**교육일지**

**교육 제목 : 딥러닝**

**교육 장소 : YGL C6 강의실**

**교육 일시 : 2021/10/27**

**# 1. 데이터 불러오기**

**# 2. 데이터 확인하기**

**# 3. x,y 나누기**

**# 4. 전체 데이터에서 학습데이터와 테스터데이터(0.2) 구분**

**# 5. 딥러닝 설계 ---> kernel\_initializer = tf.keras.initializers.he\_normal(), tf.keras.initializers.glorot\_uniform()**

**# 6. model.complie**

**# 7. model.fit**

**# 8. 그래프 결과 확인**

**# 9. test 정확도 확인**

**import tensorflow as tf**

**import matplotlib.pyplot as plt**

**import numpy**

**import os**

**(x\_train, y\_train), (x\_test, y\_test) = tf.keras.datasets.mnist.load\_data()**

**print(len(x\_train))**

**print(len(x\_train))**

**print(len(y\_train))**

**print(x\_train.shape)**

**print(y\_train.shape)**

**plt.imshow(x\_train[500], cmap=plt.cm.binary)**

**print(y\_train[500])**

**# index에 0에서 59999사이의 숫자를 지정**

**index = 387**

**plt.imshow(x\_train[index], cmap=plt.cm.binary)**

**plt.show()**

**print((index+1), "번째 이미지의 숫자는 바로 ", y\_train[index],'입니다.')**

**print(x\_train.shape)**

**print(x\_test.shape)**

**print(y\_train.shape)**

**print(y\_test.shape)**

**import numpy as np**

**print('최소값:', np.min(x\_train), '최대값:', np.max(x\_train))**

**# 정규화**

**x\_train\_norm = x\_train / 255.0**

**x\_test\_norm = x\_test / 255.0**

**print('최소값:', np.min(x\_train\_norm), '최대값:', np.max(x\_test\_norm))**

**x\_train\_norm.shape**

**# reshape 28 \* 28 --> 784,1변환**

**x\_train = x\_train\_norm.reshape(x\_train.shape[0], 784).astype('float32')**

**x\_test = x\_test\_norm.reshape(x\_test.shape[0], 784).astype('float32')**

**y\_train = tf.keras.utils.to\_categorical(y\_train, 10)**

**y\_test = tf.keras.utils.to\_categorical(y\_test, 10)**

**print(y\_train[0])**

**from tensorflow import keras**

**model = keras.models.Sequential()**

**model.add(keras.layers.Dense(16, activation='relu', input\_shape=(784,))**

**model.add(keras.layers.Dense(32, activation='relu'))**

**model.add(keras.layers.Dense(10, activation='softmax'))**

**model.summary()**

**print(len(model.layers))**

**model.compile(optimizer='adam',**

**loss = 'categorical\_crossentropy',**

**metrics = ['accuracy'])**

**history = model.fit(x\_train, y\_train, validation\_split=0.1, epochs=10, batch\_size=200, verbose=1)**

**test\_loss, test\_accuracy = model.evaluate(x\_test, y\_test, verbose=2)**

**print('test loss : {}'.format(test\_loss))**

**print('test\_accuracy: {}'.format(test\_accuracy))**

**# 예측**

**predicted\_result = model.predict(x\_test)**

**predicted\_labels = np.argmax(predicted\_result,axis=1)**

**idx = 0**

**print('model.predict() 결과 :',predicted\_result[idx])**

**print('model이 추론한 가장 가능성이 높은 결과 : ',predicted\_labels[idx])**

**print('실제 데이터의 라벨 :',y\_test[idx])**

**x\_test.shape**

**x\_test\_result = x\_test.reshape(x\_test.shape[0], 28, 28)**

**x\_test\_result.shape**

**plt.imshow(x\_test\_result[idx], cmap=plt.cm.binary)**

**plt.show()**

**# 테스트 셋의 오차**

**y\_vloss = history.history['val\_loss']**

**# 학습셋의 오차**

**y\_loss = history.history['loss']**

**# 그래프로 표현**

**x\_len = numpy.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.axis([0, 20, 0, 0.35])**

**plt.grid()**

**plt.xlabel('epoch')**

**plt.ylabel('loss')**

**plt.show()**