|  |  |
| --- | --- |
| 교육제목 | 데이터 기반 인공지능 시스템 엔지니어 양성 과정 |
| 교육일시 | 211028 |
| 교육장소 | 자택(디스코드 사용 온라인 학습) |
| **교육내용** | |
| 1. 회전된 이미지로 모델 테스트 하기   변환하고 싶은 이미지가 있다면 이미지의 행렬을 변환하면 된다.  예) cv2.getRotationMatrix2D((28/2, 28/2), 90, 1)  ((center\_\_x, center\_y), 회전, 크기)  Cv2.warpAffine(X\_train[5], M, (28,28)) <- (변환하고싶은 이미지, 변환 행렬, 출력사이즈)  캡처.JPG  #-\*- coding: utf-8 -\*-  import tensorflow as tf  import matplotlib.pyplot as plt  import numpy as np  import cv2  # MNIST 데이터 불러오기  (x\_train, y\_train), (x\_test, y\_test) = tf.keras.datasets.mnist.load\_data()  ## 이미지 회전 변환 매트릭스 구하기  M= cv2.getRotationMatrix2D((28/2, 28/2), 0, 1)  # 이미지 이동 변환 매트릭스 구하기  M[0,2] = M[0,2] +3  M[1,2] = M[1,2] +3  # 이미지 변환 매트릭스 적용  test\_image = cv2.warpAffine(x\_train[5], M, (28, 28)) # image에 matrix곱  plt.imshow(test\_image, cmap="Greys")  test\_image\_reshape = test\_image.reshape(1, 784).astype('float64')  from google.colab import drive  drive.mount('/content/drive')  model = tf.keras.models.load\_model('/content/drive/MyDrive/my\_NN\_Test.h5의 사본')  y\_pred = model.predict(test\_image\_reshape)  index = np.argmax(y\_pred)  value = y\_pred[:, index]  plt.imshow(test\_image, cmap='Greys')  plt.xlabel(str(index)+" "+str(value))  plt.show()  print(value) # 4정도로 예측을 해뿐다 먼가 잘못됨   * MLP의 문제점 : 한칸씩만 움직여도 변화하는 인풋값이 너무 많음. 이미지의 ‘특징’을 추출해서 변화에 대응해야 함.  1. Convolution Neural Network   선언된 크기의 필터를 이미지에 override하면서 feature를 추출 > 추출된 feature에서 도드라지는(중요한) feature 추출 >  최종 추출된 feature 기반으로 분류.  import matplotlib.pyplot as plt  import os  import tensorflow as tf  import numpy as np  # 데이터 불러오기  (x\_train, y\_train), (x\_test, y\_test) = tf.keras.datasets.fashion\_mnist.load\_data()  # 데이터 (배치사이즈 x 28 x 28 x 1)로 이미지로 변환 --> 그레이스케일 채널은 1  x\_train = x\_train.reshape(x\_train.shape[0], 28, 28, 1).astype('float32')  x\_test = x\_test.reshape(x\_test.shape[0], 28, 28, 1).astype('float32')  ## 정답을 바이너리화 함.  y\_train = tf.keras.utils.to\_categorical(y\_train)  y\_test = tf.keras.utils.to\_categorical(y\_test)  # CNN 모델 설계  input\_layer = tf.keras.layers.Input(shape=(28, 28, 1))  x = tf.keras.layers.Conv2D(32, (3,3), strides=1, activation='relu', padding='same')(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.Flatten()(x)  x = tf.keras.layers.Dense(512, activation='relu')(x)  output\_layer = tf.keras.layers.Dense(10, activation='softmax')(x)  model = tf.keras.Model(inputs=[input\_layer], outputs=[output\_layer])  model.summary()  loss = tf.keras.losses.categorical\_crossentropy  optimizer = tf.keras.optimizers.RMSprop(learning\_rate=0.0001)  metrics = tf.keras.metrics.categorical\_accuracy  model.compile(loss=loss,  optimizer = optimizer,  metrics= [metrics])  # 모델 최적화 설정  MODEL\_DIR = './CNN\_Fasion\_MNIST\_model/'  if not os.path.exists(MODEL\_DIR):  os.mkdir(MODEL\_DIR)  modelpath="./CNN\_Fasion\_MNIST\_model/{epoch:02d}-{val\_loss:.4f}.hdf5"  callback\_list=[tf.keras.callbacks.ModelCheckpoint(filepath=modelpath, monitor='val\_loss', verbose=1, save\_best\_only=True),  tf.keras.callbacks.EarlyStopping(monitor='val\_loss', patience=10)]  # 모델의 실행  history = model.fit(x\_train, y\_train, validation\_split=0.2, epochs=5, batch\_size=200, verbose=1, callbacks=callback\_list)  # 테스트 정확도 출력  print("\n Test Accuracy: %.4f" % (model.evaluate(x\_test, y\_test)[1]))  # 테스트 셋의 오차  y\_vloss = history.history['val\_loss']  # 학습셋의 오차  y\_loss = history.history['loss']  # 그래프로 표현  x\_len = np.arange(len(y\_loss))  plt.plot(x\_len, y\_vloss, marker='.', c="red", label='Testset\_loss')  plt.plot(x\_len, y\_loss, marker='.', c="blue", label='Trainset\_loss')  # 그래프에 그리드를 주고 레이블을 표시  plt.legend(loc='upper right')  plt.grid()  plt.xlabel('epoch')  plt.ylabel('loss')  plt.show() | |