Is that a man or woman?

CNN을 활용한 분석

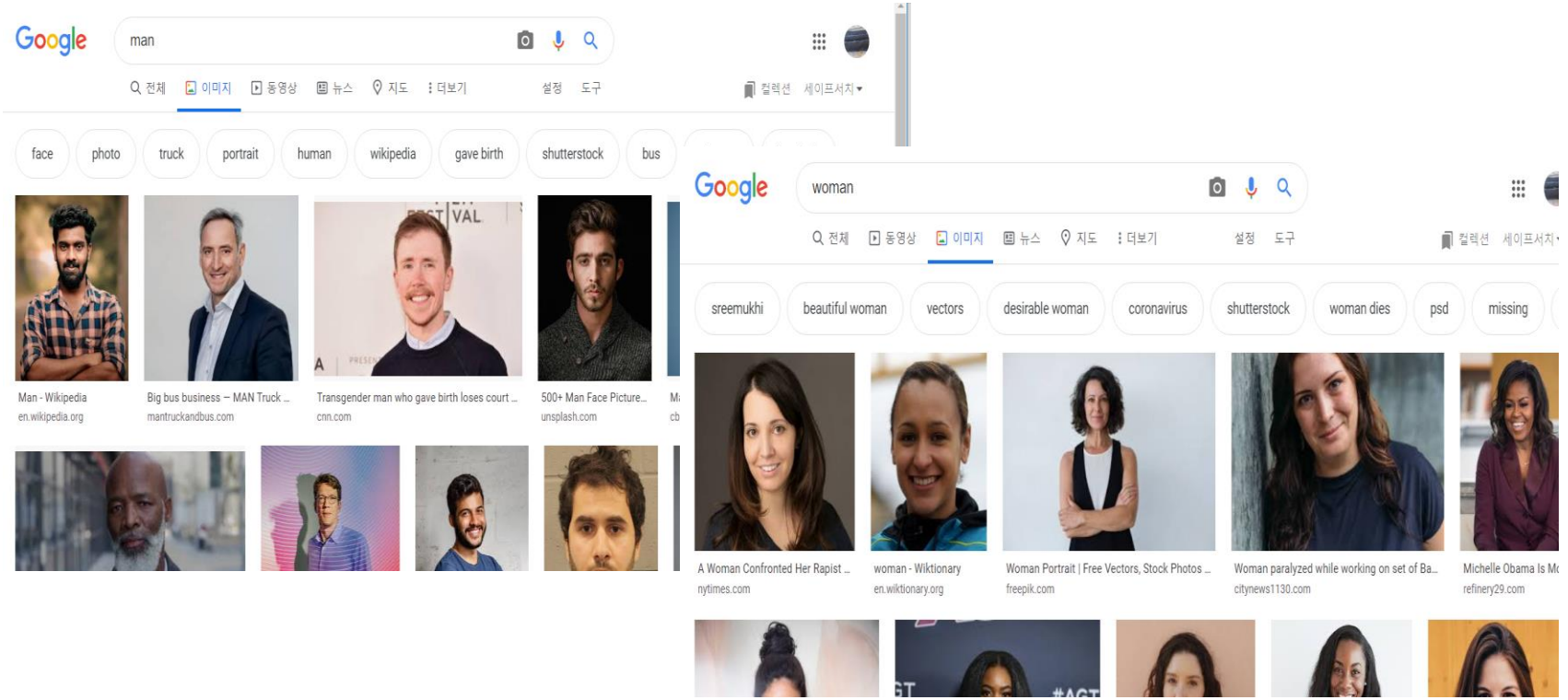
조우현

INDEX

- 데이터 수집과정
- 데이터 변환하기
- 모델 훈련시키기
- 데이터 적용해보기
- 모델 분석하기

PART 1

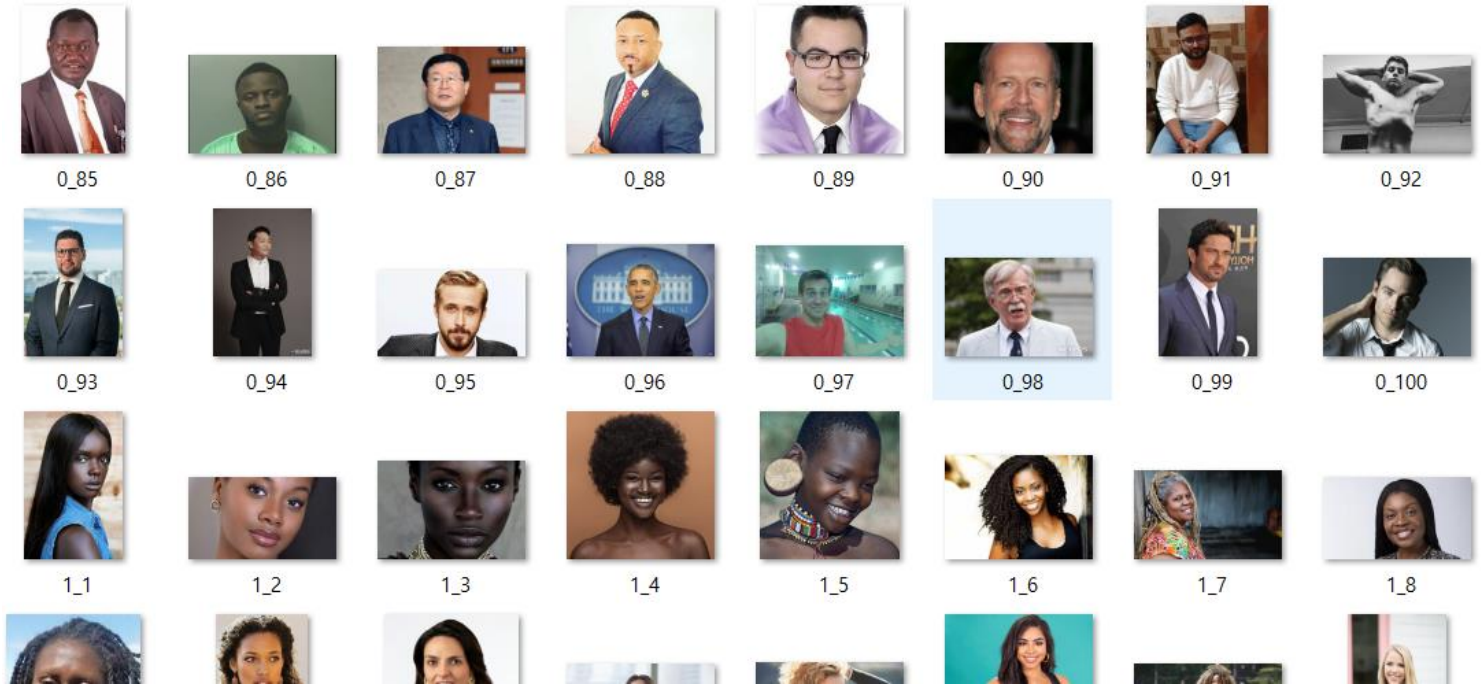
데이터 수집과정



모델을 학습시킬 데이터를 수집하기 위해 google을 이용하여 다양한 인종의 남자사진100장 여자사진 100장을 수집하여 1~100사이의 이름으로 각각 다른 폴더에 저장하였다.

PART 1

데이터 수집과정



수집한 남자사진100장 여자사진 100장은 남자일 경우 앞에 0_를 여자일 경우 앞에 1_를 붙였고, 각 데이터들은 크기를 같게 해주기 위해 100X100의 크기로 변환하였다.

PART 1

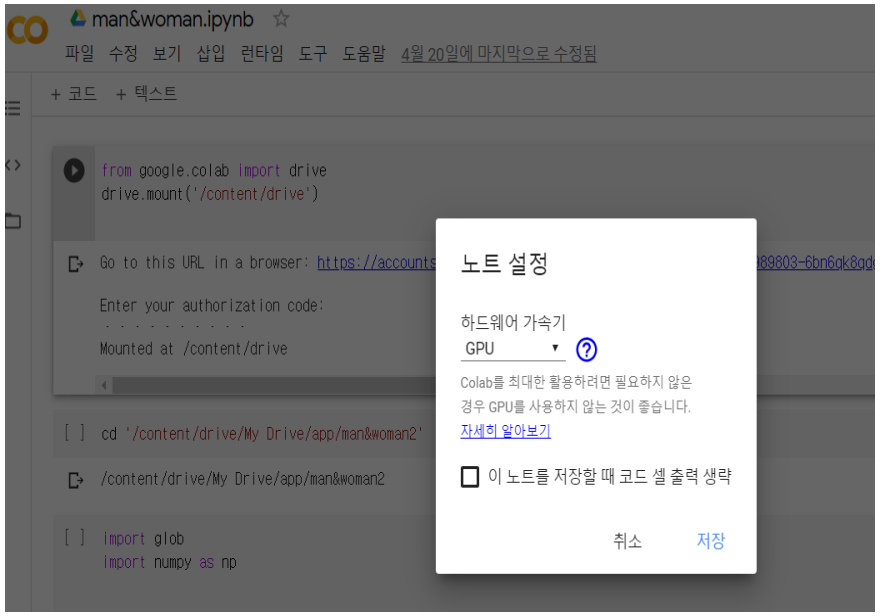
데이터 수집과정

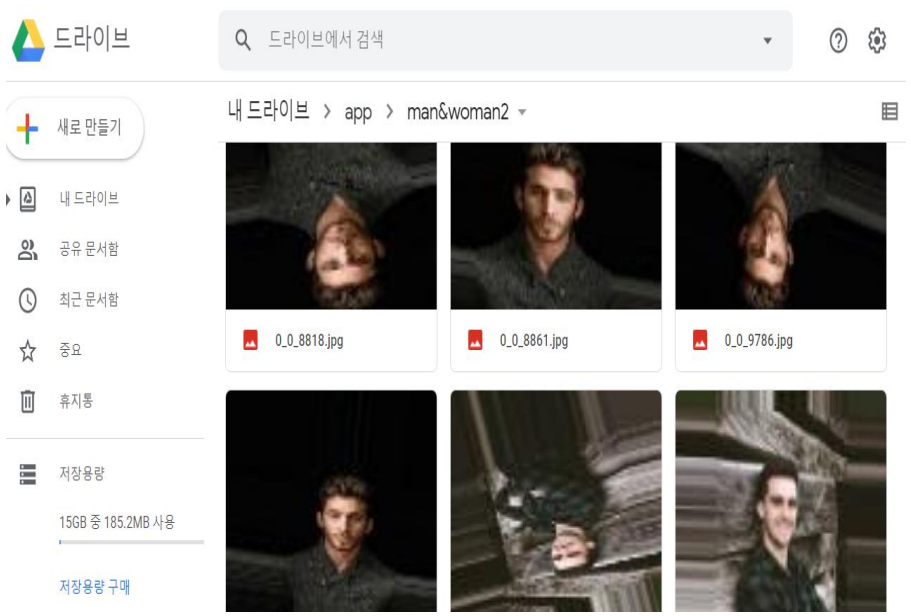
In [23]:
```
from keras.preprocessing.image import ImageDataGenerator
datagen = ImageDataGenerator(
                              rescale=1./255,
                              zoom_range = [0.8,2.0],
                              shear_range=0.5,
                              rotation_range=15,
                              horizontal_flip=True,
                              vertical_flip=True,
                              height_shift_range=0.1,
                              width_shift_range=0.1,
                              fill_mode = 'nearest')


for i in range(images.shape[0]):
    i=0
    for batch in datagen.flow(images,batch_size=1,save_to_dir='D:₩₩man&women₩₩imgen_woman'
                              ,save_prefix='1',save_format='jpg'):
        i += 1
        if i >10:
            break
```

모델을 학습시키기에 200개의 표본은 너무 적다고 판단하여, 사진에 변화를 주고 저장하게 도와주는 ImageDataGenerator를 이용해 각 파일을 변형시켜 추가적인 파일을 생성하여 저장하였고, 그 과정에서 남자는0,여자는1의 인덱스를 파일명 앞에 추가해주었다.

PART 1

데이터 수집과정





google colab의 gpu를 사용하여 모델을 훈련하기 위해
데이터들을 google drive로 옮겼다.

PART 2

데이터 변환하기

```
[1]  from google.colab import drive
     drive.mount('/content/drive')
```

```
Go to this URL in a browser: https://accounts.google.

Enter your authorization code:
· · · · · · · · · · · ·
Mounted at /content/drive
```

```
[2]  cd '/content/drive/My Drive/app/man&woman2'
```

```
/content/drive/My Drive/app/man&woman2
```

```
[ ]  import glob
     import numpy as np


     file_paths = glob.glob('*.jpg')

     file_paths[0]
```

```
'1_37.jpg'
```

모델에 적합할 데이터를 만들기 위해 google drive의 저장폴더에 접속하였고,
이미지파일들의 이름을 불러와 저장하였다.

PART 2 데이터 변환하기

```
import matplotlib.pyplot as plt
import cv2

images = [cv2.imread(path, cv2.IMREAD_COLOR) for path in file_paths]
```


```
images = [cv2.cvtColor(path,cv2.COLOR_BGR2RGB) for path in images]
```

Cv2.imread를 이용하여 이미지 데이터를 컬러로 불러서 BGR형태로 저장하였다.

추가적으로 cvtColor를 이용하여 이미지를 RGB형태로 바꾸어 주었다.

많은 양의 데이터로 인해 불러오는데 많은 시간이 소요되었다.

PART 2 데이터 변환하기

```
[ ]  images = np.asarray(images)

[ ]  images.shape

     (3298, 100, 100, 3)

[ ]
     image_size = np.asarray([images.shape[1], images.shape[2], images.shape[3]])

[ ]  images = images/255
     images.shape

     (3298, 100, 100, 3)
```

```
[ 71,  72,  77],
[ 69,  68,  74],
[ 67,  66,  72]],

...,

[[ 80,  73,  57],
 [ 75,  68,  50],
 [ 62,  52,  40],
 ...,
 [ 81,  67,  58],
 [ 89,  75,  66],
 [ 87,  73,  64]],

[[129, 119, 110],
 [109,  99,  89],
 [ 89,  76,  67],
```

```
[18]

[[0.38431373, 0.4627451 , 0.49803922],
 [0.39215686, 0.47058824, 0.49803922],
 [0.4       , 0.47843137, 0.50588235],
 ...,
 [0.27843137, 0.28235294, 0.30196078],
 [0.27058824, 0.26666667, 0.29019608],
 [0.2627451 , 0.25882353, 0.28235294]],

...,

[[0.31372549, 0.28627451, 0.22352941],
 [0.29411765, 0.26666667, 0.19607843],
 [0.24313725, 0.20392157, 0.15686275],
 ...,
 [0.31764706, 0.2627451 , 0.22745098],
 [0.34901961, 0.29411765, 0.25882353],
 [0.34117647, 0.28627451, 0.25098039]],
```

Images를 asarray형태로 저장해줬고, 그 형태를 파악하였다.

하나의 이미지파일의 모양을 따로 추출하여 image_size로 저장하였고,

모델에 데이터를 적합 시키기 위해 1~255사이의 수로 되어있는 RGB데이터를

0~1사이의 숫자로 변환하였다.

PART 2

데이터 변환하기

```
[20]  import os.path as path
      n_images = images.shape[0]
      labels = np.zeros(n_images)
      for i in range(n_images):
          filename = path.basename(file_paths[i])[0]
          labels[i] = int(filename[0])
```

```
[ ]  TRAIN_TEST_SPLIT=0.9
```

```
[ ]  split_index = int(TRAIN_TEST_SPLIT*n_images)
     shuffled_indices = np.random.permutation(n_images)
     train_indices = shuffled_indices[0:split_index]
     test_indices = shuffled_indices[split_index:]
```

```
[ ]  len(train_indices)
```

```
2968
```

np.zeros를 이용하여 파일과 길이가 같은 인자가0인 리스트를 생성하였고
각 파일명의 첫째 인자, 즉 0,과 1을 각 파일에 해당하는 라벨로 만들어주었다.
또한 훈련 때, 테스트 때 사용할 데이터를 전체 데이터의 0.9의 비율로 나누기 위해 랜덤으로 위치를 추출하였다.

PART 2

데이터 변환하기

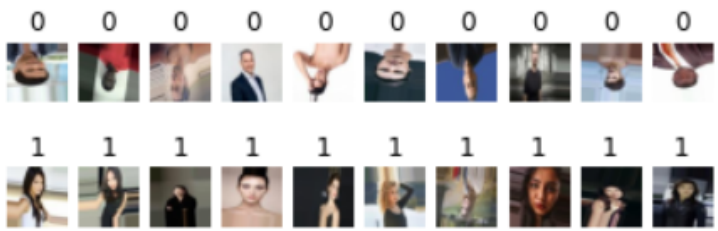
```
[25]  x_train=images[train_indices,:,:]
      y_train=labels[train_indices]
      x_test=images[test_indices,:,:]
      y_test=labels[test_indices]
```

추출한 인덱스를 통해 훈련 데이터와 라벨, 테스트 데이터와 라벨을 설정하였다.

PART 2 데이터 변환하기

```
[31] def visualize_data(woman_images,man_images):
         figure=plt.figure()
         count=0

         for i in range(woman_images.shape[0]):
             count+=1
             figure.add_subplot(1,woman_images.shape[0],count)
             plt.imshow(woman_images[i,:,:])
             plt.axis('off')
             plt.title("1"),

             figure.add_subplot(2,woman_images.shape[0],count)
             plt.imshow(man_images[i,:,:])
             plt.axis('off')
             plt.title("0")

         plt.show()
```

```
[32] N_TO_VISUALIZE=10

     woman_example_indices=(y_train == 1)
     woman_examples = x_train[woman_example_indices,:,:]
     woman_examples = woman_examples[0:N_TO_VISUALIZE,:,:]

     man_example_indices=(y_train == 0)
     man_examples = x_train[man_example_indices,:,:]
     man_examples = man_examples[0:N_TO_VISUALIZE,:,:]

     visualize_data(woman_examples, man_examples)
```



각 데이터가 올바른 형태로 추출되었는지 확인하기 위해 visualize_data 함수를 만들어서 여자와 남자 각각 10개 데이터의 이미지를 확인해보았다.

데이터는 잘 분류되어있었다.

PART 3 모델 훈련시키기

```
[33]  import keras

      from keras.models import Sequential
      from keras.layers import Activation,Dropout,Flatten,Dense,Conv2D,MaxPooling2D
      from keras.callbacks import EarlyStopping, TensorBoard
```

```
def cnn(size,n_layers):
    MIN_NEURONS = 20
    MAX_NEURONS = 120
    KERNEL = (3,3)

    steps = np.floor(MAX_NEURONS/(n_layers +1))
    neurons = np.arange(MIN_NEURONS, MAX_NEURONS, steps)
    neurons = neurons.astype(np.int32)

    model = Sequential()

    for i in range(0,n_layers):
        if i == 0 :
            shape = (size[0],size[1],size[2])
            model.add(Conv2D(neurons[i],KERNEL,input_shape = shape))
        else:
            model.add(Conv2D(neurons[i],KERNEL))
            model.add(Activation('relu'))

    model.add(MaxPooling2D(pool_size = (2, 2)))
    model.add(Flatten())
    model.add(Dense(MAX_NEURONS))
    model.add(Activation('relu'))

    # Add output layer
    model.add(Dense(1,activation='sigmoid'))


    model.summary()

    return model
```

다음과 같은 형태로 모델을 구성하였다.

모델은 20개의 필터로 시작하여 92개의 필터를 갖고 있는 모델을 구성하였고, flatten을 적용하여 최종적으로 120개의 노드를 가진 모델을 구성하였다.

PART 3 모델 훈련시키기

```
model = cnn(size=image_size,n_layers=N_LAYERS)
```

```
Model: "sequential_2"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_5 (Conv2D)            (None, 98, 98, 20)        560
_________________________________________________________________
conv2d_6 (Conv2D)            (None, 96, 96, 44)        7964
_________________________________________________________________
activation_4 (Activation)    (None, 96, 96, 44)        0
_________________________________________________________________
conv2d_7 (Conv2D)            (None, 94, 94, 68)        26996
_________________________________________________________________
activation_5 (Activation)    (None, 94, 94, 68)        0
_________________________________________________________________
conv2d_8 (Conv2D)            (None, 92, 92, 92)        56396
_________________________________________________________________
activation_6 (Activation)    (None, 92, 92, 92)        0
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 46, 46, 92)        0
_________________________________________________________________
flatten_2 (Flatten)          (None, 194672)            0
_________________________________________________________________
dense_2 (Dense)              (None, 120)               23360760
_________________________________________________________________
activation_7 (Activation)    (None, 120)               0
_________________________________________________________________
dense_3 (Dense)              (None, 1)                 121
=================================================================
Total params: 23,452,797
Trainable params: 23,452,797
Non-trainable params: 0
```

모델을 요약하면 다음과 같다.

PART 4 모델 분석하기

```
[49] EPOCHS=150
     BATCH_SIZE=20
```

```
[50] PATIENCE = 10
     early_stopping = EarlyStopping(monitor='loss',min_delta=0,patience=PATIENCE,verbose=0,mode='auto')
```

```
[51] callbacks=[early_stopping]
```

```
[52] model.compile(loss = 'binary_crossentropy',optimizer='adam',metrics = ['accuracy'])
```

반복 수는 150회, batch_size는 20회로 정하였고, early_stopping기법을 사용하여 loss가 10회동안 갱신되지 않으면 훈련을 중단시키기로 결정하였다.

밑은 훈련과정이다.

Loss가 점점 줄어들고

Accuracy가 높아진다.

즉, 훈련이 잘 되고 있다고 볼 수 있다.

```
25s 9ms/step - loss: 0.7077 - accuracy: 0.6388
18s 6ms/step - loss: 0.5315 - accuracy: 0.7456
18s 6ms/step - loss: 0.4733 - accuracy: 0.7773
18s 6ms/step - loss: 0.3883 - accuracy: 0.8282
18s 6ms/step - loss: 0.2727 - accuracy: 0.8831
18s 6ms/step - loss: 0.1792 - accuracy: 0.9279
18s 6ms/step - loss: 0.1516 - accuracy: 0.9471
18s 6ms/step - loss: 0.0870 - accuracy: 0.9673
18s 6ms/step - loss: 0.0744 - accuracy: 0.9737
18s 6ms/step - loss: 0.0194 - accuracy: 0.9936
```

PART 4

모델 분석하기

```
evel = model.evaluate(x_test,y_test,batch_size=1)
print(evel)
```

```
330/330 [==============================] - 2s 5ms/step
[1.2675318878104276, 0.7848485112190247]
```

테스트 데이터를 이용하여 모델을 평가했을 때 약 0.78프로의 확률로 올바르게 판단을 하는 모델이 되었음을 볼 수 있었다.

PART 5 데이터 적용해보기

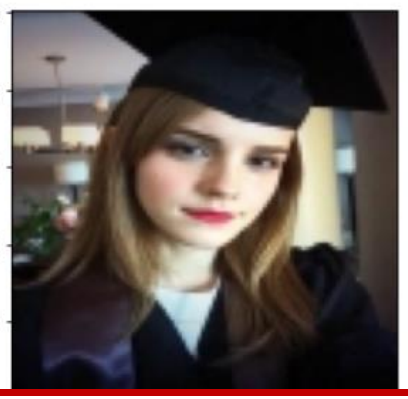
```
[56]  cd '/content/drive/My Drive/app/man&woman/onepick'

      /content/drive/My Drive/app/man&woman/onepick
```

```
[57]  #onepick2
      file_paths = glob.glob('*.jpg')
      file_paths
```

```
[58]  x_pp = cv2.imread(file_paths[6], cv2.IMREAD_COLOR)
      x_pp.shape

      (100, 100, 3)
```

```
[59]   x_pp = cv2.cvtColor(x_pp,cv2.COLOR_BGR2RGB)
```

```
[60]  plt.imshow(x_pp)
      plt.show()
```



다시 구글 드라이브에 접속하여 새로 전처리 해놓은 '엠마왓슨'의 사진데이터를 입력하고, imshow로 올바른 데이터가 들어왔는지 확인했다.

PART 5

데이터 적용해보기

```
[61] x_pp = np.asarray([x_pp])

     x_pp.shape
```

```
(1, 100, 100, 3)
```

```
[62] x_pp = x_pp/255
     x_pp.shape
```

```
(1, 100, 100, 3)
```

```
[63] x_test.shape
```

```
(330, 100, 100, 3)
```

```
[64] p_xt = model.predict(x_pp)
     p_xt = np.round(p_xt)
     res = np.asarray(p_xt).ravel()
     if res[0]==1:
       print('woman')
     else:
       print('man')
```

```
woman
```

기존에 훈련시켜놓은 모델에 엠마왓슨 데이터를 이용하여 테스트 해보았다.

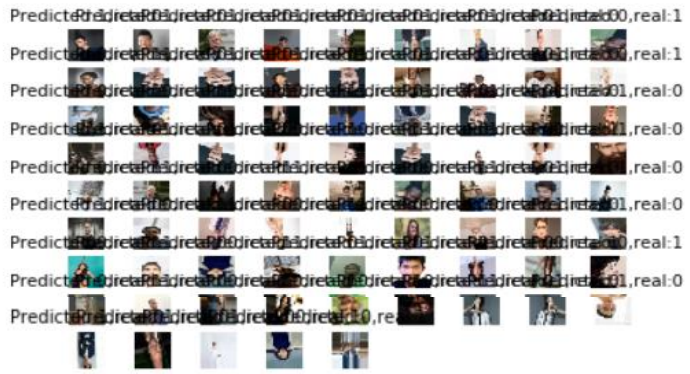
Test결과가 0이라면 남자, 1이라면 여자로 출력하도록 설정하였고

엠마왓슨은 '여자 ' 라고 올바르게 판단한 것을 볼 수 있었다.

PART 5

데이터 적용해보기

```
def visualize_incorrect_labels(x_data,y_real,y_predicted):
  count = 0
  figure = plt.figure()
  incorrect_label_indices = (y_real != y_predicted)
  y_real = y_real[incorrect_label_indices]
  y_predicted = y_predicted[incorrect_label_indices]
  x_data = x_data[incorrect_label_indices,:,:,:]

  maximum_square = np.ceil(np.sqrt(x_data.shape[0]))

  for i in range(x_data.shape[0]):
    count+=1
    figure.add_subplot(maximum_square, maximum_square,count)
    plt.imshow(x_data[i,:,:,:])
    plt.axis('off')
    plt.title("Predicted:"+str(int(y_predicted[i]))+',real:'+str(int(y_real[i])),fontsize=10)

  plt.show()
  print(x_data.shape[0])
visualize_incorrect_labels(x_test,y_test,np.asarray(p_xt).ravel())
```



77

아까 분류하였던 Test데이터를 위의 과정을 통해 모델에 적합한 결과 전체330개의 데이터 중 77개의 데이터에 대해 사실과는 다르게 평가하였다는 것을 알 수 있었고, 훈련데이터를 이용했을 때와 거의 흡사한 정확도(약80%)를 보여주었다.

추후에도 여러 데이터를 넣어서 실험해보았다.
우리 눈으로 쉽게 남자인지 여자인지 알 수 있는 사람들에 대해서는 모델도 올바르게 판단하는 것을 볼 수 있었다.

따라서.
모델이 남자인지 여자인지 평가하는 기준과 사람들이 여자인지 남자인지 평가하는 기준은 크게 다르지 않을 것이라고 생각한다.

THANK YOU