12주. Keras DNN

학번	32153180	이름	이상민

```
# libraries
from keras.datasets import mnist
from keras import optimizers
from keras.models import Sequential
from keras.layers import Dense
from keras.layers import Flatten
from keras.layers import Dropout
from keras.utils import np_utils
from sklearn.preprocessing import LabelEncoder
from sklearn.model_selection import train_test_split

import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
```

Q1 (7점) 제공된 PimaIndiansDiabetes.csv 파일에 대해 Keras를 이용한 classification 모델을 개발하고 테스트 하시오

- train/test set을 나누되 test set 은 전체 dataset 의 30% 로 한다.
- hidden layer 의 수는 3~4개, layer별 노드수는 각자 정한다.
- hidden layer 의 활성화 함수는 relu, output layer 의 노드수는 softmax 로 한다
- 기타 필요한 매개변수들은 각자 정한다.
- epoch 는 20,40,60,80, 100 으로 변화시켜 가면서 테스트한다.

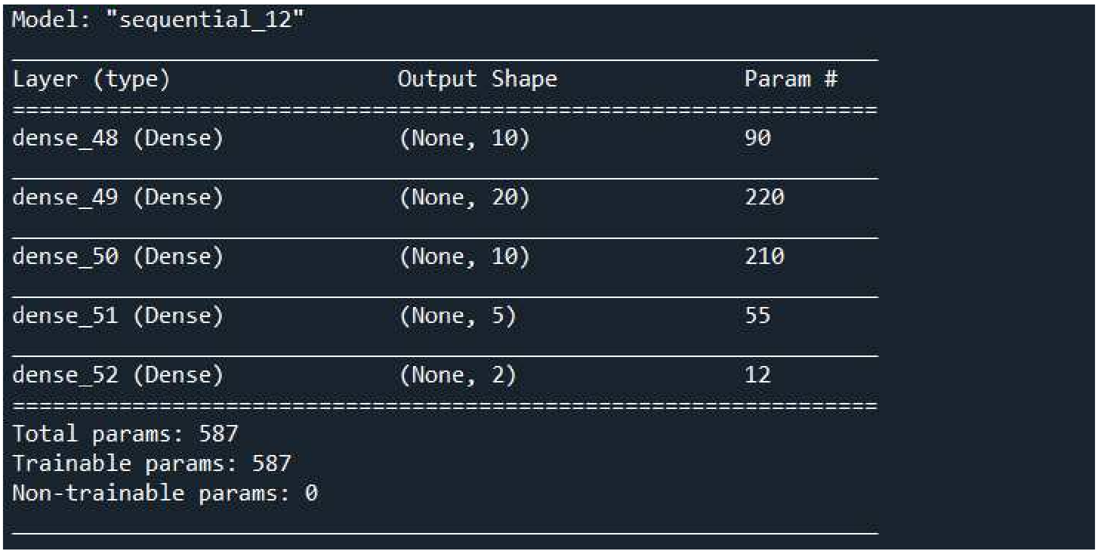
* 각 epoch별로 training accuracy 와 test accuracy를 제시한다
(slide 18과 같은 그래프를 함께 제시)

Source code :

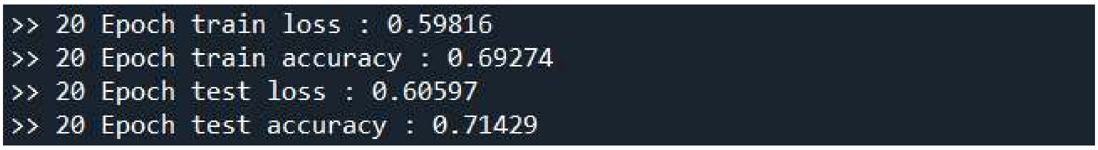
```
pima = pd.read_csv('C:/Users/sangmin/Desktop/학교생활/4-2/딥러닝클라우드
/dataset/PimaIndiansDiabetes.csv')
dataset = pima.values
X = dataset[:, 0:8]
Y = dataset[:, 8]

encoder = LabelEncoder()
encoder.fit(Y)
encoded_Y = encoder.transform(Y)
onehot_y = np_utils.to_categorical(encoded_Y)

train_X, test_X, train_y, test_y = \
   train_test_split(X, onehot_y, test_size=0.3, random_state=100)

# define model
epochs = [20, 40, 60, 80, 100]
batch_size = 10

model = Sequential()
model.add(Dense(10, input_dim=8, activation='relu'))
model.add(Dense(20, activation='relu'))
model.add(Dense(10, activation='relu'))
model.add(Dense(5, activation='relu'))
model.add(Dense(2, activation='softmax'))

model.summary()

# compile model
model.compile(loss='categorical_crossentropy',
             optimizer='adam',
             metrics=['accuracy'])
```

```
# model fitting & test
for epoch in epochs:
   disp = model.fit(train_X, train_y,
                    batch_size=batch_size,
                    epochs=epoch,
                    verbose=1,
                    validation_data=(test_X, test_y))

   pred = model.predict(test_X)
   y_classes = [np.argmax(y, axis=None, out=None) for y in pred]

   train_score = model.evaluate(train_X, train_y, verbose=0)
   test_score = model.evaluate(test_X, test_y, verbose=0)
   print('>> {} Epoch train loss : {}'.format(epoch,
round(train_score[0], 5)))
   print('>> {} Epoch train accuracy : {}'.format(epoch,
round(train_score[1], 5)))
   print('>> {} Epoch test loss : {}'.format(epoch,
round(test_score[0], 5)))
   print('>> {} Epoch test accuracy : {}'.format(epoch,
round(test_score[1], 5)))
   print('\n')

   plt.plot(disp.history['accuracy'])
   plt.plot(disp.history['val_accuracy'])
   plt.title('{} Epoch model accuracy'.format(epoch))
   plt.xlabel('epoch')
   plt.ylabel('accuracy')
   plt.legend(['train', 'test'], loc='upper left')
   plt.show()
```

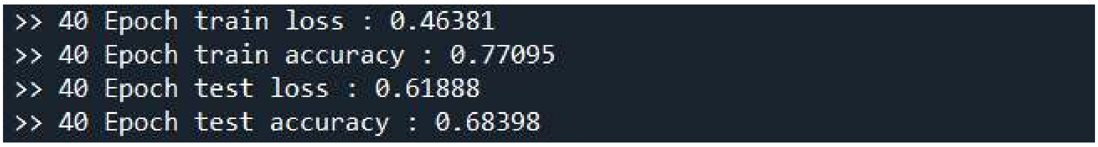
실행화면 캡처:



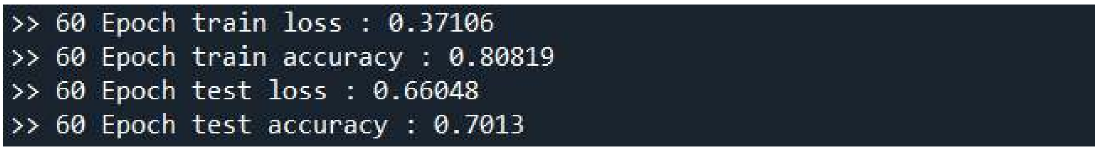
```
Model: "sequential_12"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_48 (Dense)             (None, 10)                90
_________________________________________________________________
dense_49 (Dense)             (None, 20)                220
_________________________________________________________________
dense_50 (Dense)             (None, 10)                210
_________________________________________________________________
dense_51 (Dense)             (None, 5)                 55
_________________________________________________________________
dense_52 (Dense)             (None, 2)                 12
=================================================================
Total params: 587
Trainable params: 587
Non-trainable params: 0
_________________________________________________________________
```



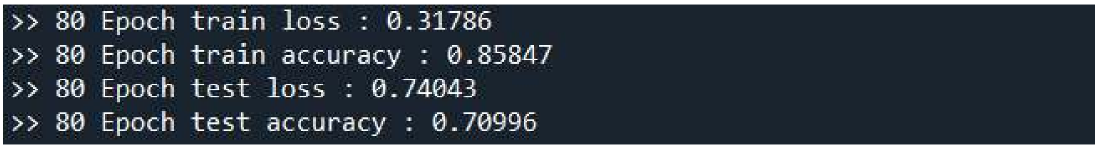
```
>> 20 Epoch train loss : 0.59816
>> 20 Epoch train accuracy : 0.69274
>> 20 Epoch test loss : 0.60597
>> 20 Epoch test accuracy : 0.71429
```



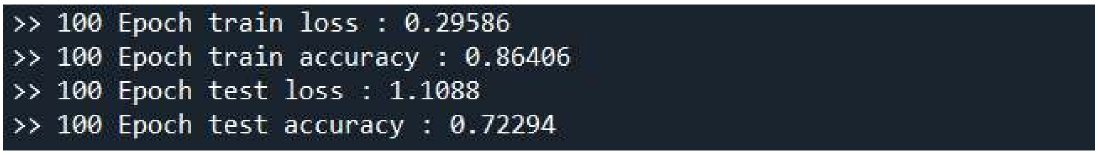
```
>> 40 Epoch train loss : 0.46381
>> 40 Epoch train accuracy : 0.77095
>> 40 Epoch test loss : 0.61888
>> 40 Epoch test accuracy : 0.68398
```



```
>> 60 Epoch train loss : 0.37106
>> 60 Epoch train accuracy : 0.80819
>> 60 Epoch test loss : 0.66048
>> 60 Epoch test accuracy : 0.7013
```

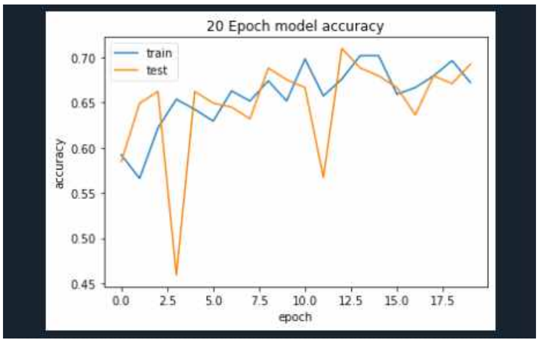


```
>> 80 Epoch train loss : 0.31786
>> 80 Epoch train accuracy : 0.85847
>> 80 Epoch test loss : 0.74043
>> 80 Epoch test accuracy : 0.70996
```

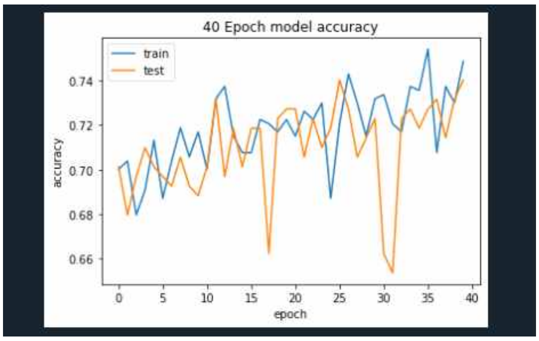


```
>> 100 Epoch train loss : 0.29586
>> 100 Epoch train accuracy : 0.86406
>> 100 Epoch test loss : 1.1088
>> 100 Epoch test accuracy : 0.72294
```

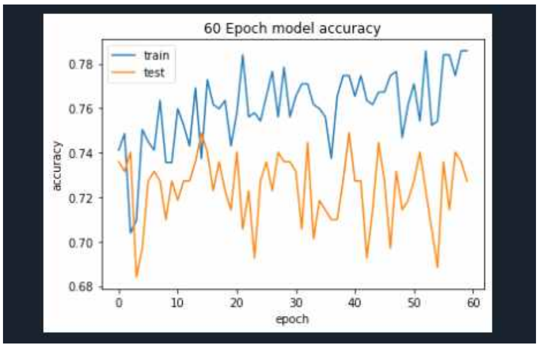
Epoch 20



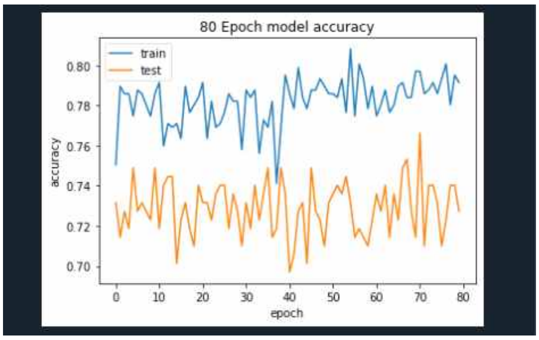
Epoch 40



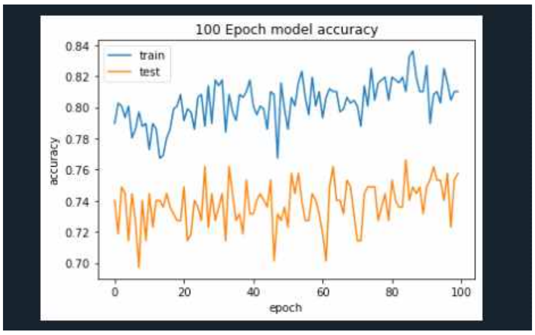
Epoch 60



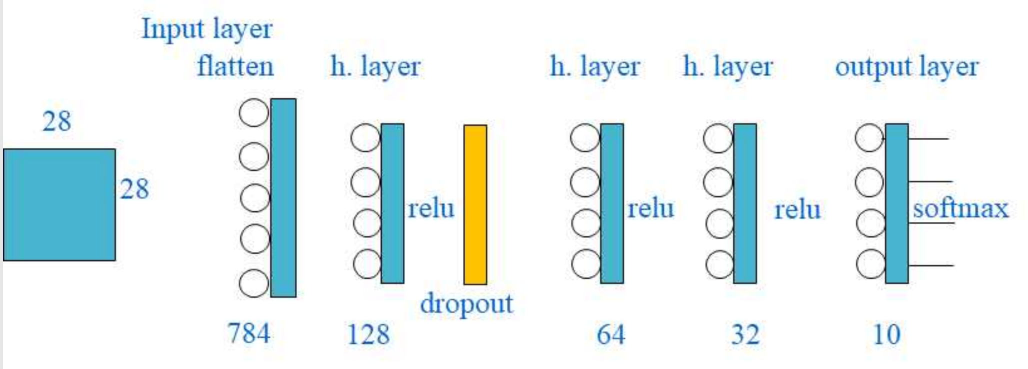
Epoch 80



Epoch 100



Q2 (3점) chap12_keras_dnn_2.pdf 파일의 source code를 다음과 같은 DNN architecture 로 수정하여 테스트 하시오

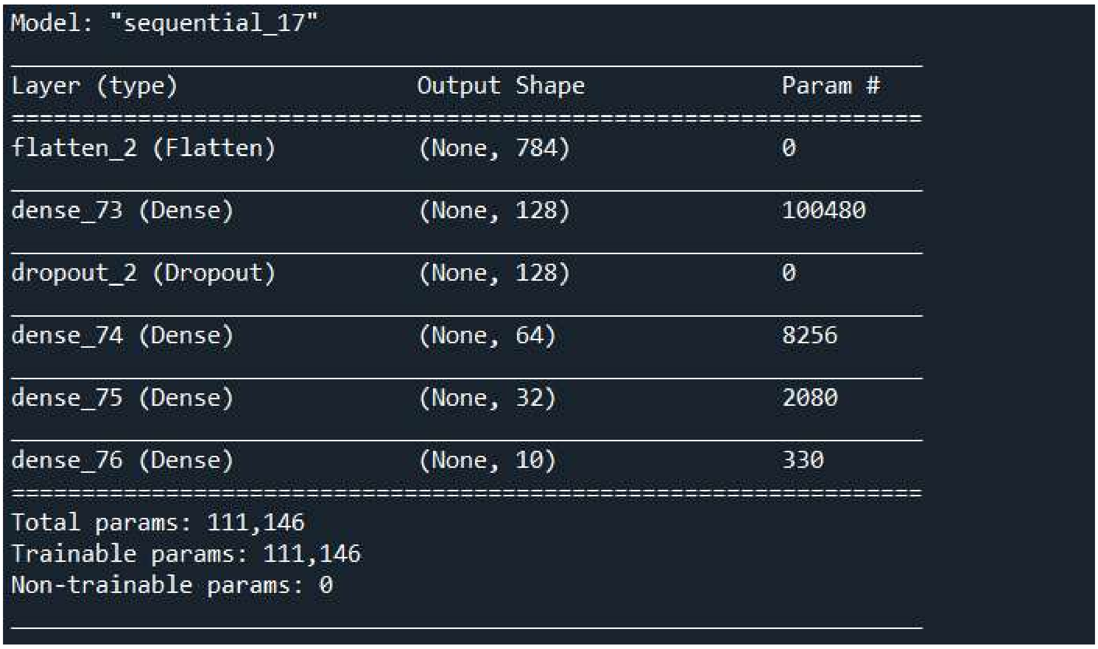


Source code :

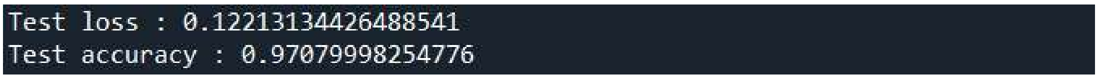
```
(train_X, train_y), (test_X, test_y) = mnist.load_data()
train_X, test_X = train_X / 255.0, test_X / 255.0

train_y = np_utils.to_categorical(train_y)
test_y = np_utils.to_categorical(test_y)

# define model
epochs = 20
batch_size = 128
learning_rate = 0.01

model = Sequential()
model.add(Flatten(input_shape=(28, 28)))
model.add(Dense(128, activation='relu'))
model.add(Dropout(rate=0.4))
model.add(Dense(64, activation='relu'))
model.add(Dense(32, activation='relu'))
model.add(Dense(10, activation='softmax'))

model.summary()

# compile model
adam = optimizers.adam(lr=learning_rate)
model.compile(loss='categorical_crossentropy',
            optimizer=adam, metrics=['accuracy'])
```

```
# model filtting & test
disp = model.fit(train_X, train_y,
              batch_size=batch_size,
              epochs=epochs,
              verbose=1,
              validation_split=0.2)

pred = model.predict(test_X)
y_classes = [np.argmax(y, axis=None, out=None) for y in pred]

score = model.evaluate(test_X, test_y, verbose=0)
print('Test loss : {}'.format(score[0]))
print('Test accuracy : {}'.format(score[1]))

plt.plot(disp.history['accuracy'])
plt.plot(disp.history['val_accuracy'])
plt.title('model accuracy')
plt.xlabel('epoch')
plt.ylabel('model accuracy')
plt.legend(['train', 'validation'], loc='upper left')
plt.show()

plt.plot(disp.history['loss'])
plt.plot(disp.history['val_loss'])
plt.title('model loss')
plt.xlabel('epoch')
plt.ylabel('model loss')
plt.legend(['train', 'validation'], loc='upper left')
plt.show()
```

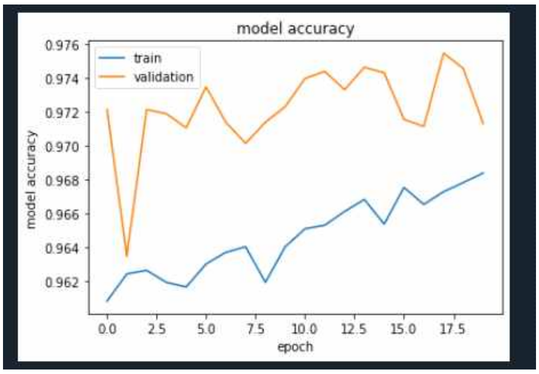
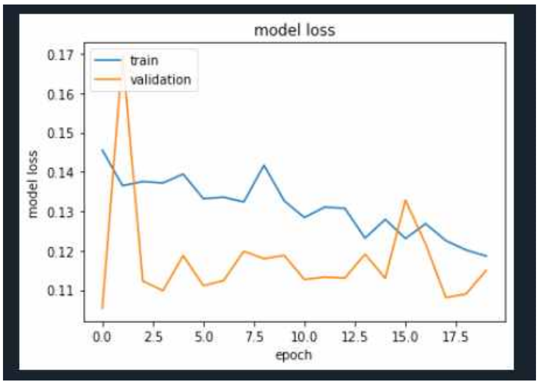
실행화면 캡처:



```
Model: "sequential_17"
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
flatten_2 (Flatten)          (None, 784)               0         
_________________________________________________________________
dense_73 (Dense)             (None, 128)               100480    
_________________________________________________________________
dropout_2 (Dropout)          (None, 128)               0         
_________________________________________________________________
dense_74 (Dense)             (None, 64)                8256      
_________________________________________________________________
dense_75 (Dense)             (None, 32)                2080      
_________________________________________________________________
dense_76 (Dense)             (None, 10)                330       
=================================================================
Total params: 111,146
Trainable params: 111,146
Non-trainable params: 0
_________________________________________________________________
```



```
Test loss : 0.12213134426488541
Test accuracy : 0.97079998254776
```

model accuracy	model loss
	

model accuracy
train
validation
model accuracy
epoch

model loss
train
validation
model loss
epoch