KERAS로 CNN을 이용하여 다중클래스 영상 부풀리고 분류하기 예제

1. conda create -n keras-cnn-aug

2. activate keras-cnn-aug

3. pip install tensorflow-gpu keras opencv pydot pillow

4. md keras-cnn-aug 후 cd keras-cnn-aug

5. md augmented 후 cd augumented

6. md circle md rectangle md triangle 차례로 실행하여 폴더 생성

- 웹사이트에서 keras-cnn-aug-folde.zip 파일을 다운로드하여 keras-cnn-aug 폴더아래 저장
  (circle001.png, circle002.png, ...circle015.png 식으로 24x24크기 아래처럼 이미지를 생성
  > train용 데이터 각 도형별로 이미지 15개씩 저장, 학습용데이터 3분류x각15장=45장
  > test용 데이터 각 도형별로 이미지 5개씩 저장, 검증용데이터 3분류x각5장=15장)

7. python keras-cnn.aug.py a 실행하여 Data 부풀리기 함
   (만들어 둔 augmented 폴더아래의 3개 서브디렉토리에 부풀려진 그림들 저장 됨)

9. python keras-cnn-augment.py n 실행하여 결과 확인

※ 과제[1]
스스로 9장의 그림 데이터(삼각형, 사각형, 원 각각 3개씩)를 만들어서 mytest라는 폴더에 넣고 인식률을 테스트하시오.
(keras-cnn-aug.py 소스에 mytest라는 data generator를 추가하여 테스트 하시오)

※ 과제[2]
Data Augmentation 사용하여 데이터 수를 늘인 후, 지폐(money) 종류별로 분류하도록 데이터를 변경해서 결과를 확인 하세요. (종류는 천원, 오천원, 만원, 돈 아닌 경우...)

```
#*******************************************************************************
# (가) keras cnn을 이용하여 영상이미지 다중분류 예제 (keras-cnn.py)
#*******************************************************************************

import numpy as np
from keras.models import Sequential
from keras.layers import Dense
from keras.layers import Flatten
from keras.layers.convolutional import Conv2D
from keras.layers.convolutional import MaxPooling2D
from keras.preprocessing.image import ImageDataGenerator

np.random.seed(5)

#==============================================================================
# (1) 데이터 준비
#==============================================================================
# Data Augmentaion(부풀리기)과 입력을 용이하게 하기 위하여 ImageDataGenerator 사용
train_datagen = ImageDataGenerator(rescale = 1/255.)
test_datagen =  ImageDataGenerator(rescale = 1/255.)

train_generator = train_datagen.flow_from_directory (
                  'train', target_size=(24,24), batch_size=3, class_mode='categorical')
test_generator = test_datagen.flow_from_directory (
                  'test', target_size=(24,24), batch_size=3, class_mode='categorical',
                  shuffle=False)

#==============================================================================
# (2) 모델 구성
#==============================================================================
model = Sequential()
model.add(Conv2D(32, kernel_size=(3,3), activation='relu',input_shape=(24,24,3)))
model.add(Conv2D(64, kernel_size=(3,3), activation='relu'))

model.add(MaxPooling2D(pool_size=(2,2)))

model.add(Flatten())

model.add(Dense(128, activation='relu'))
model.add(Dense(3, activation='softmax'))
```

```
#=============================================================================
# (3) 그래프로 모델 확인
#=============================================================================
'''
from IPython.display import SVG
from keras.utils.vis_utils import model_to_dot

%matplotlib inline

SVG(model_to_dot(model, show_shapes=True), create(prog='dot', format='svg'))
'''
#=============================================================================
# (4) 모델 학습과정 설정
#=============================================================================
model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy'])

#=============================================================================
# (5) 모델 학습시키기
#=============================================================================
model.fit_generator(train_generator, steps_per_epoch=15,epochs=50,
                    validation_data=test_generator,validation_steps=5)

#=============================================================================
# (6) 모델 평가하기
#=============================================================================
print('********** 평가하기 **********')

scores = model.evaluate_generator(test_generator, steps=5)

print("$s:%.2f 퍼센트" %(model.metrics_names[1], scores[1]*100))

#=============================================================================
# (7) 모델 사용(예측)하기
#=============================================================================
print('********** 예측하기 **********')
np.set_printoptions(formatter={'float': lambda x: "{0:0.3f}".format(x)})

print(test_generator.class_indices)

output = model.predict_generator(test_generator, steps=5)

print(output)
```

```
#*******************************************************************************
# (나) 데이터 부풀리기 후 영상이미지 다중분류 예제  (keras-cnn-aug.py)
#*******************************************************************************

import numpy as np
from keras.models import Sequential
from keras.layers import Dense
from keras.layers import Flatten
from keras.layers.convolutional import Conv2D
from keras.layers.convolutional import MaxPooling2D
from keras.preprocessing.image import ImageDataGenerator, array_to_img, img_to_array, load_img
import sys

np.random.seed(3)

#==============================================================================
# (1) Data Preparation
#==============================================================================
# Data Augmentaion and use of ImageDataGenerator
train_datagen = ImageDataGenerator(rescale = 1/255.)
test1_datagen =  ImageDataGenerator(rescale = 1/255.)
test2_datagen =  ImageDataGenerator(rescale = 1/255.)

#---------------------
# Data Augmentation
#---------------------
option = ''
if len(sys.argv) == 2 :
    option = sys.argv[1]
else :
    print("\nUsage[1]: python keras-cnn-aug.py a(ugmentation)")
    print("Usage[2]: python keras-cnn-aug.py n(ormal)")
    sys.exit()

if option == 'a'  :
    print('Data Augmentation is being done.....')
    data_aug_gen = ImageDataGenerator(rescale=1/255.,
        rotation_range=10,width_shift_range=0.2,
        height_shift_range=0.2, shear_range=0.7, zoom_range=[0.9,2.2],
        horizontal_flip=True, vertical_flip=True,fill_mode='nearest')

    category = ['circle', 'rectangle', 'triangle']
    new_category = ['cir', 'rec', 'tri']
```

```
    for n in range(0,3) :                       # for each folder
        for alldata_seqno in range(1,16) :      # for each original image
            #make file name
            src_folder = 'train/'+category[n]+'/'
            if alldata_seqno < 10  :
                src_img = src_folder+category[n]+'00'+str(alldata_seqno)+'.png'
            else :
                src_img = src_folder+category[n]+'0'+str(alldata_seqno)+'.png'
            print("original image : {0}...".format(src_img))

            #reshape image
            img = load_img(src_img)
            x = img_to_array(img)
            x = x.reshape((1,)+x.shape)

            # Augment 15 times per each original image
            target_folder = 'augmented/'+category[n]+'/'
            i = 0
            for batch in data_aug_gen.flow(x, batch_size=1, save_to_dir =target_folder,
                            save_prefix=new_category[n], save_format='png') :
                i += 1
                if i >= 15 :
                    break

    print('\n Data Augmentation is finished')
    print('\n Copy augmented images of the augmented folder to the train folder\n')
    sys.exit()

# set data folder name
train_folder_name = 'augmented'
test1_folder_name = 'test'
test2_folder_name = 'train'

# make train & test generator
train_generator = train_datagen.flow_from_directory (
    train_folder_name, target_size=(24,24), batch_size=3, class_mode='categorical')
test1_generator = test1_datagen.flow_from_directory (
    test1_folder_name, target_size=(24,24), batch_size=3, class_mode='categorical',shuffle=False)
test2_generator = test2_datagen.flow_from_directory (
    test2_folder_name, target_size=(24,24), batch_size=3, class_mode='categorical',shuffle=False)

#=============================================================================
# (2) Model Construction
#=============================================================================
model = Sequential()
model.add(Conv2D(32, kernel_size=(3,3), activation='relu',input_shape=(24,24,3)))
```

```
model.add(Conv2D(64, kernel_size=(3,3), activation='relu'))

model.add(MaxPooling2D(pool_size=(2,2)))

model.add(Flatten())

model.add(Dense(128, activation='relu'))
model.add(Dense(3, activation='softmax'))

#=============================================================================
# (3) Model Checking
#=============================================================================
'''
from IPython.display import SVG
from keras.utils.vis_utils import model_to_dot

%matplotlib inline

SVG(model_to_dot(model, show_shapes=True), create(prog='dot', format='svg'))
'''
#=============================================================================
# (4) Configuring Training
#=============================================================================
model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy'])

#=============================================================================
# (5) Model Training
#=============================================================================
'''
#In case of Small Data
model.fit_generator(train_generator, steps_per_epoch=15, epochs=50,
    validation_data=test_generator, validation_steps=5, verbose=1)
'''
#In case of Big Data
model.fit_generator(train_generator, steps_per_epoch=len(train_generator),epochs=50,
    validation_data=test1_generator,validation_steps=5,verbose=1)
    #steps_per_epoch=290

#=============================================================================
# (6) Model Evaluation
#=============================================================================
print('********** Evaluation **********')

np.set_printoptions(formatter={'float': lambda x: '{0:0.3f}'.format(x)})

scores = model.evaluate_generator(test1_generator, steps=5)
```

```
print('%s:%.2f %%' %(model.metrics_names[1], scores[1]*100))

#==============================================================================
# (7) Model Prediction
#==============================================================================
print('********** Prediction **********')

# output class mode & class indices
cls_md = "Class Mode: "+test1_generator.class_mode+"\n"
print(cls_md)
print(test1_generator.class_indices)

# output test results
print('>>>>>>>>>>  Test1 Result <<<<<<<<<<')
output1 = model.predict_generator(test1_generator, steps=None, verbose=0)
print(output1)

print('>>>>>>>>>>  Test2 Result <<<<<<<<<<')
output2 = model.predict_generator(test2_generator, steps=None, verbose=0)
print(output2)
```

<<< 실행 결과 >>>

```
********** Evaluation **********
acc:100.00 %
********** Prediction **********
Class Mode: categorical

{'circle': 0, 'rectangle': 1, 'triangle': 2}

>>  Test1 Result <<
[[0.000 0.000 1.000]
 [1.000 0.000 0.000]
 [0.000 1.000 0.000]
 [0.000 0.000 1.000]
 [0.000 1.000 0.000]
 [0.000 0.000 1.000]
 [1.000 0.000 0.000]
 [0.000 0.000 1.000]
 [1.000 0.000 0.000]
 [0.000 0.000 1.000]
 [0.000 1.000 0.000]
 [1.000 0.000 0.000]
 [0.000 1.000 0.000]
 [1.000 0.000 0.000]]

>>  Test2 Result <<
[[1.000 0.000 0.000]
 [1.000 0.000 0.000]
 [1.000 0.000 0.000]
 [1.000 0.000 0.000]
 [1.000 0.000 0.000]
 [1.000 0.000 0.000]
 [1.000 0.000 0.000]
 [1.000 0.000 0.000]
 [1.000 0.000 0.000]
 [1.000 0.000 0.000]
 [1.000 0.000 0.000]
 [1.000 0.000 0.000]
 [1.000 0.000 0.000]
 [1.000 0.000 0.000]

[0.000 1.000 0.000]
[0.000 1.000 0.000]
[0.000 1.000 0.000]
[0.000 1.000 0.000]
[0.000 1.000 0.000]
[0.000 1.000 0.000]
[0.000 1.000 0.000]
[0.000 1.000 0.000]
[0.000 1.000 0.000]
[0.000 1.000 0.000]
[0.000 1.000 0.000]
[0.000 1.000 0.000]
[0.000 1.000 0.000]
[0.000 1.000 0.000]

[0.000 0.000 1.000]
[0.000 0.000 1.000]
[0.000 0.000 1.000]
[0.000 0.000 1.000]
[0.000 0.000 1.000]
[0.000 0.000 1.000]
[0.000 0.000 1.000]
[0.000 0.000 1.000]
[0.000 0.000 1.000]
[0.000 0.000 1.000]
[0.000 0.238 0.762]
[0.000 0.000 1.000]
[0.000 0.000 1.000]
[0.000 0.000 1.000]]
```

#***
(나) 데이터 부풀리기 후 영상이미지 다중분류 예제 (keras-cnn-aug.py)
#***
라인번호 포함 소스 버전

```
1   #*********************************************************
2   # Image multi-classification using keras
3   #*********************************************************
4
5   import numpy as np
6   from keras.models import Sequential
7   from keras.layers import Dense
8   from keras.layers import Flatten
9   from keras.layers.convolutional import Conv2D
10  from keras.layers.convolutional import MaxPooling2D
11  from keras.preprocessing.image import ImageDataGenerator, array_to_img, img_to_array, load_img
12  import sys
13
14  np.random.seed(3)
15
16  #=============================================================================
17  # Data Preparation
18  #=============================================================================
19  # Data Augmentaion and use of ImageDataGenerator
20  train_datagen = ImageDataGenerator(rescale = 1/255.)
21  test1_datagen =  ImageDataGenerator(rescale = 1/255.)
22  test2_datagen =  ImageDataGenerator(rescale = 1/255.)
23
24  #---------------------
25  # Data Augmentation
26  #---------------------
27  option = ''
28  if len(sys.argv) == 2 :
29      option = sys.argv[1]
30  else :
31      print("\nUsage[1]: python keras-cnn-aug.py a(ugmentation)")
32      print("Usage[2]: python keras-cnn-aug.py n(ormal)")
33      sys.exit()
34
35  if option == 'a'  :
36      print('Data Augmentation is being done......')
37      data_aug_gen = ImageDataGenerator(rescale=1/255., rotation_range=10,width_shift_range=0.2,
38          height_shift_range=0.2, shear_range=0.7, zoom_range=[0.9,2.2],
39          horizontal_flip=True, vertical_flip=True,fill_mode='nearest')
40
41      category = ['circle', 'rectangle', 'triangle']
42      new_category = ['cir', 'rec', 'tri']
```

```
    for n in range(0,3) :                    # for each folder
        for alldata_seqno in range(1,16) :  # for each original image
            #make file name
            src_folder = 'train/'+category[n]+'/'
            if alldata_seqno < 10  :
                src_img = src_folder+category[n]+'00'+str(alldata_seqno)+'.png'
            else :
                src_img = src_folder+category[n]+'0'+str(alldata_seqno)+'.png'
            print("original image : {0}...".format(src_img))

            #reshape image
            img = load_img(src_img)
            x = img_to_array(img)
            x = x.reshape((1,)+x.shape)

            # Augment 15 times per each original image
            target_folder = 'augmented/'+category[n]+'/'
            i = 0
            for batch in data_aug_gen.flow(x, batch_size=1, save_to_dir =target_folder,
                                           save_prefix=new_category[n], save_format='png') :
                i += 1
                if i >= 15 :
                    break

    print('\n Data Augmentation is finished')
    print('\n Copy augmented images of the augmented folder to the train folder\n')
    sys.exit()

# set data folder name
train_folder_name = 'augmented'
test1_folder_name = 'test'
test2_folder_name = 'train'

# make train & test generator
train_generator = train_datagen.flow_from_directory (
    train_folder_name, target_size=(24,24), batch_size=3, class_mode='categorical')
test1_generator = test1_datagen.flow_from_directory (
    test1_folder_name, target_size=(24,24), batch_size=3, class_mode='categorical',shuffle=False)
test2_generator = test2_datagen.flow_from_directory (
    test2_folder_name, target_size=(24,24), batch_size=3, class_mode='categorical',shuffle=False)

```

```
#==============================================================================
# Model Construction
#==============================================================================
model = Sequential()
model.add(Conv2D(32, kernel_size=(3,3), activation='relu',input_shape=(24,24,3)))
model.add(Conv2D(64, kernel_size=(3,3), activation='relu'))

model.add(MaxPooling2D(pool_size=(2,2)))

model.add(Flatten())

model.add(Dense(128, activation='relu'))
model.add(Dense(3, activation='softmax'))

#==============================================================================
# Model Checking
#==============================================================================
'''
from IPython.display import SVG
from keras.utils.vis_utils import model_to_dot

%matplotlib inline

SVG(model_to_dot(model, show_shapes=True), create(prog='dot', format='svg'))
'''
#==============================================================================
# Configuring Training
#==============================================================================
model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy'])

#==============================================================================
# Model Training
#==============================================================================
'''
#In case of Small Data
model.fit_generator(train_generator, steps_per_epoch=15,epochs=50,
    validation_data=test_generator,validation_steps=5,verbose=1)
'''
#In case of Big Data
model.fit_generator(train_generator, steps_per_epoch=len(train_generator),epochs=50,
    validation_data=test1_generator,validation_steps=5,verbose=1)
    #steps_per_epoch=90
```

```
126
127 #==============================================================================
128 # Model Evaluation
129 #==============================================================================
130 print('********** Evaluation **********')
131
132 np.set_printoptions(formatter={'float': lambda x: '{0:0.3f}'.format(x)})
133
134 scores = model.evaluate_generator(test1_generator, steps=5)
135
136 print('%s:%.2f %%' %(model.metrics_names[1], scores[1]*100))
137
138 #==============================================================================
139 # Model Prediction
140 #==============================================================================
141 print('********** Prediction **********')
142
143 # output class mode & class indices
144 cls_md = "Class Mode: "+test1_generator.class_mode+"\n"
145 print(cls_md)
146 print(test1_generator.class_indices)
147
148 # output test results
149 print('>>>>>>>>>>  Test1 Result <<<<<<<<<<')
150 output1 = model.predict_generator(test1_generator, steps=None, verbose=0)
151 print(output1)
152
153 print('>>>>>>>>>>  Test2 Result <<<<<<<<<<')
154 output2 = model.predict_generator(test2_generator, steps=None, verbose=0)
155 print(output2)
156
```