Analisi di Immagini e Video (Computer Vision)

Giuseppe Manco

Outline

- Reti Neurali
- CNN
- Architetture di rete

Crediti

- Slides adattate da vari corsi e libri
  - Deep Learning (Ettore Ritacco)
  - Deep Learning (Bengio, Courville, Goodfellow, 2017)
  - Andrey Karpathy
  - Computer Vision (I. Gkioulekas) - CS CMU Edu
  - Cmputational Visual Recognition (V. Ordonez), CS Virgina Edu

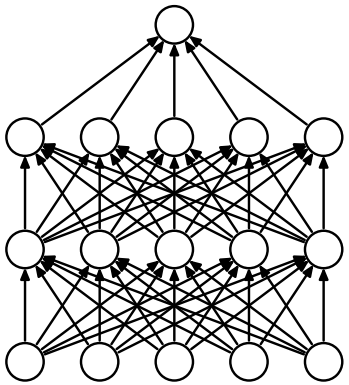
Concetti avanzati

Deep learning effettivo

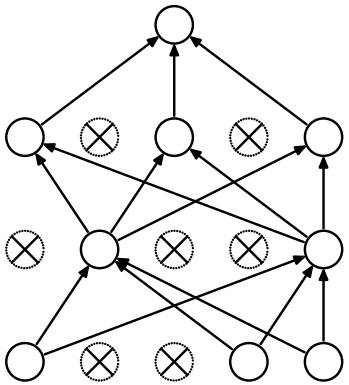
- Regolarizzazione
  - Aggiunge una penalizzazione sui pesi nella funzione di loss
  - Criteri: sparsità, norma, …
- Dropout
  - Reset di un numero random di pesi
  - Decorrela i nodi nella rete
- Gradient clipping
  - Gradient exploding
- Smart initialization
  - Better random initialization methods (Glorot and Bengio, 2010)
- Data augmentation
  - More to come later…

Dropout

- Rimozione random di nodi durante il forward pass nel training



(a) Standard Neural Net



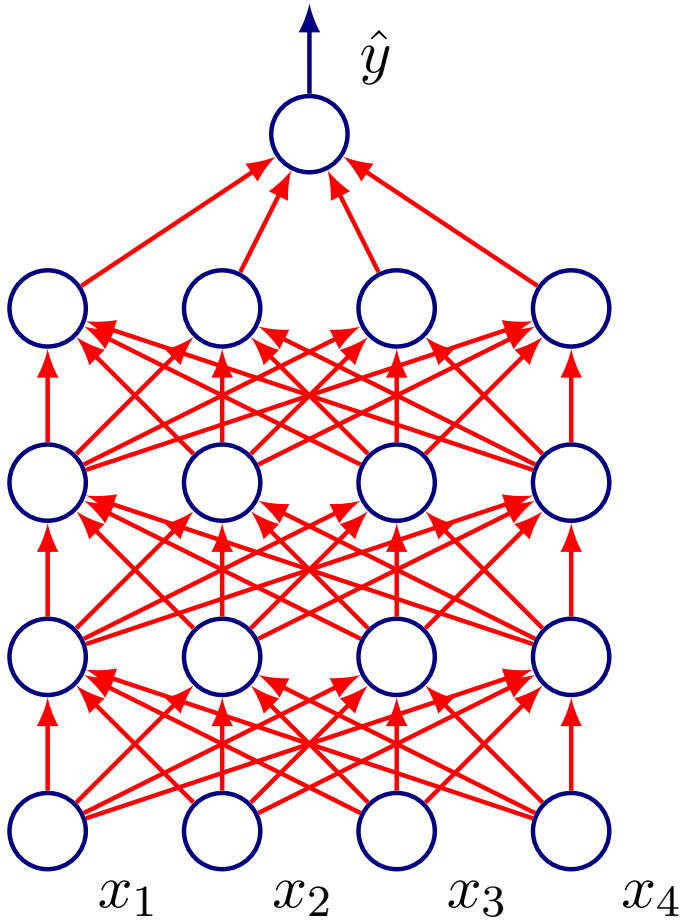
(b) After applying dropout.

Dropout

- Aumenta l’indipendenza delle unità
  - Co-adaptation
    - Una unità interna non può basarsi su altre unità
- Interpretazione in termini di ensembles

Convoluzione

Fully connected networks



$$a_i = \sum_{j \prec i} w_{i,j} z_j$$

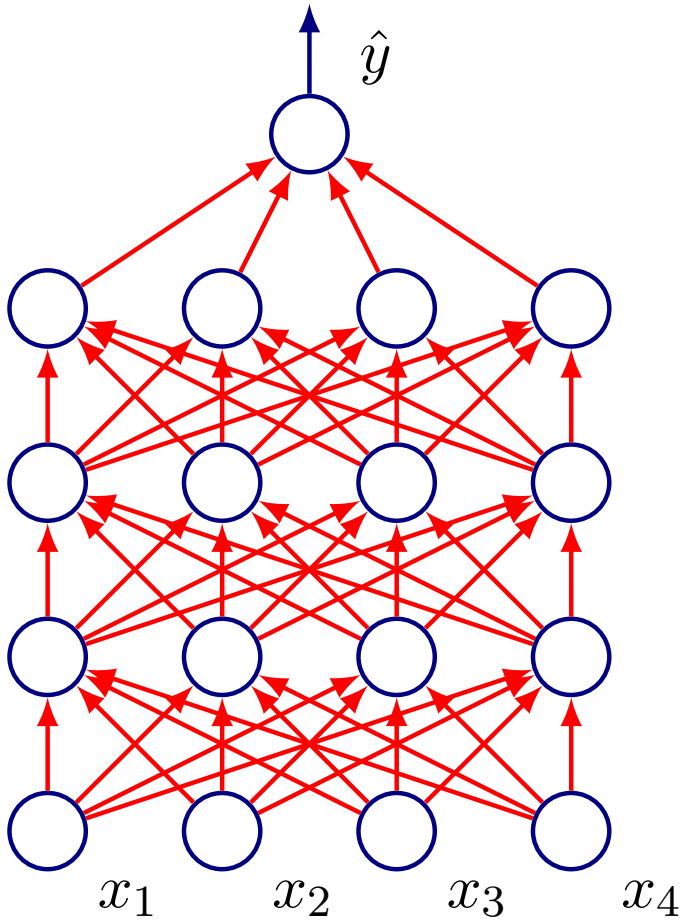
$$z_i = f(a_i)$$

$$\mathbf{a}^{(h+1)} = \mathbf{W}^{(h)} \mathbf{z}^{(h)}$$

$$\mathbf{z}^{(h+1)} = f\left(\mathbf{a}^{(h+1)}\right)$$

$$\mathbf{z}^{(0)} = \mathbf{x}$$

Fully connected networks



$$a_i = \sum_{j \prec i} w_{i,j} z_j$$

$$z_i = f(a_i)$$

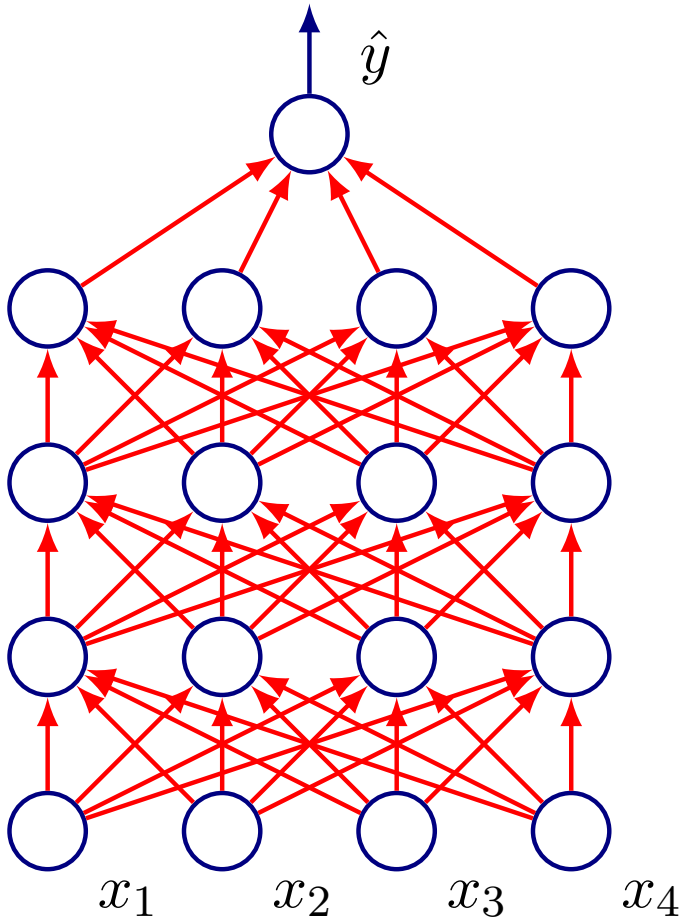
$$\mathbf{a}^{(h+1)} = \mathbf{W}^{(h)} \mathbf{z}^{(h)}$$

$$\mathbf{z}^{(h+1)} = f\left(\mathbf{a}^{(h+1)}\right)$$

$$\mathbf{z}^{(0)} = \mathbf{x}$$

- Ogni elemento connesso agli altri
  - (5*4) + (5*4) + (5*4) + 5 connections

Fully connected networks



$$a_i = \sum_{j \prec i} w_{i,j} z_j$$

$$z_i = f(a_i)$$

$$\mathbf{a}^{(h+1)} = \mathbf{W}^{(h)} \mathbf{z}^{(h)}$$

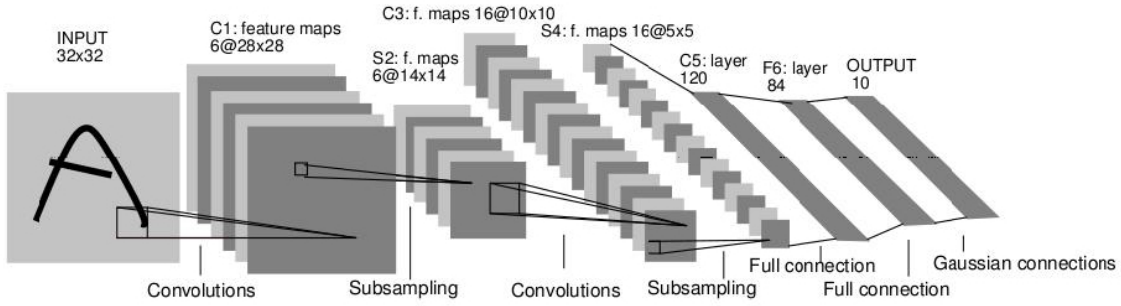
$$\mathbf{z}^{(h+1)} = f\left(\mathbf{a}^{(h+1)}\right)$$

$$\mathbf{z}^{(0)} = \mathbf{x}$$

Quanti sono i parametri di una generica rete?

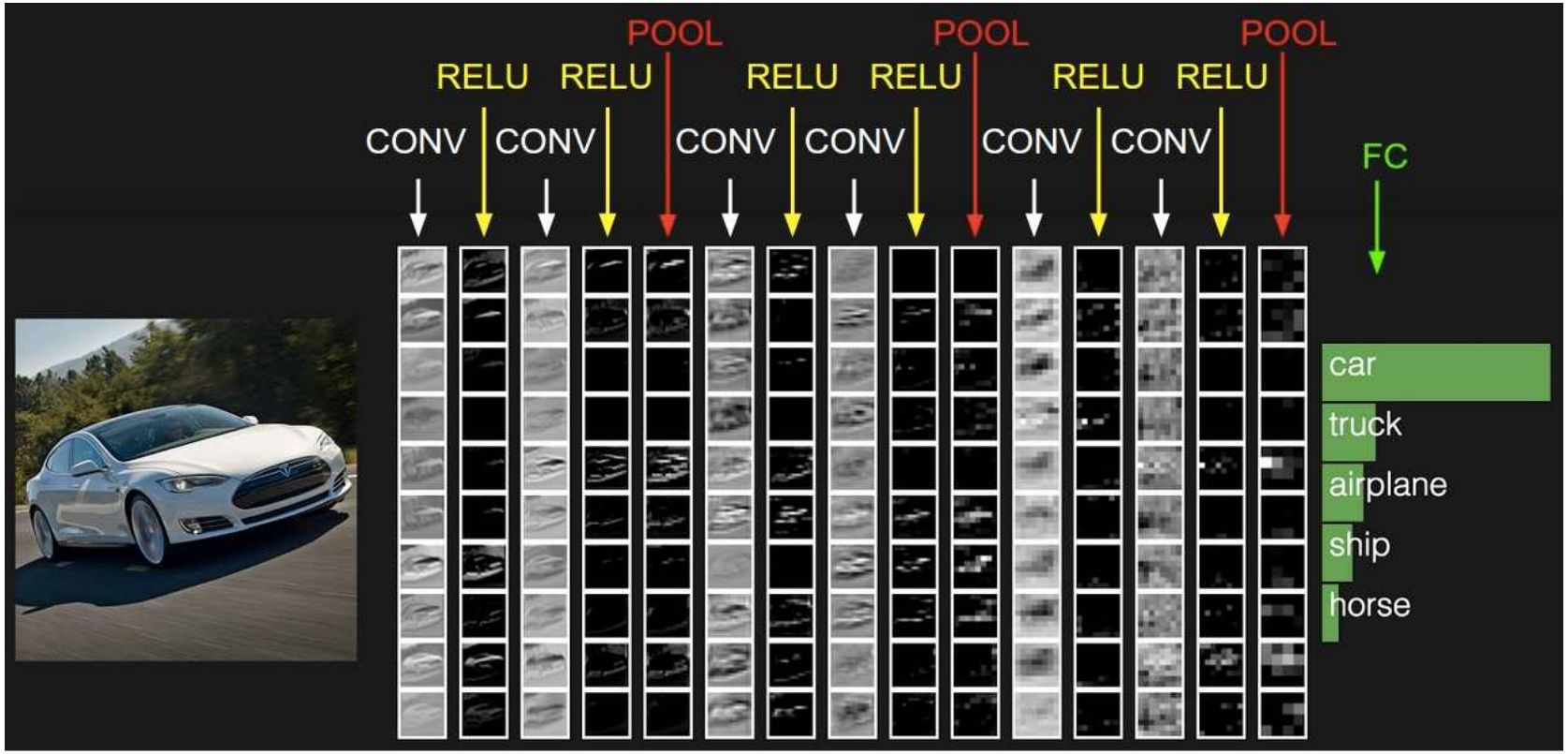
Convolutional networks

- Reti neurali che usano la convoluzione

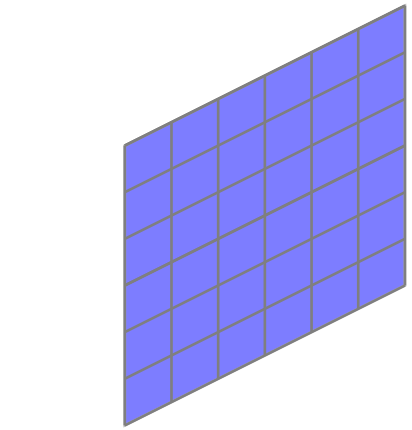


  - Convolution
  - pooling

Convolutional neural networks

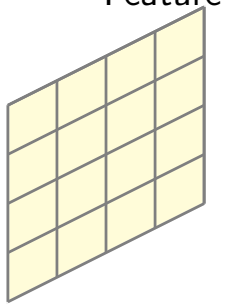


Convolution



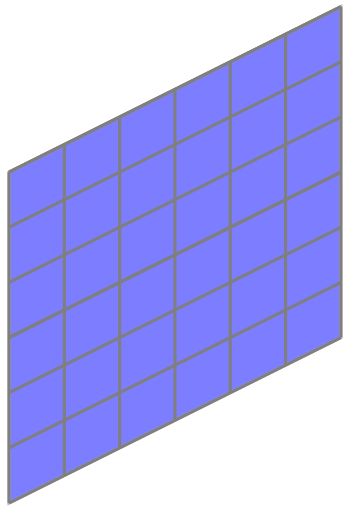
Grayscale Image

Feature Map



- Qual è il numero di parametri?

Convolution

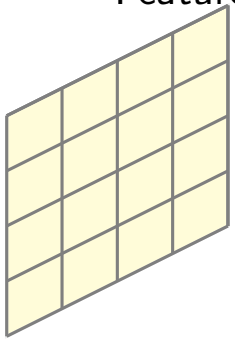


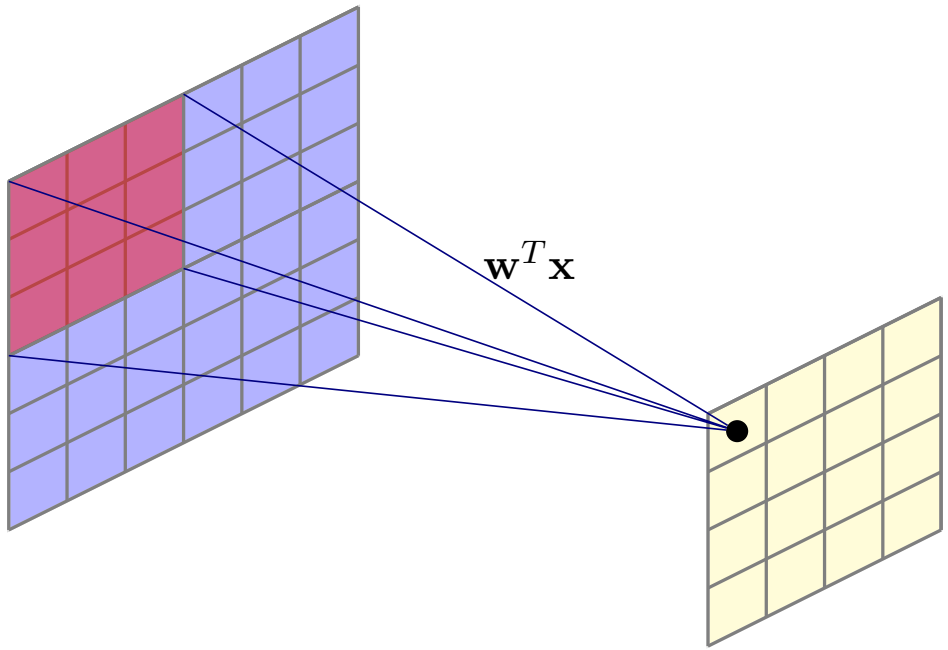
Grayscale Image

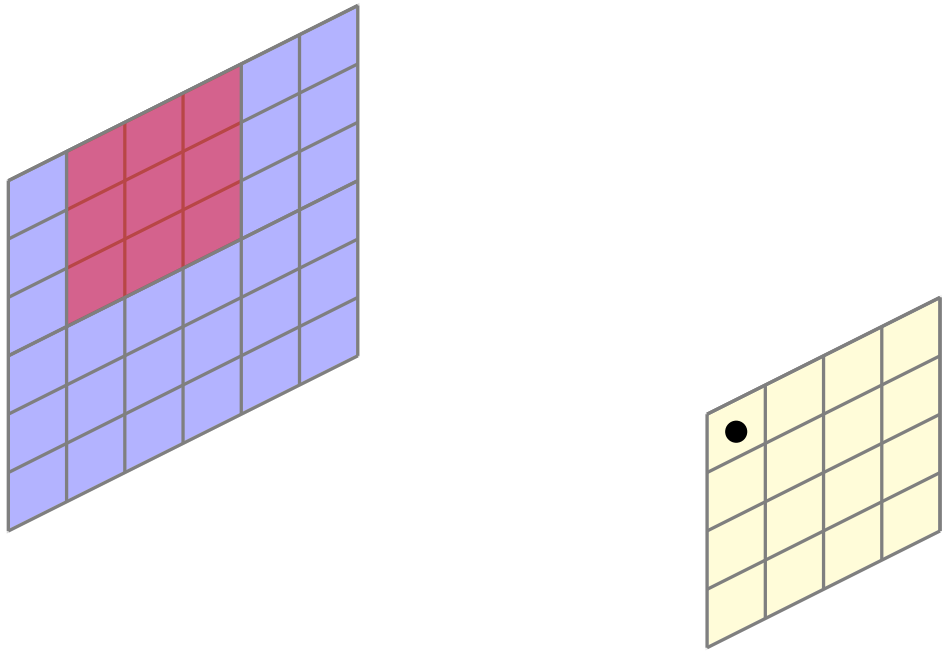
Kernel

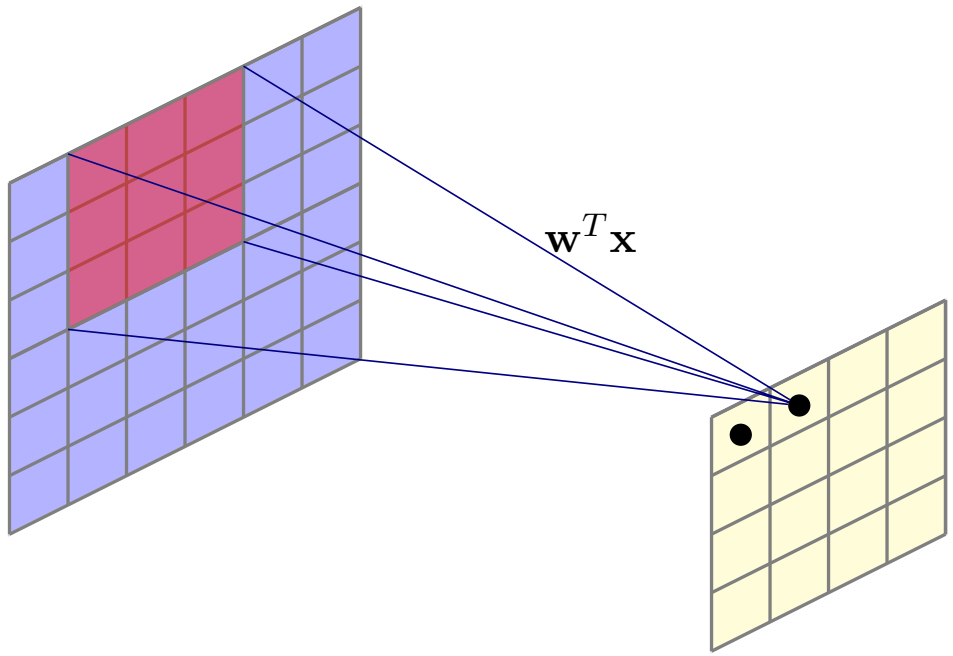
w_7	w_8	w_9
w_4	w_5	w_6
w_1	w_2	w_3

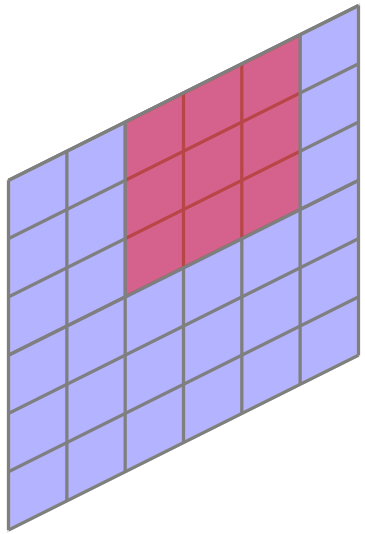
Feature Map

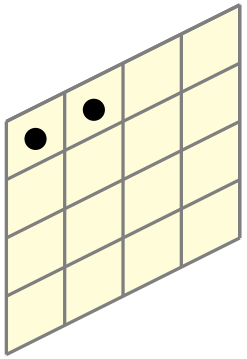


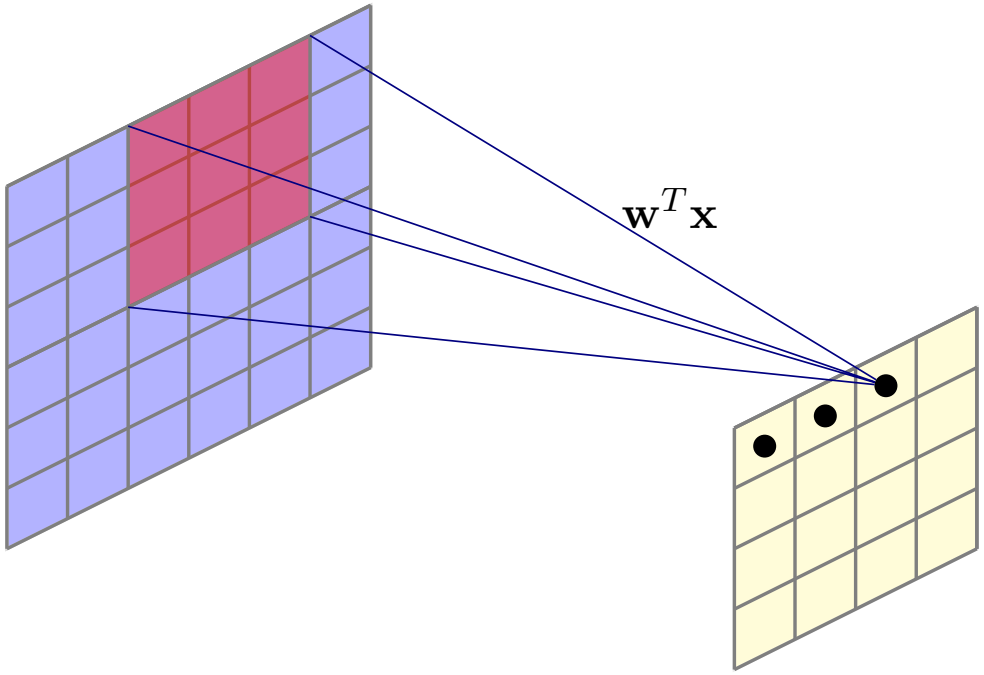

$\mathbf{w}^T\mathbf{x}$

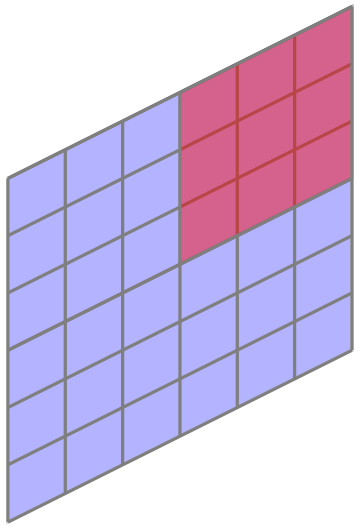


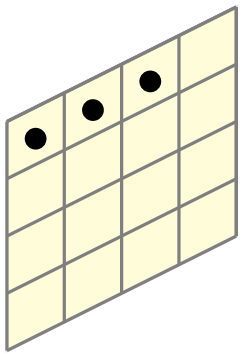

$\mathbf{w}^T\mathbf{x}$

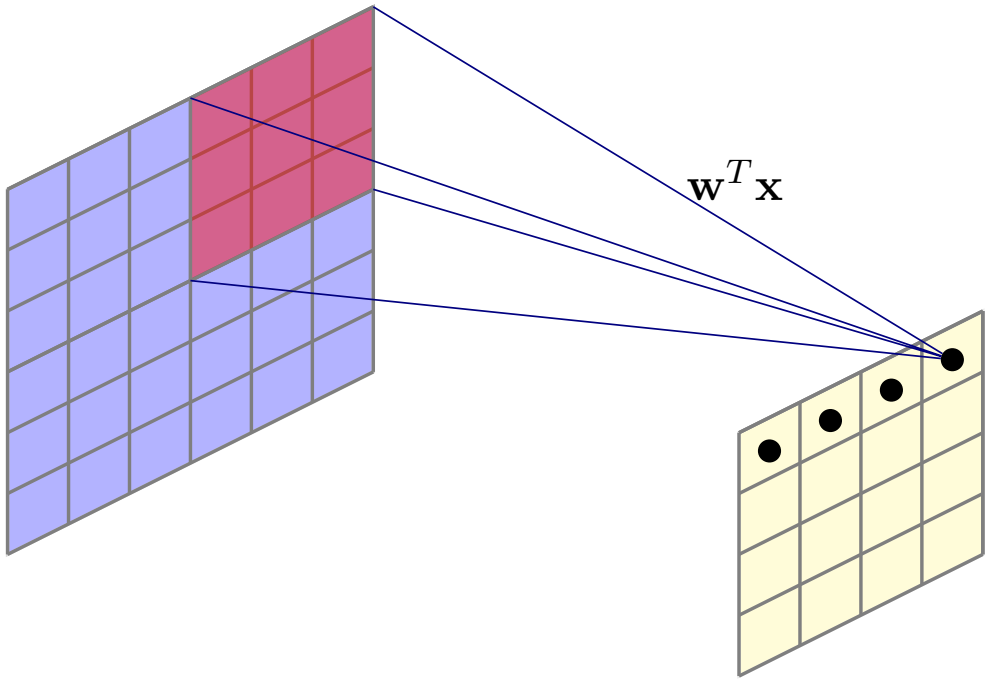


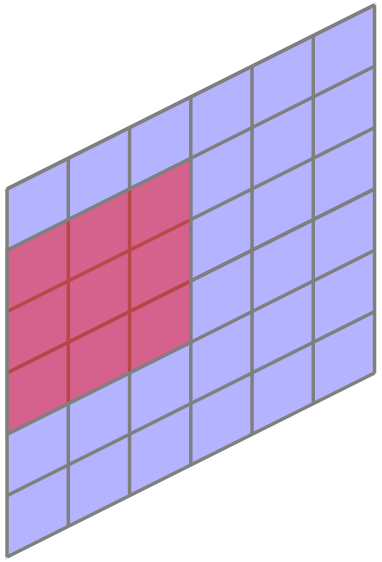


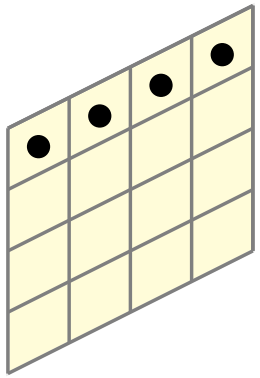

$\mathbf{w}^T\mathbf{x}$

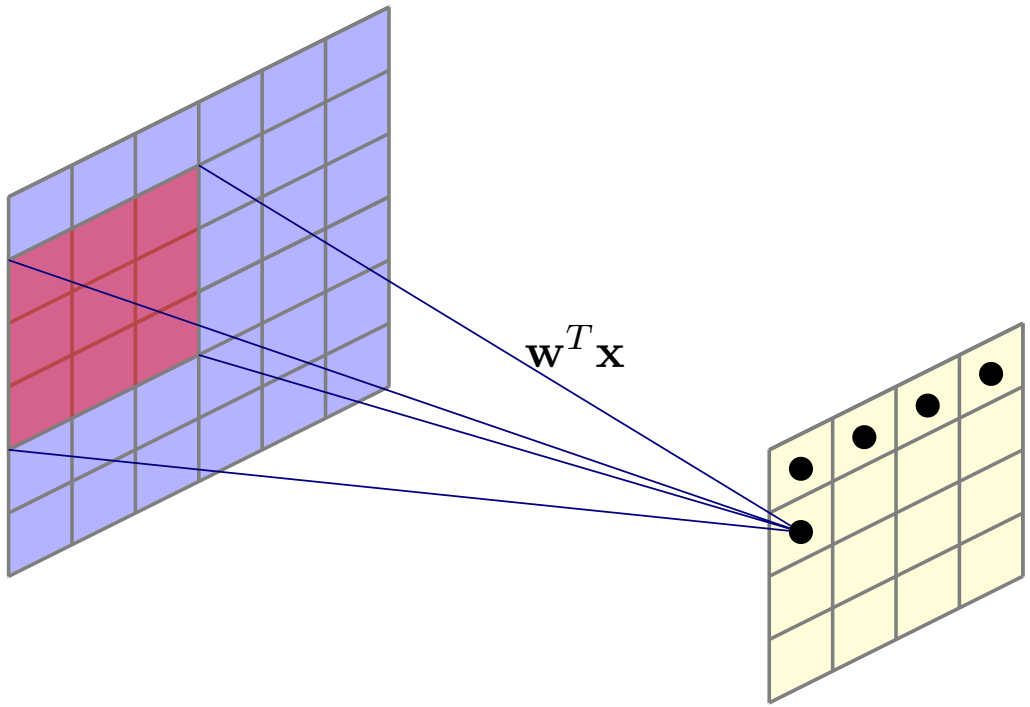


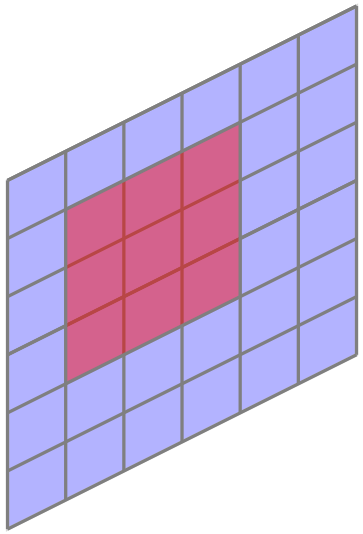


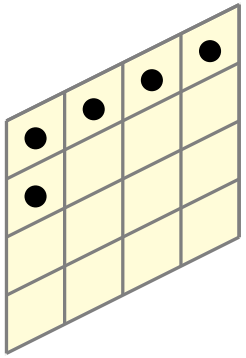

$\mathbf{w}^T\mathbf{x}$

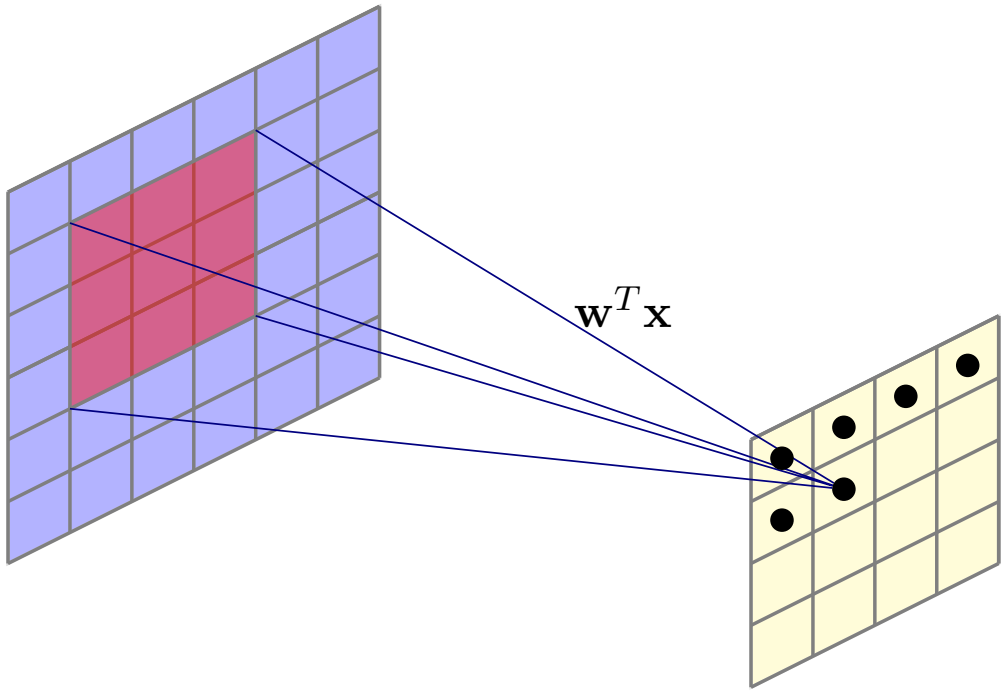


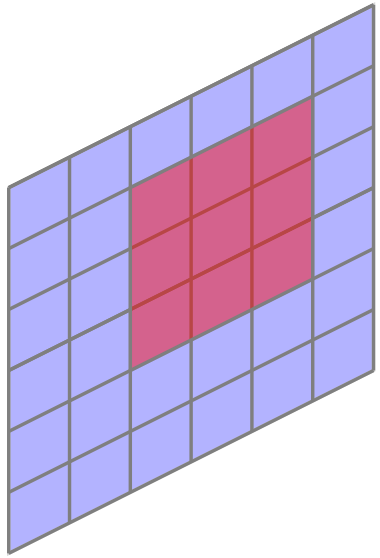


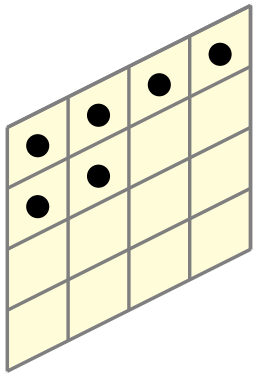

$\mathbf{w}^T\mathbf{x}$

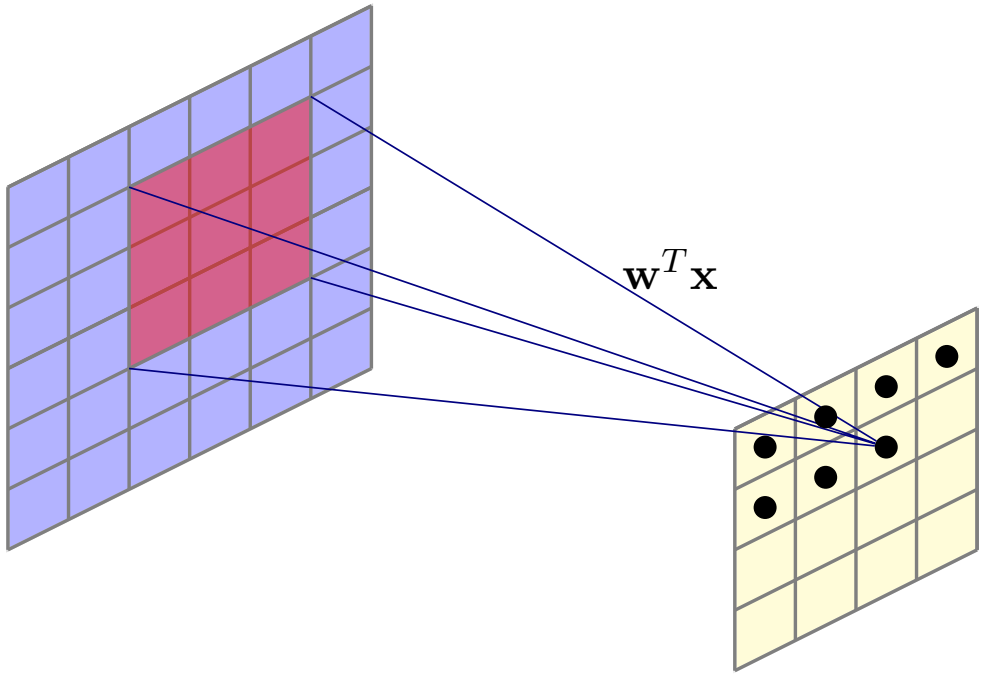


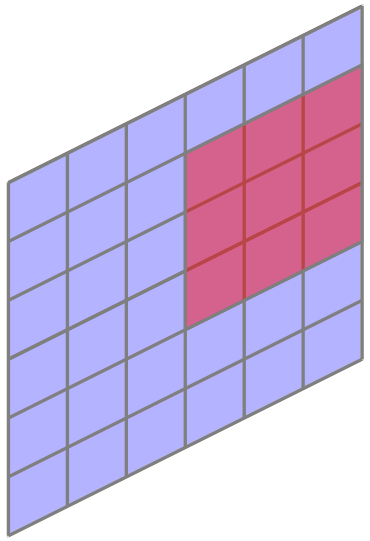


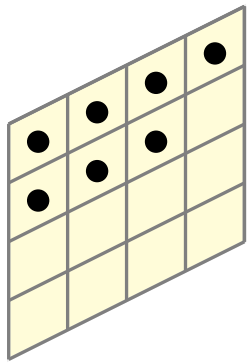

$\mathbf{w}^T\mathbf{x}$

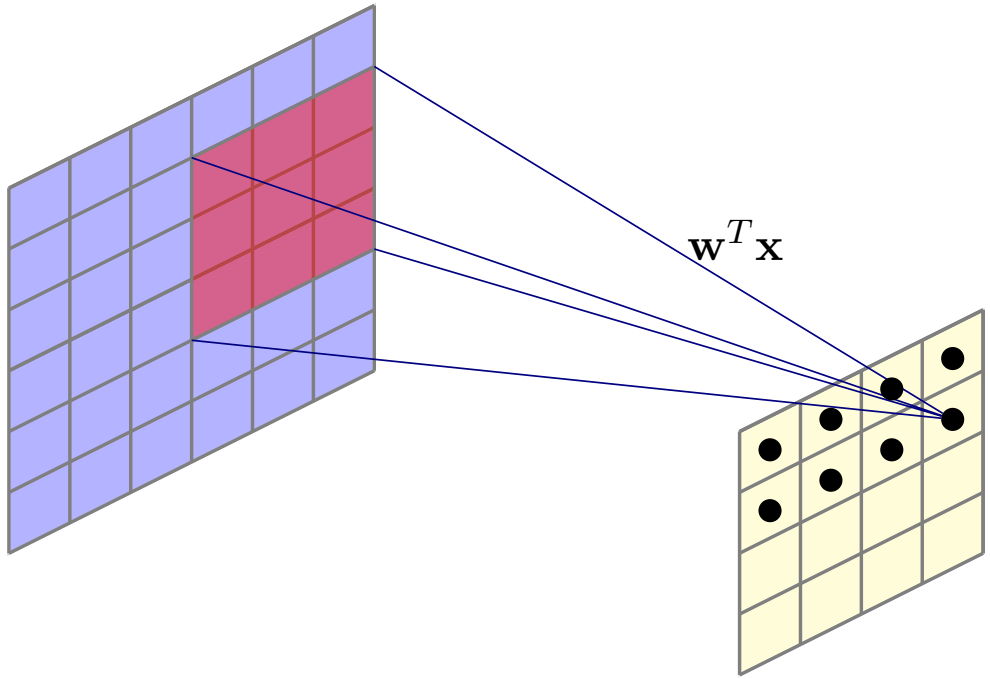


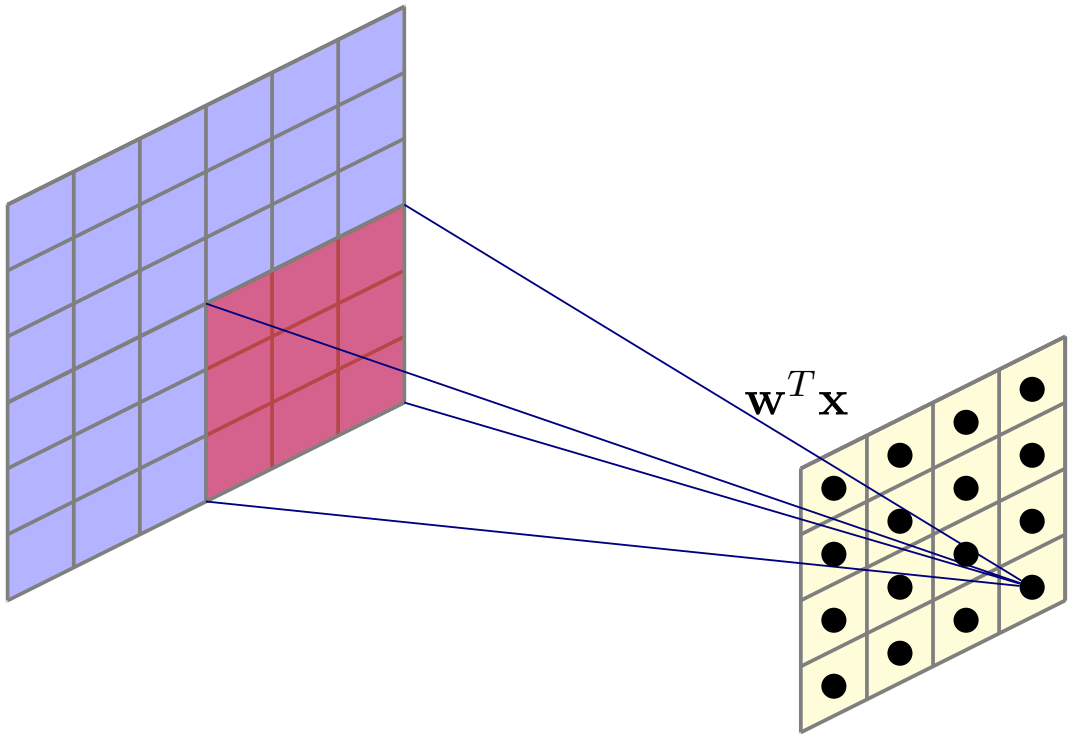



$\mathbf{w}^T\mathbf{x}$



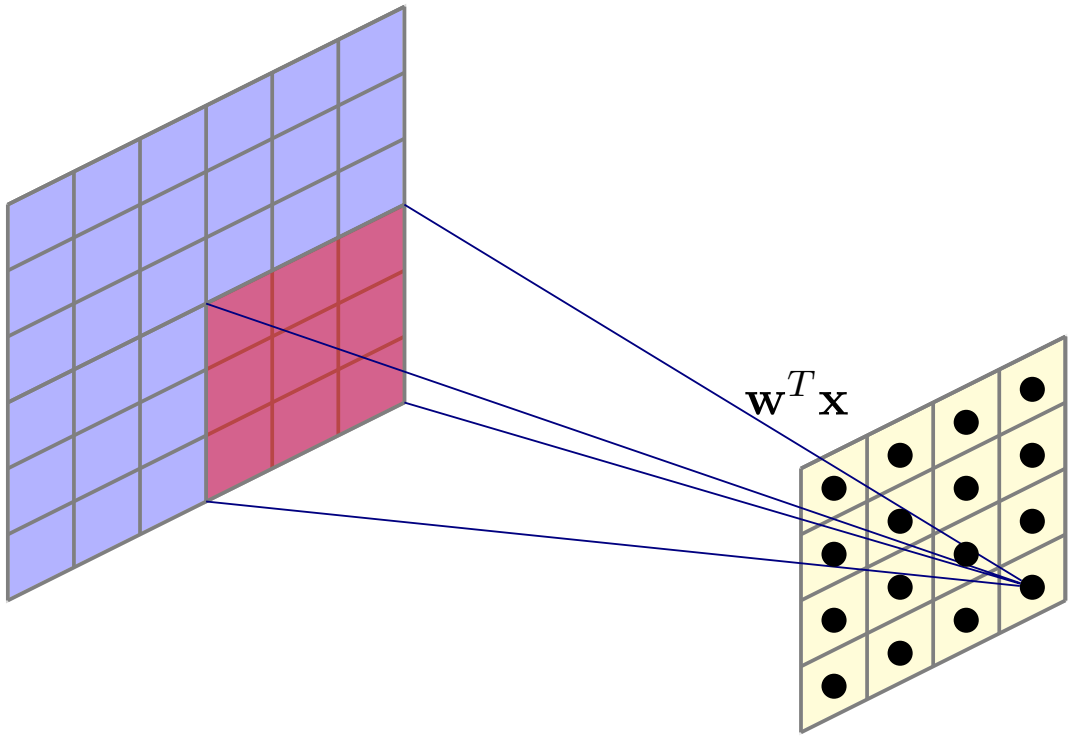



$\mathbf{w}^T\mathbf{x}$


$\mathbf{w}^T\mathbf{x}$

Convolution

- Qual è il numero di parametri?



Convoluzione

$$a_{j,k}^{(h)} = \sum_{l=1}^{c} \sum_{m=1}^{d} w_{m,l} z_{j+l,k+m}^{(h-1)}$$

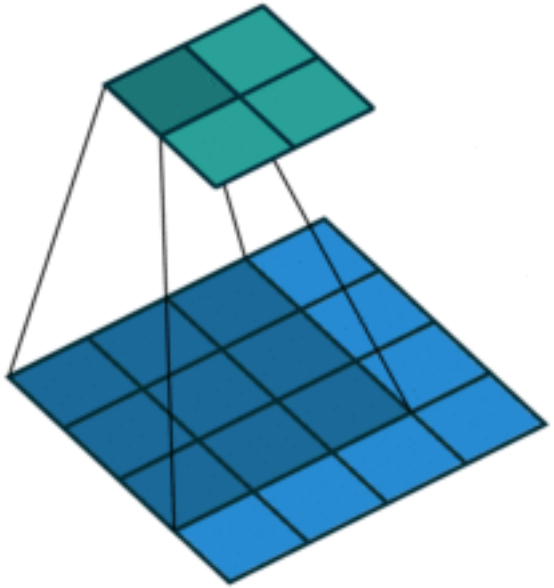
- I pesi rappresentano il kernel di dimensione (c,d)
- Condivisione!

Padding, strides, dilation

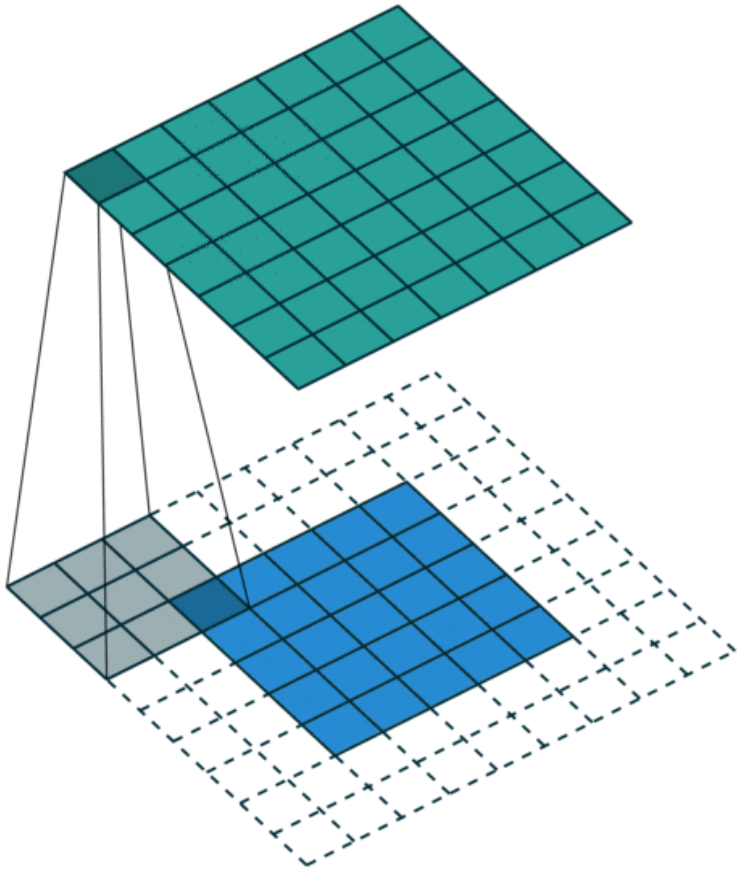
- https://github.com/vdumoulin/conv_arithmetic

1. No padding, no strides
2. Full padding, no strides
3. No padding, strides
4. Padding, strides
5. Dilation

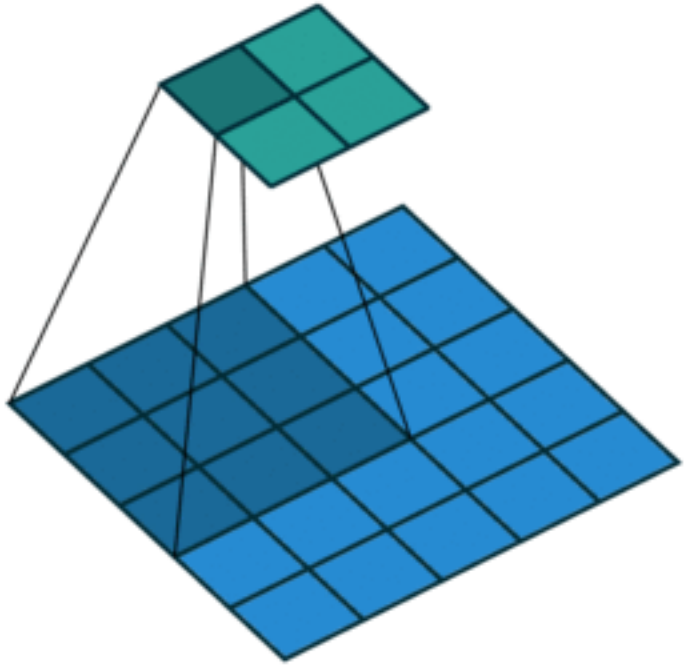
1.No padding, no strides



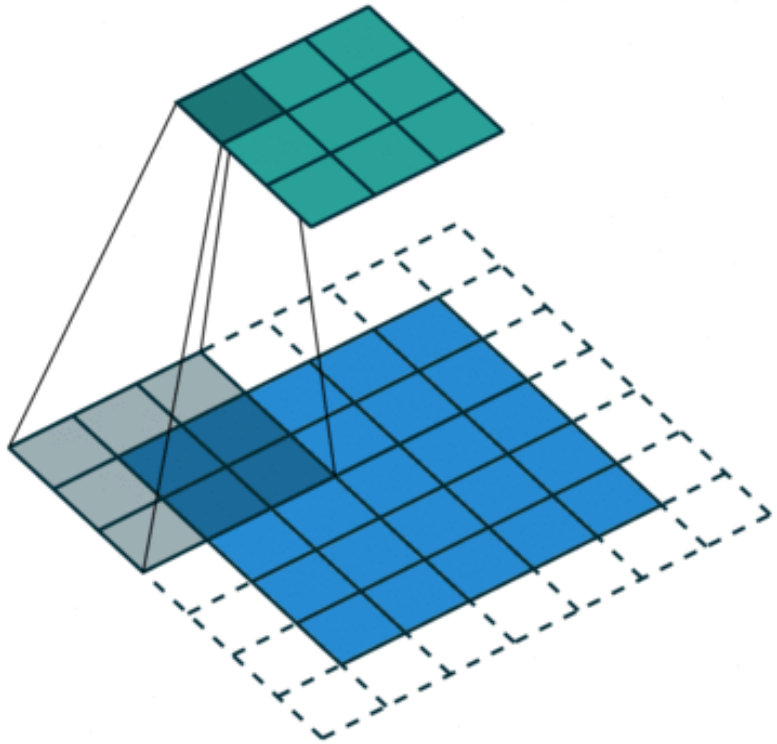
2. Full padding, strides



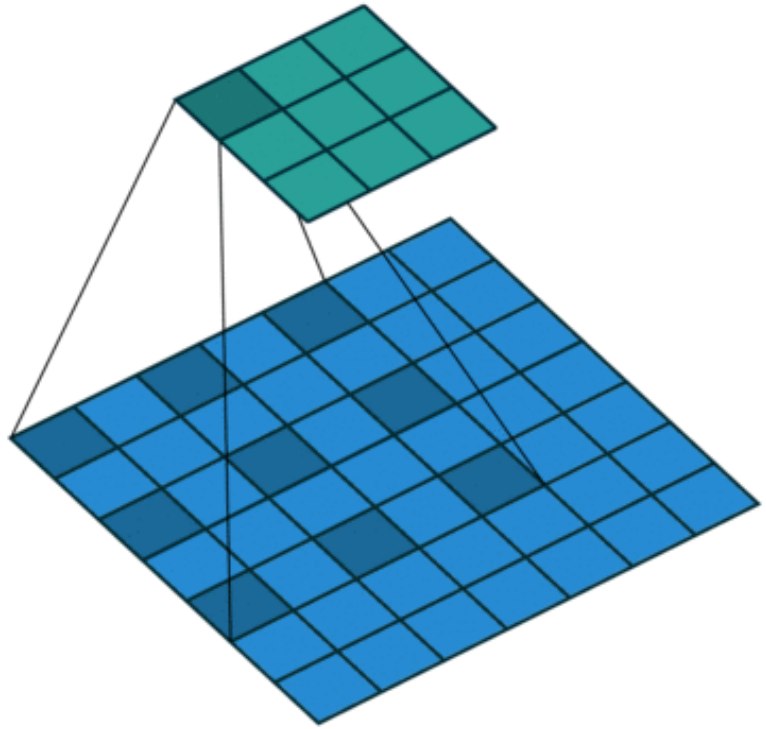
3. No padding, strides



4. Padding, strides



5. Dilation

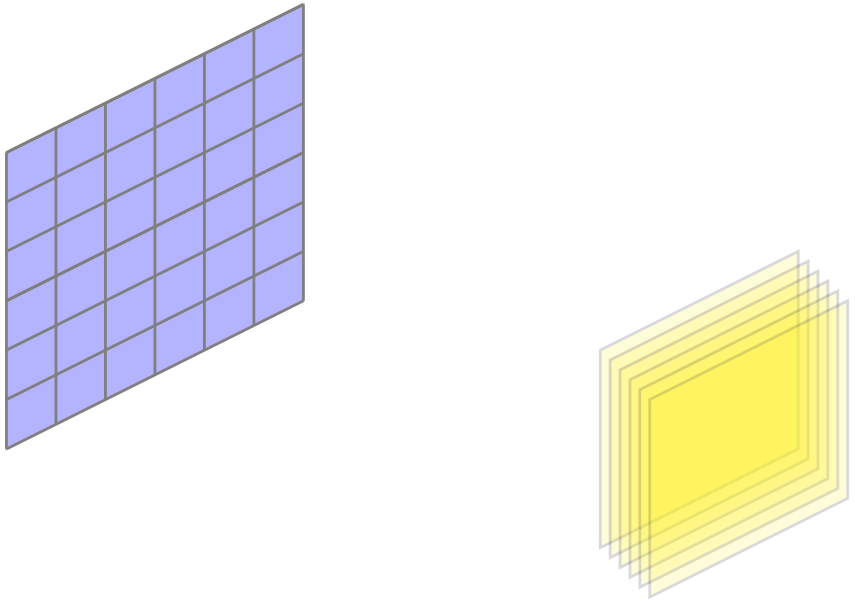


Output, revisited

- Input I
- Padding P
- Kernel size K
- Stride S
- Dilation D
- Output size:

$$\left\lfloor \frac{I - K - (K-1)(D-1) + 2P}{S} \right\rfloor + 1$$

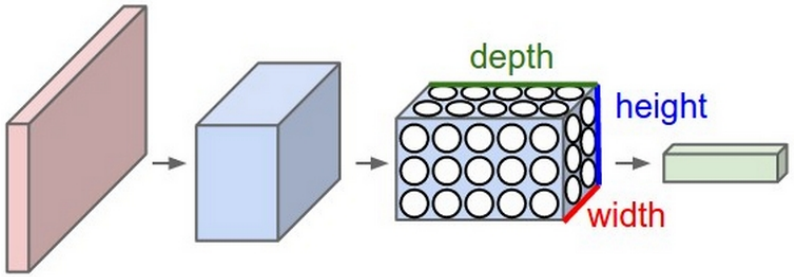
Multiple filters



- Ogni feature map idenfiticata da un kernel
  - In total, il numero dei pesi è dato dal numero di kernel per la size della feature map

Volumetrics

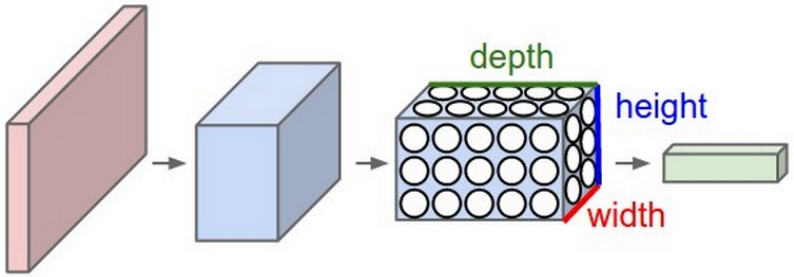
- Non solo immagini 2D
  - Volumi
    - Ad esempio, immagini RGB hanno profondità 3



  - Quanti pesi?

Volumetrics

- Non solo immagini 2D
  - Volumi
    - Ad esempio, immagini RGB hanno profondità 3

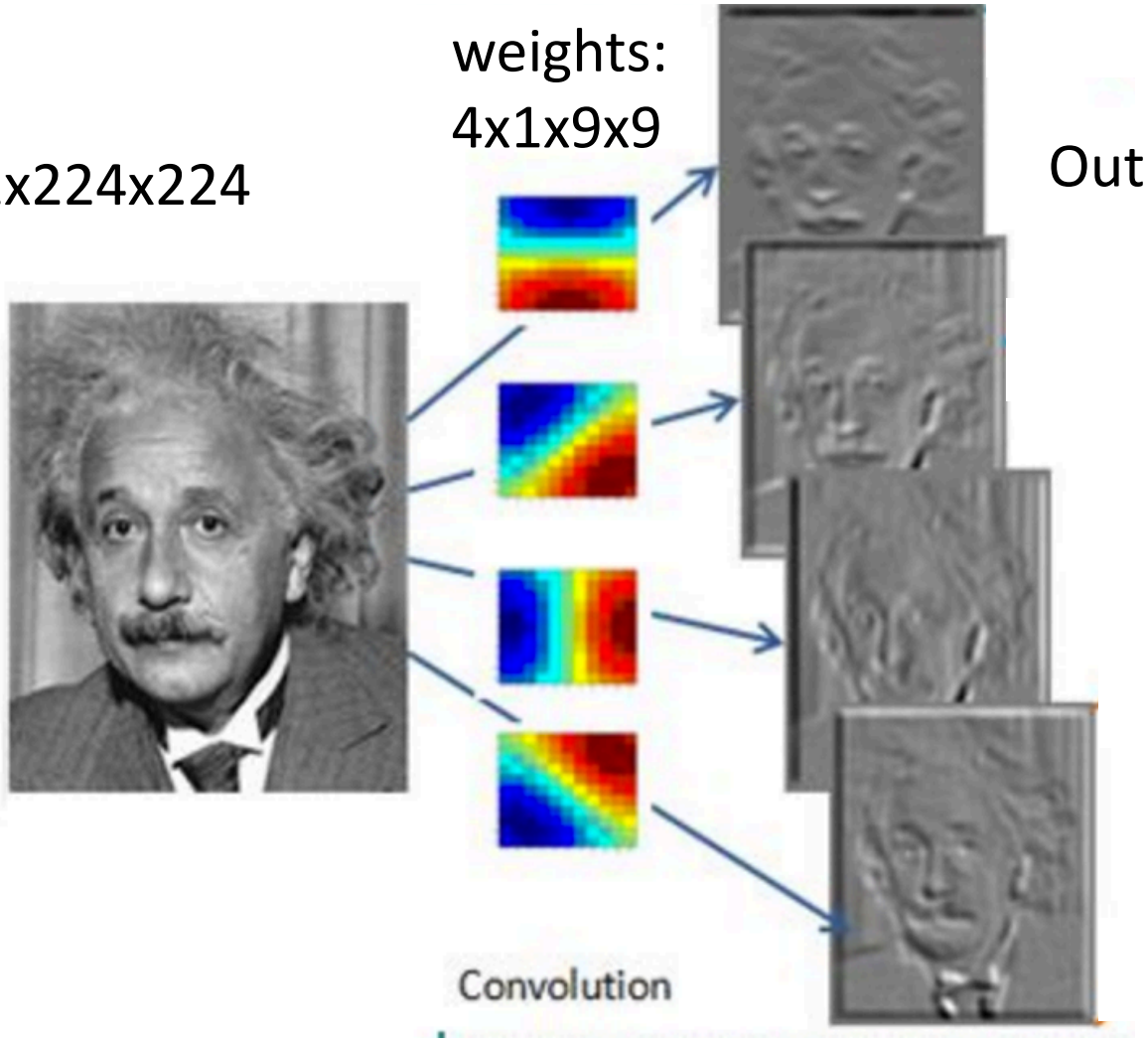


  - Quanti pesi?
    - $a_{i,j}^{f} = \sum_{c} \sum_{l,p} w_{l,p}^{c,f} \cdot x_{i-l,j-p}^{c} + b^{c,f}$

Convolutional Layer (con 4 filtri)

Input: 1x224x224

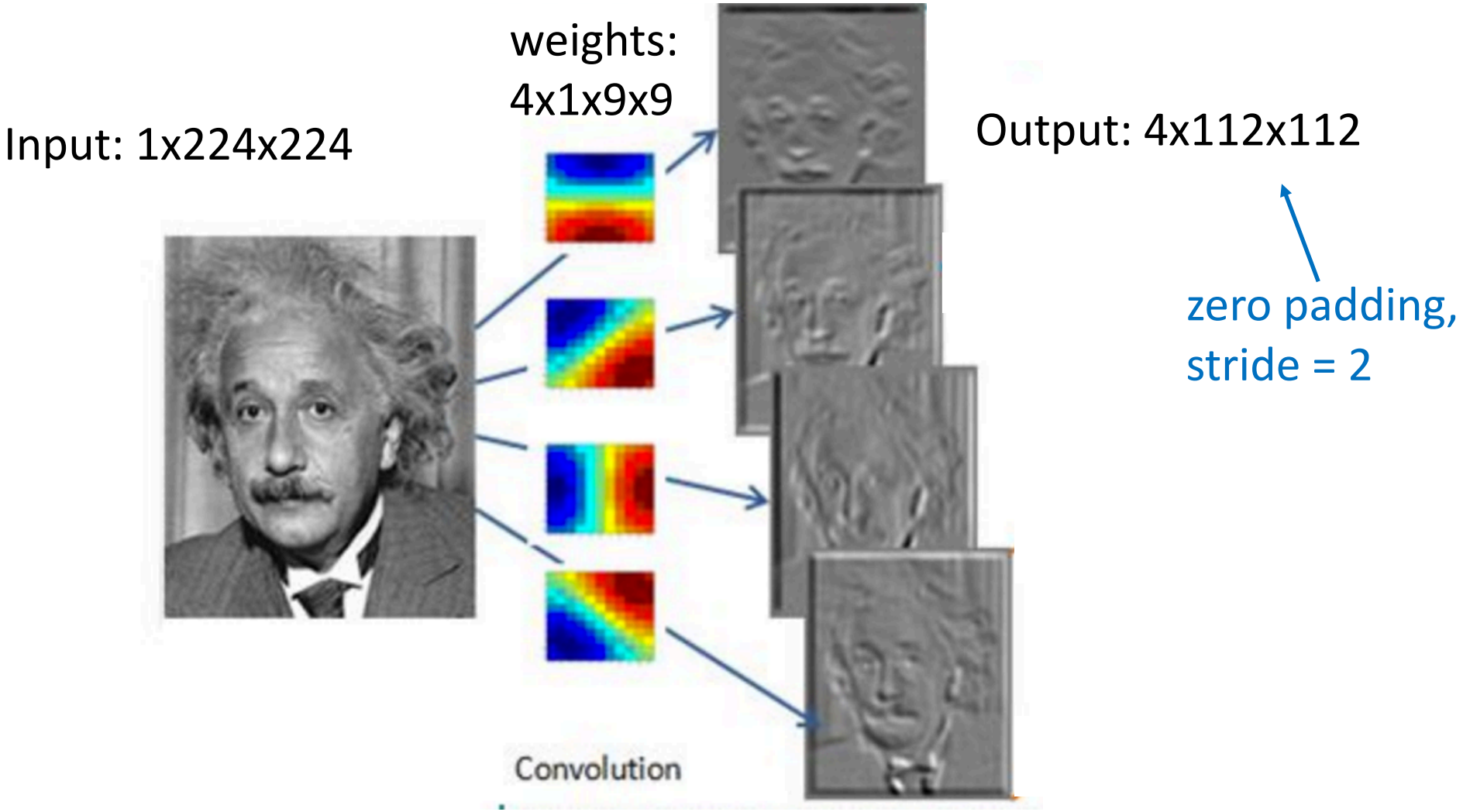
weights:
4x1x9x9



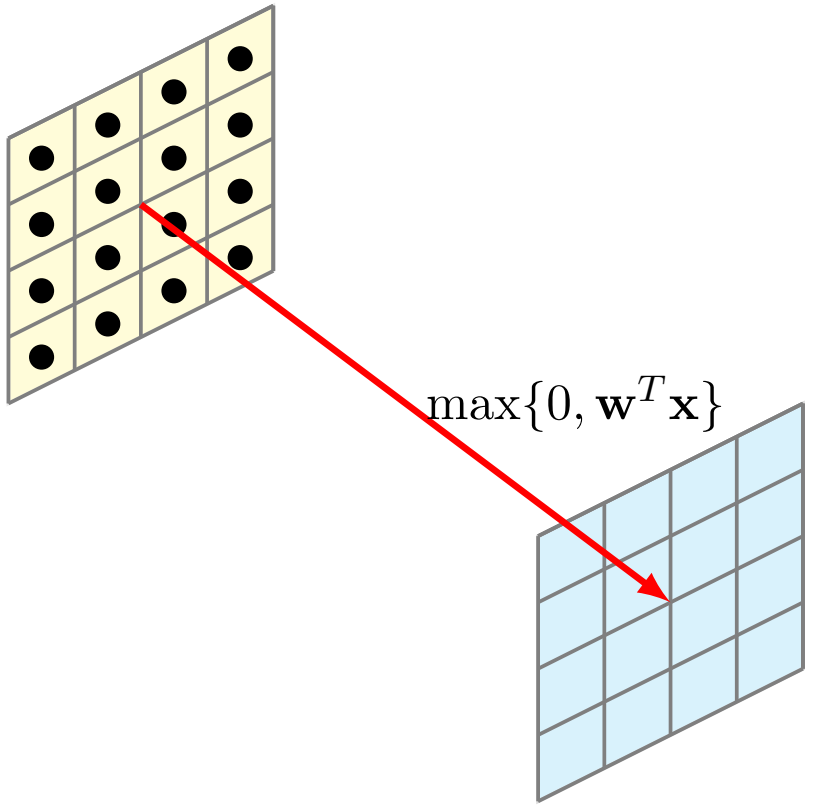
Output: 4x224x224

zero padding,
stride = 1

Convolutional Layer (con 4 filters)

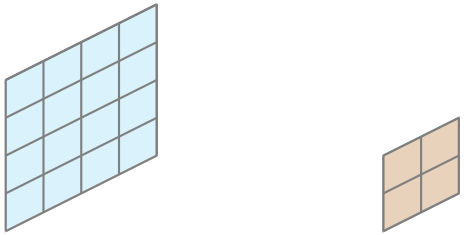


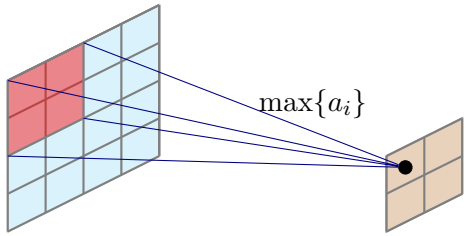
Activation Layer

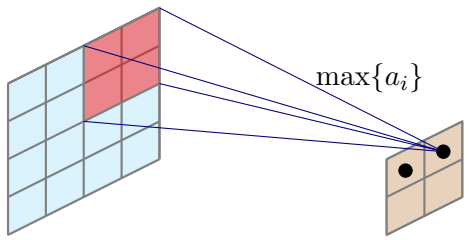


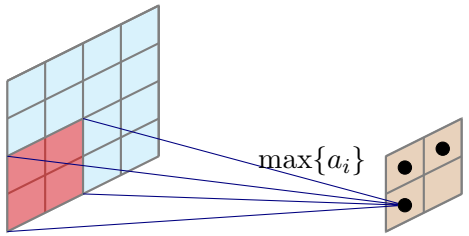
- ReLU

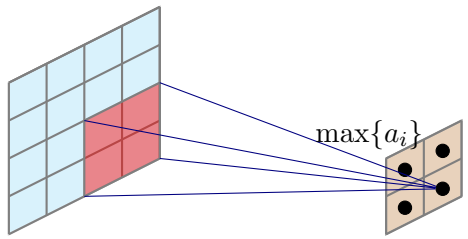
Pooling



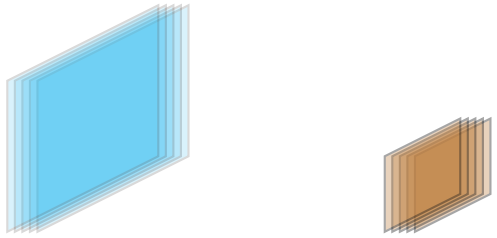

$\max\{a_i\}$


$\max\{a_i\}$


$\max\{a_i\}$

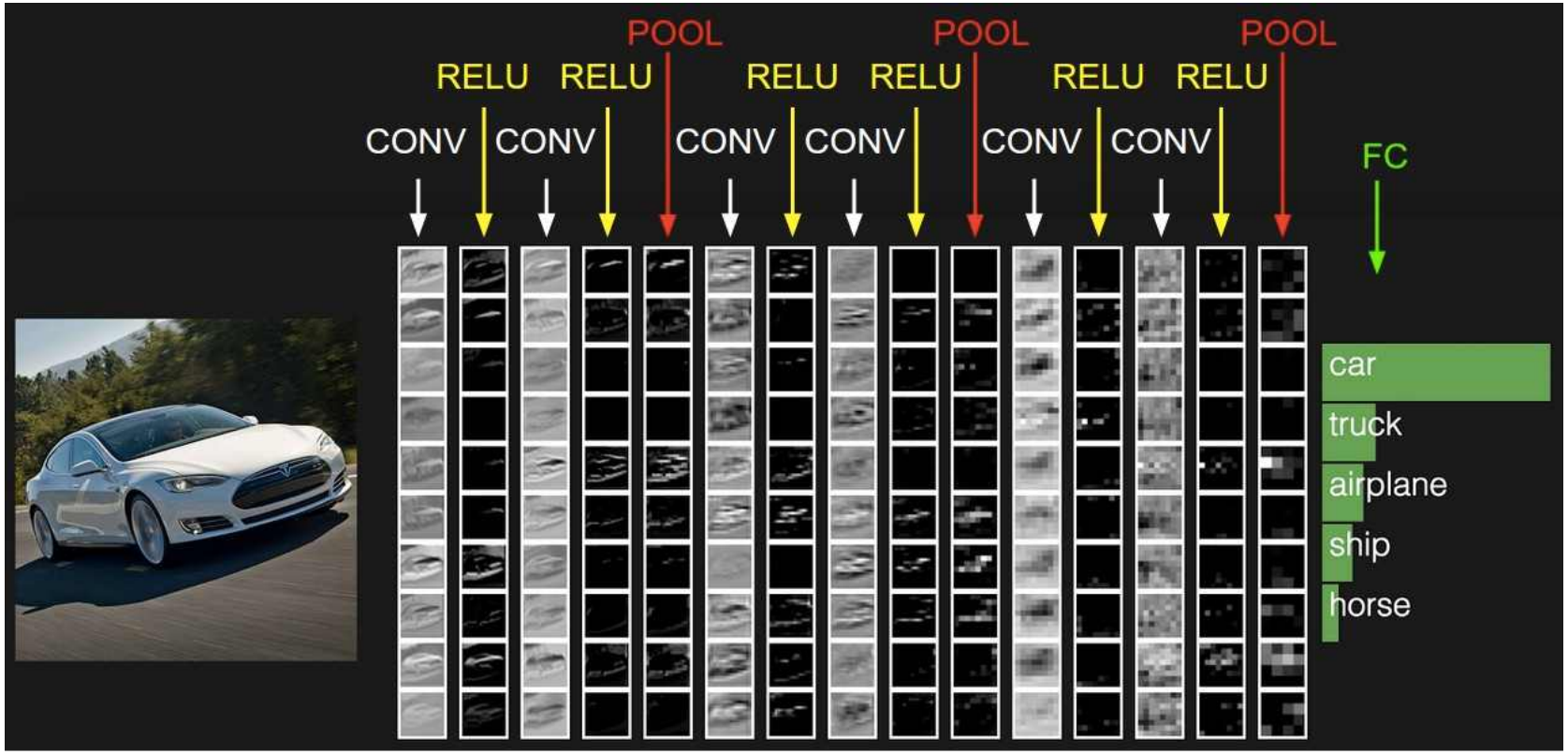

$\max\{a_i\}$

Pooling



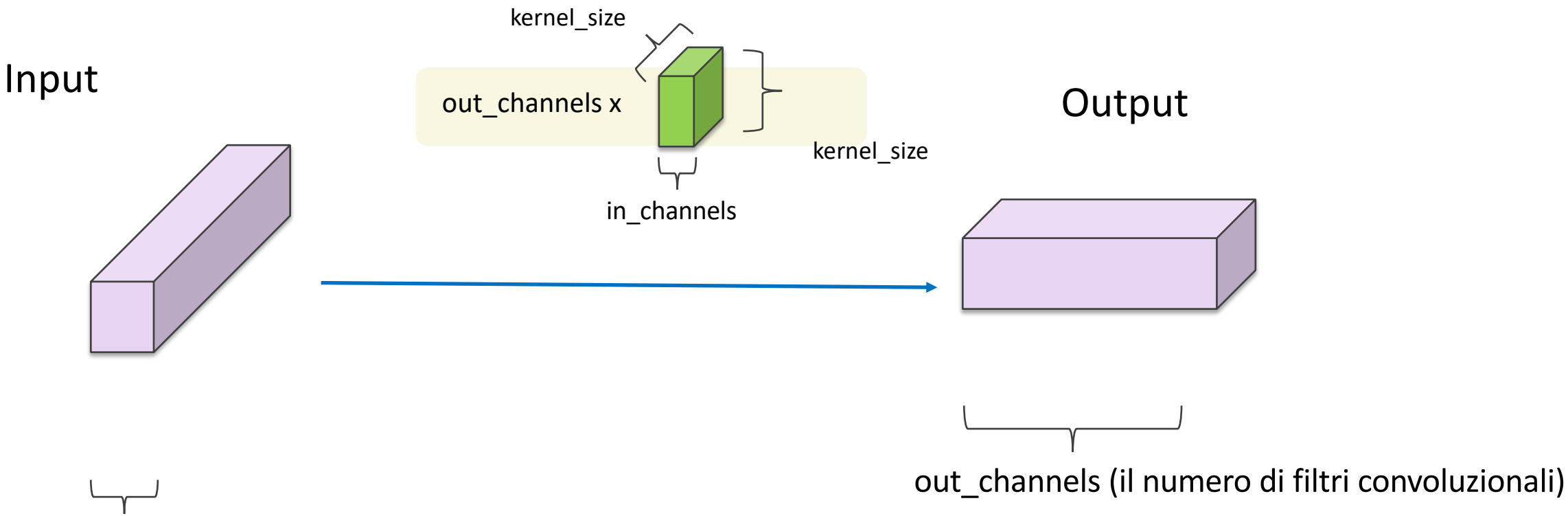
- Multiple feature maps, multiple poolings
- Max, average, L2, …

Convolutional neural networks

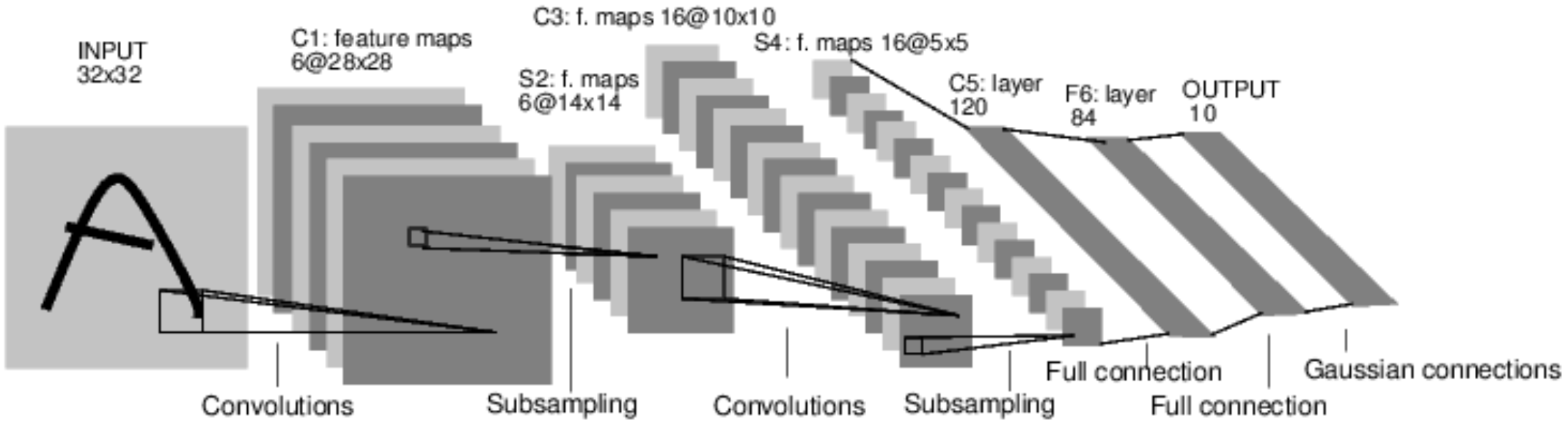


Convolutional Layer in pytorch

```
class torch.nn.Conv2d(in_channels, out_channels, kernel_size, stride=1, padding=0, dilation=1,
groups=1, bias=True)  [source]
```



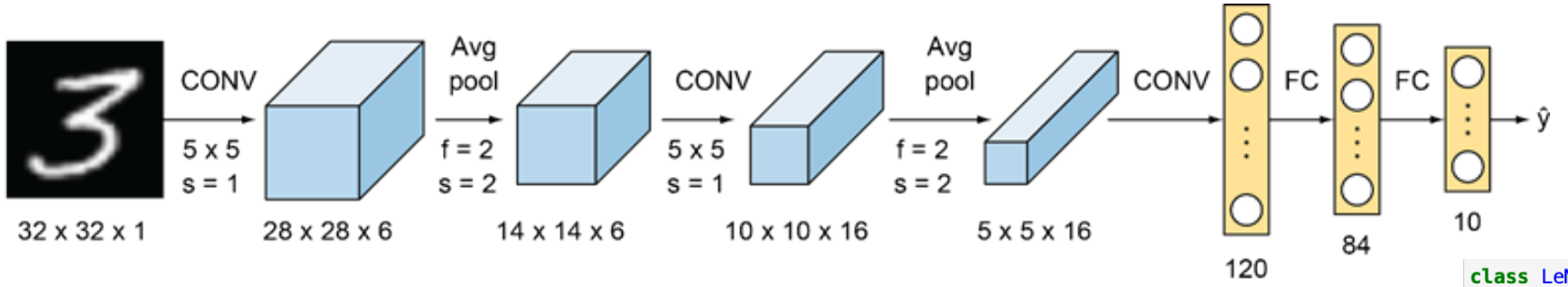
Convolutional Network: LeNet



Yann LeCun

TITLE	CITED BY	YEAR
Gradient-based learning applied to document recognition Y LeCun, L Bottou, Y Bengio, P Haffner Proceedings of the IEEE 86 (11), 2278-2324	11736	1998

LeNet in Pytorch




```python
class LeNet(nn.Module):
    def __init__(self,input_size):
        super(LeNet, self).__init__()
        # Convolutional Layers
        self.features = nn.Sequential(
            nn.Conv2d(1, 6, 5),
            nn.Tanh(),
            nn.AvgPool2d(2,stride = 2),
            nn.Conv2d(6, 16, 5),
            nn.Tanh(),
            nn.MaxPool2d(2,stride = 2)
        )
        fm_size = ((input_size - 6 )//2 - 5)//2 + 1
        fc_layer_in_size = 16*fm_size*fm_size

        # Linear layers
        self.fc = nn.Sequential(
            nn.Linear(fc_layer_in_size, 120),
            nn.Tanh(),
            nn.Linear(120, 84),
            nn.Tanh(),
            nn.Linear(84, 10)
        )


    def forward(self, x):
        features = self.features(x)

        # Flatten the tensor along the second dimension
        features_flattened = features.view(features.size(0),-1)

        out = self.fc(features_flattened)

        output = F.log_softmax(out, dim=1)

        return output
```

LeNet Summary

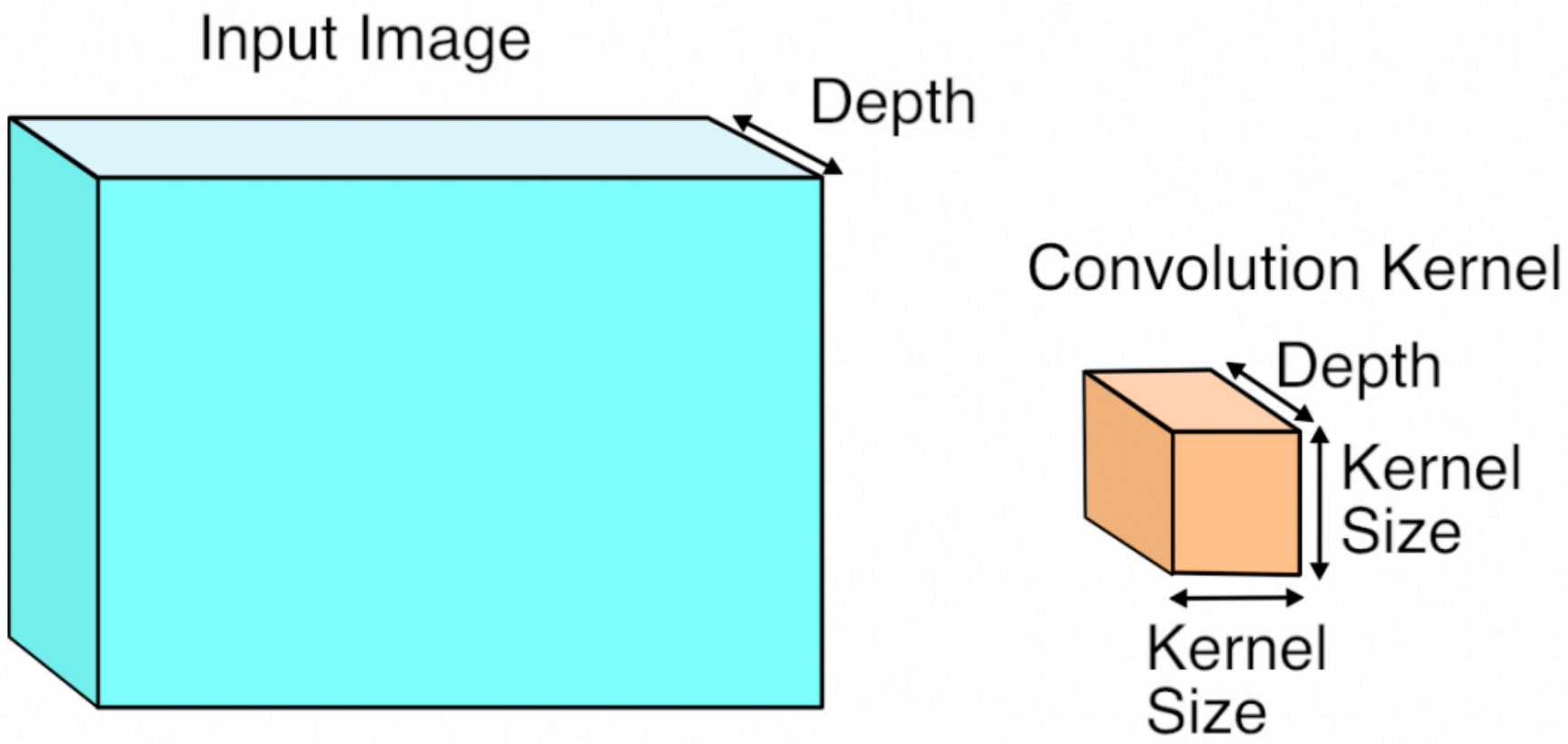
- 2 Convolutional Layers + 3 Linear Layers

- + Non-linear functions: ReLUs or Sigmoids
  + Max-pooling operations

Esercizio

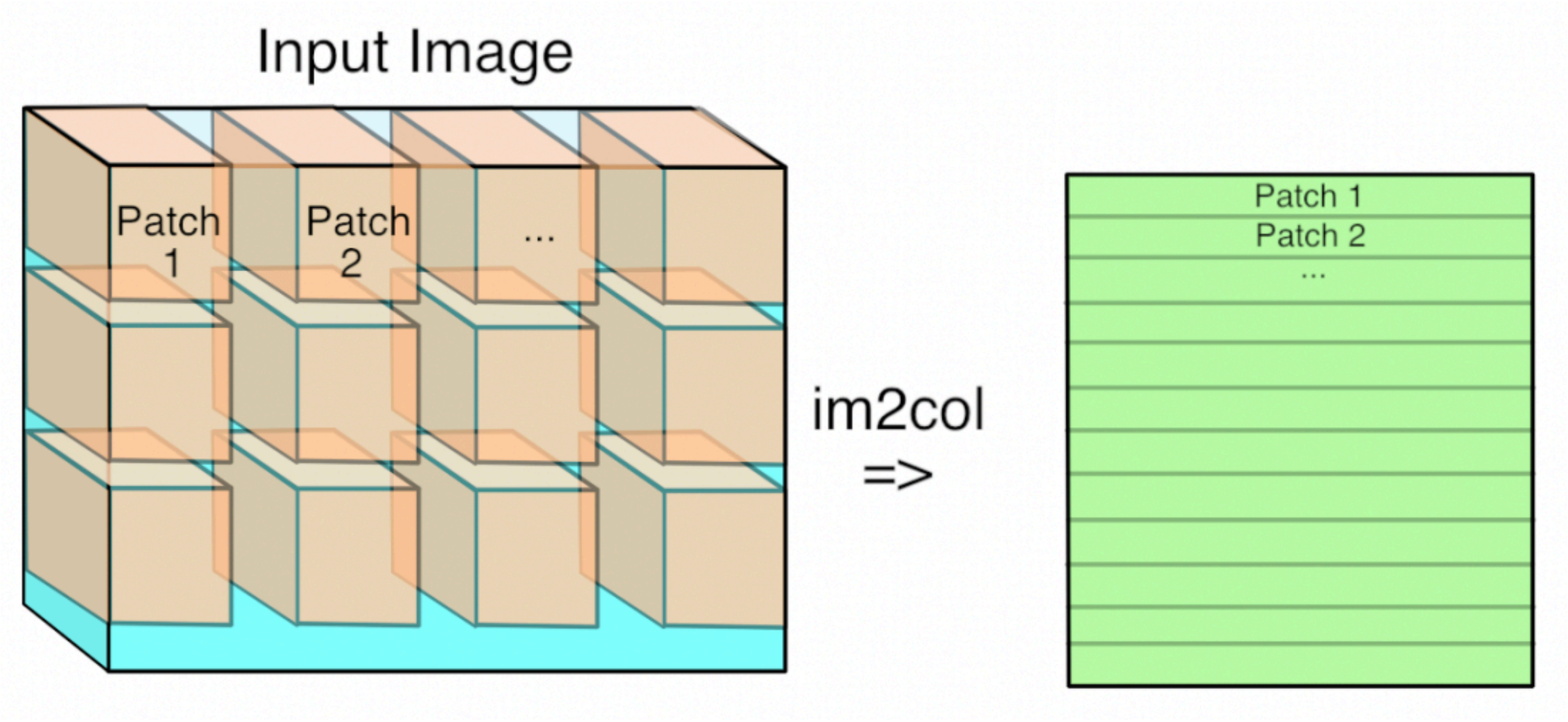
- Adattare la rete LeNet per effettuare classificazione sul dataset CIFAR10
  - CIFAR-10 consiste di 60000 immagini 32x32 (RGB), etichettate con un intero che corrisponde a 10 classi: airplane (0), automobile (1), bird (2), cat (3), deer (4), dog (5), frog (6), horse (7), ship (8), truck (9).

Convolutional Layers, Matrix Multiplication



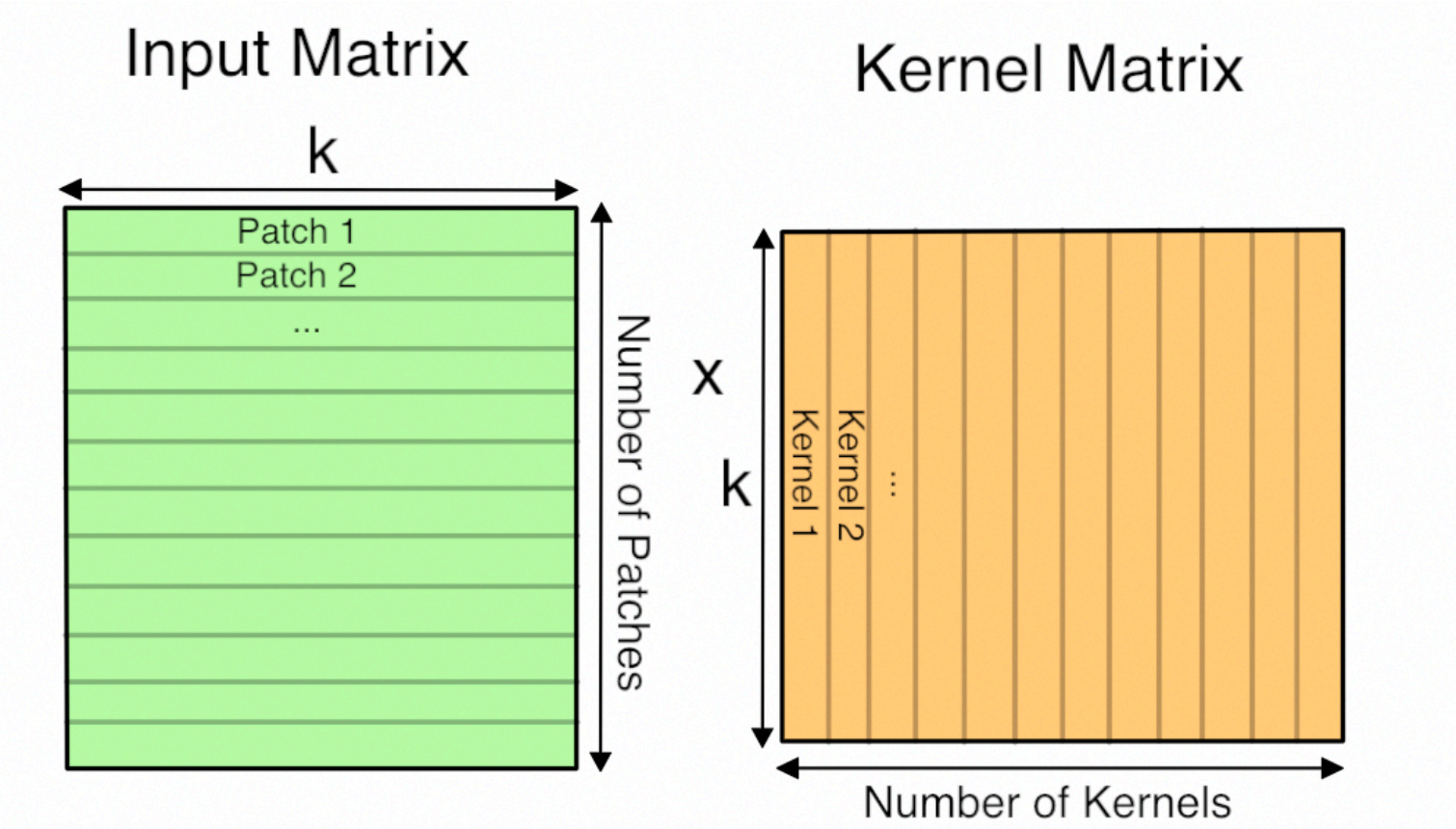
https://petewarden.com/2015/04/20/why-gemm-is-at-the-heart-of-deep-learning/

Convolutional Layers, Matrix Multiplication



https://petewarden.com/2015/04/20/why-gemm-is-at-the-heart-of-deep-learning/

Convolutional Layers, Matrix Multiplication



https://petewarden.com/2015/04/20/why-gemm-is-at-the-heart-of-deep-learning/

Conviene usare le CNN?

- Altamente parallelizzabili
- GPU Computing
- CPU Computing proibitivo

Perché CNNs?

- Sparse interactions
  - Meno parametri
- Parameter sharing
  - Kernel condivisi lungo tutta l’immagine
- Invarianza di traslazione
- Possibilità di lavorare con input di dimensione variabile