TENSORFLOW

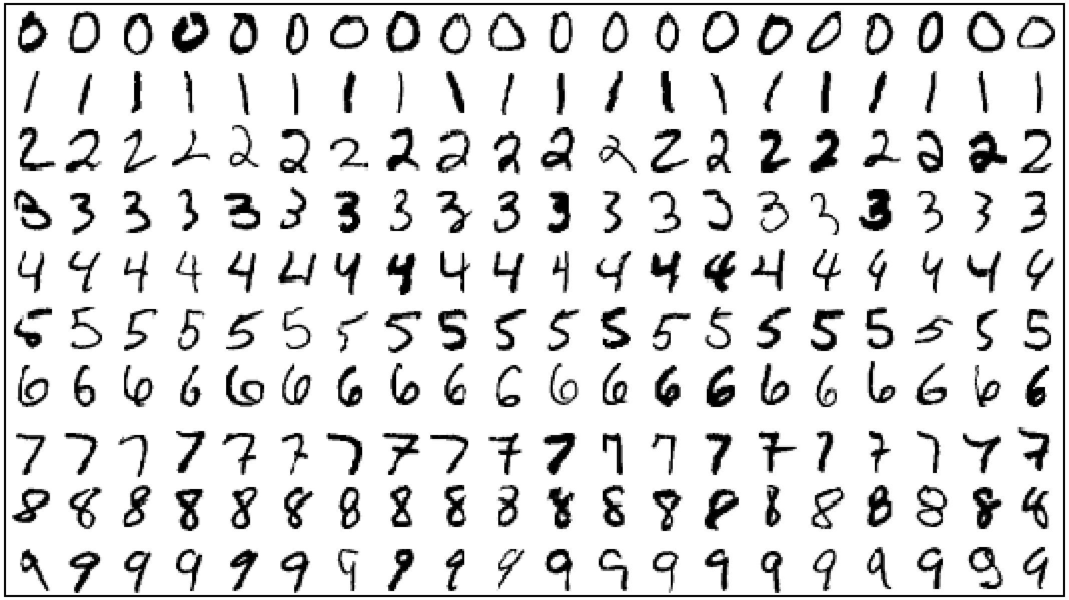
What is TensorFlow?

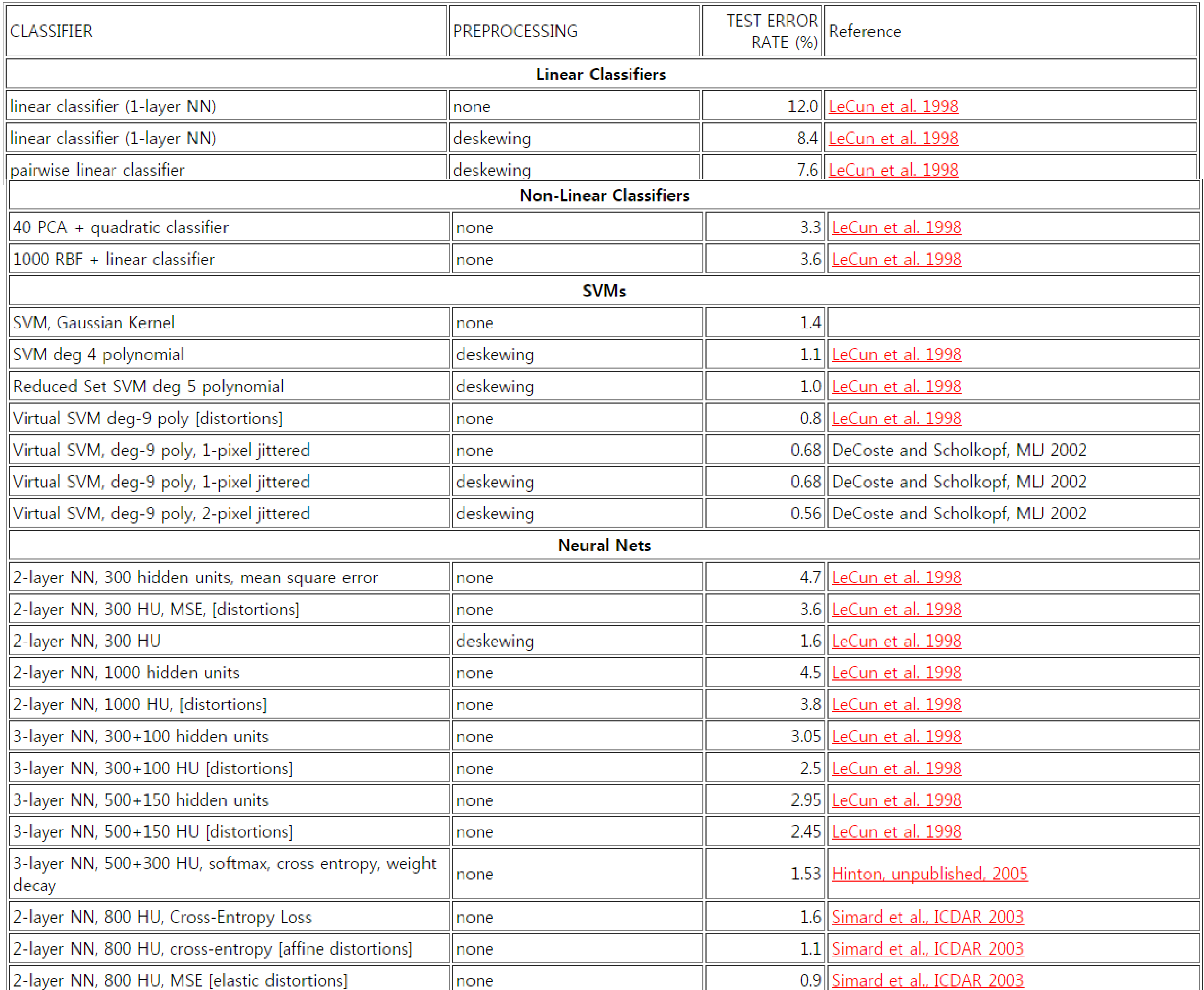
- TensorFlow is a deep learning library open-sourced by Google.
- TensorFlow provides primitives for defining functions on tensors and automatically computing their derivatives.
- Tensor is a multidimensional array of numbers

(SMALL) DATASET EXAMPLES

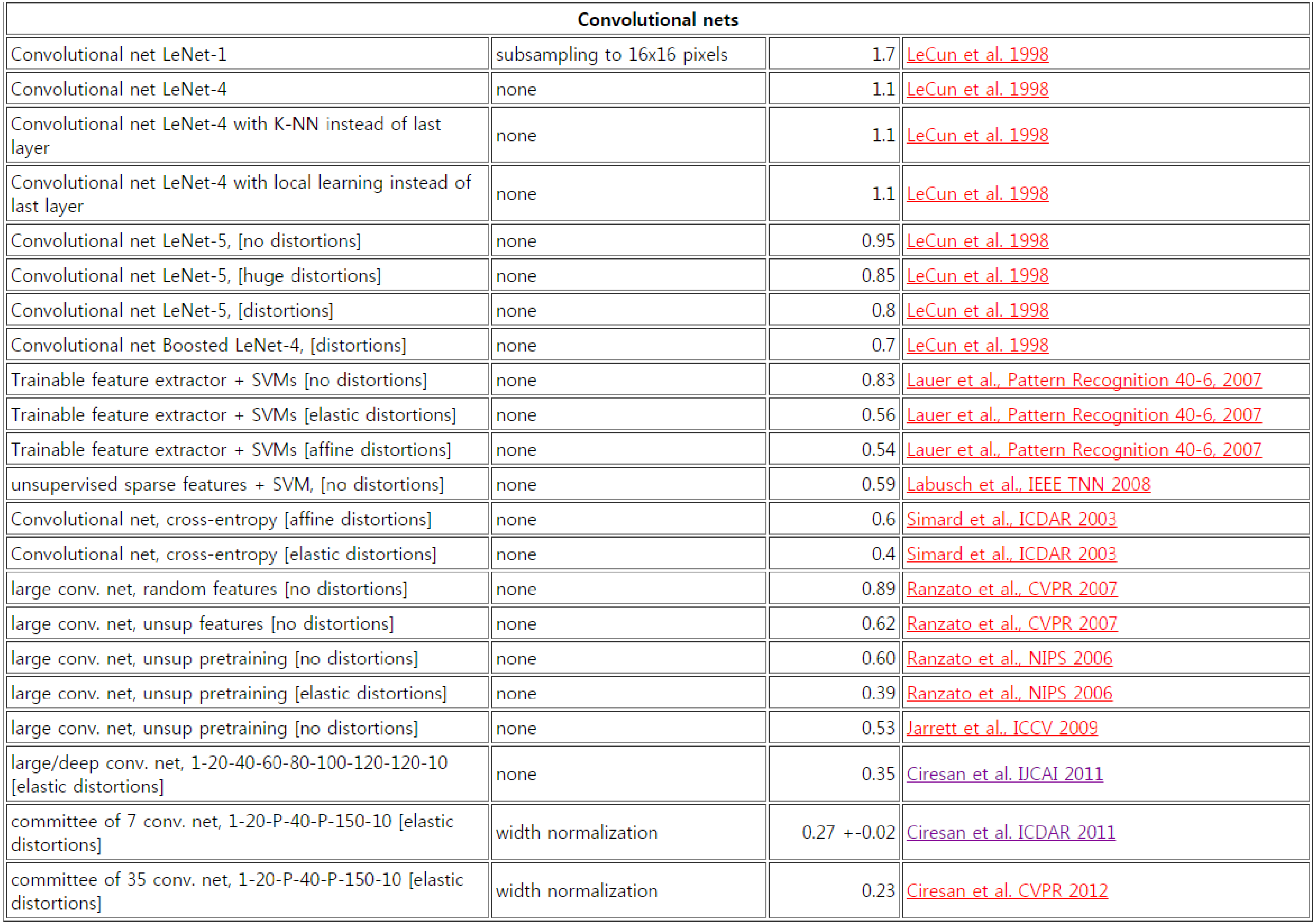
MNIST dataset

- handwritten digits
- a training set of 60,000 examples
- 24x24 images





CLASSIFIER	PREPROCESSING	TEST ERROR RATE (%)	Reference
Linear Classifiers			
linear classifier (1-layer NN)	none	12.0	LeCun et al. 1998
linear classifier (1-layer NN)	deskewing	8.4	LeCun et al. 1998
pairwise linear classifier	deskewing	7.6	LeCun et al. 1998
Non-Linear Classifiers			
40 PCA + quadratic classifier	none	3.3	LeCun et al. 1998
1000 RBF + linear classifier	none	3.6	LeCun et al. 1998
SVMs			
SVM, Gaussian Kernel	none	1.4	
SVM deg 4 polynomial	deskewing	1.1	LeCun et al. 1998
Reduced Set SVM deg 5 polynomial	deskewing	1.0	LeCun et al. 1998
Virtual SVM deg-9 poly [distortions]	none	0.8	LeCun et al. 1998
Virtual SVM, deg-9 poly, 1-pixel jittered	none	0.68	DeCoste and Scholkopf, MLJ 2002
Virtual SVM, deg-9 poly, 1-pixel jittered	deskewing	0.68	DeCoste and Scholkopf, MLJ 2002
Virtual SVM, deg-9 poly, 2-pixel jittered	deskewing	0.56	DeCoste and Scholkopf, MLJ 2002
Neural Nets			
2-layer NN, 300 hidden units, mean square error	none	4.7	LeCun et al. 1998
2-layer NN, 300 HU, MSE, [distortions]	none	3.6	LeCun et al. 1998
2-layer NN, 300 HU	deskewing	1.6	LeCun et al. 1998
2-layer NN, 1000 hidden units	none	4.5	LeCun et al. 1998
2-layer NN, 1000 HU, [distortions]	none	3.8	LeCun et al. 1998
3-layer NN, 300+100 hidden units	none	3.05	LeCun et al. 1998
3-layer NN, 300+100 HU [distortions]	none	2.5	LeCun et al. 1998
3-layer NN, 500+150 hidden units	none	2.95	LeCun et al. 1998
3-layer NN, 500+150 HU [distortions]	none	2.45	LeCun et al. 1998
3-layer NN, 500+300 HU, softmax, cross entropy, weight decay	none	1.53	Hinton, unpublished, 2005
2-layer NN, 800 HU, Cross-Entropy Loss	none	1.6	Simard et al., ICDAR 2003
2-layer NN, 800 HU, cross-entropy [affine distortions]	none	1.1	Simard et al., ICDAR 2003
2-layer NN, 800 HU, MSE [elastic distortions]	none	0.9	Simard et al., ICDAR 2003



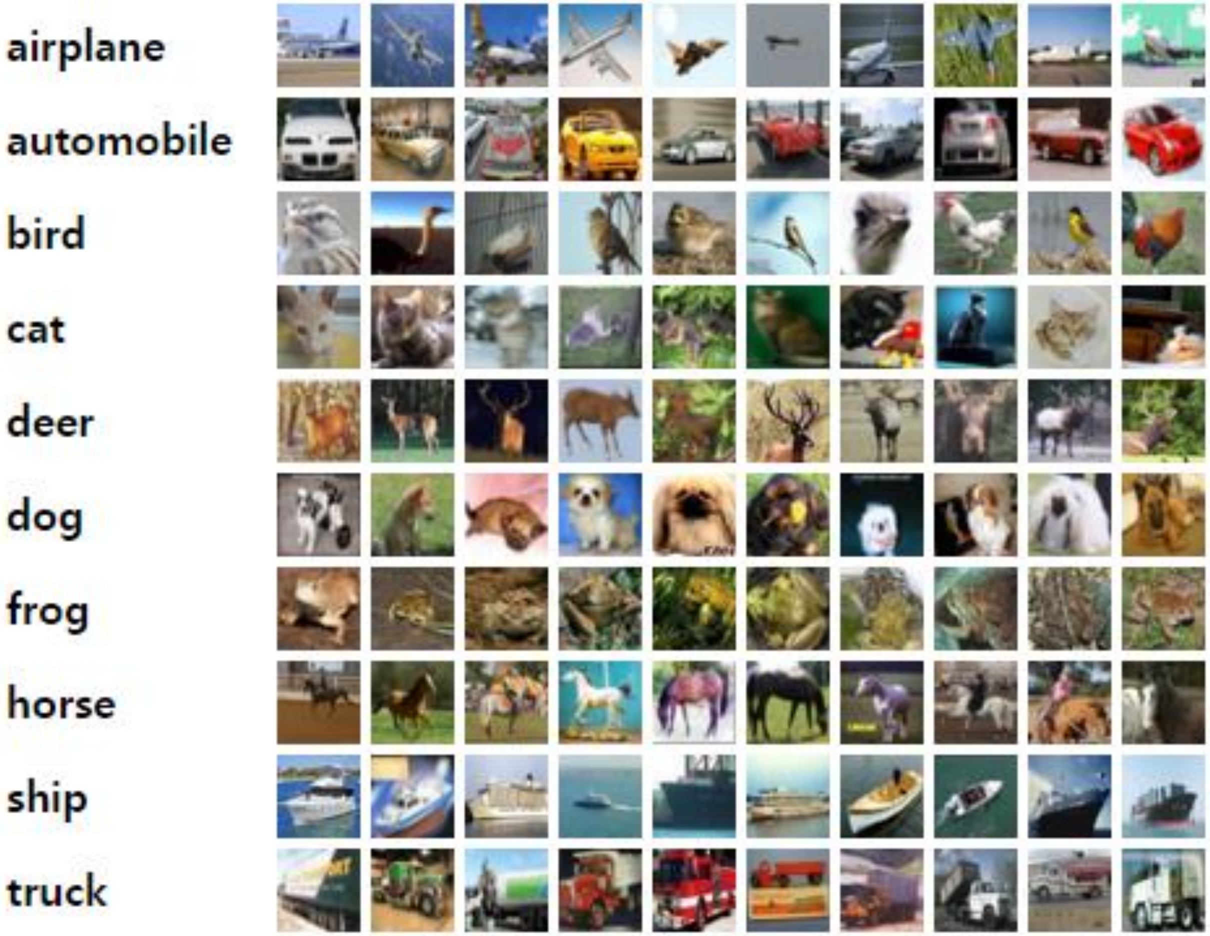
Convolutional nets

Convolutional net LeNet-1	subsampling to 16x16 pixels	1.7	LeCun et al. 1998
Convolutional net LeNet-4	none	1.1	LeCun et al. 1998
Convolutional net LeNet-4 with K-NN instead of last layer	none	1.1	LeCun et al. 1998
Convolutional net LeNet-4 with local learning instead of last layer	none	1.1	LeCun et al. 1998
Convolutional net LeNet-5, [no distortions]	none	0.95	LeCun et al. 1998
Convolutional net LeNet-5, [huge distortions]	none	0.85	LeCun et al. 1998
Convolutional net LeNet-5, [distortions]	none	0.8	LeCun et al. 1998
Convolutional net Boosted LeNet-4, [distortions]	none	0.7	LeCun et al. 1998
Trainable feature extractor + SVMs [no distortions]	none	0.83	Lauer et al., Pattern Recognition 40-6, 2007
Trainable feature extractor + SVMs [elastic distortions]	none	0.56	Lauer et al., Pattern Recognition 40-6, 2007
Trainable feature extractor + SVMs [affine distortions]	none	0.54	Lauer et al., Pattern Recognition 40-6, 2007
unsupervised sparse features + SVM, [no distortions]	none	0.59	Labusch et al., IEEE TNN 2008
Convolutional net, cross-entropy [affine distortions]	none	0.6	Simard et al., ICDAR 2003
Convolutional net, cross-entropy [elastic distortions]	none	0.4	Simard et al., ICDAR 2003
large conv. net, random features [no distortions]	none	0.89	Ranzato et al., CVPR 2007
large conv. net, unsup features [no distortions]	none	0.62	Ranzato et al., CVPR 2007
large conv. net, unsup pretraining [no distortions]	none	0.60	Ranzato et al., NIPS 2006
large conv. net, unsup pretraining [elastic distortions]	none	0.39	Ranzato et al., NIPS 2006
large conv. net, unsup pretraining [no distortions]	none	0.53	Jarrett et al., ICCV 2009
large/deep conv. net, 1-20-40-60-80-100-120-120-10 [elastic distortions]	none	0.35	Ciresan et al. IJCAI 2011
committee of 7 conv. net, 1-20-P-40-P-150-10 [elastic distortions]	width normalization	0.27 +-0.02	Ciresan et al. ICDAR 2011
committee of 35 conv. net, 1-20-P-40-P-150-10 [elastic distortions]	width normalization	0.23	Ciresan et al. CVPR 2012

CIFAR-10 dataset

- CIFAR-10
  - A labeled subset of the 80 million tiny images dataset
  - Collected by Alex Krizhevsky, Vinod Nair, and Geoffrey Hinton
  - Available at http://www.cs.toronto.edu/~kriz/cifar.html
  - Consisting of 60000 32x32 color images in 10 classes, with 6000 images per class.
  - 50000 training images and 10000 test images.

Samples


airplane
automobile
bird
cat
deer
dog
frog
horse
ship
truck

TENSORFLOW INTRODUCTION

What we should design/choose/prepare

- Input output formats
- Network structures
  - The mathematical relationship between inputs and outputs
  - Variables + Functions
- Loss function
- Training data
- Optimization schedule
  - Optimization methods
  - Hyper-parameters

TENSORFLOW BASICS

TF basics

- Hello World…
  - mul=i1*(i2+5)

- Linear Regression

- Code Examples

TF basics

- Session
- InteractiveSession
    - Session examples

TENSORFLOW
NEURAL NETWORKS EXAMPLE

예제코드-1

- import input_datamnist = input_data.read_data_sets('MNIST_data', one_hot = True)
- import tensorflow as tf

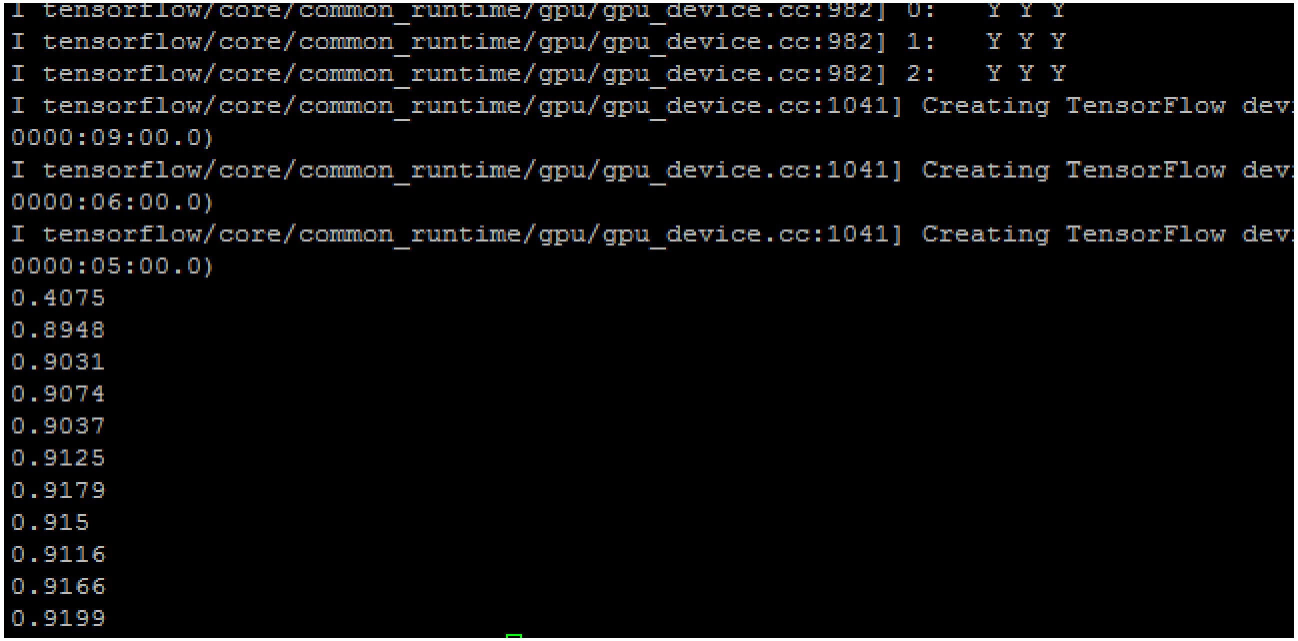
- x = tf.placeholder(tf.float32, [None, 784])
- W = tf.Variable(tf.zeros([784,10]))
- b = tf.Variable(tf.zeros([10]))
- y = tf.nn.softmax(tf.add(tf.matmul(x,W),b))
- y_ = tf.placeholder(tf.float32, [None,10])

- cross_entropy = tf.reduce_mean(-tf.reduce_sum(y_ * tf.log(y), reduction_indices=[1]))
- optimizer = tf.train.GradientDescentOptimizer(0.5)
- train_step = optimizer.minimize(cross_entropy)
- correct_prediction = tf.equal(tf.argmax(y,1), tf.argmax(y_,1))
- accuracy = tf.reduce_mean(tf.cast(correct_prediction,tf.float32))

예제코드-1

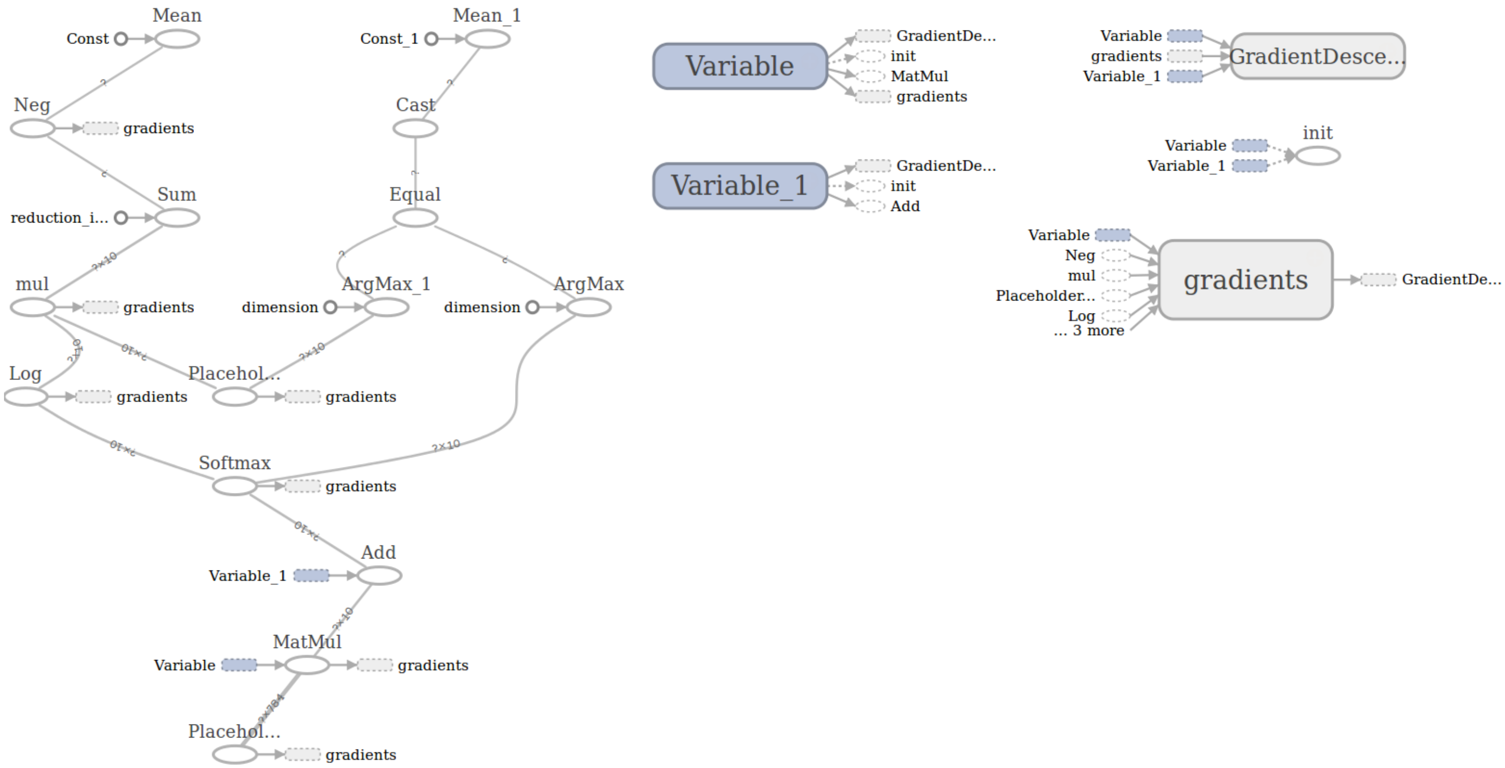
- init = tf.initialize_all_variables()
- sess = tf.Session()
- sess.run(init)
- for i in range(1000):
  - batch_xs, batch_ys = mnist.train.next_batch(100)
  - sess.run(train_step,feed_dict={x:batch_xs, y_:batch_ys})
  - if i%100 == 0:
    - print sess.run(accuracy,feed_dict={x:mnist.test.images,y_:mnist.test.labels})
- print(sess.run(accuracy,feed_dict={x:mnist.test.images,y_:mnist.test.labels}))

결과

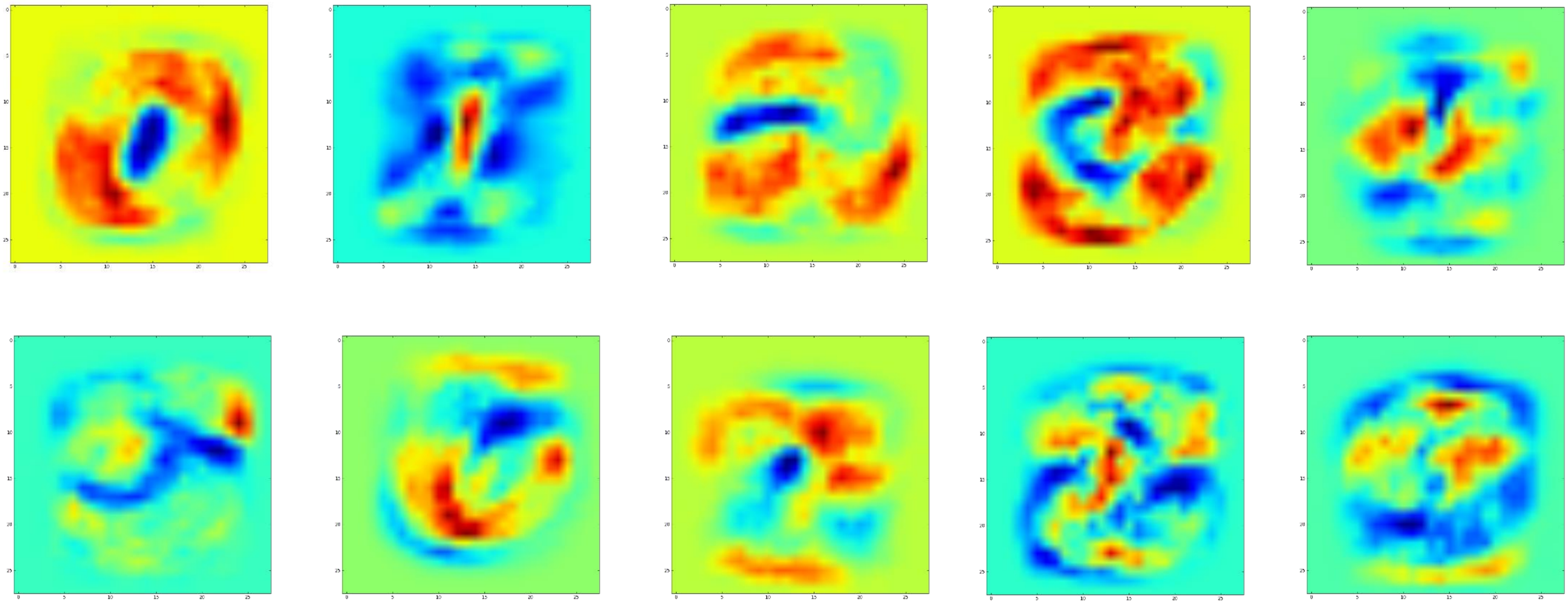


```
I tensorflow/core/common_runtime/gpu/gpu_device.cc:982] 0:   Y Y Y
I tensorflow/core/common_runtime/gpu/gpu_device.cc:982] 1:   Y Y Y
I tensorflow/core/common_runtime/gpu/gpu_device.cc:982] 2:   Y Y Y
I tensorflow/core/common_runtime/gpu/gpu_device.cc:1041] Creating TensorFlow dev
0000:09:00.0)
I tensorflow/core/common_runtime/gpu/gpu_device.cc:1041] Creating TensorFlow dev
0000:06:00.0)
I tensorflow/core/common_runtime/gpu/gpu_device.cc:1041] Creating TensorFlow dev
0000:05:00.0)
0.4075
0.8948
0.9031
0.9074
0.9037
0.9125
0.9179
0.915
0.9116
0.9166
0.9199
```

Tensorboard



결과해석



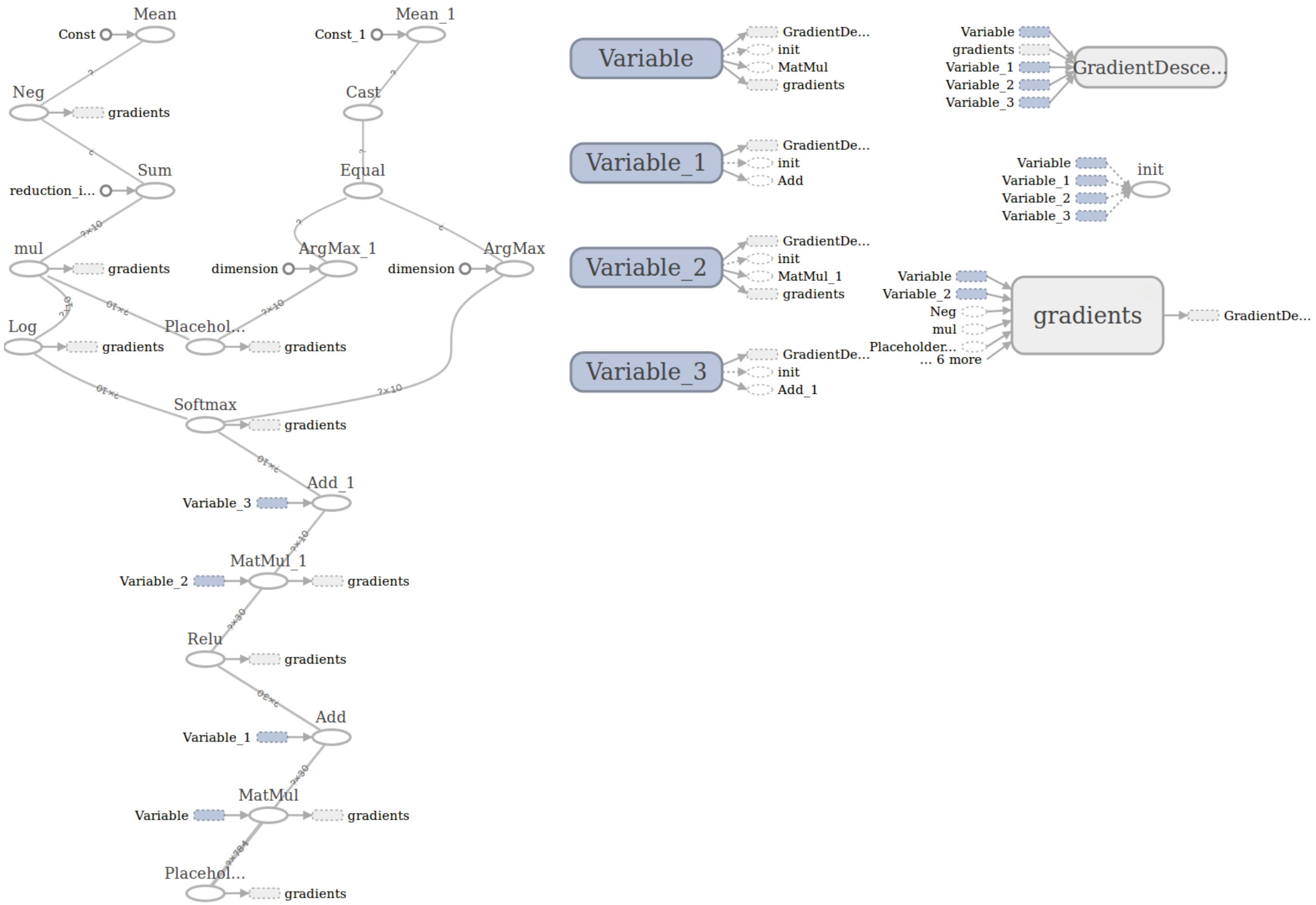
예제코드-2

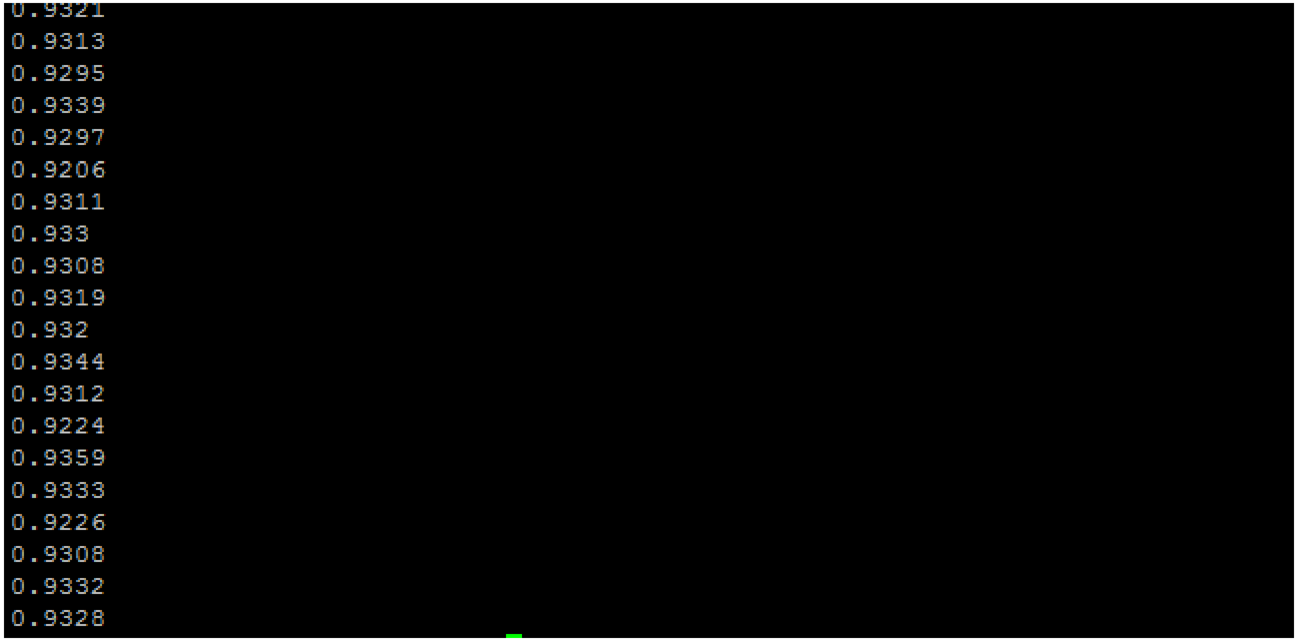
- …..

```
x = tf.placeholder(tf.float32, [None, 784])
W1 = tf.Variable(tf.zeros([784,30]))
b1 = tf.Variable(tf.ones([30]))
y1 = tf.nn.relu(tf.add(tf.matmul(x,W1),b1))

W2 = tf.Variable(tf.zeros([30,10]))
b2 = tf.Variable(tf.ones([10]))
y_ = tf.placeholder(tf.float32, [None,10])
y = tf.nn.softmax(tf.add(tf.matmul(y1,W2),b2))
```

- …..

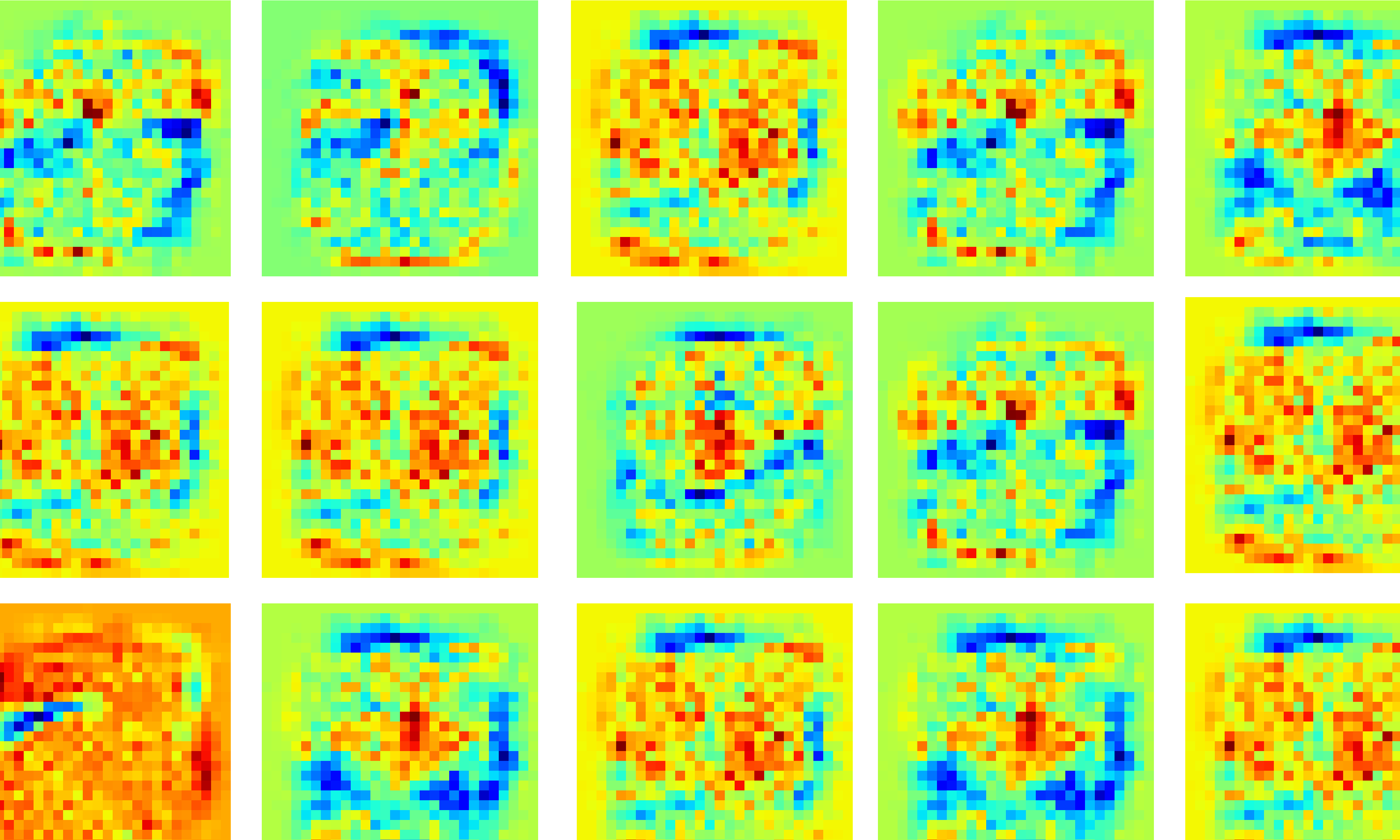
Tensorboard



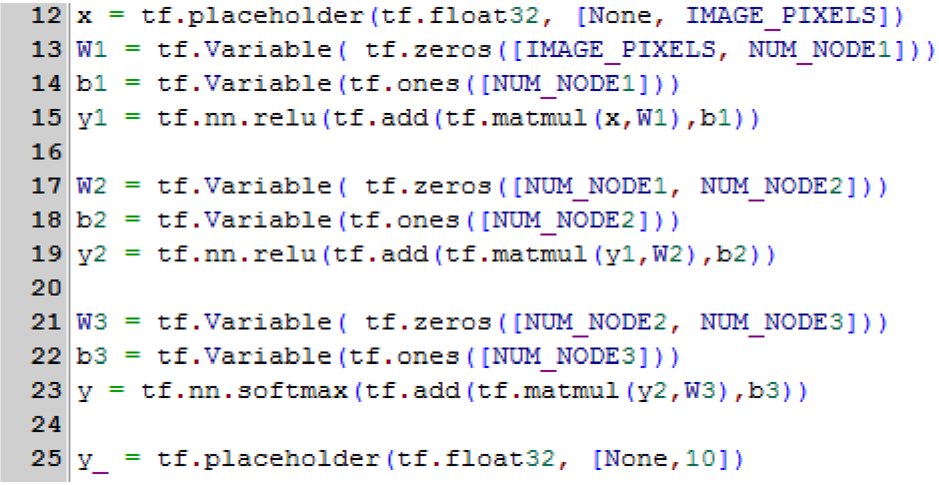


```
0.9321
0.9313
0.9295
0.9339
0.9297
0.9206
0.9311
0.933
0.9308
0.9319
0.932
0.9344
0.9312
0.9224
0.9359
0.9333
0.9226
0.9308
0.9332
0.9328
```

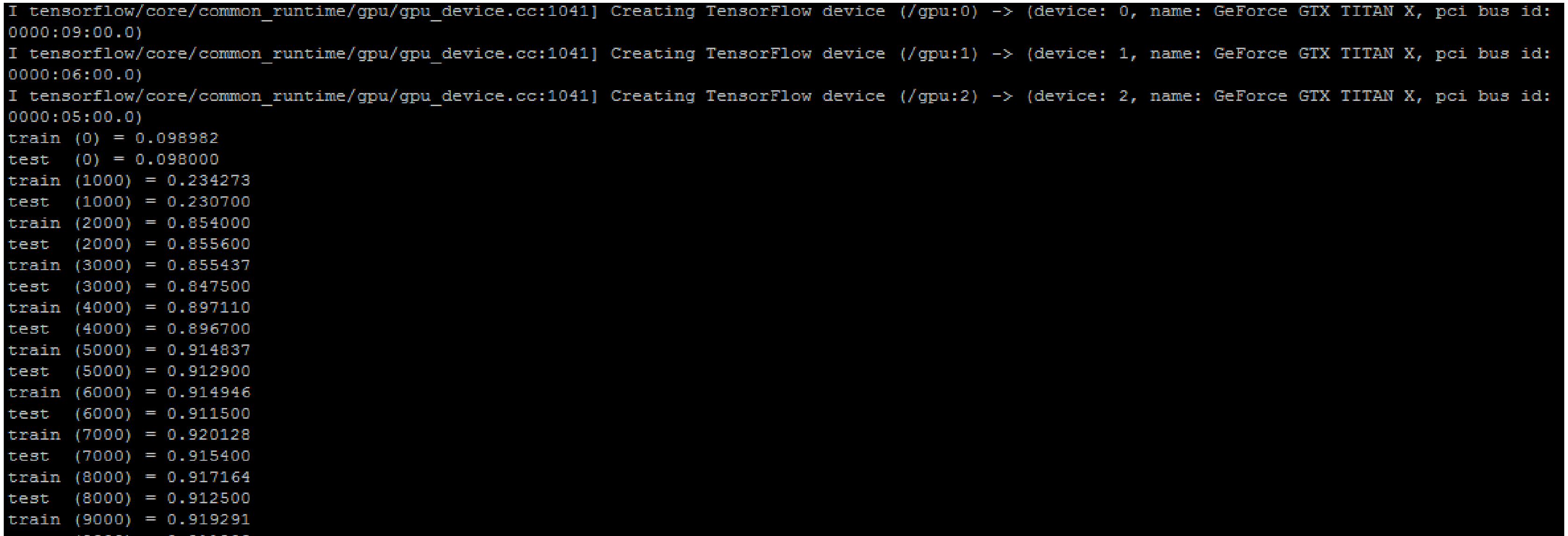
결과해석



초기화

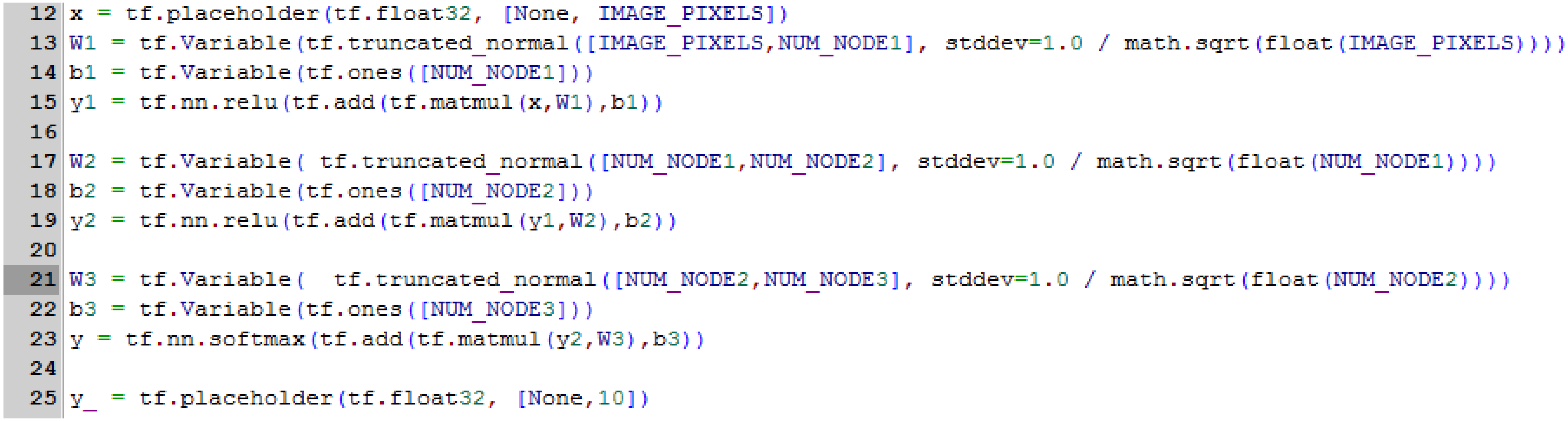


```
x = tf.placeholder(tf.float32, [None, IMAGE_PIXELS])
W1 = tf.Variable( tf.zeros([IMAGE_PIXELS, NUM_NODE1]))
b1 = tf.Variable(tf.ones([NUM_NODE1]))
y1 = tf.nn.relu(tf.add(tf.matmul(x,W1),b1))

W2 = tf.Variable( tf.zeros([NUM_NODE1, NUM_NODE2]))
b2 = tf.Variable(tf.ones([NUM_NODE2]))
y2 = tf.nn.relu(tf.add(tf.matmul(y1,W2),b2))

W3 = tf.Variable( tf.zeros([NUM_NODE2, NUM_NODE3]))
b3 = tf.Variable(tf.ones([NUM_NODE3]))
y = tf.nn.softmax(tf.add(tf.matmul(y2,W3),b3))

y_ = tf.placeholder(tf.float32, [None,10])
```

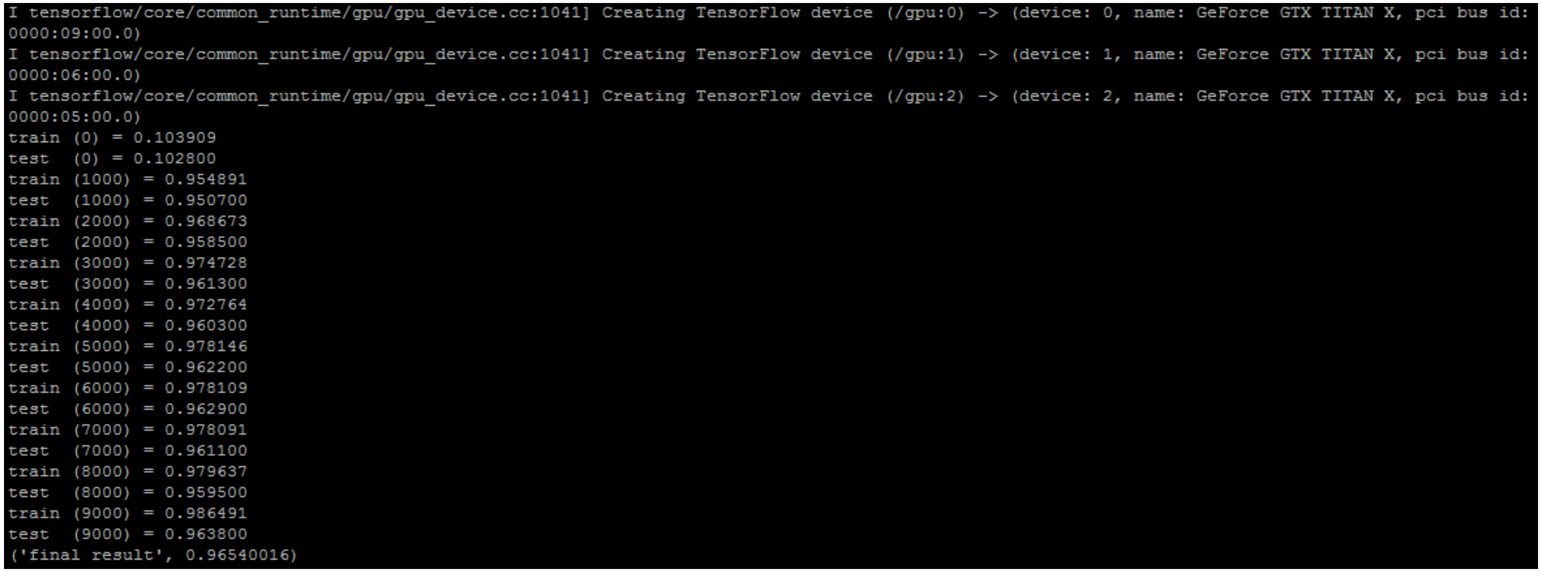


```
I tensorflow/core/common_runtime/gpu/gpu_device.cc:1041] Creating TensorFlow device (/gpu:0) -> (device: 0, name: GeForce GTX TITAN X, pci bus id:
0000:09:00.0)
I tensorflow/core/common_runtime/gpu/gpu_device.cc:1041] Creating TensorFlow device (/gpu:1) -> (device: 1, name: GeForce GTX TITAN X, pci bus id:
0000:06:00.0)
I tensorflow/core/common_runtime/gpu/gpu_device.cc:1041] Creating TensorFlow device (/gpu:2) -> (device: 2, name: GeForce GTX TITAN X, pci bus id:
0000:05:00.0)
train (0) = 0.098982
test  (0) = 0.098000
train (1000) = 0.234273
test  (1000) = 0.230700
train (2000) = 0.854000
test  (2000) = 0.855600
train (3000) = 0.855437
test  (3000) = 0.847500
train (4000) = 0.897110
test  (4000) = 0.896700
train (5000) = 0.914837
test  (5000) = 0.912900
train (6000) = 0.914946
test  (6000) = 0.911500
train (7000) = 0.920128
test  (7000) = 0.915400
train (8000) = 0.917164
test  (8000) = 0.912500
train (9000) = 0.919291
```

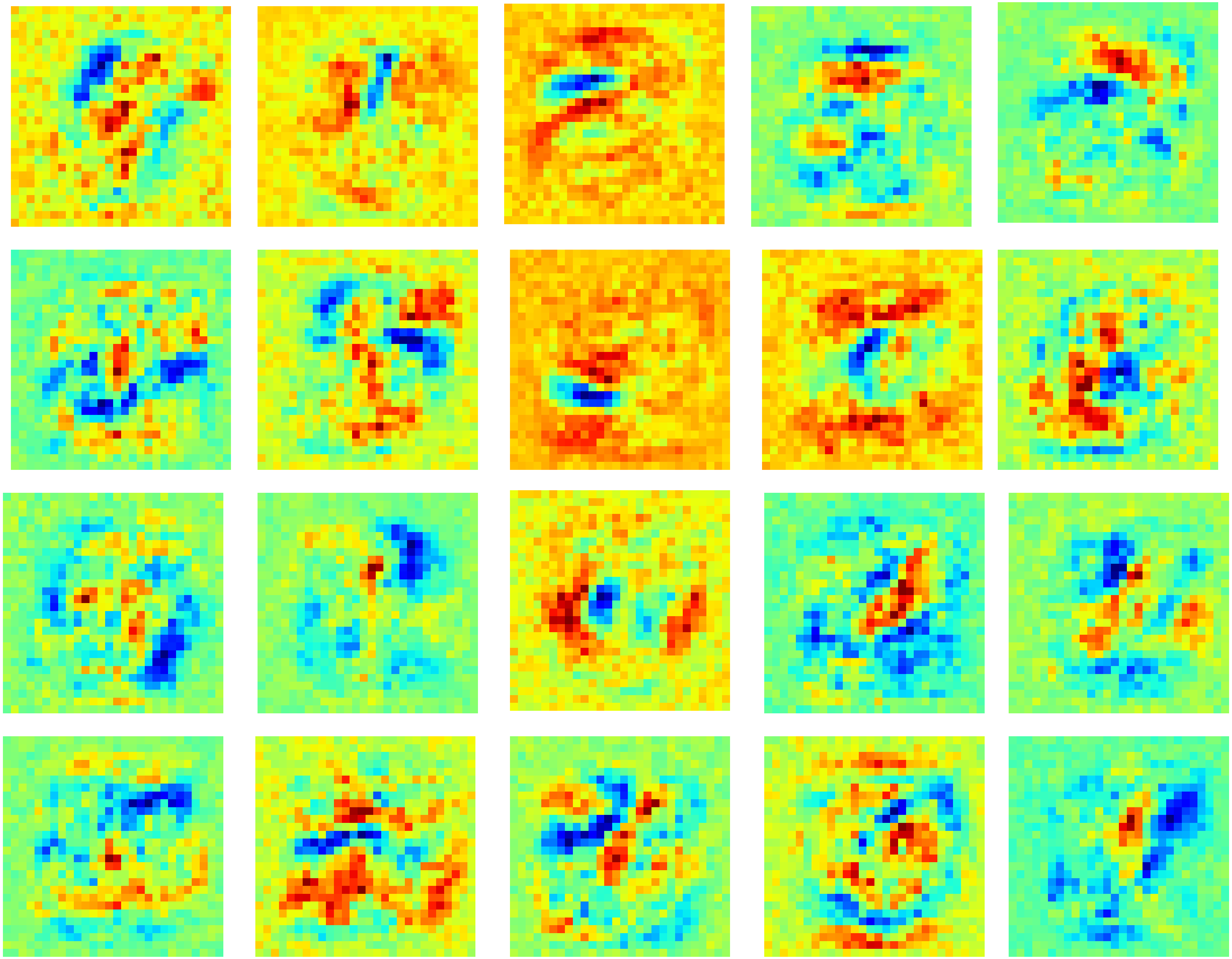
초기화 변경

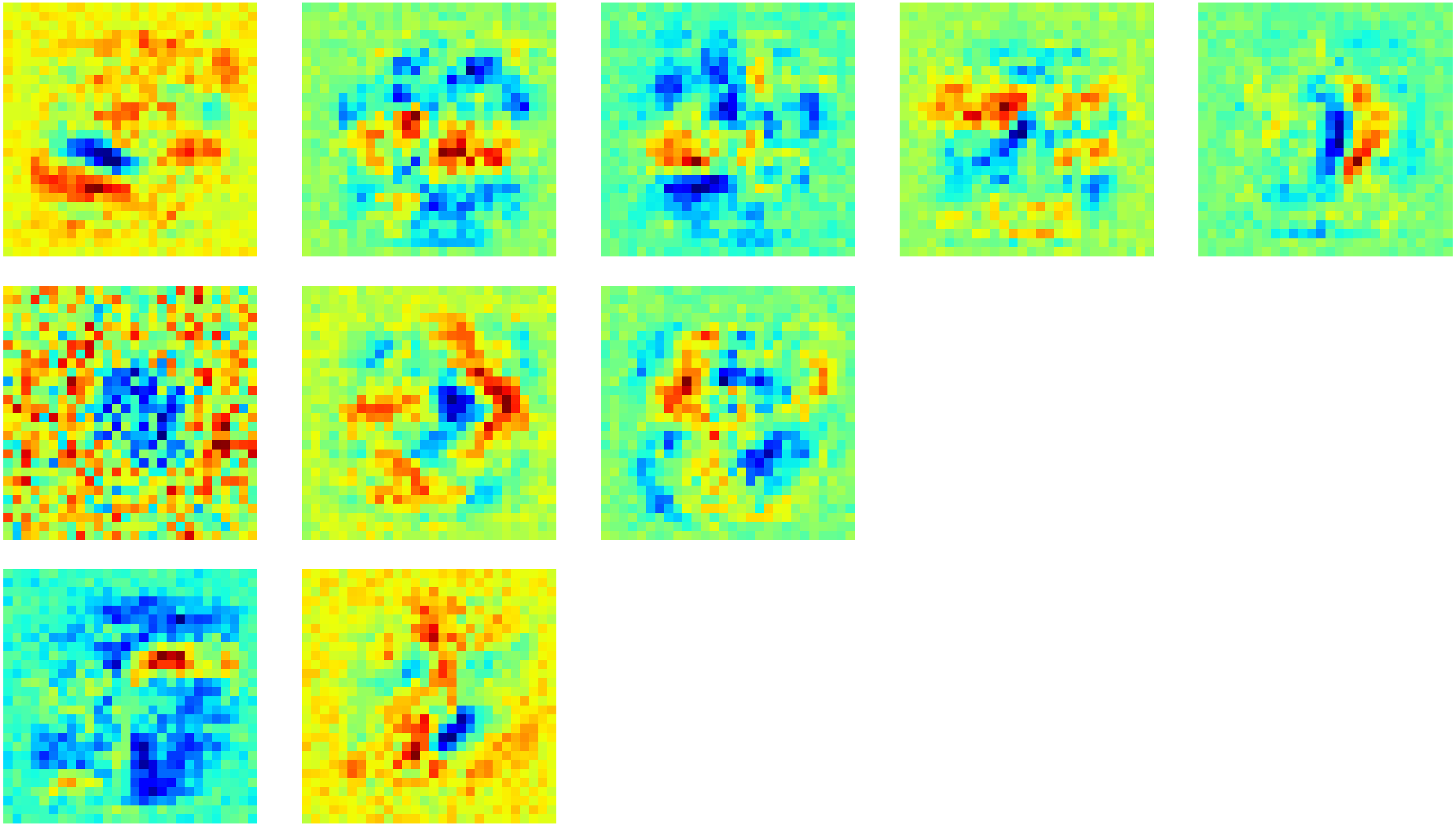


```
x = tf.placeholder(tf.float32, [None, IMAGE_PIXELS])
W1 = tf.Variable(tf.truncated_normal([IMAGE_PIXELS,NUM_NODE1], stddev=1.0 / math.sqrt(float(IMAGE_PIXELS))))
b1 = tf.Variable(tf.ones([NUM_NODE1]))
y1 = tf.nn.relu(tf.add(tf.matmul(x,W1),b1))

W2 = tf.Variable( tf.truncated_normal([NUM_NODE1,NUM_NODE2], stddev=1.0 / math.sqrt(float(NUM_NODE1))))
b2 = tf.Variable(tf.ones([NUM_NODE2]))
y2 = tf.nn.relu(tf.add(tf.matmul(y1,W2),b2))

W3 = tf.Variable(  tf.truncated_normal([NUM_NODE2,NUM_NODE3], stddev=1.0 / math.sqrt(float(NUM_NODE2))))
b3 = tf.Variable(tf.ones([NUM_NODE3]))
y = tf.nn.softmax(tf.add(tf.matmul(y2,W3),b3))

y_ = tf.placeholder(tf.float32, [None,10])
```

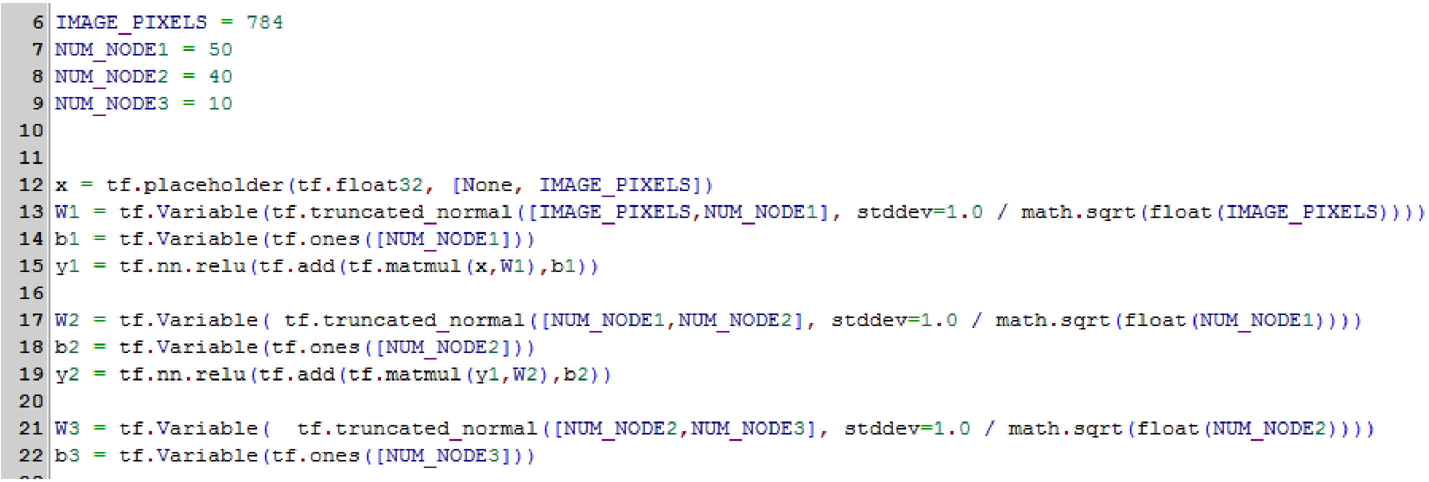


```
I tensorflow/core/common_runtime/gpu/gpu_device.cc:1041] Creating TensorFlow device (/gpu:0) -> (device: 0, name: GeForce GTX TITAN X, pci bus id:
0000:09:00.0)
I tensorflow/core/common_runtime/gpu/gpu_device.cc:1041] Creating TensorFlow device (/gpu:1) -> (device: 1, name: GeForce GTX TITAN X, pci bus id:
0000:06:00.0)
I tensorflow/core/common_runtime/gpu/gpu_device.cc:1041] Creating TensorFlow device (/gpu:2) -> (device: 2, name: GeForce GTX TITAN X, pci bus id:
0000:05:00.0)
train (0) = 0.103909
test  (0) = 0.102800
train (1000) = 0.954891
test  (1000) = 0.950700
train (2000) = 0.968673
test  (2000) = 0.958500
train (3000) = 0.974728
test  (3000) = 0.961300
train (4000) = 0.972764
test  (4000) = 0.960300
train (5000) = 0.978146
test  (5000) = 0.962200
train (6000) = 0.978109
test  (6000) = 0.962900
train (7000) = 0.978091
test  (7000) = 0.961100
train (8000) = 0.979637
test  (8000) = 0.959500
train (9000) = 0.986491
test  (9000) = 0.963800
('final result', 0.96540016)
```

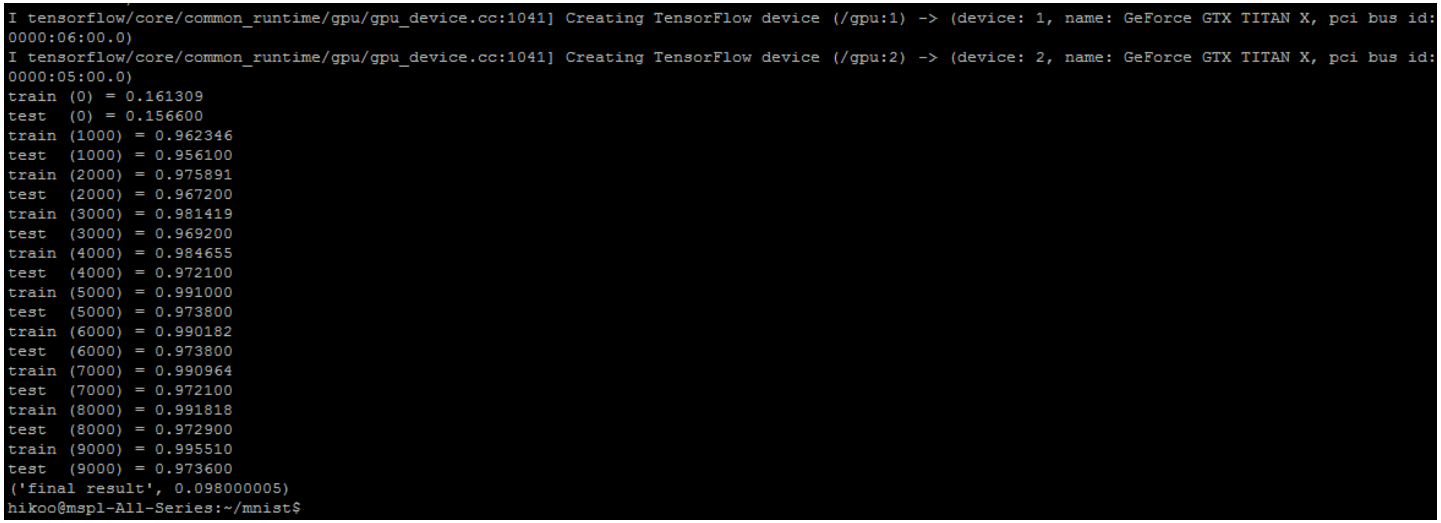




노드수 변경

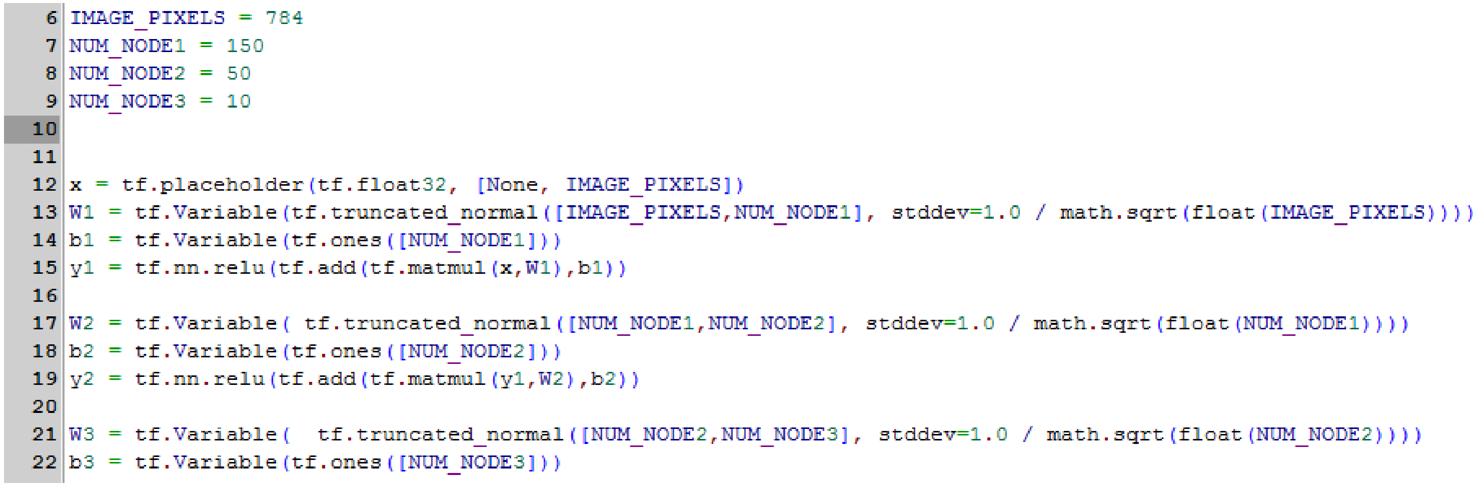


```
IMAGE_PIXELS = 784
NUM_NODE1 = 50
NUM_NODE2 = 40
NUM_NODE3 = 10


x = tf.placeholder(tf.float32, [None, IMAGE_PIXELS])
W1 = tf.Variable(tf.truncated_normal([IMAGE_PIXELS,NUM_NODE1], stddev=1.0 / math.sqrt(float(IMAGE_PIXELS))))
b1 = tf.Variable(tf.ones([NUM_NODE1]))
y1 = tf.nn.relu(tf.add(tf.matmul(x,W1),b1))

W2 = tf.Variable( tf.truncated_normal([NUM_NODE1,NUM_NODE2], stddev=1.0 / math.sqrt(float(NUM_NODE1))))
b2 = tf.Variable(tf.ones([NUM_NODE2]))
y2 = tf.nn.relu(tf.add(tf.matmul(y1,W2),b2))

W3 = tf.Variable(  tf.truncated_normal([NUM_NODE2,NUM_NODE3], stddev=1.0 / math.sqrt(float(NUM_NODE2))))
b3 = tf.Variable(tf.ones([NUM_NODE3]))
```

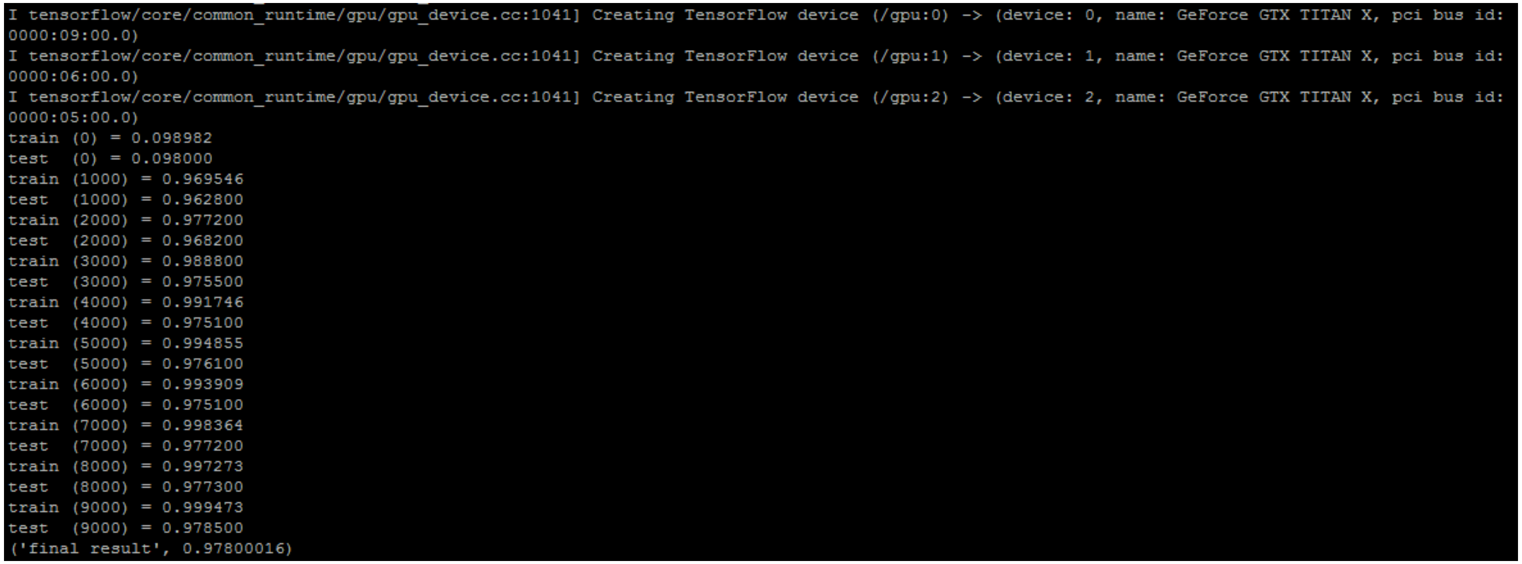


```
I tensorflow/core/common_runtime/gpu/gpu_device.cc:1041] Creating TensorFlow device (/gpu:1) -> (device: 1, name: GeForce GTX TITAN X, pci bus id:
0000:06:00.0)
I tensorflow/core/common_runtime/gpu/gpu_device.cc:1041] Creating TensorFlow device (/gpu:2) -> (device: 2, name: GeForce GTX TITAN X, pci bus id:
0000:05:00.0)
train (0) = 0.161309
test  (0) = 0.156600
train (1000) = 0.962346
test  (1000) = 0.956100
train (2000) = 0.975891
test  (2000) = 0.967200
train (3000) = 0.981419
test  (3000) = 0.969200
train (4000) = 0.984655
test  (4000) = 0.972100
train (5000) = 0.991000
test  (5000) = 0.973800
train (6000) = 0.990182
test  (6000) = 0.973800
train (7000) = 0.990964
test  (7000) = 0.972100
train (8000) = 0.991818
test  (8000) = 0.972900
train (9000) = 0.995510
test  (9000) = 0.973600
('final result', 0.098000005)
hikoo@mspl-All-Series:~/mnist$
```

노드수 변경



```
IMAGE_PIXELS = 784
NUM_NODE1 = 150
NUM_NODE2 = 50
NUM_NODE3 = 10


x = tf.placeholder(tf.float32, [None, IMAGE_PIXELS])
W1 = tf.Variable(tf.truncated_normal([IMAGE_PIXELS,NUM_NODE1], stddev=1.0 / math.sqrt(float(IMAGE_PIXELS))))
b1 = tf.Variable(tf.ones([NUM_NODE1]))
y1 = tf.nn.relu(tf.add(tf.matmul(x,W1),b1))

W2 = tf.Variable( tf.truncated_normal([NUM_NODE1,NUM_NODE2], stddev=1.0 / math.sqrt(float(NUM_NODE1))))
b2 = tf.Variable(tf.ones([NUM_NODE2]))
y2 = tf.nn.relu(tf.add(tf.matmul(y1,W2),b2))

W3 = tf.Variable(  tf.truncated_normal([NUM_NODE2,NUM_NODE3], stddev=1.0 / math.sqrt(float(NUM_NODE2))))
b3 = tf.Variable(tf.ones([NUM_NODE3]))
```



```
I tensorflow/core/common_runtime/gpu/gpu_device.cc:1041] Creating TensorFlow device (/gpu:0) -> (device: 0, name: GeForce GTX TITAN X, pci bus id:
0000:09:00.0)
I tensorflow/core/common_runtime/gpu/gpu_device.cc:1041] Creating TensorFlow device (/gpu:1) -> (device: 1, name: GeForce GTX TITAN X, pci bus id:
0000:06:00.0)
I tensorflow/core/common_runtime/gpu/gpu_device.cc:1041] Creating TensorFlow device (/gpu:2) -> (device: 2, name: GeForce GTX TITAN X, pci bus id:
0000:05:00.0)
train (0) = 0.098982
test  (0) = 0.098000
train (1000) = 0.969546
test  (1000) = 0.962800
train (2000) = 0.977200
test  (2000) = 0.968200
train (3000) = 0.988800
test  (3000) = 0.975500
train (4000) = 0.991746
test  (4000) = 0.975100
train (5000) = 0.994855
test  (5000) = 0.976100
train (6000) = 0.993909
test  (6000) = 0.975100
train (7000) = 0.998364
test  (7000) = 0.977200
train (8000) = 0.997273
test  (8000) = 0.977300
train (9000) = 0.999473
test  (9000) = 0.978500
('final result', 0.97800016)
```