TF v1.x를 이용한 신경망 실습

MNIST 실습

김 대 환

2022.

손글씨 숫자 분류

- **합성곱 신경망(Convolutional Neural Network – CNN)을 이용한 손글씨 숫자 인식**
  - 사람이 쓴 숫자는 완벽하지 않고 사람마다 다르므로 기계가 인식하기에는 어려운 문제다
  - 60,000 개의 학습 이미지와 10,000의 테스트 이미지 포함
  - 숫자 이미지는 28x28 회색조 픽셀 값을 가짐
- **주피터 노트북 실행**
  - 실습은 주피터 노트북에서 실행한다

```
$ jupyter notebook --ip=<IP_주소>  또는,
$ jupyter notebook --ip=localhost
```

손글씨 숫자 분류

- **주피터 노트북 연결**
  - 웹 브라우저 URL 필드에 노트북 서버 주소 또는 **localhost** 입력하고 "**New - tf1**" 항목선택

손글씨 숫자 분류

- **모듈을 import 하고 데이터 셋을 준비한다**
  - 노트북 셀에 다음 내용을 입력하고 "Shift + Enter" 키를 누른다

```
import tensorflow as tf
import numpy as np
import matplotlib.pyplot as plt
from tensorflow.examples.tutorials.mnist import input_data

np.random.seed(20160604)

mnist = input_data.read_data_sets("Mnistdata/", one_hot=True)
```

  - “./Mnistdata/" 폴더에 데이터가 로드 되었는지 확인한다
    - ✓ 폴더 퍼미션 문제가 발생하면 '/tmp’ 폴더를 이용하거나, 퍼미션 권한을 변경한다

손글씨 숫자 분류

- **학습을 위한 준비**

```
# 입력 데이터를 전달받을 placehold와 변수 정의
x = tf.placeholder(tf.float32, [None, 784])
w = tf.Variable(tf.zeros([784, 10]))
w0 = tf.Variable(tf.zeros([10]))

# 추론 함수와 활성화 함수 정의
f = tf.matmul(x, w) + w0
p = tf.nn.softmax(f)

# 손실 함수, 최적화 함수, 정확도 계산식 정의
t = tf.placeholder(tf.float32, [None, 10])
loss = -tf.reduce_sum(t * tf.log(p))
train_step = tf.train.AdamOptimizer().minimize(loss)

correct_prediction = tf.equal(tf.argmax(p, 1), tf.argmax(t, 1))
accuracy = tf.reduce_mean(tf.cast(correct_prediction, tf.float32))

sess = tf.InteractiveSession()
sess.run(tf.initialize_all_variables())sess.run(tf.initialize_all_variables())
```

손글씨 숫자 분류

- **학습**

```
# 파라미터 최적화를 2000회 반복
# 매회 훈련 세트에서 100개의 데이터를 추출해서 경사 하강법을 적용한다
i = 0
for _ in range(2000):
    i += 1
    batch_xs, batch_ts = mnist.train.next_batch(100)
    sess.run(train_step, feed_dict={x: batch_xs, t: batch_ts})
    if i % 100 == 0:
        loss_val, acc_val = sess.run([loss, accuracy],
            feed_dict={x:mnist.test.images, t: mnist.test.labels})
        print ('Step: %d, Loss: %f, Accuracy: %f'
               % (i, loss_val, acc_val))
```

손글씨 숫자 분류

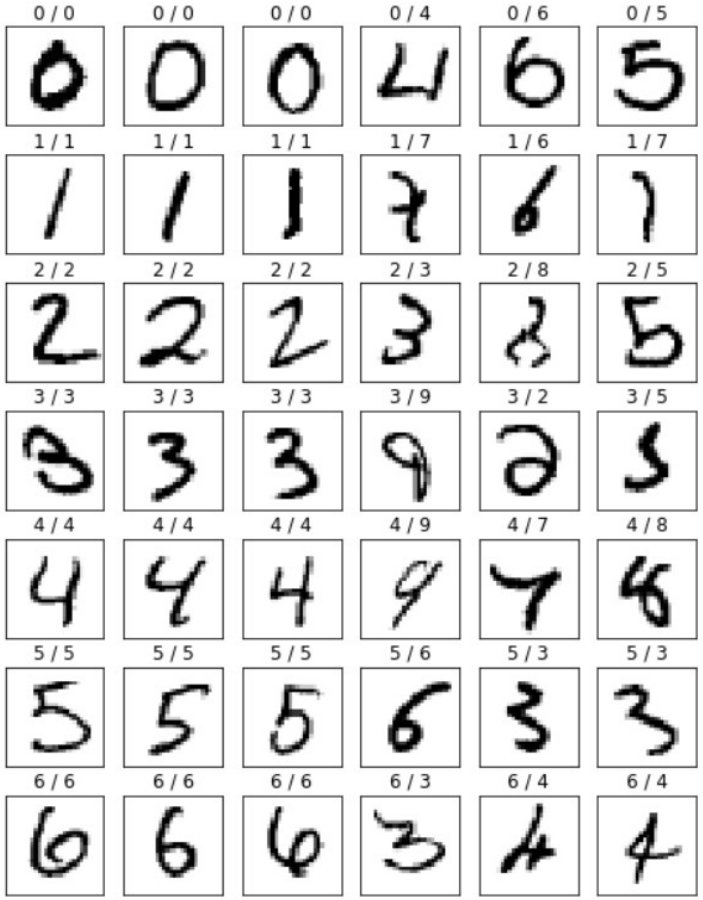
- 테스트 데이터를 이용한 학습 결과 확인

```
images, labels = mnist.test.images, mnist.test.labels
p_val = sess.run(p, feed_dict={x:images, t: labels})

fig = plt.figure(figsize=(8,15))
for i in range(10):
    c = 1
    for (image, label, pred) in zip(images, labels, p_val):
        prediction, actual = np.argmax(pred), np.argmax(label)
        if prediction != i:
            continue
        if (c < 4 and i == actual) or (c >= 4 and i != actual):
            subplot = fig.add_subplot(10,6,i*6+c)
            subplot.set_xticks([])
            subplot.set_yticks([])
            subplot.set_title('%d / %d' % (prediction, actual))
            subplot.imshow(image.reshape((28,28)), vmin=0, vmax=1,
                           cmap=plt.cm.gray_r, interpolation="nearest")
            c += 1
            if c > 6:
                break
```

손글씨 숫자 분류

- 테스트 데이터를 이용하여 학습 결과를 확인



손글씨 숫자 분류 – 네트워크 추가로 정확도 개선

- **네트워크를 추가하고 정확도가 개선되었는지 확인**
  - 노트북 셀에 다음 내용을 입력하고 "Shift + Enter" 키를 누른다

```
import tensorflow as tf
import numpy as np
import matplotlib.pyplot as plt

from tensorflow.examples.tutorials.mnist import input_data

# 모듈 import와 난수 시드 설정
tf.set_random_seed(777)  # reproducibility

# mnist dataset 준비
mnist = input_data.read_data_sets("Mnistdata", one_hot=True)
```

  - “./Mnistdata" 폴더에 데이터가 로드 되었는지 확인한다

손글씨 숫자 분류

- 학습을 위한 준비

```
# hyper parameters
learning_rate = 0.001
training_epochs = 15
batch_size = 100

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

# 신경망의 가중치와 편향 설정
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]))
```

손글씨 숫자 분류

- 라벨(정답)을 가진 훈련 데이터, 손실 함수, 최적화 함수 준비

```
# 추론 함수
hypothesis = tf.matmul(L2, W3) + b3

# 소프트맥스 활성화 함수
pred = tf.nn.softmax(hypothesis)


# 학습 데이터를 담을 placehold와 cost 함수
"""
loss = -tf.reduce_sum(Y * tf.log(pred))
optimizer = tf.train.AdamOptimizer().minimize(loss)
"""

# 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)
```

손글씨 숫자 분류

- **변수 초기화와 모델 학습**

```
# 변수 초기화
sess = tf.Session()
sess.run(tf.global_variables_initializer())
# sess = tf.InteractiveSession()
# sess.run(tf.initialize_all_variables())

# 모델 학습
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!')
```

손글씨 숫자 분류

- **학습 모델 테스트와 정확도 확인**
  - 출력되는 정확도가 개선된 것을 확인할 수 있다.

```
# Test model and check accuracy
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}))

# 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]}))
```

손글씨 숫자 분류

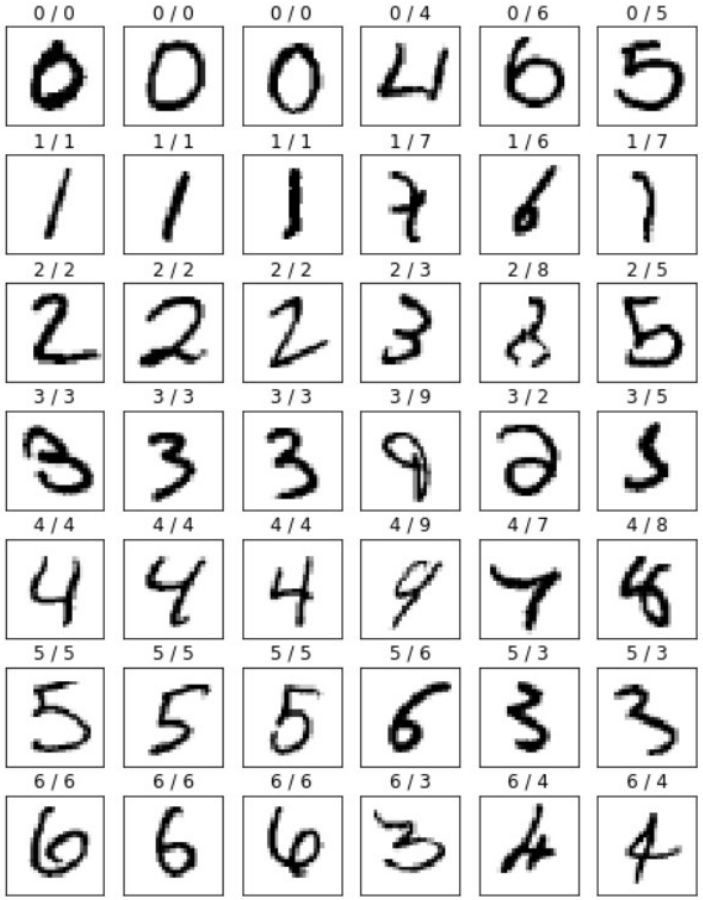
- 테스트 데이터를 이용하여 학습 결과를 확인

```
images, labels = mnist.test.images, mnist.test.labels
pred_val = sess.run(pred, feed_dict={X:images, Y: labels})

fig = plt.figure(figsize=(8,15))
for i in range(10):
    c = 1
    for (image, label, p) in zip(images, labels, pred_val):
        prediction, actual = np.argmax(p), np.argmax(label)
        if prediction != i:
            continue
        if (c < 4 and i == actual) or (c >= 4 and i != actual):
            subplot = fig.add_subplot(10,6,i*6+c)
            subplot.set_xticks([])
            subplot.set_yticks([])
            subplot.set_title('%d / %d' % (prediction, actual))
            subplot.imshow(image.reshape((28,28)), vmin=0, vmax=1,
                           cmap=plt.cm.gray_r, interpolation="nearest")
            c += 1
            if c > 6:
                break
```

손글씨 숫자 분류

- 테스트 데이터를 이용하여 학습 결과를 확인



손글씨 숫자 분류 – cnn 적용으로 정확도 개선

- **cnn을 적용하여 정확도 개선을 확인**

```
# 데이터 준비와 하이퍼 파라미터 설정
import tensorflow as tf
import random
# import matplotlib.pyplot as plt

from tensorflow.examples.tutorials.mnist import input_data

tf.set_random_seed(777)  # reproducibility

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


# hyper parameters
learning_rate = 0.001
training_epochs = 15
batch_size = 100
```

손글씨 숫자 분류 – cnn 적용으로 정확도 개선

- **cnn을 적용하여 정확도 개선을 확인**

```
# placehold 설정과 이미지 reshape
X = tf.placeholder(tf.float32, [None, 784])
X_img = tf.reshape(X, [-1, 28, 28, 1])   # img 28x28x1 (black/white)
Y = tf.placeholder(tf.float32, [None, 10])

# L1 ImgIn shape=(?, 28, 28, 1)
# 3x3x1 필터 32개 적용
W1 = tf.Variable(tf.random_normal([3, 3, 1, 32], stddev=0.01))
#    Conv     -> (?, 28, 28, 32)
#    Pool     -> (?, 14, 14, 32)

L1 = tf.nn.conv2d(X_img, W1, strides=[1, 1, 1, 1], padding='SAME')
L1 = tf.nn.relu(L1)
L1 = tf.nn.max_pool(L1, ksize=[1, 2, 2, 1],
                    strides=[1, 2, 2, 1], padding='SAME')
'''
Tensor("Conv2D:0", shape=(?, 28, 28, 32), dtype=float32)
Tensor("Relu:0", shape=(?, 28, 28, 32), dtype=float32)
Tensor("MaxPool:0", shape=(?, 14, 14, 32), dtype=float32)
'''
```

손글씨 숫자 분류 – cnn 적용으로 정확도 개선

- **cnn을 적용하여 정확도 개선을 확인**

```
# L2 ImgIn shape=(?, 14, 14, 32)
# 3x3x32 필터 64개 적용
W2 = tf.Variable(tf.random_normal([3, 3, 32, 64], stddev=0.01))
#    Conv      ->(?, 14, 14, 64)
#    Pool      ->(?, 7, 7, 64)

L2 = tf.nn.conv2d(L1, W2, strides=[1, 1, 1, 1], padding='SAME')
L2 = tf.nn.relu(L2)

# pooling 커널 사이즈 2x2
L2 = tf.nn.max_pool(L2, ksize=[1, 2, 2, 1],
                    strides=[1, 2, 2, 1], padding='SAME')
L2_flat = tf.reshape(L2, [-1, 7 * 7 * 64])

'''
Tensor("Conv2D_1:0", shape=(?, 14, 14, 64), dtype=float32)
Tensor("Relu_1:0", shape=(?, 14, 14, 64), dtype=float32)
Tensor("MaxPool_1:0", shape=(?, 7, 7, 64), dtype=float32)
Tensor("Reshape_1:0", shape=(?, 3136), dtype=float32)
'''
```

손글씨 숫자 분류 – cnn 적용으로 정확도 개선

- **cnn을 적용하여 정확도 개선을 확인**
  - Fully connected layer (or, Dense layer) 적용 – 앞선 예제의 머신 러닝 네트워크 (End-to-End)

```
# Final FC 7x7x64 inputs -> 10 outputs
W3 = tf.get_variable("W3", shape=[7 * 7 * 64, 10],
                     initializer=tf.contrib.layers.xavier_initializer())
b = tf.Variable(tf.random_normal([10]))
logits = tf.matmul(L2_flat, W3) + b

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

손글씨 숫자 분류 – cnn 적용으로 정확도 개선

- **cnn을 적용하여 정확도 개선을 확인**

```
# initialize
sess = tf.Session()
sess.run(tf.global_variables_initializer())

# train my model
print('Learning started. It takes sometime.')
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!')
```

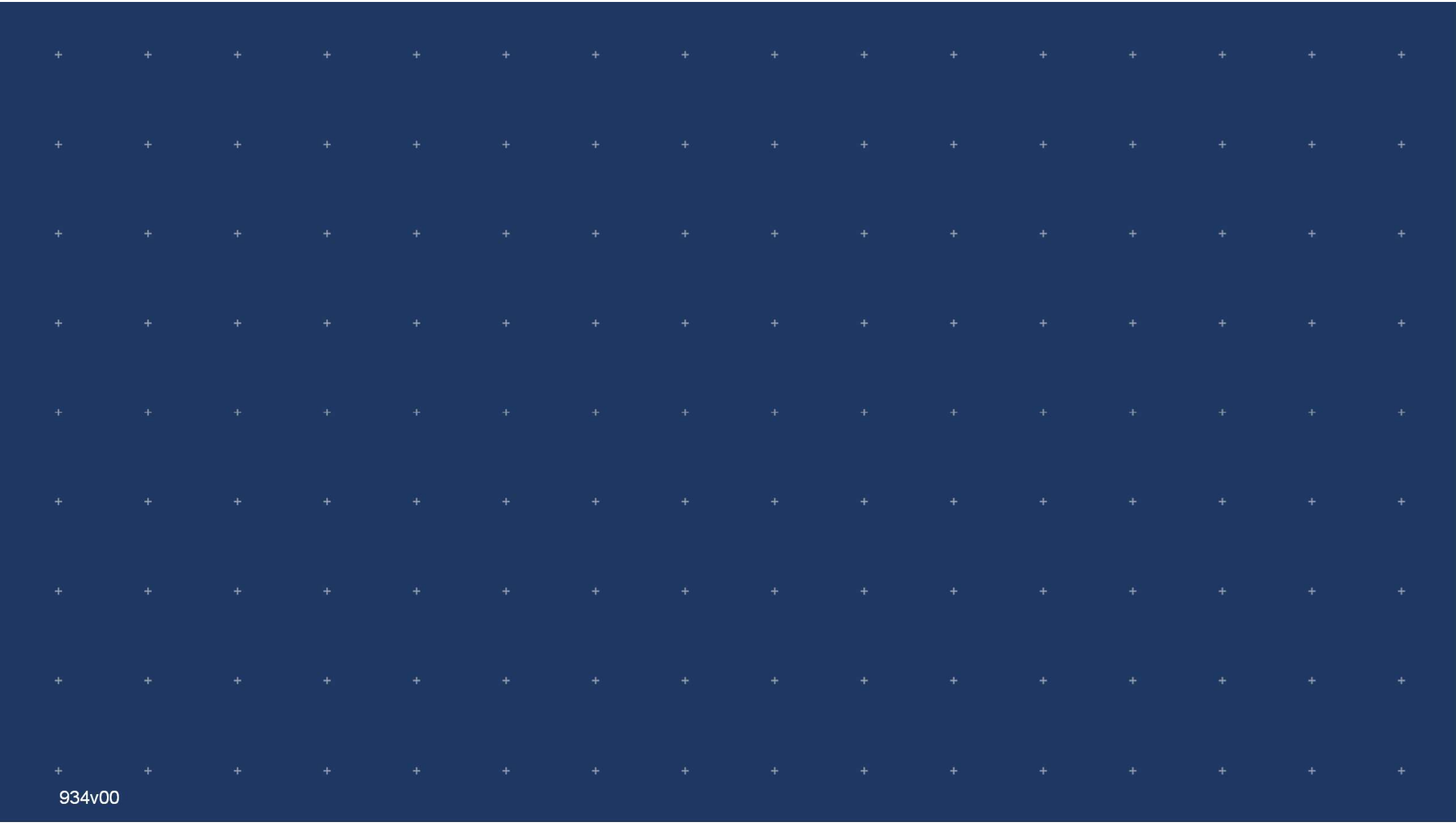
손글씨 숫자 분류 – cnn 적용으로 정확도 개선

- **cnn을 적용하여 정확도 개선을 확인**
  - 테스트 데이터를 이용하여 정확도 측정 및 샘플 이미지 하나를 선정하여 예측 결과 확인.

```
# Test model and check accuracy
correct_prediction = tf.equal(tf.argmax(logits, 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}))

# 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(logits, 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()
```


934v00