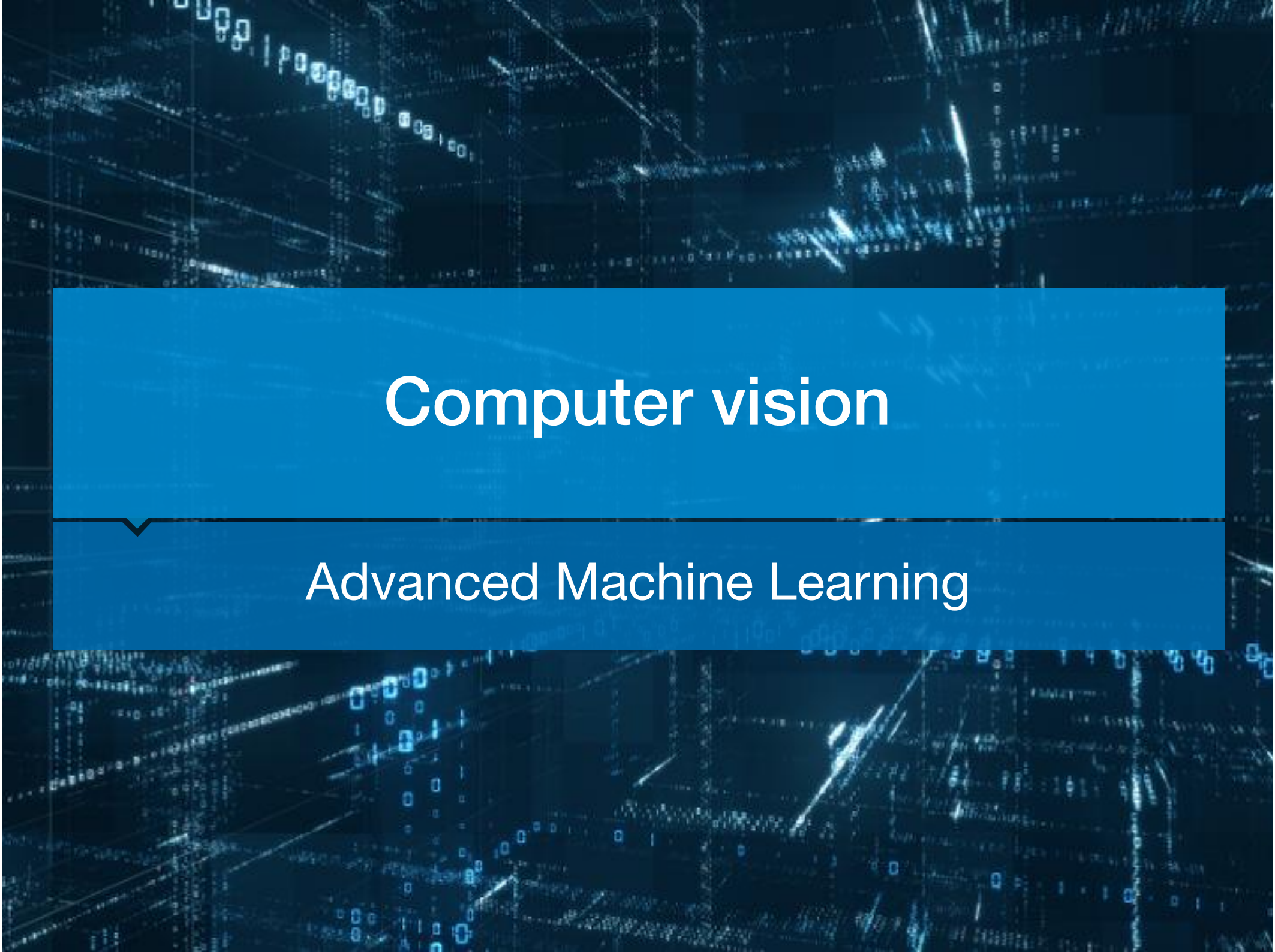
Advanced Machine Learning
Lecture 8: Convolutional Neural Networks

Sandjai Bhulai
Vrije Universiteit Amsterdam

s.bhulai@vu.nl
29 September 2023



Computer vision

Advanced Machine Learning

Computer vision problems

- Image classification

64 x 64

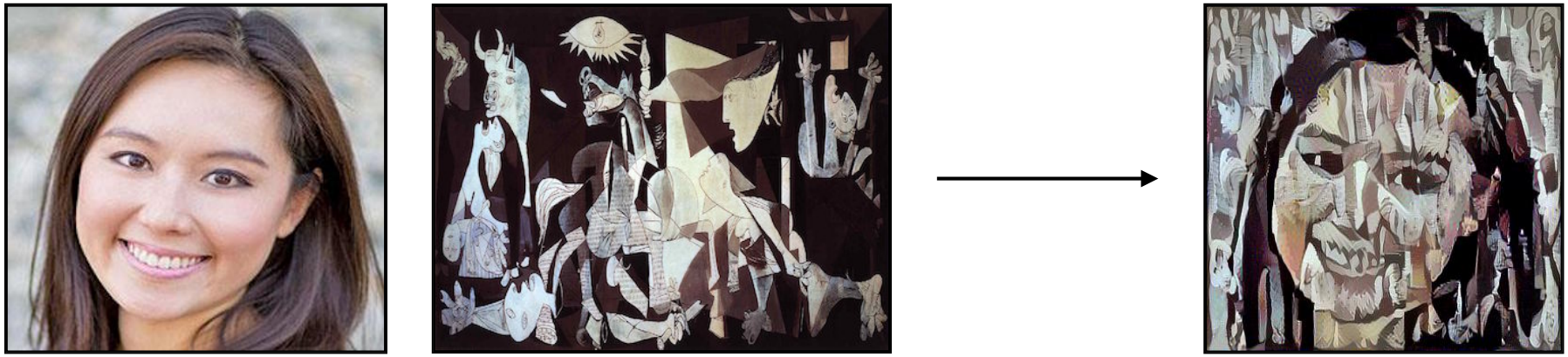
Cat? (0/1)

Computer vision problems

- Image classification
- Object detection

Computer vision problems

- Image classification
- Object detection
- Neural style transfer



Deep learning on large images

64 x 64

→ Cat? (0/1)

Input:
64 x 64 x 3 = 12,288

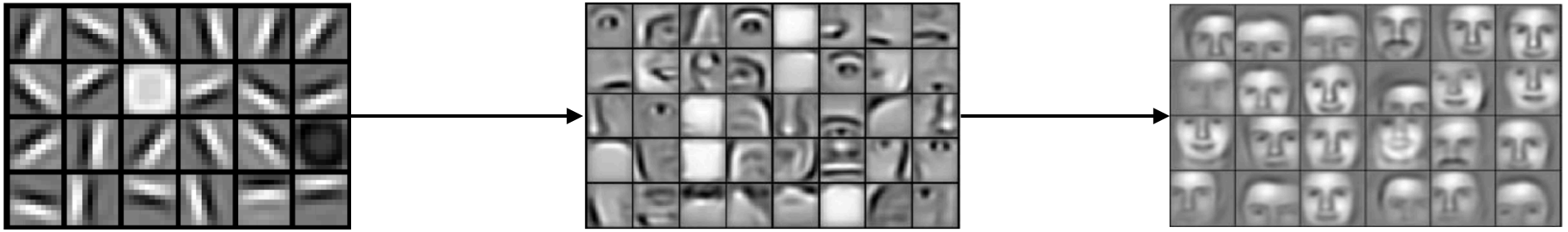
1,000 x 1,000

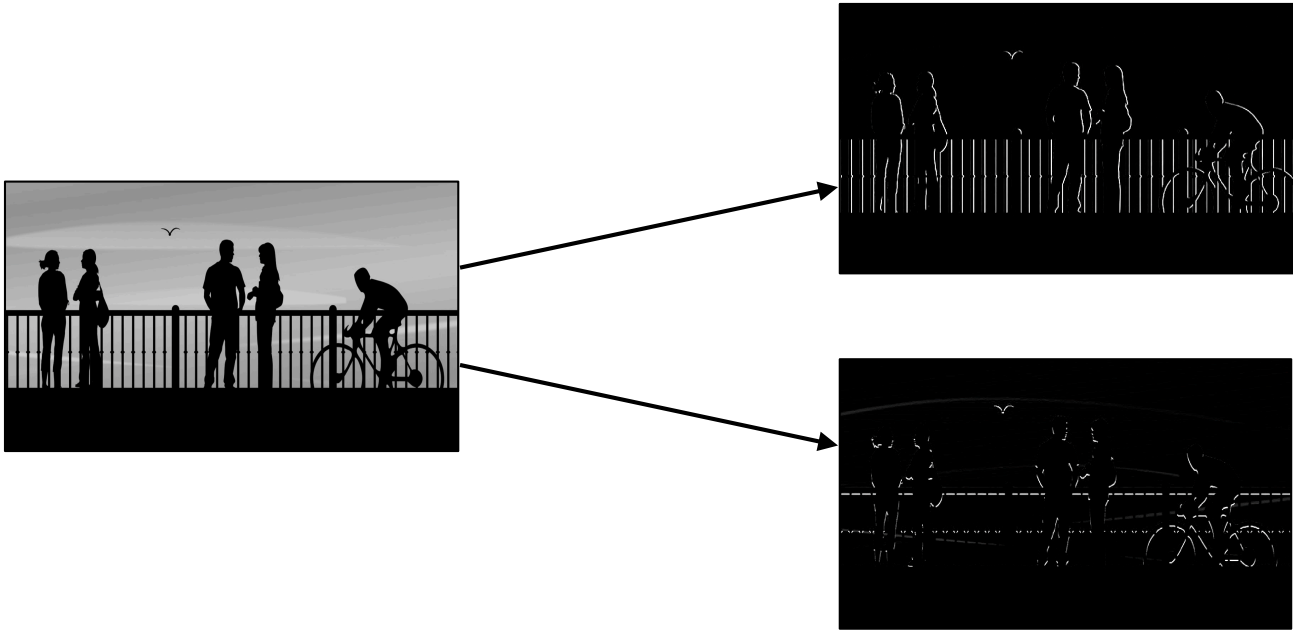
→ Cat? (0/1)

Input:
1,000 x 1,000 x 3 = 3,000,000

2-layer NN, 1,000 hidden units
→ 3 billion weights

Edge detection

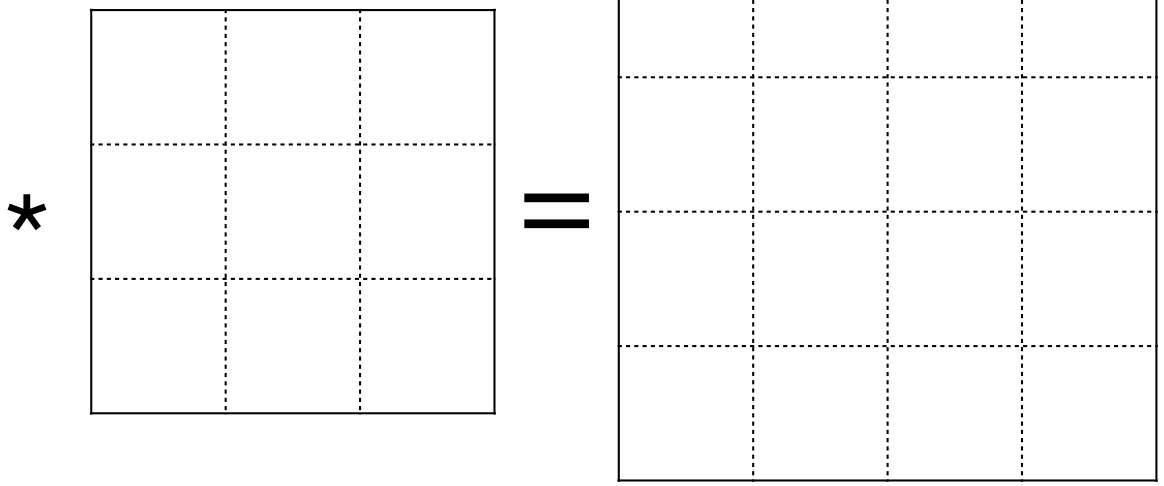




Edge detection

3	0	1	2	7	4
1	5	8	9	3	1
2	7	2	5	1	3
0	1	3	1	7	8
4	2	1	6	2	8
2	4	5	2	3	9

Input: 6 x 6
Filter: 3 x 3
Output: 4 x 4



Edge detection

3	0	1	2	7	4
1	5	8	9	3	1
2	7	2	5	1	3
0	1	3	1	7	8
4	2	1	6	2	8
2	4	5	2	3	9

Input: 6 x 6
Filter: 3 x 3
Output: 4 x 4

*

1	0	-1
1	0	-1
1	0	-1

=

Edge detection

3 (1)	0 (0)	1 (-1)	2	7	4
1 (1)	5 (0)	8 (-1)	9	3	1
2 (1)	7 (0)	2 (-1)	5	1	3
0	1	3	1	7	8
4	2	1	6	2	8
2	4	5	2	3	9

Input: 6 x 6
Filter: 3 x 3
Output: 4 x 4

$*$

1	0	-1
1	0	-1
1	0	-1

$=$

-5			

$$3 \times 1 + 1 \times 1 + 2 \times 1 + 0 \times 0 + 5 \times 0 + 7 \times 0 + 1 \times -1 + 8 \times -1 + 2 \times -1 = -5$$

Edge detection

3	0 (1)	1 (0)	2 (-1)	7	4
1	5 (1)	8 (0)	9 (-1)	3	1
2	7 (1)	2 (0)	5 (-1)	1	3
0	1	3	1	7	8
4	2	1	6	2	8
2	4	5	2	3	9

Input: 6 x 6
Filter: 3 x 3
Output: 4 x 4

*

1	0	-1
1	0	-1
1	0	-1

=

-5	-4		

$$0 \times 1 + 5 \times 1 + 7 \times 1 + 1 \times 0 + 8 \times 0 + 2 \times 0 + 2 \times -1 + 9 \times -1 + 5 \times -1 = -4$$

Edge detection

3	0	1^{1}	2^{0}	7^{-1}	4
1	5	8^{1}	9^{0}	3^{-1}	1
2	7	2^{1}	5^{0}	1^{-1}	3
0	1	3	1	7	8
4	2	1	6	2	8
2	4	5	2	3	9

Input: 6 x 6
Filter: 3 x 3
Output: 4 x 4

$*$

1	0	-1
1	0	-1
1	0	-1

$=$

-5	-4	0	

$$1 \times 1 + 8 \times 1 + 2 \times 1 + 2 \times 0 + 9 \times 0 + 5 \times 0 + 7 \times -1 + 3 \times -1 + 1 \times -1 = 0$$

Edge detection

3	0	1	2^{1}	7^{0}	4^{-1}
1	5	8	9^{1}	3^{0}	1^{-1}
2	7	2	5^{1}	1^{0}	3^{-1}
0	1	3	1	7	8
4	2	1	6	2	8
2	4	5	2	3	9

Input: 6 x 6
Filter: 3 x 3
Output: 4 x 4

*

1	0	-1
1	0	-1
1	0	-1

=

-5	-4	0	8

$$2 \times 1 + 9 \times 1 + 5 \times 1 + 7 \times 0 + 3 \times 0 + 1 \times 0 + 4 \times -1 + 1 \times -1 + 3 \times -1 = 8$$

Edge detection

3	0	1	2	7	4
1	5	8	9	3	1
2	7	2	5	1	3
0	1	3	1	7	8
4	2	1	6	2	8
2	4	5	2	3	9

Input: 6 x 6
Filter: 3 x 3
Output: 4 x 4

*

1	0	-1
1	0	-1
1	0	-1

=

-5	-4	0	8
-10	-2	2	3
0	-2	-4	-7
-3	-2	-3	-16

Edge detection

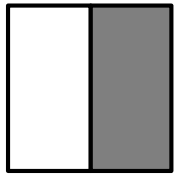
10	10	10	0	0	0
10	10	10	0	0	0
10	10	10	0	0	0
10	10	10	0	0	0
10	10	10	0	0	0
10	10	10	0	0	0

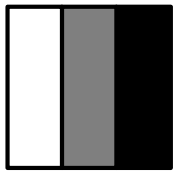
*

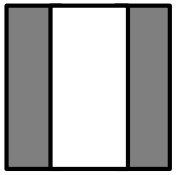
1	0	-1
1	0	-1
1	0	-1

=

0	30	30	0
0	30	30	0
0	30	30	0
0	30	30	0







Edge detection

10	10	10	0	0	0
10	10	10	0	0	0
10	10	10	0	0	0
10	10	10	0	0	0
10	10	10	0	0	0
10	10	10	0	0	0

*

1	0	-1
1	0	-1
1	0	-1

=

0	30	30	0
0	30	30	0
0	30	30	0
0	30	30	0

0	0	0	10	10	10
0	0	0	10	10	10
0	0	0	10	10	10
0	0	0	10	10	10
0	0	0	10	10	10
0	0	0	10	10	10

*

1	0	-1
1	0	-1
1	0	-1

=

0	-30	-30	0
0	-30	-30	0
0	-30	-30	0
0	-30	-30	0

Edge detection

1	0	-1
1	0	-1
1	0	-1

vertical

1	1	1
0	0	0
-1	-1	-1

horizontal

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

Sobel

3	0	-3
10	0	-10
3	0	-3

Scharr

10	10	10	0	0	0
10	10	10	0	0	0
10	10	10	0	0	0
0	0	0	10	10	10
0	0	0	10	10	10
0	0	0	10	10	10

*

1	1	1
0	0	0
-1	-1	-1

=

0	0	0	0
30	10	-10	-30
30	10	-10	-30
0	0	0	0

Edge detection

1	0	-1
1	0	-1
1	0	-1

vertical

1	1	1
0	0	0
-1	-1	-1

horizontal

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

Sobel

3	0	-3
10	0	-10
3	0	-3

Scharr

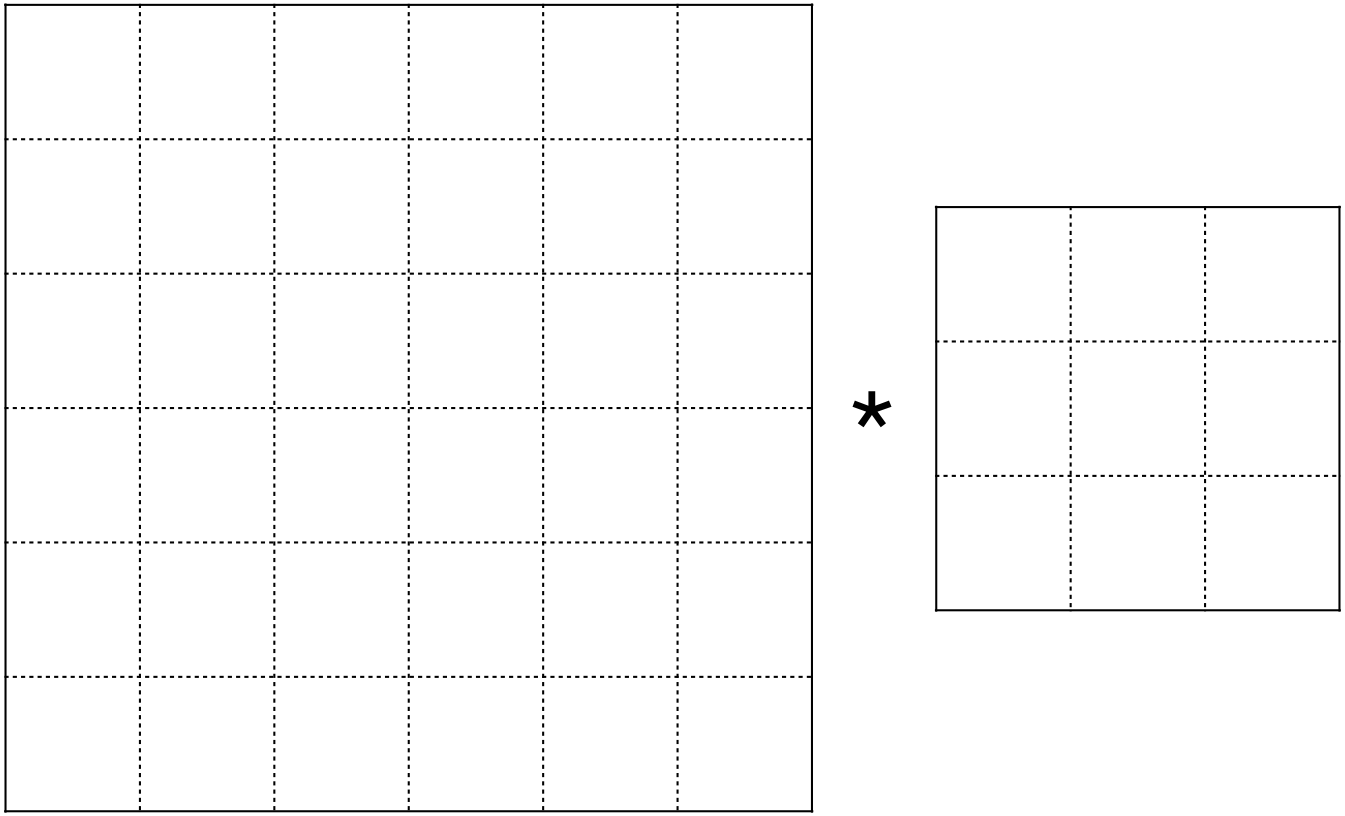
3	0	1	2	7	4
1	5	8	9	3	1
2	7	2	5	1	3
0	1	3	1	7	8
4	2	1	6	2	8
2	4	5	2	3	9

*

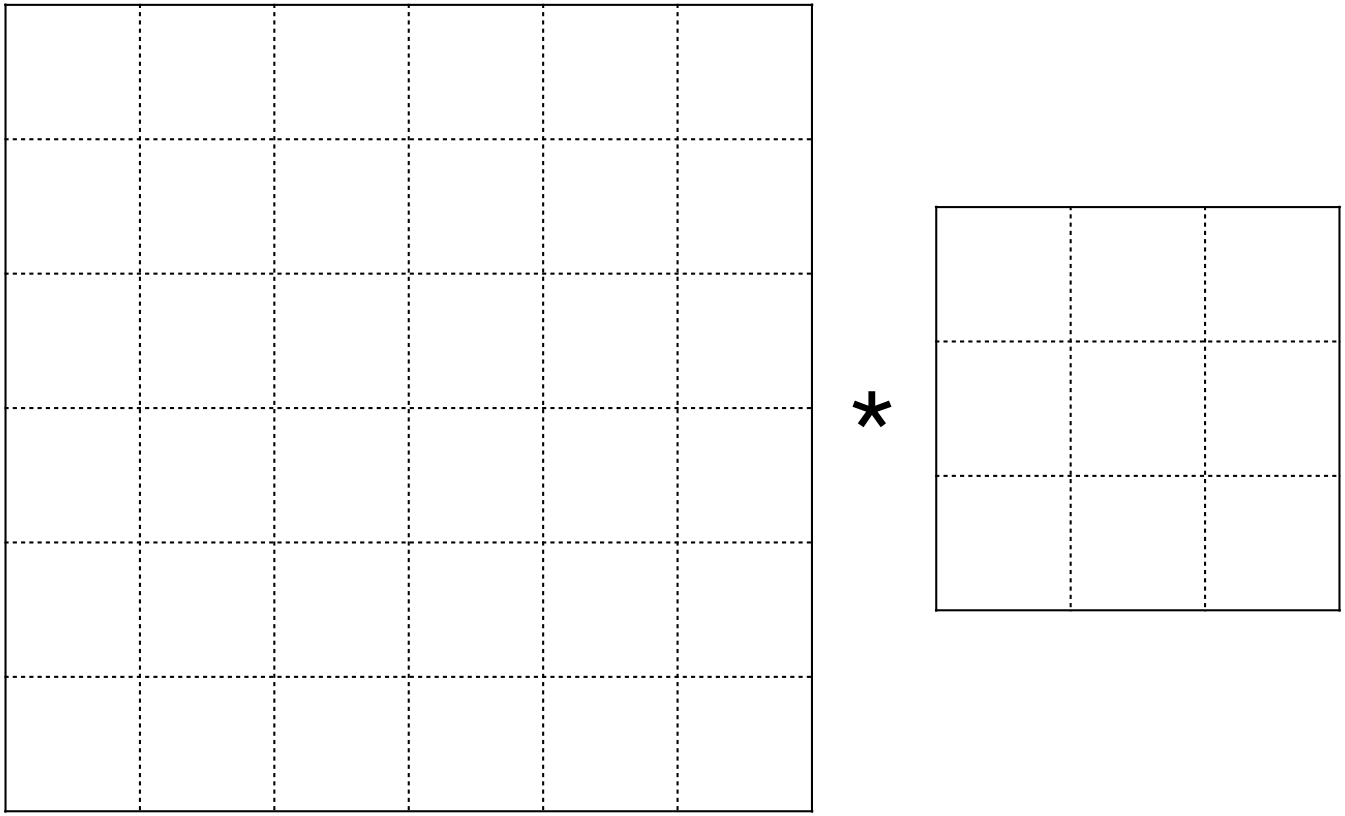
w_1	w_2	w_3
w_4	w_5	w_6
w_7	w_8	w_9

=

Padding



Padding



Input: n x n

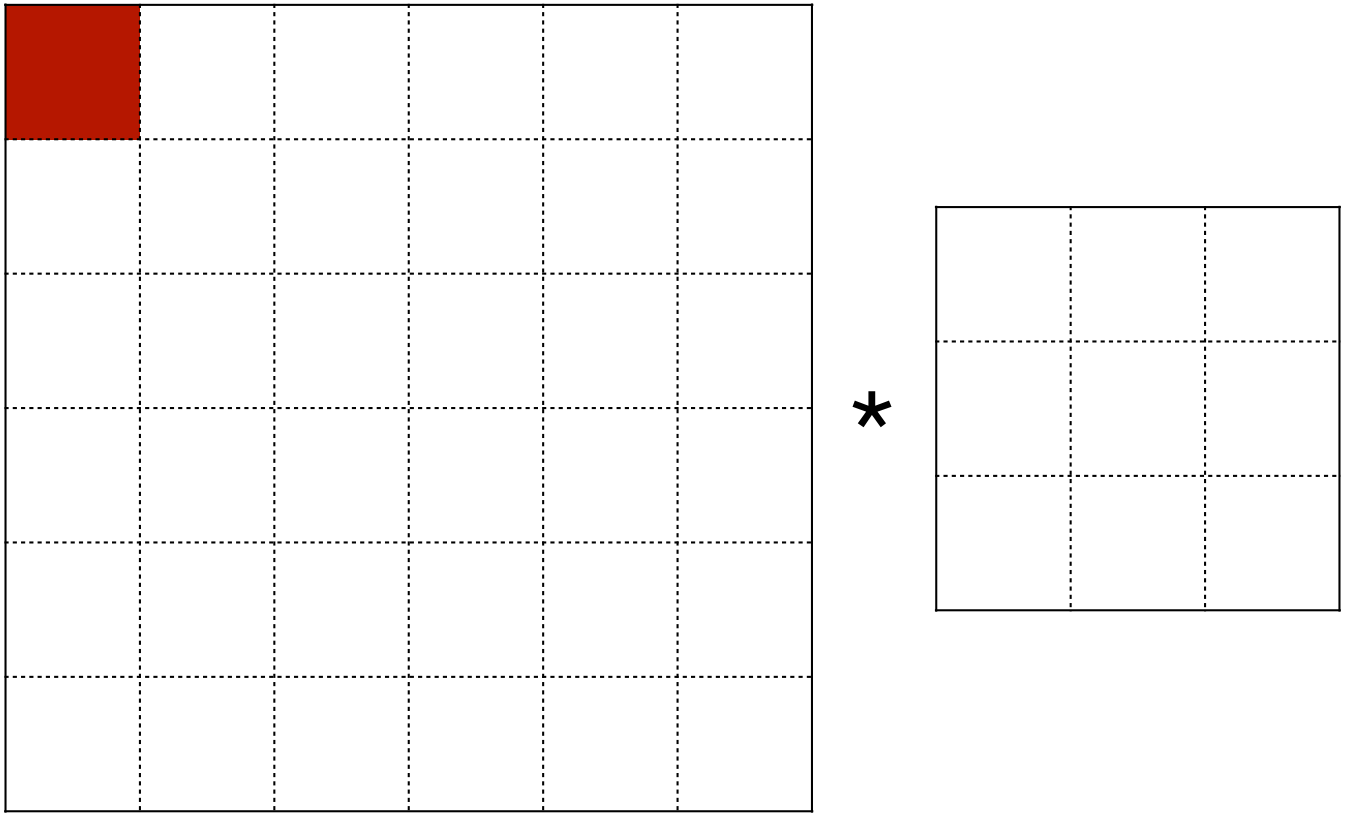
Filter: f x f

Output: $n-f+1$ x $n-f+1$

→ • Shrinking output

VU

Padding



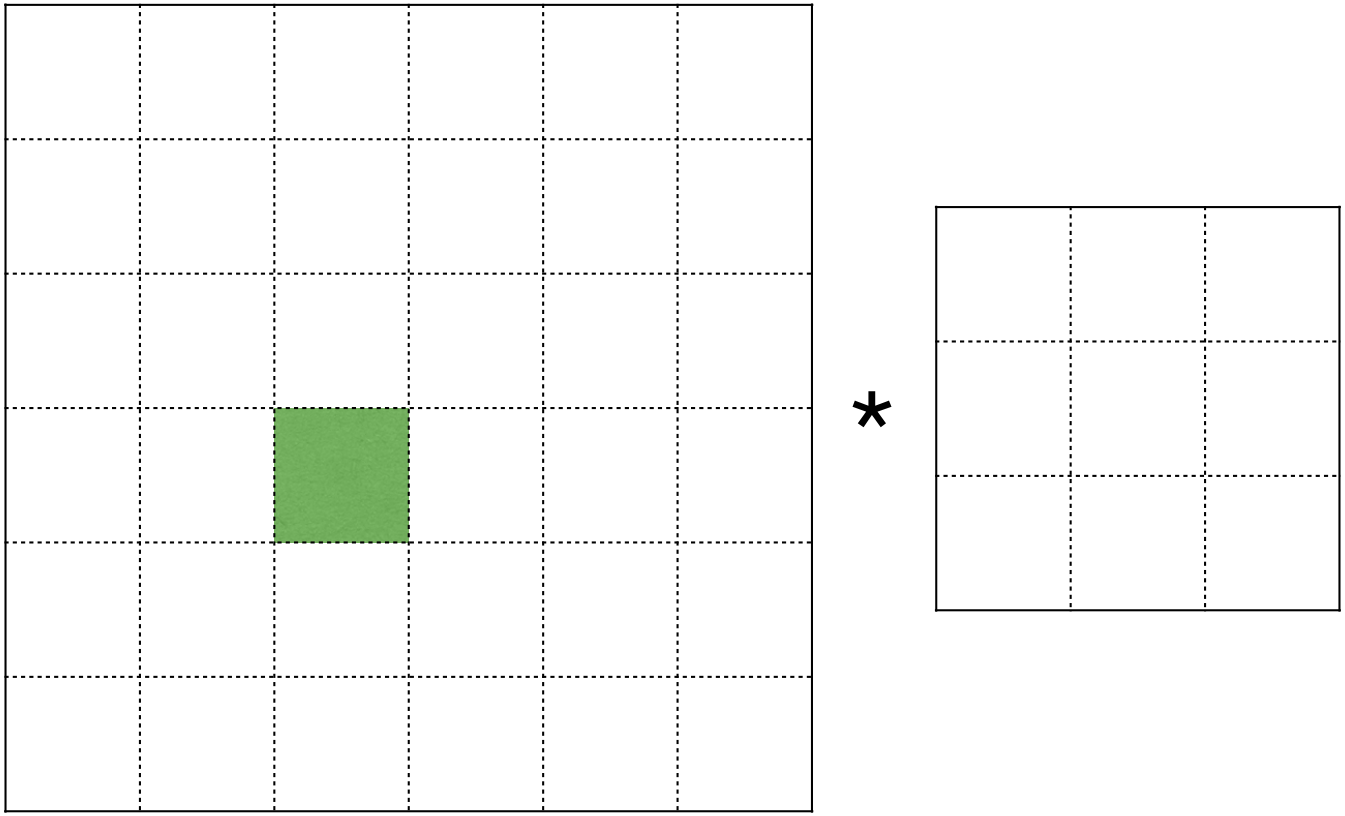
Input: n x n

Filter: f x f

Output: $n-f+1$ x $n-f+1$

→ • Shrinking output

Padding



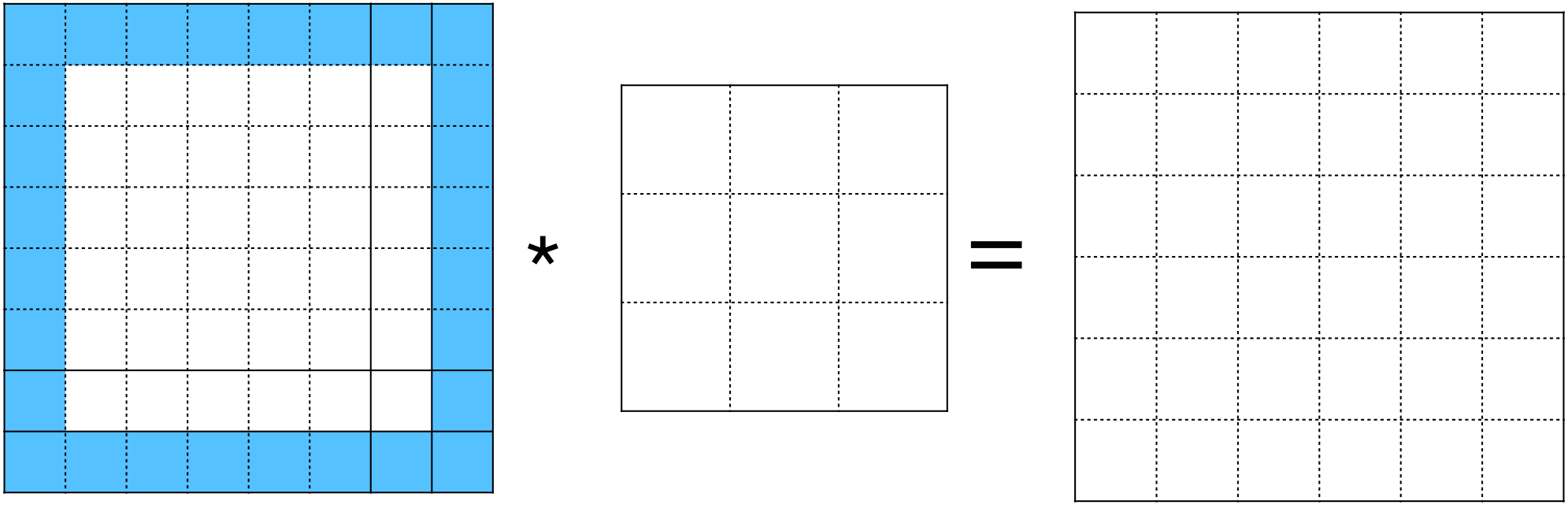
Input: n x n

Filter: f x f

Output: $n-f+1$ x $n-f+1$

- Shrinking output
- Edge information is discarded

Padding



Input: n x n

Filter: f x f

Padding: p

Output: $n + 2p - f + 1$ x $n + 2p - f + 1$

Convolutions

- Valid convolution
  - No padding
  - Output is $n - f + 1$ x $n - f + 1$

- Same convolution
  - Padding is $p = \frac{f - 1}{2}$
  - Output n x n

Convolutions

2	3	7	4	6	2	9
6	6	9	8	7	4	3
3	4	8	3	8	9	7
7	8	3	6	6	3	4
4	2	1	8	3	4	6
3	2	4	1	9	8	3
0	1	3	9	2	1	4

$*$

3	4	4
1	0	2
-1	0	3

$=$

Stride: 2

Convolutions

2^{3}	3^{4}	7^{4}	4	6	2	9
6^{1}	6^{0}	9^{2}	8	7	4	3
3^{-1}	4^{0}	8^{3}	3	8	9	7
7	8	3	6	6	3	4
4	2	1	8	3	4	6
3	2	4	1	9	8	3
0	1	3	9	2	1	4

$*$

3	4	4
1	0	2
-1	0	3

$=$

91		

Stride: 2

Convolutions

2	3	7^{3}	4^{4}	6^{4}	2	9
6	6	9^{1}	8^{0}	7^{2}	4	3
3	4	8^{-1}	3^{0}	8^{3}	9	7
7	8	3	6	6	3	4
4	2	1	8	3	4	6
3	2	4	1	9	8	3
0	1	3	9	2	1	4

$*$

3	4	4
1	0	2
-1	0	3

$=$

91	100	

Stride: 2

Convolutions

2	3	7	4	6^{3}	2^{4}	9^{4}
6	6	9	8	7^{1}	4^{0}	3^{2}
3	4	8	3	8^{-1}	9^{0}	7^{3}
7	8	3	6	6	3	4
4	2	1	8	3	4	6
3	2	4	1	9	8	3
0	1	3	9	2	1	4

$*$

3	4	4
1	0	2
-1	0	3

$=$

91	100	83

Stride: 2

Convolutions

2	3	7	4	6	2	9
6	6	9	8	7	4	3
3^{3}	4^{4}	8^{4}	3	8	9	7
7^{1}	8^{0}	3^{2}	6	6	3	4
4^{-1}	2^{0}	1^{3}	8	3	4	6
3	2	4	1	9	8	3
0	1	3	9	2	1	4

$*$

3	4	4
1	0	2
-1	0	3

$=$

91	100	83
69		

Stride: 2

Convolutions

2	3	7	4	6	2	9
6	6	9	8	7	4	3
3	4	8	3	8	9	7
7	8	3	6	6	3	4
4	2	1	8	3	4	6
3	2	4	1	9	8	3
0	1	3	9	2	1	4

$*$

3	4	4
1	0	2
-1	0	3

$=$

91	100	83
69	91	127
44	72	74

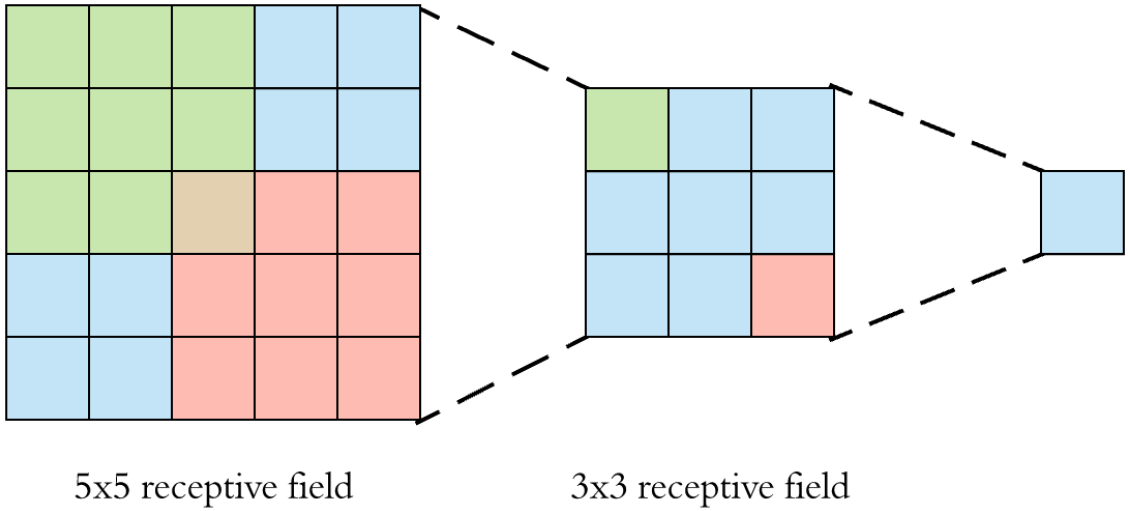
Stride: 2

Convolutions

- Image: n x n
- Filter: f x f (filter size is odd)
- Padding: p
- Stride: s
- Output: $\left\lfloor \frac{n+2p-f}{s} + 1 \right\rfloor$ x $\left\lfloor \frac{n+2p-f}{s} + 1 \right\rfloor$

Filter sizes

- Why 3 x 3 filters
- The smallest possible filter to capture 4 directions
- Two 3 x 3 filters have the receptive field of one 5 x 5
- Three 3 x 3 filters have the receptive field of one 7 x 7



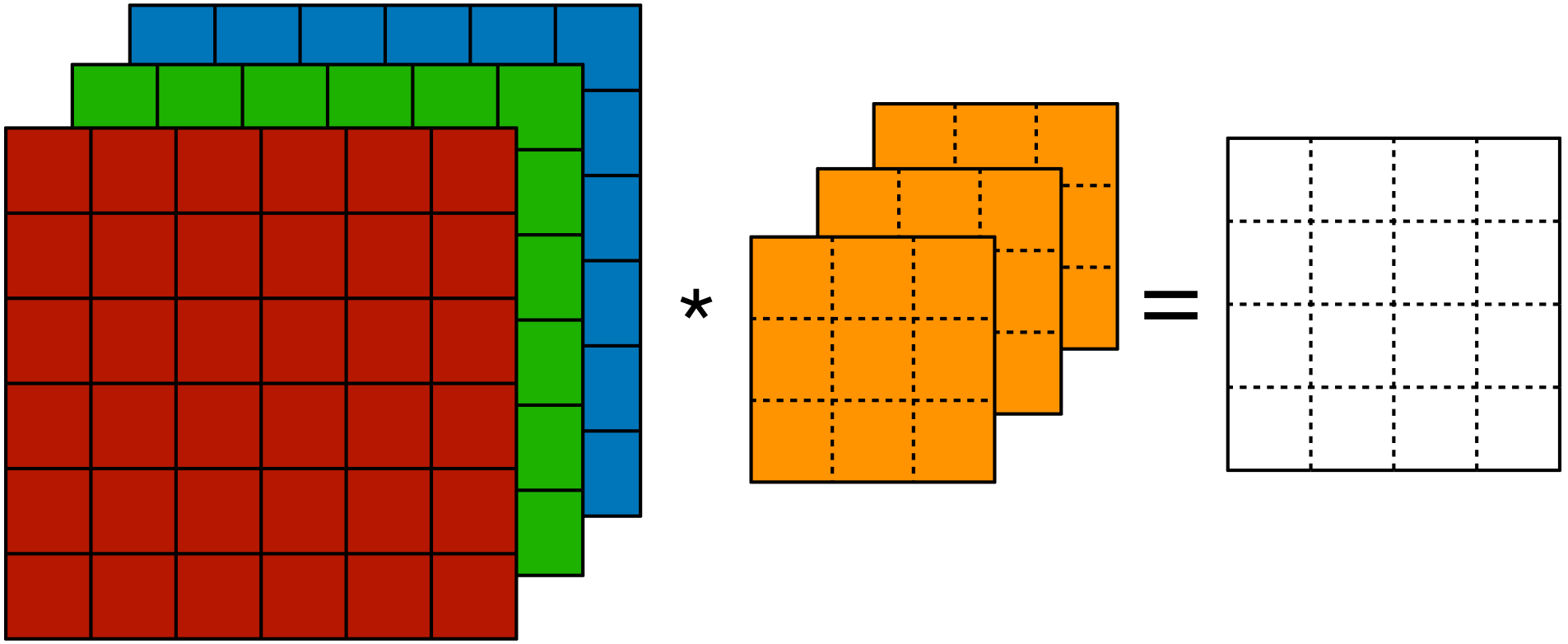
Filter sizes

- Why 3 x 3 filters
- The smallest possible filter to capture 4 directions
- Two 3 x 3 filters have the receptive field of one 5 x 5
- Three 3 x 3 filters have the receptive field of one 7 x 7

- 1 large filter can be replaced by a deeper stack of successive smaller filters
  - > More non-linearities for the same “size” of pattern learning
  - > Fewer parameters and regularization

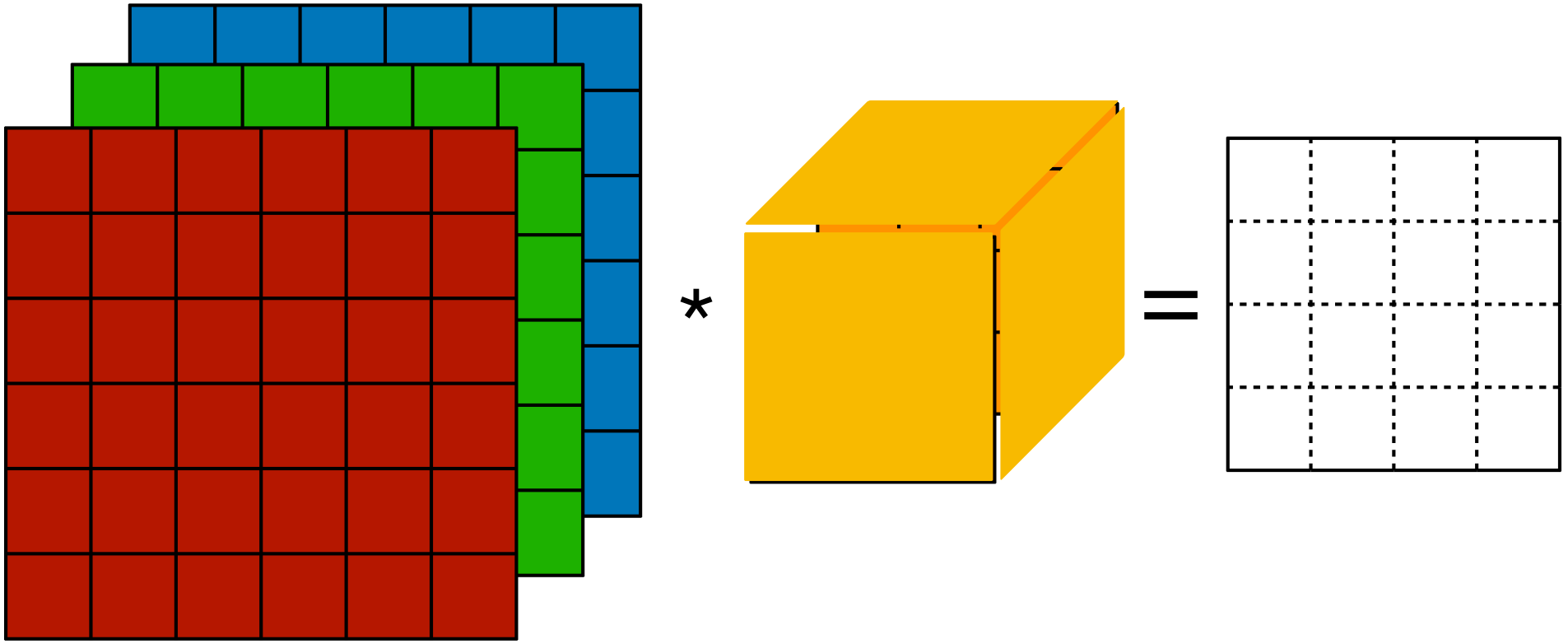
Convolutions

- Convolutions for RGB images



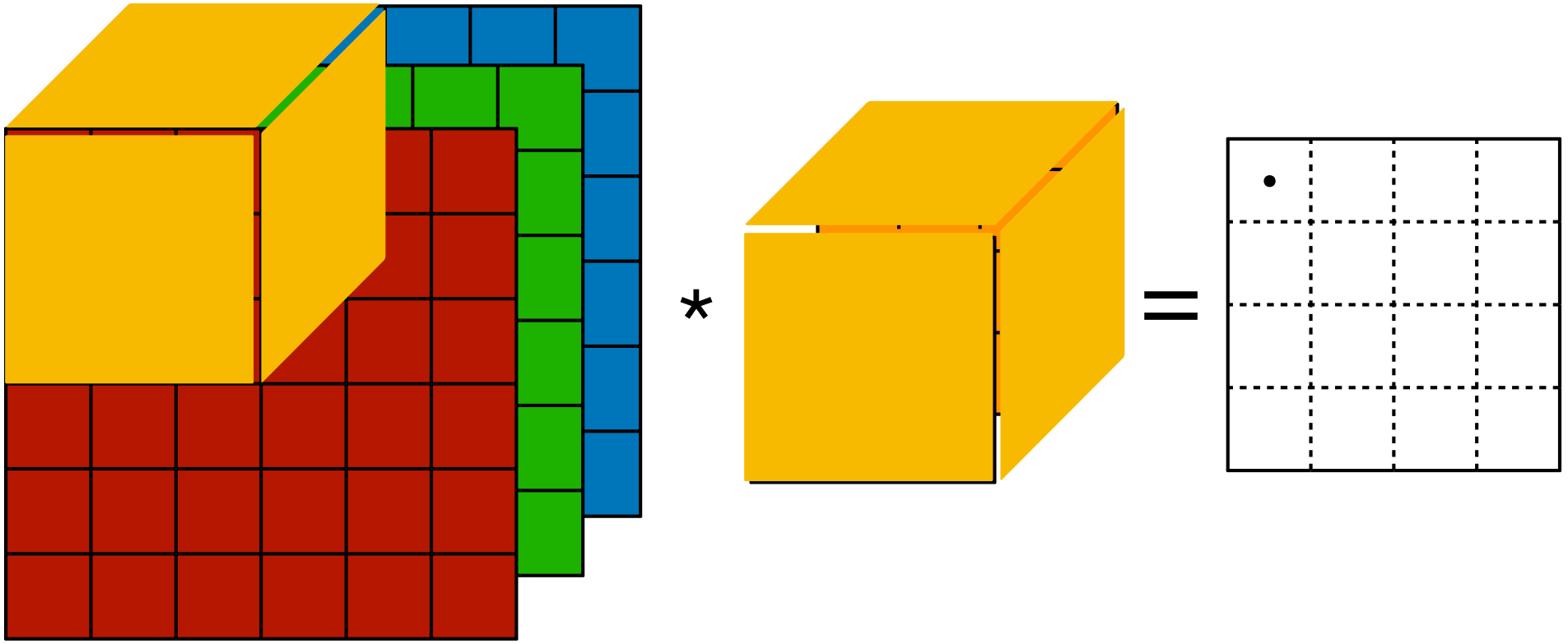
Convolutions

- Convolutions for RGB images



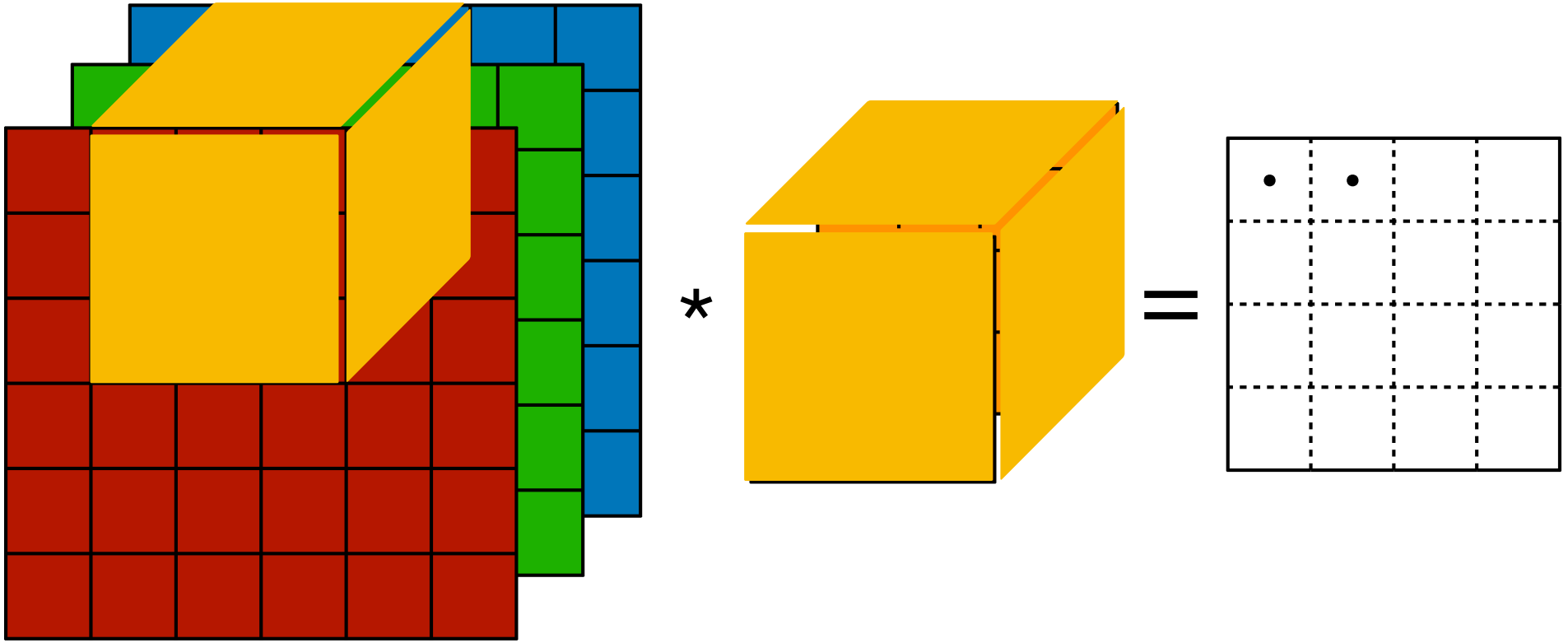
Convolutions

- Convolutions for RGB images



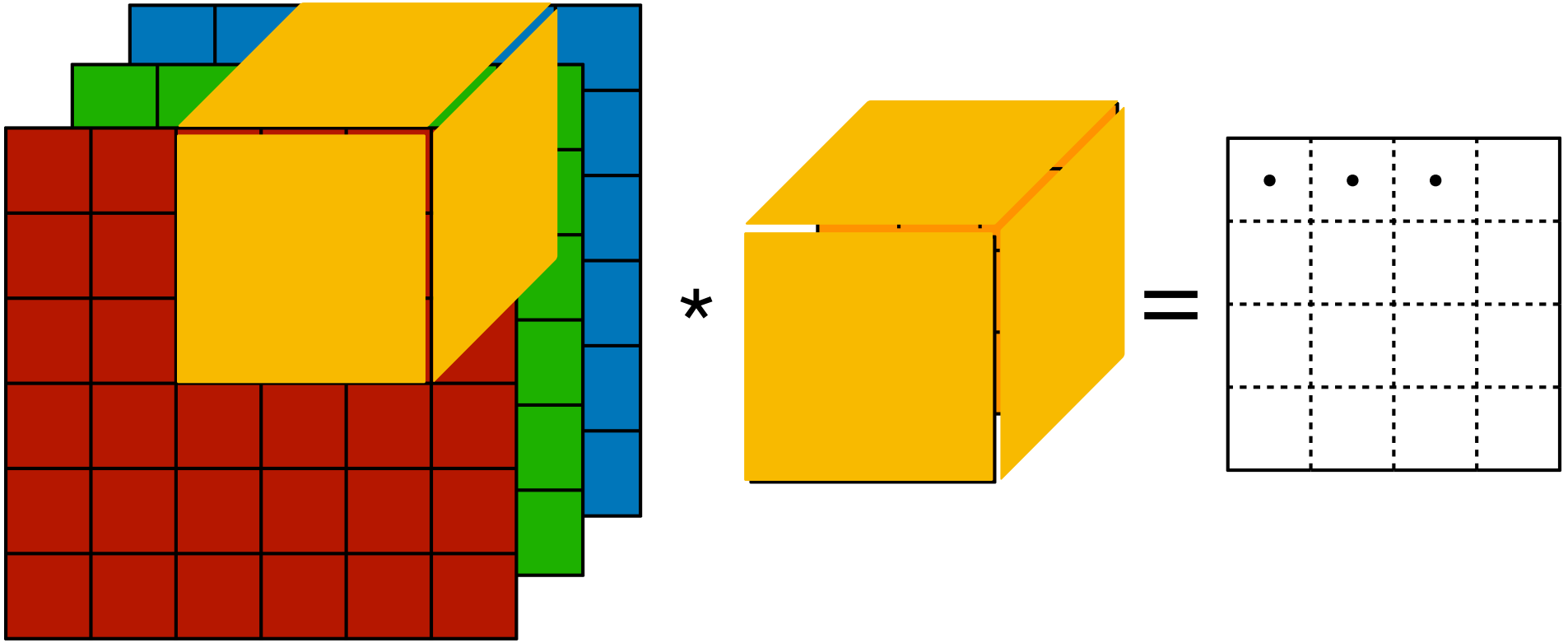
Convolutions

- Convolutions for RGB images



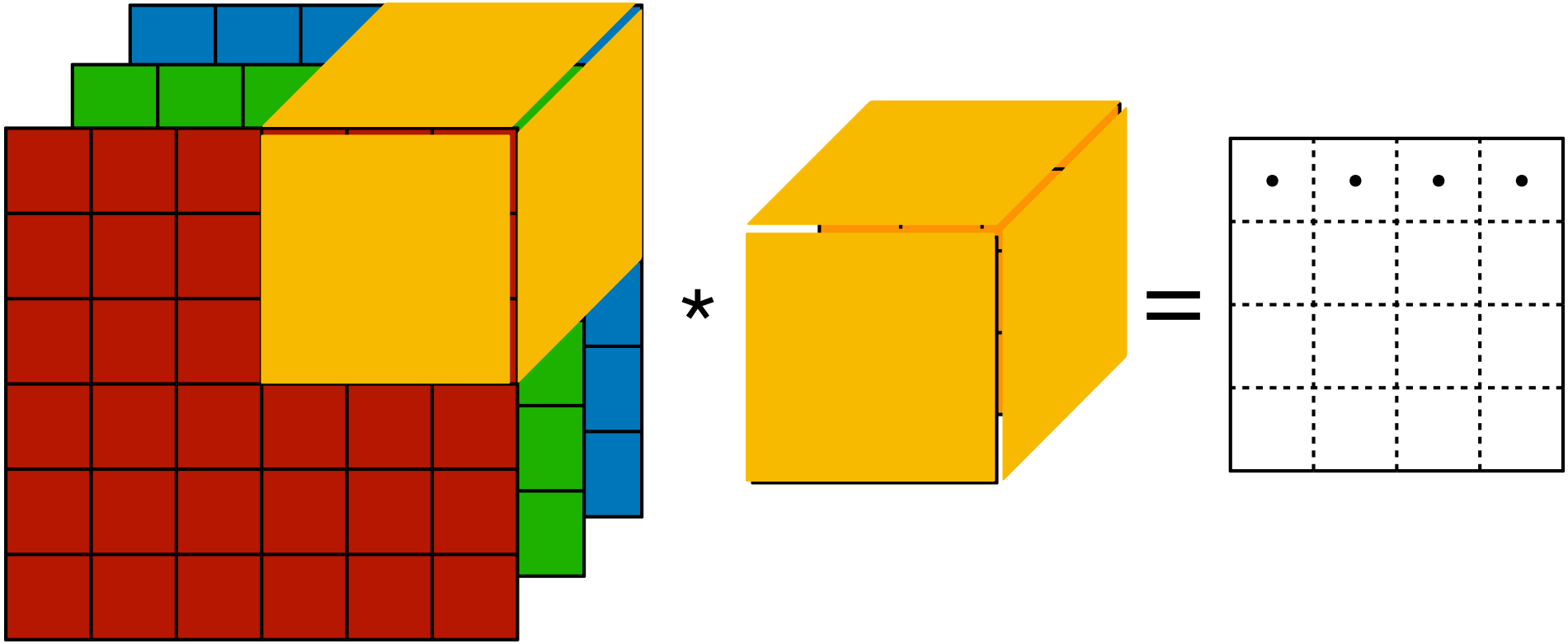
Convolutions

- Convolutions for RGB images



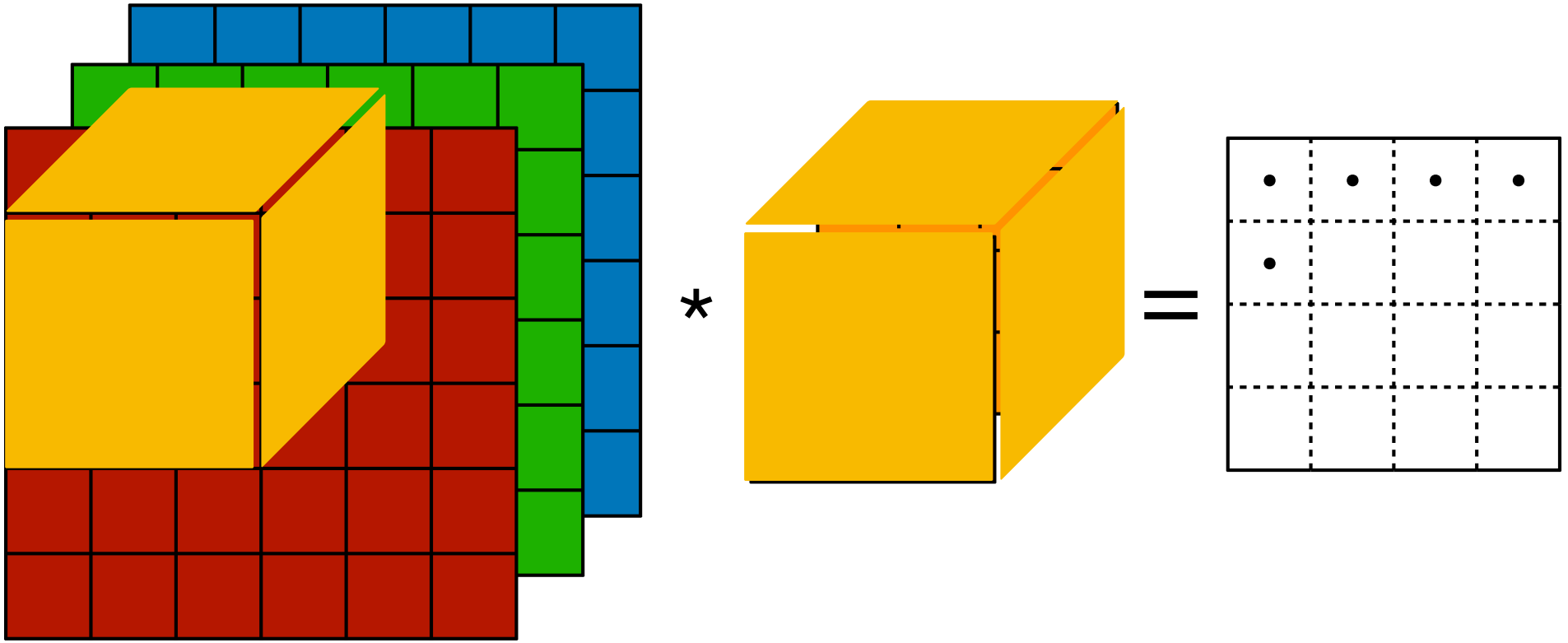
Convolutions

- Convolutions for RGB images



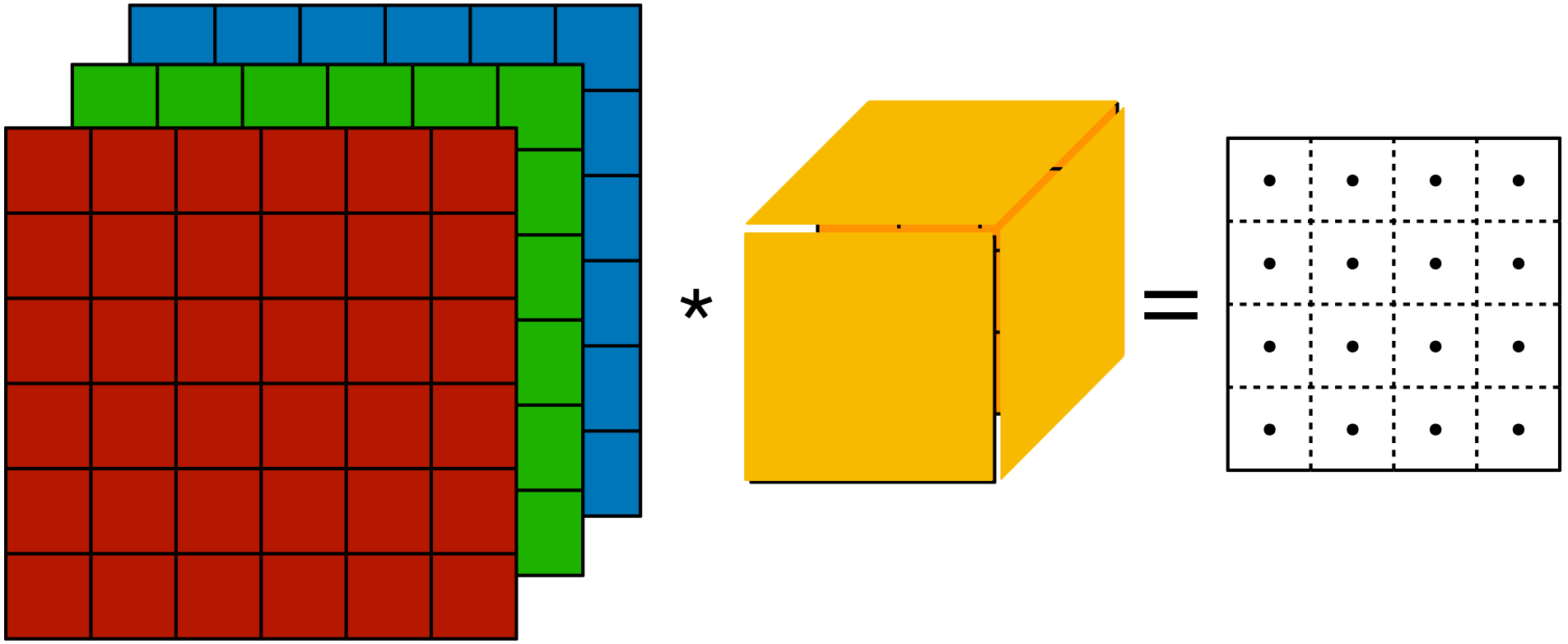
Convolutions

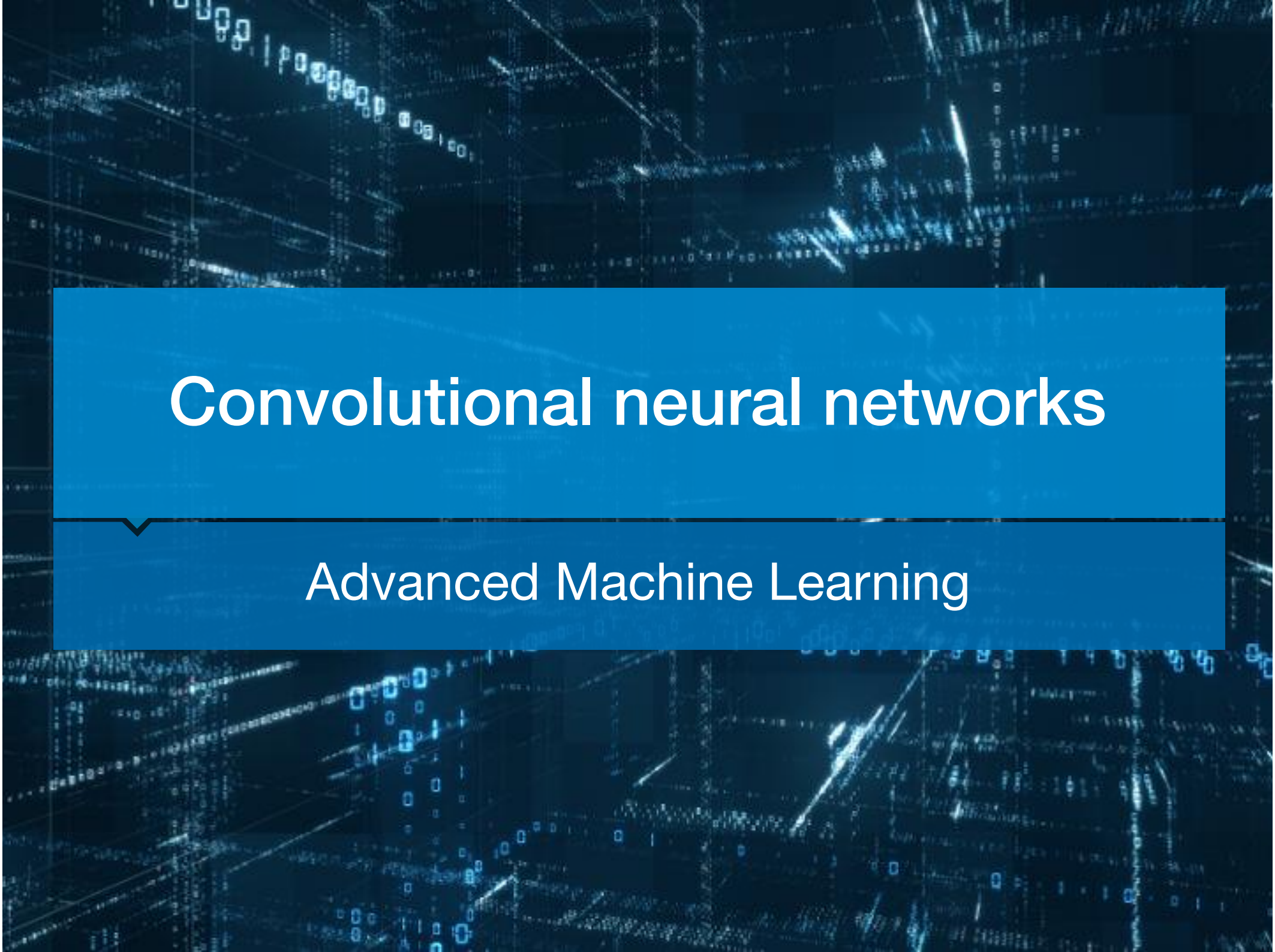
- Convolutions for RGB images



Convolutions

- Convolutions for RGB images



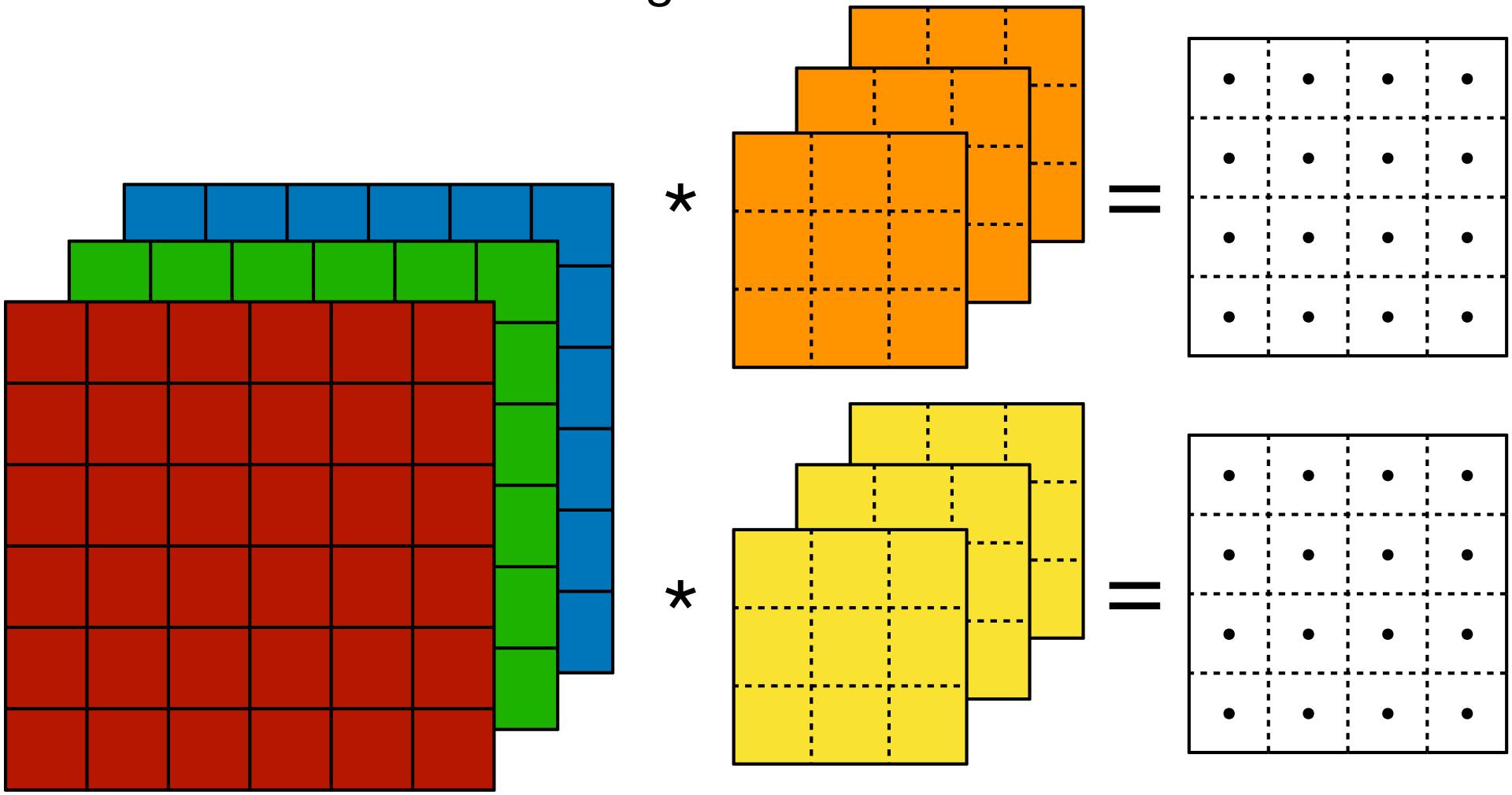


Convolutional neural networks

Advanced Machine Learning

Convolutional neural networks

- Convolutions for RGB images



Convolutional neural networks

- If you have 10 filters that are 3 x 3 x 3 in one layer of a neural network. How many parameters does that layer have?

- Answer: 3 x 3 x 3 = 27 + 1 bias parameter = 28
  10 filters x 28 = 280 parameters

- This is independent of the size of the picture!

Convolutional neural networks

Notation for layer l

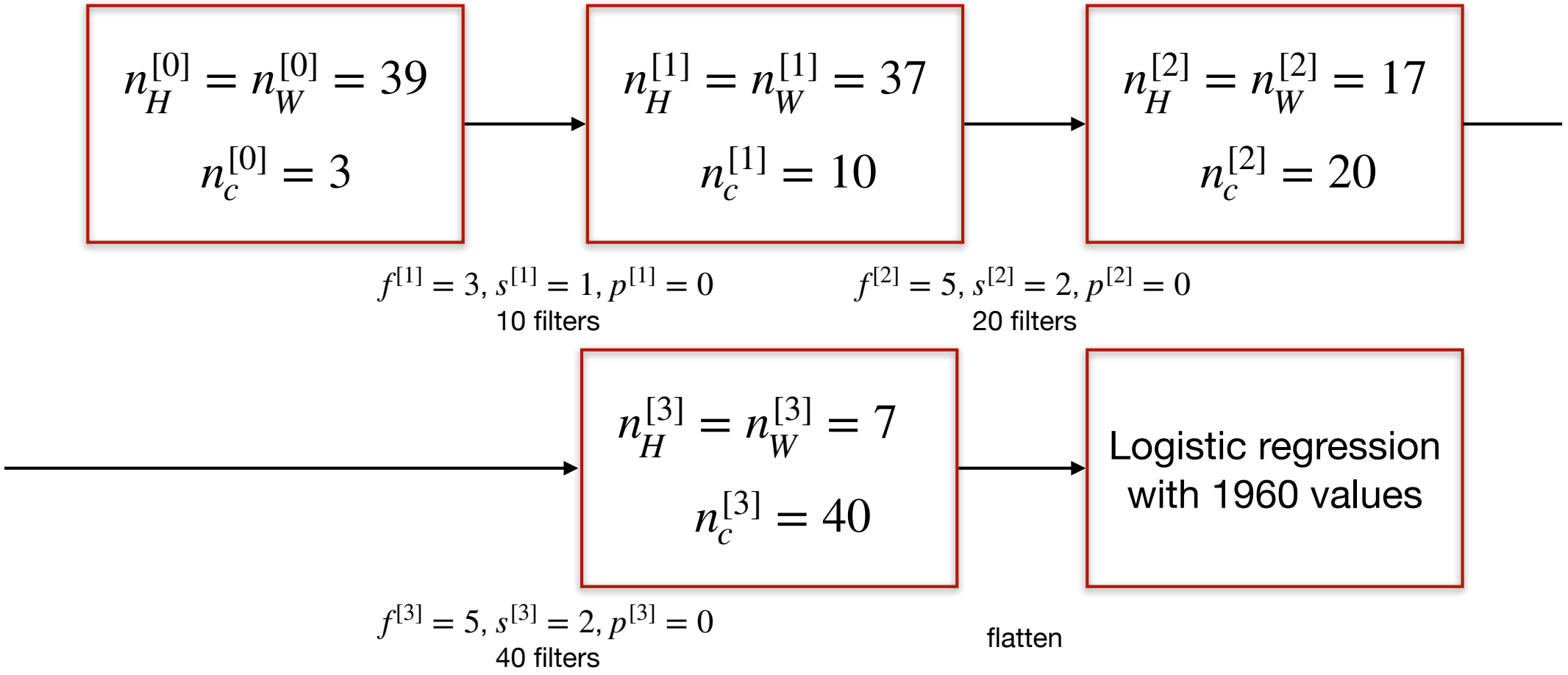
- Filter size: $f^{[l]}$
- Padding: $p^{[l]}$
- Stride: $s^{[l]}$
- Number of filters: $n_c^{[l]}$

$$n_{(H\ W)}^{[l]} = \left\lfloor \frac{n_{(H\ W)}^{[l-1]} + 2p^{[l-1]} - f^{[l-1]}}{s^{[l-1]}} + 1 \right\rfloor$$

- Input: $n_H^{[l-1]} \times n_W^{[l-1]} \times n_c^{[l-1]}$
- Output: $n_H^{[l]} \times n_W^{[l]} \times n_c^{[l]}$

Convolutional neural networks

- A simple network for images of size 39 x 39



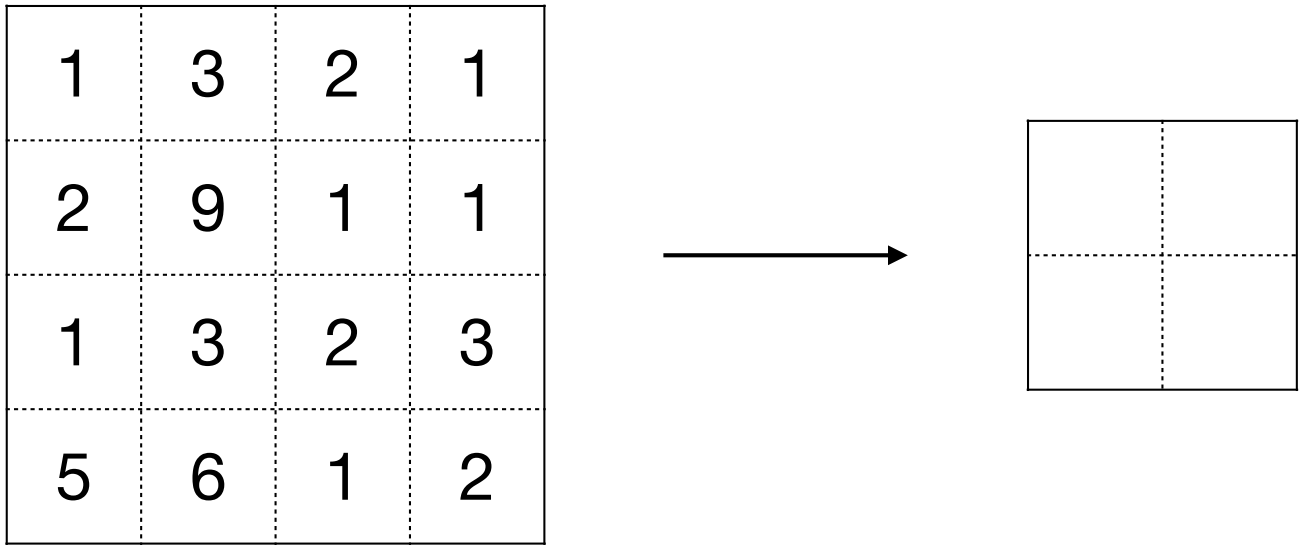
Convolutional neural networks

Types of layers in a convolutional network:

- Convolution (CONV)
- Pooling (POOL)
- Fully connected (FC)

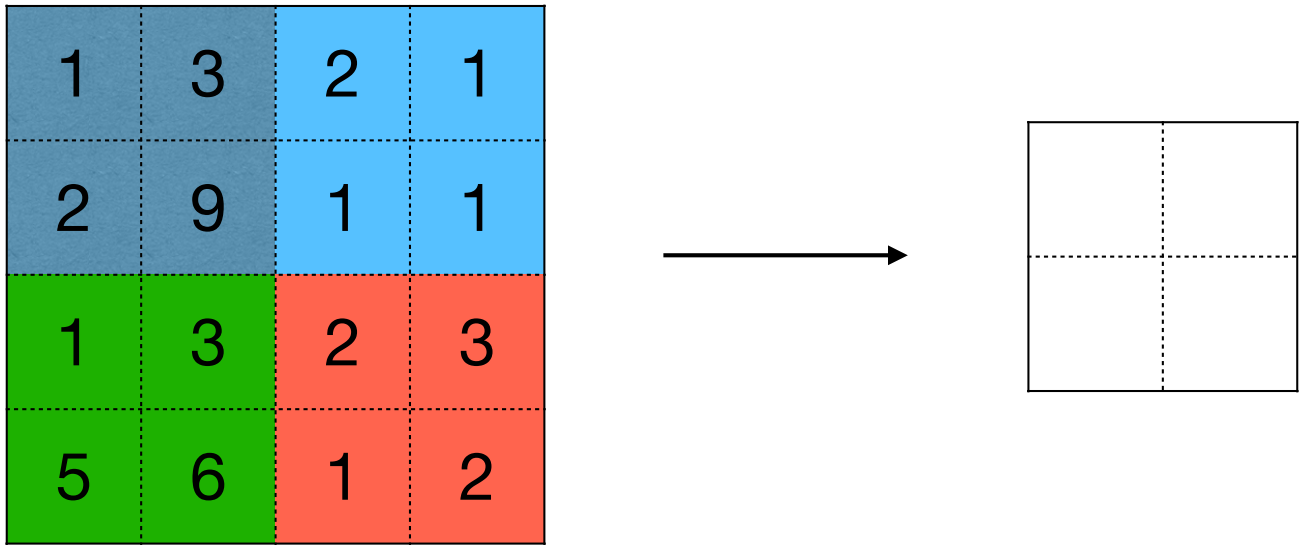
Pooling layers

- Max pooling



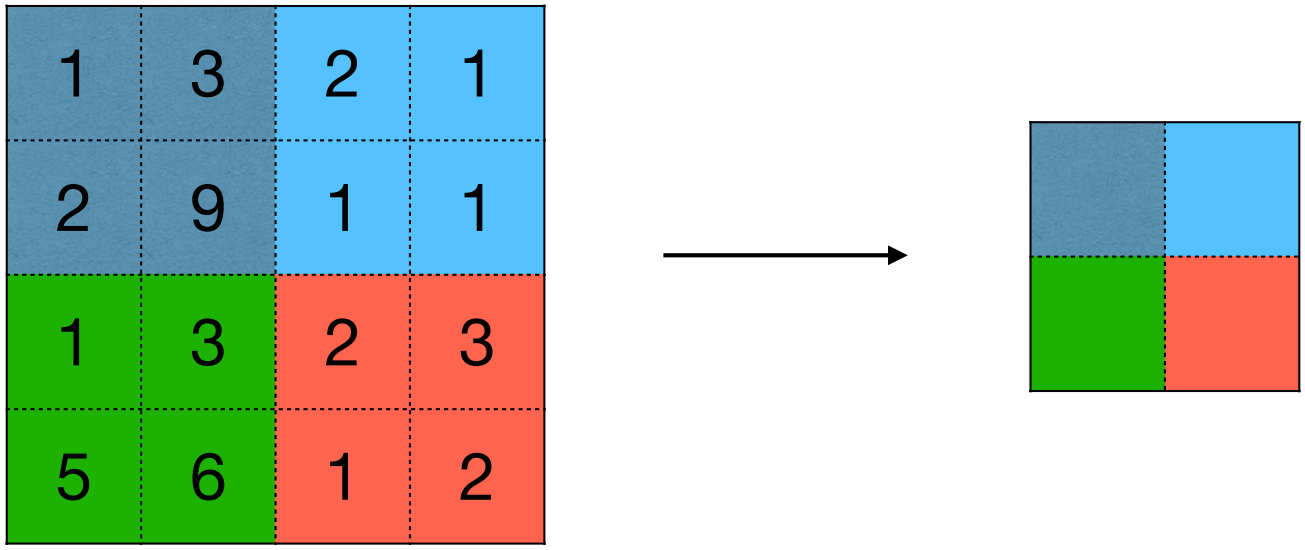
Pooling layers

- Max pooling



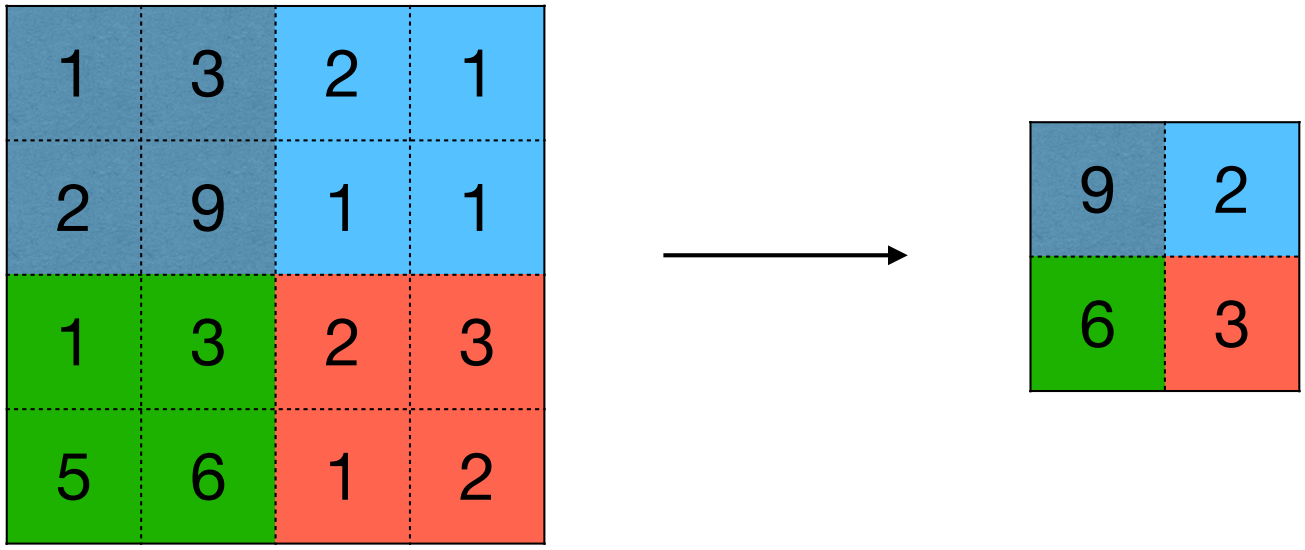
Pooling layers

- Max pooling



Pooling layers

- Max pooling



Pooling layers

- Max pooling ($f = 3, s = 1$)

1	3	2	1	3
2	9	1	1	5
1	3	2	3	2
8	3	5	1	0
5	6	1	2	9

9	9	5
9	9	5
8	6	9

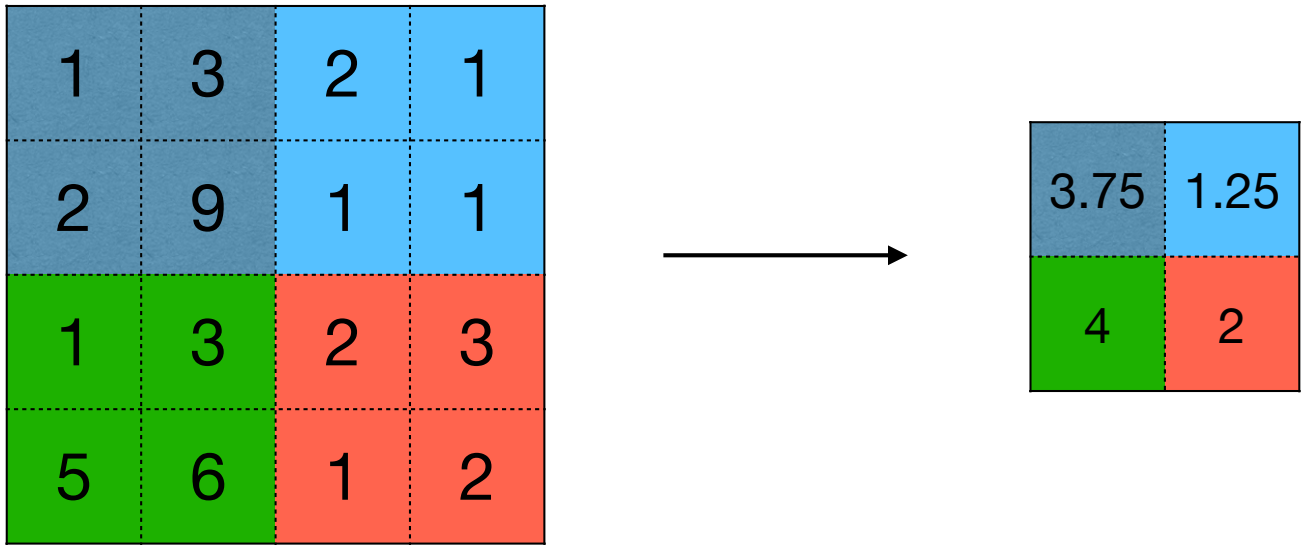
Pooling layers

- Average pooling ($f = 2, s = 2$)

1	3	2	1
2	9	1	1
1	3	2	3
5	6	1	2

Pooling layers

- Average pooling ($f = 2, s = 2$)



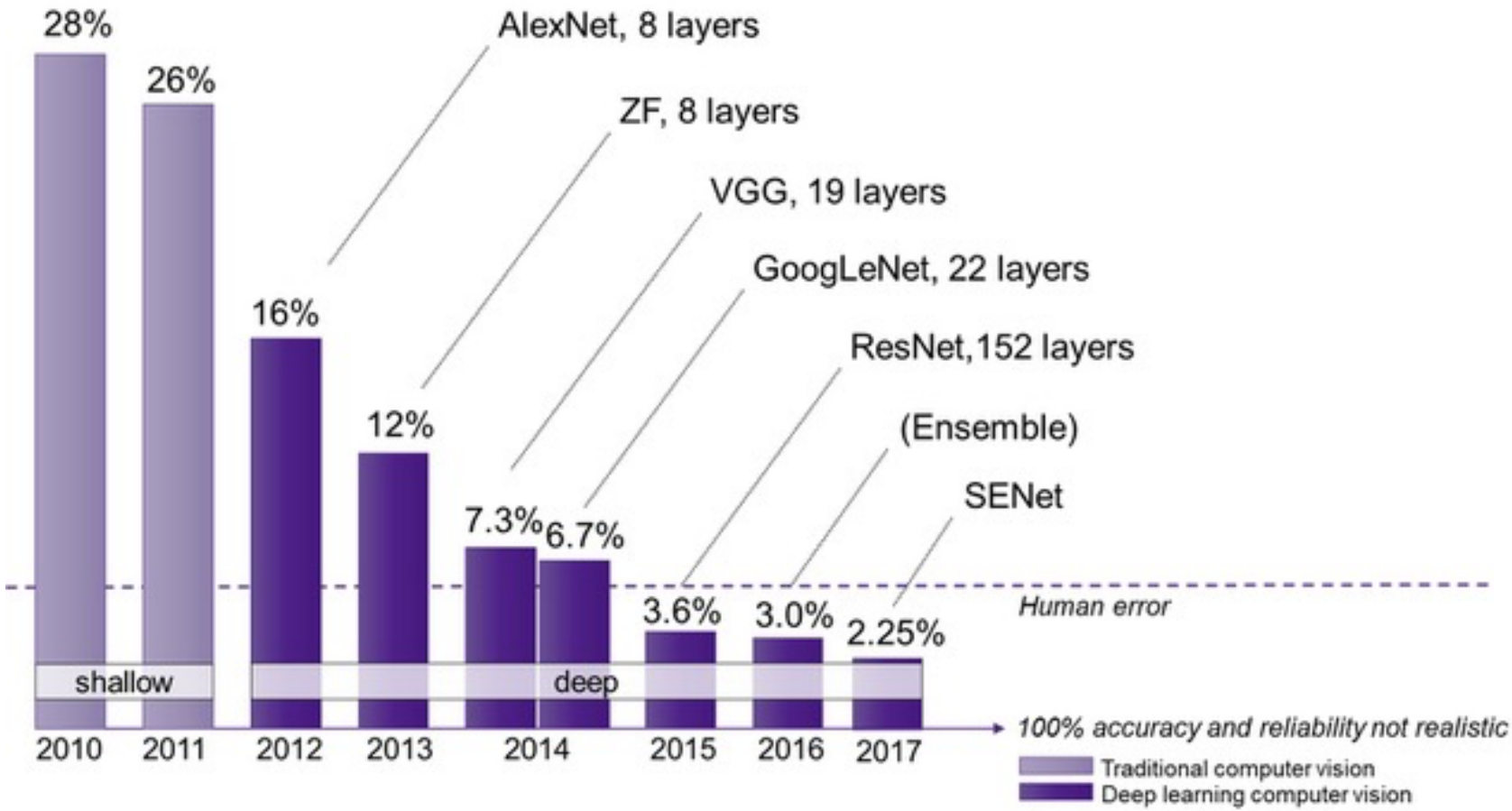
Why do convolutions work?

Convolutions work because of

- Parameter sharing: a feature detector (such as a vertical edge detector) that is useful is in one part of the image is probably useful in another part of the image

- Sparsity of connections: in each layer, each output value depends only on a small number of inputs

ImageNet Challenge (ILSVRC)



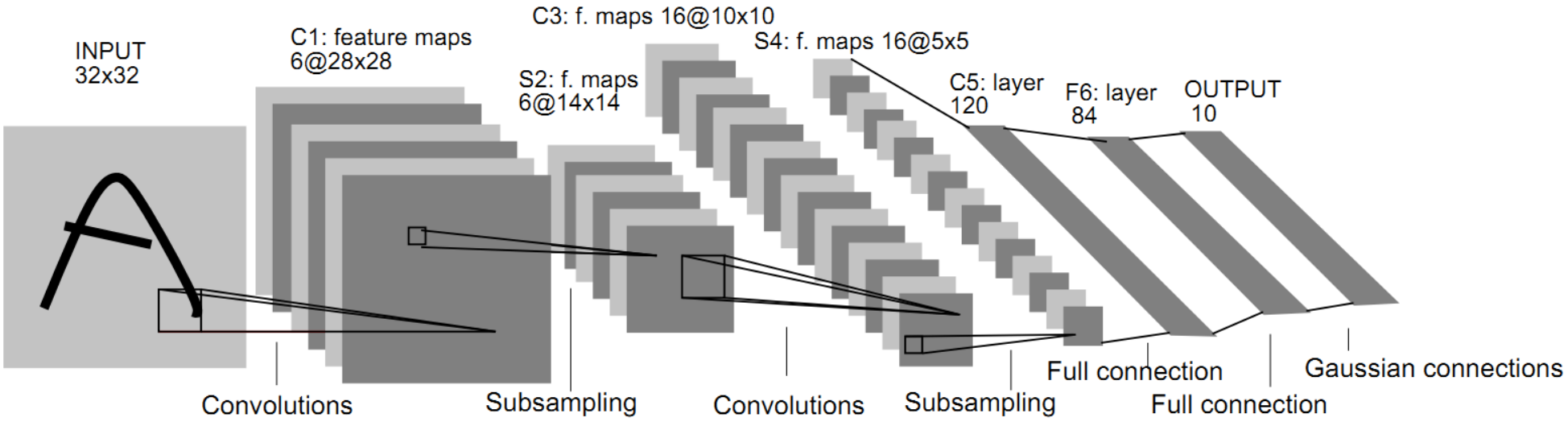
Example of CNNs

Classic networks:

- LeNet-5 (1998)
- AlexNet (2012)
- VGG (2015)
- ResNet (2015)
- Inception (2014)

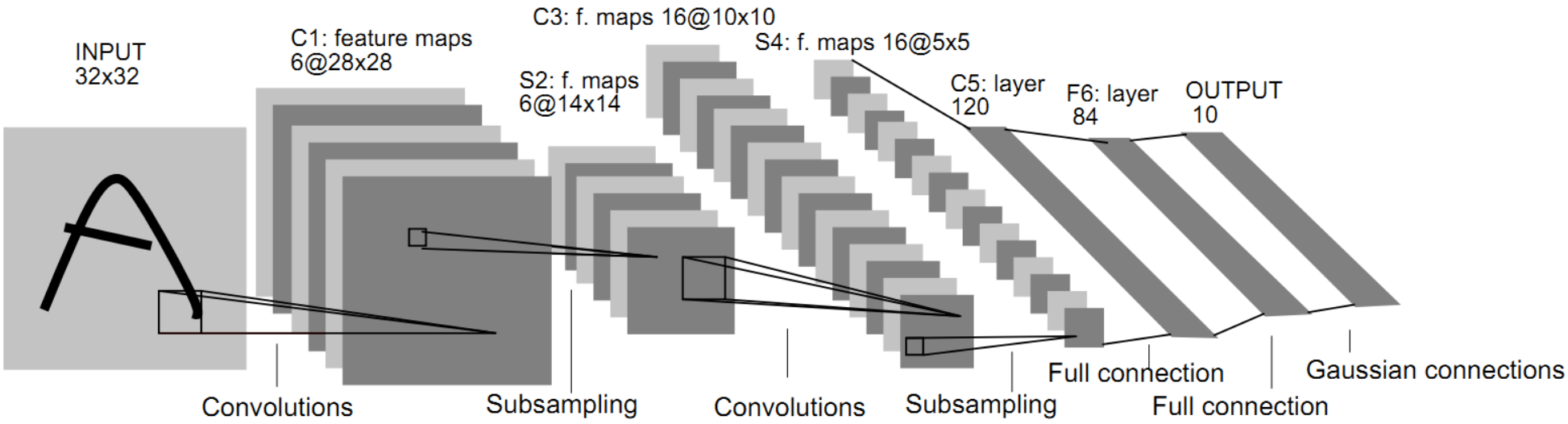
LeNet-5

- Neural network for recognizing digits



- CONV - POOL - CONV - POOL - FC - FC - FC - SOFTMAX

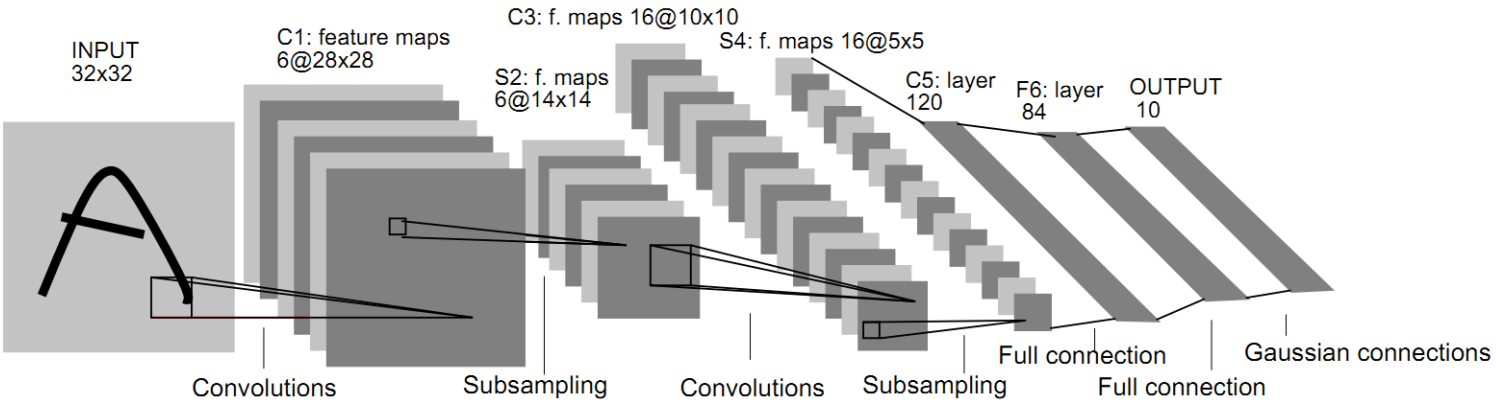
LeNet-5



- CONV($f = 5, s = 1$) - POOL ($f = 2, s = 2$)
- CONV($f = 5, s = 1$) - POOL ($f = 2, s = 2$)
- FC(120, 400), FC(84, 120), FC(10, 84)
- SOFTMAX

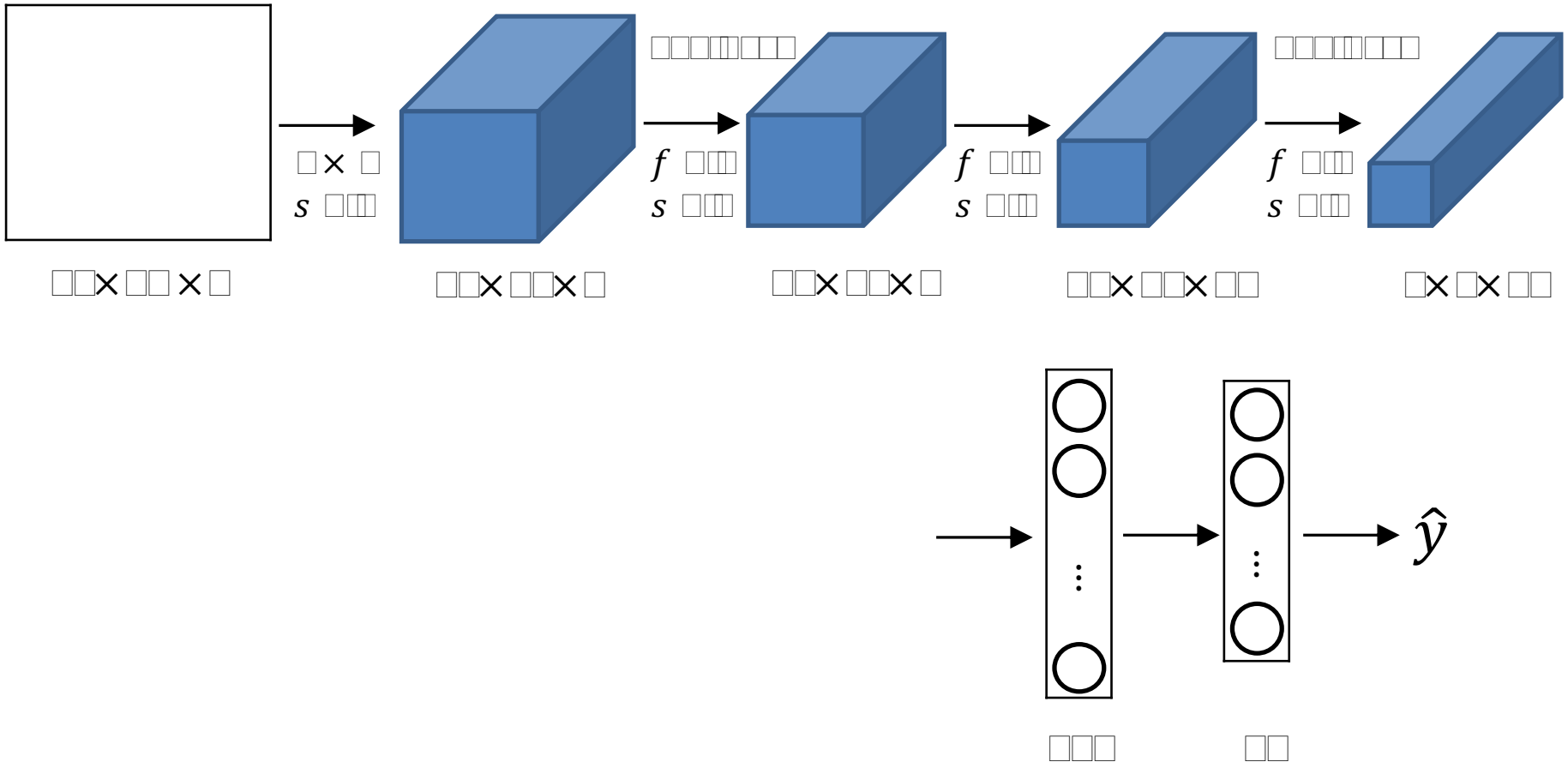
LeNet-5

	Activation	Activation size	# parameters
Input:	(32, 32, 3)	3,072	0
CONV1	(28, 28, 6)	4,704	456
POOL1	(14, 14, 6)	1,176	0
CONV2	(10, 10, 16)	1,600	2416
POOL2	(5, 5, 16)	400	0
FC3	(120, 1)	120	48,120
FC4	(84,1)	84	10,164
SOFTMAX	(10,1)	10	850

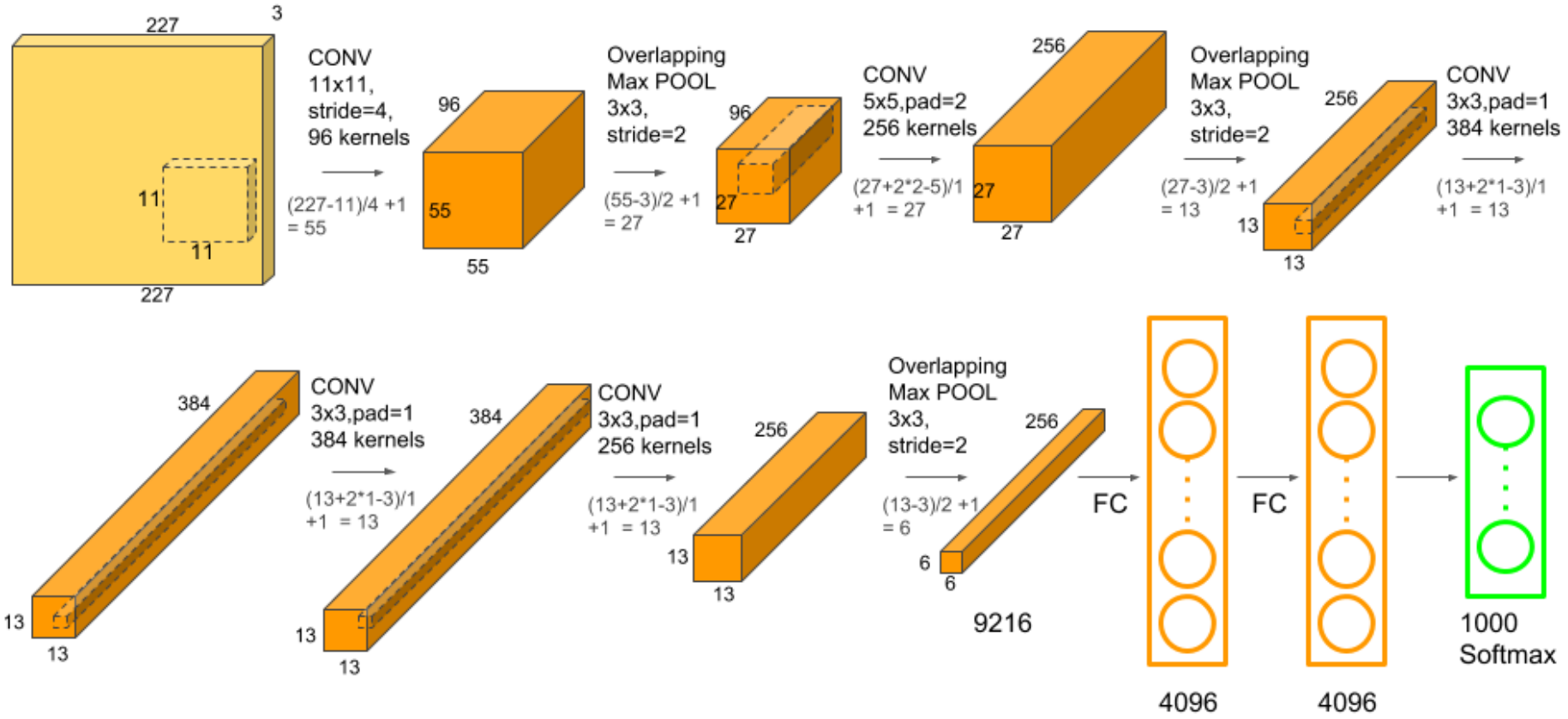


LeNet-5 (1998)

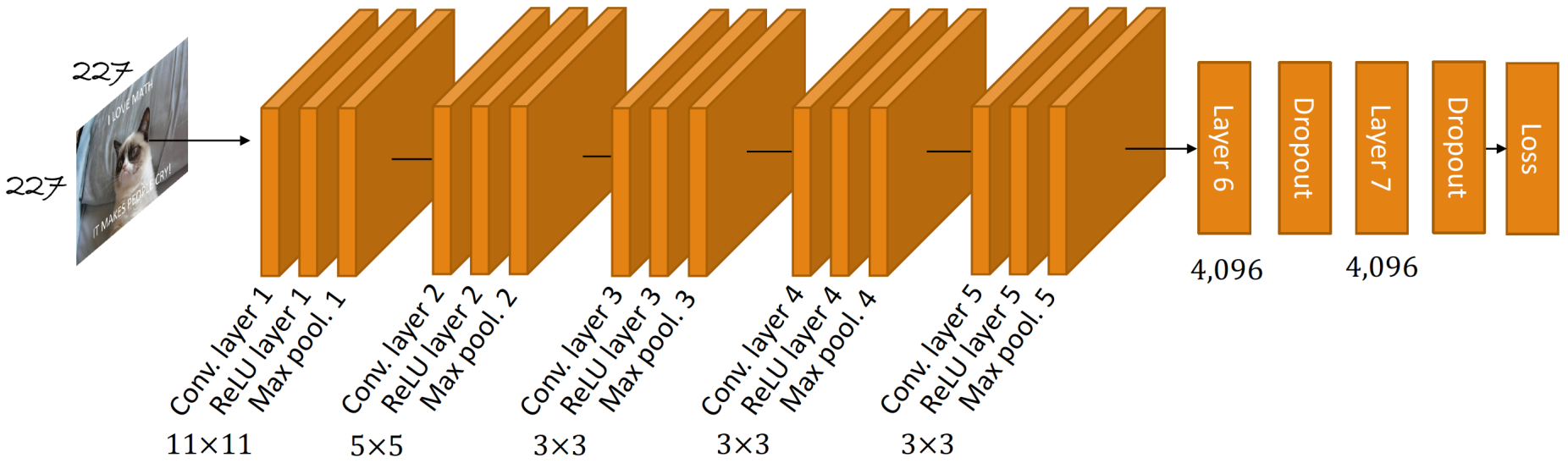
- Original network (LeCun et al., 1998)



AlexNet (2012)

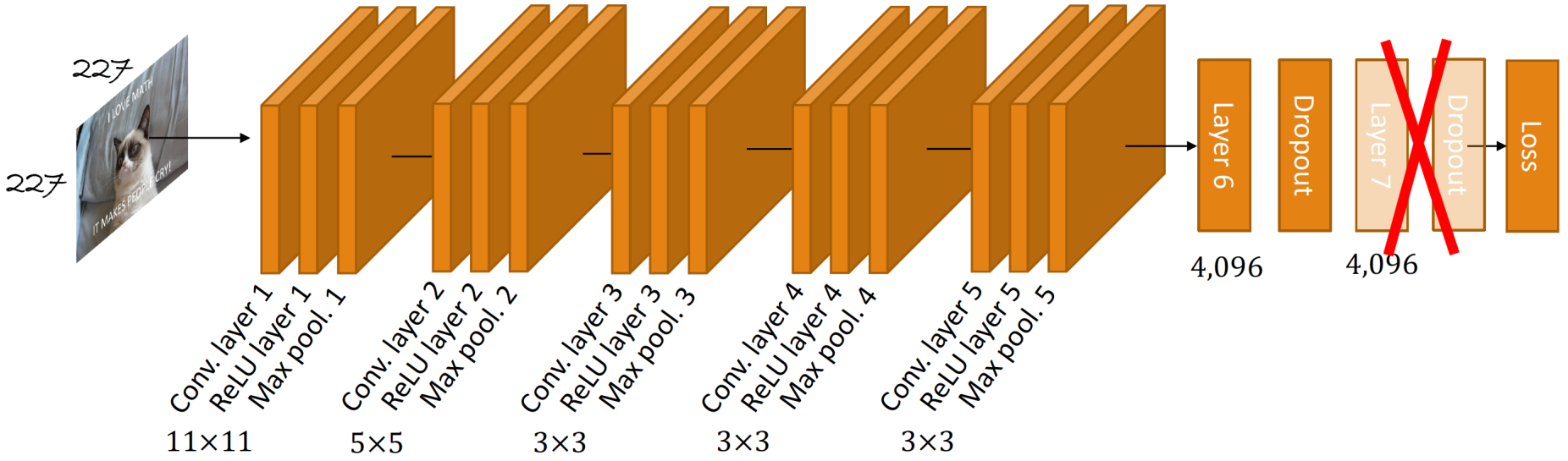


AlexNet (2012)



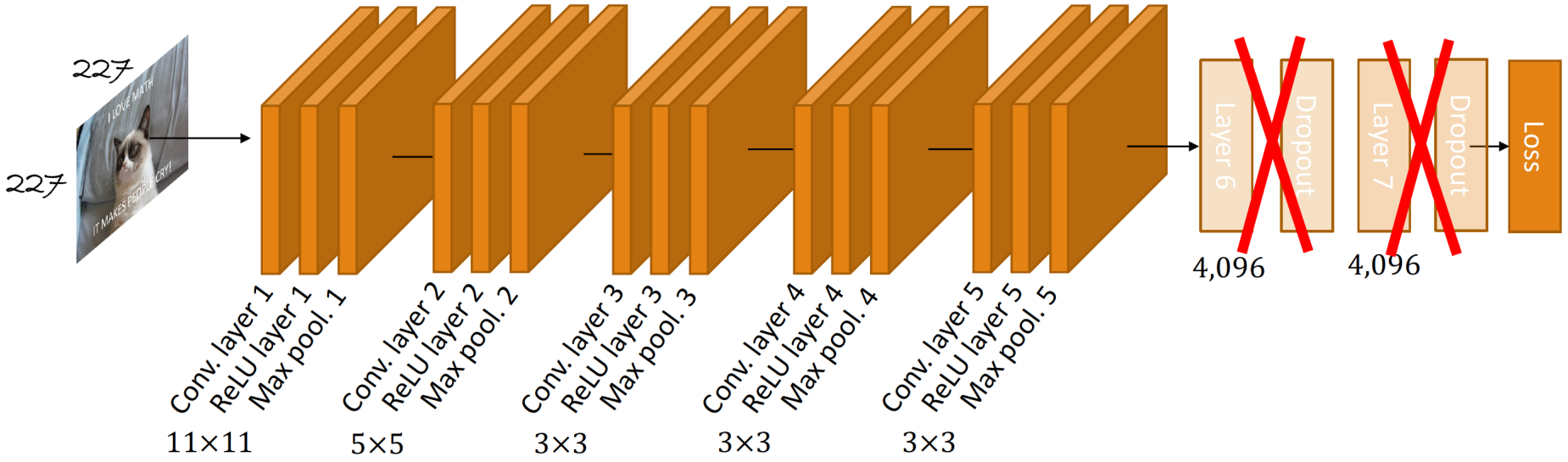
AlexNet (2012)

1.1% drop in performance, 16 million fewer parameters



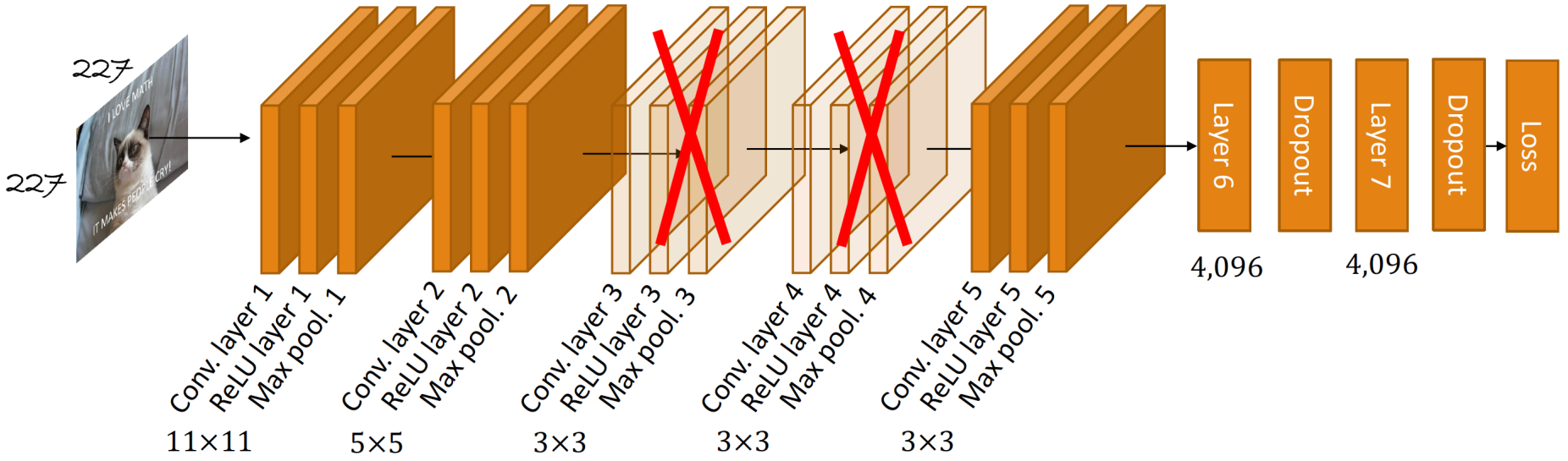
AlexNet (2012)

5.7% drop in performance, 50 million fewer parameters



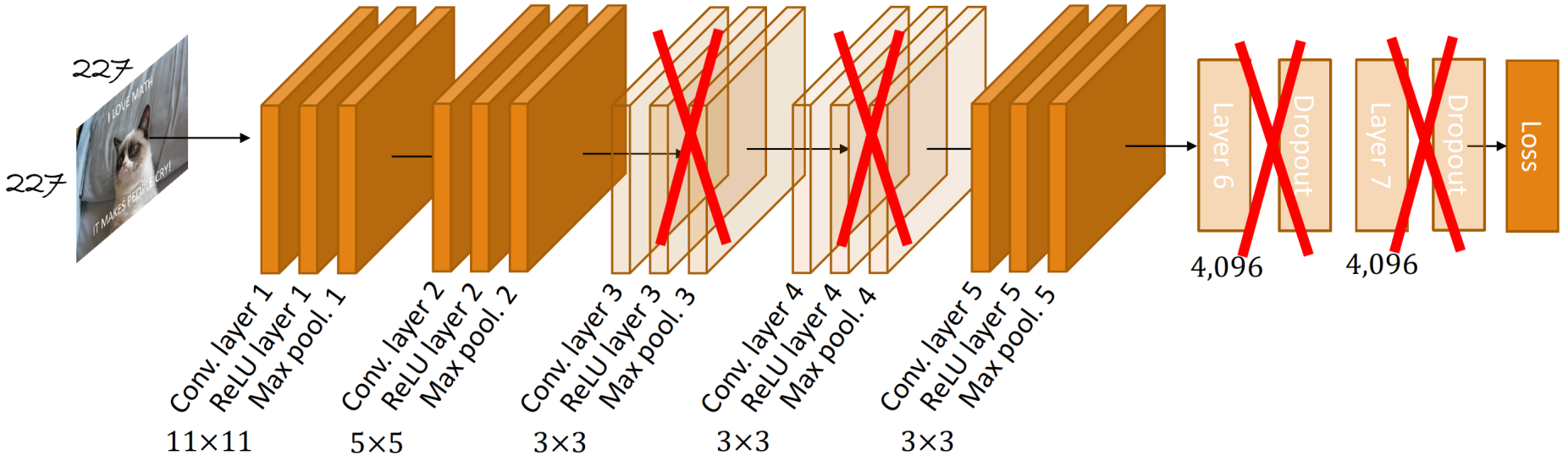
AlexNet (2012)

3.0% drop in performance, 1 million fewer parameters

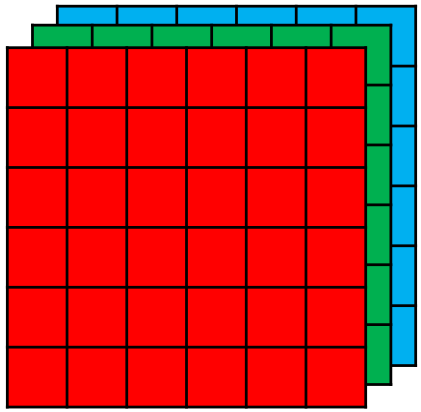


AlexNet (2012)

33.5% drop in performance



VGG (2015)



[CONV 64] ×2

POOL

[CONV 128] ×2

POOL

[CONV 256] ×3

POOL

[CONV 512] ×3

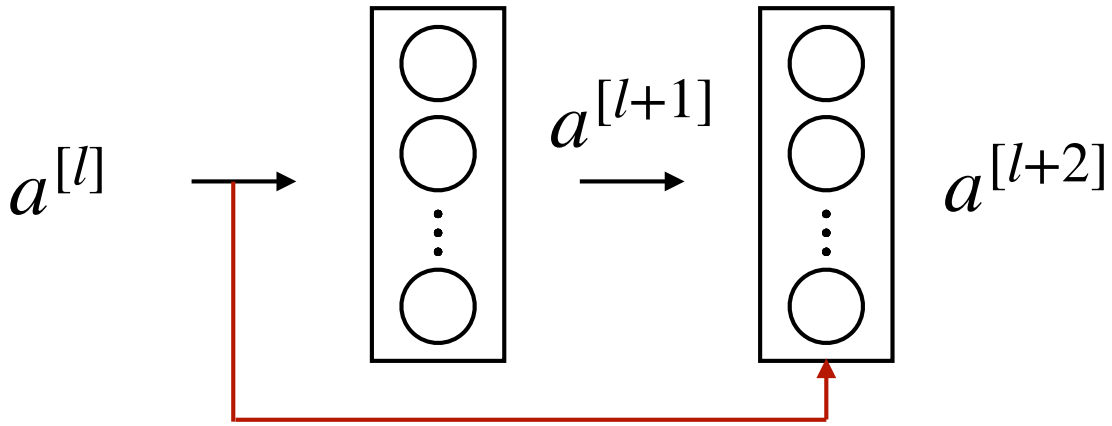
POOL

[CONV 512] ×3

POOL

ResNet (2015)

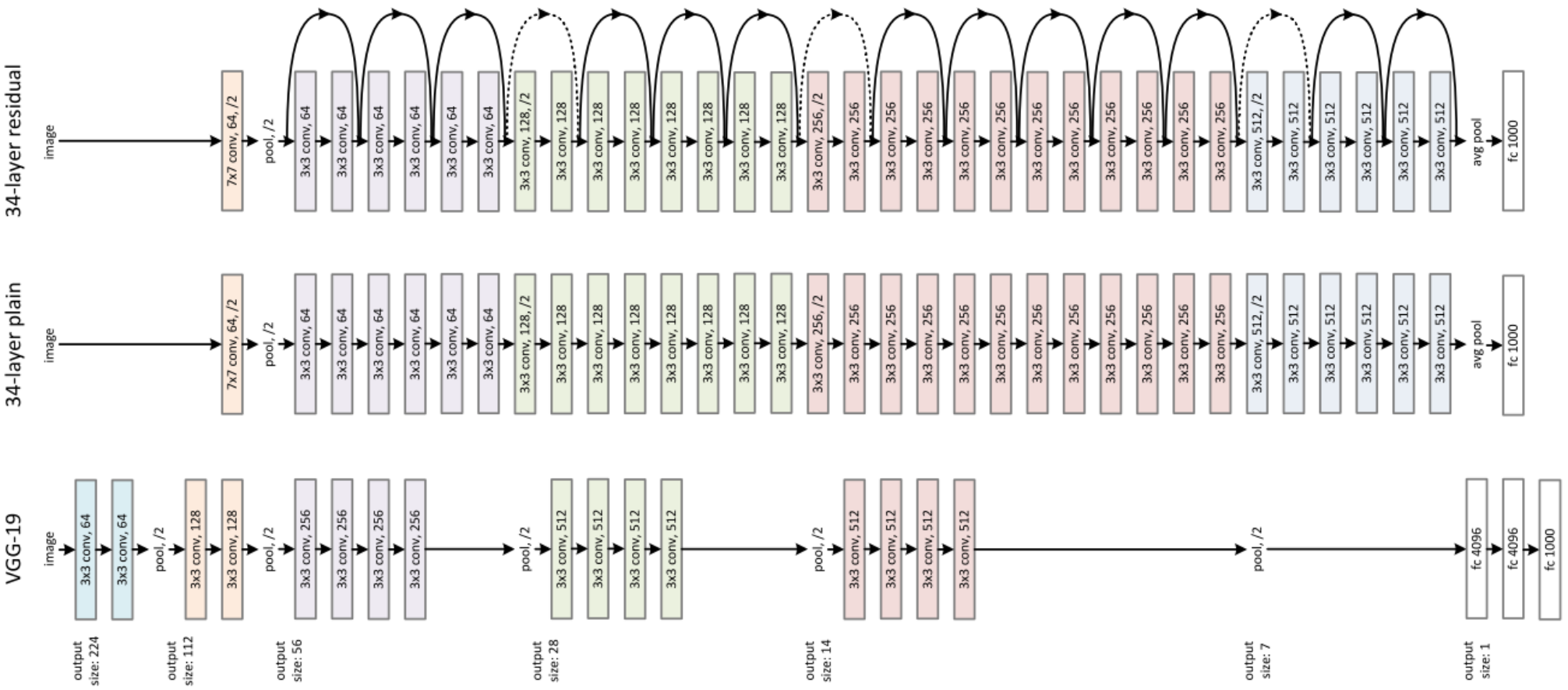
- Residual block



- $z^{[l+1]} = W^{[l+1]}a^{[l]} + b^{[l+1]}$
- $a^{[l+1]} = g(z^{[l+1]})$
- $z^{[l+2]} = W^{[l+2]}a^{[l+1]} + b^{[l+2]}$
- ~~$a^{[l+2]} = g(z^{[l+2]})$~~ $\rightarrow$ $a^{[l+2]} = g(z^{[l+2]} + a^{[l]})$

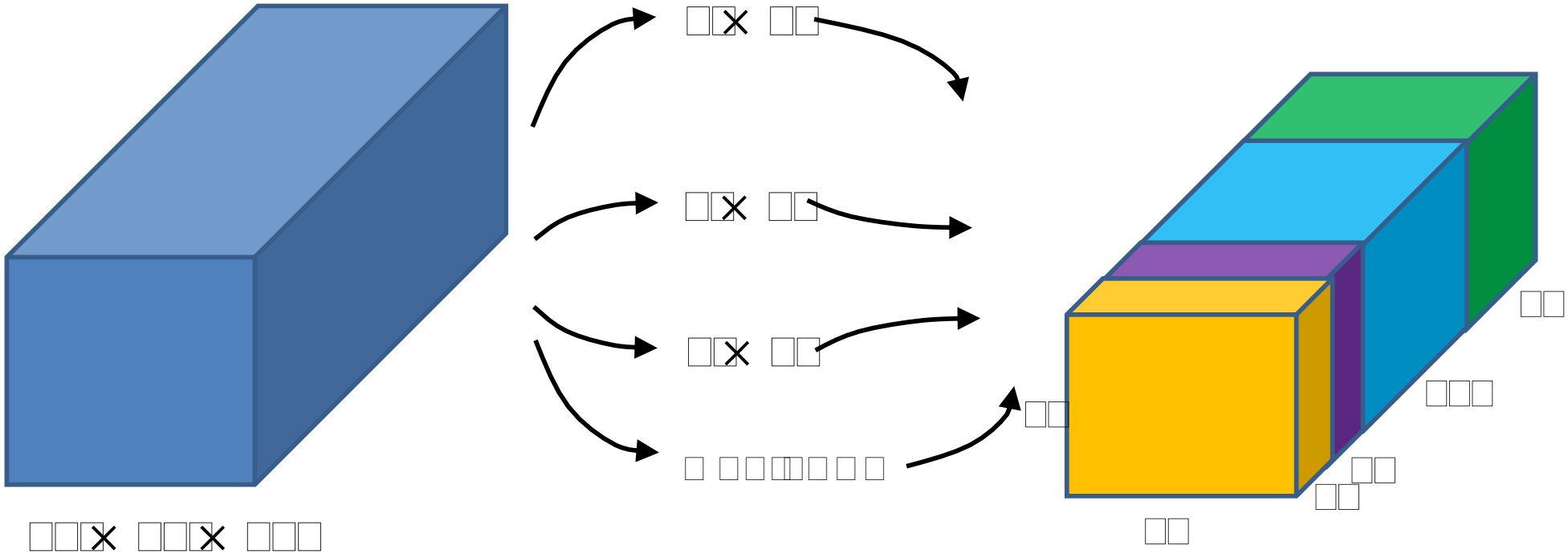
ResNet (2015)

- Example



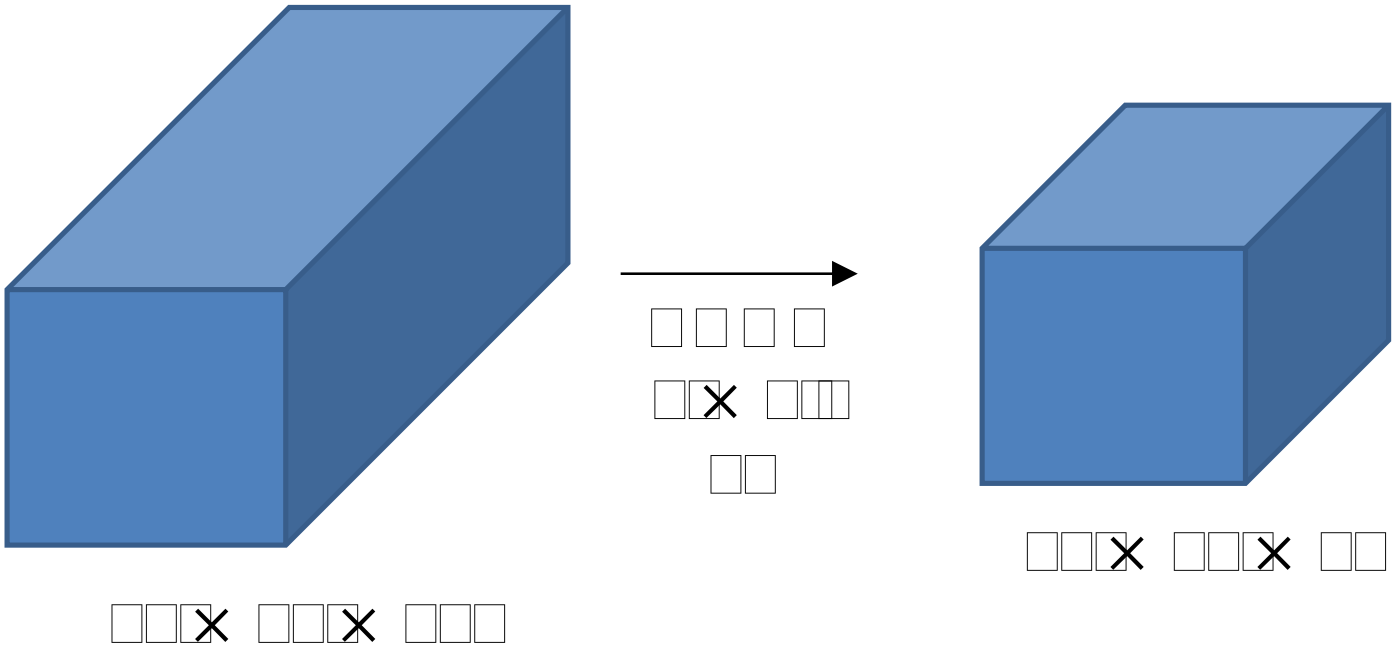
Inception networks (2014)

- Motivation



Inception networks (2014)

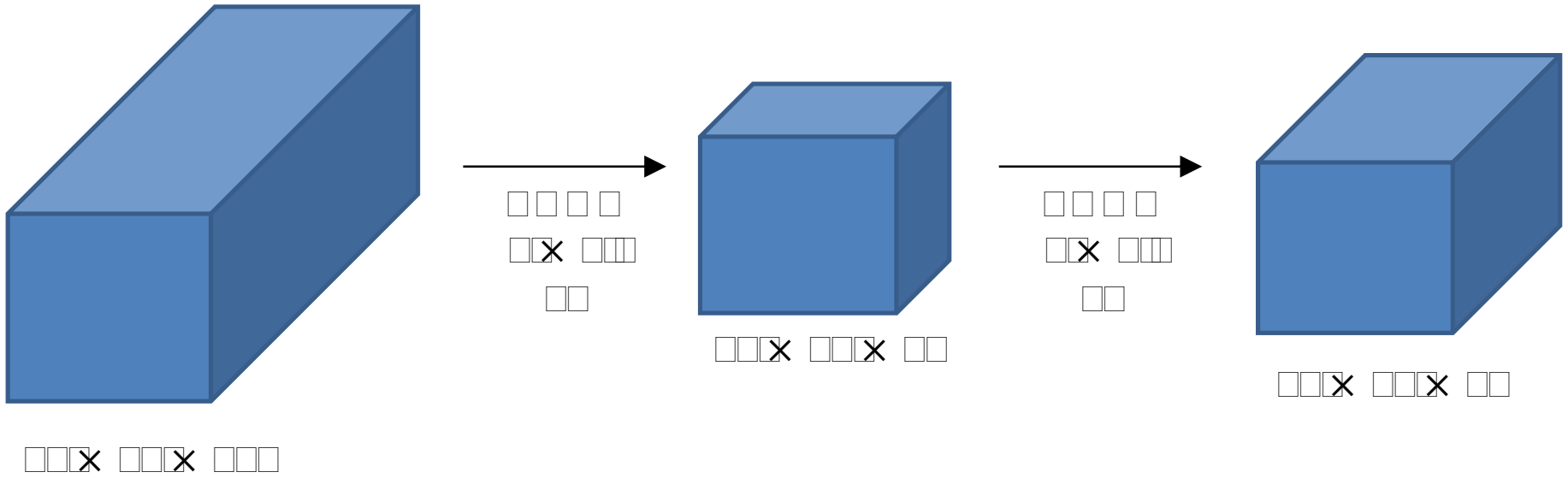
- Computational problems



- # multiplications: (28 x 28 x 32) x (5 x 5 x 192) = 120 M

Inception networks (2014)

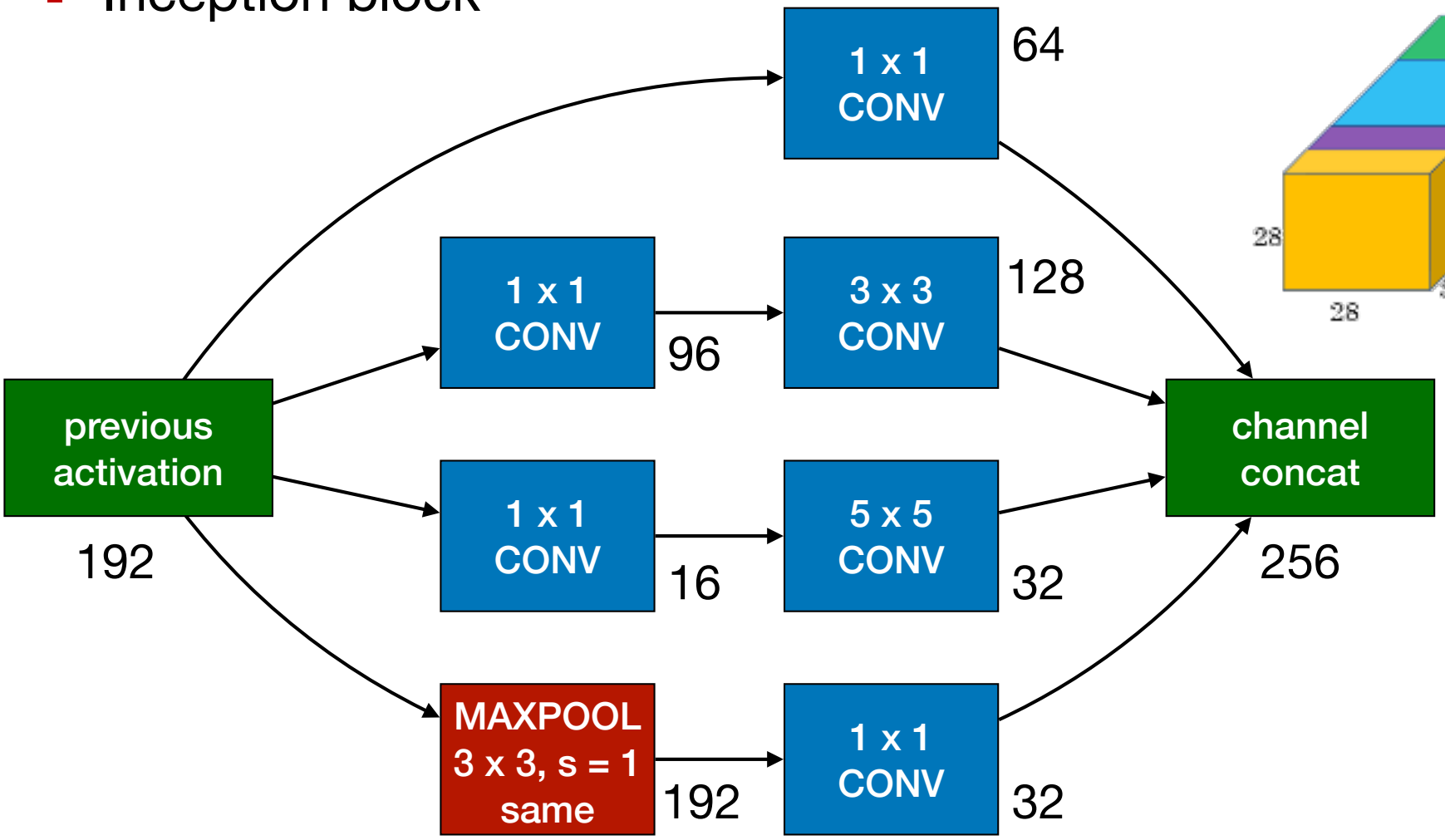
- Computational problems



- # multiplications: (28 x 28 x 16) x (1 x 1 x 192) = 2.4 M
  (28 x 28 x 32) x (5 x 5 x 16) = 10 M

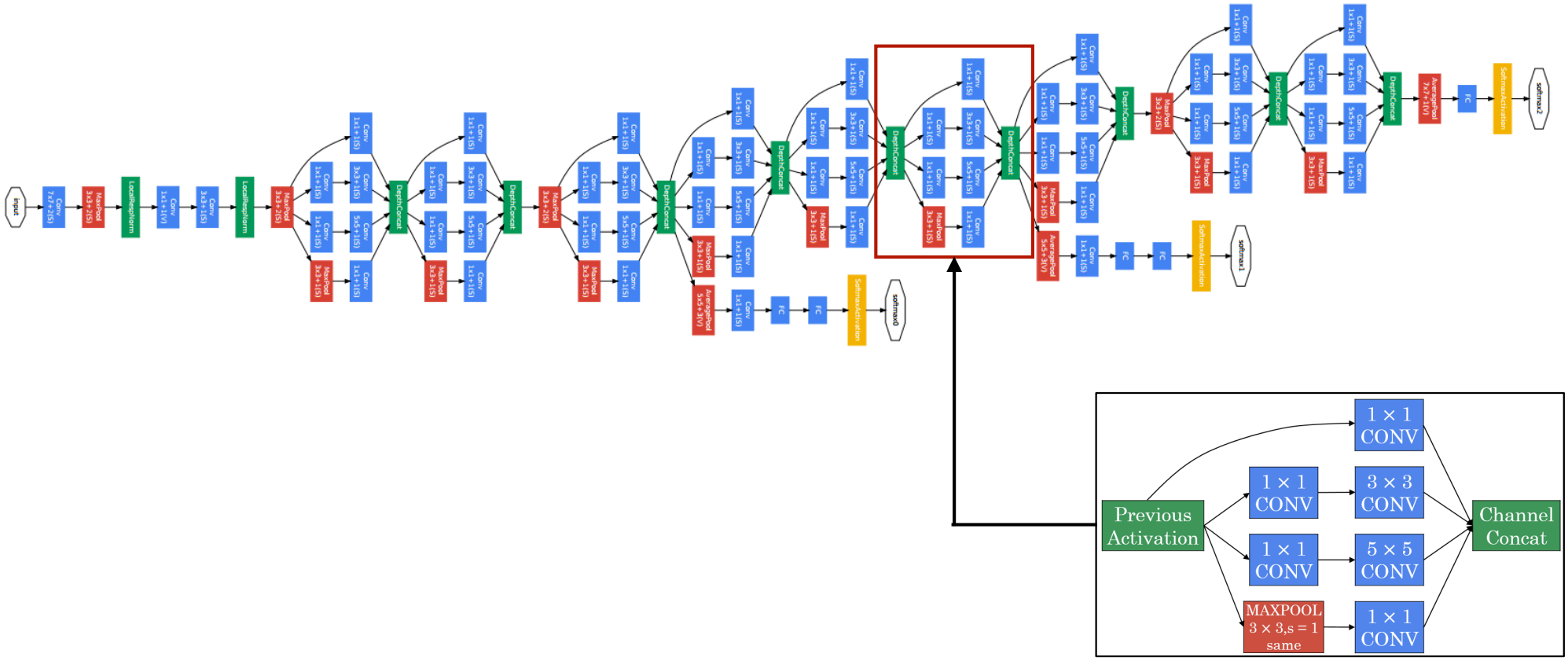
Inception networks (2014)

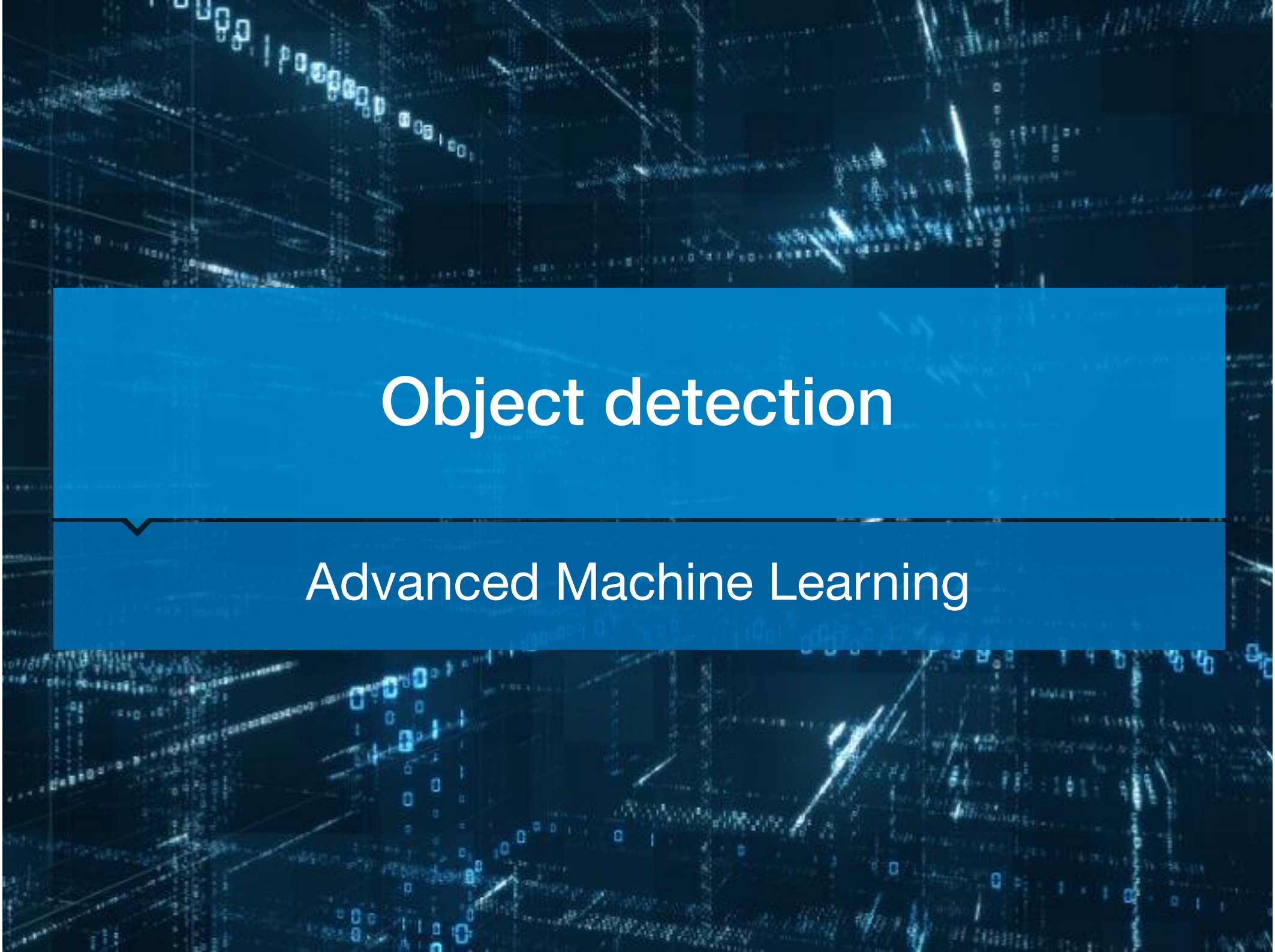
- Inception block



Inception networks (2014)

- GoogLeNet





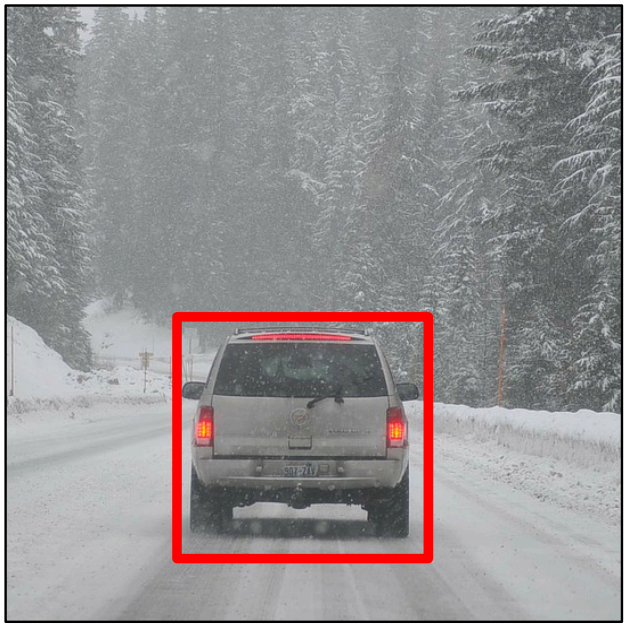
Object detection

Advanced Machine Learning

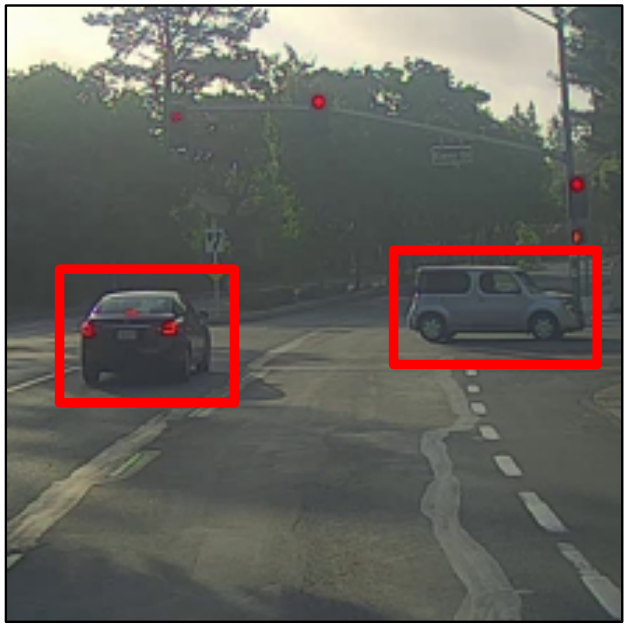
Object localization and detection

Image classification

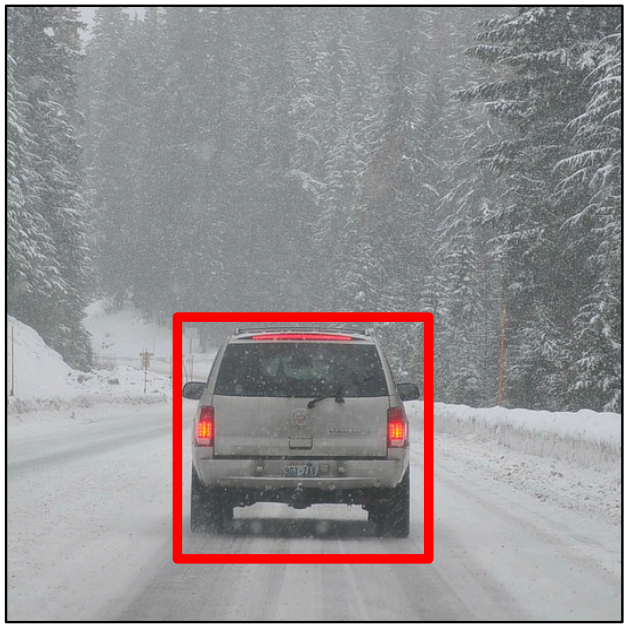
Classification with localization

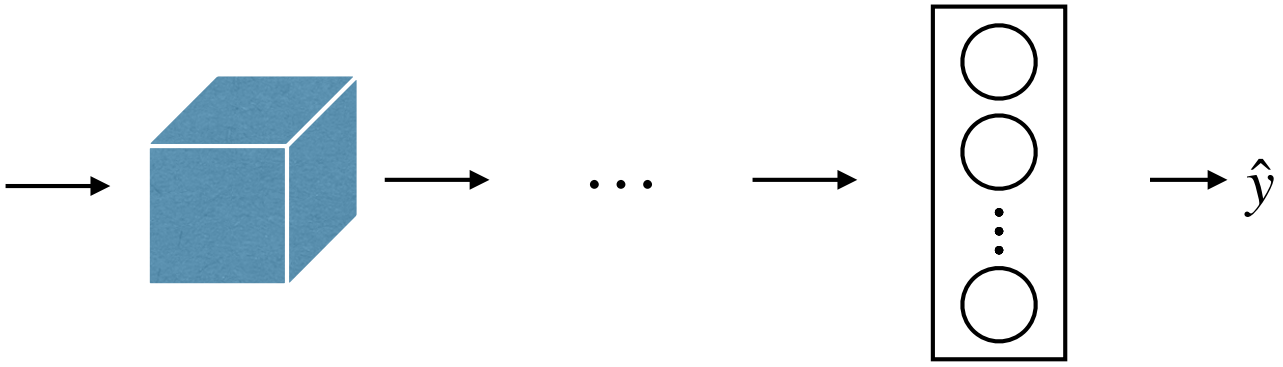


Detect



Object localization





1 - pedestrian

2 - car

3 - motorcycle

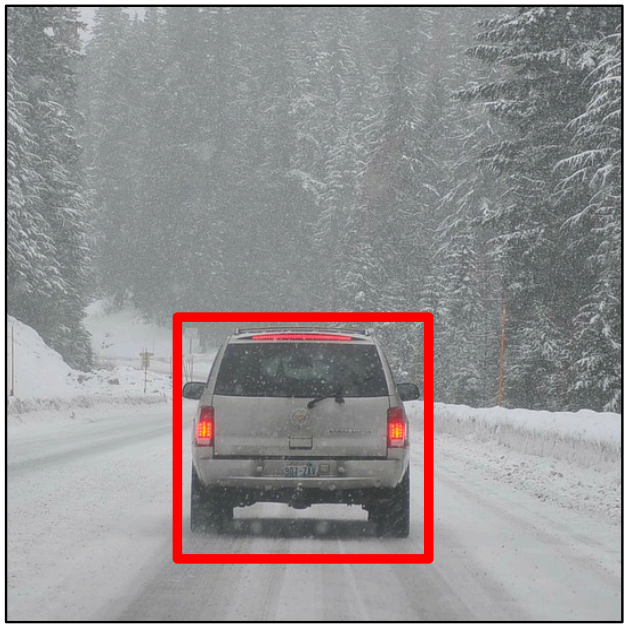
4 - background

Add the prediction to the bounding box

$$(b_x, b_y, b_h, b_w)$$

Example: $(b_x, b_y, b_h, b_w) = (0.5, 0.7, 0.3, 0.4)$

Object localization



$(1, b_x, b_y, b_h, b_w, 0, 1, 0)$

$(0, ?, ?, ?, ?, ?, ?, ?)$

1 - pedestrian

2 - car

3 - motorcycle

4 - background

Adjust the cost function appropriately:

$$(y_1 - \hat{y}_1)^2 + \cdots + (y_8 - \hat{y}_8)^2 \text{ if } y_1 = 1$$

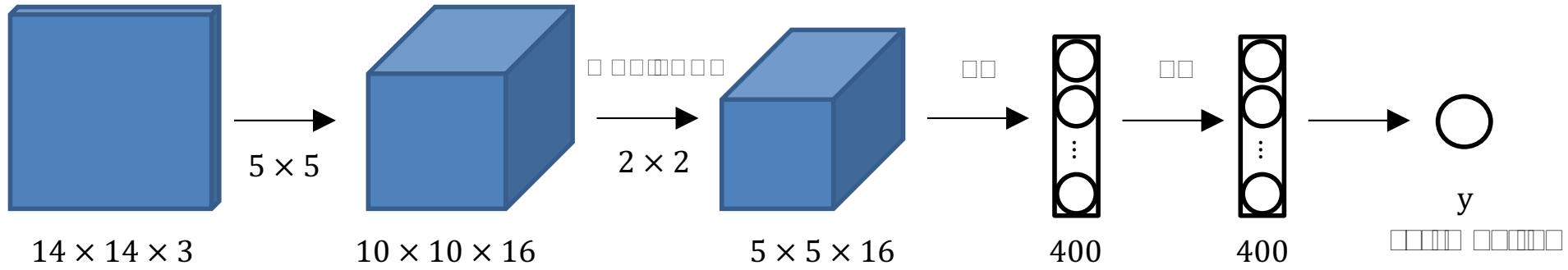
$$(y_1 - \hat{y}_1)^2 \text{ if } y_1 = 0$$

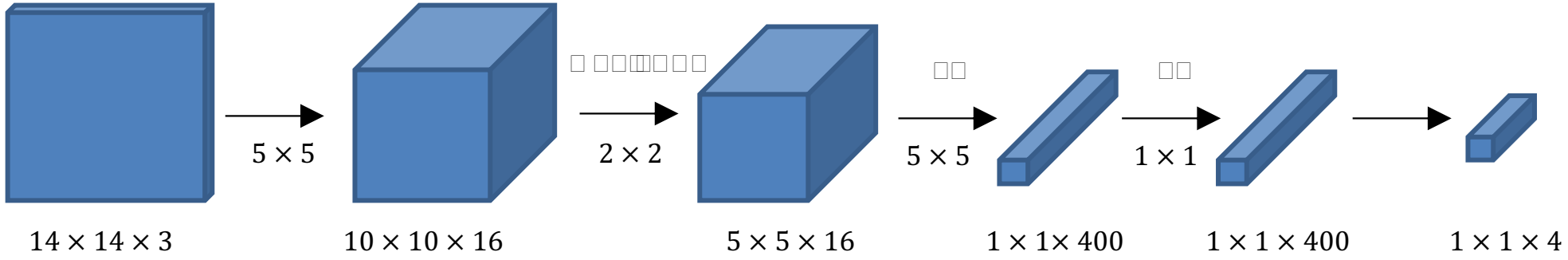
Object detection



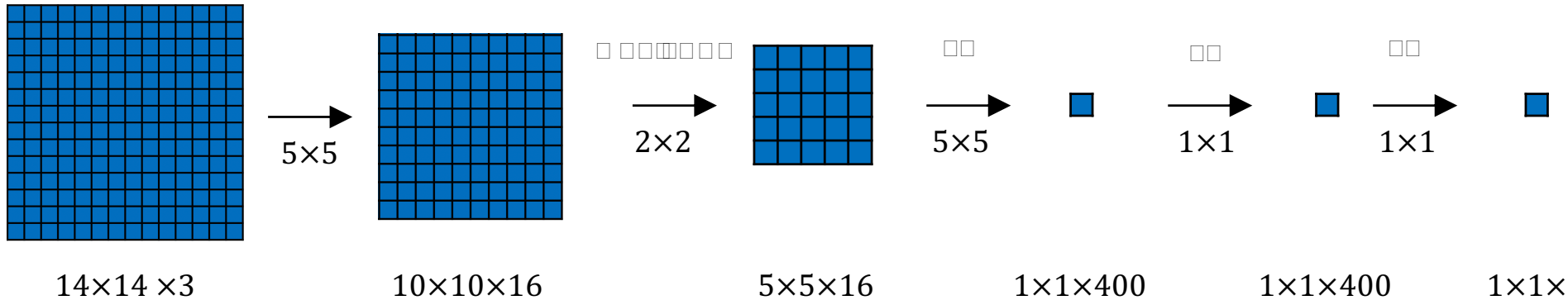
Assume that a ConvNet has been trained for object localization

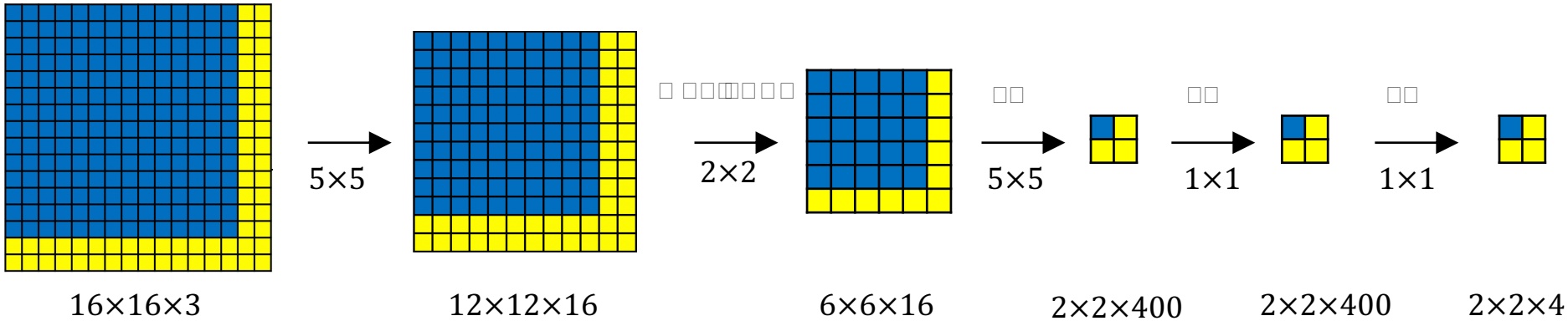
Object detection



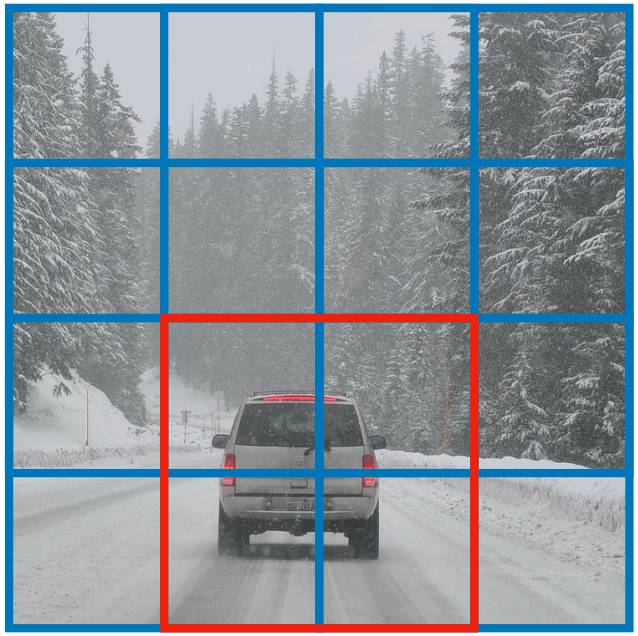


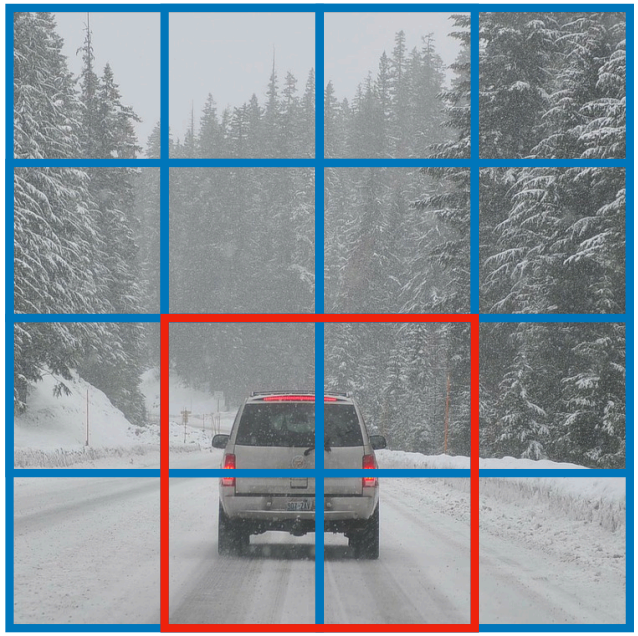
Object detection





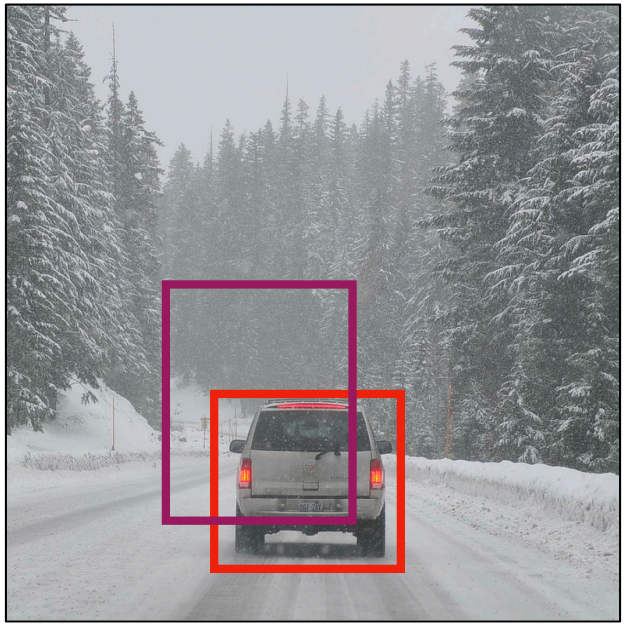
Object detection





Object detection

- How to evaluate your algorithm?
  - Intersection of Union



Object detection

- How to get unique detections?
  - > Non-max suppression algorithm

Object detection

- How to get unique detections?
  - > Non-max suppression algorithm

Object detection

- How to get unique detections?
  - > Non-max suppression algorithm

Object detection

- How to get unique detections?
  - > Non-max suppression algorithm

Object detection

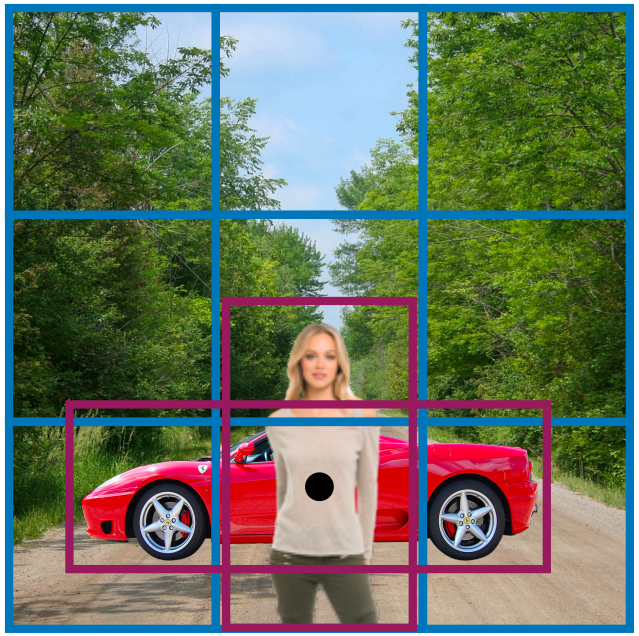
- How to get unique detections?
  - > Non-max suppression algorithm

Object detection

- How to get unique detections?
  - > Non-max suppression algorithm

Object detection

- How to detect overlapping objects?
  - > Anchor box algorithm

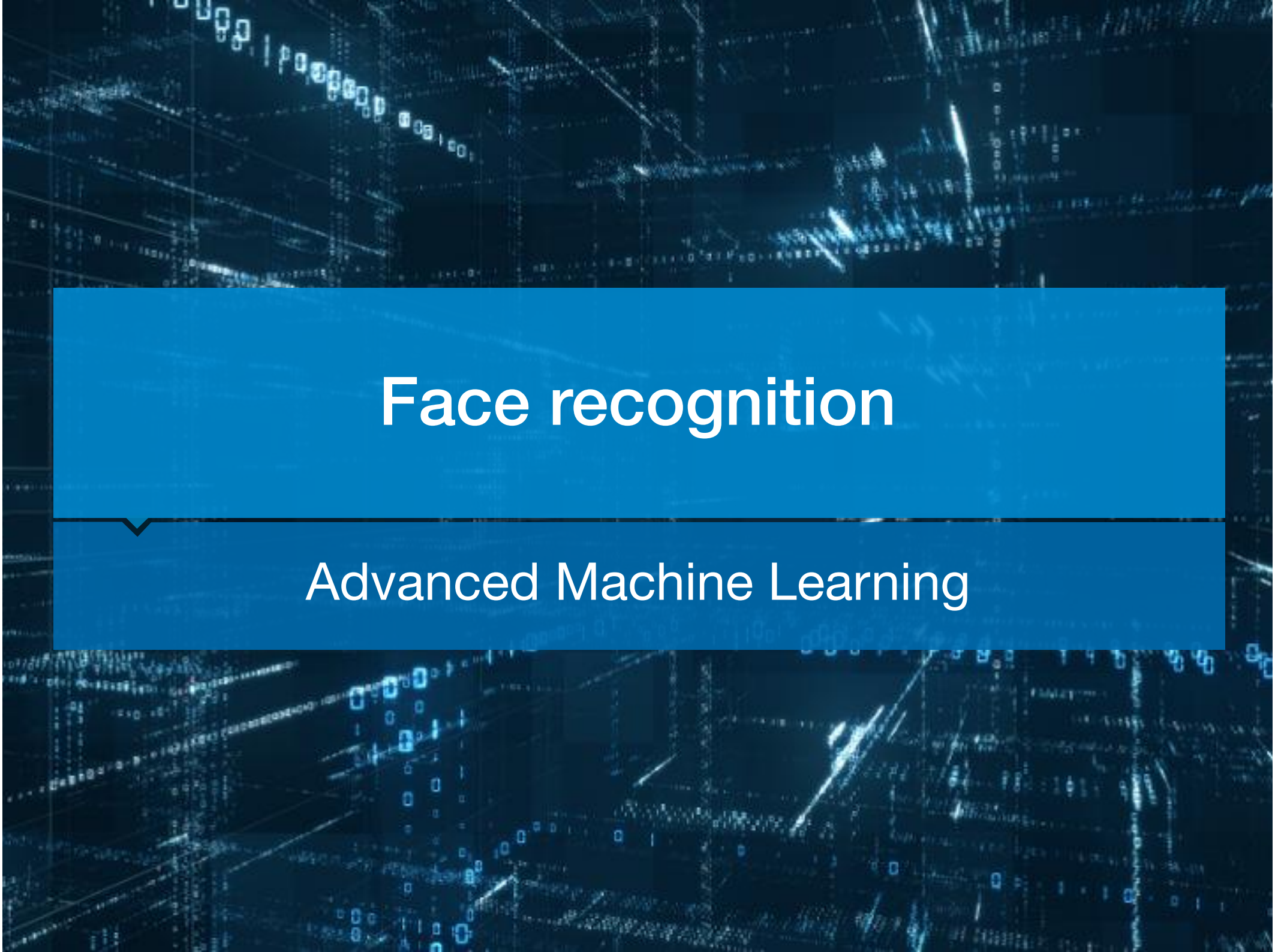


Previous response:

$$y = (p_c, b_x, b_y, b_h, b_w, c_1, c_2, c_3)$$

Anchor box response:

$$y' = (y, y, \ldots)$$



Face recognition

Advanced Machine Learning

Face verification versus recognition

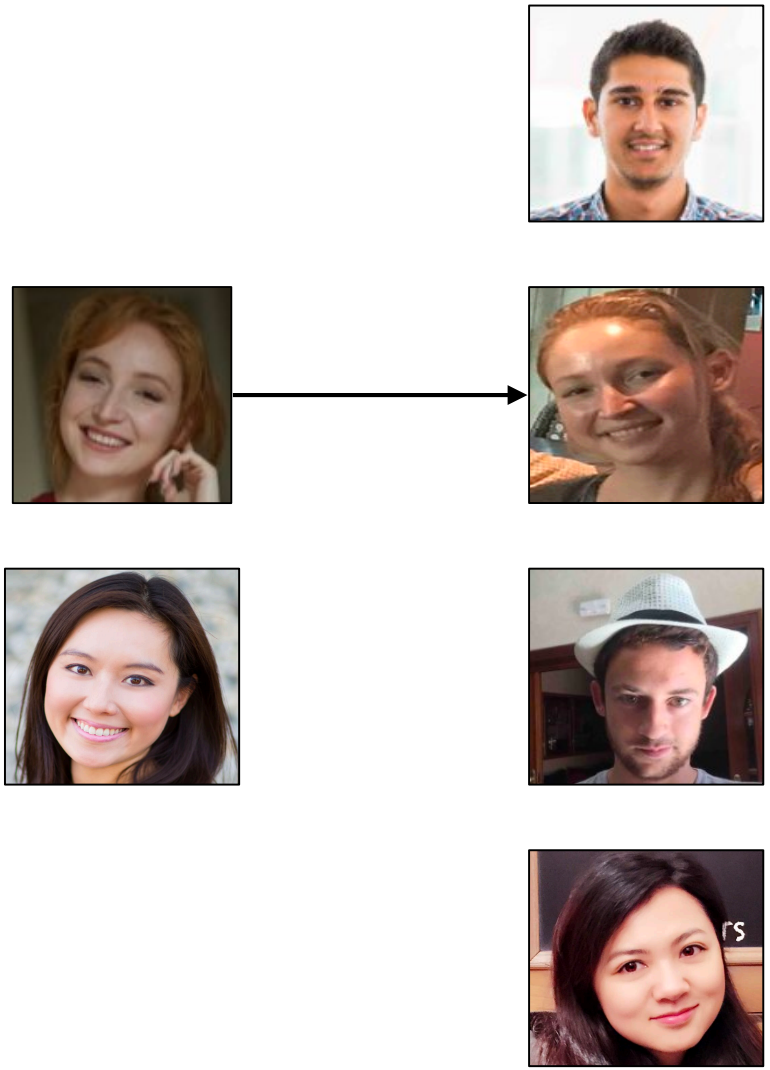
- Verification
  - Input image, name / ID
  - Output whether the input image is that of the claimed person

- Recognition
  - Has a database of K persons
  - Get an input image
  - Output ID if the image is any of the K persons (or “not recognized”)

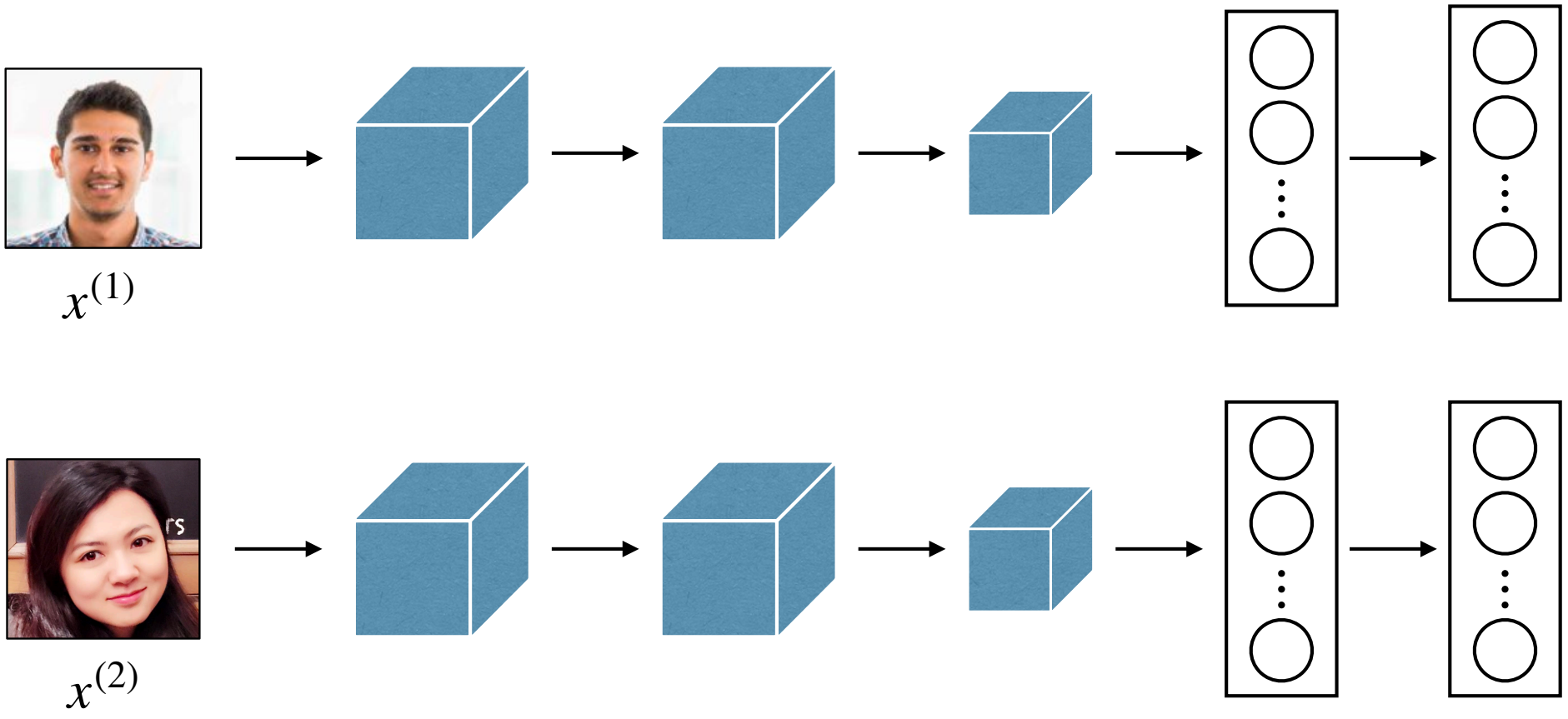
Face verification versus recognition

Problem: one-shot learning

Solution: learn a “similarity” function



Siamese network (2014)



Parameters of NN define an encoding $f(x^{(i)})$

Siamese network (2014)

Anchor

Positive

Anchor

Negative

$$d(A, P) \qquad\qquad d(A, N)$$

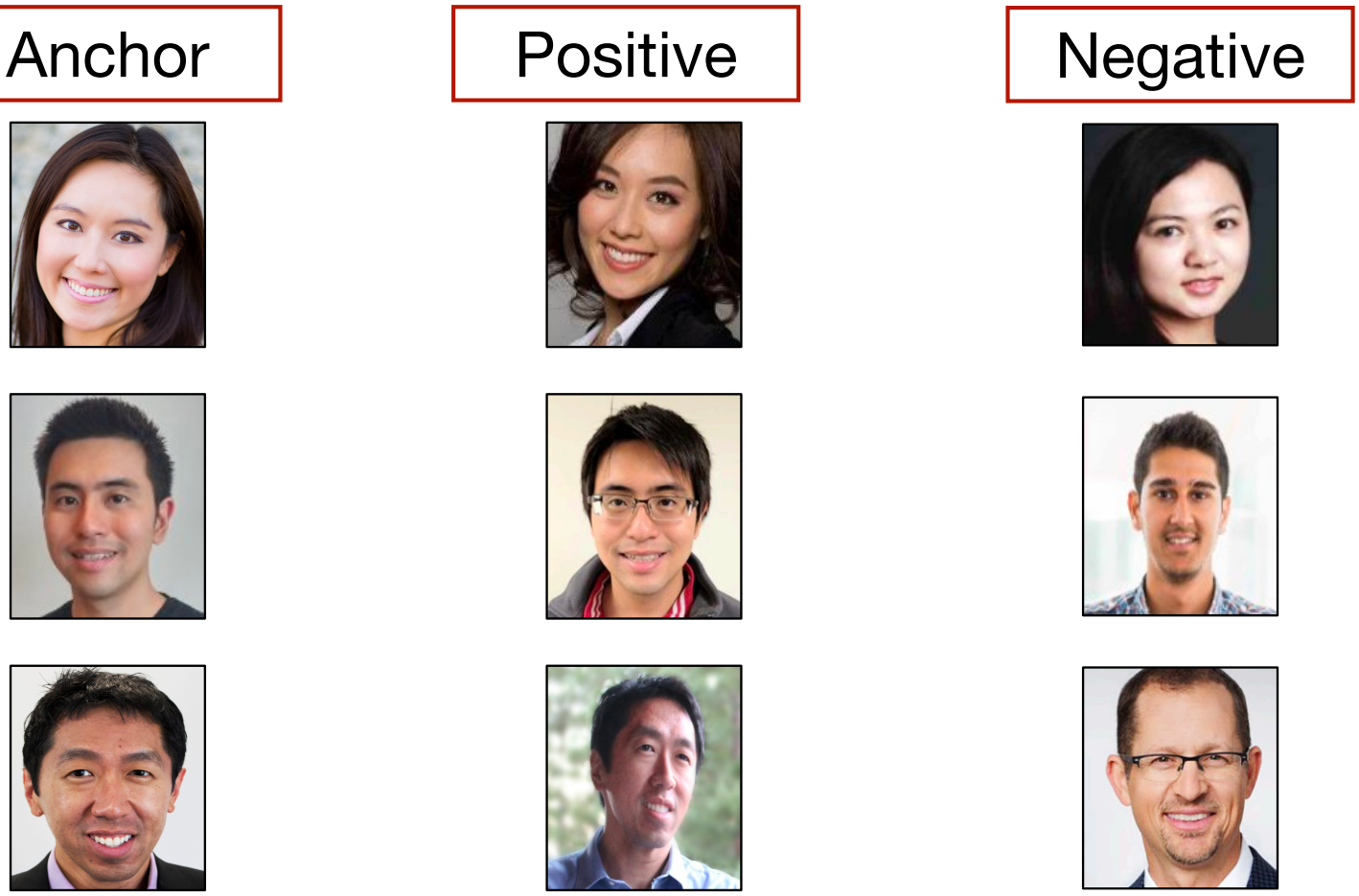
$$\|f(A) - f(P)\|^2 = d(A, P) \leq d(A, N) = \|f(A) - f(N)\|^2$$

$$\downarrow$$

$$d(A, P) + \alpha \leq d(A, N)$$

VU

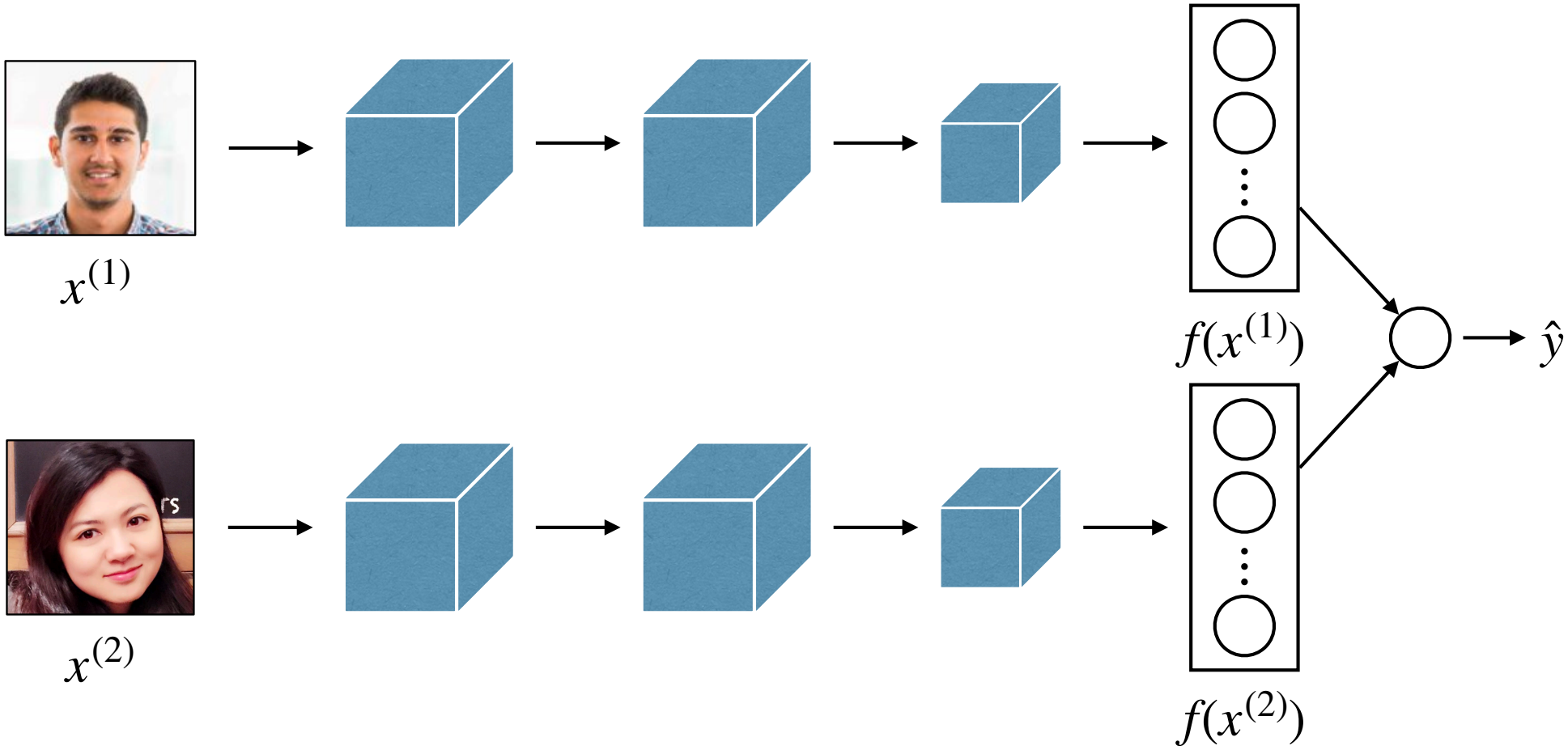
Siamese network (2014)

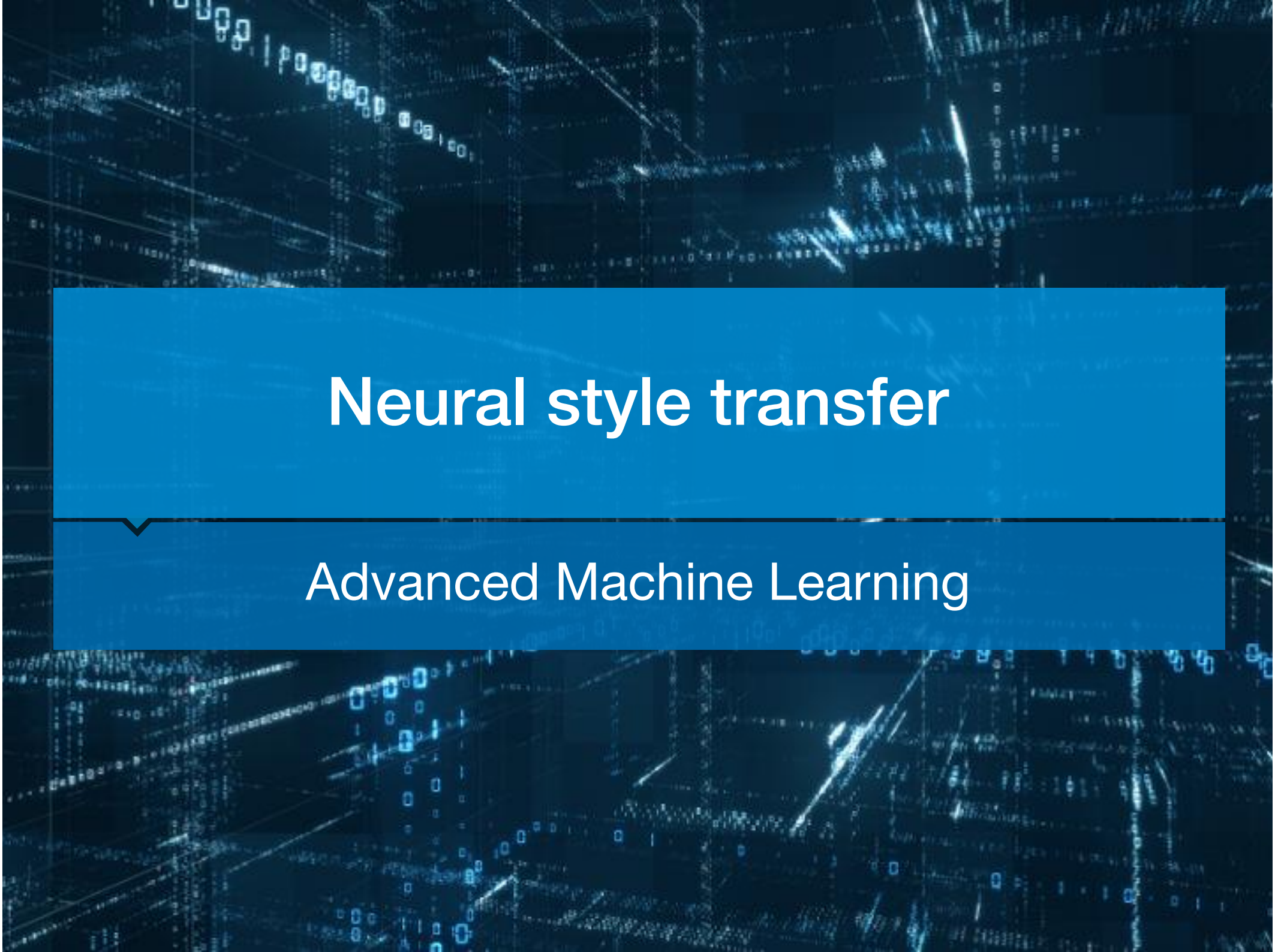


Triplet loss function

$$\mathscr{L}(A,P,N) = \max(\|f(A) - f(P)\|^2 - \|f(A) - f(N)\|^2 + \alpha, 0)$$

Siamese network (2014)

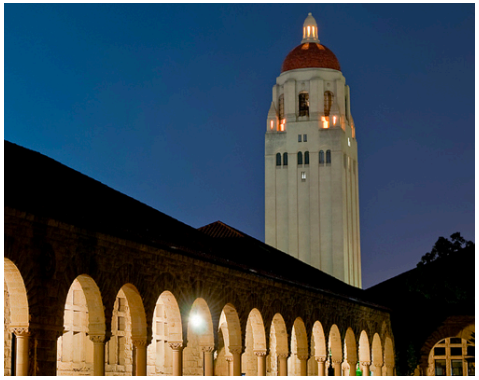




Neural style transfer

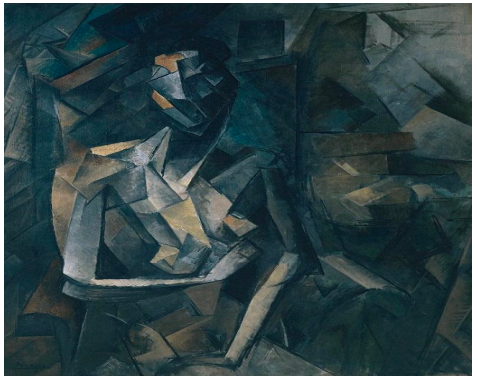
Advanced Machine Learning

Neural style transfer




Content Style

Generated image

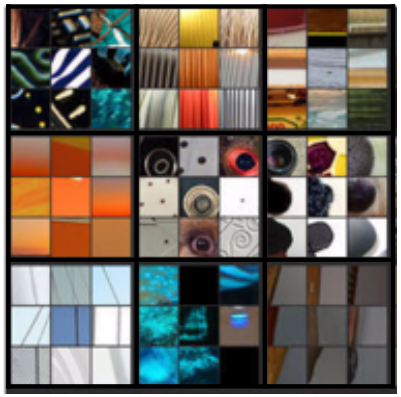



Content Style

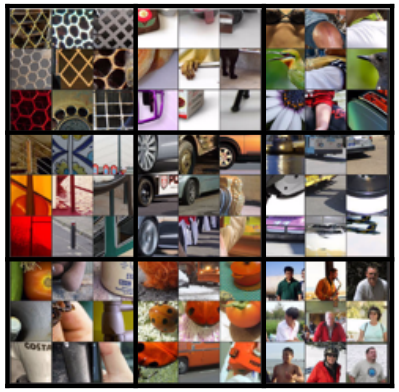
Generated image

Visualizing deep layers

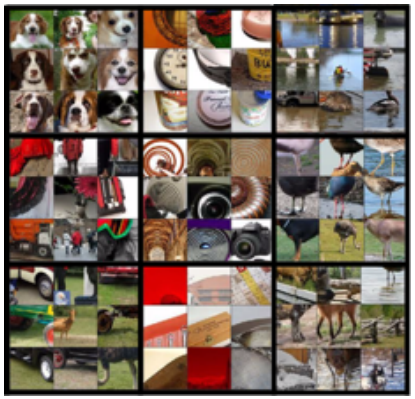
Layer 1



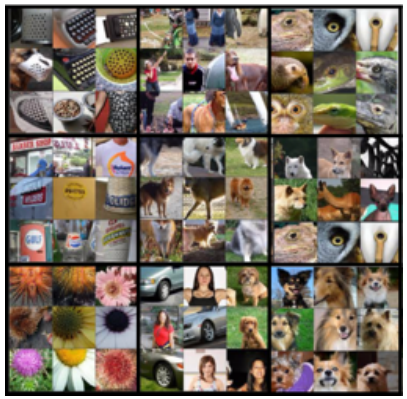
Layer 2



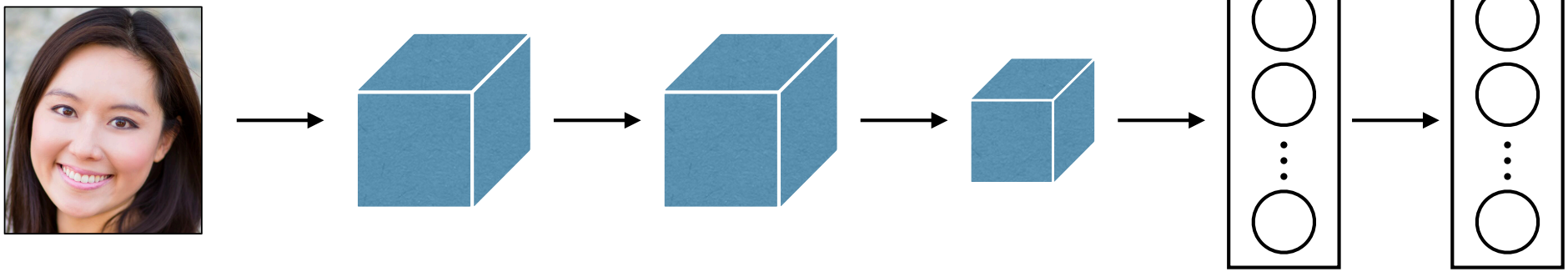
Layer 3



Layer 4

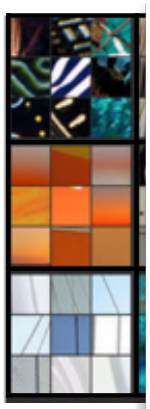


Layer 5

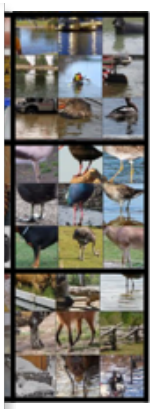


Visualizing deep layers

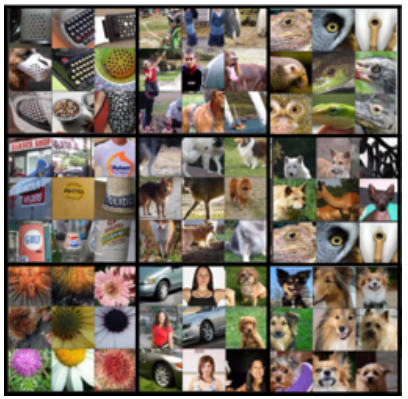
Layer 1



La



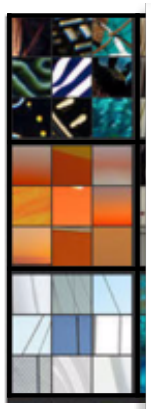
4



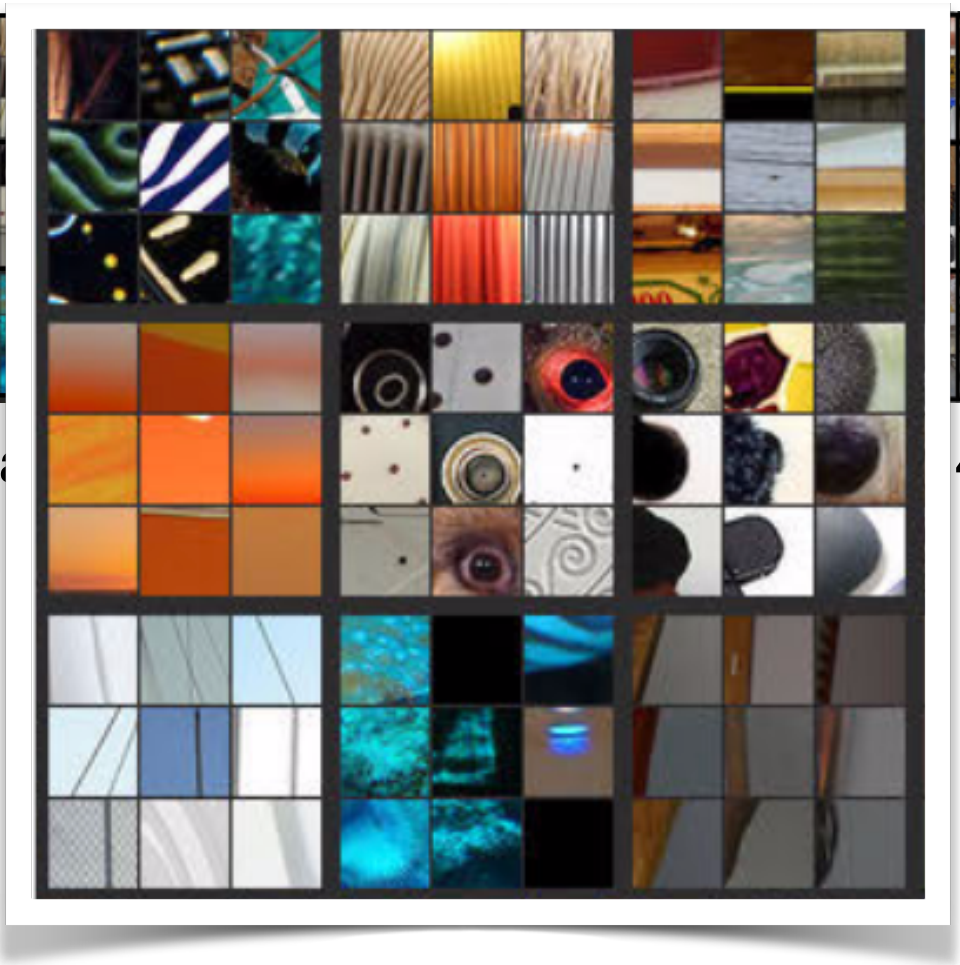
Layer 5

Visualizing deep layers

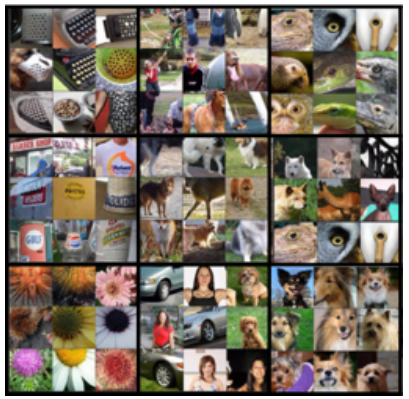
Layer 1



L



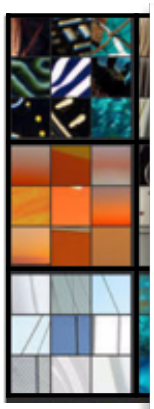
4



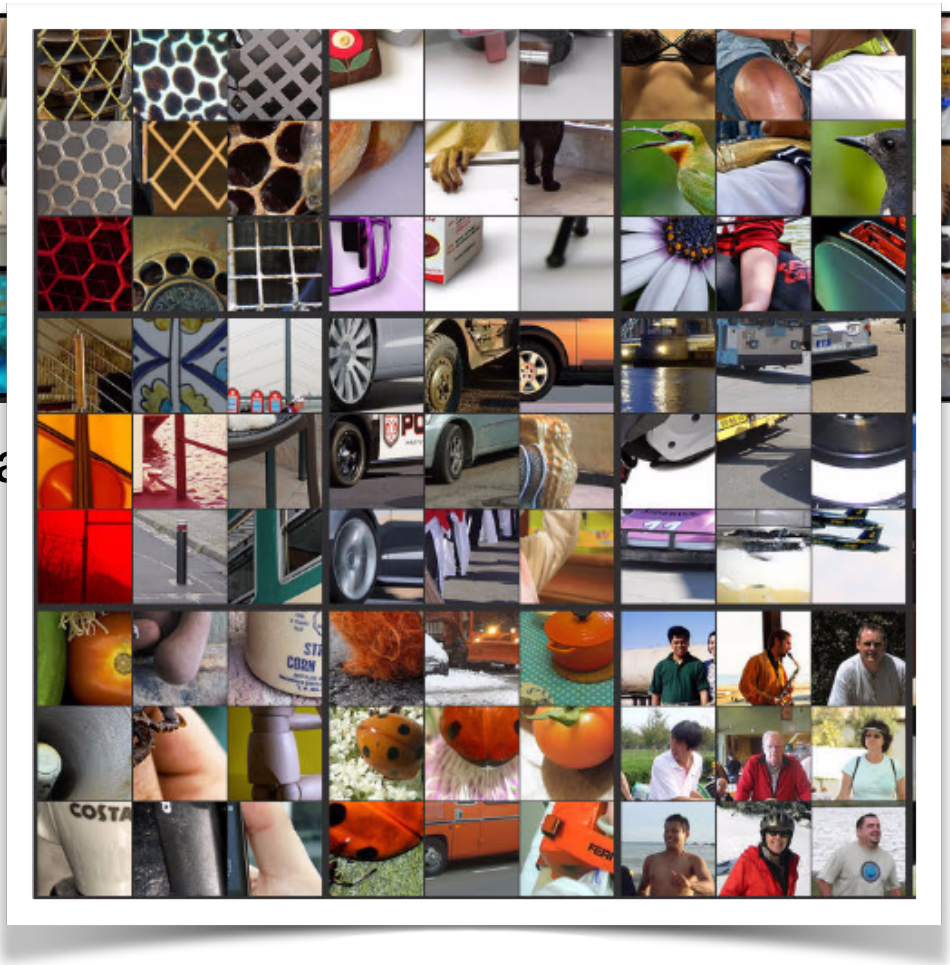
Layer 5

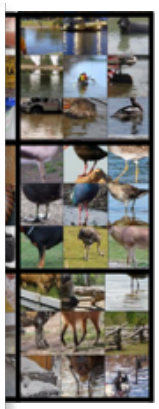
Visualizing deep layers

Layer 1

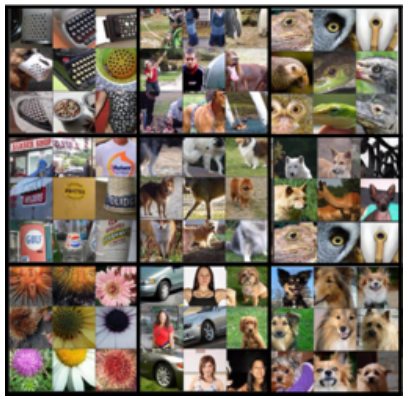


La





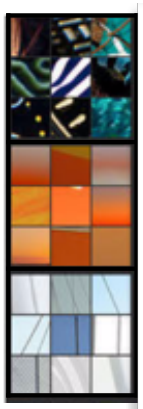
4



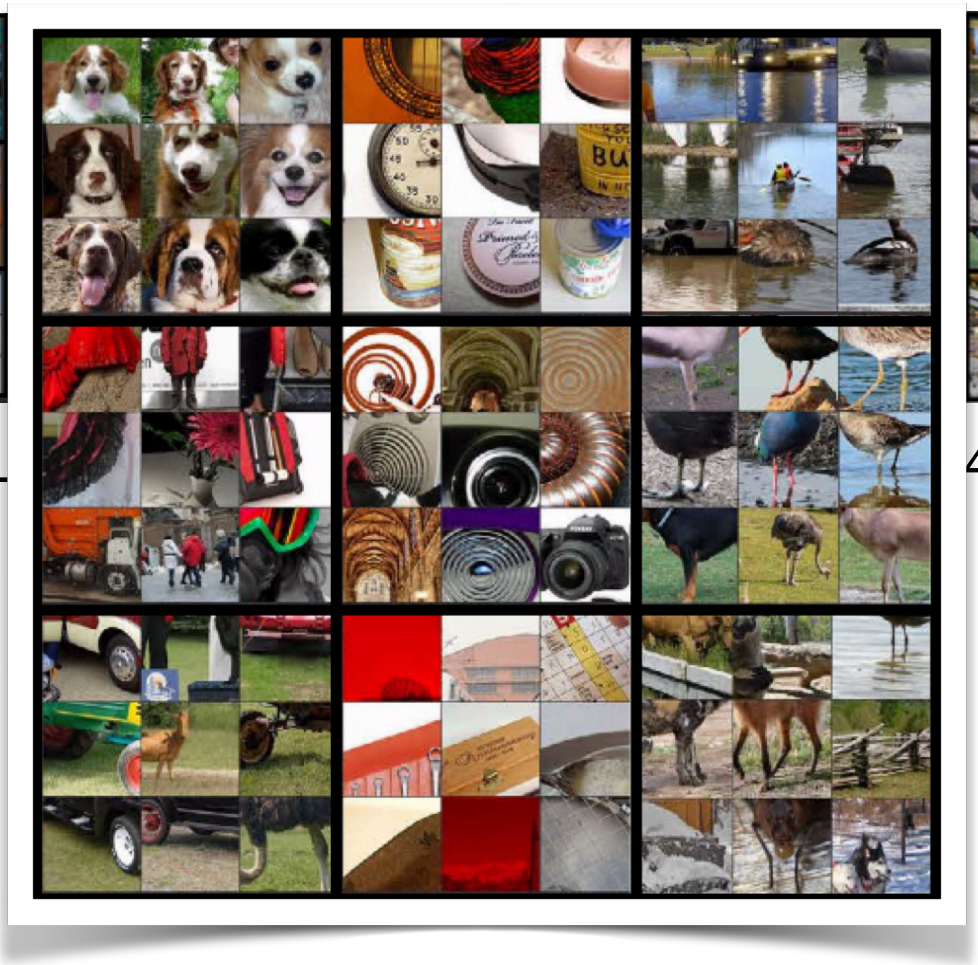
Layer 5

Visualizing deep layers

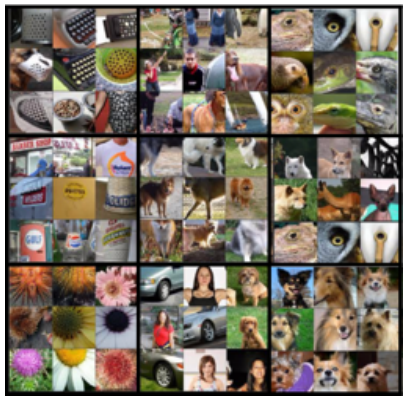
Layer 1



L



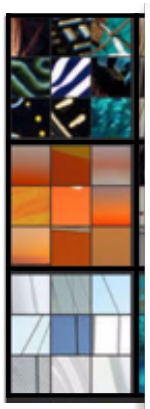
4



Layer 5

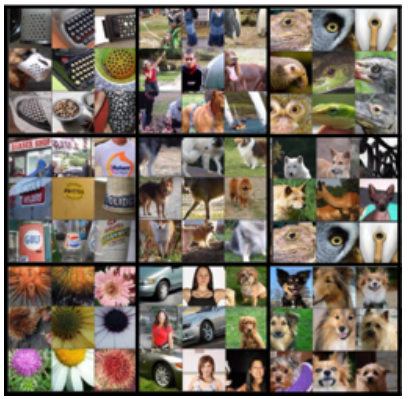
Visualizing deep layers

Layer 1



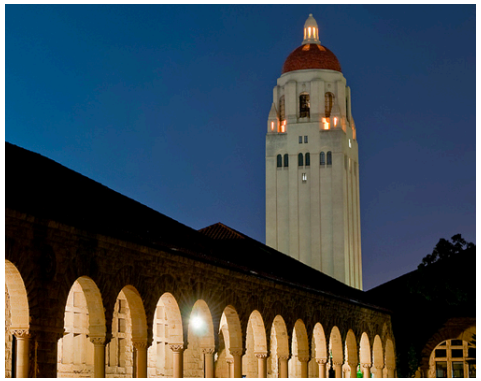
L

4



Layer 5

Cost functions



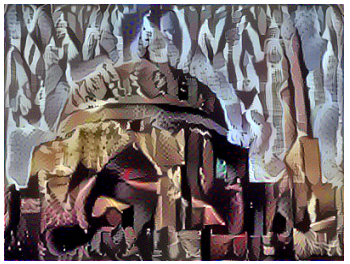
Content Style

- Cost function

$$J(G) = \alpha J_{\text{content}(C,G)} + \beta J_{\text{style}(S,G)}$$

Generated image

Cost functions



- Initialize G randomly
- Use gradient descent to minimize:

$$J(G) = \alpha J_{\text{content}(C,G)} + \beta J_{\text{style}(S,G)}$$

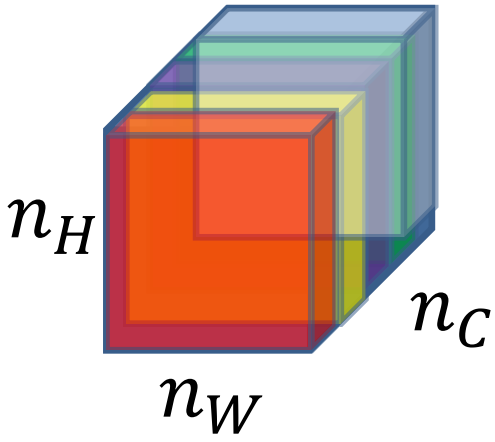
Content cost function

- Say you use hidden layer l to compute content cost
- Use pre-trained ConvNet (e.g., VGG network)
- Let $a^{[l](C)}$ and $a^{[l](G)}$ be the activation of layer l on the images
- If $a^{[l](C)}$ and $a^{[l](G)}$ are similar, both images have similar content

- $J_{\text{content}(C,G)} = \frac{1}{2}\|a^{[l](C)} - a^{[l](G)}\|^2$

Style cost function

- Let $a_{ijk}^{[l]}$ = activation at (i, j, k)
- Let $G^{[l]}$ is $n_c^{[l]} \times n_c^{[l]}$ defined by
  - $$G_{kk'}^{[l](S)} = \sum_{i=1}^{n_H^{[l]}} \sum_{j=1}^{n_W^{[l]}} a_{ijk}^{[l](S)} a_{ijk'}^{[l](S)}$$
  - $$G_{kk'}^{[l](G)} = \sum_{i=1}^{n_H^{[l]}} \sum_{j=1}^{n_W^{[l]}} a_{ijk}^{[l](G)} a_{ijk'}^{[l](G)}$$



- $$J_{\text{style}}^{[l]}(S, G) = \frac{1}{\left(2n_H^{[l]} n_W^{[l]} n_c^{[l]}\right)^2} \| G^{[l](S)} - G^{[l](G)} \|^2$$