과목 명: 시스템프로그래밍

담당 교수 명: 소 정 민

<<Assignment 5>>

**서강대학교 컴퓨터공학과**

**[20151597]**

**[임준묵]**

목 차

1. 프로그램 개요 3

1.1 3

1.2 3

1.3 3

1.4 3

2. 프로그램 설명 4

2.1 흐름도 4

2.2 문제에 대한 설명 5

3. 모듈 정의 5

3.1 Import modules 5

3.2 Tuning parameters 5

3.3 Plot image 5

3.4 Load dataset 5

3.5 Reshaping the data 5

3.6 Applying One hot encoding for the data 5

3.7 Show data 6

3.8 Creating the DNN model 6

3.9 Adding layers to the model 6

3.10 Visualization the model 6

3.11 Optimizer 6

3.12 Compiling the model 6

3.13 Training the model 6

3.14 Evaluating the model 6

3.15 Prediction the model 7

4. 코드 7

4.1 Import modules 7

4.2 Tuning parameters 7

4.3 Plot image 7

4.4 Load dataset 8

4.5 Reshaping the data 8

4.6 Applying one hot encoding for the data 8

4.7 Show data 8

4.8 Creating the DNN model 8

4.9 Adding layers to the model 8

4.10 Visualization the model 9

4.11 Optimizer 9

4.12 Compiling the model 9

4.13 Training the model 9

4.14 Evaluating the model 9

4.15 Prediction the model 9

# 프로그램 개요

## 

Python3에서 코드를 작성한다.

## 

하드웨어 가속기를 사용하면서 Colaboratory를 가지고 작성한다.

## 

Machine learning library인 Keras를 사용해라

## 

CIFAR-10 데이터 분류에서 75%이상의 정확도를 얻어야 한다.

# 프로그램 설명

## 흐름도

start

Import modules

Plot image

Load dateset

Reshaping the data

Applying one hot encoding for the data

Tuning parameter

Creating model and adding layer to the model

Visualization the model

Compiling the model

Training the model and evaluating

Prediction the model

Show data

## 문제에 대한 설명

10개의 클래스에서 60000개의 32 x 32 색의 이미지로 구성된CIFAR-10 데이터셋을 가지고서 각각의 클래스에 대해서 각각의 사진이 무슨 클래스인지 인식하기 위해서 학습을 진행한다.

총 60000개의 사진이 있는데 10000개는 test용 사진이고 50000개는 학습용 사진이다. 또한 이 dataset은 상호 배타적이다. 예를 들어 자동차와 트럭사이에는 어떠한 겹침이 없다. 즉, 자동차는 세단, suv와 같은 종류의 자동차를 포함하고 트럭은 오직 큰 트럭만을 포함한다. 우리는 이러한 dataset에서 우리가 할 수 있는 여러가지 방법을 이용해서 75%이상의 정확도를 올리면 된다.

# 모듈 정의

흐름도에 나와있는 각각의 단계가 무슨 단계인지 설명하고 각각의 단계에서 정확도를 높이기 위해서 사용한 방법을 설명하겠다.

## Import modules

학습을 하기 위해서 사용되는 모듈들을 import 해준다.

## Tuning parameters

학습에서 사용되는 epoch, learning rate, batch size, class의 갯수를 정해준다.

정확도를 높이기 위해서 여기서 cpoch의 수를 28에서 50으로 올렸다. Epoch는 전체 데이터를 학습하는 횟수를 의미하는데 학습 횟수가 이전에 쓰여있던 28보다 많게 되면 학습이 더 잘될 것이라고 생각하였기 때문이다. 하지만 학습 횟수를 너무 많이 높이면 학습하는 데이터에만 너무 정확하게 맞추려고 하기 때문에 overfit 이 발생하여 cifar-10 dataset 이외의 데이터를 만났을 경우 정확도가 더 떨어질 수 있다. 따라서 이러한 문제를 줄이기 위해서 epoch를 적당하게 50으로 설정하였다.

## Plot image

Image의 출력 형식을 정해준다. Show data와 prediction the model에서 image가 어떻게 출력되는 결정해준다.

## Load dataset

학습할 데이터를 load 해준다. 즉, test 데이터와 train 데이터를 load 해준다.

## Reshaping the data

Load한 데이터의 모양을 다시 정해준다. 여기서는 (50000, 32, 32, 3)으로 정해주었다.

## Applying One hot encoding for the data

One hot encoding이란 머신 러닝에서 categorical variables을 예측하는데 있어서 더 좋게 수행되게 하기 위해서 머신 러닝 알고리즘에 제공되어질 수 있는 형태로 변환되는 과정이다. 따라서 이 변환을 수행하기 위해서 keras.utils.to\_categorical을 사용하였다. 이 함수는 class vector을 binary class 행렬로 변환해준다.

## Show data

이미지를 출력해준다.

## Creating the DNN model

Sequential model을 만들어준다.

## Adding layers to the model

학습 모델에 layer을 추가해준다. Hidden layer을 늘리면 모형의 복잡도가 증가하면서 동시에 정확도가 좋아질 수 있다. 따라서 정확도를 증가시키기 위해서 hidden layer의 개수를 증가시켰다. 하지만 너무 많은 hidden layer을 추가해주게 된다면 gradient가 감소하는 문제가 발생하게 된다. Gradient는 변화량을 의미하는데 이 변화량이 매우 작다면 error rate가 충분히 낮아지지 못한 채 수렴하게 된다. 그리고 hidden layer의 개수를 많이 증가시키게 된다면 모형의 복잡도가 증가하면서 parameter의 개수가 증가하므로 overfitting이 발생할 수도 있다. 따라서 이러한 문제에 직면하지 않기 위해서 적당한 수의 hidden layer을 추가하였다. 그리고 적당한 수의 hidden layer을 추가하였다 하더라고 overfitting이 발생할 수 있기 때문에 drop out을 해줌으로 써 overfitting이 일어날 확률을 줄였다. Drop out 이란 학습을 수행할 때 layer의 모든 뉴런들을 학습하는 것이 아니라 일부 뉴런들을 생략하여 줄어든 신경망을 학습하는 것인데 이는 overfitting의 확률을 줄일 수 있다. 또한 max pooling의 횟수를 증가시켜서 parameter의 개수를 감소시키는 것에 의해서 계산 비용을 줄였다.

## Visualization the model

Addint layers to the model에서 설계한 모델을 보여준다.

## Optimizer

keras모델을 compile하기 위해서 사용되어진다. 이 optimizer은 gradient descent optimizer이다.

## Compiling the model

Model을 compile을 한다. Compile을 한다는 것은 loss function과 optimizer, matrics를 정의하는 것인데 loss function은 crossentropy함수로 정하였고 optimizer은 앞에서 정의하였던 optimizer로 정하였고 metrics는 [‘accuracy’]행렬로 정하였다.

## Training the model

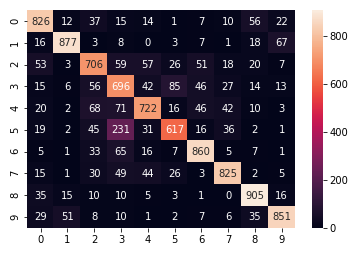
Model을 학습한다. Model을 학습하기 위해서 학습 해야할 데이터 batch size(32로 설정함), epochs(50으로 설정함), verbose(각각의 epoch에 대해서 training의 과정을 어떻게 볼 것인지를 결정한다. 여기서는 1로 설정함), validation\_data(학습을 평가하기 위해서, 모델의 parameter을 맞추기 위해서 사용함, 여기서는 x\_test, y\_test로 설정함)을 설정한다.

## Evaluating the model

학습한 모델을 평가한다. Model.evaluate는 학습 후의 정확도와 loss를 반환해준다. 또한 여기서 정확도와 loss를 출력해준다.

## Prediction the model

Model.predict를 통해서 input 에 대한 예측 결과를 만든다. 또한 이 예측 결과를 이미지와 함께 출력해준다. 또한 confusion matrix를 출력하여서 실제 값들이 어떤 값으로 예측되었는지를 알 수 있다.



# 코드

단계별로 코드를 붙였다.

## Import modules

import keras

from keras.datasets import cifar10

from keras.models import Sequential

from keras.layers import Dense, Dropout, Flatten, Activation

from keras.layers import Conv2D, MaxPooling2D

from keras import backend as K

from keras.utils.vis\_utils import model\_to\_dot

from IPython.display import SVG

%matplotlib inline

import matplotlib.pyplot as plt

from sklearn.metrics import confusion\_matrix

import pandas as pd

import seaborn as 눈

## Tuning parameters

epochs = 50

learning\_rate=0.01

batch\_size = 32

num\_classes = 10

## Plot image

def plot\_images(x, y\_true, y\_pred=None, size=(5, 5)):

assert len(x) == len(y\_true) == size[0] \* size[1]

fig, axes = plt.subplots(size[0], size[1])

fig.subplots\_adjust(hspace=0.1, wspace=0.1)

for i, ax in enumerate(axes.flat):

if x[i].shape[-1] == 1:

ax.imshow(x[i].reshape(x[i].shape[0], x[i].shape[1]))

else:

ax.imshow(x[i])

if y\_pred is None:

xlabel = "True: {0}".format(y\_true[i].argmax())

else:

xlabel = "True: {0}, Pred: {1}".format(y\_true[i].argmax(),

y\_pred[i].argmax())

ax.set\_xlabel(xlabel)

ax.set\_xticks([])

ax.set\_yticks([])

plt.show()

## Load dataset

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

## Reshaping the data

if len(x\_train.shape) < 4:

x\_train = x\_train.reshape(x\_train.shape[0], x\_train.shape[1], x\_train.shape[2], 1)

x\_test = x\_test.reshape(x\_test.shape[0], x\_train.shape[1], x\_train.shape[2], 1)

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

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

x\_train /= 255

x\_test /= 255

print('x\_train shape:', x\_train.shape)

print(x\_train.shape[0], 'train samples')

print(x\_test.shape[0], 'test samples')

## Applying one hot encoding for the data

y\_train = keras.utils.to\_categorical(y\_train, num\_classes)

y\_test = keras.utils.to\_categorical(y\_test, num\_classes)

## Show data

plot\_images(x\_train[:25], y\_train[:25])

## Creating the DNN model

model = Sequential()

## Adding layers to the model

model.add(Conv2D(32, kernel\_size=(3, 3), activation='relu',

input\_shape=x\_train.shape[1:]))

model.add(Conv2D(32, (3, 3), activation='relu'))

model.add(MaxPooling2D(pool\_size=(2, 2)))

model.add(Dropout(0.25))

model.add(Conv2D(64, (3, 3), activation='relu'))

model.add(Conv2D(64, (3, 3), activation='relu'))

model.add(MaxPooling2D(pool\_size=(2, 2)))

model.add(Dropout(0.25))

model.add(Flatten())

model.add(Dense(512, activation='relu'))

model.add(Dropout(0.5))

model.add(Dense(num\_classes, activation='softmax'))

## Visualization the model

SVG(model\_to\_dot(model, show\_shapes=True).create(prog='dot', format='svg'))

## Optimizer

optimizer = keras.optimizers.SGD(lr=learning\_rate)

## Compiling the model

model.compile(loss=keras.losses.categorical\_crossentropy,

optimizer=optimizer,

metrics=['accuracy'])

## Training the model

model.fit(x\_train, y\_train,

batch\_size=batch\_size,

epochs=epochs,

verbose=1,

validation\_data=(x\_test, y\_test))

## Evaluating the model

score = model.evaluate(x\_test, y\_test, verbose=0)

print('Test loss:', score[0])

print('Test accuracy:', score[1])

## Prediction the model

y\_pred = model.predict(x\_test)

plot\_images(x=x\_test[:25], y\_true=y\_test[:25], y\_pred=y\_pred[:25])

y\_result = confusion\_matrix(y\_test.argmax(axis=1), y\_pred.argmax(axis=1))

sns.heatmap(pd.DataFrame(y\_result, range(10), range(10)), annot=True, fmt='g')