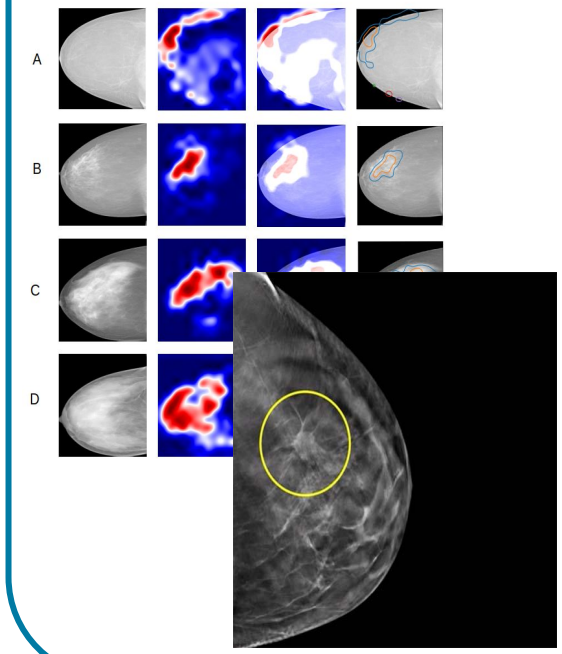
Classification of lesions in Medical Physics

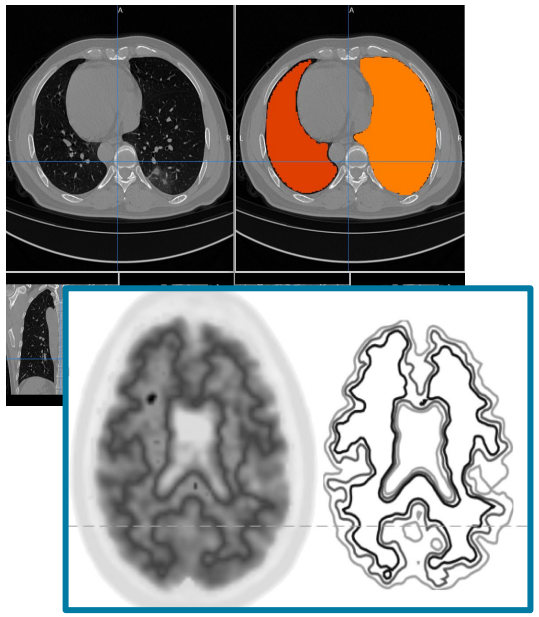
Francesca Lizzi, INFN Pisa
francesca.lizzi@pi.infn.it
9/12/2024

CNN and medical images

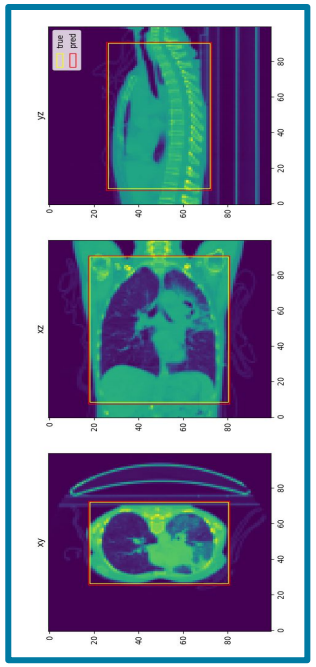
Classificazione:



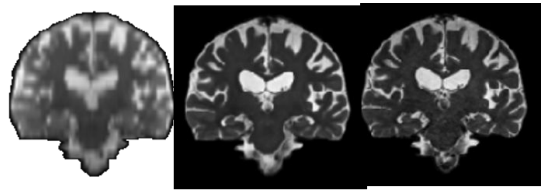
Segmentazione:

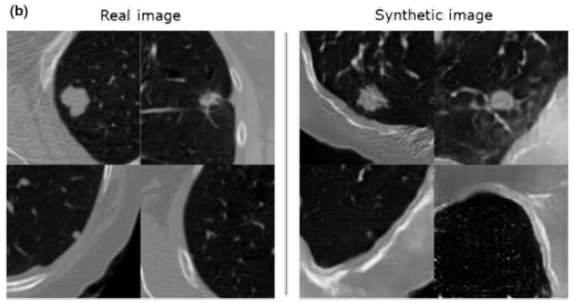


Regressione:



Super Risoluzione:

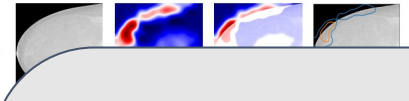




- **Generazione di nuovi dati (con qualche limite)**

CNN e immagini mediche

Classificazione:



Segmentazione:

Regressione:

Super Risoluzione:

Oggi guardiamo meglio le reti più usate per l'analisi delle immagini:
le **reti neurali convoluzionali** allenate con paradigma **supervisionato**

- **Generazione di nuovi dati (con qualche limite)**

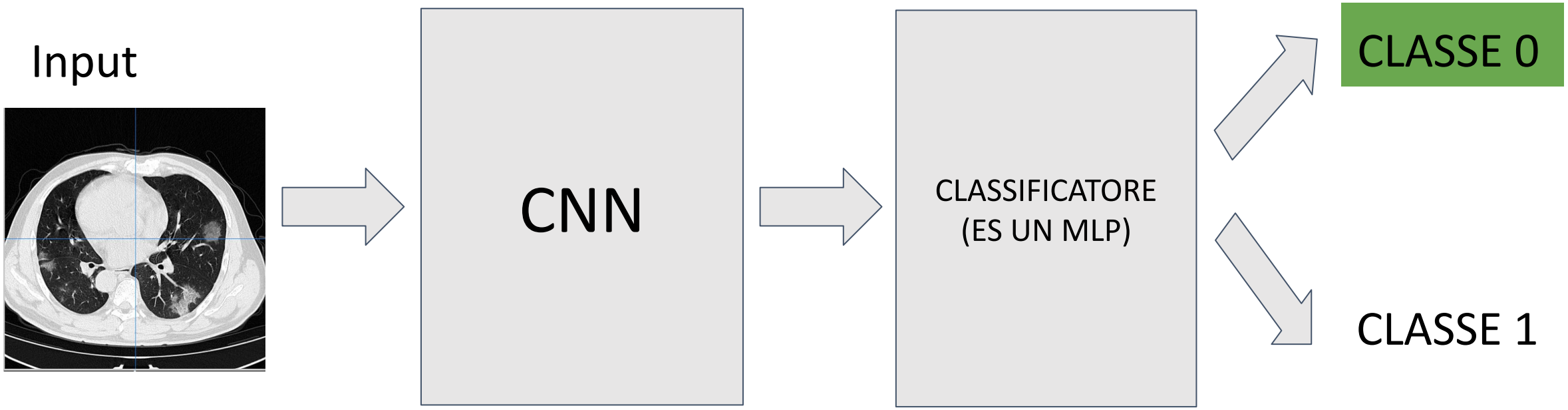
Paradigma supervisionato

IDEA BASE

diamo alla CNN, molti esempi su cui imparare, cioè coppie di immagini e label.

Esempio: classificatore binario

INPUT: è l'immagine che viene processata, supponiamo che appartenga alla classe 0



Come facciamo a dirgli che è corretto?

Definiamo una **funzione di costo (*loss function*)** che confronta la ***label* (cioè il valore vero, Y_i)** con il valore predetto, $\hat{Y}_i$.

Esempio, MSE:

più il valore predetto è simile al valore vero,

più questa funzione andrà a zero.

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}\left(Y_i - \hat{Y}_i\right)^2$$

Quello che voglio fare quindi è **MINIMIZZARE** la funzione di costo. Come?

Con un **ottimizzatore:** qualcosa che è in grado di "modificare" la mia rete neurale al fine di rendere la loss function più piccola possibile:

TROVARE UN MINIMO DELLA LOSS FUNCTION

Esempio: **Stochastic Gradient Descent** ➡ devo calcolare il gradiente della funzione di costo rispetto ai pesi della rete... come faccio?

Algoritmo di backpropagation

Applicando l'algoritmo di *backpropagation* che applica, in maniera efficiente, la **"regola della catena"** per il calcolo di derivate composte.

La minimizzazione della funzione di costo avviene tramite l'algoritmo di **BACKPROPAGATION** che usa la **CHAIN RULE** per calcolare le derivate della funzione di costo rispetto ad ogni peso della rete neurale.

IMPARARE SIGNIFICA CALCOLARE TANTE DERIVATE!

Quante? Dipende da quanti layer e da quanti neuroni ha la rete

Questo implica che le derivate più vicine all'input, saranno calcolate moltiplicando tante derivate.

Architettura e capacità di una rete neurale

Il **numero di layer**, il **numero di neuroni** per ogni layer, le **operazioni** che mettono in **relazione i layer** definiscono **l'*architettura*** della rete neurale.

Letteralmente la "struttura".

Un concetto operativo utile è la **capacità**

CAPACITÀ: definita dal numero di neuroni in un layer e dal numero di layer. È una grandezza "informale" ed è la capacità di una rete neurale di usare l'informazione in maniera efficace. **Più neuroni, più layer -> maggiore capacità.**

Evoluzione delle reti neurali convoluzionali: una breve storia

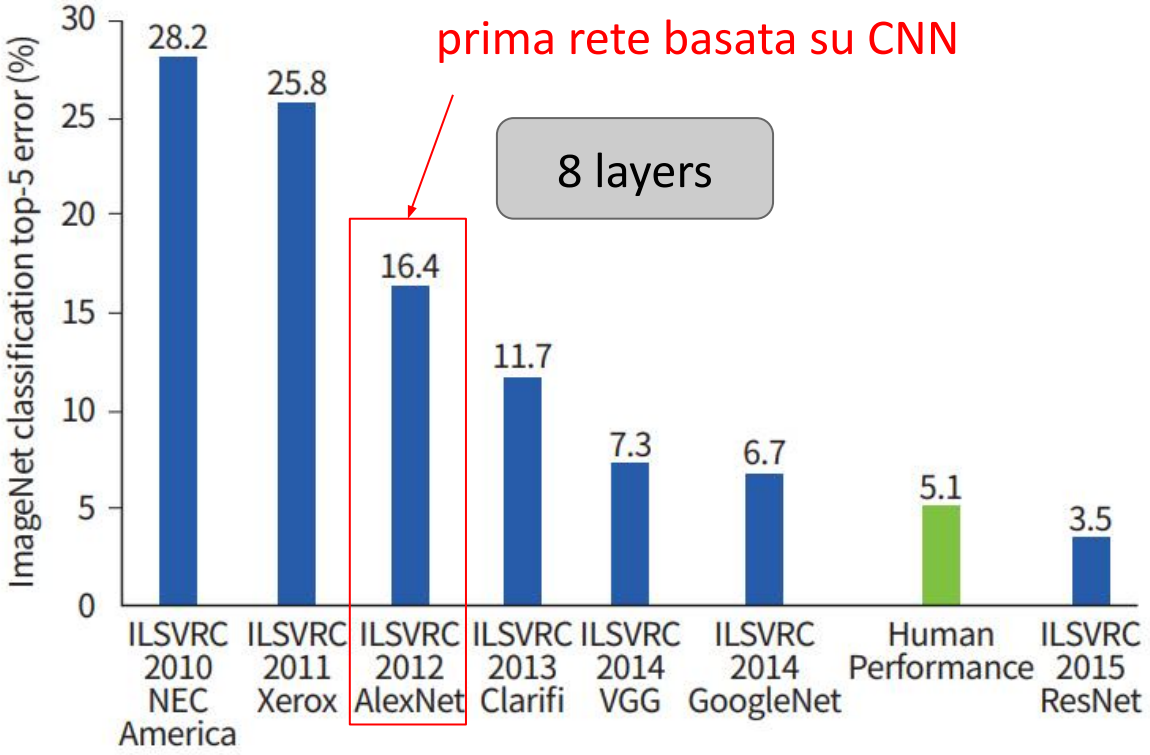
A partire dal 2012 inizia a prendere sempre più piede l'idea che le reti neurali possano aiutare a risolvere problemi molto complessi. Perché nel 2012?

- Grande quantità di dati a disposizione
- Hardware ottimizzato per il calcolo parallelo
- Soluzioni software disponibili per l'allenamento delle reti

→ Maturità tecnologica e teorica sufficiente per usarle su problemi di larga scala

Nel 2012, AlexNet vinse l' *ImageNet Large Scale Visual Recognition Competition (ILSVRC)* classificando correttamente 83.6 % del dataset di ImageNet (about $1.3 \cdot 10^6$ images belonging to 1000 classes).

Reti convoluzionali diventato lo stato dell'arte



Le reti convoluzionali diventano lo stato dell'arte in moltissimi (quasi tutti) campi della *Computer Vision.*

Dal 2012 in poi ImageNet viene vinta **SEMPRE** da reti convoluzionali profonde!

Evoluzione della tecnologia software

Quali passaggi e miglioramenti sono stati fatti affinché si potesse raggiungere una tale accuratezza?

Una rete neurale convoluzionale (CNN) è una rete neurale capace di processare dati strutturati. La convoluzione è definita:

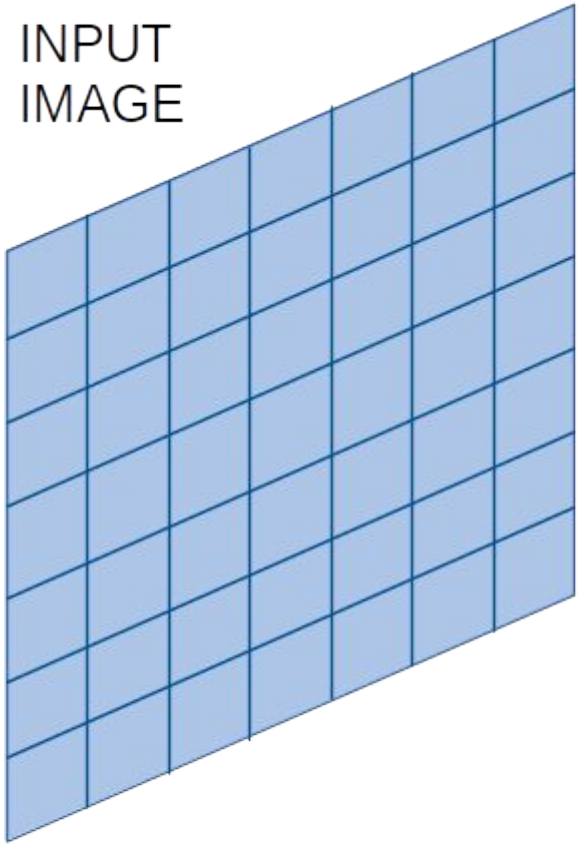
$$C_{AH} = A \otimes H = \sum_{p=0}^{k-1} \sum_{q=0}^{k-1} A(i-p, j-q) H(p, q)$$

dove A è una matrice M · N, H è una matrice quadrata k · k e k un intero dispari.

H si chiama *filtro* o *kernel*. La dimensione di H si chiama *campo recettivo*

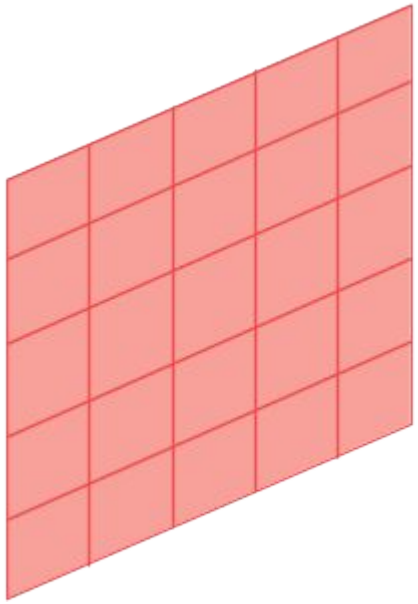
CNN: vedere la convoluzione

INPUT
IMAGE

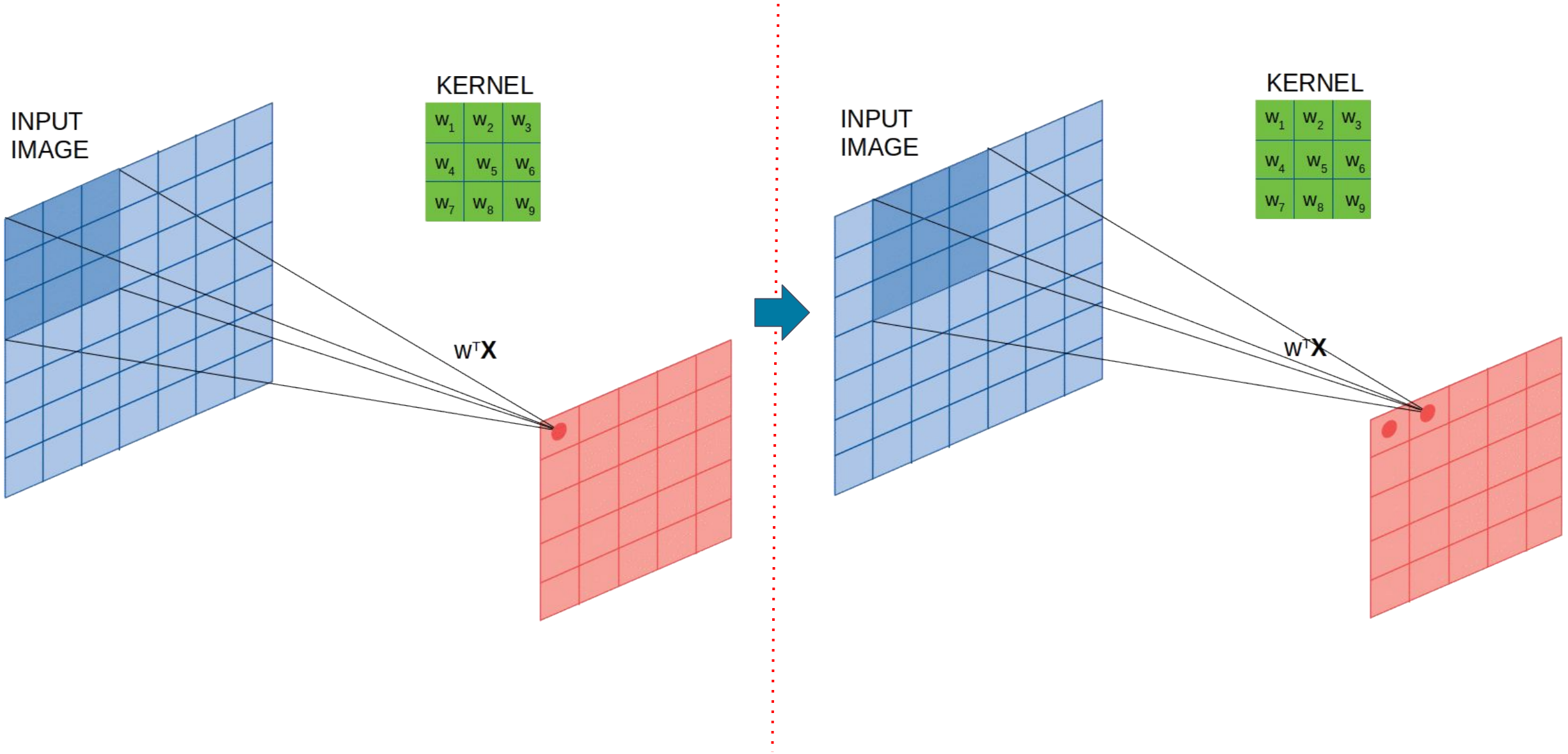


KERNEL

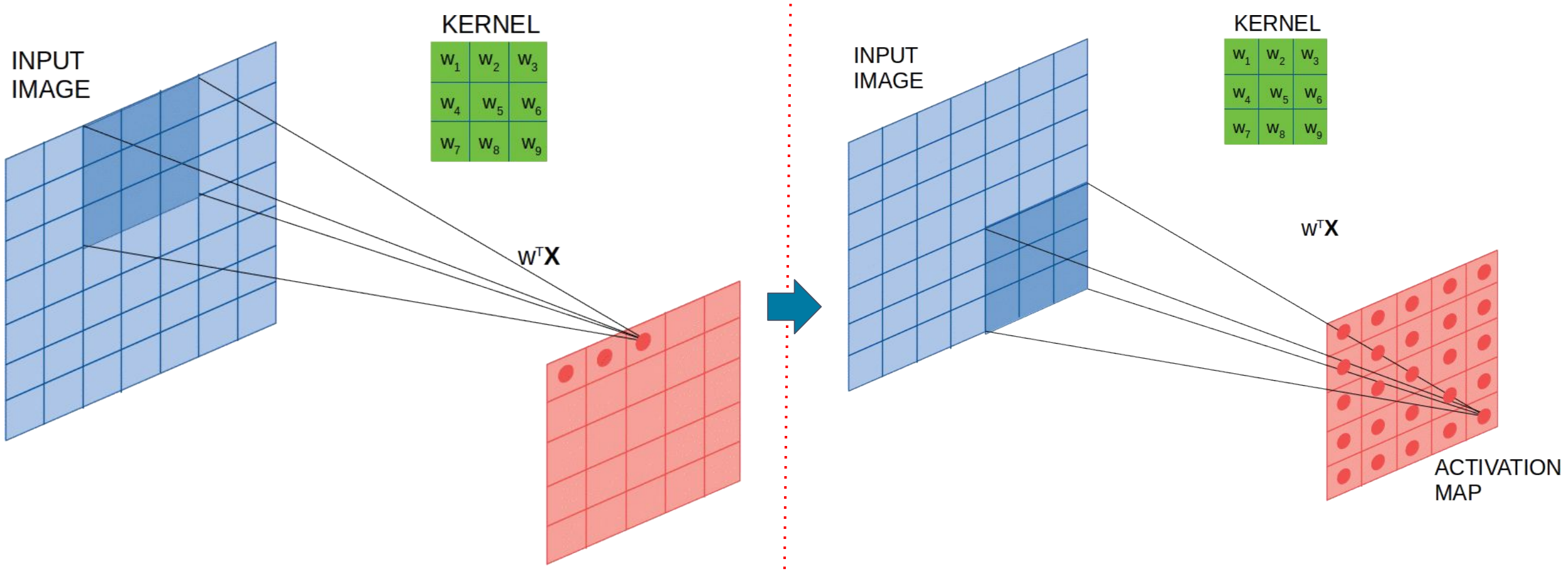
w_1	w_2	w_3
w_4	w_5	w_6
w_7	w_8	w_9



CNN: vedere la convoluzione



CNN: vedere la convoluzione



Il risultato dell'operazione di convoluzione tra kernel e immagine si chiama **MAPPA DI ATTIVAZIONE**

CNN: perché hanno inciso tanto nella *computer vision*?

Prima: **ingegnerizzazione delle features**.

Trovare analiticamente features in grado di rappresentare l'informazione necessaria alla risoluzione di un determinato problema è complicato.

Una CNN è capace di **esplorare lo spazio delle rappresentazioni da sola.**

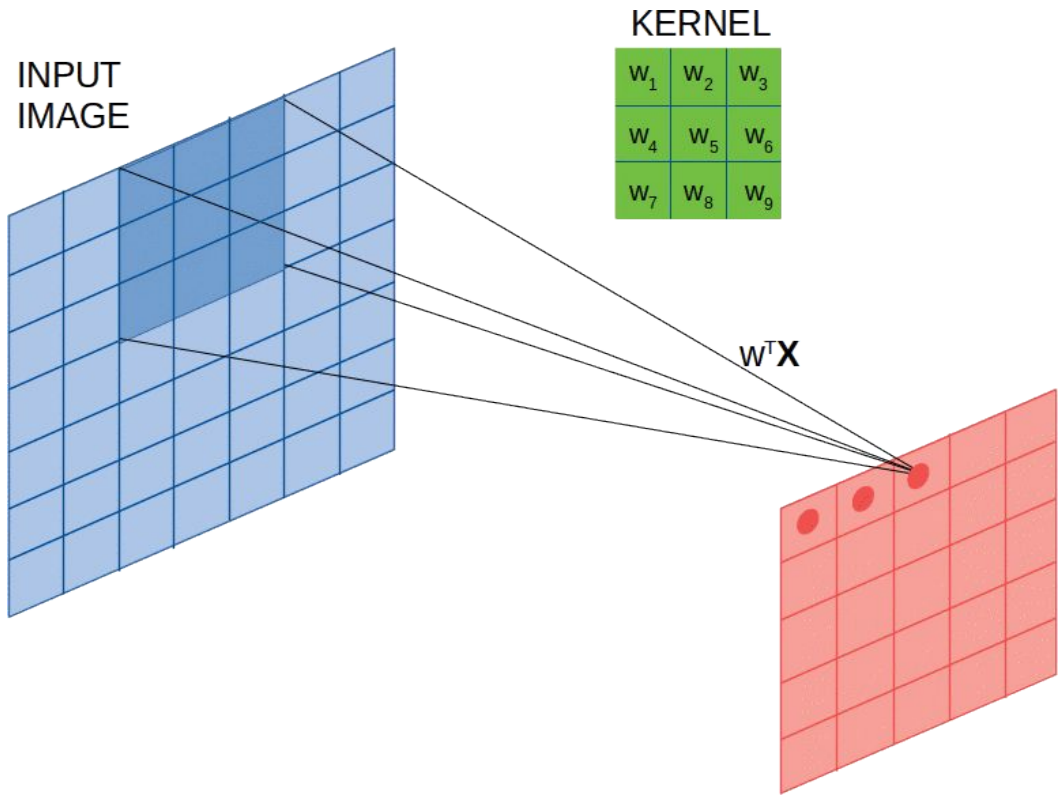
Attraverso il processo di apprendimento, una CNN modifica il valore degli elementi nei kernel per trovare la migliore rappresentazione dei dati in grado di risolvere il problema.

I kernel sono i "veri" neuroni di una CNN!

L'uso di kernel e convoluzione è risultato vincente per 3 motivi:

1. **connettività sparsa;**
2. **condivisione dei parametri;**
3. **invarianza per traslazione.**

CNN: 3 principi fondamentali

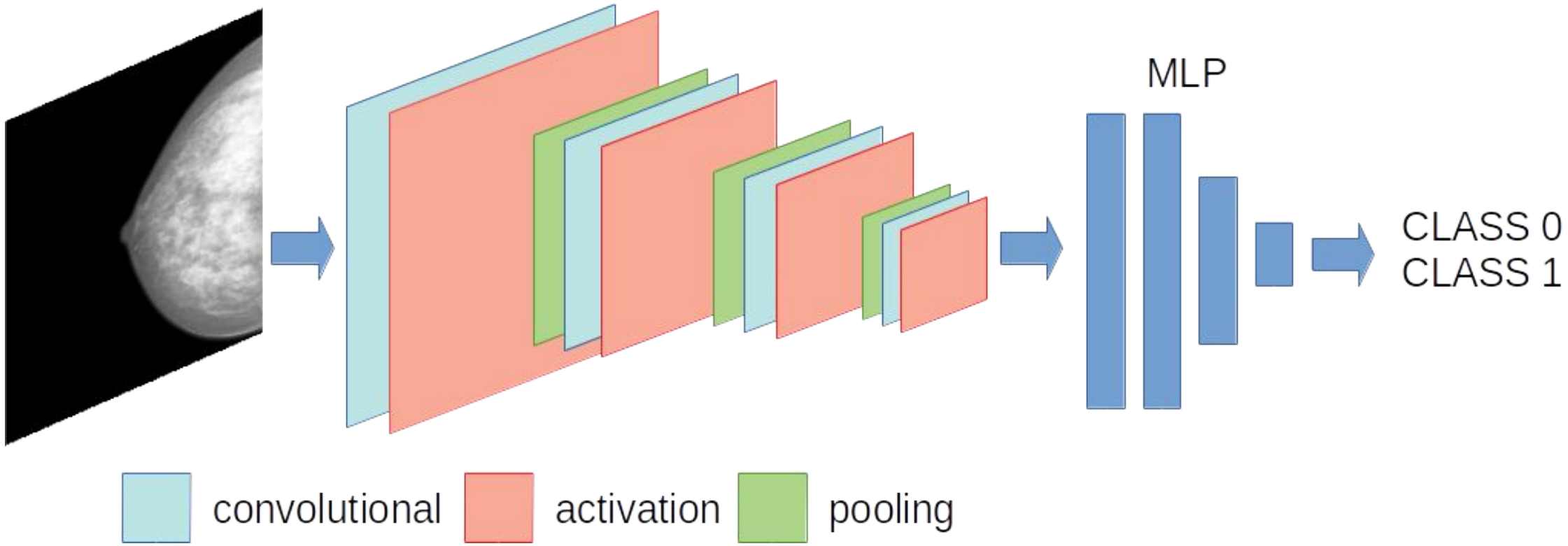


1) I kernel scorrono sull'immagine in maniera che siano **connessi solo ad una piccola parte** dell'input -> connettività sparsa

2) Tramite la convoluzione, invece di imparare un set di parametri per ogni posizione, **imparano un solo set** -> condivisione dei parametri

3) In questo modo, una caratteristica viene riconosciuta a **prescindere dalla sua posizione** nell'immagine -> invarianza per traslazioni

Struttura di una CNN standard



- L’attivazione è una funzione non lineare, ad es. SIGMOIDE
- Il pooling serve a comprimere l’informazione (diminuire drasticamente la dimensione delle mappe di attivazione).

Funzione di attivazione

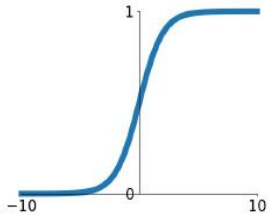
Abbiamo bisogno di un altro ingrediente fondamentale:

la non linearità.

La **non linearità** è importante perchè ci permette di avere rappresentazioni (linearmente) indipendenti tra di loro.

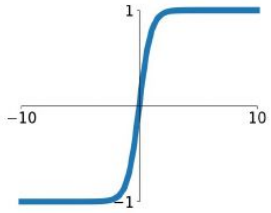
Sigmoid

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



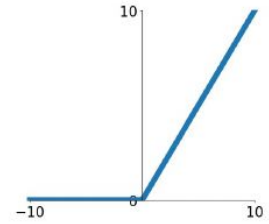
tanh

$$\tanh(x)$$



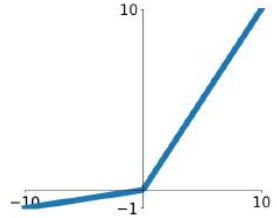
ReLU

$$\max(0, x)$$



Leaky ReLU

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

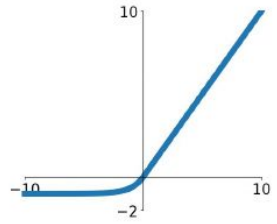


Maxout

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

ELU

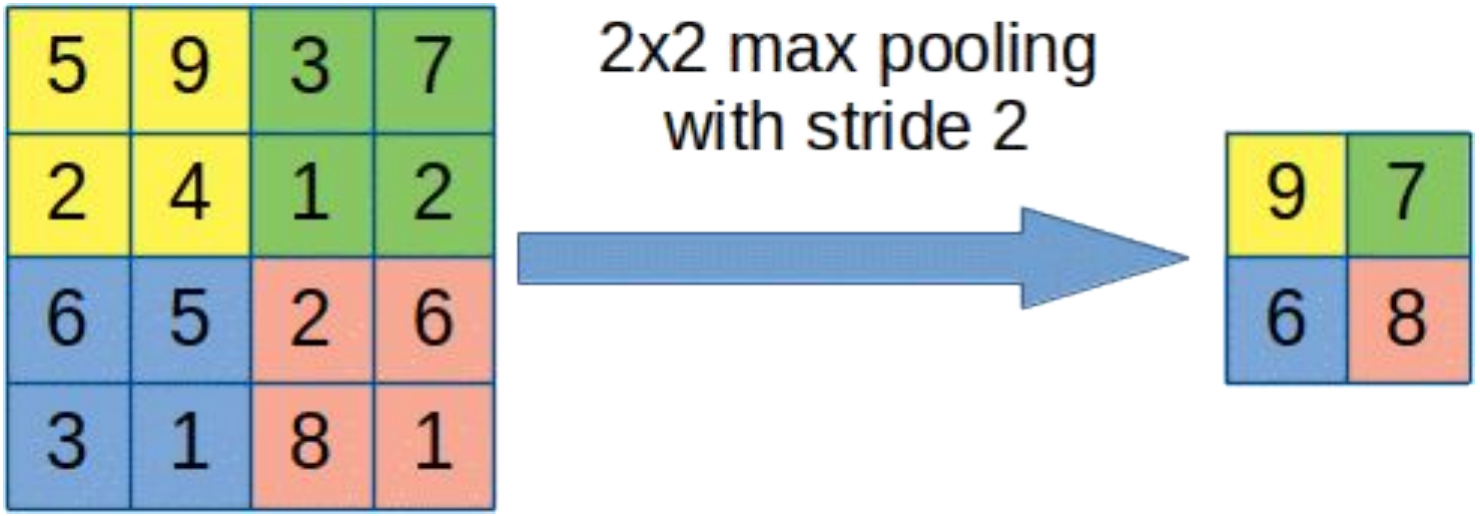
$$\begin{cases} x & x \geq 0 \\ \alpha(e^x - 1) & x < 0 \end{cases}$$



Pooling e compressione immagini

Abbiamo bisogno di **manipolare l'informazione fino a renderla qualcosa di molto piccolo**. In un classificatore binario, nella probabilità di appartenere ad una delle 2 classi, cioè in un singolo numero!

Ci sono diversi modi per farlo, ad esempio il Pooling.
È un'operazione invariante per permutazione, ad esempio un massimo.



Costo computazionale

Le CNN (NN in generale) hanno 2 tipi di parametri:

IPERPARAMETRO: è un parametro di una NN **NON allenabile**, deciso a priori: es. il # di filtri, la loro dimensione ecc ecc.

PARAMETRO: è un parametro di una NN **allenabile**, cioè è un parametro che cambia durante l'allenamento.

Il numero di parametri fa aumentare il costo computazionale.
Più parametri -> più calcolo
La dimensione dei dati fa aumentare la necessità di RAM della GPU.
La batch size fa aumentare il costo computazionale (RAM di sistema, RAM GPU).

Il TEMPO di calcolo dipende anche da tutte queste cose

Avanzamenti tecnologici delle CNN

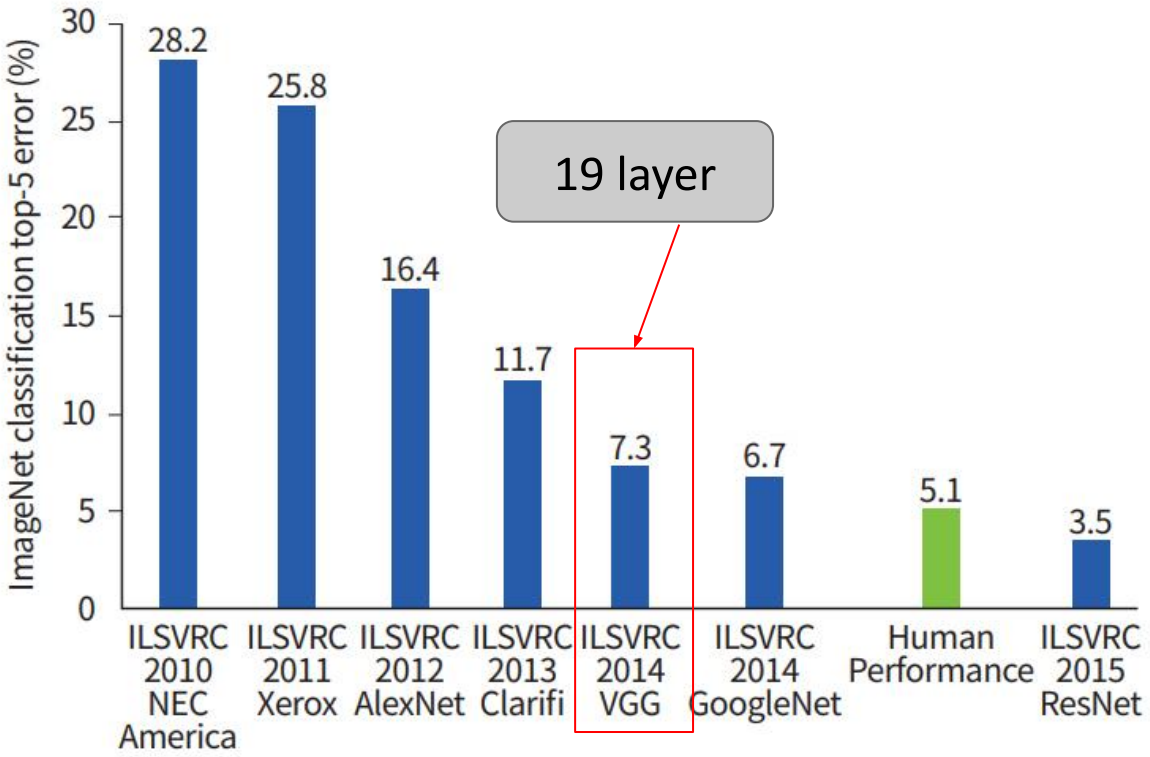
La CNN che abbiamo visto prima è molto simile alla famosa ALEXNET.

Perchè AlexNet è così famosa?

Alex Krizhevsky et al. misero insieme una serie di lavori precedenti:

1) Il **neocognitrone**, proposto da Kunihiko Fukushima nel 1979;
2) Algoritmo di **backpropagation**, termine coniato da Rosenblatt nel 1962 ed evolutosi moltissimo tra gli anni ‘70 e i primi 2000;
3) **LeNet** , proposta da Yan LeCunn nel 1998;
4) **Accelerazione con GPU** proposta per la prima volta da Kumar Chellapilla;
5) Introduce una funzione di attivazione “nuova” -> **ReLU**, saturazione dell’attivazione più difficile.
6) Data augmentation ecc ecc…

Da AlexNet a VGG19



Nel 2014 vince la competizione di ImageNet una rete chiamata VGG19

Karen Simonyan and Andrew Zisserman. “Very deep convolutional networks for large-scale image recognition”. International Conference on Learning Representations (ICLR), 2014.

VGG19 e differenze con AlexNet

In un primo momento la tendenza era quella di creare **reti più profonde.**

AlexNet convolutional layers -> VGG19 **19** convolutional layers

Campo recettivo efficace:

AlexNet: contempla l'utilizzo di filtri di convoluzione di **diverse dimensioni** (dal più grande al più piccolo)

-11x11

- 5x5

-3x3

e poi 3x3 in tutta l'architettura.

Calcoliamo il numero di parametri per il layer 11x11:

11x11x32 (# filtri) = 3872

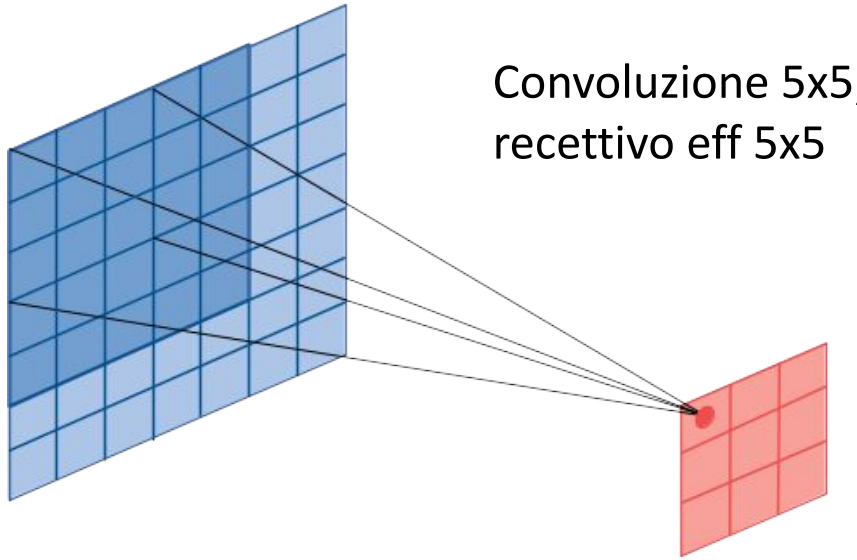
VGG19: invece di usare filtri "grandi" (11x11 o 5x5) ne usa **piccoli in sequenza** prima di fare il pooling. Cinque filtri 3x3 in sequenza hanno un campo recettivo efficace 11x11.

Calcoliamo il numero dei parametri:

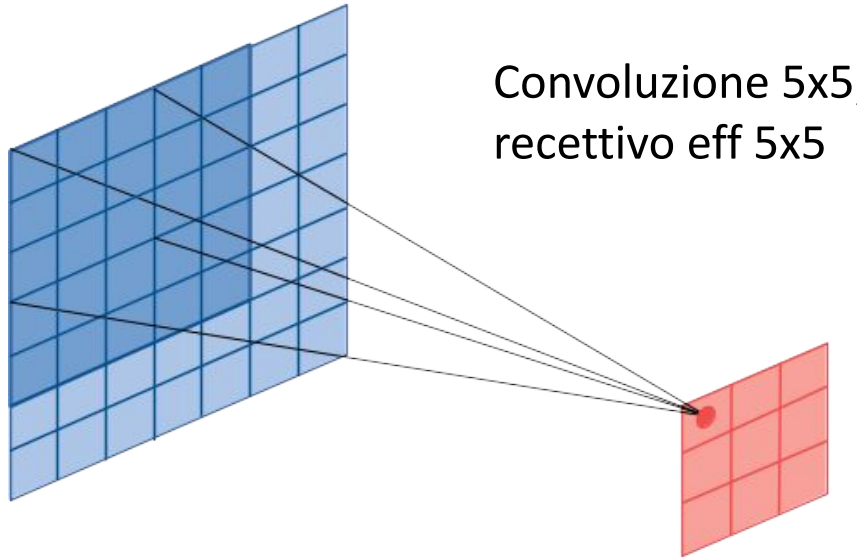
3x3x32 (# filtri)x 5 (# layer) = 1440

Si può quindi andare più in profondità riducendo il numero di parametri "imparabili" ("trainabili")

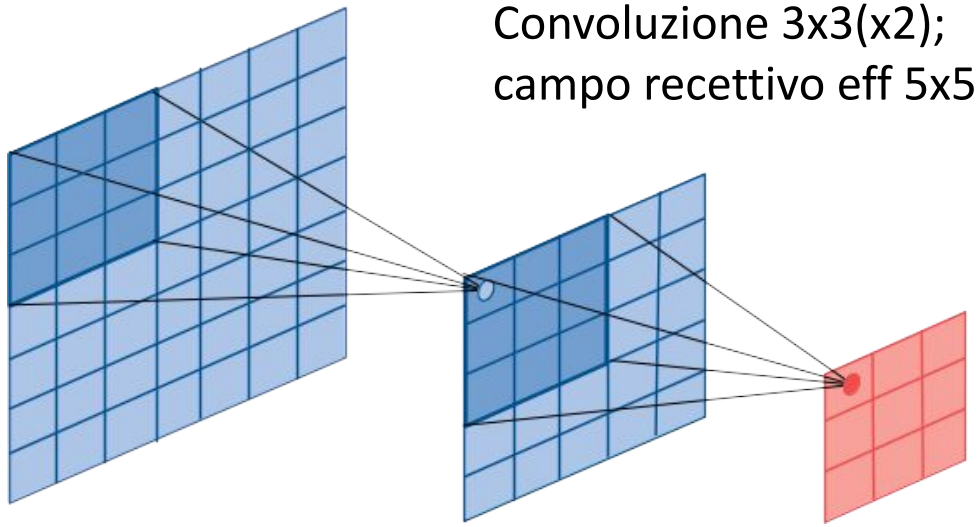
Campo recettivo efficace: 5x5 vs 3x3(x2)



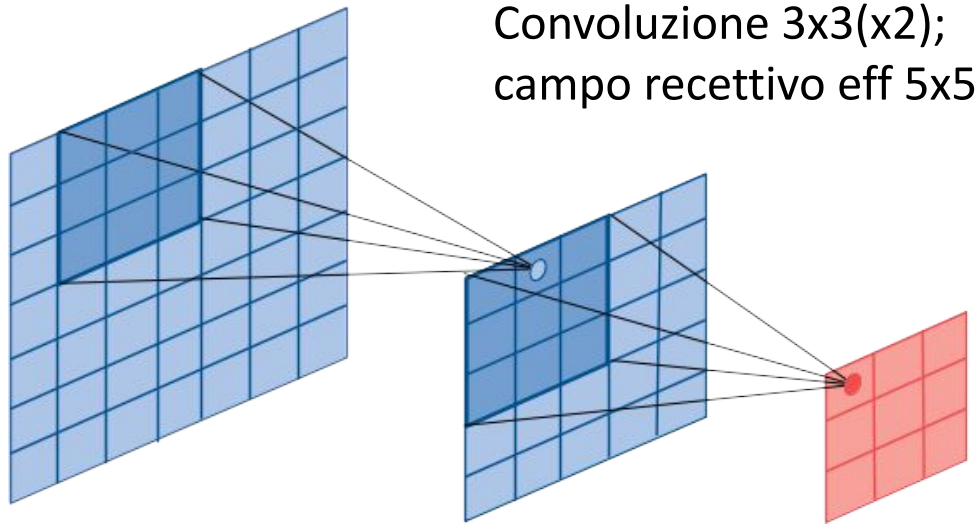
Convoluzione 5x5; campo recettivo eff 5x5



Convoluzione 5x5; campo recettivo eff 5x5

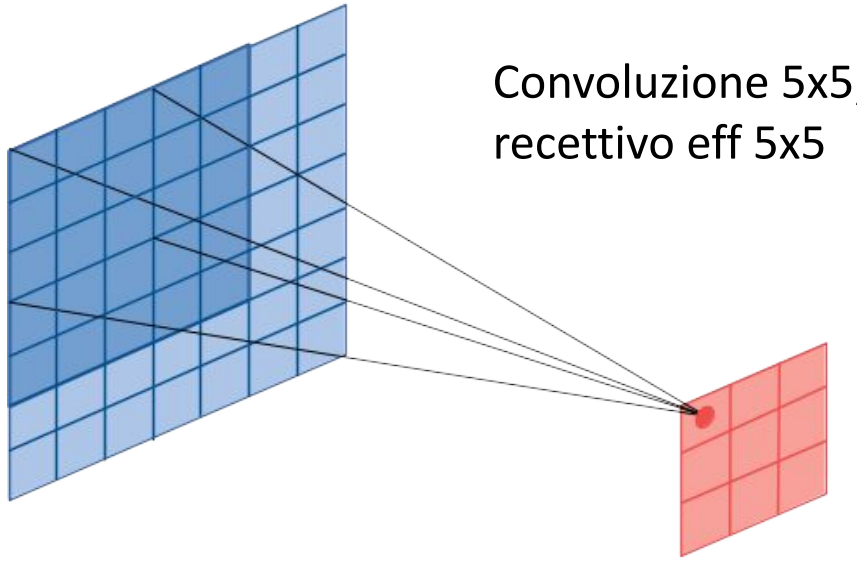


Convoluzione 3x3(x2); campo recettivo eff 5x5

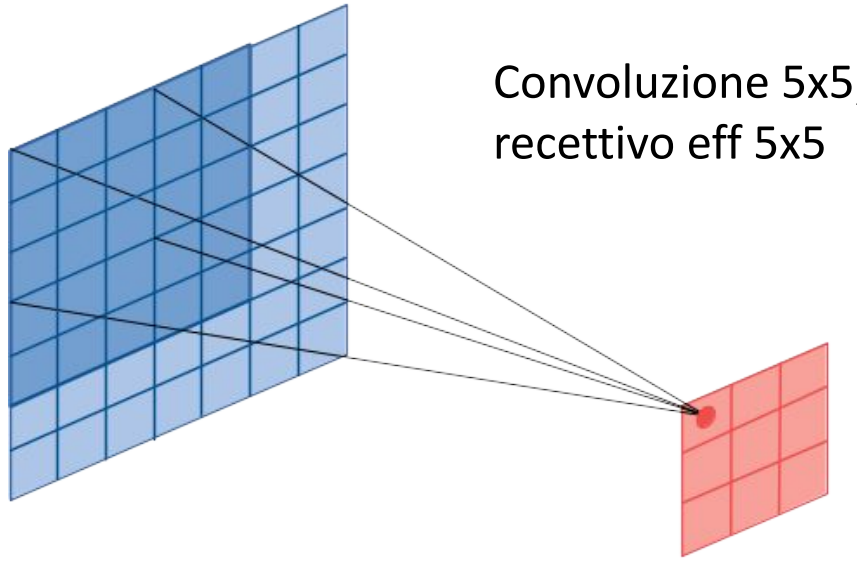


Convoluzione 3x3(x2); campo recettivo eff 5x5

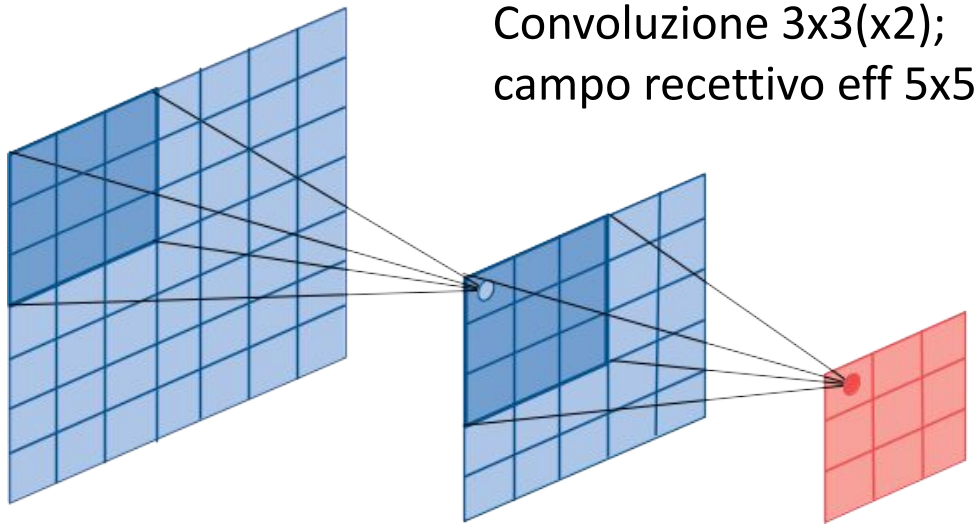
Campo recettivo efficace: 5x5 vs 3x3(x2)



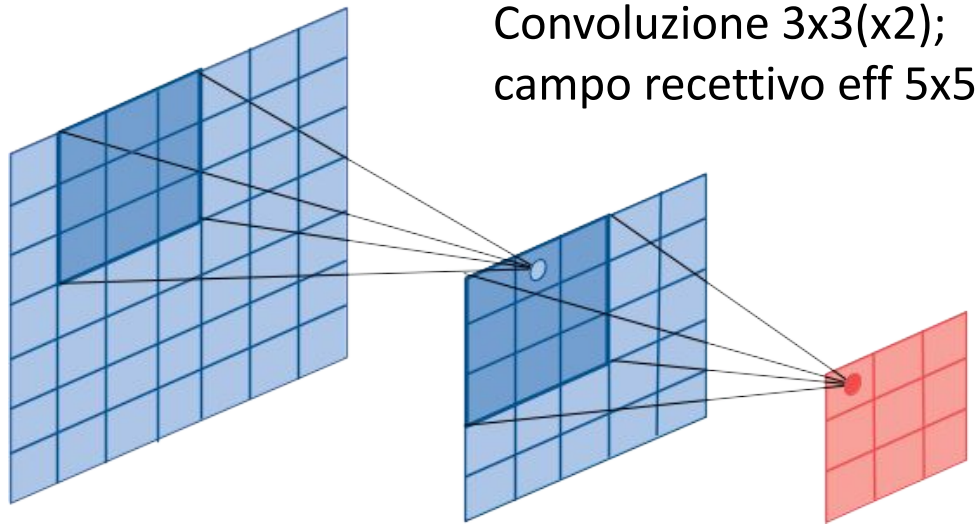
Convoluzione 5x5; campo recettivo eff 5x5



Convoluzione 5x5; campo recettivo eff 5x5

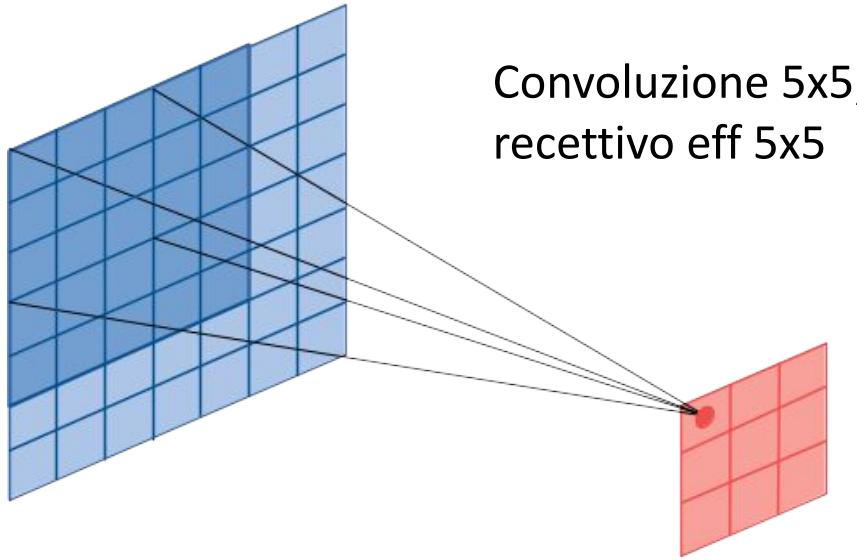


Convoluzione 3x3(x2); campo recettivo eff 5x5

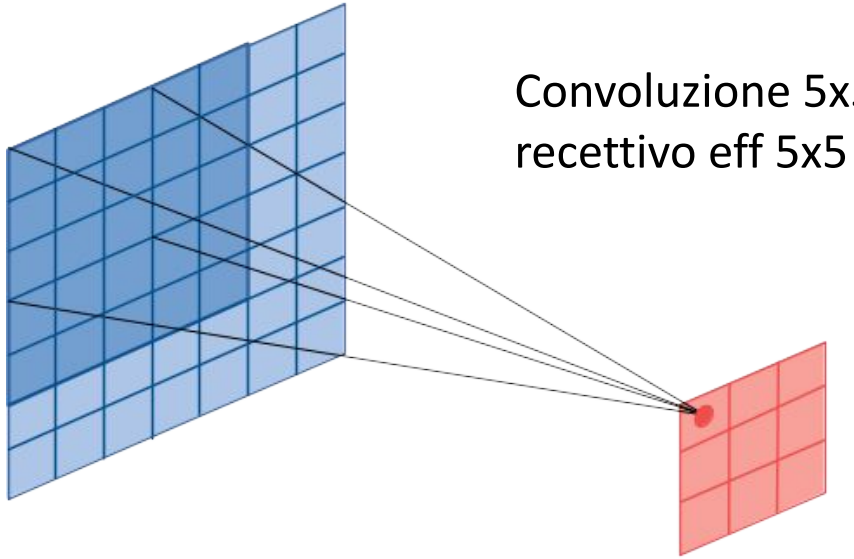


Convoluzione 3x3(x2); campo recettivo eff 5x5

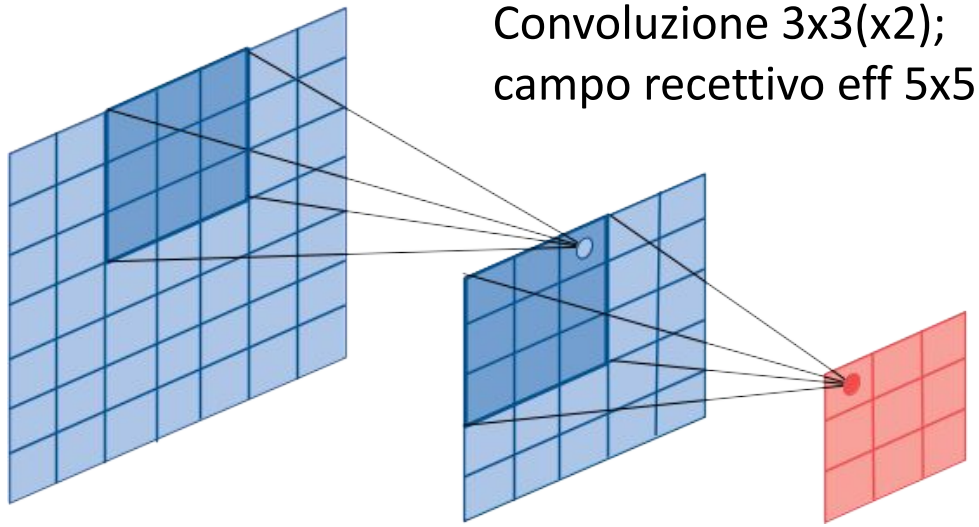
Campo recettivo efficace:



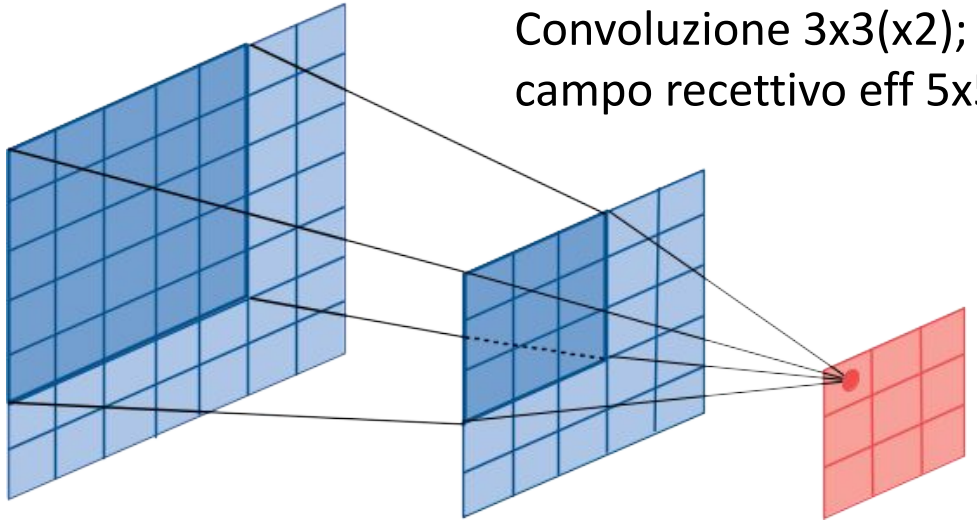
Convoluzione 5x5; campo recettivo eff 5x5



Convoluzione 5x5; campo recettivo eff 5x5

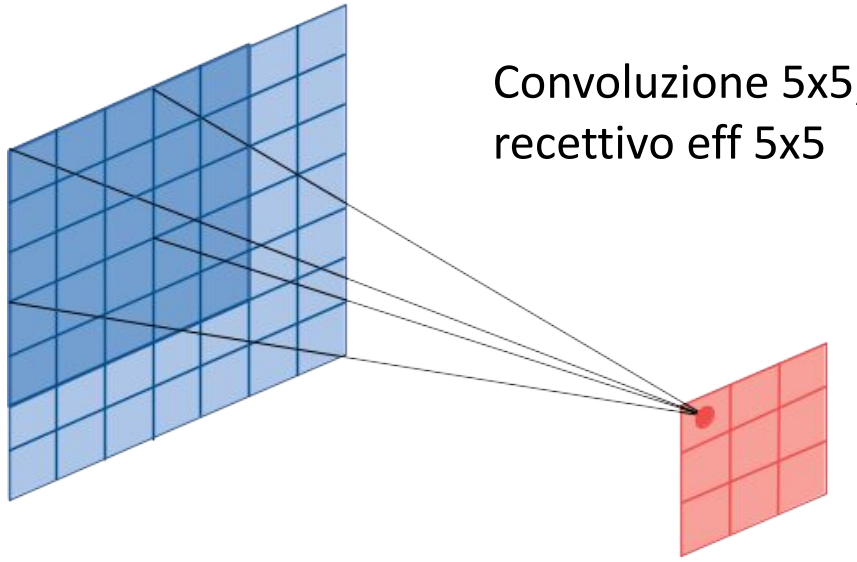


Convoluzione 3x3(x2); campo recettivo eff 5x5

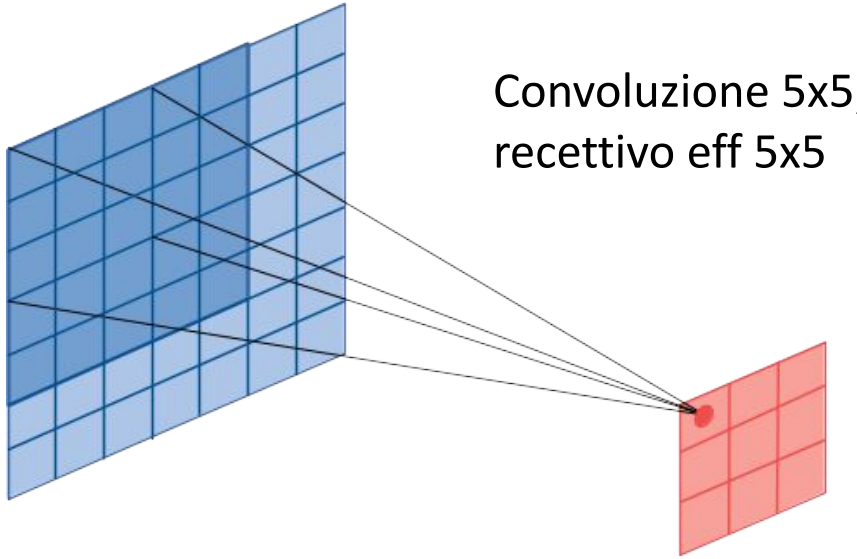


Convoluzione 3x3(x2); campo recettivo eff 5x5

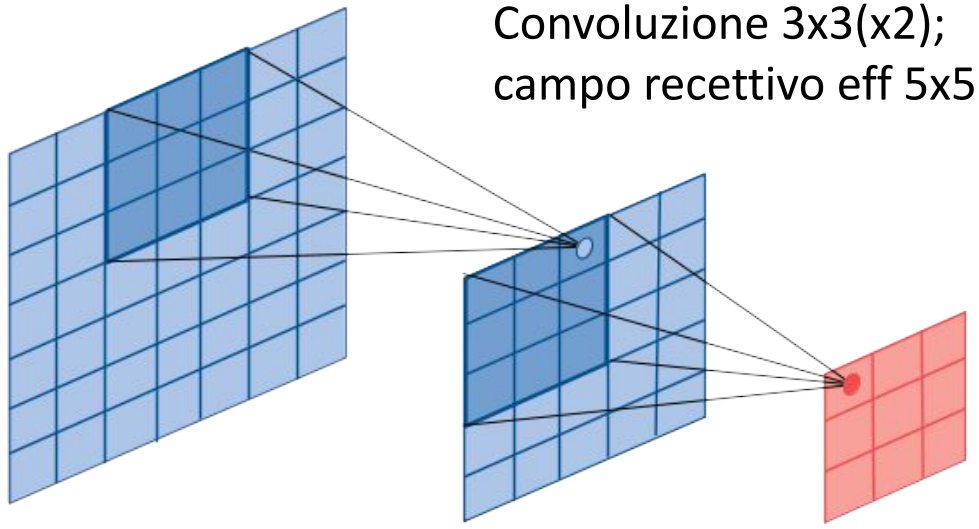
Campo recettivo efficace:



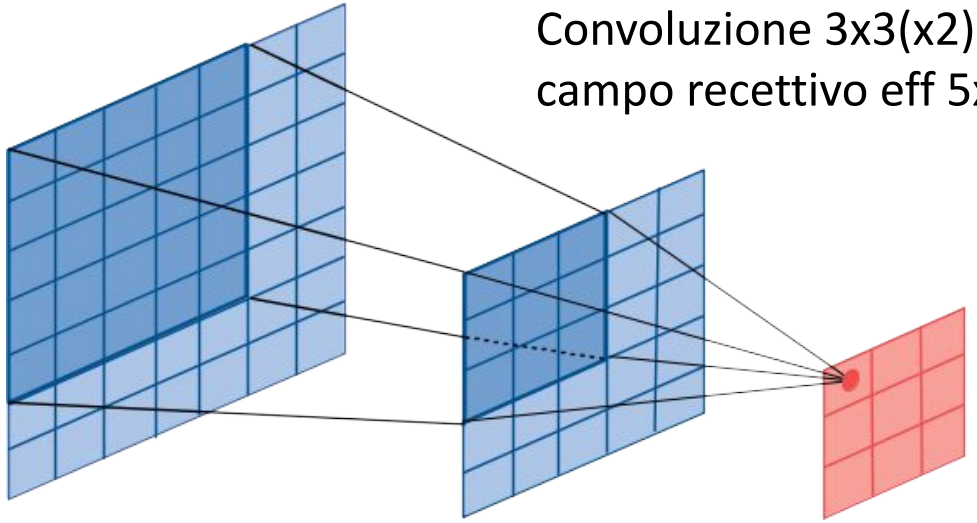
Convoluzione 5x5; campo recettivo eff 5x5



Convoluzione 5x5; campo recettivo eff 5x5



Convoluzione 3x3(x2); campo recettivo eff 5x5



Convoluzione 3x3(x2); campo recettivo eff 5x5

Campo recettivo efficace:

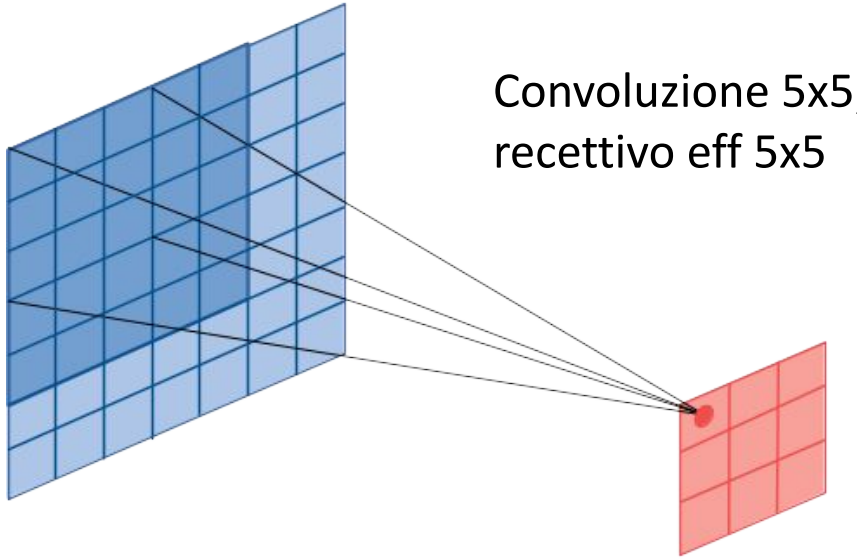
Calcoliamo il numero di parametri nei due casi:

5x5x32 = 800

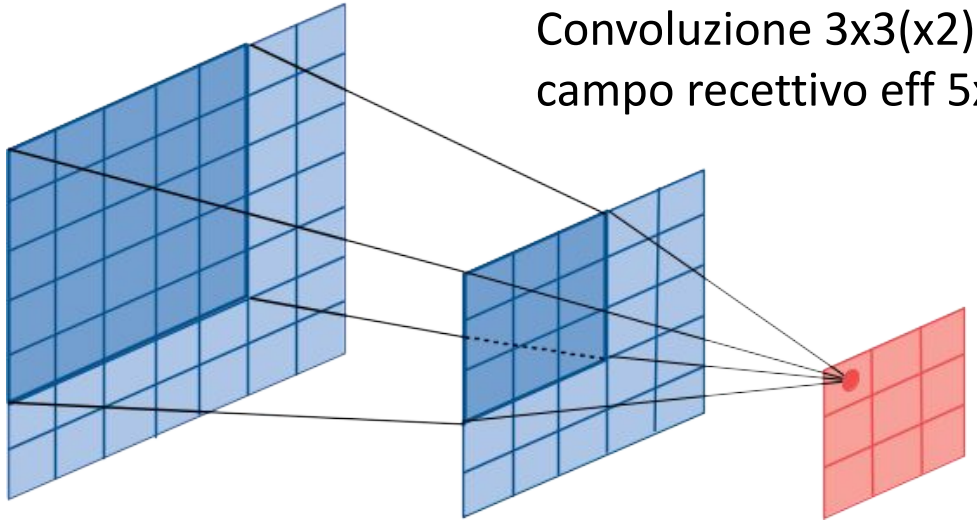
3x3x32x2 = 576

MA

GUARDANO (e.g. cercano correlazioni) **gli stessi punti dell'immagine in input**

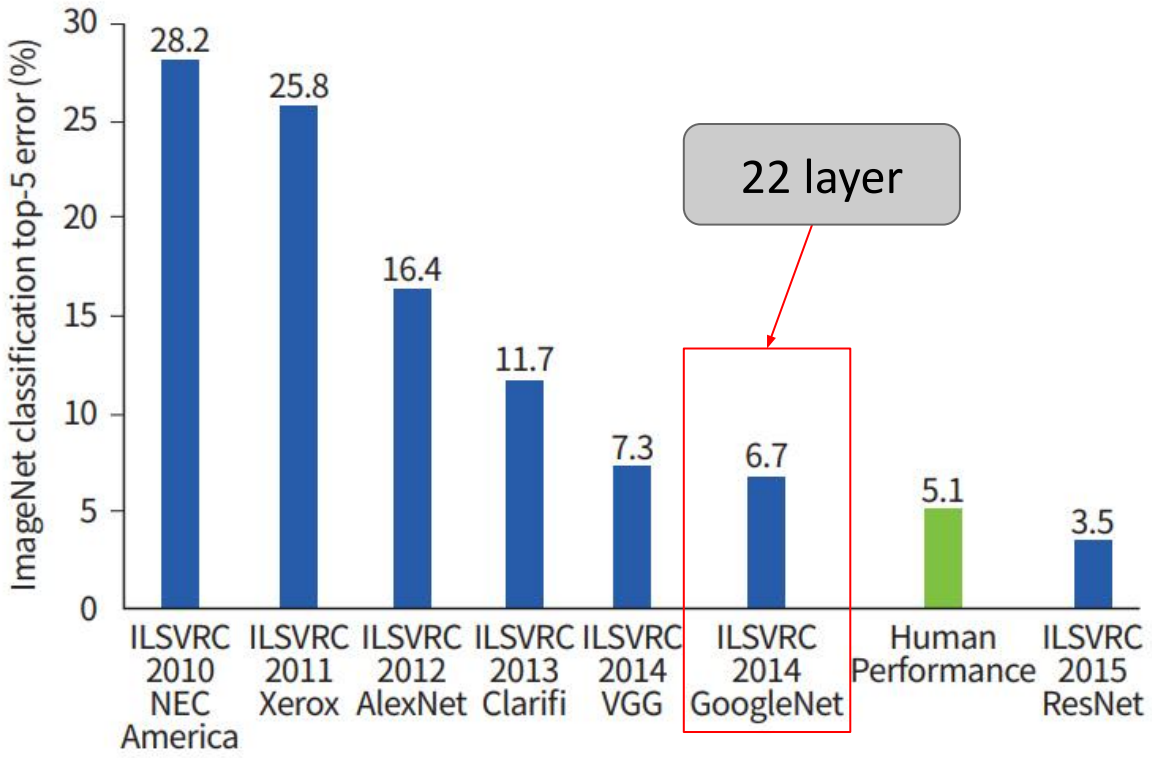


Convoluzione 5x5; campo recettivo eff 5x5



Convoluzione 3x3(x2); campo recettivo eff 5x5

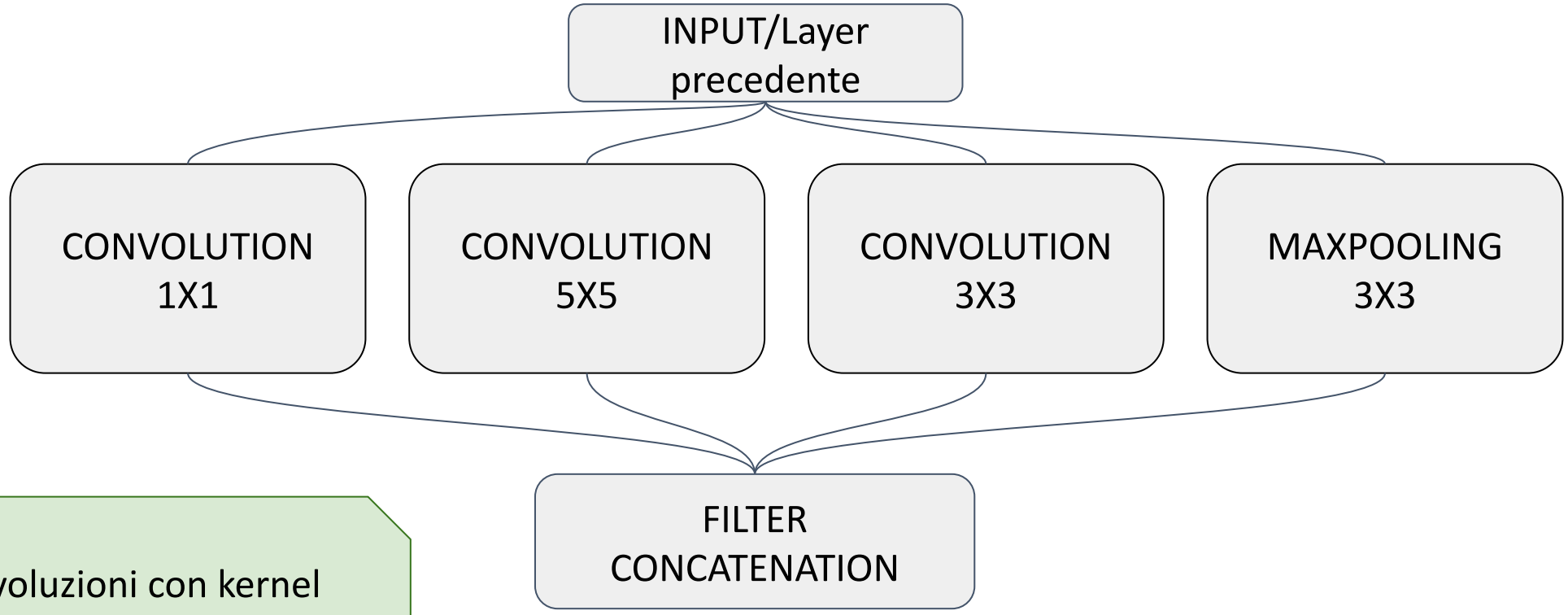
Da VGG19 a GoogLeNet



Sempre nel 2014 vince la competizione di ImageNet una rete chiamata GoogLeNet

InceptionNet (GoogLeNet)

Ricerca di rappresentazioni su più scale: i kernel di dimensione fissata cercano features alla stessa scala… come fare a metterne di diverse?

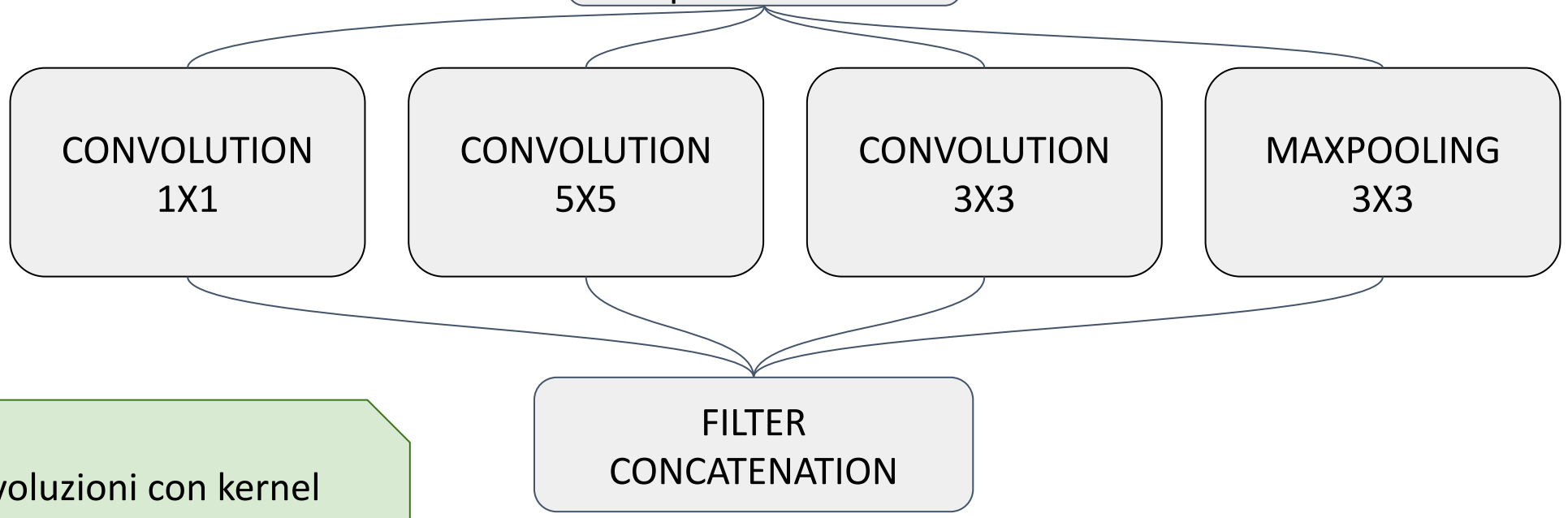


Convoluzioni con kernel diversi, guardano a scale diverse.

InceptionNet (GoogLeNet)

Ricerca di rapp[illegible] stessa scala... come f[illegible]

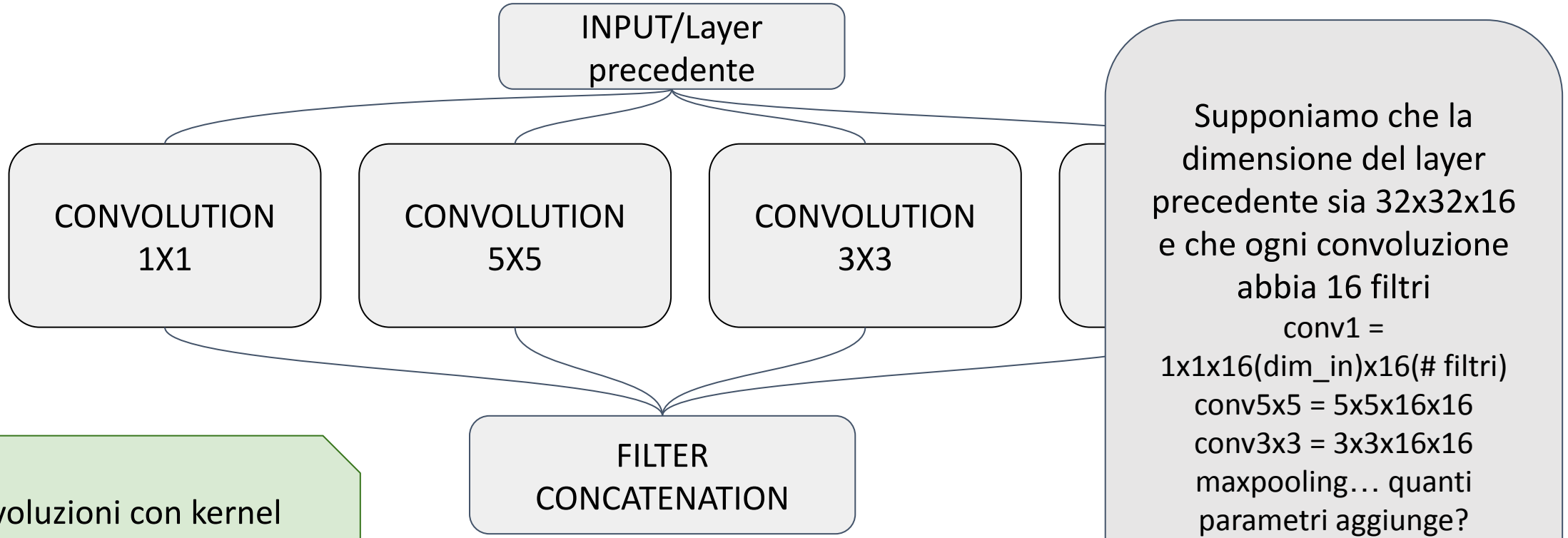
MA C'È UN PROBLEMA!! QUALCUNO/A RIESCE AD INTUIRLO?



Convoluzioni con kernel diversi, guardano a scale diverse.

InceptionNet (GoogLeNet)

Nella configurazione precedente il numero di parametri esplode



Supponiamo che la dimensione del layer precedente sia 32x32x16 e che ogni convoluzione abbia 16 filtri

conv1 = 1x1x16(dim_in)x16(# filtri)

conv5x5 = 5x5x16x16

conv3x3 = 3x3x16x16

maxpooling… quanti parametri aggiunge?

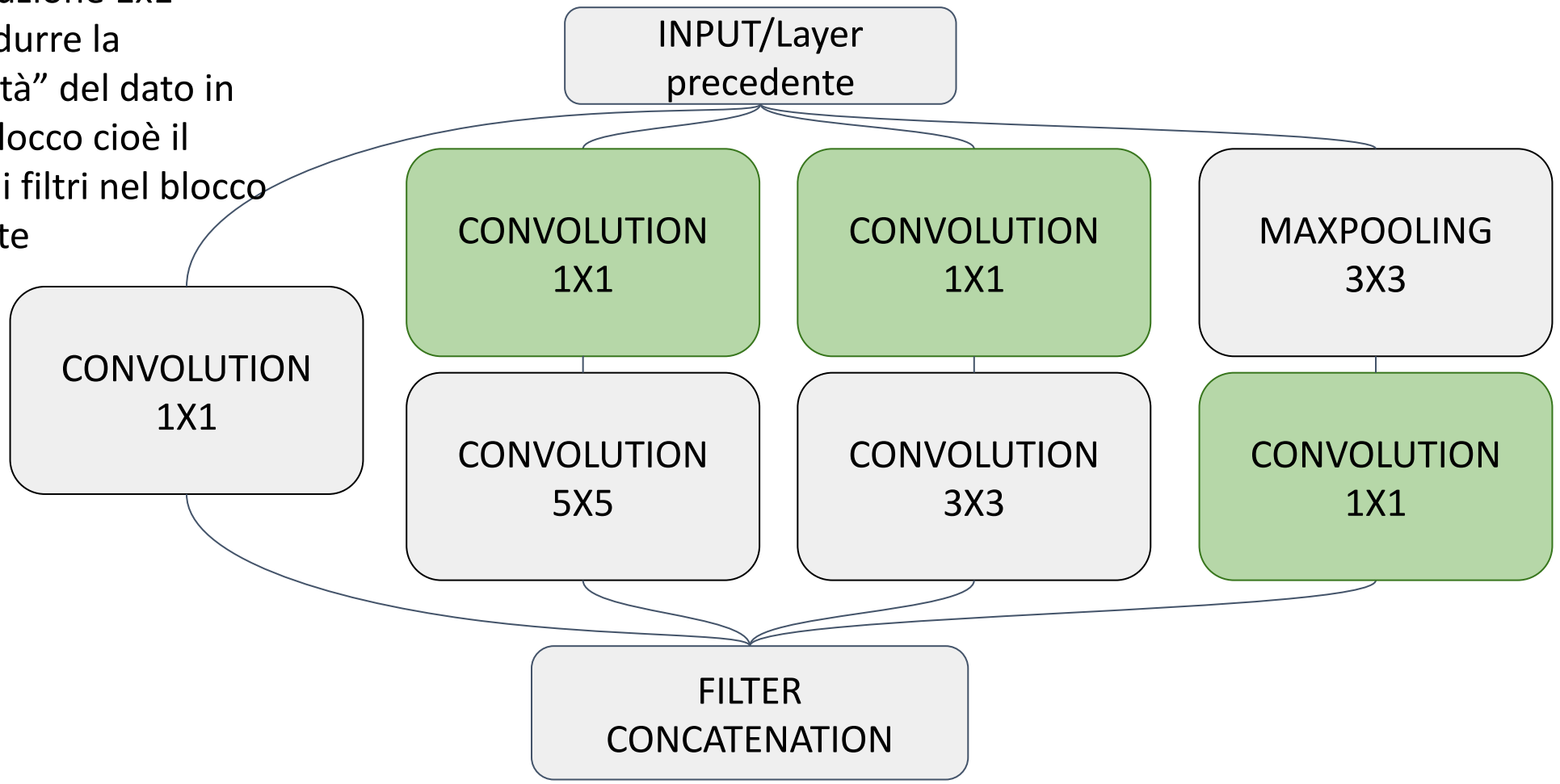
= 8960!

Convoluzioni con kernel diversi, guardano a scale diverse.

InceptionNet (GoogLeNet)

Nella configurazione precedente il numero di parametri esplode

La convoluzione 1x1 serve a ridurre la "profondità" del dato in input al blocco cioè il numero di filtri nel blocco precedente



InceptionNet (GoogLeNet)

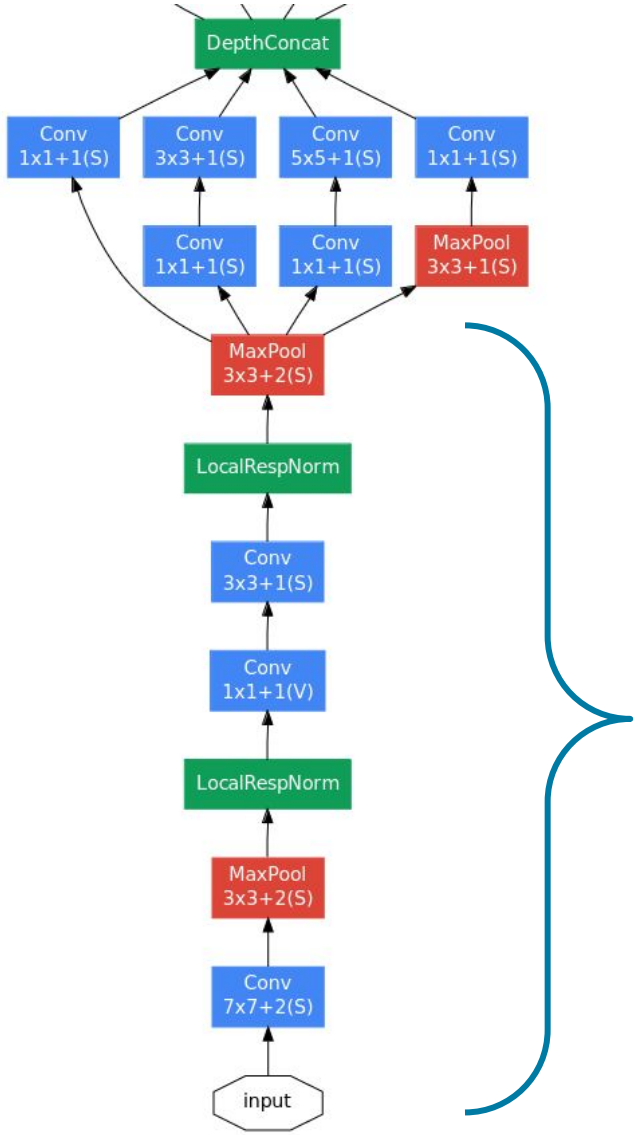


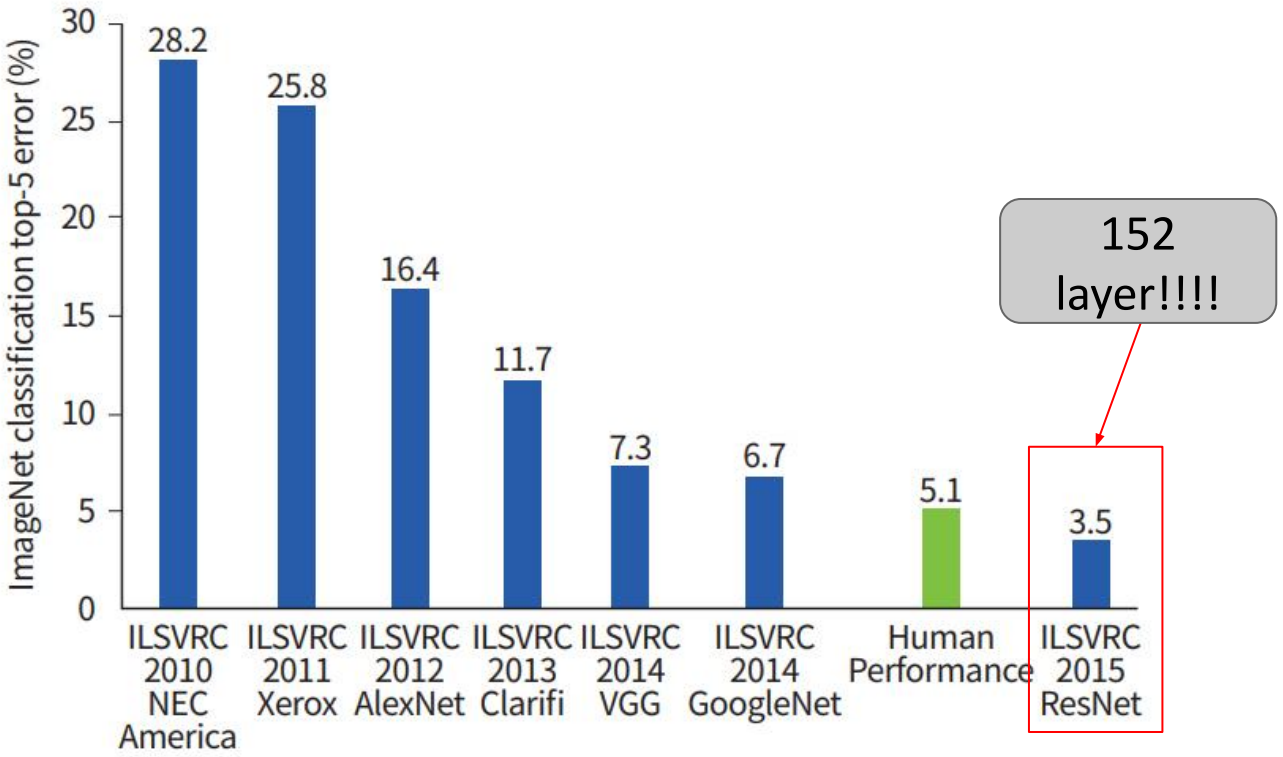
Immagine presa dal paper originale di presentazione di InceptionNet. (bibliografia in fondo)

Notare che prima di entrare nel blocco cosiddetto *inception* il dato di input viene processato tramite altri layer convoluzionali.

Questo fa sì che la dimensione del dato all'ingresso del blocco inception sia MxNx#di filtri del layer precedente

ResNet

Da GoogleNet (InceptionNet) a ResNet c'è un salto **significativo del numero di layer!!**



Le reti convoluzionali diventano lo stato dell'arte in moltissimi (quasi tutti) campi della *Computer Vision.*

Dal 2012 in poi ImageNet viene vinta **SEMPRE** da reti convoluzionali profonde!

Perchè la profondità è un problema?

All’aumentare del **numero di layer** -> Aumenta il **numero di neuroni** -> aumenta il numero di **parametri trainabili**

Questo significa che:

- aumenta il **costo computazionale** -> necessità di GPU più performanti (in particolare VRAM)
- aumenta il **tempo** di allenamento -> tempo macchina necessario maggiore

Quando l’algoritmo che voglio allenare impiega 2 settimane, è più complicato sviluppare l’algoritmo!

Ma c’è un altro problema specifico per le CNN:
l’incubo del **GRADIENTE EVANESCENTE**!

Molto in breve, allenare una CNN - che significa usare la BACKPROPAGATION per aggiornare i pesi una volta calcolata la loss function - richiede il calcolo di tante derivate.

Vanishing Gradient

Allenare una CNN - che significa usare la BACKPROPAGATION per aggiornare i pesi una volta calcolata la loss function - richiede **il calcolo di tante derivate**.

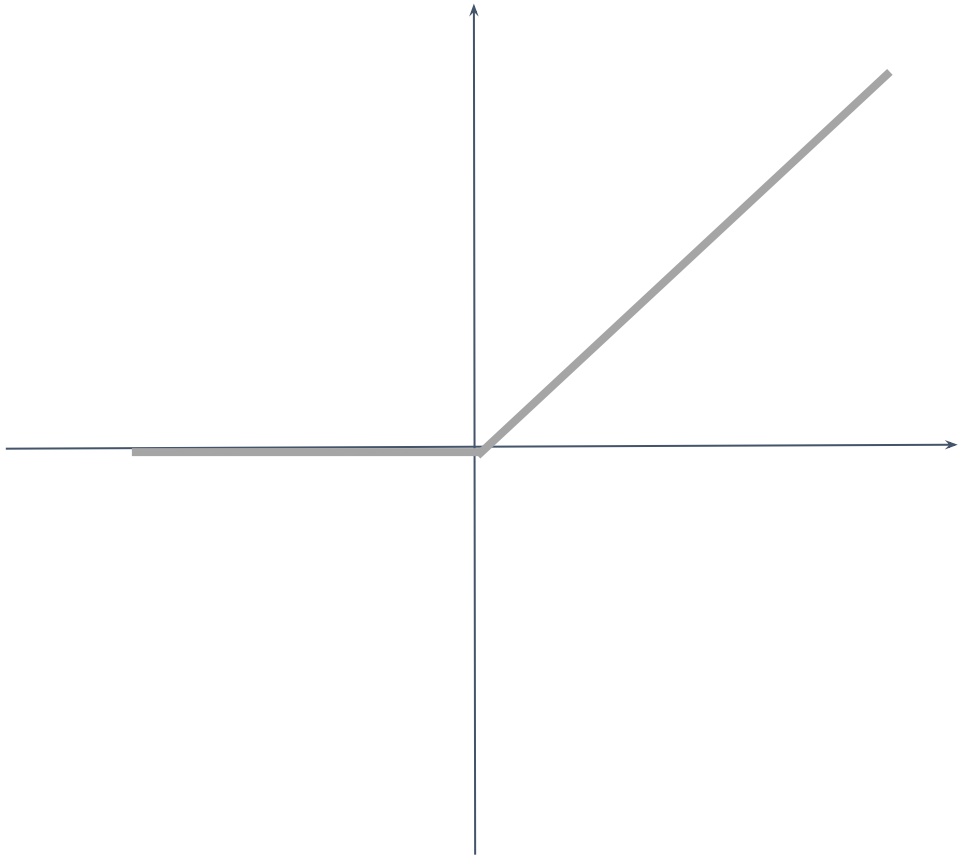
Vicino alla convergenza, quando quindi la rete neurale sarà vicina ad un minimo locale della funzione di costo, **le variazioni dei pesi diventeranno molto piccole** -> **le loro derivate** saranno molto piccole, **vicine allo 0.**

Applicare la **chain rule** per il calcolo del gradiente significa semplicemente che, per aggiornare i pesi della rete neurale, dobbiamo **moltiplicare molte di queste derivate** ->

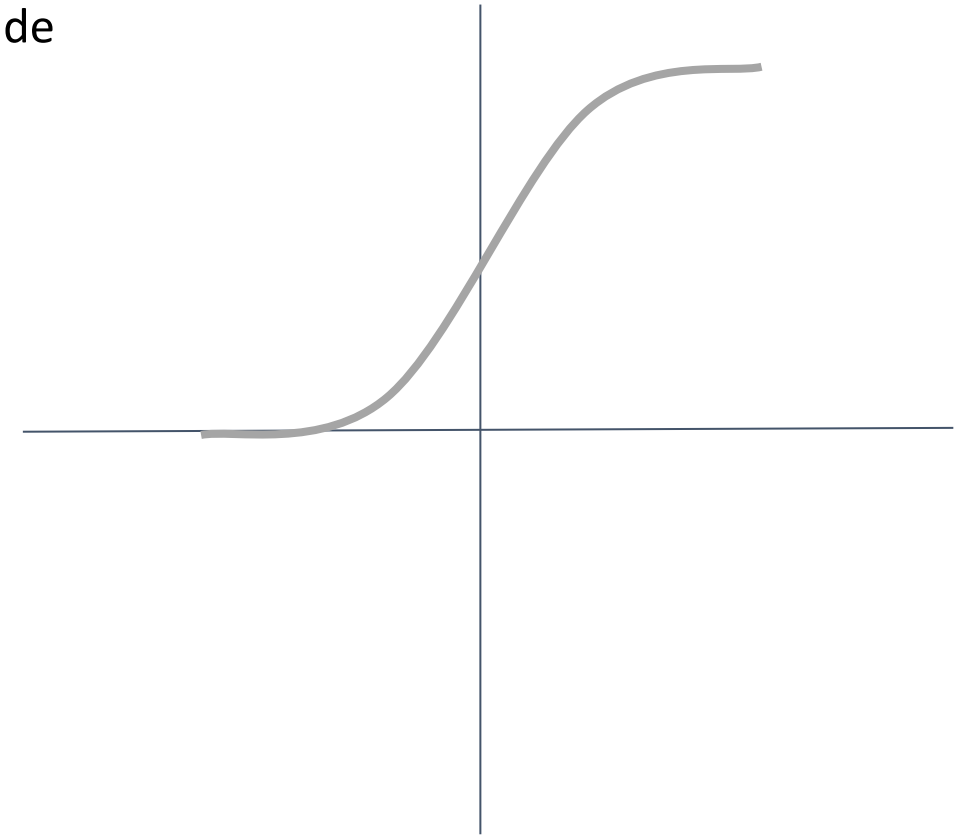
Moltiplicare tanti fattori molto vicini allo zero può portare ad avere 0 ovunque!

Vanishing Gradient

ReLU



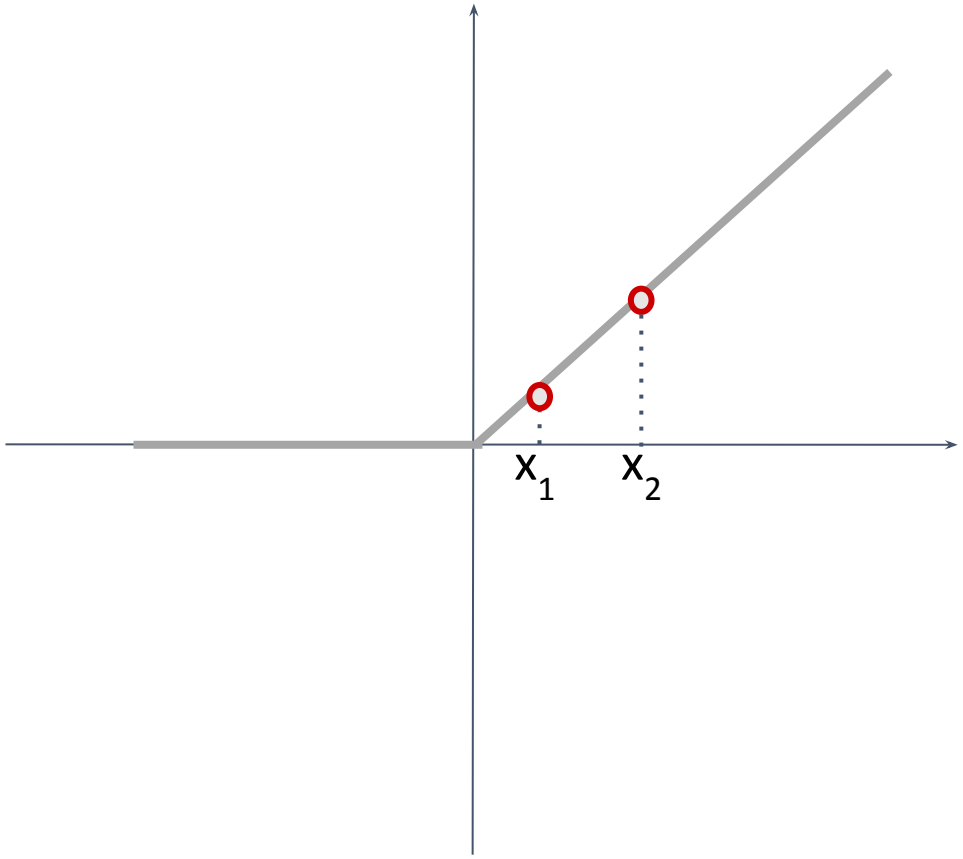
Sigmoide



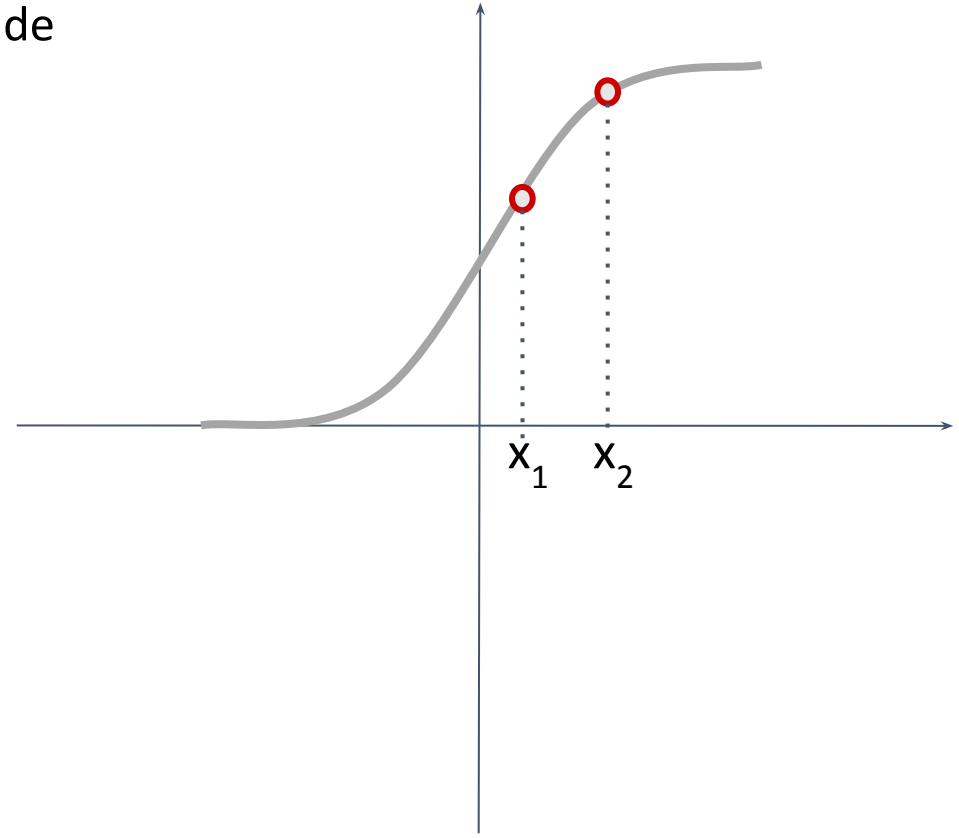
Cosa succede alle derivate di ReLU e della sigmoide all’aumentare di x?

Vanishing Gradient

ReLU



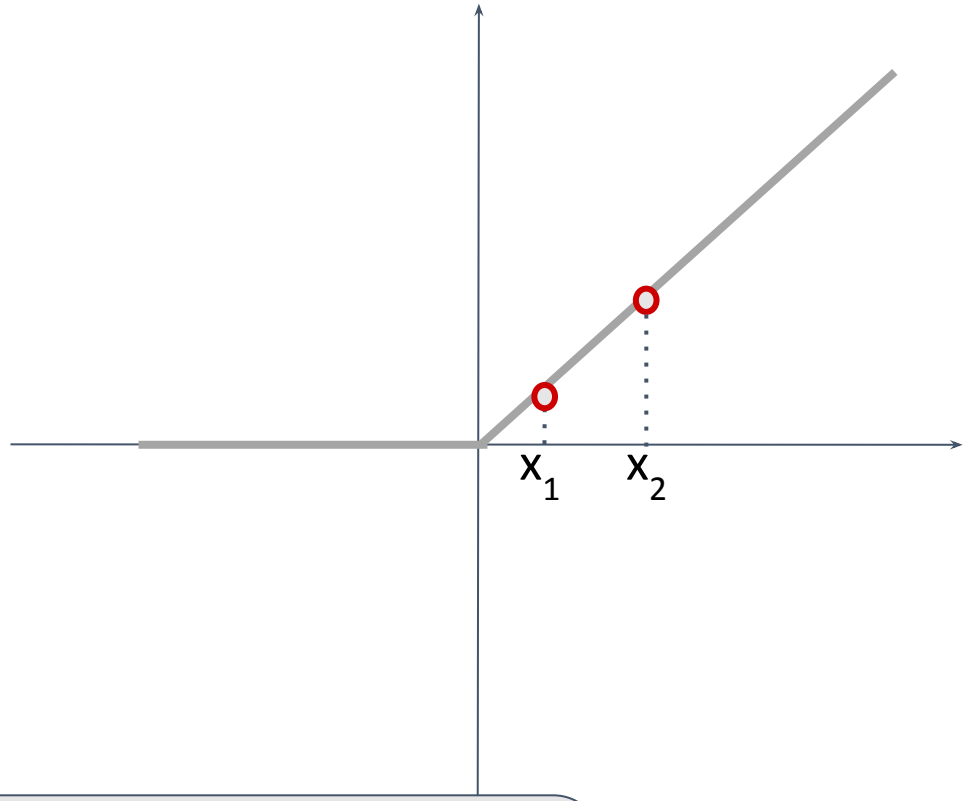
Sigmoide



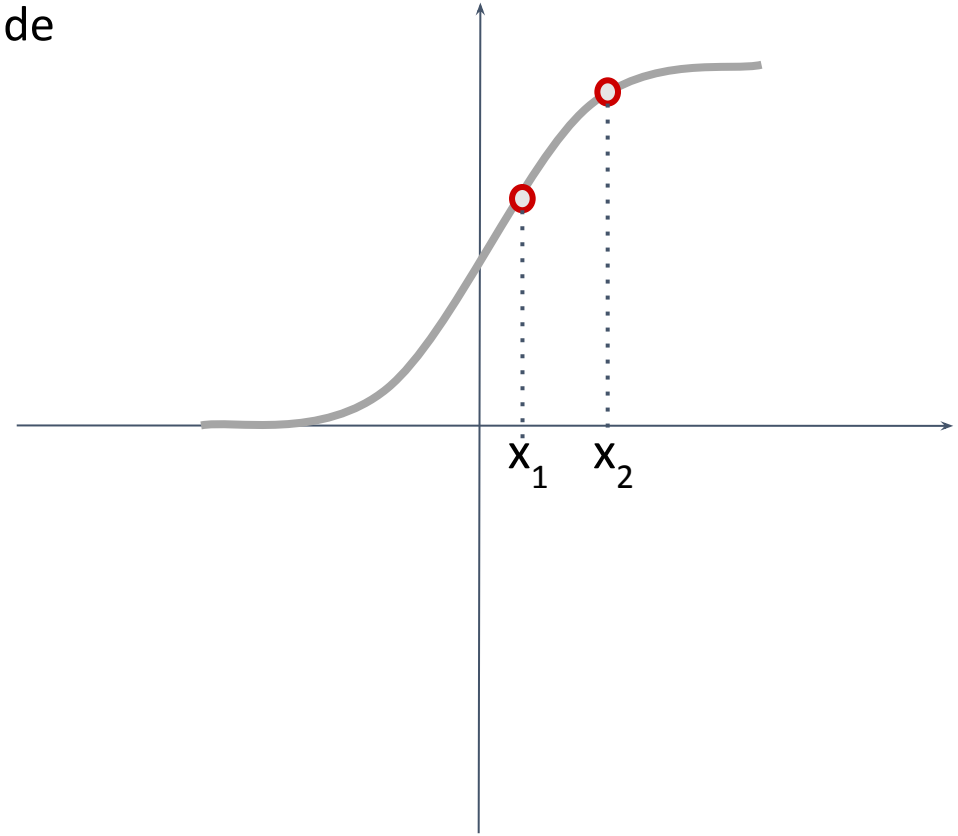
Cosa succede alle derivate di ReLU e della sigmoide all'aumentare di x?

Vanishing Gradient

ReLU



Sigmoide

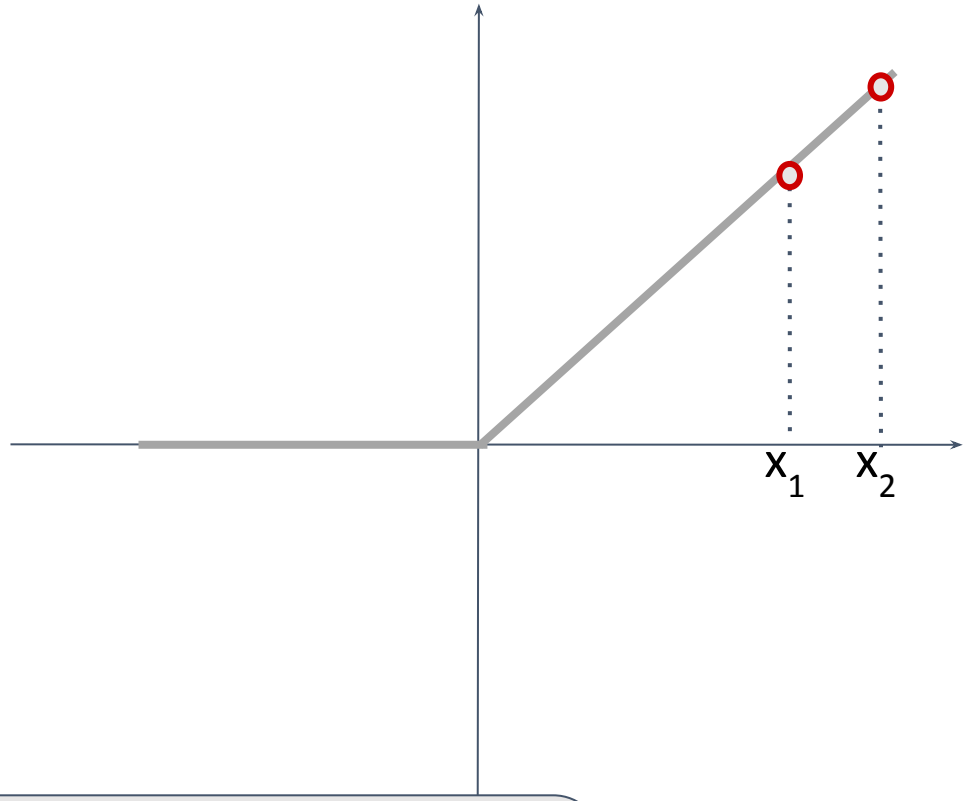


Cosa succede alle derivate di ReLU e della sigmoide all'aumentare di x?

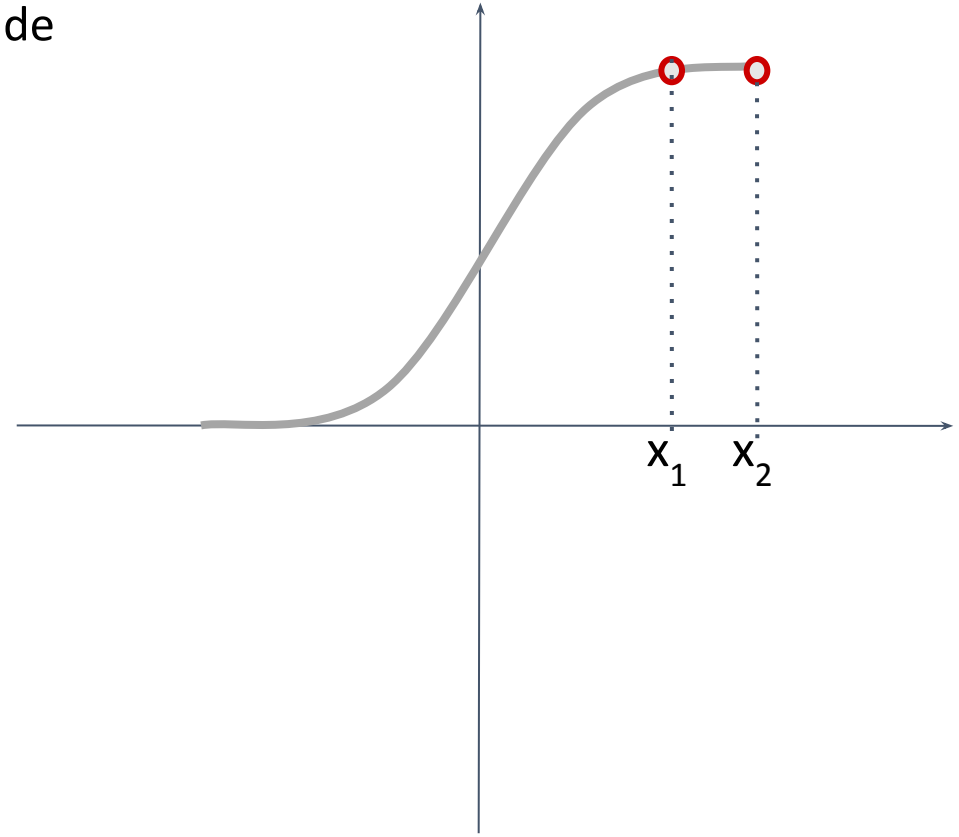
In questo caso, non molto. La variazione sembra simile tra ReLU e Sigmoide

Vanishing Gradient

ReLU



Sigmoide

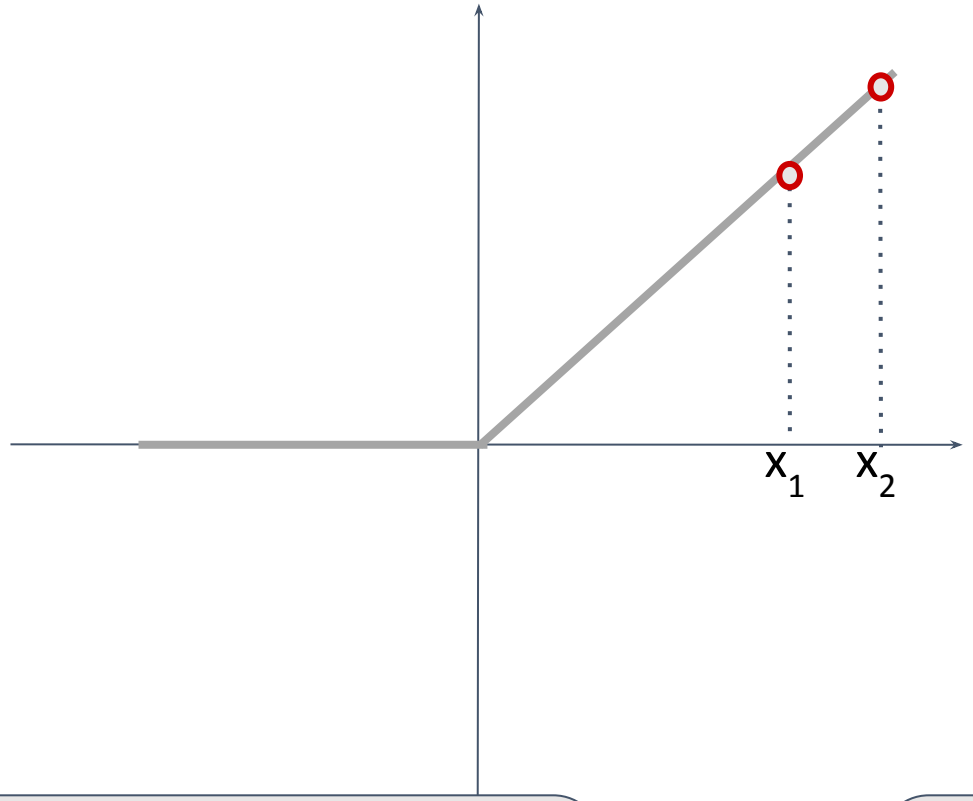


Cosa succede alle derivate di ReLU e della sigmoide all'aumentare di x?

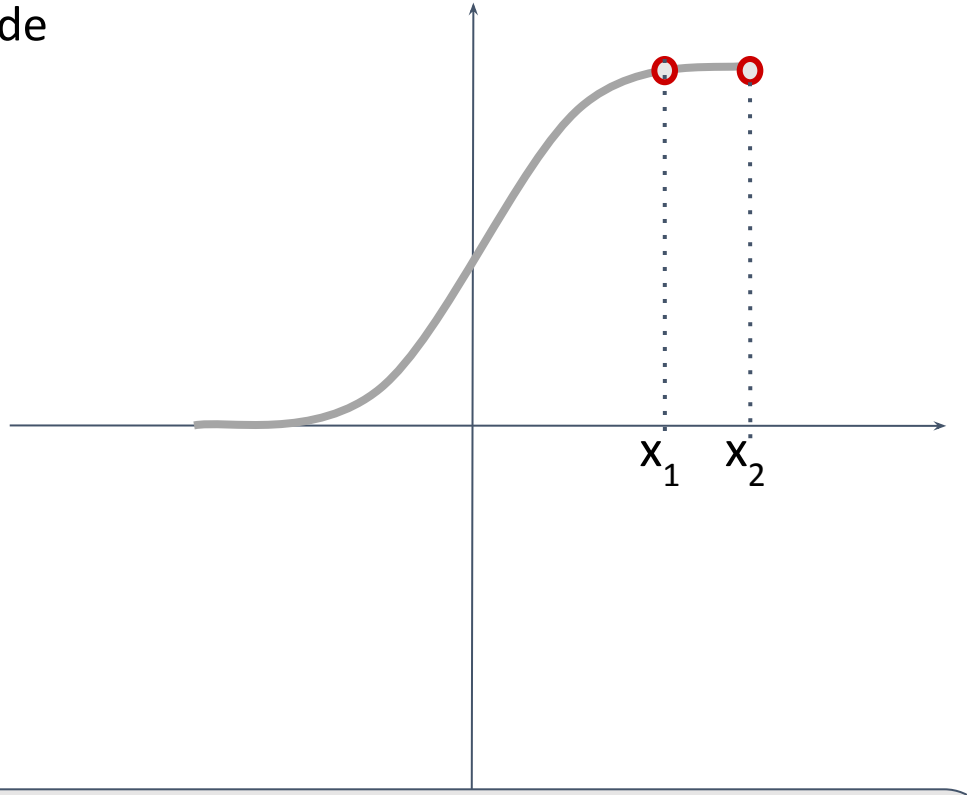
?

Vanishing Gradient

ReLU



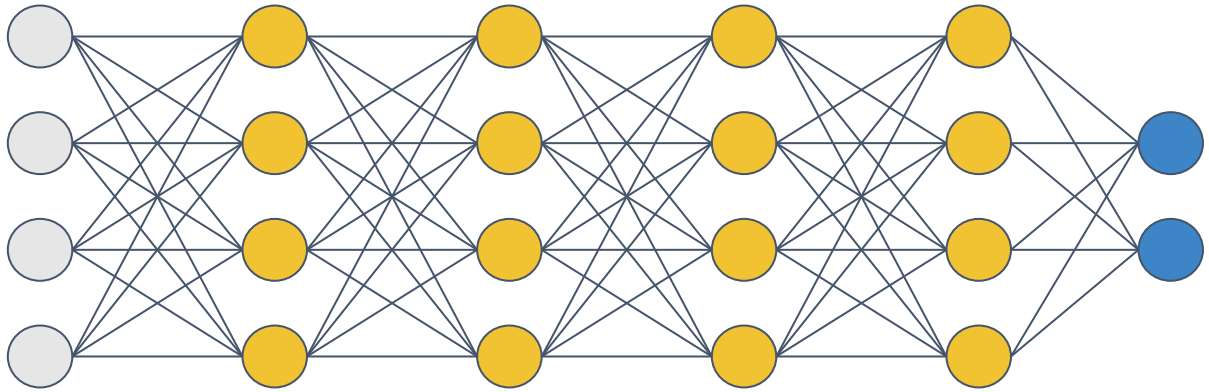
Sigmoide



Cosa succede alle derivate di ReLU e della sigmoide all'aumentare di x?

Questo è il motivo per cui usare la ReLU previene il gradiente evanescente! Tuttavia…

Going deeper

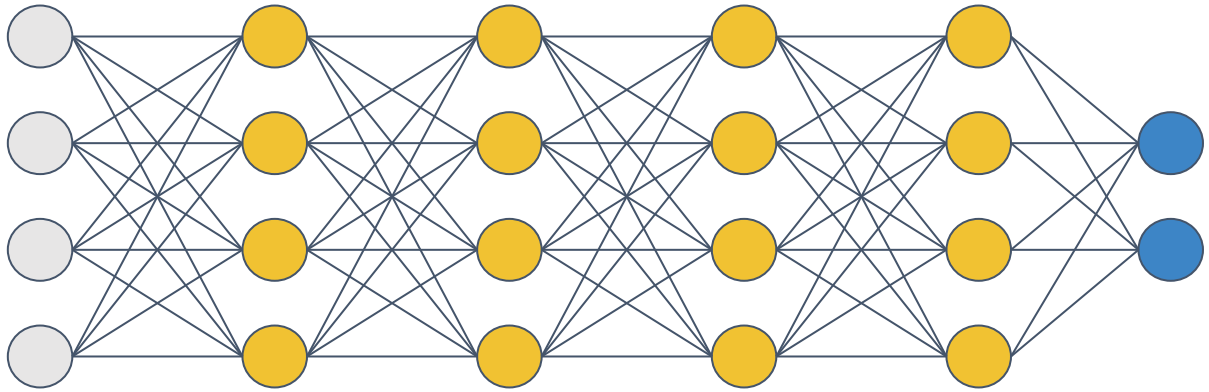


Il numero di moltiplicazioni che devono essere fatte, dipende ovviamente dal numero di layer… Abbiamo detto che moltiplicare tante derivate vicine allo zero può portare al gradiente evanescente.

Quale potrebbe essere una soluzione?

IDEE?

Going deeper



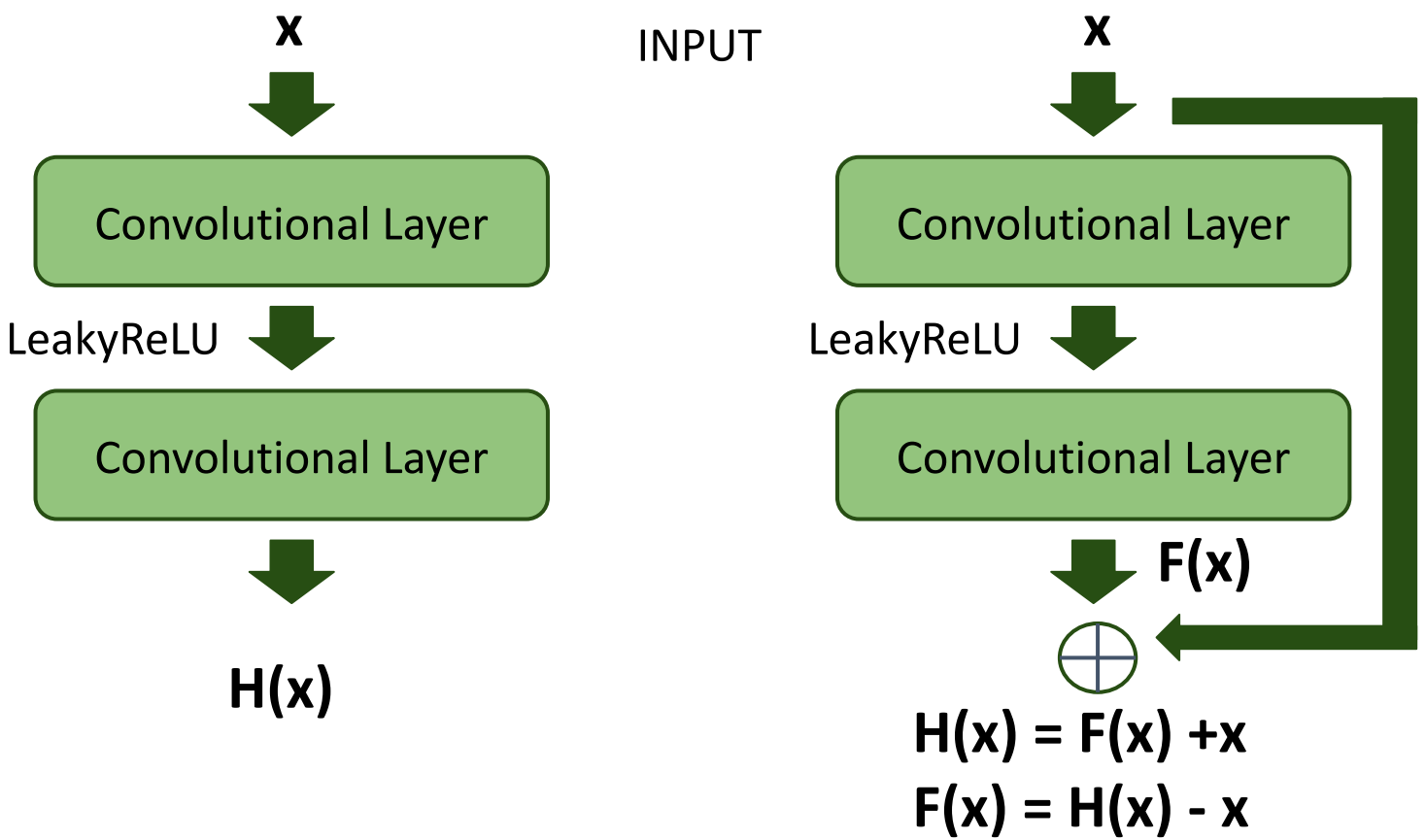
Il numero di moltiplicazioni che devono essere fatte, dipende ovviamente dal numero di layer… Abbiamo detto che moltiplicare tante derivate vicine allo zero può portare al gradiente evanescente.

Quale potrebbe essere una soluzione?

Far diminuire il numero di moltiplicazioni da fare?
COME?

Skip connection

Le skip connection sono comuni nelle Convolutional Neural Network.
Sono connessioni tra layer NON adiacenti.
ResNet si basa proprio su questo meccanismo!

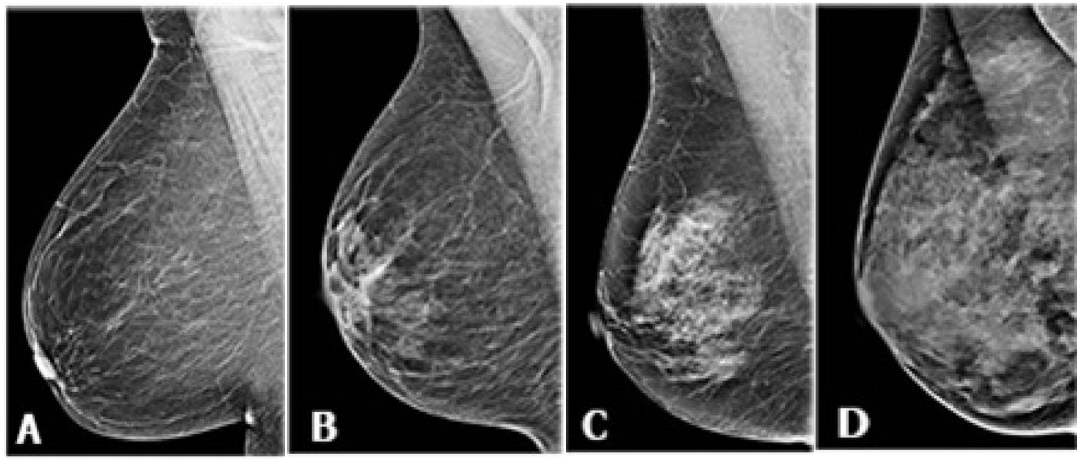


Short skip connections:

- In ResNet viene usata l'addizione
- Riducono il problema del gradiente evanescente;
- Preservano l'informazione attraverso i tanti layer

Applicazione ResNet: densità mammografica

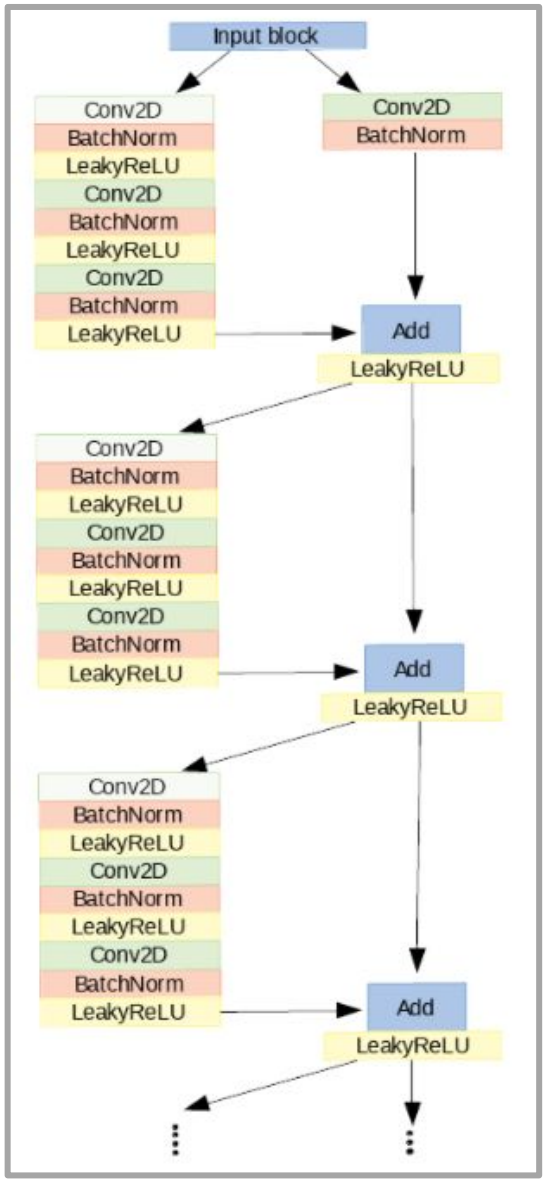
Obiettivo: identificazione della classe di densità BI-RADS (classi A,B,C,D) con Residual CNN.
Perchè: Contribuire allo sviluppo di un indice di dose personalizzato (che dipende dalla densità).



Dataset: about **2000 digital mammographic exams** collected by Azienda Ospedaliero-Universitaria Pisana (AOUP).

Dense/Non-dense	Left (%)	Right (%)	All (%)	BI-RADS	Left (%)	Right (%)	All (%)
Accuracy	84.4	88.8	89.4	Accuracy	73.3	76.7	77.3
Recall	82.3	89.9	90.0	Recall	72.1	79.2	77.1
Precision	85.5	87.7	88.9	Precision	76.6	75.2	78.6

[Lizzi F. et al., Residual convolutional neural networks to automatically extract significant breast density features. vol. 1089. Springer International Publishing; 2019]



RADIOMX

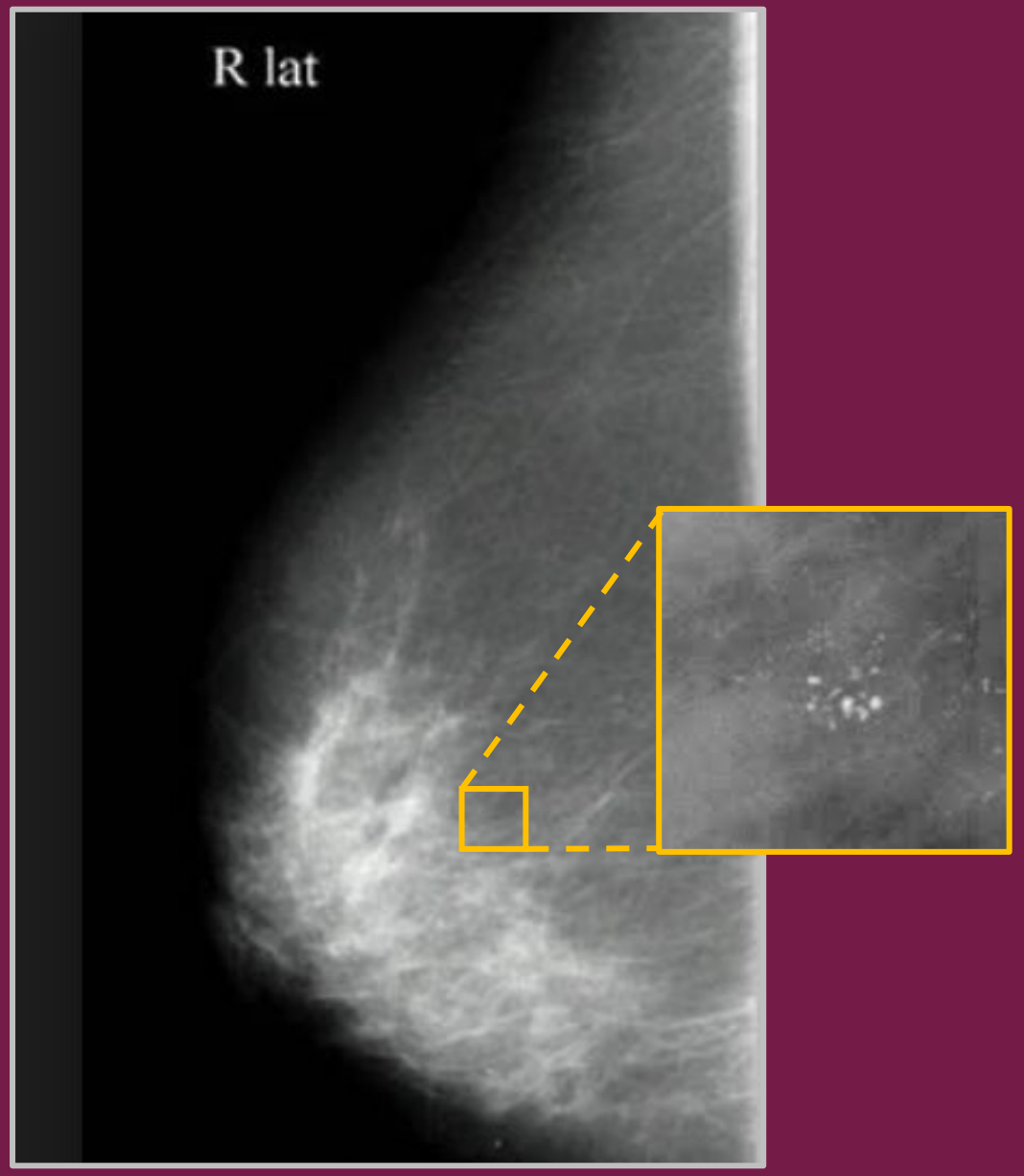
AZIENDA OSPEDALIERO UNIVERSITARIA PISANA

INFN Istituto Nazionale di Fisica Nucleare

FONDAZIONE PISA

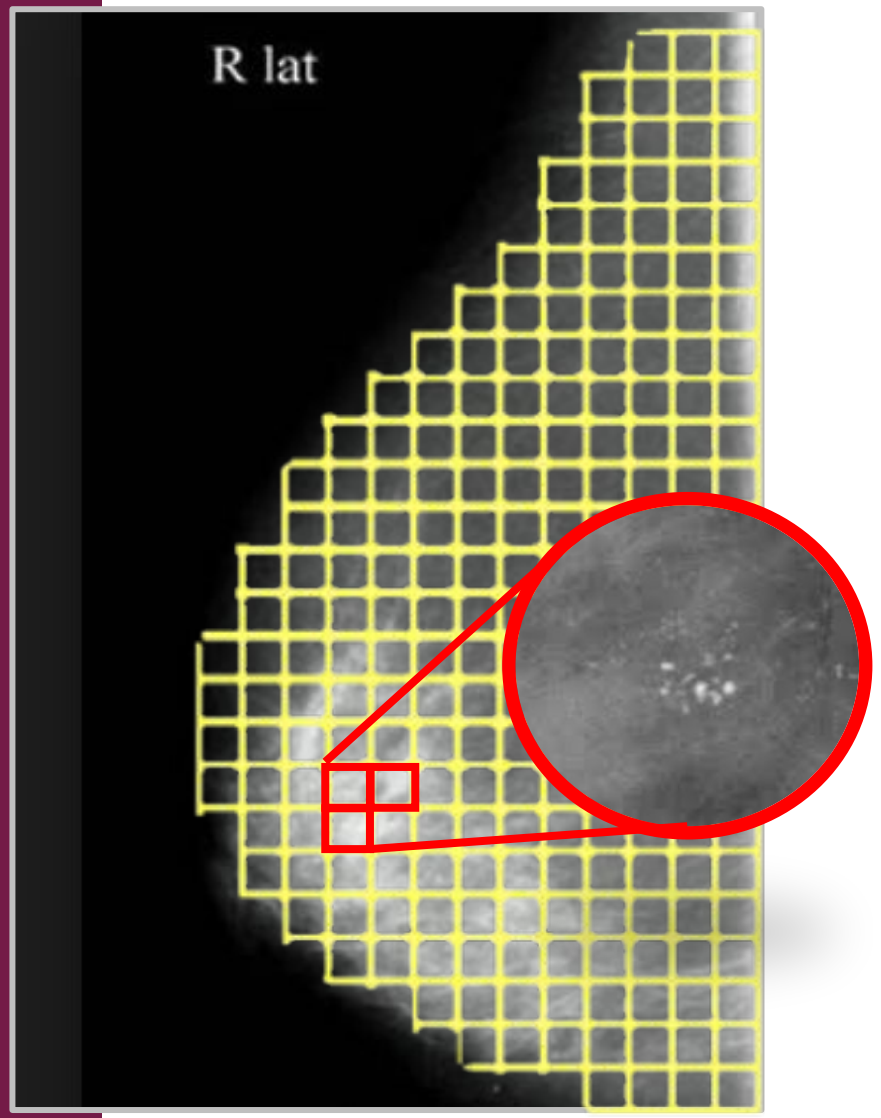
Objective

The aim of this exercise is to **create and train** a simple Convolutional Neural Network (**CNN**) for deep-learning based classification of **normal tissue vs.** tissue containing **microcalcification** clusters in mammograms.



The CNN you will develop could help to...

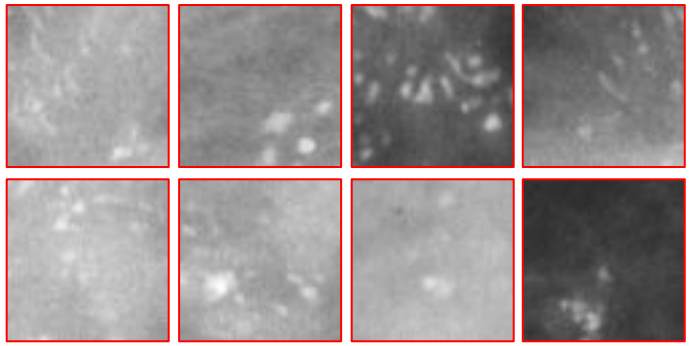
- Identify and locate regions suspected of containing a microcalcification cluster
- ... and finally build an AI-based system capable of supporting radiologists in reading screening mammograms



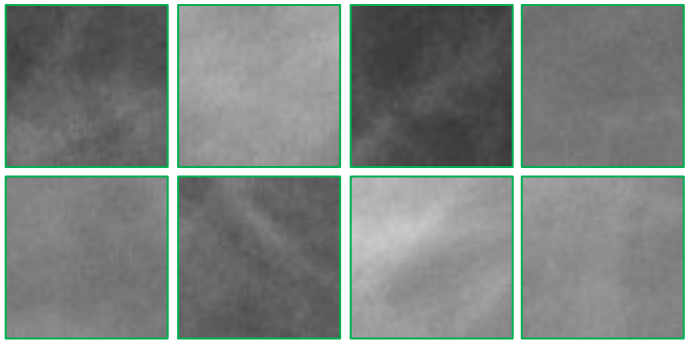
The analysis pipeline

Input data

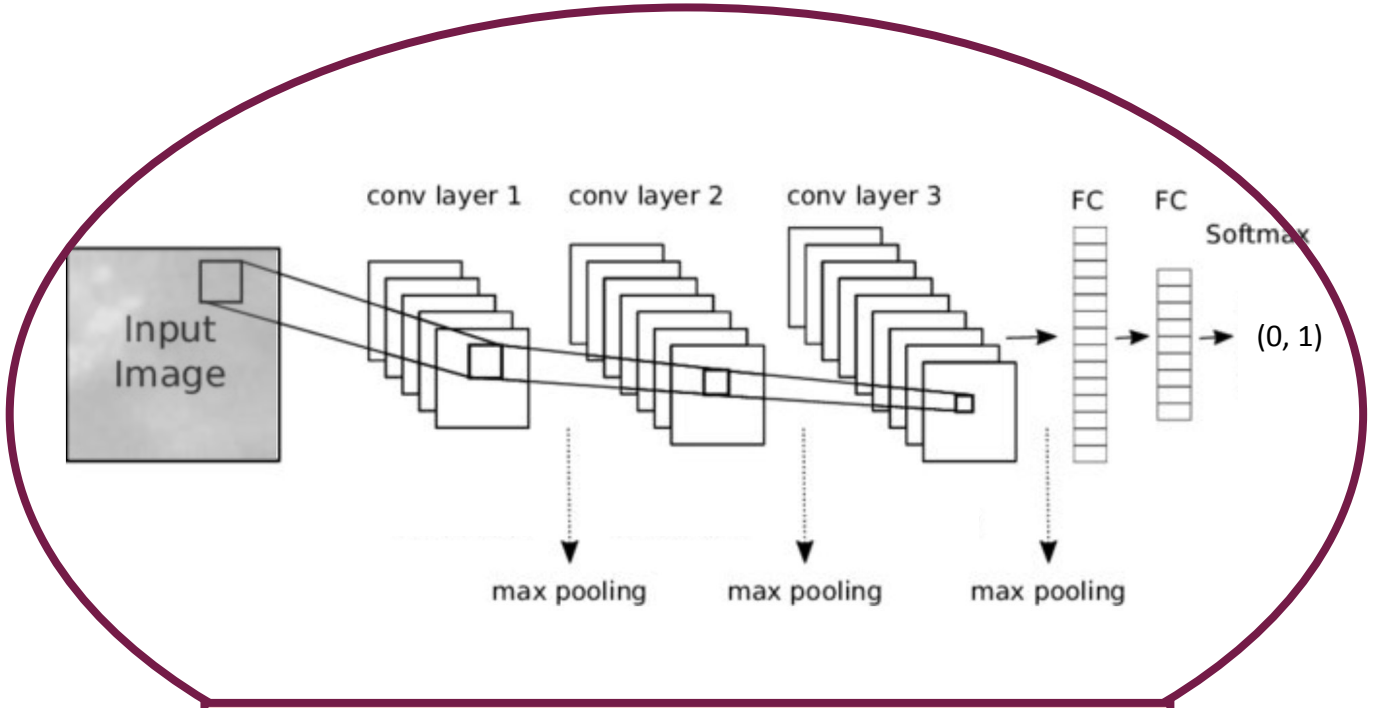
Label 1: images with microcalcifications



Label 0: images of normal tissue



CNN classifier



You have to design, train and test the CNN

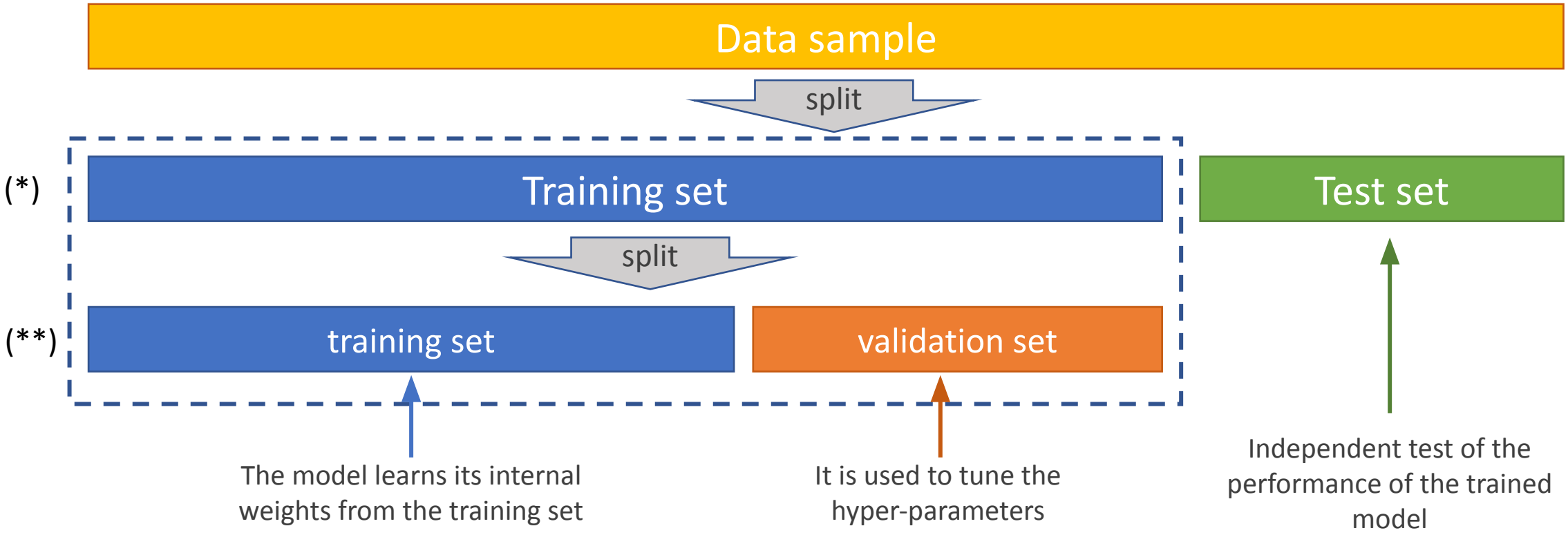
Output

Output = 1
This image contains microcalcifications

Implementation steps

- Load and visualize image data
- Define the CNN model
- Compile and train the CNN model on the training data
- Predict the labels of the validation/test data samples
- Calculate the classification accuracy
- Create the ROC curve and calculate the AUC
- Compare the performance of different models

Training, Validation and Test sets



(*) The **Training** and **Test** sets will be identified since the beginning of the exercise. This partition is kept fixed.

(**) The **Training** split into the **training and validation** sets is made with a dedicated function (*train_test_splitting*):

- This partitioning can be kept fixed to make direct comparisons among different models/training options
- Additionally, the impact of the training data variability on the classifier performance can be evaluated by training and evaluating the CNN performance several times and providing the average and standard deviation of the performances

Metrics to assess the performance

Confusion matrix

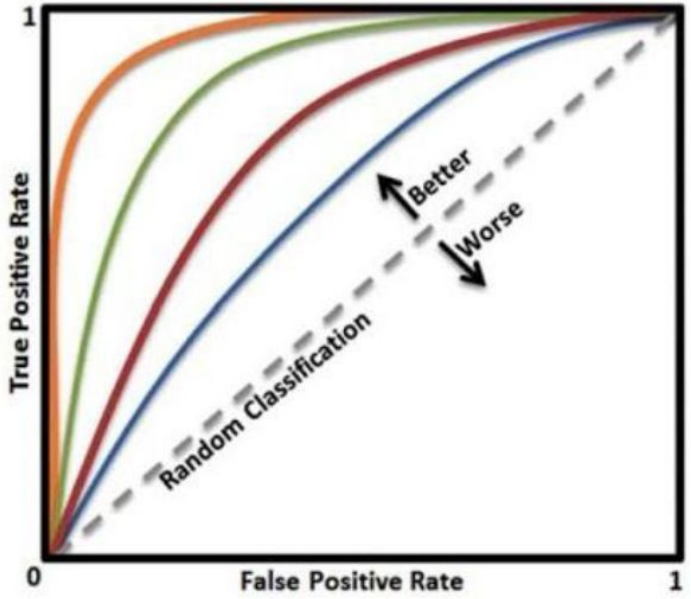
	Predicted 0	Predicted 1
Actual 0	TN	FP
Actual 1	FN	TP

True Positive Rate $TPR = \frac{TP}{TP + FN}$

False Positive Rate $FPR = \frac{FP}{FP + TN}$

Accuracy $ACC = \frac{TP + TN}{TP + TN + FP + FN}$

Receiver Operating Characteristic (ROC) curve



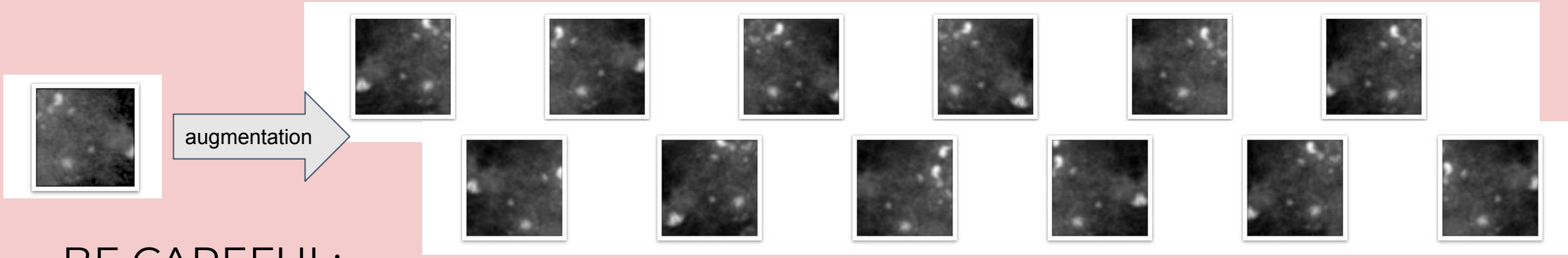
The Area Under the ROC Curve (AUC) is a global metric to evaluate the performance of a classifier

AUC = 1 for a perfect classifier

AUC = 0.5 for a random classifier

If you have extra time ...

- You can also try to implement a ***data augmentation strategy*** to improve the classification performance



- BE CAREFUL:
  - The augmented training set should retain **the same "properties"** of the original medical images.
  - Augmented images should be **realistic and consistent** with their labels.
  - Physical and quantitative information contained in the images should not be lost or ruined during data augmentation, otherwise you risk losing the relationship between the image content and its clinical relevance.