과목 명: 시스템프로그래밍

담당 교수 명: 소 정 민

<<Assignment 5>>

**서강대학교 컴퓨터공학과**

**[20151547]**

**[박재상]**

목 차

1. 프로그램 개요 3

2. 프로그램 설명 4

2.1 프로그램 흐름도 4

3. 모듈 정의 5

3.1 모듈 이름 : Tuning parameters 5

3.2 모듈 이름: Reshaping the data

3.3 모듈이름: Adding layers to the model 5

5. 삽입한 코드 7~28

# 프로그램 개요

; 해당 Assignment의 프로그램에 대한 기능 및 특이점에 대한 전반적인 개요에 대해서 설명한다.

이번 프로젝트의 목표는 Machine Learning에 대해 이해하고 이를 응용하여 CIFAR-10에서 Accuracy를 75%이상으로 만드는 것이다.

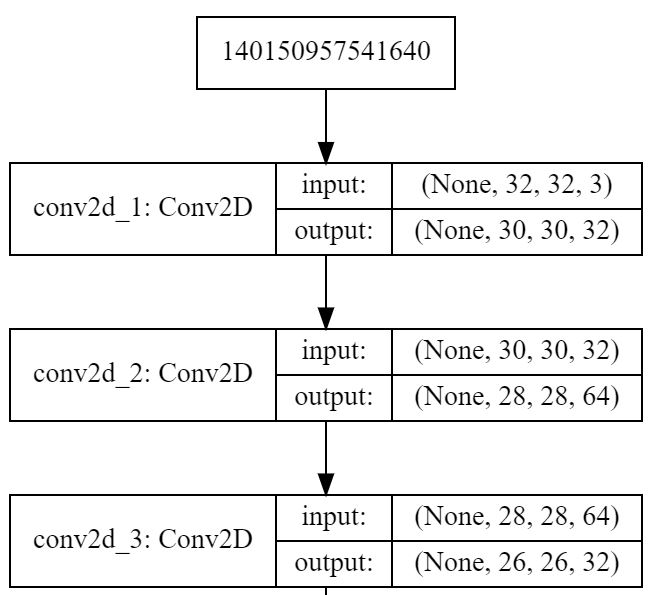
제공되는 train data를 통해서 Machine Learning을 진행하고 test data를 통해서 해당 machine의 validation을 측정한다.

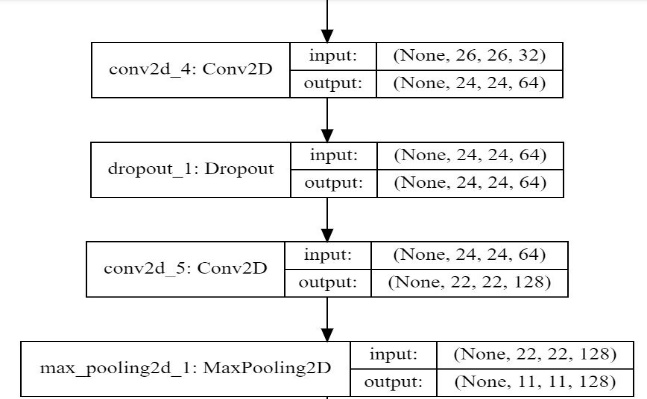
Multi perceptron으로 구성된 layer를 여러겹 쌓아 프로그램을 구성하고 train data를 통해서 각 perceptron에 적절한 weight를 할당한다. 이 과정을 epochs 만큼 반복하여 기계를 학습시킨다. Epochs 가 높을수록 많은 반복학습을 진행하지만, train data에 overfit될 수 있어서 적절한 기법을 활용해야 한다. 또한 weight의 변화량을 learning rate로 조절할 수 있는데, learning rate가 낮을수록 weight의 변화량을 작게 가져가서 accuracy가 큰폭으로 진동하는 것을 막을 수 있지만 학습되는 속도가 느려 시간이 오래 걸린다.

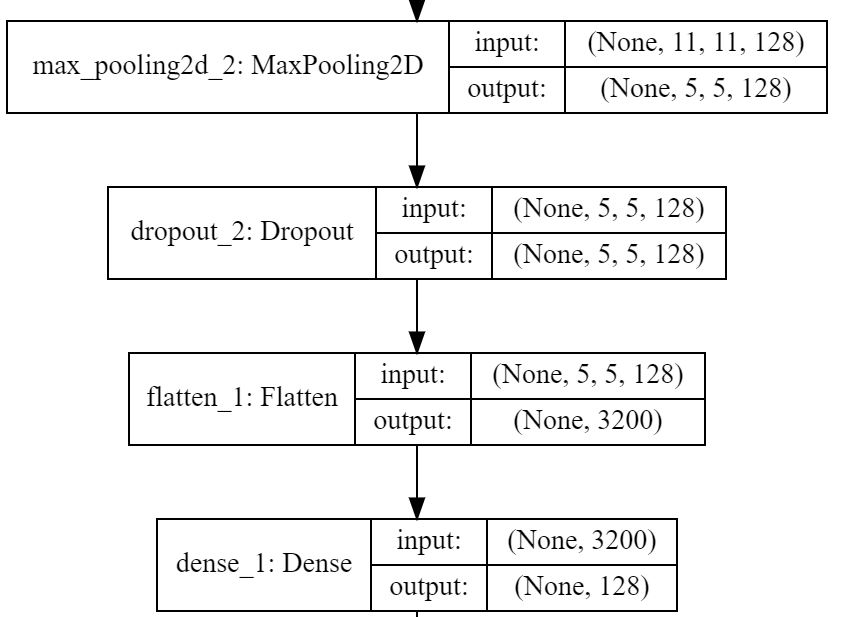
이러한 여러가지 변인들을 잘 조절하여 75%이상의 accuracy를 갖는 프로그램을 만드는 것이 이번 프로젝트의 목표이다.

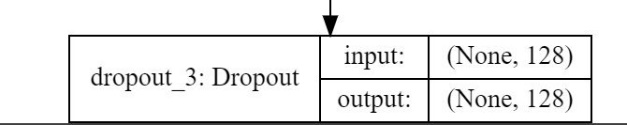
# 프로그램 설명

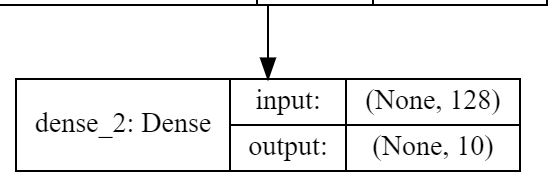
2-1 프로그램 흐름도











2-2

데이터가 처음에 들어오면 4개의 convolution layer를 거치게 된다.

원래는 5개의 convolution layer를 연결하였으나 나중에 overfit현상이 심하게 발생해서 4개의 layer를 거친 뒤 drop을 통해서 overfit현상을 최소화 시켰다.

그리고 나서 하나의 convolution layer를 추가하였다.

이미지들을 봤을 때, 비슷한 값들이 많은 특히 배경부분은 굳이 계산을 오래할 필요가 없다고 판단했고 이미지파일들을 봤을 때 이러한 영역이 많이 발견되어서 MaxPooling을 통해서 주변의 사소한 변화를 무시하도록 했다.

그리고 행렬의 형태로 들어오는 데이터를 1차원 데이터로 serialization을 시키는 flatten과정을 거치게 했다.

직렬화된 데이터들을 다시 128개의 카테고리로 나누고 drop을 한 번 더 거쳐서 마지막으로 10개의 카테고리로 나누면서 프로그램이 마무리된다.

# 모듈 정의

## Tuning parameters

기존의 epochs는 10이었다. Machine learning을 진행해보니 10번의 epoch로는 train data에 대해서도 accuracy가 턱없이 모자랐다. 그래서 충분한 반복이 필요하다고 판단하여 epochs를 100으로 설정했다.

기존의 learning rate는 0.01이었다. Epochs를 100으로 증가시켰지만 0.01의 learning rate는 너무 미진하게 진행되어서 0.05로 증가시켰다. 처음에는 0.1로 증가시켰지만 특정 accuracy에서는 accuracy가 증가하지않고 증감을 반복하는 진동상태에 빠지는 상황이 자주 연출되어서 0.05로 감소시켰다.

## Reshaping the data

프로젝트 진행 초반부에는 train\_data와 test\_data 값을 적절히 바꾸어 weight값을 계산할 때 좀 더 빠르게 계산할 수 있도록 만드는 것이 계획이었다. 그래서 train data와 test data를 0.5씩 평행이동을 시켜 0이 되는 항이 많아지도록 유도했다. 그랬더니 초반에는 잘 진행이 되다가 너무 심하게 train data set에 overfit이 되어서 ( 10 epochs만에 train data 에 대한 accuracy가 90%를 넘었다.)

test data에 대해서는 accuracy가 60%이하로 내려가는 현상이 발생했다.

아마도 train data들 중에는 RGB값이 0.5에 해당하는 부분이 많아서 그에 해당하는 weight값이 초기에 설정된 random값에서 거의 변동이 없이 learning이 진행되었고 test data들 중에는 RGB값이 0.5에 해당하는 부분이 적고 다른 값에 해당하는 data가 많아서 accuracy가 엄청나게 떨어지게 되었다고 생각한다. 그래서 data를 직접적으로 바꾸는 작업은 진행하지 않았다.

## Adding layers to model

사실 hidden layer를 추가하는 과정에서 상당히 많은 시도를 해보았다.

Model.add(Conv2D( something,(3,3),activation=’relu’)를 통해 필터를 여러가지 추가도 해봤고

Model.add(MaxPooling2D(pool\_size=(2,2))를 통해 이미지의 fragment의 작은 변화를 무시하는 과정도 추가해봤고 Model.add(Dropout(something))을 통해 overfit현상을 억제하는 과정도 추가해봤다.

필터를 많이 넣을수록 learning하는 시간도 오래 걸렸다. 그리고 overfit현상이 너무 많이 발생하여 기존의 필터 말고는 더 추가하지 않았다.

그리고 epochs가 100이다보니 후반부에 갈수록 overfit현상이 두드러지게 나타났다. Train data에 대해서 accuracy가 97%까지 치솟는 엄청난 부작용이 일어나서 Dropout을 이용하여 특정 feature에 대한 weight를 줄이고 전반적인 feature에 골고루 weight가 할당되도록 유도했다.

그리고 dense를 통해서 sigmoid를 써볼까 생각도 했지만 weight값이 random하게 주어지다보니 양 극값에 가까운 weight값이 설정되면 학습진행도가 거의없다는 치명적인 단점이 있어서 ‘relu’옵션을 통해 진행하였다.

# 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 = 70

learning\_rate=0.05

batch\_size = 128

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(64, (3, 3), activation='relu'))

model.add(Dropout(0.5))

model.add(Conv2D(128, (3, 3), activation='relu'))

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

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

model.add(Dropout(0.25))

model.add(Flatten())

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

model.add(Dropout(0.5))

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

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')