과목 명: 시스템프로그래밍

담당 교수 명: 소 정 민

<<Assignment 5>>

**서강대학교 컴퓨터공학과**

**[학번] 20141515**

**[이름] 김연후**

목 차

1. 프로그램 개요 3

2. 프로그램 설명 3

2.1 프로그램 흐름도 3

3. 모듈 정의 3

4. 전역 변수 정의 4

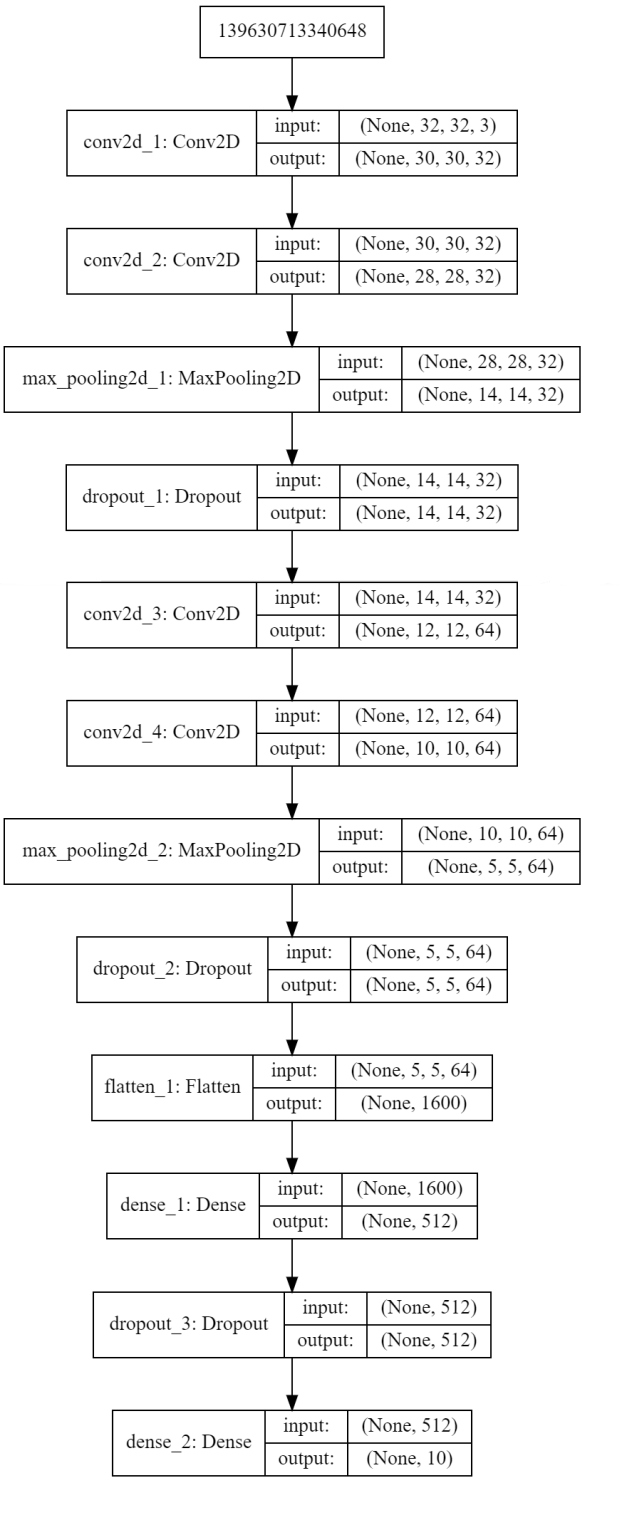
5. 코드 4

# 프로그램 개요

클라우드 서비스와 머신러닝에 대한 이해를 높이는 것이 현 프로젝트의 목표로 이 프로그램은 Python3 코드로 구성되었고, Google의 Colaboratory를 사용하였다. 머신러닝 library로는 Keras를 사용하였고, CIFAR-10 dataset classifications에서 75% 이상의 accuracy를 갖도록 하였다.

# 프로그램 설명

## 프로그램 흐름도

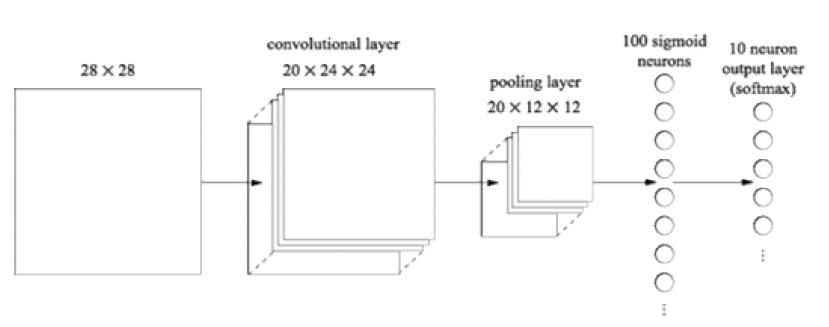


CIFAR-10 dataset을 이용하여 10종류의 사진들 (airplane, automobile, bird, cat, deer, dog, frog, horse, ship, truck)을 각각 구분하도록 하는 것이다. CIFAR-10 dataset은 60000장의 32x32 color image를 갖고 있으며, 이 중 50000장의 training image들과 10000장의 test image들이 있다. 이러한 image들을 알맞은 종류로 구분을 하는 것이고 keras CIFAR-10 CNN(합성곱신경망)을 이용하여 epoch, batch\_size 등을 조절하여 정확도를 75% 이상으로 만드는 것이 문제의 목표다.

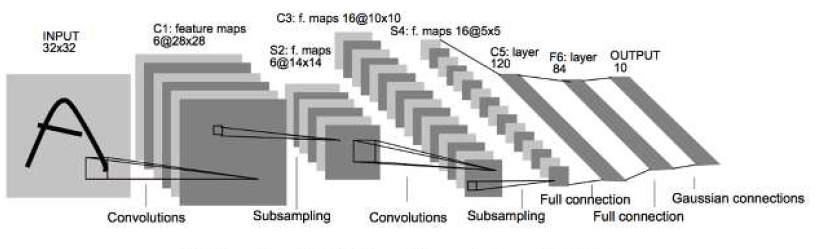
# 모듈 정의

CNN 은 합성곱 신경망으로 convolutional layer과 pooling layer를 이용한다. Convolutional layer에서는 convolution filter를 이용하여 여러 작은 filter가 해당image의 특징점 들을 찾아내어 pooling layer로 넘기고 Max pooling을 이용하여 데이터의 공간적인 크기를 축소한다.

CNN의 정확도를 높이기 위한 방법으로는 여러 가지 방법이 있는데 우선 Fully Connected Layer를 Output Layer 이전에 추가하는 것이다. 이는 pooling layer에서 나온 모든 특징점들을 이용하여 image를 구분할 수 있기에 보다 정확히 image에 대한 정보를 파악하여 구분할 수 있기에 정확도를 높일 수 있다. 그러나 너무 많은 Fully Connected Layer를 사용하는 경우 CNN의 장점이었던 적은 parameter 사용이 묻히는 단점이 있다.



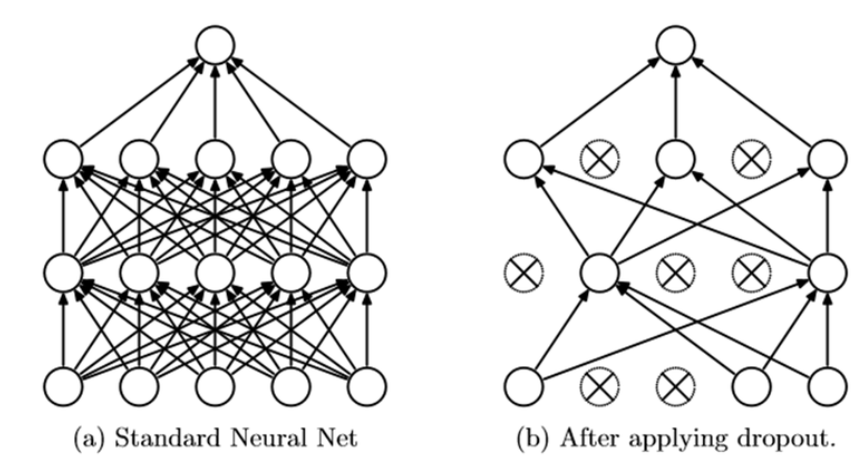
또 다른 방법으로는 convolutional layer를 여러 개를 만드는 방법도 있다. 이는 들어오는 input에 대하여 보다 condensed version의 input이 들어와서 CNN의 성능을 높여준다.



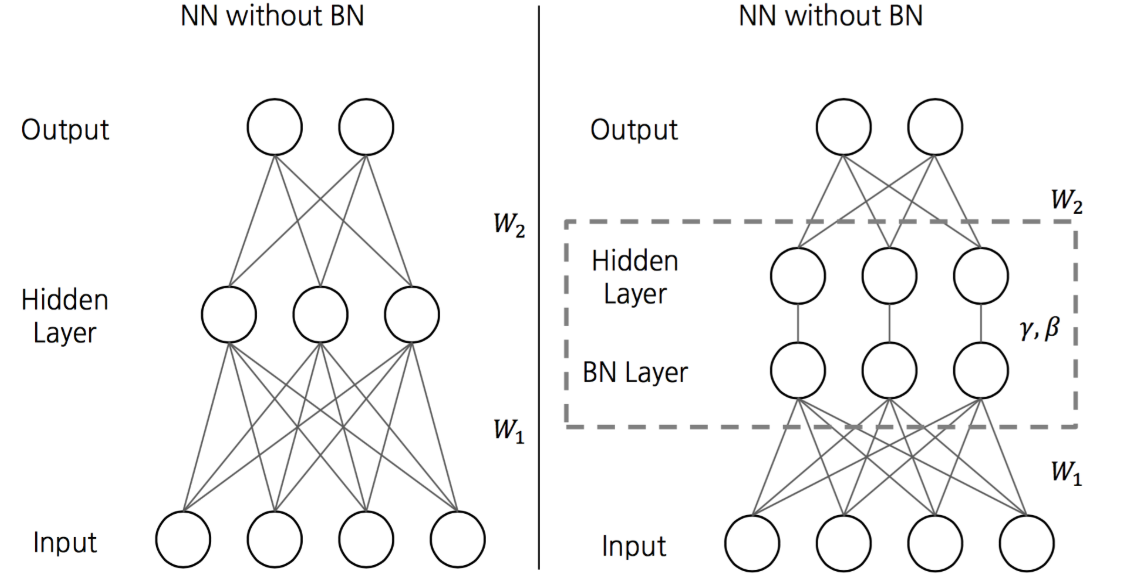
activation function으로 sigmoid function 대신 Relu(Rectifier Linear Units)를 사용하는 경우 또한 하나의 CNN의 정확도 향상에 도움이 되는데 sigmoid function 사용 시 vanishing gradient 문제가 발생하여 정확도가 떨어진다. 따라서 sigmoid 대신 Relu 함수를 사용하여 0보다 작을 때는 0을, 0보다 큰 값에 대해서는 해당값을 그대로 사용하도록 하여 정확도가 올라가도록 할 수 있다.

CNN의 정확도 향상으로 Overfitting의 해결책이 있는데 대표적으로 regularization(일반화)와 data augumentation(학습 데이터를 늘리는 것)이 있다. Regularization의 경우 지나치게 학습 데이터에 집중되는 것을 피하기 위해 일종의 penalty를 부과하는 것으로 Dropout과 Batch Normalization이 있다.

DropOut은 네트워크의 일부를 생략하여 학습을 진행하는 것으로 생략된 네트워크는 학습에 영향을 끼치지 않게되어서 학습데이터에 지나치게 집중되는 것을 피하는 방식이다.



Batch Normalization은 activation function의 출력값을 normalization하는 작업으로 각 hidden layer를 normalization하면서 입력분포가 일정하게 되고, Learning rate를 크게 설정해도 괜찮아지게 된다. 결과적으로는 학습속도가 빨라지게 되는 장점이 있다.



마지막으로 Softmax 를 사용하는 방법이 있는데 이는 10가지 종류로 ‘구분해야 할 것이 정해진’ 상황에서는 정확도를 높여주는 방법이다. Activation function 중 하나인 softmax는 output layer에서 쓰이고, output을 확률분포로 바꿔주어 정해진 구분 종류들을 확률적으로 구분하여 제일 높은 확률을 가진 분류로 정해준다.

# 전역 변수 정의

없음

# 코드

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 sns

epochs = 50

learning\_rate=0.01

batch\_size = 32

num\_classes = 10

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=0.5, 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()

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

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

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

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

model = Sequential()

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'))

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

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

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

optimizer=optimizer,

metrics=['accuracy'])

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

batch\_size=batch\_size,

epochs=epochs,

verbose=1,

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

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

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

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

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')