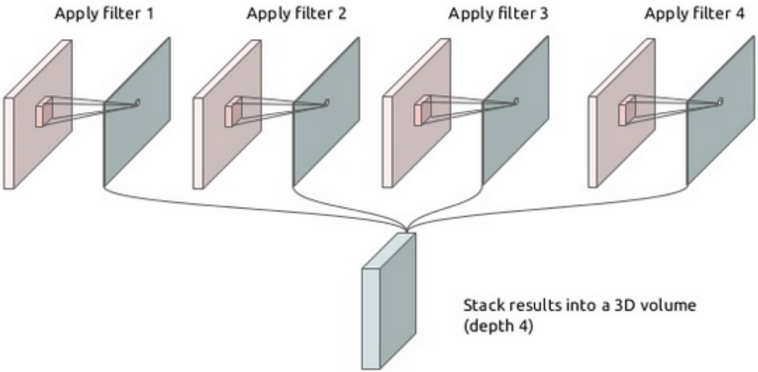
Neural Networks and Deep Learning: Convolution Layer

Nicolas Thome

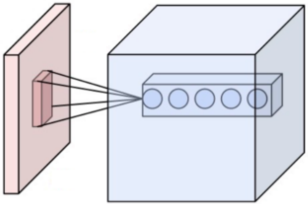
Conservatoire National des Arts et Métiers (Cnam)
Département Informatique

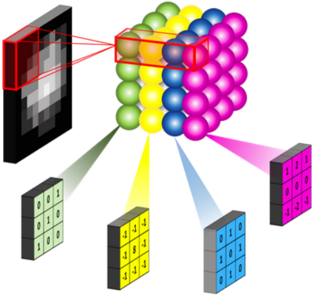
Convolution Layer

- 2D convolution: each filter $\Rightarrow$ 2D map (image)
- Convolution Layer: staking maps from multiple Filters
  $\Rightarrow$ **Tensor: multi-dimensional array**



Convolution Layer



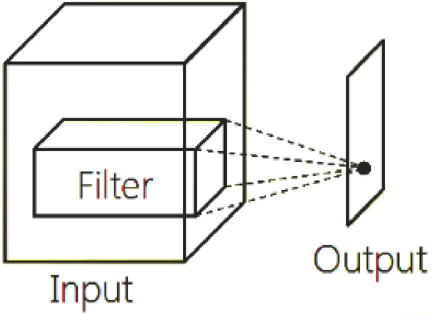


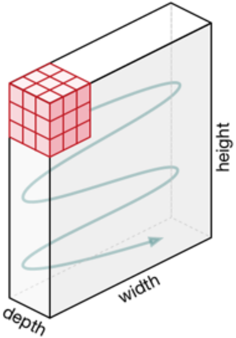
- Tensor: stacking several filters outputs
  - Depth $\Leftrightarrow$ # filters
  - Each spatial position: output for the different filters
- Ex: 2D convolution with gray-scale images
  - Input tensor depth $= 1$
- **Convolution on color images / hierarchies:**
  - **Convolution on tensors!**
  - **Input Tensor $\Rightarrow$ output Tensor**

Convolution Layer for Tensors

$$f'(i,j) = (f \star h)(i,j) = \sum_{k=1}^{K} \sum_{n=-\frac{d-1}{2}}^{\frac{d-1}{2}} \sum_{m=-\frac{d-1}{2}}^{\frac{d-1}{2}} f(i-n, m-j, k) h(n,m,k) \; {\color{red} + \; \mathbf{b}}$$

- **Convolution: linear, bias b $\Rightarrow$ affine**

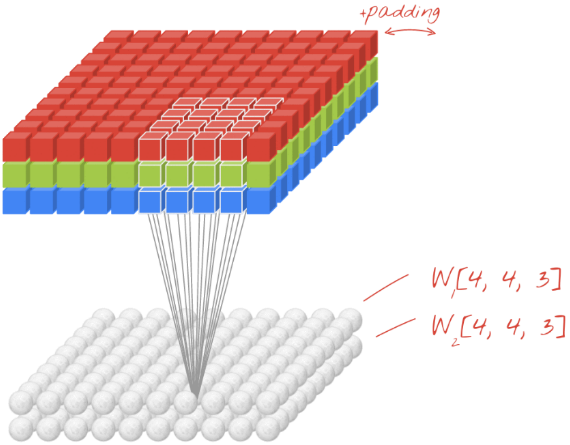




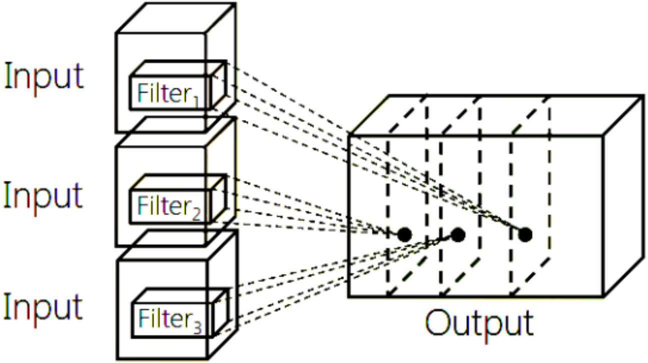
- **Filtering on depth:** correlation between feature maps

Convolution Layer for Tensors

Ex: input color image

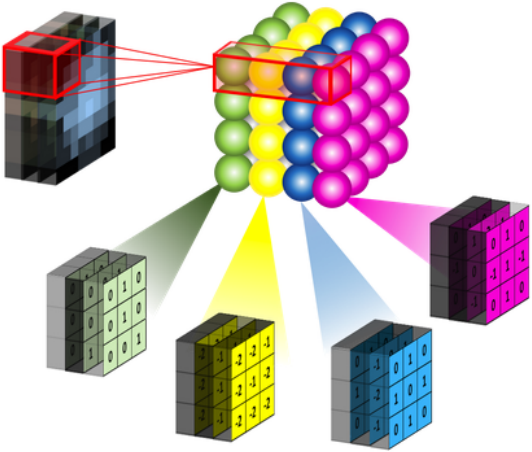


Natural extension for multiple filters

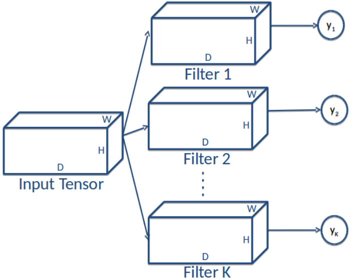


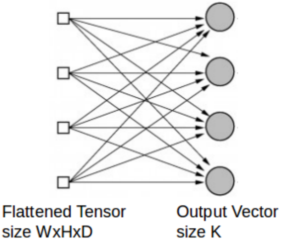
Convolution Layer for Tensors

Ex: input color image



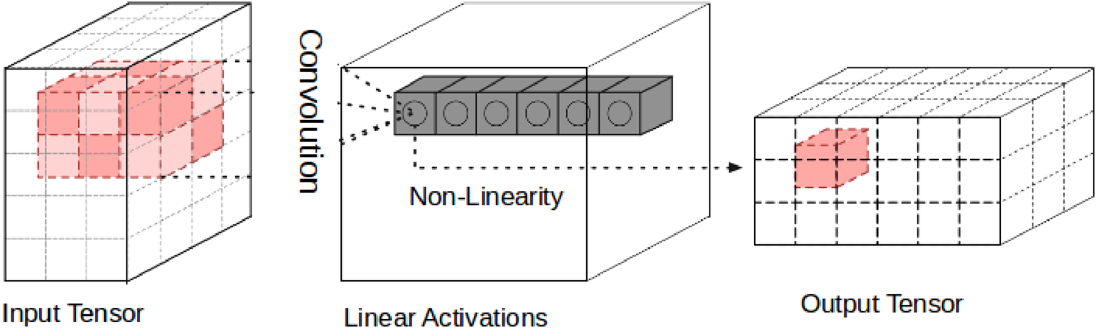
Specific Tensor Convolution Filters





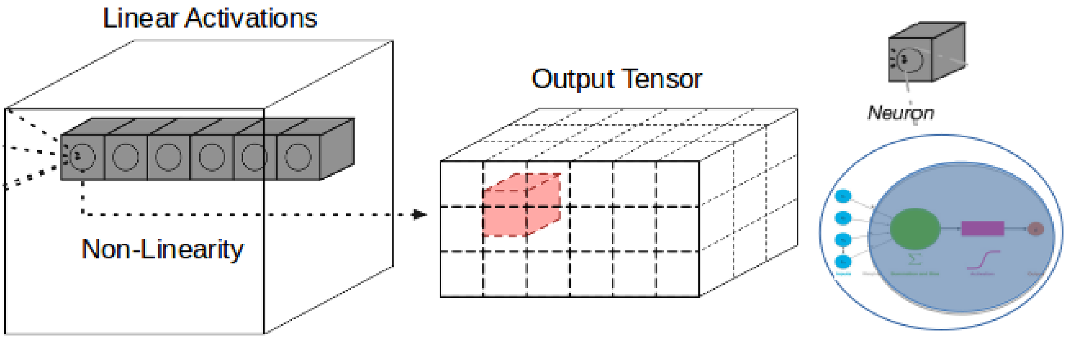
- Input tensor size $W \times H \times D$
- Filter size $= W \times H \times D =$ tensor size, no padding
  $\Rightarrow$ No possible displacement for filter
  - Output: single scalar value
  - Use of K filters $\Rightarrow$ output: K-dim vector
- **Convolution $\sim$ fully connected on flattened tensor**

Convolution Layer: Non-Linearity



- Convolutional Layer:
  Input Tensor → Output Tensor
  1. Convolution: linear / affine filtering
  2. Follwed by point wise non-linearity
     ~ non-linearity on spatial maps

Convolution Layer: Non-Linearity



- Each activation in tensor map $\Leftrightarrow$ formal neuron
- Ex: sigmoid activation:

$$\sigma(z) = (1 + e^{-az})^{-1}$$

Convolution Layer: Conclusion

- Convolution for tensors: generalizes 2D convolution (depth)
- **Convolution Layer: Core of Deep Learning**
- Deep: Convolution Hierarchies?
  $\Rightarrow$ **next!**