2018.10.13

NEURAL NETWORK (2)

3기 천용희

Growth Hackers

CONTENTS

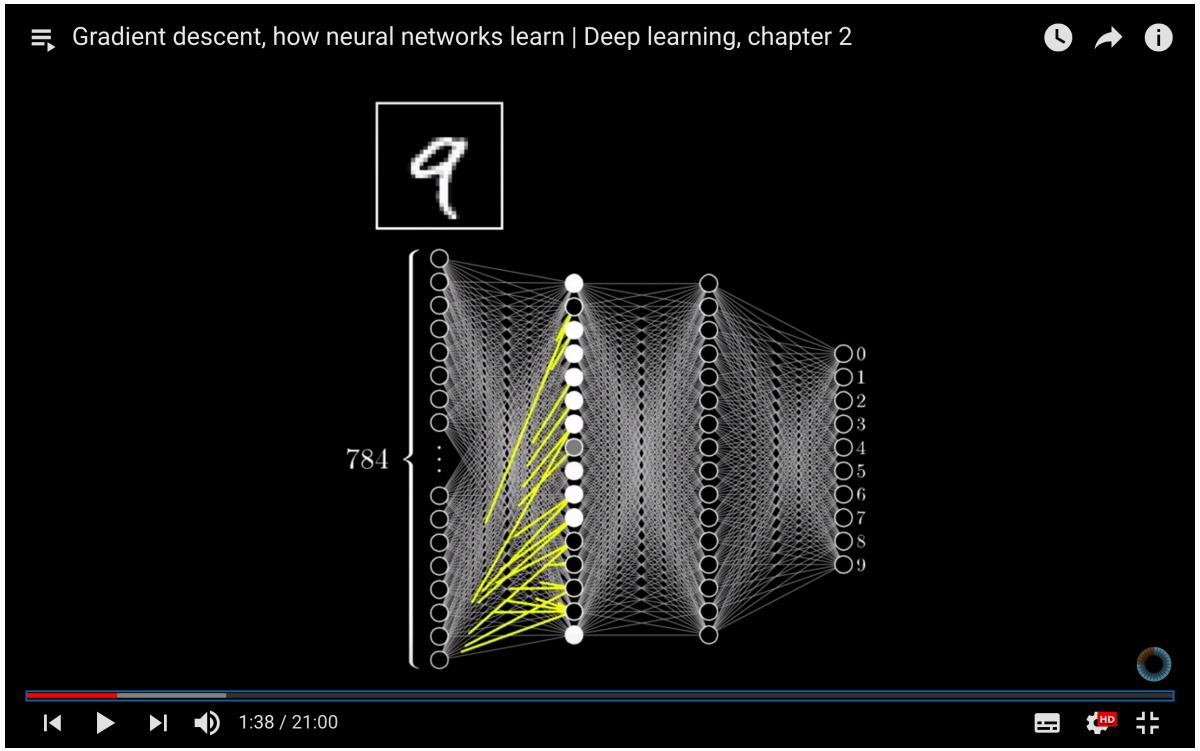
Chapter 1

머신러닝 복습 및 딥러닝 이해

Growth Hackers

Chapter 1.1 I What is a Neural Network?

3Blue1Brown Video

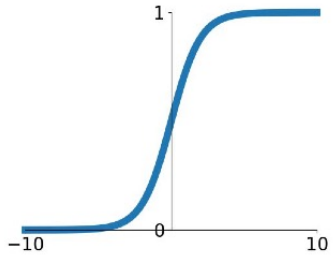


Growth Hackers

Chapter 1.2 I 활성화 함수의 종류

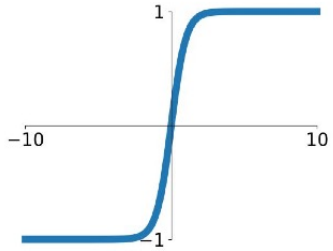
Sigmoid

$$\sigma(x) = \frac{1}{1+e^{-x}}$$



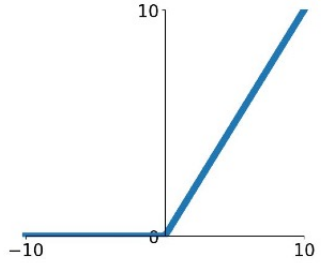
tanh

$$\tanh(x)$$



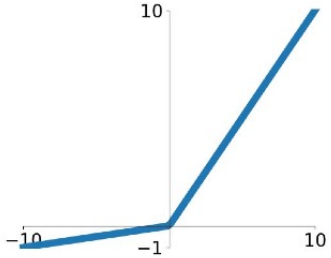
ReLU

$$\max(0, x)$$



Leaky ReLU

$$\max(0.1x, x)$$

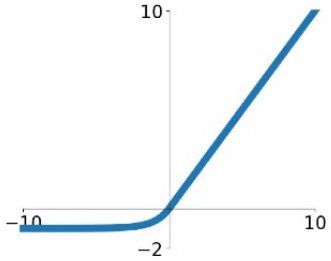


Maxout

$$\max(w_1^T x + b_1, w_2^T x + b_2)$$

ELU

$$\begin{cases} x & x \geq 0 \\ \alpha(e^x - 1) & x < 0 \end{cases}$$



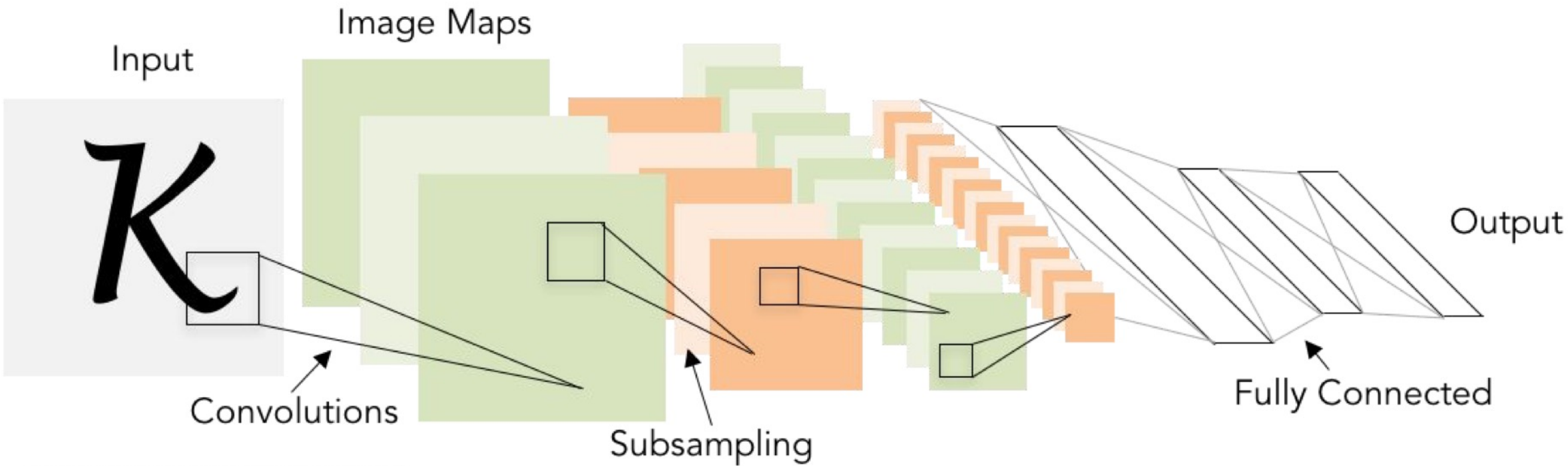
Growth Hackers

Chapter 2

합성곱 신경망
Convolutional Neural Networks

Growth Hackers

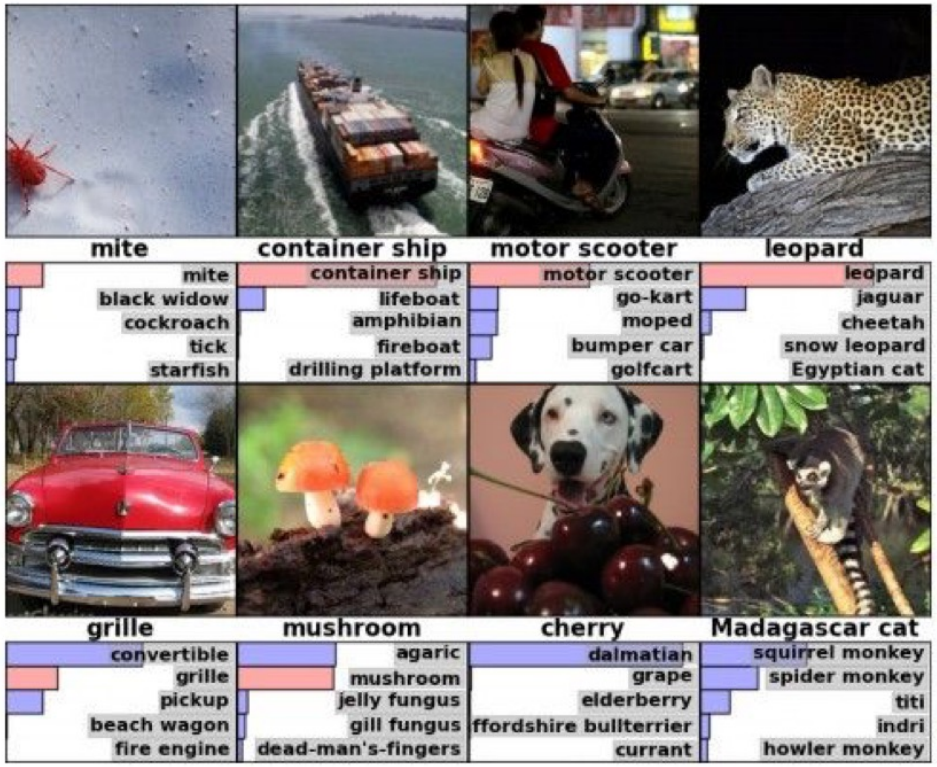
Chapter 2.1 I CNN의 아키텍쳐



Growth Hackers

Chapter 2.2 I CNN 활용 예시

Classification

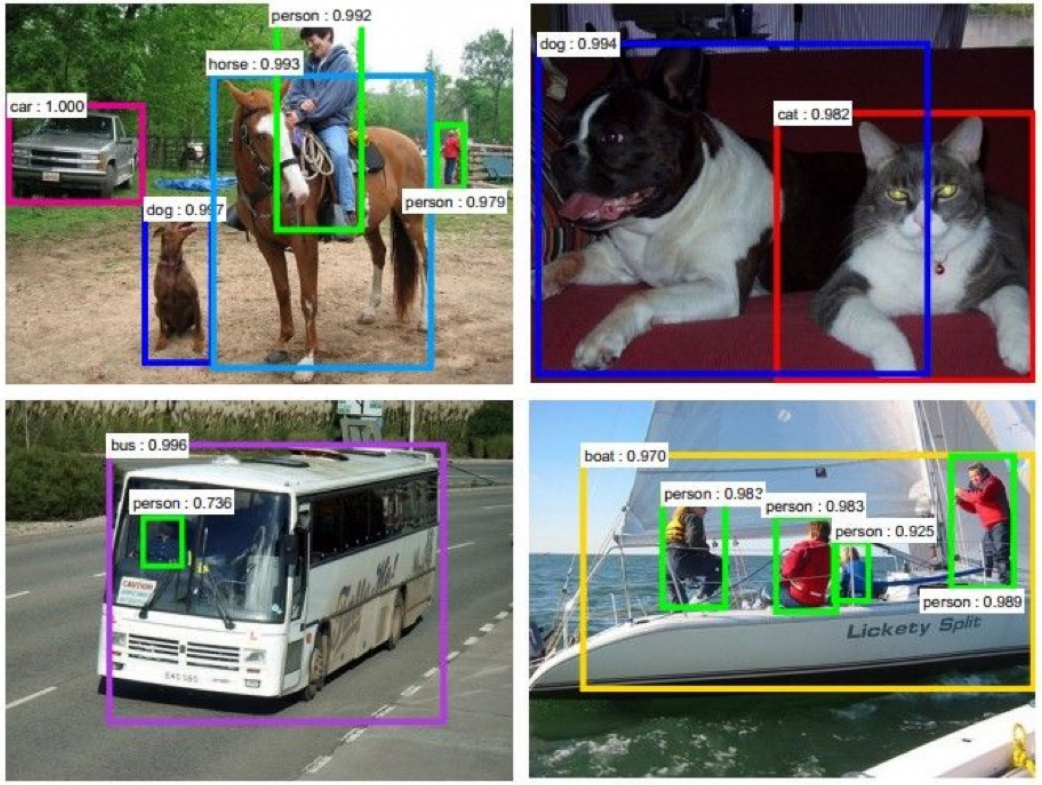


Clustering

Growth Hackers

Chapter 2.2 I CNN 활용 예시

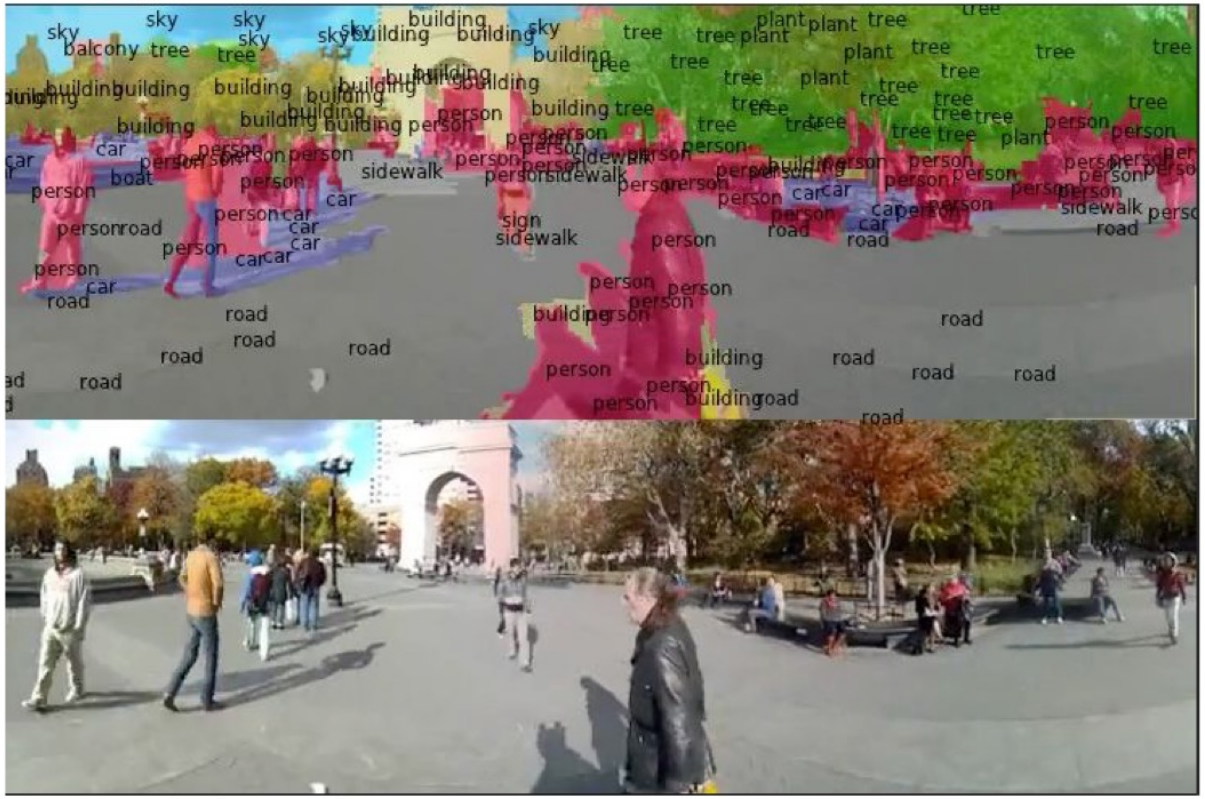
Detection



Figures copyright Shaoqing Ren, Kaiming He, Ross Girschick, Jian Sun, 2015. Reproduced with permission.

[Faster R-CNN: Ren, He, Girshick, Sun 2015]

Segmentation



Figures copyright Clement Farabet, 2012. Reproduced with permission.

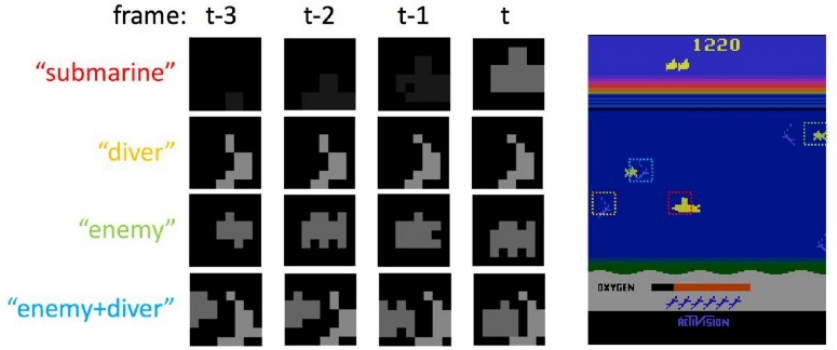
[Farabet et al., 2012]

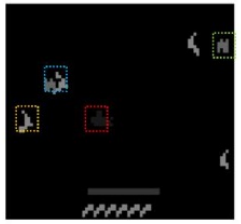
Growth Hackers

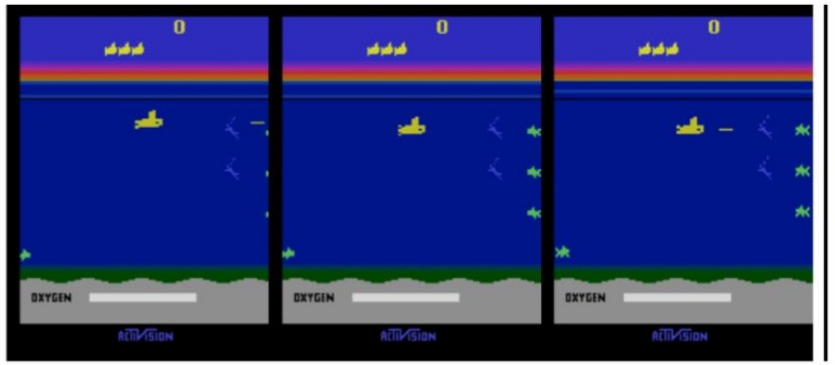
Chapter 2.2 I CNN 활용 예시

Images are examples of pose estimation, not actually from Toshev & Szegedy 2014. Copyright Lane McIntosh.

[Toshev, Szegedy 2014]





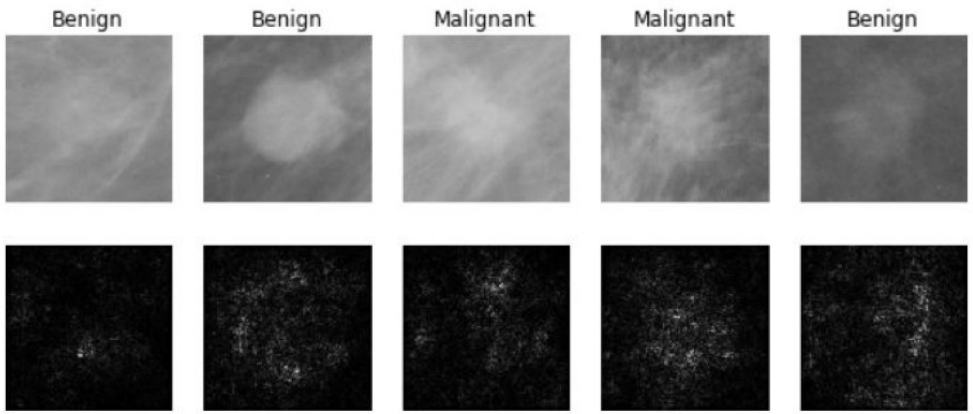


[Guo et al. 2014]

Figures copyright Xiaoxiao Guo, Satinder Singh, Honglak Lee, Richard Lewis, and Xiaoshi Wang, 2014. Reproduced with permission.

Growth Hackers

Chapter 2.2 I CNN 활용 예시



[Levy et al. 2016]

Figure copyright Levy et al. 2016.
Reproduced with permission.

[Dieleman et al. 2014]

From left to right: public domain by NASA, usage permitted by ESA/Hubble, public domain by NASA, and public domain.

[Sermanet et al. 2011]
[Ciresan et al.]

Photos by Lane McIntosh.
Copyright CS231n 2017.

Growth Hackers

Chapter 2.2 I CNN 활용 예시

This image by Christin Khan is in the public domain and originally came from the U.S. NOAA.

Whale recognition, Kaggle Challenge

Photo and figure by Lane McIntosh; not actual example from Mnih and Hinton, 2010 paper.

Mnih and Hinton, 2010

Chapter 2.2 I CNN 활용 예시

No errors

A white teddy bear sitting in the grass

Minor errors

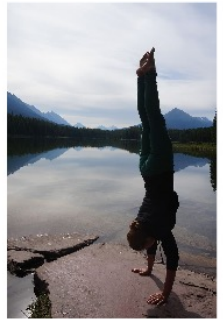
A man in a baseball uniform throwing a ball

Somewhat related

A woman is holding a cat in her hand

A man riding a wave on top of a surfboard

A cat sitting on a suitcase on the floor



A woman standing on a beach holding a surfboard

Image Captioning

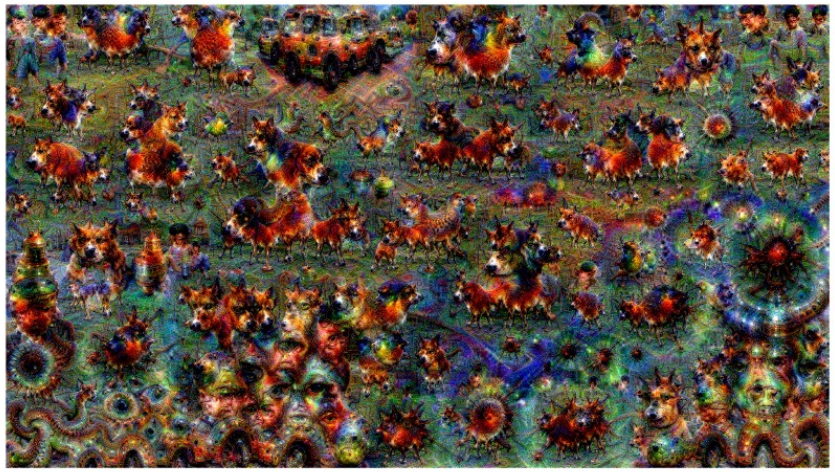
[Vinyals et al., 2015]
[Karpathy and Fei-Fei, 2015]

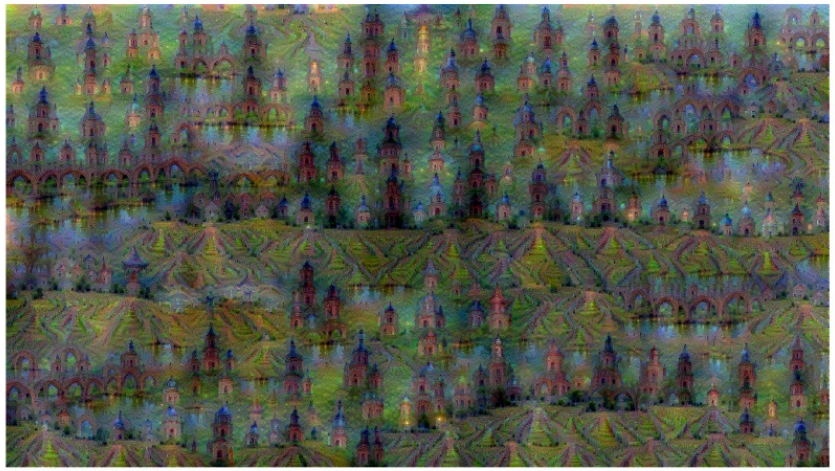
All images are CC0 Public domain:
https://pixabay.com/en/luggage-antique-cat-1643010/
https://pixabay.com/en/teddy-plush-bears-cute-teddy-bear-1623436/
https://pixabay.com/en/surf-wave-summer-sport-litoral-1668716/
https://pixabay.com/en/woman-female-model-portrait-adult-983967/
https://pixabay.com/en/handstand-lake-meditation-496008/
https://pixabay.com/en/baseball-player-shortstop-infield-1045263/

Captions generated by Justin Johnson using Neuraltalk2

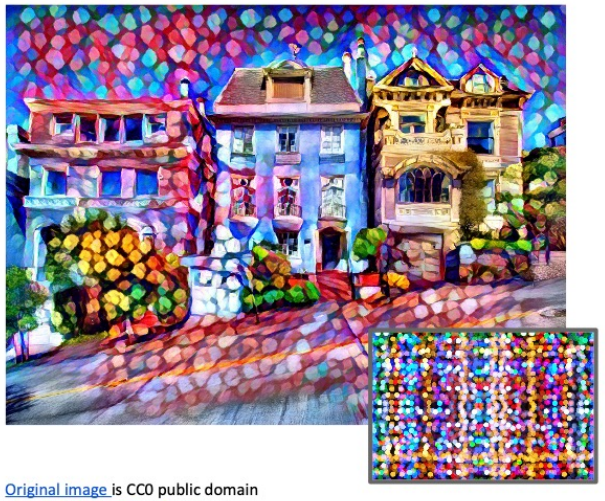
Growth Hackers

Chapter 2.2 I CNN 활용 예시





Figures copyright Justin Johnson, 2015. Reproduced with permission. Generated using the Inceptionism approach from a blog post by Google Research.

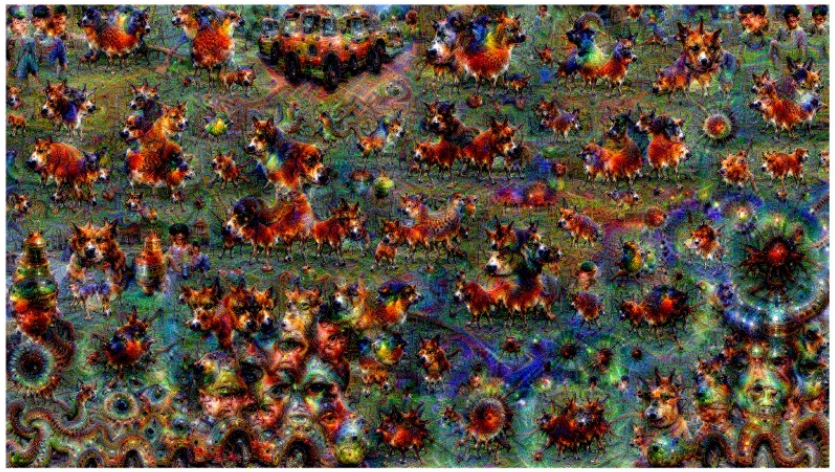


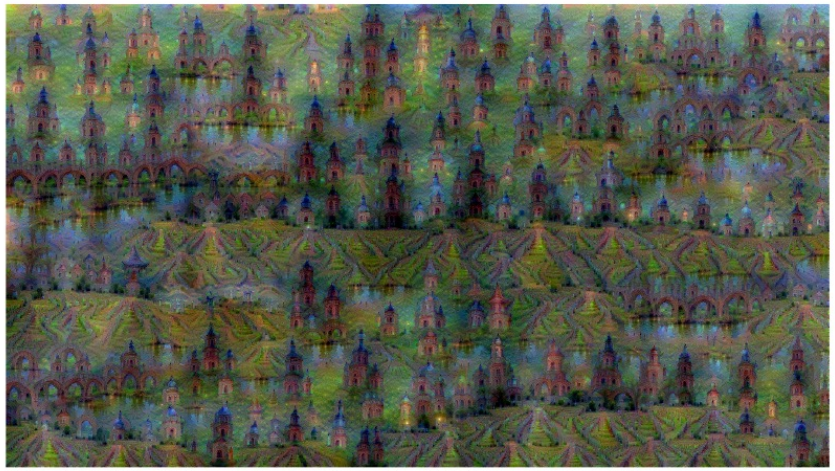
Original image is CC0 public domain
Starry Night and Tree Roots by Van Gogh are in the public domain
Bokeh image is in the public domain
Stylized images copyright Justin Johnson, 2017;
reproduced with permission

Gatys et al, "Image Style Transfer using Convolutional Neural Networks", CVPR 2016
Gatys et al, "Controlling Perceptual Factors in Neural Style Transfer", CVPR 2017

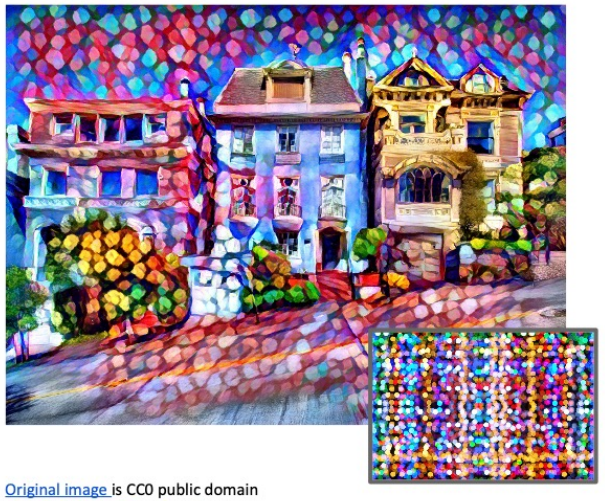
Growth Hackers

Chapter 2.2 I CNN 활용 예시





Figures copyright Justin Johnson, 2015. Reproduced with permission. Generated using the Inceptionism approach from a blog post by Google Research.



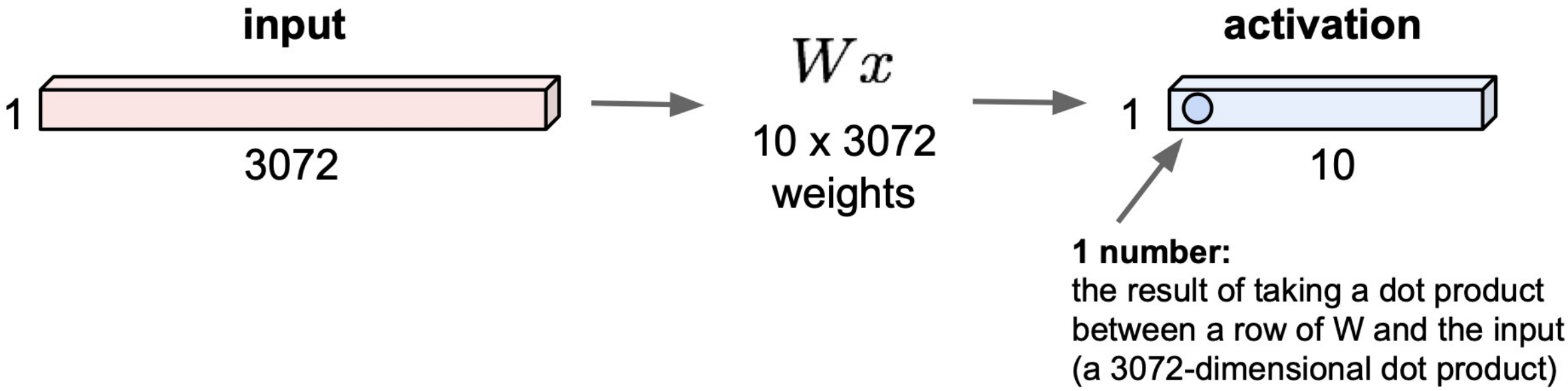
Original image is CC0 public domain
Starry Night and Tree Roots by Van Gogh are in the public domain
Bokeh image is in the public domain
Stylized images copyright Justin Johnson, 2017;
reproduced with permission

Gatys et al, "Image Style Transfer using Convolutional Neural Networks", CVPR 2016
Gatys et al, "Controlling Perceptual Factors in Neural Style Transfer", CVPR 2017

Growth Hackers

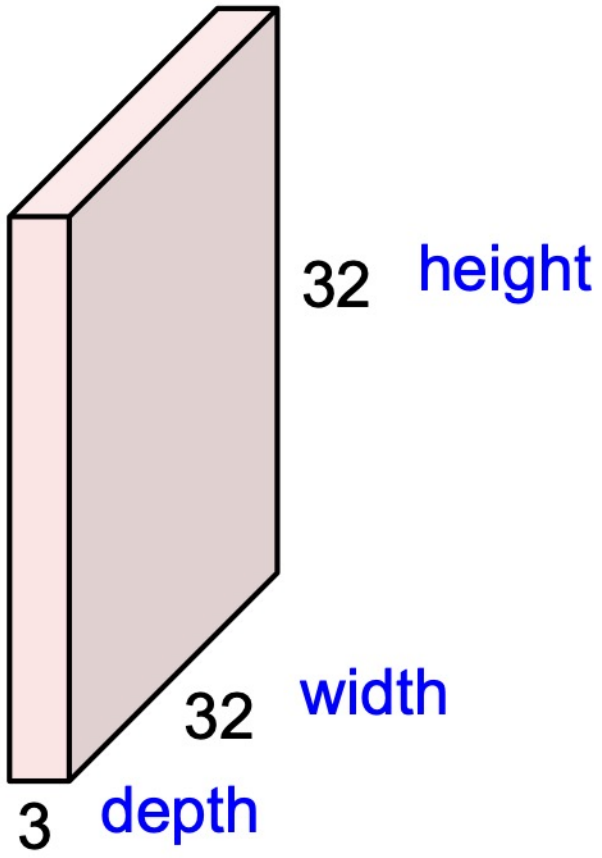
Chapter 2.3.1 l Fully Connected Layer

32x32x3 image -> stretch to 3072 x 1



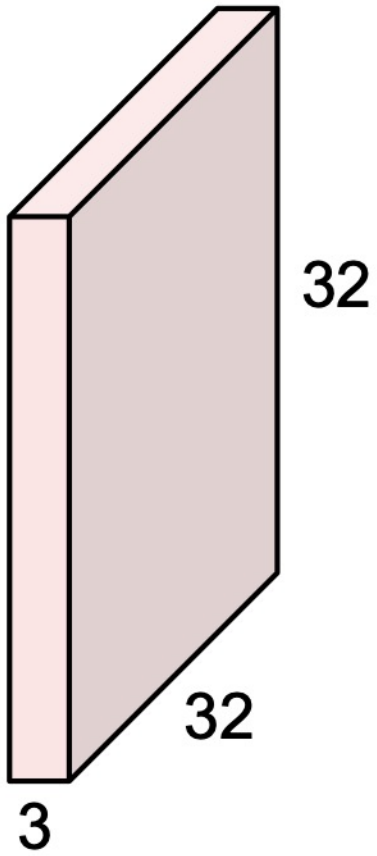
Chapter 2.3.2 l Convolutional Layer

32x32x3 image -> preserve spatial structure

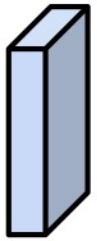


Chapter 2.3.2 l Convolutional Layer

32x32x3 image

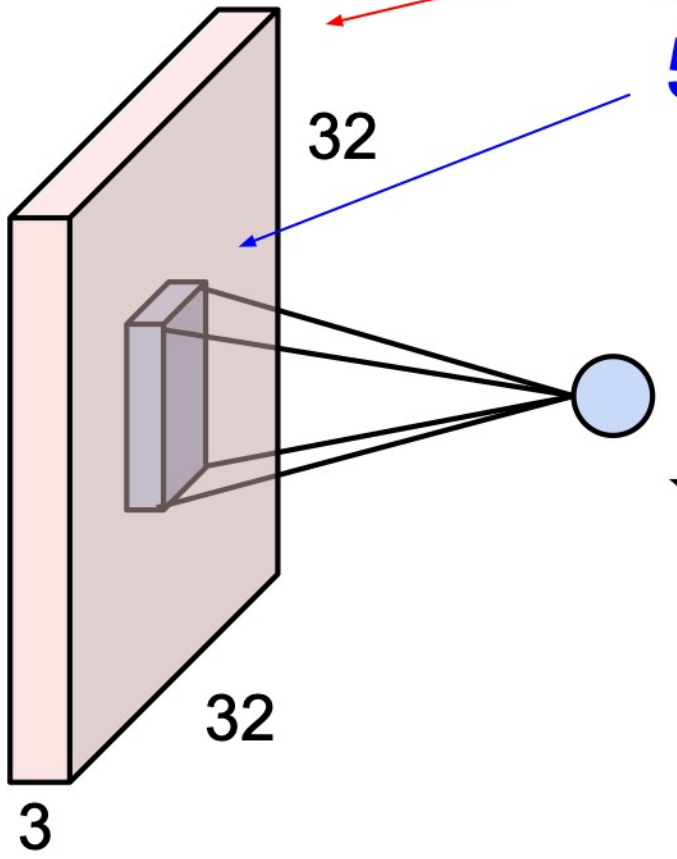


5x5x3 filter



Convolve the filter with the image
i.e. “slide over the image spatially,
computing dot products”

Chapter 2.3.2 l Convolutional Layer



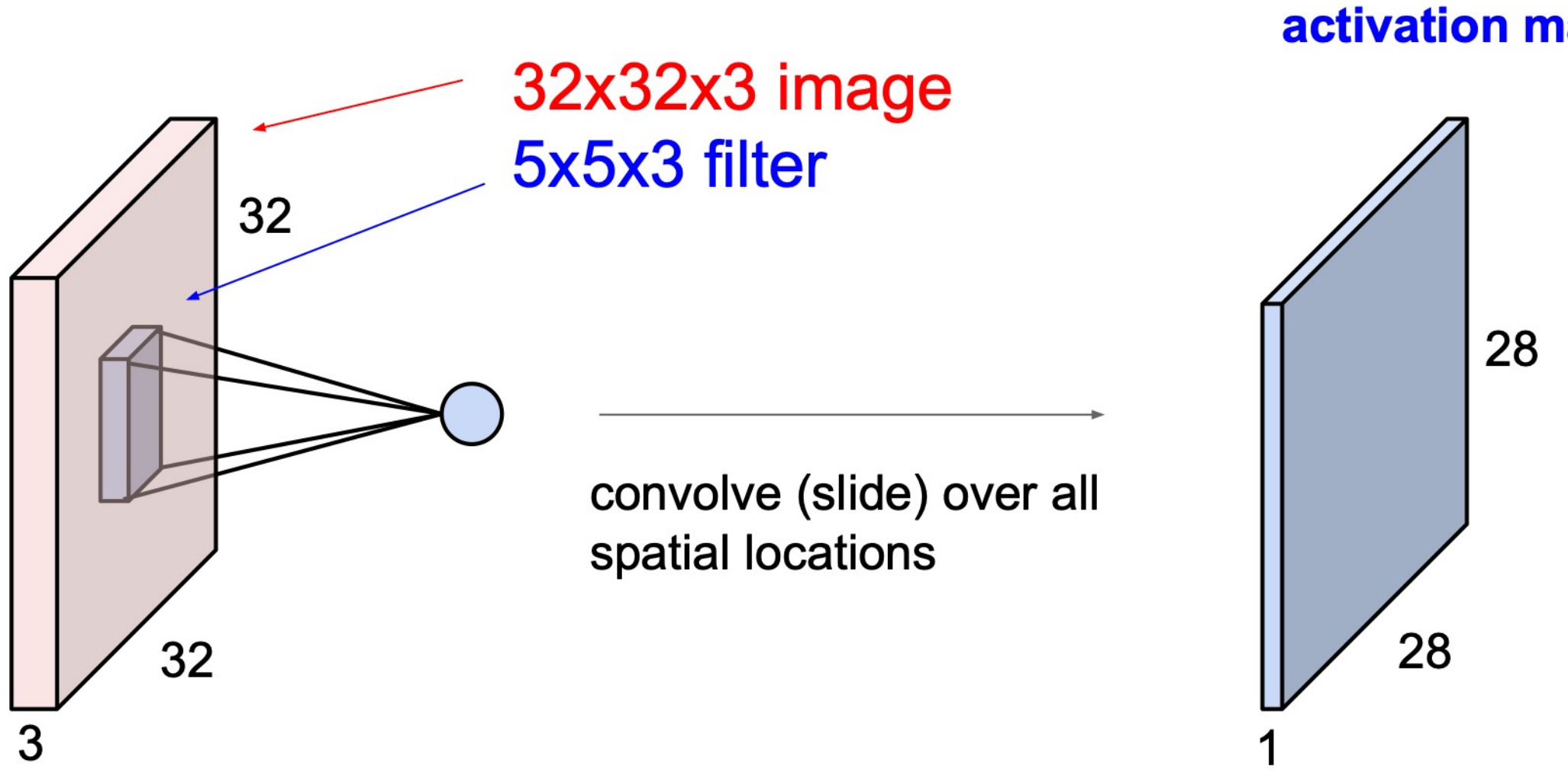
32x32x3 image

5x5x3 filter w

1 number:
the result of taking a dot product between the filter and a small 5x5x3 chunk of the image
(i.e. 5*5*3 = 75-dimensional dot product + bias)

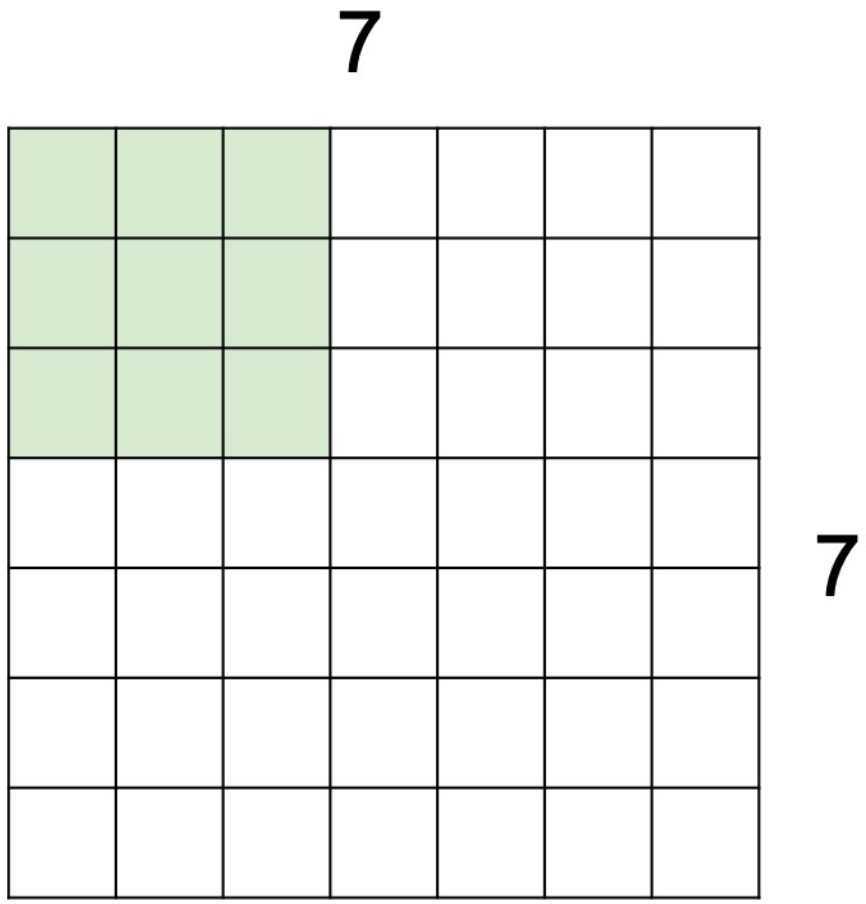
$$w^T x + b$$

Chapter 2.3.3 l Activation Map



Chapter 2.3.3.1 l Activation Map in Toy Example

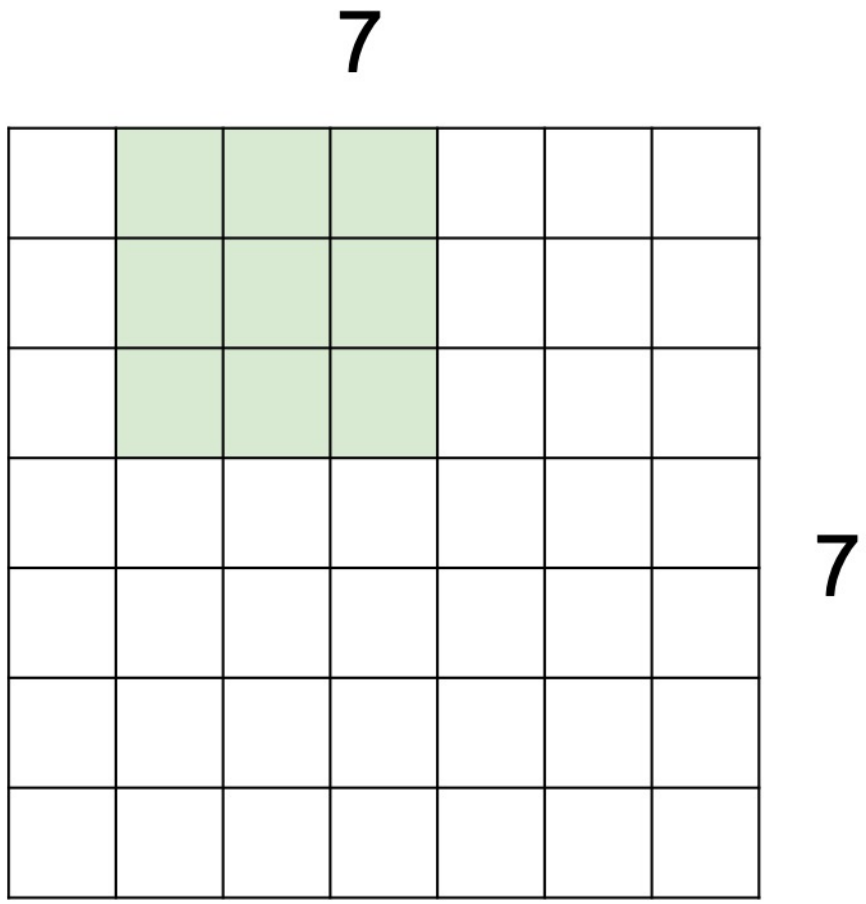
A closer look at spatial dimensions:



7x7 input (spatially)
assume 3x3 filter

Chapter 2.3.3.1 l Activation Map in Toy Example

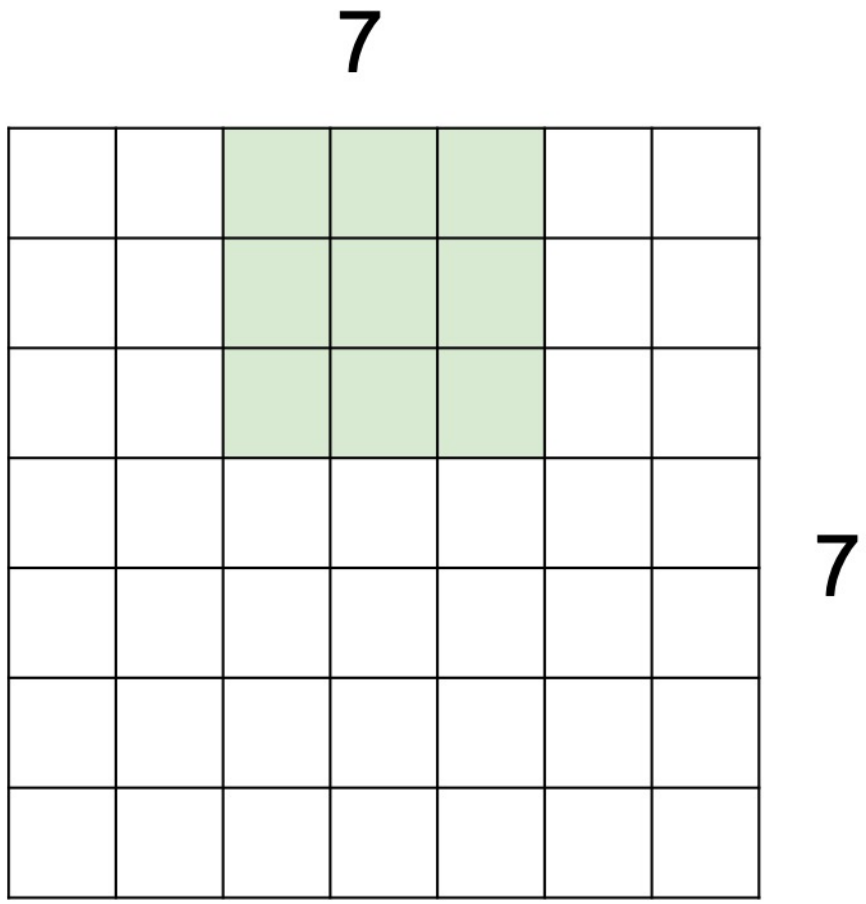
A closer look at spatial dimensions:



7x7 input (spatially)
assume 3x3 filter

Chapter 2.3.3.1 l Activation Map in Toy Example

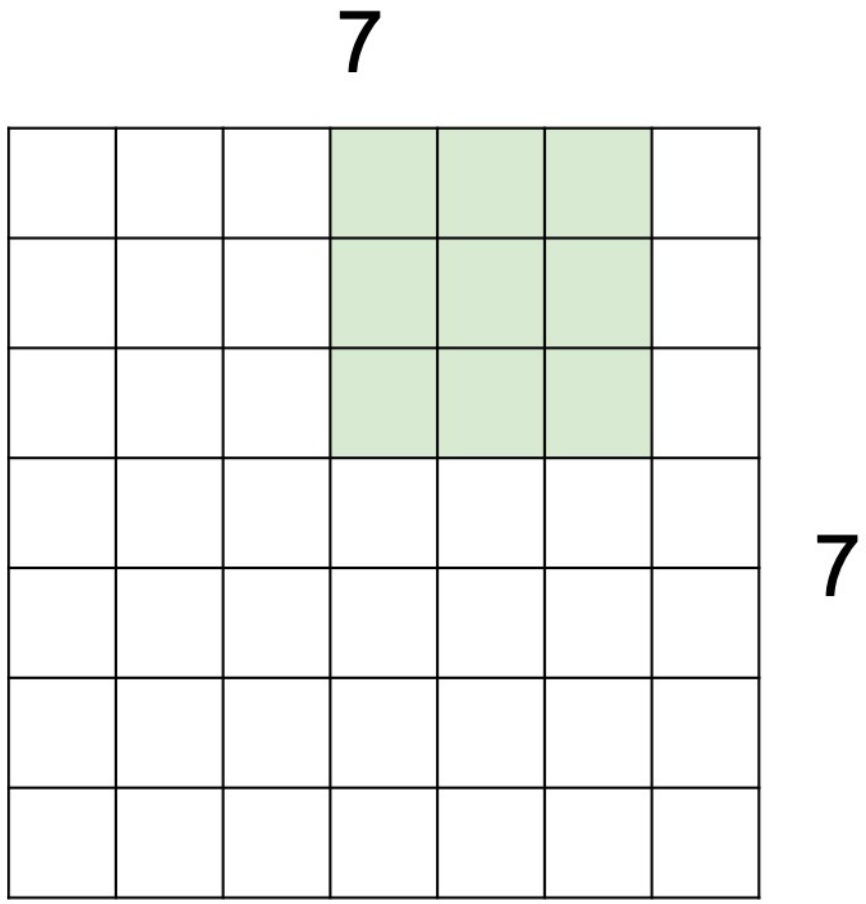
A closer look at spatial dimensions:



7x7 input (spatially)
assume 3x3 filter

Chapter 2.3.3.1 l Activation Map in Toy Example

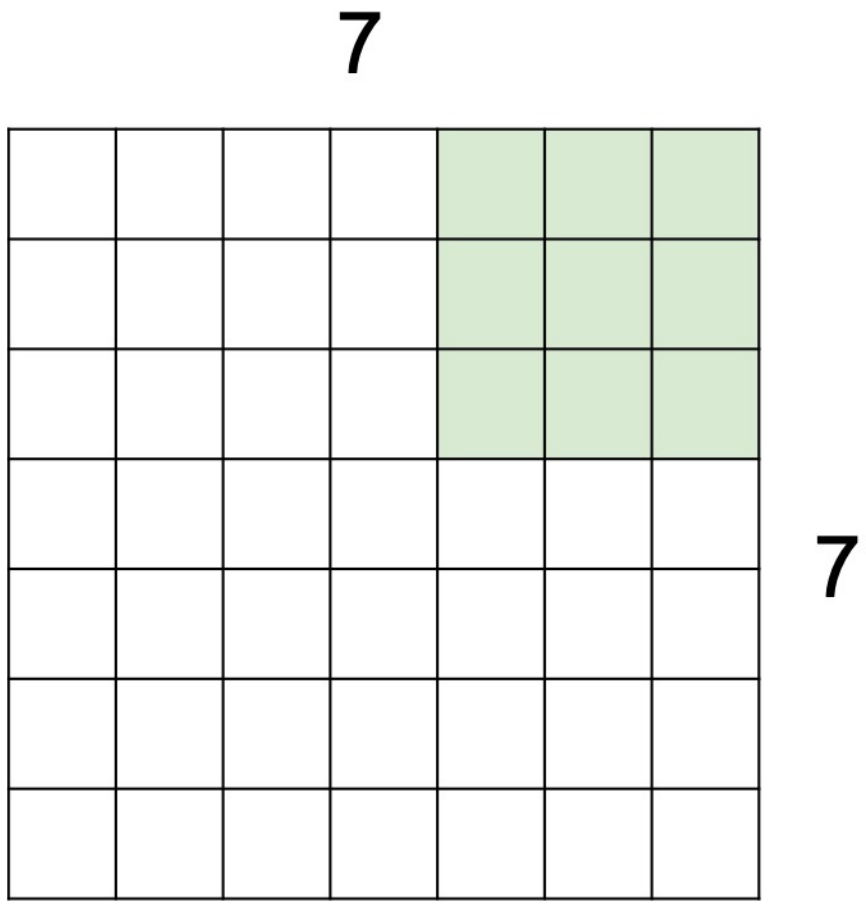
A closer look at spatial dimensions:



7x7 input (spatially)
assume 3x3 filter

Chapter 2.3.3.1 | Activation Map in Toy Example

A closer look at spatial dimensions:

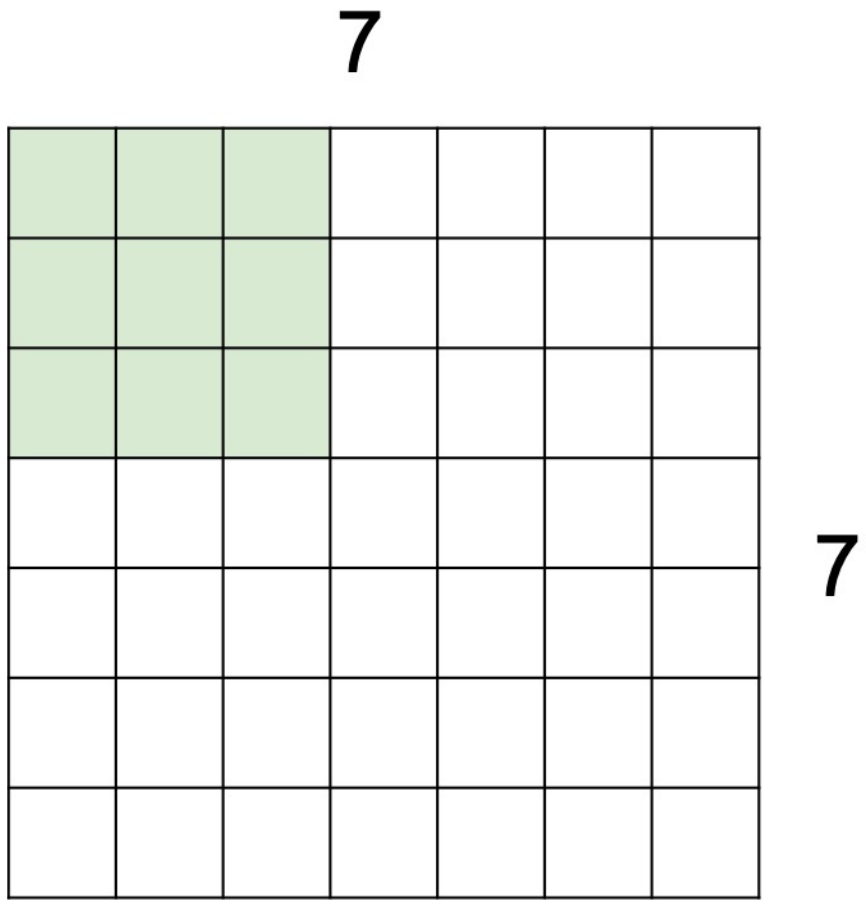


7x7 input (spatially)
assume 3x3 filter

=> 5x5 output

Chapter 2.3.3.1 l Activation Map in Toy Example

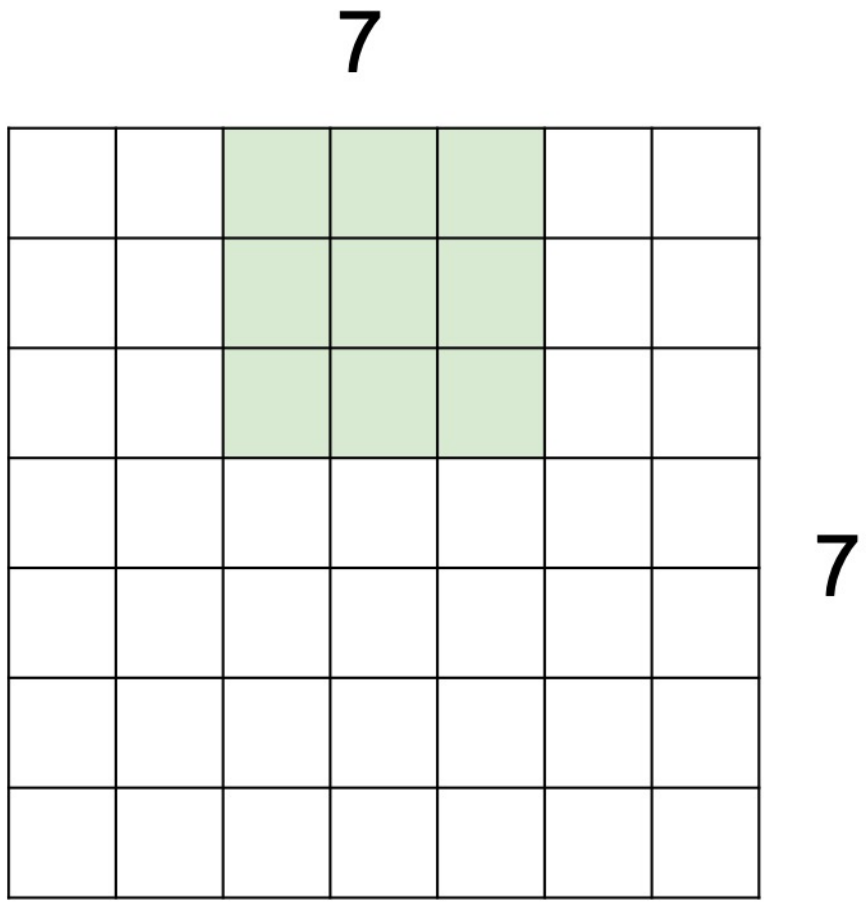
A closer look at spatial dimensions:



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

Chapter 2.3.3.1 l Activation Map in Toy Example

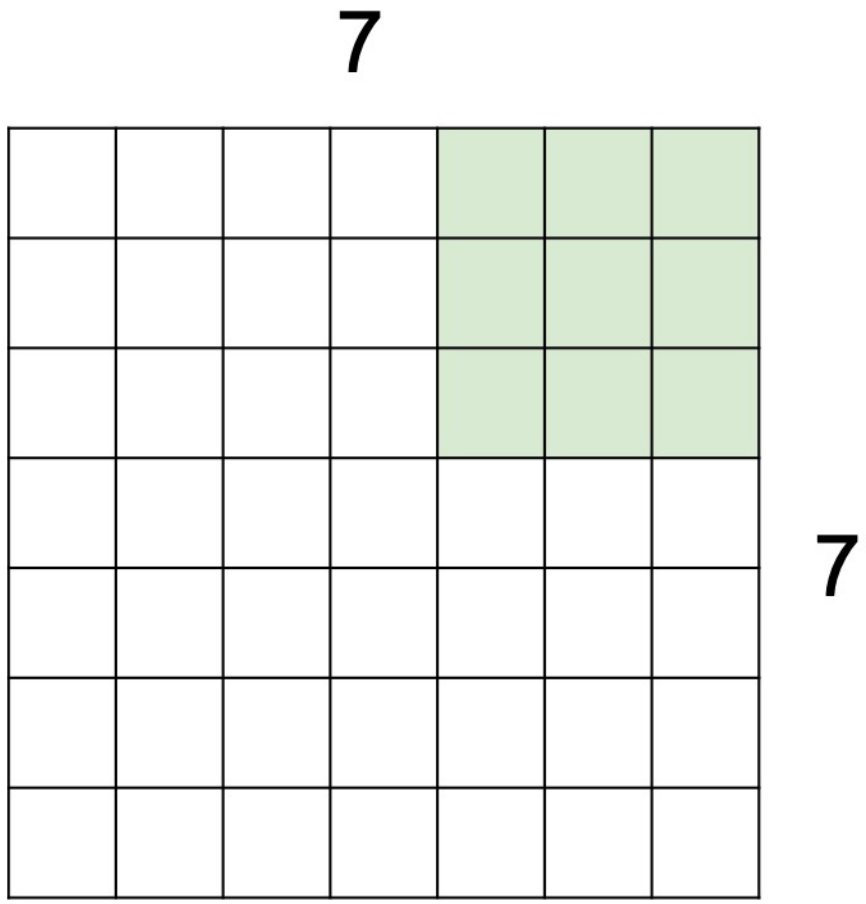
A closer look at spatial dimensions:



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

Chapter 2.3.3.1 | Activation Map in Toy Example

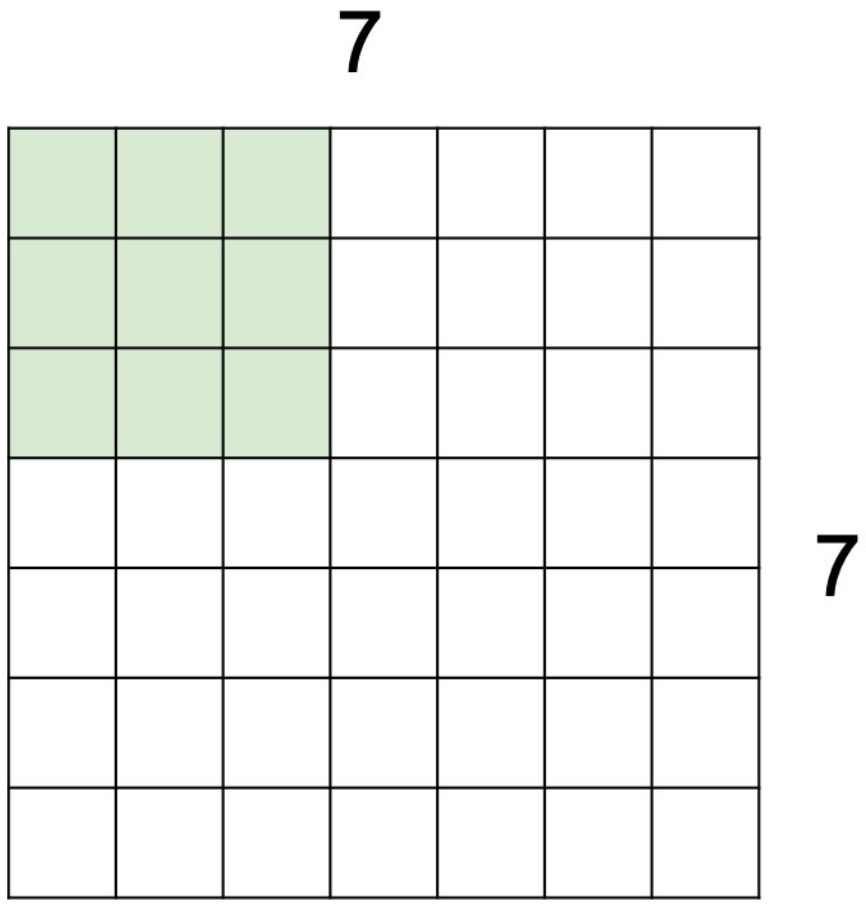
A closer look at spatial dimensions:



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

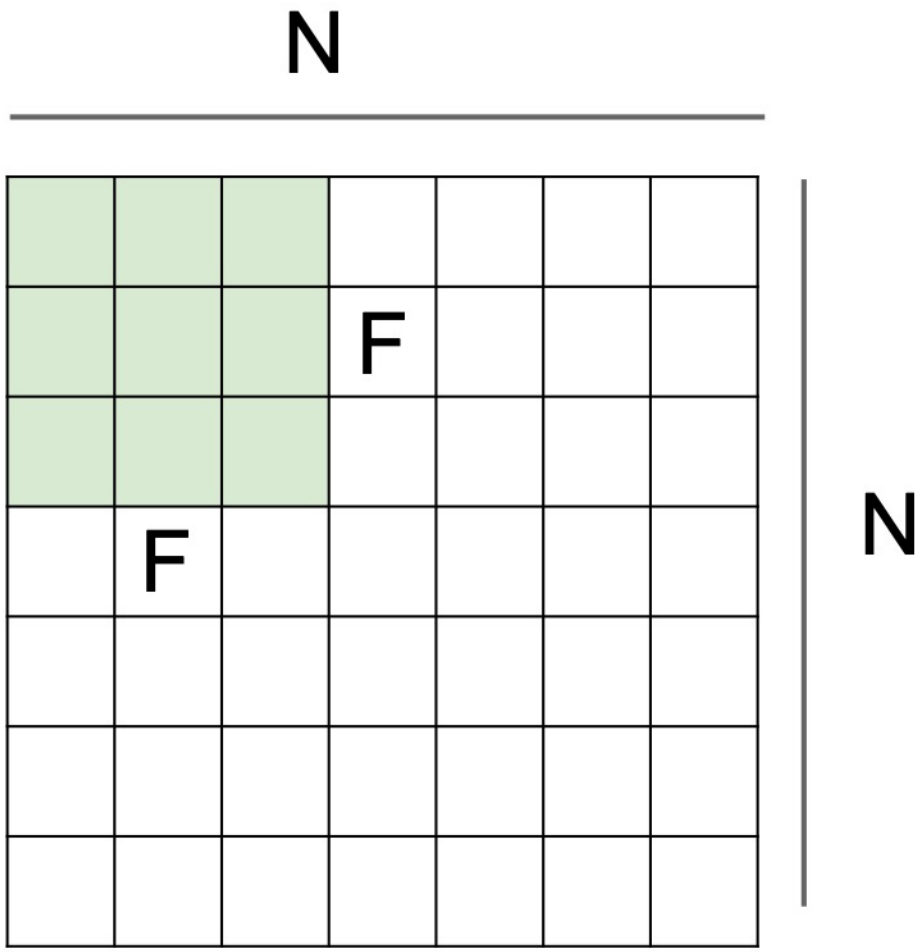
Chapter 2.3.3.1 l Activation Map in Toy Example

A closer look at spatial dimensions:



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

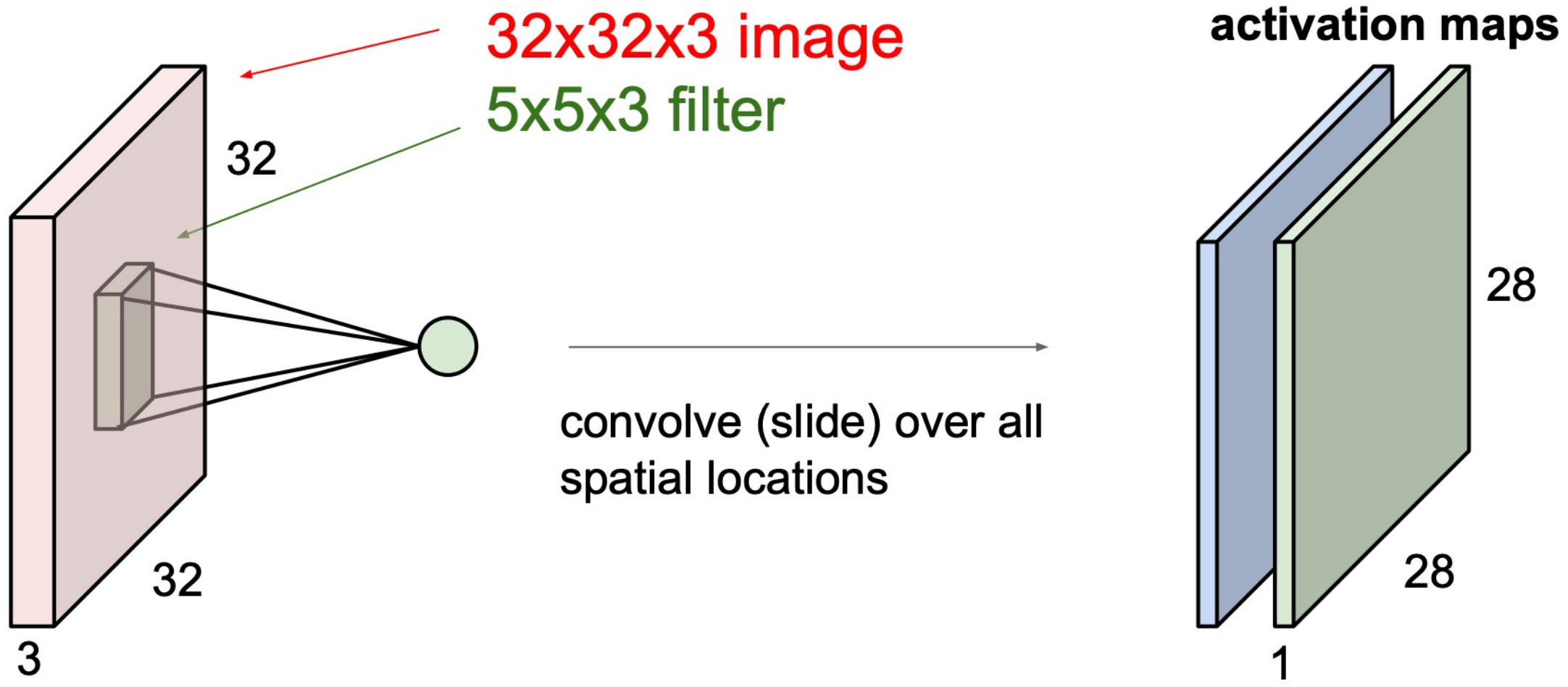
Chapter 2.3.3.1 I Activation Map in Toy Example



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

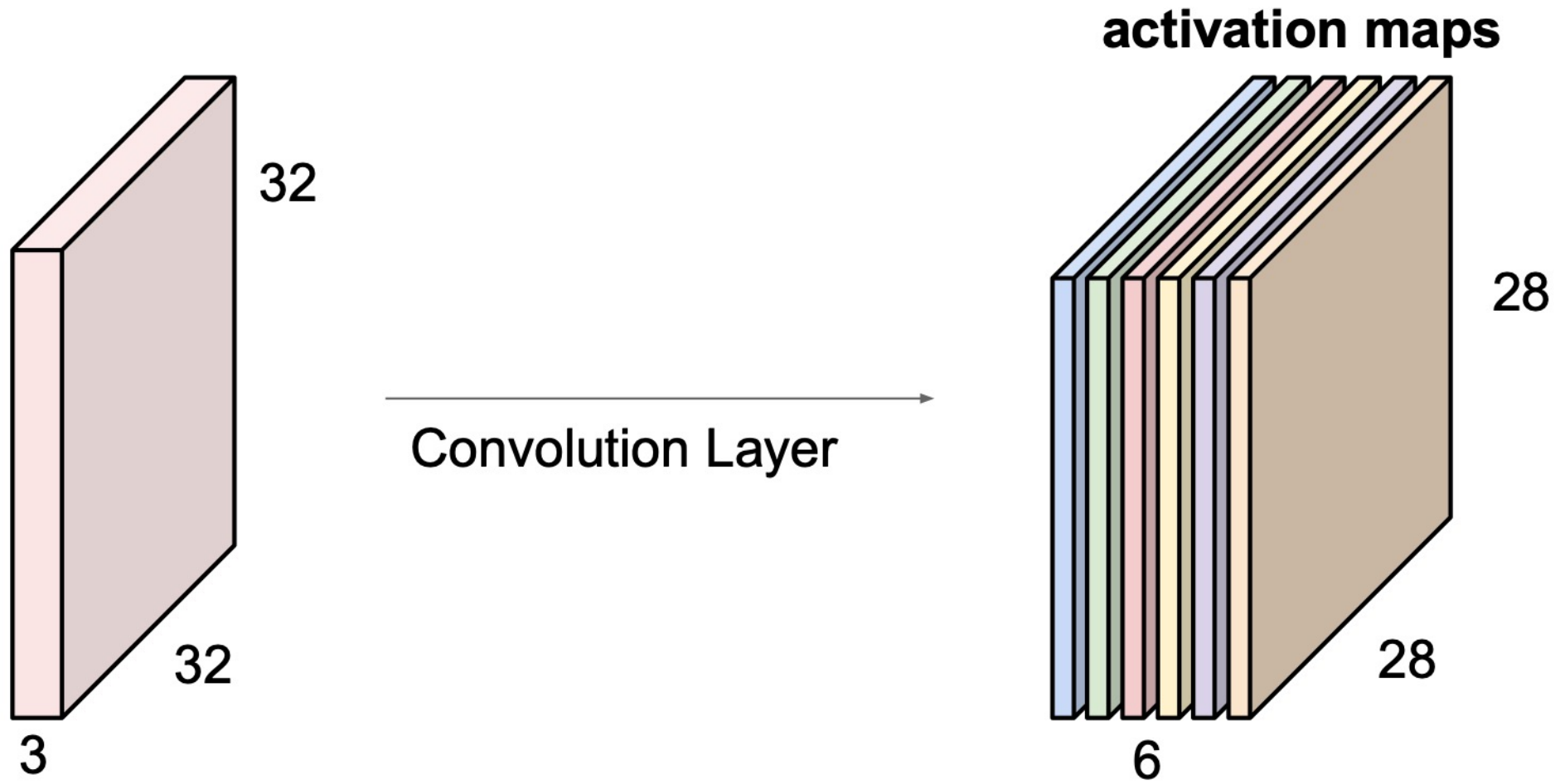
e.g. N = 7, F = 3:
stride 1 => (7 - 3)/1 + 1 = 5
stride 2 => (7 - 3)/2 + 1 = 3
stride 3 => (7 - 3)/3 + 1 = 2.33 :\

Chapter 2.3.3 l Activation Map



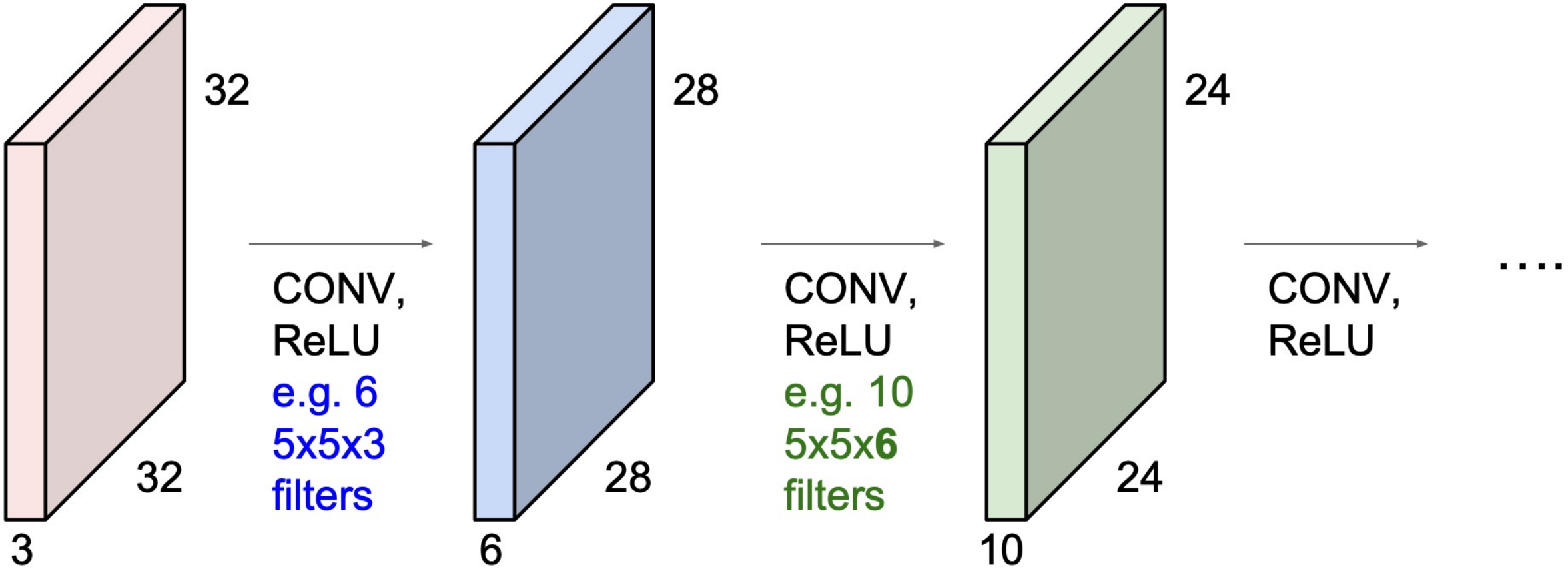
Growth Hackers

Chapter 2.3.3 l Activation Map



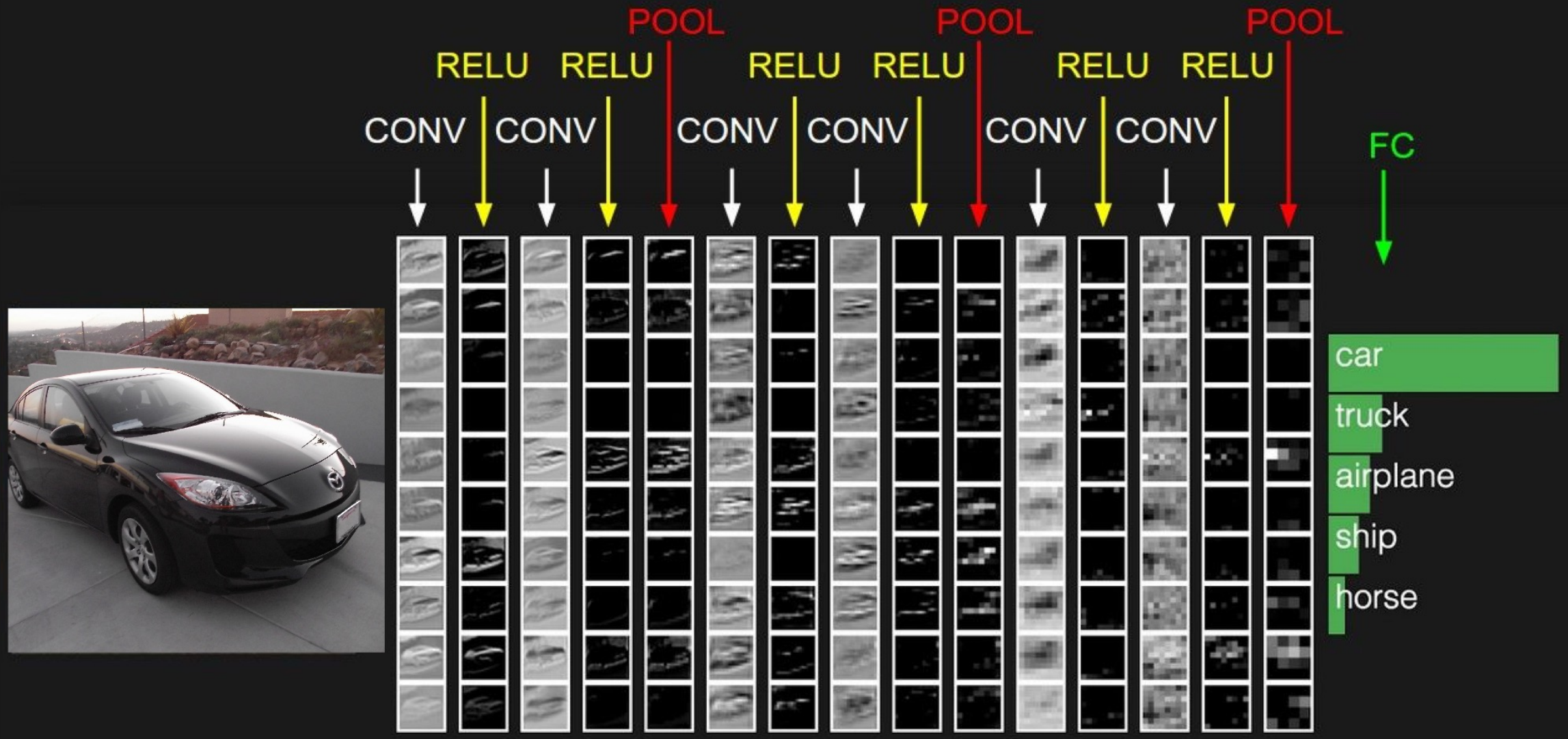
Growth Hackers

Chapter 2.4 I Convnet 구조

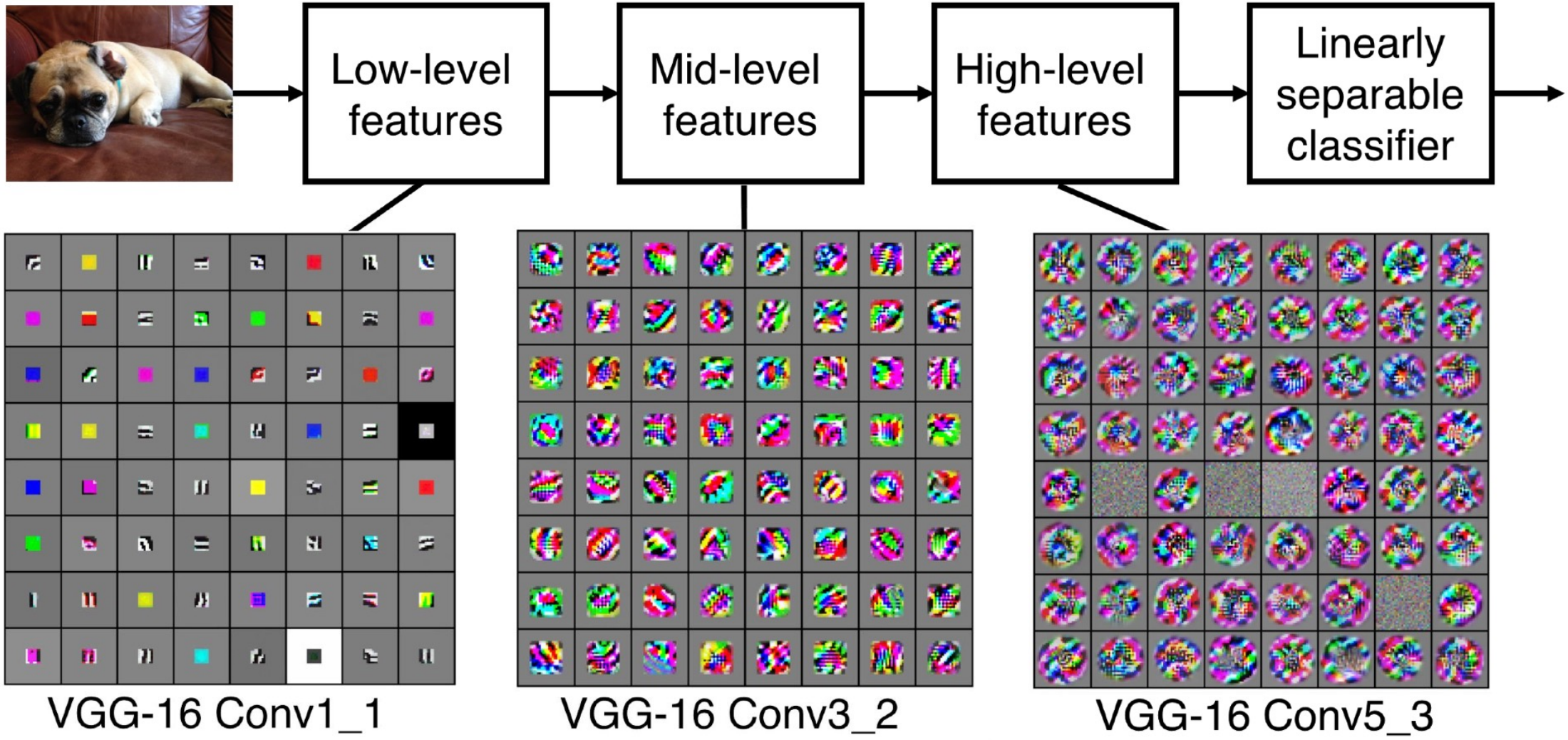


Growth Hackers

Chapter 2.4 I Convnet 구조

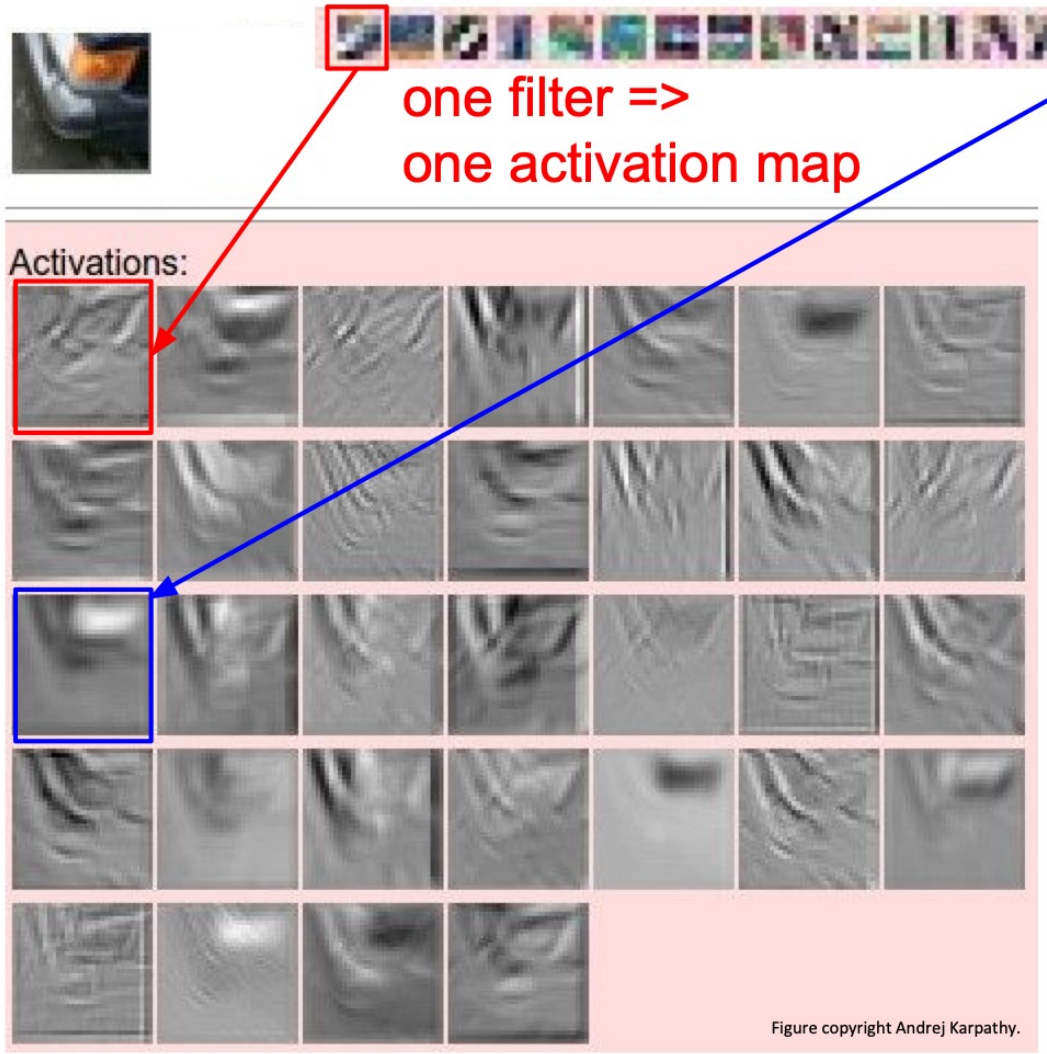


Chapter 2.4 I Convnet 구조



Growth Hackers

Chapter 2.5 I Activation Preview



example 5x5 filters
(32 total)

We call the layer convolutional because it is related to convolution of two signals:

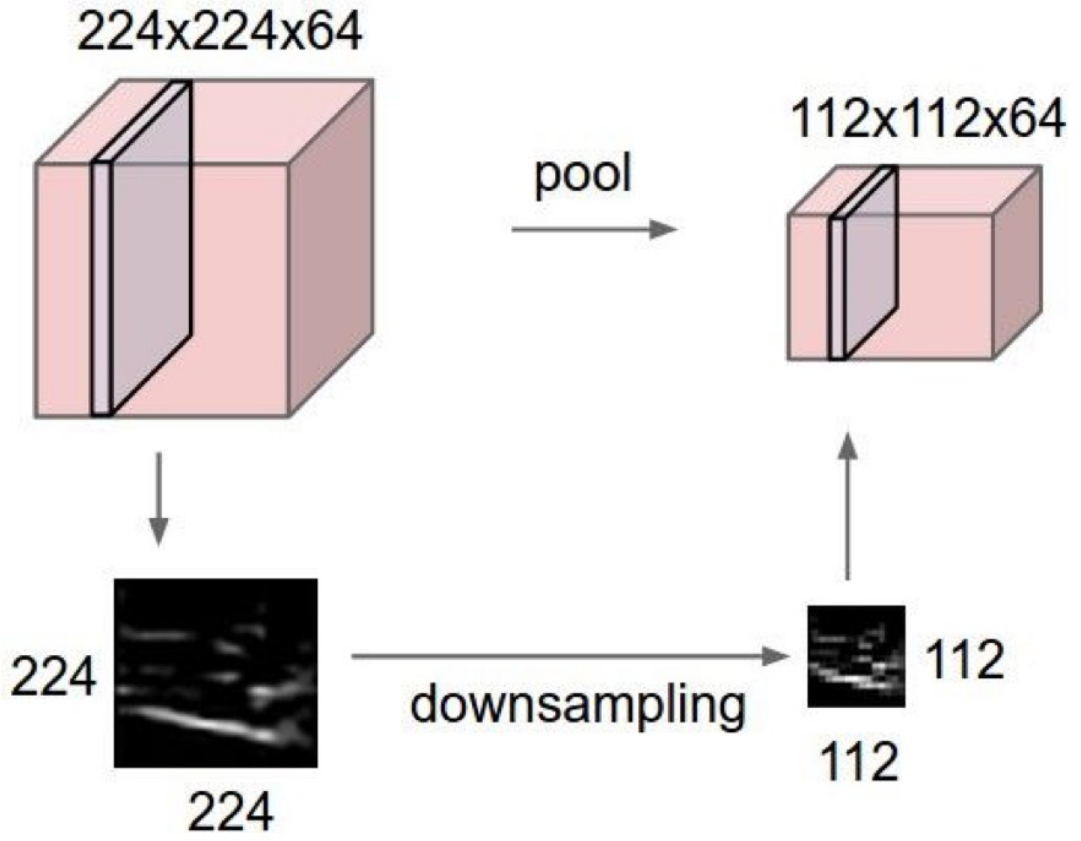
$$f[x,y] * g[x,y] = \sum_{n_1=-\infty}^{\infty} \sum_{n_2=-\infty}^{\infty} f[n_1,n_2] \cdot g[x-n_1, y-n_2]$$

elementwise multiplication and sum of a filter and the signal (image)

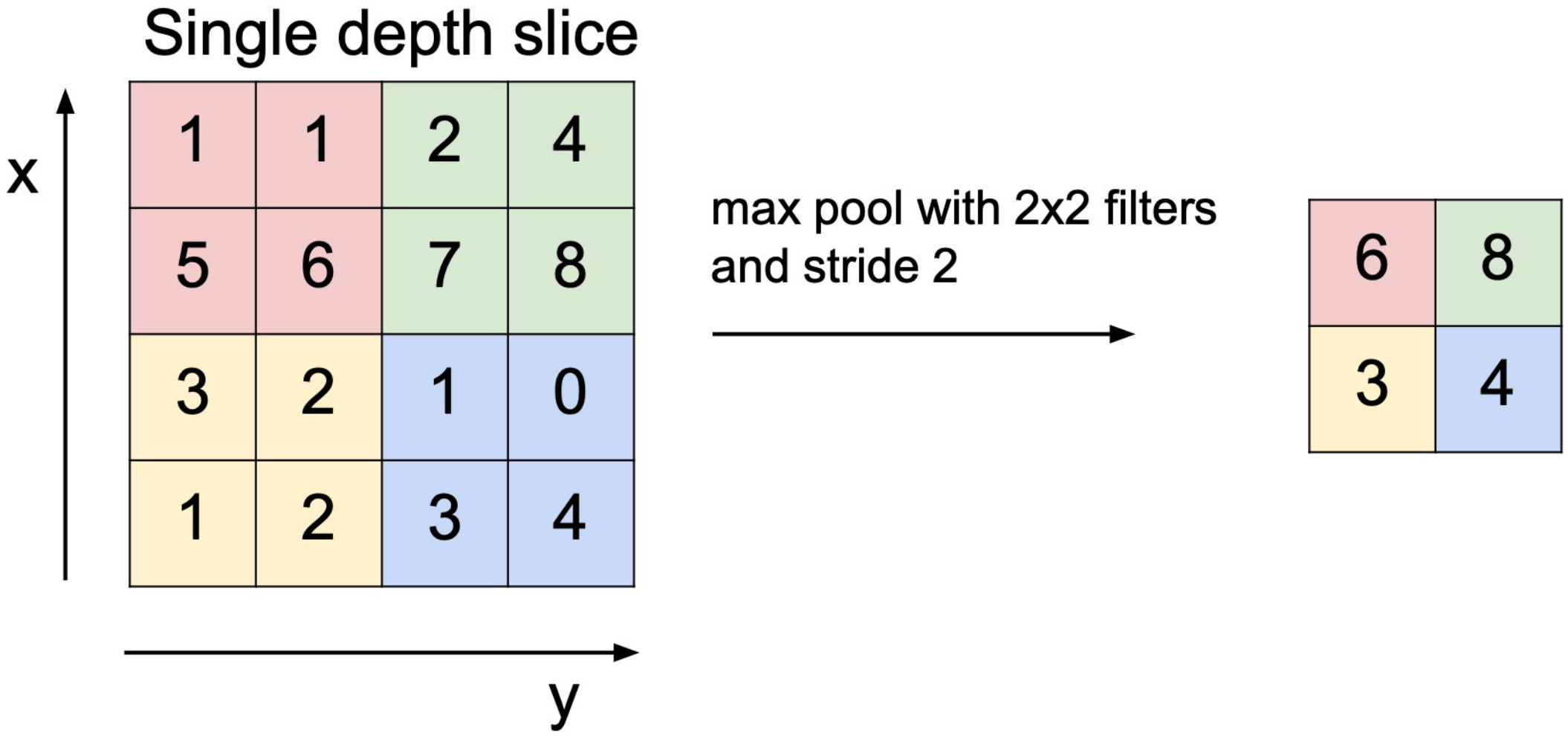
Growth Hackers

Chapter 2.6.1 l Pooling

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

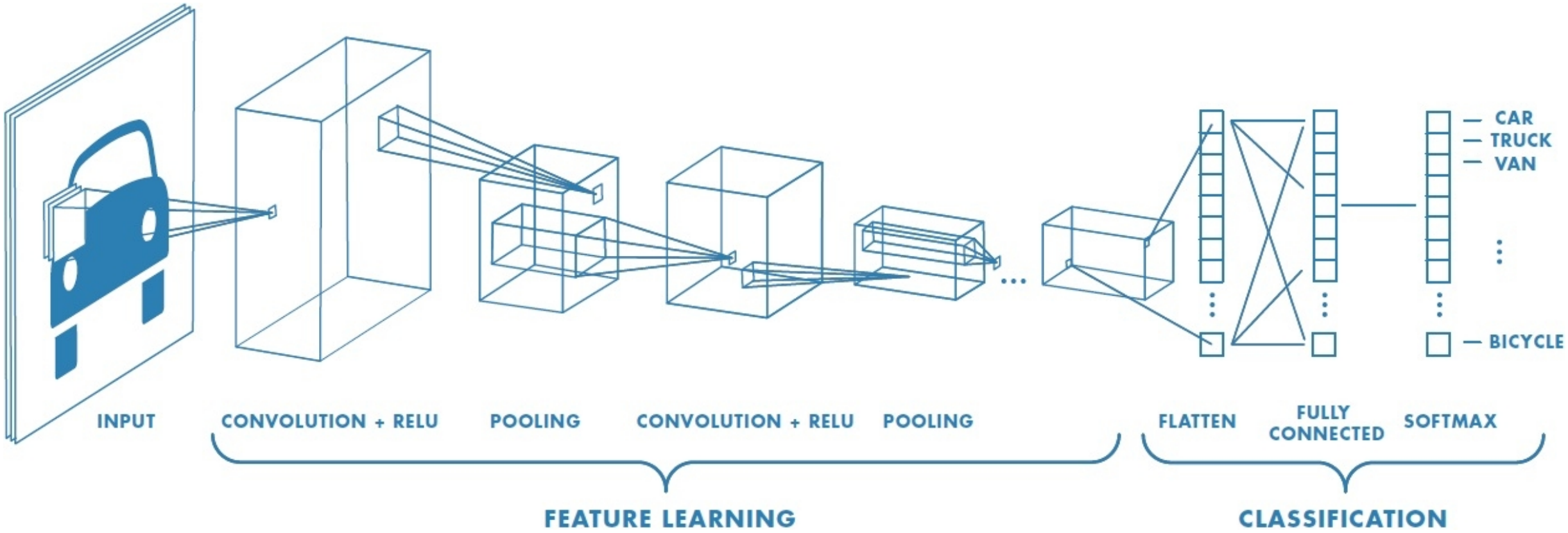


Chapter 2.6.2 l Max Pooling



Growth Hackers

Chapter 2.7 I Convnet 전체 과정



Growth Hackers

Chapter 3

실전 툴

Growth Hackers

Chapter 3.1 I 딥러닝 툴 발전 개요 (2016)

Caffe
(UC Berkeley)

Torch
(NYU / Facebook)

Theano
(U Montreal)

TensorFlow
(Google)

Chapter 3.1 I 딥러닝 툴 발전 개요 (2018)

Caffe
(UC Berkeley)

Caffe2
(Facebook)

Torch
(NYU / Facebook)

PyTorch
(Facebook)

Theano
(U Montreal)

TensorFlow
(Google)

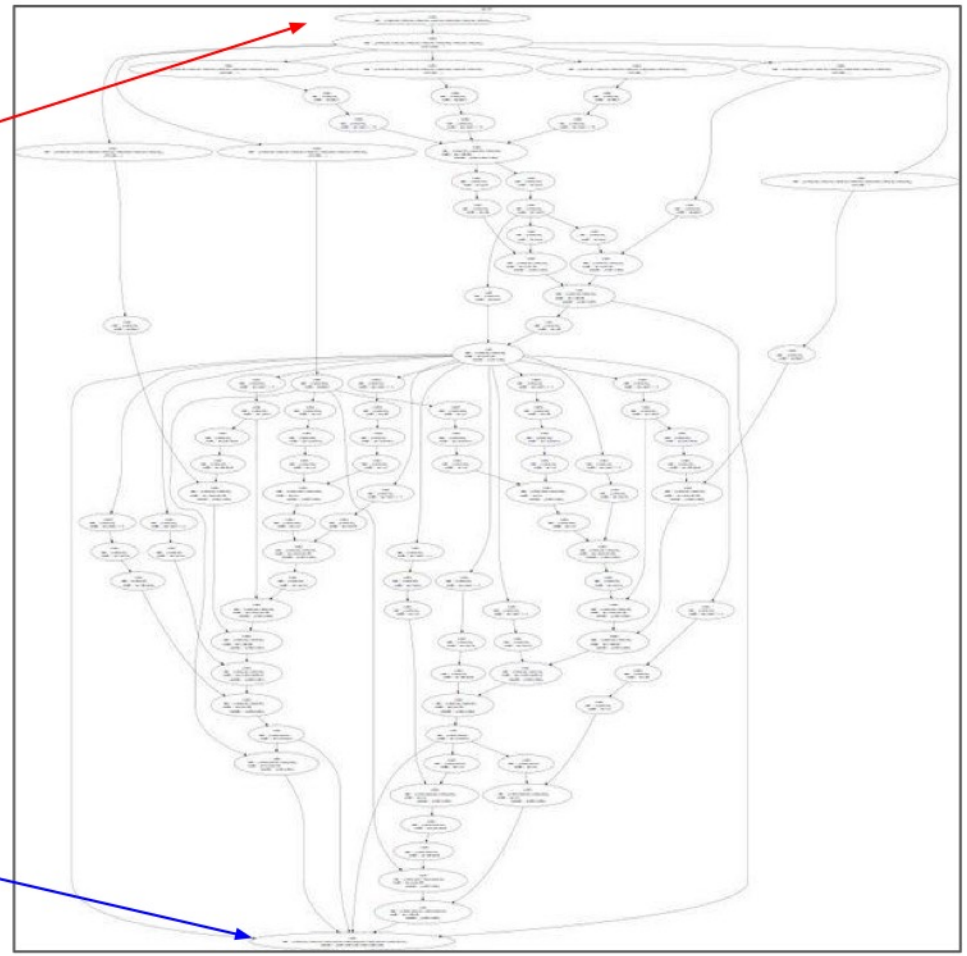
CNTK
(Microsoft)

MXNet
(Amazon)
Developed by U Washington, CMU, MIT, Hong Kong U, etc but main framework of choice at AWS

Chapter 3.2 I Computational Graph 표현

input image

loss

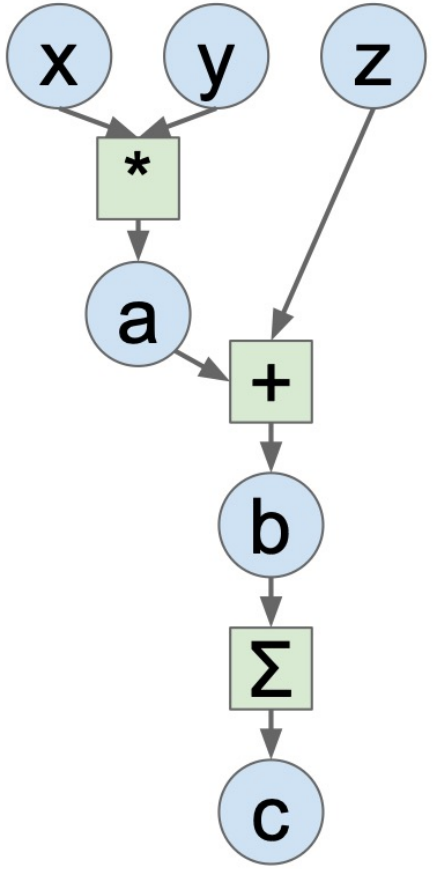


Growth Hackers

Chapter 3.3 I Numpy

Numpy

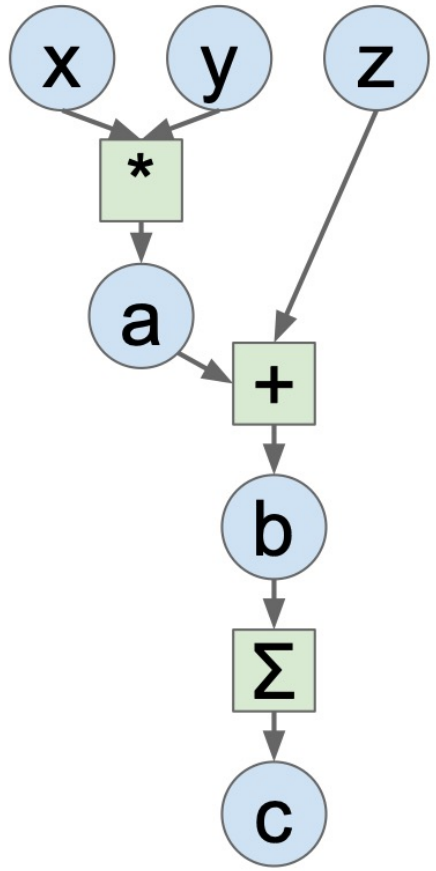
```
import numpy as np
np.random.seed(0)

N, D = 3, 4

x = np.random.randn(N, D)
y = np.random.randn(N, D)
z = np.random.randn(N, D)

a = x * y
b = a + z
c = np.sum(b)

grad_c = 1.0
grad_b = grad_c * np.ones((N, D))
grad_a = grad_b.copy()
grad_z = grad_b.copy()
grad_x = grad_a * y
grad_y = grad_a * x
```



Chapter 3.3 I Numpy

Numpy

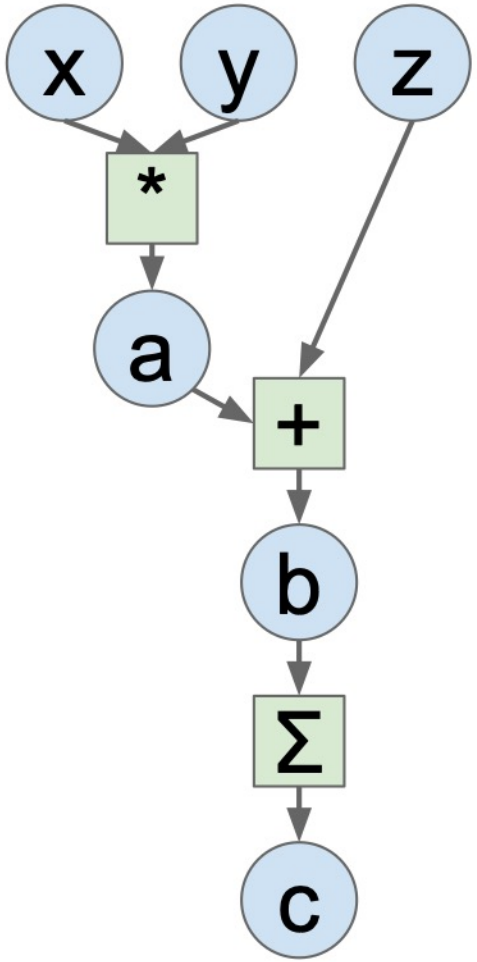
```
import numpy as np
np.random.seed(0)

N, D = 3, 4

x = np.random.randn(N, D)
y = np.random.randn(N, D)
z = np.random.randn(N, D)

a = x * y
b = a + z
c = np.sum(b)

grad_c = 1.0
grad_b = grad_c * np.ones((N, D))
grad_a = grad_b.copy()
grad_z = grad_b.copy()
grad_x = grad_a * y
grad_y = grad_a * x
```



Problems:

- Can't run on GPU
- Have to compute our own gradients

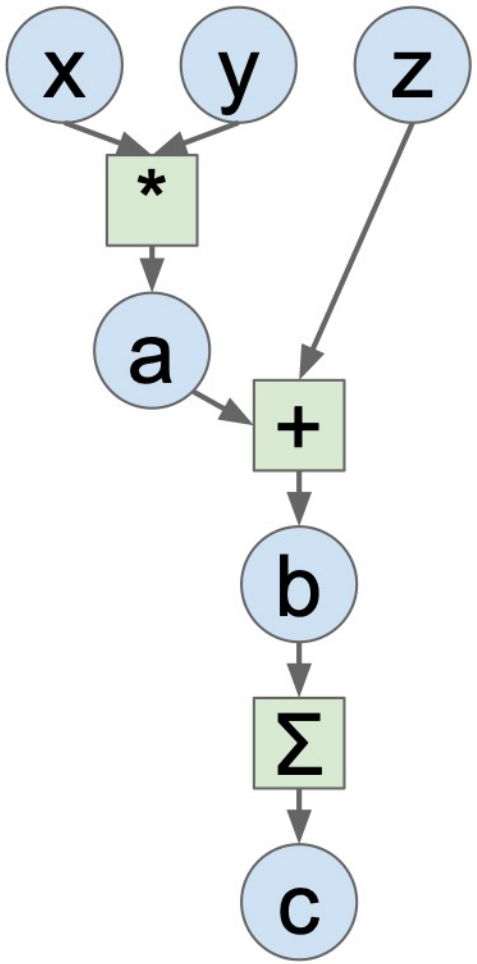
Chapter 3.4 I PyTorch



Define **Variables** to start building a computational graph

```
import torch
from torch.autograd import Variable

N, D = 3, 4

x = Variable(torch.randn(N, D),
             requires_grad=True)
y = Variable(torch.randn(N, D),
             requires_grad=True)
z = Variable(torch.randn(N, D),
             requires_grad=True)

a = x * y
b = a + z
c = torch.sum(b)

c.backward()

print(x.grad.data)
print(y.grad.data)
print(z.grad.data)
```

Chapter 3.4 I PyTorch

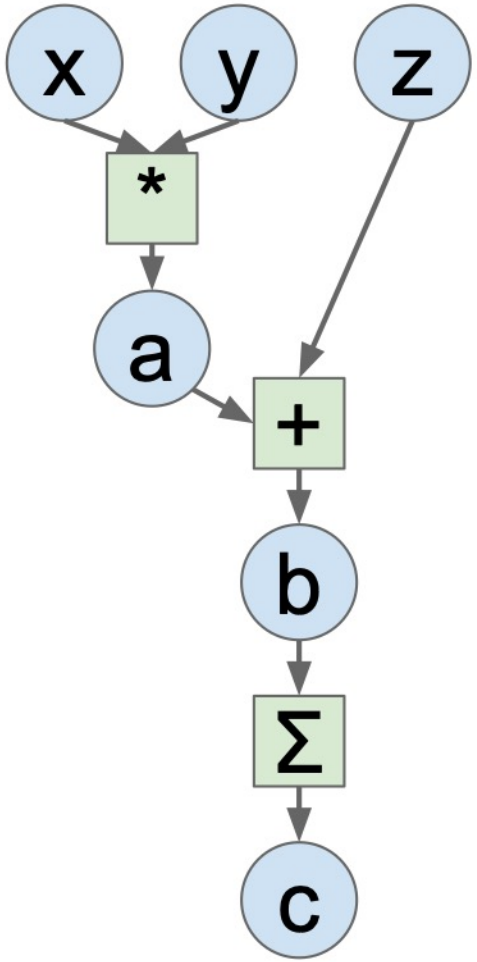


Forward pass looks just like numpy

```
import torch
from torch.autograd import Variable

N, D = 3, 4

x = Variable(torch.randn(N, D),
             requires_grad=True)
y = Variable(torch.randn(N, D),
             requires_grad=True)
z = Variable(torch.randn(N, D),
             requires_grad=True)

a = x * y
b = a + z
c = torch.sum(b)

c.backward()

print(x.grad.data)
print(y.grad.data)
print(z.grad.data)
```


Growth Hackers

Chapter 3.4 I PyTorch

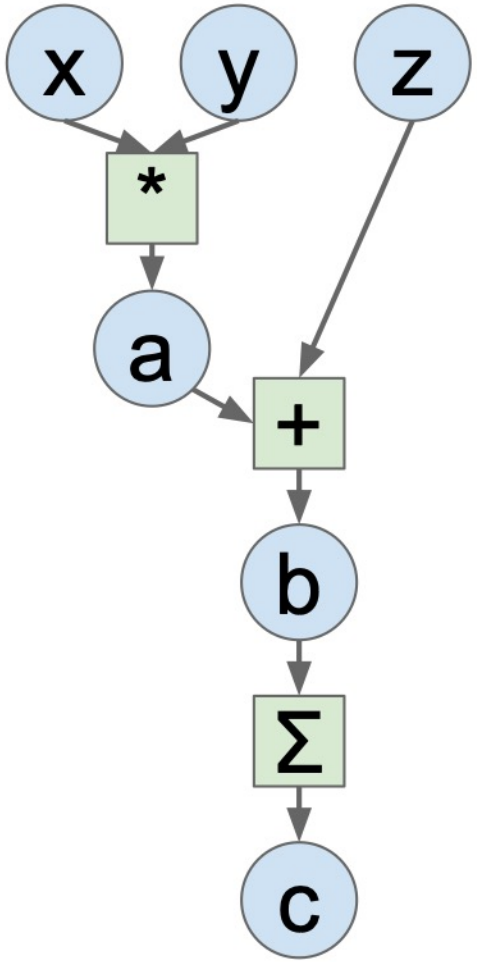


Calling c.backward() computes all gradients

```
import torch
from torch.autograd import Variable

N, D = 3, 4

x = Variable(torch.randn(N, D),
             requires_grad=True)
y = Variable(torch.randn(N, D),
             requires_grad=True)
z = Variable(torch.randn(N, D),
             requires_grad=True)

a = x * y
b = a + z
c = torch.sum(b)

c.backward()

print(x.grad.data)
print(y.grad.data)
print(z.grad.data)
```


Growth Hackers

Chapter 3.4 I PyTorch



Run on GPU by casting to .cuda()

```
import torch
from torch.autograd import Variable

N, D = 3, 4

x = Variable(torch.randn(N, D).cuda(),
             requires_grad=True)
y = Variable(torch.randn(N, D).cuda(),
             requires_grad=True)
z = Variable(torch.randn(N, D).cuda(),
             requires_grad=True)

a = x * y
b = a + z
c = torch.sum(b)

c.backward()

print(x.grad.data)
print(y.grad.data)
print(z.grad.data)
```

Chapter 4

코딩 (ipynb 활용)

Growth Hackers

E.O.D

Growth Hackers