AI School 6기 4주차

파이썬 기초 – 클래스

Convolutional Neural Network(CNN) 기초

CNN 모델을 활용한 MNIST 데이터 분류

AI School 6기 4주차

파이썬 기초 - 클래스

파이썬의 클래스

클래스

객체(인스턴스)

파이썬의 클래스

- 클래스는 객체의 구조와 행동을 정의
- 객체의 클래스는 초기화를 통해 제어
- 클래스는 복잡한 문제를 다루기 쉽도록 만들어줌

airtravel.py

```
class Flight:
    pass
```

flight_test.py

```
from Python_basic.airtravel import Flight
#생성한 클래스를 import

f = Flight() #클래스 객체 생성 및 변수에 할당
print(type(f))
```

파이썬의 클래스

- 메소드란 클래스 내의 함수
- self: 파이썬 메소드의 첫번째 파라미터명
- 인스턴스 메소드란 객체에서 호출되어질수 있는 함수

airtravel.py

```
class Flight:
    def number (self): #메소드 작성
        return 'KE081'
```

flight_test.py

```
from Python_basic.airtravel import Flight
#생성한 클래스를 import

f = Flight() #클래스 객체 생성 및 변수에 할당, 생성자
print(f.number())
```

파이썬의 클래스

- 초기화

airtravel.py

```
class Flight:
    def __init__(self, number):
        self._number = number

    def number(self):
            return self._number
```

flight_test.py

```
from Python_basic.airtravel import Flight

f = Flight('KE082')

print(f.number())
print(f._number)
```

파이썬의 클래스

- 유효성 검증

airtravel.py

```
class Flight:
    def __init__(self, number):
        # print('init')
        if not number[:2].isalpha():
            raise ValueError("첫 두글자가 알파벳이 아닙니다.")
        if not number[:2].isupper():
            raise ValueError("첫 두글자가 대문자가 아닙니다.")
        if not number[2:].isdigit():
            raise ValueError("세번째 글자 이상이 양의 숫자가 아닙니다.")
        self._number = number
```

flight_test.py

```
from Python_basic.airtravel import Flight

f = Flight('Ke082') #KEE82, 0KE082
```

파이썬의 클래스

- 더블 언더바 __: 클래스 변수에 외부접근을 막아줌

airtravel.py

```
class Flight:
    def __init__(self, number):
        # print('init')
        if not number[:2].isalpha():
            raise ValueError("첫 두글자가 알파벳이 아닙니다.")
        if not number[:2].isupper():
            raise ValueError("첫 두글자가 대문자가 아닙니다.")
        if not number[2:].isdigit():
            raise ValueError("세번째 글자 이상이 양의 숫자가 아닙니다.")
        self.__number = number
```

flight_test.py

```
from Python_basic.airtravel import Flight
f = Flight('KE082')
print(f.number())
f.__number = 'KE081'
print(f.number())
```

파이썬의 클래스

- 인스턴스 속성(변수)

airtravel.py

```
class Flight:
    def __init__(self, number, passenger_num):
        self.__number = number
        self._passenger_num = passenger_num

    def number(self): #메소드 작성
            return self.__number

    def add_passenger(self, num):
        self._passenger_num += num
```

flight_test.py

```
f1 = Flight('KE082', 0) #클래스 객체 생성 및 변수에 할당
f2 = Flight('KE081', 0)
f1.add_passenger(2)
f2.add_passenger(3)
print(f1._passenger_num)
print(f2._passenger_num)
```

파이썬의 클래스

- 클래스 속성

airtravel.py

```
class Flight:
    nation = 'Korea'
.
.
.
```

flight_test.py

```
f1 = Flight('KE082', 0) #클래스 객체 생성 및 변수에 할당
f2 = Flight('KE081', 0)
print(f1.nation)
print(f2.nation)
```

숙제1

- fourcal.py에서 사칙연산을 수행하는 Calculator 클래스를 만드세요.
- def __init__(self, num1, num2)
- def add(self)

  .

  .

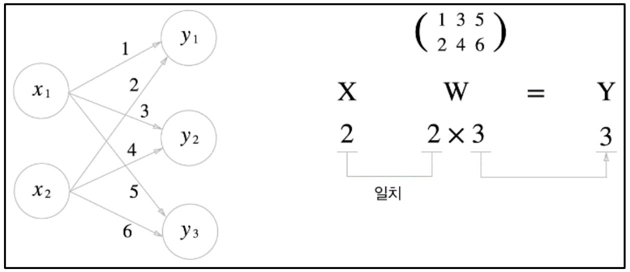
  return result
- def sub(self)
- def mul(self)
- def div(self)

- calculator_test.py에서 객체 생성 후 4가지 메소드 사용 결과 출력

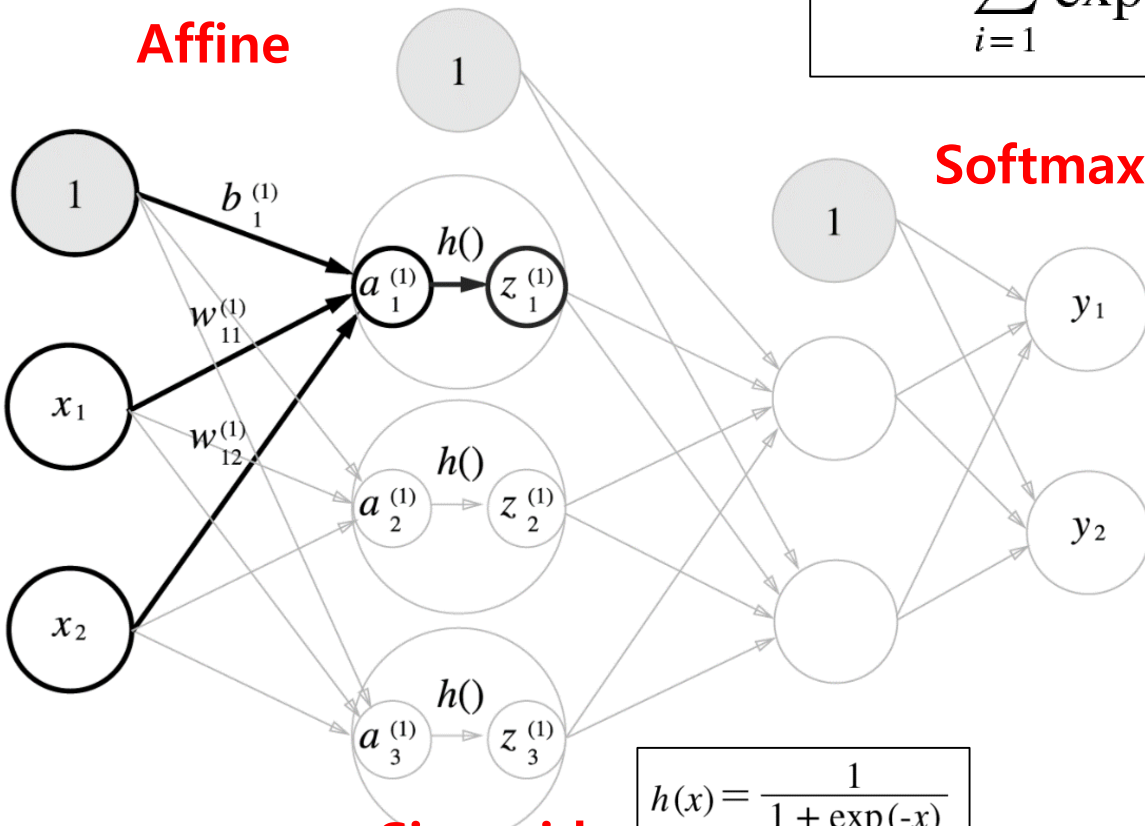
AI School 6기 4주차

CNN 기초

Multi Layer Perceptron (MLP), Fully Connected Layer (FC)



$$y_k = \frac{\exp(a_k)}{\sum_{i=1}^{n} \exp(a_i)}$$



Loss function

$$E = -\sum_k t_k \log y_k$$

$$h(x) = \frac{1}{1 + \exp(-x)}$$

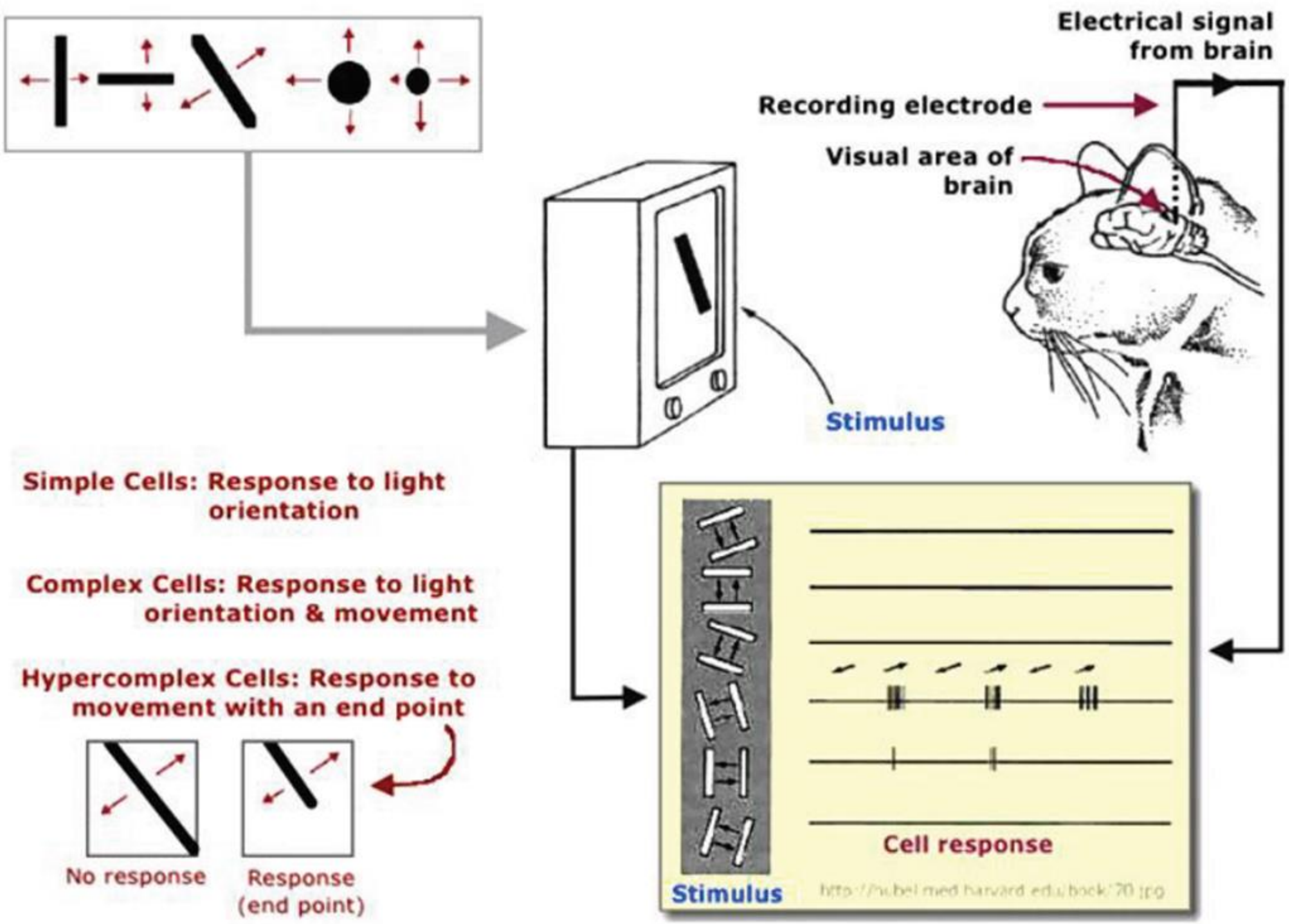
Sigmoid

ReLU

$$h(x) = \begin{cases} x & (x > 0) \\ 0 & (x \leq 0) \end{cases}$$

AI School
6기 입문반

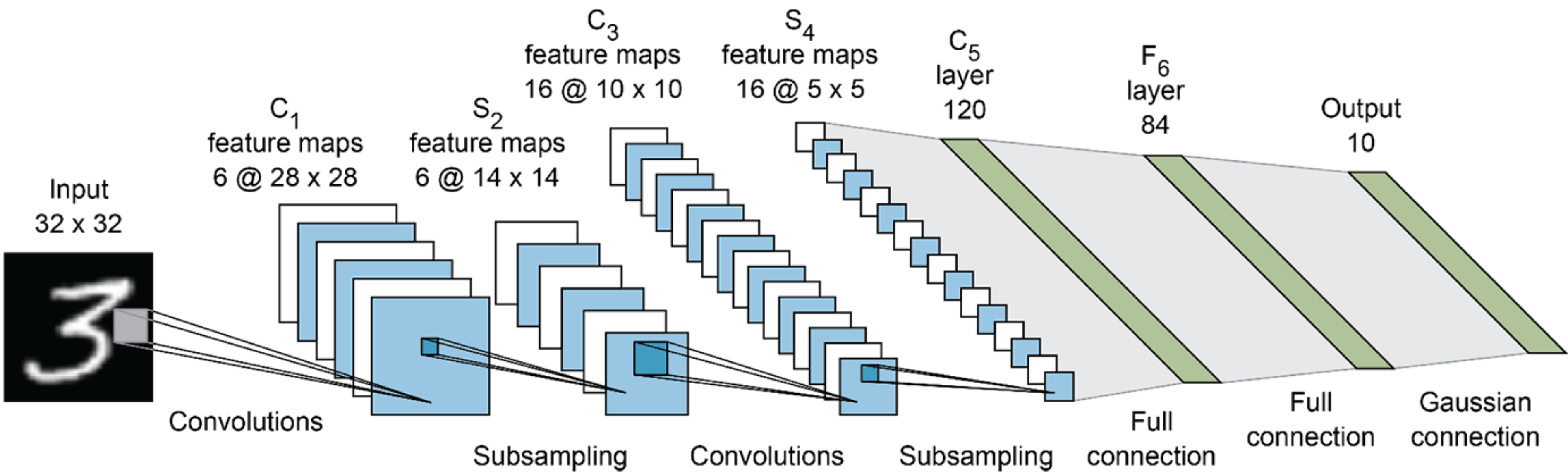
Convolutional Neural Networks



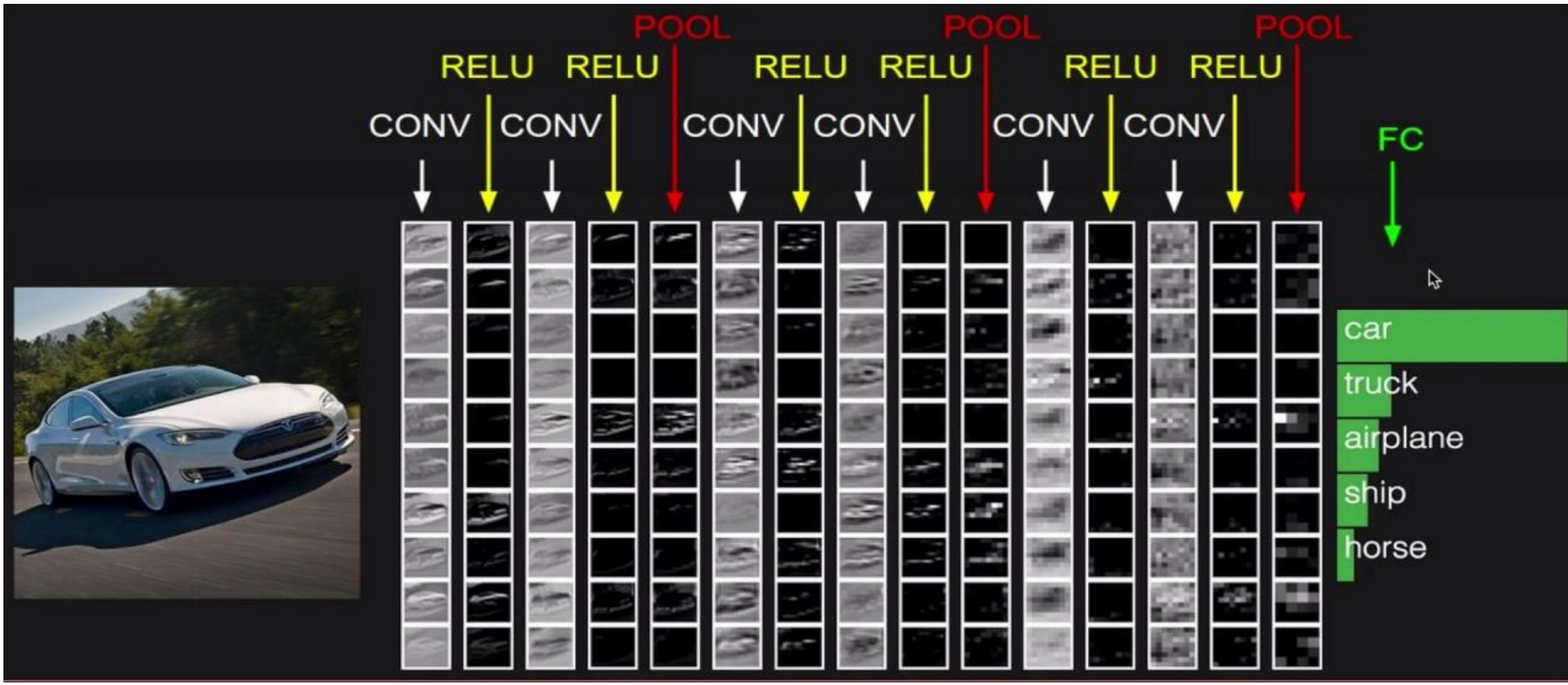
Hubel & Wiesel, 1959

Convolutional Neural Networks

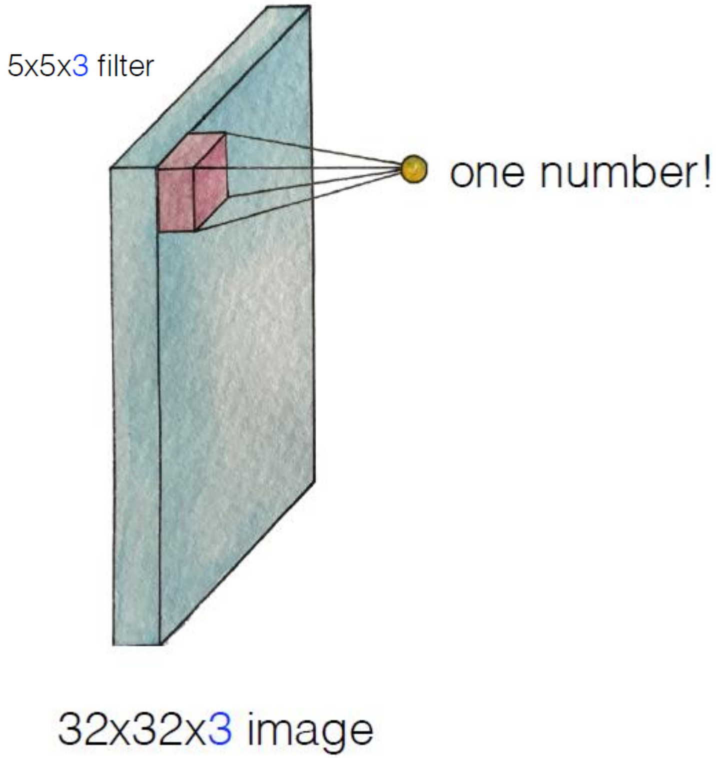
- Convolution과 Pooling을 반복하여 상위 Feature를 구성
- Convolution은 Local영역에서의 특정 Feature를 얻는 과정
- Pooling은 Dimension을 줄이면서도, Translation-invariant 한 Feature를 얻는 과정



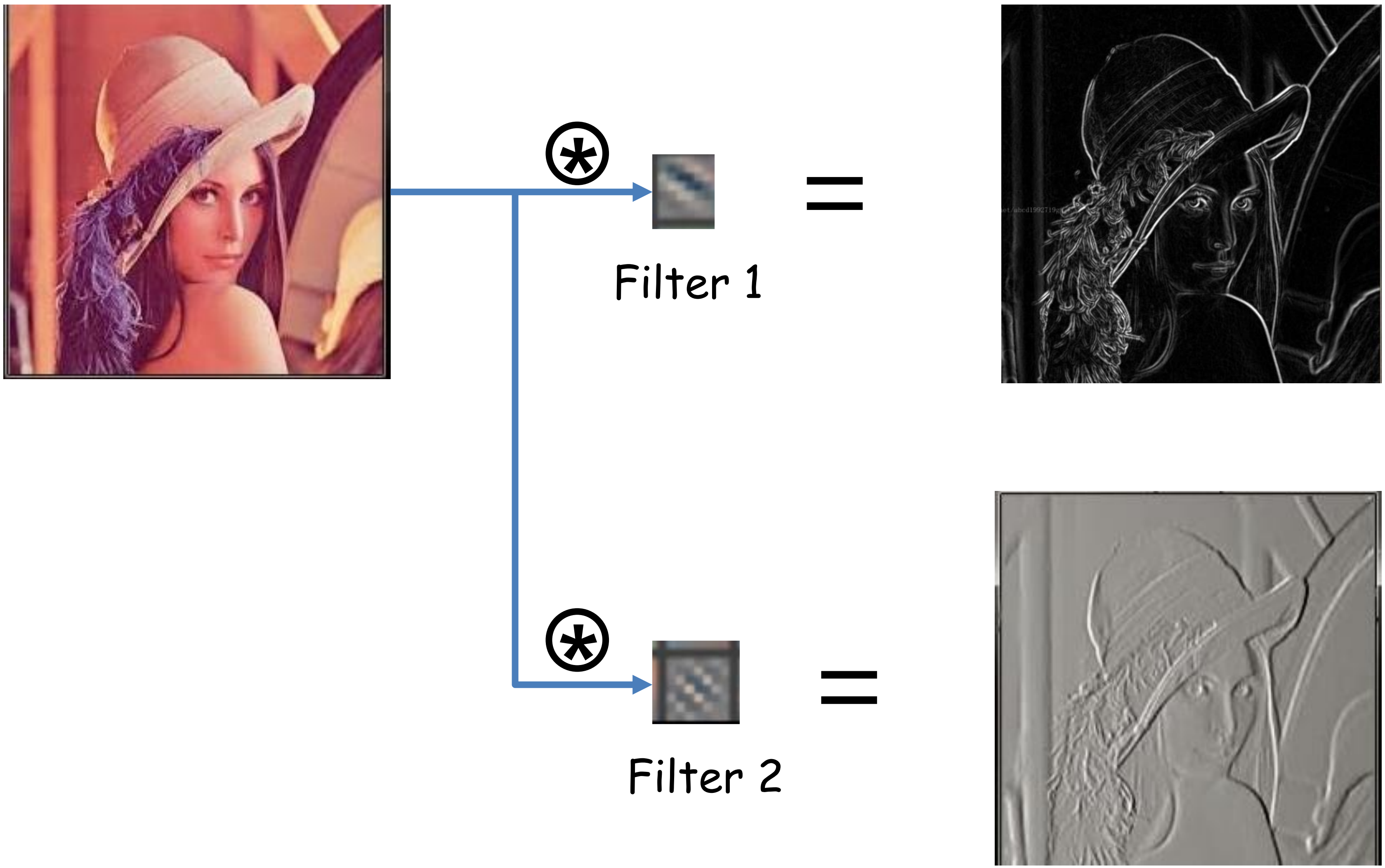
Convolutional Neural Networks



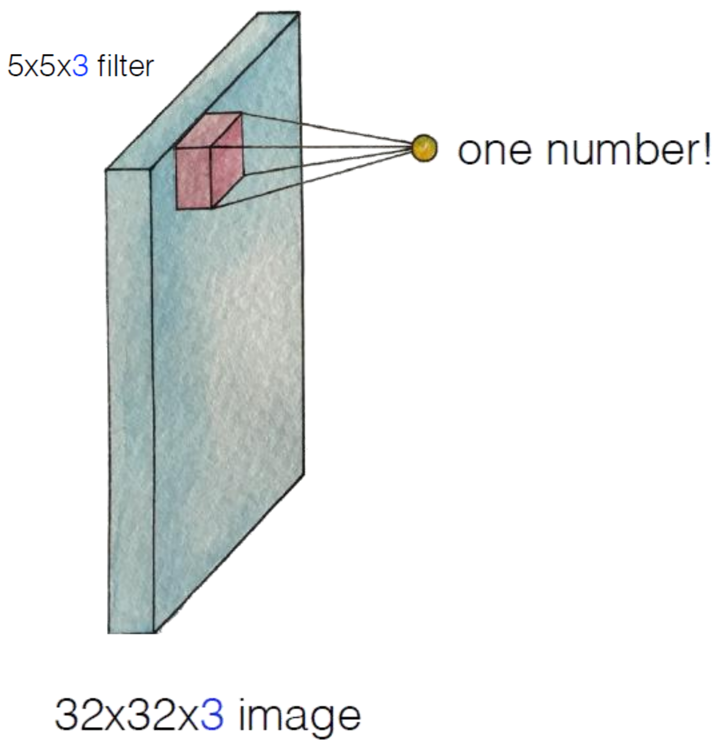
Convolution, Filter



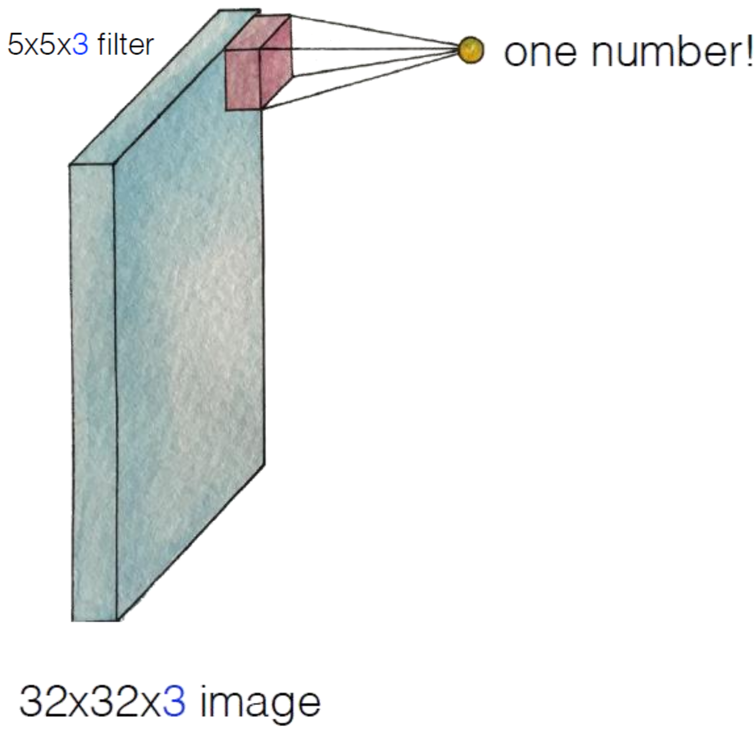
Convolution, Filter



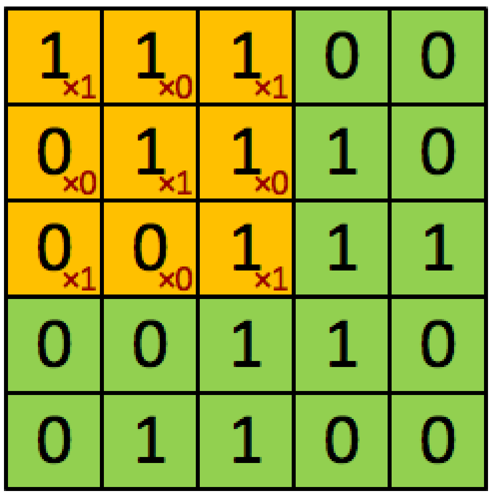
Convolution, Filter



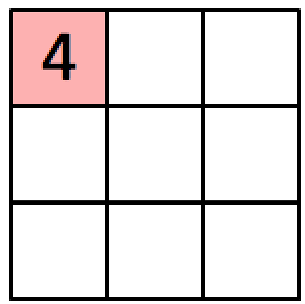
Convolution, Filter



Convolution, Filter

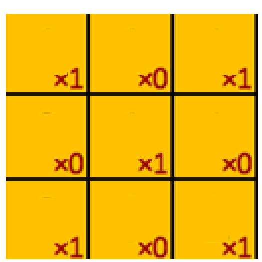


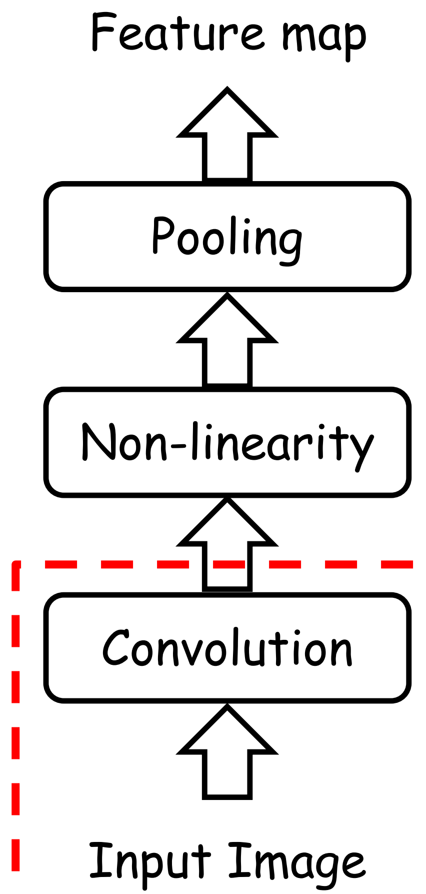
Image



Convolved Feature

*filter





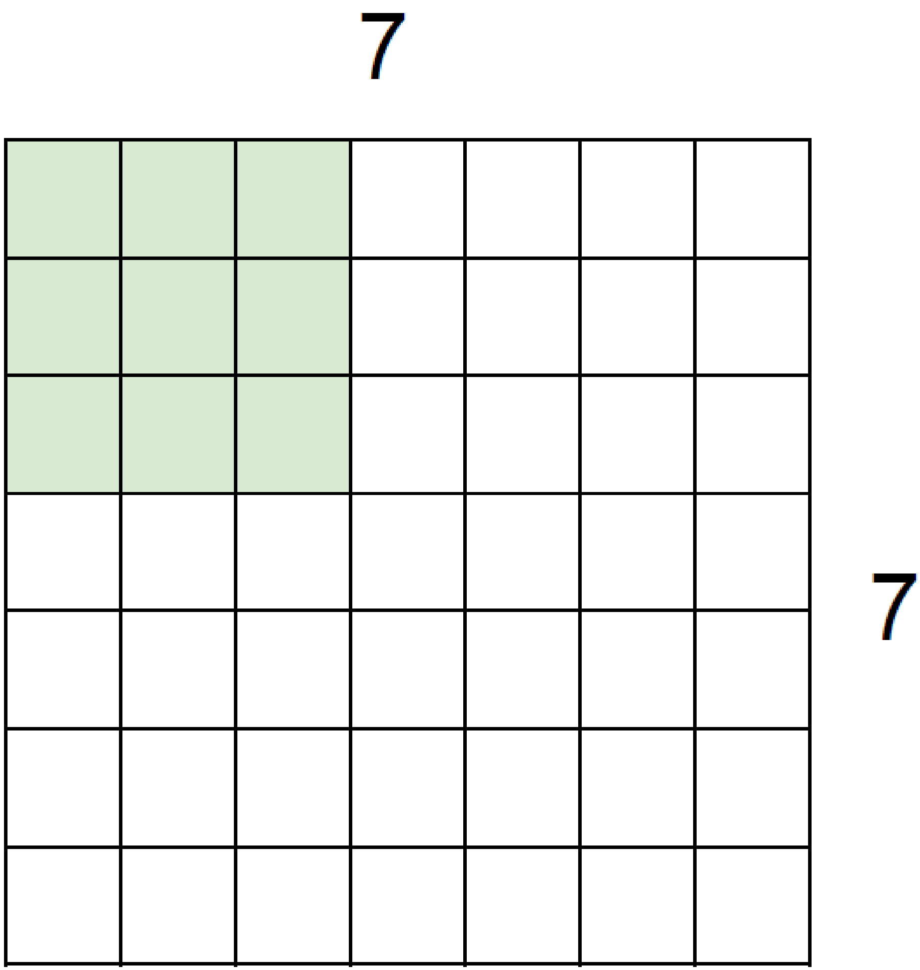
Apply convolution operation!

$$\begin{bmatrix} 1 & 1 & 1 & 1 \\ 2 & 0 & 3 & 2 \\ 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 2 \end{bmatrix} \circledast \begin{bmatrix} 1 & 1 & 0 \\ 2 & 0 & 0 \\ 1 & 0 & 1 \end{bmatrix} = \begin{bmatrix} & \\ & \end{bmatrix} \text{?}$$

숙제2 – Apply convolution operation!

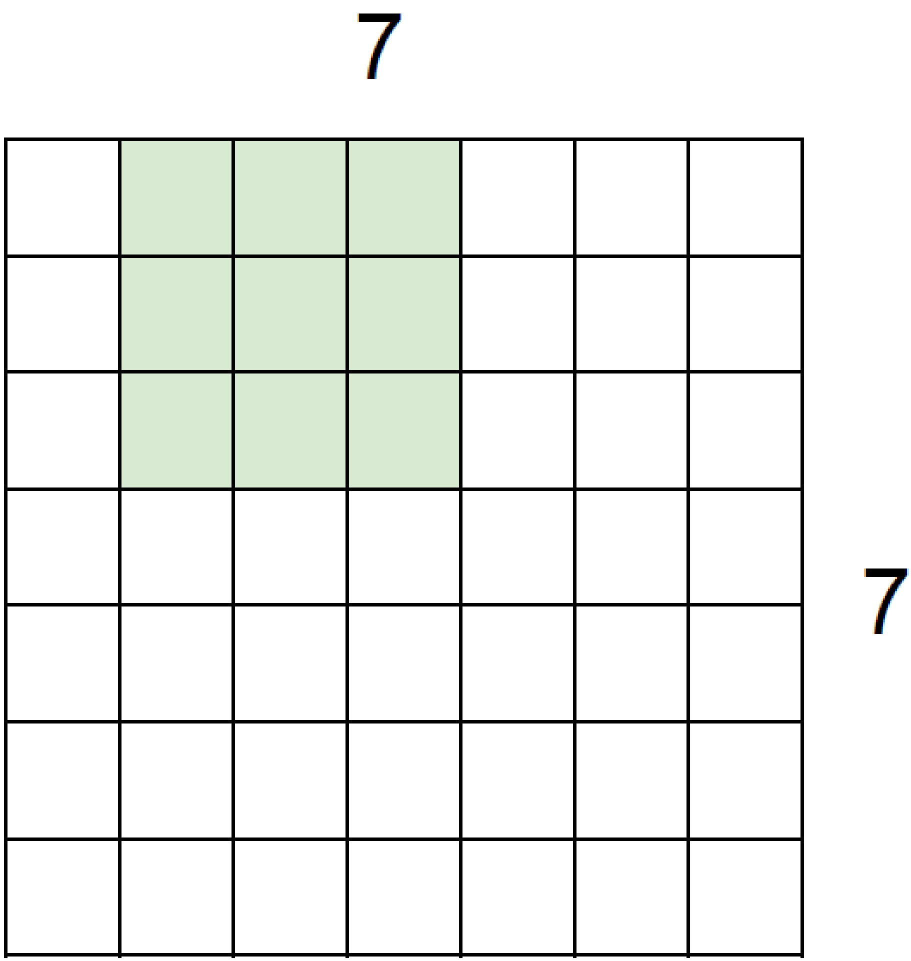
$$\begin{bmatrix} 1 & 2 & 1 & 0 \\ 0 & 1 & 2 & 2 \\ 1 & 3 & 0 & 1 \\ 1 & 0 & 1 & 1 \end{bmatrix} \circledast \begin{bmatrix} 1 & 0 & 2 \\ 1 & 1 & 1 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} & \\ & \end{bmatrix} ?$$

Convolution, Filter



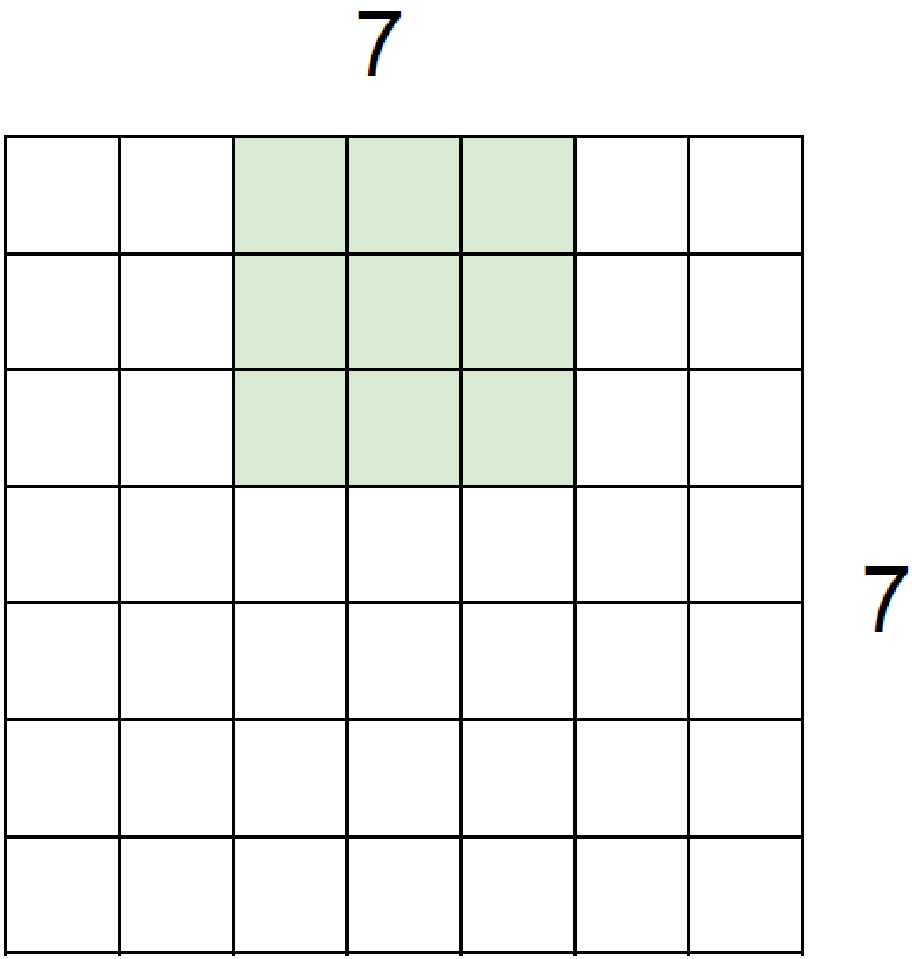
7x7 input (spatially)
assume 3x3 filter

Convolution, Filter



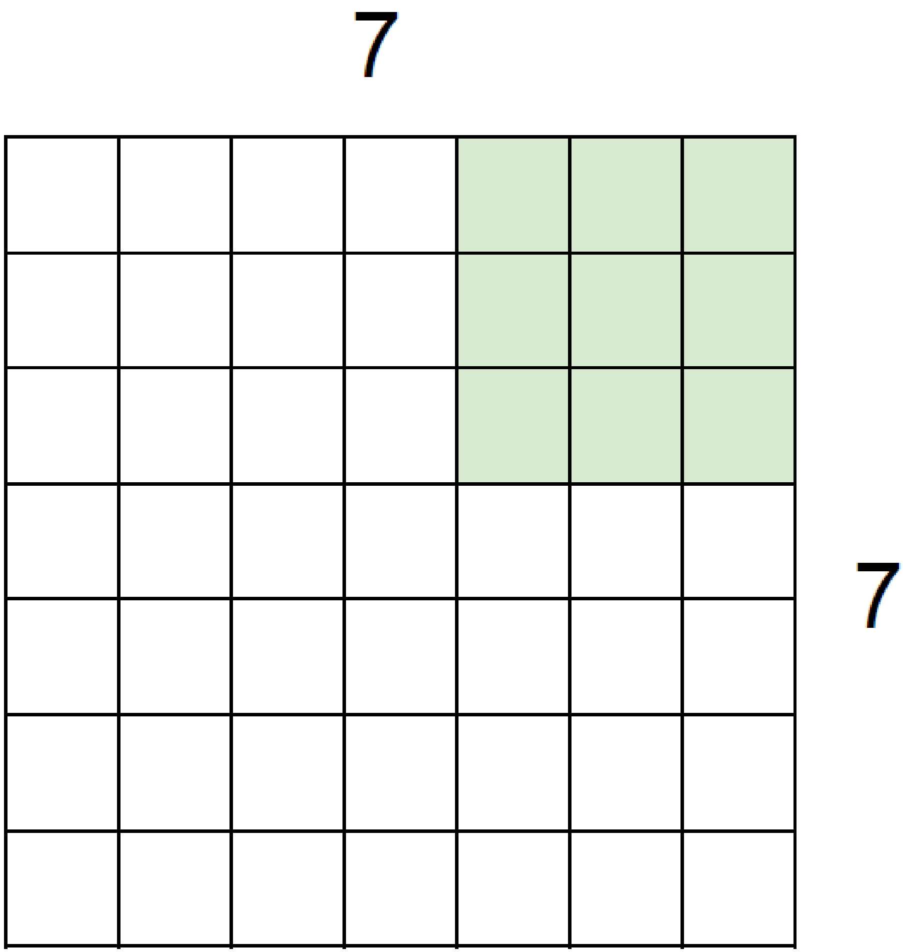
7x7 input (spatially)
assume 3x3 filter

Convolution, Filter



7x7 input (spatially)
assume 3x3 filter

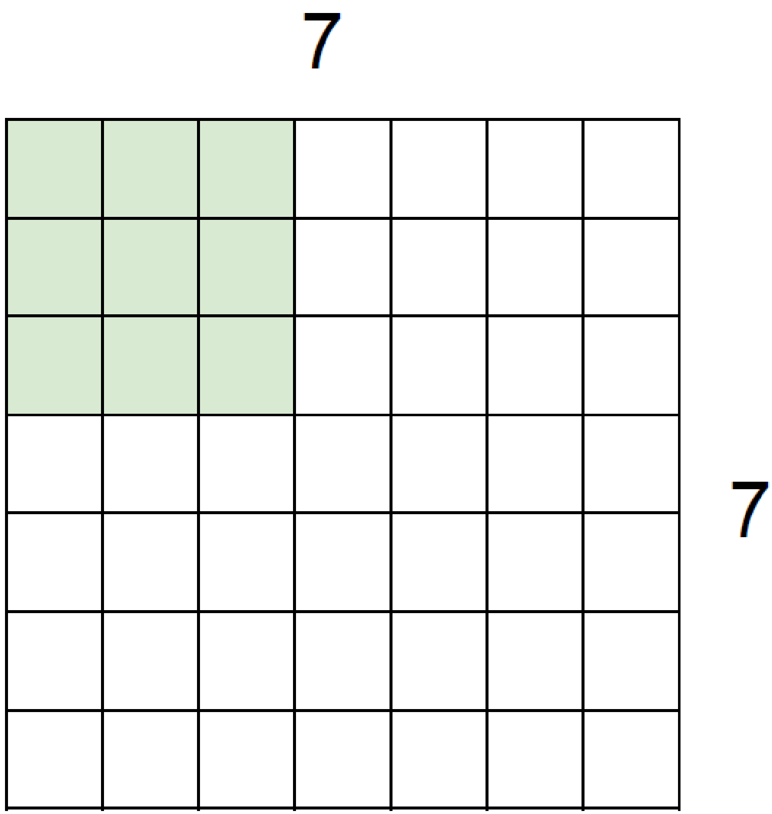
Convolution, Filter



7x7 input (spatially)
assume 3x3 filter

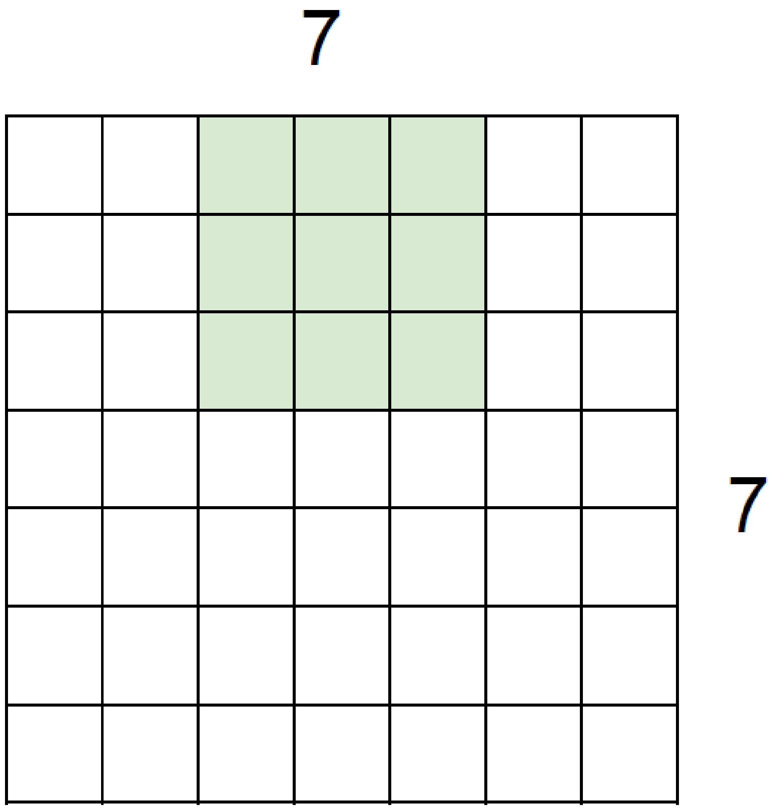
=> 5x5 output

Convolution, Filter



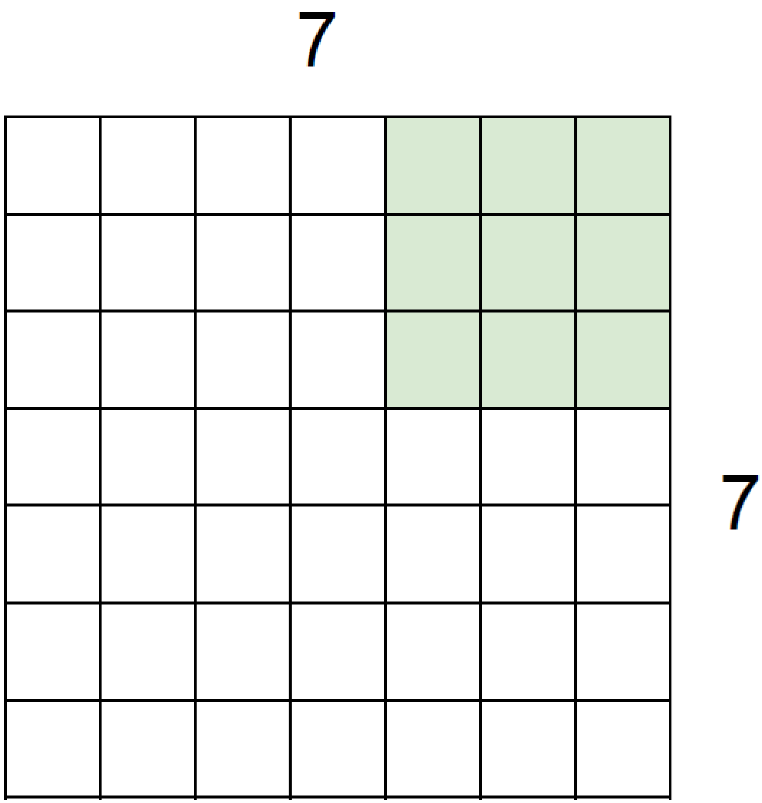
7x7 input (spatially)
assume 3x3 filter
applied **with stride 2**

Convolution, Filter



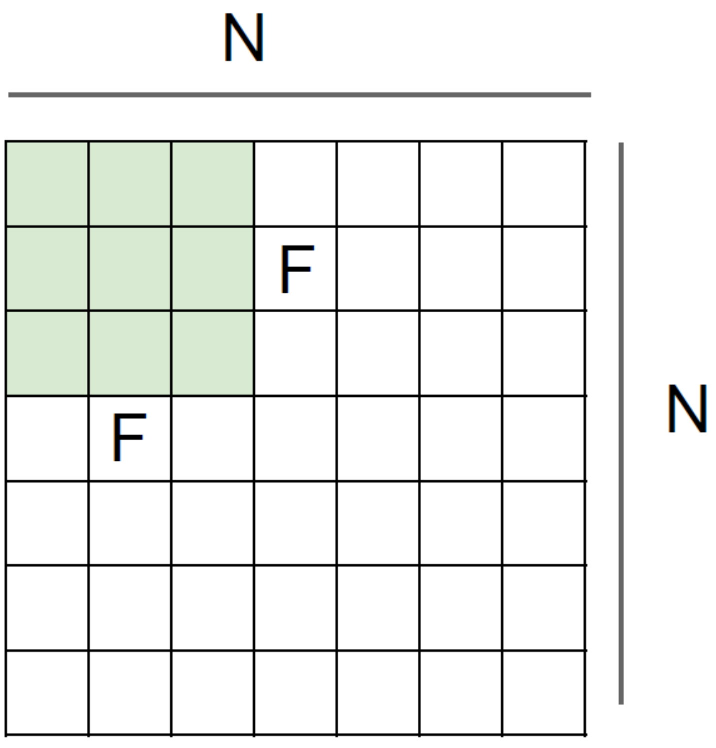
7x7 input (spatially)
assume 3x3 filter
applied **with stride 2**

Convolution, Filter



7x7 input (spatially)
assume 3x3 filter
applied **with stride 2**
=> 3x3 output!

Convolution, Filter

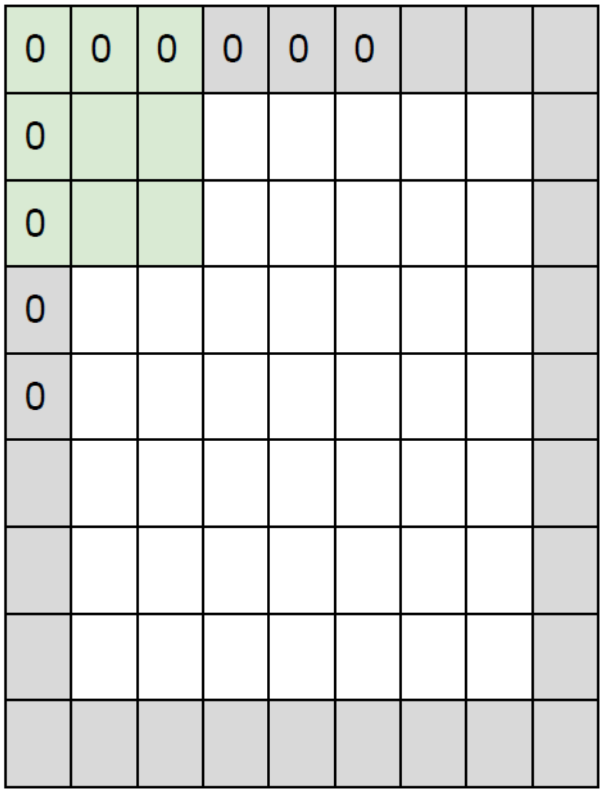


Output size:
(N - F) / stride + 1

e.g. N = 7, F = 3:
stride 1 => (7 - 3)/1 + 1 = 5
stride 2 => (7 - 3)/2 + 1 = 3
stride 3 => (7 - 3)/3 + 1 = 2.33 :\

Convolution, Filter

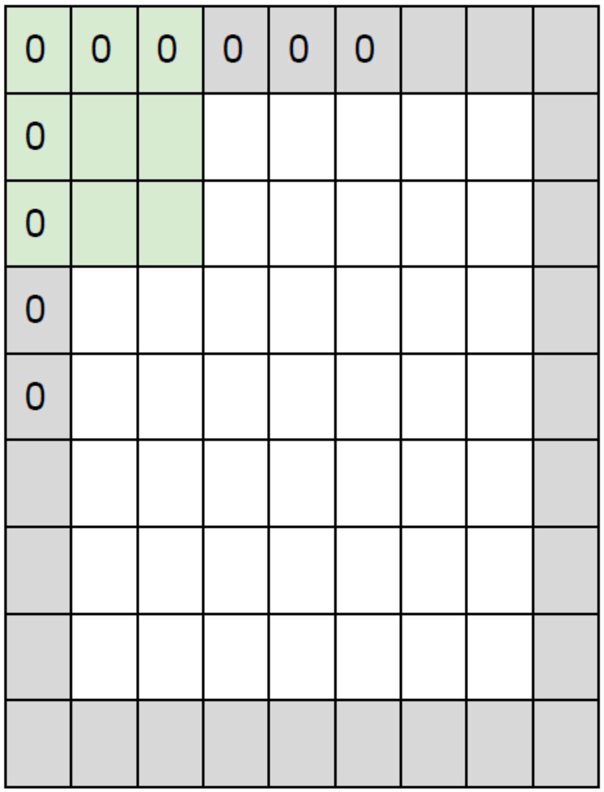
In practice: Common to zero pad the border



e.g. input 7x7
3x3 filter, applied with **stride 1**
pad with 1 pixel border => what is the output?

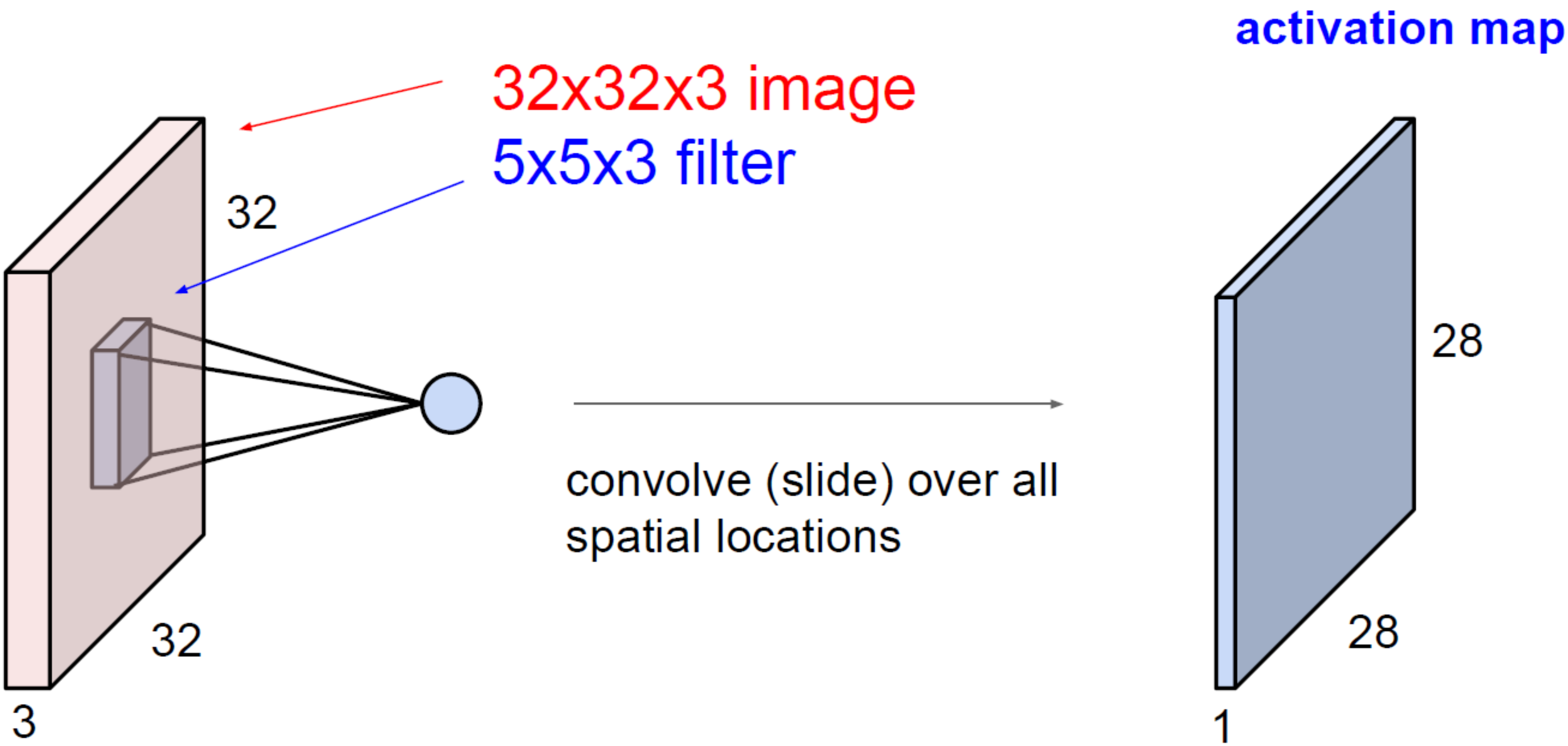
(recall:)
(N - F) / stride + 1

Homework – calculate shape!



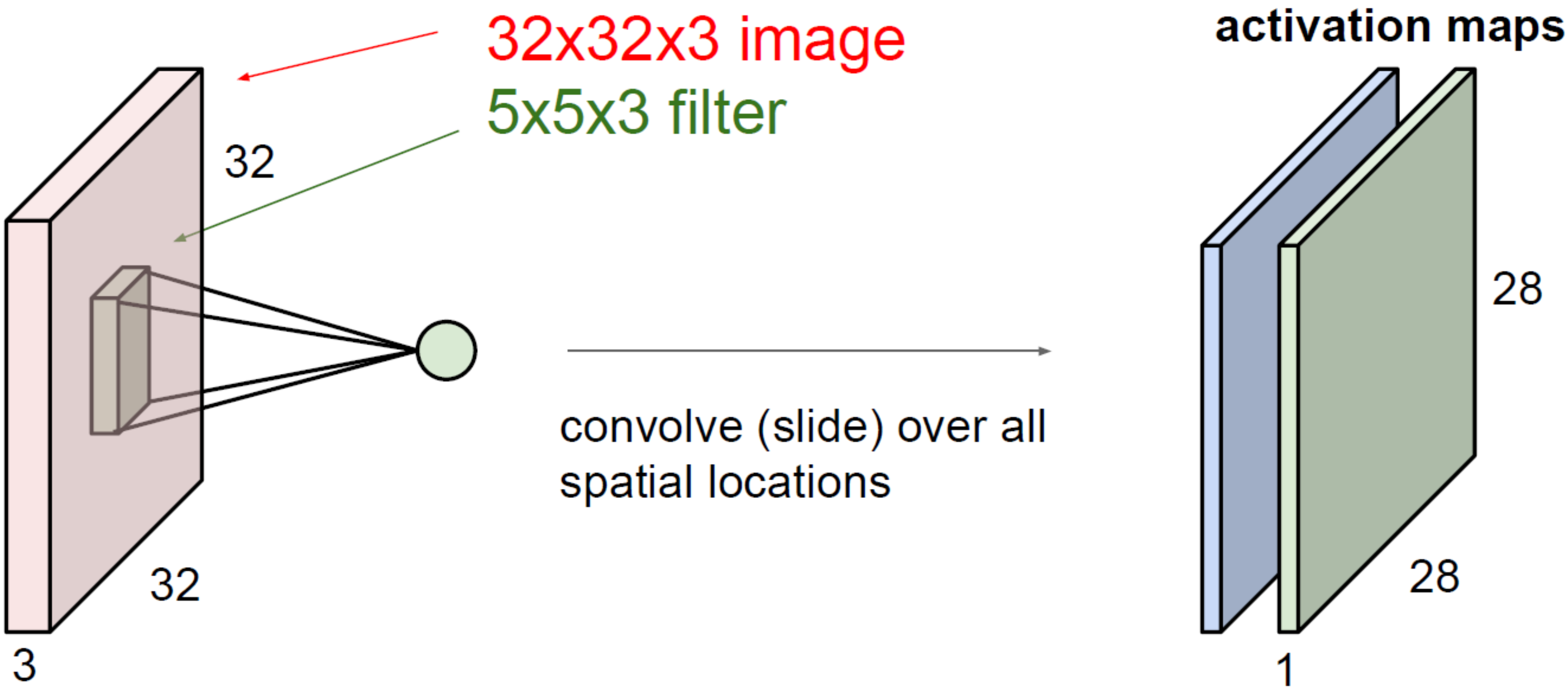
Input 7x7
5x5 filter, applied with stride 2
Pad with 1 pixel border
What is the output?

Convolution, Filter



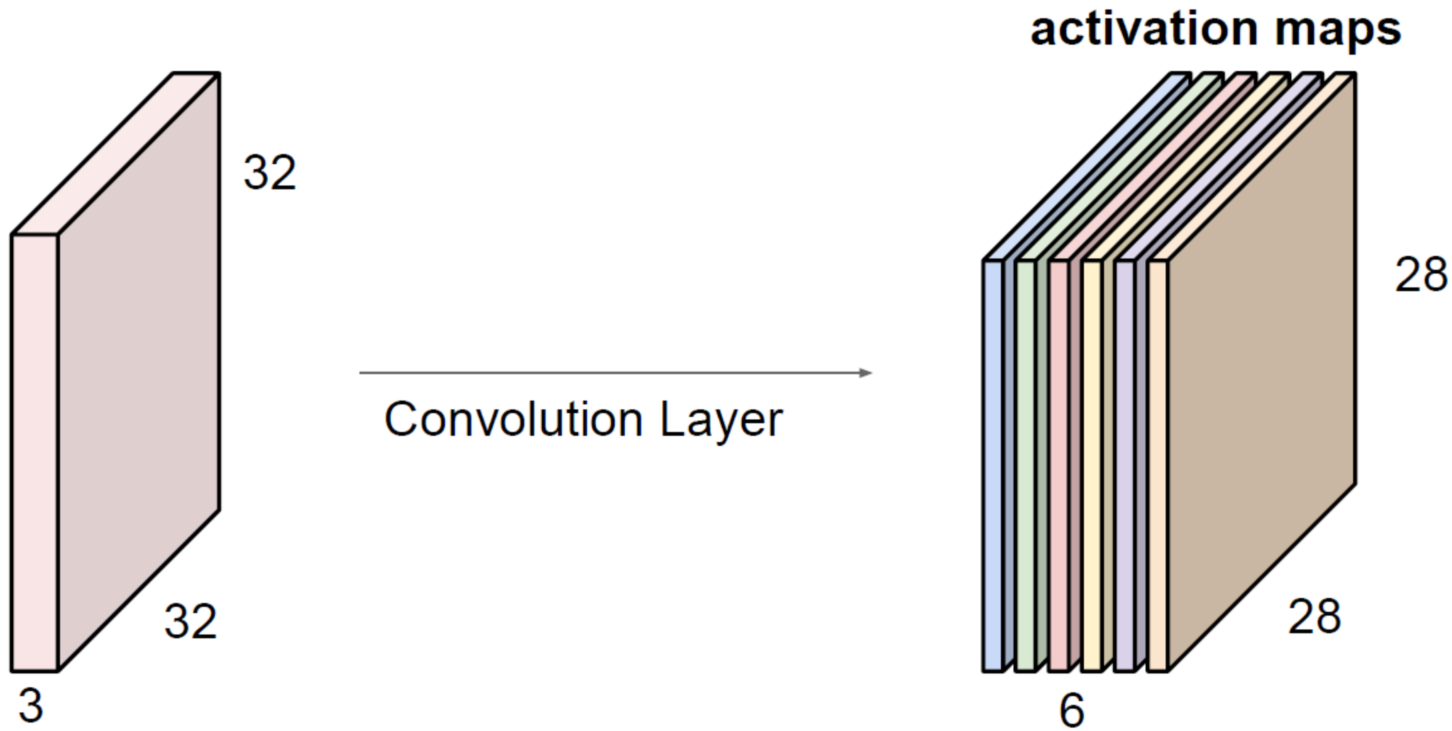
Convolution, Filter

consider a second, green filter



Convolution, Filter

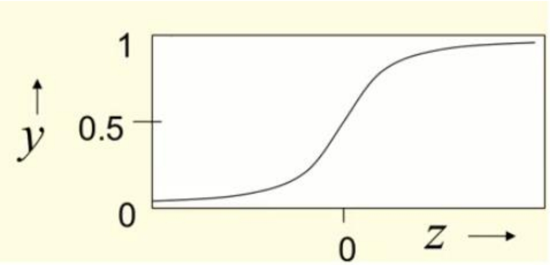
For example, if we had 6 5x5 filters, we'll get 6 separate activation maps:



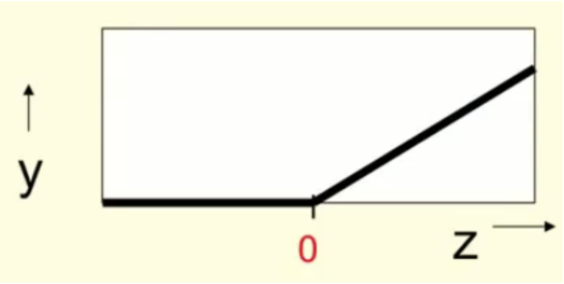
We stack these up to get a "new image" of size 28x28x6!

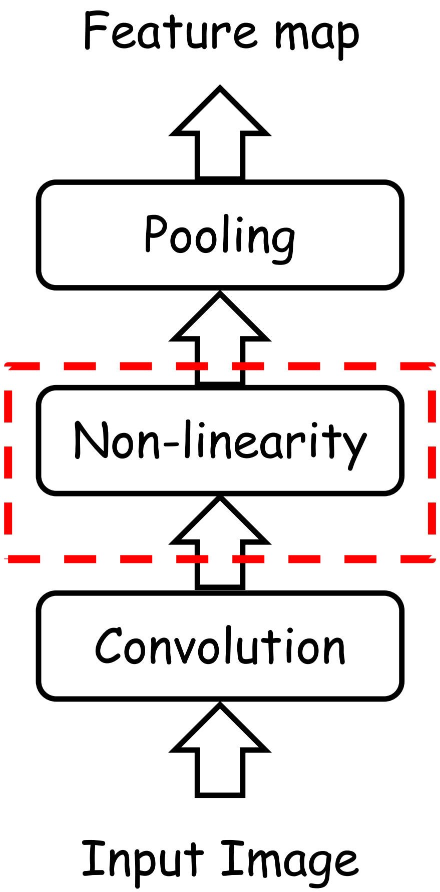
Non-linearity

Sigmoid

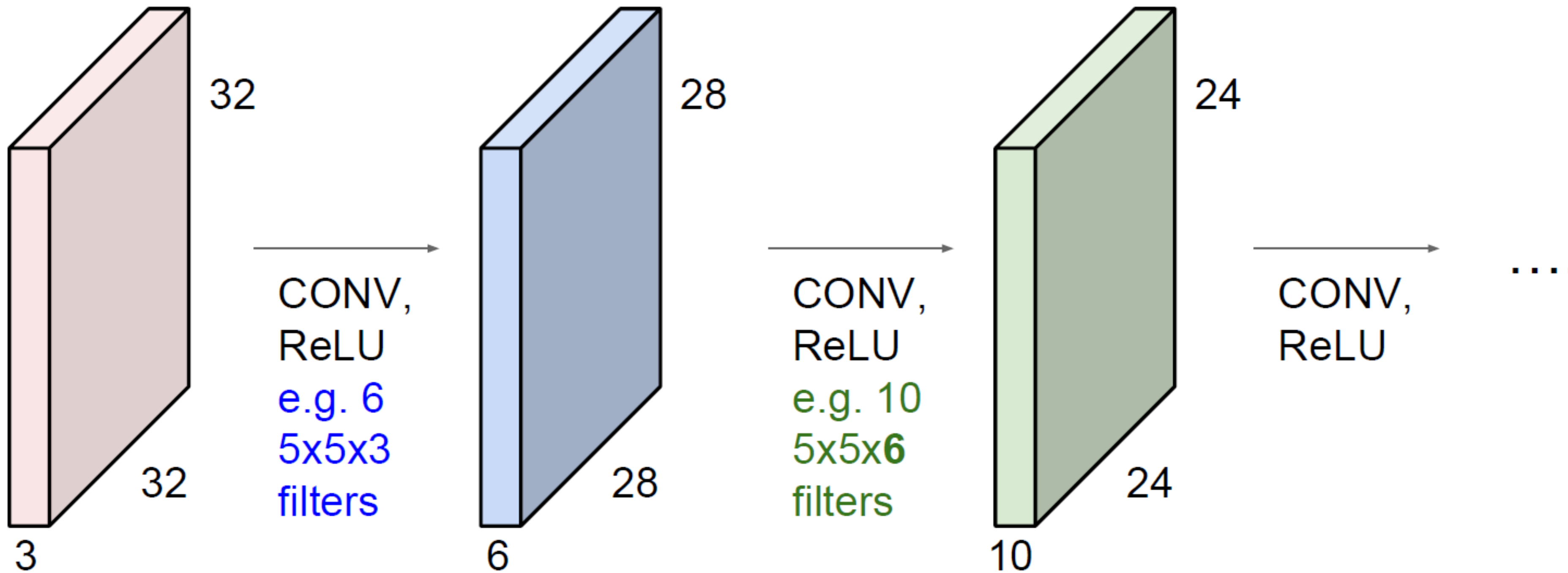


Rectified linear unit

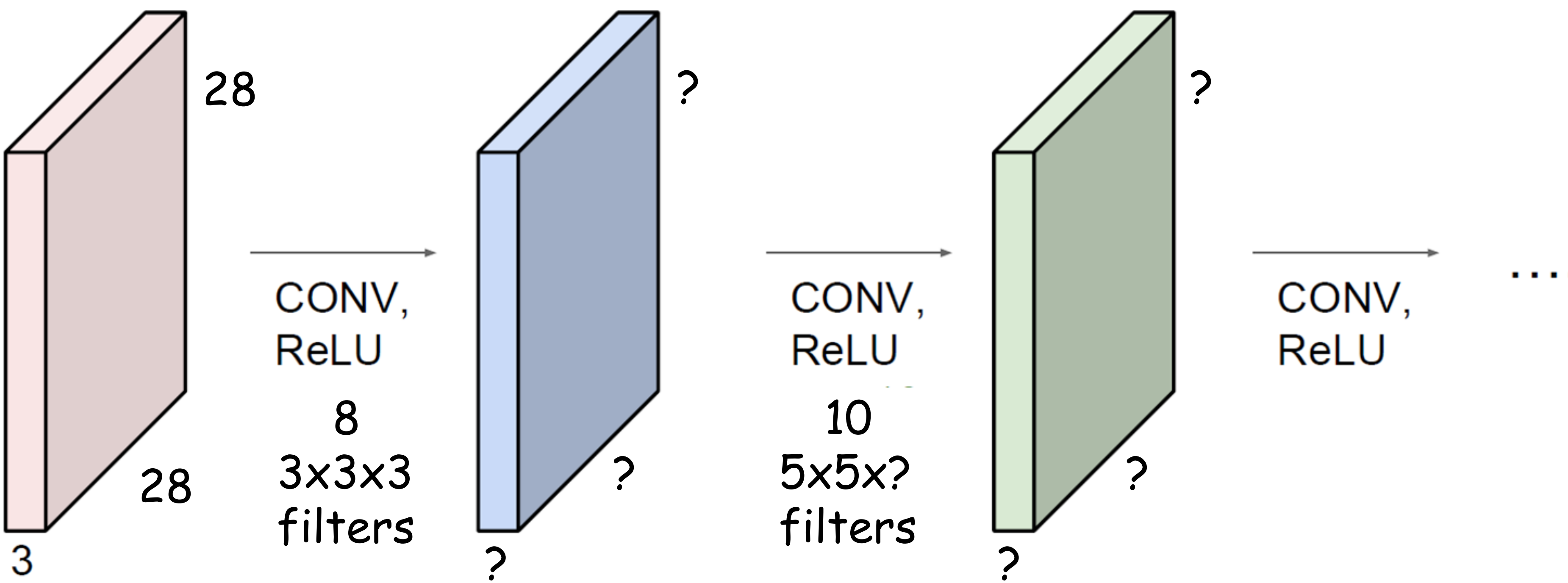




Convolution, Filter

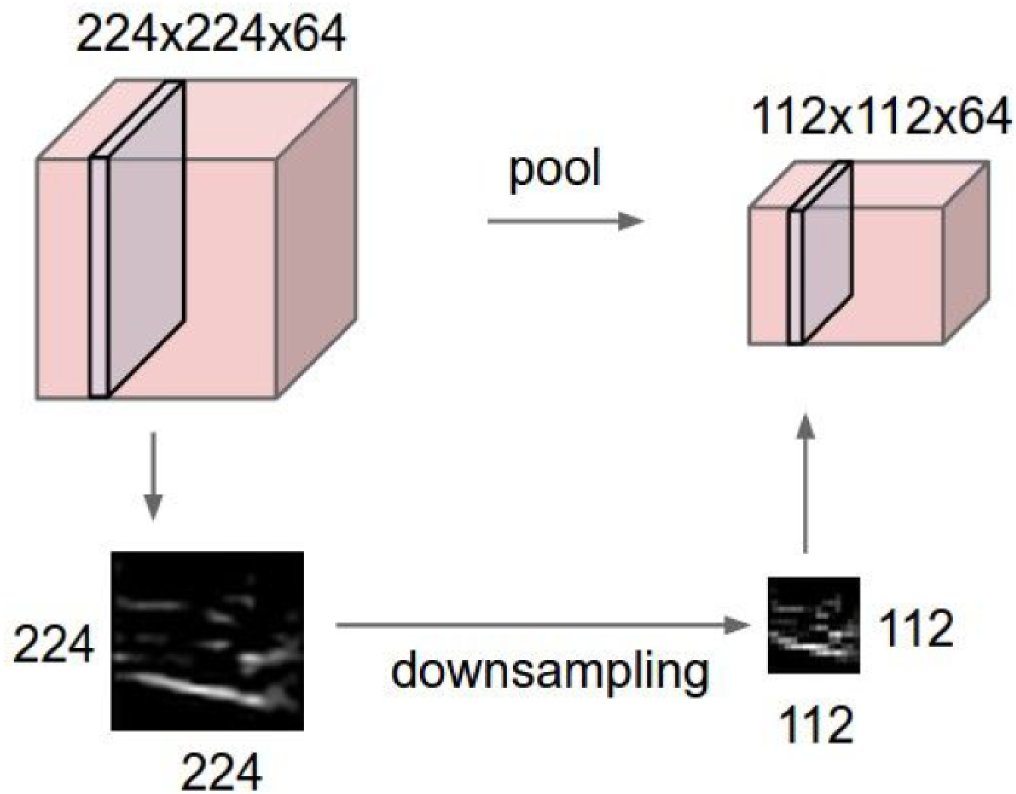


숙제3 - Convolution, Filter

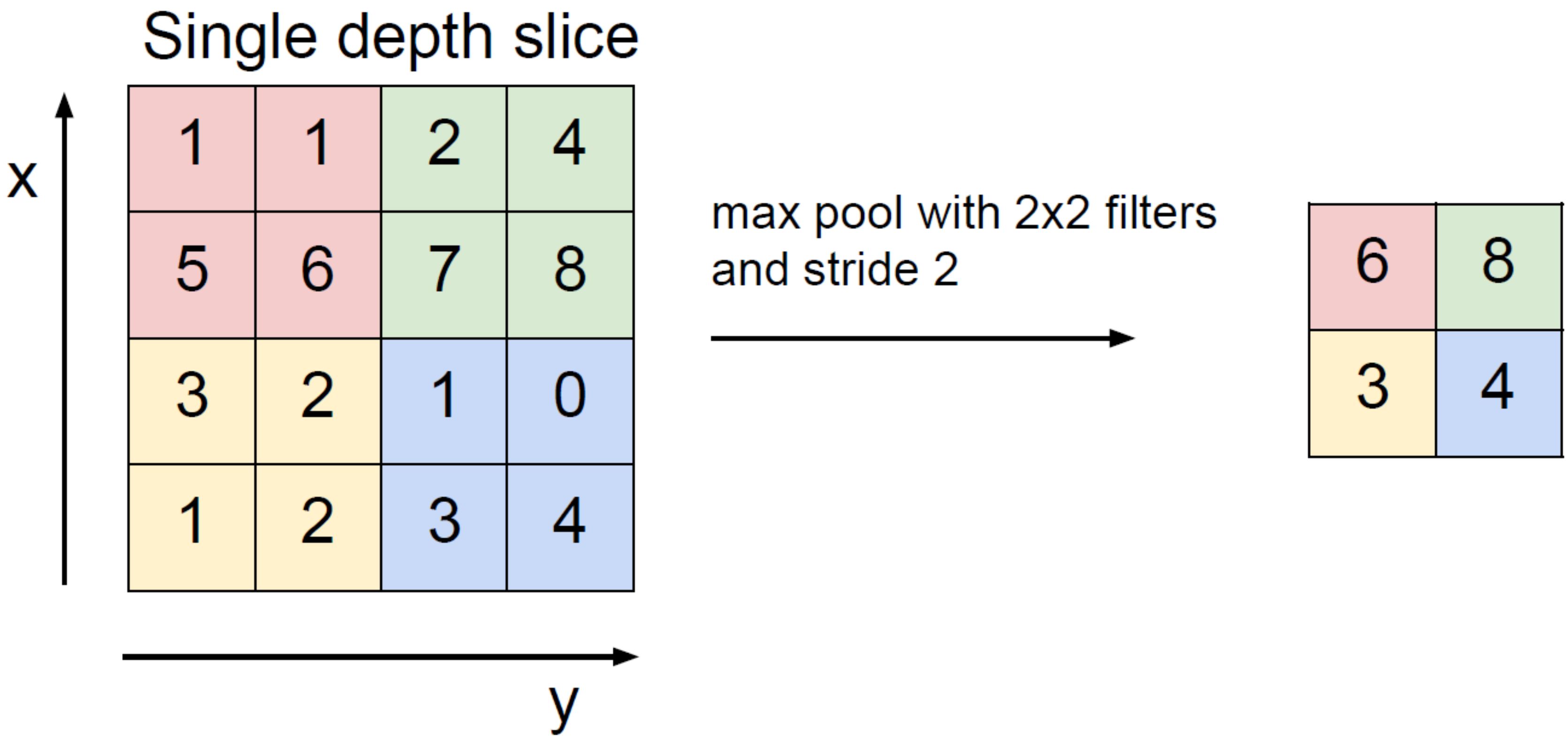


Pooling

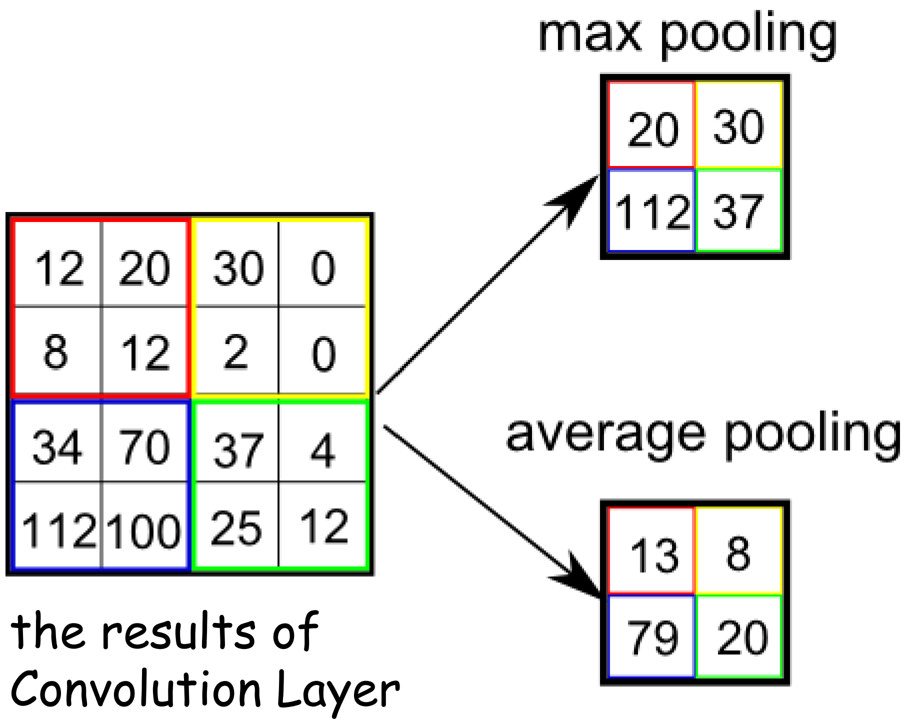
- makes the representations smaller and more manageable
- operates over each activation map independently:

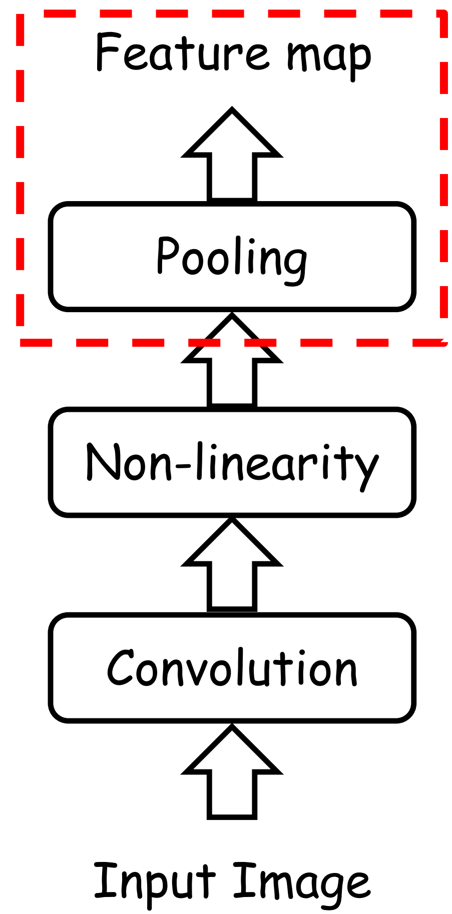


Pooling

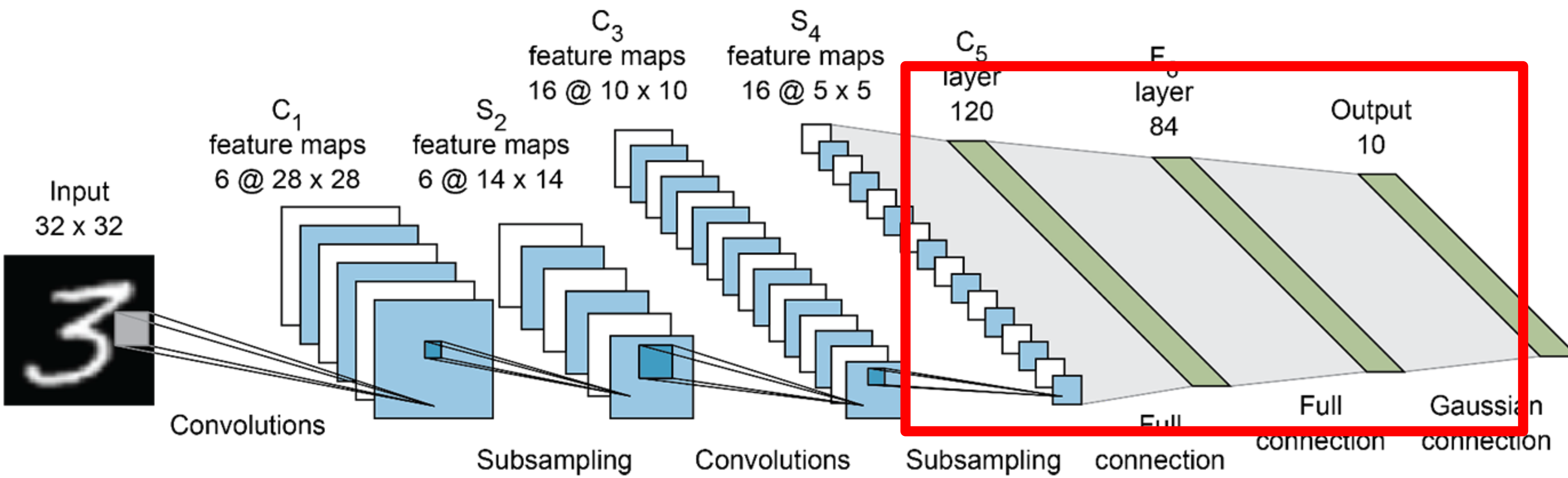

Single depth slice
x
y
1 1 2 4
5 6 7 8
3 2 1 0
1 2 3 4
max pool with 2x2 filters
and stride 2
6 8
3 4

Pooling

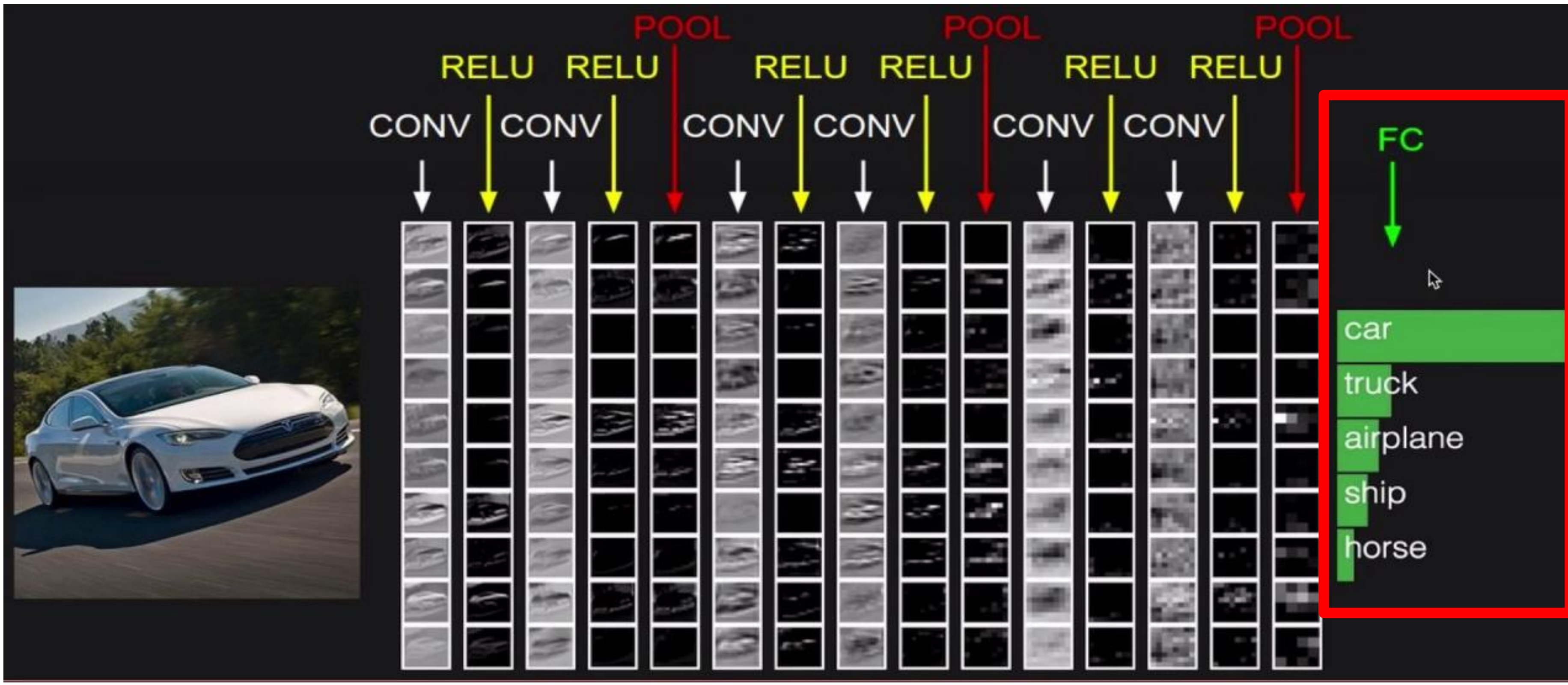

max pooling
20 30
112 37
average pooling
13 8
79 20
12 20 30 0
8 12 2 0
34 70 37 4
112 100 25 12
the results of
Convolution Layer


Feature map
Pooling
Non-linearity
Convolution
Input Image

Fully connected layer



Fully connected layer



AI School 6기 4주차

CNN 모델을 활용한 MNIST 데이터 분류

Tensorboard

http://localhost:6006

```
accuracy = tf.reduce_mean(tf.cast(correct_prediction, tf.float32))
summary_op = tf.summary.scalar("accuracy", accuracy)
.
.
.
timestamp = str(int(time.time()))
out_dir = os.path.abspath(os.path.join(os.path.curdir, "runs", timestamp))
train_summary_dir = os.path.join(out_dir, "summaries", "train")
train_summary_writer = tf.summary.FileWriter(train_summary_dir, sess.graph)
val_summary_dir = os.path.join(out_dir, "summaries", "dev")
val_summary_writer = tf.summary.FileWriter(val_summary_dir, sess.graph)
checkpoint_dir = os.path.abspath(os.path.join(out_dir, "checkpoints"))
checkpoint_prefix = os.path.join(checkpoint_dir, "model")
if not os.path.exists(checkpoint_dir):
    os.makedirs(checkpoint_dir)
saver = tf.train.Saver(tf.global_variables(), max_to_keep=10)
max = 0
```

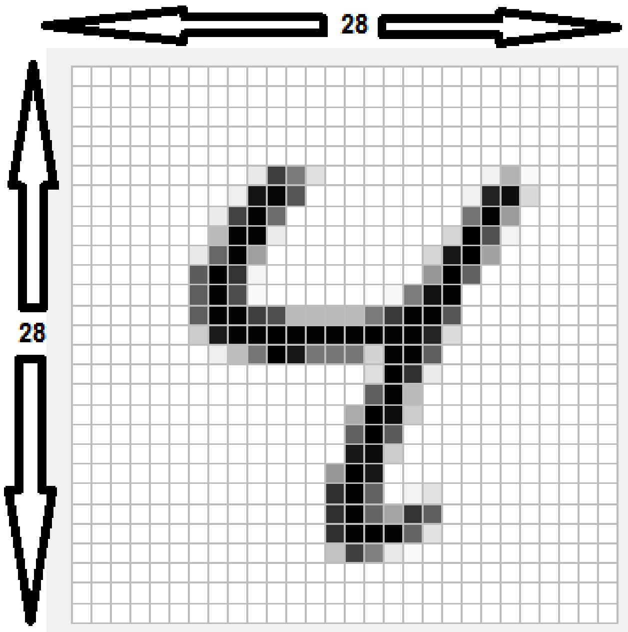
Tensorboard

http://localhost:6006

```
for i in range(total_batch):
        batch_xs, batch_ys = mnist.train.next_batch(batch_size)
        feed_dict = {X: batch_xs, Y: batch_ys, keep_prob: 0.8}
        c, _, a = sess.run([cost, optimizer, summary_op], feed_dict=feed_dict)
        avg_cost += c / total_batch

    print('Epoch:', '%04d' % (epoch + 1), 'training cost =', '{:.9f}'.format(avg_cost))
    train_summary_writer.add_summary(a, early_stopped)
    val_accuracy, summaries = sess.run([accuracy, summary_op], feed_dict={X:
mnist.validation.images, Y: mnist.validation.labels, keep_prob: 1.0})
    val_summary_writer.add_summary(summaries, early_stopped)
print('Validation Accuracy:', val_accuracy)
    if val_accuracy > max:
        max = val_accuracy
        early_stopped = epoch + 1
        saver.save(sess, checkpoint_prefix, global_step=early_stopped)
```

MNIST data

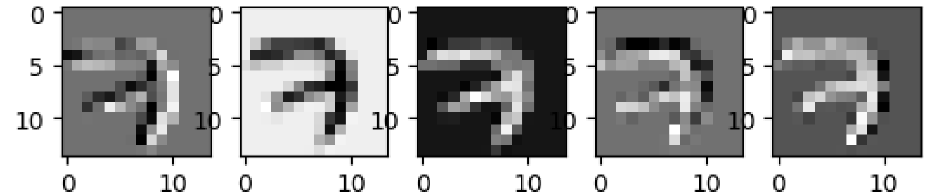



```
# MNIST data image of shape 28 * 28 = 784
X = tf.placeholder(tf.float32, [None, 784])
# 0 - 9 digits recognition = 10 classes
Y = tf.placeholder(tf.float32, [None, nb_classes])
```

Image Input

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

Convolution



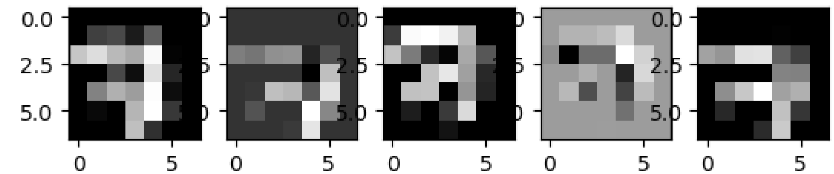
```
import matplotlib.pyplot as plt
import numpy as np
import tensorflow as tf

from tensorflow.examples.tutorials.mnist import input_data

mnist = input_data.read_data_sets("MNIST_data/", one_hot=True)
img = mnist.train.images[0].reshape(28,28)
sess = tf.InteractiveSession()

img = img.reshape(-1,28,28,1)
W1 = tf.Variable(tf.random_normal([3, 3, 1, 5], stddev=0.01))
conv2d = tf.nn.conv2d(img, W1, strides=[1, 2, 2, 1], padding='SAME')
print(conv2d)
sess.run(tf.global_variables_initializer())
conv2d_img = conv2d.eval()
conv2d_img = np.swapaxes(conv2d_img, 0, 3)
for i, one_img in enumerate(conv2d_img):
    plt.subplot(1,5,i+1), plt.imshow(one_img.reshape(14,14), cmap='gray')
plt.show()
```

Pooling




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

mnist = input_data.read_data_sets("MNIST_data/", one_hot=True)
img = mnist.train.images[0].reshape(28,28)
sess = tf.InteractiveSession()

img = img.reshape(-1,28,28,1)
W1 = tf.Variable(tf.random_normal([3, 3, 1, 5], stddev=0.01))
conv2d = tf.nn.conv2d(img, W1, strides=[1, 2, 2, 1], padding='SAME')
pool = tf.nn.max_pool(conv2d, ksize=[1, 2, 2, 1], strides=[1, 2, 2, 1], padding='SAME')
print(pool)
sess.run(tf.global_variables_initializer())
pool_img = pool.eval()
pool_img = np.swapaxes(pool_img, 0, 3)
for i, one_img in enumerate(pool_img):
    plt.subplot(1,5,i+1), plt.imshow(one_img.reshape(7, 7), cmap='gray')
plt.show()
```

CNN for MNIST

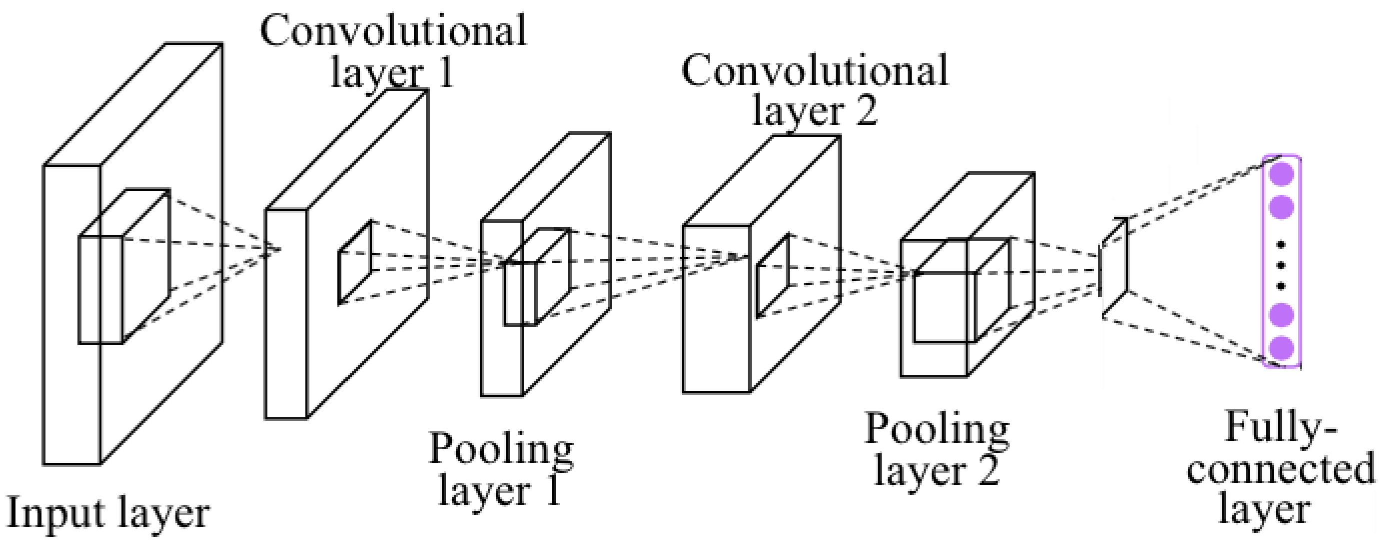


Image Input

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

Convolutional & pooling layer 1, 2

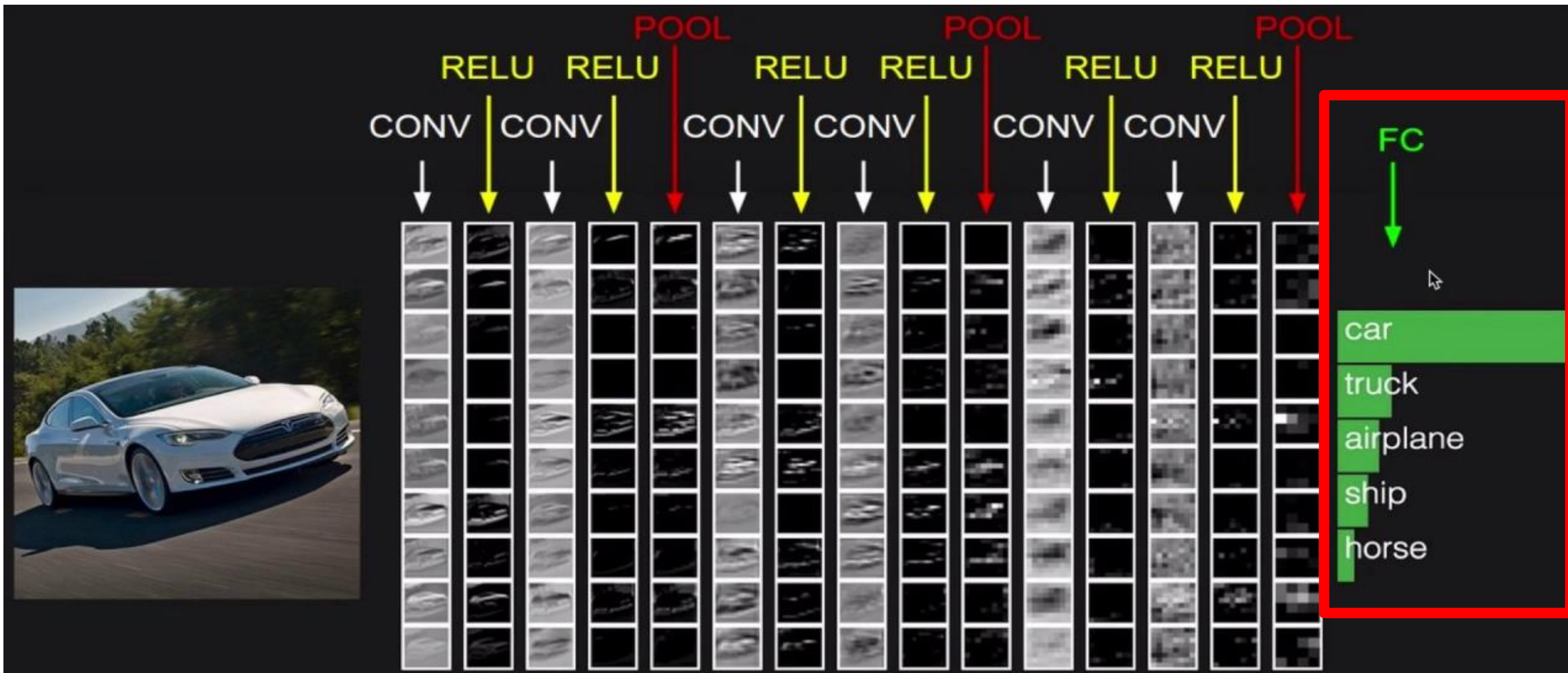
```
# L1 ImgIn shape=(?, 28, 28, 1)
W1 = tf.Variable(tf.random_normal([3, 3, 1, 32], stddev=0.01))
#    Conv      -> (?, 28, 28, 32)
#    Pool      -> (?, 14, 14, 32)
L1 = tf.nn.conv2d(X_img, W1, strides=[1, 1, 1, 1], padding='SAME')
L1 = tf.nn.relu(L1)
L1 = tf.nn.max_pool(L1, ksize=[1, 2, 2, 1], strides=[1, 2, 2, 1], padding='SAME')
L1 = tf.nn.dropout(L1, keep_prob=keep_prob)
```

```
# L2 ImgIn shape=(?, 14, 14, 32)
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)
L2 = tf.nn.max_pool(L2, ksize=[1, 2, 2, 1], strides=[1, 2, 2, 1], padding='SAME’)
L2 = tf.nn.dropout(L2, keep_prob=keep_prob)
L2_flat = tf.reshape(L2, [-1, 7 * 7 * 64])
```

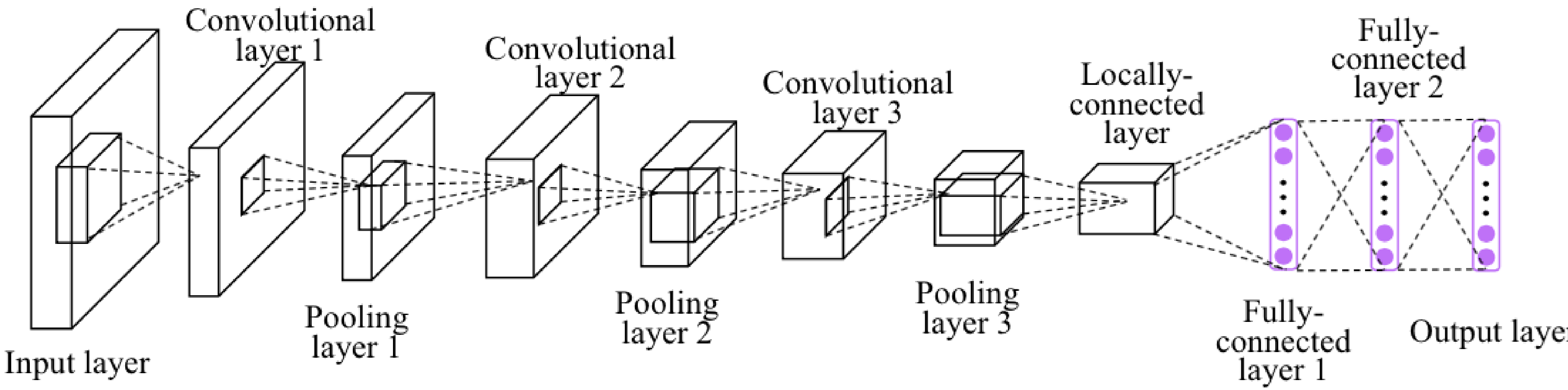
Fully connected layer

```
W3 = tf.get_variable("W3", shape=[7 * 7 * 64, 10], initializer=tf.contrib.layers.xavier_initializer())
b = tf.Variable(tf.random_normal([10]))
hypothesis = tf.nn.xw_plus_b(L2_flat, W3, b, name="hypothesis")

cost = tf.reduce_mean(tf.nn.softmax_cross_entropy_with_logits(logits=hypothesis, labels=Y))
```



Deeper!


Convolutional
layer 1
Convolutional
layer 2
Convolutional
layer 3
Locally-
connected
layer
Fully-
connected
layer 2
Input layer
Pooling
layer 1
Pooling
layer 2
Pooling
layer 3
Fully-
connected
layer 1
Output layer

Q&A