🏨

12주차 CNN + autoencoder

Color Images from Gray : CNN + Autoencoder

```
import numpy as np
import tensorflow as tf
import tensorflow as keras
from tensorflow.keras import layers
import cv2
from tensorflow.keras.preprocessing.image import img_to_array
import os
from tqdm import tqdm #tqdm 이미지 불러오기 용이한 라이브러리
import re
import matplotlib.pyplot as plt
import natsort
from tensorflow.keras.layers import MaxPool2D,Conv2D,UpSampling2D,Input,Dropout
from tensorflow.keras.models import Sequential
```

```
SIZE = 160
color_img = []
path = '/content/drive/MyDrive/landscapeImages/color/'
files = os.listdir(path)
files = natsort.natsorted(files) #text로 된 문자정렬하기

#tqdm : 이미지 불러오기 용이한 라이브러리
for i in tqdm(files):
  if i=='2500.jpg': 2500장만 우선 처리해
    break
else:
  img=cv2.imread(path+'/'+i,1)
  img=cv2.cvtColor(img,cv2.COLOR_BGR2RGB)
  img=cv2.resize(img,(SIZE,SIZE)
  img=img.asytpe('float32')/255 #정규화
  color_img.append(img_to_array(img)) #COLOR IMAGE ARRAY 생성
```

```
SIZE = 160
gray_img = []
path = '/content/drive/MyDrive/landscapeImages/gray/'
files = os.listdir(path)
files = natsort.natsorted(files) #text로 된 문자정렬하기 = 문자열 순서대로

#tqdm
for i in tqdm(files):
  if i=='2500.jpg':
    break
else:
  img=cv2.imread(path+'/'+i,1)
  img=cv2.cvtColor(img,cv2.COLOR_BGR2RGB)
  img=cv2.resize(img,(SIZE,SIZE)
  img=img.asytpe('float32')/255 #정규화
  gray_img.append(img_to_array(img)) # GRAYIMAGE ARRAY 생성
```

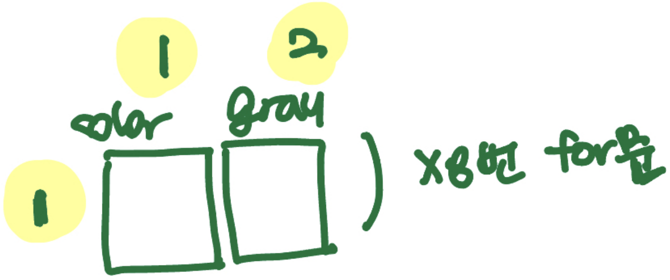
```
def plot_images(color,grayscale):
  plt.figure(figsize=(5,5))
  plt.subplot(1,3,1)
  plt.title("Color",color='green',fontsize=8)
  plt.imshow(color)
  plt.subplot(1,3,2)
  plt.title("GrayScale",color='black',fontsize=8)
  plt.imshow(grayscale)

  plt.show()
```

subplot: 축 공유하기

plt.subplot(1,3,1) row. column. index

plt.subplot(1,3,2)



```
for i range(3,9):
  plot_images(color_img[i],gray_img[i])
```

```
train_gray_image=gray_img[:2300]
train_color_image=color_img[:2300]

test_gray_image=gray_img[2300:2500]
test_color_image=color_img[2300:2500]
```

```
#reshaping -> array형태로 변환
train_g=np.reshape(train_gray_image,(len(train_gray_image),SIZE,SIZE,3))
train_c=np.reshape(train_color_image,(len(train_color_image),SIZE,SIZE,3))
print('Train color image shape:',train_c.shape)

test_g=np.reshape(test_gray_image,(len(test_gray_image),SIZE,SIZE,3))
test_c=np.reshape(test_color_image,(len(test_color_image),SIZE,SIZE,3))

print('Test color image shape:',test_c.shape)
```

Generative Images (CNN + autoencoder (2Dtranspose))

```
def model():
  inp=tf.keras.layers.Input(shape=(160,160,3))
  conv1=tf.keras.layers.Conv2D(16,3,padding='same',activation='relu')(inp)
  conv2=tf.keras.layers.Conv2D(32,3,padding='same',activation='relu')(conv1)
  conv3=tf.keras.layers.Conv2D(64,3,padding='same',activation='relu')(conv2)
  conv4=tf.keras.layers.Conv2D(128,3,padding='same',activation='relu')(conv3)

#padding값이 많아진다 = 차원이 축소된다

  convt1=tf.keras.layers.Conv2DTranspose(128,3,padding='same'=activation='relu')(conv4)
  convt2=tf.keras.layers.Conv2DTranspose(64,3,padding='same'=activation='relu')(convt1)
  convt3=tf.keras.layers.Conv2DTranspose(32,3,padding='same'=activation='relu')(convt2)
  convt4=tf.keras.layers.Conv2DTranspose(16,3,padding='same'=activation='relu')(convt3)

  out=tf.keras.layers.Conv2DTranspose(3,3,padding='same'=activation='relu')(convt4) #Color channels = 3

  model=tf.keras.model.Model(inp,out)
  model.summary()
  return model


model=model()
```

Model: "model_1"

Layer (type) Output Shape Param

input_2 (InputLayer)   [(None, 160, 160, 3)]   0 **3: 특징 갯수**

conv2d_4 (Conv2D)   (None, 160, 160, 16)   448

conv2d_5 (Conv2D)   (None, 160, 160, 32)   4640

conv2d_6 (Conv2D)   (None, 160, 160, 64)   18496

conv2d_7 (Conv2D)   (None, 160, 160, 128)   73856

```
conv2d_transpose_5 (Conv2DT  (None, 160, 160, 128)    147584
ranspose)

conv2d_transpose_6 (Conv2DT  (None, 160, 160, 64)     73792
ranspose)

conv2d_transpose_7 (Conv2DT  (None, 160, 160, 32)     18464
ranspose)

conv2d_transpose_8 (Conv2DT  (None, 160, 160, 16)     4624
ranspose)

conv2d_transpose_9 (Conv2DT  (None, 160, 160, 3)     435 3 : RGB channels
ranspose)

=================================================================

Total params: 342,339
Trainable params: 342,339
Non-trainable params: 0
```

```
최적화 기법 중 하나인 AdaGrad는 학습이 진행될 때 학습률(Learning rate)이 꾸준히 감소하다 나중에는
0
으로 수렴하여 학습이 더 이상 진행되지 않는다는 한계가 있습니다. RMSProp은 이러한 한계점을 보완한 최적화 기법
으로써 제프리 힌튼 교수가 Coursea 강의 중에 발표한 알고리즘
```

```
optimizer=tf.keras.optimizers.RMSprop(0.0001)
model.compile(optimizer=optimizer,loss='mse')

model.fit(train_g,train_c,epochs=10,batch_size=32)
```

```
def plot_images(color,grayscale,predicted):
  plt.figure(figsize=(15,15))
  plt.subplot(1,3,1)
  plt.title("Color",color="green",fontsize=8)
  plt.imshow(color)

  plt.subplot(1,3,2)
  plt.title("GrayScale",color="black",fontsize=8)

  plt.subplot(1,3,3)
  plt.title('Predicted',color="Red",fontsize=8)
  plt.imshow(predicted)

  plt.show()

#clip 최대 최소값 제한하기
```

```
np.clip(a, -1.0, 1.0)은 어레이 a의 값을 최소 -1.0에서 최대 1.0 사이의 범위로 제한

for i in range(60,70):
  predicted=np.clip(model.predict(test_g[i].reshape(1,160,160,3)).0.0,1.0).reshape(160,
160,3)
  ##predicted는 np.clip에서 이미 array형태가 아니라 하나를 골라서 받음
  test_g[i] 이미지 하나에 대한 값
  plot_images(test_c[i],test_g[i],predicted)
```