



CS540 Introduction to Artificial Intelligence

Deep Learning I: Convolutional Neural Networks

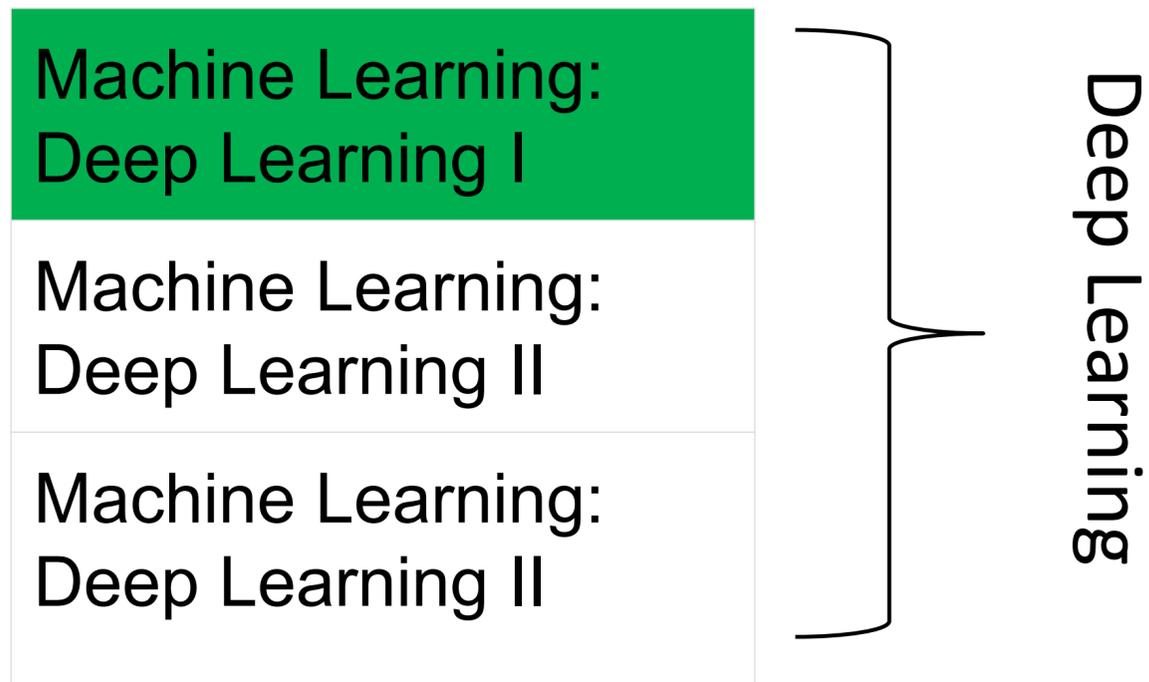
University of Wisconsin-Madison
Spring 2026 Sections 1 & 2

Announcements

- **Homework:**
 - HW6 due **Wednesday March 11th at 11:59PM**
 - HW7 will also be released on Wednesday

- Course evaluation

- Class roadmap:



Midterm Information

- **Time: March 24th 5:45-7:15 PM**
- **Location (by section **):**
 - Section 001 (Tuesday/Thursday 11-12:15PM): 6210 Social Sciences Bldg
 - Section002 (Tuesday/Thursday 2:30-3:45PM): B10 Ingraham Hall
 - Students with McBurney accommodations should have received an email with additional information.
 - Students who cannot take the exam on the specified time should contact their instructor if they have not done it yet.
- **Format:** MCQ
- **Cheat sheet:** a handwritten single piece of paper, front and back
- **Calculator:** optional, if it doesn't have an Internet connection
- **Bring:** your WISC ID, pencil (No 2 or softer), your 1-sheet notes.
- **Past exam questions:** on Canvas → Files → Past Exams

Today's Goals

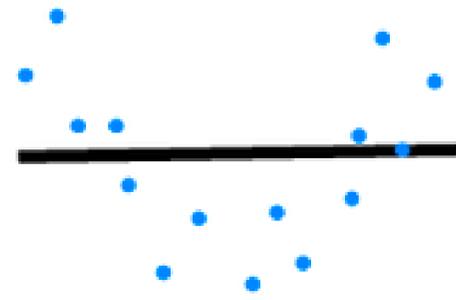
- Finish up Generalization and Regularization
- Build an understanding of convolutional neural networks.
- Why do we want convolutional layers?
- What are convolutional neural networks?
 - 2D vs 3D convolutional networks.
 - Padding and stride.
 - Multiple input and output channels
 - Pooling

Training Error and Generalization Error

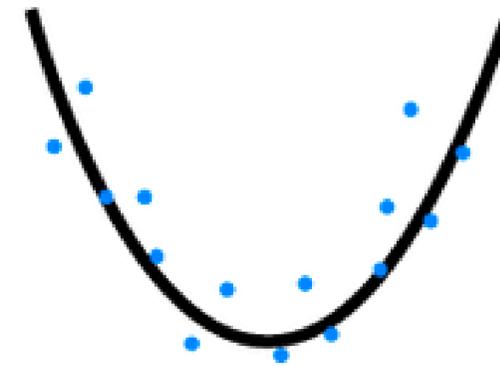
- Training error: model error on the training data
- **Generalization error:** model error on new data
- Example: practice a future exam with past exams
- Doing well on past exams (training error) doesn't guarantee a good score on the future exam (generalization error)

Underfitting

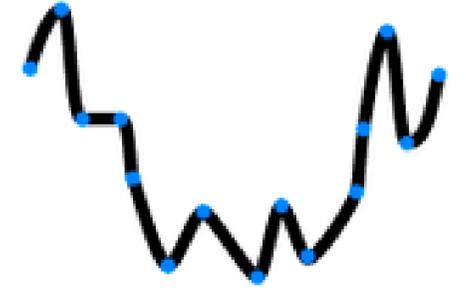
Overfitting



Underfitting



Desired



Overfitting

Image credit: hackernoon.com

Quiz Break 4.1: When training a neural network, which one below indicates that the network has overfit the training data?

- A. Training loss is low and generalization loss is high.
- B. Training loss is low and generalization loss is low.
- C. Training loss is high and generalization loss is high.
- D. Training loss is high and generalization loss is low.
- E. None of these.

Quiz Break 4.1: When training a neural network, which one below indicates that the network has overfit the training data?

- A. Training loss is low and generalization loss is high.
- B. Training loss is low and generalization loss is low.
- C. Training loss is high and generalization loss is high.
- D. Training loss is high and generalization loss is low.
- E. None of these.

Quiz Break 4.2: Adding more layers to a multi-layer perceptron may cause _____.

- A. Vanishing gradients during back propagation.
- B. A more complex decision boundary.
- C. Underfitting.
- D. Higher test loss.
- E. None of these.

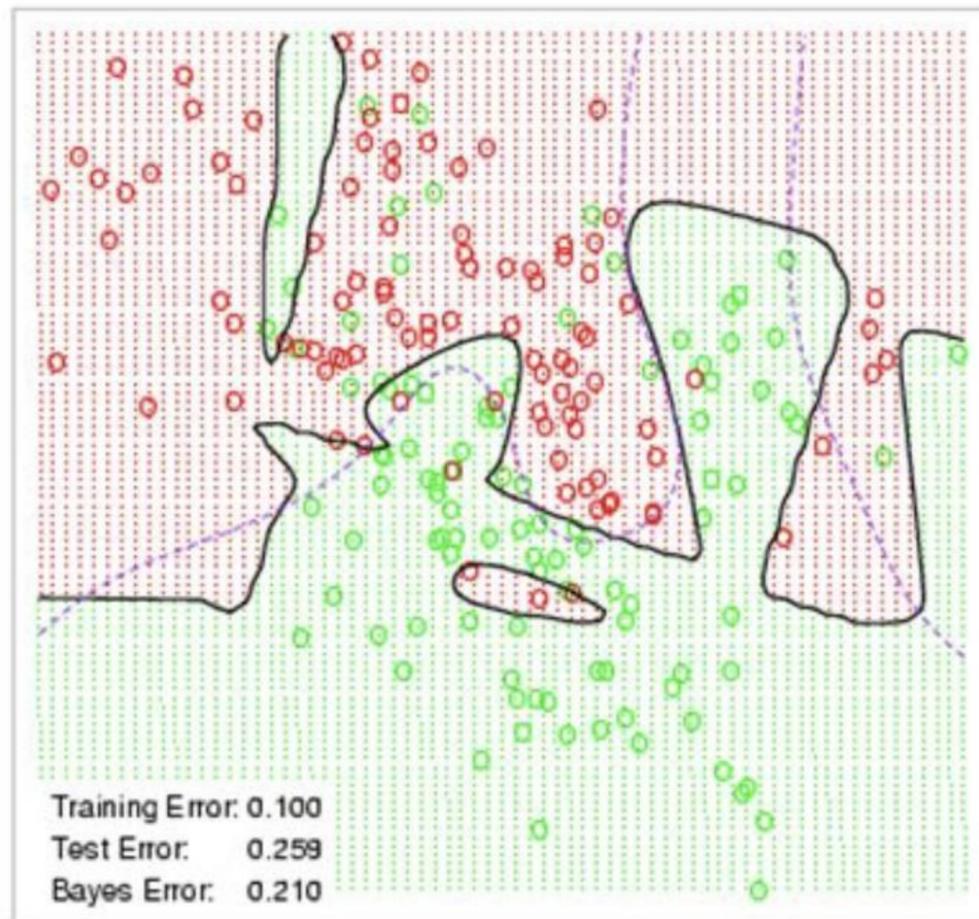
Quiz Break 4.2: Adding more layers to a multi-layer perceptron may cause _____. (Multiple answers)

- A. Vanishing gradients during back propagation.
- B. A more complex decision boundary.
- C. Underfitting.
- D. Higher test loss.
- E. None of these.

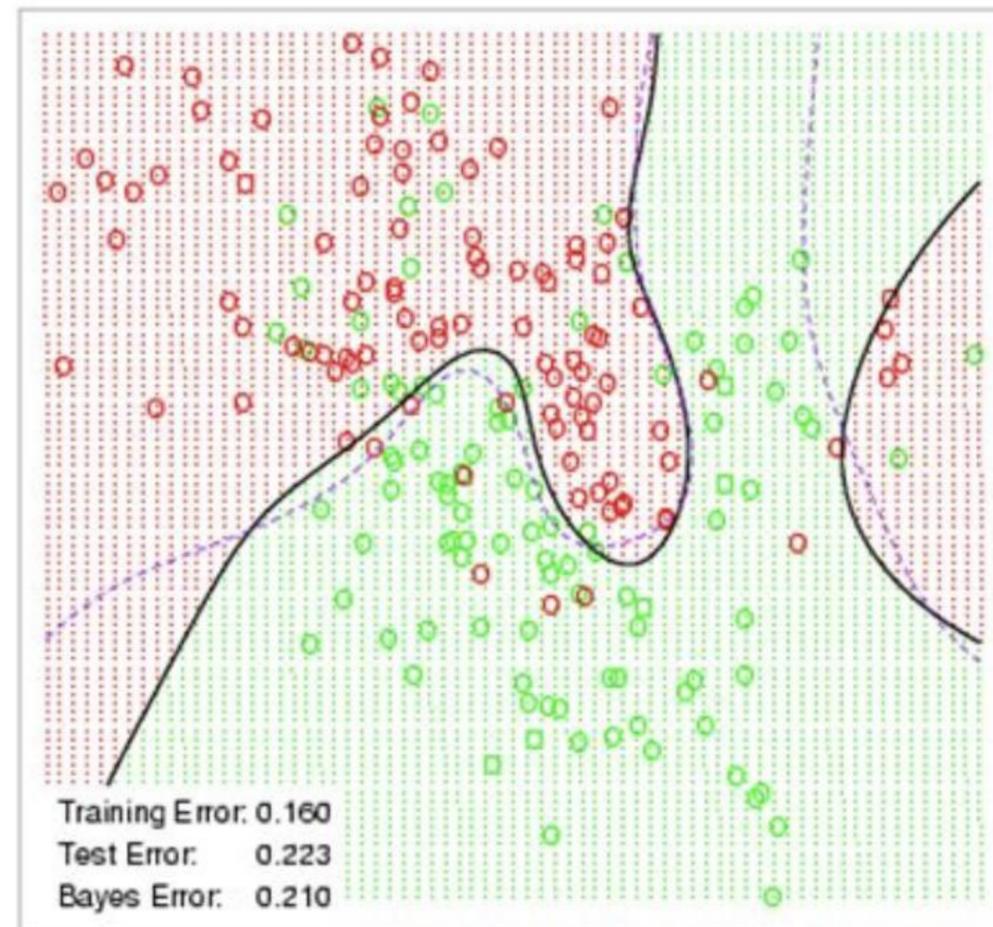
**How to regularize the model for
better generalization?**

Weight Decay

Neural Network - 10 Units, No Weight Decay



Neural Network - 10 Units, Weight Decay=0.02

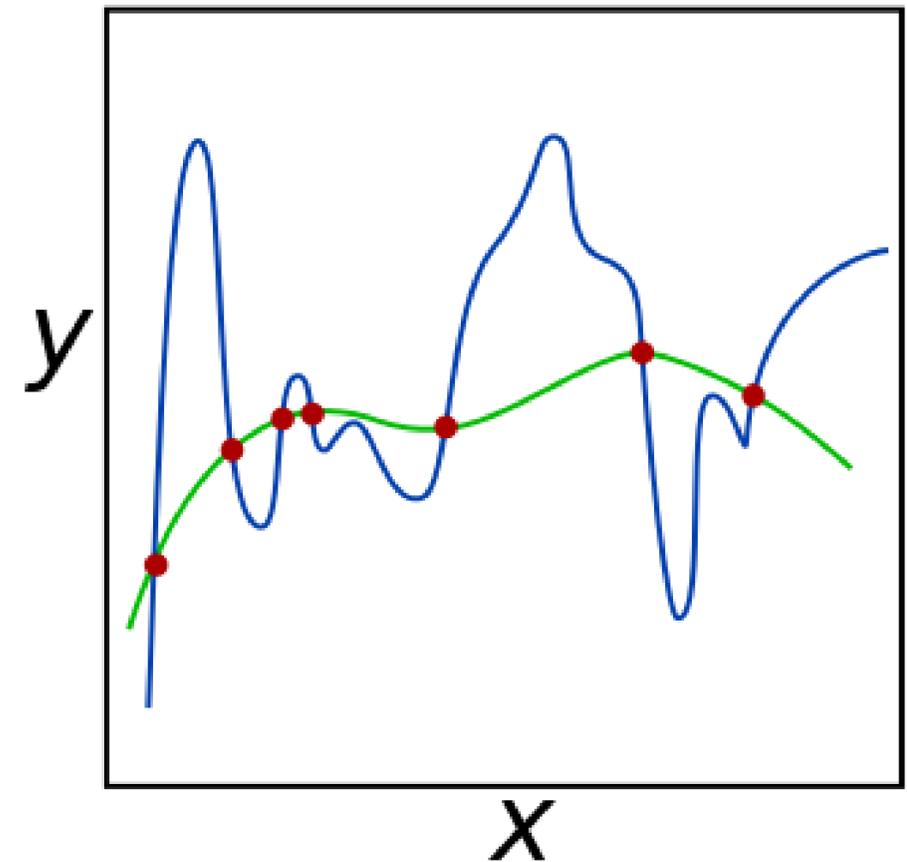


Squared Norm Regularization as Hard Constraint

Reduce model complexity by limiting value range

$$\min L(\mathbf{w}, b) \text{ subject to } \|\mathbf{w}\|^2 \leq B$$

- Often do not regularize bias b
- Doing or not doing has little difference in practice
- A small B means more regularization



Squared Norm Regularization as Soft Constraint

We can rewrite the hard constraint version as

$$\min L(\mathbf{w}, b) + \frac{\lambda}{2} \|\mathbf{w}\|^2$$

Squared Norm Regularization as Soft Constraint

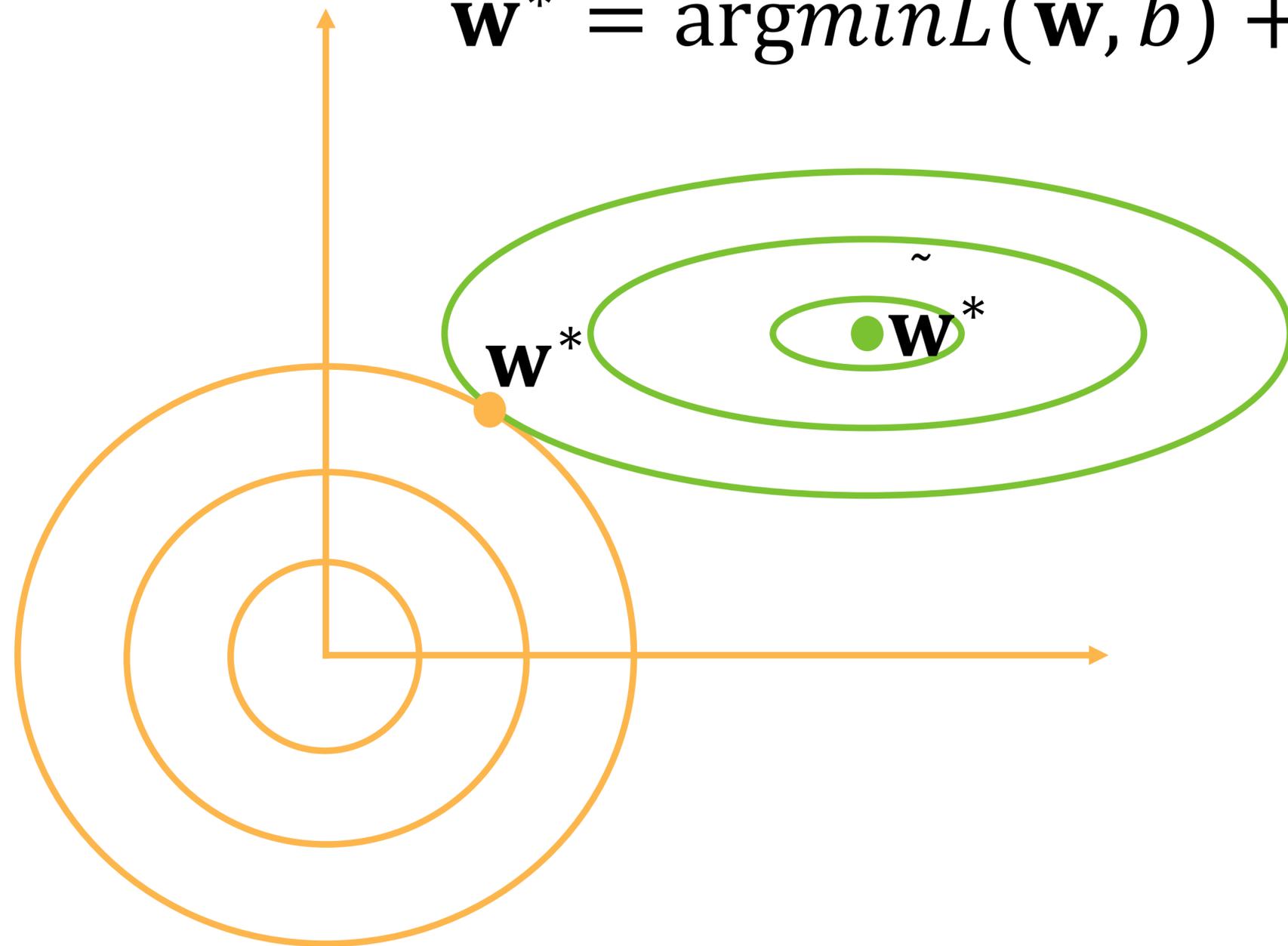
We can rewrite the hard constraint version as

$$\min L(\mathbf{w}, b) + \frac{\lambda}{2} \|\mathbf{w}\|^2$$

- Hyper-parameter λ controls regularization importance
- $\lambda = 0$: no effect
- $\lambda \rightarrow \infty, \mathbf{w}^* \rightarrow \mathbf{0}$

Illustrate the Effect on Optimal Solutions

$$\mathbf{w}^* = \operatorname{argmin} L(\mathbf{w}, b) + \frac{\lambda}{2} \|\mathbf{w}\|^2$$



$$\tilde{\mathbf{w}}^* = \operatorname{argmin} L(\mathbf{w}, b)$$

Dropout

Hinton et al.



Apply Dropout

- Often apply dropout on the output of hidden fully-connected layers

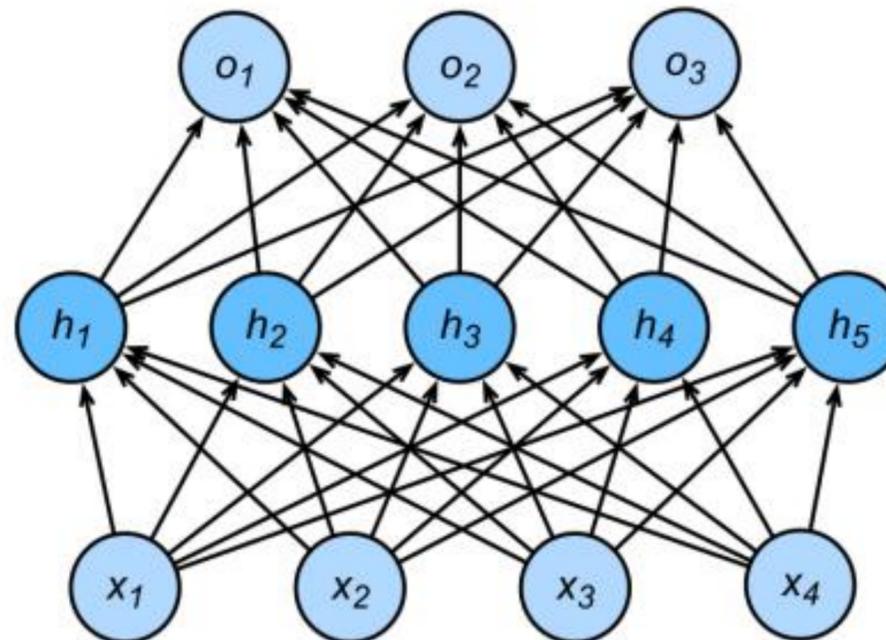
$$\mathbf{h} = \sigma(\mathbf{W}^{(1)}\mathbf{x} + \mathbf{b}^{(1)})$$

$$\mathbf{h}' = \text{dropout}(\mathbf{h})$$

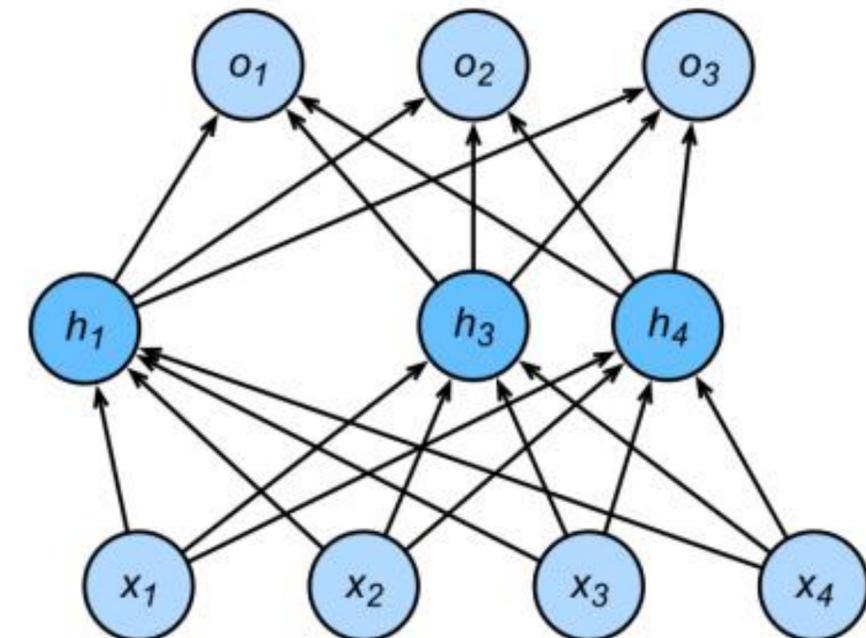
$$\mathbf{o} = \mathbf{W}^{(2)}\mathbf{h}' + \mathbf{b}^{(2)}$$

$$\mathbf{p} = \text{softmax}(\mathbf{o})$$

MLP with one hidden layer



Hidden layer after dropout



Dropout

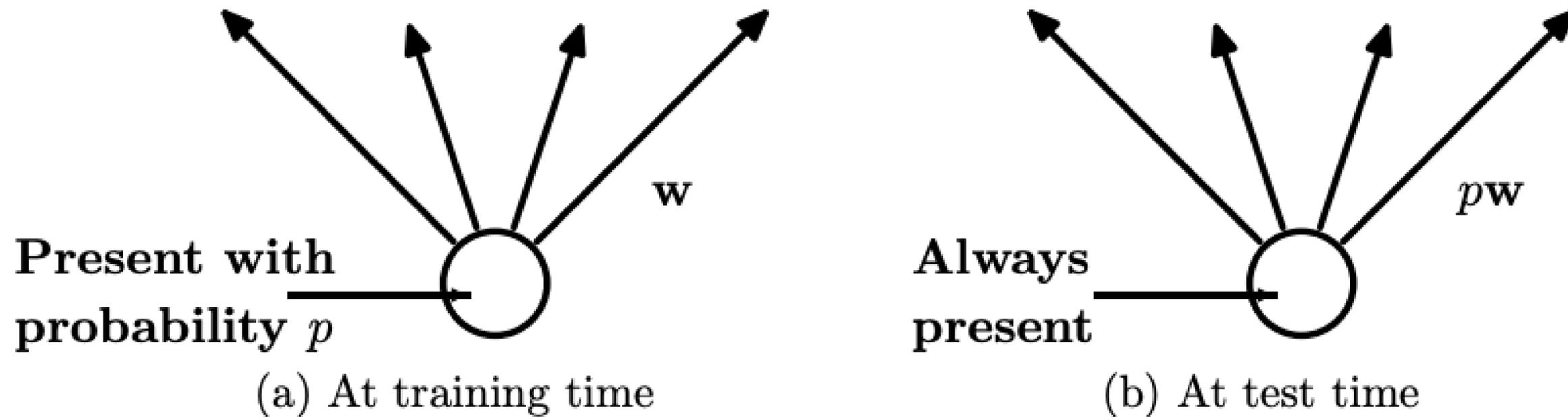


Figure 2: **Left:** A unit at training time that is present with probability p and is connected to units in the next layer with weights \mathbf{w} . **Right:** At test time, the unit is always present and the weights are multiplied by p . The output at test time is same as the expected output at training time.

Dropout

Hinton et al.

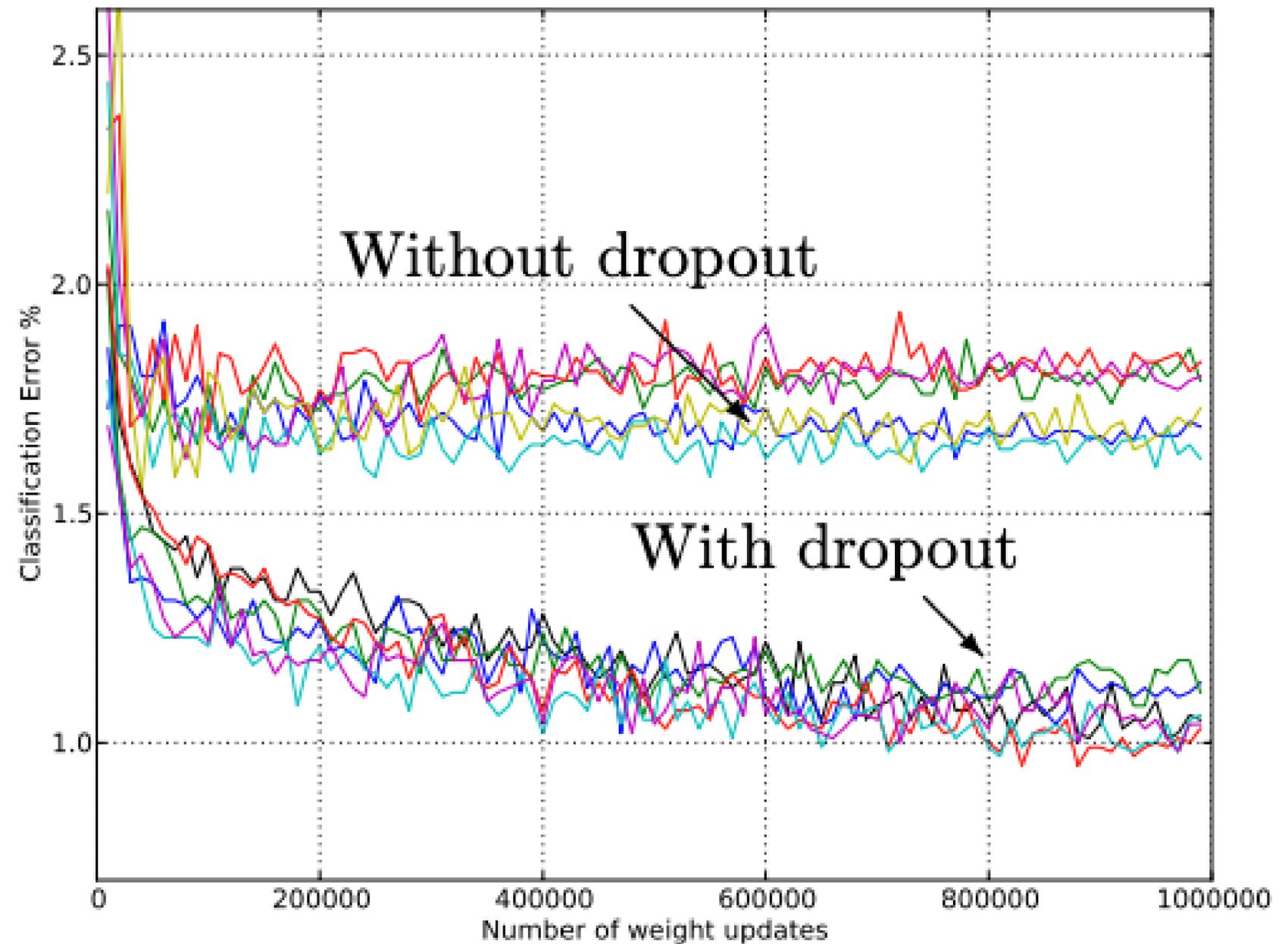


Figure 4: Test error for different architectures with and without dropout. The networks have 2 to 4 hidden layers each with 1024 to 2048 units.

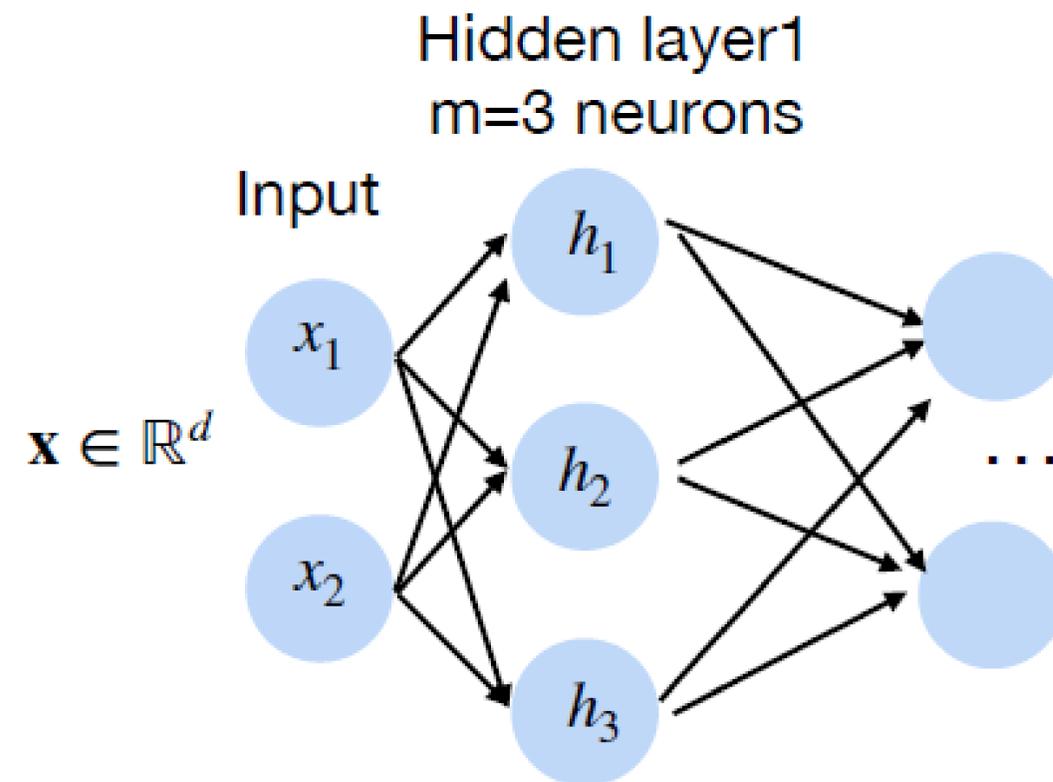
Quiz Break, Q5.1:

In standard dropout regularization, with dropout probability p , each intermediate activation h is replaced by a random variable h' as:

To make $E[h'] = h$. What is “?” ?

$$h' = \begin{cases} 0 & \text{with probability } p \\ ? & \text{otherwise} \end{cases}$$

- A. h
- B. h/p
- C. $h/(1-p)$
- D. $h(1-p)$



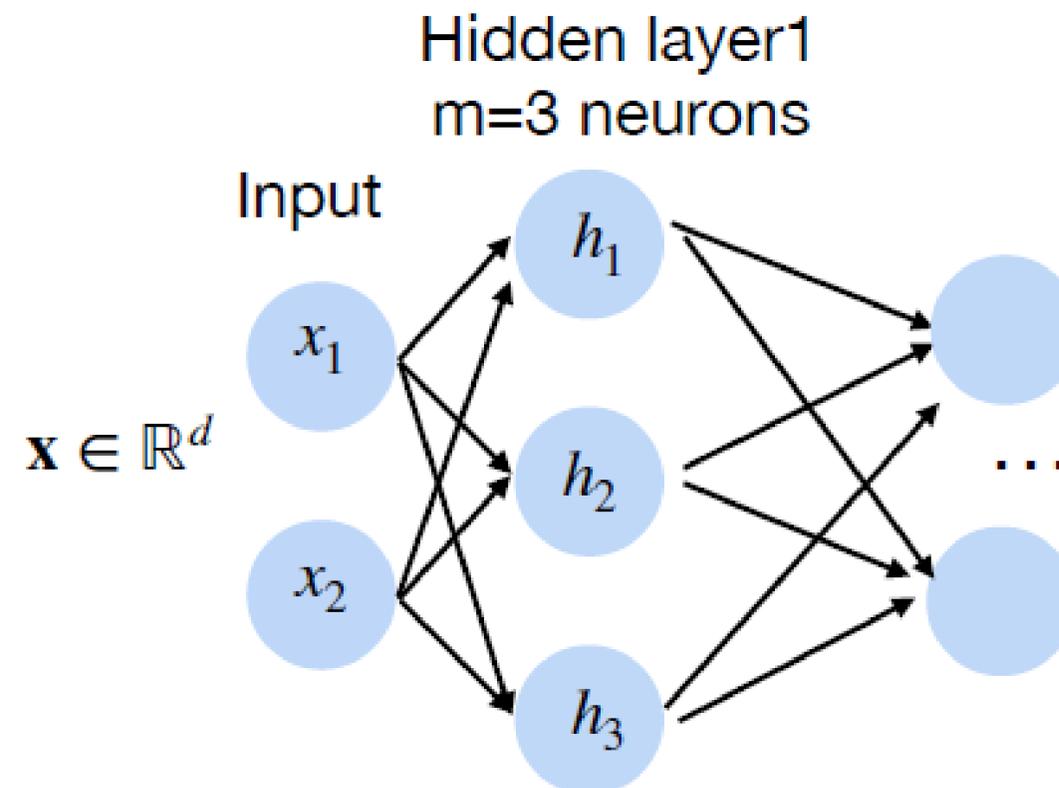
Quiz Break, Q5.1:

In standard dropout regularization, with dropout probability p , each intermediate activation h is replaced by a random variable h' as:

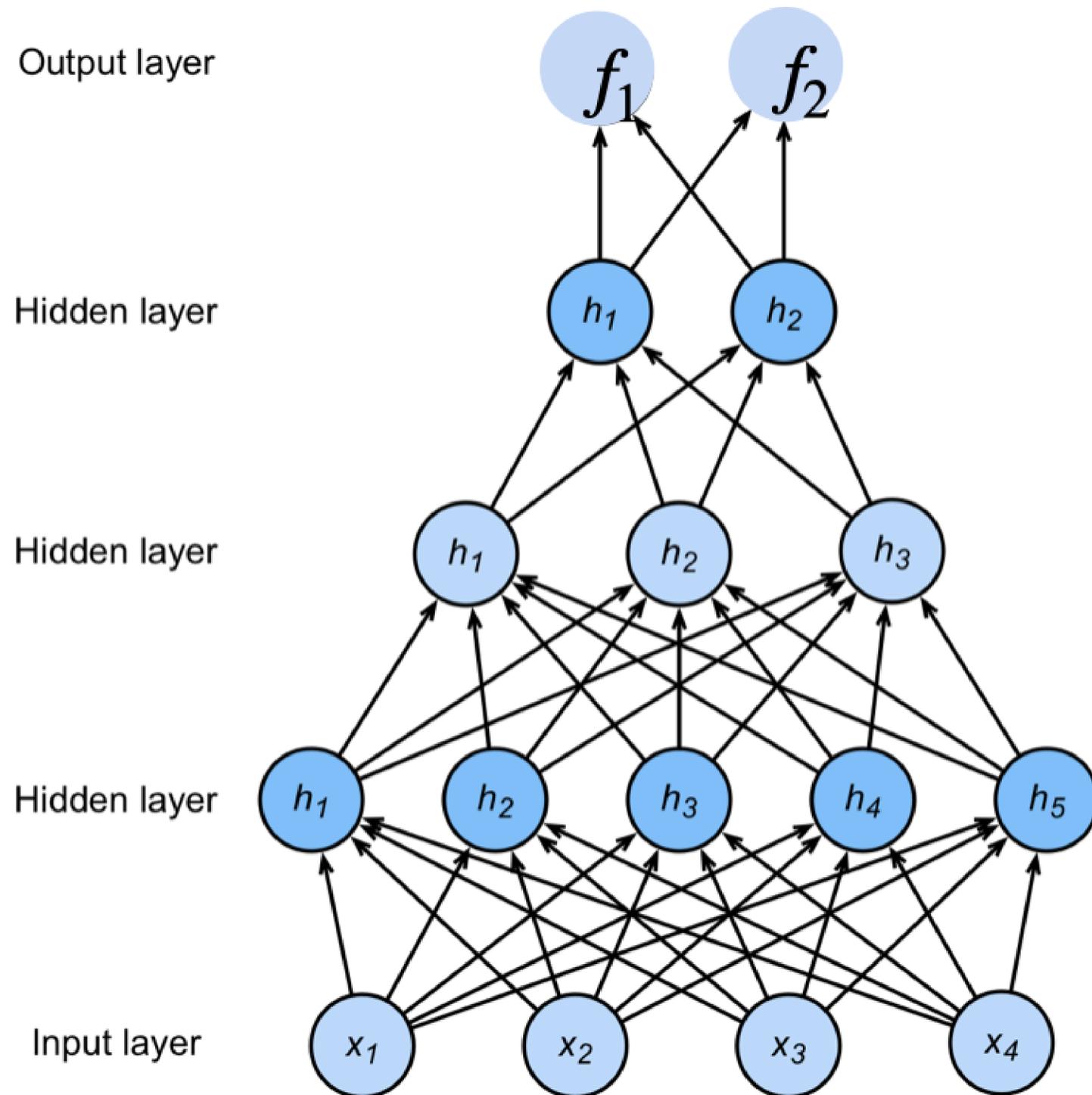
To make $E[h'] = h$. What is “?” ?

$$h' = \begin{cases} 0 & \text{with probability } p \\ ? & \text{otherwise} \end{cases}$$

- A. h
- B. h/p
- C. $h/(1-p)$
- D. $h(1-p)$



Review: Multi-Layer Neural Networks



$$\mathbf{h}_1 = \sigma(\mathbf{W}^{(1)}\mathbf{x} + \mathbf{b}^{(1)})$$

$$\mathbf{h}_2 = \sigma(\mathbf{W}^{(2)}\mathbf{h}_1 + \mathbf{b}^{(2)})$$

$$\mathbf{h}_3 = \sigma(\mathbf{W}^{(3)}\mathbf{h}_2 + \mathbf{b}^{(3)})$$

$$\mathbf{f} = \mathbf{W}^{(4)}\mathbf{h}_3 + \mathbf{b}^{(4)}$$

$$\mathbf{p} = \text{softmax}(\mathbf{f})$$

**NNs are composition
of nonlinear
functions**

How to classify Cats vs. dogs?

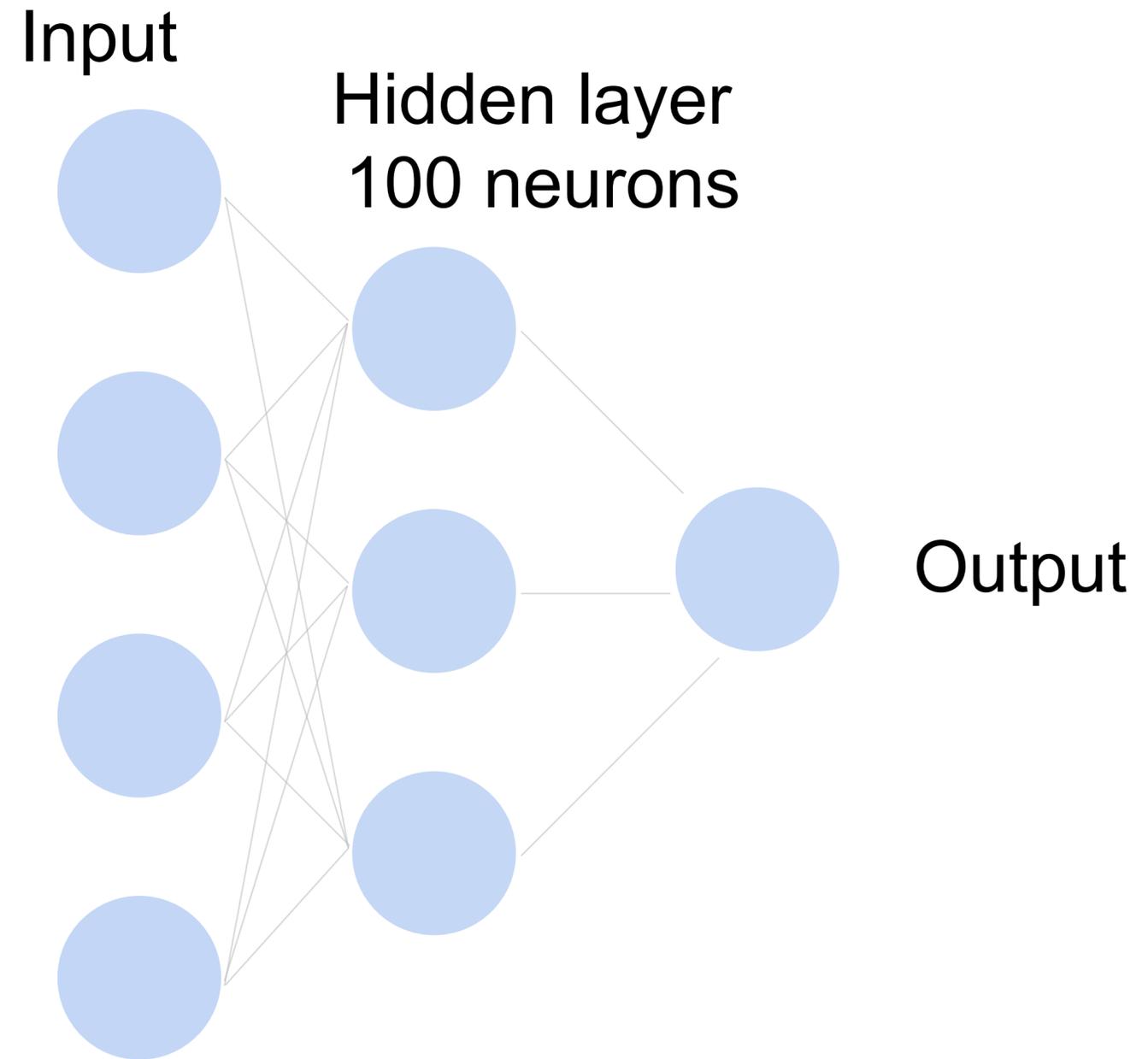


Dual
12MP
wide-angle and
telephoto cameras

36M floats in a RGB image!

Fully Connected Networks

Cats vs. dogs?



$\sim 36\text{M elements} \times 100 = \sim 3.6\text{B parameters!}$

Convolutions come to rescue!

Where is
Waldo?



Why Convolution?

- Translation Invariance
- Locality



2-D Convolution

Input

0	1	2
3	4	5
6	7	8

Kernel

0	1
2	3

*

=

Output

19	25
37	43

$$0 \times 0 + 1 \times 1 + 3 \times 2 + 4 \times 3 = 19$$

2-D Convolution

Input

0	1	2
3	4	5
6	7	8

Kernel

0	1
2	3

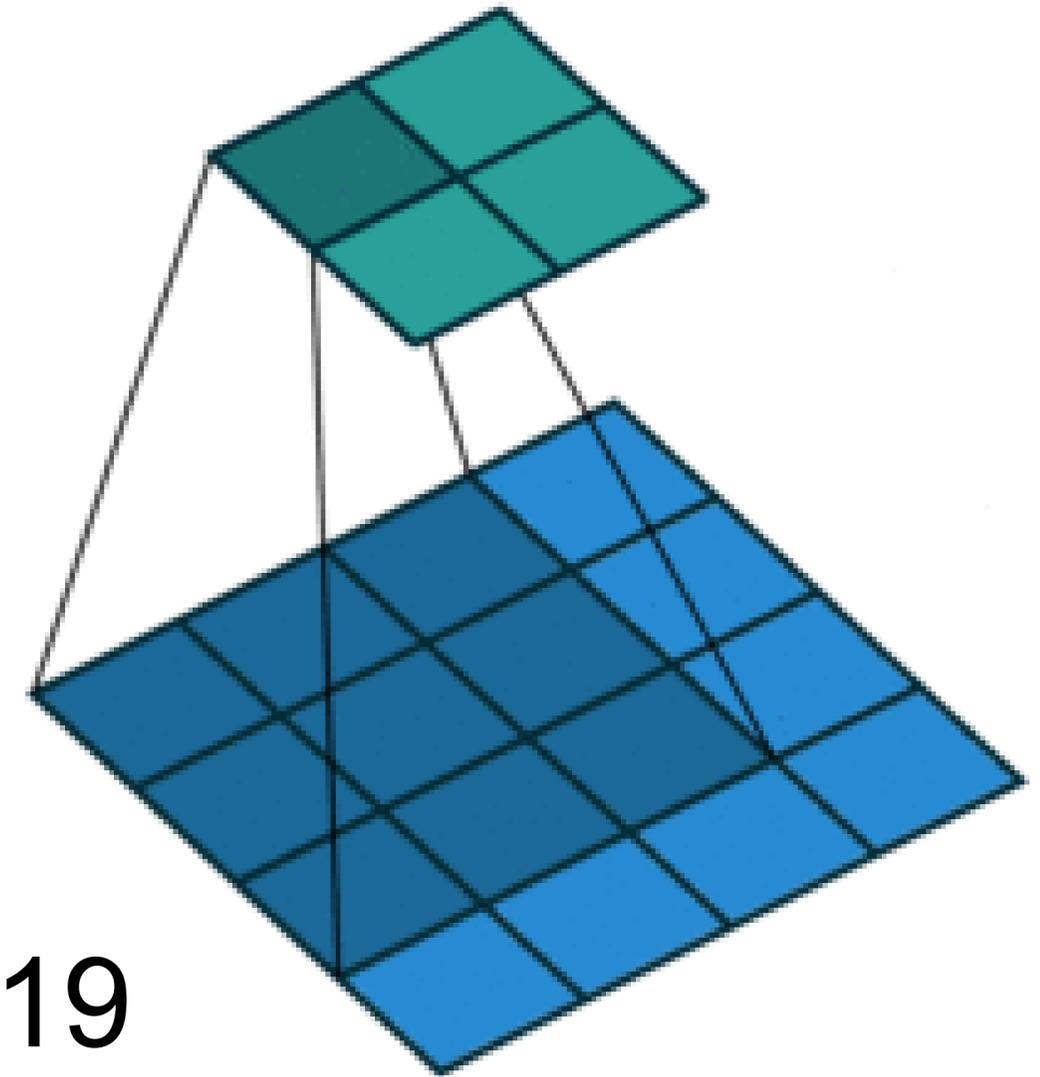
*

=

Output

19	25
37	43

$$0 \times 0 + 1 \times 1 + 3 \times 2 + 4 \times 3 = 19$$



(vdumoulin@ Github)

2-D Convolution

Input

0	1	2
3	4	5
6	7	8

*

Kernel

0	1
2	3

=

Output

19	25
37	43

$$1 \times 0 + 2 \times 1 + 4 \times 2 + 5 \times 3 = 25$$

2-D Convolution

Input

0	1	2
3	4	5
6	7	8

*

Kernel

0	1
2	3

=

Output

19	25
37	43

$$3 \times 0 + 4 \times 1 + 6 \times 2 + 7 \times 3 = 37$$

2-D Convolution

Input

0	1	2
3	4	5
6	7	8

Kernel

0	1
2	3

*

=

Output

19	25
37	43

$$4 \times 0 + 5 \times 1 + 7 \times 2 + 8 \times 3 = 43$$

2-D Convolution Layer

0	1	2
3	4	5
6	7	8

 *

0	1
2	3

 =

19	25
37	43

- **X**: $n_h \times n_w$ input matrix
- **W**: $k_h \times k_w$ kernel matrix
- **Y**: $(n_h - k_h + 1) \times (n_w - k_w + 1)$ output matrix

$$Y = X * W$$

Convolution operator not multiplication

2-D Convolution Layer

0	1	2
3	4	5
6	7	8

 *

0	1
2	3

 + 1 =

20	26
38	44

- **X** : $n_h \times n_w$ input matrix
- **W** : $k_h \times k_w$ kernel matrix
- **b** : scalar bias
- **Y** : $(n_h - k_h + 1) \times (n_w - k_w + 1)$ output matrix

$$Y = X * W + b$$

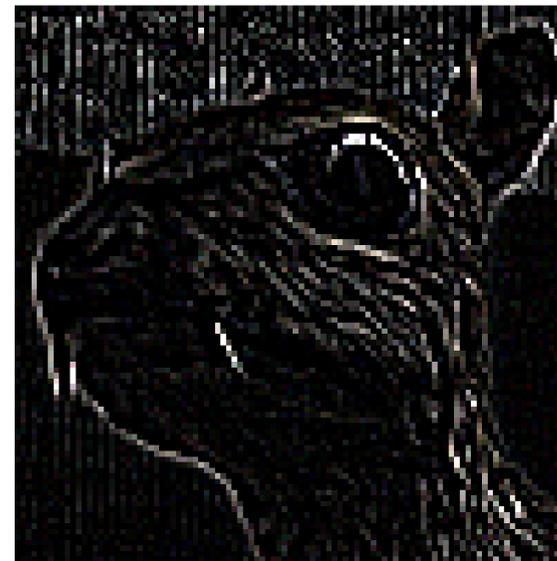
- **W** and b are learnable parameters

Examples



(wikipedia)

$$\begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$$



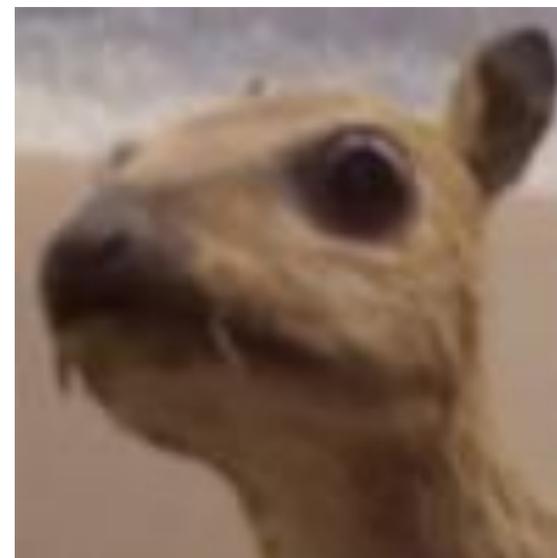
Edge Detection

$$\begin{bmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{bmatrix}$$



Sharpen

$$\frac{1}{16} \begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}$$



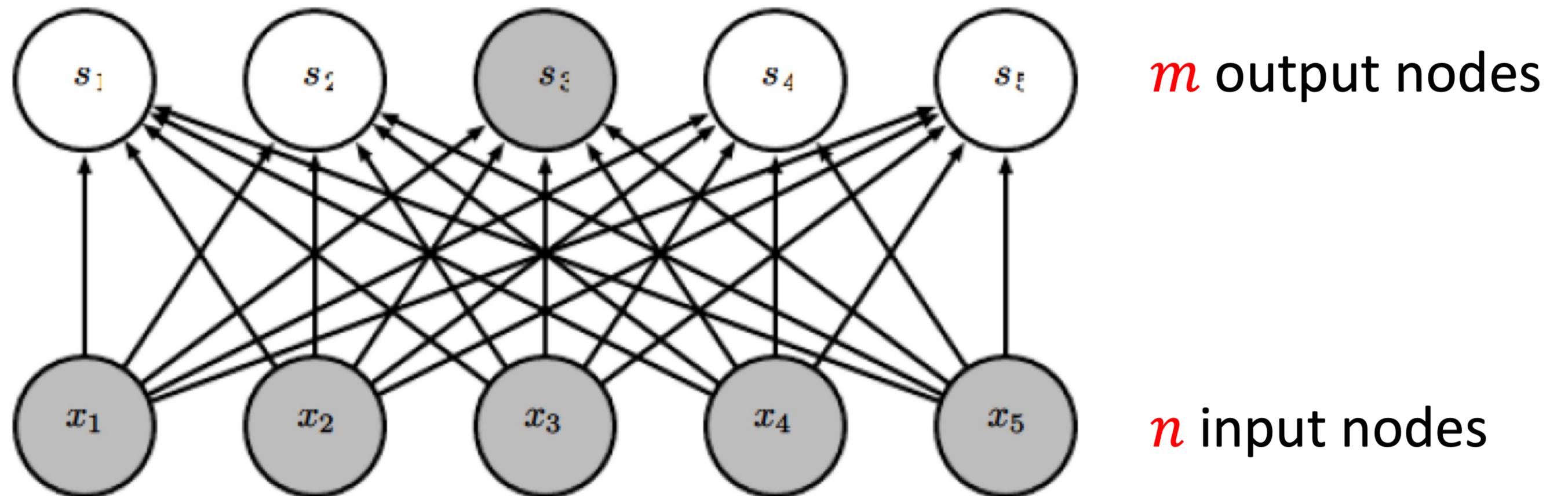
Gaussian Blur

Convolutional Neural Networks

- Convolutional networks: neural networks that use convolution in place of general matrix multiplication in at least one of their layers
- Strong empirical performance in applications – particularly computer vision.
- Examples: image classification, object detection.

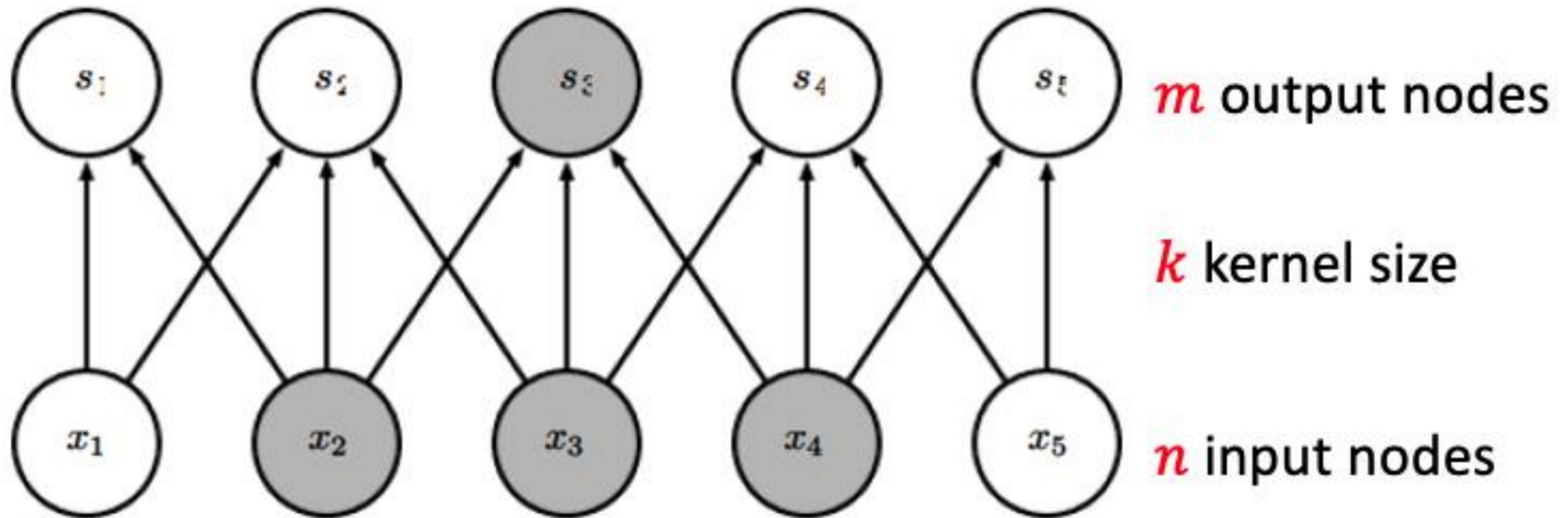
Advantage: sparse interaction

Fully connected layer, $m \times n$ edges



Advantage: sparse interaction

Convolutional layer, $\leq m \times k$ edges



Q1. Suppose we want to perform convolution as follows. What's the output?

0	1	2
3	4	5
6	7	8

*

0	1
1	-1

+ 1 = ?

A.

1	2
4	5

B.

1	2
3	4

C.

1	3
3	5

D.

0	1
3	4

Q1. Suppose we want to perform convolution as follows. What's the output?

0	1	2
3	4	5
6	7	8

*

0	1
1	-1

+ 1

=

1	2
4	5

A.

1	2
4	5

B.

1	2
3	4

B.

1	3
3	5

0	1
3	4

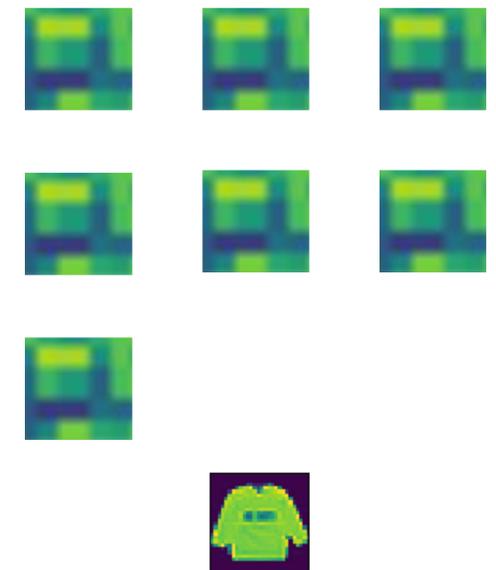
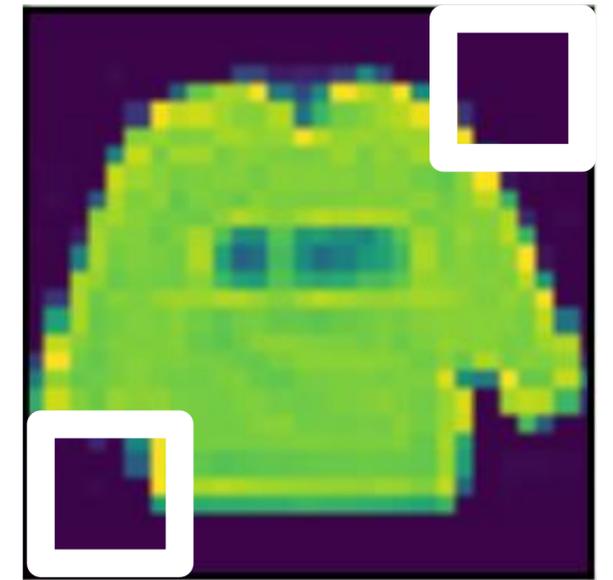
$$\begin{aligned}
 0 \times 0 + 1 \times 1 + 3 \times 1 + 4 \times (-1) + 1 &= 1 \\
 1 \times 0 + 2 \times 1 + 4 \times 1 + 5 \times (-1) + 1 &= 2 \\
 3 \times 0 + 4 \times 1 + 6 \times 1 + 7 \times (-1) + 1 &= 4 \\
 4 \times 0 + 5 \times 1 + 7 \times 1 + 8 \times (-1) + 1 &= 5
 \end{aligned}$$



Padding and Stride

Padding

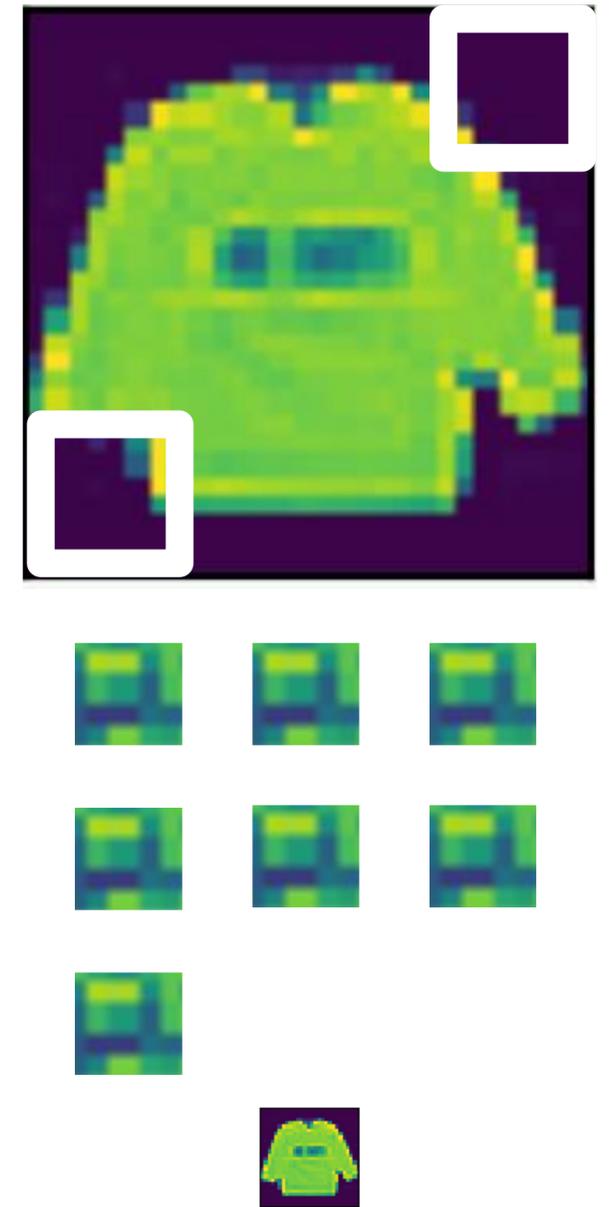
- Given a 32 x 32 input image
- Apply convolution with 5 x 5 kernel
 - 28 x 28 output with 1 layer
 - 4 x 4 output with 7 layers



Padding

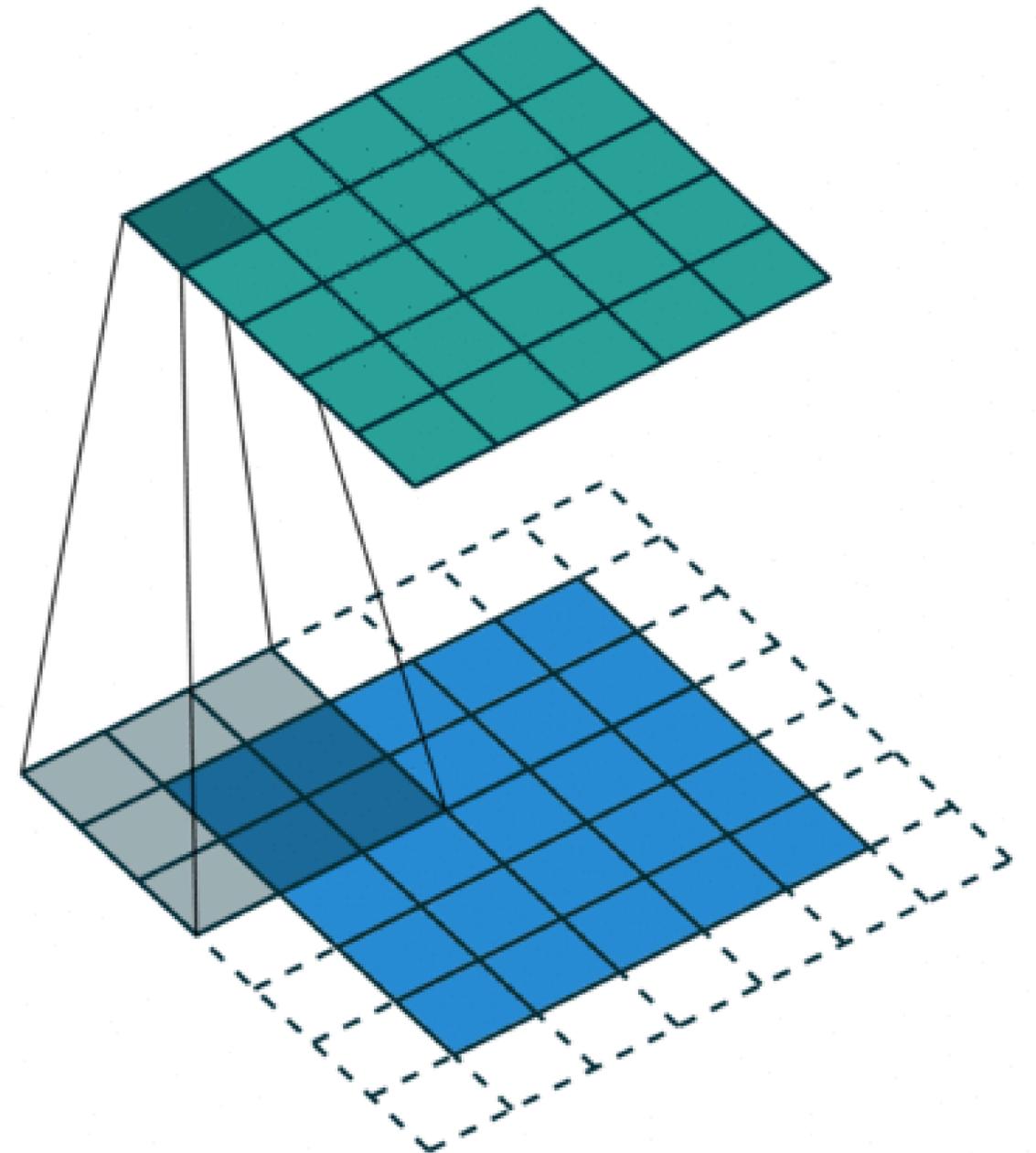
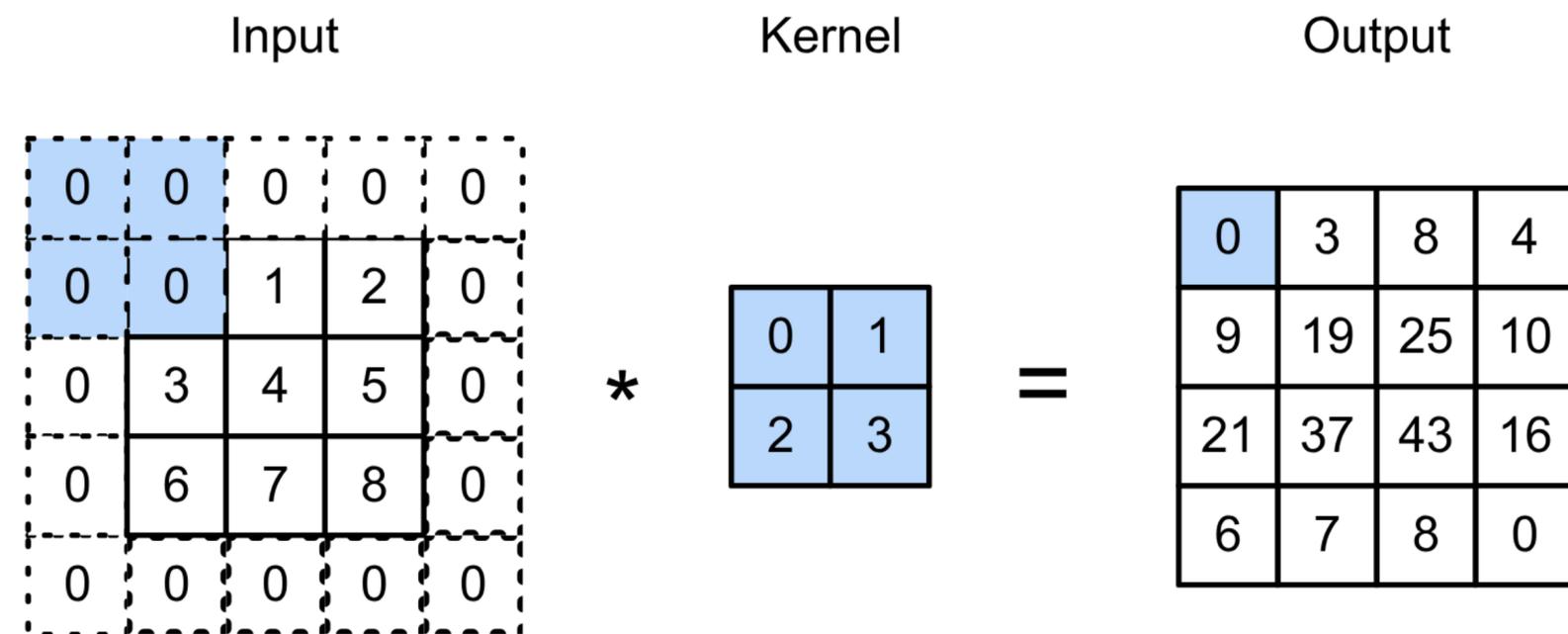
- Given a 32 x 32 input image
- Apply convolution with 5 x 5 kernel
 - 28 x 28 output with 1 layer
 - 4 x 4 output with 7 layers
- Shape decreases faster with larger kernels
- Shape reduces from $n_h \times n_w$ to

$$(n_h - k_h + 1) \times (n_w - k_w + 1)$$



Convolutional Layers: Padding

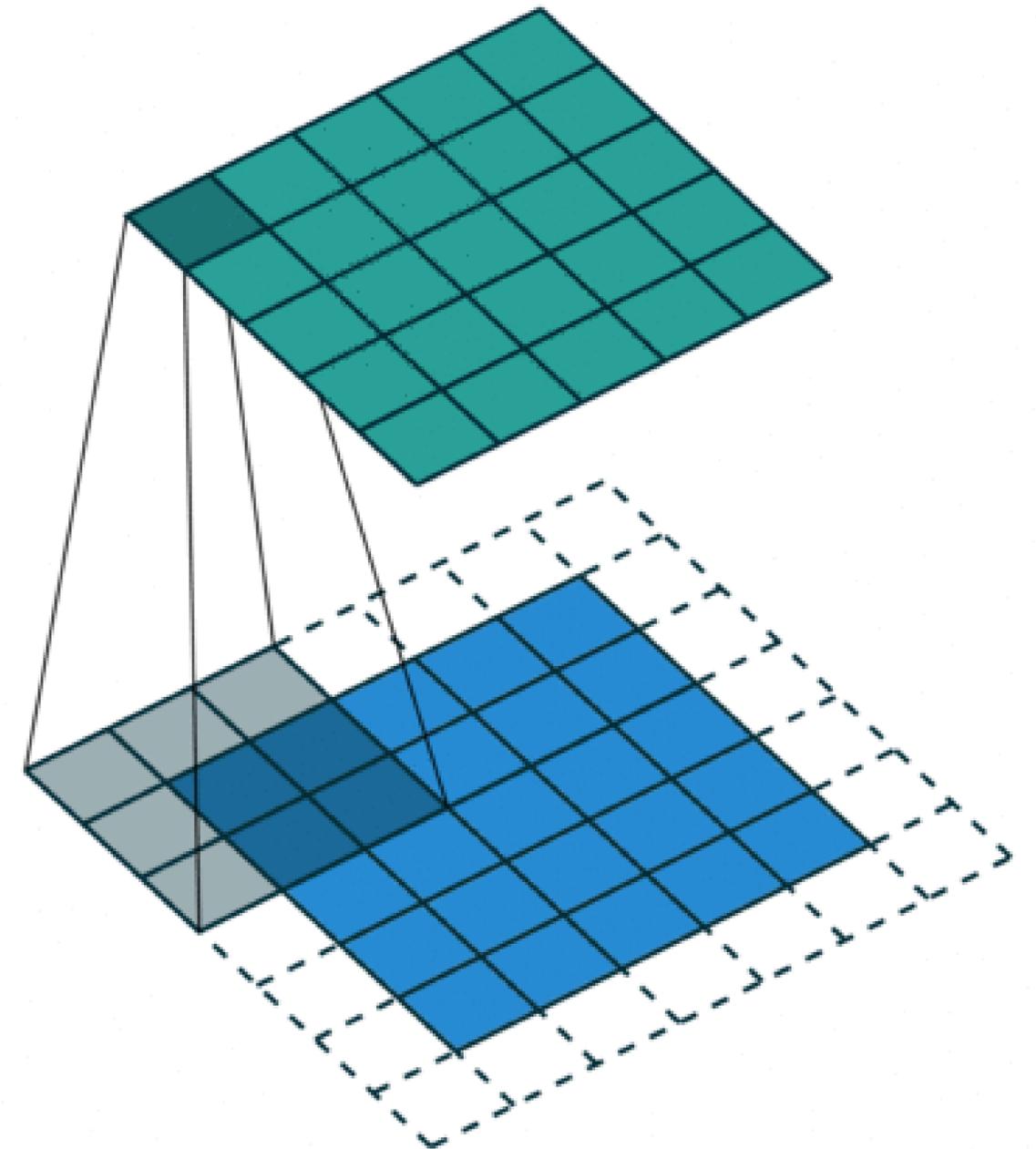
Padding adds rows/columns around input



Convolutional Layers: Padding

Padding adds rows/columns around input

- Why?
 1. Keeps **edge information**
 2. Preserves sizes / allows deep networks
 - ie, for a 32x32 input image, 5x5 kernel, after 1 layer, get 28x28, after 7 layers, **only 4x4**
 3. Can combine different filter sizes



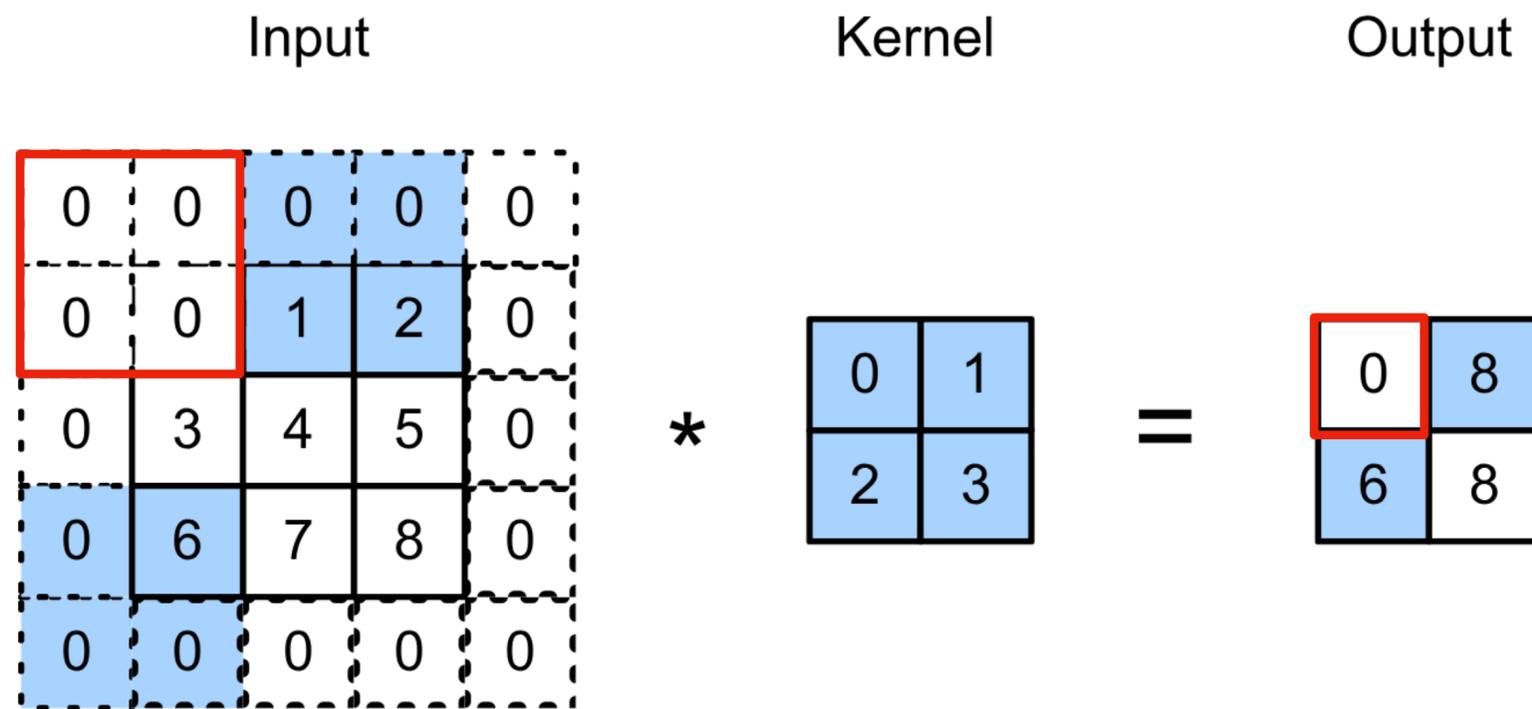
Convolutional Layers: Padding

- Padding p_h rows and p_w columns, output shape is
$$(n_h - k_h + p_h + 1) \times (n_w - k_w + p_w + 1)$$
- Common choice is $p_h = k_h - 1$ and $p_w = k_w - 1$
 - Odd k_h : pad $p_h/2$ on both top and bottom
 - Even k_h : pad $\text{ceil}(p_h/2)$ on top, $\text{floor}(p_h/2)$ on bottom

Stride

- Stride is the #rows / #columns per slide

Example: strides of 3 and 2 for height and width



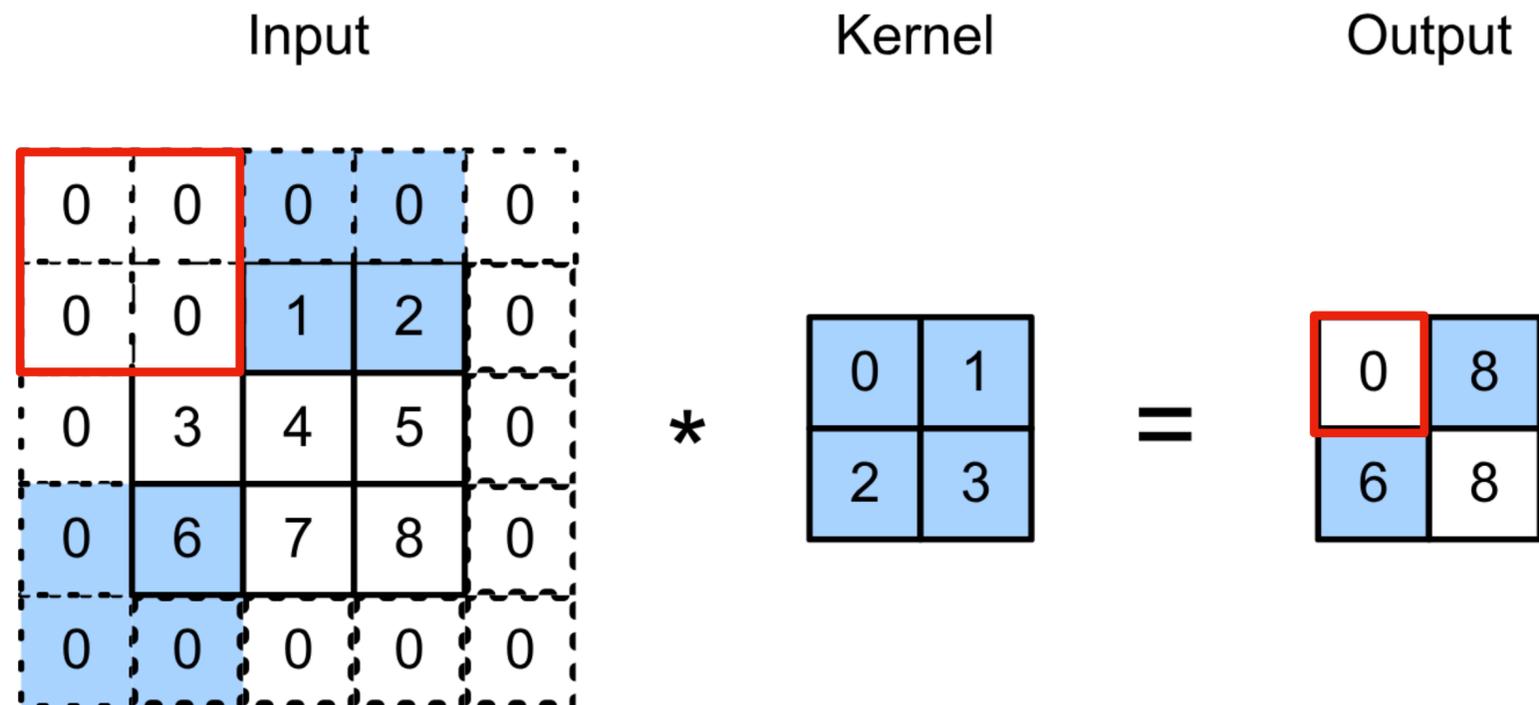
$$0 \times 0 + 0 \times 1 + 1 \times 2 + 2 \times 3 = 8$$

$$0 \times 0 + 6 \times 1 + 0 \times 2 + 0 \times 3 = 6$$

Stride

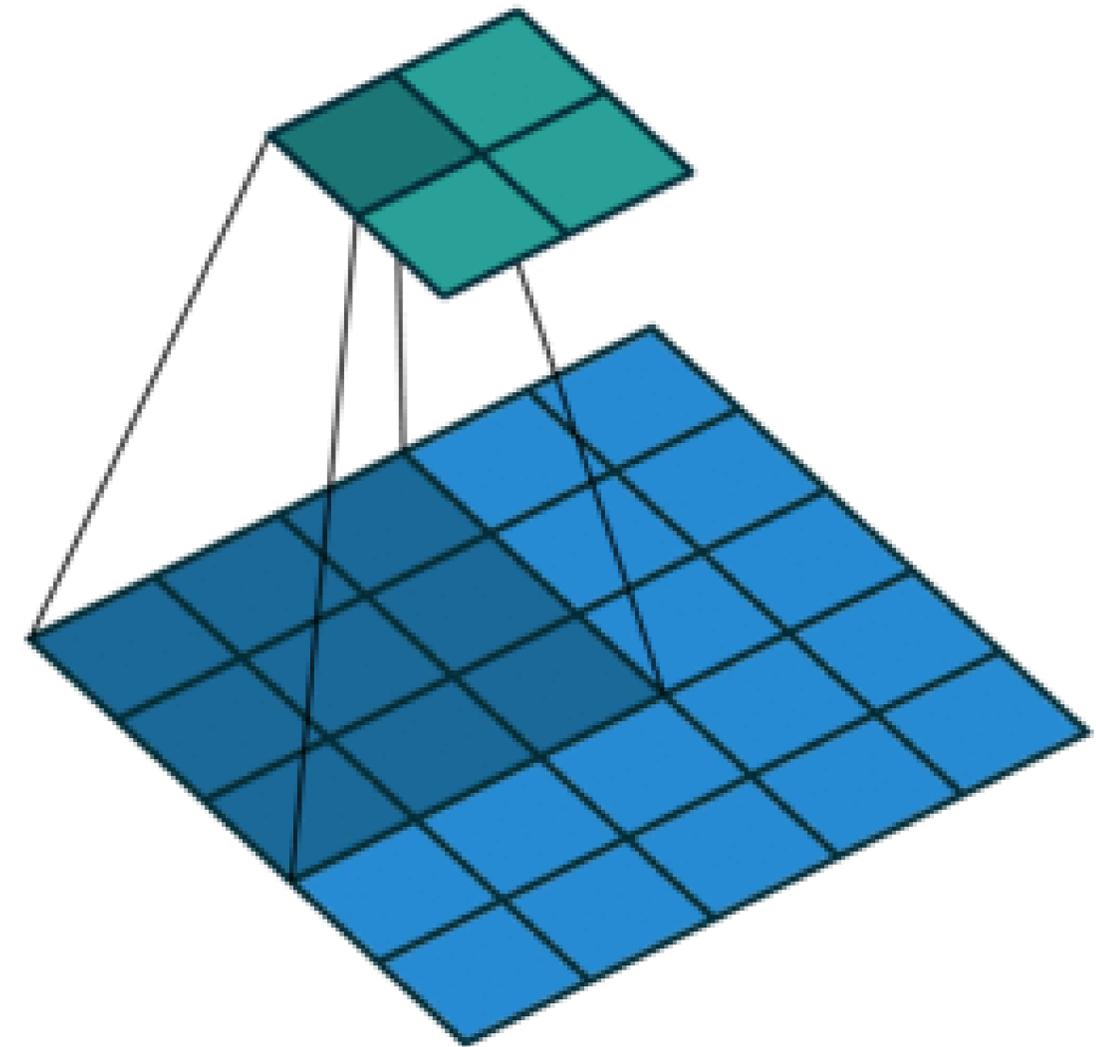
- Stride is the #rows / #columns per slide

Example: strides of 3 and 2 for height and width



$$0 \times 0 + 0 \times 1 + 1 \times 2 + 2 \times 3 = 8$$

$$0 \times 0 + 6 \times 1 + 0 \times 2 + 0 \times 3 = 6$$



Stride 2,2

Convolutional Layers: Stride

- Given stride s_h for the height and stride s_w for the width, the output shape is

$$\lfloor (n_h - k_h + p_h + s_h) / s_h \rfloor \times \lfloor (n_w - k_w + p_w + s_w) / s_w \rfloor$$

- Set $p_h = k_h - 1$, $p_w = k_w - 1$, then get

$$\lfloor (n_h + s_h - 1) / s_h \rfloor \times \lfloor (n_w + s_w - 1) / s_w \rfloor$$

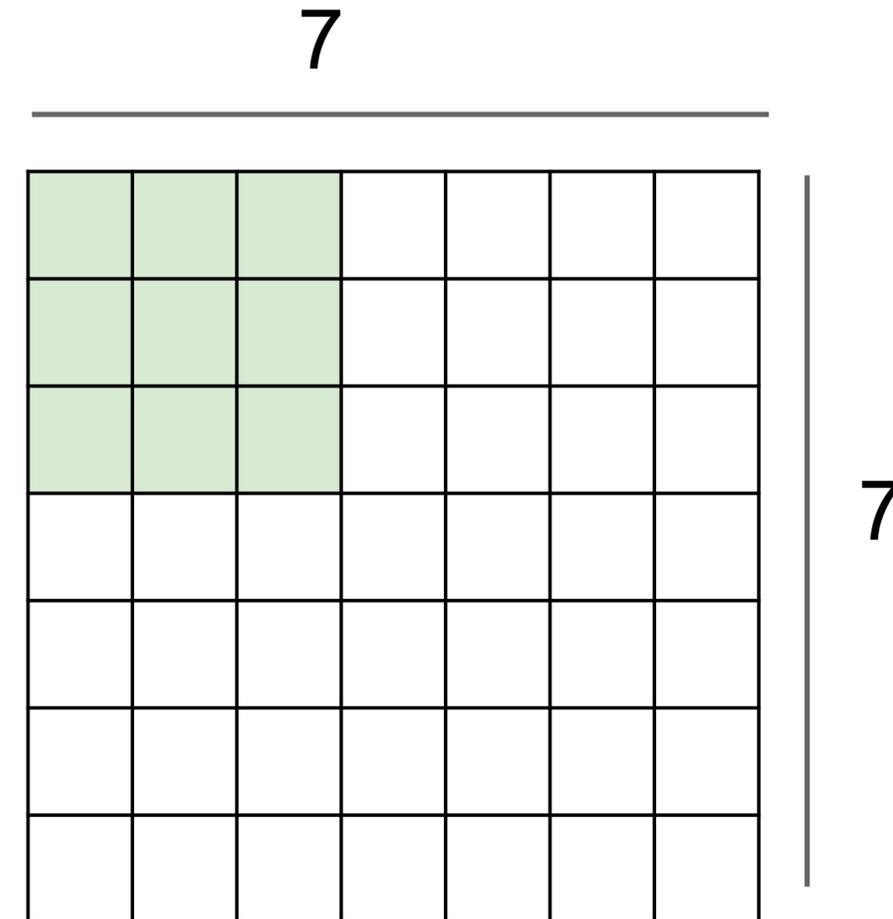
Q2. Suppose we want to perform convolution on a single channel image of size 7×7 (no padding) with a kernel of size 3×3 , and stride = 2. What is the dimension of the output?

A. 3×3

B. 7×7

C. 5×5

D. 2×2



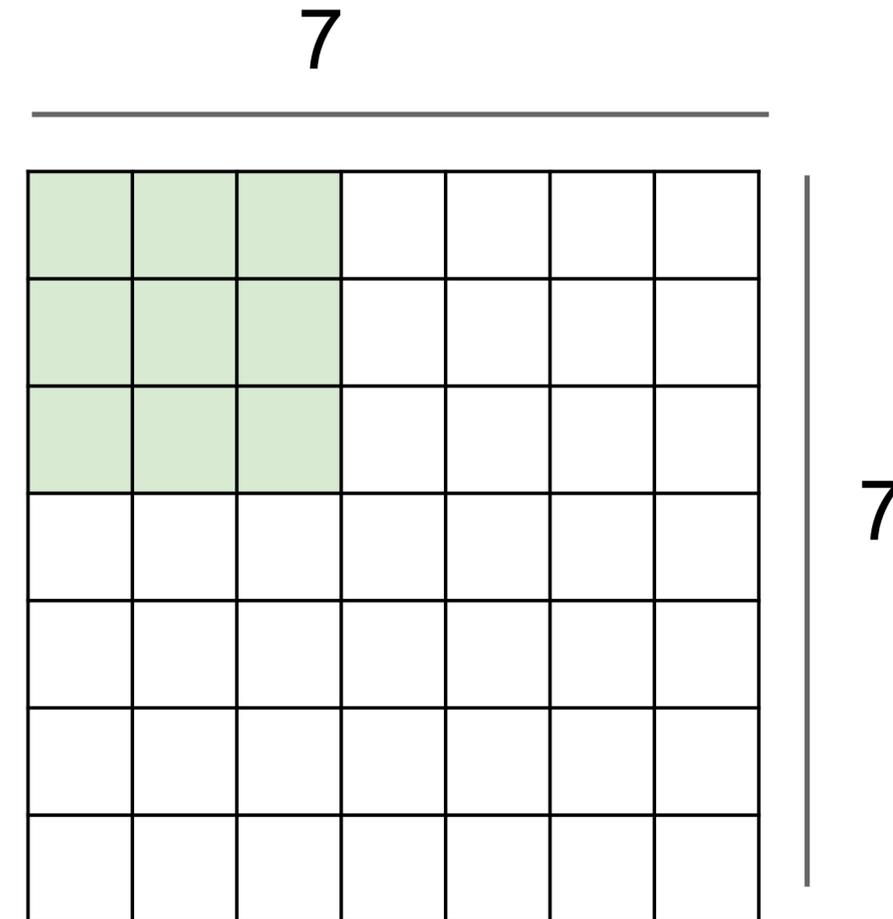
Q2. Suppose we want to perform convolution on a single channel image of size 7x7 (no padding) with a kernel of size 3x3, and stride = 2. What is the dimension of the output?

A. 3x3

B. 7x7

C. 5x5

D. 2x2



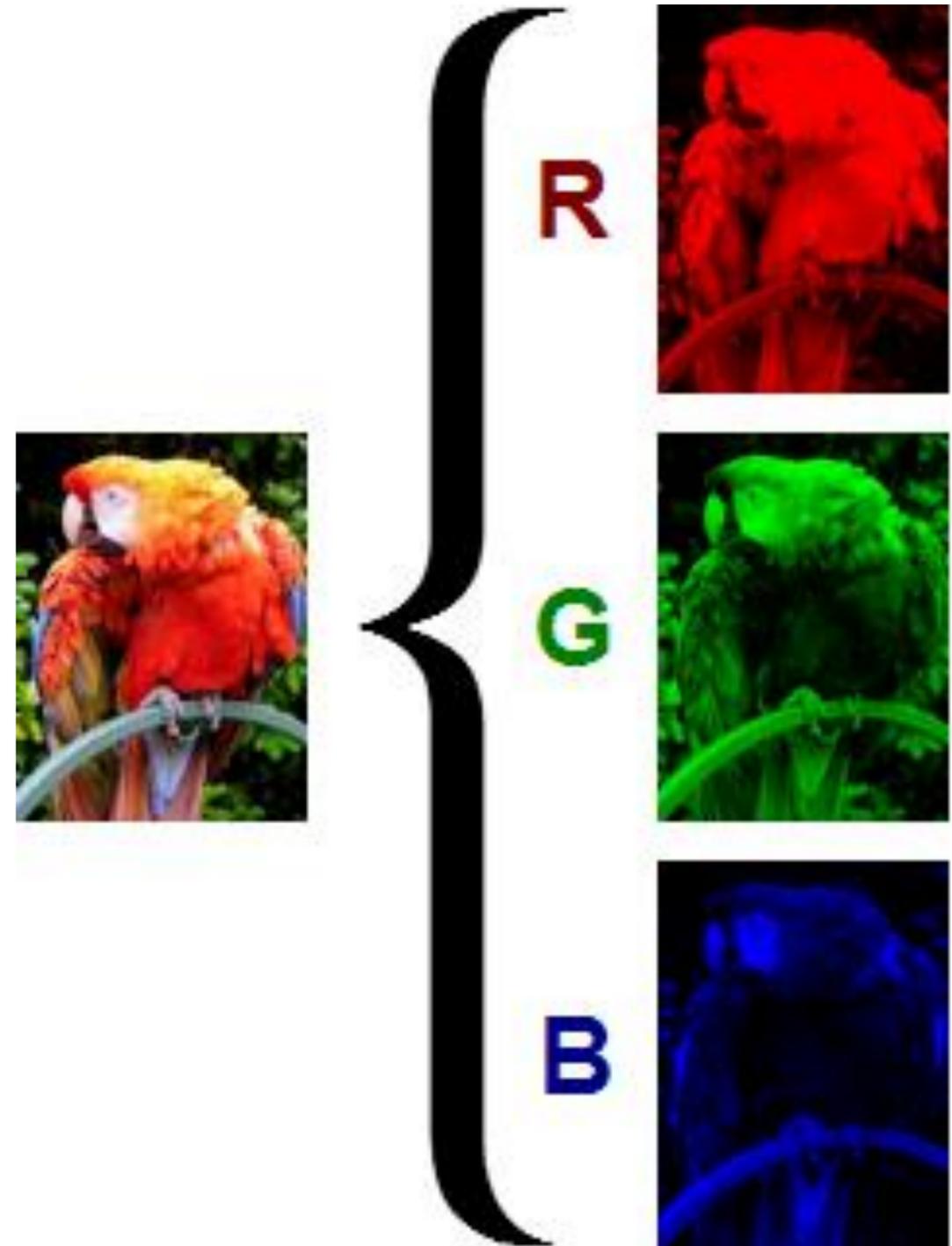
$$\lfloor (n_h - k_h + p_h + s_h) / s_h \rfloor \times \lfloor (n_w - k_w + p_w + s_w) / s_w \rfloor$$



Multiple Input and Output Channels

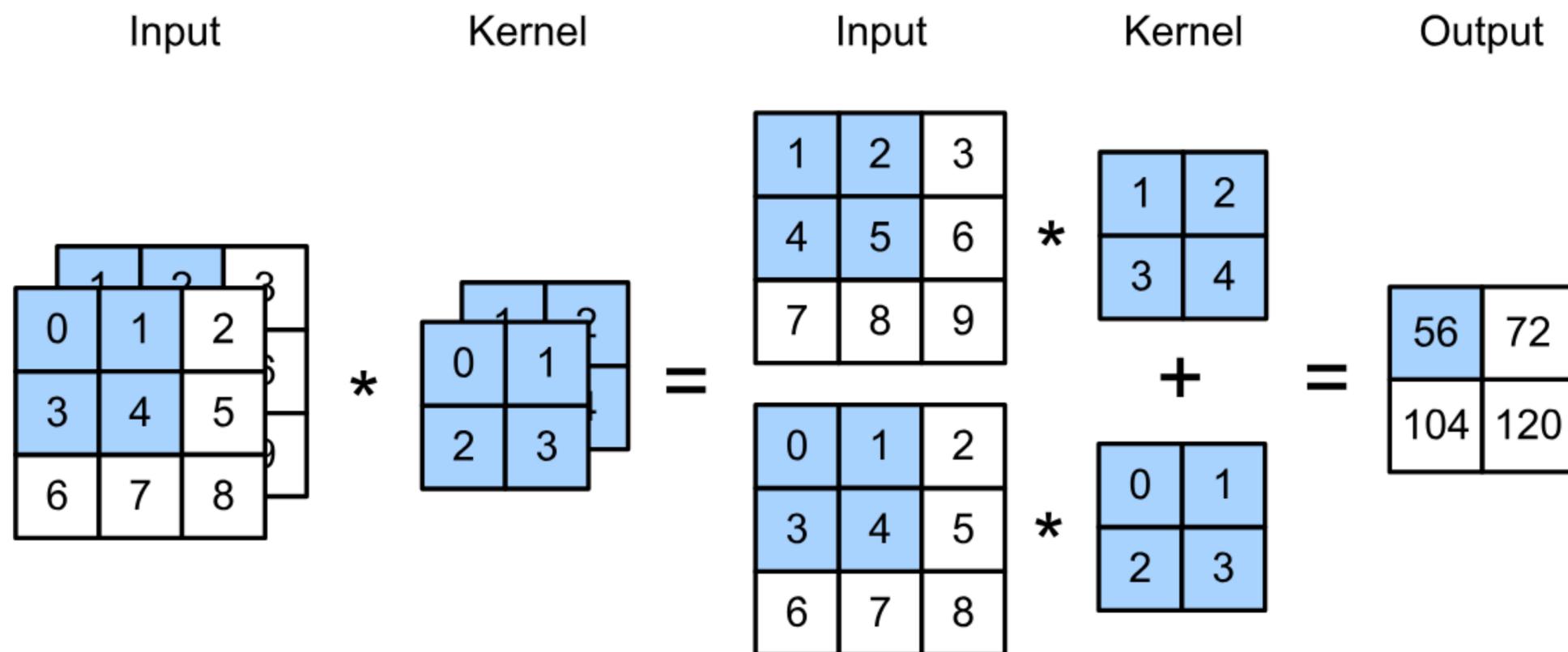
Multiple Input Channels

- Color image may have three RGB channels
- Converting to grayscale loses information



Multiple Input Channels

- Have a kernel matrix for each channel, and then sum results over channels



$$(1 \times 1 + 2 \times 2 + 4 \times 3 + 5 \times 4) \\ + (0 \times 0 + 1 \times 1 + 3 \times 2 + 4 \times 3) \\ = 56$$

Convolutional Layers: Channels

- How to integrate multiple channels?
 - Have a kernel for each channel, and then sum results over channels

$$\mathbf{X} : c_i \times n_h \times n_w$$

$$\mathbf{W} : c_i \times k_h \times k_w$$

$$\mathbf{Y} : m_h \times m_w$$

$$\mathbf{Y} = \sum_{i=0}^{c_i} \mathbf{X}_{i,:,:} \star \mathbf{W}_{i,:,:}$$

“Slices” of tensors

Tensor: generalization of matrix to higher dimensions

Multiple Output Channels

- No matter how many inputs channels, so far we always get single output channel
- We can have **multiple 3-D kernels**, each one generates an output channel

Multiple Output Channels

- No matter how many input channels, so far we always get single output channel
- We can have **multiple 3-D kernels**, each one generates an output channel

- Input $\mathbf{X} : c_i \times n_h \times n_w$

- Kernels $\mathbf{W} : c_o \times c_i \times k_h \times k_w$

- Output $\mathbf{Y} : c_o \times m_h \times m_w$

$$\mathbf{Y}_{i,:,:} = \mathbf{X} \star \mathbf{W}_{i,:,:,:}$$

for $i = 1, \dots, c_o$

Multiple Input/Output Channels

- Each 3-D kernel may recognize a particular pattern



(Gabor filters)

Q3. Suppose we want to perform convolution on an RGB image of size 224×224 (no padding) with 64 kernels, each with height 3 and width 3. Stride = 1. Which is a reasonable estimate of the total number of scalar multiplications involved in this operation (without considering any optimization in matrix multiplication)?

A. $64 \times 3 \times 3 \times 222 \times 222$

B. $64 \times 3 \times 3 \times 222$

C. $3 \times 3 \times 222 \times 222$

D. $64 \times 3 \times 3 \times 3 \times 222 \times 222$

Q3. Suppose we want to perform convolution on an RGB image of size 224×224 (no padding) with 64 kernels, each with height 3 and width 3. Stride = 1. Which is a reasonable estimate of the total number of scalar multiplications involved in this operation (without considering any optimization in matrix multiplication)?

A. $64 \times 3 \times 3 \times 222 \times 222$

B. $64 \times 3 \times 3 \times 222$

C. $3 \times 3 \times 222 \times 222$

D. $64 \times 3 \times 3 \times 3 \times 222 \times 222$

Q3. Suppose we want to perform convolution on an RGB image of size 224×224 (no padding) with 64 kernels, each with height 3 and width 3. Stride = 1. Which is a reasonable estimate of the total number of scalar multiplications involved in this operation (without considering any optimization in matrix multiplication)?

A. $64 \times 3 \times 3 \times 222 \times 222$

B. $64 \times 3 \times 3 \times 222$

C. $3 \times 3 \times 222 \times 222$

D. $64 \times 3 \times 3 \times 3 \times 222 \times 222$

For each kernel, we slide the window to 222×222 different locations. For each location, the number of multiplication is $3 \times 3 \times 3$. So in total $64 \times 3 \times 3 \times 3 \times 222 \times 222$

Q4. Suppose we want to perform convolution on a RGB image of size 224×224 (no padding) with 64 kernels, each with height 3 and width 3. Stride = 1. The convolution layer has bias parameters. Which is a reasonable estimate of the total number of learnable parameters?

A. $64 \times 222 \times 222$

B. $64 \times 3 \times 3 \times 222$

C. $3 \times 3 \times 3 \times 64$

D. $(3 \times 3 \times 3 + 1) \times 64$

Q4. Suppose we want to perform convolution on a RGB image of size 224×224 (no padding) with 64 kernels, each with height 3 and width 3. Stride = 1. The convolution layer has bias parameters. Which is a reasonable estimate of the total number of learnable parameters?

A. $64 \times 222 \times 222$

B. $64 \times 3 \times 3 \times 222$

C. $3 \times 3 \times 3 \times 64$

D. $(3 \times 3 \times 3 + 1) \times 64$

Q4. Suppose we want to perform convolution on a RGB image of size 224×224 (no padding) with 64 kernels, each with height 3 and width 3. Stride = 1. The convolution layer has bias parameters. Which is a reasonable estimate of the total number of learnable parameters?

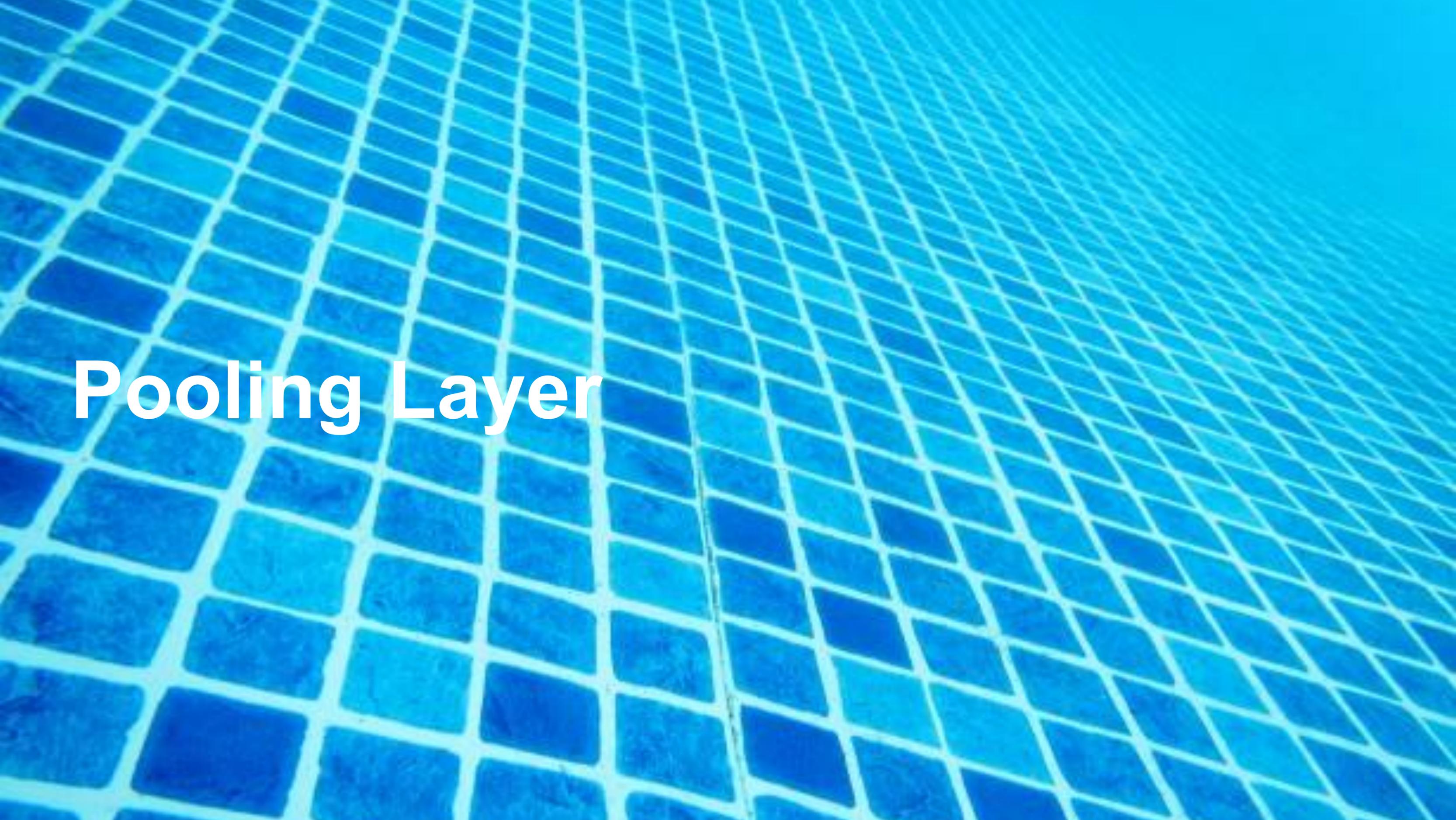
A. $64 \times 222 \times 222$

B. $64 \times 3 \times 3 \times 222$

C. $3 \times 3 \times 3 \times 64$

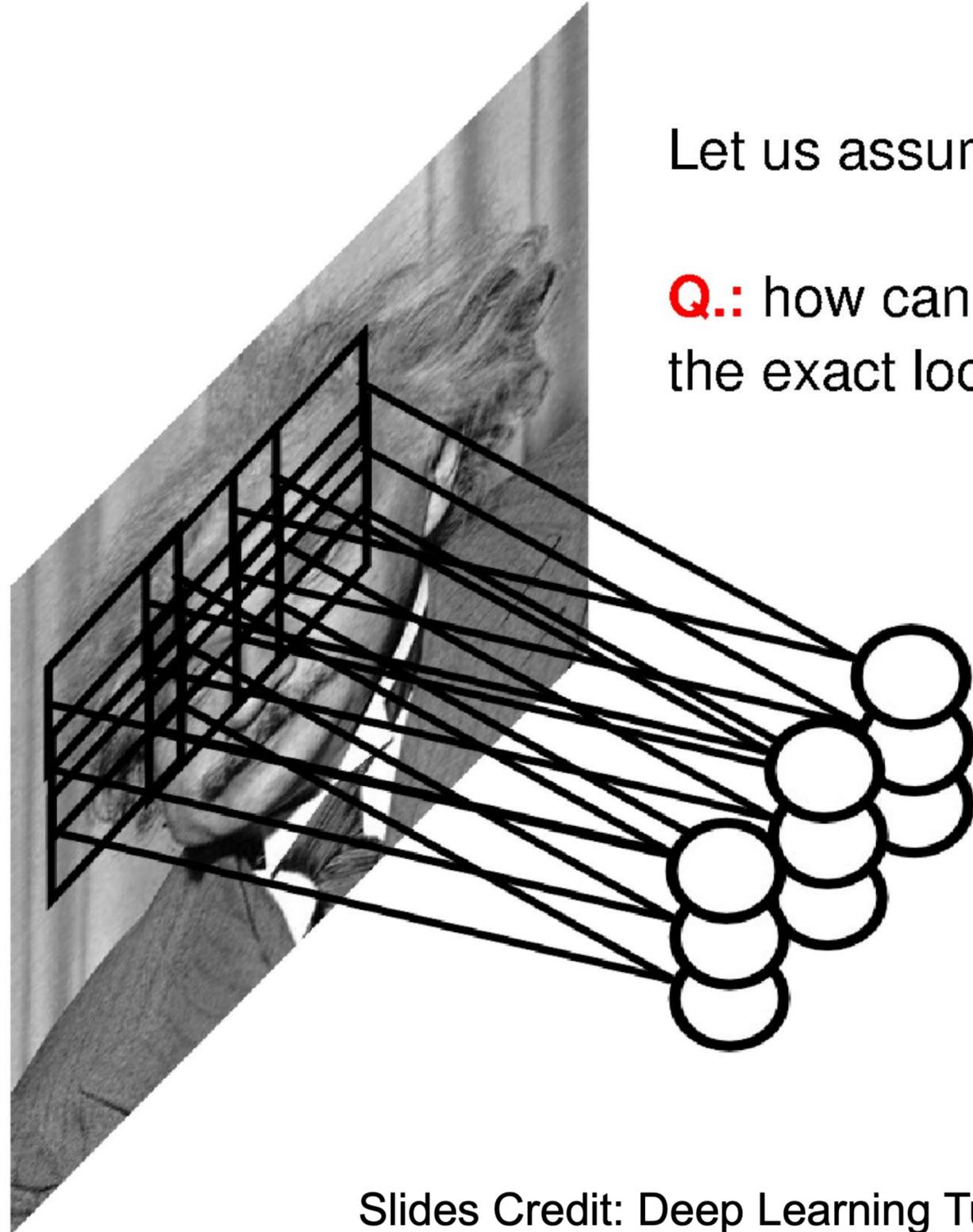
D. $(3 \times 3 \times 3 + 1) \times 64$

Each kernel is 3D kernel across 3 input channels, so has $3 \times 3 \times 3$ parameters. Each kernel has 1 bias parameter. So in total $(3 \times 3 \times 3 + 1) \times 64$



Pooling Layer

Pooling



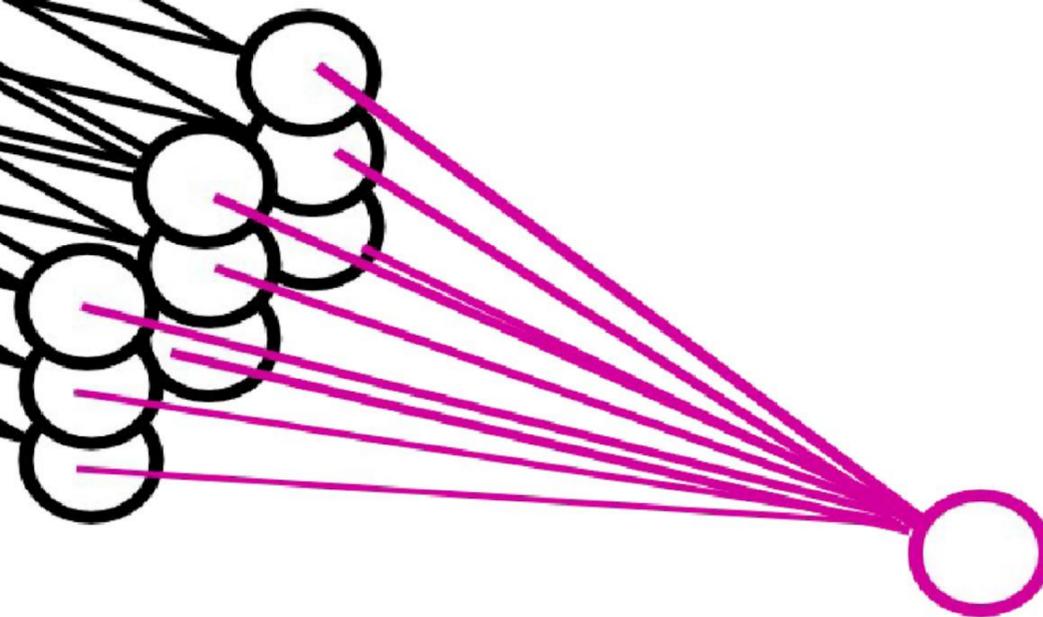
Let us assume filter is an “eye” detector.

Q.: how can we make the detection robust to the exact location of the eye?

Pooling

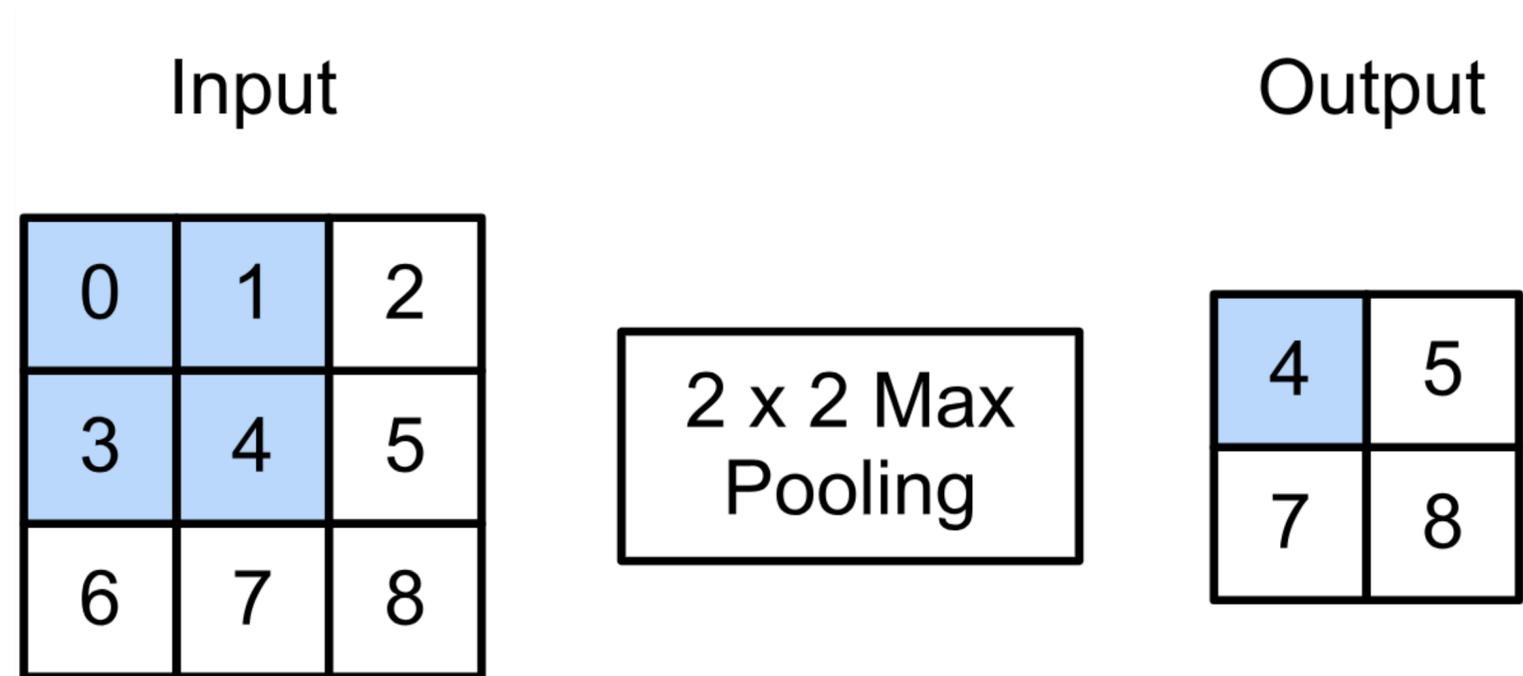


By “pooling” (e.g., taking max) filter responses at different locations we gain robustness to the exact spatial location of features.



2-D Max Pooling

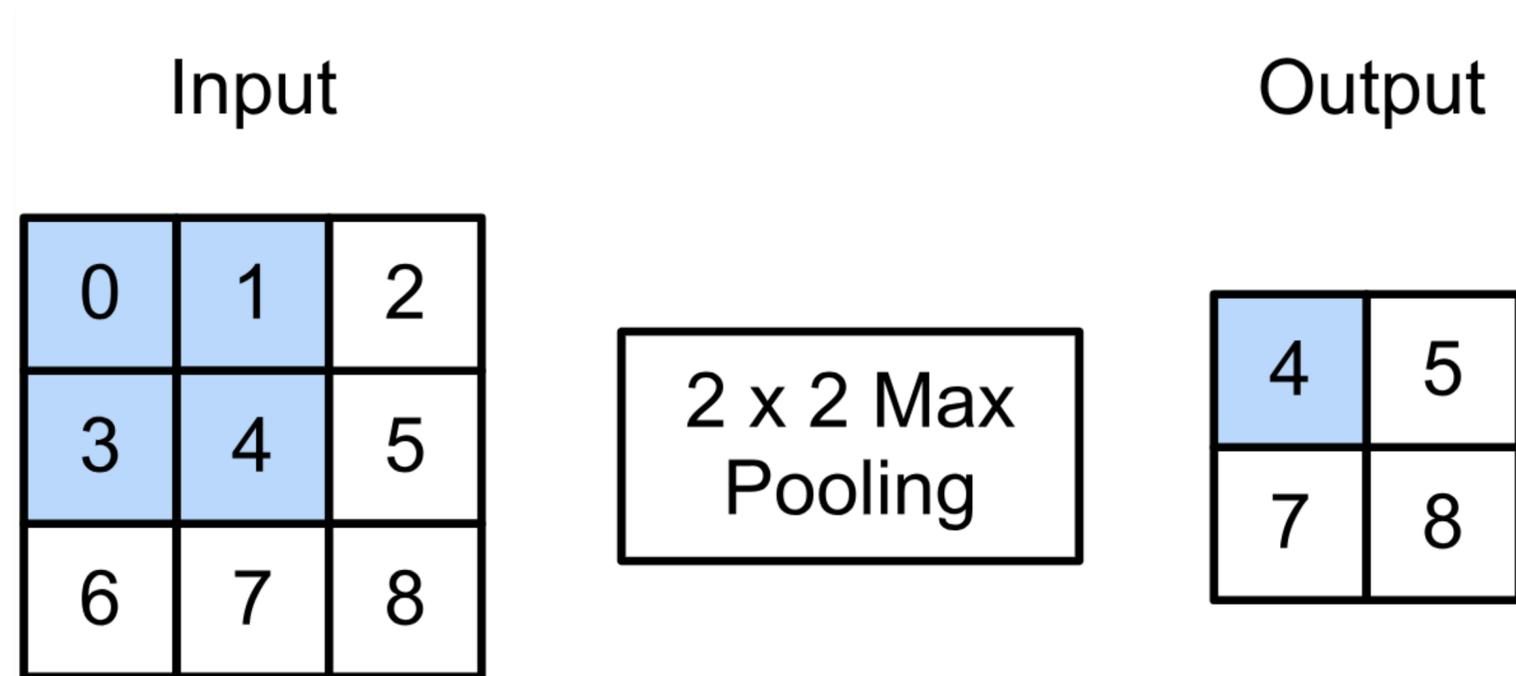
- Returns the maximal value in the sliding window



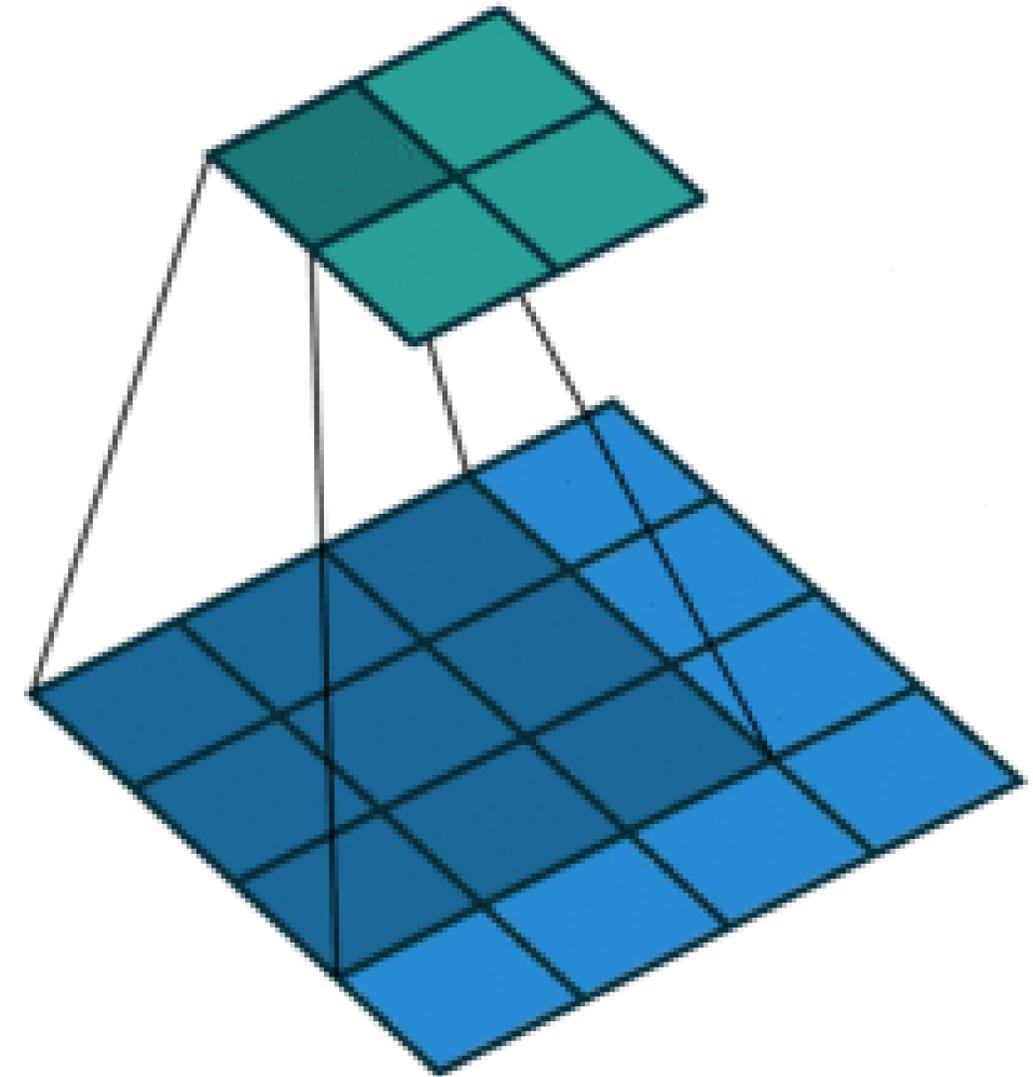
$$\max(0, 1, 3, 4) = 4$$

2-D Max Pooling

- Returns the maximal value in the sliding window



$$\max(0, 1, 3, 4) = 4$$



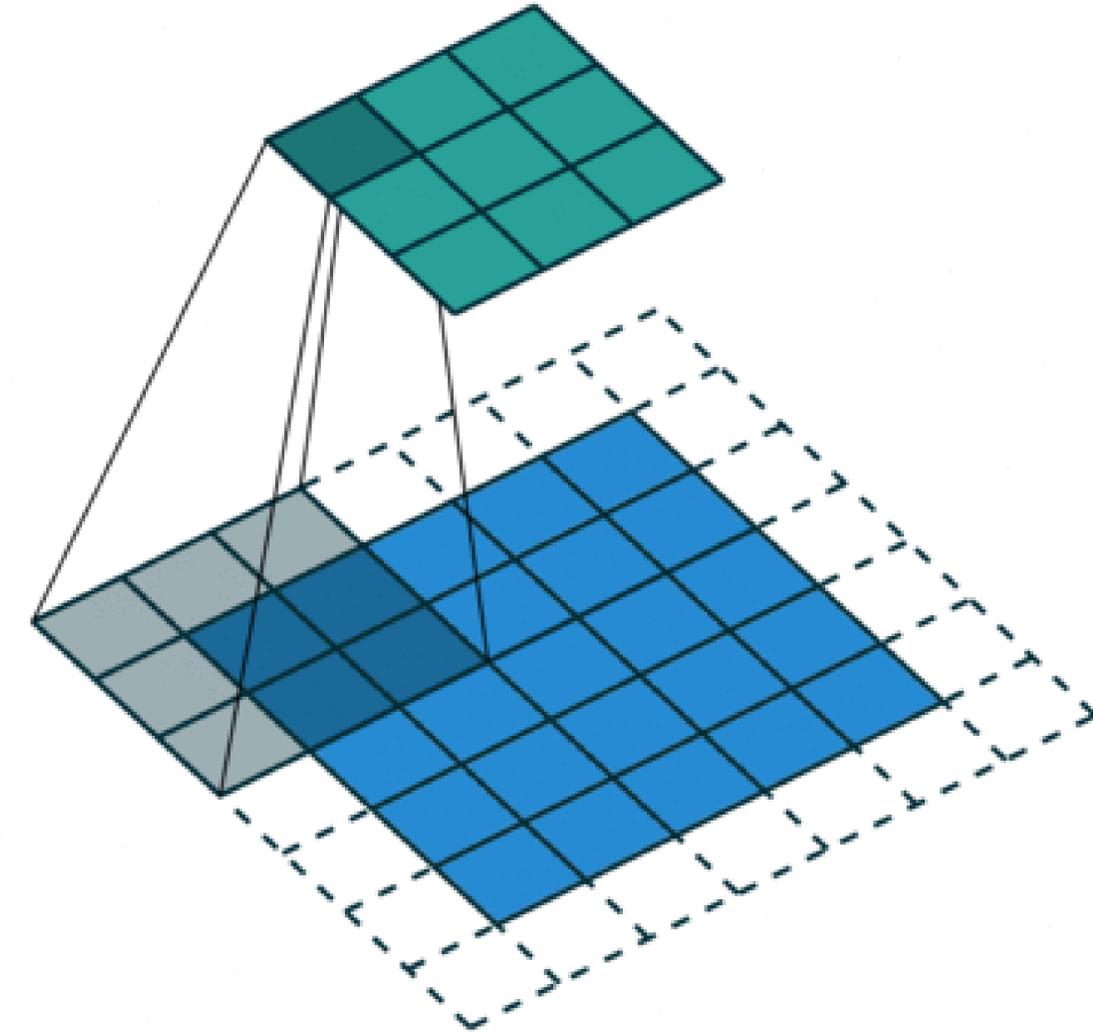
Padding, Stride, and Multiple Channels

- Pooling layers have similar padding and stride as convolutional layers
- No learnable parameters
- Apply pooling for each input channel to obtain the corresponding output channel

#output channels = #input channels

Padding, Stride, and Multiple Channels

- Pooling layers have similar padding and stride as convolutional layers
- No learnable parameters
- Apply pooling for each input channel to obtain the corresponding output channel

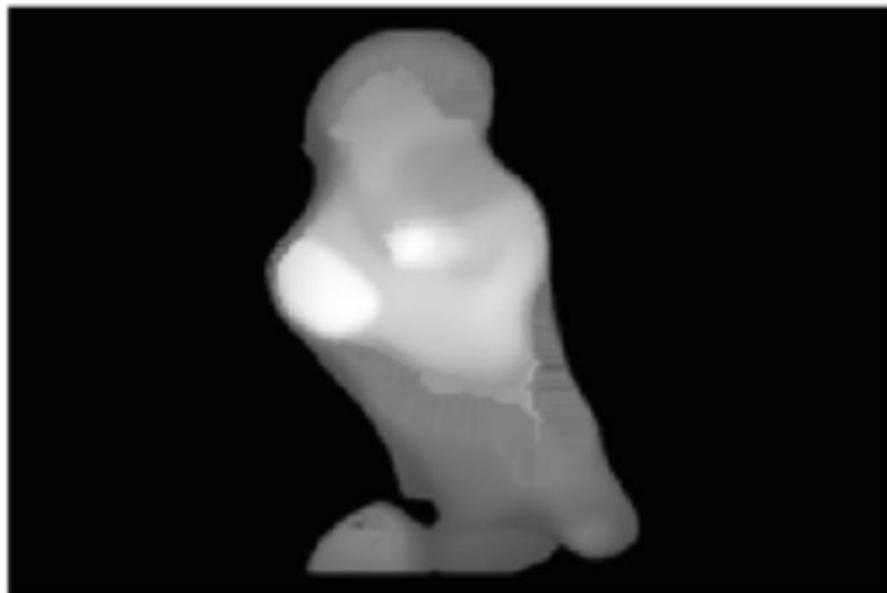


#output channels = #input channels

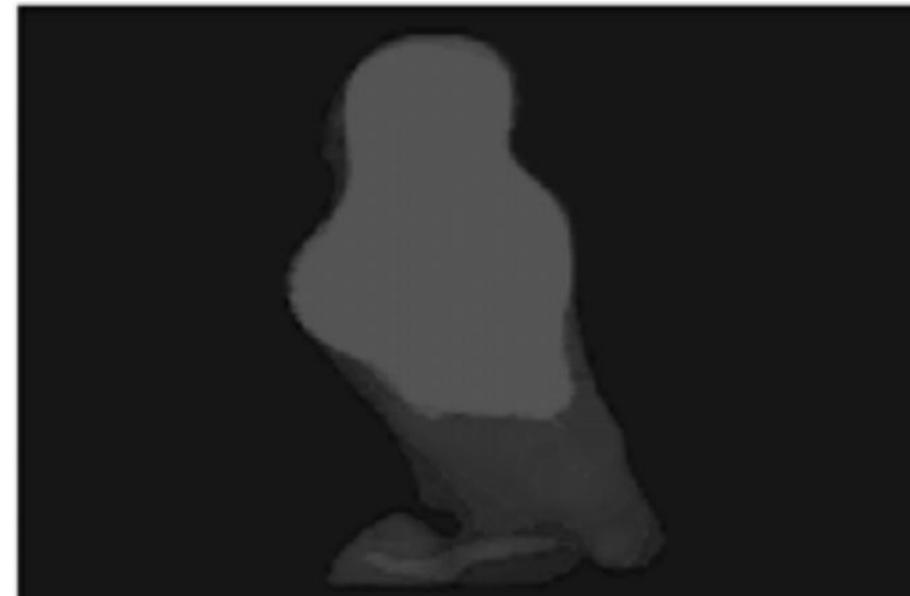
Average Pooling

- Max pooling: the strongest pattern signal in a window
- Average pooling: replace max with mean in max pooling
- The average signal strength in a window

Max pooling



Average pooling



Q5. Suppose we want to perform 2x2 average pooling on the following single channel feature map of size 4x4 (no padding), and stride = 2. What is the output?

A.

20	30
70	90

B.

16	8
20	25

C.

20	30
20	25

D.

12	2
70	5

12	20	30	0
20	12	2	0
0	70	5	2
8	2	90	3

Q5. Suppose we want to perform 2x2 average pooling on the following single channel feature map of size 4x4 (no padding), and stride = 2. What is the output?

A.

20	30
70	90

B.

16	8
20	25

C.

20	30
20	25

D.

12	2
70	5

12	20	30	0
20	12	2	0
0	70	5	2
8	2	90	3

Q6. What is the output if we replace average pooling with 2 x 2 max pooling (other settings are the same)?

A.

20	30
70	90

B.

16	8
20	25

C.

20	30
20	25

D.

12	2
70	5

12	20	30	0
20	12	2	0
0	70	5	2
8	2	90	3

Q6. What is the output if we replace average pooling with 2 x 2 max pooling (other settings are the same)?

A.

20	30
70	90

B.

16	8
20	25

C.

20	30
20	25

D.

12	2
70	5

12	20	30	0
20	12	2	0
0	70	5	2
8	2	90	3

Summary

- Intro of convolutional computations
 - 2D convolution
 - Padding, stride
 - Multiple input and output channels
 - Pooling

Suggested Reading

Textbook: Artificial Intelligence: A Modern Approach (4th edition).
Stuart Russell and Peter Norvig. Pearson, 2020. Section 21.3



Acknowledgement:

Some of the slides in these lectures have been adapted from materials developed by Alex Smola and Mu Li:

<https://courses.d2l.ai/berkeley-stat-157/index.html>