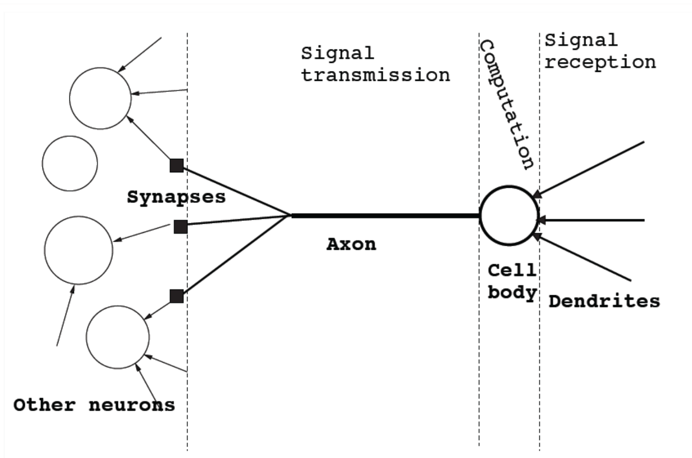
Day 1 Lecture 3

Deep Networks

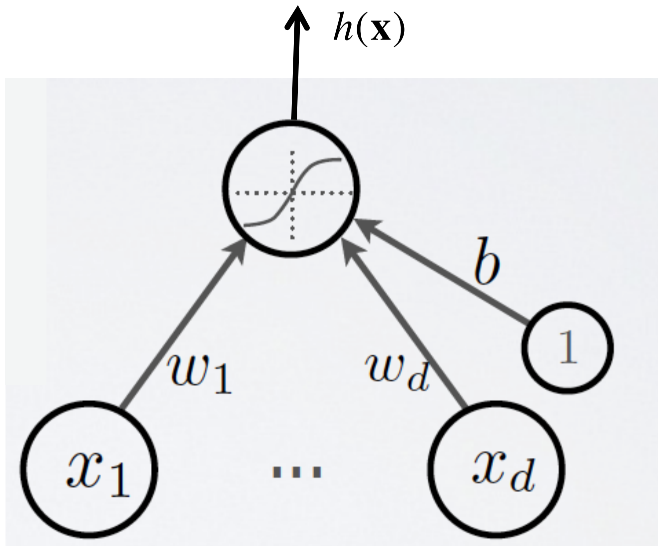
Elisa Sayrol

UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH
Department of Signal Theory and Communications
Image Processing Group

From Neurons to Convolutional Neural Networks



Figures Credit: Hugo Laroche NN course



$$a(\mathbf{x}) = b + \sum_j w_j x_j = b + \mathbf{w}^{\mathrm{T}}\mathbf{x}$$

$$h(\mathbf{x}) = g(a(\mathbf{x})) = g(b + \sum_j w_j x_j)$$

From Neurons to Convolutional Neural Networks

Hidden pre-activation

$$\mathbf{a}(\mathbf{x}) = \mathbf{b}^{(1)} + \mathbf{W}^{(1)}\mathbf{x}$$

$$a(\mathbf{x}) = b_i^{(1)} + \sum_j W_{i,j}^{(1)} x_j$$

Hidden activation

$$\mathbf{h}(\mathbf{x}) = \mathbf{g}(\mathbf{a}(\mathbf{x}))$$

g(**x**) activation function:

sigmoid: $g(a) = sigm(a) = \frac{1}{1+\exp(-a)}$

tangh: $g(a) = \tanh(a)$

ReLU: $g(a) = \max(0,a)$

Output activation

$$f(\mathbf{x}) = o(\mathbf{b}^{(2)} + \mathbf{W}^{(2)}\mathbf{h}(\mathbf{x}))$$

o(**x**) output activation function:

Softmax: $o(\mathbf{a}) = \text{softmax}(\mathbf{a}) = \left[\frac{\exp(a_1)}{\sum_c \exp(a_c)} \cdots \frac{\exp(a_C)}{\sum_c \exp(a_c)}\right]^{\mathrm{T}}$

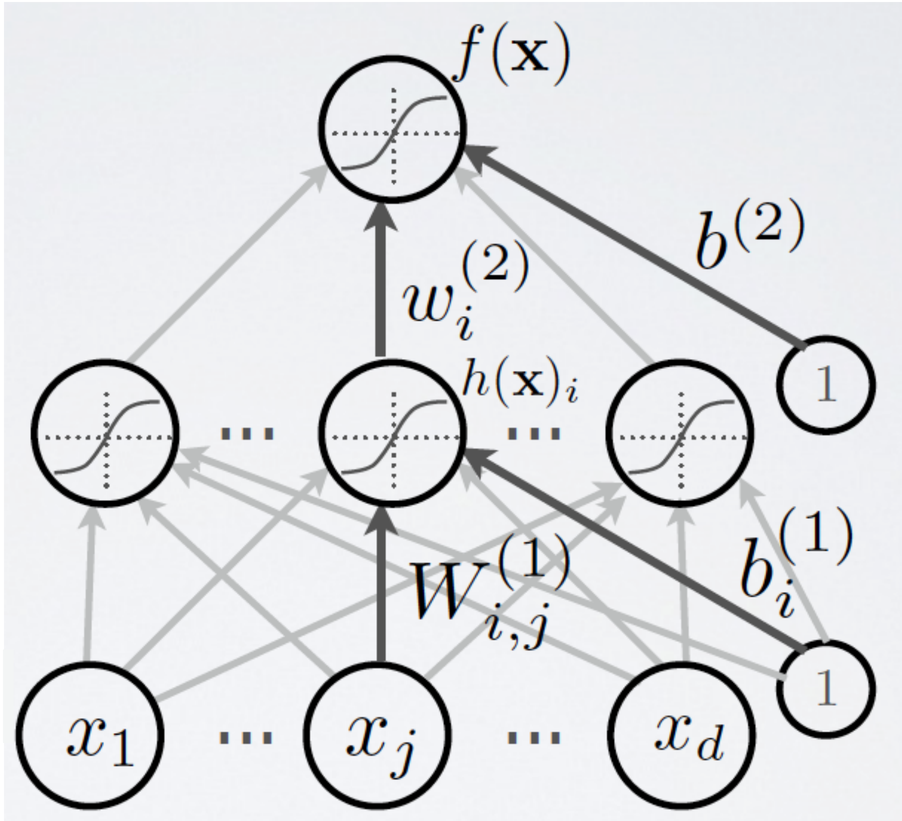


Figure Credit: Hugo Laroche NN course

From Neurons to Convolutional Neural Networks

L Hidden Layers

Hidden pre-activation ($k>0$)

$$\mathbf{a}^{(k)}(\mathbf{x}) = \mathbf{b}^{(k)} + \mathbf{W}^{(k)}\mathbf{h}^{(k-1)}(\mathbf{x})$$

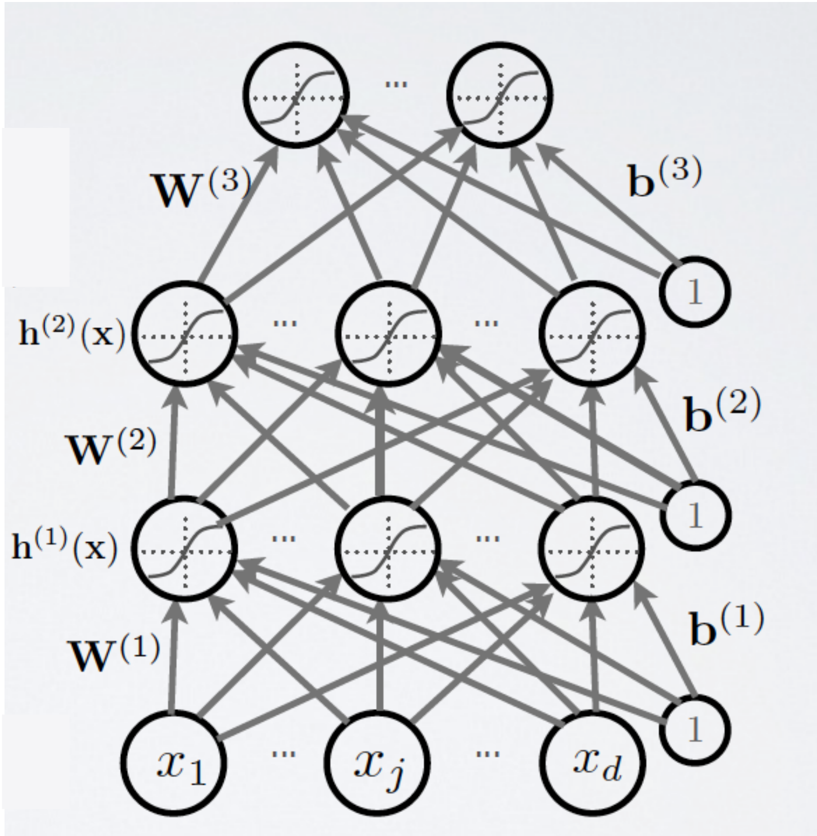
$$\mathbf{h}^{(0)}(\mathbf{x}) = \mathbf{x}$$

Hidden activation ($k=1,\ldots L$)

$$\mathbf{h}^{(k)}(\mathbf{x}) = \mathbf{g}(\mathbf{a}^{(k)}(\mathbf{x}))$$

Output activation ($k=L+1$)

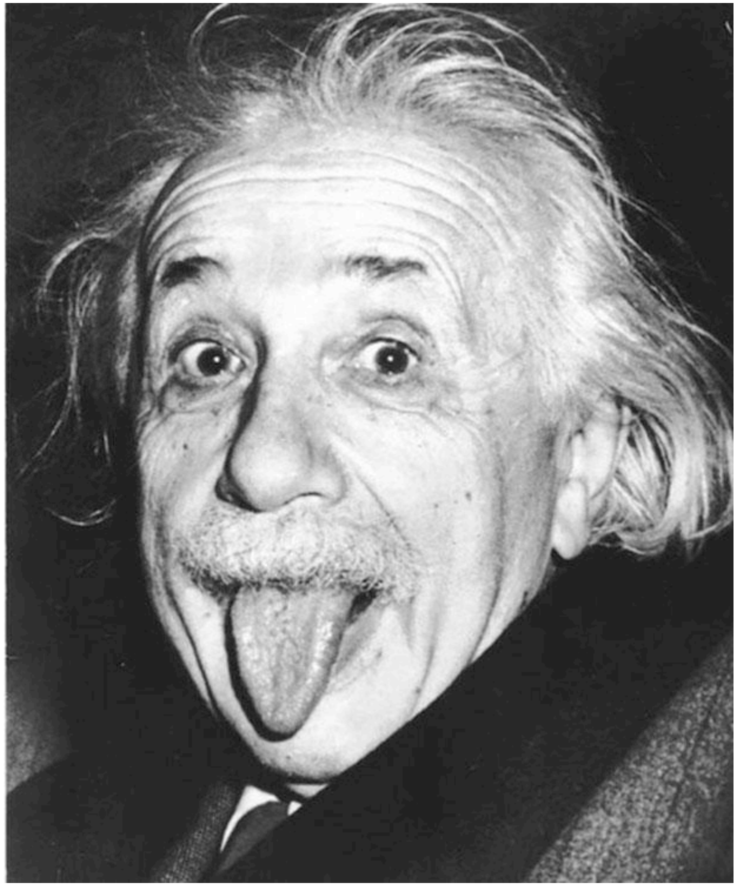
$$\mathbf{h}^{(L+1)}(\mathbf{x}) = \mathbf{o}(\mathbf{a}^{(L+1)}(\mathbf{x})) = \mathbf{f}(\mathbf{x})$$



Slide Credit: Hugo Laroche NN course

From Neurons to Convolutional Neural Networks

What if the input is all the pixels within an image?



From Neurons to Convolutional Neural Networks

For a 200x200 image, we have $4x10^4$ neurons each one with $4x10^4$ inputs, that is $16x10^8$ parameters, only for one layer!!!

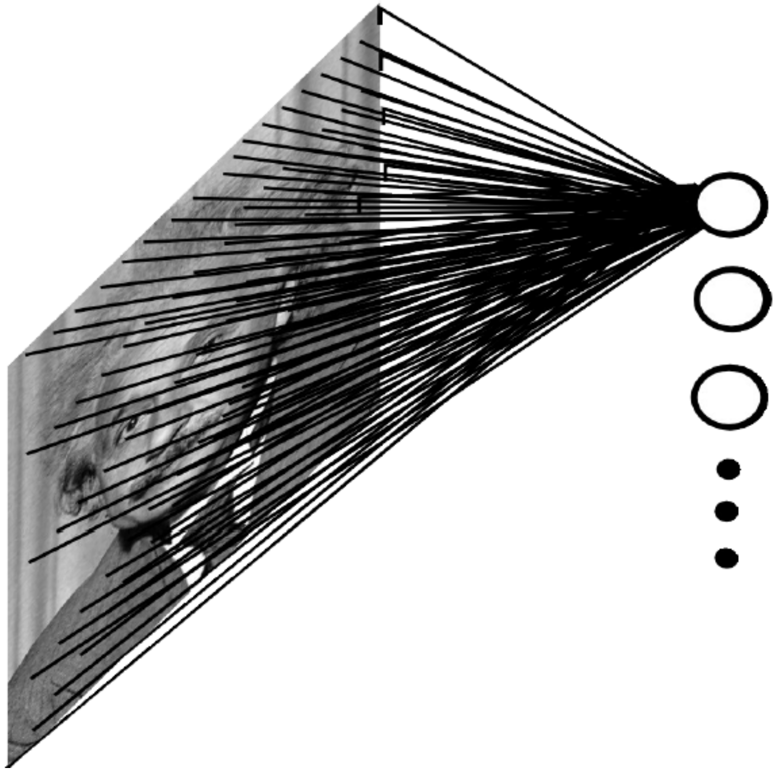


Figure Credit: Ranzatto

From Neurons to Convolutional Neural Networks

For a 200x200 image, we have 4×10^4 neurons each one with 10x10 "**local connections**" (also called receptive field) inputs, that is 4×10^6

What else can we do to reduce the number of parameters?

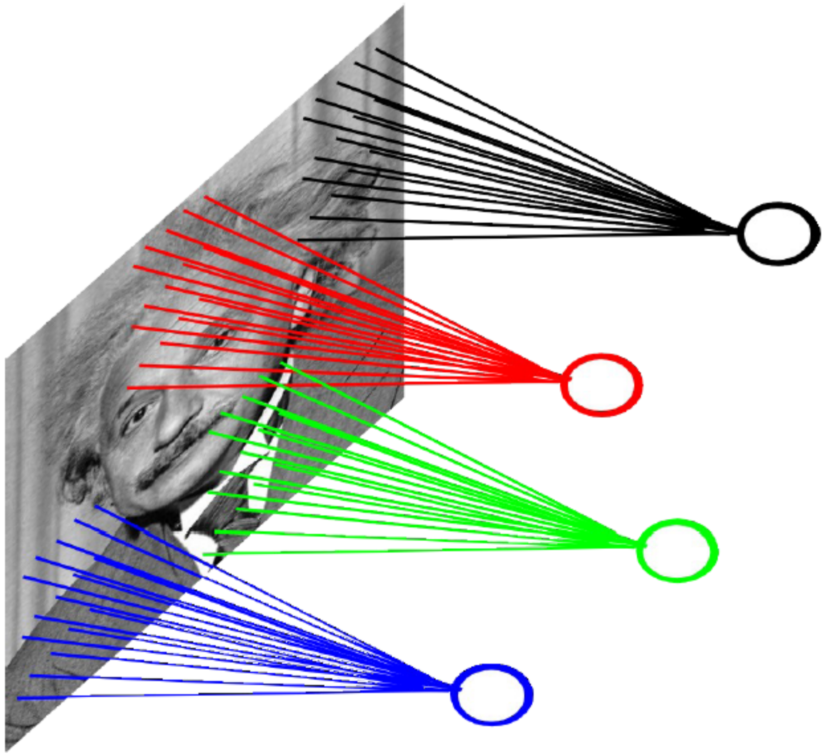


Figure Credit: Ranzatto

From Neurons to Convolutional Neural Networks

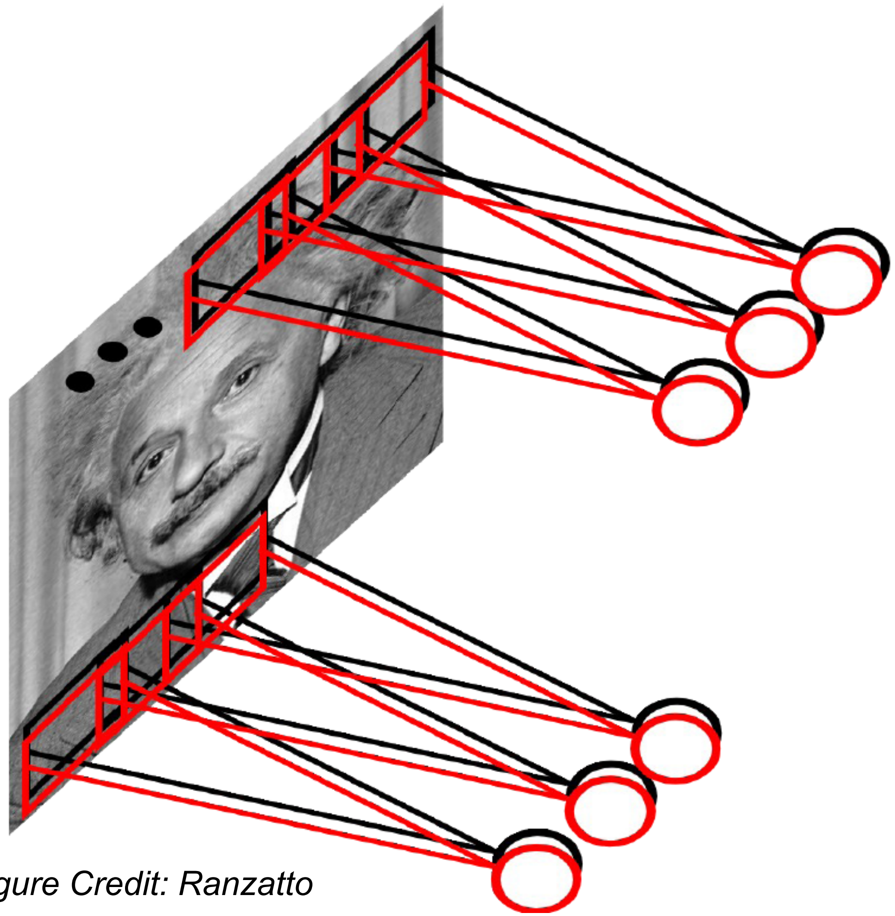


Figure Credit: Ranzatto

Translation invariance: we can use same parameters to capture a specific "feature" in any area of the image. We can try different sets of parameters to capture different features.

These operations are equivalent to perform **convolutions** with different filters.

Ex: With100 different filters (or feature extractors) of size 10x10, the number of parameters is 10^4

That is why they are called **Convolutional Neural Networks, (ConvNets or CNNs)**

From Neurons to Convolutional Neural Networks

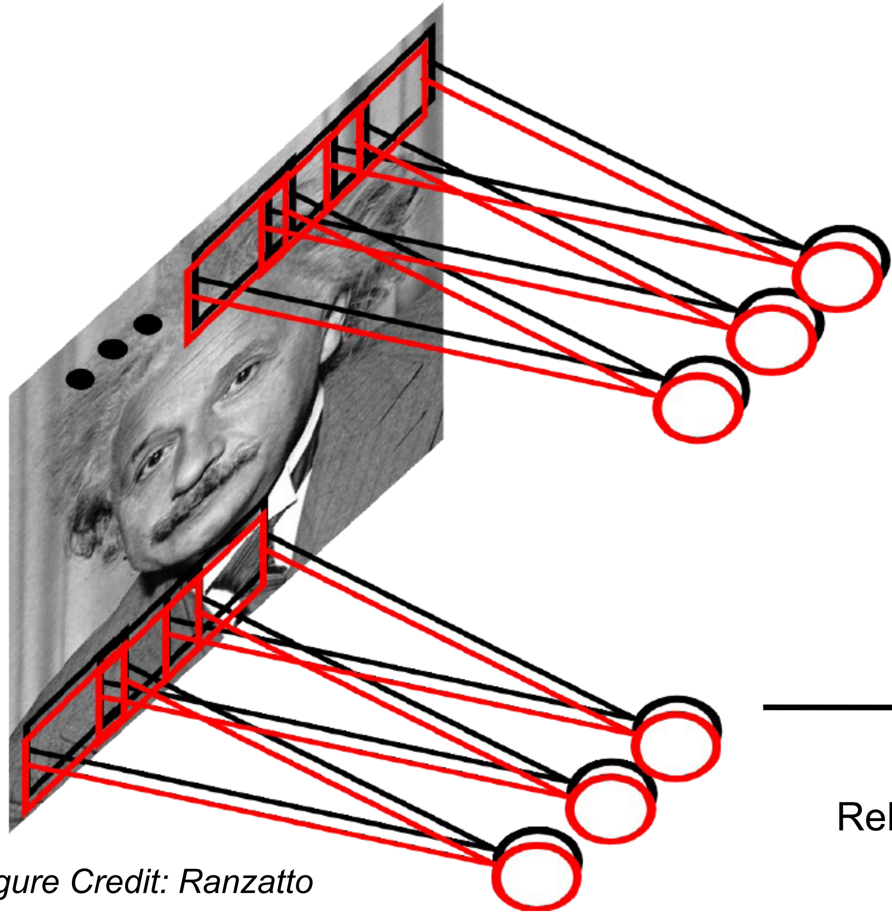
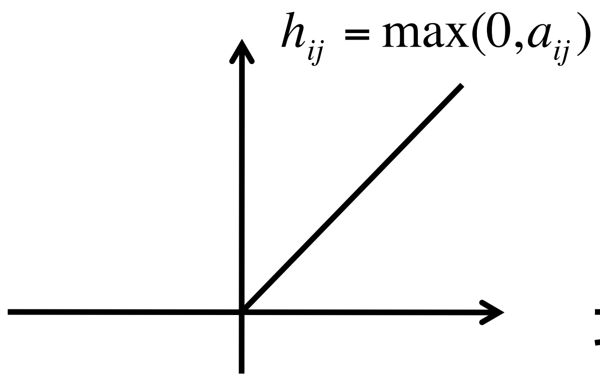


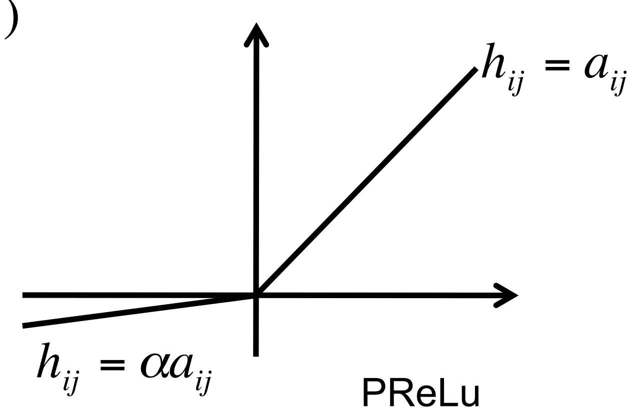
Figure Credit: Ranzatto

…and don't forget the activation function!

$$a_{ij} = \sum_{k,l} w_{kl} x_{k-i,l-j} + b$$



ReLu



PReLu

From Neurons to Convolutional Neural Networks

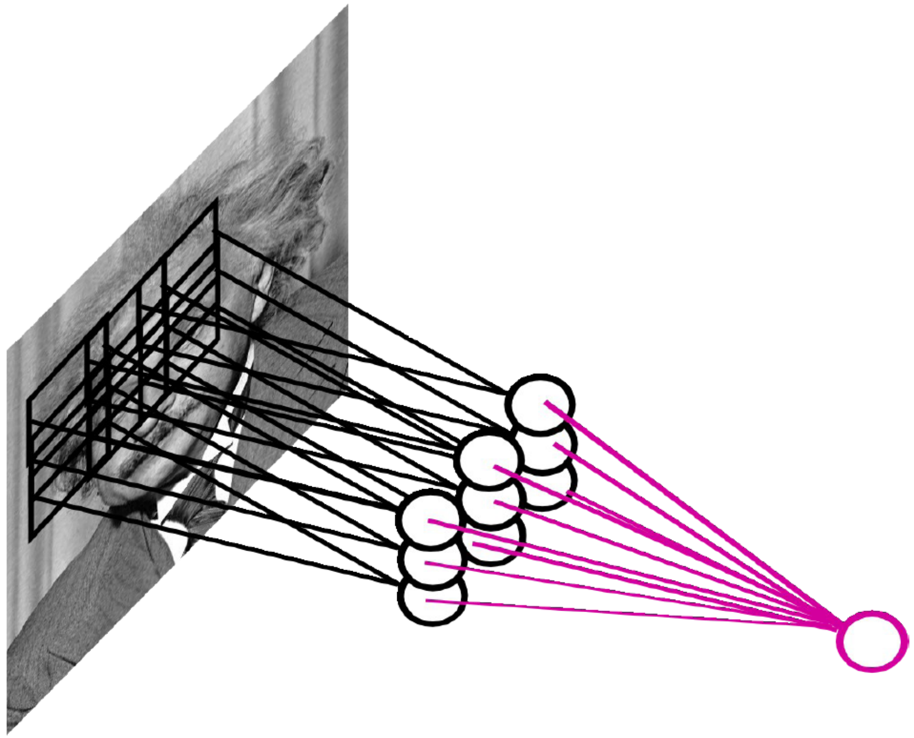


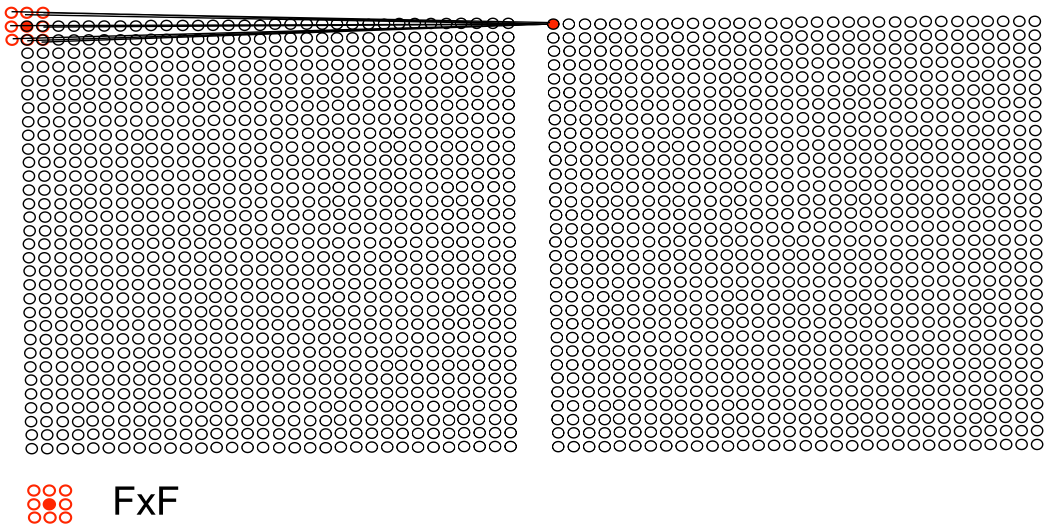
Figure Credit: Ranzatto

Most ConvNets use **Pooling** (or subsampling) to reduce dimensionality and provide invariance to small local changes.

Pooling options:

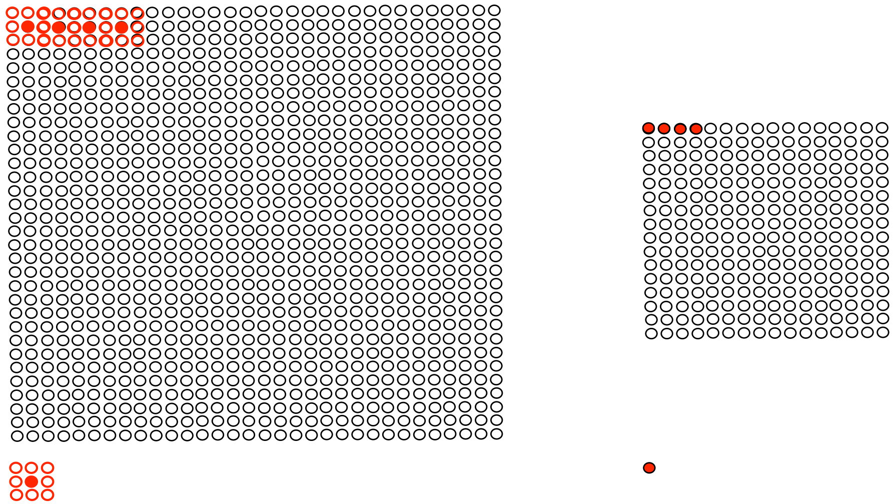
- **Max**
- Average
- Stochastic pooling

From Neurons to Convolutional Neural Networks



Padding (P): When doing the convolution in the borders, you may add values to compute the convolution.
When the values are zero, that is quite common, the technique is called zero-padding.
When padding is not used the output size is reduced.

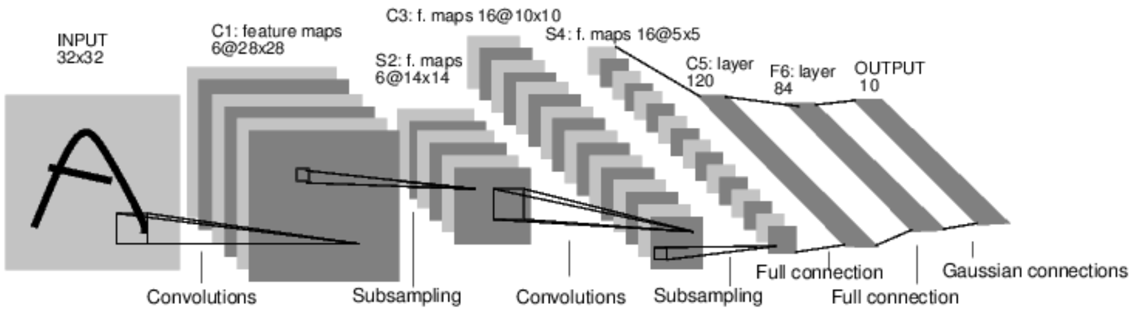
From Neurons to Convolutional Neural Networks



Stride (S): When doing the convolution or another operation, like pooling, we may decide to slide not pixel by pixel but every 2 or more pixels. The number of pixels that we skip is the value of the stride.
It might be used to reduce the dimensionality of the output

From Neurons to Convolutional Neural Networks

Example: Most convnets contain several convolutional layers, interspersed with pooling layers, and followed by a small number of fully connected layers
A layer is characterized by its width, height and depth (that is, the number of filters used to generate the feature maps)
An architecture is characterized by the number of layers



LeNet-5 From Lecun ´98