

CPEN 455: Deep Learning

Lecture 5.1: Convolutional Neural Networks I

Renjie Liao

University of British Columbia

2026 Winter Term 1

October 2, 2026

Outline

- Invariance & Equivariance
- Convolution
 - 1D Convolution
 - Matrix Multiplication Views
 - Translation Equivariance
 - 2D Convolution
- Convolution Variants
 - Transposed Convolution
 - Dilated Convolution
 - Grouped Convolution
 - Separable Convolution
- Pooling
- Example Architectures

Outline

- **Invariance & Equivariance**
- Convolution
 - 1D Convolution
 - Matrix Multiplication Views
 - Translation Equivariance
 - 2D Convolution
- Convolution Variants
 - Transposed Convolution
 - Dilated Convolution
 - Grouped Convolution
 - Separable Convolution
- Pooling
- Example Architectures

Invariance & Equivariance

Invariance:

The output stays unchanged when the input is transformed.

Example: translating an object keeps its class label.

$$f(X) = f(g(X))$$

Invariance & Equivariance

Invariance:

The output stays unchanged when the input is transformed.

Example: translating an object keeps its class label.

$$f(X) = f(g(X))$$

Equivariance:

Applying a transformation before or after the function gives the same result.

Here, f is the function and g is a transformation (e.g., translation).

Example: a translated image produces a translated segmentation map.

$$g(f(X)) = f(g(X))$$

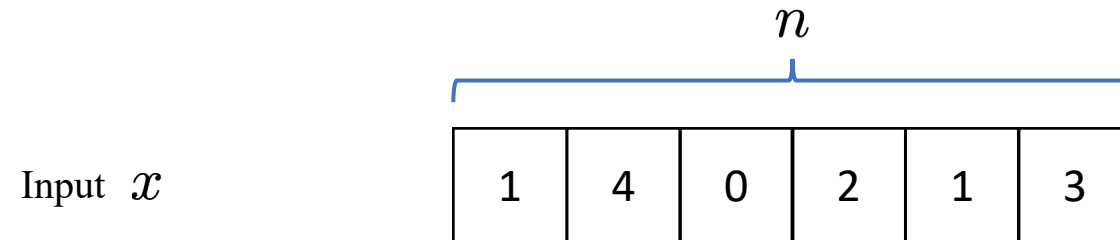
We will see when convolution is translation equivariant.

Outline

- Invariance & Equivariance
- Convolution
 - **1D Convolution**
 - Matrix Multiplication Views
 - Translation Equivariance
 - 2D Convolution
- Convolution Variants
 - Transposed Convolution
 - Dilated Convolution
 - Grouped Convolution
 - Separable Convolution
- Pooling
- Example Architectures

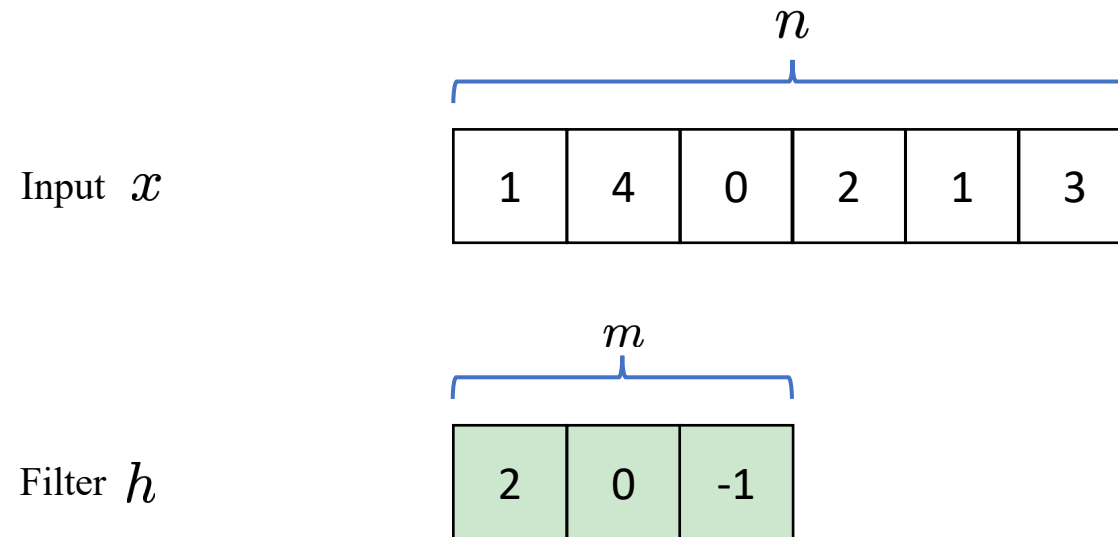
1D (Discrete) Convolutions

Let us see what 1D (Discrete) Convolution looks like



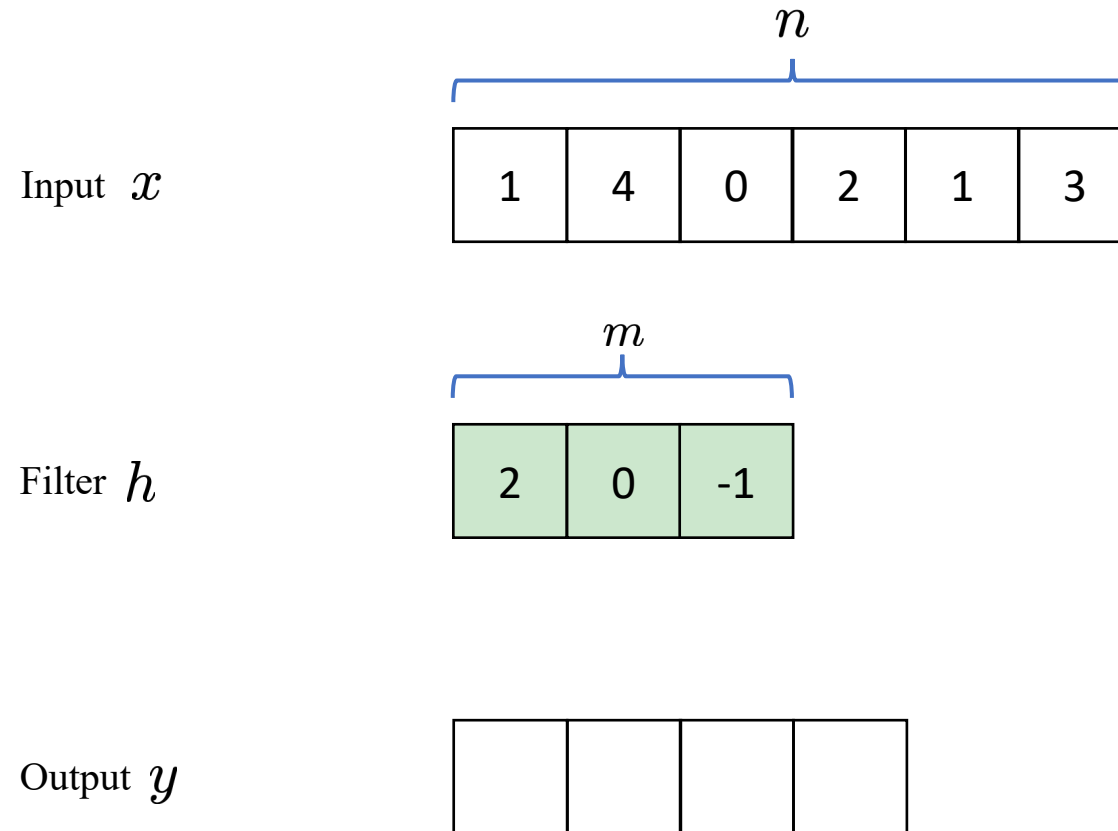
1D (Discrete) Convolutions

Let us see what 1D (Discrete) Convolution looks like



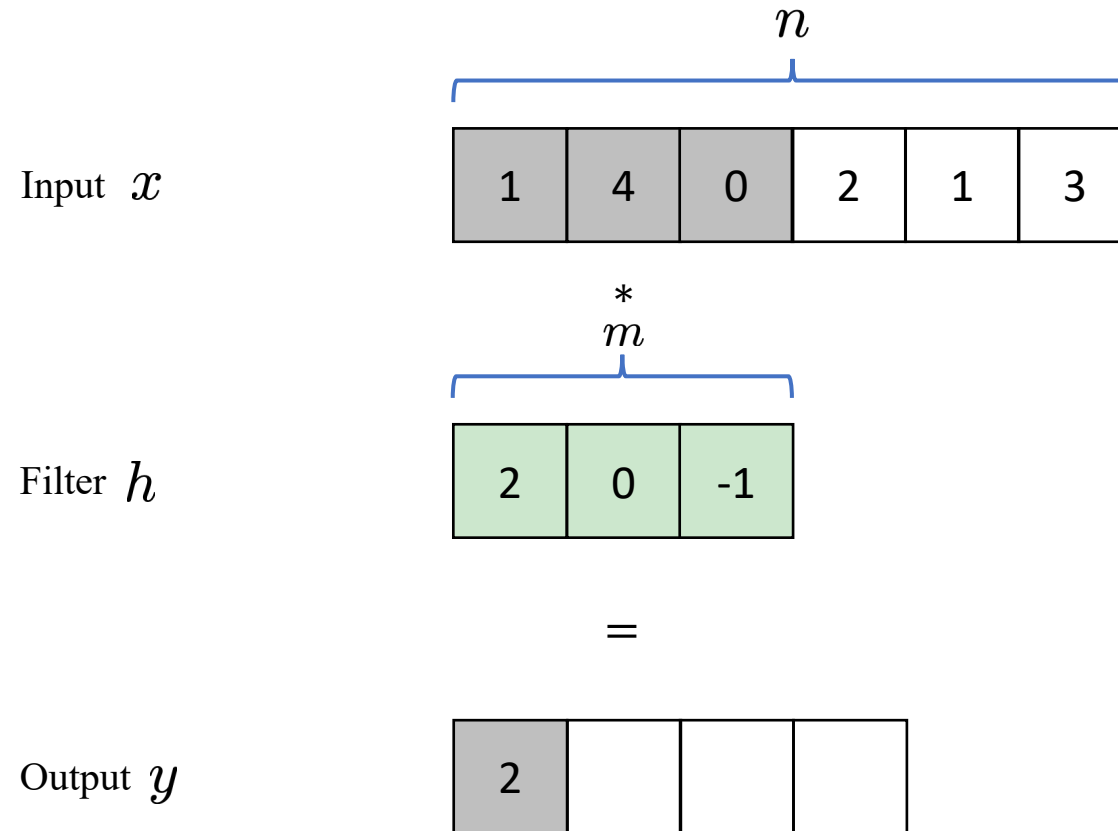
1D (Discrete) Convolutions

Let us see what 1D (Discrete) Convolution looks like



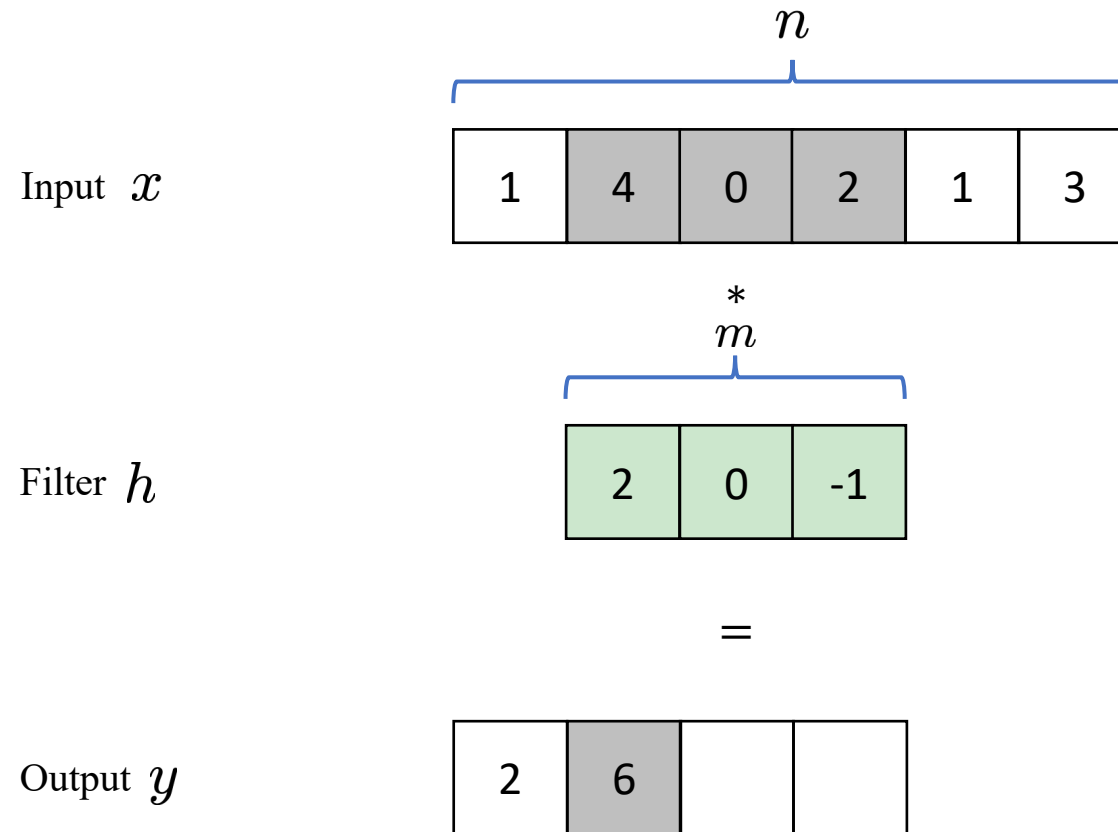
1D (Discrete) Convolutions

Let us see what 1D (Discrete) Convolution looks like



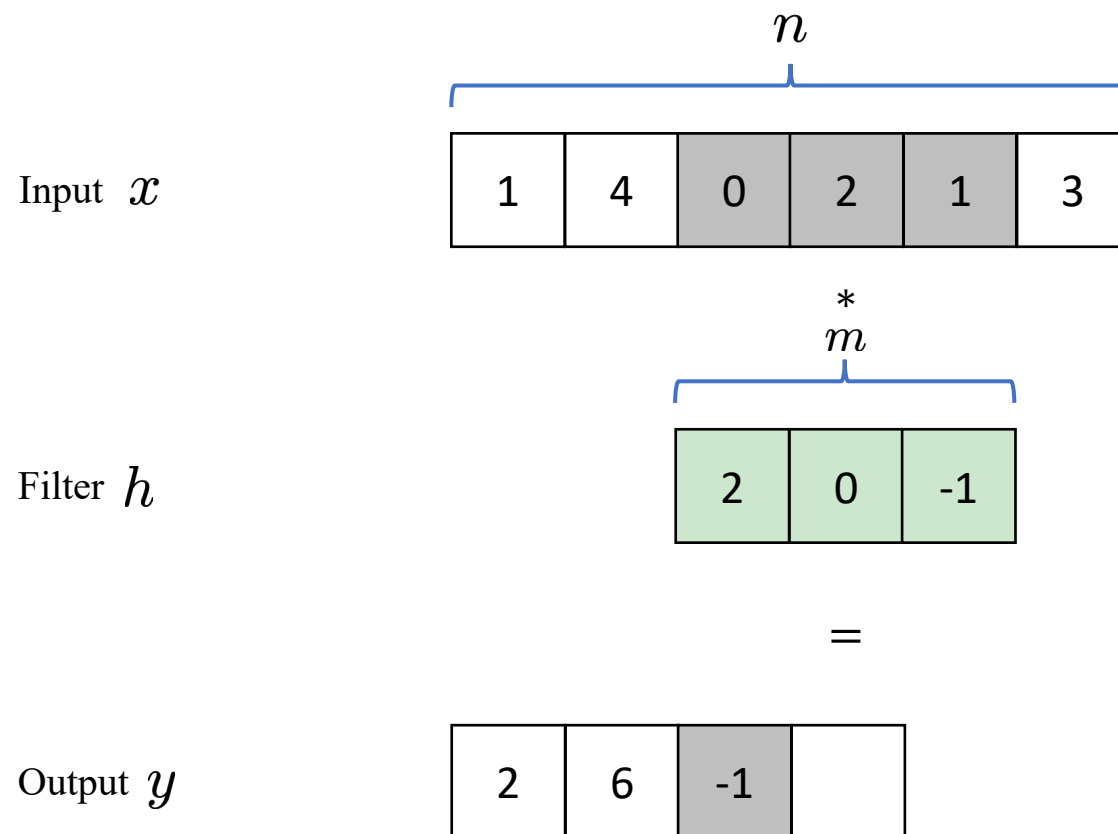
1D (Discrete) Convolutions

Let us see what 1D (Discrete) Convolution looks like



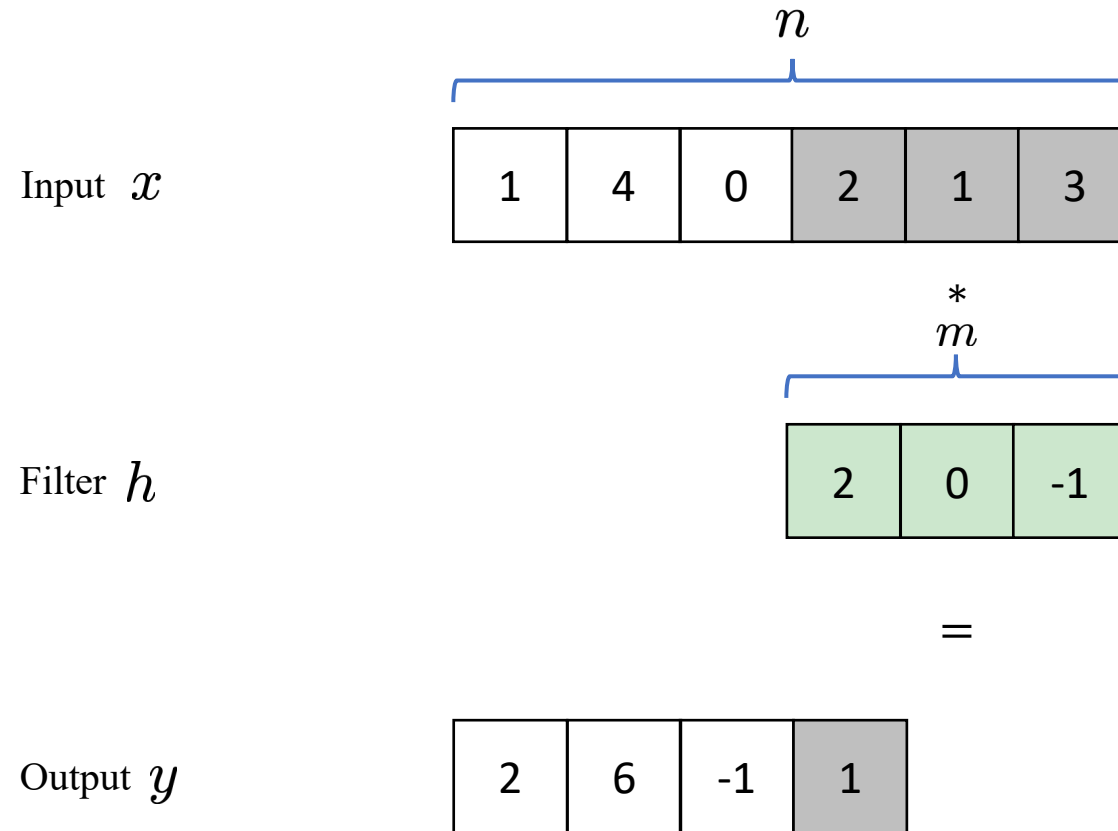
1D (Discrete) Convolutions

Let us see what 1D (Discrete) Convolution looks like



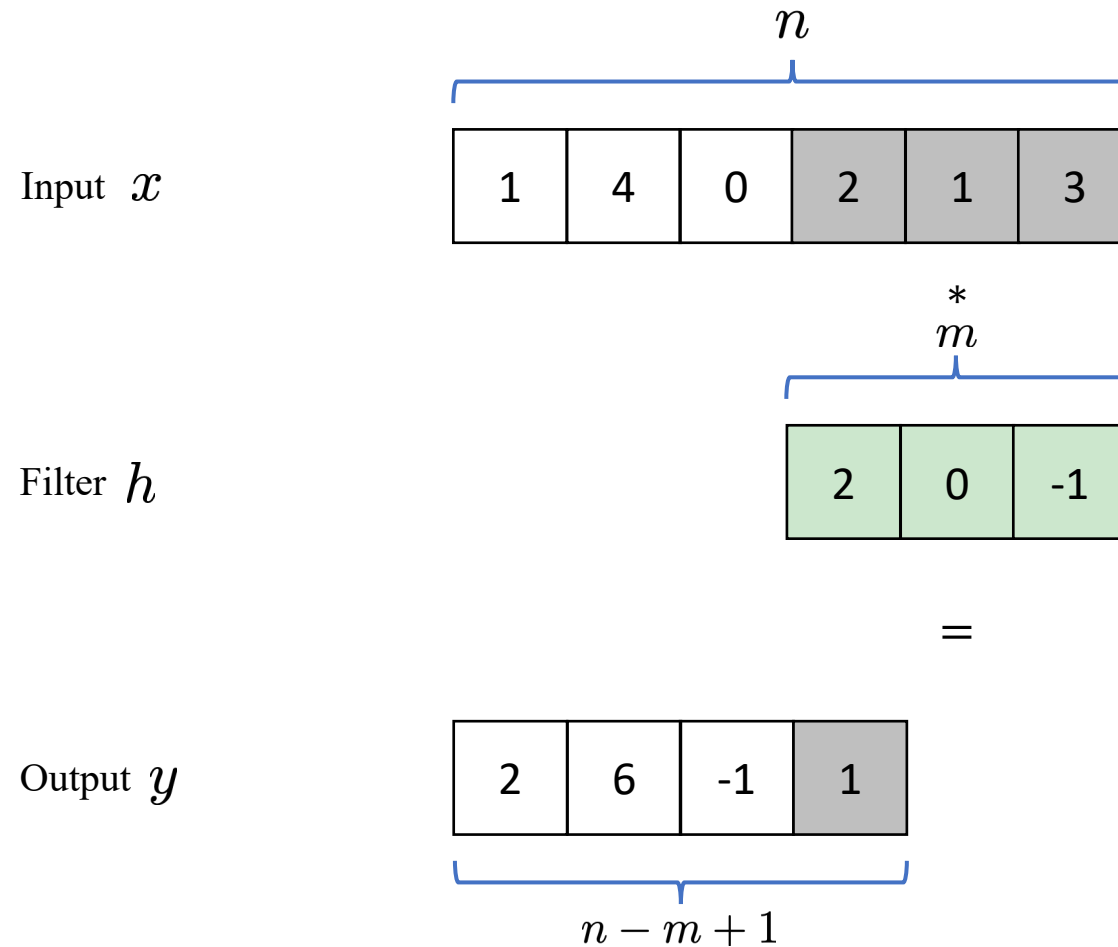
1D (Discrete) Convolutions

Let us see what 1D (Discrete) Convolution looks like



1D (Discrete) Convolutions

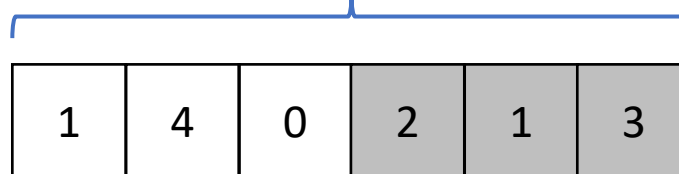
Let us see what 1D (Discrete) Convolution looks like



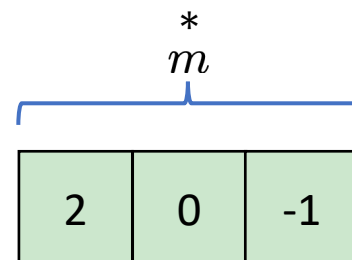
1D (Discrete) Convolutions

We denote 1D convolution by $y = h * x$ and $y[i] = \sum_{j=1}^m h_j x_{i+j-1}$

Input x

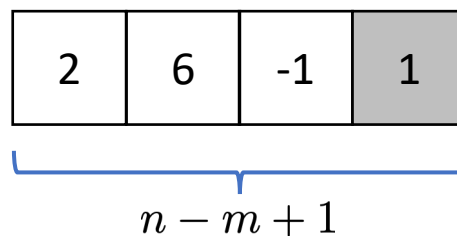


Filter h



=

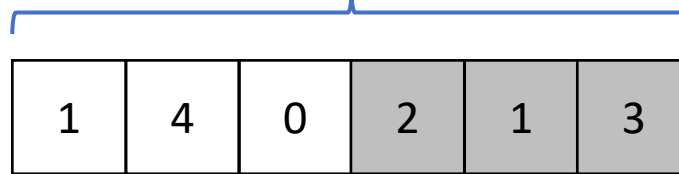
Output y



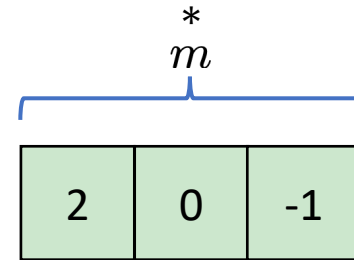
1D (Discrete) Convolutions

We denote 1D convolution by $y = h * x$ and $y[i] = \sum_{j=1}^m h_j x_{i+j-1}$

Input x

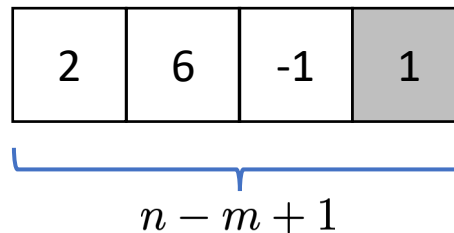


Filter h



=

Output y



Cross-correlation (deep learning):

$$\sum_{j=1}^m h_j x_{i+j-1}$$

Convolution (reversed filter):

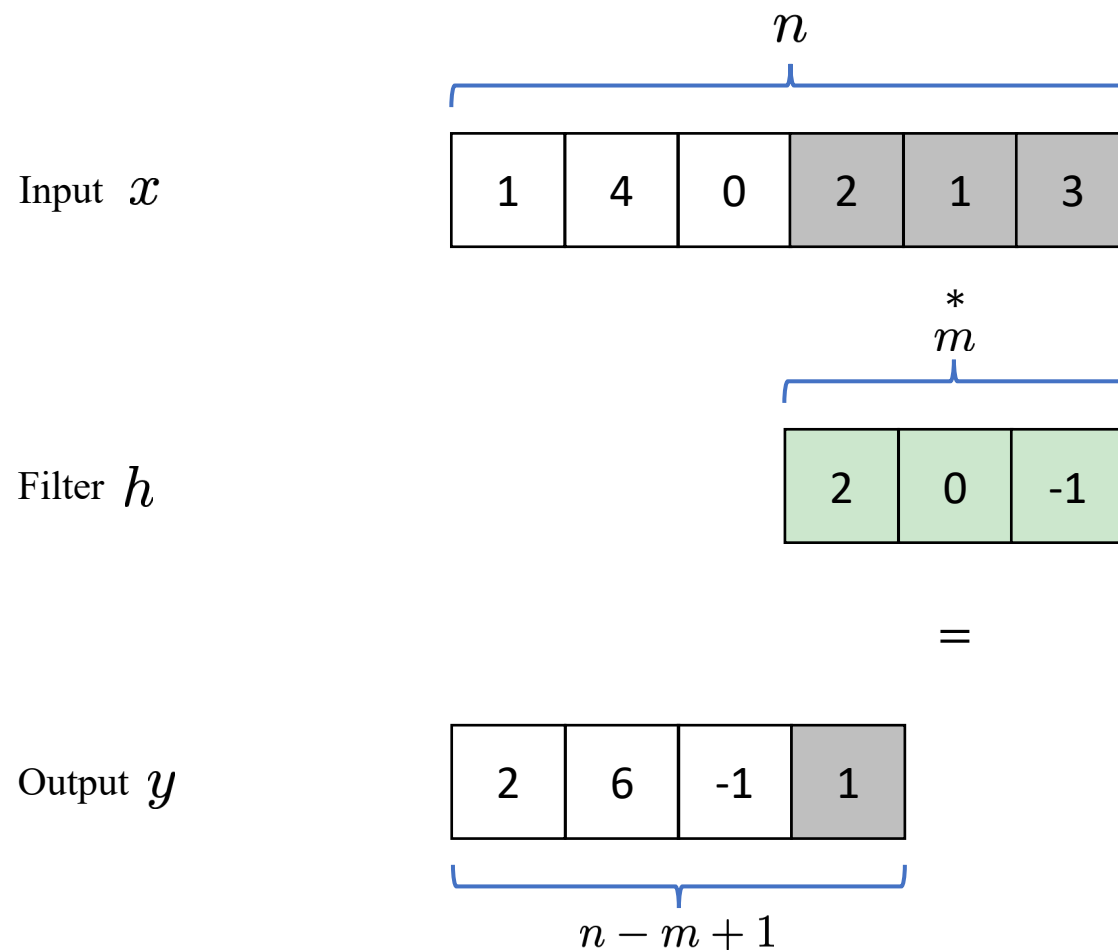
$$\sum_{j=1}^m h_{m-j+1} x_{i+j-1}$$

Reversing the filter changes the convention.

Learned filters give the same model class.

1D (Discrete) Convolutions

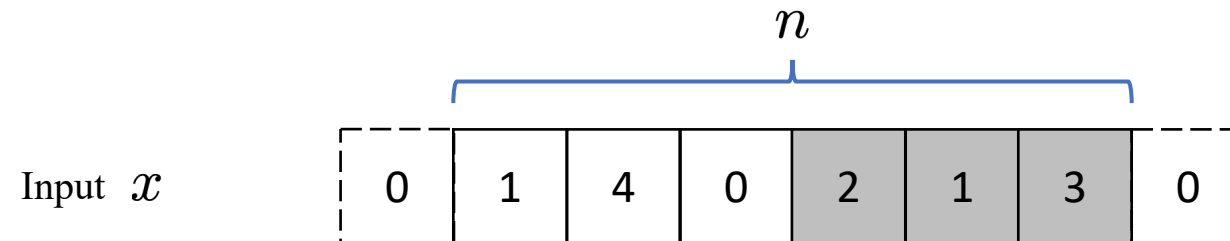
How can the output keep the input's shape?



1D (Discrete) Convolutions

How can the output keep the input's shape?

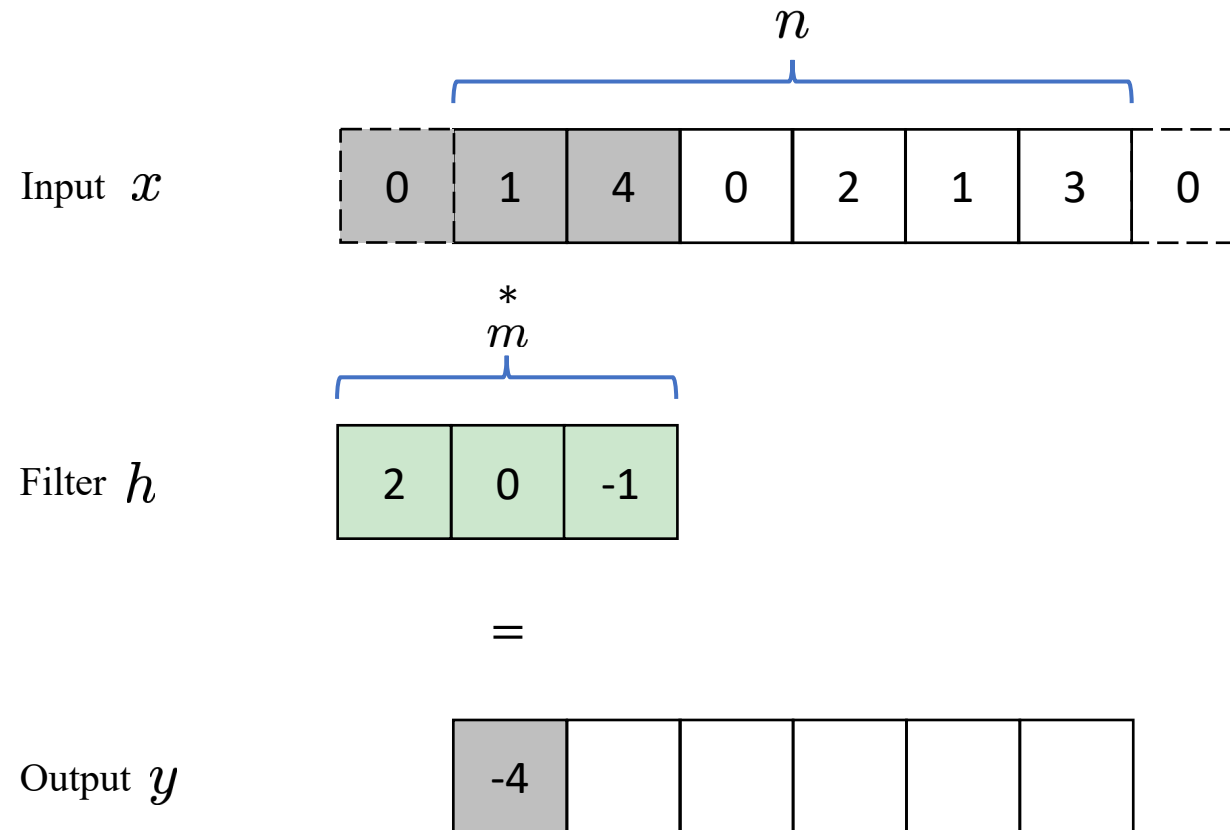
Padding!



1D (Discrete) Convolutions

How can the output keep the input's shape?

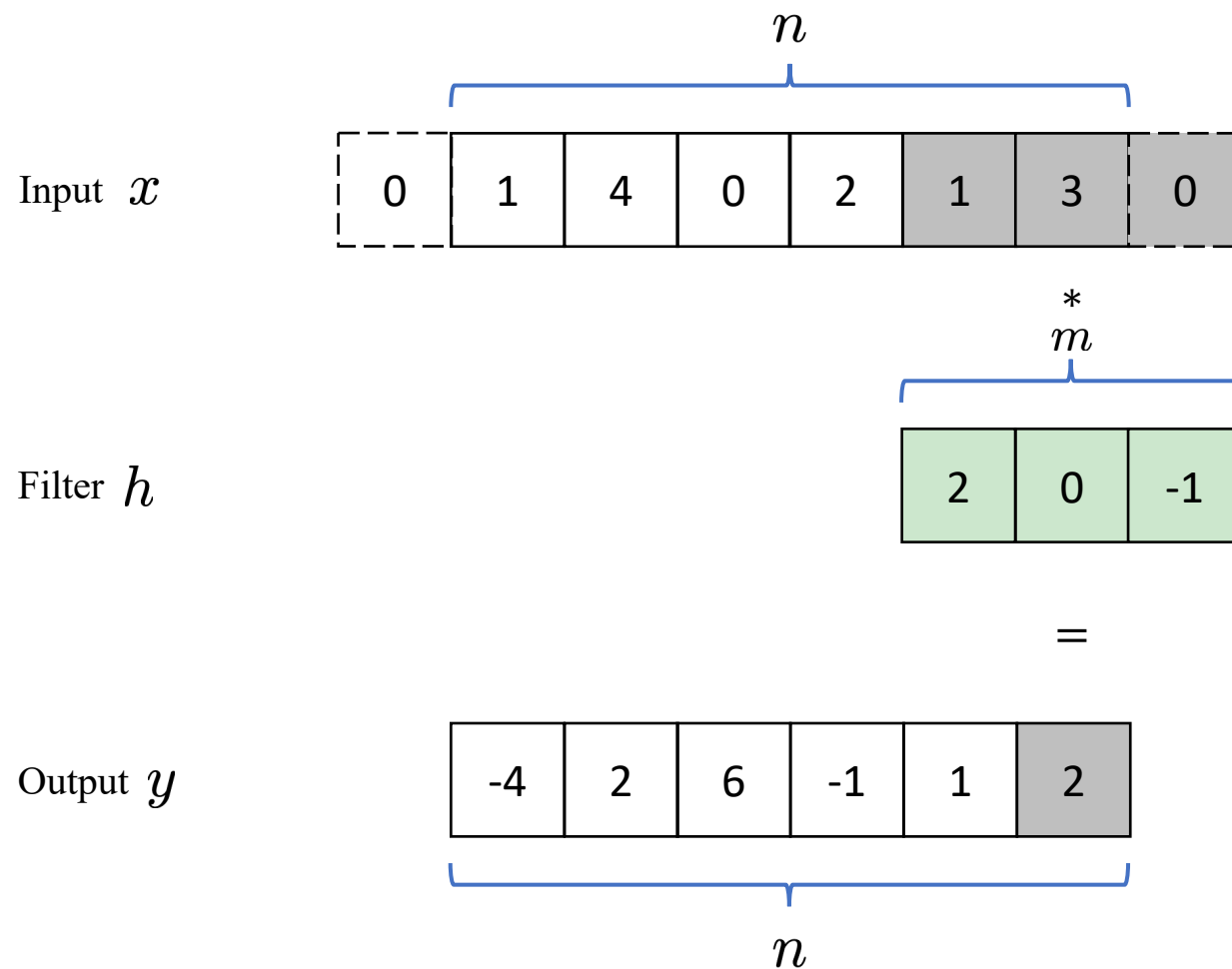
Padding!



1D (Discrete) Convolutions

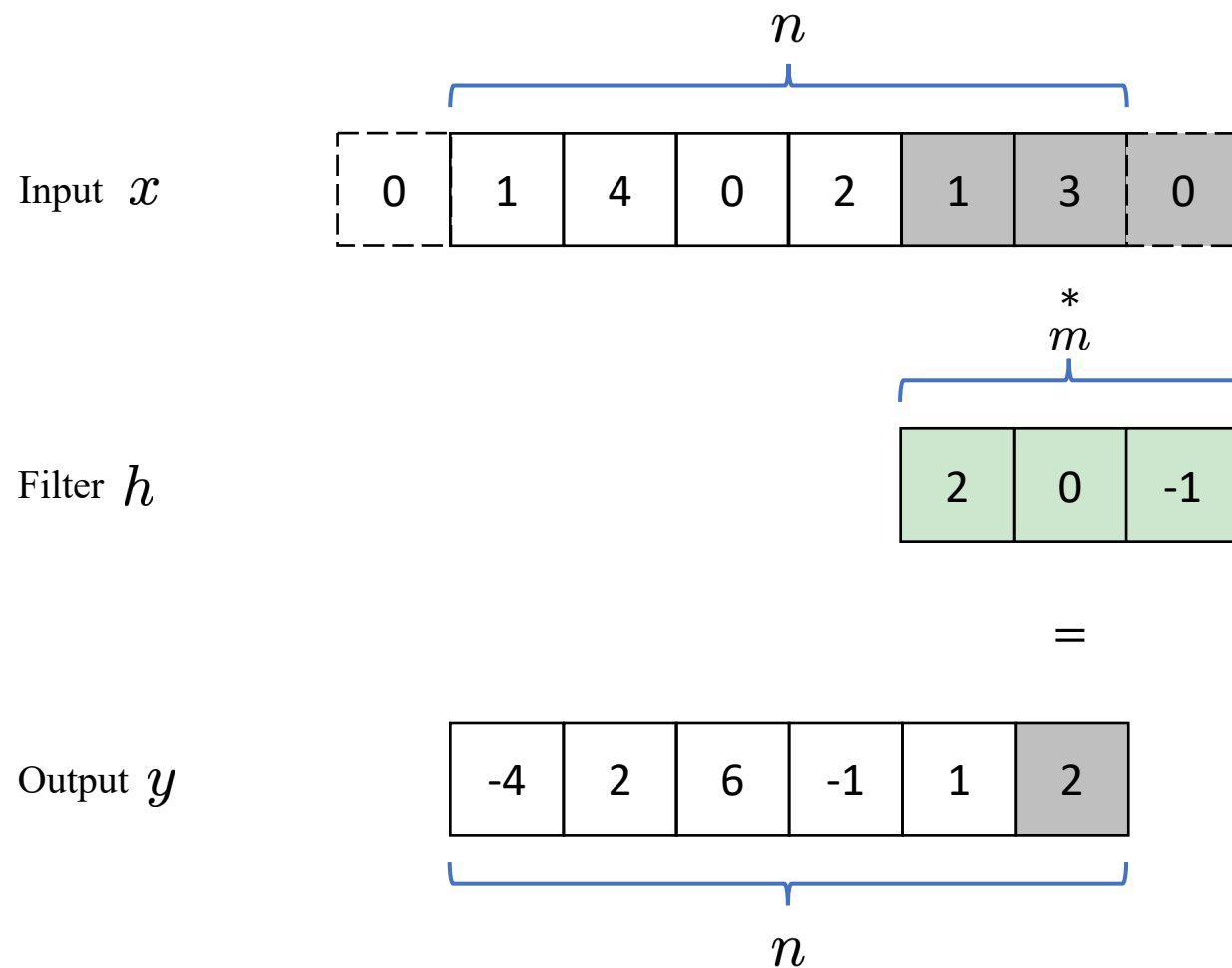
How can the output keep the input's shape?

Padding!



1D (Discrete) Convolutions

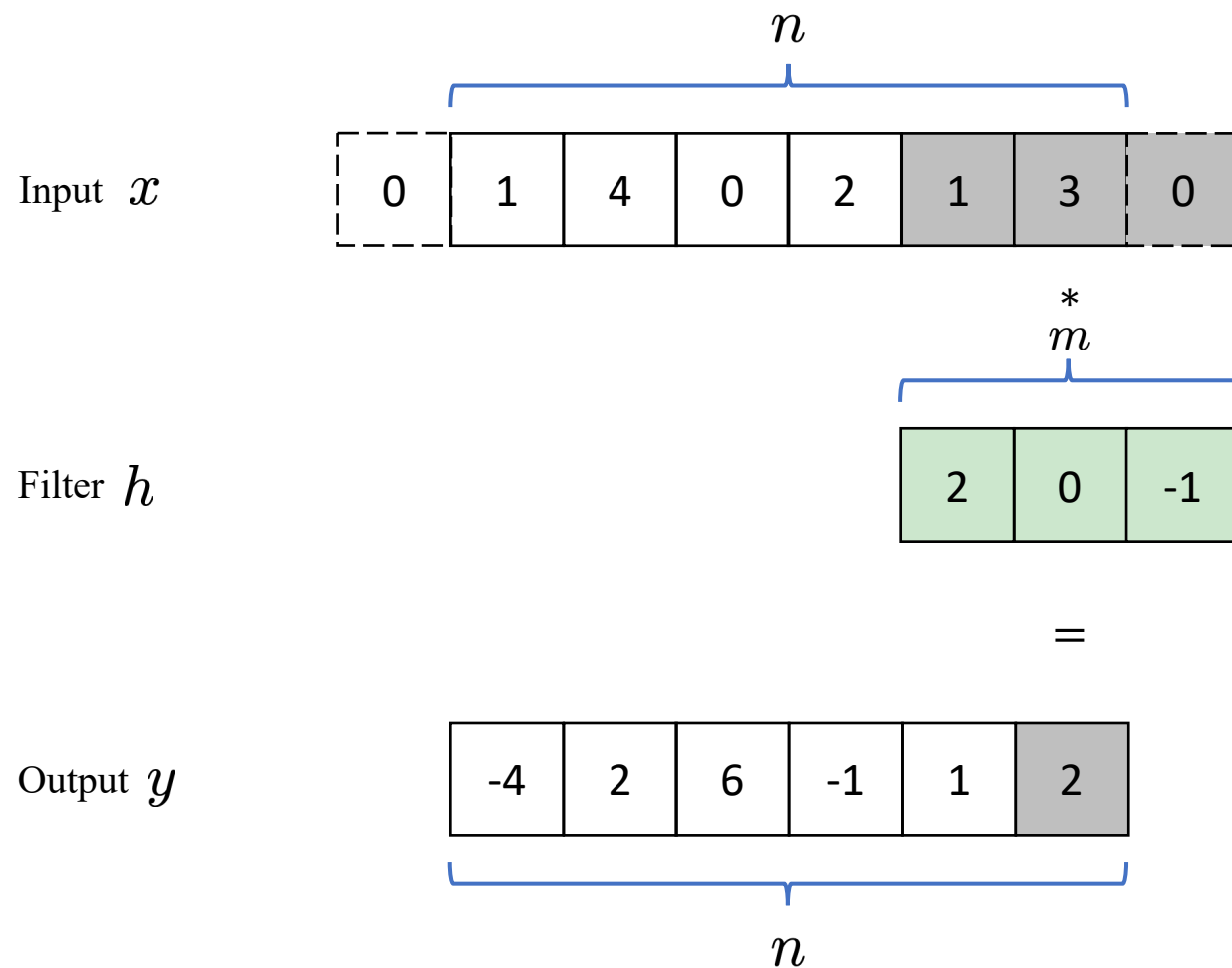
How can we make the output smaller?



1D (Discrete) Convolutions

How can we make the output smaller?

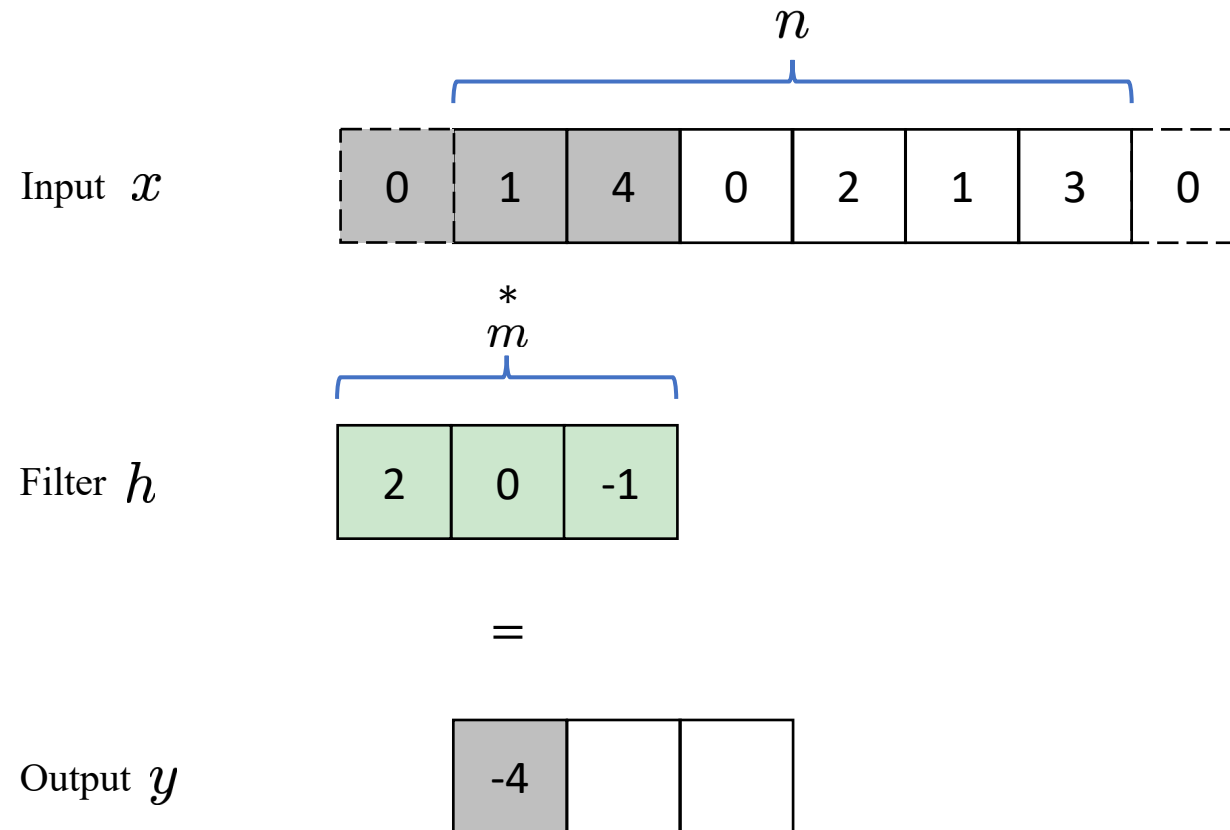
Stride!



1D (Discrete) Convolutions

How can we make the output smaller?

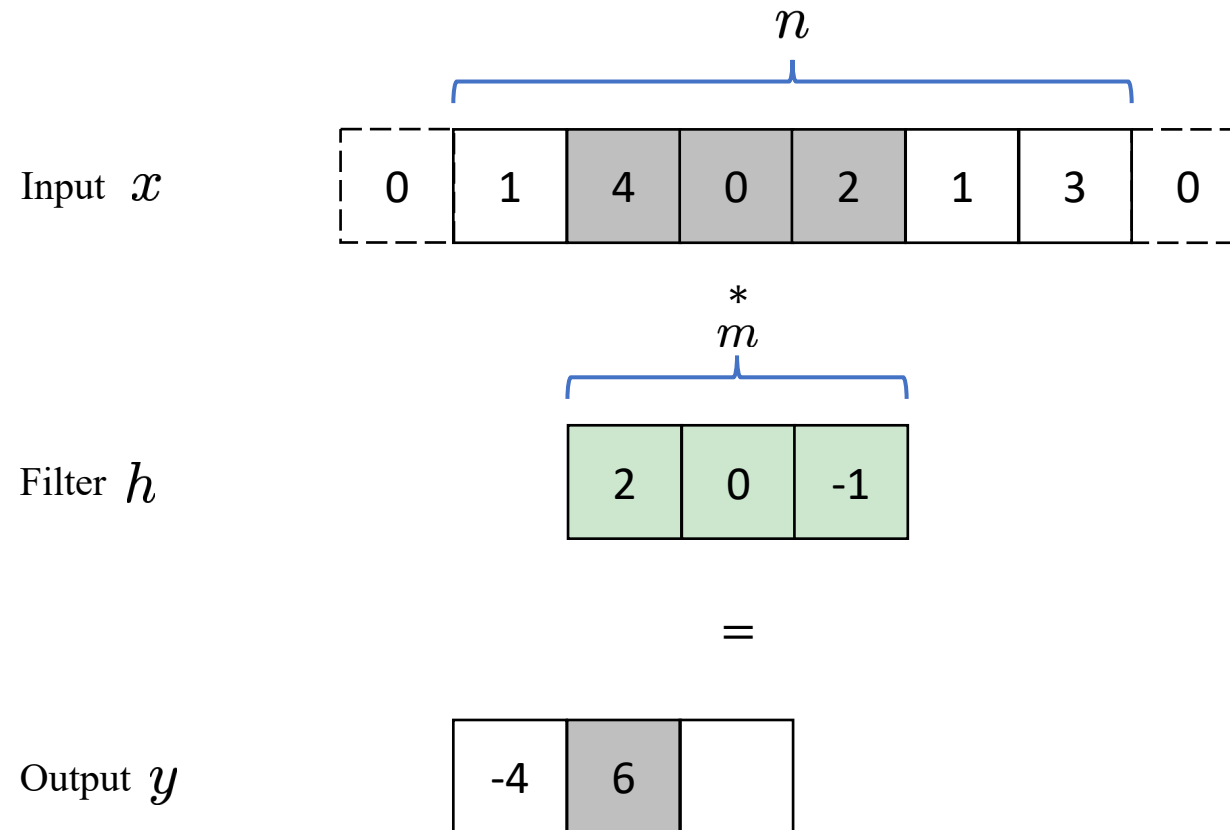
Stride = 2!



1D (Discrete) Convolutions

How can we make the output smaller?

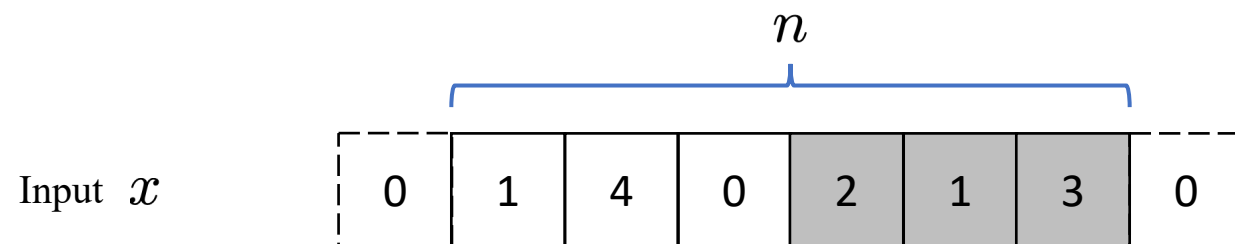
Stride = 2!



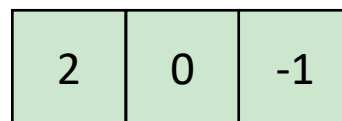
1D (Discrete) Convolutions

How can we make the output smaller?

Stride = 2!

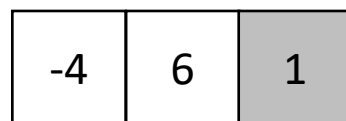


Filter h



=

Output y



$$\left\lfloor \frac{n + 2p - m}{s} \right\rfloor + 1$$

Stride: s

Padding: p

Outline

- Invariance & Equivariance
- Convolution
 - 1D Convolution
 - **Matrix Multiplication Views**
 - Translation Equivariance
 - 2D Convolution
- Convolution Variants
 - Transposed Convolution
 - Dilated Convolution
 - Grouped Convolution
 - Separable Convolution
- Pooling
- Example Architectures

Matrix Multiplication View I

1D convolution as matrix multiplication

Filter $\Rightarrow$ Toeplitz matrix (diagonal-constant)

Matrix Multiplication View I

1D convolution as matrix multiplication

Filter => Toeplitz matrix (diagonal-constant)

$$y = h * x = \begin{bmatrix} h_1 & \cdots & h_{\lfloor m/2 \rfloor + 1} & \cdots & h_m & 0 & \cdots & \cdots & \cdots & 0 \\ 0 & h_1 & \cdots & h_{\lfloor m/2 \rfloor + 1} & \cdots & h_{m-1} & h_m & 0 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & \cdots & \cdots & h_1 & h_2 & \cdots & h_m & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & \cdots & 0 & h_1 & \cdots & h_{\lfloor m/2 \rfloor + 2} & \cdots & h_m & 0 \\ 0 & 0 & \cdots & \cdots & 0 & h_1 & \cdots & h_{\lfloor m/2 \rfloor + 1} & \cdots & h_m \end{bmatrix} \begin{bmatrix} 0 \\ \vdots \\ x_1 \\ x_2 \\ \vdots \\ x_n \\ \vdots \\ 0 \end{bmatrix}$$

padding: $\lfloor m/2 \rfloor$

padding: $\lfloor m/2 \rfloor$

Matrix Multiplication View I

1D convolution as matrix multiplication

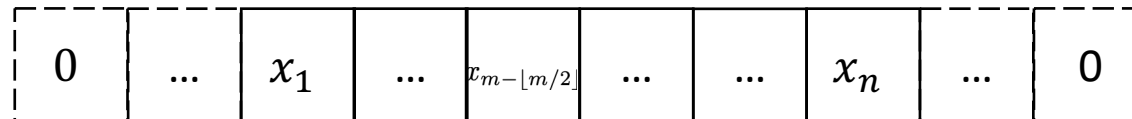
Filter => Toeplitz matrix (diagonal-constant)

$$y = h * x = \begin{bmatrix} h_1 & \cdots & h_{\lfloor m/2 \rfloor + 1} & \cdots & h_m & 0 & \cdots & \cdots & \cdots & 0 \\ 0 & h_1 & \cdots & h_{\lfloor m/2 \rfloor + 1} & \cdots & h_{m-1} & h_m & 0 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & \cdots & \cdots & h_1 & h_2 & \cdots & h_m & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & \cdots & 0 & h_1 & \cdots & h_{\lfloor m/2 \rfloor + 2} & \cdots & h_m & 0 \\ 0 & 0 & \cdots & \cdots & 0 & h_1 & \cdots & h_{\lfloor m/2 \rfloor + 1} & \cdots & h_m \end{bmatrix} \begin{bmatrix} 0 \\ \vdots \\ x_1 \\ x_2 \\ \vdots \\ x_n \\ \vdots \\ 0 \end{bmatrix}$$

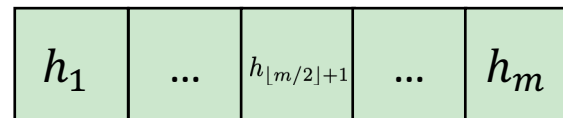
padding: $\lfloor m/2 \rfloor$

padding: $\lfloor m/2 \rfloor$

Input x



Filter h



Matrix Multiplication View I

1D convolution as matrix multiplication

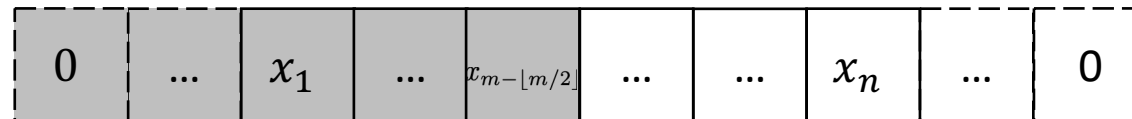
Filter $\Rightarrow$ Toeplitz matrix (diagonal-constant)

$$y = h * x = \begin{bmatrix} h_1 & \cdots & h_{\lfloor m/2 \rfloor + 1} & \cdots & h_m & 0 & \cdots & \cdots & \cdots & 0 \\ 0 & h_1 & \cdots & h_{\lfloor m/2 \rfloor + 1} & \cdots & h_{m-1} & h_m & 0 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & \cdots & \cdots & h_1 & h_2 & \cdots & h_m & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & \cdots & 0 & h_1 & \cdots & h_{\lfloor m/2 \rfloor + 2} & \cdots & h_m & 0 \\ 0 & 0 & \cdots & \cdots & 0 & h_1 & \cdots & h_{\lfloor m/2 \rfloor + 1} & \cdots & h_m \end{bmatrix} \begin{bmatrix} 0 \\ \vdots \\ x_1 \\ x_2 \\ \vdots \\ x_n \\ \vdots \\ 0 \end{bmatrix}$$

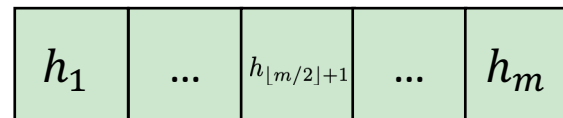
padding: $\lfloor m/2 \rfloor$

padding: $\lfloor m/2 \rfloor$

Input x



Filter h



Matrix Multiplication View I

1D convolution as matrix multiplication

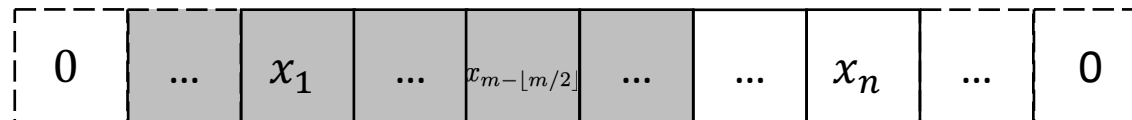
Filter $\Rightarrow$ Toeplitz matrix (diagonal-constant)

$$y = h * x = \begin{bmatrix} h_1 & \cdots & h_{\lfloor m/2 \rfloor + 1} & \cdots & h_m & 0 & \cdots & \cdots & \cdots & 0 \\ 0 & h_1 & \cdots & h_{\lfloor m/2 \rfloor + 1} & \cdots & h_{m-1} & h_m & 0 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & \cdots & \cdots & h_1 & h_2 & \cdots & h_m & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & \cdots & 0 & h_1 & \cdots & h_{\lfloor m/2 \rfloor + 2} & \cdots & h_m & 0 \\ 0 & 0 & \cdots & \cdots & 0 & h_1 & \cdots & h_{\lfloor m/2 \rfloor + 1} & \cdots & h_m \end{bmatrix} \begin{bmatrix} 0 \\ \vdots \\ x_1 \\ x_2 \\ \vdots \\ x_n \\ \vdots \\ 0 \end{bmatrix}$$

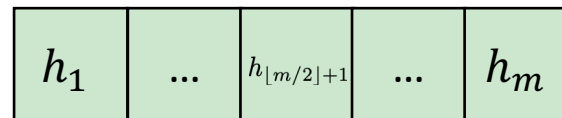
padding: $\lfloor m/2 \rfloor$

padding: $\lfloor m/2 \rfloor$

Input x



Filter h



Matrix Multiplication View I

1D convolution as matrix multiplication

Filter $\Rightarrow$ Toeplitz matrix (diagonal-constant)

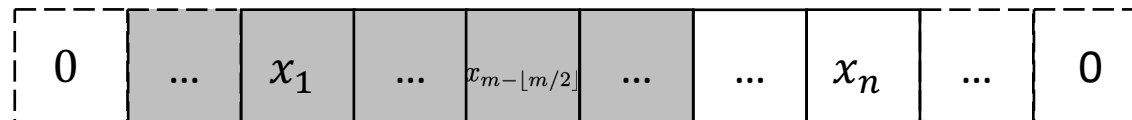
Very sparse when $n \gg m$.

$$y = h * x = \begin{bmatrix} h_1 & \cdots & h_{\lfloor m/2 \rfloor + 1} & \cdots & h_m & 0 & \cdots & \cdots & \cdots & 0 \\ 0 & h_1 & \cdots & h_{\lfloor m/2 \rfloor + 1} & \cdots & h_{m-1} & h_m & 0 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & \cdots & \cdots & h_1 & h_2 & \cdots & h_m & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & \cdots & 0 & h_1 & \cdots & h_{\lfloor m/2 \rfloor + 2} & \cdots & h_m & 0 \\ 0 & 0 & \cdots & \cdots & 0 & h_1 & \cdots & h_{\lfloor m/2 \rfloor + 1} & \cdots & h_m \end{bmatrix} \begin{bmatrix} 0 \\ \vdots \\ x_1 \\ x_2 \\ \vdots \\ x_n \\ \vdots \\ 0 \end{bmatrix}$$

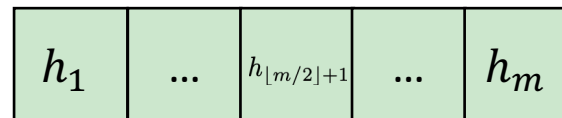
padding: $\lfloor m/2 \rfloor$

padding: $\lfloor m/2 \rfloor$

Input x



Filter h



Matrix Multiplication View II

1D convolution as matrix multiplication

Data $\Rightarrow$ Toeplitz matrix (diagonal-constant)

Matrix Multiplication View II

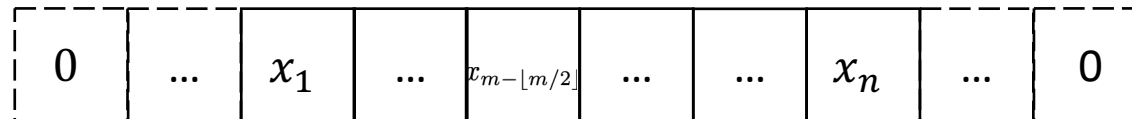
1D convolution as matrix multiplication

Data => Toeplitz matrix (diagonal-constant)

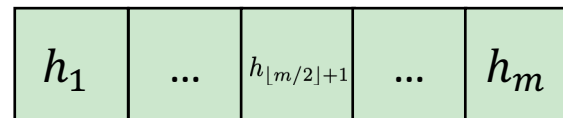
$$y^\top = (h * x)^\top$$

$$= \begin{bmatrix} h_1 & \cdots & h_{\lfloor m/2 \rfloor + 1} & \cdots & h_m \end{bmatrix} \begin{bmatrix} \text{padding: } \lfloor m/2 \rfloor \left\{ \begin{array}{ccccccc} 0 & 0 & \cdots & x_1 & \cdots & x_{n-\lfloor m/2 \rfloor - 1} & x_{n-\lfloor m/2 \rfloor} \\ \vdots & \vdots & \cdots & \vdots & \cdots & \vdots & \vdots \\ 0 & x_1 & \cdots & x_{\lfloor m/2 \rfloor} & \cdots & x_{n-2} & x_{n-1} \\ x_1 & x_2 & \cdots & x_{\lfloor m/2 \rfloor + 1} & \cdots & x_{n-1} & x_n \\ x_2 & x_3 & \cdots & x_{\lfloor m/2 \rfloor + 2} & \cdots & x_n & 0 \\ \vdots & \vdots & \cdots & \vdots & \cdots & \vdots & \vdots \\ x_{m-\lfloor m/2 \rfloor} & x_{m-\lfloor m/2 \rfloor + 1} & \cdots & x_m & \cdots & 0 & 0 \end{array} \right. \end{bmatrix}$$

Input x



Filter h



Matrix Multiplication View II

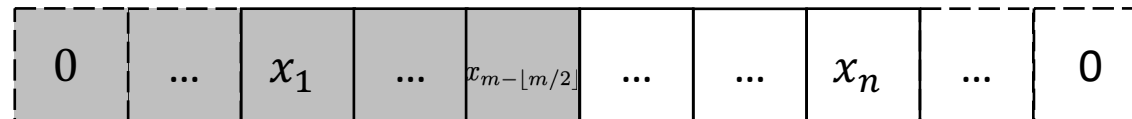
1D convolution as matrix multiplication

Data => Toeplitz matrix (diagonal-constant)

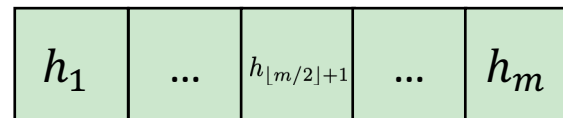
$$y^\top = (h * x)^\top$$

$$= \begin{bmatrix} h_1 & \cdots & h_{\lfloor m/2 \rfloor + 1} & \cdots & h_m \end{bmatrix} \begin{bmatrix} \text{padding: } \lfloor m/2 \rfloor \left\{ \begin{array}{c} 0 \\ \vdots \\ 0 \end{array} \right. & 0 & \cdots & x_1 & \cdots & x_{n-\lfloor m/2 \rfloor - 1} & x_{n-\lfloor m/2 \rfloor} \\ \vdots & \vdots & \cdots & \vdots & \cdots & \vdots & \vdots \\ x_1 & x_2 & \cdots & x_{\lfloor m/2 \rfloor + 1} & \cdots & x_{n-2} & x_{n-1} \\ x_2 & x_3 & \cdots & x_{\lfloor m/2 \rfloor + 2} & \cdots & x_n & 0 \\ \vdots & \vdots & \cdots & \vdots & \cdots & \vdots & \vdots \\ x_{m-\lfloor m/2 \rfloor} & x_{m-\lfloor m/2 \rfloor + 1} & \cdots & x_m & \cdots & 0 & 0 \end{bmatrix}$$

Input x



Filter h



Matrix Multiplication View II

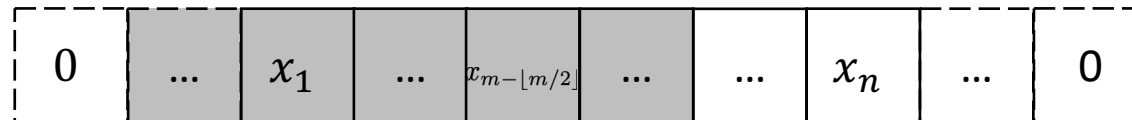
1D convolution as matrix multiplication

Data => Toeplitz matrix (diagonal-constant)

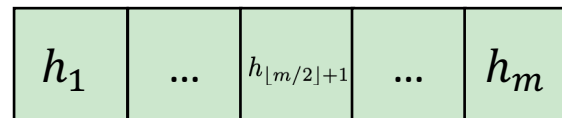
$$y^\top = (h * x)^\top$$

$$= \begin{bmatrix} h_1 & \cdots & h_{\lfloor m/2 \rfloor + 1} & \cdots & h_m \end{bmatrix} \begin{bmatrix} \text{padding: } \lfloor m/2 \rfloor & \begin{bmatrix} 0 & 0 & \cdots & x_1 & \cdots & x_{n-\lfloor m/2 \rfloor - 1} & x_{n-\lfloor m/2 \rfloor} \\ \vdots & \vdots & \cdots & \vdots & \cdots & \vdots & \vdots \\ 0 & x_1 & \cdots & x_{\lfloor m/2 \rfloor} & \cdots & x_{n-2} & x_{n-1} \\ x_1 & x_2 & \cdots & x_{\lfloor m/2 \rfloor + 1} & \cdots & x_{n-1} & x_n \\ x_2 & x_3 & \cdots & x_{\lfloor m/2 \rfloor + 2} & \cdots & x_n & 0 \\ \vdots & \vdots & \cdots & \vdots & \cdots & \vdots & \vdots \\ x_{m-\lfloor m/2 \rfloor} & x_{m-\lfloor m/2 \rfloor + 1} & \cdots & x_m & \cdots & 0 & 0 \end{bmatrix} \end{bmatrix}$$

Input x



Filter h



Matrix Multiplication View II

1D convolution as matrix multiplication

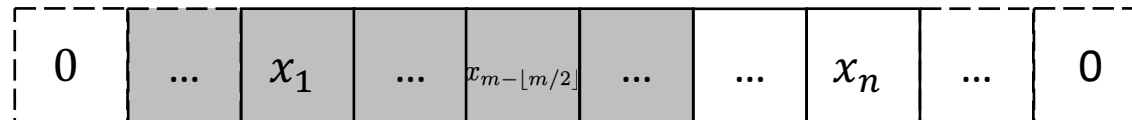
$$y^\top = (h * x)^\top$$

$$= \begin{bmatrix} h_1 & \cdots & h_{\lfloor m/2 \rfloor + 1} & \cdots & h_m \end{bmatrix} \begin{bmatrix} \text{padding: } \lfloor m/2 \rfloor \left\{ \begin{array}{ccccccc} 0 & 0 & \cdots & x_1 & \cdots & x_{n-\lfloor m/2 \rfloor - 1} & x_{n-\lfloor m/2 \rfloor} \\ \vdots & \vdots & \cdots & \vdots & \cdots & \vdots & \vdots \\ 0 & x_1 & \cdots & x_{\lfloor m/2 \rfloor} & \cdots & x_{n-2} & x_{n-1} \\ x_1 & x_2 & \cdots & x_{\lfloor m/2 \rfloor + 1} & \cdots & x_{n-1} & x_n \\ x_2 & x_3 & \cdots & x_{\lfloor m/2 \rfloor + 2} & \cdots & x_n & 0 \\ \vdots & \vdots & \cdots & \vdots & \cdots & \vdots & \vdots \\ x_{m-\lfloor m/2 \rfloor} & x_{m-\lfloor m/2 \rfloor + 1} & \cdots & x_m & \cdots & 0 & 0 \end{array} \right. \end{bmatrix}$$

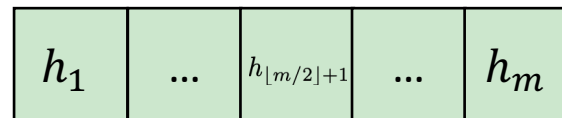
Data => Toeplitz matrix (diagonal-constant)

Can be dense when $n \gg m$.

Input x



Filter h



Matrix Multiplication View II

1D convolution as matrix multiplication

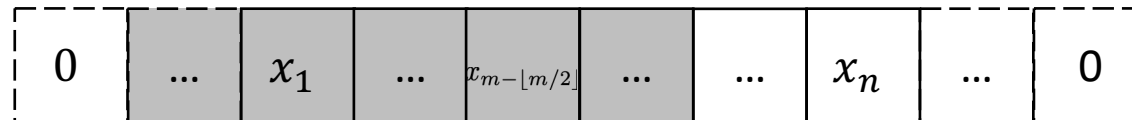
$$y^\top = (h * x)^\top$$

$$= \begin{bmatrix} h_1 & \cdots & h_{\lfloor m/2 \rfloor + 1} & \cdots & h_m \end{bmatrix} \begin{bmatrix} \text{padding: } \lfloor m/2 \rfloor \left\{ \begin{array}{ccccccc} 0 & 0 & \cdots & x_1 & \cdots & x_{n-\lfloor m/2 \rfloor - 1} & x_{n-\lfloor m/2 \rfloor} \\ \vdots & \vdots & \cdots & \vdots & \cdots & \vdots & \vdots \\ 0 & x_1 & \cdots & x_{\lfloor m/2 \rfloor} & \cdots & x_{n-2} & x_{n-1} \\ x_1 & x_2 & \cdots & x_{\lfloor m/2 \rfloor + 1} & \cdots & x_{n-1} & x_n \\ x_2 & x_3 & \cdots & x_{\lfloor m/2 \rfloor + 2} & \cdots & x_n & 0 \\ \vdots & \vdots & \cdots & \vdots & \cdots & \vdots & \vdots \\ x_{m-\lfloor m/2 \rfloor} & x_{m-\lfloor m/2 \rfloor + 1} & \cdots & x_m & \cdots & 0 & 0 \end{array} \right. \end{bmatrix}$$

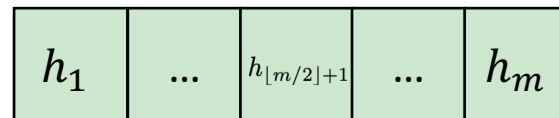
Data => Toeplitz matrix (diagonal-constant)

Can be dense when $n \gg m$.

Input x



Filter h



GPUs often use implicit GEMM [4].

Matrix Multiplication View II

1D convolution as matrix multiplication

This view extends to 2D and higher dimensions.

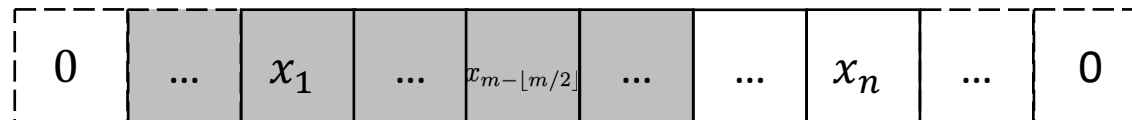
$$y^\top = (h * x)^\top$$

Data => Toeplitz matrix (diagonal-constant)

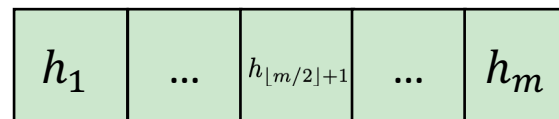
Can be dense when $n \gg m$.

$$= \begin{bmatrix} h_1 & \cdots & h_{\lfloor m/2 \rfloor + 1} & \cdots & h_m \end{bmatrix} \begin{bmatrix} \text{padding: } \lfloor m/2 \rfloor & \begin{bmatrix} 0 & 0 & \cdots & x_1 & \cdots & x_{n-\lfloor m/2 \rfloor - 1} & x_{n-\lfloor m/2 \rfloor} \\ \vdots & \vdots & \cdots & \vdots & \cdots & \vdots & \vdots \\ 0 & x_1 & \cdots & x_{\lfloor m/2 \rfloor} & \cdots & x_{n-2} & x_{n-1} \\ x_1 & x_2 & \cdots & x_{\lfloor m/2 \rfloor + 1} & \cdots & x_{n-1} & x_n \\ x_2 & x_3 & \cdots & x_{\lfloor m/2 \rfloor + 2} & \cdots & x_n & 0 \\ \vdots & \vdots & \cdots & \vdots & \cdots & \vdots & \vdots \\ x_{m-\lfloor m/2 \rfloor} & x_{m-\lfloor m/2 \rfloor + 1} & \cdots & x_m & \cdots & 0 & 0 \end{bmatrix} \end{bmatrix}$$

Input x



Filter h



GPUs often use implicit GEMM [4].

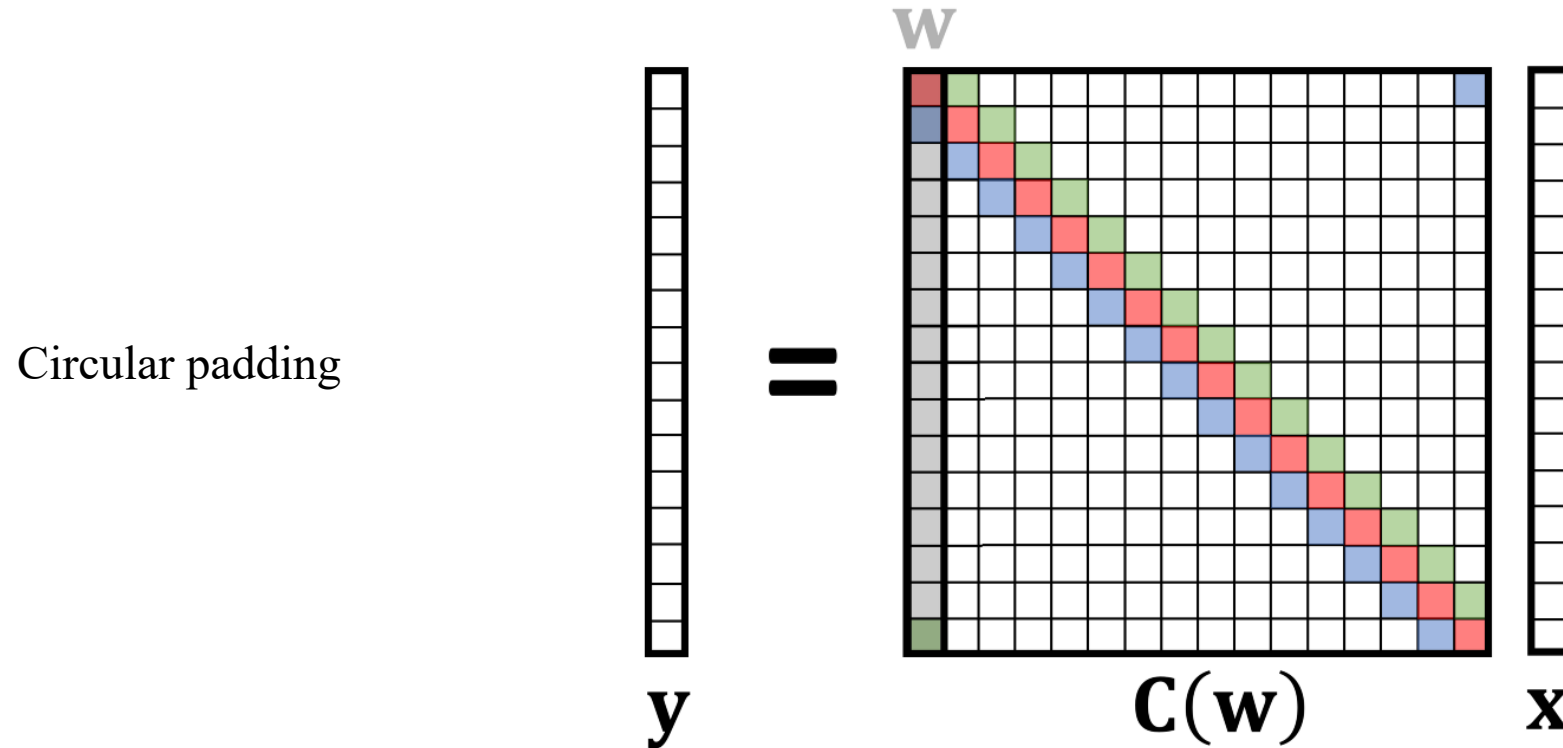
Outline

- Invariance & Equivariance
- Convolution
 - 1D Convolution
 - Matrix Multiplication Views
 - **Translation Equivariance**
 - 2D Convolution
- Convolution Variants
 - Transposed Convolution
 - Dilated Convolution
 - Grouped Convolution
 - Separable Convolution
- Pooling
- Example Architectures

1D (Discrete) Convolutions

Matrix multiplication view (Filter $\Rightarrow$ Toeplitz matrix) of 1D convolution:

A circulant matrix is a square Toeplitz matrix.



Translation/Shift Operator

$$\mathbf{y} = \mathbf{S} \mathbf{x}$$

Diagram illustrating the forward shift operation $\mathbf{y} = \mathbf{S} \mathbf{x}$. The input vector $\mathbf{x}$ (yellow, blue, green, white) is shifted down by one position to produce the output vector $\mathbf{y}$ (white, yellow, blue, green). The shift matrix $\mathbf{S}$ is a 4x4 matrix with black cells at (1,4), (2,1), (3,2), and (4,3).

$$\mathbf{y} = \mathbf{S}^T \mathbf{x}$$

Diagram illustrating the backward shift operation $\mathbf{y} = \mathbf{S}^T \mathbf{x}$. The input vector $\mathbf{x}$ (yellow, blue, green, white) is shifted up by one position to produce the output vector $\mathbf{y}$ (blue, green, white, yellow). The shift matrix $\mathbf{S}^T$ is a 4x4 matrix with black cells at (4,1), (3,4), (2,3), and (1,2).

$$\mathbf{S} \mathbf{S}^T = \mathbf{S}^T \mathbf{S} = \mathbf{I}$$

Diagram illustrating the property that the shift matrix $\mathbf{S}$ and its transpose $\mathbf{S}^T$ are inverses: $\mathbf{S} \mathbf{S}^T = \mathbf{S}^T \mathbf{S} = \mathbf{I}$, where $\mathbf{I}$ is the 4x4 identity matrix.

Translation/Shift Operator

A shift is also a circulant matrix!

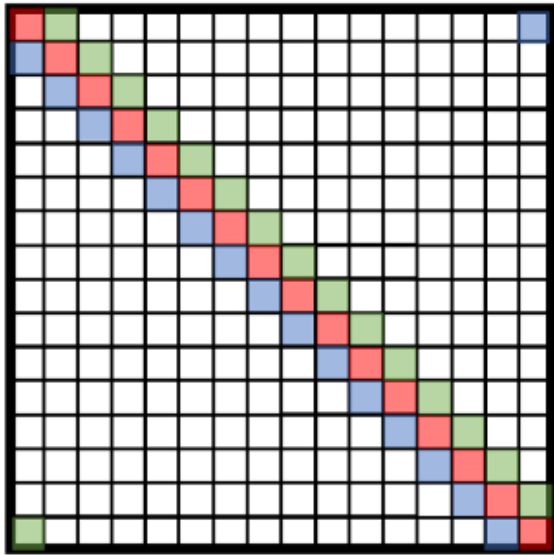
$$\mathbf{y} = \mathbf{S} \mathbf{x}$$

$$\mathbf{y} = \mathbf{S}^T \mathbf{x}$$

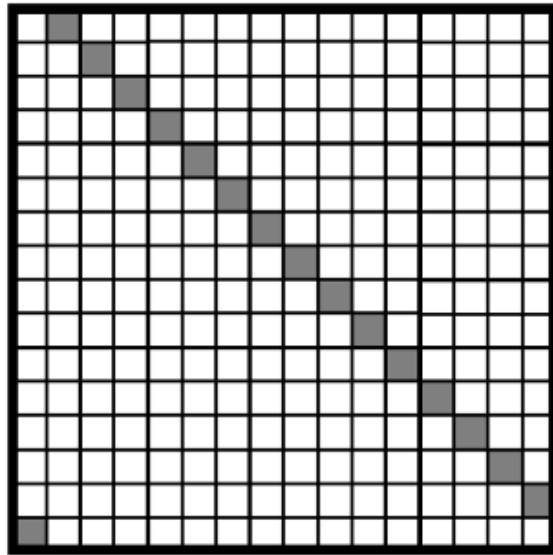
$$\mathbf{S} \mathbf{S}^T = \mathbf{S}^T \mathbf{S} = \mathbf{I}$$

Translation/Shift Equivariance

Matrix multiplication is non-commutative. But not for circulant matrices!



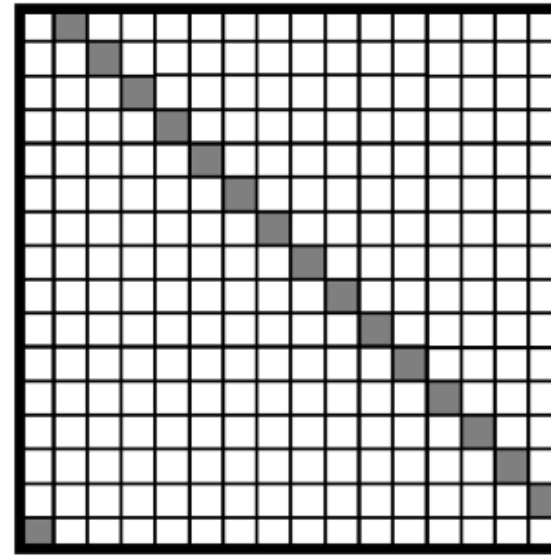
$\mathbf{C}(\mathbf{w})$



$\mathbf{S}^T$

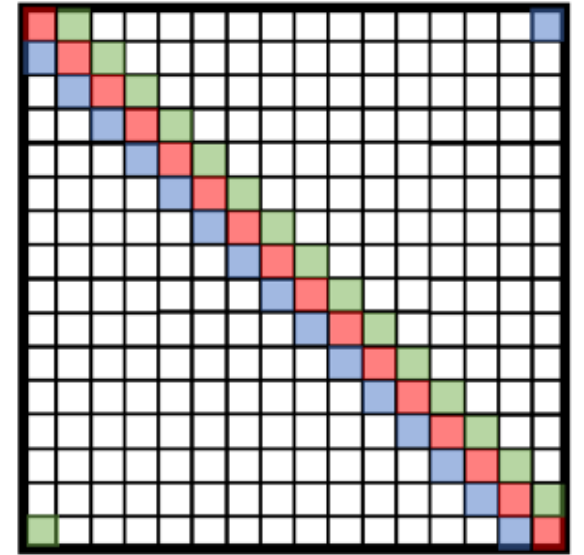
shift operator

$=$



$\mathbf{S}^T$

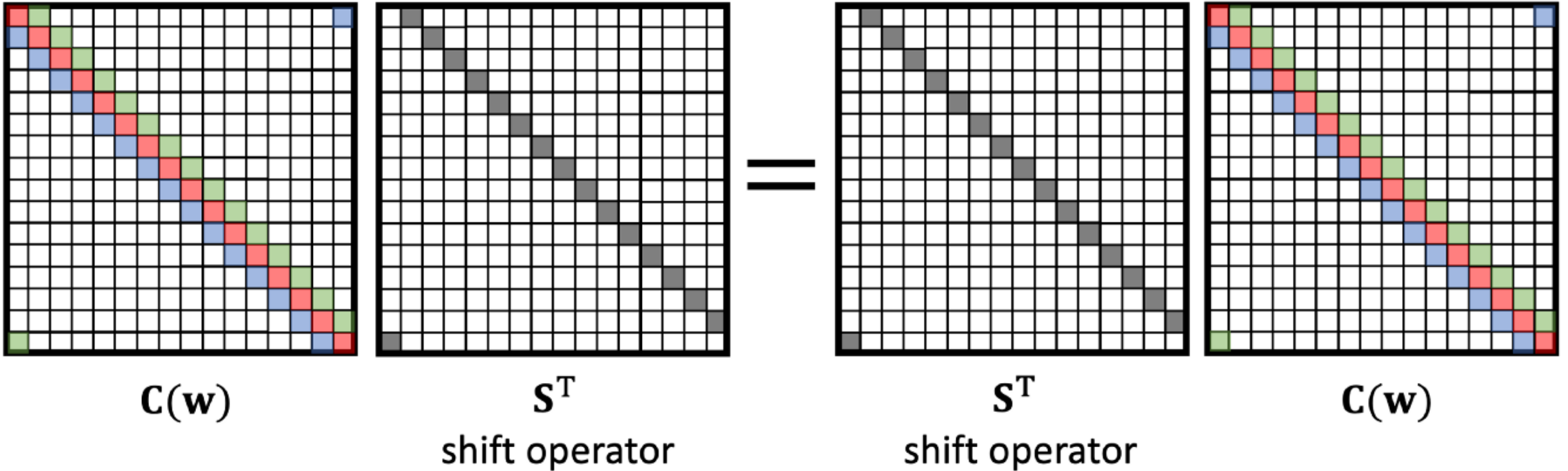
shift operator



$\mathbf{C}(\mathbf{w})$

Translation/Shift Equivariance

Matrix multiplication is non-commutative. But not for circulant matrices!



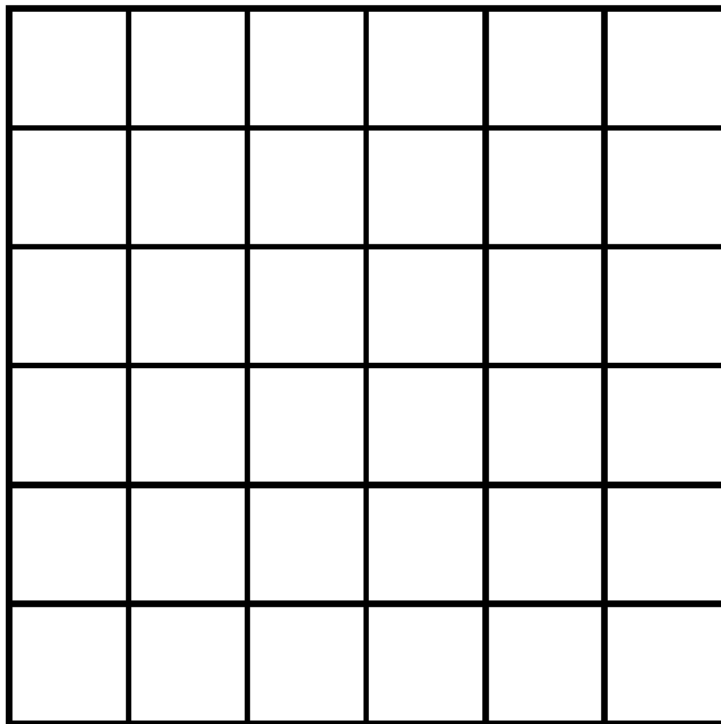
Circular convolution with stride 1 is equivariant to cyclic shifts.
Fixed zero-padding boundaries and stride > 1 require extra care.

Outline

- Invariance & Equivariance
- Convolution
 - 1D Convolution
 - Matrix Multiplication Views
 - Translation Equivariance
 - **2D Convolution**
- Convolution Variants
 - Transposed Convolution
 - Dilated Convolution
 - Grouped Convolution
 - Separable Convolution
- Pooling
- Example Architectures

2D (Discrete) Convolution

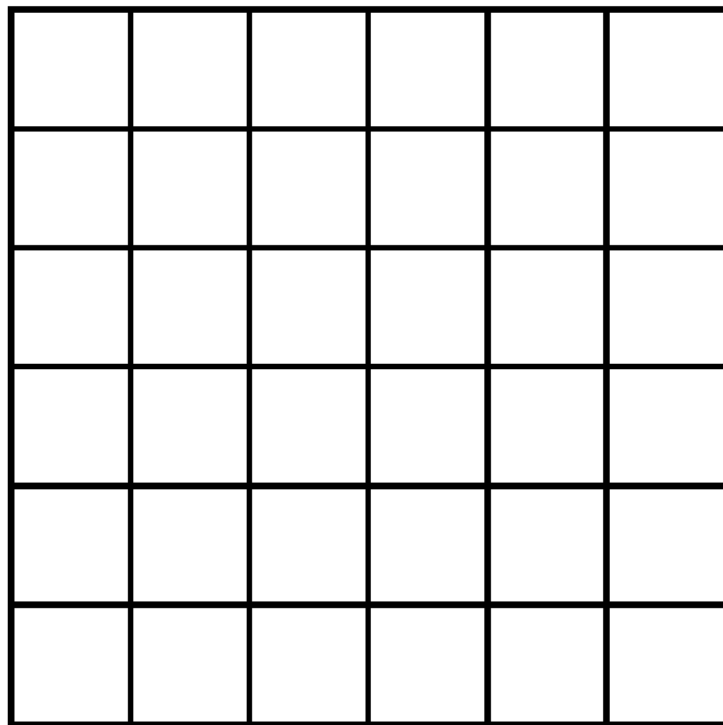
Let us see what convolution is in 2D



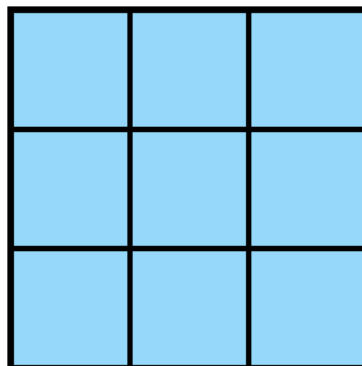
Input $\mathbf{X}$

2D (Discrete) Convolution

Let us see what convolution is in 2D



Input $\mathbf{X}$

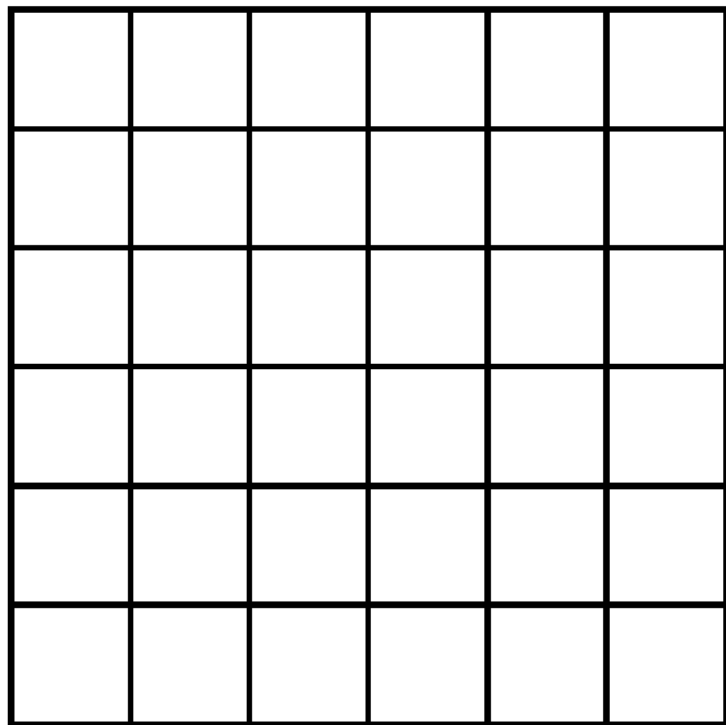


Convolutional Filter

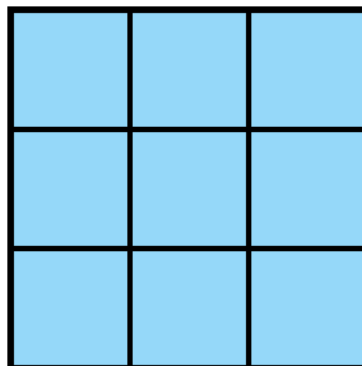
$$W \in \mathbb{R}^{K \times K}$$

2D (Discrete) Convolution

Let us see what convolution is in 2D

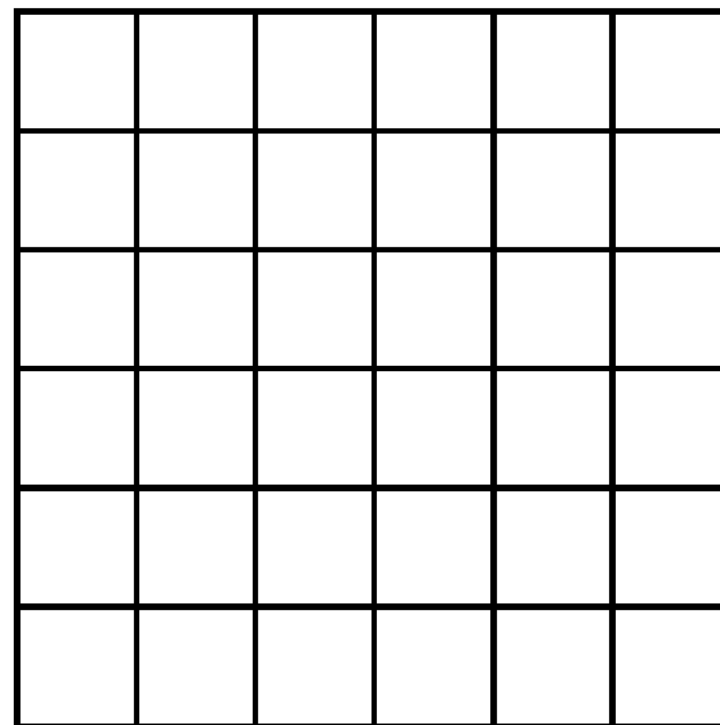


Input $\mathbf{X}$



Convolutional Filter

$$W \in \mathbb{R}^{K \times K}$$

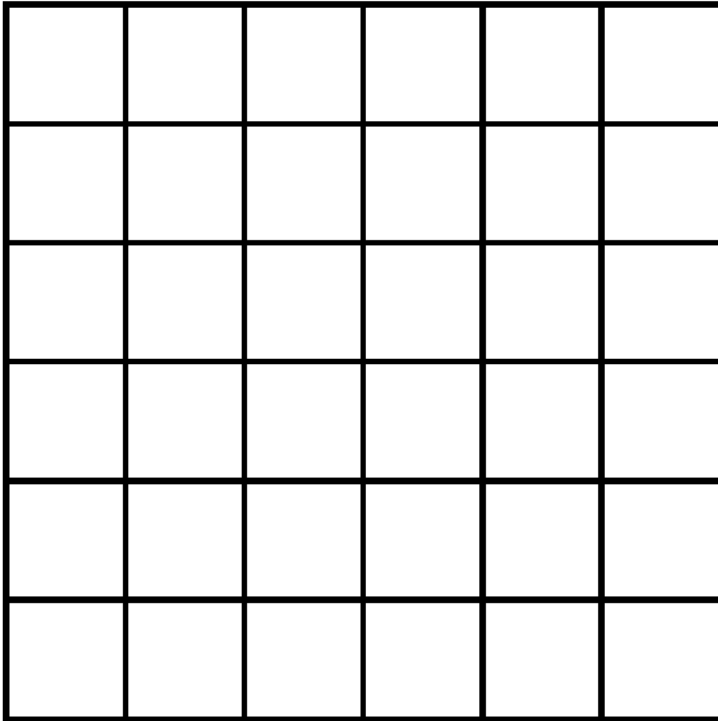


Sliding Window

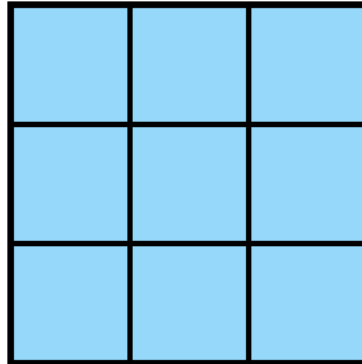
2D (Discrete) Convolution

Let us see what convolution is in 2D

$$\mathbf{y}_{i,j} = \sum_{m=1}^K \sum_{n=1}^K W_{m,n} \mathbf{x}_{i+m-\lceil K/2 \rceil, j+n-\lceil K/2 \rceil}$$

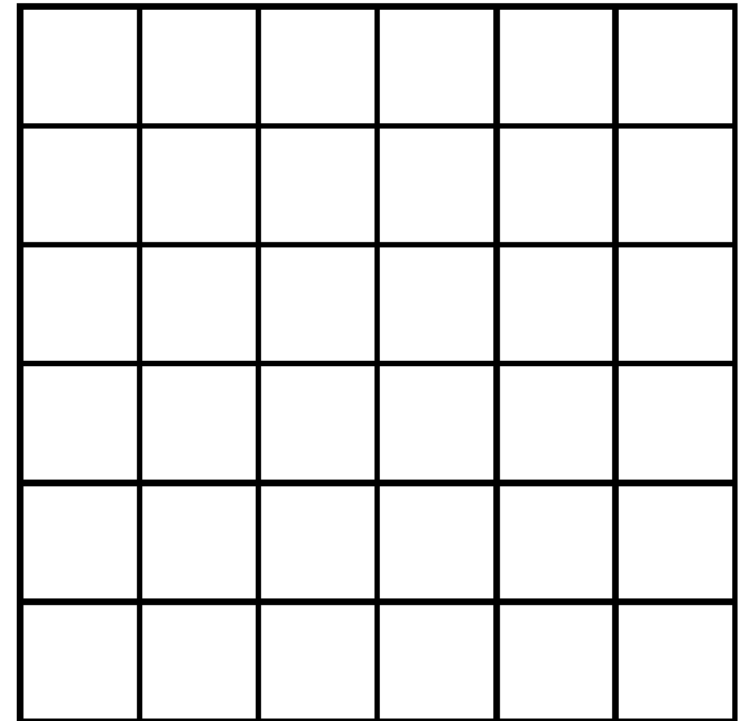


Input $\mathbf{x}$



Convolutional Filter

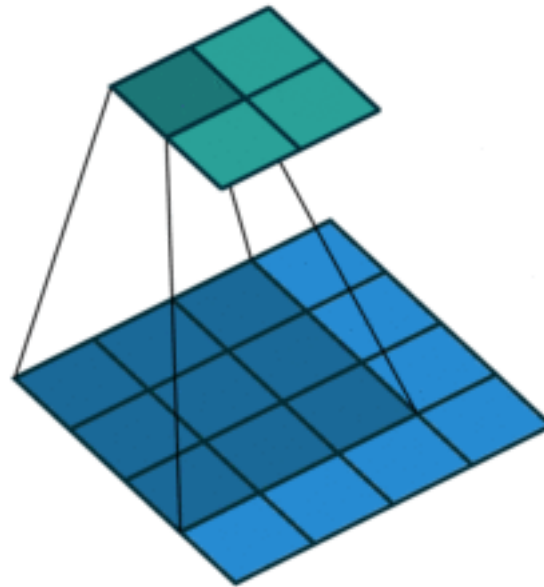
$$W \in \mathbb{R}^{K \times K}$$



Output $\mathbf{y}$

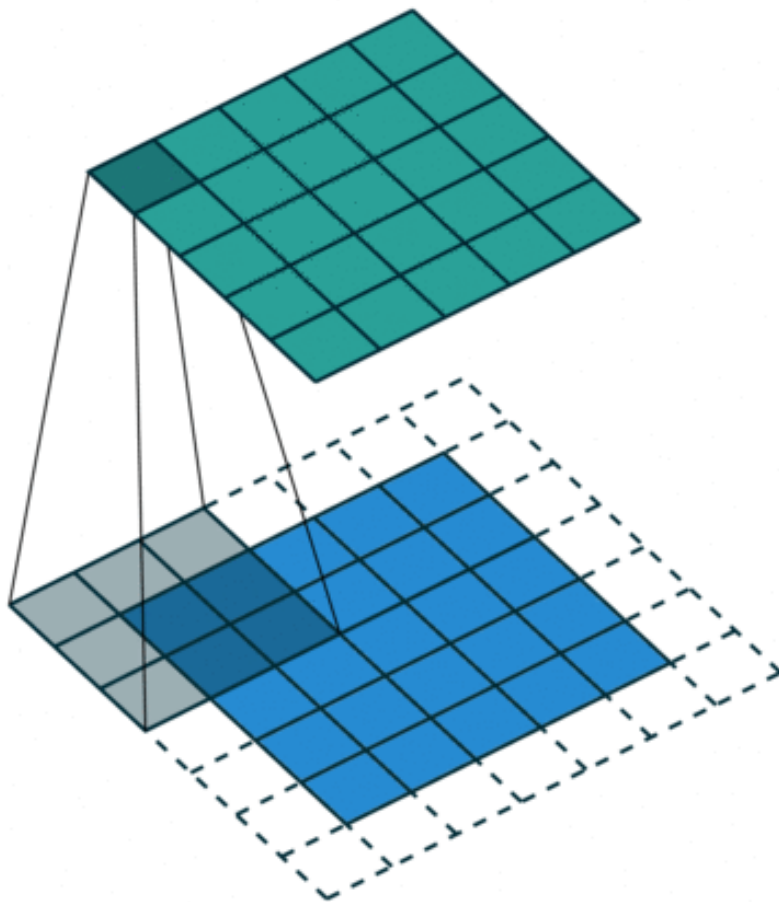
2D (Discrete) Convolution

2D Convolution with Stride = 1



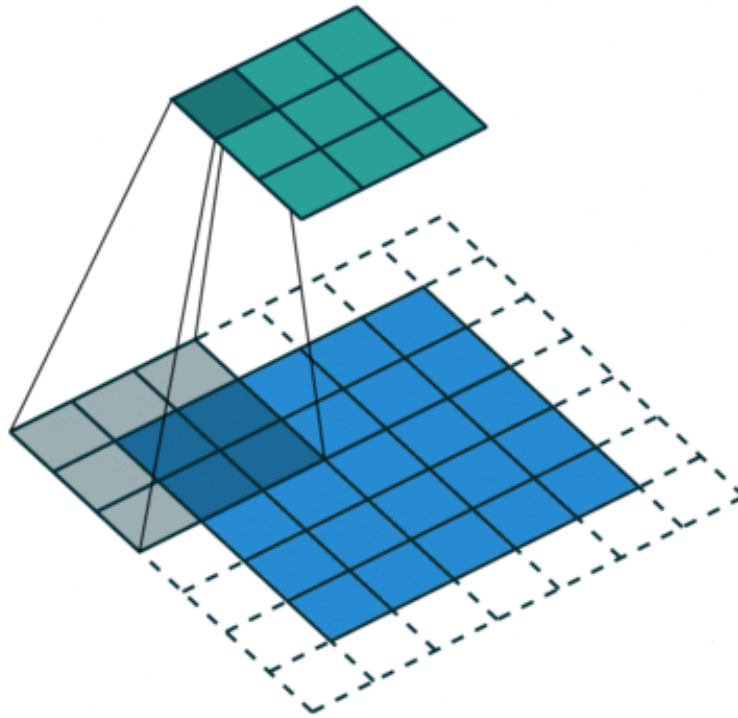
2D (Discrete) Convolution

2D Convolution with Stride = 1, Half Padding



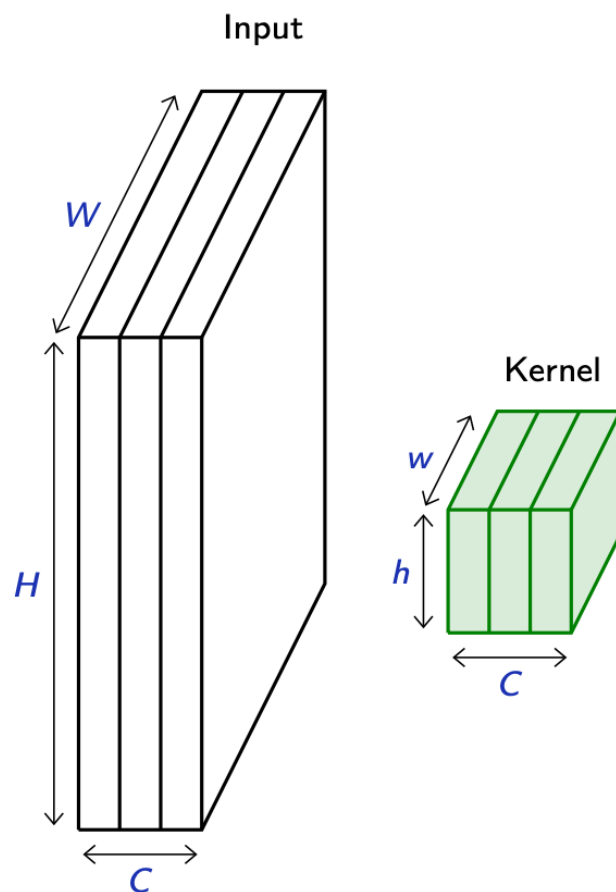
2D (Discrete) Convolution

2D Convolution with Stride = 2, Half Padding



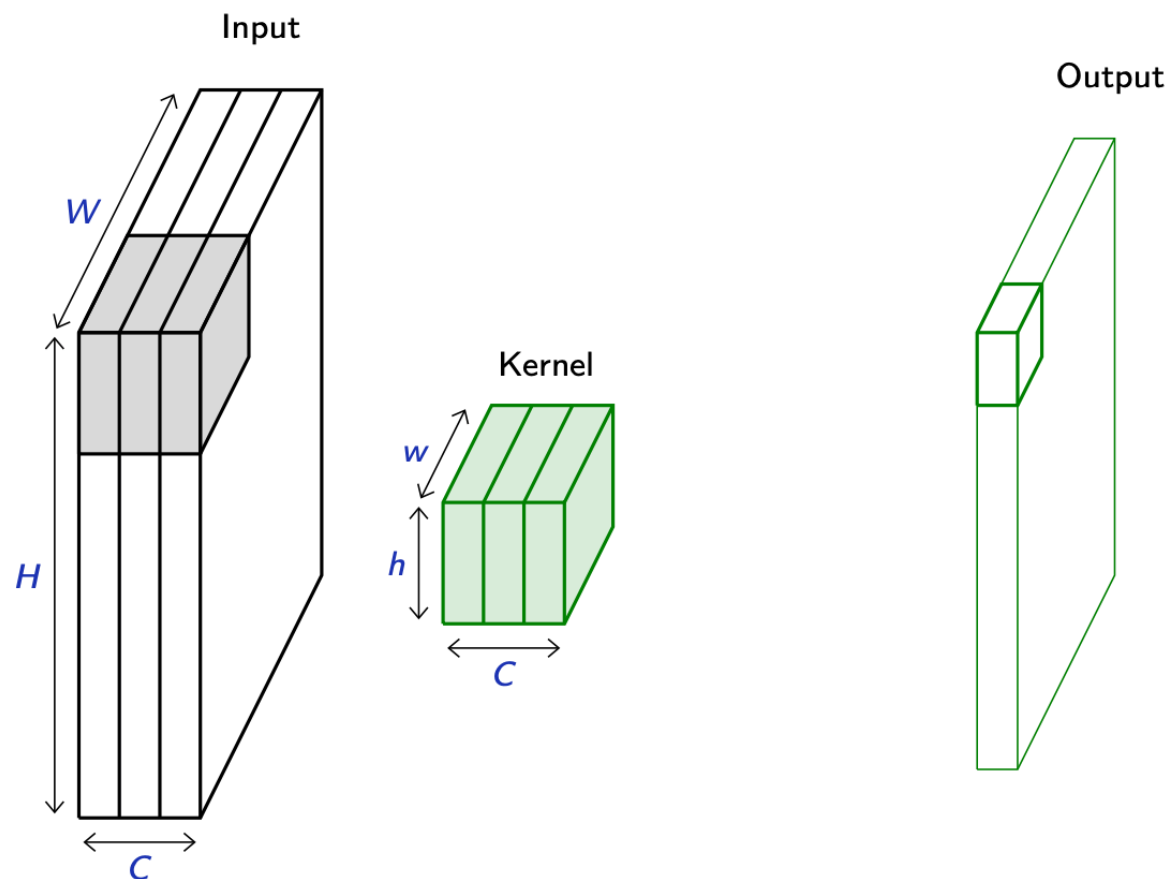
2D (Discrete) Convolution

2D Convolution with multiple input channels



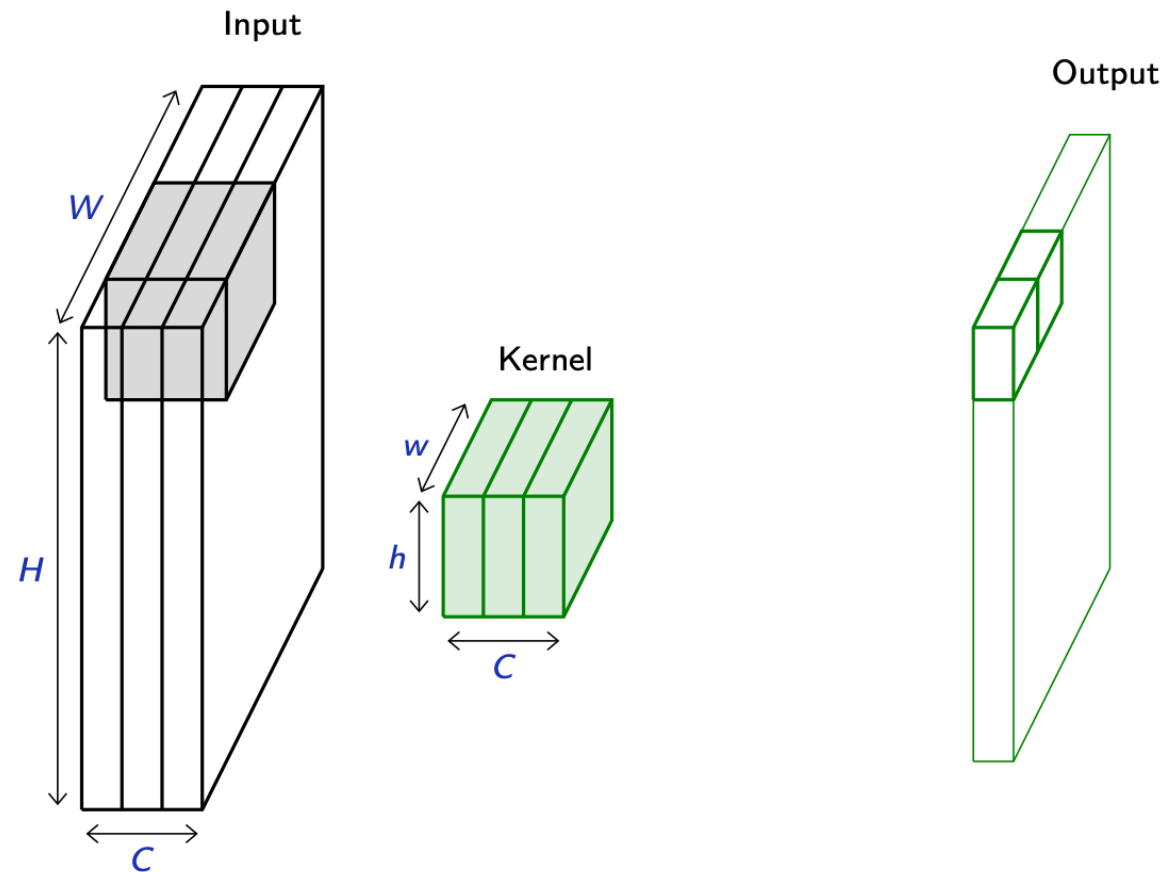
2D (Discrete) Convolution

2D Convolution with multiple input channels



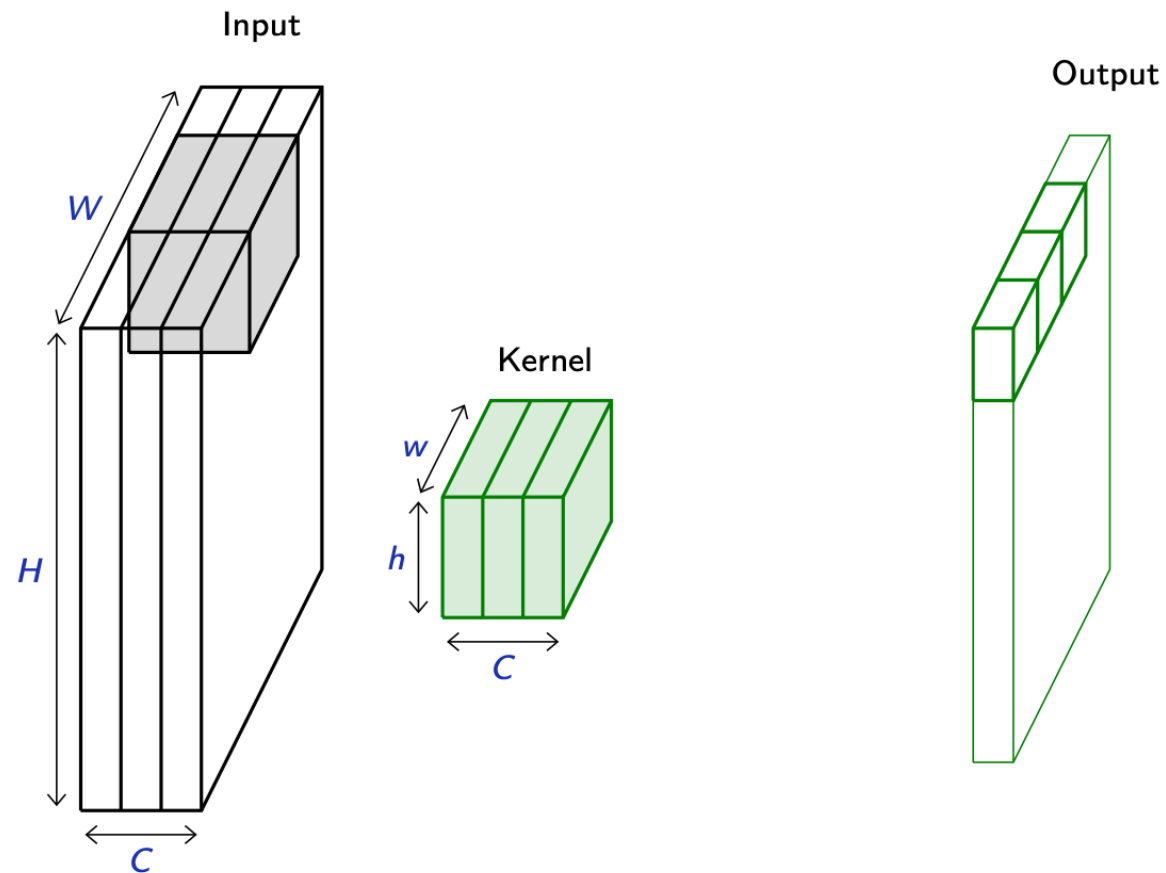
2D (Discrete) Convolution

2D Convolution with multiple input channels



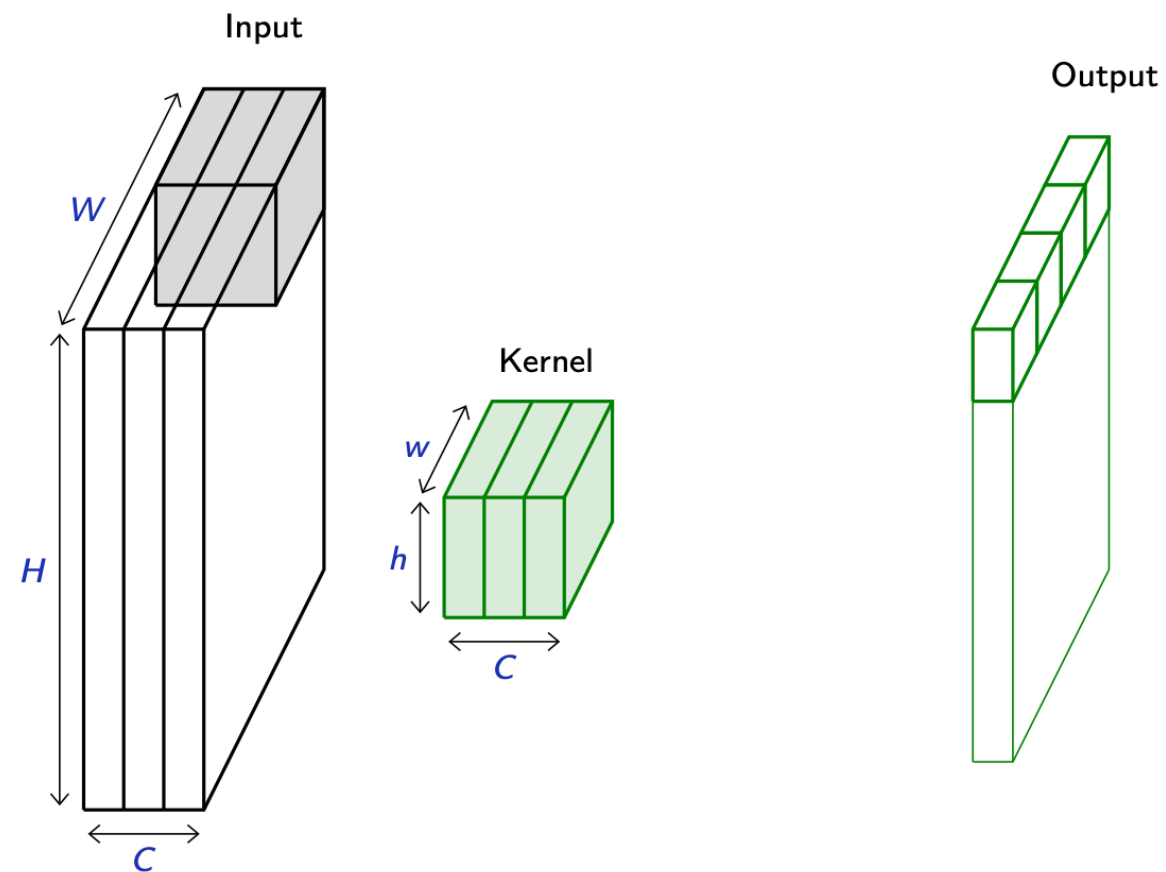
2D (Discrete) Convolution

2D Convolution with multiple input channels



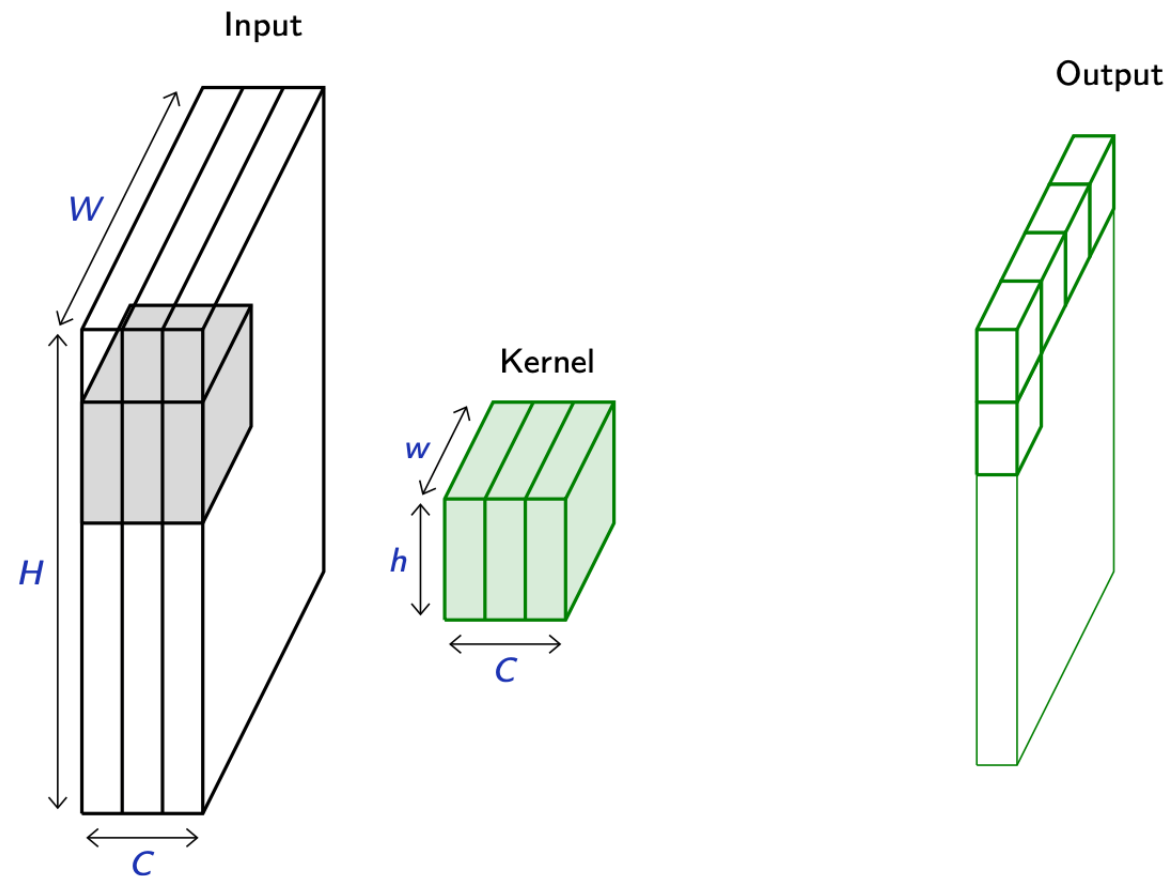
2D (Discrete) Convolution

2D Convolution with multiple input channels



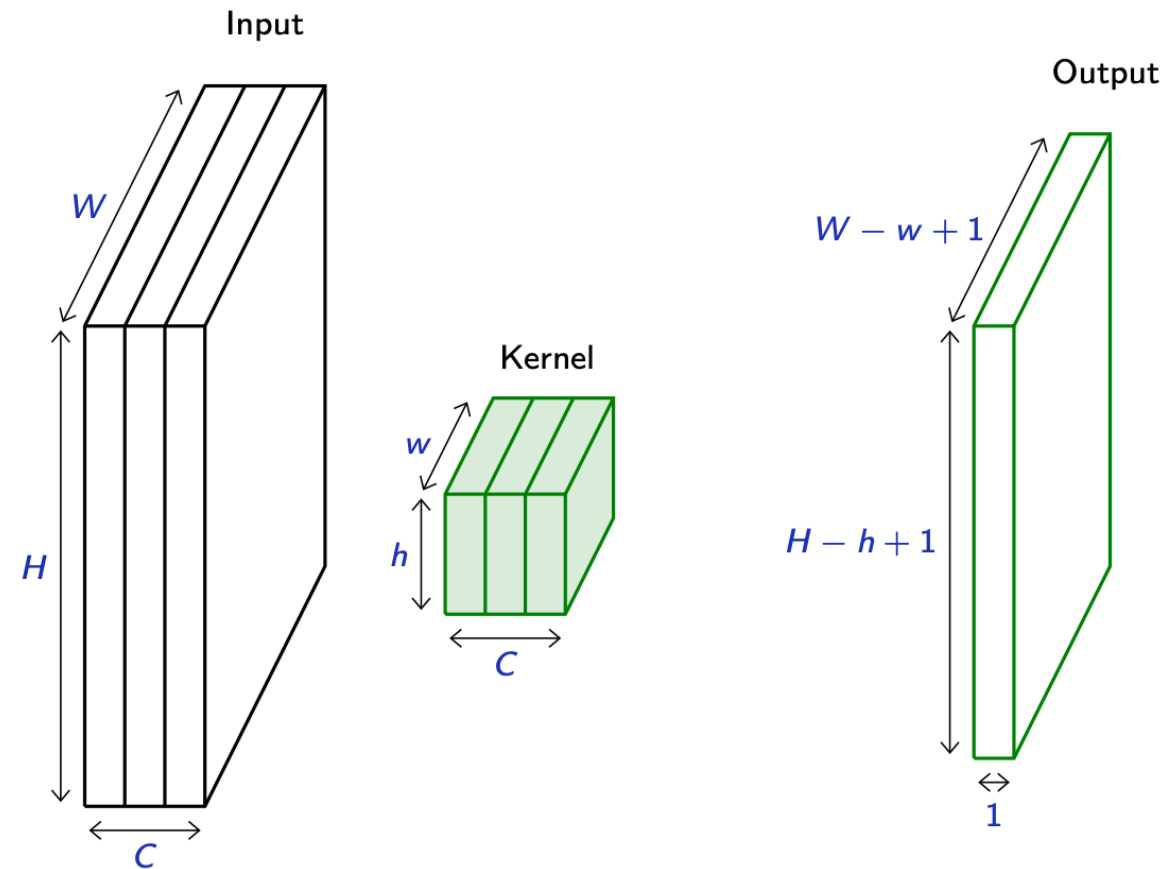
2D (Discrete) Convolution

2D Convolution with multiple input channels



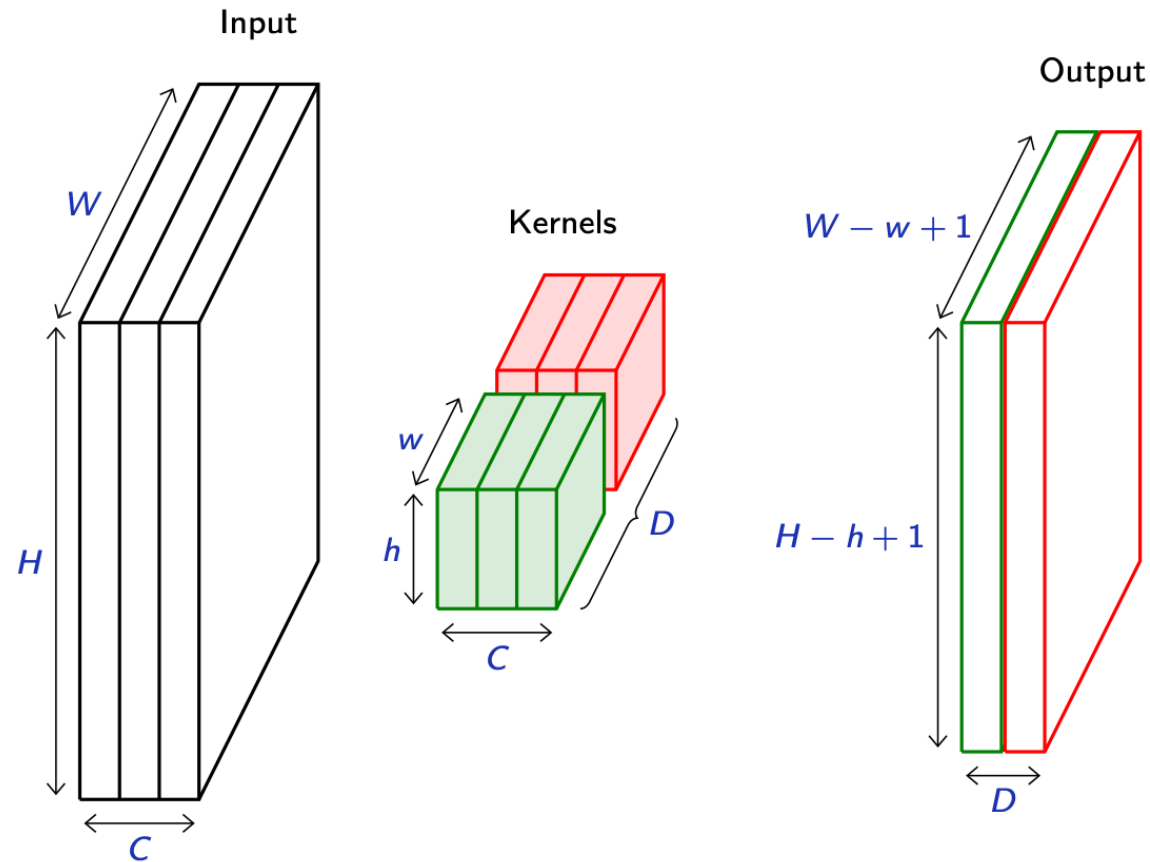
2D (Discrete) Convolution

2D Convolution with multiple input channels



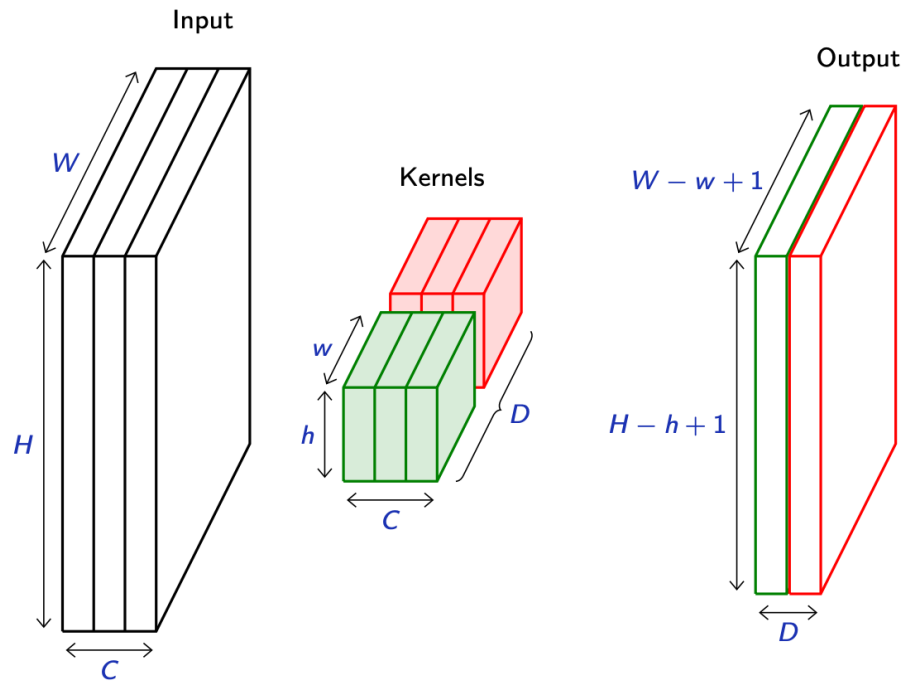
2D (Discrete) Convolution

2D Convolution with multiple input channels and multiple filters



2D (Discrete) Convolution

Tensor shapes and shared parameters [5]



Input:

$$\mathbf{x} \in \mathbb{R}^{C \times H \times W}$$

Kernel bank:

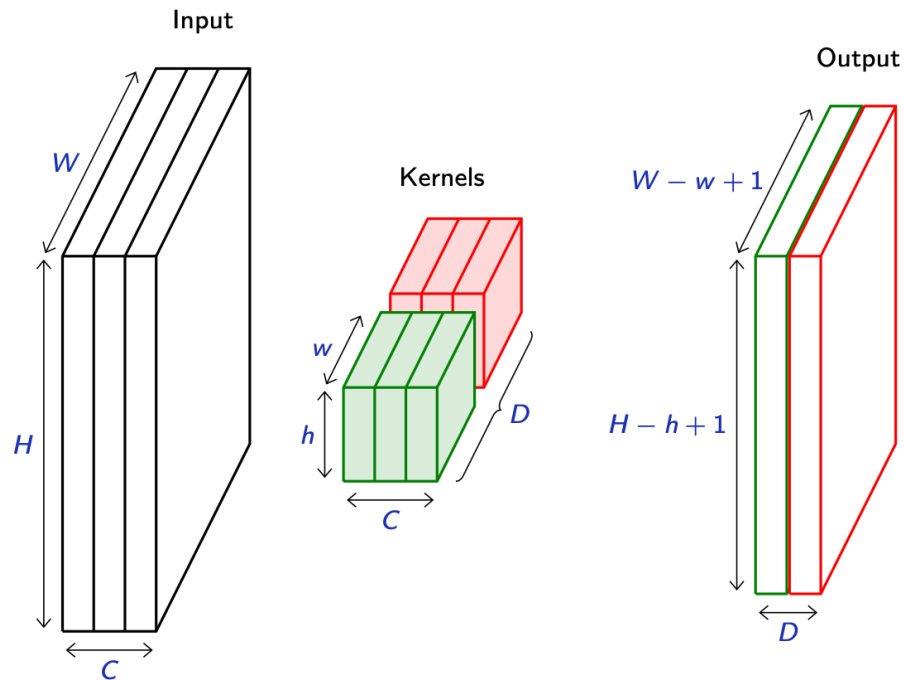
D filters, each $C \times h \times w$

Output:

$$\mathbf{y} \in \mathbb{R}^{D \times H_{\text{out}} \times W_{\text{out}}}$$

2D (Discrete) Convolution

Tensor shapes and shared parameters [5]



Input:

$$\mathbf{x} \in \mathbb{R}^{C \times H \times W}$$

Kernel bank:

D filters, each $C \times h \times w$

Output:

$$\mathbf{y} \in \mathbb{R}^{D \times H_{\text{out}} \times W_{\text{out}}}$$

Parameters, including one bias per filter:

$$P = D(Chw + 1)$$

Example:

$$C = 3, D = 16, h = w = 3 \implies P = 448$$

2D (Discrete) Convolution

A quick check: output shape and parameter count

For dilation 1, apply the 1D output-size formula along height and width [5].

$$H_{\text{out}} = \left\lfloor \frac{H+2p-h}{s} \right\rfloor + 1$$

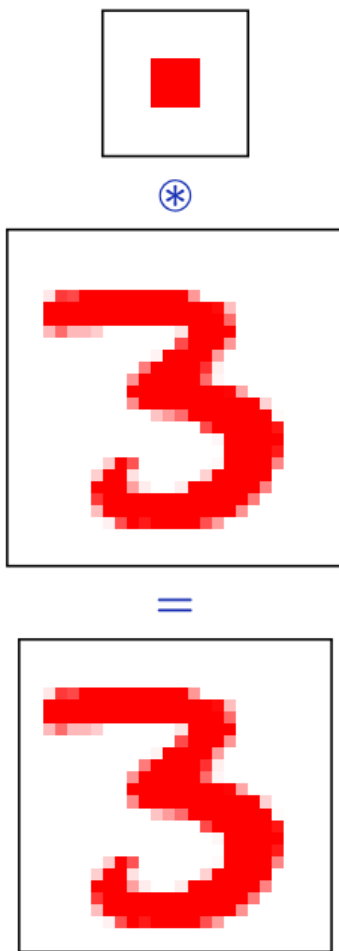
Apply the same formula to the width; output channels equal the number of filters.

Input: 3 channels, 32 x 32 pixels. Use 16 filters of size 3 x 3, stride 2, and padding 1. What are the output shape and parameter count?

Count the channels separately from the spatial output size.

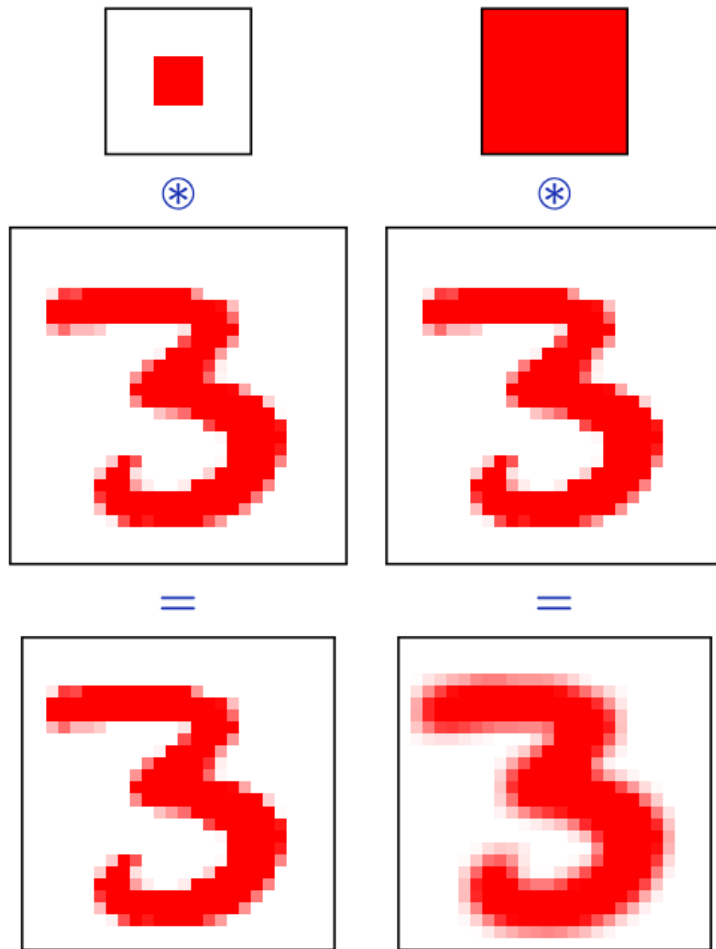
2D (Discrete) Convolution

Let us see the effect of 2D convolution:



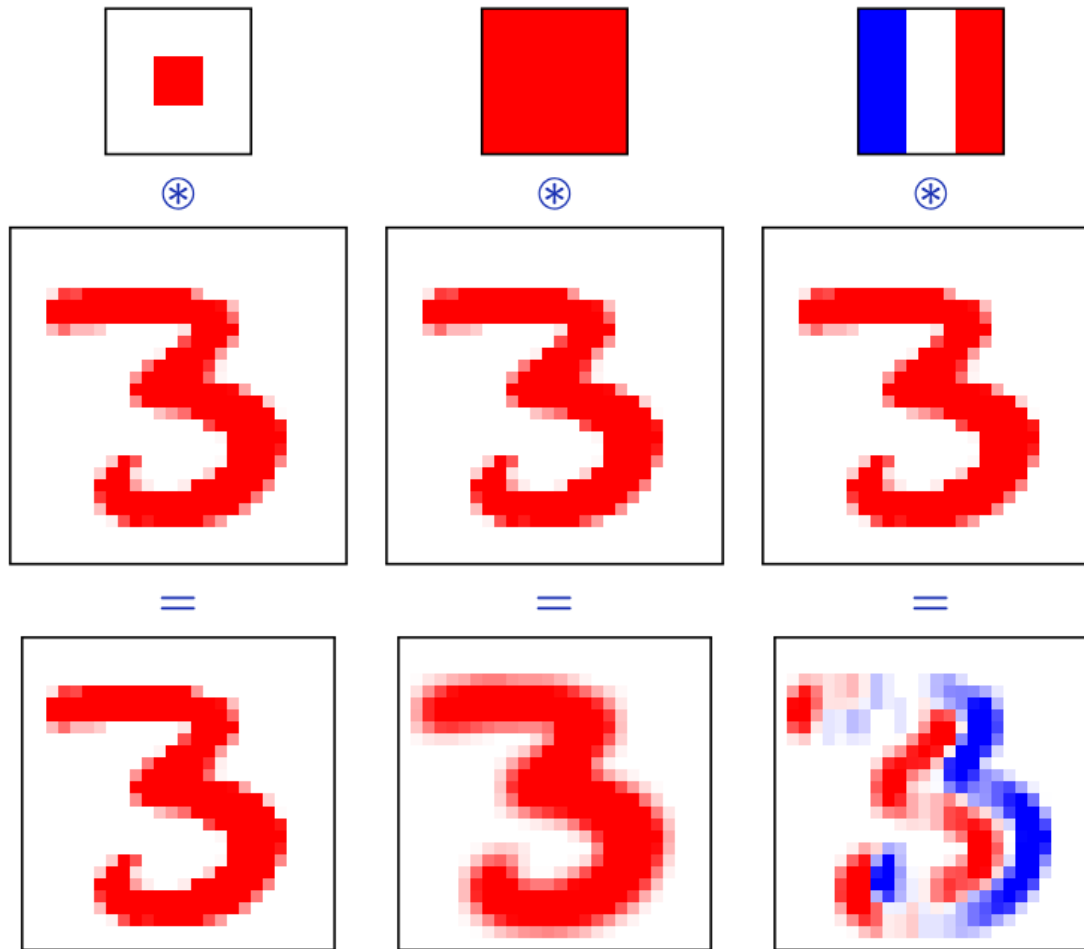
2D (Discrete) Convolution

Let us see the effect of 2D convolution:



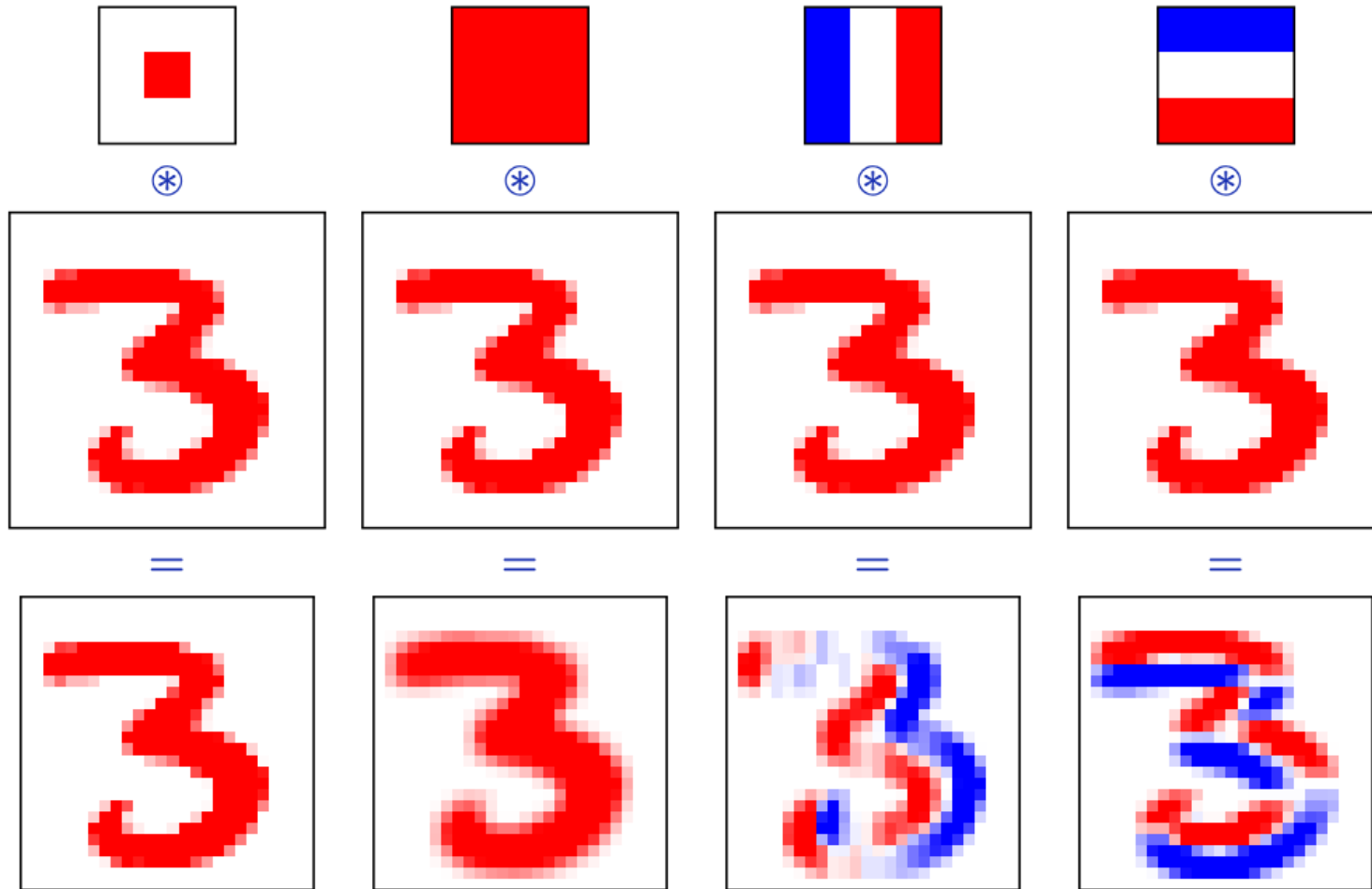
2D (Discrete) Convolution

Let us see the effect of 2D convolution:



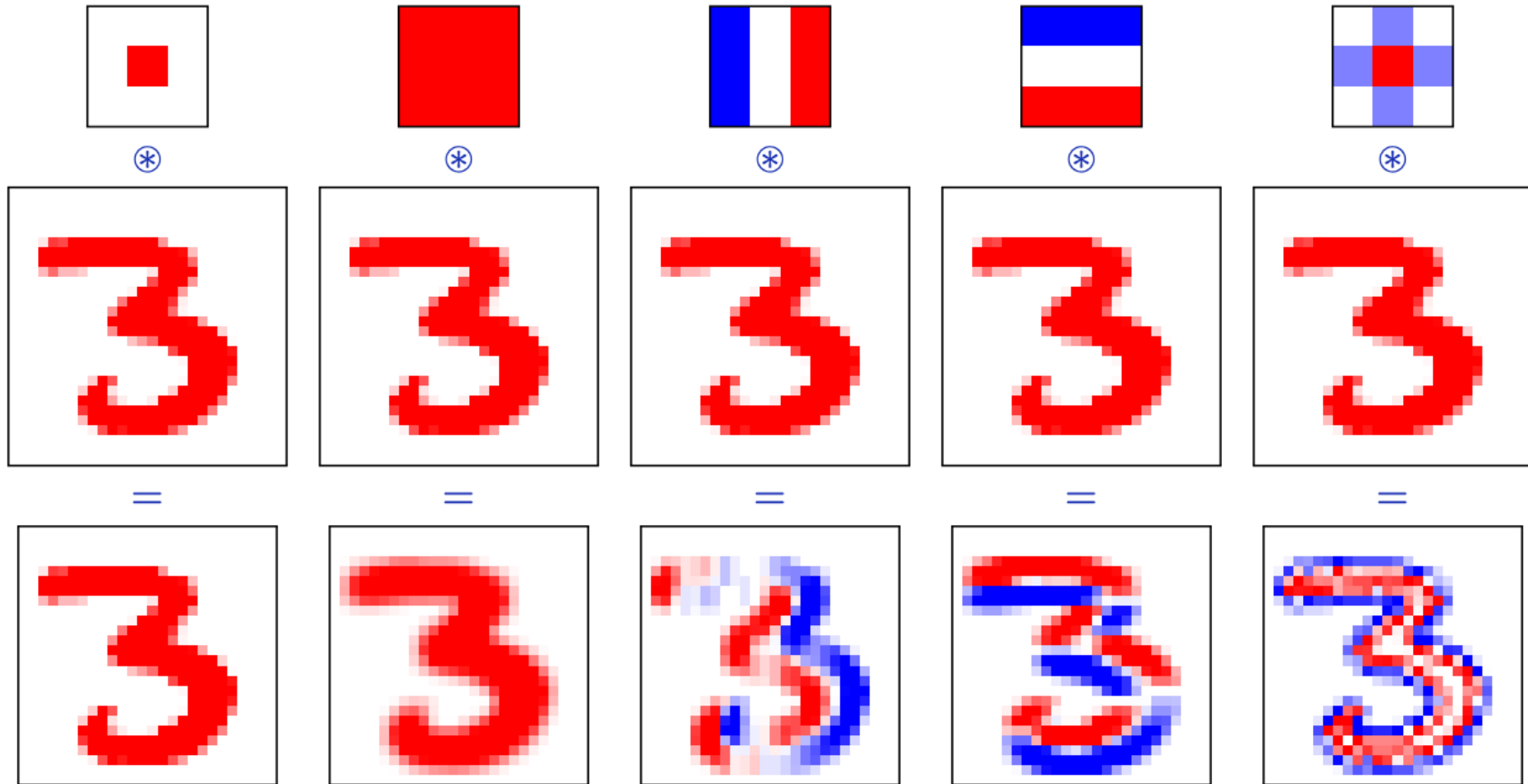
2D (Discrete) Convolution

Let us see the effect of 2D convolution:



2D (Discrete) Convolution

Let us see the effect of 2D convolution:



References

- [1] <https://towardsdatascience.com/deriving-convolution-from-first-principles-4ff124888028>
- [2] https://github.com/vdumoulin/conv_arithmetic/blob/master/README.md
- [3] <https://fleuret.org/dlc/materials/dlc-slides-4-4-convolutions.pdf>
- [4] <https://docs.nvidia.com/deeplearning/performance/dl-performance-convolutional/index.html>
- [5] <https://docs.pytorch.org/docs/stable/generated/torch.nn.Conv2d.html>

Questions?