the "pixels" column.

The training set consists of 28,709 examples. The public test set used for the leaderboard was used to determine the winner of the competition, consists of another 3,589 examples.

This dataset was prepared by Pierre-Luc Carrier and Aaron Courville, as part of a workshop organizers with a preliminary version of their dataset to use for this competition.

	A	B	C	D
1	emotion	pixels	Usage	
2		0 70 80 82 72 58 58 6	Training	
3		0 151 150 147 155 14	Training	
4		2 231 212 156 164 17	Training	
5		4 24 32 36 30 32 23	Training	
6		6 4 0 0 0 0 0 0 0 0	Training	
7		2 55 55 55 55 55 54	Training	
8		4 20 17 19 21 25 38	Training	
9		3 77 78 79 79 78 75	Training	
10		3 85 84 90 121 101 1	Training	

Kaggle의 Facial expression recognition challenge

-emotion 분석

Pixel, emotion(감정)

labeling 되어있는

fer2013.csv file 활용

## Dataset(2)

### UTKFace

Large Scale Face Dataset



UTKFace dataset -sex,age분석

다양한 연령대 및 인종 포함된 대규모 데이터셋

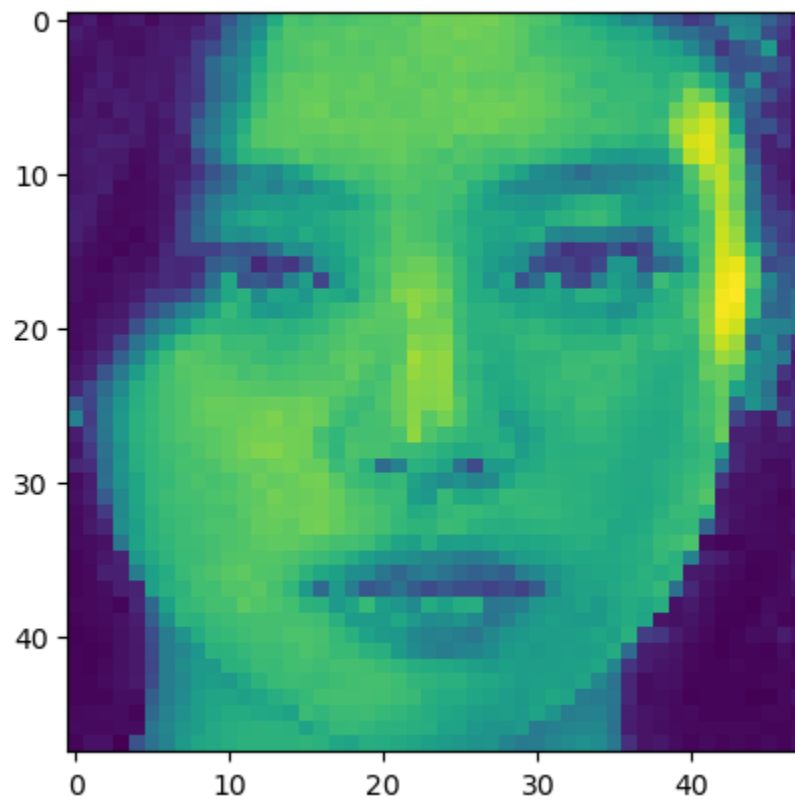
연령, 성별, 인종 정보 레이블 포함

(예를 들어, 파일명이 23\_1\_2\_2017.jpg인 경우  
23은 나이, 1은 성별(남자) , 2는 인종(아시아인))



# Model –pretrained

Haar 기반의 Cascade 얼굴 인식 pretrained model 활용하여 얼굴 객체 인식



# Model(emotion)-preprocessing

1. 이미지 pixel list로 만들기
2. Train-test split (train\_data, test\_data분할)
3. Data set array reshape
4. pixel, emotion 각 X\_train, y\_train , X\_test, y\_test
5. np.vstack 을 통한 구조 변경
6. 4차원 데이터셋 reshape
7. 데이터 type float로 변경
8. Scaling (/255)
9. y\_train,y\_test one hot encoding 진행
10. Input\_shape 설정
11. 라벨 숫자를 문자로 변경  
(0:'angry', 1:'disgust', 2:'fear', 3:'happy', 4:'sad',  
5:'surprise', 6:'neutral')



# Model -emotion prediction(mobilenetV2)

```
from tensorflow.keras.applications import MobileNetV2
from tensorflow.keras.models import Sequential, Model
from tensorflow.keras.layers import Dense, GlobalAveragePooling2D, Dropout
```

```
def mobilenetv2_model(input_shape):
    # MobileNetV2 모델 불러오기 (사전 학습)
    mobilenetv2_model = MobileNetV2(weights='imagenet', include_top=False, input_shape=input_shape)
```

```
    # 재학습을 막기 위한 코드
    for layer in mobilenetv2_model.layers:
        layer.trainable = False
```

```
    model = Sequential()
    model.add(mobilenetv2_model)
```

```
    # GlobalAveragePooling2D를 사용하여 Flatten 대신
    model.add(GlobalAveragePooling2D())
```

```
    # Fully connected layer
    model.add(Dense(512, activation='relu'))
    model.add(Dropout(0.2))
```

```
    # Output layer with 7 classes
    model.add(Dense(7, activation='softmax'))
```

```
    return model
```

```
# 입력 이미지의 shape을 지정하세요 (예: (224, 224, 3))
input_shape = (48, 48, 3)
```

```
# MobileNetV2 모델 생성
model = mobilenetv2_model(input_shape)
```

```
# 모델 요약 출력
model.summary()
```

mobilenetV2로 구현된 감정분석 model

(이미지 처리에 성능이 좋고 가벼운 구조라고  
소문나서 기대하며 진행 해봤는데..)

Model: "sequential\_1"

Layer (type)	Output Shape	Param #
mobilenetv2_1.00_224 (Functional)	(None, 2, 2, 1280)	2257984
global_average_pooling2d_1 (GlobalAveragePooling2D)	(None, 1280)	0
dense_2 (Dense)	(None, 512)	655872
dropout_1 (Dropout)	(None, 512)	0
dense_3 (Dense)	(None, 7)	3591

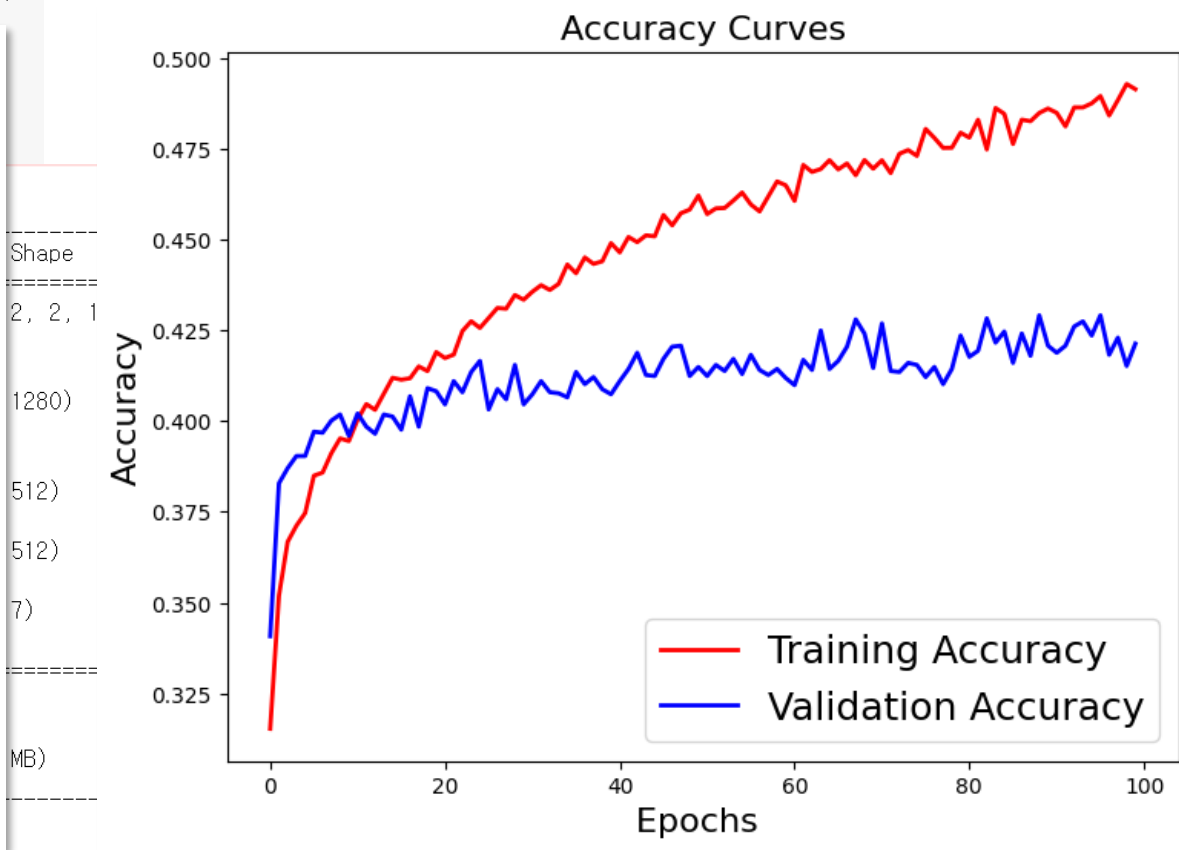
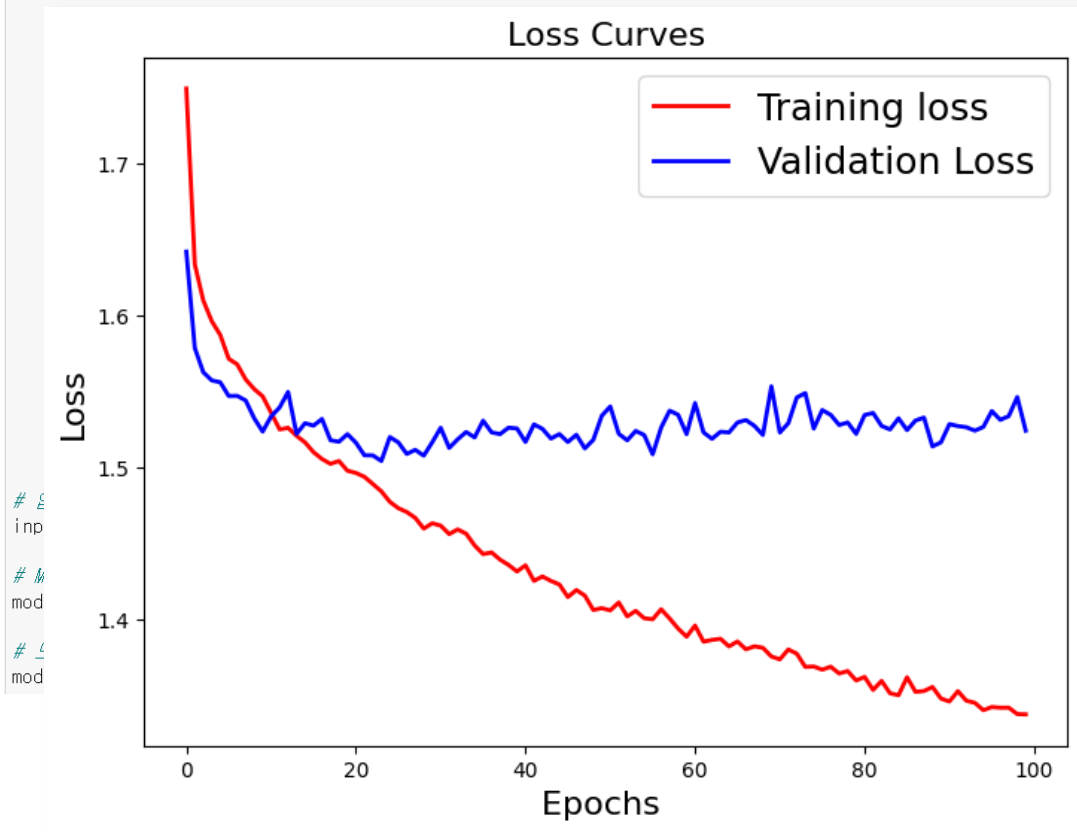
=====  
Total params: 2917447 (11.13 MB)  
Trainable params: 659463 (2.52 MB)  
Non-trainable params: 2257984 (8.61 MB)  
=====

# Model - emotion prediction(mobilenetV2)

감정분석 먼저 mobilenetV2모델 구축 이후, 성능 확인 해보니  
Epoch100을 학습해도 다소 낮은 성능.. (속도보단 성능이지)  
그래서 3개의 모델 모두 CNN구조로 모델을 구축하기로 결정!

```
from tensorflow.keras.applications import MobileNetV2
from tensorflow.keras.models import load_model
from tensorflow.keras.layers import Input, Dense, GlobalAveragePooling2D

def mobilenetv2_model(input_shape):
    # MobileNetV2 모델 불러오기 (가중치 포함)
    mobilenetv2_model = MobileNetV2(weights='imagenet', include_top=False, input_shape=input_shape)
```



# Model -emotion prediction(CNN)

#감정분석 모델 라인

```
def cnn_model():
    model = Sequential()

    # Input layer
    model.add(Conv2D(32, (3, 3), padding='same', activation='relu'))

    # Add layers
    model.add(Conv2D(32, (3, 3), activation='relu'))
    model.add(MaxPooling2D(pool_size=(2, 2)))
    model.add(Dropout(0.2))

    model.add(Conv2D(64, (3, 3), padding='same', activation='relu'))
    model.add(Conv2D(64, (3, 3), activation='relu'))
    model.add(MaxPooling2D(pool_size=(2, 2)))
    model.add(Dropout(0.2))

    model.add(Conv2D(64, (3, 3), padding='same', activation='relu'))
    model.add(Conv2D(64, (3, 3), activation='relu'))
    model.add(MaxPooling2D(pool_size=(2, 2)))
    model.add(Dropout(0.2))

    # Flatten
    model.add(Flatten())

    # Fully connected layer
    model.add(Dense(512, activation='relu'))

    # Output layer : n_classes=7
    model.add(Dense(7, activation='softmax'))

    return model
```

Model: "sequential"

Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 48, 48, 32)	320
conv2d_1 (Conv2D)	(None, 46, 46, 32)	9248
max_pooling2d (MaxPooling2D)	(None, 23, 23, 32)	0
dropout (Dropout)	(None, 23, 23, 32)	0
conv2d_2 (Conv2D)	(None, 23, 23, 64)	18496
conv2d_3 (Conv2D)	(None, 21, 21, 64)	36928
max_pooling2d_1 (MaxPooling2D)	(None, 10, 10, 64)	0
dropout_1 (Dropout)	(None, 10, 10, 64)	0
conv2d_4 (Conv2D)	(None, 10, 10, 64)	36928
conv2d_5 (Conv2D)	(None, 8, 8, 64)	36928
max_pooling2d_2 (MaxPooling2D)	(None, 4, 4, 64)	0
dropout_2 (Dropout)	(None, 4, 4, 64)	0
flatten (Flatten)	(None, 1024)	0
dense (Dense)	(None, 512)	524800
dense_1 (Dense)	(None, 7)	3591

=====  
Total params: 667239 (2.55 MB)  
Trainable params: 667239 (2.55 MB)  
Non-trainable params: 0 (0.00 Byte)  
=====

## Input layer

32개의 3\*3 크기의 filter

입력 이미지의 크기와 형태 그대로(padding = 'same')

Activation function = ReLu

## Convolution layer

여러 개의 Conv2D와 MaxPoolin2D계층을 쌓음

Dropout(0.2) 를 통해 노드(뉴런)의 20%를 임의로 비활성화 (모델이 특정 뉴런에 과도하게 의존하는 것을 방지하기 위함)

## Flatten layer

이전 계층에서 추출된 특징을 1차원 배열로 바꿔줌.

## Fully Connected Layer

512개의 뉴런을 가진 계층

## Output layer

7개의 노드를 가짐 activation function = softmax

# Model -emotion prediction(CNN)

```
# 이미지 데이터 증강
datagen = ImageDataGenerator(zoom_range=0.2,          # 랜덤하게 이미지 줌 하는 비율
                             rotation_range=10,      # 랜덤하게 이미지 회전하는 비율 (0도~180도)
                             width_shift_range=0.1,   # 랜덤하게 이미지 가로로 이동하는 비율
                             height_shift_range=0.1,  # 랜덤하게 이미지 세로로 이동하는 비율
                             horizontal_flip=True,    # 랜덤하게 이미지 수평 뒤집기
                             vertical_flip=False)     # 랜덤하게 이미지 수직 뒤집기

# 모델 학습을 위한 파라미터 설정
batch_size = 256
n_epochs = 100
model.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'])
history = model.fit_generator(datagen.flow(train_data, train_labels_onehot, batch_size=batch_size),
                             steps_per_epoch=int(np.ceil(train_data.shape[0]/float(batch_size))),
                             epochs=n_epochs,
                             validation_data=(test_data, test_labels_onehot))
```

이미지 데이터 증강을 위한 ImageDataGenerator 클래스

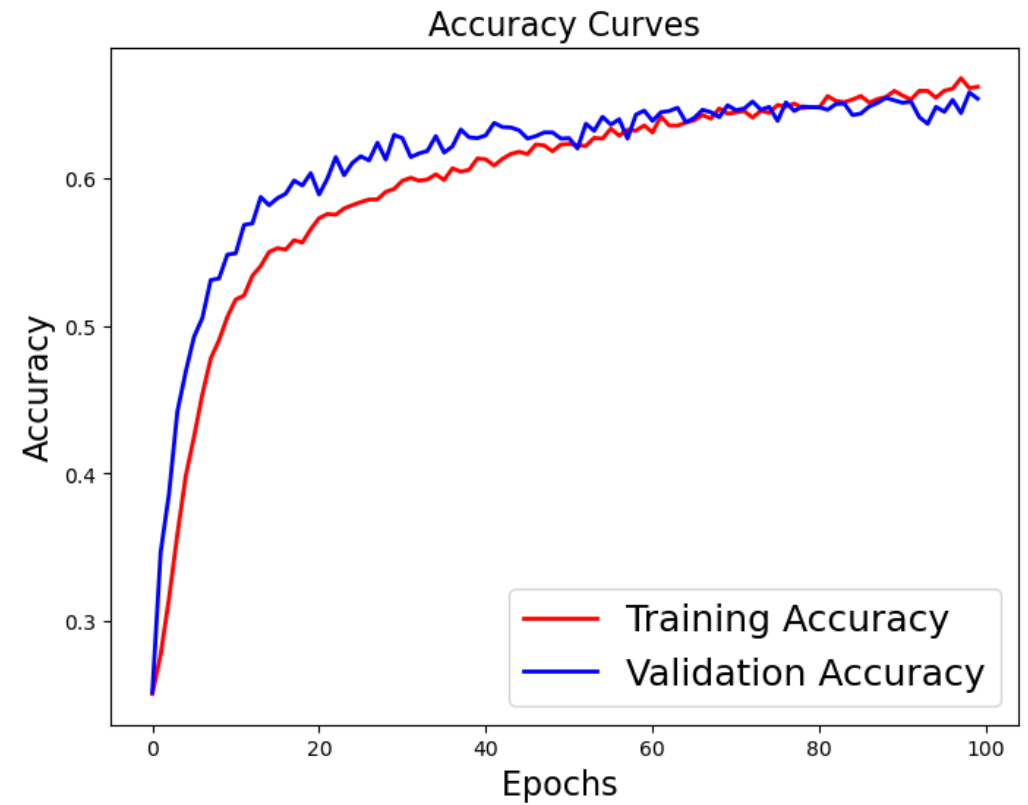
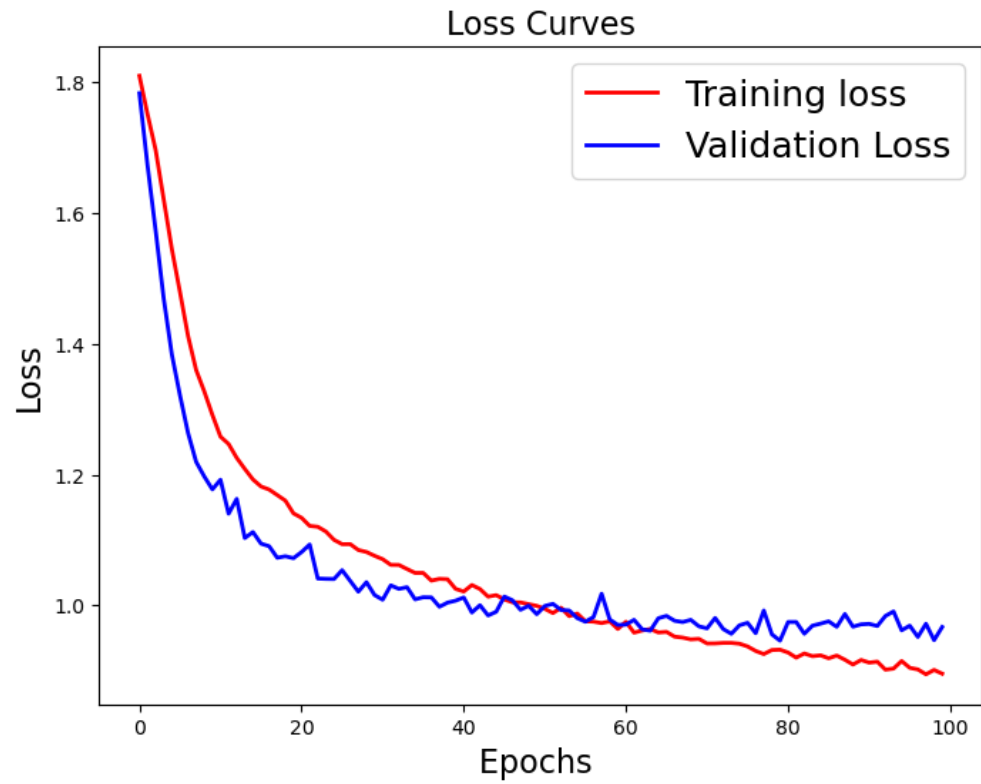
- zoom\_range : 이미지를 랜덤하게 확대/축소
- rotation\_range: 이미지를 랜덤하게 회전
- width\_shift\_range: 이미지를 수평방향으로 랜덤하게 이동
- height\_shift\_range: 이미지를 수직방향으로 랜덤하게 이동
- horizontal\_flip: 이미지를 수평방향으로 랜덤하게 뒤집
- vertical\_flip: 이미지를 수직방향으로 뒤집기 (False 니까 안 뒤집음)

Optimizer = Adam

loss function = categorical\_crossentropy

성능 평가 accuracy

# Model –emotion prediction(CNN)



# Model –preprocessing

1. tar.gz (압축파일) 압축 해제 (이미지파일)
2. Age, gender, race, filename의 Dataframe (각 이미지 파일명을 분석하여 csv파일로 저장)
3. 각 image를 RGB로 변환
4. Resize(48,48)
5. Scaling(/255)
6. 결측치 제거 (이미지로드 중 오류 발생 항목)
7. Train-test split (8:2)

	age	gender	race	filename
0	2	1	2	2_1_2_20161219202547820.jpg
1	77	1	0	77_1_0_20170110122639530.jpg
2	1	1	0	1_1_0_20170109190844250.jpg
3	29	1	2	29_1_2_20170105164315483.jpg
4	76	1	0	76_1_0_20170110131744527.jpg
...	...	...	...	...
908	1	0	2	1_0_2_20161219201528716.jpg
909	53	0	0	53_0_0_20170109012721891.jpg
910	99	1	2	99_1_2_20170110182418864.jpg
911	52	1	0	52_1_0_20170105173045268.jpg
912	18	1	3	18_1_3_20170109214031495.jpg



# Model -age prediction(CNN)

```
# Define the sex model
input_shape = (48, 48, 3)
input_layer = Input(shape=input_shape)

input = Input(shape=(48, 48, 3))

cnn1 = Conv2D(128, kernel_size=3, activation='relu')(input)
cnn1 = Conv2D(128, kernel_size=3, activation='relu')(cnn1)
cnn1 = Conv2D(128, kernel_size=3, activation='relu')(cnn1)
cnn1 = MaxPool2D(pool_size=3, strides=2)(cnn1)

cnn2 = Conv2D(128, kernel_size=3, activation='relu')(cnn1)
cnn2 = Conv2D(128, kernel_size=3, activation='relu')(cnn2)
cnn2 = Conv2D(128, kernel_size=3, activation='relu')(cnn2)
cnn2 = MaxPool2D(pool_size=3, strides=2)(cnn2)

dense = Flatten()(cnn2)
dense = Dropout(0.2)(dense)
dense = Dense(1024, activation='relu')(dense)
dense = Dense(1024, activation='relu')(dense)

output = Dense(1, activation='linear', name='age')(dense)

model_age = Model(input, output)
model_age.compile(optimizer=Adam(0.0001), loss='mse', metrics=['mae'])
```

## Input layer

48\*48 크기의 3채널(RGB)이미지를 입력으로 받음

## Convolution layer

연속된 세개의 Conv2D layer는 128개의 필터와 3\*3의 커널 크기를 가지고 있음  
activation function = Relu

## Pooling layer

MaxPool2D(pool\_size = 3, strides = 2)로 설정하여 첫번째 pooling 계층으로 이미지의 크기를 줄이고 중요한 특징을 유지하고자 함.

## Convolution-pooling block

구조를 살펴보면 또 다른 세 개의 'Conv2D' 계층 후에 MaxPool2D풀링레이어가 뒤따름.

## Flatten layer

1차원 벡터로 변환

## Dense layer(fully connected layer)

Dense(1024, activation = 'relu')

Dropout(0.2) (앞선 모델과 동일한 설정)

두번째 Dense (위와 동일한 구조)

## Output layer

Dense(1, activation = 'linear', name = 'age') 최종 출력 계층 한 개의 뉴런으로 나이를 예측하는 연속값이며 선형함수를 사용하여 모델을 구축한다는 이야기

Optimizer로 Adam

Loss function은 MSE

성능지표로 MAE 사용

# Model -sex prediction(CNN)

```
# Define the sex model
input_shape = (48, 48, 3)
input_layer = Input(shape=input_shape)

cnn1 = Conv2D(36, kernel_size=3, activation='relu')(input)
cnn1 = MaxPool2D(pool_size=3, strides=2)(cnn1)
cnn2 = Conv2D(64, kernel_size=3, activation='relu')(cnn1)
cnn2 = MaxPool2D(pool_size=3, strides=2)(cnn2)
cnn3 = Conv2D(128, kernel_size=3, activation='relu')(cnn2)
cnn3 = MaxPool2D(pool_size=3, strides=2)(cnn3)

dense = Flatten()(cnn3)
dense = Dropout(0.2)(dense)
dense = Dense(512, activation='relu')(dense)
dense = Dense(512, activation='relu')(dense)
output = Dense(1, activation='sigmoid', name='gender')(dense)
model_sex = Model(input, output)
model_sex.compile(optimizer=Adam(learning_rate=0.0001), loss='binary_crossentropy', metrics=['accuracy'])
```

## Input layer

48\*48 크기의 3채널(RGB)이미지 입력

## Convolution and Pooling layer

첫번째 컨볼루션 레이어, 36개의 필터와 relu 활성화 함수 정의

첫번째 pooling layer로서 피쳐맵 크기 줄이고 특징 보존

두번째 컨볼루션 레이어, 64개 필터 사용

세번째 컨볼루션 레이어, 128개의 필터사용 그리고 각각에 대한 MaxPooling적용

## Flatten and Dense Layer

1차원 벡터로 변환 해주고 앞서 진행한 구성과 같이 Dropout(0.2)로 설정

두개의 Dense layer로 각각 512개의 뉴런을 가지고 ReLu활성화 함수 사용

## Output layer

성별 분류를 위한 하나의 노드

Sigmoid 활성화 함수사용 (이진분류 문제에 적합)

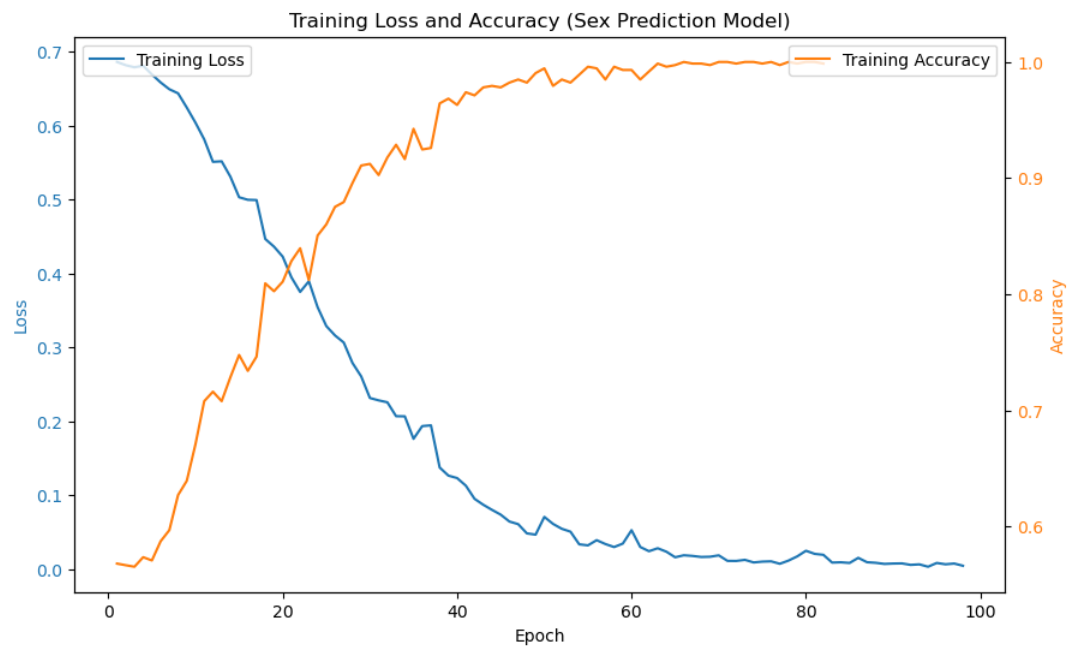
Optimizer : Adam (LR = 0.0001)

Loss function= 'binary\_crossentropy'

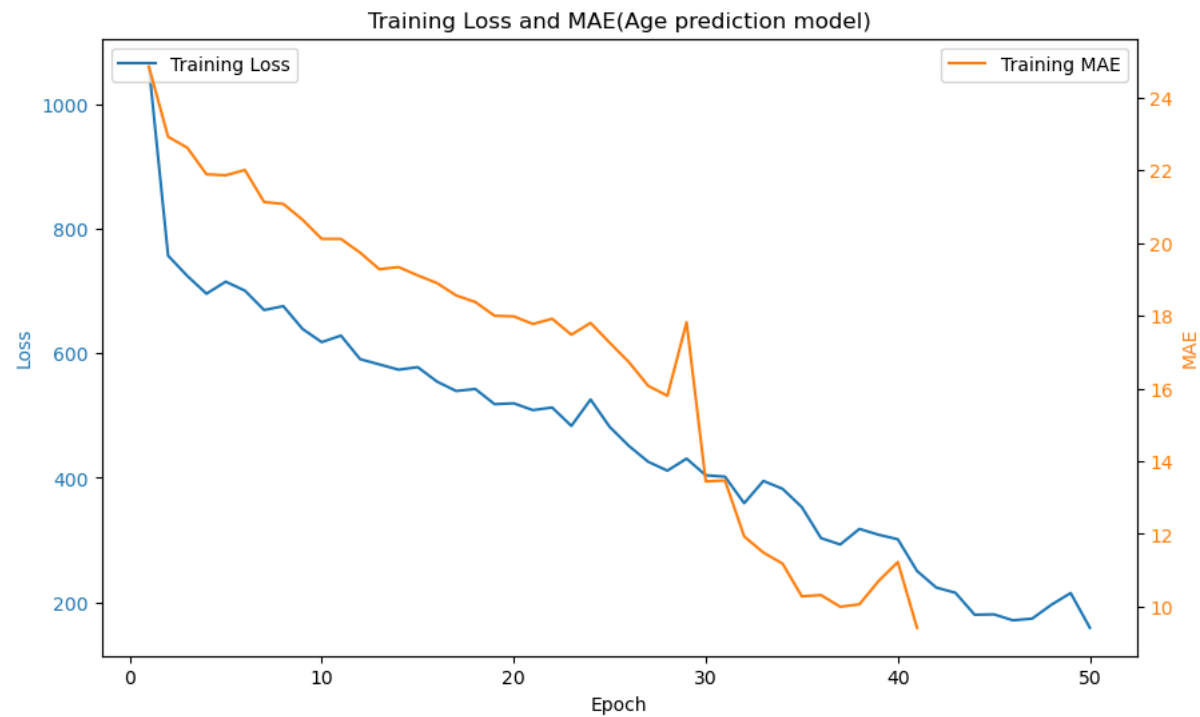
평가지표 : accuracy

# Model -성능확인

성별(SEX)



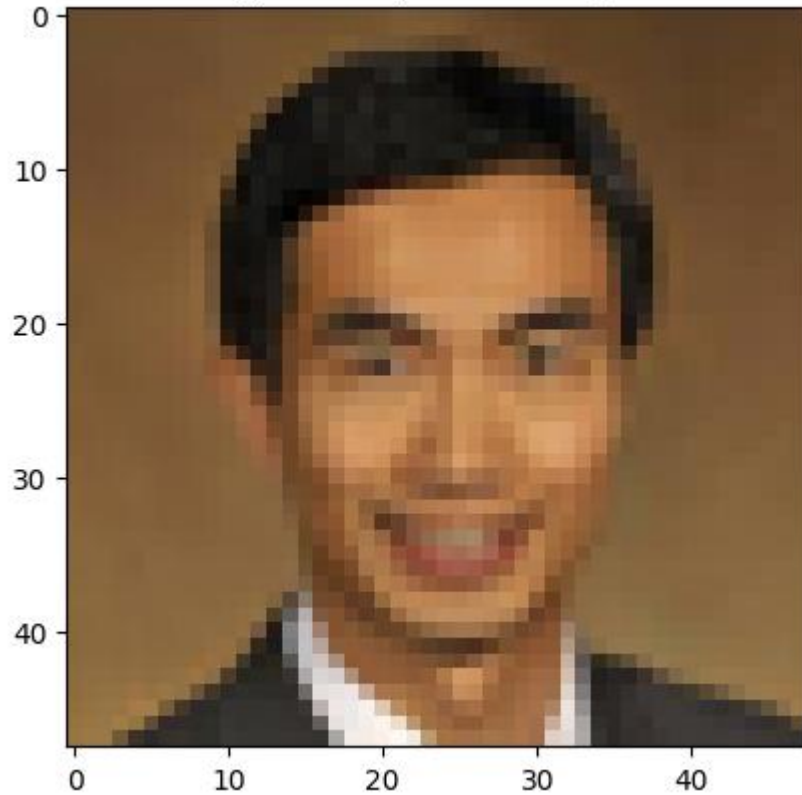
나이(AGE)



# Model - 성능 확인

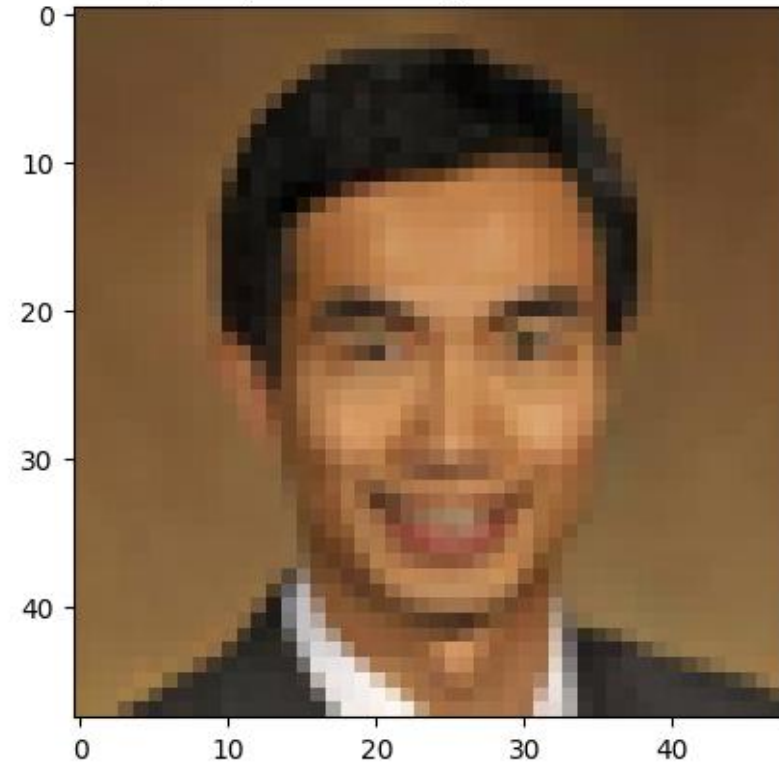
성별(SEX)(0:남자, 1:여자)

Actual gender: 0, Predicted gender: 0



나이(AGE)

Actual Age: 42, Predicted Age: 21.37274169921875



# Webcam 불러오기

1. shape\_x,y (48\*48)로 기본 설정
2. 얼굴 감지 함수(cascade pretrained model(.xml) load)
3. 얼굴 추출함수(extract\_face\_features) 정의
4. 3개 Model load (emotion\_model,age\_model,sex\_model)
5. Cv2.VideoCapture(0)을 통해 비디오 캡처 시작
6. 무한 루프안에서 비디오 캡처 객체에서 프레임을 읽어와 RGB와 grayscale로 변환해줌
7. 검출된 성별, 나이,감정 예측

> 비디오 프레임을 입력으로 받아 성별,나이, 감정 예측 진행

```
if not ret:
    break

# RGB로 변환
rgb_frame = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)
# 그레이스케일로 변환
gray_frame = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)

faces = detect_face(frame)

for (x, y, w, h) in faces:
    # 원본 컬러 이미지에서 얼굴 부분 추출 (성별 및 나이 예측용)
    color_face = rgb_frame[y:y+h, x:x+w]
    # 원본 그레이스케일 이미지에서 얼굴 부분 추출 (감정 예측용)
    gray_face = gray_frame[y:y+h, x:x+w]

    # 컬러 얼굴 이미지 처리
    resized_color_face = cv2.resize(color_face, (48, 48))
    normalized_color_face = resized_color_face / 255.0
    reshaped_color_face = np.reshape(normalized_color_face, (1, 48, 48, 3))

    # 그레이스케일 얼굴 이미지 처리
    resized_gray_face = cv2.resize(gray_face, (48, 48))
    normalized_gray_face = resized_gray_face / 255.0
    reshaped_gray_face = np.reshape(normalized_gray_face, (1, 48, 48, 1))

    # 예측
    sex_preds = sex_model.predict(reshaped_color_face)
    age_preds = age_model.predict(reshaped_color_face)
    emotion_preds = emotion_model.predict(reshaped_gray_face)

    # 결과 처리
    sex_text = 'Female' if sex_preds[0][0] > 0.5 else 'Male'
    age_text = str(int(age_preds[0][0]))
    emotion_text = emotion_dict[np.argmax(emotion_preds[0])]

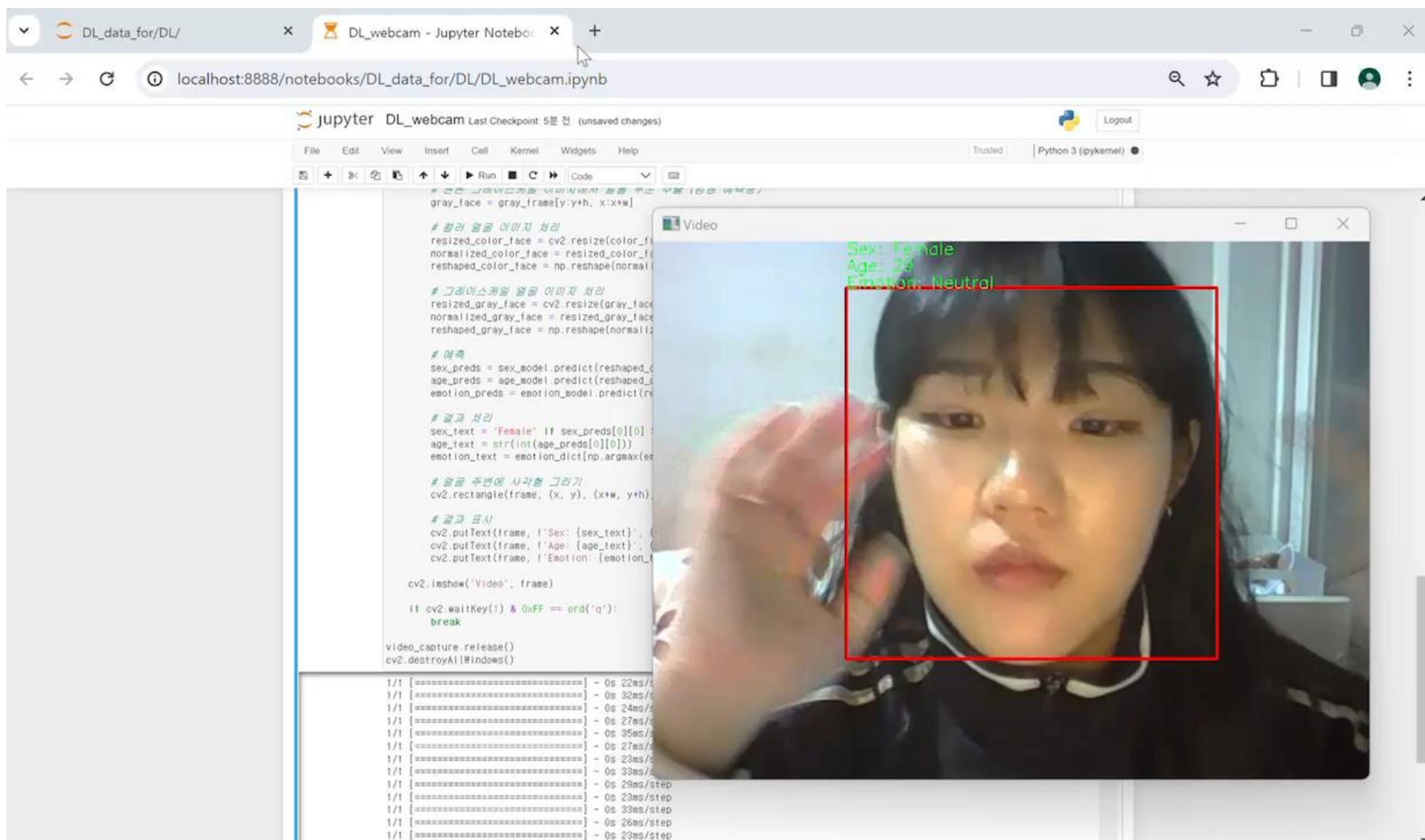
    # 얼굴 주변에 사각형 그리기
    cv2.rectangle(frame, (x, y), (x+w, y+h), (0, 0, 255), 2)

    # 결과 표시
    cv2.putText(frame, f'Sex: {sex_text}', (x, y-30), cv2.FONT_HERSHEY_SIMPLEX, 0.5, (36,255,12), 1)
    cv2.putText(frame, f'Age: {age_text}', (x, y-15), cv2.FONT_HERSHEY_SIMPLEX, 0.5, (36,255,12), 1)
    cv2.putText(frame, f'Emotion: {emotion_text}', (x, y), cv2.FONT_HERSHEY_SIMPLEX, 0.5, (36,255,12), 1)

cv2.imshow('Video', frame)

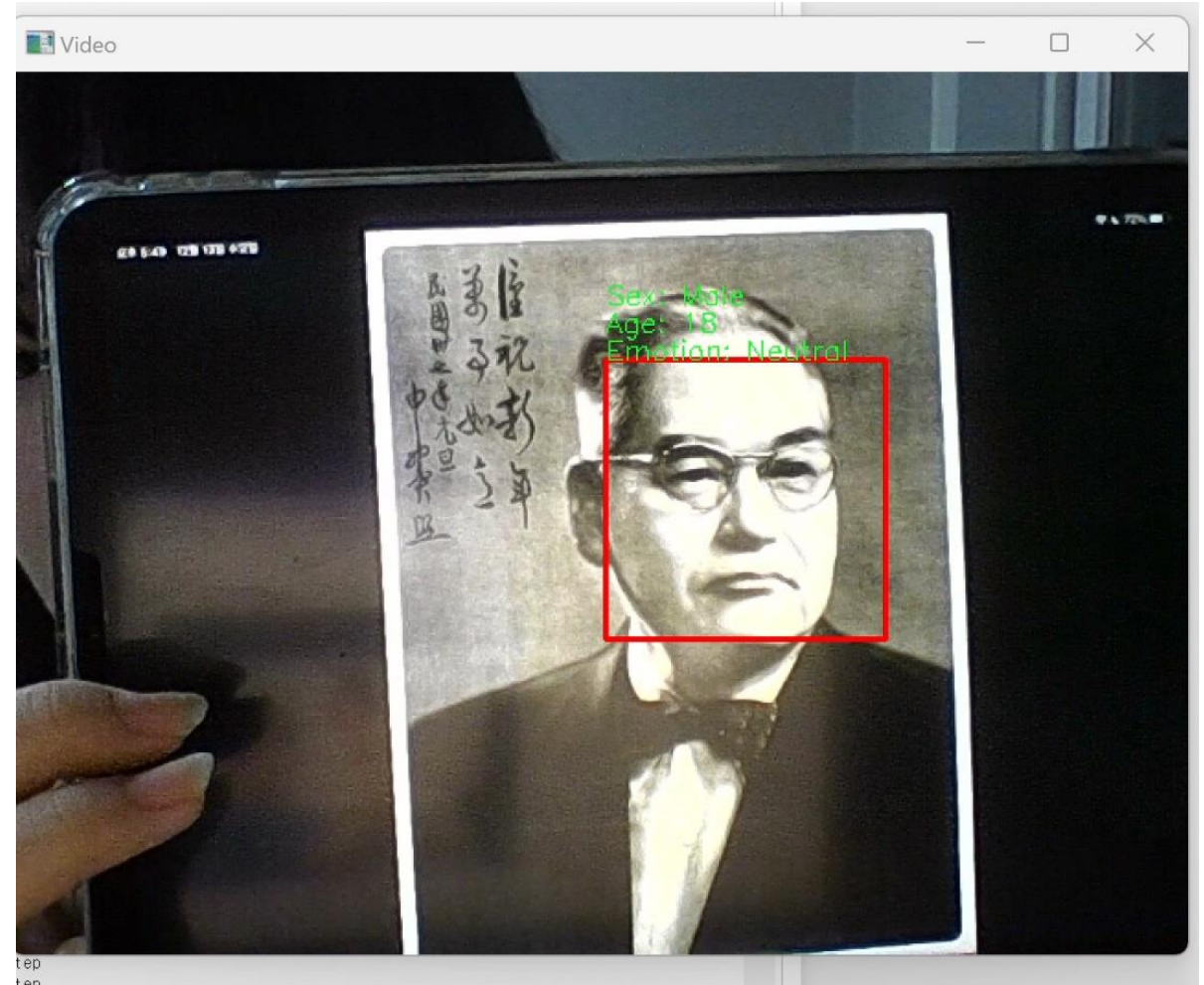
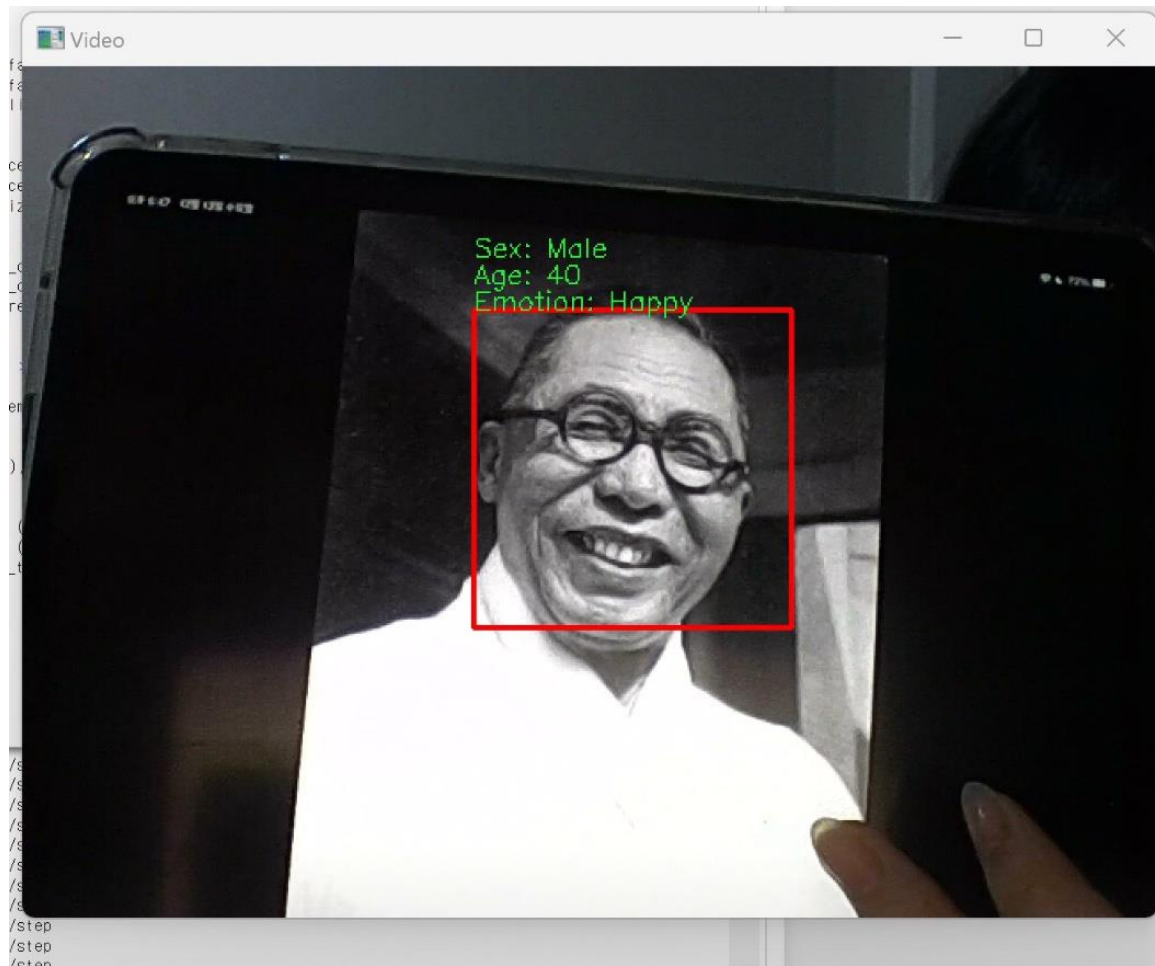
if cv2.waitKey(1) & 0xFF == ord('q'):
    break
```

# 결과

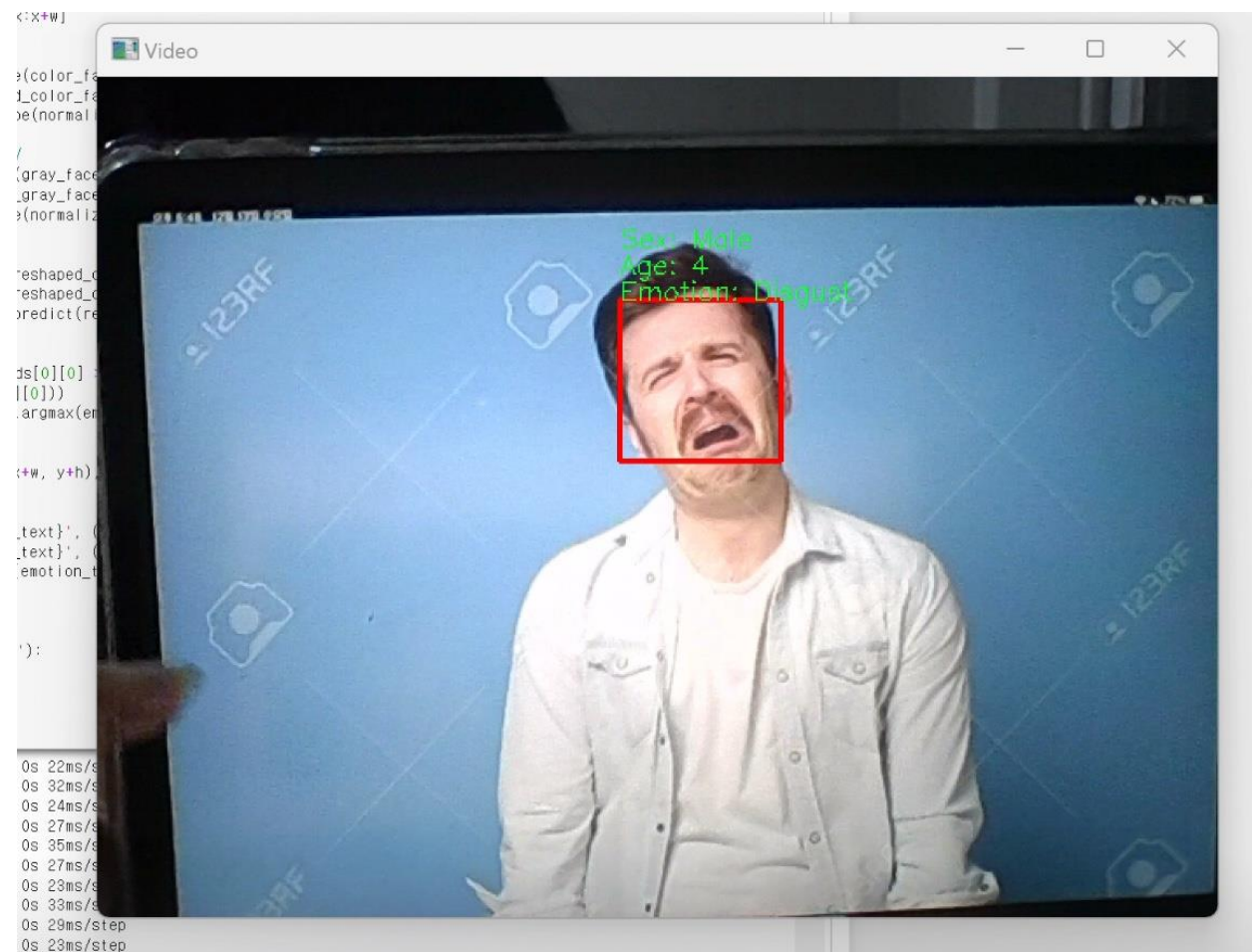
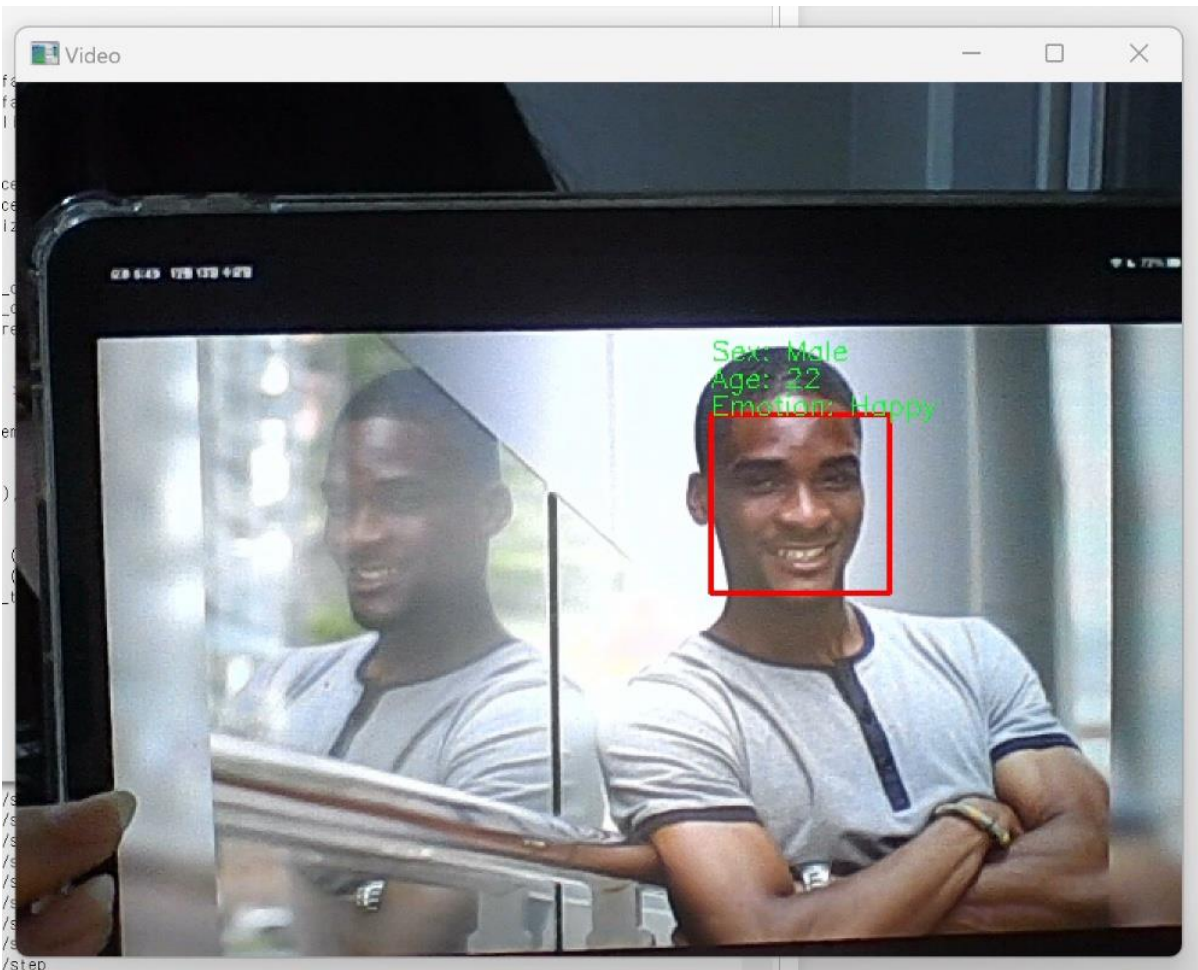




# 결과

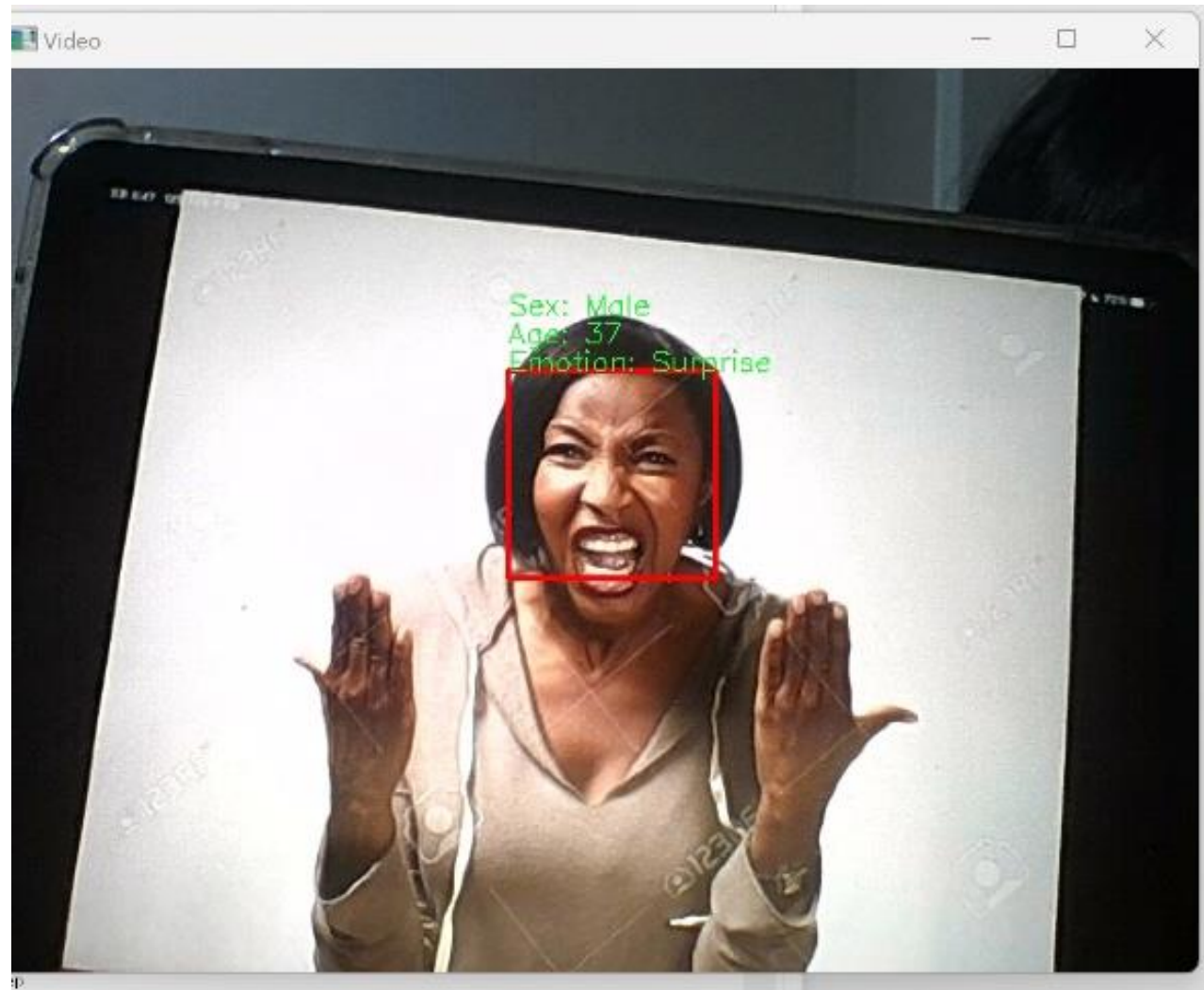


# 결과

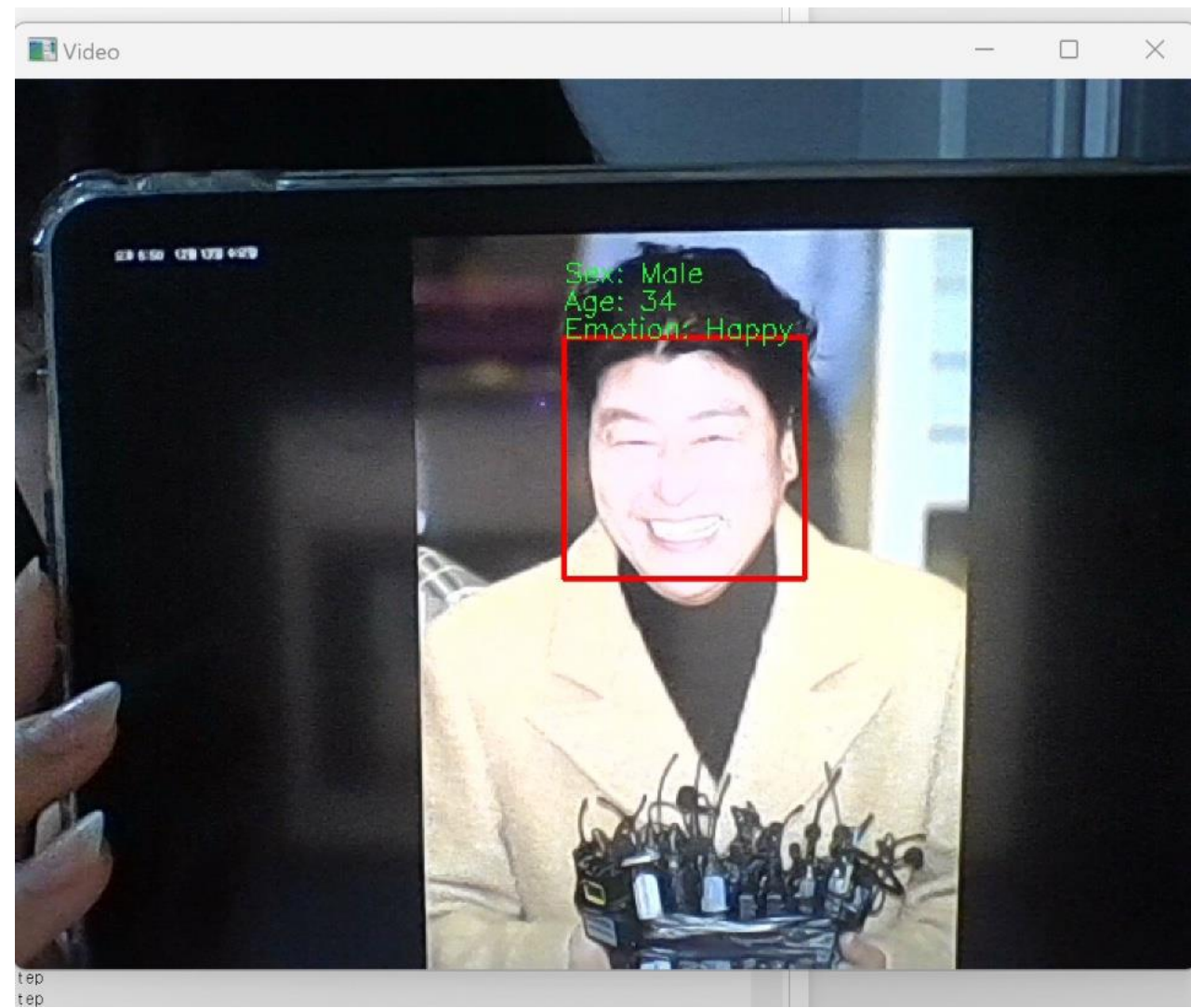




# 결과



## 결과

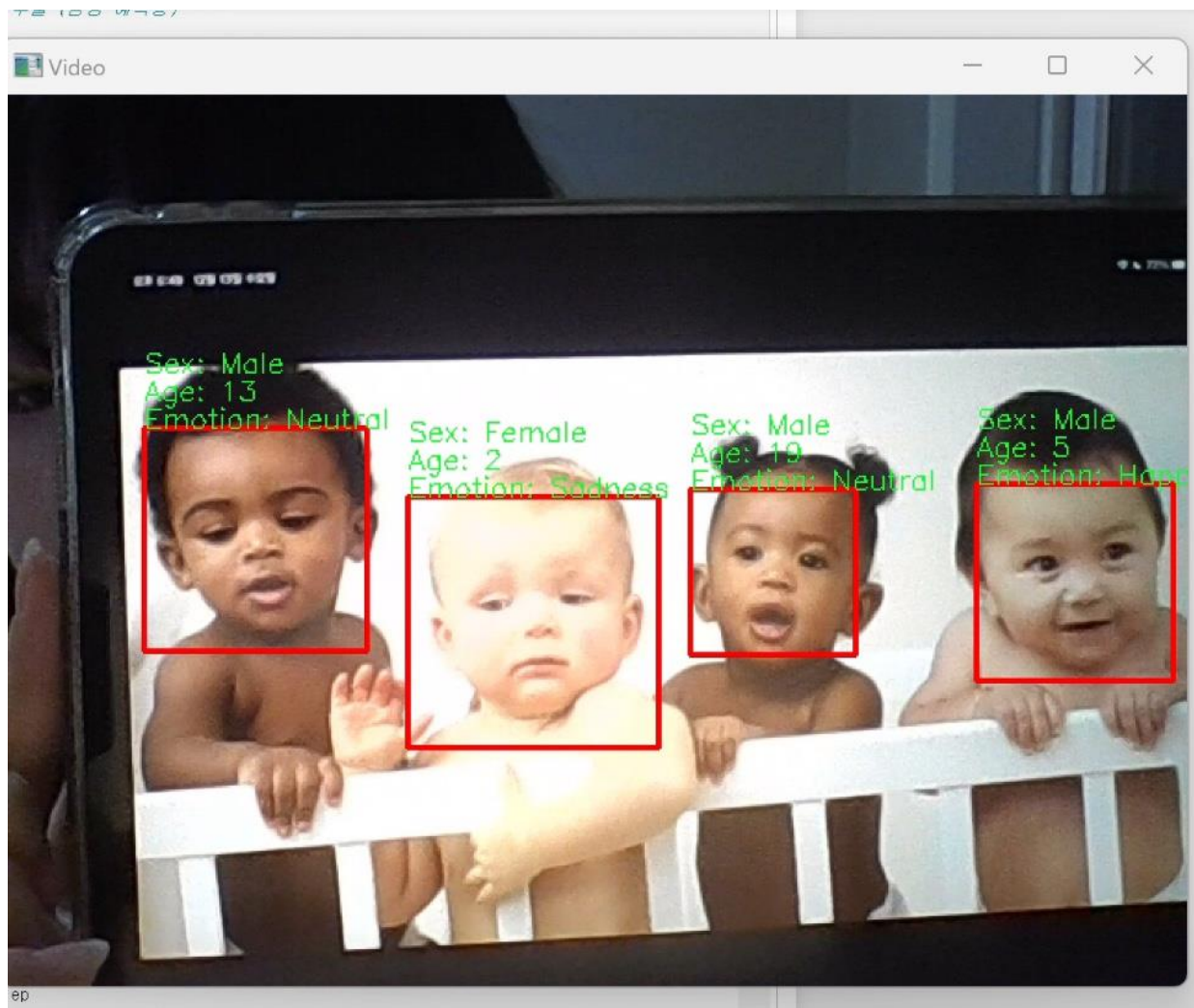


# 결과

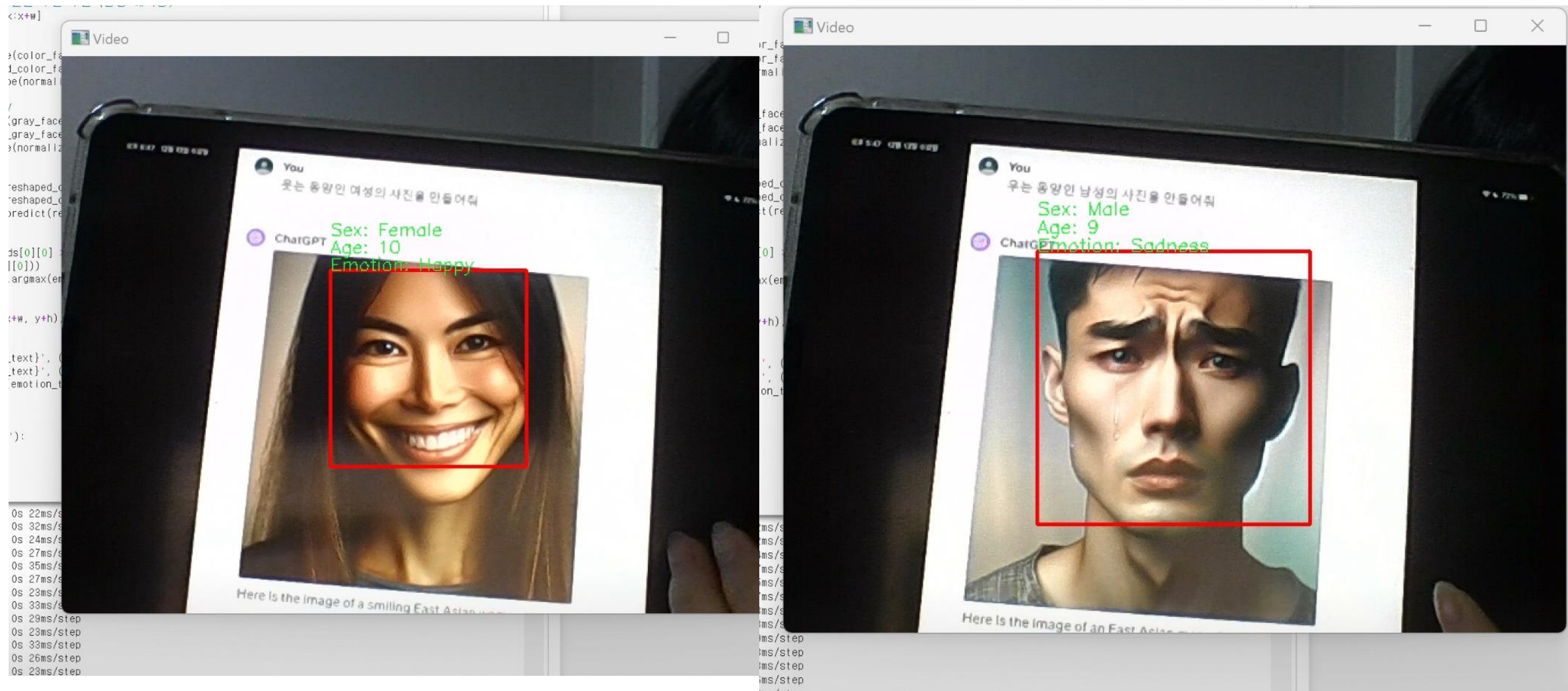




# 결과

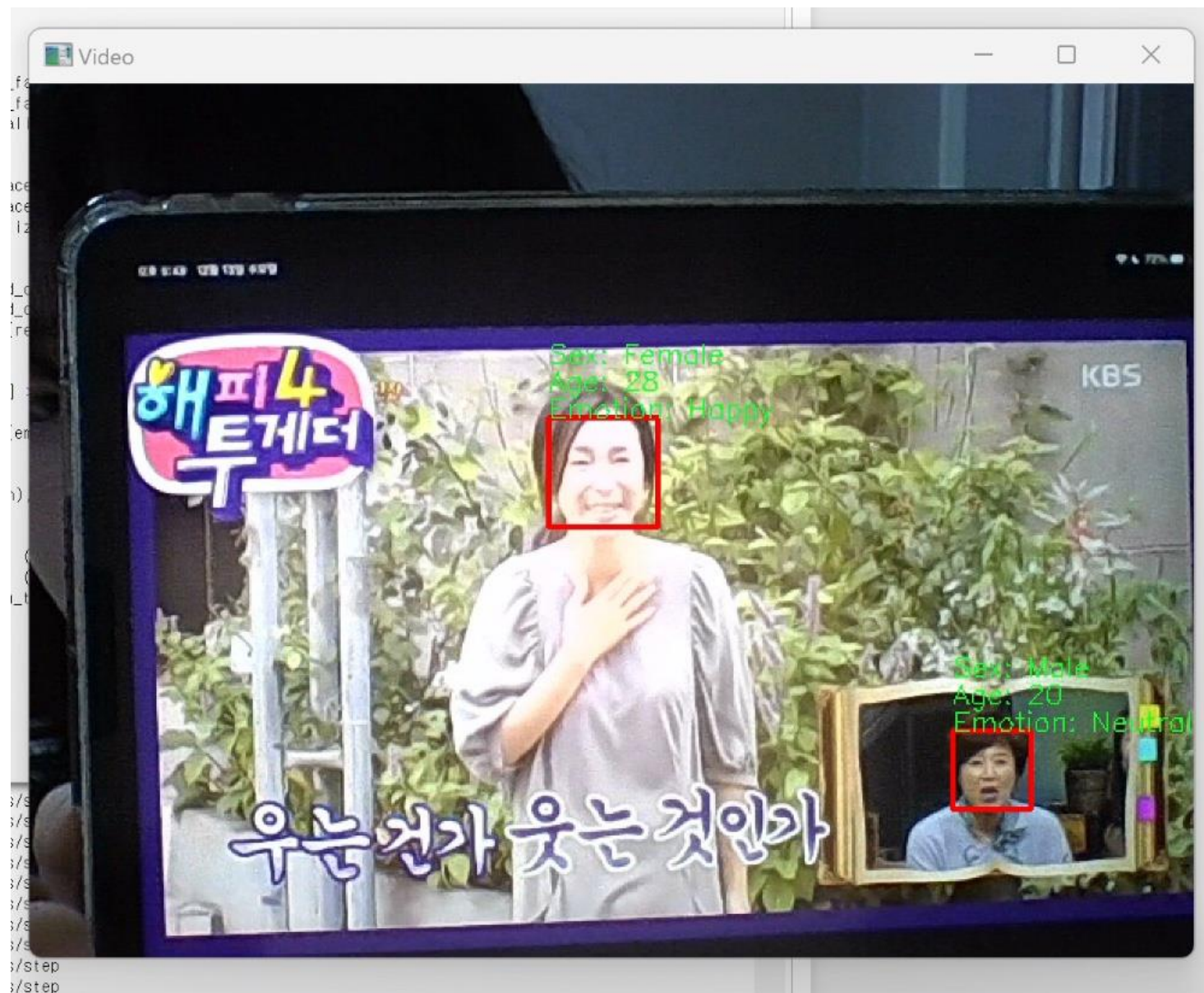


# 결과





# 결과



## 보완점

- 머리가 짧으면 남자로 인식하고 머리가 길면 여자로 인식하는 경우가 많음
- 복합적인 감정 설정 보완 필요 (행복해서 비명을 지르는 사진은 그냥 disgust로 구별..
- Mobilenet, CNN보다 성능이 좋은 모델을 통해 성능을 강화하고 싶었으나 한계
- 라벨링된 이미지 데이터를 구하는데 있어서 한계가 있어 더 많은 학습데이터가 있었다면,  
그리고 데이터증강과 일반화를 더욱 진행했다면 성능이 조금 더 좋지 않았을까..

# Q&A



Thank you