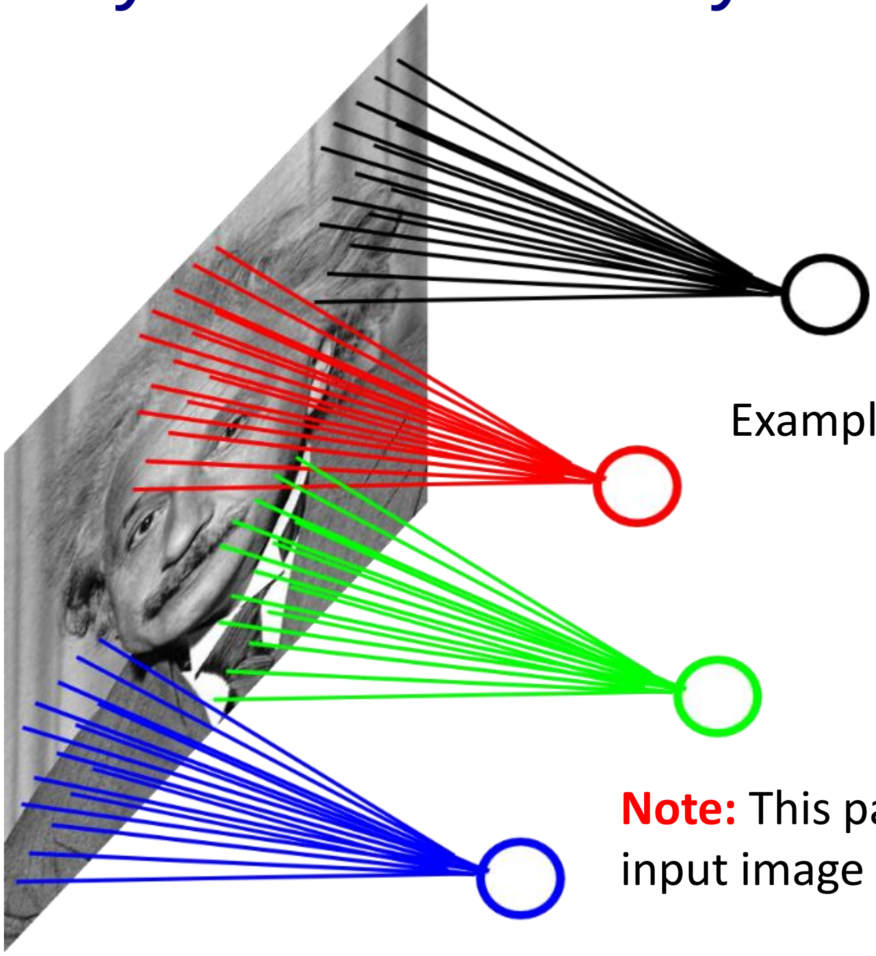
Locally Connected Layers

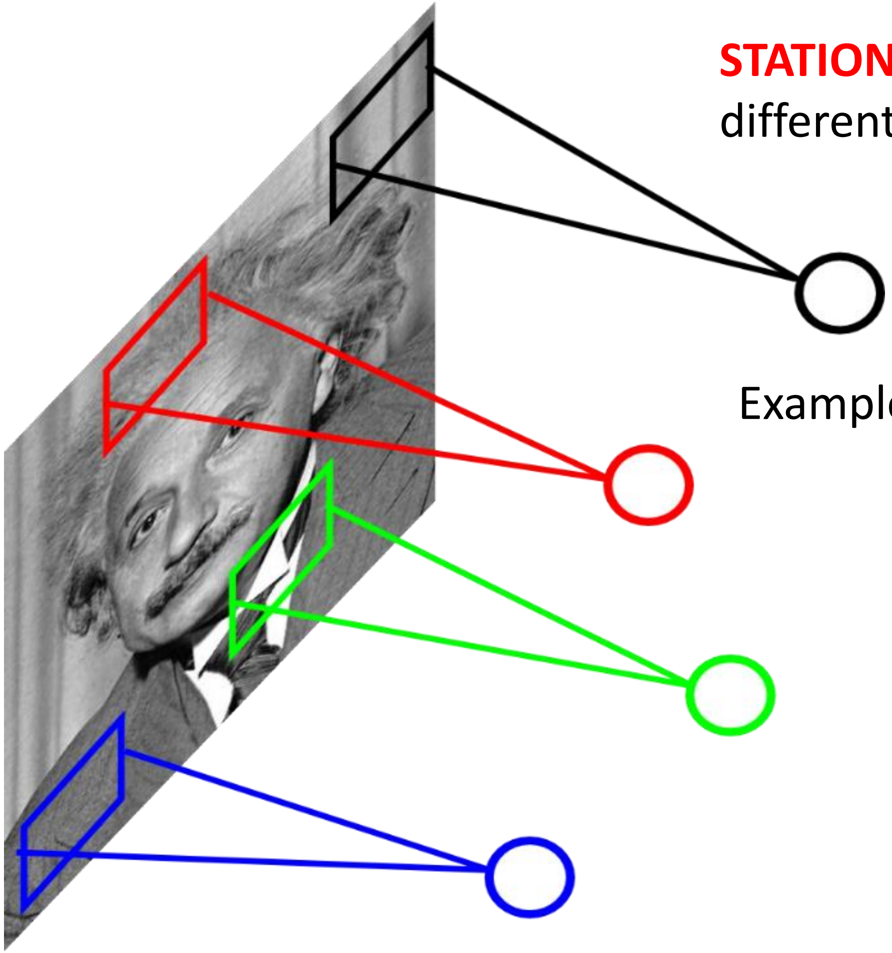


Example: 200x200 image
40K hidden units
Filter size: 10x10
→ 4M parameters

Note: This parameterization is good when input image is registered (e.g., face recognition).

Ranzato

Locally Connected Layers

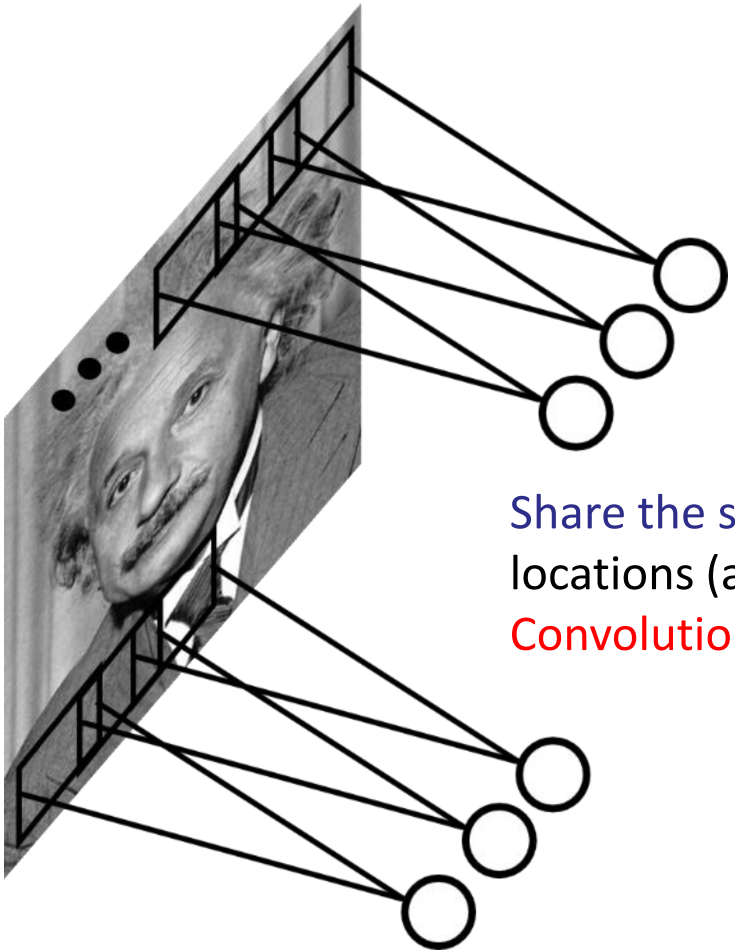


STATIONARITY? Statistics is similar at different locations

Example: 200x200 image
40K hidden units
Filter size: 10x10
→ 4M parameters

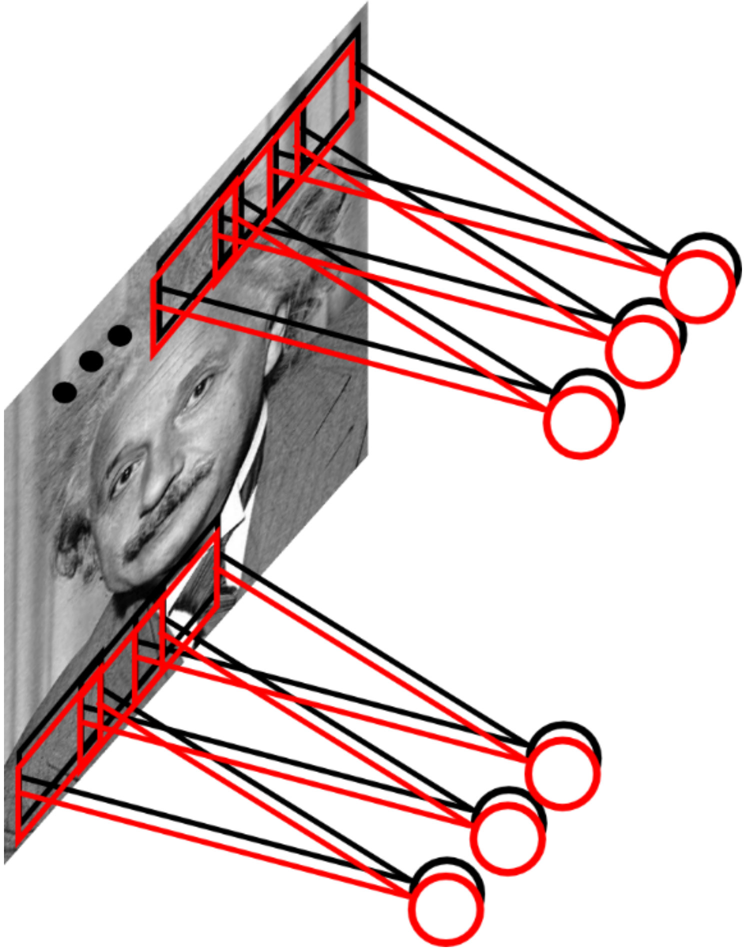
Ranzato

Convolutional Layer



Share the same parameters across different locations (assuming input is stationary):
Convolutions with learned kernels

Convolutional Layer

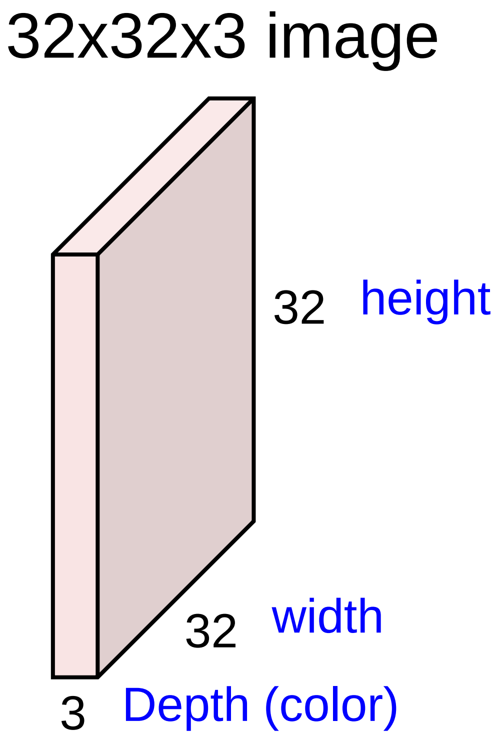


Learn multiple filters.

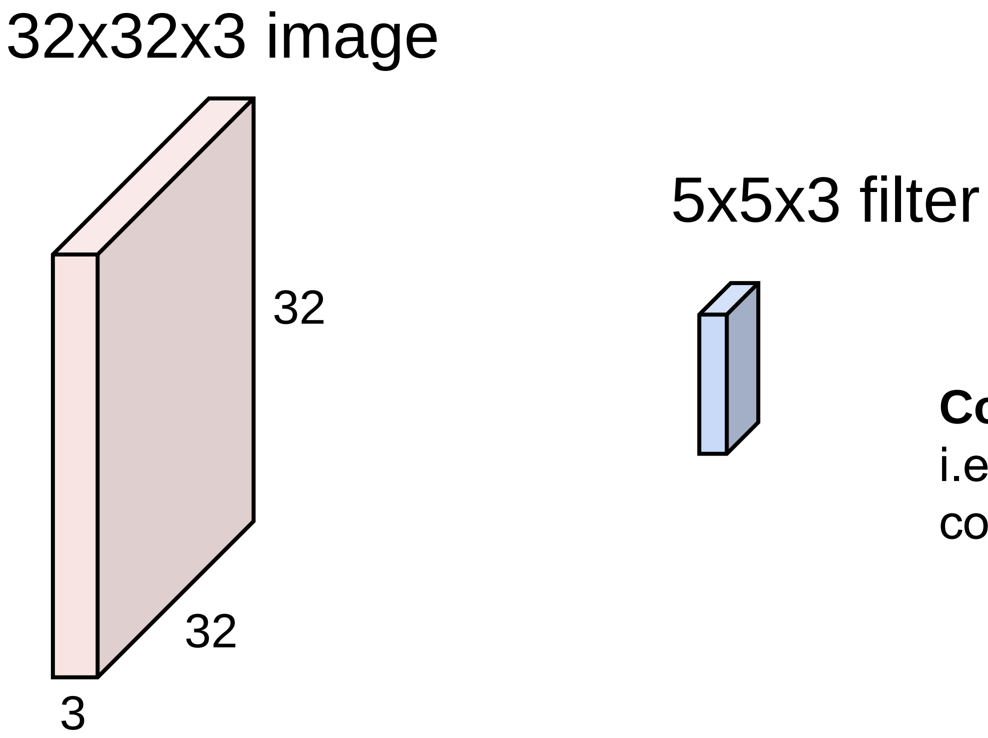
E.g.: 200x200 image
100 Filters
Filter size: 10x10
10K parameters

Ranzato

Convolution Layer



Convolution Layer

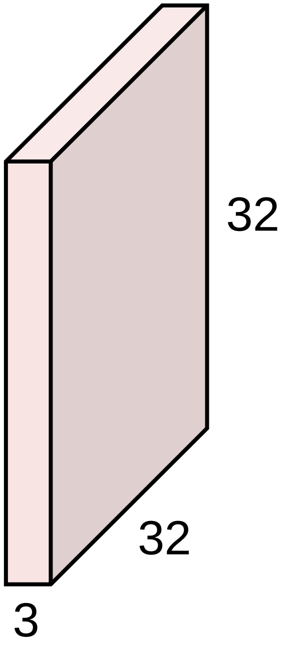


Convolve the filter with the image
i.e. "slide over the image spatially,
computing dot products"

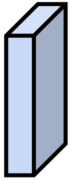
Convolution Layer

Filters always extend to the full depth of the input volume

32x32x3 image

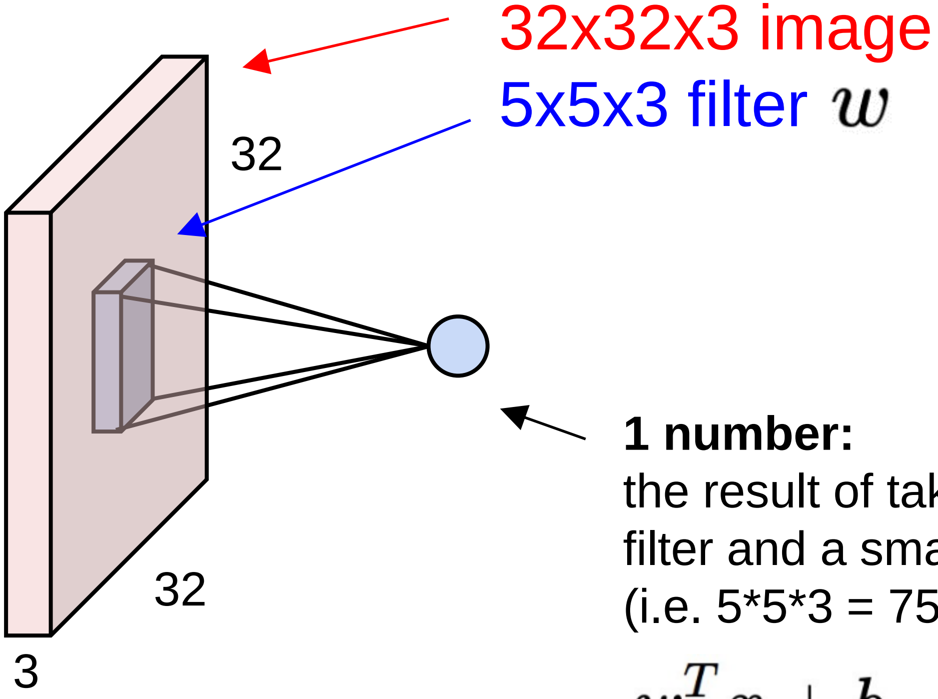


5x5x3 filter



Convolve the filter with the image i.e. “slide over the image spatially, computing dot products”

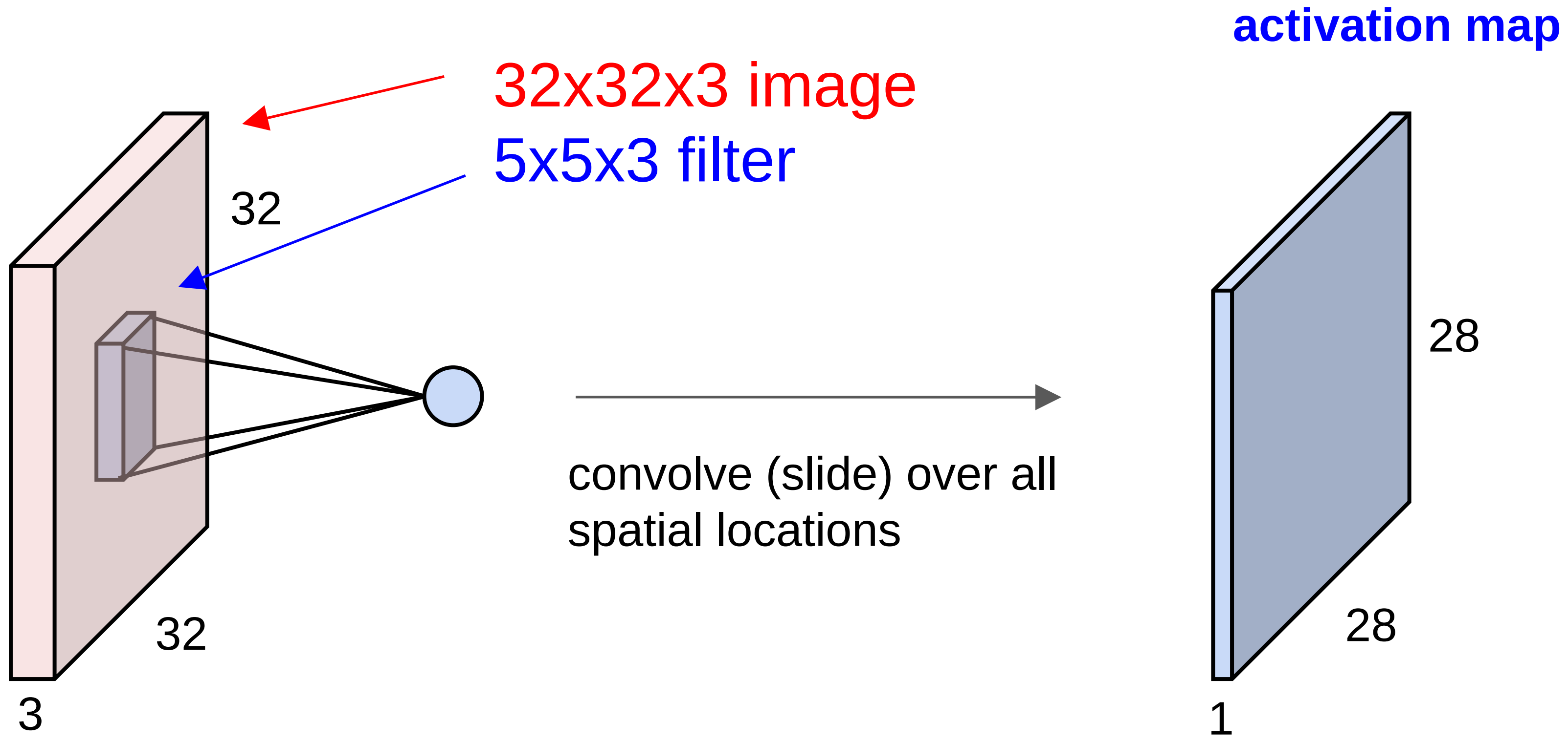
Convolution Layer



1 number:
the result of taking a dot product between the filter and a small 5x5x3 chunk of the image
(i.e. 5*5*3 = 75-dimensional dot product + bias)

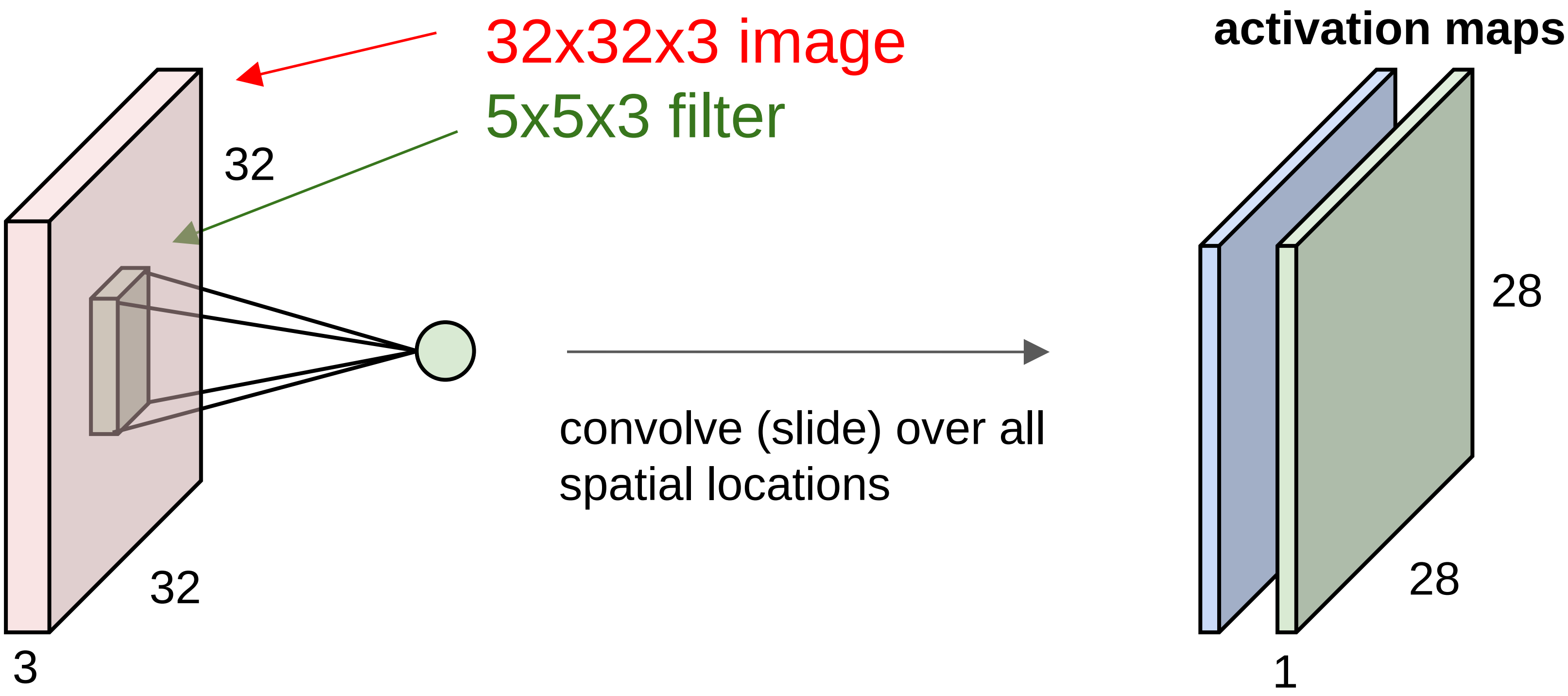
$$w^T x + b$$

Convolution Layer

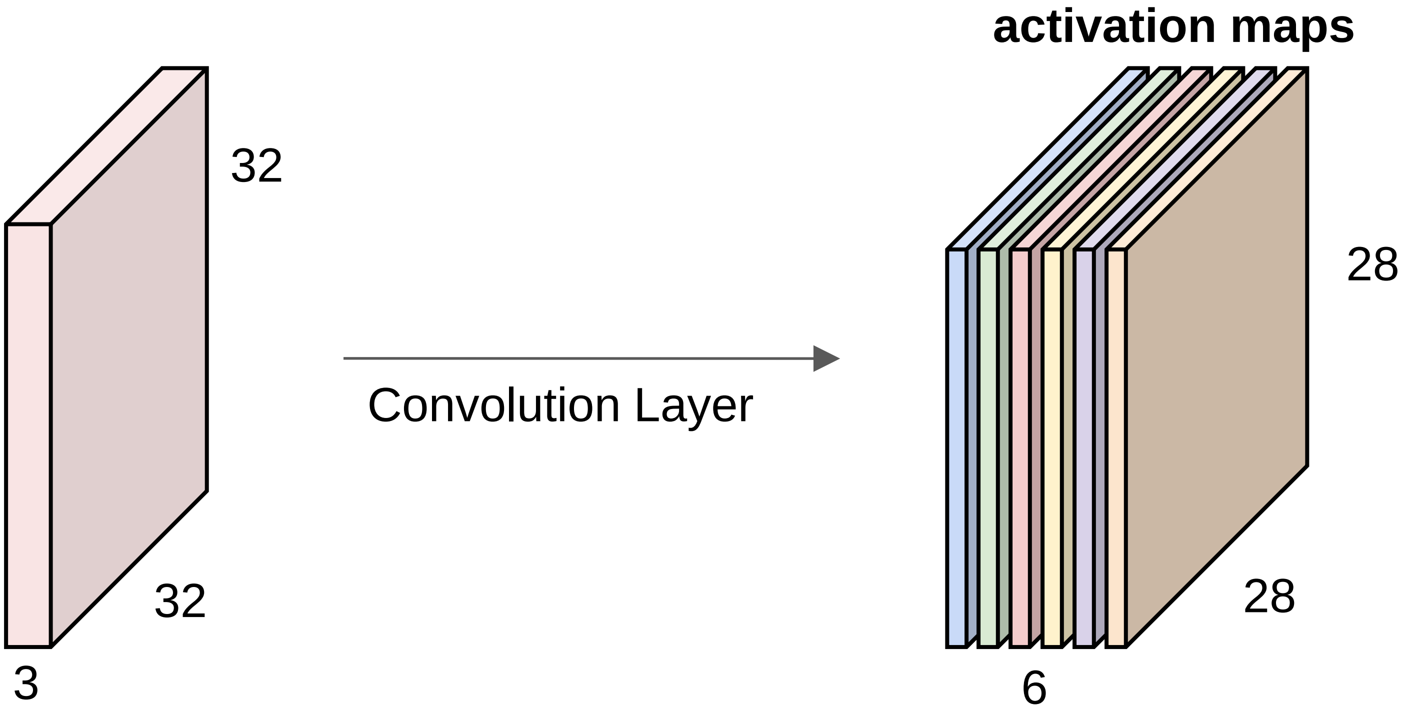


Convolution Layer

consider a second, green filter



For example, if we had 6 5x5 filters, we'll get 6 separate activation maps:



We stack these up to get a "new image" of size 28x28x6!

“Tensors” again

Because there are multiple channels in each data block, convolutional filters are normally specified by 4D arrays. Common dimensions are:

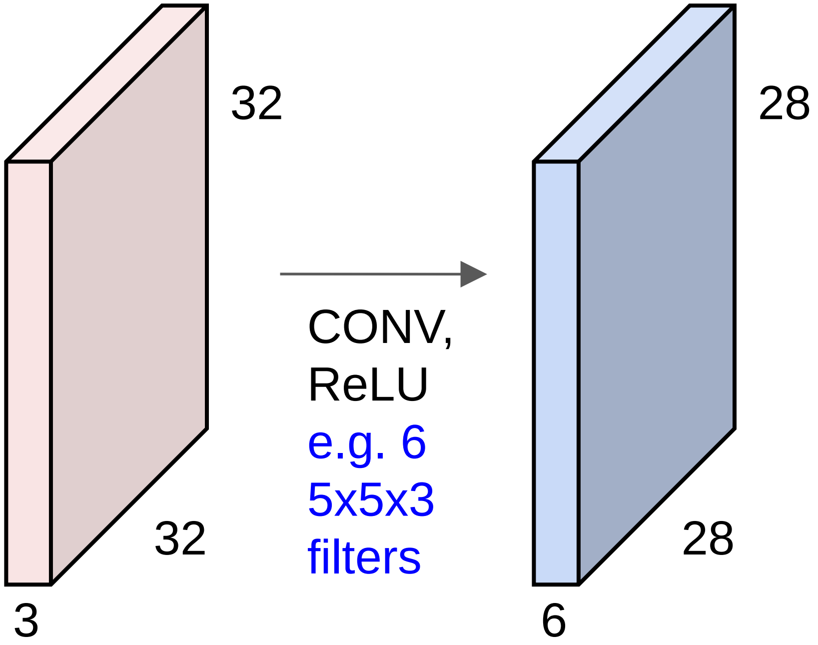
= number of output channels

= number of input channels

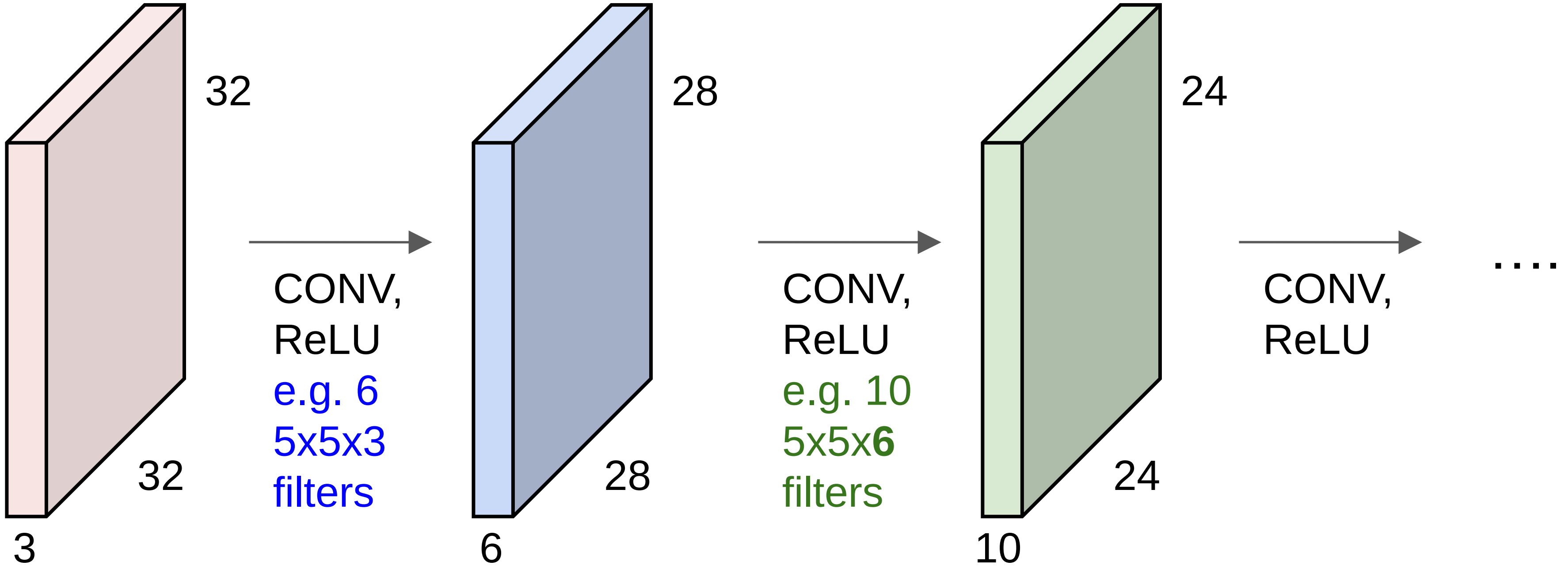
= filter height

= filter width

Preview: ConvNet is a sequence of Convolution Layers, interspersed with non-linear activation functions



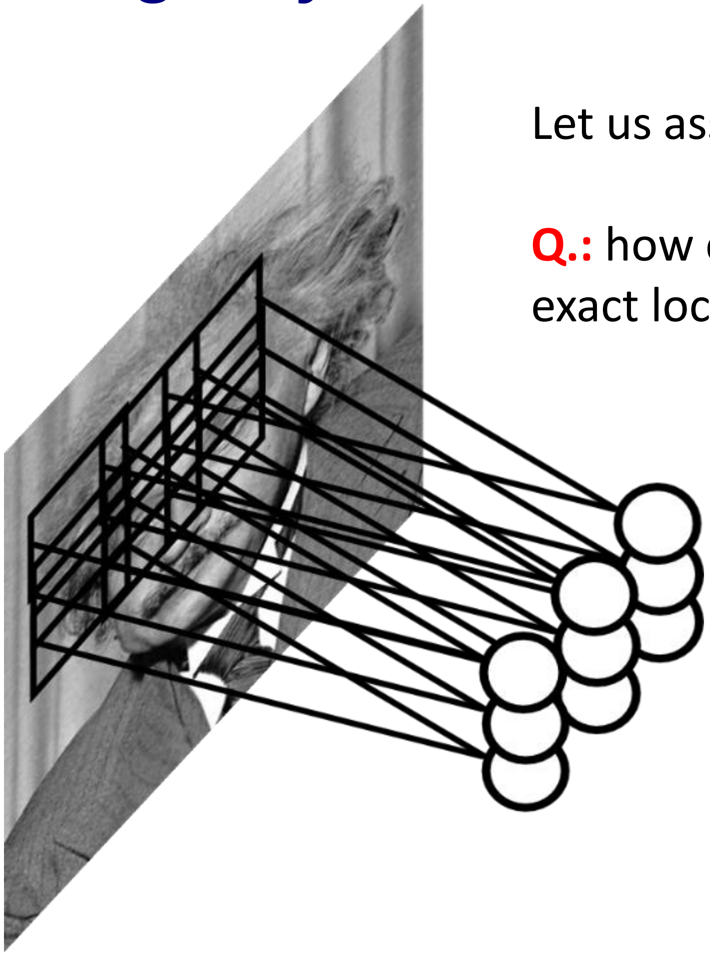
Preview: ConvNet is a sequence of Convolution Layers, interspersed with non-linear activation functions



Summary. To summarize, the Conv Layer:

- Accepts a volume of size $W_1 \times H_1 \times D_1$
- Requires four hyperparameters:
  - Number of filters K,
  - their spatial extent F,
  - the stride S,
  - the amount of zero padding P.
- Produces a volume of size $W_2 \times H_2 \times D_2$ where:
  - $W_2 = (W_1 - F + 2P)/S + 1$
  - $H_2 = (H_1 - F + 2P)/S + 1$ (i.e. width and height are computed equally by symmetry)
  - $D_2 = K$
- With parameter sharing, it introduces $F \cdot F \cdot D_1$ weights per filter, for a total of $(F \cdot F \cdot D_1) \cdot K$ weights and K biases.
- In the output volume, the d-th depth slice (of size $W_2 \times H_2$) is the result of performing a valid convolution of the d-th filter over the input volume with a stride of S, and then offset by d-th bias.

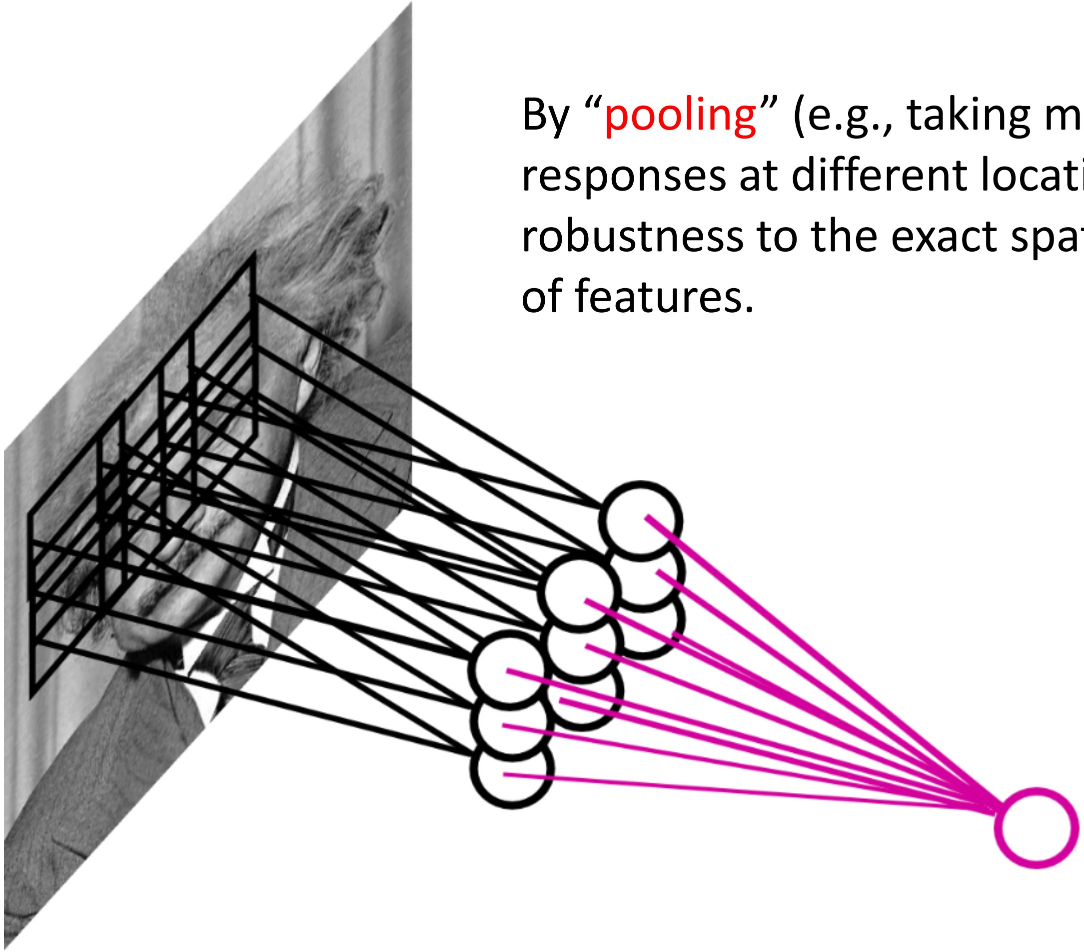
Pooling Layer



Let us assume filter is an “eye” detector.

Q.: how can we make the detection robust to the exact location of the eye?

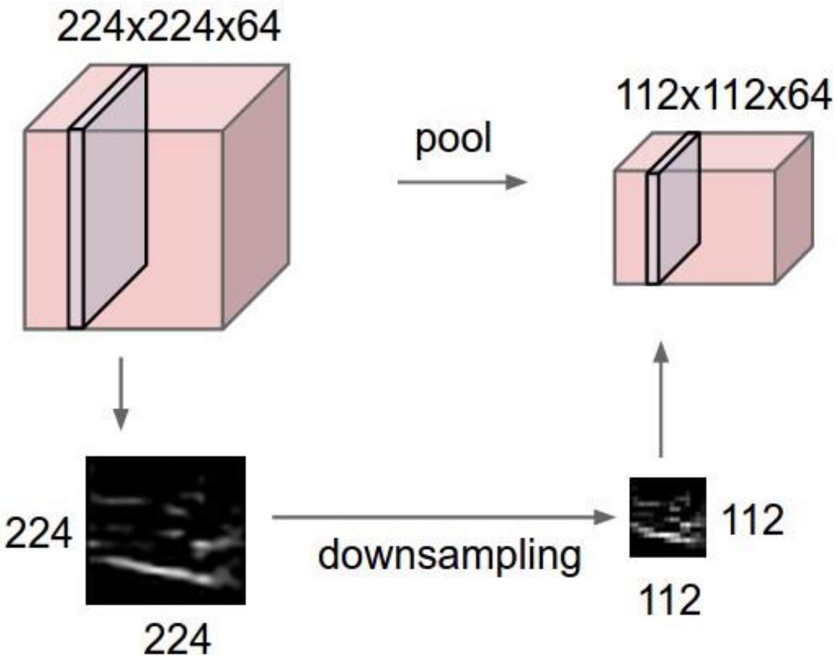
Pooling Layer



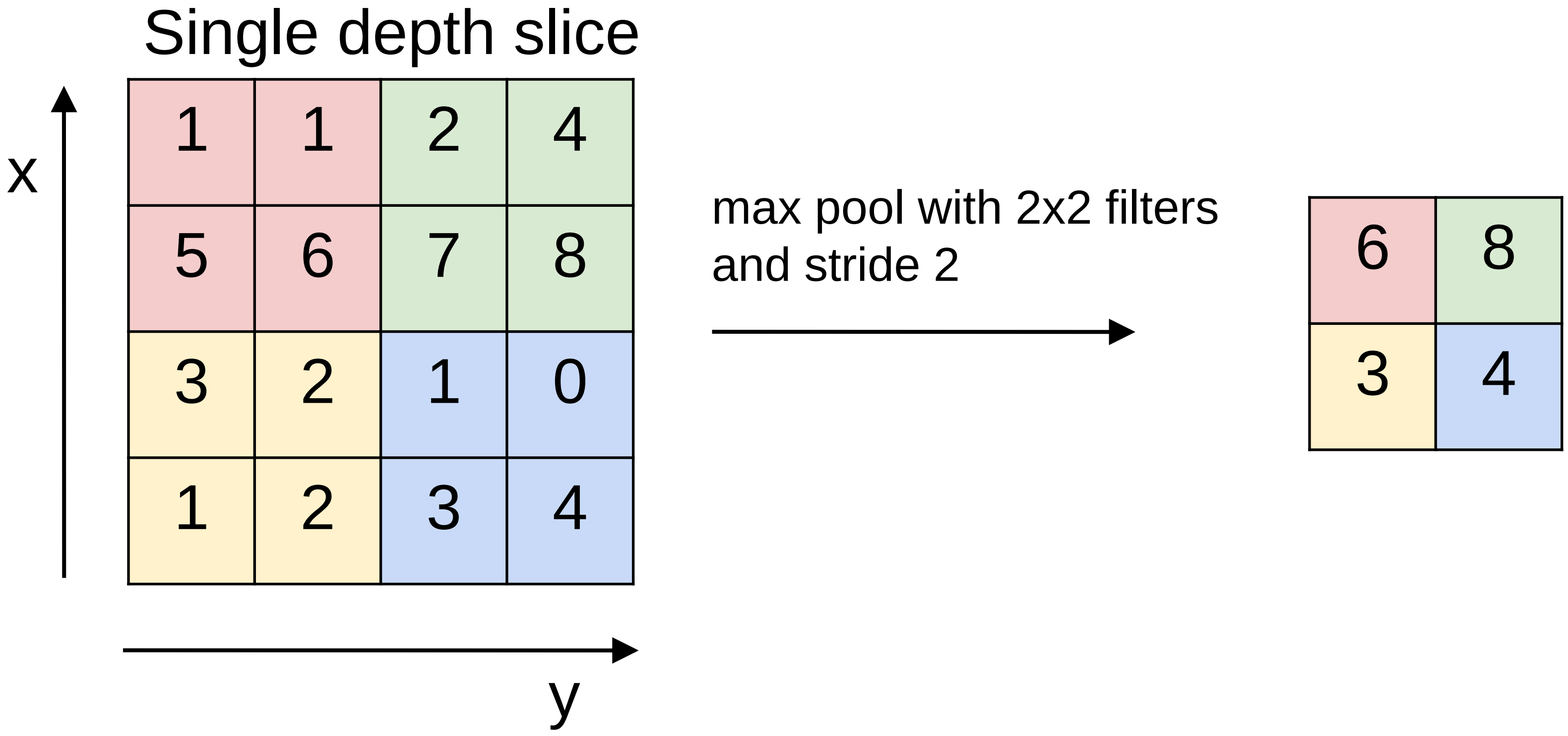
By “pooling” (e.g., taking max) filter responses at different locations we gain robustness to the exact spatial location of features.

Pooling layer

- makes the representations smaller and more manageable
- operates over each activation map independently:

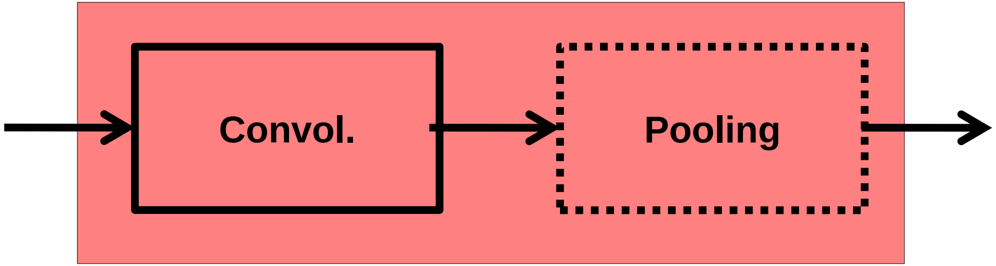


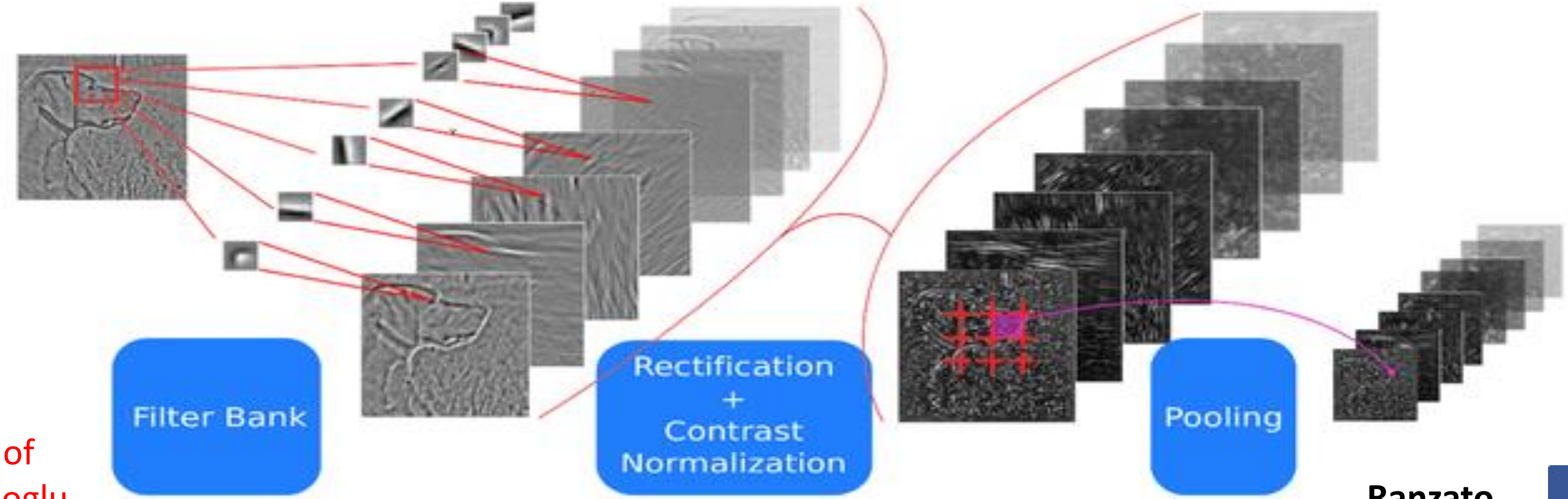
MAX POOLING


Single depth slice
x
1 1 2 4
5 6 7 8
3 2 1 0
1 2 3 4
y
max pool with 2x2 filters
and stride 2
6 8
3 4

ConvNets: Typical Stage

One stage (zoom)





courtesy of
K. Kavukcuoglu

Ranzato