CS 4803 / 7643: Deep Learning

Topics:

- Convolutional Neural Networks
  - Stride, padding
  - Pooling layers
  - Fully-connected layers as convolutions

Dhruv Batra

Georgia Tech

Administrativia

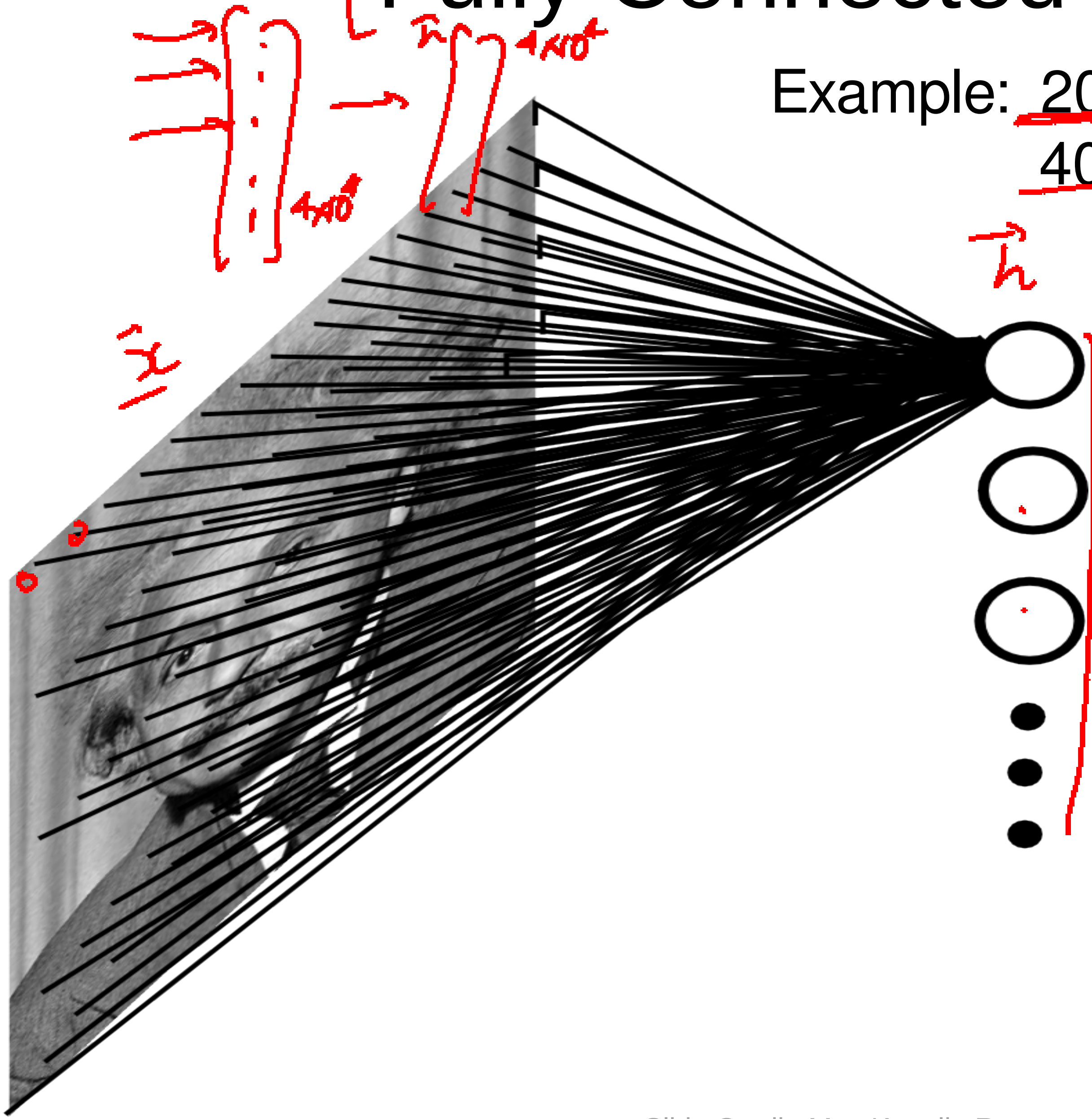
- HW2 Reminder
  - Due: 09/23, 11:59pm
  - https://evalai.cloudcv.org/web/challenges/challenge-page/684/leaderboard/1853
- Project Teams
  - https://gtvault-my.sharepoint.com/:x:/g/personal/dbatra8_gatech_edu/EY4_65XOzWtOkXSSz2WgpoUBY8ux2gY9PsRzR6KngIIFEQ?e=4tnKWI
  - Project Title
  - 1-3 sentence project summary TL;DR
  - Team member names

Recap from last time

Convolutional Neural Networks

(without the brain stuff)

Fully Connected Layer

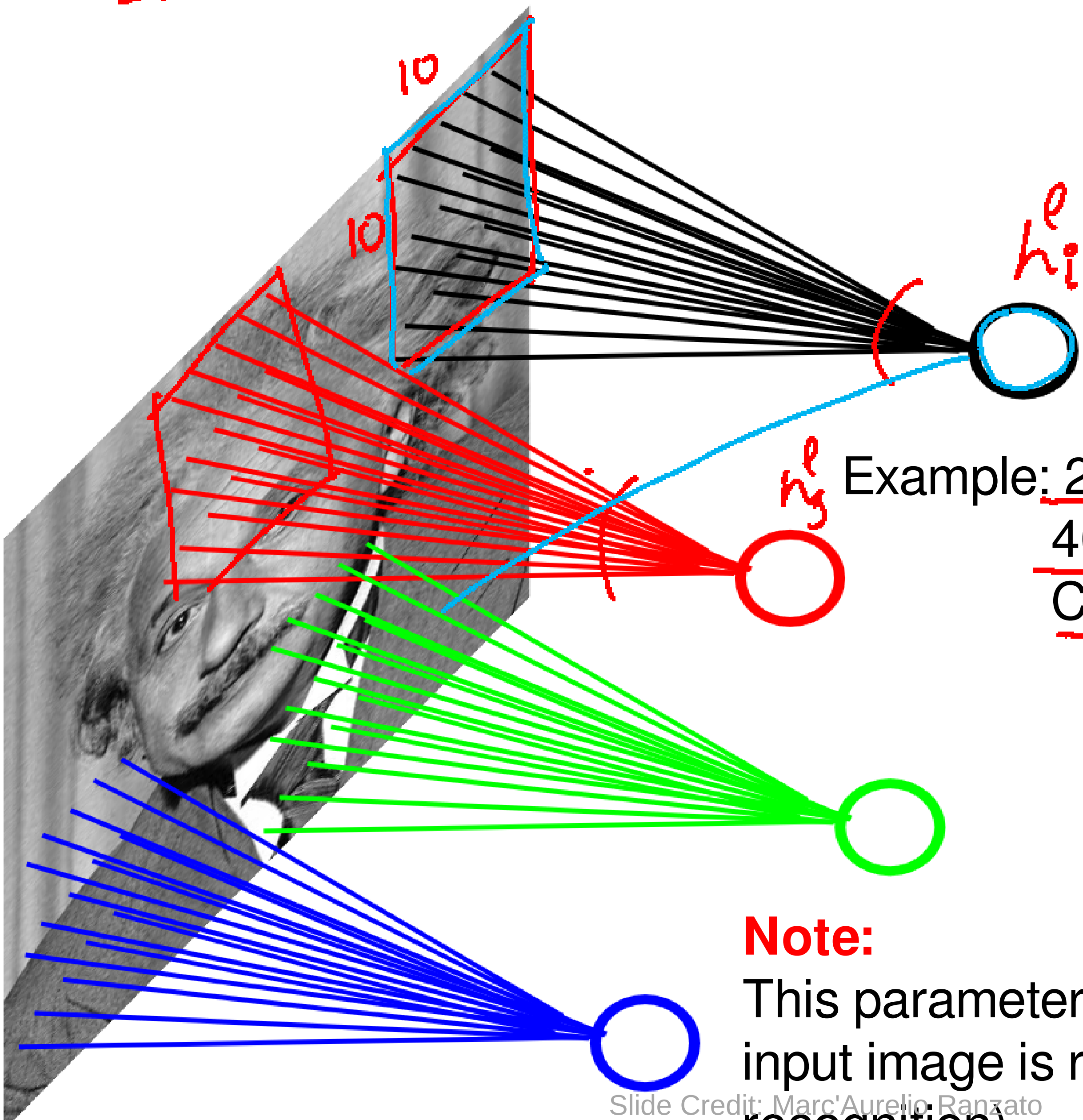


Example: 200x200 image
40K hidden units

Q: what is the number of parameters in this FC layer?
A: 1.6B

1.6×10^9
40k
x40k

Assumption 1: Locally Connected Layer



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

40k x 100

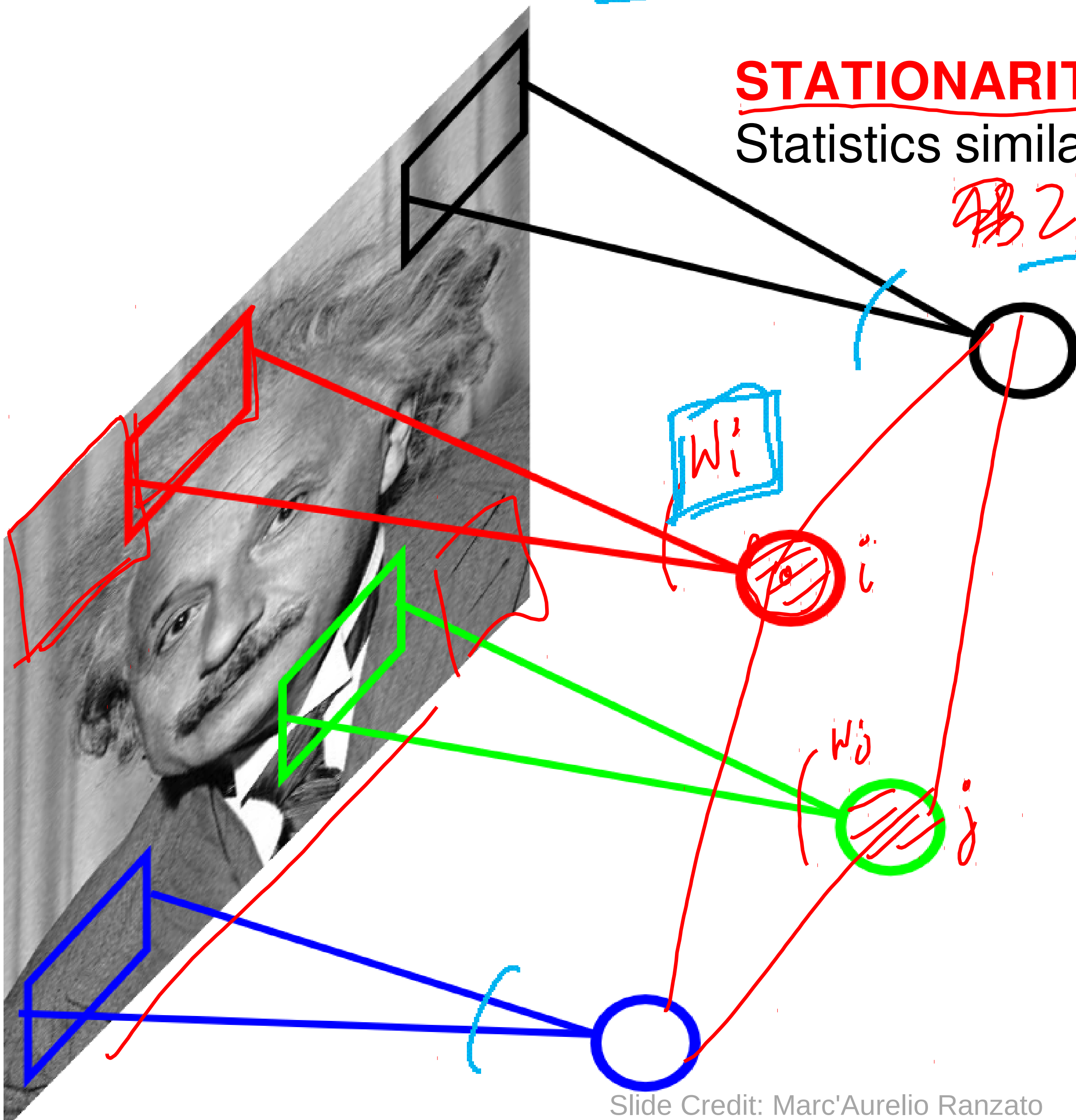
Note:
This parameterization is good when input image is registered (e.g., face recognition).

Assumption 2: Stationarity / Parameter Sharing

STATIONARITY?

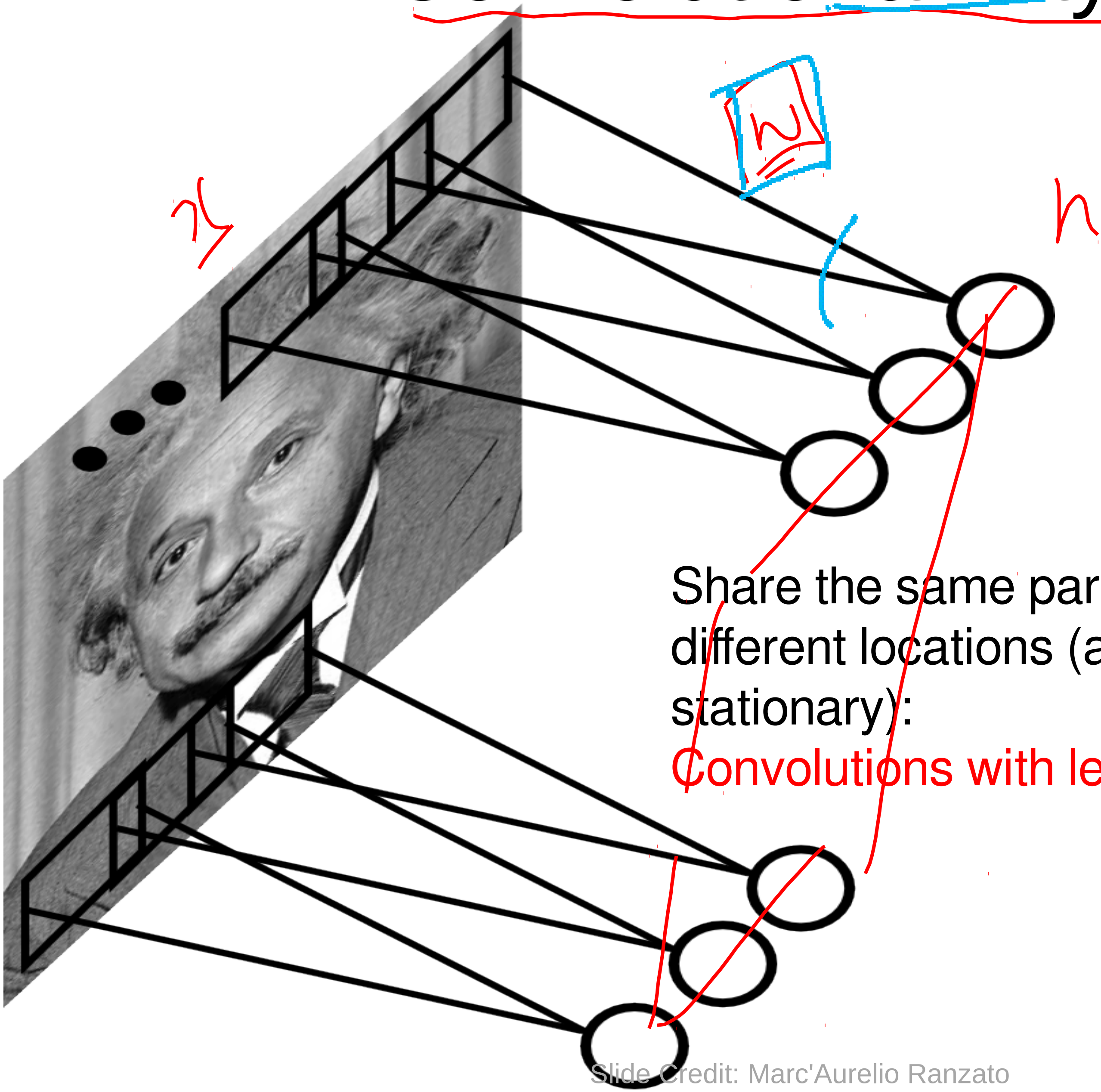
Statistics similar at all locations

2B → 4M → 100



$W_i = W_j$

Convolutional Layer



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

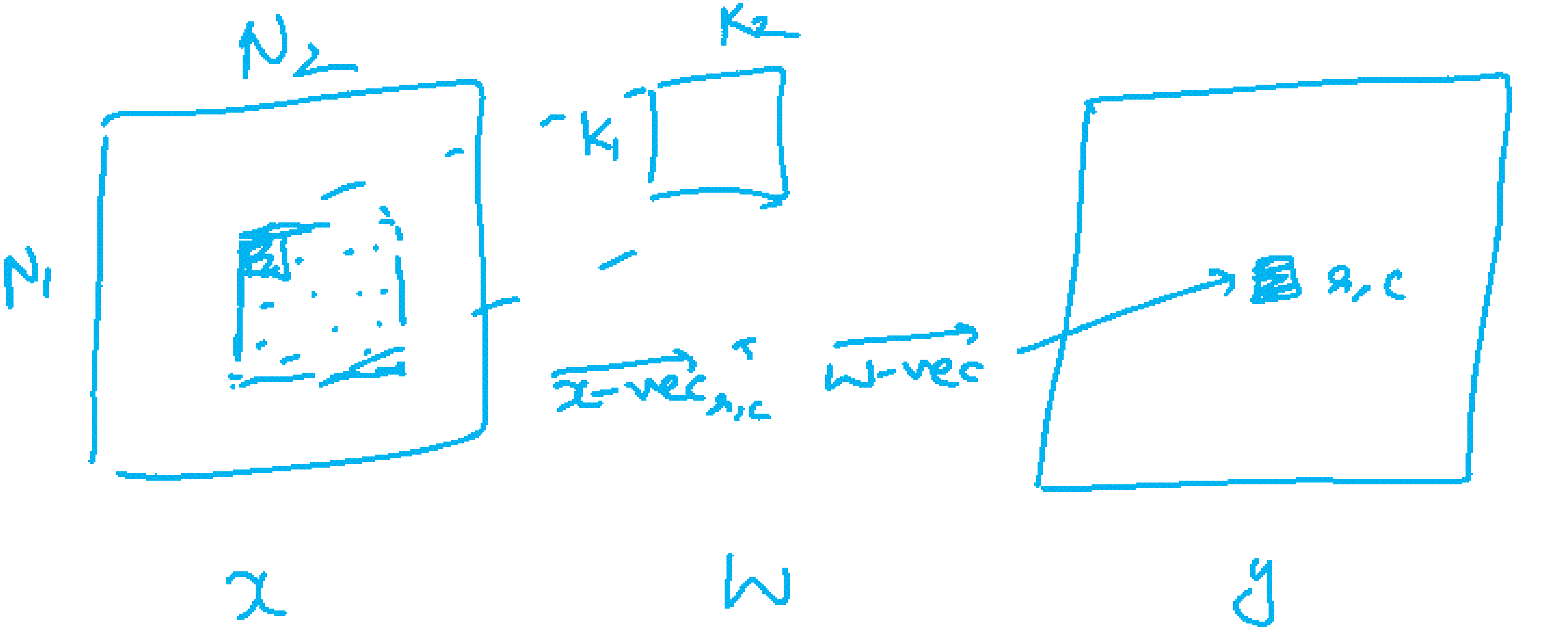
Convolutions!

math → CS → programming

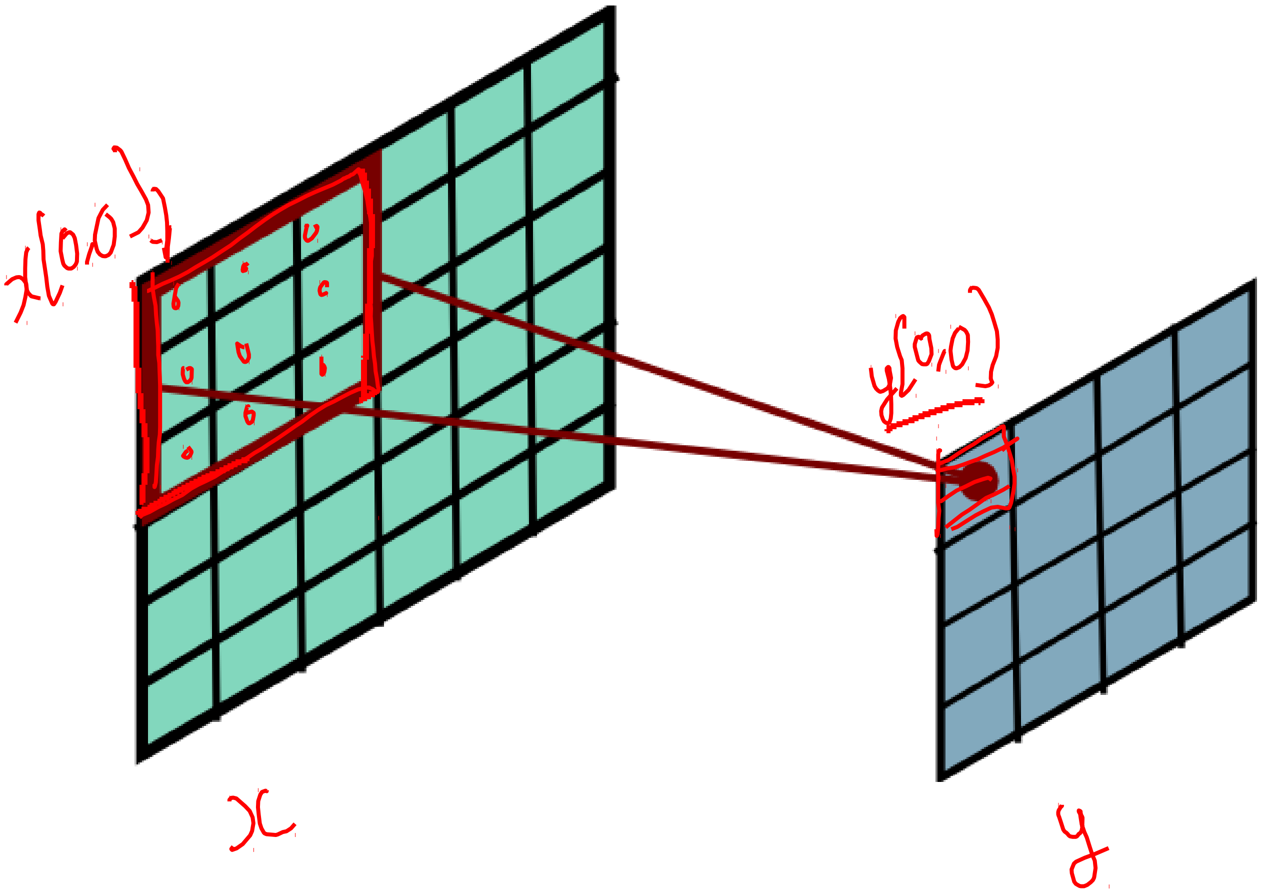
Convolutions for programmers

$$y[r, c] = \sum_{a=0}^{k_1-1} \sum_{b=0}^{k_2-1} x[r+a, c+b]\, w[a, b]$$

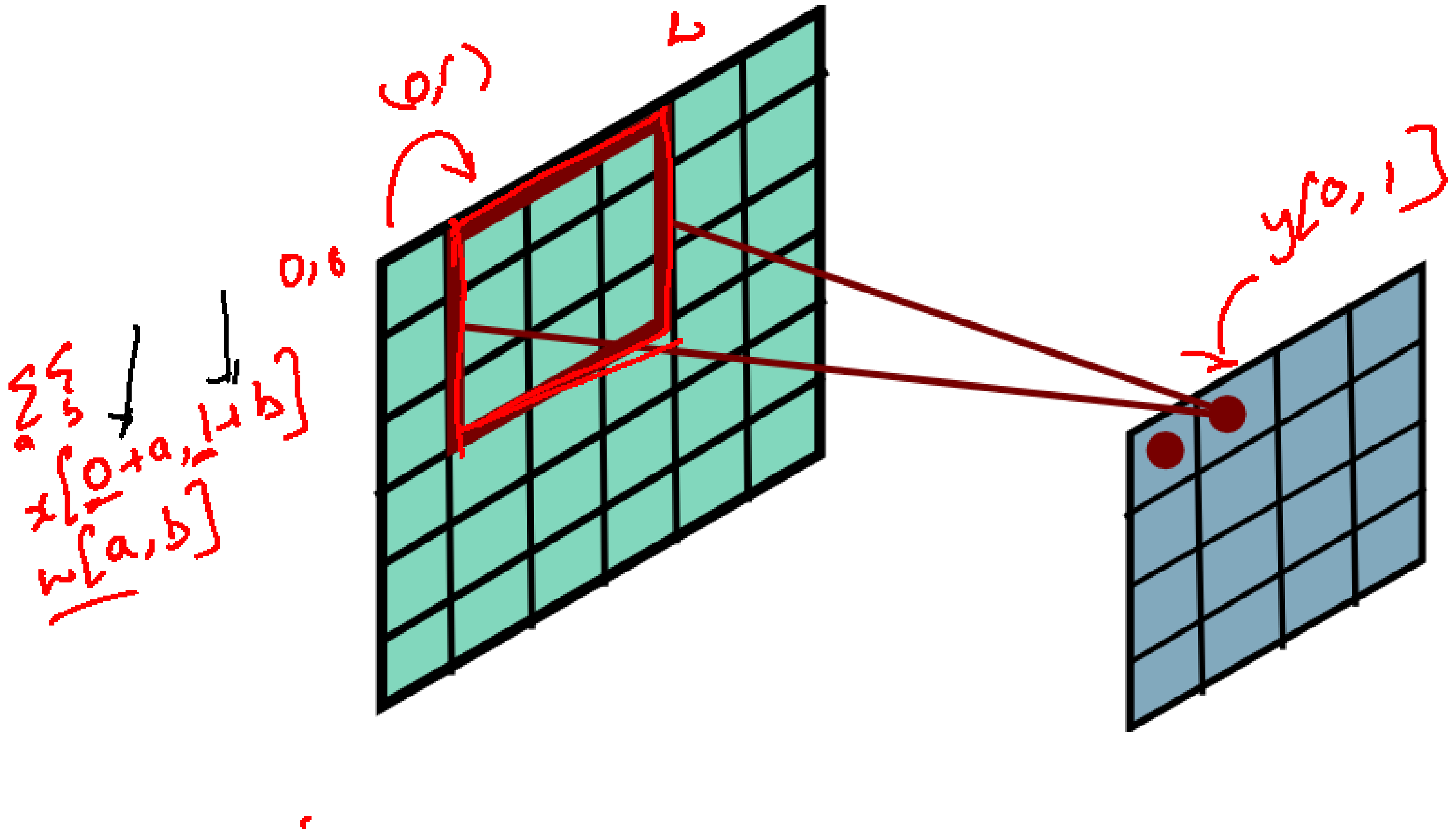
rows cols



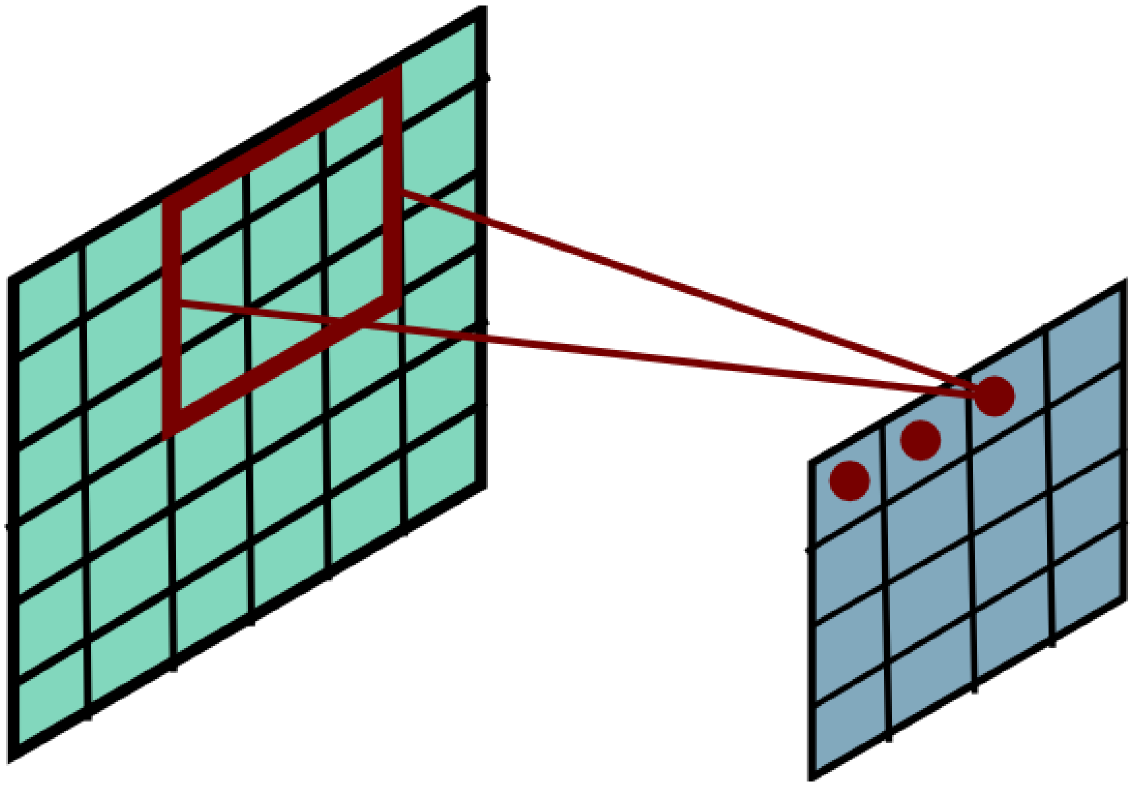
Convolution



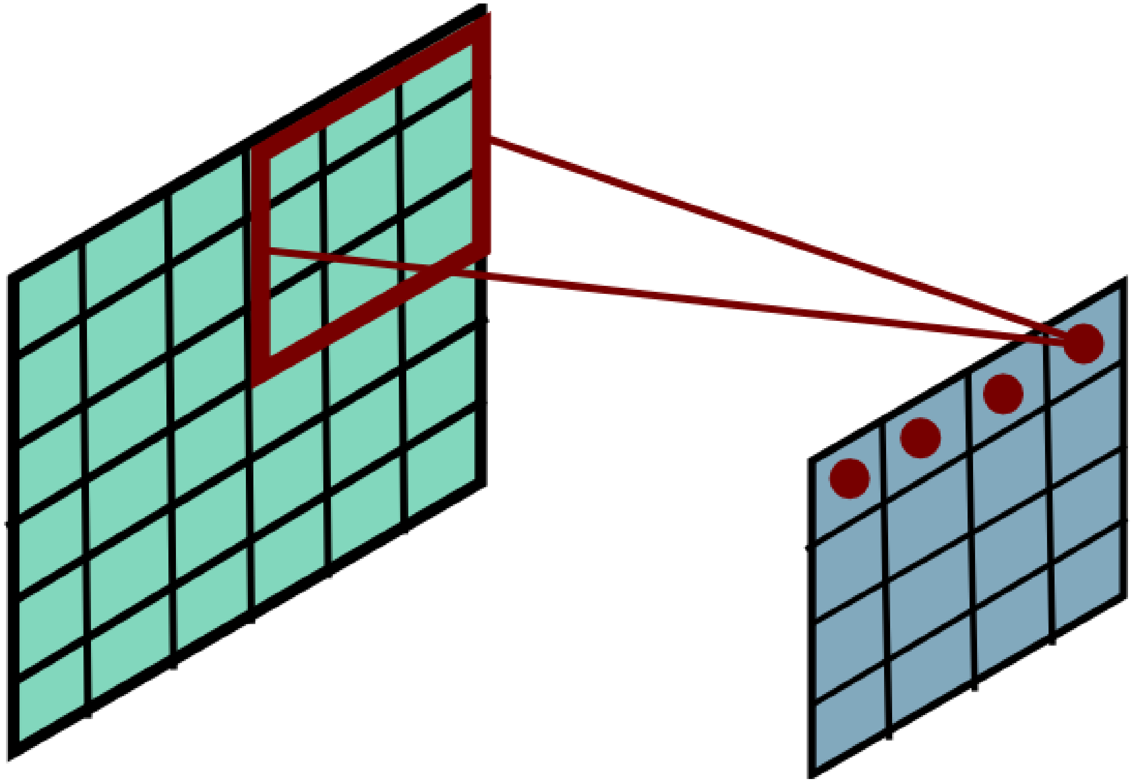
Convolutional Layer



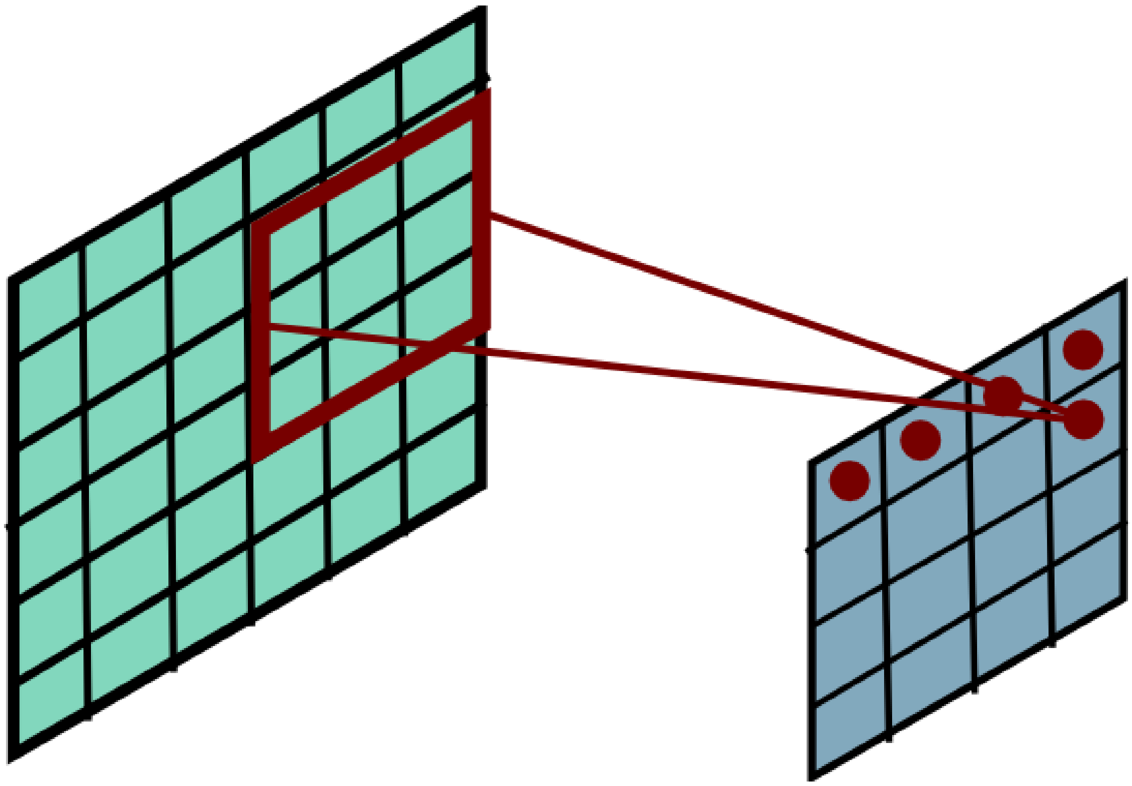
Convolution



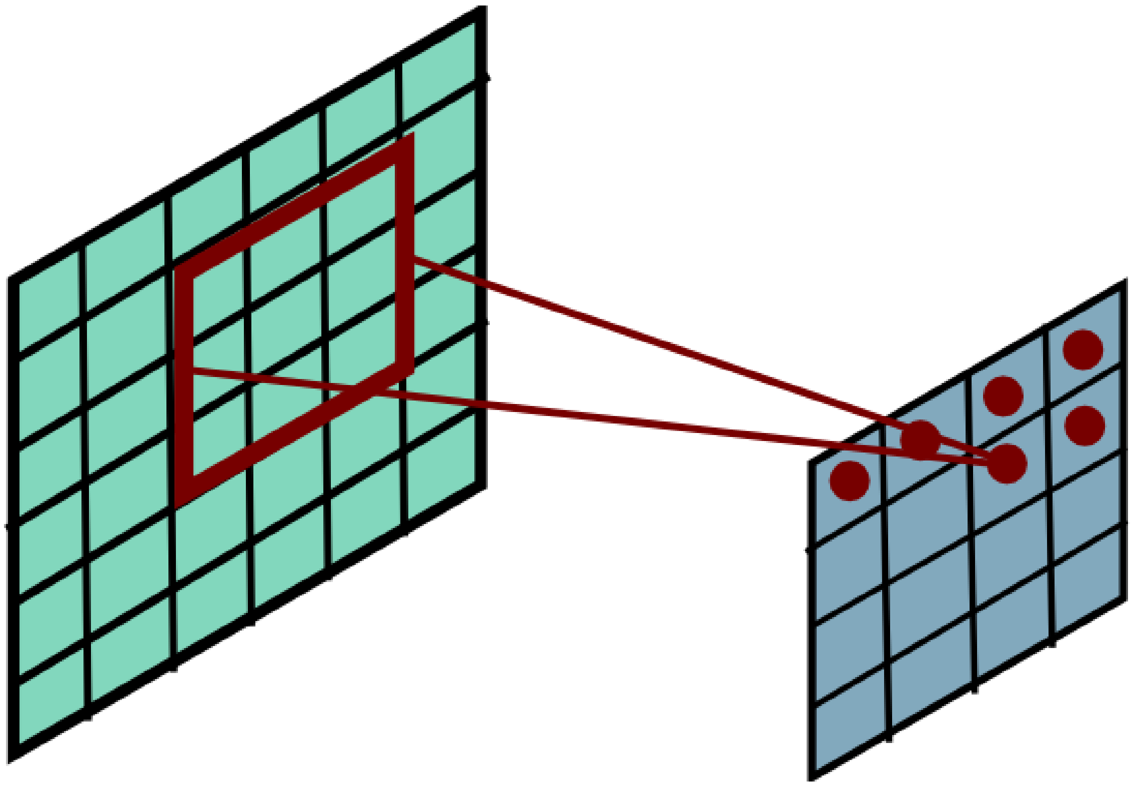
Convolution



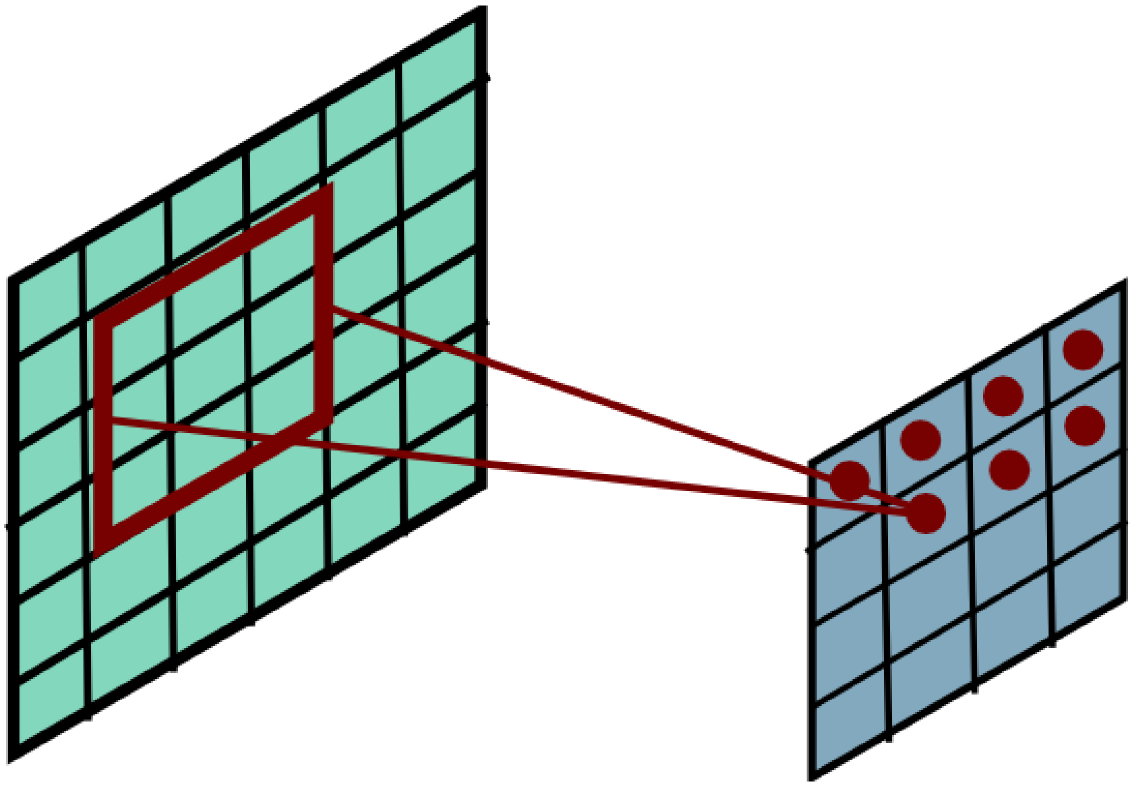
Convolution



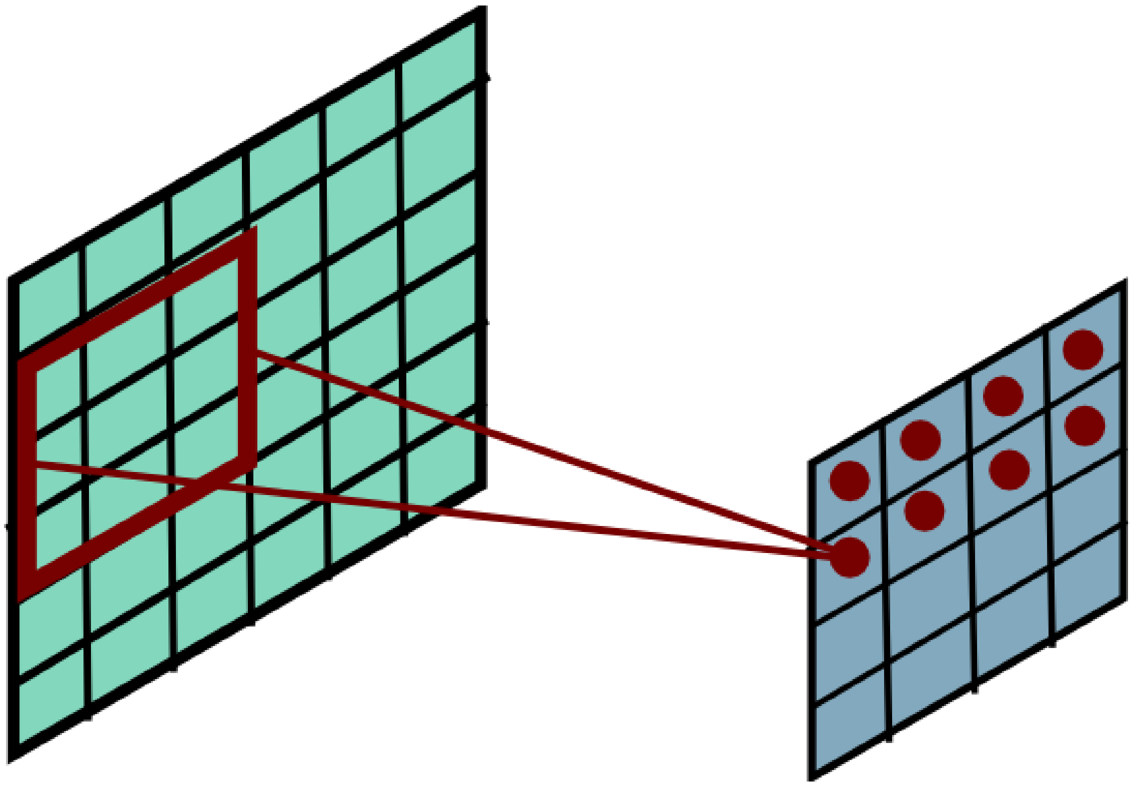
Convolution



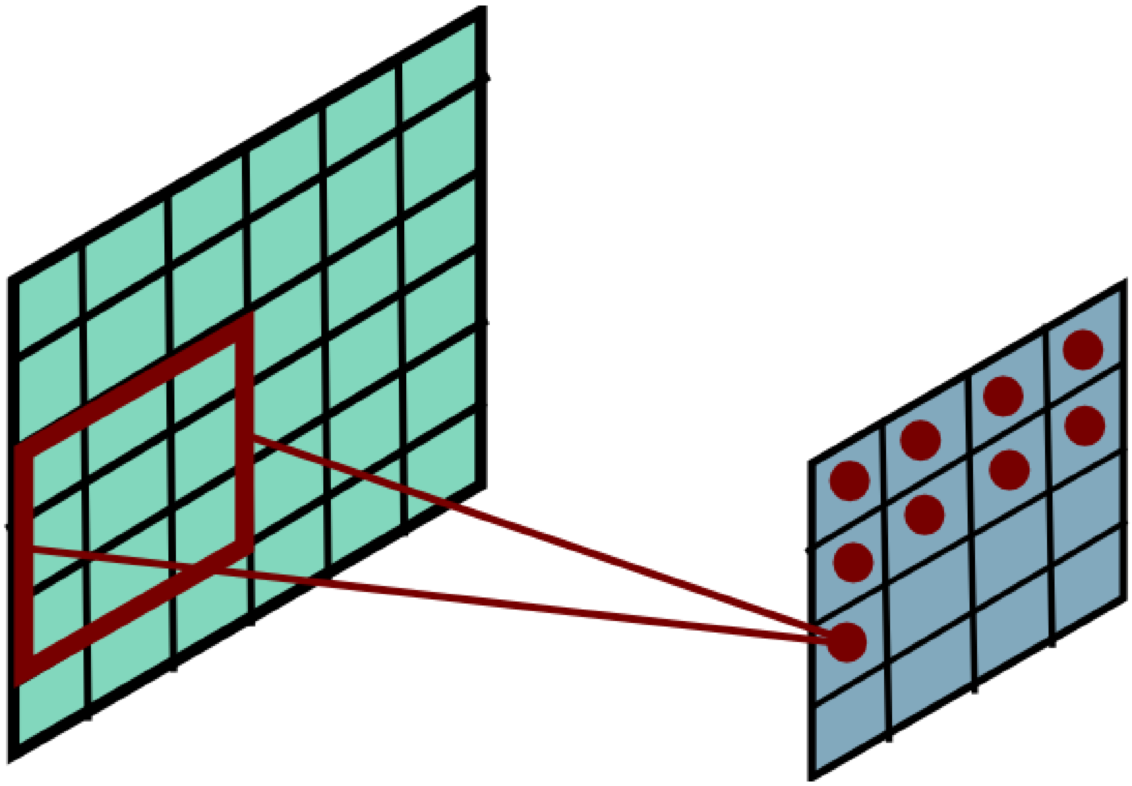
Convolution



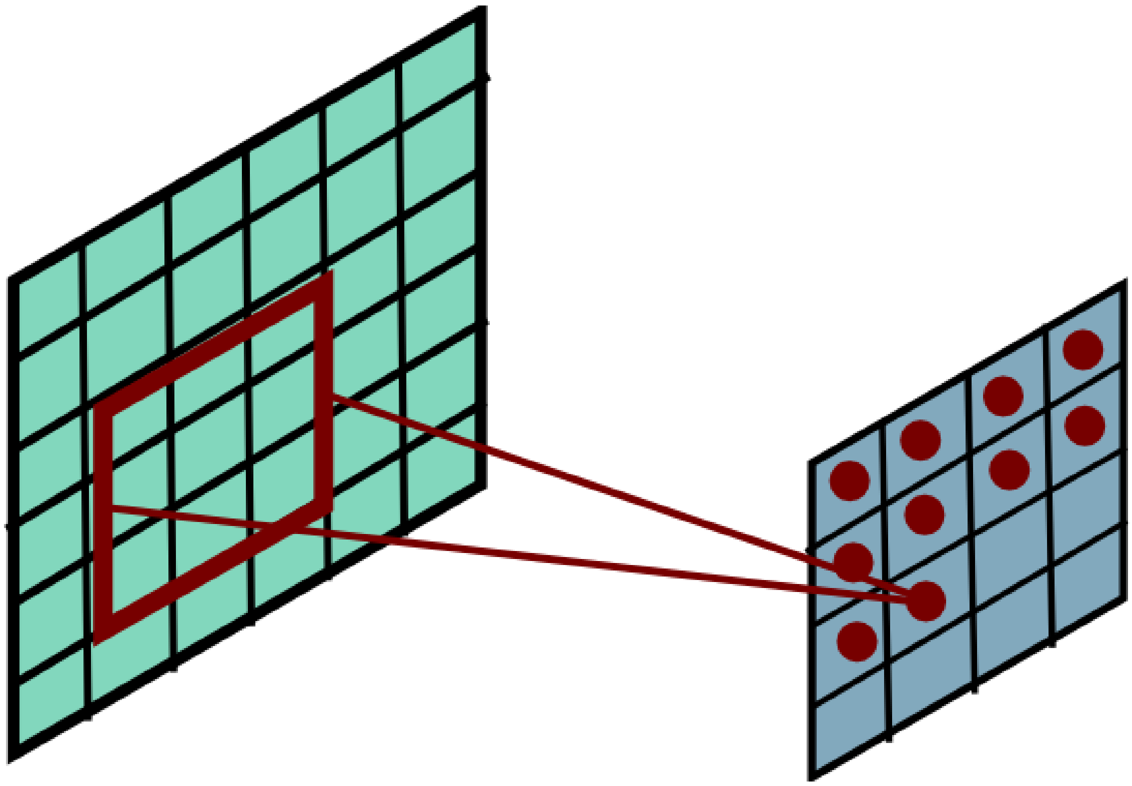
Convolution



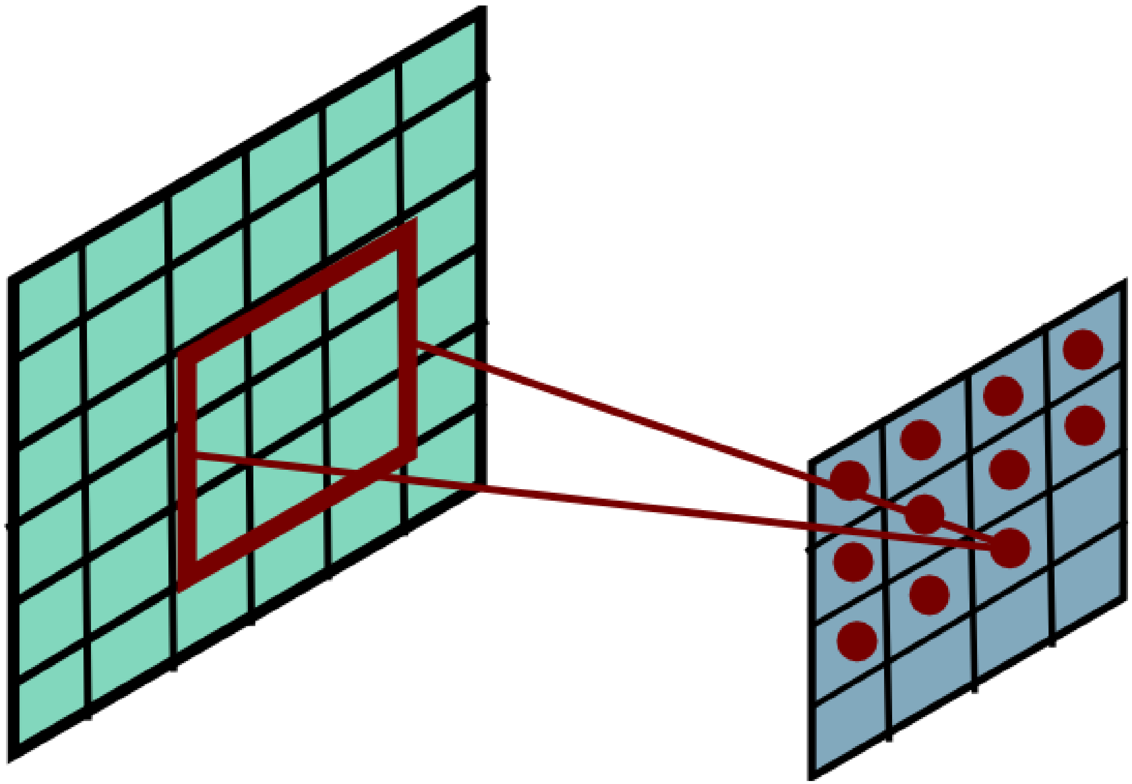
Convolution



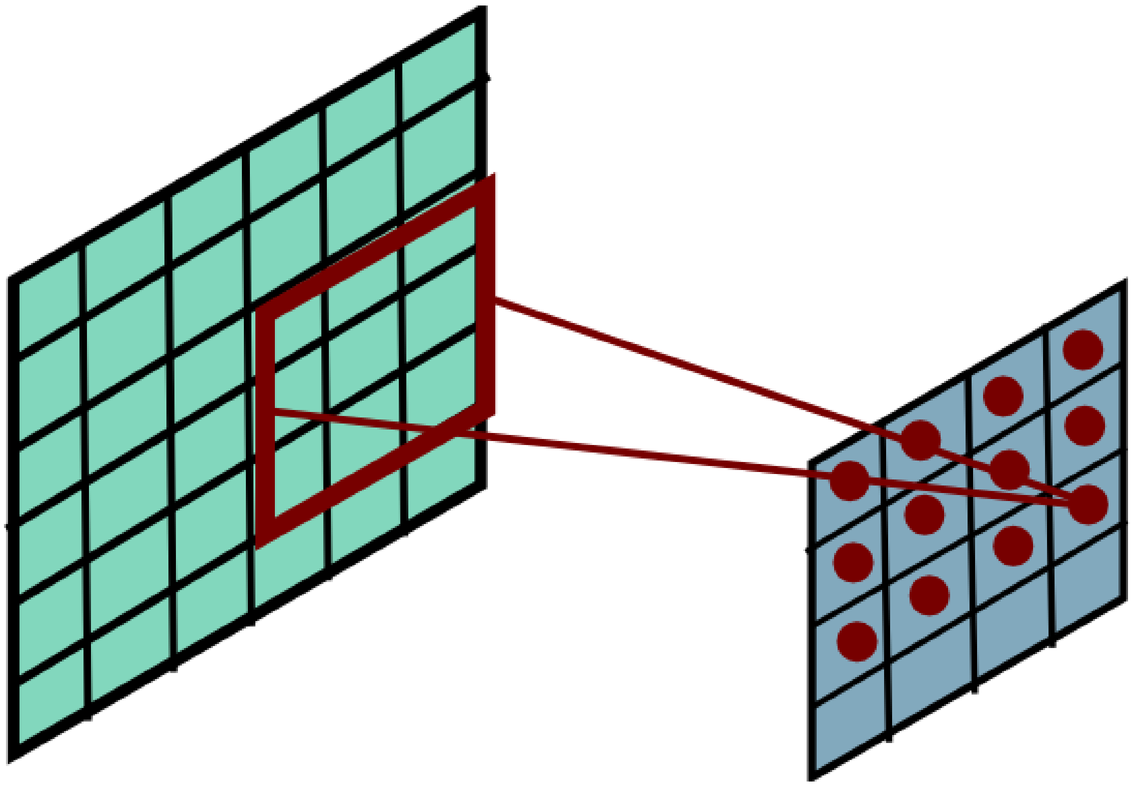
Convolution



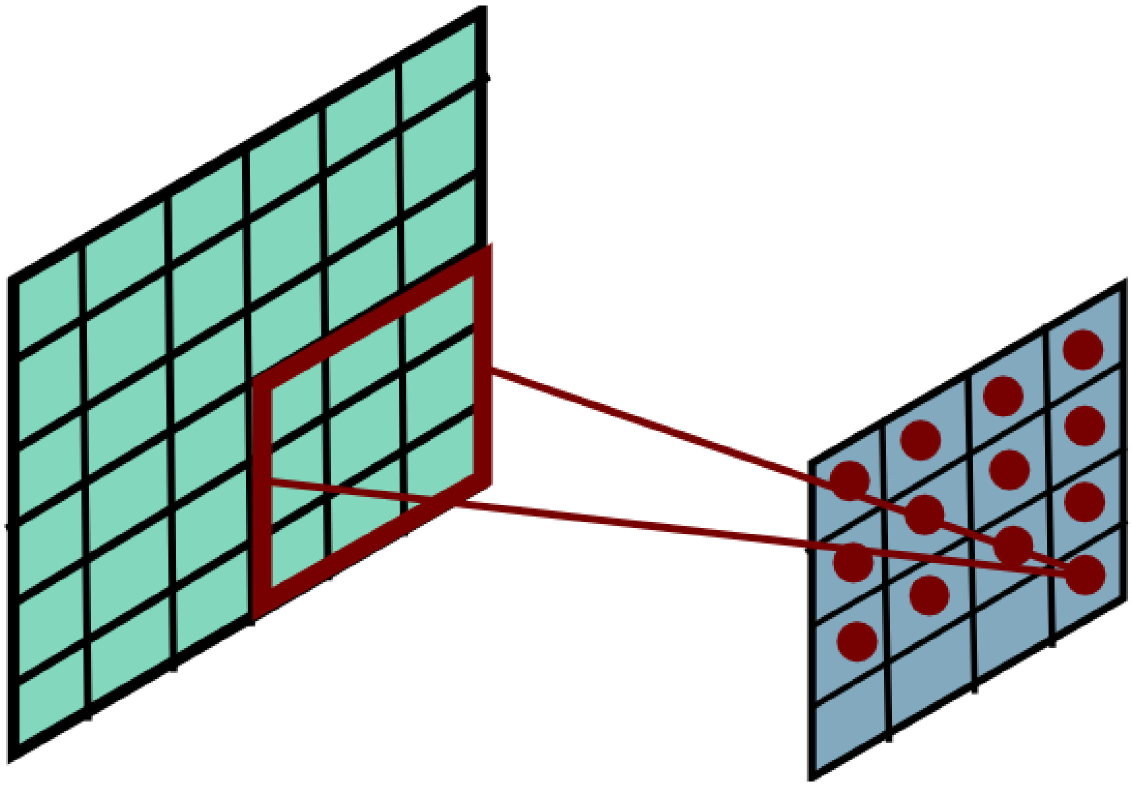
Convolution



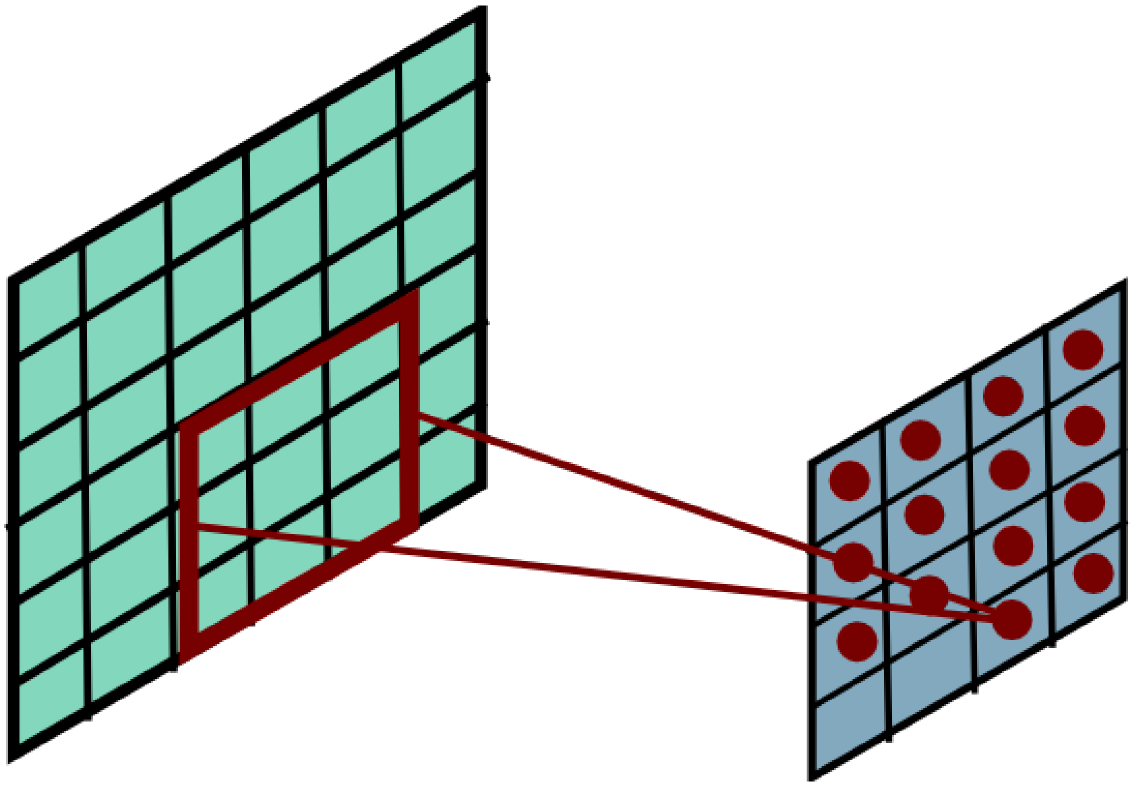
Convolution



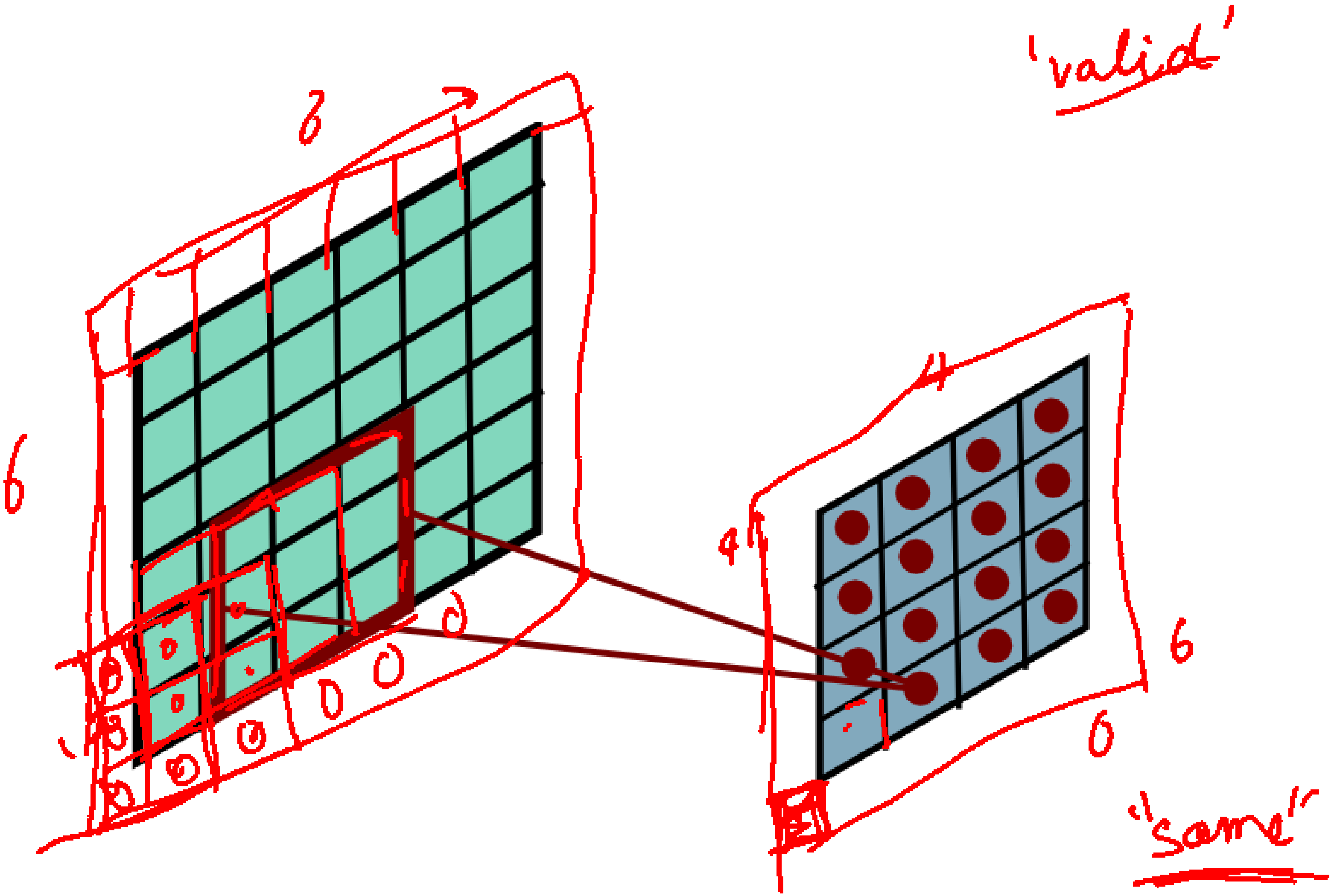
Convolution



Convolution



Convolution



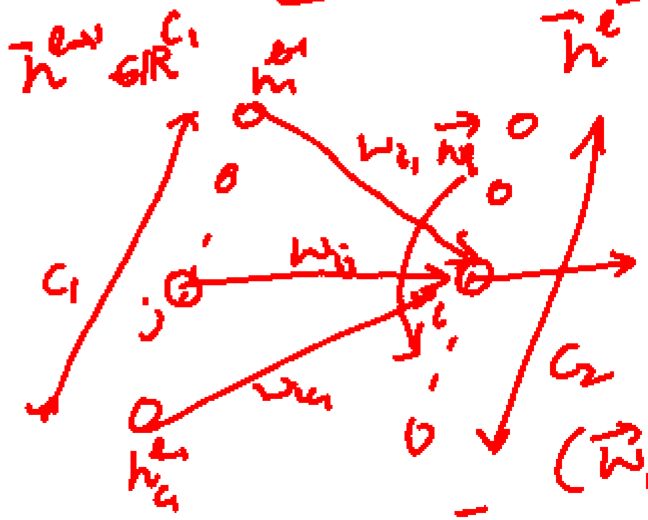
Slide Credit: Marc'Aurelio Ranzato

Plan for Today

- Convolutional Neural Networks
  - Features learned by CNN layers
  - Stride, padding
  - 1x1 convolutions
  - Pooling layers
  - Fully-connected layers as convolutions

FC vs Conv Layer

FC Layer

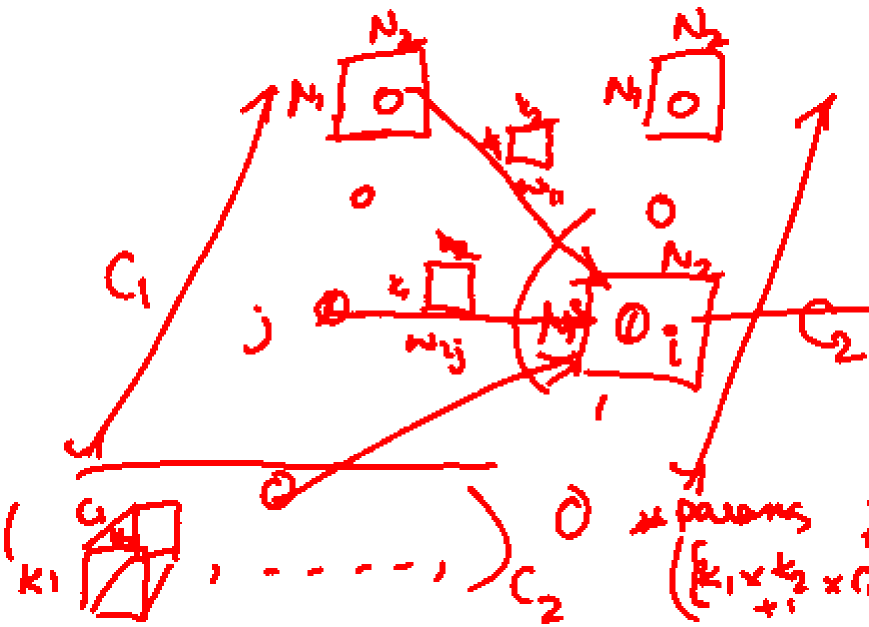


$$h_i^l = \sum_{j=1}^{C_1} h_j^{l-1} \cdot w_{ij}$$

Scalar prod

$(\vec{w}_1^T, \dots, \vec{w}_i^T, \dots \vec{w}_{C_2}^T)$ $\quad C_2 \times [C_1+1]$

Conv Layer



2D Conv

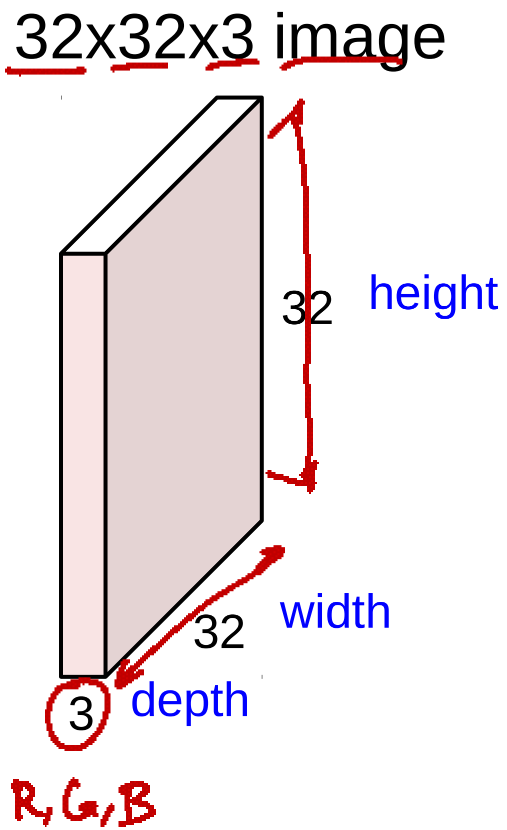
$$H_i^l = \sum_{j=1}^{C_1} H_j^{l-1} \boxed{*} w_{ij}$$

params $([k_1 \times k_2 \times C_1] + 1) \times C_2$

$$H_i^l[r,c] = \sum_{j=1}^{C_1} \sum_{a=0}^{k_1-1} \sum_{b=0}^{k_2-1} H_i^{l-1}[r+a, c+b]\, w_{ij}[a,b]$$

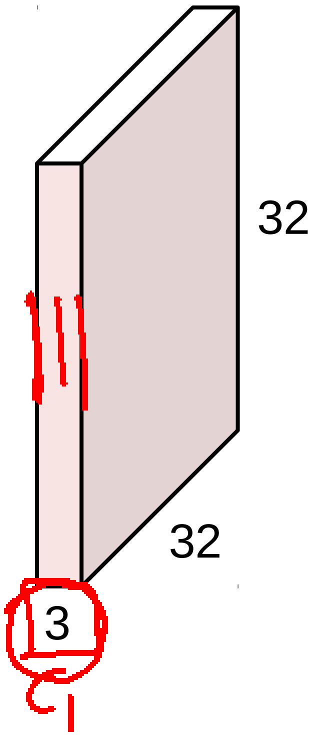
FC vs Conv Layer

Convolution Layer



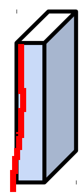
Convolution Layer

32x32x3 image



5x5x3 filter

C1



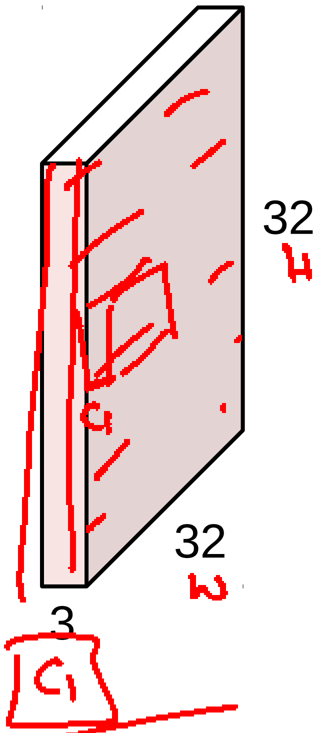
Convolve the filter with the image i.e. "slide over the image spatially, computing dot products"

Convolution Layer

C_1

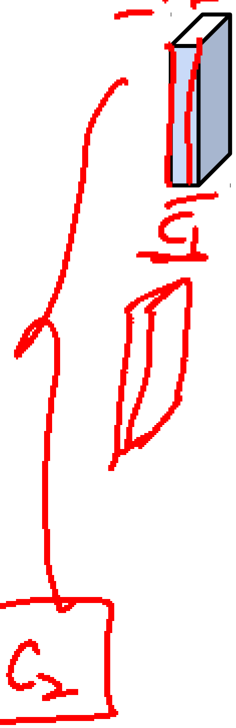
+10, 32, 64,

32x32x3 image



5x5x3 filter

$k_1 \times k_2 \times C_1$

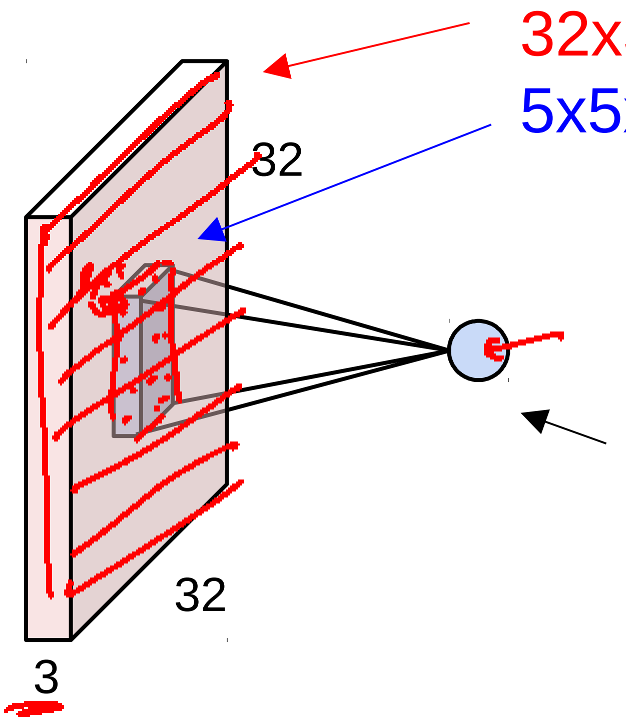


Filters always extend the full depth of the input volume

C_2

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

Convolution Layer

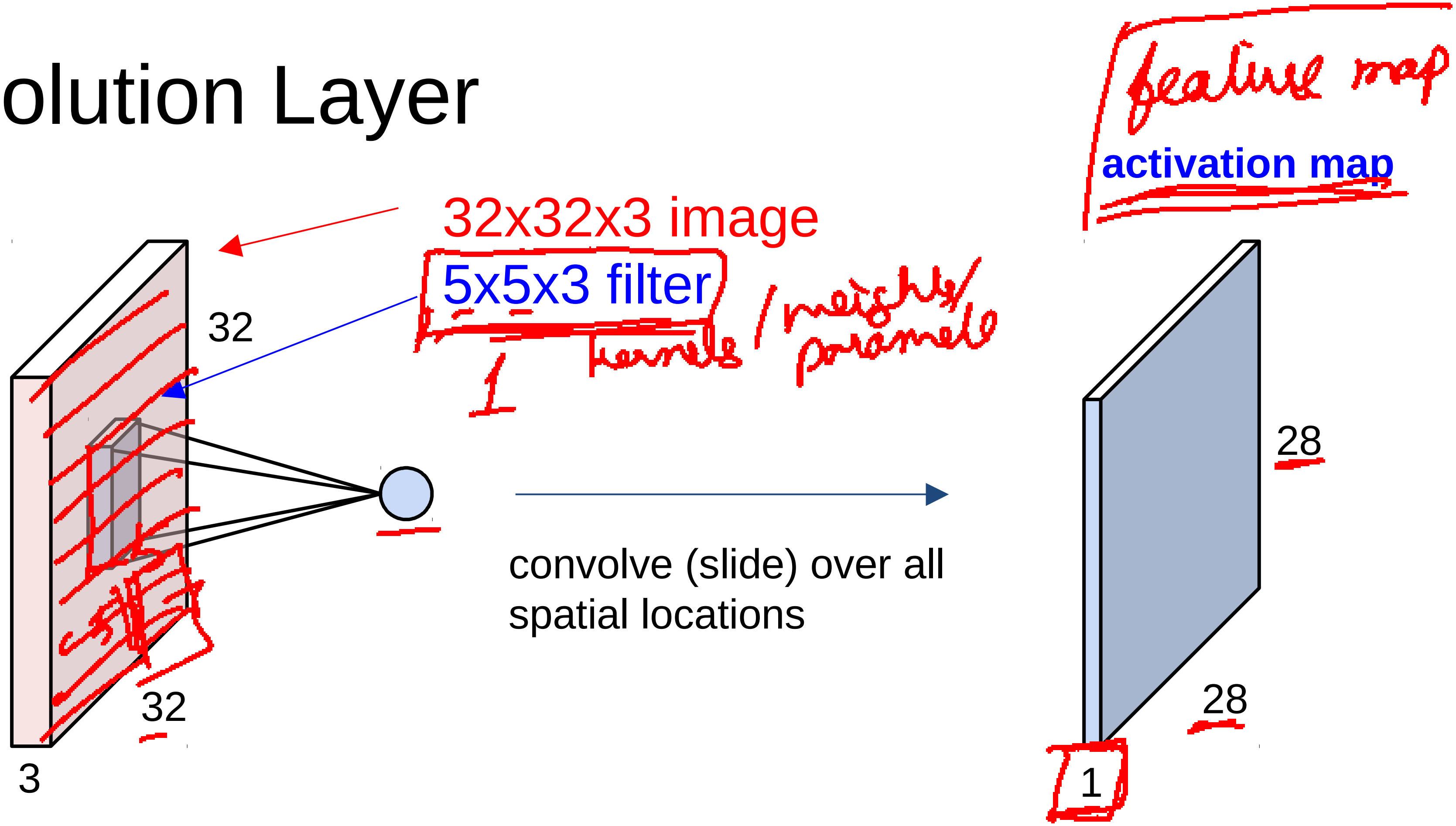


1 number:
the result of taking a dot product between the filter and a small 5x5x3 chunk of the image
(i.e. 5*5*3 = 75-dimensional dot product + bias)

$$w^T x + b$$

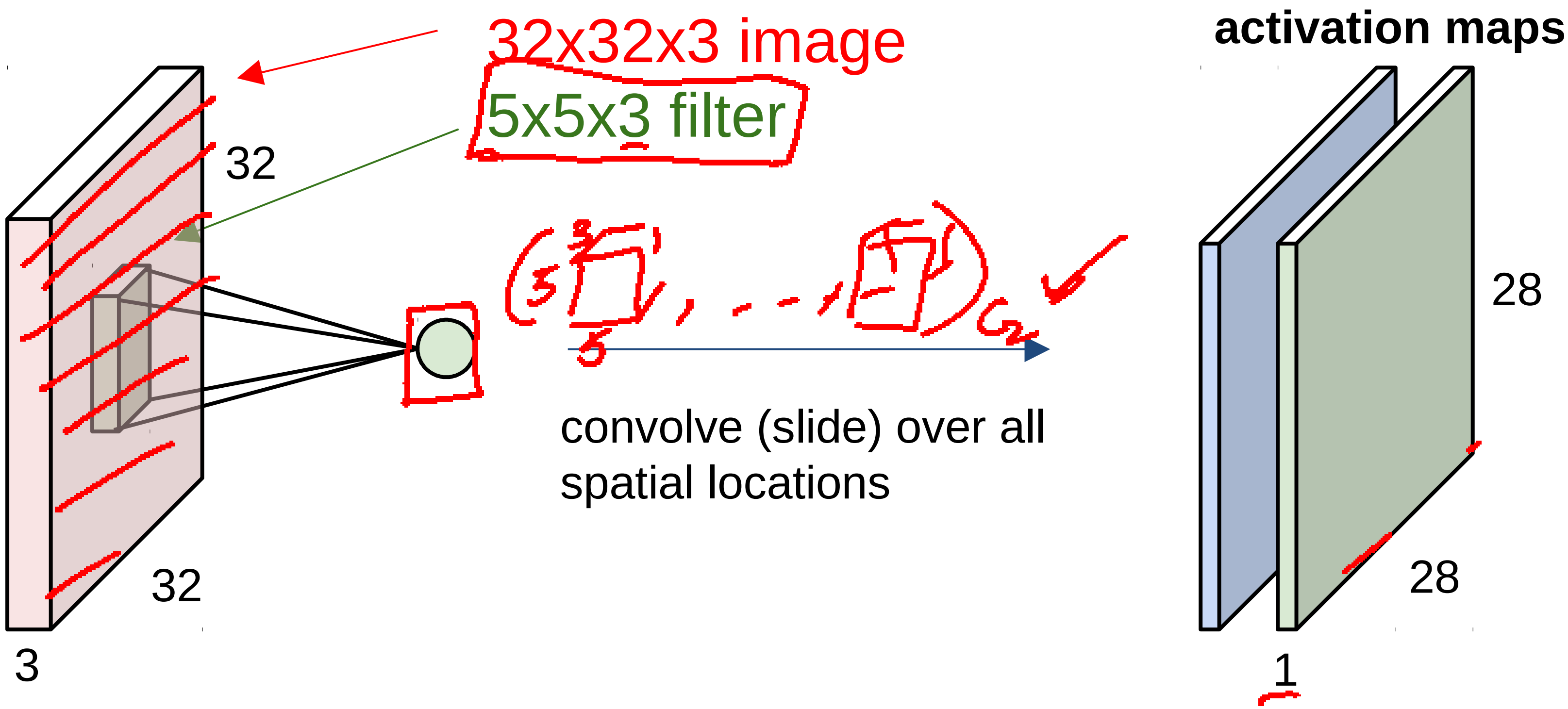
w-vec x-vec
x,c

Convolution Layer

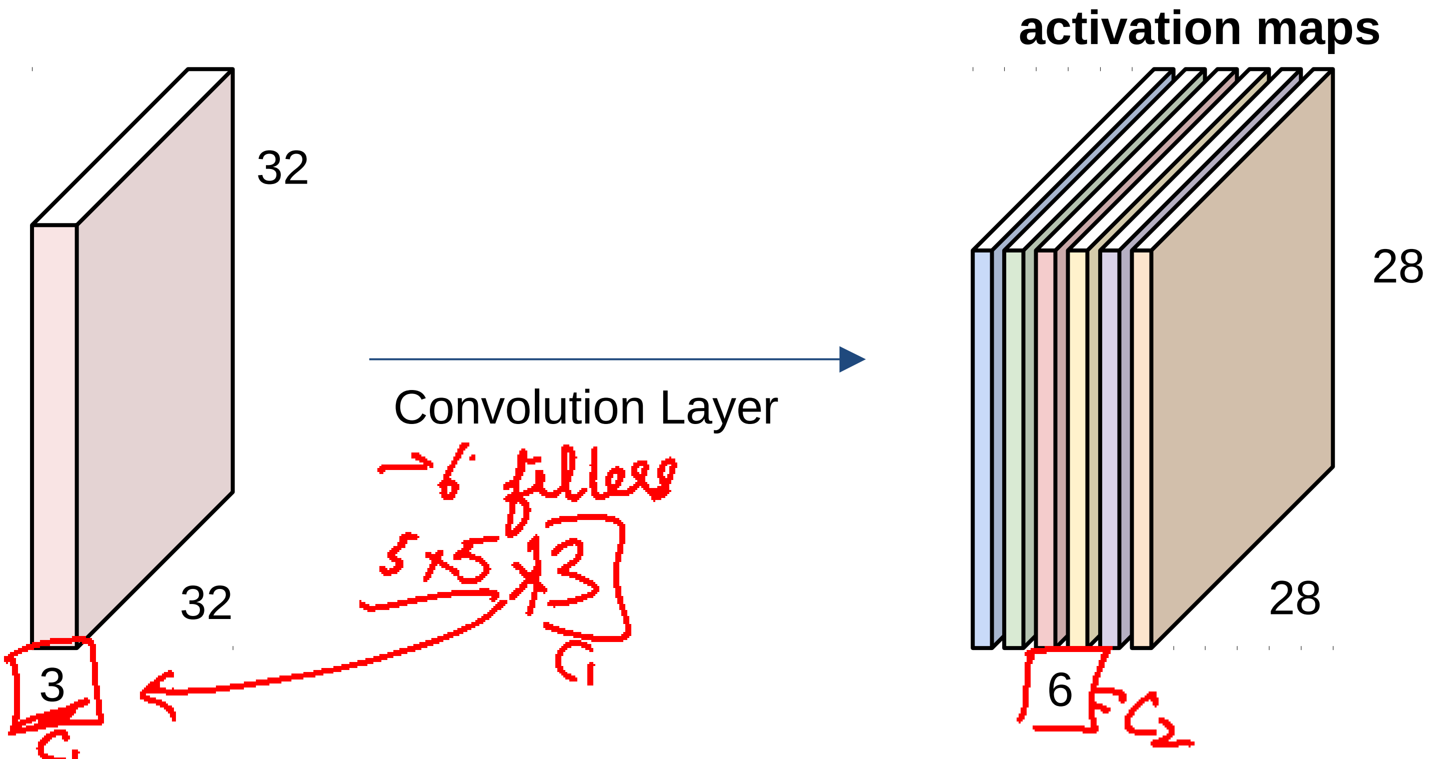


Convolution Layer

consider a second, green filter

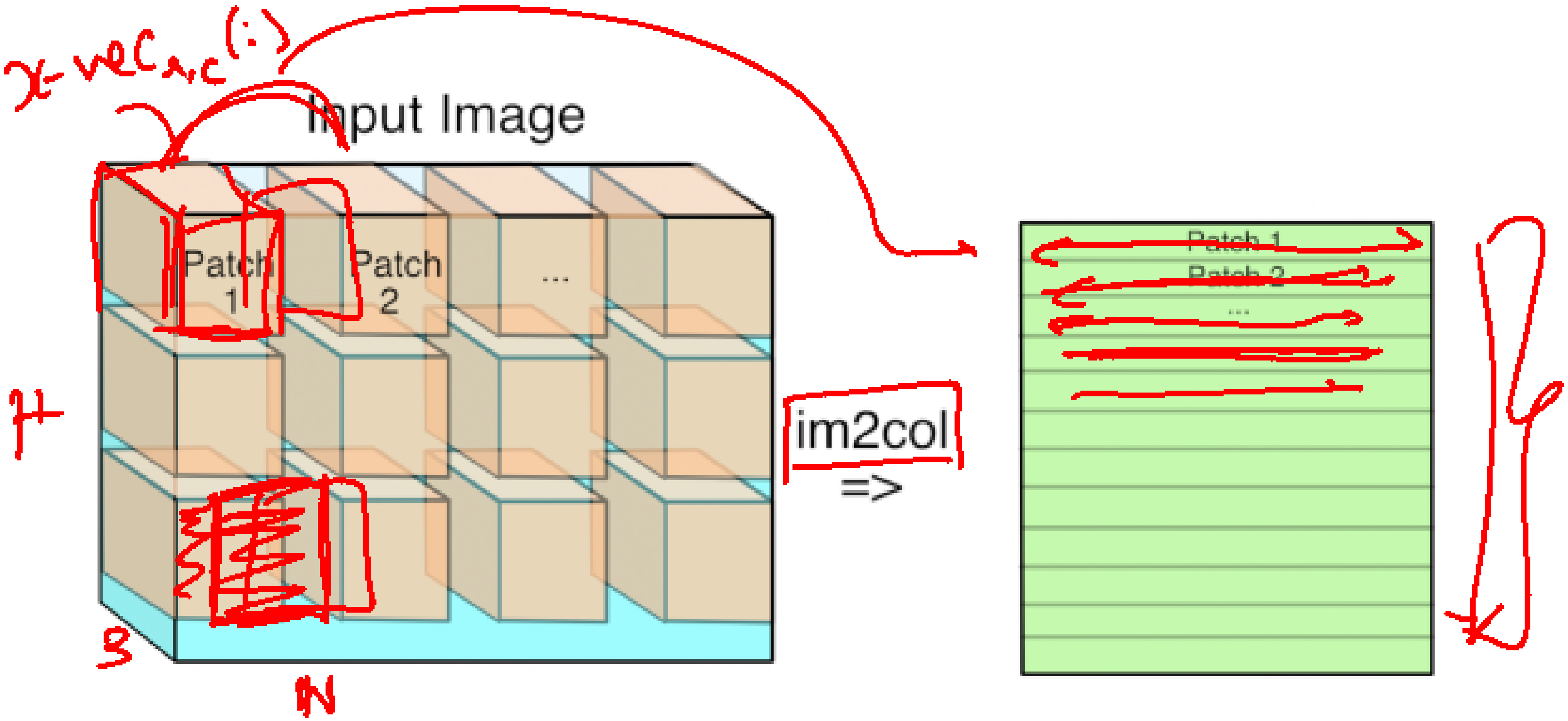


For example, if we had 6 5x5 filters, we'll get 6 separate activation maps:



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

Im2Col



GEMM

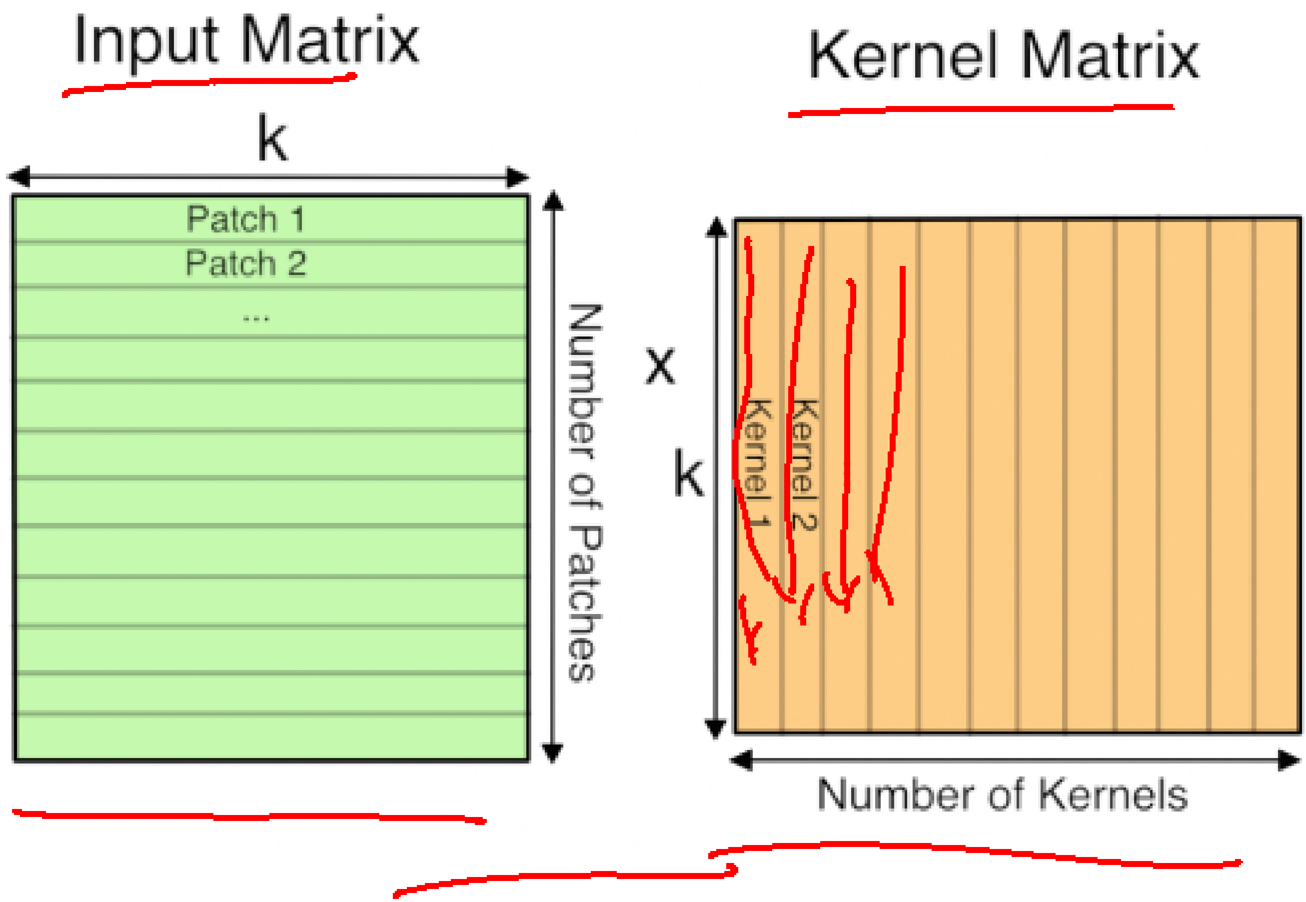
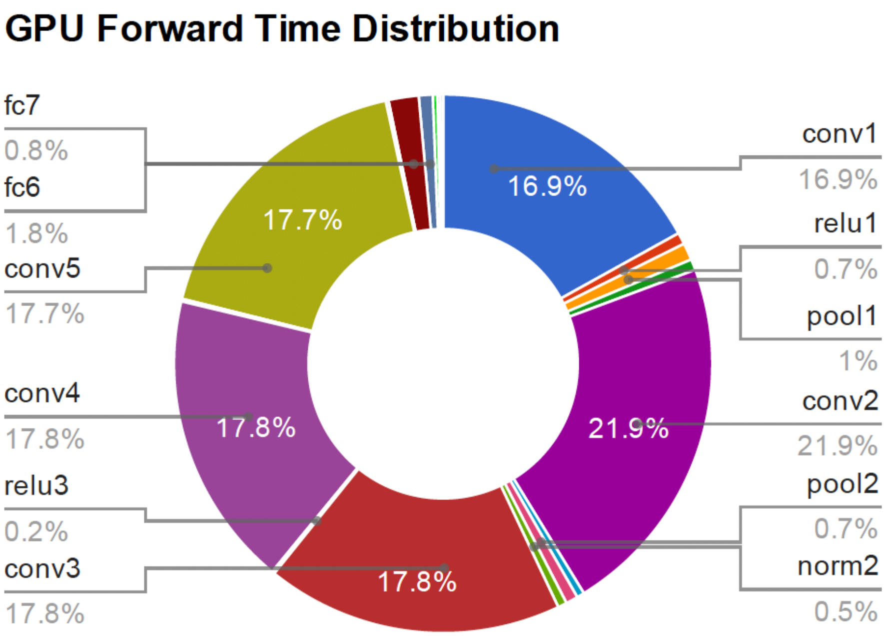


Figure Credit: https://petewarden.com/2015/04/20/why-gemm-is-at-the-heart-of-deep-learning/

Time Distribution of AlexNet



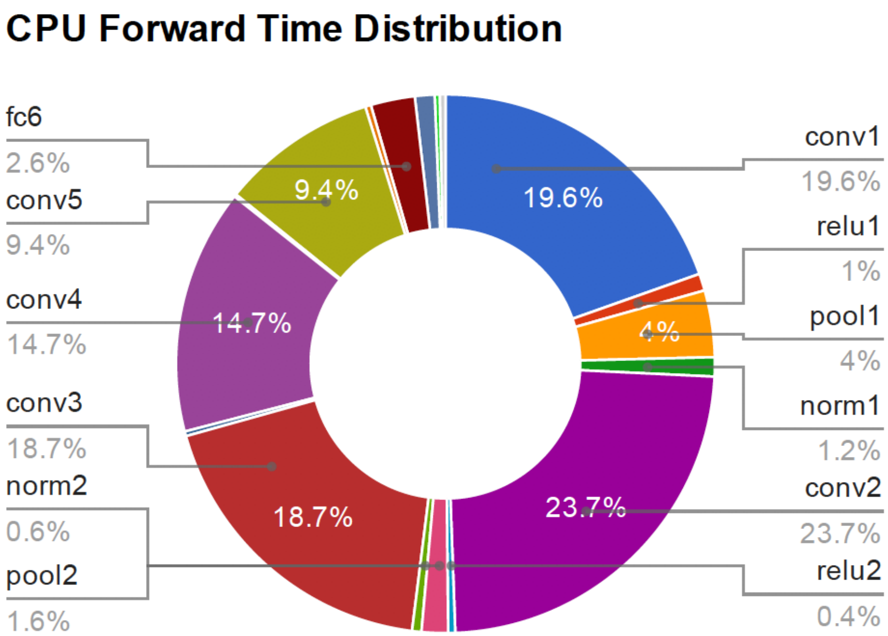
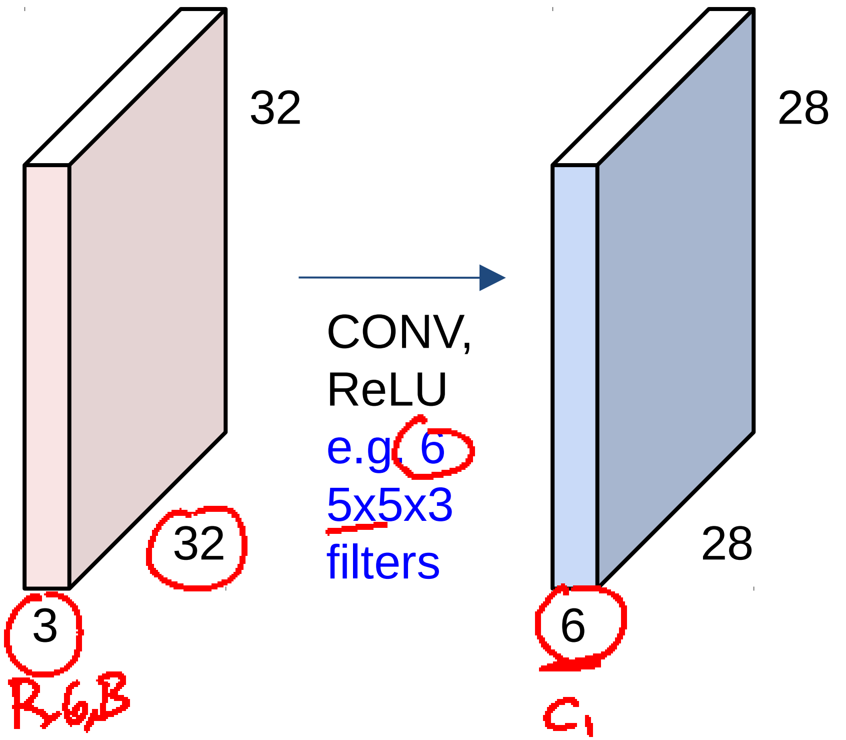
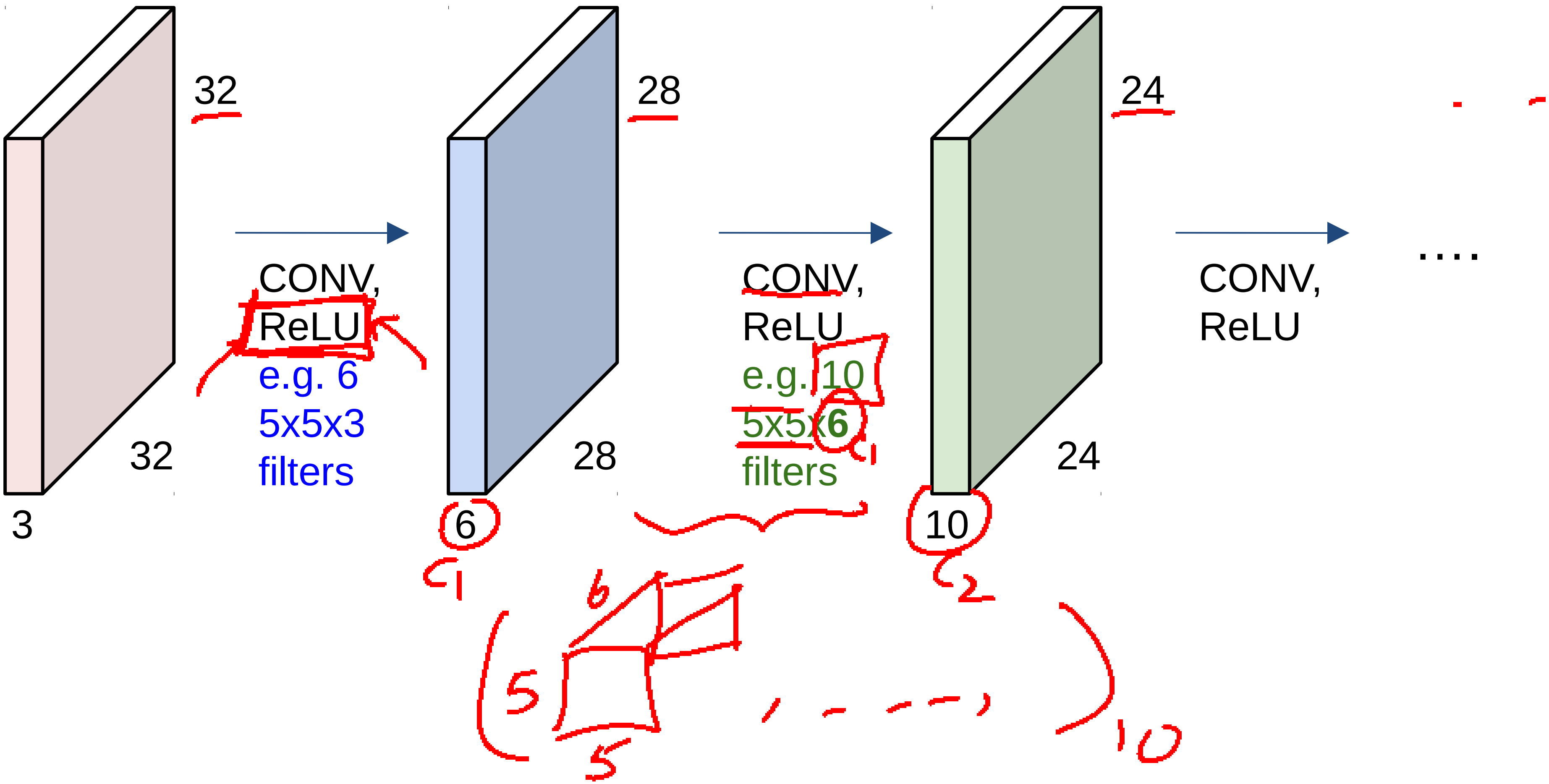


Figure Credit: Yangqing Jia, PhD Thesis

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

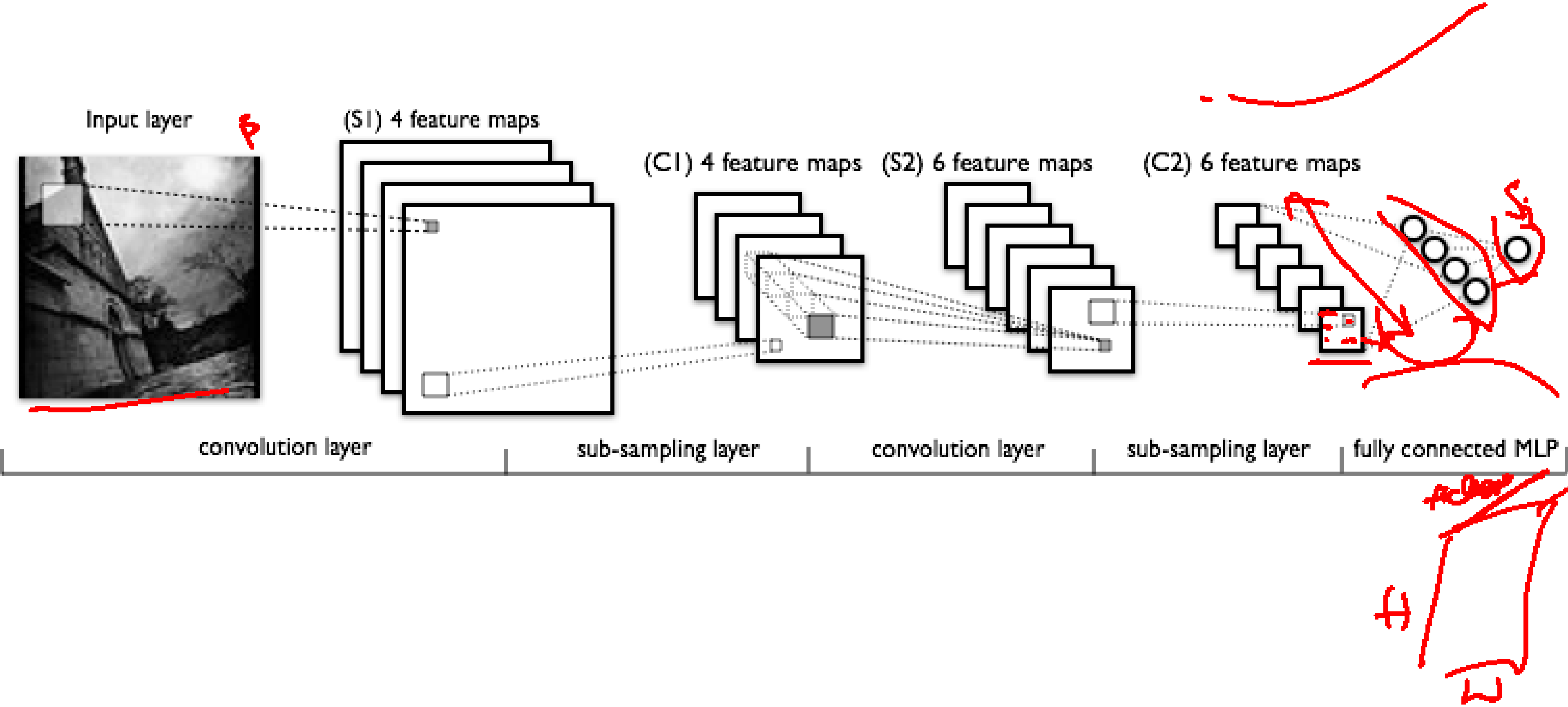


Preview: ConvNet is a sequence of Convolutional Layers, interspersed with activation functions

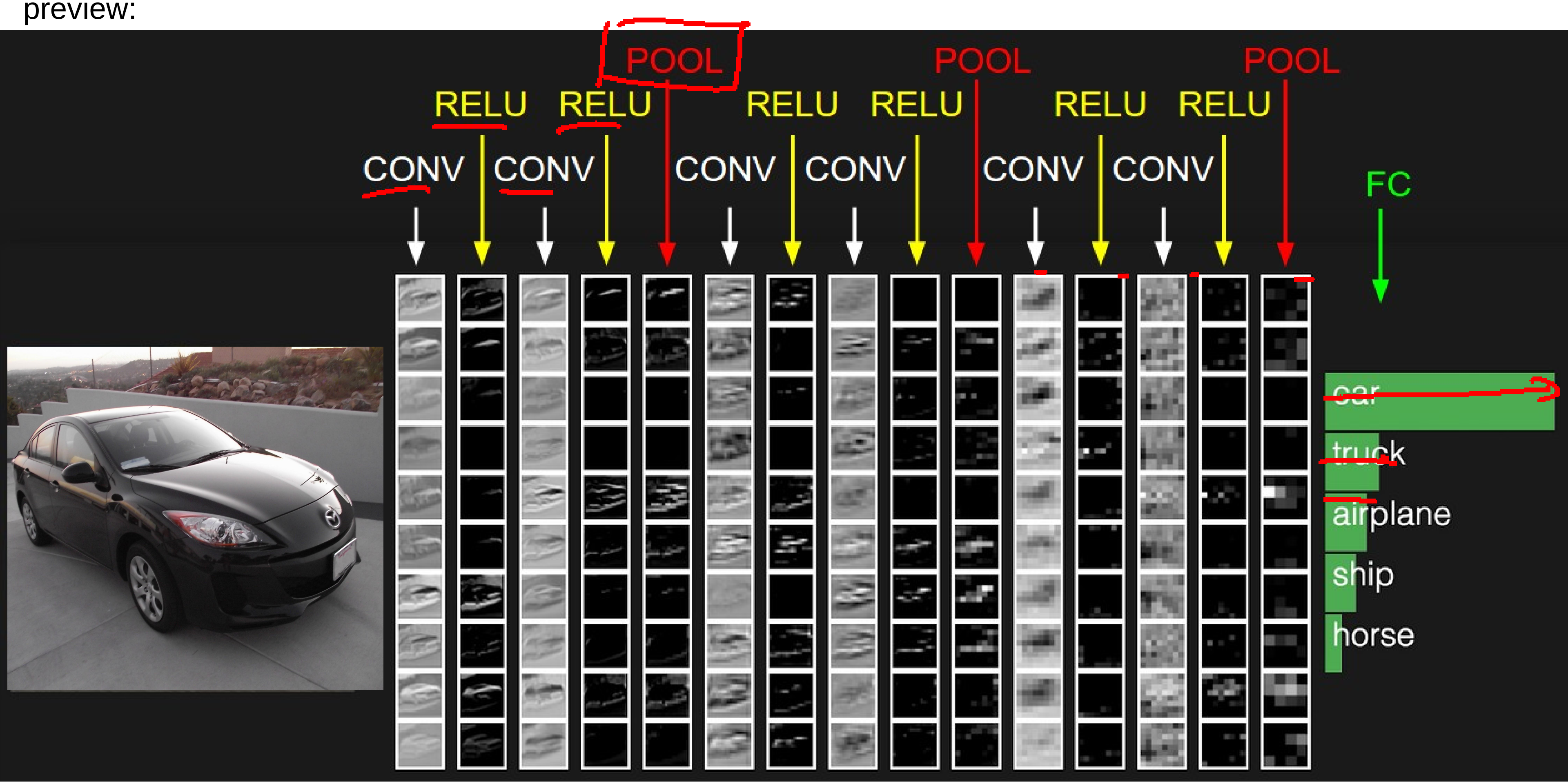


Slide Credit: Fei-Fei Li, Justin Johnson, Serena Yeung, CS 231n

Convolutional Neural Networks



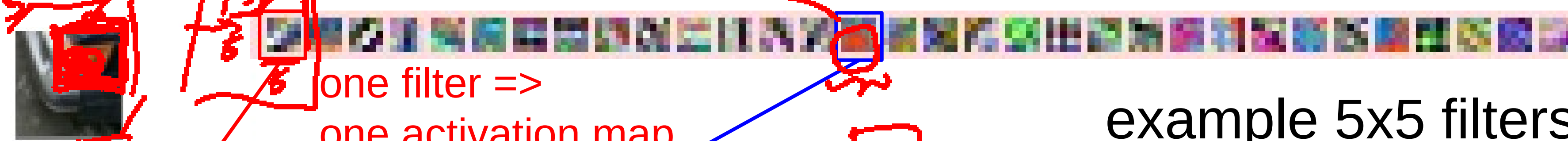
preview:



POOL
POOL
POOL
RELU
RELU
RELU
RELU
RELU
RELU
CONV
CONV
CONV
CONV
CONV
CONV
FC
car
truck
airplane
ship
horse

layer 1

$w^* = \min L(w, D)$



one filter =>
one activation map

example 5x5 filters
(32 total)

red blob

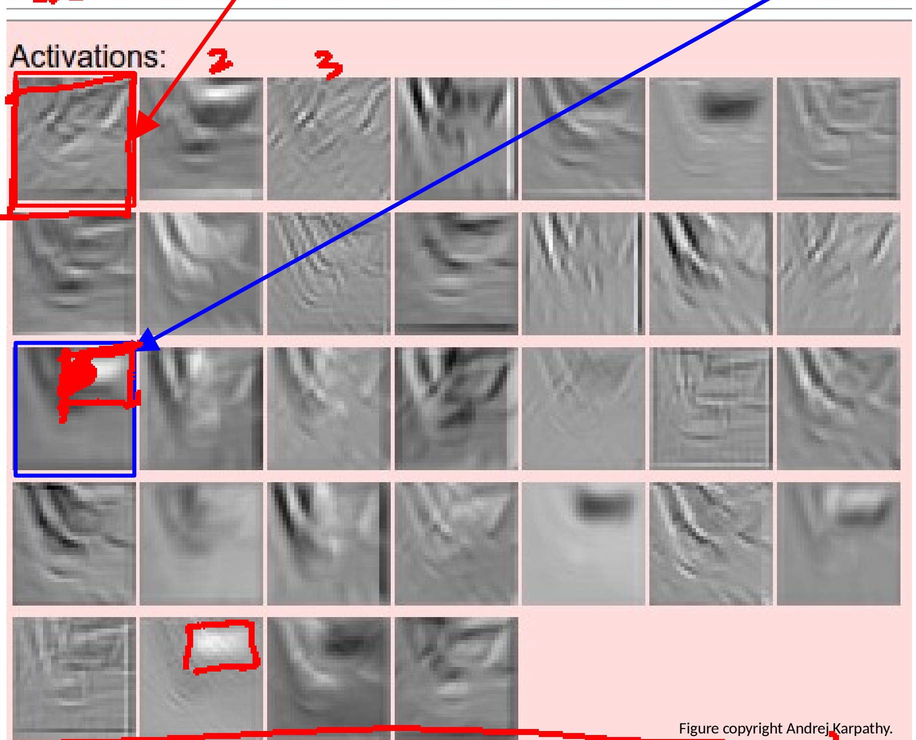
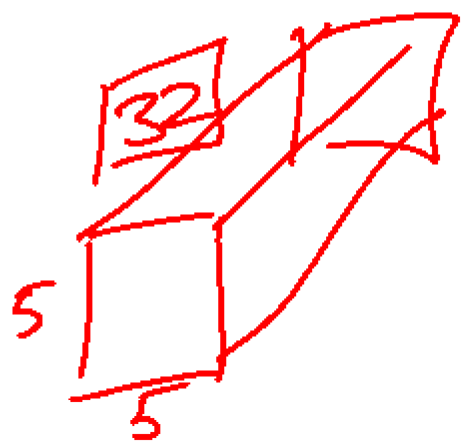


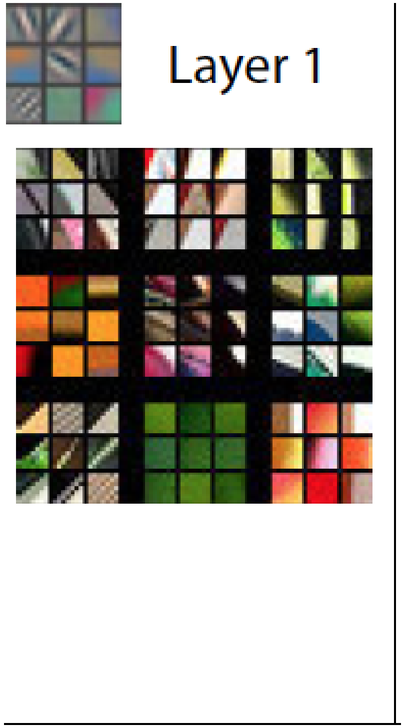
Figure copyright Andrej Karpathy.

32



Layer 2 weight

Visualizing Learned Filters

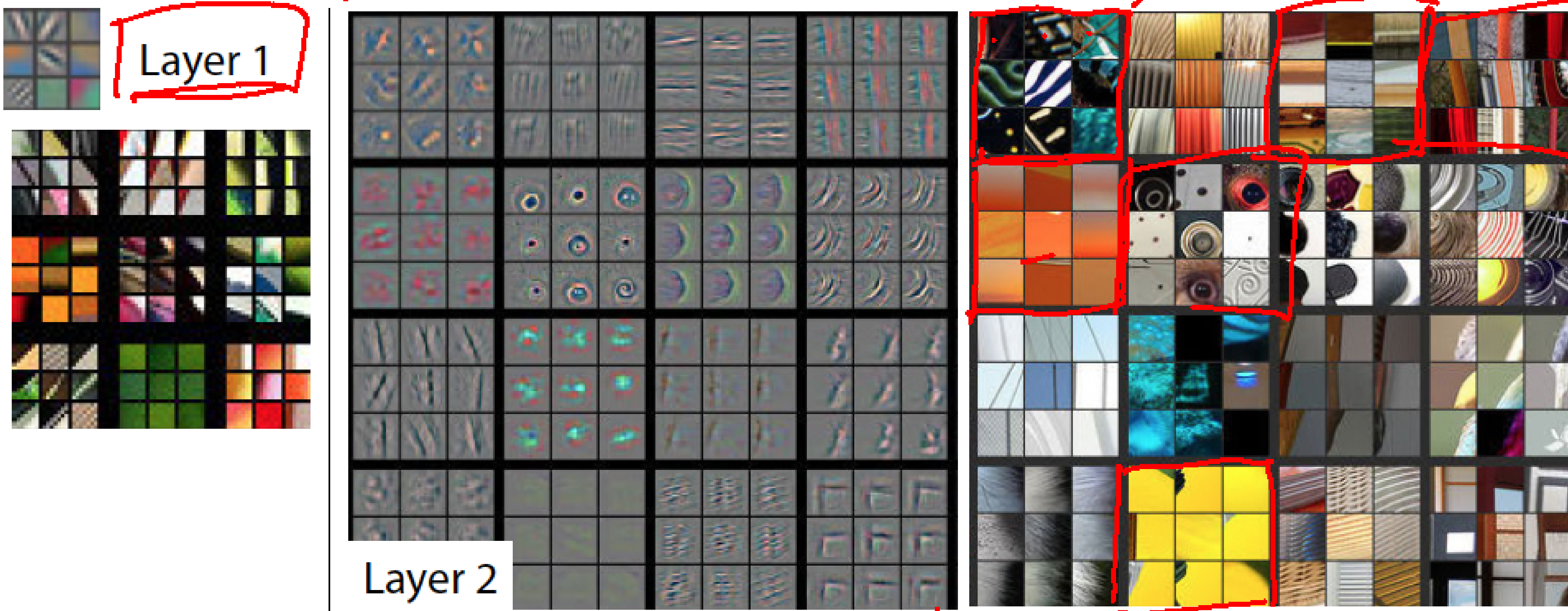


Visualizing Learned Filters

L1-3 / Ignore for now

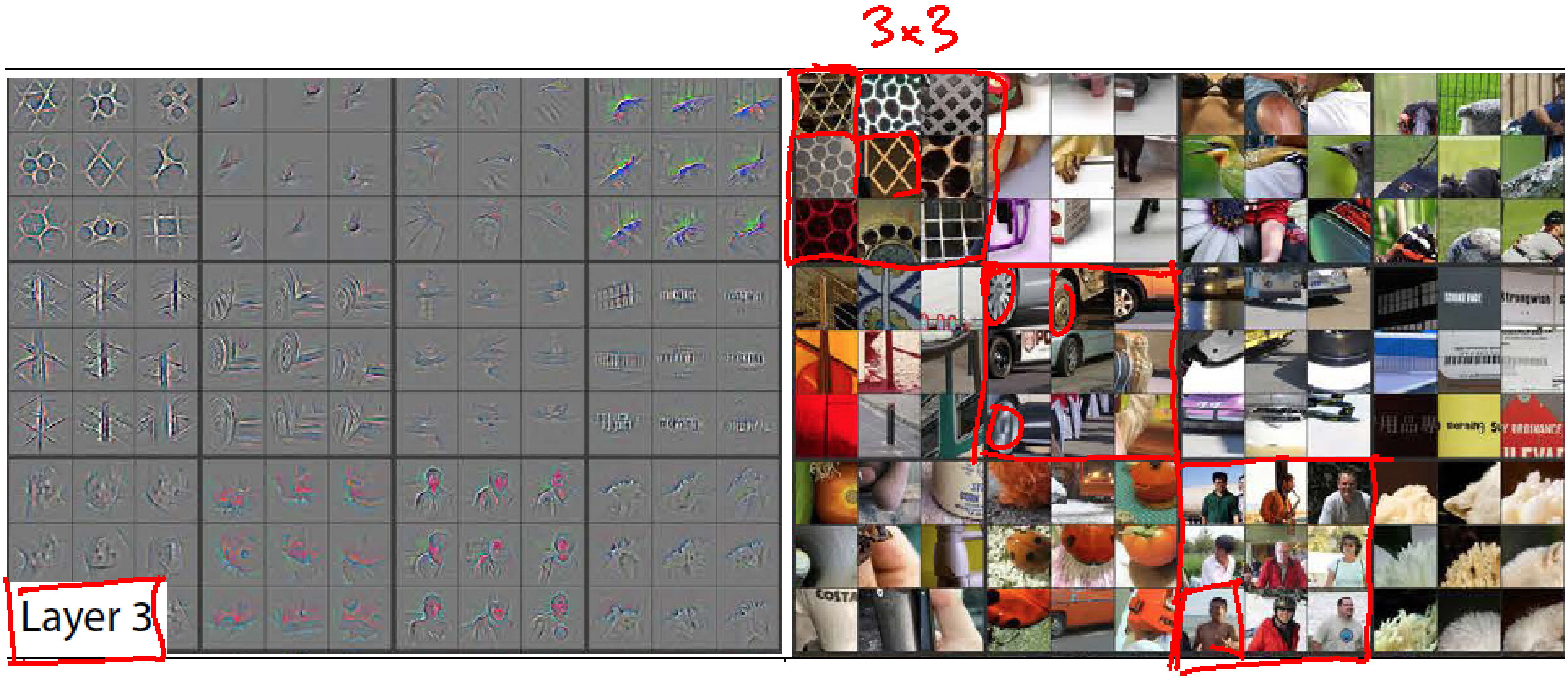
“neuron”/channel

3x3

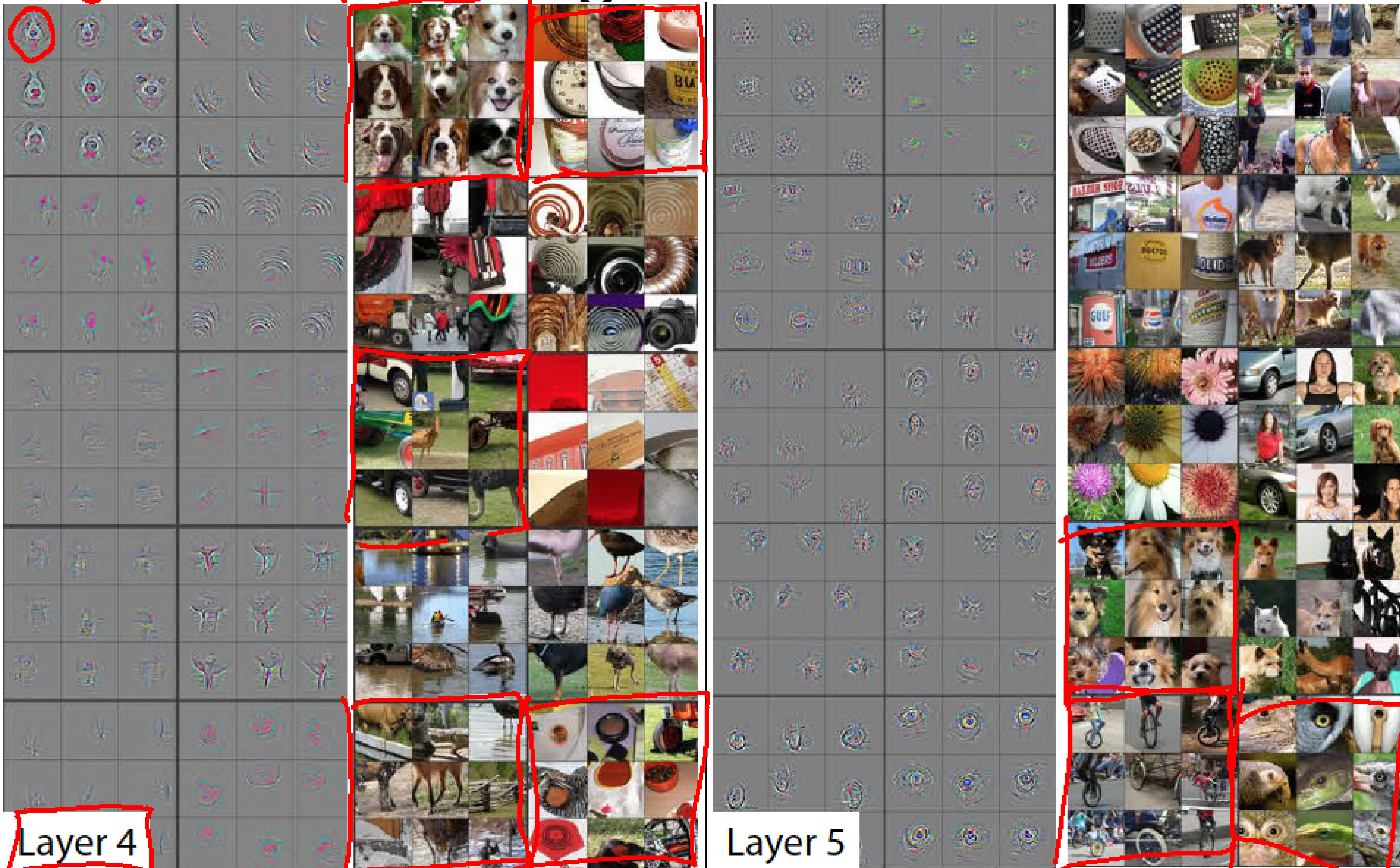


Ignore

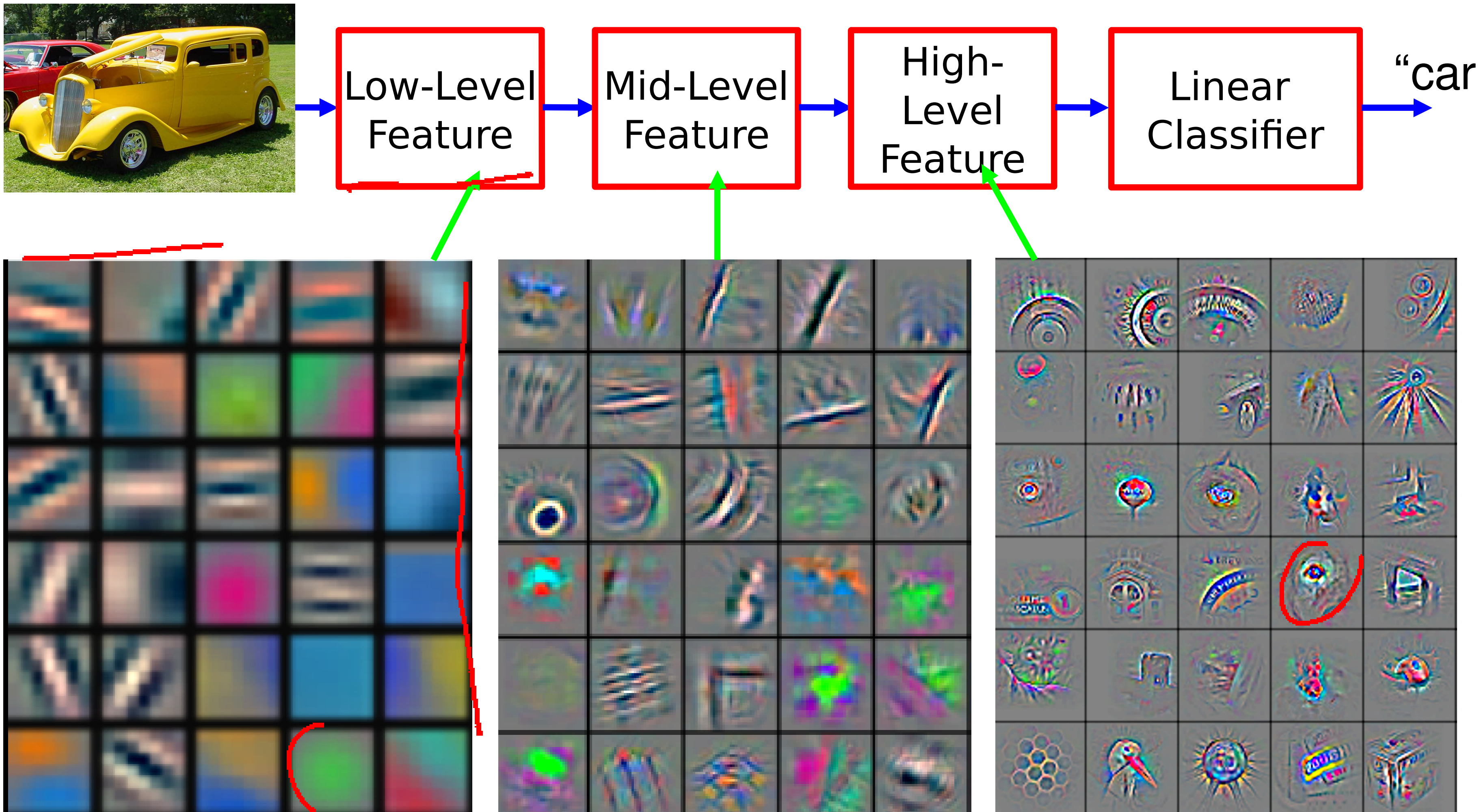
Visualizing Learned Filters



Visualizing Learned Filters



We can learn image features now!



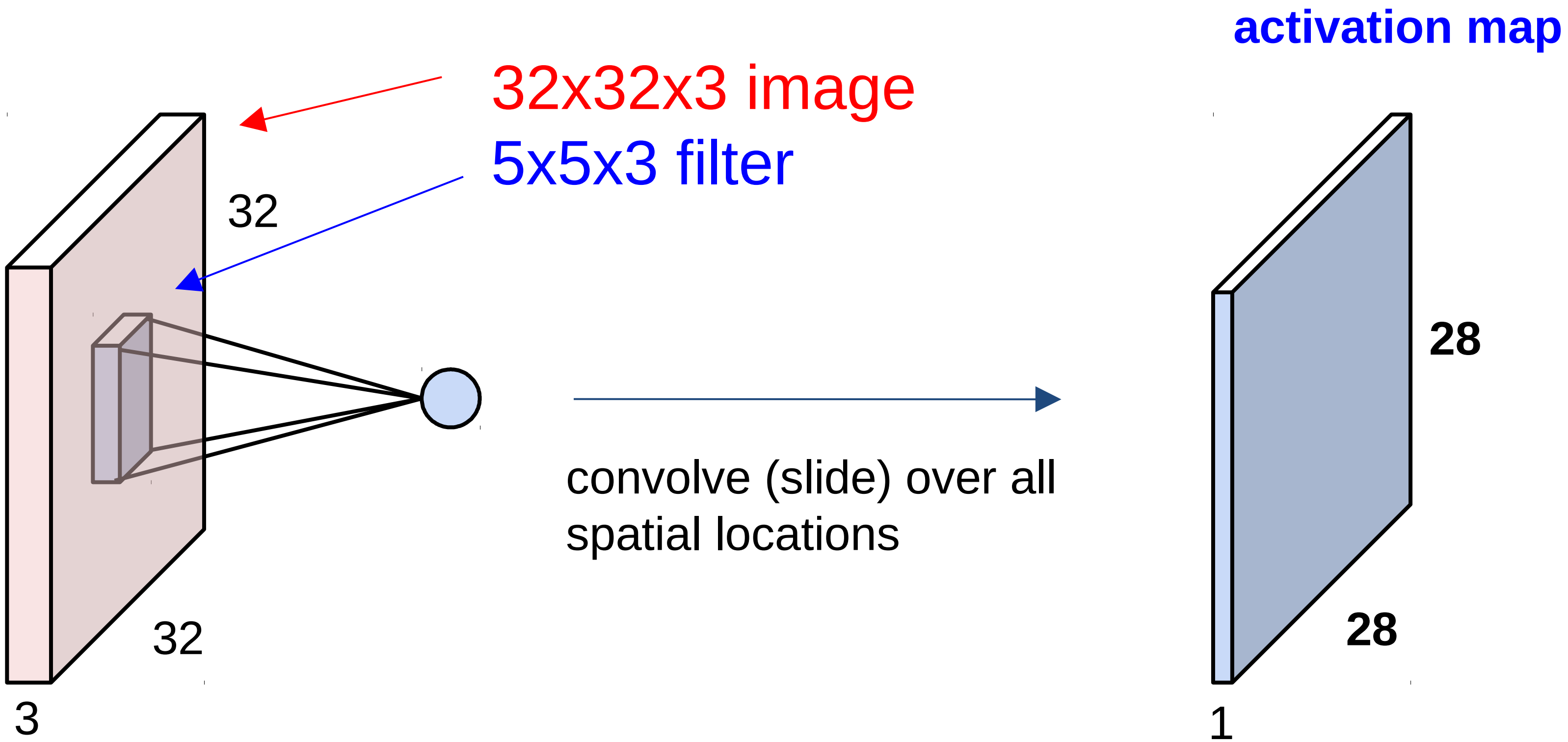
Feature visualization of convolutional net trained on ImageNet from [Zeiler & Fergus 2013]

Slide Credit: Marc'Aurelio Ranzato, Yann LeCun

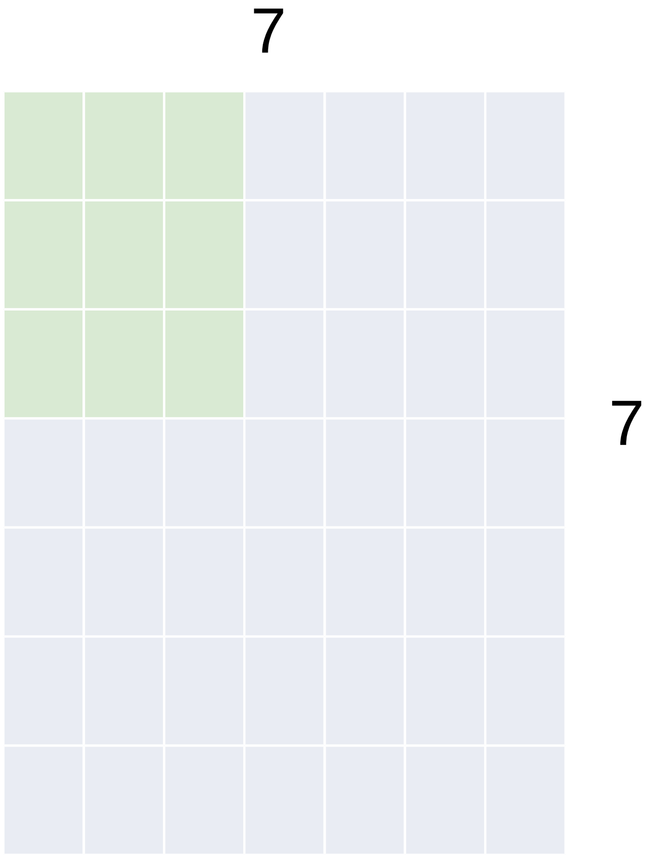
Plan for Today

- Convolutional Neural Networks
  - Features learned by CNN layers
  - Stride, padding
  - 1x1 convolutions
  - Pooling layers
  - Fully-connected layers as convolutions

A closer look at spatial dimensions:

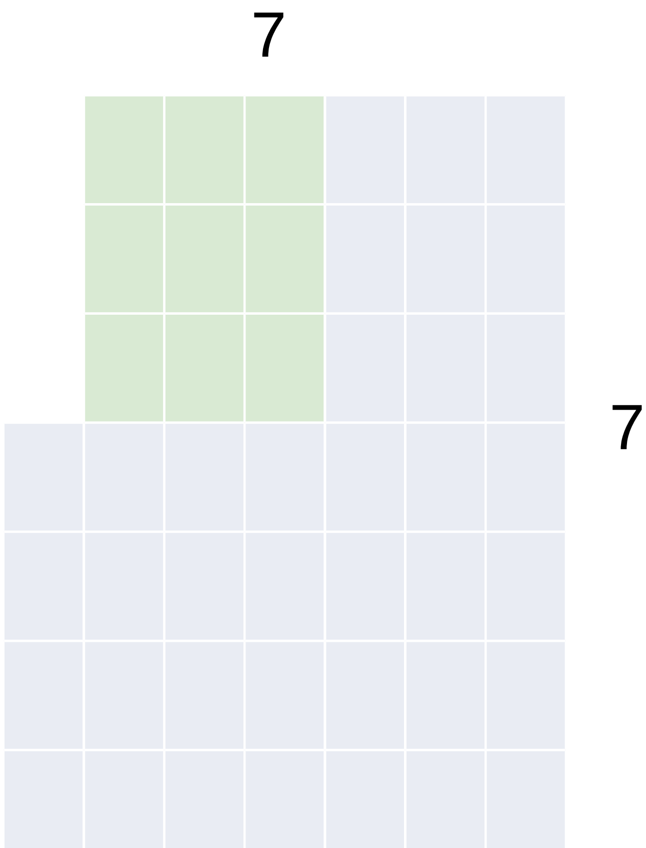


A closer look at spatial dimensions:



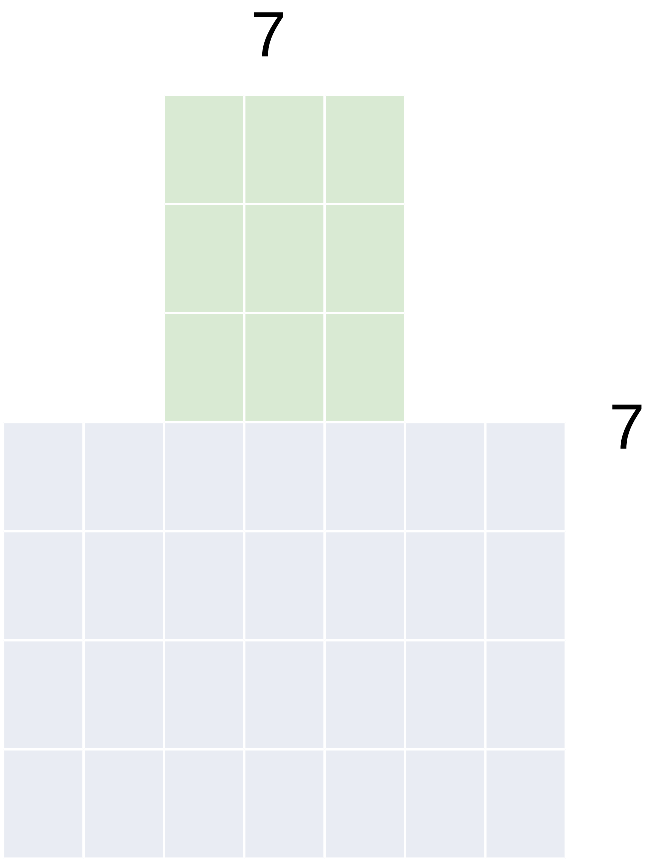
7x7 input (spatially)
assume 3x3 filter

A closer look at spatial dimensions:



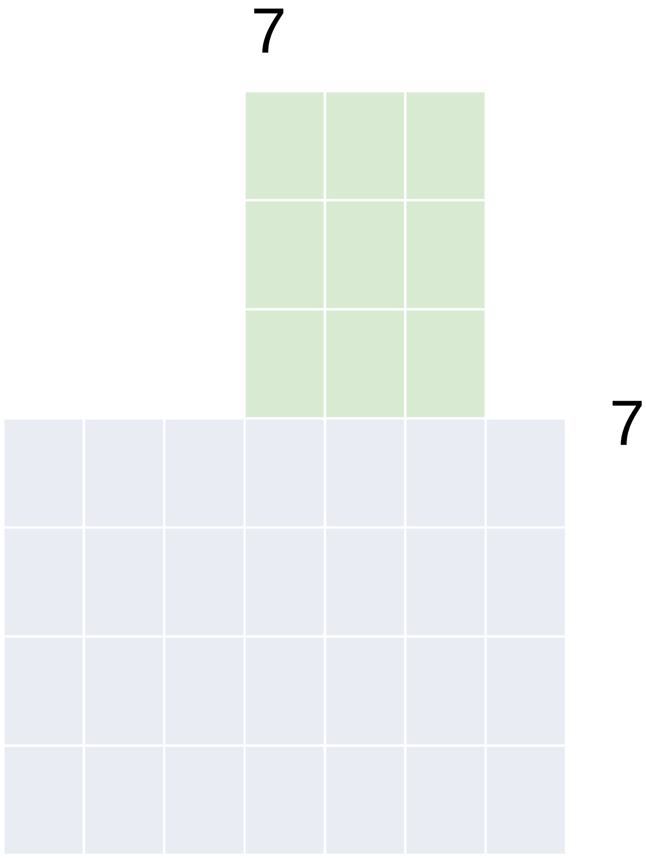
7x7 input (spatially)
assume 3x3 filter

A closer look at spatial dimensions:



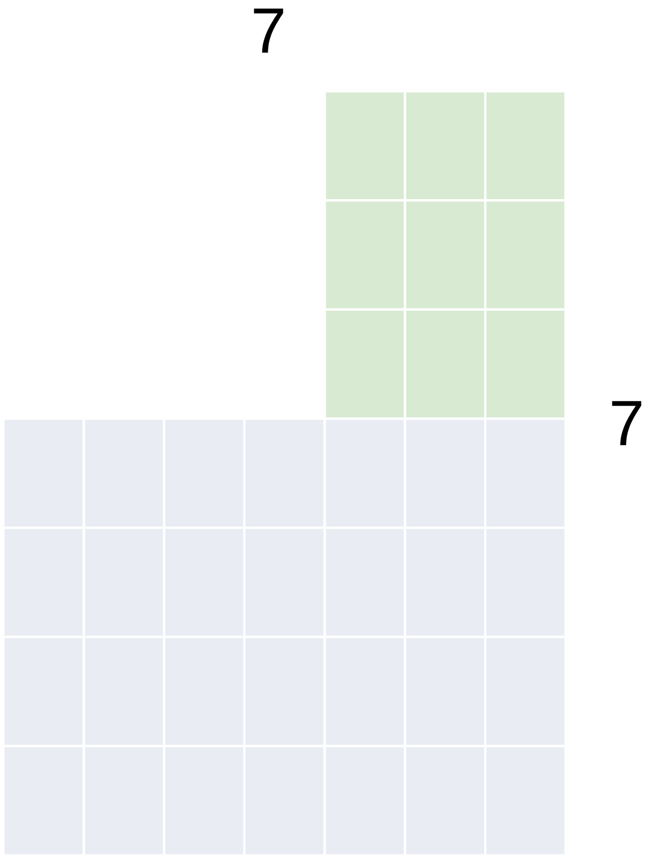
7x7 input (spatially)
assume 3x3 filter

A closer look at spatial dimensions:



7x7 input (spatially)
assume 3x3 filter

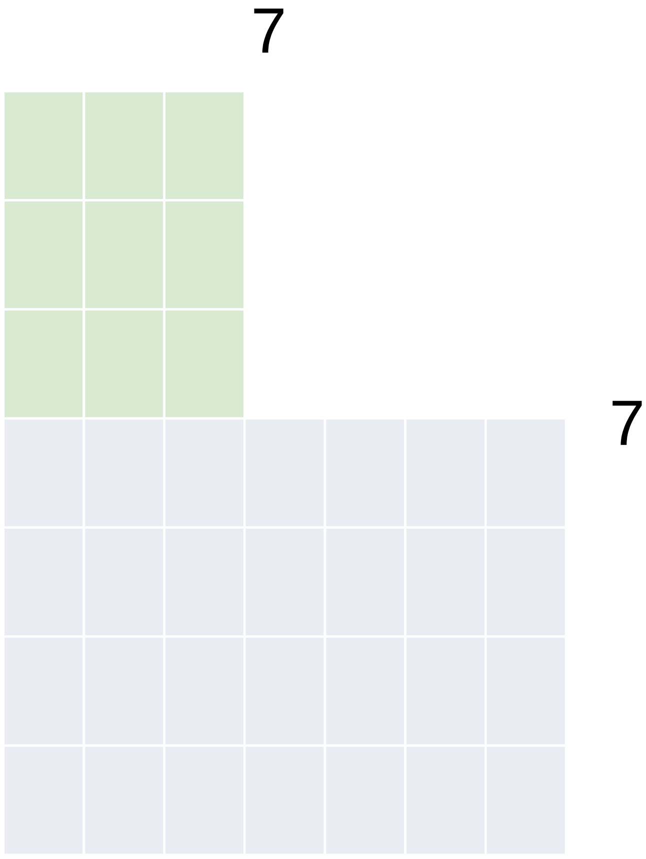
A closer look at spatial dimensions:



7x7 input (spatially)
assume 3x3 filter

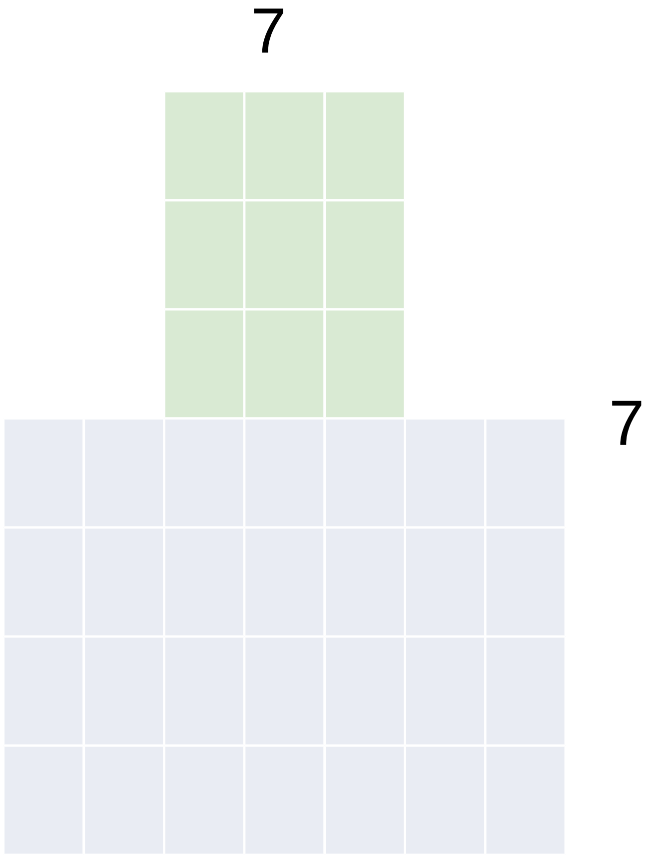
=> 5x5 output

A closer look at spatial dimensions:



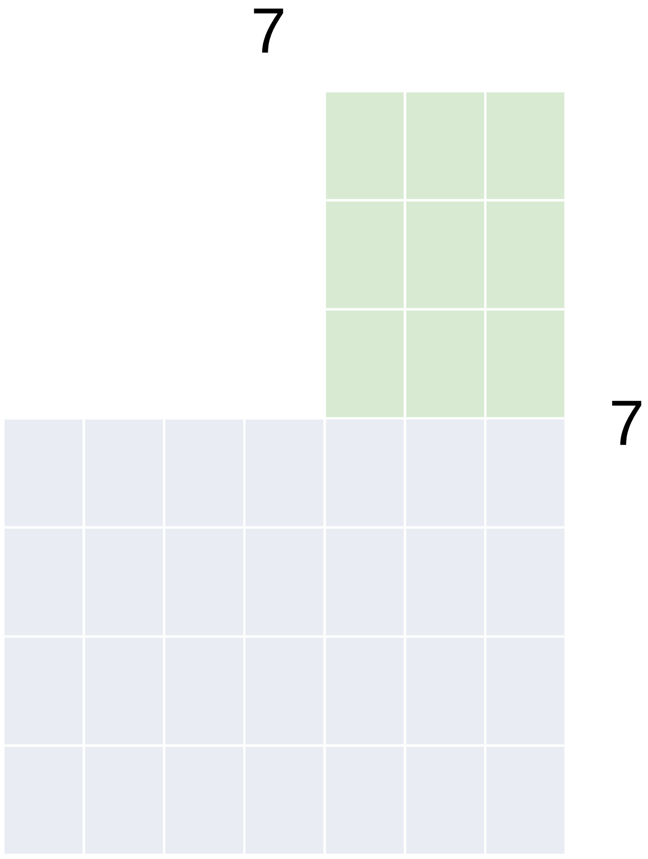
7x7 input (spatially)
assume 3x3 filter
applied **with stride 2**

A closer look at spatial dimensions:



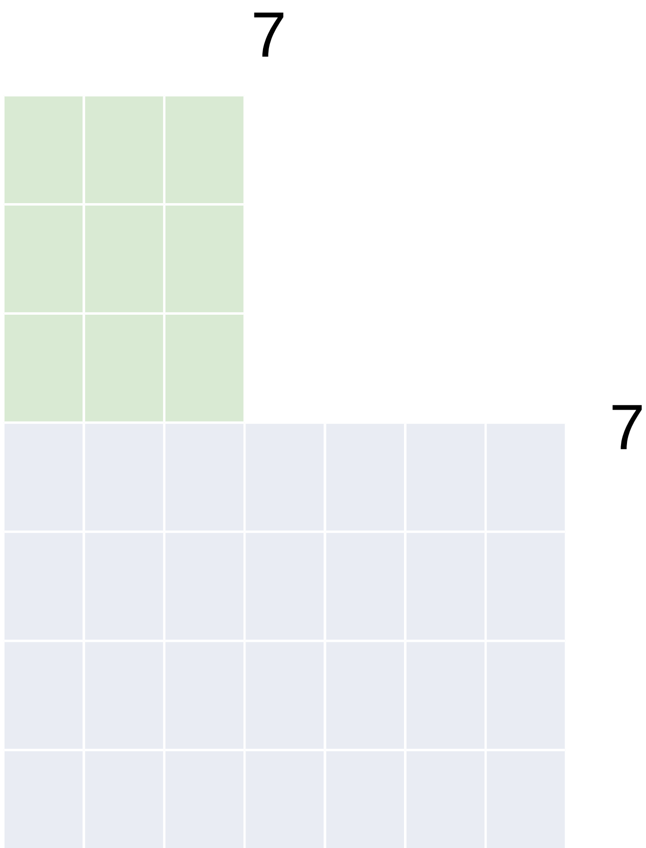
7x7 input (spatially)
assume 3x3 filter
applied **with stride 2**

A closer look at spatial dimensions:



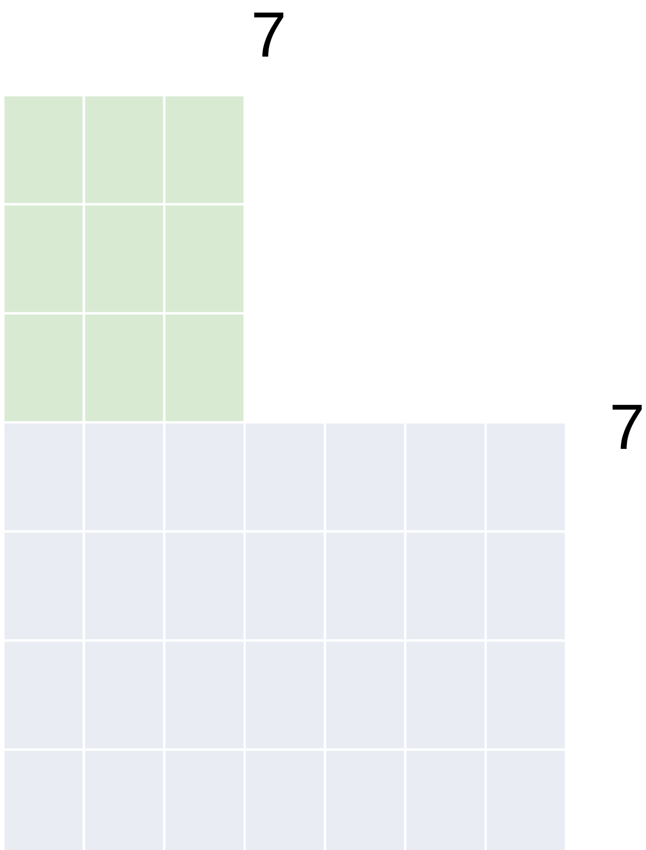
7x7 input (spatially)
assume 3x3 filter
applied **with stride 2**
=> 3x3 output!

A closer look at spatial dimensions:



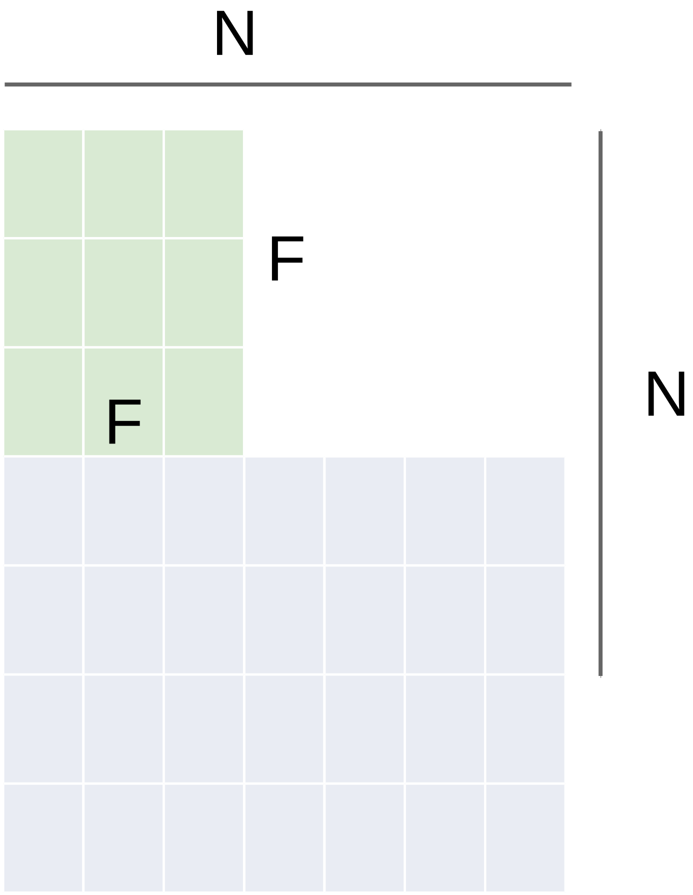
7x7 input (spatially)
assume 3x3 filter
applied **with stride 3?**

A closer look at spatial dimensions:



7x7 input (spatially)
assume 3x3 filter
applied **with stride 3?**

doesn’t fit!
cannot apply 3x3 filter on
7x7 input with stride 3.



Output size:

(N - F) / stride + 1

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

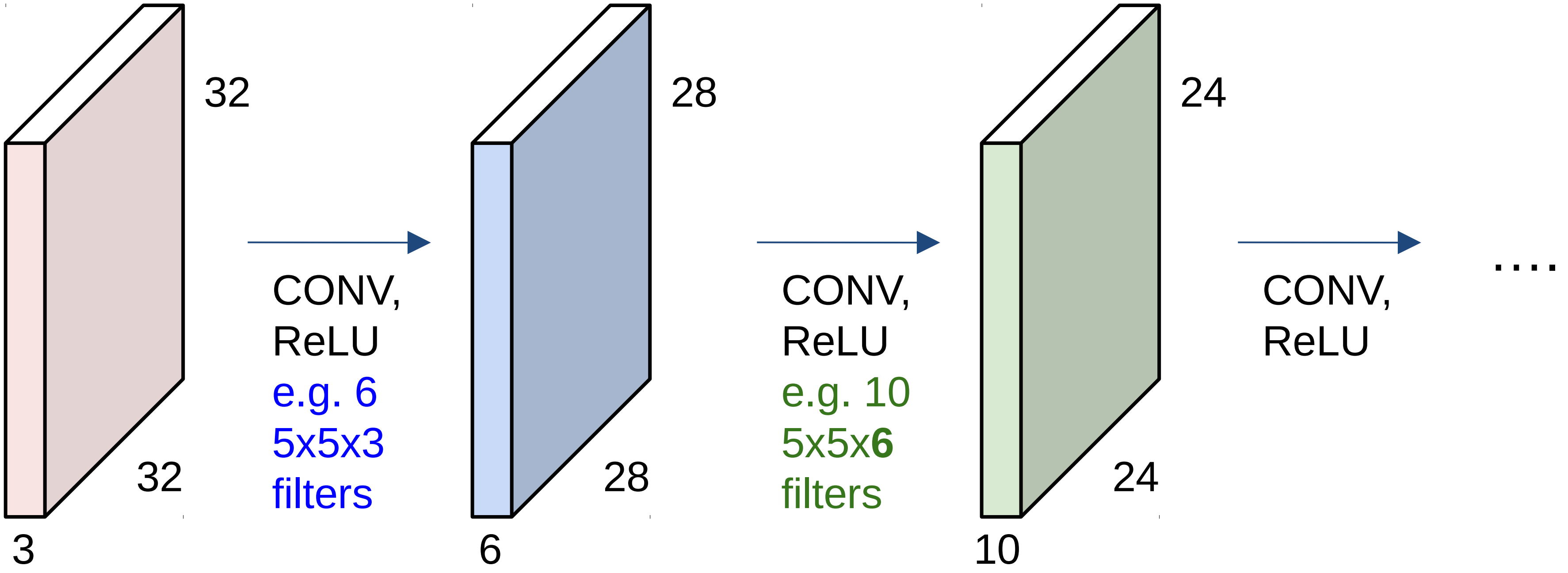
stride 1 => (7 - 3)/1 + 1 = 5

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

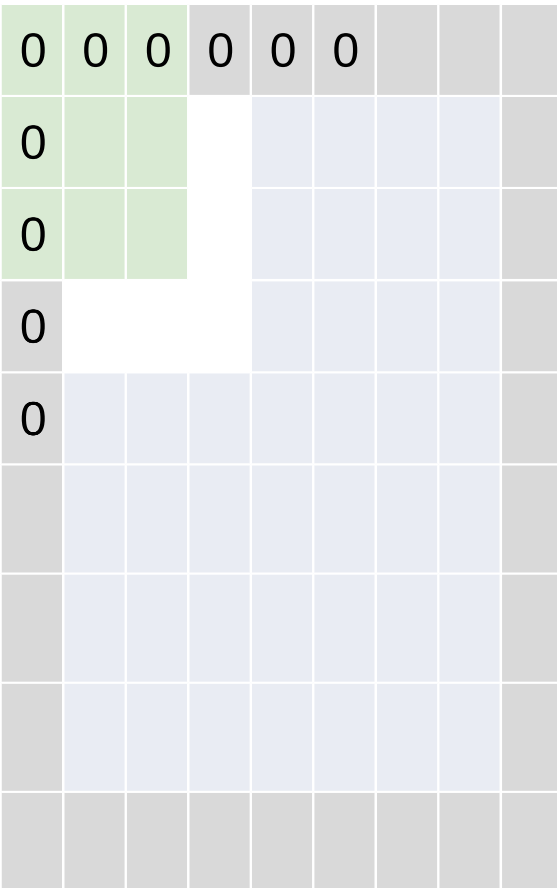
stride 3 => (7 - 3)/3 + 1 = 2.33 :\

Remember back to…

E.g. 32x32 input convolved repeatedly with 5x5 filters shrinks volumes spatially! (32 -> 28 -> 24 ...). Shrinking too fast is not good, doesn’t work well.



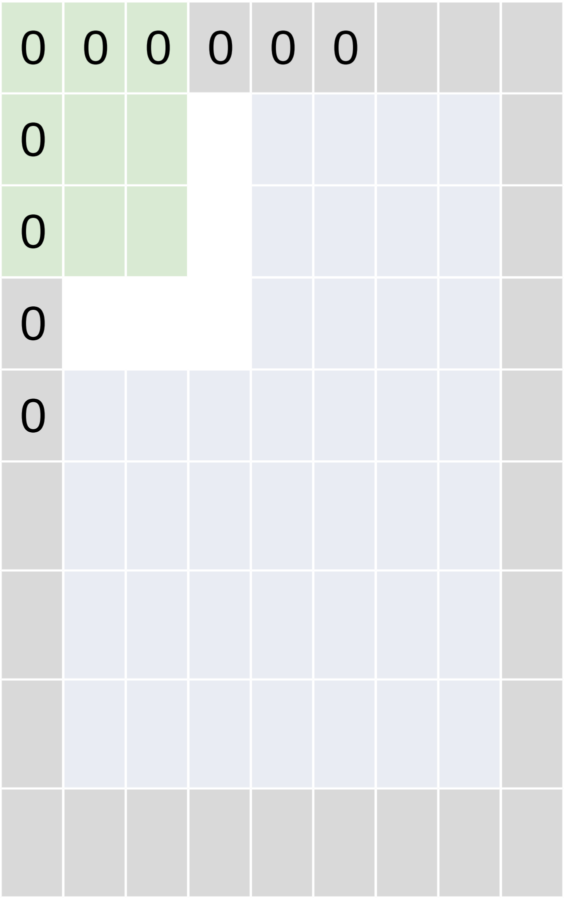
In practice: Common to zero pad the border



e.g. input 7x7
3x3 filter, applied with **stride 1**
pad with 1 pixel border => what is the output?

(recall:)
(N - F) / stride + 1

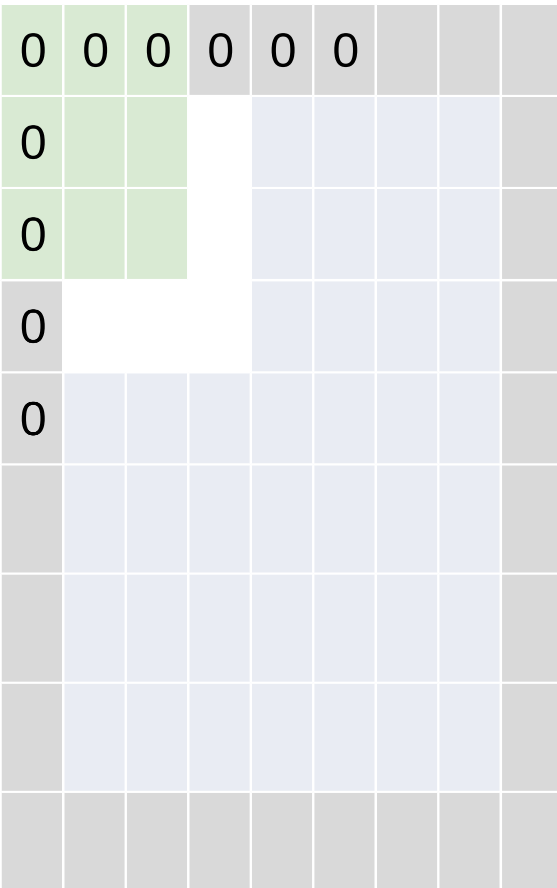
In practice: Common to zero pad the border



e.g. input 7x7
3x3 filter, applied with **stride 1**
pad with 1 pixel border => what is the output?

7x7 output!

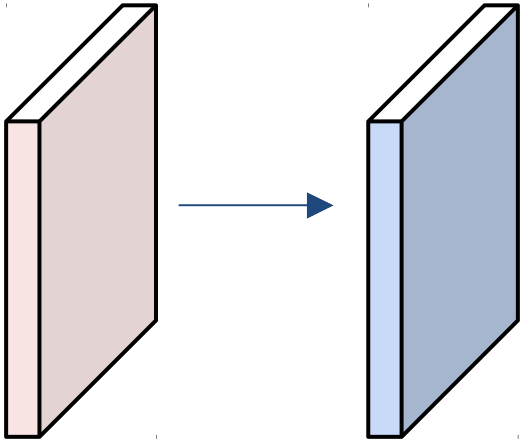
In practice: Common to zero pad the border



e.g. input 7x7
3x3 filter, applied with **stride 1**
pad with 1 pixel border => what is the output?

7x7 output!
in general, common to see CONV layers with stride 1, filters of size FxF, and zero-padding with (F-1)/2. (will preserve size spatially)
e.g. F = 3 => zero pad with 1
F = 5 => zero pad with 2
F = 7 => zero pad with 3

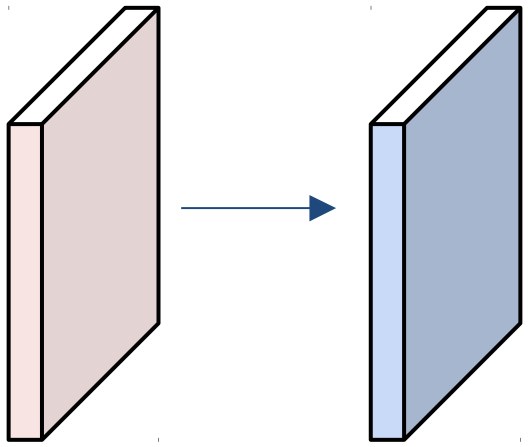
Examples time:



Input volume: **32x32x3**
10 5x5 filters with stride 1, pad 2

Output volume size: ?

Examples time:



Input volume: **32x32x3**

10 5x5 filters with stride 1, pad 2

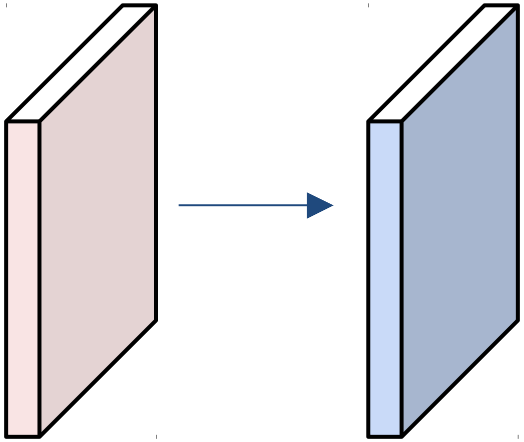
Output volume size:

(32+2*2-5)/1+1 = 32 spatially, so

32x32x10

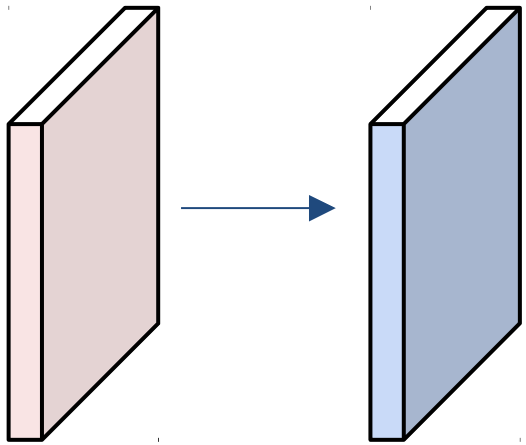
Examples time:

Input volume: **32x32x3**
10 5x5 filters with stride 1, pad 2



Number of parameters in this layer?

Examples time:



Input volume: **32x32x3**

10 5x5 filters with stride 1, pad 2

Number of parameters in this layer?

each filter has 5*5*3 + 1 = 76 params (+1 for bias)

=> 76*10 = **760**

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.

Common settings:

K = (powers of 2, e.g. 32, 64, 128, 512)

- F = 3, S = 1, P = 1
- F = 5, S = 1, P = 2
- F = 5, S = 2, P = ? (whatever fits)
- F = 1, S = 1, P = 0

Summary. To summarize, the Conv Layer:

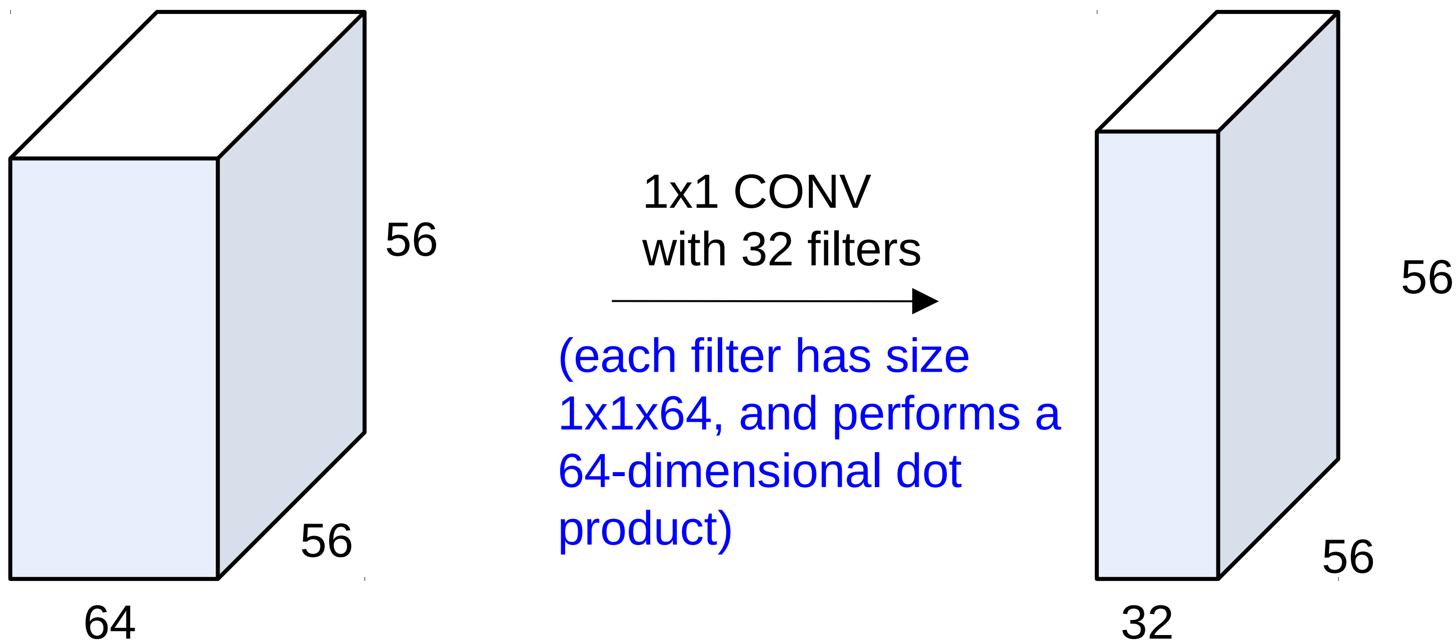
- Accepts a volume of size $W_1 \times H_1 \times D_1$
- Requires four hyperparameters:
  - Number of filters K,
  - their spatial extent F,
  - the stride S,
  - the amount of zero padding P.
- Produces a volume of size $W_2 \times H_2 \times D_2$ where:
  - $W_2 = (W_1 - F + 2P)/S + 1$
  - $H_2 = (H_1 - F + 2P)/S + 1$ (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.

Plan for Today

- Convolutional Neural Networks
  - Features learned by CNN layers
  - Stride, padding
  - 1x1 convolutions
  - Pooling layers
  - Fully-connected layers as convolutions
  - Backprop in conv layers

Can we have 1x1 filters?

1x1 convolution layers make perfect sense



Fully Connected Layer as 1x1 Conv

32x32x3 image -> stretch to 3072 x 1