[예제] ImageDataGenerator 기반의 Cats and Dogs Transfer Learning

TransferLearning_Example_1 에서 구현한 Cats and Dogs Transfer Learning 코드를 ImageDataGenerator 이용해서 다시 구현하고. 인터넷에서 다운받은 임의의 cat ,dog 이미지에 대해 예측을 실행하시오

```
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.applications import MobileNet
from tensorflow.keras.layers import Flatten, Dense, Dropout, GlobalAveragePooling2D
from tensorflow.keras.preprocessing.image import ImageDataGenerator
```

```
# 파일 다운로드

!wget https://storage.googleapis.com/mledu-datasets/cats_and_dogs_filtered.zip
```

```
import os
import shutil

if os.path.exists('/content/cats_and_dogs_filtered/'):

    shutil.rmtree('/content/cats_and_dogs_filtered/')
    print('/content/cats_and_dogs_filtered/  is removed !!!')
```

```
import zipfile

with zipfile.ZipFile('/content/cats_and_dogs_filtered.zip', 'r') as target_file:

    target_file.extractall('/content/cats_and_dogs_filtered/')
```

```
train_dir = '/content/cats_and_dogs_filtered/cats_and_dogs_filtered/train'
test_dir = '/content/cats_and_dogs_filtered/cats_and_dogs_filtered/validation'
```

ImageDataGenerator 생성

```
IMG_WIDTH = 224
IMG_HEIGHT = 224
```

```
# ImageDataGenerator 이용하여 데이터 불러옴

train_data_gen = ImageDataGenerator(rescale=1./255,
                                    rotation_range=10,
                                    width_shift_range=0.2,
                                    height_shift_range=0.2,
                                    shear_range=0.2,
                                    zoom_range=0.2,
                                    horizontal_flip=True)

test_data_gen = ImageDataGenerator(rescale=1./255)
```

다양한 효과는 개발자가 알아서 선택함

(1) flow_from_directory 이용해서 train_data, test_data 생성

[train_data, test_data 생성 조건]

- batch_size =32, color_mode='rgb', shuffle=True,
  class_mode='sparse' or 'categorical' or 'binary' 각각에 대해서

(2) train_data, test_data 생성 후,

각 데이터에 대한 class_indices, classes, num_classes 출력함

(3) class_mode='sparse' or 'categorical' or 'binary' 각각에 대해서,
전이학습의 파인튜닝을 이용해서 95% 이상 정확도 달성
(epochs 10번 이상 20번 미만 실행)

```
import matplotlib.pyplot as plt

plt.title('accuracy trend')
plt.grid()
plt.xlabel('epochs')
plt.ylabel('accuracy')
plt.plot(hist.history['accuracy'], label='train')
plt.plot(hist.history['val_accuracy'], label='validation')
plt.legend(loc='best')
plt.show()
```

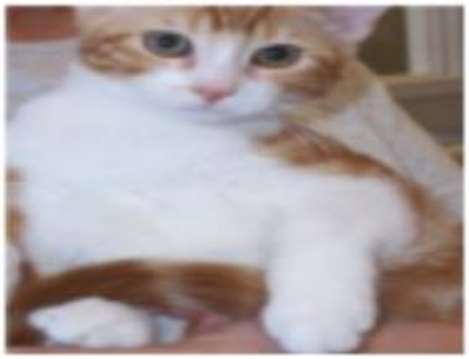
```
import matplotlib.pyplot as plt

plt.title('loss trend')
plt.grid()
plt.xlabel('epochs')
plt.ylabel('loss')
plt.plot(hist.history['loss'], label='train')
plt.plot(hist.history['val_loss'], label='validation')
plt.legend(loc='best')
plt.show()
```

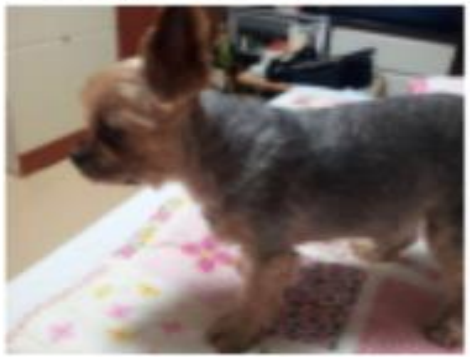
임의의 이미지에 대해, 다음과 같이 예측 확률과 함께 테스트 이미지를 출력하시오

cat , 99.99%

cat , 99.96%



dog , 99.54%



dog , 99.96%

dog , 99.83%

dog , 99.31%