CNN과 자연어 처리를 활용한 신발 분류 및 태그 추천

20.11.18 - 20.11.23

김성운 김홍구 조종운 윤채영

Contents,

1. 프로젝트 개요

〈아이디어〉

신발 이미지를 대입하면 '브랜드', '종류 ' 를
예측하고 어울리는 태그를 추천해주는 모델

〈사용 기술〉

•파이썬 웹 크롤링(Selenium, BeautifulSoup)
•CNN 모델링 (Conv2D, MaxPooling2D, Dense)
•자연어 처리 (Mecap, Okt, hanspell, konlpy)

-이 신발의 브랜드는 ' 나이키 ' 입니다.
-이 신발의 종류는 ' 운동화 ' 입니다.

[추천태그]

#편안 #이쁨 #만족 #클래식 #기본 #무난

2. 이미지 모델링 (수집)

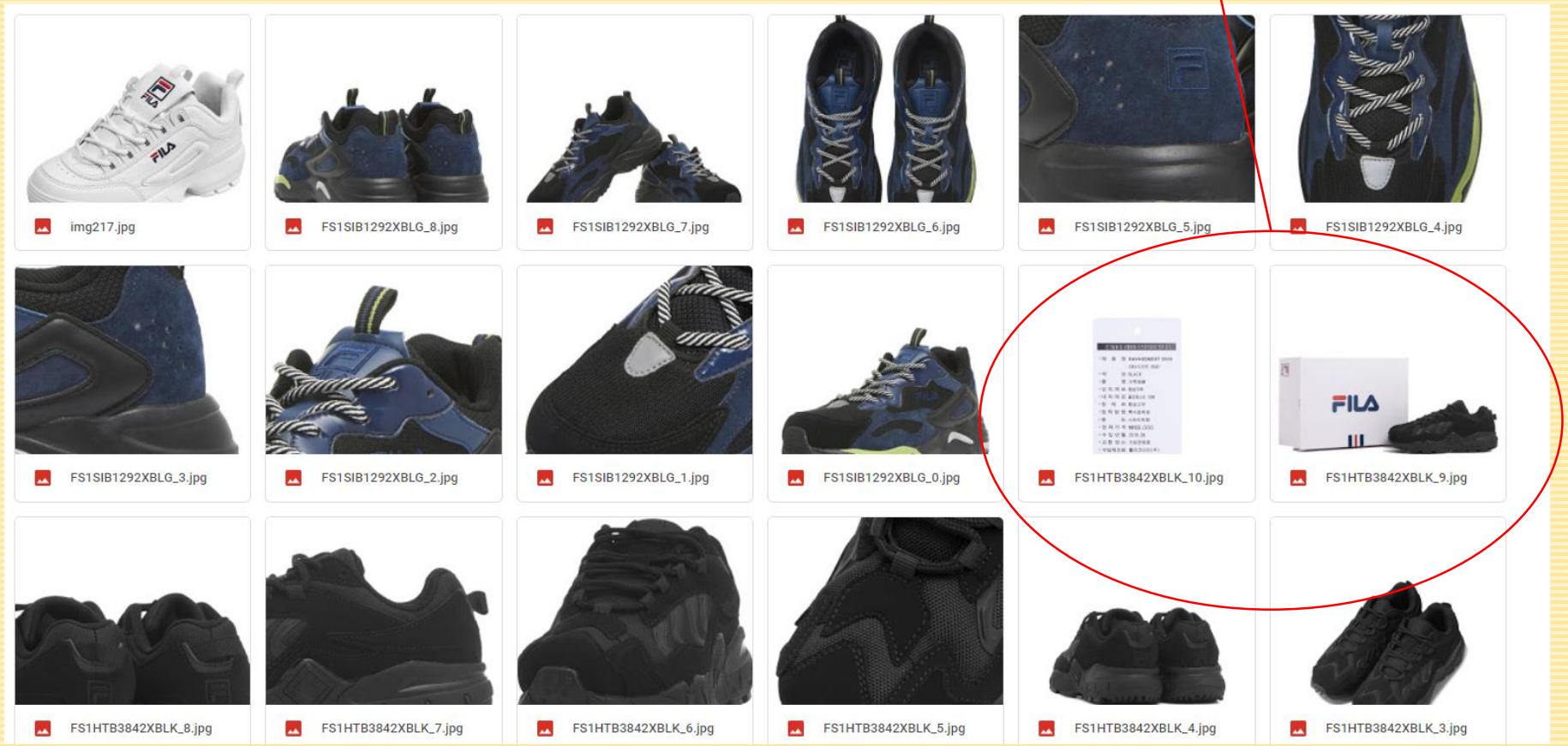
- 이미지 수집 출처 : 무신사 스토어 “https://store.musinsa.com/app/”

```
def crawling_canvas(driver):

    canvas_data = []

    canvas_btn = driver.find_elements_by_xpath('//*[@id="ui-id-16"]/ul[1]/li[1]/a')[0]
    canvas_btn.click()

    driver.implicitly_wait(2)

    ### 브랜드 체크
    brand_len = len(driver.find_elements_by_xpath('//*[@id="best_brand_list"]/dl/dd/ul/li'))
    for i in range(1, brand_len):
        brand = WebDriverWait(driver, 5).until(EC.presence_of_element_located((By.XPATH, f'//*[@id="best_brand_list"]/dl/dd/ul/li[{i}]')))
        if int(brand.get_attribute('data-count')) > 100:
            brand.click()

            driver.implicitly_wait(1)

    # 크롤링할 총 페이지 갯수 추출
    max_page_cnt = int(driver.find_elements_by_xpath('//*[@id="goods_list"]/div[2]/div[4]/span/span[1]')[0].text)
    # 현재 페이지
    count = 1
    # 앵커 갯수
    now_total_page = len(driver.find_elements_by_xpath('//*[@id="goods_list"]/div[2]/div[1]/div/div/a'))

    while True:
        # 크롤링을 위한 작업들
        # 현재 페이지의 상품 갯수를 가져온다.
        max_img_cnt = len(driver.find_elements_by_name('goods_link'))
```

```python
# 크롤링할 최대 페이지 갯수
max_page_cnt = int(driver.find_elements_by_xpath('//*[@id="goods_list"]/div[2]/div[4]/span/span[1]')[0].text)
# 현재 페이지 넘버
count = 1
# 현재 페이지 기준 앵커의 갯수
now_total_page = len(driver.find_elements_by_xpath('//*[@id="goods_list"]/div[2]/div[1]/div/div/a'))

while True:
    # 크롤링 작업들
    # 현재 페이지 내 이미지 갯수
    max_img_cnt = len(driver.find_elements_by_name('goods_link'))

    # 이미지 갯수만큼 반복문을 돌면서..
    for i in range(1, max_img_cnt, 2):
        # i번째 신발 객체 가져오기
        shoe_img = driver.find_elements_by_name('goods_link')[i]

        # 신발 이미지를 클릭해서, 세부페이지로 접근
        shoe_img.send_keys(Keys.ENTER)

        driver.implicitly_wait(2)

        # 세부페이지 내 신발에대한 상세 이미지 갯수 세기
        img_cnt = len(driver.find_elements_by_xpath('//*[@id="detail_thumb"]/ul/li'))

        # 상세이미지 링크를 담을 배열
        img_src = []
        # 신발의 브랜드
        img_brand = ""
        # 신발의 카테고리
        img_category = ""
        # 상세이미지 갯수만큼 반복문을 돌면서..
        for i in range(1, img_cnt):
            # 선택한 객체가 이미지가 아닌경우 반복문 중단
            if len(driver.find_elements_by_xpath(f'//*[@id="thum_{i}"]')) == 0:
                break
            else:
                # 이미지 가져오고 클릭 ( 상세이미지를 클릭하면, 이미지가 커진다 )
                detail_img.WebDriverWait(driver, 10).until(EC.presence_of_element_located((By.XPATH, 'f//*[@id="thum_{i}"]')))
                detail_img = driver.find_elements_by_xpath(f'//*[@id="thum_{i}"]')[0]
```

영수증, 포장박스 등 비정상적 이미지

〈크롤링 된 이미지 폴더〉



〈데이터 수집 과정에서 발생한 문제점〉

- ‘반스-Canvas’ 카테고리의 신발이 압도적으로 많음
- 다리 나온 사진, 영수증 사진, 신발의 일부만 나온 사진 등 불필요한 사진 존재

〈해결 방안〉

- 데이터의 개수가 적은 카테고리의 이미지 -> Generator를 이용해서 Image Augmentation 기법 활용 -> 모델에 과적합됨 -> Generator 사용x
- 정규화 방법 이용 -> Min-Max Scaler 방법 이용 -> 성능이 좋아짐 -> 사용o

2. 이미지 모델링 (전처리)

〈전처리 과정〉

- 사진 용량의 감소를 위해 500x500 -> 64x64 사이즈 조절
- 범주형 자료인 카테고리 -> 원 핫 인코딩을 통한 카테고리 라벨링
- Min-Max Scaler를 이용한 정규화 -> 원본 이미지의 픽셀값 0~255 -> 0~1
- 데이터의 shape, pixel, 차원수, 이미지 확인
- 이미지 데이터를 일일이 옮기기 힘드니 numpy 배열 파일로(.npy) 저장

```
# 사진 용량 감소를 위해서 크기 조정
image_w = 64
image_h = 64

X = []
Y = []

# 카테고리 인덱스 라벨링
for idx, cate in enumerate(categories):
  # 카테고리별로 돌면서 0으로 초기화, categories의 인덱스 위치에 1을 넣어준다(OneHotEncoding)
  label = [0 for i in range(nb_classes)]
  label[idx] = 1

  image_dir = imagePath + '/' + cate + '/'
  files = glob.glob(image_dir+'/*.*')
  img_num = 0

# 파일 별로 64 * 64의 이미지로 줄어주고 255로 나누어 정규화를 진행해 라벨별로 categorie에 저장 한다.
  for top, dir, f in os.walk(image_dir):
        for filename in f:
            print(image_dir+filename)
            img = cv2.imread(image_dir+filename)
            img = cv2.resize(img, None, fx=image_w/img.shape[1], fy=image_h/img.shape[0])
            X.append(img/255)
            Y.append(label)

X = np.array(X)
Y = np.array(Y)

X_train, X_test, Y_train, Y_test = train_test_split(X,Y)
xy = (X_train, X_test, Y_train, Y_test)
```

```
[ ]  # 전체 데이터 갯수와 shape 확인하기
     print(X_train.shape)
     print(X_train.shape[0])
```

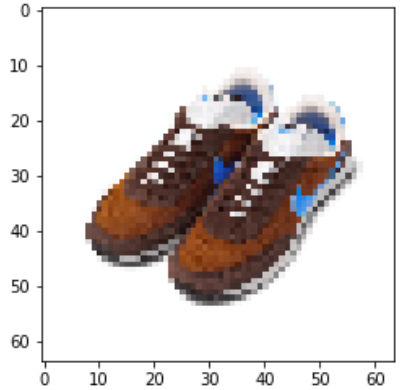
```
(11208, 64, 64, 3)
11208
```

```
[ ]  # label[1] = 나이키-running 값
     Y_train[6]
```

```
array([0, 1, 0, 0, 0, 0, 0, 0, 0, 0])
```

```
[ ]  # 실제 나이키running 확인하기
     plt.imshow(X_train[6])
```

```
<matplotlib.image.AxesImage at 0x7f714c7f2e10>
```




```
[ ]  print(len(X_train)) , print(len(X_test))
```

```
11208
3737
(None, None)
```

2. 이미지 모델링 (모델링)

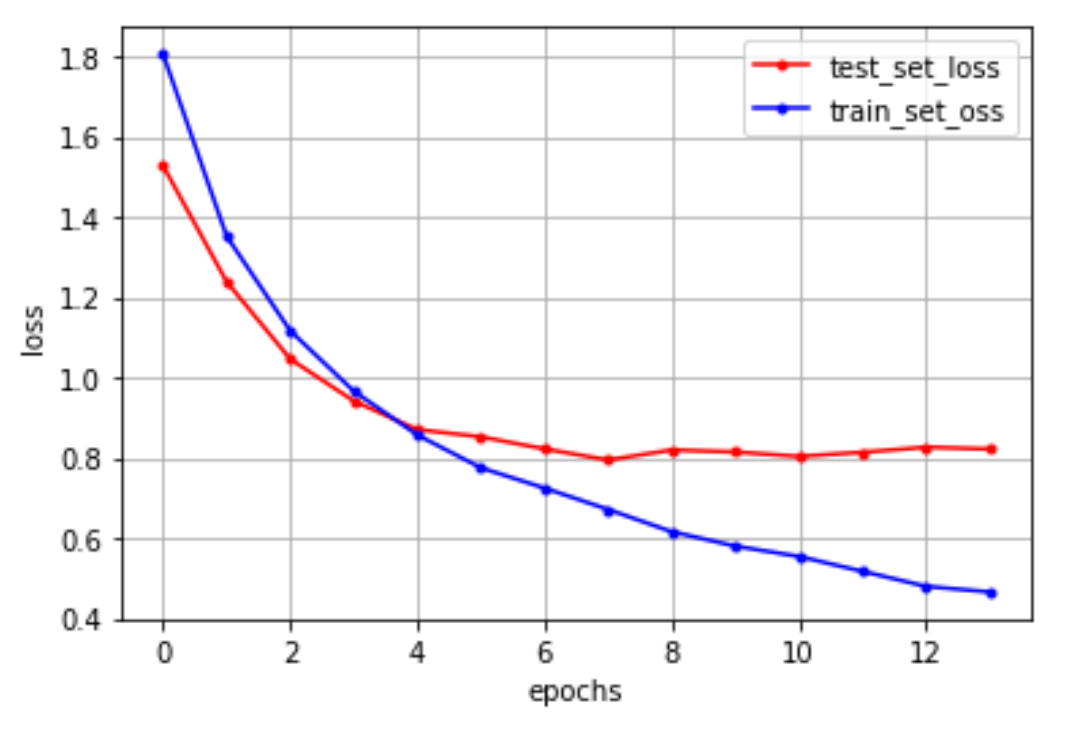
〈모델링 과정〉

- Conv2D, MaxPooling2D, Dropout, Flatten, Dense 레이어 사용하여 CNN 모델 설계
- Early Stopping 방법을 사용하여 6번의 진행동안 loss값이 갱신되지 않으면 훈련을 정지
- Accuracy, Loss 그래프 확인
- 성능이 가장 좋은 순간을 ModelCheckpoint를 통해 파일로(.h5) 저장
- 새로운 신발 이미지를 입력해 제대로 예측이 되는 지 확인

```python
# 모델 생성

#Sequence 클래스 생성
model = Sequential()

# 3*3 크기의 컨볼루션 레이어 32개의 필터를 생성 , input_shape 는 64,64,3 이니 [3] 을 설정
model.add(Conv2D(32, (3,3), padding="same", input_shape=X_train.shape[1:], activation='relu'))
#maxpooling을 통해 영역 셀값의 최댓값(주요값) 을 채택한다.
model.add(MaxPooling2D(pool_size=(2,2)))
model.add(Dropout(0.25))

model.add(Conv2D(64, (3,3), padding="same", activation='relu'))
model.add(MaxPooling2D(pool_size=(2,2)))
model.add(Dropout(0.25))

model.add(Conv2D(128, (3,3), padding="same", activation='relu'))
model.add(MaxPooling2D(pool_size=(2,2)))
model.add(Dropout(0.25))


#Fully Connection
#2D로 되어있는 임베딩을 1차원 단일 벡터로 변환함
model.add(Flatten())
# 인풋과 아웃풋 사이에 히든레이어를 둔다.
model.add(Dense(256, activation='relu'))
model.add(Dropout(0.25))
#노드의 수(카테고리의 수)
model.add(Dense(10, activation='softmax'))
# loss 함수로는 3개 이상의 label값, OneHotEncoding 되어있어 categorical_crossentropy사용
model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy'])
model_dir = './model'
# 모델 저장
if not os.path.exists(model_dir):
    os.mkdir(model_dir)

model_path = model_dir + '/multi_img_classification.model'
```

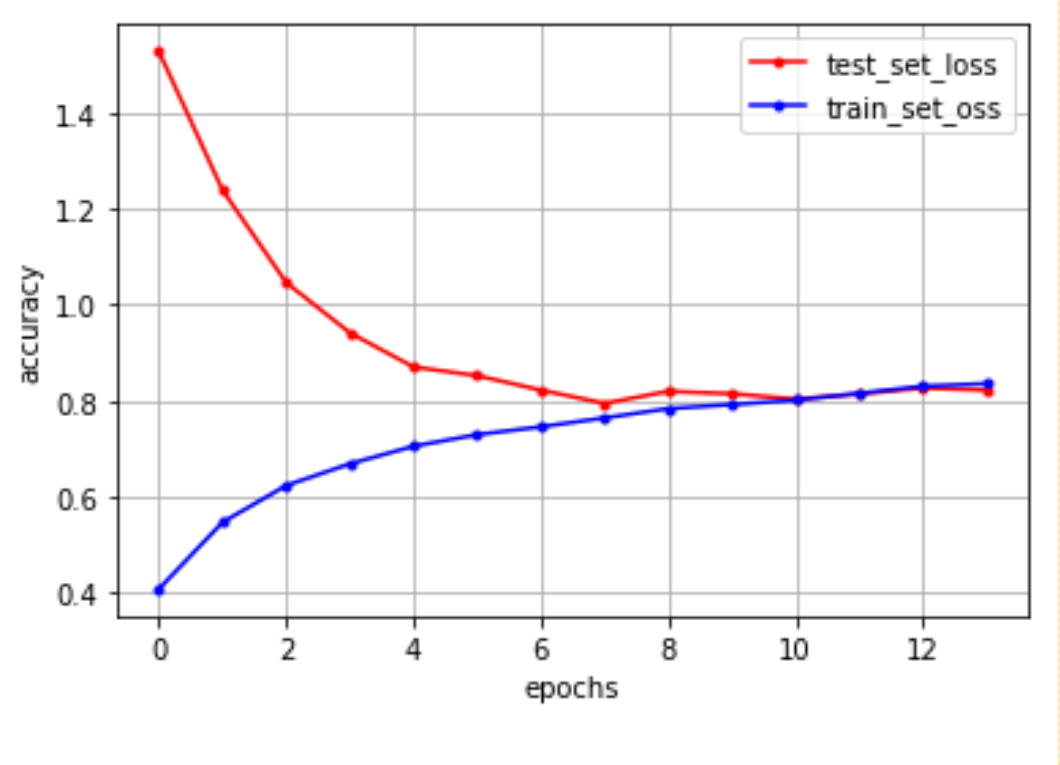
```
[ ]  model.summary()
```

```
Model: "sequential_4"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_38 (Conv2D)           (None, 64, 64, 32)        896
_________________________________________________________________
max_pooling2d_5 (MaxPooling2 (None, 32, 32, 32)        0
_________________________________________________________________
dropout_4 (Dropout)          (None, 32, 32, 32)        0
_________________________________________________________________
conv2d_39 (Conv2D)           (None, 32, 32, 64)        18496
_________________________________________________________________
max_pooling2d_6 (MaxPooling2 (None, 16, 16, 64)        0
_________________________________________________________________
dropout_5 (Dropout)          (None, 16, 16, 64)        0
_________________________________________________________________
conv2d_40 (Conv2D)           (None, 16, 16, 128)       73856
_________________________________________________________________
max_pooling2d_7 (MaxPooling2 (None, 8, 8, 128)         0
_________________________________________________________________
dropout_6 (Dropout)          (None, 8, 8, 128)         0
_________________________________________________________________
flatten_3 (Flatten)          (None, 8192)              0
_________________________________________________________________
dense_6 (Dense)              (None, 256)               2097408
_________________________________________________________________
dropout_7 (Dropout)          (None, 256)               0
_________________________________________________________________
dense_7 (Dense)              (None, 10)                2570
=================================================================
Total params: 2,193,226
Trainable params: 2,193,226
Non-trainable params: 0
_________________________________________________________________
```



Loss 그래프

Callback 함수를 이용해서 6번 이내에 loss값이 떨어지지 않으면 EarlyStopping하도록 했다.



Accuracy 그래프

갑자기 초반에 훅 떨어지는 이유는 불필요한 이미지 데이터가 많이 들어가 있기 때문에 초반에 loss값이 높은 것으로 판단하였다.

전이학습 시도

- 예측률을 높히기 위해 미리학습된 모델을 불러 사용
- 학습데이터에 적절하지 않은 사진이 많아서 로스율이 더 높아진것으로 예측 됐다.

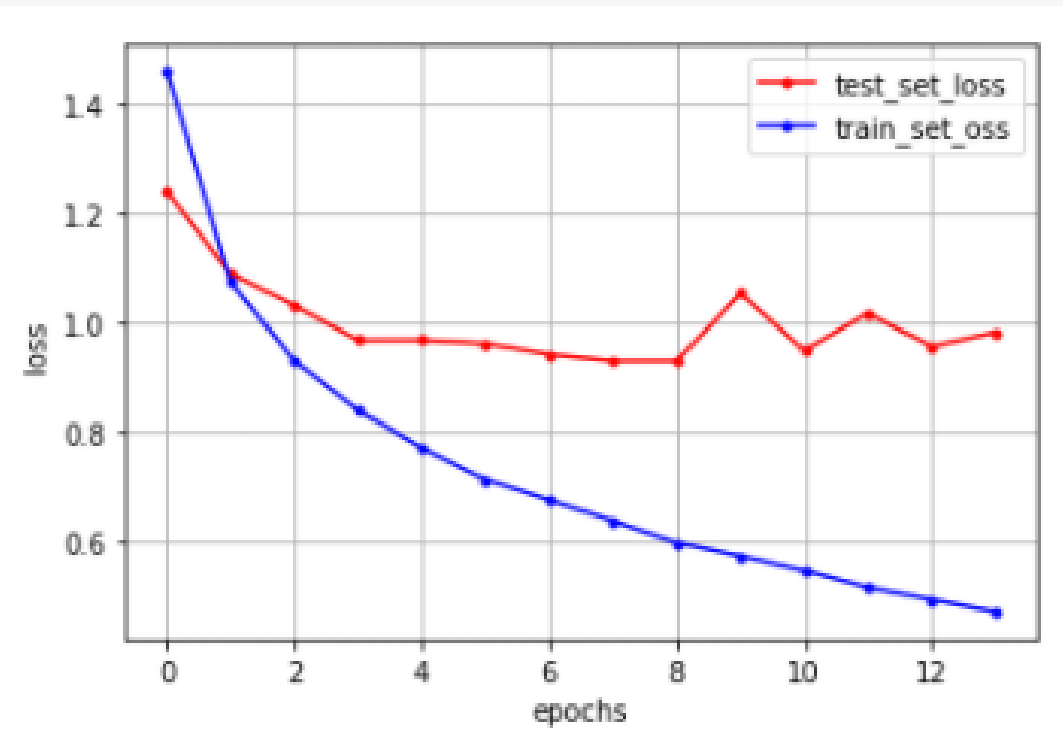
```
]  from keras.applications.vgg16 import VGG16
   #weigenet: 로딩할 가중치.  include_top : 출력 레이어를 포함할 것인지 여부, 개별 문제에 적합되어있으면 불필요
   transfer_model = VGG16(weights='imagenet', include_top=False, input_shape=(64,64,3))
   transfer_model.trainable =False
   transfer_model.summary()
```

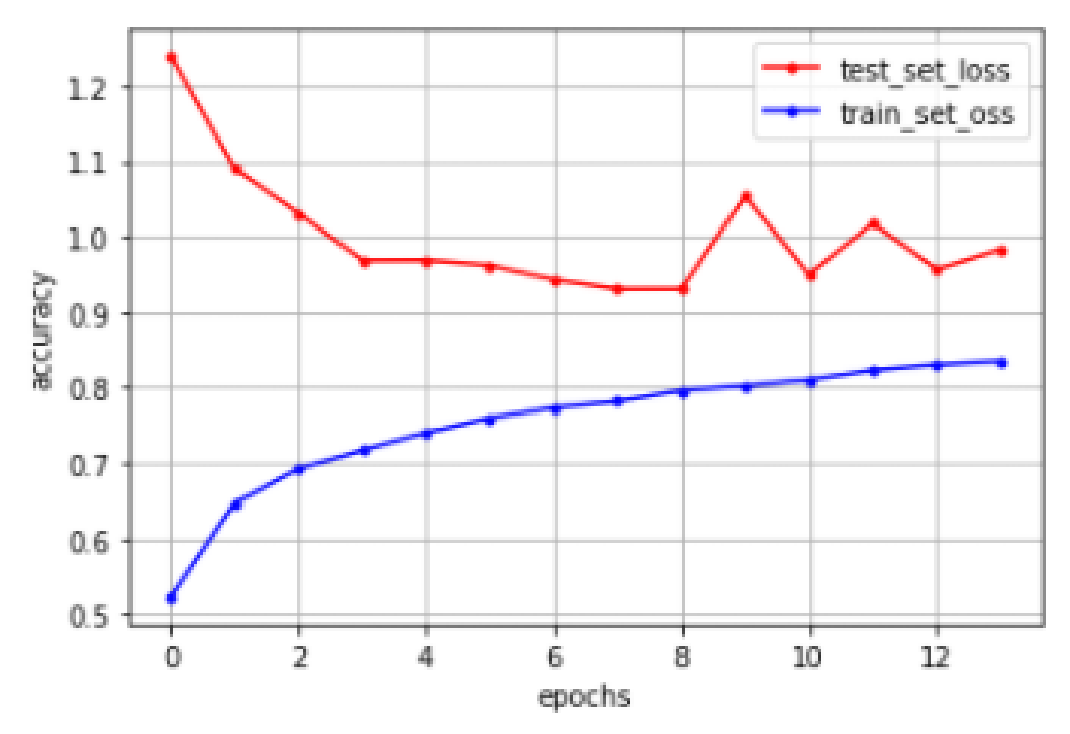
```
Model: "vgg16"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_2 (InputLayer)         [(None, 64, 64, 3)]       0
_________________________________________________________________
block1_conv1 (Conv2D)        (None, 64, 64, 64)        1792
_________________________________________________________________
block1_conv2 (Conv2D)        (None, 64, 64, 64)        36928
_________________________________________________________________
block1_pool (MaxPooling2D)   (None, 32, 32, 64)        0
_________________________________________________________________
block2_conv1 (Conv2D)        (None, 32, 32, 128)       73856
_________________________________________________________________
block2_conv2 (Conv2D)        (None, 32, 32, 128)       147584
_________________________________________________________________
block2_pool (MaxPooling2D)   (None, 16, 16, 128)       0
_________________________________________________________________
block3_conv1 (Conv2D)        (None, 16, 16, 256)       295168
_________________________________________________________________
block3_conv2 (Conv2D)        (None, 16, 16, 256)       590080
_________________________________________________________________
block3_conv3 (Conv2D)        (None, 16, 16, 256)       590080
_________________________________________________________________
block3_pool (MaxPooling2D)   (None, 8, 8, 256)         0
_________________________________________________________________
block4_conv1 (Conv2D)        (None, 8, 8, 512)         1180160
_________________________________________________________________
block4_conv2 (Conv2D)        (None, 8, 8, 512)         2359808
_________________________________________________________________
block4_conv3 (Conv2D)        (None, 8, 8, 512)         2359808
_________________________________________________________________
block4_pool (MaxPooling2D)   (None, 4, 4, 512)         0
_________________________________________________________________
block5_conv1 (Conv2D)        (None, 4, 4, 512)         2359808
_________________________________________________________________
block5_conv2 (Conv2D)        (None, 4, 4, 512)         2359808
_________________________________________________________________
block5_conv3 (Conv2D)        (None, 4, 4, 512)         2359808
_________________________________________________________________
block5_pool (MaxPooling2D)   (None, 2, 2, 512)         0
=================================================================
Total params: 14,714,688
Trainable params: 0
Non-trainable params: 14,714,688
_________________________________________________________________
```

```
from tensorflow.keras import models , optimizers
```

```
model2 = models.Sequential()
model2.add(transfer_model)
model2.add(Flatten())
model2.add(Dense(64,activation='relu'))
model2.add(Dense(10,activation='softmax'))
model2.summary()
```

```
Model: "sequential_2"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
vgg16 (Functional)           (None, 2, 2, 512)         14714688
_________________________________________________________________
flatten_2 (Flatten)          (None, 2048)              0
_________________________________________________________________
dense_4 (Dense)              (None, 64)                131136
_________________________________________________________________
dense_5 (Dense)              (None, 10)                650
=================================================================
Total params: 14,846,474
Trainable params: 131,786
Non-trainable params: 14,714,688
_________________________________________________________________
```





전이학습을 시도했으나 기존 모델보다 성능이 안 좋고 이상한 결과가 나와서 기존 모델 사용

3. 텍스트 수집

〈데이터 수집〉

무신사스토어 -> 스니커즈 & 신발 카테고리 -> 상품평순 정렬 -> 상위 20개 상품 -> 일반 후기 최대 10 페이지 크롤링

〈수집 과정에서 발생한 문제점〉

- 리뷰가 없는 신발 존재
- 리뷰 작성자가 동일한 경우 존재

〈해결 방안〉

- 리뷰가 없는 신발의 공식 브랜드 홈페이지에 가서 후기를 크롤링
- 텍스트 전처리 과정에서 중복 데이터를 제거하도록 코딩

〈분석 과정〉

1. 중복 데이터, 결측값, 특수기호, 이모티콘, 숫자 제거 (한글, 띄어쓰기만 남기기)

2. 띄어쓰기, 맞춤법 교정

3. 토큰화

- Mecab, Okt
- 다른 라이브러리를 사용하지 않은 이유 : 속도가 너무 느리고 성능도 Mecab, Okt와 비슷하기 때문
- 일반적인 토큰화 방법을 사용하지 않고, 각 라이브러리의 품사 홈페이지에 들어가서 명사, 형용사, 부사만 토큰화 하도록 품사의 형태를 지정

4. 불용어 제거

5. 토큰의 최대 길이 정하기

6. 단어 사전 만들기

- 단어의 빈도수 높은 순으로 나열

- 제일 많이 사용된 단어 10개를 해당 카테고리('브랜드-종류')의 추천 태그로 사용

7. 최종 데이터프레임에 카테고리와 추천 태그를 미리 넣어놓기

```
# 상품평순 맨 처음 5개의 상품 3페이지까지 크롤링한 정보로 테스트..
import pandas as pd
column_names=["review"]
file_path = '/content/drive/MyDrive/GoogleColab/MiniProject/txtFile_sample'
file_name = '반스-스니커즈'
file_format = '.txt'
data = pd.read_csv(file_path+'/'+file_name+file_format, delimiter = '₩t', header=None, names=column_names)
data
```

	review
0	좋아요 여자친구선물줬는데 긋이뻐요 추천드림
1	역시 반스에요 너무 이쁘고 어디에다 매치해도 어울려요
2	일단 신발이 너무이쁘고 착용감도 편안하고 무난하게 신기 편해요
3	컨버스는 있지만 반스가없던 나 오늘 드디어 반스님을 받았다 감사합니다 ^^
4	비교적만족스럽습니다. 자주이용할것같습니다
...	...
88	치노 팬츠랑 매치하니까 예뻐요 저렴하게 잘 산 것 같아요
89	적당하고 색도 이쁘고 나중에 또 사려고요 ㅎㅎ 좋아요
90	와이드한 팬츠 크롭된 팬츠 어디든 다 잘어울려서 정말 좋네여
91	예뻐요 좋습니다 배송도 빨랐어요 잘신겠습니다
92	이뽀염 맘에듭니당 생각했던거보다 색상도 이쁘고 사이즈도 딱 좋구 아주 맘에들어용

93 rows × 1 columns

특수기호, 이모티콘, 숫자 제거

- 한글, 띄어쓰기만 뽑아내자
- 정규식 이용

```
[ ] import re
    import pandas as pd
    only_korean_list = []
    for i in data.index:
      s = re.sub("[^ㄱ-ㅎㅏ-ㅣ가-힣 ]", "", str(data['review'][i]))
      data['review'][i] = s
    data
```

	review	label
0	좋아요 여자친구선물줬는데 긋이뻐요 추천드림	반스-스니커즈
1	역시 반스에요 너무 이쁘고 어디에다 매치해도 어울려요	반스-스니커즈
2	일단 신발이 너무이쁘고 착용감도 편안하고 무난하게 신기 편해요	반스-스니커즈
3	컨버스는 있지만 반스가없던 나 오늘 드디어 반스님을 받았다 감사합니다	반스-스니커즈
4	비교적만족스럽습니다 자주이용할것같습니다	반스-스니커즈
...	...	...
84	예전에 구매하여 잘 신었던 기억에 같은 사이즈로 샀는데 이번에는 좀 커요 하지만 ...	반스-스니커즈
85	기본적인 디자인인데 귀엽고 이뻐용 무난하게 자주신어요	반스-스니커즈
86	그냥 뭐 말이 필요있나요 클래식이죠 이거 하나쯤은 신발장에 무조건 있어야죠	반스-스니커즈
87	컨버스 로우 버리고 반스로 갈아탔는데 아주 찰떡입니다 반바지에 긴양말에 신어도 이쁘...	반스-스니커즈
88	치노 팬츠랑 매치하니까 예뻐요 저렴하게 잘 산 것 같아요	반스-스니커즈

78 rows × 2 columns

```python
# 예측 부분
for idx, i in enumerate(prediction):  # i -> [1, 0, 0, 0, 0, 0, 0, 0, 0, 0]의 형태
  # print(idx)
  pre_ans = i.argmax()  # 예측 레이블 -> 정수 한 글자
  pre_ans_str = ''
  if pre_ans == 0: pre_ans_str = categories[0]
  elif pre_ans == 1: pre_ans_str = categories[1]
  elif pre_ans == 2: pre_ans_str = categories[2]
  elif pre_ans == 3: pre_ans_str = categories[3]
  elif pre_ans == 4: pre_ans_str = categories[4]
  elif pre_ans == 5: pre_ans_str = categories[5]
  elif pre_ans == 6: pre_ans_str = categories[6]
  elif pre_ans == 7: pre_ans_str = categories[7]
  elif pre_ans == 8: pre_ans_str = categories[8]
  elif pre_ans == 9: pre_ans_str = categories[9]

  # if i[idx] >= 0.8:
  print("해당 {} 이미지는 {} 카테고리로 추정됩니다.".format(test_file_name_list[idx], pre_ans_str))
  predict_tag = TAG_df.loc[pre_ans_str].values
  print("또한, 추천 태그로는 {}들이 있습니다.".format(predict_tag))  # 예측한 '브랜드-종류'에 해당하
  print('')
```

	태그
Other-Sneakers	정 디자인 편하고 무난 가격 좋습니다 넓은 엄청 예뻐요 편이
나이키-Running	무난 정 디자인 업 이쁘고 조금 편하고 예뻐요 좋습니다 추천
반스-Canvas	무난 정 어디 이뻐요 편하고 색감 업 기본 이쁘고 오래
슈펜-Canvas	가격 디자인 편하고 조금 색감 이뻐요 좋습니다 업 좋네요 스니커즈
아디다스-Running	정 이뻐요 편하고 가격 크게 주문 평소 같습니다 작게 아주
아식스-Running	편하고 디자인 정 무난 가격 좋습니다 엄청 넓어서 추천 좋네요
엑셀시오르-Canvas	이뻐요 정 무난 조금 디자인 예뻐요 색감 이쁩니다 업 이쁘고
컨버스-Canvas	업 이뻐요 무난 예뻐요 정 색감 기본 조금 좋습니다 편하고
프로스펙스-Running	정 평소 편해요 이쁜 편하고 이뻐요 있어서 넉넉하고 디자인 주문
휠라-Running	키 편하고 크게 가격 착용 높이 선물 원래 디자인 주문

```
WARNING:tensorflow:7 out of the last 7 calls to <function Model.make_predict_function.<locals>.predict_function at 0x7fc6ba047a60>
해당 1.jpg 이미지는 나이키-Running 카테고리로 추정됩니다.
또한, 추천 태그로는 ['무난 정 디자인 업 이쁘고 조금 편하고 예뻐요 좋습니다 추천 ']들이 있습니다.

해당 다운로드.jpg 이미지는 Other-Sneakers 카테고리로 추정됩니다.
또한, 추천 태그로는 ['정 디자인 편하고 무난 가격 좋습니다 넓은 엄청 예뻐요 편이 ']들이 있습니다.

해당 [_[TV CF상품]_]_0.jpg 이미지는 반스-Canvas 카테고리로 추정됩니다.
또한, 추천 태그로는 ['무난 정 어디 이뻐요 편하고 색감 업 기본 이쁘고 오래 ']들이 있습니다.

해당 [_써모 펠트 척 70 러브포션 169519C_]_2.jpg 이미지는 컨버스-Canvas 카테고리로 추정됩니다.
또한, 추천 태그로는 ['업 이뻐요 무난 예뻐요 정 색감 기본 조금 좋습니다 편하고 ']들이 있습니다.

해당 아디다스.jpg 이미지는 나이키-Running 카테고리로 추정됩니다.
또한, 추천 태그로는 ['무난 정 디자인 업 이쁘고 조금 편하고 예뻐요 좋습니다 추천 ']들이 있습니다.

해당 컨버스.jpg 이미지는 컨버스-Canvas 카테고리로 추정됩니다.
또한, 추천 태그로는 ['업 이뻐요 무난 예뻐요 정 색감 기본 조금 좋습니다 편하고 ']들이 있습니다.
```

4. 결과 확인 및 모델 수정

- 새로운 테스트 이미지를 함수에 집어넣게 되면, CNN 모델을 통해 해당 신발의 카테고리('브랜드-종류')를 예측하고, 그 예측결과에 맞는 추천태그를 데이터프레임에서 찾아서 결과적으로 신발에 해당되는 카테고리와 추천태그를 출력

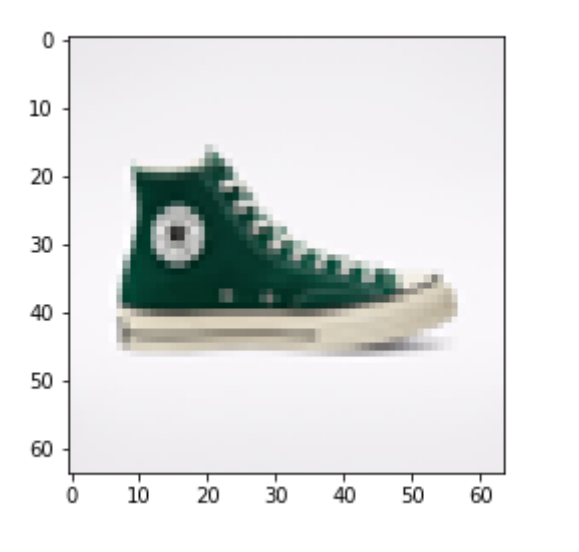
- 이미지 CNN 모델링

  - 메인 사진 외에 세부 사진비율이 더 많아서 예측율과 Loss가 떨어지는 결과가 발생
  - 추가적으로 전이 학습 방법 이용 -> 다른 사람들이 미리 학습해 놓은 모델 이용 -> 모델의 정확도가 높아지기를 기대 -> 오히려 이전보다 성능이 안 좋게 나옴 -> 전이 학습 방법을 사용x

- 텍스트 전처리

  - 두 가지 라이브러리를 사용했는데 Okt가 우리가 원하는 결과에 더 가까운 글자의 형태를 나타내므로 Okt를 사용함
  - 일반적인 불용어 뿐만 아니라, 신발을 분석하는 것이니까 신발의 브랜드 이름, 신발의 종류, 기타 불용어들을 더 추가함

5. 최종 결과



해당 컨버스.jpg 이미지는 컨버스-Canvas 카테고리로 추정됩니다.
또한, 추천 태그로는 ['업 이뻐요 무난 정 예뻐요 색감 좋습니다 기본 조금 옷 ']들이 있습니다.

감사합니다.