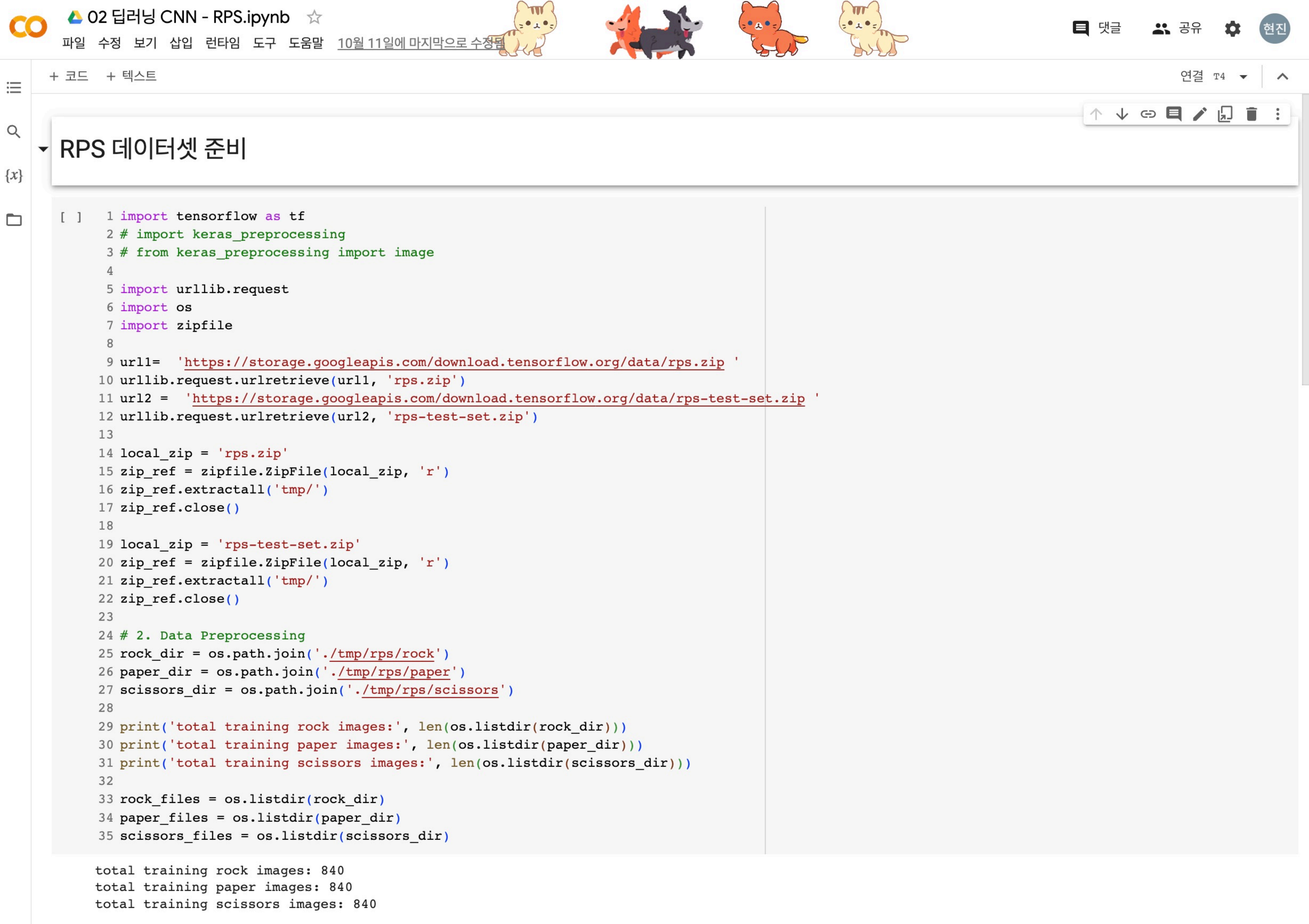
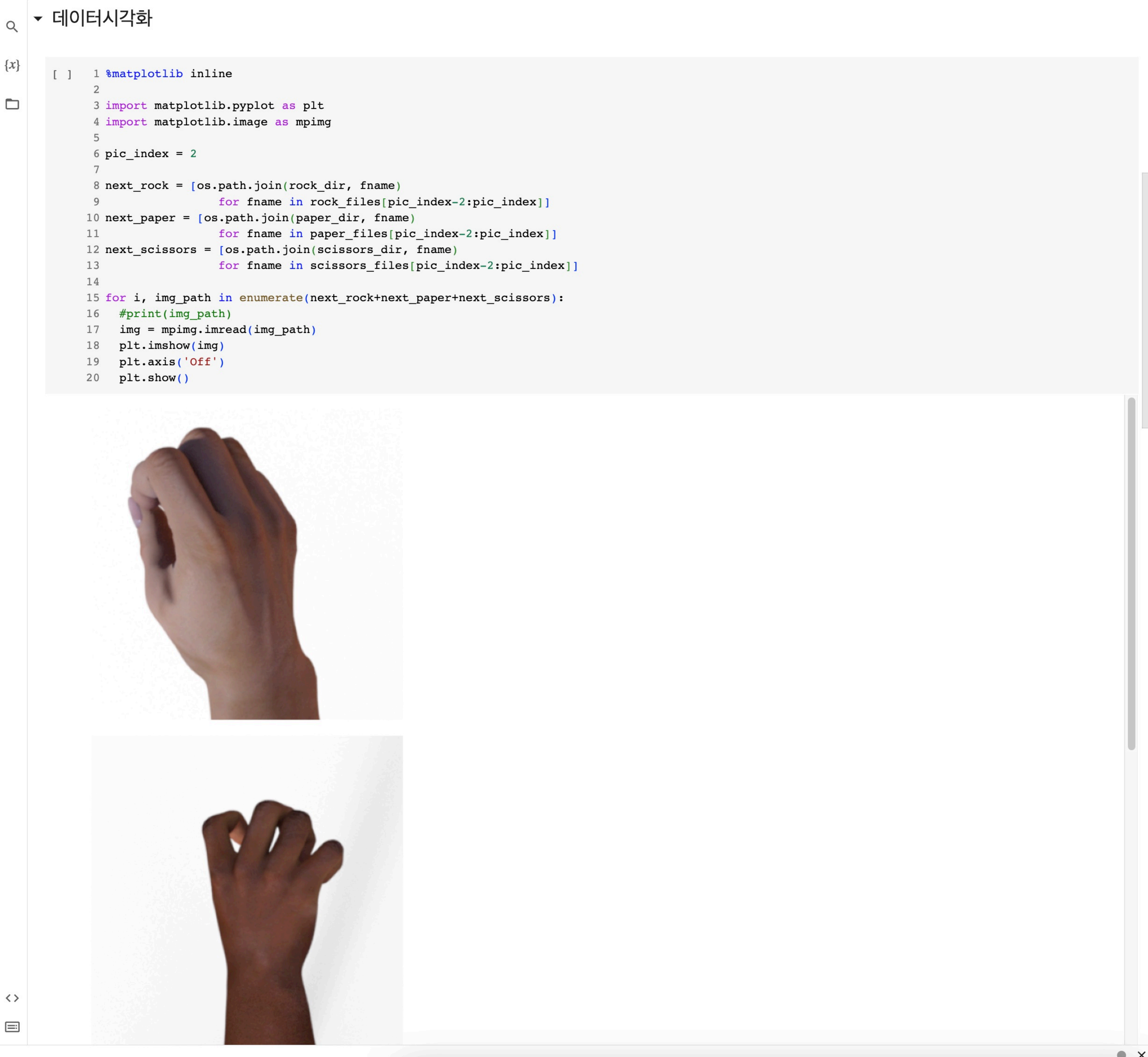

02 딥러닝 CNN - RPS.ipynb
파일 수정 보기 삽입 런타임 도구 도움말
10월 11일에 마지막으로 수정됨
댓글
공유
현진
+ 코드
+ 텍스트
연결 T4
[ ]

RPS 데이터셋 준비

```python
import tensorflow as tf
# import keras_preprocessing
# from keras_preprocessing import image

import urllib.request
import os
import zipfile

url1=  'https://storage.googleapis.com/download.tensorflow.org/data/rps.zip '
urllib.request.urlretrieve(url1, 'rps.zip')
url2 =  'https://storage.googleapis.com/download.tensorflow.org/data/rps-test-set.zip '
urllib.request.urlretrieve(url2, 'rps-test-set.zip')

local_zip = 'rps.zip'
zip_ref = zipfile.ZipFile(local_zip, 'r')
zip_ref.extractall('tmp/')
zip_ref.close()

local_zip = 'rps-test-set.zip'
zip_ref = zipfile.ZipFile(local_zip, 'r')
zip_ref.extractall('tmp/')
zip_ref.close()

# 2. Data Preprocessing
rock_dir = os.path.join('./tmp/rps/rock')
paper_dir = os.path.join('./tmp/rps/paper')
scissors_dir = os.path.join('./tmp/rps/scissors')

print('total training rock images:', len(os.listdir(rock_dir)))
print('total training paper images:', len(os.listdir(paper_dir)))
print('total training scissors images:', len(os.listdir(scissors_dir)))

rock_files = os.listdir(rock_dir)
paper_files = os.listdir(paper_dir)
scissors_files = os.listdir(scissors_dir)
```

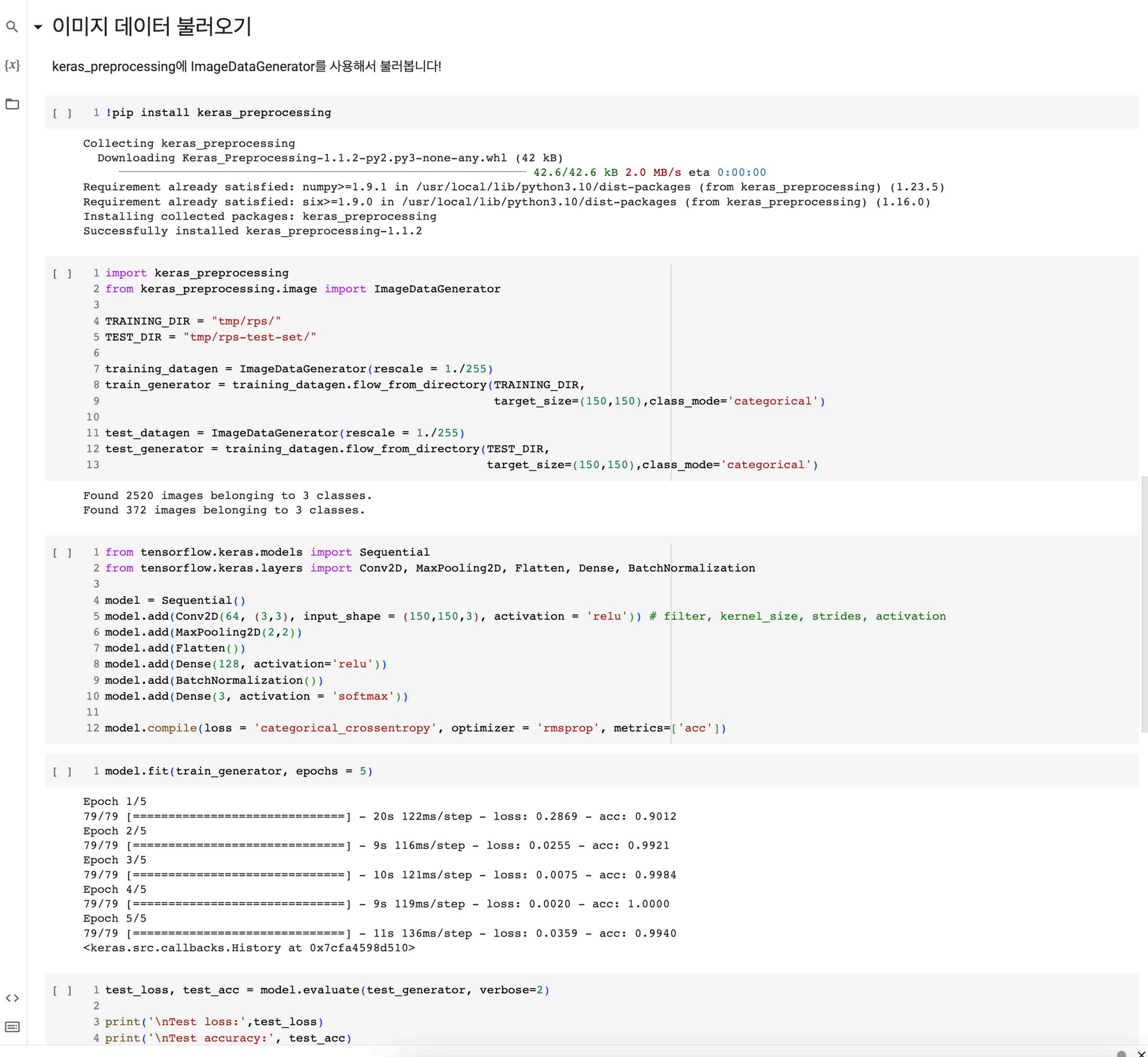
```
total training rock images: 840
total training paper images: 840
total training scissors images: 840
```



데이터시각화

[ ]

```
%matplotlib inline

import matplotlib.pyplot as plt
import matplotlib.image as mpimg

pic_index = 2

next_rock = [os.path.join(rock_dir, fname)
                for fname in rock_files[pic_index-2:pic_index]]
next_paper = [os.path.join(paper_dir, fname)
                for fname in paper_files[pic_index-2:pic_index]]
next_scissors = [os.path.join(scissors_dir, fname)
                for fname in scissors_files[pic_index-2:pic_index]]

for i, img_path in enumerate(next_rock+next_paper+next_scissors):
  #print(img_path)
  img = mpimg.imread(img_path)
  plt.imshow(img)
  plt.axis('Off')
  plt.show()
```



이미지 데이터 불러오기

keras_preprocessing에 ImageDataGenerator를 사용해서 불러봅니다!

```
[ ]
!pip install keras_preprocessing
```

```
Collecting keras_preprocessing
  Downloading Keras_Preprocessing-1.1.2-py2.py3-none-any.whl (42 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 42.6/42.6 kB 2.0 MB/s eta 0:00:00
Requirement already satisfied: numpy>=1.9.1 in /usr/local/lib/python3.10/dist-packages (from keras_preprocessing) (1.23.5)
Requirement already satisfied: six>=1.9.0 in /usr/local/lib/python3.10/dist-packages (from keras_preprocessing) (1.16.0)
Installing collected packages: keras_preprocessing
Successfully installed keras_preprocessing-1.1.2
```

```
[ ]
import keras_preprocessing
from keras_preprocessing.image import ImageDataGenerator

TRAINING_DIR = "tmp/rps/"
TEST_DIR = "tmp/rps-test-set/"

training_datagen = ImageDataGenerator(rescale = 1./255)
train_generator = training_datagen.flow_from_directory(TRAINING_DIR,
                                                        target_size=(150,150),class_mode='categorical')

test_datagen = ImageDataGenerator(rescale = 1./255)
test_generator = training_datagen.flow_from_directory(TEST_DIR,
                                                       target_size=(150,150),class_mode='categorical')
```

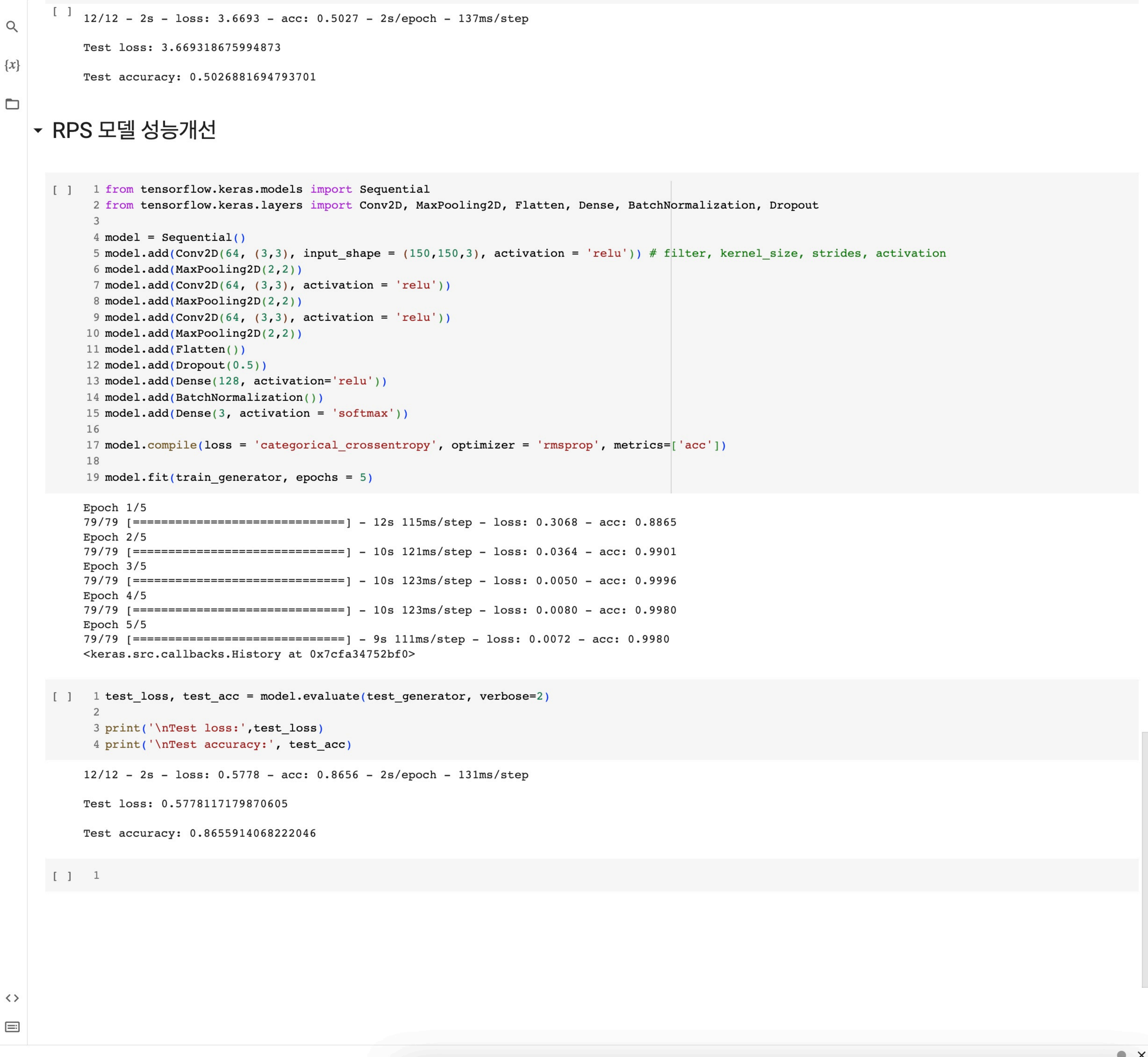
```
Found 2520 images belonging to 3 classes.
Found 372 images belonging to 3 classes.
```

```
[ ]
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Conv2D, MaxPooling2D, Flatten, Dense, BatchNormalization

model = Sequential()
model.add(Conv2D(64, (3,3), input_shape = (150,150,3), activation = 'relu')) # filter, kernel_size, strides, activation
model.add(MaxPooling2D(2,2))
model.add(Flatten())
model.add(Dense(128, activation='relu'))
model.add(BatchNormalization())
model.add(Dense(3, activation = 'softmax'))

model.compile(loss = 'categorical_crossentropy', optimizer = 'rmsprop', metrics=['acc'])
```

```
[ ]
model.fit(train_generator, epochs = 5)
```

```
Epoch 1/5
79/79 [==============================] - 20s 122ms/step - loss: 0.2869 - acc: 0.9012
Epoch 2/5
79/79 [==============================] - 9s 116ms/step - loss: 0.0255 - acc: 0.9921
Epoch 3/5
79/79 [==============================] - 10s 121ms/step - loss: 0.0075 - acc: 0.9984
Epoch 4/5
79/79 [==============================] - 9s 119ms/step - loss: 0.0020 - acc: 1.0000
Epoch 5/5
79/79 [==============================] - 11s 136ms/step - loss: 0.0359 - acc: 0.9940
<keras.src.callbacks.History at 0x7cfa4598d510>
```

```
[ ]
test_loss, test_acc = model.evaluate(test_generator, verbose=2)

print('\nTest loss:',test_loss)
print('\nTest accuracy:', test_acc)
```



```
12/12 - 2s - loss: 3.6693 - acc: 0.5027 - 2s/epoch - 137ms/step

Test loss: 3.669318675994873

Test accuracy: 0.5026881694793701
```

RPS 모델 성능개선

```python
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Conv2D, MaxPooling2D, Flatten, Dense, BatchNormalization, Dropout

model = Sequential()
model.add(Conv2D(64, (3,3), input_shape = (150,150,3), activation = 'relu')) # filter, kernel_size, strides, activation
model.add(MaxPooling2D(2,2))
model.add(Conv2D(64, (3,3), activation = 'relu'))
model.add(MaxPooling2D(2,2))
model.add(Conv2D(64, (3,3), activation = 'relu'))
model.add(MaxPooling2D(2,2))
model.add(Flatten())
model.add(Dropout(0.5))
model.add(Dense(128, activation='relu'))
model.add(BatchNormalization())
model.add(Dense(3, activation = 'softmax'))

model.compile(loss = 'categorical_crossentropy', optimizer = 'rmsprop', metrics=['acc'])

model.fit(train_generator, epochs = 5)
```

```
Epoch 1/5
79/79 [==============================] - 12s 115ms/step - loss: 0.3068 - acc: 0.8865
Epoch 2/5
79/79 [==============================] - 10s 121ms/step - loss: 0.0364 - acc: 0.9901
Epoch 3/5
79/79 [==============================] - 10s 123ms/step - loss: 0.0050 - acc: 0.9996
Epoch 4/5
79/79 [==============================] - 10s 123ms/step - loss: 0.0080 - acc: 0.9980
Epoch 5/5
79/79 [==============================] - 9s 111ms/step - loss: 0.0072 - acc: 0.9980
<keras.src.callbacks.History at 0x7cfa34752bf0>
```

```python
test_loss, test_acc = model.evaluate(test_generator, verbose=2)

print('\nTest loss:',test_loss)
print('\nTest accuracy:', test_acc)
```

```
12/12 - 2s - loss: 0.5778 - acc: 0.8656 - 2s/epoch - 131ms/step

Test loss: 0.5778117179870605

Test accuracy: 0.8655914068222046
```