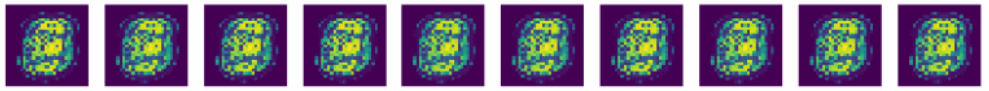
한림대학교 Summer Deep Learning Workshop

Image Classification

July 13, 2017

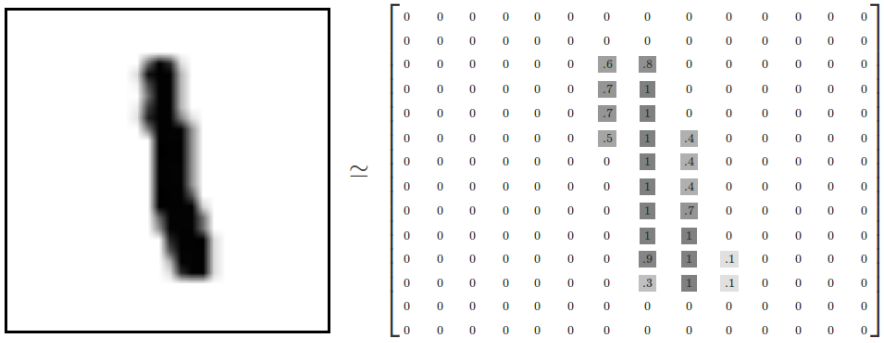
Seung-Chan Kim, Ph. D



- 1. 머신러닝 개론 및 주요 개념의 이해. Tensorflow 시스템 설치 및 환경설정 (7/4 화)
- 2. Tensorflow 에 익숙해지기 실습 및 Regression의 이해 (7/6 목)
- 3. Neural Network 이해 및 tensorflow 를 이용한 구현 (7/11 화)
- **4. 이미지 분류 이해 및 Tensorflow를 이용한 구현 (7/13 목)**

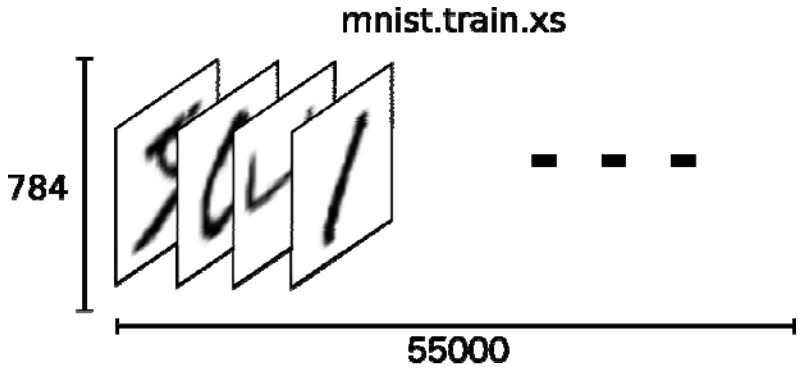
MNIST dataset

- MNIST Dataset
  - Handwritten digits, which has a training set of 55,000 examples (학습용 이미지) and a test set of 5,000 examples (검증용 이미지).
  - Includes 28 x 28 gray-scaled image and labels for each image.



MNIST dataset

- MNIST Dataset
  - 각 이미지는 28x28 크기를 가집니다. 이것을 일렬로 펼치면 784 (= 28x28) 차원의 벡터가 됨

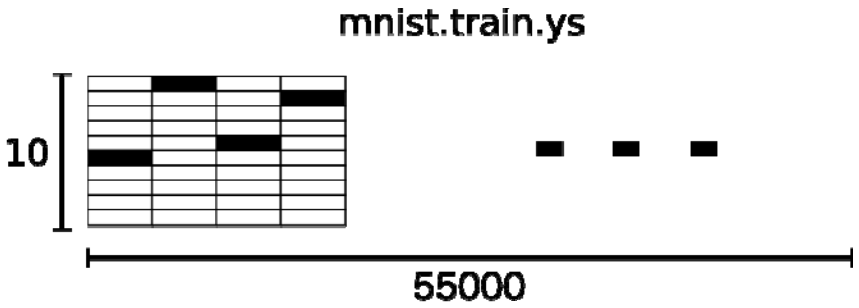



```
# input place holders
X = tf.placeholder(tf.float32, [None, 784])
Y = tf.placeholder(tf.float32, [None, 10])
```

https://gist.github.com/haje01/202ac276bace4b25dd3f

One-hot encoding

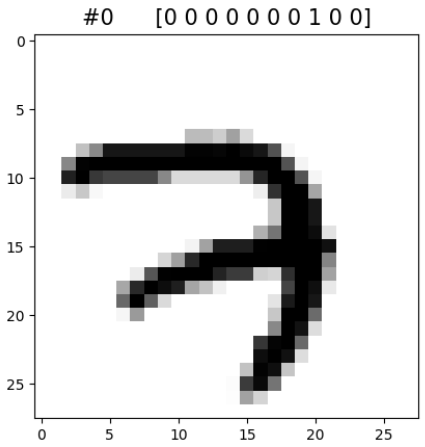
MNIST의 레이블은 0~9의 값이지만, 이것은 연속된 숫자가 아닌 카테고리 값

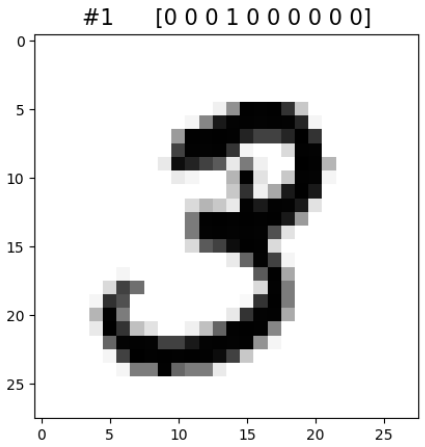


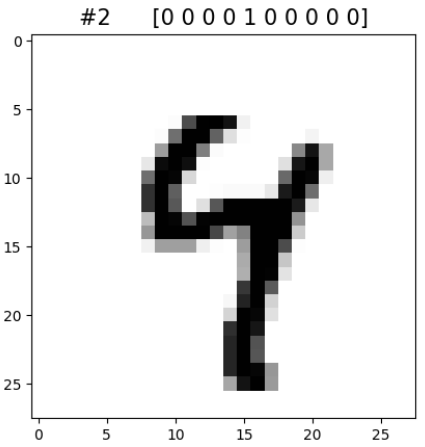
5는 [0, 0, 0, 0, 0, 1, 0, 0, 0, 0],
0은 [1, 0, 0, 0, 0, 0, 0, 0, 0, 0]

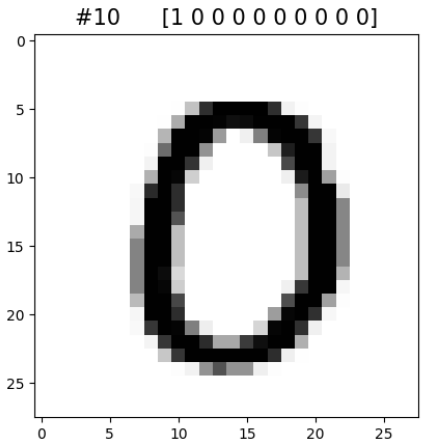
```
idx_to_test = 100
img     = mnist.train.images[idx_to_test]
label   = mnist.train.labels[idx_to_test]
```

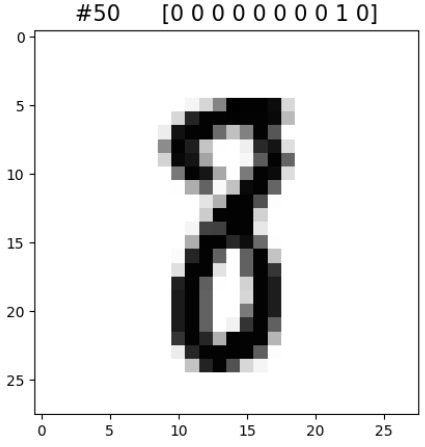
03-3-mnist-show.py 를 열어주세요

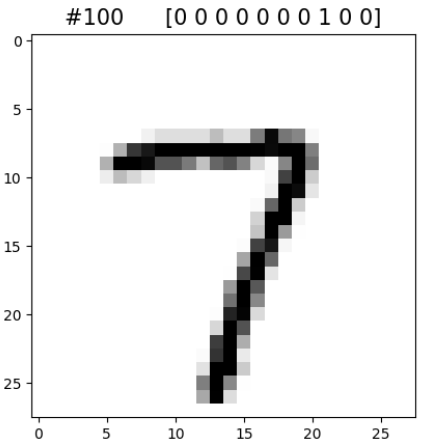


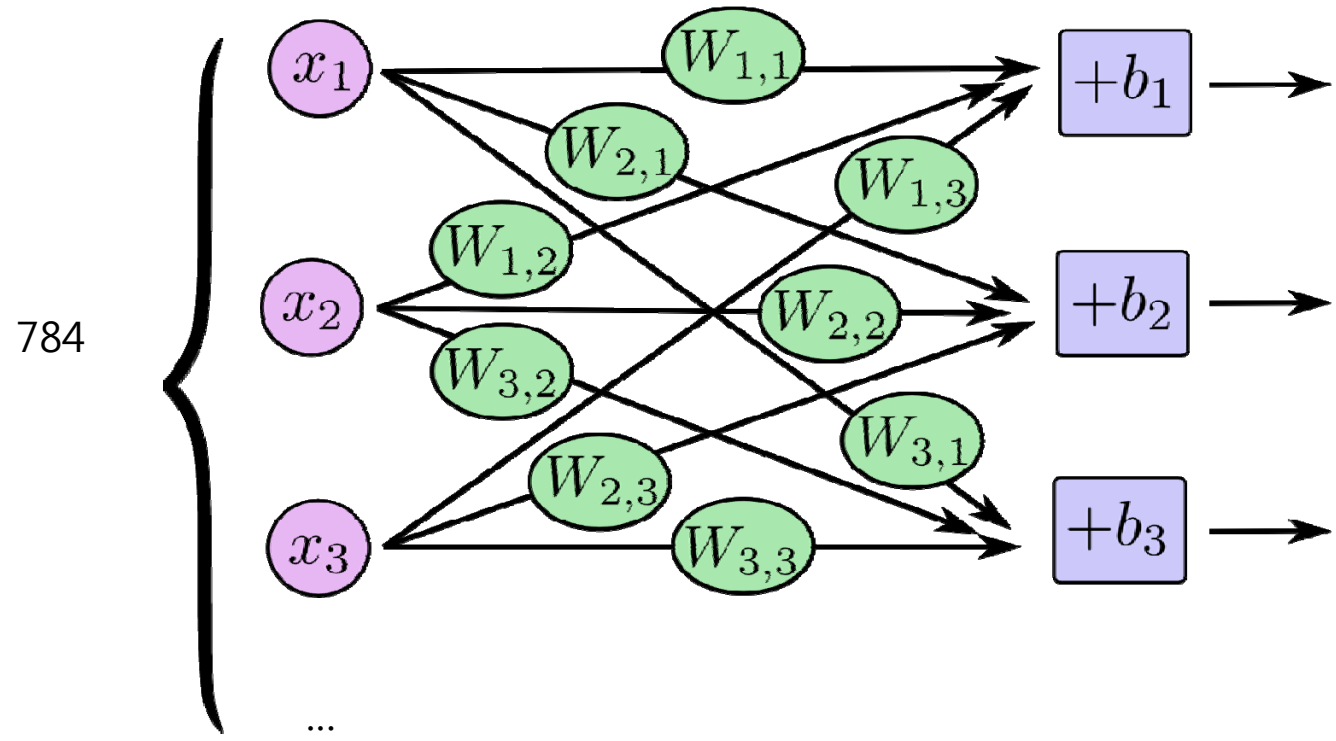











784
x_1
x_2
x_3
...
$W_{1,1}$
$W_{2,1}$
$W_{1,2}$
$W_{3,2}$
$W_{2,3}$
$W_{3,3}$
$W_{1,3}$
$W_{2,2}$
$W_{3,1}$
$+b_1$
$+b_2$
$+b_3$

Softmax ?!?

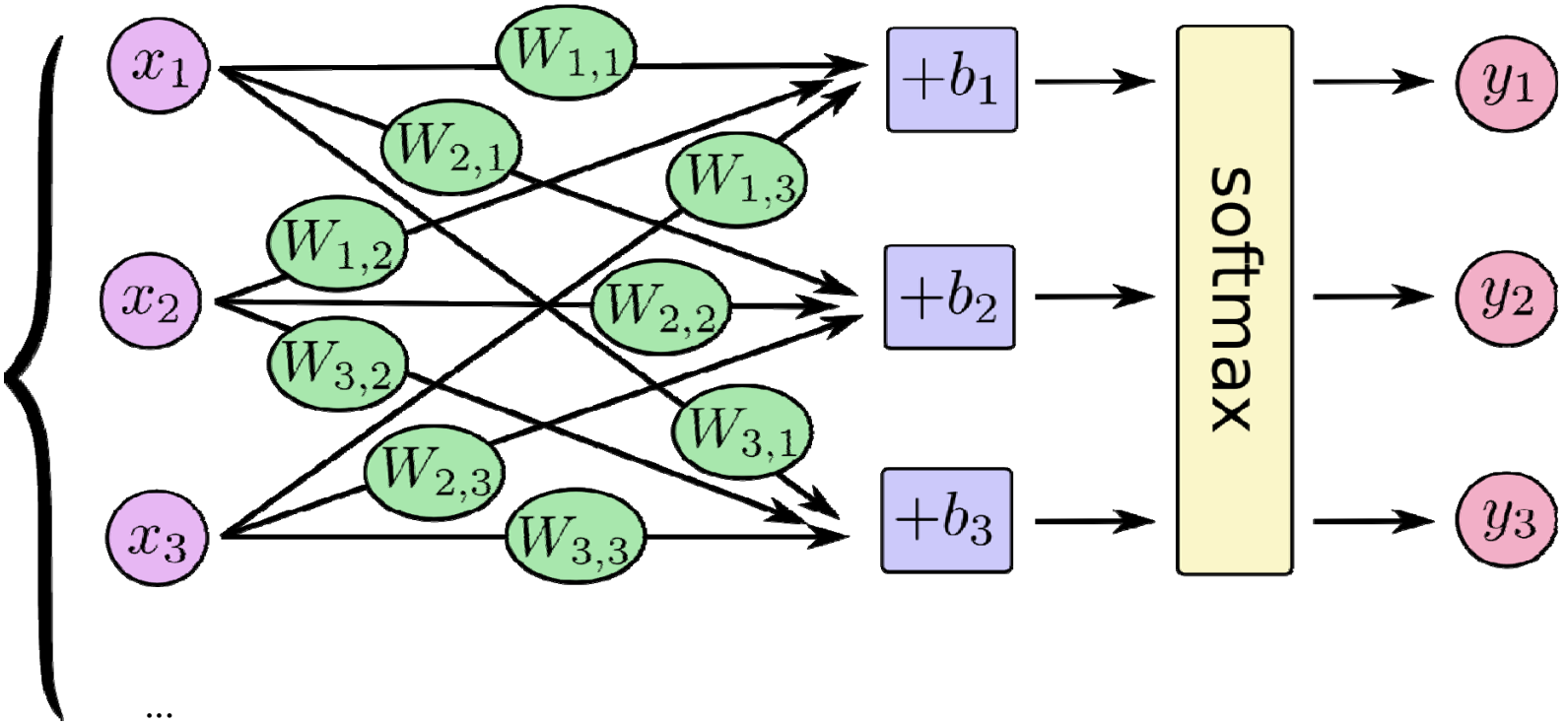
$$\text{evidence}_i = \sum_j W_{i,\,j} x_j + b_i$$

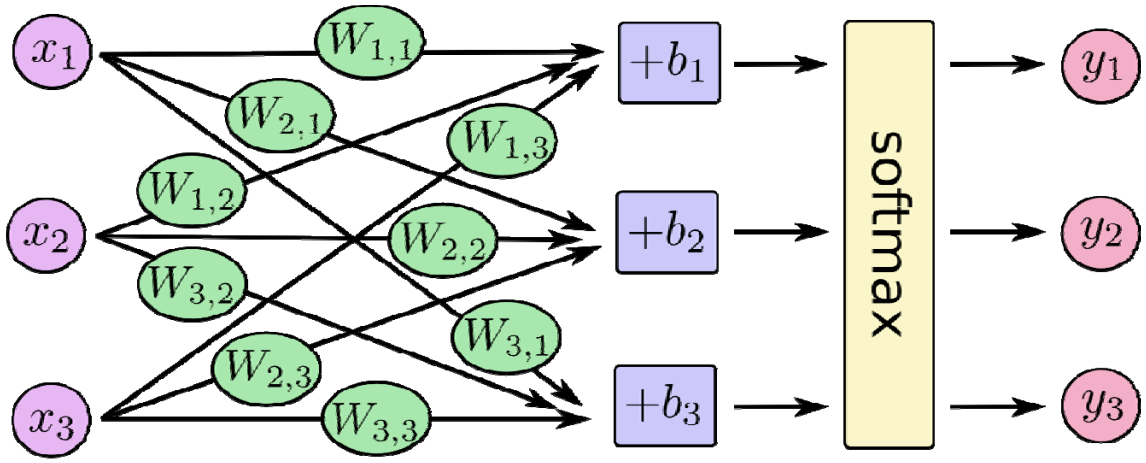
	Scoring Function	Unnormalized Probabilities	Normalized Probabilities
Dog	-3.44	0.0321	0.0006
Cat	1.16	3.1899	0.0596
Boat	-0.81	0.4449	0.0083
Airplane	3.91	49.8990	0.9315

$$\text{softmax}(x)_i = \frac{\exp(x_i)}{\sum_j \exp(x_j)}$$

- Softmax 함수의 출력은 0~1 사의 값이며, 출력들의 총 합은 1이 됨.
- 총합이 1이므로 출력을 '확률'로 해석할 수 있음
  -> 문제를 확률/통계적으로 대응할 수 있게 됨 !

http://www.pyimagesearch.com/2016/09/12/softmax-classifiers-explained/


784
x_1
x_2
x_3
...
$W_{1,1}$
$W_{2,1}$
$W_{1,2}$
$W_{3,2}$
$W_{2,3}$
$W_{3,3}$
$W_{1,3}$
$W_{2,2}$
$W_{3,1}$
$+b_1$
$+b_2$
$+b_3$
softmax
y_1
y_2
y_3



$$\begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix} = \text{softmax}\left(\begin{bmatrix} W_{1,1}x_1 + W_{1,2}x_2 + W_{1,3}x_3 + b_1 \\ W_{2,1}x_1 + W_{2,2}x_2 + W_{2,3}x_3 + b_2 \\ W_{3,1}x_1 + W_{3,2}x_2 + W_{3,3}x_3 + b_3 \end{bmatrix} \right)$$

$$y = \text{softmax}(Wx + b)$$

$$\begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix} = \text{softmax}\left(\begin{bmatrix} W_{1,1} & W_{1,2} & W_{1,3} \\ W_{2,1} & W_{2,2} & W_{2,3} \\ W_{3,1} & W_{3,2} & W_{3,3} \end{bmatrix} \cdot \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} + \begin{bmatrix} b_1 \\ b_2 \\ b_3 \end{bmatrix} \right)$$

```
y = tf.nn.softmax(tf.matmul(x, W) + b)
```

신경망 구성 (hypothesis)

X	W	Y
784	784*10	10

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

# weights & bias for nn layers
W = tf.Variable(tf.random_normal([784, 10]))
b = tf.Variable(tf.random_normal([10]))

hypothesis = tf.matmul(X, W) + b
```

Cross-Entropy

Cost 함수는 정보 이론의 크로스 엔트로피(Cross-Entropy) 방식으로 정의

$$H_{y'}(y) = -\sum_i y'_i \log(y_i)$$

y = our *predicted* probability distribution, (예측된 분포)
y′ = the true distribution (the one-hot vector with the digit labels)., 라벨

In some rough sense, the cross-entropy is measuring how inefficient our predictions are for describing the truth.

hypothesis

```
# define cost/loss
y = tf.nn.softmax(tf.matmul(x,W) + b)
cost = -tf.reduce_sum(y_*tf.log(y))
train_step = tf.train.GradientDescentOptimizer(0.01).minimize(cost )
```

Epoch & Batch

- batch_size = 100

```
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
```

계속 100개씩 뽑아서 55000개 될 때까지 학습 !

Q: 왜 하나씩 하면 안되나요?

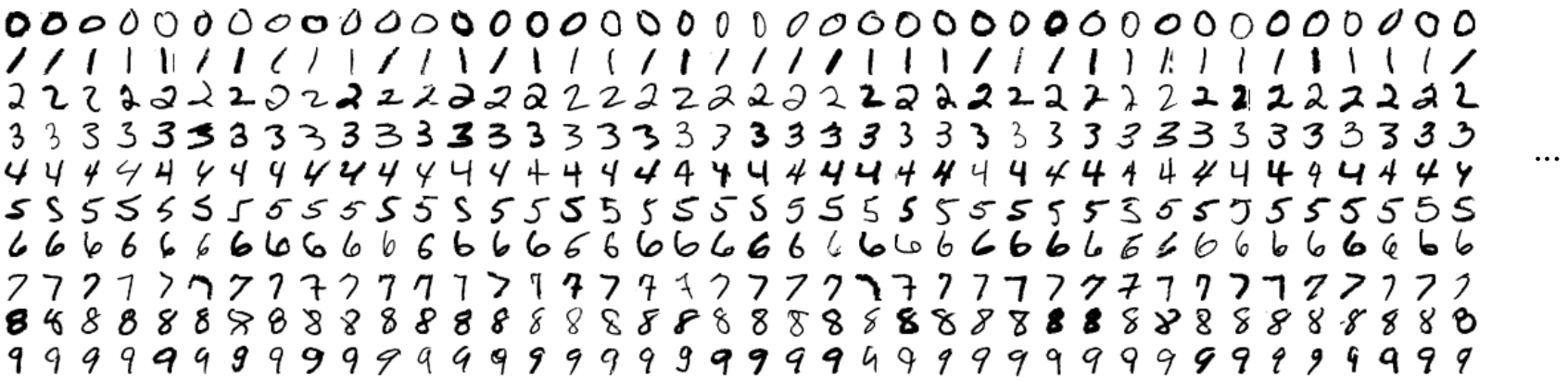
[참고] next_batch

```
def next_batch(self, batch_size, fake_data=False):
  """Return the next `batch_size` examples from this data set."""
  if fake_data:
    fake_image = [1] * 784
    if self.one_hot:
      fake_label = [1] + [0] * 9
    else:
      fake_label = 0
    return [fake_image for _ in xrange(batch_size)], [
        fake_label for _ in xrange(batch_size)
    ]
  start = self._index_in_epoch
  self._index_in_epoch += batch_size
  if self._index_in_epoch > self._num_examples:
    # Finished epoch
    self._epochs_completed += 1
    # Shuffle the data
    perm = numpy.arange(self._num_examples)
    numpy.random.shuffle(perm)
    self._images = self._images[perm]
    self._labels = self._labels[perm]
    # Start next epoch
    start = 0
    self._index_in_epoch = batch_size
    assert batch_size <= self._num_examples
  end = self._index_in_epoch
  return self._images[start:end], self._labels[start:end]
```

한 epoch가 끝나면 shuffle 하고 특정갯 수 뽑아줌

The images are returned as a 2-D NumPy array of size [batch_size, 784]

https://stackoverflow.com/a/41454722



pip install Pillow

03-4-mnist-softmax.py를 열어주세요

```
('mnist.train.num_examples = ', 55000)

('Epoch:', '0001', 'cost =', '5.745170846')
('Epoch:', '0002', 'cost =', '1.780056692')
('Epoch:', '0003', 'cost =', '1.122778638')
…
('Epoch:', '0049', 'cost =', '0.270640435')
('Epoch:', '0050', 'cost =', '0.269054373')
Learning Finished!

('Accuracy:', 0.91940016)
```

91.94 %

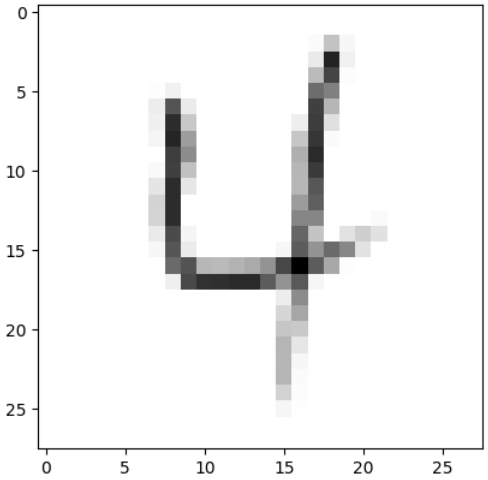
tf.argmax함수는 텐서 내의 지정된 축에서 가장 높은 값의 인덱스를 반환.

```
correct_prediction =
tf.equal(tf.argmax(hypothesis, 1), tf.argmax(Y, 1))

accuracy = tf.reduce_mean(tf.cast(correct_prediction,
tf.float32))

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

Mnist with a custom data

본인의 이미지를 만들어 넣어보세요

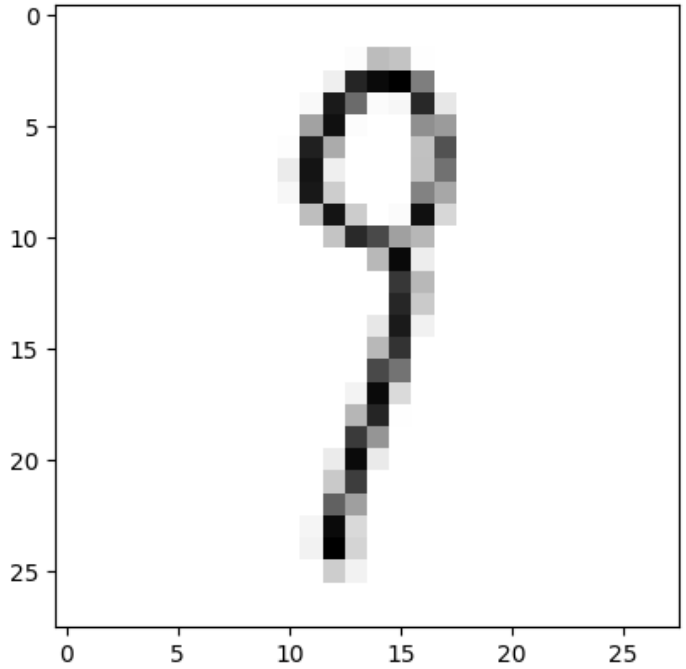
28x28




```
x = img.imread("data/4_28x.png")
x = x.astype(float)
x1 = np.array(x)
x1 = x1/255.0
x1 = 1.0 - x1
x1_flat = x1.reshape(1,784 )

print("Label: ", 4)
print("Prediction: ", sess.run(
    tf.argmax(hypothesis, 1), feed_dict={X:x1_flat}))
fig3 = plt.figure()
plt.imshow(x1_flat.
           reshape(28, 28), cmap='Greys',
interpolation='nearest')

plt.show()
```

('Label: ', 4)
('Prediction: ', array([4]))

28x28



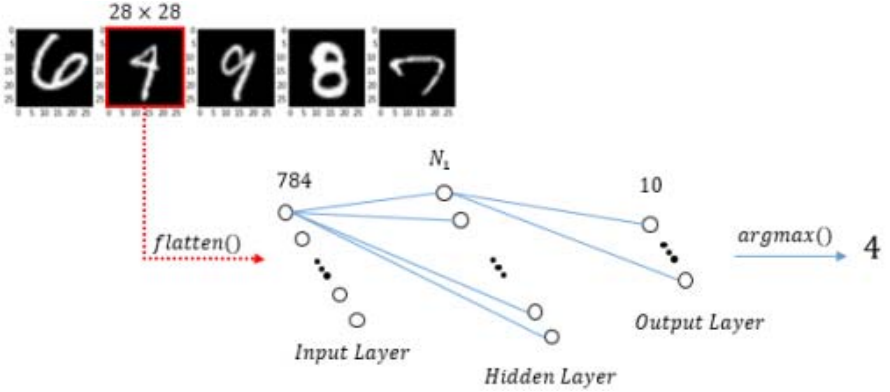
('Label: ', 9)
('Prediction: ', array([3]))

☹

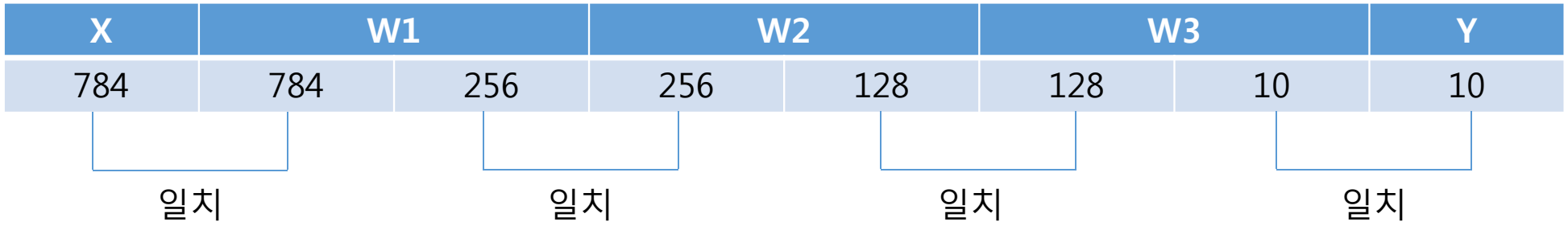
더 깊게.

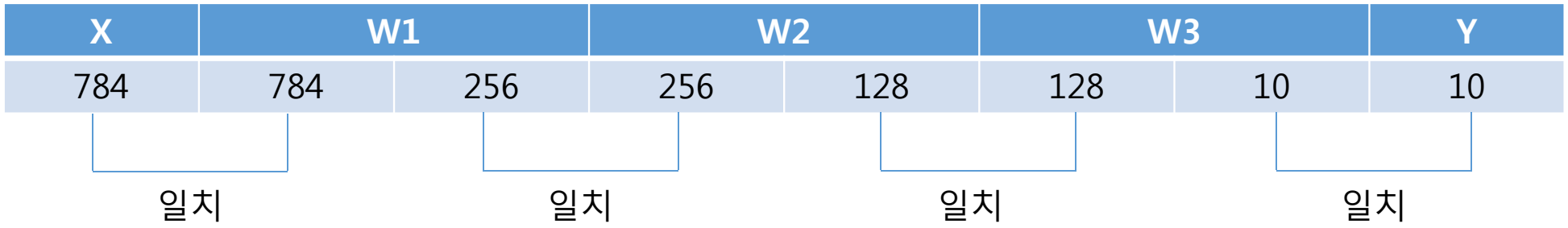
- 92% 에서 더 올릴 수 있을까?

```
# 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
```

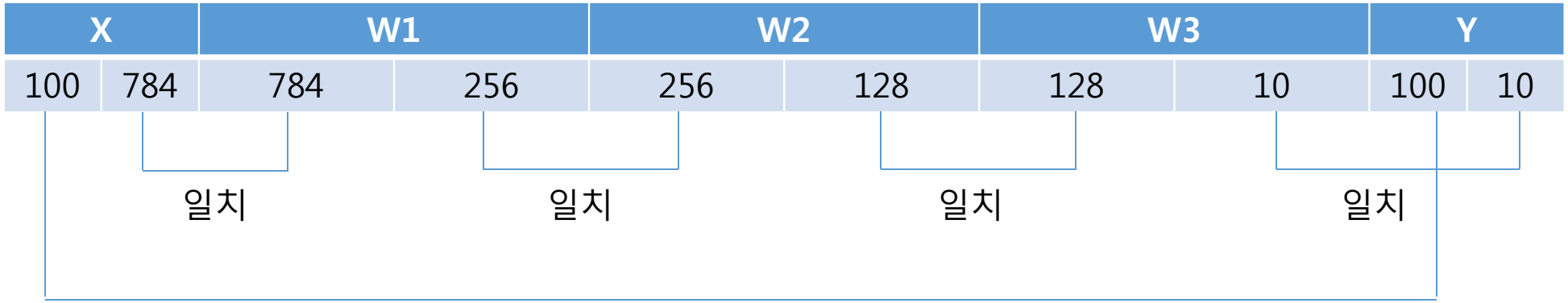


```
# input place holders
X = tf.placeholder(tf.float32, [None, 784])
Y = tf.placeholder(tf.float32, [None, 10])

# 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, 128]))
b2 = tf.Variable(tf.random_normal([128]))
L2 = tf.nn.relu(tf.matmul(L1, W2) + b2)

W3 = tf.Variable(tf.random_normal([128, 10]))
b3 = tf.Variable(tf.random_normal([10]))
hypothesis = tf.matmul(L2, W3) + b3
```



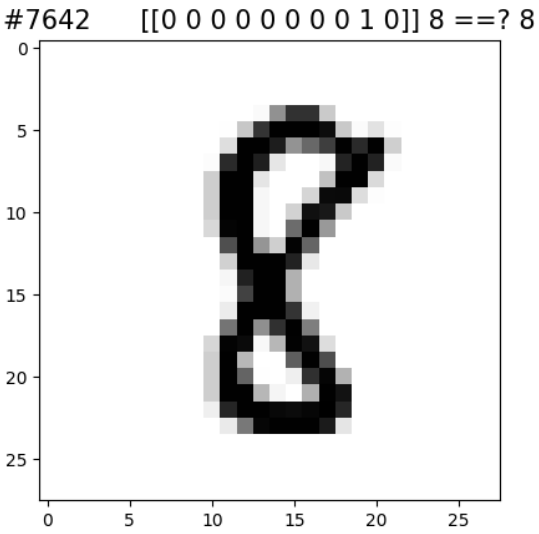


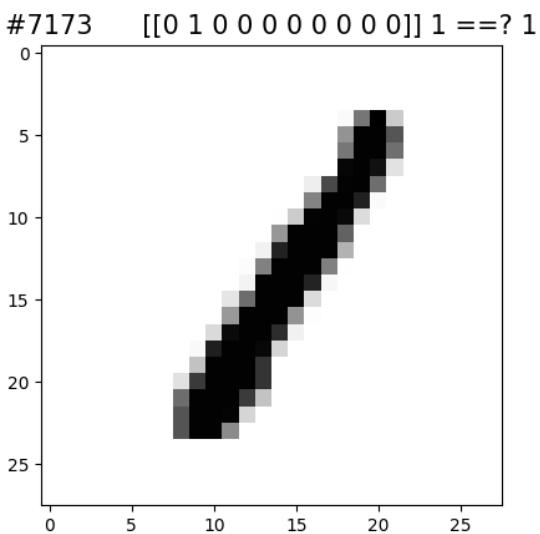
배치처리를 위한 형상

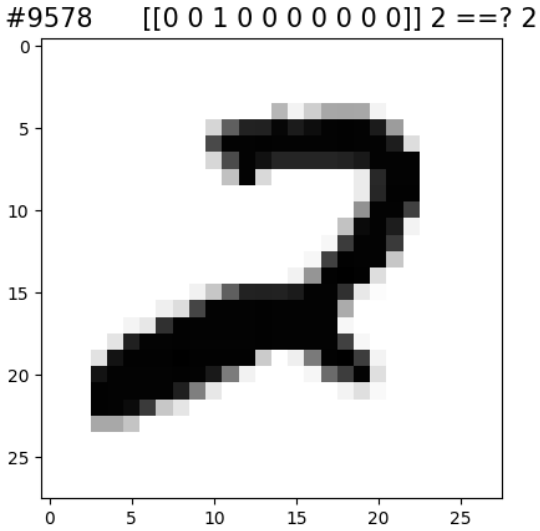


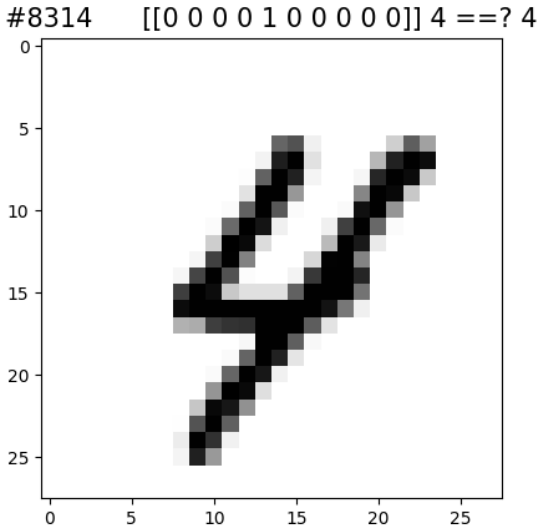
입력 형상 : 100×784

출력 형상 : 100×10







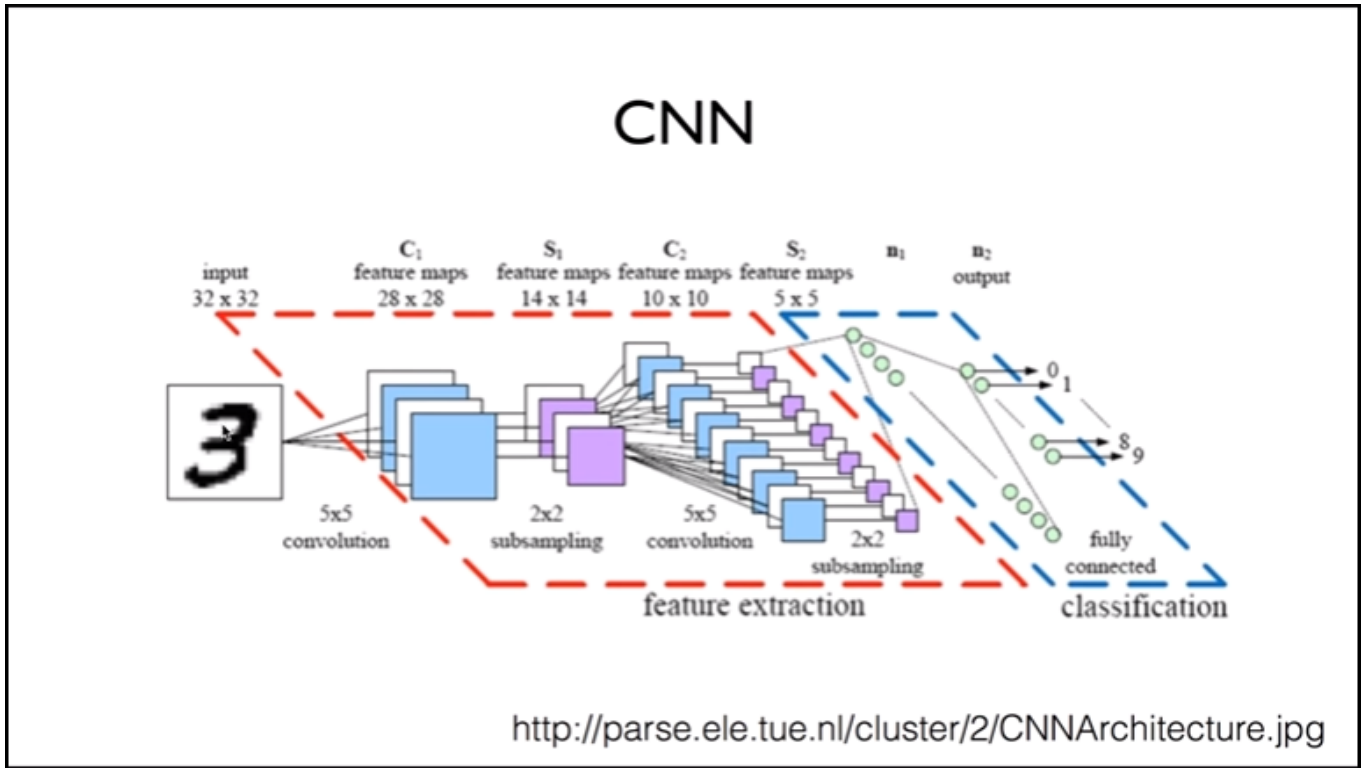


03-5-mnist-deeper.py 를 열어주세요

Random pick 결과들

What’s next ? MNIST 99%

- CNN : Convolutional Neural Network



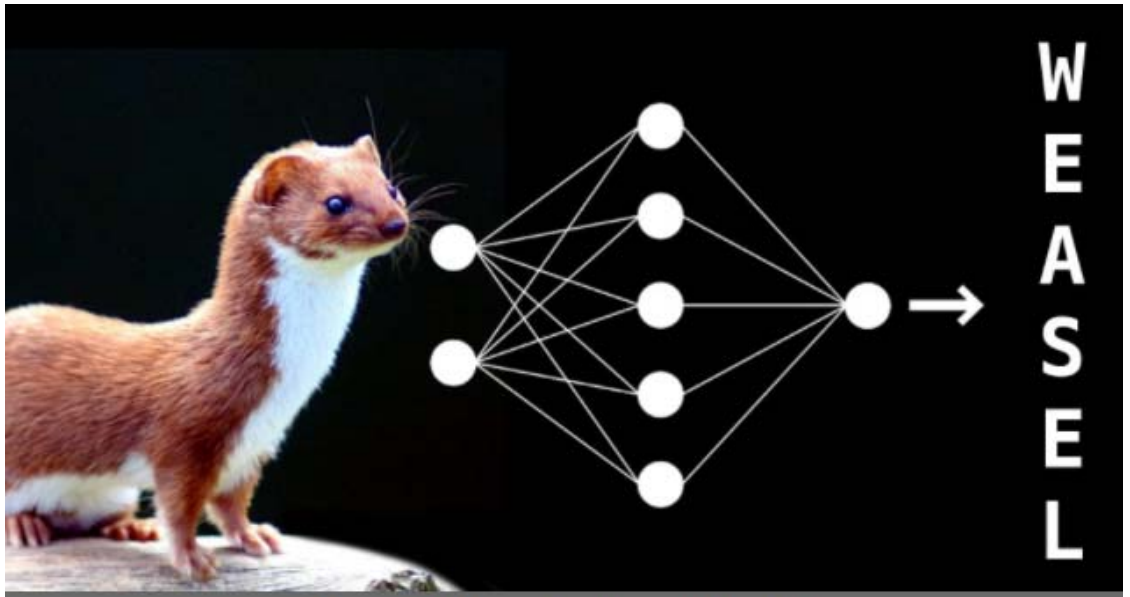
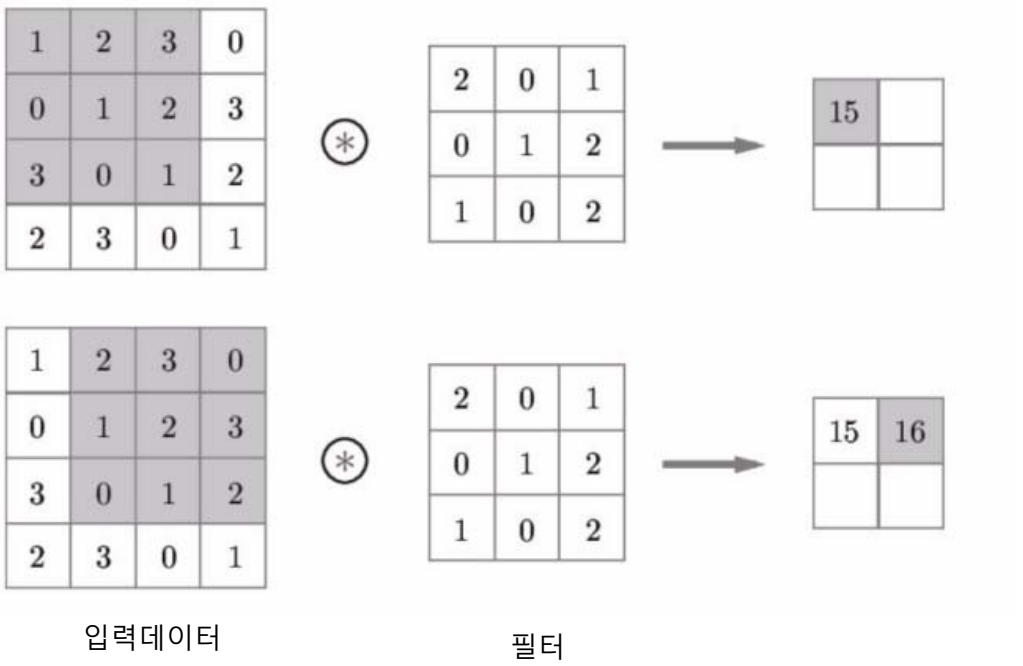
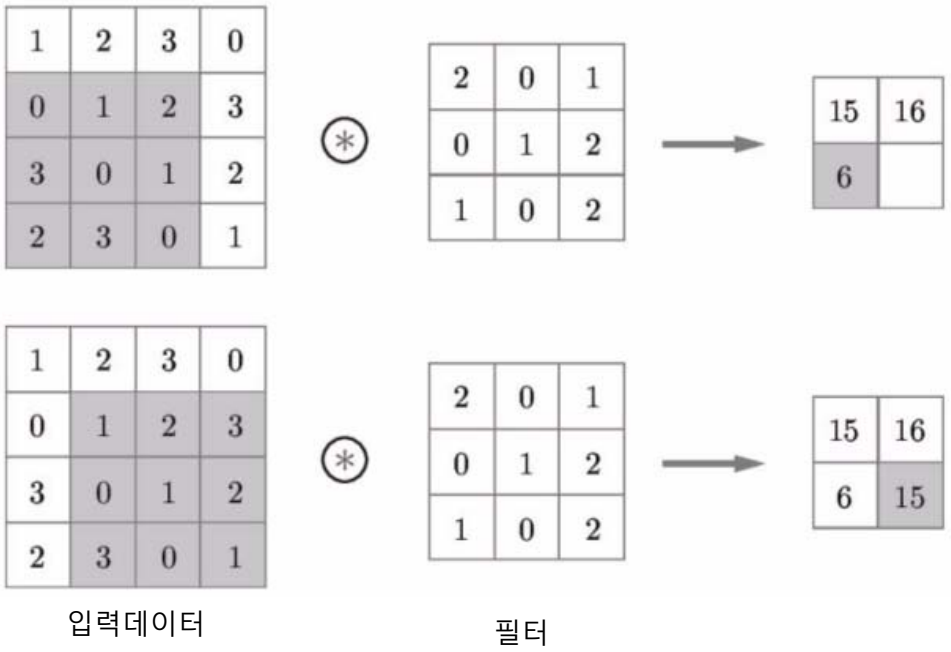


FIG. 1 - IMAGE CLASSIFICATION TASK

https://www.cs.toronto.edu/~frossard/post/vgg16/

약간의 산수 !!





03-5-mnist-cnn.py 를 열어주세요

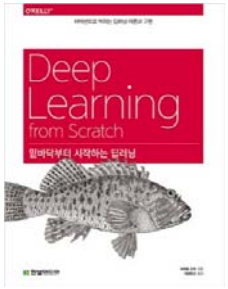
2017년도 2학기 수업계획서

* 출력시간 : 2017-07-13 00:07:53

수강과목				담당교수			
과목명	인공지능개론	교과목번호/분반	717005/01	소속	융합전공	연구실	
이수구분	전선	시간	월7,8 수7	대표교수	김승찬	합동강좌	
강의실	10B111	학점-수업-실습	3-3-0	전자우편	seungchankim12@gmail.com		
				연락처		면담가능시간	화,수 화 6교시, 수 5-6교시, 기타 사전 연락

Acknowledgement

universitätbonn ais



모두를 위한 머신러닝/딥러닝 강의

Seung-han

Seung-Chan

Jeung-Chan

감사합니다.