```
#*******************************************************
#    mnist DB 이용한 keras 동작 확인 프로그램 소스
#*******************************************************

#-----------------------------------------
#    [1] ANACONDA, Jupyter Notebook, 필요 패키지 설치
#-----------------------------------------
1. anaconda 설치 (https://repo.continuum.io/archive/)
        PATH설정 체크박스 체크
        설치후에 Command Prompt에서 conda —version 실행해서 설치 확인
                                        # 윈도우즈 시작메뉴-Anaconda Prompt 실행후
        conda update conda 로 conda 최신버전으로 업데이트
        conda —version
        (※ 특정 가상환경 제거 : conda env remove -n 가상환경명)
        conda info —envs 로 가상환경 목록 확인
2. 프로젝트 디렉토리+가상환경 mnist 만들기 (관리자 권한으로 Command Prompt 실행한 후)
        d:
        d:₩>md mnist
        d:₩>cd mnist
                                        # 'mnist' 가상환경 생성 및 필요 패키지 동시 설치
        d:₩mnist> conda create -n mnist git nodejs tensorflow-gpu python=3.6 pydotplus pydot
                        libpython m2w64-toolchain python-graphviz opencv keras anaconda
        d:₩mnist> conda activate mnist

3. 시작메뉴에서 '주피터 노트북' 실행하여 코드 입력
        cd ..
        d:₩mnist> jupyter notebook

4. 주피터 노트북에서 우측 상단에서 "new->Python 3" 실행하여
        아래 코드 입력 후 Shift + Enter로 실행


#-----------------------------------------
#  [2] mnist DB 이용한 keras 동작 확인 프로그램 소스
#-----------------------------------------
import scipy
import numpy
import matplotlib
import pandas
import sklearn
import pydotplus
import h5py
import tensorflow
import keras
import graphviz

from keras.utils import np_utils
from keras.datasets import mnist
from keras.models import Sequential
from keras.layers import Dense, Activation


#============================================================
# 데이터 설정
#============================================================
# 필기숫자 데이터 불러오기
(X_train, Y_train), (X_test, Y_test) = mnist.load_data()

# 6만개 학습용 입력 데이터 포맷을 float32 1이하 숫자로 reshape
X_train = X_train.reshape(60000, 784).astype('float32') / 255.0
# 1만개 테스트용 입력 데이터 포맷을 float32 1이하 숫자로 reshape
X_test = X_test.reshape(10000, 784).astype('float32') / 255.0
```

```
# Target 값 설정
Y_train = np_utils.to_categorical(Y_train)
Y_test = np_utils.to_categorical(Y_test)

#=============================================================
# 학습 모델 설정
#=============================================================
    model = Sequential([
            Dense(32, input_shape=(784,)),
            Activation('relu'),
            Dense(10),
            Activation('softmax'),
        ])

#=============================================================================================
# 최적화방법 설정 (참고:https://keras.io/optimizers/)
#=============================================================================================
# (1) SGD(Stochastic Gradient Descent) optimizer
# momentum, learning rate decay, and Nesterov momentum
# lr: float >= 0. Learning rate
# momentum: float >= 0.  Accelerates SGD in the relevant direction and dampens oscillations
# decay: float >= 0. Learning rate decay over each update
# nesterov: boolean. Whether to apply Nesterov momentum
# sgd = optimizers.SGD(lr=0.01, decay=1e-6, momentum=0.9, nesterov=True)

# (2) RMSProp optimizer
# Recurrent Neural Network에 주로 사용하고 lr 외의 변수들은 default 값 사용 권장
# reference : Divide the gradient by a running average of its recent magnitude
# keras.optimizers.RMSprop(lr=0.001, rho=0.9, epsilon=None, decay=0.0)

# (3) Adagrad optimizer
# default 변수 값 사용 권장
# reference : Adaptive Subgradient Methods for Online Learning and Stochastic Optimization
# keras.optimizers.Adagrad(lr=0.01, epsilon=None, decay=0.0)

# (4) Adadelta optimizer
# default 변수 값 사용 권장
# reference : an adaptive learning rate method
# keras.optimizers.Adadelta(lr=1.0, rho=0.95, epsilon=None, decay=0.0)

# (5) Adam optimizer
# lr외의 변수들은 default 값 사용 권장
# reference : A Method for Stochastic Optimization
# keras.optimizers.Adam(lr=0.001, beta_1=0.9, beta_2=0.999, epsilon=None, decay=0.0, amsgrad=False)

# (6) Adamax optimizer
# Adam optimizer의 변형
# keras.optimizers.Adamax(lr=0.002, beta_1=0.9, beta_2=0.999, epsilon=None, decay=0.0)

# (7) Ndam optimizer
# Nesterov Adam optimizer
# keras.optimizers.Nadam(lr=0.002, beta_1=0.9, beta_2=0.999, epsilon=None, schedule_decay=0.004)

# (8) TFOptimizer optimizer
# Wrapper class for native TensorFlow optimizers.
# keras.optimizers.TFOptimizer(optimizer)

# model.compile(loss='mean_squared_error', optimizer='sgd')

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


#=============================================================================================
# 손실측정방법 설정  (참고:https://keras.io/losses/)
#=============================================================================================
# (1) mean_squared_error(y_true, y_pred)
```

```
# (2) mean_absolute_error(y_true, y_pred)
# (3) mean_absolute_percentage_error(y_true, y_pred)
# (4) mean_squared_logarithmic_error(y_true, y_pred)
# (5) squared_hinge(y_true, y_pred)
# (6) hinge(y_true, y_pred)
# (7) categorical_hinge(y_true, y_pred)
# (8) logcosh(y_true, y_pred)                //Logarithm of the hyperbolic cosine of the prediction error
# (9) categorical_crossentropy(y_true, y_pred) 외 5개 더 있음
#                 categorical format : 타겟값 1개 만 1이고 나머진 0의 값으로 설정

#=============================================================
# 학습 및 평가
#=============================================================
# 실제로 학습용데이터로 학습시키기, 반복횟수 5회, 웨이트 업데이트 시키는 데이터 단위 32개
model.fit(X_train, Y_train, epochs=5, batch_size=32)

# 테스트 데이터로 결과 평가하기
loss_and_metrics = model.evaluate(X_test, Y_test, batch_size=32)

print('LOSS & METRICS : ' +str(loss_and_metrics))

# 학습결과 저장
model.save('mnist_mlp_model.h5')

#=============================================================
# 시각화
#=============================================================

from IPython.display import SVG
from keras.utils.vis_utils import model_to_dot
from keras.models import load_model

# 학습결과 불러오기
model = load_model('mnist_mlp_model.h5')

# notebook을 실행한 브라우저에서 바로 그림을 볼 수 있게 해주는 명령
%matplotlib inline

SVG(model_to_dot(model, show_shapes=True).create(prog='dot', format='svg'))

'''
 ----------------------------------------------------------
 실행시 graphvis 파일 못찾아서 오류나는 경우 해결
 ----------------------------------------------------------
 https://graphviz.gitlab.io/_pages/Download/Download_windows.html 에 접속하여 graphviz-2.38.msi
 다운/설치하고,
 탐색기의 "내 PC"에서 오른쪽마우스->속성->고급시스템 설정->환경변수 선택하여
 새 시스템변수로 변수이름으로 GRAPHVIZ_DOT, 변수값을 Programfiles(86) 아래 있는
Graphviz2.38₩bin₩dot.exe 으로 추가하고 시스템변수 Path 에 찾아보기 버튼 이용하여 Programfiles(86) 아래 있는 Graphviz2.38₩bin 폴더 추가한 후 Command창 종료 후 다시 실행, activate keras 실행 후, jupyter notebook 다시 실행
'''
```

Jupyter Notebook에서 실행 장면

In [16]:
```
import scipy
import numpy
import matplotlib
import pandas
import sklearn
import pydotplus
import h5py
import tensorflow
import keras
import graphviz

from keras.utils import np_utils
from keras.datasets import mnist
from keras.models import Sequential
from keras.layers import Dense, Activation
```

In [17]:
```
(X_train, Y_train), (X_test, Y_test) = mnist.load_data()

# 6만개 학습용 입력 데이터 포맷을 float32 1이하 숫자로 reshape
X_train = X_train.reshape(60000, 784).astype('float32') / 255.0
# 1만개 테스트용 입력 데이터 포맷을 float32 1이하 숫자로 reshape
X_test = X_test.reshape(10000, 784).astype('float32') / 255.0

# Target 값 설정
Y_train = np_utils.to_categorical(Y_train)
Y_test = np_utils.to_categorical(Y_test)
```

In [18]:
```
'''
model = Sequential()

# DNN 네트워크 노드수/활성함수 설정
model.add(Dense(units=64, input_dim=28*28, activation='relu'))
model.add(Dense(units=10, activation='relu'))
'''
model = Sequential([
    Dense(32, input_shape=(784,)),
    Activation('relu'),
    Dense(10),
    Activation('softmax'),
])
```

In [19]:
```
model.compile(loss='categorical_crossentropy', optimizer='sgd', metrics=['accuracy'])
# 실제로 학습용데이터로 학습시키기, 반복횟수 5회, 웨이트 업데이트 시키는 데이터 단위 32개
model.fit(X_train, Y_train, epochs=5, batch_size=32)

# 테스트 데이터로 결과 평가하기
loss_and_metrics = model.evaluate(X_test, Y_test, batch_size=32)

print('LOSS & METRICS : ' +str(loss_and_metrics))

# 학습결과 저장
model.save('mnist_mlp_model.h5')
```

```
Epoch 1/5
60000/60000 [==============================] - 3s 55us/step - loss: 0.6871 - acc: 0.8232
Epoch 2/5
60000/60000 [==============================] - 3s 51us/step - loss: 0.3552 - acc: 0.9001: 2s - loss: 0.3945 - acc: -  - ETA: 0s - loss: 0.3
618 - ac
Epoch 3/5
60000/60000 [==============================] - 3s 50us/step - loss: 0.3085 - acc: 0.9126
Epoch 4/5
60000/60000 [==============================] - 3s 50us/step - loss: 0.2824 - acc: 0.9203
Epoch 5/5
60000/60000 [==============================] - 3s 50us/step - loss: 0.2635 - acc: 0.9258
10000/10000 [==============================] - 0s 27us/step
LOSS & METRICS : [0.24980599198639392, 0.9289]
```

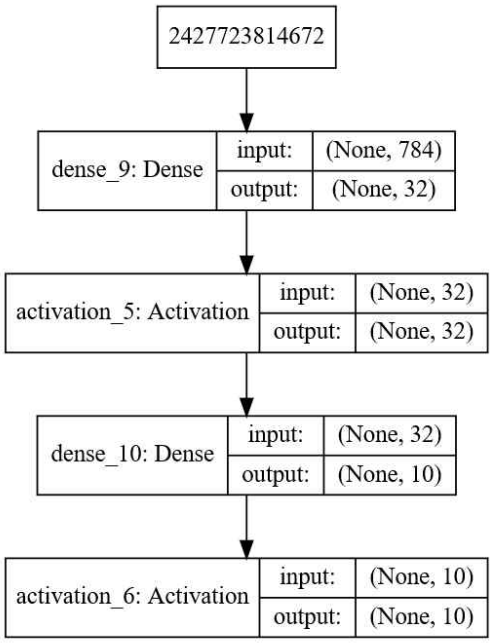
In [20]:
```
from IPython.display import SVG
from keras.utils.vis_utils import model_to_dot
from keras.models import load_model

# 학습결과 불러오기
model = load_model('mnist_mlp_model.h5')

# notebook을 실행한 브라우저에서 바로 그림을 볼 수 있게 해주는 명령
%matplotlib inline

SVG(model_to_dot(model, show_shapes=True).create(prog='dot', format='svg'))
```

Out[20]:



Keras 사용 예 (참고: https://keras.io/getting-started/sequential-model-guide/)

Multilayer Perceptron (MLP) for multi-class softmax classification:

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

# Generate dummy data
import numpy as np
x_train = np.random.random((1000, 20))
y_train = keras.utils.to_categorical(np.random.randint(10, size=(1000, 1)), num_classes=10)
x_test = np.random.random((100, 20))
y_test = keras.utils.to_categorical(np.random.randint(10, size=(100, 1)), num_classes=10)

model = Sequential()
# Dense(64) is a fully-connected layer with 64 hidden units.
# in the first layer, you must specify the expected input data shape:
# here, 20-dimensional vectors.
model.add(Dense(64, activation='relu', input_dim=20))
model.add(Dropout(0.5))
model.add(Dense(64, activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(10, activation='softmax'))

sgd = SGD(lr=0.01, decay=1e-6, momentum=0.9, nesterov=True)
model.compile(loss='categorical_crossentropy',
              optimizer=sgd,
              metrics=['accuracy'])

model.fit(x_train, y_train,
          epochs=20,
          batch_size=128)
score = model.evaluate(x_test, y_test, batch_size=128)
```

MLP for binary classification:

```
import numpy as np
from keras.models import Sequential
from keras.layers import Dense, Dropout

# Generate dummy data
x_train = np.random.random((1000, 20))
y_train = np.random.randint(2, size=(1000, 1))
x_test = np.random.random((100, 20))
y_test = np.random.randint(2, size=(100, 1))

model = Sequential()
model.add(Dense(64, input_dim=20, activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(64, activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(1, activation='sigmoid'))

model.compile(loss='binary_crossentropy',
              optimizer='rmsprop',
              metrics=['accuracy'])

model.fit(x_train, y_train,
          epochs=20,
          batch_size=128)
score = model.evaluate(x_test, y_test, batch_size=128)
```