COMP/EECE 7/8740 Neural Networks

Topics:

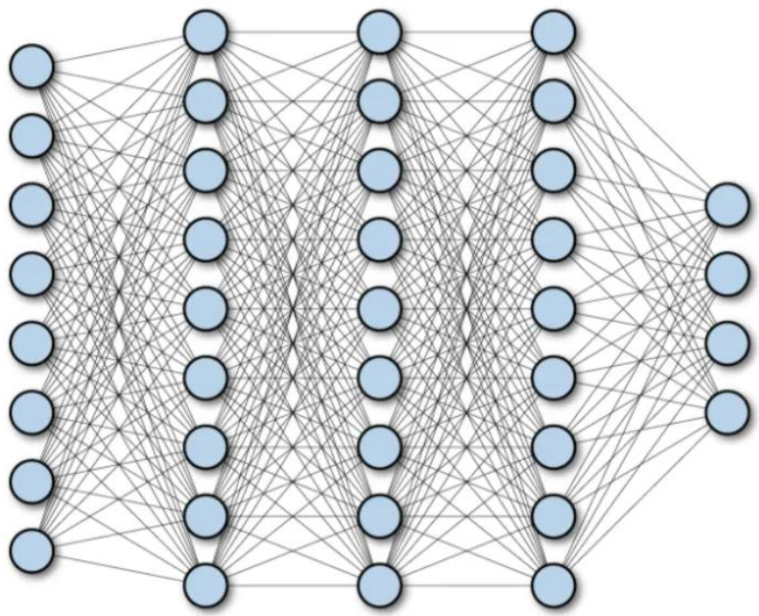
- Fully connected network (FCN)
- FC layer to convolution layer
- Convolutional operations
  - Input and outputs dimensions
  - Stride size and padding
- Convolutional Neural Networks
  - Convolution layers
  - Activation layers
  - Pooling layers

Md Zahangir Alom
Department of Computer Science
University of Memphis, TN

Fully Connected Neural Network(FCNN)

FCNN consists of a series of **fully connected layers** that connect every neuron in one layer to every neuron in the other layer.

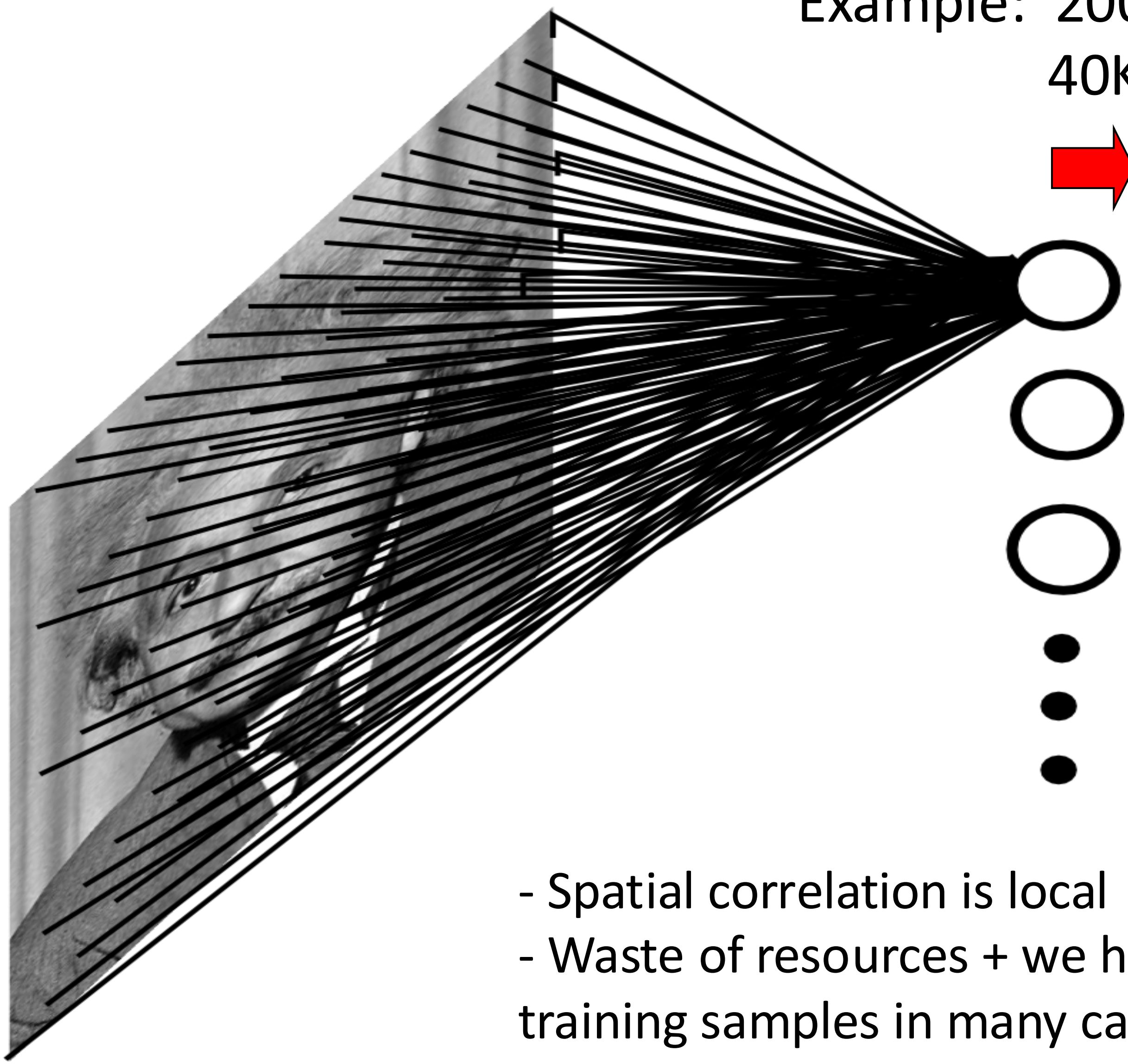
- Advantages :
  - FCNNs are “structure agnostic” (i.e. no special assumptions needed about the inputs).
- Disadvantages:
  - FCNNs show weaker performance than special-purpose networks (structure of a problem space).



Architecture : 8 (Inputs) →9→9→9→4 (Outputs)

Image credit : https://www.oreilly.com/library/view/tensorflow-for-deep/

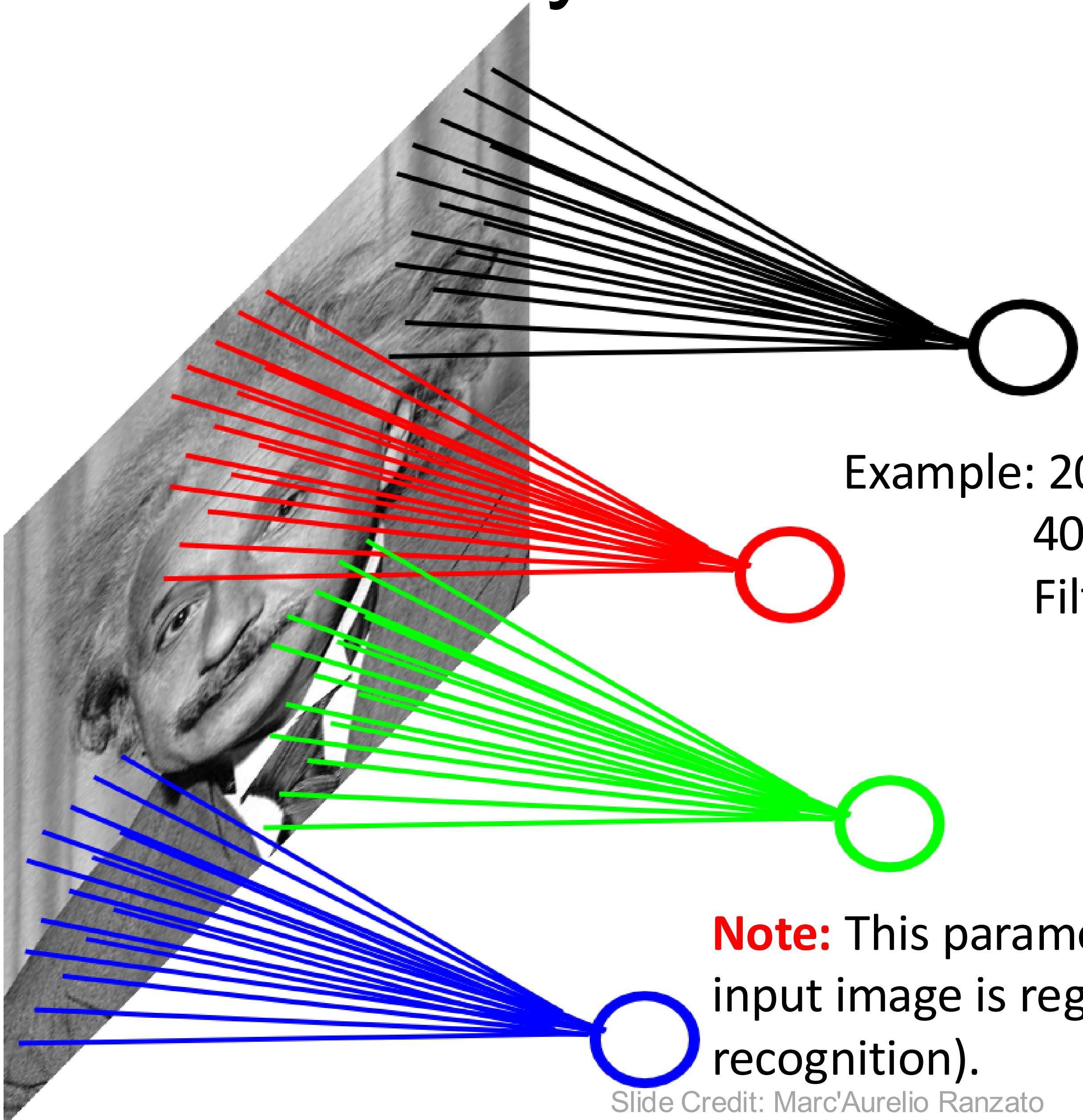
Fully Connected Layer



Example: 200x200 image
40K hidden units
→ **~2B parameters**!!!

- Spatial correlation is local
- Waste of resources + we have not enough training samples in many cases

Locally Connected Layer

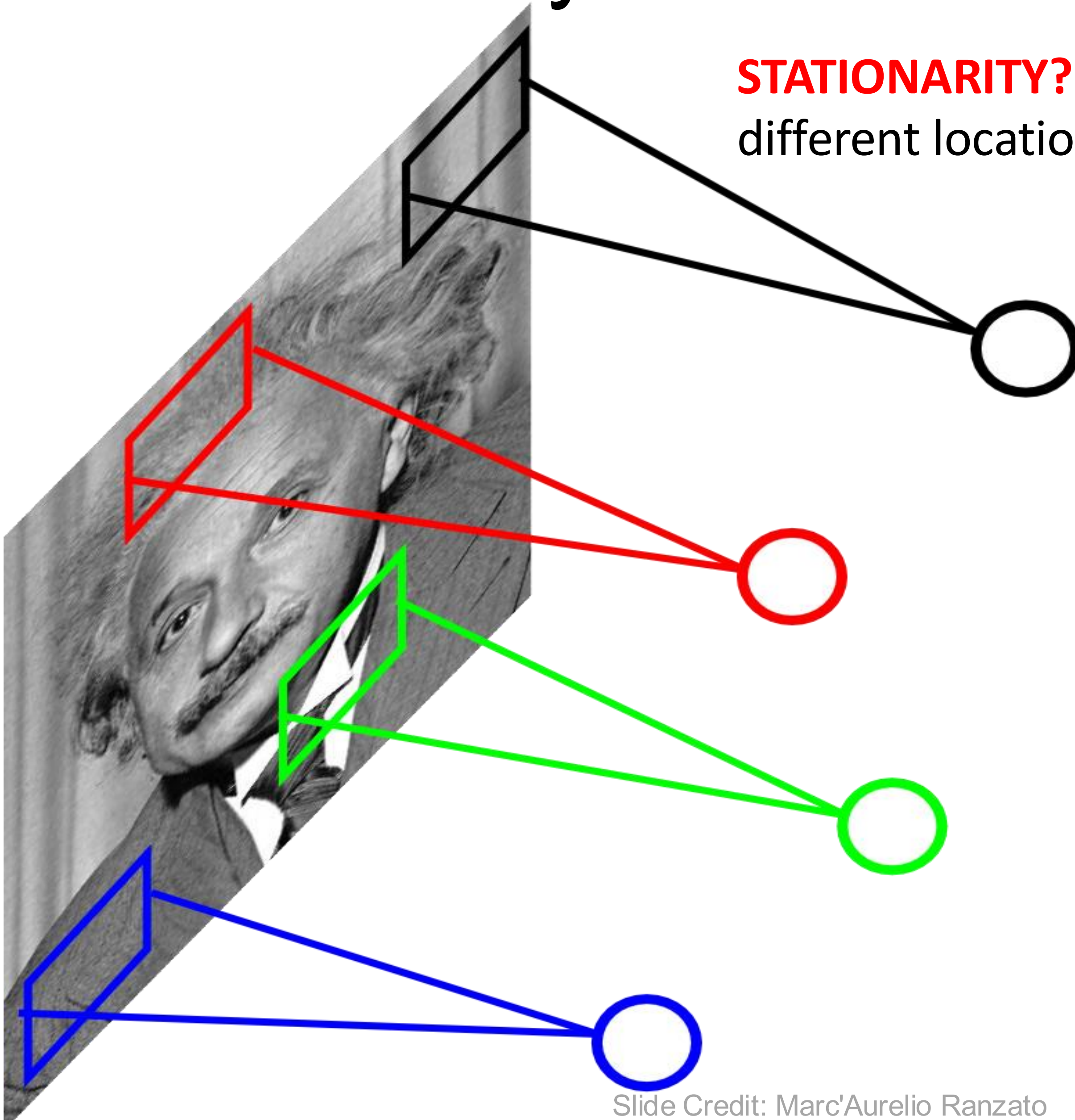


Example: 200x200 image
40K hidden units
Filter size: 10x10
4M parameters

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

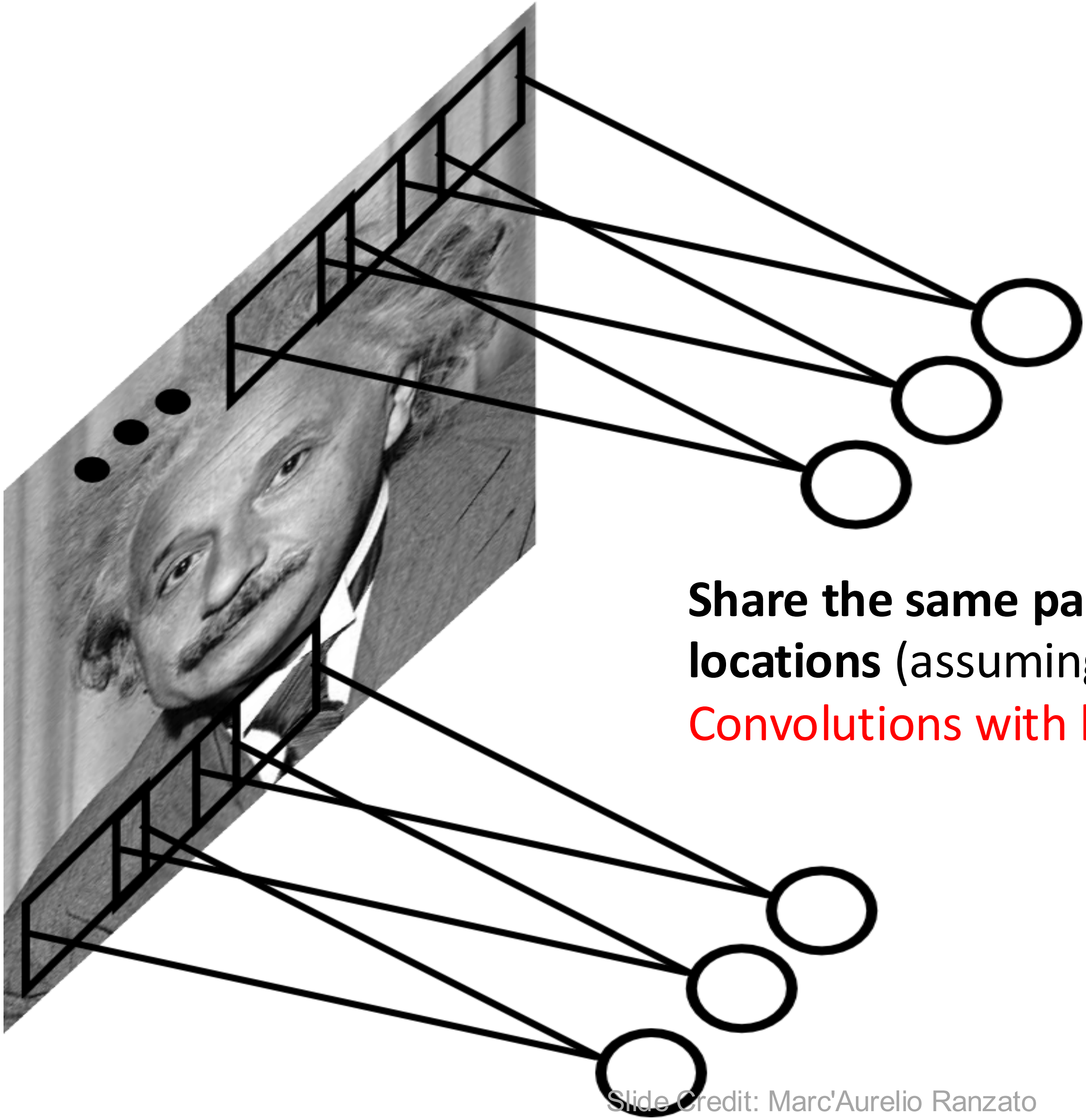
Locally Connected Layer

STATIONARITY? Statistics is similar at different locations



Slide Credit: Marc'Aurelio Ranzato

Convolutional Layer



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

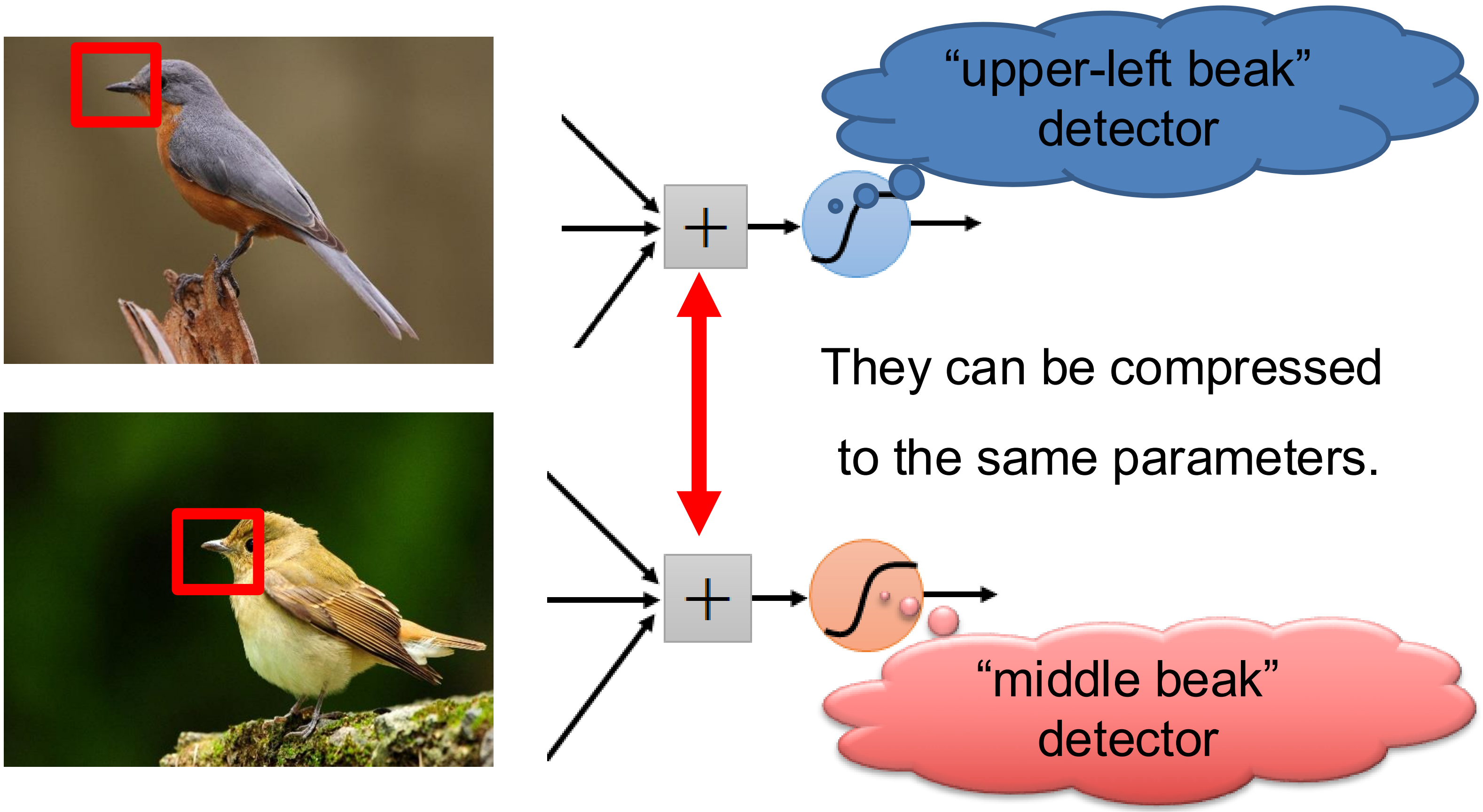
Consider learning an image:

- Some patterns are much smaller than the whole image

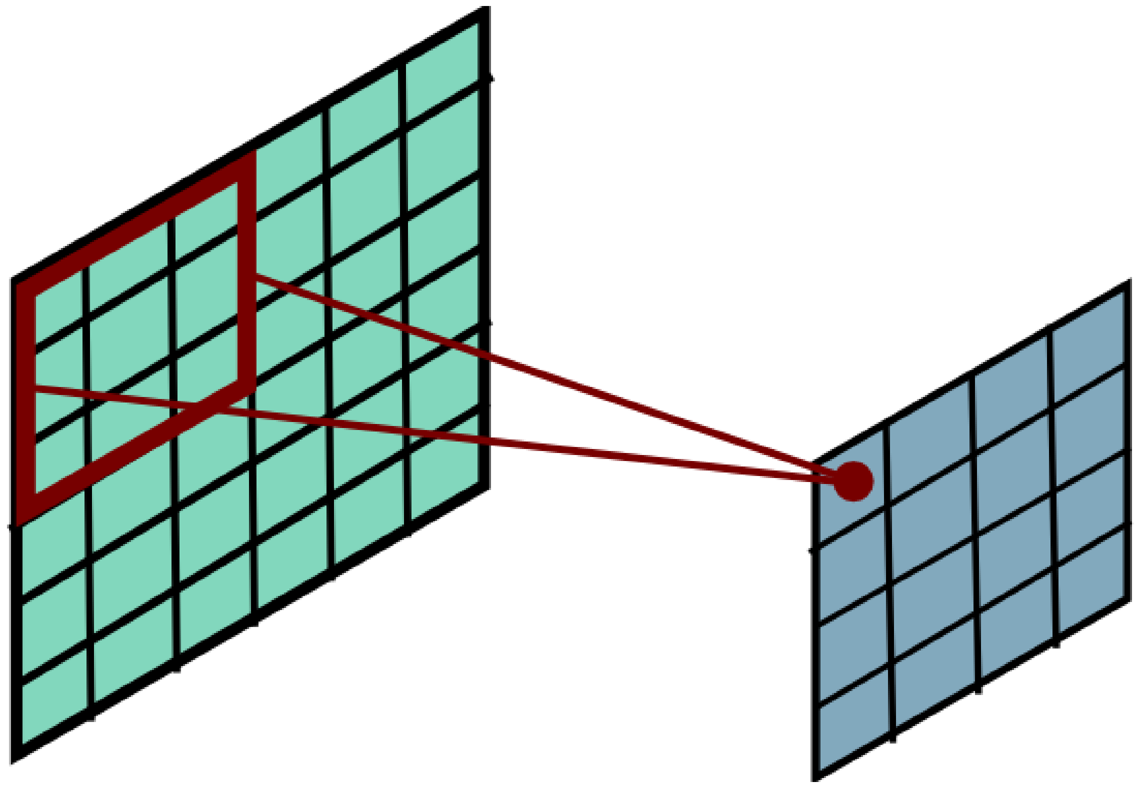
Can represent a small region with fewer parameters

Same pattern appears in different places: They can be compressed!

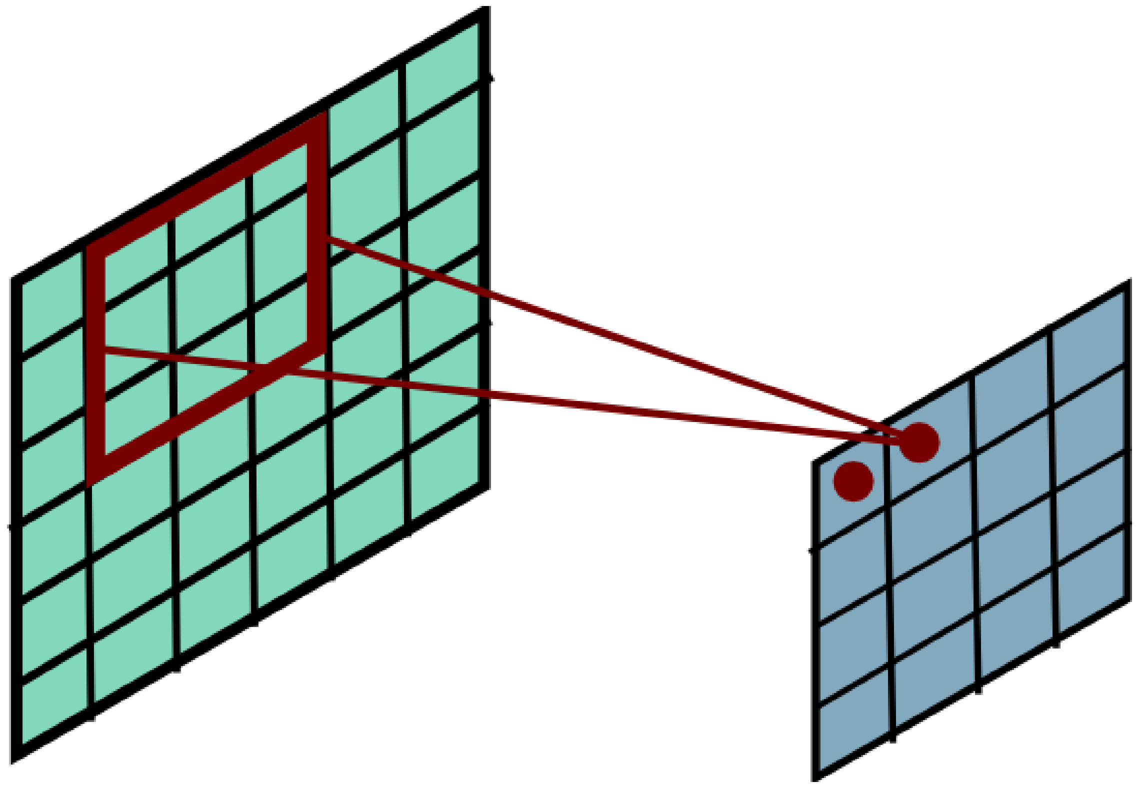
What about training a lot of such "small" detectors and each detector must "move around".



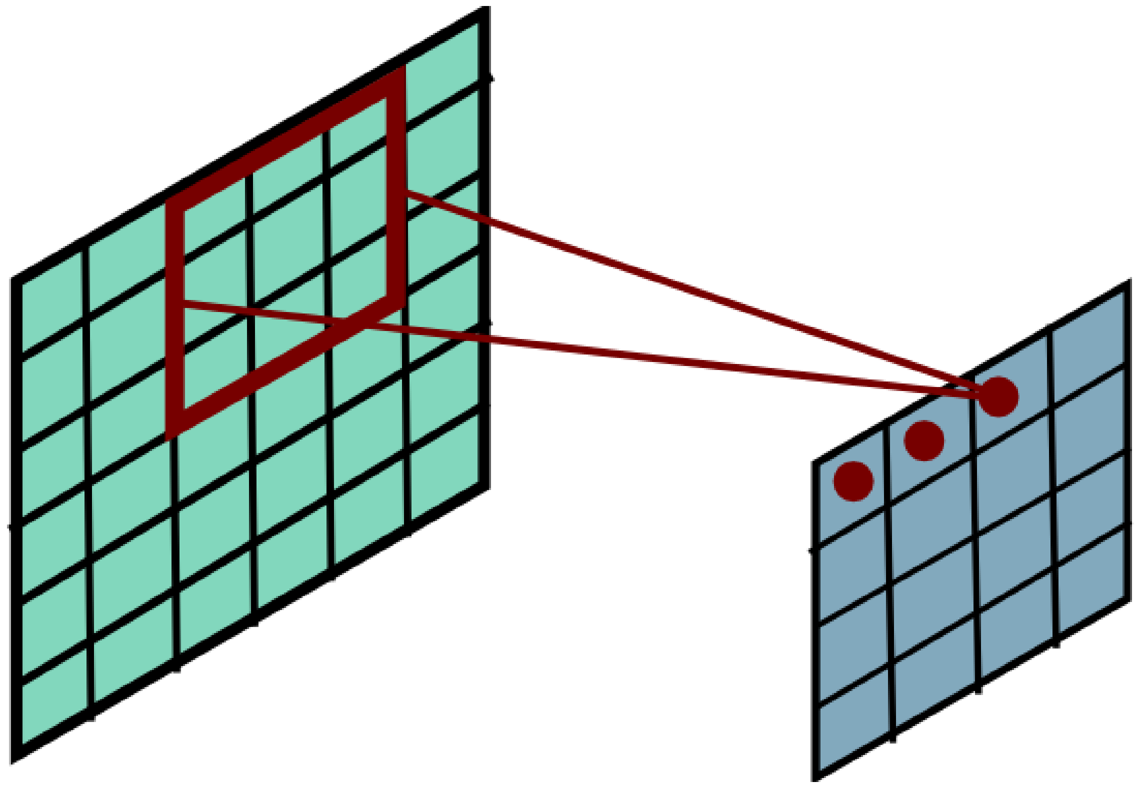
Convolution



Convolution

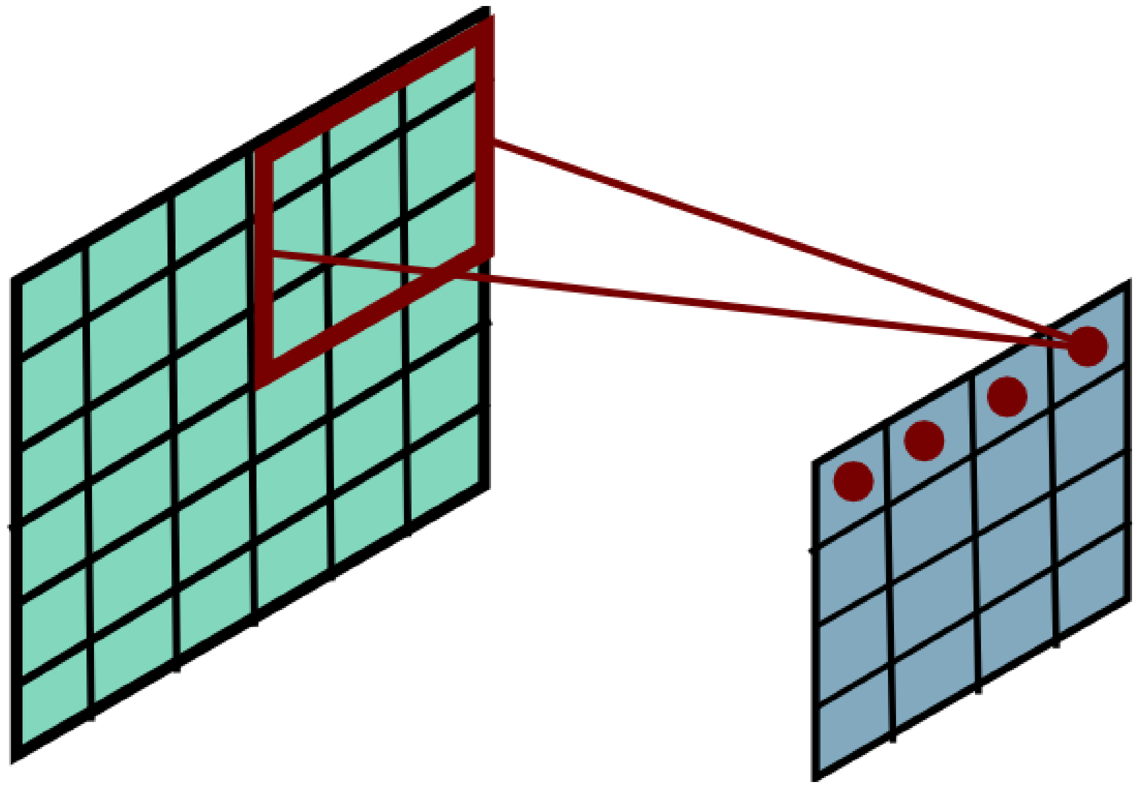


Convolution

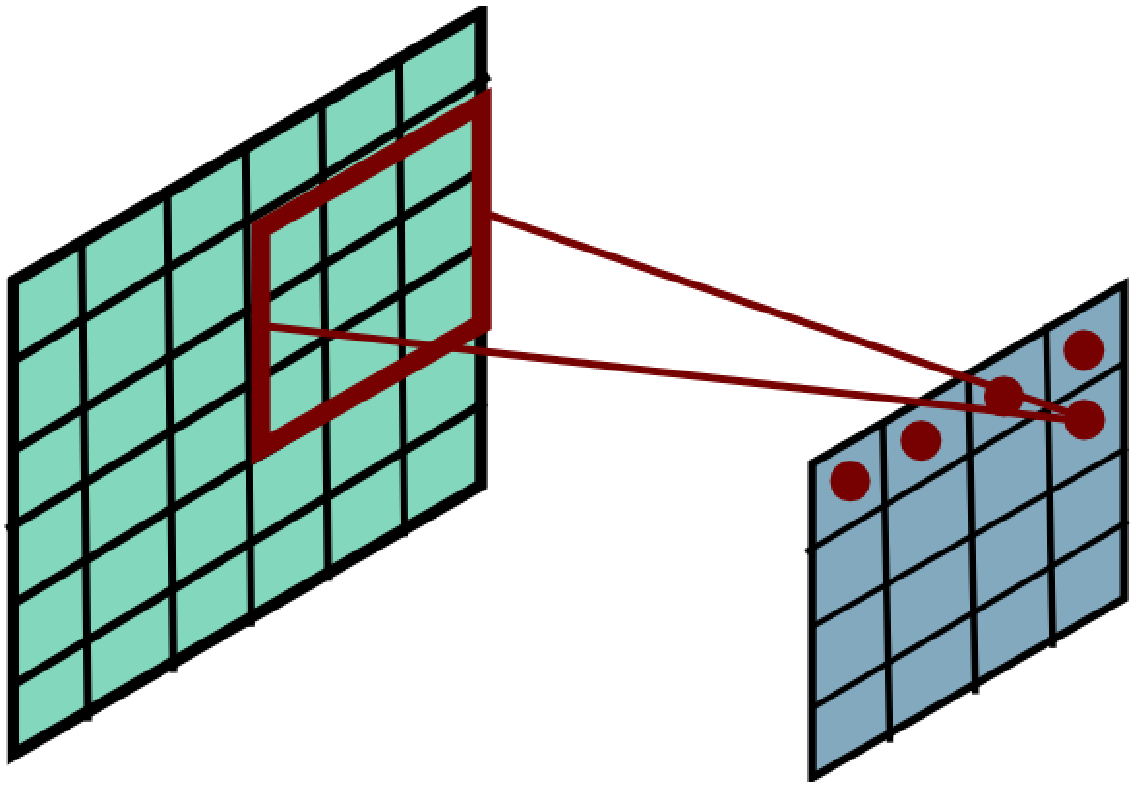


Slide Credit: Marc'Aurelio Ranzato

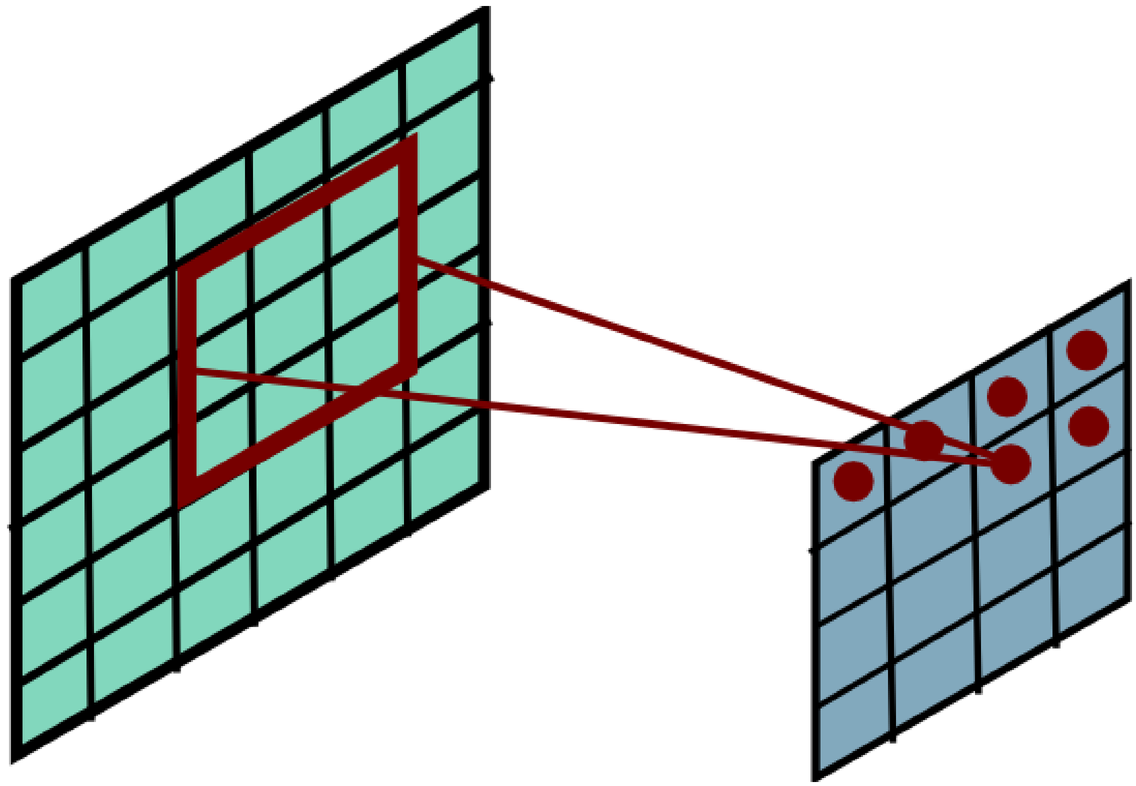
Convolution



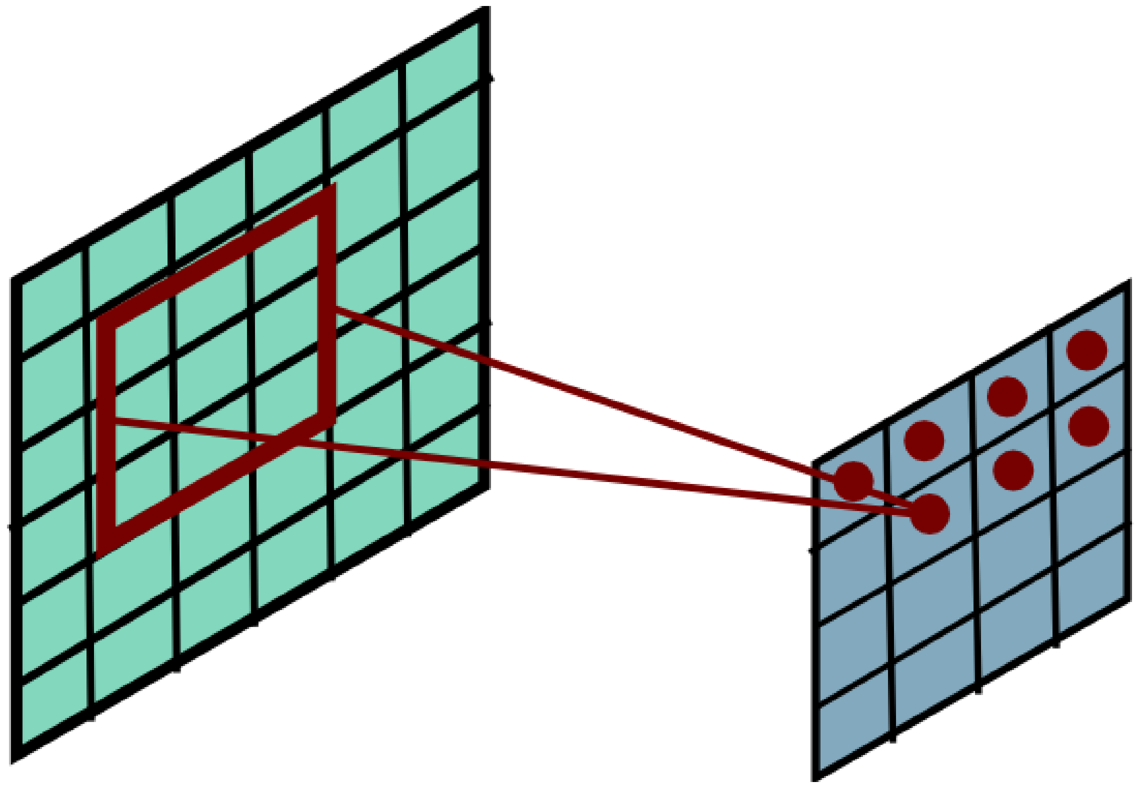
Convolution



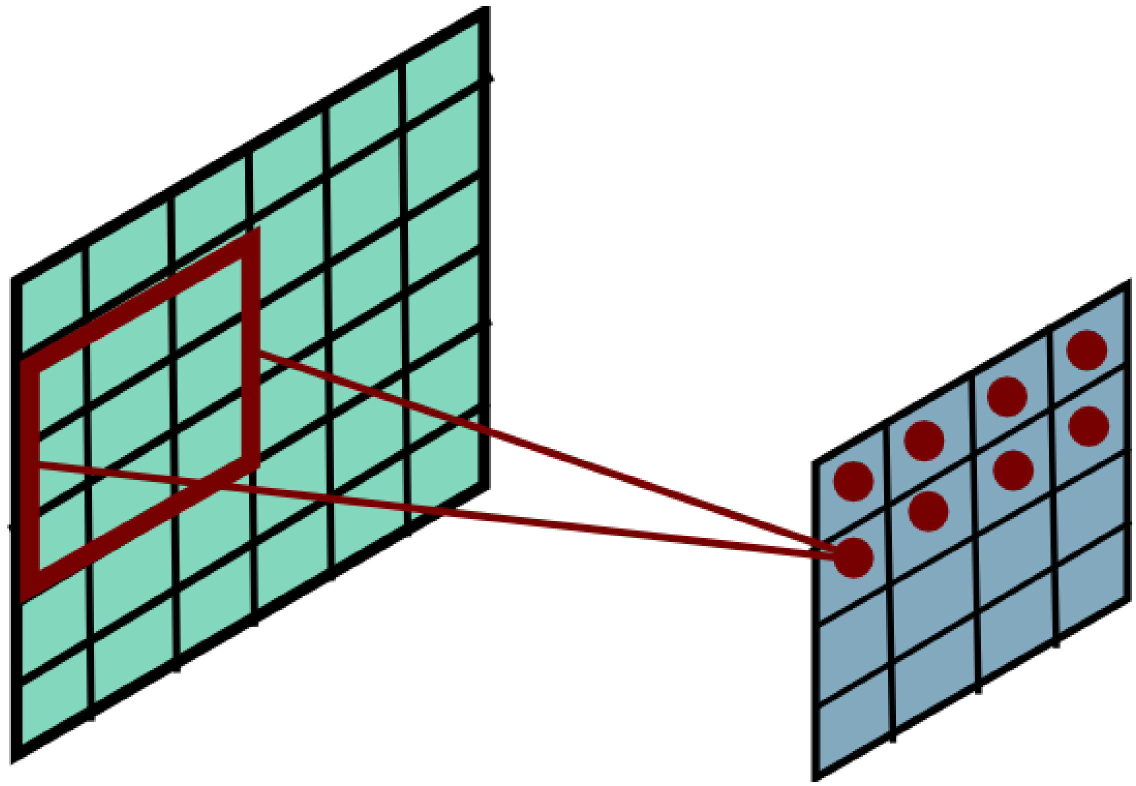
Convolution



Convolution

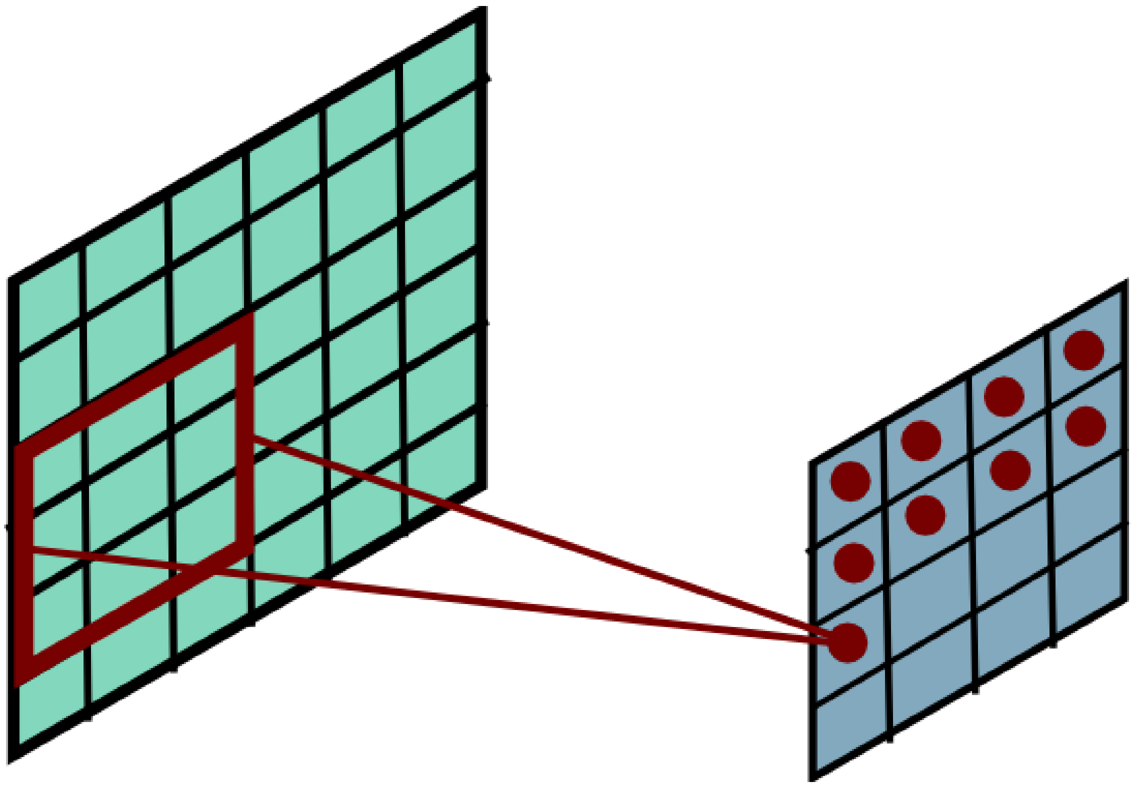


Convolution

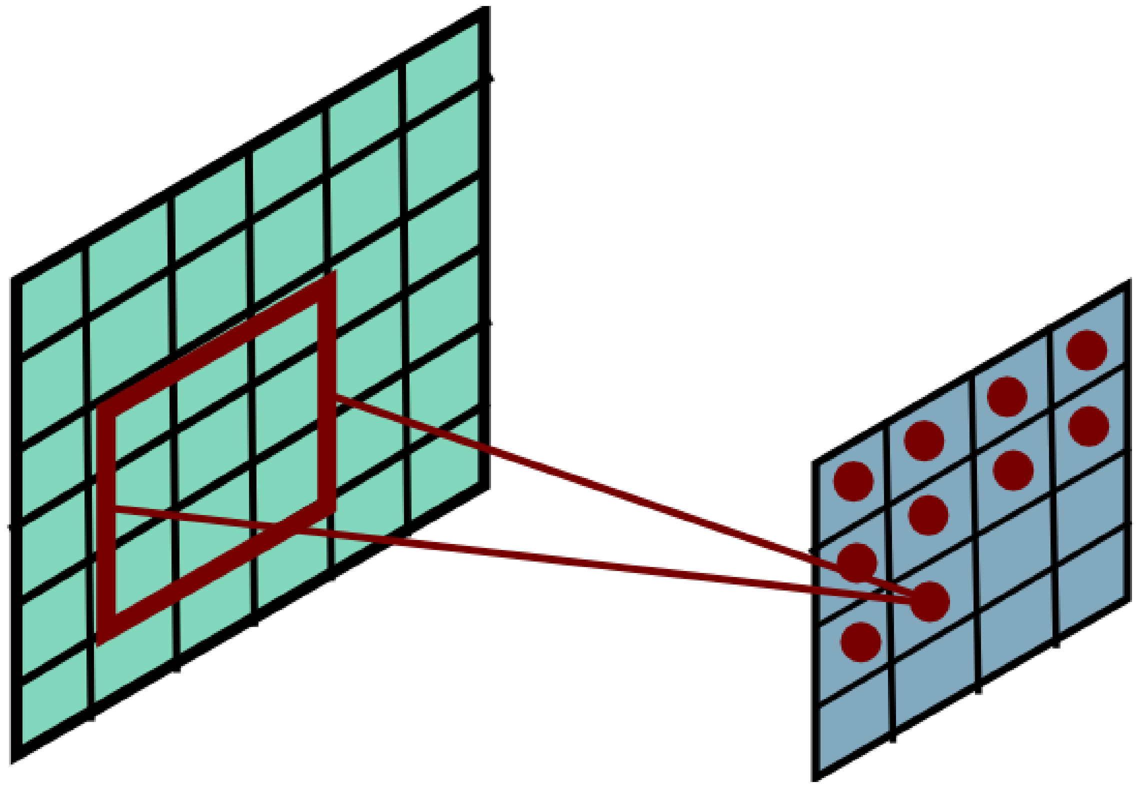


Slide Credit: Marc'Aurelio Ranzato

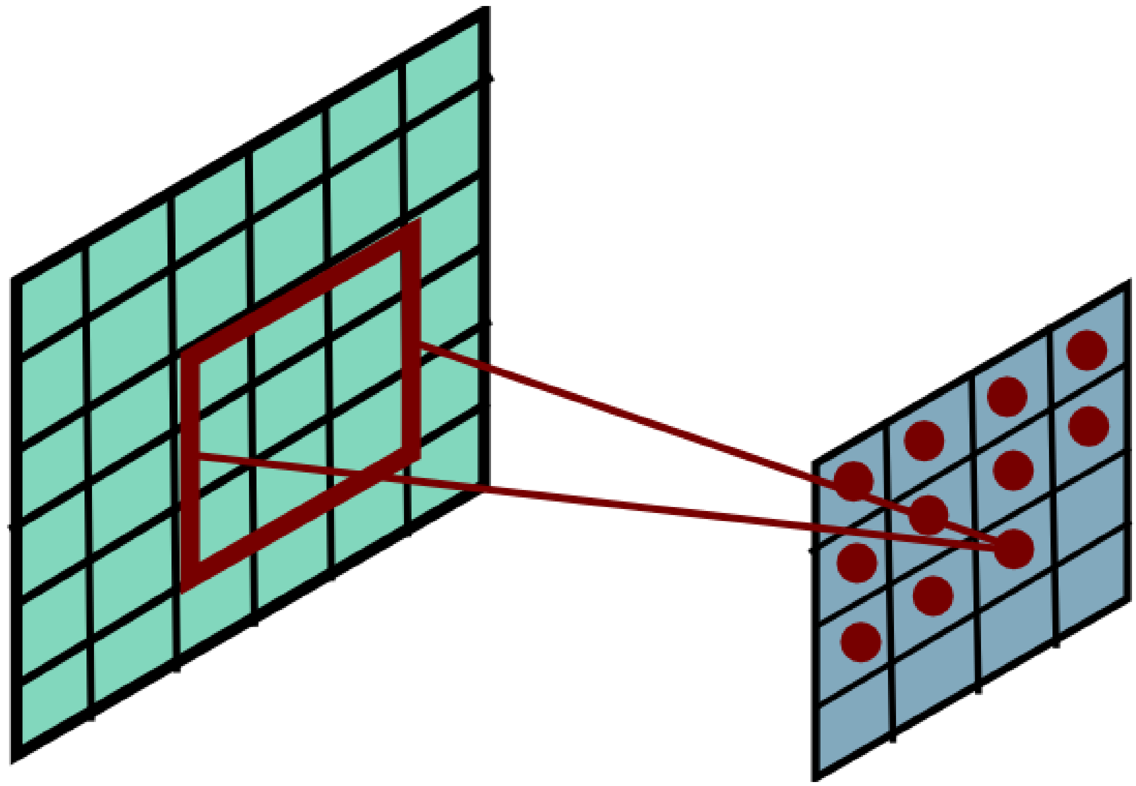
Convolution



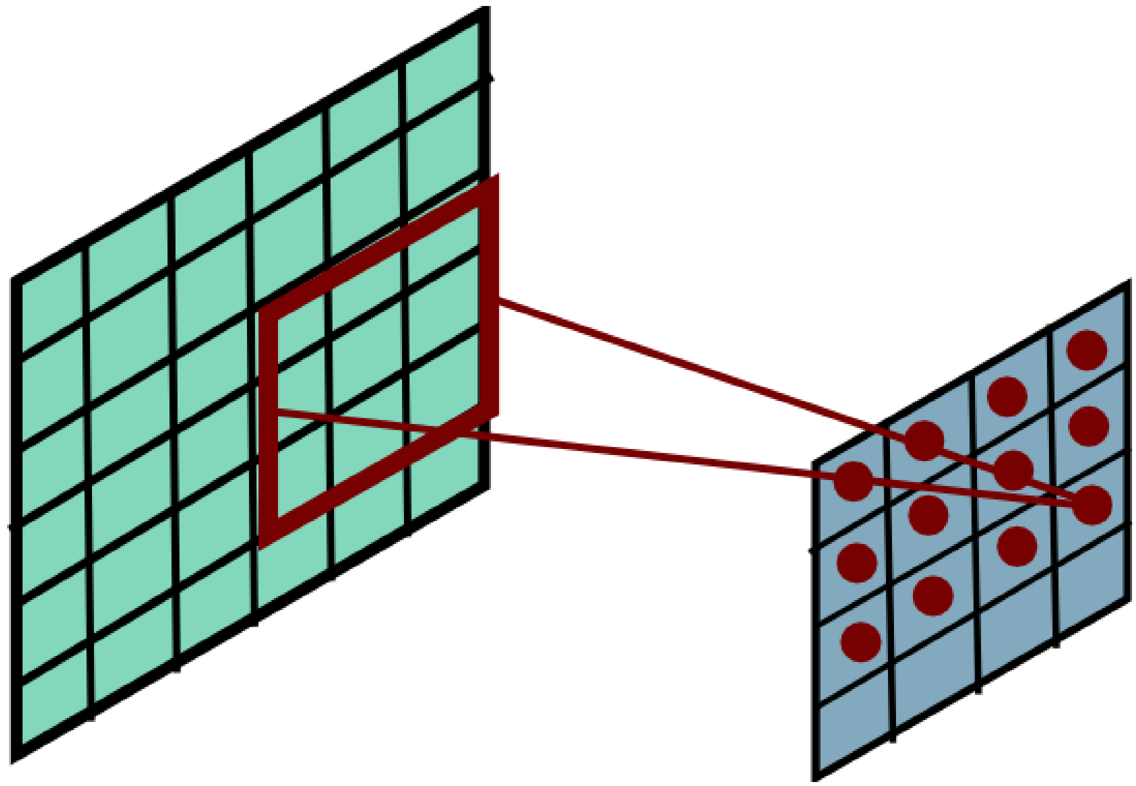
Convolution



Convolution

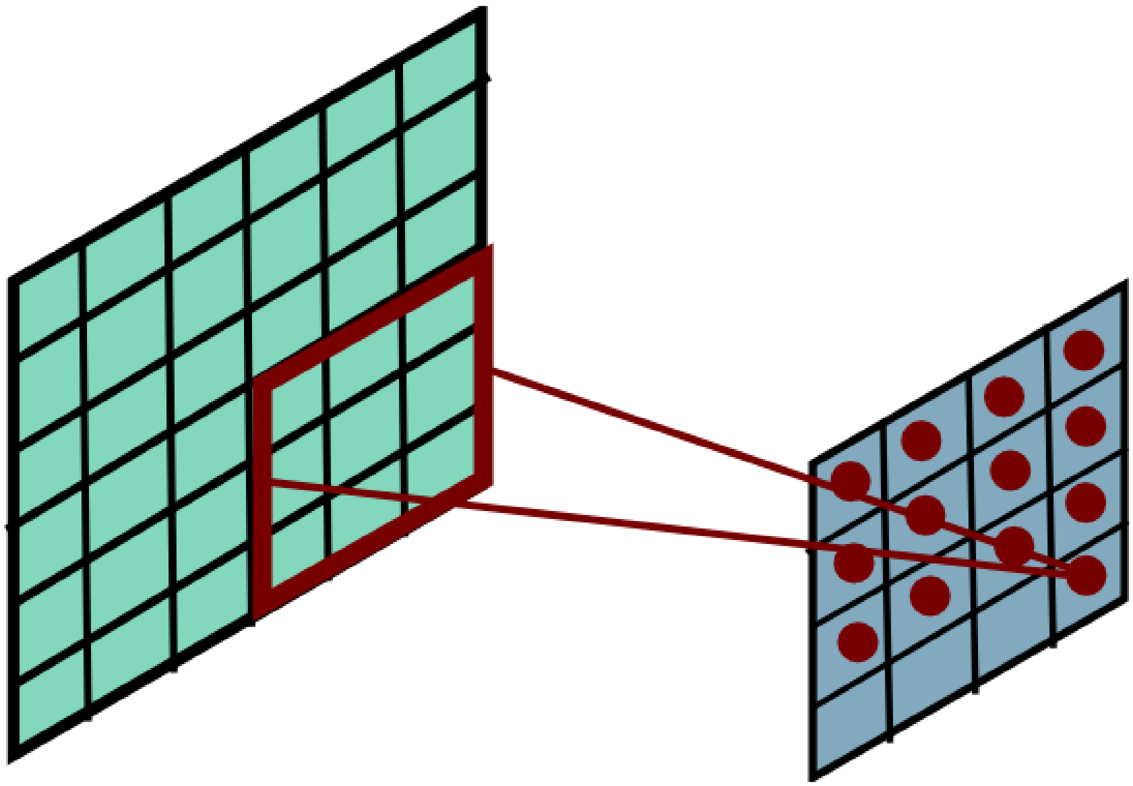


Convolution

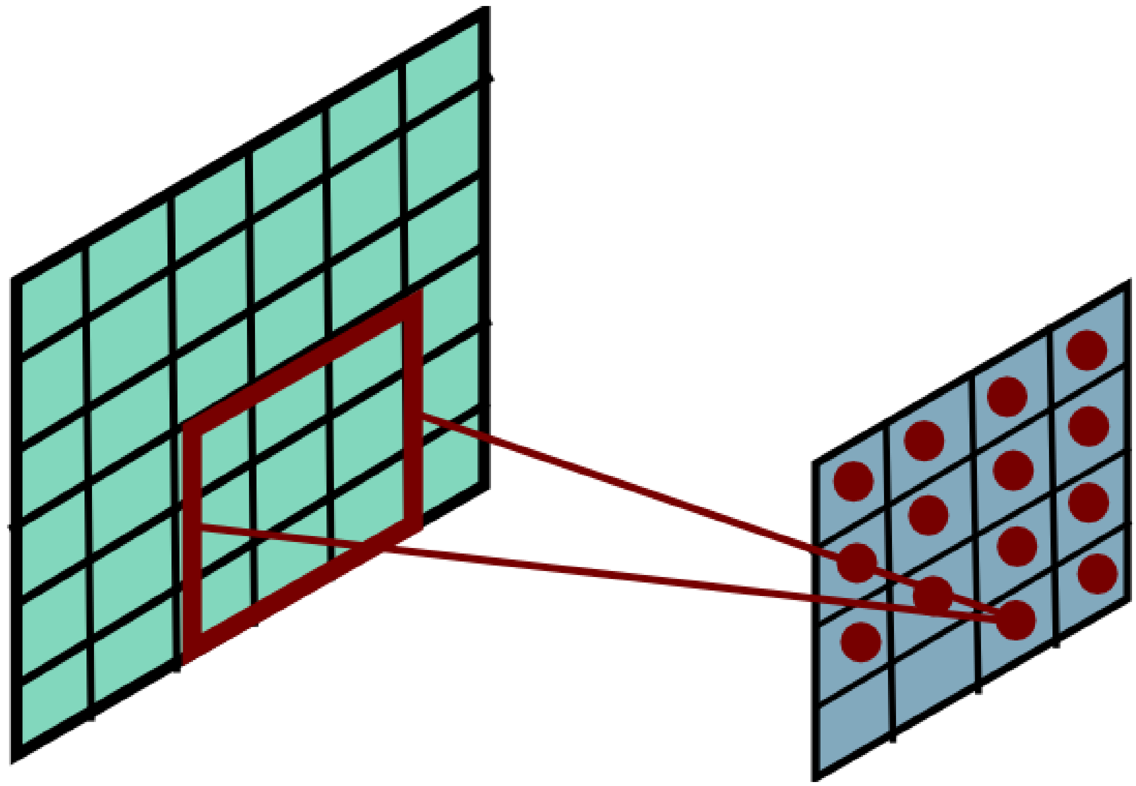


Slide Credit: Marc'Aurelio Ranzato

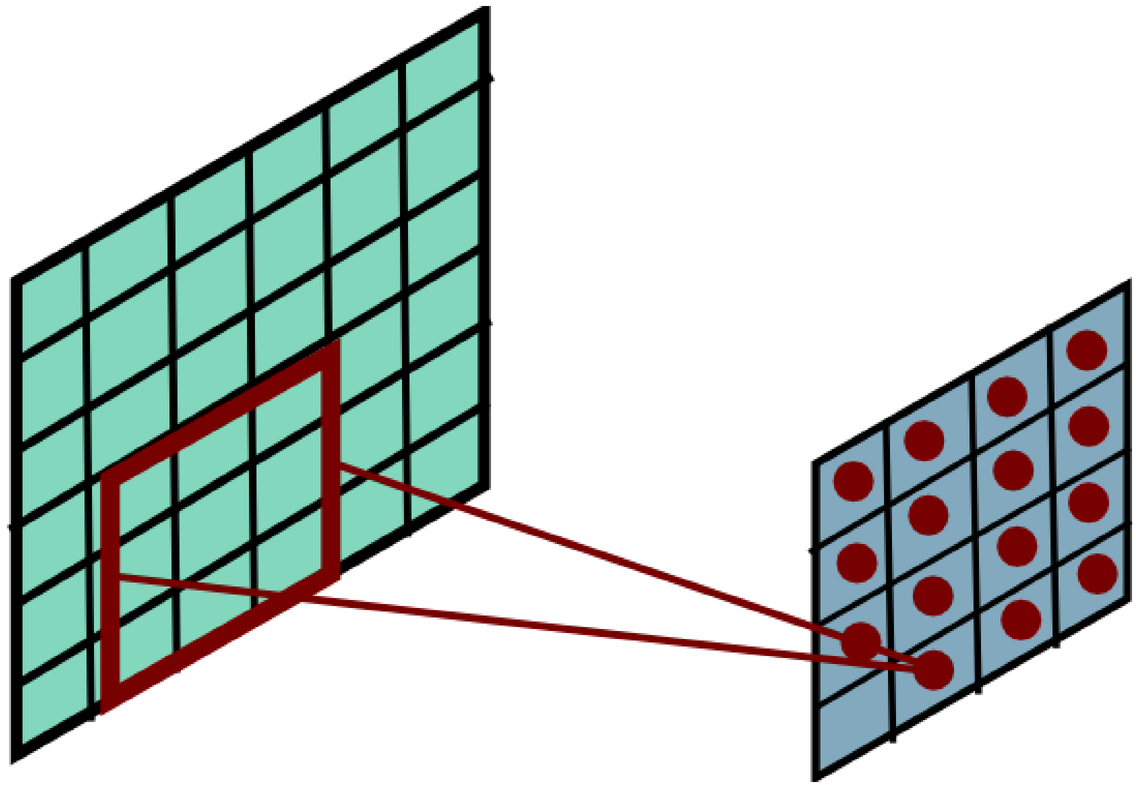
Convolution



Convolution

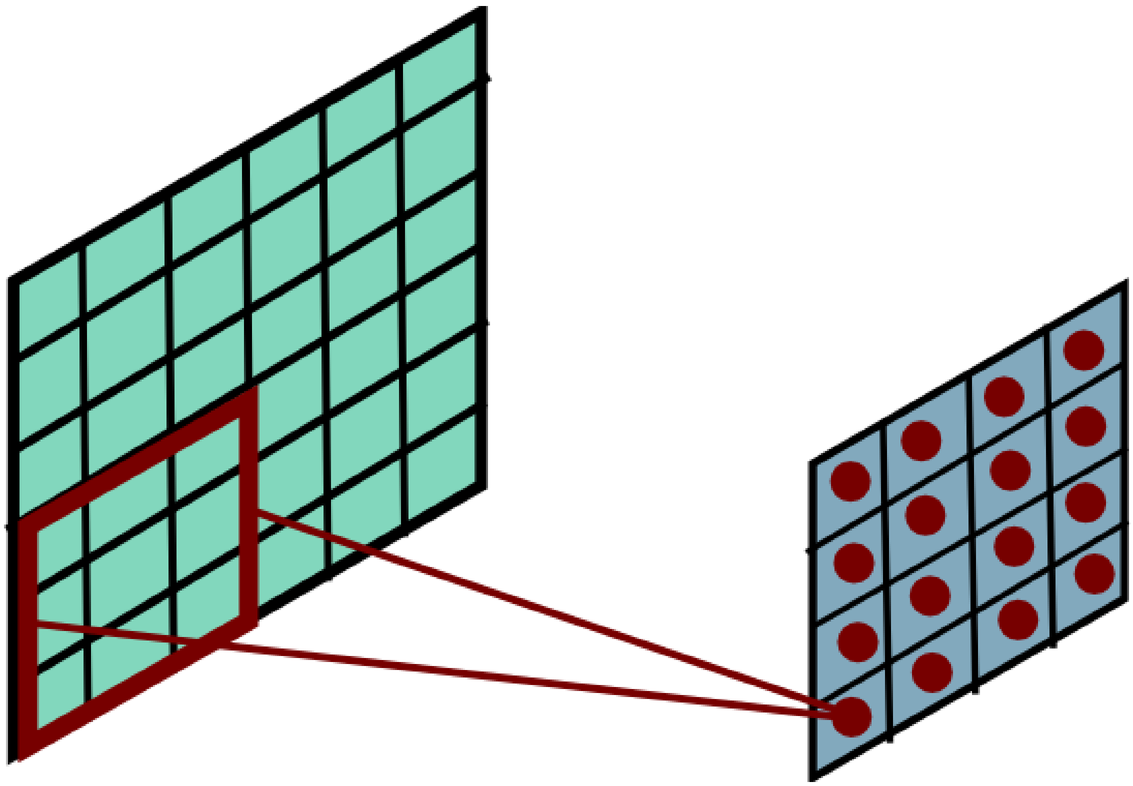


Convolution



Slide Credit: Marc'Aurelio Ranzato

Convolution



Mathieu et al. “Fast training of CNNs through FFTs” ICLR 2014

Convolution

These are the network parameters to be learned.

1	0	0	0	0	1
0	1	0	0	1	0
0	0	1	1	0	0
1	0	0	0	1	0
0	1	0	0	1	0
0	0	1	0	1	0

6 x 6 image

1	-1	-1
-1	1	-1
-1	-1	1

Filter 1

-1	1	-1
-1	1	-1
-1	1	-1

Filter 2

⋮ ⋮

Each filter detects a small pattern (3 x 3).

Convolution

1	-1	-1
-1	1	-1
-1	-1	1

Filter 1

stride=1

1	0	0	0	0	1
0	1	0	0	1	0
0	0	1	1	0	0
1	0	0	0	1	0
0	1	0	0	1	0
0	0	1	0	1	0

Dot product

3 -1

6 x 6 image

Convolution

1	-1	-1
-1	1	-1
-1	-1	1

Filter 1

If stride=2

1	0	0	0	0	1
0	1	0	0	1	0
0	0	1	1	0	0
1	0	0	0	1	0
0	1	0	0	1	0
0	0	1	0	1	0

6 x 6 image

3 -3

Convolution

1	-1	-1
-1	1	-1
-1	-1	1

Filter 1

stride=1

1	0	0	0	0	1
0	1	0	0	1	0
0	0	1	1	0	0
1	0	0	0	1	0
0	1	0	0	1	0
0	0	1	0	1	0

6 x 6 image

3	-1	-3	-1
-3	1	0	-3
-3	-3	0	1
3	-2	-2	-1

Convolution

-1	1	-1
-1	1	-1
-1	1	-1

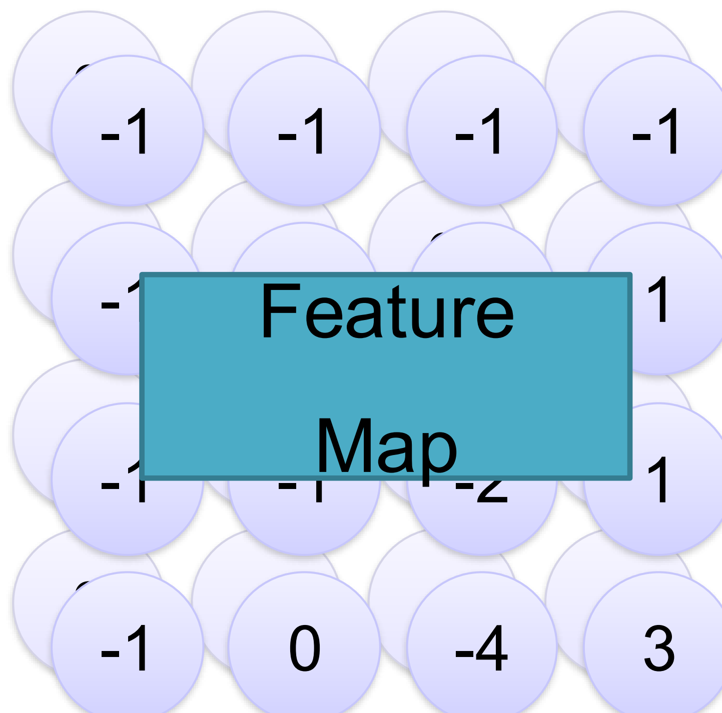
Filter 2

stride=1

Repeat this for each filter

1	0	0	0	0	1
0	1	0	0	1	0
0	0	1	1	0	0
1	0	0	0	1	0
0	1	0	0	1	0
0	0	1	0	1	0

6 x 6 image



Two 4 x 4 images

Forming 2 x 4 x 4 matrix

Example: convolution

$$* \begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix} =$$

Slide Credit: Marc'Aurelio Ranzato

Color image: RGB 3 channels

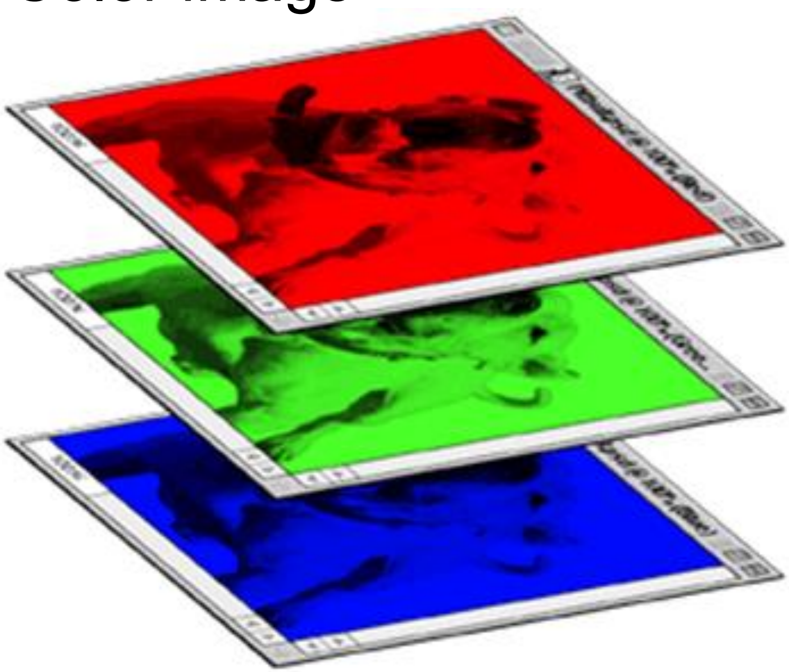
1	-1	-1
-1	1	-1
-1	-1	1

Filter 1

-1	1	-1
-1	1	-1
-1	1	-1

Filter 2

Color image



1	0	0	0	0	1
0	1	0	0	1	0
0	0	1	1	0	0
1	0	0	0	1	0
0	1	0	0	1	0
0	0	1	0	1	0

Convolution: output dimension

- **Stride controls** how the filter convolves around the input volume.
- The **amount by which the filter shifts is the stride**.

Output size:

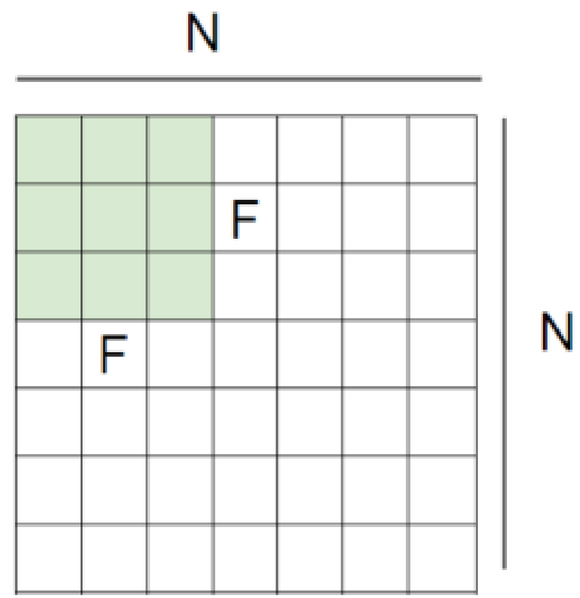
(N - F) / stride + 1

e.g. N = 7, F = 3:

stride 1 => (7 - 3)/1 + 1 = 5

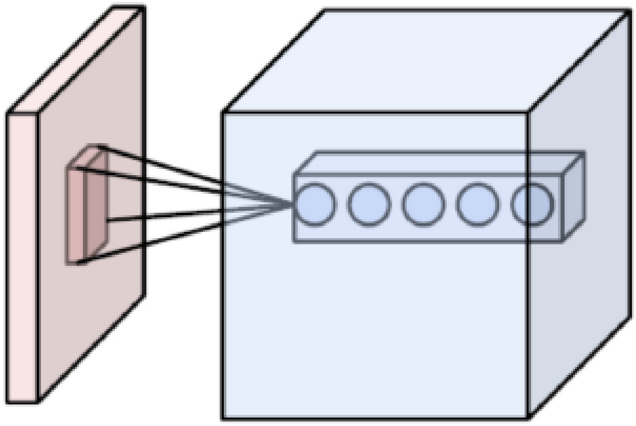
stride 2 => (7 - 3)/2 + 1 = 3

stride 3 => (7 - 3)/3 + 1 = ...



Source: Jie Chen slides

Example



Examples time:

Input volume: **32x32x3**

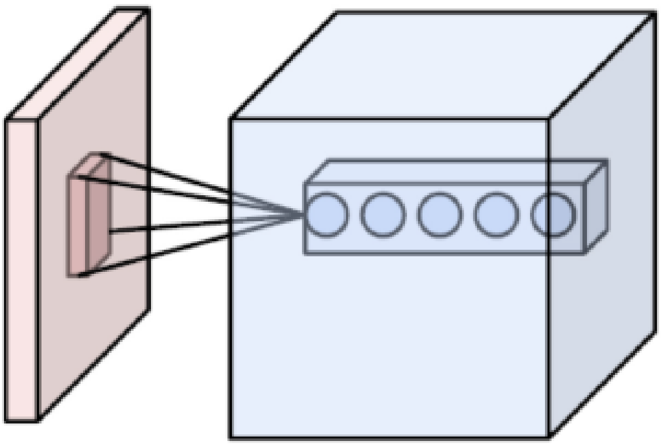
Receptive fields: **5x5, stride 1**

Number of neurons: **5**

Output volume: ?

Source: Jie Chen slides

Example



Examples time:

Input volume: **32x32x3**

Receptive fields: **5x5, stride 1**

Number of neurons: 5

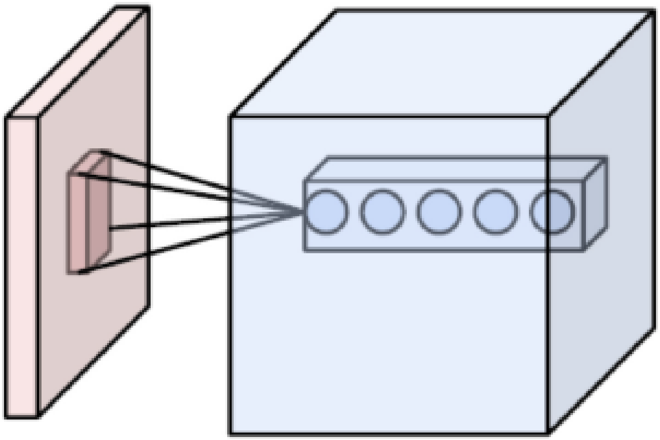
(N - F) / stride + 1

Output volume: (32 - 5) / 1 + 1 = 28, so: **28x28x5**

How many weights for each of the 28x28x5 neurons?

Source: Jie Chen slides

Example



Examples time:

Input volume: **32x32x3**

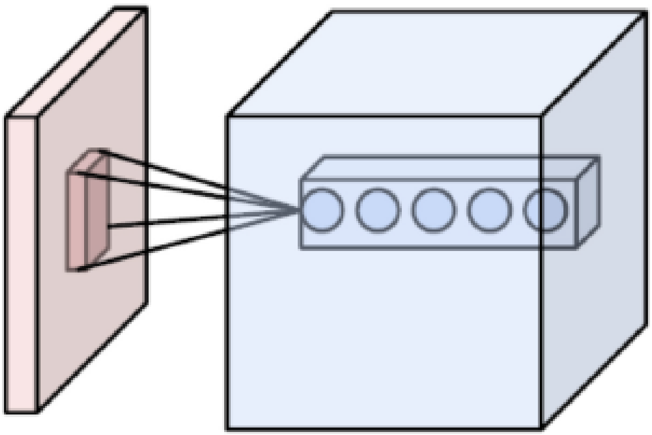
Receptive fields: **5x5, stride 2**

Number of neurons: **5**

Output volume: ?

Source: Jie Chen slides

Example



Examples time:

Input volume: **32x32x3**

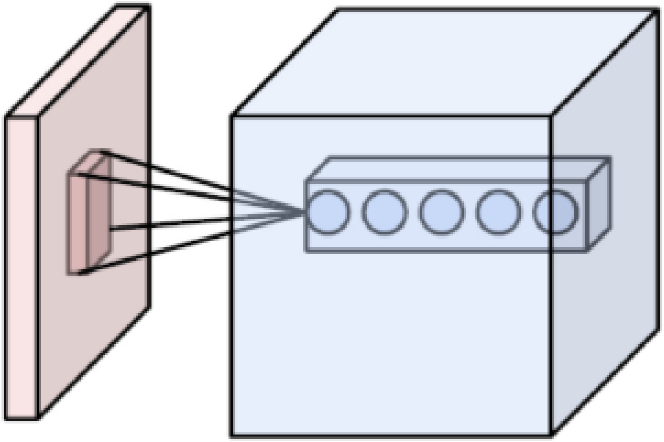
Receptive fields: **5x5, stride 2**

Number of neurons: **5**

Output volume: ?

Source: Jie Chen slides

Example



Examples time:

Input volume: **32x32x3**

Receptive fields: **5x5, stride 2**

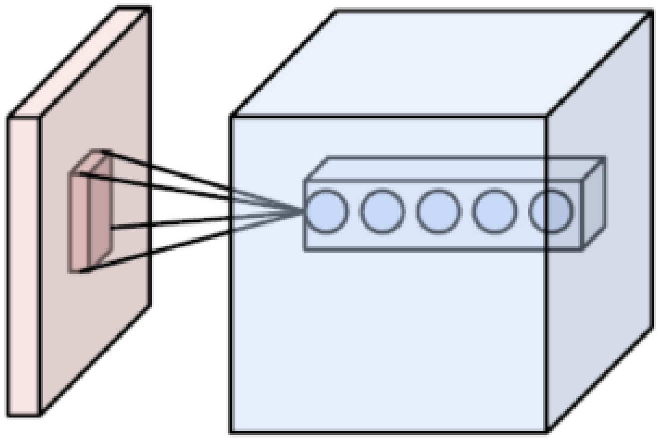
Number of neurons: **5**

(N - F) / stride + 1

Output volume: ? Cannot: (32-5)/2 + 1 = 14.5

Source: Jie Chen slides

Example



Examples time:

Input volume: **32x32x3**

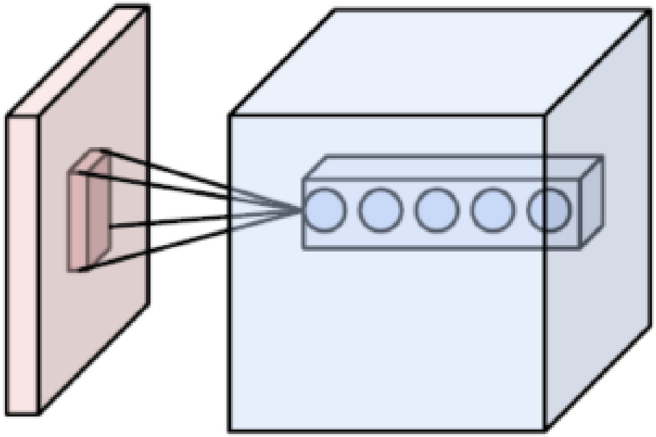
Receptive fields: **5x5, stride 3**

Number of neurons: **5**

Output volume: ?

Source: Jie Chen slides

Example



Examples time:

Input volume: **32x32x3**

Receptive fields: **5x5, stride 3**

Number of neurons: **5**

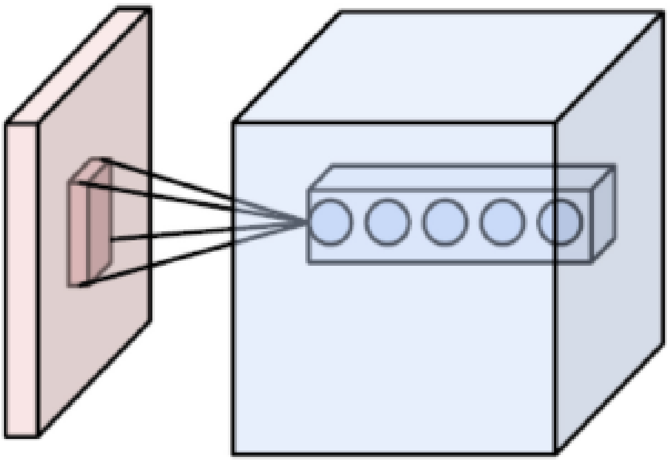
(N - F) / stride + 1

Output volume: (32 - 5) / 3 + 1 = 10, so: **10x10x5**

How many weights for each of the 10x10x5 neurons?

Source: Jie Chen slides

Example



Examples time:

Input volume: **32x32x3**

Receptive fields: **5x5, stride 3**

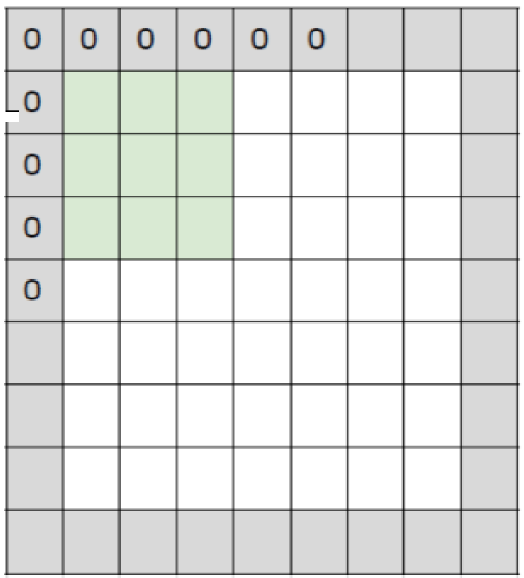
Number of neurons: **5**

Output volume: (32 - 5) / 3 + 1 = 10, so: **10x10x5**

Source: Jie Chen slides

In practice: Common to apply zero padding

- Padding zero in the border of images (for each channels/ feature maps)



e.g. input 7x7
neuron with receptive field 3x3, stride 1
pad with 1 pixel border => what is the output?

Source: Jie Chen slides

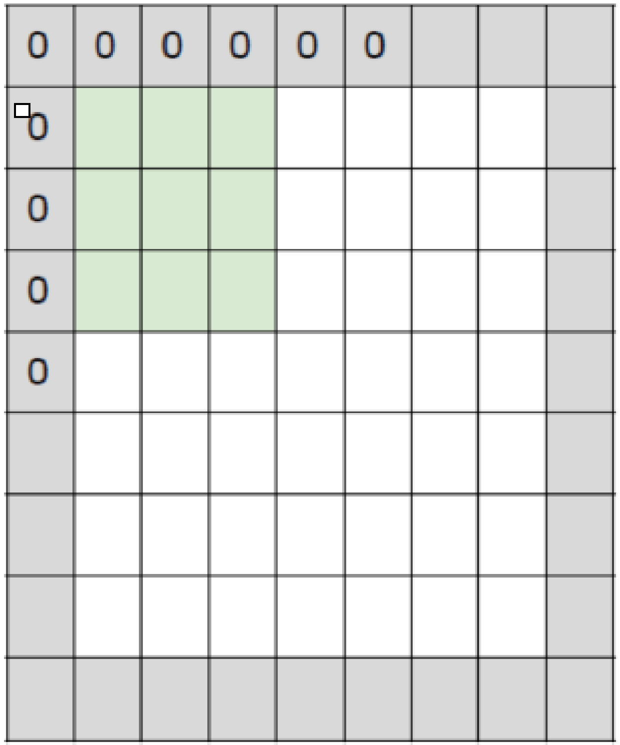
Output dimension with padding

- The mathematical representation with padding p as follows:

$$n_{out} = \left\lfloor \frac{n_{in} + 2p - k}{s} \right\rfloor + 1$$

n_{in}: number of input features
n_{out}: number of output features
k: convolution kernel size
p: convolution padding size
s: convolution stride size

In practice: Common to zero padding



e.g. input 7x7
neuron with receptive field 3x3, stride 1
pad with 1 pixel border => what is the output?

(N - F) / stride + 1=(9-3)/1+1=7

7x7 => preserved size!
in general, common to see stride 1, size F, and zero-padding with (F-1)/2.
(Will preserve input size spatially)

Source: Jie Chen slides

Types of Convolution

"Same convolution" (preserves size)

Input [9x9]

3x3 neurons, stride 1, pad **1** =>[9x9]

3x3 neurons, stride 1, pad **1** =>[9x9]

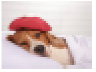
- No headaches when sizing architectures
- Works well

"Valid convolution" (shrinks size)

Input [9x9]

3x3 neurons, stride 1, pad **0** =>[7x7]

3x3 neurons, stride 1, pad **0** =>[5x5]



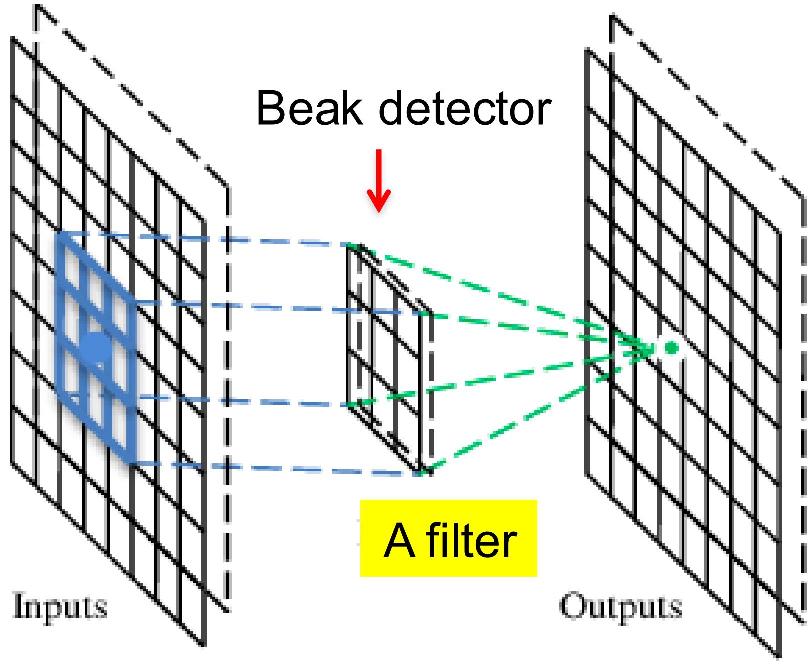
- **Headaches** with sizing the full architecture
- **Works Worse!** Border information will "wash away", since those values are only used once in the forward function

Source: Jie Chen slides

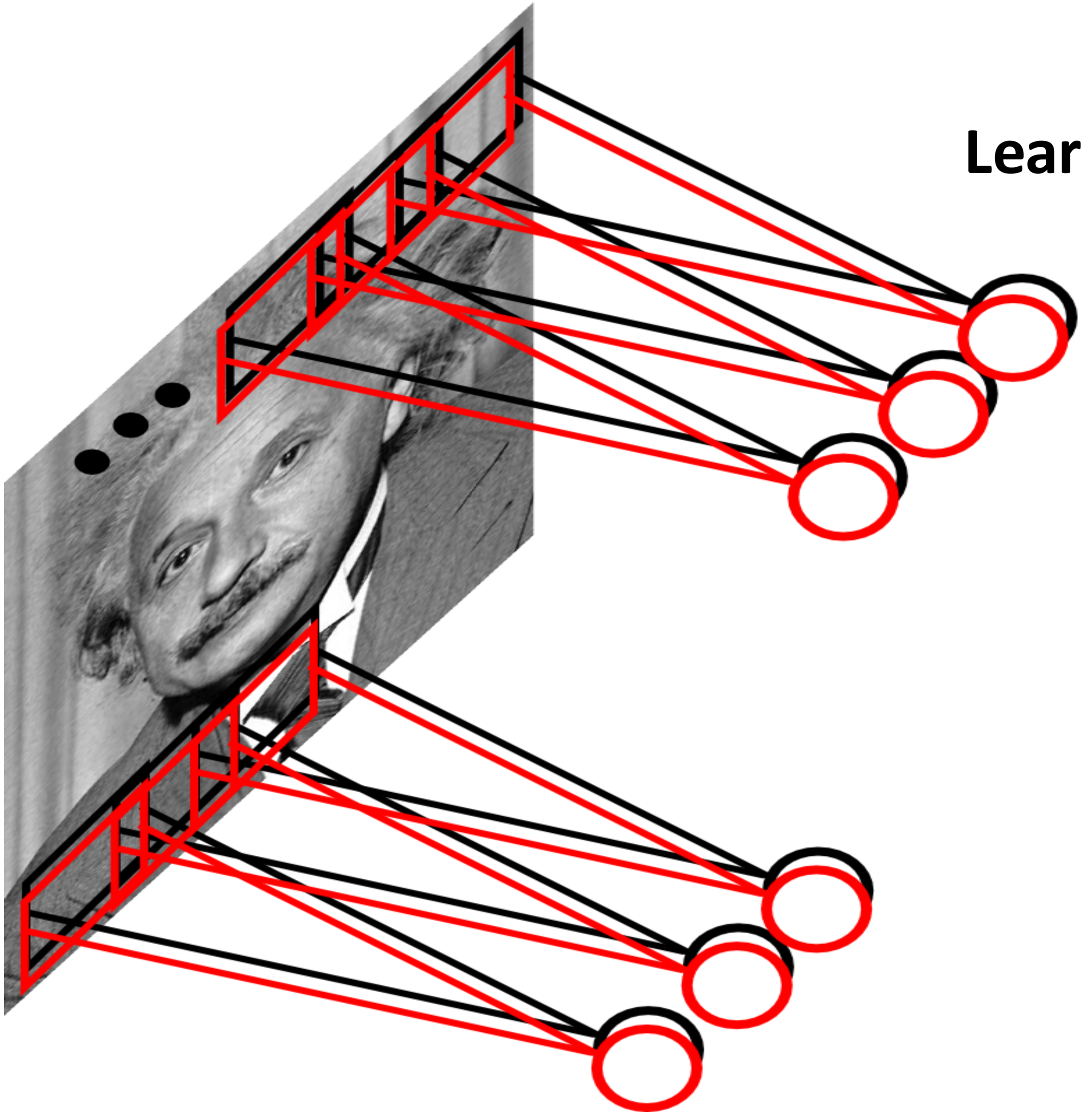
A convolutional layer

A convolutional layer has a number

of filters that does convolutional operation.



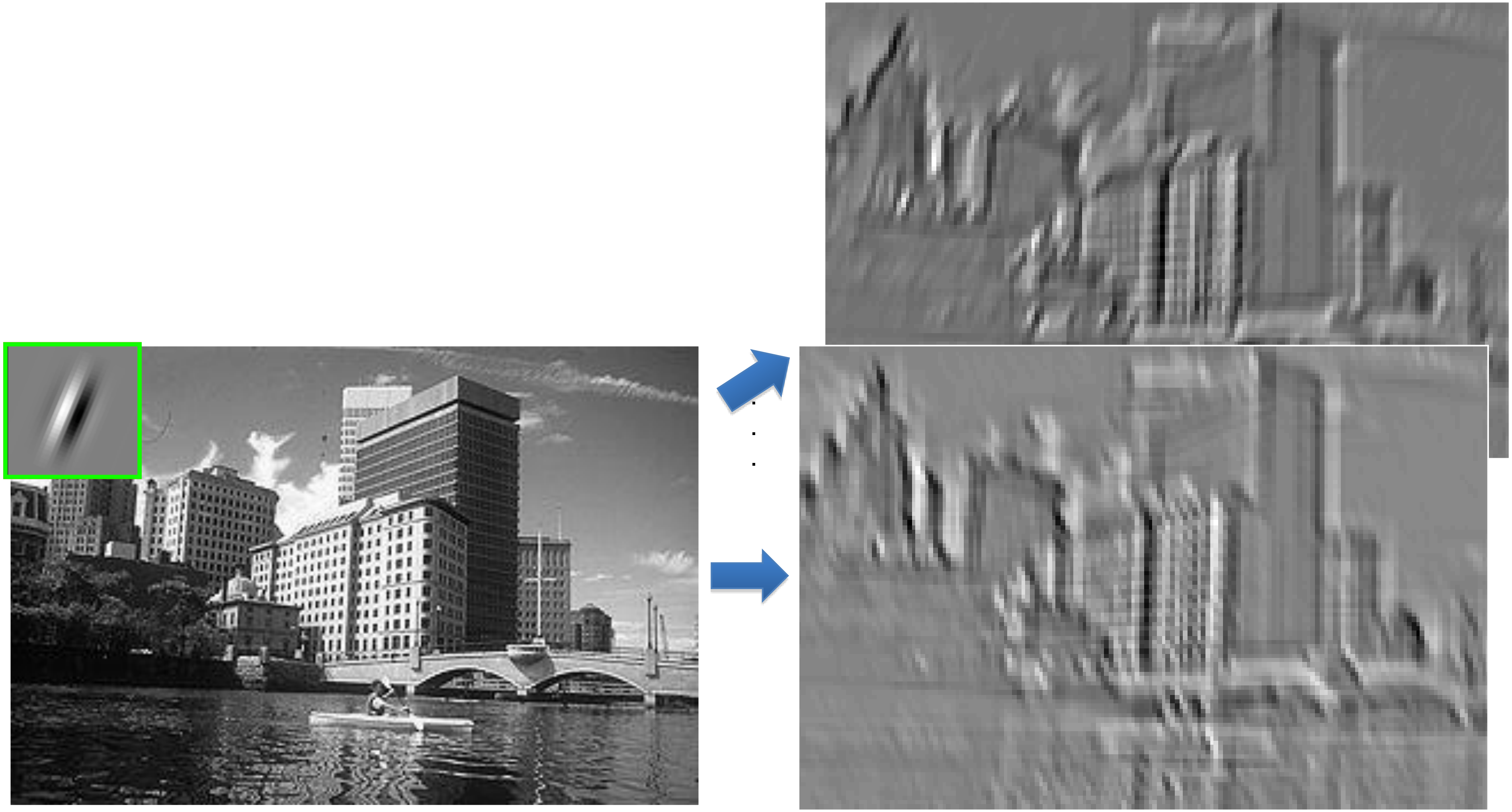
Convolutional Layer



Learn multiple filters.

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

Convolution for feature extraction

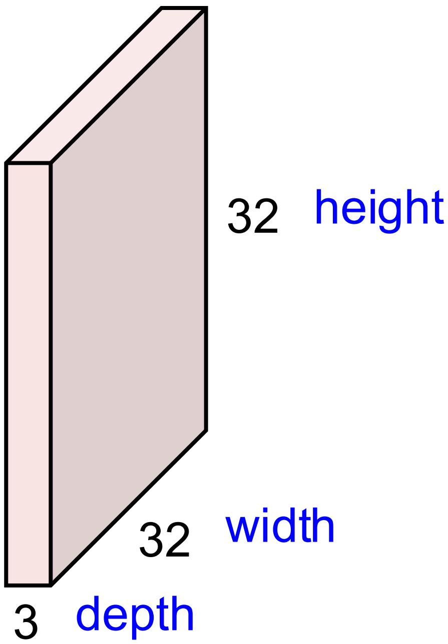


Input

Feature Map

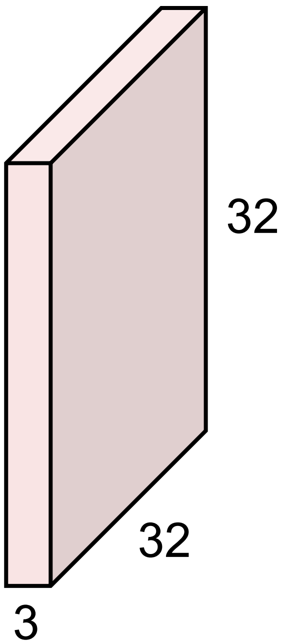
Convolution Layer

32x32x3 image -> preserve spatial structure

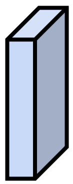


Convolution Layer

32x32x3 image



5x5x3 filter

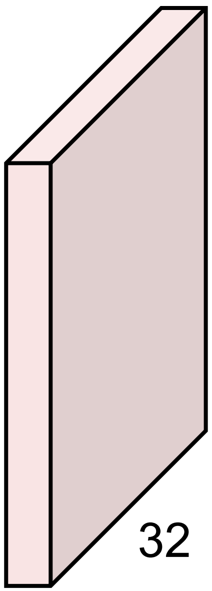


Convolve the filter with the image
i.e. "slide over the image spatially, computing dot products"

Convolution Layer

Filters always extend the full depth of the input volume

32x32x3 image

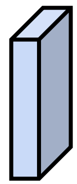


32

32

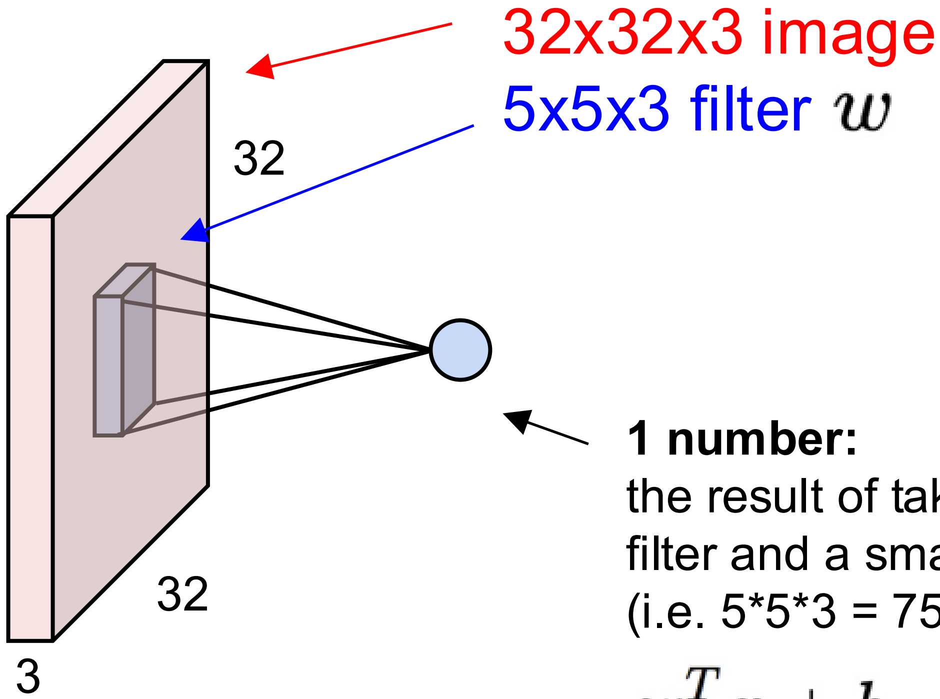
3

5x5x3 filter



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

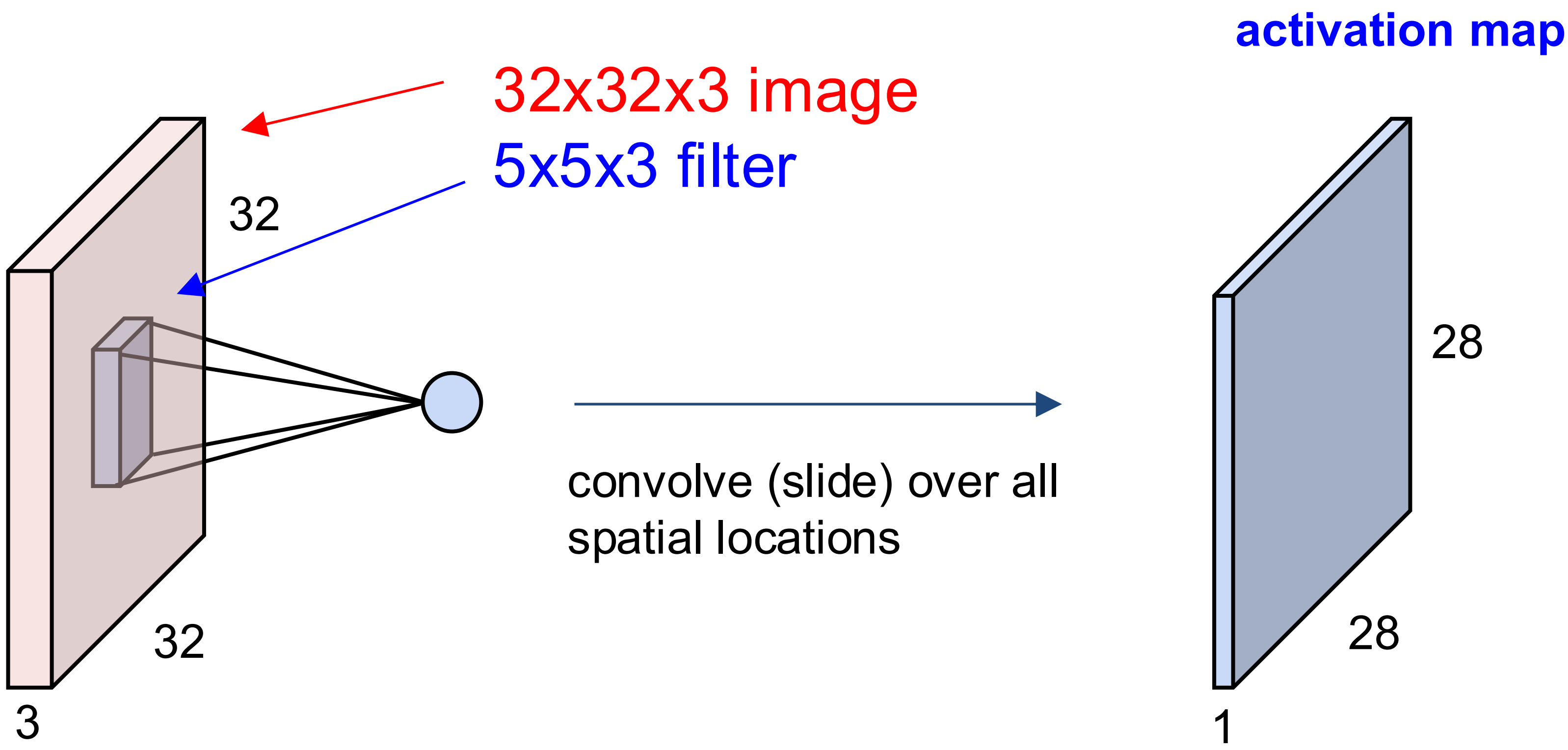
Convolution Layer



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)

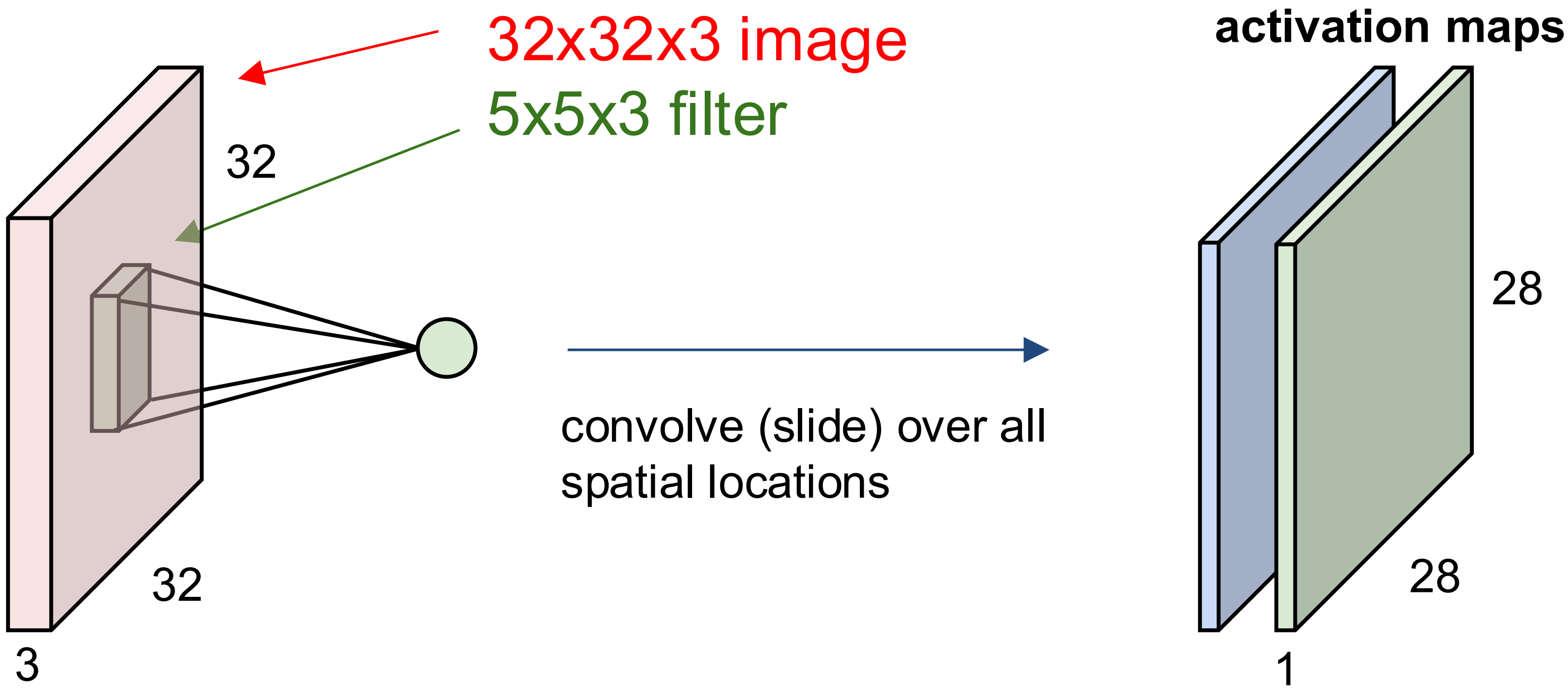
$$w^T x + b$$

Convolution Layer

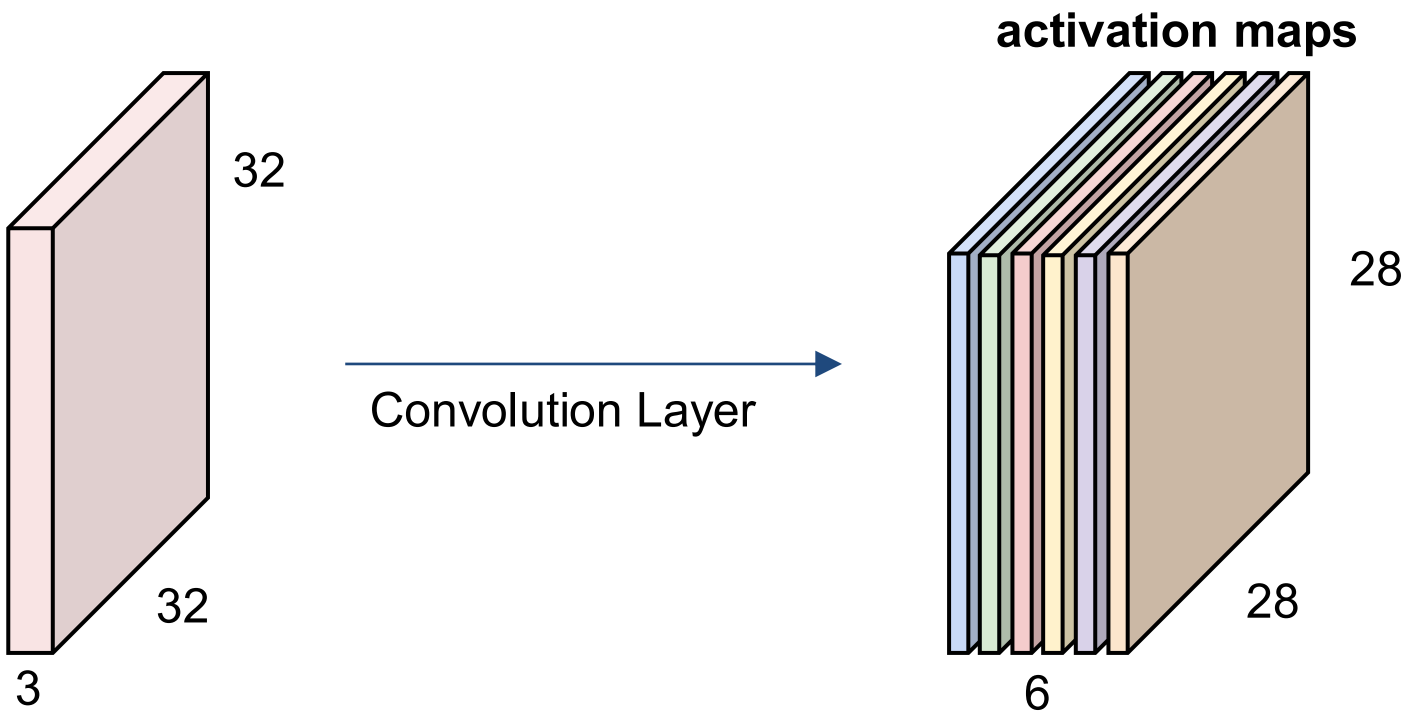


Convolution Layer

consider a second, green filter

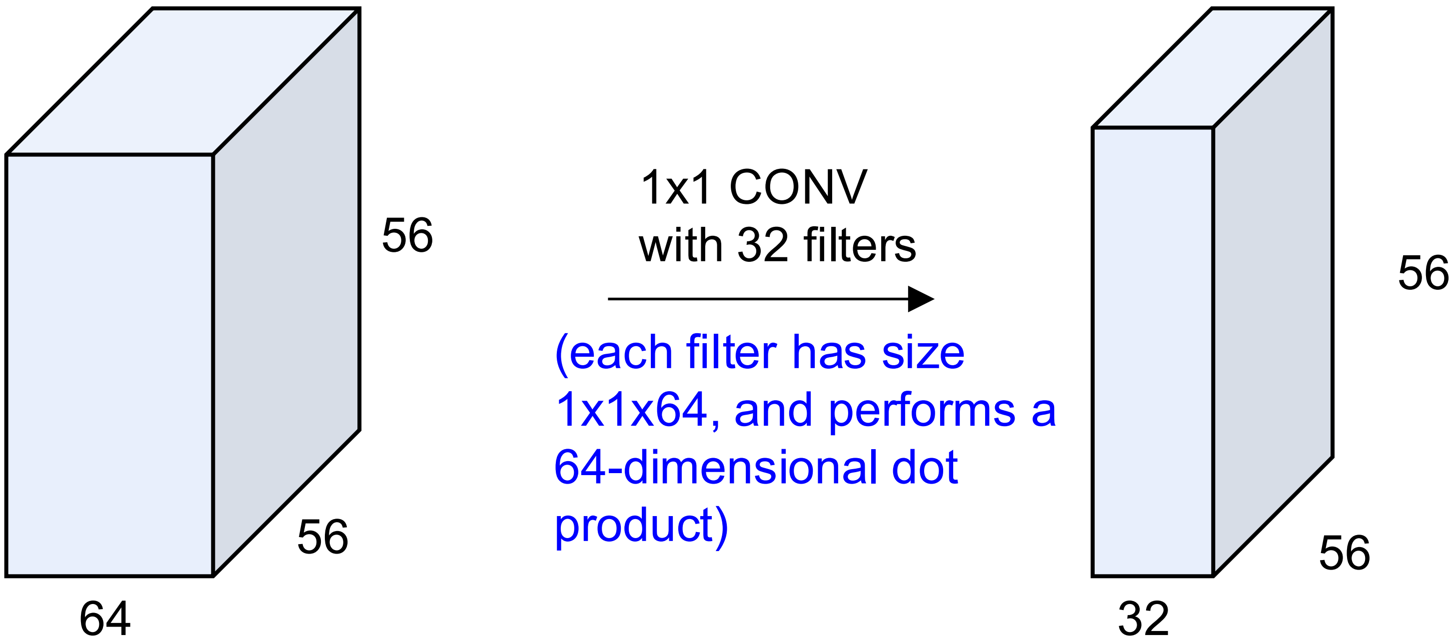


For example, if we had 6x5x5 filters, we'll get 6 separate activation maps:



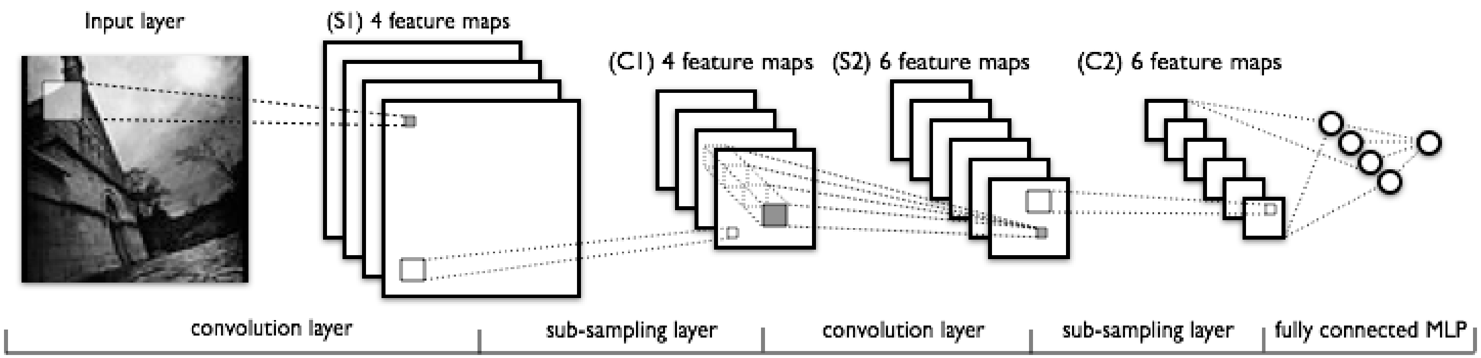
We stack these up to get a "new image" of size 28x28x6!

(Between, 1x1 convolution layers make perfect sense)



Convolutional Neural Networks (CNN)

- A CNN is a neural network with some convolutional layers (and some other layers).



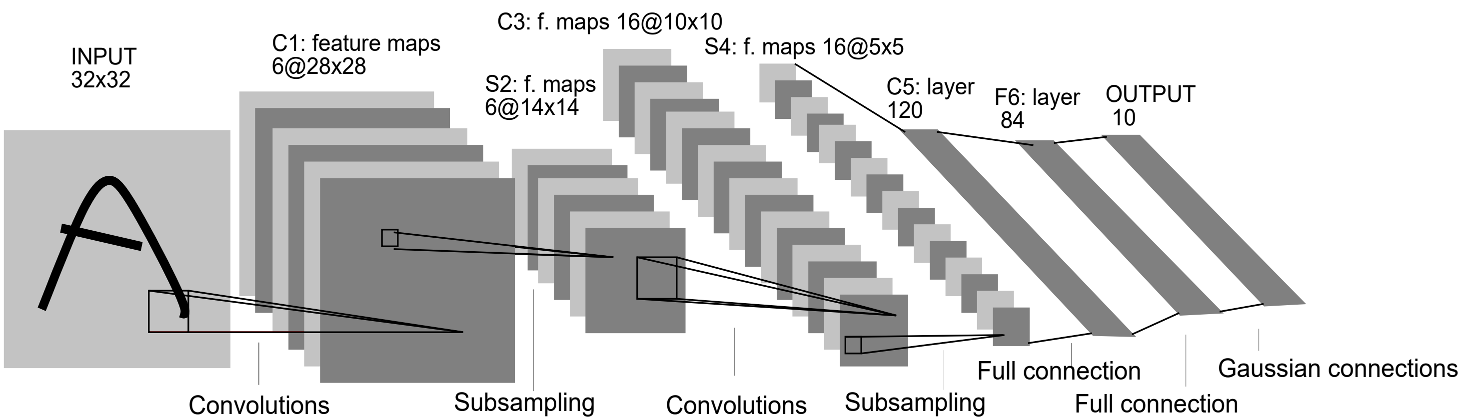
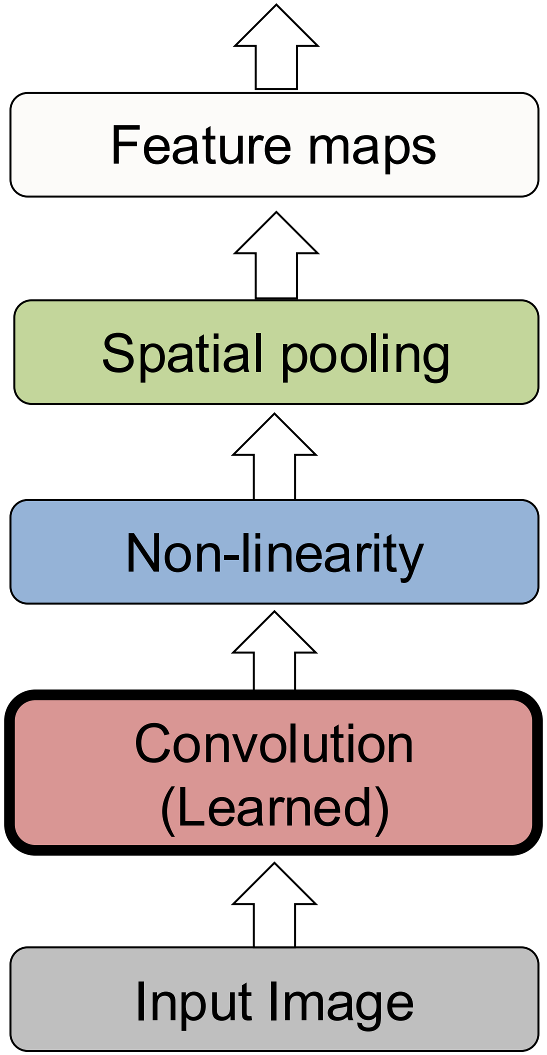
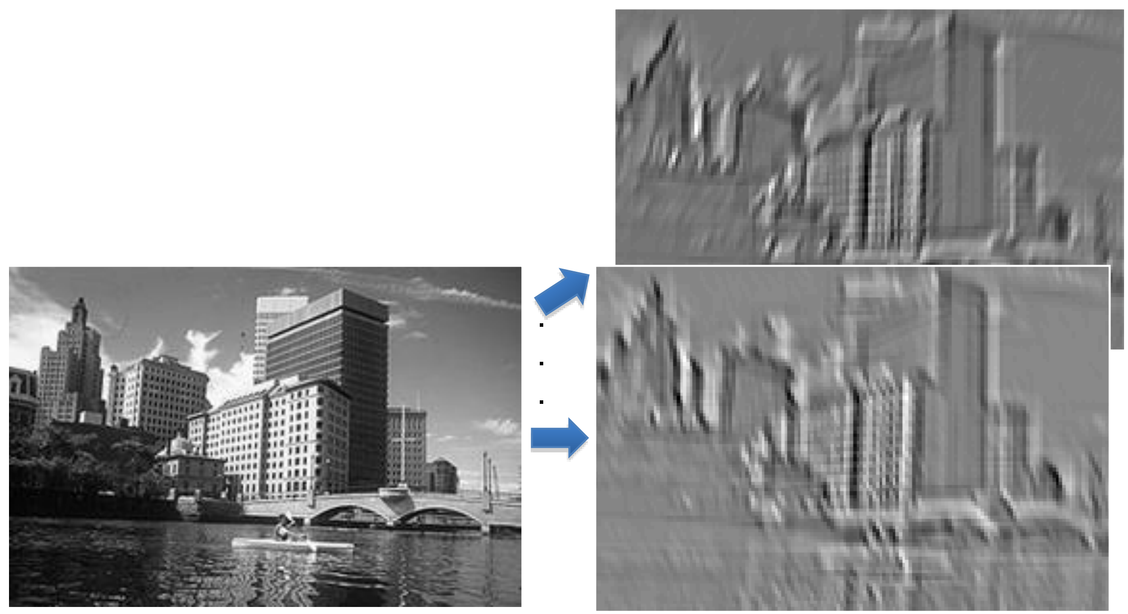


Image Credit: Yann LeCun, Kevin Murphy

Key operations in a CNN



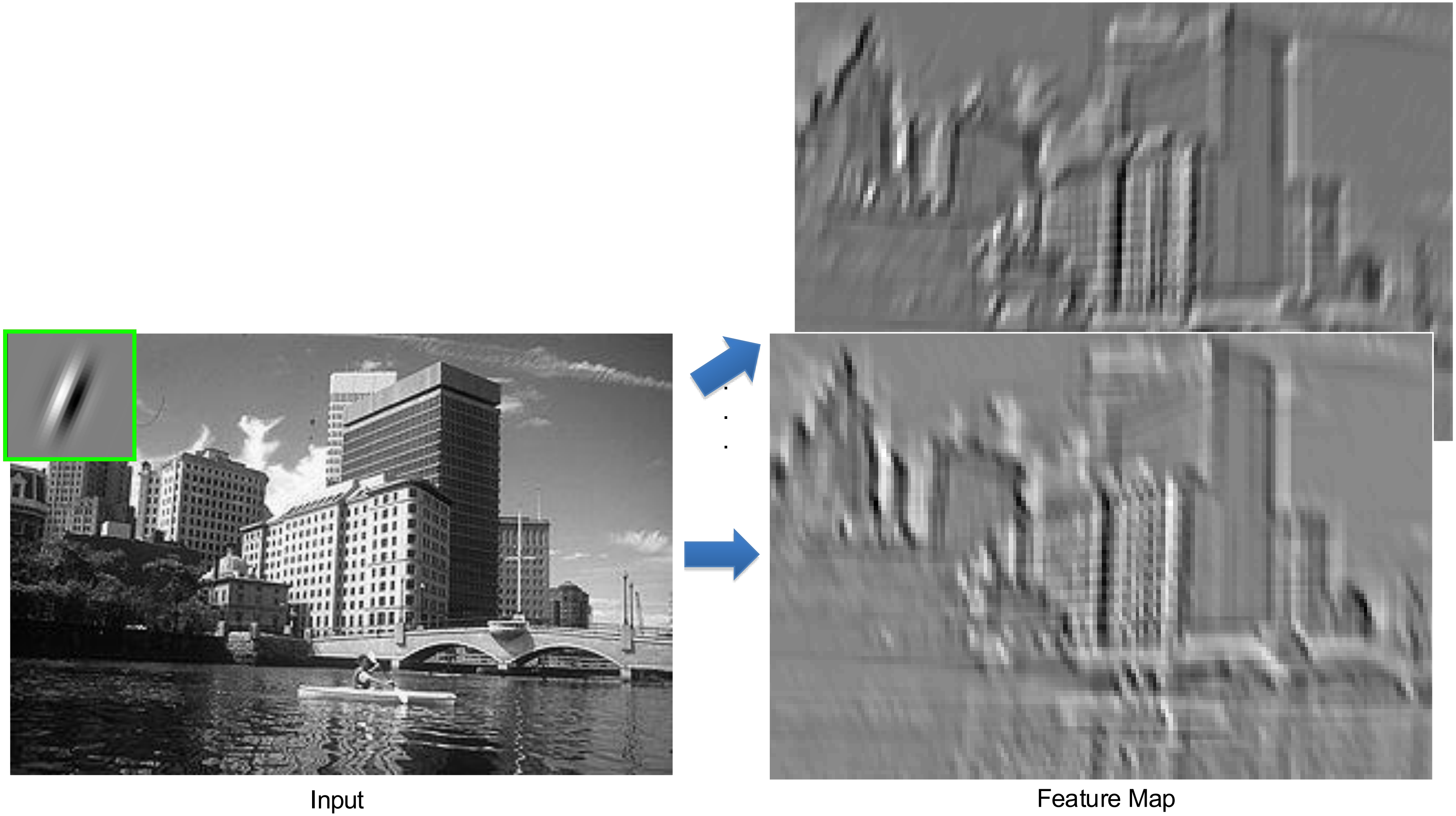


Input

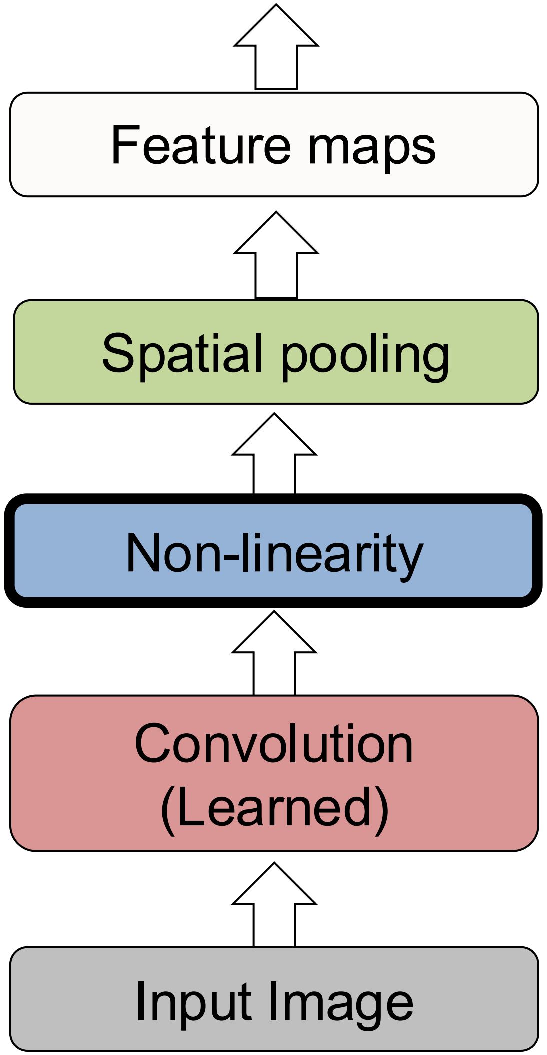
Feature Map

Source: R. Fergus, Y. LeCun

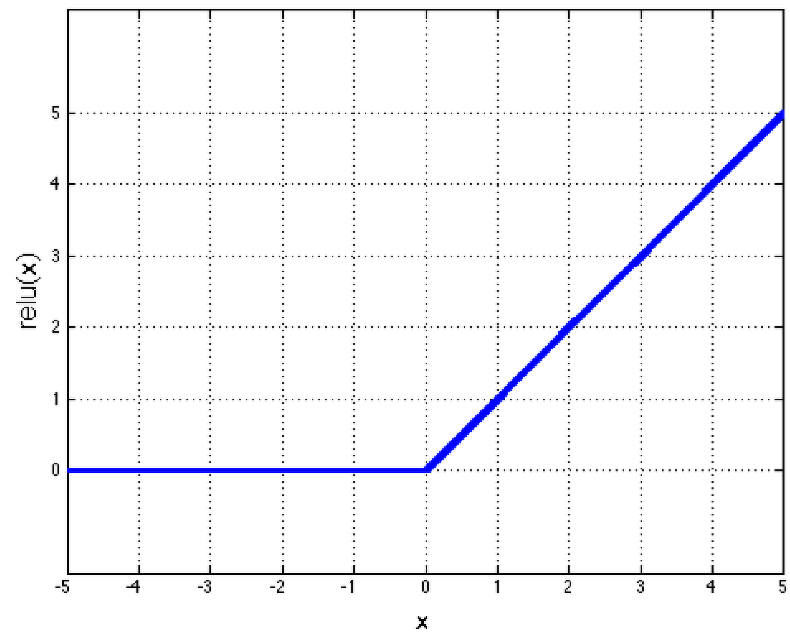
Convolution as feature extraction



Key operations : Activation function



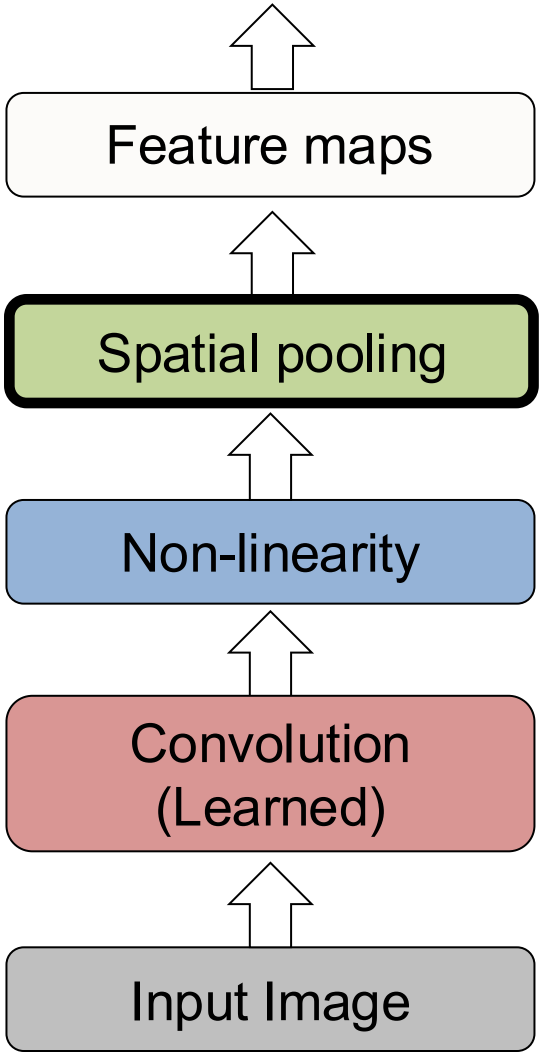
Rectified Linear Unit (ReLU)

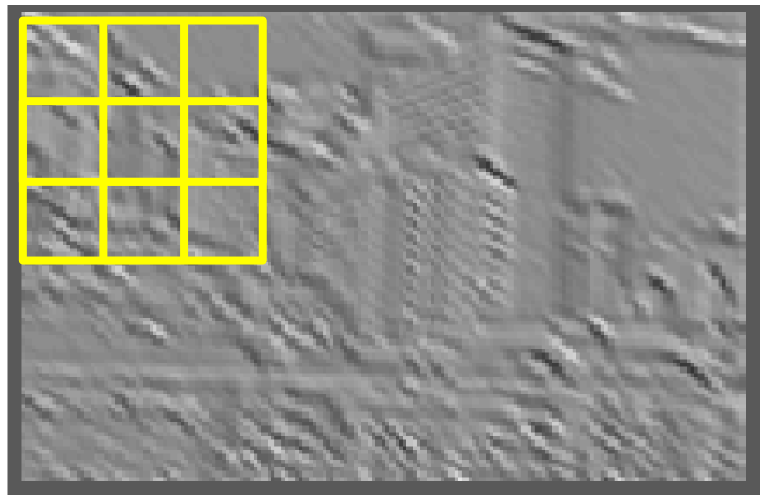


Y=max(0, X)

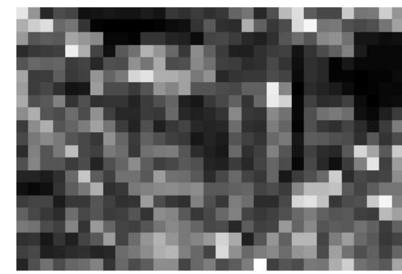
Source: R. Fergus, Y. LeCun

Key operation : pooling





Max



Number of feature maps don't change

Source: R. Fergus, Y. LeCun

Why Pooling

- Subsampling pixels will not change the object

bird

Subsampling

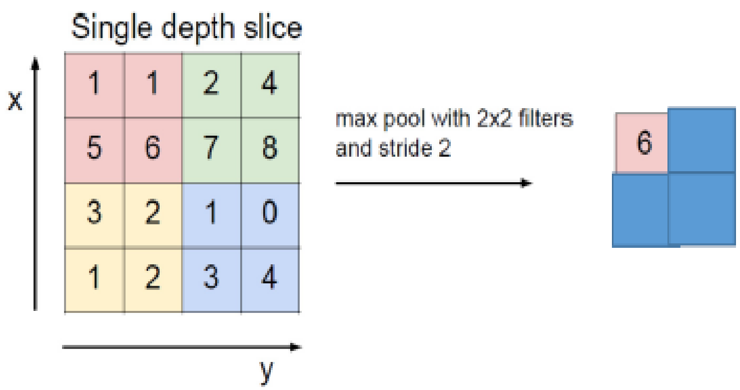
bird

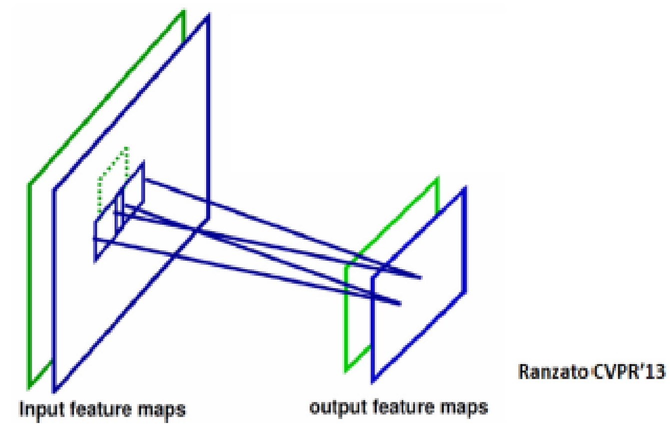
We can subsample the pixels to make image smaller

➧ fewer parameters to characterize the image

Key operations : pooling

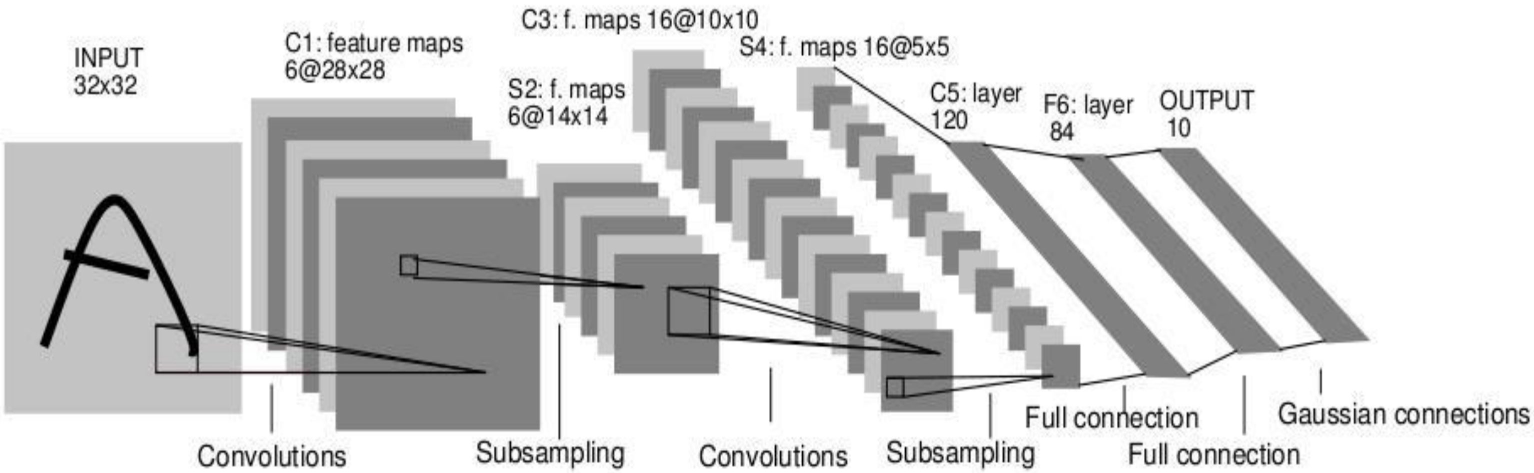
- Max-pooling
  - partitions the input image into a set of rectangles, and for each sub-region, outputs the maximum value
  - Non-linear down-sampling
  - The number of output maps is the same as the number of input maps, but the resolution is reduced
  - Reduce the computational complexity for upper layers and provide a form of translation invariance
- Average pooling can also be used
- for **pool layers**, use pool size 2x2 (more = worse)





Source: Jie Chen slides

LeNet-5



- Average pooling
- Sigmoid or tanh nonlinearity
- Fully connected layers at the end
- Trained on MNIST digit dataset with 60K training examples

Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, [Gradient-based learning applied to document recognition](), Proc. IEEE 86(11): 2278–2324, 1998.

AlexNet model: feature visualization

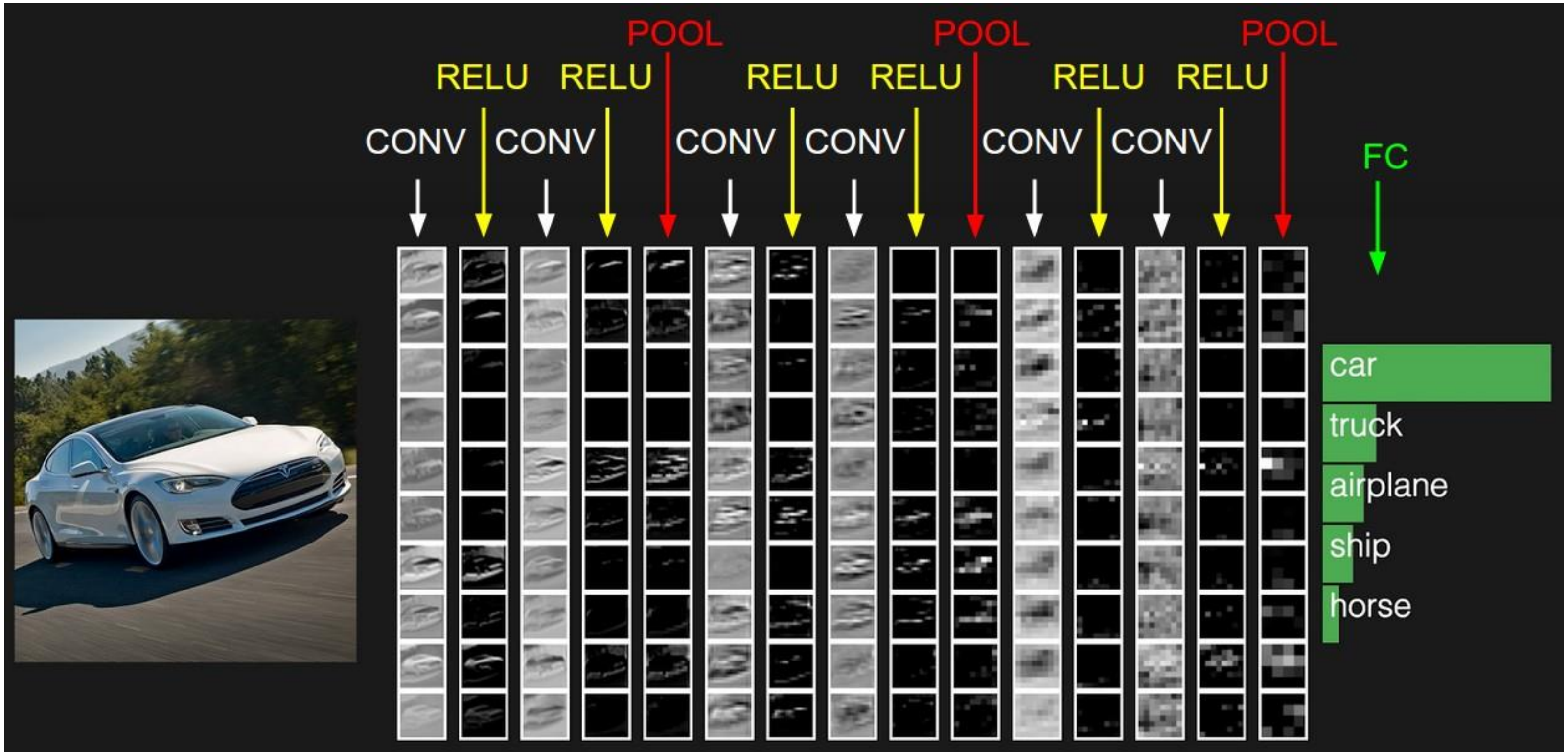
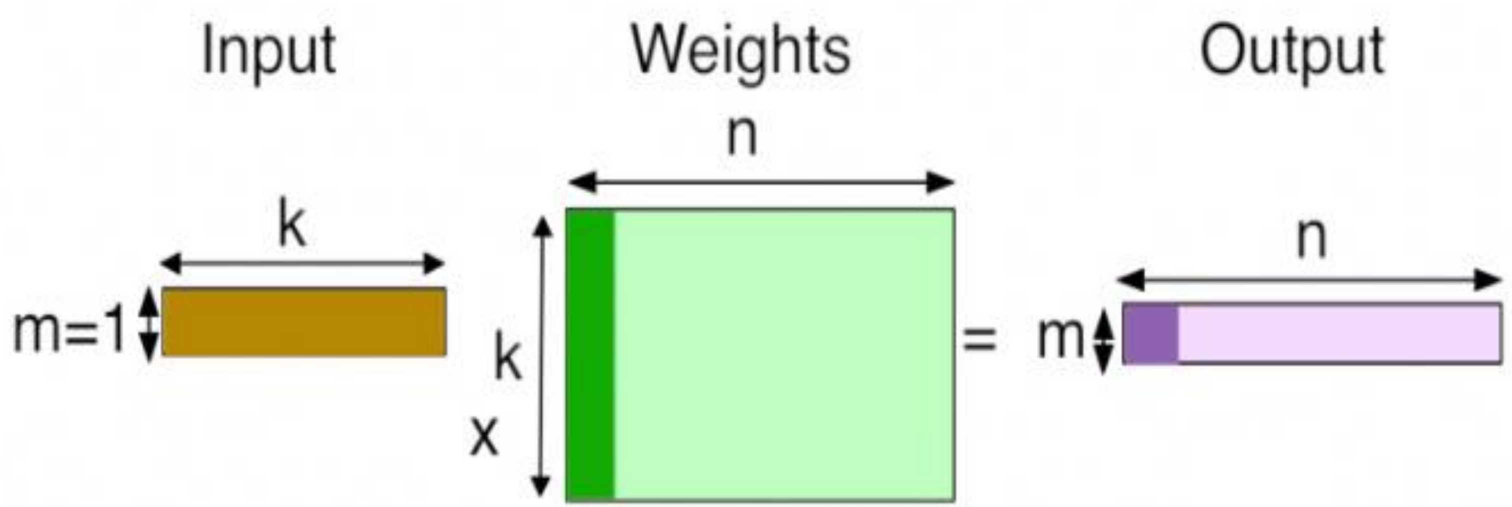


Figure source: A. Karpathy

Back-prop in convolutional network

- Notes
  - https://www.cc.gatech.edu/classes/AY2018/cs7643_fall/slides/L6_cnns_backprop_notes.pdf

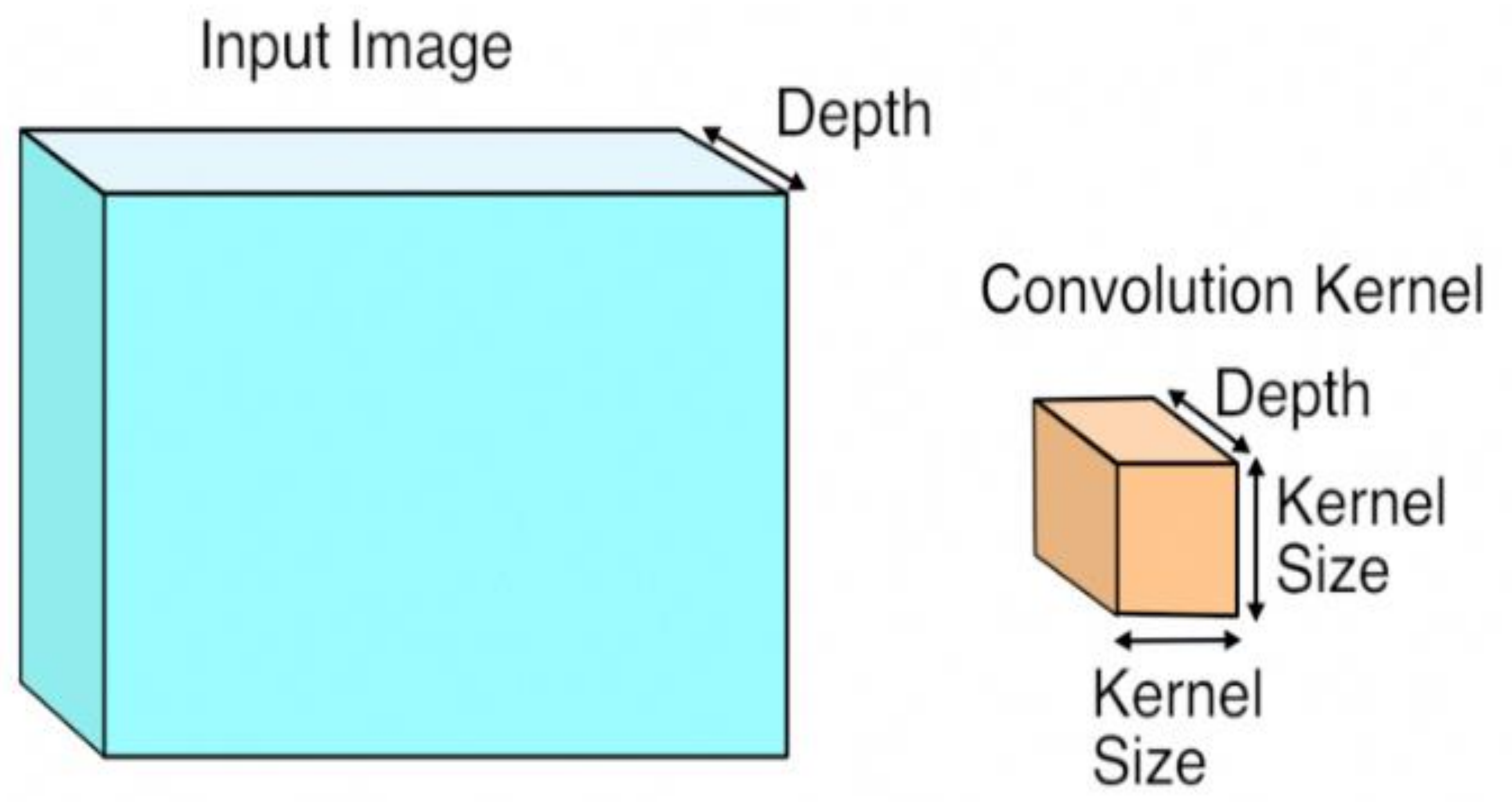
FC Layer Implementation



Operations in fully connected layer

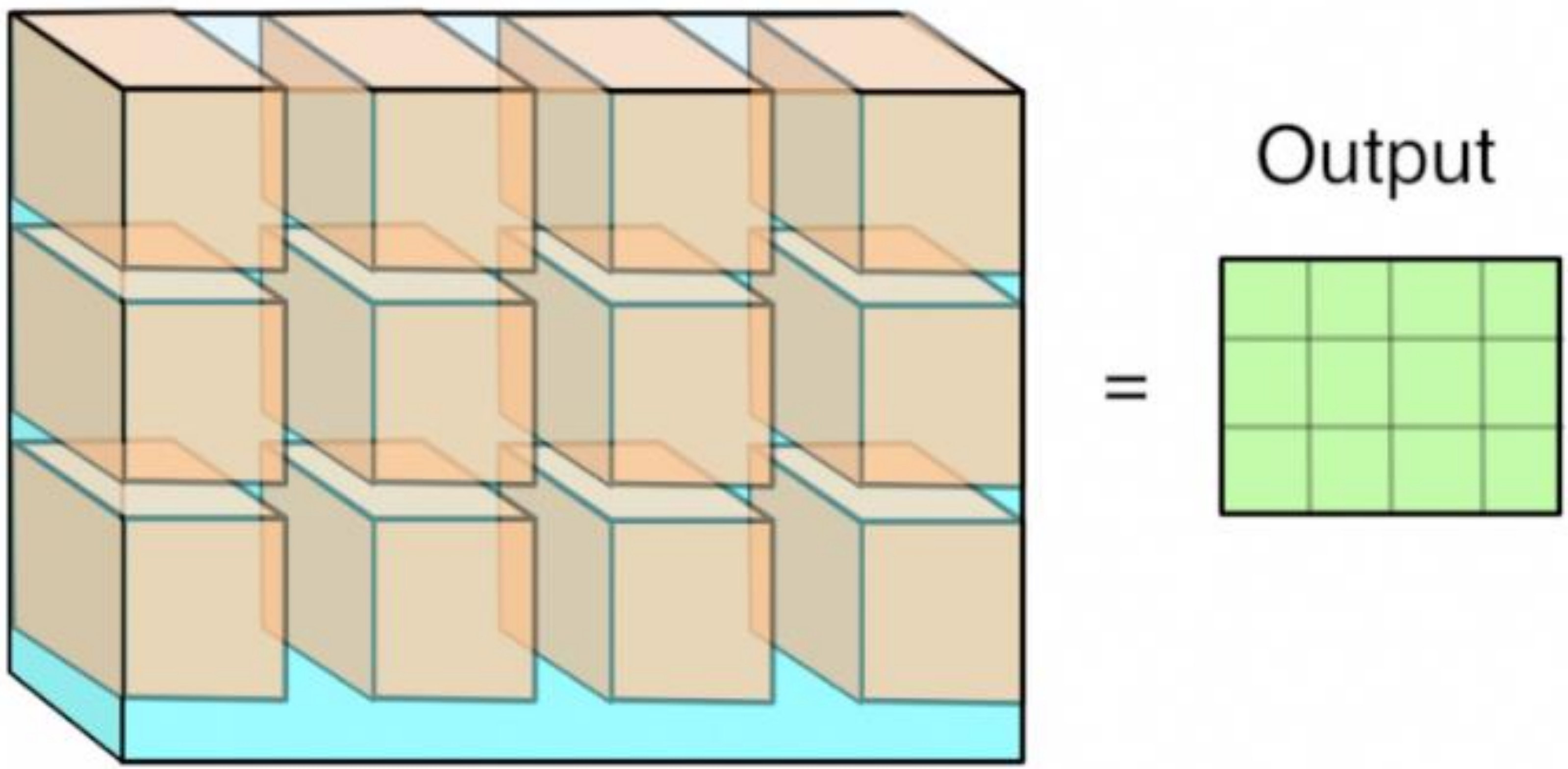
https://petewarden.com/2015/04/20/why-gemm-is-at-the-heart-of-deep-learning/

CNN layer Implementation



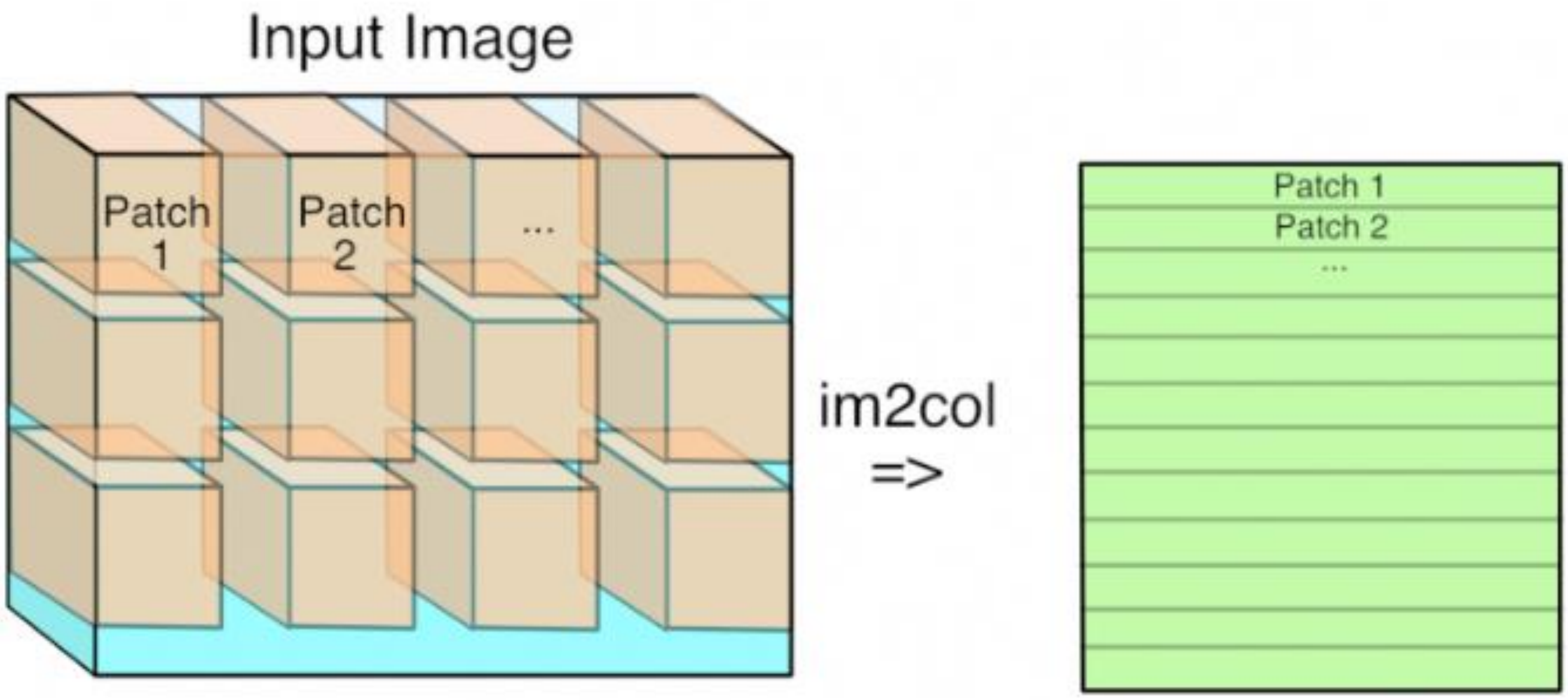
https://petewarden.com/2015/04/20/why-gemm-is-at-the-heart-of-deep-learning/

Im2col



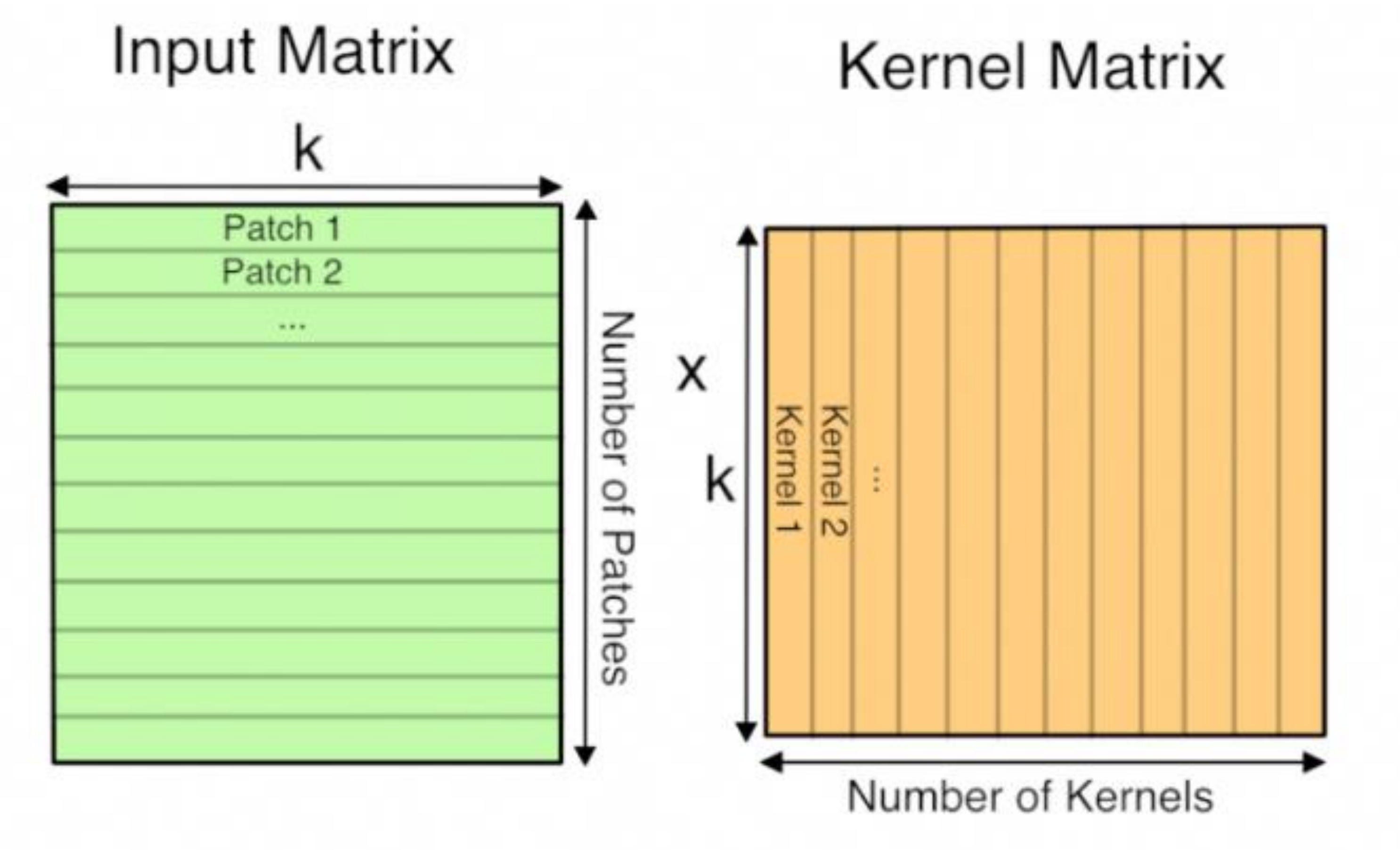
https://petewarden.com/2015/04/20/why-gemm-is-at-the-heart-of-deep-learning/

Im2col



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

CNN layer Implementation



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

GEMM-framework

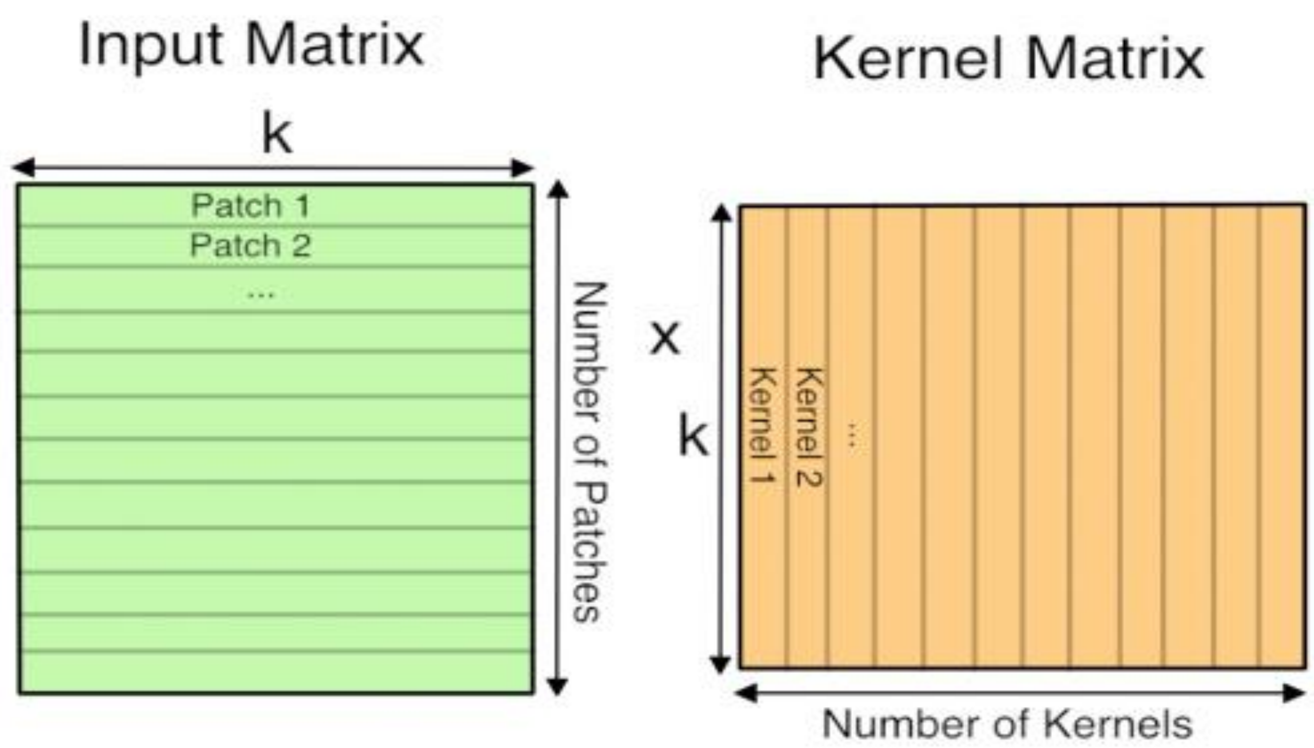


Figure Credit: https://petewarden.com/2015/04/20/why-gemm-is-at-the-heart-of-deep-learning/

Computation parameters

- Number of parameters in a CONV layer without bias:

$$\mathbf{(m * n)*k)}$$

- Number of parameters in a CONV layer with bias:

$$\mathbf{((m * n)+1)*k)}$$

added 1 because of the bias term for each filter.

- Here:

m: shape of width of the filter

n : shape of height of the filter

k : number of filters

Summary

- Fully connected to convolutional networks
- Convolution and sub-sampling operations
- Stride size and padding for convolution
- Convolutional Neural Networks
- LeNet architecture and AlexNet layers

- What's next?
  - How to implement better model?
    - Initialization methods
    - Activation function
    - Batch normalization
    - Regularization and
    - Optimization methods

Dumoulin, Vincent, and Francesco Visin. "A guide to convolution arithmetic for deep learning." *arXiv preprint arXiv:1603.07285* (2016).