POLITECNICO
MILANO 1863

2024-25

CNN: Convolutional Neural Networks

Theoretical Project of
Numerical Analysis for Machine Learning

Giorgio Monaco - 10775329

Advisor: Prof. Edie Miglio

Introduction

- **Deep Learning (DL)** revolutionized machine learning (ML) with automatic extraction of representations from raw data.
- CNN stands as one of the most influential architecture, originally born for computer vision, speech recognition and natural language processing.

Key principles:

- **Convolution** : local connectivity & weight sharing.
- **Pooling** : invariance to small translations.
- **Hierarchical feature representation** : from simple to abstract features.

This architecture was inspired by some revolutionary studies in neuroscience.

What is Convolution?

It's mathematical operation between two functions, graphically expressing how the *shape* of one function is modified by the other.

$$s(t) = (f * g)(t) = \int f(\tau)g(t-\tau)d\tau$$

Where:

- f is the **input**
- g is the **kernel**
- The output is the **feature map**

It's a fundamental building block of Convolutional Neural Networks (CNNs), allowing them to extract efficiently local features from structured data.

What is Convolution?

When we work with CNNs, the time will be usually discretized, and the input is usually two-dimensional (images):

$$S(i,j) = (K * I)(i,j) = \sum_m \sum_n I(m,n)\ K(i-m, j-n)$$

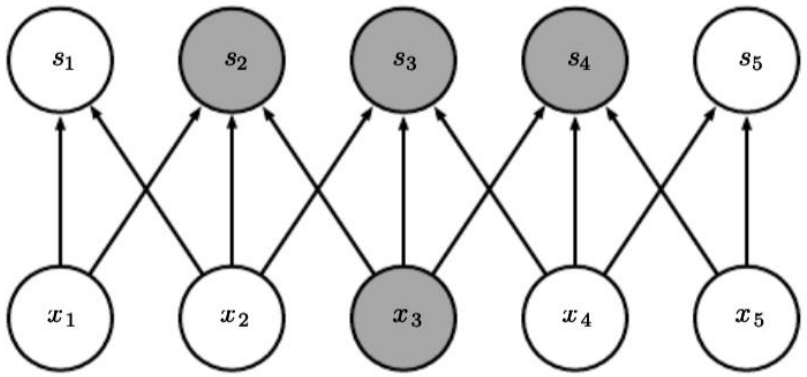
Without flipping the kernel, we obtain a slightly different operation called **cross-correlation**, more used in CNNs, defined as:

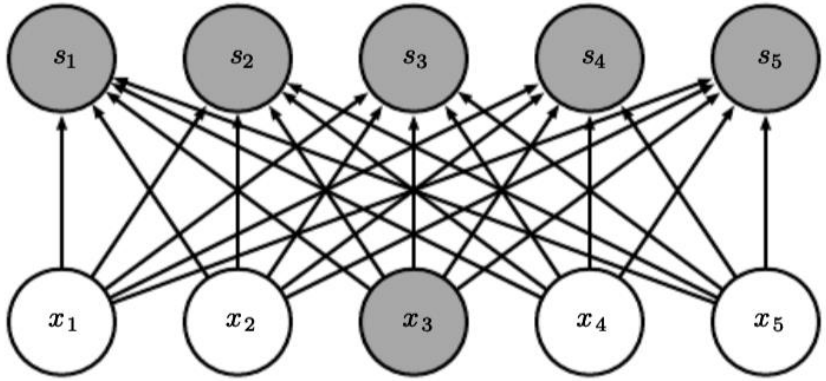
$$S(i,j) = (K * I)(i,j) = \sum_m \sum_n I(i+m, j+n)\ K(m,n)$$

Motivation for Convolution – Sparse Interactions

- Three key principles: **sparse interactions**, **parameter sharing** and **equivariance to translation**

Convolution connect every output unit in a neural network only to a local neighbourhood of the input, known as its *receptive field.* This sparsity not only reduces the number of parameters, but also encodes the prior knowledge that local groups of variables are often more strongly correlated than distant ones.





Motivation for Convolution – Parameter Sharing

In CNNs the same kernel is applied across all spatial locations. We can say that the network has *tied weights*, because the values of the weights are the same everywhere.

This means that instead of learning a separate set of weights for each location, the model learns only one kernel that is reused.

Motivation for Convolution – Equivariance to Translation

Formally, a function f is equivariant to an operation T if, applying T to the input and then f , gives the same result as applying f first and then T to the output.
Convolution satisfies this property with respect to translations:

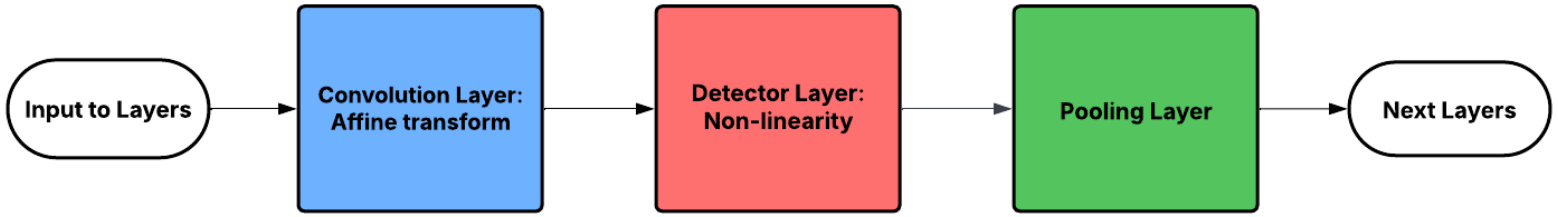
$$(K * I)(x + \Delta) = T_{\Delta}\big((K * I)(x)\big)$$

where T_{Δ} denotes a translation by Δ.

This means that if the input image is shifted, the output feature map shifts in the same way.

Pooling

A standard layer in a convolutional network is typically composed of three stages.



Pooling aims to reduce the spatial resolution of feature maps while retaining the most important information.

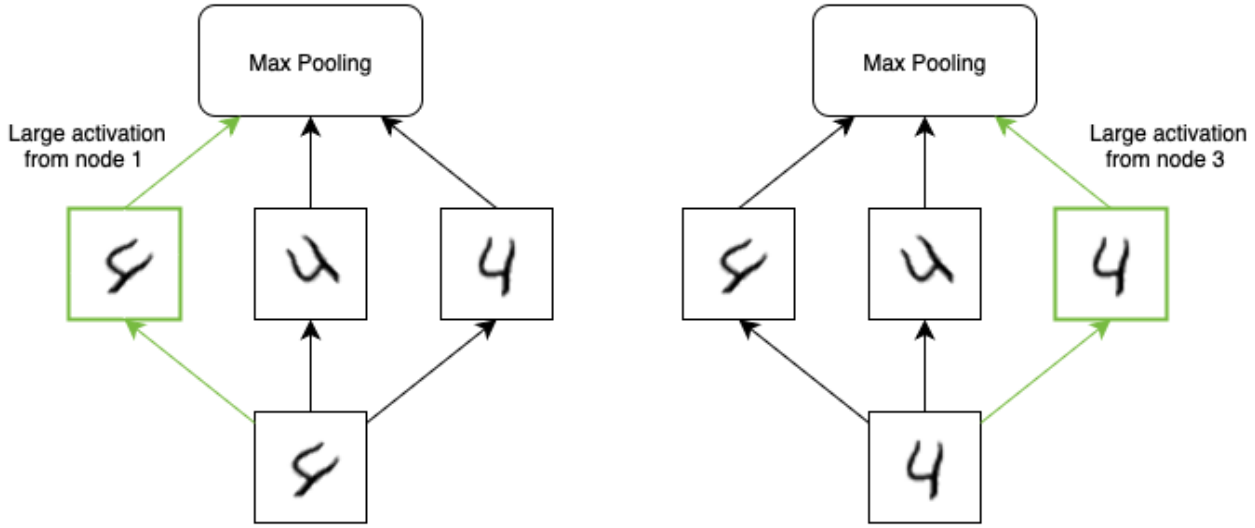
The two most common pooling operations are:

- **Max pooling**, which reports the maximum value within a rectangular neighbourhood.
- **Average pooling**, which computes the mean value.

Pooling

The central benefit introduced by pooling lies in the approximately **invariance to small translations** of the input, particularly valuable in visual recognition tasks.

It's also possible, if pooling over the output of separately parametrized convolutions, to make features able to learn which transformations to become invariant to.



Pooling

Alternatives to pooling have been explored in recent research:

- **Strided convolution** propose an extremely simple architecture composed of only convolutional layers and application of stride greater or equal than 2.
- **SoftPool** aims to address the natural limitation of pooling represented by the lost information produced by max or average pooling.

Both techniques reached impressive results, raising important questions about the necessity of pooling in CNNs.

Variants of Convolution – Strided Convolution

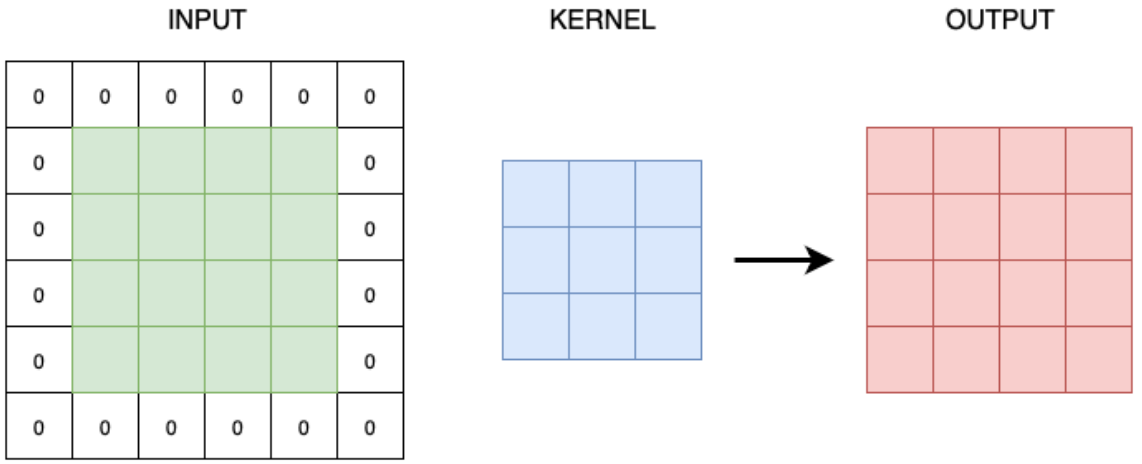
The convolution operation at the core of CNNs admits several useful modifications that expand its flexibility and efficiency

- Usually, kernel is applied at every possible location in the input. A simple modification consists of applying the kernel only at locations separated by a fixed step size, known as the **stride**.

$$Z_{i,j,k} = c(K, V, s)_{i,j,k} = \sum_{l,m,n} \left[V_{l,(j-1)\times s+m,(k-1)\times s+n} K_{i,l,m,n} \right]$$

Variants of Convolution – Zero Padding

- Another common variant is related to the ability of **zero-padding** the input. It allows control over the size of the output feature maps.



The most common conventions are **valid convolution**, **same convolution** and **full convolution**.

Variants of Convolution – Unshared Convolution

- We can relax the assumption of weight sharing of CNNs, leading to the concept of *locally connected layers*.

The most general case is the **unshared convolution** where each output unit of the neural network has its own distinct set of weights rather than sharing the same kernel.

$$y_i = \sum_{j=1}^{k} w_j^{(i)} \, x_{i+j-1}, \quad i = 1, \dots, m$$

Locally connected layers are useful when we know that each feature should be a function of a small part of space, but there is no reason to think that the same feature should occur across all of space.

Variants of Convolution – Tiled Convolution

There exists also an intermediate approach known as **tiled convolution**. In this variant, weights are shared, but only among a restricted set of positions.

$$y_i = \sum_{j=1}^{k} w_j^{(i \bmod T)} \, x_{i+j-1}.$$

This design relaxes the strong prior of full translation equivariance while still controlling the number of parameters compared to the fully unshared case.

Variants of Convolution – Separable Convolution

- A kernel can sometimes be factorized into simpler components. This is known as **spatially separable convolution** and it reduces computation when the factorization is exact or a good approximation

An important extension is the *depthwise separable convolution*, that factorizes convolution into two steps: a depthwise convolution and a pointwise convolution. This decomposition greatly reduces the number of parameters and multiplications, enabling efficient CNNs suitable for mobile and embedded devices.

Applications of Convolution

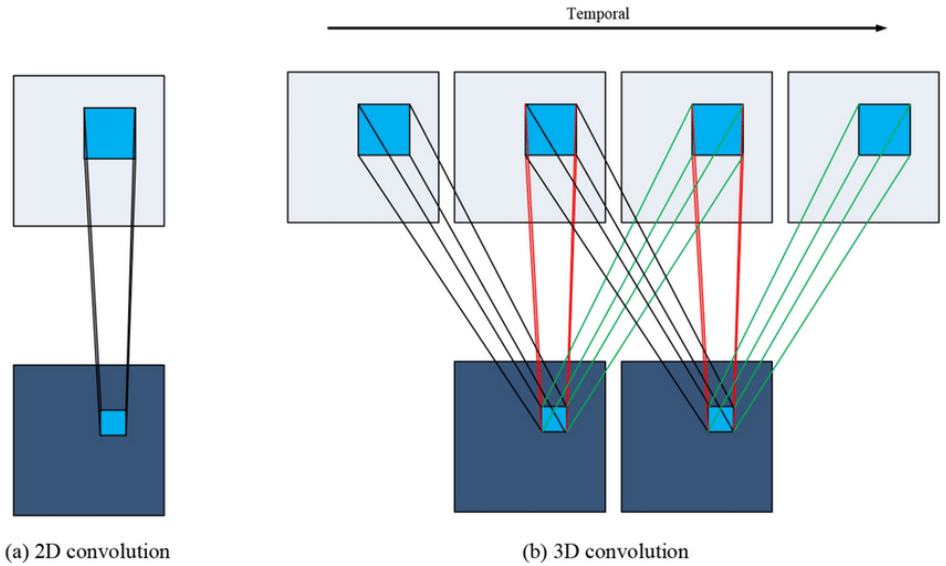
The simplest practical applications of convolution are to one-dimensional and two-dimensional arrays.

- **1D convolution** is suitable for speech recognition, time-series forecasting and other tasks where local temporal context is important.
- **2D convolution** is characteristic of image data and created the foundation of modern computer vision systems.

Applications of Convolution

- Convolution naturally extends to **three or more dimensions**. A key application is in video understanding, where the input has two spatial dimensions and one temporal dimension.

- Recent reaserch proposed one of the first effective 3D CNN architectures for *human action recognition*.



(a) 2D convolution (b) 3D convolution

Applications of Convolution

- Formally, the value at position (x, y, z) on the j-th feature map in the i-th layer is given by

$$v_{ij}^{xyz} = \tanh\left(b_{ij} + \sum_{m}\sum_{p=0}^{P_i-1}\sum_{q=0}^{Q_i-1}\sum_{r=0}^{R_i-1} w_{ijm}^{pqr}\, v_{(i-1)m}^{(x+p)(y+q)(z+r)}\right)$$

The results showed that 3D CNNs out-perform traditional frame-based (2D CNNs) methods and handcrafted spatio-temporal features, particularly in real-world environments with cluttered backgrounds and view-point variations.

Neuroscientific Principles

Convolutional networks are perhaps the greatest success story of **biologically inspired** artificial intelligence.

- Hubel and Wiesel conducted pioneering experiments on the visual cortex of cats and monkeys. They showed that neurons in early visual system responded selectively to specific regions of the visual field.

- We can consider a *very simplified* view of brain function in which we call that early visual system identified **V1**.

Neuroscientific Principles

CNNs reproduce several key properties observed in the **V1**.

- V1 is organized as a **spatial map**. CNNs can capture this property by defining features on two-dimensional maps, where each unit corresponds to a localized region of the input.

- V1 contains many **simple cells**, whose activity can be approximated as a linear function over a small receptive field. The detector units in CNNs are designed to emulate these cells.

- V1 includes also **complex cells** which are insensitive to small shifts in the position of features and sometimes to changes such as lighting conditions. This biological mechanism inspires the use of pooling in CNNs.

Neuroscientific Principles

Although CNNs were inspired by the described visual system, there are many important **differences**.

- Human vision is not uniformly high resolution, CNNs instead, usually process entire images at full resolution in a single pass.

- Human visual system operates in conjunction with other senses, CNNs instead are purely visual system.

- Biological visual system does much more than object recognition.

- In V1, activity is strongly shaped by top-down signals from higher visual areas. Feedback connections in CNNs have not yet produced consistent advantages comparable to those observed in biology.

Conclusions

- CNNs are **a cornerstone of deep learning**: efficient, scalable, biologically inspired
- Exploit **locality, weight sharing, translation equivariance**.
- Applied successfully to **vision, audio, text and volumetric data.**
- **Variants & modern pooling alternatives** improve flexibility and efficiency.
- **Future directions:**
  - Models with foveation and feedback
  - Expansion beyond vision → multimodal & scientific applications