Lecture 3 Deep Learning and Convolutional Neural Network

Tian Sheuan Chang

Outline

- Basic structures
  - Convolution
  - Pooling
- What does CNN learn? Visualization of CNN

CNN INTRODUCTION

2-dimensional Inputs

- NN 的輸入是一維的向量，那二維的矩陣呢? 例如 圖形資料

Figures reference
https://twitter.com/gonainlive/status/507563446612013057

Ordinary Feedforward DNN with Image

- 將圖形轉換成一維向量
  - Weight 數過多 => training 所需時間太長
  - 左上的圖形跟右下的圖形真的有關係嗎?

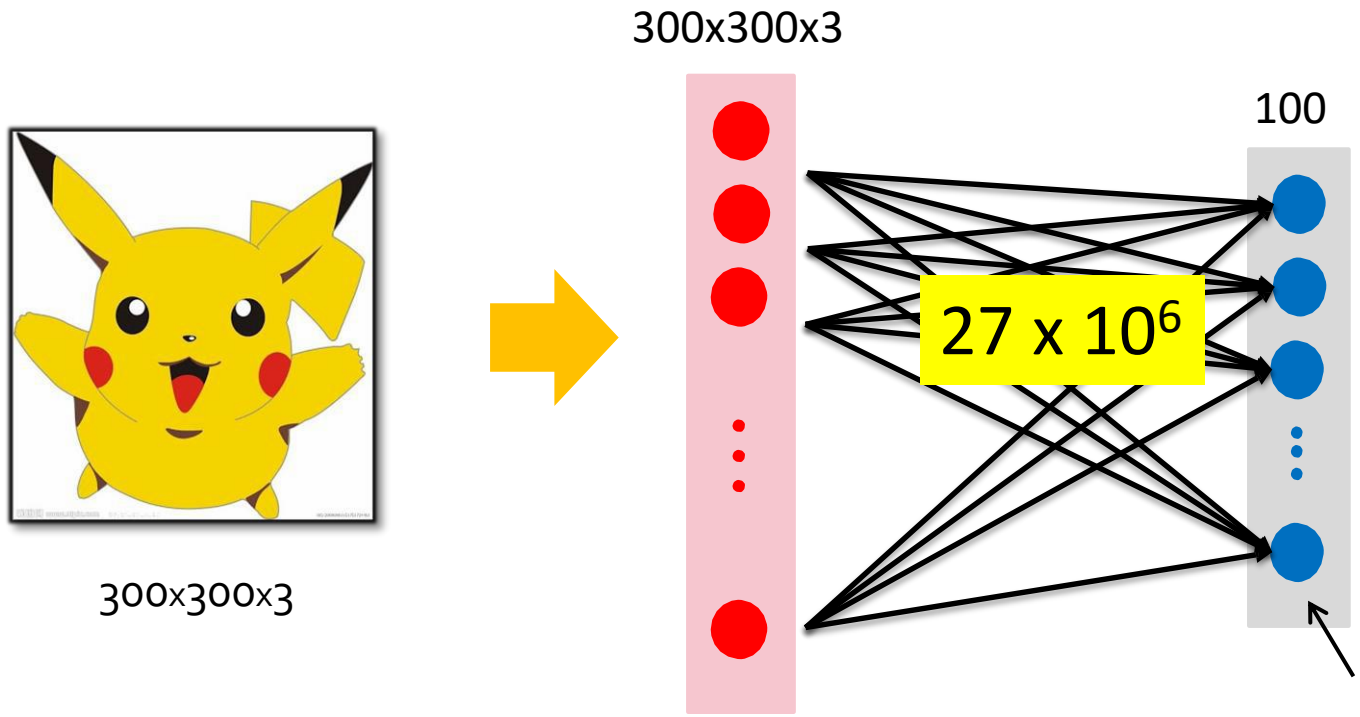


Figure reference
http://www.ettoday.net/dalemon/post/12934

From Fully Connected NN to Convolutional NN

引進先驗知識
Prior knowledge

- Some patterns are much smaller than the whole image

A neuron does not have to see the whole image to discover the pattern.

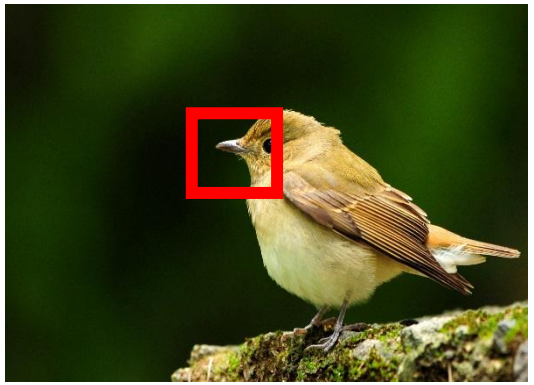
Connecting to small region with fewer parameters

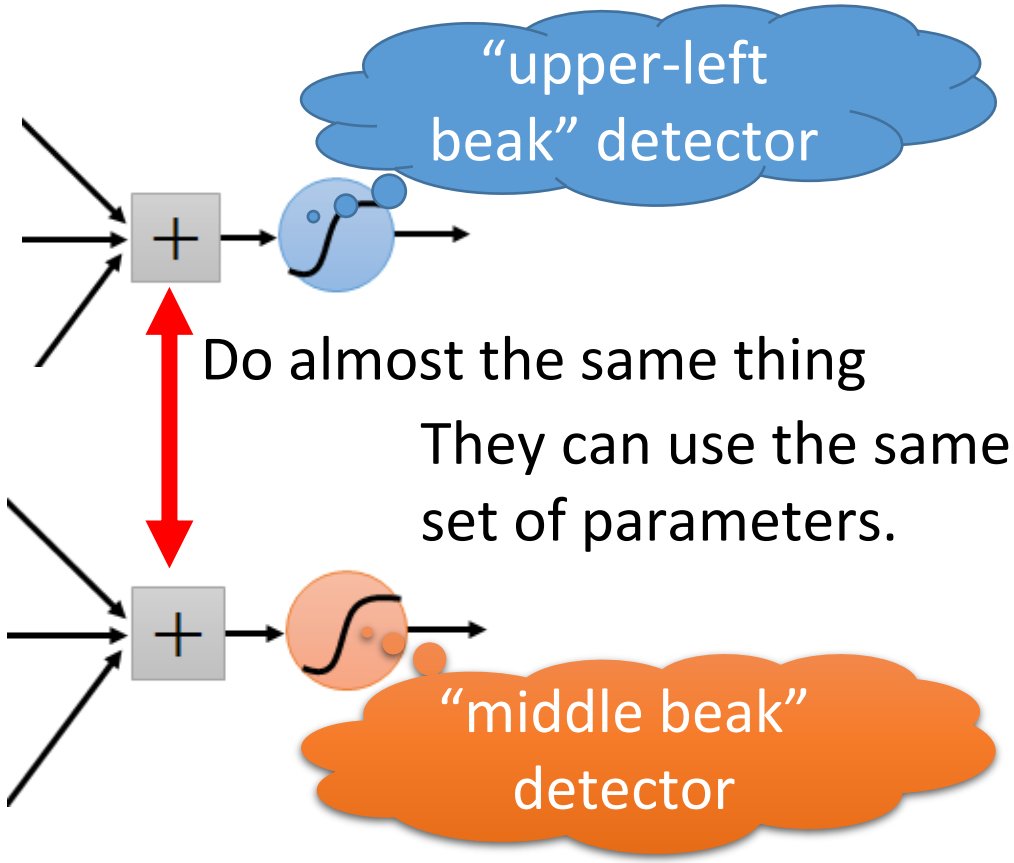
Sparse Interactions (connectivity)　辨識需要的特徵大小遠比整張影像小

Source: Convolutional Neural Network, Hung-yi Lee

引進先驗知識
Prior knowledge

- The same patterns appear in different regions.





Parameter Sharing
Translation invariant

位置不影響特徵辨識

Source: Convolutional Neural Network, Hung-yi Lee

引進先驗知識
Prior knowledge

- Subsampling the pixels will not change the object

bird

subsampling

bird

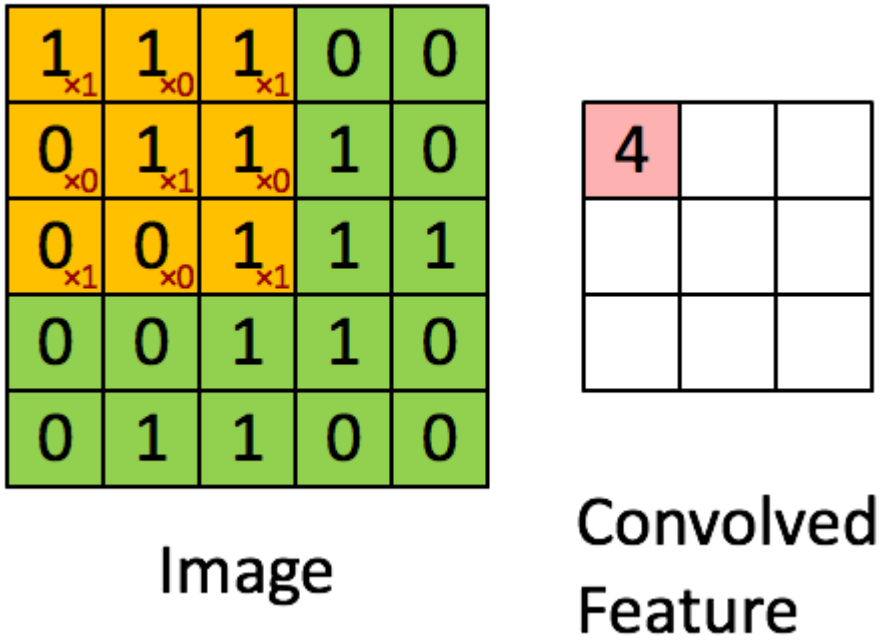
We can subsample the pixels to make image smaller

→ Less parameters for the network to process the image

大小的變化並沒有太多影響

Source: Convolutional Neural Network, Hung-yi Lee

Convolution



source: http://deeplearning.stanford.edu/wiki/index.php/Feature_extraction_using_convolution

找影像特徵=> Convolution Filter in Computer Vision (CV)

- E.g. Sobel edge filter

-1	0	1
-2	0	2
-1	0	1

Horizontal

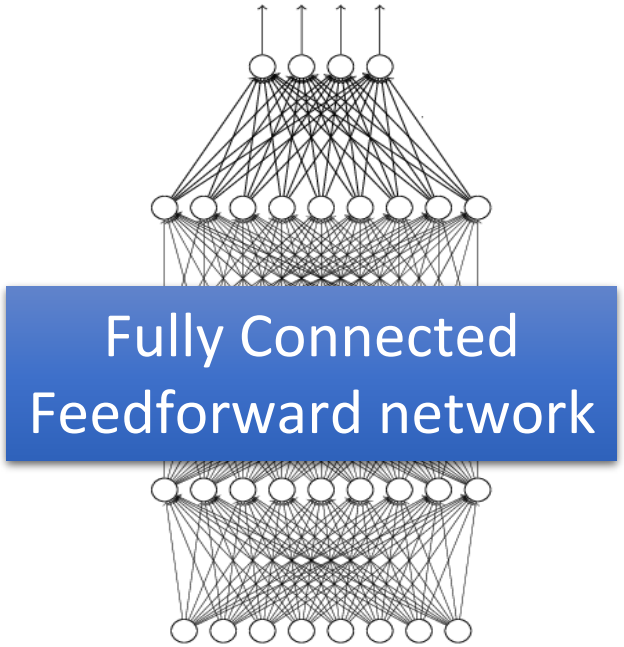
-1	-2	-1
0	0	0
-1	-2	-1

Vertical

An example of Sobel edge detection.
From: http://wikis.ece.iastate.edu/cpre584/images/b/bd/Sobel_edge_detection.jpg

A filter could be seen as a pattern
可以自動學filter (pattern)? => DL

The whole CNN



Note. We do not show the activation after convolutional layer

Source: Convolutional Neural Network, Hung-yi Lee

The whole CNN

Property 1

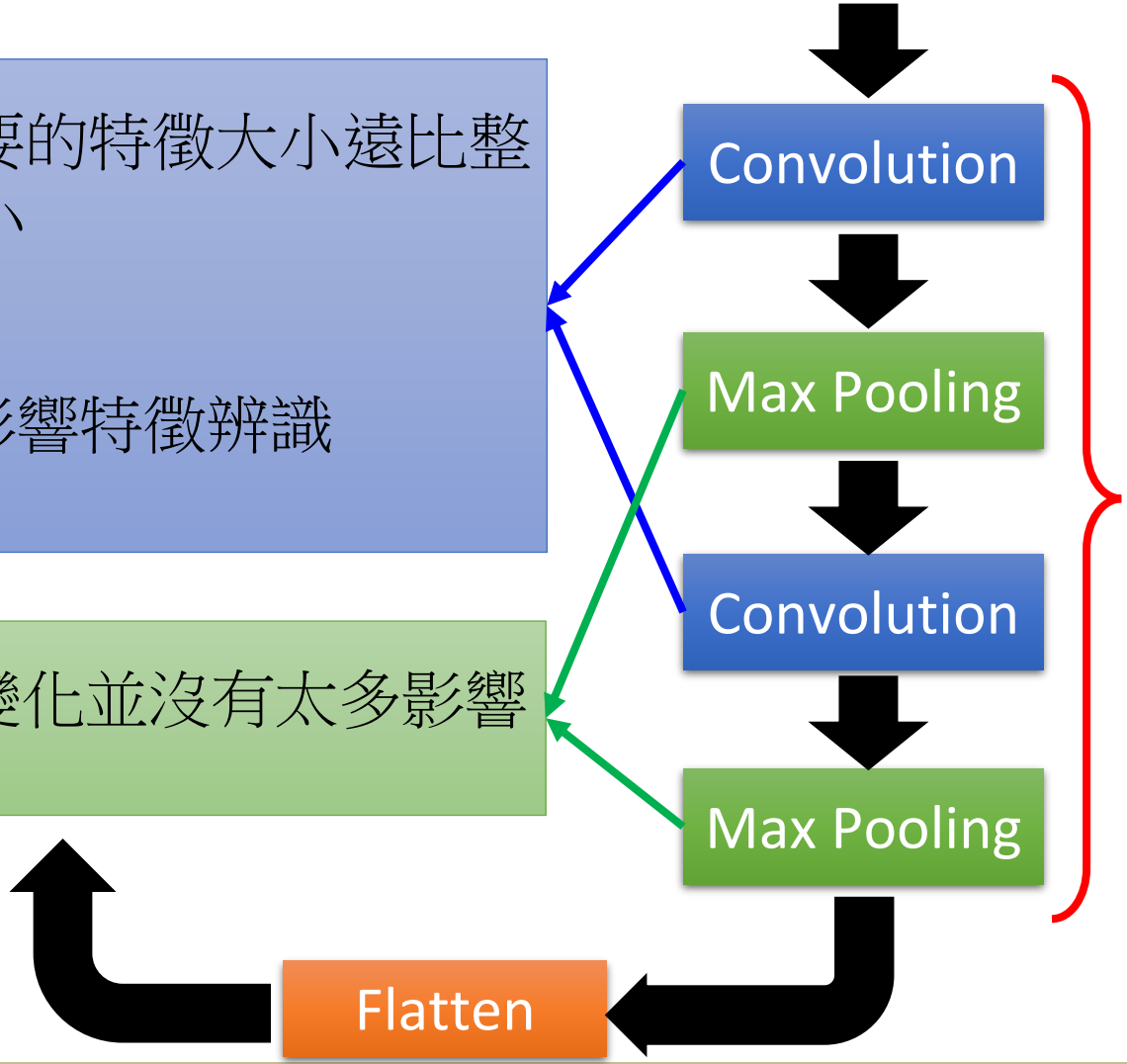
- 辨識需要的特徵大小遠比整張影像小

Property 2

- 位置不影響特徵辨識

Property 3

- 大小的變化並沒有太多影響

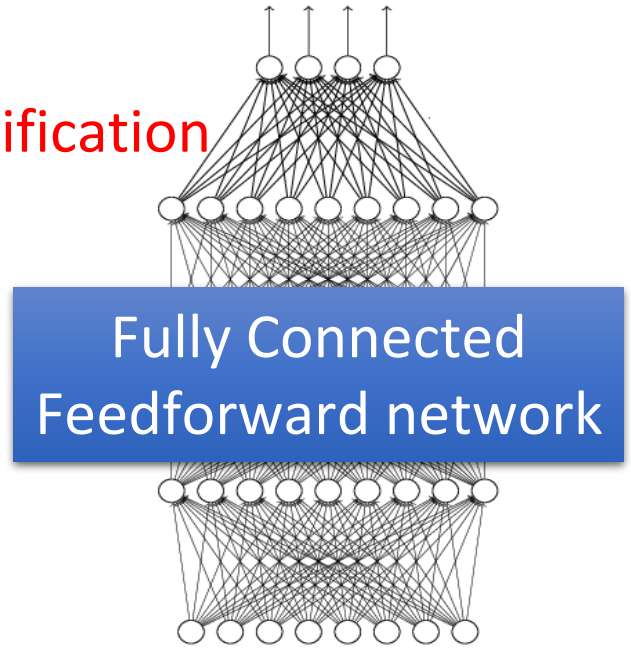


Basic layer structure

- Convolutional layer
- Pooling layer
- Activations
- Fully connected layer

Source: Convolutional Neural Network, Hung-yi Lee

The whole CNN



Note. We do not show the activation after convolutional layer

Source: Convolutional Neural Network, Hung-yi Lee

Summary: Convolution

- Convolutions leverage three important ideas that can help improve a machine learning system:
  - Sparse Interactions (connectivity)
    - achieved by making the kernel smaller than the input
    - Fewer parameters/fewer computations
  - Parameter Sharing
    - applying the same kernel to every position of the input
    - not affect the runtime at inference, but fewer memory requirements (K filters)
  - Equivariant Representation (translation invariance)
    - A function f (x) is equivariant to a function g (x) if f(g(x)) = g(f(x)).
    - Convolution operators are equivariant to translation of the input.
    - Need other mechanisms for rotation or scale changes (e.g. pooling)
- Moreover, convolutions provide a way to handle input of different sizes.
  - Fully connected NN has fixed input size

CNN – Convolution

Those are the network parameters to be learned.

1	0	0	0	0	1
0	1	0	0	1	0
0	0	1	1	0	0
1	0	0	0	1	0
0	1	0	0	1	0
0	0	1	0	1	0

6 x 6 image

1	-1	-1
-1	1	-1
-1	-1	1

Filter 1

Matrix

-1	1	-1
-1	1	-1
-1	1	-1

Filter 2

Matrix

⋮

Property 1

Each filter detects a small pattern (3 x 3)辨識需要的特徵大小遠比整張影像小

Source: Convolutional Neural Network, Hung-yi Lee

CNN – Convolution

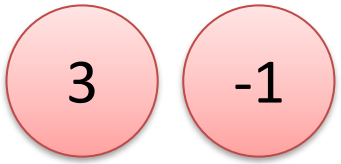
1	-1	-1
-1	1	-1
-1	-1	1

Filter 1

stride=1

1	0	0	0	0	1
0	1	0	0	1	0
0	0	1	1	0	0
1	0	0	0	1	0
0	1	0	0	1	0
0	0	1	0	1	0

6 x 6 image



Source: Convolutional Neural Network, Hung-yi Lee

CNN – Convolution

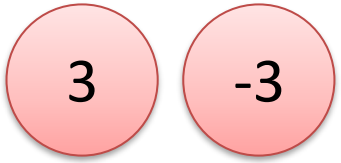
1	-1	-1
-1	1	-1
-1	-1	1

Filter 1

If stride=2

1	0	0	0	0	1
0	1	0	0	1	0
0	0	1	1	0	0
1	0	0	0	1	0
0	1	0	0	1	0
0	0	1	0	1	0

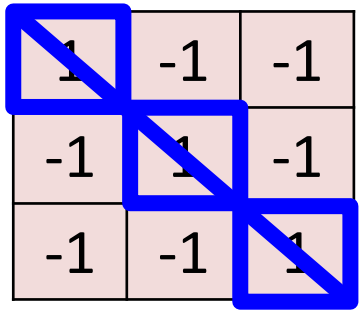
6 x 6 image



We set stride=1 below

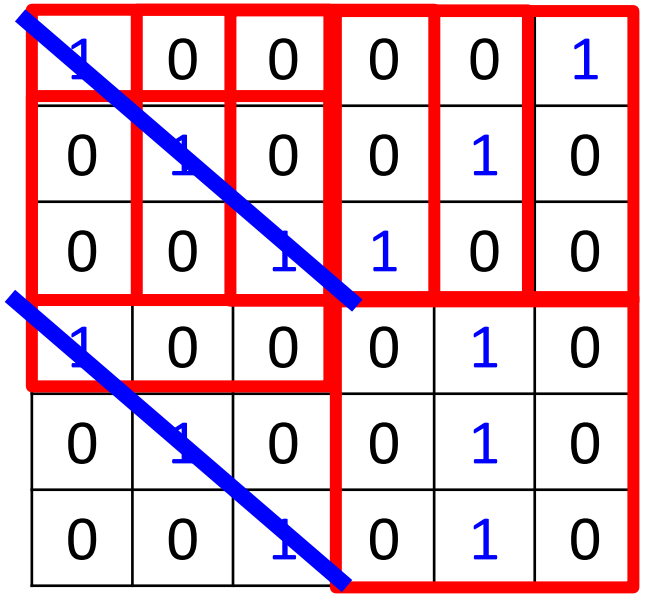
Source: Convolutional Neural Network, Hung-yi Lee

CNN – Convolution

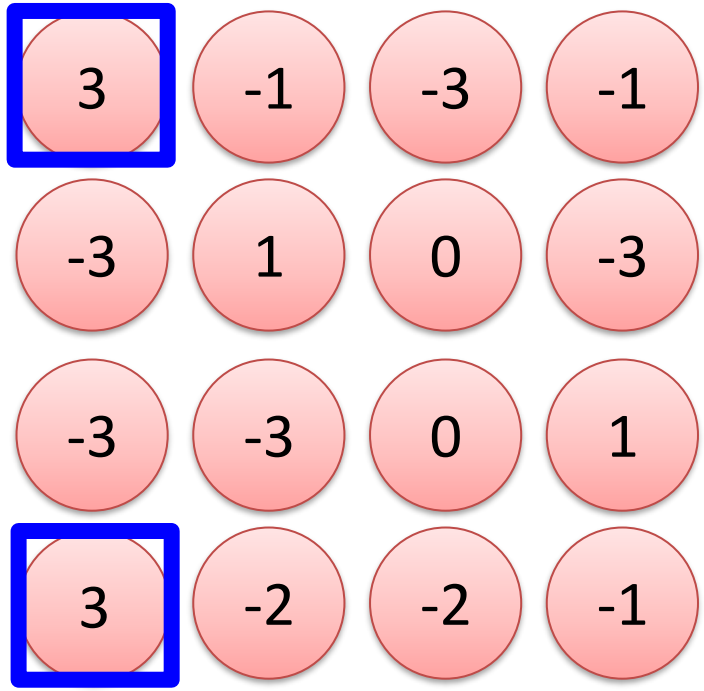


Filter 1

stride=1



6 x 6 image



Property 2

位置不影響特徵辨識

Source: Convolutional Neural Network, Hung-yi Lee

CNN – Convolution

-1	1	-1
-1	1	-1
-1	1	-1

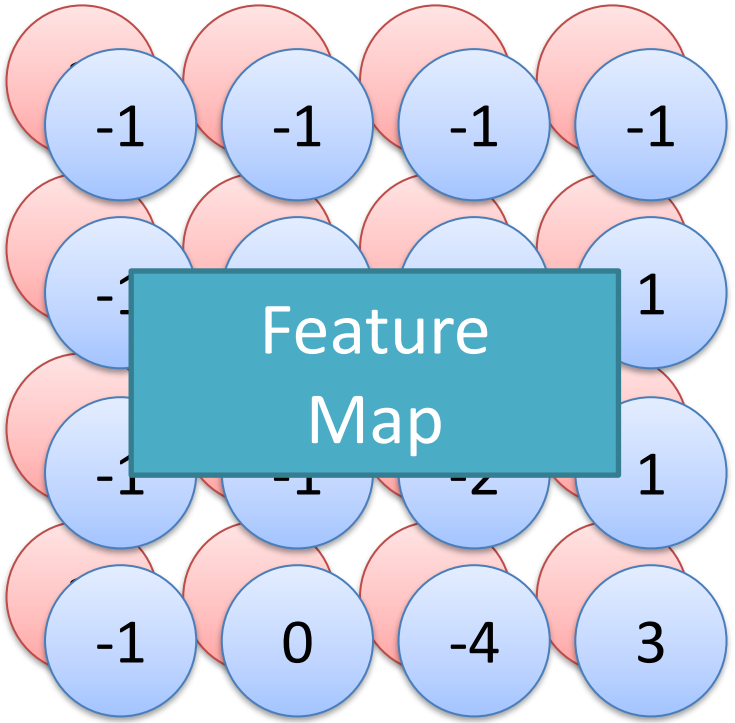
Filter 2

stride=1

1	0	0	0	0	1
0	1	0	0	1	0
0	0	1	1	0	0
1	0	0	0	1	0
0	1	0	0	1	0
0	0	1	0	1	0

6 x 6 image

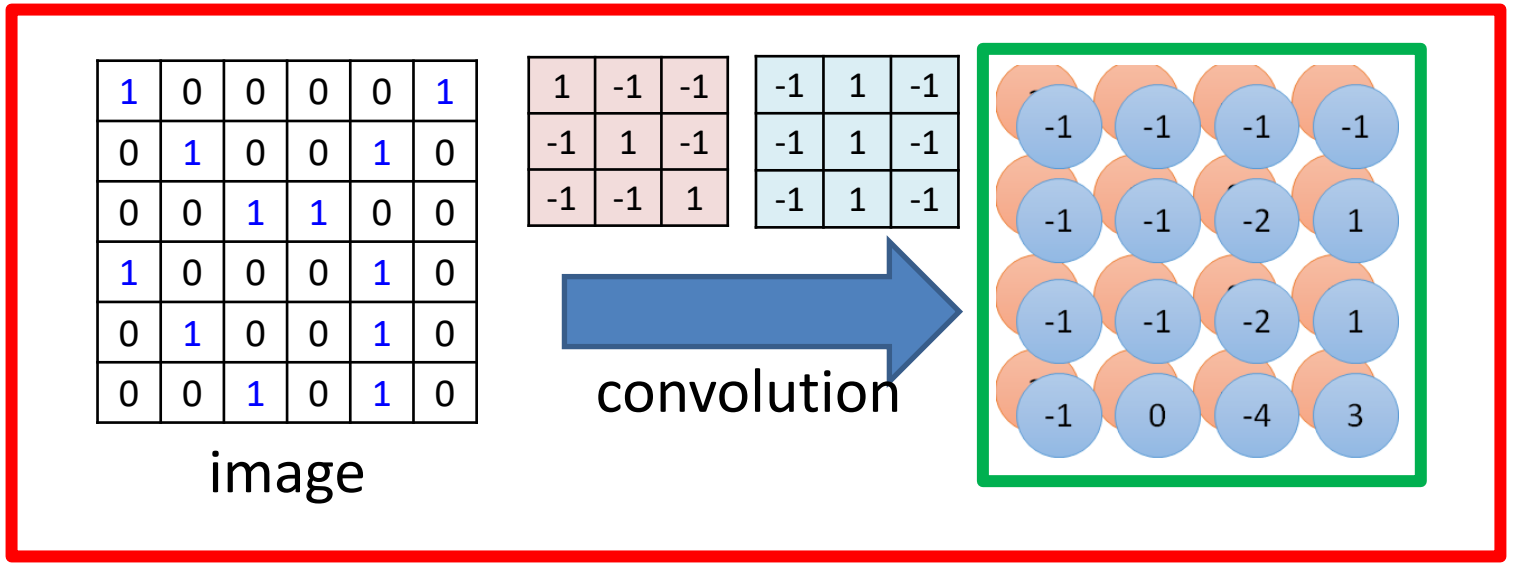
Do the same process for every filter



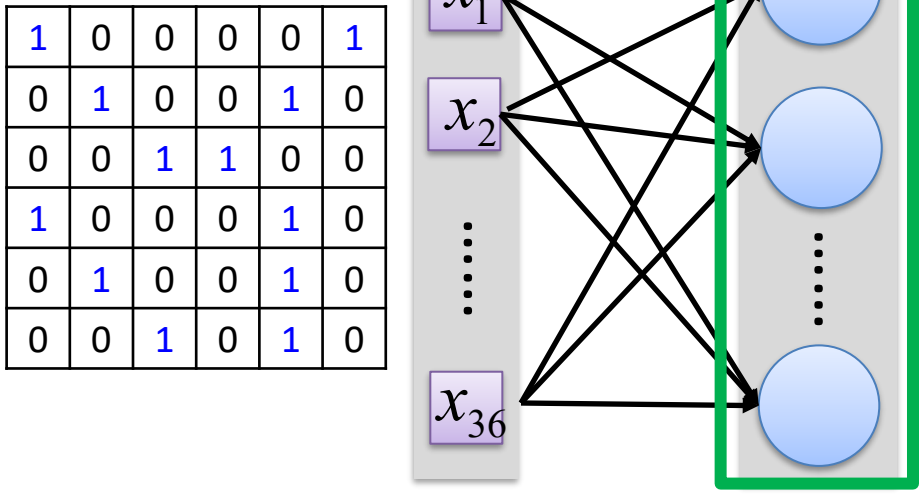
4 x 4 image

Source: Convolutional Neural Network, Hung-yi Lee

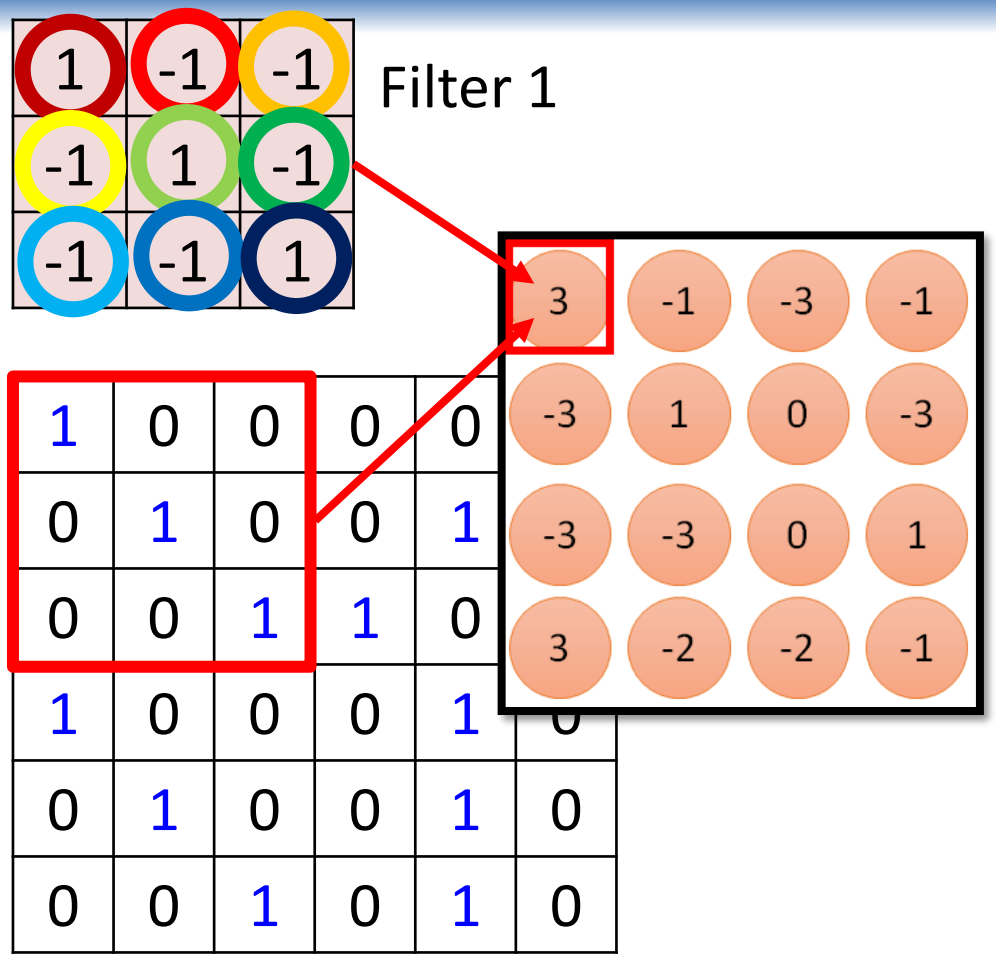
Convolution layer is a special case of Fully Connected layer

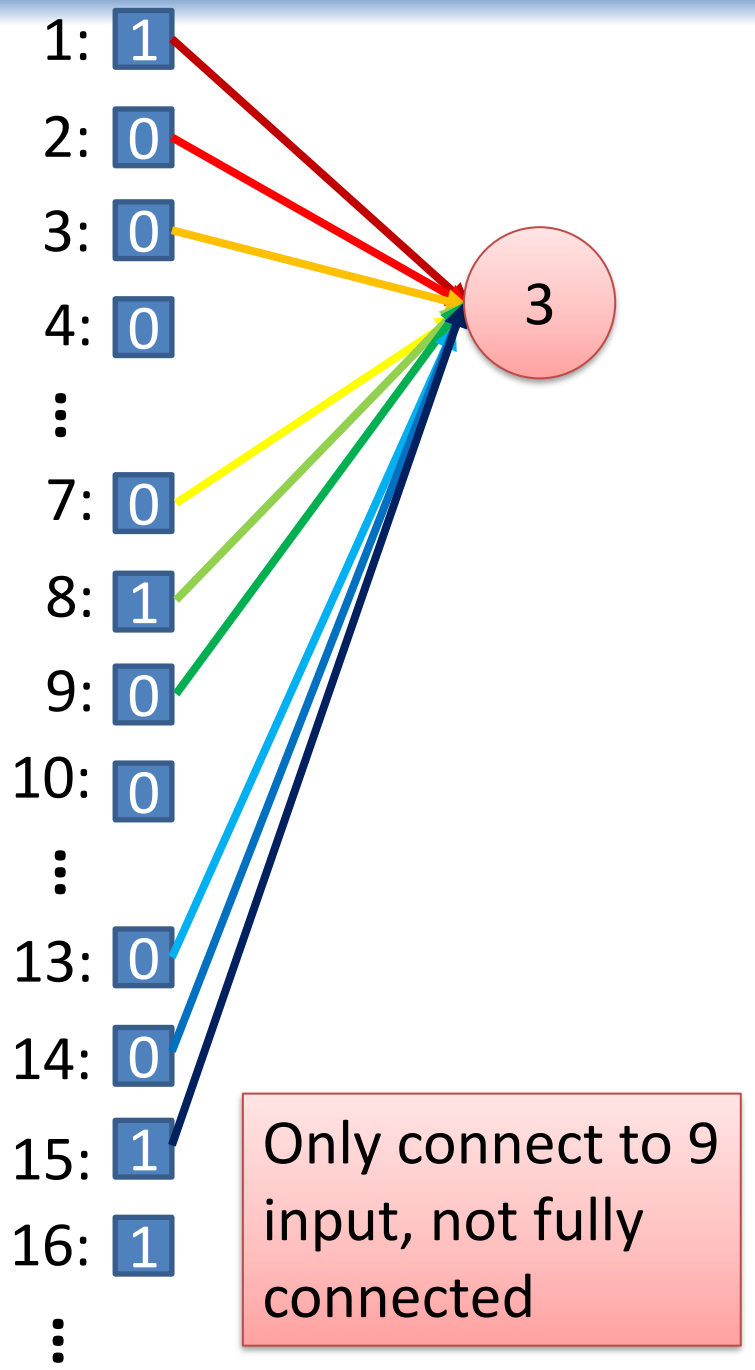


Fully-connected



Source: Convolutional Neural Network, Hung-yi Lee

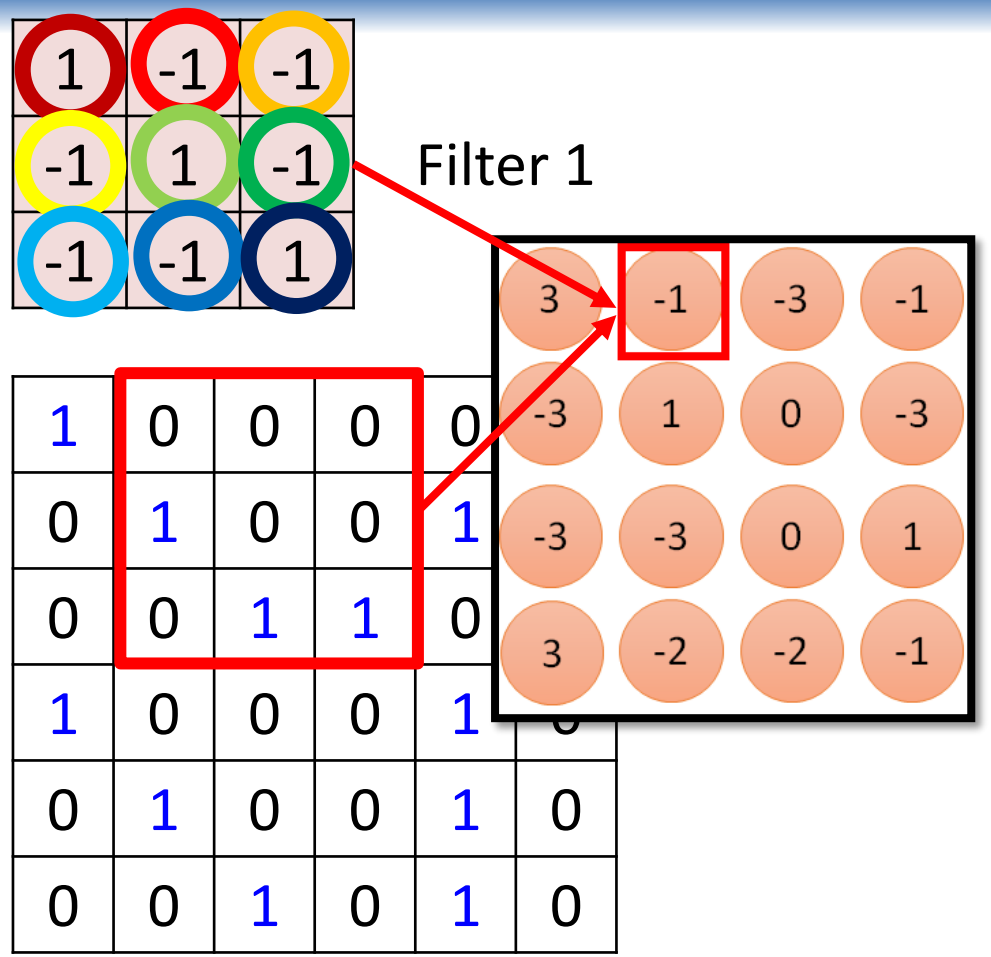


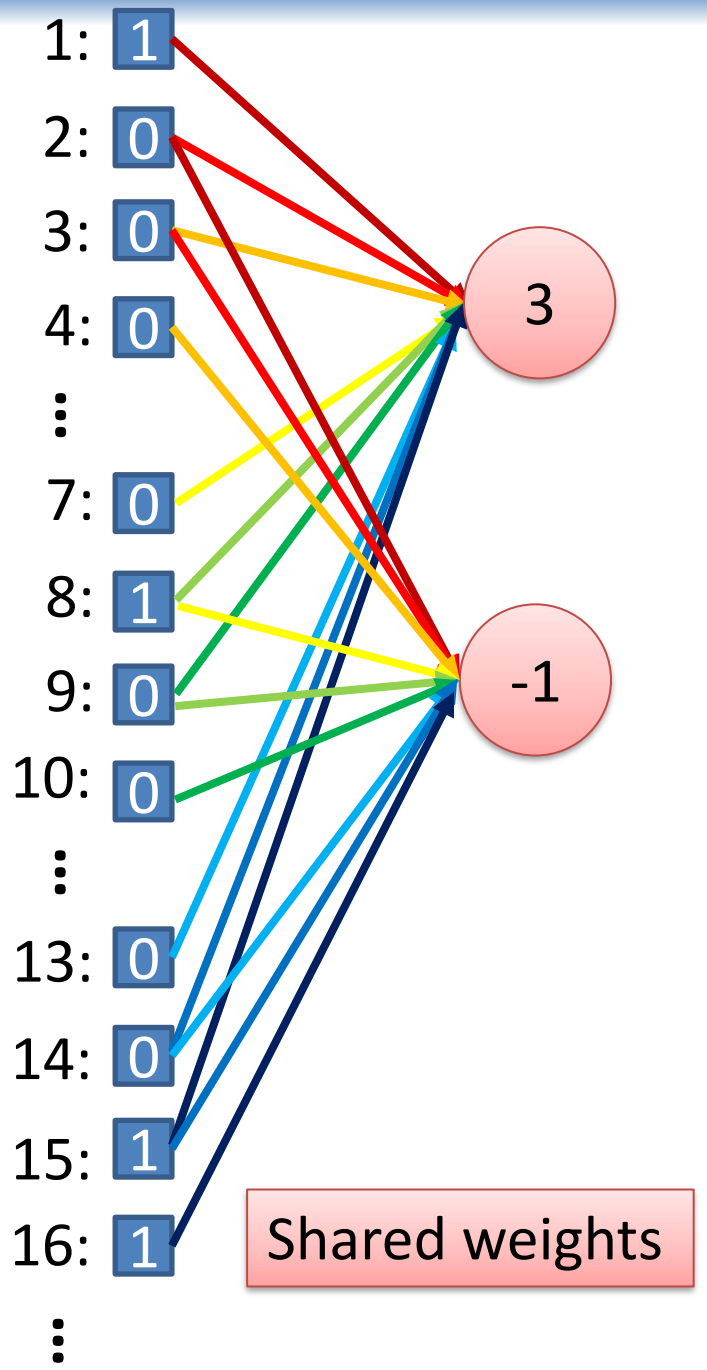


Sparse connection **Less parameters!**

Only connect to 9 input, not fully connected

Source: Convolutional Neural Network, Hung-yi Lee



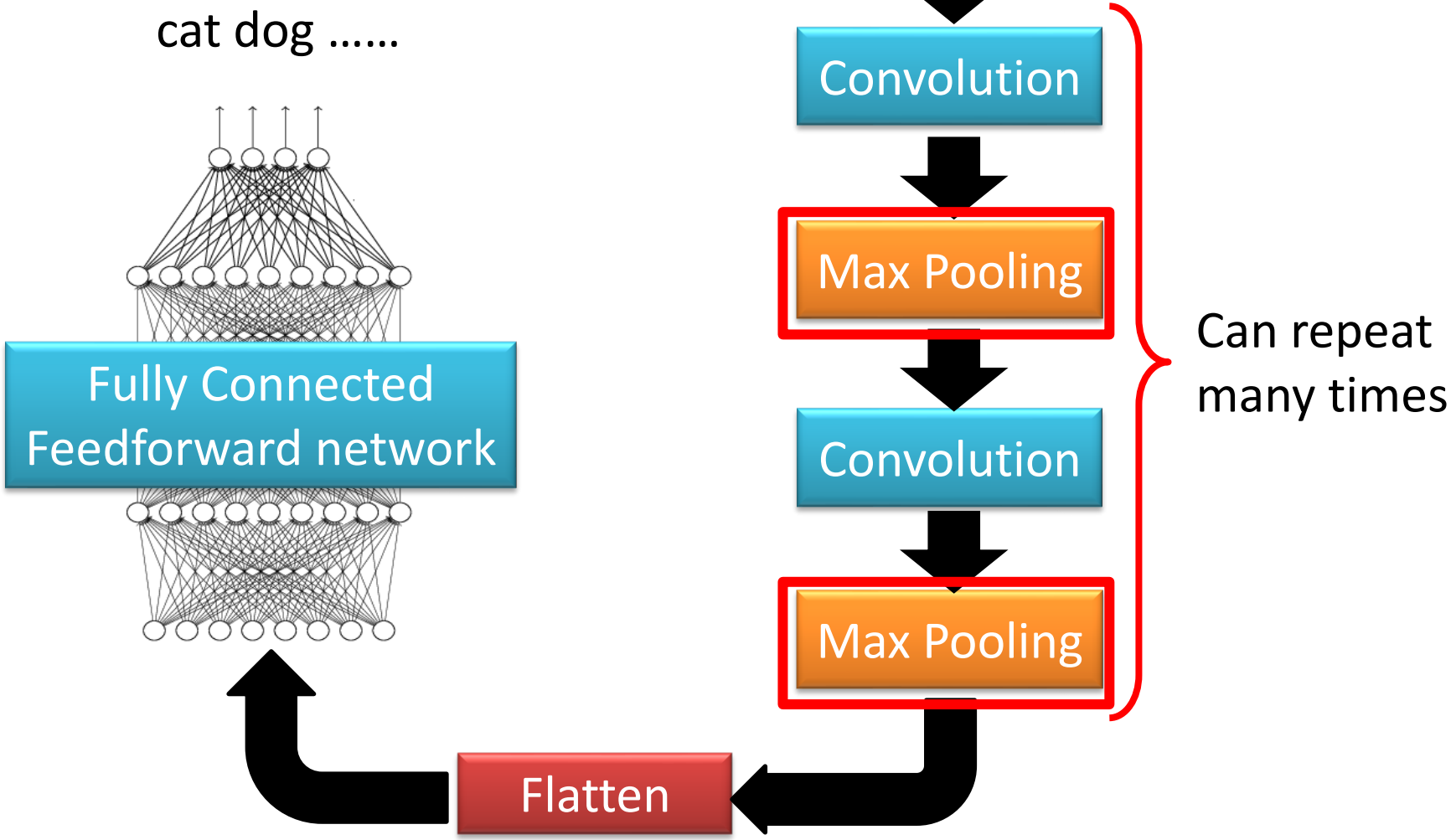


Sparse connection **fewer parameters!**

Shared parameters **Even fewer parameters!**

Source: Convolutional Neural Network, Hung-yi Lee

The whole CNN



Source: Convolutional Neural Network, Hung-yi Lee

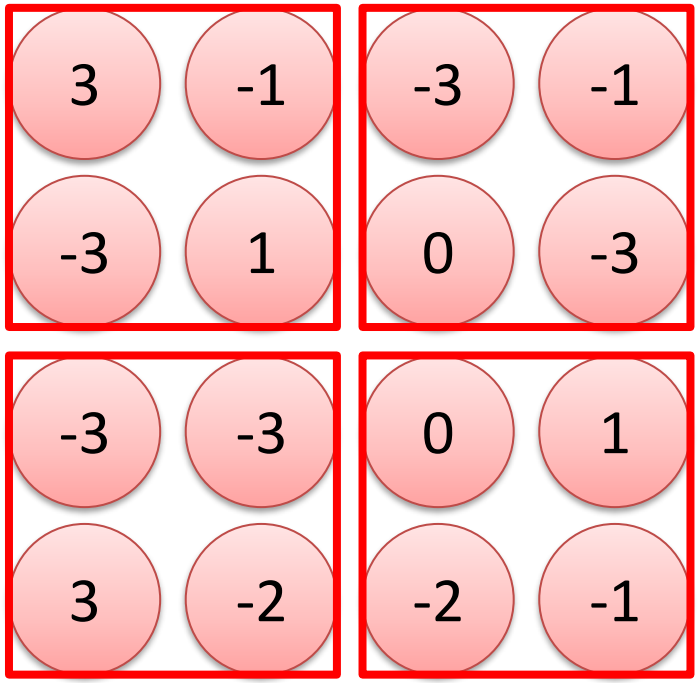
CNN – Max Pooling

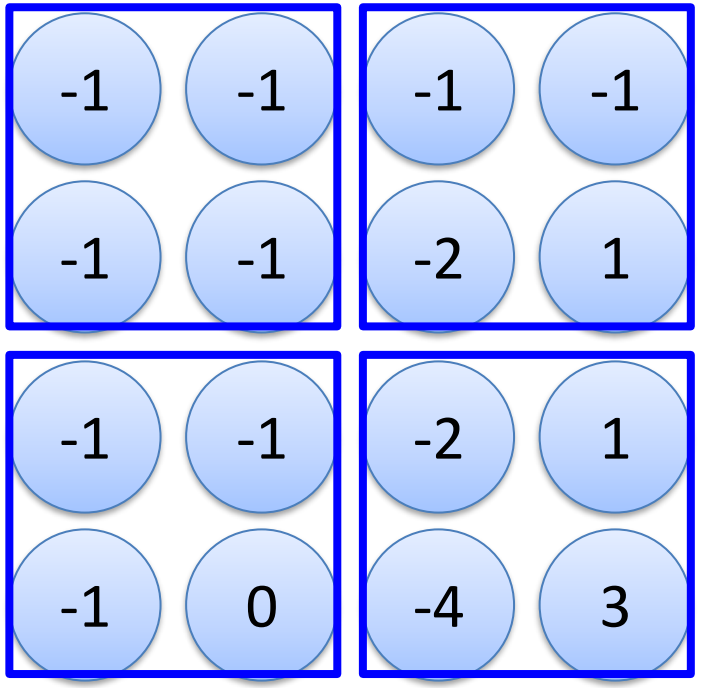
1	-1	-1
-1	1	-1
-1	-1	1

Filter 1

-1	1	-1
-1	1	-1
-1	1	-1

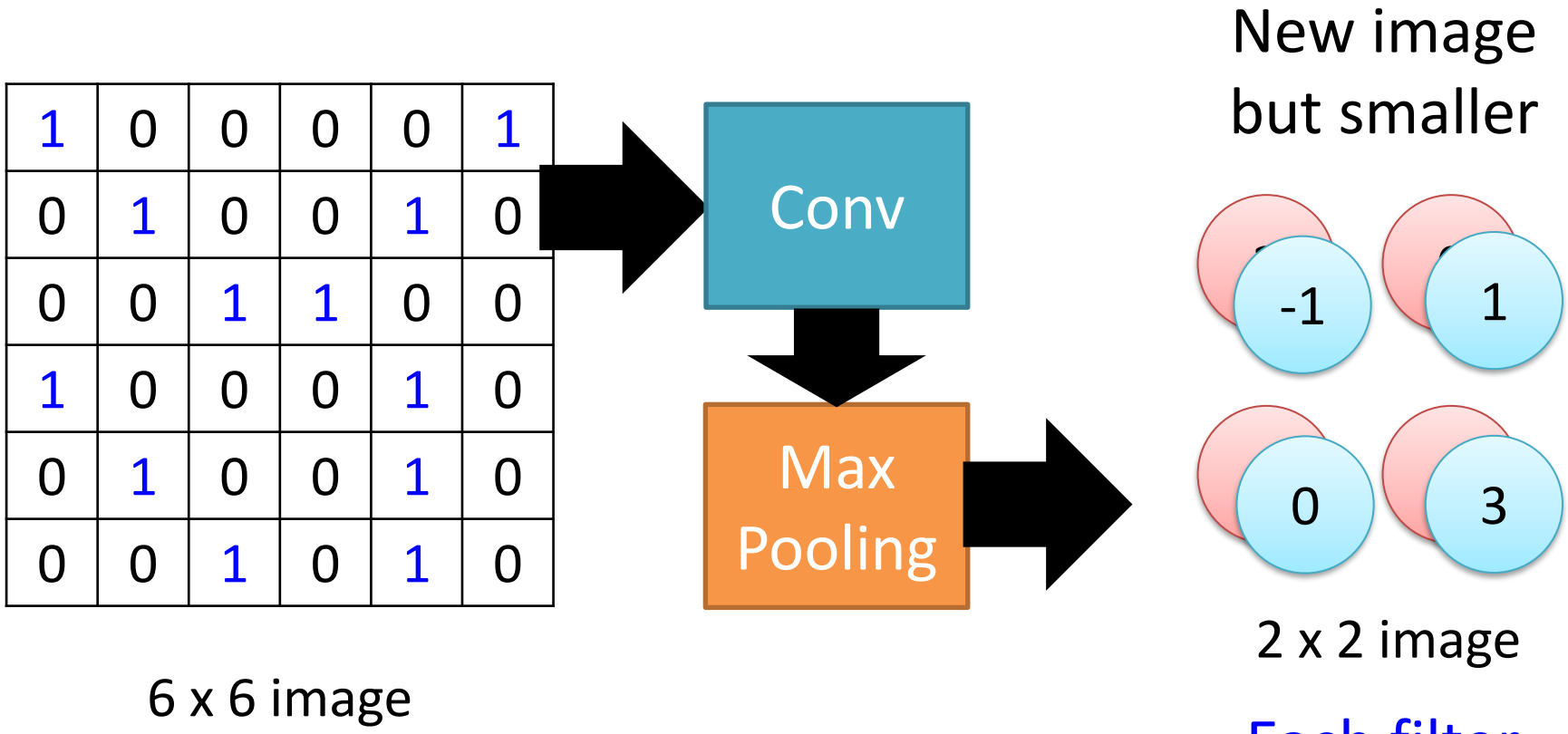
Filter 2





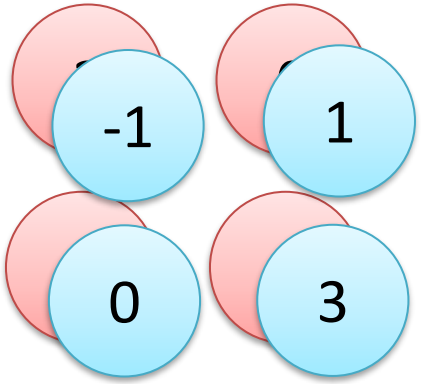
Source: Convolutional Neural Network, Hung-yi Lee

CNN – Max Pooling



Source: Convolutional Neural Network, Hung-yi Lee

The whole CNN



A new image

Convolution

Max Pooling

Convolution

Max Pooling

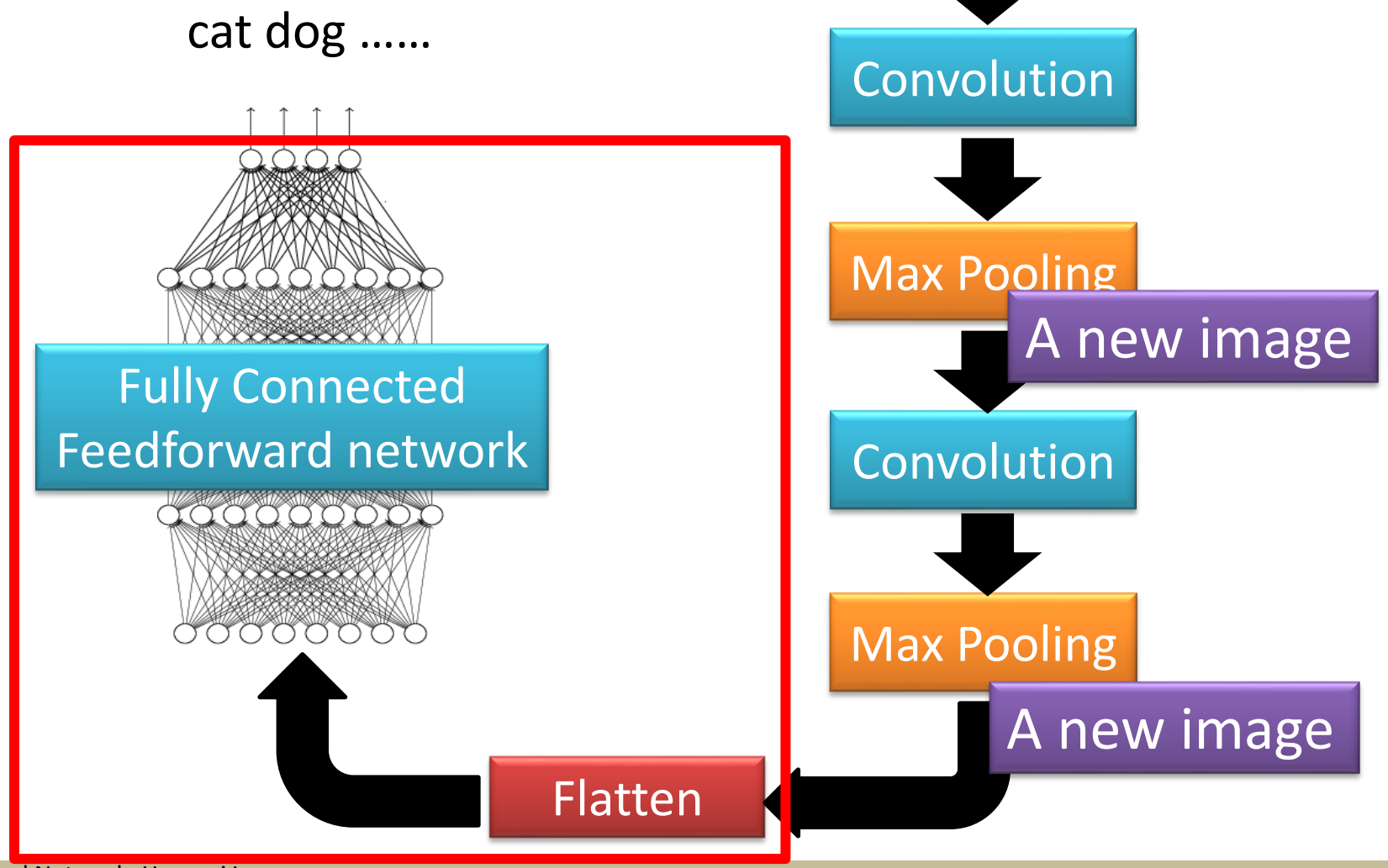
Can repeat many times

Smaller than the original image

The number of the channel is the number of filters

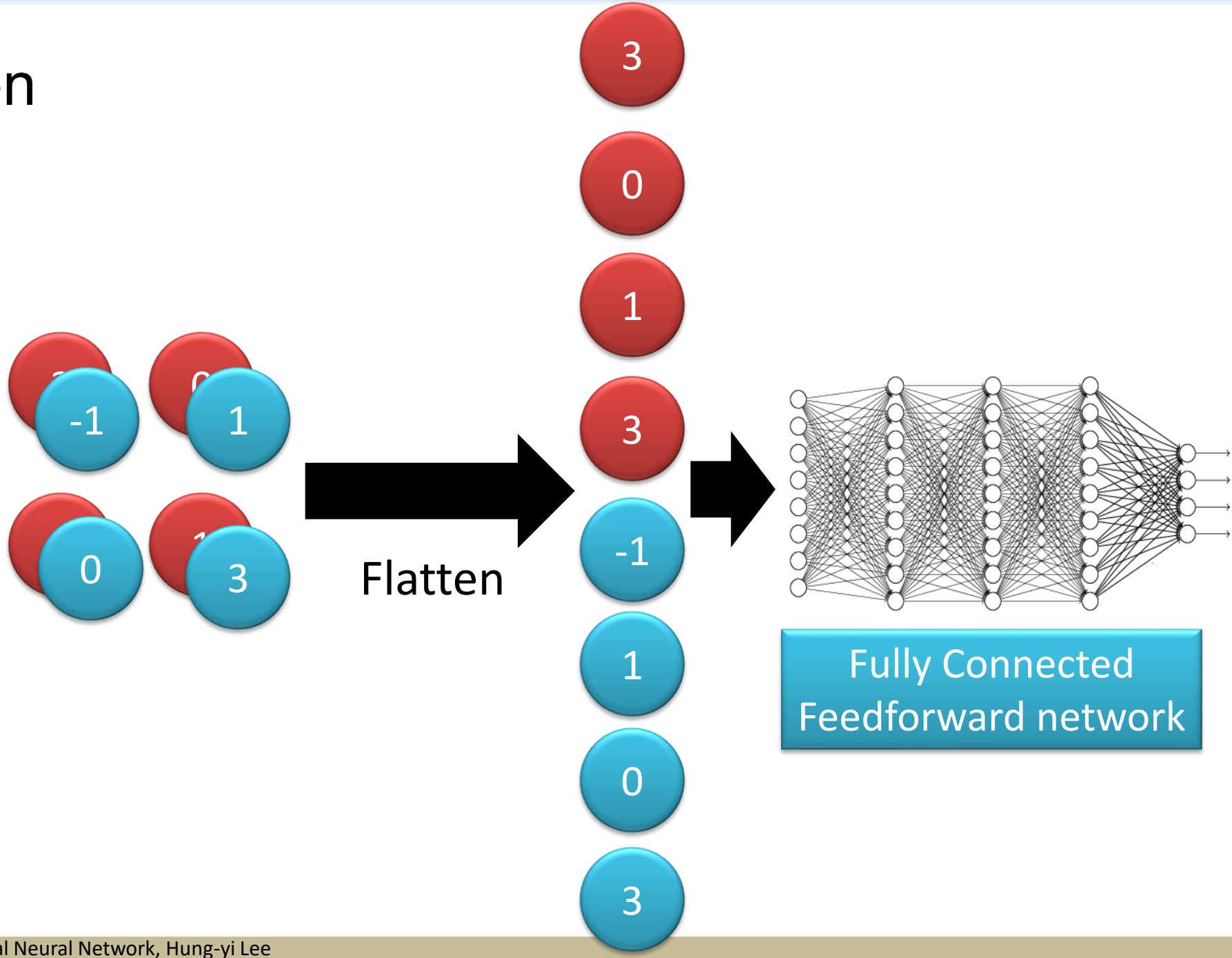
Source: Convolutional Neural Network, Hung-yi Lee

The whole CNN



Source: Convolutional Neural Network, Hung-yi Lee

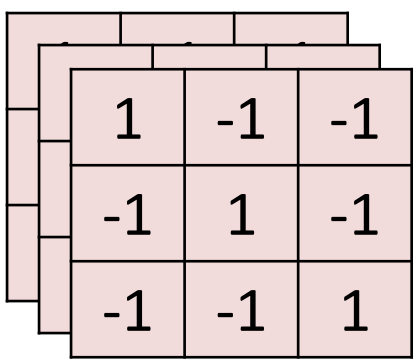
Flatten



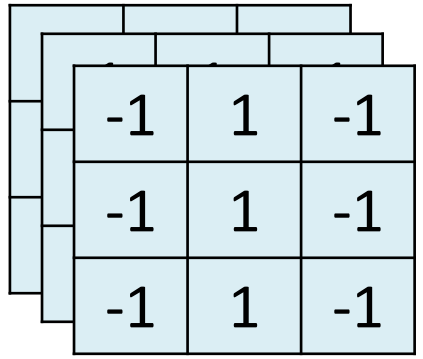
Source: Convolutional Neural Network, Hung-yi Lee

CNN – Colorful image

- Multi-channels

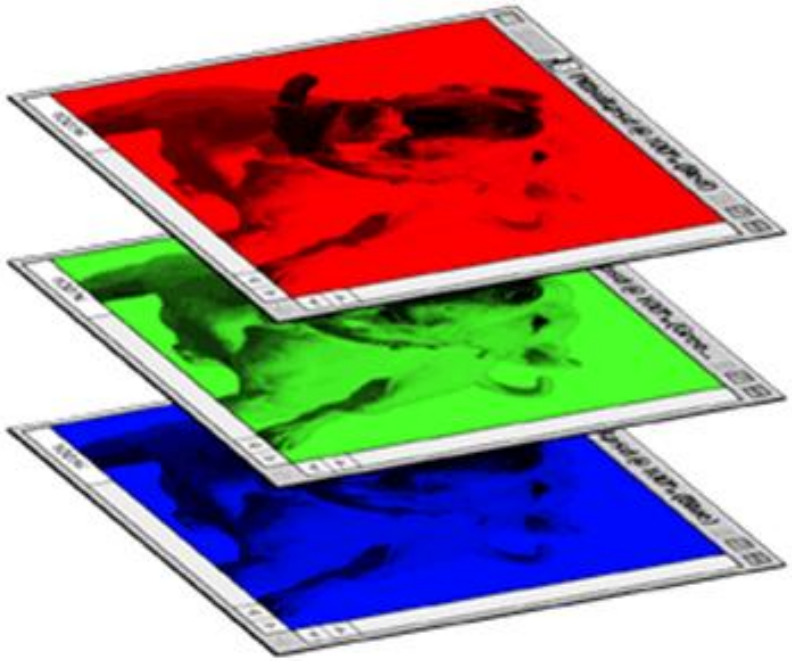


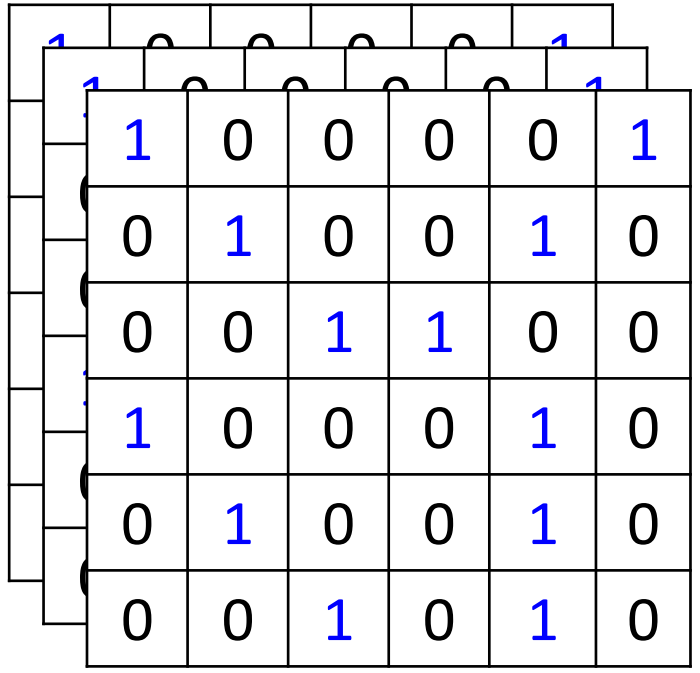
Filter 1



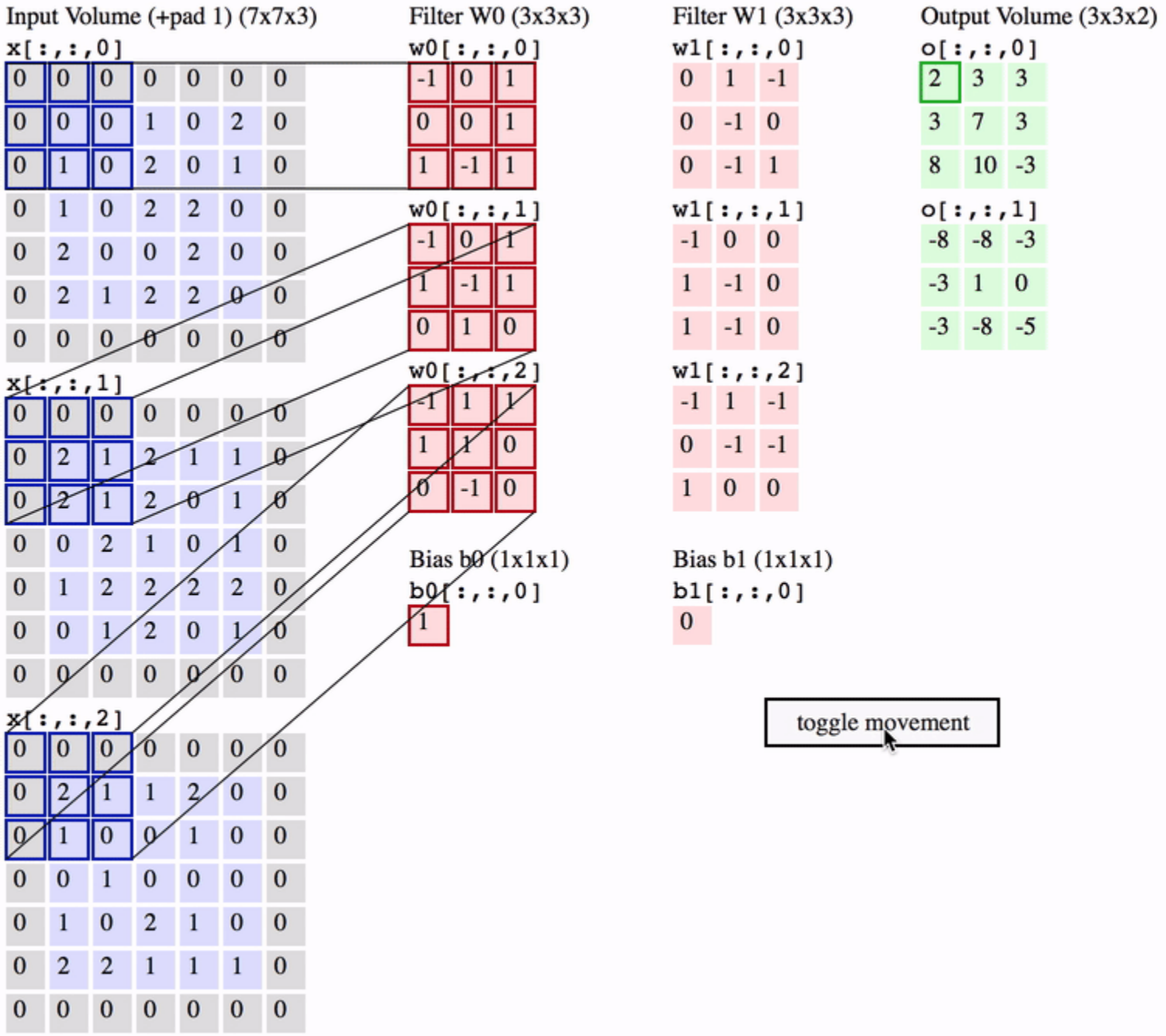
Filter 2

Colorful image



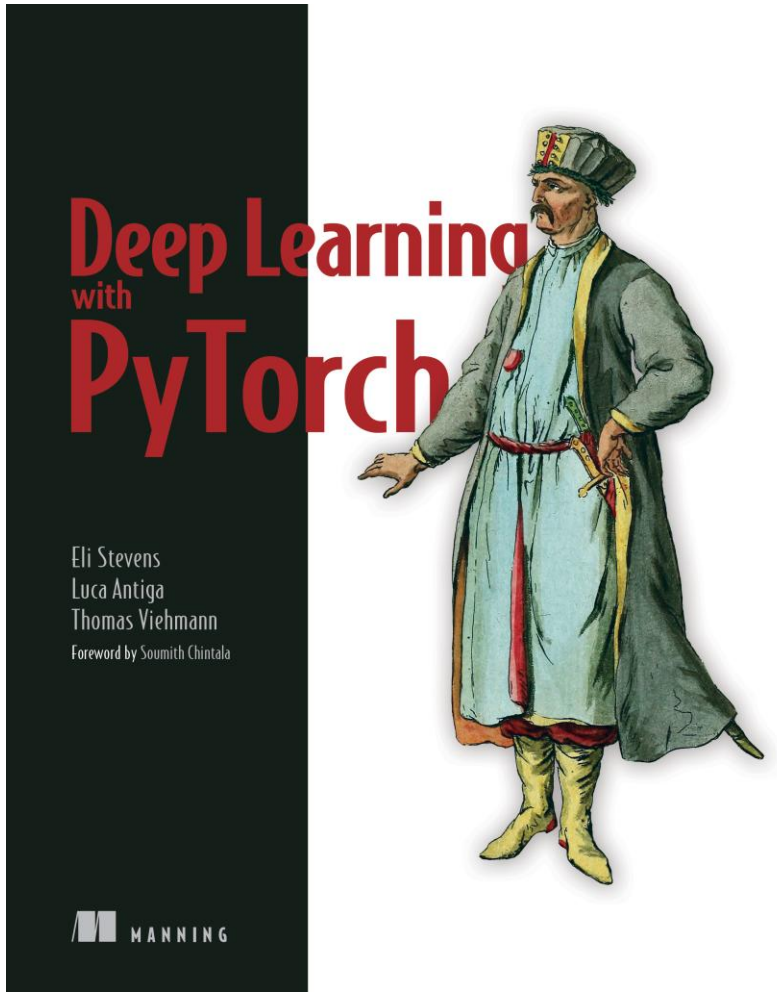


Source: Convolutional Neural Network, Hung-yi Lee



Source: CS231n

PYTORCH (OPTIONAL)



https://isip.piconepress.com/courses/temple/ece_4822/resources/books/Deep-Learning-with-PyTorch.pdf

https://www.learnpytorch.io/

In [1]:
```
import torch
import torch.nn as nn
import torch.nn.functional as F
import torch.optim as optim
from torchvision import datasets, transforms
```

In [2]:
```
torch.manual_seed(4242)
```

Out[2]:
```
<torch._C.Generator at 0x25660166a70>
```

In [3]:
```
train_loader = torch.utils.data.DataLoader(
    datasets.MNIST('../data/p1ch2/mnist', train=True, download=True,
                   transform=transforms.Compose([
                       transforms.ToTensor(),
                       transforms.Normalize((0.1307,), (0.3081,))
                   ])),
    batch_size=64, shuffle=True)
```

```
Downloading http://yann.lecun.com/exdb/mnist/train-images-idx3-ubyte.gz
Downloading http://yann.lecun.com/exdb/mnist/train-labels-idx1-ubyte.gz
Downloading http://yann.lecun.com/exdb/mnist/t10k-images-idx3-ubyte.gz
Downloading http://yann.lecun.com/exdb/mnist/t10k-labels-idx1-ubyte.gz
Processing...
Done!
```

In [4]:
```
class Net(nn.Module):
    def __init__(self):
        super(Net, self).__init__()
        self.conv1 = nn.Conv2d(1, 10, kernel_size=5)
        self.conv2 = nn.Conv2d(10, 20, kernel_size=5)
        self.conv2_drop = nn.Dropout2d()
        self.fc1 = nn.Linear(320, 50)
        self.fc2 = nn.Linear(50, 10)

    def forward(self, x):
        x = F.relu(F.max_pool2d(self.conv1(x), 2))
        x = F.relu(F.max_pool2d(self.conv2_drop(self.conv2(x)), 2))
        x = x.view(-1, 320)
        x = F.relu(self.fc1(x))
        x = F.dropout(x, training=self.training)
        x = self.fc2(x)
        return F.log_softmax(x, dim=1)
```

In [5]:
```
model = Net()
```

In [6]:
```
optimizer = optim.SGD(model.parameters(), lr=0.01, momentum=0.5)
```

In [7]:
```
for epoch in range(10):
    for batch_idx, (data, target) in enumerate(train_loader):
        optimizer.zero_grad()
        output = model(data)
        loss = F.nll_loss(output, target)
        loss.backward()
        optimizer.step()
    print('Current loss', float(loss))
```

```
Current loss 0.4354310631752014
Current loss 0.2379325628280639 6
Current loss 0.382179856300354
Current loss 0.39005237817764 28
Current loss 0.283257395029068
Current loss 0.1536979377269745
Current loss 0.10767409205436707
Current loss 0.14431846141815186
Current loss 0.30025267601013184
Current loss 0.18810895085334778
```

In [8]:
```
torch.save(model.state_dict(), '../data/p1ch2/mnist/mnist.pth')
```

In [9]:
```
pretrained_model = Net()
pretrained_model.load_state_dict(torch.load('../data/p1ch2/mnist/mnist.pth'))
```

Pytorch detach

- The derivative will be 4a^3 + 6a^5.
  - The gradient of a will be 4*2^3 + 6*2^5 = 224. a.grad produces the vector with 20 elements where each element has a value of 224.
- detach: remove a tensor from computational graph, no gradient calculated
  - derivative value will be 3a^2 which is 12

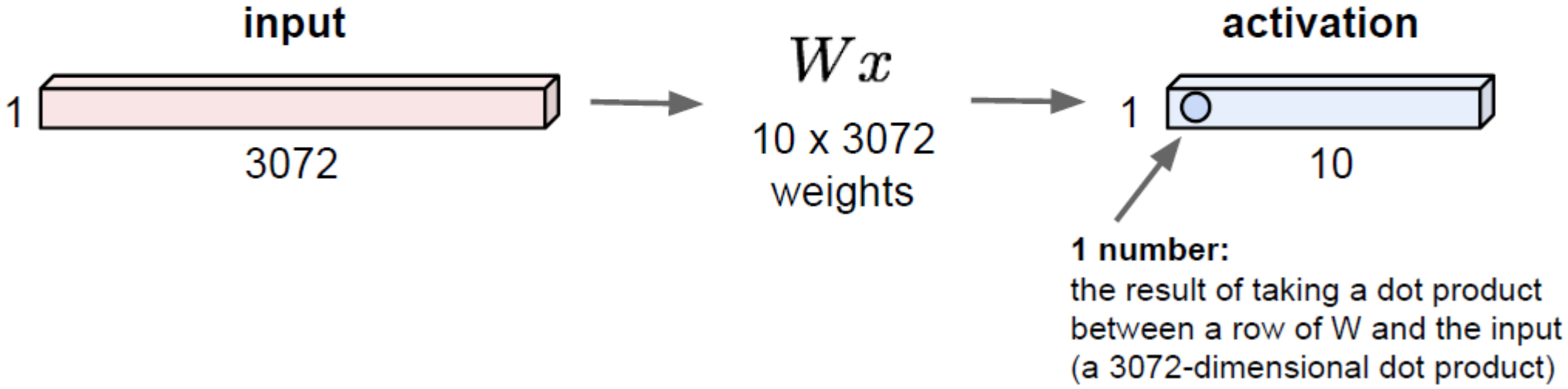
```
a=torch.ones(20, requires_grad=True)
b=a**4
c=a**6
i=(b+c).sum()
i.backward()
print(a.grad)
```

```
a=torch.ones(20, requires_grad=True)
b=a**3
c=a.detach()**6
i=(b+c).sum()
i.backward()
print(a.grad)
```

DIMENSION OF CONVOLUTIONAL LAYER

Fully Connected Layer

32x32x3 image -> stretch to 3072 x 1

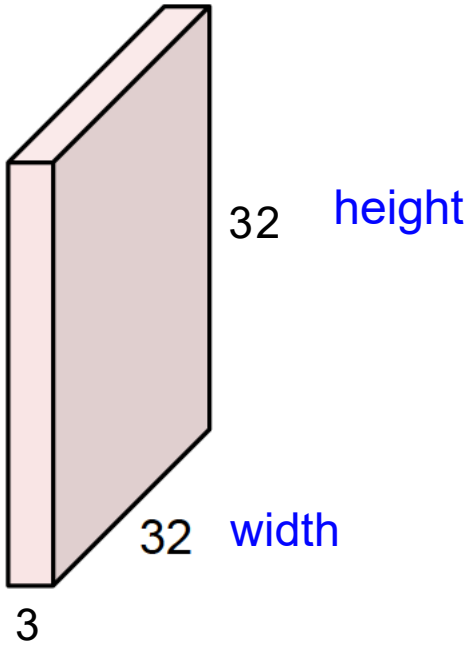


Convolutional Layer

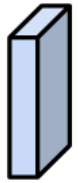
- Depth == channels

Filters always extend the full depth of the input volume

32x32x3 image



5x5x3 filter



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

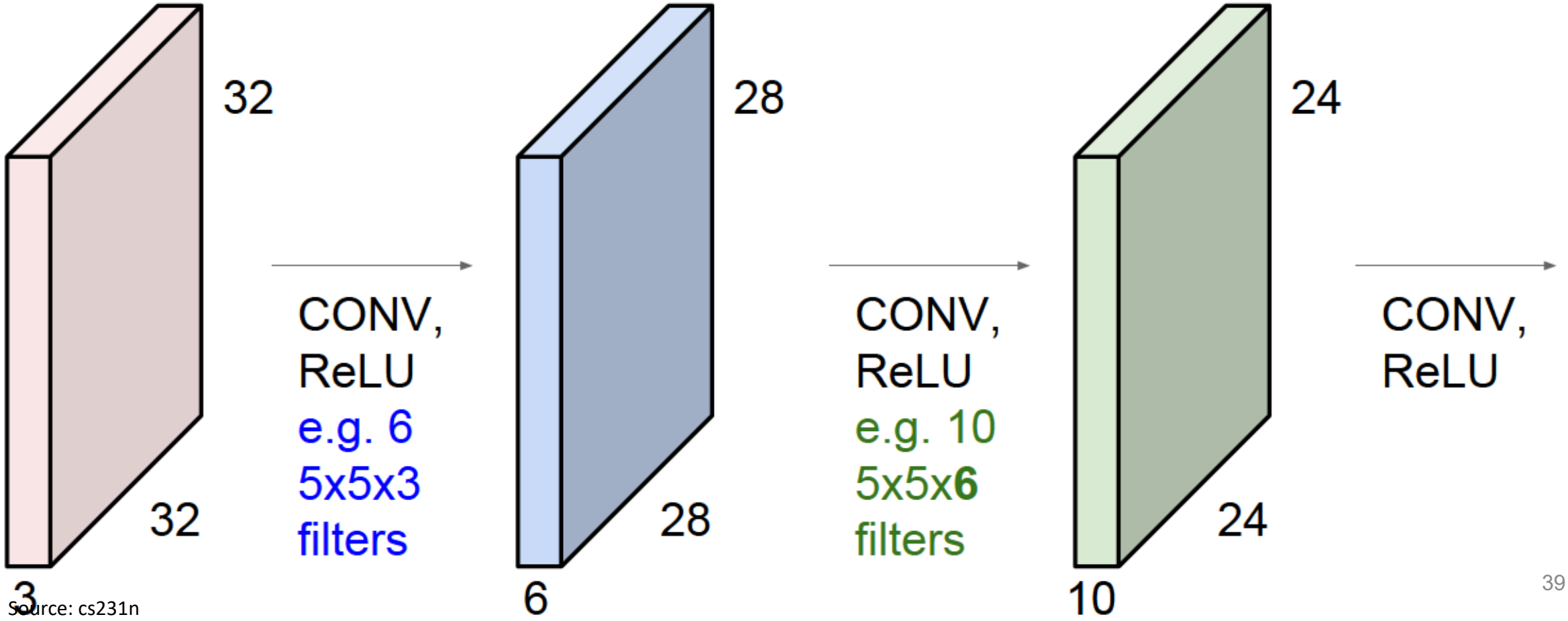
the result of taking a dot product between the filter and a small 5x5x3 chunk of the image
(i.e. 5*5*3 =75-dimensional dot product + bias)

$$w^T x + b$$

Source: cs231n

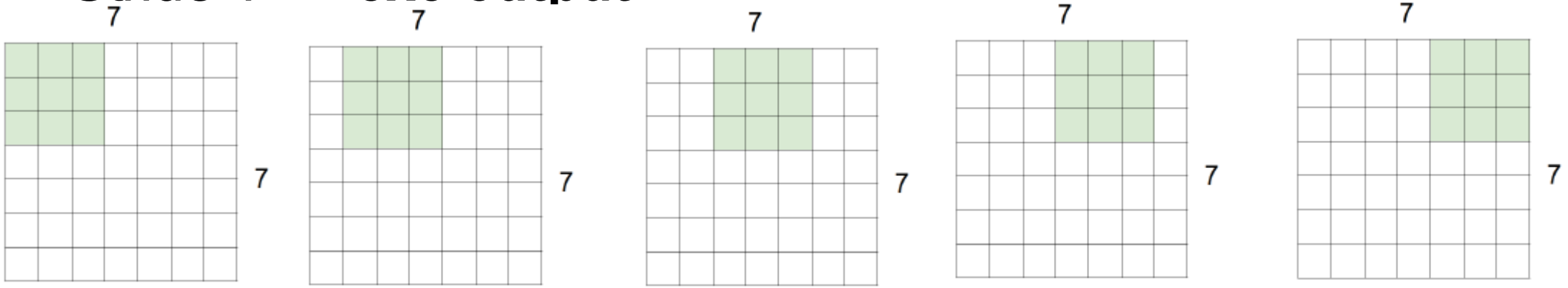
Convolutional NN

Preview: ConvNet is a sequence of Convolutional Layers, interspersed with activation functions

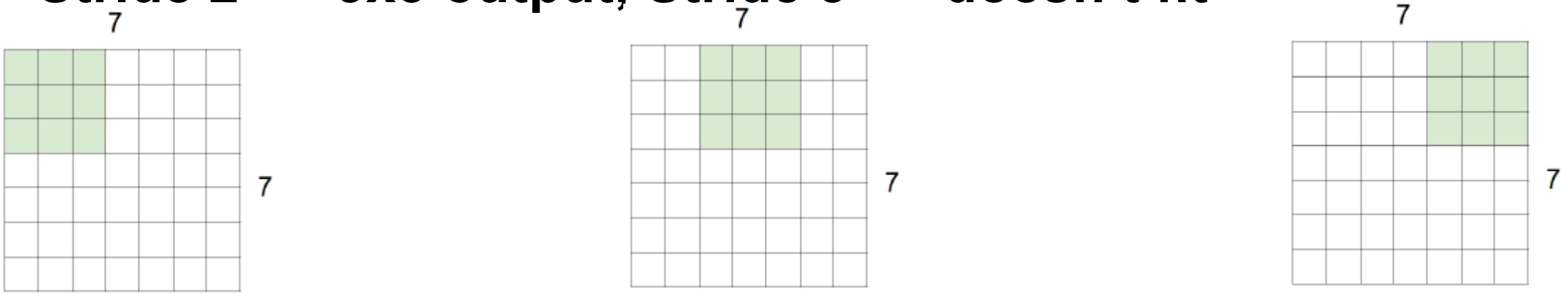


Source: cs231n

- 7x7 input (spatially) assume 3x3 filter
- Stride 1 => **5x5 output**

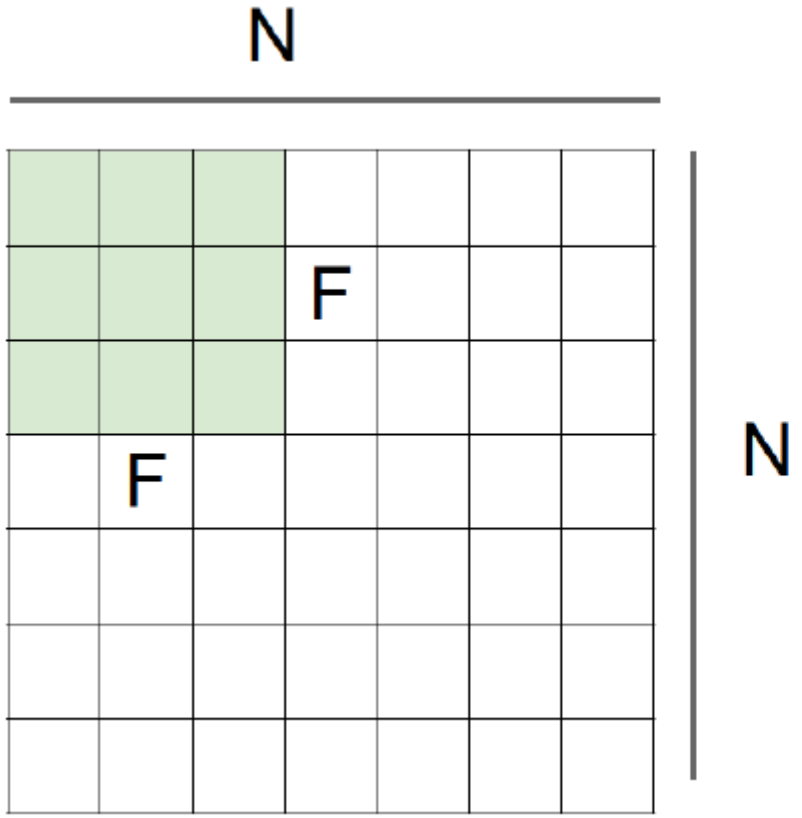


- **Stride 2 => 3x3 output, Stride 3 => doesn't fit**



Source: cs231n

Output Size

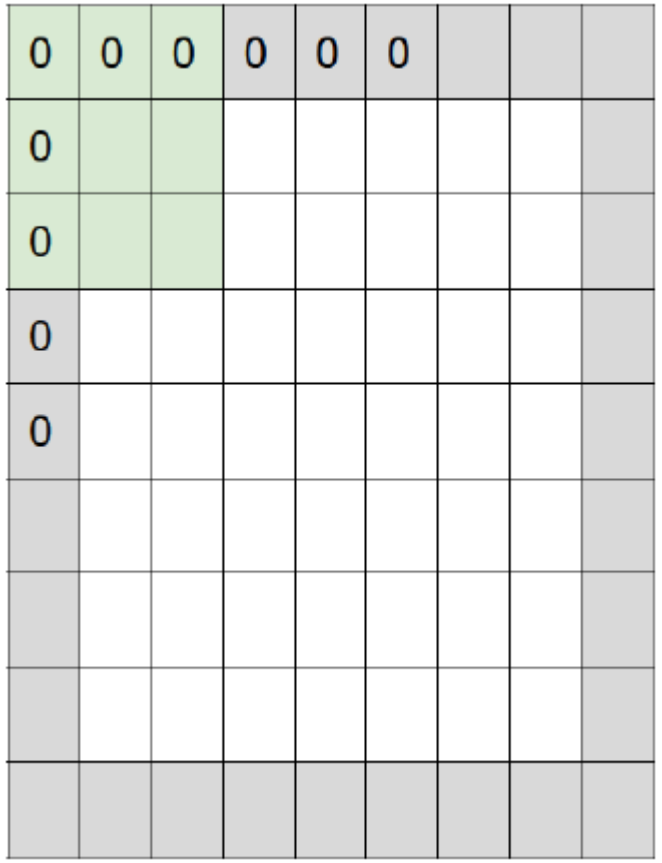


Output size:
(N - F) / stride + 1

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 :\

Source: cs231n

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?

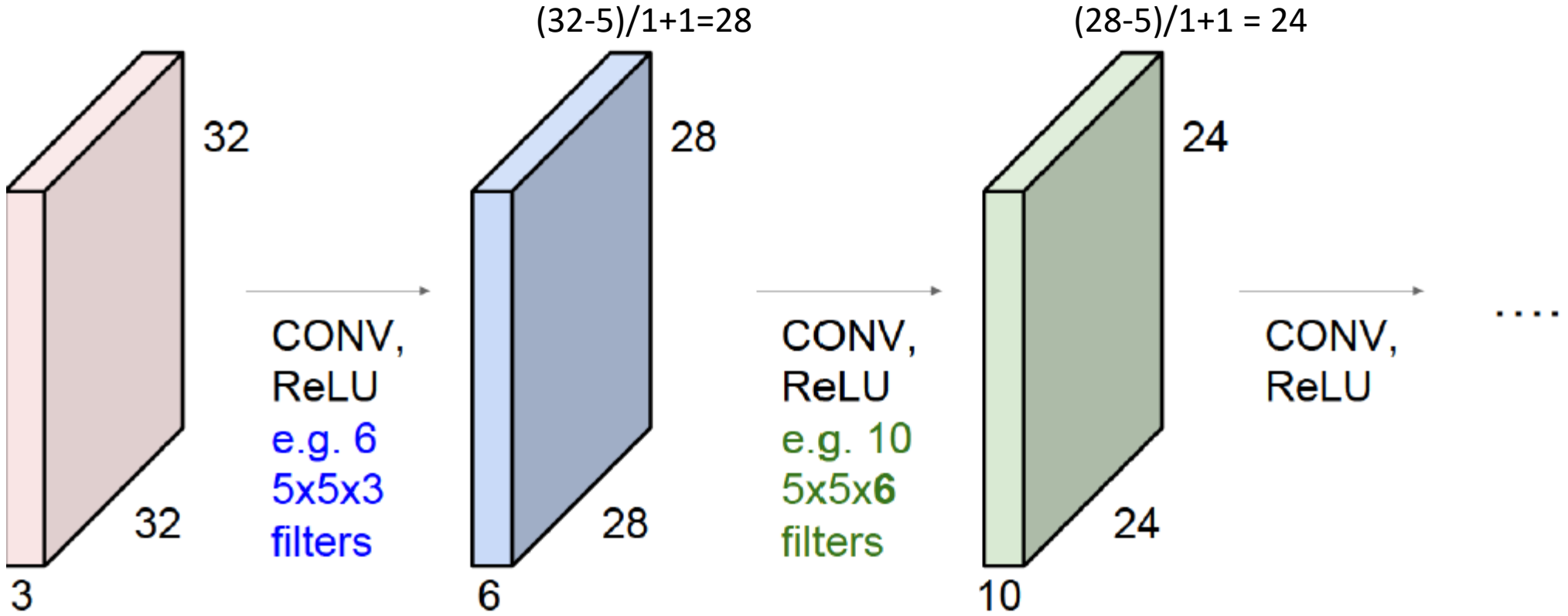
7x7 output!
in general, common to see CONV layers with stride 1, filters of size FxF, and zero-padding with (F-1)/2. (will preserve size spatially)
e.g. F = 3 => zero pad with 1
F = 5 => zero pad with 2
F = 7 => zero pad with 3

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

Source: cs231n

Remember back to…

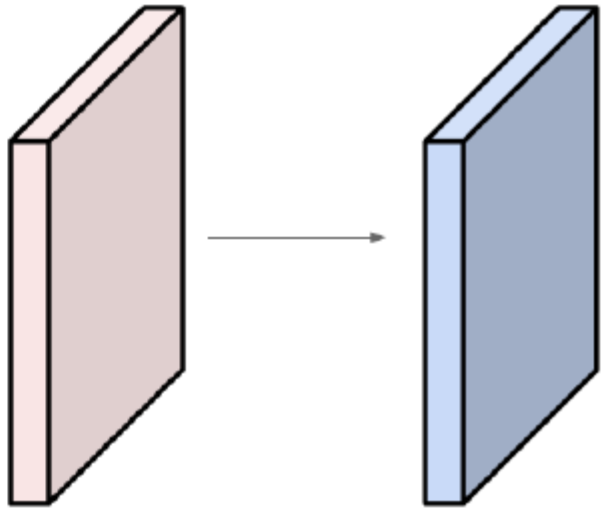
E.g. 32x32 input convolved repeatedly with 5x5 filters shrinks volumes spatially! (32 -> 28 -> 24 ...). Shrinking too fast is not good, doesn't work well.



Source: cs231n

Output volume size

Examples time:



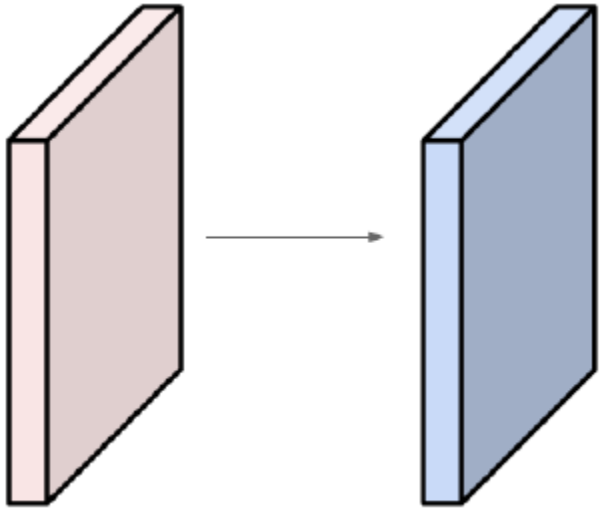
Input volume: **32x32x3**
10 5x5 filters with stride 1, pad 2

Output volume size:
(32+2*2-5)/1+1 = 32 spatially, so
32x32x10

Source: cs231n

Parameter Numbers

Examples time:



Input volume: **32x32x3**
10 5x5 filters with stride 1, pad 2

Number of parameters in this layer?
each filter has 5*5*3 + 1 = 76 params (+1 for bias)
=> 76*10 = **760**

Source: cs231n

Common settings:

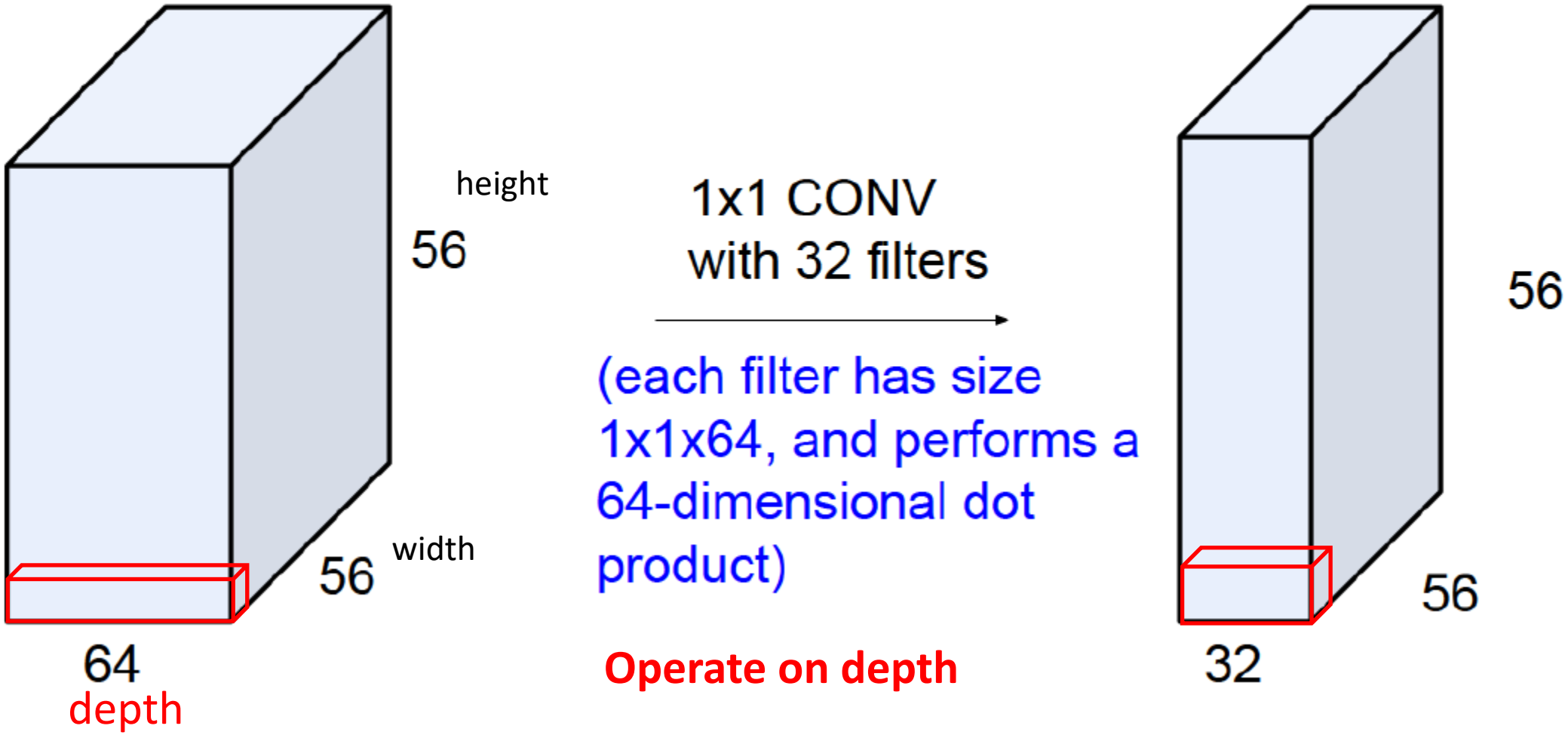
K = (powers of 2, e.g. 32, 64, 128, 512)
- F = 3, S = 1, P = 1
- F = 5, S = 2, P = 2
- F = 5, S = 2, P = ? (whatever fits)
- F = 1, S = 1, P = 0

Summary. To summarize, the Conv Layer:

- Accepts a volume of size $W_1 \times H_1 \times D_1$
- Requires four hyperparameters:
  - Number of filters K,
  - their spatial extent F,
  - the stride S,
  - the amount of zero padding P.
- Produces a volume of size $W_2 \times H_2 \times D_2$ where:
  - $W_2 = (W_1 - F + 2P)/S + 1$
  - $H_2 = (H_1 - F + 2P)/S + 1$ (i.e. width and height are computed equally by symmetry)
  - $D_2 = K$
- With parameter sharing, it introduces $F \cdot F \cdot D_1$ weights per filter, for a total of $(F \cdot F \cdot D_1) \cdot K$ weights and K biases.
- In the output volume, the d-th depth slice (of size $W_2 \times H_2$) is the result of performing a valid convolution of the d-th filter over the input volume with a stride of S, and then offset by d-th bias.

Source: cs231n

(btw, 1x1 convolution layers make perfect sense)

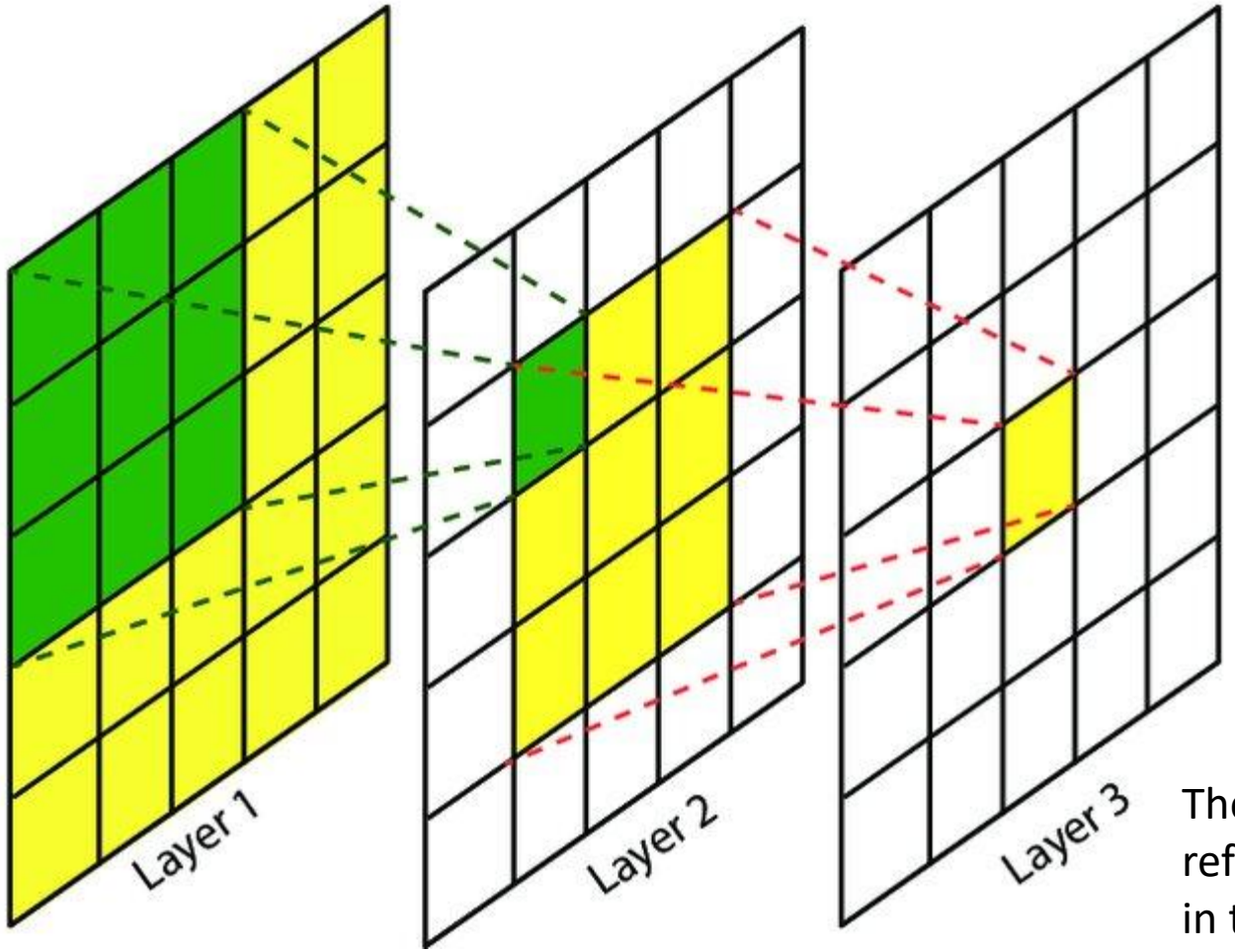


Source: cs231n

RECEPTIVE FIELD

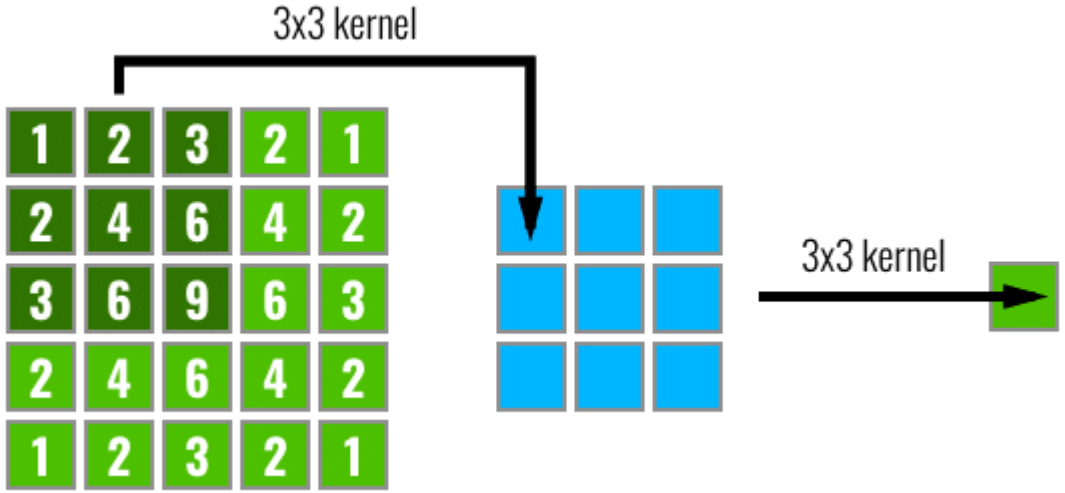
https://distill.pub/2019/computing-receptive-fields/#solving-receptive-field-region

Receptive field



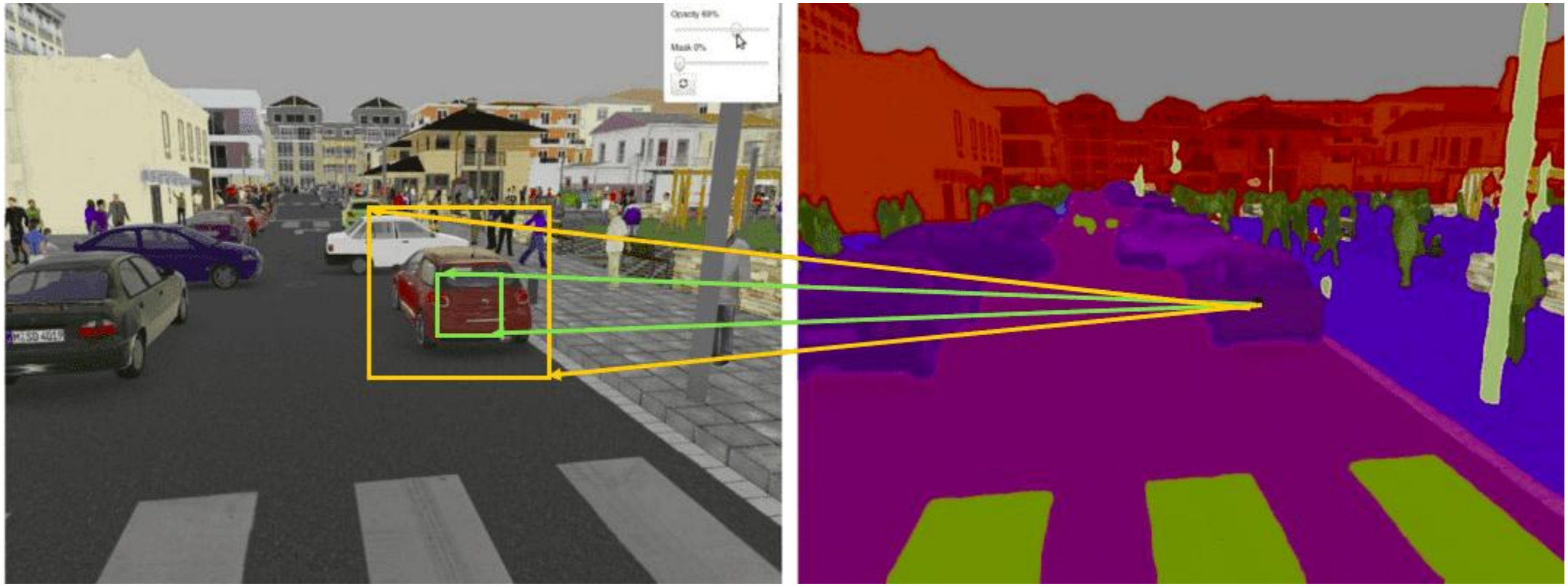
5x5 3x3 1

影響輸出值的輸入範圍



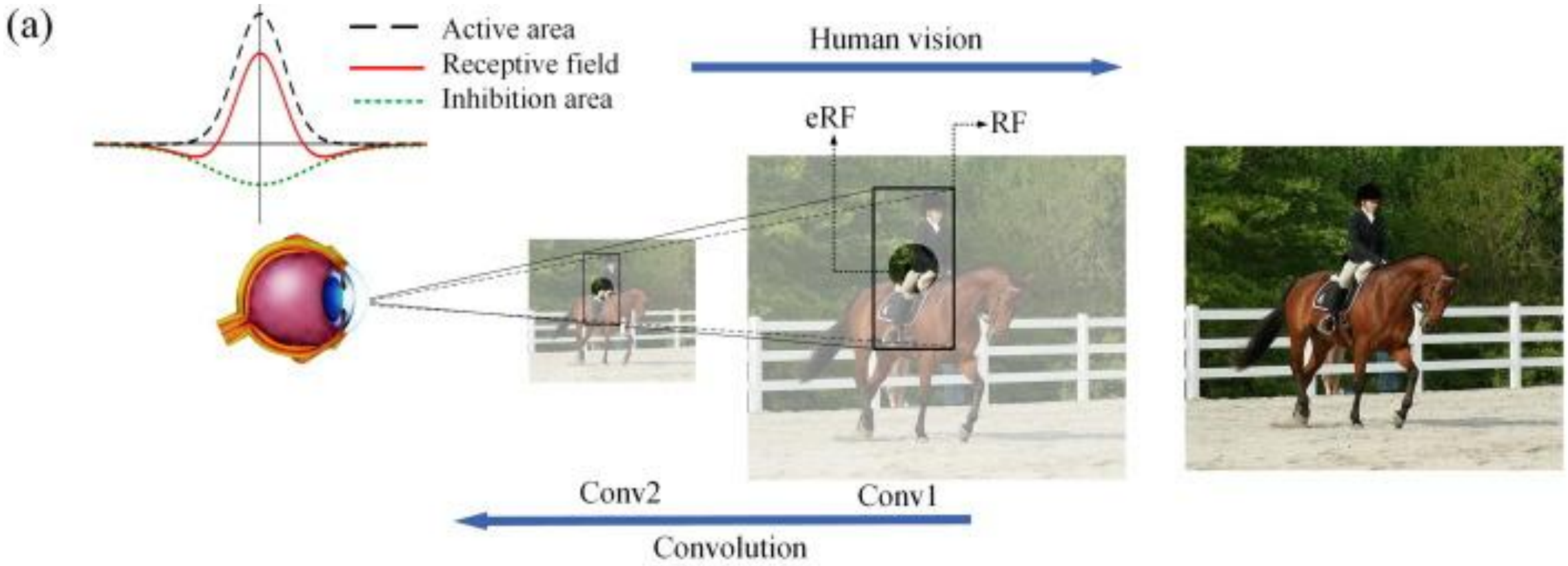
The receptive field (RF) of a Convolutional Neural Network (CNN) refers to the region in the input image that a particular feature in the CNN responds to. In other words, it is the spatial extent of the input data which affects the value of a given pixel in the output feature map.

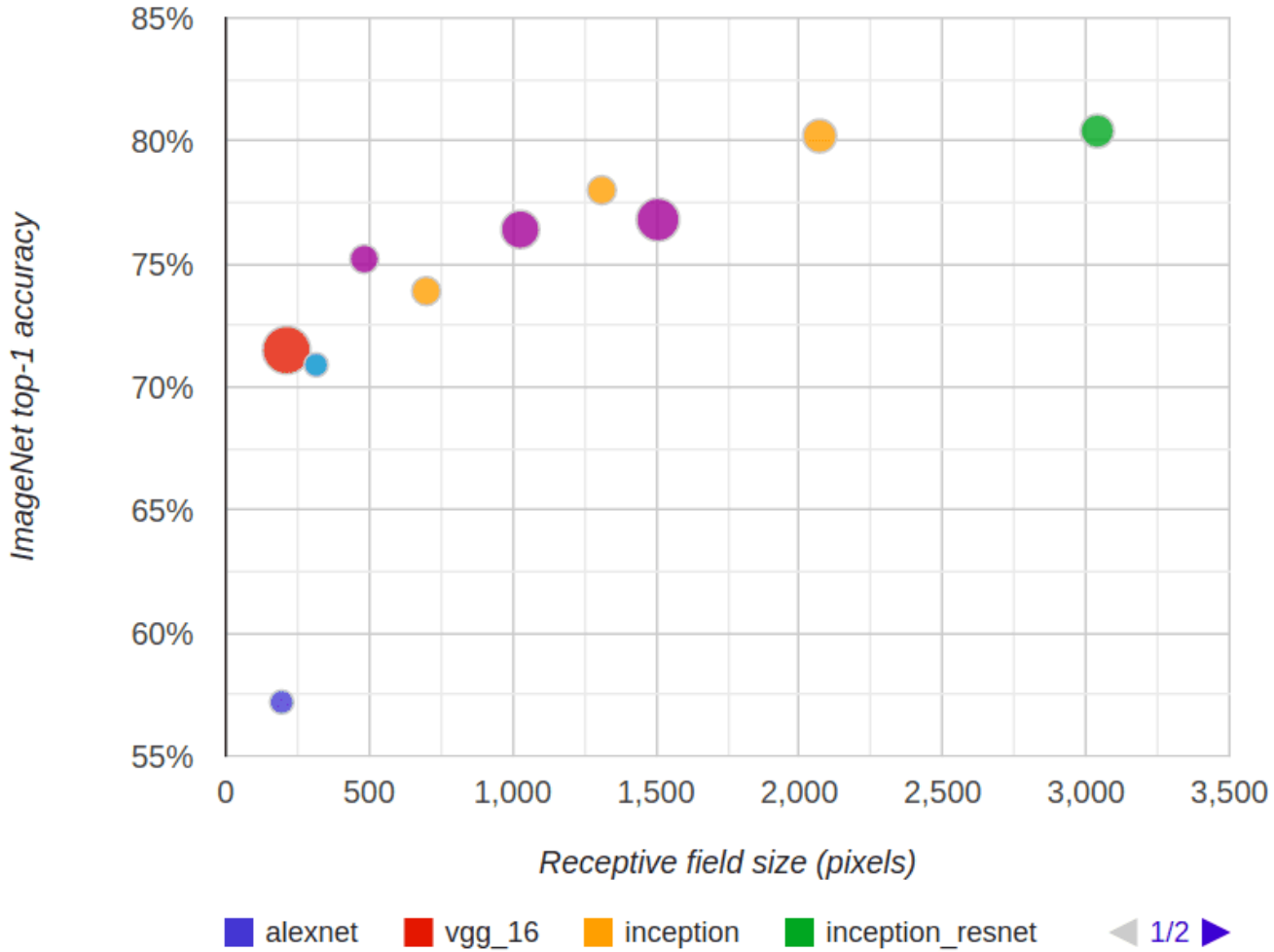
Why do we care about the receptive field of a convolutional network?



The green and orange rectangles are two different receptive fields. Which one would you prefer? The green and orange rectangles are two different receptive fields. Which one would you prefer? Source: [Nvidia's blog]

https://theaisummer.com/receptive-field/


(a)
Active area
Receptive field
Inhibition area
Human vision
eRF
RF
Conv2
Conv1
Convolution



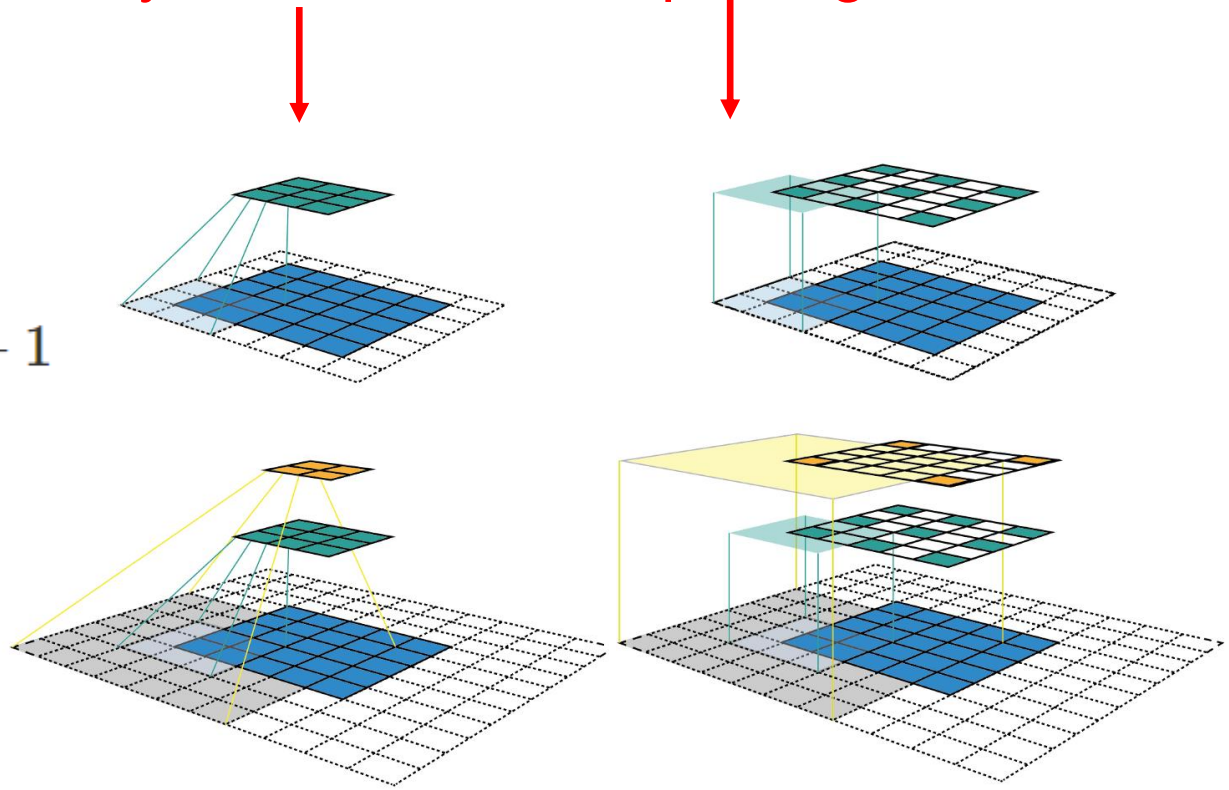
"We observe a logarithmic relationship between classification accuracy and receptive field size, which suggests that large receptive fields are necessary for high-level recognition tasks, but with diminishing rewards."

Receptive Field of Convolutional Layer

- receptive field
  - The region in the input space that a particular CNN's feature is looking at
  - How to increase: multi-layer convolution or pooling or dilated convolution

$$r_{l-1} = s_l \cdot r_l + (k_l - s_l)$$

$$r_0 = \sum_{l=1}^{L} \left((k_l - 1) \prod_{i=1}^{l-1} s_i \right) + 1$$



Kernel size k, stride s,

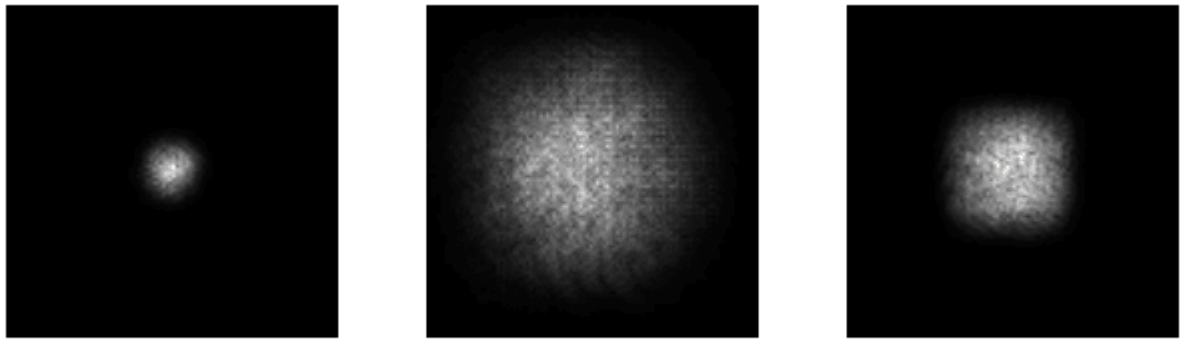
3 3x3 layers

R0= (3-1) + (3-1) + 1 = 5=> 5x5

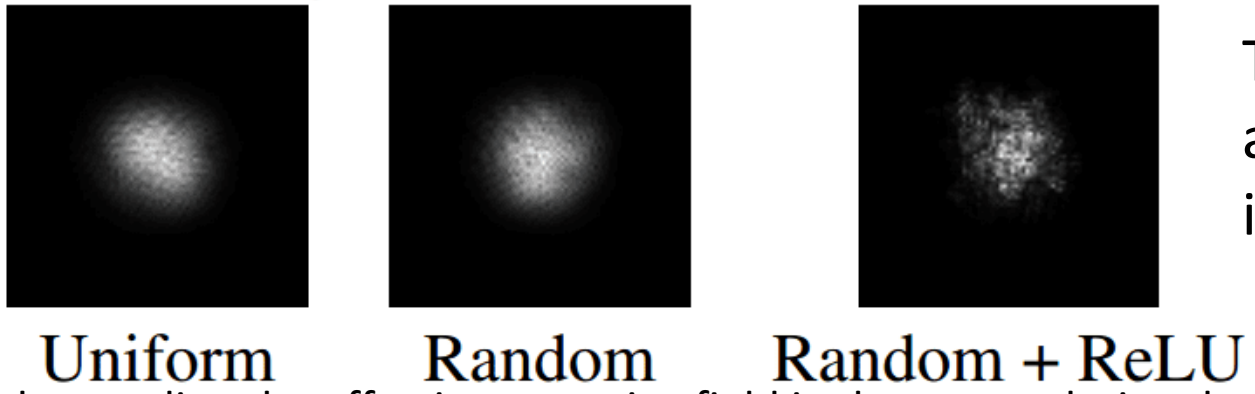
https://distill.pub/2019/computing-receptive-fields/#solving-receptive-field-region

Understanding the effective receptive field

- **not all pixels in a receptive field contribute equally** to an output unit's response



Insight: pooling operations and dilated convolutions turn out to be effective ways to increase the receptive field size quickly.



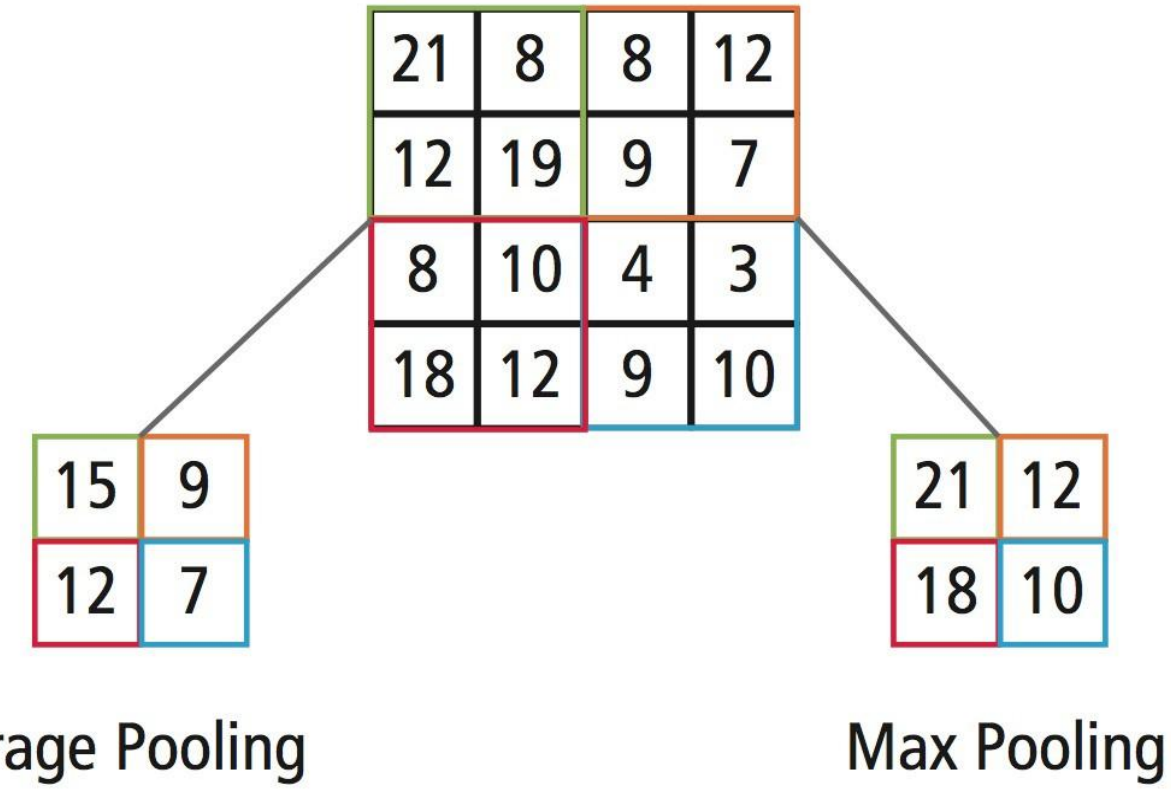
The ERF in deep convolutional networks actually grows a lot slower than we calculate in theory

Understanding the effective receptive field in deep convolutional neural networks

POOLING LAYER

Max Pooling v.s. Average Pooling

- Max pooling is favored for better performance



All Convolutional Network

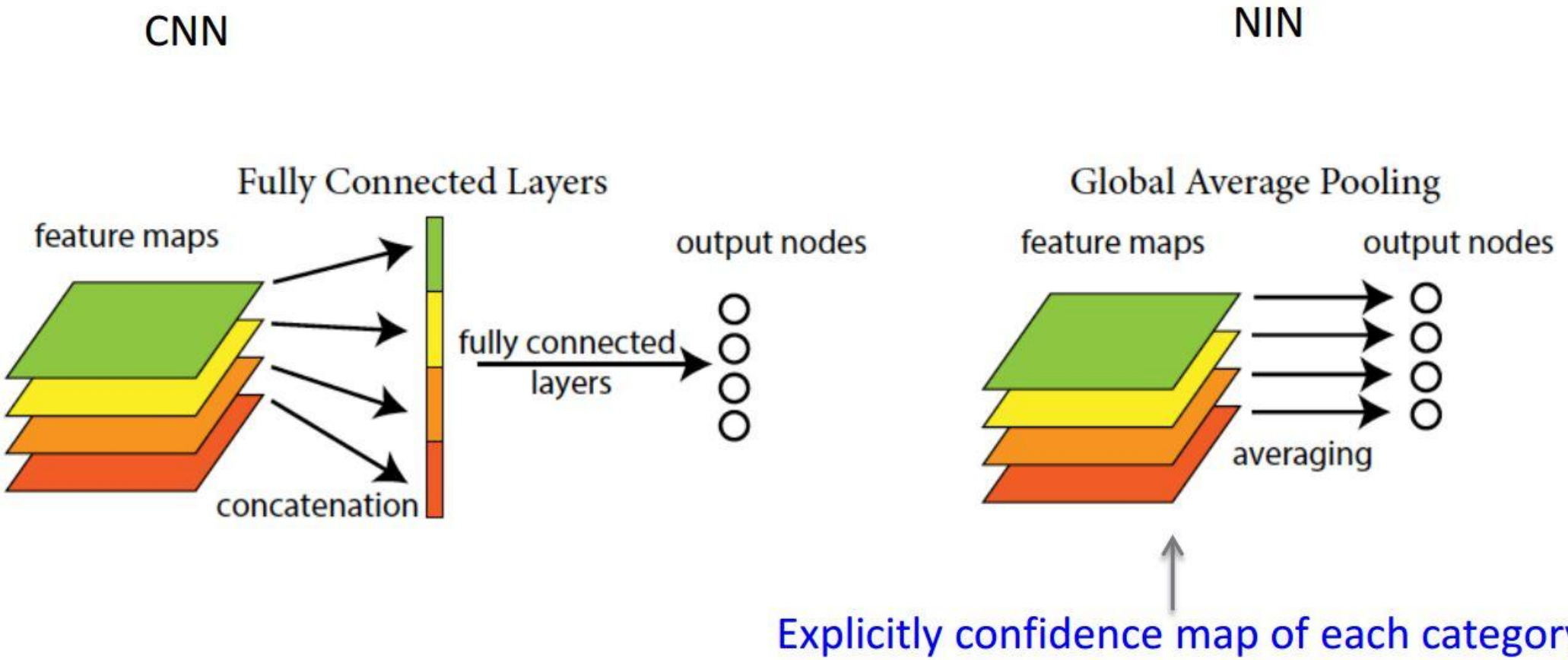
Strided-CNN-C	ConvPool-CNN-C	All-CNN-C	C
	Input 32 × 32 RGB image		
3 × 3 conv. 96 ReLU 3 × 3 conv. 96 ReLU with stride $r = 2$	3 × 3 conv. 96 ReLU 3 × 3 conv. 96 ReLU 3 × 3 conv. 96 ReLU	3 × 3 conv. 96 ReLU 3 × 3 conv. 96 ReLU	3 × 3 conv. 96 ReLU 3 × 3 conv. 96 ReLU
	3 × 3 max-pooling stride 2	3 × 3 conv. 96 ReLU with stride $r = 2$	3 × 3 max-pooling stride 2
3 × 3 conv. 192 ReLU 3 × 3 conv. 192 ReLU with stride $r = 2$	3 × 3 conv. 192 ReLU 3 × 3 conv. 192 ReLU 3 × 3 conv. 192 ReLU	3 × 3 conv. 192 ReLU 3 × 3 conv. 192 ReLU	3 × 3 conv. 192 ReLU 3 × 3 conv. 192 ReLU
	3 × 3 max-pooling stride 2	3 × 3 conv. 192 ReLU with stride $r = 2$	3 × 3 max-pooling stride 2
	3 × 3 conv. 192 ReLU		
	1 × 1 conv. 192 ReLU		
	1 × 1 conv. 10 ReLU		
	global averaging over 6 × 6 spatial dimensions		
	10 or 100-way softmax		

Model C	9.74%	≈ 1.3 M
Strided-CNN-C	10.19%	≈ 1.3 M
ConvPool-CNN-C	9.31%	≈ 1.4 M
ALL-CNN-C	**9.08%**	≈ 1.4 M

CIFAR-10

Global Average Pooling

- Save a large portion of parameters



OTHER LAYERS

Other Layers

- Normalization layers
  - E.g. Local contrast normalization
  - fallen out of favor because in practice their contribution has been shown to be minimal, if any.
- Output layers
  - Cross-entropy, softmax
- Fully connected layers
  - Just a normal neural network
- Conversion between FC and Conv layer
  - Conv => FC (link)
    - The weight matrix would be a large matrix that is mostly zero except for at certain blocks (due to local connectivity) where the weights in many of the blocks are equal (due to parameter sharing)
  - FC => Conv: particularly useful in practice

Source: cs231n

Conversion from FC => Conv Layers

- Notes.
  - Input $W_1 \times H_1 \times D_1$
  - Parameters:
    - Number of filters *K*, their spatial extent *F*, the stride *S*, the amount of zero padding P
  - Output $W_2 \times H_2 \times D_2$
    - $W_2 = (W_1 - F + 2P)/S + 1$
    - $H_2 = (H_1 - F + 2P)/S + 1$
    - $D_2 = K$
- Example: The first FC layer in AlextNet
  - Input 7x7x512: Conversion: $F = 7, P = 0, S = 1, K = 4096$
    - setting the filter size to be exactly the size of the input volume
  - 1x1x4096: $F = 1$
  - 1x1x4096: $F = 1$
  - 1x1x1000

K: channel number

Why FC => Conv ? No need for fixed input

- Save computation for different input size
- Example: AlexNet
  - Assume a network (AlexNet)
    - 224x224 image => conv => [7x7x512] (i.e. a reduction by 32) => FC=> [1x1x1000]
  - Now apply image with 384x384 size to above network
    - 384x384 => /32 => [12x12x512] => converted FC layers => [6x6x1000]
  - Note.
    - Evaluating the original ConvNet (with FC layers) independently across 224x224 crops of the 384x384 image in strides of 32 pixels gives an identical result to forwarding the converted ConvNet one time.
  - 36 v.s. 1 computation
    - Forwarding the converted ConvNet a single time is much more efficient than iterating the original ConvNet over all those 36 locations, since the 36 evaluations share computation

Source: cs231n

Layer Patterns

- Prefer a stack of small filter CONV to one large receptive field CONV layer

E.g. stack three 3x3 CONV layers

- Receptive field: 3x3 => 5x5 =>7x7,
- Can we replace this with a 7x7 CONV? No.
  - Three stacks of CONV layers with nonlinearity are more expressive
  - Computation:
    - single 7x7 filter has $C \times (7 \times 7 \times C) = 49C^2$ parameters
    - Three 3x3 filters have $3 \times (C \times (3 \times 3 \times C)) = 27C^2$ parameters
- Recent departures: New paradigm like GoogleNet and Residual Net
- In practice: use whatever works best on ImageNet
  - download a pretrained model and finetune it on your data

Source: cs231n

Layer Size Patterns

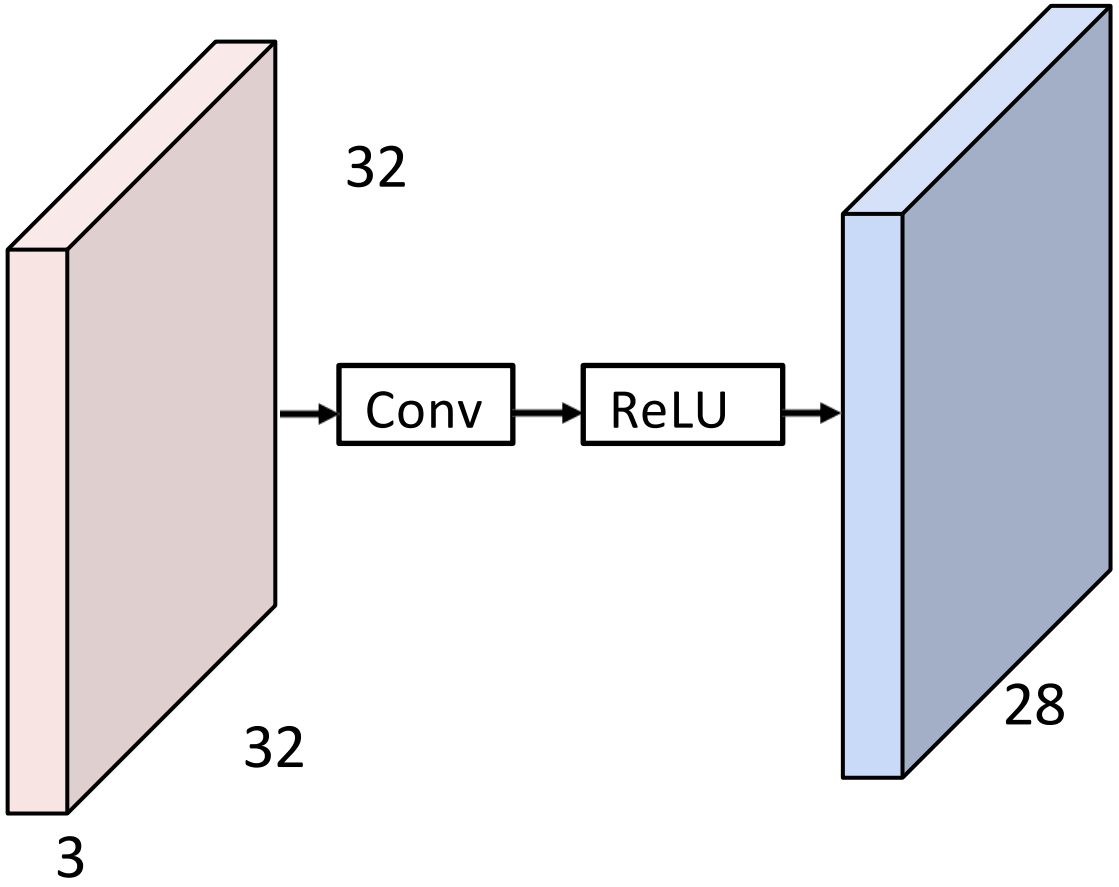
- Input layer: for image, divisible by 2 many times
  - 32 (e.g. CIFAR-10), 64, 96 (e.g. STL-10), or 224 (e.g. common ImageNet ConvNets), 384, and 512
- Conv layer
  - small filters (e.g. 3x3 or at most 5x5), using a stride of S=1, and padding to keep size unchanged
  - bigger filter sizes (7x7 or so) only on the very first conv layer at input
  - Smaller strides (= 1) work better in practice
- Pooling layer
  - most common setting: max-pooling with **2x2** and stride = 2
- Compromising based on memory constraints
  - 7x7 with stride=2 in the first Conv filter of ZFnet
  - 11x11 with stride =4 in AlexNet

Source: cs231n

WHAT CNN LEARNS

See some demo at http://cs.stanford.edu/people/karpathy/convnetjs/

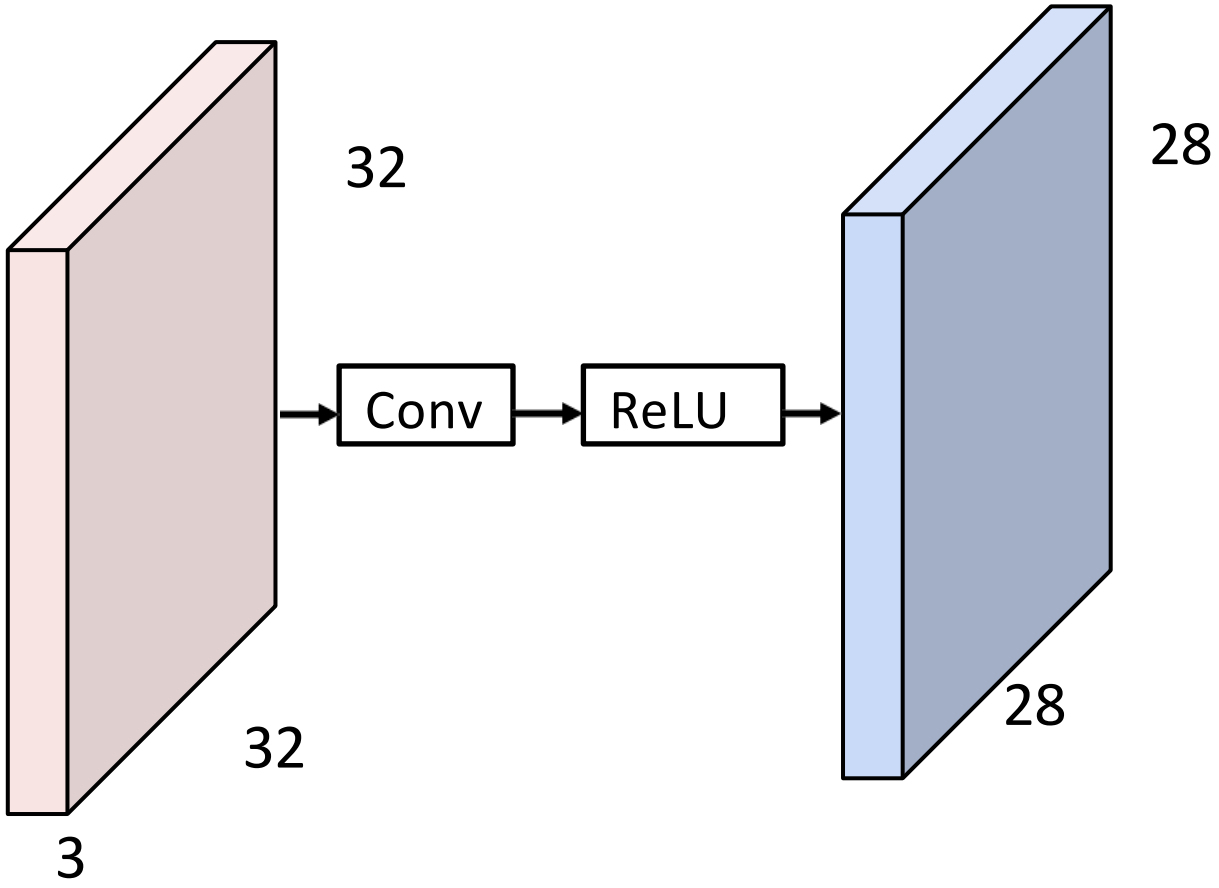
Preview: What do convolutional filters learn?



Linear classifier: One template per class

Source: CS231n

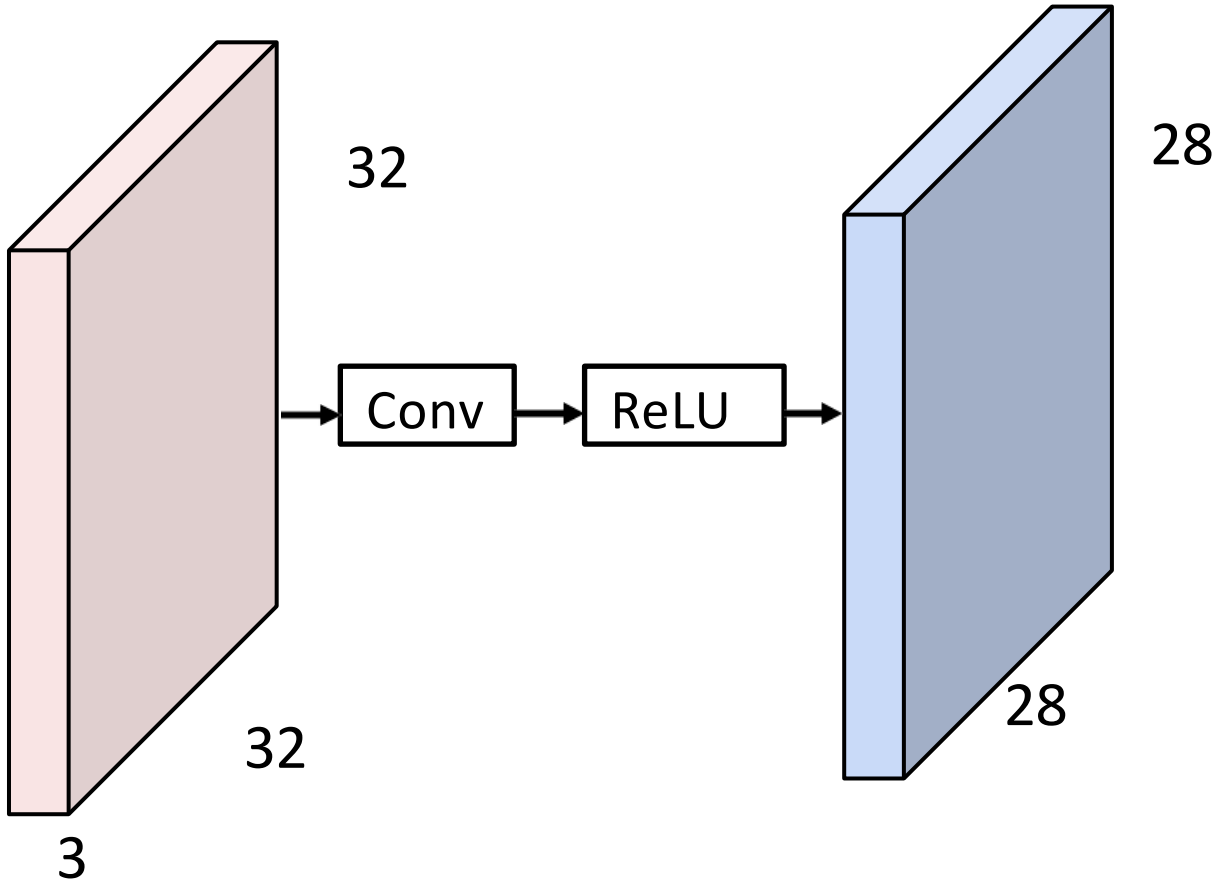
Preview: What do convolutional filters learn?



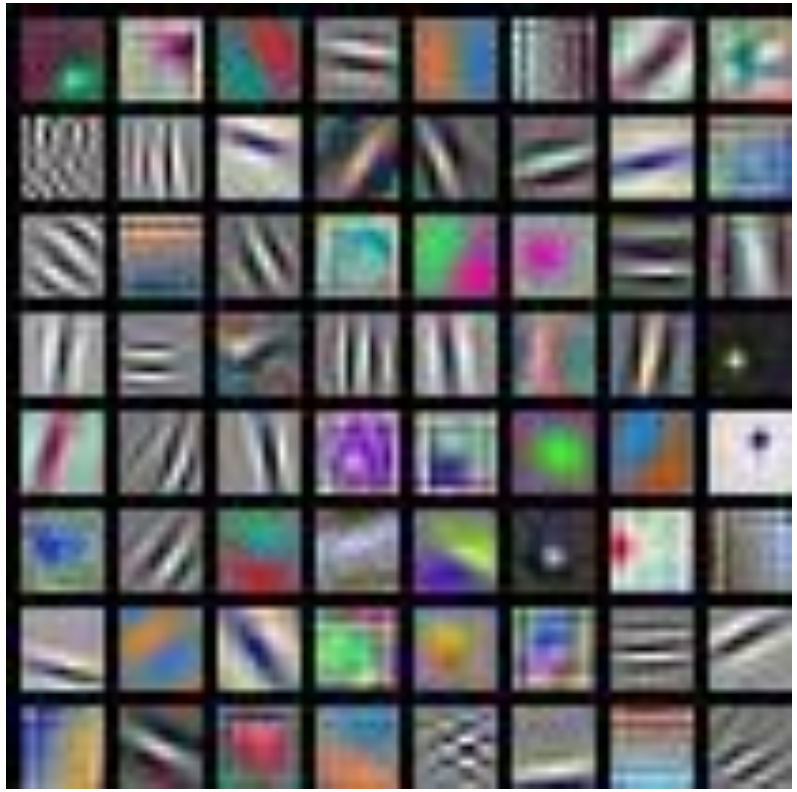
MLP: Bank of whole-image templates

Source: CS231n

Preview: What do convolutional filters learn?



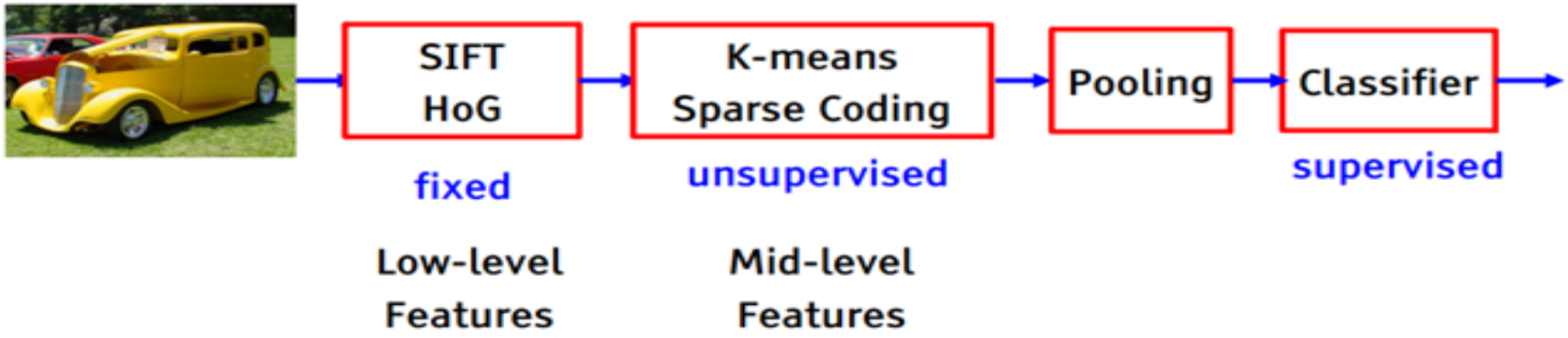
First-layer conv filters: local image templates
(Often learns oriented edges, opposing colors)



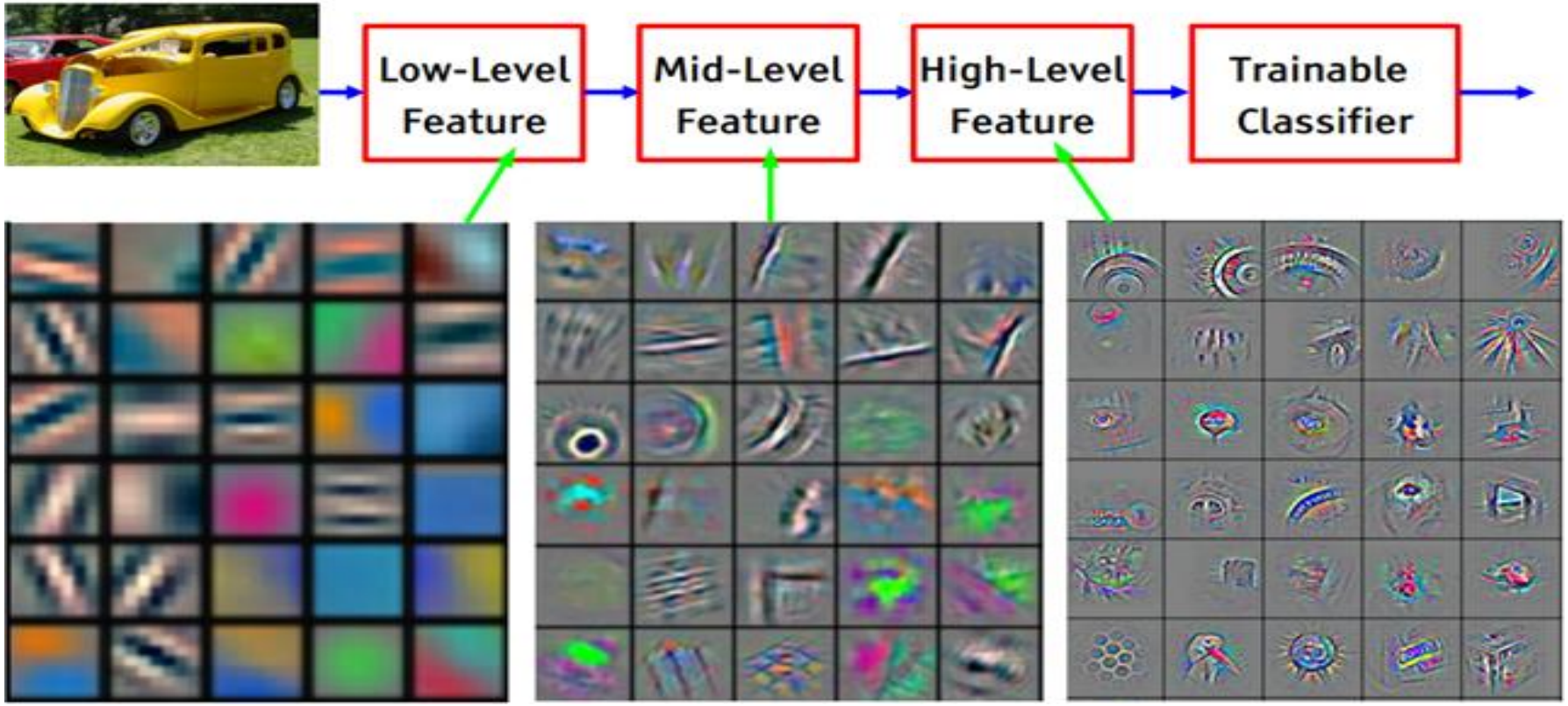
AlexNet: 64 filters, each 3x11x11

Source: CS231n

Object recognition 2006-2012



State of the art object recognition using CNNs

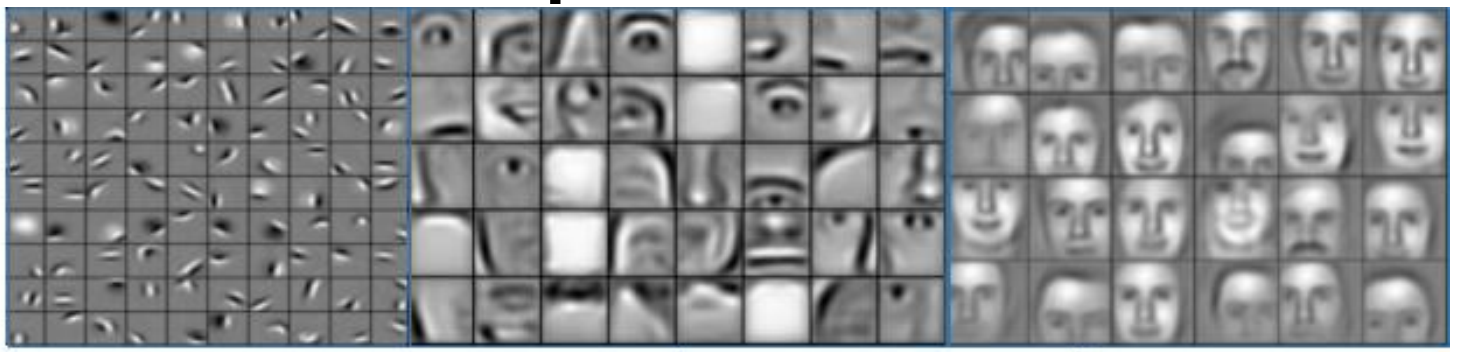


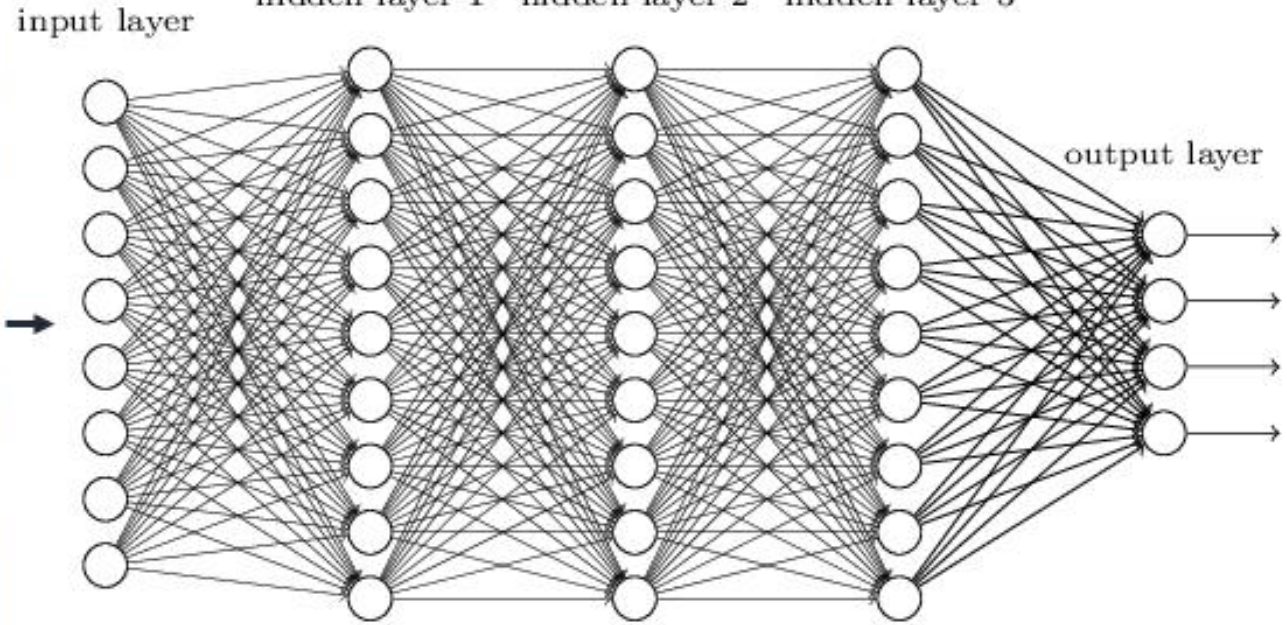
Source: Yann Lecun

DL到底學了什麼?越深越好嗎?

Hierarchical Feature Representations

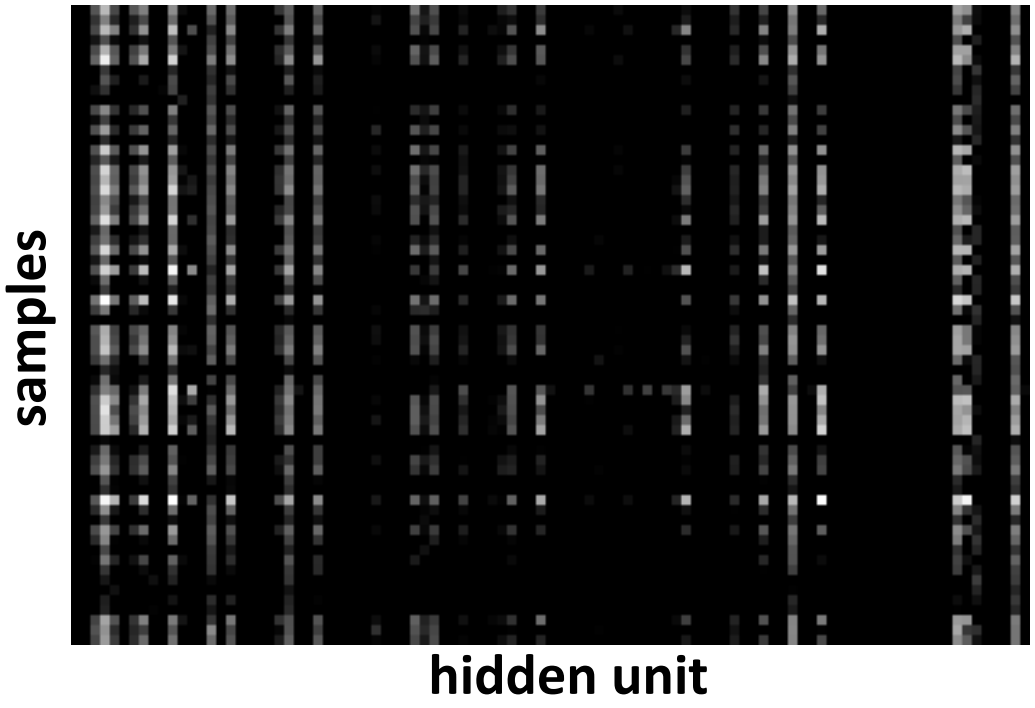
Deep neural networks learn hierarchical feature representations





What Are Good Training: Features

- Visualize features (feature maps need to be uncorrelated) and have high variance.



Bad training: many hidden units ignore the input and/or exhibit strong correlations.

Source: Ranzato

What Are Good Features

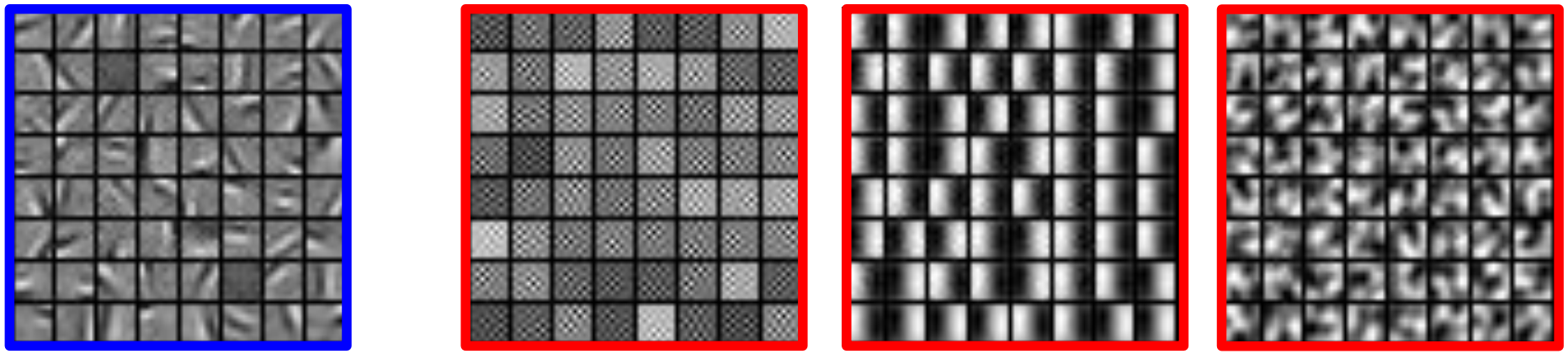
- Visualize features (feature maps need to be uncorrelated) and have high variance.

Good training: hidden units are sparse across samples and across features.

Source: Ranzato

What Are Good Training: Features

- Visualize parameters



Good training: learned filters exhibit structure and are uncorrelated.

Source: Ranzato

TAKE HOME MESSAGES

The whole CNN

Property 1

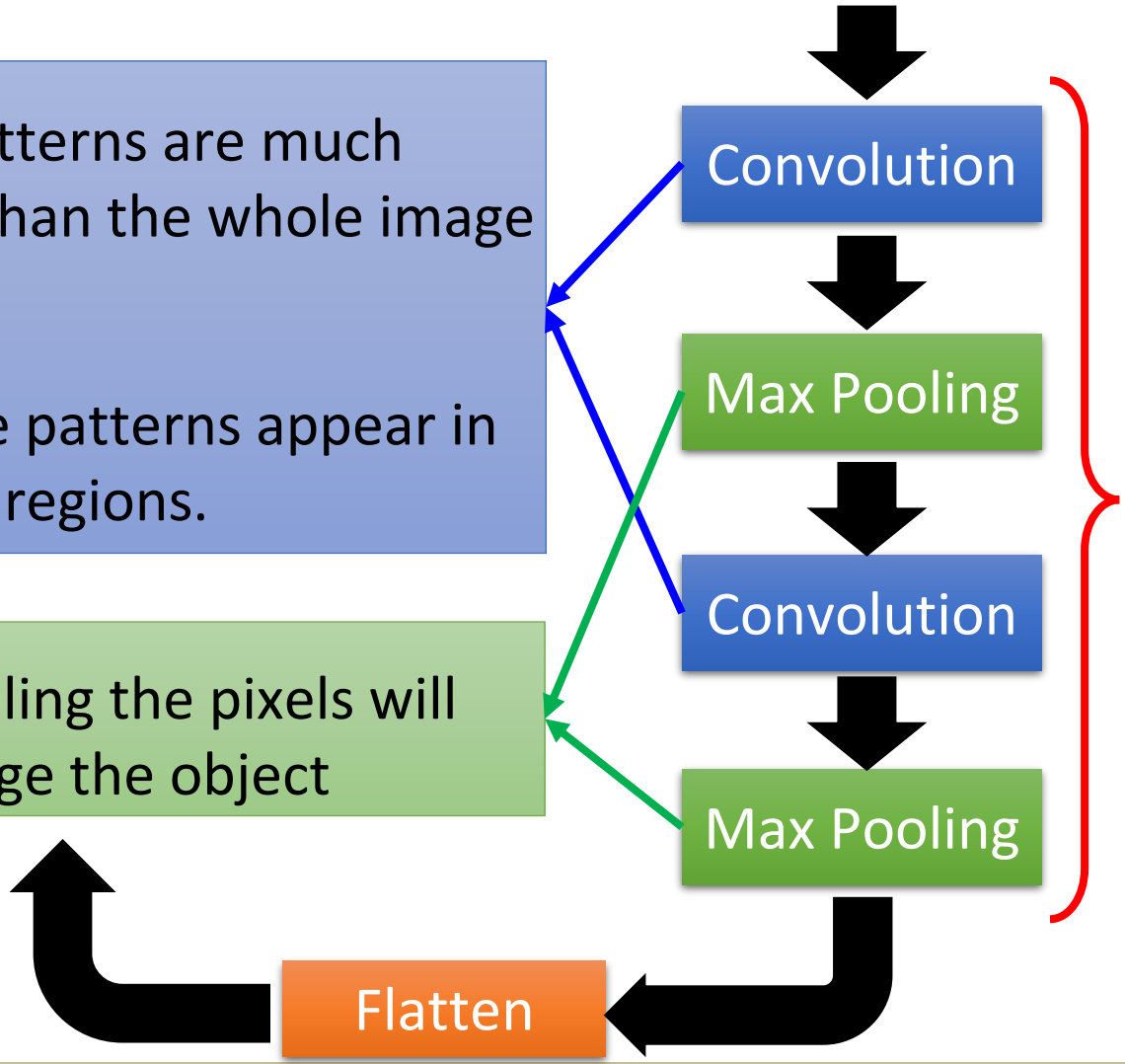
- Some patterns are much smaller than the whole image

Property 2

- The same patterns appear in different regions.

Property 3

- Subsampling the pixels will not change the object



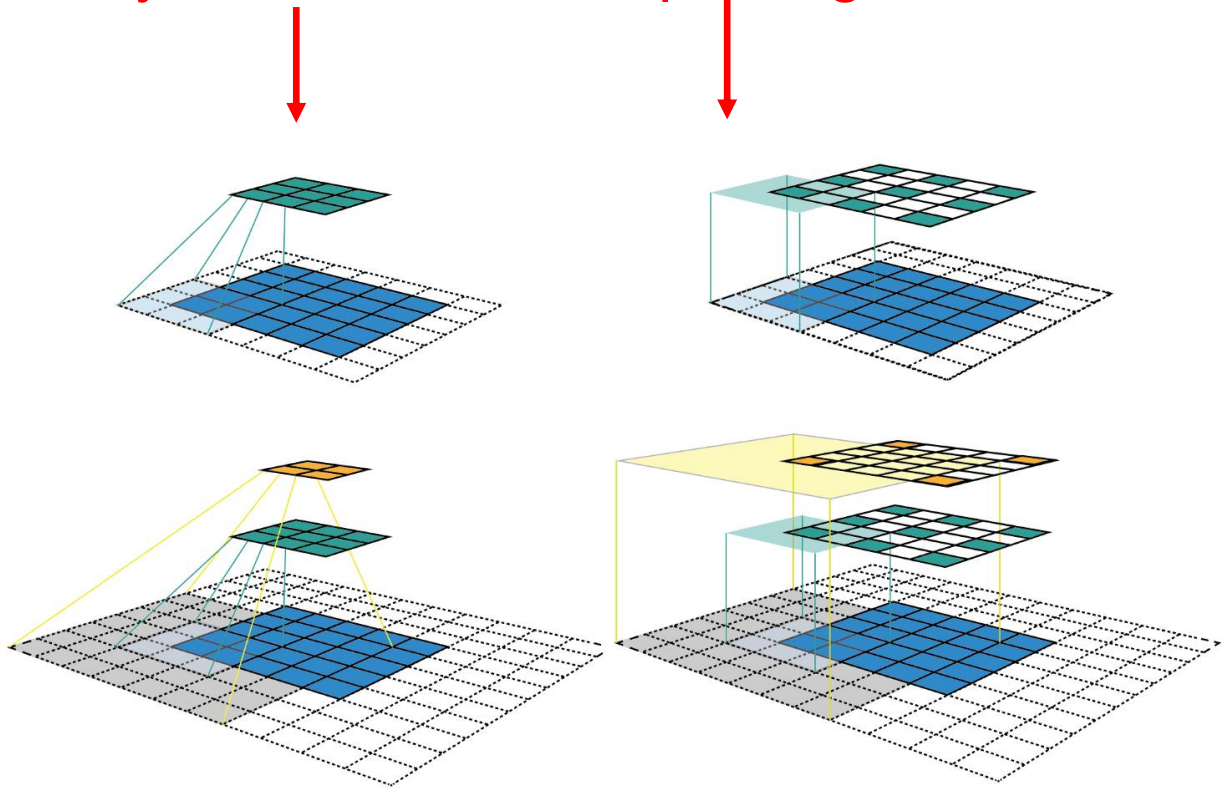
Basic layer structure

- Convolutional layer
- Pooling layer
- Activations
- Fully connected layer

Source: Convolutional Neural Network, Hung-yi Lee

Receptive Field of Convolutional Layer

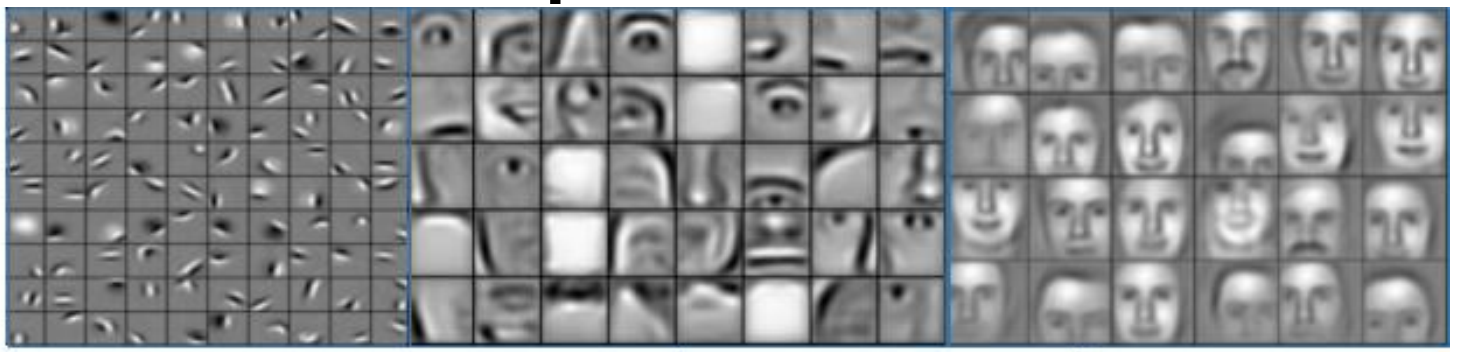
- receptive field
  - The region in the input space that a particular CNN's feature is looking at
  - How to increase: multi-layer convolution or pooling or dilated convolution

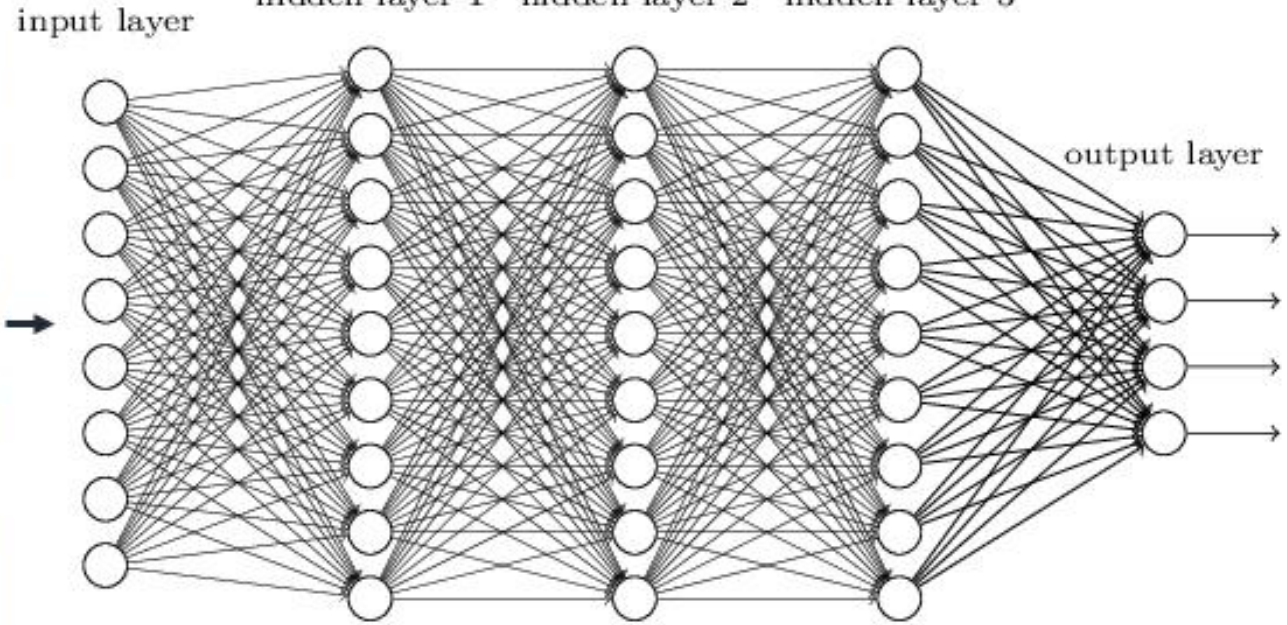


DL到底學了什麼?越深越好嗎?

Hierarchical Feature Representations

Deep neural networks learn hierarchical feature representations





Tools

- https://www.deepspeed.ai/

DeepSpeed

Getting Started Blog Tutorials Documentation GitHub

Home / Tutorials / **Flops Profiler**

TRAINING
INFERENCE
COMPRESSION
GETTING STARTED
DS_CONFIG

Autotuning
Batch size
Optimizer
FP16
BFLOAT16
ZeRO optimizations
Logging

Flops Profiler

In this tutorial, we introduce the DeepSpeed Flops Profiler and provide examples of its usage.

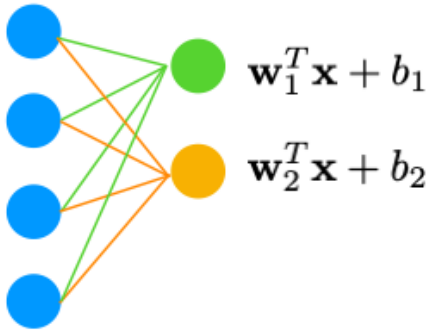
- Overview
- Flops Measurement
- Multi-GPU, Multi-node, Data Parallelism, and Model Parallelism
- Usage

Contents

Overview
Flops Measurement
Multi-GPU, Multi-node, Data Parallelism, and Model Parallelism
Usage
Usage With the DeepSpeed Runtime
Example: Megatron-

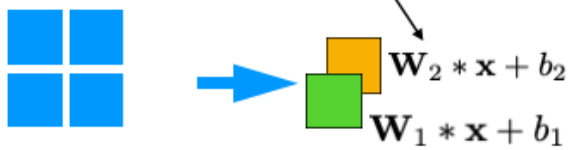
FC-Conv

- choosing a convolutional kernel that has the same size as the input feature map or 2) using 1x1 convolutions with multiple channels.



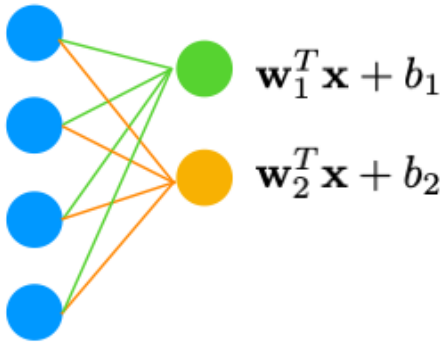
Fully connected layer

remember, these also involve dot products between the receptive fields and kernels

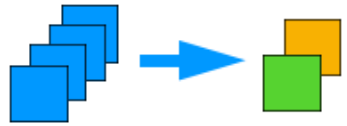


where $\mathbf{W}_1 = \begin{bmatrix} w_{1,1} & w_{1,2} \\ w_{1,3} & w_{1,4} \end{bmatrix}$

$\mathbf{W}_2 = \begin{bmatrix} w_{2,1} & w_{2,2} \\ w_{2,3} & w_{2,4} \end{bmatrix}$



Fully connected layer



Or, we can concatenate the inputs into 1x1 images with 4 channels and then use 2 kernels (remember, each kernel then also has 4 channels)