2024 Fall
20190741 김지수

인공지능과 마케팅 코딩과제3

: Emotion Detection data

Project 기초 정보

1) Datasets: https://www.kaggle.com/datasets/msambare/fer2013

- 48*48 픽셀의 얼굴 사진 데이터로, 7가지 감정의 28000장의 학습 데이터와 3500장의 테스트 데이터로 구성됨

- 0=Angry, 1=Disgust, 2=Fear, 3=Happy, 4=Sad, 5=Surprise, 6=Neutral

2) 목표 정확도는 Kaggle에서 순위권에 있는 팀들의 Valid acc로 0.65를 내외

3) 최종적으로 몇 장의 잔망 루피 사진들을 통해서 감정 예측이 성공적으로 수행되는지 확인함

Coding 과제 3

MNIST의 Fashion 혹은 숫자, 혹은 다른 자료들을 통해서 CNN (가져온것도 가능) 해보기

Import library

```
import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)
import os
#for dirname, _, filenames in os.walk('/kaggle/input'):
    #for filename in filenames:
        #print(os.path.join(dirname, filename))
import matplotlib.pyplot as plt
import seaborn as sns
import tensorflow as tf
import keras
from keras.preprocessing import image
from keras.models import Sequential
from keras.layers import Conv2D, MaxPool2D, Flatten,Dense,Dropout,BatchNormalization
from tensorflow.keras.preprocessing.image import ImageDataGenerator
import cv2
from tensorflow.keras.applications import VGG16, InceptionResNetV2
from keras import regularizers
from tensorflow.keras.optimizers import Adam,RMSprop,SGD,Adamax
```

Data Preprocessing

1) Datasets: https://www.kaggle.com/datasets/msambare/fer2013

- 48*48 픽셀의 얼굴 사진 데이터로, 7가지 감정의 28000장의 학습 데이터와 3500장의 테스트 데이터로 구성됨

- 0=Angry, 1=Disgust, 2=Fear, 3=Happy, 4=Sad, 5=Surprise, 6=Neutral

2) 데이터 불러오기

- Kaggle 데이터를 직접 활용하여 해당 디렉토리(경로)를 설정

```
train_dir = "../input/emotion-detection-fer/train"
test_dir = "../input/emotion-detection-fer/test"
```

Preparing the Dataset

데이터 증강

```
img_size = 48 #48*48
train_datagen = ImageDataGenerator(#rotation_range = 180,
            width_shift_range = 0.1, height_shift_range = 0.1,
            horizontal_flip = True, rescale = 1./255,
            #zoom_range = 0.2, validation_split = 0.2)
test_datagen = ImageDataGenerator(rescale = 1./255)
```

1. 입력 이미지 데이터를 255.0으로 나누어 정규화 작업을 수행한다.

2. X_train과 X_test 데이터를 CNN이 처리할 수 있는 형식으로 변환한다.

: 원본 데이터셋을 확대(zoom), 회전(rotate), 이동(shift)하는 등의 방법으로 변형하여 새로운 이미지를 만들고 이를 학습 데이터로 활용하여 모델이 더 다양한 패턴을 학습해 과적합(overfitting)을 방지한다.

위의 코드에서는 가로 방향으로 최대 10%까지 이동시키며, 세로 방향으로 최대 10%까지 이동시키고 수평 뒤집기를 실행하는 방법을 사용한다.

3. Train 데이터와 Test 데이터를 분리하고 학습 데이터의 20%는 검증 데이터로 사용하였다.

Preparing the Dataset

데이터 학습 준비

```
train_generator = train_datagen.flow_from_directory
(directory = train_dir, target_size = (img_size,img_size), batch_size = 64,
  color_mode = "grayscale", class_mode = "categorical", subset = "training")

validation_generator = validation_datagen.flow_from_directory
( directory = test_dir, target_size = (img_size,img_size), batch_size = 64,
  color_mode = "grayscale", class_mode = "categorical", subset = "validation")

test_generator = test_datagen.flow_from_directory
( directory=test_dir, target_size=(img_size, img_size), batch_size=64,
  color_mode="grayscale", class_mode="categorical",
  shuffle=False  # 평가에서는 섞지 않음
)
```

Found 22968 images belonging to 7 classes.

Found 5741 images belonging to 7 classes.

Found 7178 images belonging to 7 classes.

Train the model

모델 생성

```
model= tf.keras.models.Sequential()
model.add(Conv2D(32, kernel_size=(3, 3), padding='same', activation='relu', input_shape=(48, 48,1)))
model.add(Conv2D(64,(3,3), padding='same', activation='relu' ))
model.add(BatchNormalization())
model.add(MaxPool2D(pool_size=(2, 2)))
model.add(Dropout(0.25))

model.add(Conv2D(128,(5,5), padding='same', activation='relu'))
model.add(BatchNormalization())
model.add(MaxPool2D(pool_size=(2, 2)))
model.add(Dropout(0.25))

model.add(Conv2D(512,(3,3), padding='same', activation='relu', kernel_regularizer=regularizers.l2(0.01)))
model.add(BatchNormalization())
model.add(MaxPool2D(pool_size=(2, 2)))
model.add(Dropout(0.25))

model.add(Conv2D(512,(3,3), padding='same', activation='relu', kernel_regularizer=regularizers.l2(0.01)))
model.add(BatchNormalization())
model.add(MaxPool2D(pool_size=(2, 2)))
model.add(Dropout(0.25))

model.add(Flatten())
model.add(Dense(256,activation = 'relu'))
model.add(BatchNormalization())
model.add(Dropout(0.25))

model.add(Dense(512,activation = 'relu'))
model.add(BatchNormalization())
model.add(Dropout(0.25))

model.add(Dense(7, activation='softmax'))
```

Train the model

모델 학습

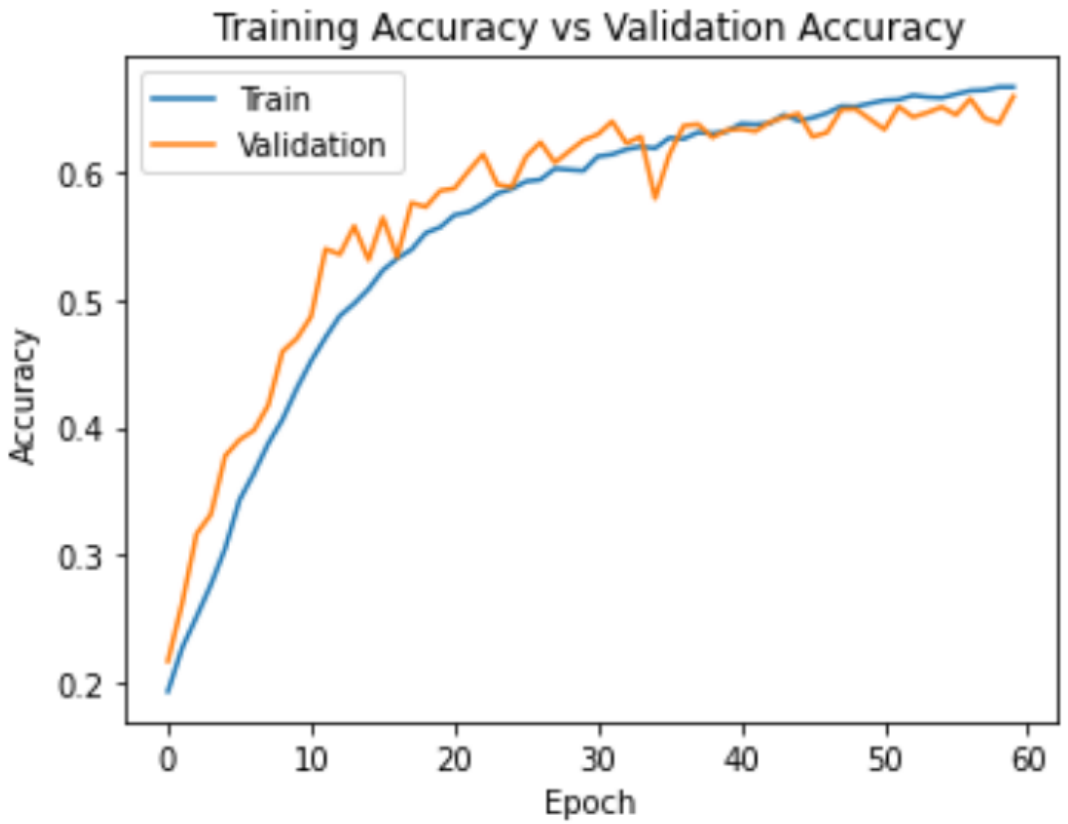
```
model.compile(
    optimizer = Adam(lr=0.0001),
    loss='categorical_crossentropy',
    metrics=['accuracy']
  )

epochs = 60
batch_size = 64

model.summary()
```

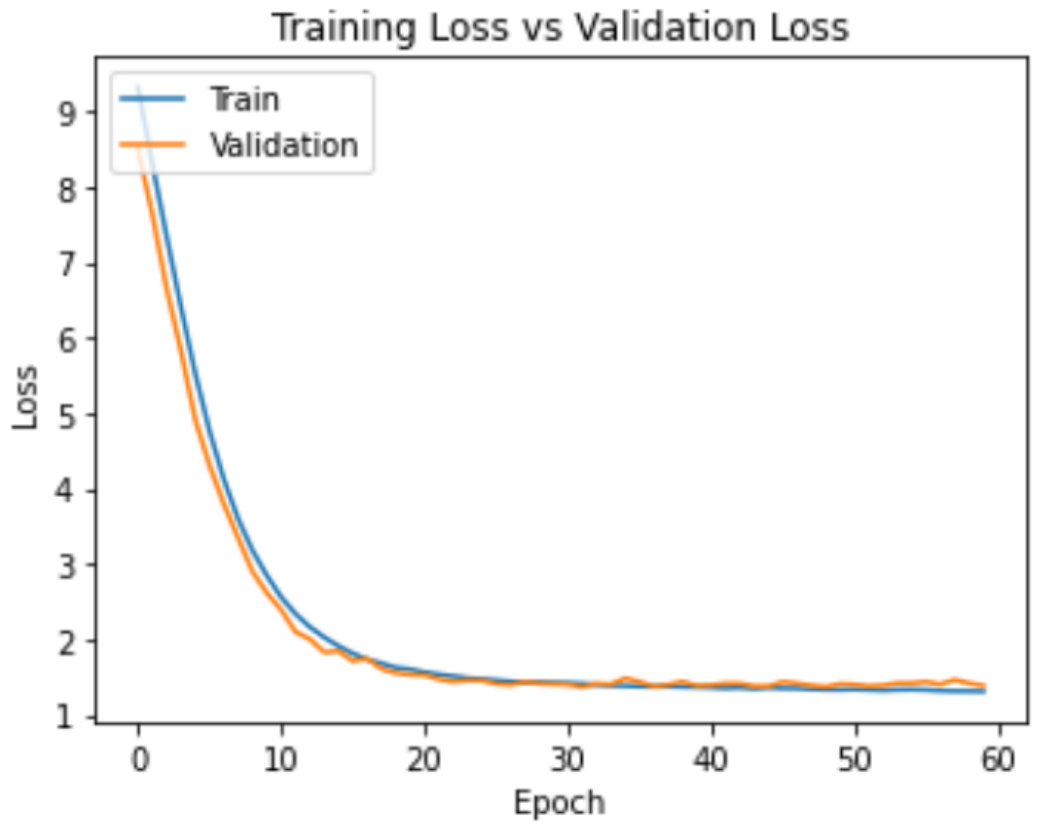
```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_5 (Conv2D)            (None, 48, 48, 32)        320
_________________________________________________________________
conv2d_6 (Conv2D)            (None, 48, 48, 64)        18496
_________________________________________________________________
batch_normalization_6 (Batch (None, 48, 48, 64)        256
_________________________________________________________________
max_pooling2d_4 (MaxPooling2 (None, 24, 24, 64)        0
_________________________________________________________________
dropout_6 (Dropout)          (None, 24, 24, 64)        0
_________________________________________________________________
conv2d_7 (Conv2D)            (None, 24, 24, 128)       204928
_________________________________________________________________
batch_normalization_7 (Batch (None, 24, 24, 128)       512
_________________________________________________________________
max_pooling2d_5 (MaxPooling2 (None, 12, 12, 128)       0
_________________________________________________________________
dropout_7 (Dropout)          (None, 12, 12, 128)       0
_________________________________________________________________
conv2d_8 (Conv2D)            (None, 12, 12, 512)       590336
_________________________________________________________________
batch_normalization_8 (Batch (None, 12, 12, 512)       2048
_________________________________________________________________
max_pooling2d_6 (MaxPooling2 (None, 6, 6, 512)         0
_________________________________________________________________
dropout_8 (Dropout)          (None, 6, 6, 512)         0
_________________________________________________________________
conv2d_9 (Conv2D)            (None, 6, 6, 512)         2359808
_________________________________________________________________
batch_normalization_9 (Batch (None, 6, 6, 512)         2048
_________________________________________________________________
max_pooling2d_7 (MaxPooling2 (None, 3, 3, 512)         0
_________________________________________________________________
dropout_9 (Dropout)          (None, 3, 3, 512)         0
_________________________________________________________________
flatten_1 (Flatten)          (None, 4608)              0
_________________________________________________________________
dense_3 (Dense)              (None, 256)               1179904
_________________________________________________________________
batch_normalization_10 (Batc (None, 256)               1024
_________________________________________________________________
dropout_10 (Dropout)         (None, 256)               0
_________________________________________________________________
dense_4 (Dense)              (None, 512)               131584
_________________________________________________________________
batch_normalization_11 (Batc (None, 512)               2048
_________________________________________________________________
dropout_11 (Dropout)         (None, 512)               0
_________________________________________________________________
dense_5 (Dense)              (None, 7)                 3591
=================================================================
Total params: 4,496,903
Trainable params: 4,492,935
Non-trainable params: 3,968
```

Evaluate the Dataset

모델 평가

```
history = model.fit(x = train_generator,epochs = epochs, validation_data = validation_generator)
```





Evaluate the Dataset

최종 모델 평가

```
train_loss, train_acc = model.evaluate(train_generator)
test_loss, test_acc   = model.evaluate(test_generator)
print("final train accuracy = {:.2f} , validation accuracy = {:.2f}".format(train_acc*100,
test_acc*100))
model.save_weights('model_weights.h5')
```

```
359/359 [==============================] - 27s 75ms/step - loss: 1.1792 - accuracy:
0.7208 113/113 [==============================] - 57s 508ms/step - loss: 1.3850 -
accuracy: 0.6498 final train accuracy = 72.08 , validation accuracy = 64.98
```

모델 평가 (루피 사진)

```
# 이미지 전처리 함수
def preprocess_image(image_path):
    img = image.load_img(image_path, target_size=(img_size, img_size), color_mode='grayscale')
    img_array = image.img_to_array(img)
    img_array = np.expand_dims(img_array, axis=0)  # 배치 차원 추가
    img_array = img_array / 255.0  # 정규화
    return img_array

# 예측 함수 정의
def predict_image(model, image_path):
    img_array = preprocess_image(image_path)

    # 예측 실행
    model_output = model.predict(img_array)  # TensorFlow/Keras 모델에서 예측

    predicted_class = np.argmax(model_output, axis=1)  # 예측된 클래스 인덱스
    return predicted_class[0]  # 예측된 클래스 반환
```

Evaluate the Dataset

모델 평가 (루피 사진)

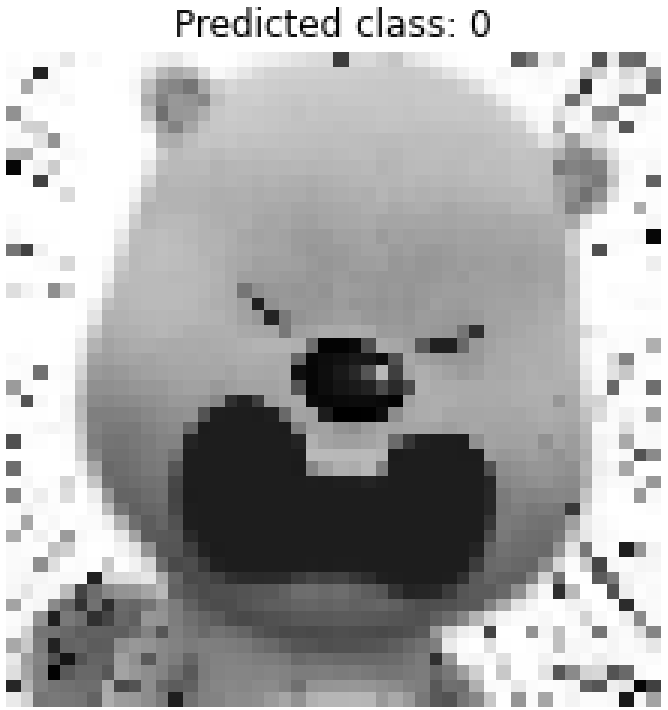
```
Predicted class: 4
```

4=Sad

Evaluate the Dataset

모델 평가 (루피 사진)

Predicted class: 0

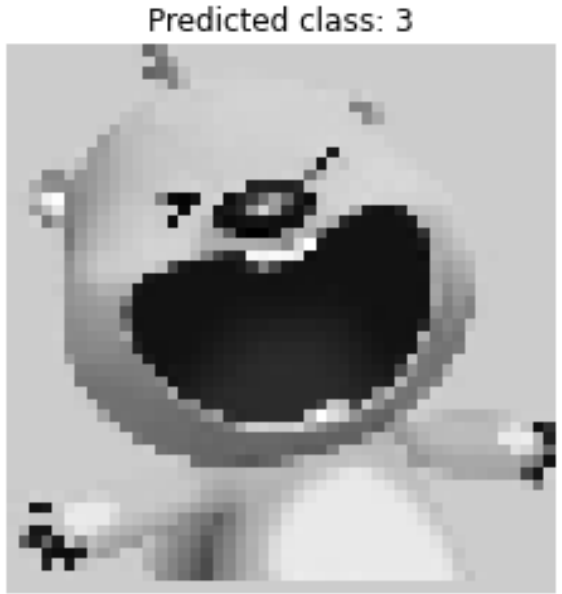


0 = Angry

Evaluate the Dataset

모델 평가 (루피 사진)

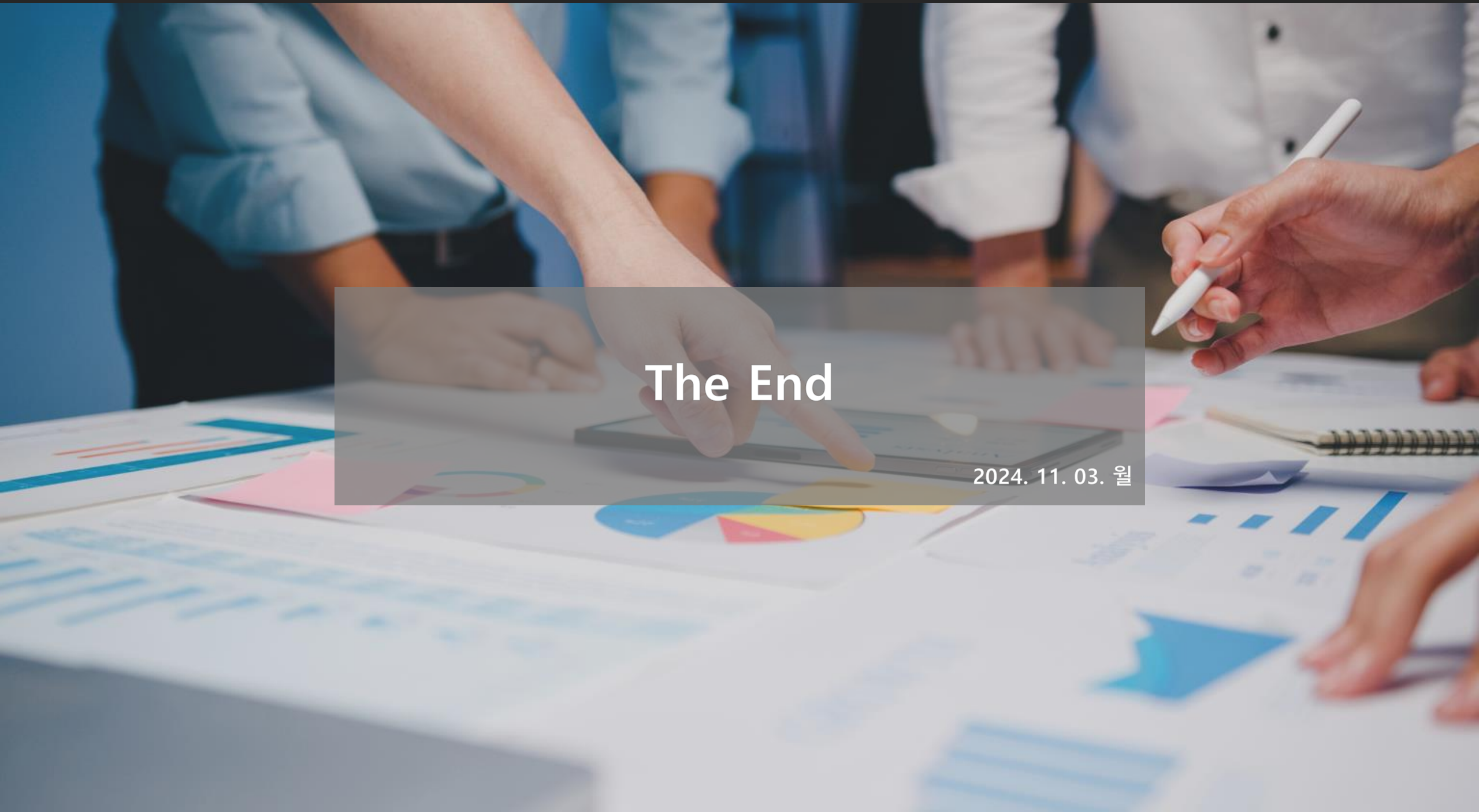
Predicted class: 3



3 = HAPPY

Conclusion

1. 캐글 그랜드 마스터의 코드를 참고하였는데, 해당 사용자도 Test 데이터를 Validation에 사용하고 또 다시 테스트할 때 사용하는 잘못된 방법을 사용하여 이를 수정하였다.
2. 사람 얼굴의 섬세한 감정을 학습하는 것에는 확실히 어려움이 있다는 것을 알 수 있었다. 데이터의 수가 훨씬 많아 보다 다양하게 학습이 필요하다고 생각했다.
3. 기존에 사전에 학습된 RESNET 50, RESNET 101 등을 사용하여 학습하면 해당 모델은 굉장히 방대한 데이터로 학습되어 모델 구성이 복잡해 동일한 데이터셋으로 학습하는데 epoch 30을 기준으로 4시간 ~5시간 정도의 시간이 필요했다. 하지만 성능은 크게 다르지 않았다.



The End

2024. 11. 03. 월