Introduction to Deep Learning

Convolutional Neural Networks (1)

Prof. Songhwai Oh

ECE, SNU

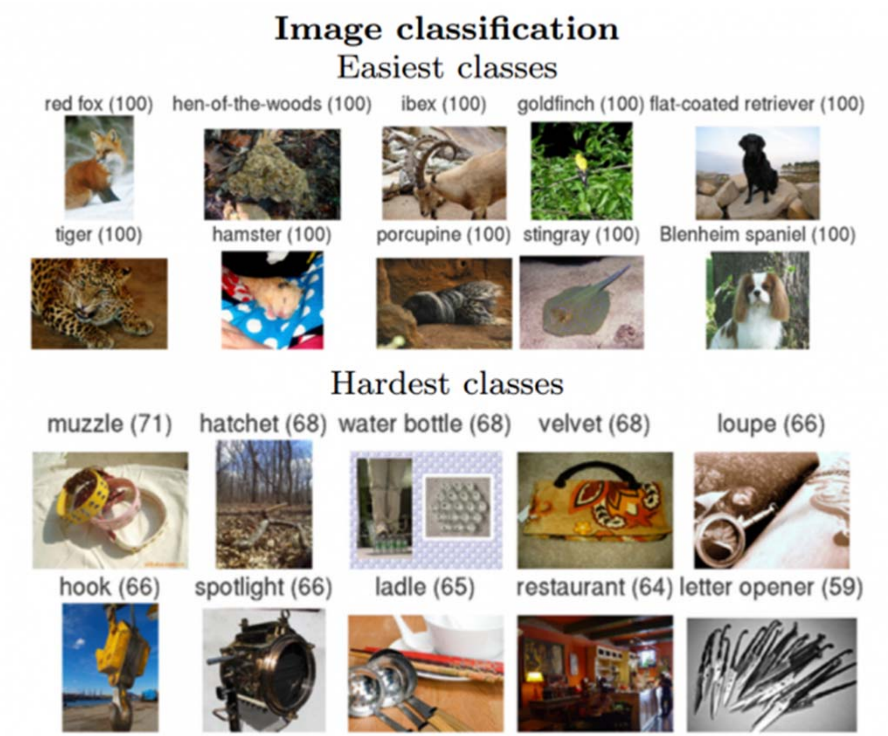
Krizhevsky, Alex, Ilya Sutskever, and Geoffrey E. Hinton. "Imagenet classification with deep convolutional neural networks." *NIPS*. 2012.

AlexNet

ImageNet Large-Scale Visual Recognition Challenge, 2012

Tasks:

- Decide whether a given image contains a particular type of object or not. For example, a contestant might decide that there are cars in this image but no tigers.
- Find a particular object and draw a box around it. For example, a contestant might decide that there is a screwdriver at a certain position with a width of 50 pixels and a height of 30 pixels.



- 1000 different categories
- Over 1 million images
- Training set: 456,567 images

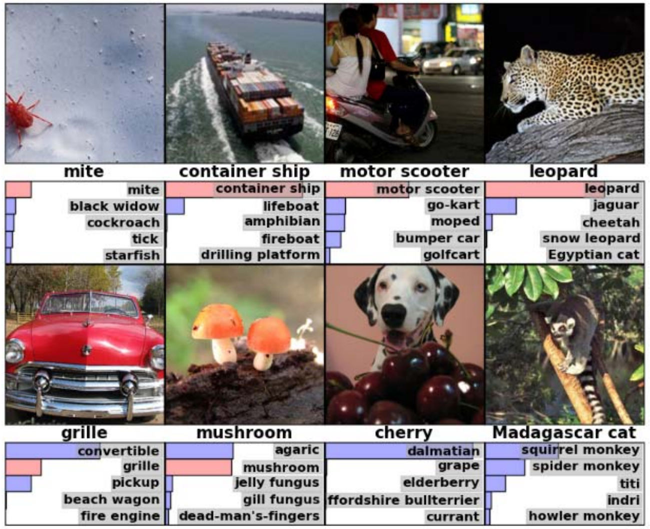
Year	Winning Error Rate
2010	28.2%
2011	25.8%
2012	16.4% (2nd 25.2%)
2013	11.2%
2014	6.7%
2015	3.57%
Human	**About 5.1%**

ImageNet Large Scale Visual Recognition Challenge. Russakovsky et al. *arXiv preprint arXiv:1409.0575*. URL: http://arxiv.org/abs/1409.0575v1

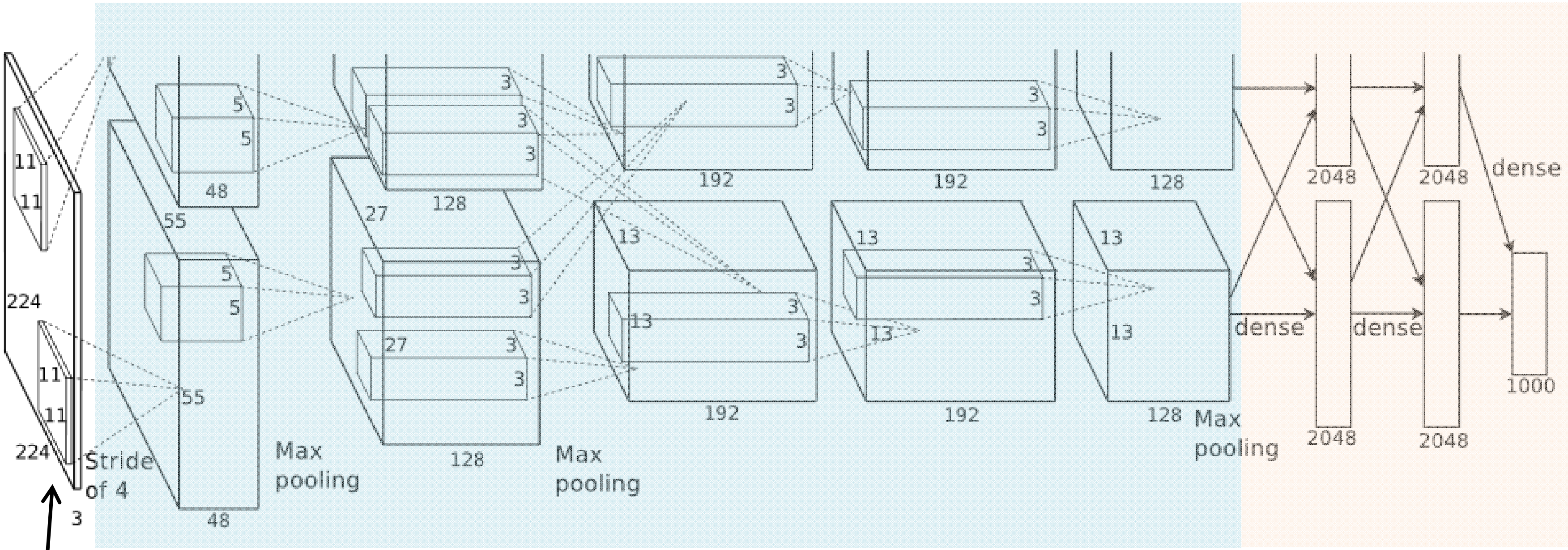
ImageNet Dataset

Source: https://cs.stanford.edu/people/karpathy/cnnembed/cnn_embed_full_1k.jpg

AlexNet on ImageNet



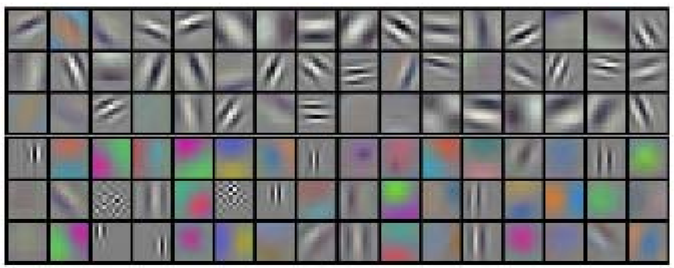
Architecture



5 convolutional layers

3 fully connected layers

Learned 11x11x3 filters



Key ideas:

- Rectified Linear Unit (ReLU): an activation function
- GPU implementation (2 GPUs)
- Local response normalization, Overlapping pooling
- Data augmentation, Dropout

CONVOLUTION

Convolution

- Continuous convolution

$$(f * g)(t) = \int f(\tau)g(t-\tau)d\tau = \int f(t-\tau)g(t)d\tau$$

- Discrete convolution

$$(f * g)(t) = \sum_{i=-\infty}^{\infty} f(i)g(t-i) = \sum_{i=-\infty}^{\infty} f(t-i)g(i)$$

- 2D image I (K is a 2D filter)

$$(I*K)(i,j) = \sum_m \sum_n I(m,n)K(i-m,j-n) = \sum_m \sum_n I(i-m,i-n)K(m,n)$$

2D Convolution

K (3x3 filter)

K_{11}	K_{12}	K_{13}
K_{21}	K_{22}	K_{23}
K_{31}	K_{32}	K_{33}

*

I (7x7 image)

I_{11}	I_{12}	I_{13}	I_{14}	I_{15}	I_{16}	I_{17}
I_{21}	I_{22}	I_{23}	I_{24}	I_{25}	I_{26}	I_{27}
I_{31}	I_{32}	I_{33}	I_{34}	I_{35}	I_{36}	I_{37}
I_{41}	I_{42}	I_{43}	I_{44}	I_{45}	I_{46}	I_{47}
I_{51}	I_{52}	I_{53}	I_{54}	I_{55}	I_{56}	I_{57}
I_{61}	I_{62}	I_{63}	I_{64}	I_{65}	I_{66}	I_{67}
I_{71}	I_{72}	I_{73}	I_{74}	I_{75}	I_{76}	I_{77}

=

Output (5x5)

O_{11}	O_{12}	O_{13}	O_{14}	O_{15}
O_{21}	O_{22}	O_{23}	O_{24}	O_{25}
O_{31}	O_{32}	O_{33}	O_{34}	O_{35}
O_{41}	O_{42}	O_{43}	O_{44}	O_{45}
O_{51}	O_{52}	O_{53}	O_{54}	O_{55}

2D Convolution

K (3x3 filter)

K_{11}	K_{12}	K_{13}
K_{21}	K_{22}	K_{23}
K_{31}	K_{32}	K_{33}

*

I (7x7 image)

I_{11}	I_{12}	I_{13}	I_{14}	I_{15}	I_{16}	I_{17}
I_{21}	I_{22}	I_{23}	I_{24}	I_{25}	I_{26}	I_{27}
I_{31}	I_{32}	I_{33}	I_{34}	I_{35}	I_{36}	I_{37}
I_{41}	I_{42}	I_{43}	I_{44}	I_{45}	I_{46}	I_{47}
I_{51}	I_{52}	I_{53}	I_{54}	I_{55}	I_{56}	I_{57}
I_{61}	I_{62}	I_{63}	I_{64}	I_{65}	I_{66}	I_{67}
I_{71}	I_{72}	I_{73}	I_{74}	I_{75}	I_{76}	I_{77}

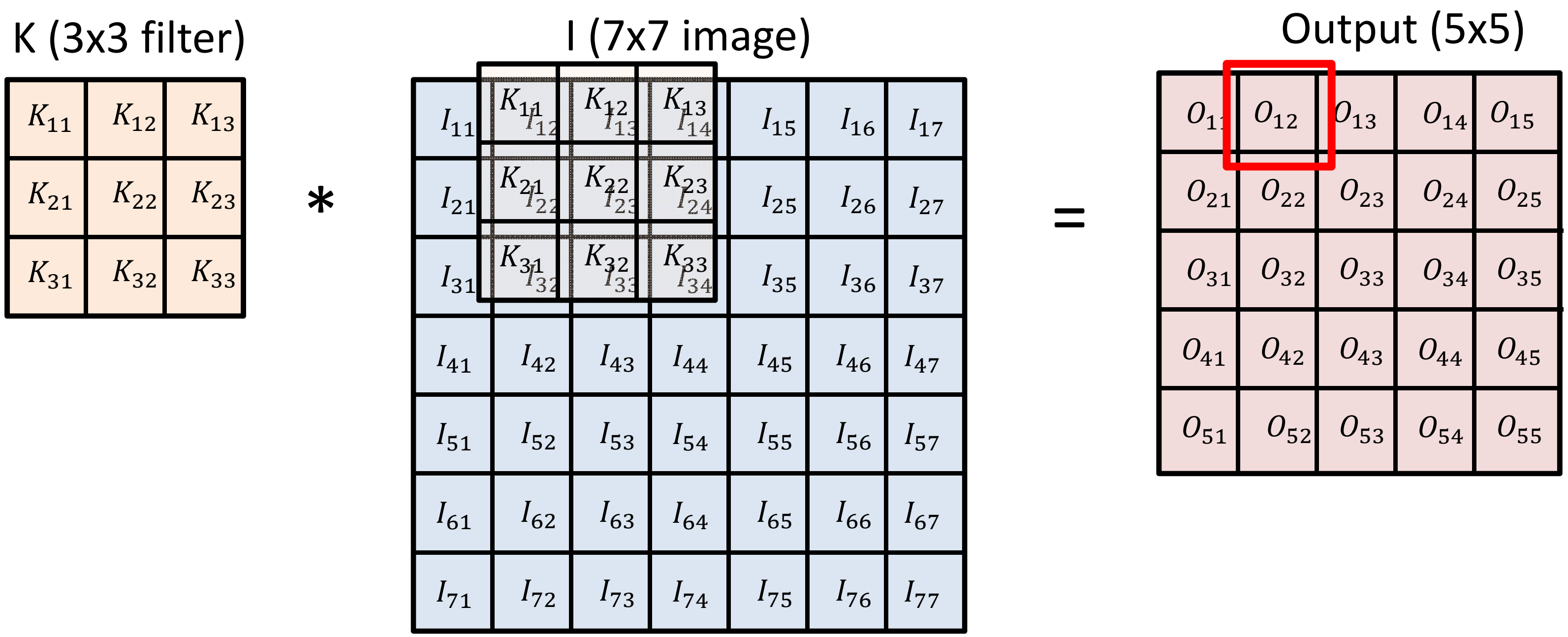
=

Output (5x5)

O_{11}	O_{12}	O_{13}	O_{14}	O_{15}
O_{21}	O_{22}	O_{23}	O_{24}	O_{25}
O_{31}	O_{32}	O_{33}	O_{34}	O_{35}
O_{41}	O_{42}	O_{43}	O_{44}	O_{45}
O_{51}	O_{52}	O_{53}	O_{54}	O_{55}

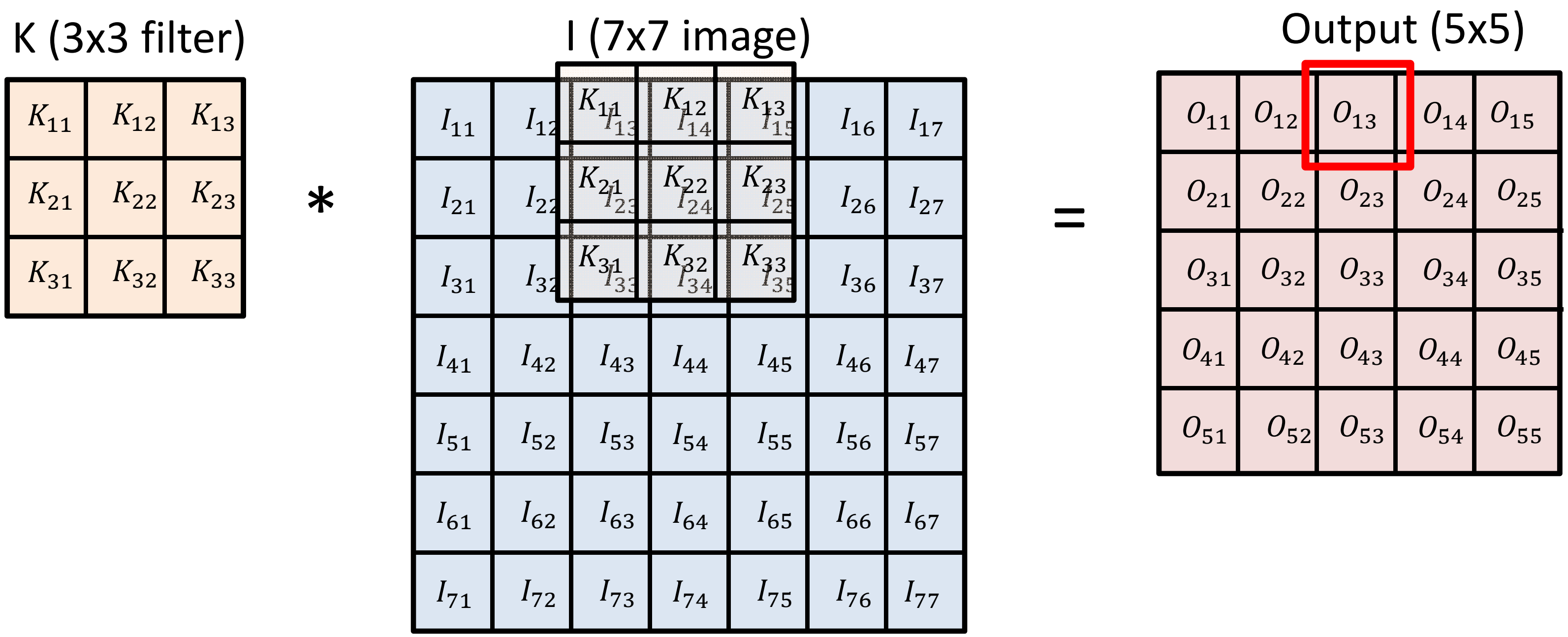
$$O_{11} = I_{11}K_{11}+I_{12}K_{12}+I_{13}K_{13}+I_{21}K_{21}+I_{22}K_{22}+I_{23}K_{23}+I_{31}K_{31}+I_{32}K_{32}+I_{33}K_{33}+bias$$

2D Convolution



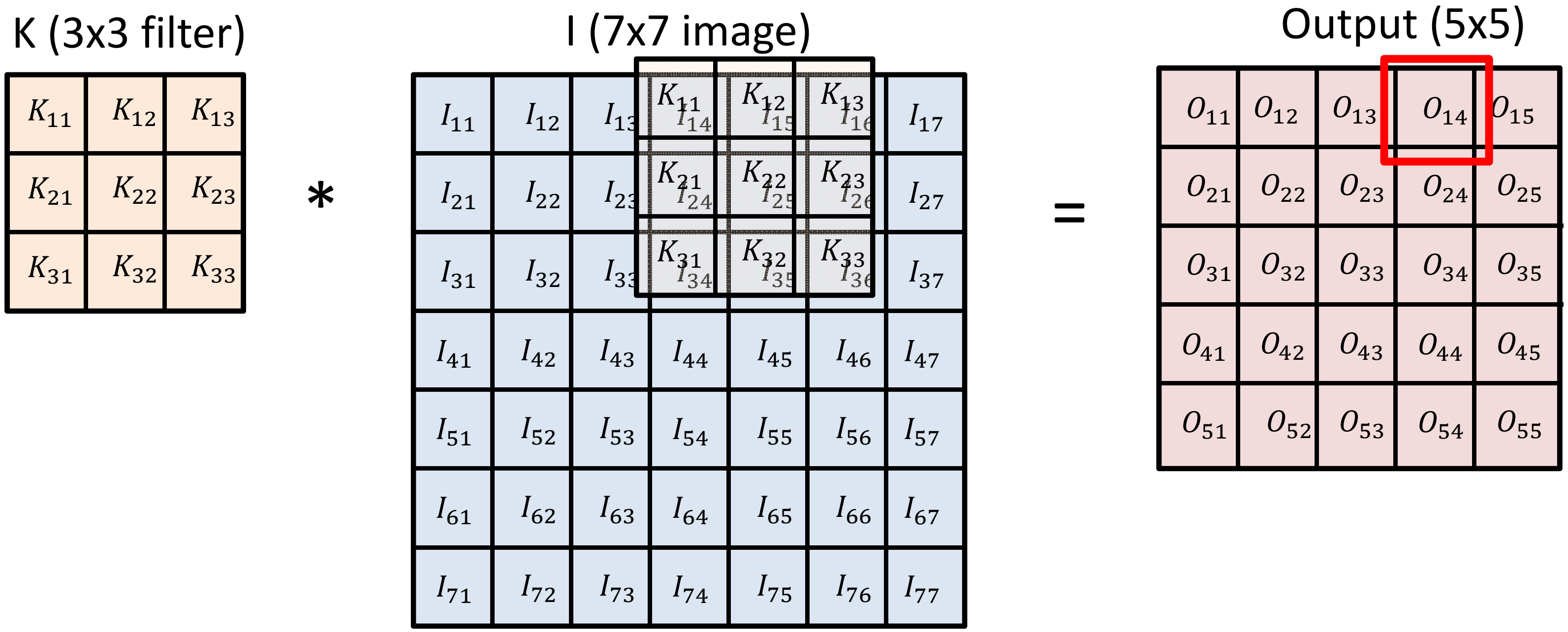
$$O_{11} = I_{11}K_{11} + I_{12}K_{12} + I_{13}K_{13} + I_{21}K_{21} + I_{22}K_{22} + I_{23}K_{23} + I_{31}K_{31} + I_{32}K_{32} + I_{33}K_{33} + bias$$
$$O_{12} = I_{12}K_{11} + I_{13}K_{12} + I_{14}K_{13} + I_{22}K_{21} + I_{23}K_{22} + I_{24}K_{23} + I_{32}K_{31} + I_{33}K_{32} + I_{34}K_{33} + bias$$

2D Convolution



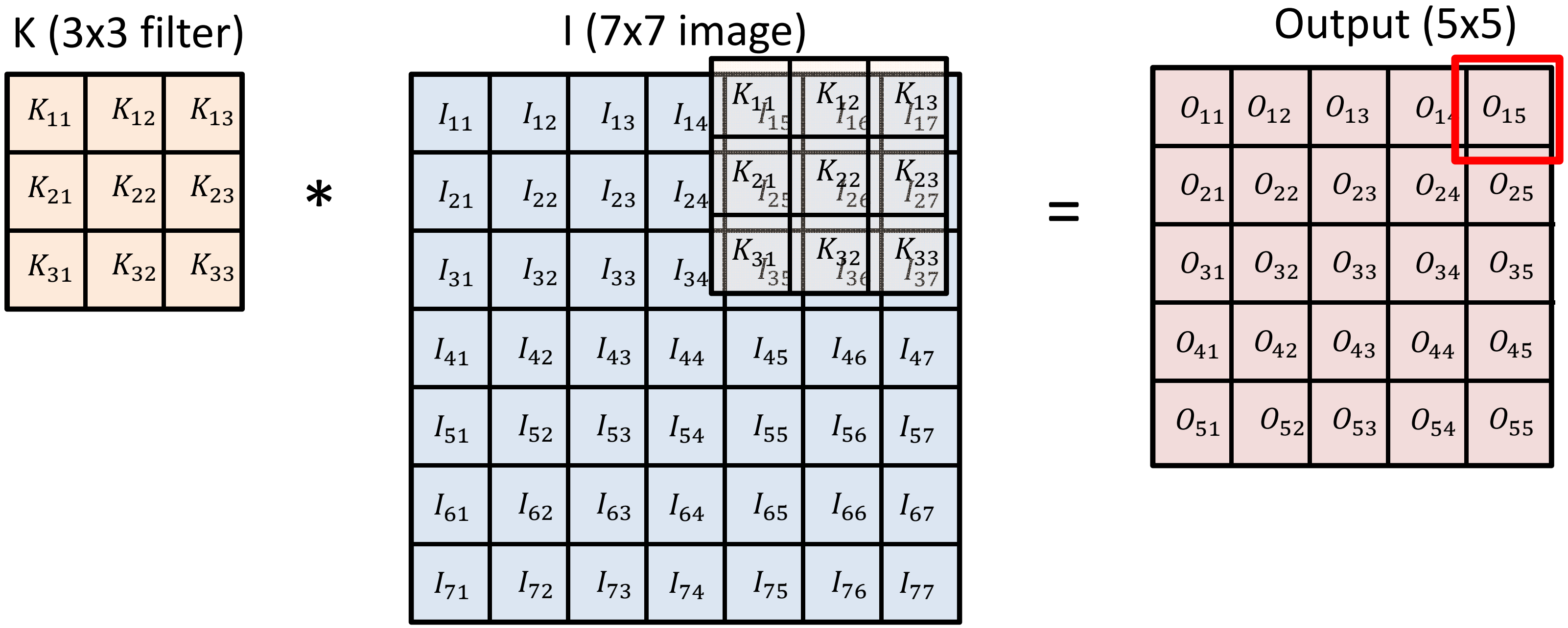
$$O_{11} = I_{11}K_{11}+I_{12}K_{12}+I_{13}K_{13}+I_{21}K_{21}+I_{22}K_{22}+I_{23}K_{23}+I_{31}K_{31}+I_{32}K_{32}+I_{33}K_{33}+bias$$
$$O_{12} = I_{12}K_{11}+I_{13}K_{12}+I_{14}K_{13}+I_{22}K_{21}+I_{23}K_{22}+I_{24}K_{23}+I_{32}K_{31}+I_{33}K_{32}+I_{34}K_{33}+bias$$
$$O_{13} = I_{13}K_{11}+I_{14}K_{12}+I_{15}K_{13}+I_{23}K_{21}+I_{24}K_{22}+I_{25}K_{23}+I_{33}K_{31}+I_{34}K_{32}+I_{35}K_{33}+bias$$

2D Convolution



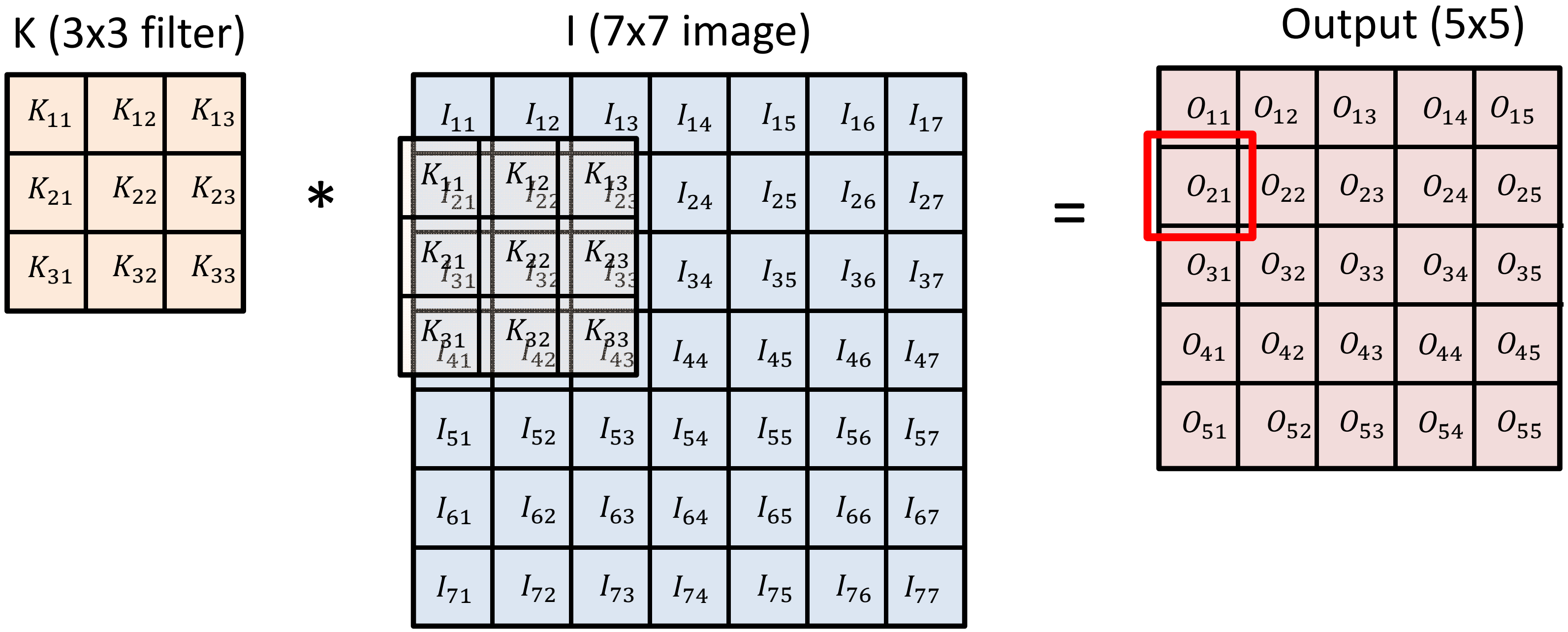
$$O_{11} = I_{11}K_{11}+I_{12}K_{12}+I_{13}K_{13}+I_{21}K_{21}+I_{22}K_{22}+I_{23}K_{23}+I_{31}K_{31}+I_{32}K_{32}+I_{33}K_{33}+bias$$
$$O_{12} = I_{12}K_{11}+I_{13}K_{12}+I_{14}K_{13}+I_{22}K_{21}+I_{23}K_{22}+I_{24}K_{23}+I_{32}K_{31}+I_{33}K_{32}+I_{34}K_{33}+bias$$
$$O_{13} = I_{13}K_{11}+I_{14}K_{12}+I_{15}K_{13}+I_{23}K_{21}+I_{24}K_{22}+I_{25}K_{23}+I_{33}K_{31}+I_{34}K_{32}+I_{35}K_{33}+bias$$
$$O_{14} = I_{14}K_{11}+I_{15}K_{12}+I_{16}K_{13}+I_{24}K_{21}+I_{25}K_{22}+I_{26}K_{23}+I_{34}K_{31}+I_{35}K_{32}+I_{36}K_{33}+bias$$

2D Convolution



$$O_{11} = I_{11}K_{11}+I_{12}K_{12}+I_{13}K_{13}+I_{21}K_{21}+I_{22}K_{22}+I_{23}K_{23}+I_{31}K_{31}+I_{32}K_{32}+I_{33}K_{33}+bias$$
$$O_{12} = I_{12}K_{11}+I_{13}K_{12}+I_{14}K_{13}+I_{22}K_{21}+I_{23}K_{22}+I_{24}K_{23}+I_{32}K_{31}+I_{33}K_{32}+I_{34}K_{33}+bias$$
$$O_{13} = I_{13}K_{11}+I_{14}K_{12}+I_{15}K_{13}+I_{23}K_{21}+I_{24}K_{22}+I_{25}K_{23}+I_{33}K_{31}+I_{34}K_{32}+I_{35}K_{33}+bias$$
$$O_{14} = I_{14}K_{11}+I_{15}K_{12}+I_{16}K_{13}+I_{24}K_{21}+I_{25}K_{22}+I_{26}K_{23}+I_{34}K_{31}+I_{35}K_{32}+I_{36}K_{33}+bias$$
$$O_{15} = I_{15}K_{11}+I_{16}K_{12}+I_{17}K_{13}+I_{25}K_{21}+I_{26}K_{22}+I_{27}K_{23}+I_{35}K_{31}+I_{36}K_{32}+I_{37}K_{33}+bias$$

2D Convolution



$$O_{11} = I_{11}K_{11} + I_{12}K_{12} + I_{13}K_{13} + I_{21}K_{21} + I_{22}K_{22} + I_{23}K_{23} + I_{31}K_{31} + I_{32}K_{32} + I_{33}K_{33} + bias$$
$$O_{12} = I_{12}K_{11} + I_{13}K_{12} + I_{14}K_{13} + I_{22}K_{21} + I_{23}K_{22} + I_{24}K_{23} + I_{32}K_{31} + I_{33}K_{32} + I_{34}K_{33} + bias$$
$$O_{13} = I_{13}K_{11} + I_{14}K_{12} + I_{15}K_{13} + I_{23}K_{21} + I_{24}K_{22} + I_{25}K_{23} + I_{33}K_{31} + I_{34}K_{32} + I_{35}K_{33} + bias$$
$$O_{14} = I_{14}K_{11} + I_{15}K_{12} + I_{16}K_{13} + I_{24}K_{21} + I_{25}K_{22} + I_{26}K_{23} + I_{34}K_{31} + I_{35}K_{32} + I_{36}K_{33} + bias$$
$$O_{15} = I_{15}K_{11} + I_{16}K_{12} + I_{17}K_{13} + I_{25}K_{21} + I_{26}K_{22} + I_{27}K_{23} + I_{35}K_{31} + I_{36}K_{32} + I_{37}K_{33} + bias$$

……

2D Convolution

K (3x3 filter)

K_{11}	K_{12}	K_{13}
K_{21}	K_{22}	K_{23}
K_{31}	K_{32}	K_{33}

*

I (7x7 image)

I_{11}	I_{12}	I_{13}	I_{14}	I_{15}	I_{16}	I_{17}
I_{21}	I_{22}	I_{23}	I_{24}	I_{25}	I_{26}	I_{27}
I_{31}	I_{32}	I_{33}	I_{34}	I_{35}	I_{36}	I_{37}
I_{41}	I_{42}	I_{43}	I_{44}	I_{45}	I_{46}	I_{47}
I_{51}	I_{52}	I_{53}	I_{54}	I_{55}	I_{56}	I_{57}
I_{61}	I_{62}	I_{63}	I_{64}	I_{65}	I_{66}	I_{67}
I_{71}	I_{72}	I_{73}	I_{74}	I_{75}	I_{76}	I_{77}

=

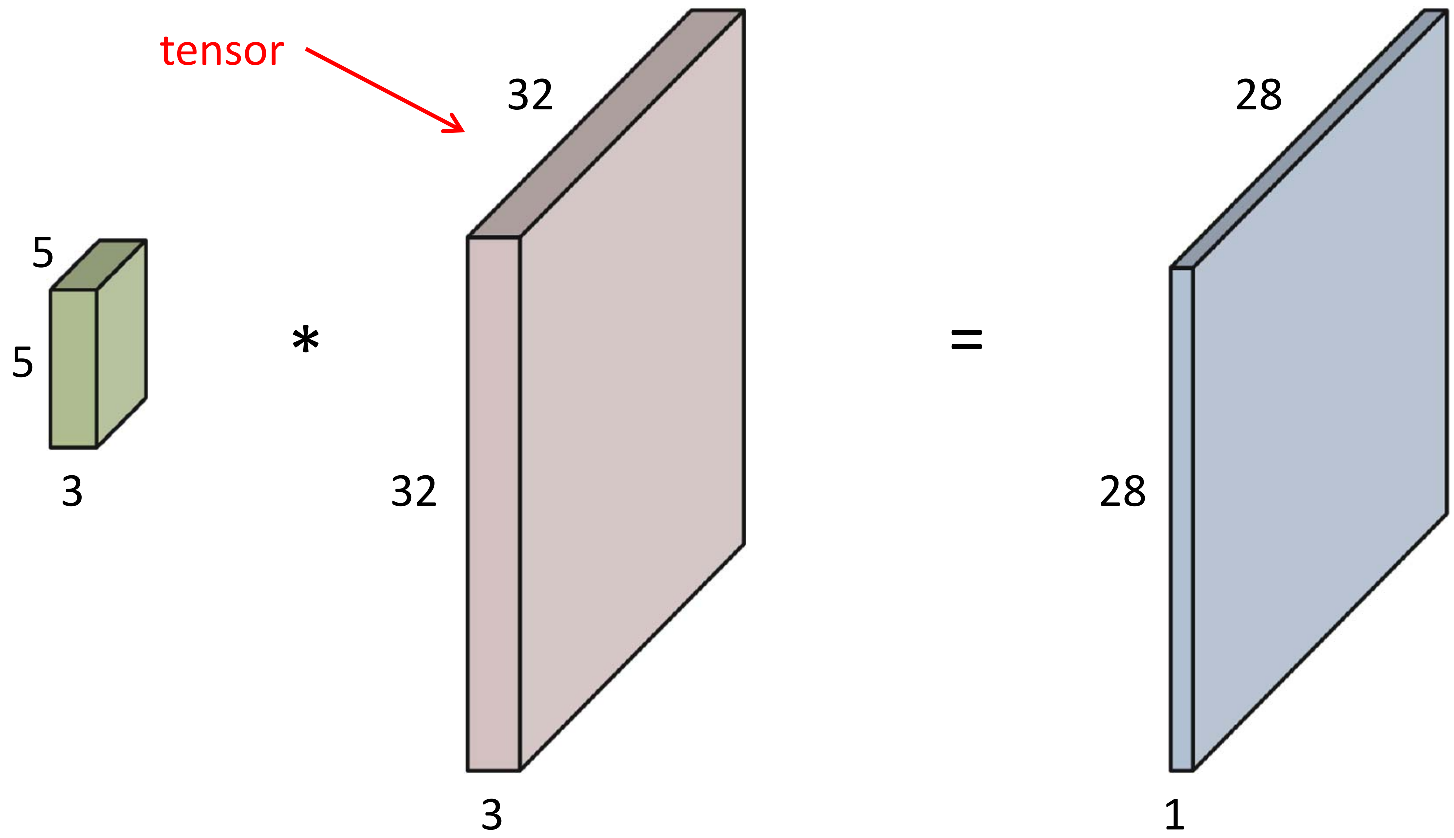
Output (5x5)

O_{11}	O_{12}	O_{13}	O_{14}	O_{15}
O_{21}	O_{22}	O_{23}	O_{24}	O_{25}
O_{31}	O_{32}	O_{33}	O_{34}	O_{35}
O_{41}	O_{42}	O_{43}	O_{44}	O_{45}
O_{51}	O_{52}	O_{53}	O_{54}	O_{55}

$$O_{11} = I_{11}K_{11}+I_{12}K_{12}+I_{13}K_{13}+I_{21}K_{21}+I_{22}K_{22}+I_{23}K_{23}+I_{31}K_{31}+I_{32}K_{32}+I_{33}K_{33}+bias$$
$$O_{12} = I_{12}K_{11}+I_{13}K_{12}+I_{14}K_{13}+I_{22}K_{21}+I_{23}K_{22}+I_{24}K_{23}+I_{32}K_{31}+I_{33}K_{32}+I_{34}K_{33}+bias$$
$$O_{13} = I_{13}K_{11}+I_{14}K_{12}+I_{15}K_{13}+I_{23}K_{21}+I_{24}K_{22}+I_{25}K_{23}+I_{33}K_{31}+I_{34}K_{32}+I_{35}K_{33}+bias$$
$$O_{14} = I_{14}K_{11}+I_{15}K_{12}+I_{16}K_{13}+I_{24}K_{21}+I_{25}K_{22}+I_{26}K_{23}+I_{34}K_{31}+I_{35}K_{32}+I_{36}K_{33}+bias$$
$$O_{15} = I_{15}K_{11}+I_{16}K_{12}+I_{17}K_{13}+I_{25}K_{21}+I_{26}K_{22}+I_{27}K_{23}+I_{35}K_{31}+I_{36}K_{32}+I_{37}K_{33}+bias$$

……

RGB Image Convolution

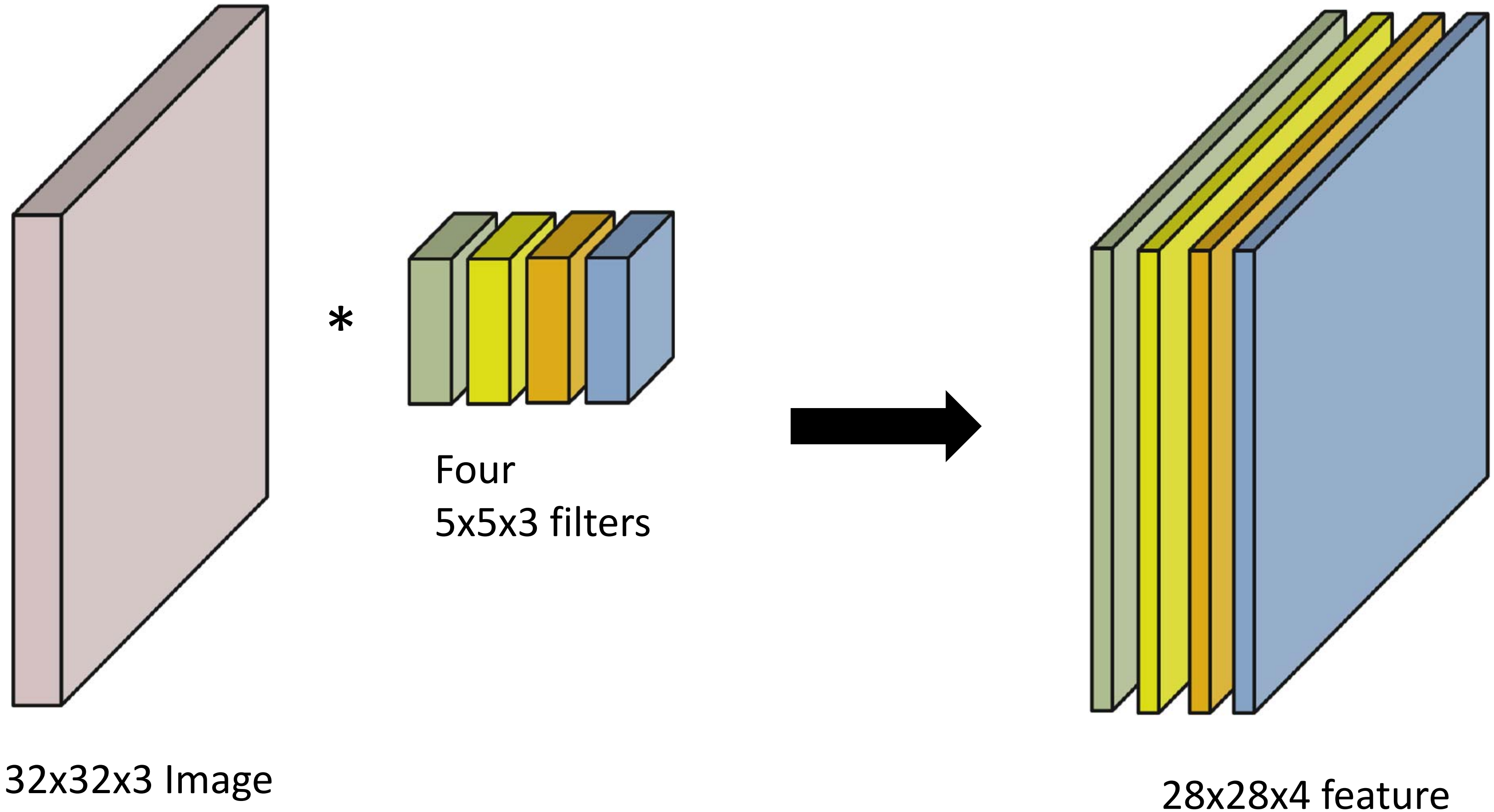


5x5x3 filter
w

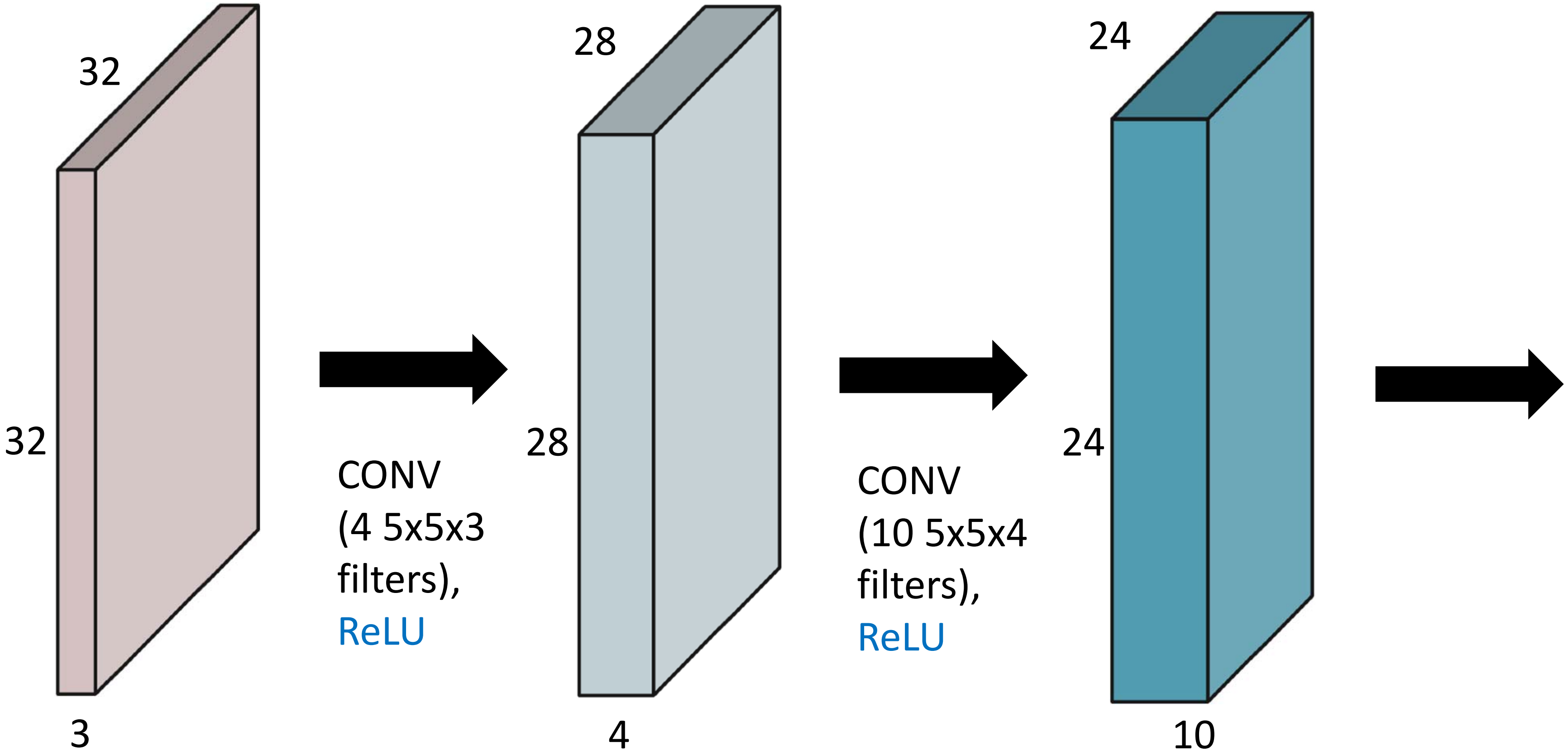
32x32x3 Image
x

28x28x1 feature
$w^T x + bias$

RGB Image Convolution



Convolutional Neural Network



ReLU: Rectified Linear Unit

Stride

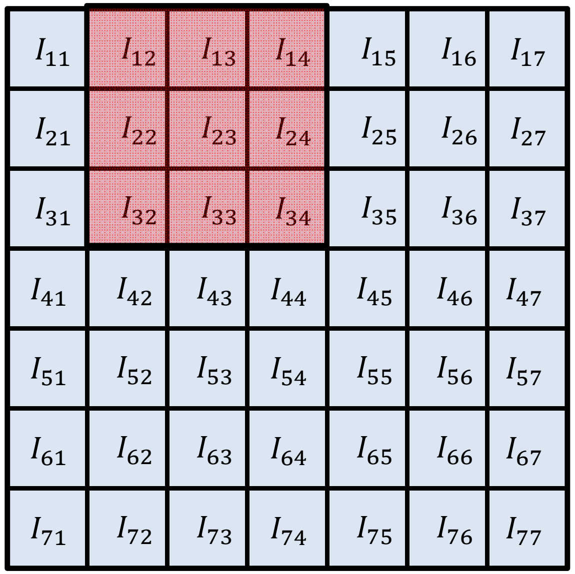
stride 1

I_{11}	I_{12}	I_{13}	I_{14}	I_{15}	I_{16}	I_{17}
I_{21}	I_{22}	I_{23}	I_{24}	I_{25}	I_{26}	I_{27}
I_{31}	I_{32}	I_{33}	I_{34}	I_{35}	I_{36}	I_{37}
I_{41}	I_{42}	I_{43}	I_{44}	I_{45}	I_{46}	I_{47}
I_{51}	I_{52}	I_{53}	I_{54}	I_{55}	I_{56}	I_{57}
I_{61}	I_{62}	I_{63}	I_{64}	I_{65}	I_{66}	I_{67}
I_{71}	I_{72}	I_{73}	I_{74}	I_{75}	I_{76}	I_{77}

3x3 filter

Stride

stride 1



3x3 filter

Stride

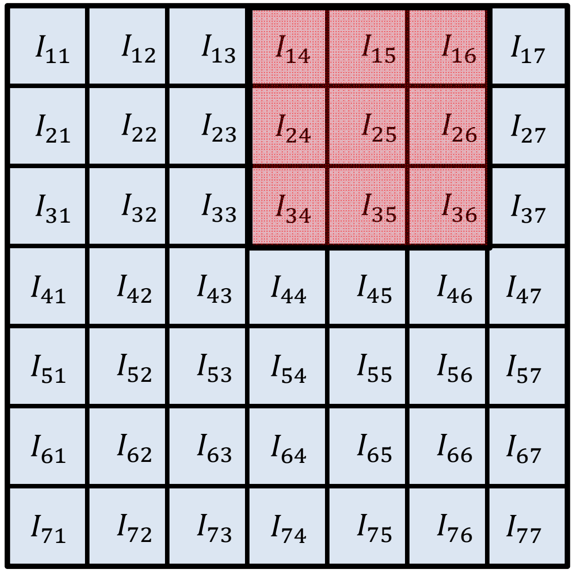
stride 1

I_{11}	I_{12}	I_{13}	I_{14}	I_{15}	I_{16}	I_{17}
I_{21}	I_{22}	I_{23}	I_{24}	I_{25}	I_{26}	I_{27}
I_{31}	I_{32}	I_{33}	I_{34}	I_{35}	I_{36}	I_{37}
I_{41}	I_{42}	I_{43}	I_{44}	I_{45}	I_{46}	I_{47}
I_{51}	I_{52}	I_{53}	I_{54}	I_{55}	I_{56}	I_{57}
I_{61}	I_{62}	I_{63}	I_{64}	I_{65}	I_{66}	I_{67}
I_{71}	I_{72}	I_{73}	I_{74}	I_{75}	I_{76}	I_{77}

3x3 filter

Stride

stride 1



3x3 filter

Stride

stride 1

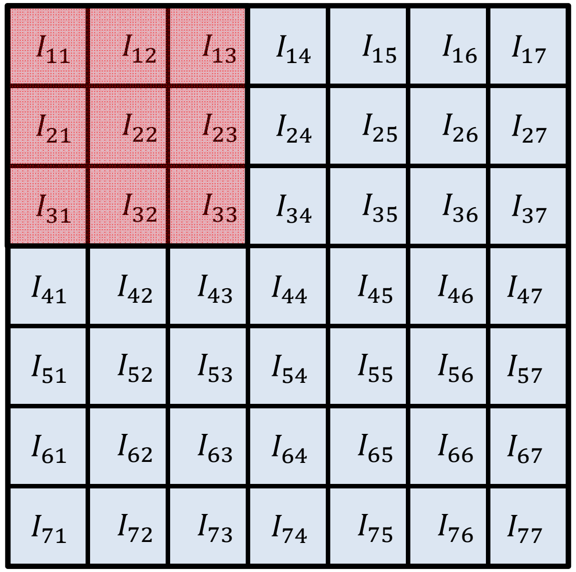
I_{11}	I_{12}	I_{13}	I_{14}	I_{15}	I_{16}	I_{17}
I_{21}	I_{22}	I_{23}	I_{24}	I_{25}	I_{26}	I_{27}
I_{31}	I_{32}	I_{33}	I_{34}	I_{35}	I_{36}	I_{37}
I_{41}	I_{42}	I_{43}	I_{44}	I_{45}	I_{46}	I_{47}
I_{51}	I_{52}	I_{53}	I_{54}	I_{55}	I_{56}	I_{57}
I_{61}	I_{62}	I_{63}	I_{64}	I_{65}	I_{66}	I_{67}
I_{71}	I_{72}	I_{73}	I_{74}	I_{75}	I_{76}	I_{77}

3x3 filter

=> 5x5 output

Stride

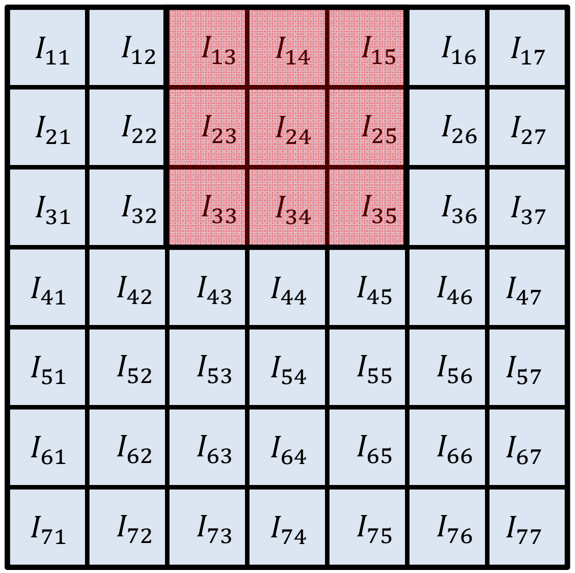
stride 2



3x3 filter

Stride

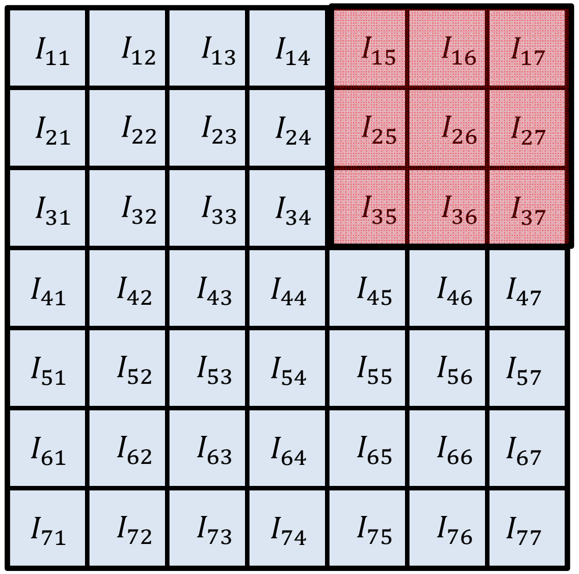
stride 2



3x3 filter

Stride

stride 2



3x3 filter

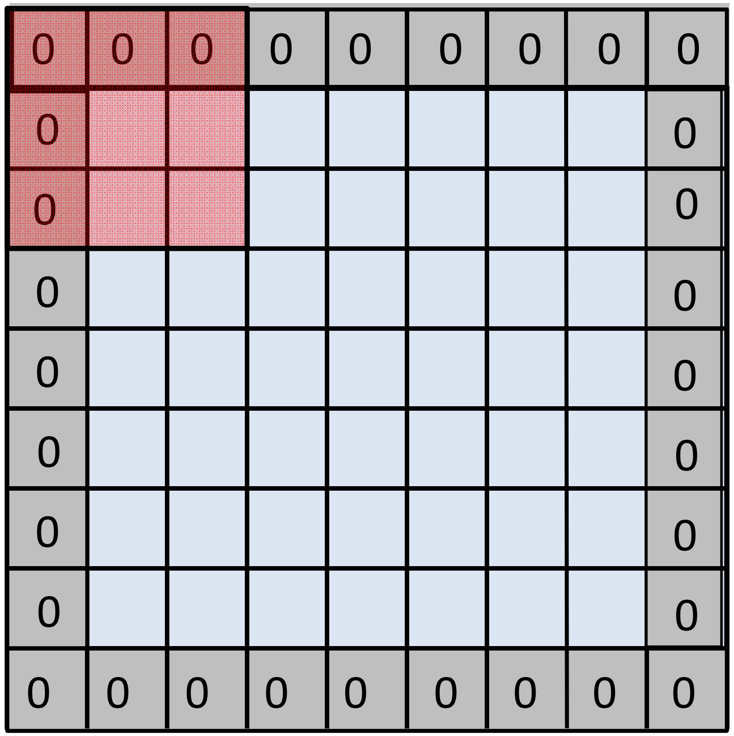
=> 3x3 output

Output Size

- N = input size
- F = filter size
- S = stride

- Output size = (N – F) / S + 1

Zero Padding



7x7 input
Zero padding with 1 pixel border
3x3 filter

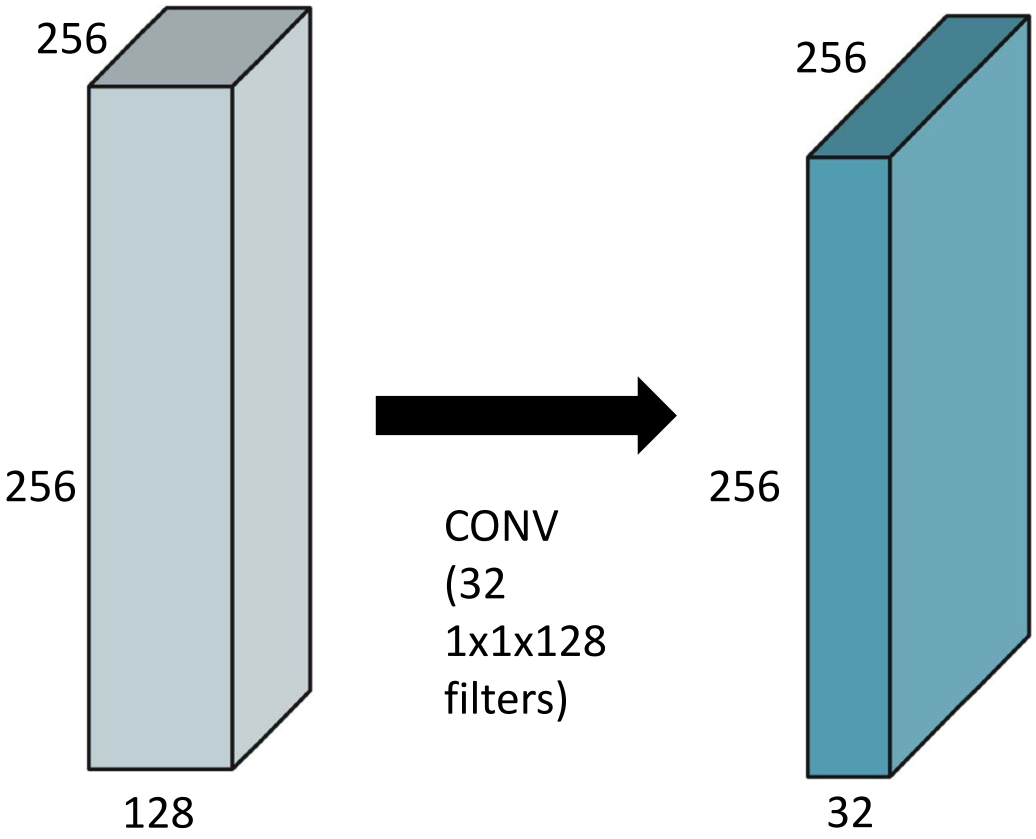
=> 7x7 output

Output Size

- N = input size
- F = filter size
- S = stride
- P = padding size

- Output size = (N + 2P – F) / S + 1

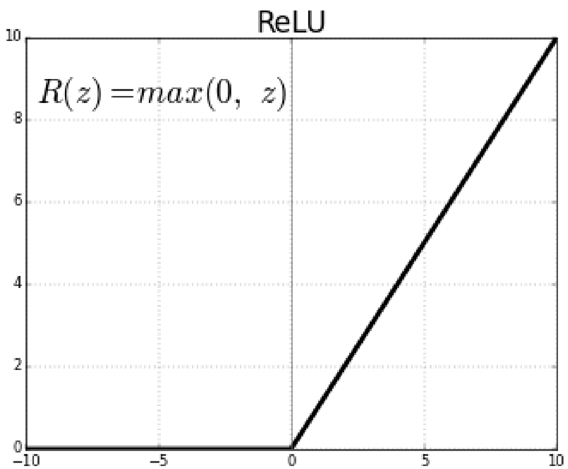
1x1 Convolution



- **Dimension reduction**
- Same output size (H x W)

ReLU Activation

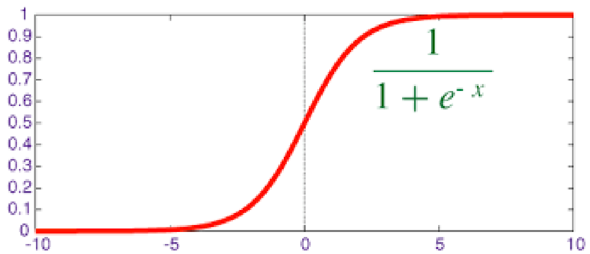
- Preserves properties of linear models
  - Easy to optimize with gradient descent
  - Good generalization
  - Large and consistent gradients
- Overcomes the vanishing gradient problem



Other Activation Functions

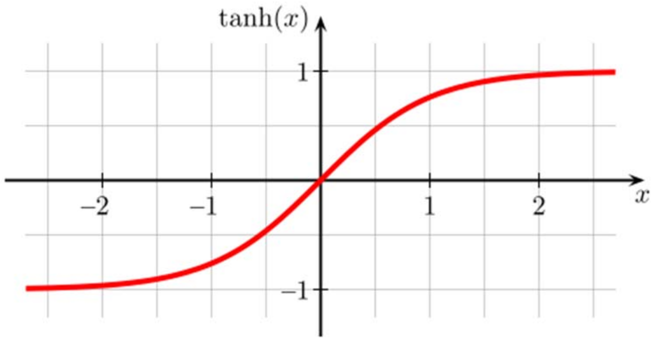
Sigmoid/Logistic

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



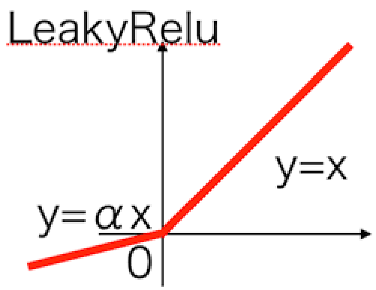
tanh (hyperbolic tangent)

$$g(x) = \tanh x$$



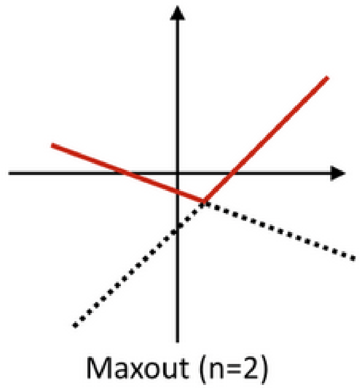
Leaky ReLU

$$g(x) = \max(ax, x), a \leq 1$$



maxout

$$g(x) = \max({a_1}^T x + b_1, {a_2}^T x + b_2)$$



Pooling

1	2	1	0
5	0	0	3
8	0	0	5
0	2	2	0

→ max pooling with 2x2 filter with stride 2 →

5	3
8	5

1	2	1	0
5	0	0	3
8	0	0	5
0	2	2	0

→ average pooling with 2x2 filter with stride 2 →

2	1
2.5	1.4

- **Pooling** makes features **invariant** to local translations of input
- Dimension reduction

Wrap Up

- Convolutional Neural Networks
  - Convolution
  - Activation function: ReLU
  - Pooling