DDA3020 Machine Learning: L10 Convolutional Neural Networks

Baoyuan Wu
School of Data Science, CUHK-SZ

March 27, 2024

Outline

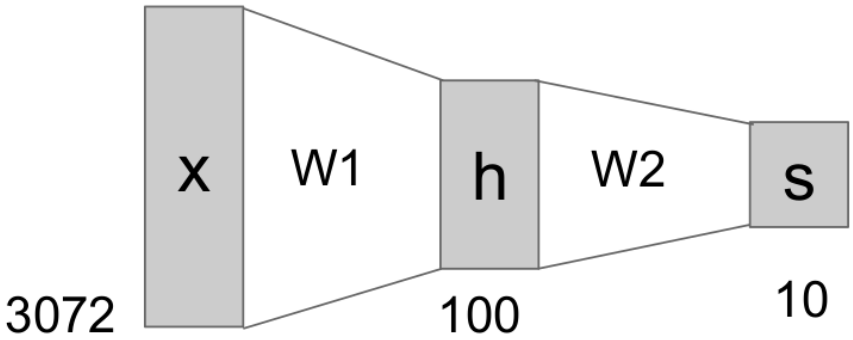
1. History of CNN

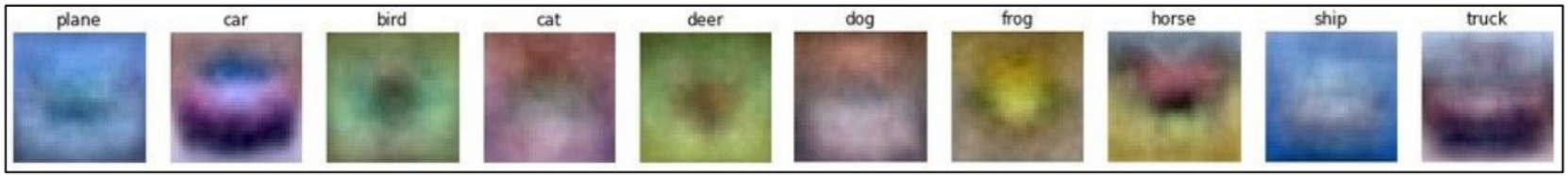
2. Convolutional Neural Networks
   - Preview
   - Convolutional layer
   - Pooling layer
   - Fully connected layer

Last week: Neural Networks

linear score function: $f = Wx$

2-layer Neural Network $f = W_2 \max(0, W_1 x)$





Today: Convolutional Neural Networks

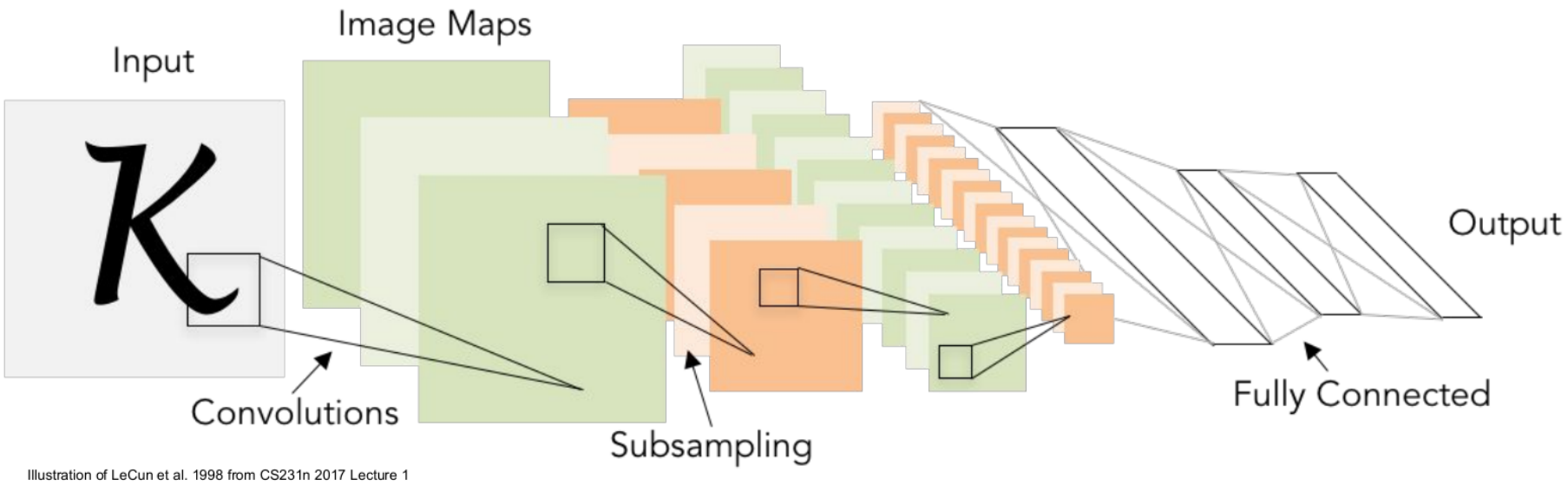
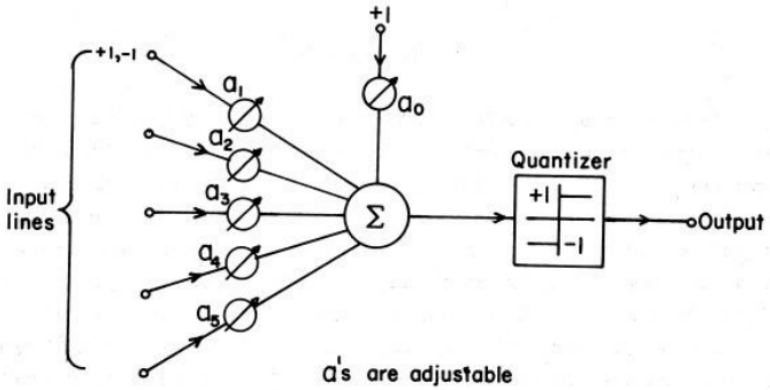
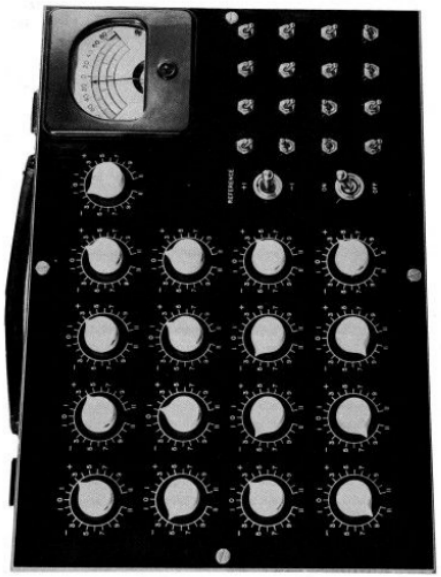
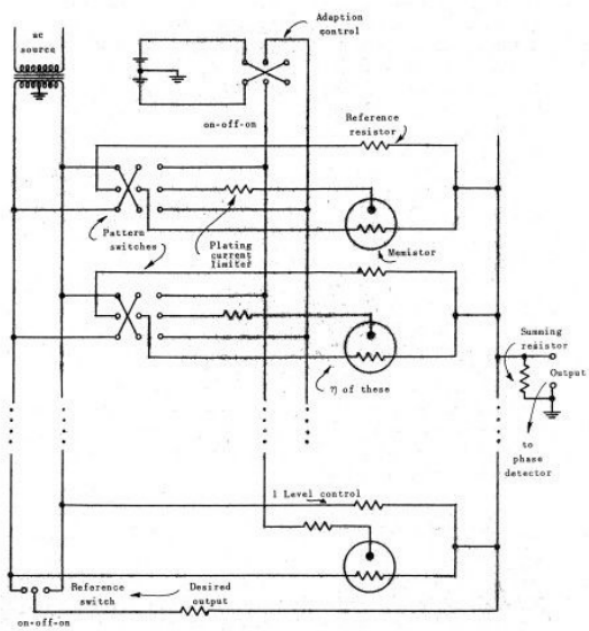


Illustration of LeCun et al. 1998 from CS231n 2017 Lecture 1

A bit of history...







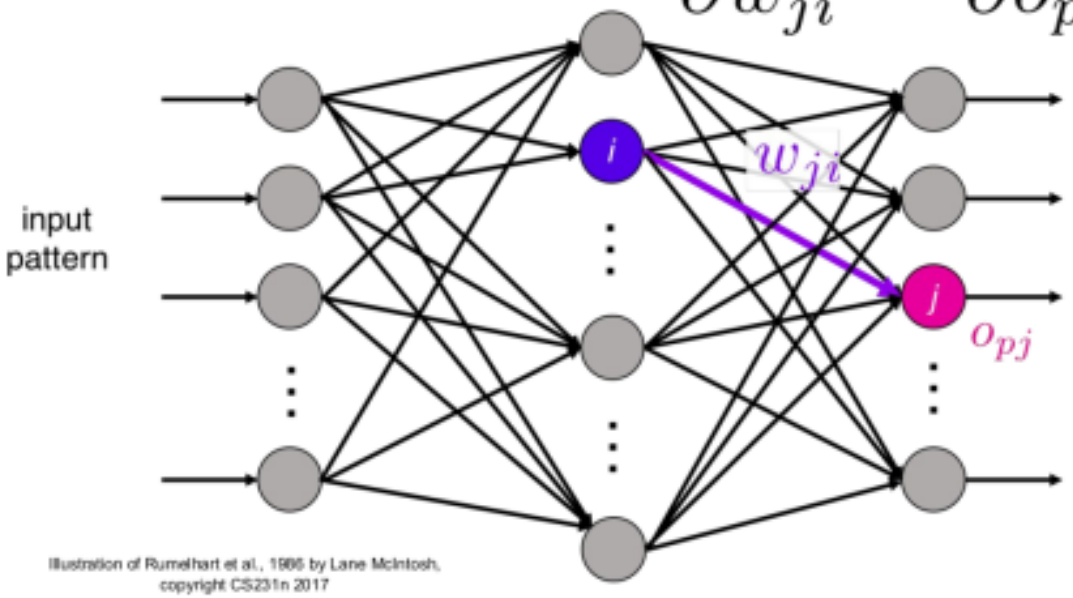
Widrow and Hoff, ~1960: Adaline/Madaline

These figures are reproduced from Widrow 1960, Stanford Electronics Laboratories Technical Report with permission from Stanford University Special Collections.

A bit of history...

$$\frac{\partial E_p}{\partial w_{ji}} = \frac{\partial E_p}{\partial o_{pj}} \frac{\partial o_{pj}}{\partial w_{ji}}$$

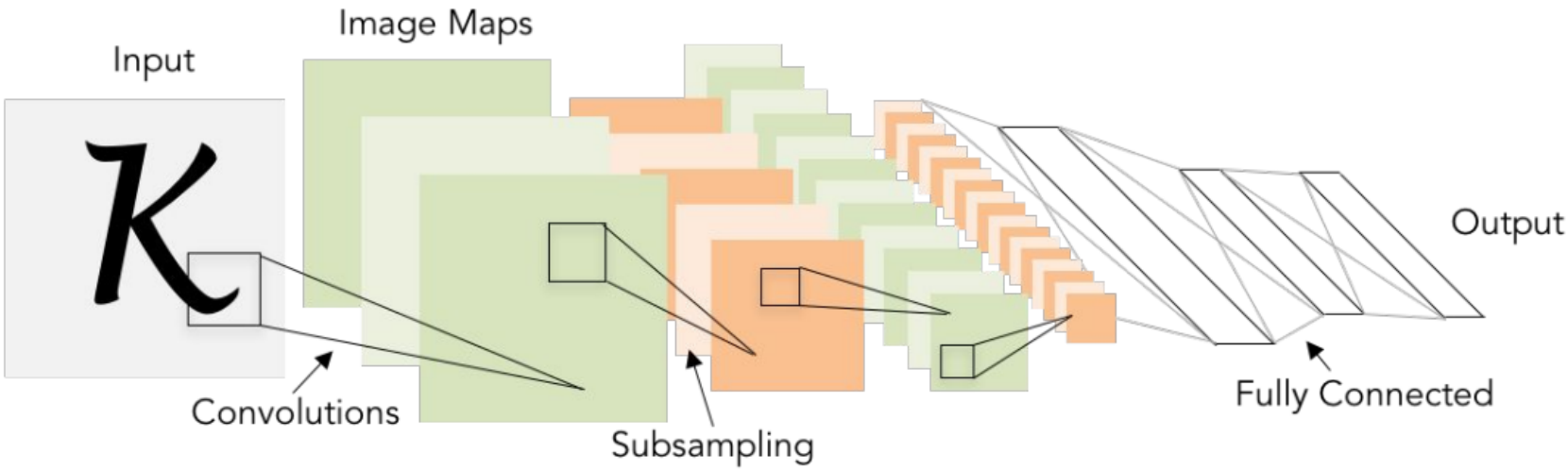
recognizable math



Rumelhart et al., 1986: First time back-propagation became popular

A bit of history:

Gradient-based learning applied to document recognition
[LeCun, Bottou, Bengio, Haffner 1998]



LeNet-5

A bit of history...

[Hinton and Salakhutdinov 2006]
Reinvigorated research in Deep Learning

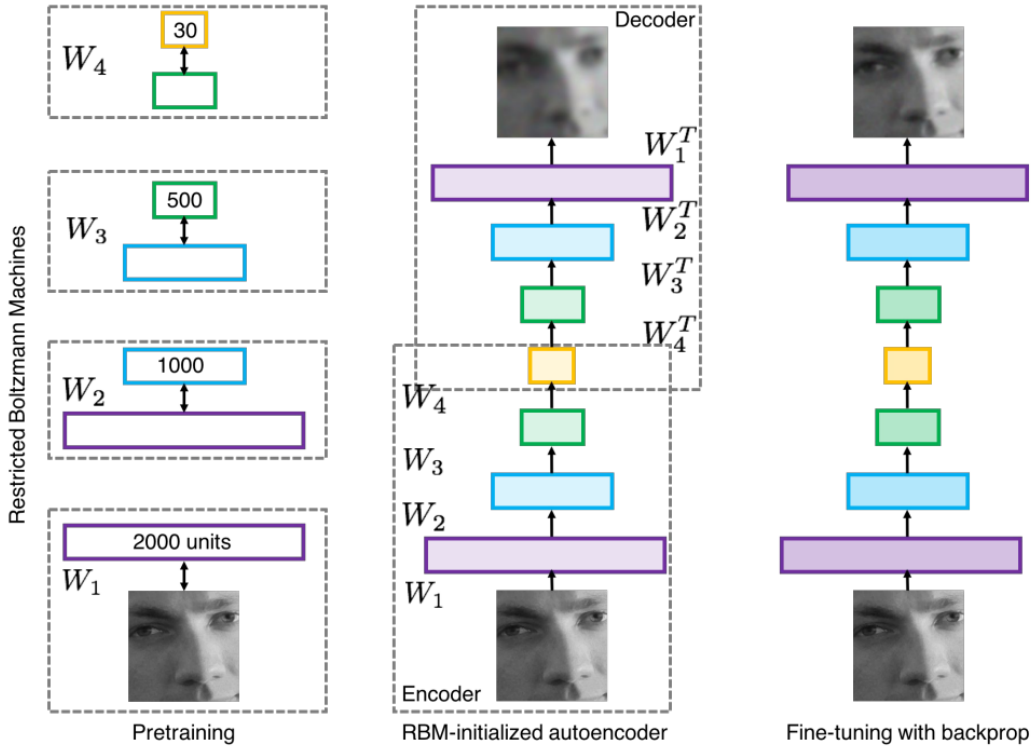


Illustration of Hinton and Salakhutdinov 2006 by Lane McIntosh, copyright CS231n 2017

A bit of history:

ImageNet Classification with Deep Convolutional Neural Networks [Krizhevsky, Sutskever, Hinton, 2012]

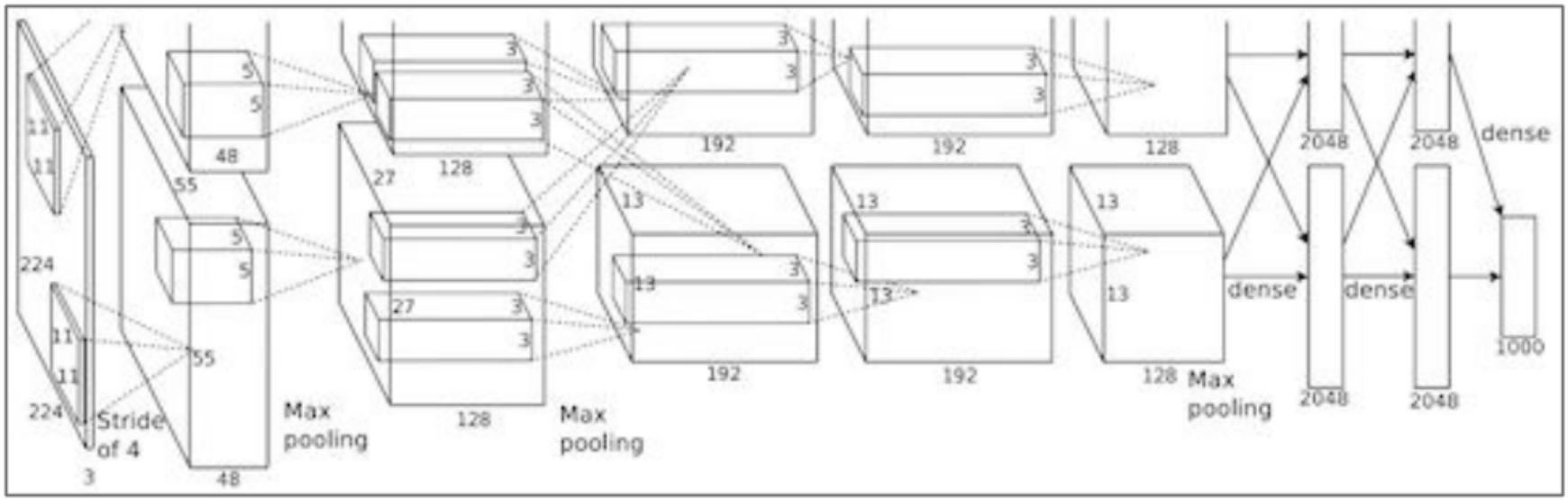


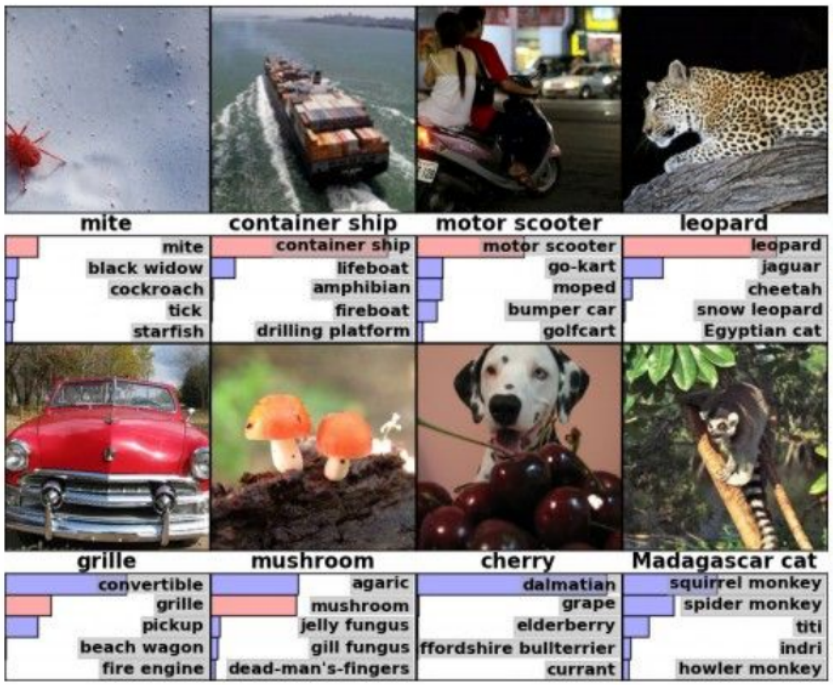
Figure copyright Alex Krizhevsky, Ilya Sutskever, and Geoffrey Hinton, 2012. Reproduced with permission.

"AlexNet"

Fast-forward to today: ConvNets are everywhere

Classification

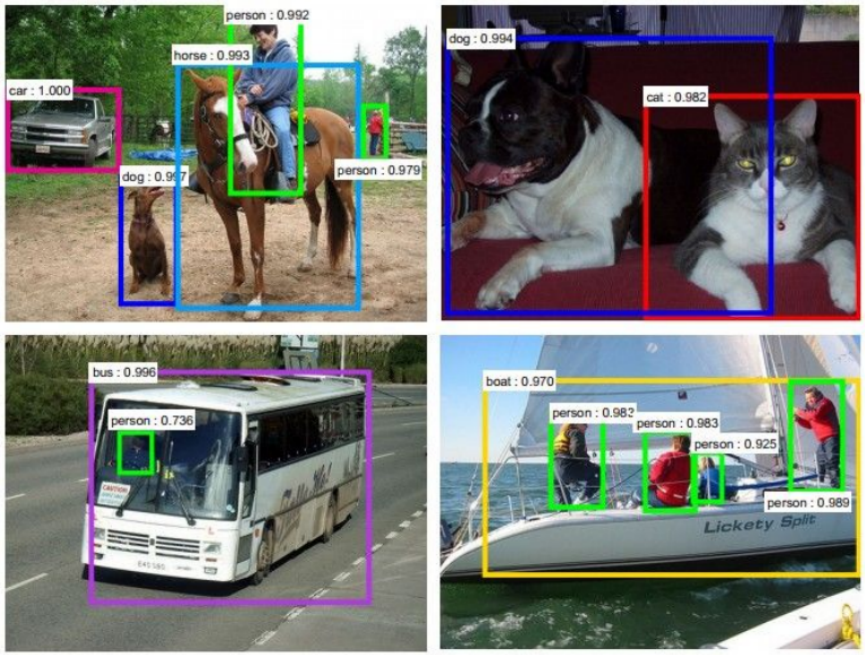
Retrieval



Figures copyright Alex Krizhevsky, Ilya Sutskever, and Geoffrey Hinton, 2012. Reproduced with permission.

Fast-forward to today: ConvNets are everywhere

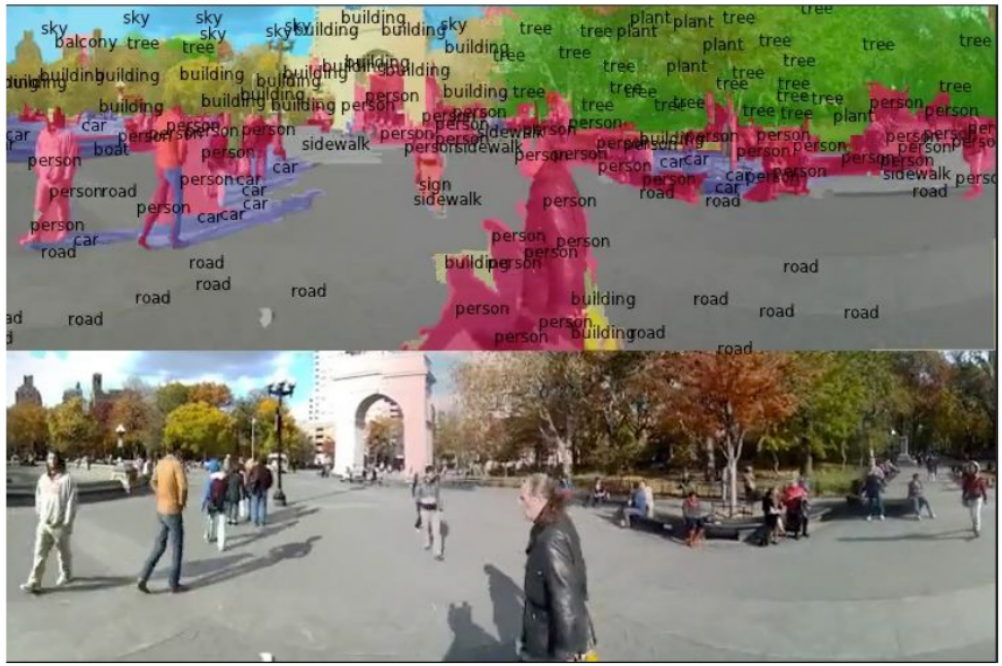
Detection



Figures copyright Shaoqing Ren, Kaiming He, Ross Girschick, Jian Sun, 2015. Reproduced with permission.

[Faster R-CNN: Ren, He, Girshick, Sun 2015]

Segmentation



Figures copyright Clement Farabet, 2012. Reproduced with permission.

[Farabet et al., 2012]

Fast-forward to today: ConvNets are everywhere

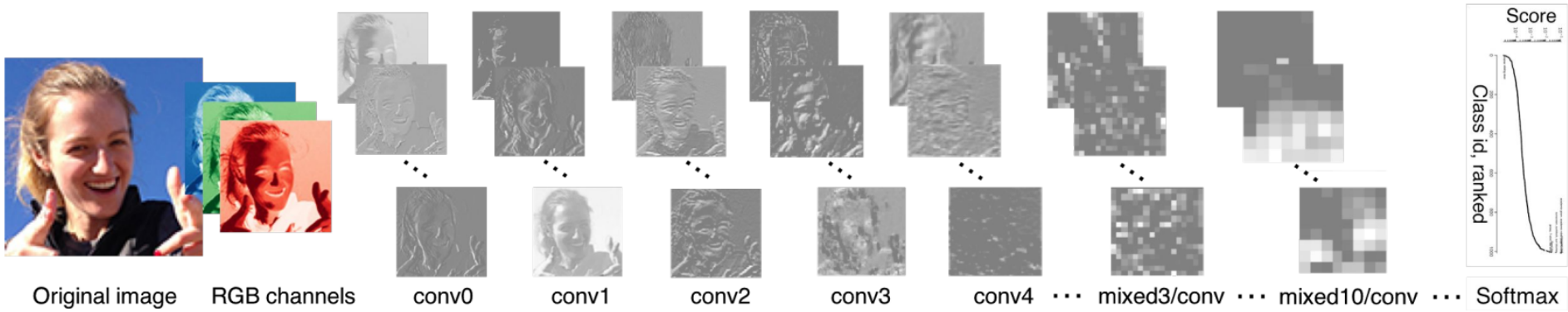
Photo by Lane McIntosh. Copyright CS231n 2017.

self-driving cars

NVIDIA Tesla line
(these are the GPUs on rye01.stanford.edu)

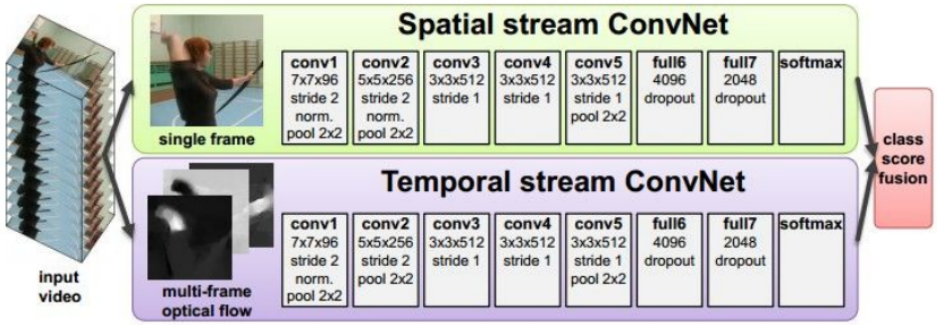
Note that for embedded systems a typical setup would involve NVIDIA Tegras, with integrated GPU and ARM-based CPU cores.

Fast-forward to today: ConvNets are everywhere



[Taigman et al. 2014]

Activations of inception-v3 architecture [Szegedy et al. 2015] to image of Emma McIntosh, used with permission. Figure and architecture not from Taigman et al. 2014.



[Simonyan et al. 2014]

Figures copyright Simonyan et al., 2014. Reproduced with permission.

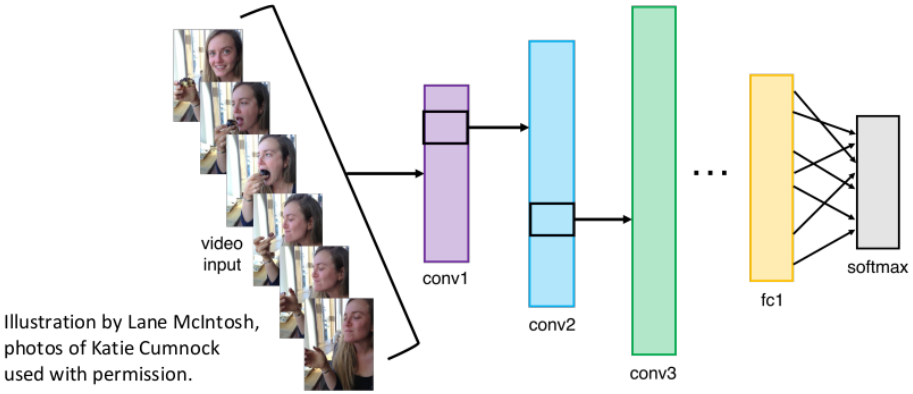
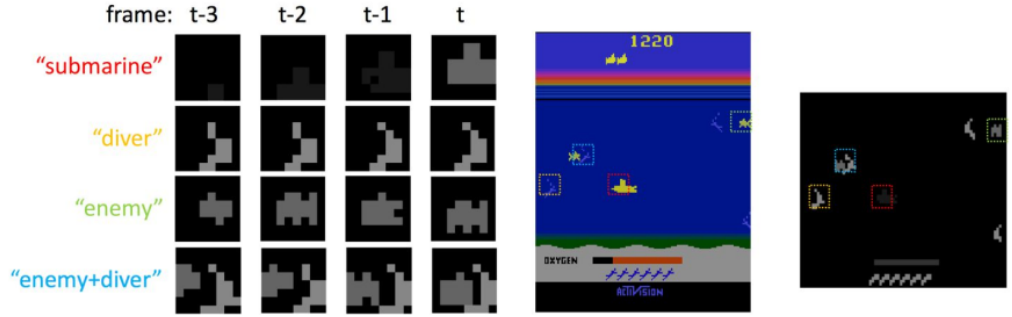


Illustration by Lane McIntosh, photos of Katie Cumnock used with permission.

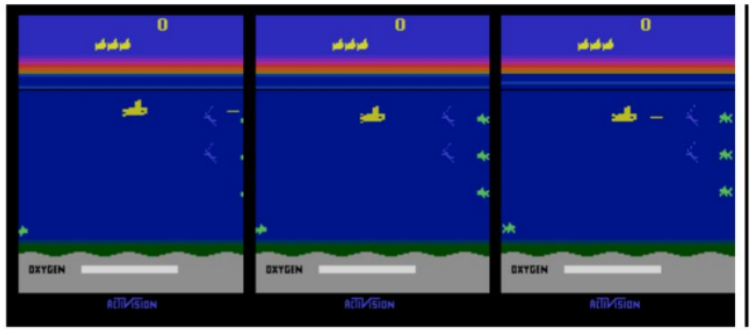
Fast-forward to today: ConvNets are everywhere

Images are examples of pose estimation, not actually from Toshev & Szegedy 2014. Copyright Lane McIntosh.

[Toshev, Szegedy 2014]

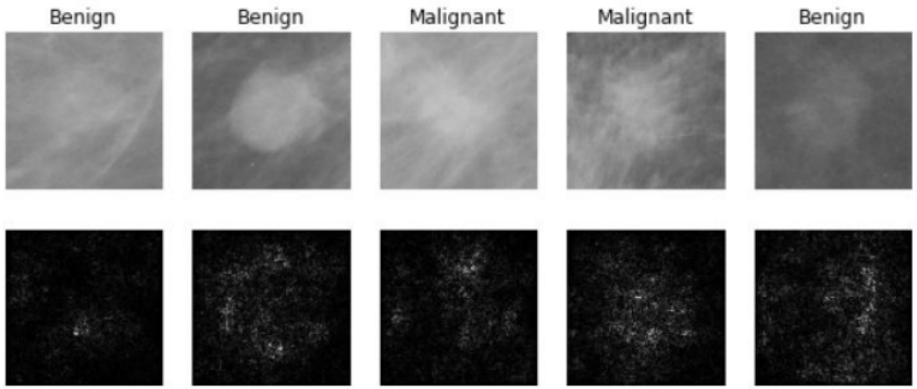


[Guo et al. 2014]



Figures copyright Xiaoxiao Guo, Satinder Singh, Honglak Lee, Richard Lewis, and Xiaoshi Wang, 2014. Reproduced with permission.

Fast-forward to today: ConvNets are everywhere



[Levy et al. 2016]

Figure copyright Levy et al. 2016.
Reproduced with permission.

[Dieleman et al. 2014]

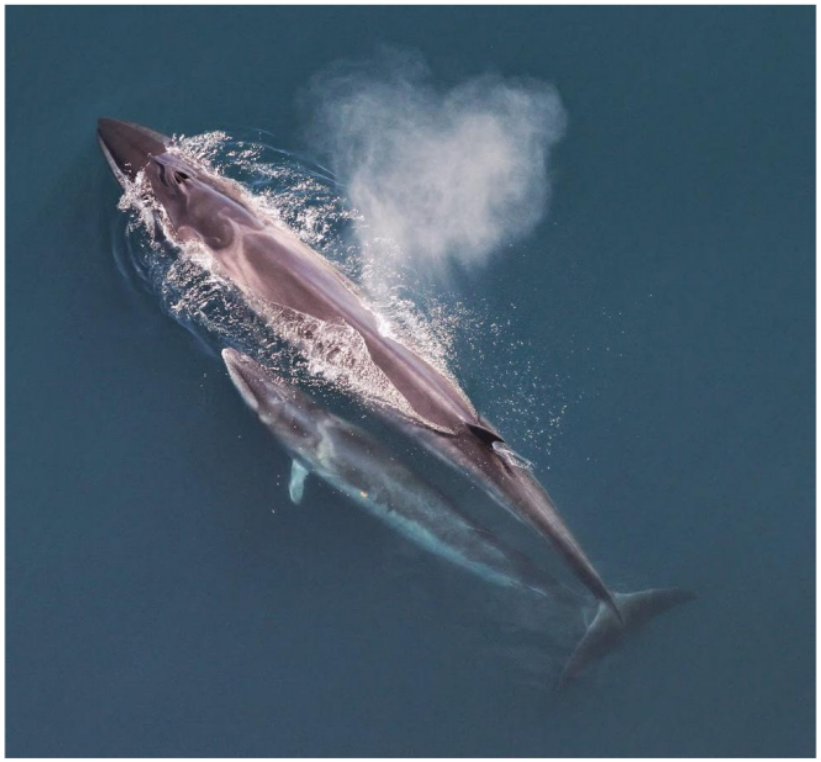
From left to right: public domain by NASA, usage permitted by ESA/Hubble, public domain by NASA, and public domain.

[Sermanet et al. 2011]
[Ciresan et al.]

Photos by Lane McIntosh.
Copyright CS231n 2017.

Fast-forward to today: ConvNets are everywhere

This image by Christin Khan is in the public domain and originally came from the U.S. NOAA.



Whale recognition, Kaggle Challenge

Photo and figure by Lane McIntosh; not actual example from Mnih and Hinton, 2010 paper.

Mnih and Hinton, 2010

Fast-forward to today: ConvNets are everywhere

No errors

A white teddy bear sitting in the grass

A man riding a wave on top of a surfboard

Minor errors

A man in a baseball uniform throwing a ball

A cat sitting on a suitcase on the floor

Somewhat related

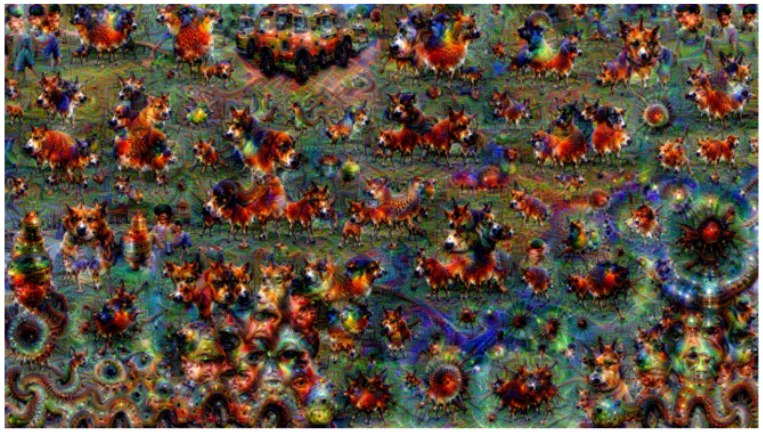
A woman is holding a cat in her hand

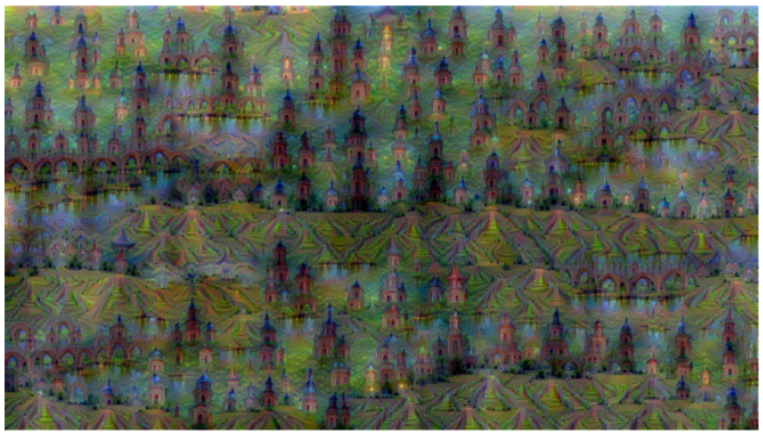
A woman standing on a beach holding a surfboard

Image Captioning

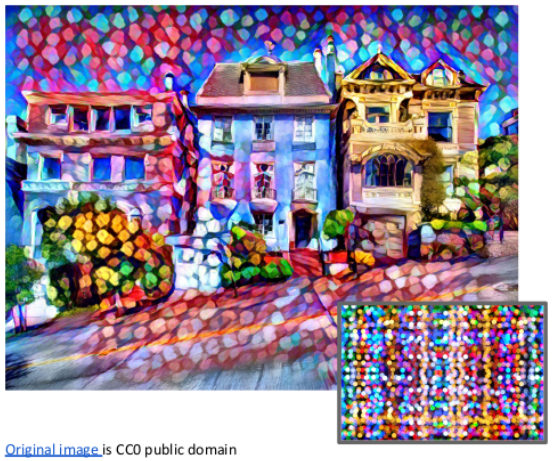
[Vinyals et al., 2015]
[Karpathy and Fei-Fei, 2015]

Fast-forward to today: ConvNets are everywhere





Figures copyright Justin Johnson, 2015. Reproduced with permission. Generated using the Inceptionism approach from a blog post by Google Research.



Original image is CC0 public domain
Starry Night and Tree Roots by Van Gogh are in the public domain
Bokeh image is in the public domain
Stylized images copyright Justin Johnson, 2017;
reproduced with permission

Gatys et al, "Image Style Transfer using Convolutional Neural Networks", CVPR 2016
Gatys et al, "Controlling Perceptual Factors in Neural Style Transfer", CVPR 2017

1 History of CNN

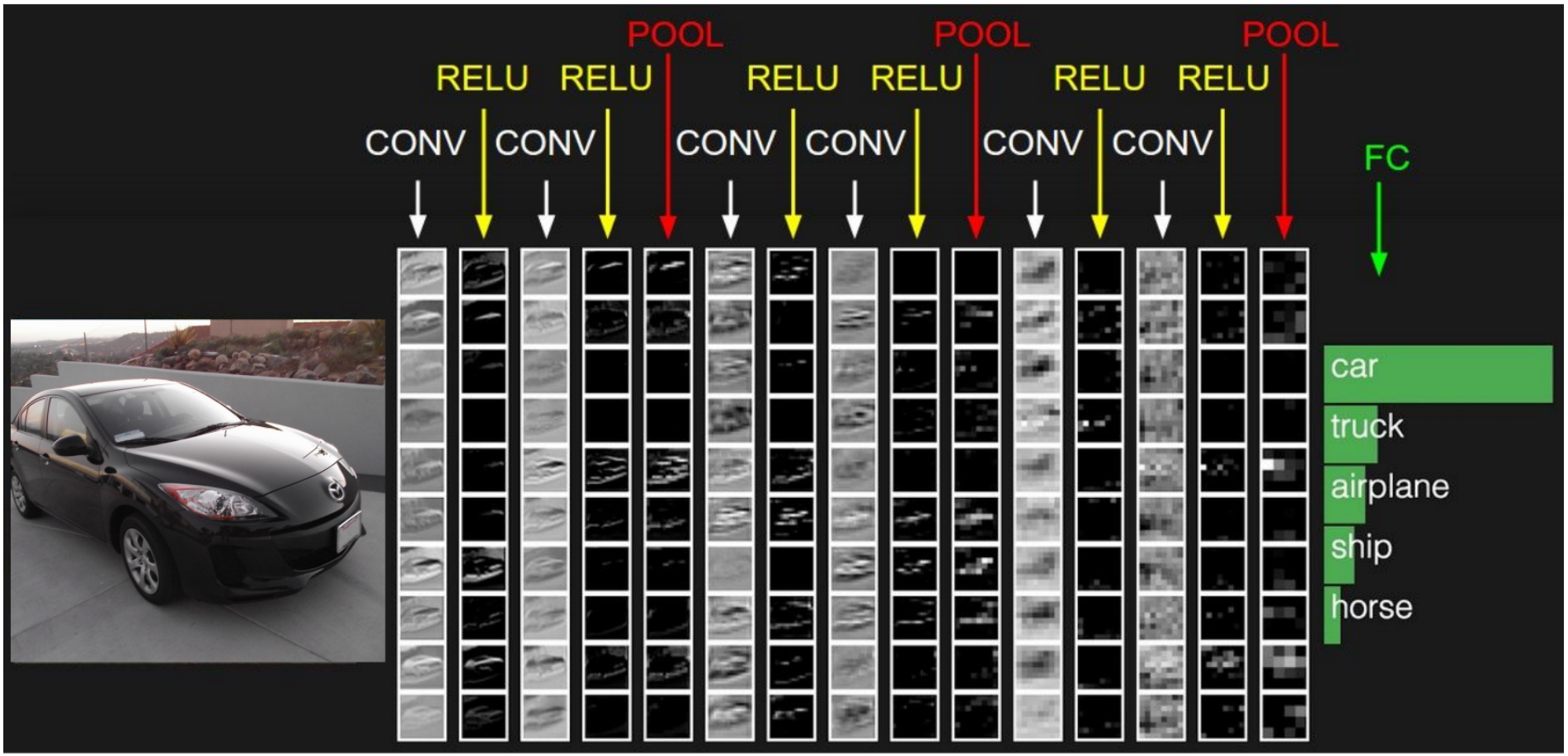
2 Convolutional Neural Networks
- Preview
- Convolutional layer
- Pooling layer
- Fully connected layer

1. History of CNN

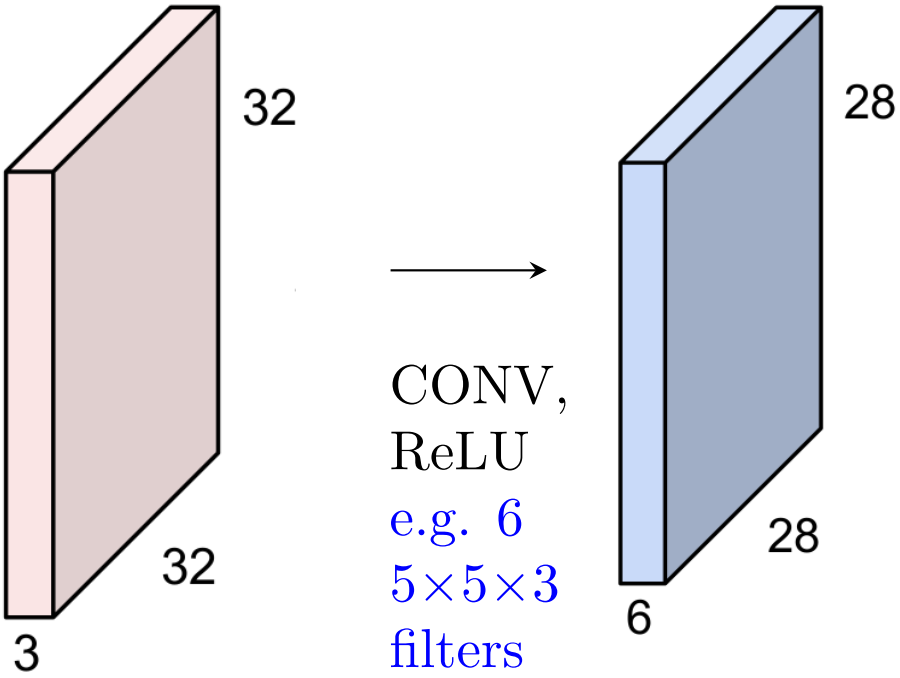
2. Convolutional Neural Networks
   - Preview
   - Convolutional layer
   - Pooling layer
   - Fully connected layer

Preview

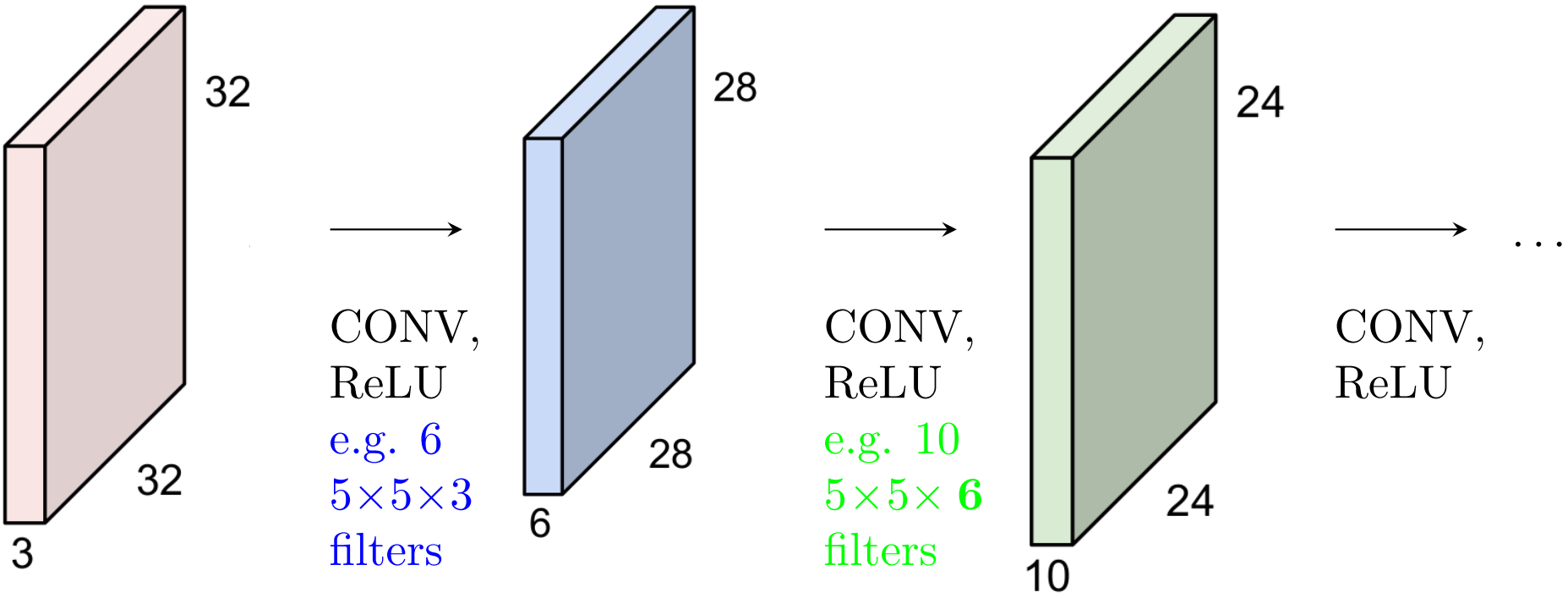
ConvNet is a sequence of Convolution Layers, interspersed with activation functions



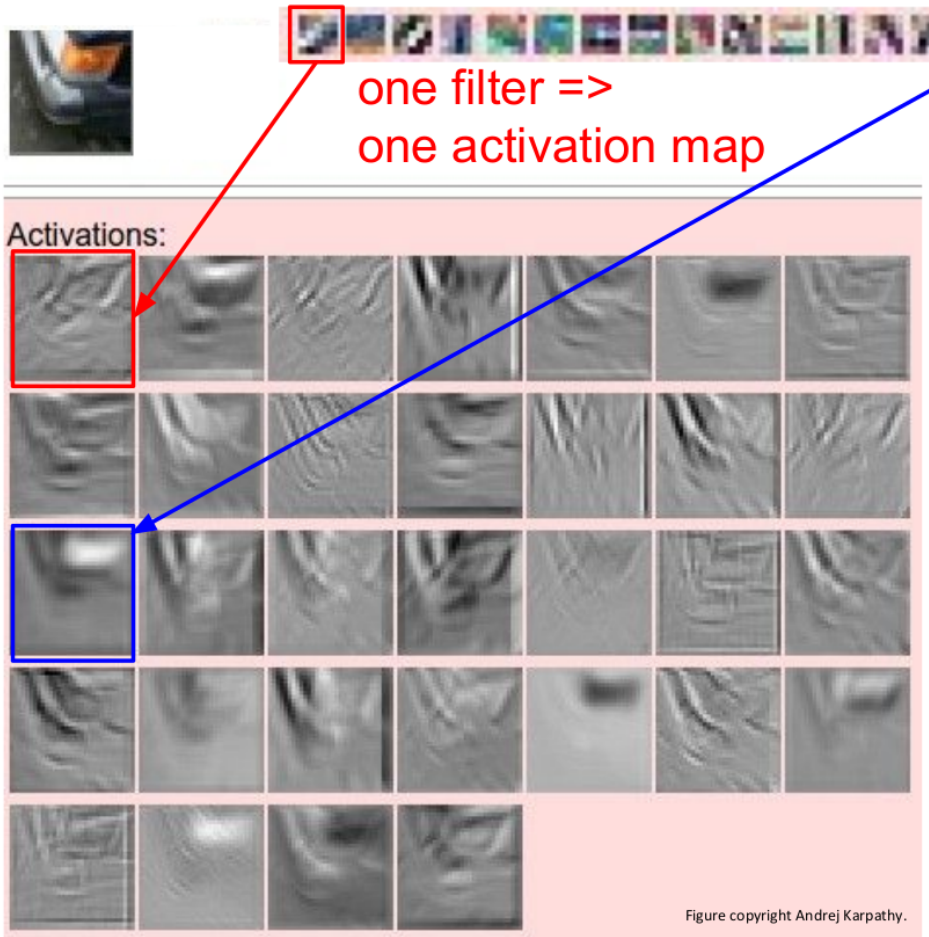
Preview



Preview


32
32
3
CONV,
ReLU
e.g. 6
5×5×3
filters
28
28
6
CONV,
ReLU
e.g. 10
5×5× 6
filters
24
24
10
CONV,
ReLU
. . .

Preview



example 5x5 filters
(32 total)

We call the layer convolutional because it is related to convolution of two signals:

$$f[x,y] * g[x,y] = \sum_{n_1=-\infty}^{\infty} \sum_{n_2=-\infty}^{\infty} f[n_1,n_2] \cdot g[x-n_1, y-n_2]$$

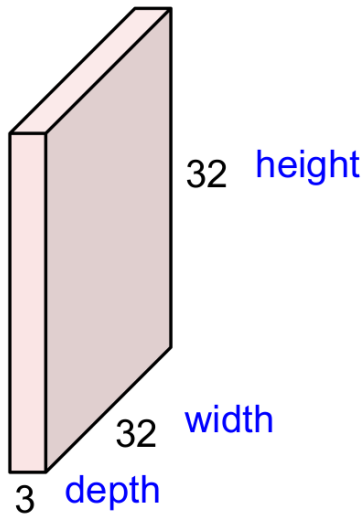
elementwise multiplication and sum of a filter and the signal (image)

1. History of CNN

2. Convolutional Neural Networks
   - Preview
   - Convolutional layer
   - Pooling layer
   - Fully connected layer

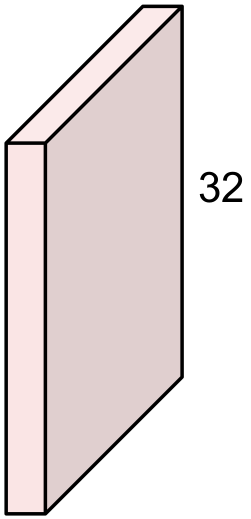
Convolution Layer

$32 \times 32 \times 3$ image $\rightarrow$ preserve spatial structure

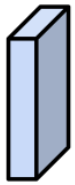


Convolution Layer

$32 \times 32 \times 3$ image



$5 \times 5 \times 3$ filter

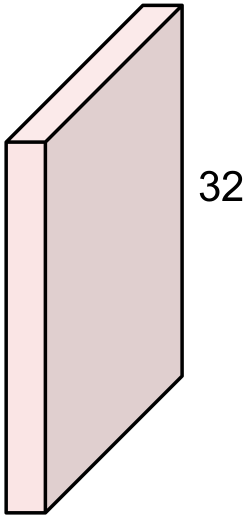


Convolve the filter with the image i.e. "slide over the image spatially, computing dot products"

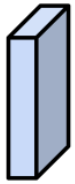
Convolution Layer

Filters always extend the full depth of the input volume

$32 \times 32 \times 3$ image

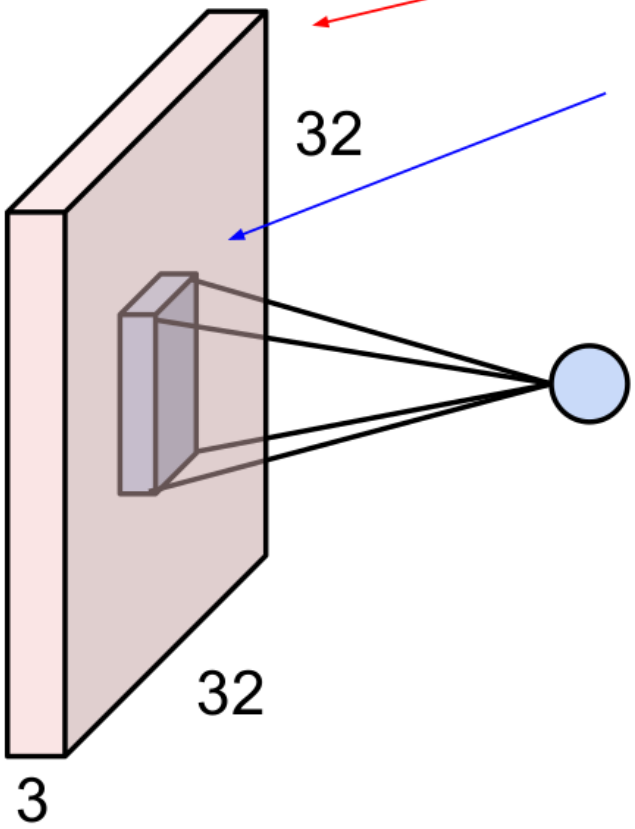


$5 \times 5 \times 3$ filter



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

Convolution Layer



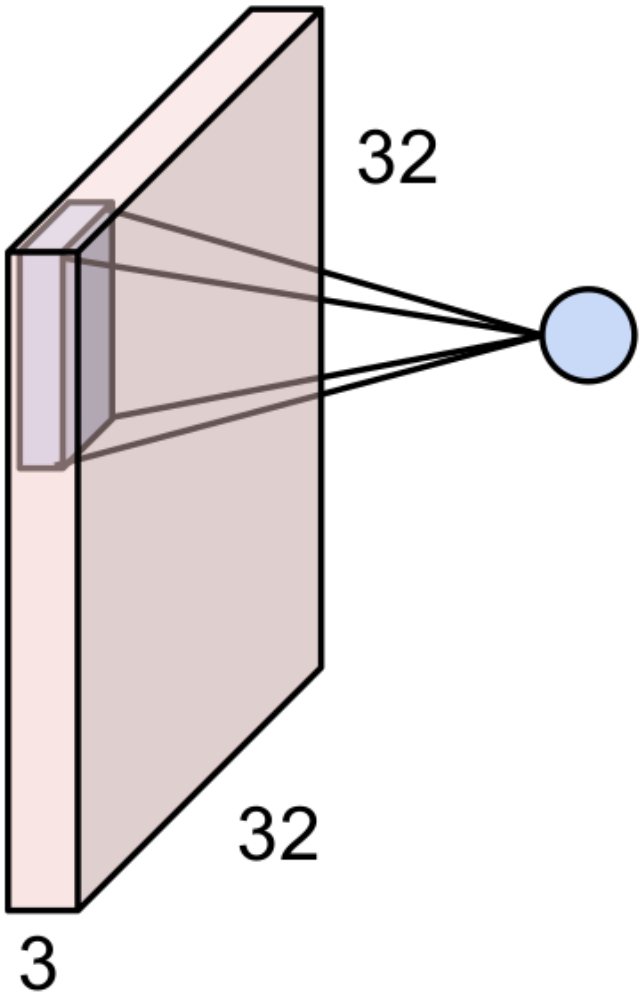
$32 \times 32 \times 3$ image

$5 \times 5 \times 3$ filter w

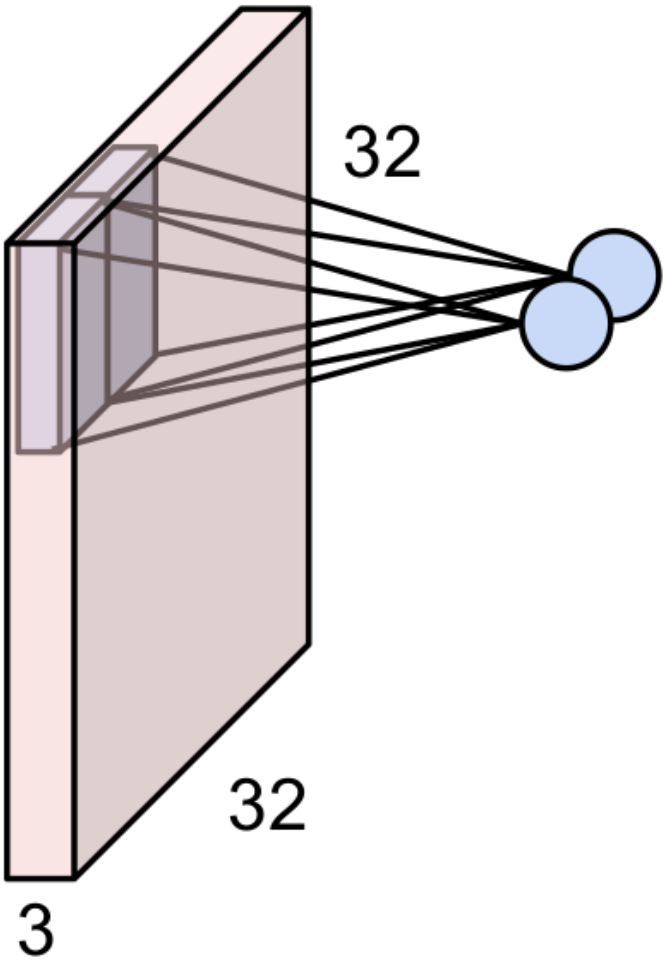
1 number: the result of taking a dot product between the filter and a small $5 \times 5 \times 3$ chunk of the image (i.e. $5 \times 5 \times 3 = 75$-dimensional dot product + bias)

$w^T x + b$

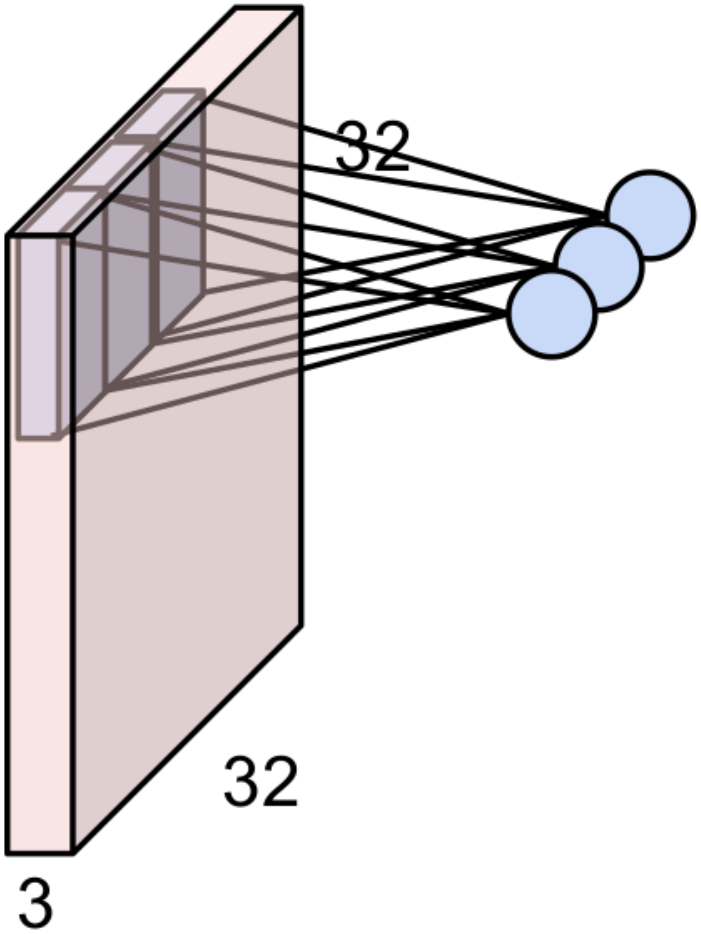
Convolution Layer



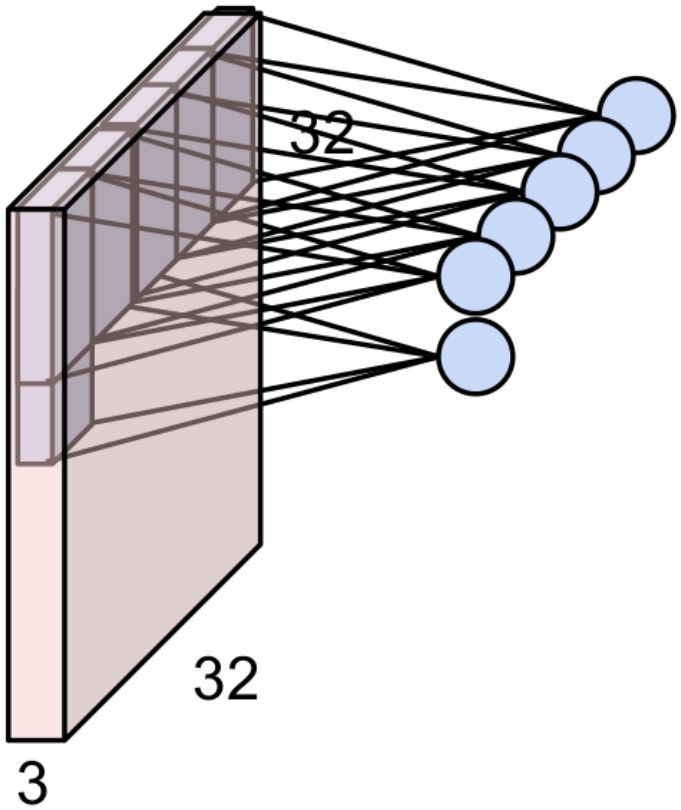
Convolution Layer



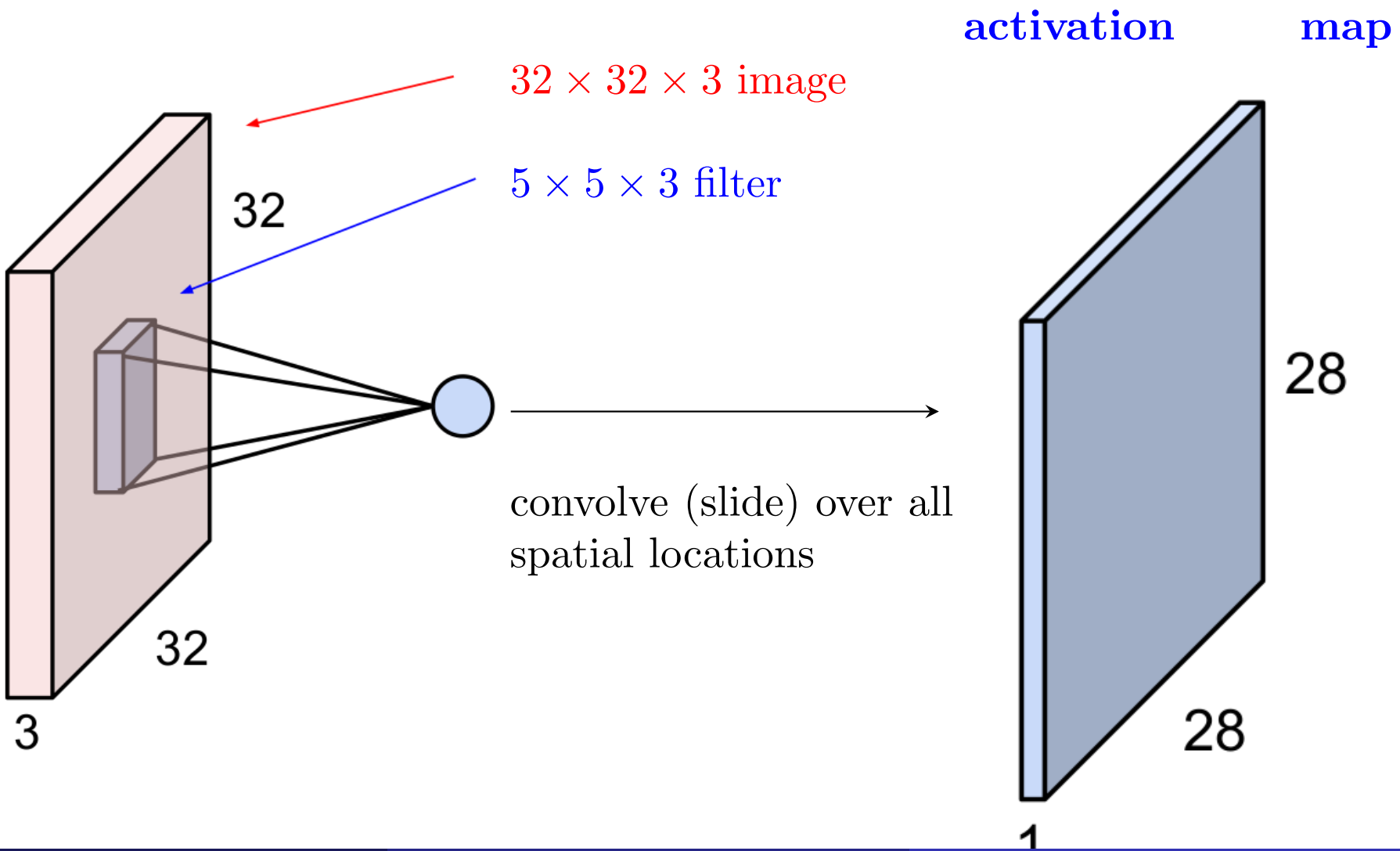
Convolution Layer



Convolution Layer

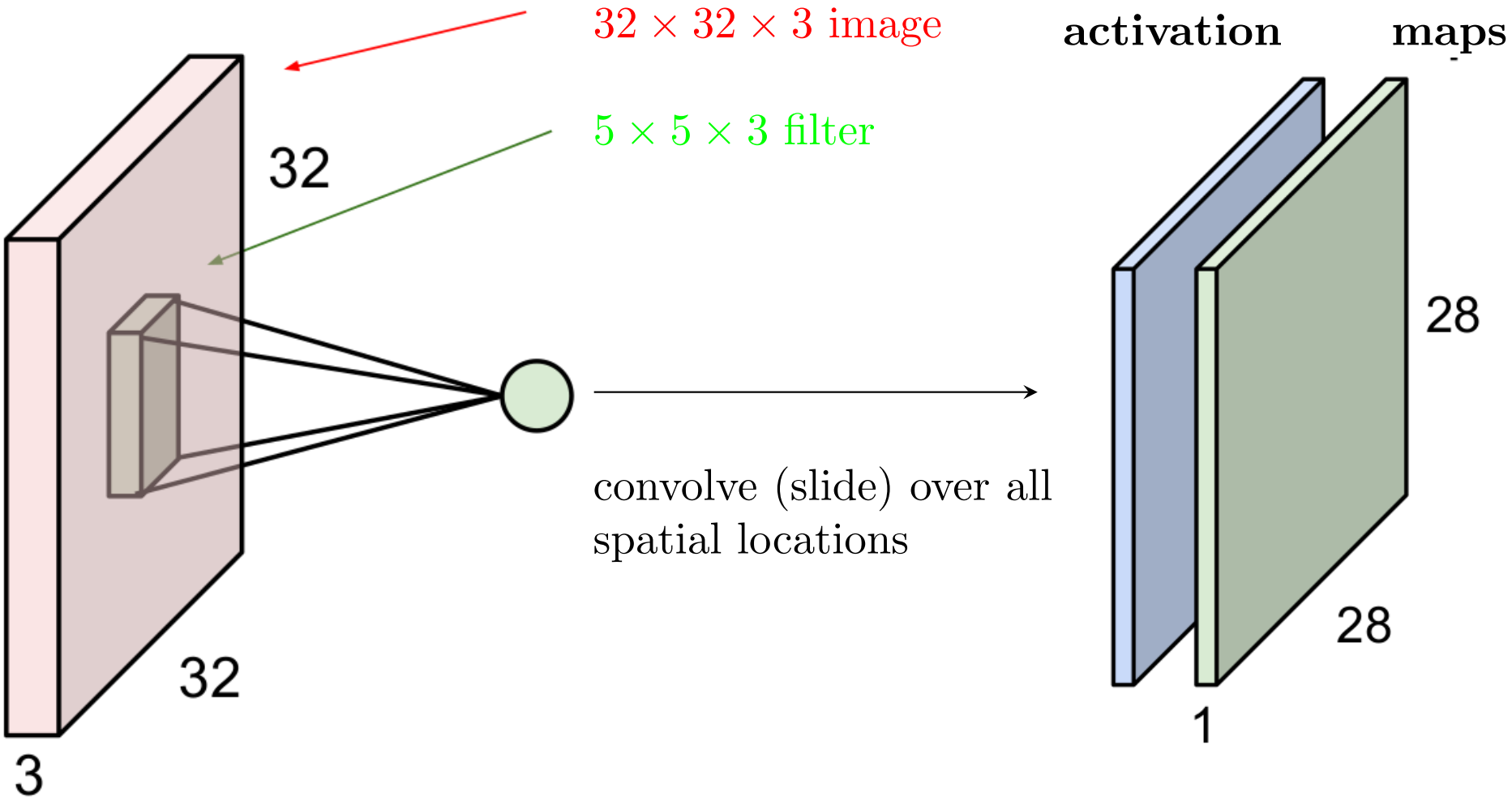


Convolution Layer



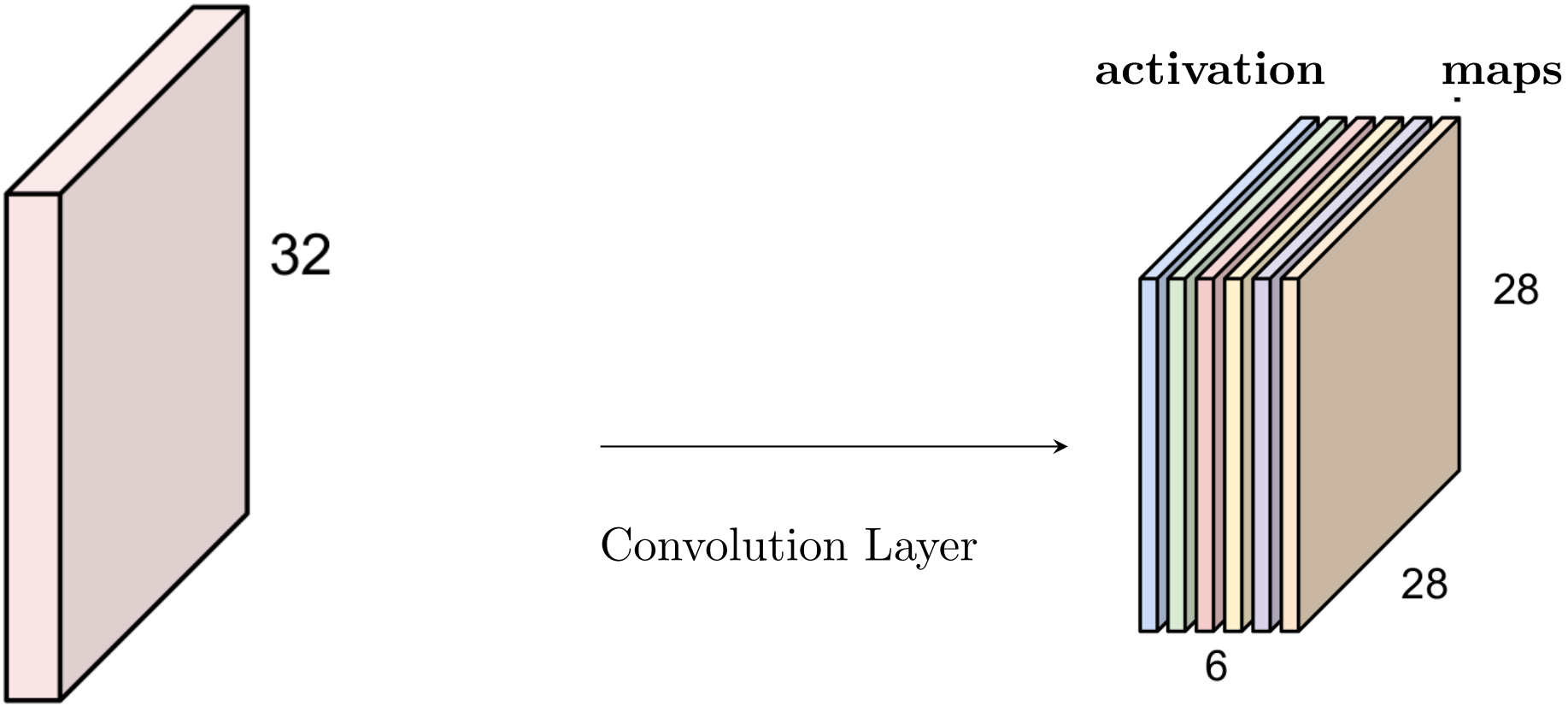
Convolution Layer

consider a second, green filter



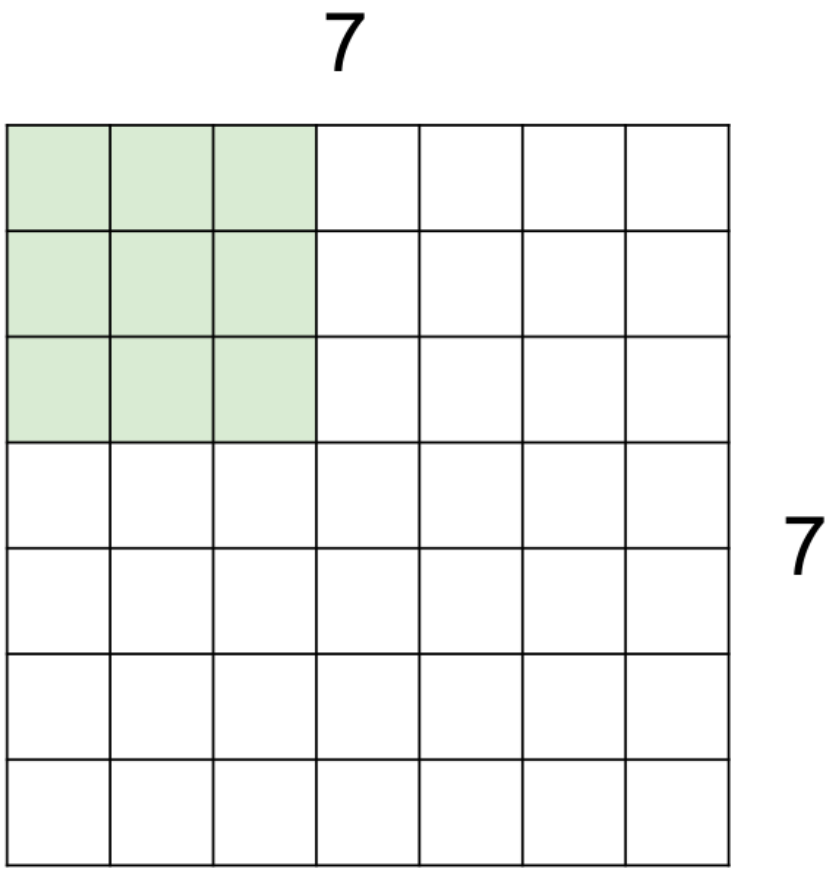
Convolution Layer

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



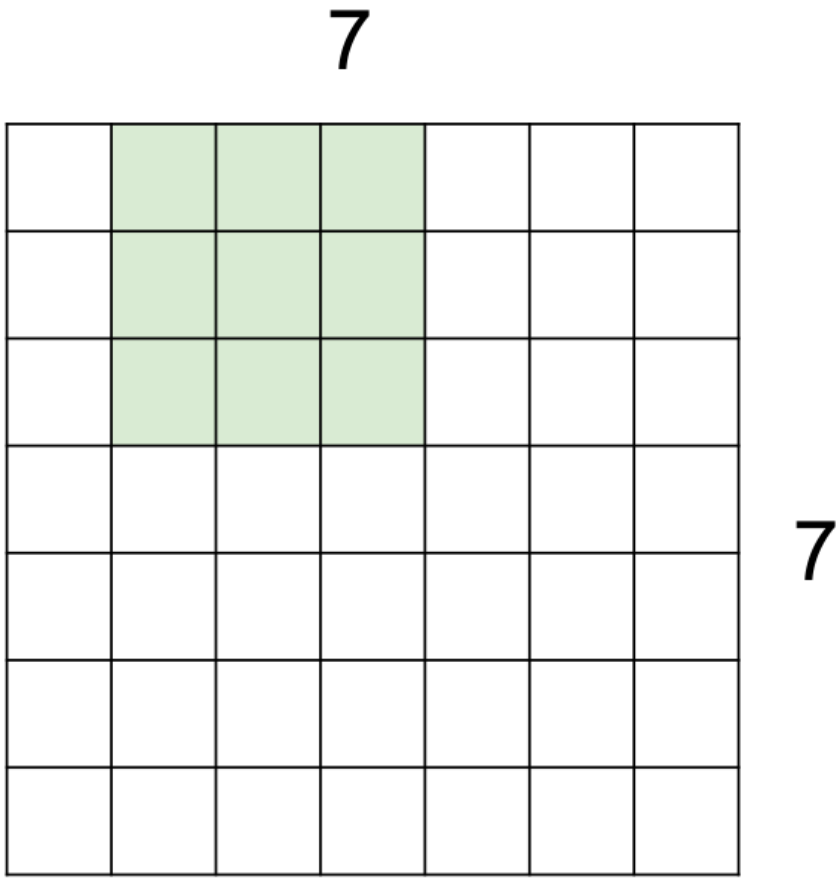
We stack these up to get a "new image" of size $28 \times 28 \times 6$!

A closer look at spatial dimensions:



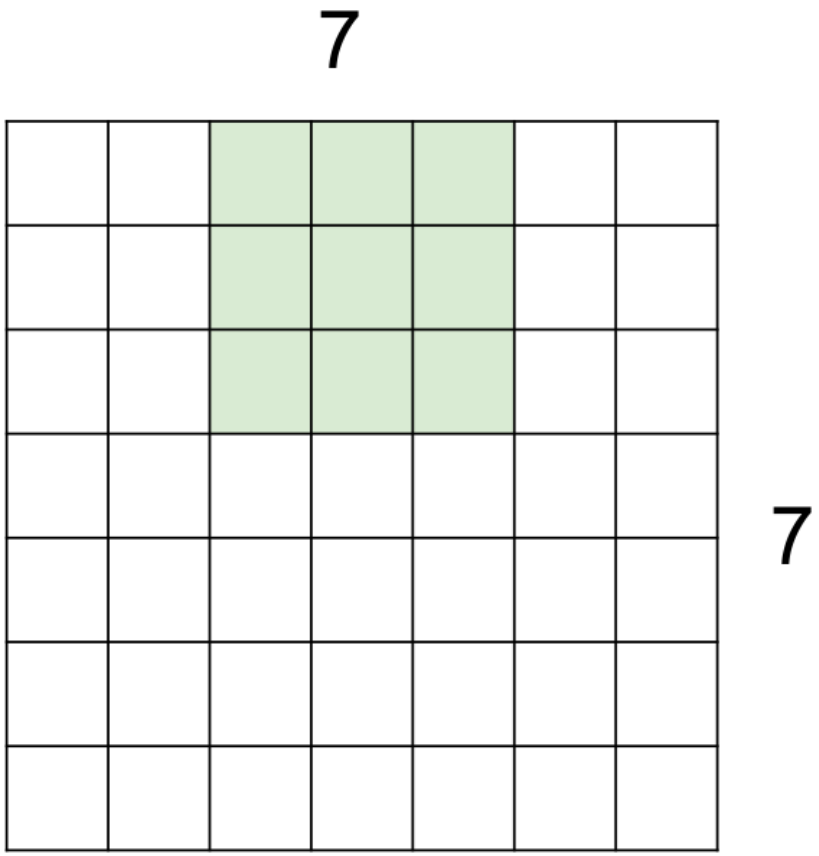
7×7 input (spatially) assume 3×3 filter

A closer look at spatial dimensions:



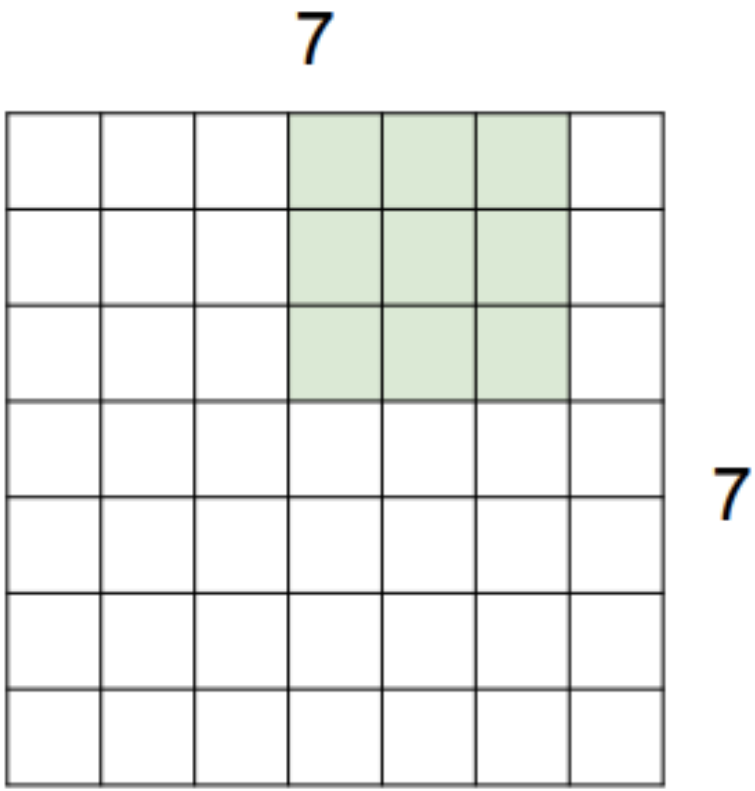
7×7 input (spatially) assume 3×3 filter

A closer look at spatial dimensions:



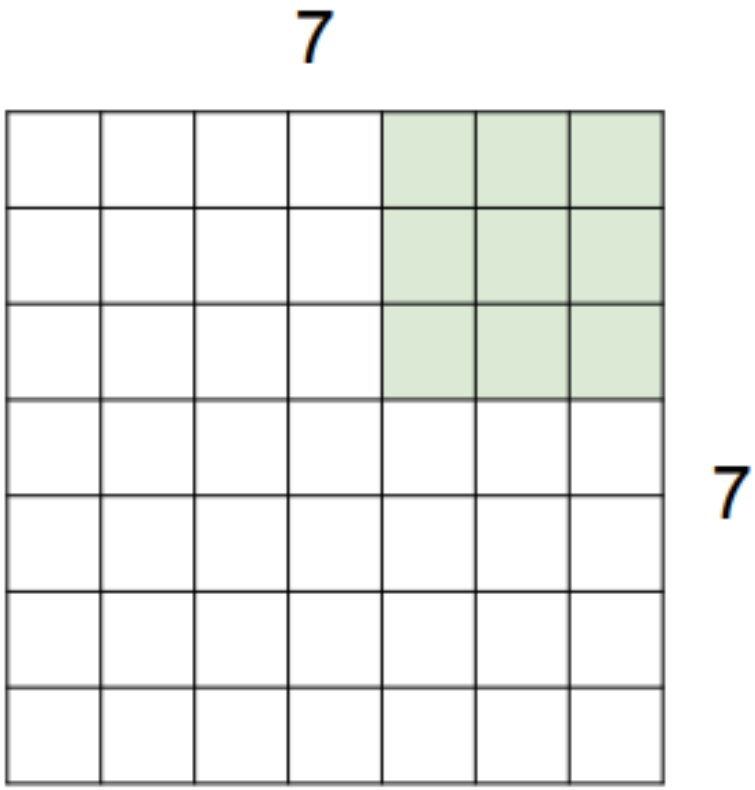
7×7 input (spatially) assume 3×3 filter

A closer look at spatial dimensions:



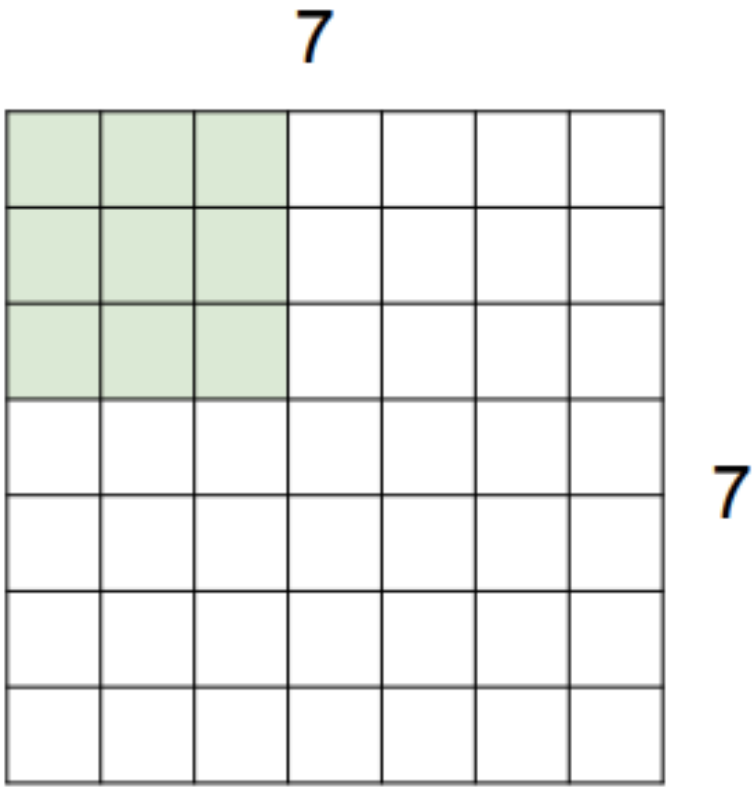
7×7 input (spatially) assume 3×3 filter

A closer look at spatial dimensions:



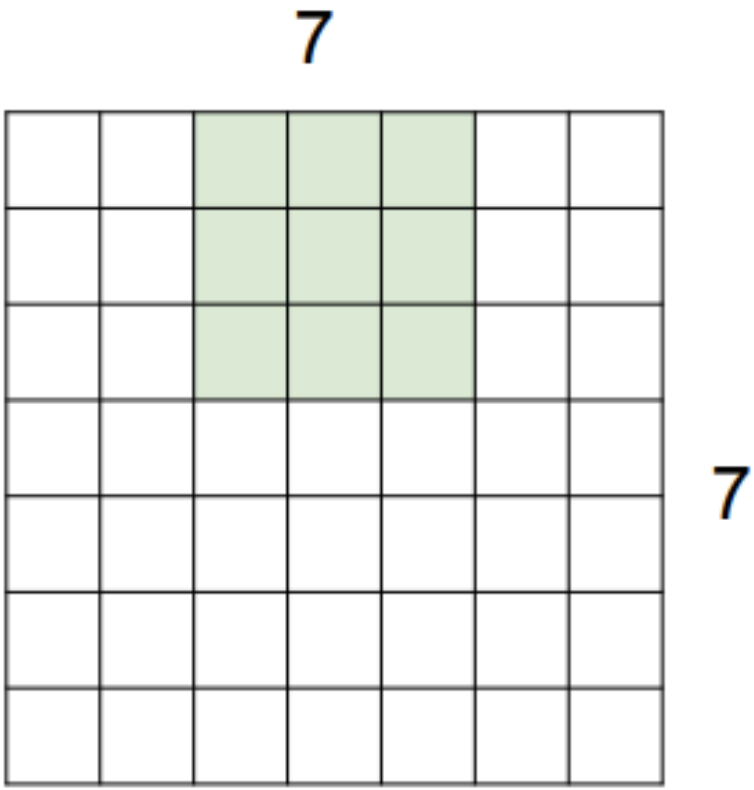
7×7 input (spatially) assume 3×3 filter $\Rightarrow 5 \times 5$ **output**

A closer look at spatial dimensions:



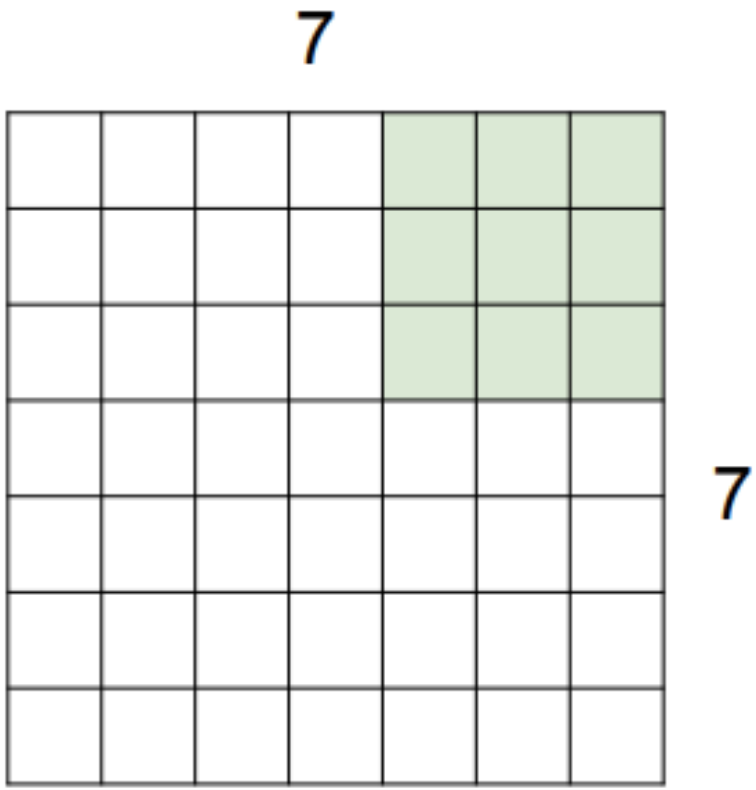
7×7 input (spatially) assume 3×3 filter applied with **stride 2**

A closer look at spatial dimensions:



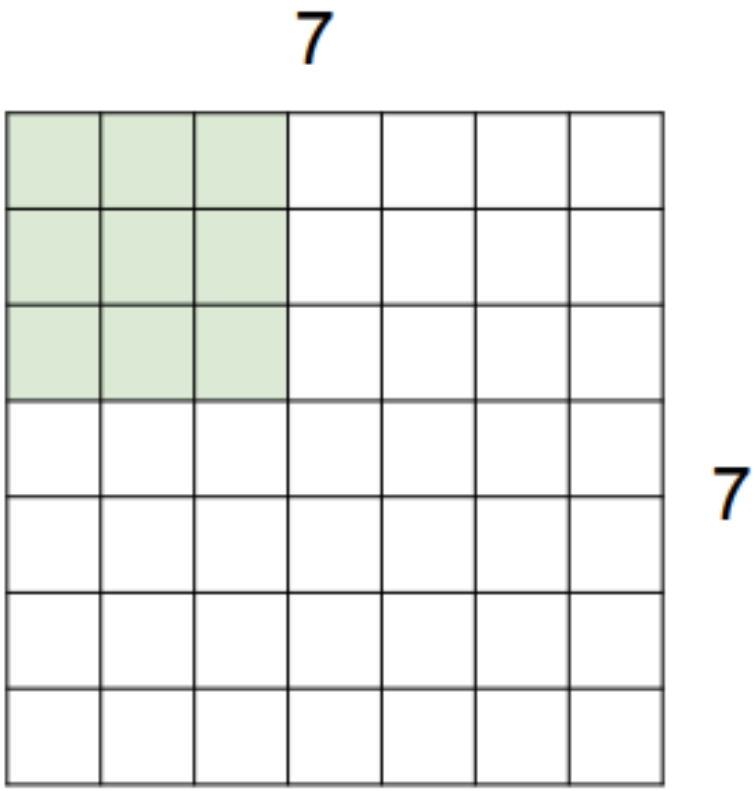
7×7 input (spatially) assume 3×3 filter applied with **stride 2**

A closer look at spatial dimensions:



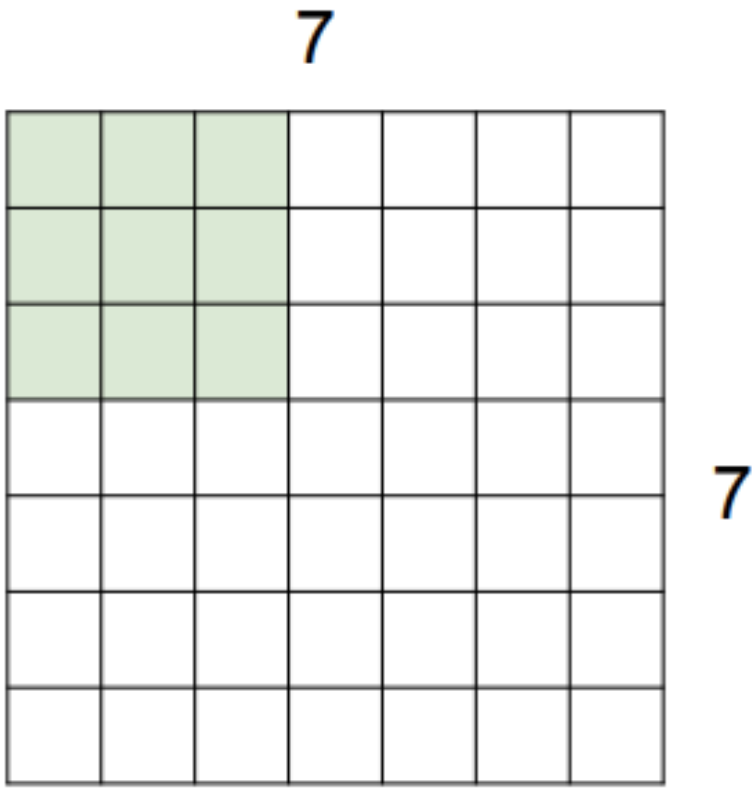
7×7 input (spatially) assume 3×3 filter applied **with stride 2** $=>$ 3×3 **output!**

A closer look at spatial dimensions:



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

A closer look at spatial dimensions:

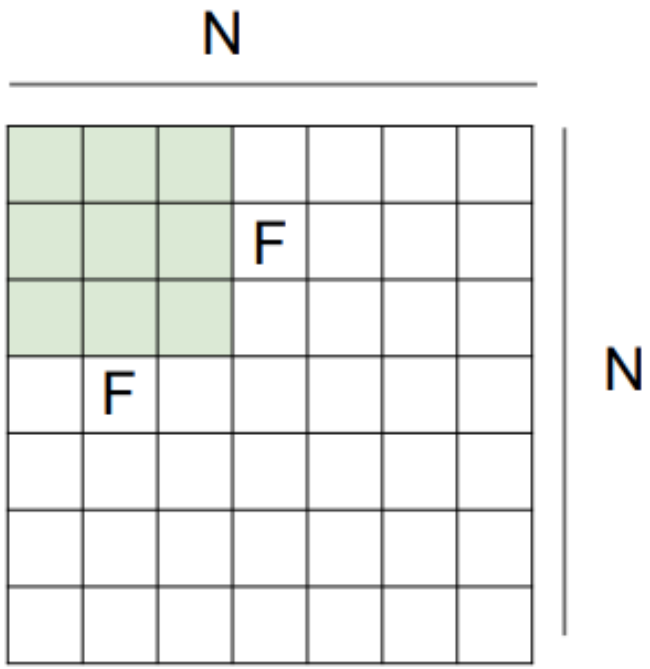


7×7 input (spatially) assume 3×3 filter applied with stride 3?

doesn't fit!

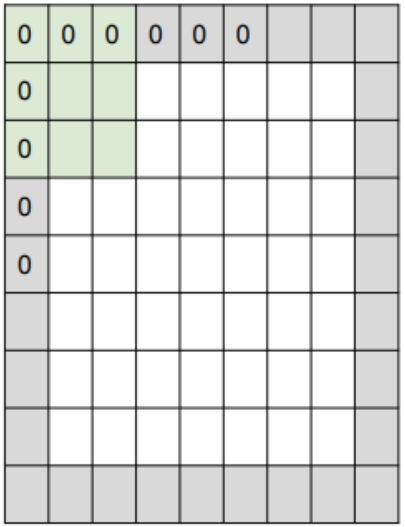
cannot apply 3×3 filter on 7×7 input with stride 3 .

A closer look at spatial dimensions:



Output size: (N - F) / stride +1 e.g. $N = 7, F = 3$:
stride 1 $\Rightarrow (7-3)/1+1 = 5$
stride 2 $\Rightarrow (7-3)/2+1 = 3$
stride 3 $=> (7-3)/3+1 = 2.33$

In practice: Common to zero pad the border

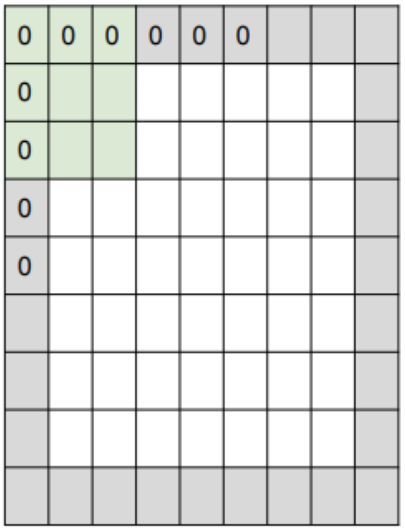


e.g. input 7×7
3×3 filter, applied with stride 1
pad with 1 pixel border => what is the output?

(recall:)

$(N - F)/$ stride $+ 1$

In practice: Common to zero pad the border

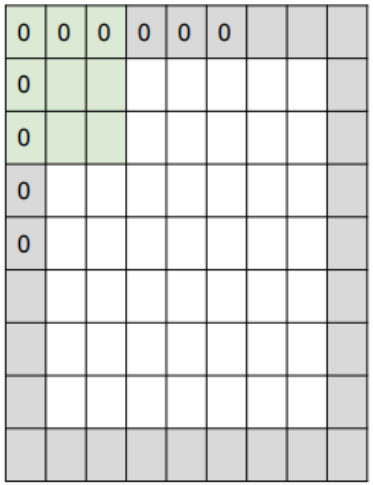


e.g. input 7×7
3x3 filter, applied with **stride 1**
pad with 1 pixel border => what is the output?
7×7 **output!**

(recall:)

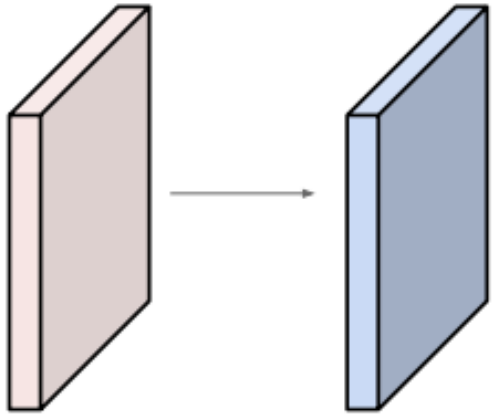
$(N + 2P - F)/$ stride $+ 1$

In practice: Common to zero pad the border



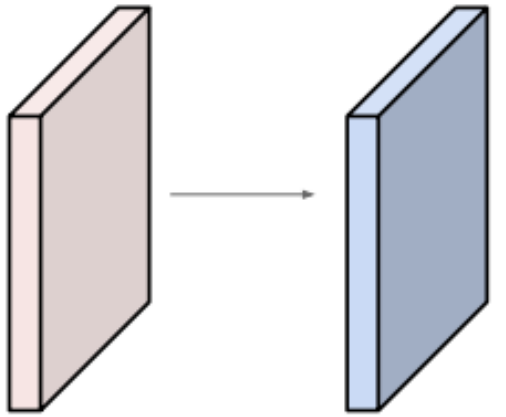
e.g. input 7×7
3x3 filter, applied with **stride 1**
pad with 1 pixel border => what is the output?
7×7 **output!**
In general, common to see CONV layers with stride 1 , filters of size FxF, and zero-padding with $(F-1)/2$. (will preserve size spatially)
e.g. $F = 3 \Rightarrow$ zero pad with 1
$F = 5 \Rightarrow$ zero pad with 2
$F = 7 \Rightarrow$ zero pad with 3

Examples time:



Input volume: $32 \times 32 \times 3$
10 $5 \times 5 \times 3$ filters with stride 1, pad 2
Output volume size: ?

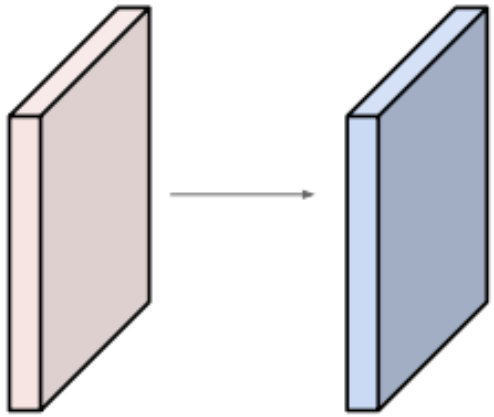
Examples time:



Input volume: $32 \times 32 \times 3$
10 $5 \times 5 \times 3$ filters with stride 1, pad 2
Output volume size:

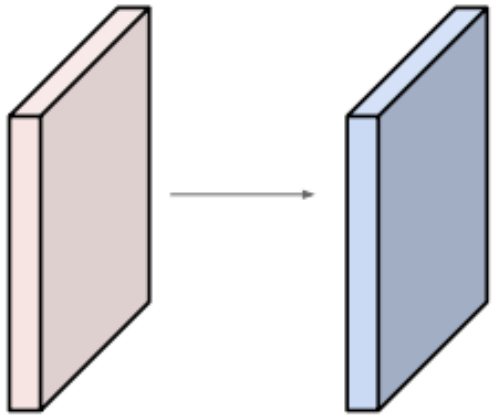
$(32 + 2 * 2 - 5)/1 + 1 = 32$ spatially, so
$32 \times 32 \times 10$

Examples time:



Input volume: $32 \times 32 \times 3$
10 $5 \times 5 \times 3$ filters with stride 1 , pad 2
Number of parameters in this layer?

Examples time:



Input volume: $32 \times 32 \times 3$ 10 $5 \times 5 \times 3$ filters with stride 1, pad 2
Number of parameters in this layer? each filter has $5 * 5 * 3 + 1 = 76$ params (+1 for bias)

$$=> 76 * 10 = 760$$

Convolution layer: summary

Let's assume input is $\mathrm{W}_1 \times \mathrm{H}_1 \times \mathrm{C}$ Conv layer needs 4 hyperparameters:

- Number of filters K
- The filter size F
- The stride S
- The zero padding P

This will produce an output of $\mathrm{W}_2 \times \mathrm{H}_2 \times \mathrm{K}$ where:
- $W_2 = (W_1 - F + 2P)/S + 1$
$-\mathrm{H}_2 = (\mathrm{H}_1 - \mathrm{F} + 2\mathrm{P})/\mathrm{S} + 1$
Number of parameters: F^2CK and K biases

Convolution layer: summary

Let's assume input is $W_1 \times H_1 \times C$ Conv layer needs 4 hyperparameters:

- Number of filters K
- The filter size F
- The stride S
- The zero padding P

This will produce an output of $W_2 \times H_2 \times K$ where:
- $W_2 = (W_1 - F + 2P)/S + 1$
$-H_2 = (H_1 - F + 2P)/S + 1$
Number of parameters: F^2CK and K biases
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)
- $F = 1, S = 1, P = 0$

1. History of CNN

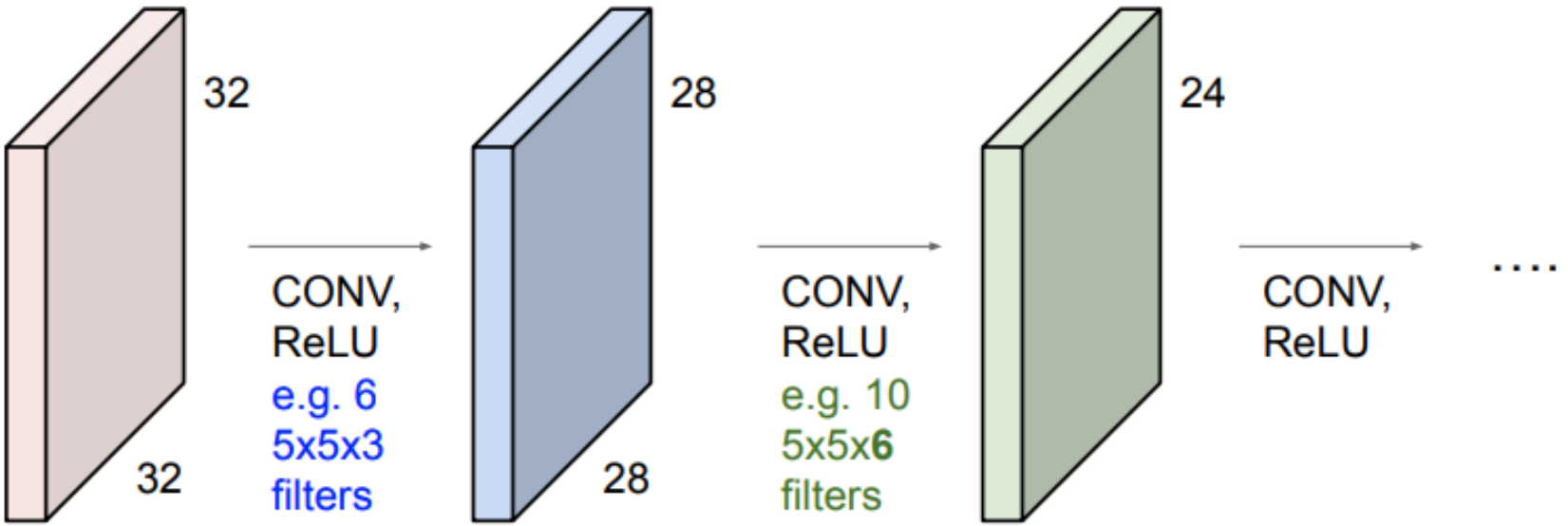
2. Convolutional Neural Networks
   - Preview
   - Convolutional layer
   - Pooling layer
   - Fully connected layer

Pooling layer

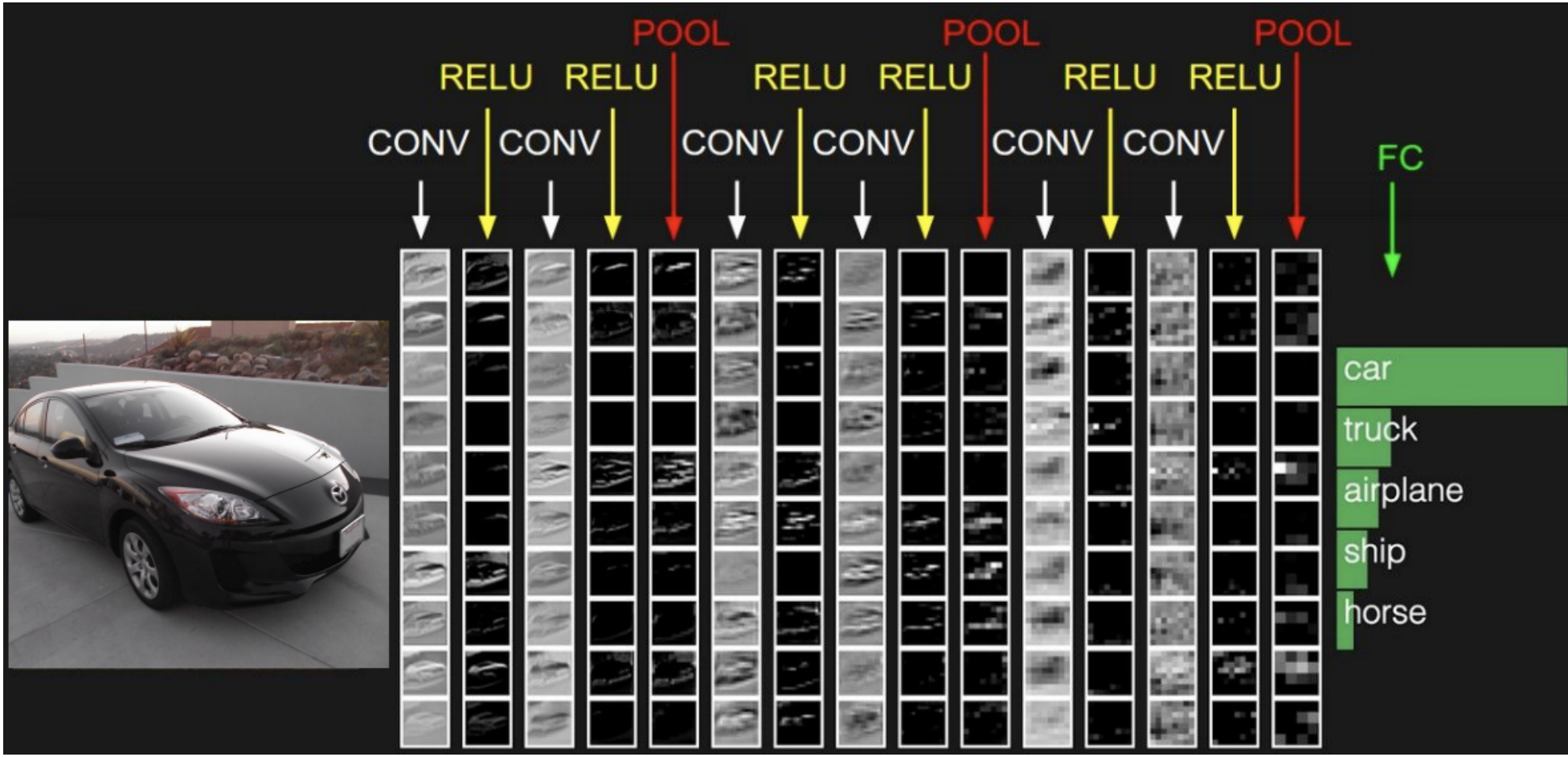
Remember back to...

E.g. 32x32 input convolved repeatedly with 5×5 filters shrinks volumes spatially! $(32 -> 28 -> 24 \ldots .$.

Shrinking too slow is very costly; Shrinking too fast is not good, doesn't work well.

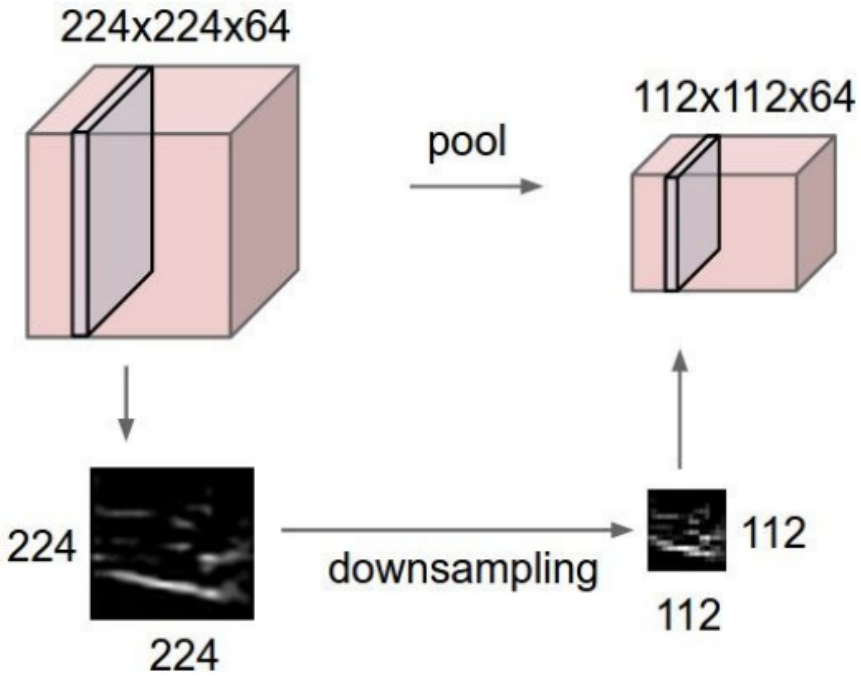


Pooling layer

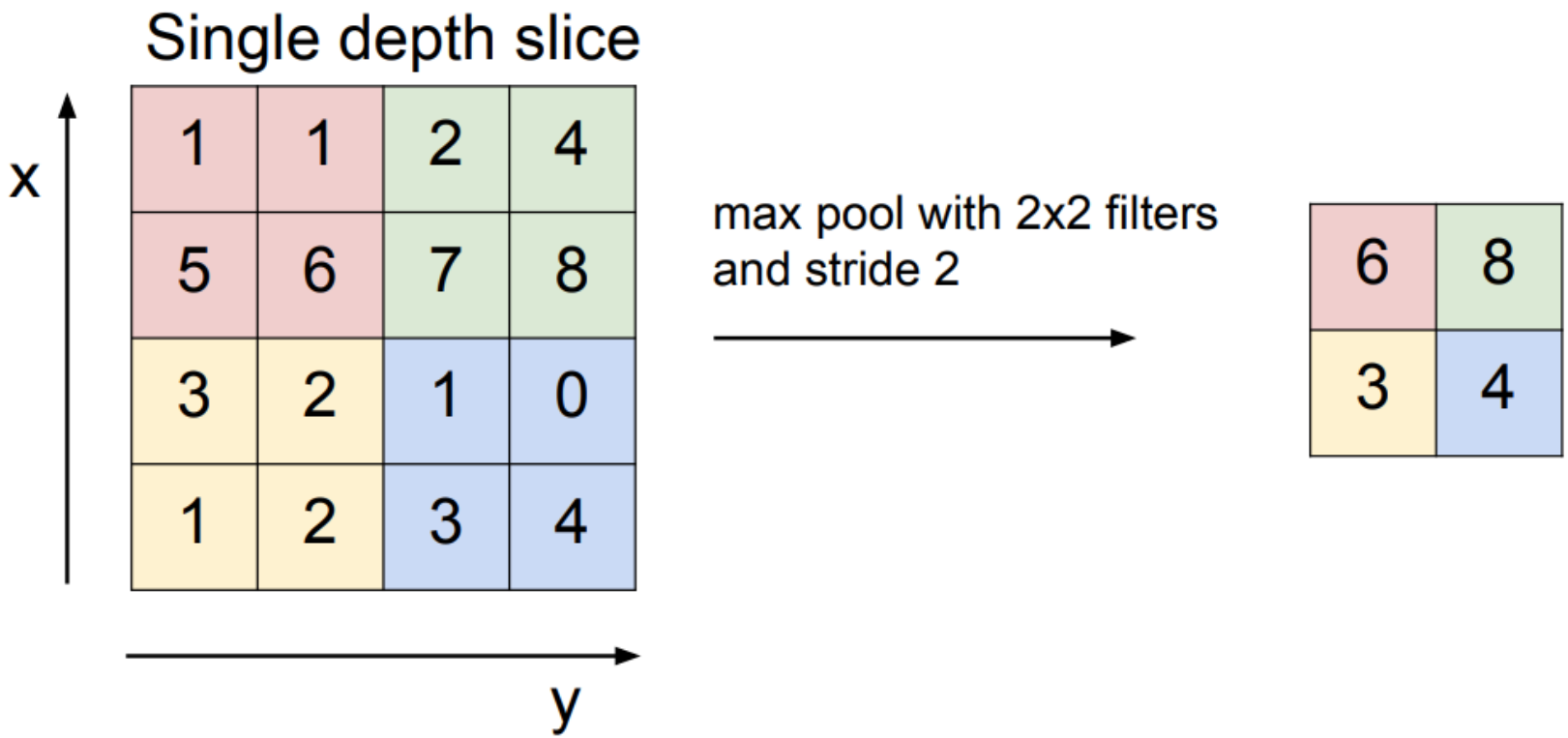


Pooling Layer

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



MAX POOLING

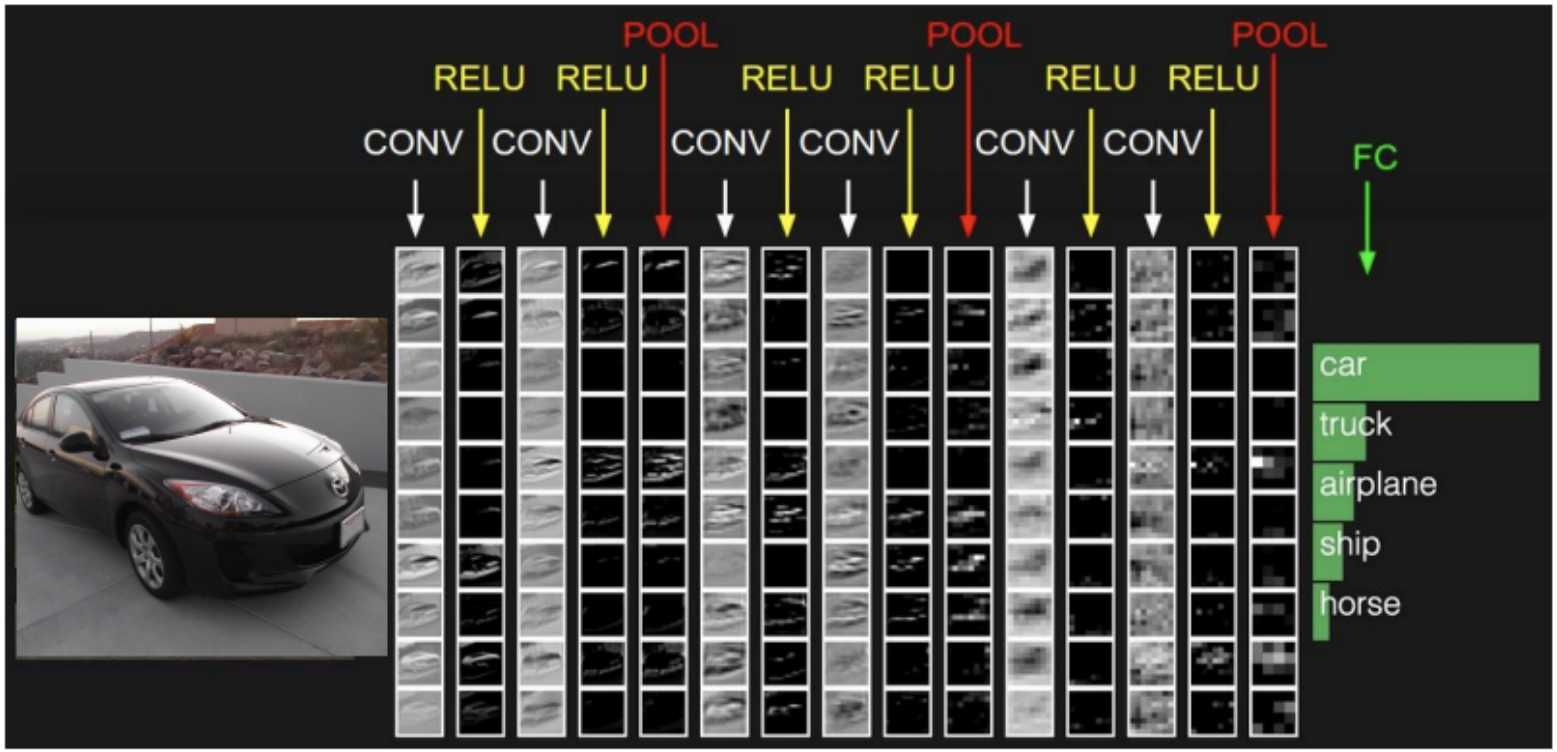

Single depth slice
x
y
1 1 2 4
5 6 7 8
3 2 1 0
1 2 3 4
max pool with 2x2 filters
and stride 2
6 8
3 4

1. History of CNN

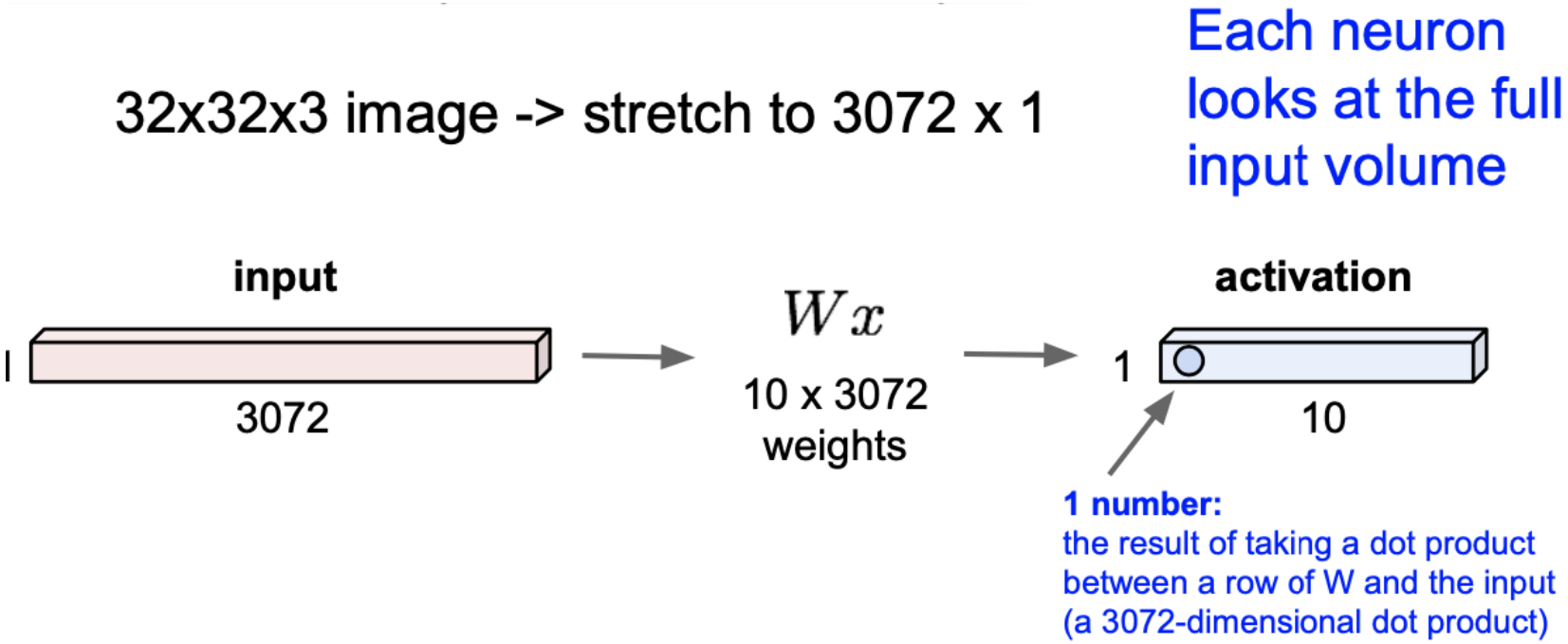
2. Convolutional Neural Networks
   - Preview
   - Convolutional layer
   - Pooling layer
   - Fully connected layer

Fully Connected Layer (FC layer)

Contains neurons that connect to the entire input volume, as in ordinary Neural Networks



Fully Connected Layer (FC layer)



Further reading

Some resources for learning CNNs:

- Course: CS231n: Deep Learning for Computer Vision
- Packages: Pytorch, Tensorflow