# Convolutional Networks

Yifan Gao

# Outline

- The Convolution Operation
- Motivation of Convolution
- Pooling
- Example Architecture: AlexNet

# The Convolution Operation

- Mathematical Definition of Convolution (Continues):
  - $s(t) = (x * w)(t) = \int x(a)w(t-a)da$

    Input Kernel

- For example, x(t) represents location signal (with noise). To obtain a less noisy estimate of x(t), we can do this with a weighted function w(a):
  - $s(t) = \int_{t-T}^{t} x(a)w(t-a)da$

# The Convolution Operation

- Mathematical Definition of Convolution (Continues):
  - $s(t) = (x * w)(t) = \int x(a)w(t-a)da$

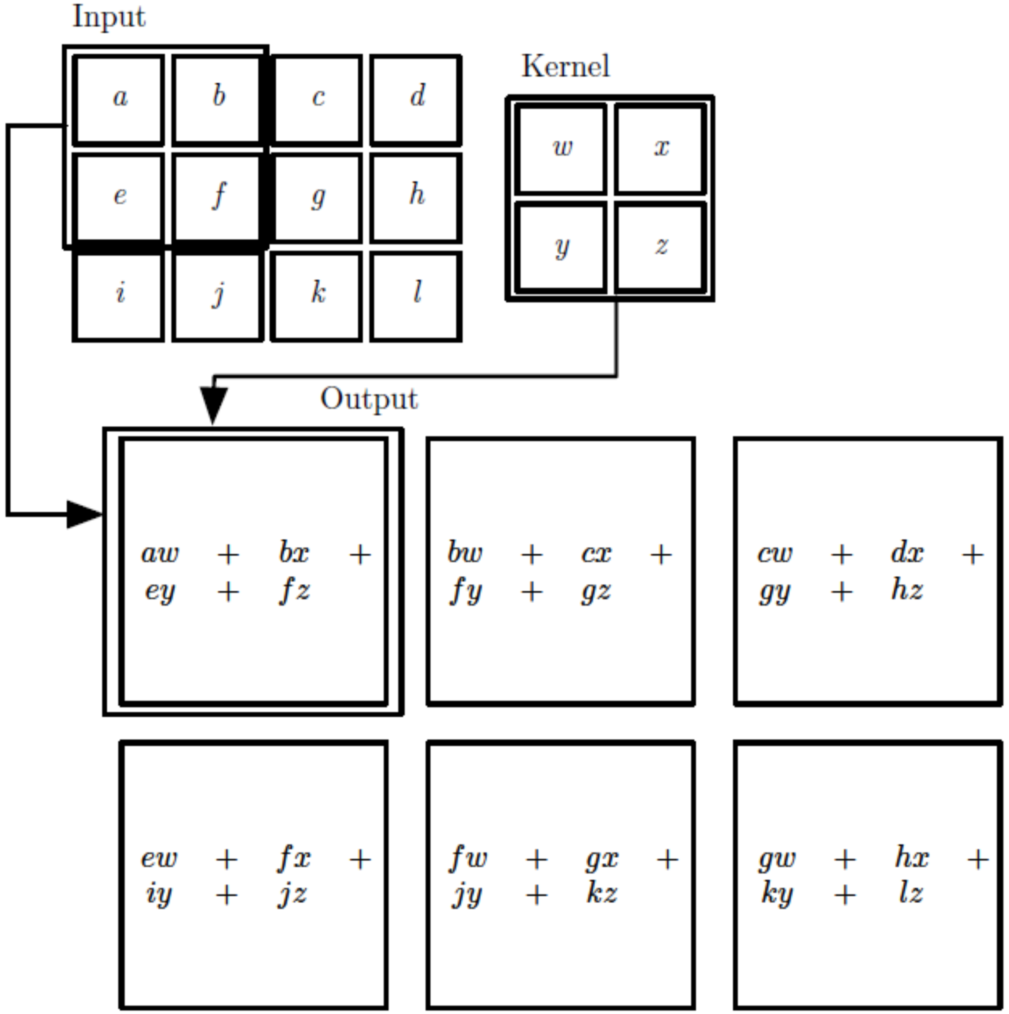
Input Kernel

- Discrete Representation:
  - $s(t) = (x * w)(t) = \Sigma_{-\infty}^{\infty} x(a)w(t-a)$

# The Convolution Operation

- For two-dimensional image I as input, and a two-dimensional kernel K:
  - $S(i,j) = (I * K)(i,j) = \Sigma_m \Sigma_n I(m,n) K(i-m, j-n)$
- Convolution is **commutative**, meaning we can equivalently write:
  - $S(i,j) = (I * K)(i,j) = \Sigma_m \Sigma_n I(i-m, j-n) K(m,n)$
- In fact, many neural network libraries implement a related function called **cross-correlation** but call it convolution…
  - $S(i,j) = (I * K)(i,j) = \Sigma_m \Sigma_n I(i+m, j+n) K(m,n)$

# 2D Convolution



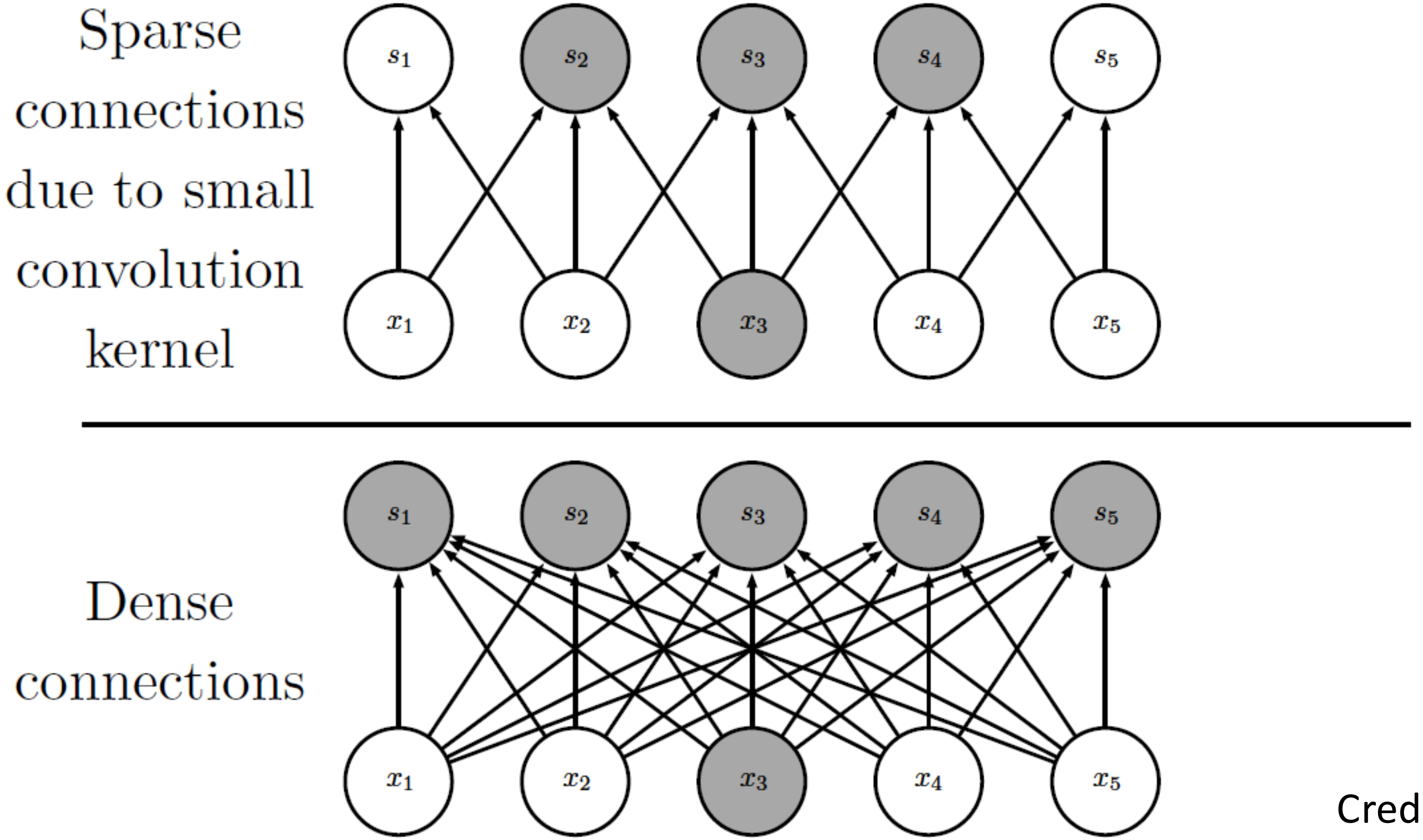

Credit: (Goodfellow 2016)

# Motivation of Convolution

- Sparse Interactions
- Parameter Sharing
- Equivariant Representations

# Sparse Interactions

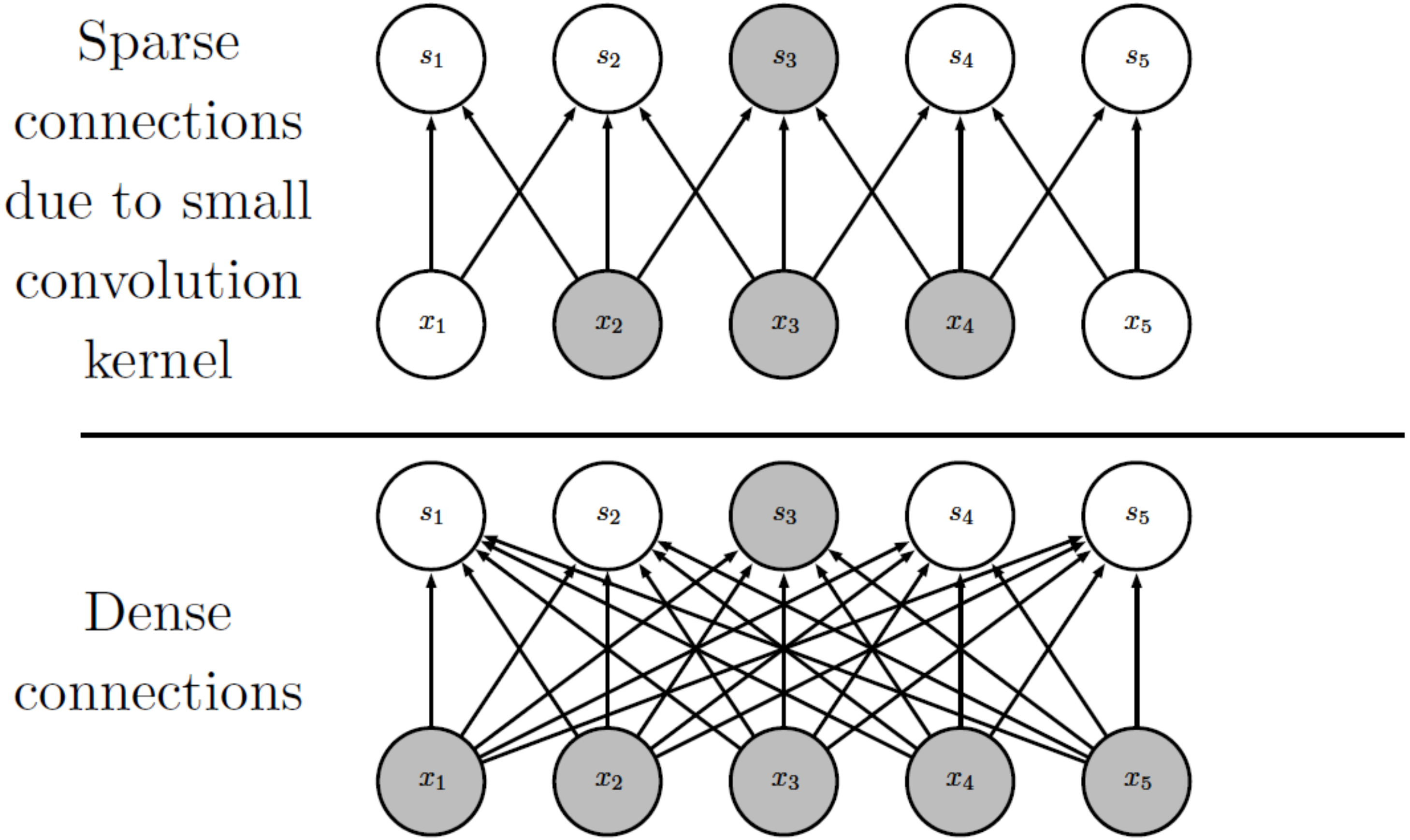
- (Top) Convolutional Network: Only $s_2, s_3, s_4$ are affected by $x_3$
- (Bottom) Fully Connected Network: All the outputs are affected by $x_3$




Credit: (Goodfellow 2016)

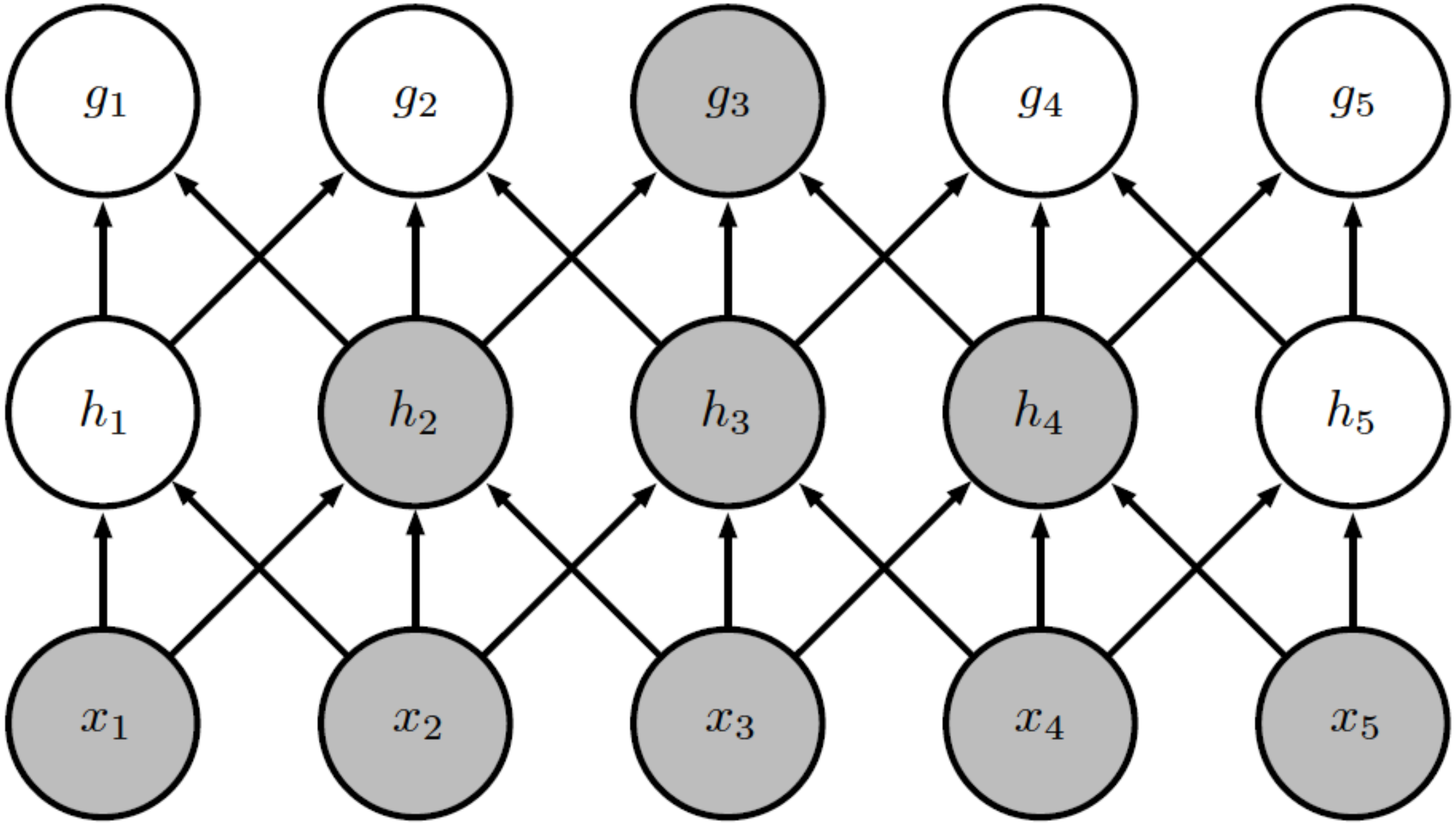
# Sparse Interactions

- (Top) **Respective Field** of $s_3$: $x_2, x_{3,}x_4$
- (Bottom) **Respective Field** of $s_3$: $x_1, x_2, x_{3,}x_4, x_5, x_6$



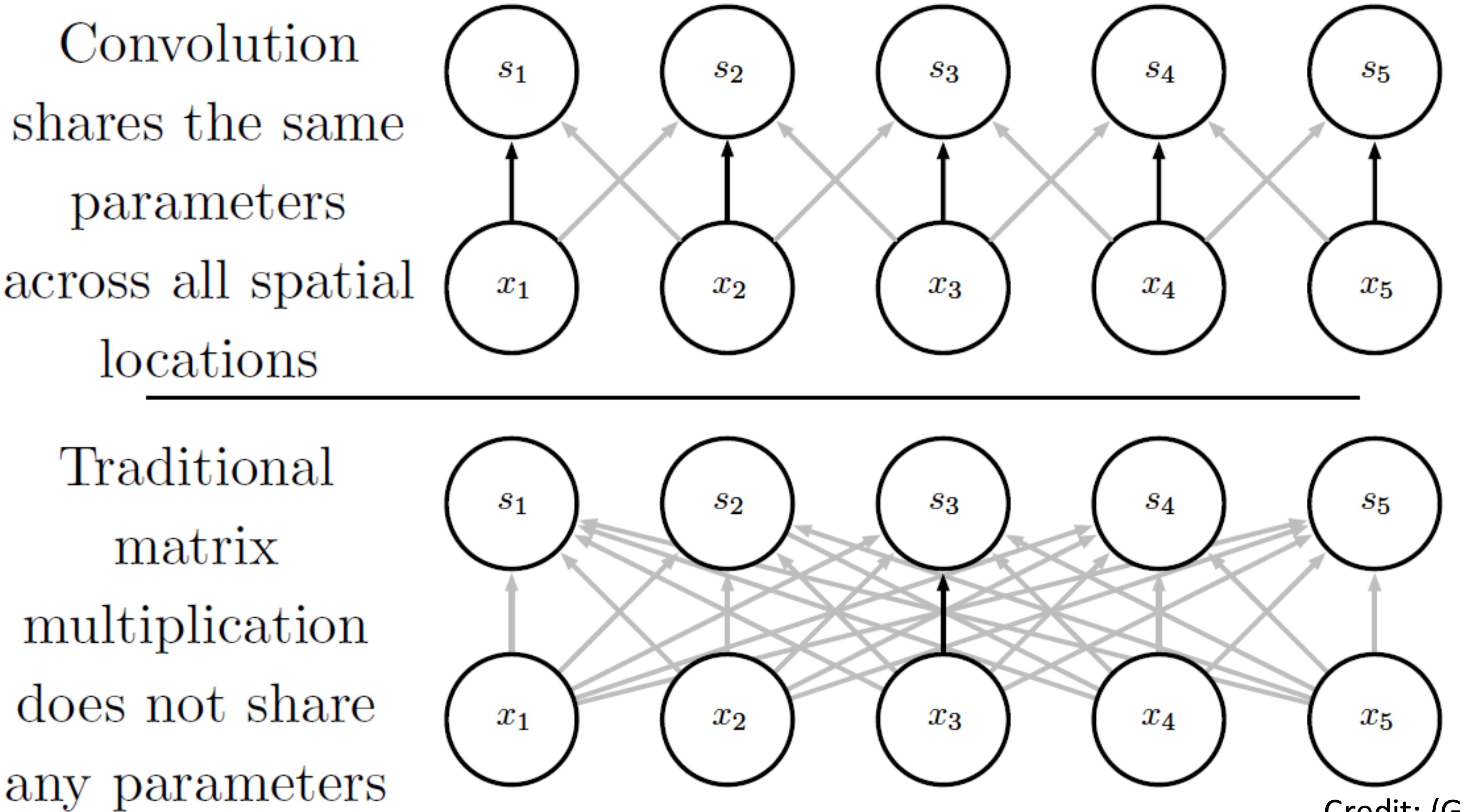

Credit: (Goodfellow 2016)

# Sparse Interactions: Growing Receptive Fields



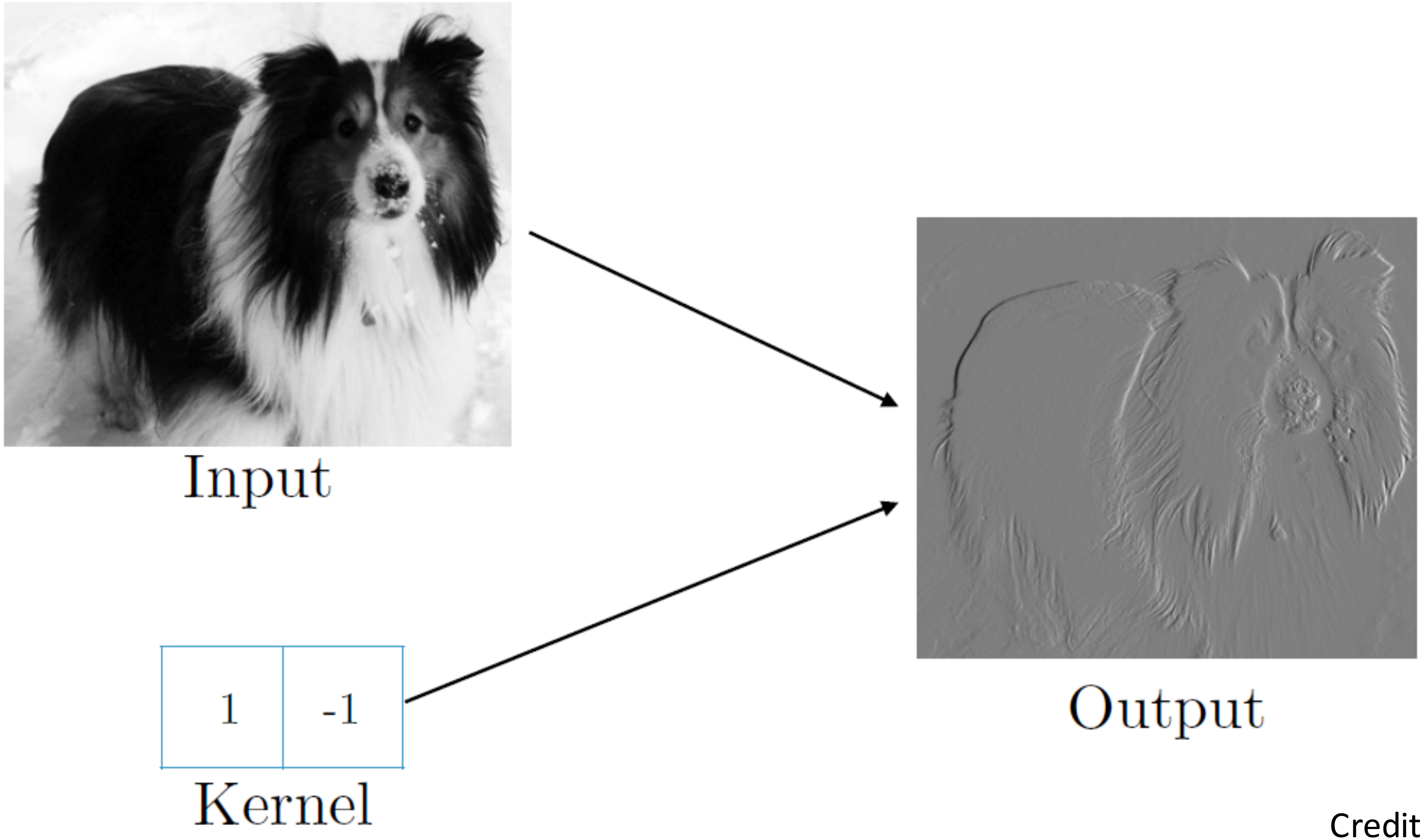

Credit: (Goodfellow 2016)

# Parameter Sharing




Credit: (Goodfellow 2016)

# Example: Edge Detection by Convolution




Credit: (Goodfellow 2016)

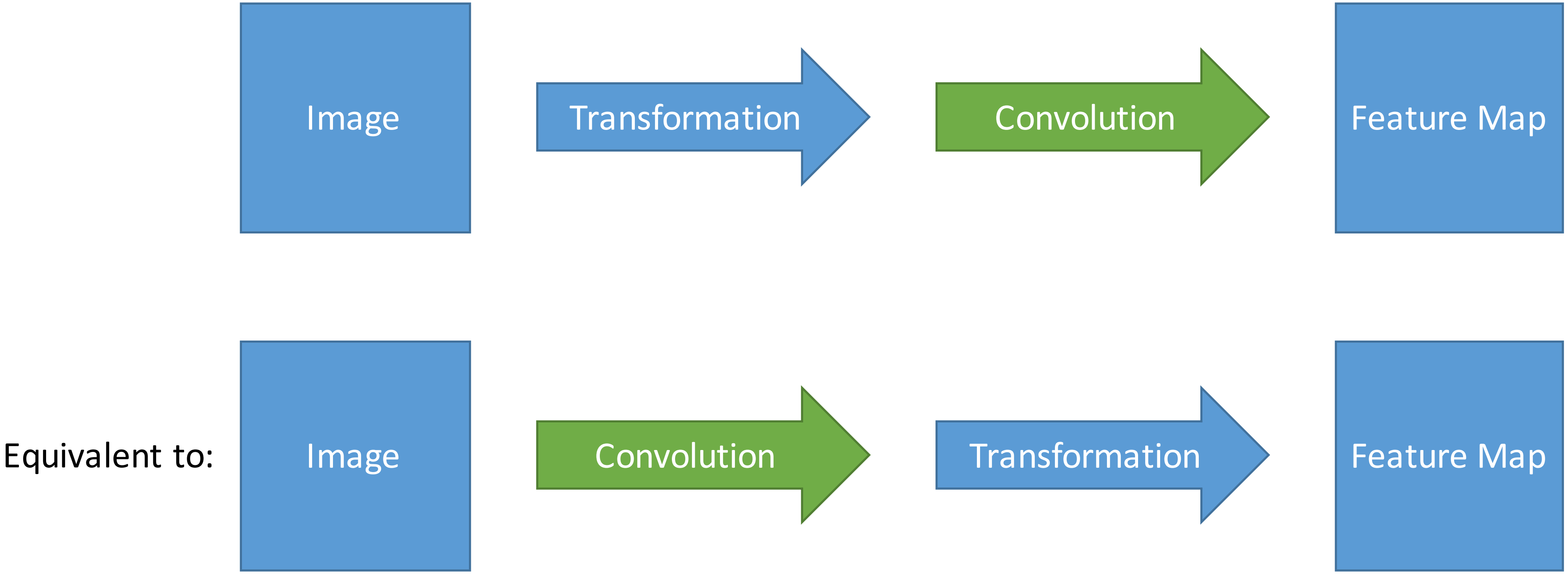
# Efficiency of Convolution

- Input size: 320 by 280
- Kernel size: 2 by 1
- Output size: 319 by 280

| | Convolution | Dense matrix | Sparse matrix |
|---|---|---|---|
| **Stored floats** | 2 | 319*280*320*280 > 8e9 | 2*319*280 = 178,640 |
| **Float muls or adds** | 319*280*3 = 267,960 | > 16e9 | Same as convolution (267,960) |

Credit: (Goodfellow)

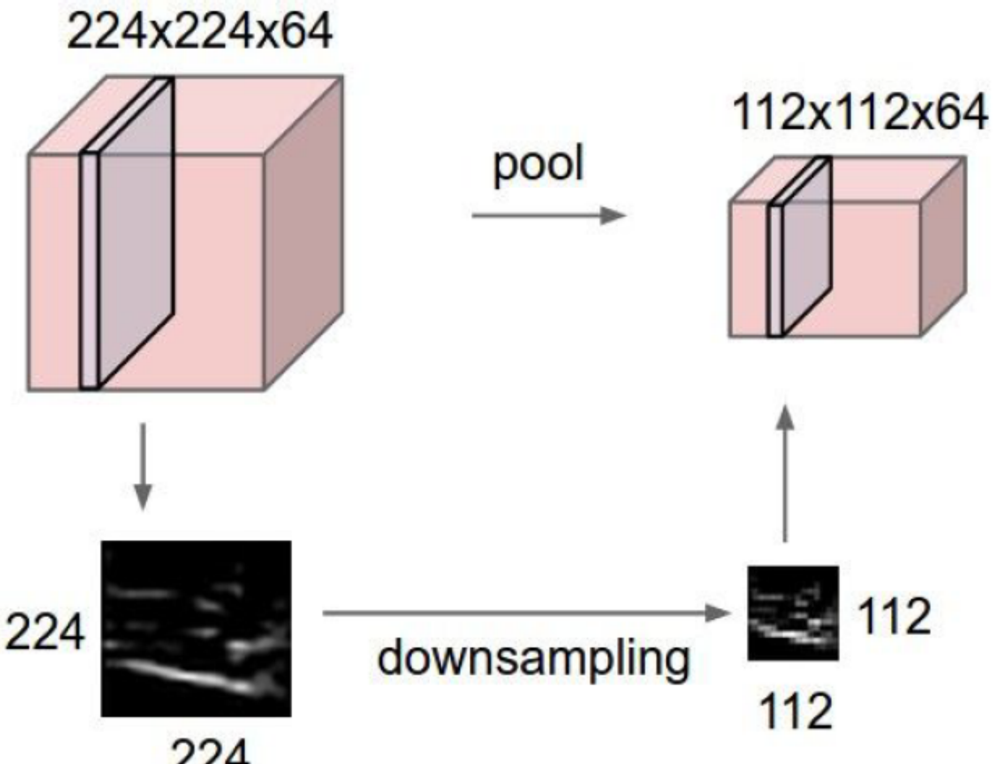
# Equivariant Representations




- For example, when processing images, it is useful to detect edges in different regions of the image.

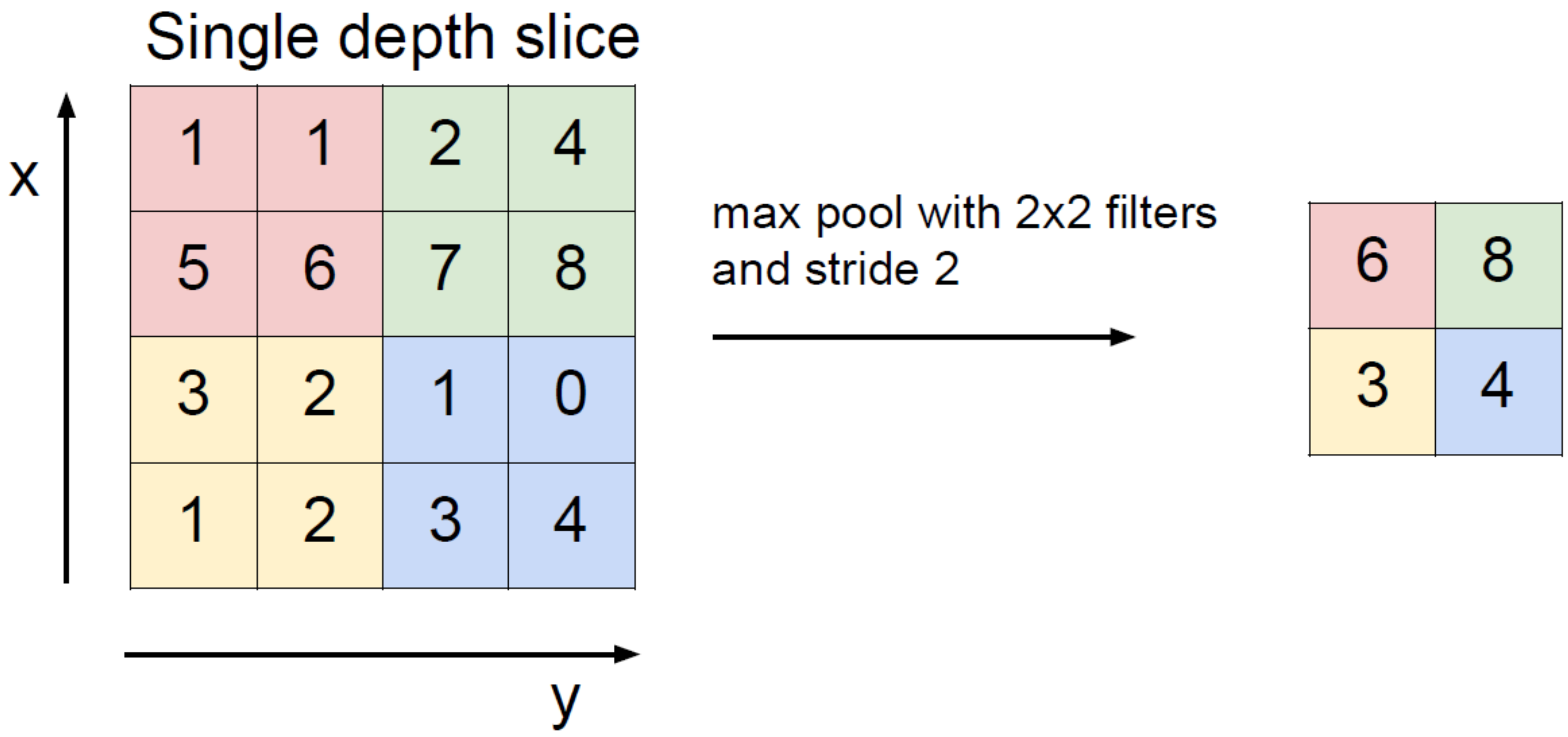
# Pooling

- makes the representations smaller and more manageable
- operates over each activation map independently




Credit: Fei-Fei Li

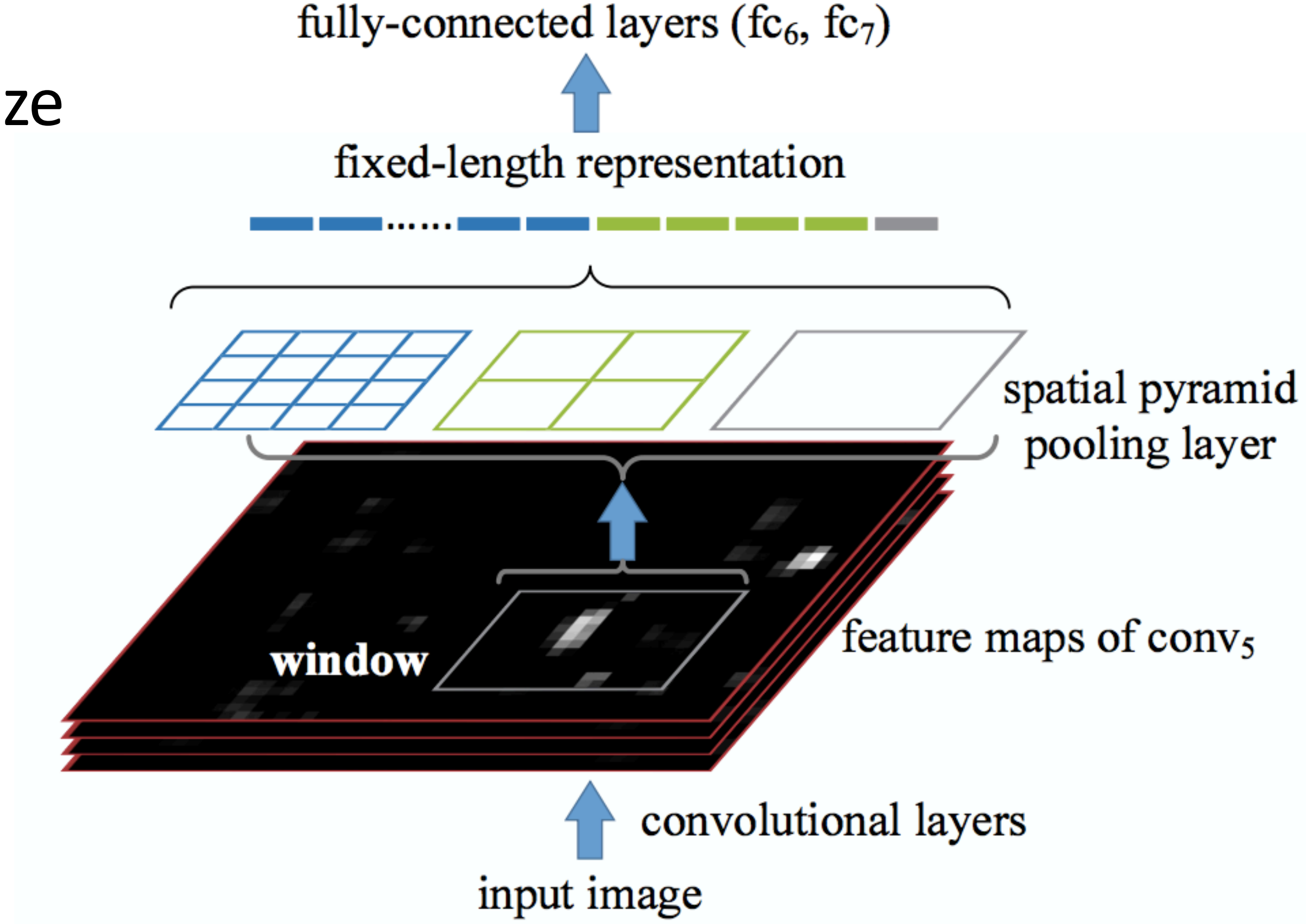
# Max Pooling




Credit: Fei-Fei Li

# Why Pooling

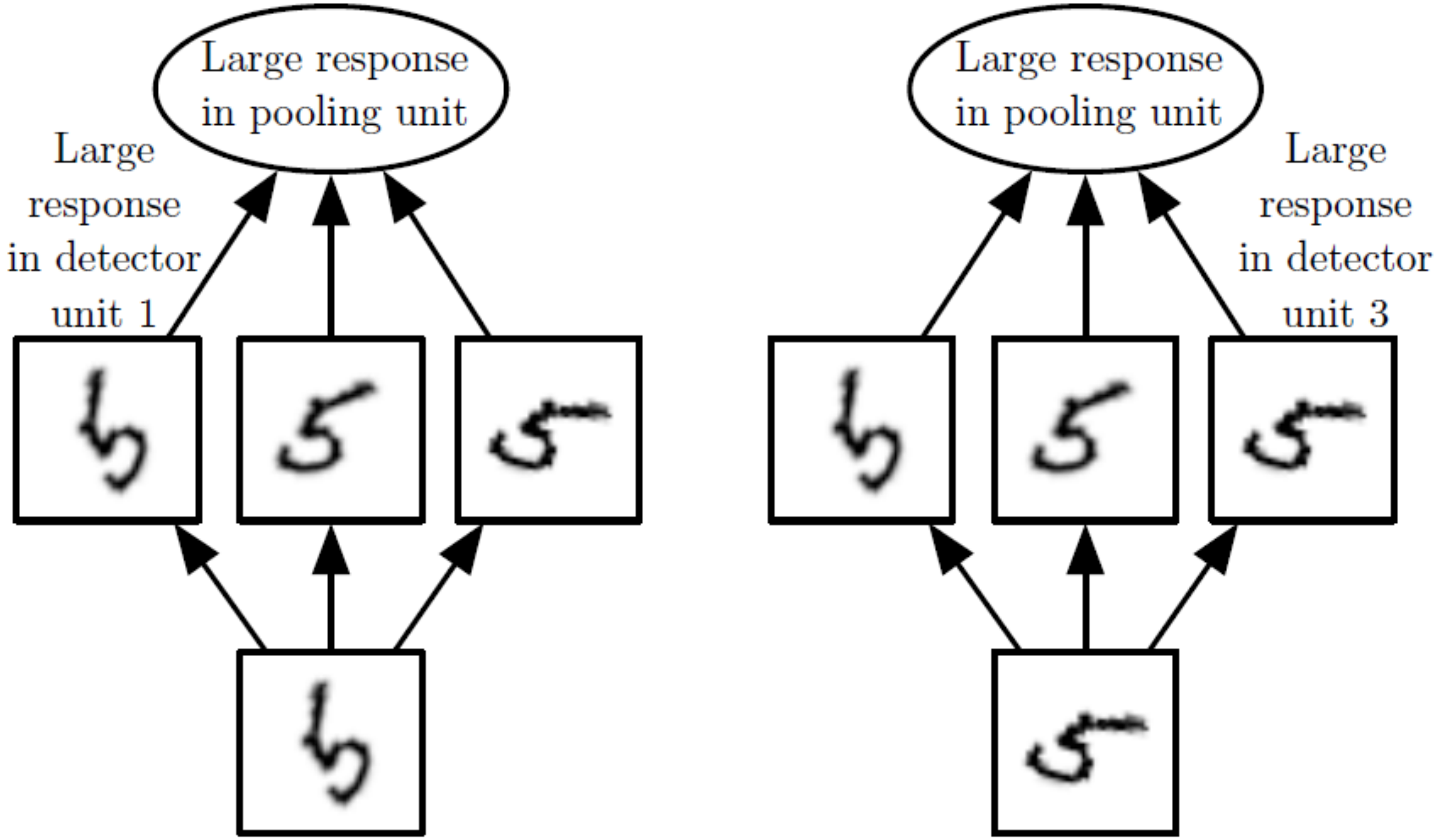
- Invariant Representation
  - It is very useful if we care more about whether some feature is present than exactly where it is.
- Reducing the representation size
  - Efficiency
  - Handling inputs of varying size




Credit: Kaiming He ECCV14

# Why Pooling

If we pool over the outputs of separately parametrized convolutions, the features can learn which transformations to become invariant to.



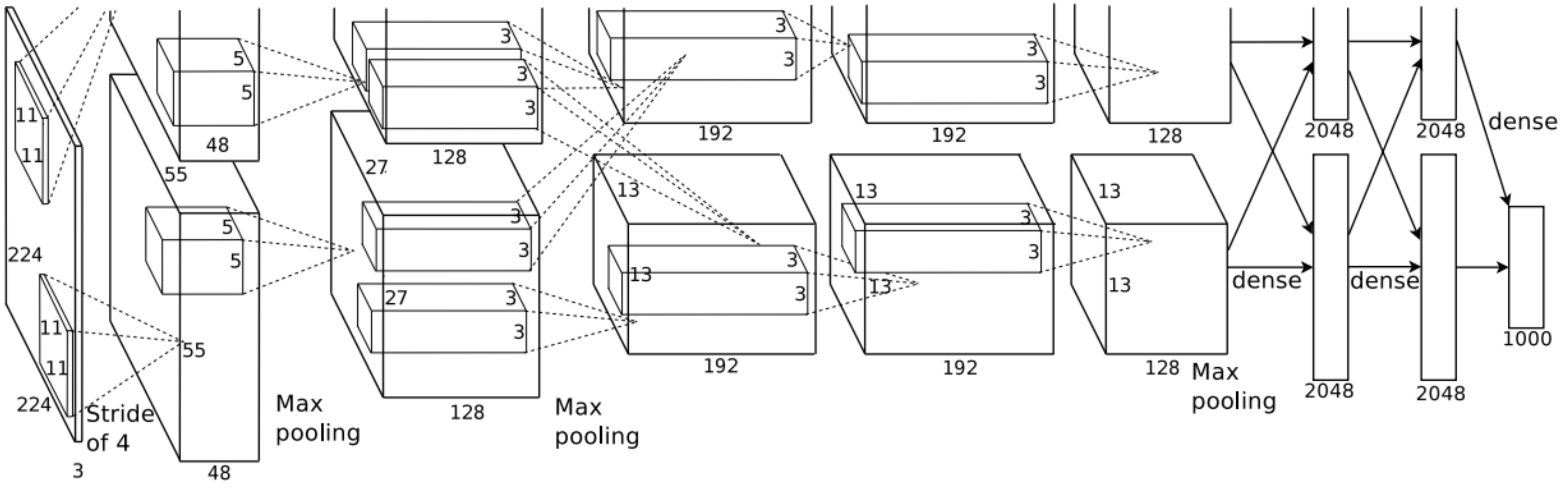

Credit: (Goodfellow 2016)

# Example Classification Architectures: AlexNet

*[Krizhevsky et al. NIPS 2012]*

**Architecture:**

CONV1
MAX POOL1
NORM1
CONV2
MAX POOL2
NORM2
CONV3
CONV4
CONV5
Max POOL3
FC6
FC7
FC8



# Q&A

Thanks