과목 명: 시스템프로그래밍

담당 교수 명: 소 정 민

<<Assignment 5>>

**서강대학교 컴퓨터공학과**

**[20151561]**

**[신용하]**

목 차

1. 프로그램 개요 3

2. 프로그램 설명 3

2.1 프로그램 흐름도 3

3. 모듈 정의 4

3.1 모듈 이름 : 전체 네트워크의 흐름 4

3.2 모듈 이름: 정확도를 높이기 위한 노력 4

3.2.1 epoch 4

3.2.2 learning rate 4

3.2.3 batch size 4

3.2.4 hidden layer 추가 5

4. 코드 5

# 프로그램 개요

머신 러닝에 대한 이해도를 높이고 클라우드 서비스의 필요성에 대해 알아본다.

# 프로그램 설명

Colaboratory 의 머신 러닝 지원 기능을 이용하여 CIFAR-10 데이터를 75% 이상의 정확도로 분류할 수 있게 한다. CIFAR-10 dataset은 32 X 32 크기의 한 사진이 60000장 들어있고, 각각은 10 항목 중 한 곳에 해당된다. 모든 항목은 6000장씩 가진다. 또한, 한 사진은 한 항목에만 해당된다. 이 사진들을 75% 이상의 정확도로 해당 항목을 맞출 수 있도록 여러 방법을 시도해 보는 것이 목표이다.

## 프로그램 흐름도

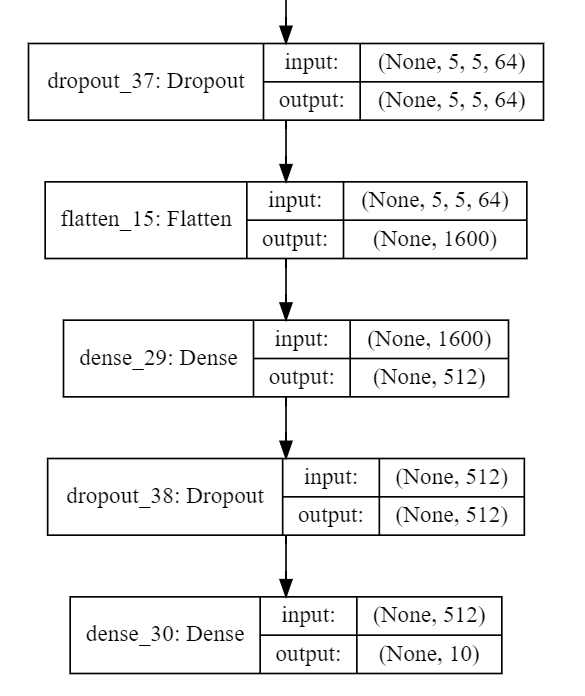
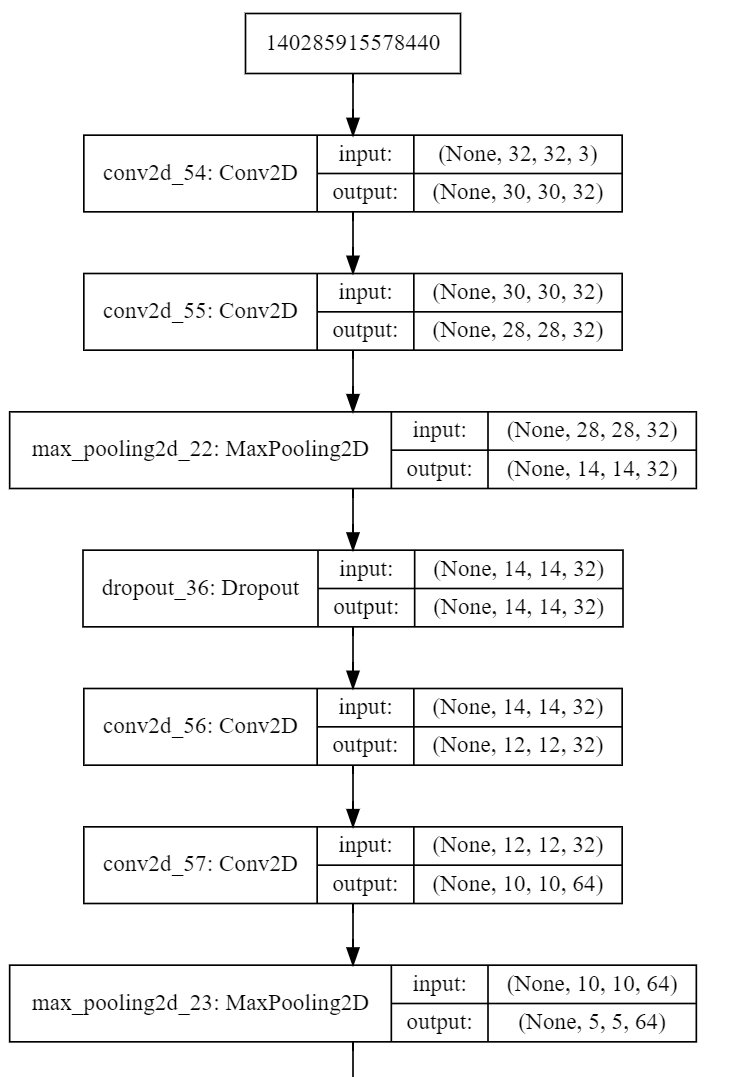


그림 1> 다이어그램

# 모듈 정의

## 전체 네트워크의 흐름

1) machine learning을 지원하는 keras 모듈을 import한다.

2) epochs, learning rate, batch size, num classes를 설정한다.

epoch: 학습용 사진 전체를 사용하였을 때 한 세대(epoch)가 지났다고 한다. CIFAR-10의 경우 50000장의 train, 10000장의 test 샘플로 이루어져 있기 때문에 한 세대는 50000장의 사진에 대해 학습하는 것을 의미한다. epoch의 숫자에 따라 이 50000장을 train 하는 것을 반복한다.

learning rate: 뉴런의 weight를 수정할 때, 이를 얼마만큼의 비율로 수정할 것인지 정하는 것이다. 너무 작으면 값이 거의 바뀌지 않을 수 있고, 너무 크면 목표로 하는 근사치를 벗어날 수 있어 적절한 값이 필요하다.

batch size: 한 번에 처리하는 사진의 장수를 의미한다.

num classes: CIFAR-10의 사진 종류는 10이기 때문에 num classes는 10이다.

3) train set과 test set을 로드.

4) 크기가 작은 사진의 크기를 크게 맞춤.

5) sequential 모델 사용: 한 sequential 모델은 하나의 층(layer)를 의미한다.

6) layer를 직접 쌓는다.

MaxPooling: 사진의 특징들을 추출한다.

Flatten: 2차원 배열을 1차원으로 표현하는 데 쓰인다. ex) 10 X 10을 100의 1차원 배열로 만듬.

Dropout: MaxPooling을 통해 추출한 특징들이 사실 그 항목의 특징이 아닌데 우연히 test set의 사진들에 그 특징이 들어가 있었다면, 그러한 특징이 없는 항목의 사진에 대해서는 틀린 답을 말할 것이다. 이를 줄이기 위해 hidden layer들 중 일부를 사용하지 않아 그러한 특징들이 도출되는 것을 최대한 줄이는 것이 목표이다. 입력 받는 숫자의 비율로 hidden layer들 중 일부를 사용하지 않는다.

Dense: input layer와 output layer를 모두 연결한다.

7) optimizer: gradient descent를 적용하여 learning rate만큼 곱한 것으로 weight를 수정한다.

8) 모델을 컴파일한다.

9) 모델을 학습시킨다.

loss: 매 epoch 마다의 훈련 손실값

acc: 매 epoch 마다의 훈련 정확도

val\_loss: 매 에포크 마다의 검증 손실값

val\_acc: 매 에포크 마다의 검증 정확도

10) 학습의 loss와 accuracy를 확인한다.(75% 이상)

## 정확도를 높이기 위한 노력

### epoch

epoch를 늘리면 더 많은 횟수를 반복하므로 정확도를 늘릴 수 있다. 다만, 수행 시간이 길어진다.

기존의 10회를 50회로 늘렸더니 단순 epoch 만으로 65%의 정확도까지 접근할 수 있었다.

### learning rate

learning rate는 기존의 1을 0.01로 설정하였다. learning rate를 낮출수록 더 정밀한 weight의 수정이 가능하다. 그러나 50회의 epoch에서는 0.01 보다 낮은 learning rate는 학습을 너무 더디게 하여 75%의 정확도에 미치지 못하고 종료되었다. epoch를 100회로 하면 learning rate를 더 낮추는 것도 가능할 것 같다. 대신 수행 시간이 더 오래 걸릴 것이다.

### batch size

batch size를 기존의 128에서 32로 줄였다. 큰 batch size는 gradient를 정확히 찾는 데 도움을 주고, 수행 속도가 빠른 반면, 작은 batch size는 수행 속도는 느리고 performance가 낮아도 수행에 문제가 없고 weight가 자주 수정되므로 learning rate의 넓은 범위 사용을 가능하게 한다.

### hidden layer 추가

hidden layer를 한 층 더 쌓아서 정확도를 78% 정도까지 끌어올릴 수 있었다.

# 코드

# 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 sns

# 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=0.5, 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()

# 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(32, (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')