Mask or No mask?

2017108273 하정호

개요

끝날 기미가 보이지 않는 코로나 때문에 시민들의 마스크 착용 여부 확인이 필수가 되었습니다. Kaggle의 데이터셋을 활용하여 마스크 착용 여부를 분별해주는 딥 러닝 모델을 사용해보았습니다.

모듈 불러오기

```
import cv2
import inspect
from keras import Sequential, applications
from keras.callbacks import Callback, EarlyStopping
from keras.layers import Flatten, Dense
from keras.preprocessing.image import ImageDataGenerator
from math import ceil
import matplotlib.pyplot as plt
import random
from timeit import default_timer as timer
from tqdm import tqdm
```

제일 먼저 프로그램을 위해 필요한 필요한 모듈을 불러왔습니다.

데이터 준비

```
import numpy as np
import pandas as pd

import os
for dirname, _, filenames in os.walk('/kaggle/input'):
    for filename in filenames:
        print(os.path.join(dirname, filename))
```

```
/kaggle/input/covid-face-mask-detection-dataset/New Masks Dataset/Validation/Mask/2357.png
/kaggle/input/covid-face-mask-detection-dataset/New Masks Dataset/Validation/Mask/2025.jpg
/kaggle/input/covid-face-mask-detection-dataset/New Masks Dataset/Validation/Mask/2601.png
/kaggle/input/covid-face-mask-detection-dataset/New Masks Dataset/Validation/Mask/2004.jpg
/kaggle/input/covid-face-mask-detection-dataset/New Masks Dataset/Validation/Mask/2399.png
/kaggle/input/covid-face-mask-detection-dataset/New Masks Dataset/Validation/Mask/2586.png
/kaggle/input/covid-face-mask-detection-dataset/New Masks Dataset/Validation/Mask/6415.png
/kaggle/input/covid-face-mask-detection-dataset/New Masks Dataset/Validation/Mask/1841.jpg
/kaggle/input/covid-face-mask-detection-dataset/New Masks Dataset/Validation/Mask/2339.png
/kaggle/input/covid-face-mask-detection-dataset/New Masks Dataset/Test/Non Mask/real_01071.j
pg
/kaggle/input/covid-face-mask-detection-dataset/New Masks Dataset/Test/Non Mask/real_01061.j
pg
/kaggle/input/covid-face-mask-detection-dataset/New Masks Dataset/Test/Non Mask/real_01060.j
pg
/kaggle/input/covid-face-mask-detection-dataset/New Masks Dataset/Test/Non Mask/real_01055.j
pg
/kaggle/input/covid-face-mask-detection-dataset/New Masks Dataset/Test/Non Mask/real_01052.j
pg
/kaggle/input/covid-face-mask-detection-dataset/New Masks Dataset/Test/Non Mask/real_01044.j
pg
/kaggle/input/covid-face-mask-detection-dataset/New Masks Dataset/Test/Non Mask/real_01080.j
pg
/kaggle/input/covid-face-mask-detection-dataset/New Masks Dataset/Test/Non Mask/real_01036.j
pg
```

Kaggle에 있는 마스크를 쓴 사진과 마스크를 쓰지 않은 사진의 데이터셋을 불러옵니다.

데이터 준비

```
# 사진들의 디렉토리와 타겟 값을 dataframe으로 묶음
def folder_to_df(dirs, labels):
    file_list = []
    num_folders = len(dirs)
    for count, folder in enumerate(dirs, start=1):
        for file in sorted(os.listdir(folder)):
            if count < num_folders / 2:
                file_list.append((folder + '/' + file, labels[0]))
            else:
                file_list.append((folder + '/' + file, labels[1]))
    return pd.DataFrame(file_list, columns=['filename', 'class'])

labels = ['With Mask', 'Without Mask'] # 타겟 값: 마스크 착용, 마스크 미착용
all_mask_df = folder_to_df(with_mask_dirs + without_mask_dirs, labels) # 모든 사진들의 디렉토리를 d
ataframe으로 합침

shuffled_mask_df = all_mask_df.sample(frac=1) # 고른 선택을 위해 데이터 전체의 순서를 섞음

train_df, val_df, test_df = np.split(shuffled_mask_df,
                                     [int(0.8*len(shuffled_mask_df)), int(0.9*len(shuffled_mask_
df))]) # 데이터를 80:10:10의 비율로 train, validation, test 분할

# train, validation, test 세트 확인
print('total dataset : {}\ntraining set  : {}\nvalidation set: {}\ntest set      : {}'.format(al
l_mask_df.shape, train_df.shape, val_df.shape, test_df.shape))
display(train_df.head(5), val_df.head(5), test_df.head(5))
```

```
total dataset : (12798, 2)
training set  : (10238, 2)
validation set: (1280, 2)
test set      : (1280, 2)
```

◀결과 데이터

1. 사진들의 디렉토리를 dataframe으로 묶어주었습니다.

2. 타겟 값을 with mask와 without mask로 통일해주었습니다.

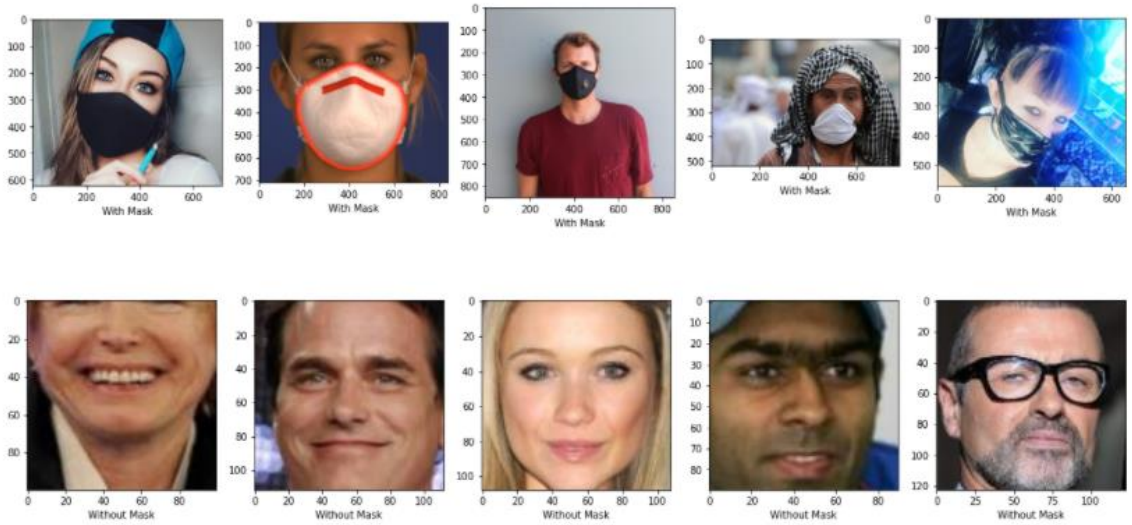
3. 데이터를 80 : 10 : 10의 비율로 훈련용, 확인용, 테스트용으로 분류해주었습니다.

데이터 준비

With/Without Masks

- 마스크 착용한/미착용한 이미지들을 확인합니다.

```
show_img_samples(with_mask_dirs, 5, 'With Mask')
show_img_samples(without_mask_dirs, 5, 'Without Mask')
```



디렉토리와 타겟값이 잘 적용 되었는지 확인해보았습니다.

훈련 모델 선정

```
# Keras Applications에 있는 모델들을 불러옴
model_dictionary = {m[0]:m[1] for m in inspect.getmembers(applications, inspect.isfunction)}
for key in model_dictionary:
    print(key)

# 모델들의 성능 기록
model_benchmarks = {'model_name': [], 'num_model_params': [], 'val_loss': [], 'val_accuracy':
[], 'avg_train_time': []}

for model_name, model in tqdm(model_dictionary.items()):
    # NASNet 모델들은 모델에 맞는 input_shape를 사용해야함
    if 'NASNetLarge' in model_name:
        input_shape=(331,331,3)
    elif 'NASNetMobile' in model_name:
        input_shape=(224,224,3)
    # 이외의 모델들은 이미지 최대 크기인 128x128으로 설정
    else:
        input_shape=(128,128,3)

    # 베이스 모델 설정
    pre_trained_model = model(
        include_top=False,
        pooling='avg',
        input_shape=input_shape)
    pre_trained_model.trainable = False

    # 전이학습 모델 설정
    clf_model = Sequential([
        pre_trained_model,
        Flatten(),
        Dense(1, activation='sigmoid')])
    clf_model.compile(
        optimizer='adam',
        loss='binary_crossentropy',
        metrics=['accuracy'])

    # 모델 훈련
    history = clf_model.fit(
        train_generator,
        epochs=3, # 모든 모델들을 비교해야하므로 epoch를 3으로 설정
        callbacks=[timing_callback],
        validation_data=val_generator)

    # 모델 성능 비교를 위한 값들을 기록
    model_benchmarks['model_name'].append(model_name)
    model_benchmarks['num_model_params'].append(pre_trained_model.count_params())
    model_benchmarks['val_loss'].append(history.history['val_loss'][-1])
    model_benchmarks['val_accuracy'].append(history.history['val_accuracy'][-1])
    model_benchmarks['avg_train_time'].append(sum(timing_callback.logs)/3)
```

1. 여러 모델들의 손실값, 정확도, 훈련시간을 비교합니다.

```
benchmark_df = pd.DataFrame(model_benchmarks)

benchmark_df.to_csv('pretrained_model_benchmarks.csv')

bm_params_df = benchmark_df.sort_values('num_model_params')
bm_params_df.head(10)
```

	model_name	num_model_params	val_loss	val_accuracy	avg_train_time
6	MobileNetV2	2257984	0.131125	0.953906	489.716313
5	MobileNet	3228864	0.105017	0.967188	423.961969
8	NASNetMobile	4269716	0.162294	0.969531	642.616535
0	DenseNet121	7037504	0.089965	0.975000	79.569398
1	DenseNet169	12642880	0.113288	0.975000	149.552018
15	VGG16	14714688	0.628976	0.660156	1130.113425
2	DenseNet201	18321984	0.098963	0.973437	220.808253
16	VGG19	20024384	0.571156	0.825781	1197.417554
17	Xception	20861480	0.102058	0.982813	1265.309791
4	InceptionV3	21802784	0.140824	0.958594	358.989423

2. 비교 후 손실값이 작고 정확도가 높은 DenseNet121을 베이스 모델로 선정합니다.

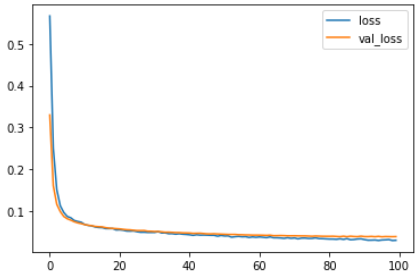
모델 훈련

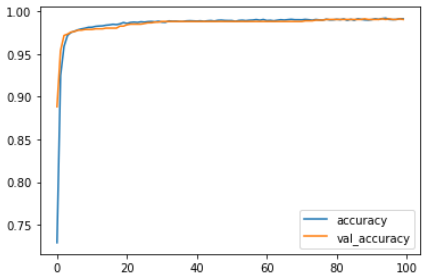
Check Model Performance

- 학습된 모델의 손실값과 정확도를 차트로 확인합니다.

```
history_df = pd.DataFrame(history.history)
history_df
history_df.loc[:, ['loss', 'val_loss']].plot();
history_df.loc[:, ['accuracy', 'val_accuracy']].plot();
print("Minimum validation loss: {}".format(history_df['val_loss'].min()))
print("Maximum validation accuracy: {}".format(history_df['val_accuracy'].max()))
```

```
Minimum validation loss: 0.03820304572582245
Maximum validation accuracy: 0.991406261920929
```





모델을 학습시키고 손실값과 정확도를 확인해보았습니다.

이미 많이 학습된 베이스 모델을 사용해서 초반부터 손실값과 정확도가 좋게 나온 것을 확인할 수 있습니다.

또한 확인용 데이터로 모델을 평가해보았습니다.

```
model.evaluate(test_generator)
```

```
10/10 [==============================] - 10s 1s/step - loss: 0.0332 - accuracy: 0.9898

[0.03321497142314911, 0.9898437261581421]
```

이후 값이 괜찮게 나온 것을 확인하고 모델을 저장하였습니다.

예측

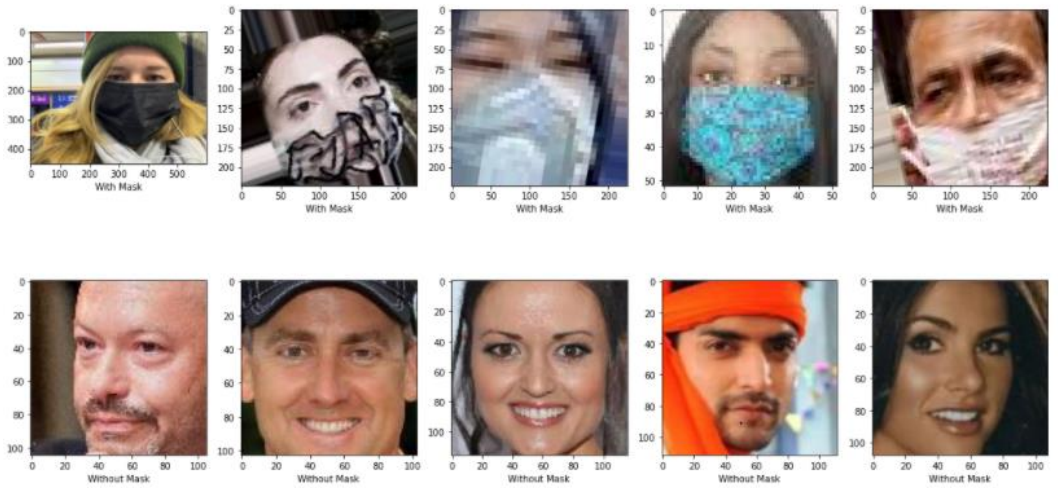
```
def predict_rand_sample(num, label):
    plt.figure(figsize=(20,10))
    img_dirs = test_df.loc[test_df['class'] == label].sample(num).values[:, 0] # 사진들을 임의로
선택
    for i in range(num):
        plt.subplot(num // 5 + 1, 5, i + 1)

        img = cv2.cvtColor(cv2.imread(img_dirs[i]), cv2.COLOR_BGR2RGB)
        processed_img = cv2.resize(img,(128,128))
        processed_img = np.reshape(processed_img,[1,128,128,3])
        processed_img = processed_img/255.0

        prediction = model.predict(processed_img)[0][0] # 사진의 결과값 예측
        if prediction < 0.5:
            plt.xlabel('With Mask')
        else:
            plt.xlabel('Without Mask')
        plt.imshow(img)
    plt.show
```

학습된 모델로 다른 사진들 속 인물이 마스크를 착용했는지 예측합니다.

```
predict_rand_sample(5, 'With Mask')
predict_rand_sample(5, 'Without Mask')
```



예측이 성공적으로 된 것을 확인할 수 있습니다.

결론

학습된 딥 러닝 모델로 사진 속 인물의 마스크 착용 여부를 확인하는 것이 가능했습니다.
또한 모델을 구축하면서 마스크를 귀에 걸었지만 입을 가리지 않은 사람들을 분류해내는 작업도 가능하면 좋을 것 같다는 생각이 들었습니다.

인공지능을 매우 어렵게 생각하고 있었는데 직접 딥 러닝을 통해 인공지능 모델을 사용해 보니 인공지능과 조금은 친해진 것 같다고 생각하였습니다.

감사합니다!