|  |  |
| --- | --- |
| 교육제목 | 데이터 기반 인공지능 시스템 엔지니어 양성 과정 |
| 교육일시 | 211029 |
| 교육장소 | 자택(디스코드 사용 온라인 학습) |
| **교육내용** | |
| 1. ImageDataGenerator을 이용한 cat or dat classifier   캡처.JPG  import numpy as np  import matplotlib.pyplot as plt  import tensorflow as tf  from tensorflow import keras  train\_dir = '/content/drive/MyDrive/Colab Notebooks/영우\_4기\_딥러닝/dataset/cats\_and\_dogs\_small/train'  valid\_dir = '/content/drive/MyDrive/Colab Notebooks/영우\_4기\_딥러닝/dataset/cats\_and\_dogs\_small/validation'  test\_dir = '/content/drive/MyDrive/Colab Notebooks/영우\_4기\_딥러닝/dataset/cats\_and\_dogs\_small/test'  # parameter  batch\_size=20  learning\_rate=0.0001  epochs=5  ## DATA를 만듬  train\_datagen = tf.keras.preprocessing.image.ImageDataGenerator( # Augmentation 추가  rotation\_range =40, # 회전 범위  width\_shift\_range=0.2, # 가로로 이동 비율  height\_shift\_range=0.2, # 세로로 이동 비율  shear\_range=0.2, # 전단의 강도  zoom\_range=0.2, # 확대와 축소 범위 (1-0.2 ~ 1+0.2)  horizontal\_flip =True  )  valid\_datagen = tf.keras.preprocessing.image.ImageDataGenerator()  test\_datagen = tf.keras.preprocessing.image.ImageDataGenerator()  ## 만든데이터를 불러와서 파씽함.  train\_generator = train\_datagen.flow\_from\_directory(  directory=train\_dir, # 타깃 디렉터리  target\_size=(128, 128), # 모든 이미지를 128 × 128 크기로 바꿉니다  batch\_size=batch\_size,  interpolation='bilinear', ## resize시 interpolatrion 기법  color\_mode ='rgb',  shuffle='True',  # binary\_crossentropy 손실을 사용하기 때문에 이진 레이블이 필요합니다  class\_mode='binary') # binary, categorical , sparse , input  ## class의 인덱스를 확인.  print(train\_generator.class\_indices)  print(train\_generator.classes)  validation\_generator = valid\_datagen.flow\_from\_directory(  directory=valid\_dir,  target\_size=(128, 128),  batch\_size=batch\_size,  shuffle='True',  interpolation='bilinear', ## resize시 interpolatrion 기법  color\_mode='rgb',  class\_mode='binary') #categorical  test\_generator = test\_datagen.flow\_from\_directory(  directory=test\_dir,  target\_size=(128, 128),  batch\_size=batch\_size,  shuffle='True',  interpolation='bilinear', ## resize시 interpolatrion 기법  color\_mode='rgb',  class\_mode='binary') #categorical  ## 파씽한 데이터의 배치사이즈 확인하기  for data\_batch, labels\_batch in train\_generator:  print('배치 데이터 크기:', data\_batch.shape)  print('배치 레이블 크기:', labels\_batch.shape)  print('class :',train\_generator.class\_indices)  break  ## 모델  input\_Layer = tf.keras.layers.Input(shape=(128,128,3))  x=tf.keras.layers.Conv2D(32,(3,3),strides=1, activation='relu',kernel\_regularizer=tf.keras.regularizers.l2(0.001))(input\_Layer)  x=tf.keras.layers.MaxPool2D((2,2))(x)  x=tf.keras.layers.Conv2D(64,(3,3),strides=1,activation='relu', padding='same')(x)  x=tf.keras.layers.MaxPool2D((2,2))(x)  x=tf.keras.layers.Conv2D(128,(3,3),strides=1,activation='relu')(x)  x=tf.keras.layers.BatchNormalization()(x)  x=tf.keras.layers.Activation('relu')(x)  x=tf.keras.layers.Conv2D(64,(3,3),strides=1,activation='relu')(x)  x=tf.keras.layers.Dropout(0.5)(x)  x=tf.keras.layers.MaxPool2D((2,2))(x)  x=tf.keras.layers.Flatten()(x)  x= tf.keras.layers.Dense(512, activation='relu')(x)  Out\_Layer= tf.keras.layers.Dense(1, activation='sigmoid')(x)  model = tf.keras.Model(inputs=[input\_Layer], outputs=[Out\_Layer])  model.summary()  loss\_function=tf.keras.losses.binary\_crossentropy  optimize=tf.keras.optimizers.RMSprop(learning\_rate=learning\_rate)  metric=tf.keras.metrics.binary\_accuracy  model.compile(loss=loss\_function,  optimizer=optimize,  metrics=[metric])  import os  history = model.fit(  train\_generator,  #steps\_per\_epoch=int((len(os.listdir(train\_dir)))/batch\_size),  steps\_per\_epoch = 100,  epochs=epochs,  validation\_data=validation\_generator,  # callbacks=callbacks\_list,  validation\_freq=1  )  print(model.evaluate(test\_generator))  model.save('cats\_and\_dogs\_binary\_classification.hdf5')  history.history.keys()  import matplotlib.pyplot as plt  acc = history.history['binary\_accuracy']  val\_acc = history.history['val\_binary\_accuracy']  loss = history.history['loss']  val\_loss = history.history['val\_loss']  epochs = range(len(acc))  plt.plot(epochs, acc, 'r-', label='Training acc')  plt.plot(epochs, val\_acc, 'b', label='Validation acc')  plt.title('Training and validation accuracy')  plt.legend()  plt.figure()  plt.plot(epochs, loss, 'r-', label='Training loss')  plt.plot(epochs, val\_loss, 'b', label='Validation loss')  plt.title('Training and validation loss')  plt.legend()  plt.show()   * 오버피팅 경향이 보인다면 image augmentation 적용해본다.  1. Image augmentation   이미지에 회전, 반전, 확대, 축소 등을 줘서 오버피팅을 막는 방법.  캡처.JPG | |