개와 고양이를 구분하는 CNN 네트워크 구성

TOPCIT 대체 과제 1

컴퓨터공학과 1771025 방수정

제출일: 20.06.22

1. 해결 과정

모델을 자체적으로 구성할 계획이었으나,

그 경우 정확도가 최대 80%대에 머물렀음

↓

원인 분석

많은 데이터를 노트북의 CPU로 처리하기 버거움

구글링을 통해 정확도를 높이는 방법을 검색했지만, 크게 향상되지 않거나 GPU를 필요로 하는 솔루션

↓

해결 방안

Transfer Training [전이학습]

2. Transfer training

Transfer training이란?

이미 학습된 신경망을 가져와서 특징 추출능력을 이용하고, 추가 레이어를 연결해 사용하는 방법

사용한 모델: Inceptionv3 (Google에서 만든 모델로, 이미지 비교에 성능이 우수)

적은 데이터에서 효과적

빠른 training 속도

높은 정확도

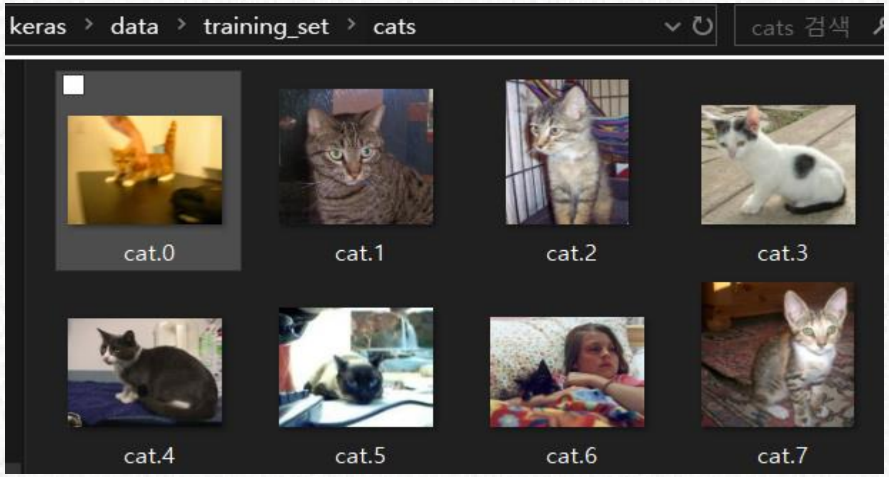
3. Dataset 준비

Data 출처

Kaggle Dog vs cat

Training data: dog image 4000,
cat image 4000

validation data: dog image 1000,
cat image 1000



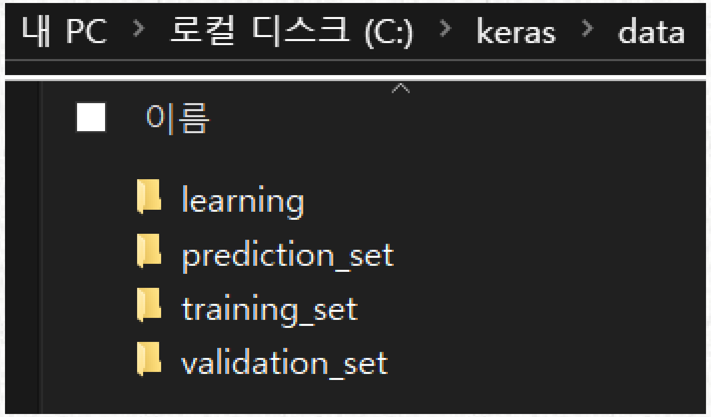
keras > data > training_set > cats

cats 검색

cat.0 cat.1 cat.2 cat.3
cat.4 cat.5 cat.6 cat.7

Data path

c:₩keras₩data



내 PC > 로컬 디스크 (C:) > keras > data

□ 이름

- learning
- prediction_set
- training_set
- validation_set

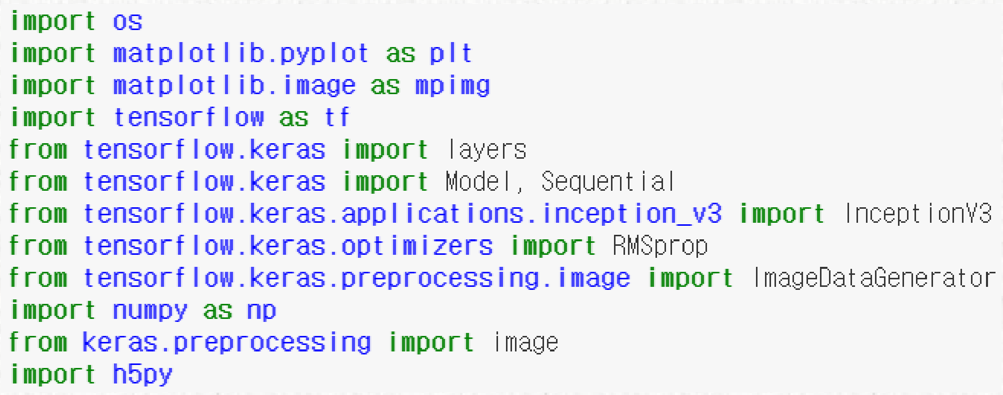
training_set
cats //4000개 cat image
dogs //4000개 dog image

validation set
cats //1000개 cat image
dogs //1000개 dog image

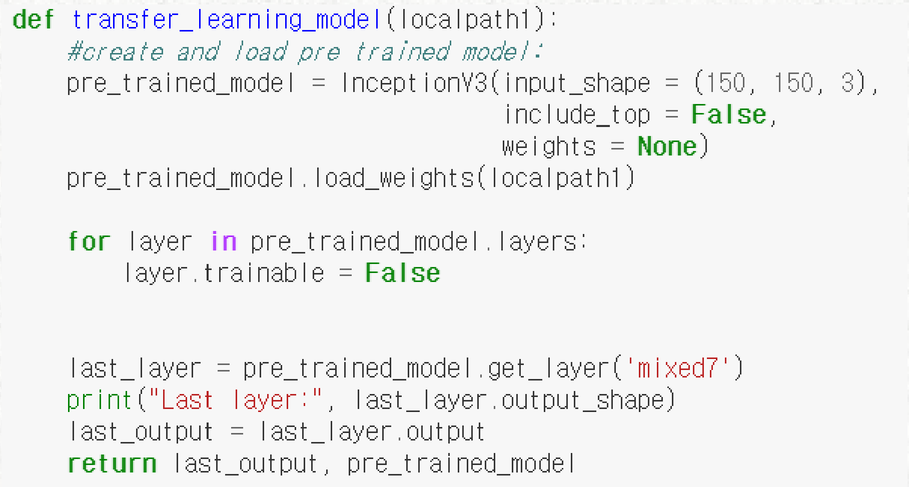
Prediction_set
Prediction에 사용할 이미지

learning
Inception v3 weight가 저장된 모델 (.h5)

4. 프로그램 구성 (1) dogNcat_training.ipynb



```
import os
import matplotlib.pyplot as plt
import matplotlib.image as mpimg
import tensorflow as tf
from tensorflow.keras import layers
from tensorflow.keras import Model, Sequential
from tensorflow.keras.applications.inception_v3 import InceptionV3
from tensorflow.keras.optimizers import RMSprop
from tensorflow.keras.preprocessing.image import ImageDataGenerator
import numpy as np
from keras.preprocessing import image
import h5py
```



```
def transfer_learning_model(localpath1):
    #create and load pre trained model:
    pre_trained_model = InceptionV3(input_shape = (150, 150, 3),
                                    include_top = False,
                                    weights = None)
    pre_trained_model.load_weights(localpath1)

    for layer in pre_trained_model.layers:
        layer.trainable = False


    last_layer = pre_trained_model.get_layer('mixed7')
    print("Last layer:", last_layer.output_shape)
    last_output = last_layer.output
    return last_output, pre_trained_model
```

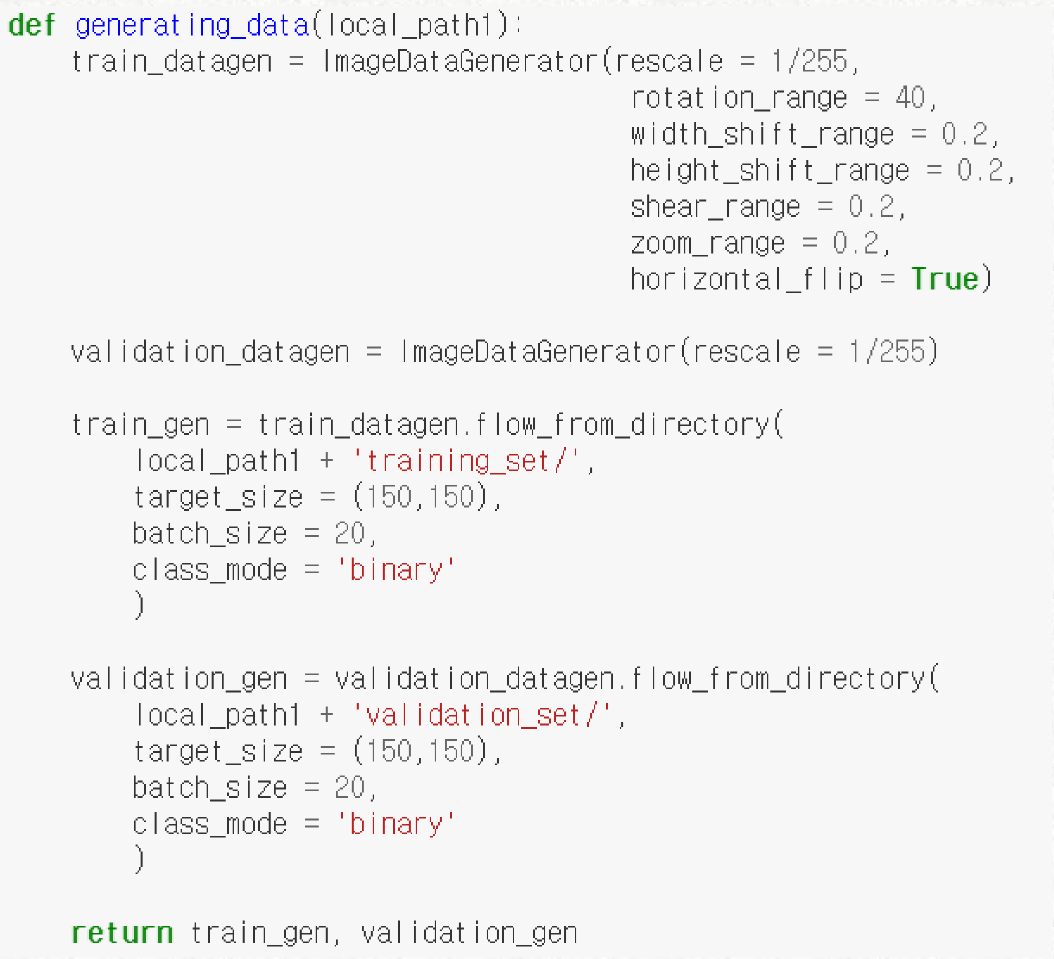
import library

- 필요한 라이브러리 가져오기
- 모델 구성, 트레이닝, 결과 확인 등에서 사용

def transfer_learning_model

- Pre_trained_model을 생성(Inception V3)
- 가중치를 가져와 모델에 넣기
- trainable은 False로 set

4. 프로그램 구성 (1) dogNcat_training.ipynb

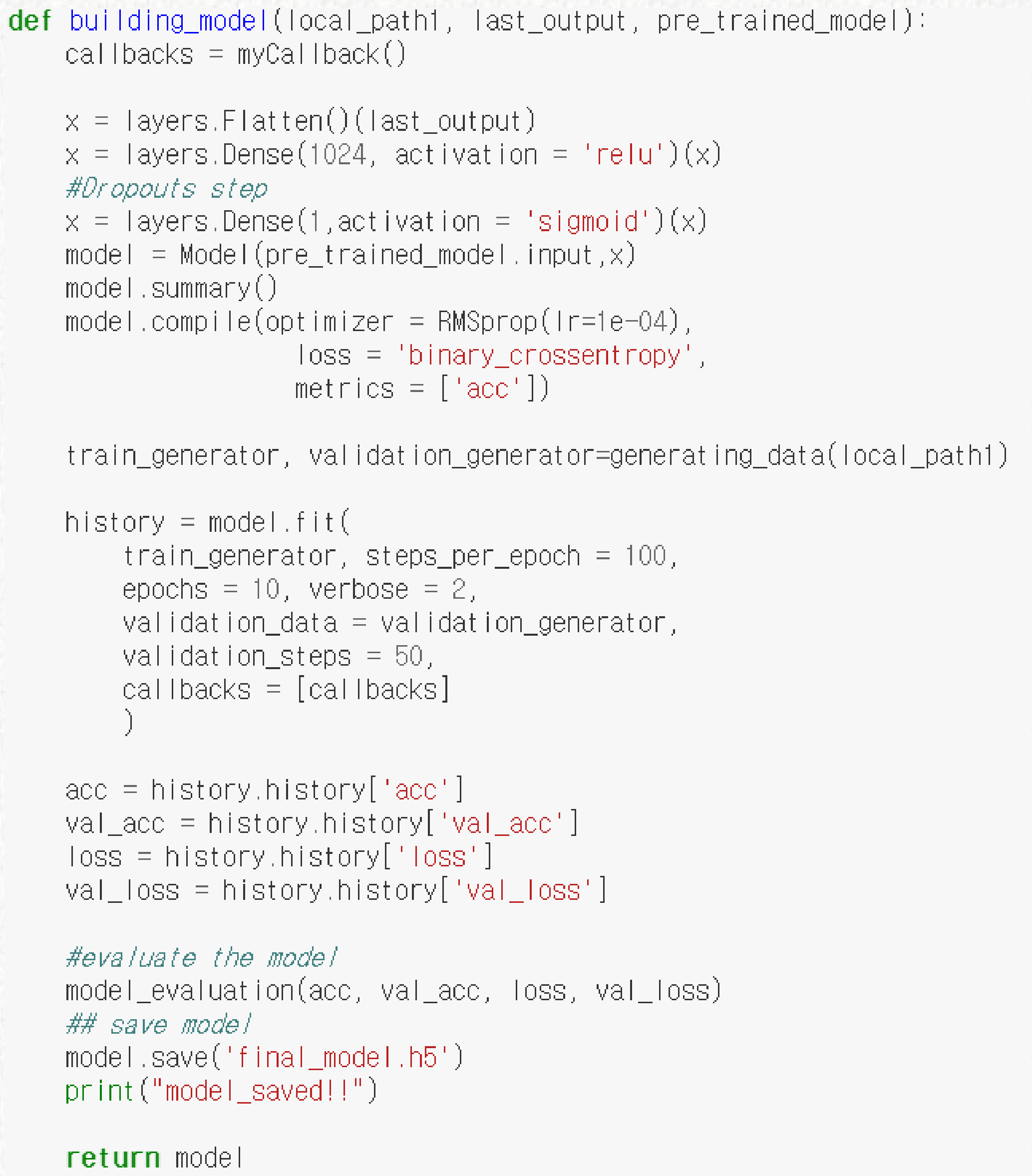


```
def generating_data(local_path1):
    train_datagen = ImageDataGenerator(rescale = 1/255,
                                       rotation_range = 40,
                                       width_shift_range = 0.2,
                                       height_shift_range = 0.2,
                                       shear_range = 0.2,
                                       zoom_range = 0.2,
                                       horizontal_flip = True)

    validation_datagen = ImageDataGenerator(rescale = 1/255)

    train_gen = train_datagen.flow_from_directory(
        local_path1 + 'training_set/',
        target_size = (150,150),
        batch_size = 20,
        class_mode = 'binary'
        )

    validation_gen = validation_datagen.flow_from_directory(
        local_path1 + 'validation_set/',
        target_size = (150,150),
        batch_size = 20,
        class_mode = 'binary'
        )

    return train_gen, validation_gen
```

def generating_data

- local_path에 저장되어 있는 이미지를 training에 사용할 수 있게 변환
- training에 사용할 train_gen 생성
- validation에 사용할 validation_gen 생성

4. 프로그램 구성 (1) dogNcat_training.ipynb

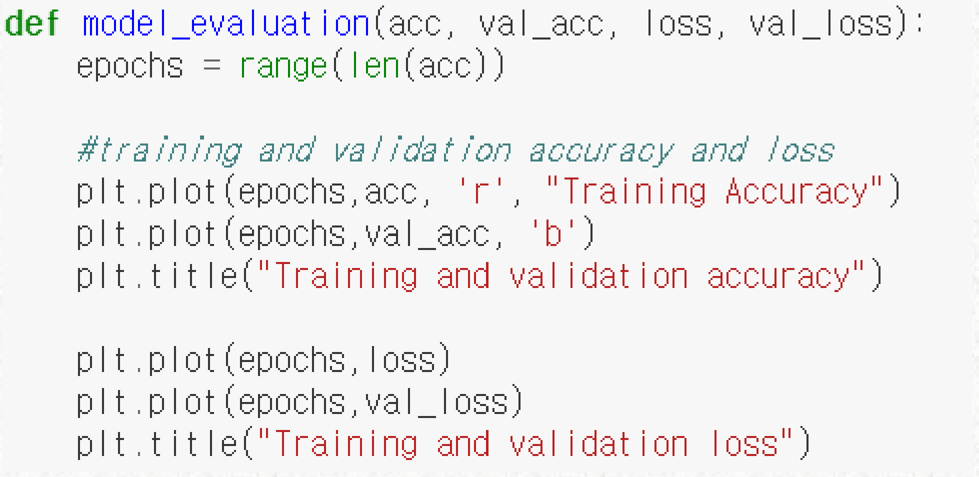


```python
def building_model(local_path1, last_output, pre_trained_model):
    callbacks = myCallback()

    x = layers.Flatten()(last_output)
    x = layers.Dense(1024, activation = 'relu')(x)
    #Dropouts step
    x = layers.Dense(1,activation = 'sigmoid')(x)
    model = Model(pre_trained_model.input,x)
    model.summary()
    model.compile(optimizer = RMSprop(lr=1e-04),
                  loss = 'binary_crossentropy',
                  metrics = ['acc'])

    train_generator, validation_generator=generating_data(local_path1)

    history = model.fit(
        train_generator, steps_per_epoch = 100,
        epochs = 10, verbose = 2,
        validation_data = validation_generator,
        validation_steps = 50,
        callbacks = [callbacks]
        )

    acc = history.history['acc']
    val_acc = history.history['val_acc']
    loss = history.history['loss']
    val_loss = history.history['val_loss']

    #evaluate the model
    model_evaluation(acc, val_acc, loss, val_loss)
    ## save model
    model.save('final_model.h5')
    print("model_saved!!")

    return model
```

def building_model

- callback함수로 overfitting시 early stop
- pre_trained_model과 추가 레이어 x를 연결
- model compile
- model fit으로 training
- final_model.h5로 저장하고 predict에 사용

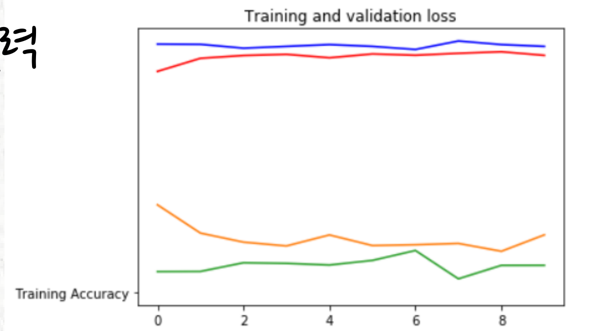
4. 프로그램 구성 (1) dogNcat_training.ipynb

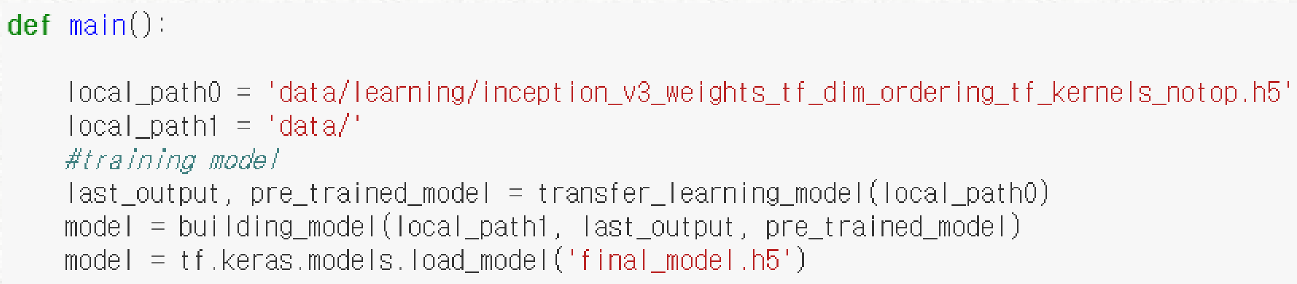


```
def model_evaluation(acc, val_acc, loss, val_loss):
    epochs = range(len(acc))

    #training and validation accuracy and loss
    plt.plot(epochs,acc, 'r', "Training Accuracy")
    plt.plot(epochs,val_acc, 'b')
    plt.title("Training and validation accuracy")

    plt.plot(epochs,loss)
    plt.plot(epochs,val_loss)
    plt.title("Training and validation loss")
```

def model_evaluation

- Training과 validation에 대한 accuracy, loss를 그래프로 출력



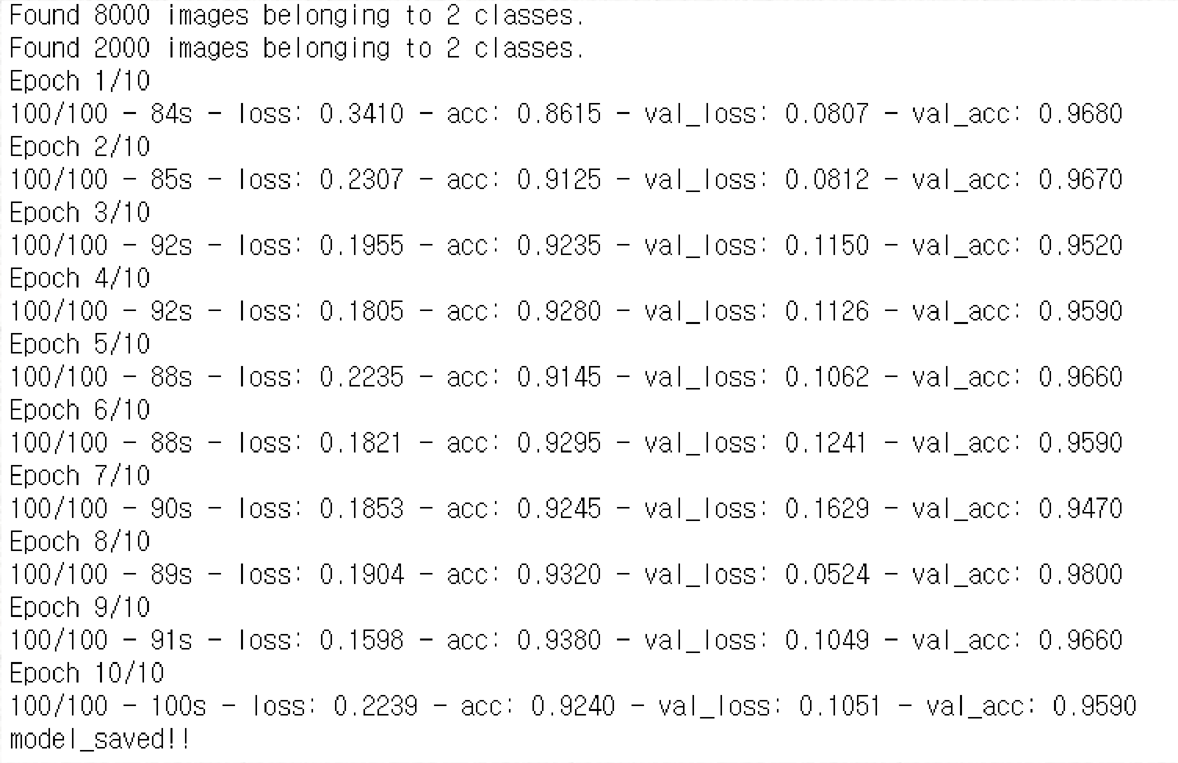


```
def main():

    local_path0 = 'data/learning/inception_v3_weights_tf_dim_ordering_tf_kernels_notop.h5'
    local_path1 = 'data/'
    #training model
    last_output, pre_trained_model = transfer_learning_model(local_path0)
    model = building_model(local_path1, last_output, pre_trained_model)
    model = tf.keras.models.load_model('final_model.h5')
```

def main

- transfer_learning_model함수로 pre_trained_model 생성
- pre_trained_model과 병합 & 트레이닝

4. 프로그램 구성 (1) dogNcat_training.ipynb



```
Found 8000 images belonging to 2 classes.
Found 2000 images belonging to 2 classes.
Epoch 1/10
100/100 - 84s - loss: 0.3410 - acc: 0.8615 - val_loss: 0.0807 - val_acc: 0.9680
Epoch 2/10
100/100 - 85s - loss: 0.2307 - acc: 0.9125 - val_loss: 0.0812 - val_acc: 0.9670
Epoch 3/10
100/100 - 92s - loss: 0.1955 - acc: 0.9235 - val_loss: 0.1150 - val_acc: 0.9520
Epoch 4/10
100/100 - 92s - loss: 0.1805 - acc: 0.9280 - val_loss: 0.1126 - val_acc: 0.9590
Epoch 5/10
100/100 - 88s - loss: 0.2235 - acc: 0.9145 - val_loss: 0.1062 - val_acc: 0.9660
Epoch 6/10
100/100 - 88s - loss: 0.1821 - acc: 0.9295 - val_loss: 0.1241 - val_acc: 0.9590
Epoch 7/10
100/100 - 90s - loss: 0.1853 - acc: 0.9245 - val_loss: 0.1629 - val_acc: 0.9470
Epoch 8/10
100/100 - 89s - loss: 0.1904 - acc: 0.9320 - val_loss: 0.0524 - val_acc: 0.9800
Epoch 9/10
100/100 - 91s - loss: 0.1598 - acc: 0.9380 - val_loss: 0.1049 - val_acc: 0.9660
Epoch 10/10
100/100 - 100s - loss: 0.2239 - acc: 0.9240 - val_loss: 0.1051 - val_acc: 0.9590
model_saved!!
```

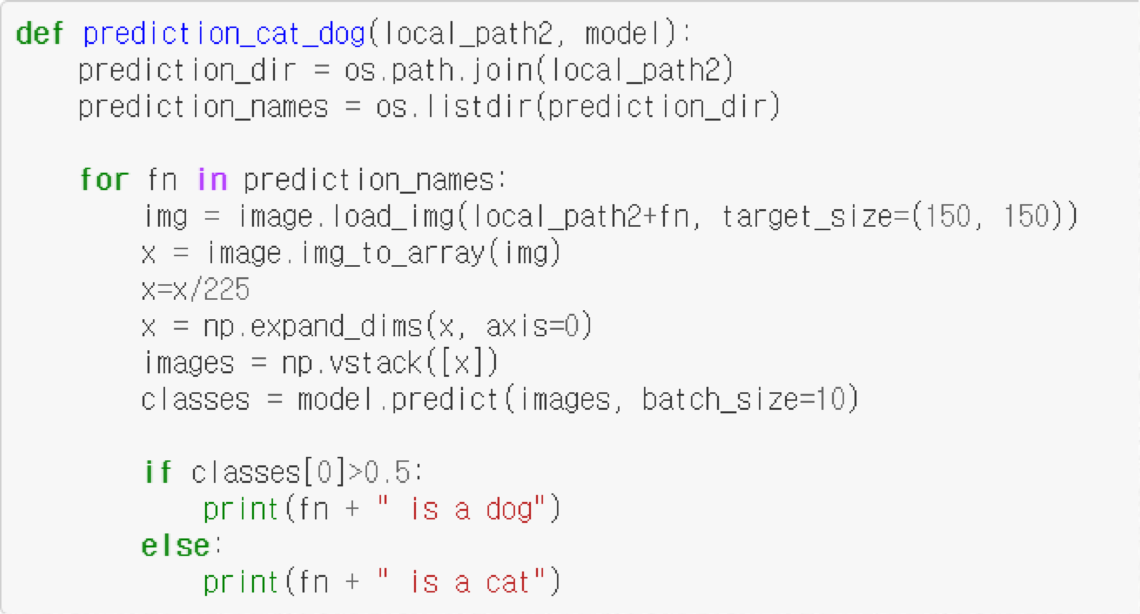
Training process

- 8000개의 이미지로 training
- 2000개의 이미지로 validation
- epoch=10

evaluation result

- training accuracy: 92%
- validation accuracy: 95.9%

4. 프로그램 구성 (2) dogNcat_predict.ipynb

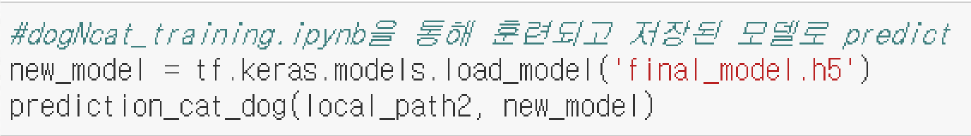


```
def prediction_cat_dog(local_path2, model):
    prediction_dir = os.path.join(local_path2)
    prediction_names = os.listdir(prediction_dir)

    for fn in prediction_names:
        img = image.load_img(local_path2+fn, target_size=(150, 150))
        x = image.img_to_array(img)
        x=x/225
        x = np.expand_dims(x, axis=0)
        images = np.vstack([x])
        classes = model.predict(images, batch_size=10)

        if classes[0]>0.5:
            print(fn + " is a dog")
        else:
            print(fn + " is a cat")
```

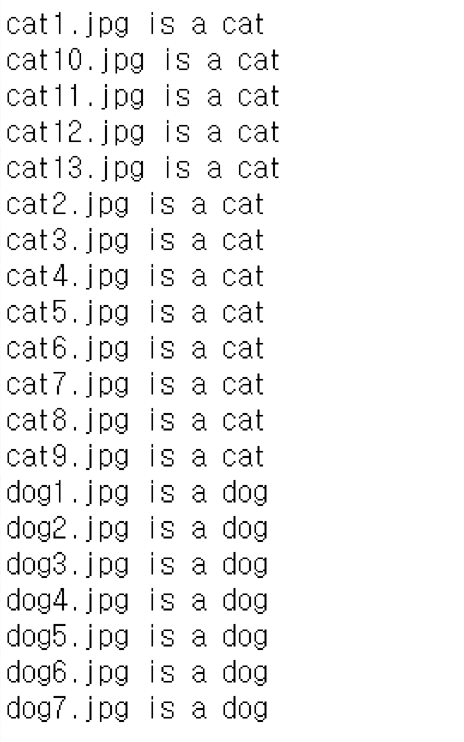
def prediction_cat_dog

- 이미지를 가져와 cat인지 dog인지 예측
- 이미지를 디렉토리에서 가져와 입력에 맞게 조정
- 결과값이 0.5를 기준으로 cat, dog을 구분

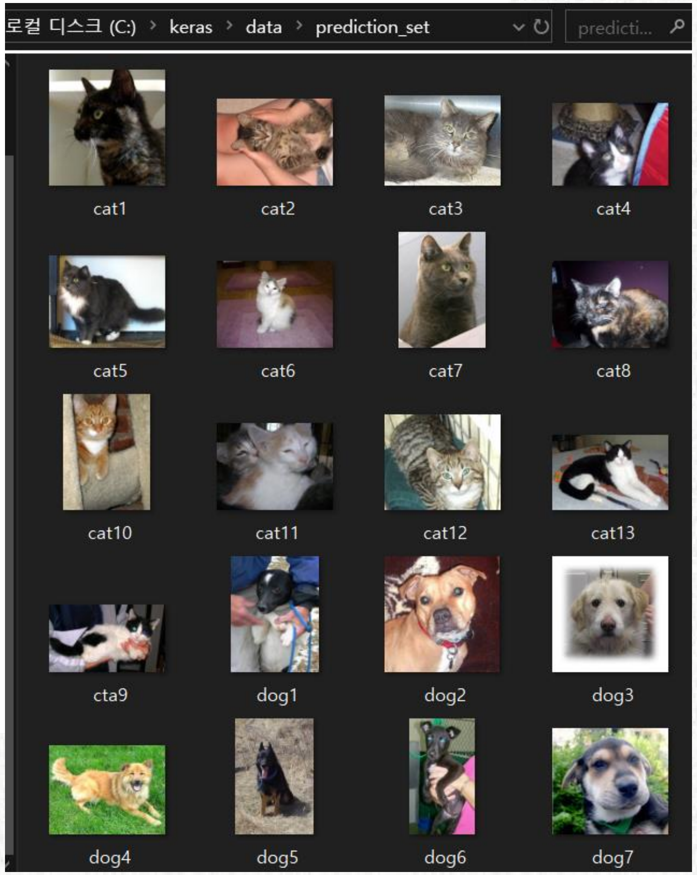
4. 프로그램 구성 (2) dogNcat_predict.ipynb



```
#dogNcat_training.ipynb을 통해 훈련되고 저장된 모델로 predict
new_model = tf.keras.models.load_model('final_model.h5')
prediction_cat_dog(local_path2, new_model)
```



```
cat1.jpg is a cat
cat10.jpg is a cat
cat11.jpg is a cat
cat12.jpg is a cat
cat13.jpg is a cat
cat2.jpg is a cat
cat3.jpg is a cat
cat4.jpg is a cat
cat5.jpg is a cat
cat6.jpg is a cat
cat7.jpg is a cat
cat8.jpg is a cat
cat9.jpg is a cat
dog1.jpg is a dog
dog2.jpg is a dog
dog3.jpg is a dog
dog4.jpg is a dog
dog5.jpg is a dog
dog6.jpg is a dog
dog7.jpg is a dog
```



Prediction process

- 미리 훈련된 모델을 load
- prediction image에 대해 예측

Prediction result

- 20개의 이미지에 대해 예측 수행
- 13개 cat image, 7개 dog image
- 100% 예측 성공

감사합니다!

1771025 방수정

https://github.com/tnwjd7732/cat-N-dog-classification