Convolution Neural Network (CNN)

- Local details in CNN
- Properties of visual structure
- Layers of CNN
- Weights updating (Backward propagation)
- Examples

CIFAR: from Wikipedia

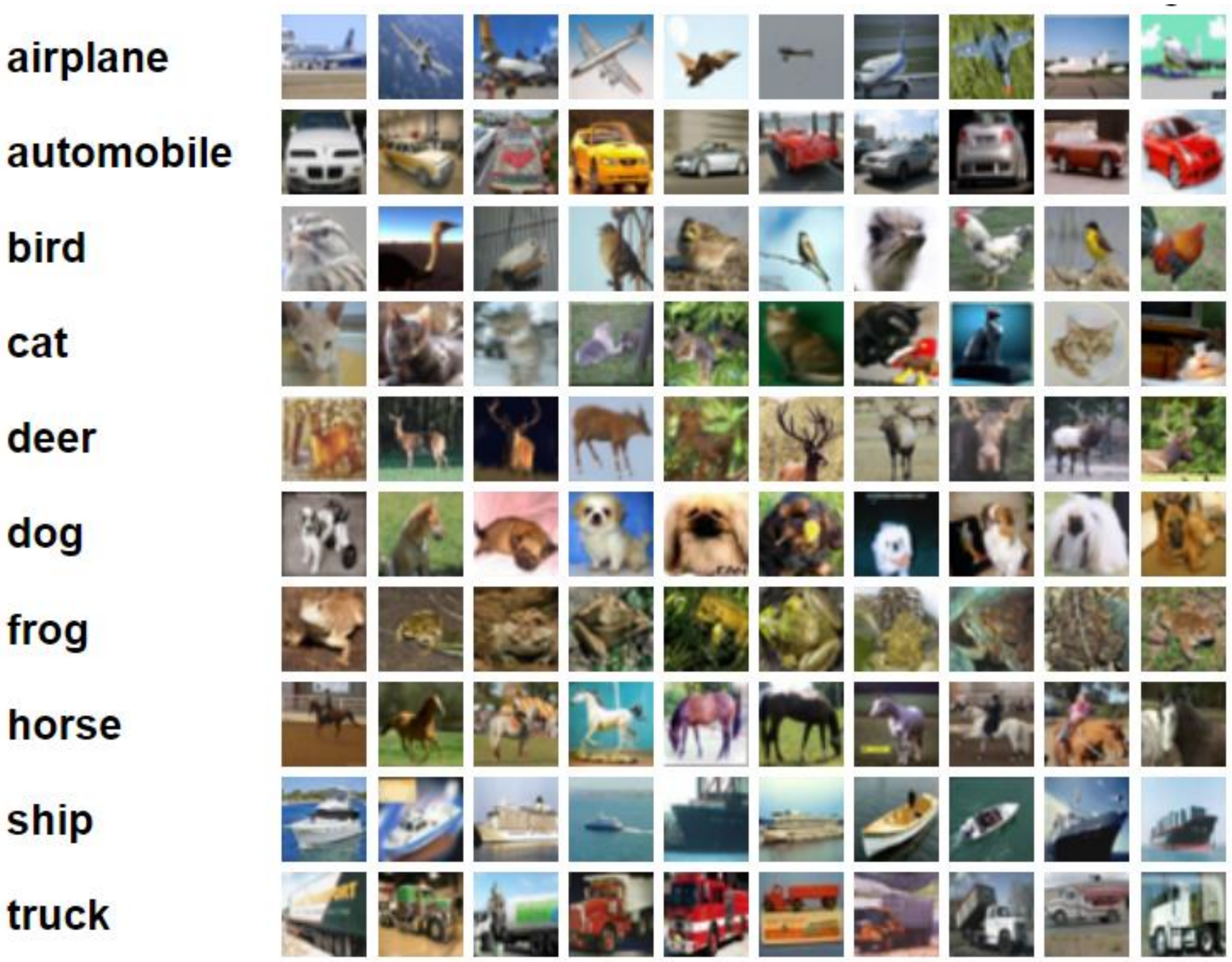
In 2004, Geoffrey Hinton began leading CIFAR's Neural Computation & Adaptive Perception program. Its members included Yoshua Bengio and Yann LeCun, among other neuroscientists, computer scientists, biologists, electrical engineers, physicists, and psychologists.
Together, they confirmed Hinton's conviction about the power of neural networks when they created computing systems that mimicked human intelligence.
Today, the three are widely acknowledged as the pioneers of deep learning.
In 2019, the Association for Computing Machinery (ACM) named Hinton, Bengio and LeCun as recipients of the 2018 ACM A.M. Turing Award for conceptual and engineering breakthroughs that have made deep neural networks a critical component of computing.[13]

CIFAR-10:

A dataset contains 60,000 color images of 32x32 pixels in 3 channels divided into 10 classes. Each class contains 6,000 Images. The training set contains 50,000 images, while the test set provides 10,000 images.

Deep learning: In 2012, Alex Krizhevsky, Ilya Sutskever, and Geoffrey Hinton published an article titled "ImageNet Classification with Deep Convolutional Neural Networks" in the Proceedings of Neural Information Processing Systems(NIPS). At the end of the paper, they wrote:

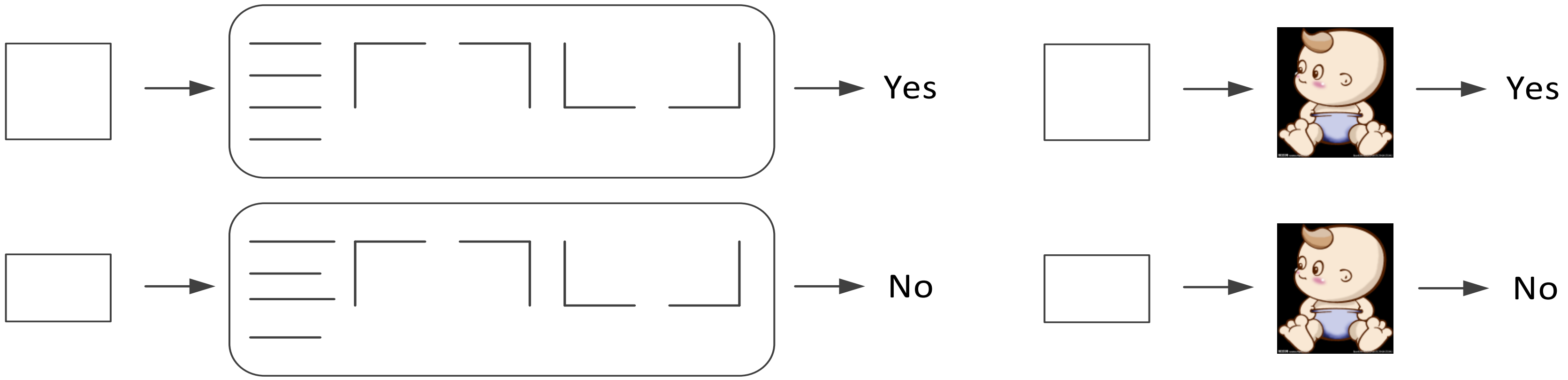
"It is notable that our network's performance degrades if a single convolutional layer is removed. For example, removing any of the middle layers results in a loss of about 2% for the top-1 performance of the network. So the depth really is important for achieving our results."


airplane
automobile
bird
cat
deer
dog
frog
horse
ship
truck

What is CNN?

If we want to judge whether a quadrangle is square or not, the rationale approach is to seek features of square, such as same length for four sides and four 90 degree corner angles

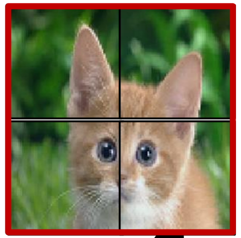
Use filters to extract features, and use features for classification.



Properties of visual structure

Local Processing: pixels close together go together

receptive fields capture local detail



Across Space: the same what, no matter where

recognize the same pattern in different places, translation invariant

Properties of visual structure- subsampling

Why CNN for Image

- 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

Basic Concept of Convolution and Polling in CNN Operations

- **Consider an image of 500x500 pixels. The no. of neurons in input layer is set to 500x500 with 10^8 neurons in hidden layer. Each connection between one input neuron and one neuron in hidden layer is set with a weight parameter**

- **The no.of weight parameters between input layer and hidden layer is $500 \times 500 \times 10^8 = 25 \times 10^{12}$. This may demand enormous amount of computations**

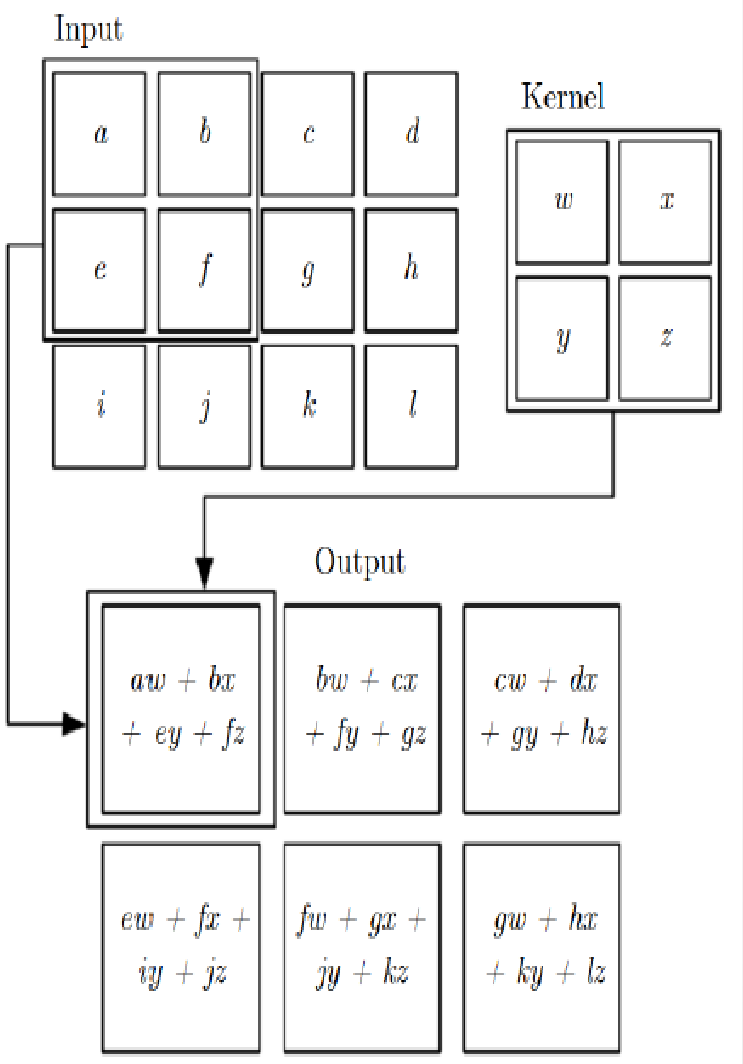
- **Design a 10x10 filter to extract the local features in an image. Then a hidden layer neuron is connected to a 10x10 area of the image through the filter**

Properties of CNN

- Patterns are much smaller than the whole image; extract features for classification.
- The same pattern appears in different regions; convolution detects patterns in different regions of an image
- Weights sharing of neurons to reduce the number of neurons in CNN, (same filter to extract features at different locations)
- Use subsampling to reduce the amount of features (features in a bigger range) and the computational complexity of the network.

2-D Convolution (without kernel flipping)

Example of 'valid' 2-D convolution (without kernel flipping) where a 3x4 matrix convolved with a 2x2 kernel to output a 2x3 matrix



http://goodfeli.github.io/dlbook/contents/convnets.html

CNN Property 1

6 x 6 image

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

convolve →

Filter 1

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

Filter 2

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

Output

3	-1	-3	1
-3	1	0	-3
-3	-3	0	1
3	-2	-2	-1

-1	-1	-1	-1
-1	-1	-1	1
-1	-1	-2	1
-1	0	-4	3

Same patterns produce the largest outputs

CNN – Convolution

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

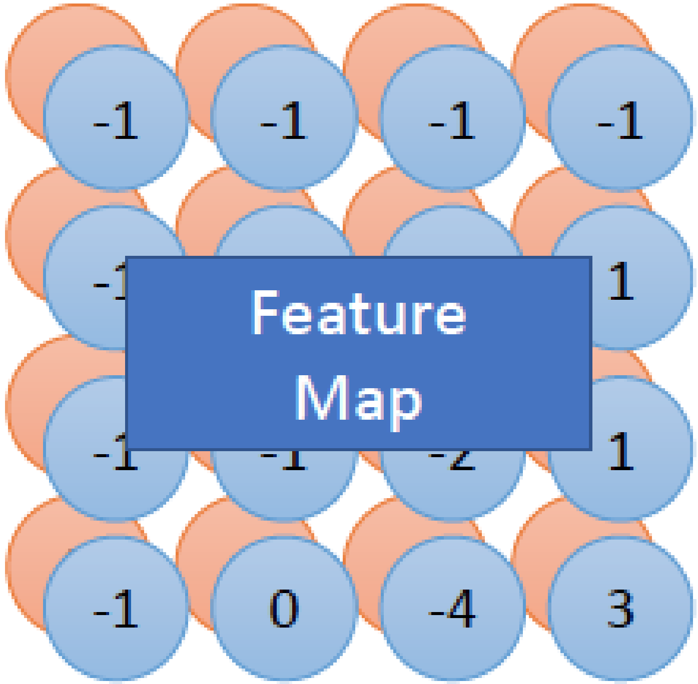
Filter 2

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

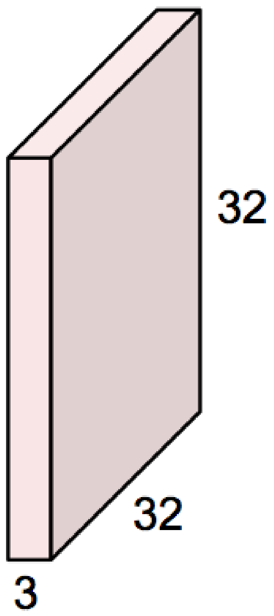
Do the same process for every filter



4 x 4 image

A Filter

A filter is a **spatially local** and **cross-channel** template
Convnet filters are **learned**

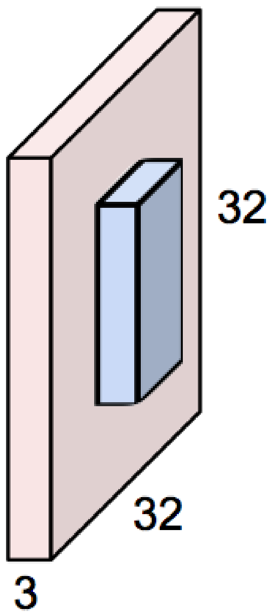


input is **3**x32x32 data
a color image (3 RGB channels) and square (32x32)

[figure adapted from A. Karpathy]

A Filter

A filter is a **spatially local** and **cross-channel** template
Convnet filters are **learned**



input is **3**x32x32 data
a color image (3 RGB channels) and square (32x32)

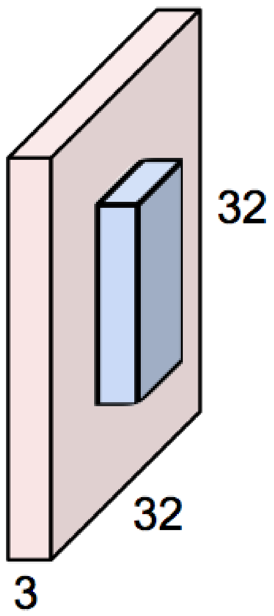
filter is **3**x5x5 weights

- spatially local: kernel size is 5x5
- cross-channel: connected across all input channels

[figure adapted from A. Karpathy]

A Filter

A filter is a **spatially local** and **cross-channel** template
Convnet filters are **learned**



input is **3**x32x32 data
a color image (3 RGB channels) and square (32x32)

filter is **3**x5x5 weights
- spatially local: kernel size is 5x5
- cross-channel: connected across all input channels

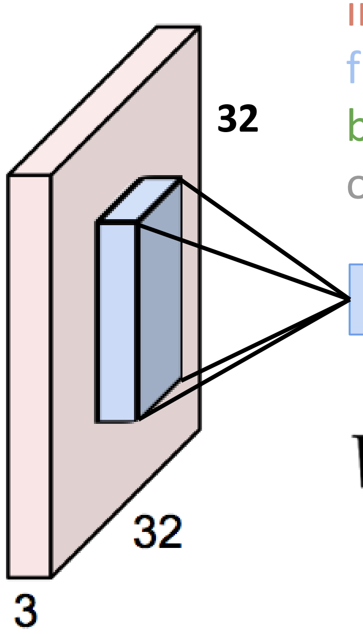
total parameters:
$3*5^2$ = 75 filter weights + 1 bias

[figure adapted from A. Karpathy]

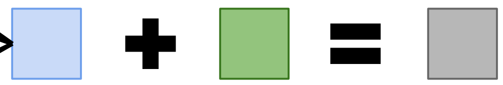
Credit to Cafe

Convolution

One filter evaluation is a dot product between the input window and weights + bias



input	3x32x32
filter	3x5x5
bias	1
output	1



$$Wx + b$$

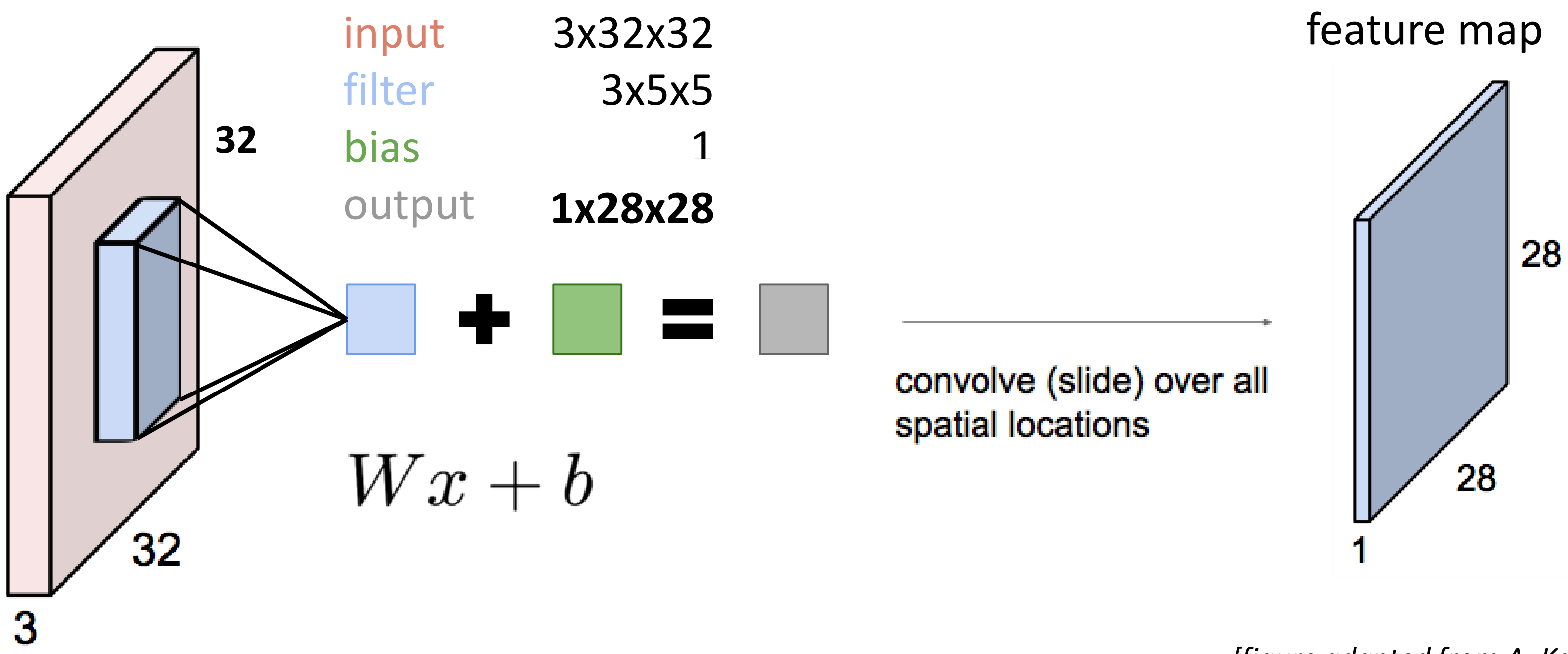
DL4j

[figure adapted from A. Karpathy]

Credit to Cafe

Convolution

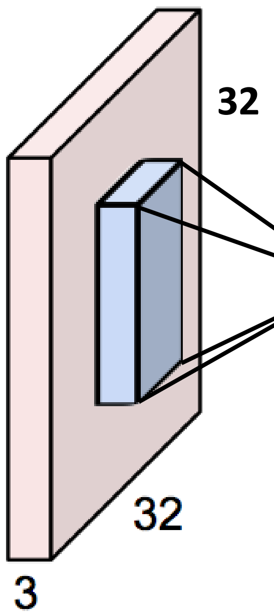
Convolving the filter with the input gives a **feature map**.



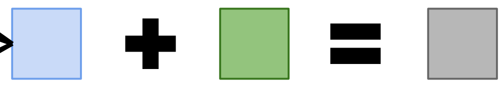
[figure adapted from A. Karpathy]

Convolution

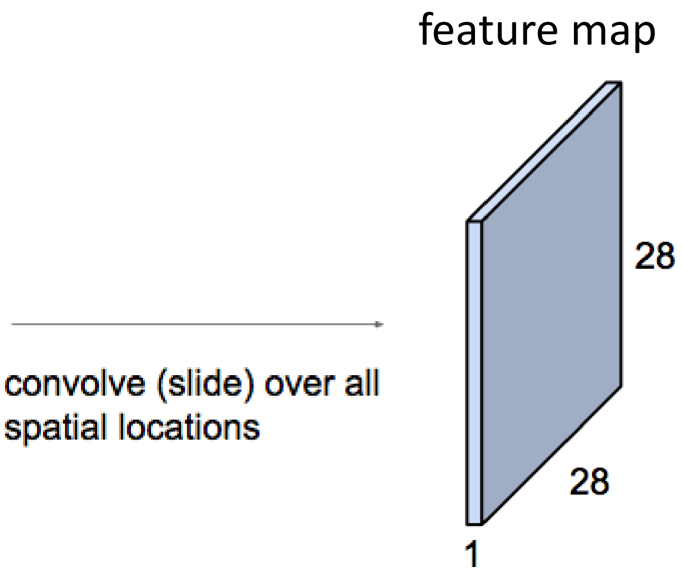
Convolving the filter with the input gives a **feature map**.



input	3x32x32
filter	3x5x5
bias	1
output	**1x28x28**



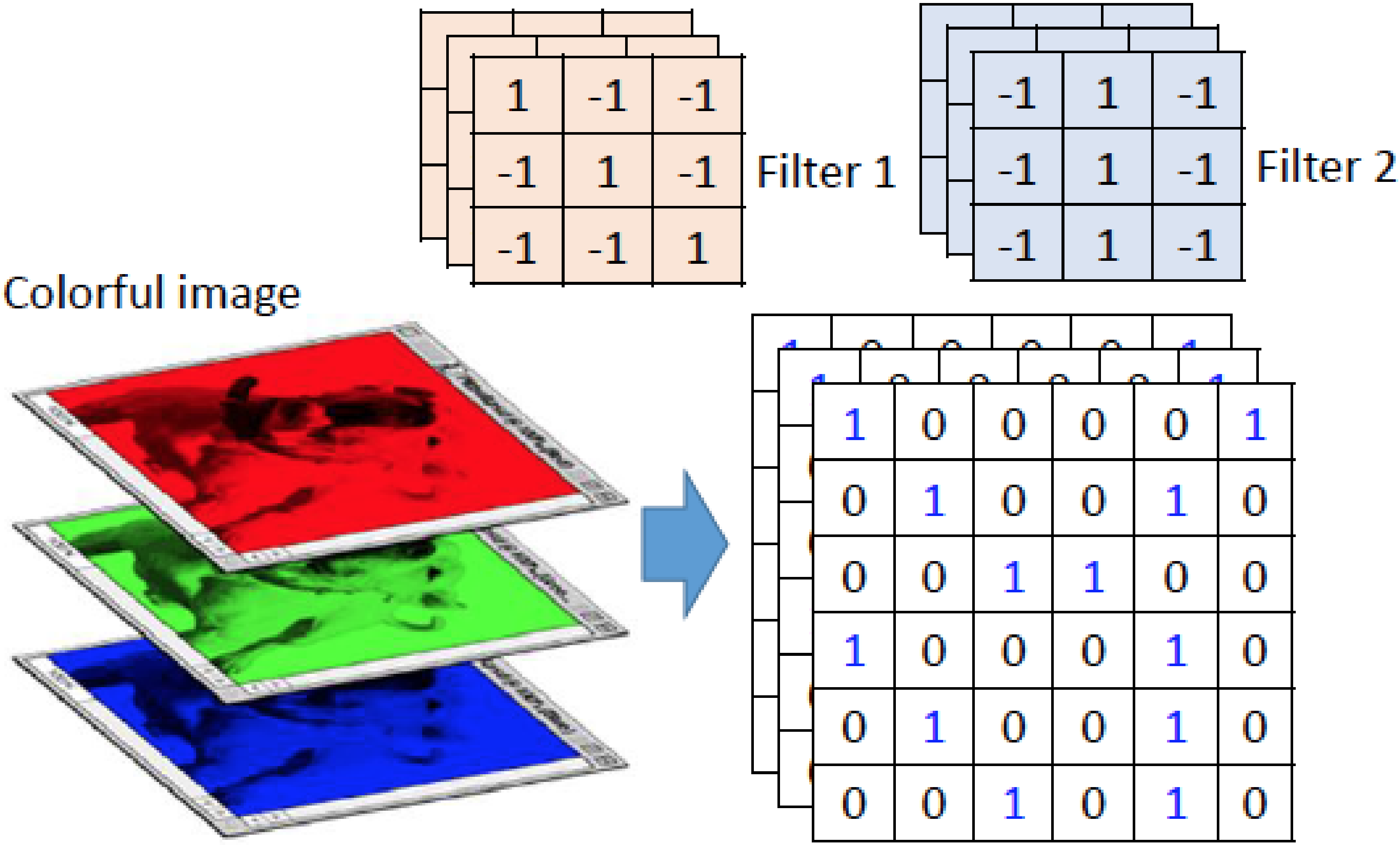
$$Wx + b$$

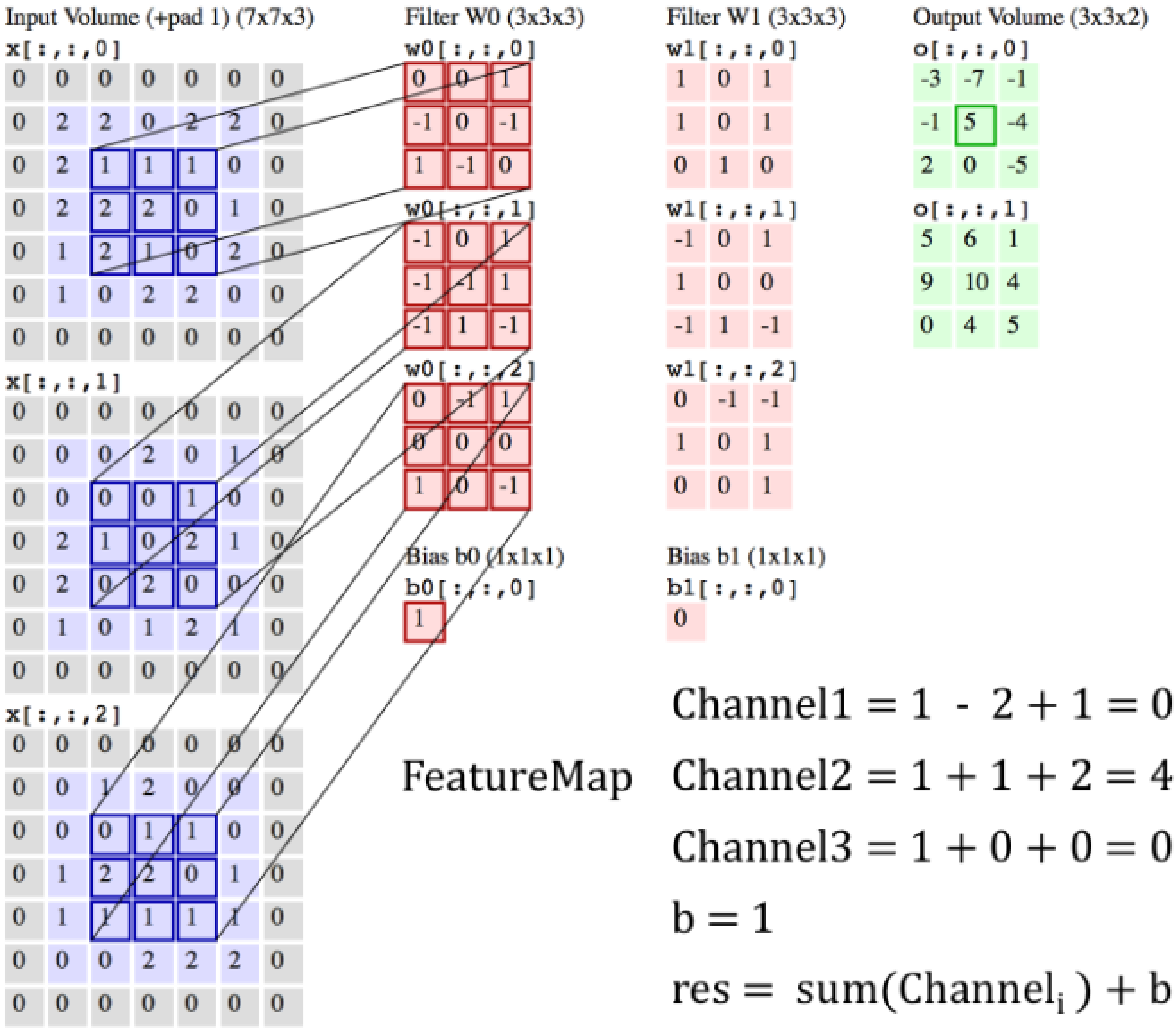


Filter parameters:	$3*5^2$ =	**75**
FC parameters:	$3*32^2$ =	**3,072**

[figure adapted from A. Karpathy]

CNN – Colorful image





X
InputShape = (7, 7,3)
strid = 1
pad = 1

W
FilterShape = (3,3,3,2)
(inChannel, Shape,Shape, outChannel)
b Shape = (1,1,1)

Y
OutputShape = (3, 3,2)

FeatureMap

$\text{Channel1} = 1 - 2 + 1 = 0$

$\text{Channel2} = 1 + 1 + 2 = 4$

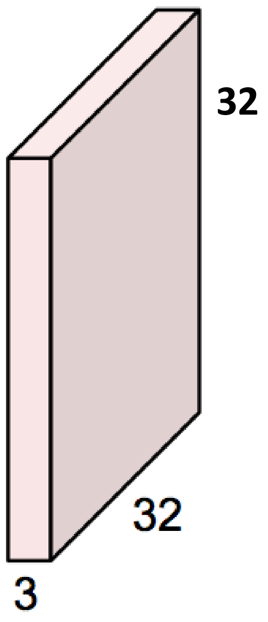
$\text{Channel3} = 1 + 0 + 0 = 0$

$b = 1$

$\text{res} = \text{sum}(\text{Channel}_i) + b = 5$

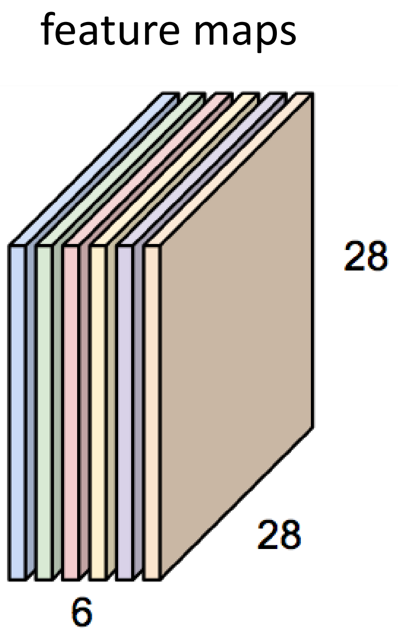
Convolution Layer (conv)

Convolution layers have multiple **filters** for more modeling capacity



input	3x32x32
filters	**6**x3x5x5
bias	**6**
output	**6**x28x28

Convolution Layer

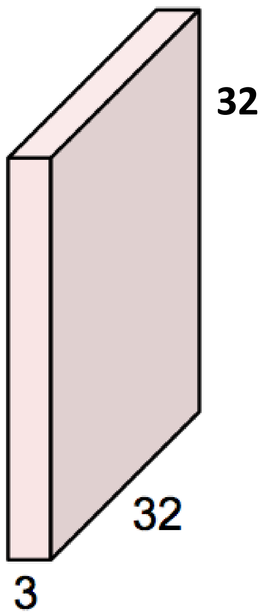


[figure adapted from A. Karpathy]

Credit to Cafe

Convolution Layer (conv)

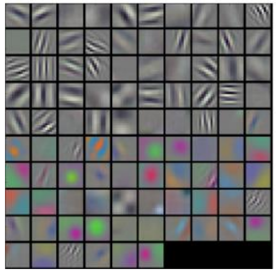
Convolution layers have multiple **filters** for more modeling capacity



input	3x32x32
filters	**6**x3x5x5
bias	**6**
output	**6**x28x28

Convolution Layer

Learned Filters from AlexNet

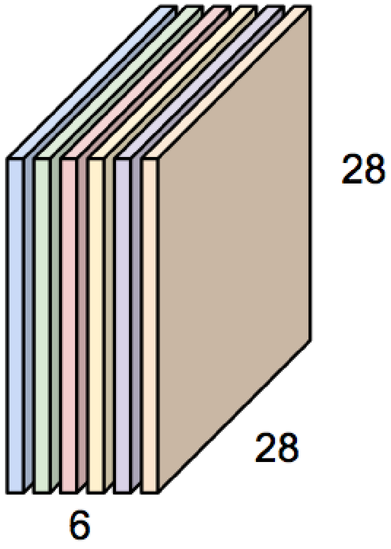


conv_1

conv_1 has 96 filters for edge, color, and frequency

richer than 3D RGB

feature maps



[figure adapted from A. Karpathy]

m*m image, k*k filter

Padding

Full

- Add zero-padding to the image enough for every pixel to be visited k times in each direction, with output size: **(m + k - 1) x (m + k - 1)**

Valid

- With no zero-padding, kernel is restricted to traverse only within the image, with output size: **(m - k + 1) x (m - k + 1)**

Same

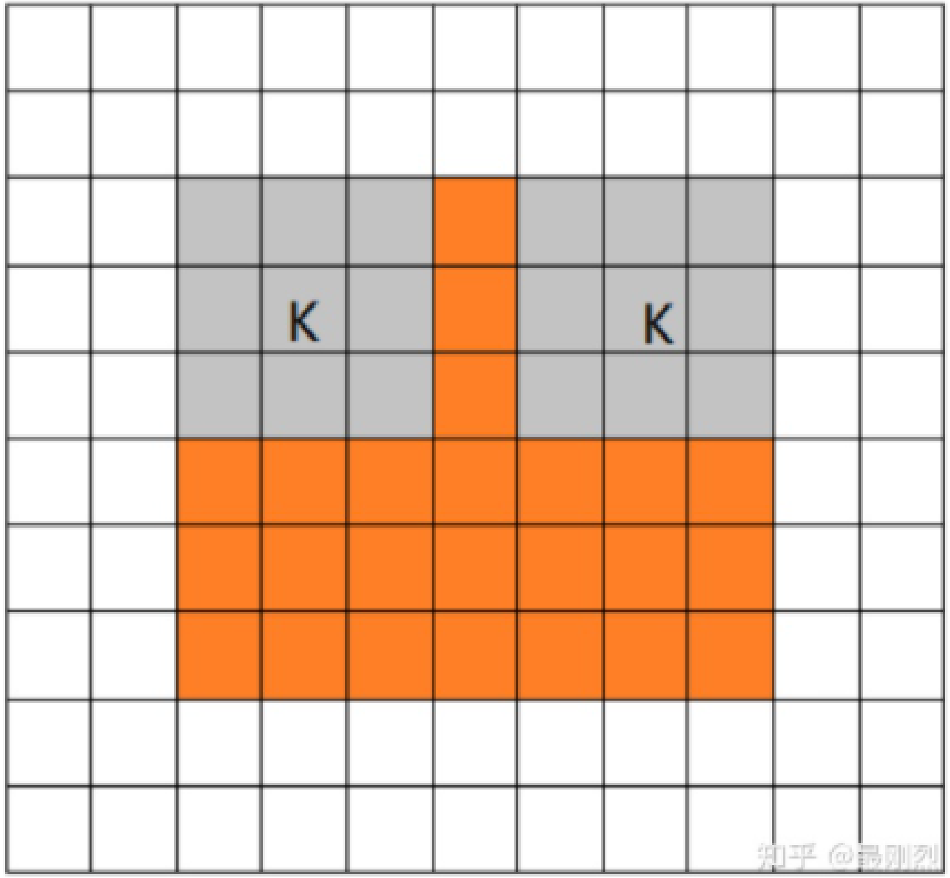
- Add zero-padding to the image to have the output of the same size as the image, i.e., m x m

Stride ***s***

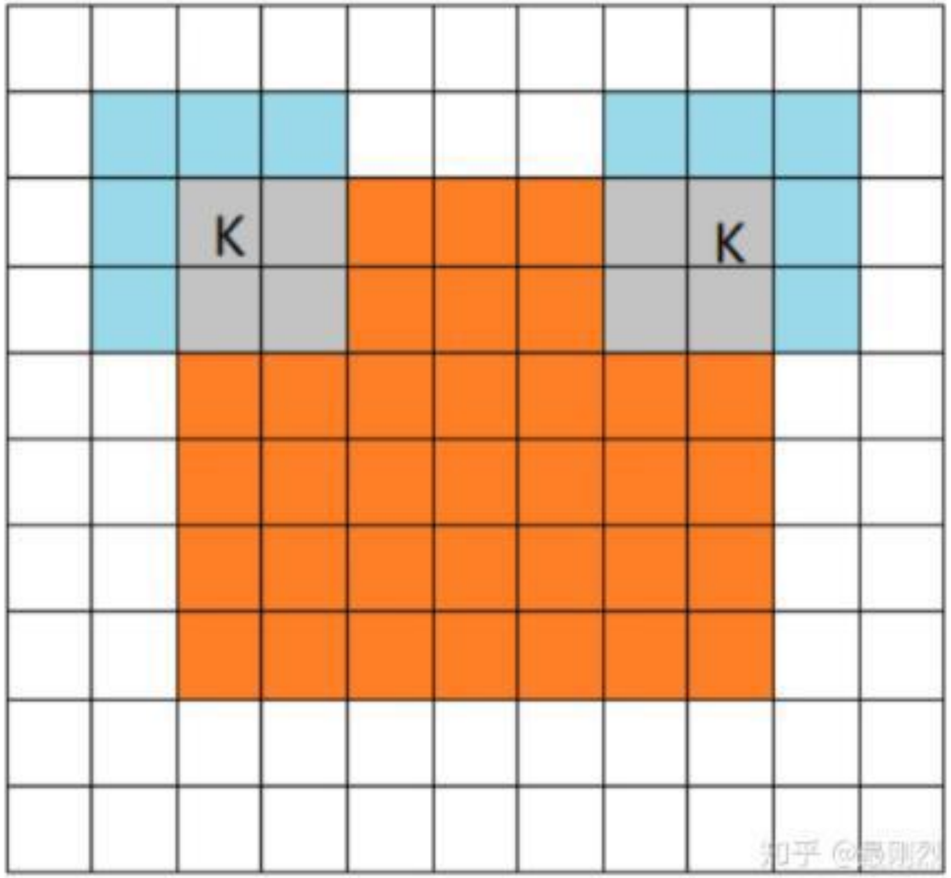
- Down-sampling the output of convolution by sampling only every s pixels in each direction.
- For instance, the output of 'valid' convolution with stride s results in an output of size $\frac{m-k+s}{s}$ **x** $\frac{m-k+s}{s}$

http://goodfeli.github.io/dlbook/contents/convnets.html

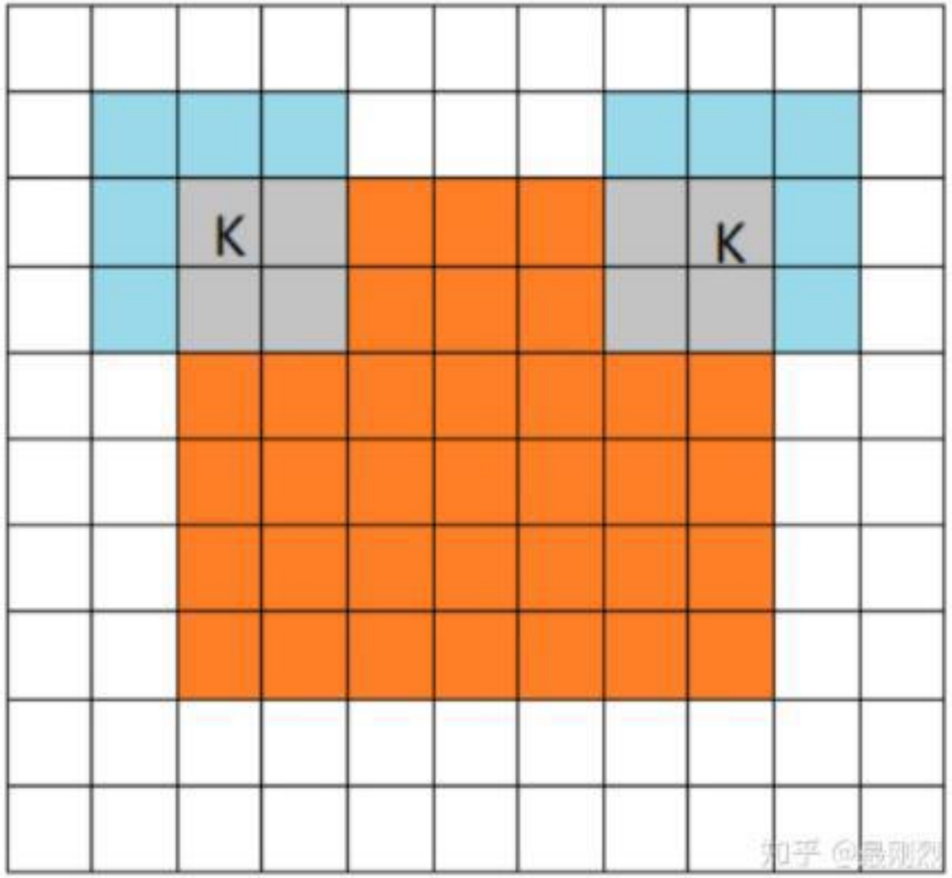
Valid, no zero-padding, k*k filter, k=3, m=7
Dim. of feature map=(m-k+1)*(m-k+1)



Full, k*k filter, k=3, m=7, output (m+k-1)*(m+k-1)



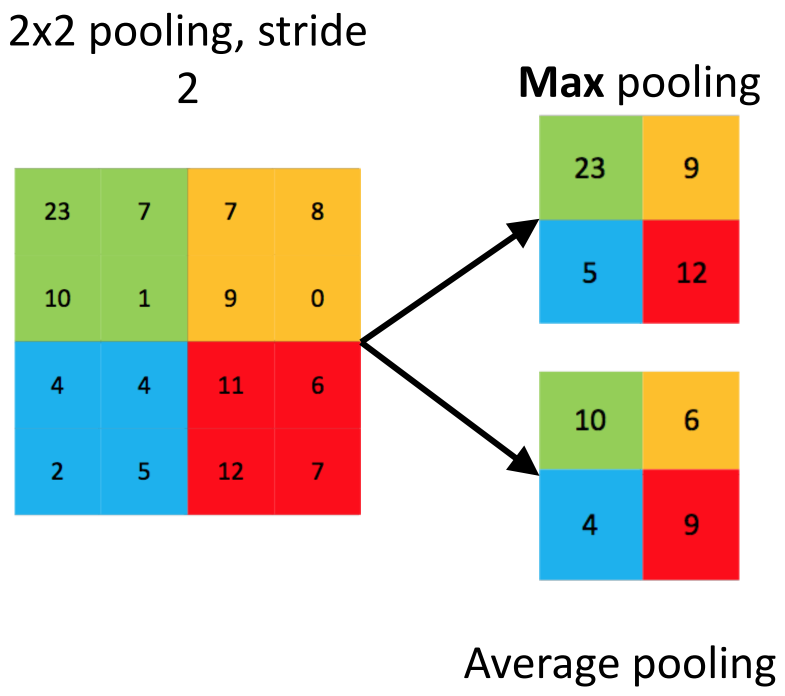
Same padding, x-k+1=m, x=m+k-1



Pooling (Pool)

Spatial summary by computing **operation** over **window** with **stride**

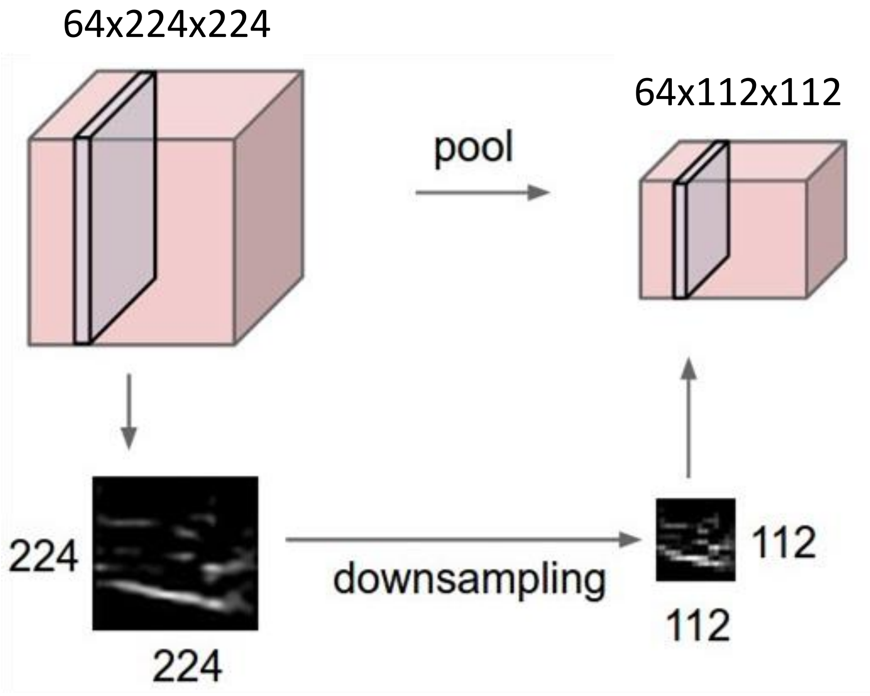
- overlapping or non-overlapping
- separate across channels
- Current fashion:
  3x3 max pooling with stride 2



[figure credit BDTI]

Pooling

- reduce resolution
- increase receptive field size
  for later layers
- save computation
- add invariance to translation/noise
  within pooling window

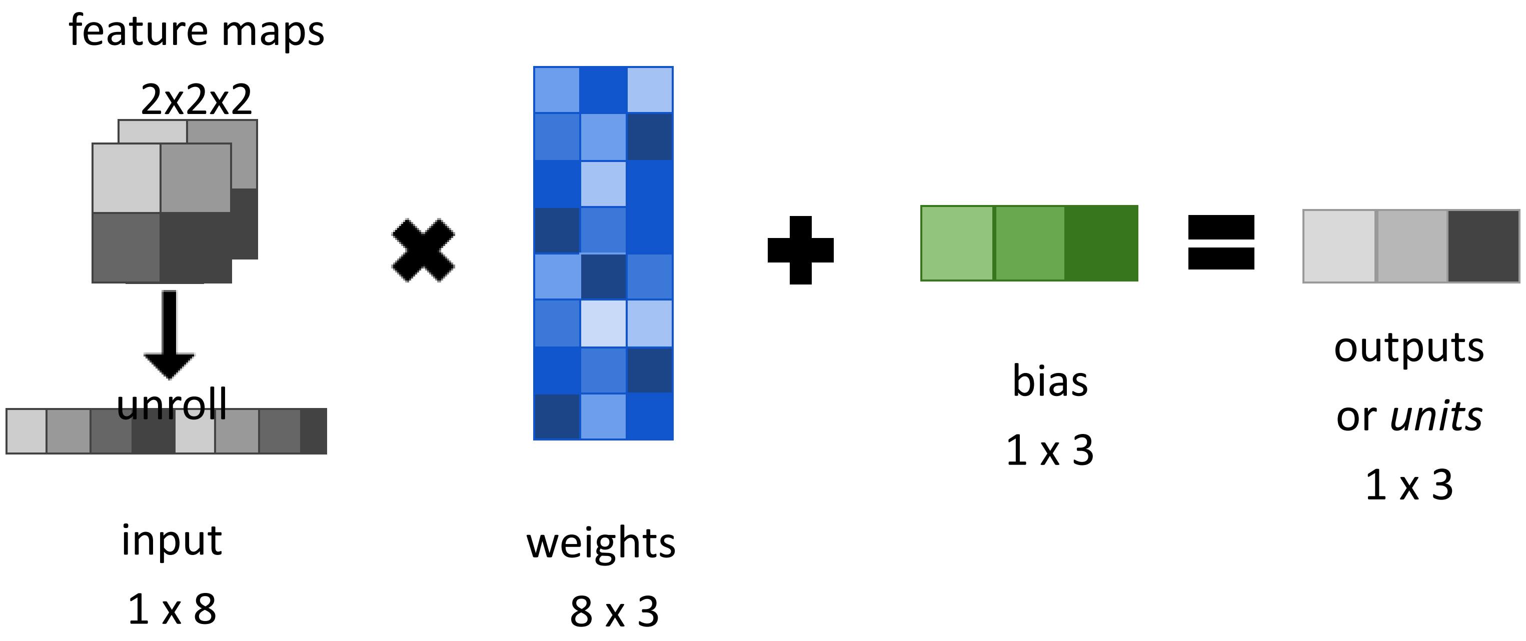


[figure credit A. Karpathy]

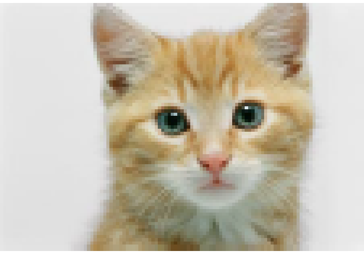
Fully Connected Layers (FC)

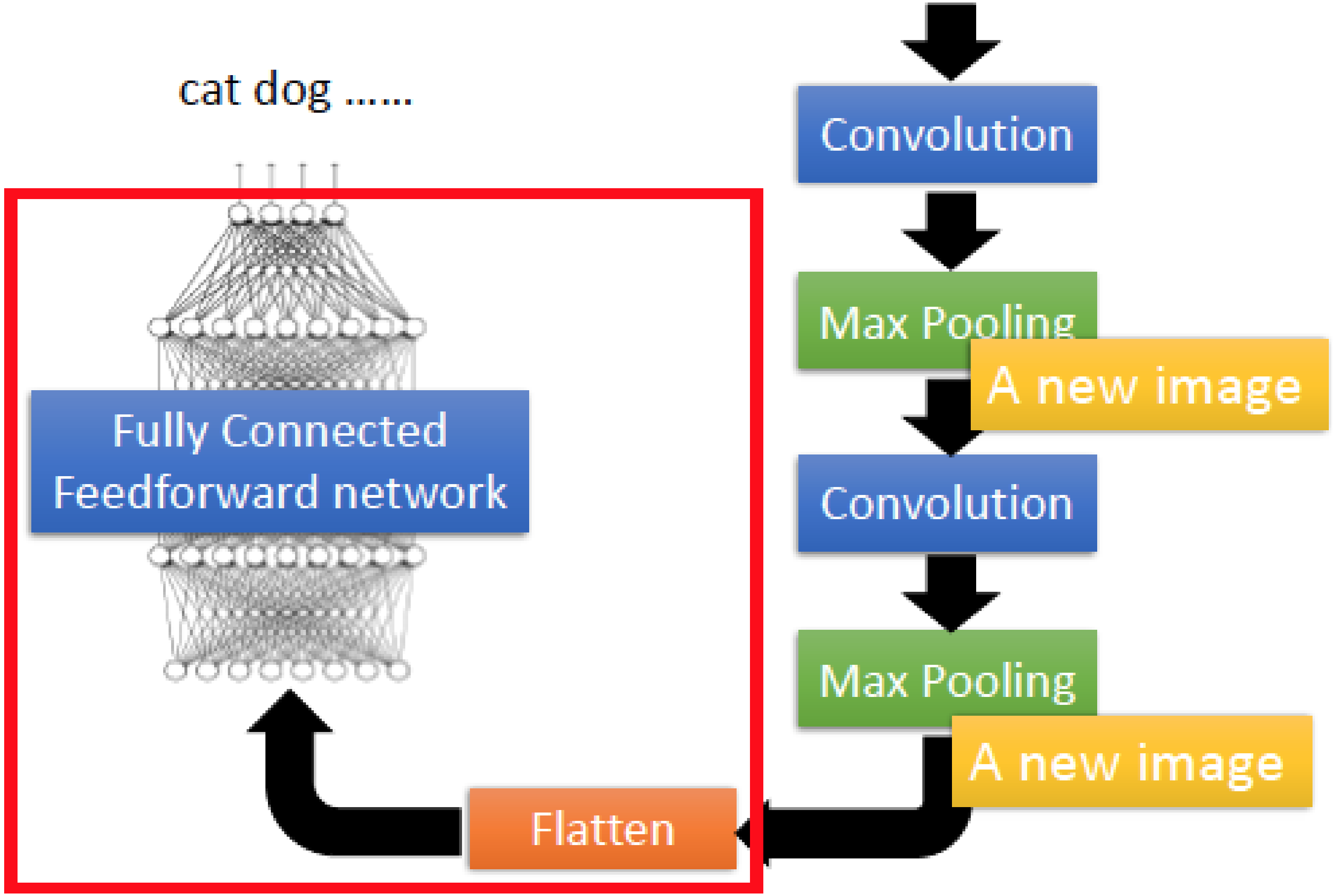
Learn a global feature from the full feature maps
Often found at the end of convnets

Note: unroll is also called **flatten**

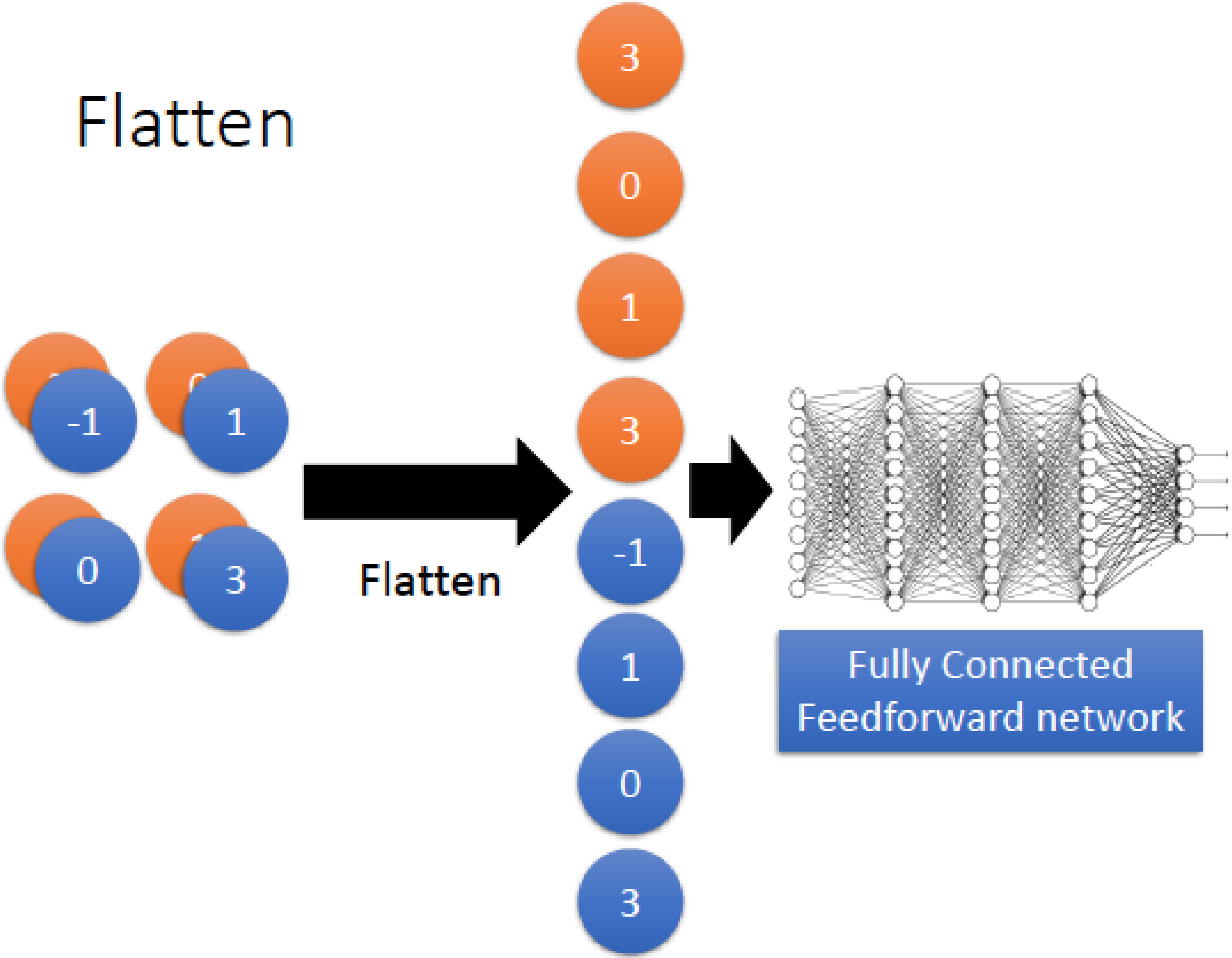


The whole CNN

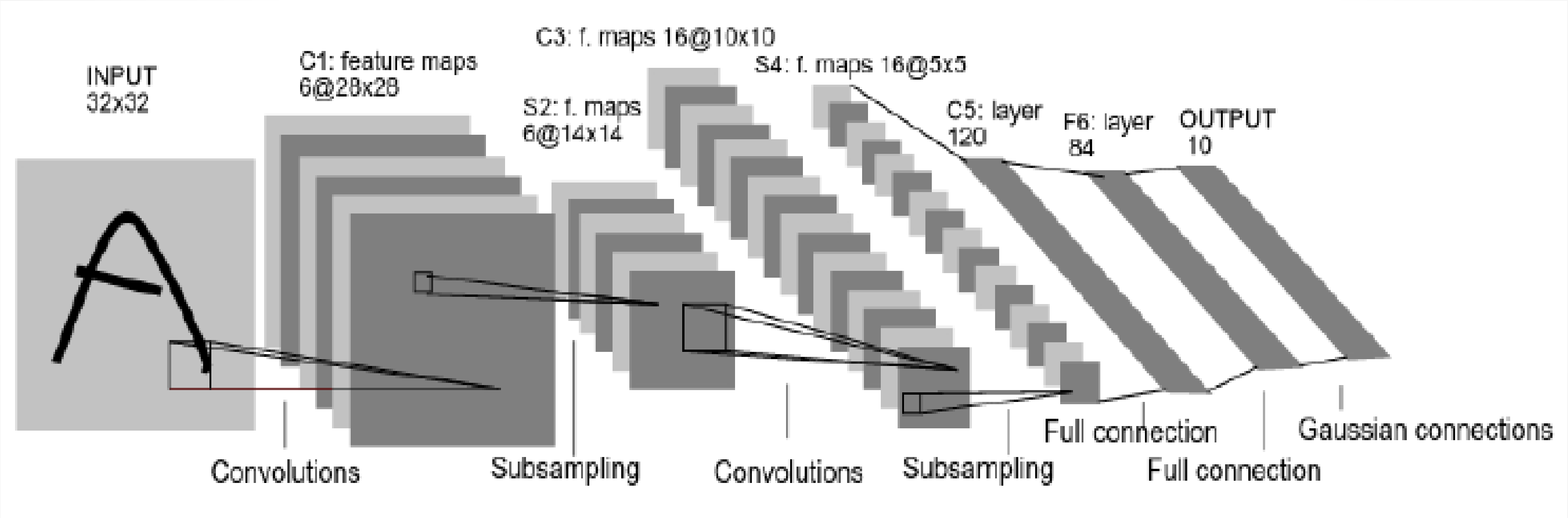




Flatten

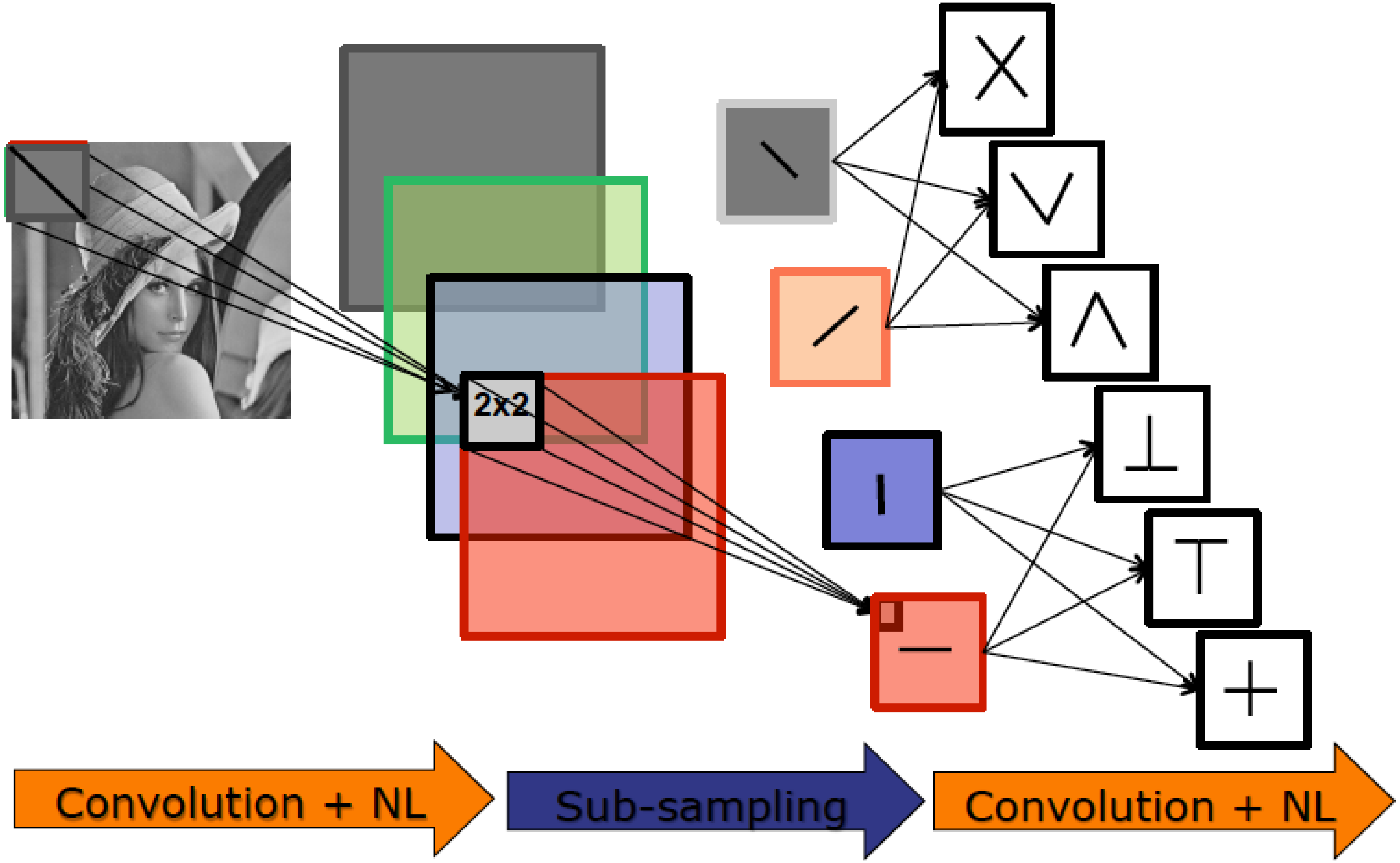

-1
1
0
3
Flatten
3
0
1
3
-1
1
0
3
Fully Connected
Feedforward network

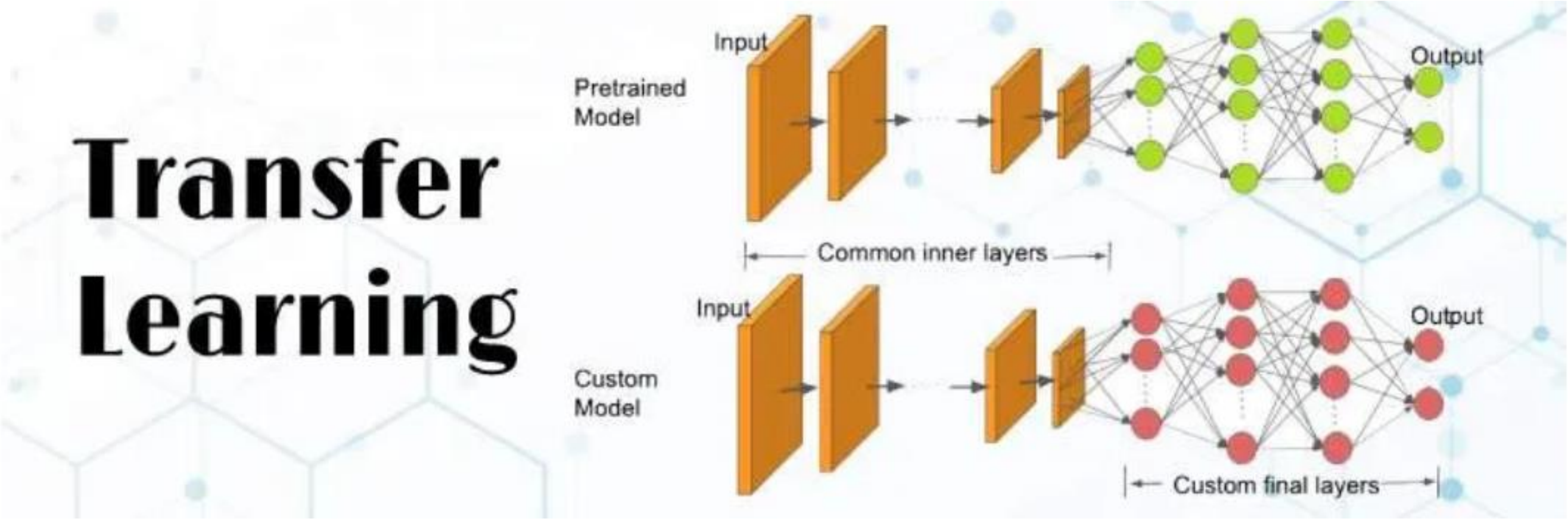
Putting It All Together!



Lenet-5 (Lecun-98), Convolutional Neural Network for digits recognition

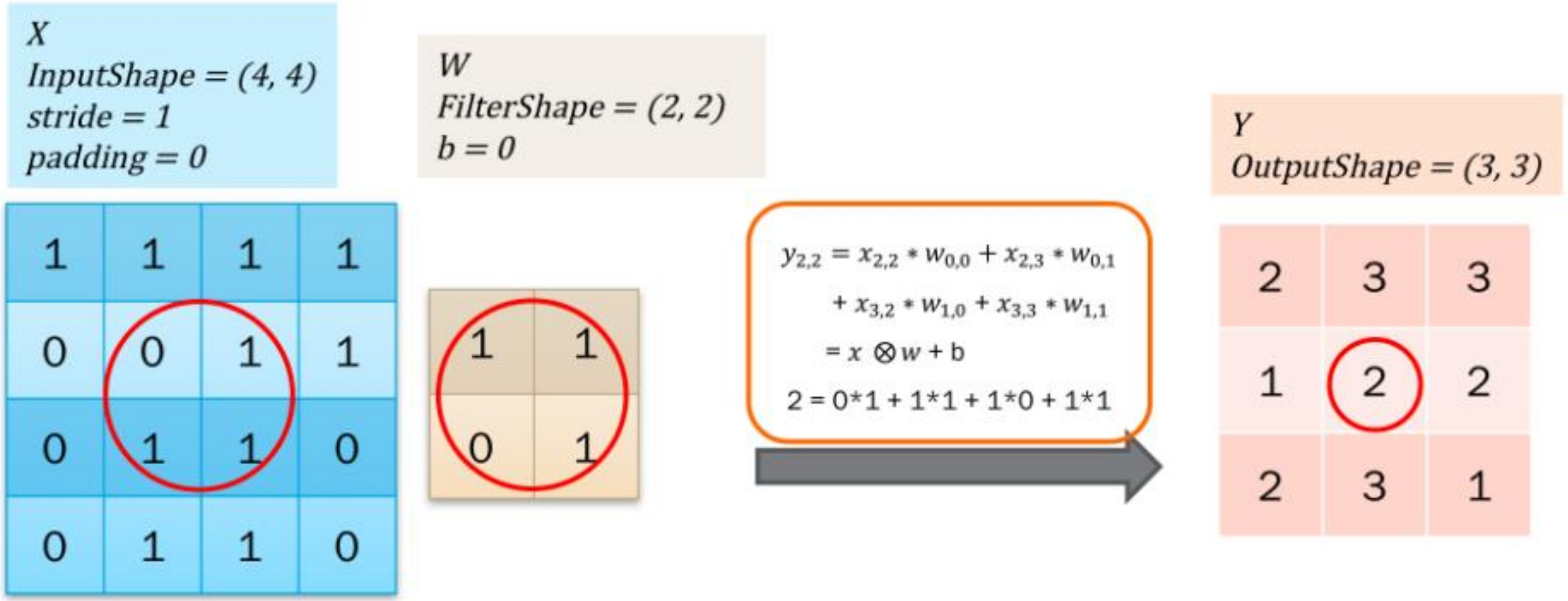
What is Convolutional NN ?




Transfer
Learning
Pretrained
Model
Input
Output
Common inner layers
Custom
Model
Input
Output
Custom final layers

Learning of CNN

Forward propagation of CNN



$$OutputShape = \frac{InputShape - FilterShape + 2 * pad}{stride} + 1$$

Let's use the following notations:

$$\partial h_{ij} \text{ represents } \frac{\partial L}{\partial h_{ij}}$$

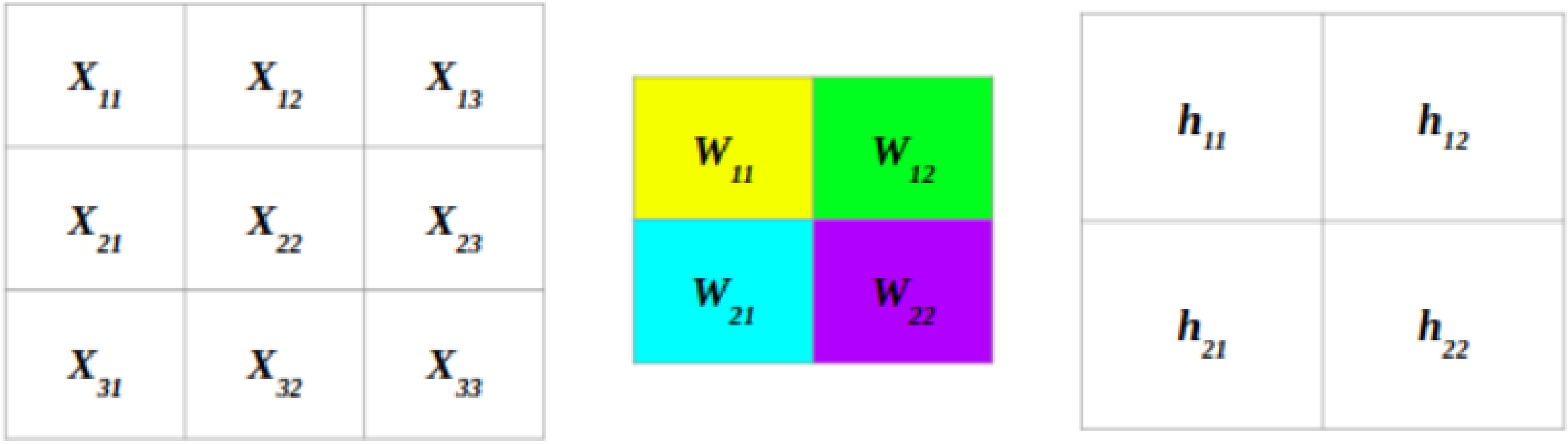
$$\partial w_{ij} \text{ represents } \frac{\partial L}{\partial w_{ij}}$$

Notations

Update on weight matrix of CNN

Note that there is only one way to reason the updates: Gradient descent, that is , mov
along the opposite direction of the gradient vector.

Consider the following simple filter and the gradient h_{ij}



Input Size : 3x3, **Filter Size** : 2x2, **Output Size** : 2x2

Relationship of input, filter and output; note that
Input and output could be just other hidden layers

Here, h_{11} is not the gradient, it is output!

$$h_{11} = W_{11}X_{11} + W_{12}X_{12} + W_{21}X_{21} + W_{22}X_{22}$$

$$h_{12} = W_{11}X_{12} + W_{12}X_{13} + W_{21}X_{22} + W_{22}X_{23}$$

$$h_{21} = W_{11}X_{21} + W_{12}X_{22} + W_{21}X_{31} + W_{22}X_{32}$$

$$h_{22} = W_{11}X_{22} + W_{12}X_{23} + W_{21}X_{32} + W_{22}X_{33}$$

Output Equations

Let’s use the following notations:

$$\partial h_{ij} \textit{ represents } \frac{\partial L}{\partial h_{ij}}$$

$$\partial w_{ij} \textit{ represents } \frac{\partial L}{\partial w_{ij}}$$

Notations

To find Δw_{ij} we need the following partial directive:

$$\frac{\partial L}{\partial w_{ij}} = \sum_{all\,related\,pk} \frac{\partial L}{\partial h_{pk}} \frac{\partial h_{pk}}{\partial w_{ij}}$$

So, we have:

$$\partial W_{11} = X_{11}\partial h_{11} + X_{12}\partial h_{12} + X_{21}\partial h_{21} + X_{22}\partial h_{22}$$

$$\partial W_{12} = X_{12}\partial h_{11} + X_{13}\partial h_{12} + X_{22}\partial h_{21} + X_{23}\partial h_{22}$$

$$\partial W_{21} = X_{21}\partial h_{11} + X_{22}\partial h_{12} + X_{31}\partial h_{21} + X_{32}\partial h_{22}$$

$$\partial W_{22} = X_{22}\partial h_{11} + X_{23}\partial h_{12} + X_{32}\partial h_{21} + X_{33}\partial h_{22}$$

Final derivatives after performing back propagation

X11	X12	X13
X21	X22	X23
X31	X32	X33

$\otimes$

∂h11	∂h12
∂h21	∂h22

$\Delta wij = -\eta Dw$

Should also consider learning rate!

X

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

W

1	1
0	1

Recall : $w_{0,0}$ => *the output shape in X*

Delta, d

0.2	0.1	0.1
0.1	0.2	0.2
0.2	0.1	0.2

$$\frac{\partial d}{\partial w_{0,0}} = d_{0,0} * x_{0,0} + d_{0,1} * x_{1,0} + \cdots + d_{3,3} * x_{3,3}$$

$$= x \otimes d = 0.4 + 0.2 + 0.3 = 0.9$$

$$\frac{\partial d}{\partial b} = sum(d)$$

Dw

0.9	1.1
0.8	0.8

Update on bias b of CNN

If bias is considered, we have:

$h_{11}=w_{11}X_{11}+w_{12}X_{12}+W_{21}X_{21}+W_{22}X_{22}+b$
$h_{12}=w_{11}X_{12}+w_{12}X_{13}+W_{21}X_{22}+W_{22}X_{23}+b$
$h_{21}=w_{11}X_{21}+w_{12}X_{22}+W_{21}X_{31}+W_{22}X_{32}+b$
$h_{22}=w_{11}X_{22}+w_{12}X_{23}+W_{21}X_{32}+W_{22}X_{33}+b$

$$\frac{\partial L}{\partial b}=\sum_{all\ h_{ij}}\frac{\partial L}{\partial h_{ij}}\frac{\partial h_{ij}}{\partial b}$$

Gradients of the previous layer

Need to change the subscript of y, d, w by adding 1

X

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

W

1	1
0	1

Recall

$x_{2,2} \Rightarrow y_{0,0}, y_{0,1}, y_{1,0}, y_{1,1}$

Delta, d

0.2	0.1	0.1
0.1	0.2	0.2
0.2	0.1	0.2

$$\frac{\partial d}{\partial x_{2,2}} = d_{0,0} * w_{1,1} + d_{0,1} * w_{1,0} + d_{1,0} * w_{0,1} + d_{1,1} * w_{0,0}$$

$$= d \otimes w^{rotate}$$

$$= 0.2*1 + 0.1*0 + 0.1*1 + 0.2*1 = 0.5$$

Rotate (180 degree): switch rows and reverse order

x11	**x12**	**x13**
x21	x22	x23
x31	x32	x33

w11	**w12**
w21	w22

h11	**h12**
h21	h22

∂h11	**∂h12**
∂h21	∂h22

w22	**w21**
w12	w11

$$x_{22} * w_{22} + \cdots = h_{11}$$
$$x_{22} * w_{21} + \cdots = h_{12}$$
$$x_{22} * w_{12} + \cdots = h_{21}$$
$$x_{22} * w_{11} + \cdots = h_{22}$$

$$\frac{\partial L}{\partial x_{22}} = \frac{\partial L}{\partial h_{11}} * \overset{\text{w22}}{\frac{\partial h_{11}}{\partial x_{22}}} + \frac{\partial L}{\partial h_{12}} * \overset{\text{w21}}{\frac{\partial h_{12}}{\partial x_{22}}} + \frac{\partial L}{\partial h_{21}} * \frac{\partial h_{21}}{\partial x_{22}} + \frac{\partial L}{\partial h_{22}} * \frac{\partial h_{22}}{\partial x_{22}}$$

dx

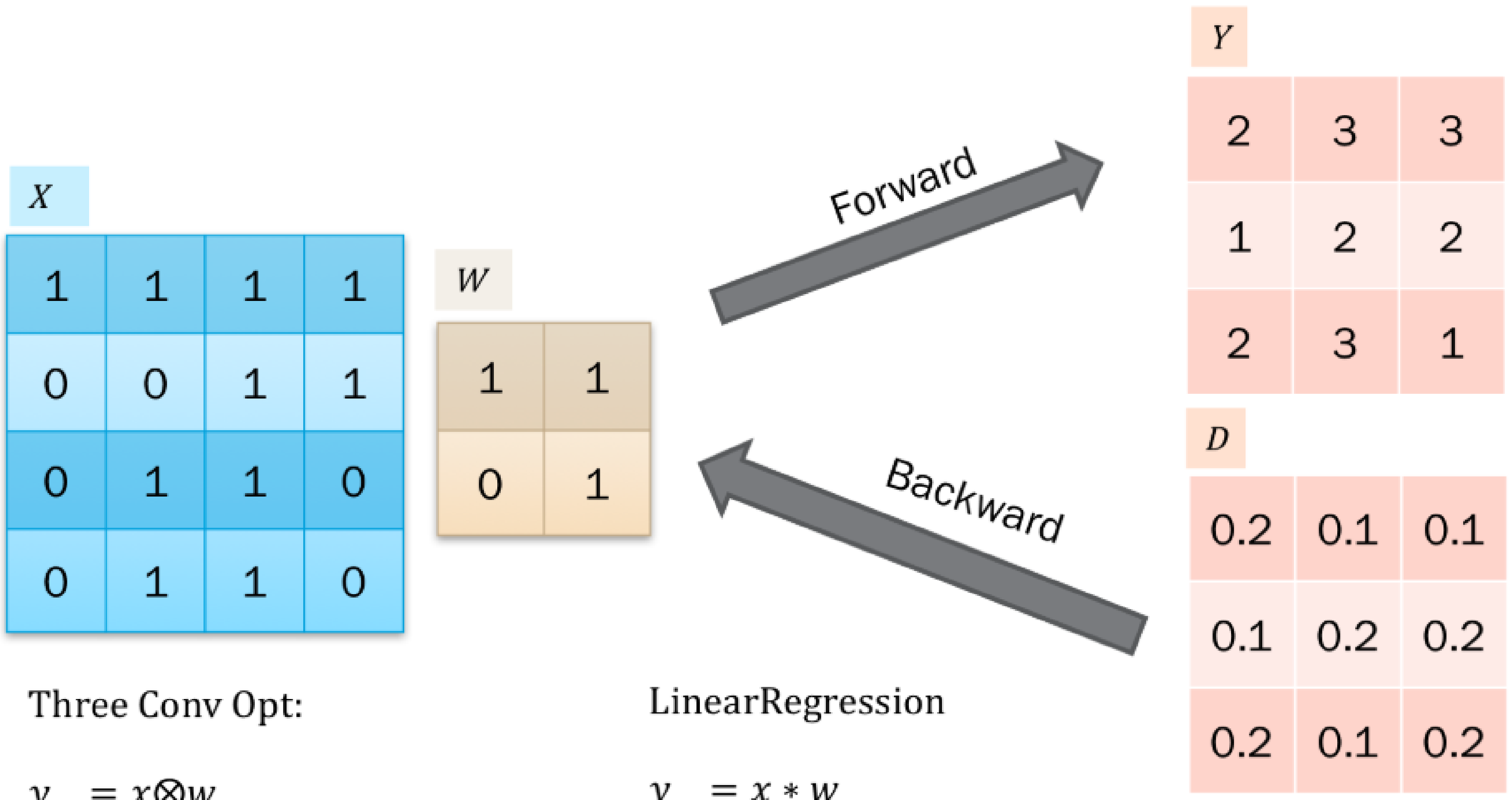
0.2	0.3	0.2	0.1
0.1	0.5	0.5	0.3
0.2	0.4	0.5	0.4
0.0	0.2	0.1	0.2

$NewFilter = w^{rotate}$

1	0
1	1

Delta, d

0	0	0	0	0
0	0.2	0.1	0.1	0
0	0.1	0.2	0.2	0
0	0.2	0.1	0.2	0
0	0	0	0	0



Three Conv Opt:

$$y \quad = x \otimes w$$

$$dw = x \otimes d$$

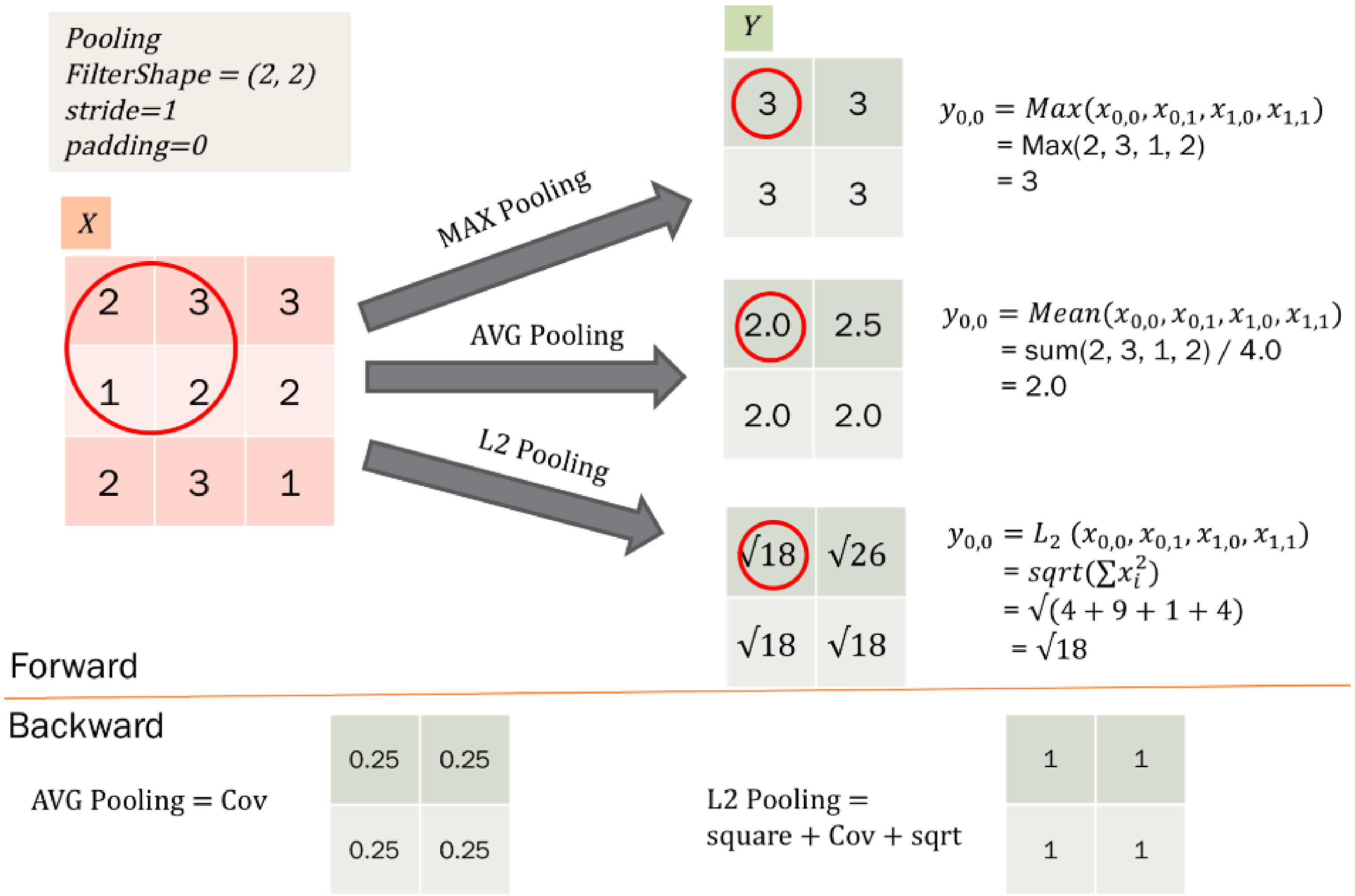
$$dx = d \otimes w^{rotate}$$

LinearRegression

$$y \quad = x * w$$

$$dw = x^T * d$$

$$dx = d * w^T$$


Pooling
FilterShape = (2, 2)
stride=1
padding=0
X
2 3 3
1 2 2
2 3 1
MAX Pooling
AVG Pooling
L2 Pooling
Y
3 3
3 3
$y_{0,0} = Max(x_{0,0}, x_{0,1}, x_{1,0}, x_{1,1})$
= Max(2, 3, 1, 2)
= 3
2.0 2.5
2.0 2.0
$y_{0,0} = Mean(x_{0,0}, x_{0,1}, x_{1,0}, x_{1,1})$
= sum(2, 3, 1, 2) / 4.0
= 2.0
√18 √26
√18 √18
$y_{0,0} = L_2\,(x_{0,0}, x_{0,1}, x_{1,0}, x_{1,1})$
$= sqrt(\sum x_i^2)$
$= \sqrt{(4+9+1+4)}$
$= \sqrt{18}$
Forward
Backward
AVG Pooling = Cov
0.25 0.25
0.25 0.25
L2 Pooling =
square + Cov + sqrt
1 1
1 1