Deep Learning Essentials

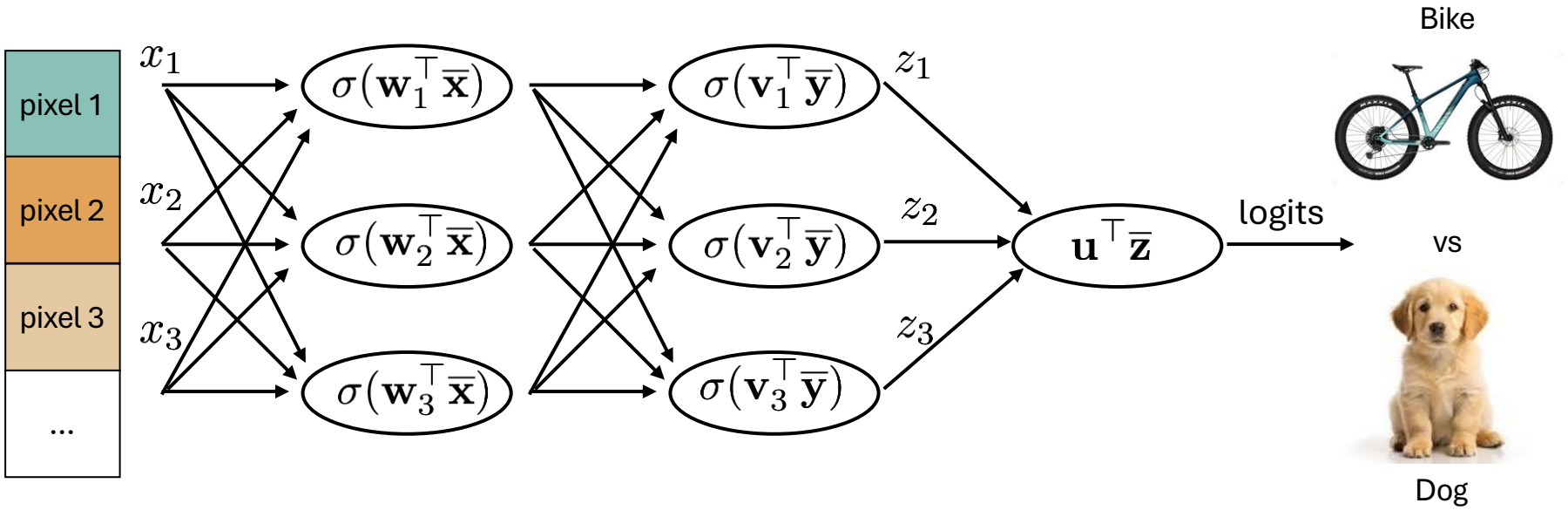
5. Convolutional Neural Networks (CNNs)

The story of a cat's brain surgery

Lukáš Neumann

Adapted from B3B33UROB slides of Karel Zimmerman

Multi-layer perceptron (MLP)

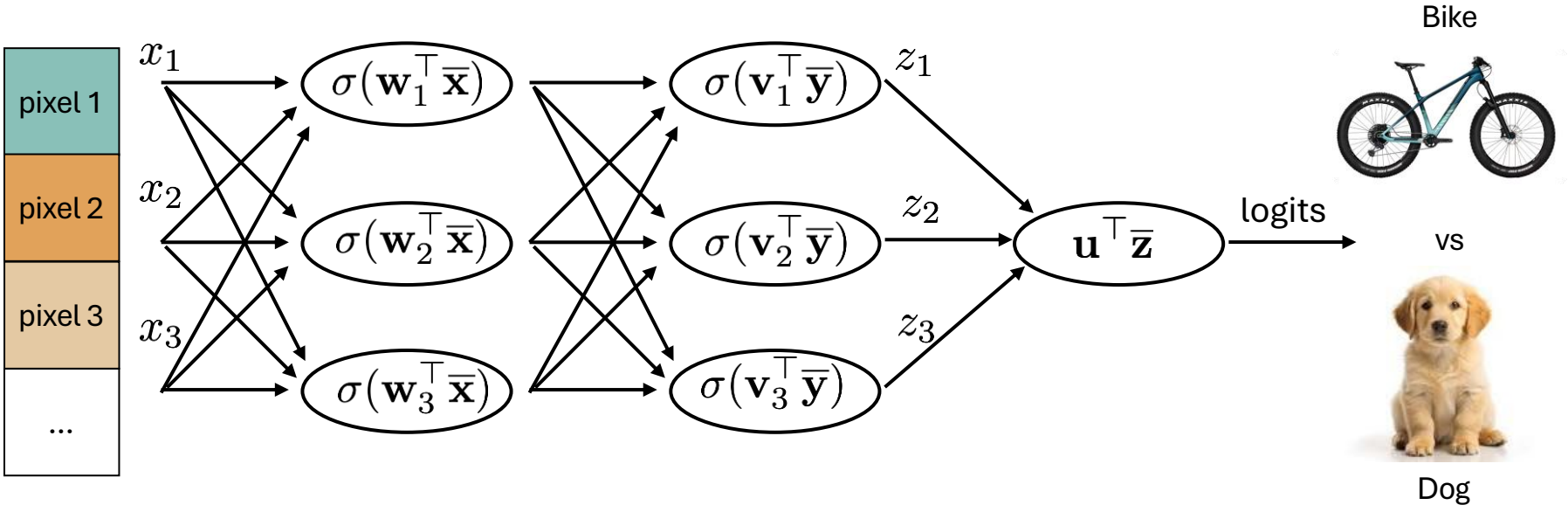


Multi-layer perceptron (MLP)

- How many inputs would the MLP have?
  The number of pixels!

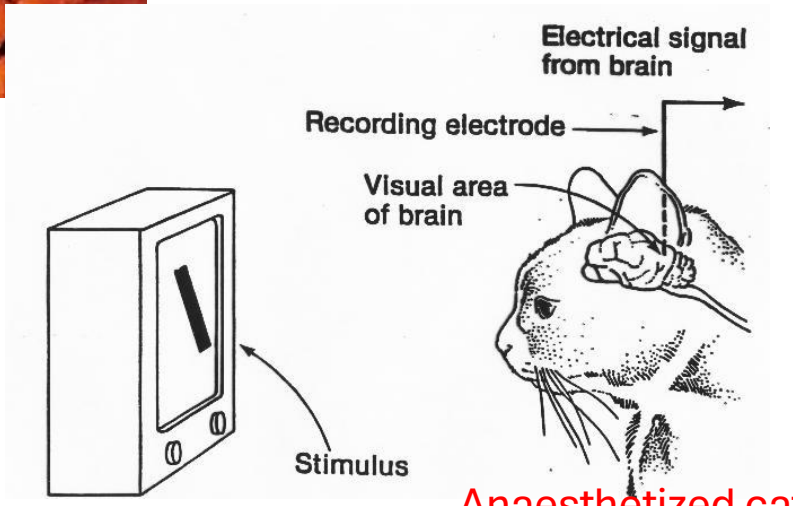
- How many weights would the first layer of the MLP have?
  Number of pixels times the number of neurons!

- Does the position of individual parts (eye, wheel) actually matter?
  No...



Hubel and Wiesel

Hubel, David H. "Tungsten microelectrode for recording from single units." Science (1957)



Anaesthetized cat

Hubel, David H., and Torsten N. Wiesel. "Receptive fields, binocular interaction and functional architecture in the cat's visual cortex." The Journal of physiology (1962)

Hubel and Wiesel

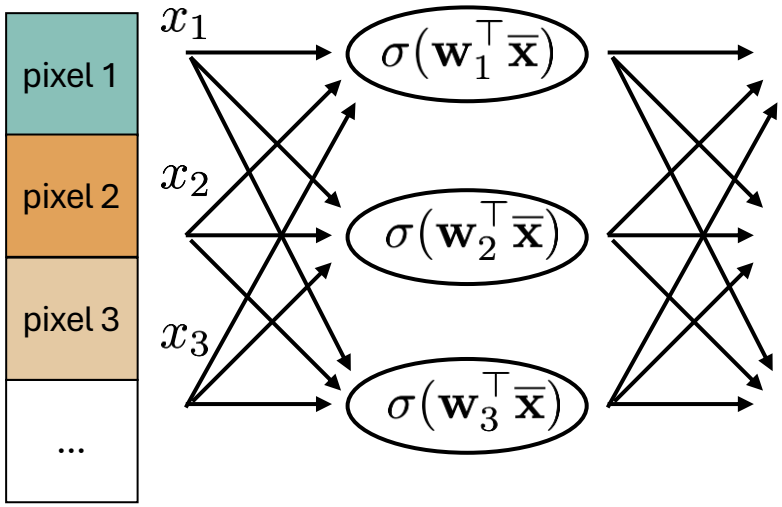
Hubel and Wiesel

- Edge sensitivity
- Nearby neurons process information from nearby visual fields
  $\rightarrow$ Topographical mapping
- The same edge is detected in all positions
  $\rightarrow$ Translation equivariance

Nobel Prize in Physiology and Medicine (1981)

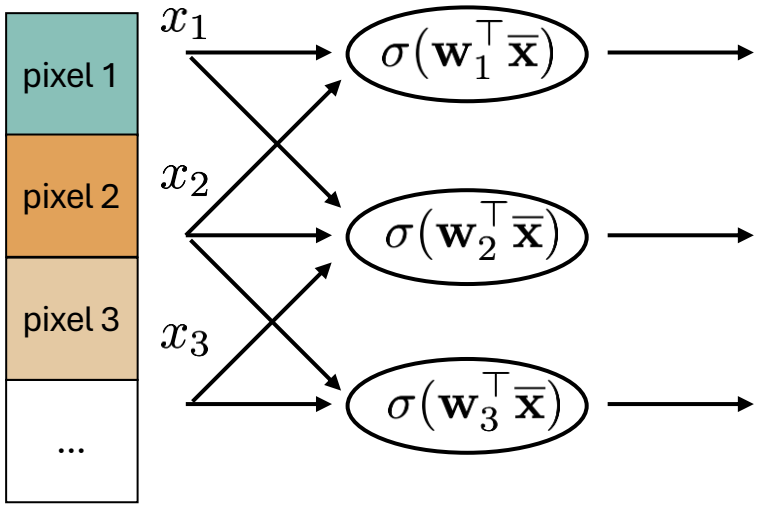
Topological mapping

- The processing of visual information is not fully connected



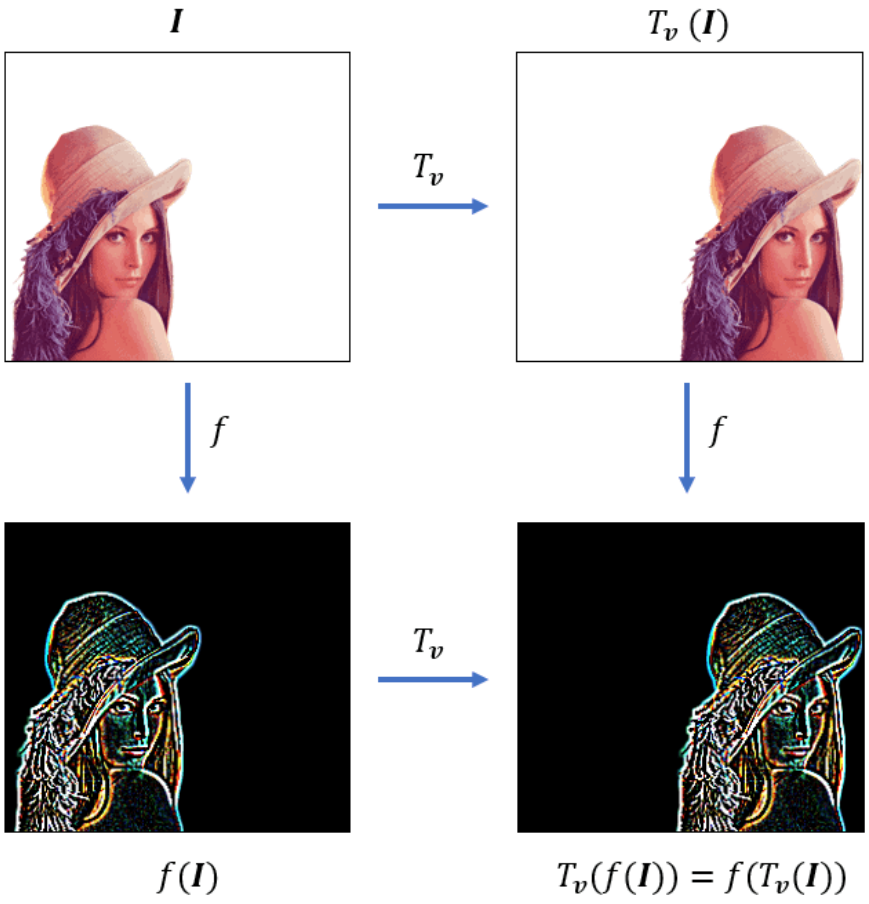
Topological mapping

- The processing of visual information is not fully connected



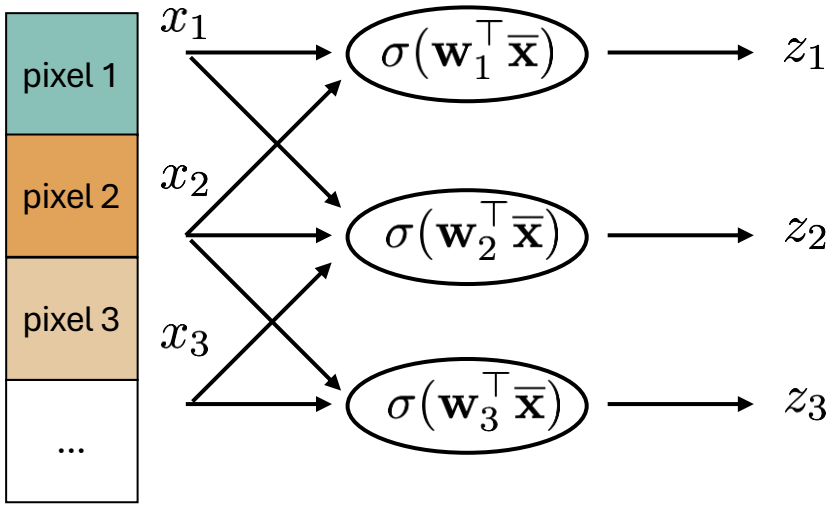
Translation equivariance

- The response should be same, regardless of position



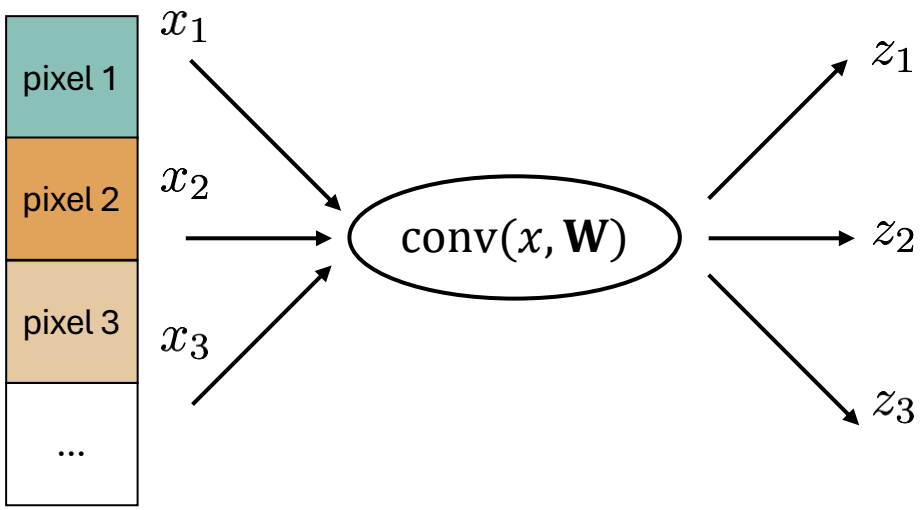
Translation equivariance

- The response should be same, regardless of position

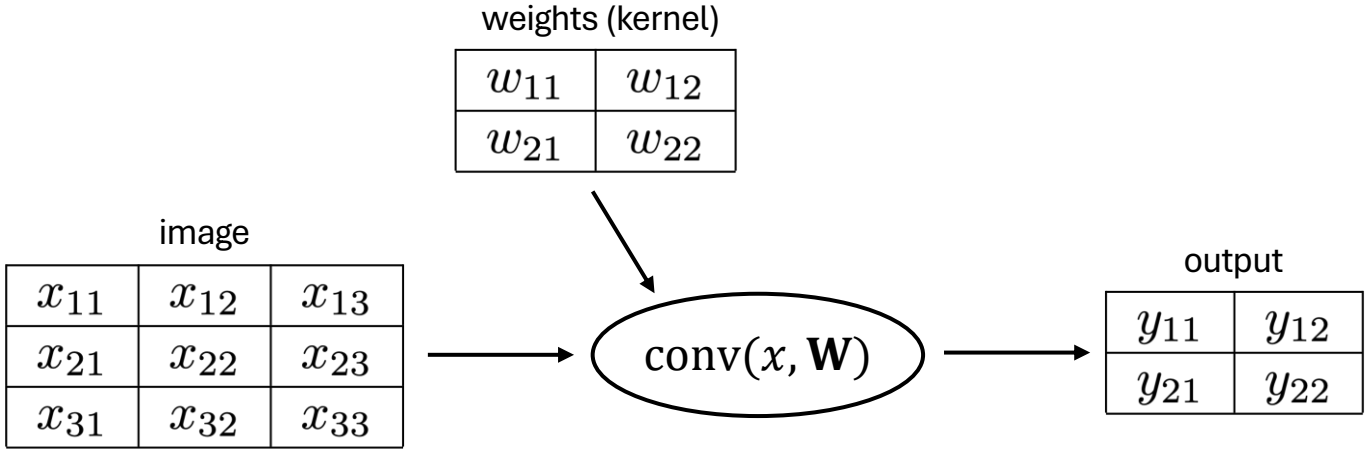


Translation equivariance

- The response should be same, regardless of position



Convolution



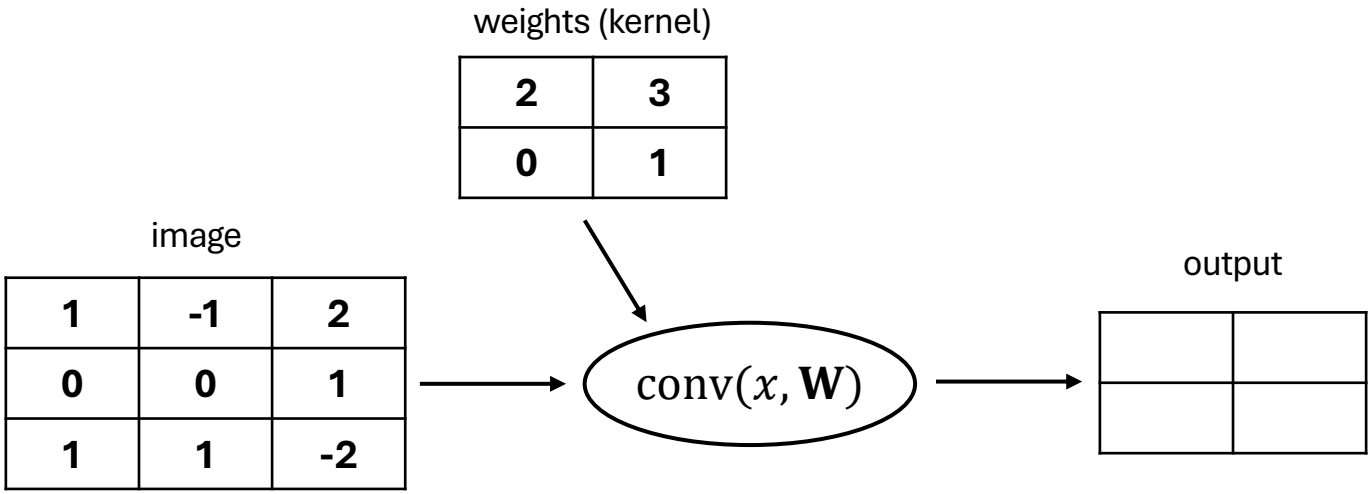
$$y_{11} = w_{11}x_{11} + w_{12}x_{12} + w_{21}x_{21} + w_{22}x_{22}$$

$$y_{12} = w_{11}x_{12} + w_{12}x_{13} + w_{21}x_{22} + w_{22}x_{23}$$

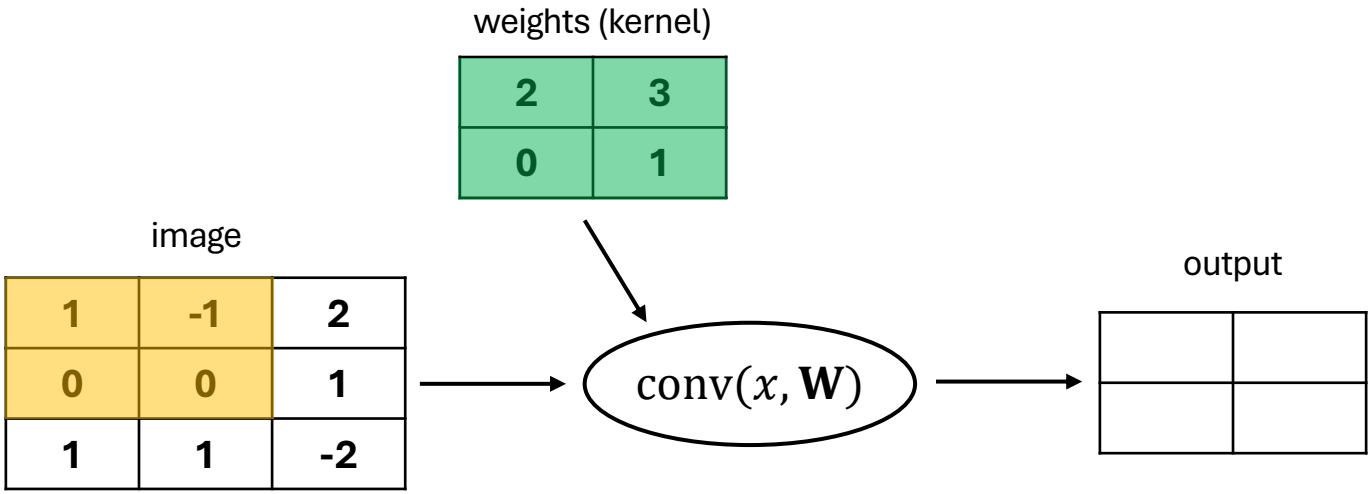
$$y_{21} = w_{11}x_{21} + w_{12}x_{22} + w_{21}x_{31} + w_{22}x_{32}$$

$$y_{22} = w_{11}x_{22} + w_{12}x_{23} + w_{21}x_{32} + w_{22}x_{33}$$

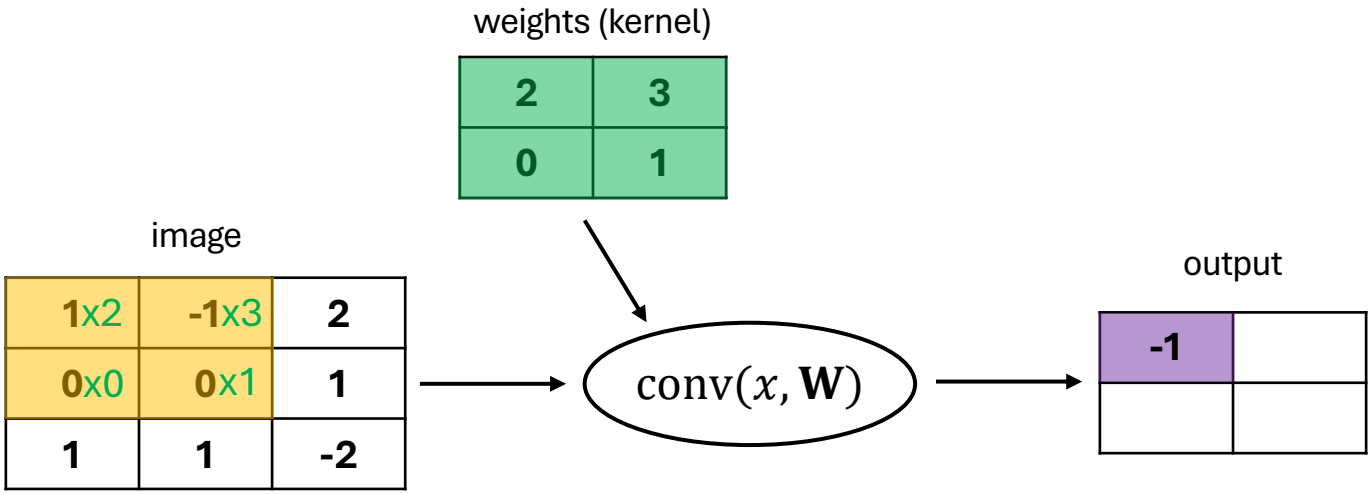
Convolution



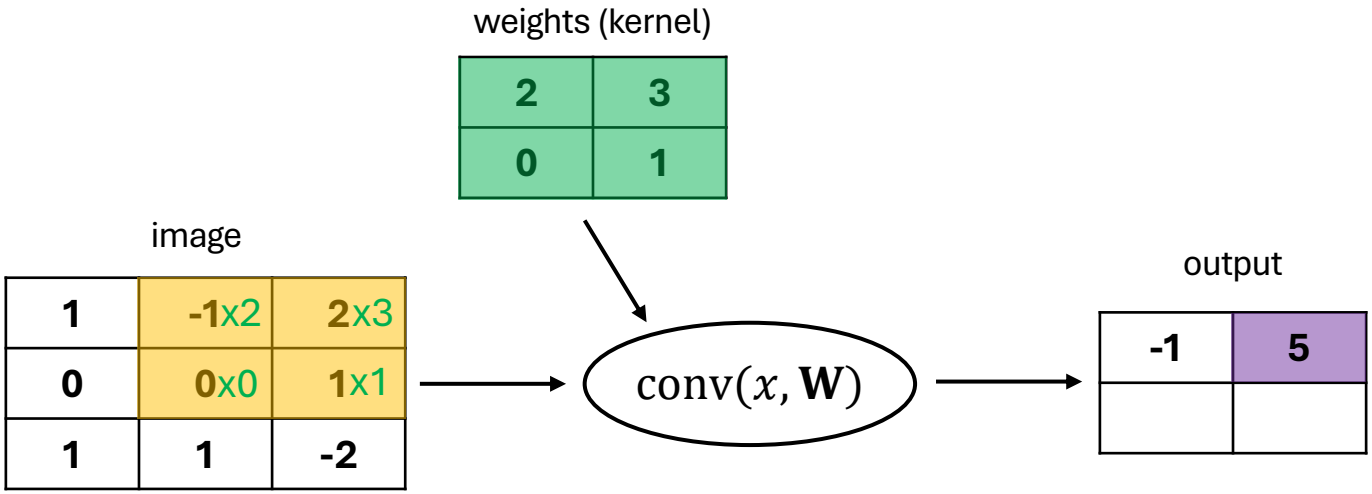
Convolution



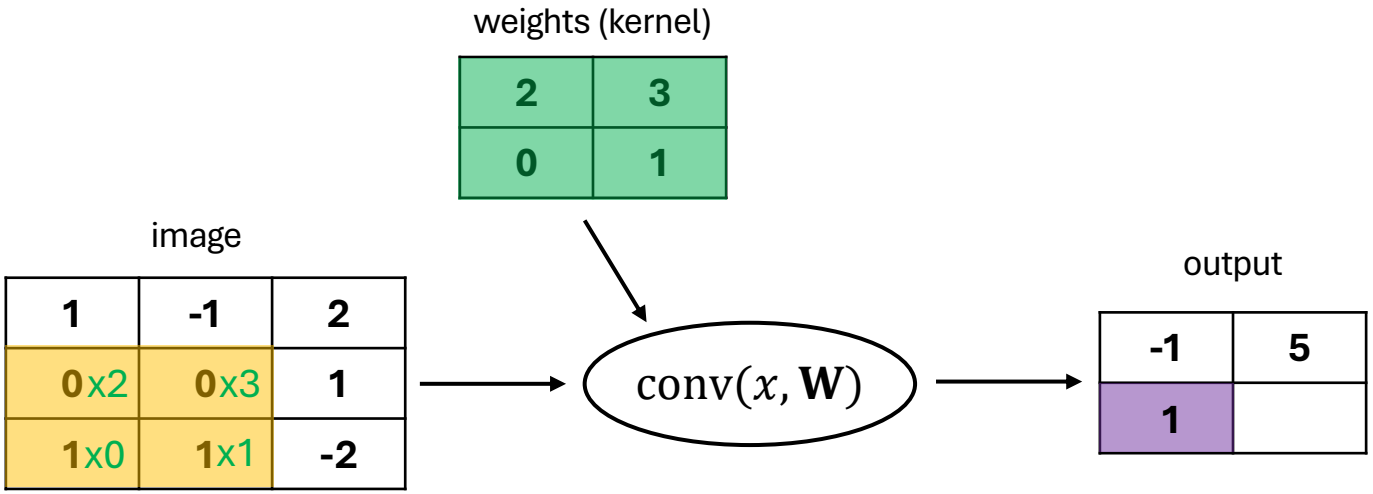
Convolution



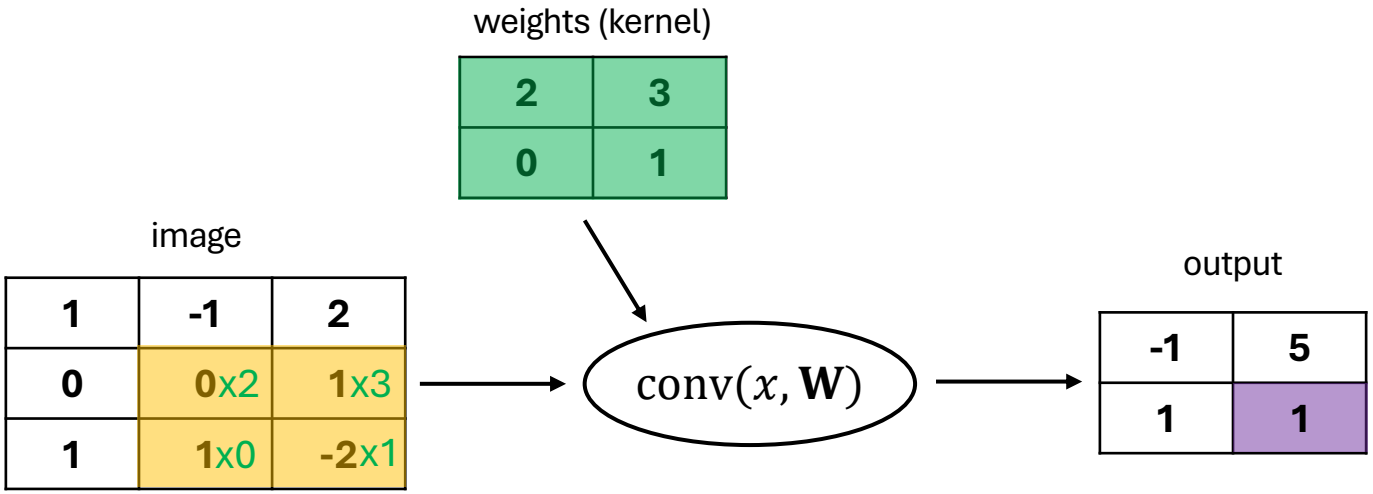
Convolution



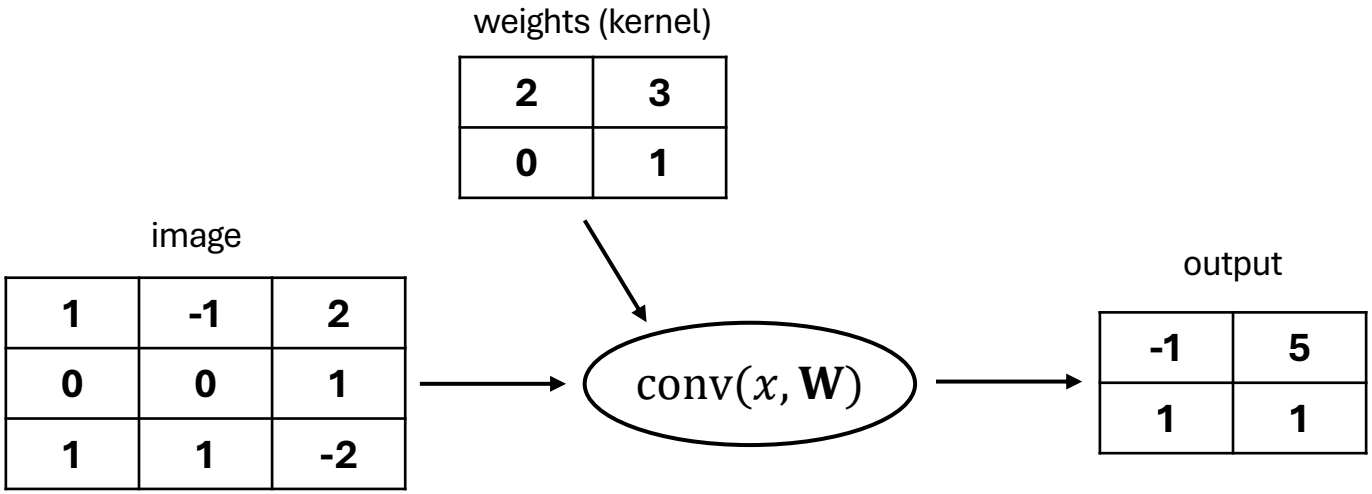
Convolution



Convolution



Convolution



Output Size

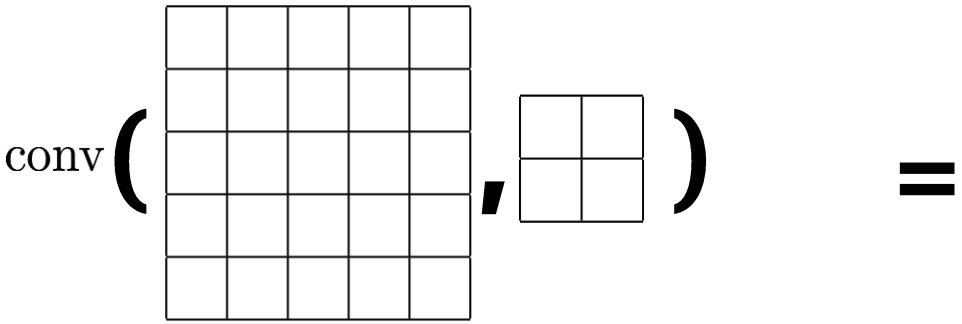


image
(5x5)

kernel
(2x2)

output
(? x ?)

Output Size

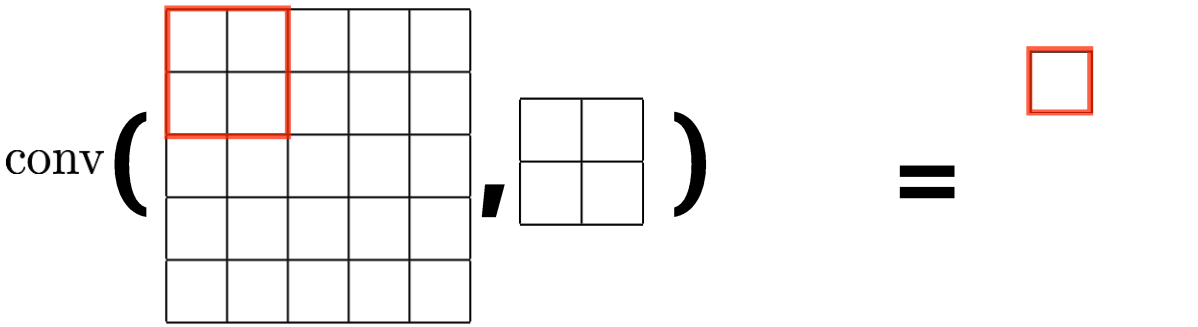


image
(5x5)

kernel
(2x2)

output
(? x ?)

Output Size

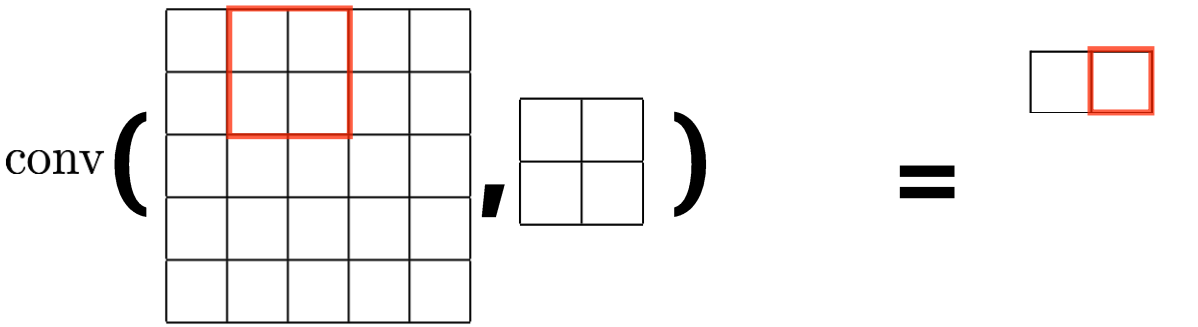


image
(5x5)

kernel
(2x2)

output
(? x ?)

Output Size

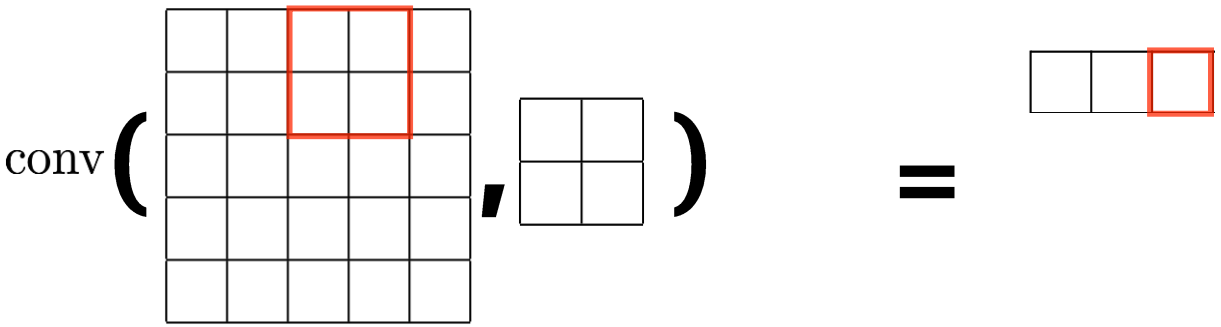


image
(5x5)

kernel
(2x2)

output
(? x ?)

Output Size

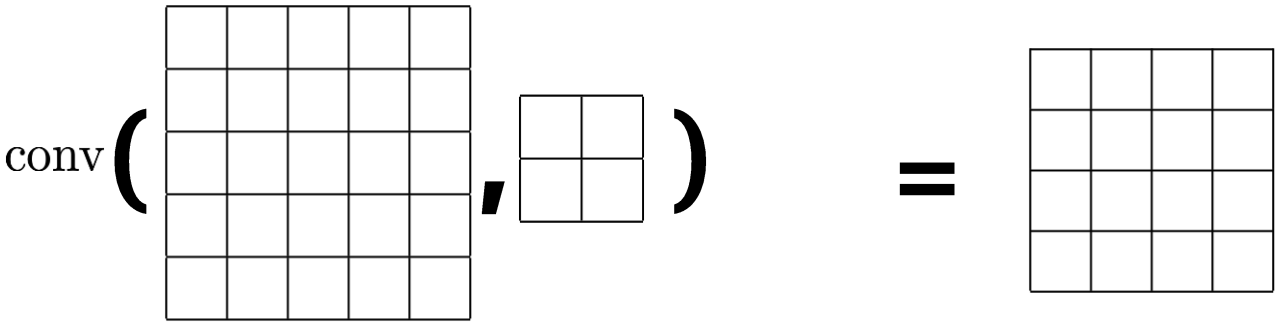


image
(5x5)

kernel
(2x2)

output
(**4 x 4**)

Output Size

$$N = M - K + 1$$

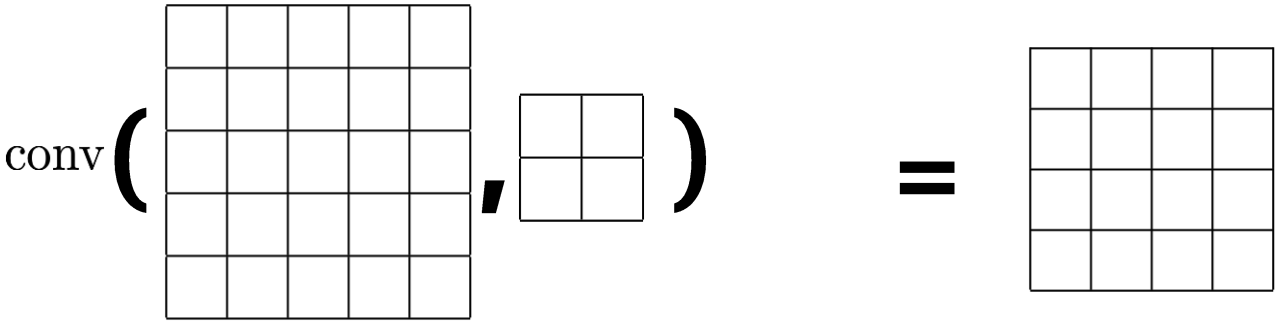


image (MxM) kernel (KxK) output (NxN)

Stride

Stride = 1

Window moves by 1 pixel in each iteration

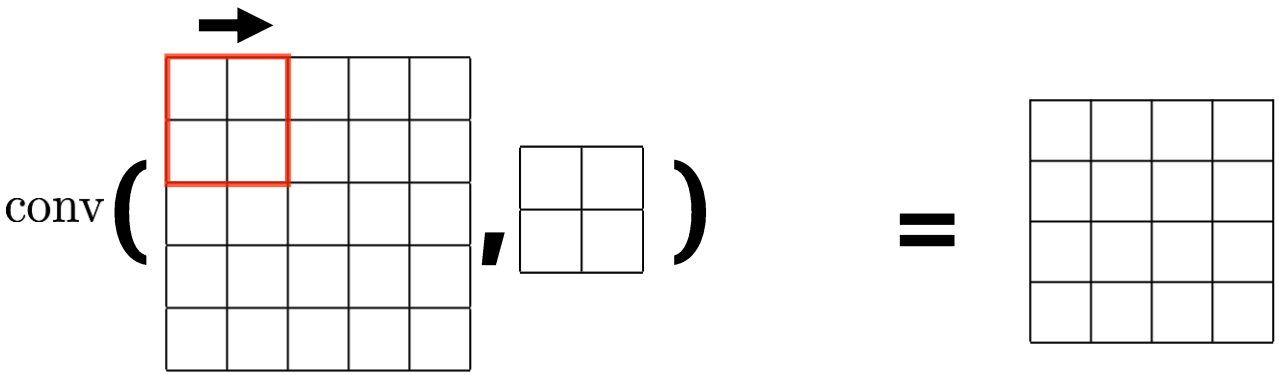


image
(5x5)

kernel
(2x2)

output
(4 x 4)

Stride

Stride = 3

Window moves by 3 pixels in each iteration

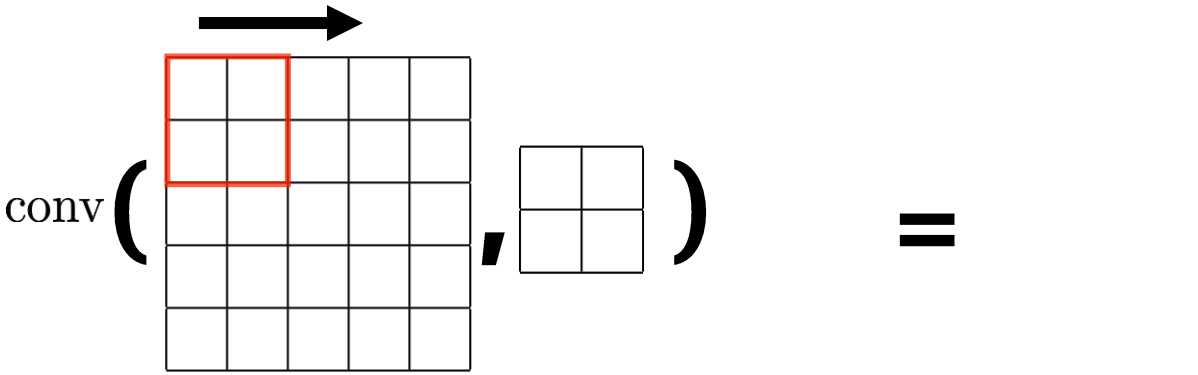


image (5x5) | kernel (2x2) | output (? x ?)

Stride

Stride = 3

Window moves by 3 pixels in each iteration

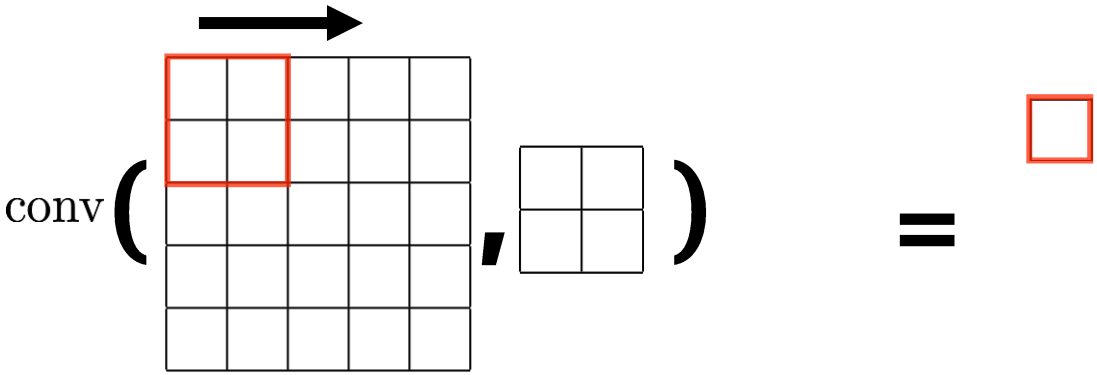


image (5x5)

kernel (2x2)

output (? x ?)

Stride

Stride = 3

Window moves by 3 pixels in each iteration

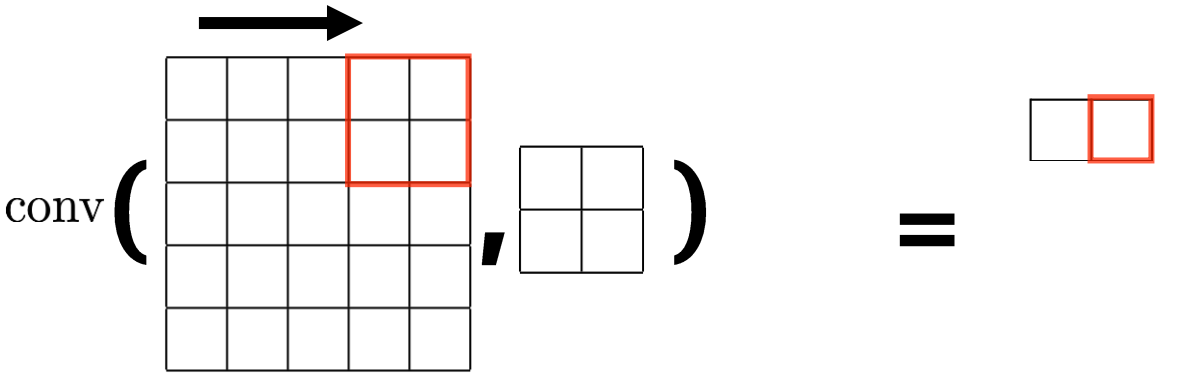


image (5x5)

kernel (2x2)

output (? x ?)

Stride

Stride = 3

Window moves by 3 pixels in each iteration

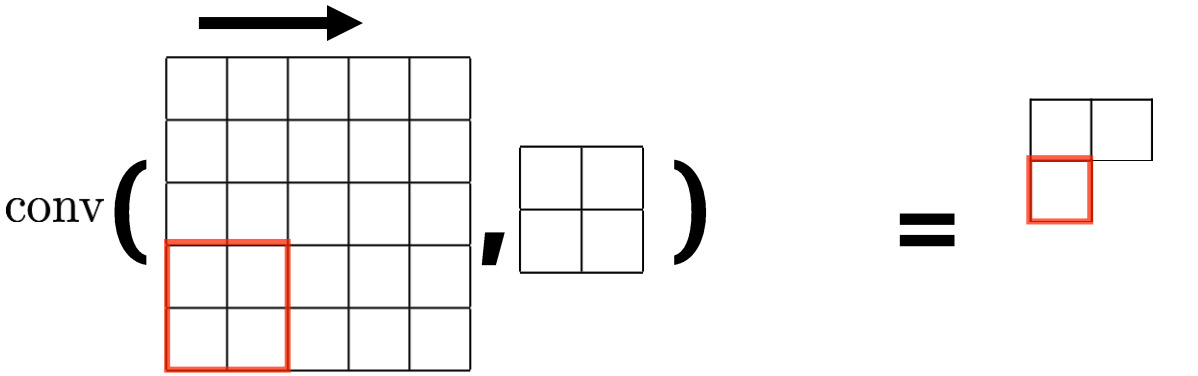


image (5x5)

kernel (2x2)

output (? x ?)

Stride

Stride = 3

Window moves by 3 pixels in each iteration

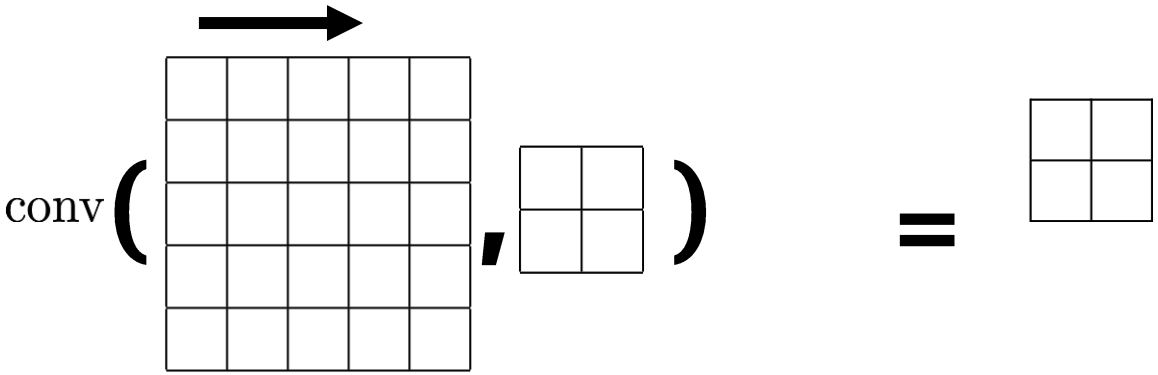


image
(5x5)

kernel
(2x2)

output
(**2 x 2**)

Stride

$$N = \texttt{floor}((M - K)/\text{stride} + 1)$$

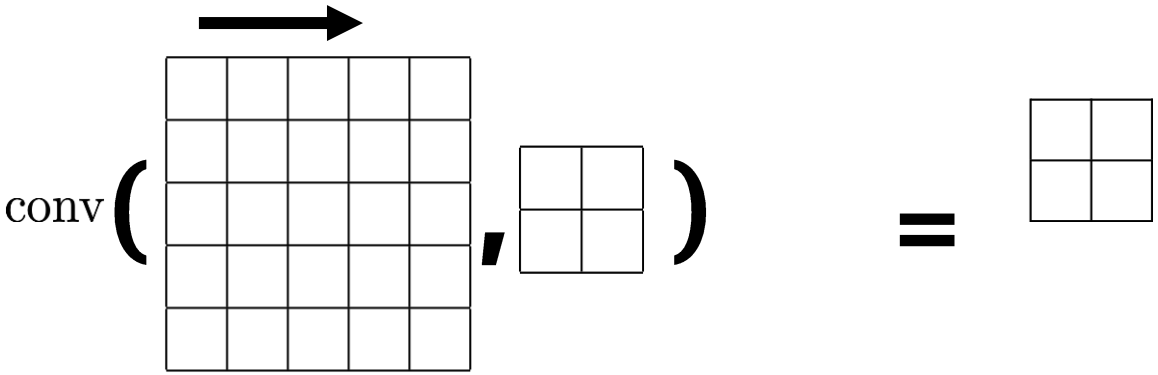


image (MxM) kernel (KxK) output (NxN)

Padding

Pad = 1

Virtually adds zero padding to the input image

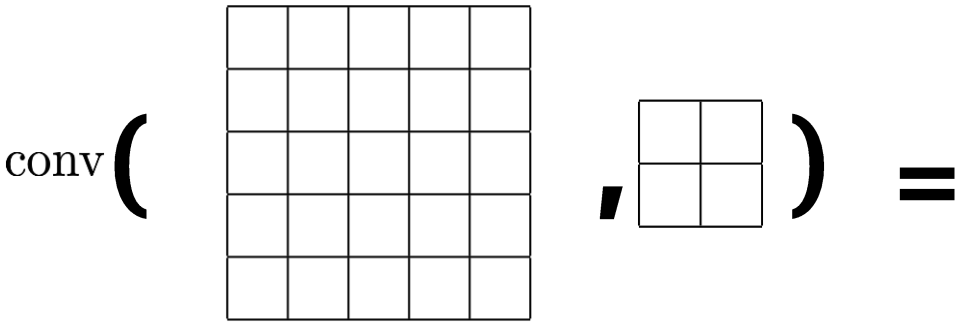


image
(5x5)

kernel
(2x2)

output
(? x ?)

Padding

Pad = 1

Virtually adds zero padding to the input image

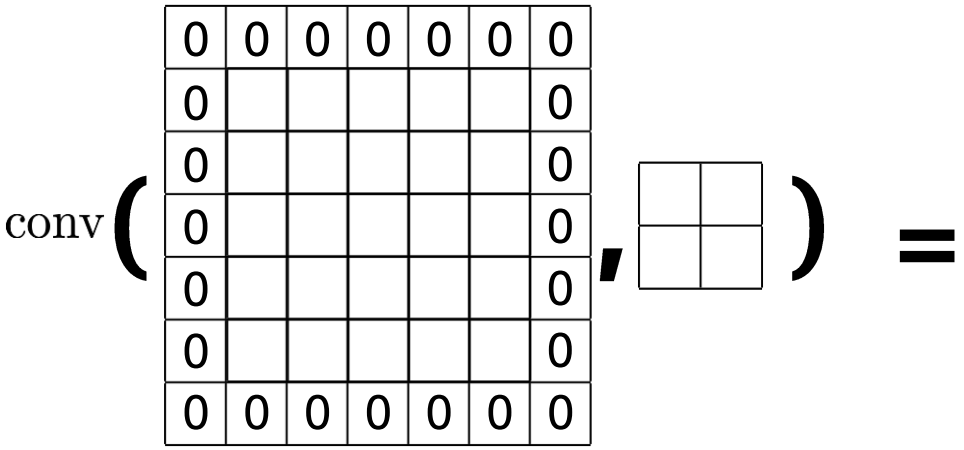


image (5x5) | kernel (2x2) | output (? x ?)

Padding

Pad = 1

Virtually adds zero padding to the input image

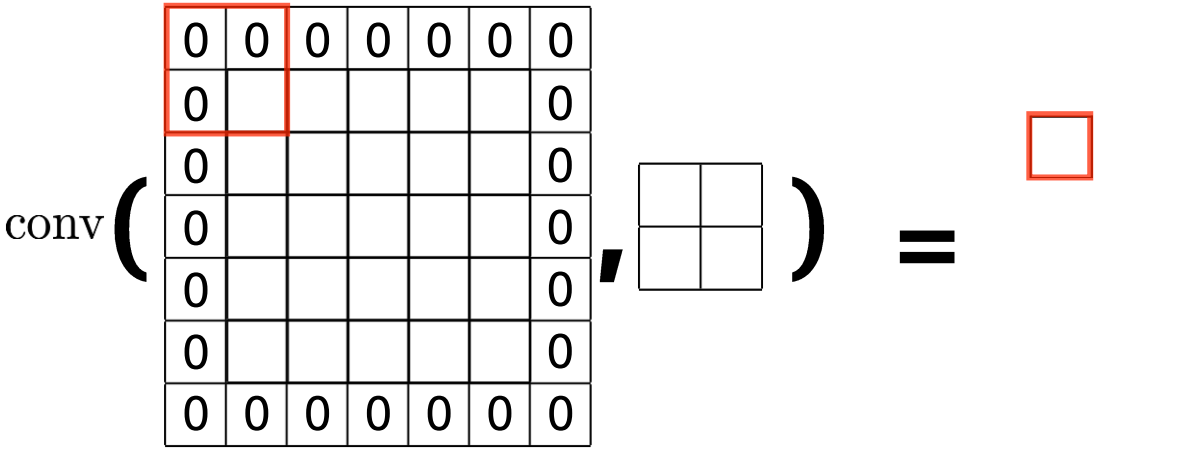


image (5x5) | kernel (2x2) | output (? x ?)

Padding

Pad = 1

Virtually adds zero padding to the input image

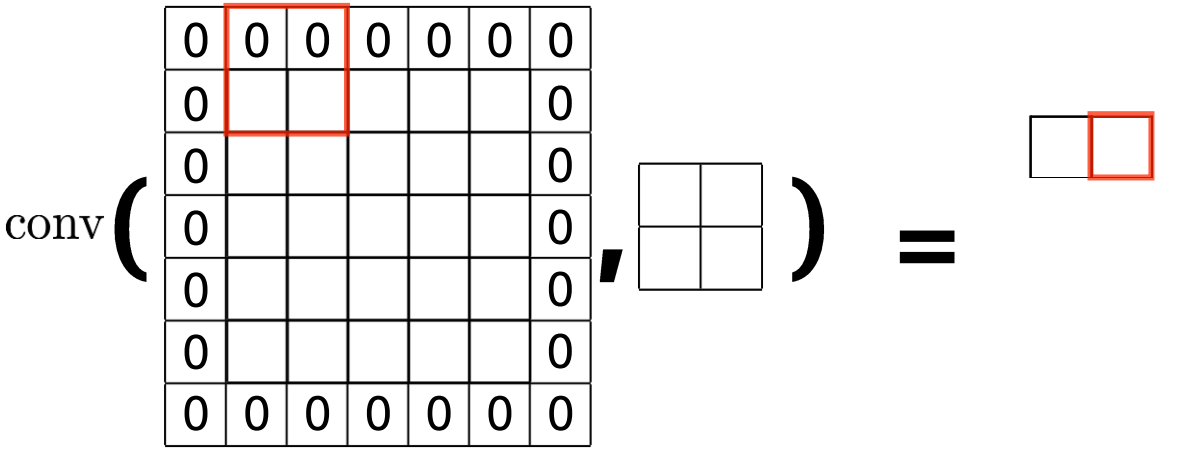


image (5x5) | kernel (2x2) | output (? x ?)

Padding

Pad = 1

Virtually adds zero padding to the input image

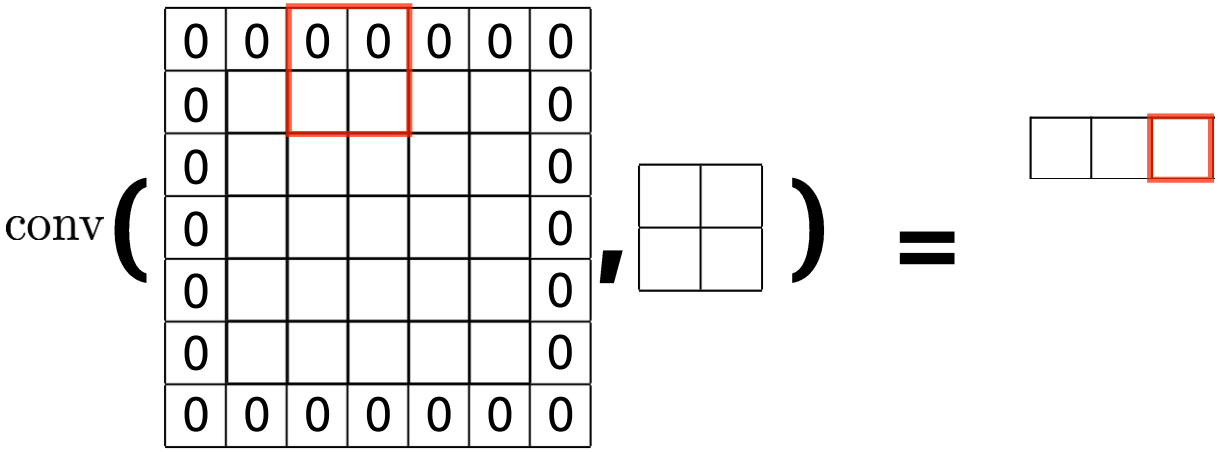


image (5x5) | kernel (2x2) | output (? x ?)

Padding

Pad = 1

Virtually adds zero padding to the input image

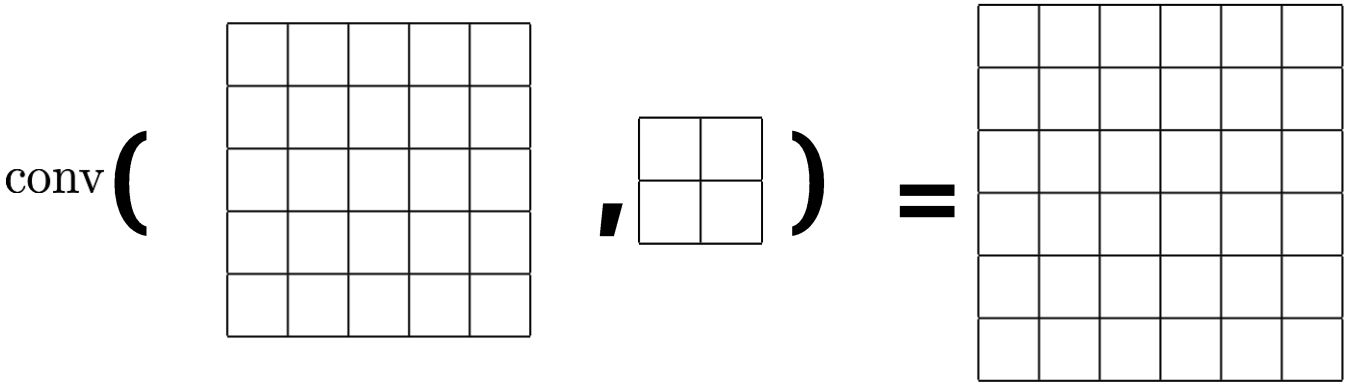


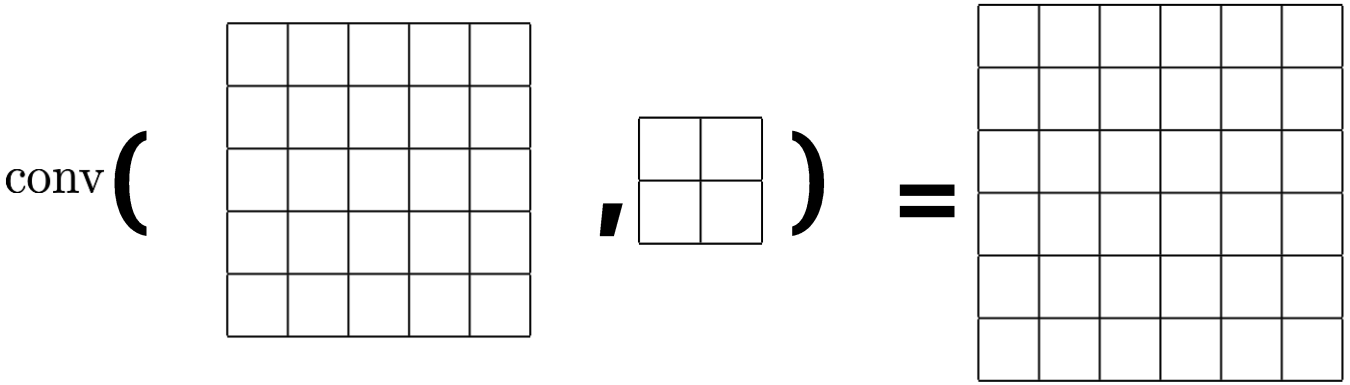
image
(5x5)

kernel
(2x2)

output
(**6 x 6**)

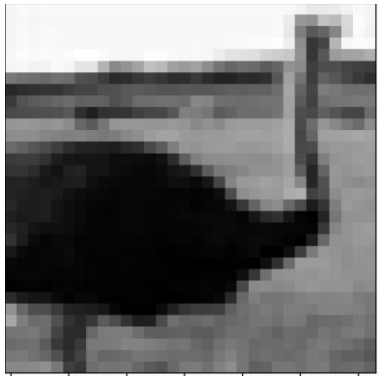
Padding

$$N = \texttt{floor}((M + 2*\text{padding} - K)/\text{stride} + 1)$$



Convolution Examples

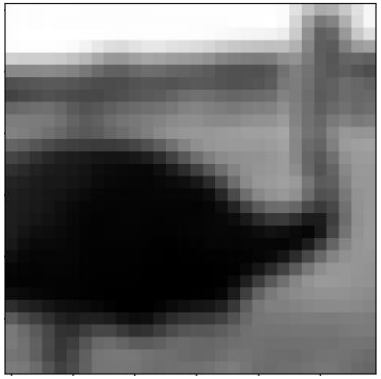
Input image



Input kernel

1	1	1
1	1	1
1	1	1

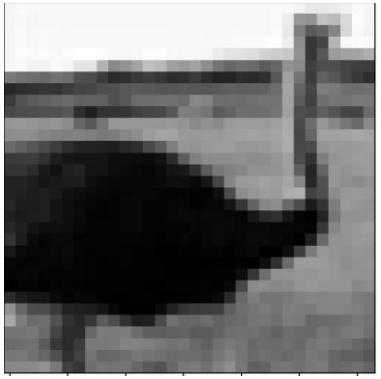
Output



```
nn.Conv2d(1,1,3)
```

Convolution Examples

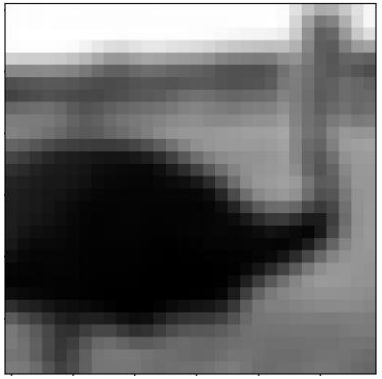
Input image



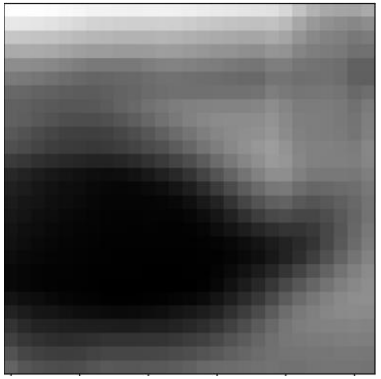
Input kernel

1	1	1	1	1
1	1	1	1	1
1	1	1	1	1
1	1	1	1	1
1	1	1	1	1

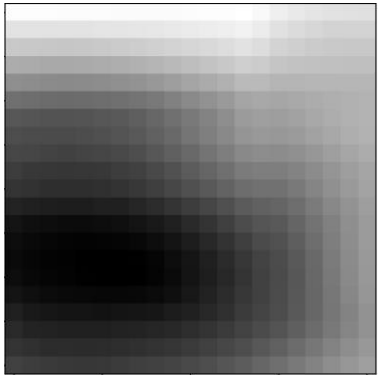
Output



```
nn.Conv2d(1,1,3)
```



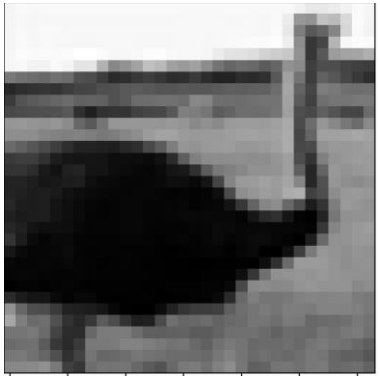
```
nn.Conv2d(1,1,5)
```



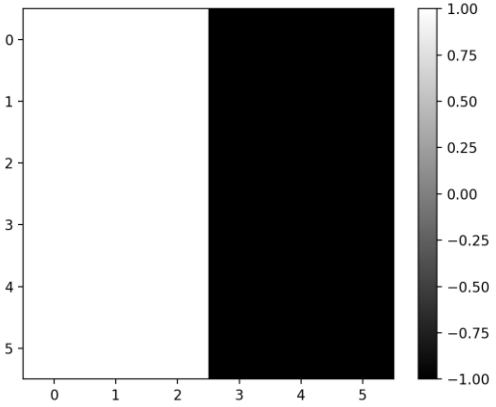
```
nn.Conv2d(1,1,13)
```

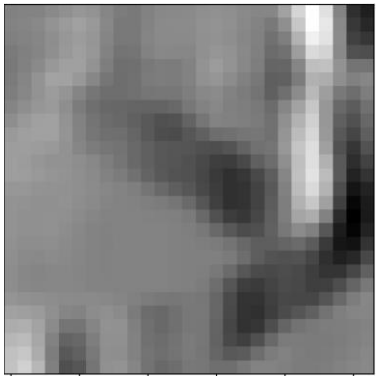
Convolution Examples

Input image

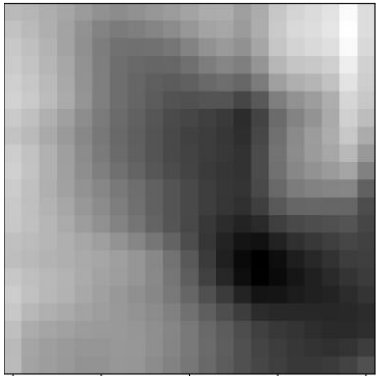


Input kernel

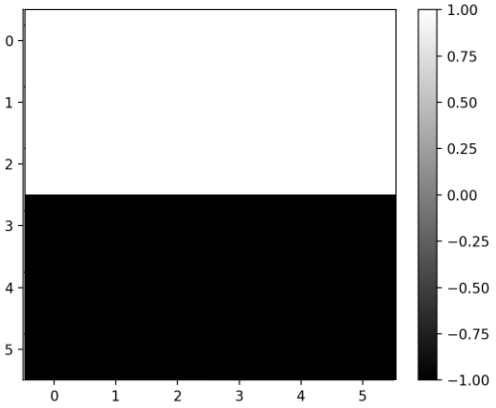


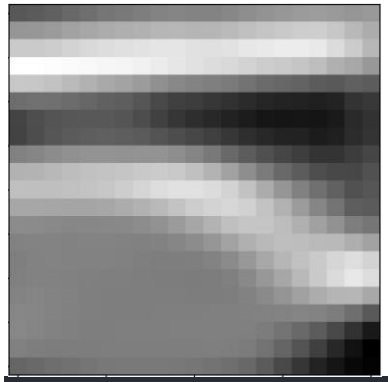


nn.Conv2d(1,1,6)

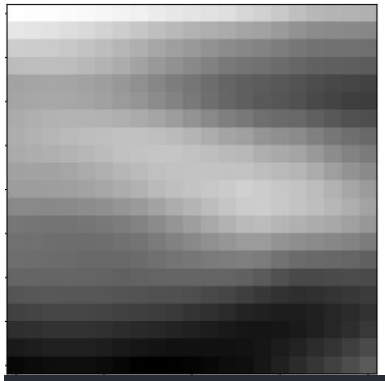


nn.Conv2d(1,1,12)





nn.Conv2d(1,1,6)



nn.Conv2d(1,1,12)

Multi-channel convolution

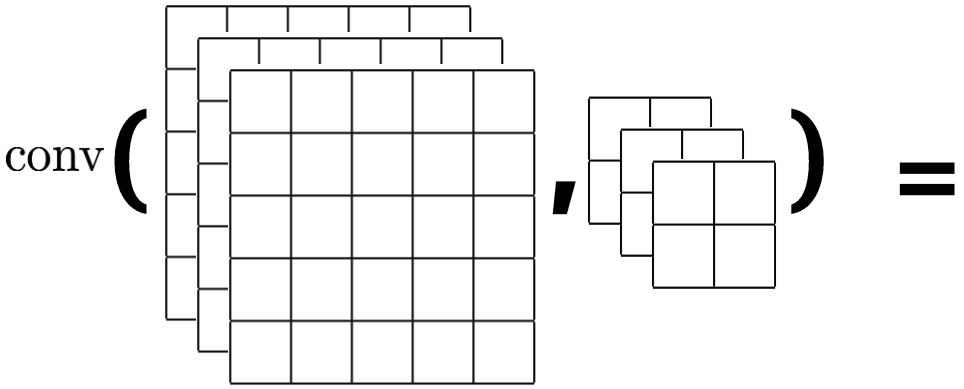
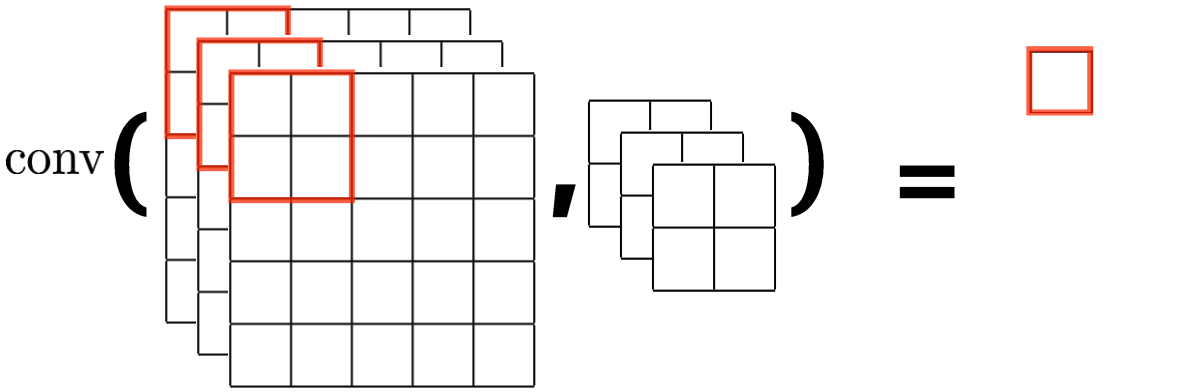


image
(3x5x5)

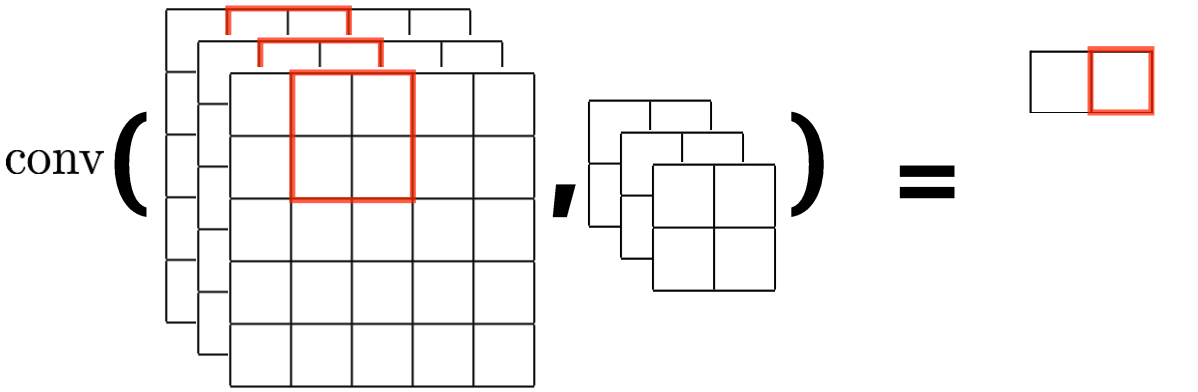
kernel
(3x2x2)

output
(? x ?)

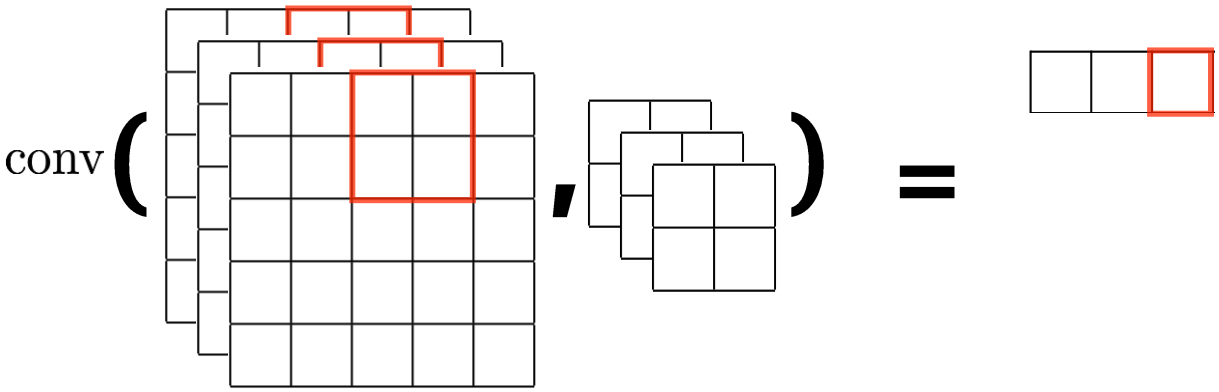
Multi-channel convolution



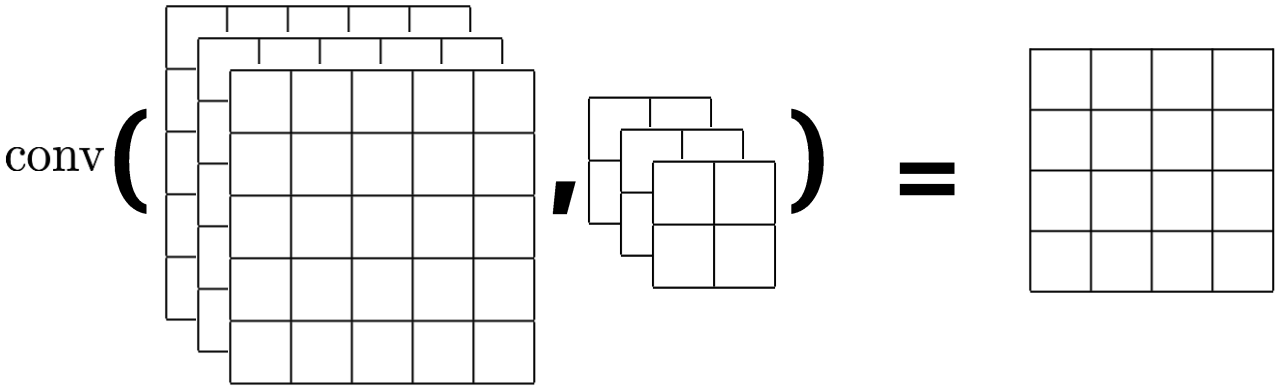
Multi-channel convolution



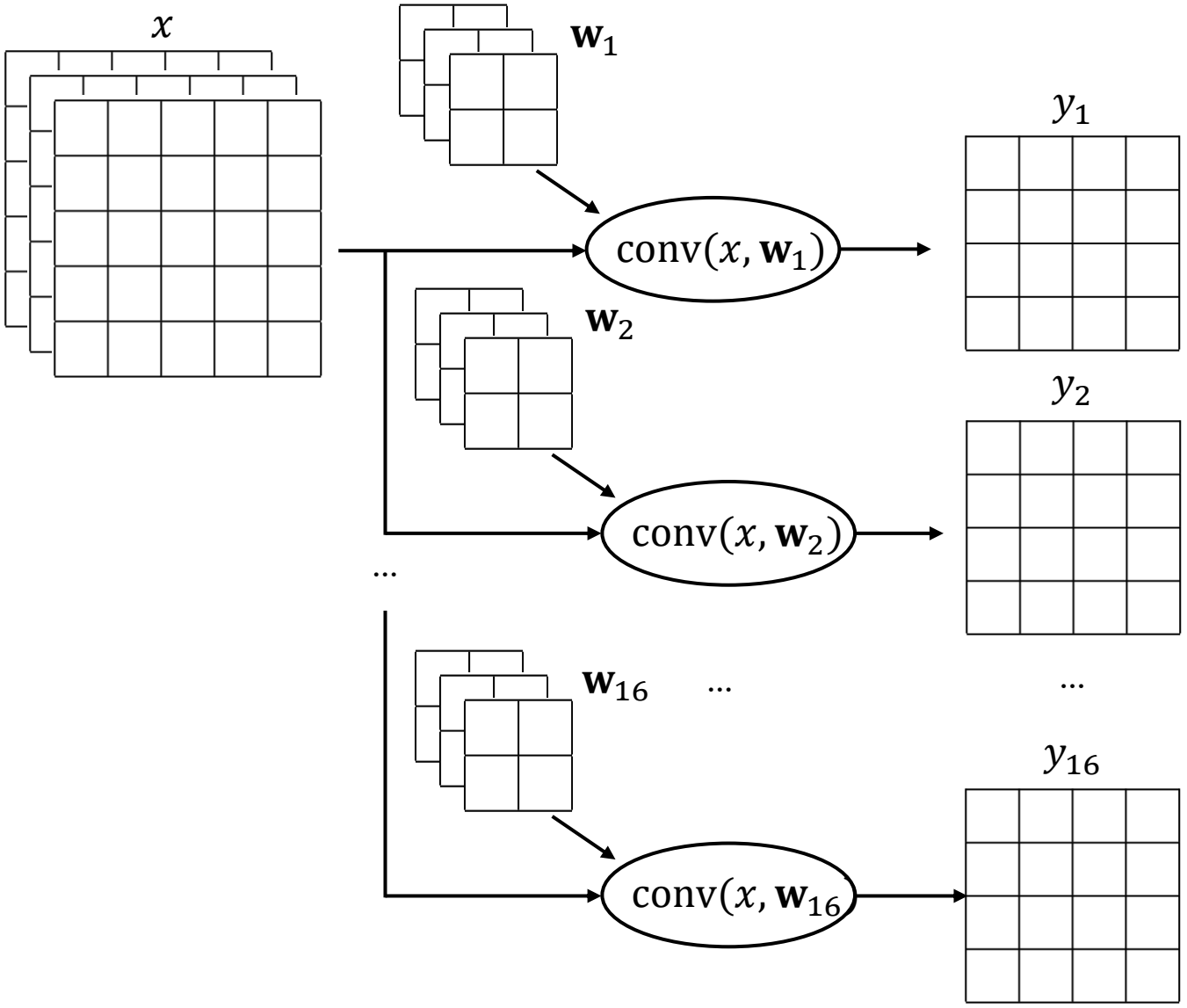
Multi-channel convolution



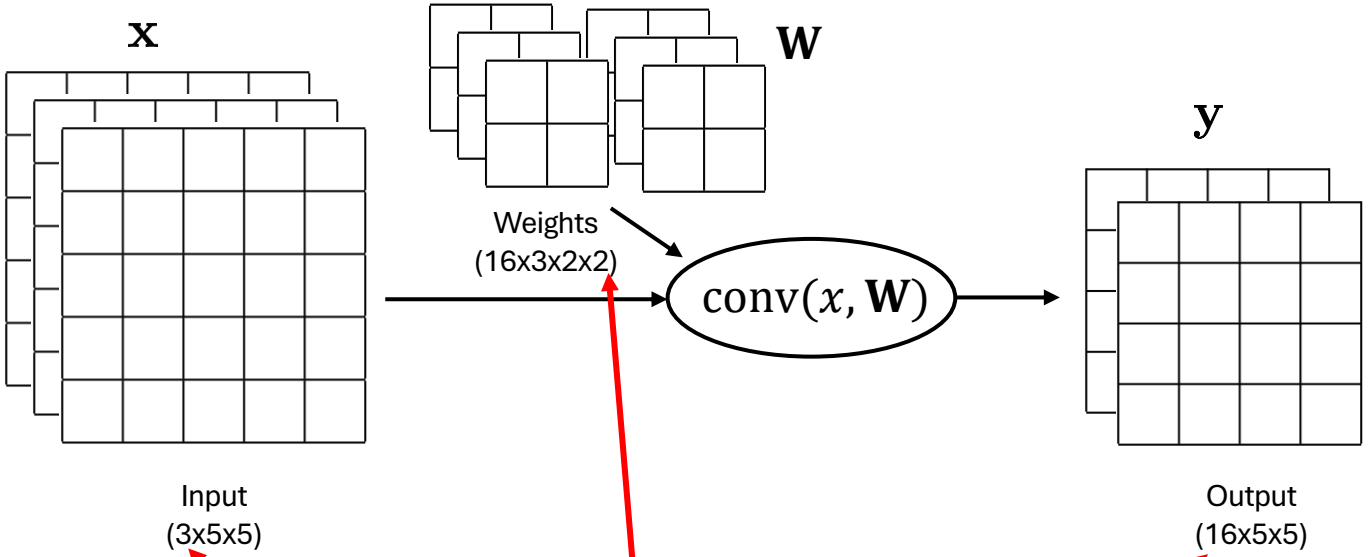
Multi-channel convolution



Convolutional Layer



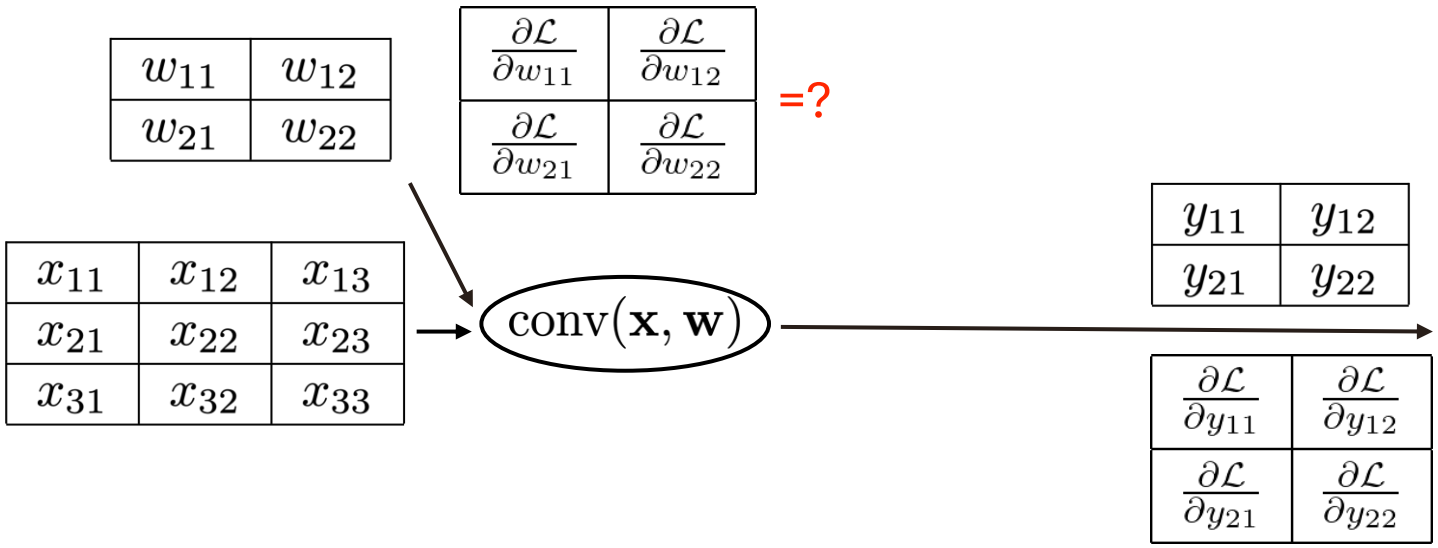
Convolutional Layer




```
# define 2D convolutional layer
first_layer = nn.Conv2d(in_channels=3,
                        out_channels=16,
                        kernel_size=2,
                        stride=1,
                        padding=1)
```

Backward Pass

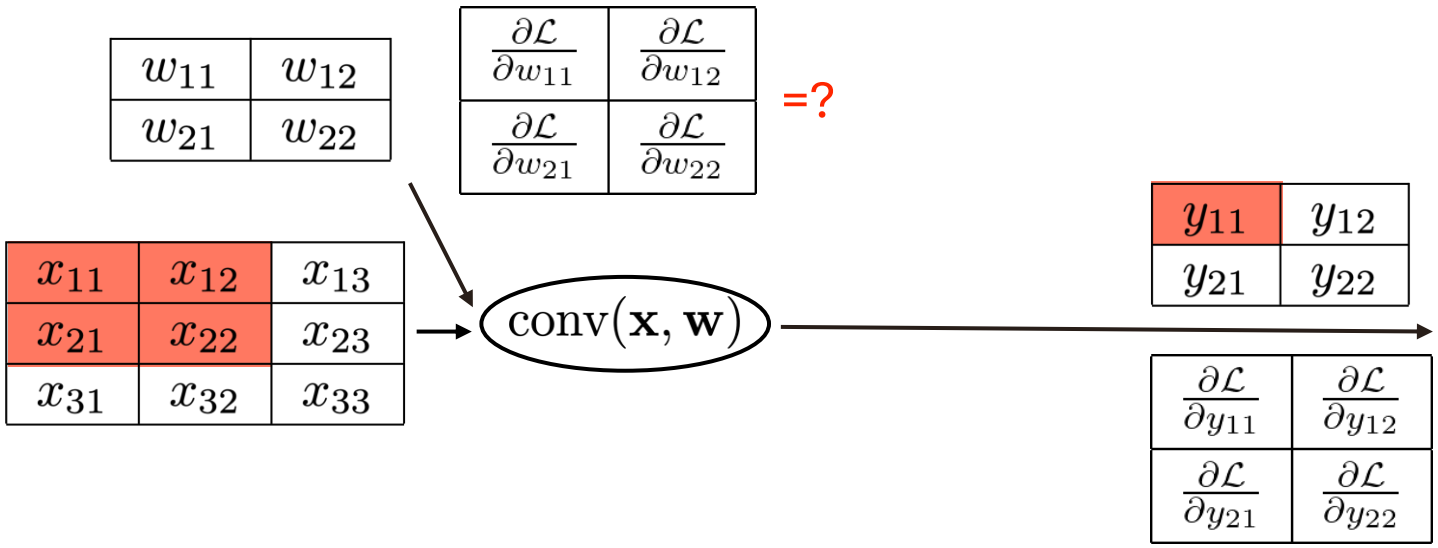
$$\frac{\partial \mathcal{L}}{\partial w_{11}} = \frac{\partial \mathcal{L}}{\partial y_{11}}\frac{\partial y_{11}}{\partial w_{11}} + \frac{\partial \mathcal{L}}{\partial y_{12}}\frac{\partial y_{12}}{\partial w_{11}} + \frac{\partial \mathcal{L}}{\partial y_{21}}\frac{\partial y_{21}}{\partial w_{11}} + \frac{\partial \mathcal{L}}{\partial y_{22}}\frac{\partial y_{22}}{\partial w_{11}}$$



Backward Pass

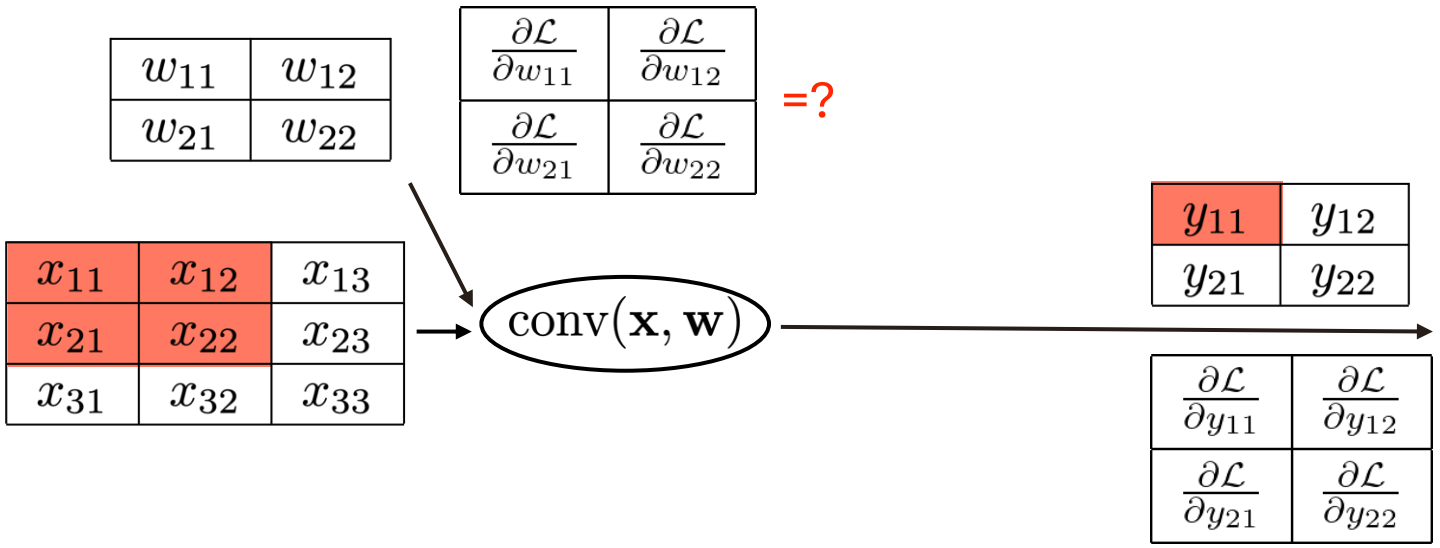
$$\frac{\partial \mathcal{L}}{\partial w_{11}} = \frac{\partial \mathcal{L}}{\partial y_{11}}\frac{\partial y_{11}}{\partial w_{11}} + \frac{\partial \mathcal{L}}{\partial y_{12}}\frac{\partial y_{12}}{\partial w_{11}} + \frac{\partial \mathcal{L}}{\partial y_{21}}\frac{\partial y_{21}}{\partial w_{11}} + \frac{\partial \mathcal{L}}{\partial y_{22}}\frac{\partial y_{22}}{\partial w_{11}}$$

$$\frac{\partial y_{11}}{\partial w_{11}} = \frac{\partial(w_{11}x_{11} + w_{12}x_{12} + w_{21}x_{21} + w_{22}x_{22})}{\partial w_{11}} = x_{11}$$



Backward Pass

$$\frac{\partial \mathcal{L}}{\partial w_{11}} = \frac{\partial \mathcal{L}}{\partial y_{11}} x_{11} + \frac{\partial \mathcal{L}}{\partial y_{12}} x_{12} + \frac{\partial \mathcal{L}}{\partial y_{21}} x_{21} + \frac{\partial \mathcal{L}}{\partial y_{22}} x_{22}$$



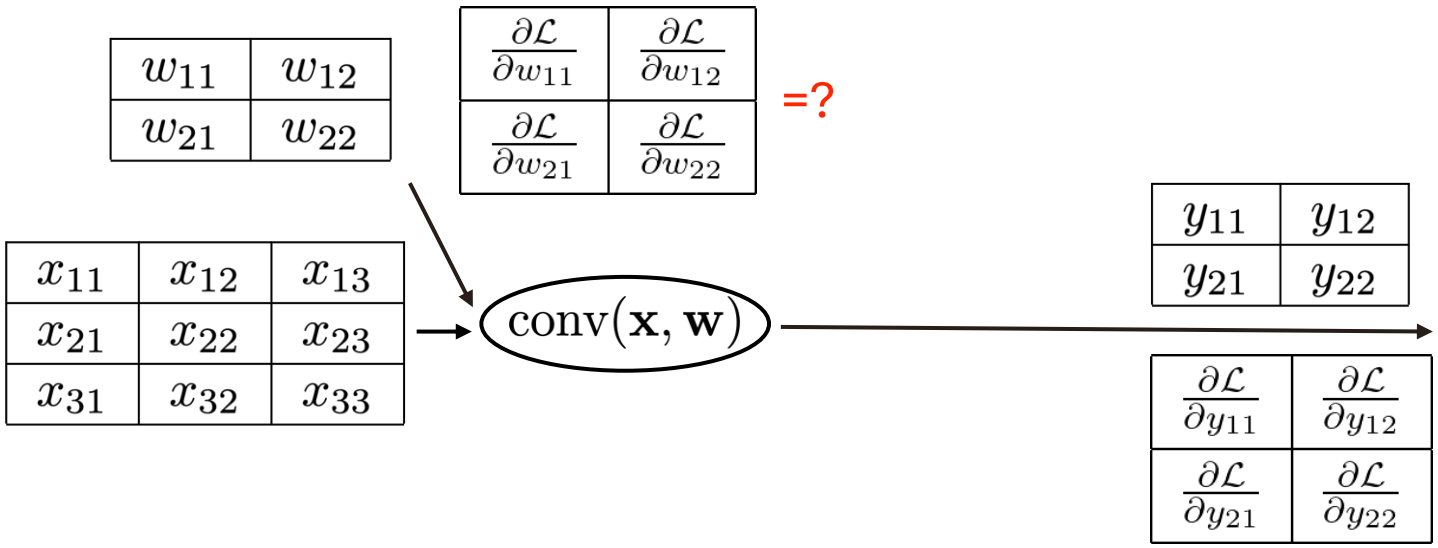
Backward Pass

$$\frac{\partial \mathcal{L}}{\partial w_{11}} = \frac{\partial \mathcal{L}}{\partial y_{11}} x_{11} + \frac{\partial \mathcal{L}}{\partial y_{12}} x_{12} + \frac{\partial \mathcal{L}}{\partial y_{21}} x_{21} + \frac{\partial \mathcal{L}}{\partial y_{22}} x_{22}$$

$$\frac{\partial \mathcal{L}}{\partial w_{12}} = \frac{\partial \mathcal{L}}{\partial y_{11}} x_{12} + \frac{\partial \mathcal{L}}{\partial y_{12}} x_{13} + \frac{\partial \mathcal{L}}{\partial y_{21}} x_{22} + \frac{\partial \mathcal{L}}{\partial y_{22}} x_{23}$$

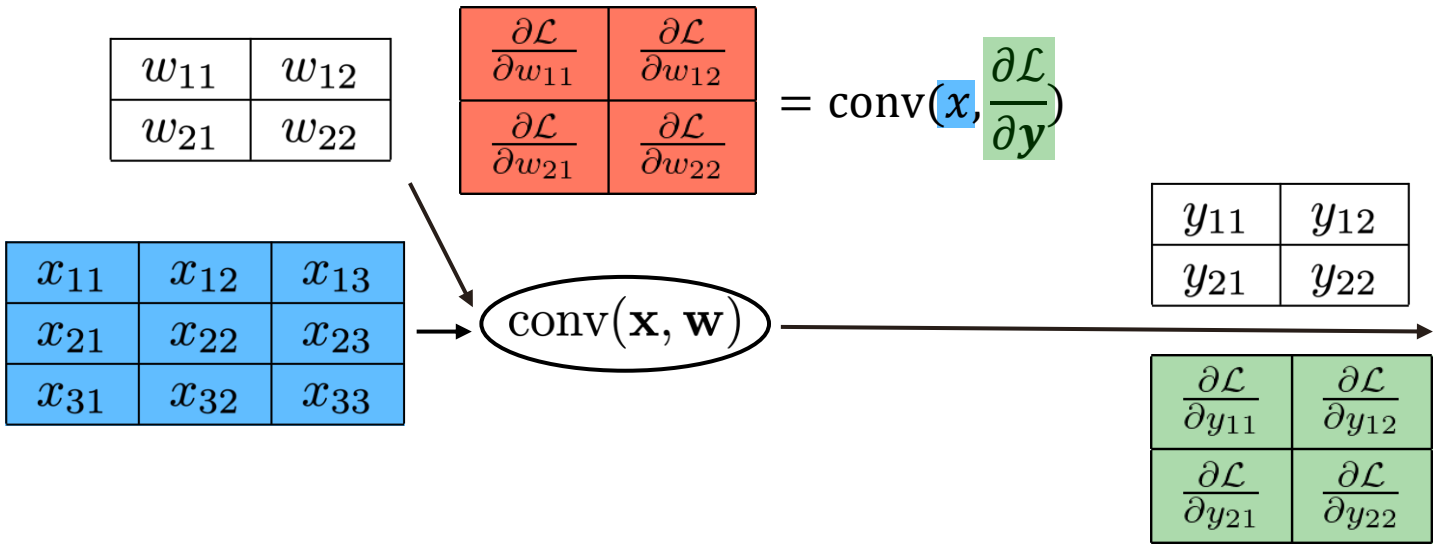
$$\frac{\partial \mathcal{L}}{\partial w_{21}} = \frac{\partial \mathcal{L}}{\partial y_{11}} x_{21} + \frac{\partial \mathcal{L}}{\partial y_{12}} x_{22} + \frac{\partial \mathcal{L}}{\partial y_{21}} x_{31} + \frac{\partial \mathcal{L}}{\partial y_{22}} x_{32}$$

$$\frac{\partial \mathcal{L}}{\partial w_{22}} = \frac{\partial \mathcal{L}}{\partial y_{11}} x_{22} + \frac{\partial \mathcal{L}}{\partial y_{12}} x_{23} + \frac{\partial \mathcal{L}}{\partial y_{21}} x_{32} + \frac{\partial \mathcal{L}}{\partial y_{22}} x_{33}$$



Backward Pass

$$\frac{\partial \mathcal{L}}{\partial w_{11}} = \frac{\partial \mathcal{L}}{\partial y_{11}} x_{11} + \frac{\partial \mathcal{L}}{\partial y_{12}} x_{12} + \frac{\partial \mathcal{L}}{\partial y_{21}} x_{21} + \frac{\partial \mathcal{L}}{\partial y_{22}} x_{22}$$
$$\frac{\partial \mathcal{L}}{\partial w_{12}} = \frac{\partial \mathcal{L}}{\partial y_{11}} x_{12} + \frac{\partial \mathcal{L}}{\partial y_{12}} x_{13} + \frac{\partial \mathcal{L}}{\partial y_{21}} x_{22} + \frac{\partial \mathcal{L}}{\partial y_{22}} x_{23}$$
$$\frac{\partial \mathcal{L}}{\partial w_{21}} = \frac{\partial \mathcal{L}}{\partial y_{11}} x_{21} + \frac{\partial \mathcal{L}}{\partial y_{12}} x_{22} + \frac{\partial \mathcal{L}}{\partial y_{21}} x_{31} + \frac{\partial \mathcal{L}}{\partial y_{22}} x_{32}$$
$$\frac{\partial \mathcal{L}}{\partial w_{22}} = \frac{\partial \mathcal{L}}{\partial y_{11}} x_{22} + \frac{\partial \mathcal{L}}{\partial y_{12}} x_{23} + \frac{\partial \mathcal{L}}{\partial y_{21}} x_{32} + \frac{\partial \mathcal{L}}{\partial y_{22}} x_{33}$$



Backward Pass

- Loss gradient wrt the weights is defined as
  "convolution of input feature map with upstream gradient"

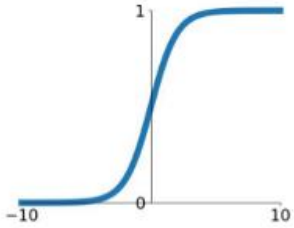
- Loss gradient wrt the input is defined as
  "convolution of padded upstream gradient with mirrored weights"

- **Summary:** Backward pass through a convolutional layer is also a convolution!

Activation Function Layers

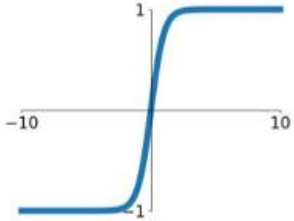
Sigmoid

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



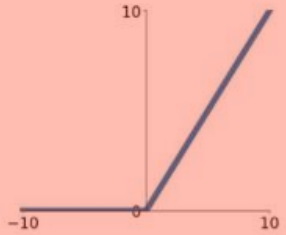
tanh

$$\tanh(x)$$



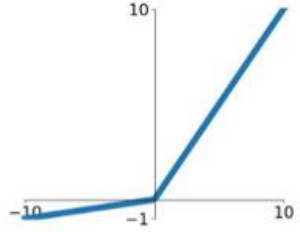
ReLU

$$\max(0, x)$$



Leaky ReLU

$$\max(0.1x, x)$$

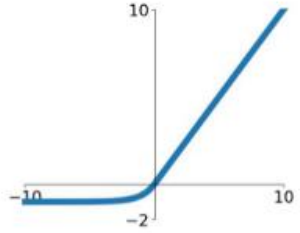


Maxout

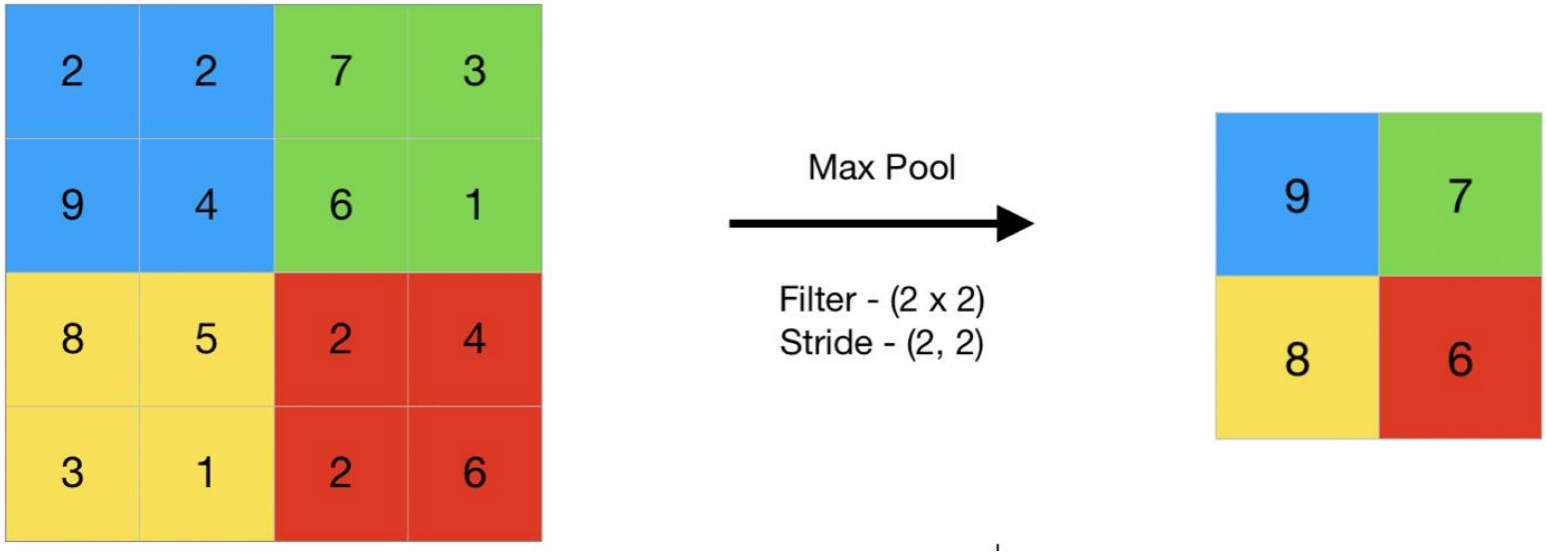
$$\max(w_1^T x + b_1, w_2^T x + b_2)$$

ELU

$$\begin{cases} x & x \geq 0 \\ \alpha(e^x - 1) & x < 0 \end{cases}$$



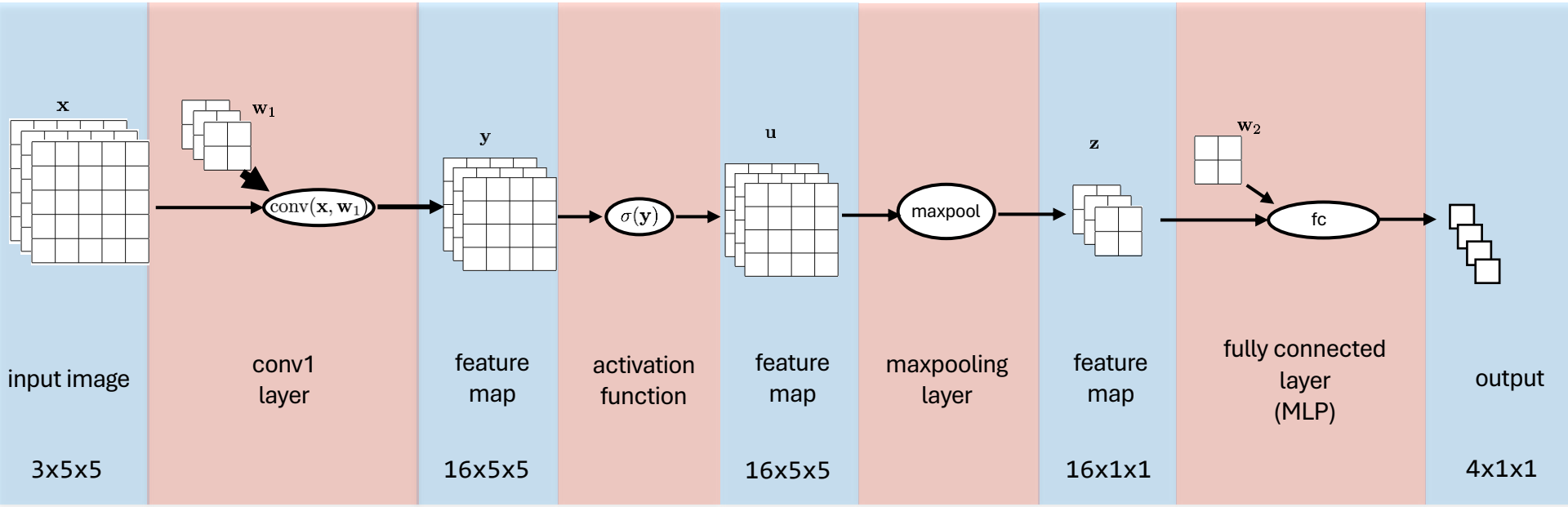
Max pooling layer



Simple CNN

Input: 3D tensor [channels x height x width]

Output: 4-way classification [no. of classes]

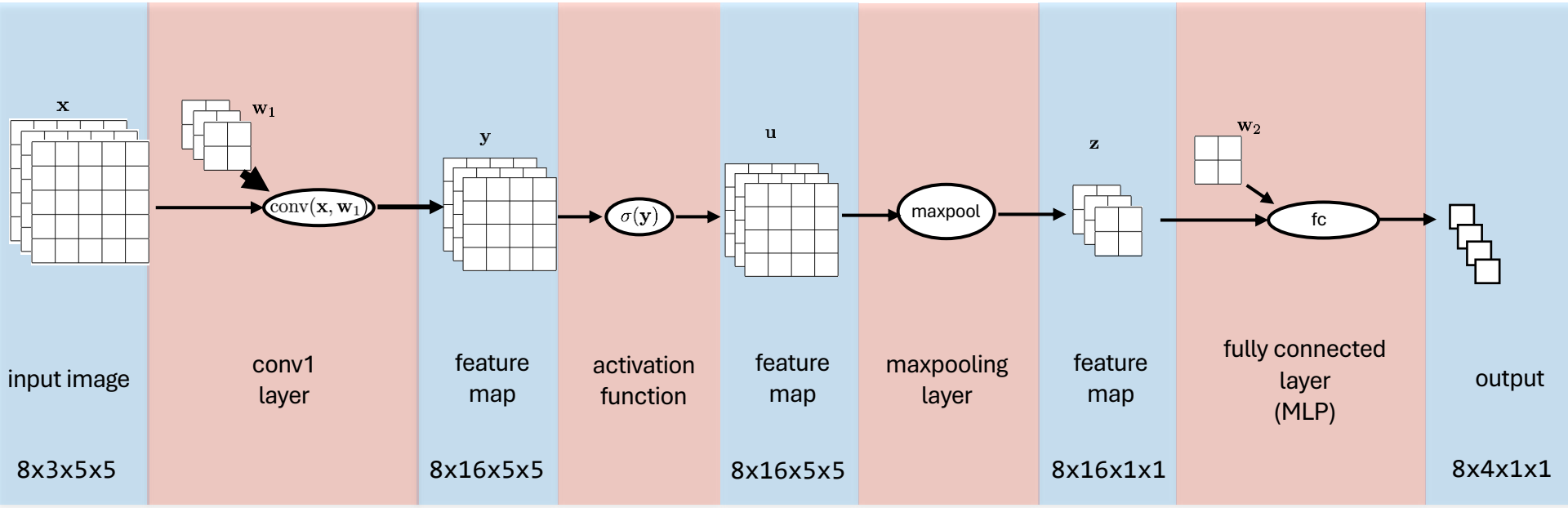


Simple CNN with mini-batches

Input: 4D tensor [batch x channels x height x width]

Output: 4-way classification [batch x no. of classes]

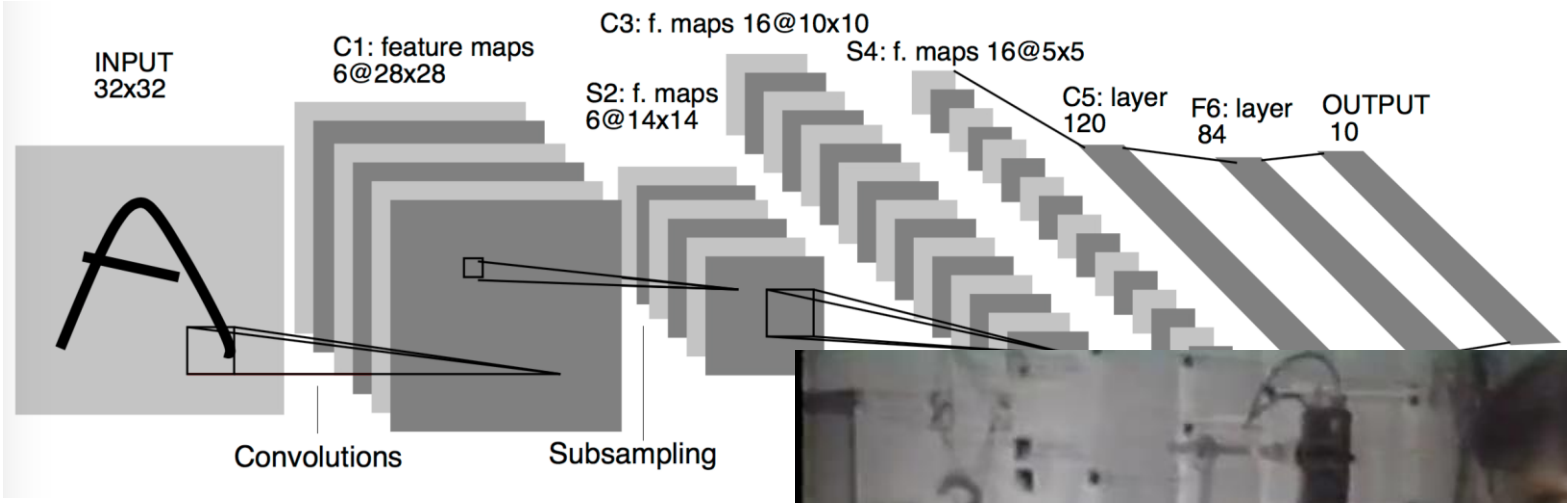
Batch size: 8



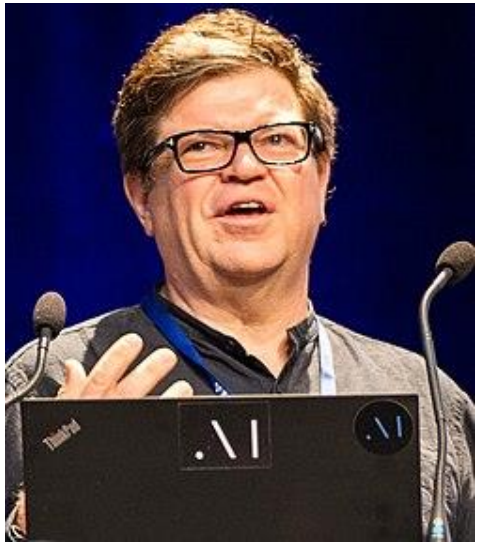
Convolutional Neural Networks (CNNs)

- **Convolutional network** is a concatenation of convolutional layers and activation functions
- Both forward and backward pass are convolutions → efficient implementation on GPUs (NVidia)

LeNet [1989-1998]



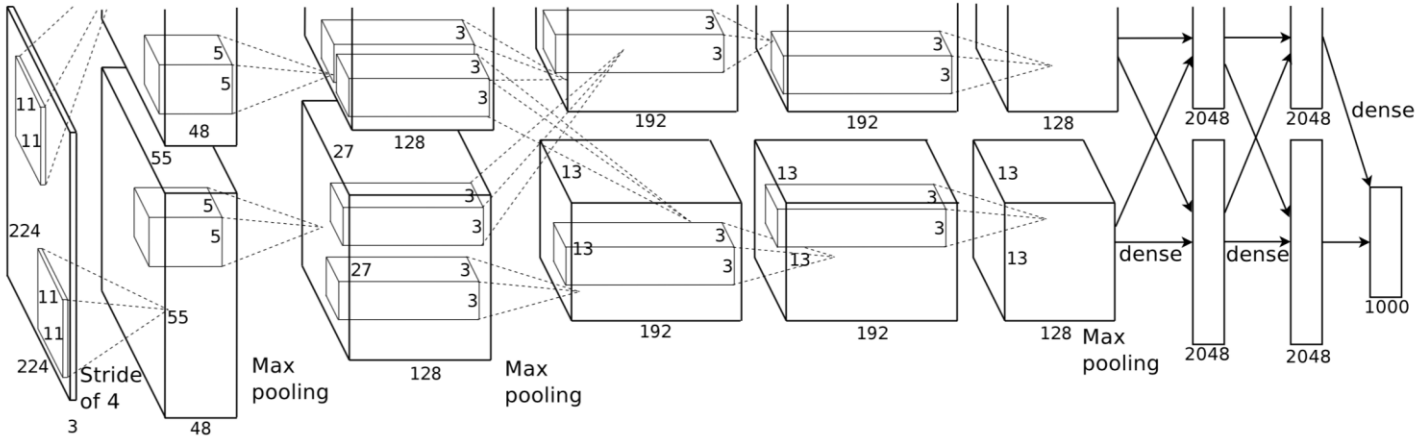
LeCun, Yann, et al. "Backpropagation applied to handwritten zip code recogn



AlexNet [2012]

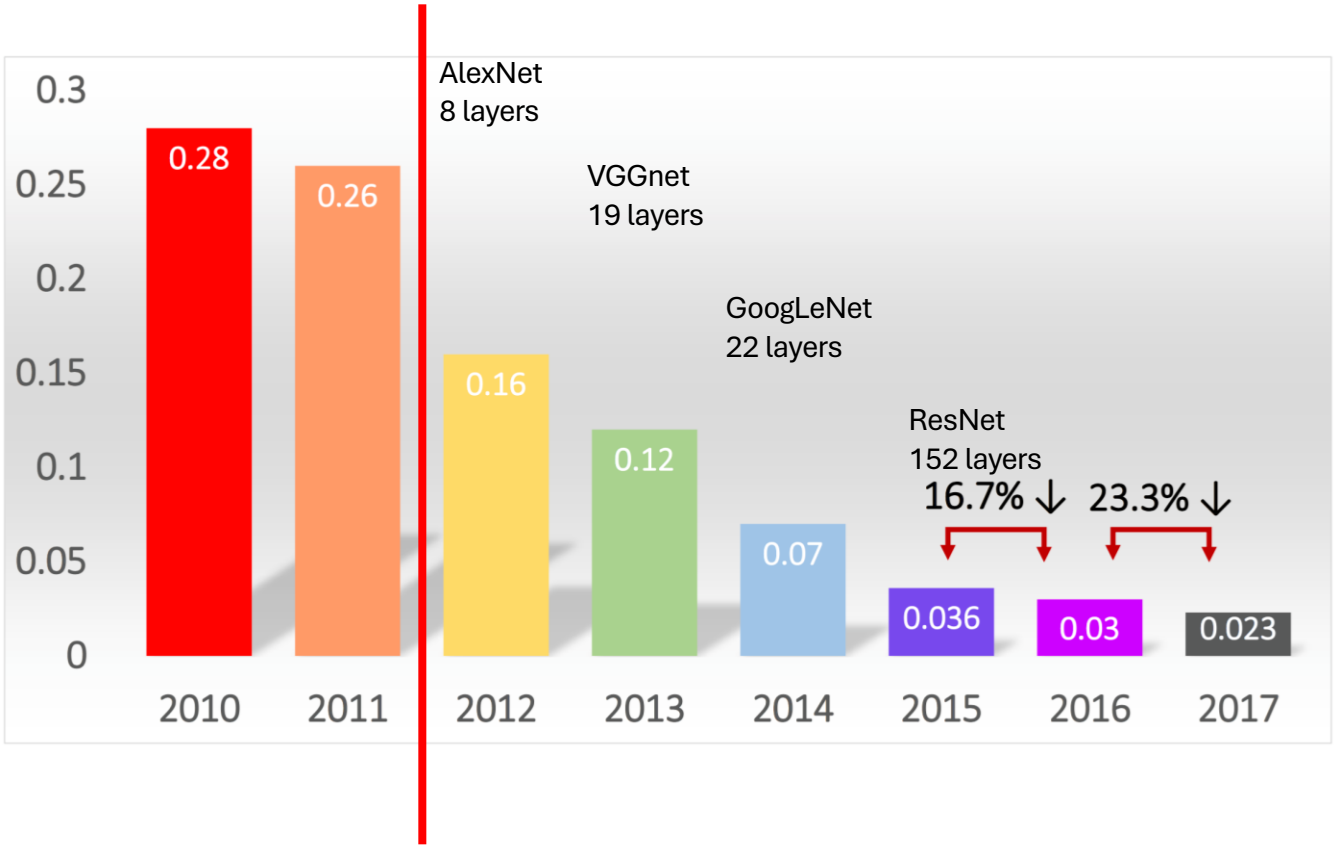
IMAGENET

1.2M images, 1000 classes



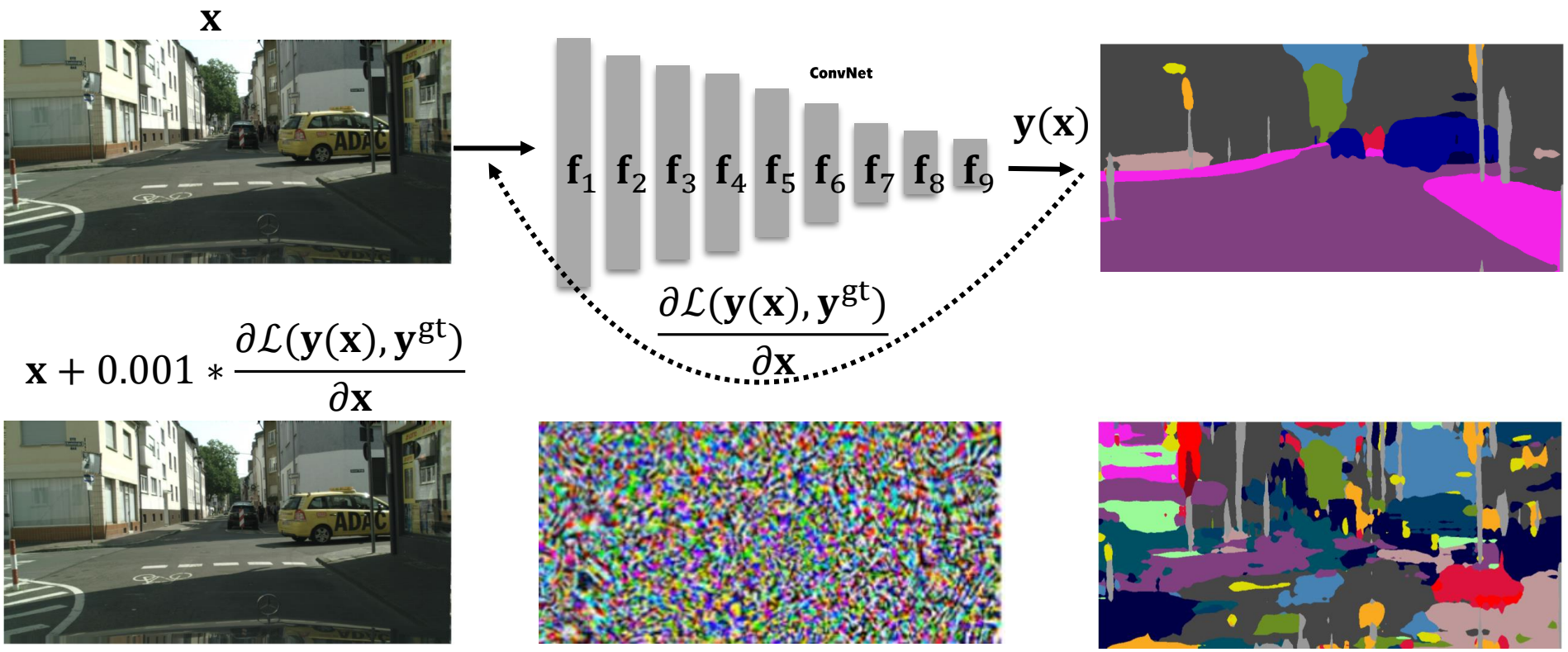
Alex Krizhevsky et al, ImageNet classification with deep convolutional neural networks, NIPS, 2012

ImageNet challenge



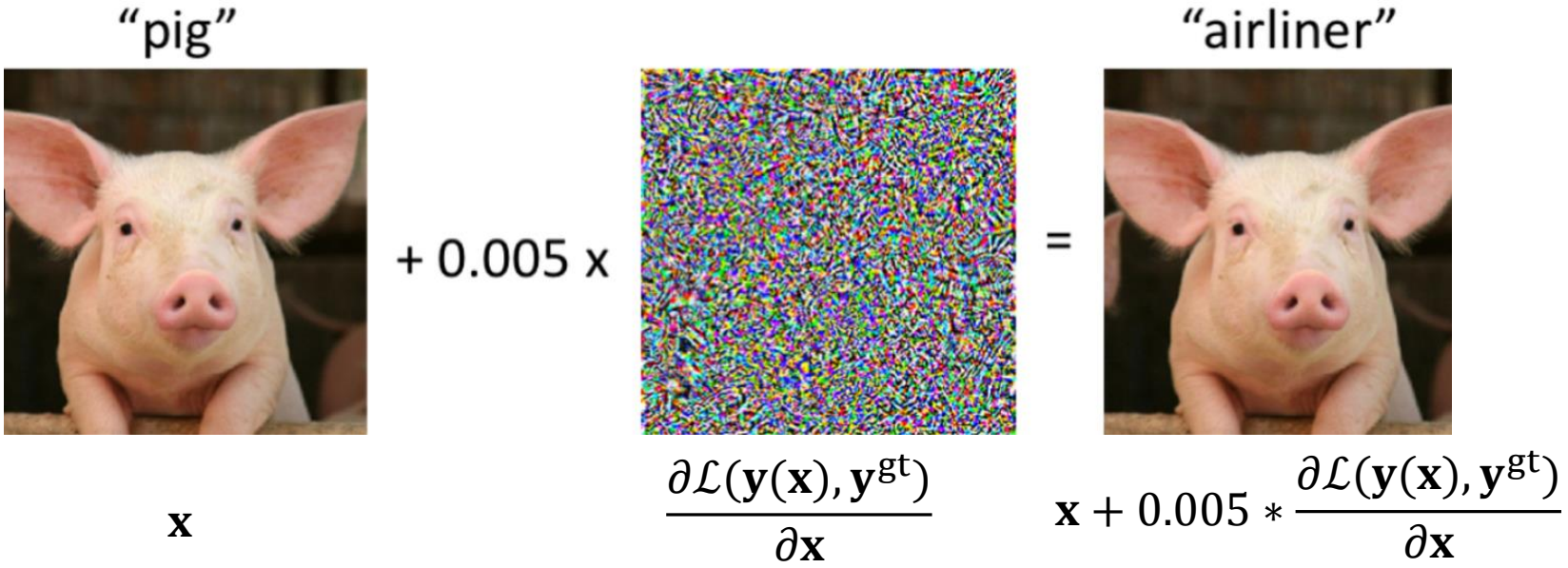
Adversarial attacks

- Given a trained network and an image, find the closest image on which the classification fails



Adversarial attacks

- Given a trained network and an image, find the closest image on which the classification fails



- White-box attacks (full access to the model)
- Black-box attacks (limited access)
- Perturbations (noise), adversary patches, adversary pixels

Adversarial attacks

Competencies gained for the test

- Convolutional forward and backward pass
- Convolutional layer hyper-parameters (channels, kernel size, stride, padding)
- Pros and cons of CNNs