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: 09/26, 11:55pm
  - https://evalai.cloudcv.org/web/challenges/challenge-page/431/leaderboard/1200

- Project Teams Google Doc
  - https://docs.google.com/spreadsheets/d/1ouD6ctaemV_3nb2MQHs7rUOAaW9DFLu8I5Zd3yOFs7E/edit?usp=sharing
  - Project Title
  - 1-3 sentence project summary TL;DR
  - Team member names

Recap from last time

Convolutional Neural Networks

(without the brain stuff)

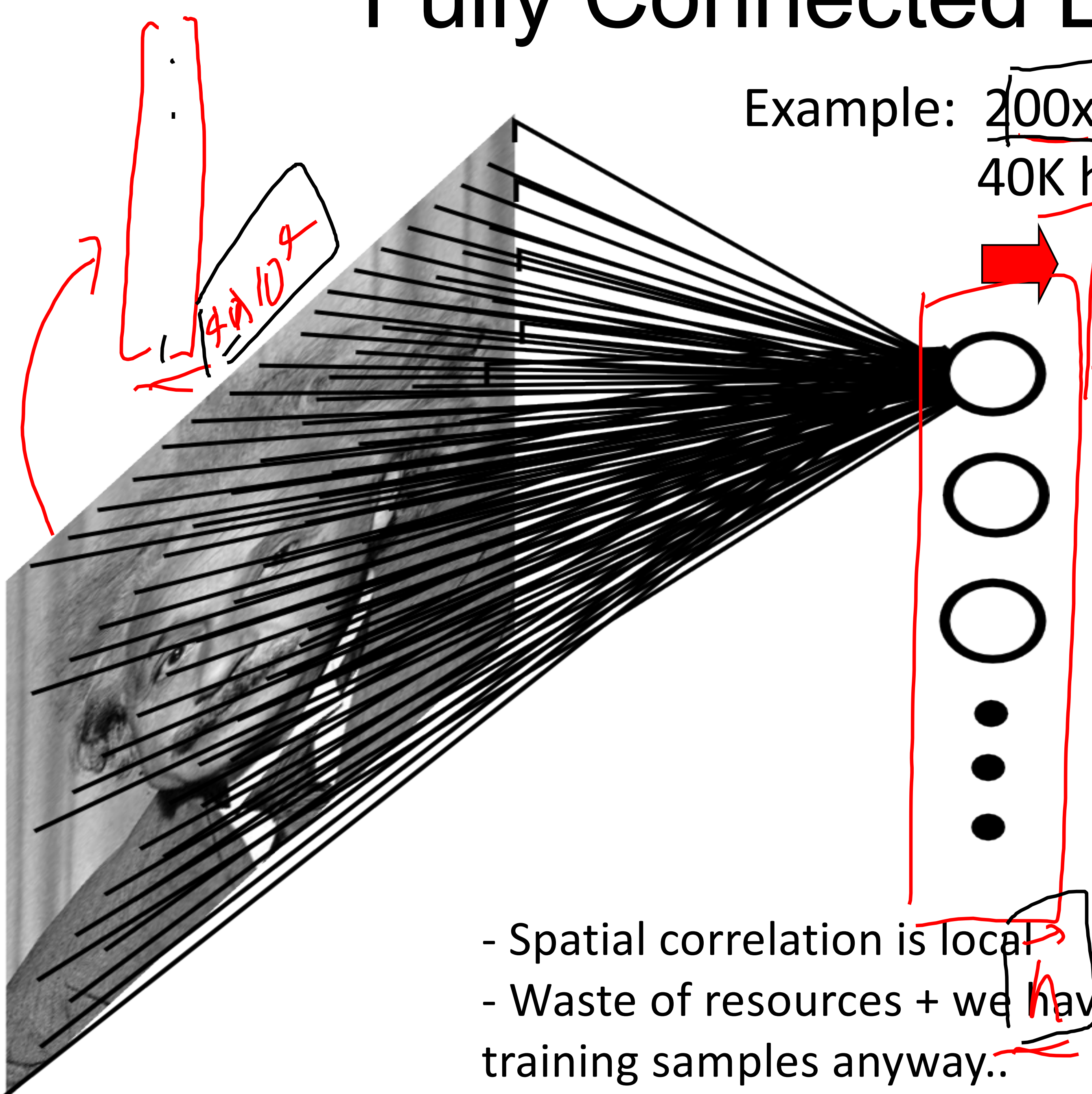
Slide Credit: Fei-Fei Li, Justin Johnson, Serena Yeung, CS 231n

Fully Connected Layer

4×10^4

Example: 200x200 image
40K hidden units

~2B parameters!!! 1.6×10^9



W_1

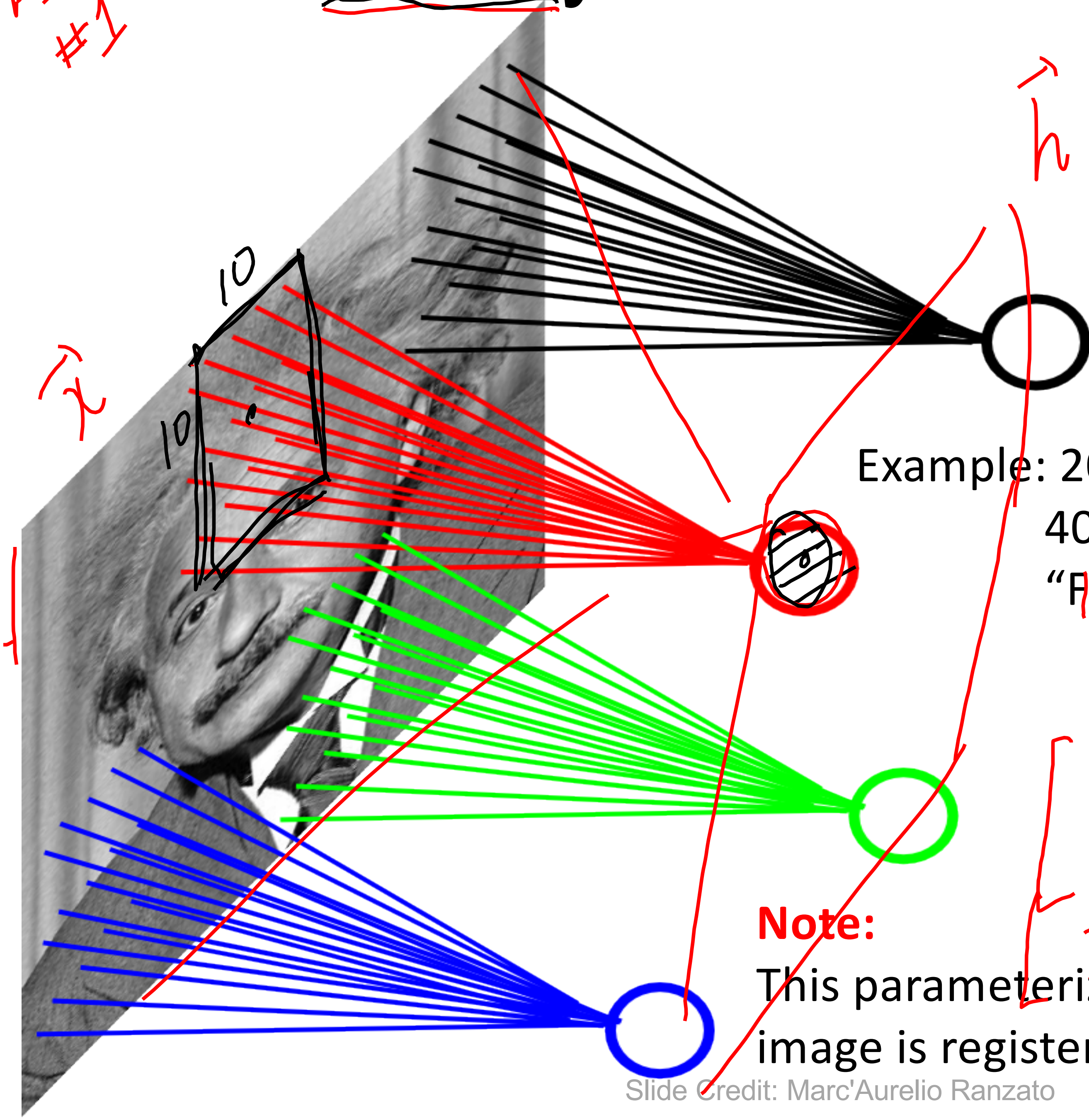
$40k \times 40k$

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

Slide Credit: Marc'Aurelio Ranzato

Assumption #1

Locally Connected Layer



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

4M

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

@ 40k x 100

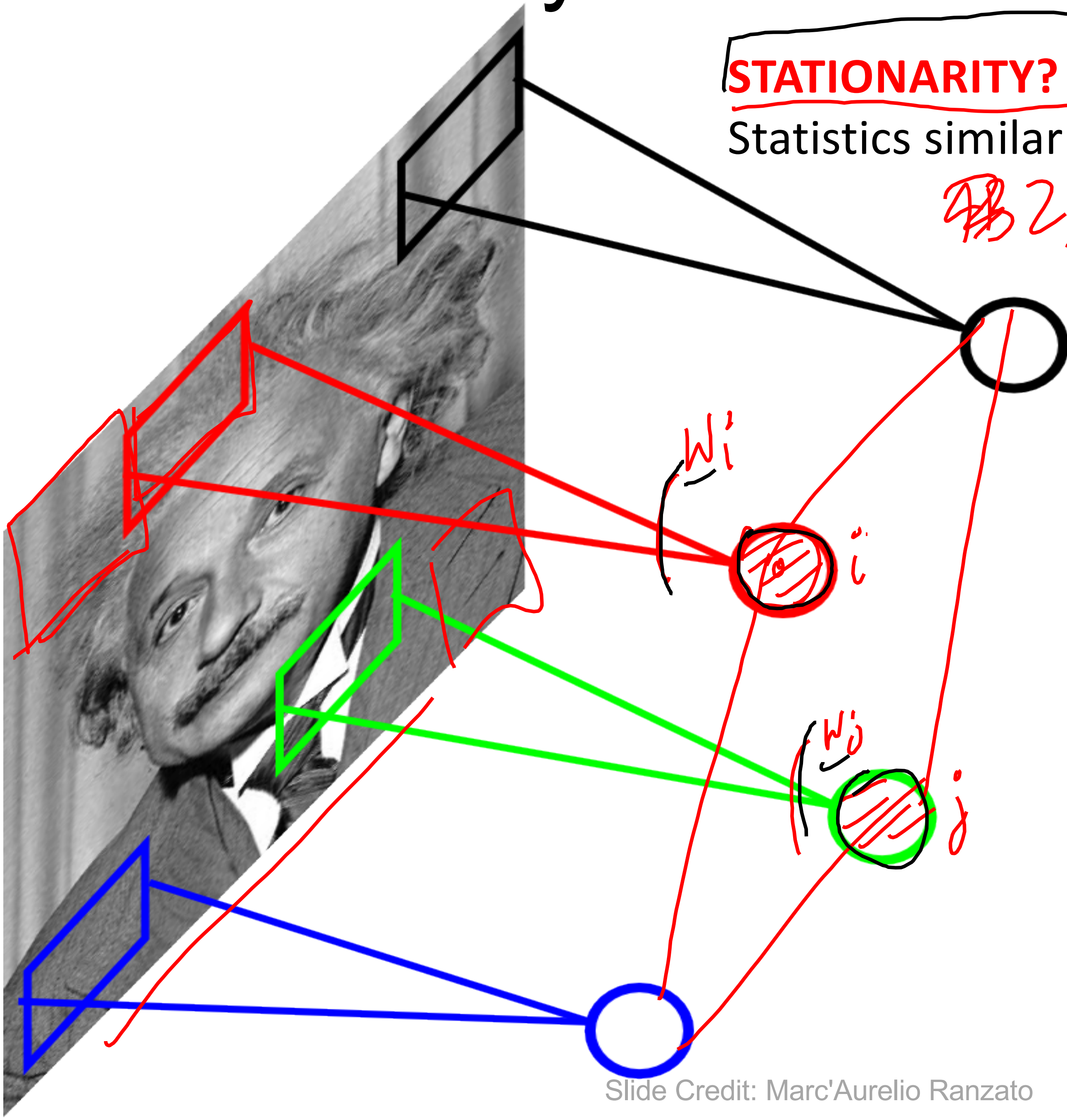
Slide Credit: Marc'Aurelio Ranzato

Locally Connected Layer

STATIONARITY?

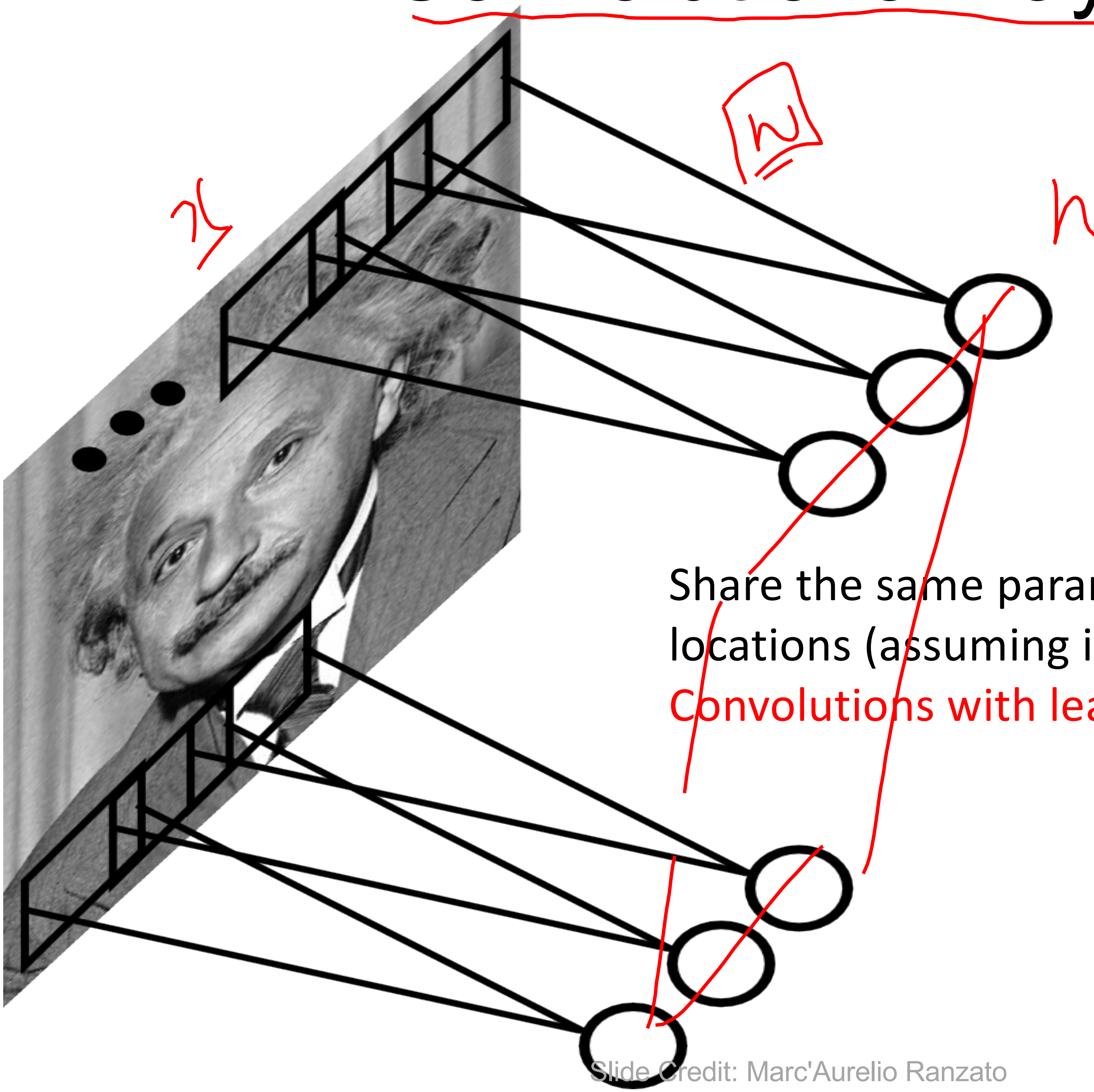
Statistics similar at all locations

$2B \rightarrow 4M \rightarrow \boxed{100}$



$\boxed{W_i = W_j}$

Convolutional Layer



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

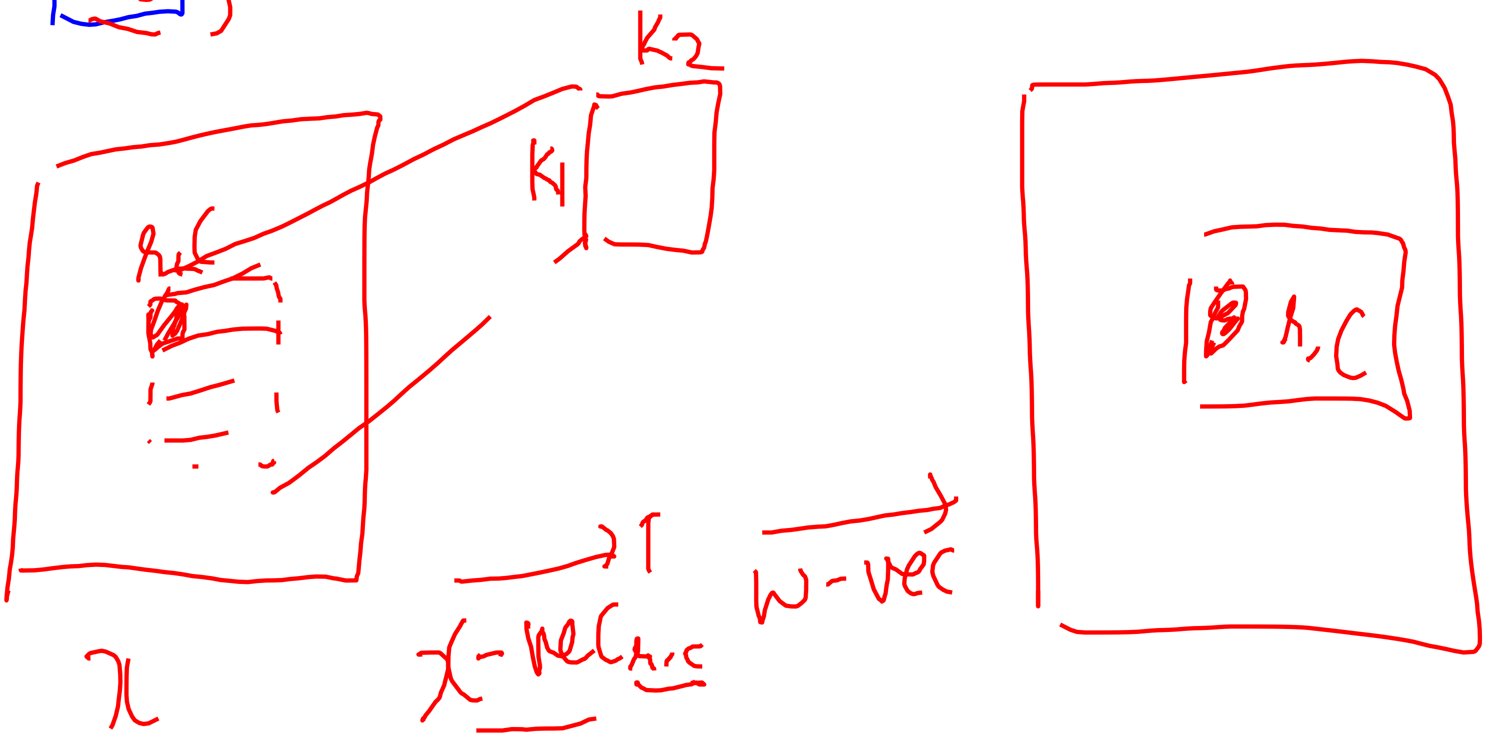
Convolutions!

math → CS → programming

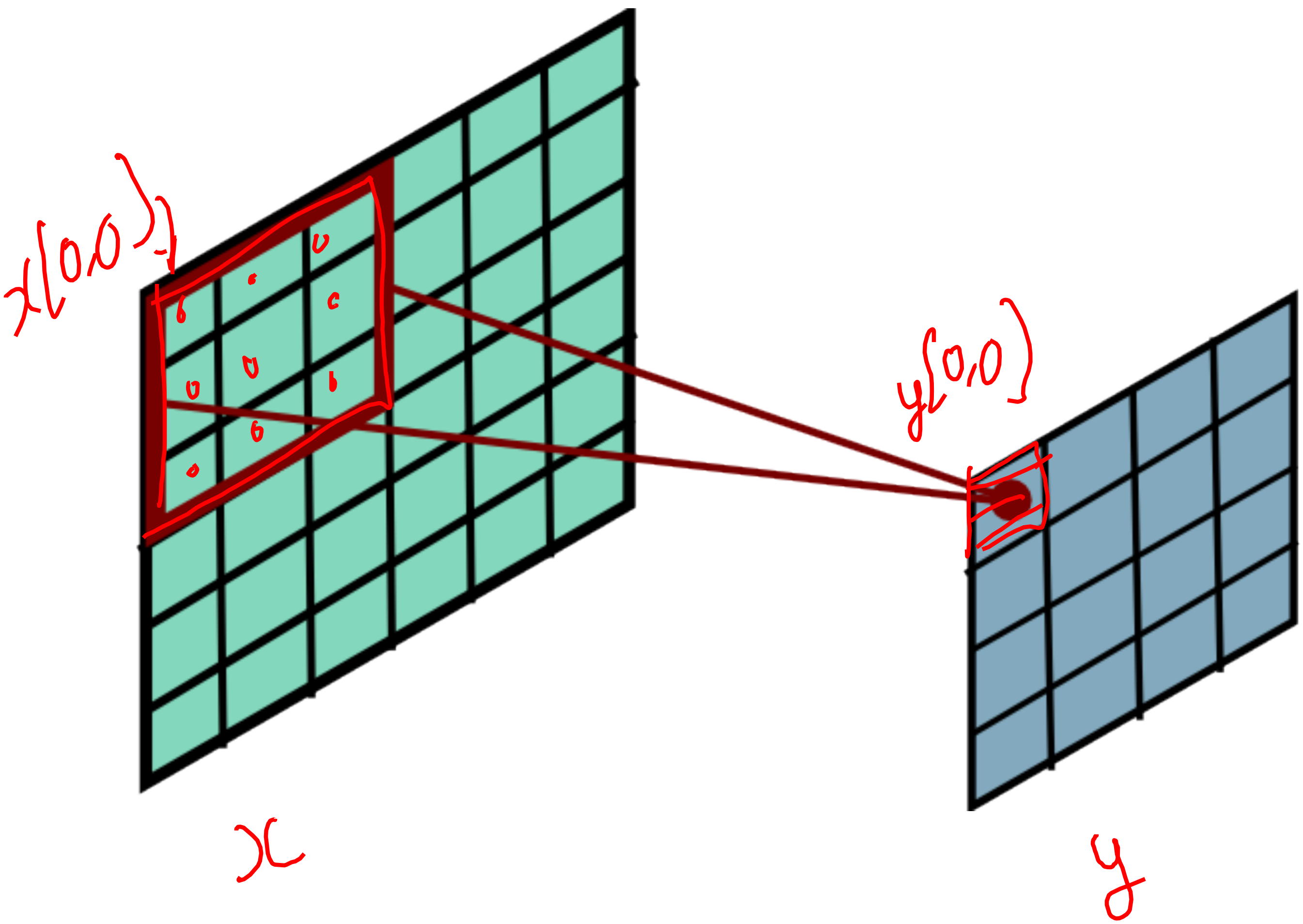
Convolutions for programmers

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

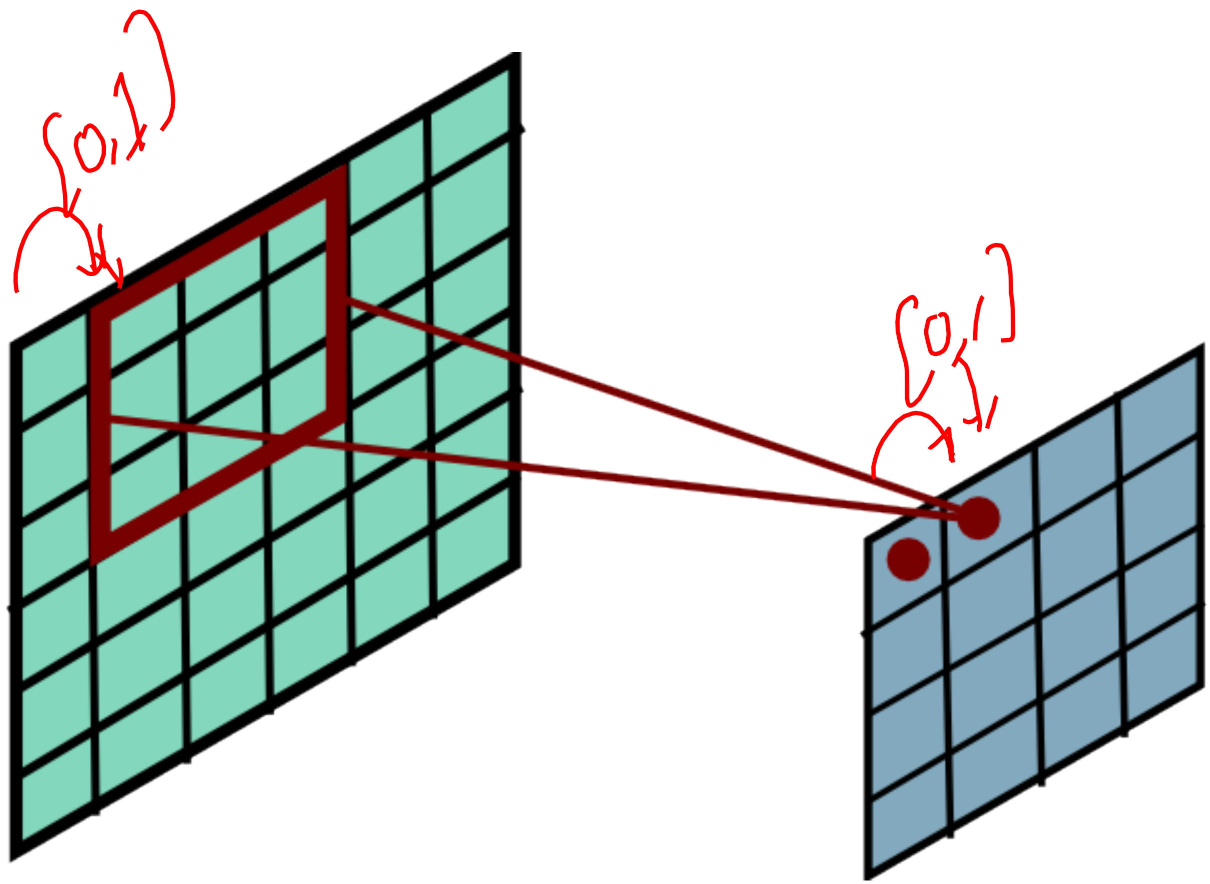
rows cols



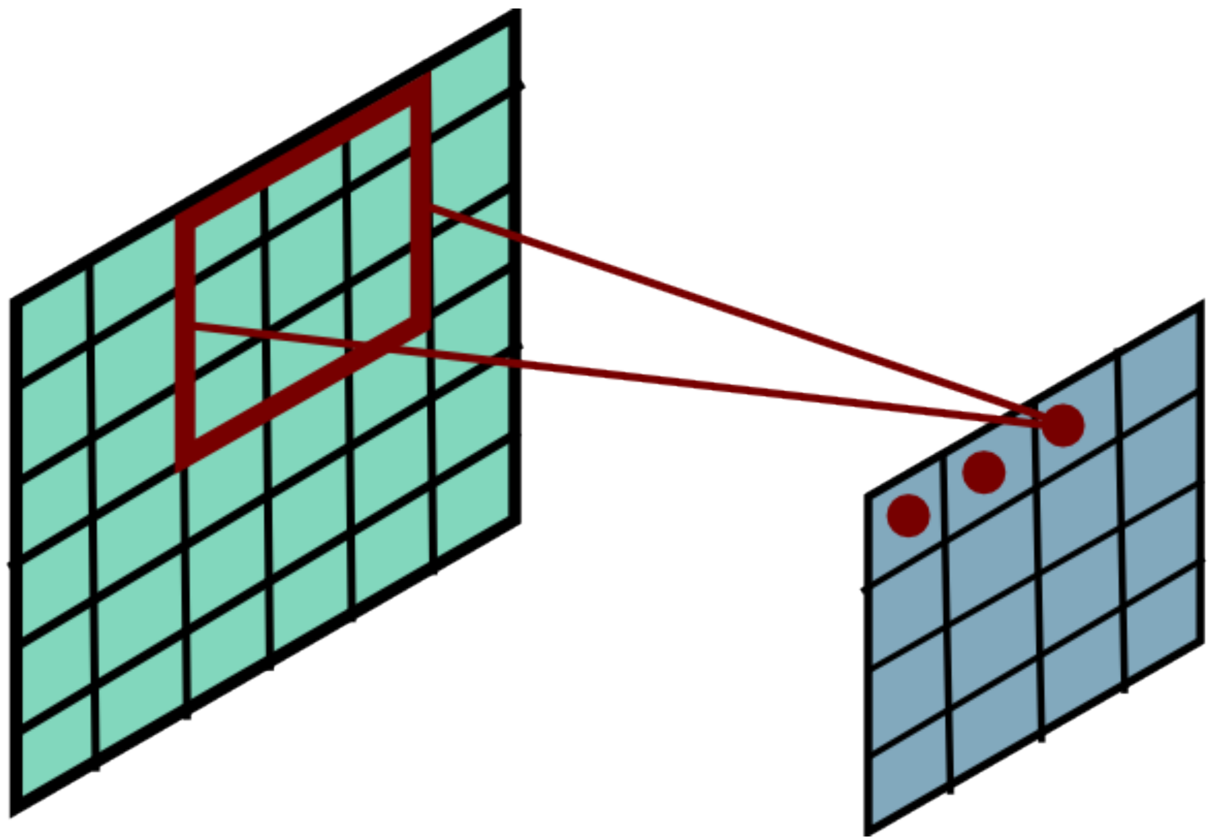
Convolution



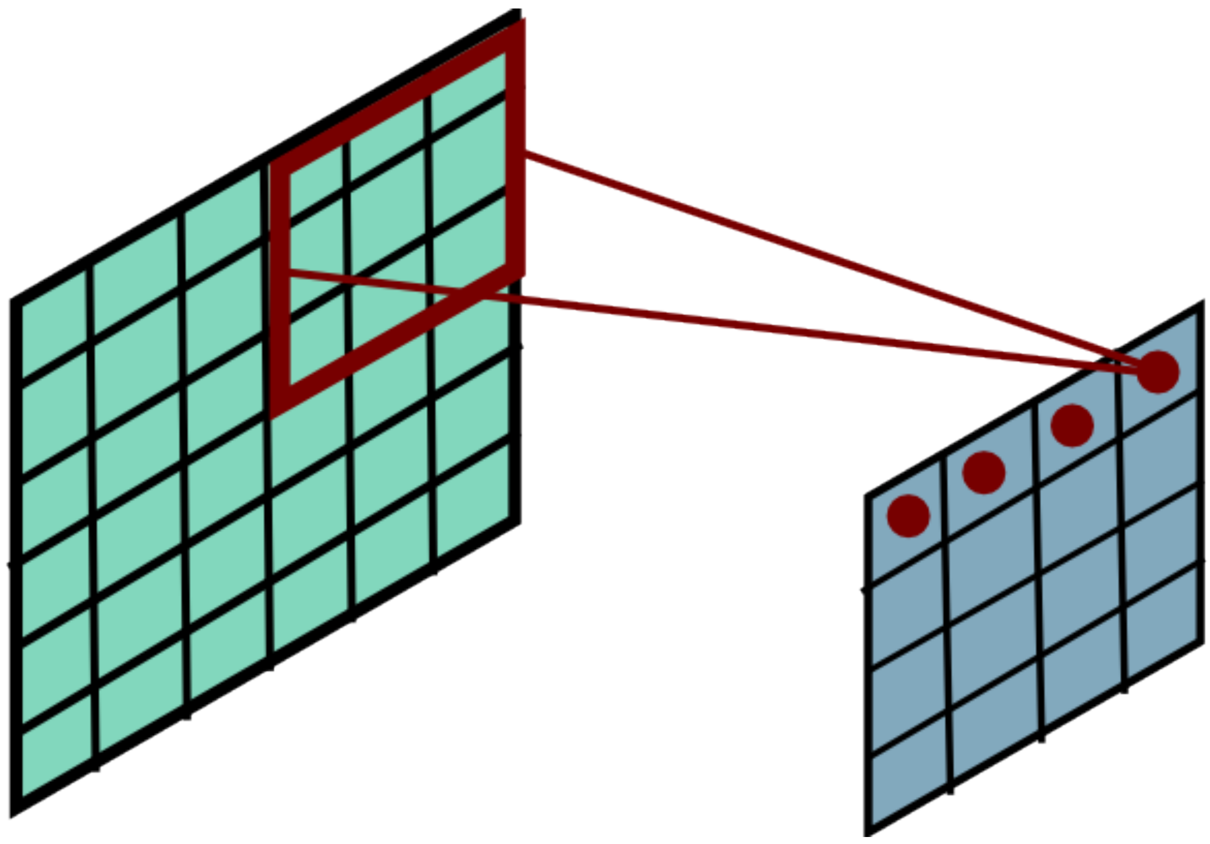
Convolution



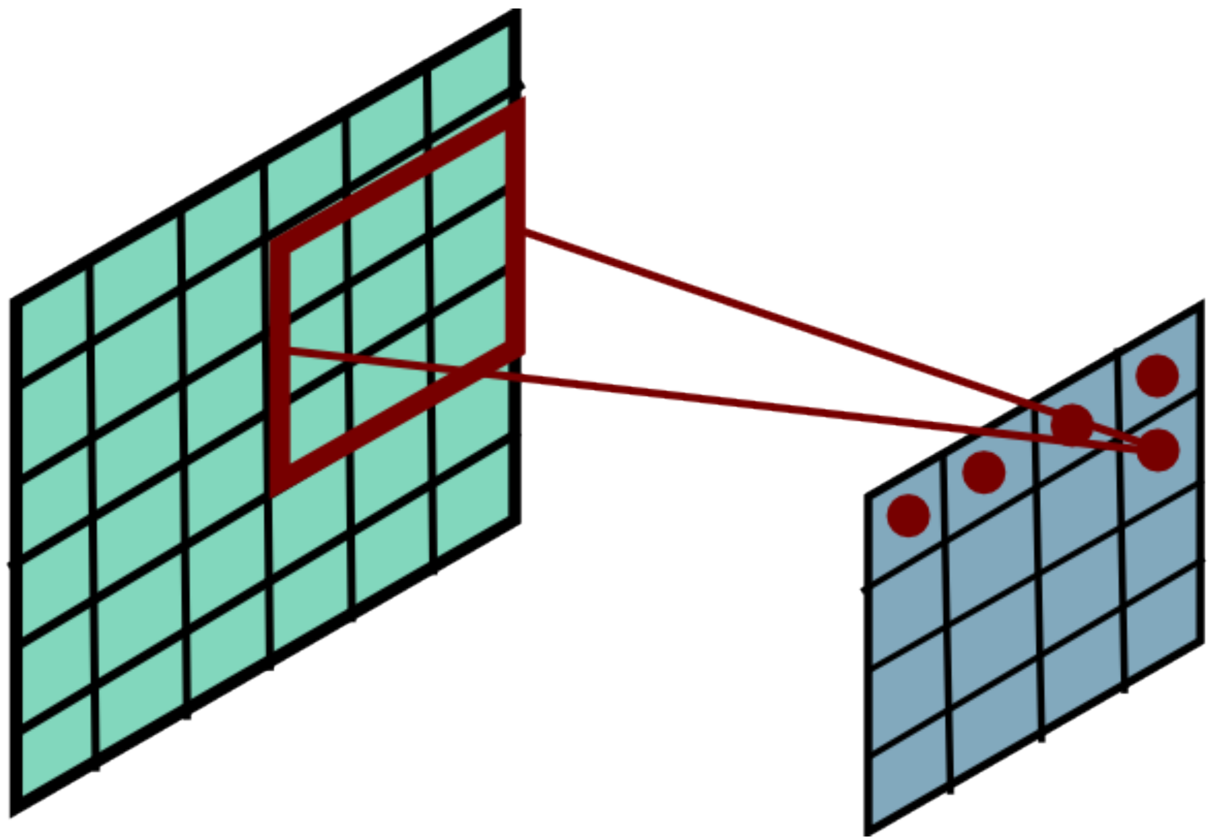
Convolution



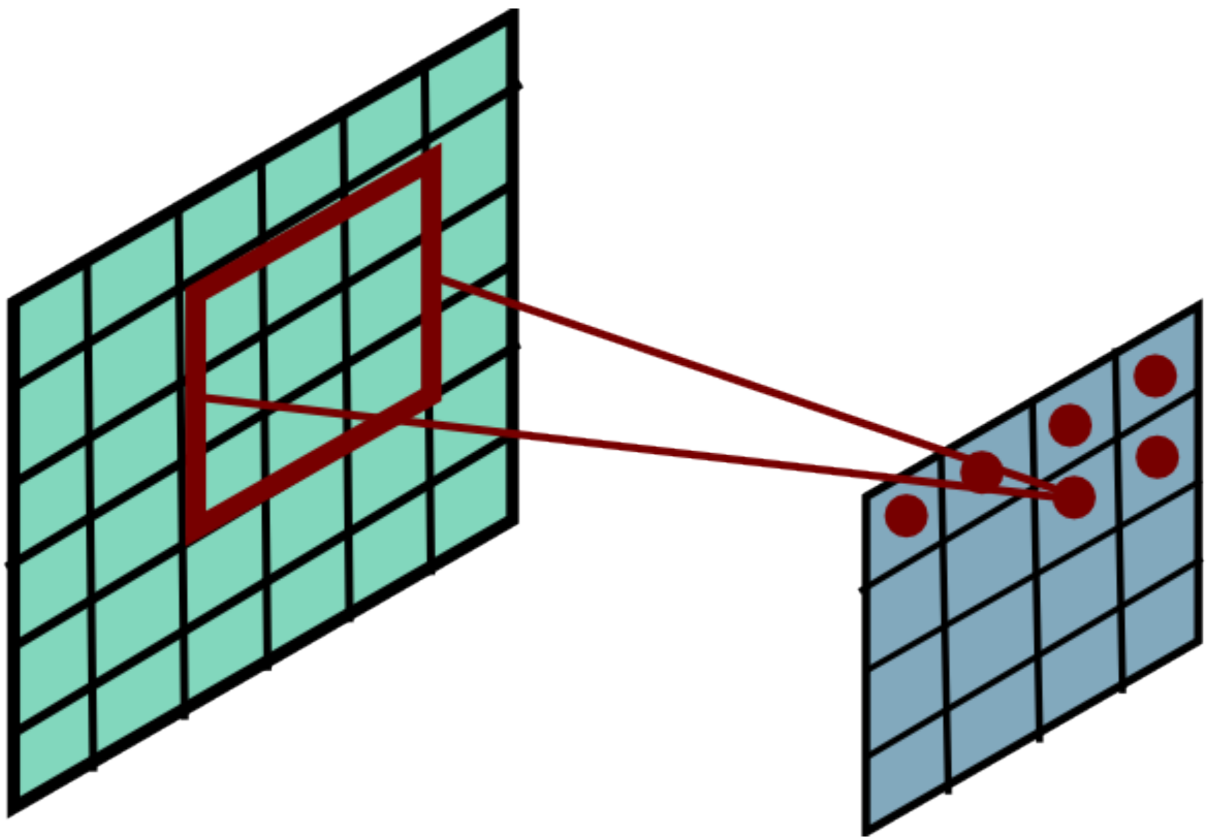
Convolution



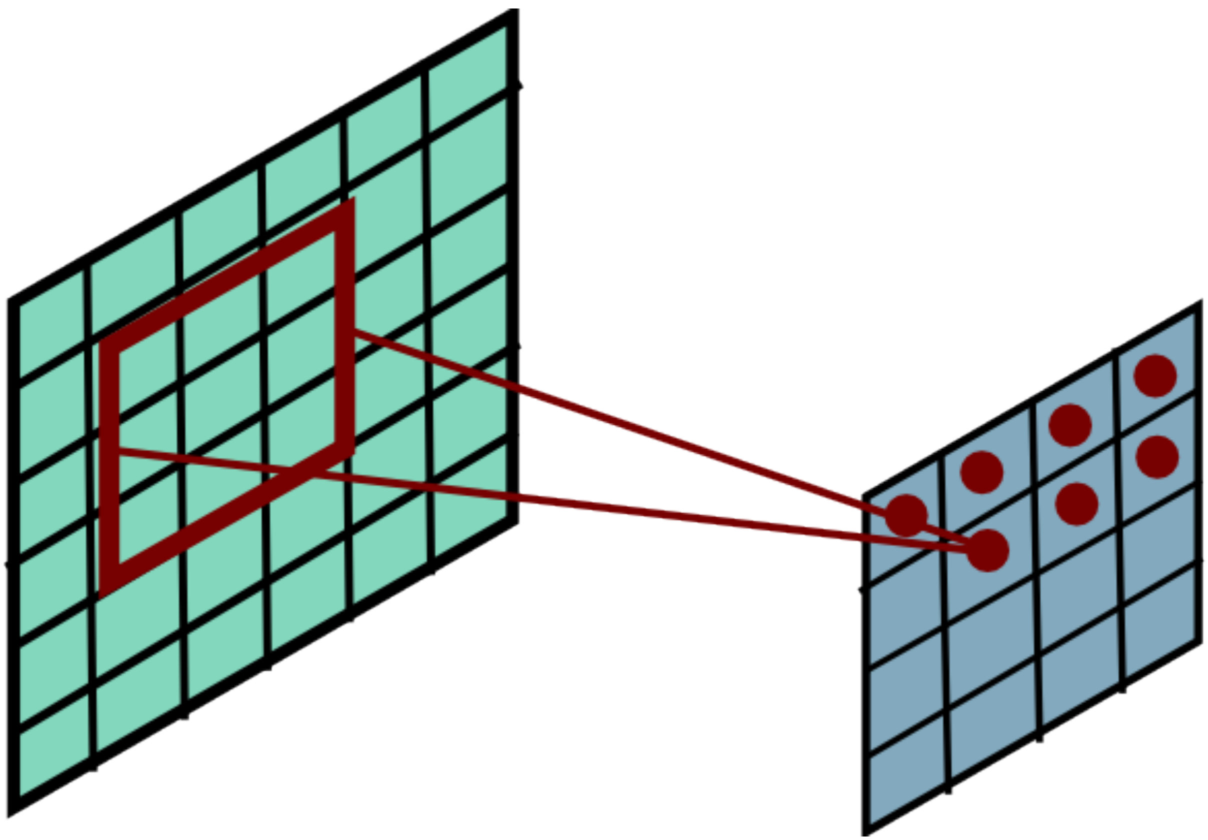
Convolution



Convolution

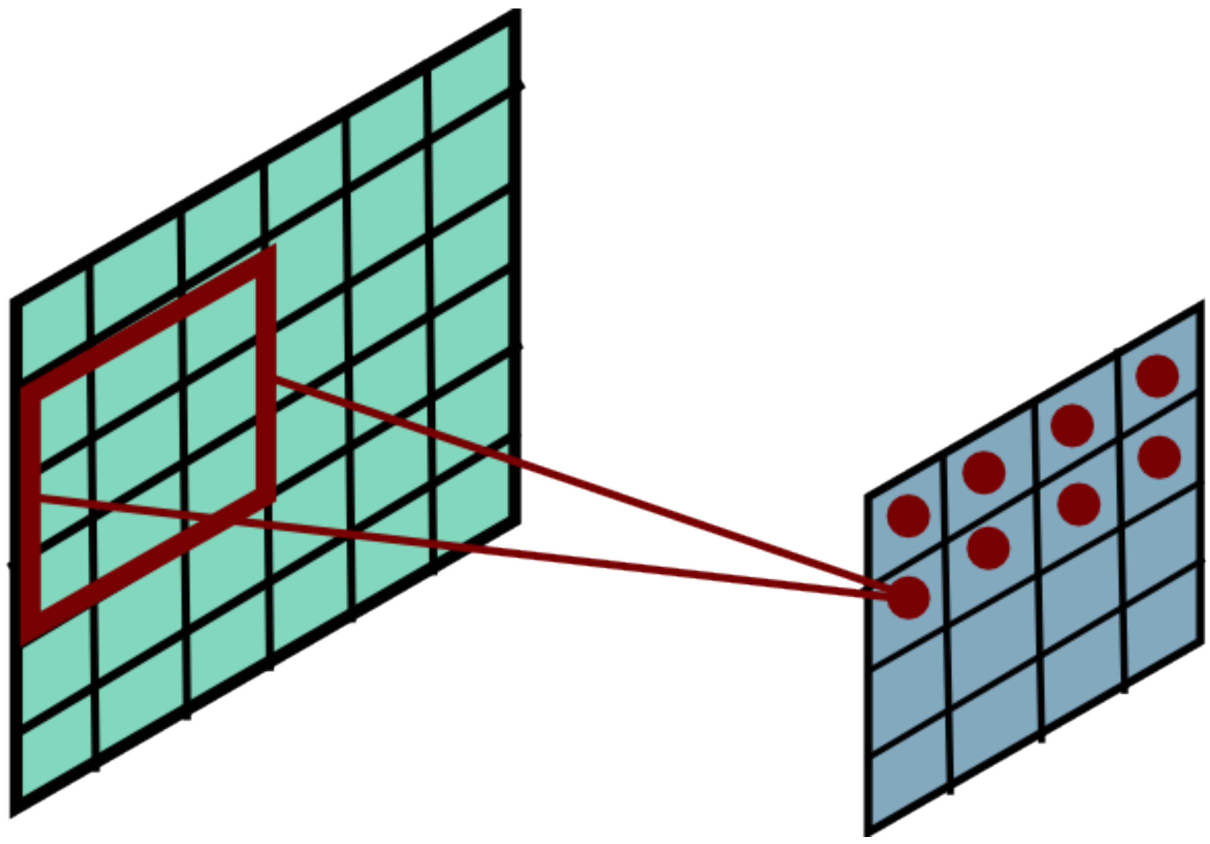


Convolution



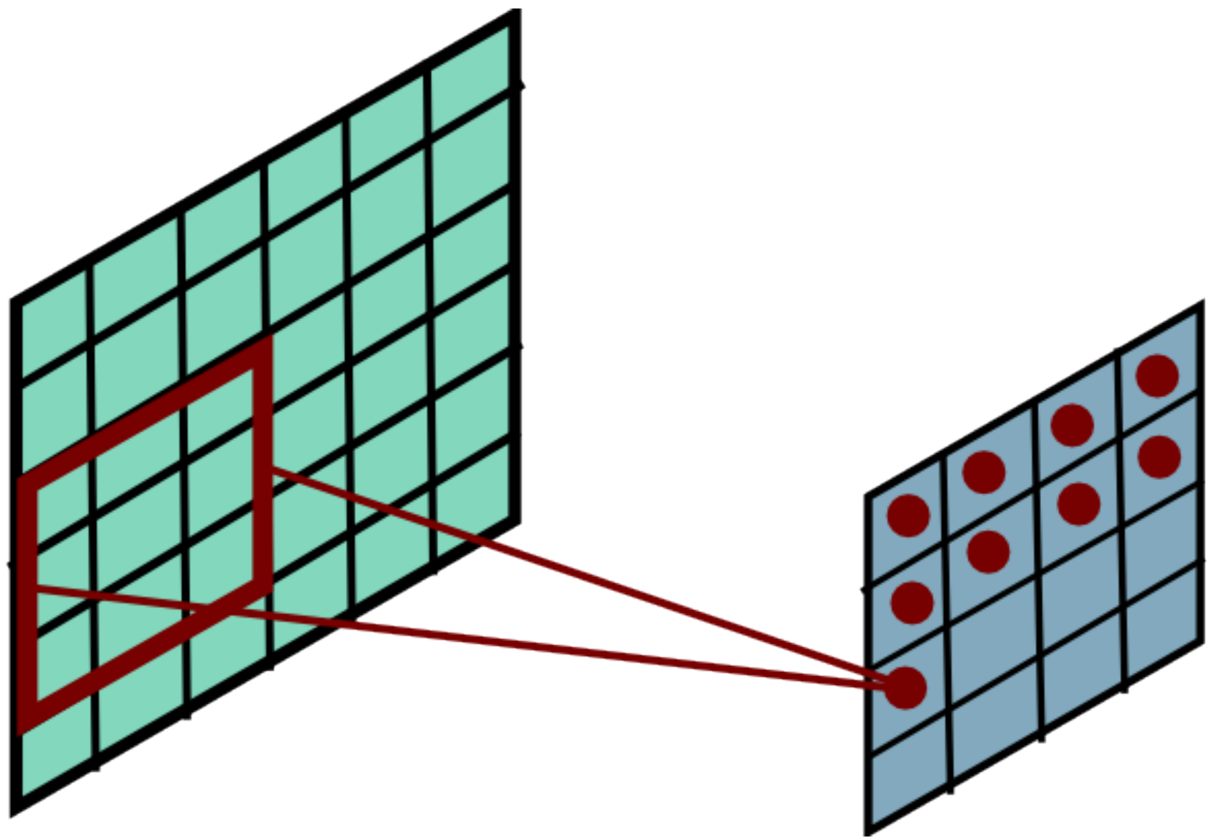
Slide Credit: Marc'Aurelio Ranzato

Convolution

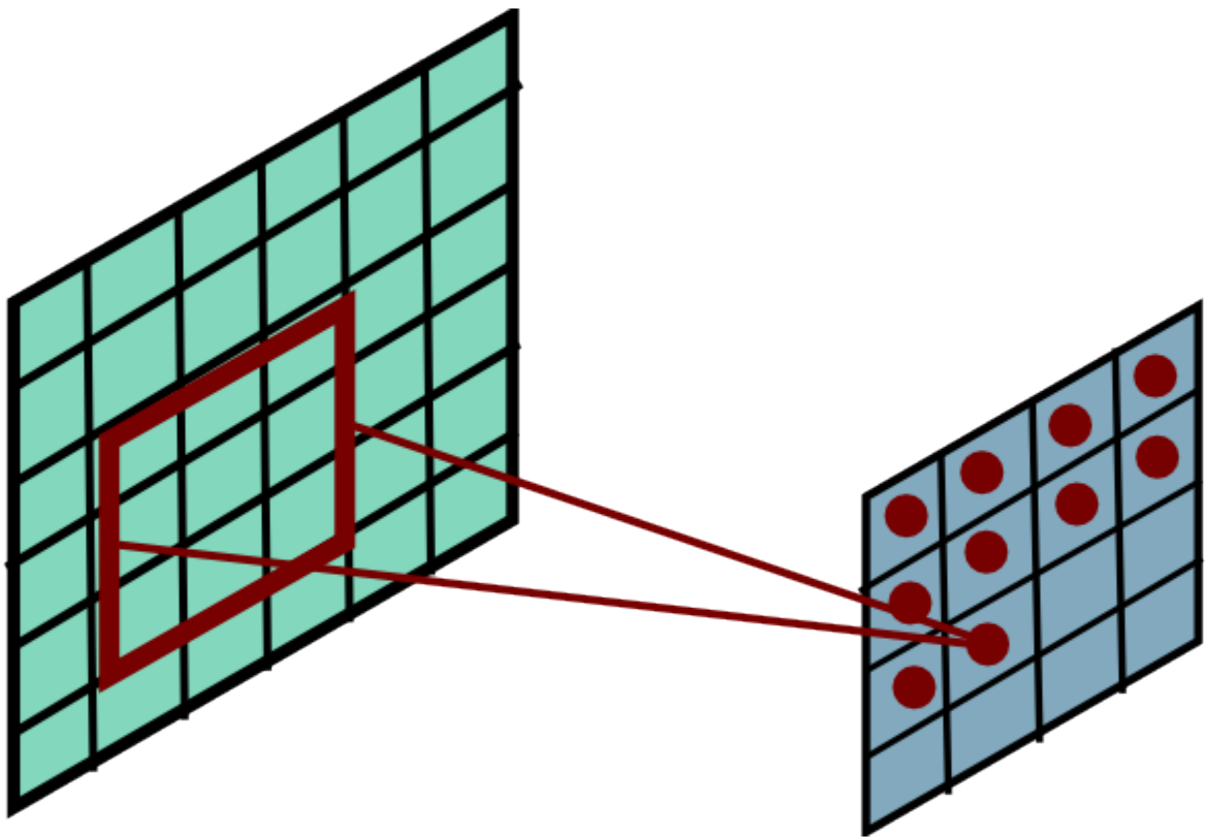


Slide Credit: Marc'Aurelio Ranzato

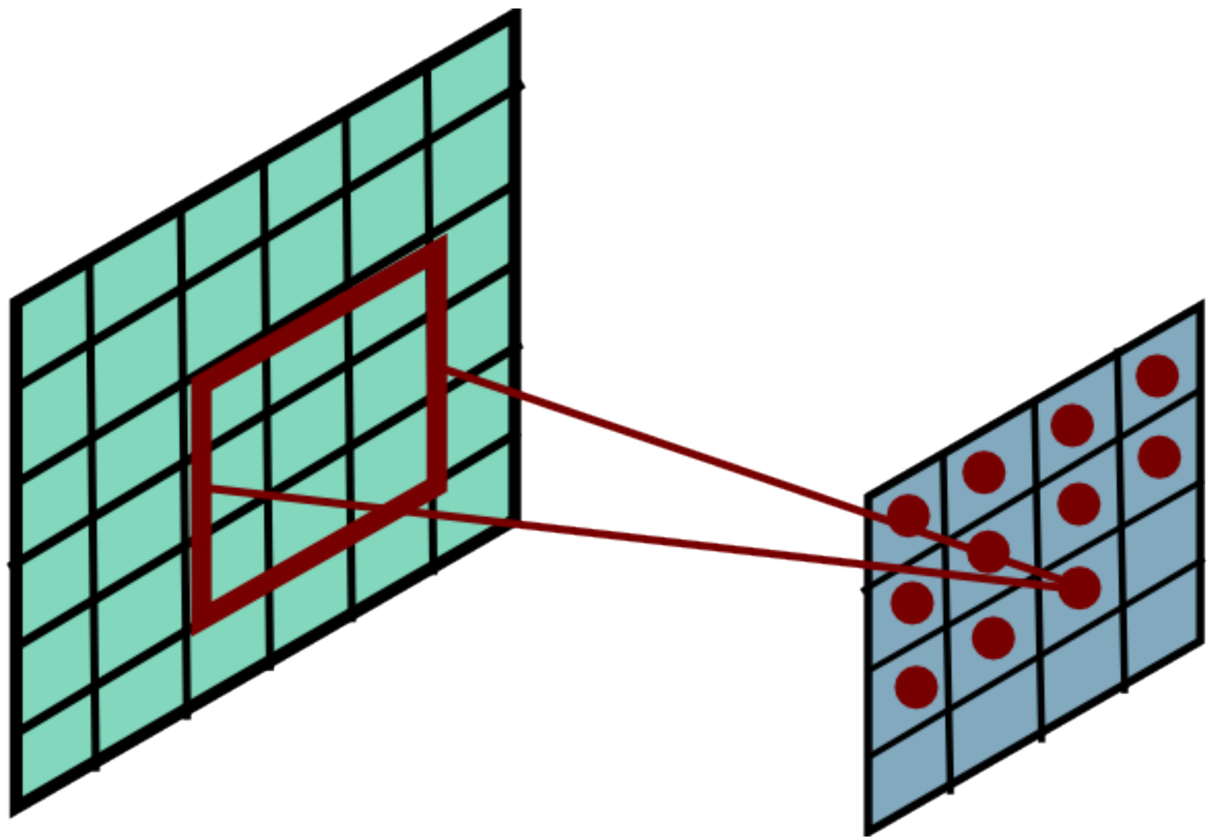
Convolution



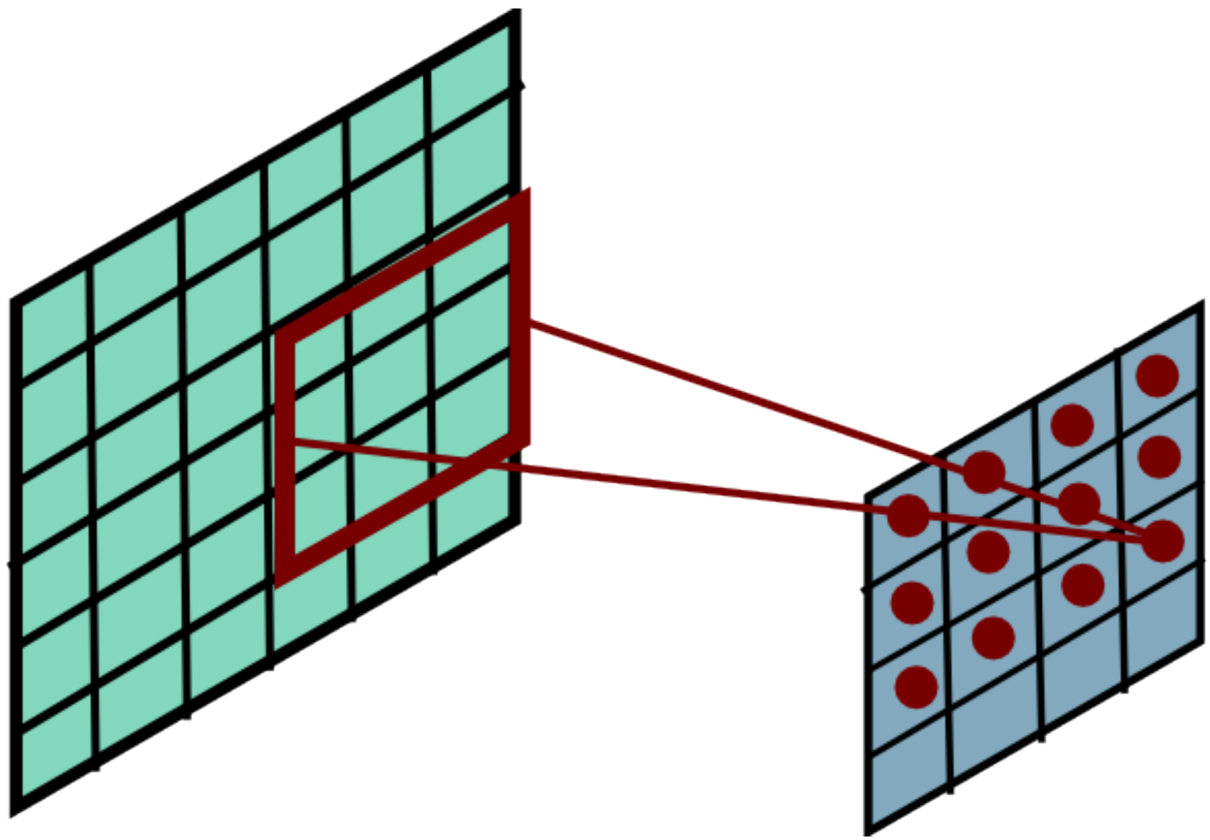
Convolution



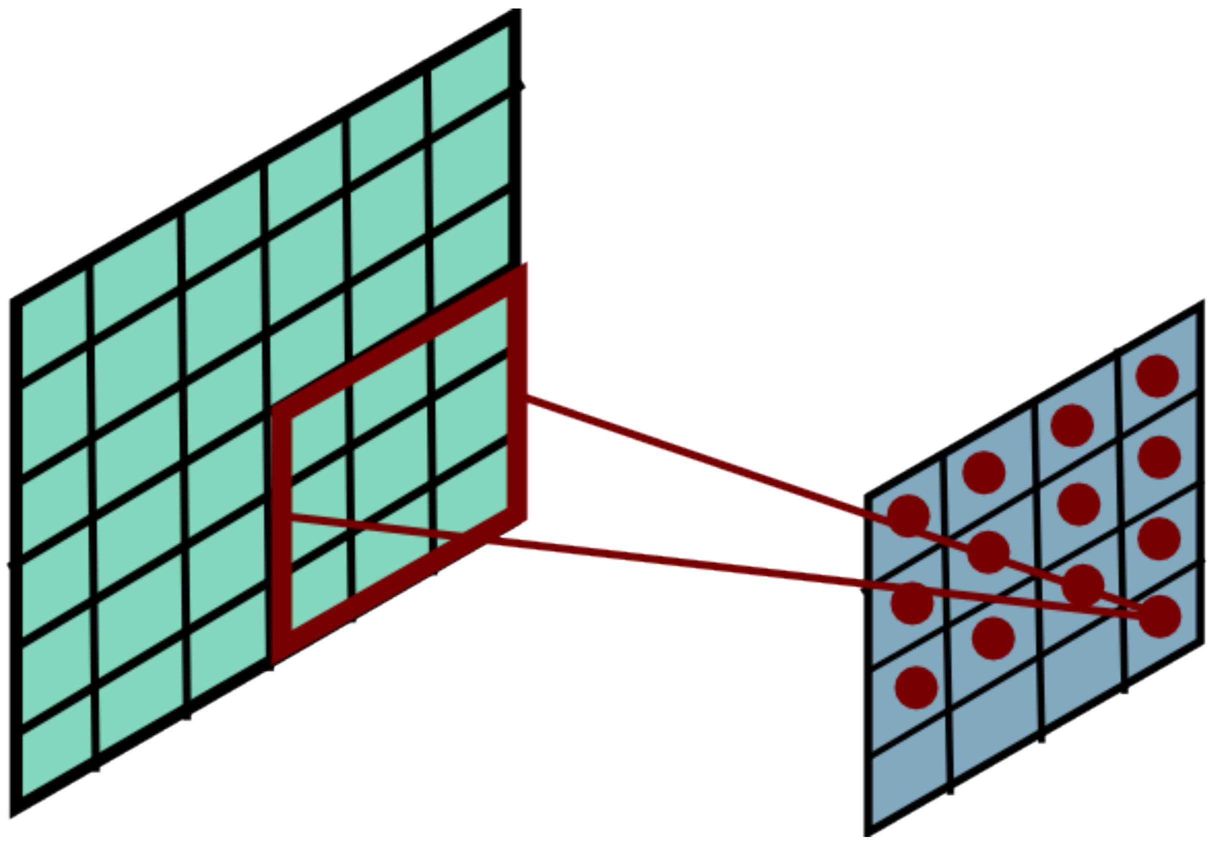
Convolution



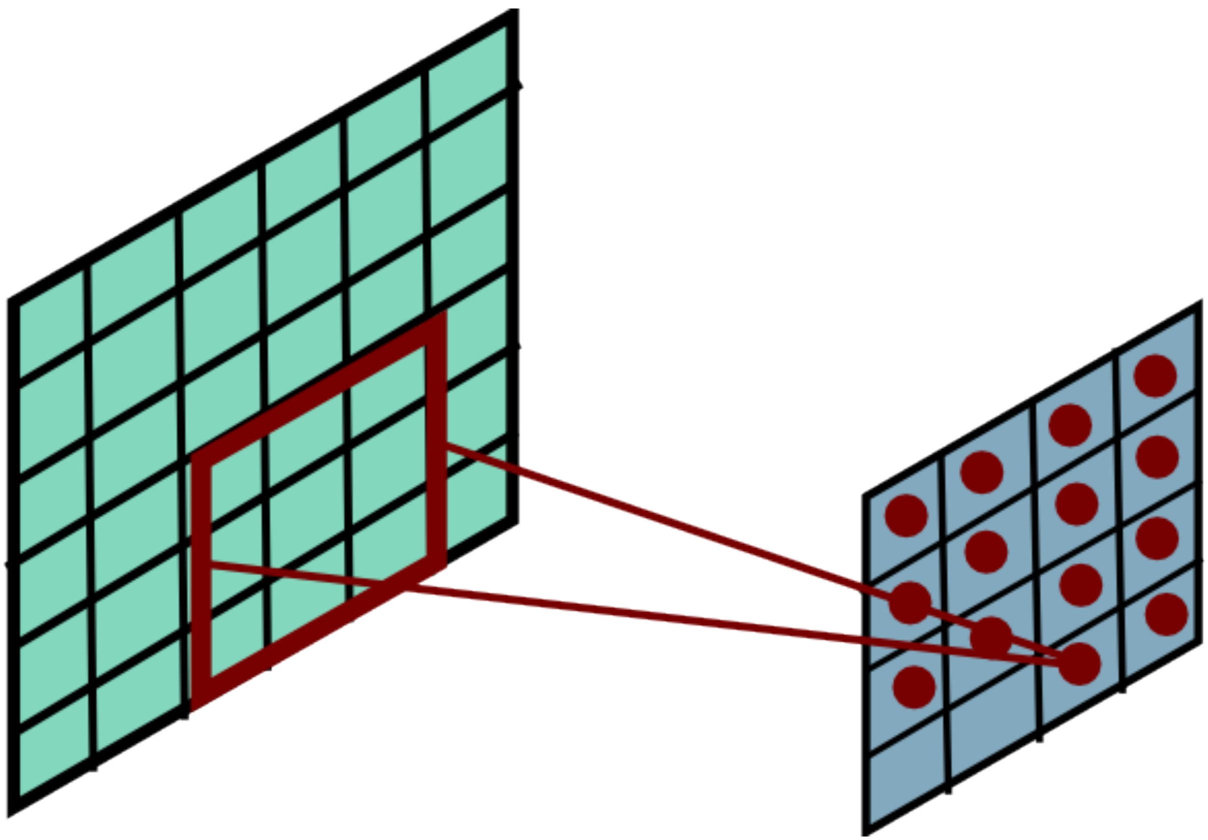
Convolution



Convolution

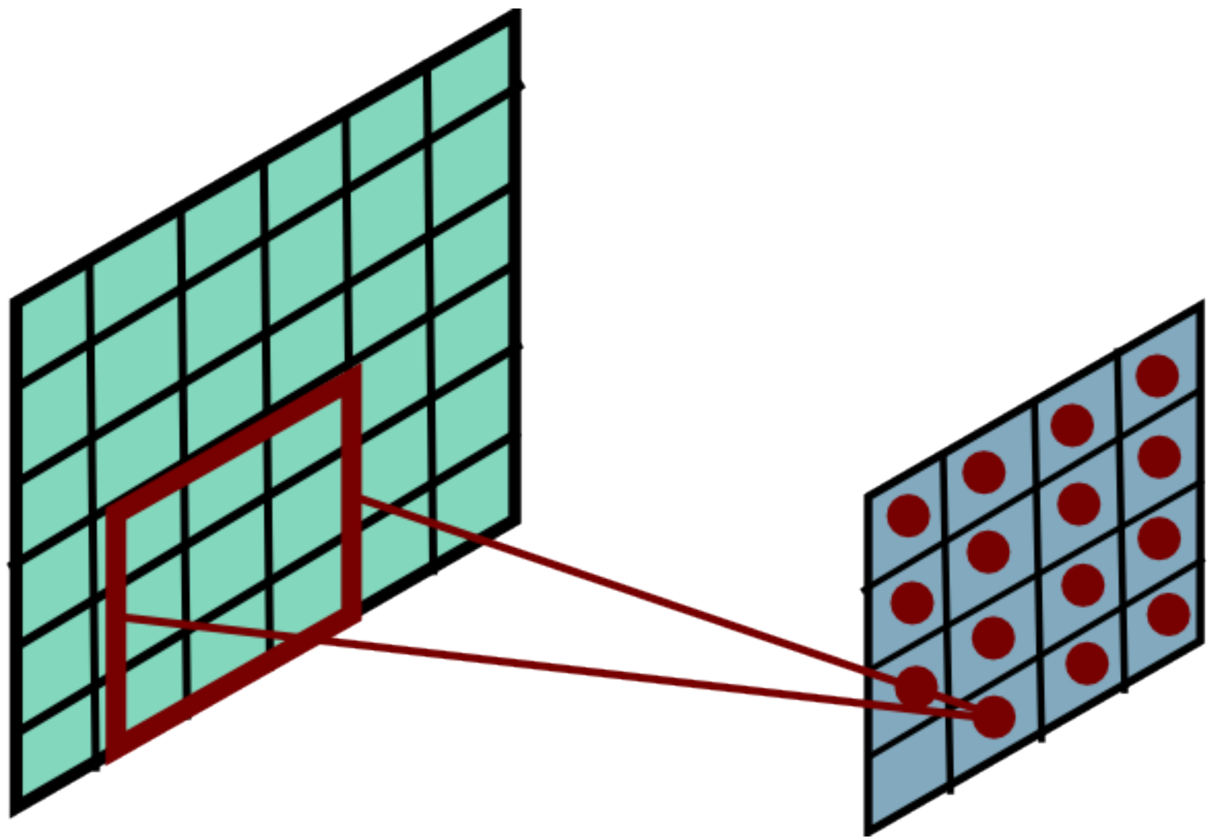


Convolution

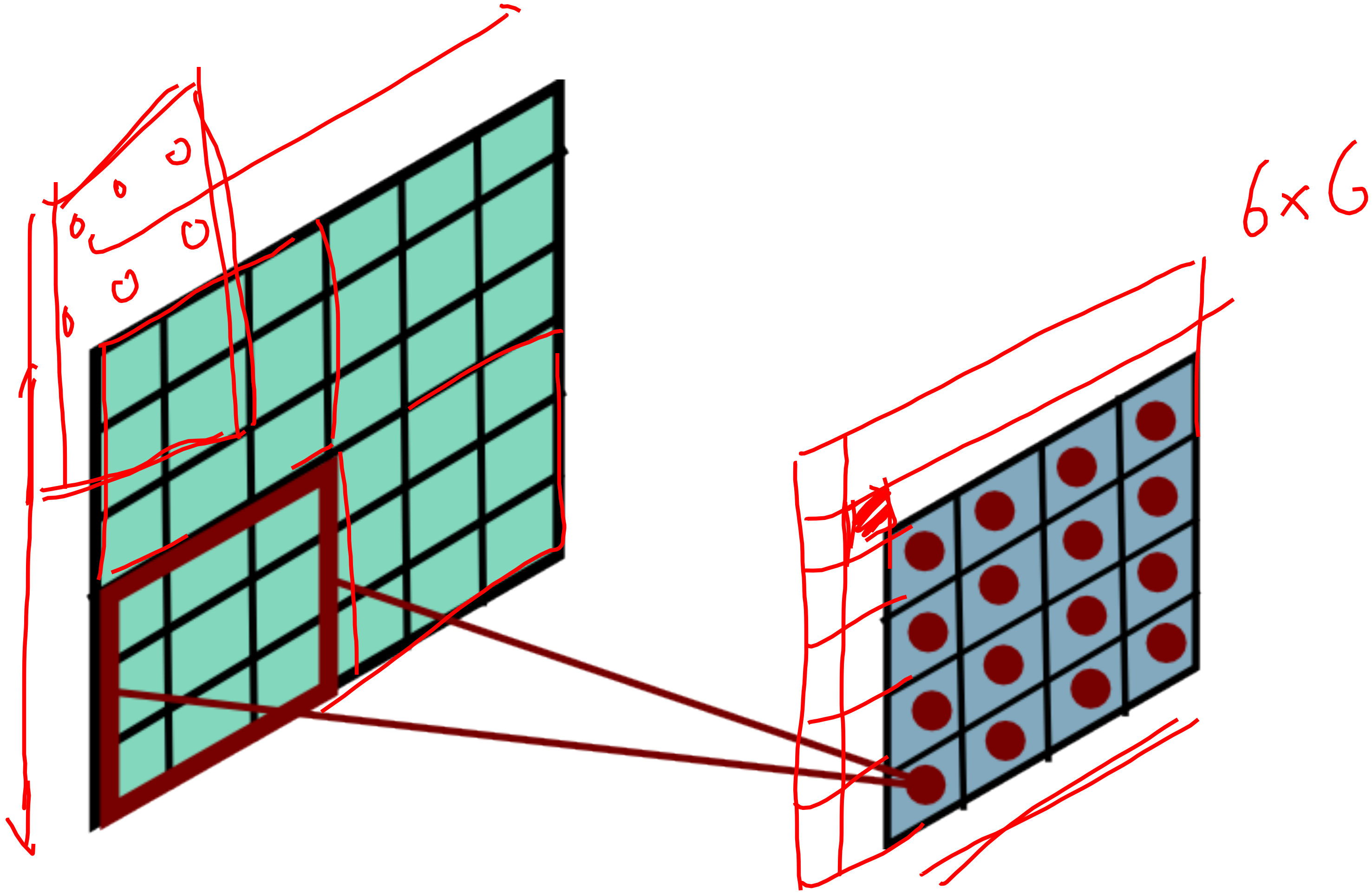


Slide Credit: Marc'Aurelio Ranzato

Convolution

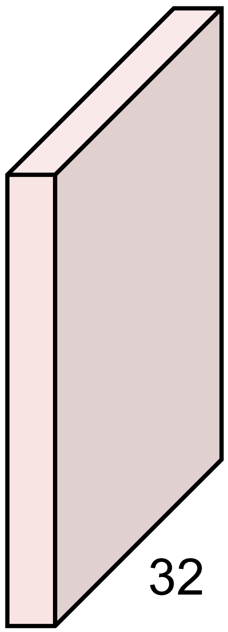


Convolution

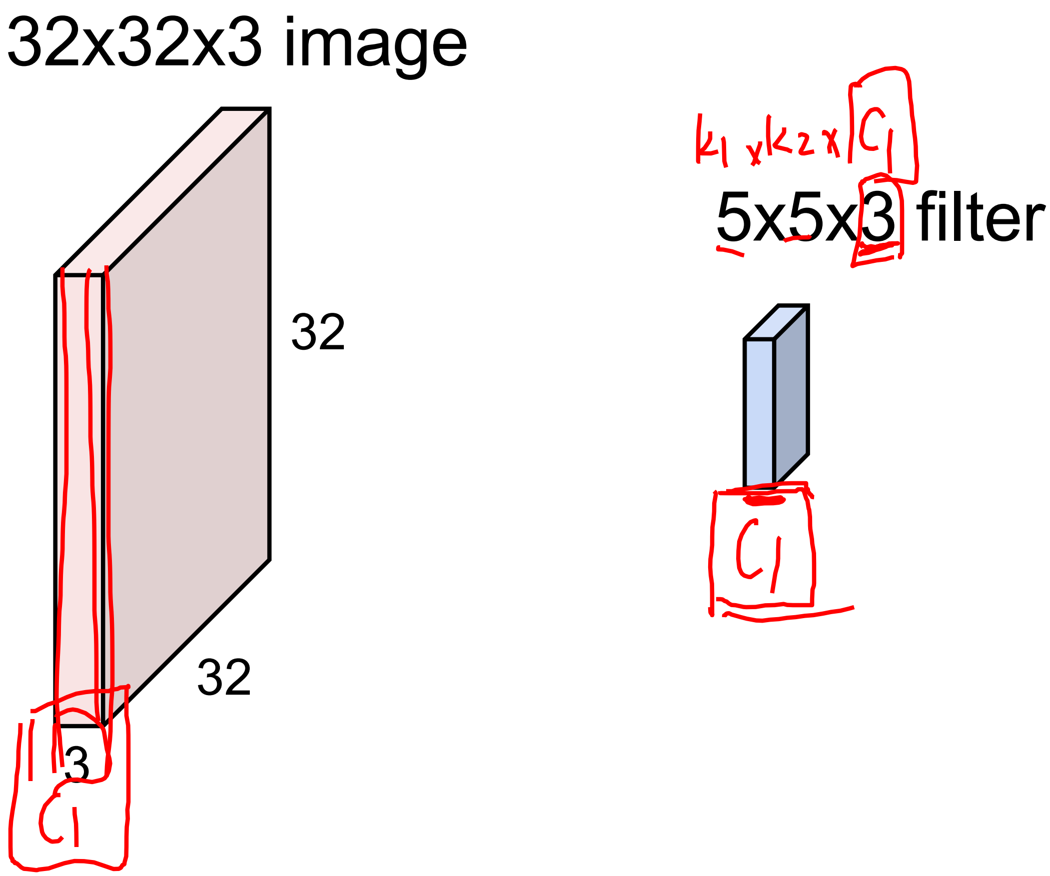


Convolution Layer

32x32x3 image -> preserve spatial structure



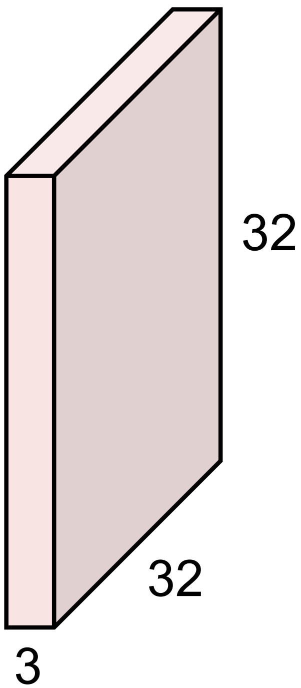
Convolution Layer



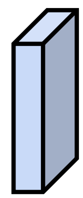
Convolution Layer

Filters always extend the full depth of the input volume

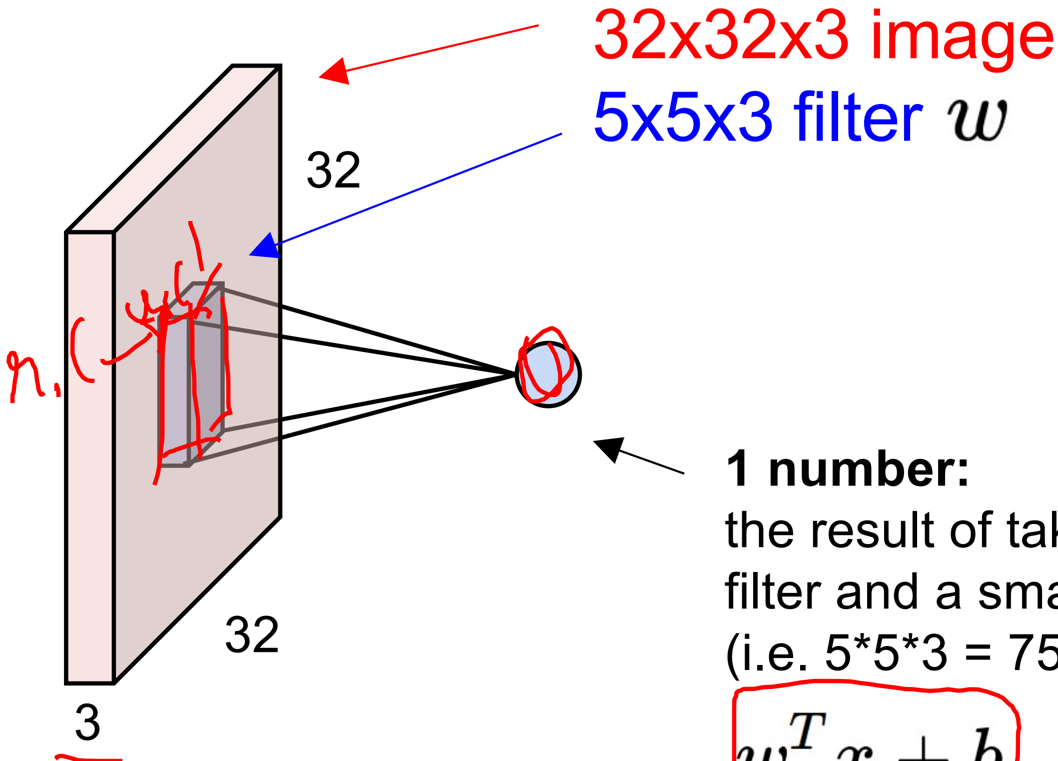
32x32x3 image



5x5x3 filter

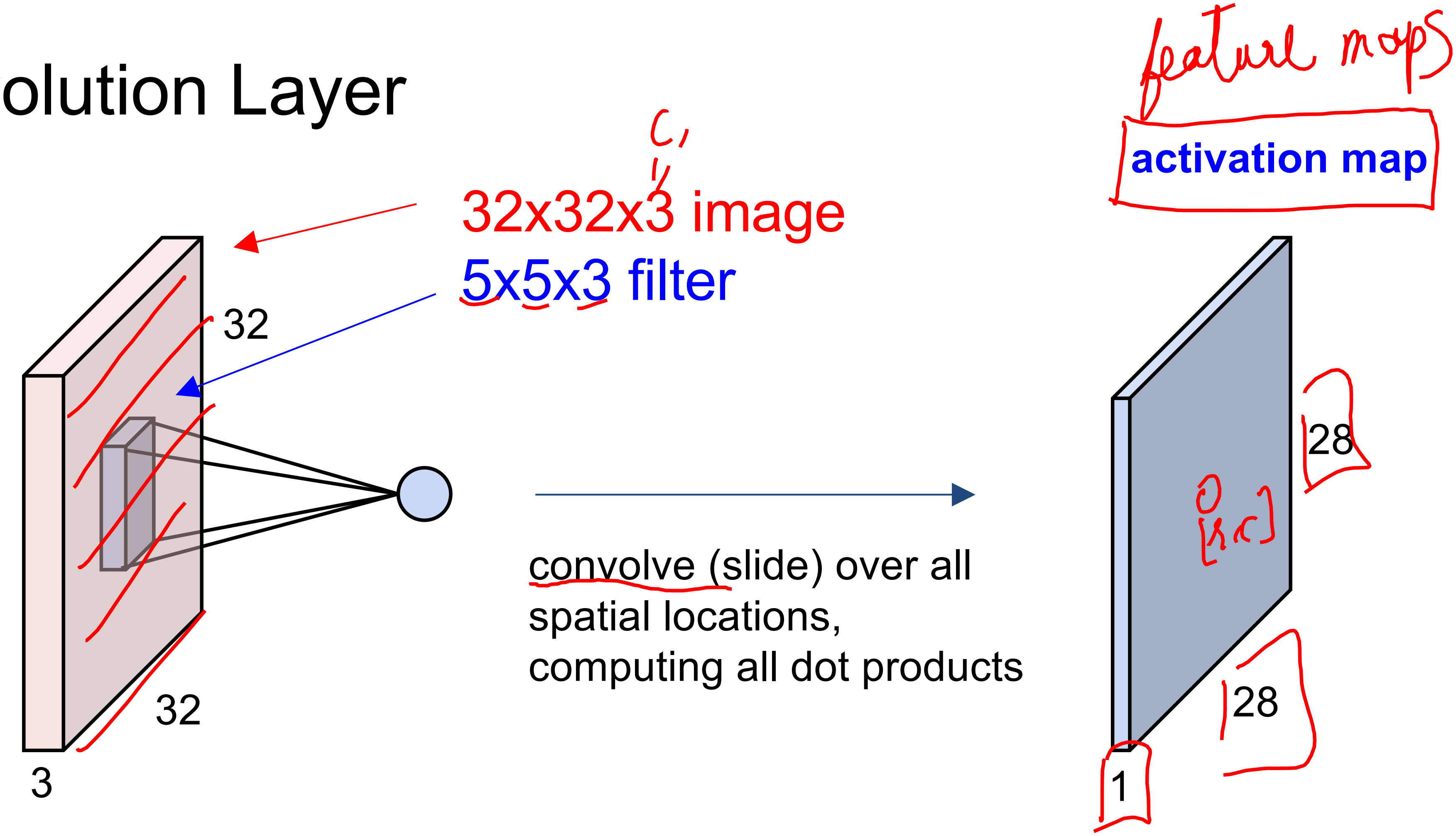


Convolution Layer



$$w^T x + b$$

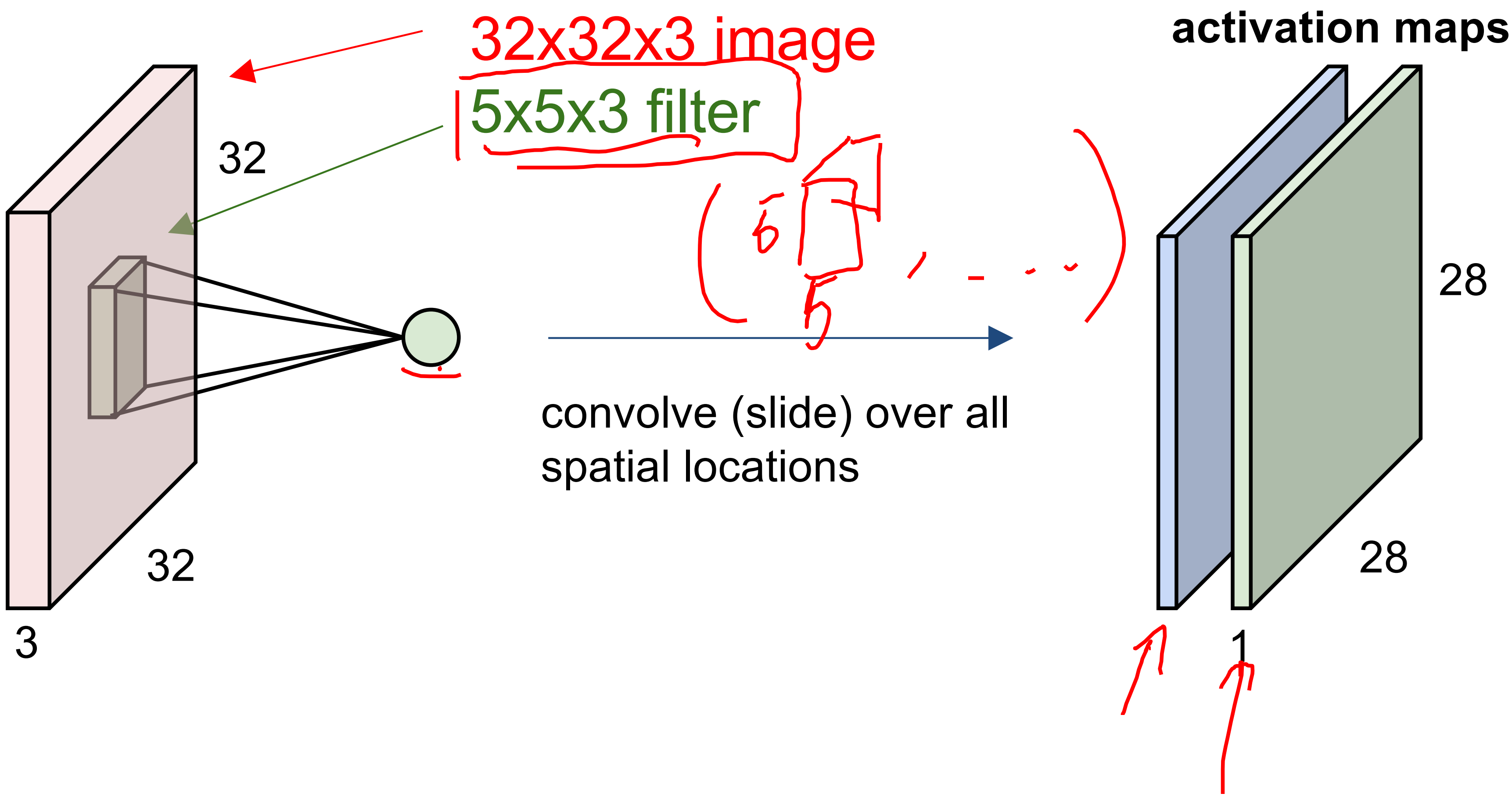
Convolution Layer



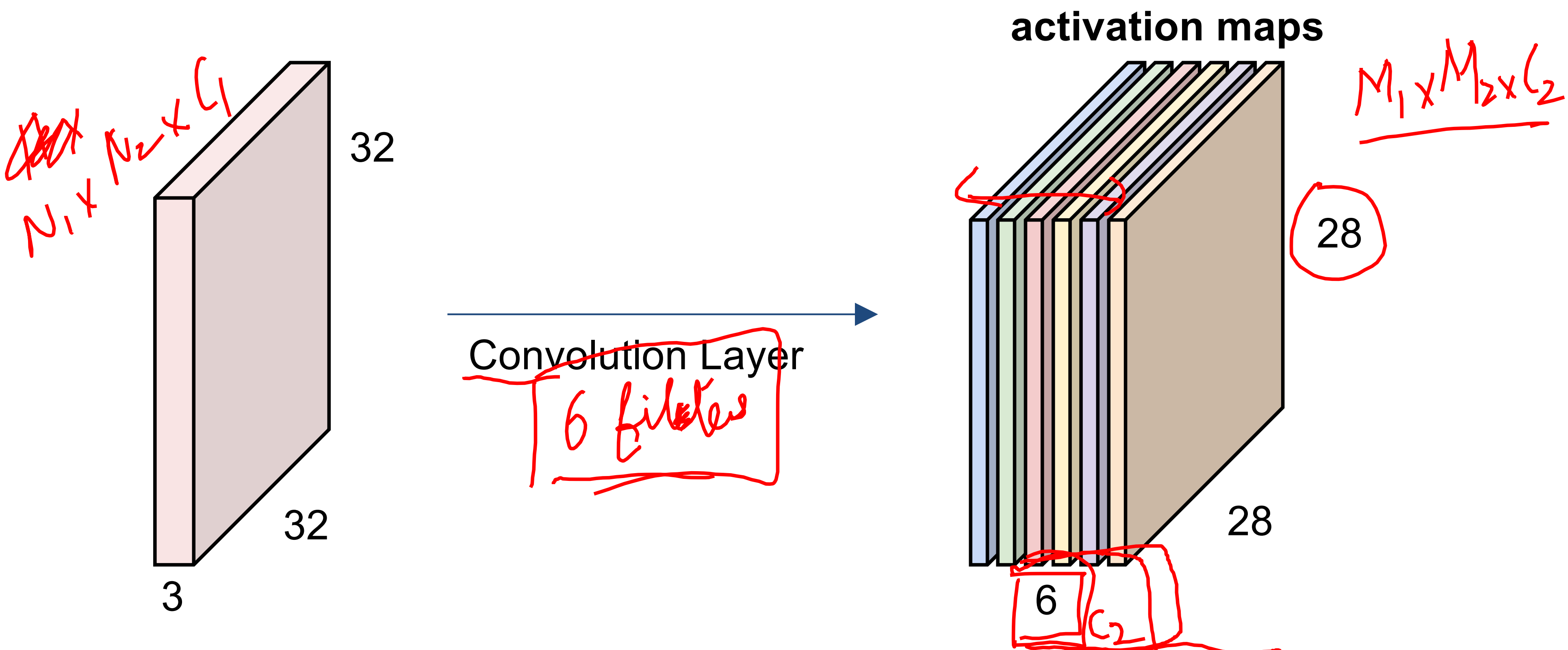
Slide Credit: Fei-Fei Li, Justin Johnson, Serena Yeung, CS 231n

Convolution Layer

consider a second, green filter

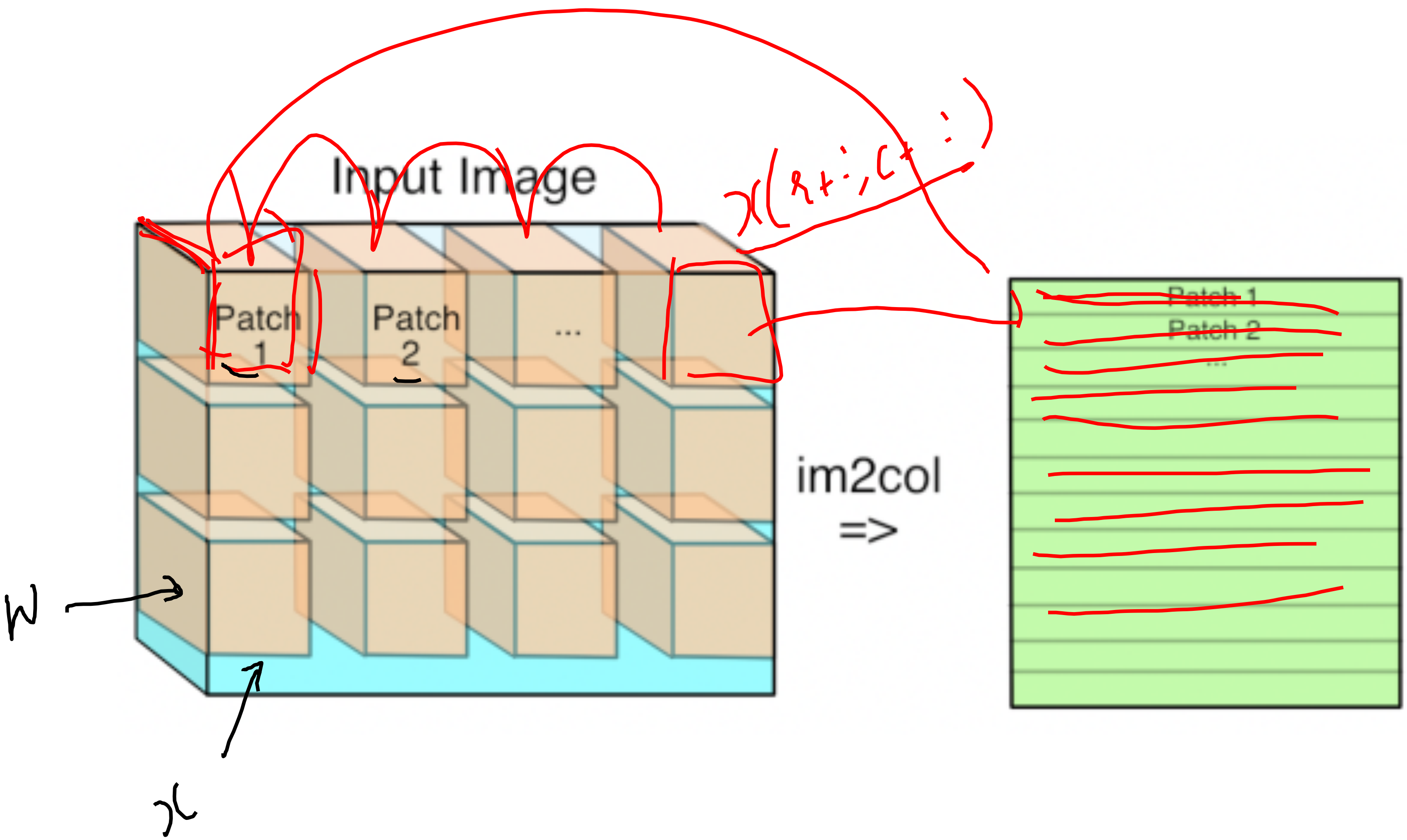


For example, if we had 6 5x5 filters, we'll get 6 separate activation maps:

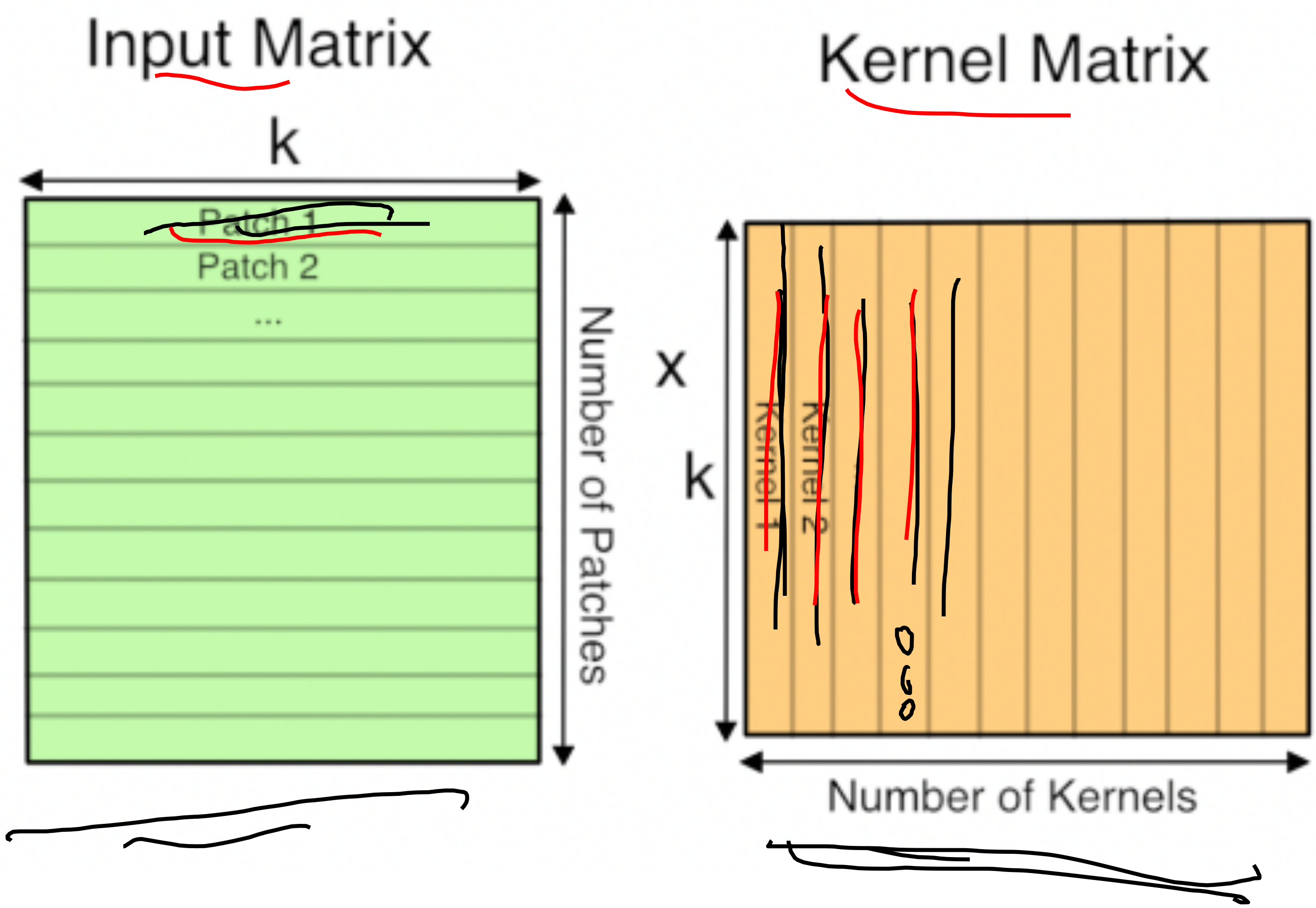


We stack these up to get a "new image" of size 28x28x6!

Im2Col

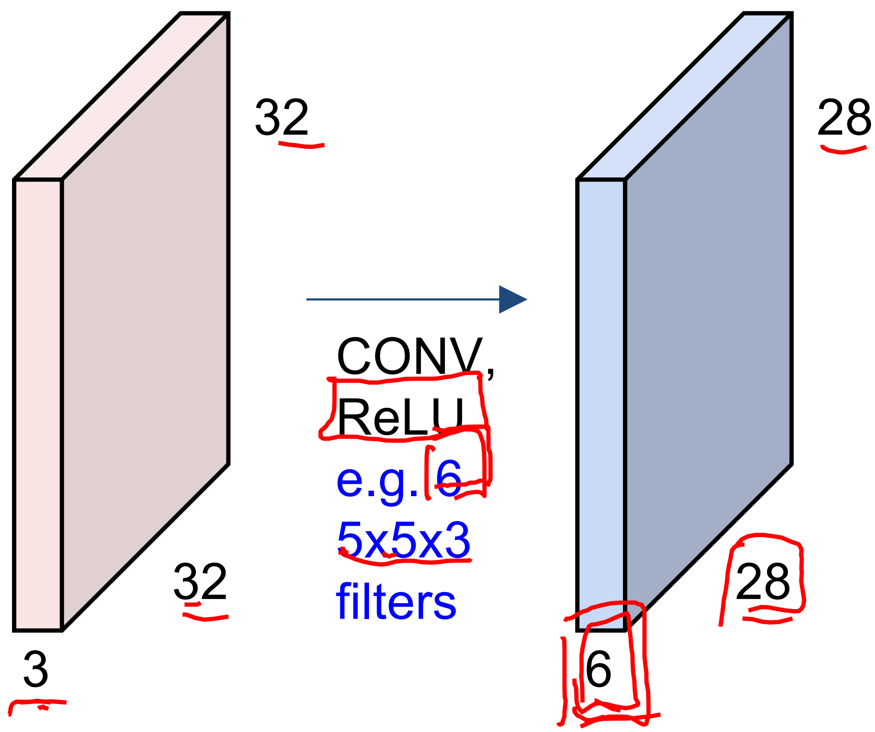


GEMM

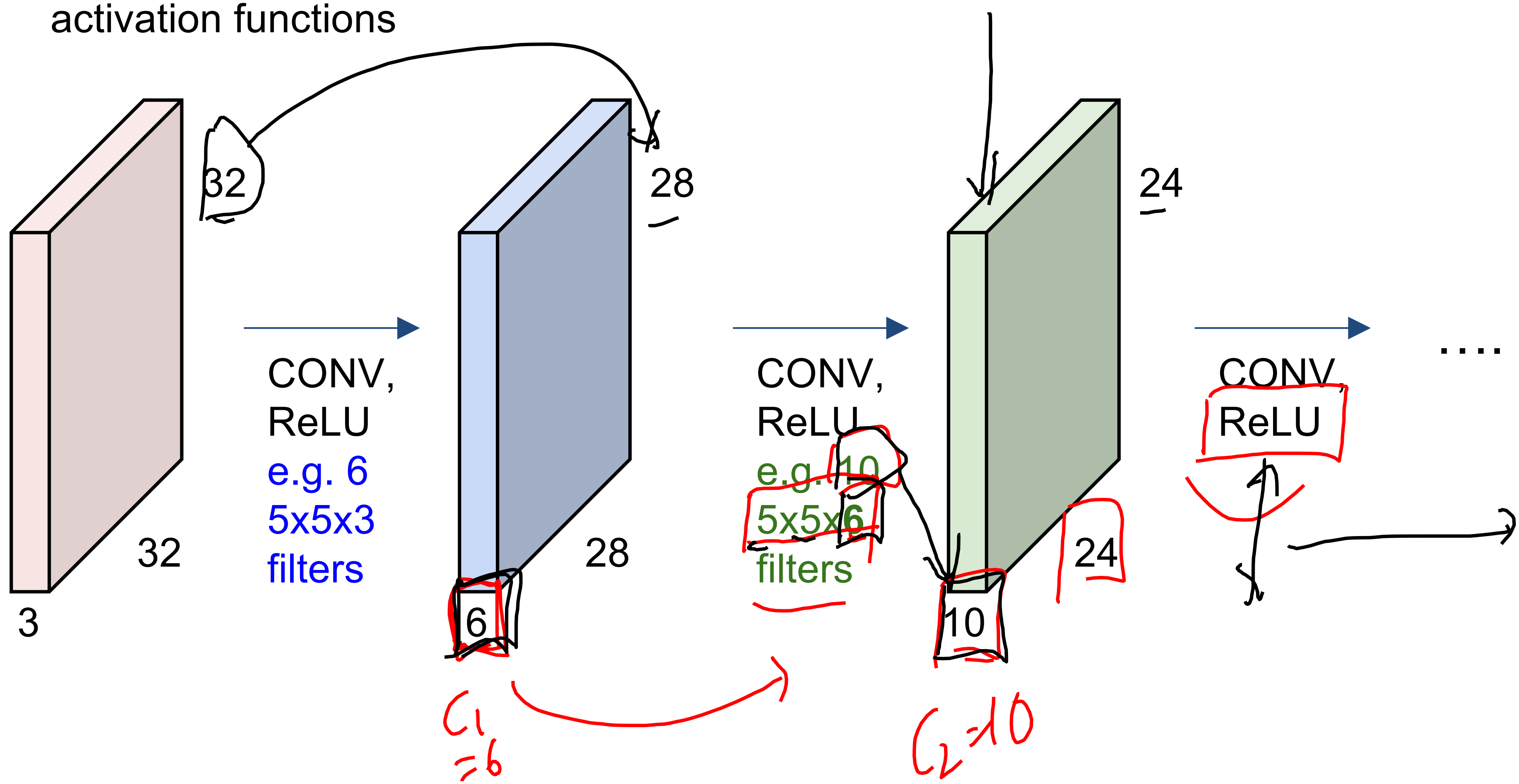


Input Matrix
k
Patch 1
Patch 2
...
Number of Patches
x
Kernel Matrix
k
Kernel 1
Kernel 2
Number of Kernels

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



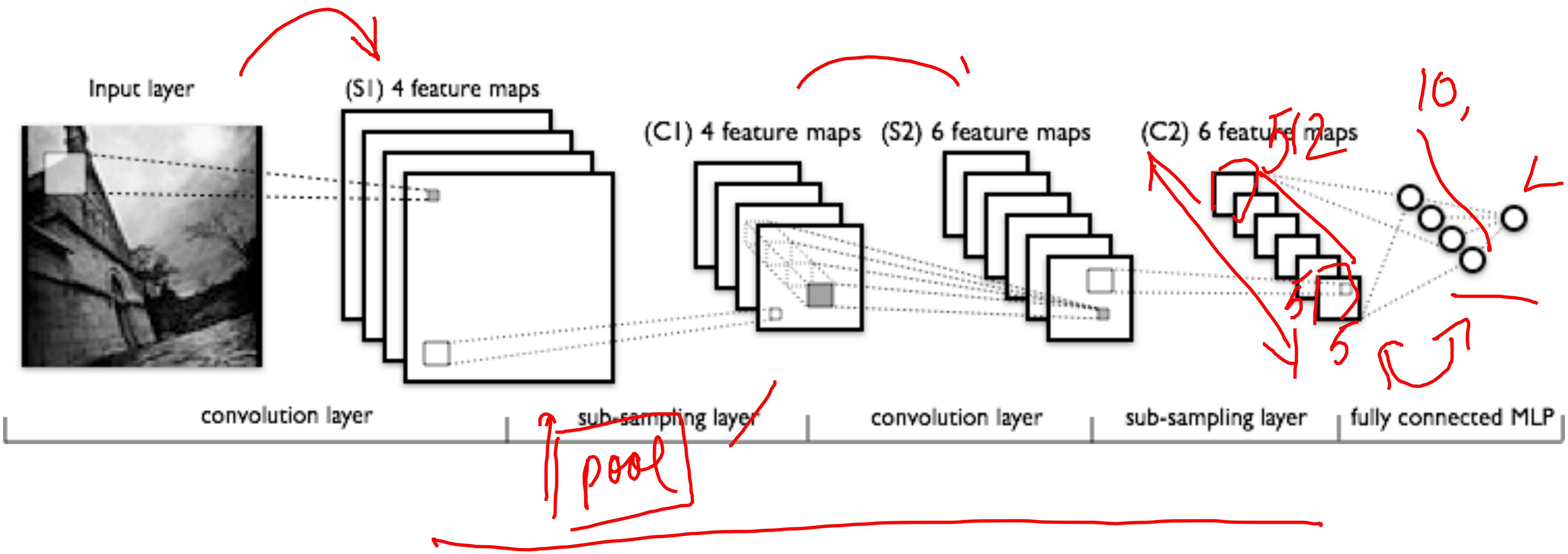
Preview: ConvNet is a sequence of Convolutional Layers, interspersed with activation functions



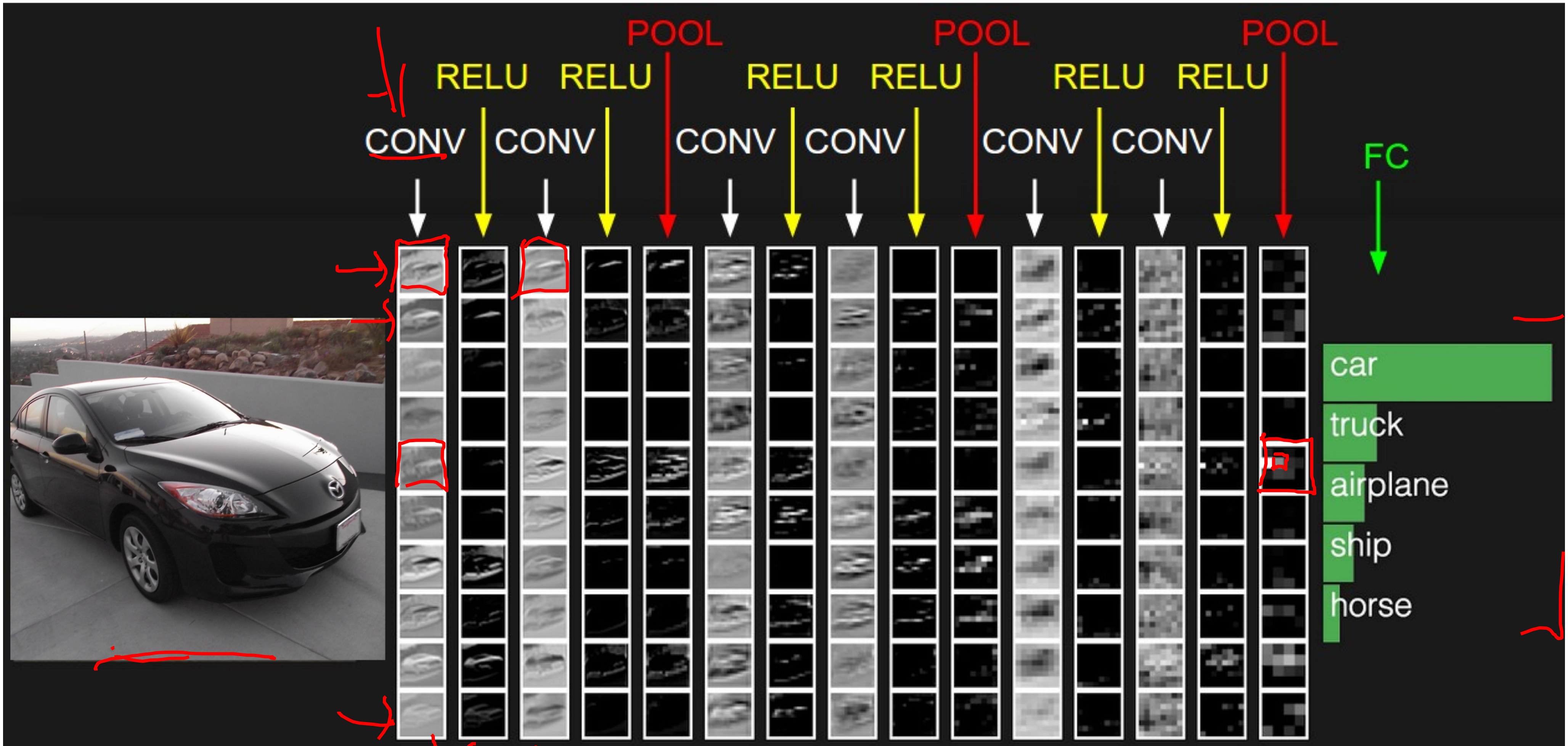
Plan for Today

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

Convolutional Neural Networks



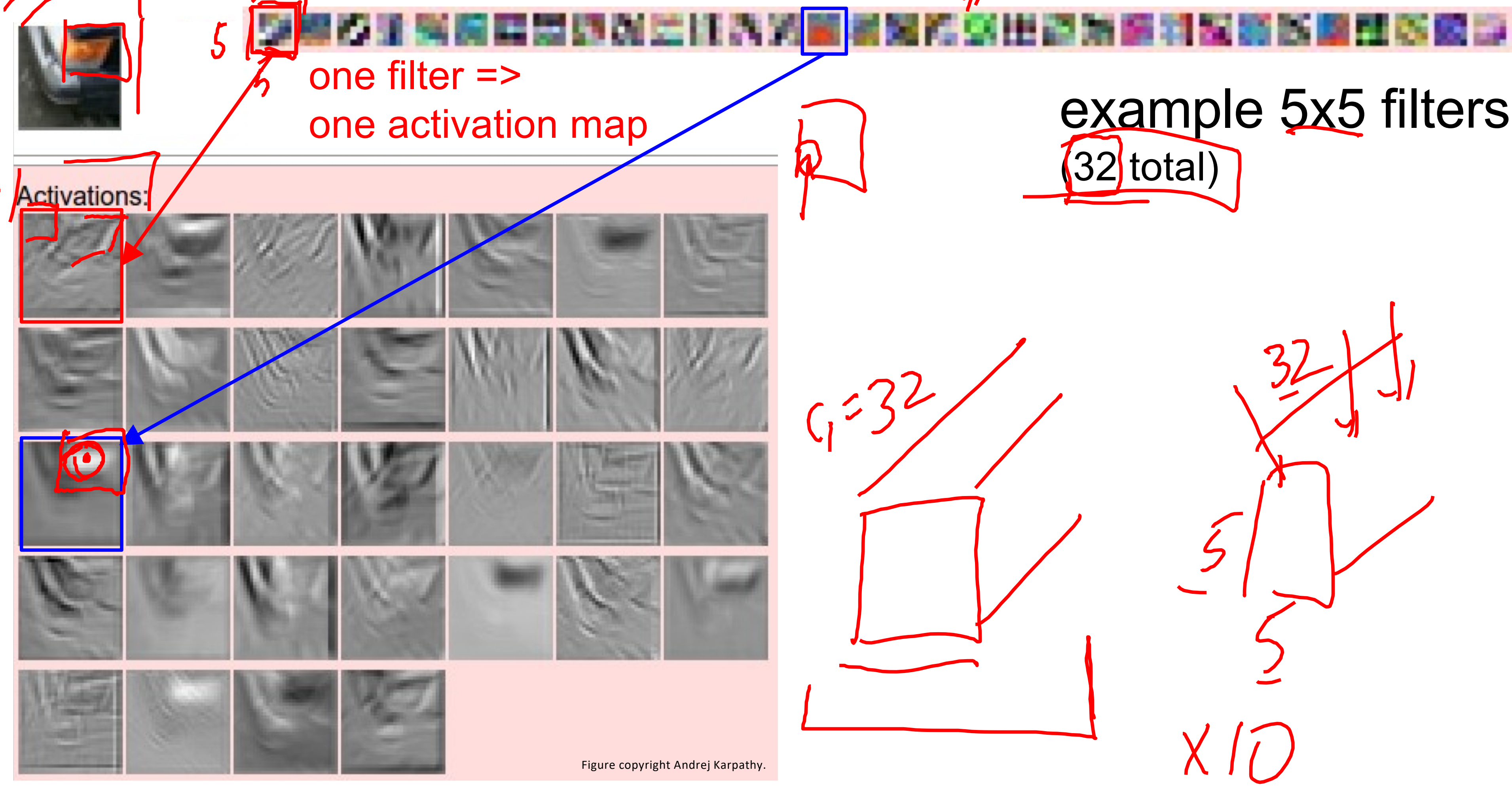
preview:



12

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

$w^* = \arg\min L(W)$



example 5x5 filters
(32 total)

$C_1 = 32$

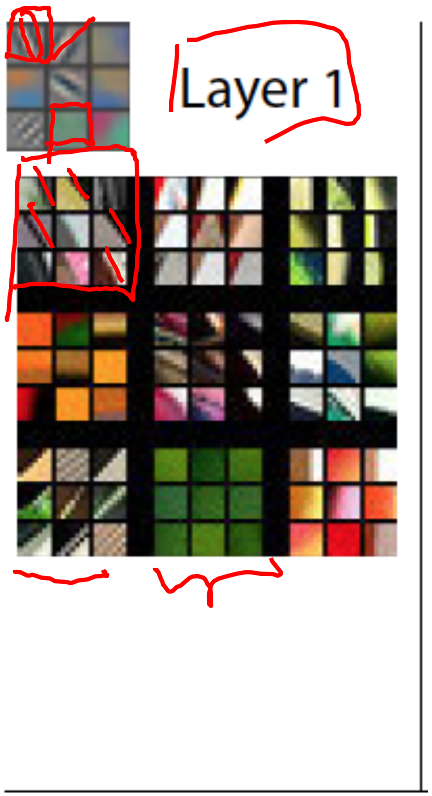
3 5 3 5 d

32 5 5

x10

Layer 2

Visualizing Learned Filters



Visualizing Learned Filters

Ignore

Channel

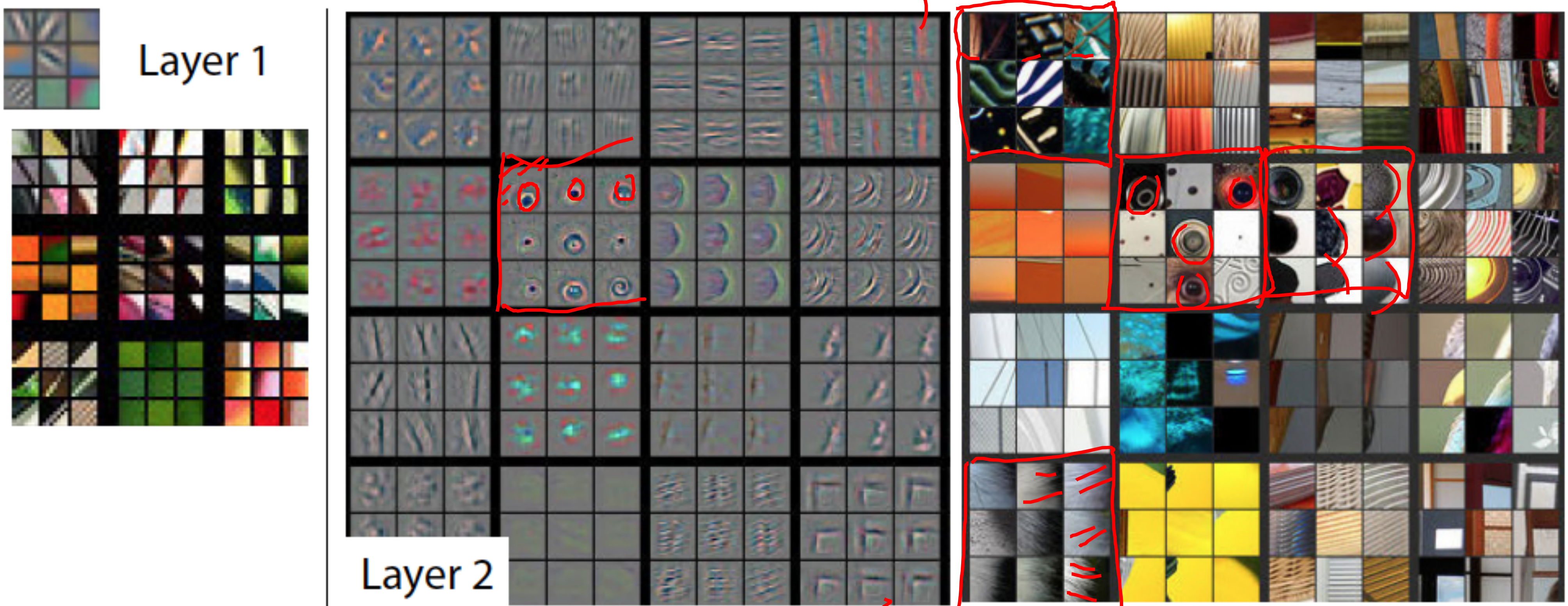
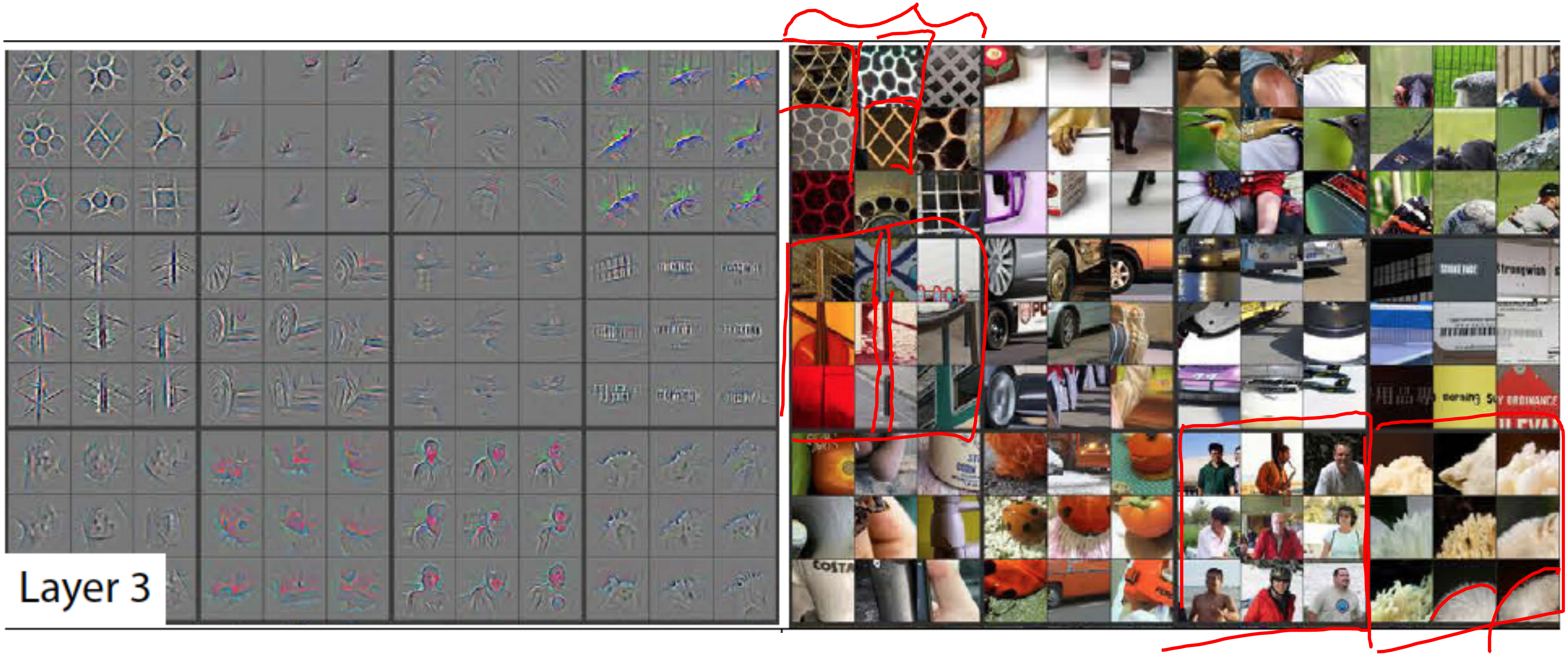


Figure Credit: [Zeiler & Fergus ECCV14]

Visualizing Learned Filters



Visualizing Learned Filters

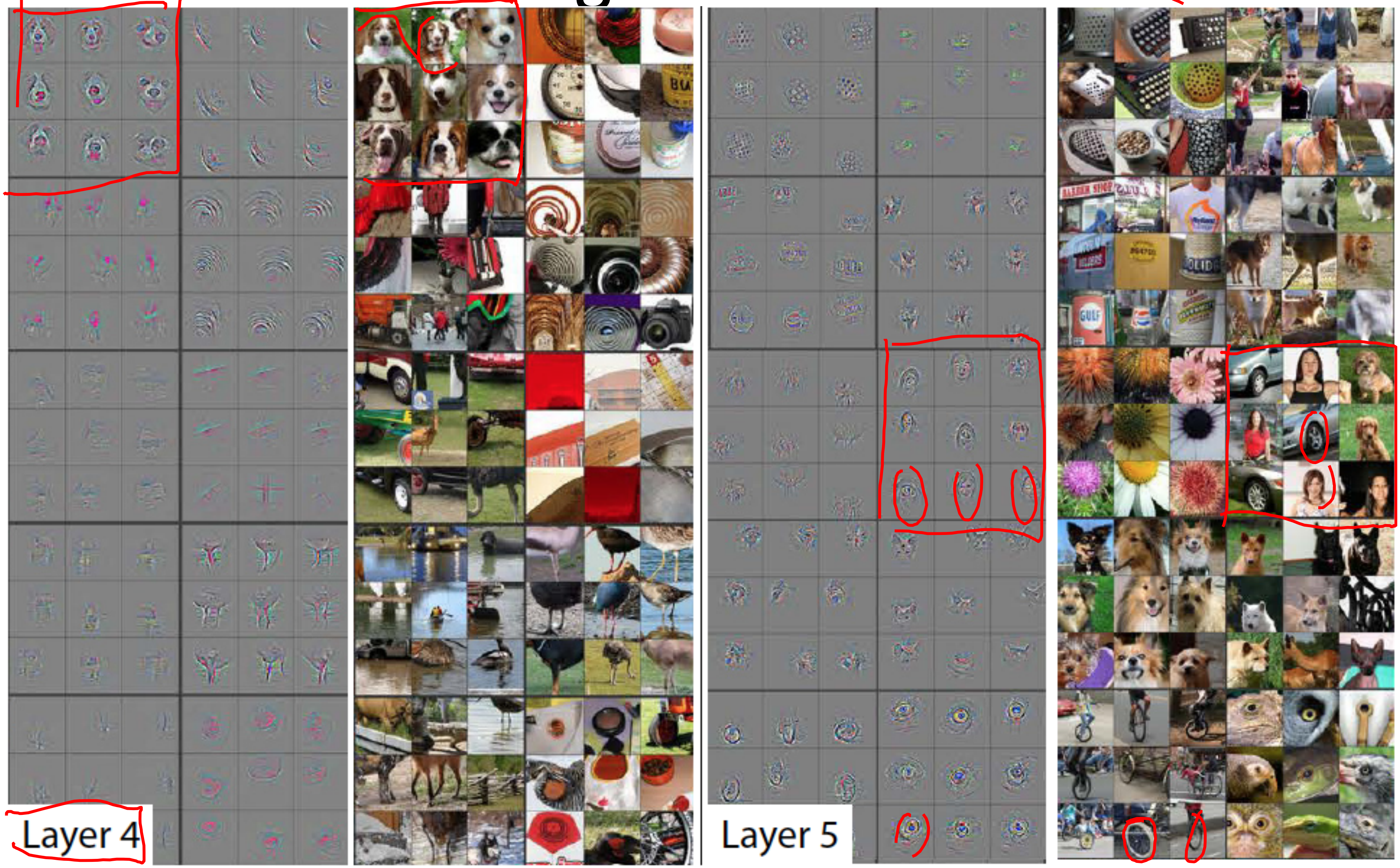
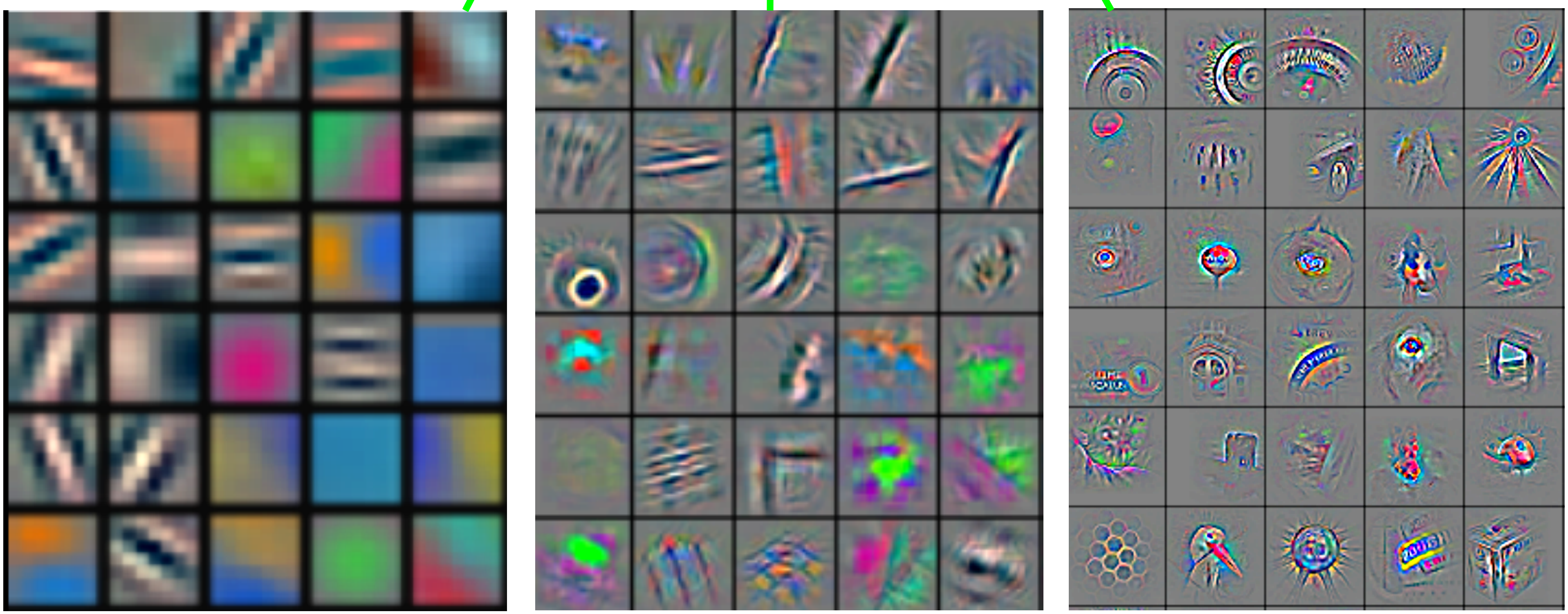


Figure Credit: [Zeiler & Fergus ECCV14]

We can learn image features now!

→ Low-Level Feature → Mid-Level Feature → High-Level Feature → Linear Classifier → "car"



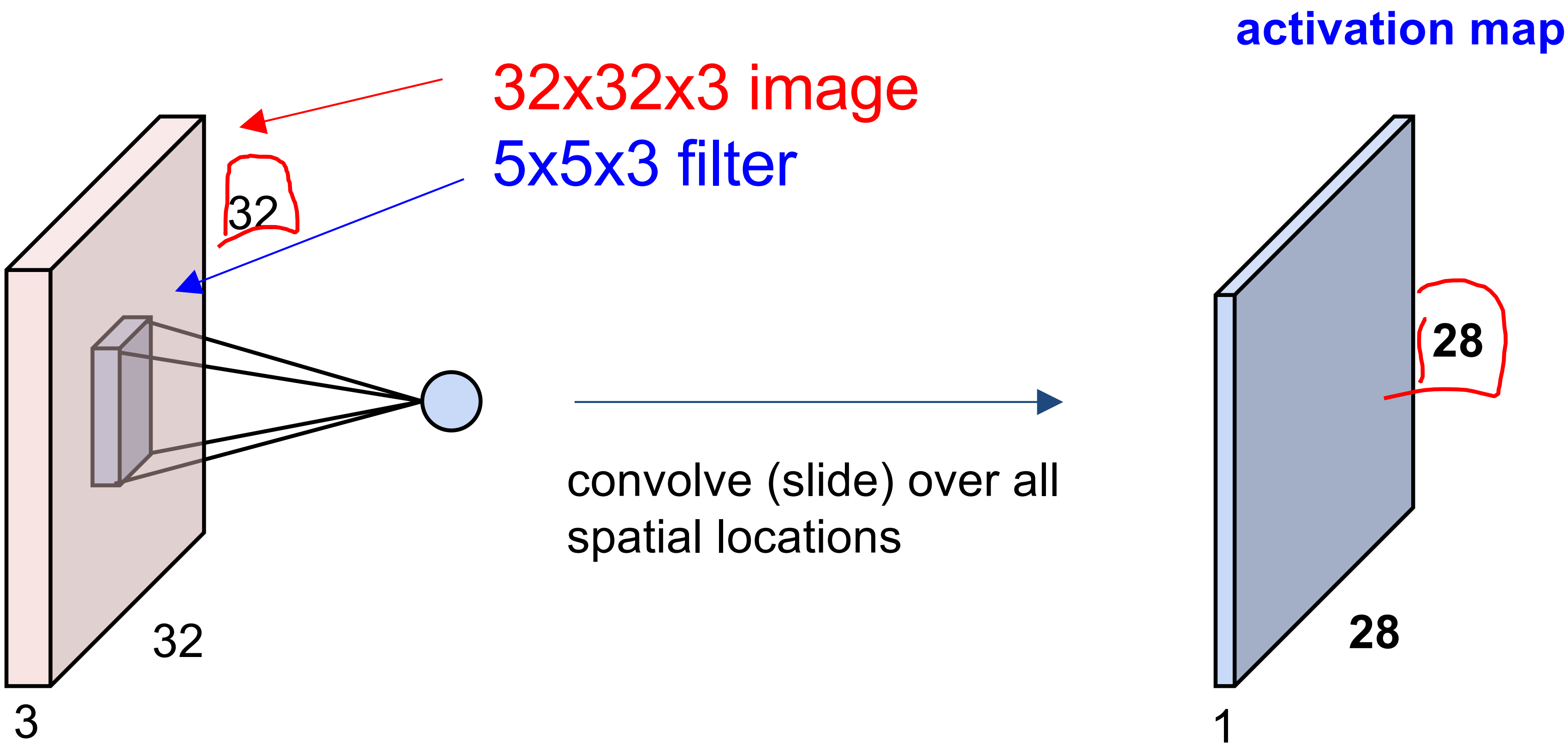
Feature visualization of convolutional net trained on ImageNet from [Zeiler & Fergus 2013]

Slide Credit: Marc'Aurelio Ranzato, Yann LeCun

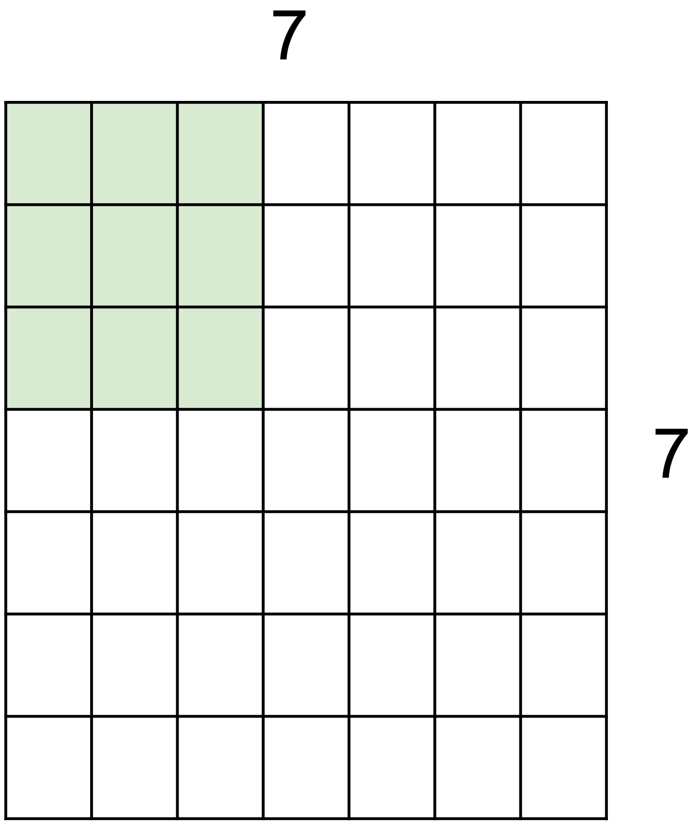
Plan for Today

- 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:

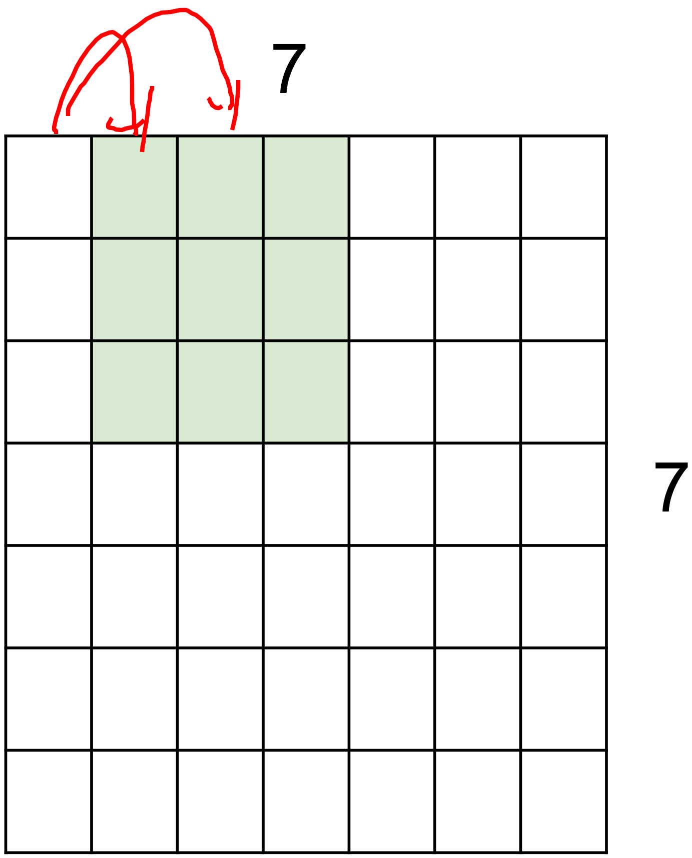


7x7 input (spatially)
assume 3x3 filter

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

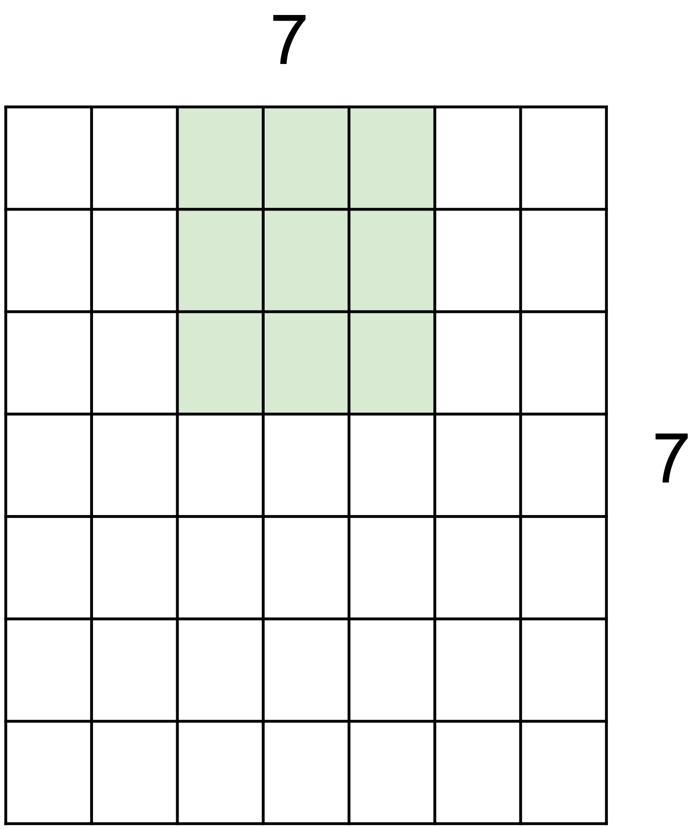
Stride

A closer look at spatial dimensions:



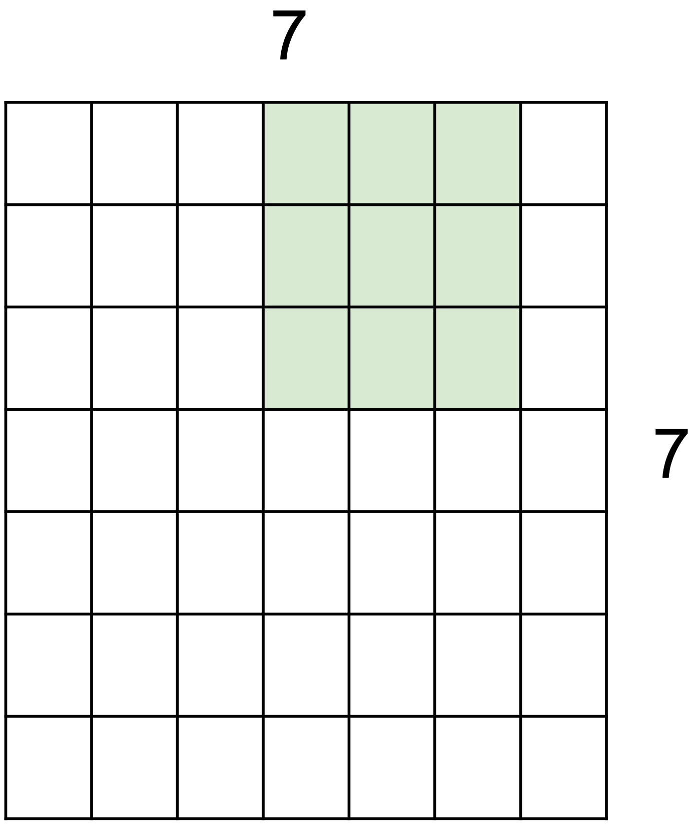
7x7 input (spatially)
assume 3x3 filter

A closer look at spatial dimensions:



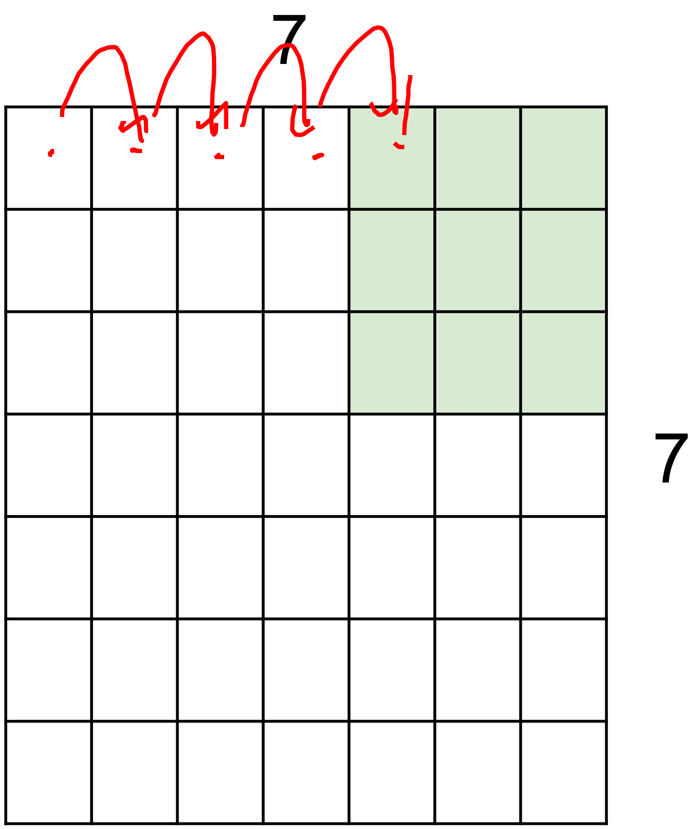
7x7 input (spatially)
assume 3x3 filter

A closer look at spatial dimensions:



7x7 input (spatially)
assume 3x3 filter

A closer look at spatial dimensions:



7x7 input (spatially)
assume 3x3 filter

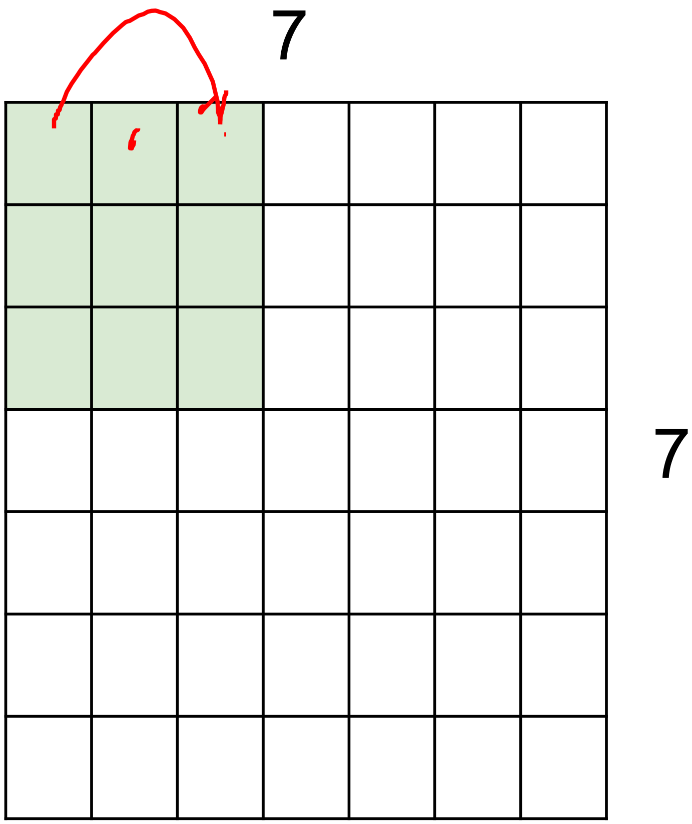
=> 5x5 output

'valid'

'same'

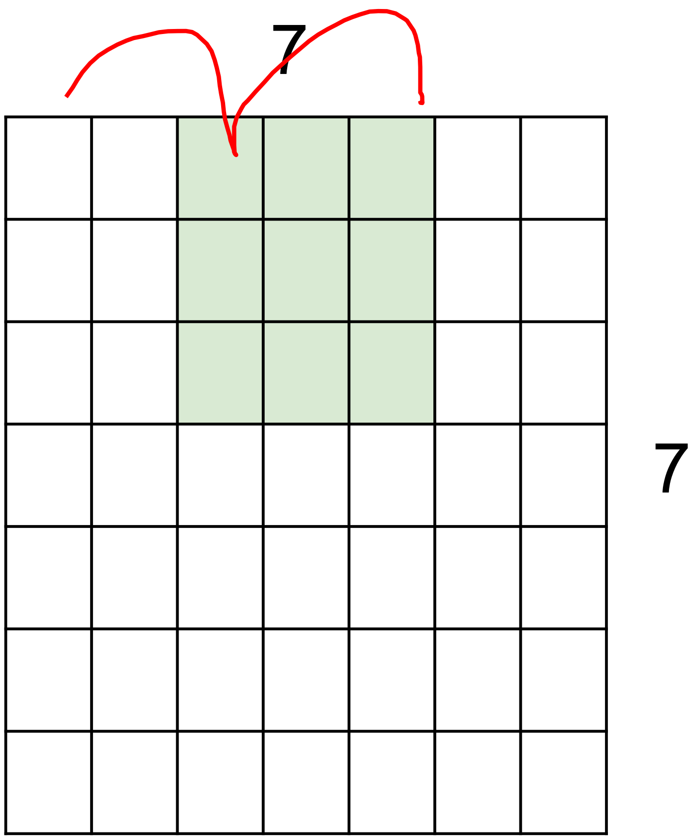
Slide Credit: Fei-Fei Li, Justin Johnson, Serena Yeung, CS 231n

A closer look at spatial dimensions:



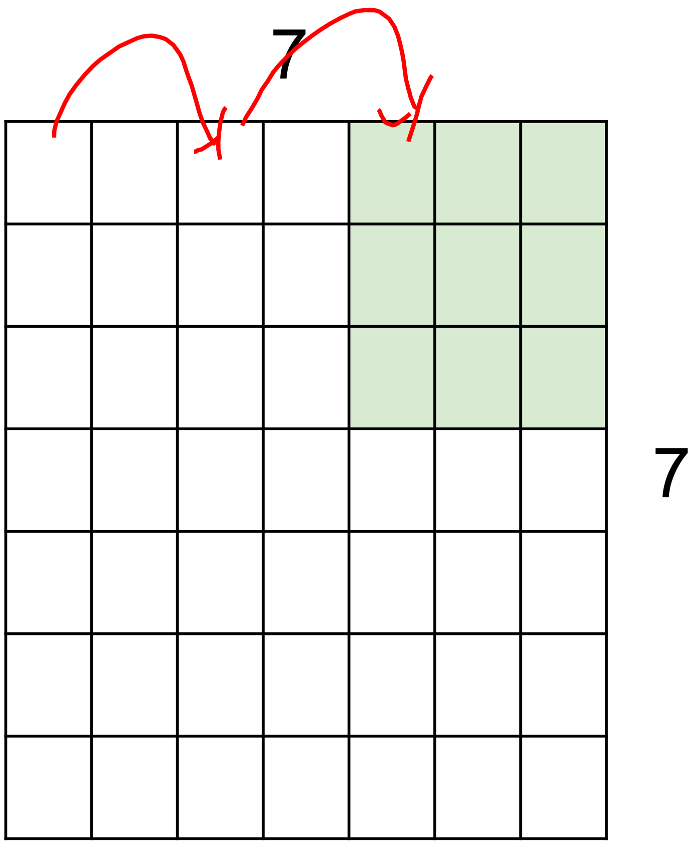
7x7 input (spatially)
assume 3x3 filter
applied **with stride 2**

A closer look at spatial dimensions:



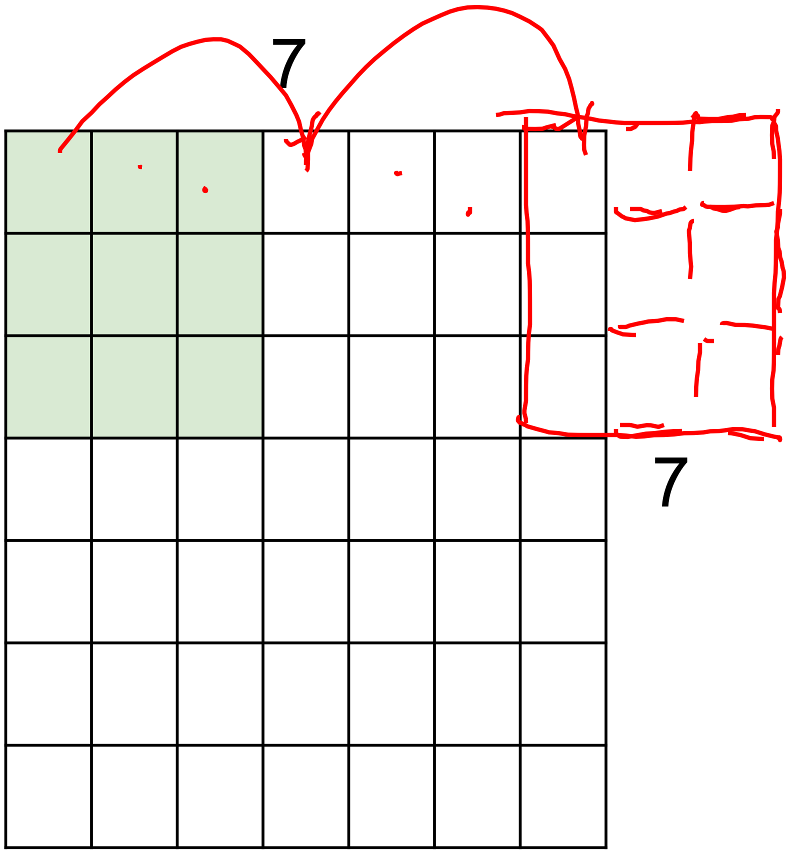
7x7 input (spatially)
assume 3x3 filter
applied **with stride 2**

A closer look at spatial dimensions:



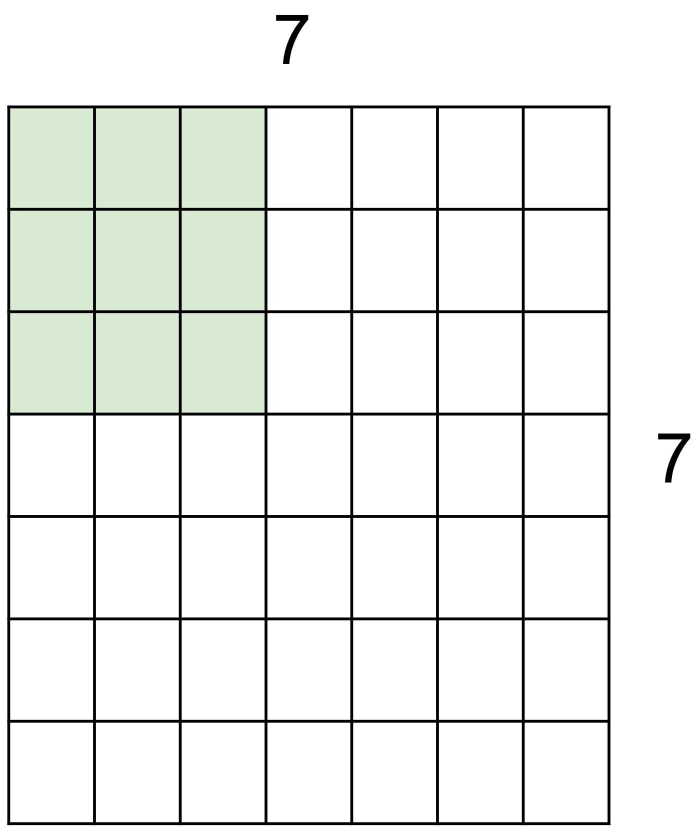
7x7 input (spatially)
assume 3x3 filter
applied **with stride 2**
=> 3x3 output!

A closer look at spatial dimensions:



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

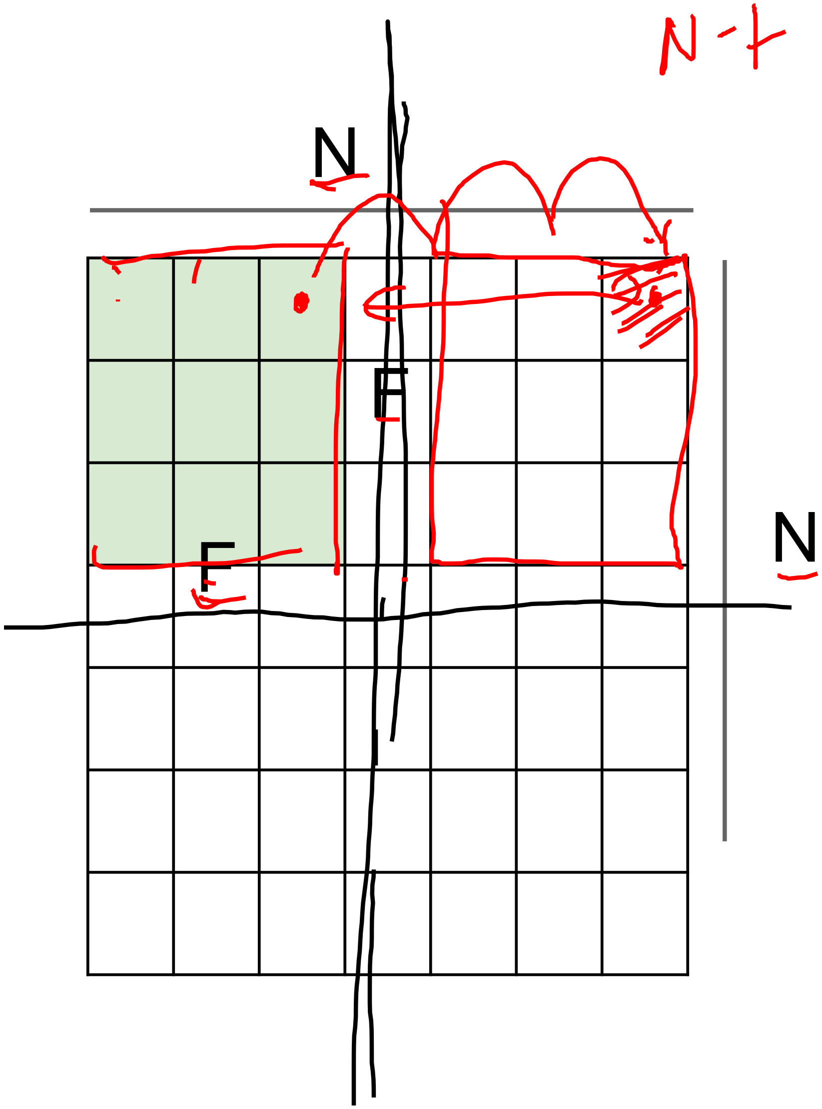
A closer look at spatial dimensions:



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

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

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



Output size: $= \frac{N-F}{\text{stride}}$

(N - F) / stride + $\boxed{1}$

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

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

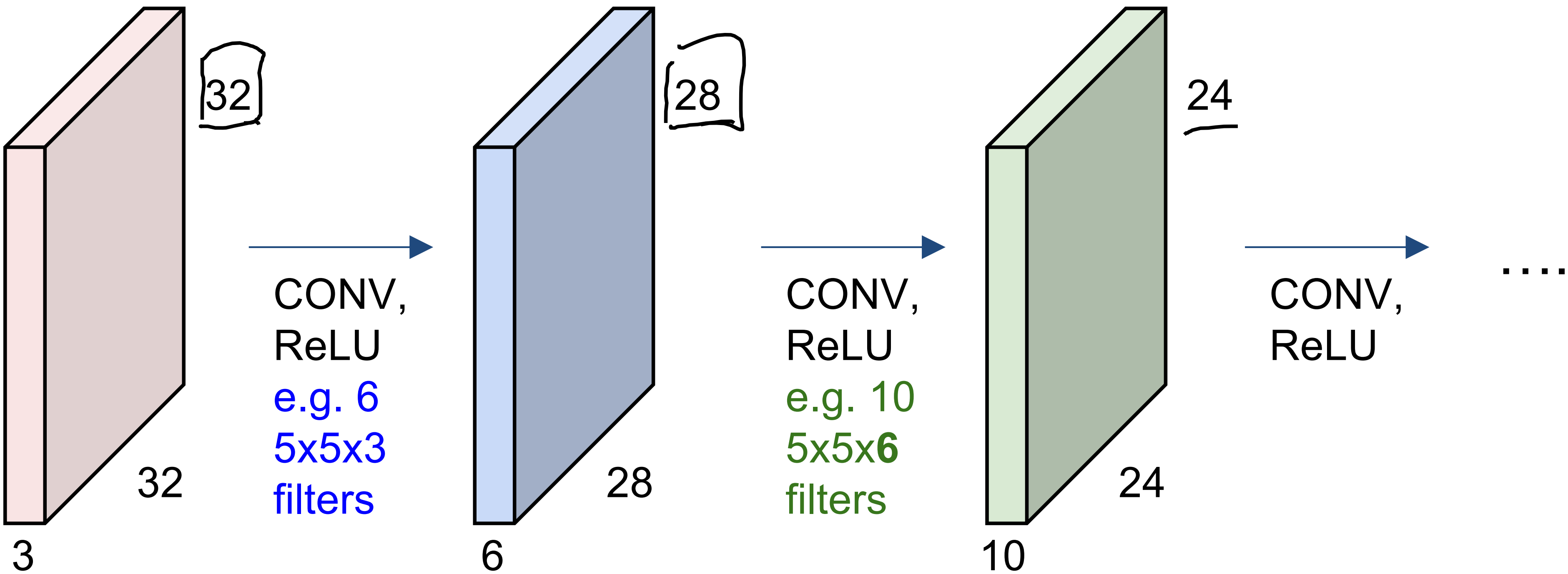
stride 2 => (7 - 3)/2 + 1 = 3

stride 3 => (7 - 3)/3 + 1 = $\boxed{2.33}$:\

$\frac{7-3}{4} + 1$

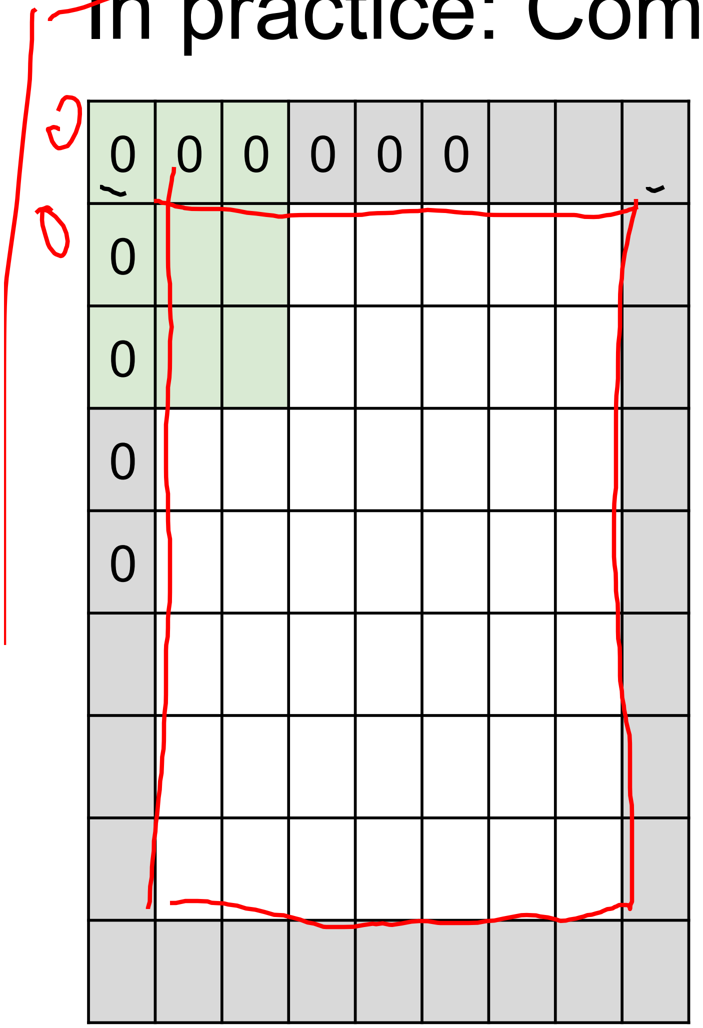
Remember back to…

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.



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

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?

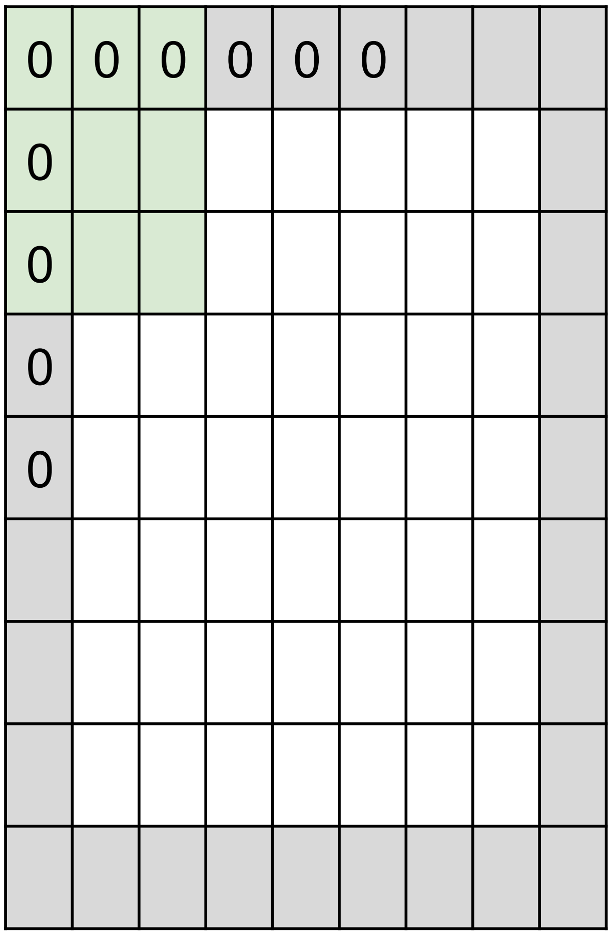
$$N = \left[\frac{N + 2pad - F}{stride} + 1\right]$$

$$pad = \frac{F-1}{2}$$

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

$$N: \frac{9-3}{1} + 1 = 6 + 1 = 7$$

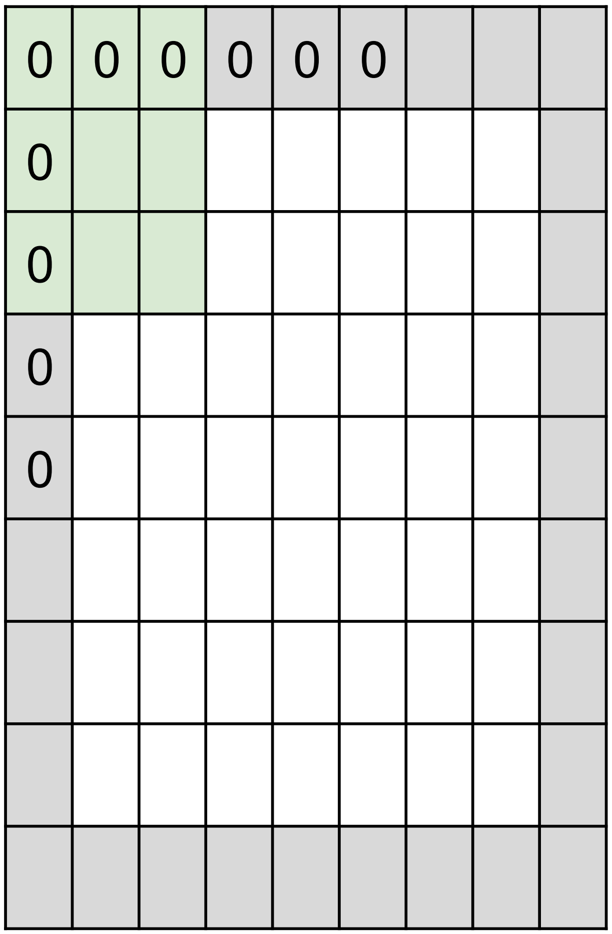
In practice: Common to zero pad the border



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

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

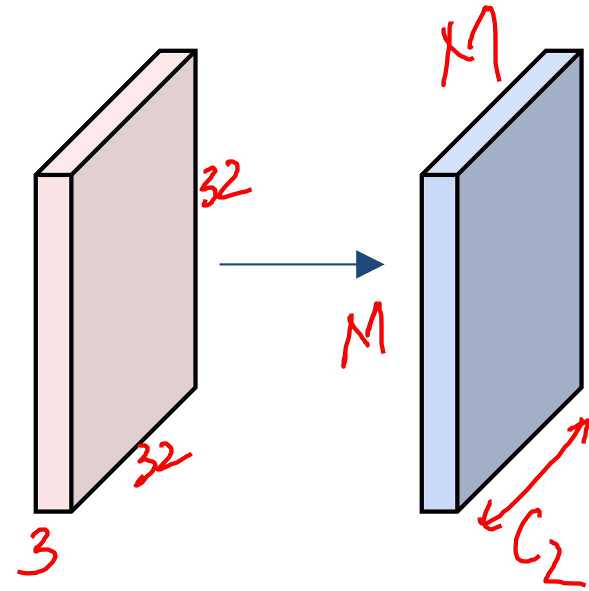
Examples time:

Input volume: **32x32x3**

10 5x5 filters with stride 1, pad 2

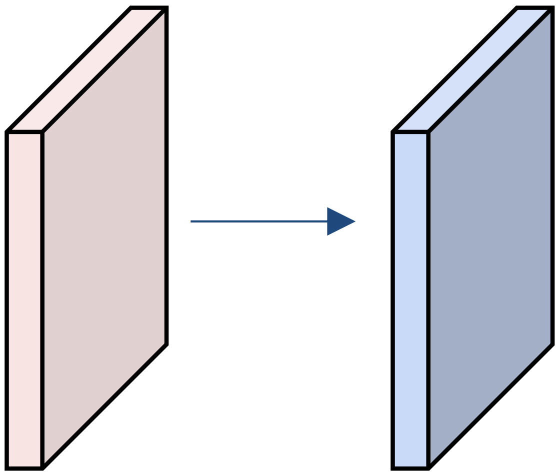
Output volume size: ?

$$\left(\frac{32+4-5}{1}+1\right) \qquad M \times M \times \underbrace{\boxed{C_2}}_{10}$$



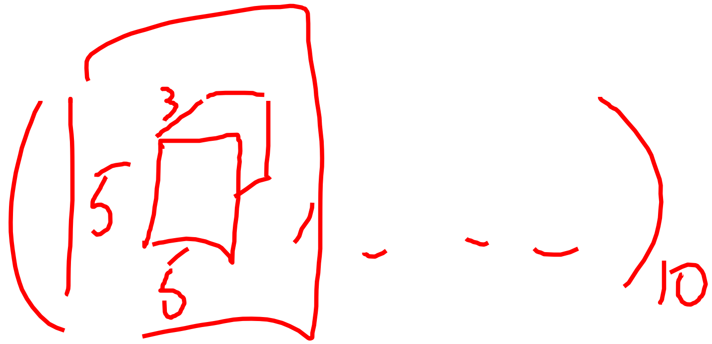
Examples time:

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



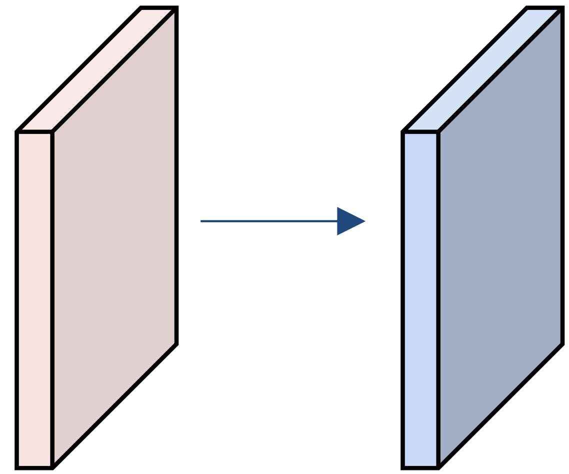
Output volume size:
(32+2*2-5)/1+1 = 32 spatially, so
32x32x10

Examples time:



Input volume: **32x32x3**

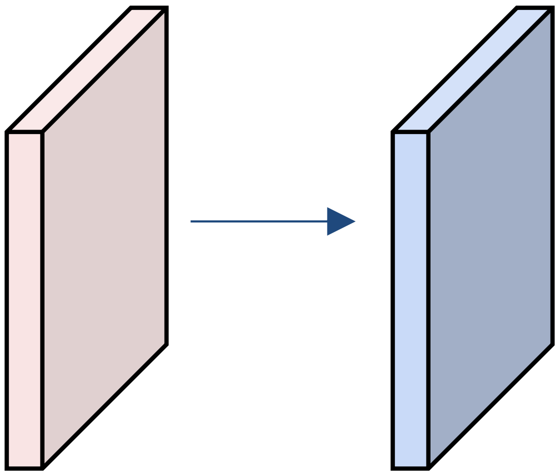
10 5x5 filters with stride 1, pad 2



Number of parameters in this layer?

$$(\underbrace{5\times5\times3}_{75} + 1)\times 10$$

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.

Example: CONV layer in Torch

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.

SpatialConvolution

```
module = nn.SpatialConvolution(nInputPlane, nOutputPlane, kW, kH, [dW], [dH], [padW], [padH])
```

Applies a 2D convolution over an input image composed of several input planes. The `input` tensor in `forward(input)` is expected to be a 3D tensor (`nInputPlane x height x width`).

The parameters are the following:

- `nInputPlane` : The number of expected input planes in the image given into `forward()` .
- `nOutputPlane` : The number of output planes the convolution layer will produce.
- `kW` : The kernel width of the convolution
- `kH` : The kernel height of the convolution
- `dW` : The step of the convolution in the width dimension. Default is `1` .
- `dH` : The step of the convolution in the height dimension. Default is `1` .
- `padW` : The additional zeros added per width to the input planes. Default is `0` , a good number is `(kW-1)/2` .
- `padH` : The additional zeros added per height to the input planes. Default is `padW` , a good number is `(kH-1)/2` .

Note that depending of the size of your kernel, several (of the last) columns or rows of the input image might be lost. It is up to the user to add proper padding in images.

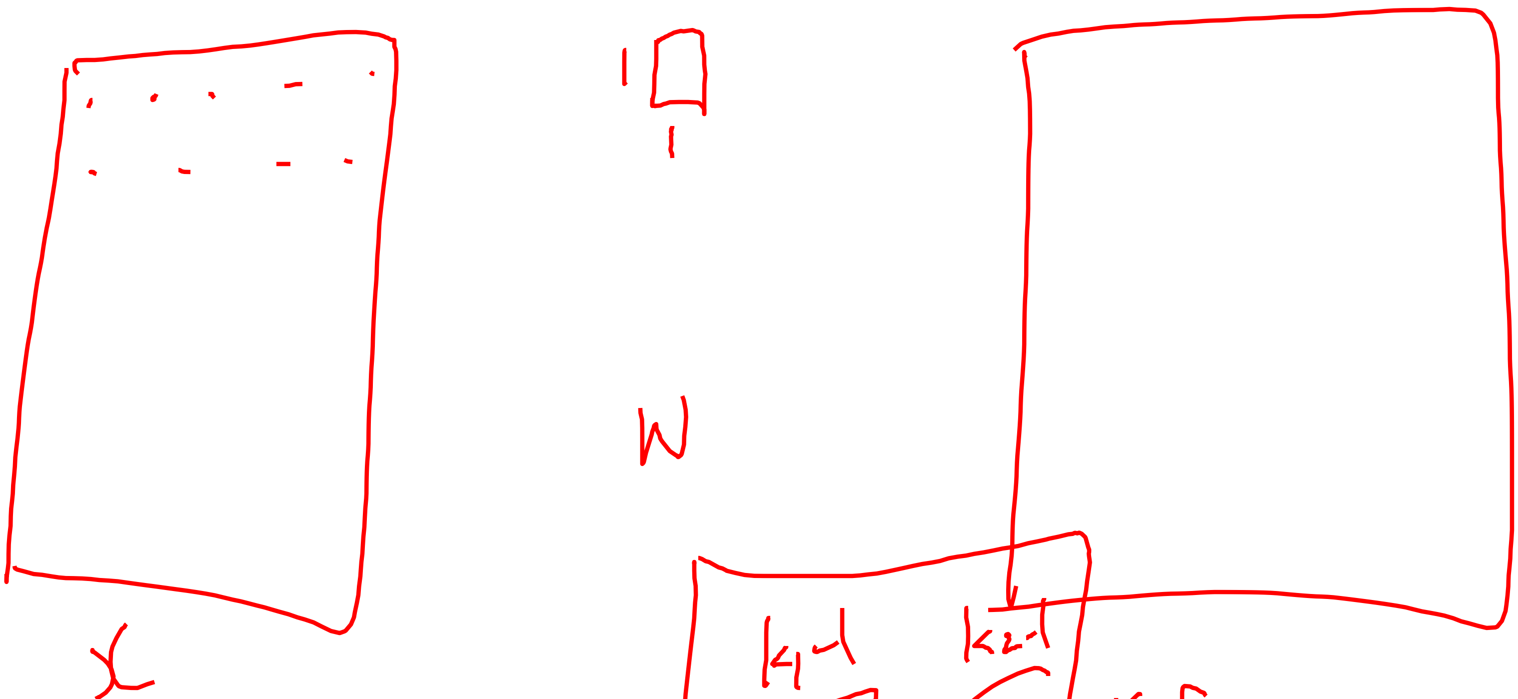
If the input image is a 3D tensor `nInputPlane x height x width` , the output image size will be `nOutputPlane x oheight x owidth` where

```
owidth  = floor((width  + 2*padW - kW) / dW + 1)
oheight = floor((height + 2*padH - kH) / dH + 1)
```

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

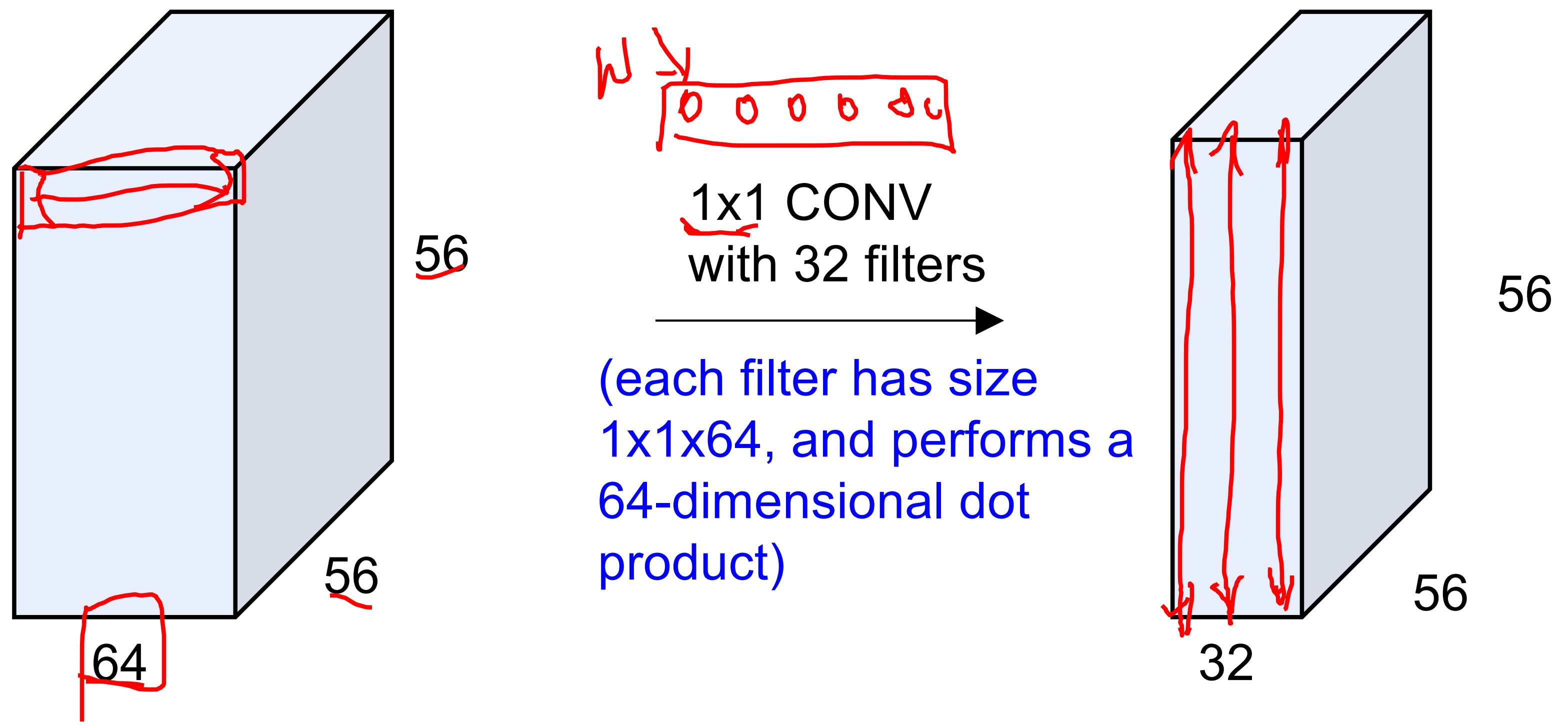
Can we have 1x1 filters?



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

$$= \boxed{x[r,c]}\, w[0,0]$$

1x1 convolution layers make perfect sense



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

Fully Connected Layer as 1x1 Conv

32x32x3 image -> stretch to 3072 x 1