CS 4803 / 7643: Deep Learning

Topics:

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

Dhruv Batra

Georgia Tech

Administrativia

- HW1 Reminder
  - Due: 10/02, 11:55pm

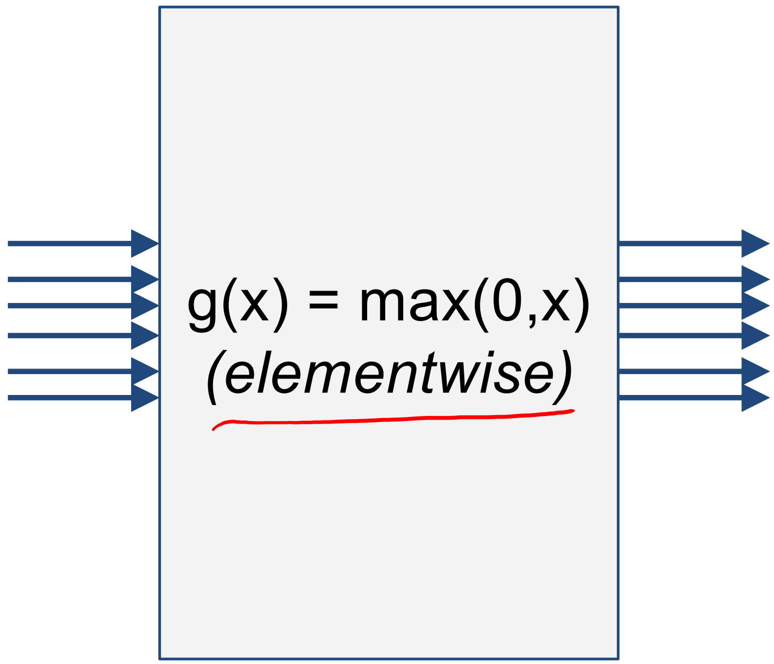
Recap from last time

Jacobian of ReLU

$\vec{h}^{\ell-1} \in \mathbb{R}^{4096}$

$\vec{h}^{\ell} \in \mathbb{R}^{4096}$

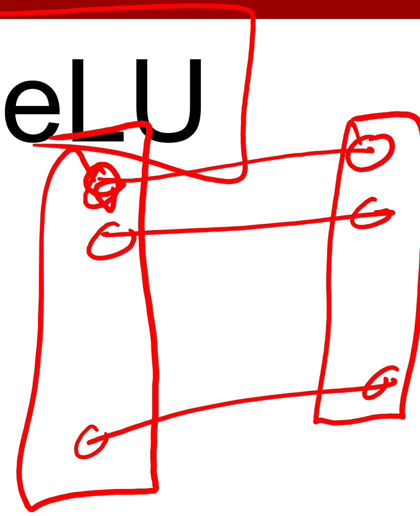
4096-d
input vector



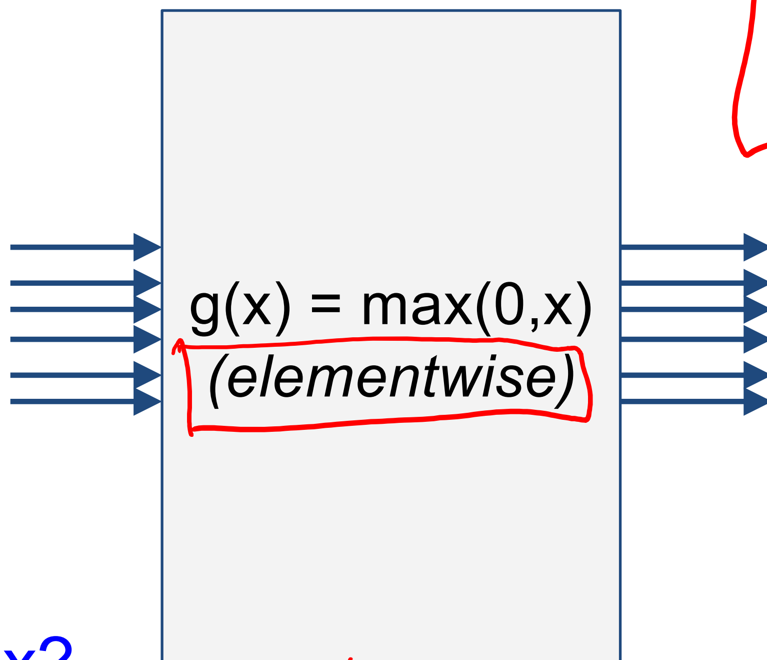
4096-d
output vector

$[\quad\quad]$ $\begin{bmatrix} \ddots & 0 \\ 0 & \ddots \end{bmatrix}$

Jacobian of ReLU



4096-d
input vector



4096-d
output vector

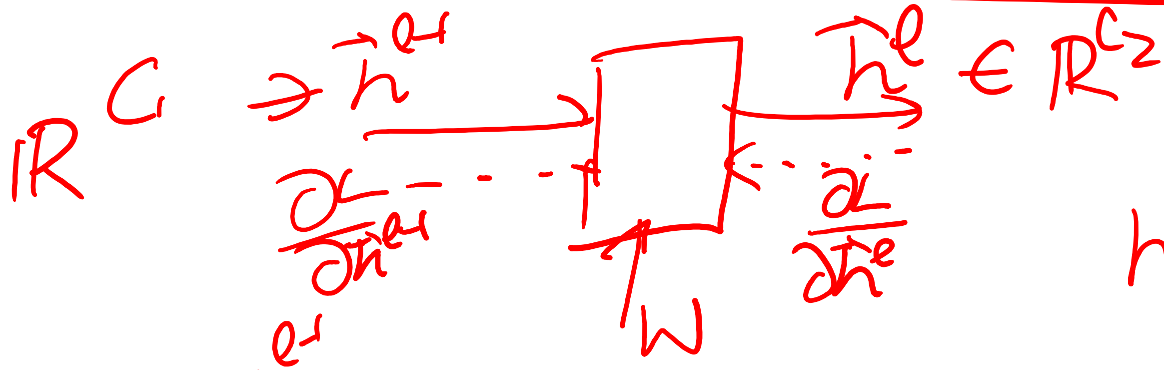
Q: what is the size of the Jacobian matrix?

$$\frac{\partial h_i^l}{\partial h_j^{l-1}} = 0 \ \forall i \neq j$$

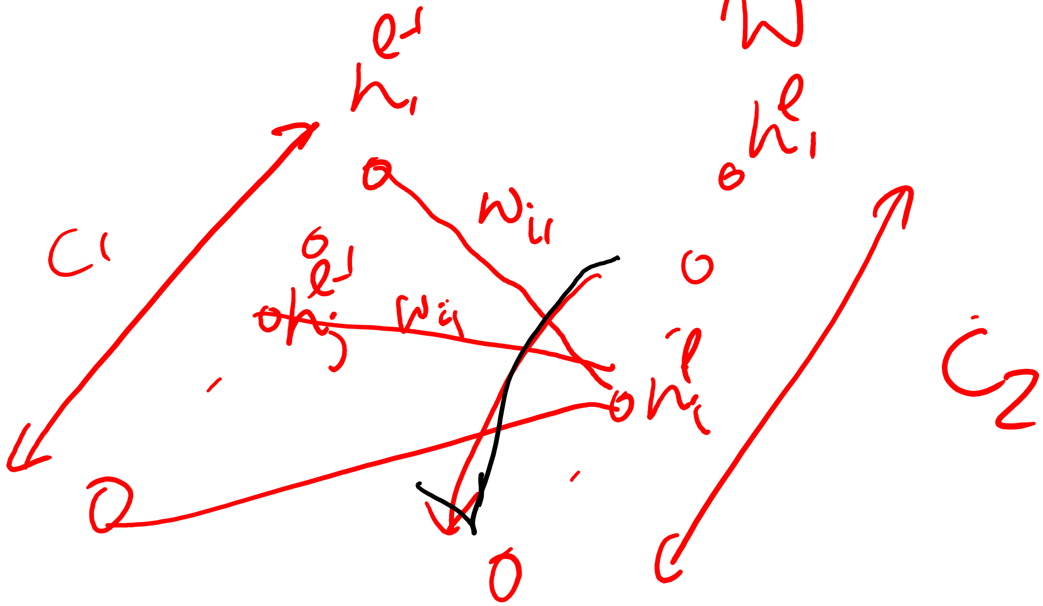
$$\frac{\partial \vec{h}^l}{\partial \vec{h}^{l-1}} = \begin{array}{c|c} & j \\ \hline i & \left[\frac{\partial h_i^l}{\partial h_j^{l-1}} \right] \end{array} \rightarrow \begin{bmatrix} \ddots & 0 \\ 0 & \ddots \end{bmatrix}$$

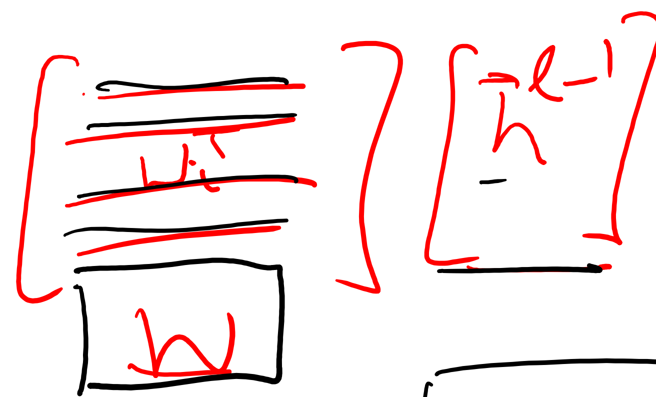
4096 x 4096

Jacobians of FC-Layer



$$h_i^{\ell} = \vec{w}_i^{T} \vec{h}^{\ell-1}$$





$$\vec{h}^{\ell} = \boxed{W}\, \vec{h}^{\ell-1}$$

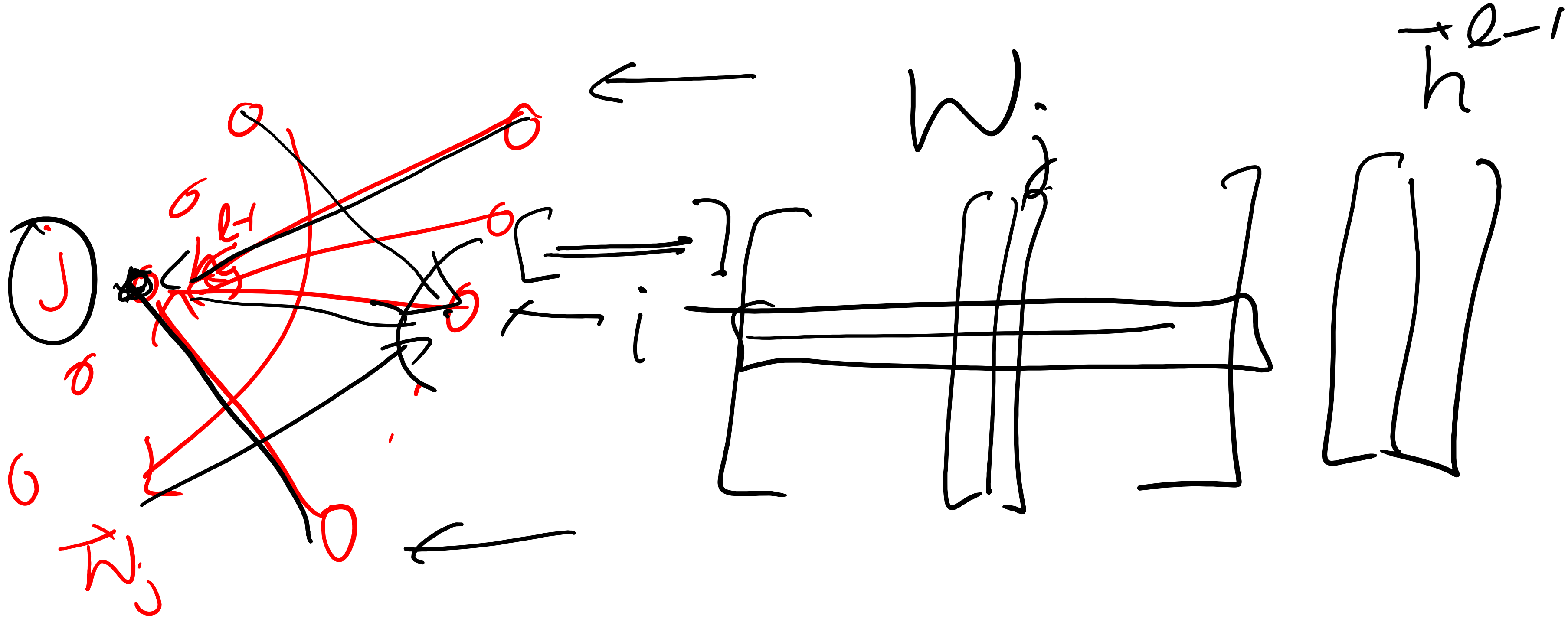
F-Pass

$\vec{w}_i$

$$\boxed{\frac{\partial L}{\partial \vec{h}^{\ell-1}}} = \boxed{\frac{\partial L}{\partial \vec{h}^{\ell}}} \cdot \underbrace{\frac{\partial \vec{h}^{\ell}}{\partial \vec{h}^{\ell-1}}}_{\to\, W}$$

B-Pass

Jacobians of FC-Layer



Jacobians of FC-Layer

Convolutional Neural Networks

(without the brain stuff)

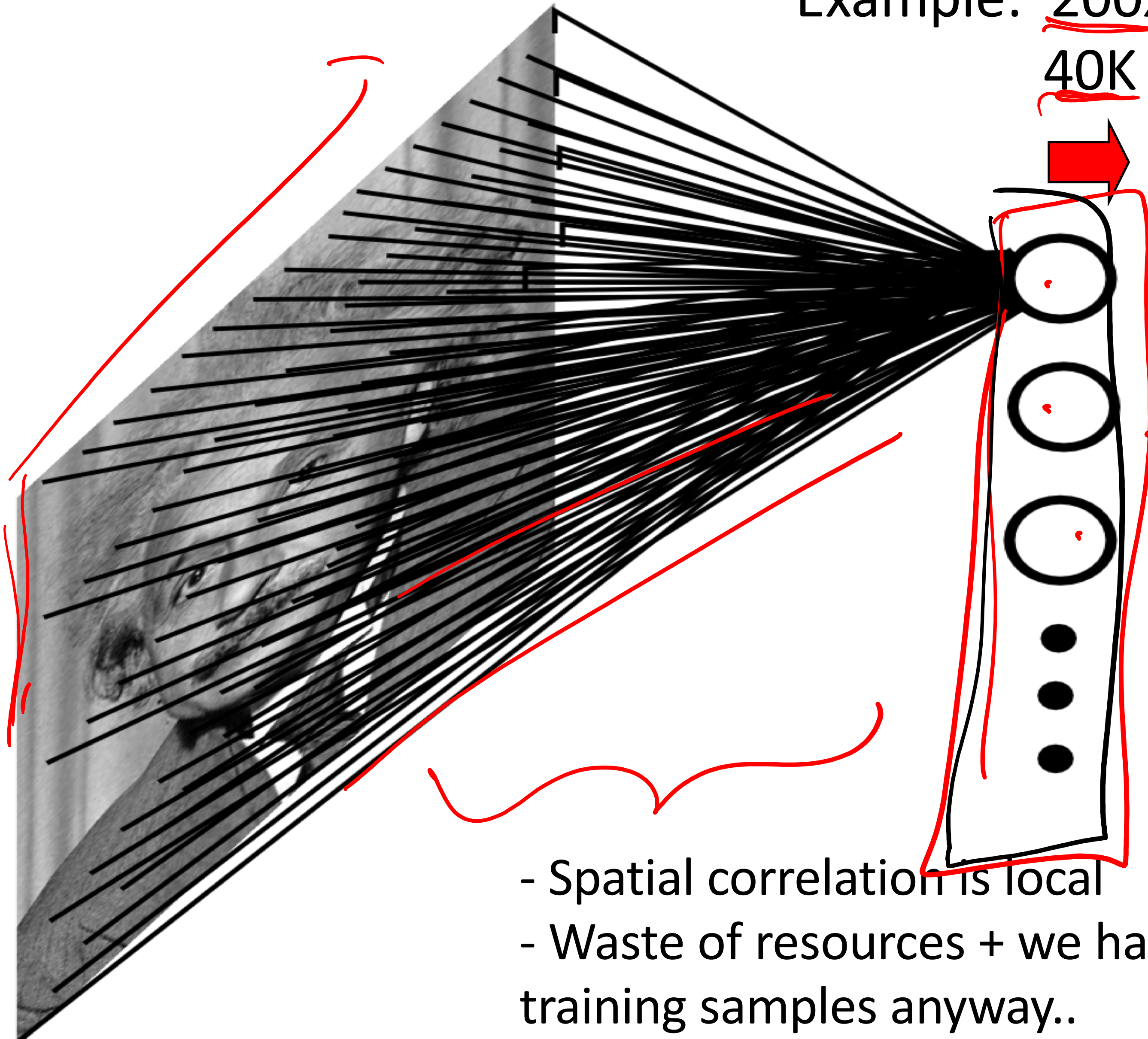
Slide Credit: Fei-Fei Li, Justin Johnson, Serena Yeung, CS 231n

Fully Connected Layer

Example: 200x200 image 4×10^4

40K hidden units

~2B parameters!!!



4×10^4

4×10^4

4×10^4

W

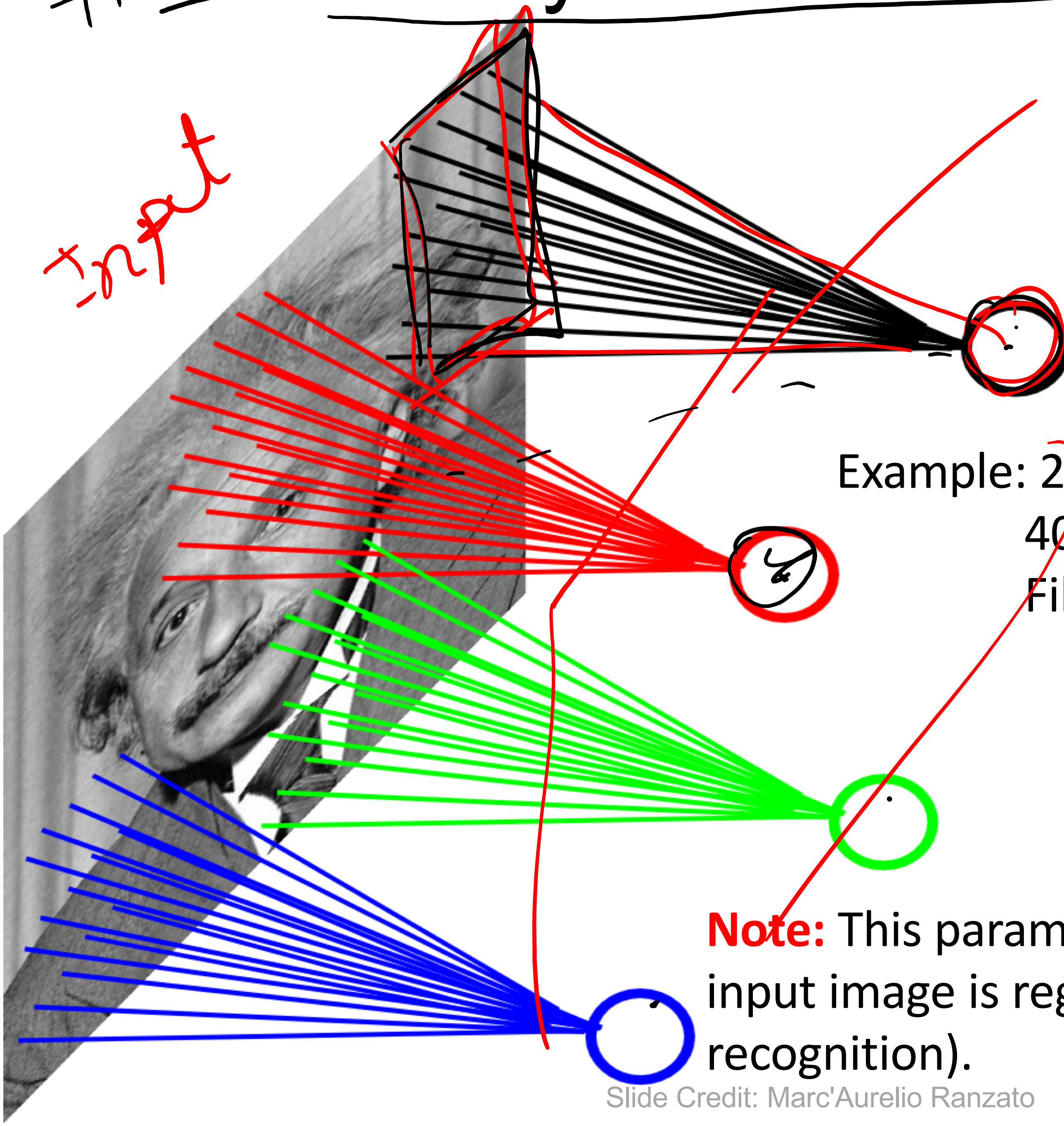
$\boxed{1.6 \times 10^9}$

- Spatial correlation is local
- Waste of resources + we have not enough training samples anyway..

#1 Locally Connected Layer

W

Input



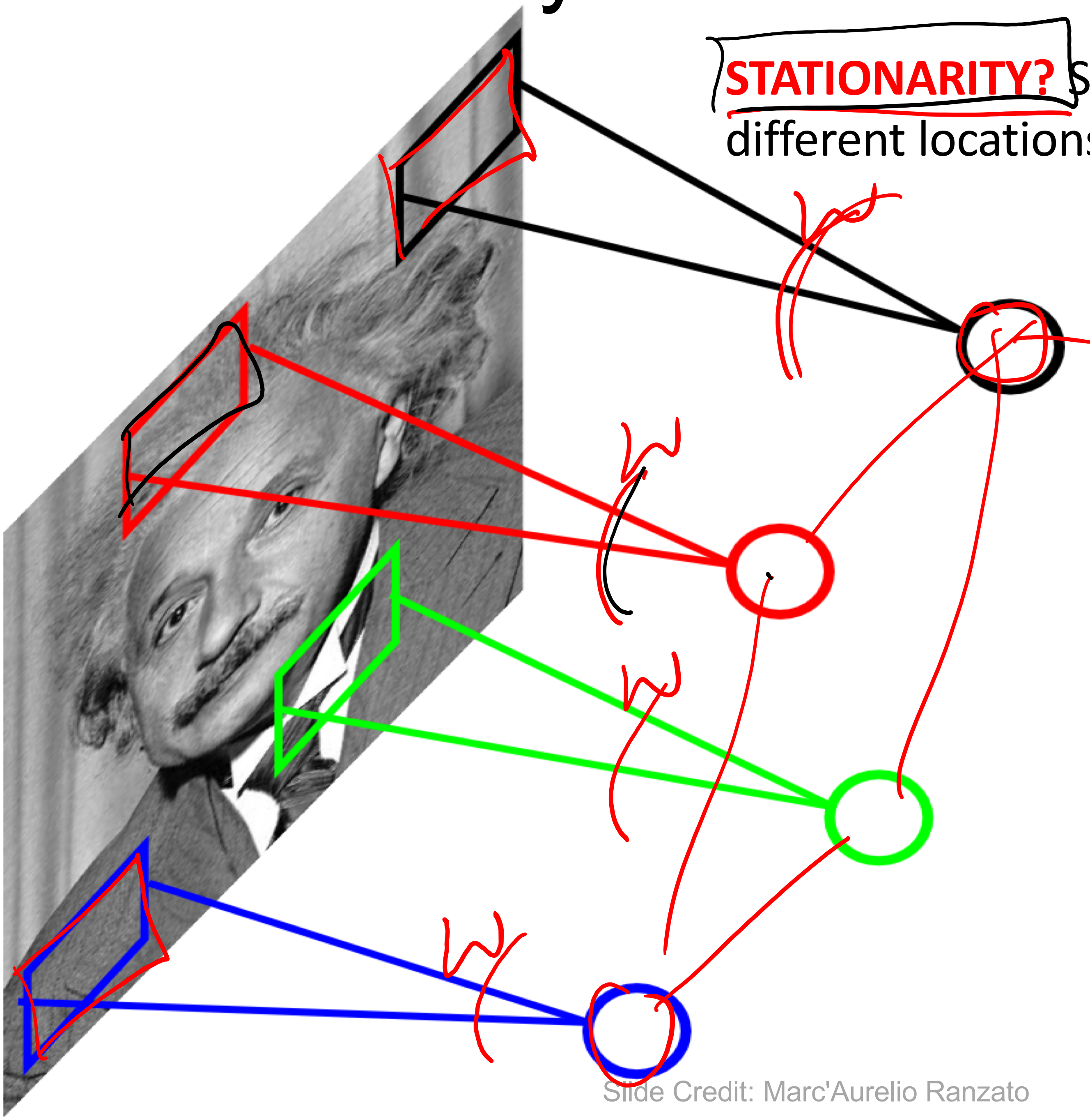
Example: 200x200 image
40K hidden units
Filter size: 10x10
4M parameters

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

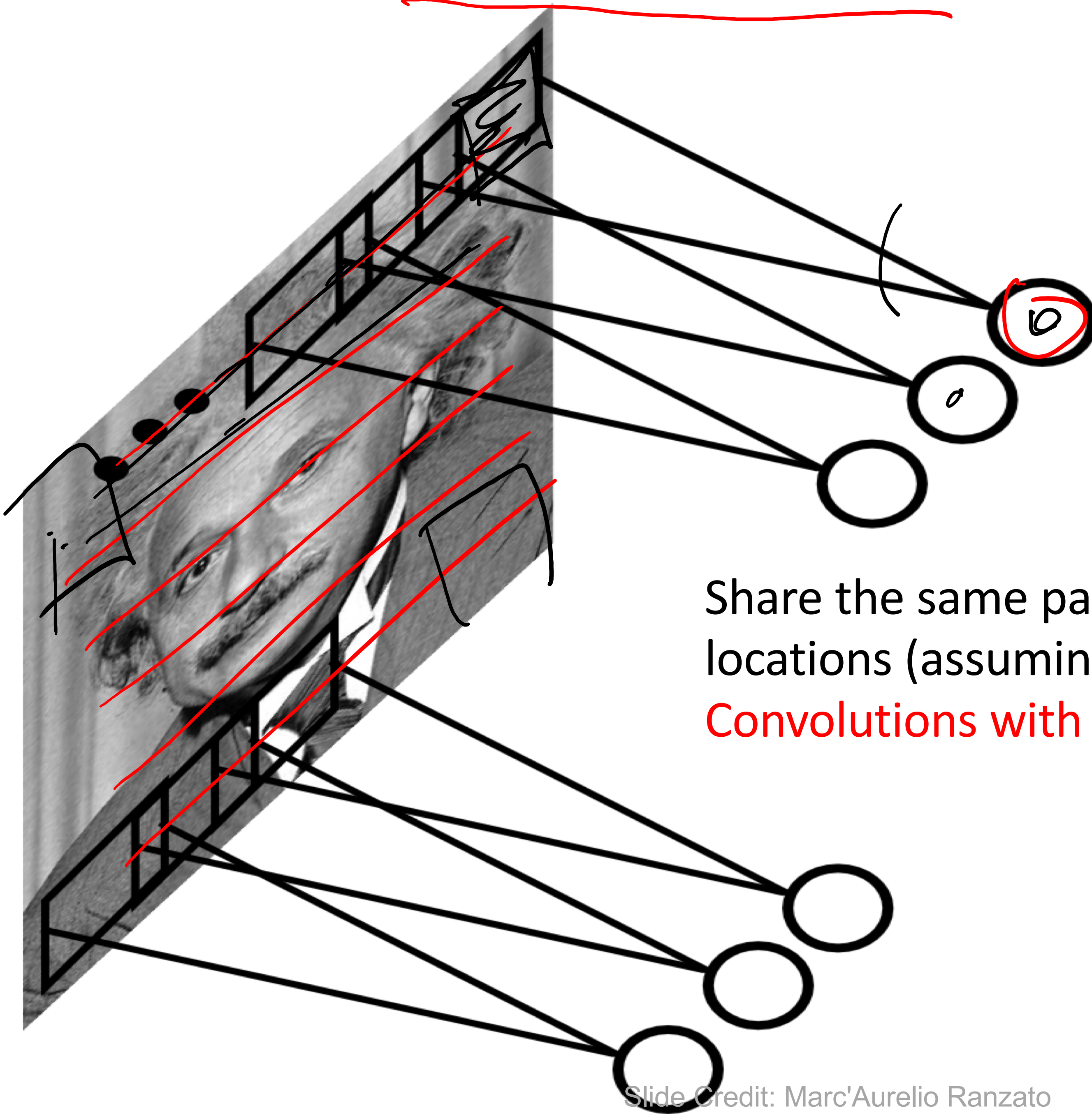
Slide Credit: Marc'Aurelio Ranzato

Locally Connected Layer

STATIONARITY? Statistics is similar at different locations



Convolutional Layer



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

Convolutions!

math → CS → programming

Convolutions for mathematicians

$x(t)$ $y(t)$ $w(t)$

$$y(t) = (x * w)(t)$$

$$= (w * x)(t)$$

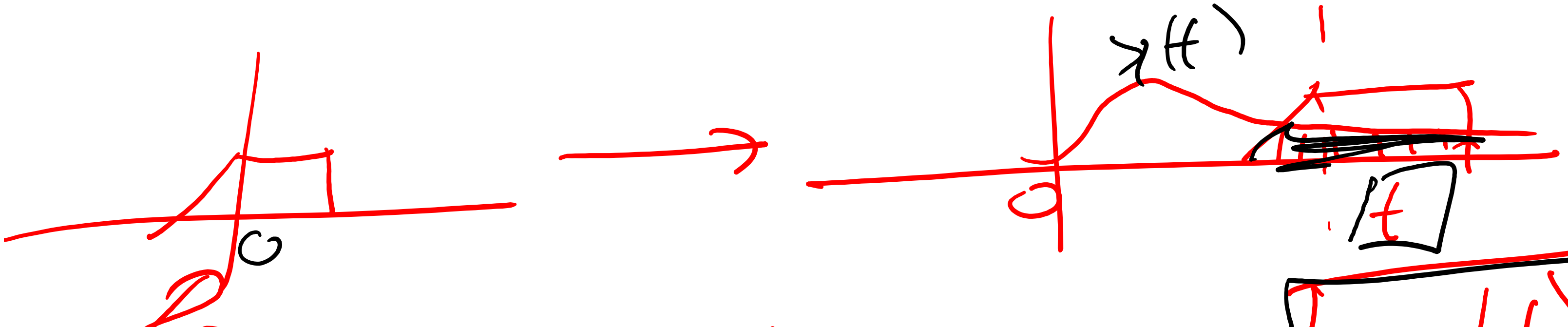
$$= \int_{a=-\infty}^{\infty} x(t-a)\, w(a)\, da$$

$$= \int_{a=-\infty}^{\infty} x(a)\, w(t-a)\, da$$

Convolutions for mathematicians

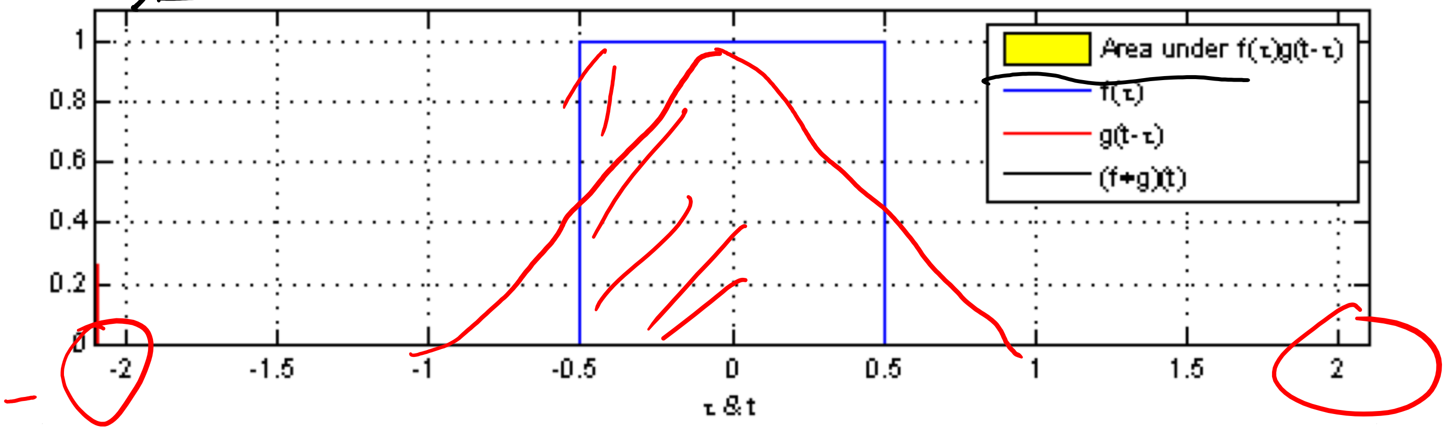
$w(a) \longrightarrow w(-a)$

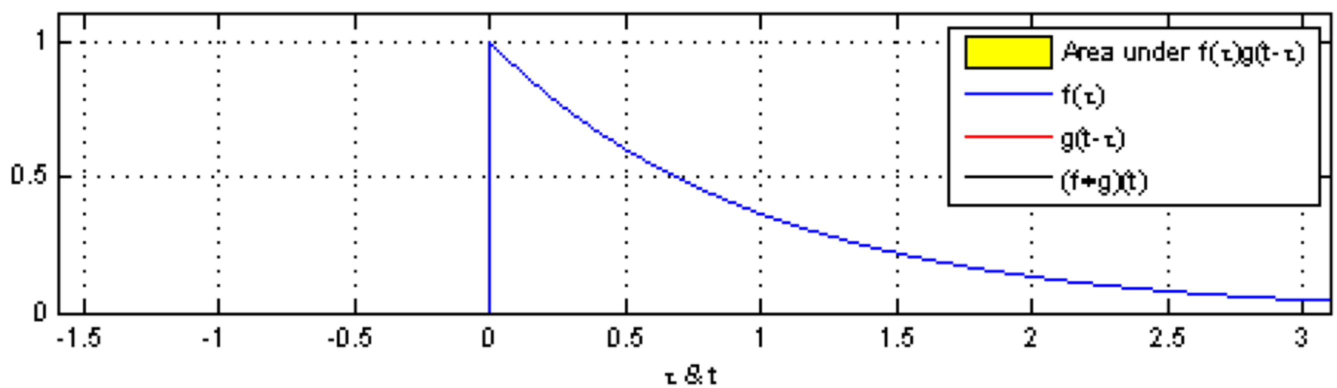
$w(-a) \xrightarrow[\text{translate}]{} w(-(a-t))$



$$\int_{a=-\infty}^{\infty} x(t)\, w(t-a) = \boxed{y(t)}$$

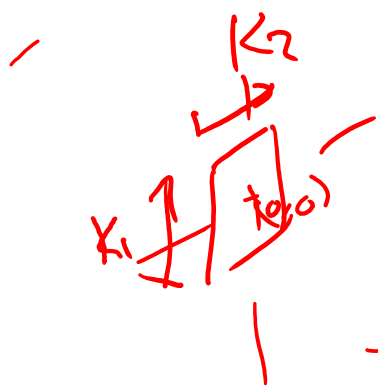
$$y(t_1, t_2) = \boxed{\int_{a=-\infty}^{\infty} \int_{b=-\infty}^{\infty}} x(\underline{t_1 - a}, \underline{t_2 - b})\, w(a,b)\, da\, db$$





Convolutions for computer scientists

$$y[r,c] = \sum_{\boxed{a=-\infty}}^{\infty} \sum_{b=-\infty}^{\infty} x[r-a,\ c-b]\, w[a,b]$$

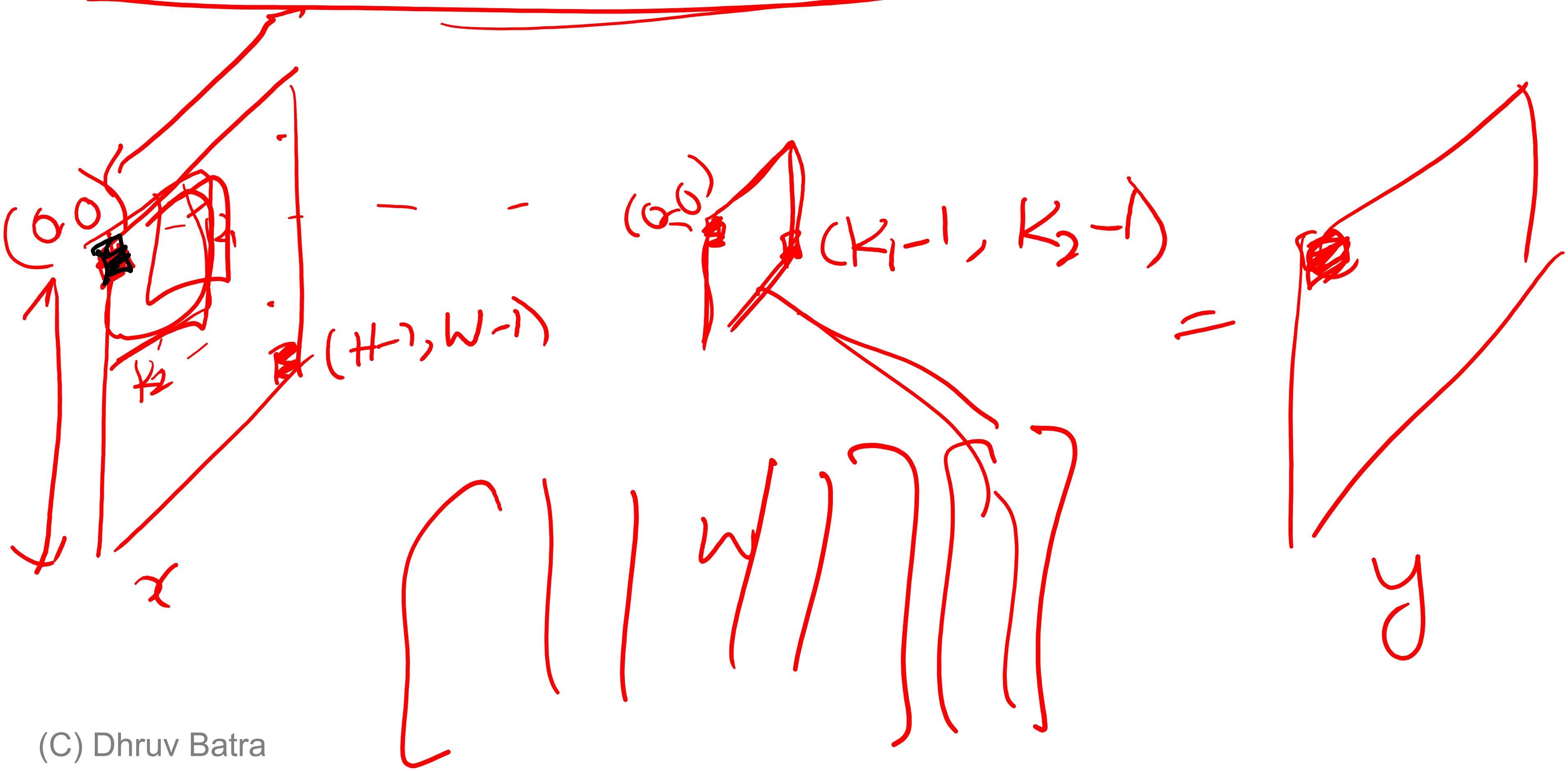


$$y[r,c] = \sum_{a=-\frac{K_1-1}{2}}^{\frac{K_1-1}{2}} \sum_{b=-\frac{K_2-1}{2}}^{\frac{K_2-1}{2}} [\quad]$$

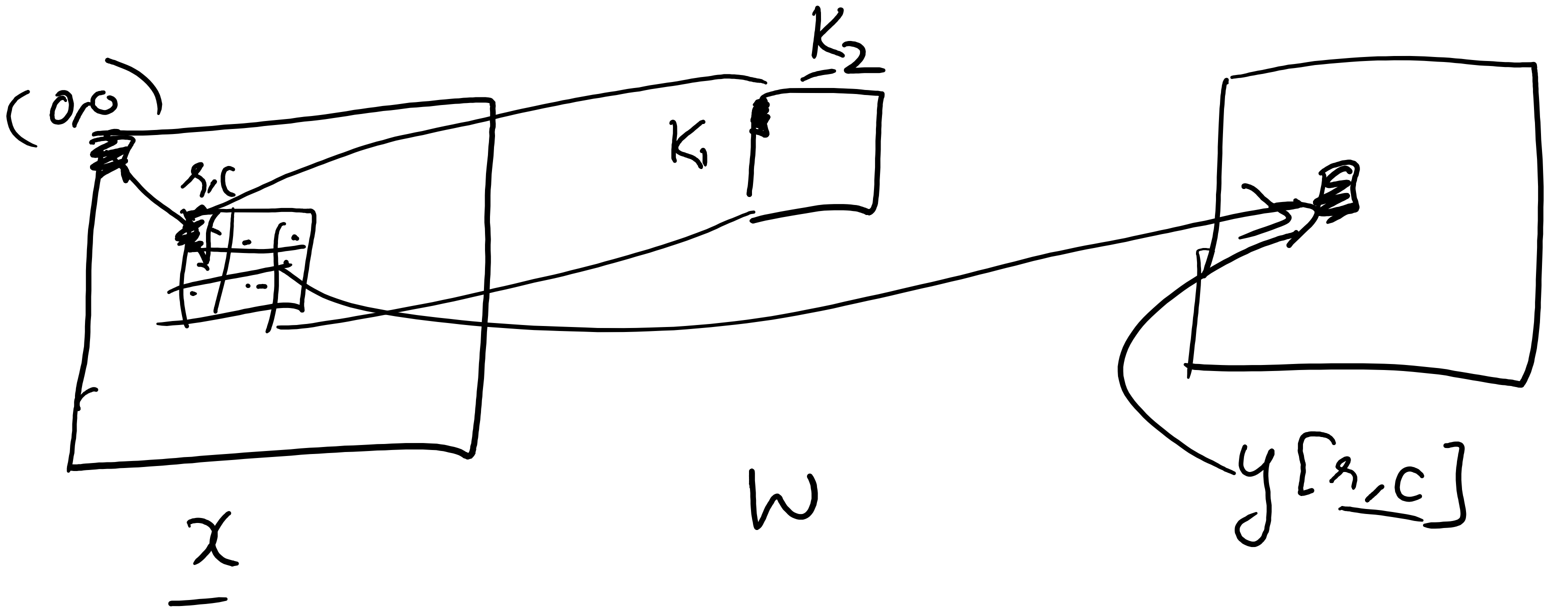
Convolutions for computer scientists

Convolutions for programmers

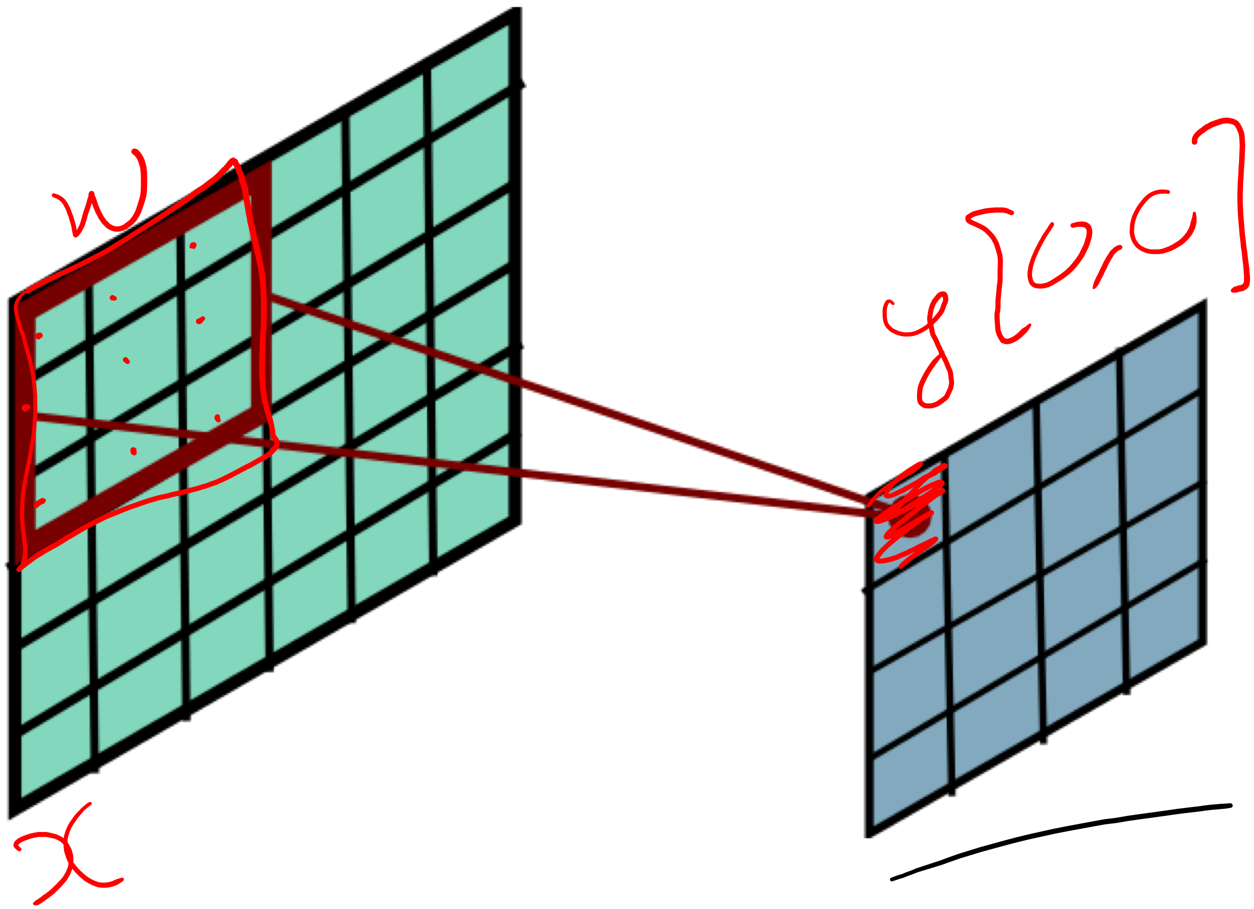
$$y[r,c] = \sum_{a=0}^{K_1-1} \sum_{b=0}^{K_2-1} x[r+a,\, c+b]\; w[a,b]$$



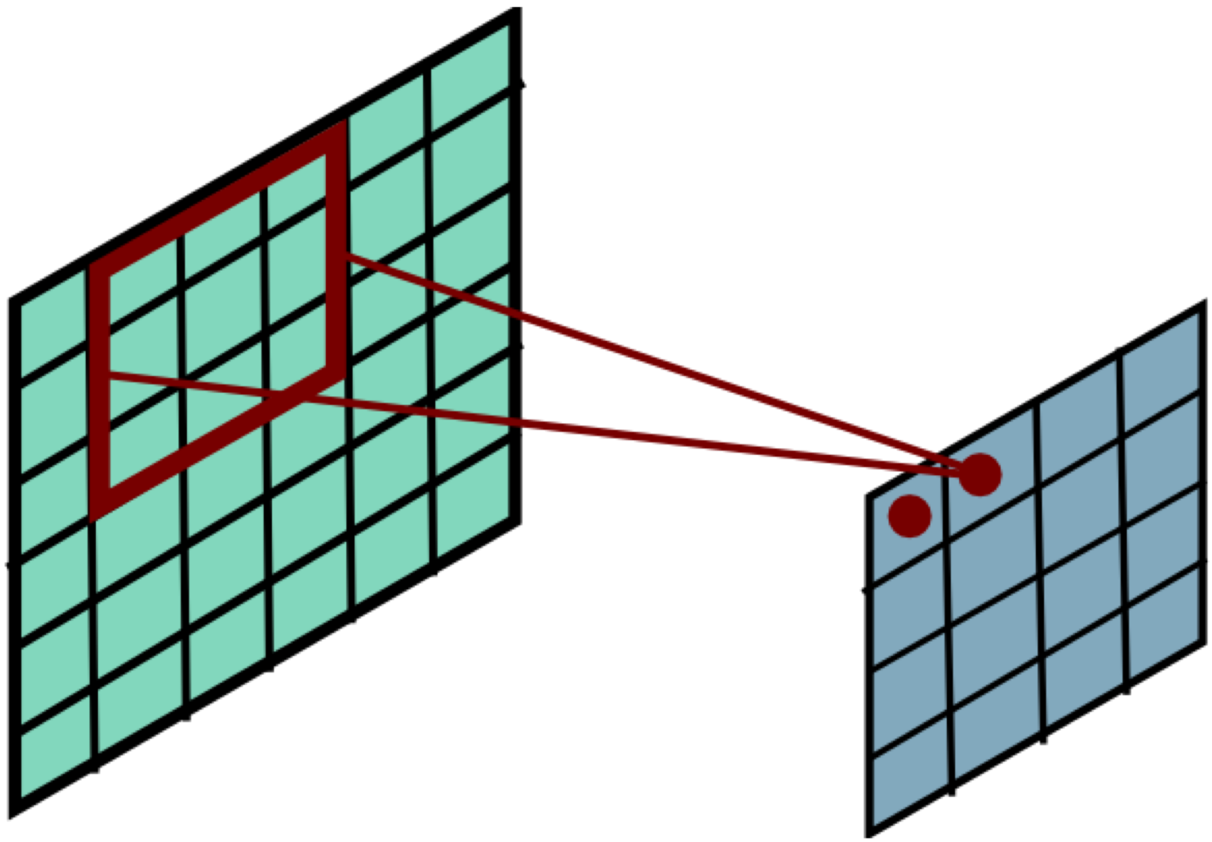
Convolutions for programmers



Convolutional Layer

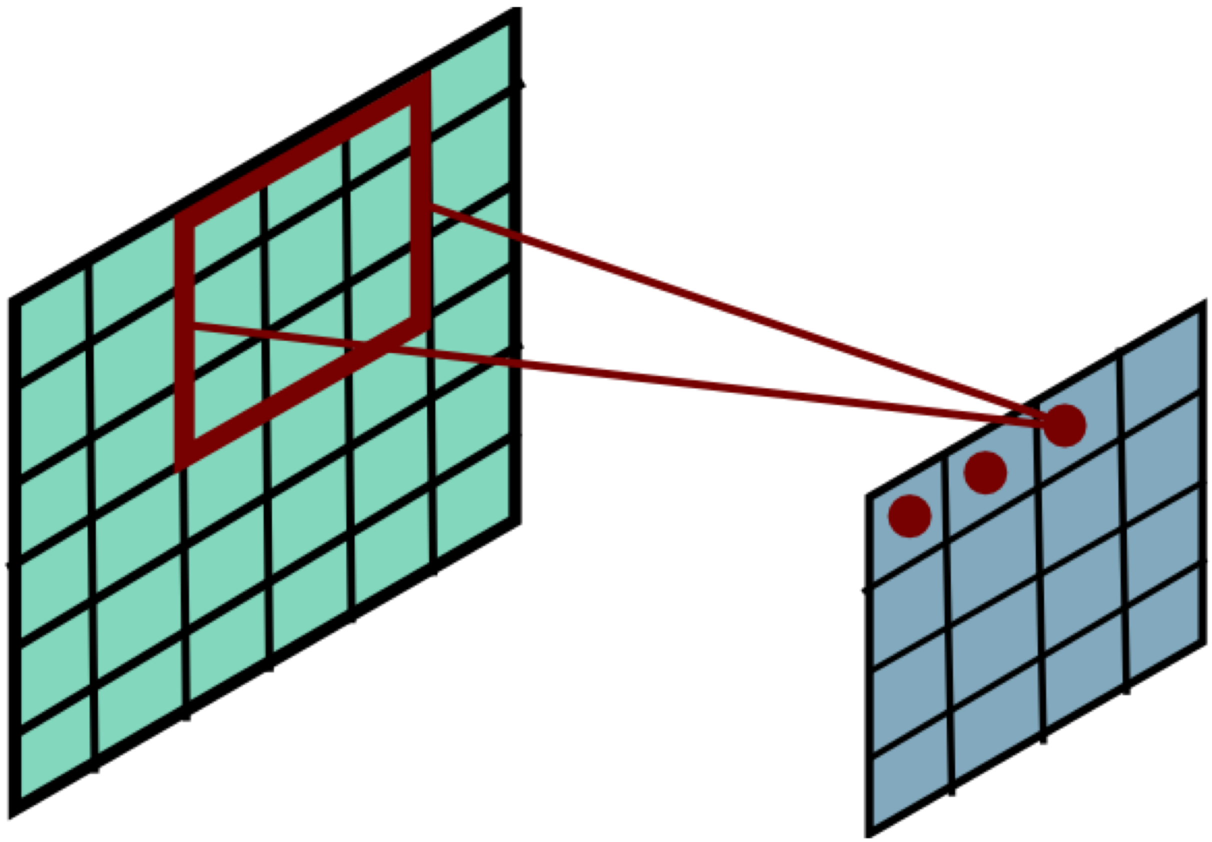


Convolutional Layer

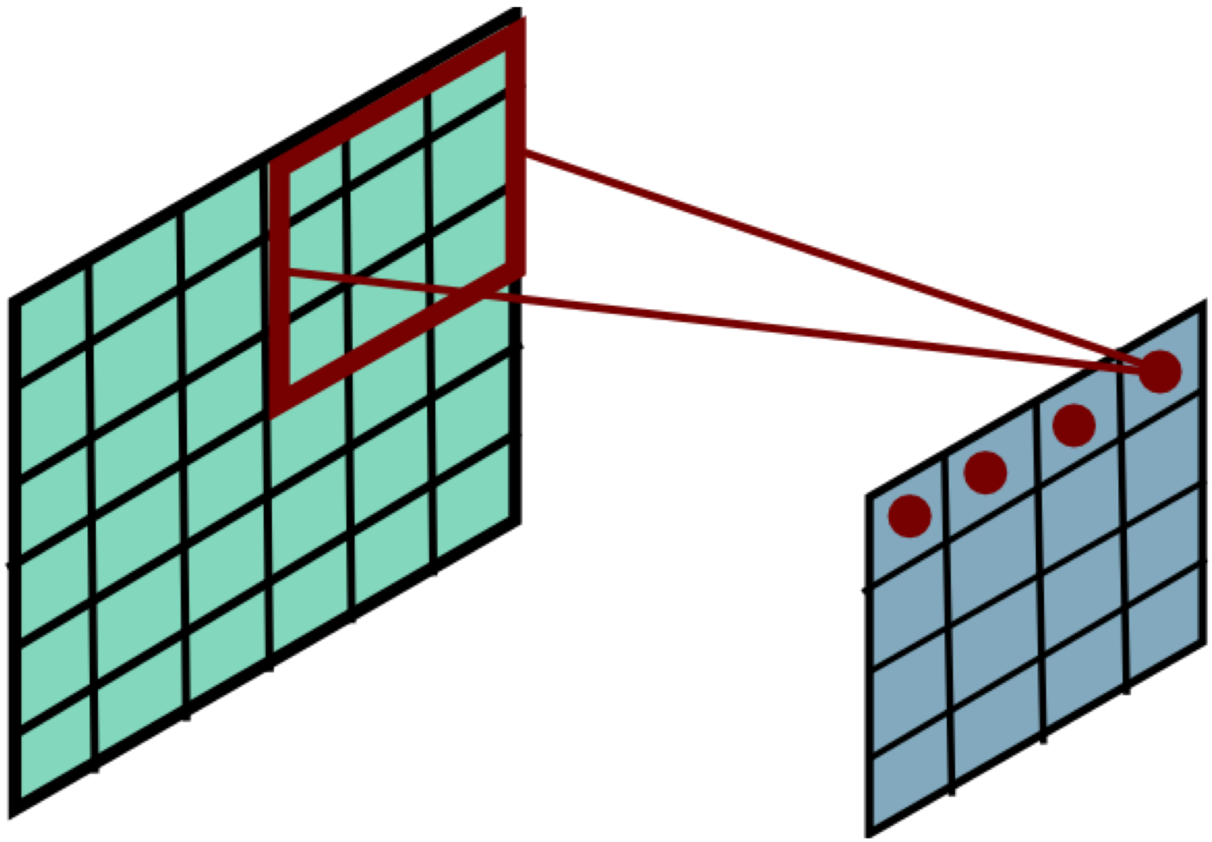


Slide Credit: Marc'Aurelio Ranzato

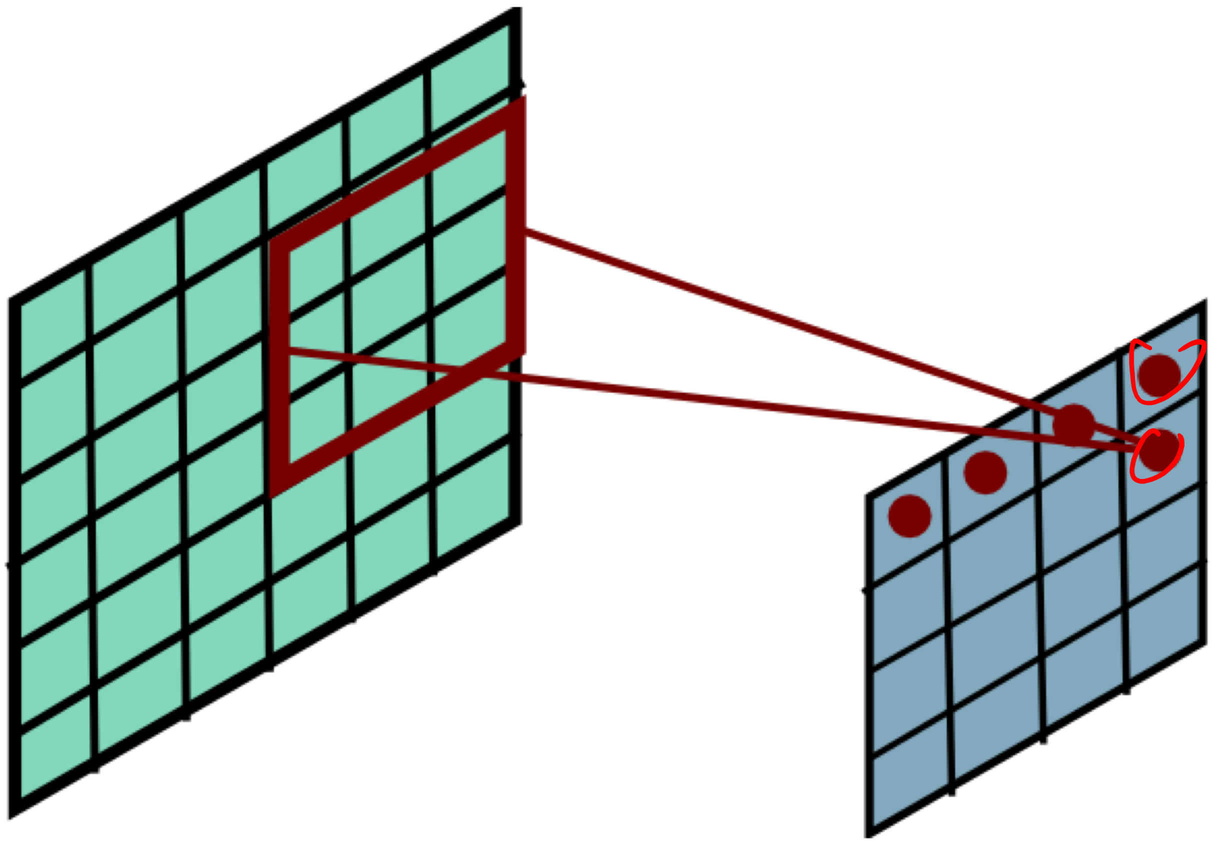
Convolutional Layer



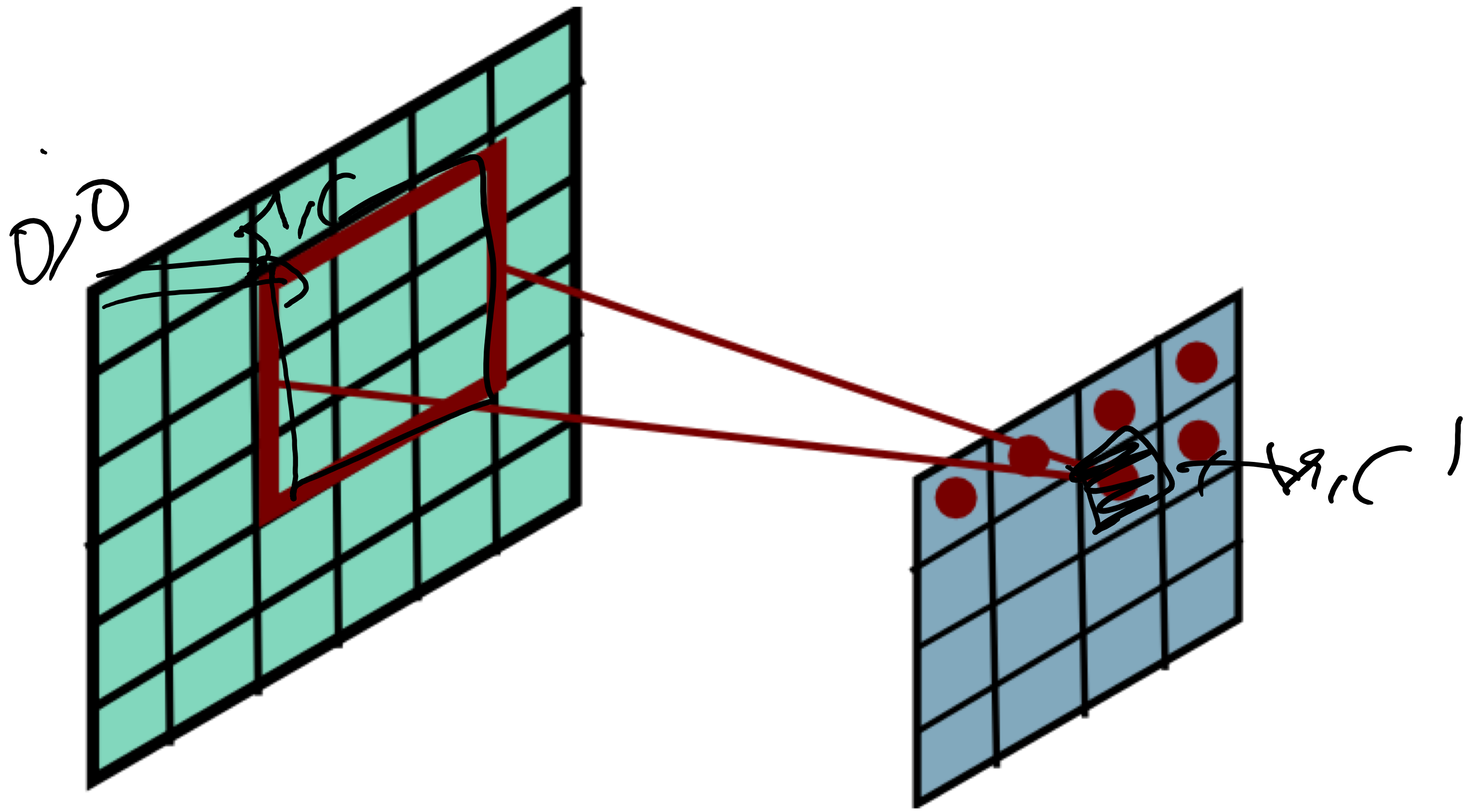
Convolutional Layer



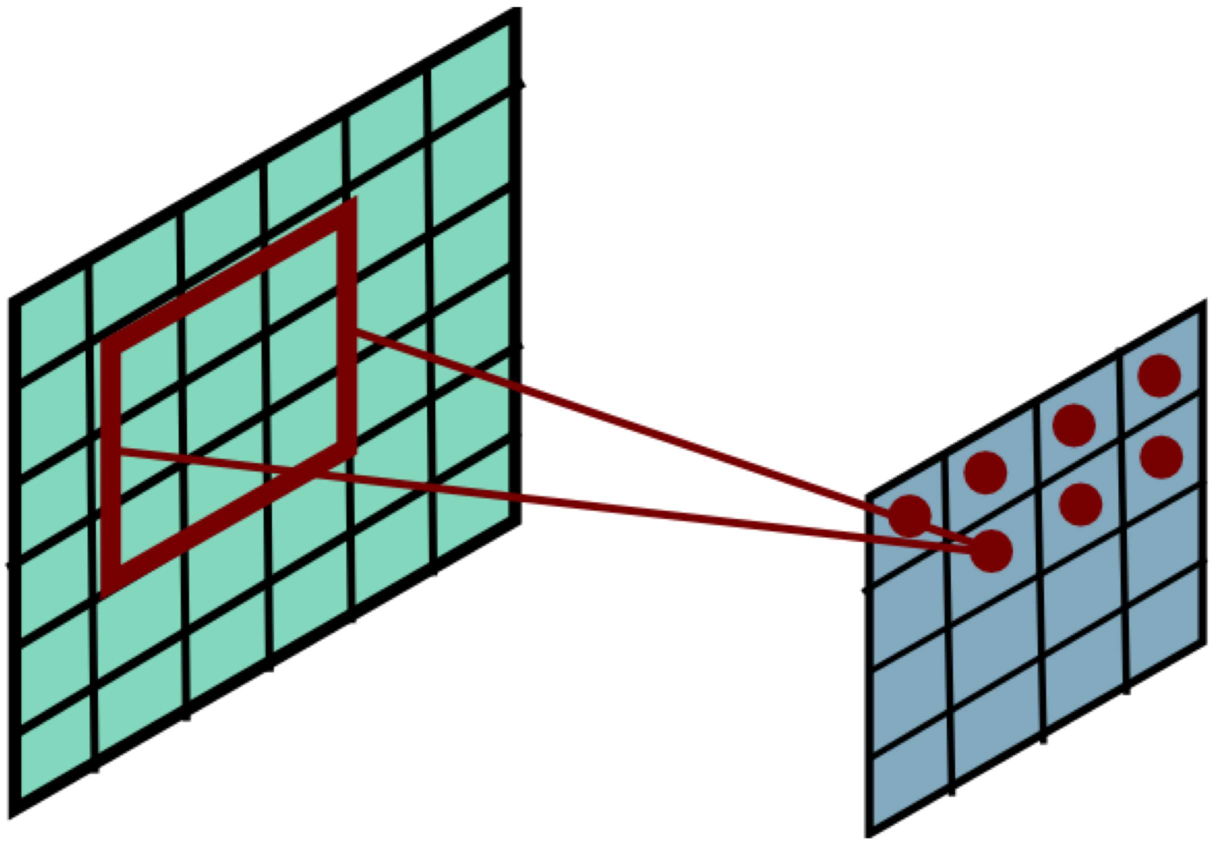
Convolutional Layer



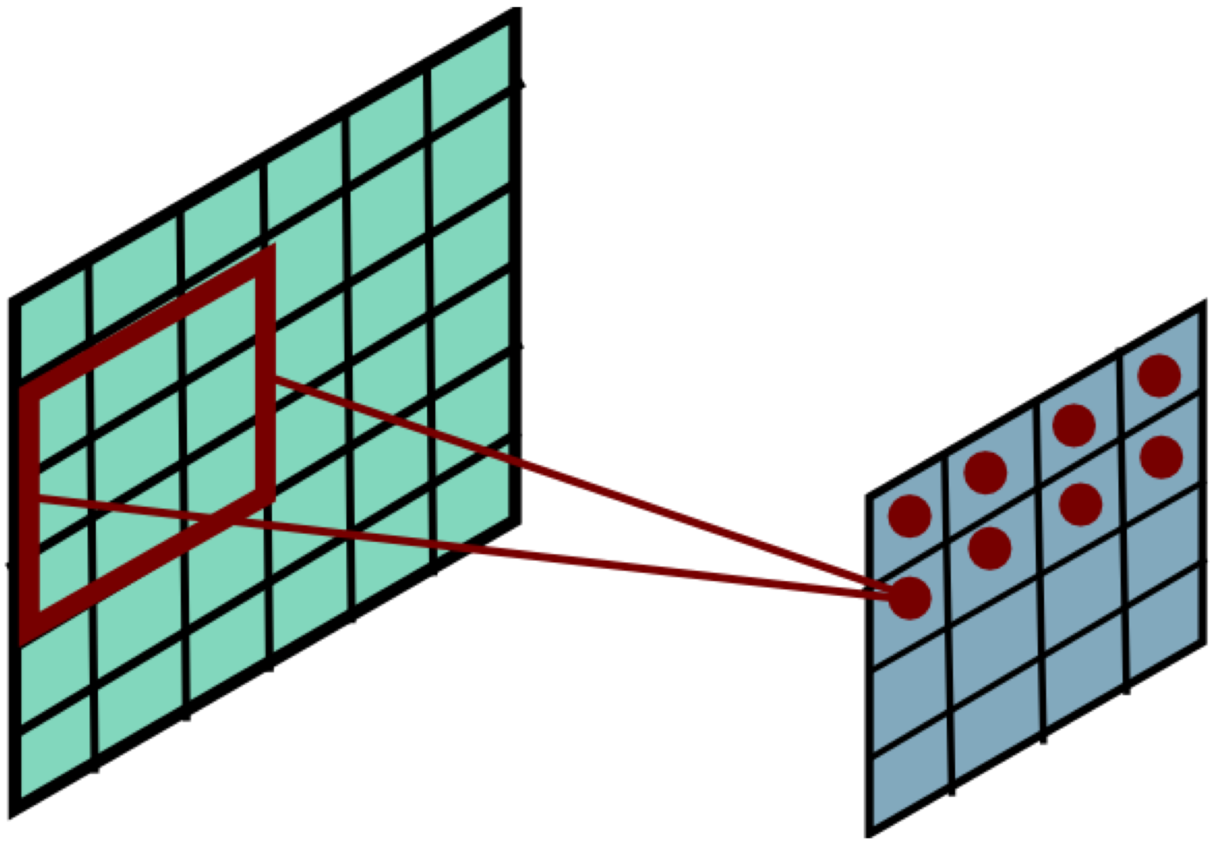
Convolutional Layer



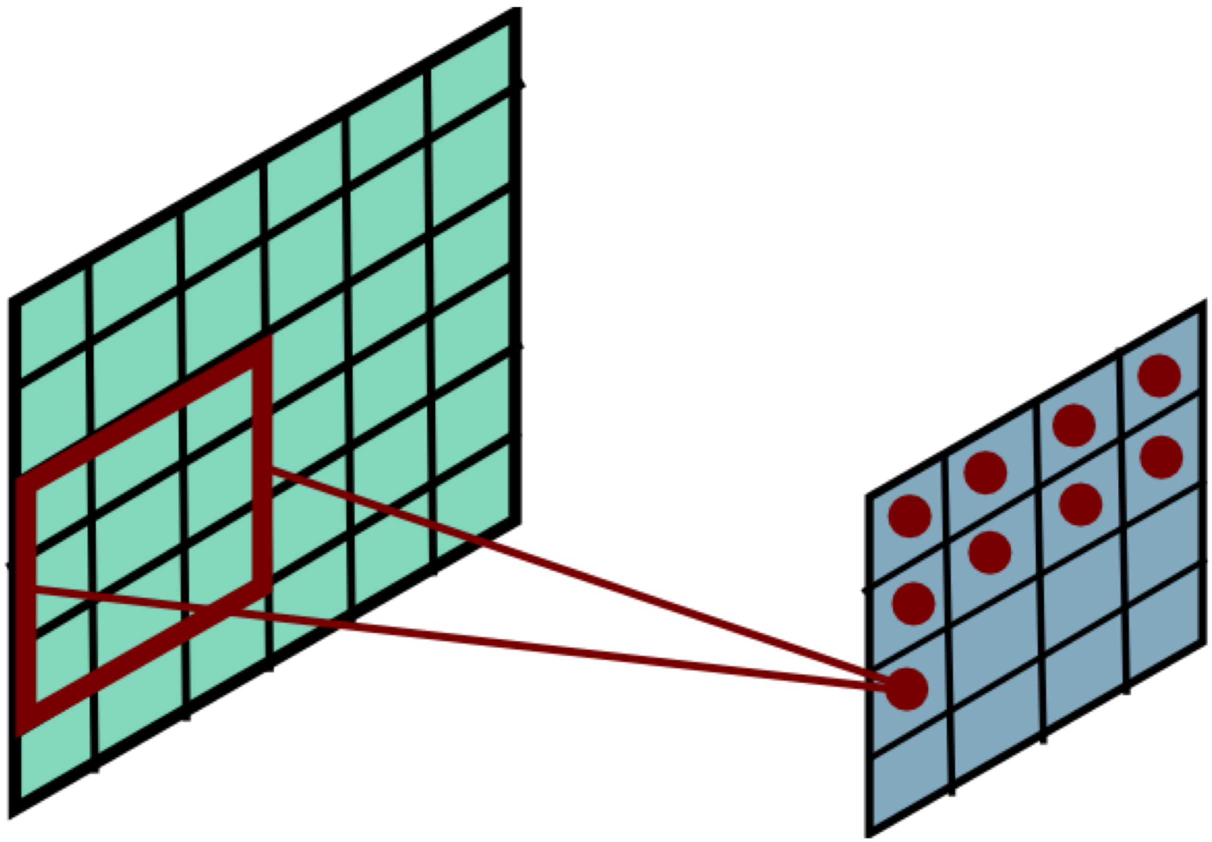
Convolutional Layer



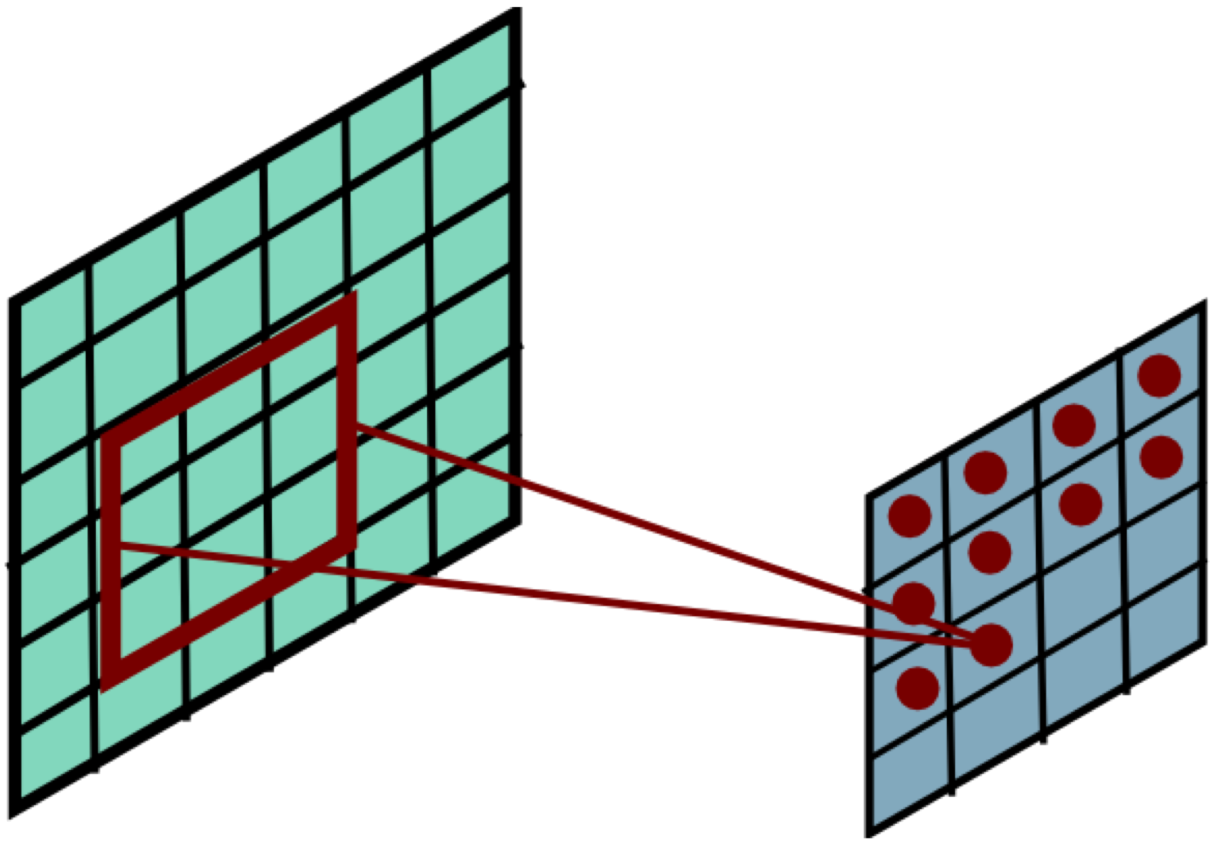
Convolutional Layer



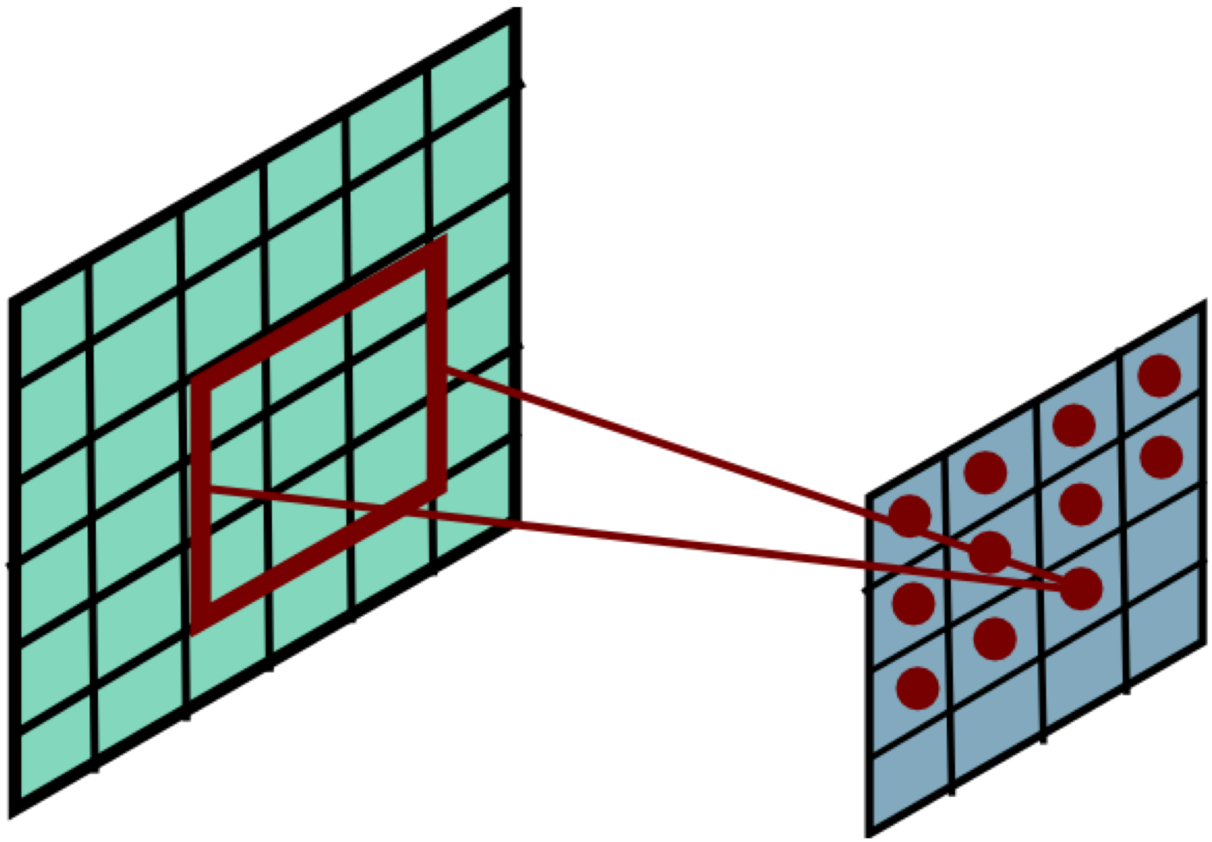
Convolutional Layer



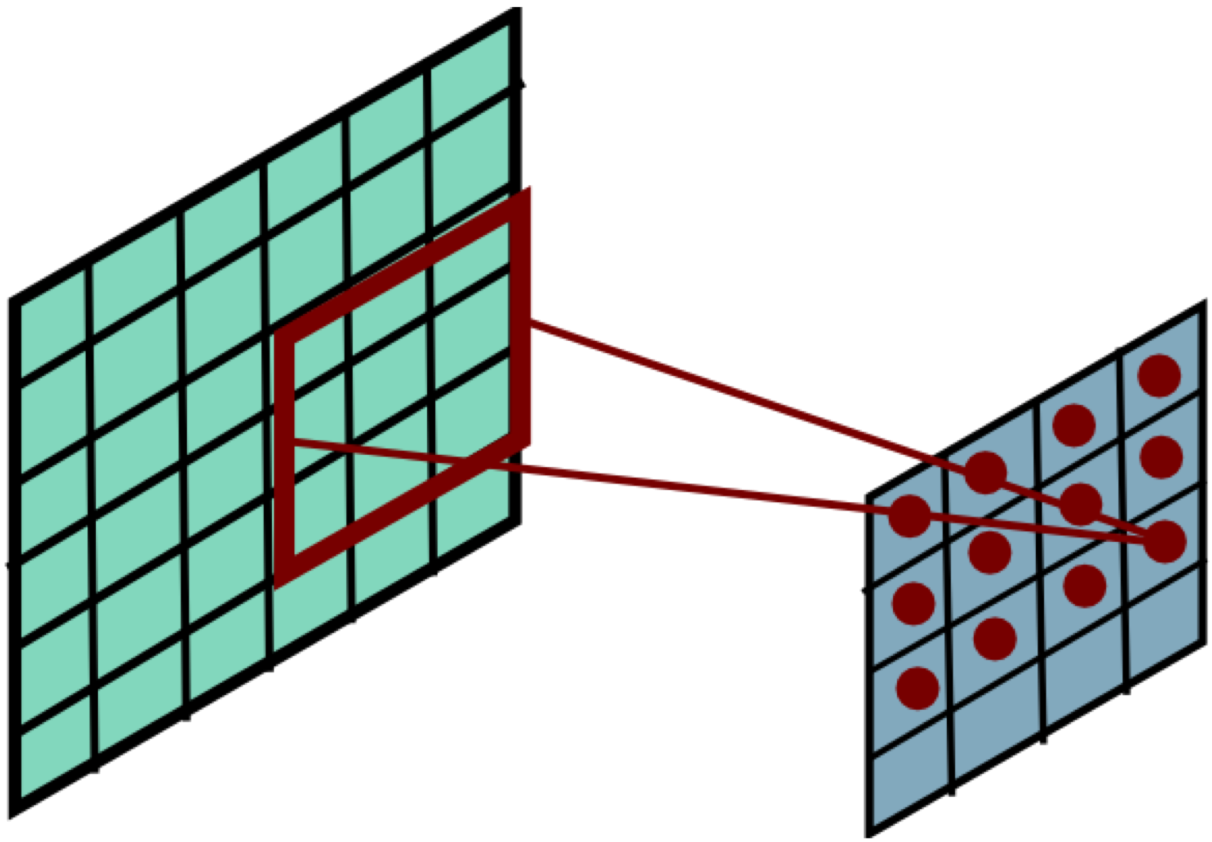
Convolutional Layer



Convolutional Layer

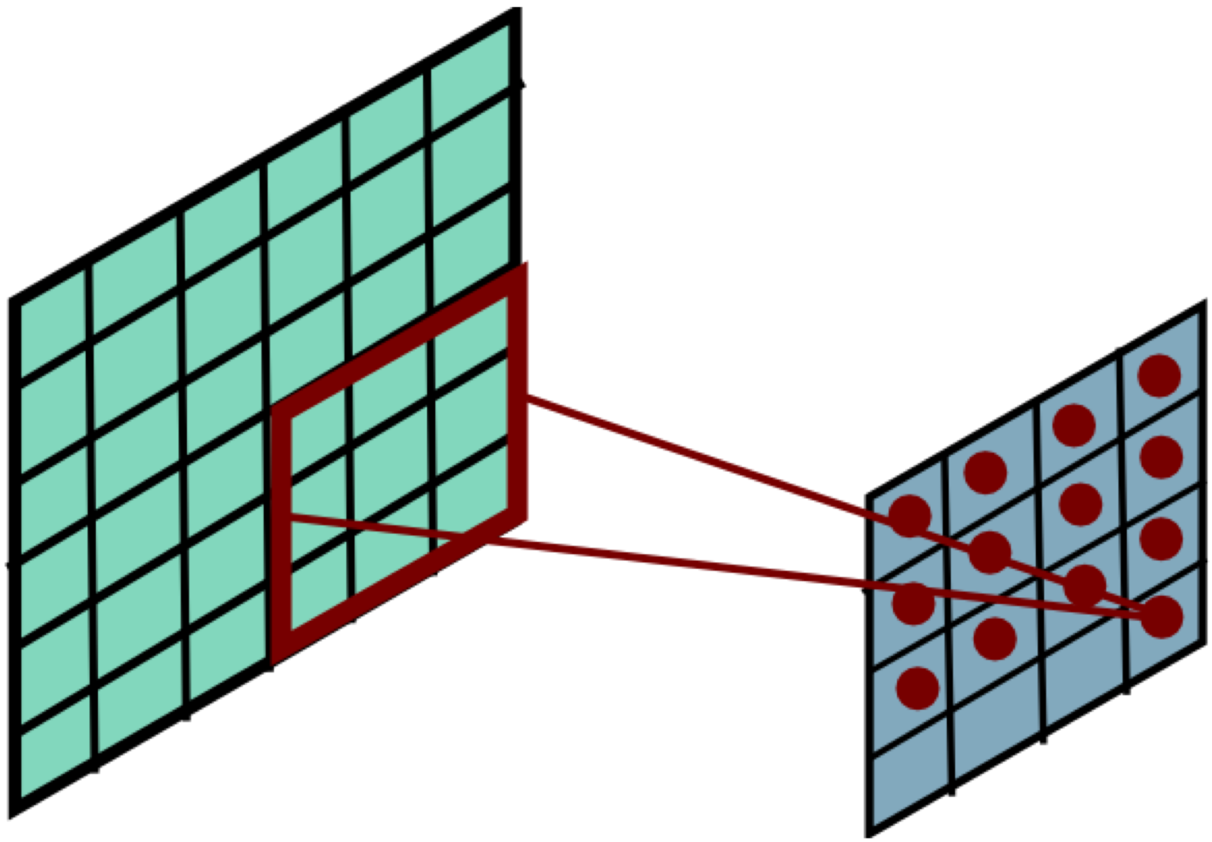


Convolutional Layer



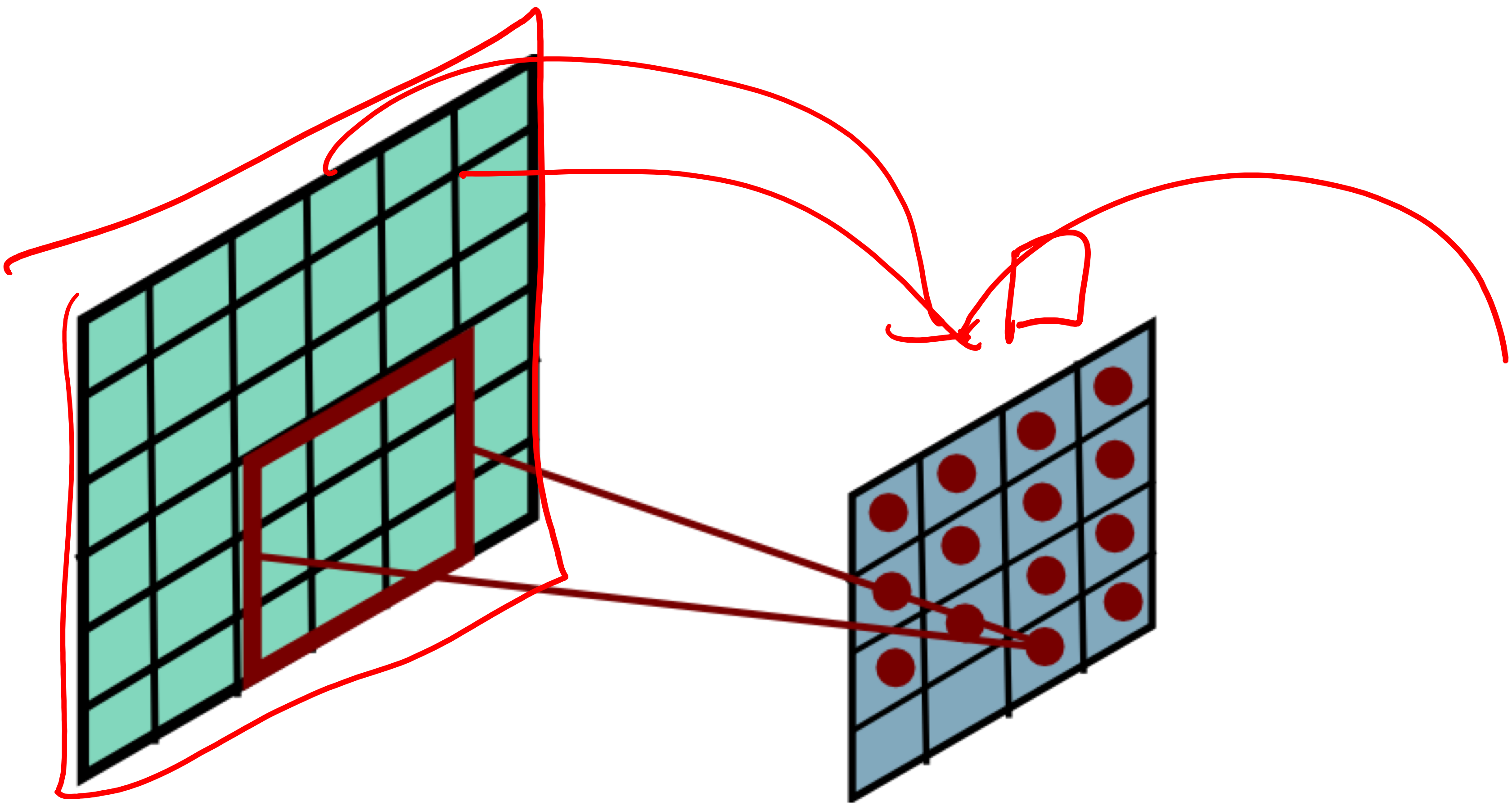
Slide Credit: Marc'Aurelio Ranzato

Convolutional Layer

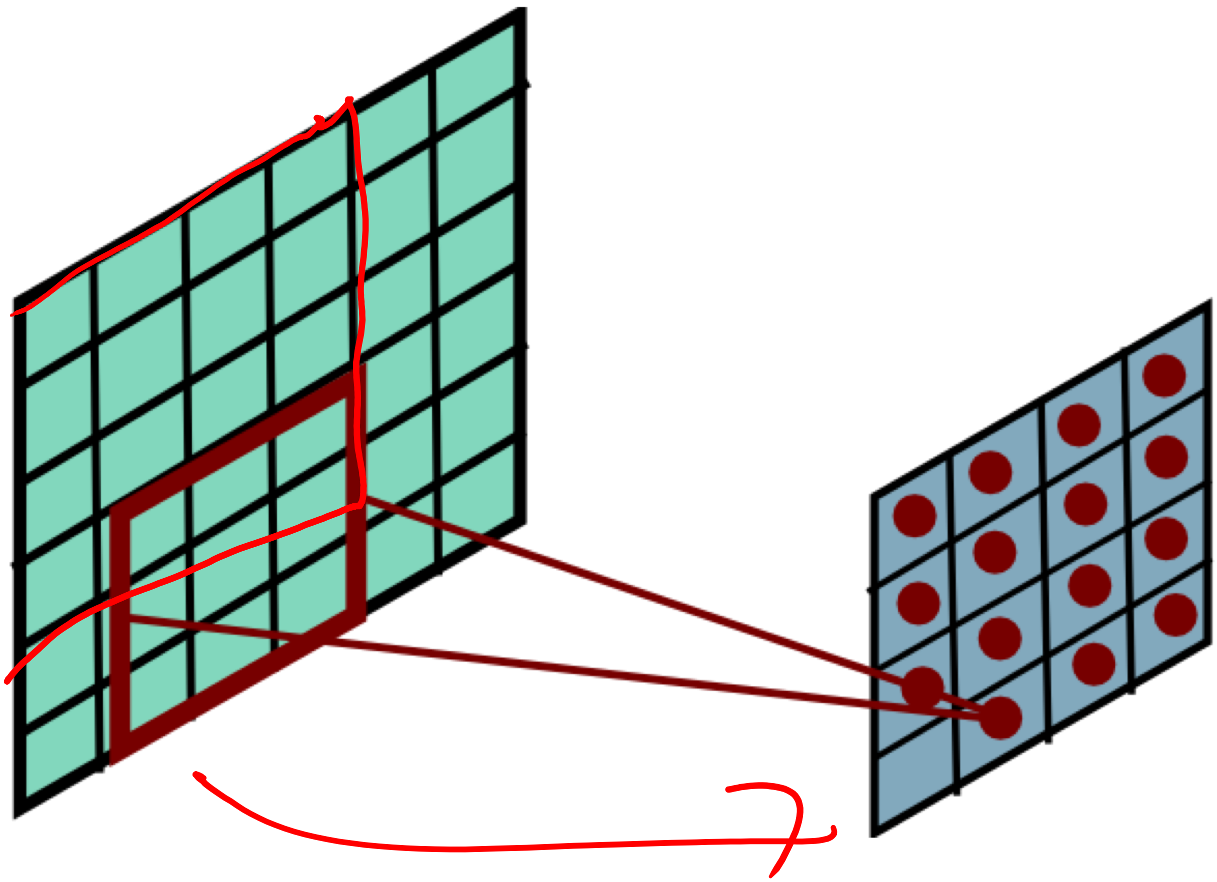


Slide Credit: Marc'Aurelio Ranzato

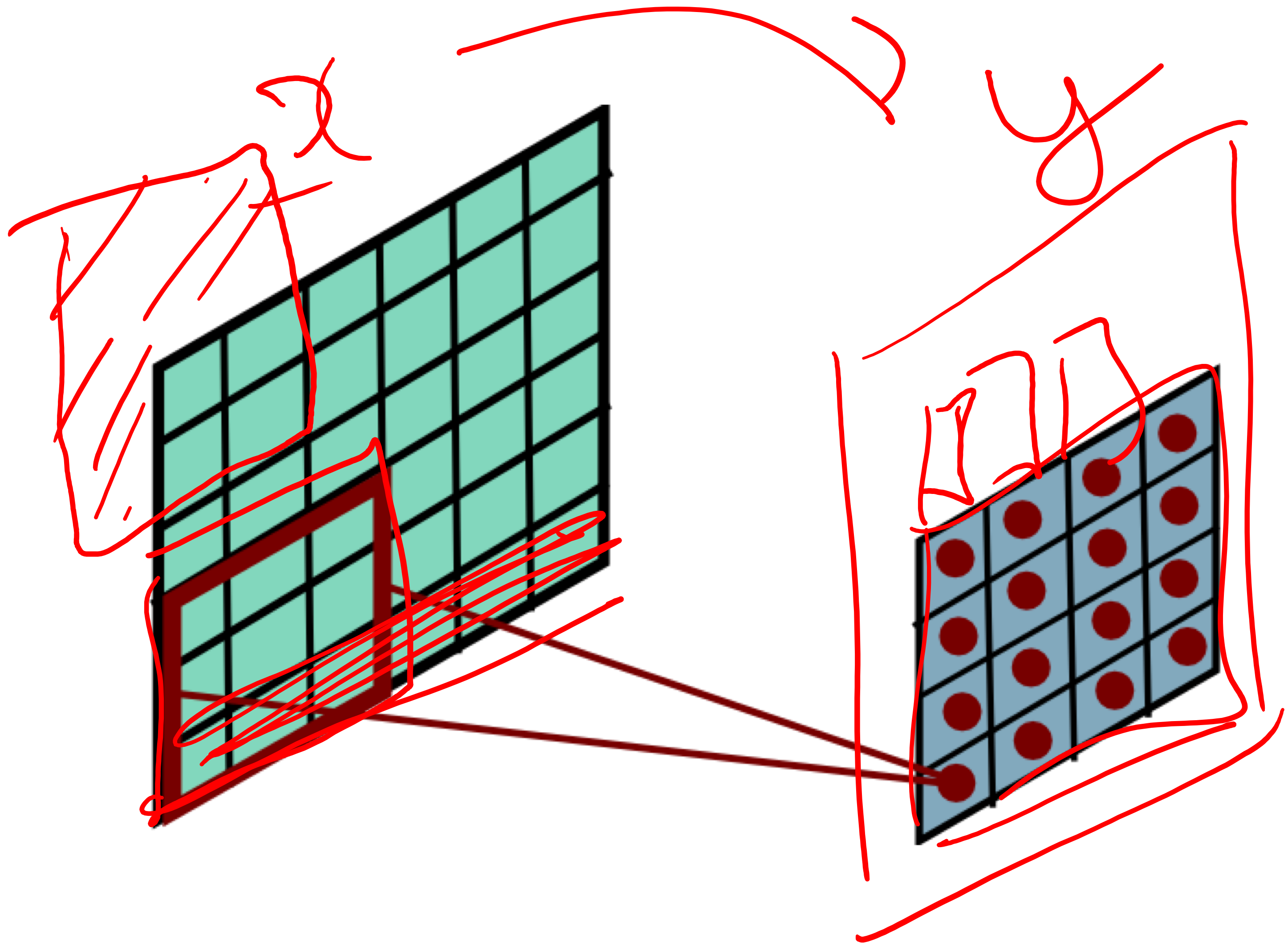
Convolutional Layer



Convolutional Layer



Convolutional Layer



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

Plan for Today

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

Convolution Explained

- http://setosa.io/ev/image-kernels/
- https://github.com/bruckner/deepViz

Convolutional Layer

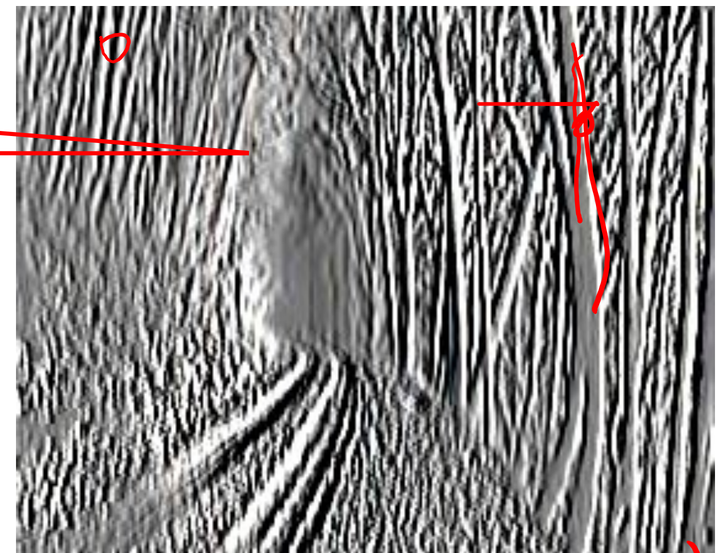
x

w

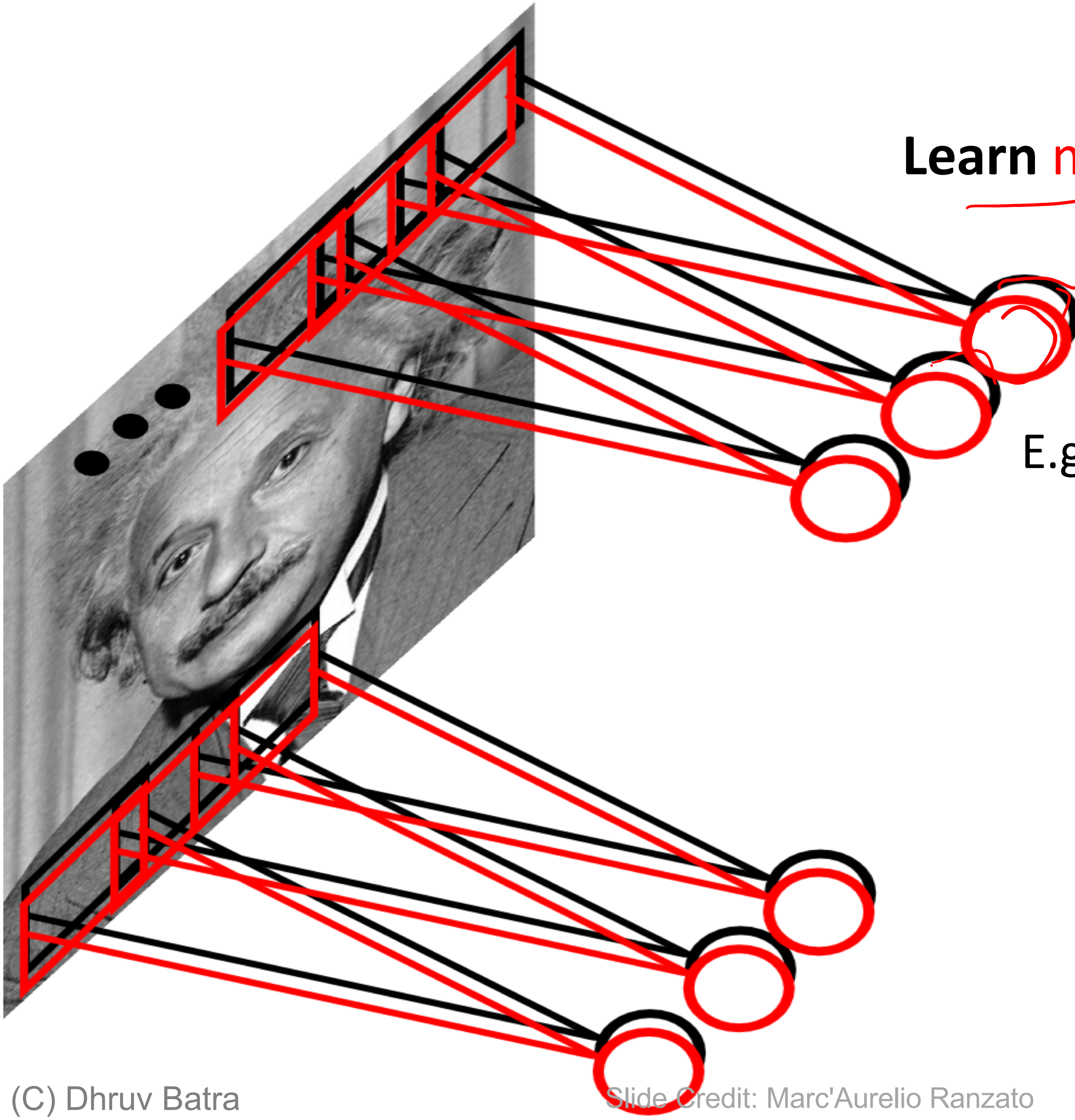
$$* \begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix} =$$

y

activation



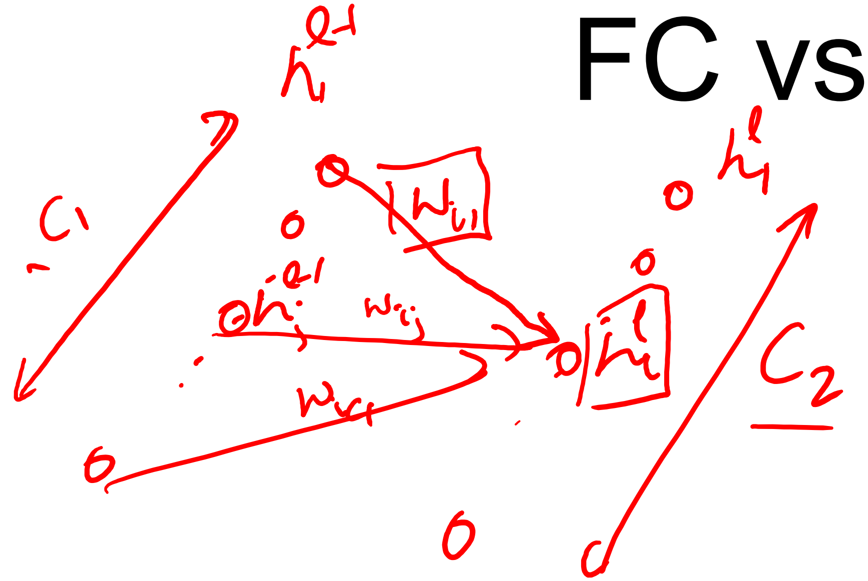
Convolutional Layer



Learn multiple filters.

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

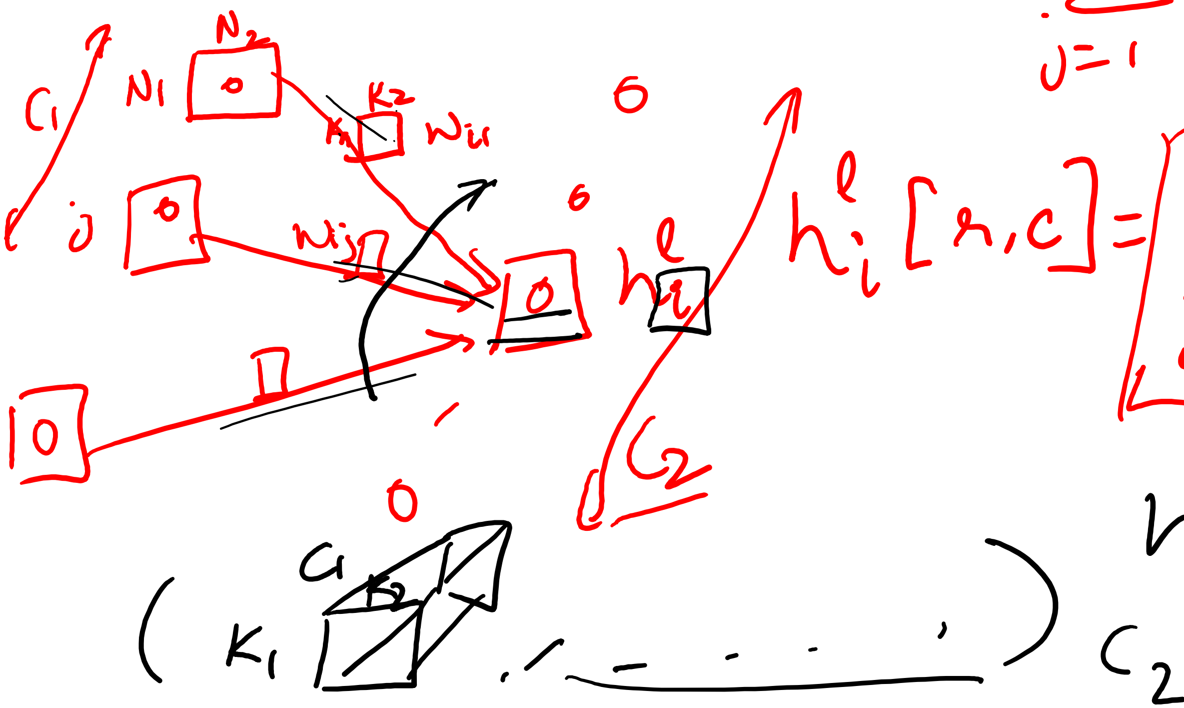
FC vs Conv Layer



$$h_i^l = \sum_{j=1}^{C_1} \underbrace{h_j^{l-1} \cdot w_{ij}}_{\text{Scalar product}} + b_i$$

conv

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



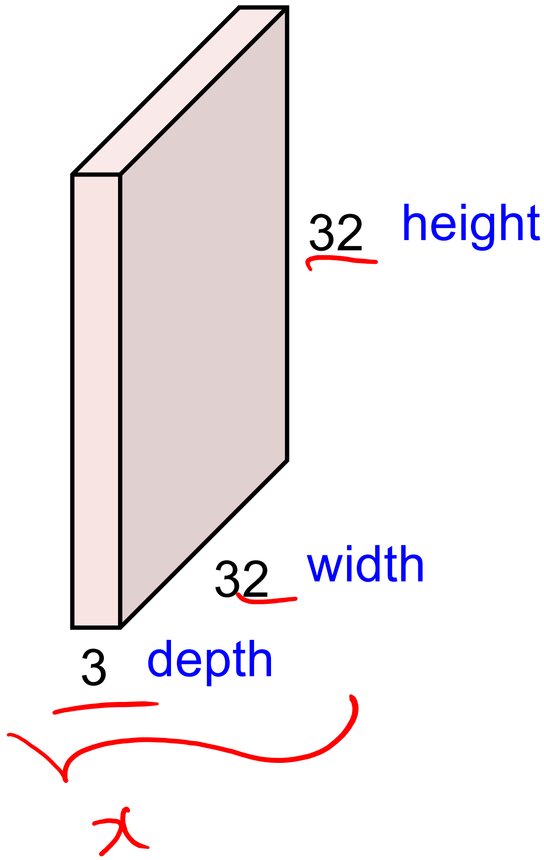
$$h_i^l[r,c] = \sum_{j=1}^{C_1} \sum_{a=0}^{K_1-1} \sum_{b=0}^{K_2-1} h_j^{l-1}[r+a, c+b]\, w[a,b]$$

$$W \; \boxed{(K_1 \times K_2 \times C_1 \times C_2)}$$

FC vs Conv Layer

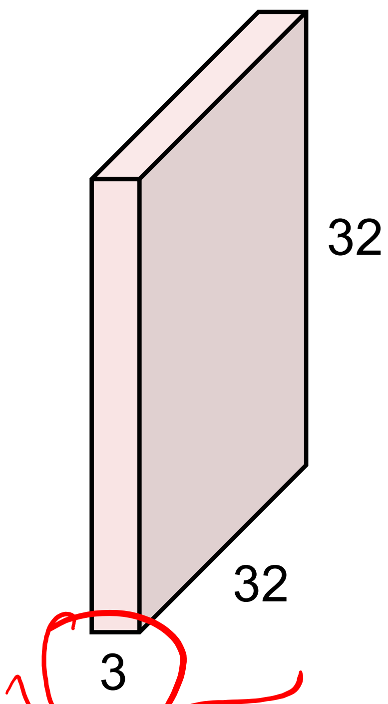
Convolution Layer

32x32x3 image -> preserve spatial structure



Convolution Layer

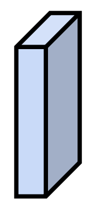
32x32x3 image



x

k_1 k C

5x5x3 filter



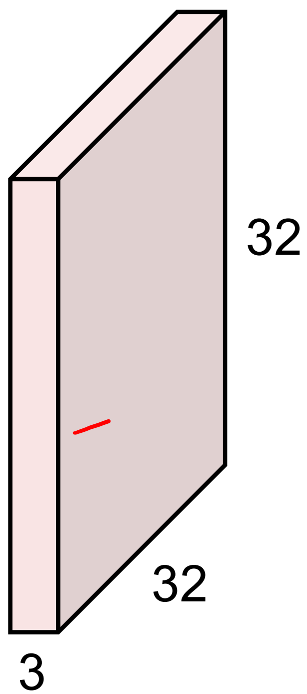
w

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

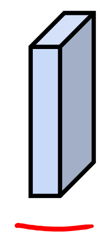
Convolution Layer

Filters always extend the full depth of the input volume

32x32x3 image



5x5x3 filter



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

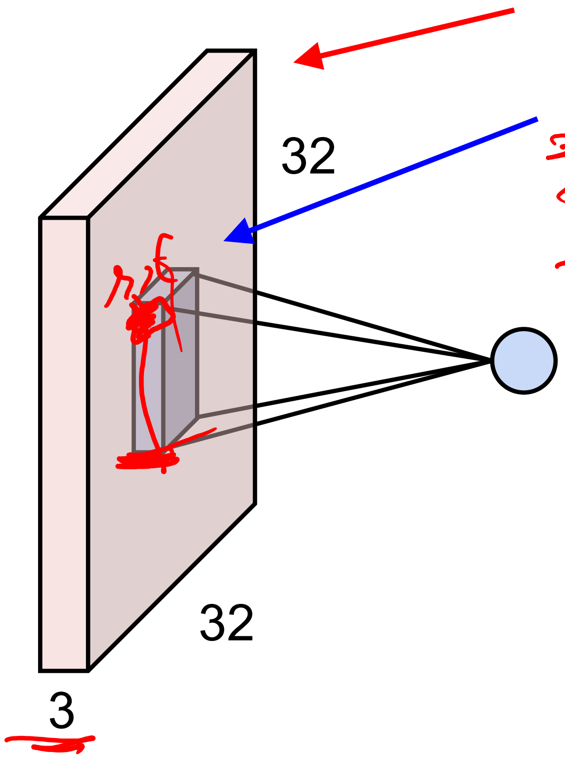
Convolution Layer

32x32x3 image

5x5x3 filter w

3x3

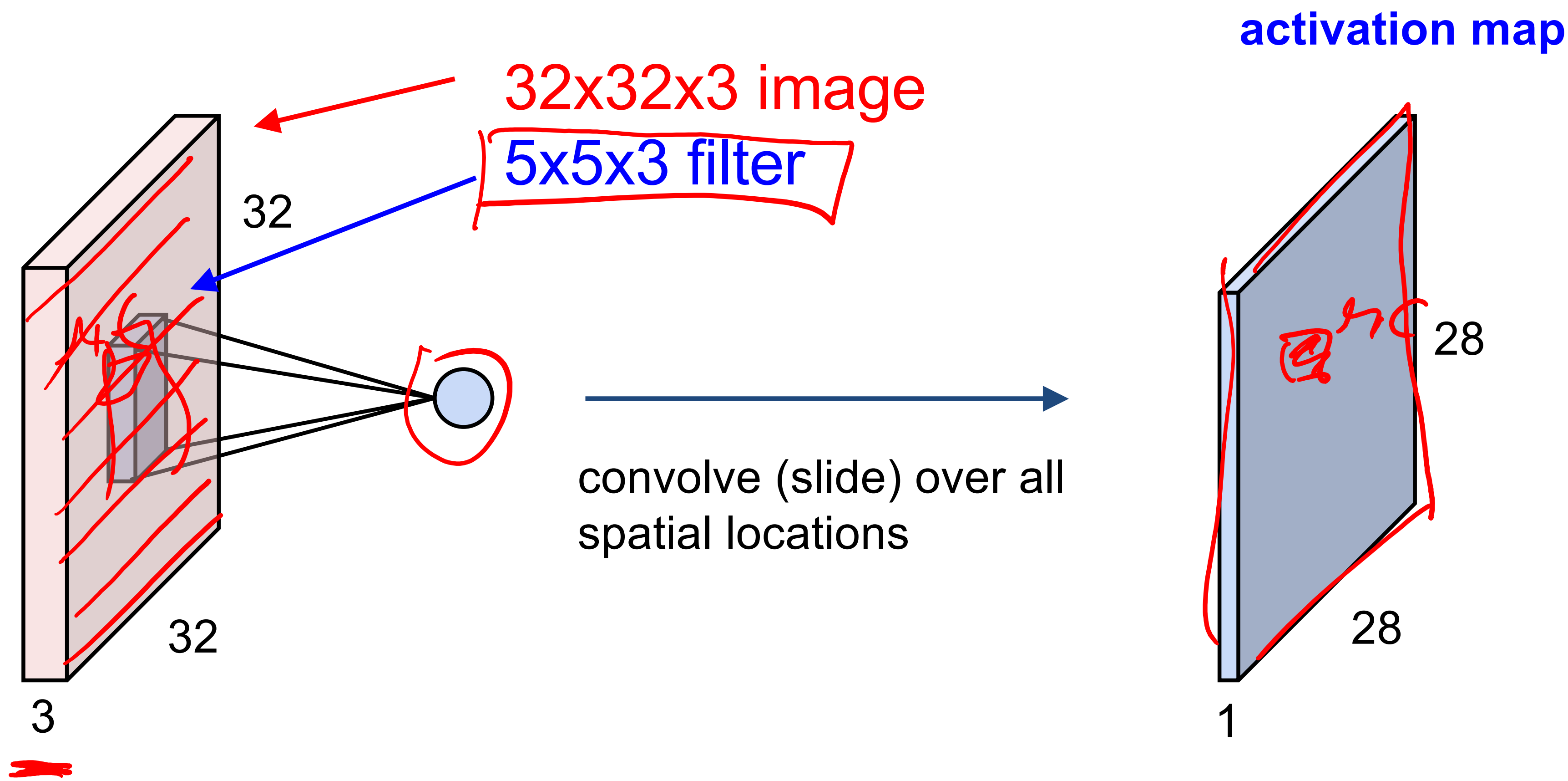
7x7



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)

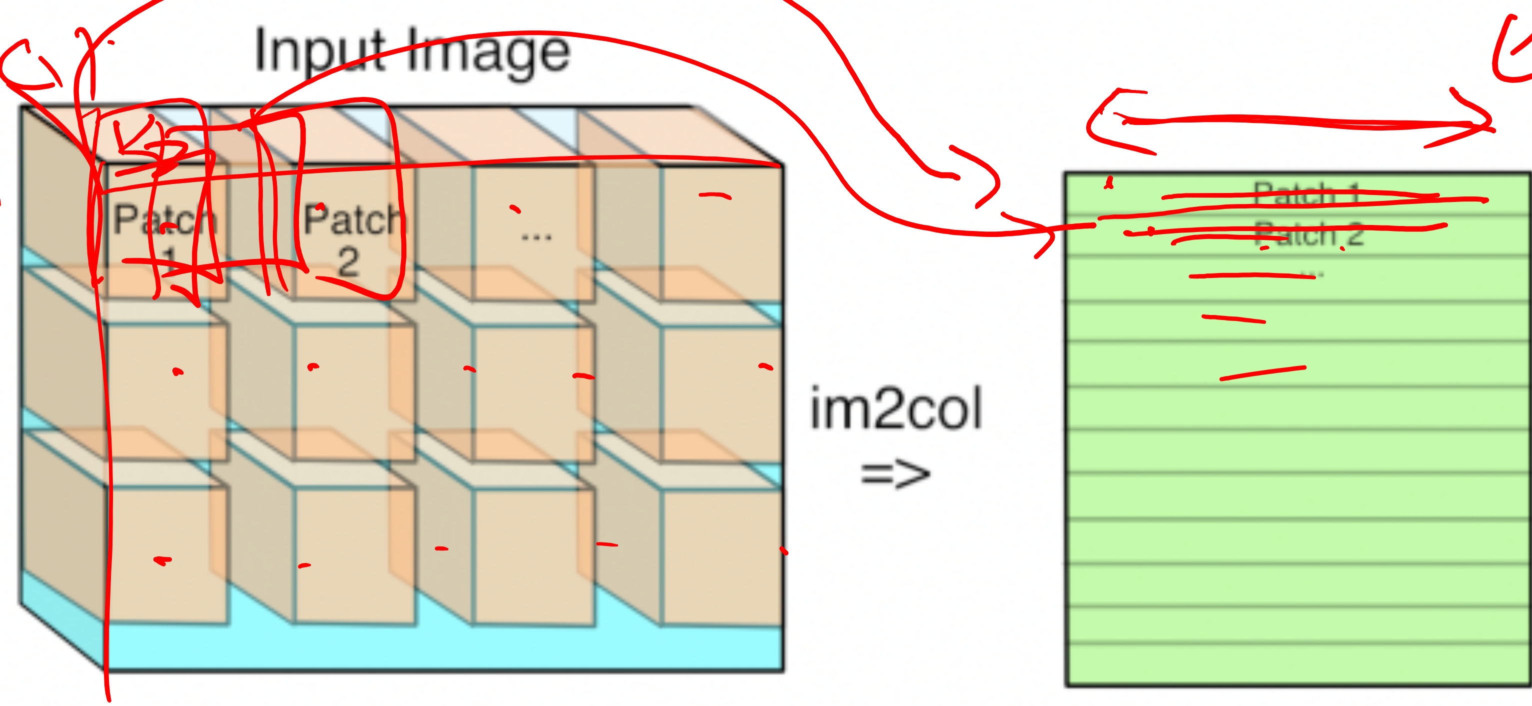
$$w^T x + b$$

Convolution Layer



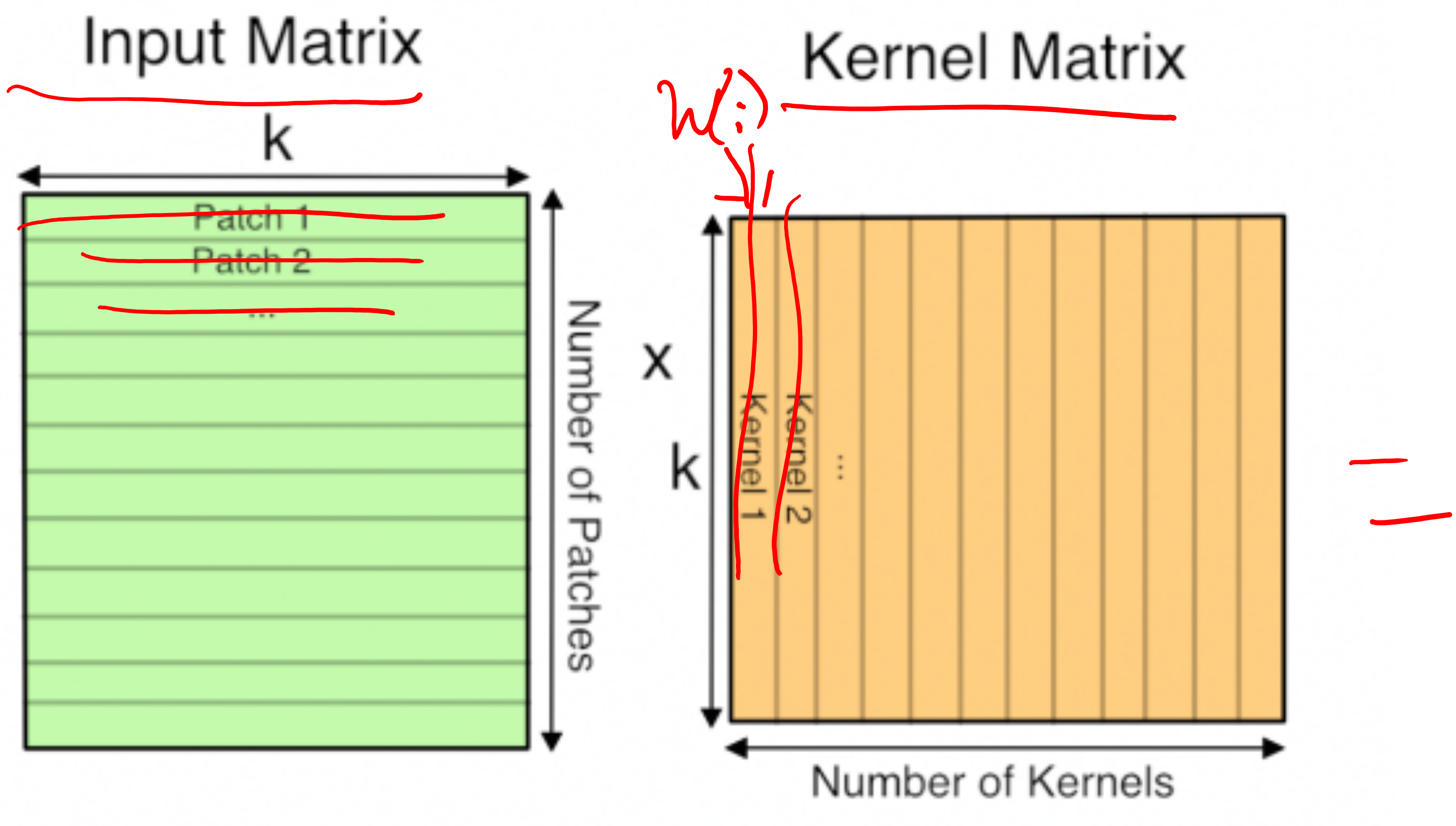
Slide Credit: Fei-Fei Li, Justin Johnson, Serena Yeung, CS 231n

Im2Col

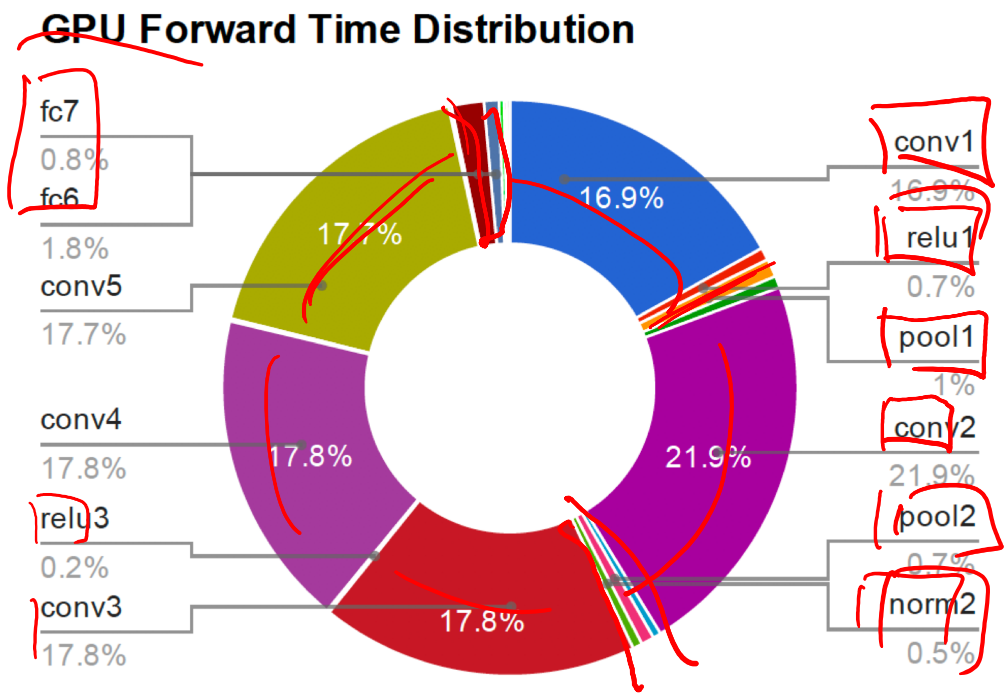


H x W x C

GEMM


Input Matrix
k
Patch 1
Patch 2
...
Number of Patches
x
Kernel Matrix
k
Kernel 1
Kernel 2
...
Number of Kernels
=

Time Distribution of AlexNet



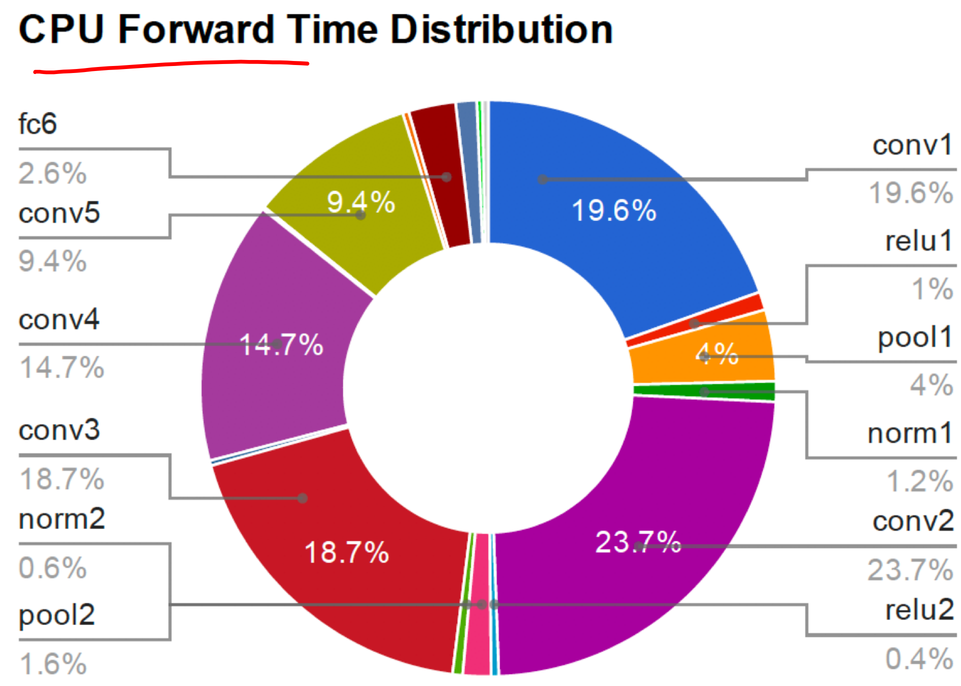
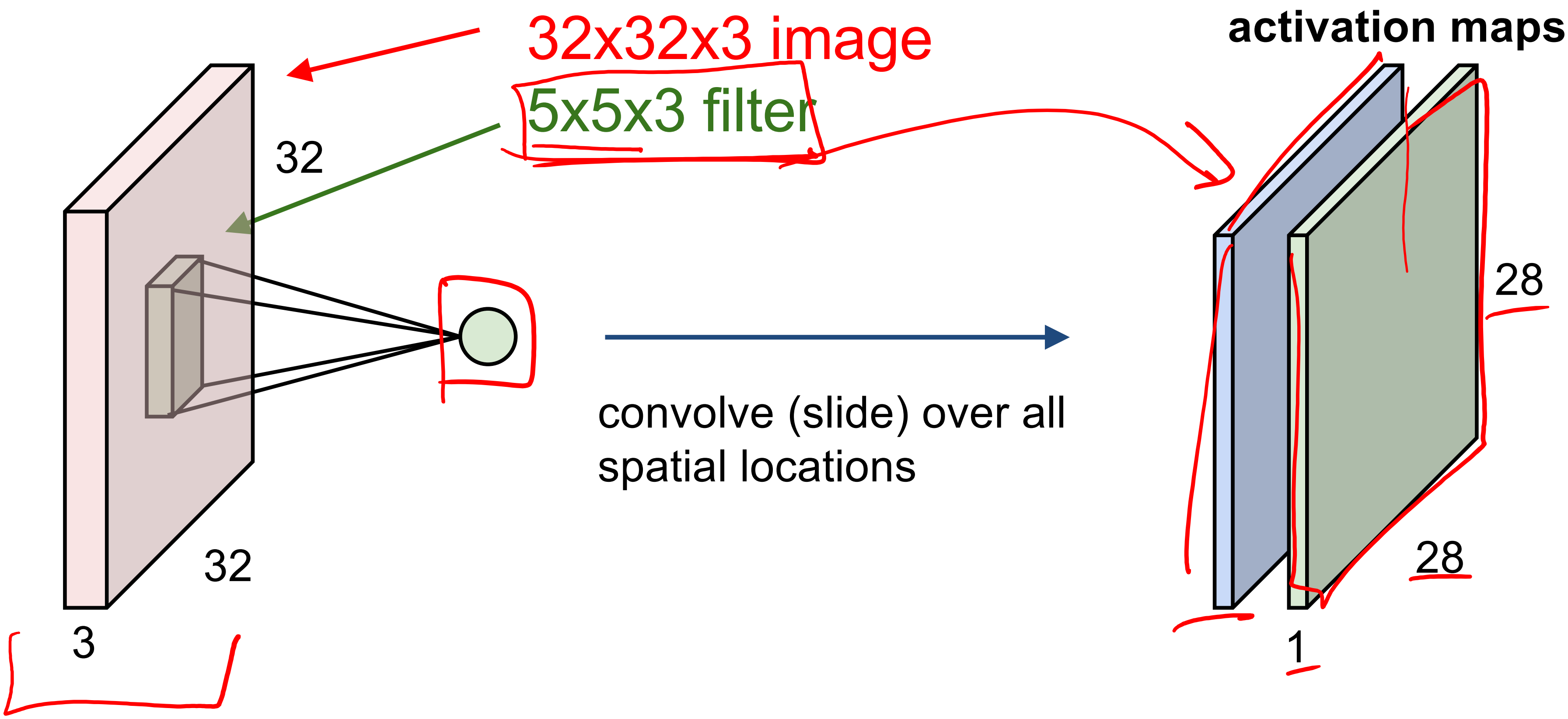


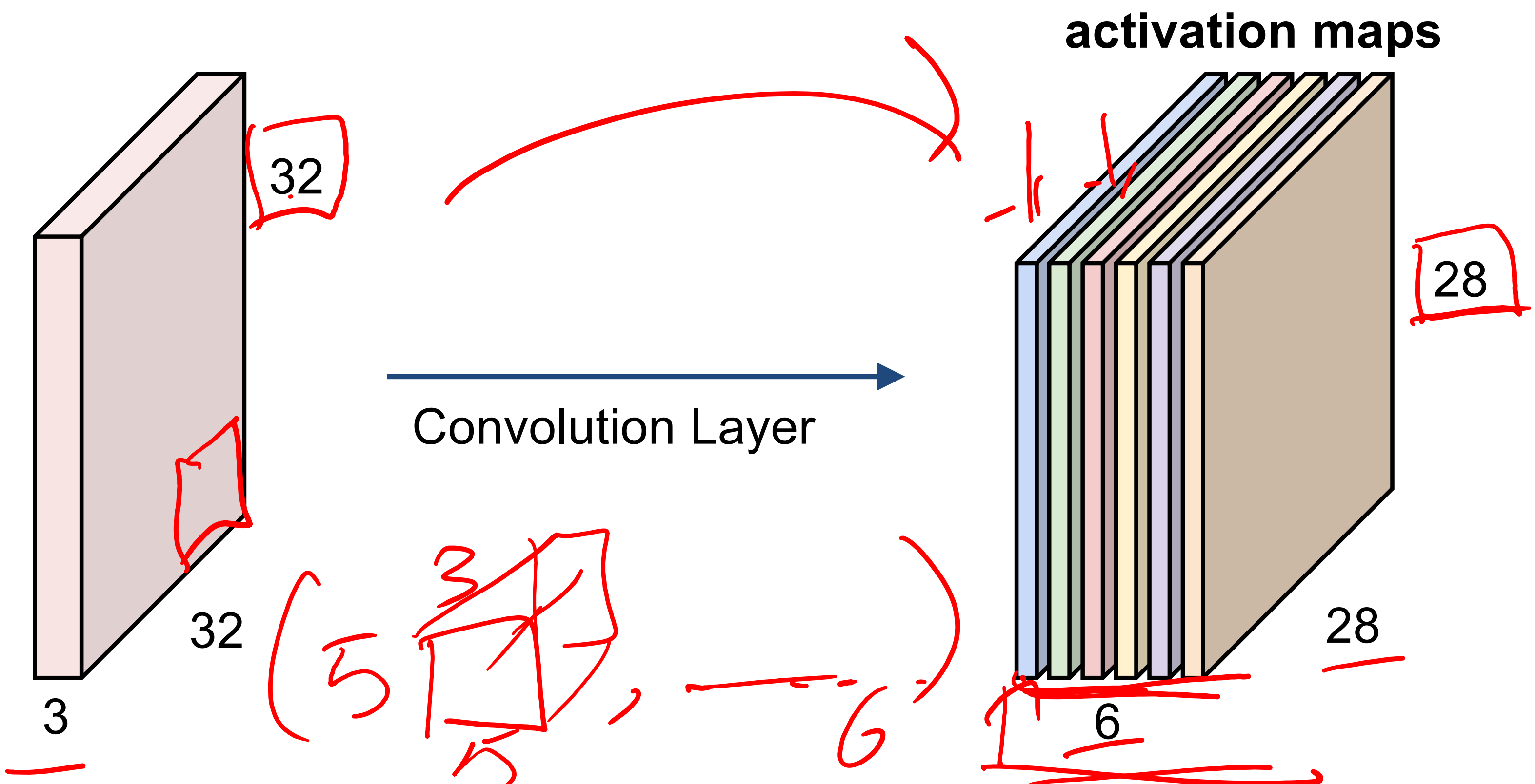
Figure Credit: Yangqing Jia, PhD thesis

Convolution Layer

consider a second, green filter



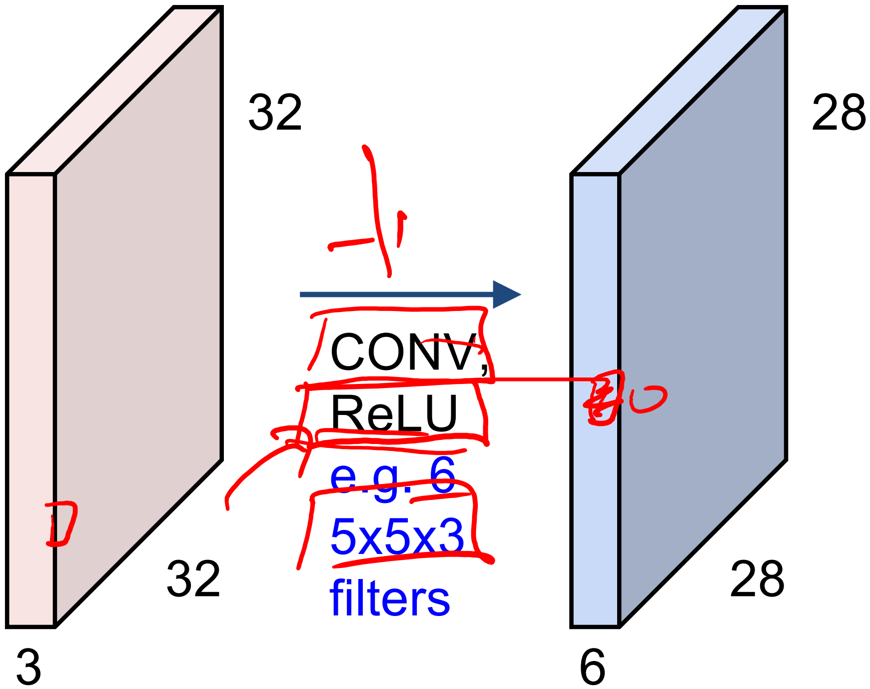
For example, if we had 6 5x5 filters, we'll get 6 separate activation maps:



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

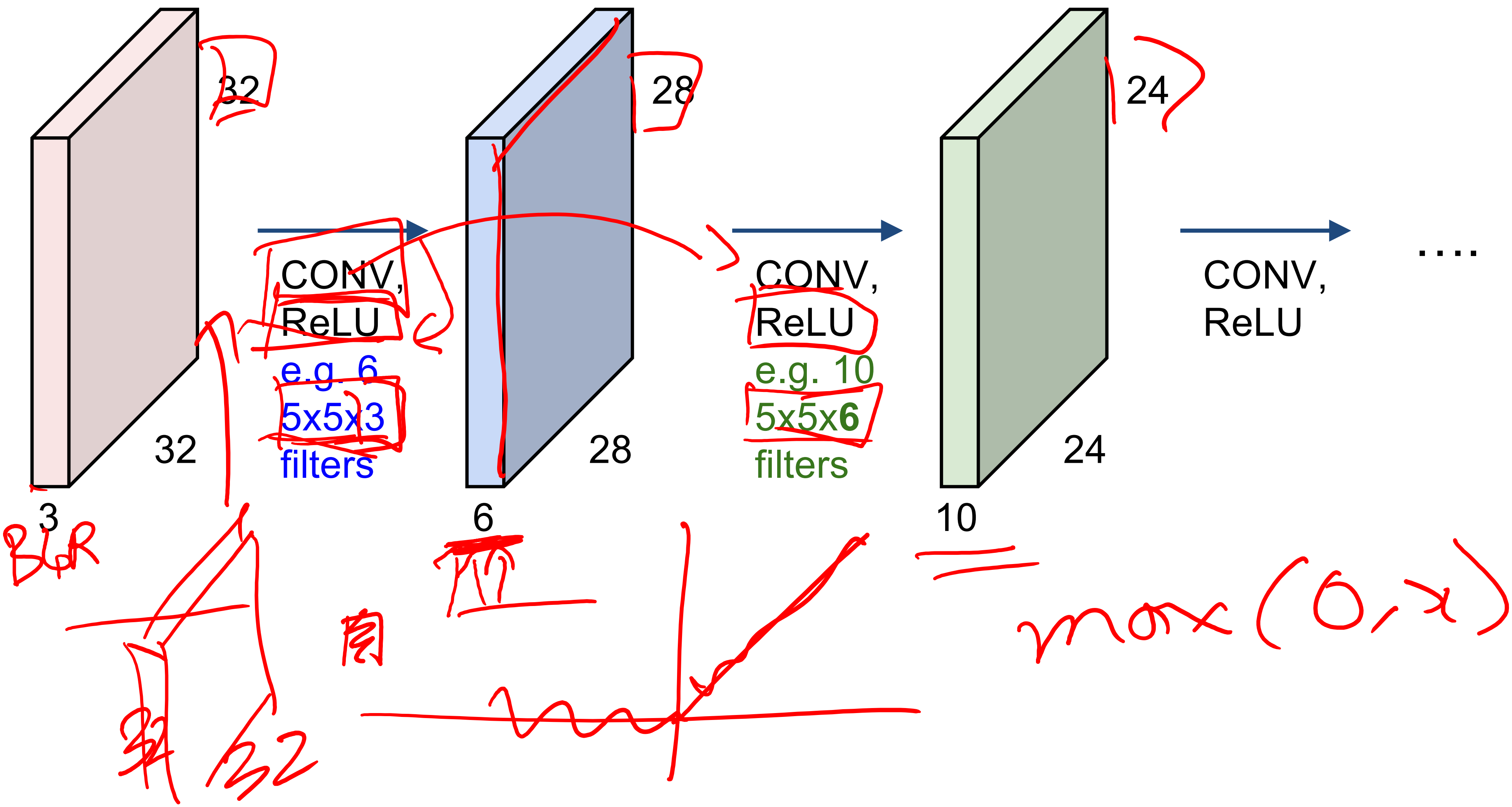
Slide Credit: Fei-Fei Li, Justin Johnson, Serena Yeung, CS 231n

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

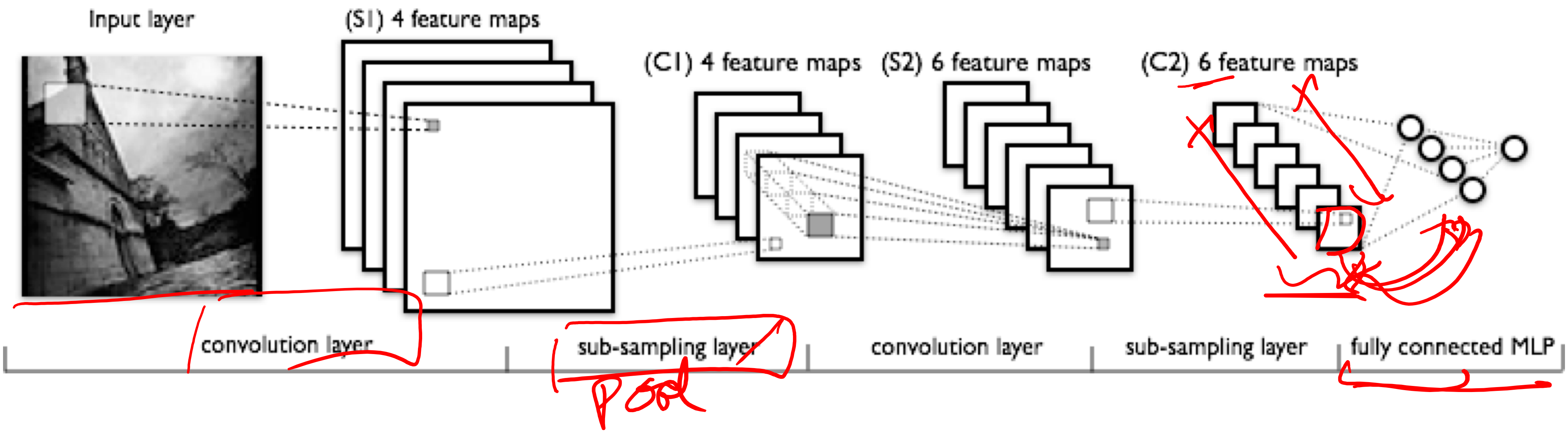


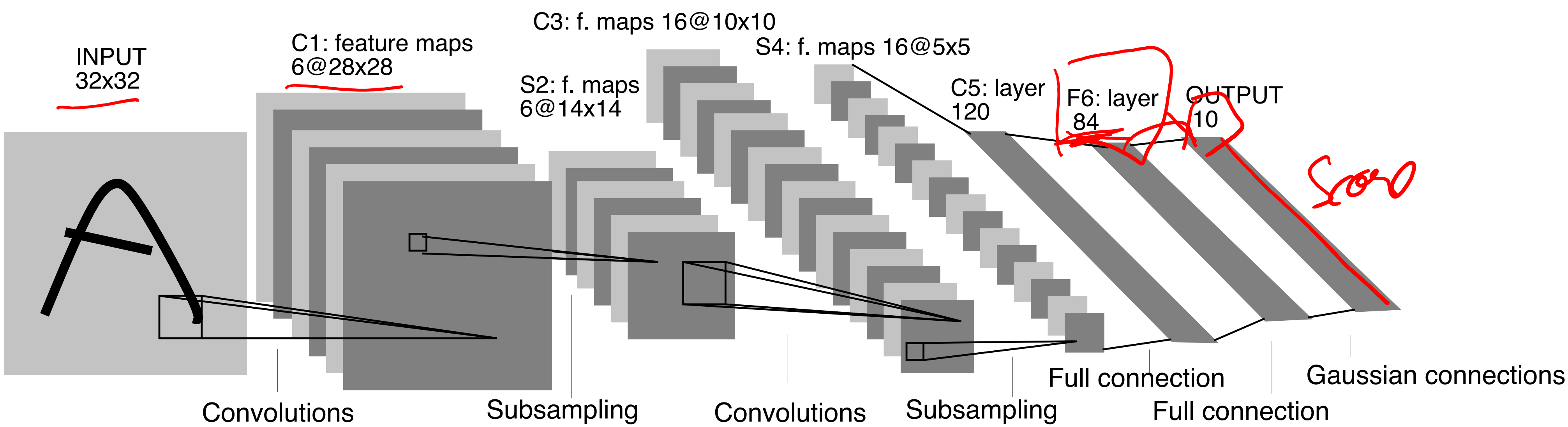
$max(0, x)$

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

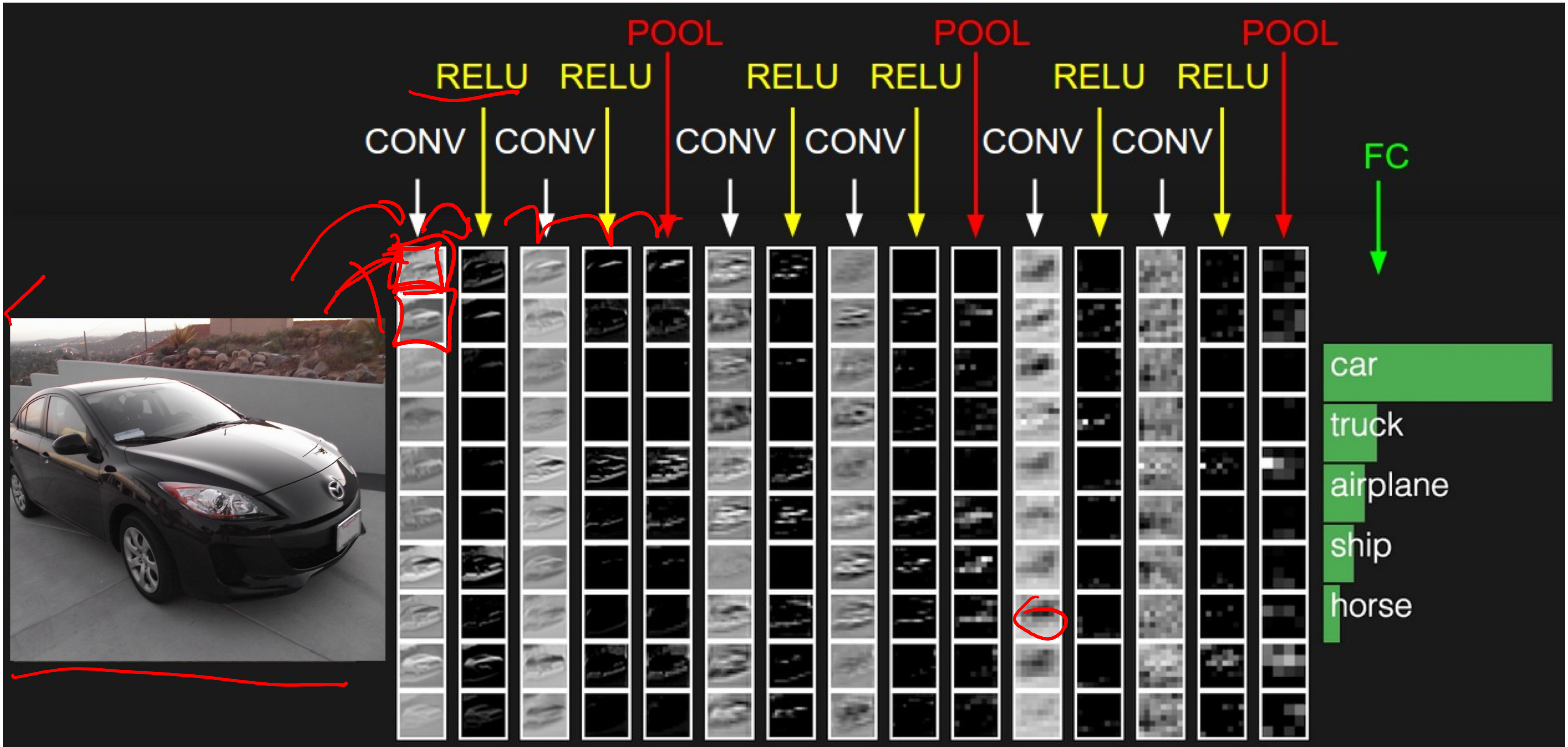


Convolutional Neural Networks





preview:

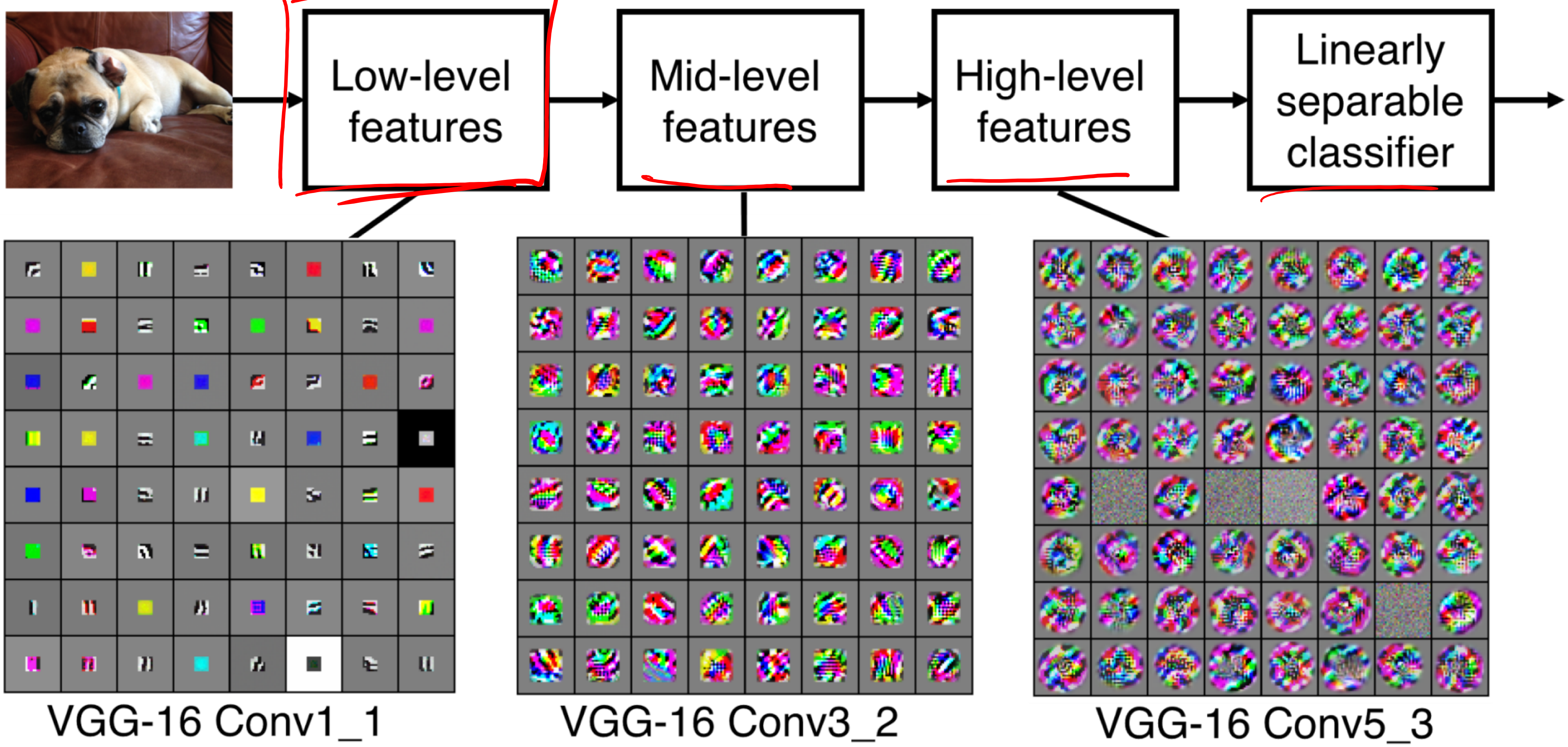


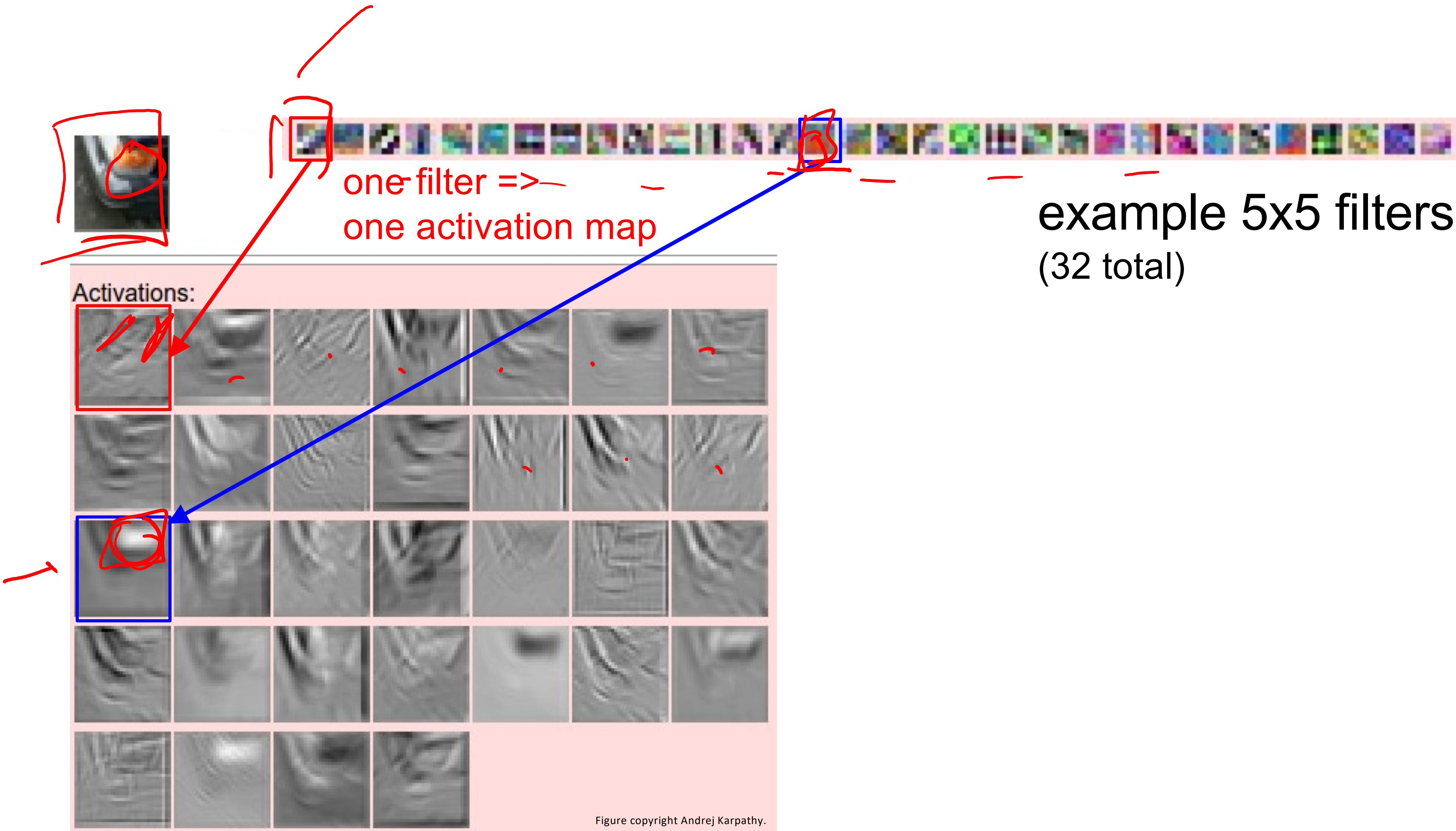
Slide Credit: Fei-Fei Li, Justin Johnson, Serena Yeung, CS 231n

Preview

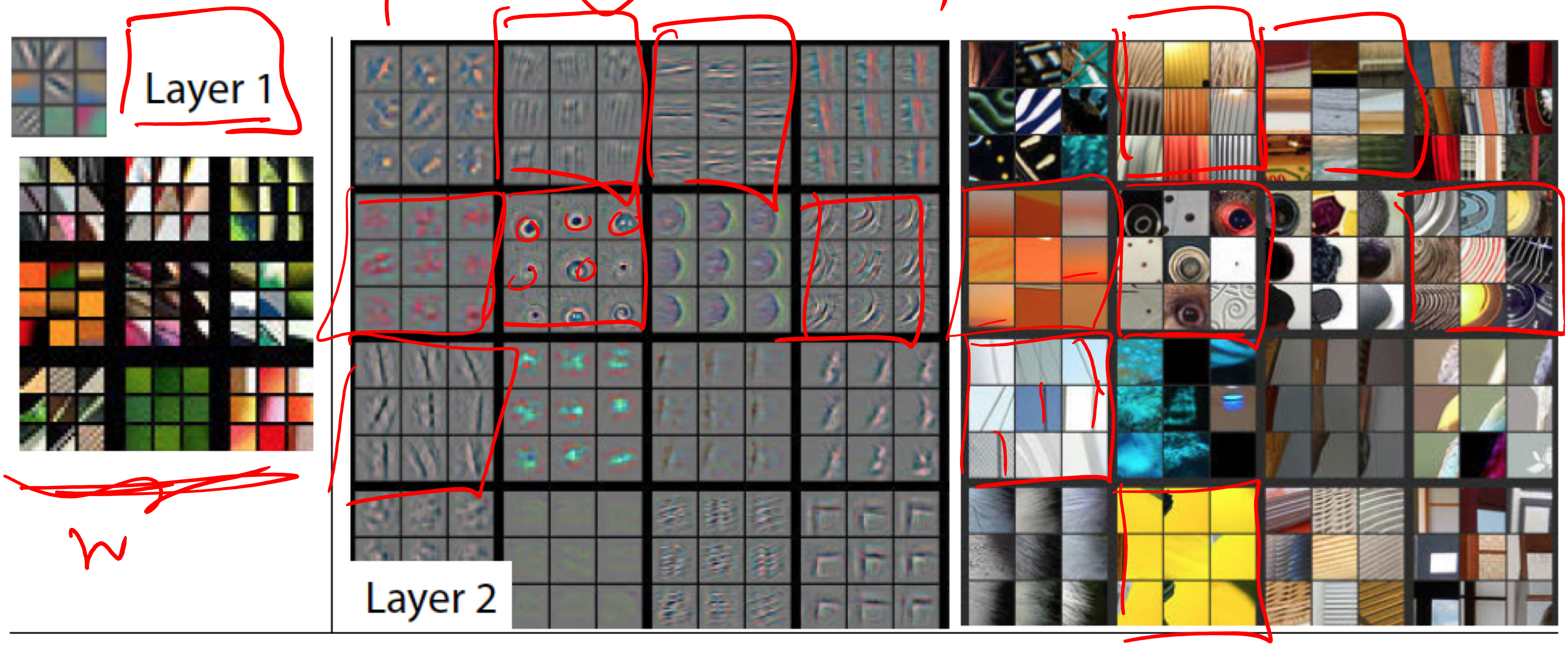
[Zeiler and Fergus 2013]

Visualization of VGG-16 by Lane McIntosh. VGG-16 architecture from [Simonyan and Zisserman 2014].

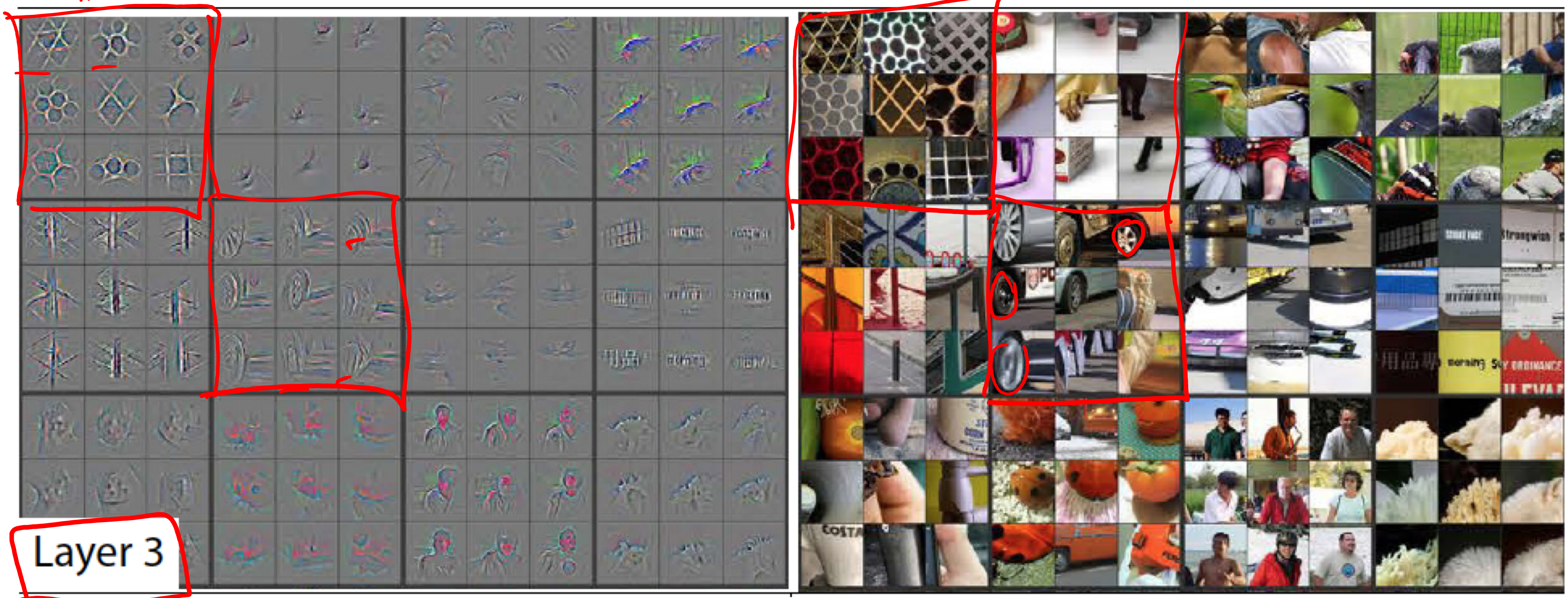



one filter =>
one activation map
example 5x5 filters
(32 total)
Activations:
Figure copyright Andrej Karpathy.

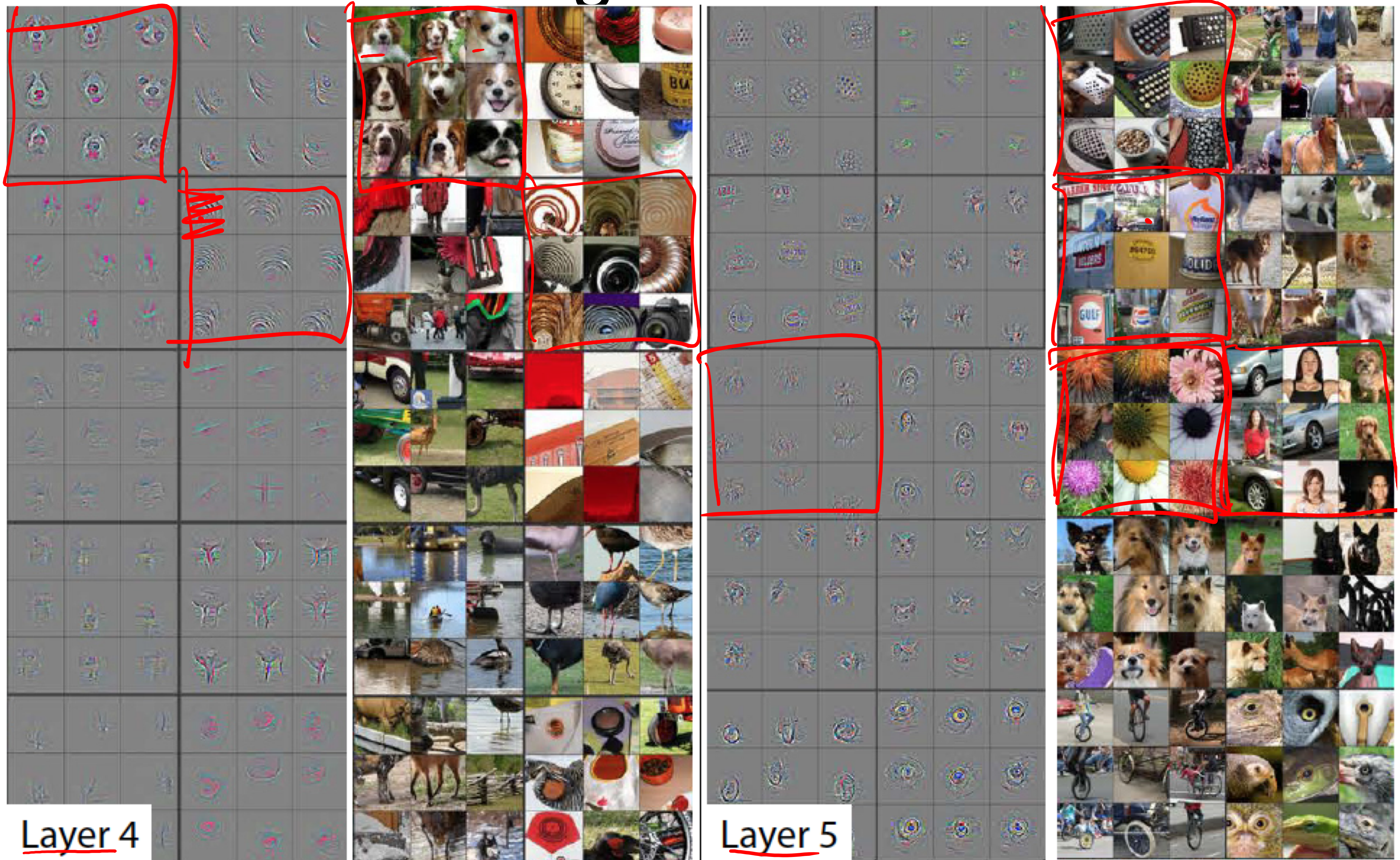
Visualizing Learned Filters



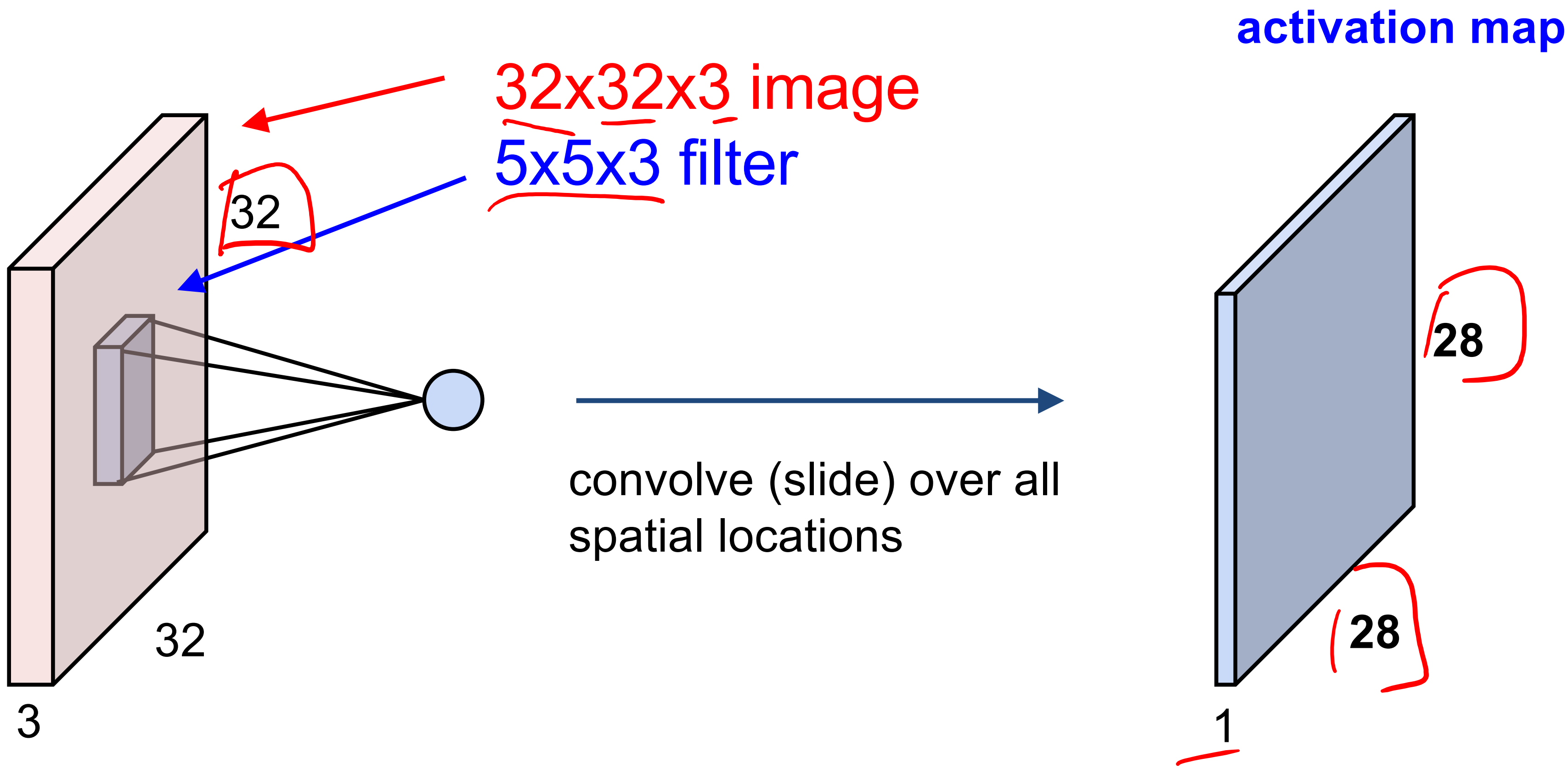
Visualizing Learned Filters



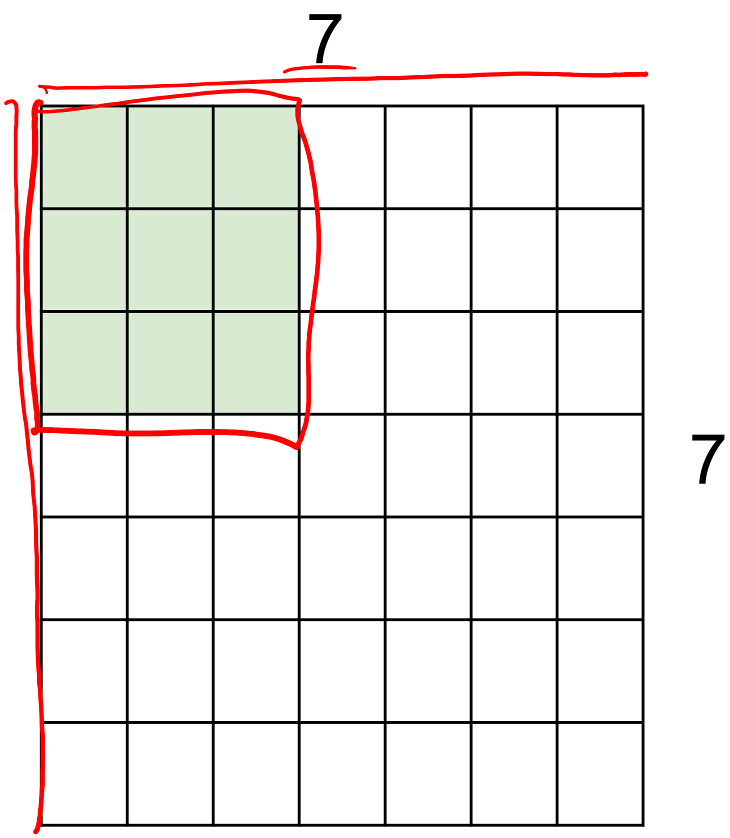
Visualizing Learned Filters $x \to \phi(x)$



A closer look at spatial dimensions:



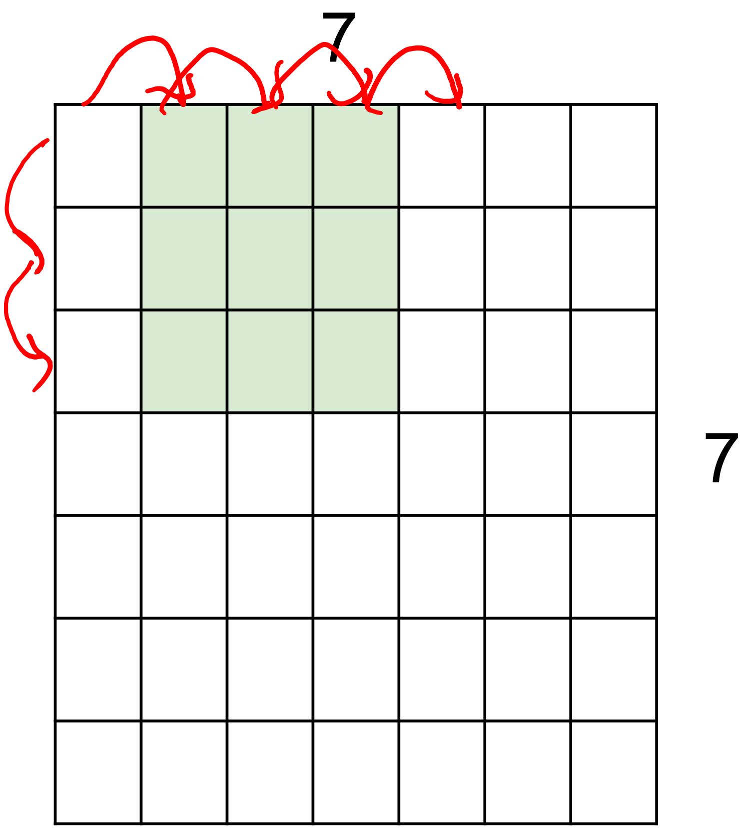
A closer look at spatial dimensions:



7x7 input (spatially)
assume 3x3 filter

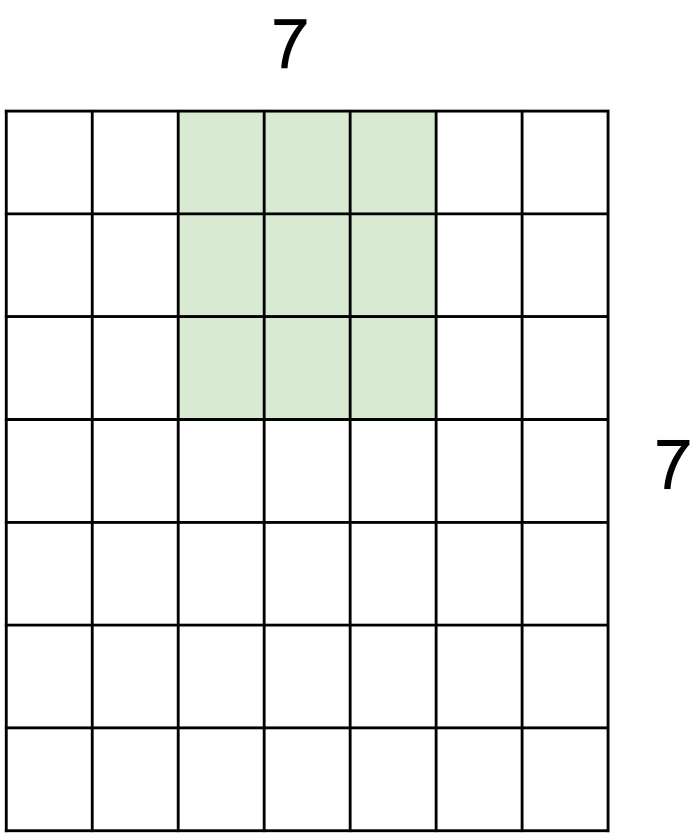
A closer look at spatial dimensions:

stride = 1



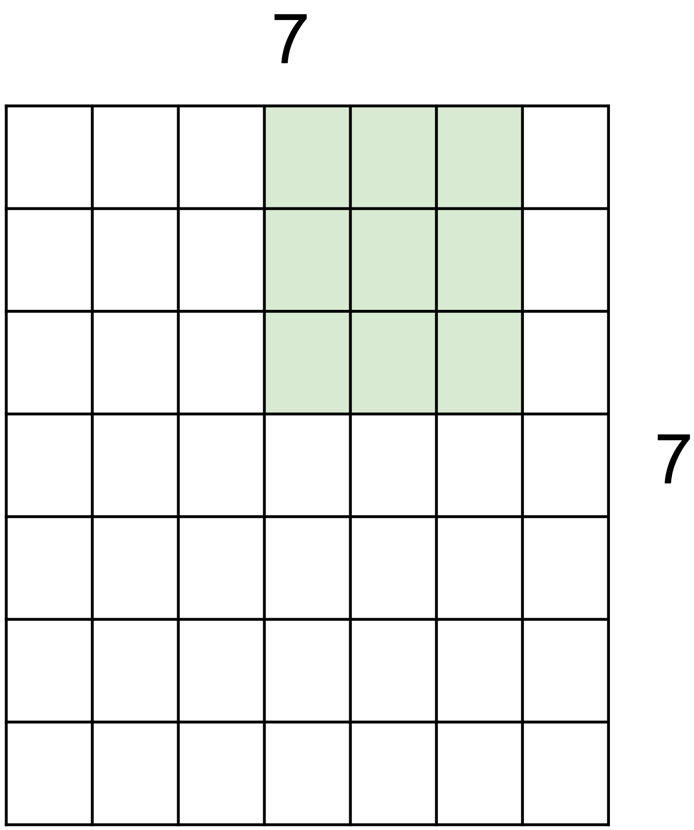
7x7 input (spatially)
assume 3x3 filter

A closer look at spatial dimensions:



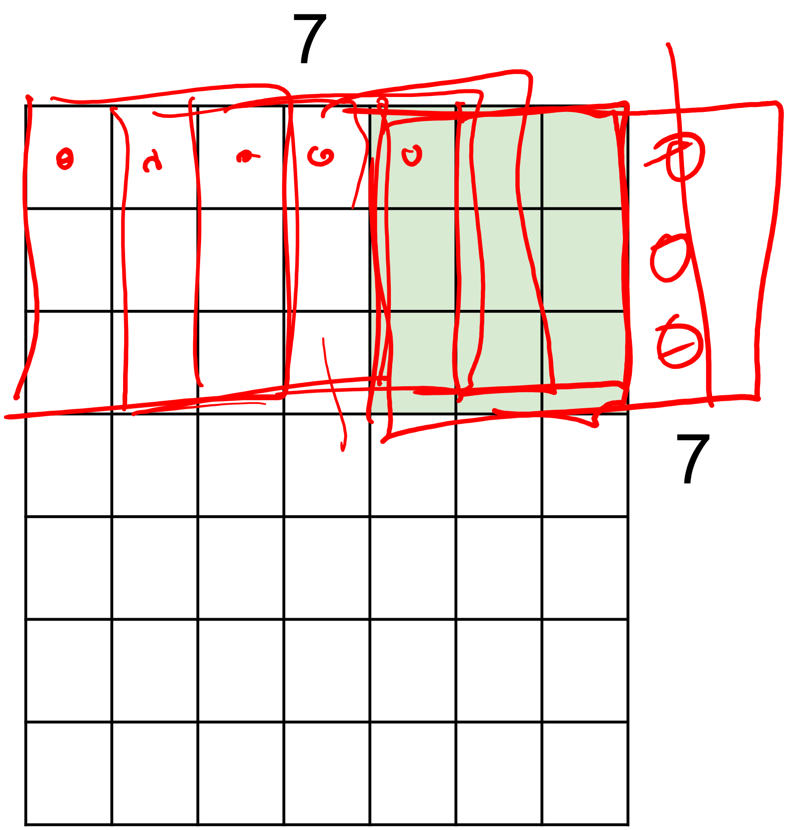
7x7 input (spatially)
assume 3x3 filter

A closer look at spatial dimensions:



7x7 input (spatially)
assume 3x3 filter

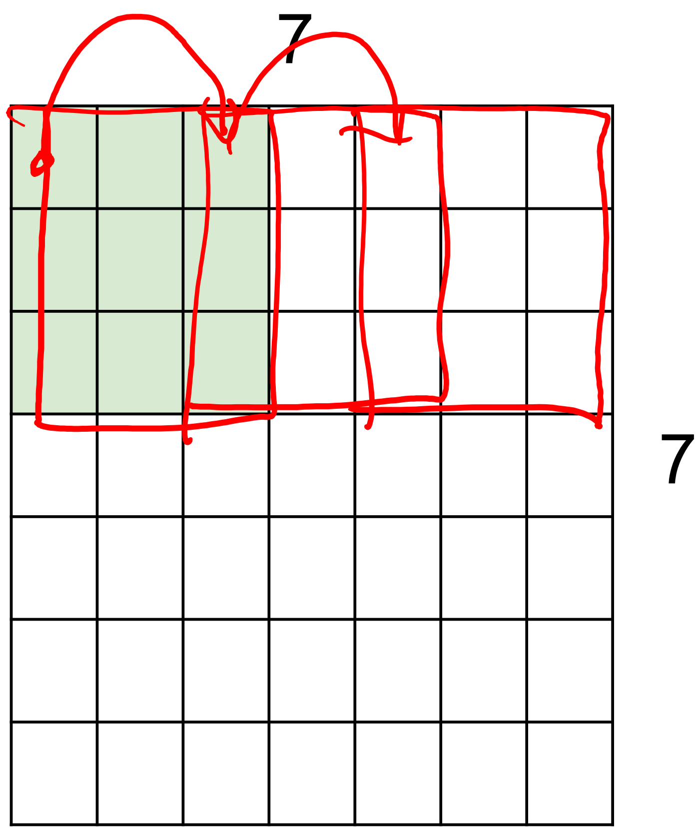
A closer look at spatial dimensions:



7x7 input (spatially)
assume 3x3 filter

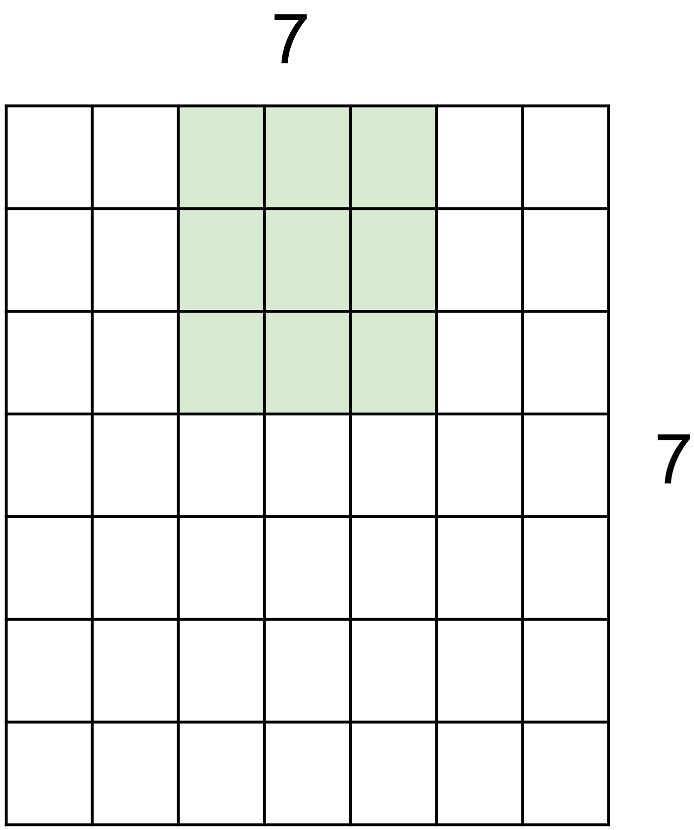
=> 5x5 output

A closer look at spatial dimensions:



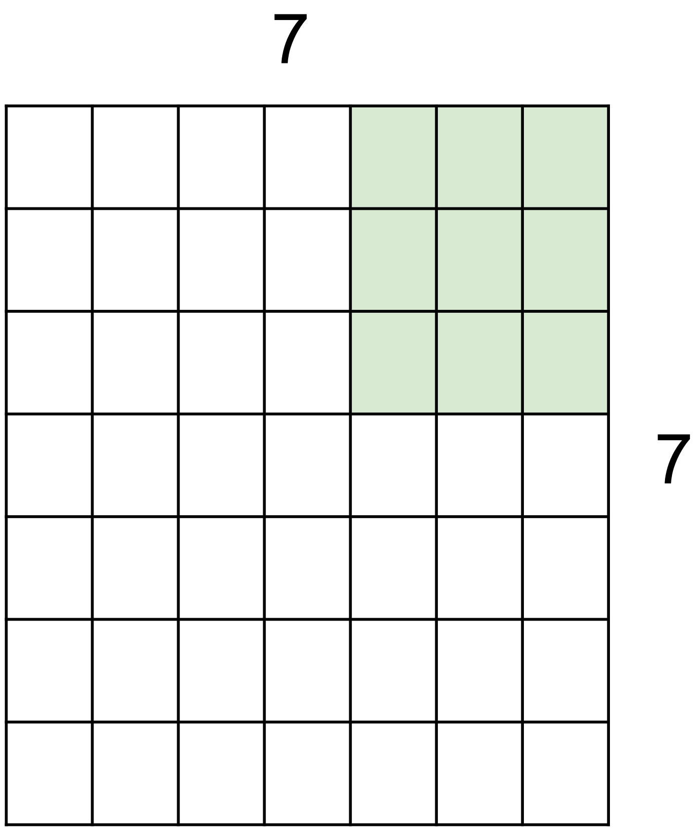
7x7 input (spatially)
assume 3x3 filter
applied **with stride 2**

A closer look at spatial dimensions:



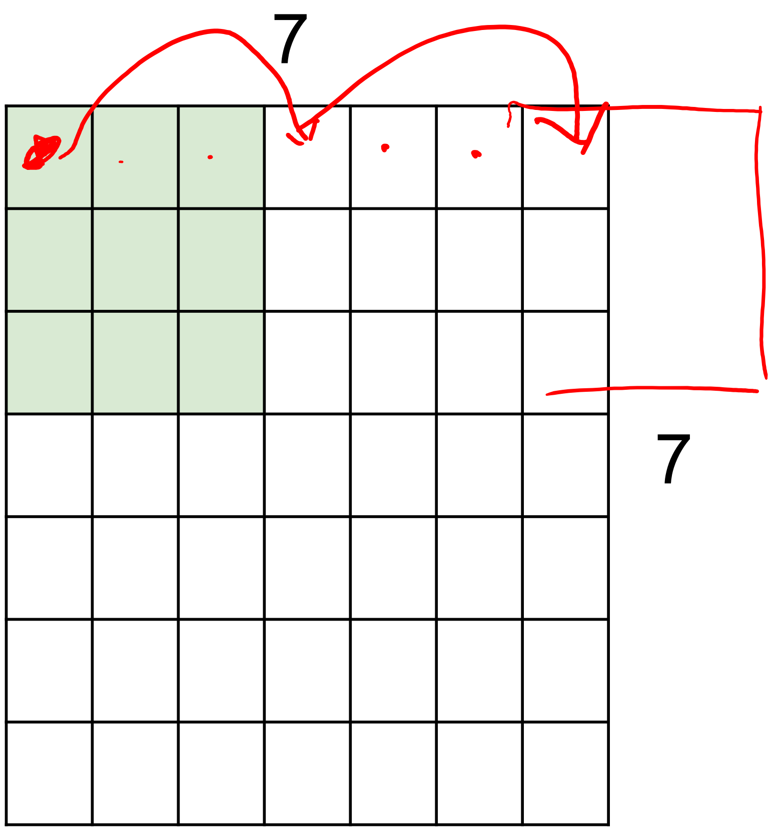
7x7 input (spatially)
assume 3x3 filter
applied **with stride 2**

A closer look at spatial dimensions:



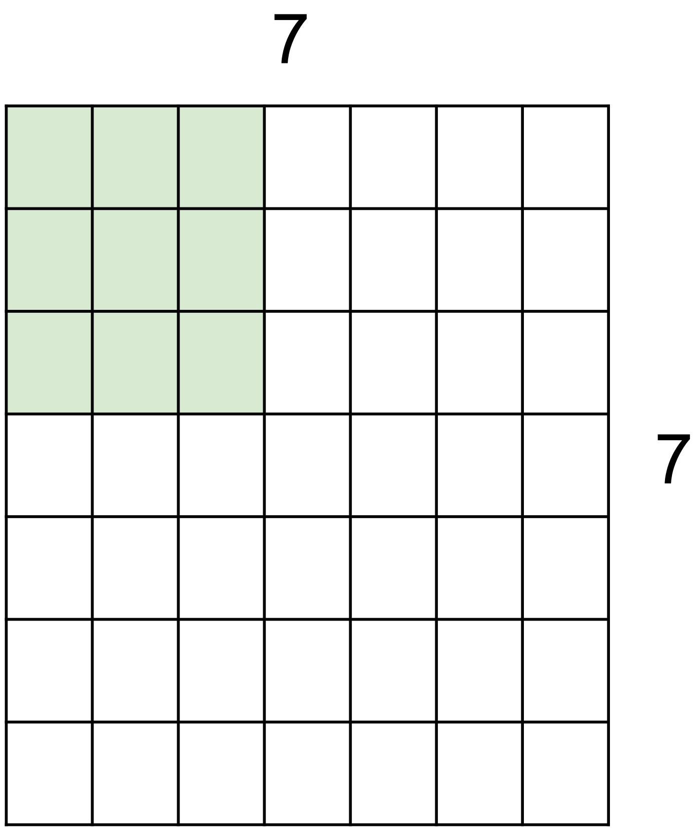
7x7 input (spatially)
assume 3x3 filter
applied **with stride 2**
=> 3x3 output!

A closer look at spatial dimensions:



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

A closer look at spatial dimensions:

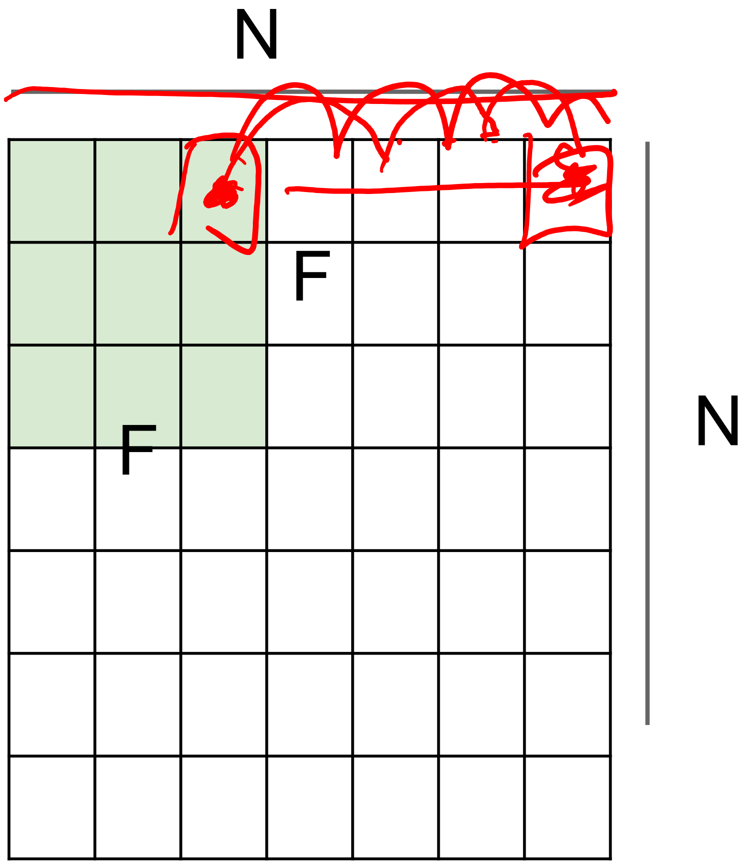


7x7 input (spatially)
assume 3x3 filter
applied **with stride 3?**

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

$(N-F) \propto \text{stride}$

$\frac{N-F}{\text{stride}} + 1$



Output size:

(N - F) / stride + 1

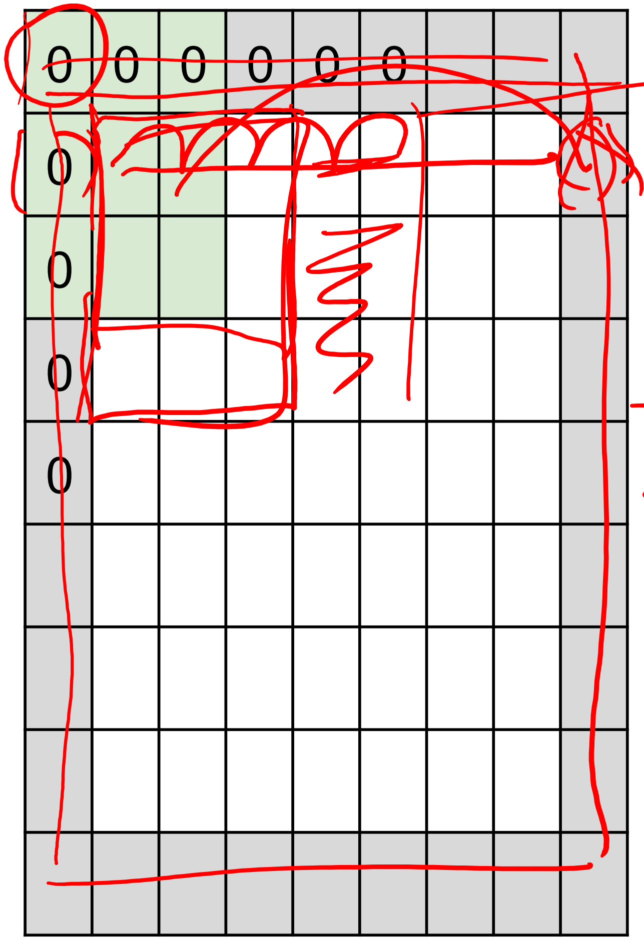
e.g. N = 7, F = 3:

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

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

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

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?

$N \Rightarrow \hat{N}$

7x7 9x9

$$\frac{9-3}{1} + 1$$

$$\boxed{7\times 7}$$

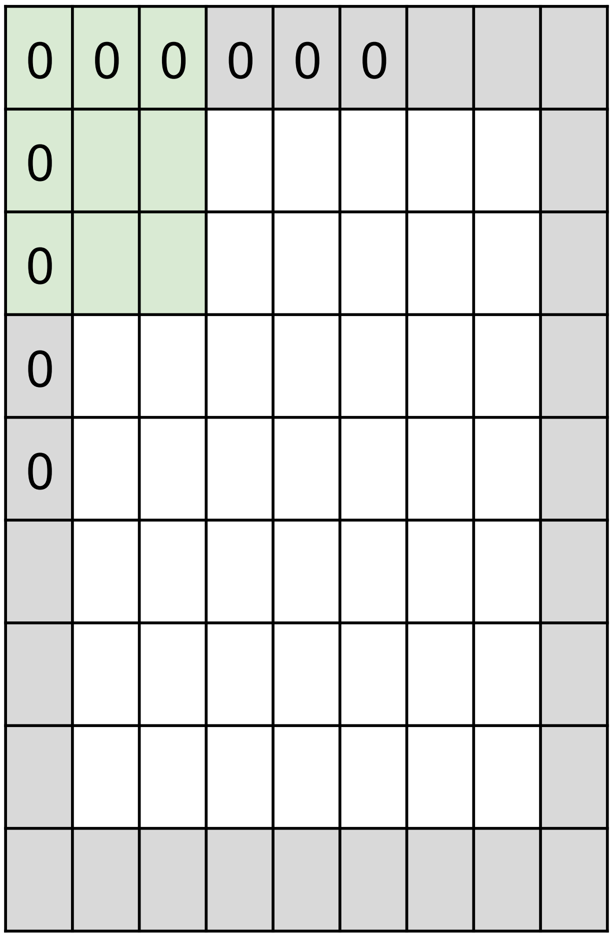
(recall:)
(N - F) / stride + 1

$\hat{N}$

$$\frac{N - F + 2pad}{stride} + 1$$

$$\boxed{\frac{F-1}{2}}$$

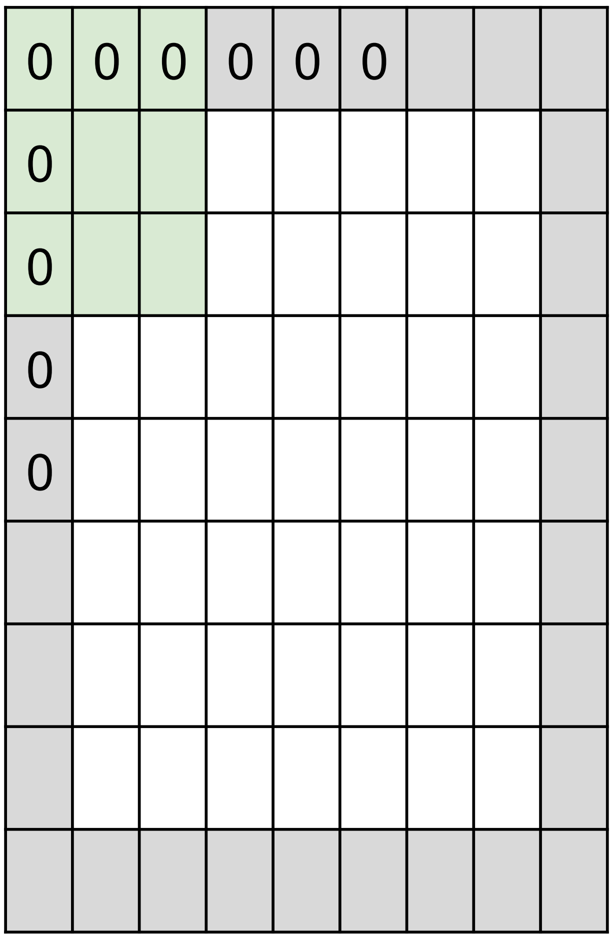
In practice: Common to zero pad the border



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

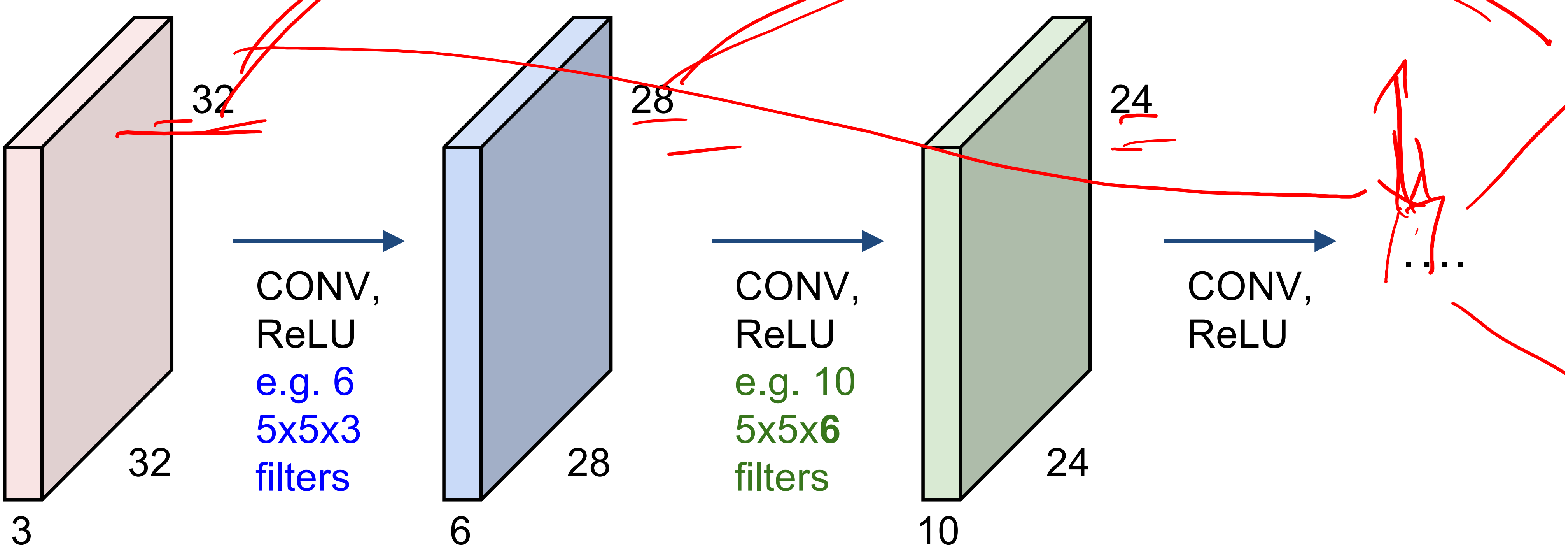


e.g. input 7x7
3x3 filter, applied with **stride 1**
pad with 1 pixel border => what is the output?

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

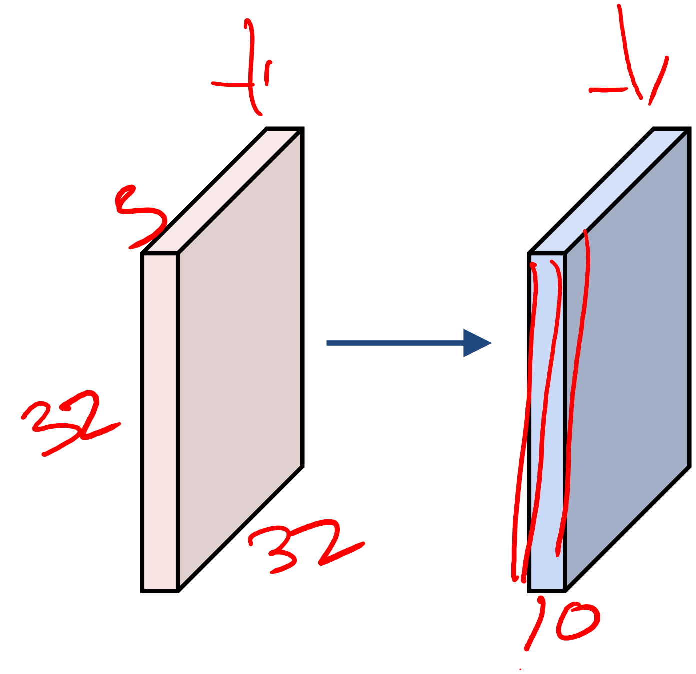
Remember back to…

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.



Examples time:

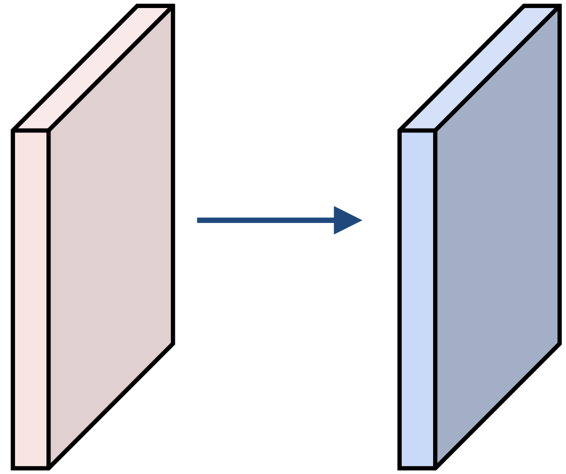
Input volume: **32x32x3**
10 5x5 filters with stride 1, pad 2



Output volume size: ?

$$\underbrace{(32)}_{N_1} \times \underbrace{(32)}_{N_2} \times \boxed{\underbrace{(10)}_{C_2}}$$

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

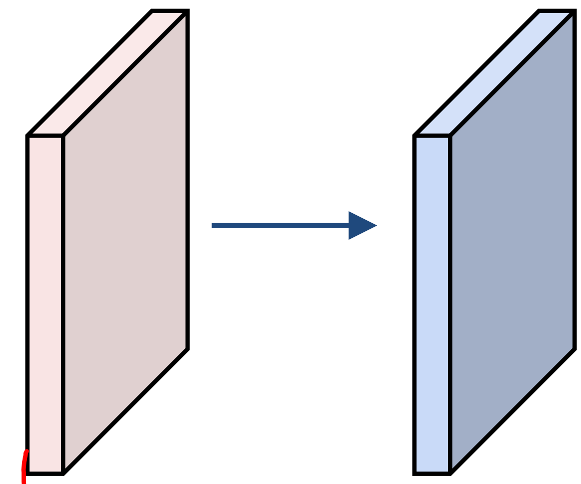
32x32x10

Examples time:

Input volume: **32x32x3**

10 5x5 filters with stride 1, pad 2

Number of parameters in this layer?



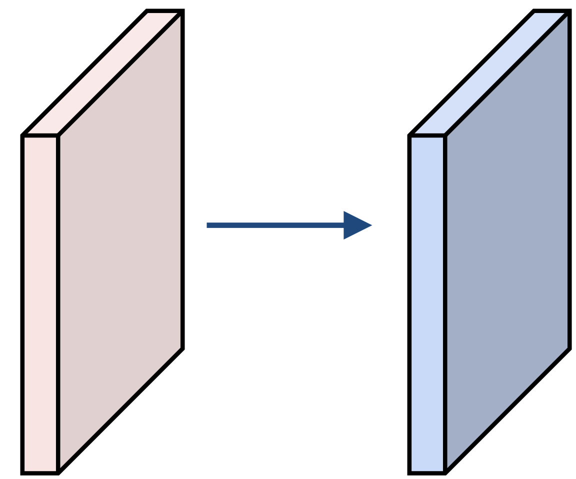
3

$(5\times5\times3 + 1)\times 10 = 760$

$\cancel{(\ )\times(\ )\times(\ )\times(\ )}$

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

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.