12주. Keras DNN			
학번	32152339	이름	송준영

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 노드 수는 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 :

```
// source code 의 폰트는 Courier10 BT Bold으로 하시오
import os
os.getcwd()
os.chdir('C:\\Users\\ATIV\\Desktop\\Deeplearning_Cloud')
import TrainPlot
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import keras
from keras.models import Sequential
from keras.layers import Dense,Flatten,Dropout
from keras.utils import np_utils
from sklearn.preprocessing import StandardScaler,LabelEncoder
from sklearn.model_selection import train_test_split
import warnings
warnings.filterwarnings(action='ignore')


pd.set_option("max_columns",10)
df=pd.read_csv("dataset/PimaIndiansDiabetes.csv")

#scaling
scaler=StandardScaler()
scaler.fit(df.drop("diabetes",axis=1))
x=scaler.transform(df.drop("diabetes",axis=1))
```

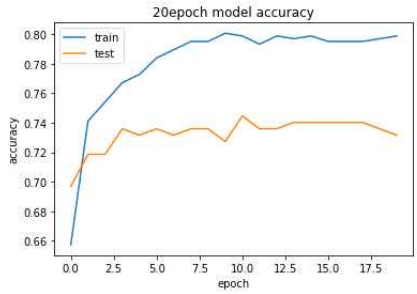
```
#label encoding
y=df.diabetes
encoder=LabelEncoder()
encoder.fit(y)
encoded_y=encoder.transform(y)

#convert integer to dummy variables
dummy_y=np_utils.to_categorical(encoded_y)

#train, test split
train_x,test_x,train_y,test_y=train_test_split(x,dummy_y,test_size=
0.3,random_state=32152339)

#define model
batch_size=10

model=Sequential()
model.add(Dense(10,input_dim=train_x.shape[1],activation='relu'))
model.add(Dense(10,activation='relu'))
model.add(Dense(10,activation='relu'))
model.add(Dense(2,activation='softmax'))

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

#model fitting (learning)
epoch=[20,40,60,80,100]
for i in epoch:
    dnn_clf=model.fit(train_x,train_y,
                  batch_size=batch_size,
                  epochs=i,
                  verbose=0,
                  validation_data=(test_x,test_y))
    trn_score=model.evaluate(train_x,train_y,verbose=0)
    tst_score=model.evaluate(test_x,test_y,verbose=0)
    print(f"{i}epoch train accuracy",trn_score[1])
```

```
    print(f"{i}epoch test accuracy",tst_score[1])
    plt.plot(dnn_clf.history['acc'])
    plt.plot(dnn_clf.history['val_acc'])
    plt.title(f'{i}epoch model accuracy')
    plt.ylabel('accuracy')
    plt.xlabel('epoch')
    plt.legend(['train','test'],loc='upper left')
    plt.show()
```

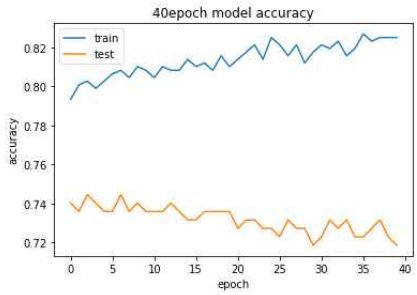
실행화면 캡쳐:

```
In [29]: df=pd.read_csv("dataset/PimaIndiansDiabetes.csv")
    ...:
    ...: #scaling
    ...: scaler=StandardScaler()
    ...: scaler.fit(df.drop("diabetes",axis=1))
    ...: x=scaler.transform(df.drop("diabetes",axis=1))
    ...:
    ...: #label encoding
    ...: y=df.diabetes
    ...: encoder=LabelEncoder()
    ...: encoder.fit(y)
    ...: encoded_y=encoder.transform(y)
    ...:
    ...: #convert integer to dummy variables
    ...: dummy_y=np_utils.to_categorical(encoded_y)
    ...:
    ...: #train, test split
    ...: train_x,test_x,train_y,test_y=\
    ...: train_test_split(x,dummy_y,test_size=0.3,random_state=32152339)
    ...:
    ...: #define model
    ...: batch_size=10
    ...:
    ...: model=Sequential()
    ...: model.add(Dense(10,input_dim=train_x.shape[1],activation='relu'))
    ...: model.add(Dense(10,activation='relu'))
    ...: model.add(Dense(10,activation='relu'))
    ...: model.add(Dense(2,activation='softmax'))
    ...:
    ...: #compile model
    ...: model.compile(optimizer='adam',
    ...:               loss='categorical_crossentropy',
    ...:               metrics=['accuracy'])
```

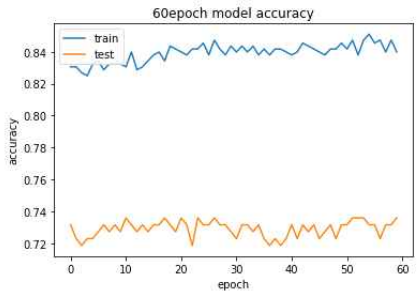
```
In [30]: epoch=[20,40,60,80,100]
    ...: for i in epoch:
    ...:     dnn_clf=model.fit(train_x,train_y,
    ...:                       batch_size=batch_size,
    ...:                       epochs=i,
    ...:                       verbose=0,
    ...:                       validation_data=(test_x,test_y))
    ...:     trn_score=model.evaluate(train_x,train_y,verbose=0)
    ...:     tst_score=model.evaluate(test_x,test_y,verbose=0)
    ...:     print('-----------',f'epoch : {i}','-----------')
    ...:     print(f"{i}epoch train accuracy",trn_score[1])
    ...:     print(f"{i}epoch test accuracy",tst_score[1])
    ...:     plt.plot(dnn_clf.history['acc'])
    ...:     plt.plot(dnn_clf.history['val_acc'])
    ...:     plt.title(f'{i}epoch model accuracy')
    ...:     plt.ylabel('accuracy')
    ...:     plt.xlabel('epoch')
    ...:     plt.legend(['train','test'],loc='upper left')
    ...:     plt.show()
    ...:     print()
----------- epoch : 20 -----------
20epoch train accuracy 0.8026070769050712
20epoch test accuracy 0.7316017323758179
```



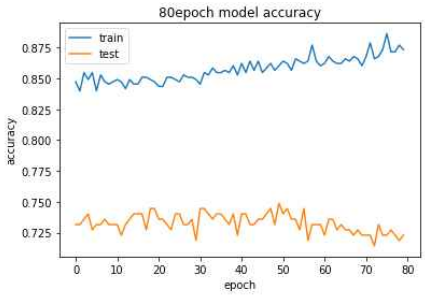

```
----------- epoch : 40 -----------
40epoch train accuracy 0.8305400377989259
40epoch test accuracy 0.718614719388805
```



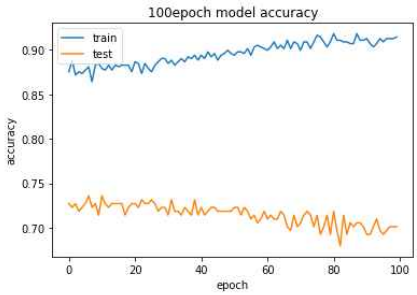

```
----------- epoch : 60 -----------
60epoch train accuracy 0.8528864047380799
60epoch test accuracy 0.7359307367048222
```



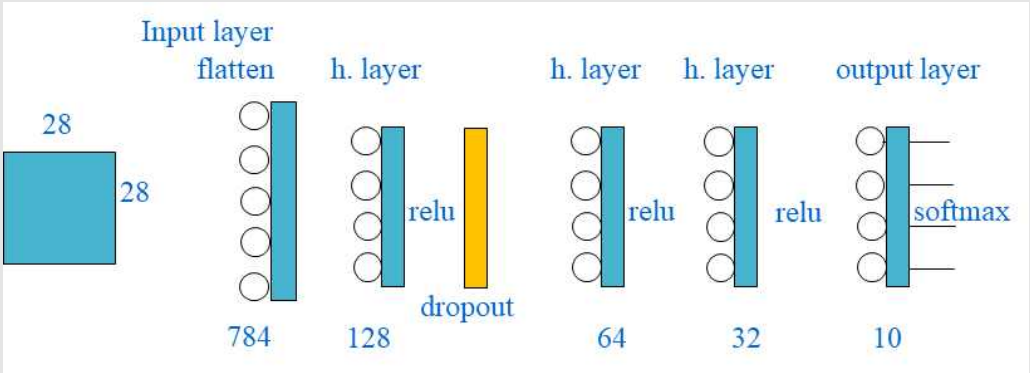
```
---------- epoch : 80 ----------
80epoch train accuracy 0.8882681562025898
80epoch test accuracy 0.7229437237178092
```




```
---------- epoch : 100 ----------
100epoch train accuracy 0.9199255118822919
100epoch test accuracy 0.7012987020727875
```



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



Source code :

```
// source code 의 폰트는 Courier10 BT Bold으로 하시오
#load data
from keras.datasets import mnist
(train_x,train_y),(test_x,test_y)=mnist.load_data()
train_x,test_x=train_x/255.0,test_x/255.0

#one hot encoding
```

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

#define model
epoch=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'))

#compile model
from keras import optimizers
adam = optimizers.adam(lr=learning_rate)
model.compile(loss='categorical_crossentropy',
           optimizer=adam,
           metrics=['accuracy'])
#model fitting
disp = model.fit(train_x,train_y,
              batch_size=batch_size,
              epochs=epoch,
              verbose=1,
              validation_split=0.2)
#model performance
score = model.evaluate(test_x,test_y,verbose=0)
print('test loss:',score[0])
print('test accuracy:',score[1])
#summarize history for accuracy
plt.plot(disp.history['acc'])
plt.plot(disp.history['val_acc'])
plt.title('model accuracy')
plt.ylabel('accuracy')
plt.xlabel('epoch')
plt.legend(['train','test'],loc='upper left')
plt.show()
```

실행화면 캡쳐:

```
In [52]: from keras.datasets import mnist
    ...: (train_x,train_y),(test_x,test_y)=mnist.load_data()
    ...: train_x,test_x=train_x/255.0,test_x/255.0
    ...:
    ...: #one hot encoding
    ...: train_y=np_utils.to_categorical(train_y)
    ...: test_y=np_utils.to_categorical(test_y)
    ...:
    ...: #define model
    ...: epoch=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'))
    ...:
    ...: #compile model
    ...: from keras import optimizers
    ...: adam = optimizers.adam(lr=learning_rate)
    ...: model.compile(loss='categorical_crossentropy',
    ...:               optimizer=adam,
    ...:               metrics=['accuracy'])
    ...:
    ...: #model fitting
    ...: disp = model.fit(train_x,train_y,
    ...:                  batch_size=batch_size,
    ...:                  epochs=epoch,
    ...:                  verbose=1,
    ...:                  validation_split=0.2)
Train on 48000 samples, validate on 12000 samples
Epoch 1/20
48000/48000 [==============================] - 4s 92us/step - loss: 0.4205 -
acc: 0.8728 - val_loss: 0.1786 - val_acc: 0.9467
Epoch 2/20
48000/48000 [==============================] - 4s 75us/step - loss: 0.2577 -
acc: 0.9234 - val_loss: 0.1478 - val_acc: 0.9577
Epoch 17/20
48000/48000 [==============================] - 4s 90us/step - loss: 0.1400 -
acc: 0.9616 - val_loss: 0.1173 - val_acc: 0.9705
Epoch 18/20
48000/48000 [==============================] - 4s 90us/step - loss: 0.1418 -
acc: 0.9609 - val_loss: 0.1070 - val_acc: 0.9729
Epoch 19/20
48000/48000 [==============================] - 4s 91us/step - loss: 0.1391 -
acc: 0.9616 - val_loss: 0.1154 - val_acc: 0.9717
Epoch 20/20
48000/48000 [==============================] - 4s 88us/step - loss: 0.1444 -
acc: 0.9604 - val_loss: 0.1205 - val_acc: 0.9703
```

```
In [53]: score = model.evaluate(test_x,test_y,verbose=0)
    ...: print('test loss:',score[0])
    ...: print('test accuracy:',score[1])
    ...:
    ...: #summarize history for accuracy
    ...: plt.plot(disp.history['acc'])
    ...: plt.plot(disp.history['val_acc'])
    ...: plt.title('model accuracy')
    ...: plt.ylabel('accuracy')
    ...: plt.xlabel('epoch')
    ...: plt.legend(['train','test'],loc='upper left')
    ...: plt.show()
test loss: 0.11119928864911198
test accuracy: 0.9725
```