구글 텐서플로우로 배우는 인공지능 체험

대구대학교 컴퓨터정보공학부(컴퓨터공학전공)
김종완교수

Part 3. 손 글씨 숫자 인식 실습 체험

구글 텐서플로우 소개 및 설치

MNIST 벤치마크 숫자 데이터 설명

텐서플로우로 손글씨 인식 과정 실습

학생 스스로 손글씨 인식 체험

1 PYTHON INSTALL

Python.org/downloads
접속 후 3.6.0 파일 다운

Looking for a specific release?

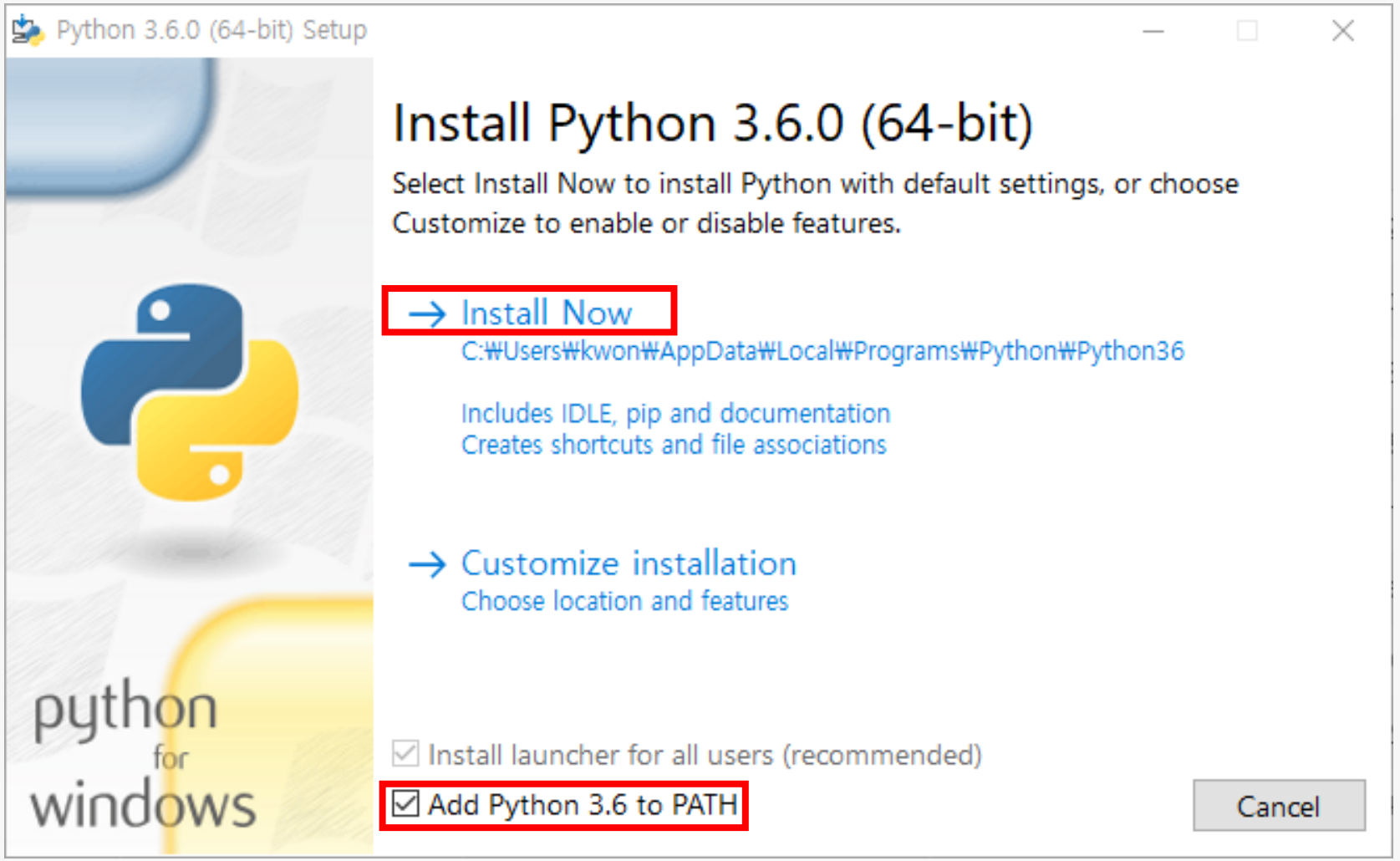
Python releases by version number:

Release version	Release date		Click for more
Python 3.6.1	2017-03-21	Download	Release Notes
Python 3.4.6	2017-01-17	Download	Release Notes
Python 3.5.3	2017-01-17	Download	Release Notes
Python 3.6.0	2016-12-23	Download	Release Notes
Python 2.7.13	2016-12-17	Download	Release Notes
Python 3.4.5	2016-06-27	Download	Release Notes
Python 3.5.2	2016-06-27	Download	Release Notes

View older releases

Files

Version	Operating System	Description	MD5 Sum	File Size	GPG
Gzipped source tarball	Source release		3f7062ccf8be76491884d0e47ac8b251	22256403	SIG
XZ compressed source tarball	Source release		82b143ebbf4514d7e05876bed7a6b1f5	16805836	SIG
Mac OS X 64-bit/32-bit installer	Mac OS X	for Mac OS X 10.6 and later	72acb0175e7622dec7e1b160a43b8c42	27442222	SIG
Windows help file	Windows		6a842a15ab3b4aa316c91a9779db82ec	7940890	SIG
Windows x86-64 embeddable zip file	Windows	for AMD64/EM64T/x64	0ec0caeea75bae5d2771cf619917c71f	6925798	SIG
Windows x86-64 executable installer	Windows	for AMD64/EM64T/x64	71c9d30c1110abf7f80a428970ab8ec2	31505640	SIG
Windows x86-64 web-based installer	Windows	for AMD64/EM64T/x64	25b8b6c93a098dfade3b014630f9508e	1312376	SIG
Windows x86 embeddable zip file	Windows		1adf2fb735c5000af32d42c39136727c	6315855	SIG
Windows x86 executable installer	Windows		38d9b036b25725f6acb553d4aece4db4	30566536	SIG
Windows x86 web-based installer	Windows		f71f4590be2cc5cdc43069594d4ea98d	1286984	SIG

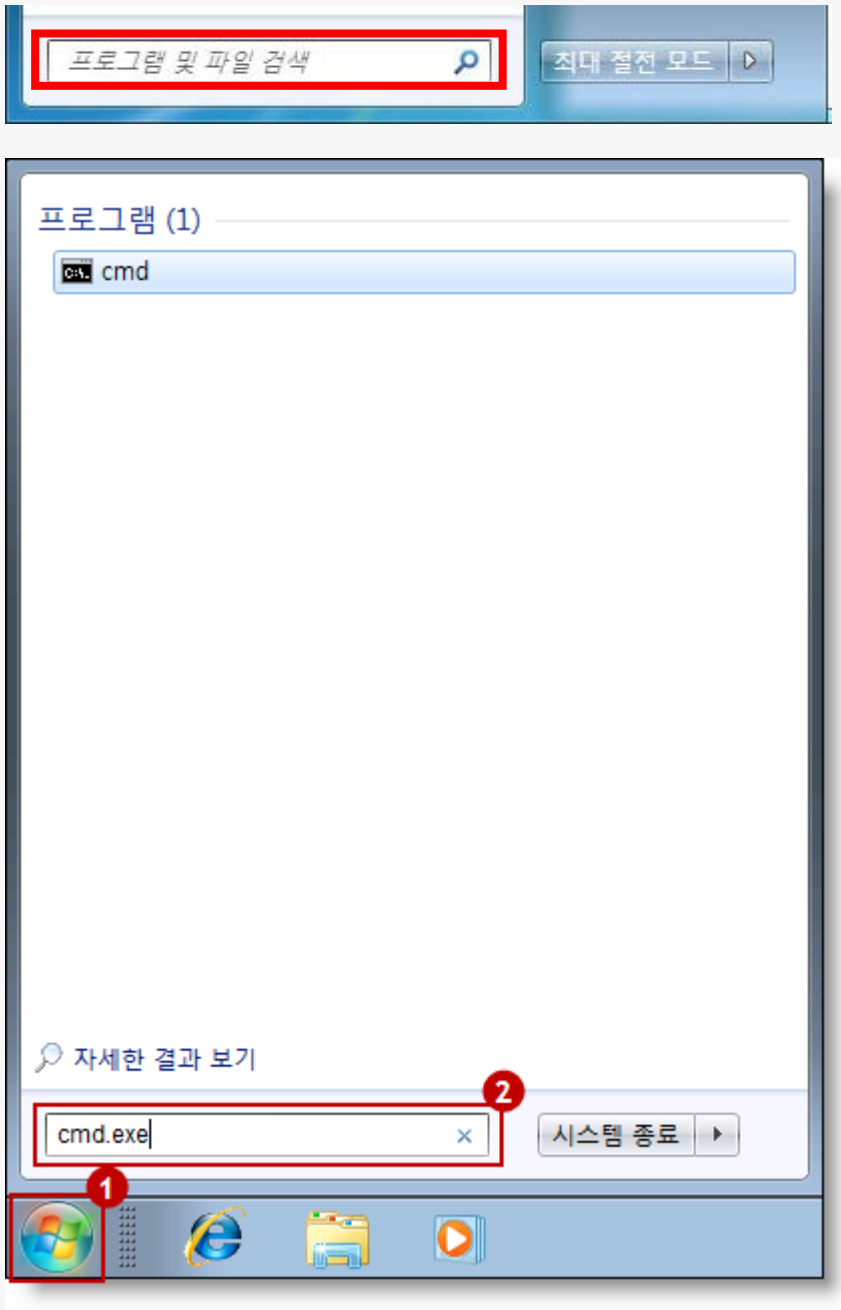

Python 3.6.0 (64-bit) Setup
Install Python 3.6.0 (64-bit)
Select Install Now to install Python with default settings, or choose
Customize to enable or disable features.
Install Now
C:₩Users₩kwon₩AppData₩Local₩Programs₩Python₩Python36
Includes IDLE, pip and documentation
Creates shortcuts and file associations
Customize installation
Choose location and features
python
for
windows
Install launcher for all users (recommended)
Add Python 3.6 to PATH
Cancel

설치 완료 후

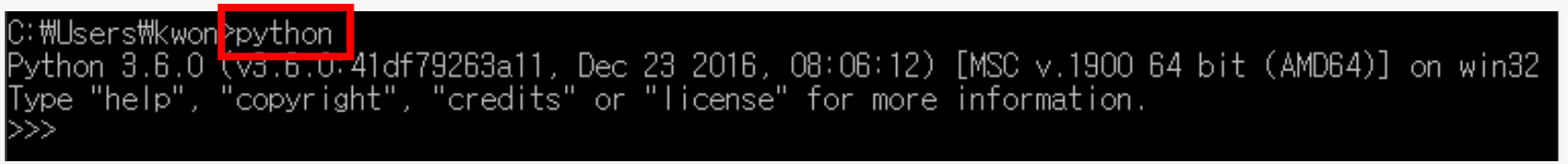
시작-검색: cmd.exe 실행

설치확인: python 입력

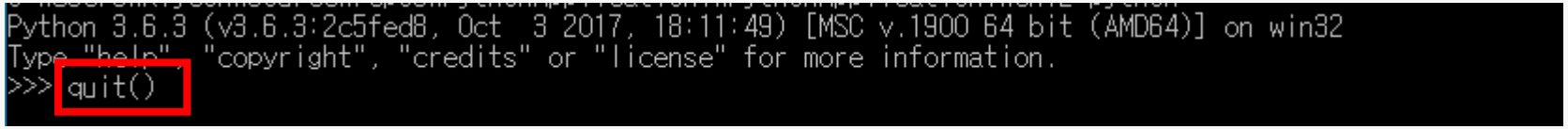
파이썬 종료 : quit() 입력



cmd 검색 후 cmd.exe 실행



'python' 입력 후 엔터. Python 3.6.x.... 이 나온다면 설치 완료



파이썬 인터프리터 종료 : quit()

2 LIBRARY INSTALL

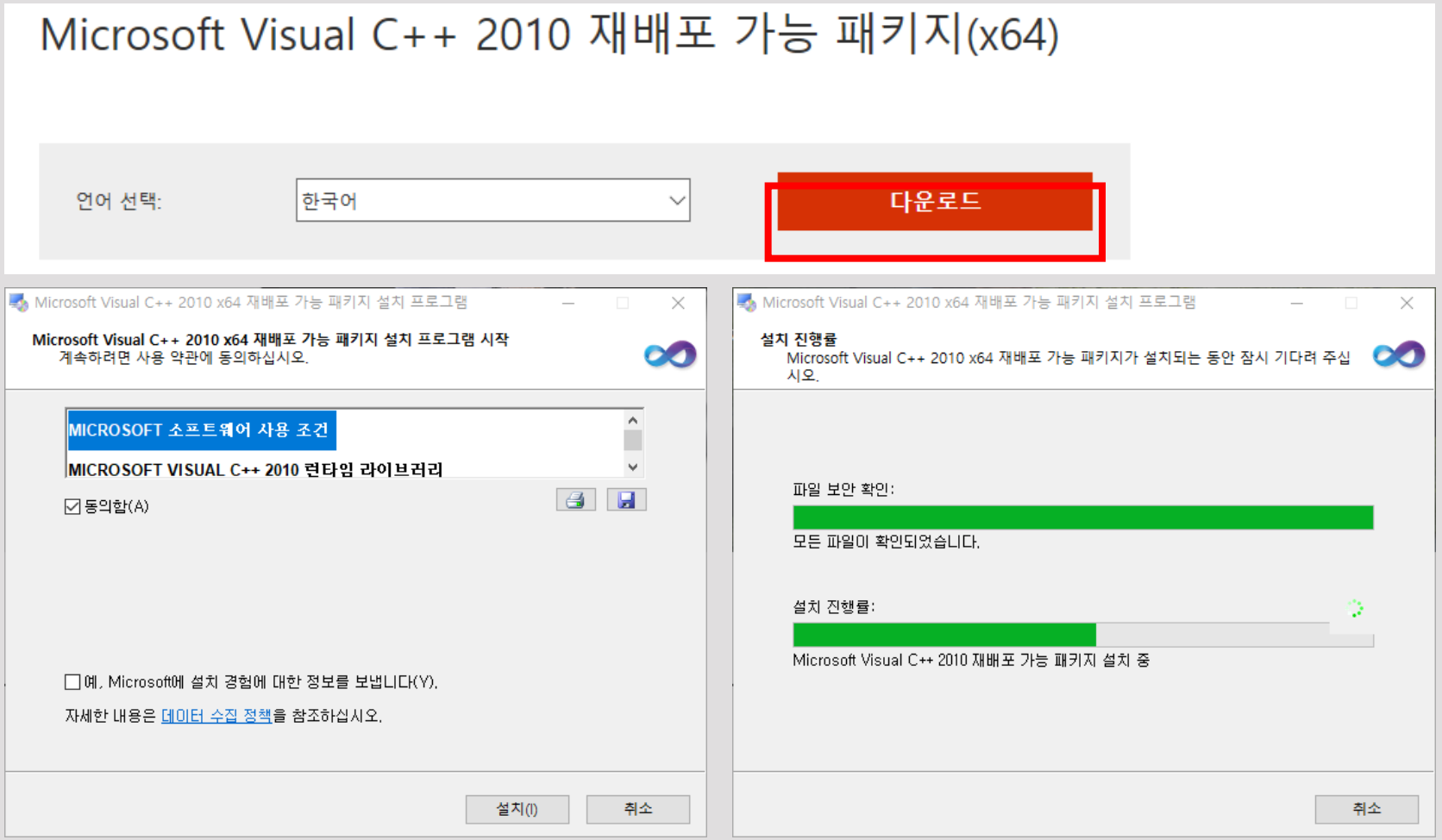
```
C:₩Users₩kwon>pip install tensorflow
Collecting tensorflow
  Downloading tensorflow-1.4.0-cp36-cp36m-win_amd64.whl (28.3MB)
    100% |████████████████████████████████| 28.3MB 49kB/s
Collecting numpy>=1.12.1 (from tensorflow)
  Using cached numpy-1.13.3-cp36-none-win_amd64.whl
Collecting protobuf>=3.3.0 (from tensorflow)
  Using cached protobuf-3.4.0-py2.py3-none-any.whl
Collecting tensorflow-tensorboard<0.5.0,>=0.4.0rc1 (from tensorflow)
  Downloading tensorflow_tensorboard-0.4.0rc2-py3-none-any.whl (1.7MB)
    100% |████████████████████████████████| 1.7MB 602kB/s
Collecting wheel>=0.26 (from tensorflow)
  Using cached wheel-0.30.0-py2.py3-none-any.whl
  Running setup.py install for html5lib ... done
  Running setup.py install for markdown ... done
Successfully installed bleach-1.5.0 enum34-1.1.6 html5lib-0.9999999 markdown-2.6.9 numpy-1.13.3 protobuf
0 tensorflow-1.4.0 tensorflow-tensorboard-0.4.0rc2 werkzeug-0.12.2 wheel-0.30.0
```

```
C:₩Users₩kwon>pip install matplotlib
Collecting matplotlib
  Using cached matplotlib-2.1.0-cp36-cp36m-win_amd64.whl
Installing collected packages: pyparsing, cycler, pytz, python-dateutil, matplotlib
Successfully installed cycler-0.10.0 matplotlib-2.1.0 pyparsing-2.2.0 python-dateutil-2.6.1 pytz-2017.3
```

- ✔ pip install tensorflow
- ✔ pip install matplotlib

Matplotlib 설치 이전 visual c++ 2010 설치 필요!

matplotlib을 이용하기 위해서는 C++ Package 필요



새로운 버전의 Visual C++ 2010 재배포 가능 패키지를 찾았습니다. 라는 오류가 나도 정상.

다운로드 : https://www.microsoft.com/ko-kr/download/details.aspx?id=14632

3 RUN EXAMPLE

1. 실습파일
test.py, test2.py를
내 문서 폴더에 두기

2. cmd에서
cd Documents 입력

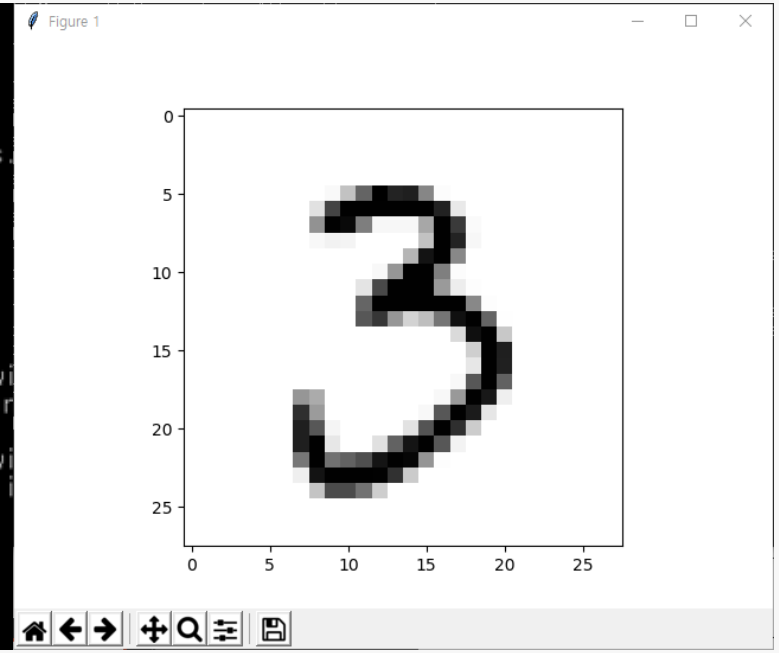
3. test.py 입력하여 실행

실습파일(test.py, test2.py) 다운로드

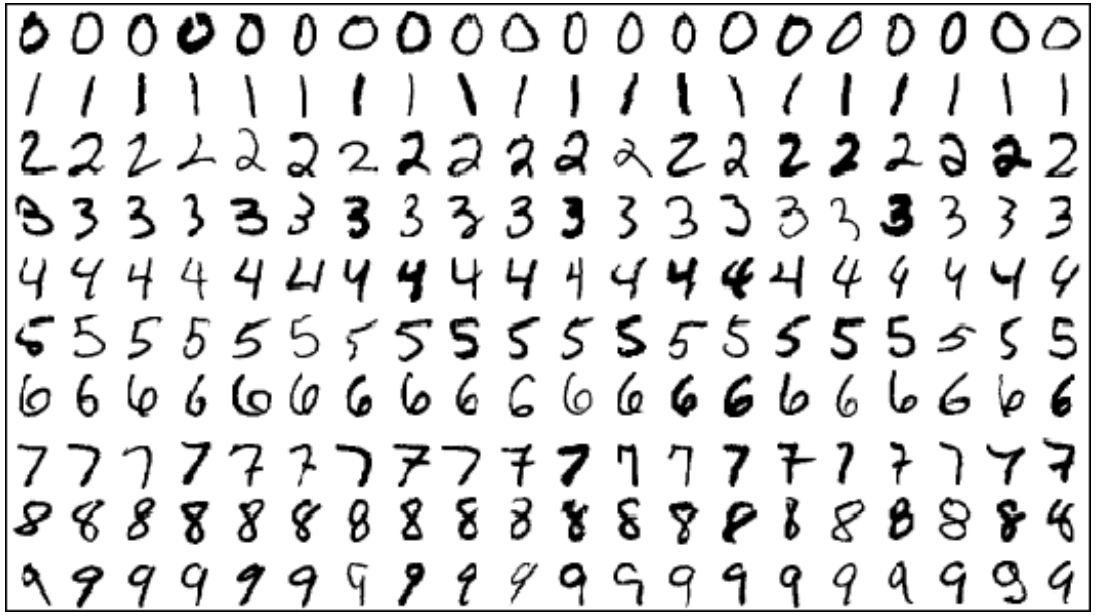
https://1drv.ms/f/s!AjV0uXQmsnP9i4hmGLc4JguNmnDZgw

실행 결과

```
C:¥Users¥kwon>cd Documents

C:¥Users¥kwon¥Documents>

C:¥Users¥kwon¥Documents>test.py
Successfully downloaded train-images-idx3-ubyte.gz 9912422 bytes.
Extracting MNIST_data/train-images-idx3-ubyte.gz
Successfully downloaded train-labels-idx1-ubyte.gz 28881 bytes.
Extracting MNIST_data/train-labels-idx1-ubyte.gz
Successfully downloaded t10k-images-idx3-ubyte.gz 1648877 bytes.
Extracting MNIST_data/t10k-images-idx3-ubyte.gz
Successfully downloaded t10k-labels-idx1-ubyte.gz 4542 bytes.
Extracting MNIST_data/t10k-labels-idx1-ubyte.gz
2017-11-12 19:19:59.388488: W C:¥tf_jenkins¥home¥workspace¥rel-wi
guard.cc:45] The TensorFlow library wasn't compiled to use AVX ir
 could speed up CPU computations.
2017-11-12 19:19:59.388605: W C:¥tf_jenkins¥home¥workspace¥rel-wi
guard.cc:45] The TensorFlow library wasn't compiled to use AVX2 i
d could speed up CPU computations.
Learning finished
Accuracy:  0.8951
Label:  [3]
Prediction:  [3]
```



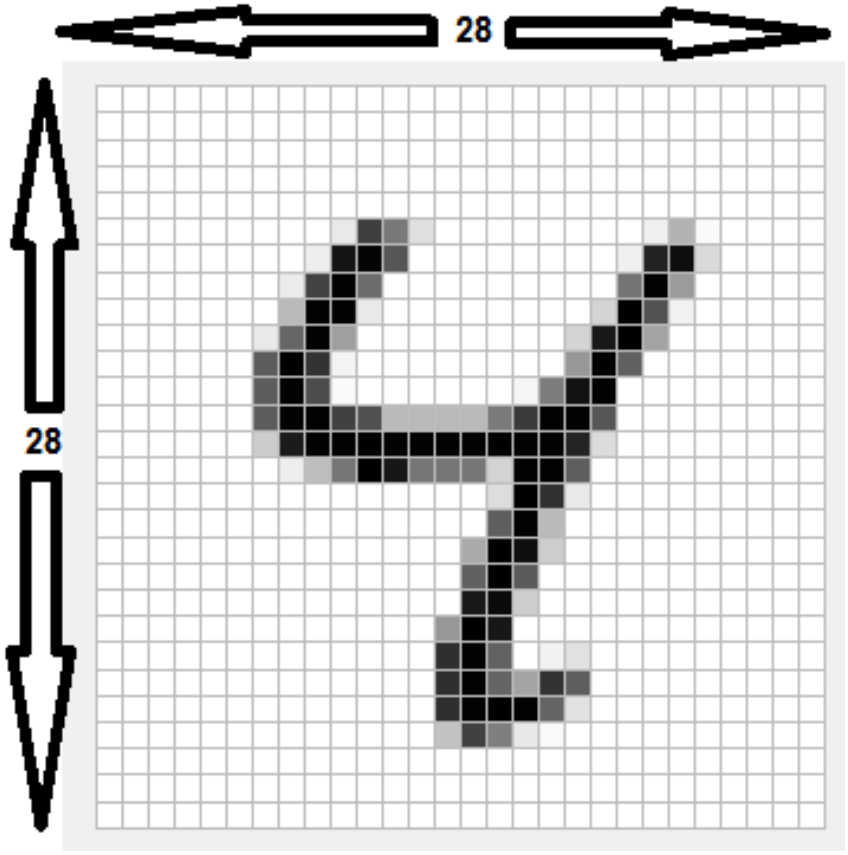
MNIST Dataset



train-images-idx3-ubyte.gz:	training set images (9912422 bytes)
train-labels-idx1-ubyte.gz:	training set labels (28881 bytes)
t10k-images-idx3-ubyte.gz:	test set images (1648877 bytes)
t10k-labels-idx1-ubyte.gz:	test set labels (4542 bytes)

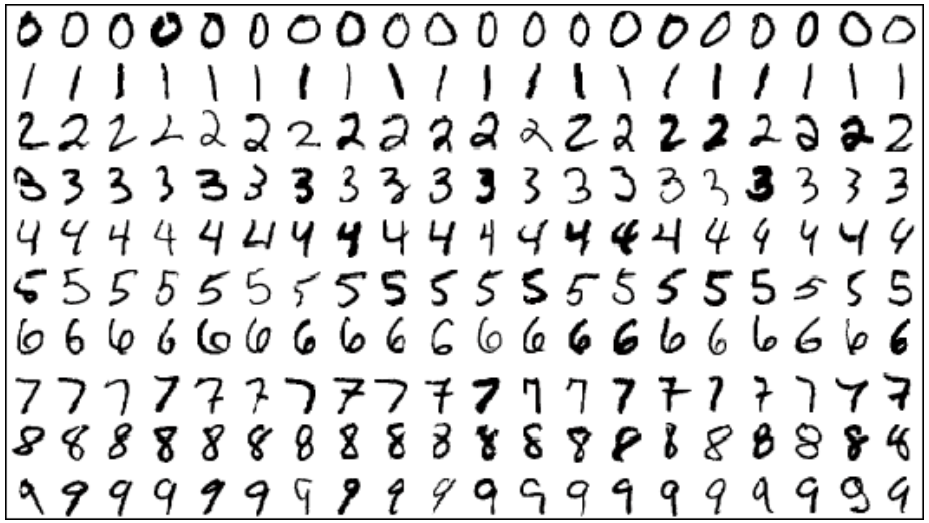
http://yann.lecun.com/exdb/mnist/

28x28x1 image




```
# MNIST data image of shape 28 * 28 = 784
X = tf.placeholder(tf.float32, [None, 784])
# 0 - 9 digits recognition = 10 classes
Y = tf.placeholder(tf.float32, [None, nb_classes])
```

http://derindelimavi.blogspot.hk/2015/04/mnist-el-yazs-rakam-veri-seti.html



MNIST Dataset

```
from tensorflow.examples.tutorials.mnist import input_data
# Check out https://www.tensorflow.org/get_started/mnist/beginners for
# more information about the mnist dataset
mnist = input_data.read_data_sets("MNIST_data/", one_hot=True)
…
batch_xs, batch_ys = mnist.train.next_batch(100)
…
print("Accuracy: ", accuracy.eval(session=sess,
        feed_dict={X: mnist.test.images, Y: mnist.test.labels}))
```

https://github.com/hunkim/DeepLearningZeroToAll/blob/master/lab-07-4-mnist_introduction.py

Reading data and set variables

```
from tensorflow.examples.tutorials.mnist import input_data
# Check out https://www.tensorflow.org/get_started/mnist/beginners for
# more information about the mnist dataset
mnist = input_data.read_data_sets("MNIST_data/", one_hot=True)

nb_classes = 10

# MNIST data image of shape 28 * 28 = 784
X = tf.placeholder(tf.float32, [None, 784])
# 0 - 9 digits recognition = 10 classes
Y = tf.placeholder(tf.float32, [None, nb_classes])

W = tf.Variable(tf.random_normal([784, nb_classes]))
b = tf.Variable(tf.random_normal([nb_classes]))
```

https://github.com/hunkim/DeepLearningZeroToAll/blob/master/lab-07-4-mnist_introduction.py

Softmax!

```
# Hypothesis (using softmax)
hypothesis = tf.nn.softmax(tf.matmul(X, W) + b)

cost = tf.reduce_mean(-tf.reduce_sum(Y * tf.log(hypothesis), axis=1))
optimizer = tf.train.GradientDescentOptimizer(learning_rate=0.1).minimize(cost)

# Test model
is_correct = tf.equal(tf.arg_max(hypothesis, 1), tf.arg_max(Y, 1))
# Calculate accuracy
accuracy = tf.reduce_mean(tf.cast(is_correct, tf.float32))
```

https://github.com/hunkim/DeepLearningZeroToAll/blob/master/lab-07-4-mnist_introduction.py

Training epoch/batch

```
# parameters
training_epochs = 15
batch_size = 100

with tf.Session() as sess:
  # Initialize TensorFlow variables
  sess.run(tf.global_variables_initializer())
  # Training cycle
  for epoch in range(training_epochs):
    avg_cost = 0
    total_batch = int(mnist.train.num_examples / batch_size)

    for i in range(total_batch):
      batch_xs, batch_ys = mnist.train.next_batch(batch_size)
      c, _ = sess.run([cost, optimizer], feed_dict={X: batch_xs, Y: batch_ys})
      avg_cost += c / total_batch

    print('Epoch:', '%04d' % (epoch + 1), 'cost =', '{:.9f}'.format(avg_cost))
```

https://github.com/hunkim/DeepLearningZeroToAll/blob/master/lab-07-4-mnist_introduction.py

Training epoch/batch

In the neural network terminology:

one **epoch** = one forward pass and one backward pass of *all* the training examples

batch size = the number of training examples in one forward/backward pass. The higher the batch size, t he more memory space you'll need.

number of **iterations** = number of passes, each pass using [batch size] number of examples. To be clear, o ne pass = one forward pass + one backward pass (we do not count the forward pass and backward pass a s two different passes).

Example: if you have *1000 training examples*, and your *batch size is 500*, then it will take 2 iterations to complete 1 epoch.

http://stackoverflow.com/questions/4752626/epoch-vs-iteration-when-training-neural-networks

Training epoch/batch

```
# parameters
training_epochs = 15
batch_size = 100

with tf.Session() as sess:
  # Initialize TensorFlow variables
  sess.run(tf.global_variables_initializer())
  # Training cycle
  for epoch in range(training_epochs):
    avg_cost = 0
    total_batch = int(mnist.train.num_examples / batch_size)

    for i in range(total_batch):
      batch_xs, batch_ys = mnist.train.next_batch(batch_size)
      c, _ = sess.run([cost, optimizer], feed_dict={X: batch_xs, Y: batch_ys})
      avg_cost += c / total_batch

    print('Epoch:', '%04d' % (epoch + 1), 'cost =', '{:.9f}'.format(avg_cost))
```

https://github.com/hunkim/DeepLearningZeroToAll/blob/master/lab-07-4-mnist_introduction.py

Report results on test dataset

```
# Test the model using test sets
print("Accuracy: ", accuracy.eval(session=sess,
      feed_dict={X: mnist.test.images, Y: mnist.test.labels}))
```

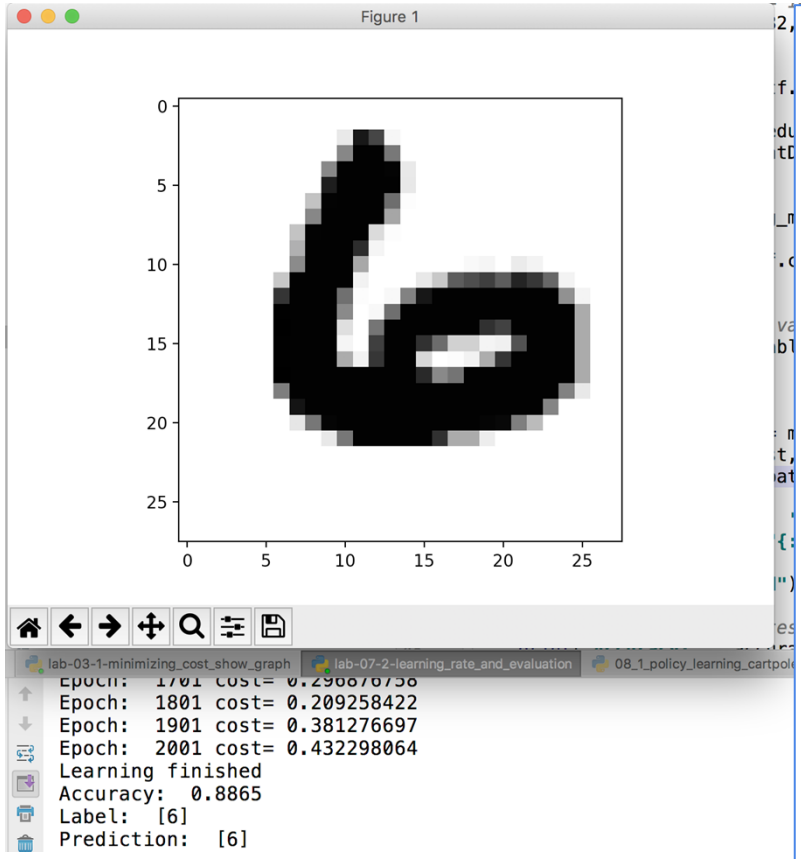
https://github.com/hunkim/DeepLearningZeroToAll/blob/master/lab-07-4-mnist_introduction.py

```
hypothesis = tf.nn.softmax(tf.matmul(X, W) + b)
cost = tf.reduce_mean(-tf.reduce_sum(Y * tf.log(hypothesis), axis=1))
optimizer = tf.train.GradientDescentOptimizer(learning_rate=0.1).minimize(cost)

is_correct = tf.equal(tf.arg_max(hypothesis, 1), tf.arg_max(Y, 1))
accuracy = tf.reduce_mean(tf.cast(is_correct, tf.float32))

# parameters
training_epochs = 15
batch_size = 100

with tf.Session() as sess:
  # Initialize TensorFlow variables
  sess.run(tf.global_variables_initializer())
  # Training cycle
  for epoch in range(training_epochs):
    avg_cost = 0
    total_batch = int(mnist.train.num_examples / batch_size)

    for i in range(total_batch):
      batch_xs, batch_ys = mnist.train.next_batch(batch_size)
      c, _ = sess.run([cost, optimizer],
                      feed_dict={X: batch_xs, Y: batch_ys})
      avg_cost += c / total_batch

    print('Epoch:', '%04d' % (epoch + 1),
                      'cost =', '{:.9f}'.format(avg_cost))
```

```
Epoch: 0001 cost = 2.868104637
Epoch: 0002 cost = 1.134684615
Epoch: 0003 cost = 0.908220728
Epoch: 0004 cost = 0.794199896
Epoch: 0005 cost = 0.721815854
Epoch: 0006 cost = 0.670184430
Epoch: 0007 cost = 0.630576546
Epoch: 0008 cost = 0.598888191
Epoch: 0009 cost = 0.573027079
Epoch: 0010 cost = 0.550497213
Epoch: 0011 cost = 0.532001859
Epoch: 0012 cost = 0.515517795
Epoch: 0013 cost = 0.501175288
Epoch: 0014 cost = 0.488425370
Epoch: 0015 cost = 0.476968593
Learning finished
Accuracy:  0.888
```

https://github.com/hunkim/DeepLearningZeroToAll/blob/master/lab-07-4-mnist_introduction.py

Sample image show and prediction




```
import matplotlib.pyplot as plt
import random

# Get one and predict
r = random.randint(0, mnist.test.num_examples - 1)
print("Label:", sess.run(tf.argmax(mnist.test.labels[r:r+1], 1)))
print("Prediction:", sess.run(tf.argmax(hypothesis, 1),
                              feed_dict={X: mnist.test.images[r:r + 1]}))

plt.imshow(mnist.test.images[r:r + 1].
      reshape(28, 28), cmap='Greys', interpolation='nearest')
plt.show()
```

https://github.com/hunkim/DeepLearningZeroToAll/blob/master/lab-07-4-mnist_introduction.py

NN for MNIST

```
# input place holders
X = tf.placeholder(tf.float32, [None, 784])
Y = tf.placeholder(tf.float32, [None, 10])

# weights & bias for nn layers
W1 = tf.Variable(tf.random_normal([784, 256]))
b1 = tf.Variable(tf.random_normal([256]))
L1 = tf.nn.relu(tf.matmul(X, W1) + b1)

W2 = tf.Variable(tf.random_normal([256, 256]))
b2 = tf.Variable(tf.random_normal([256]))
L2 = tf.nn.relu(tf.matmul(L1, W2) + b2)

W3 = tf.Variable(tf.random_normal([256, 10]))
b3 = tf.Variable(tf.random_normal([10]))
hypothesis = tf.matmul(L2, W3) + b3

# define cost/loss & optimizer
cost = tf.reduce_mean(tf.nn.softmax_cross_entropy_with_logits(
  logits=hypothesis, labels=Y))
optimizer = tf.train.AdamOptimizer(learning_rate=learning_rate).minimize(cost)
```

https://github.com/hunkim/DeepLearningZeroToAll/blob/master/lab-10-2-mnist_nn.py

```
# define cost/loss & optimizer
cost = tf.reduce_mean(tf.nn.softmax_cross_entropy_with_logits(logits=hypothesis, labels=Y))
optimizer = tf.train.AdamOptimizer(learning_rate=learning_rate).minimize(cost)
# initialize
sess = tf.Session()
sess.run(tf.global_variables_initializer())

# train my model
for epoch in range(training_epochs):
  avg_cost = 0
  total_batch = int(mnist.train.num_examples / batch_size)

  for i in range(total_batch):
    batch_xs, batch_ys = mnist.train.next_batch(batch_size)
    feed_dict = {X: batch_xs, Y: batch_ys}
    c, _ = sess.run([cost, optimizer], feed_dict=feed_dict)
    avg_cost += c / total_batch

  print('Epoch:', '%04d' % (epoch + 1), 'cost =', '{:.9f}'.format(avg_cost))

print('Learning Finished!')

# Test model and check accuracy
correct_prediction = tf.equal(tf.argmax(hypothesis, 1), tf.argmax(Y, 1))
accuracy = tf.reduce_mean(tf.cast(correct_prediction, tf.float32))
print('Accuracy:', sess.run(accuracy, feed_dict={X: mnist.test.images, Y: mnist.test.labels}))
```

https://github.com/hunkim/DeepLearningZeroToAll/blob/master/lab-10-1-mnist_softmax.py

```
# Get one and predict
r = random.randint(0, mnist.test.num_examples - 1)
print("Label: ", sess.run(tf.argmax(mnist.test.labels[r:r + 1], 1)))
print("Prediction: ", sess.run(
    tf.argmax(hypothesis, 1), feed_dict={X: mnist.test.images[r:r + 1]}))

plt.imshow(mnist.test.images[r:r + 1].
          reshape(28, 28), cmap='Greys', interpolation='nearest')
plt.show()
```

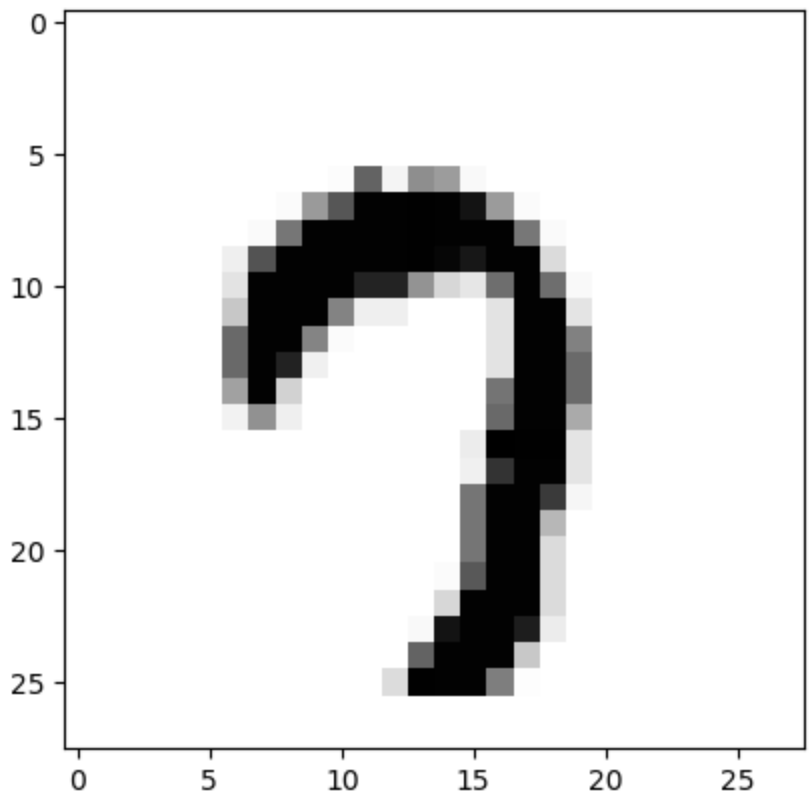
```
Epoch: 0001 cost = 142.446750370
Epoch: 0002 cost = 38.833989278
Epoch: 0003 cost = 24.367315463
Epoch: 0004 cost = 16.944873828
Epoch: 0005 cost = 12.304241501
Epoch: 0006 cost = 9.198538420
Epoch: 0007 cost = 6.912528261
Epoch: 0008 cost = 5.159236677
Epoch: 0009 cost = 3.764268593
Epoch: 0010 cost = 2.908739477
Epoch: 0011 cost = 2.250137948
Epoch: 0012 cost = 1.619561418
Epoch: 0013 cost = 1.243446105
Epoch: 0014 cost = 1.036443907
Epoch: 0015 cost = 0.765386158
Learning Finished!
Accuracy: 0.9427
```

https://github.com/hunkim/DeepLearningZeroToAll/blob/master/lab-10-1-mnist_softmax.py

C:₩Users₩Kiyeon₩source₩repos₩PythonApplication1₩P

```
Instructions for updating:

Future major versions of TensorFlow will a
into the labels input on backprop by defau

See tf.nn.softmax_cross_entropy_with_logit

2018-04-04 21:03:40.685353: I T:₩src₩githu
rts instructions that this TensorFlow bina
Epoch: 0001 cost = 142.446750370
Epoch: 0002 cost = 38.833989278
Epoch: 0003 cost = 24.367315463
Epoch: 0004 cost = 16.944873828
Epoch: 0005 cost = 12.304241501
Epoch: 0006 cost = 9.198538420
Epoch: 0007 cost = 6.912528261
Epoch: 0008 cost = 5.159236677
Epoch: 0009 cost = 3.764268593
Epoch: 0010 cost = 2.908739477
Epoch: 0011 cost = 2.250137948
Epoch: 0012 cost = 1.619561418
Epoch: 0013 cost = 1.243446105
Epoch: 0014 cost = 1.036443907
Epoch: 0015 cost = 0.765386158
Learning Finished!
Accuracy: 0.9427
Label:  [7]
Prediction:  [7]
```

Figure 1



THANK

YOU