

인공 신경망과 딥 러닝 실습 - 심화

원 중 호

서울대학교 통계학과

Dec. 2015

목차

- 딥 뉴럴 네트워크 코드
- 딥 러닝 데이터와 결과 처리를 위한 python 기초
- MNIST 데이터를 위한 deep convolutional network
- CIFAR10 데이터를 위한 deep convolutional network

딥 뉴럴 네트워크 코드

- 다운로드 / Github
- `mnist_shallow.py`
 - ▶ MNIST 데이터의 구분을 위한 얇은 네트워크(shallow network)를 구성해 훈련하고, 훈련을 마친 네트워크의 정확도를 확인하는 코드
- `mnist_shallow2.py`
 - ▶ 더 많은 변수를 가지는 얇은 네트워크 모델을 구성해 훈련하고, 훈련을 마친 네트워크의 정확도를 확인하는 코드
- `mnist_cnn_shallow.py`
 - ▶ CNN(Convolutional Neural Network) 네트워크 구조를 이용해 MNIST 데이터를 훈련하고, 훈련을 마친 네트워크의 정확도를 확인하는 코드
- `mnist_cnn.py`
 - ▶ CNN 네트워크 구조를 더 쌓아 구성한 deep convolutional Network를 이용해 MNIST 데이터를 훈련하고, 훈련을 마친 네트워크의 정확도를 확인하는 코드

딥 뉴럴 네트워크 코드 (Cont'd)

- `mnist_cnn_part2.py`

- ▶ CNN 네트워크 구조를 이용해 MNIST 데이터를 훈련하고, 훈련을 마친 네트워크의 정확도를 확인하는 코드
- ▶ Model, weight, history를 저장

- `cifar10_cnn.py`

- ▶ CNN 네트워크 구조를 이용해 CIFAR-10 데이터를 훈련하고, 훈련을 마친 네트워크의 정확도를 확인하는 코드
- ▶ Model, weight, history를 저장

- `test_mnist_cnn.py`

- ▶ MNIST 데이터를 이용해 훈련된 네트워크를 불러와, model, history, 각 층의 결과를 확인하는 코드

- `test_cifar10_cnn.py`

- ▶ CIFAR-10 데이터를 이용해 훈련된 네트워크를 불러와, model, history, 각 층의 결과를 확인하는 코드

실습 - GPU 사용

- GPU 사용이 가능한 컴퓨터의 경우, Theano에서 제공하는 GPU 병렬화 기능을 사용하기 위해서는 Theano flag를 GPU 계산을 위해 셋팅해야 함.
 - ▶ 예를 들어 `mnist_shallow.py` 프로그램을 GPU에서 실행시키려 할 경우, Command-line Prompt에 다음과 같이 입력해 Theano flag 셋팅 Theano flag를 GPU 계산을 위해 셋팅 후 코드를 실행
- ```
> THEANO_FLAGS=mode=FAST_RUN,device=gpu,floatX=float32
> python mnist_cnn_part2.py
```

## Section 1

딥 러닝 데이터와 결과 처리를 위한 python 기초

# 실습 목표

- ① Keras 라이브러리의 다양한 데이터 베이스 활용
- ② 딥 러닝을 위한 Numpy 데이터 처리
- ③ 데이터 확인을 위한 matplotlib 기초

# Keras Datasets 활용

- Image data : CIFAR10 image classification, MNIST database of handwritten digits
- Word data : IMDB Movie reviews sentiment classification, Reuters newswire topics classification
- `(X_train, y_train), (X_test, y_test) = datasets.load_data(param)`
  - ▶ `X_train, X_test`: dd
  - ▶ `y_train, y_test`: 정수형 라벨



# 딥 러닝을 위한 Numpy 데이터 처리

- 딥 러닝을 위한 Numpy 데이터의 형태 변형
- `data_numpy.py`

```
from keras.utils import np_utils
import numpy as np
```

```
a = np.array([0,0,1,1,2,2,1,1,3,3,0,0])
b = np.array([(0,1,2),(1,3,0)])
```

```
print 'a'
print 'type=',type(a), ', shape=', a.shape, ', ndim=', a.ndim
print 'data type=', a.dtype.name
print 'item size=', a.itemsize, ', size=', a.size
```

```
print 'b'
print 'type=',type(b), ', shape=', b.shape, ', ndim=', b.ndim
print 'data type=', b.dtype.name
print 'item size=', b.itemsize, ', size=', b.size
```

# 딥 러닝을 위한 Numpy 데이터 처리 (Cont'd)

- 딥 러닝을 위한 Numpy 데이터의 형태 변형

```
a = a.reshape(6,2)
```

```
b = b.reshape(6)
```

```
a = a.astype('float32')
```

```
c = np_utils.to_categorical(b, 4)
```

```
print 'a'
```

```
print 'type=',type(a), ', shape=', a.shape, ', ndim=', a.ndim
```

```
print 'data type=', a.dtype.name
```

```
print 'item size=', a.itemsize, ', size=', a.size
```

```
print 'c'
```

```
print 'type=',type(c), ', shape=', c.shape, ', ndim=', c.ndim
```

```
print 'data type=', c.dtype.name
```

```
print 'item size=', c.itemsize, ', size=', c.size
```

# 딥 러닝을 위한 Numpy 데이터 처리 (Cont'd)

- 딥 러닝을 위한 Numpy 데이터의 형태 변형

```
from keras.models import Sequential
from keras.layers.core import Dense, Activation
from keras.optimizers import SGD

model = Sequential()
model.add(Dense(4, init='uniform', input_shape=(2,), activation='softmax'))
model.compile(loss='mse', optimizer='sgd')
hist = model.fit(a, c, batch_size=1, nb_epoch=3,
 show_accuracy=True, verbose=1)
print hist.history['acc']
```

# 딥 러닝을 위한 Numpy 데이터 처리 (Cont'd)

- 딥 러닝을 위한 Numpy 데이터의 형태 변형

```
C:\Windows\system32\cmd.exe
e:\work\Wdnn-cs-posco>python data_numpy.py
a
type= <type 'numpy.ndarray'> , shape= (12L,) , ndim= 1
data type= int32
item size= 4 , size= 12
b
type= <type 'numpy.ndarray'> , shape= (2L, 3L) , ndim= 2
data type= int32
item size= 4 , size= 6
a
type= <type 'numpy.ndarray'> , shape= (6L, 2L) , ndim= 2
data type= float32
item size= 4 , size= 12
c
type= <type 'numpy.ndarray'> , shape= (6L, 4L) , ndim= 2
data type= float64
item size= 8 , size= 24
Epoch 1/3
6/6 [=====] - 0s - loss: 0.1873 - acc: 0.6667
Epoch 2/3
6/6 [=====] - 0s - loss: 0.1871 - acc: 0.6667
Epoch 3/3
6/6 [=====] - 0s - loss: 0.1868 - acc: 0.3333
[0.6666666666666663, 0.6666666666666663, 0.3333333333333331]
e:\work\Wdnn-cs-posco>
```

## 딥 러닝을 위한 Numpy 데이터 처리 (Cont'd)

- PNG 이미지로 딥 러닝을 위한 Numpy 데이터 만들기
- `data_png.py`

```
import matplotlib.pyplot as plt
import matplotlib.image as mpimg
import numpy as np
from keras.models import Sequential
from keras.layers.core import Dense, Activation, Flatten, Dropout
from keras.layers.convolutional import Convolution2D, MaxPooling2D
from keras.utils import np_utils
```

```
img = mpimg.imread('../prep-dnn-lecture/figure/lena.png')
print 'img shape', img.shape
plt.imshow(img, cmap='bone')
plt.show()
```

```
img = img.reshape(1,1,512,512)
print 'type=', type(img), ', shape=', img.shape, ', ndim=', img.ndim
print 'data type=', img.dtype.name
print 'item size=', img.itemsize, ', size=', img.size
```

# 딥 러닝을 위한 Numpy 데이터 처리 (Cont'd)

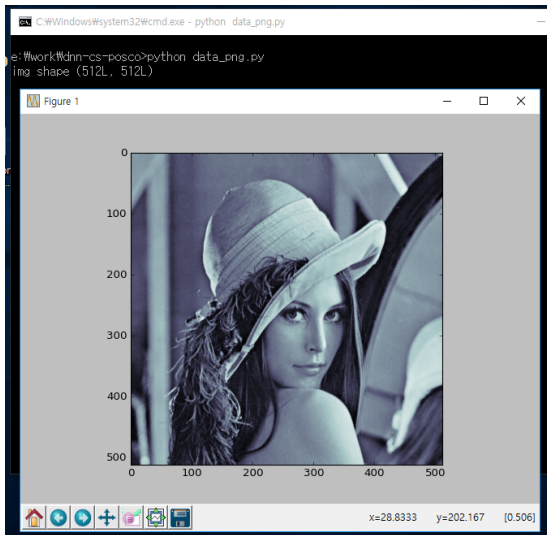
- PNG 이미지로 딥 러닝을 위한 Numpy 데이터 만들기

```
b = np.array([[0., 1.]])
```

```
model = Sequential()
model.add(Convolution2D(2, 3, 3, border_mode='valid',
 input_shape=(1, 512, 512)))
model.add(Activation('sigmoid'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.25))
model.add(Flatten())
model.add(Dense(2))
model.add(Activation('softmax'))
model.compile(loss='mse', optimizer='sgd')
hist = model.fit(img, b, batch_size=1, nb_epoch=2,
 show_accuracy=True, verbose=1)
print hist.history['acc']
```

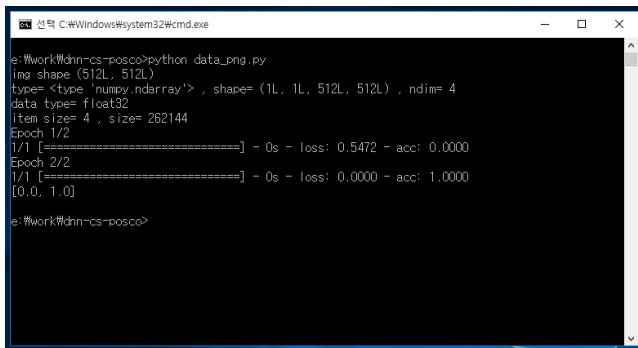
# 딥 러닝을 위한 Numpy 데이터 처리 (Cont'd)

- PNG 이미지로 딥 러닝을 위한 Numpy 데이터 만들기



# 딥 러닝을 위한 Numpy 데이터 처리 (Cont'd)

- PNG 이미지로 딥 러닝을 위한 Numpy 데이터 만들기



```
선택 C:\Windows\system32\cmd.exe
e:\work\dhn-cs-posco>python data_png.py
img shape (512L, 512L)
type= <type 'numpy.ndarray'> , shape= (1L, 1L, 512L, 512L) , ndim= 4
data type= float32
item size= 4 , size= 262144
Epoch 1/2
1/1 [=====] - 0s - loss: 0.5472 - acc: 0.0000
Epoch 2/2
1/1 [=====] - 0s - loss: 0.0000 - acc: 1.0000
[0.0, 1.0]

e:\work\dhn-cs-posco>
```



## 딥 러닝을 위한 Numpy 데이터 처리 (Cont'd)

- PNG color 이미지로 딥 러닝을 위한 Numpy 데이터 만들기
- data\_png2.py

```
import matplotlib.pyplot as plt
import matplotlib.image as mpimg
import numpy as np
from keras.models import Sequential
from keras.layers.core import Dense, Activation, Flatten, Dropout
from keras.layers.convolutional import Convolution2D, MaxPooling2D
from keras.utils import np_utils
```

```
img = mpimg.imread('../prep-dnn-lecture/figure/lena_color.png')
print 'img shape', img.shape
plt.imshow(img)
plt.show()
```

```
img = img.transpose(2, 0, 1)
img = img.reshape(1, 3, 512, 512)
print 'type=', type(img), ', shape=', img.shape, ', ndim=', img.ndim
print 'data type=', img.dtype.name
print 'item size=', img.itemsize, ', size=', img.size
```

## 딥 러닝을 위한 Numpy 데이터 처리 (Cont'd)

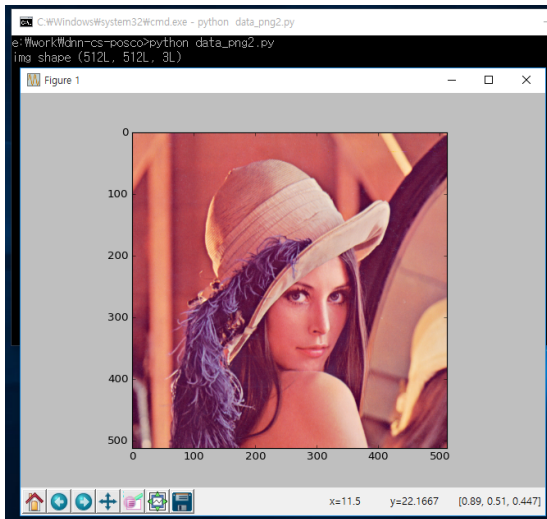
- PNG color 이미지로 딥 러닝을 위한 Numpy 데이터 만들기

```
b = np.array([[1., 0., 0.], [0., 1., 0.], [0., 0., 1.]])
```

```
model = Sequential()
model.add(Convolution2D(2, 3, 3, border_mode='valid',
 input_shape=(3, 512, 512)))
model.add(Activation('sigmoid'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.25))
model.add(Flatten())
model.add(Dense(3))
model.add(Activation('softmax'))
model.compile(loss='mse', optimizer='sgd')
hist = model.fit(img, b, batch_size=1, nb_epoch=2,
 show_accuracy=True, verbose=1)
print hist.history['acc']
```

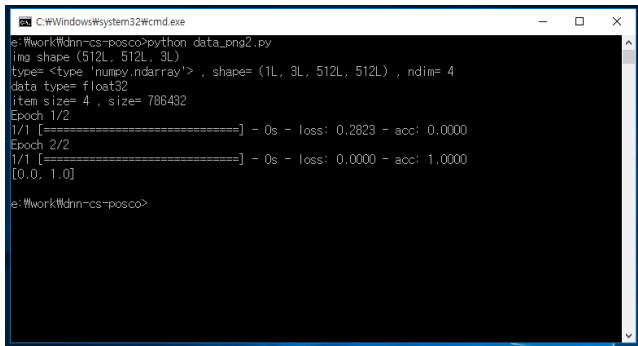
# 딥 러닝을 위한 Numpy 데이터 처리 (Cont'd)

- PNG color 이미지로 딥 러닝을 위한 Numpy 데이터 만들기



# 딥 러닝을 위한 Numpy 데이터 처리 (Cont'd)

- PNG color 이미지로 딥 러닝을 위한 Numpy 데이터 만들기



```
C:\Windows\system32\cmd.exe
e:\work\ldnn-cs-posco>python data_png2.py
img shape (512L, 512L, 3L)
type= <type 'numpy.ndarray'> , shape= (1L, 3L, 512L, 512L) , ndim= 4
data type= float32
item size= 4 , size= 786432
Epoch 1/2
1/1 [=====] - 0s - loss: 0.2823 - acc: 0.0000
Epoch 2/2
1/1 [=====] - 0s - loss: 0.0000 - acc: 1.0000
[0.0, 1.0]
e:\work\ldnn-cs-posco>
```

## Section 2

# MNIST 데이터를 위한 deep convolutional network

# 실습 목표

- **MNIST datasets** : [Sample\_size, Rows, Cols]
  - ▶ Rows = 28, Cols=28
  - ▶ Sample\_size : 60000개의 training set, 10000개의 test set
- ① MNIST 데이터를 이용해 훈련한 네트워크의 model, weight, history 저장하기
- ② MNIST 데이터를 이용해 훈련한 네트워크 불러오기
- ③ History 확인 하기
- ④ 각 층의 결과 확인 하기

# 실습

- mnist\_cnn\_part2.py

- ▶ 20개의 epoch, mini-batch size = 128
- ▶ IDLE에서 File-New File(Ctrl+N) 클릭 후, 새 파일에 다음을 입력하고 mnist\_cnn\_part2.py로 저장

```
from __future__ import print_function
import numpy as np
np.random.seed(1337) # for reproducibility
import json
```

```
from keras.datasets import mnist
from keras.models import Sequential
from keras.layers.core import Dense, Dropout, Activation, Flatten
from keras.layers.convolutional import Convolution2D, MaxPooling2D
from keras.utils import np_utils
```

```
batch_size = 128
nb_classes = 10
nb_epoch = 20
```

## 실습 (Cont'd)

```
input image dimensions
img_rows, img_cols = 28, 28
number of convolutional filters to use
nb_filters = 3
convolution kernel size
nb_conv = 5
size of pooling area for max pooling
nb_pool = 2

the data, shuffled and split between tran and test sets
(X_train, y_train), (X_test, y_test) = mnist.load_data()

X_train = X_train.reshape(X_train.shape[0], 1, img_rows, img_cols)
X_test = X_test.reshape(X_test.shape[0], 1, img_rows, img_cols)
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')
```



## 실습 (Cont'd)

```
convert class vectors to binary class matrices
Y_train = np_utils.to_categorical(y_train, nb_classes)
Y_test = np_utils.to_categorical(y_test, nb_classes)

model = Sequential()
model.add(Convolution2D(nb_filters, nb_conv, nb_conv,
 border_mode='valid',
 input_shape=(1, img_rows, img_cols)))
model.add(Activation('sigmoid'))
model.add(Convolution2D(nb_filters, nb_conv, nb_conv))
model.add(Activation('sigmoid'))
model.add(MaxPooling2D(pool_size=(nb_pool, nb_pool)))
model.add(Dropout(0.25))

model.add(Flatten())
model.add(Dense(128))
model.add(Activation('sigmoid'))
model.add(Dropout(0.5))
model.add(Dense(nb_classes))
model.add(Activation('softmax'))
```

## 실습 (Cont'd)

```
model.compile(loss='categorical_crossentropy', optimizer='adadelta')

hist = model.fit(X_train, Y_train, batch_size=batch_size,
 nb_epoch=nb_epoch, show_accuracy=True,
 verbose=1, validation_split=0.2)

score = model.evaluate(X_test, Y_test, show_accuracy=True, verbose=0)
print('Test score:', score[0])
print('Test accuracy:', score[1])
```

- ① MNIST 데이터를 이용해 훈련한 네트워크의 model, weight, history 저장하기

```
Save the model and weights
json_string = model.to_json()
open('mnist_model_architecture.json', 'w').write(json_string)
model.save_weights('mnist_model_weights.h5')

Save History
with open('mnist_model_history.json', 'w') as fp:
 json.dump(hist.history, fp)
```

## 실습 (Cont'd)

- `mnist_cnn_part2.py`

- ▶ 20개의 epoch, mini-batch size = 128
- ▶ WinPython Command Prompt내에서 다음과 같이 입력해 실행

> python 파일 경로\mnist\_cnn\_part2.py

# 실습 (Cont'd)

- mnist\_cnn\_part2.py
  - 20개의 epoch, mini-batch size = 128

```
C:\Windows\system32\cmd.exe
45568/48000 [=====>.] - ETA: 1s - loss: 0.1573 - acc: 0.
45696/48000 [=====>.] - ETA: 1s - loss: 0.1572 - acc: 0.
45824/48000 [=====>.] - ETA: 0s - loss: 0.1574 - acc: 0.
45952/48000 [=====>.] - ETA: 0s - loss: 0.1574 - acc: 0.
46080/48000 [=====>.] - ETA: 0s - loss: 0.1574 - acc: 0.
46208/48000 [=====>.] - ETA: 0s - loss: 0.1573 - acc: 0.
46336/48000 [=====>.] - ETA: 0s - loss: 0.1573 - acc: 0.
46464/48000 [=====>.] - ETA: 0s - loss: 0.1572 - acc: 0.
46592/48000 [=====>.] - ETA: 0s - loss: 0.1572 - acc: 0.
46720/48000 [=====>.] - ETA: 0s - loss: 0.1572 - acc: 0.
46848/48000 [=====>.] - ETA: 0s - loss: 0.1573 - acc: 0.
46976/48000 [=====>.] - ETA: 0s - loss: 0.1572 - acc: 0.
47104/48000 [=====>.] - ETA: 0s - loss: 0.1571 - acc: 0.
47232/48000 [=====>.] - ETA: 0s - loss: 0.1573 - acc: 0.
47360/48000 [=====>.] - ETA: 0s - loss: 0.1573 - acc: 0.
47488/48000 [=====>.] - ETA: 0s - loss: 0.1574 - acc: 0.
47616/48000 [=====>.] - ETA: 0s - loss: 0.1574 - acc: 0.
47744/48000 [=====>.] - ETA: 0s - loss: 0.1572 - acc: 0.
47872/48000 [=====>.] - ETA: 0s - loss: 0.1572 - acc: 0.
48000/48000 [=====] - 24s - loss: 0.1574 - acc: 0.9524
- val_loss: 0.0782 - val_acc: 0.9772
Test score: 0.0767079292386
Test accuracy: 0.9758
e:\work\dnn-cs-posco>
```

# 실습 (Cont'd)

- `test_mnist_cnn.py`
  - ▶ IDLE에서 File-New File(Ctrl+N) 클릭 후, 새 파일에 다음을 입력하고 `test_mnist_cnn.py`로 저장

```
from __future__ import print_function

from keras.datasets import mnist
from keras.models import Sequential

from keras.models import model_from_json
import matplotlib.pyplot as plt
import theano
import json

batch_size = 128
nb_classes = 10
nb_epoch = 20
img_rows, img_cols = 28, 28
```

## 실습 (Cont'd)

```
the data, shuffled and split between tran and test sets
(X_train, y_train), (X_test, y_test) = mnist.load_data()
X_test = X_test.reshape(X_test.shape[0], 1, img_rows, img_cols)
X_test = X_test.astype('float32')
X_test /= 255
print(X_test.shape[0], 'test samples')
```

### ② MNIST 데이터를 이용해 훈련한 네트워크 불러오기

```
Reconstruct model
model = model_from_json(open('mnist_model_architecture.json').read())
model.load_weights('mnist_model_weights.h5')
```

## 실습 (Cont'd)

### ③ History 확인 하기

```
Plot history
```

```
hist = json.loads(open('mnist_model_history.json').read())
```

```
plt.figure('history')
plt.subplot(211)
plt.title('Loss over epochs')
plt.plot(hist['loss'], 'r', label='loss')
plt.plot(hist['val_loss'], 'b', label='val_loss')
plt.xlabel('Epochs')
plt.ylabel('Loss')
```

```
plt.subplot(212)
plt.title('Accuracy over epochs')
plt.plot(hist['acc'], 'r', label='acc')
plt.plot(hist['val_acc'], 'b', label='val_acc')
plt.xlabel('Epochs')
plt.ylabel('Accuracy')
```

```
plt.tight_layout()
plt.show()
```

## 실습 (Cont'd)

### ④ 각 층의 결과 확인 하기

*# Get output of each layer*

```
get_1st_layer_output = theano.function([model.layers[0].input],
 model.layers[1].get_output(train=False))
get_2nd_layer_output = theano.function([model.layers[0].input],
 model.layers[3].get_output(train=False))
get_3rd_layer_output = theano.function([model.layers[0].input],
 model.layers[8].get_output(train=False))
get_last_layer_output = theano.function([model.layers[0].input],
 model.layers[11].get_output(train=False))

print('X_test image shape:', X_test.shape)
layer_1_output = get_1st_layer_output(X_test)
print('Print 1st layer output', layer_1_output.shape)
layer_2_output = get_2nd_layer_output(X_test)
print('Print 2nd layer output', layer_2_output.shape)
layer_3_output = get_3rd_layer_output(X_test)
print('Print 3rd layer output', layer_3_output.shape)
layer_last_output = get_last_layer_output(X_test)
print('Print last layer output', layer_last_output.shape)
```



## 실습 (Cont'd)

### ④ 각 층의 결과 확인 하기

```
Predict classes and probability
```

```
print('Predict classes')
```

```
classes = model.predict_classes(X_test, batch_size)
```

```
print('Predict probability')
```

```
proba = model.predict_proba(X_test, batch_size)
```

```
Plot output of each layer
```

```
def plotvalue(index):
```

```
 plt.figure('Input data and 1~4 layer output value of
 X_test[{idx}]'.format(idx=index), figsize=(12,9), dpi=100)
```

```
 plt.subplot2grid((5,6),(0,0),rowspan=2,colspan=2)
```

```
 plt.title('Input data')
```

```
 plt.imshow(X_test[index][0], cmap='bone', interpolation='nearest')
```

```
 plt.subplot2grid((5,6),(0,2))
```

```
 plt.imshow(layer_1_output[index][0],
 cmap='bone', interpolation='nearest')
```

```
 plt.subplot2grid((5,6),(0,3))
```

```
 plt.imshow(layer_1_output[index][1],
 cmap='bone', interpolation='nearest')
```

## 실습 (Cont'd)

### ④ 각 층의 결과 확인 하기

```
plt.subplot2grid((5,6),(1,2))
plt.imshow(layer_1_output[index][2], cmap='bone',
 interpolation='nearest')
```

```
plt.subplot2grid((5,6),(0,4))
plt.imshow(layer_2_output[index][0],
 cmap='bone', interpolation='nearest')
```

```
plt.subplot2grid((5,6),(0,5))
plt.imshow(layer_2_output[index][1],
 cmap='bone', interpolation='nearest')
```

```
plt.subplot2grid((5,6),(1,4))
plt.imshow(layer_2_output[index][2],
 cmap='bone', interpolation='nearest')
```

```
plt.subplot2grid((5,6),(2,0),colspan=6)
plt.imshow(layer_3_output[index].reshape(1,layer_3_output.shape[1]),
 cmap='bone', interpolation='nearest')
```

## 실습 (Cont'd)

### ④ 각 층의 결과 확인 하기

```
plt.subplot2grid((5,6),(3,0),colspan=6)
plt.title('probability prediction')
plt.plot(proba[index], 'r', label='probability')
```

```
plt.subplot2grid((5,6),(4,0),colspan=6)
plt.title('Last layer output={ans},
 y_test={sol}'.format(ans=classes[index], sol=y_test[index]))
plt.imshow(layer_last_output[index].reshape(1,
 layer_last_output.shape[1])
 cmap='bone', interpolation='nearest')
```

```
plt.tight_layout()
plt.show()
```

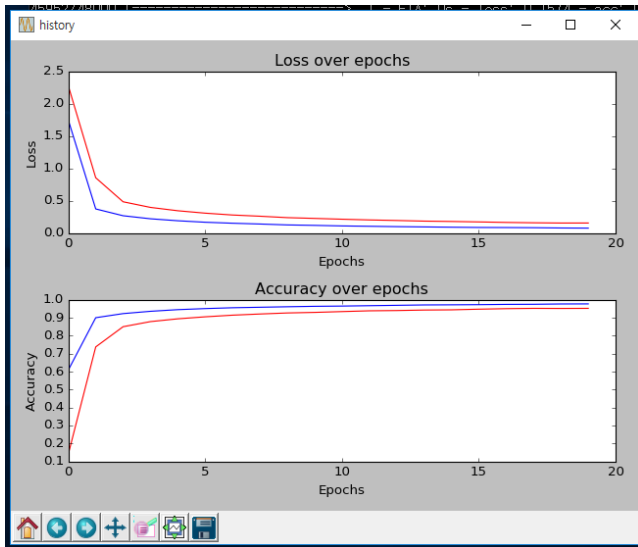
```
Plot test data
plotvalue(0)
plotvalue(1)
plotvalue(2)
```

## 실습 (Cont'd)

- `test_mnist_cnn.py`
    - ▶ WinPython Command Prompt내에서 다음과 같이 입력해 실행
- ```
> python 파일 경로\mnist_cnn_part2.py
```

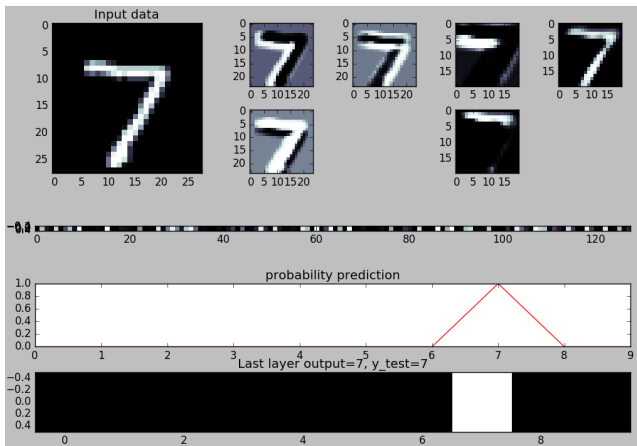
실습 (Cont'd)

- test_mnist_cnn.py



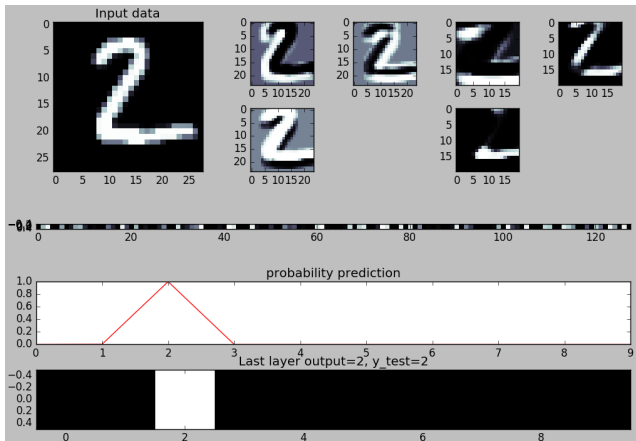
실습 (Cont'd)

- test_mnist_cnn.py



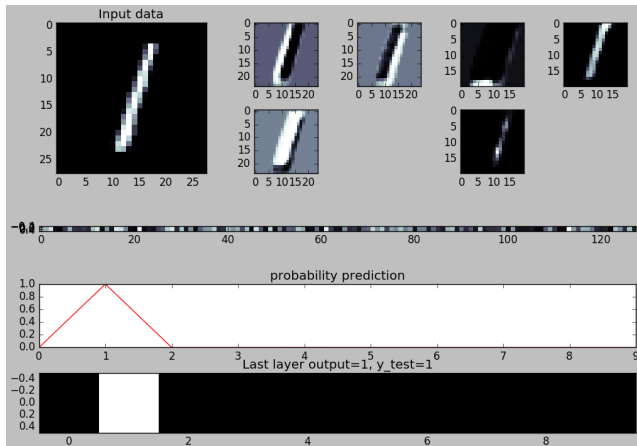
실습 (Cont'd)

- test_mnist_cnn.py



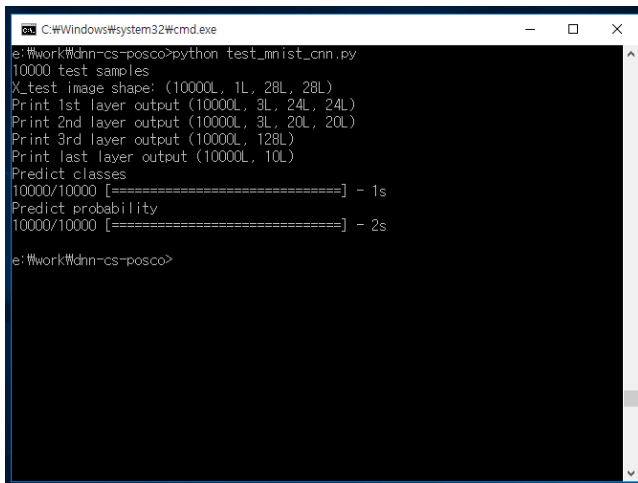
실습 (Cont'd)

- test_mnist_cnn.py



실습 (Cont'd)

- test_mnist_cnn.py



```
C:\Windows\system32\cmd.exe
e:\work\ddnn-cs-posco>python test_mnist_cnn.py
10000 test samples
X_test image shape: (10000L, 1L, 28L, 28L)
Print 1st layer output (10000L, 3L, 24L, 24L)
Print 2nd layer output (10000L, 3L, 20L, 20L)
Print 3rd layer output (10000L, 128L)
Print last layer output (10000L, 10L)
Predict classes
10000/10000 [=====] - 1s
Predict probability
10000/10000 [=====] - 2s

e:\work\ddnn-cs-posco>
```

Section 3

CIFAR-10 데이터를 위한 deep convolutional network

실습 목표

- The CIFAR-10 dataset
 - ▶ 10개의 class를 가지는 32×32 크기의 RGB(0~255) 컬러 이미지 데이터 베이스
 - ▶ 50000 개의 training 이미지, 10000 개의 test 이미지로 구성되어 있음
- 각 class 당 10개의 random 이미지

실습 목표 (Cont'd)

- **CIFAR-10 dataset** : [Sample_size, Channels, Rows, Cols]
 - ▶ Channels = RGB (각 0~255), Rows = 32, Cols=32
 - ▶ Sample_size : 50000개의 training set, 10000개의 test set
- ① CIFAR-10 데이터를 이용해 훈련한 네트워크의 model, weight, history 저장하기
- ② CIFAR-10 데이터를 이용해 훈련한 네트워크 불러오기
- ③ History 확인 하기
- ④ 각 층의 결과 확인 하기

실습 (Cont'd)

- `cifar10_cnn.py`

- ▶ 20개의 epoch, mini-batch size = 128
- ▶ IDLE에서 File-New File(Ctrl+N) 클릭 후, 새 파일에 다음을 입력하고 `cifar10_cnn.py`로 저장

```
from __future__ import print_function
import numpy as np
np.random.seed(1337) # for reproducibility
import json
```

```
from keras.datasets import cifar10
from keras.preprocessing.image import ImageDataGenerator
from keras.models import Sequential
from keras.layers.core import Dense, Dropout, Activation, Flatten
from keras.layers.convolutional import Convolution2D, MaxPooling2D
from keras.optimizers import SGD, Adadelta, Adagrad
from keras.utils import np_utils, generic_utils
```

실습 (Cont'd)

```
batch_size = 128
nb_classes = 10
nb_epoch = 80

# input image dimensions
img_rows, img_cols = 32, 32
# the CIFAR10 images are RGB
img_channels = 3
# number of convolutional filters to use
nb_filters = 5
# convolution kernel size
nb_conv = 5
# size of pooling area for max pooling
nb_pool = 2

# the data, shuffled and split between train and test sets
(X_train, y_train), (X_test, y_test) = cifar10.load_data()

X_train = X_train.astype('float32')
X_test = X_test.astype('float32')
```

실습 (Cont'd)

```
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')

# convert class vectors to binary class matrices
Y_train = np_utils.to_categorical(y_train, nb_classes)
Y_test = np_utils.to_categorical(y_test, nb_classes)

model = Sequential()

model.add(Convolution2D(nb_filters, nb_conv, nb_conv,
                        border_mode='same',
                        input_shape=(img_channels, img_rows, img_cols)))
model.add(Activation('sigmoid'))
model.add(Convolution2D(nb_filters, nb_conv, nb_conv))
model.add(Activation('sigmoid'))
model.add(MaxPooling2D(pool_size=(nb_pool, nb_pool)))
model.add(Dropout(0.25))
```

실습 (Cont'd)

```
model.add(Flatten())
model.add(Dense(128))
model.add(Activation('sigmoid'))
model.add(Dropout(0.5))
model.add(Dense(nb_classes))
model.add(Activation('softmax'))

model.compile(loss='categorical_crossentropy', optimizer='adadelta')

hist = model.fit(X_train, Y_train, batch_size=batch_size,
                 nb_epoch=nb_epoch, show_accuracy=True,
                 verbose=1, validation_split=0.2)
score = model.evaluate(X_test, Y_test, show_accuracy=True, verbose=0)
print('Test score:', score[0])
print('Test accuracy:', score[1])
```


실습 (Cont'd)

- ① CIFAR-10 데이터를 이용해 훈련한 네트워크의 model, weight, history 저장하기

```
# Save the model and weights
```

```
json_string = model.to_json()
```

```
open('cifar10_model_architecture.json', 'w').write(json_string)
```

```
model.save_weights('cifar10_model_weights.h5')
```

```
# Save History
```

```
with open('cifar10_model_history.json', 'w') as fp:
```

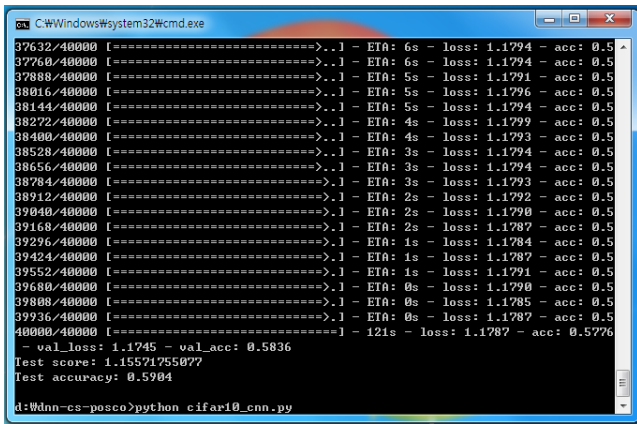
```
    json.dump(hist.history, fp)
```

실습 (Cont'd)

- `cifar10_cnn.py`
 - ▶ 80개의 epoch, mini-batch size = 128
 - ▶ WinPython Command Prompt내에서 다음과 같이 입력해 실행
- > python 파일 경로\cifar10_cnn.py

실습 (Cont'd)

- `cifar10_cnn.py`
 - ▶ 80개의 epoch, mini-batch size = 128



```
C:\Windows\system32\cmd.exe

37632/40000 [=====>...] - ETA: 6s - loss: 1.1794 - acc: 0.5
37760/40000 [=====>...] - ETA: 6s - loss: 1.1794 - acc: 0.5
37888/40000 [=====>...] - ETA: 5s - loss: 1.1791 - acc: 0.5
38016/40000 [=====>...] - ETA: 5s - loss: 1.1796 - acc: 0.5
38144/40000 [=====>...] - ETA: 5s - loss: 1.1794 - acc: 0.5
38272/40000 [=====>...] - ETA: 4s - loss: 1.1799 - acc: 0.5
38400/40000 [=====>...] - ETA: 4s - loss: 1.1793 - acc: 0.5
38528/40000 [=====>...] - ETA: 3s - loss: 1.1794 - acc: 0.5
38656/40000 [=====>...] - ETA: 3s - loss: 1.1794 - acc: 0.5
38784/40000 [=====>...] - ETA: 3s - loss: 1.1793 - acc: 0.5
38912/40000 [=====>...] - ETA: 2s - loss: 1.1792 - acc: 0.5
39040/40000 [=====>...] - ETA: 2s - loss: 1.1790 - acc: 0.5
39168/40000 [=====>...] - ETA: 2s - loss: 1.1787 - acc: 0.5
39296/40000 [=====>...] - ETA: 1s - loss: 1.1784 - acc: 0.5
39424/40000 [=====>...] - ETA: 1s - loss: 1.1787 - acc: 0.5
39552/40000 [=====>...] - ETA: 1s - loss: 1.1791 - acc: 0.5
39680/40000 [=====>...] - ETA: 0s - loss: 1.1790 - acc: 0.5
39808/40000 [=====>...] - ETA: 0s - loss: 1.1785 - acc: 0.5
39936/40000 [=====>...] - ETA: 0s - loss: 1.1787 - acc: 0.5
40000/40000 [=====] - 121s - loss: 1.1787 - acc: 0.5776
- val_loss: 1.1745 - val_acc: 0.5836
Test score: 1.15571755077
Test accuracy: 0.5904

d:\dnn-cs-posco>python cifar10_cnn.py
```

실습 (Cont'd)

- test_cifar10_cnn.py

```
from __future__ import print_function

from keras.datasets import cifar10
from keras.models import Sequential

from keras.models import model_from_json
import matplotlib.pyplot as plt
import theano
import json

batch_size = 128
nb_classes = 10
nb_epoch = 20

# input image dimensions
img_rows, img_cols = 32, 32
# the CIFAR10 images are RGB
img_channels = 3
```

실습 (Cont'd)

```
# the data, shuffled and split between tran and test sets
(X_train, y_train), (X_test, y_test) = cifar10.load_data()
X_test = X_test.astype('float32')
X_test /= 255
print(X_test.shape[0], 'test samples')
```

② CIFAR-10 데이터를 이용해 훈련한 네트워크 불러오기

```
# Reconstruct model
model = model_from_json(open('cifar10_model_architecture.json').read())
model.load_weights('cifar10_model_weights.h5')
```

실습 (Cont'd)

③ History 확인 하기

```
# Plot history
```

```
hist = json.loads(open('cifar10_model_history.json').read())
```

```
plt.figure('history')
plt.subplot(211)
plt.title('Loss over epochs')
plt.plot(hist['loss'], 'r', label='loss')
plt.plot(hist['val_loss'], 'b', label='val_loss')
plt.xlabel('Epochs')
plt.ylabel('Loss')
```

```
plt.subplot(212)
plt.title('Accuracy over epochs')
plt.plot(hist['acc'], 'r', label='acc')
plt.plot(hist['val_acc'], 'b', label='val_acc')
plt.xlabel('Epochs')
plt.ylabel('Accuracy')
```

```
plt.tight_layout()
plt.show()
```

실습 (Cont'd)

④ 각 층의 결과 확인 하기

Get output of each layer

```
get_1st_layer_output = theano.function([model.layers[0].input],
                                         model.layers[1].get_output(train=False))
get_2nd_layer_output = theano.function([model.layers[0].input],
                                         model.layers[3].get_output(train=False))
get_3rd_layer_output = theano.function([model.layers[0].input],
                                         model.layers[8].get_output(train=False))
get_last_layer_output = theano.function([model.layers[0].input],
                                         model.layers[11].get_output(train=False))

print('X_test image shape:', X_test.shape)
layer_1_output = get_1st_layer_output(X_test)
print('Print 1st layer output', layer_1_output.shape)
layer_2_output = get_2nd_layer_output(X_test)
print('Print 2nd layer output', layer_2_output.shape)
layer_3_output = get_3rd_layer_output(X_test)
print('Print 3rd layer output', layer_3_output.shape)
layer_last_output = get_last_layer_output(X_test)
print('Print last layer output', layer_last_output.shape)
```

실습 (Cont'd)

④ 각 층의 결과 확인 하기

```
# Predict classes and probability
```

```
print('Predict classes')
```

```
classes = model.predict_classes(X_test, batch_size)
```

```
print('Predict probability')
```

```
proba = model.predict_proba(X_test, batch_size)
```

```
# Plot output of each layer
```

```
def plotvalue(index):
```

```
    plt.figure('Input data and 1~4 layer output value of
```

```
                X_test[{idx}]'.format(idx=index), figsize=(12,9), dpi=100)
```

```
    plt.subplot2grid((5,8),(0,0),rowspan=2,colspan=2)
```

```
    plt.title('Input data')
```

```
    plt.imshow(X_test[index].transpose(1,2,0))
```

```
    plt.subplot2grid((5,8),(0,2))
```

```
    plt.imshow(layer_1_output[index][0],
```

```
               cmap='bone', interpolation='nearest')
```

```
    plt.subplot2grid((5,8),(0,3))
```

```
    plt.imshow(layer_1_output[index][1],
```

```
               cmap='bone', interpolation='nearest')
```


실습 (Cont'd)

④ 각 층의 결과 확인 하기

```
plt.subplot2grid((5,8),(0,4))
plt.imshow(layer_1_output[index][2],
           cmap='bone', interpolation='nearest')
plt.subplot2grid((5,8),(1,2))
plt.imshow(layer_1_output[index][3],
           cmap='bone', interpolation='nearest')
plt.subplot2grid((5,8),(1,3))
plt.imshow(layer_1_output[index][4],
           cmap='bone', interpolation='nearest')
plt.subplot2grid((5,8),(0,5))

plt.imshow(layer_2_output[index][0],
           cmap='bone', interpolation='nearest')
plt.subplot2grid((5,8),(0,6))
plt.imshow(layer_2_output[index][1],
           cmap='bone', interpolation='nearest')
```

실습 (Cont'd)

④ 각 층의 결과 확인 하기

```
plt.subplot2grid((5,8),(0,7))
plt.imshow(layer_2_output[index][2],
           cmap='bone', interpolation='nearest')
plt.subplot2grid((5,8),(1,5))
plt.imshow(layer_2_output[index][3],
           cmap='bone', interpolation='nearest')
plt.subplot2grid((5,8),(1,6))
plt.imshow(layer_2_output[index][4],
           cmap='bone', interpolation='nearest')

plt.subplot2grid((5,8),(2,0),colspan=8)
plt.imshow(layer_3_output[index].reshape(1,layer_3_output.shape[1]),
           cmap='bone', interpolation='nearest')
```

실습 (Cont'd)

4 각 층의 결과 확인 하기

```
plt.subplot2grid((5,8),(3,0),colspan=8)
plt.title('probability prediction')
plt.plot(proba[index], 'r', label='probability')
```

```
plt.subplot2grid((5,8),(4,0),colspan=8)
plt.title('Last layer output={ans},
          y_test={sol}'.format(ans=classes[index], sol=y_test[index]))
plt.imshow(layer_last_output[index].reshape(1,
                                              layer_last_output.shape[1])
           cmap='bone', interpolation='nearest')
```

```
plt.tight_layout()
plt.show()
```

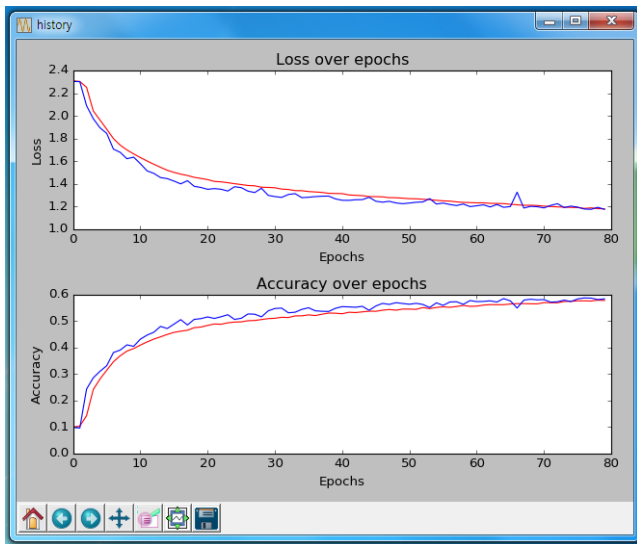
```
# Plot test data
plotvalue(0)
plotvalue(1)
plotvalue(2)
```

실습 (Cont'd)

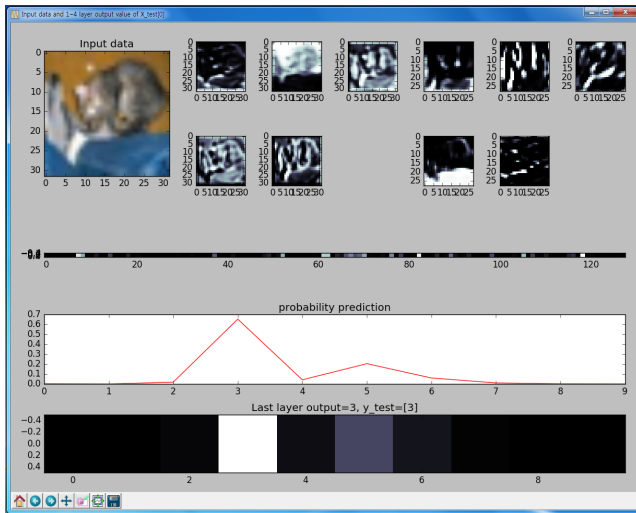
- `test_cifar10_cnn.py`
 - ▶ WinPython Command Prompt내에서 다음과 같이 입력해 실행
- > `python 파일 경로\test_cifar10_cnn.py`

실습 (Cont'd)

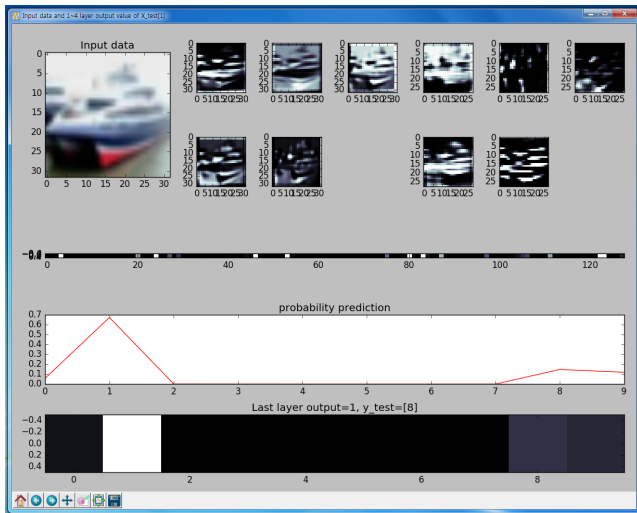
- test_cifar10_cnn.py



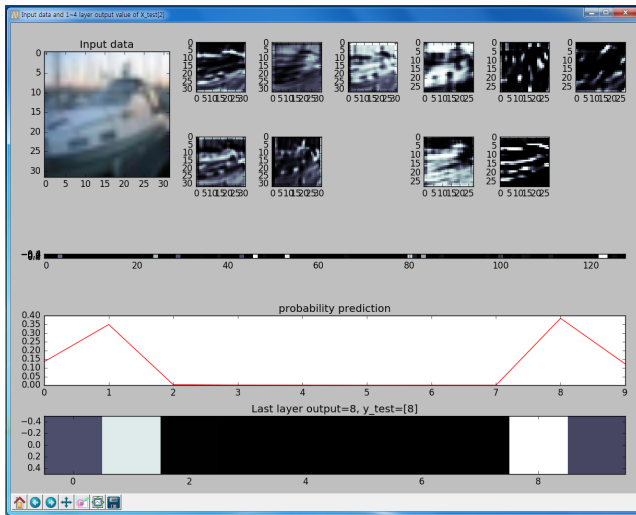
실습 (Cont'd)



실습 (Cont'd)



실습 (Cont'd)



실습 (Cont'd)

```
C:\Windows\system32\cmd.exe

39168/40000 [=====>.] - ETA: 2s - loss: 1.1787 - acc: 0.5
39296/40000 [=====>.] - ETA: 1s - loss: 1.1784 - acc: 0.5
39424/40000 [=====>.] - ETA: 1s - loss: 1.1787 - acc: 0.5
39552/40000 [=====>.] - ETA: 1s - loss: 1.1791 - acc: 0.5
39680/40000 [=====>.] - ETA: 0s - loss: 1.1790 - acc: 0.5
39808/40000 [=====>.] - ETA: 0s - loss: 1.1785 - acc: 0.5
39936/40000 [=====>.] - ETA: 0s - loss: 1.1787 - acc: 0.5
40000/40000 [=====] - 121s - loss: 1.1787 - acc: 0.5776
- val_loss: 1.1745 - val_acc: 0.5836
Test score: 1.15571755077
Test accuracy: 0.5904

d:\w4nn-cs-posco>python test_cifar10_cnn.py
10000 test samples
X_test image shape: (10000L, 3L, 32L, 32L)
Print 1st layer output (10000L, 5L, 32L, 32L)
Print 2nd layer output (10000L, 5L, 28L, 28L)
Print 3rd layer output (10000L, 128L)
Print last layer output (10000L, 10L)
Predict classes
10000/10000 [=====] - 14s
Predict probability
10000/10000 [=====] - 13s

d:\w4nn-cs-posco>
```