CNN and Pytorch

Thomas Gerald

March 30, 2024

Laboratoire Interdisciplinaire des Sciences du Numérique – LISN, CNRS

The lecture

A short introduction to CNN and Pytorch

The lecture

A short introduction to CNN and Pytorch

1. The Convolutional neural networks: A short presentation

The lecture

A short introduction to CNN and Pytorch

1. The Convolutional neural networks: A short presentation
2. Introduction to pytorch (using slides of Caio Corro)

Convolutional Neural Networks

Convolutional Neural Networks : Introduction

Convolutional Neural Networks (CNN)
A neural network specialised using convolution functions as module

Convolutional Neural Networks : Introduction

Convolutional Neural Networks (CNN)
A neural network specialised using convolution functions as module

- Repeatidely use filters on image or previously "filtered" features map (using many convolution layer) creating new features map

Convolutional Neural Networks : Introduction

Convolutional Neural Networks (CNN)
A neural network specialised using convolution functions as module

- Repeatidely use filters on image or previously "filtered" features map (using many convolution layer) creating new features map
- Select interrest part of the features map (pooling layers)

Convolutional Neural Networks : Introduction

Convolutional Neural Networks (CNN)
A neural network specialised using convolution functions as module

- Repeatidely use filters on image or previously "filtered" features map (using many convolution layer) creating new features map
- Select interrest part of the features map (pooling layers)
- Use a MLP (or fully connected) for classification based on the last features extracted from convolution

Convolutional Neural Networks : Introduction

Convolutional Neural Networks (CNN)
A neural network specialised using convolution functions as module

- Repeatidely use filters on image or previously "filtered" features map (using many convolution layer) creating new features map
- Select interrest part of the features map (pooling layers)
- Use a MLP (or fully connected) for classification based on the last features extracted from convolution

What are the use cases

- In image processing using 2D convolutions

Convolutional Neural Networks : Introduction

Convolutional Neural Networks (CNN)
A neural network specialised using convolution functions as module

- Repeatidely use filters on image or previously "filtered" features map (using many convolution layer) creating new features map
- Select interrest part of the features map (pooling layers)
- Use a MLP (or fully connected) for classification based on the last features extracted from convolution

What are the use cases

- In image processing using 2D convolutions
- In signal processing such as audio (1D convolutions)

Convolutional Neural Networks : Introduction

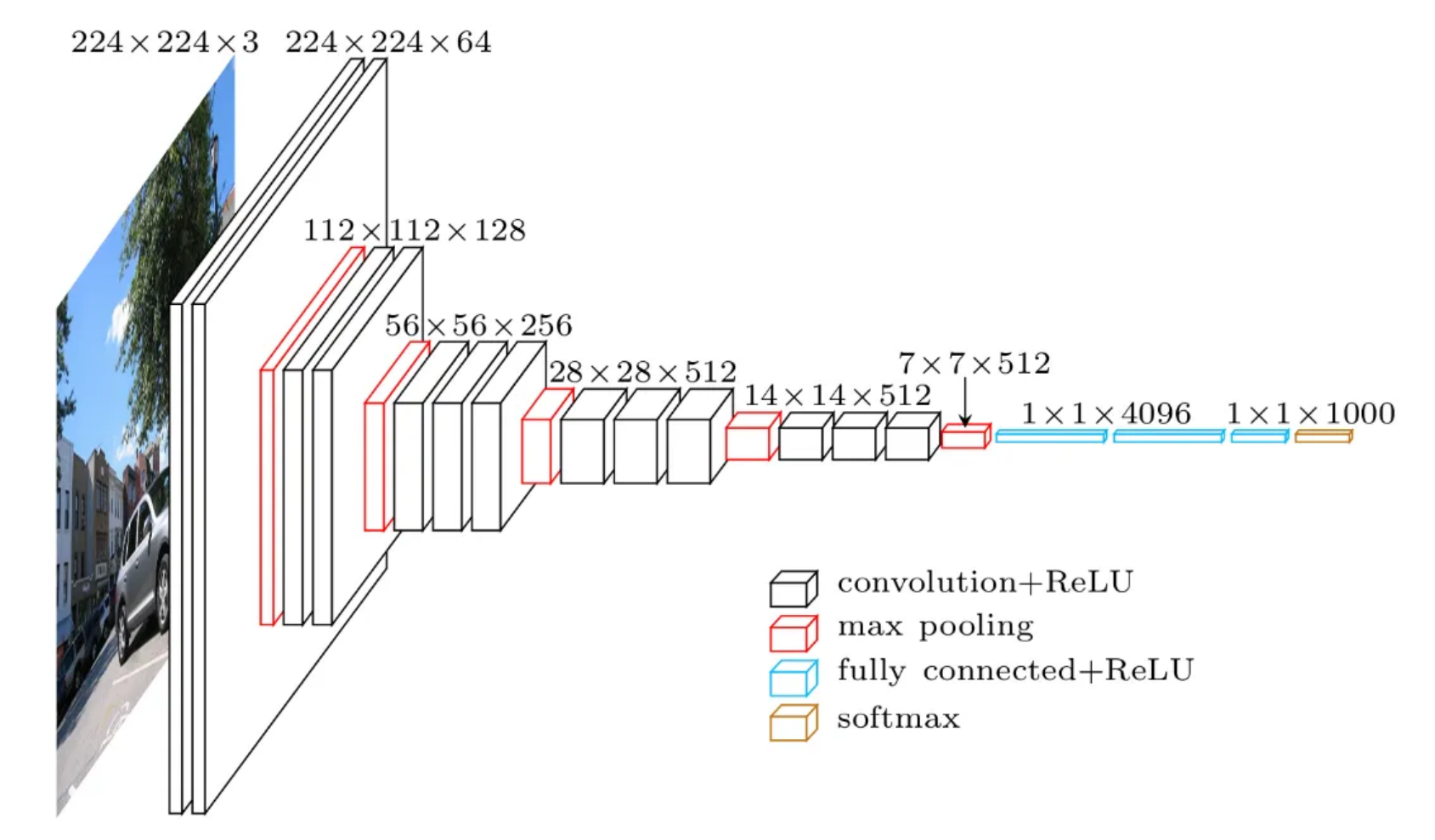
Convolutional Neural Networks (CNN)
A neural network specialised using convolution functions as module

- Repeatidely use filters on image or previously "filtered" features map (using many convolution layer) creating new features map
- Select interrest part of the features map (pooling layers)
- Use a MLP (or fully connected) for classification based on the last features extracted from convolution

What are the use cases

- In image processing using 2D convolutions
- In signal processing such as audio (1D convolutions)
- In text processing such as classification (1D convolutions)

Convolutional Neural Networks: Image classification

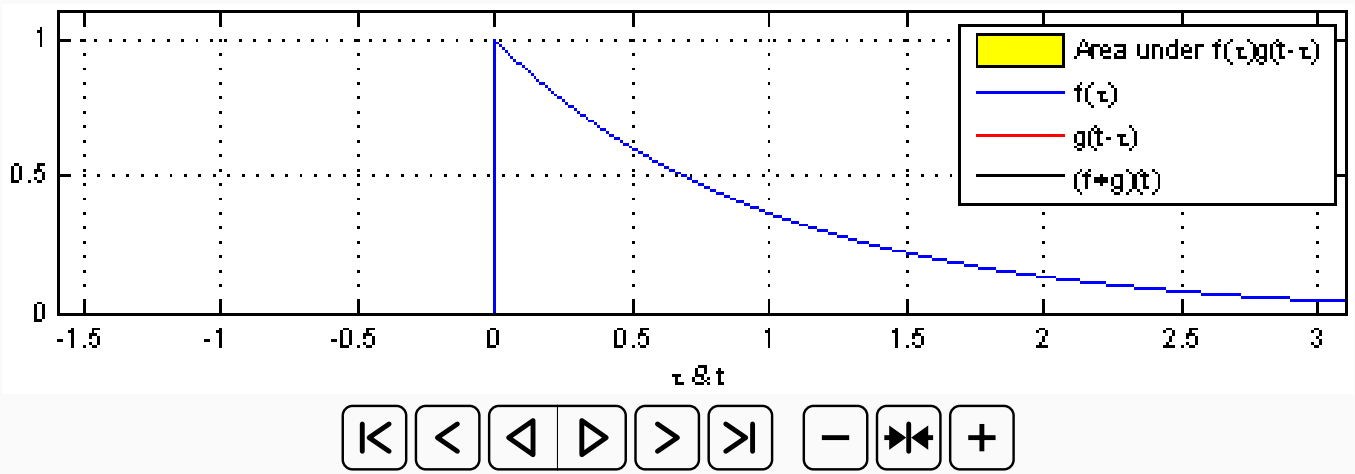


[0]VGG16 architecture for image classification image from https://miro.medium.com/v2/resize:fit:4800/format:webp/1*CrjJwSX9S7f759dK2EtGJQ.png

Convolution operation

Let f and g be two functions the convolution operation between f and g is generally written $f * g$ such that:

$$(f * g)(t) := \int_{-\infty}^{\infty} f(\tau)g(t-\tau)d\tau$$



Discrete Convolution operation

Working with computer data, we often consider discretized time steps. Thus, the time index t take integer values only.

Discrete Convolution operation

Working with computer data, we often consider discretized time steps. Thus, the time index t take integer values only.

Let consider $t \in \mathbb{Z}$ (we rename it m), we can define the discrete convolution as:

Discrete Convolution operation

Working with computer data, we often consider discretized time steps. Thus, the time index t take integer values only.

Let consider $t \in \mathbb{Z}$ (we rename it m), we can define the discrete convolution as:

$$(f * g)(m) := \sum_{n=-\infty}^{\infty} f(n)g(m-n)$$

Discrete Convolution operation

Working with computer data, we often consider discretized time steps. Thus, the time index t take integer values only.

Let consider $t \in \mathbb{Z}$ (we rename it m), we can define the discrete convolution as:

$$(f * g)(m) := \sum_{n=-\infty}^{\infty} f(n)g(m-n)$$

- The input function f is often a multidimensional array X

Discrete Convolution operation

Working with computer data, we often consider discretized time steps. Thus, the time index t take integer values only.

Let consider $t \in \mathbb{Z}$ (we rename it m), we can define the discrete convolution as:

$$(f * g)(m) := \sum_{n=-\infty}^{\infty} f(n)g(m-n)$$

- The input function f is often a multidimensional array X
- The kernel function g is often a multidimensional array K

Discrete Convolution operation

Working with computer data, we often consider discretized time steps. Thus, the time index t take integer values only.

Let consider $t \in \mathbb{Z}$ (we rename it m), we can define the discrete convolution as:

$$(f * g)(m) := \sum_{n=-\infty}^{\infty} f(n)g(m-n)$$

- The input function f is often a multidimensional array X
- The kernel function g is often a multidimensional array K

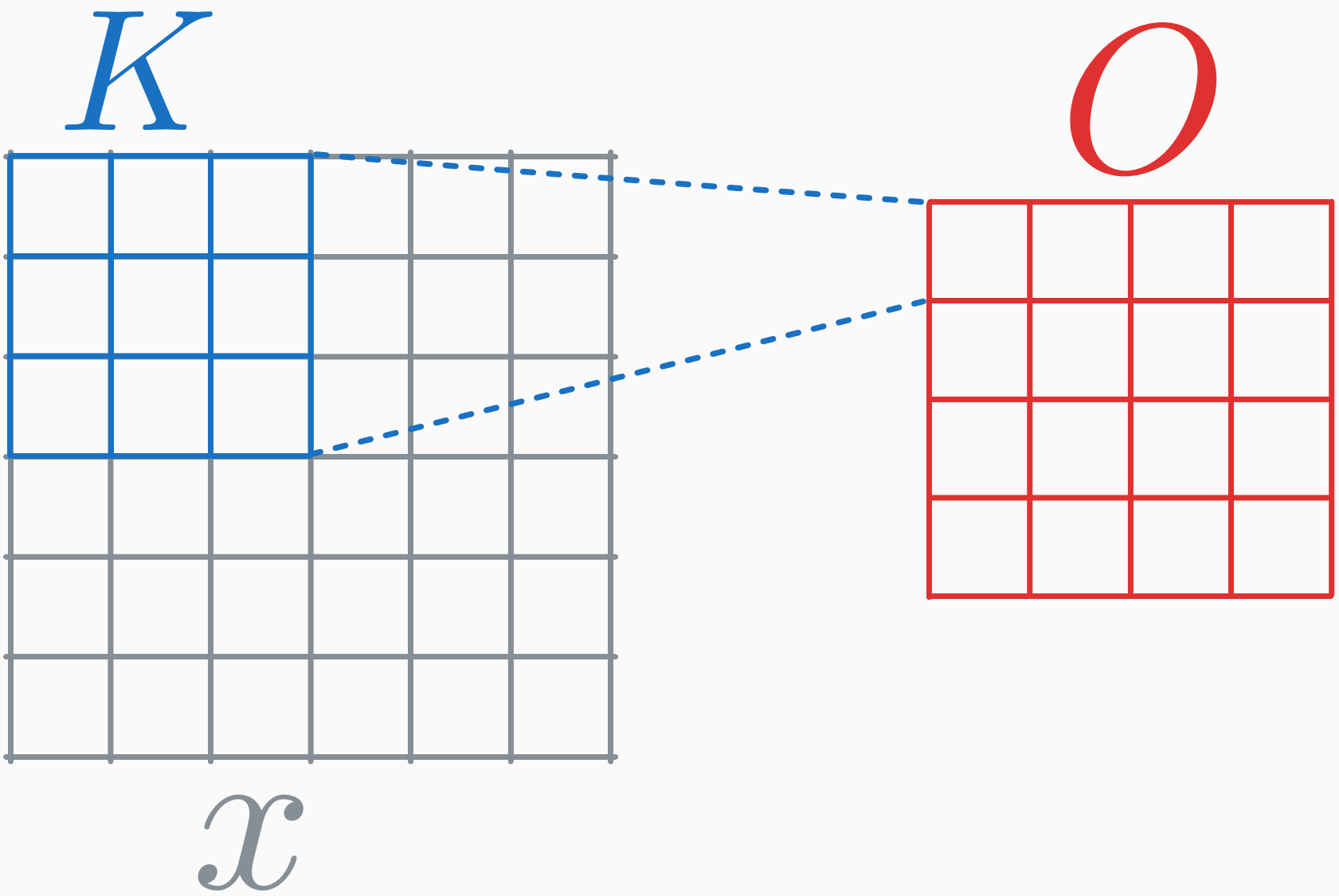
$\rightarrow$ Thus these functions are assumed as zero elsewere (defined only for the value stored)

Convolutional Neural Networks: 2D convolution

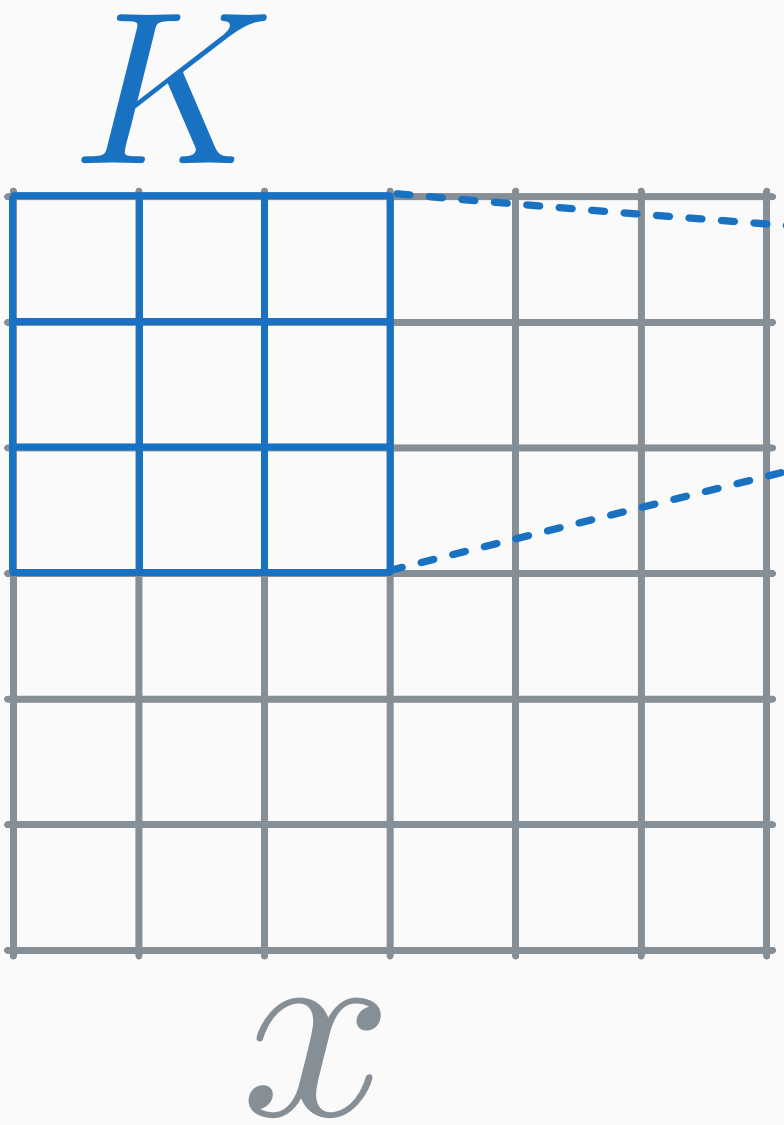
Let $x \in \mathbb{R}^{A,B}$ be a matrix (for instance an image) and $K \in \mathbb{R}^{N,M}$ an other matrix (the convolution kernel), we define the output of the convolution (in i and j) as:

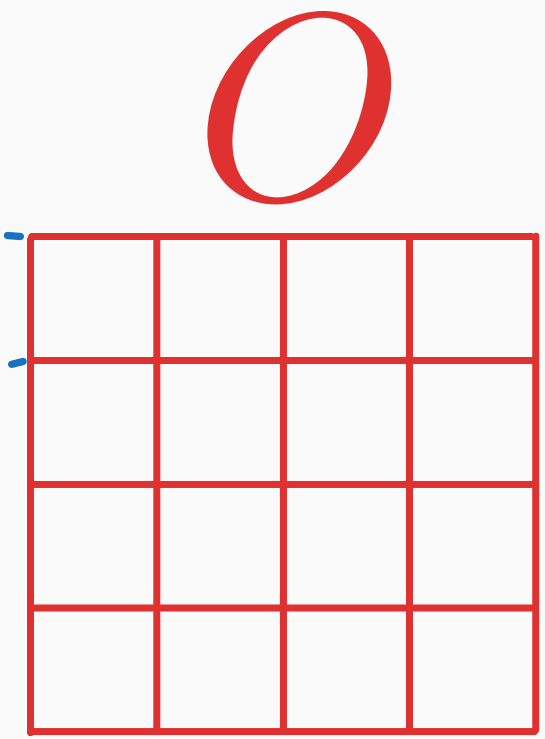
$$O_{i,j} = (x * K)_{i,j} = \sum_{n=1}^{N} \sum_{m=1}^{M} x_{i+n,j+m} K_{n,m}$$

Convolutional Neural Networks: The convolution on matrix



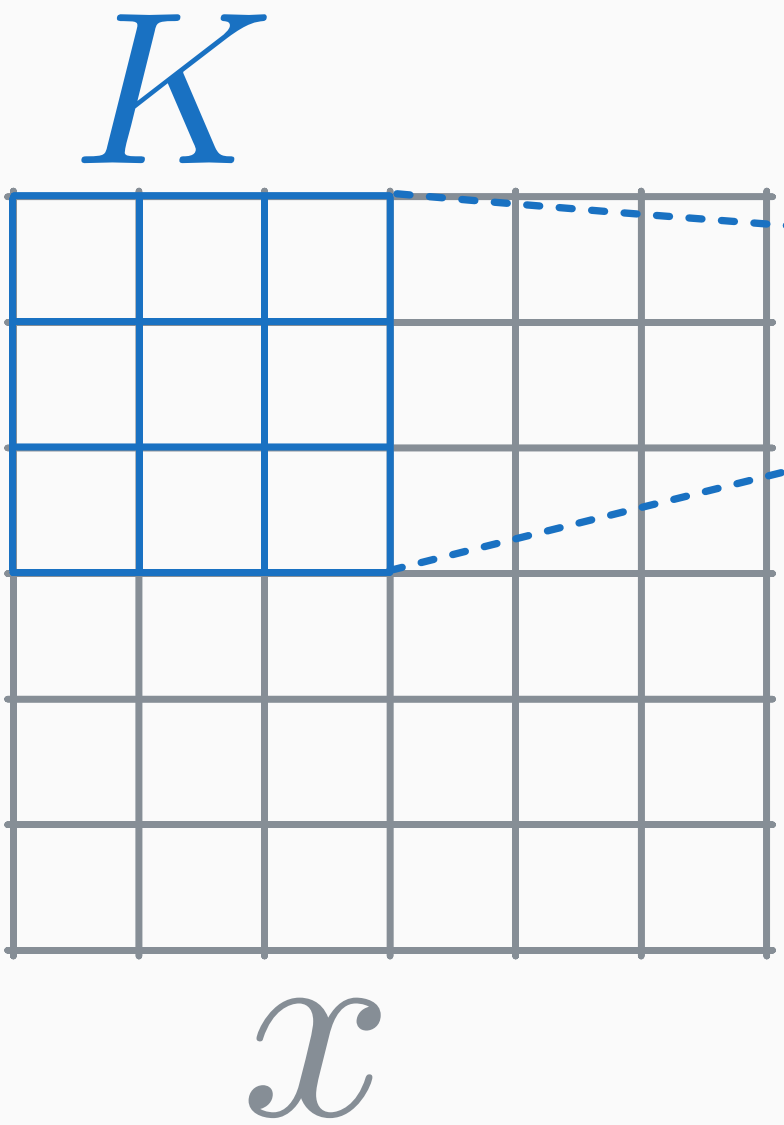
Convolutional Neural Networks: The convolution on matrix





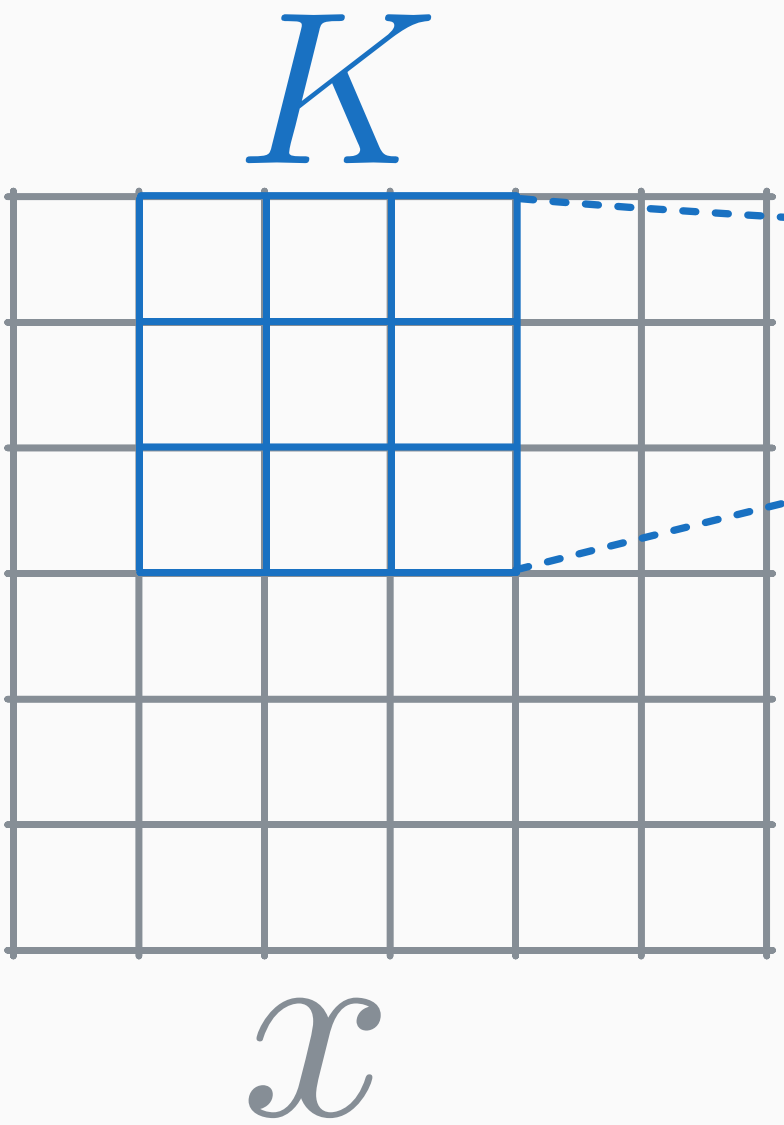
$$O_{i,j} = \sum_{n=1}^{W} \sum_{m=1}^{H} x_{i+n,j+m} K_{n,m}$$

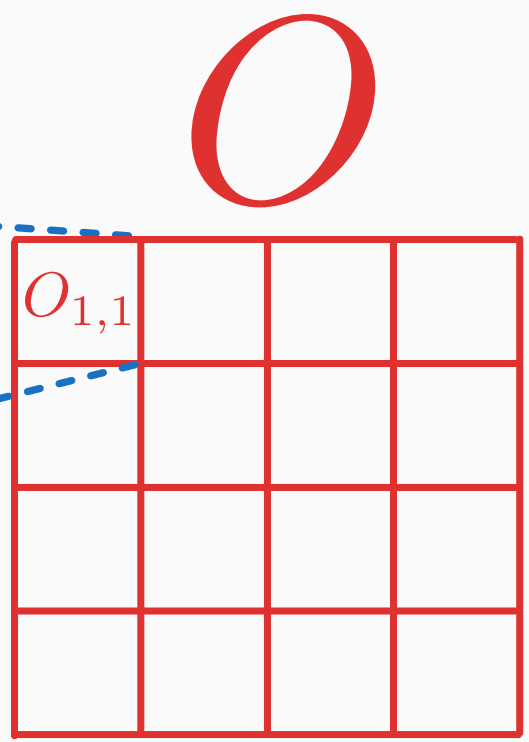
Convolutional Neural Networks: The convolution on matrix



$$O_{i,j} = \sum_{n=1}^{W} \sum_{m=1}^{H} x_{i+n,j+m} K_{n,m}$$

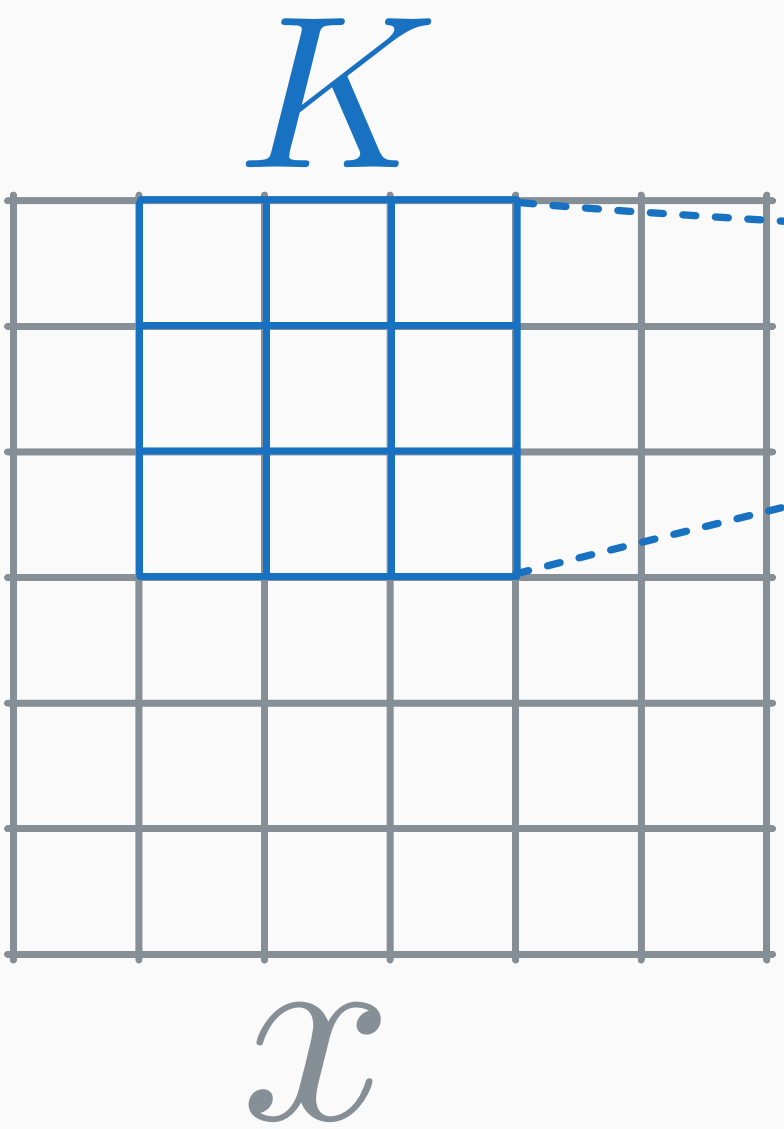
Convolutional Neural Networks: The convolution on matrix

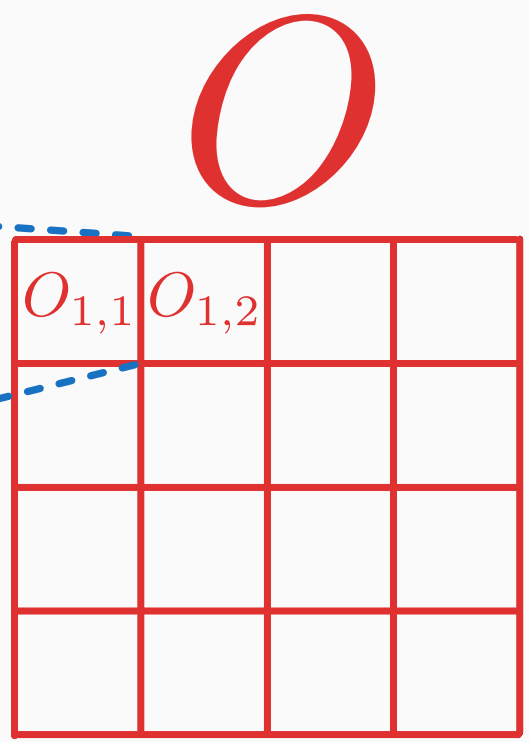




$$O_{i,j} = \sum_{n=1}^{W} \sum_{m=1}^{H} x_{i+n,j+m} K_{n,m}$$

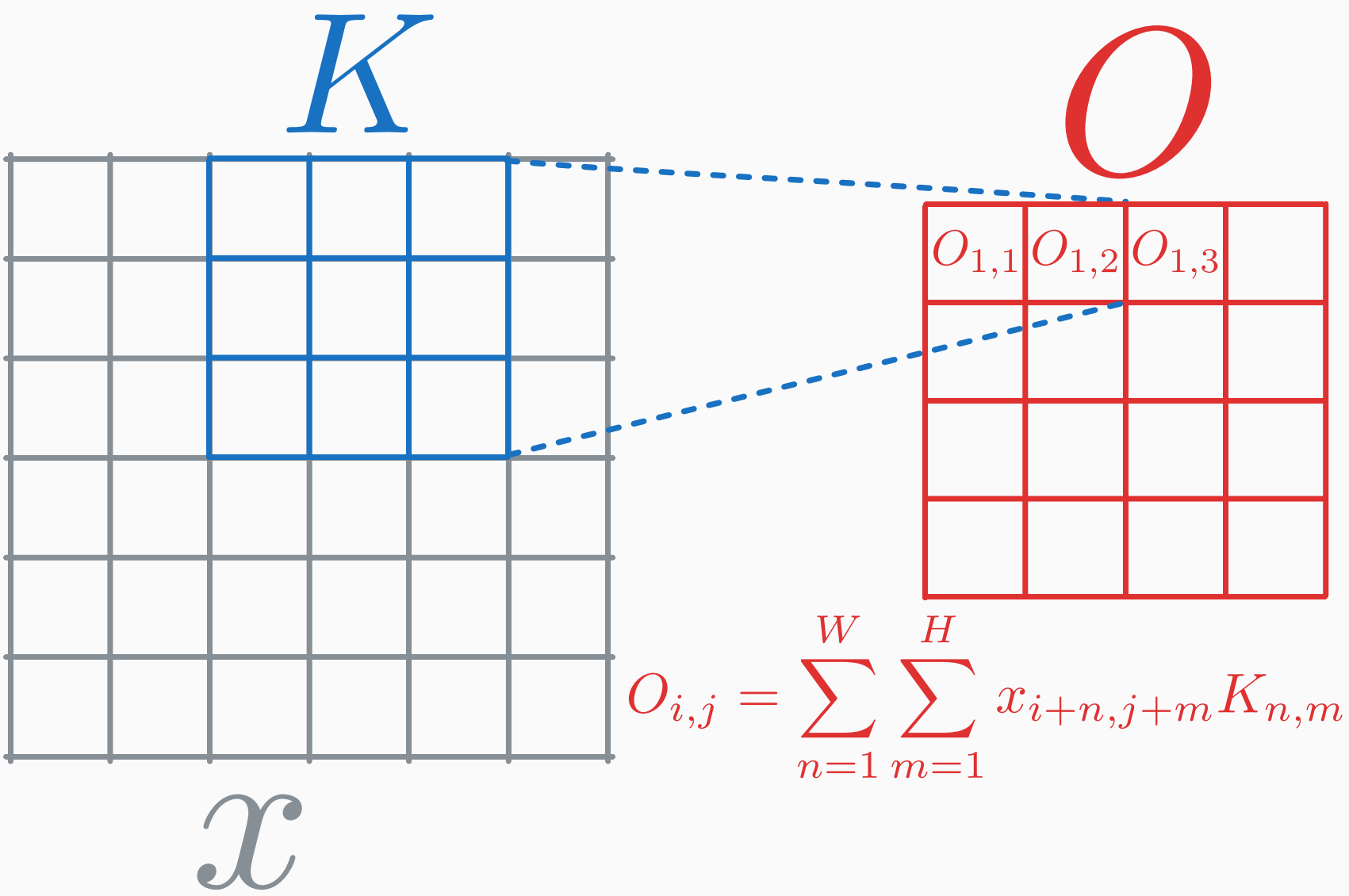
Convolutional Neural Networks: The convolution on matrix





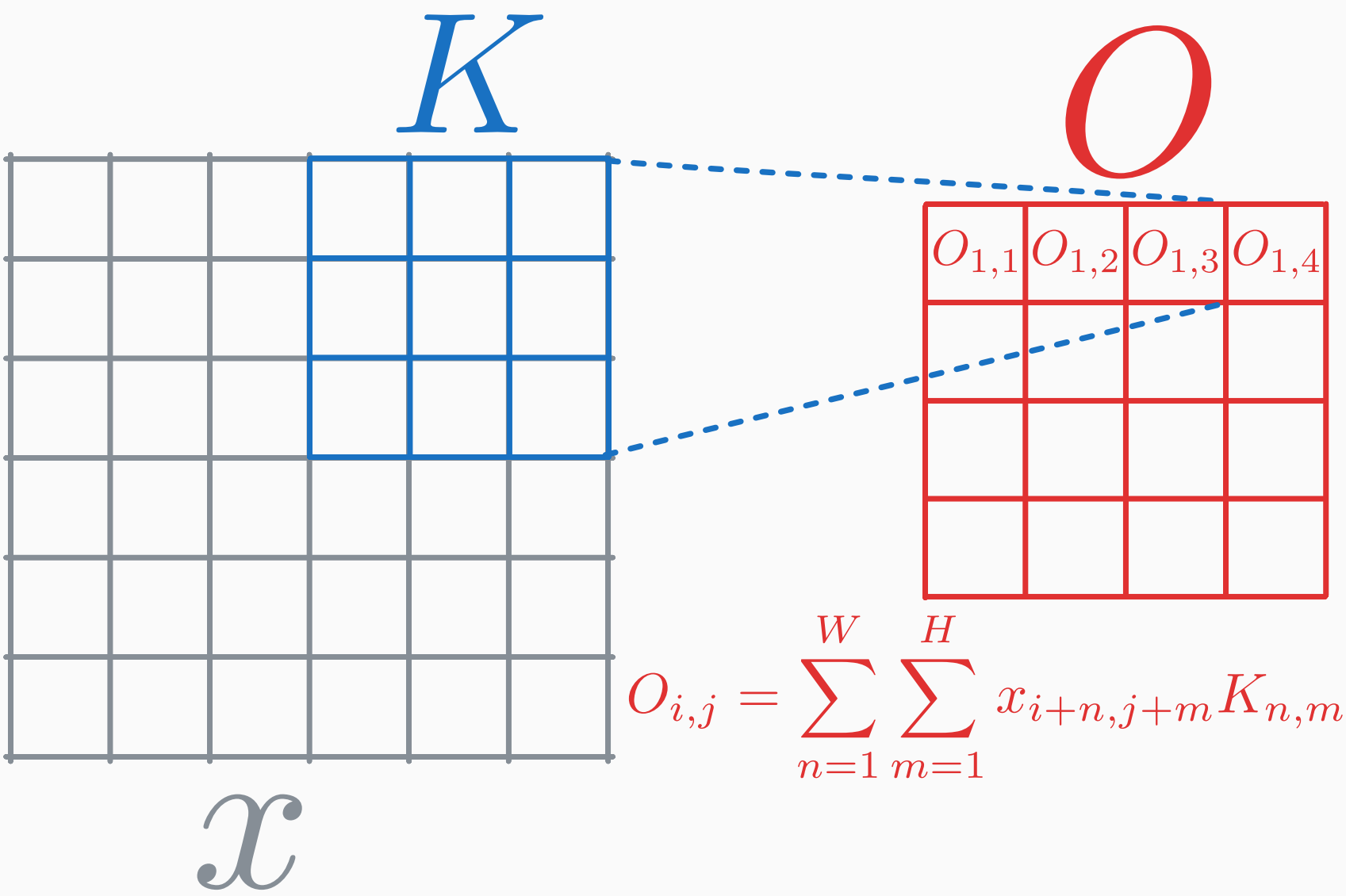
$$O_{i,j} = \sum_{n=1}^{W} \sum_{m=1}^{H} x_{i+n,j+m} K_{n,m}$$

Convolutional Neural Networks: The convolution on matrix



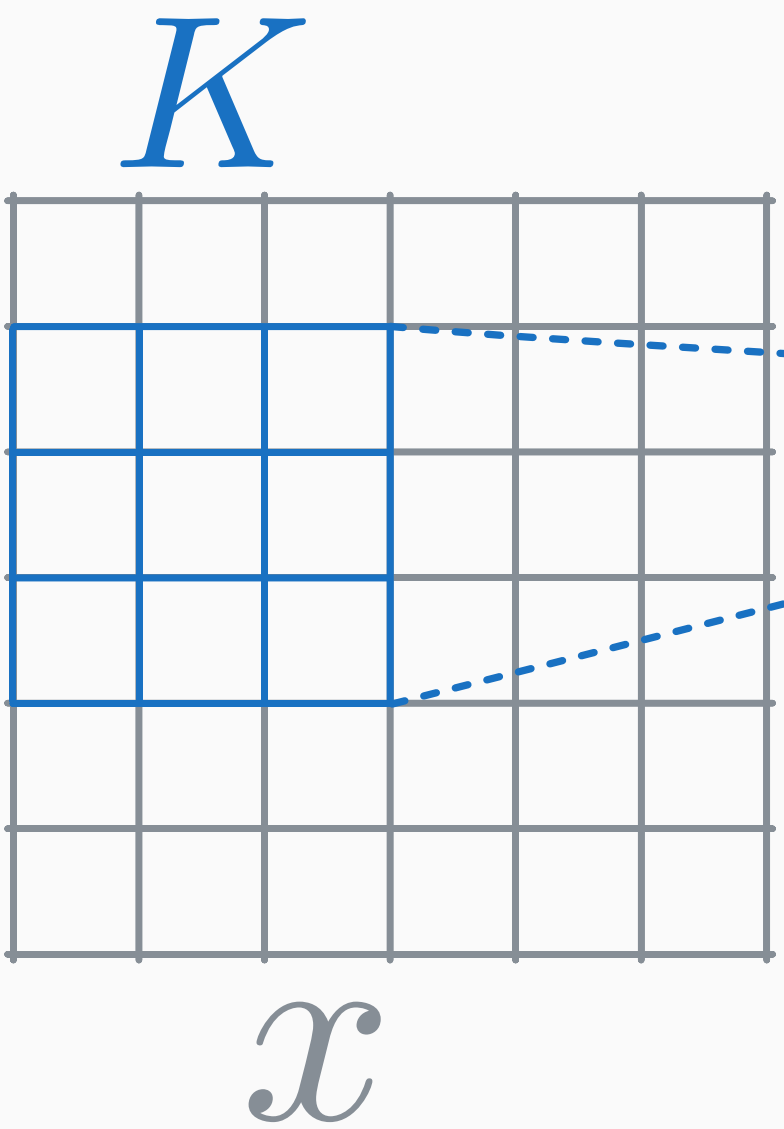
$$O_{i,j} = \sum_{n=1}^{W} \sum_{m=1}^{H} x_{i+n,j+m} K_{n,m}$$

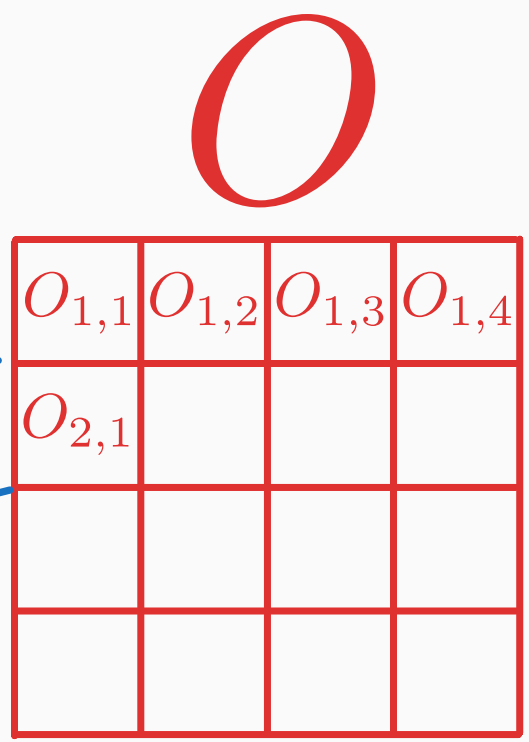
Convolutional Neural Networks: The convolution on matrix



$$O_{i,j} = \sum_{n=1}^{W} \sum_{m=1}^{H} x_{i+n,j+m} K_{n,m}$$

Convolutional Neural Networks: The convolution on matrix





$$O_{i,j} = \sum_{n=1}^{W} \sum_{m=1}^{H} x_{i+n,j+m} K_{n,m}$$

Finding borders (sharpening kernel)

Let

$$K = \begin{pmatrix} 0 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 0 \end{pmatrix}$$

be the convolution kernel

$$* K \quad =$$

Finding borders (sharpening kernel)

Let

$$K = \begin{pmatrix} 0 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 0 \end{pmatrix}$$

be the convolution kernel

$$*K \quad =$$

Convolution for different utilities

Bluring kernel

Let

$$K = \begin{pmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{pmatrix}$$

be the convolution kernel

$$* K \quad =$$

Convolution for different utilities

Bluring kernel

Let

$$K = \begin{pmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{pmatrix}$$

be the convolution kernel

$$* K \quad =$$

Convolution for different utilities

Bluring kernel

Let

$$K = \begin{pmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{pmatrix}$$

be the convolution kernel

$$* K \quad =$$

$\rightarrow$ a new pixel is the average of all its neighbor pixel

Objectives

Objectives

- Learn different filters/kernel (weights of the model)

Objectives

- Learn different filters/kernel (weights of the model)
- Each output of a convolution is called a feature map

Objectives

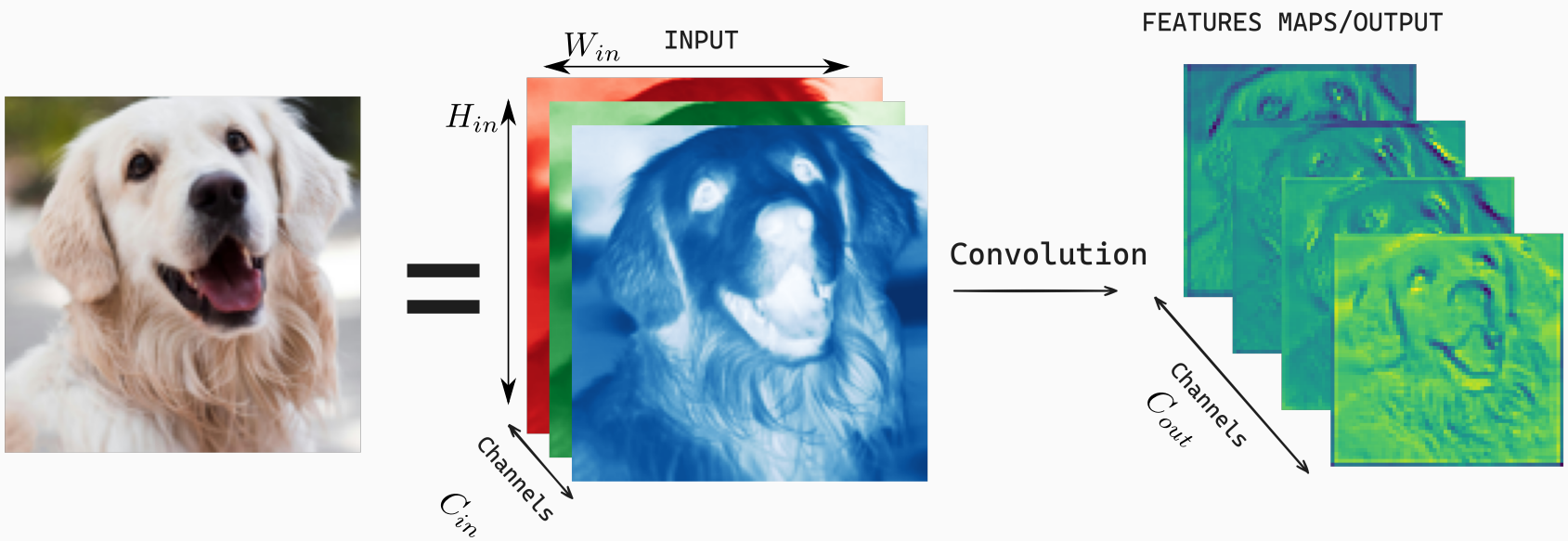
- Learn different filters/kernel (weights of the model)
- Each output of a convolution is called a feature map
- We want to capture class specificities in those features maps

Objectives

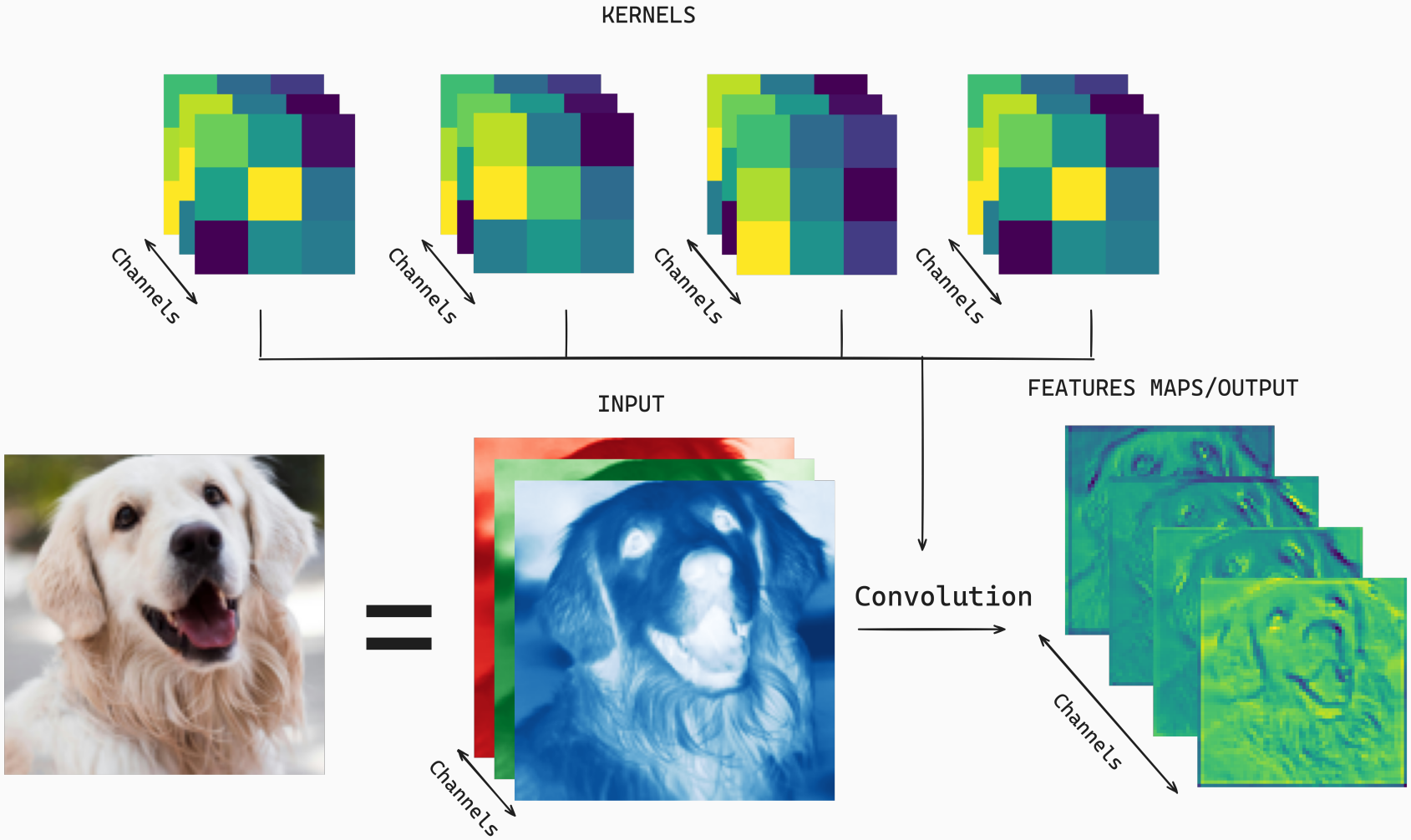
- Learn different filters/kernel (weights of the model)
- Each output of a convolution is called a feature map
- We want to capture class specificities in those features maps
- **Many input channel and output channel**
  $\rightarrow x \in \mathcal{R}^{W_{in} \times H_{in} \times C_{in}}$ and $K \in \mathcal{R}^{W_{ker} \times H_{ker} \times C_{in} \times C_{out}}$

Objectives

- Learn different filters/kernel (weights of the model)
- Each output of a convolution is called a feature map
- We want to capture class specificities in those features maps
- **Many input channel and output channel**
  $\rightarrow x \in \mathcal{R}^{W_{in} \times H_{in} \times C_{in}}$ and $K \in \mathcal{R}^{W_{ker} \times H_{ker} \times C_{in} \times C_{out}}$



Convolutional Neural Networks



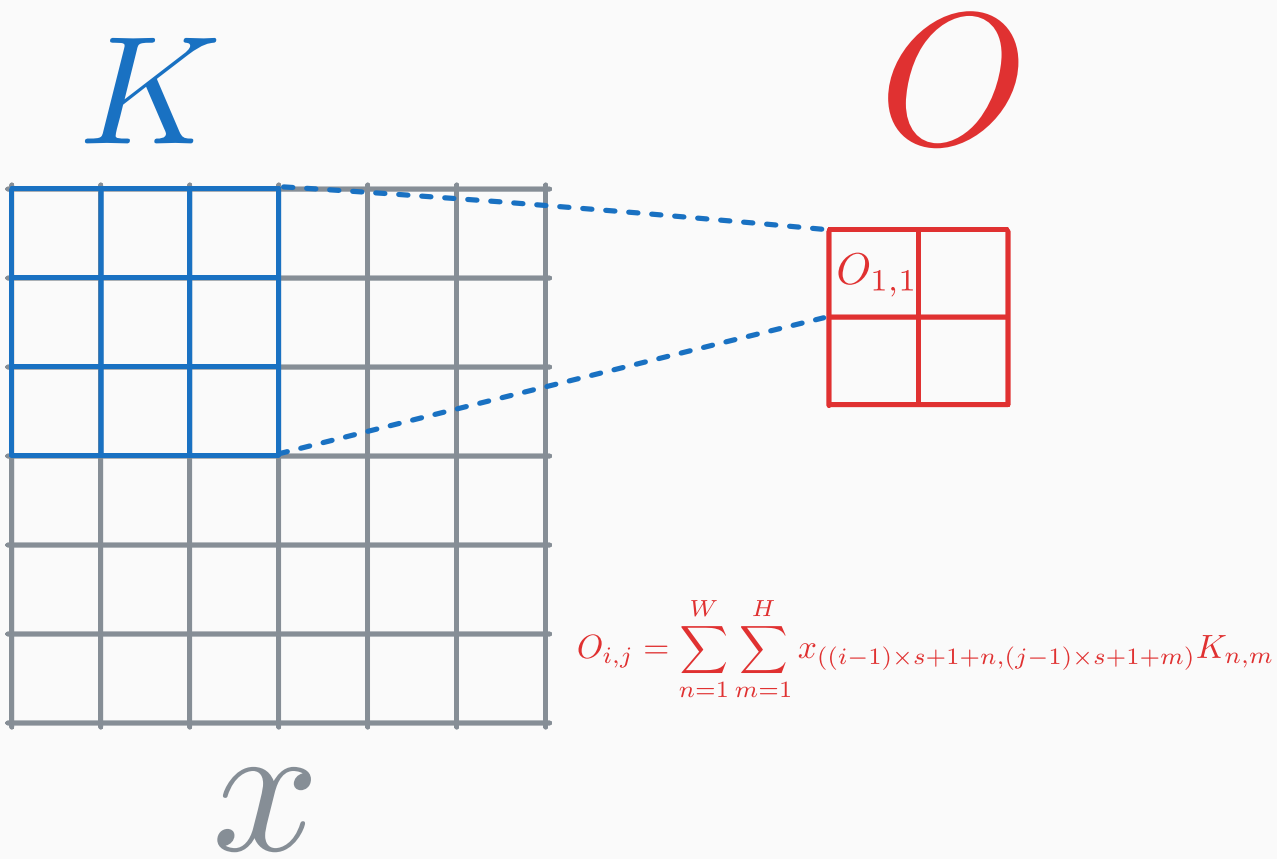
Convolutional Neural Networks: with stride

Stride in convolution

Move the window in each convolution steps of s pixels (an integer)

Stride in convolution

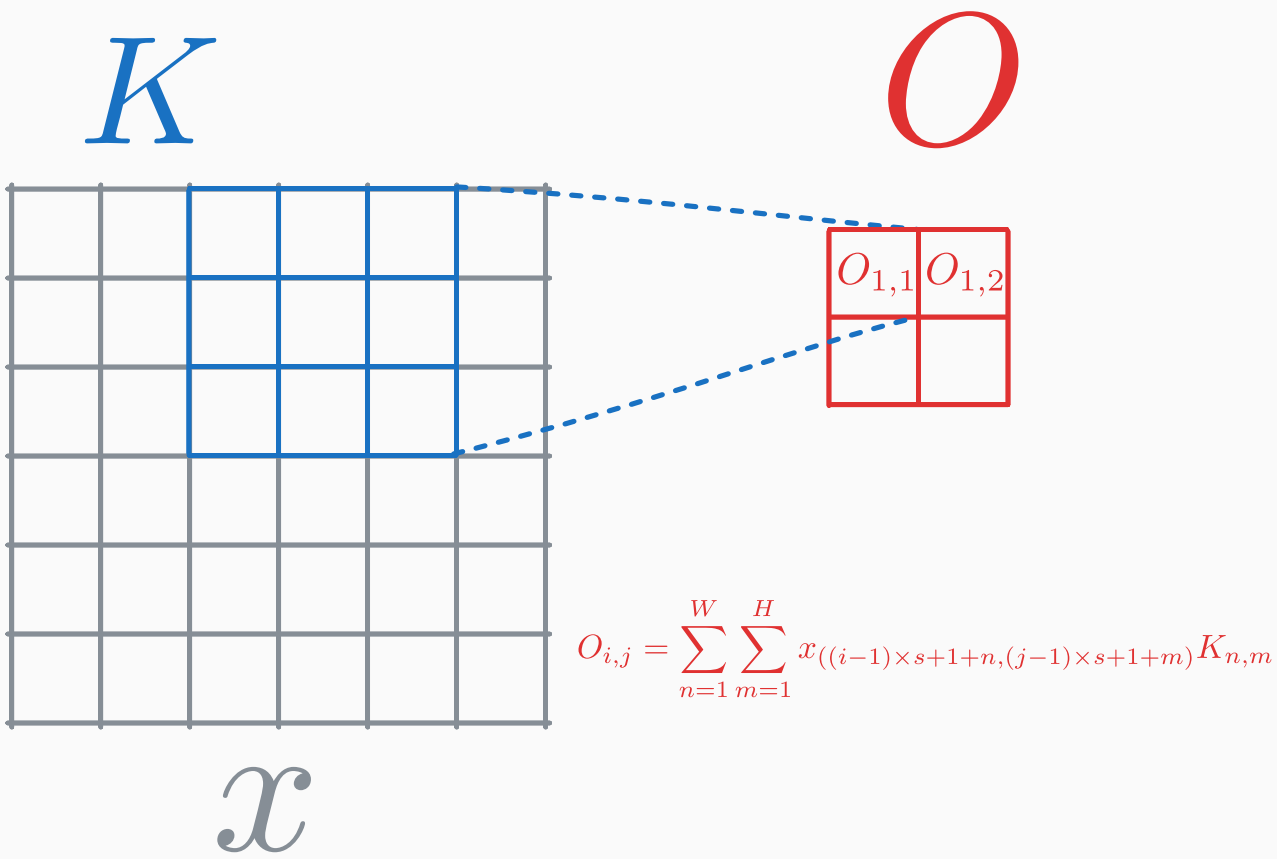
Move the window in each convolution steps of s pixels (an integer)



$$O_{i,j} = \sum_{n=1}^{W} \sum_{m=1}^{H} x_{((i-1)\times s+1+n,(j-1)\times s+1+m)} K_{n,m}$$

Stride in convolution

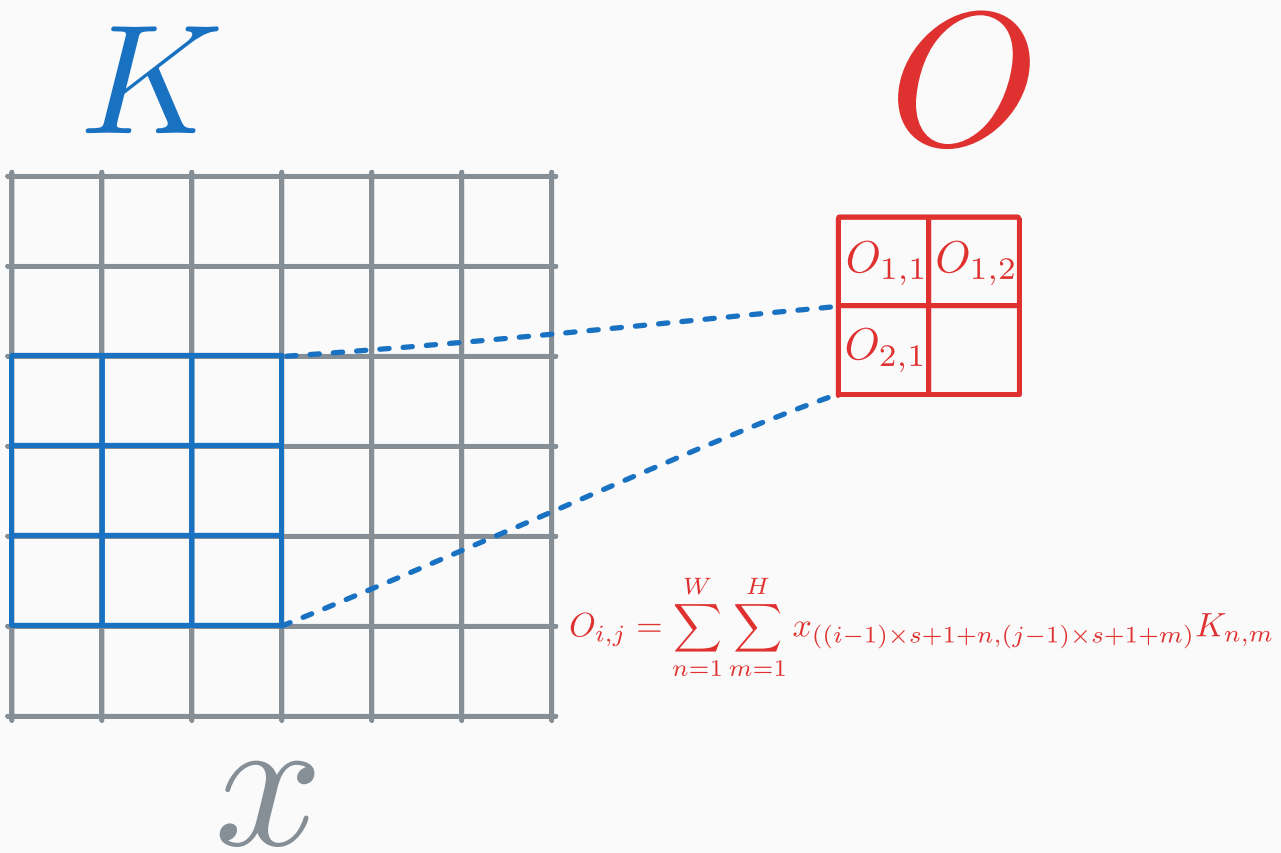
Move the window in each convolution steps of s pixels (an integer)



$$O_{i,j} = \sum_{n=1}^{W} \sum_{m=1}^{H} x_{((i-1)\times s+1+n,(j-1)\times s+1+m)} K_{n,m}$$

Stride in convolution

Move the window in each convolution steps of s pixels (an integer)



Padding in convolution

Padding

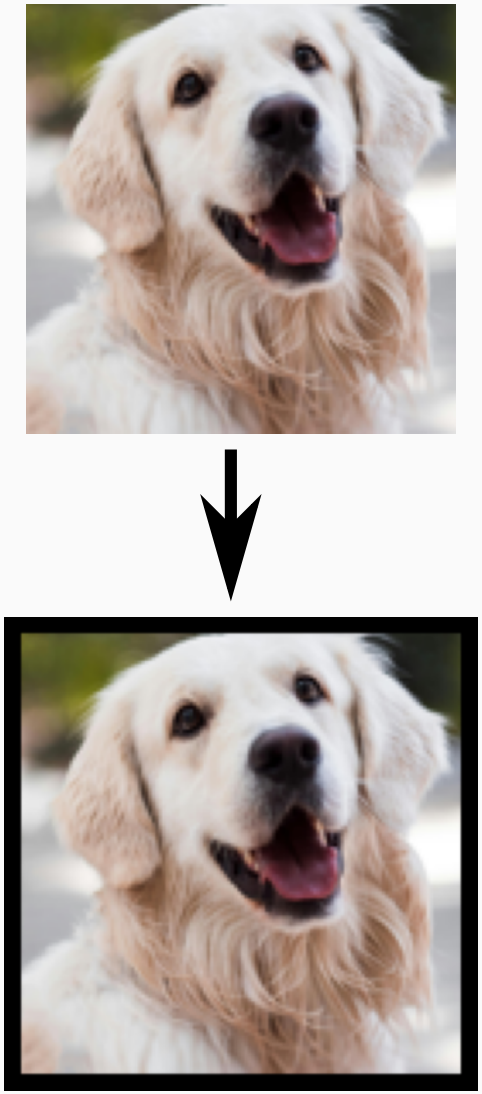
Pad the border of the image with a value

- Having a same size image
- Process border of the image

Zero padding

Padding with zero (border of image are set to zero)

$$\begin{pmatrix} a_{1,1} & a_{1,2} \\ a_{2,1} & a_{2,2} \end{pmatrix} \rightarrow \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & a_{1,1} & a_{1,2} & 0 \\ 0 & a_{2,1} & a_{2,2} & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}$$



Convolution summary

Convolution module/operator

Let be $s \in \mathbb{N}$ the stride (consider same stride along both axis), let be $x \in \mathcal{R}^{W_{in} \times H_{in} \times C_{in}}$ the input and $K \in \mathcal{R}^{W_{ker} \times H_{ker} \times C_{in} \times C_{out}}$ the kernel.

[0] Notice that we consider in the example the input padded

Convolution summary

Convolution module/operator

Let be $s \in \mathbb{N}$ the stride (consider same stride along both axis), let be $x \in \mathcal{R}^{W_{in} \times H_{in} \times C_{in}}$ the input and $K \in \mathcal{R}^{W_{ker} \times H_{ker} \times C_{in} \times C_{out}}$ the kernel.

The ouput O will be of width:

[0] Notice that we consider in the example the input padded

Convolution module/operator

Let be $s \in \mathbb{N}$ the stride (consider same stride along both axis), let be $x \in \mathcal{R}^{W_{in} \times H_{in} \times C_{in}}$ the input and $K \in \mathcal{R}^{W_{ker} \times H_{ker} \times C_{in} \times C_{out}}$ the kernel.

The ouput O will be of width:

$$W_{out} \leq \frac{W_{in} - W_{ker}}{s} + 1$$

[0] Notice that we consider in the example the input padded

Convolution module/operator

Let be $s \in \mathbb{N}$ the stride (consider same stride along both axis), let be $x \in \mathcal{R}^{W_{in} \times H_{in} \times C_{in}}$ the input and $K \in \mathcal{R}^{W_{ker} \times H_{ker} \times C_{in} \times C_{out}}$ the kernel.

The ouput O will be of width:

$$W_{out} \leq \frac{W_{in} - W_{ker}}{s} + 1$$

($W_{out} \in \mathbb{N}$) and height:

[0] Notice that we consider in the example the input padded

Convolution module/operator

Let be $s \in \mathbb{N}$ the stride (consider same stride along both axis), let be $x \in \mathcal{R}^{W_{in} \times H_{in} \times C_{in}}$ the input and $K \in \mathcal{R}^{W_{ker} \times H_{ker} \times C_{in} \times C_{out}}$ the kernel.

The ouput O will be of width:

$$W_{out} \leq \frac{W_{in} - W_{ker}}{s} + 1$$

($W_{out} \in \mathbb{N}$) and height:

$$H_{out} \leq \frac{H_{in} - H_{ker}}{s} + 1$$

[0] Notice that we consider in the example the input padded

Convolution module/operator

Let be $s \in \mathbb{N}$ the stride (consider same stride along both axis), let be $x \in \mathcal{R}^{W_{in} \times H_{in} \times C_{in}}$ the input and $K \in \mathcal{R}^{W_{ker} \times H_{ker} \times C_{in} \times C_{out}}$ the kernel.

The ouput O will be of width:

$$W_{out} \leq \frac{W_{in} - W_{ker}}{s} + 1$$

($W_{out} \in \mathbb{N}$) and height:

$$H_{out} \leq \frac{H_{in} - H_{ker}}{s} + 1$$

($H_{out} \in \mathbb{N}$). Thus, $O \in \mathbb{R}^{W_{out} \times H_{out} \times C_{out}}$

[0] Notice that we consider in the example the input padded

Convolution summary

Convolution module/operator

Let be $s \in \mathbb{N}$ the stride (consider same stride along both axis), let be $x \in \mathcal{R}^{W_{in} \times H_{in} \times C_{in}}$ the input and $K \in \mathcal{R}^{W_{ker} \times H_{ker} \times C_{in} \times C_{out}}$ the kernel. The ouput $O_{i,j,l}$ will be

$$O_{i,j,l} = \sum_{n=0}^{W_{ker}} \sum_{m=0}^{H_{ker}} \sum_{k=0}^{C_{in}} x_{((i-1)\times s+1+n,(j-1)\times s+1+m,k)} K_{n,m,k,l}$$

- **Stride:** Window shift between each step (number of pixels)
- **Padding:** Involves adding extra pixels around the border of the input feature map before convolution

[0] Notice that we consider in the example the input already padded

Convolutional Neural Networks: Pooling operation

Objectives

- Select most relevant part of the features map

Convolutional Neural Networks: Pooling operation

Objectives

- Select most relevant part of the features map
- We want to capture class specificities in those features maps

Convolutional Neural Networks: Pooling operation

Objectives

- Select most relevant part of the features map
- We want to capture class specificities in those features maps

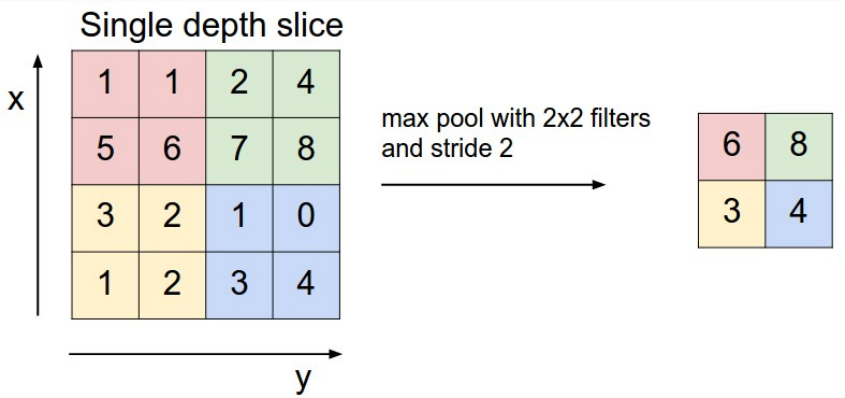
$\rightarrow$ Use pooling operation

Convolutional Neural Networks: Pooling operation

Objectives

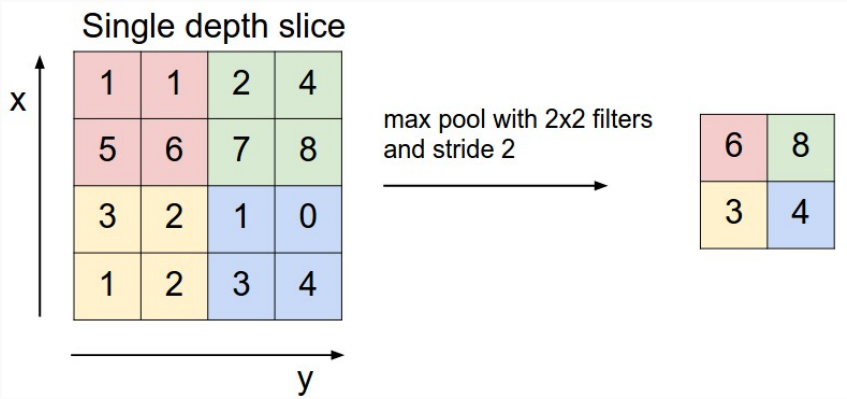
- Select most relevant part of the features map
- We want to capture class specificities in those features maps

→ Use pooling operation



Objectives

- Select most relevant part of the features map
- We want to capture class specificities in those features maps



$\rightarrow$ Use pooling operation

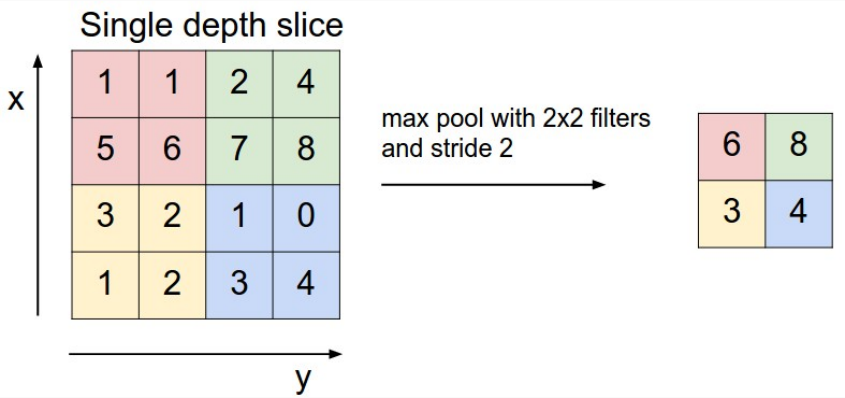
Average Pooling

Window averaging on features maps

$$O_{i,j,l} = \frac{1}{NM} \sum_{n=1}^{N} \sum_{m=1}^{M} x_{((i-1)\times s+1+n,(j-1)\times s+1+m,l)}$$

Objectives

- Select most relevant part of the features map
- We want to capture class specificities in those features maps



$\rightarrow$ Use pooling operation

Average Pooling

Window averaging on features maps

$$O_{i,j,l} = \frac{1}{NM} \sum_{n=1}^{N} \sum_{m=1}^{M} x_{((i-1)\times s+1+n,(j-1)\times s+1+m,l)}$$

Max Pooling

Window maximum on features maps

$$O_{i,j,l} = \max_{n,m} x_{((i-1)\times s+1+n,(j-1)\times s+1+m,l)}$$

Convolutional Neural Networks: Whole architecture

Convolutional neural network architecture

- Alternate pooling and convolution (with non linear activations)

Convolutional Neural Networks: Whole architecture

Convolutional neural network architecture

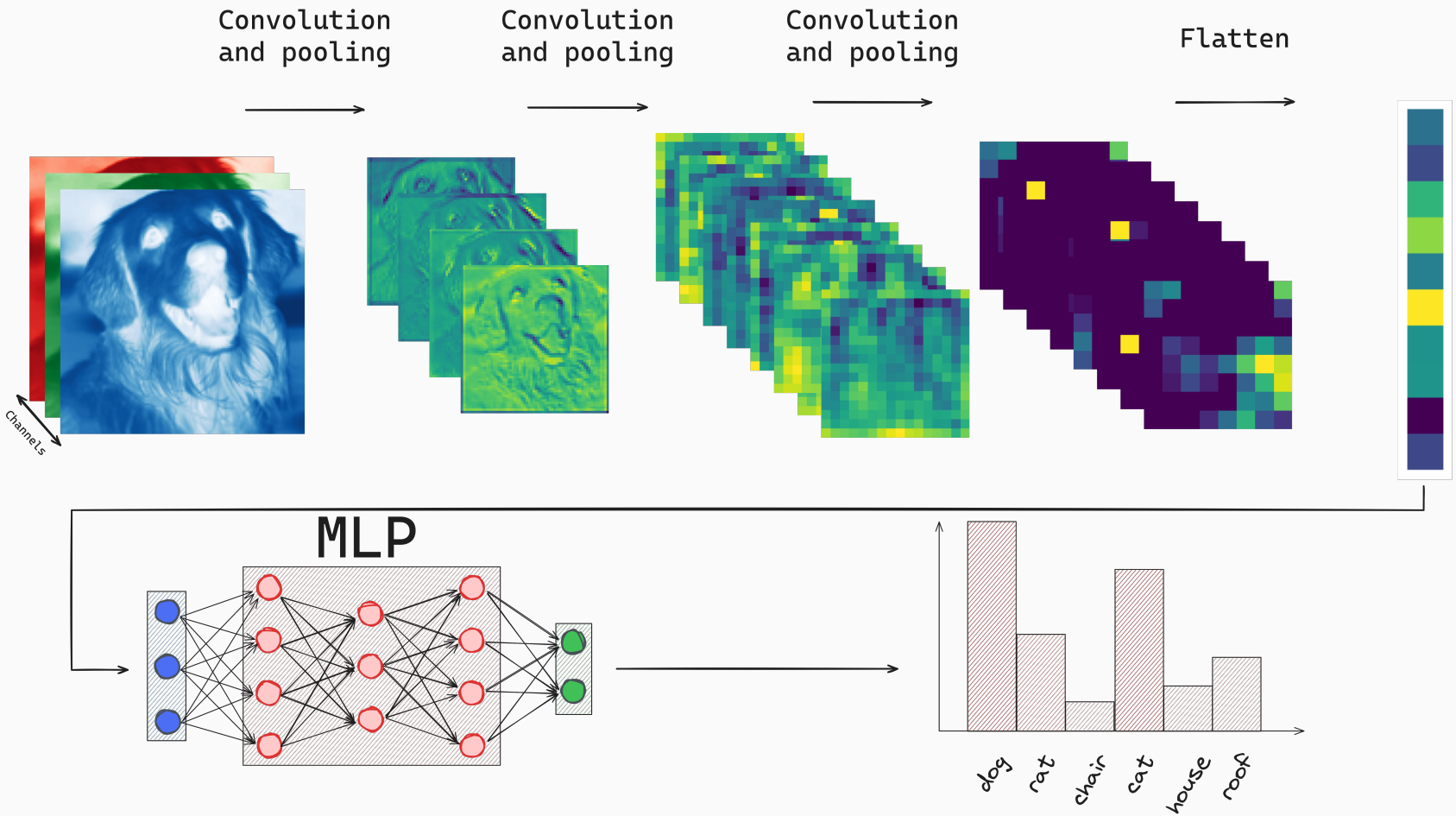
- Alternate pooling and convolution (with non linear activations)
- At a certain layer $\rightarrow$ flatten features map to get a 1 dimensional vector

Convolutional Neural Networks: Whole architecture

Convolutional neural network architecture

- Alternate pooling and convolution (with non linear activations)
- At a certain layer $\rightarrow$ flatten features map to get a 1 dimensional vector
- Use a classical MLP to predict the class (also called a Feed-Forward network)

Convolutional Neural Network



The VGG-16 architecture

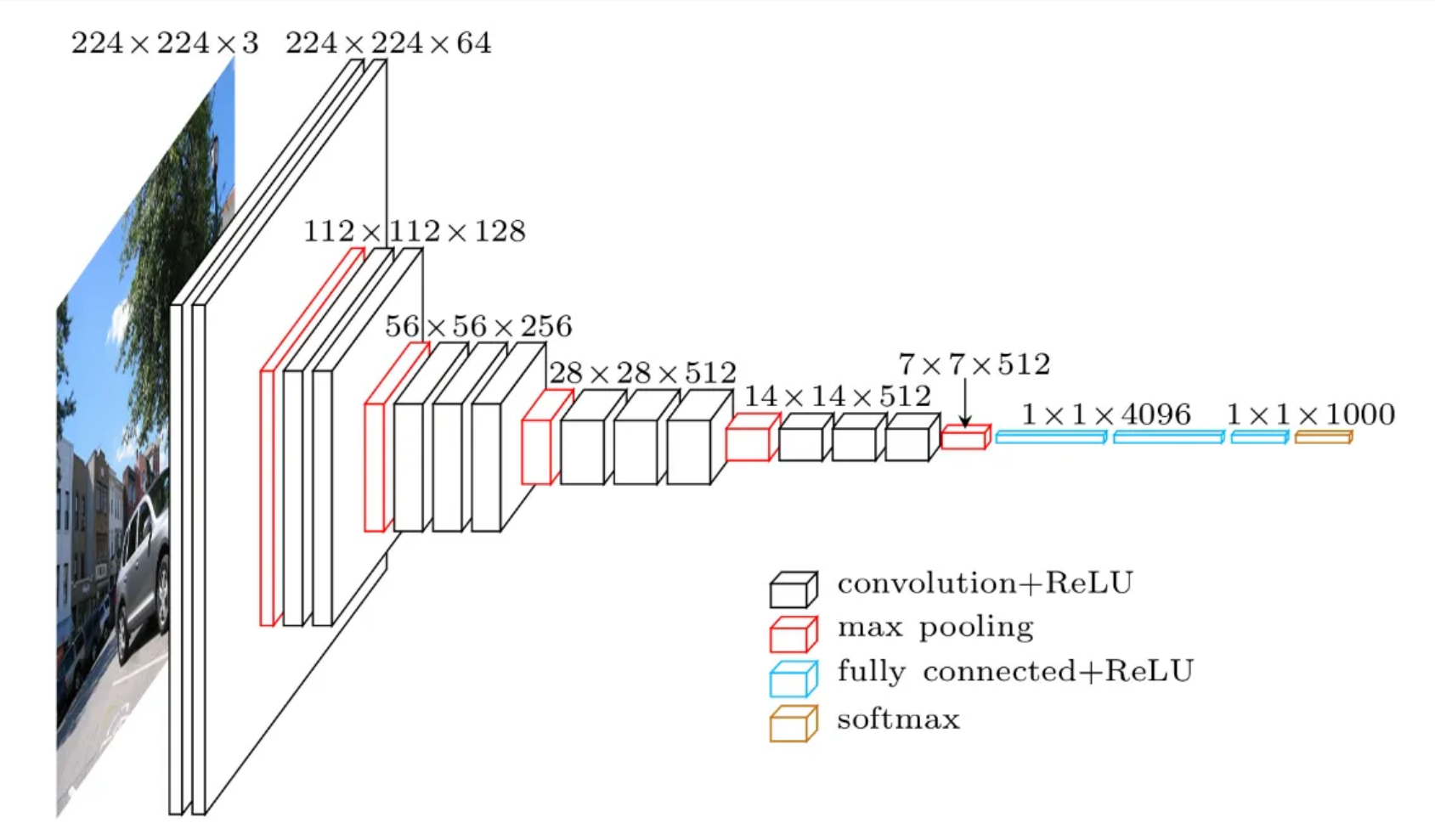


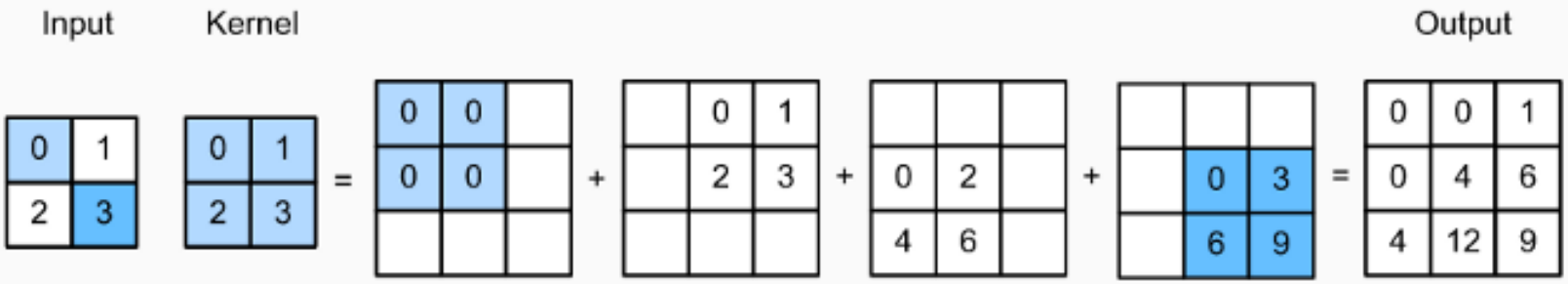
Figure 1: The VGG16 (Visual Geometry Group) CNN[1]

[1] https://lekhuyen.medium.com/an-overview-of-vgg16-and-nin-models-96e4bf398484

Image generation?

$\rightarrow$ Generate images using "inverse" convolution layers

Main idea:



(see deconvolution or transposed convolution layers)

Exercises

Implementation Of CNN

- Implement Convolution function
- Use pytorch to implement a CNN
- Comparison of the results

Temporary page!

LaTeX was unable to guess the total number of pages correctly. As there was some unprocessed data that should have been added to the final page this extra page has been added to receive it.

If you rerun the document (without altering it) this surplus page will go away, because LaTeX now knows how many pages to expect for this docume