Convolutional Neural Networks (CNNs)

Machine Learning Course

Lecture 2

Prof.ssa Valentina Poggioni
Dott. Cristian Cosci

Convolutional Neural Networks (CNNs)

- CNN is a type of Deep Learning neural network.
- CNN is used to extract the feature from the grid-like matrix dataset.
- CNN performs well for an image or pixel-based data.
- CNNs can capture the spatial information much better than other models.
- CNN performs much better than Feed Forward Network by for Images and Videos.

CNNs are used in:

- Image/Video Analysis
- Classification tasks,
- Computer Vision
- Object detection,
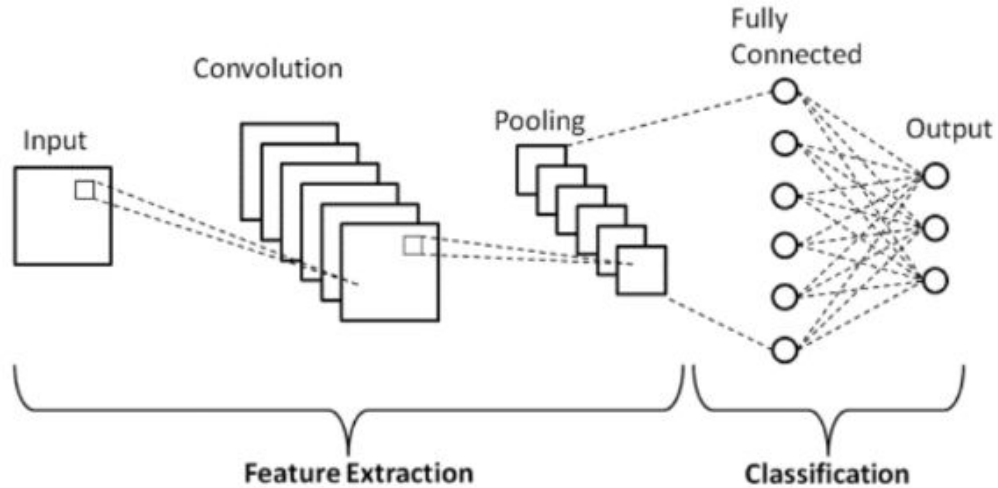- Image segmentation
- Face recognition problem.

CNN architecture:

There are mainly three types of layers:

1. **Convolution Layer**

2. **Pooling Layer**

3. **Fully Connected Layer**

CNN architecture:

1. Convolution Layer

- It's the main building block of CNNs.
- CNN layer is followed by a Non-linear activation function (Tanh, Sigmoid, ReLU)
- ReLU is usually used in CNN models as they deliver the best results without the vanishing/exploding gradient problem.
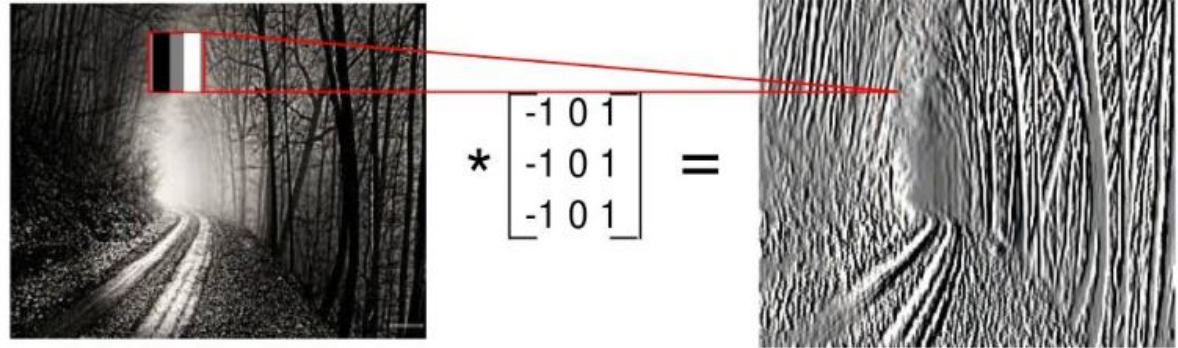
1. Convolution Layer

In this layer, the mathematical operation of convolution is performed between the input image and a filter of a particular size *k x k*.

By sliding the filter over the input image, the dot product is taken between the filter and the parts of the input image with respect to the size of the filter.

6 x 6 x 3   *   3 x 3 x 3   =   4 x 4
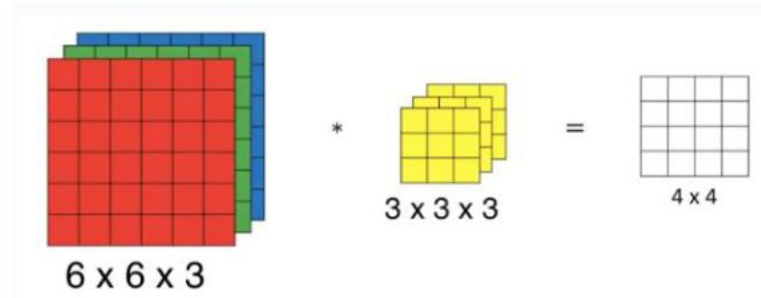
1. Convolution Layer

- The output of a Convolutional layer is called a **feature map**
- If it were like an **edge detector** it would act as a **Sobel** or **Prewitt** filter.

1. Convolution Layer

A Convolutional layer of a CNN acts like **filtering**. The difference wrt conventional filters is that convolutional layer's parameters are **learned**.

During the forward pass, we slide each filter across the width and height of the input and compute dot products between the entries of the filter and the input. Each calculation of a node's value is called a **convolution**
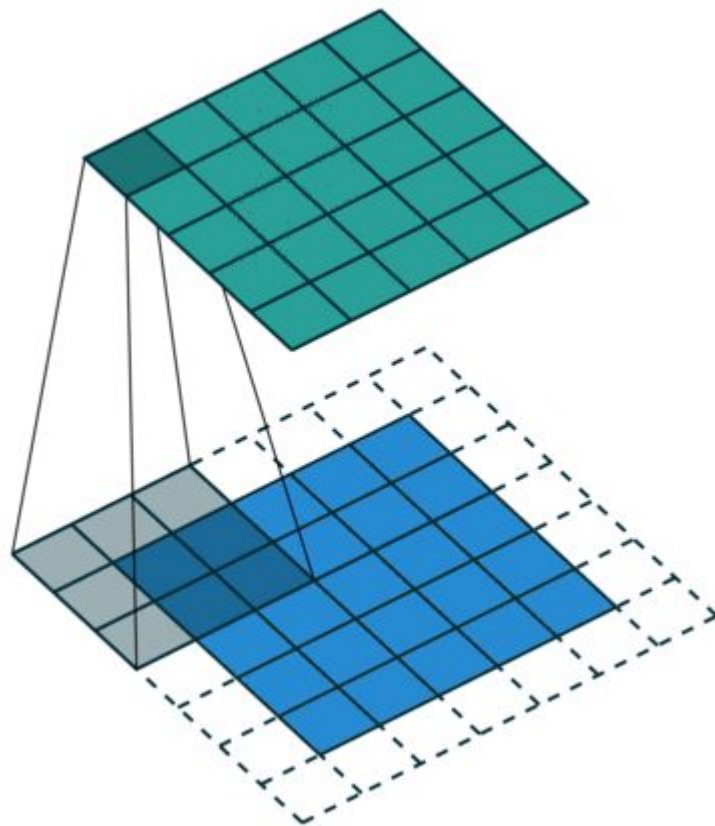
The 2D convolution is given by:

$$o[m, n] = f[m, n] * g[m, n] = \Sigma_u \Sigma_v f[u, v] g[m - u, n - v]$$

1. Convolution Layer

Operations required for a convolution

- To create one element of one output feature *k x k x D* **multiply**-**accumulate** operations are required
- **Computations:** (*k x k x D*) *x [(N - k + 1) / s] x [(N - k + 1) / s] x H*
- **H** feature maps. i.e. H different kernels

1. Convolution Layer

For **the hyperparameters**:

- **n** size of the **input feature map** (eg. *32x32 x3* images)

- **p** is the **padding** (tells how many zeros we pad around the border)

- **f** is the **filter dimension**

- **s** is the **stride** (corresponds to how much we slide the filter)

The output shape of the convolution is
$$\left(\frac{n + 2p - f}{s} + 1 \right) \times \left(\frac{n + 2p - f}{s} + 1 \right)$$
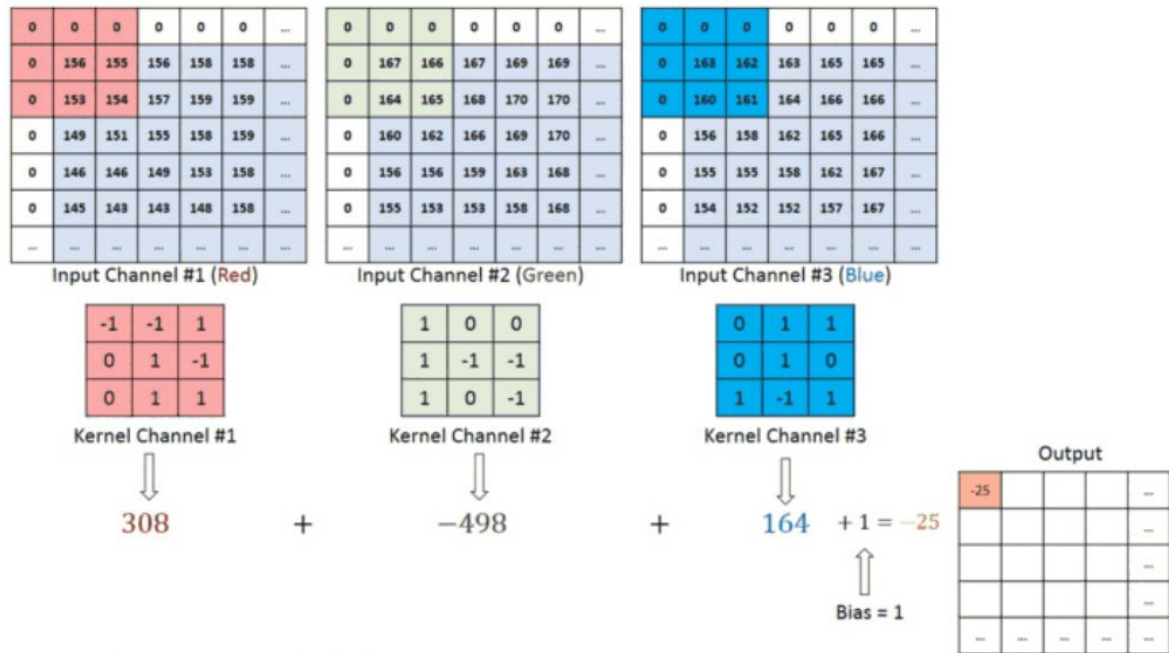
1. Convolution Layer

About Depth D … why many filters?

- **Depth** corresponds to the **number** of **filters** we use for the convolution operation
- To capture different features we must learn multiple filters.
- Different values of the filter matrix will produce different feature maps for the same input image.
- If the depth of the input and the depth of the filter are equal, the output of the convolution is 2D

1. Convolution Layer

- At the input, images are with a **width** and **height** and a **depth** =3 for the RGB channels
- As we slide the filter over the width and height of the input volume we will produce a 2-dimensional activation map that gives the responses of that filter at every spatial position.
- It is a Matrix multiplication performed between K_n and I_n stack and all the results are summed

Input Channel #1 (Red)   Input Channel #2 (Green)   Input Channel #3 (Blue)

Kernel Channel #1   Kernel Channel #2   Kernel Channel #3

308 + -498 + 164 $+ 1 = -25$

Bias = 1

Output

-25

2. Pooling Layer

- The Pooling layer is used to reduce the dimension of the image.
- It is also known as **Downsampling**.
- The pooling Layer also uses a kernel and moves across the image but performs the pooling operation.
- Pooling operation reduces the data within the kernel into a single pixel data.
- It makes the computation fast, reduces memory and also prevents overfitting.
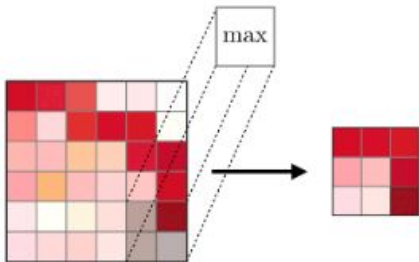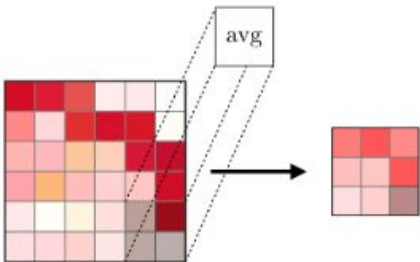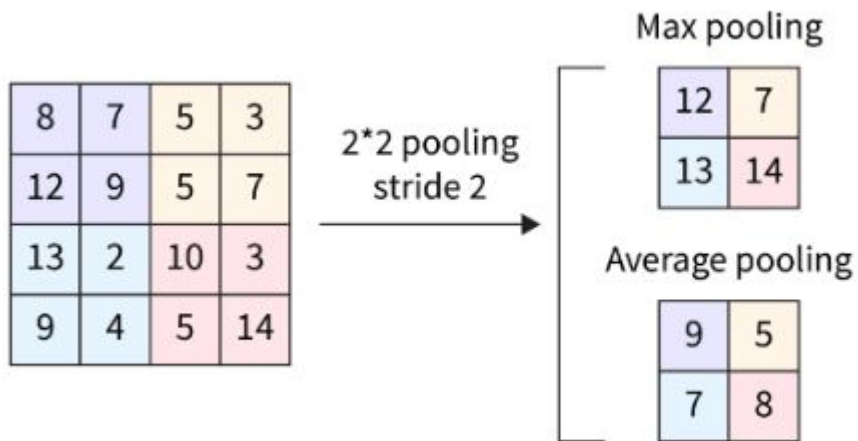
2. Pooling Layer

There are different types of pooling operation. Here we can see 2 examples:

1. **Max Pooling** – This operation outputs the maximum value contained in the kernel. In addition, this pooling performs as a "Noise Suppressant".

2. **Average Pooling** – The average value of the pixels contained in the kernel is returned as the output.

2. Pooling Layer

2. Pooling Layer

Type	Max pooling	Average pooling
Purpose	Each pooling operation selects the maximum value of the current view	Each pooling operation averages the values of the current view
Illustration		
Comments	• Preserves detected features • Most commonly used	• Downsamples feature map • Used in LeNet

# Before the Fully-connected Layer

Flattening:

The resulting feature maps are flattened into a one-dimensional vector after the convolution and pooling layers so they can be passed into a completely linked layer for categorization or regression.

3. Fully Connected Layer

- Fully connected layer(FC layer) is the last stage of the CNN.
- The input FC layer flattens all the images into an array of values.
- It takes the input from the previous layer and computes the final classification or regression task.
- The fully connected layer (FC) operates on a flattened input where each input is connected to all neurons.

3. Fully Connected Layer
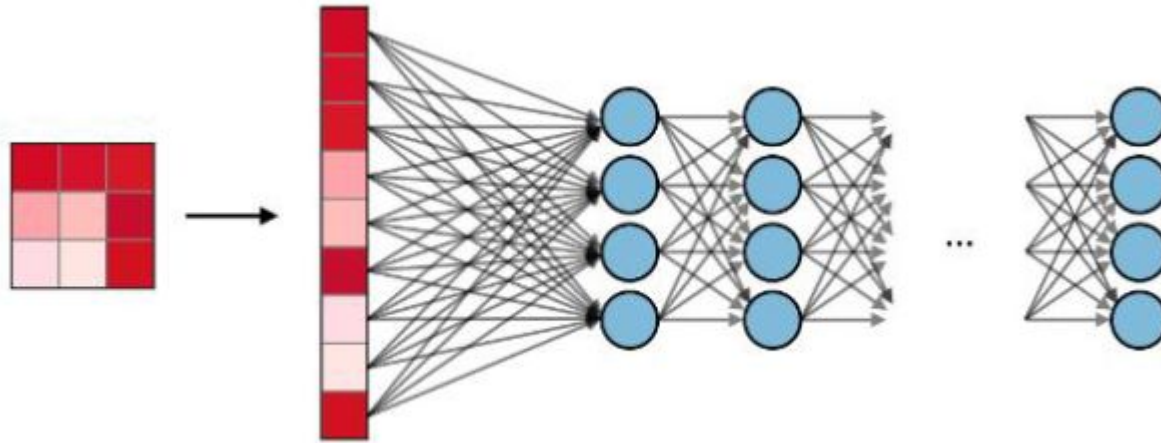
Dropout

- Dropout is a technique where randomly selected neurons are ignored during training.
- Their contribution to the activation of downstream neurons is temporally removed on the forward pass, and any weight updates are not applied to the neuron on the backward pass.
- If neurons are randomly dropped out of the network during training, other neurons will have to step in and handle the representation required to make predictions for the missing neurons.
- **Better generalization and less probability of overfitting**

Advantages of CNNs:

1. Good at detecting patterns and features in images, videos, and audio signals.
2. Robust to translation, rotation, and scaling invariance.
3. End-to-end training, no need for manual feature extraction.
4. Can handle large amounts of data and achieve high accuracy.

Disadvantages of CNNs:

1. Computationally expensive to train and require a lot of memory.
2. Can be prone to overfitting if not enough data or proper regularization is used.
3. Requires large amounts of labeled data.
4. Interpretability is limited, it's hard to understand what the network has learned.

Parameter Sharing

- Share parameters over multiple image locations
- Parameter sharing in CNN makes it translation invariant: i.e., we can find a cat with the same cat detector whether the cat appears at column *i* or column *i+1* in the image
- Parameter sharing in CNN also dramatically lowers the model parameters, and significantly increases network sizes without requiring a corresponding increase in training data

CNN Architectures: LeNet, AlexNet, ZFNet, GoogleNet, VGG and ResNet

Year	CNN	Developed by	Place	Top-5 error rate	No. of parameters
1998	LeNet(8)	Yann LeCun et al			60 thousand
2012	AlexNet(7)	Alex Krizhevsky, Geoffrey Hinton, Ilya Sutskever	1st	15.3%	60 million
2013	ZFNet()	Matthew Zeiler and Rob Fergus	1st	14.8%	
2014	GoogLeNet(19)	Google	1st	6.67%	4 million
2014	VGG Net(16)	Simonyan, Zisserman	2nd	7.3%	138 million
2015	ResNet(152)	Kaiming He	1st	3.6%	