Your Face

Deokju Sull

Contents

01 Introduction

02 Tools

03 Dataset

04 Model

05 Result

06 End

01

■ Introduction

Image Analysis

Security

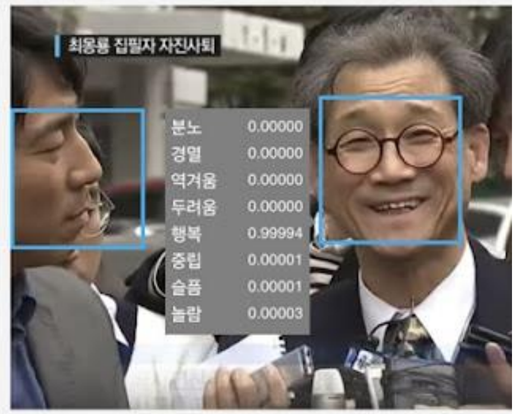
Object Recognition

Entertainment

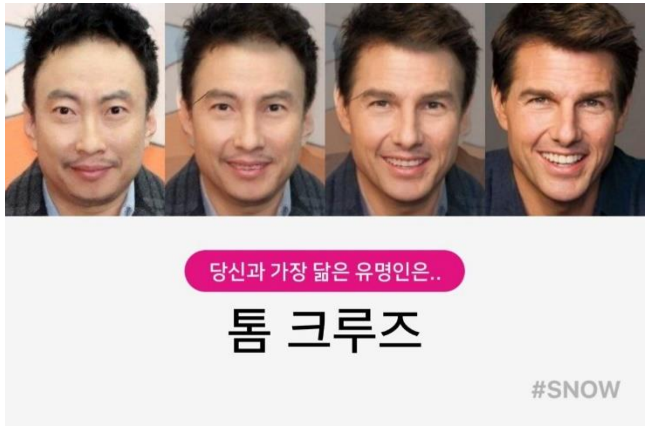
이미지 분석 / 얼굴 분석

- 보안, 객체 인식, 엔터테인먼트 등 다양한 분야
- 결과물의 시각화와, 분석 결과가 명확함
- 이미지 / 영상 처리 분야 산업 인력 수요 多

Face Analysis



```
분노    0.00000
경멸    0.00000
역겨움  0.00000
두려움  0.00000
행복    0.99994
중립    0.00001
슬픔    0.00001
놀람    0.00003
```



당신과 가장 닮은 유명인은..

톰 크루즈

#SNOW

주제 선정

- 기존 시중에 많은 단일 특성 분석 Application

➢ 정제된 데이터셋이 아닌 직접 수집

➢ 이미지 분석의 기반인 전처리 및 학습 정확도 상승

02

Tools

# Data / Preprocessing

Image Downloader

NumPy

Model Train / Visualize

K

Visual Studio Code

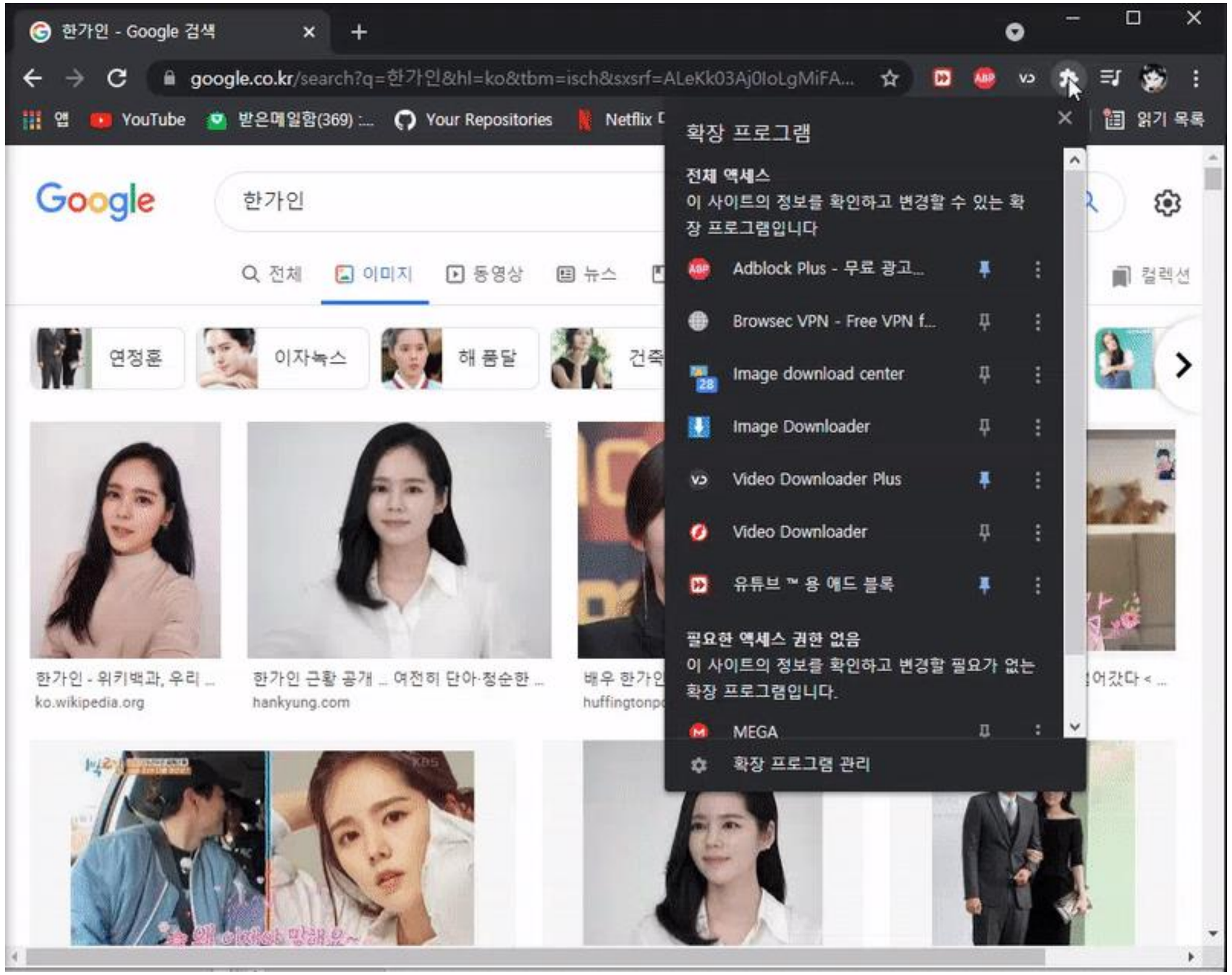
matplotlib

IMAGENET

03

Dataset

# Scrap Google Image



확장 프로그램

전체 액세스
이 사이트의 정보를 확인하고 변경할 수 있는 확장 프로그램입니다

- Adblock Plus - 무료 광고...
- Browsec VPN - Free VPN f...
- Image download center
- Image Downloader
- Video Downloader Plus
- Video Downloader
- 유튜브 ™ 용 애드 블록

필요한 액세스 권한 없음
이 사이트의 정보를 확인하고 변경할 필요가 없는 확장 프로그램입니다.

- MEGA

확장 프로그램 관리

IMAGE DOWNLODER

- Scrapper 구현 시간 부족 판단
- 전체 사진 빠른 판단 후 검색어 변경을 통해 연관성 높은 이미지만 나오게 검색
- 저장 별 폴더 명 지정 간편함

- 45 person , 240 pics each

0	공유	5	김우빈	10	민경훈	15	서인국
1	공명	6	김윤석	11	박보검	16	성규
2	김옥빈	7	나나	12	박보영	17	솔라
3	김태우	8	동해	13	박성웅	18	송중기
4	김완선	9	려욱	14	비니	19	신민아

20	아이린	25	옥택연	30	윤시윤	35	조진웅	40	하연수
21	아이유	26	유인나	31	은하	36	주원	41	한소희
22	안재홍	27	유인영	32	이특	37	차예련	42	한예슬
23	예리	28	유정	33	한혜진	38	청하	43	한지민
24	오하영	29	육성재	34	전소민	39	태민	44	한채영

03

Dataset

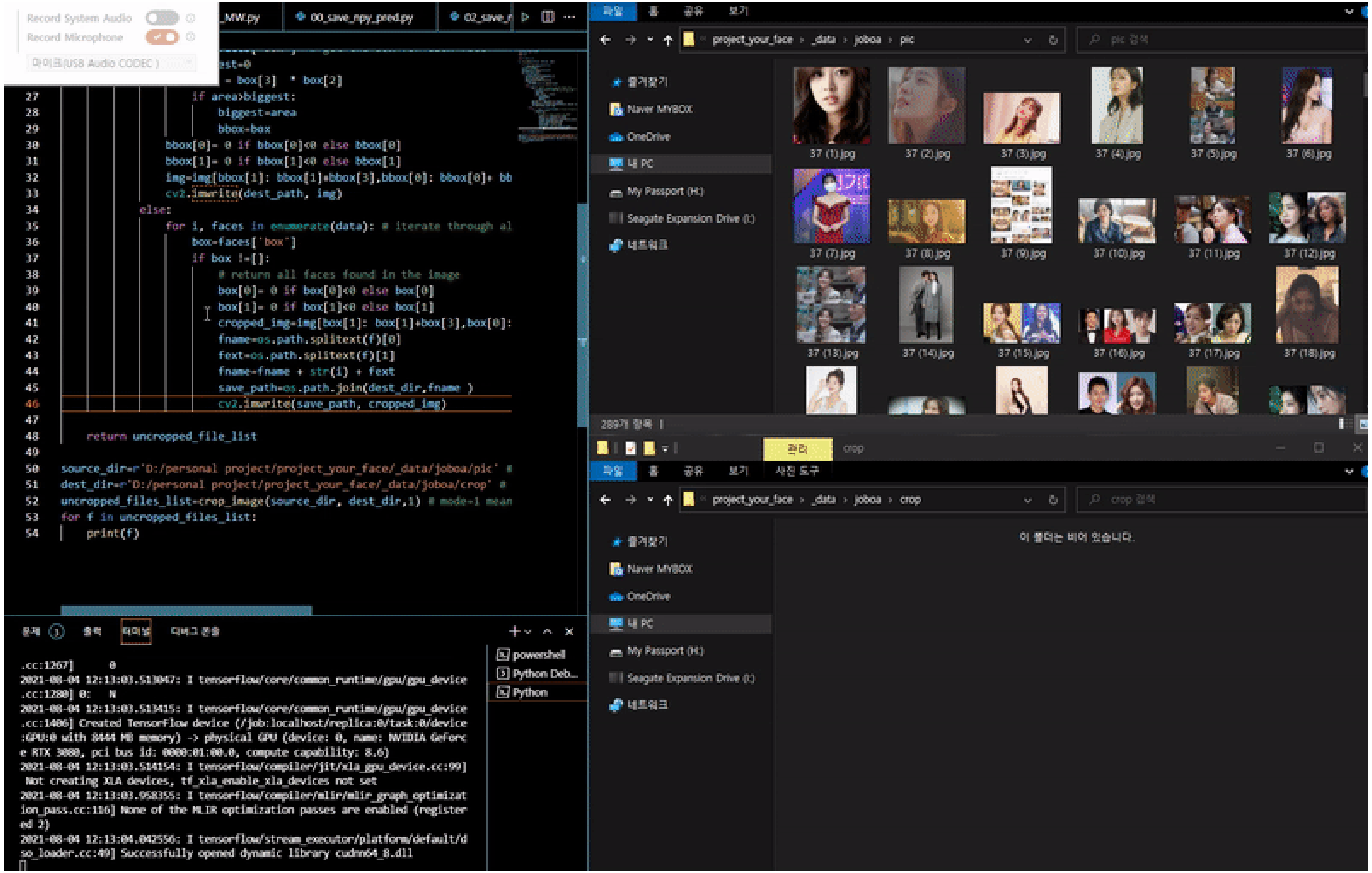
# Crop Face

Use MTCNN

# MTCNN

- **입력 이미지를 다중 resize**
- **각 결과물의 얼굴 존재 여부 판단**
- **얼굴의 랜드마크(눈, 코, 입 등) 기준으로 얼굴 주위의 box 생성**

➢ 생성 좌표 기준 Crop 진행 및 지정 좌표에 이미지 저장



```
                    est=0
                    = box[3]  * box[2]
                    if area>biggest:
                        biggest=area
                        bbox=box
                bbox[0]= 0 if bbox[0]<0 else bbox[0]
                bbox[1]= 0 if bbox[1]<0 else bbox[1]
                img=img[bbox[1]: bbox[1]+bbox[3],bbox[0]: bbox[0]+ bb
                cv2.imwrite(dest_path, img)
            else:
                for i, faces in enumerate(data): # iterate through al
                    box=faces['box']
                    if box !=[]:
                        # return all faces found in the image
                        box[0]= 0 if box[0]<0 else box[0]
                        box[1]= 0 if box[1]<0 else box[1]
                        cropped_img=img[box[1]: box[1]+box[3],box[0]:
                        fname=os.path.splitext(f)[0]
                        fext=os.path.splitext(f)[1]
                        fname=fname + str(i) + fext
                        save_path=os.path.join(dest_dir,fname )
                        cv2.imwrite(save_path, cropped_img)

    return uncropped_file_list

source_dir=r'D:/personal project/project_your_face/_data/joboa/pic' #
dest_dir=r'D:/personal project/project_your_face/_data/joboa/crop' #
uncropped_files_list=crop_image(source_dir, dest_dir,1) # mode=1 mean
for f in uncropped_files_list:
    print(f)
```

```
.cc:1267]      0
2021-08-04 12:13:03.513047: I tensorflow/core/common_runtime/gpu/gpu_device
.cc:1280] 0:   N
2021-08-04 12:13:03.513415: I tensorflow/core/common_runtime/gpu/gpu_device
.cc:1406] Created TensorFlow device (/job:localhost/replica:0/task:0/device
:GPU:0 with 8444 MB memory) -> physical GPU (device: 0, name: NVIDIA GeForc
e RTX 3080, pci bus id: 0000:01:00.0, compute capability: 8.6)
2021-08-04 12:13:03.514154: I tensorflow/compiler/jit/xla_gpu_device.cc:99]
 Not creating XLA devices, tf_xla_enable_xla_devices not set
2021-08-04 12:13:03.958355: I tensorflow/compiler/mlir/mlir_graph_optimizat
ion_pass.cc:116] None of the MLIR optimization passes are enabled (register
ed 2)
2021-08-04 12:13:04.042556: I tensorflow/stream_executor/platform/default/d
so_loader.cc:49] Successfully opened dynamic library cudnn64_8.dll
```

03

Dataset

# Classification by feature group

Base Dataset (10,800, 150, 150, 3)

- 45 person , 240 pics each
- Total : 10,800
- Category : 3 (each 10,800)
  - ➢ Gender
  - ➢ Job Predict
  - ➢ Face Type
- Flow : 8600 * 3 (each category)

Classified Dataset (58,200, 150, 150, 3)

◆ Gender : Women / Men

- ➢ Pics num : train 17,240, test 2,160
- ➢ Class : 2 (categorical)

◆ Job : Actor / Singer

- ➢ Pics num : train 17,240, test 2,160
- ➢ Class : 2 (categorical)

◆ Face Type : Dog, Cat, Rabbit, fox, Dinosaur, Frog, Snake, Turtle, Bear, Mouse, Tiger

- ➢ Pics num : train 17,240, test 2,160
- ➢ Class : 11 (categorical)

Model

# Data save in NPY

Image Data Generator

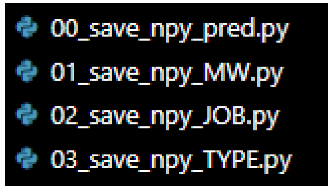
- ➢ **Base Dataset 중 8,600 장의 데이터에 대해 증폭 시행**

Save NPY

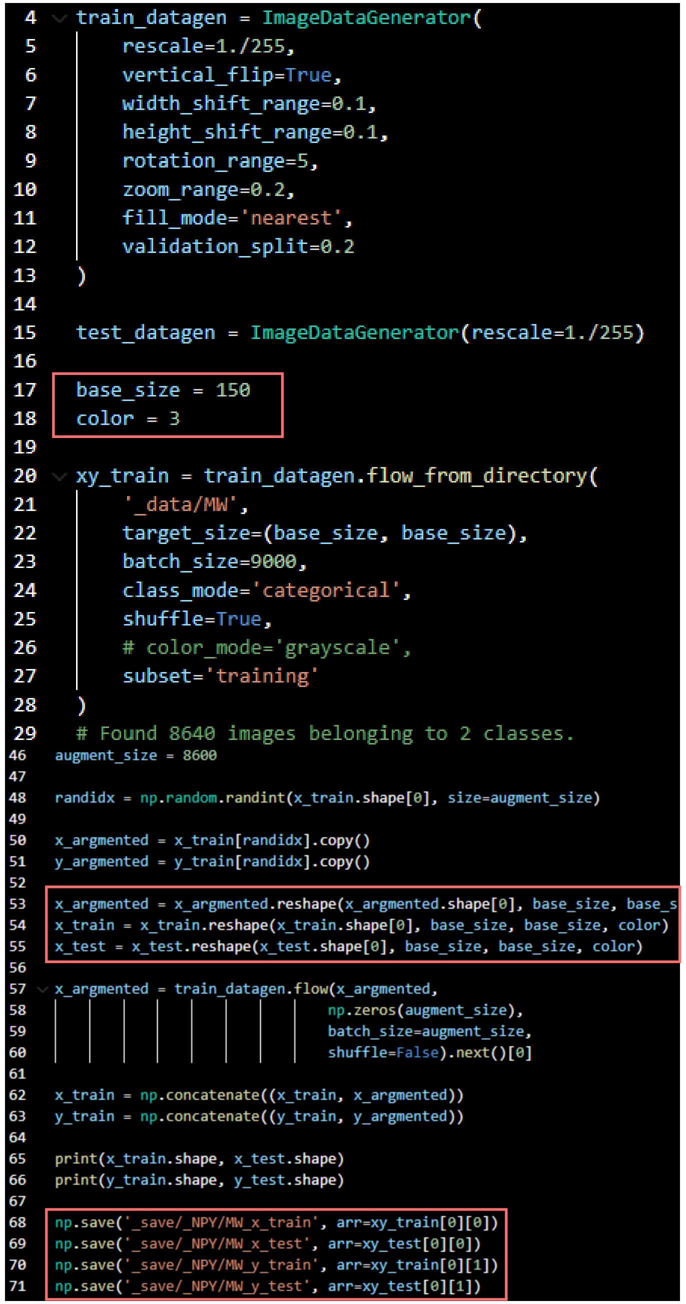
데이터의 증폭 및 flow_from_directory 를 활용해
(150, 150, 3) 사이즈로 정제

정제된 데이터를 각 x, y / train, test로 분리하여
NPY file로 저장

- ➢ 3 Category each, Predict Data 대상 총 네 번 진행



```
00_save_npy_pred.py
01_save_npy_MW.py
02_save_npy_JOB.py
03_save_npy_TYPE.py
```



```
train_datagen = ImageDataGenerator(
    rescale=1./255,
    vertical_flip=True,
    width_shift_range=0.1,
    height_shift_range=0.1,
    rotation_range=5,
    zoom_range=0.2,
    fill_mode='nearest',
    validation_split=0.2
)

test_datagen = ImageDataGenerator(rescale=1./255)

base_size = 150
color = 3

xy_train = train_datagen.flow_from_directory(
    '_data/MW',
    target_size=(base_size, base_size),
    batch_size=9000,
    class_mode='categorical',
    shuffle=True,
    # color_mode='grayscale',
    subset='training'
)
# Found 8640 images belonging to 2 classes.
augment_size = 8600

randidx = np.random.randint(x_train.shape[0], size=augment_size)

x_argmented = x_train[randidx].copy()
y_argmented = y_train[randidx].copy()

x_argmented = x_argmented.reshape(x_argmented.shape[0], base_size, base_s
x_train = x_train.reshape(x_train.shape[0], base_size, base_size, color)
x_test = x_test.reshape(x_test.shape[0], base_size, base_size, color)

x_argmented = train_datagen.flow(x_argmented,
                                 np.zeros(augment_size),
                                 batch_size=augment_size,
                                 shuffle=False).next()[0]

x_train = np.concatenate((x_train, x_argmented))
y_train = np.concatenate((y_train, y_argmented))

print(x_train.shape, x_test.shape)
print(y_train.shape, y_test.shape)

np.save('_save/_NPY/MW_x_train', arr=xy_train[0][0])
np.save('_save/_NPY/MW_x_test', arr=xy_test[0][0])
np.save('_save/_NPY/MW_y_train', arr=xy_train[0][1])
np.save('_save/_NPY/MW_y_test', arr=xy_test[0][1])
```

Model

# Use **Ensemble** with **CNN** layer

✓ Conv2D, Concatenate

Ensemble Model

한 번의 학습으로 다중 라벨 반환 필요

각각의 카테고리가 다른 특성으로 분류되므로
다른 깊이의 레이어 적용 필요하다 판단

앙상블을 활용, 각 세 쌍의 x_train / test, y_train / test 를
Input *3 – concatenate – output *3 으로 학습

➢ **Total params: 8,165,683**

$ CNN 앙상블의 한계

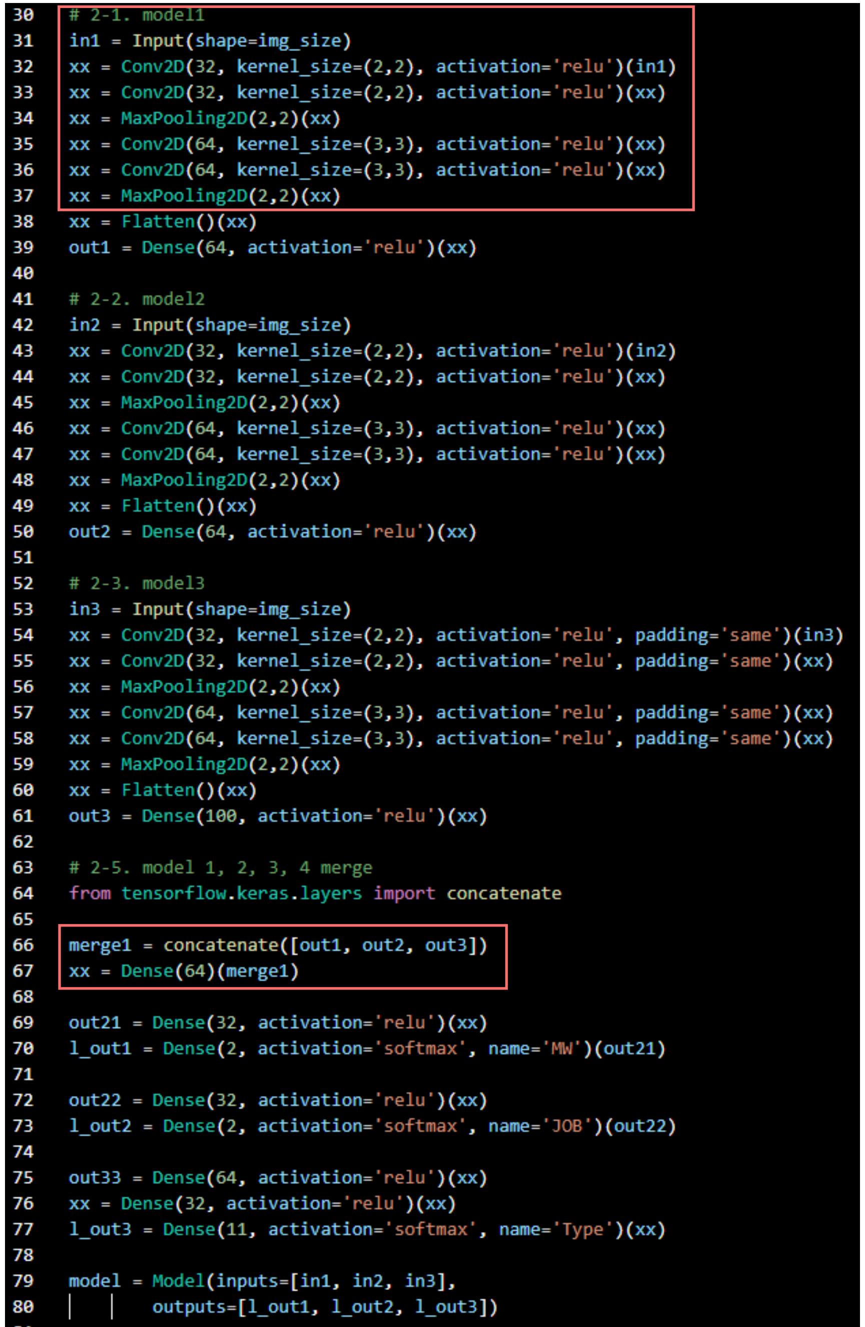
메모리 부족으로 인한 에러

$ CNN 다층 적층 / 고용량 연산 / 이미지 'RGB scale' 불가

➢ **이미지 (100, 100, 1)로 변환하여 학습**

➢ **만족스러운 학습 결과 나오지 않음**

```
MW_acc:        0.7000 JOB_acc:      0.5745 Type_acc:       0.1733
val_MW_acc:    0.6539 val_JOB_acc:  0.5579 val_Type_acc:   0.1597
```



```
# 2-1. model1
in1 = Input(shape=img_size)
xx = Conv2D(32, kernel_size=(2,2), activation='relu')(in1)
xx = Conv2D(32, kernel_size=(2,2), activation='relu')(xx)
xx = MaxPooling2D(2,2)(xx)
xx = Conv2D(64, kernel_size=(3,3), activation='relu')(xx)
xx = Conv2D(64, kernel_size=(3,3), activation='relu')(xx)
xx = MaxPooling2D(2,2)(xx)
xx = Flatten()(xx)
out1 = Dense(64, activation='relu')(xx)

# 2-2. model2
in2 = Input(shape=img_size)
xx = Conv2D(32, kernel_size=(2,2), activation='relu')(in2)
xx = Conv2D(32, kernel_size=(2,2), activation='relu')(xx)
xx = MaxPooling2D(2,2)(xx)
xx = Conv2D(64, kernel_size=(3,3), activation='relu')(xx)
xx = Conv2D(64, kernel_size=(3,3), activation='relu')(xx)
xx = MaxPooling2D(2,2)(xx)
xx = Flatten()(xx)
out2 = Dense(64, activation='relu')(xx)

# 2-3. model3
in3 = Input(shape=img_size)
xx = Conv2D(32, kernel_size=(2,2), activation='relu', padding='same')(in3)
xx = Conv2D(32, kernel_size=(2,2), activation='relu', padding='same')(xx)
xx = MaxPooling2D(2,2)(xx)
xx = Conv2D(64, kernel_size=(3,3), activation='relu', padding='same')(xx)
xx = Conv2D(64, kernel_size=(3,3), activation='relu', padding='same')(xx)
xx = MaxPooling2D(2,2)(xx)
xx = Flatten()(xx)
out3 = Dense(100, activation='relu')(xx)

# 2-5. model 1, 2, 3, 4 merge
from tensorflow.keras.layers import concatenate

merge1 = concatenate([out1, out2, out3])
xx = Dense(64)(merge1)

out21 = Dense(32, activation='relu')(xx)
l_out1 = Dense(2, activation='softmax', name='MW')(out21)

out22 = Dense(32, activation='relu')(xx)
l_out2 = Dense(2, activation='softmax', name='JOB')(out22)

out33 = Dense(64, activation='relu')(xx)
xx = Dense(32, activation='relu')(xx)
l_out3 = Dense(11, activation='softmax', name='Type')(xx)

model = Model(inputs=[in1, in2, in3],
        outputs=[l_out1, l_out2, l_out3])
```

04

Model

# Use Transfer Learning layer with Ensemble

- ✓ ResNet50V2, VGG19, InceptionResNetV2
- ✓ Model Check Point, Early Stopping

Transfer Learning

세 Input layer에 각 동일한 전이학습 모델 사용 시
지정된 레이어 명이 중복되어 충돌 발생

➢ **각각 다른 전이학습 모델 사용**

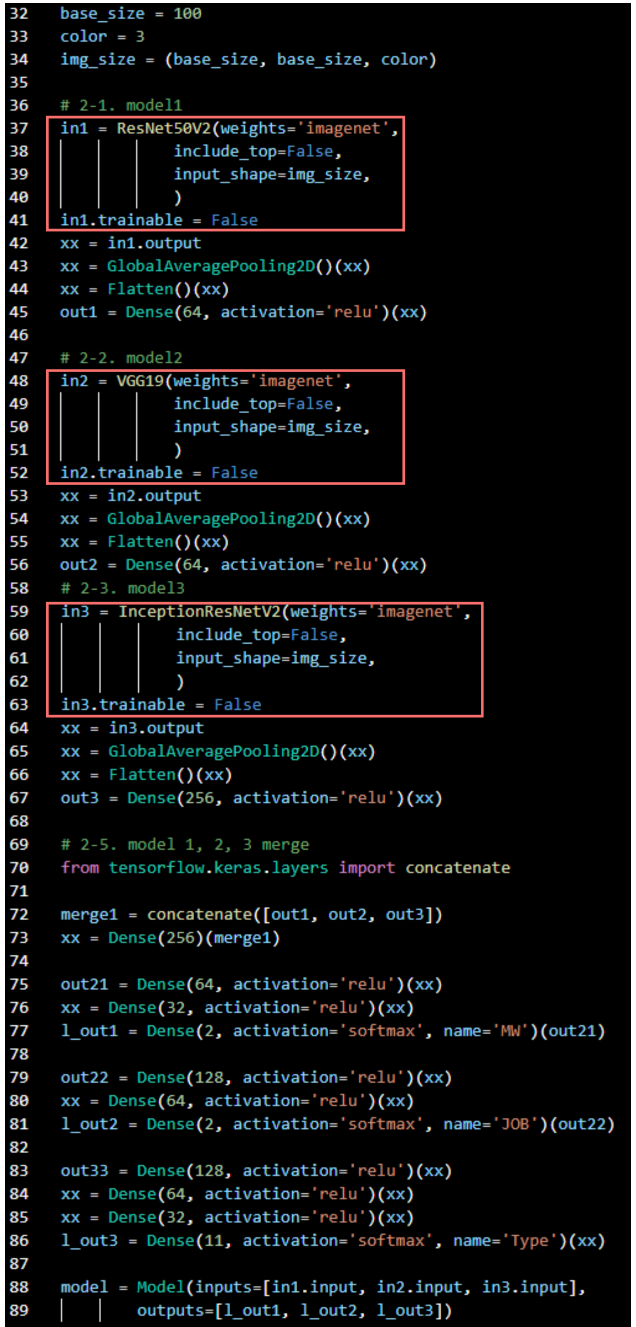
전이학습(특성 분류) – DNN Layer(Label 분류)
형태의 Fine Tuning 진행

➢ **학습 시간 축소 / 고용량 연산 가능 (150, 150, 3)**

Save Model Check Point

➢ **학습한 최적의 weight 저장**

```
MW_acc:     0.9311 33%  JOB_acc:     0.6896 20%  Type_acc:     0.3416  97%
val_MW_acc: 0.8084 24%  val_JOB_acc: 0.5174 -7%  val_Type_acc: 0.2697  69%
```



```
base_size = 100
color = 3
img_size = (base_size, base_size, color)

# 2-1. model1
in1 = ResNet50V2(weights='imagenet',
                 include_top=False,
                 input_shape=img_size,
                 )
in1.trainable = False
xx = in1.output
xx = GlobalAveragePooling2D()(xx)
xx = Flatten()(xx)
out1 = Dense(64, activation='relu')(xx)

# 2-2. model2
in2 = VGG19(weights='imagenet',
                 include_top=False,
                 input_shape=img_size,
                 )
in2.trainable = False
xx = in2.output
xx = GlobalAveragePooling2D()(xx)
xx = Flatten()(xx)
out2 = Dense(64, activation='relu')(xx)
# 2-3. model3
in3 = InceptionResNetV2(weights='imagenet',
                 include_top=False,
                 input_shape=img_size,
                 )
in3.trainable = False
xx = in3.output
xx = GlobalAveragePooling2D()(xx)
xx = Flatten()(xx)
out3 = Dense(256, activation='relu')(xx)

# 2-5. model 1, 2, 3 merge
from tensorflow.keras.layers import concatenate

merge1 = concatenate([out1, out2, out3])
xx = Dense(256)(merge1)

out21 = Dense(64, activation='relu')(xx)
xx = Dense(32, activation='relu')(xx)
l_out1 = Dense(2, activation='softmax', name='MW')(out21)

out22 = Dense(128, activation='relu')(xx)
xx = Dense(64, activation='relu')(xx)
l_out2 = Dense(2, activation='softmax', name='JOB')(out22)

out33 = Dense(128, activation='relu')(xx)
xx = Dense(64, activation='relu')(xx)
xx = Dense(32, activation='relu')(xx)
l_out3 = Dense(11, activation='softmax', name='Type')(xx)

model = Model(inputs=[in1.input, in2.input, in3.input],
        outputs=[l_out1, l_out2, l_out3])
```

Model

# Use Saved weight to Predict

✓ Matplotlip, Load_Model

Load Model

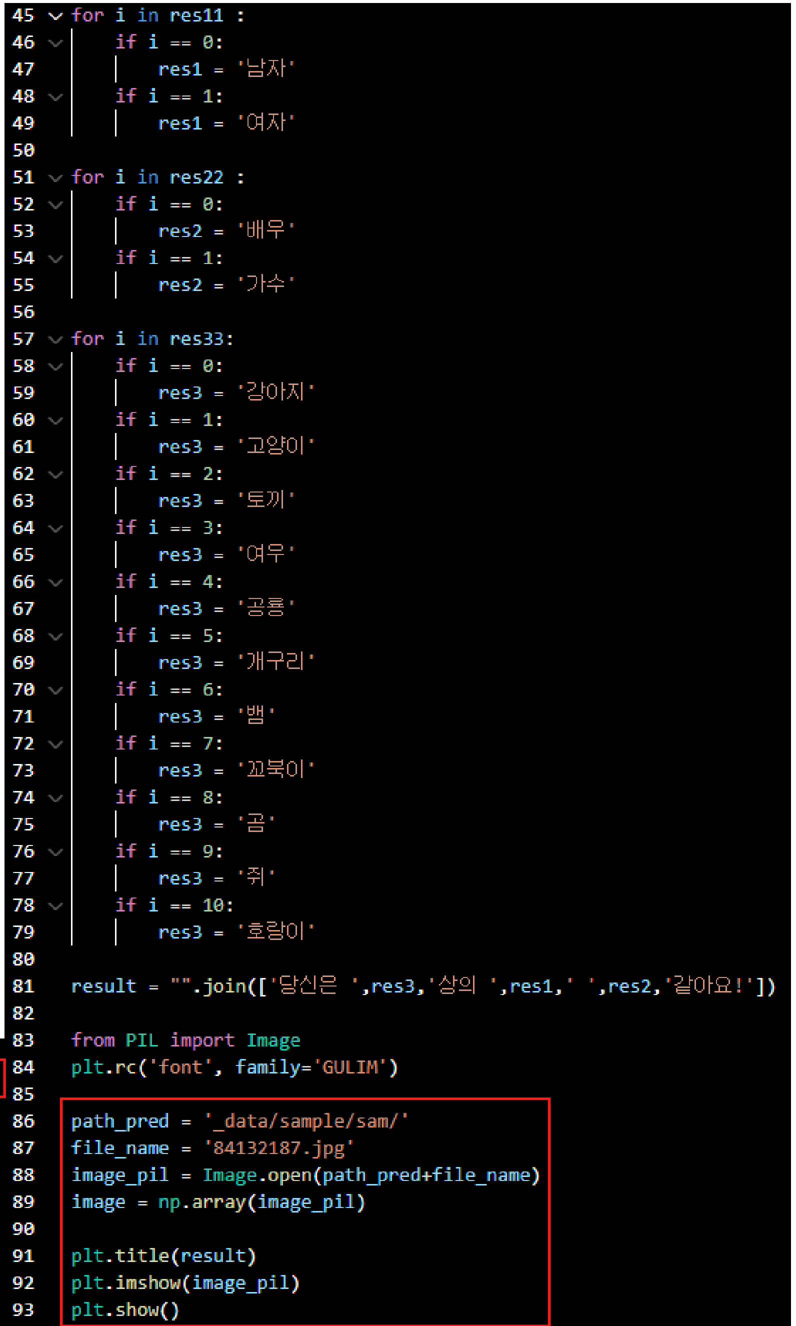
이전 단계에서 학습한 Weight 를 .hdf5 파일 형태로 Load

➢ **학습(Fit) 없이 예측 진행 : 시간 단축, 동일 결과 반환**

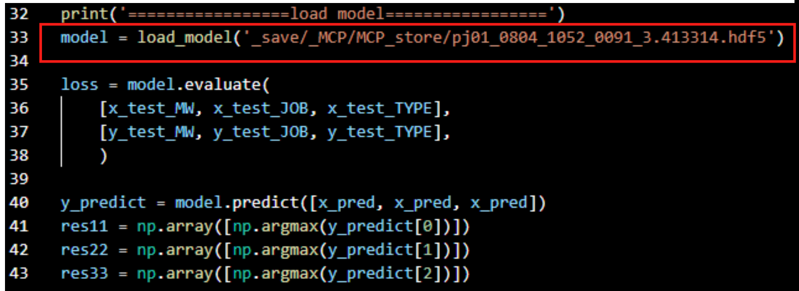
Visualize Result

Argmax, for loop 활용 category 에 결과 할당

➢ **Matplotlib Imshow 활용, 입력 사진 및 결과 동시 반환**



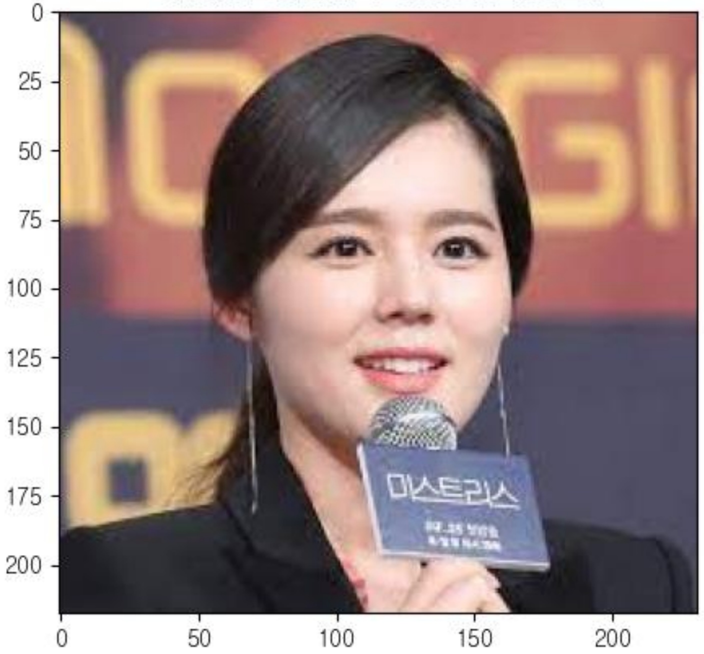
```python
for i in res11 :
    if i == 0:
        res1 = '남자'
    if i == 1:
        res1 = '여자'

for i in res22 :
    if i == 0:
        res2 = '배우'
    if i == 1:
        res2 = '가수'

for i in res33:
    if i == 0:
        res3 = '강아지'
    if i == 1:
        res3 = '고양이'
    if i == 2:
        res3 = '토끼'
    if i == 3:
        res3 = '여우'
    if i == 4:
        res3 = '공룡'
    if i == 5:
        res3 = '개구리'
    if i == 6:
        res3 = '뱀'
    if i == 7:
        res3 = '꼬북이'
    if i == 8:
        res3 = '곰'
    if i == 9:
        res3 = '쥐'
    if i == 10:
        res3 = '호랑이'

result = "".join(['당신은 ',res3,'상의 ',res1,' ',res2,'같아요!'])

from PIL import Image
plt.rc('font', family='GULIM')

path_pred = '_data/sample/sam/'
file_name = '84132187.jpg'
image_pil = Image.open(path_pred+file_name)
image = np.array(image_pil)

plt.title(result)
plt.imshow(image_pil)
plt.show()
```

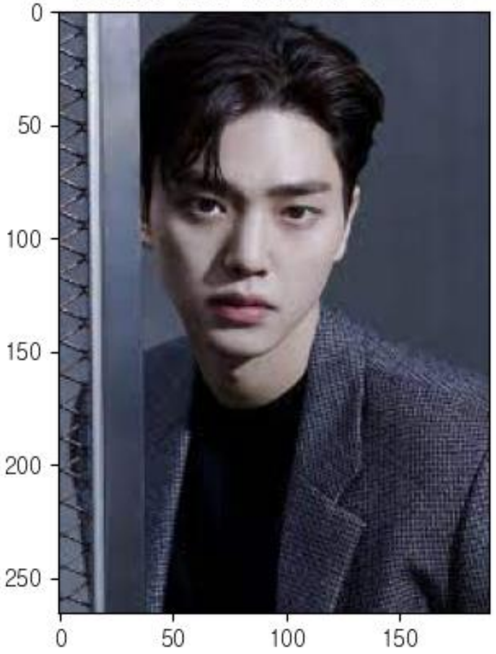


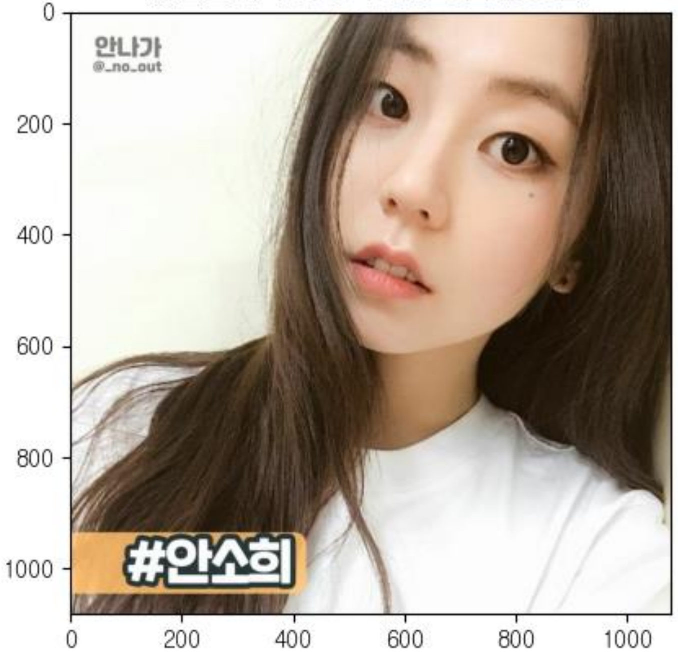
```python
print('=================load model=================')
model = load_model('_save/_MCP/MCP_store/pj01_0804_1052_0091_3.413314.hdf5')

loss = model.evaluate(
    [x_test_MW, x_test_JOB, x_test_TYPE],
    [y_test_MW, y_test_JOB, y_test_TYPE],
    )

y_predict = model.predict([x_pred, x_pred, x_pred])
res11 = np.array([np.argmax(y_predict[0])])
res22 = np.array([np.argmax(y_predict[1])])
res33 = np.array([np.argmax(y_predict[2])])
```

Result

# Find Characteristic In face pic.


당신은 호랑이상의 여자 가수같아요!


당신은 쥐상의 남자 배우같아요!

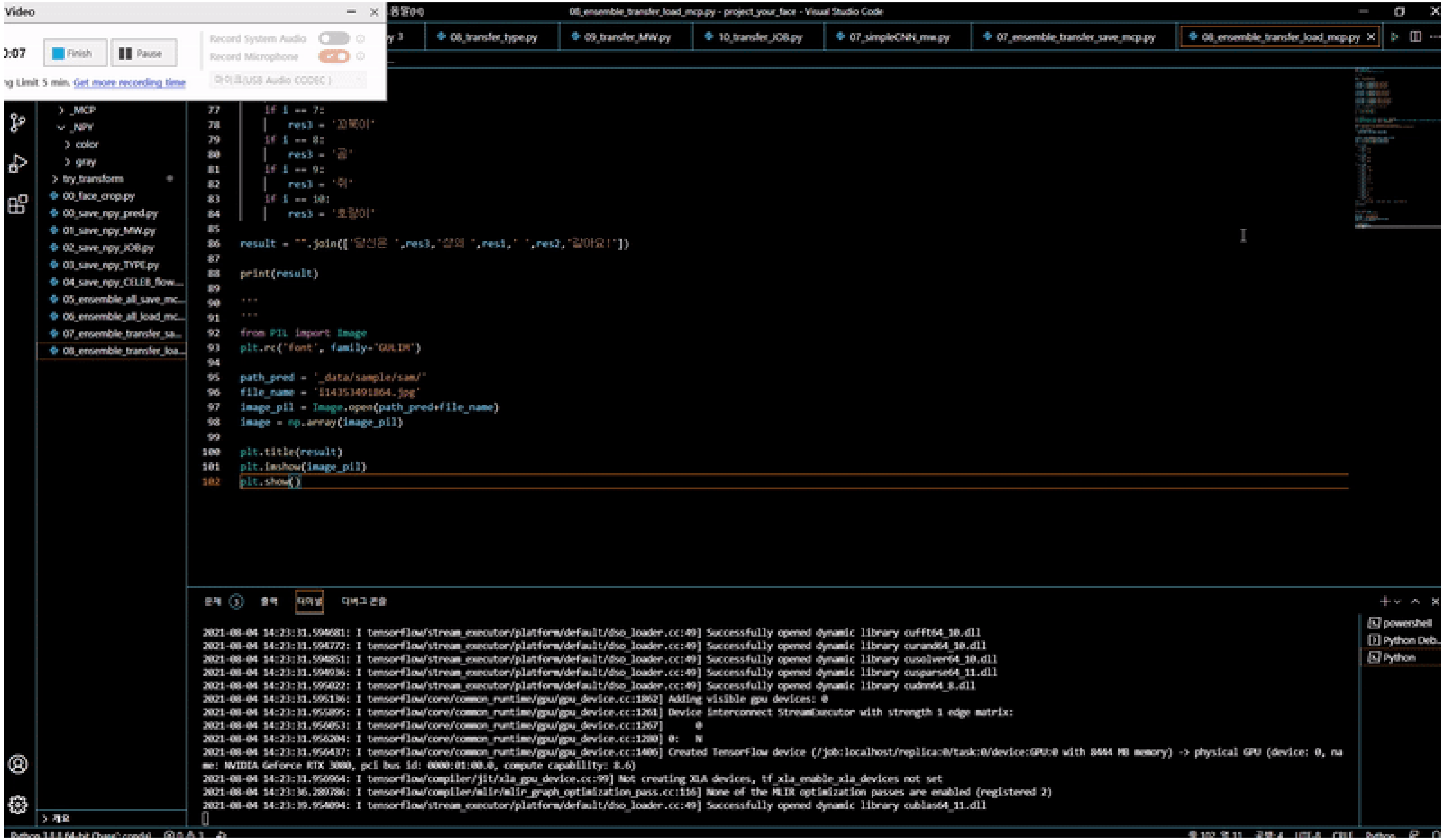

당신은 개구리상의 여자 가수같아요!

05

p r o j e c t .

Result

1



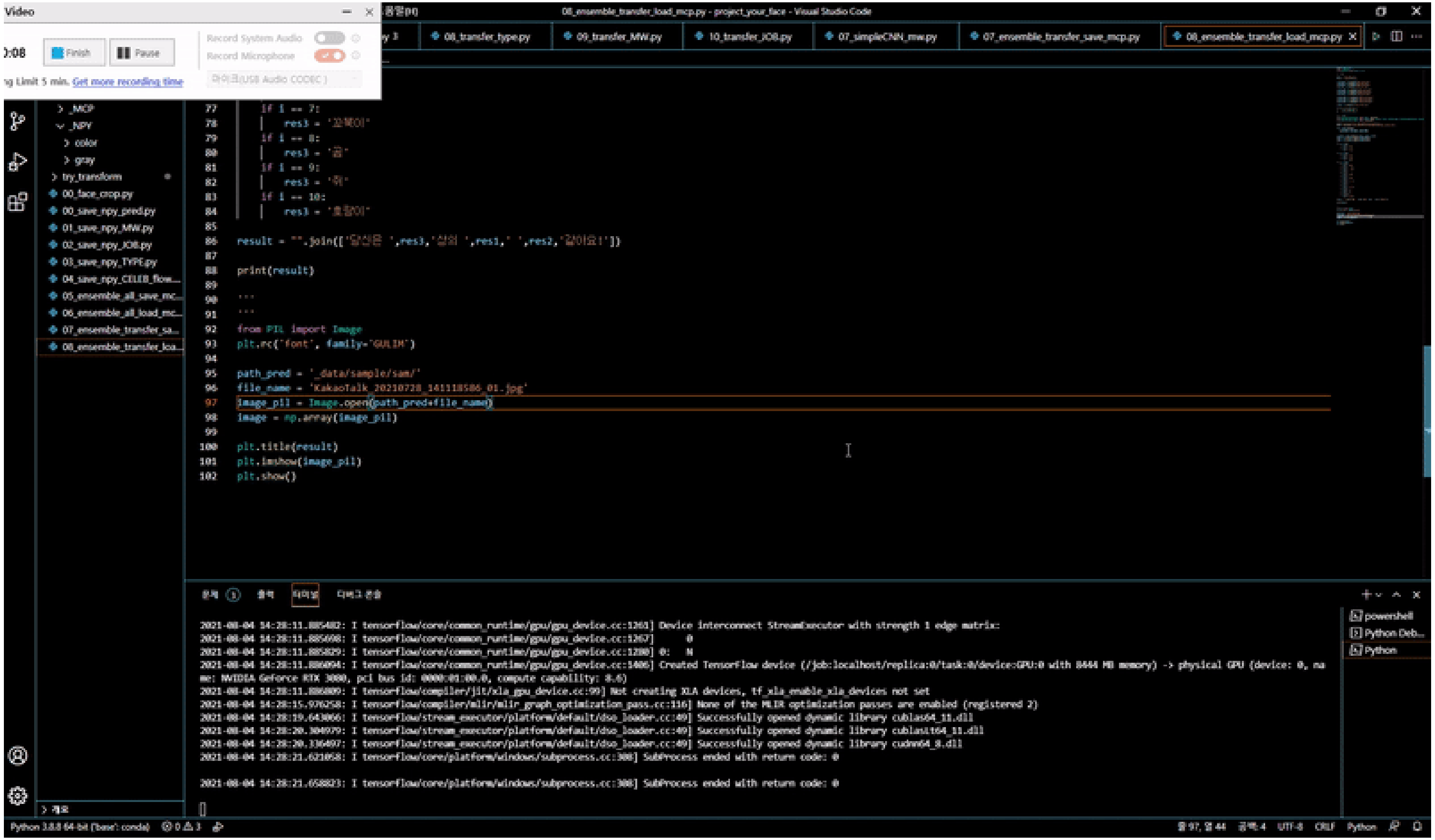
```python
    if i == 7:
        res3 = '꼬북이'
    if i == 8:
        res3 = '곰'
    if i == 9:
        res3 = '쥐'
    if i == 10:
        res3 = '호랑이'

result = "".join(['당신은 ',res3,'상의 ',res1,' ',res2,'같아요!'])

print(result)

'''
'''
from PIL import Image
plt.rc('font', family='GULIM')

path_pred = '_data/sample/sam/'
file_name = '114353491864.jpg'
image_pil = Image.open(path_pred+file_name)
image = np.array(image_pil)

plt.title(result)
plt.imshow(image_pil)
plt.show()
```

```
2021-08-04 14:23:31.594681: I tensorflow/stream_executor/platform/default/dso_loader.cc:49] Successfully opened dynamic library cufft64_10.dll
2021-08-04 14:23:31.594772: I tensorflow/stream_executor/platform/default/dso_loader.cc:49] Successfully opened dynamic library curand64_10.dll
2021-08-04 14:23:31.594851: I tensorflow/stream_executor/platform/default/dso_loader.cc:49] Successfully opened dynamic library cusolver64_10.dll
2021-08-04 14:23:31.594936: I tensorflow/stream_executor/platform/default/dso_loader.cc:49] Successfully opened dynamic library cusparse64_11.dll
2021-08-04 14:23:31.595022: I tensorflow/stream_executor/platform/default/dso_loader.cc:49] Successfully opened dynamic library cudnn64_8.dll
2021-08-04 14:23:31.595136: I tensorflow/core/common_runtime/gpu/gpu_device.cc:1862] Adding visible gpu devices: 0
2021-08-04 14:23:31.955895: I tensorflow/core/common_runtime/gpu/gpu_device.cc:1261] Device interconnect StreamExecutor with strength 1 edge matrix:
2021-08-04 14:23:31.956053: I tensorflow/core/common_runtime/gpu/gpu_device.cc:1267]      0
2021-08-04 14:23:31.956204: I tensorflow/core/common_runtime/gpu/gpu_device.cc:1280] 0:   N
2021-08-04 14:23:31.956437: I tensorflow/core/common_runtime/gpu/gpu_device.cc:1406] Created TensorFlow device (/job:localhost/replica:0/task:0/device:GPU:0 with 8444 MB memory) -> physical GPU (device: 0, na
me: NVIDIA GeForce RTX 3080, pci bus id: 0000:01:00.0, compute capability: 8.6)
2021-08-04 14:23:31.956964: I tensorflow/compiler/jit/xla_gpu_device.cc:99] Not creating XLA devices, tf_xla_enable_xla_devices not set
2021-08-04 14:23:36.289786: I tensorflow/compiler/mlir/mlir_graph_optimization_pass.cc:116] None of the MLIR optimization passes are enabled (registered 2)
2021-08-04 14:23:39.954094: I tensorflow/stream_executor/platform/default/dso_loader.cc:49] Successfully opened dynamic library cublas64_11.dll
```

<Model Check Point 를 불러와 사진 예측 시행 # 1 >

Result

2



```
        if i == 7:
            res3 = '꼬북이'
        if i == 8:
            res3 = '곰'
        if i == 9:
            res3 = '쥐'
        if i == 10:
            res3 = '호랑이'

result = "".join(['당신은 ',res3,'상의 ',res1,' ',res2,'같아요!'])

print(result)

'''
'''
from PIL import Image
plt.rc('font', family='GULIM')

path_pred = '_data/sample/sam/'
file_name = 'KakaoTalk_20210728_141118586_01.jpg'
image_pil = Image.open(path_pred+file_name)
image = np.array(image_pil)

plt.title(result)
plt.imshow(image_pil)
plt.show()
```

```
2021-08-04 14:28:11.885482: I tensorflow/core/common_runtime/gpu/gpu_device.cc:1261] Device interconnect StreamExecutor with strength 1 edge matrix:
2021-08-04 14:28:11.885698: I tensorflow/core/common_runtime/gpu/gpu_device.cc:1267]      0
2021-08-04 14:28:11.885829: I tensorflow/core/common_runtime/gpu/gpu_device.cc:1280] 0:   N
2021-08-04 14:28:11.886094: I tensorflow/core/common_runtime/gpu/gpu_device.cc:1406] Created TensorFlow device (/job:localhost/replica:0/task:0/device:GPU:0 with 8444 MB memory) -> physical GPU (device: 0, na
me: NVIDIA GeForce RTX 3080, pci bus id: 0000:01:00.0, compute capability: 8.6)
2021-08-04 14:28:11.886809: I tensorflow/compiler/jit/xla_gpu_device.cc:99] Not creating XLA devices, tf_xla_enable_xla_devices not set
2021-08-04 14:28:15.976258: I tensorflow/compiler/mlir/mlir_graph_optimization_pass.cc:116] None of the MLIR optimization passes are enabled (registered 2)
2021-08-04 14:28:19.643066: I tensorflow/stream_executor/platform/default/dso_loader.cc:49] Successfully opened dynamic library cublas64_11.dll
2021-08-04 14:28:20.304979: I tensorflow/stream_executor/platform/default/dso_loader.cc:49] Successfully opened dynamic library cublasLt64_11.dll
2021-08-04 14:28:20.336497: I tensorflow/stream_executor/platform/default/dso_loader.cc:49] Successfully opened dynamic library cudnn64_8.dll
2021-08-04 14:28:21.621058: I tensorflow/core/platform/windows/subprocess.cc:308] SubProcess ended with return code: 0

2021-08-04 14:28:21.658823: I tensorflow/core/platform/windows/subprocess.cc:308] SubProcess ended with return code: 0
```

<Model Check Point 를 불러와 사진 예측 시행 # 2 >

End

# Conclusion

- 데이터 수집 및 전처리에 많은 시간 소요
- MTCNN, Transfer Learning 에 대한 불충분한 학습
- 고용량 데이터 연산에서 Deep Learning의 한계 경험

06

End

QnA

Appendix

# Source

$ MTCNN

- https://hwangtoemat.github.io/paper-review/2020-03-28-MTCNN-%EB%82%B4%EC%9A%A9/

$ Crop face using MTCNN

- http://5.9.10.113/65105644/how-to-face-extraction-from-images-in-a-folder-with-mtcnn-in-python

$ Transfer Learning

- https://bskyvision.com/1082

- https://ichi.pro/ko/jeon-i-hagseub-eul-sayonghan-eolgul-insig-168802726462525