**교육일지**

**교육 제목 : 딥러닝**

**교육 장소 : YGL C6 강의실**

**교육 일시 : 2021/11/01**

**CNN 모델 구현하기**

**import numpy as np**

**import matplotlib.pyplot as plt**

**import tensorflow as tf**

**from tensorflow import keras**

**from tensorflow.keras.datasets import cifar10**

**(x\_train, y\_train), (x\_test, y\_test) = cifar10.load\_data()**

**x\_train, y\_train, x\_test, y\_test shape을 뽑기**

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

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

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

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

**위에서 뽑은 train데이터의 라벨값을 출력**

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

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

**최대/최소로 정규화**

**x\_train = x\_train.astype('float32')**

**x\_test = x\_test.astype('float32')**

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

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

**CNN모델을 설계**

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

**model.add(keras.layers.Conv2D(16, (3, 3), padding='same', activation='relu', input\_shape=(32, 32, 3)))**

**model.add(keras.layers.Conv2D(16, (3, 3), padding='same', activation='relu'))**

**model.add(keras.layers.MaxPool2D(2,2))**

**model.add(keras.layers.Conv2D(32, (3,3), padding='same', activation='relu'))**

**model.add(keras.layers.Conv2D(32, (3,3), padding='same', activation='relu'))**

**model.add(keras.layers.MaxPool2D(2,2))**

**model.add(keras.layers.Conv2D(64, (3,3), padding='same', activation='relu'))**

**model.add(keras.layers.Conv2D(64, (3,3), padding='same', activation='relu'))**

**model.add(keras.layers.Conv2D(64, (3,3), padding='same', activation='relu'))**

**model.add(keras.layers.MaxPool2D(2,2))**

**model.add(keras.layers.Conv2D(128, (3,3), padding='same', activation='relu'))**

**model.add(keras.layers.Conv2D(128, (3,3), padding='same', activation='relu'))**

**model.add(keras.layers.Conv2D(128, (3,3), padding='same', activation='relu'))**

**model.add(keras.layers.MaxPool2D(2,2))**

**model.add(keras.layers.Conv2D(256, (3,3), padding='same', activation='relu'))**

**model.add(keras.layers.Conv2D(256, (3,3), padding='same', activation='relu'))**

**model.add(keras.layers.Conv2D(256, (3,3), padding='same', activation='relu'))**

**model.add(keras.layers.MaxPool2D(1,1))**

**model.add(keras.layers.Flatten())**

**model.add(keras.layers.Dense(256, activation='relu'))**

**model.add(keras.layers.Dense(128, activation='relu'))**

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

**def build\_vgg\_block(input\_layer,**

**num\_cnn=3,**

**channel=64,**

**block\_num=1,**

**):**

**# 입력레이어**

**x = input\_layer**

**# CNN 레이어**

**for cnn\_num in range(num\_cnn):**

**x = keras.layers.Conv2D(**

**filters = channel,**

**kernel\_size = (3,3),**

**activation='relu',**

**kernel\_initializer='he\_normal',**

**padding='same',**

**name=f'block{block\_num}\_conv{cnn\_num}'**

**)(x)**

**# Max Pooling 레이어**

**x = keras.layers.MaxPooling2D(**

**pool\_size = (2,2),**

**strides = 2,**

**name=f'block{block\_num}\_pooling'**

**)(x)**

**return x**

**vgg\_input\_layer = keras.layers.Input(shape=(32, 32, 3)) #입력레이어 생성**

**vgg\_block\_output = build\_vgg\_block(vgg\_input\_layer) # VGG 블록 생성**

**model = keras.Model(inputs=vgg\_input\_layer, outputs=vgg\_block\_output)**

**model.summary()**