Deep Learning

Winter 2026

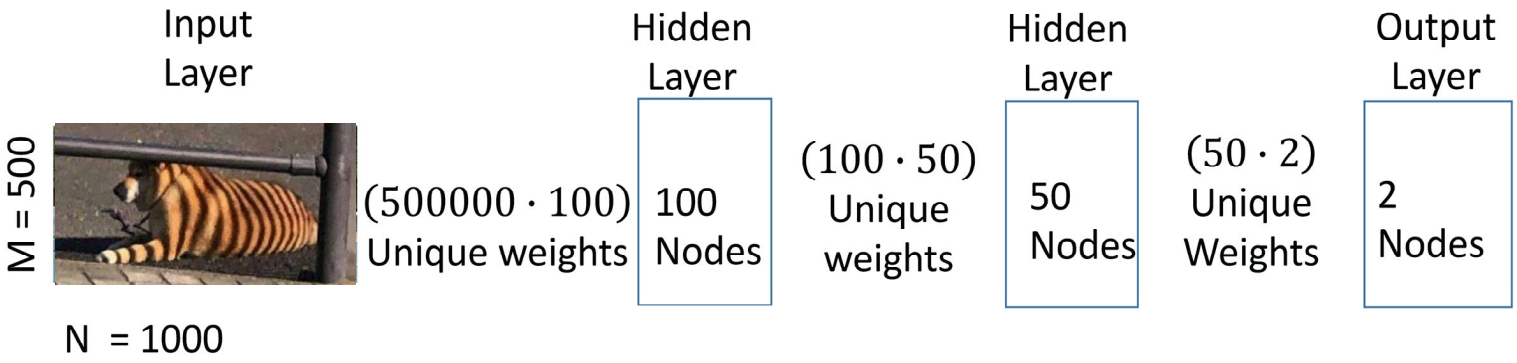
Convolutional Neural Networks

How Big Is This Image?

500

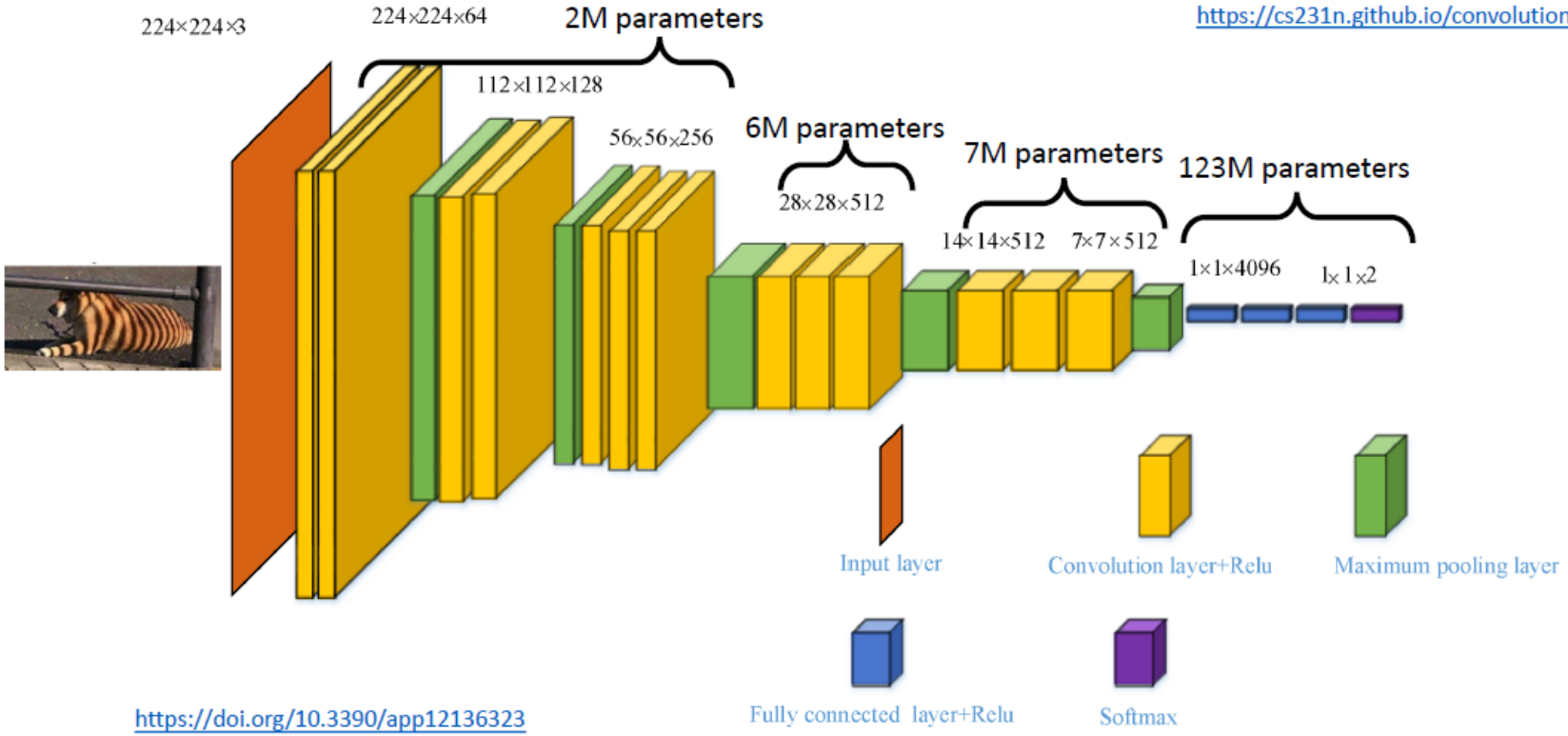
1000

How Many Weights in a Fully Connected Network?



$$50{,}000{,}000 + 5{,}000 + 100 = 50{,}005{,}100 \text{ weights}$$

VGG16 Network: 138MM Parameters



https://cs231n.github.io/convolutional-networks

https://doi.org/10.3390/app12136323

Biological Motivation

How does the eye do this?

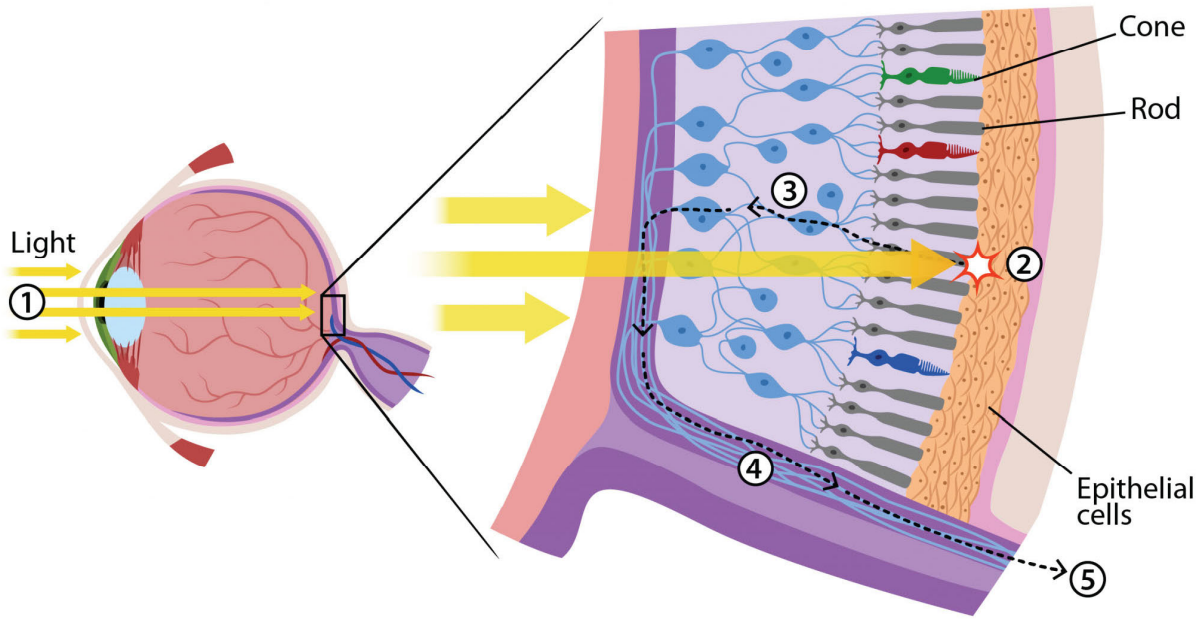
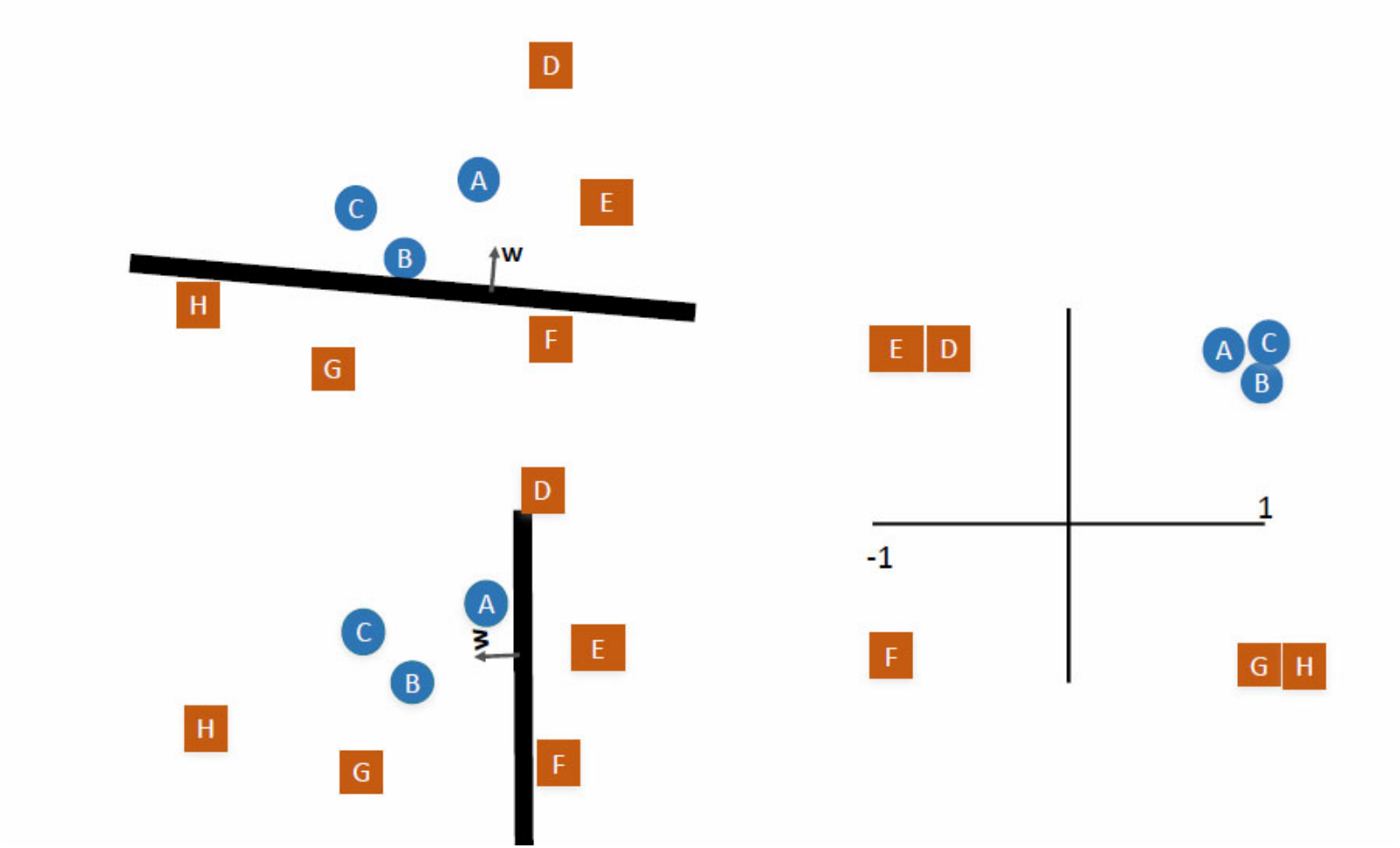


Image from https://askabiologist.asu.edu/rods-and-cones

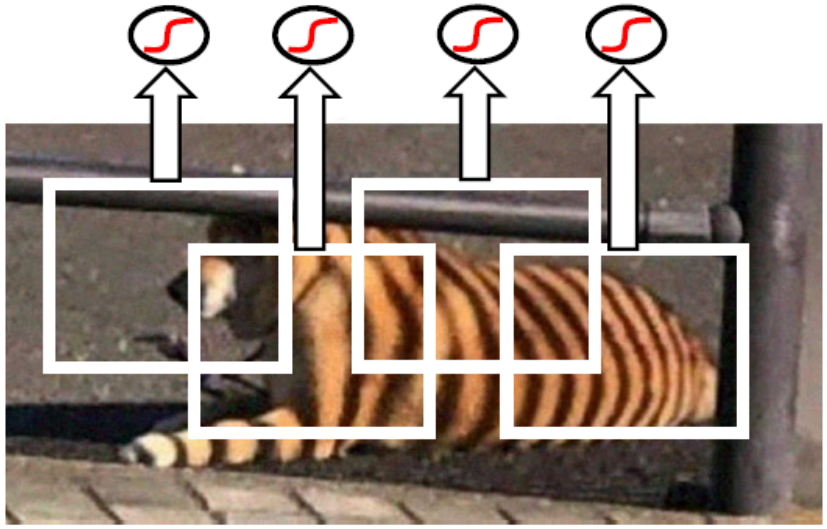
MLPs as Layered Feature Maps



Receptive Field

Small Fixed Windows (filter size/receptive field)

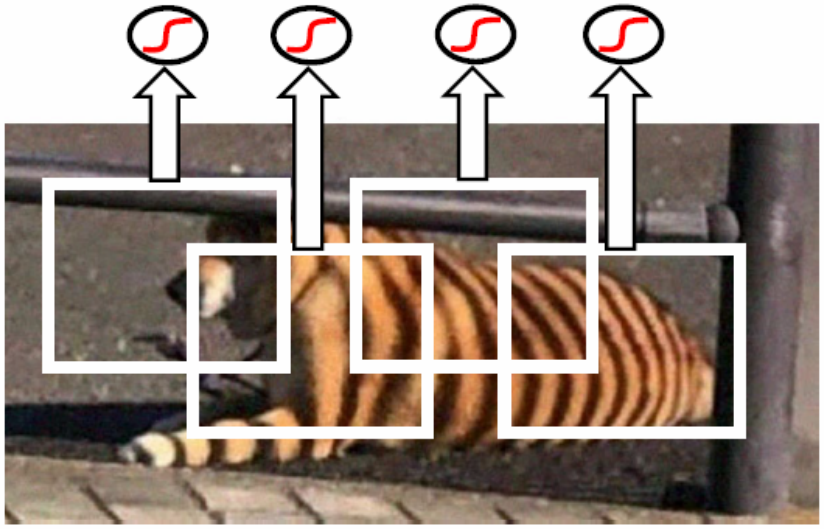
- If important features fall within a bounded size region, we can bound the receptive field of each unit to that size.
- This greatly reduces the number of weights.



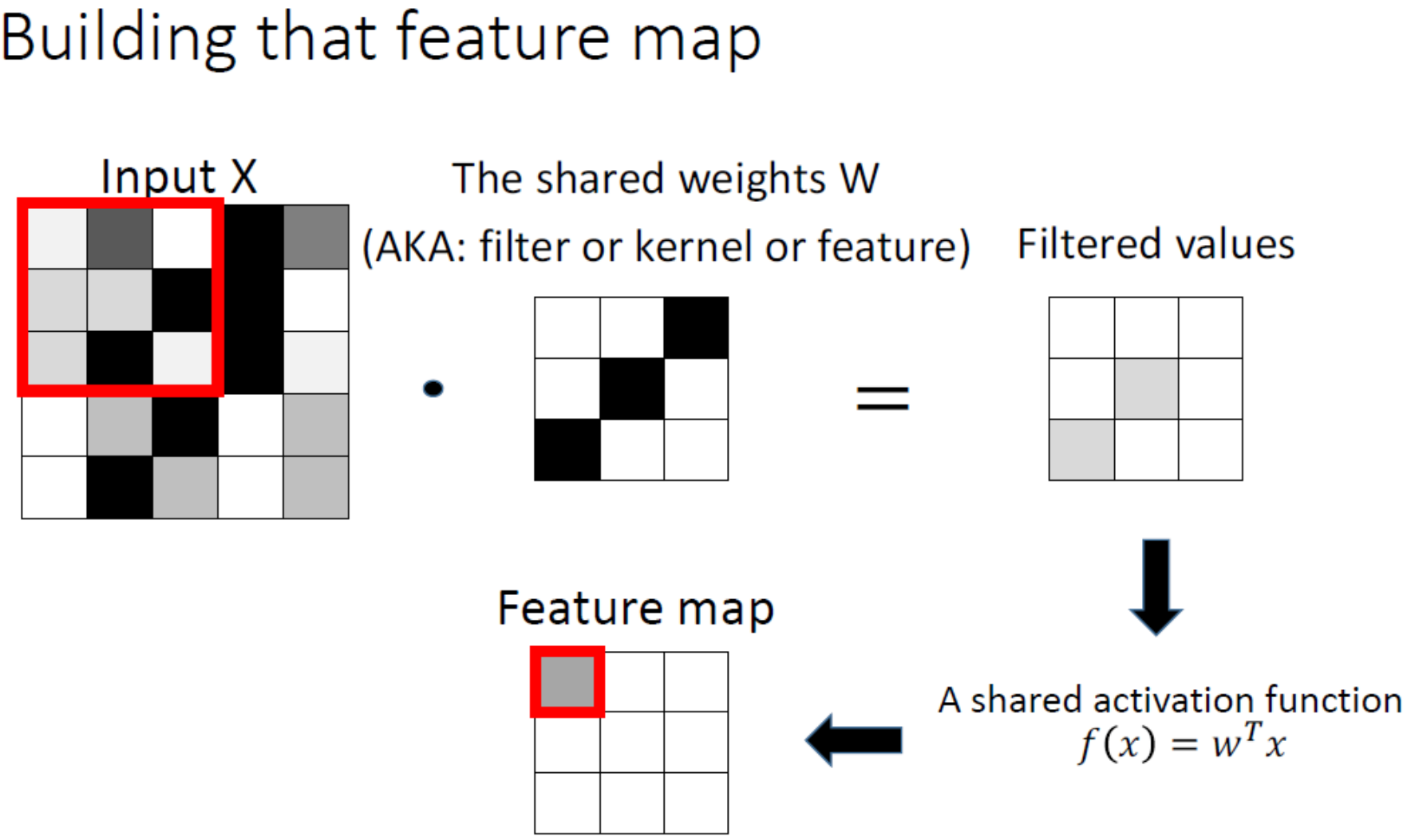
Receptive Field

Shared weights

- If a feature is good to find in one region, it may be good to find in other regions.
- Units look for the same feature if they share weights.
- A set of units that share weights is a feature map (aka ”channel”)

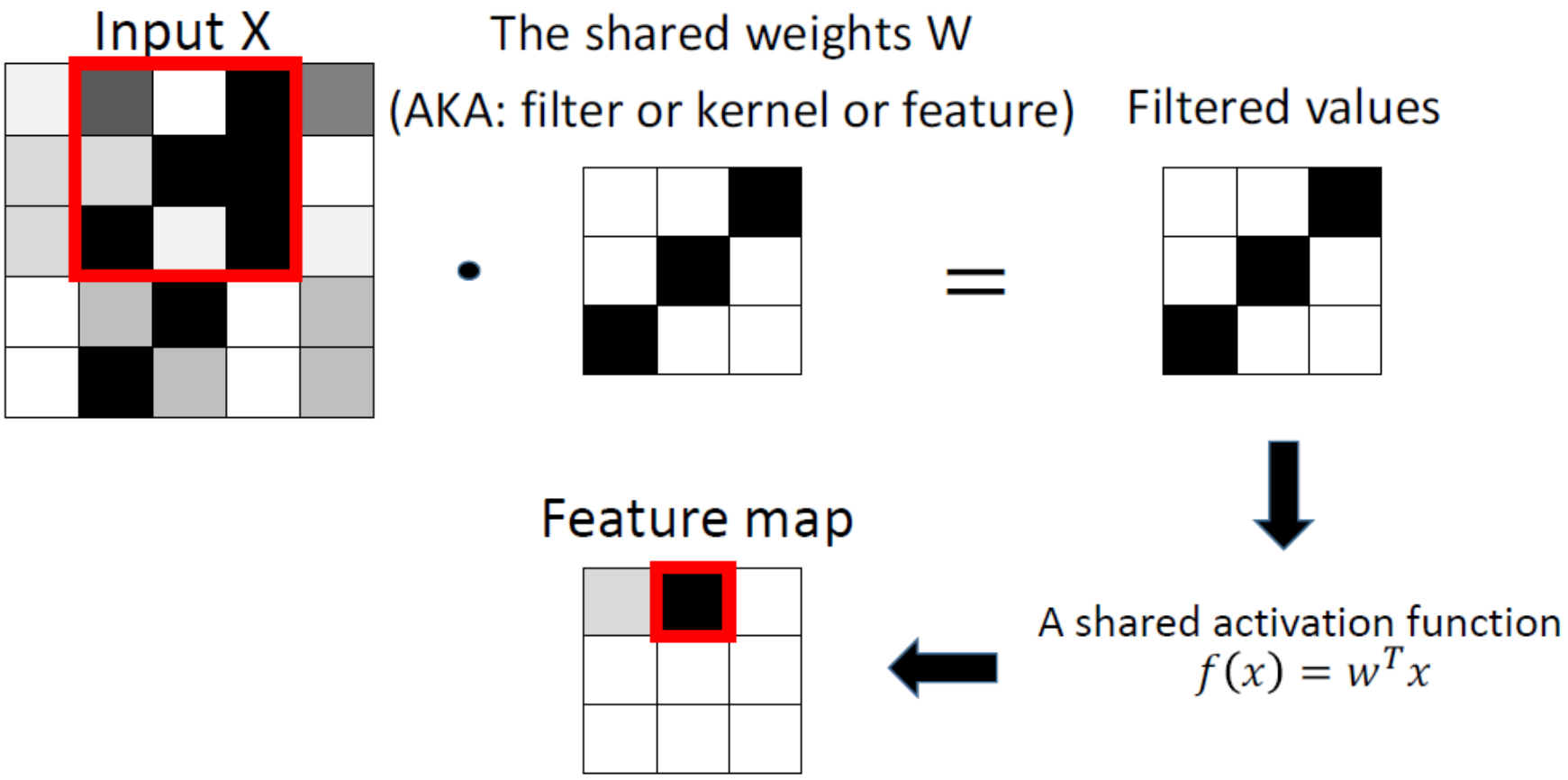


Convolution: Feature Map



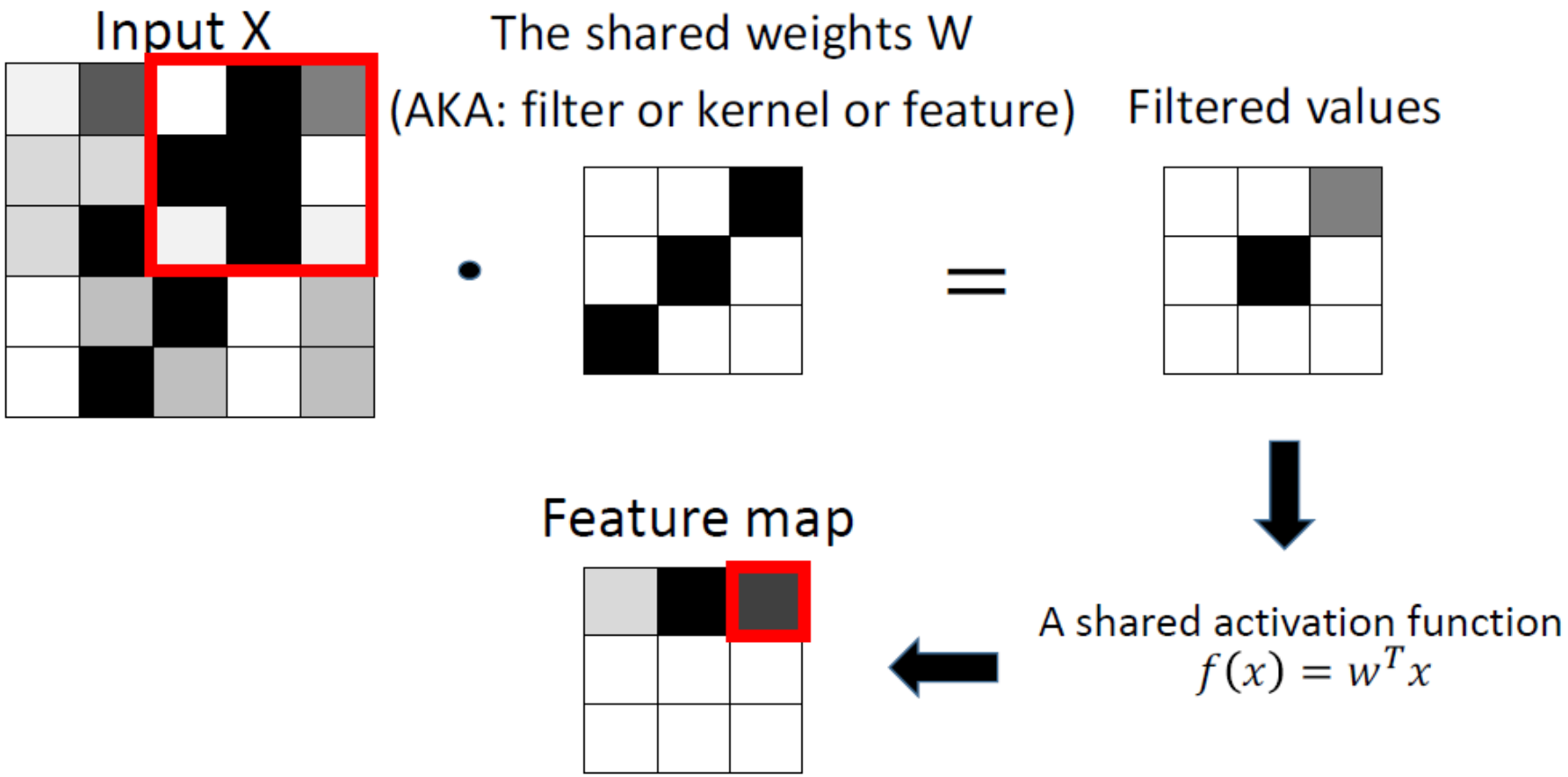
Convolution: Feature Map

Building that feature map

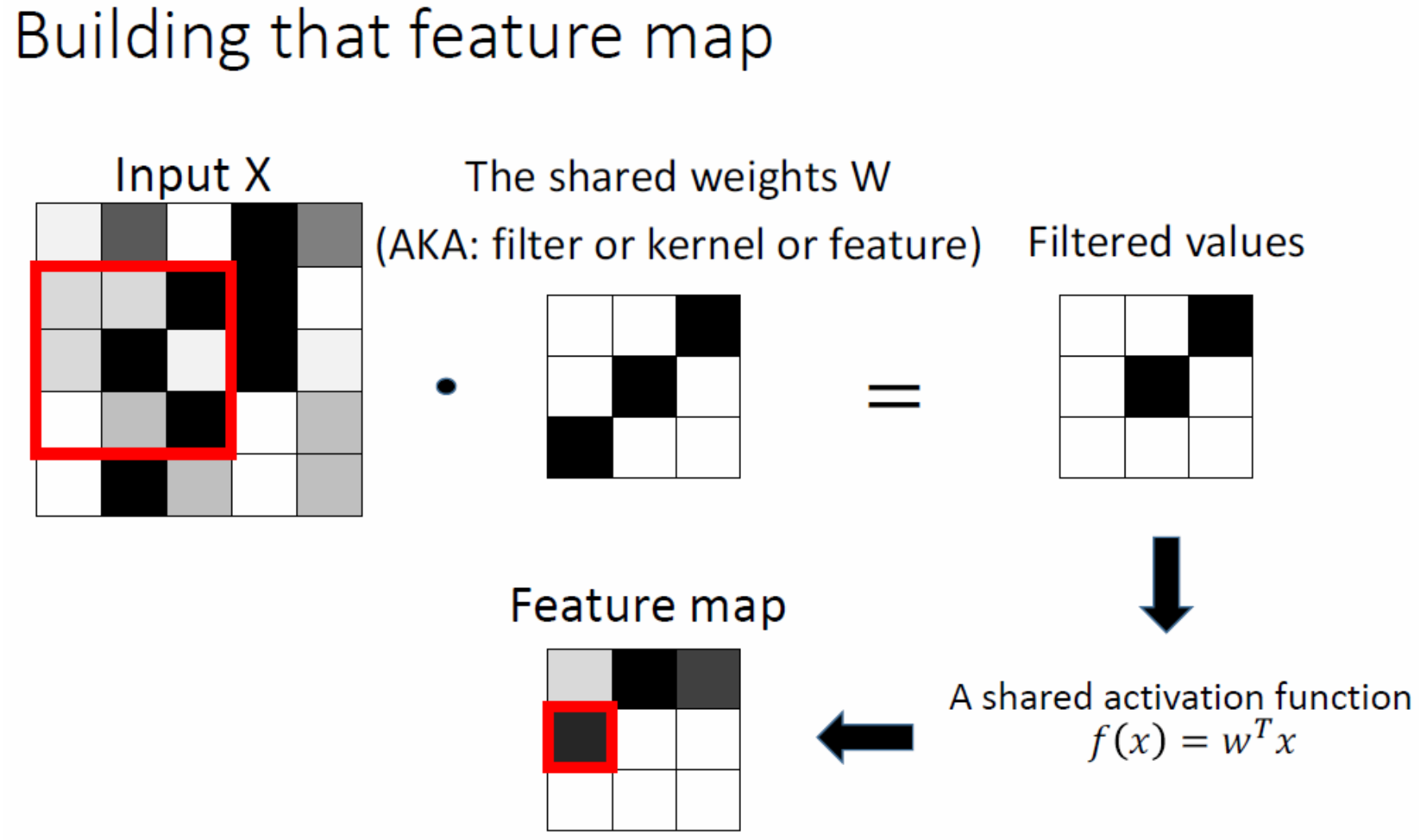


Convolution: Feature Map

Building that feature map



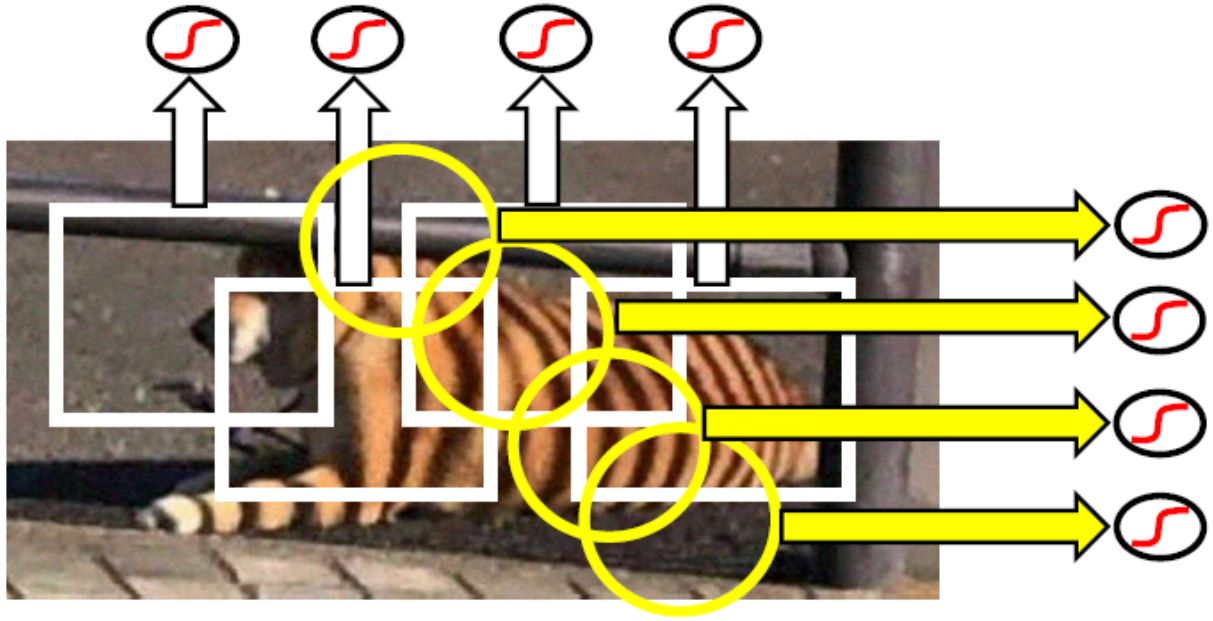
Convolution: Feature Map



Convolution: Feature Map

Multiple Feature Maps

- To look for multiple features, use multiple feature maps.
- Each map will specialize on one thing.
- Even with many feature maps, you still have far fewer weights



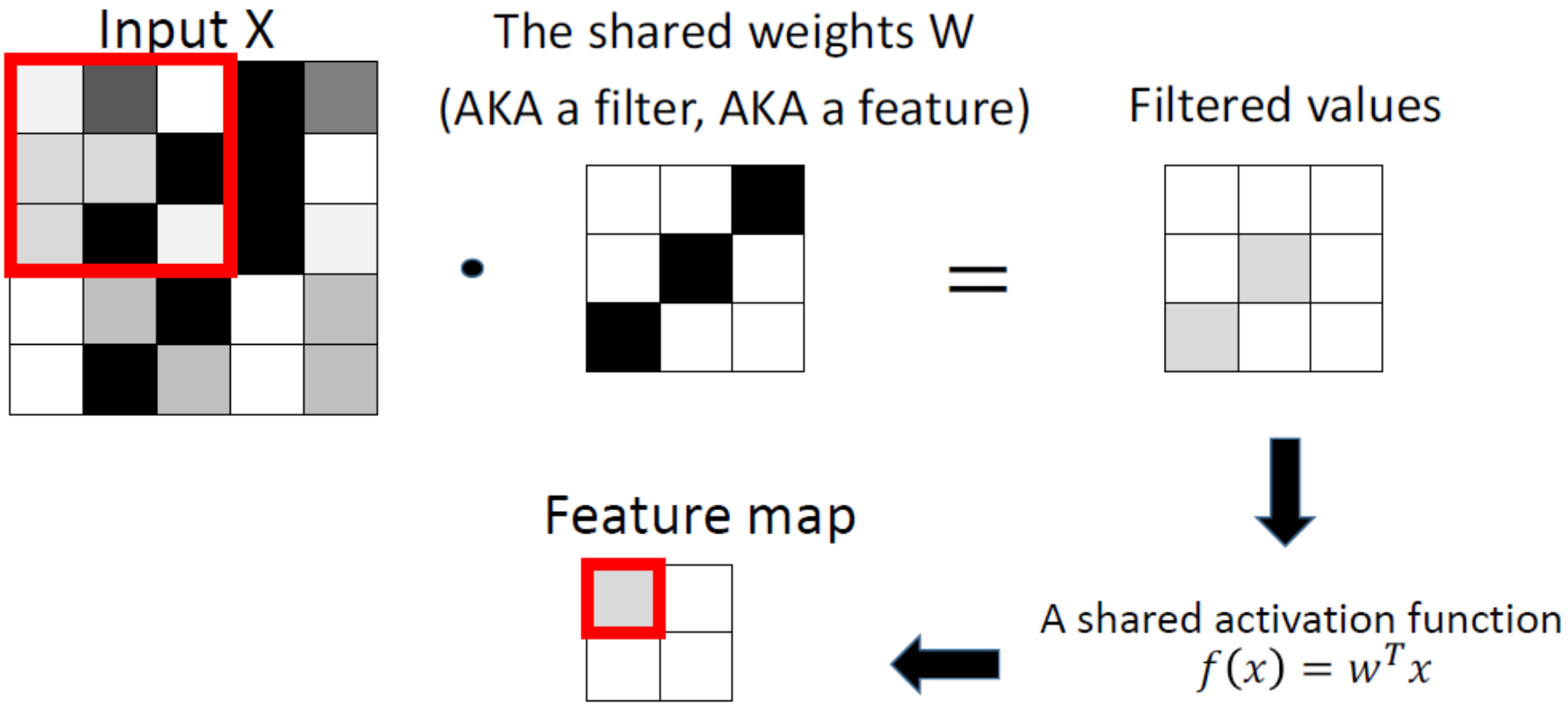
Convolution: Stride

Stride

- How many units you move with each step of your filter/kernel

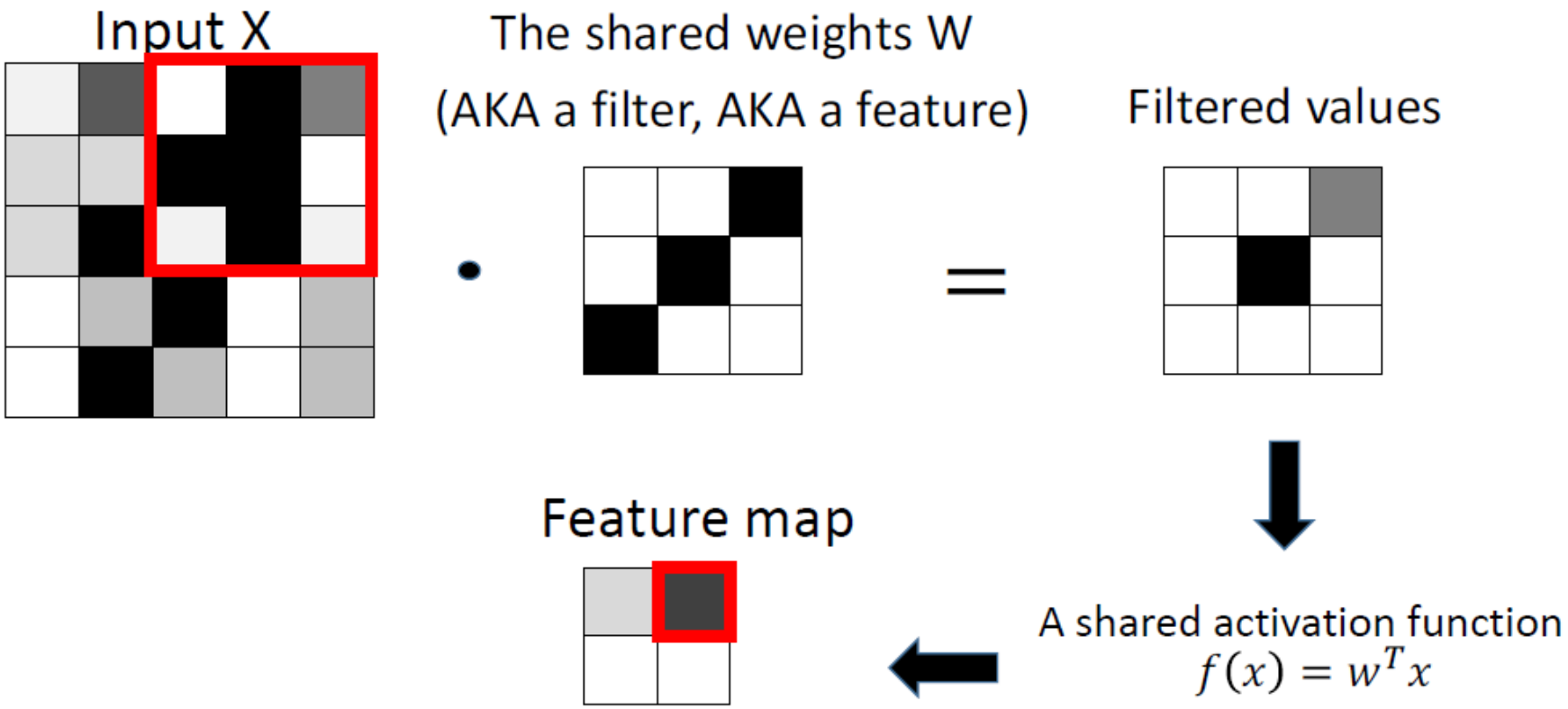
Convolution: Stride

Let's make that stride = 2



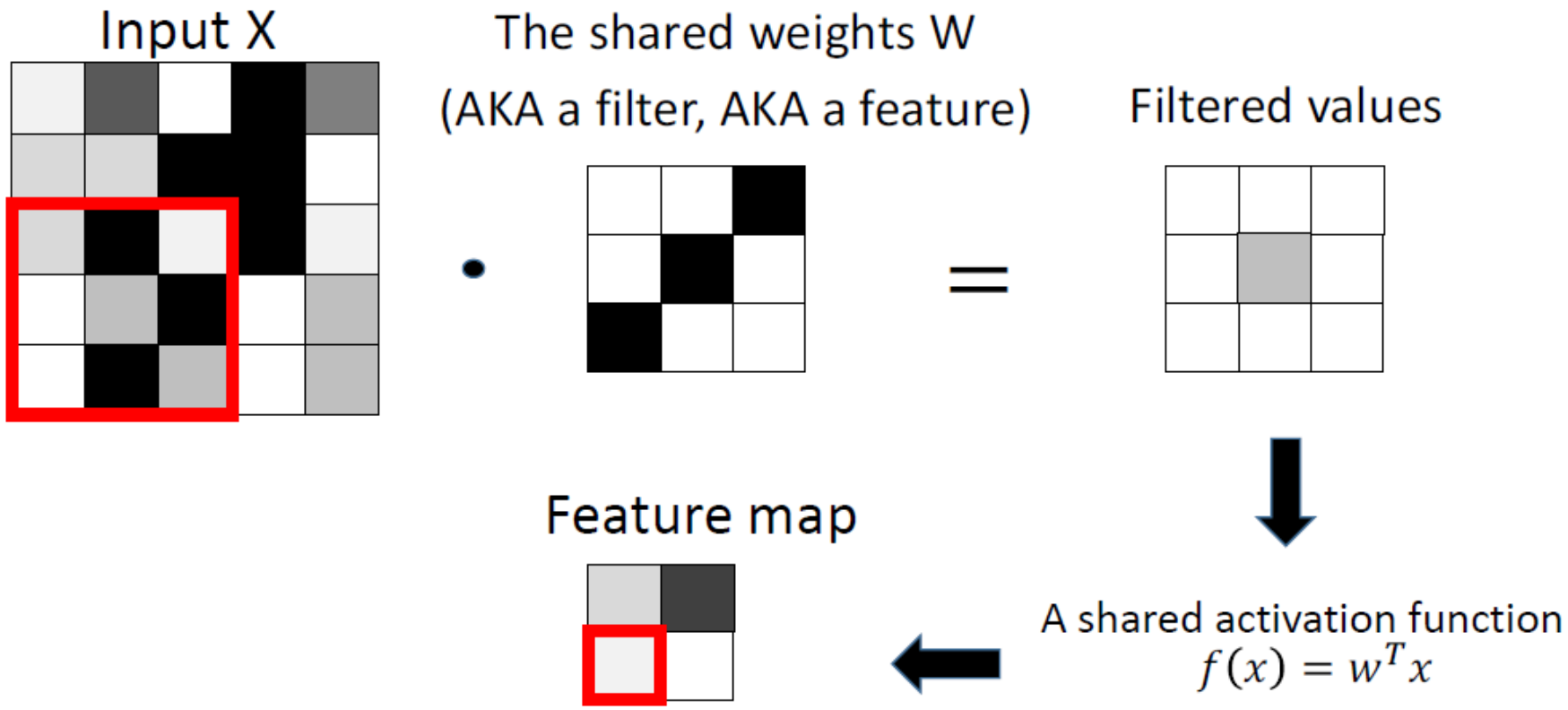
Convolution: Stride

Let's make that stride = 2



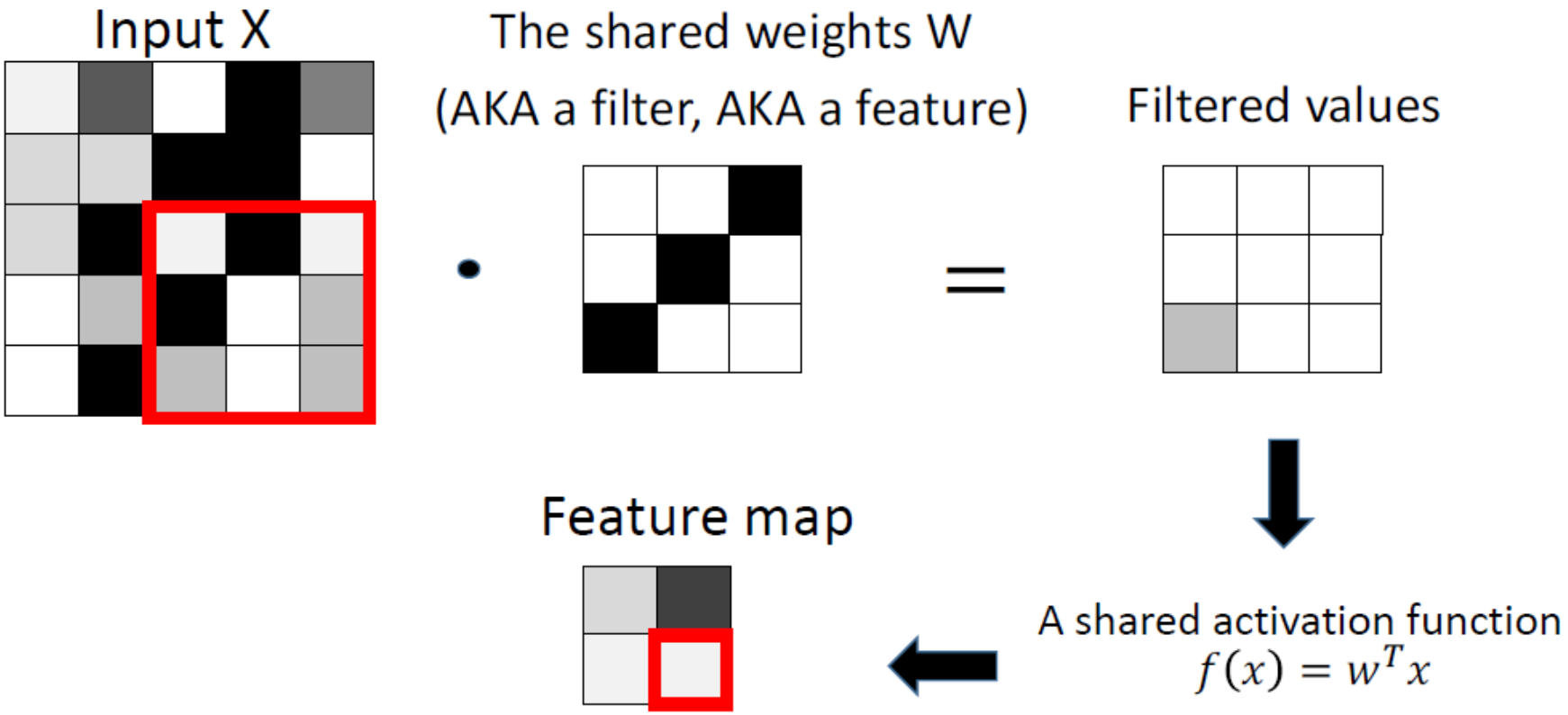
Convolution: Stride

Let's make that stride = 2

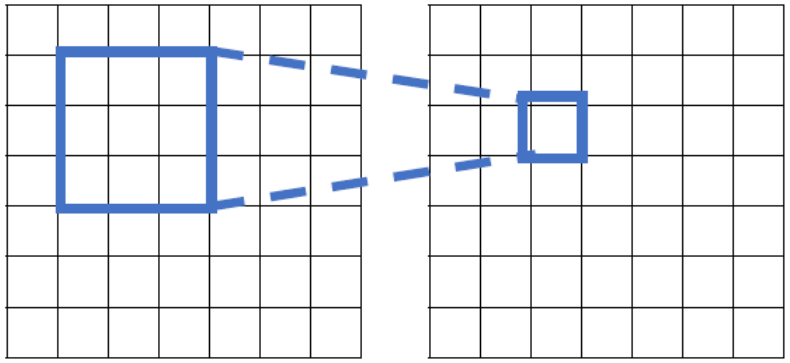


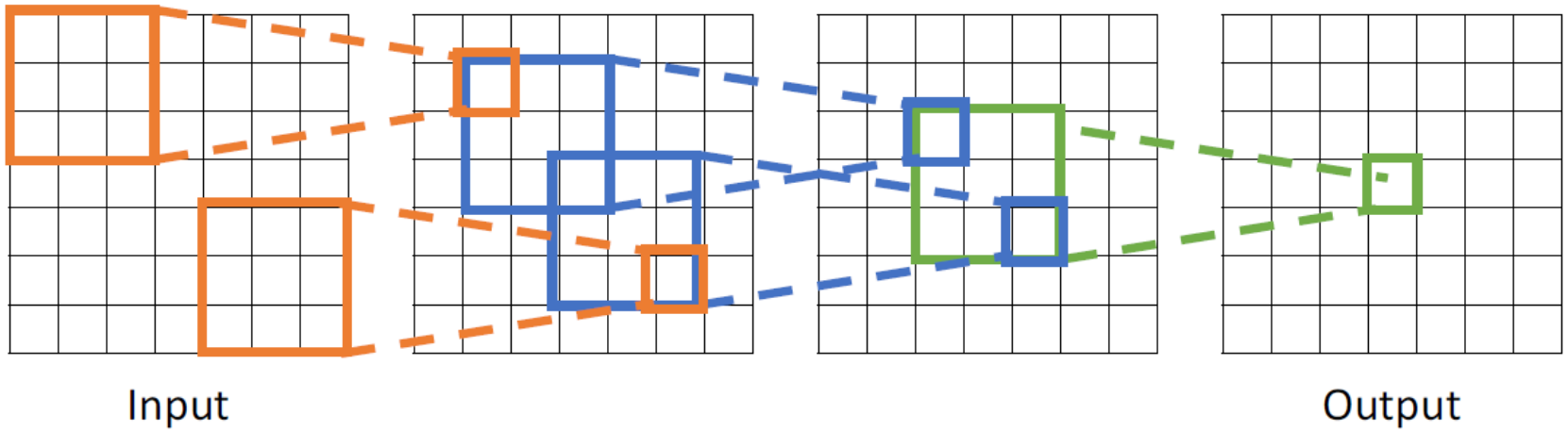
Convolution: Stride

Let's make that stride = 2



Convolution: Stride and Receptive Field



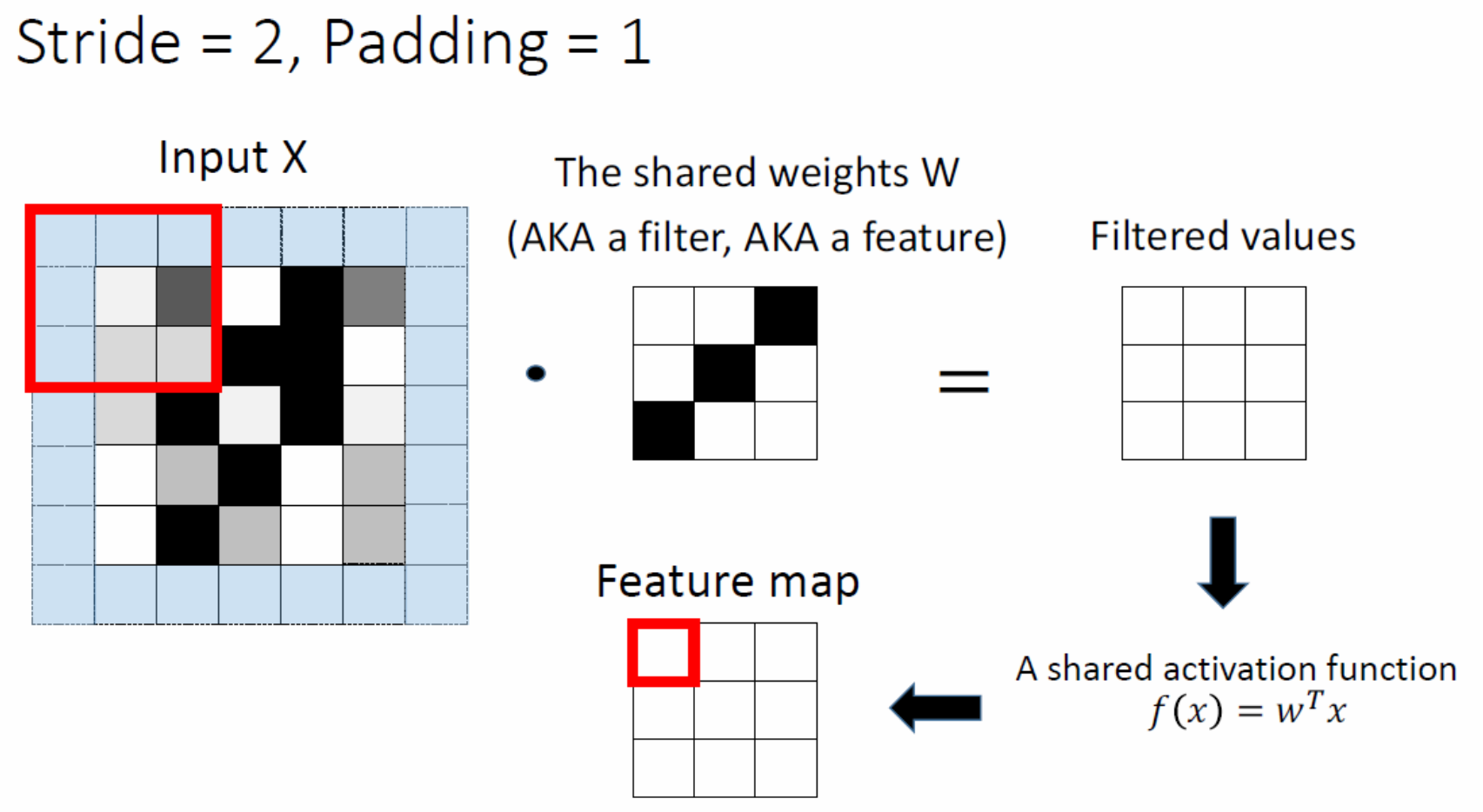


Convolution: Padding

Padding

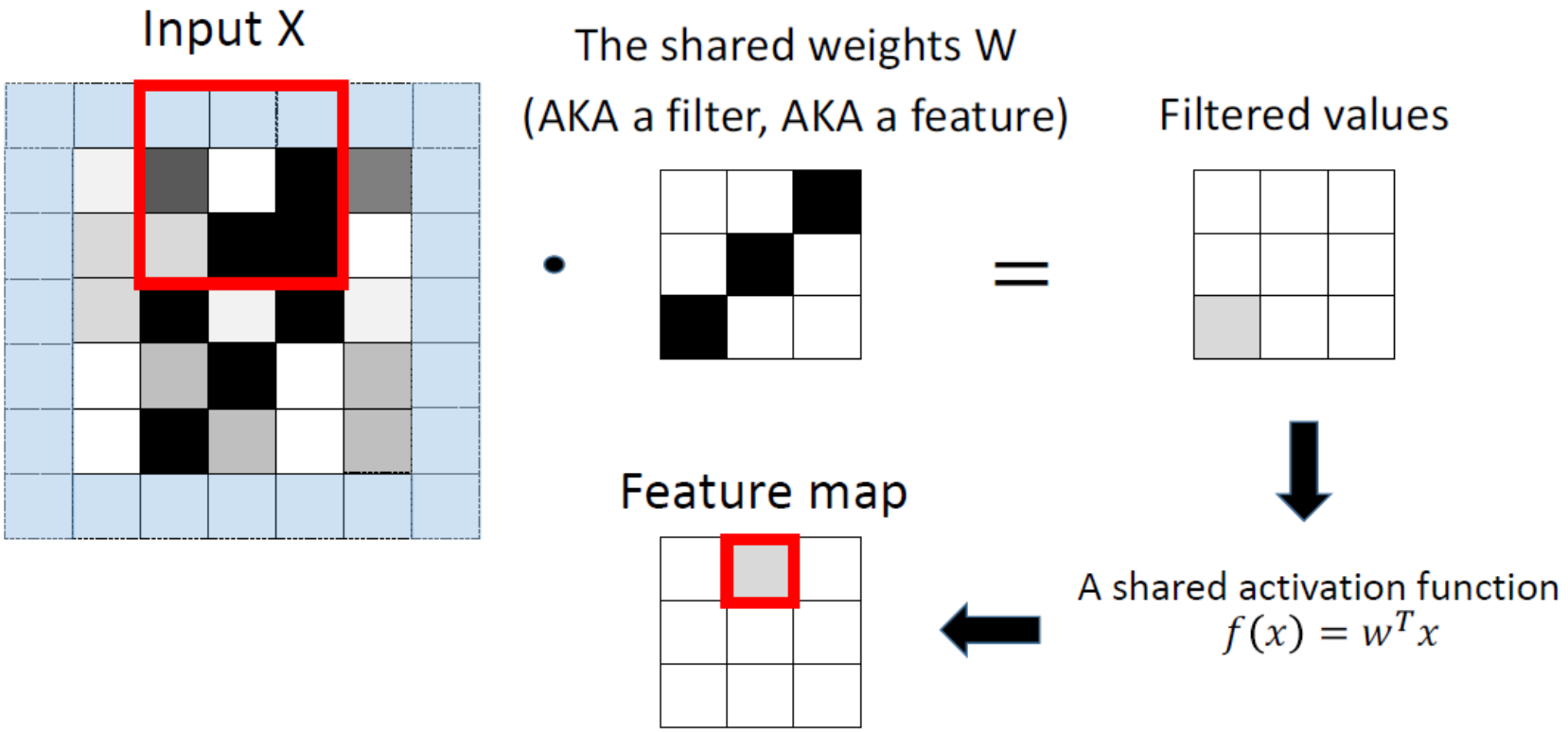
- Extra blank rows and columns added around your input.

Convolution: Padding

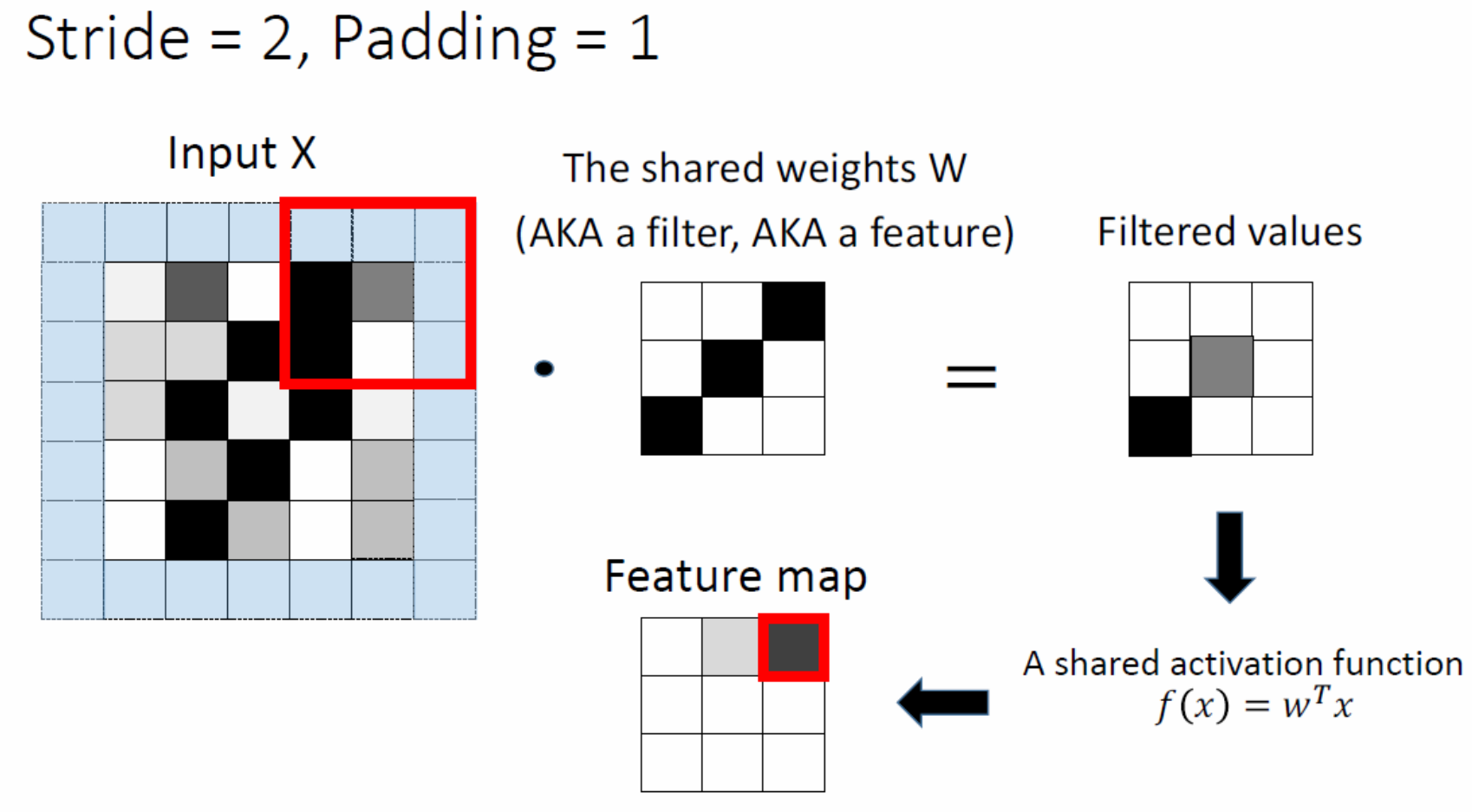


Convolution: Padding

Stride = 2, Padding = 1



Convolution: Padding



Convolution: Padding

Stride = 2, Padding = 1

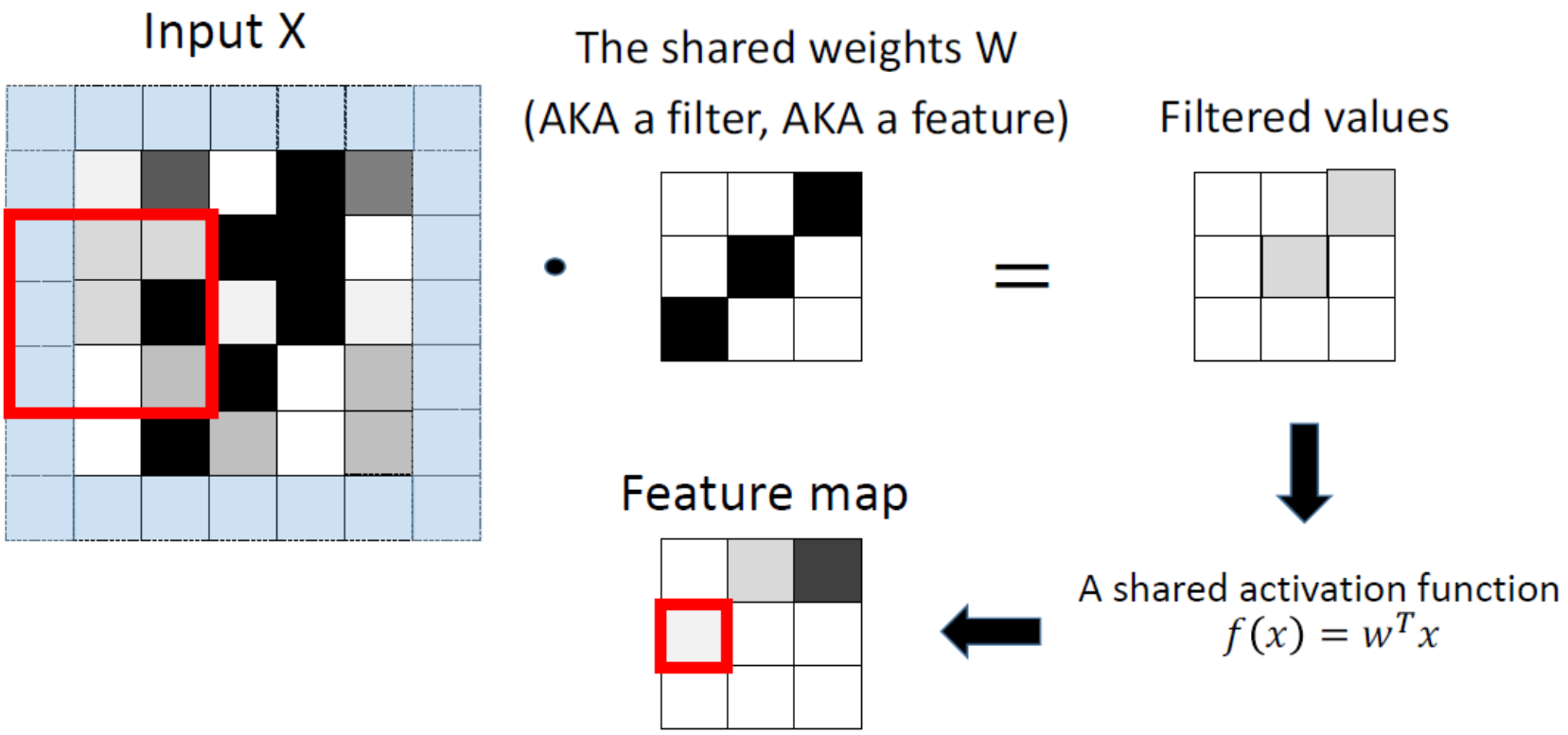
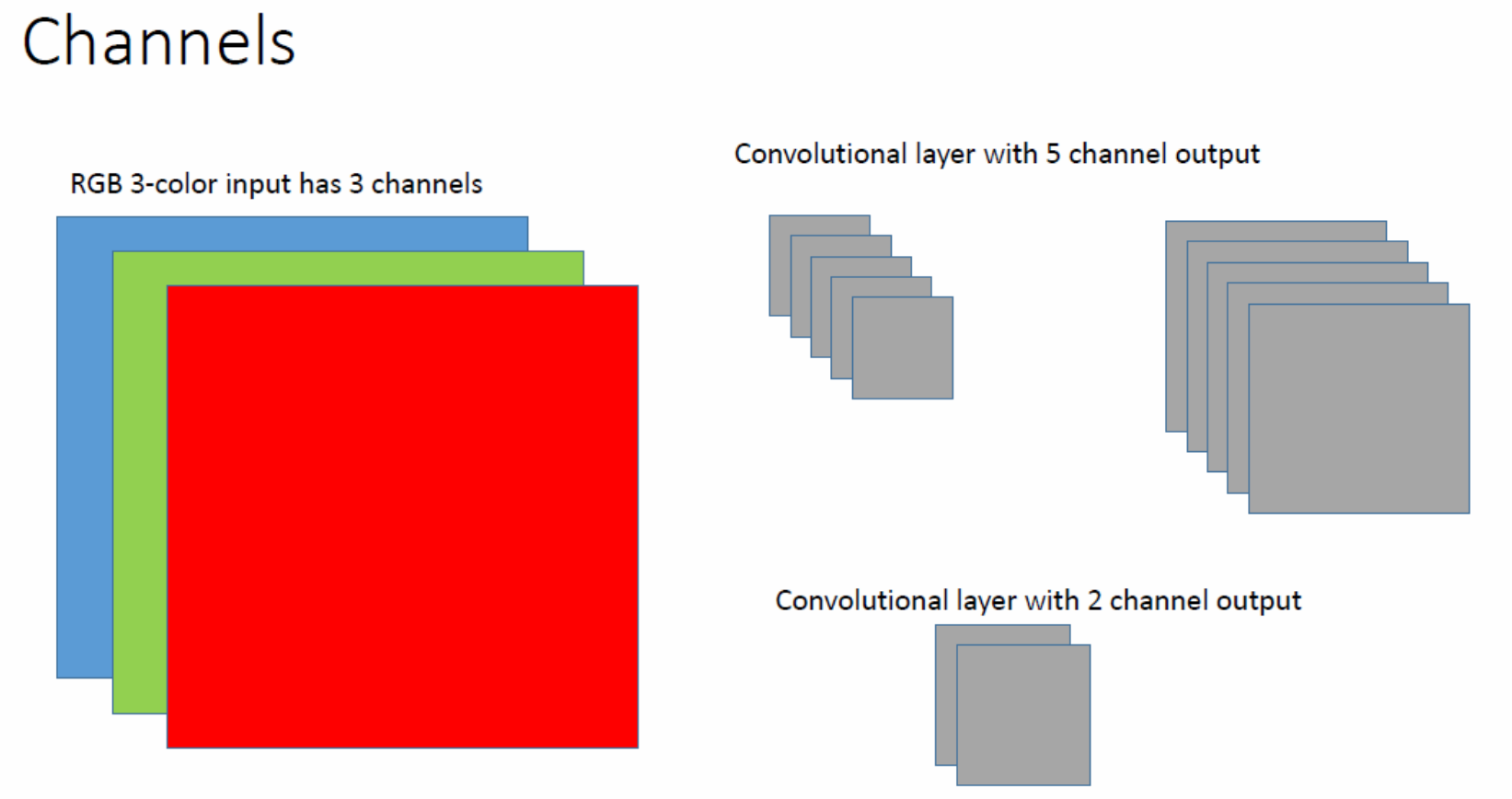
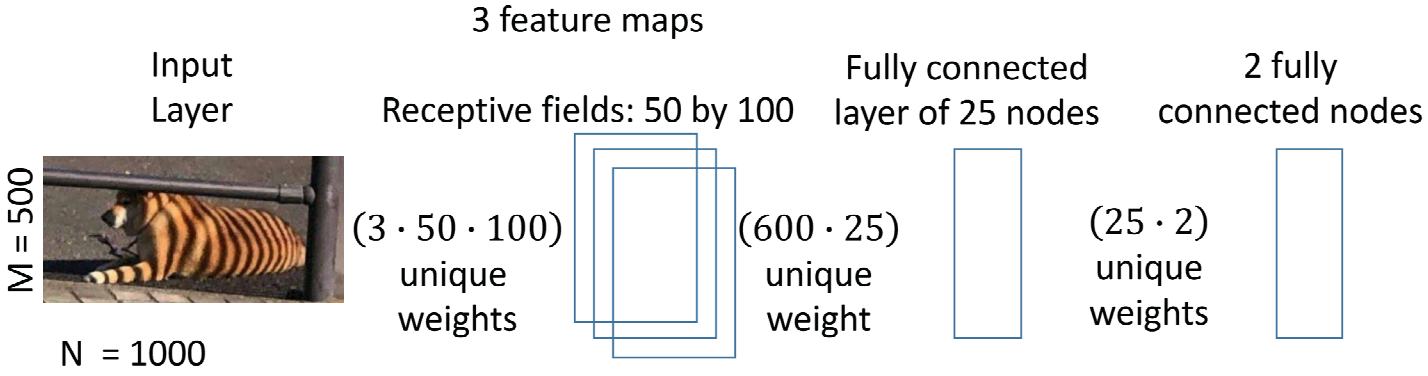


Image Channels



Convolution: Parameters

How many weights in a convolutional net?



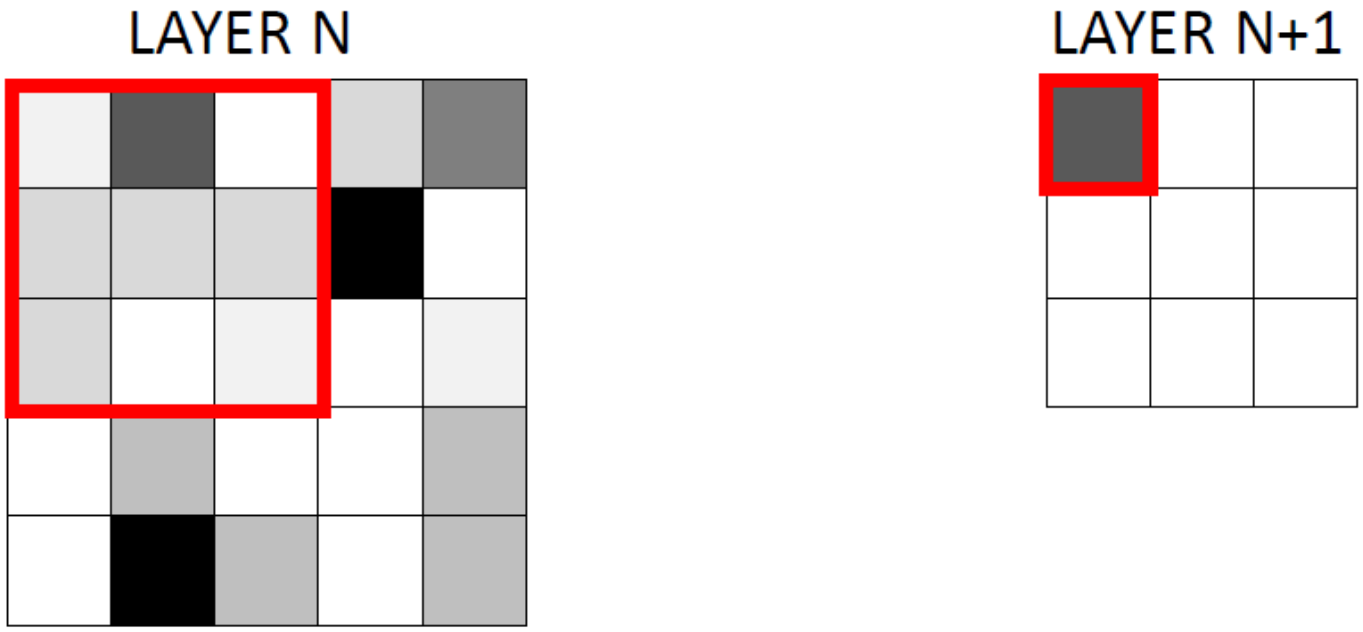
$15{,}000 + 15{,}000 + 50 = 30{,}050$ unique weights

Compare that to the $50{,}005{,}100$ weights in the other network

Max Pooling

Max Pool Layer: A kind of downsampling

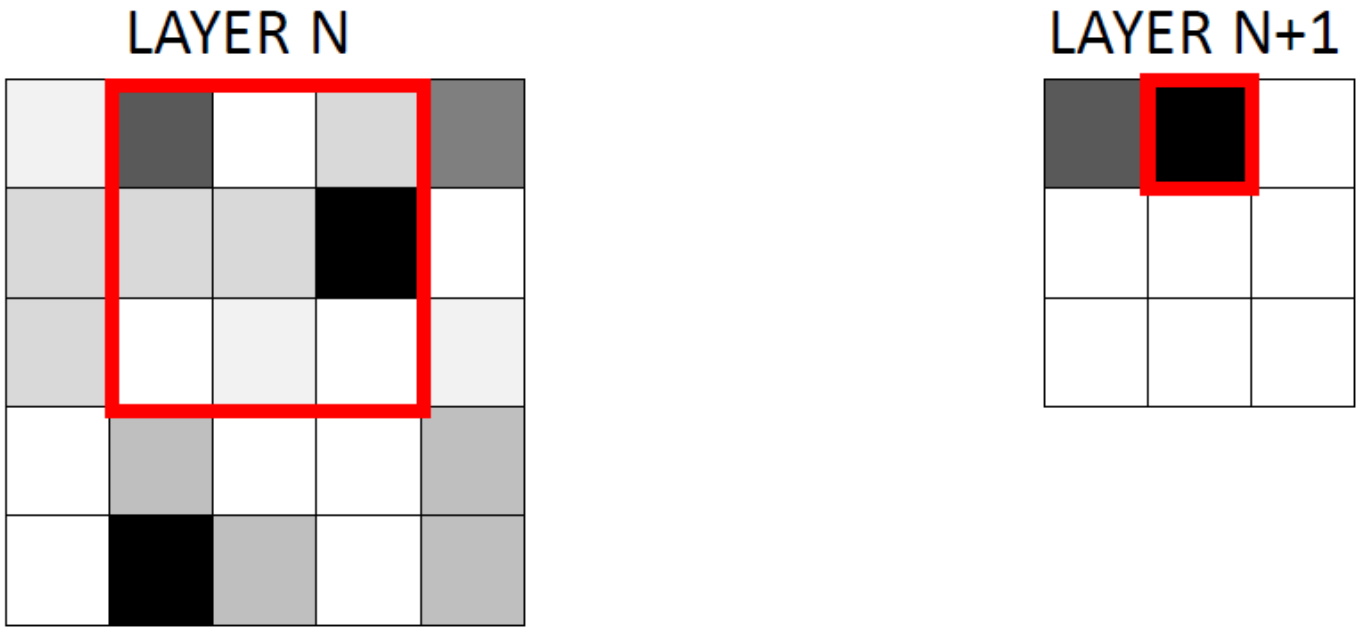
- Max Pool $f(x) = \max(x_1, x_2, \ldots x_n)$



Max Pooling

Max Pool Layer: A kind of downsampling

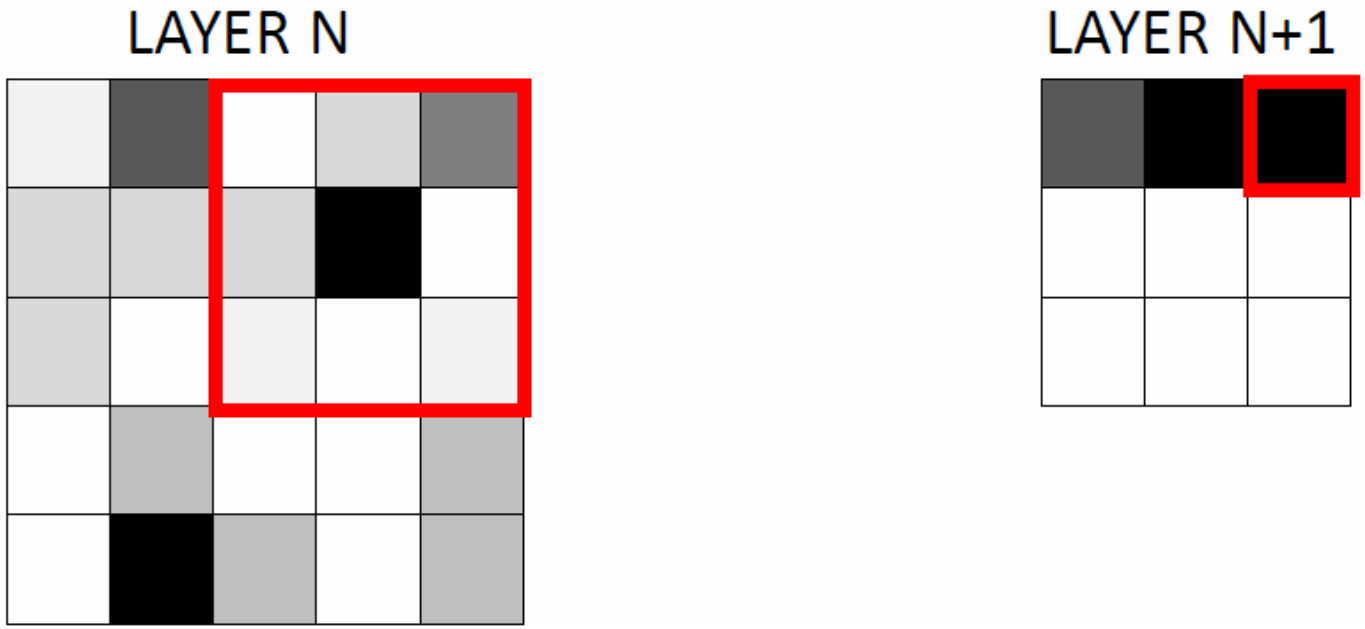
- Max Pool $f(x) = \max(x_1, x_2, \ldots x_n)$



Max Pooling

Max Pool Layer: A kind of downsampling

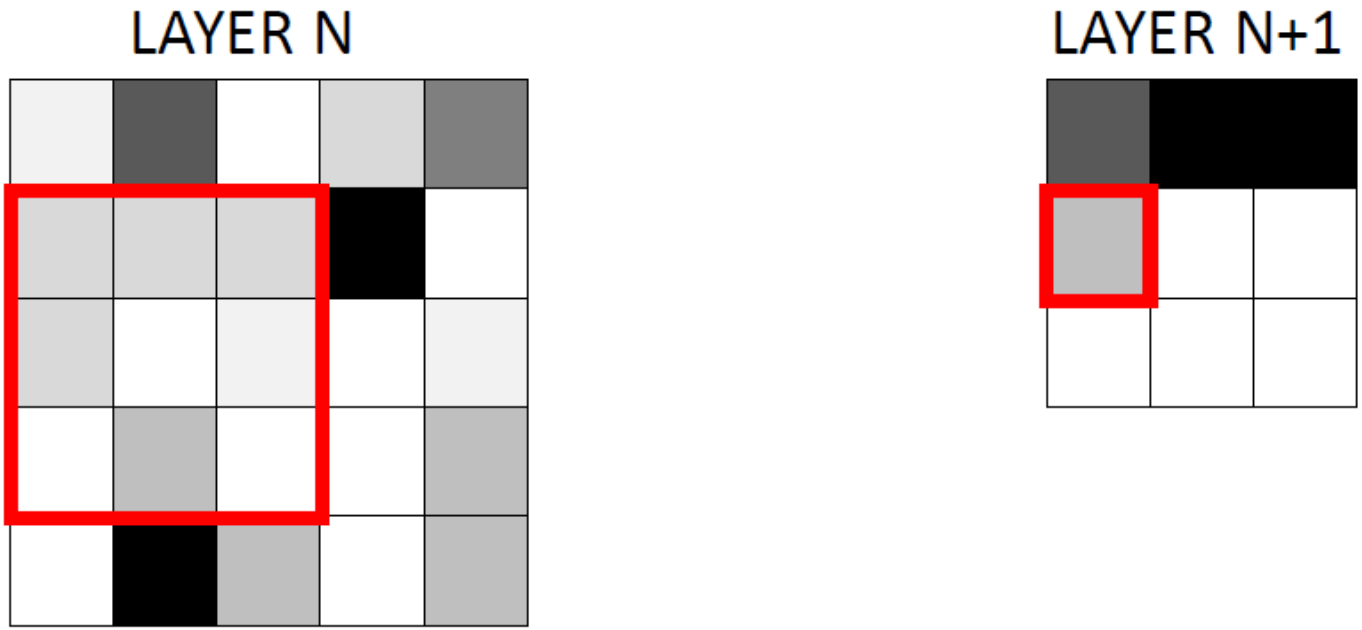
- Max Pool $f(x) = \max(x_1, x_2, \ldots x_n)$



Max Pooling

Max Pool Layer: A kind of downsampling

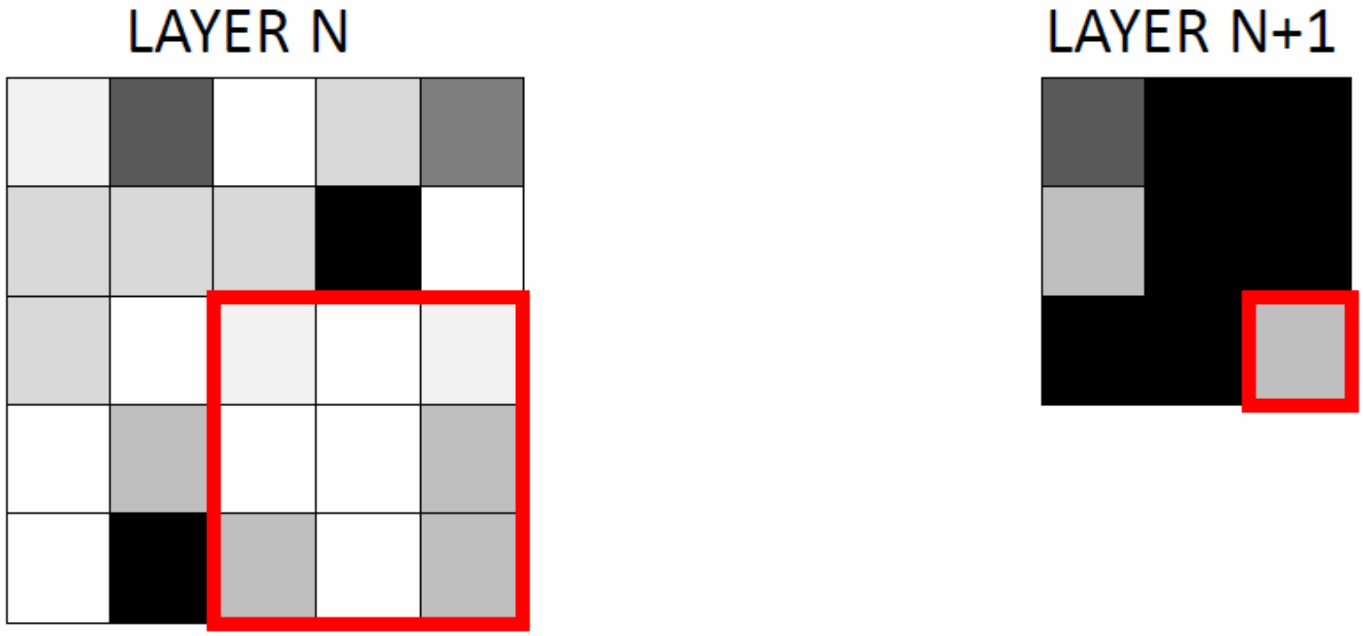
- Max Pool $f(x) = \max(x_1, x_2, \ldots x_n)$



Max Pooling

Max Pool Layer: A kind of downsampling

- Max Pool $f(x) = \max(x_1, x_2, \ldots x_n)$



Other Kinds of Pooling

Beyond Max Pooling:

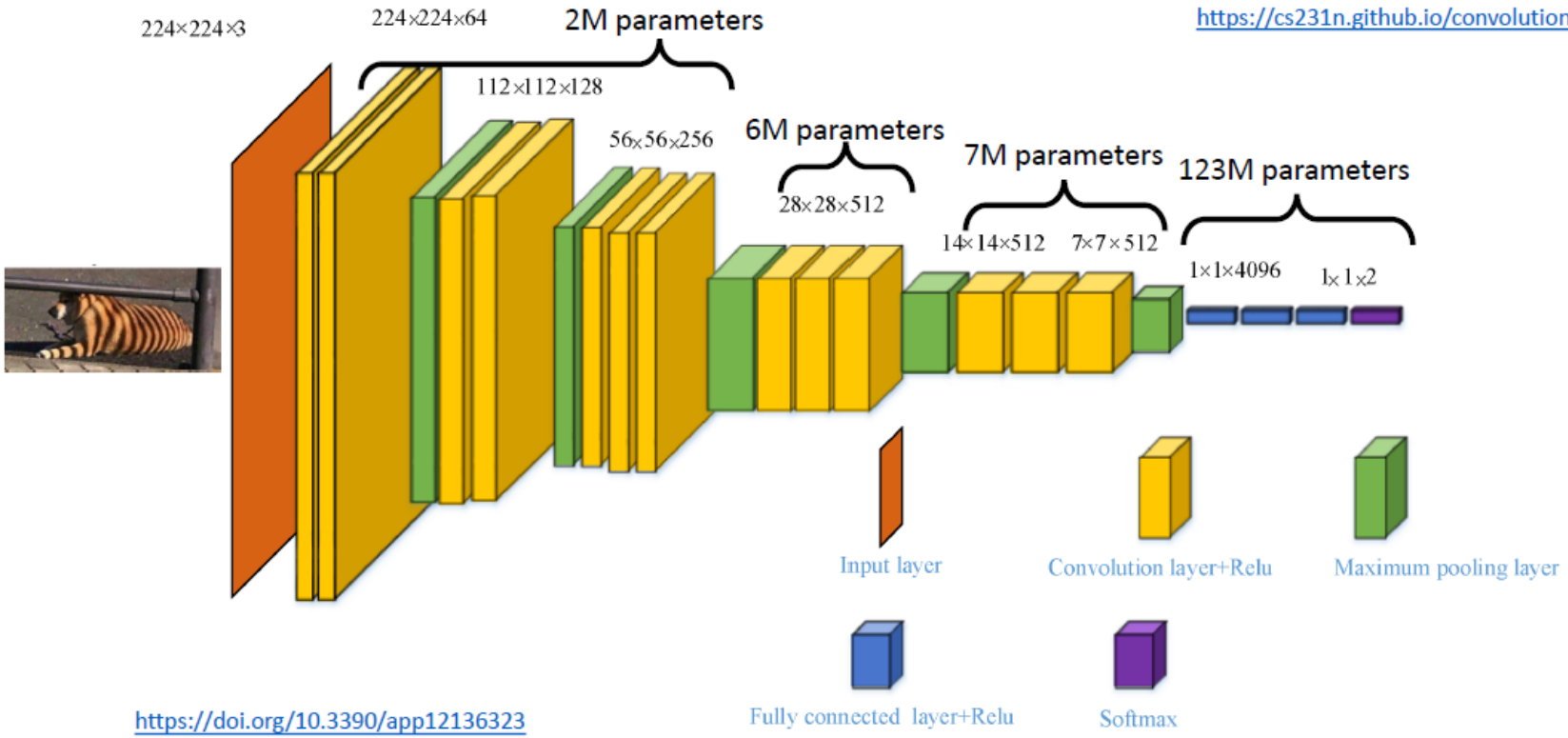
- Min pooling
- Average pooling
- There's also a difference between "local max pooling" (take the maximum value from a 2x2 region) and "global max pooling" (take the maximum value anywhere in the filter)

Convolutional Neural Net

So…what is a convolutional net?

- A network with one or more layers that are feature maps
- A layer with feature maps is called a “convolutional layer”
- Often, convolutional layers are alternated with pooling layers.
- Since these nets have many fewer connections
  - They train faster than fully-connected networks
  - They need comparatively fewer training examples

VGG16 Network: 138MM Parameters

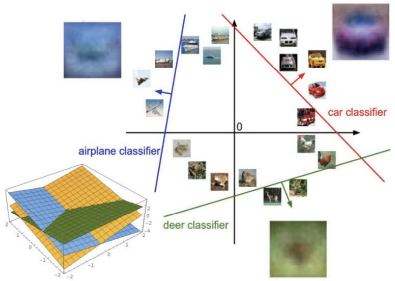


https://cs231n.github.io/convolutional-networks

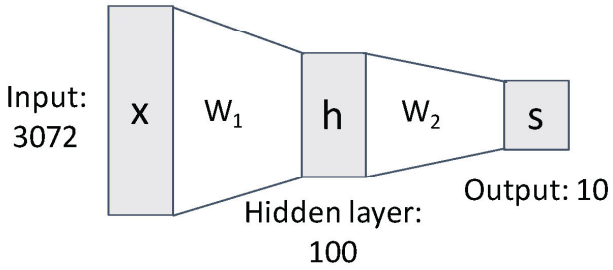
https://doi.org/10.3390/app12136323

Computer Vision: Without Convolution

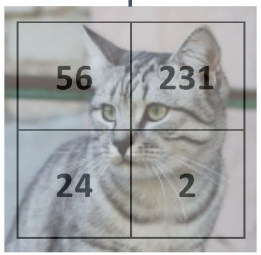
f(x,W) = Wx



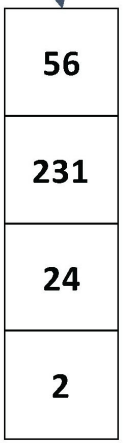
$$f = W_2 \max(0, W_1 x)$$



Stretch pixels into column



Input image (2, 2)



(4,)

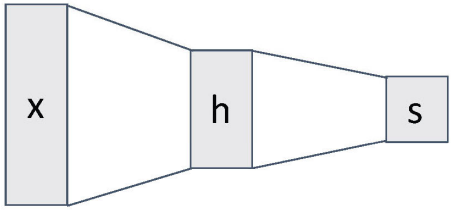
Problem: So far our classifiers don't respect the spatial structure of images!

Solution: Define new computational nodes that operate on images!

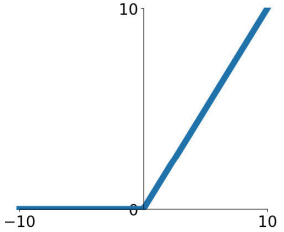
Computer Vision: Convolutional Components

Components of a Convolutional Network

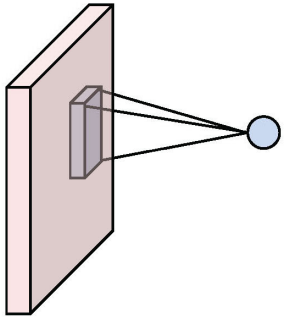
Fully-Connected Layers



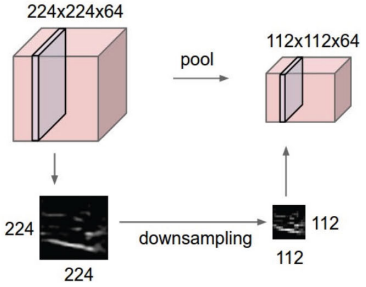
Activation Function



Convolution Layers



Pooling Layers



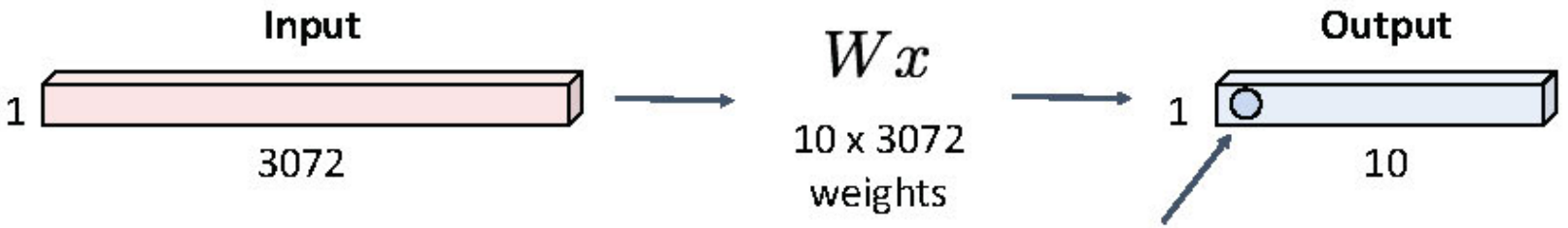
Normalization

$$\hat{x}_{i,j} = \frac{x_{i,j} - \mu_j}{\sqrt{\sigma_j^2 + \varepsilon}}$$

Computer Vision: Without Convolution

Fully-Connected Layer

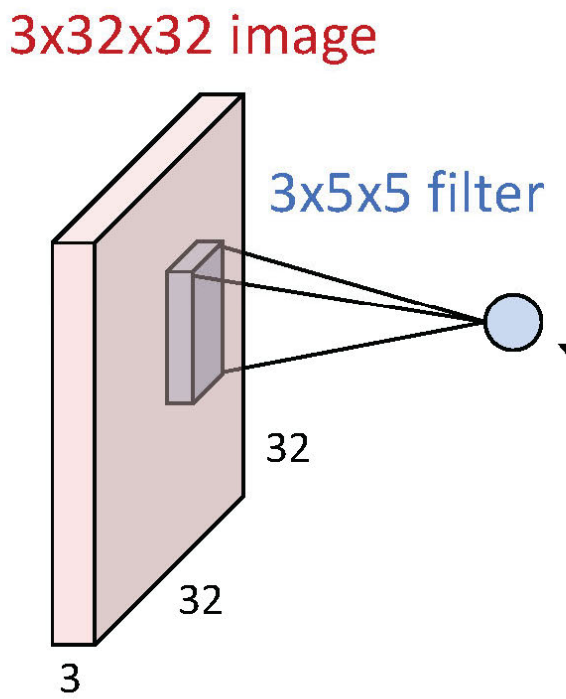
32x32x3 image -> stretch to 3072 x 1



1 number:
the result of taking a dot product between a row of W and the input (a 3072-dimensional dot product)

Convolution Layer

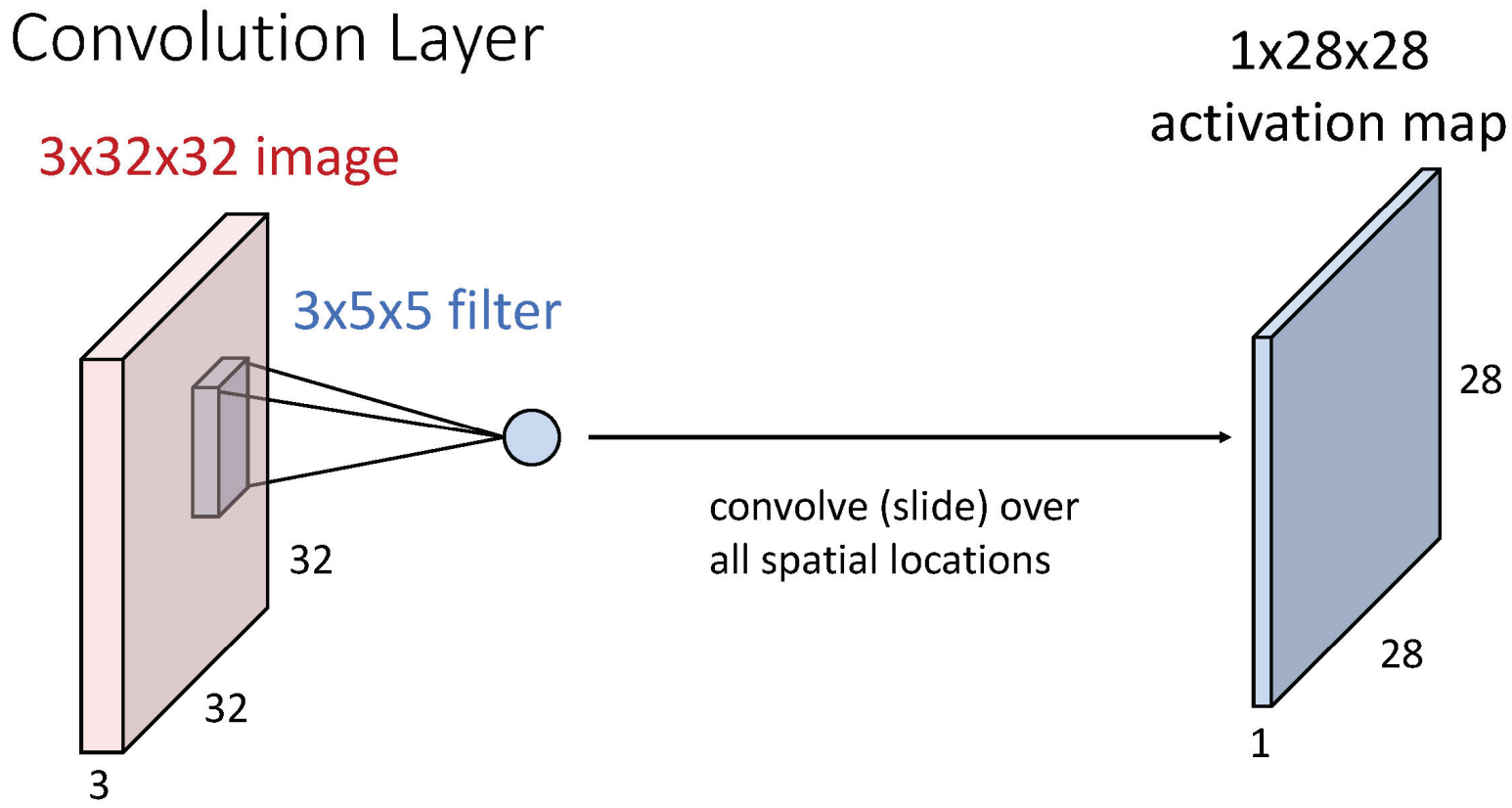
Convolution Layer



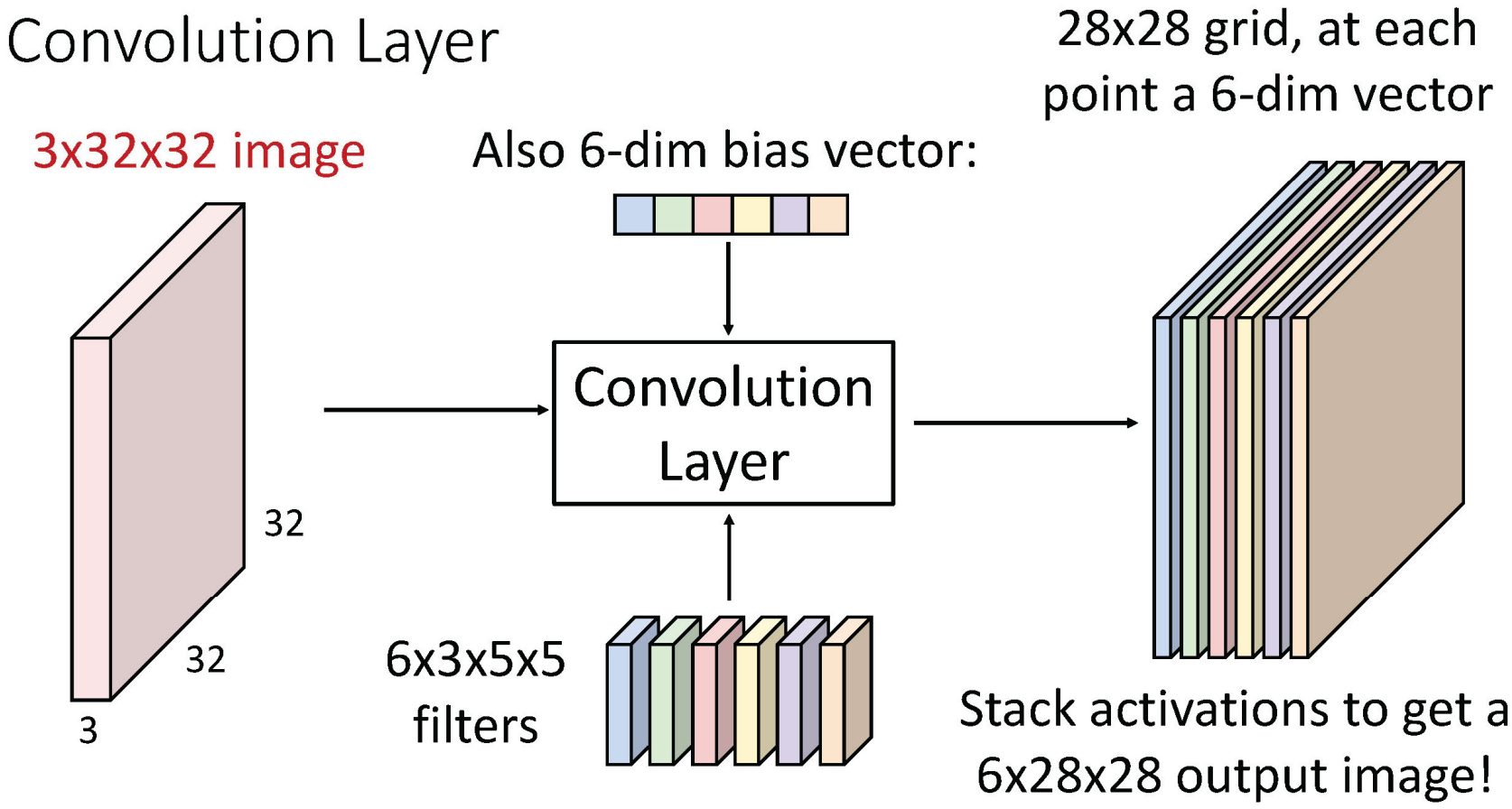
1 number:
the result of taking a dot product between the filter and a small 3x5x5 chunk of the image
(i.e. 3*5*5 = 75-dimensional dot product + bias)

$$w^T x + b$$

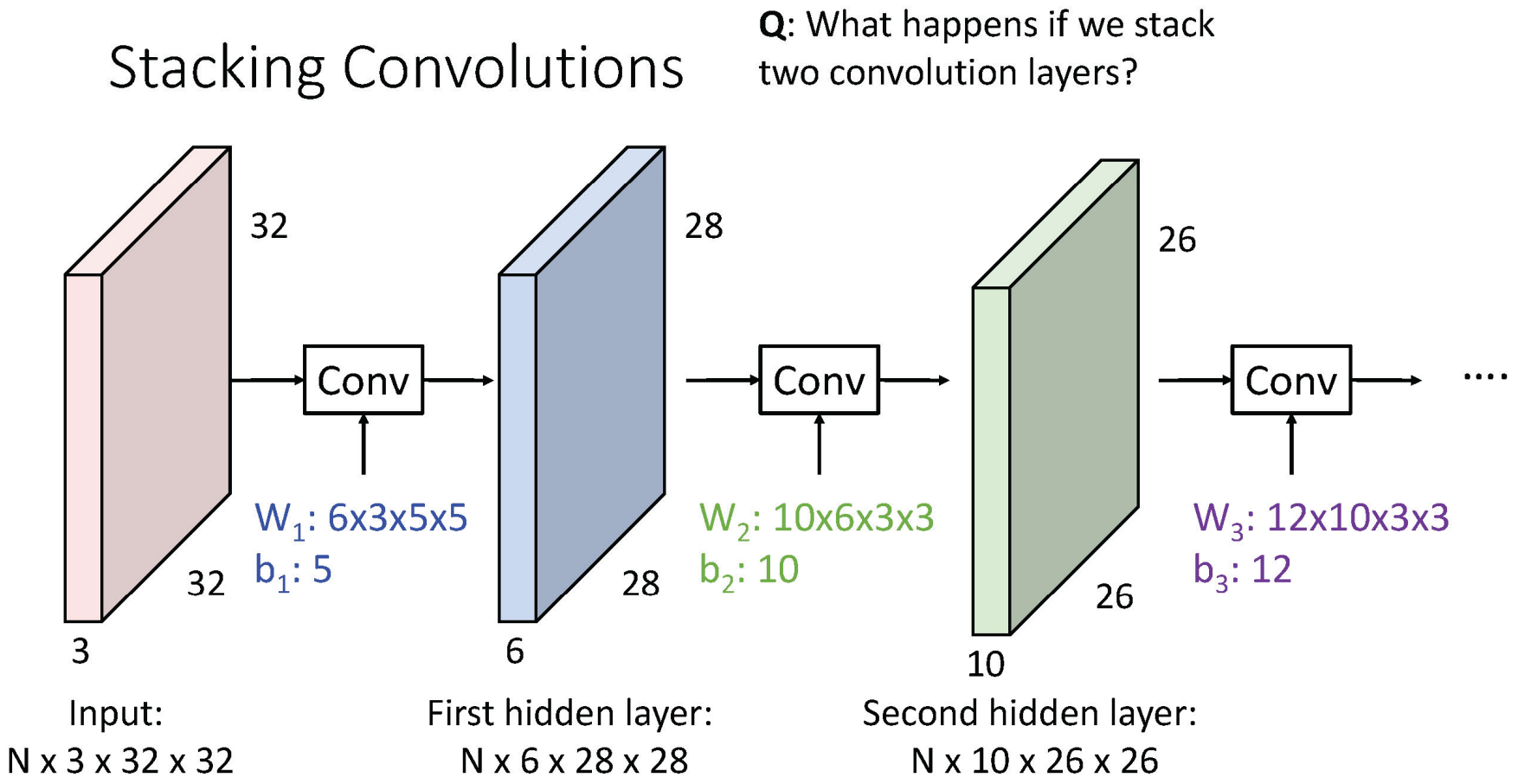
Convolution Layer



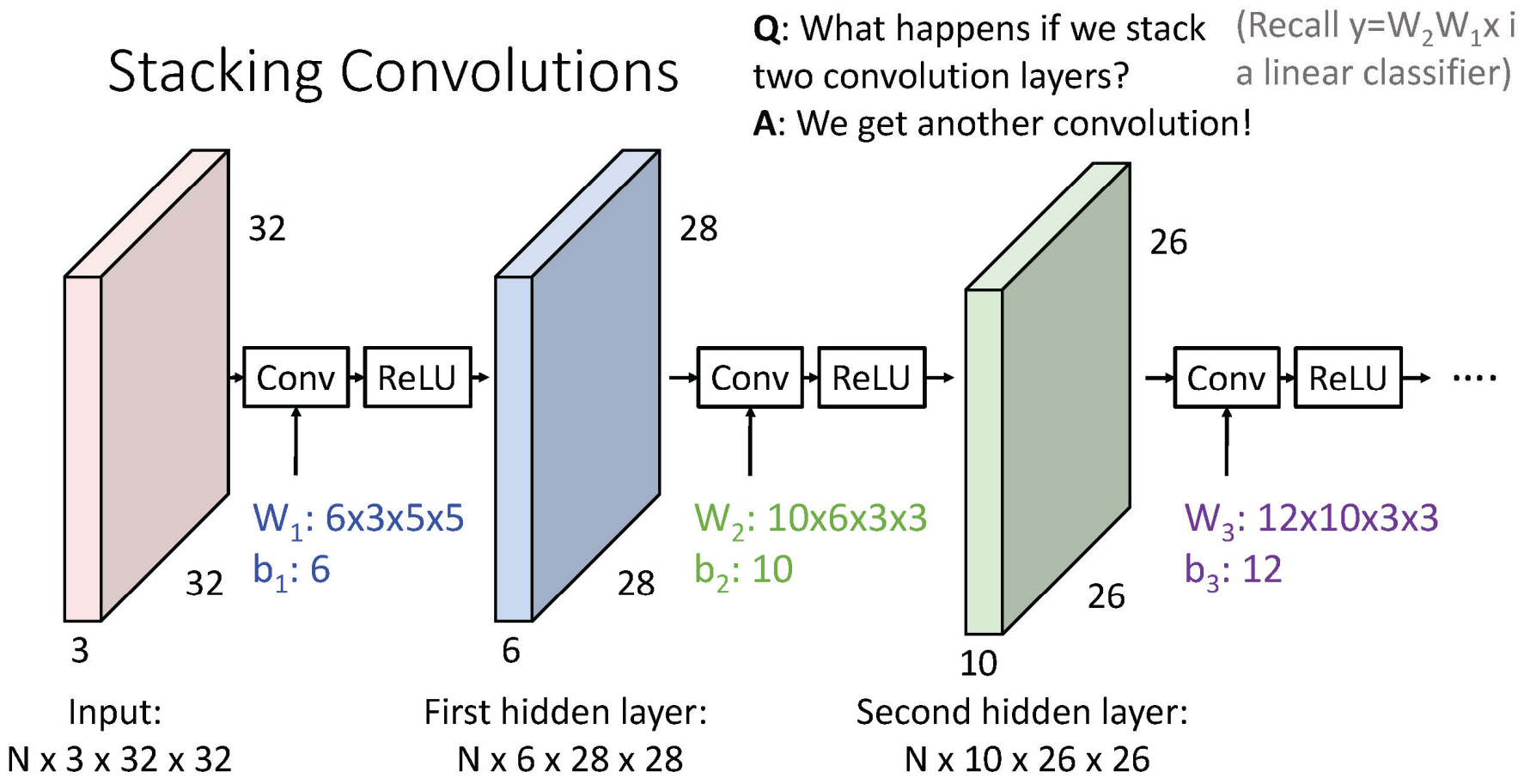
Convolution Layer



Stacking Convolution Layer

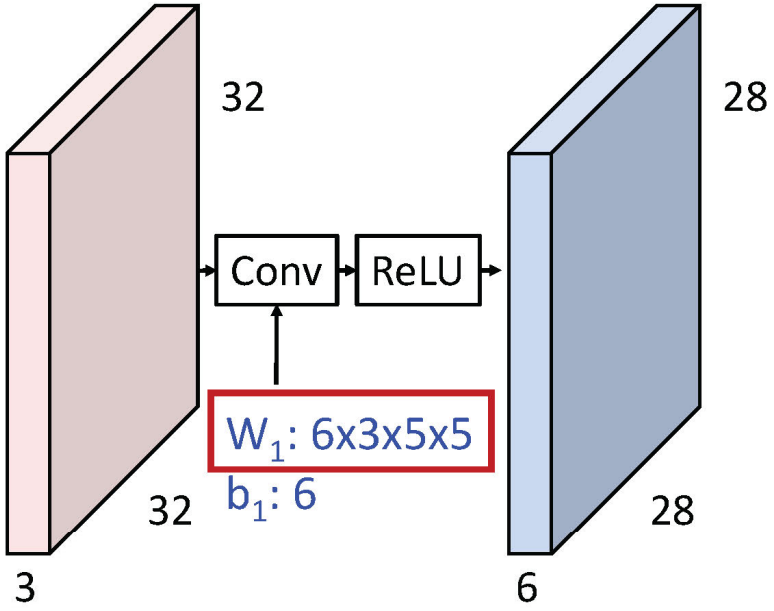


Stacking Convolution Layer



Convolution Layers

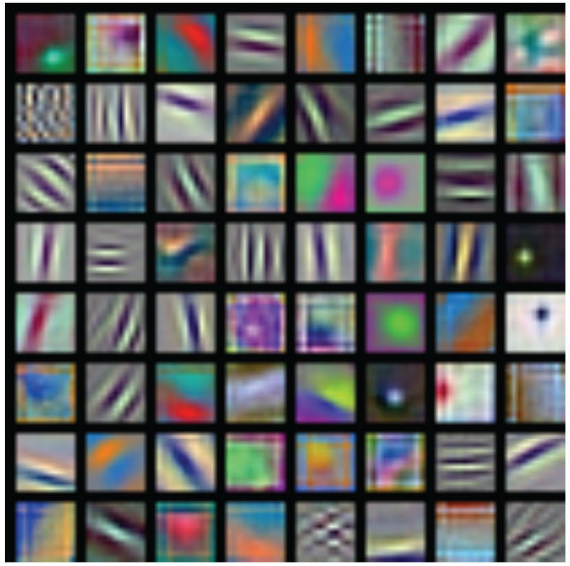
What do convolutional filters learn?



Input:
N x 3 x 32 x 32

First hidden layer:
N x 6 x 28 x 28

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



AlexNet: 64 filters, each 3x11x11

Convolution: Output Size

A closer look at spatial dimensions

0	0	0	0	0	0	0	0	0
0								0
0								0
0								0
0								0
0								0
0								0
0								0
0	0	0	0	0	0	0	0	0

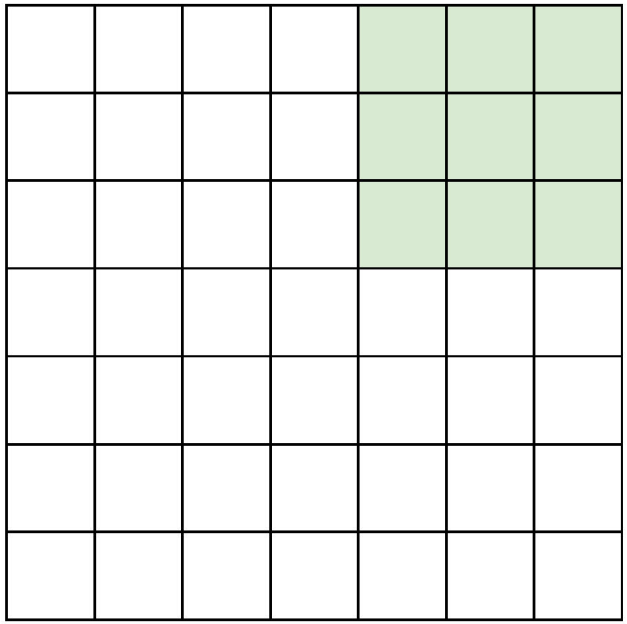
Input: 7x7
Filter: 3x3
Output: 5x5

In general:
Input: W
Filter: K
Padding: P
Output: W – K + 1 + 2P

Very common:
Set P = (K – 1) / 2 to make output have same size as input!

Convolution: Output Size

Strided Convolution



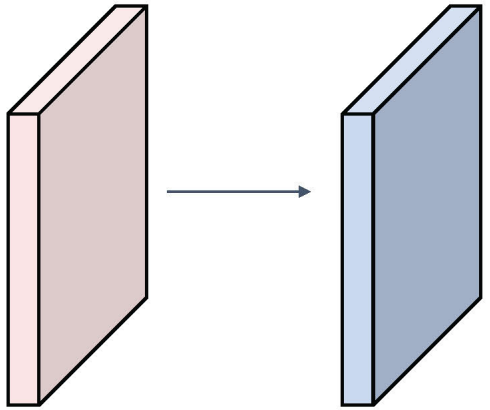
Input: 7x7
Filter: 3x3
Stride: 2

Output: 3x3

In general:
Input: W
Filter: K
Padding: P
Stride: S
Output: (W – K + 2P) / S + 1

Convolution: Learnable Parameters

Convolution Example



Input volume: 3 x 32 x 32
10 5x5 filters with stride 1, pad 2

Output volume size: 10 x 32 x 32
Number of learnable parameters: **760**
Parameters per filter: 3*5*5 + 1 (for bias) = 76
10 filters, so total is 10 * 76 = **760**

Convolution: Calculations

Convolution Summary

Input: C_{in} x H x W

Hyperparameters:

- **Kernel size**: K_H x K_W
- **Number filters**: C_{out}
- **Padding**: P
- **Stride**: S

Weight matrix: C_{out} x C_{in} x K_H x K_W
giving C_{out} filters of size C_{in} x K_H x K_W

Bias vector: C_{out}

Output size: C_{out} x H' x W' where:

- H' = (H – K + 2P) / S + 1
- W' = (W – K + 2P) / S + 1

Common settings:

$K_H = K_W$ (Small square filters)

P = (K – 1) / 2 ("Same" padding)

C_{in}, C_{out} = 32, 64, 128, 256 (powers of 2)

K = 3, P = 1, S = 1 (3x3 conv)

K = 5, P = 2, S = 1 (5x5 conv)

K = 1, P = 0, S = 1 (1x1 conv)

K = 3, P = 1, S = 2 (Downsample by 2)

Convolution: PyTorch

PyTorch Convolution Layer

Conv2d

CLASS `torch.nn.Conv2d(`*`in_channels, out_channels, kernel_size, stride=1, padding=0, dilation=1, groups=1, bias=True, padding_mode='zeros'`*`)` [SOURCE]

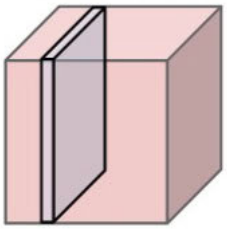
Applies a 2D convolution over an input signal composed of several input planes.

In the simplest case, the output value of the layer with input size (N, C_{in}, H, W) and output $(N, C_{\text{out}}, H_{\text{out}}, W_{\text{out}})$ can be precisely described as:

$$\text{out}(N_i, C_{\text{out}_j}) = \text{bias}(C_{\text{out}_j}) + \sum_{k=0}^{C_{\text{in}}-1} \text{weight}(C_{\text{out}_j}, k) \star \text{input}(N_i, k)$$

Convolution: Max Pooling

Max Pooling



Single depth slice

x

1	1	2	4
5	**6**	7	**8**
3	2	1	0
1	2	3	**4**

y

Max pooling with 2x2 kernel size and stride 2

6	8
3	4

Introduces **invariance** to small spatial shifts

No learnable parameters!

Convolution: Max Pooling

Pooling Summary

Input: C x H x W

Hyperparameters:

- Kernel size: K
- Stride: S
- Pooling function (max, avg)

Output: C x H' x W' where

- H' = (H – K) / S + 1
- W' = (W – K) / S + 1

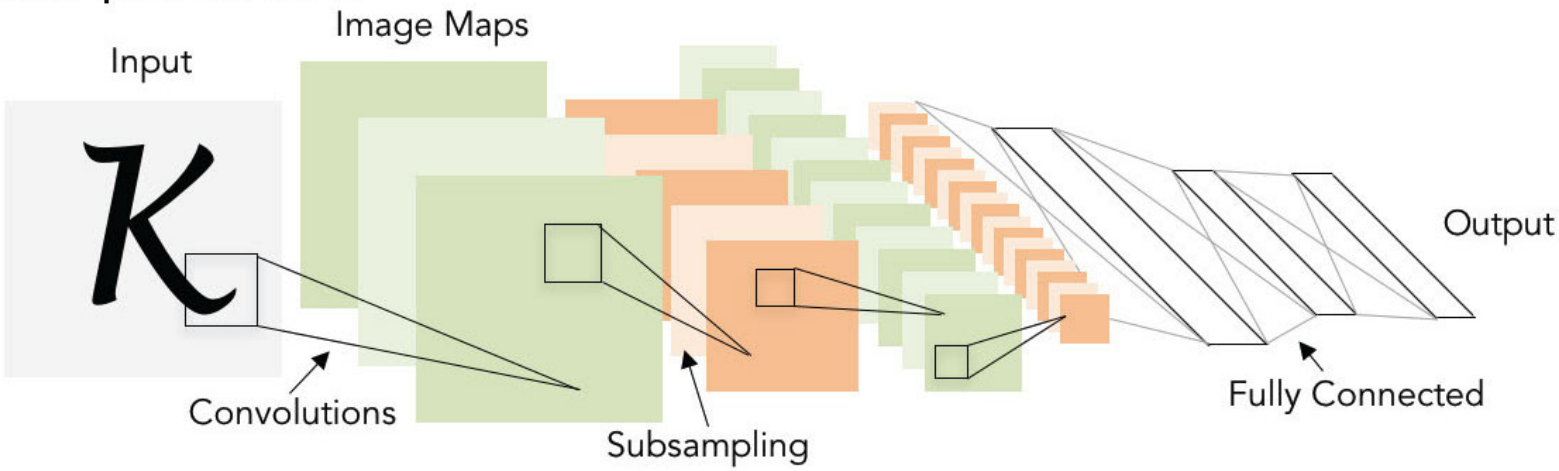
Learnable parameters: None!

Common settings:
max, K = 2, S = 2
max, K = 3, S = 2 (AlexNet)

Convolutional Networks

Classic architecture: [Conv, ReLU, Pool] x N, flatten, [FC, ReLU] x N, FC

Example: LeNet-5



Convolutional Architectures

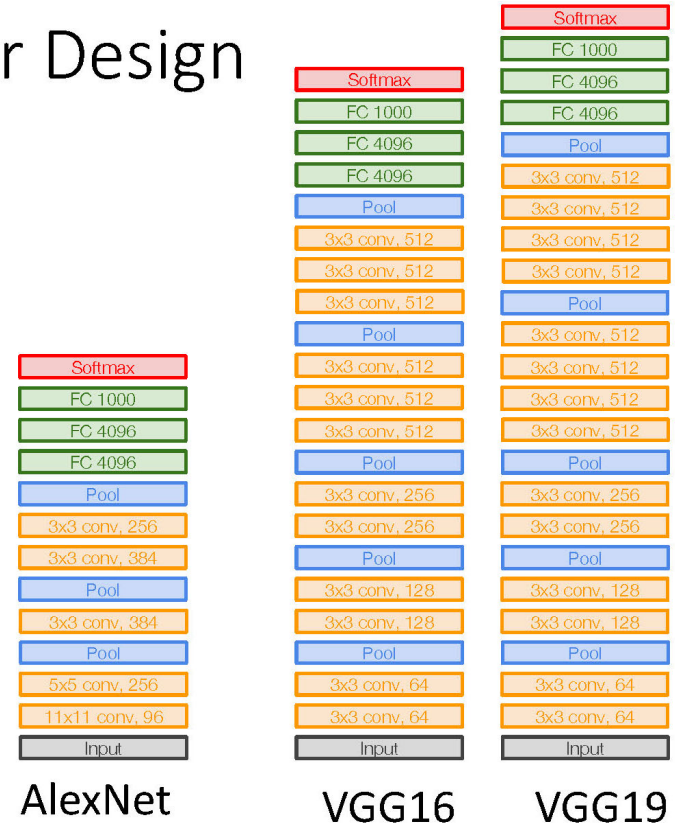
VGG: Deeper Networks, Regular Design

VGG Design rules:

All conv are 3x3 stride 1 pad 1

All max pool are 2x2 stride 2

After pool, double #channels



Residual Networks

Solution: Change the network so learning identity functions with extra layers is easy!