

딥러닝

과제4

2019305059

이현수

문제 1> 교재 7장 감성분석 RNN에 관련한 내용이다.

다음 테스트 문장을 사용하기 전에 학습을 시킨다.

```
import tensorflow as tf
import numpy as np

path_to_train_file = tf.keras.utils.get_file('train.txt', 'https://raw.githubusercontent.com/e9t/nsmc/master/ratings_train.txt')
path_to_test_file = tf.keras.utils.get_file('test.txt', 'https://raw.githubusercontent.com/e9t/nsmc/master/ratings_test.txt')

train_text = open(path_to_train_file, 'rb').read().decode(encoding='utf-8')
test_text = open(path_to_test_file, 'rb').read().decode(encoding='utf-8')

train_Y = np.array([[int(row.split('#t')[2])] for row in train_text.split('\n')[1:] if row.count('#t') > 0])
test_Y = np.array([[int(row.split('#t')[2])] for row in test_text.split('\n')[1:] if row.count('#t') > 0])
```

교재코드 그대로 적용해
학습시킴.

```
import re

def clean_str(string):
    string = re.sub(r"[가-힣A-Za-z0-9(),!@#%&]", " ", string)
    string = re.sub(r"%s", " %s", string)
    string = re.sub(r"%ve", " %ve", string)
    string = re.sub(r"%t", " %t", string)
    string = re.sub(r"%re", " %re", string)
    string = re.sub(r"%d", " %d", string)
    string = re.sub(r"%ll", " %ll", string)
    string = re.sub(r",", " ", string)
    string = re.sub(r"!", " ! ", string)
    string = re.sub(r"(", " ( ", string)
    string = re.sub(r"%", " % ", string)
    string = re.sub(r"%?", " %? ", string)
    string = re.sub(r"%s[2,]", " ", string)
    string = re.sub(r"%[2,]", " ", string)
    string = re.sub(r"%'", " ", string)
    return string.lower()

train_text_X = [row.split('#t')[1] for row in train_text.split('\n')[1:] if row.count('#t') > 0]
train_text_X = [clean_str(sentence) for sentence in train_text_X]
sentences = [sentence.split(' ') for sentence in train_text_X]

sentences_new = []
for sentence in sentences:
    sentences_new.append([word[:5] for word in sentence][:25])
sentences = sentences_new

from tensorflow.keras.preprocessing.text import Tokenizer
from tensorflow.keras.preprocessing.sequence import pad_sequences

tokenizer = Tokenizer(num_words=20000)
tokenizer.fit_on_texts(sentences)
train_X = tokenizer.texts_to_sequences(sentences)
train_X = pad_sequences(train_X, padding='post')
```

```
model = tf.keras.Sequential([
    tf.keras.layers.Embedding(20000, 300, input_length=25),
    tf.keras.layers.LSTM(units=50),
    tf.keras.layers.Dense(2, activation='softmax')
])

model.compile(optimizer='adam', loss='sparse_categorical_crossentropy', metrics=['accuracy'])
history = model.fit(train_X, train_Y, epochs=5, batch_size=128, validation_split=0.2)
```

Epoch 1/5
938/938 [=====] - 113s 120ms/step - loss: 0.4312 - accuracy: 0.7859 - val_loss: 0.3781 - val_accuracy: 0.8197
Epoch 2/5
938/938 [=====] - 112s 120ms/step - loss: 0.3225 - accuracy: 0.8477 - val_loss: 0.3806 - val_accuracy: 0.8222
Epoch 3/5
938/938 [=====] - 112s 119ms/step - loss: 0.2694 - accuracy: 0.8702 - val_loss: 0.4326 - val_accuracy: 0.8189
Epoch 4/5
938/938 [=====] - 112s 119ms/step - loss: 0.2275 - accuracy: 0.8884 - val_loss: 0.4794 - val_accuracy: 0.8139
Epoch 5/5
938/938 [=====] - 112s 119ms/step - loss: 0.1920 - accuracy: 0.9066 - val_loss: 0.5659 - val_accuracy: 0.8069

①다음 조건에 따라 테스트 문장 3개를 만든다. *테스트 문장 3개를 제출

영화 리뷰 - Windows 메모장

파일(F) 편집(E) 서식(O) 보기(V) 도움말(H)
출연배우들의 연기가 훌륭했습니다. 무엇보다 마지막장면에서 반전이있는데 매우 감동했습니다! 재미와 감동 모두 만족시켜준 명작입니다. 무조건 보세요.
영화줄거리가 이해가 안되는건 나뿐인가요? 형편없는 영화입니다. 영화보는내내 졸려서 죽는줄 알았습니다... 돈버리지마시고 이영화 보지마세요.
스파이더맨의 전투장면이 인상깊었습니다. 아직도 길거리전투가 생생히 기억납니다! 역시 기대에 부응했습니다.

```
from google.colab import drive
drive.mount('/content/gdrive')
```

Mounted at /content/gdrive

```
mytest_text = open('/content/gdrive/MyDrive/Colab Notebooks/레포트4/영화 리뷰.txt', 'rb').read().decode(encoding='utf-8')
print('Length of text: {} characters'.format(len(mytest_text)))
print(mytest_text)
```

Length of text: 221 characters
출연배우들의 연기가 훌륭했습니다. 무엇보다 마지막장면에서 반전이있는데 매우 감동했습니다! 재미와 감동 모두 만족시켜준 명작입니다. 무조건 보세요.
영화줄거리가 이해가 안되는건 나뿐인가요? 형편없는 영화입니다. 영화보는내내 졸려서 죽는줄 알았습니다... 돈버리지마시고 이영화 보지마세요.
스파이더맨의 전투장면이 인상깊었습니다. 아직도 길거리전투가 생생히 기억납니다! 역시 기대에 부응했습니다.

②테스트 문장 3개를 예제 7.22를 이용하여 [OUT]을 제출

```
mytest_text = [row for row in mytest_text.split('\n')]

mytest_text = [clean_str(sentence) for sentence in mytest_text]

# 문장을 띄어쓰기 단위로 단어 분리
sentences = [sentence.split(' ') for sentence in mytest_text]

for i in range(3):
    print(sentences[i])

['출연한배우들의', '연기가', '훌륭했습니다', '무엇보다', '마지막장면에서', '반전이있는데', '매우', '감동했습니다', '!', '재미와', '감동', '모두', '만족시켜준', '명작입니다', '무조건', '보세요', '']
['영화줄거리가', '이해가', '안되는건', '나뿐인가요', '###?', '형편없는', '영화입니다', '영화보는내내', '줄려서', '죽는줄', '알았습니다', '돈버리지마시고', '이영화', '보지마세요', '']
['스파이더맨의', '전투장면이', '인상깊었습니다', '아직도', '길거리전투가', '생생히', '기억납니다', '!', '역시', '기대에', '부응했습니다', '']
```

③테스트 문장 3개를 예제 7.24를 이용하여 [OUT]을 제출

```
sentences_new = []
for sentence in sentences:
    sentences_new.append([word[:5] for word in sentence][:25])
sentences = sentences_new
for i in range(3):
    print(sentences[i])

['출연한배우', '연기가', '훌륭했습니', '무엇보다', '마지막장면', '반전이있는', '매우', '감동했습니', '!', '재미와', '감동', '모두', '만족시켜준', '명작입니다', '무조건', '보세요', '']
['영화줄거리', '이해가', '안되는건', '나뿐인가요', '###?', '형편없는', '영화입니다', '영화보는내', '줄려서', '죽는줄', '알았습니다', '돈버리지마', '이영화', '보지마세요', '']
['스파이더맨', '전투장면이', '인상깊었습', '아직도', '길거리전투', '생생히', '기억납니다', '!', '역시', '기대에', '부응했습니', '']
```

④테스트 문장 3개를 예제 7.25를 이용하여 [OUT]을 제출

```
mytest_X = tokenizer.texts_to_sequences(sentences)
mytest_X = pad_sequences(mytest_X, padding='post')

print(mytest_X)
```

[[106	14739	604	1170	238	19324	2	1199	100	130	1255	448
	198	1]										
[201	5	2315	151	3662	4157	1307	6055	182	611	1	0
	0	0]										
[2169	139	8083	2	45	4067	1	0	0	0	0	0
	0	0]]										

실제로 입력의 길이인 25로 전처리되고 패딩이 되었지만 mytest_X출력결과와는 다르게 나옴. 하지만 실제로 25길 이로 전처리, 패딩처리가 되었기때문에 테스트 시 정상적으로 테스트됨.

⑤테스트 문장 3개를 예제 7.30을 이용하여 evaluate [OUT]을 제출

```
import numpy as np
mytest_Y=np.array([[1],[0], [1]])

model.evaluate(mytest_X, mytest_Y, verbose=0)


[0.032164137810468674, 1.0]
```

테스트할 때 정답은 직접만든다. 첫문장과 세번째문장은 긍정(1), 두번째문장은 부정(0)이다.


정확도 100%로 나옴.

문제 3> 교재 8장 2절 전이학습에 관련한 내용이다.

①kaggle 계정을 만들고 예제 8.8을 수행한 [IN]과 [OUT]을 제출

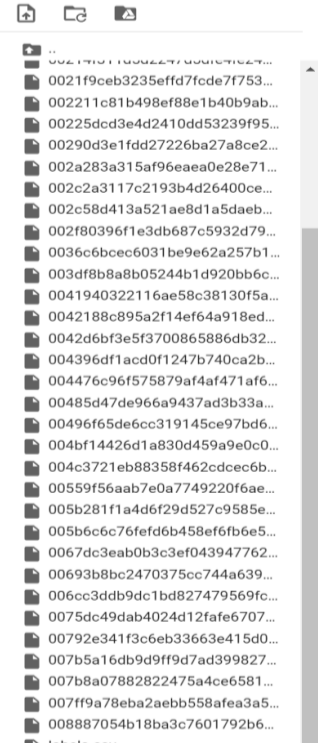


ehs2019305059
Joined 4 days ago · last seen in the past day


Competitions
Novice

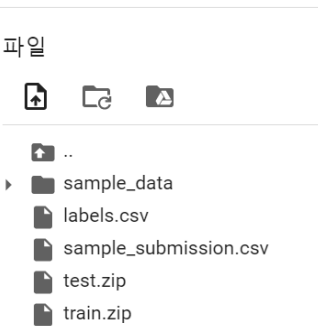
!pip install kaggle

```
Requirement already satisfied: kaggle in /usr/local/lib/python3.6/dist-packages (1.5.9)
Requirement already satisfied: certifi in /usr/local/lib/python3.6/dist-packages (from kaggle) (2020.11.8)
Requirement already satisfied: tqdm in /usr/local/lib/python3.6/dist-packages (from kaggle) (4.41.1)
Requirement already satisfied: urllib3 in /usr/local/lib/python3.6/dist-packages (from kaggle) (1.24.3)
Requirement already satisfied: python-dateutil in /usr/local/lib/python3.6/dist-packages (from kaggle) (2.8.1)
Requirement already satisfied: slugify in /usr/local/lib/python3.6/dist-packages (from kaggle) (0.0.1)
Requirement already satisfied: python-slugify in /usr/local/lib/python3.6/dist-packages (from kaggle) (4.0.1)
Requirement already satisfied: requests in /usr/local/lib/python3.6/dist-packages (from kaggle) (2.23.0)
Requirement already satisfied: six>=1.10 in /usr/local/lib/python3.6/dist-packages (from kaggle) (1.15.0)
Requirement already satisfied: text-unidecode>=1.3 in /usr/local/lib/python3.6/dist-packages (from python-slugify->kaggle) (1.3)
Requirement already satisfied: idna<3,>=2.5 in /usr/local/lib/python3.6/dist-packages (from requests->kaggle) (2.10)
Requirement already satisfied: chardet<4,>=3.0.2 in /usr/local/lib/python3.6/dist-packages (from requests->kaggle) (3.0.4)
```



```
import os
os.environ['KAGGLE_USERNAME'] = 'ehs2019305059'
os.environ['KAGGLE_KEY'] = '914467037bd74512956e403c5b1d0abc'
!kaggle competitions download -c dog-breed-identification

Warning: Looks like you're using an outdated API Version, please consider updating (server 1.5.10 / client 1.5.4)
001510bc8570bbeee98c8d80c8a95ec1.jpg: Skipping, found more recently modified local copy (use --force to force download)
0012a730dfa437f5f3613fb75efcd4ce.jpg: Skipping, found more recently modified local copy (use --force to force download)
0036c6bccc6031be9e62a527b1c3c442.jpg: Skipping, found more recently modified local copy (use --force to force download)
002c2a3117c2193b4d26400ce431ebdb.jpg: Skipping, found more recently modified local copy (use --force to force download)
002c58d413a521ae8d1a5daeb3c803.jpg: Skipping, found more recently modified local copy (use --force to force download)
004bf14426d1a830d459a9e0c0721309.jpg: Skipping, found more recently modified local copy (use --force to force download)
00102ee9d8eb90812350685311fe5890.jpg: Skipping, found more recently modified local copy (use --force to force download)
004479c96f575879af4af471af65cae8.jpg: Skipping, found more recently modified local copy (use --force to force download)
000621fb3cbb32d8935728e48679680e.jpg: Skipping, found more recently modified local copy (use --force to force download)
00485d47de966eaa437ad3b33ac193b6f.jpg: Skipping, found more recently modified local copy (use --force to force download)
0042d6bf3e5f3700865886db32689436.jpg: Skipping, found more recently modified local copy (use --force to force download)
0041940322116ae58c38130f5a5a.jpg: Skipping, found more recently modified local copy (use --force to force download)
0042188c895a2f14ef64a918ed.jpg: Skipping, found more recently modified local copy (use --force to force download)
0042d6bf3e5f3700865886db32.jpg: Skipping, found more recently modified local copy (use --force to force download)
004396df1acd0f1247b740ca2b.jpg: Skipping, found more recently modified local copy (use --force to force download)
004479c96f575879af4af471af65cae8.jpg: Skipping, found more recently modified local copy (use --force to force download)
00485d47de966eaa437ad3b33ac193b6f.jpg: Skipping, found more recently modified local copy (use --force to force download)
00496f65de6cc319145ce97bd6.jpg: Skipping, found more recently modified local copy (use --force to force download)
004bf14426d1a830d459a9e0c0721309.jpg: Skipping, found more recently modified local copy (use --force to force download)
004c3721eb88358f462cdcecb2380b7.jpg: Skipping, found more recently modified local copy (use --force to force download)
005b281f1a4d6f29d527c9585e9bd33c.jpg: Skipping, found more recently modified local copy (use --force to force download)
005b6c6c76fefd6b458ef6fb6e5.jpg: Skipping, found more recently modified local copy (use --force to force download)
00225dc3e4d2410dd53239f95c0352f.jpg: Skipping, found more recently modified local copy (use --force to force download)
006cc3ddb9dc1bd82747956f1cd52dc.jpg: Skipping, found more recently modified local copy (use --force to force download)
002a283a315af96eaa0e28e7163b21b.jpg: Skipping, found more recently modified local copy (use --force to force download)
00214f311d5d2247d5dfe4fe24b2303d.jpg: Skipping, found more recently modified local copy (use --force to force download)
0067dc3eab0b3c3ef0439477624d85d6.jpg: Skipping, found more recently modified local copy (use --force to force download)
00290d3e1fd226ba27a8ce248ce85.jpg: Skipping, found more recently modified local copy (use --force to force download)
003df8b8a8b0524b1d920b6c145f19.jpg: Skipping, found more recently modified local copy (use --force to force download)
000bec180eb18c7604dcecc8f0db0a07.jpg: Skipping, found more recently modified local copy (use --force to force download)
0021f3cbe3235ef1d7cde7f7538e682.jpg: Skipping, found more recently modified local copy (use --force to force download)
001cdf01b096e08d799e5f12d41937.jpg: Skipping, found more recently modified local copy (use --force to force download)
0071f9a78eba2aeb558afea3a51c489.jpg: Skipping, found more recently modified local copy (use --force to force download)
0042188c895a2f14ef64a918ed9c7b64.jpg: Skipping, found more recently modified local copy (use --force to force download)
00693b8bc2470375cc744a639f9c.jpg: Skipping, found more recently modified local copy (use --force to force download)
008887054b18ba3c7601792b8a453cc3.jpg: Skipping, found more recently modified local copy (use --force to force download)
004396df1acd0f1247b740ca2b14616e.jpg: Skipping, found more recently modified local copy (use --force to force download)
0075dc49dab4024d12f4fe6707.jpg: Skipping, found more recently modified local copy (use --force to force download)
002211c81b498ef88e1b40b9abf84e1d.jpg: Skipping, found more recently modified local copy (use --force to force download)
007b8a07882822475a4ce6581.jpg: Skipping, found more recently modified local copy (use --force to force download)
00792e341f3c6eb3363e415d0.jpg: Skipping, found more recently modified local copy (use --force to force download)
007f9a78eba2aeb558afea3a51c489.jpg: Skipping, found more recently modified local copy (use --force to force download)
007b5a16db9d9f9d7ad39982703e429.jpg: Skipping, found more recently modified local copy (use --force to force download)
001513dfcb2f1af8c2cc4d8bbaba97.jpg: Skipping, found more recently modified local copy (use --force to force download)
labels.csv: Skipping, found more recently modified local copy (use --force to force download)
sample_submission.csv.zip: Skipping, found more recently modified local copy (use --force to force download)
```



```
# 8.8 Stanford Dog Dataset을 Kaggle에서 불러오기
import tensorflow as tf
# 2020.02.01 현재 kaggle의 Stanford Dog Dataset 파일 구조가 변경되었습니다.
# kaggle API를 사용하는 대신에 아래 링크에서 파일을 직접 받아오도록 수정되었습니다.
tf.keras.utils.get_file('/content/labels.csv', 'http://bit.ly/2GDxsYS')
tf.keras.utils.get_file('/content/sample_submission.csv', 'http://bit.ly/2GGnMNd')
tf.keras.utils.get_file('/content/train.zip', 'http://bit.ly/31nlvel')
tf.keras.utils.get_file('/content/test.zip', 'http://bit.ly/2GHEsn0')

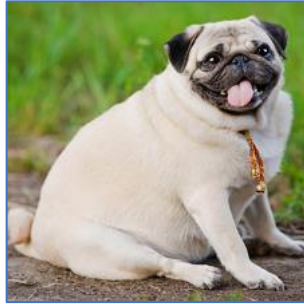
import os
os.environ['KAGGLE_USERNAME'] = 'ehs2019305059' # 독자의 캐글 ID
os.environ['KAGGLE_KEY'] = '914467037bd74512956e403c5b1d0abc' # 독자의 캐글 API Token
!kaggle competitions download -c dog-breed-identification

Downloading data from http://bit.ly/2GDxsYS
483328/482063 [=====] - 0s 0us/step
Downloading data from http://bit.ly/2GGnMNd
25206784/25200295 [=====] - 1s 0us/step
Downloading data from http://bit.ly/31nlvel
361357312/361353329 [=====] - 8s 0us/step
Downloading data from http://bit.ly/2GHEsn0
362848256/362841195 [=====] - 8s 0us/step
```

교재코드로 하면 train.zip, test.zip
이 압축이 풀린상태로 다운되고 모
든 데이터가 다운로드가 안되어
교재 출판사 github에 올라와있는
수정코드로 진행하니 정상적으로
다운되어 수정코드로 과제함.

②5장의 개 이미지를 만든다.(사진을 찍거나 인터넷 활용)

*테스트 이미지 5장을 제출



구글에서 검색해 캡처후 그림판에서 224 x 224픽셀 크기로 조정함.

③5장의 테스트 이미지로 예제8.13을 수행하여 [OUT]을 제출

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

Mounted at /content/drive

```
import PIL.Image as Image
import matplotlib.pyplot as plt

lst = ['/content/drive/MyDrive/Colab Notebooks/레포트4/dingo.PNG',
       '/content/drive/MyDrive/Colab Notebooks/레포트4/kelpie.PNG',
       '/content/drive/MyDrive/Colab Notebooks/레포트4/malamute.PNG',
       '/content/drive/MyDrive/Colab Notebooks/레포트4/pug.PNG',
       '/content/drive/MyDrive/Colab Notebooks/레포트4/samoyed.PNG']
name=['dingo', 'kelpie', 'malamute', 'pug', 'samoyed']
plt.figure(figsize=(10,10))
for c in range(5):
    plt.subplot(3,3,c+1)
    plt.imshow(plt.imread(lst[c]))
    plt.title(name[c])
    plt.axis('off')
plt.show()
```

dingo



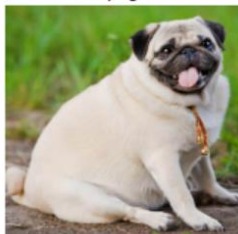
kelpie



malamute



pug



samoyed



④예제8.21, 8.22의 전이학습을 실행하고 5장의 테스트 이미지로 evaluate

```
!unzip train.zip
!unzip labels.csv.zip
```

```
import pandas as pd
label_text = pd.read_csv('labels.csv')
```

```
import os
import shutil

os.mkdir('/content/train_sub')

for i in range(len(label_text)):
    if os.path.exists('/content/train_sub/' + label_text.loc[i]['breed']) == False:
        os.mkdir('/content/train_sub/' + label_text.loc[i]['breed'])
        shutil.copy('/content/train/' + label_text.loc[i]['id'] + '.jpg', '/content/train_sub/' + label_text.loc[i]['breed'])

# 8.25 ImageDataGenerator를 이용한 train/validation 데이터 분리, Image Augmentation
from tensorflow.python.keras.preprocessing.image import ImageDataGenerator
from keras.applications.inception_resnet_v2 import preprocess_input

image_size = 224 # 이미지 사이즈가 299에서 224로 바뀌었습니다.
batch_size = 32

train_datagen = ImageDataGenerator(rescale=1./255., horizontal_flip=True, shear_range=0.2,
    zoom_range=0.2, width_shift_range=0.2, height_shift_range=0.2, validation_split=0.25)
valid_datagen = ImageDataGenerator(rescale=1./255., validation_split=0.25)

train_generator = train_datagen.flow_from_directory(directory="/content/train_sub/",
    subset="training", batch_size=batch_size, seed=42,
    shuffle=True, class_mode="categorical", target_size=(image_size, image_size))
valid_generator = valid_datagen.flow_from_directory(directory="/content/train_sub/",
    subset="validation", batch_size=1, seed=42, shuffle=True,
    class_mode="categorical", target_size=(image_size, image_size))

Found 7718 images belonging to 120 classes.
Found 2504 images belonging to 120 classes.
```

```
import numpy as np

unique_Y = label_text['breed'].unique().tolist()
train_Y = [unique_Y.index(breed) for breed in label_text['breed']]
train_Y = np.array(train_Y)
```

```
# 8.21 Dogs Dataset 학습을 위한 Transfer Learning 모델 정의
from tensorflow.keras.applications import MobileNetV2
mobilev2 = MobileNetV2()

x = mobilev2.layers[-2].output
predictions = tf.keras.layers.Dense(120, activation='softmax')(x)
model = tf.keras.Model(inputs=mobilev2.input, outputs=predictions)

# 뒤에서 20개까지의 레이어는 훈련 가능, 나머지는 가중치 고정
for layer in model.layers[:-20]:
    layer.trainable = False
for layer in model.layers[-20:]:
    layer.trainable = True

# model.compile(optimizer='sgd', loss='sparse_categorical_crossentropy', metrics=['accuracy'])
model.compile(optimizer='sgd', loss='categorical_crossentropy', metrics=['accuracy'])
# 라벨이 원-핫 인코딩을 사용하기 때문에 sparse가 아닌 categorical_crossentropy를 사용합니다.
model.summary()
```

Downloading data from https://storage.googleapis.com/tensorflow/keras-applications/mobilenet_v2/mobilenet_v2_weights_tf_dim_ordering_tf_kernels_1.0_224.h5
14540800/14536120 [=====] - 0s 0us/step
Model: "functional_1"

Layer (type)	Output Shape	Param #	Connected to
input_1 (InputLayer)	(None, 224, 224, 3)	0	
Conv1_pad (ZeroPadding2D)	(None, 225, 225, 3)	0	input_1[0][0]
Conv1 (Conv2D)	(None, 112, 112, 32)	864	Conv1_pad[0][0]

(생략)

Conv_1 (Conv2D)	(None, 7, 7, 1280)	409600	block_16_project_BN[0][0]
Conv_1_bn (BatchNormalization)	(None, 7, 7, 1280)	5120	Conv_1[0][0]
out_relu (ReLU)	(None, 7, 7, 1280)	0	Conv_1_bn[0][0]
global_average_pooling2d (GlobalAveragePooling2D)	(None, 1280)	0	out_relu[0][0]
dense (Dense)	(None, 120)	153720	global_average_pooling2d[0][0]

Total params: 2,411,704
Trainable params: 1,204,280
Non-trainable params: 1,207,424

```
steps_per_epoch = int(7718/32)
history = model.fit_generator(train_generator, validation_data=valid_generator, epochs=10, steps_per_epoch=steps_per_epoch)

import matplotlib.pyplot as plt
plt.figure(figsize=(12, 4))

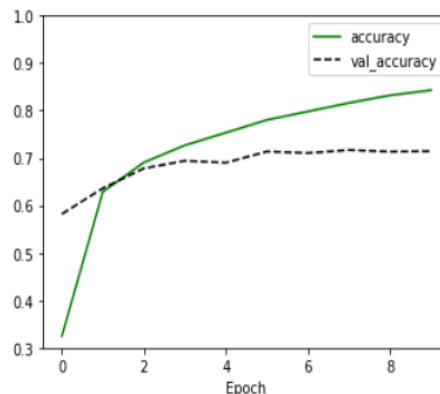
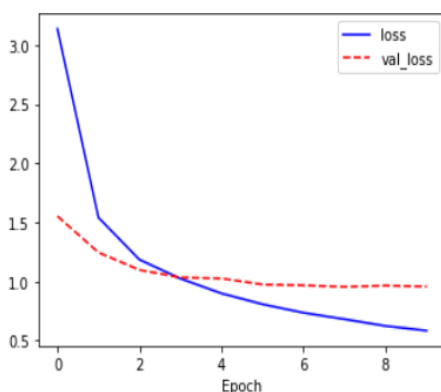
plt.subplot(1, 2, 1)
plt.plot(history.history['loss'], 'b-', label='loss')
plt.plot(history.history['val_loss'], 'r--', label='val_loss')
plt.xlabel('Epoch')
plt.legend()

plt.subplot(1, 2, 2)
plt.plot(history.history['accuracy'], 'g-', label='accuracy')
plt.plot(history.history['val_accuracy'], 'k--', label='val_accuracy')
plt.xlabel('Epoch')
plt.ylim(0.3, 1)
plt.legend()

plt.show()
```

WARNING:tensorflow:From <ipython-input-11-9ceee29e6bae>:2: Model.fit_generator (from tensorflow.python.keras.engine.training) is deprecated. Instructions for updating:
Please use Model.fit, which supports generators.

```
Epoch 1/10
241/241 [=====] - 121s 502ms/step - loss: 3.1364 - accuracy: 0.3241 - val_loss: 1.5528 - val_accuracy: 0.5819
Epoch 2/10
241/241 [=====] - 111s 463ms/step - loss: 1.5407 - accuracy: 0.6288 - val_loss: 1.2455 - val_accuracy: 0.6362
Epoch 3/10
241/241 [=====] - 112s 463ms/step - loss: 1.1849 - accuracy: 0.6906 - val_loss: 1.0977 - val_accuracy: 0.6781
Epoch 4/10
241/241 [=====] - 112s 467ms/step - loss: 1.0241 - accuracy: 0.7268 - val_loss: 1.0340 - val_accuracy: 0.6941
Epoch 5/10
241/241 [=====] - 115s 476ms/step - loss: 0.8994 - accuracy: 0.7534 - val_loss: 1.0259 - val_accuracy: 0.6901
Epoch 6/10
241/241 [=====] - 116s 480ms/step - loss: 0.8078 - accuracy: 0.7802 - val_loss: 0.9746 - val_accuracy: 0.7137
Epoch 7/10
241/241 [=====] - 116s 483ms/step - loss: 0.7355 - accuracy: 0.7981 - val_loss: 0.9681 - val_accuracy: 0.7105
Epoch 8/10
241/241 [=====] - 118s 489ms/step - loss: 0.6812 - accuracy: 0.8160 - val_loss: 0.9543 - val_accuracy: 0.7169
Epoch 9/10
241/241 [=====] - 118s 490ms/step - loss: 0.6238 - accuracy: 0.8320 - val_loss: 0.9658 - val_accuracy: 0.7133
Epoch 10/10
241/241 [=====] - 118s 492ms/step - loss: 0.5847 - accuracy: 0.8430 - val_loss: 0.9571 - val_accuracy: 0.7145
```



```
▶ unique_sorted_Y = sorted(unique_Y)
```

```
▶ import cv2

plt.figure(figsize=(16,16))

def softmax(x):
    e_x = np.exp(x - np.max(x))
    return e_x / e_x.sum(axis=0)

all_image_paths=[ '/content/drive/MyDrive/Colab Notebooks/레포트4/dingo.PNG',
                  '/content/drive/MyDrive/Colab Notebooks/레포트4/kelpie.PNG',
                  '/content/drive/MyDrive/Colab Notebooks/레포트4/malamute.PNG',
                  '/content/drive/MyDrive/Colab Notebooks/레포트4/pug.PNG',
                  '/content/drive/MyDrive/Colab Notebooks/레포트4/samoyed.PNG']
name = ['dingo', 'kelpie', 'malamute', 'pug', 'samoyed']
for c in range(5):
    image_path = all_image_paths[c]
    idx = unique_sorted_Y.index(name[c])
    # 이미지 표시
    plt.subplot(5,2,c*2+1)
    plt.imshow(plt.imread(image_path))
    plt.title(name[c])
    plt.axis('off')

    # 예측값 표시
    plt.subplot(5,2,c*2+2)
    img = cv2.imread(image_path)
    img = cv2.resize(img, dsize=(224, 224))
    img = img / 255.0
    img = np.expand_dims(img, axis=0)

    prediction = model.predict(img)[0]

    top_5_predict = prediction.argsort()[::-1][:5]
    labels = [unique_sorted_Y[index] for index in top_5_predict]
    color = ['gray']*5
    if idx in top_5_predict:
        color[top_5_predict.tolist().index(idx)]= 'green'
    color = color[::-1]
    plt.barh(range(5), prediction[top_5_predict][::-1] * 100, color=color)
    plt.yticks(range(5), labels[::-1])
```




dingo



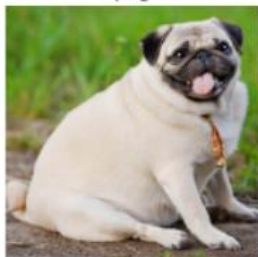
kelpie



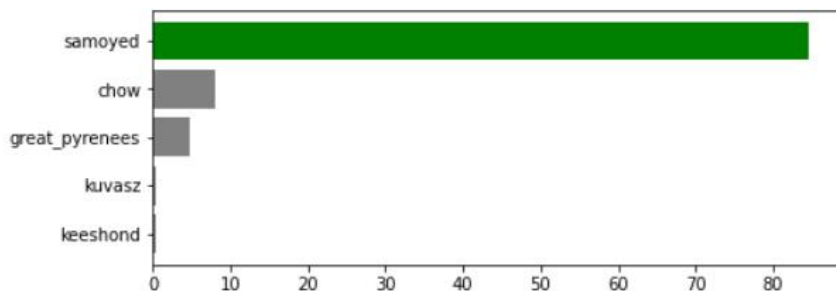
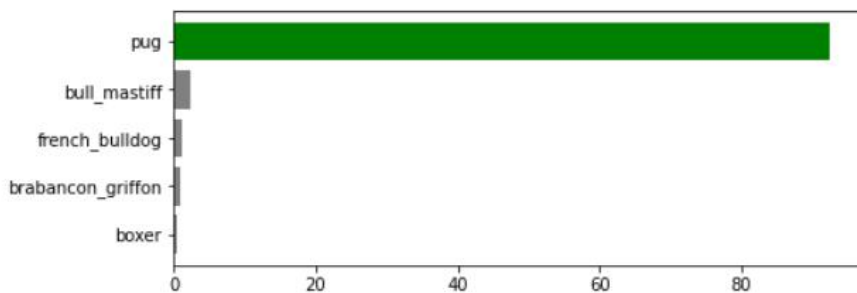
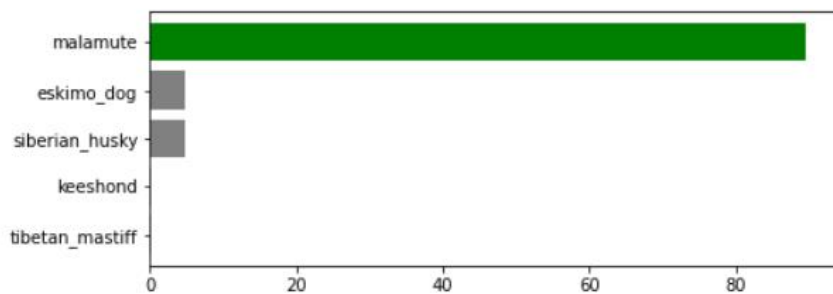
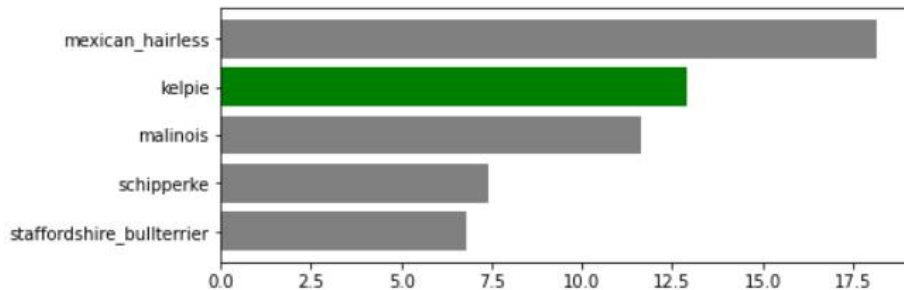
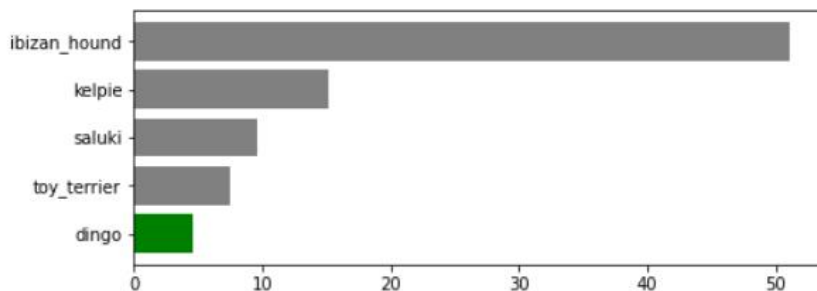
malamute



pug



samoyed



문제 4> 교재 6장~8장 코드에서 사용하는 함수와 관련한 내용이다.

다음 함수의 기능과 함수파라미터(교재 코드에서 사용한)를 설명하시오.

①예제6.14 image_generator.flow()

(함수기능)

데이터 및 레이블 어레이를 사용하고, 증강 데이터 배치 생성. 실제로 보강된 이미지 생성. Iterator라는 객체 만들. flow()함수는 실제로 보강된 이미지를 생성.

(교재 코드)

```
x_augmented = image_generator.flow(np.tile(train_X[0].reshape(28*28), 100).reshape(-1, 28, 28, 1),  
np.zeros(augment_size), batch_size=augment_size, shuffle=False).next()[0]
```

(함수파라미터)

np.tile(train_X[0].reshape(28*28), 100).reshape(-1, 28, 28, 1) : 데이터 입력. 튜플의 경우, 첫 번째 요소는 이미지를 포함하고 두 번째 요소는 출력에 전달되는 또 다른 numpy 배열 또는 numpy 배열 목록을 포함해야 한다.

np.zeros(augment_size) : 라벨

batch_size=augment_size : 한번에 생성할 이미지의 양

shuffle=False : 섞기 유무. False는 섞지 않는다는 의미.

②예제6.15 np.random.randint()

(함수기능)

지정된 수 사이에서 랜덤한 값을 생성한다.

(교재코드)

```
randidx = np.random.randint(train_X.shape[0], size=augment_size)
```

(함수파라미터)

0부터 train_X.shape[0] - 1 즉 60000-1=59,999 범위의 정수를 size = augment_size이므로 augment_size개수 만큼 랜덤한 정수를 생성한다. 이때 중복 가능.

③예제6.16 tf.keras.layers.Conv2D()

(함수기능)

텐서 출력물을 생산하기 위해 레이어 입력과 함께 컨볼루션 커널을 만든다. 즉 2D에 대해서 컨볼루션 연산을 하는 레이어이다. 이 레이어를 모델의 첫 번째 레이어로 사용하는 경우, 128x128 RGB 그림에 대해 키워드 인수 input_shape 제공해야함.

(함수파라미터)

kernel_size = (3,3) : 필터행렬의 크기 앞의숫자는 높이, 뒤의숫자는 너비, 숫자하나만 쓸경우 높이,너비 동일한값
strides=(2,2) : 필터가 계산과정에서 한 스텝마다 이동하는 크기. 기본값은 (1,1)

padding='valid' or padding='same' : 컨볼루션 연산 전에 입력이미지 주변에 빈값을 넣을지 지정하는 옵션. 'valid'는 빈값을 사용하지 않고, 'same'은 빈값을 넣어서 출력이미지의 크기를 입력과 같도록 보존함.

filters=16 : 필터의 개수.

④예제7.2 list(map())

(함수기능)

우선 map함수는 특정 범위내에 있는 집합형태의 것들을 원하는 하나의 것으로 바꿔주는 기능을 하고, list함수는 어떤 것들을 리스트로 만들어주는 기능을 한다. 그래서 list(map())은 map함수를 통해 어떤 집합형태의 원소들을 바꾼 후 그것을 리스트로 변환하는 기능을 한다.

(교재코드)

```
X.append(list(map(lambda c: [c/10], lst)))
```

(함수파라미터)

lst에 있는 원소들을 하나씩 꺼낸다. 그 원소를 c라고 할 때 c/10하고, 그것들을 list함수를 통해서 리스트형태로 변환된다.

⑤예제7.21 int(row.split())

(함수기능)

먼저 int함수는 매개변수를 정수로 바꿔주는 역할을 한다. split함수는 매개변수에 있는 것을 기준으로 문자열을 나누는 기능을 한다.

(교재코드)

```
int(row.split('Wt')[2]) : row를 'Wt'(탭문자)로 나눈 리스트에서 2번째인덱스(실제로는 3번째 원소)를 정수로 변환
```

⑥예제7.22 re.sub()

(함수기능)

re는 모듈이다. 사용하기위해서 import re를 해줘야 한다. re.sub('찾을문자열', '변경할문자열', 대상)이다. 대상이되는 문자열에서 '찾을문자열'이 있으면, 그것을 '변경할문자열'로 변경해준다음에 변경된문자열을 반환한다.

(교재코드)

```
string = re.sub(r"!", " ! ", string) : string 문자열에 "!"이 발견되면 " ! "로 변경후 반환해 string에 저장한다.
```

⑦예제7.24 append()

(함수기능)

리스트 마지막에 원소를 추가한다.

(교재코드)

```
sentence_new.append([word[:5] for word in sentence][:25]) : append함수 매개변수로 들어있는 [word[:5] for word in sentence][:25]을 리스트sentence_new 마지막에 원소로 추가한다.
```

⑧예제7.25 tokenizer.fit_on_texts()

(함수기능)

tokenizer.fit_on_texts()함수는 Tokenizer에 데이터를 실제로 입력한다.

(교재코드)

tokenizer.fit_on_texts(sentences) : sentences리스트 원소 하나씩 읽어 Tokenizer에 데이터를 실제로 입력한다.

⑨예제7.25 tokenizer.texts_to_sequence()

(함수기능)

tokenizer.texts_to_sequence() 함수는 문장을 입력받아 숫자로 반환한다.

(교재코드)

train_X = tokenizer.text_to_sequences(sentences) : sentences리스트를 입력받아 train_X에 숫자로 반환한다.

⑩예제8.3 iterdir()

(함수기능)

iterdir함수를 사용해 지정한 폴더 안에 있는 파일과 하위 디렉토리 이름을 취득할 수 있다.

(교재코드)

data_root.iterdir() : data_root폴더 안에 있는 하위 디렉토리를 순회하며 디렉토리의 경로를 취득한다.

⑪예제8.4 f.read().split()

(함수기능)

f.read()는 파일 f를 읽어온다. 그리고 split()함수는 매개변수 안에 있는 문자열을 기준으로 나눠 리스트로 반환한다.

(교재코드)

f.read().split('\n')[:-1] : 파일 f를 읽어들이면서 \n을 기준으로 나눈다. 그리고 마지막에[:-1]을 함으로써 '\n'을 제외한 문자열만을 반환해 리스트형태로 만든다.

⑫예제8.6 cv2.imread()

(함수기능)

모듈 cv2에 있는 함수로 import cv2를 해야 사용가능.

이미지 파일을 flag값에 따라 읽는 함수이다. cv2.imread(filename, flag)

(교재코드)

img = cv2.imread(image_path) : image_path이름을 가진 이미지파일을 Default flag인 Color로 읽어들이어 img에 반환한다.