COMPUTER VISION

LECTURE 13 16.10.2019

Laura Ruotsalainen, Associate Professor
Department of Computer Science

READINGS

Goodfellow et al Deep learning

Neural networks

http://cs231n.github.io/neural-networks-1/

http://cs231n.github.io/neural-networks-2/

http://cs231n.github.io/neural-networks-3/

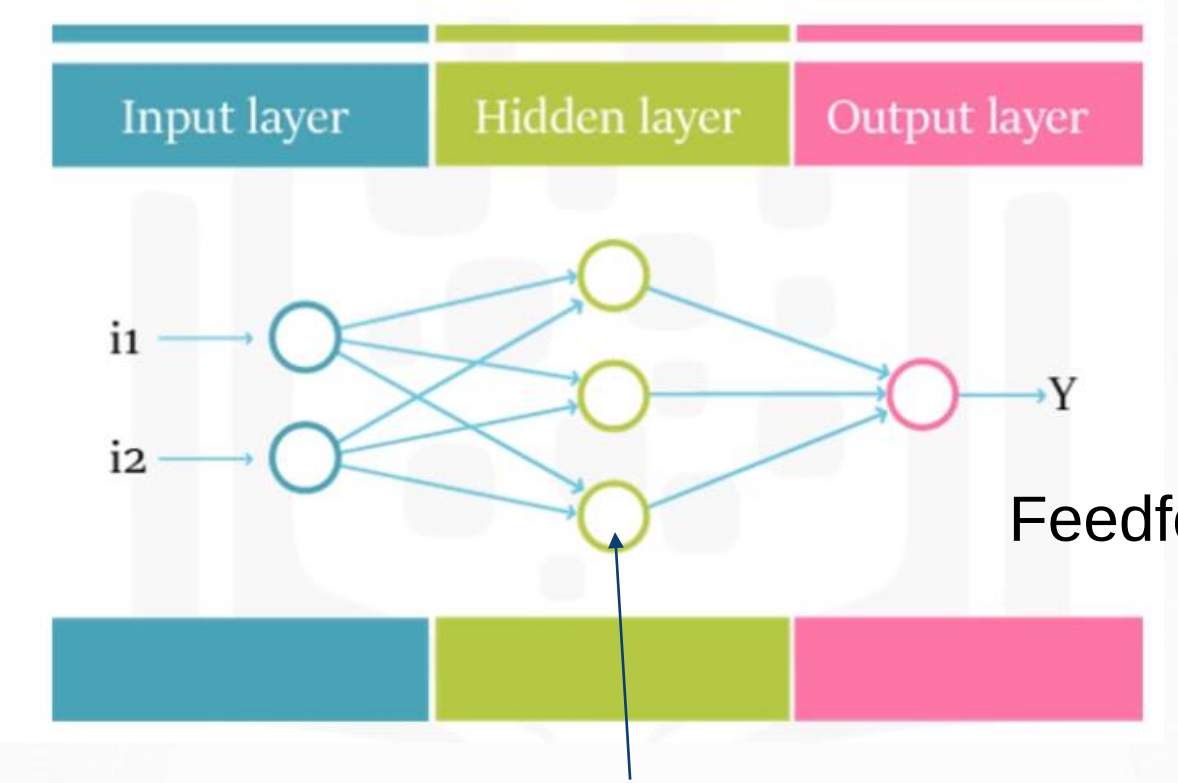
http://cs231n.github.io/neural-networks-case-study/

Convolutional neural networks

http://cs231n.github.io/convolutional-networks/

Architecture of Neural Networks

Two-layer neural network



Feedforward

Neuron or *unit*

RECAP: LINEAR CLASSIFICATION

Have score function and loss function

Currently, score function is based on linear classifier

$$f(x,W) = Wx + b$$

Find W and b to minimize loss, e.g. cross-entropy loss

$$L = \frac{1}{N}\sum_i -\log\left(\frac{e^{f_{y_i}}}{\sum_j e^{f_j}}\right)$$

Slide credit Fei-Fei Li

SIDE NOTE: GRADIENT DESCENT

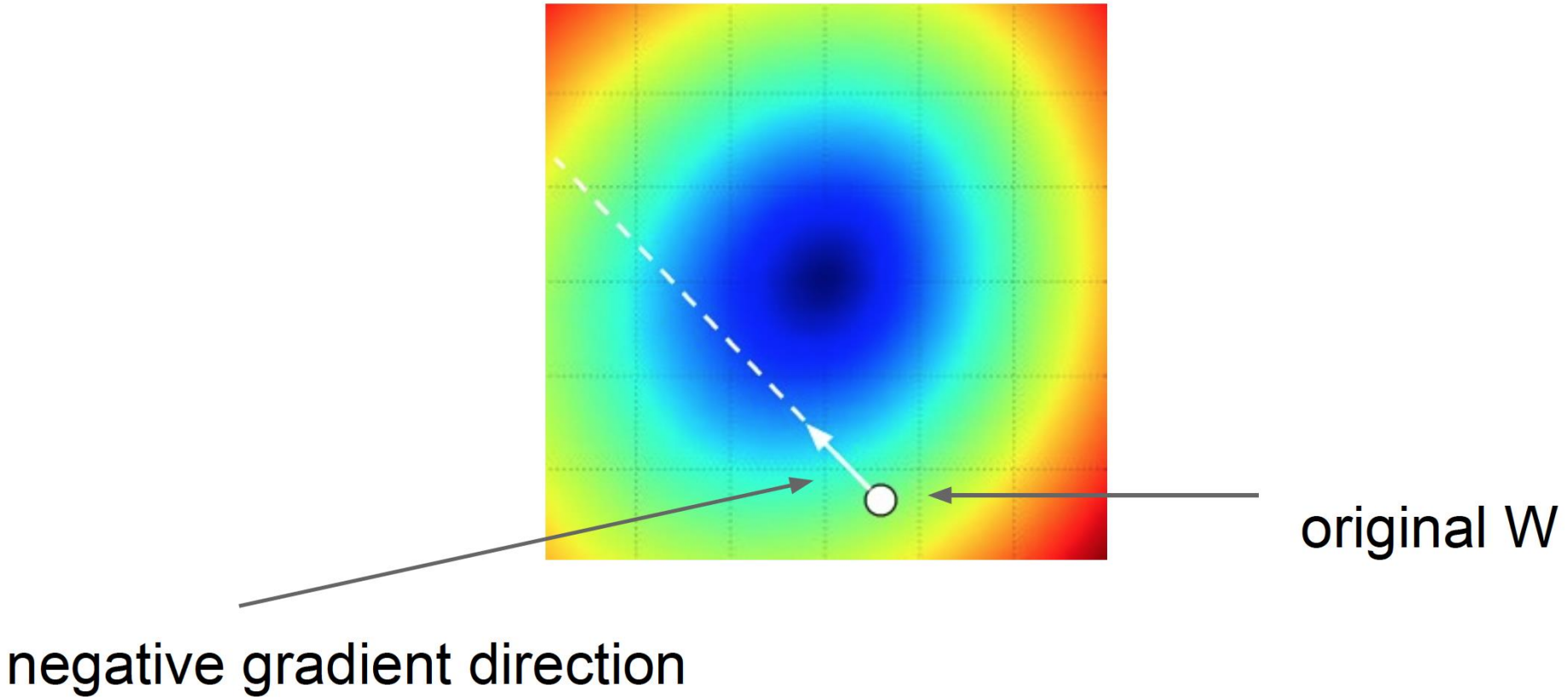
How do we find the best **W** and **b** parameters?

In general: *gradient descent*

1. Start with a guess of a good **W** and **b** (or randomly initialize them)
2. Compute the loss function for this initial guess and the *gradient* of the loss function
3. Step some distance in the negative gradient direction (direction of steepest descent)
4. Repeat steps 2 & 3

Note: efficiently performing step 2 for deep networks is called *backpropagation*

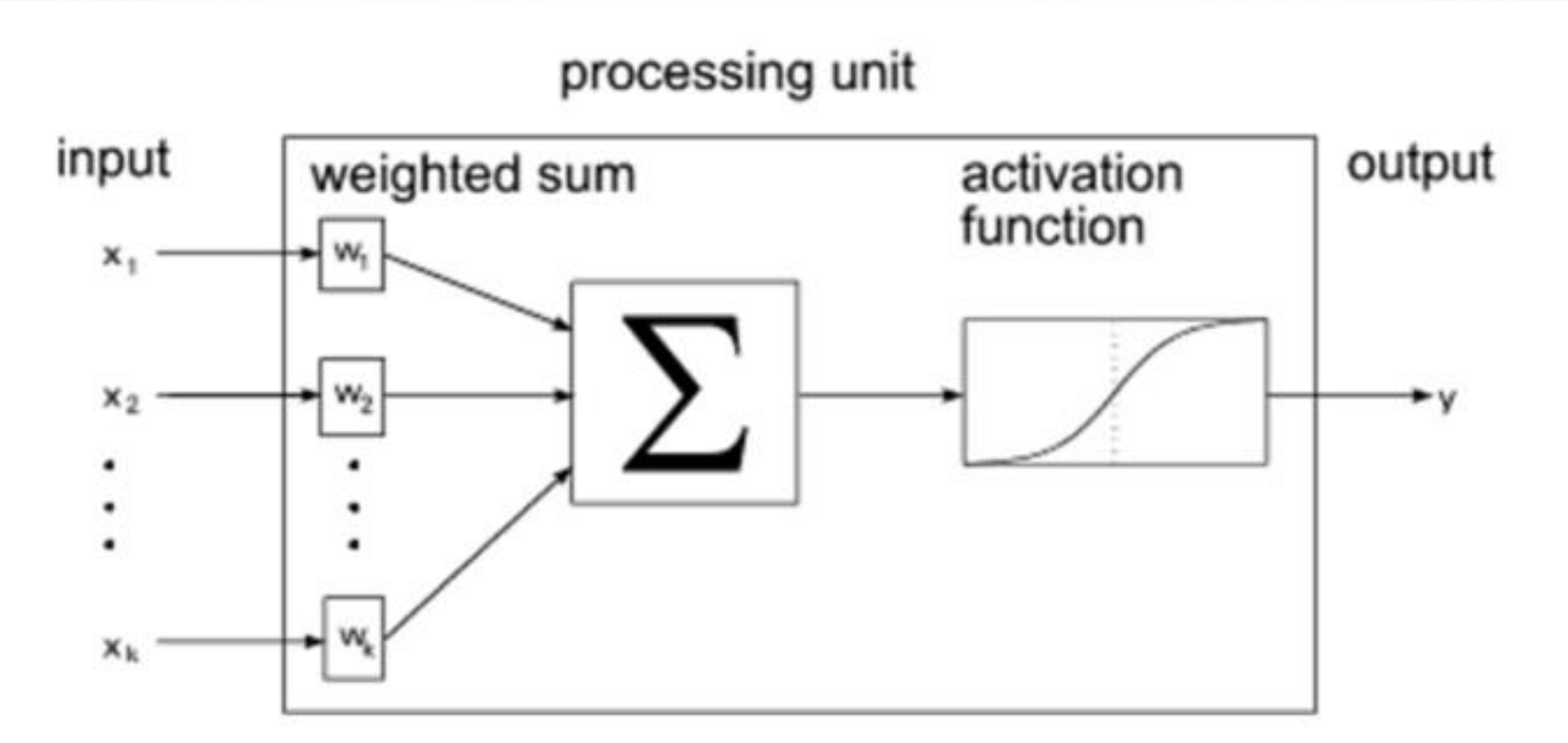
Slide credit Fei-Fei Li



Gradient descent: walk in the direction opposite gradient

- **Q**: How far?
- **A**: Step size: *learning rate*
- Too big: will miss the minimum
- Too small: slow convergence

PERCEPTON – SIMPLE NEURAL NETWORK



NEURAL NETWORKS

(**Before**) Linear score function: $f = Wx$

Slide credit Fei-Fei Li

NEURAL NETWORKS

(**Before**) Linear score function: $f = Wx$

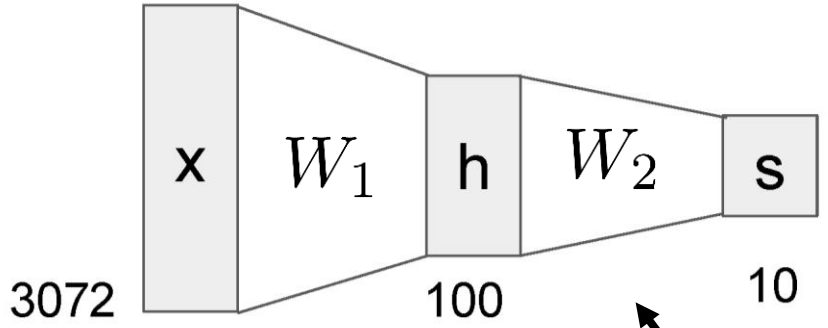
(**Now**) 2-layer Neural Network $f = W_2 \max(0, W_1 x)$

Slide credit Fei-Fei Li

NEURAL NETWORKS

(**Before**) Linear score function: $f = Wx$

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



W_1
(100 x 3072 matrix)

h
100D intermediate vector

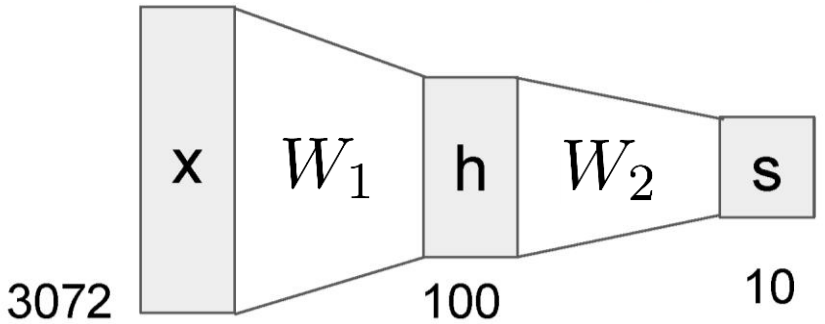
W_2
(10 x 100 matrix)

Slide credit Fei-Fei Li

NEURAL NETWORKS

(**Before**) Linear score function: $f = Wx$

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



Total number of weights to learn:

3,072 x 100 + 100 x 10 = 308,200

Slide credit Fei-Fei Li

NEURAL NETWORKS

(**Before**) Linear score function: $f = Wx$

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

or 3-layer Neural Network

$$f = W_3 \max(0, W_2 \max(0, W_1 x))$$

Slide credit Fei-Fei Li

NEURAL NETWORKS

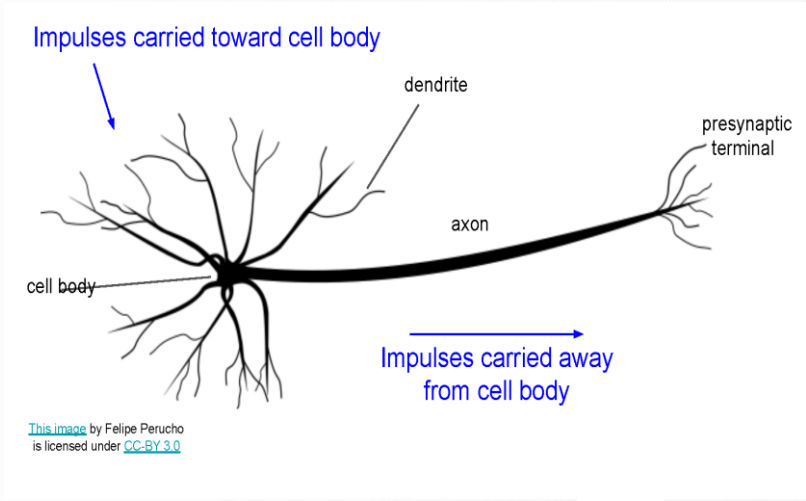
Very coarse generalization:

Linear functions chained together and separated by non-linearities (*activation functions*), e.g. “max”

$$f = W_3 \max(0, W_2 \max(0, W_1 x))$$

Why separate linear functions with non-linear functions?

Very roughly inspired by real neurons

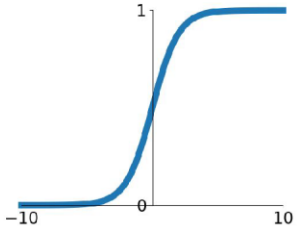


Slide credit Fei-Fei Li

Activation functions

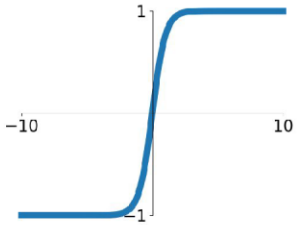
Sigmoid

$$\sigma(x) = \frac{1}{1+e^{-x}}$$



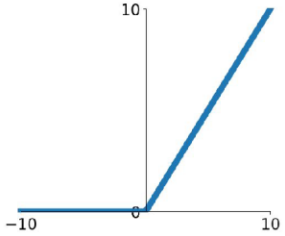
tanh

$$\tanh(x)$$



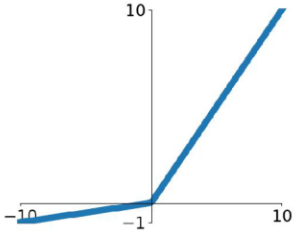
ReLU

$$\max(0, x)$$



Leaky ReLU

$$\max(0.1x, x)$$

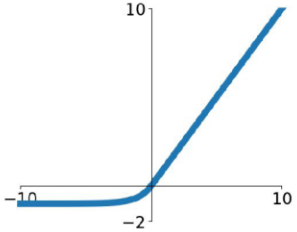


Maxout

$$\max(w_1^T x + b_1, w_2^T x + b_2)$$

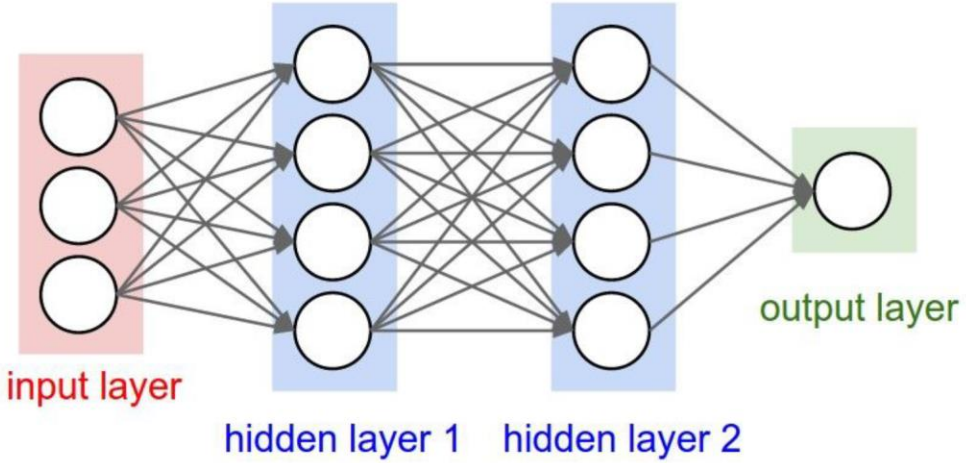
ELU

$$\begin{cases} x & x \geq 0 \\ \alpha(e^x - 1) & x < 0 \end{cases}$$



Slide credit Fei-Fei Li

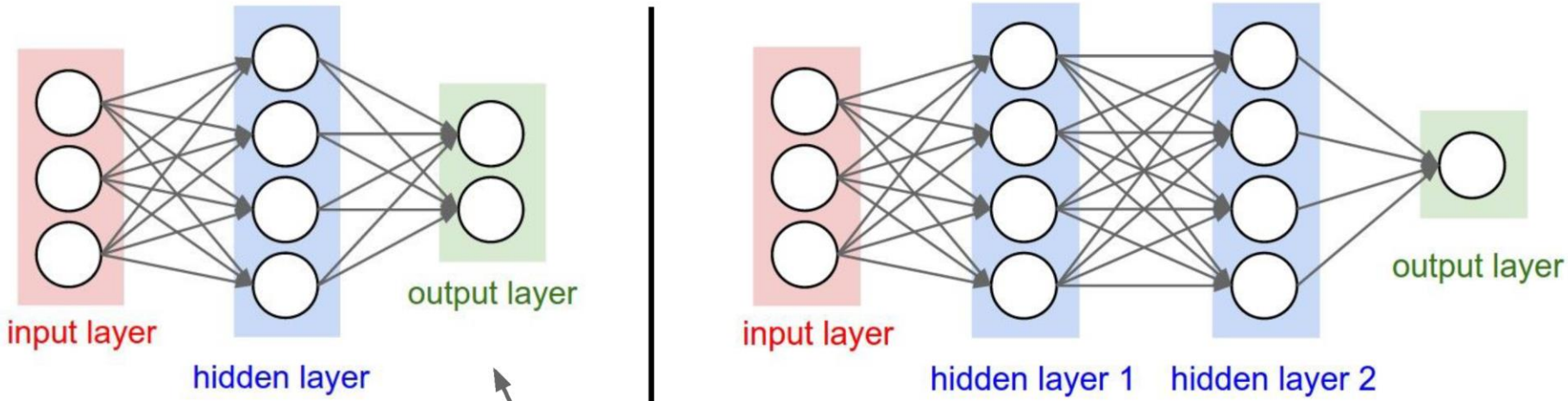
Example feed-forward computation of a neural network




```
# forward-pass of a 3-layer neural network:
f = lambda x: 1.0/(1.0 + np.exp(-x)) # activation function (use sigmoid)
x = np.random.randn(3, 1) # random input vector of three numbers (3x1)
h1 = f(np.dot(W1, x) + b1) # calculate first hidden layer activations (4x1)
h2 = f(np.dot(W2, h1) + b2) # calculate second hidden layer activations (4x1)
out = np.dot(W3, h2) + b3 # output neuron (1x1)
```


Slide credit Fei-Fei Li

Neural networks: Architectures



"2-layer Neural Net", or
"1-hidden-layer Neural Net"

"3-layer Neural Net", or
"2-hidden-layer Neural Net"

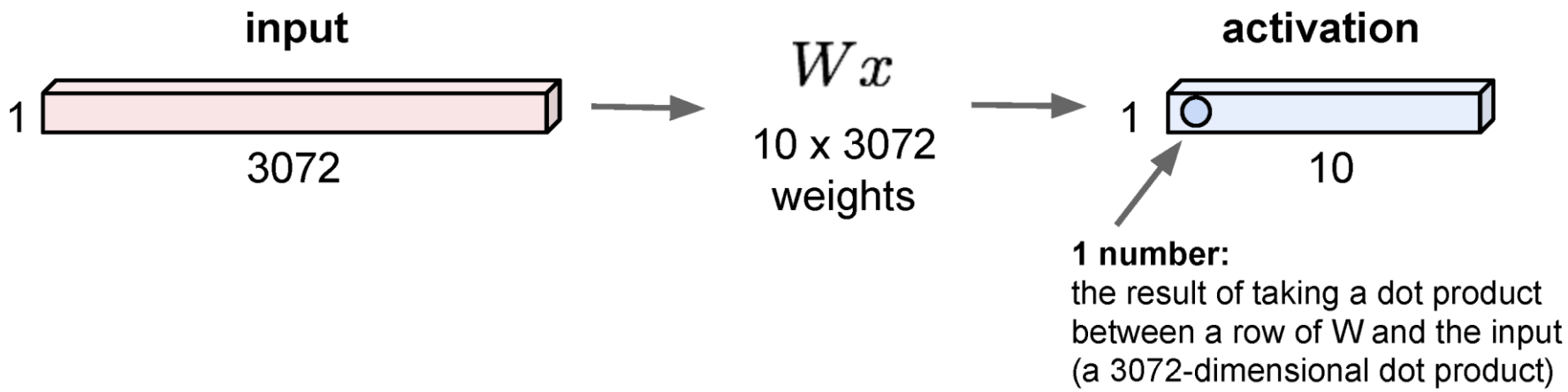
"Fully-connected" layers

Deep networks typically have many layers and potentially millions of parameters

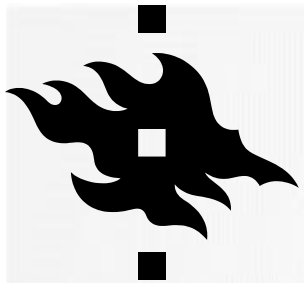
Slide credit Fei-Fei Li

Fully Connected Layer

32x32x3 image -> stretch to 3072 x 1



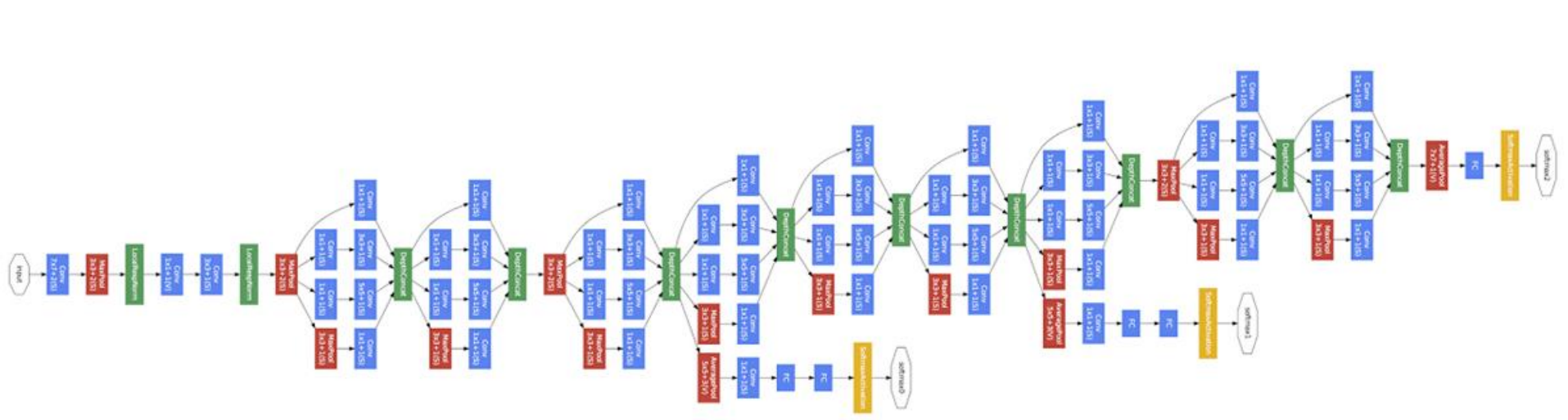
Slide credit Fei-Fei Li



Deep Learning

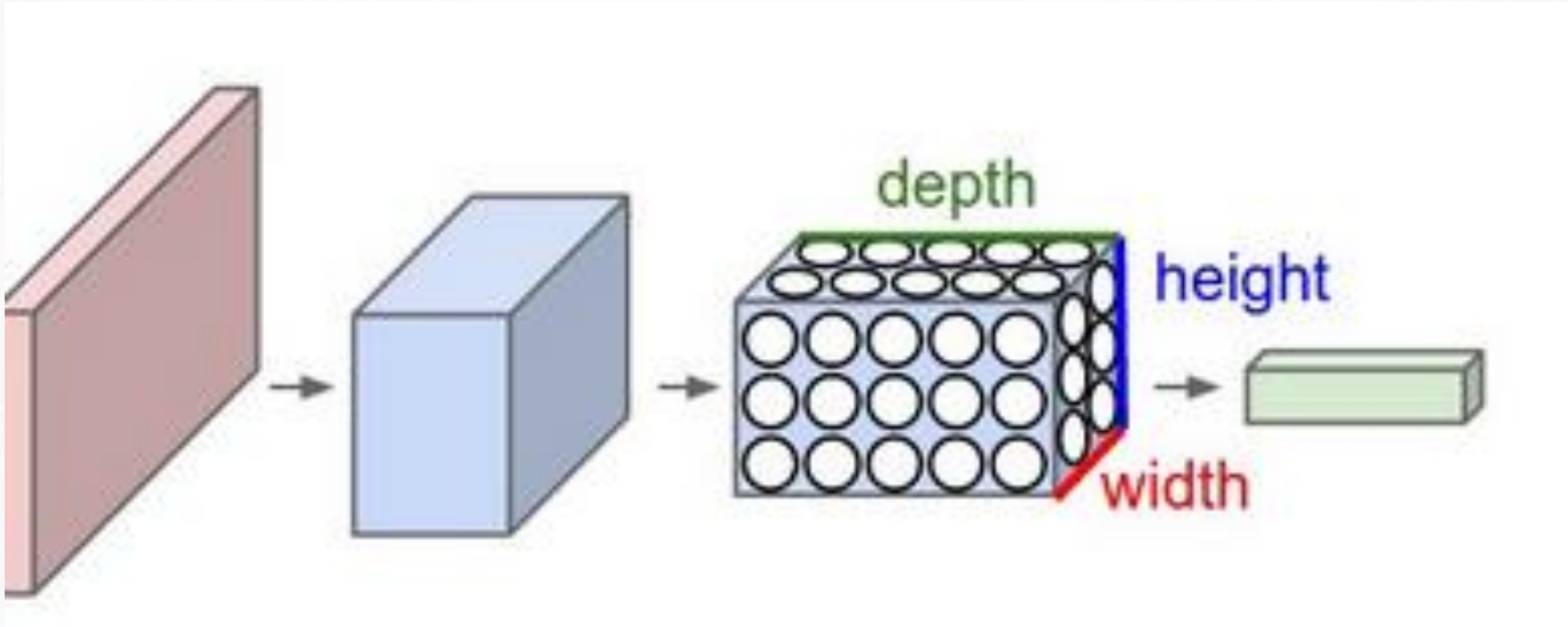
- The real breakthrough in deep learning was to realize that it's practical to go beyond the shallow 1- and 2-hidden layer networks that dominated work until the mid-2000s
  - significant breakthrough, opening up the exploration of much more expressive models
- Beyond that, the number of layers is not of primary fundamental interest, the use of deeper networks is a tool to use to help achieve other goals - like better classification accuracies
- Deep Learning algorithms developed that discover patterns without labeled data => Unsupervised learning

DEEP NEURAL NETWORK



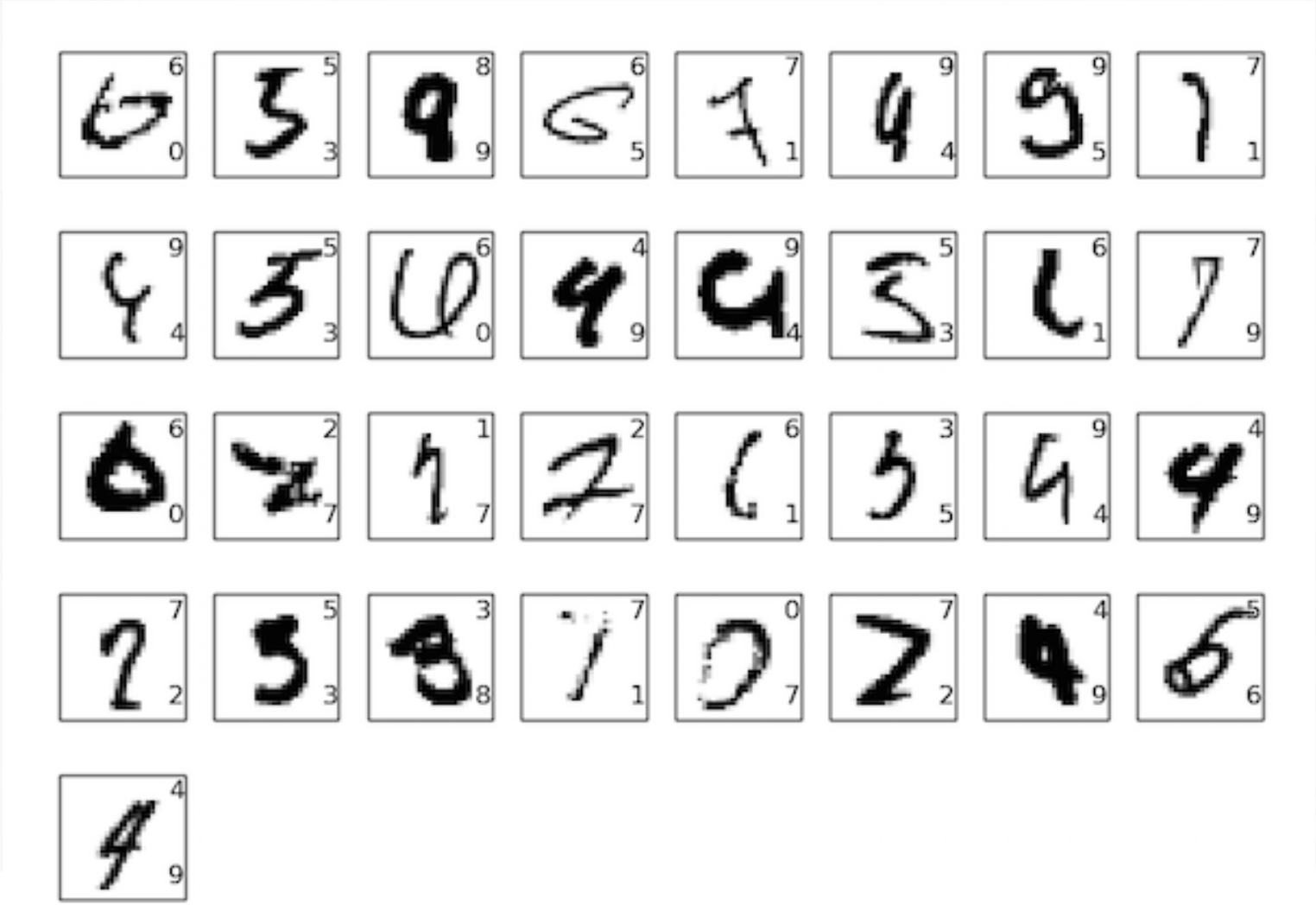
Inception network (Szegedy et al, 2015), 22 layers

- For a complex problem (such as in image classification), more hidden layers are required
- Hundreds of hidden layers improve the accuracy of the classification => deep learning



Slide credit Fei-Fei Li

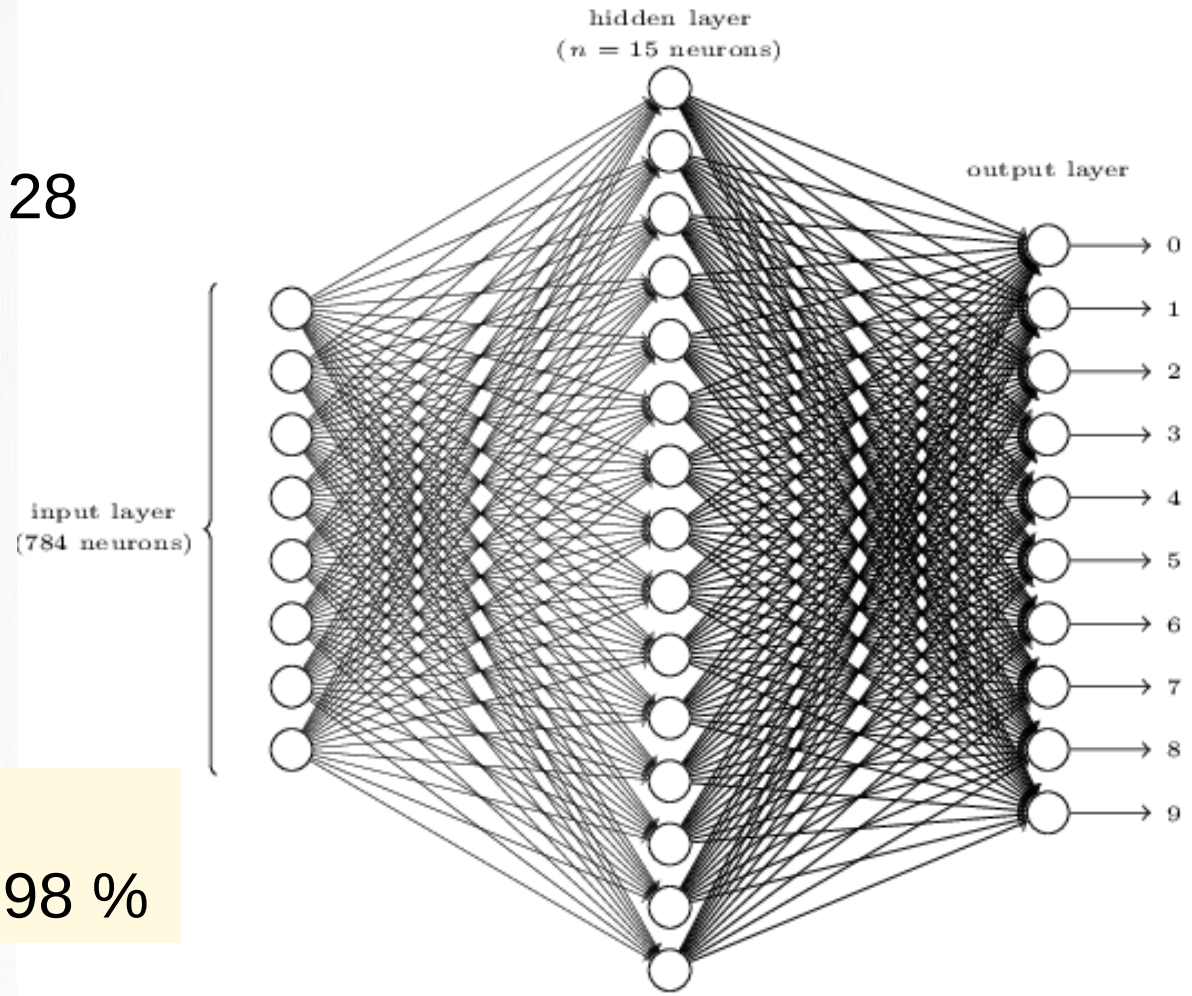
Digit classification using CNN



Digit classification - NN

Images 28 x 28 pixels, greyscale

Input 784-dimensional vector

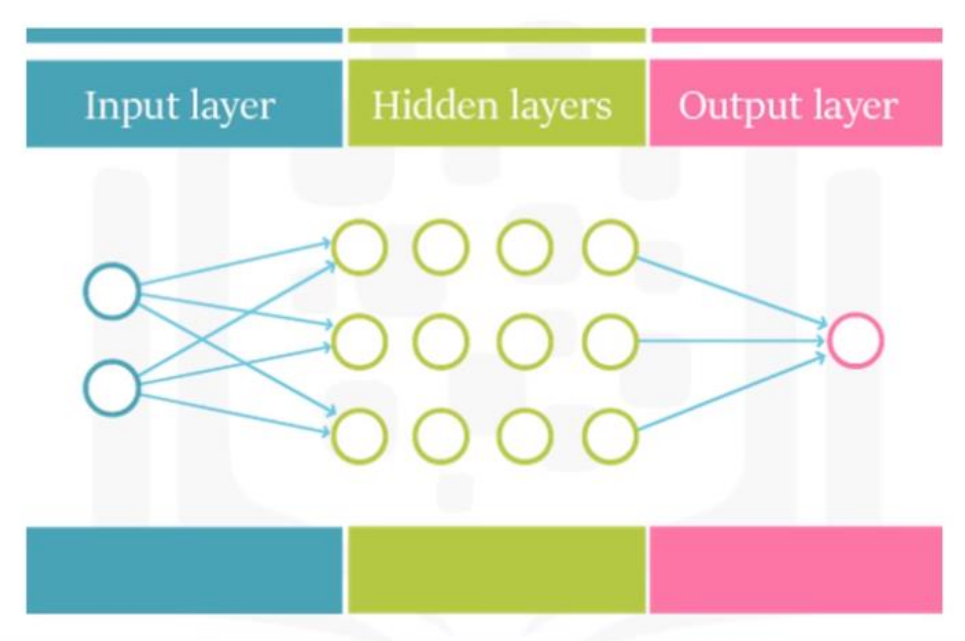


Output 10-dimensional vector,
e.g.
$y = (0,0,0,0,0,1,0,0,0,0)^T$
is 6

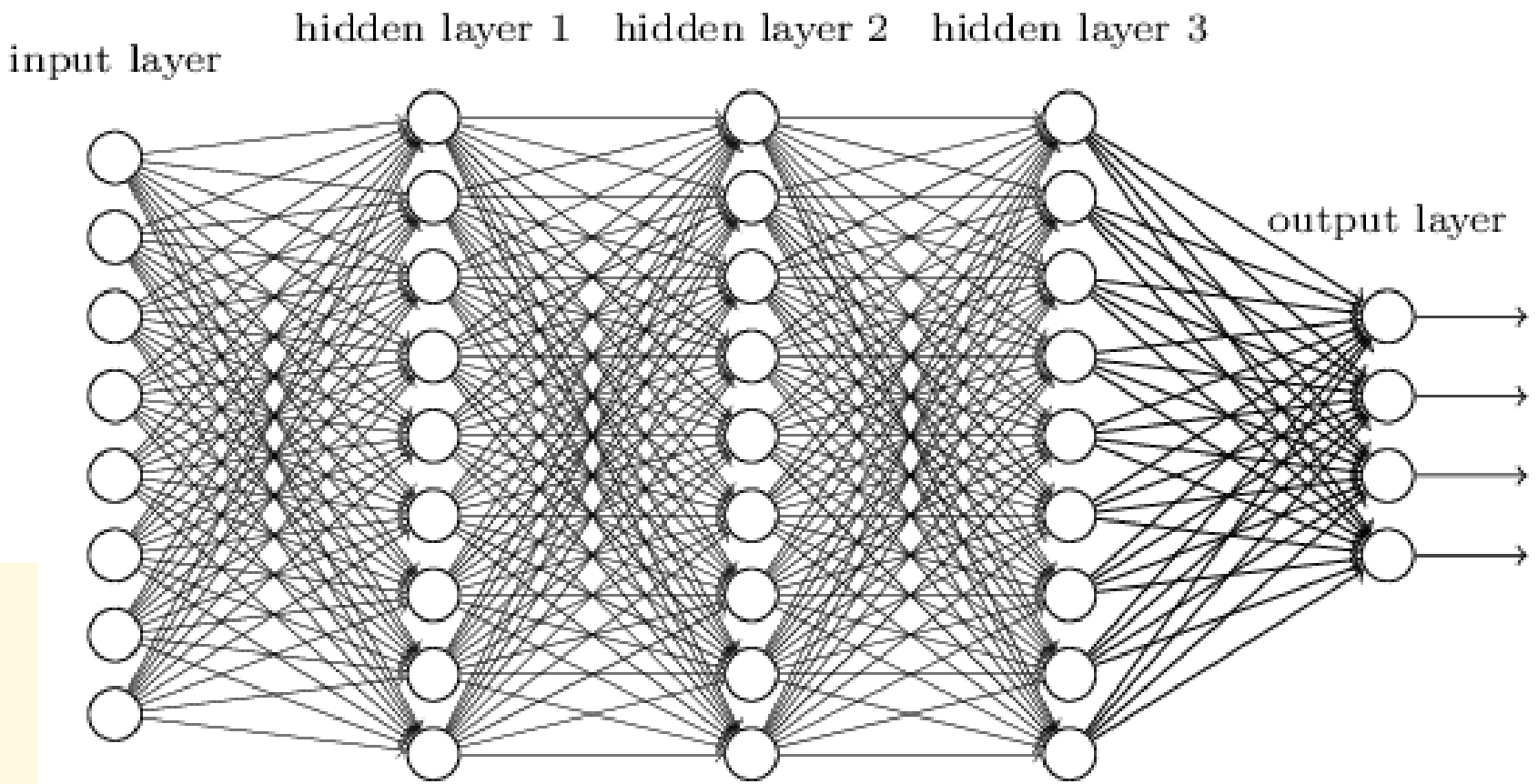
Classification accuracy around 98 %

Deep Learning

- Deep learning enables development of more complex systems and inclusion of more sophisticated rules
- It’s not very sensible to use networks with fully connected layers to classify e.g. images which have spatial structure
  - Pixels close to each other treated similarly as pixels far apart



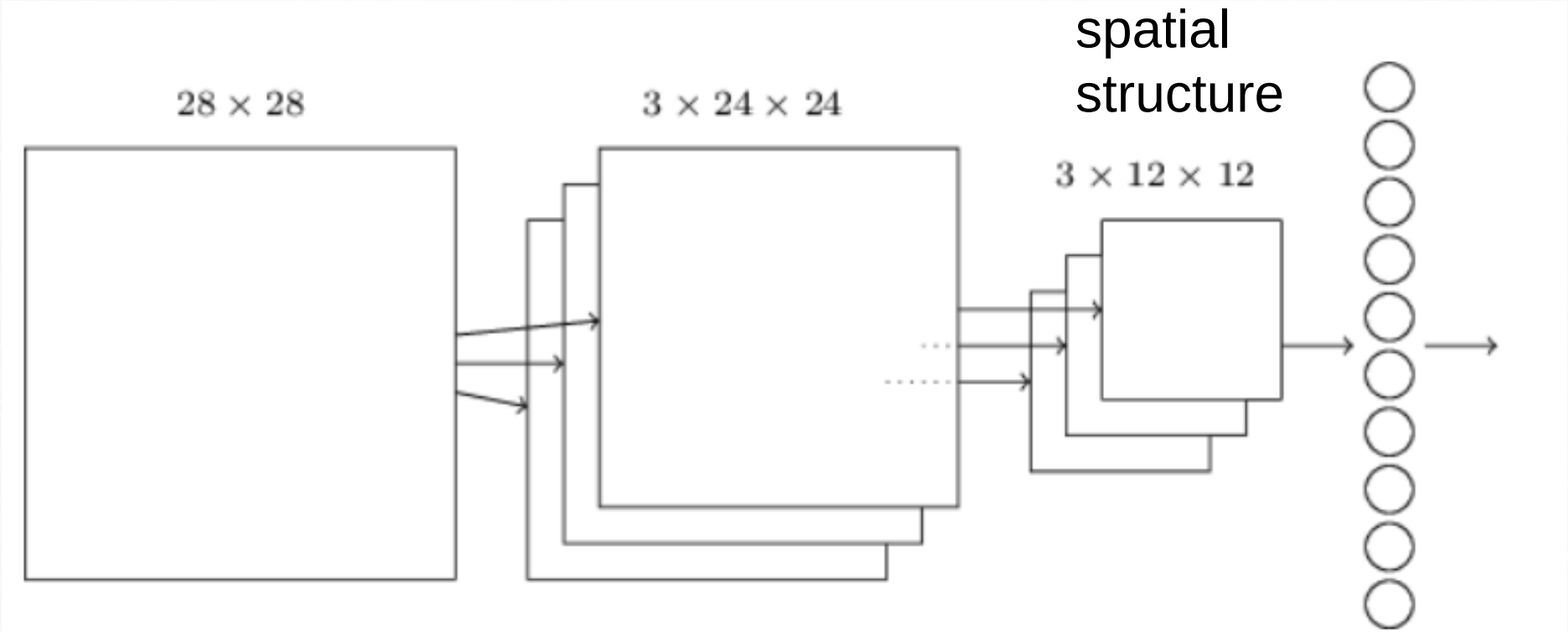
Digit classification - CNN



Classification accuracy around 99,67 %

CNN for digit example

Local spatial structure



Global information over the image

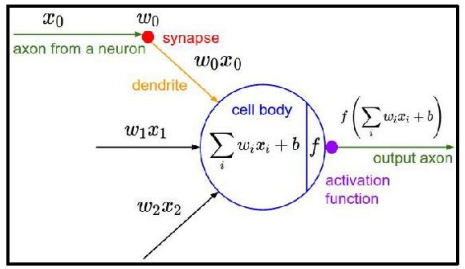
A bit of history...

The **Mark I Perceptron** machine was the first implementation of the perceptron algorithm.

The machine was connected to a camera that used 20×20 cadmium sulfide photocells to produce a 400-pixel image.

recognized
letters of the alphabet

$$f(x) = \begin{cases} 1 & \text{if } w \cdot x + b > 0 \\ 0 & \text{otherwise} \end{cases}$$



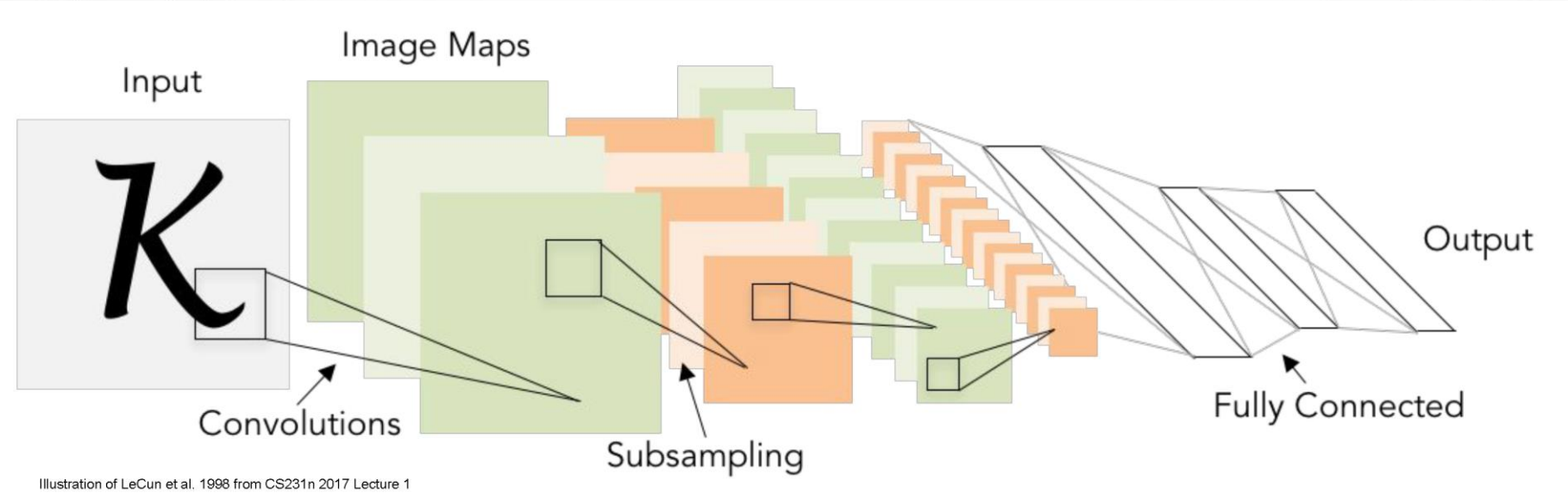
update rule:

$$w_i(t+1) = w_i(t) + \alpha(d_j - y_j(t))x_{j,i},$$

Frank Rosenblatt, ~1957: Perceptron

Slide credit Fei-Fei Li

CONVOLUTIONAL NEURAL NETWORKS



- Developed 1970, LeCun et al (1998) Gradient-based learning applied to document recognition
  - Changed the vocabulary a bit, neurons = units
  - However, all important concepts are from NNs: backpropagation, gradient descent,…

A bit of history…

[Hinton and Salakhutdinov 2006]

Reinvigorated research in Deep Learning

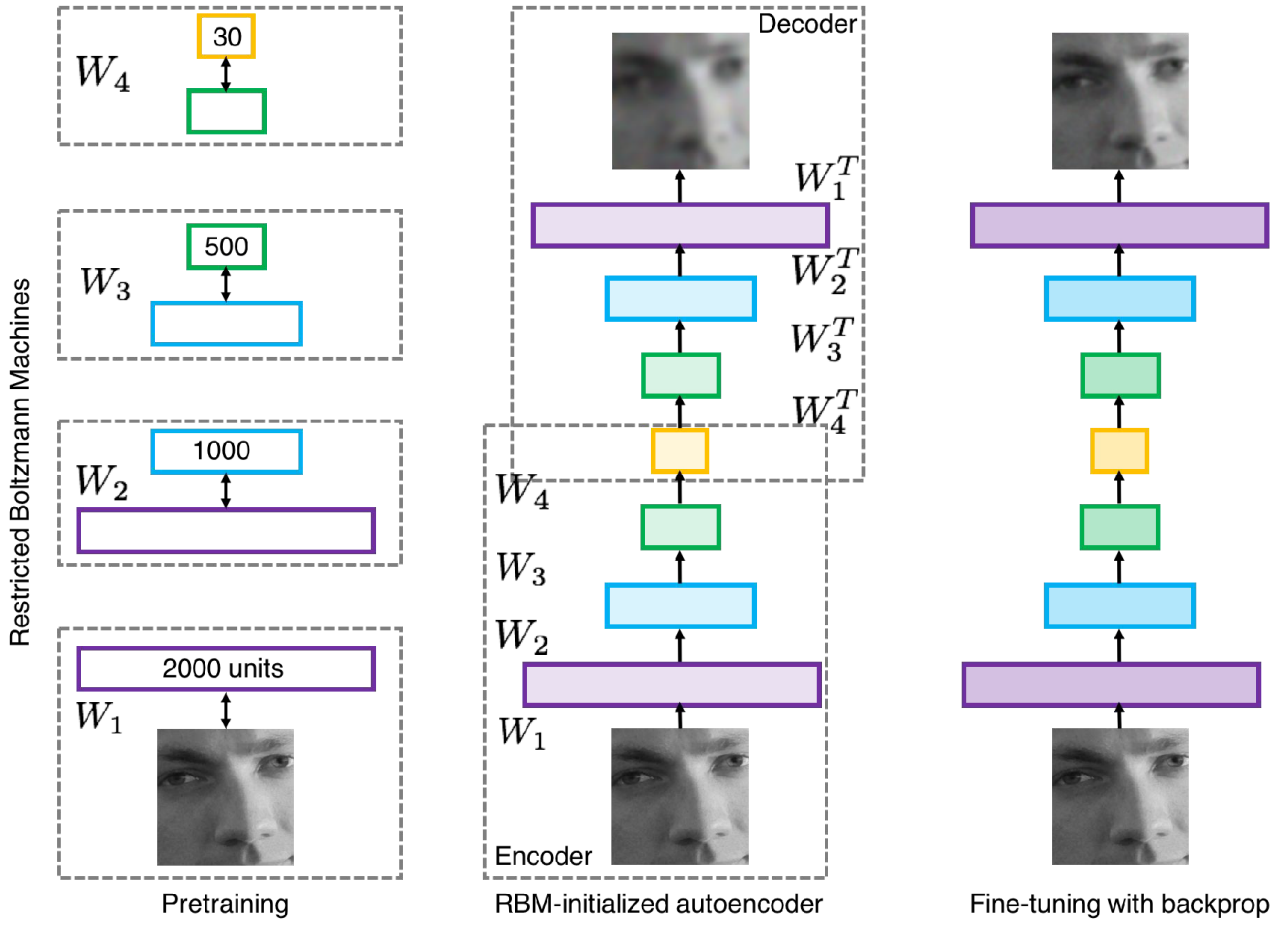


Illustration of Hinton and Salakhutdinov 2006 by Lane McIntosh, copyright CS231n 2017

Hinton and Salakhutdinov. Reducing the Dimensionality of Data with Neural Networks. *Science*, 2016.

First strong results

Acoustic Modeling using Deep Belief Networks
Abdel-rahman Mohamed, George Dahl, Geoffrey Hinton, 2010
Context-Dependent Pre-trained Deep Neural Networks for Large Vocabulary Speech Recognition
George Dahl, Dong Yu, Li Deng, Alex Acero, 2012

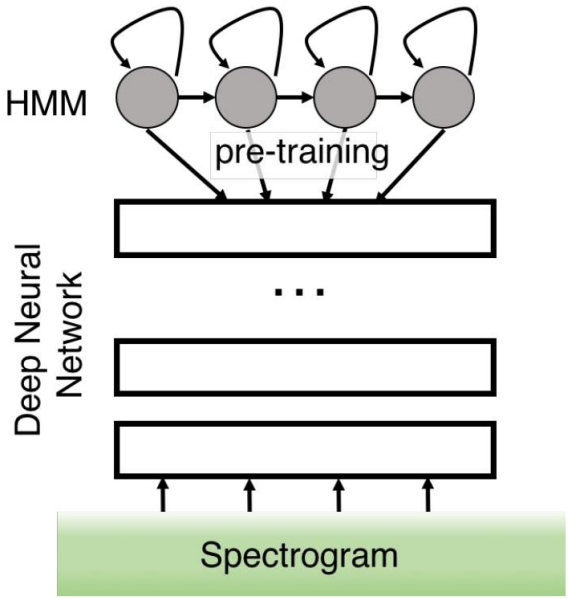
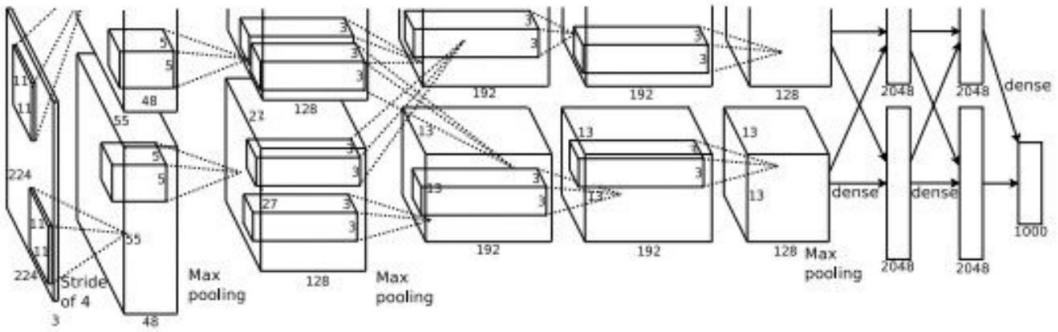
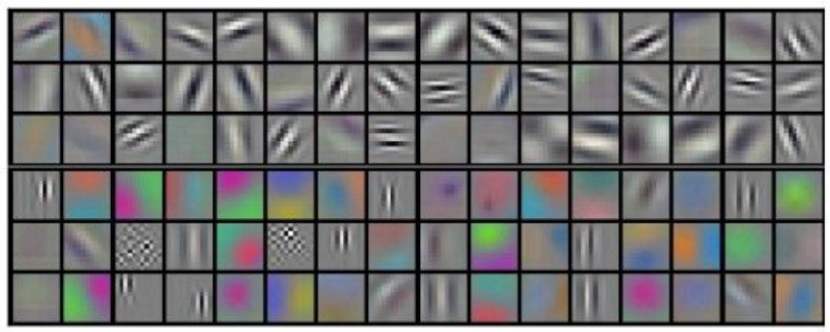


Illustration of Dahl et al. 2012 by Lane McIntosh, copyright CS231n 2017

Imagenet classification with deep convolutional neural networks
Alex Krizhevsky, Ilya Sutskever, Geoffrey E Hinton, 2012





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

Slide credit Fei-Fei Li

A bit of history:

ImageNet Classification with Deep Convolutional Neural Networks

[Krizhevsky, Sutskever, Hinton, 2012]

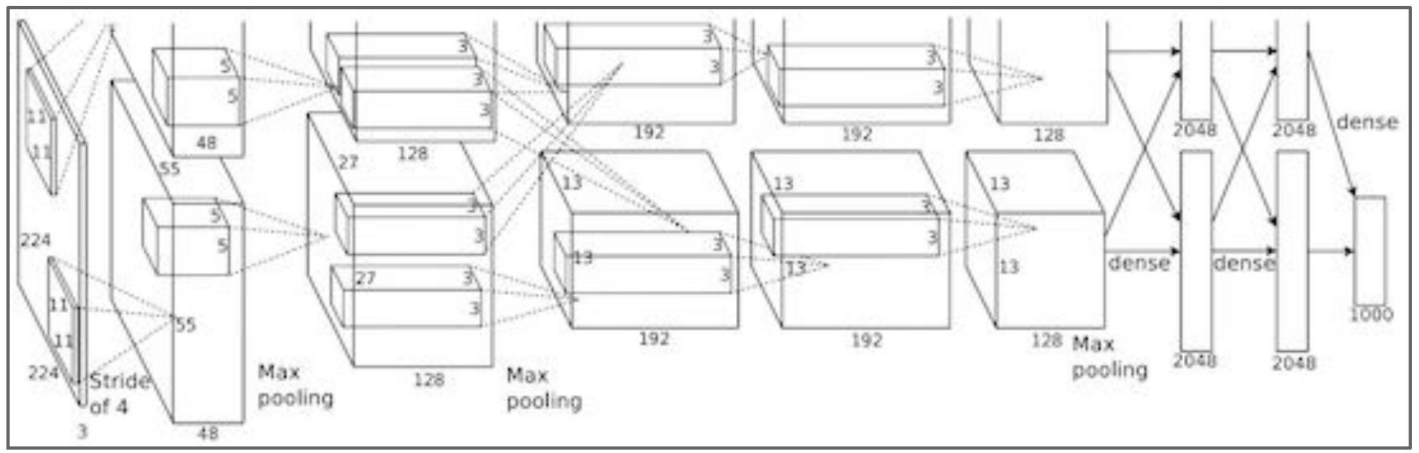


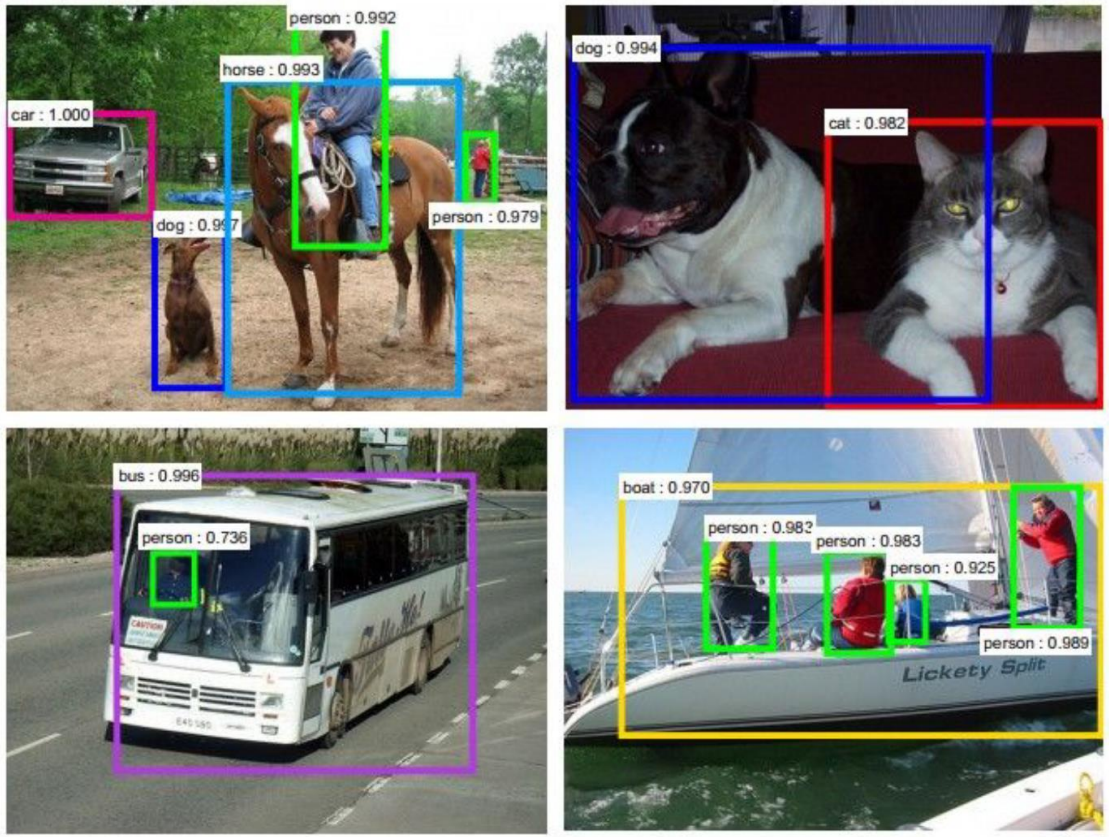
Figure copyright Alex Krizhevsky, Ilya Sutskever, and Geoffrey Hinton, 2012. Reproduced with permission.

"AlexNet"

Slide credit Fei-Fei Li

Fast-forward to today: ConvNets are everywhere

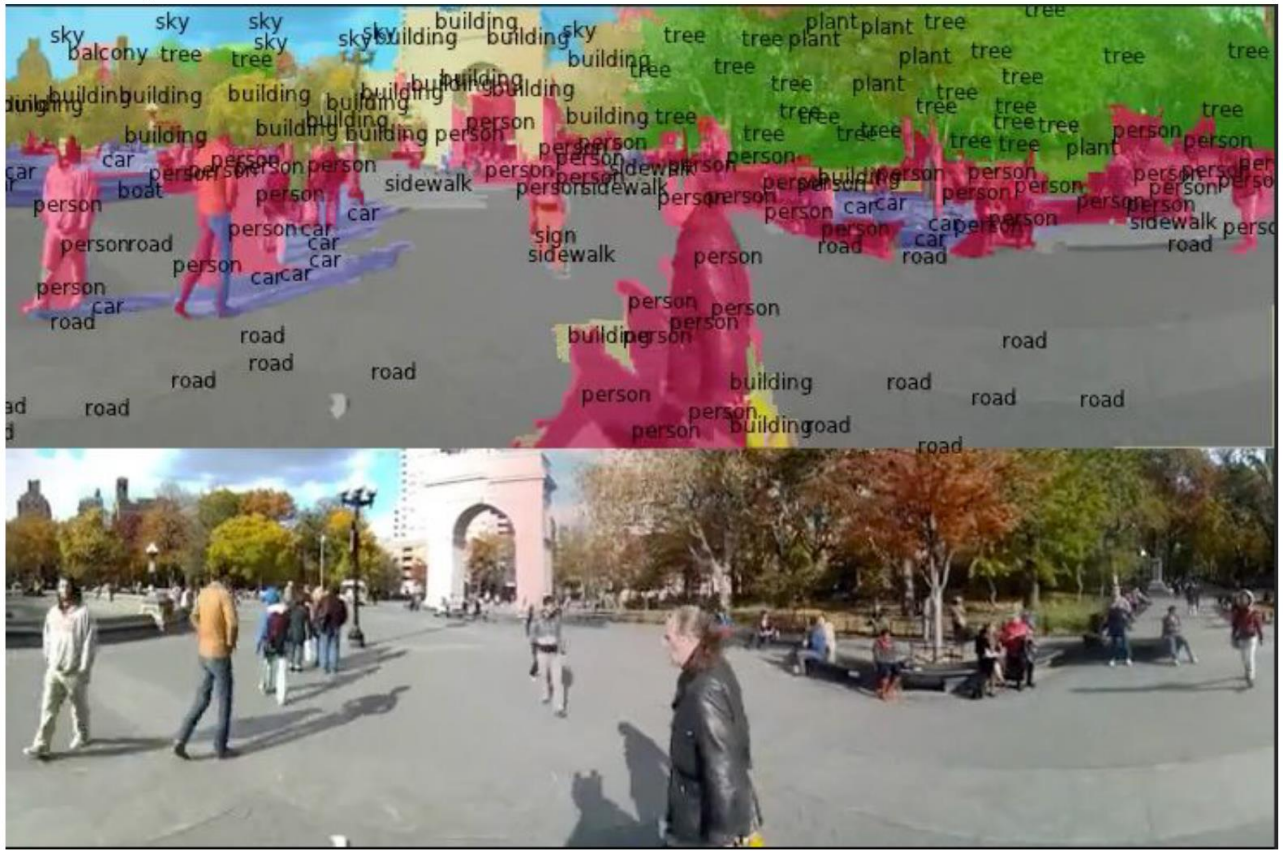
Detection



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]

Slide credit Fei-Fei Li

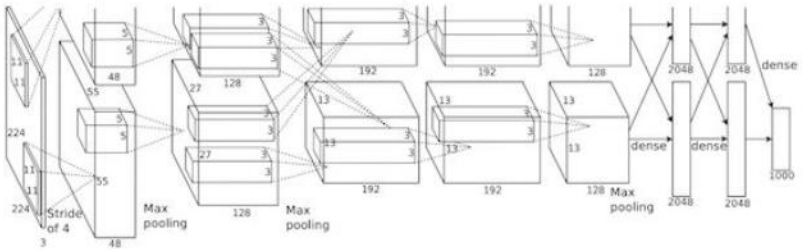
Image features vs ConvNets

Feature Extraction

f

10 numbers giving scores for classes

training



Krizhevsky, Sutskever, and Hinton, "Imagenet classification with deep convolutional neural networks", NIPS 2012. Figure copyright Krizhevsky, Sutskever, and Hinton, 2012. Reproduced with permission.

10 numbers giving scores for classes

training

Slide credit Fei-Fei Li

THE LAST LAYER OF (MOST) CNNS ARE LINEAR CLASSIFIERS

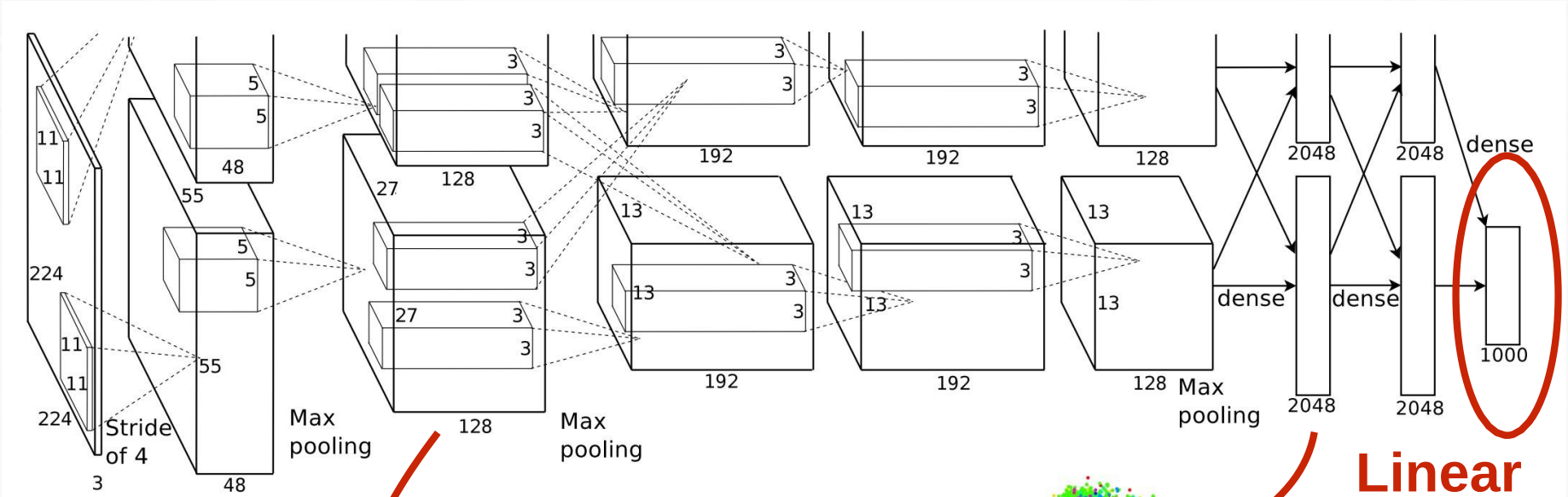
Input Pixels

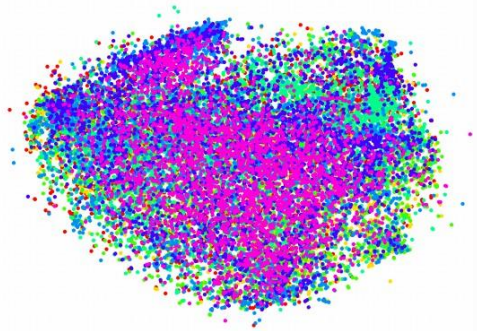
Perform everything with a big neural network, trained end-to-end

Key: perform enough processing so that by the time you get to the end of the network, the classes are linearly separable

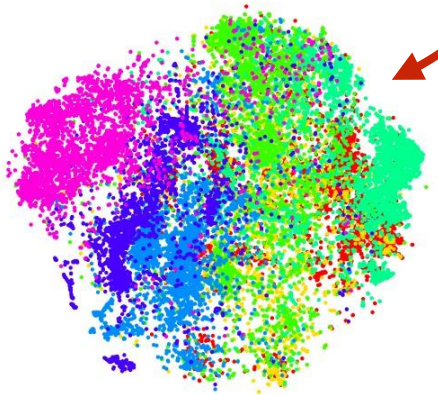
Slide credit Fei-Fei Li

EXAMPLE: VISUALIZING ALEXNET IN 2D WITH T-SNE





(c) DeCAF_1

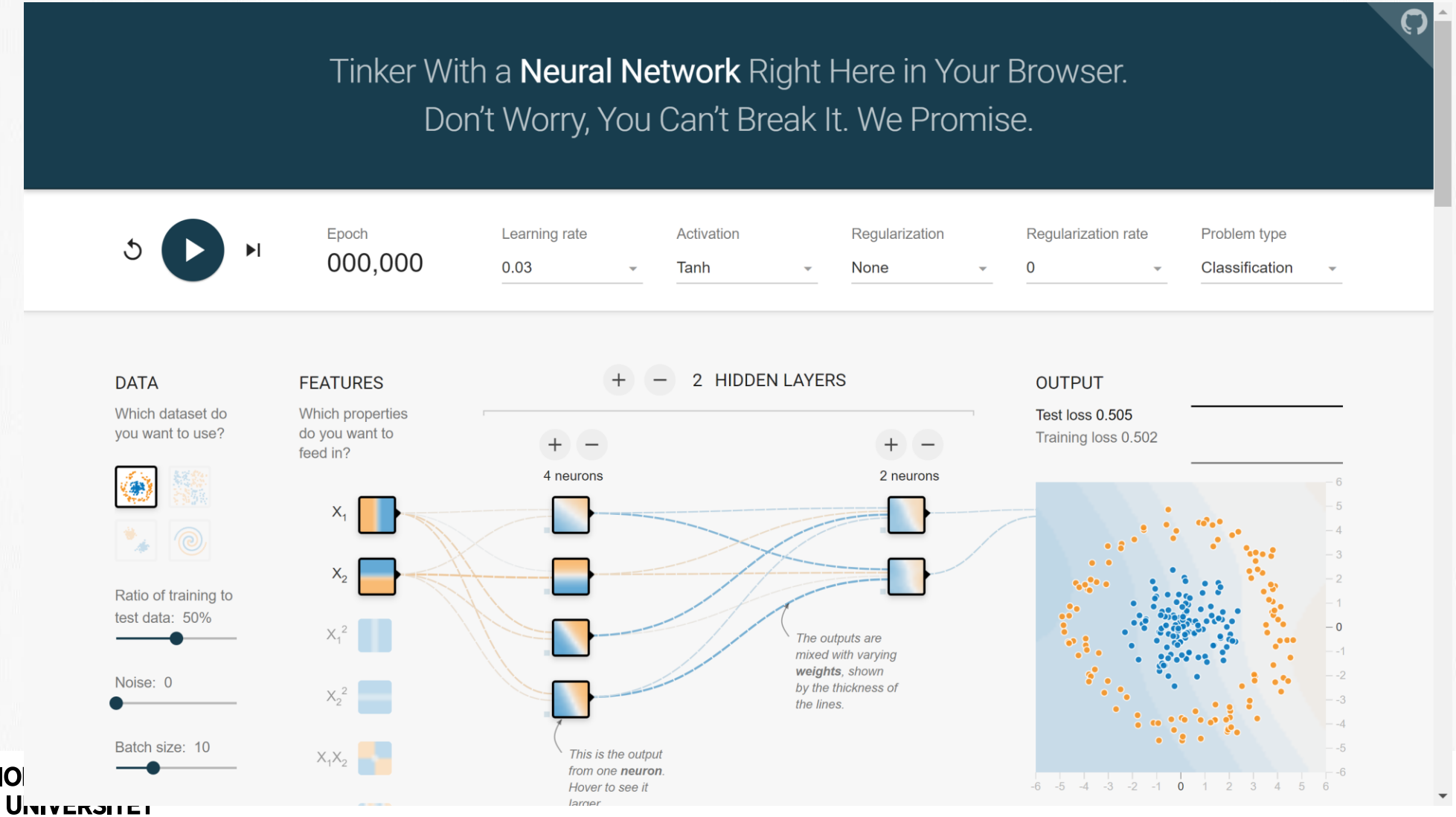


(d) DeCAF_6

(t-SNE visualization using t-SNE)

[Donahue, "DeCAF: DeCAF: A Deep Convolutional …", arXiv 2013]

2D EXAMPLE: TENSORFLOW PLAYGROUND



HELSINGIN YLIO
HELSINGFORS U
UNIVERSITY OF HELSINKI

https://playground.tensorflow.org

CONVOLUTIONAL NEURAL NETWORKS

- A convolutional neural network can be thought of as a function from images to class scores

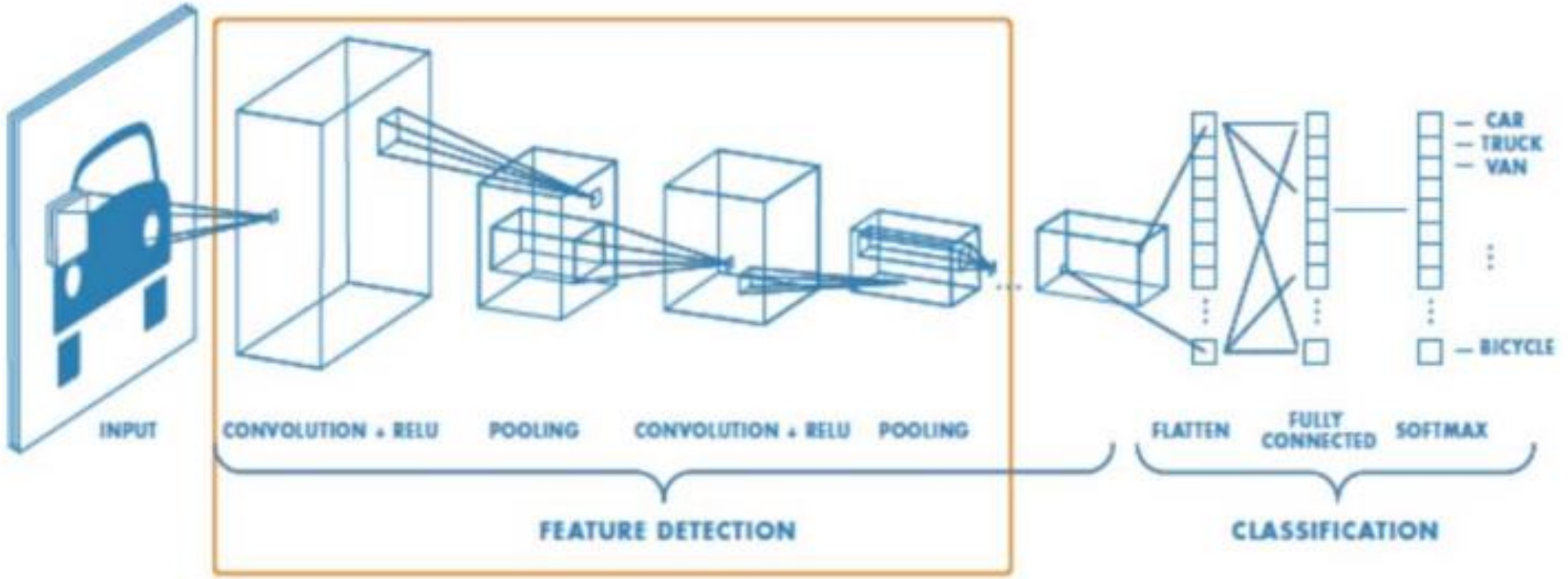
  With millions of adjustable weights…

  … leading to a very non-linear mapping from images to features / class scores.

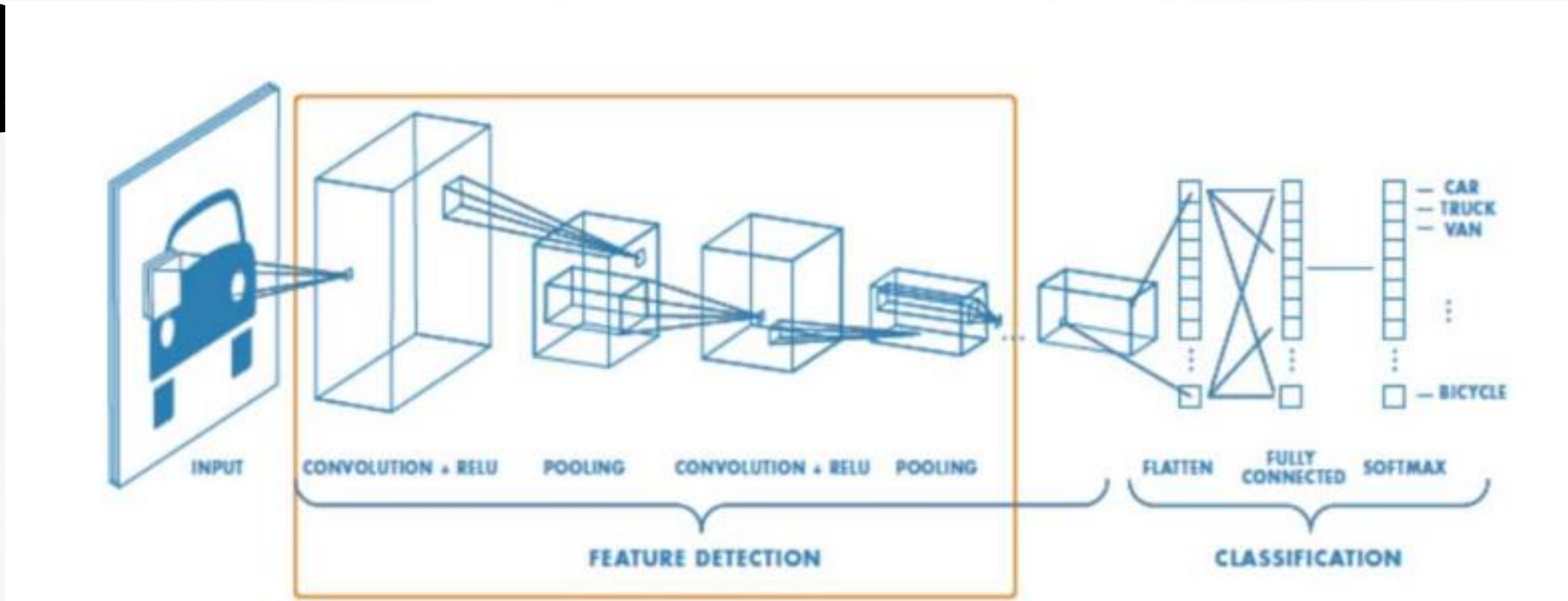
  We will set these weights based on classification accuracy on training data…

  … and hopefully our network will generalize to new images at test time

- Layer types:
  - Fully-connected layer
  - *Convolutional layer*
  - Pooling layer



Vasuki and Govindaraju (2017). Deep Neural Networks for Image Classification. Deep Learning for Image Processing Applications, IOS Press.



INPUT
32x32x3
raw pixels of an image

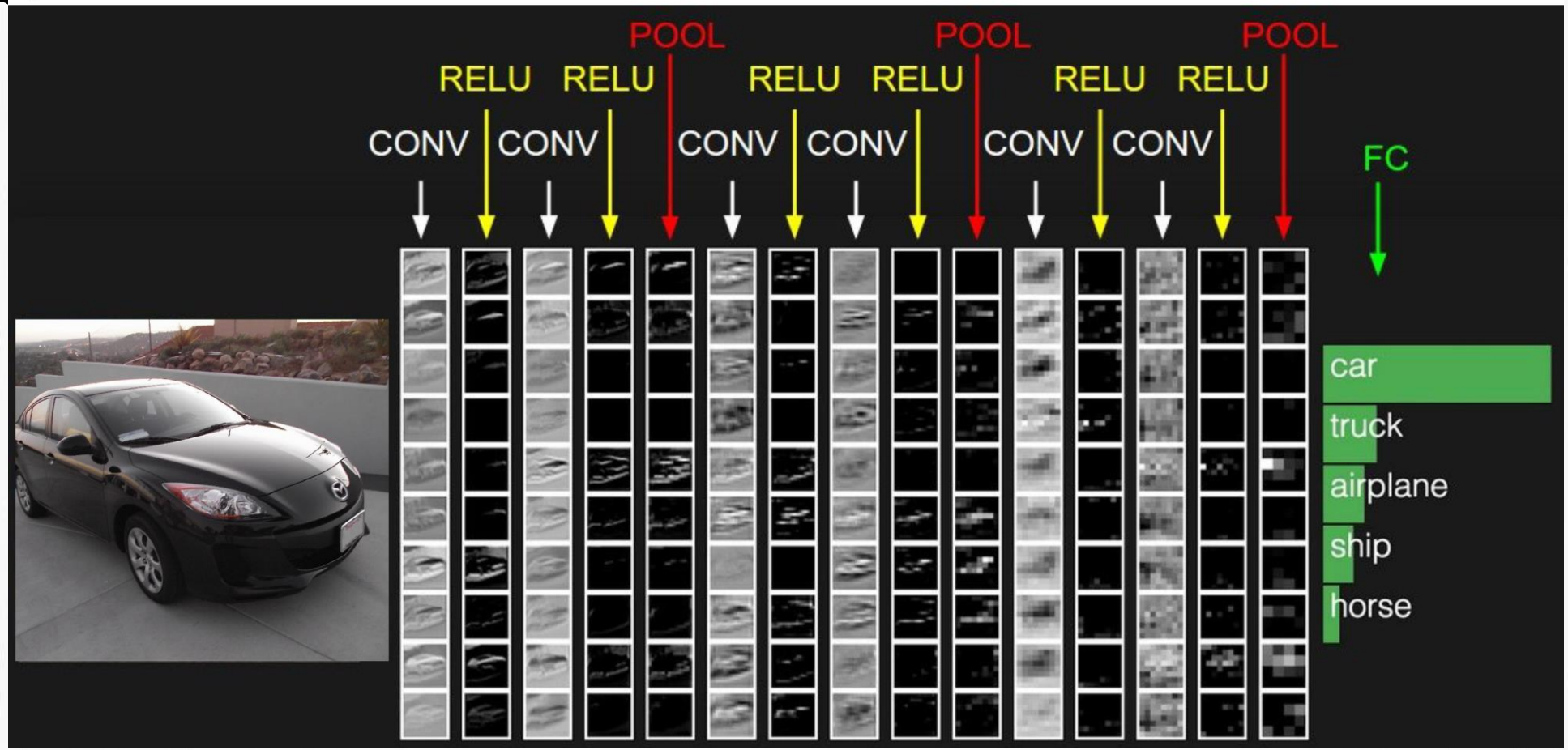
CONV layer
32x32x12
if 12 filters

RELU layer
Activation function
32x32x12

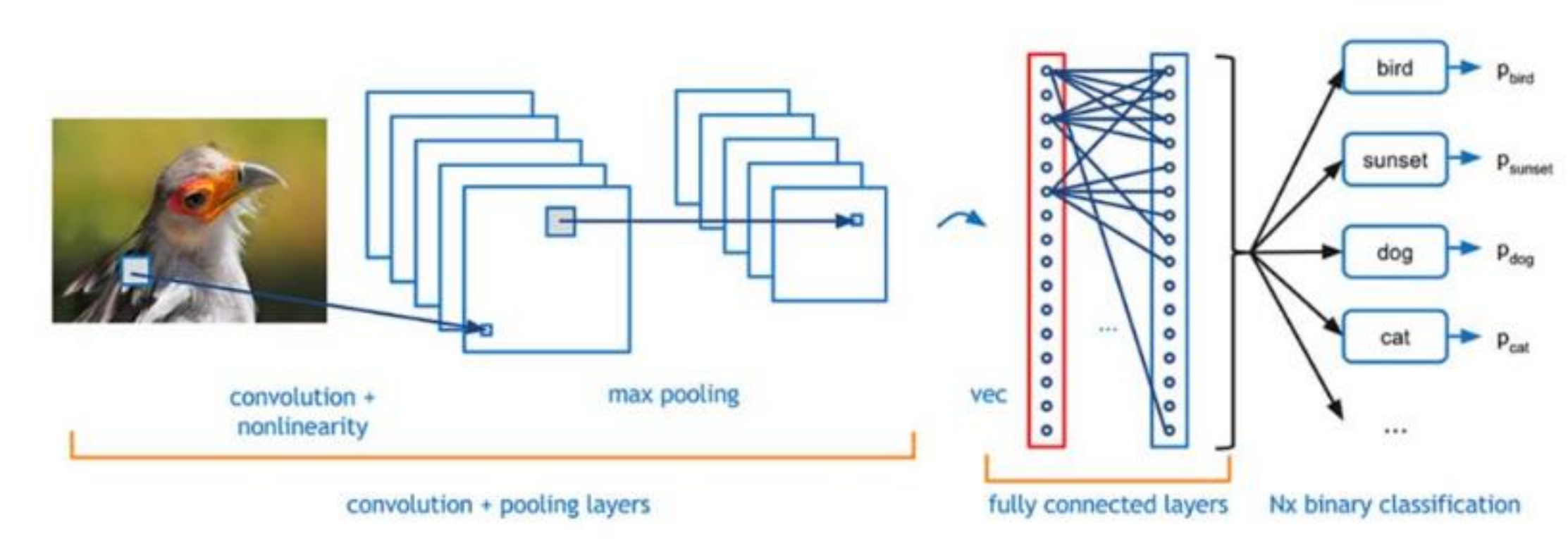
POOL layer
Down sampling
16x16x12

Fully-connected layer
Class scores
1x1x10

preview:

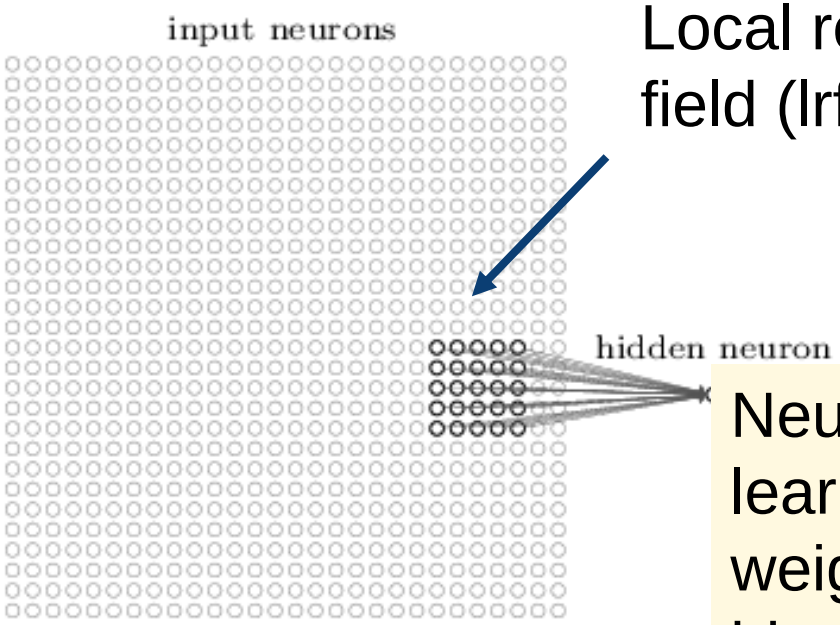


Slide credit Fei-Fei Li


bird
sunset
dog
cat
...
p_{bird}
p_{sunset}
p_{dog}
p_{cat}
convolution + nonlinearity
max pooling
vec
convolution + pooling layers
fully connected layers
Nx binary classification

Local receptive fields

- Earlier in the digit example, input was vertical line of neurons, now 28x28 square of neurons (pixel intensities)
- Now, not every pixel is connected to every hidden neuron => connections in localized small regions of image

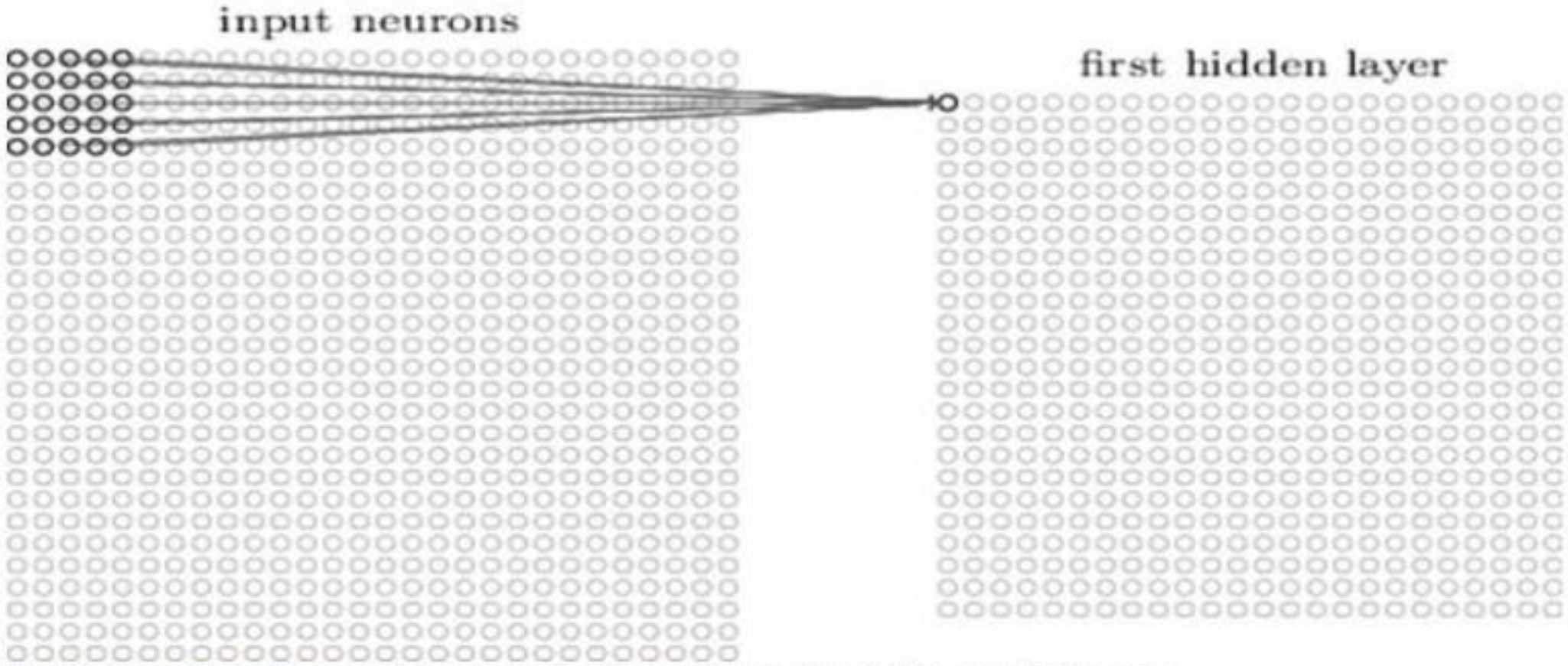


Local receptive field (lrf)

Neuron learns a weight and a bias

The lrf is slid ahead one pixel per time => 5x5 lrf => 24x24 neurons

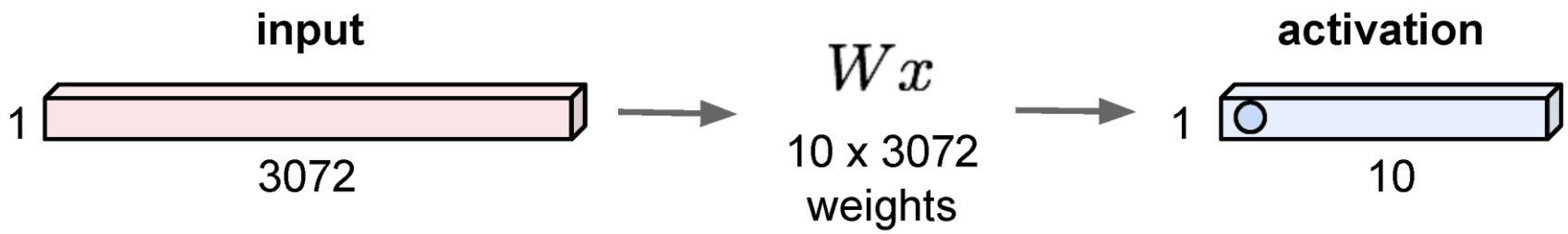
stride length, can be larger



Visualization of 5 x 5 filter convolving around an input volume and producing an activation map

Fully Connected Layer

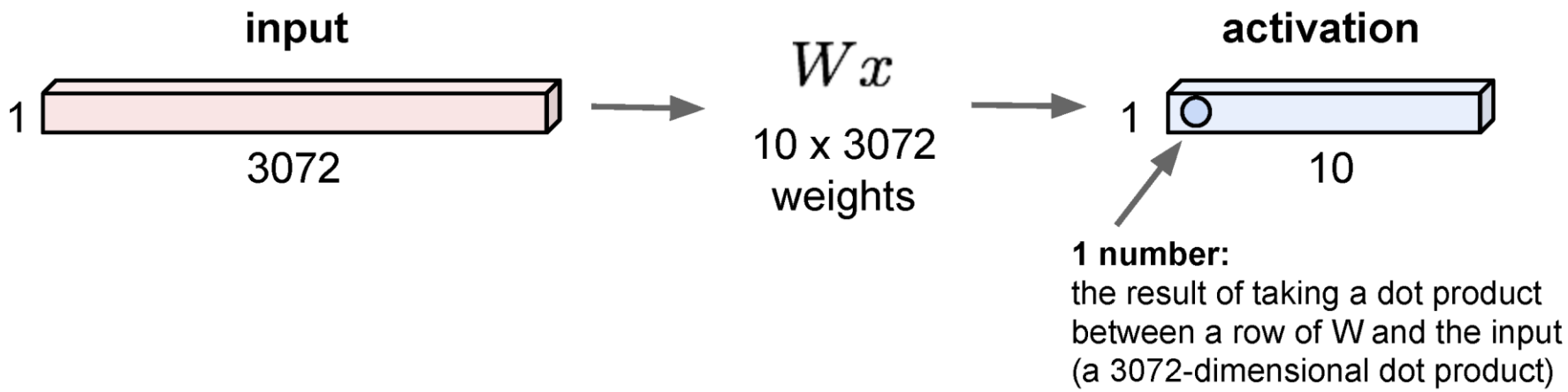
32x32x3 image -> stretch to 3072 x 1



Slide credit Fei-Fei Li

Fully Connected Layer

32x32x3 image -> stretch to 3072 x 1

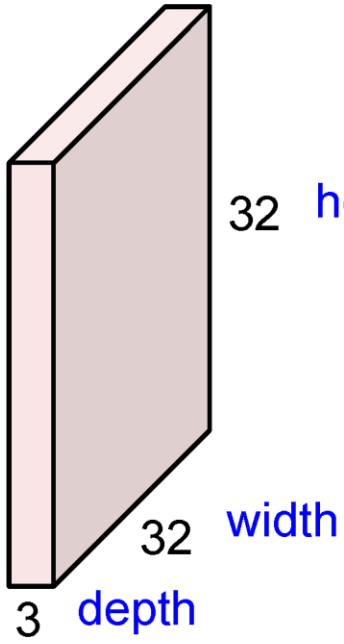


Same as a linear classifer!

Slide credit Fei-Fei Li

Convolution Layer

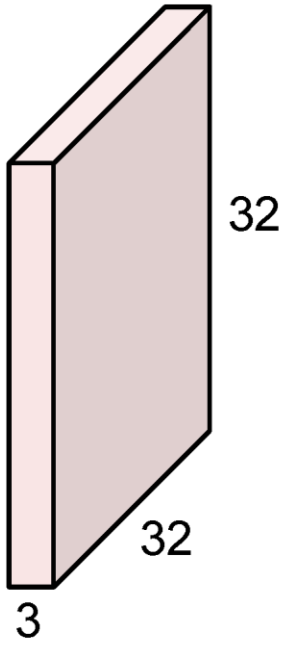
32x32x3 image -> preserve spatial structure



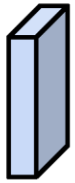
Slide credit Fei-Fei Li

Convolution Layer

32x32x3 image



5x5x3 filter



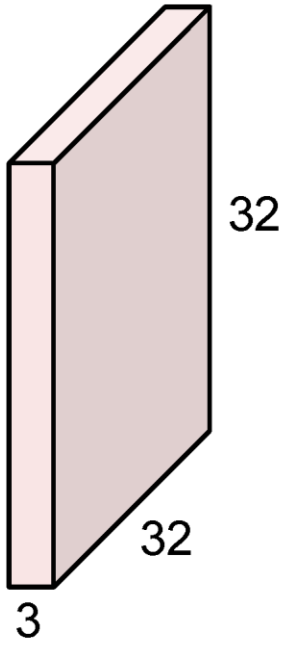
Convolve the filter with the image
i.e. “slide over the image spatially,
computing dot products”

Slide credit Fei-Fei Li

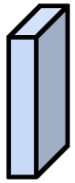
Convolution Layer

Filters always extend the full depth of the input volume

32x32x3 image



5x5x3 filter

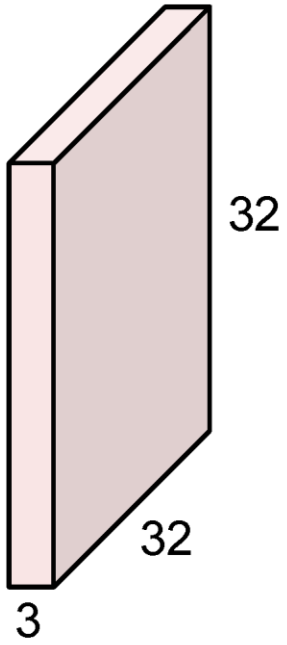


Convolve the filter with the image i.e. “slide over the image spatially, computing dot products”

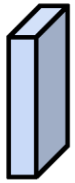
Slide credit Fei-Fei Li

Convolution Layer

32x32x3 image



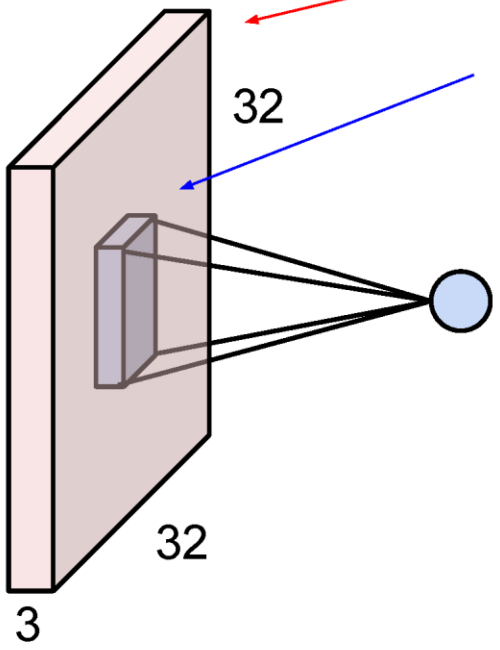
5x5x3 filter



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

Number of weights: 5 x 5 x 3 + 1 = **76**
(vs. 3072 for a fully-connected layer)
(+1 for bias)

Convolution Layer



32x32x3 image

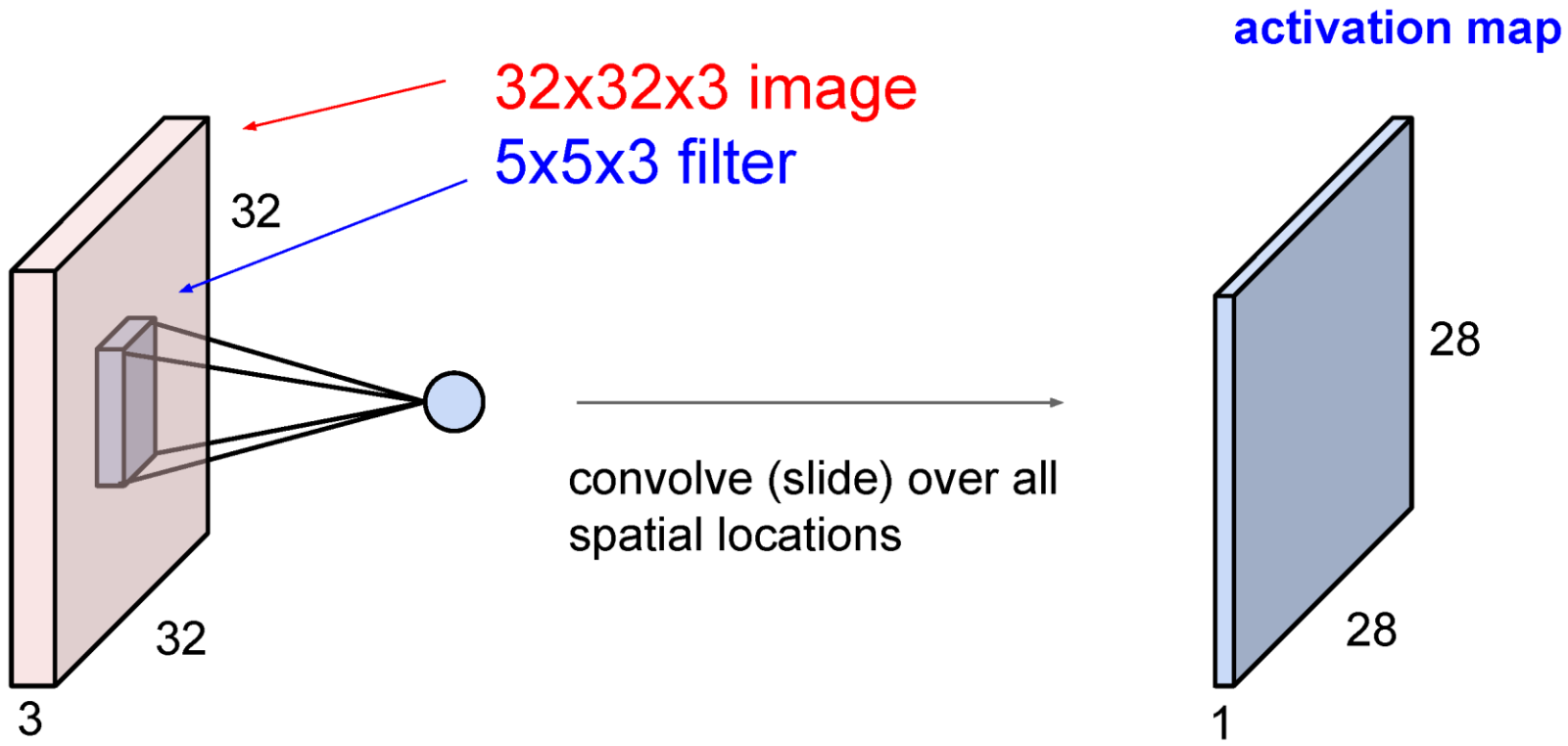
5x5x3 filter w

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)

$$w^T x + b$$

Slide credit Fei-Fei Li

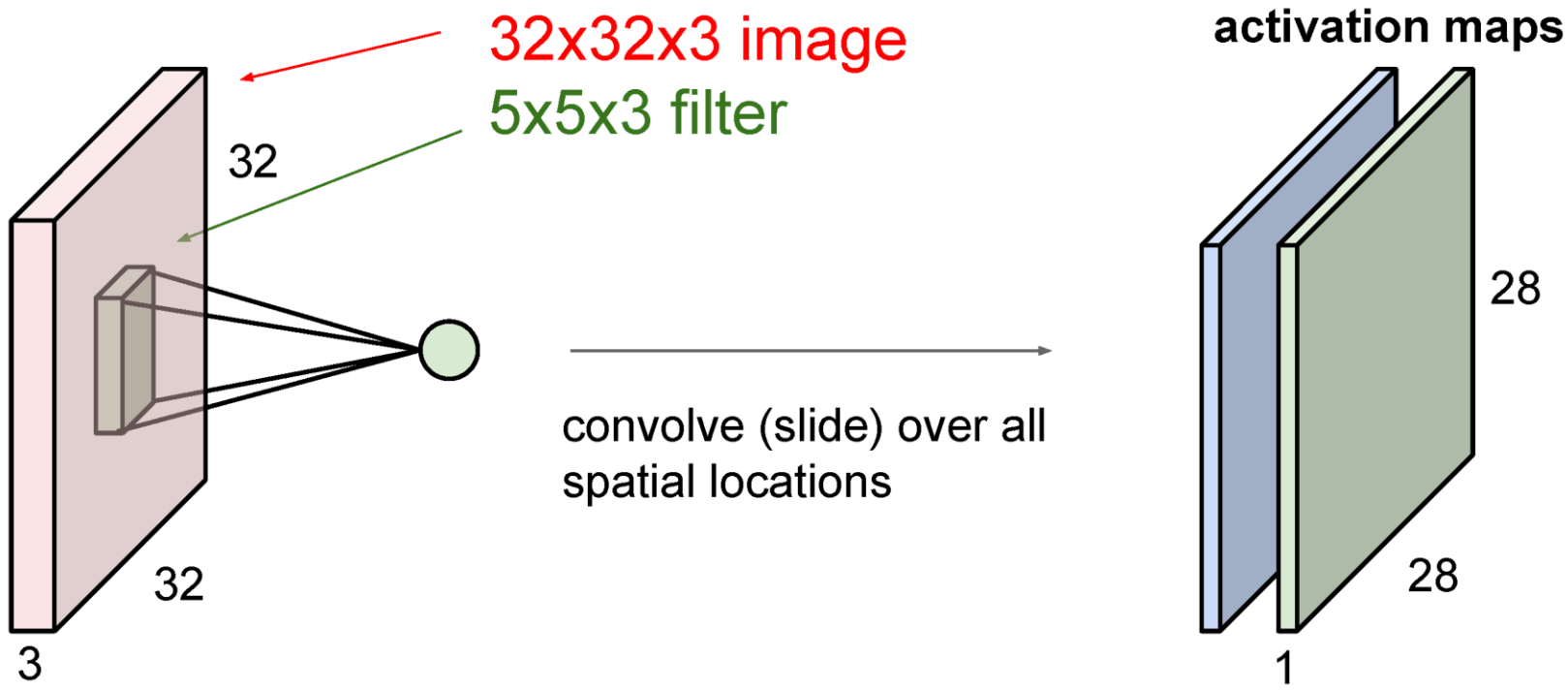
Convolution Layer



Slide credit Fei-Fei Li

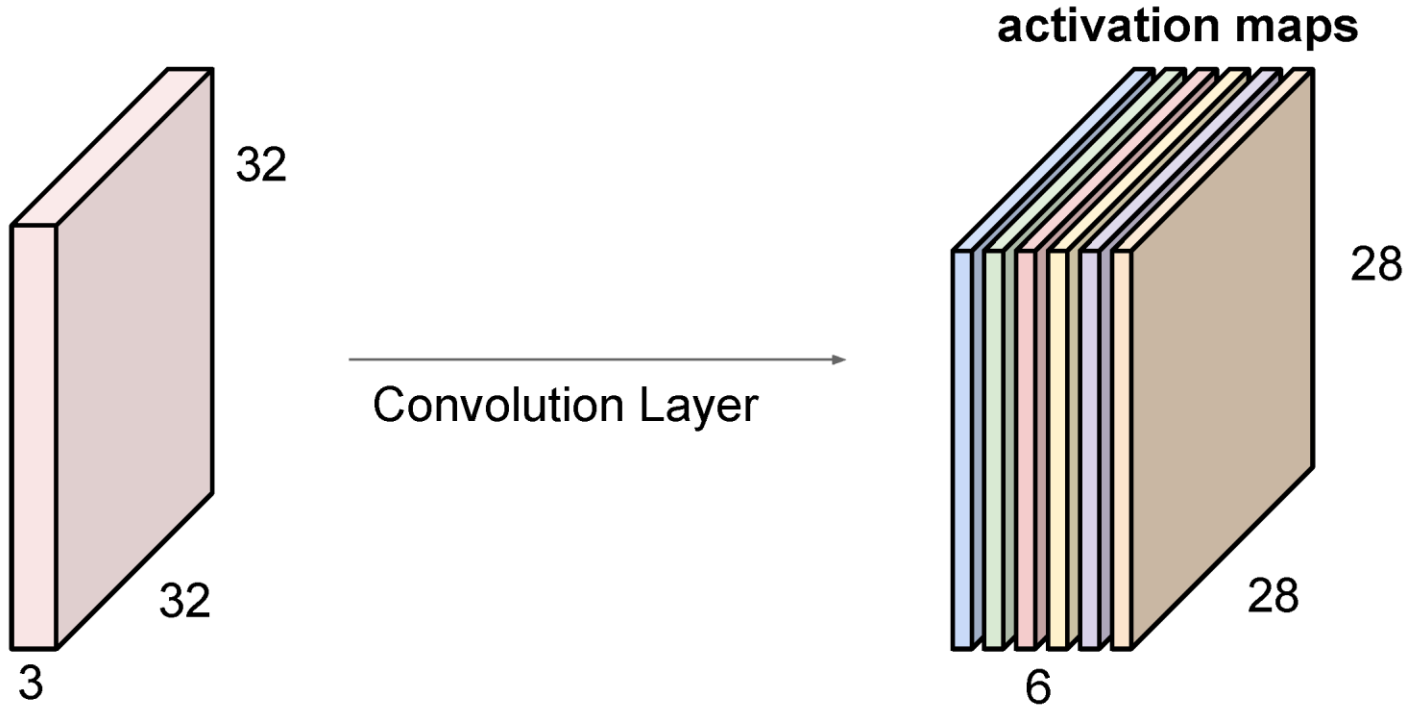
Convolution Layer

consider a second, green filter



Slide credit Fei-Fei Li

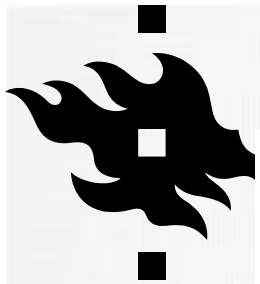
For example, if we had 6 5x5 filters, we'll get 6 separate activation maps:



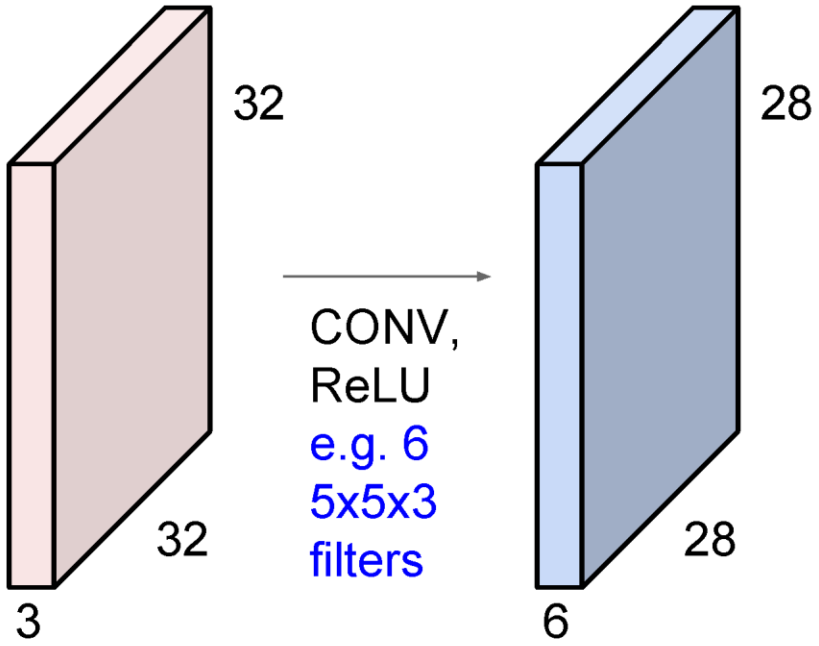
We stack these up to get a "new image" of size 28x28x6!

(total number of parameters: 6 x (75 + 1) = **456**)

Slide credit Fei-Fei Li

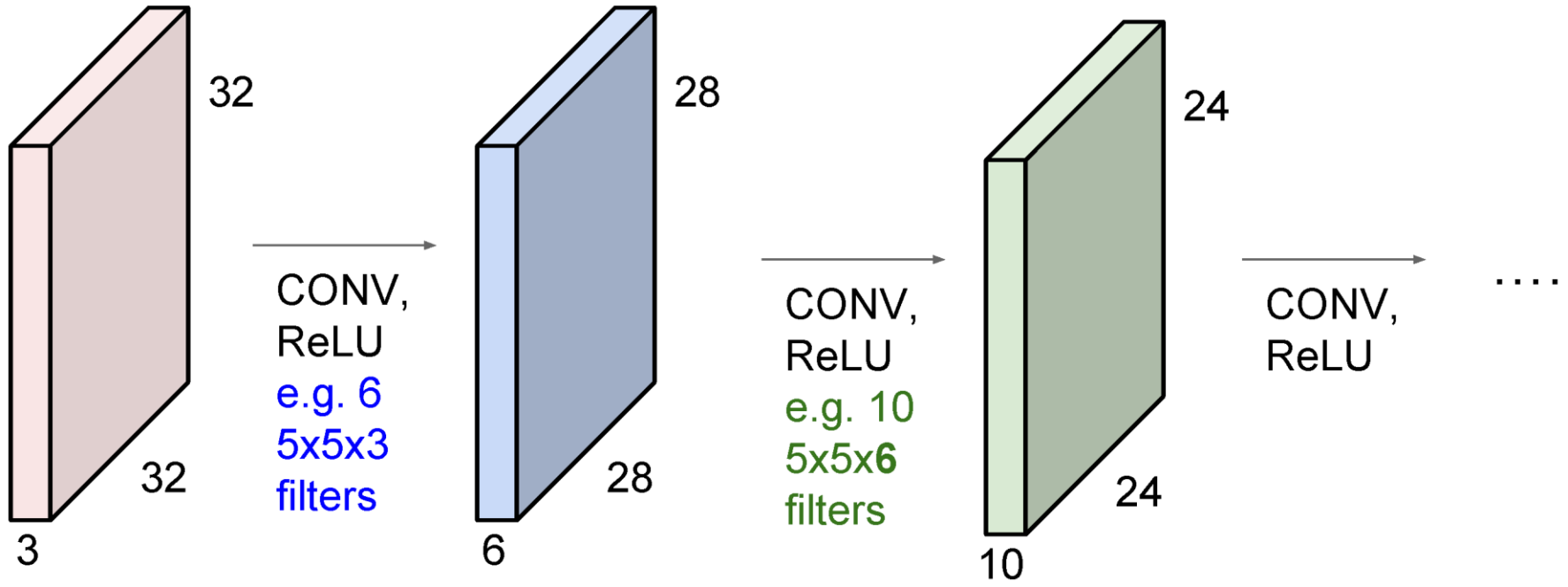


Preview: ConvNet is a sequence of Convolution Layers, interspersed with activation functions

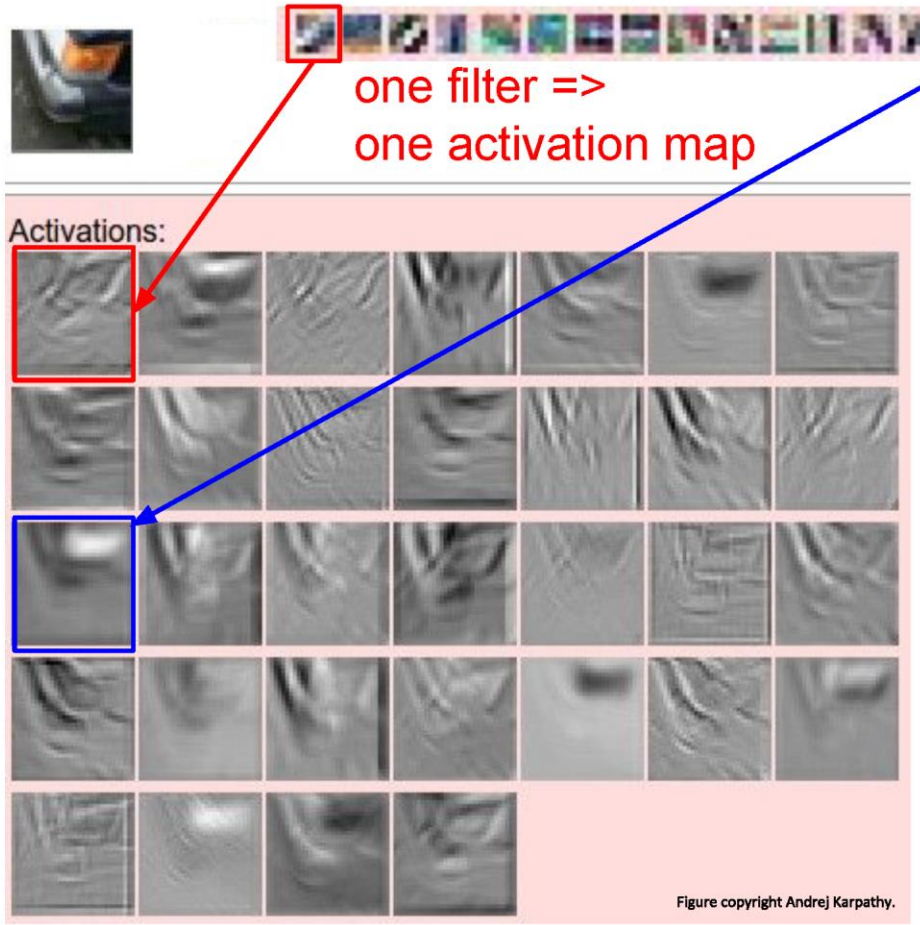


Slide credit Fei-Fei Li

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



Slide credit Fei-Fei Li



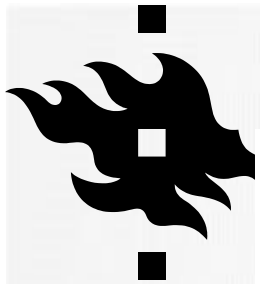
example 5x5 filters
(32 total)

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

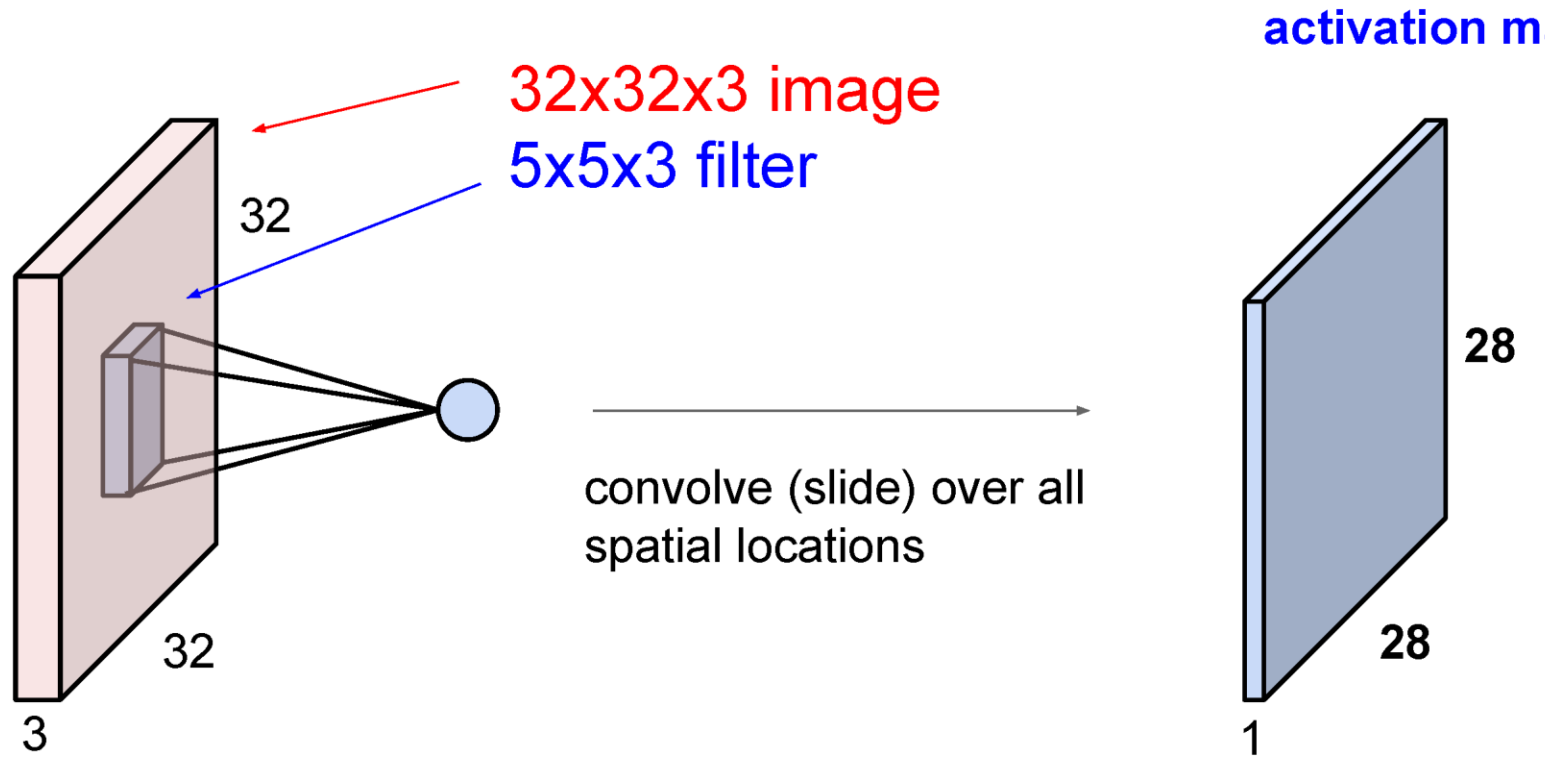
$$f[x,y] * g[x,y] = \sum_{n_1=-\infty}^{\infty} \sum_{n_2=-\infty}^{\infty} f[n_1,n_2] \cdot g[x-n_1, y-n_2]$$

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

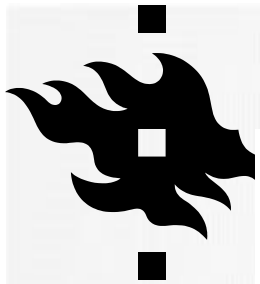
Slide credit Fei-Fei Li



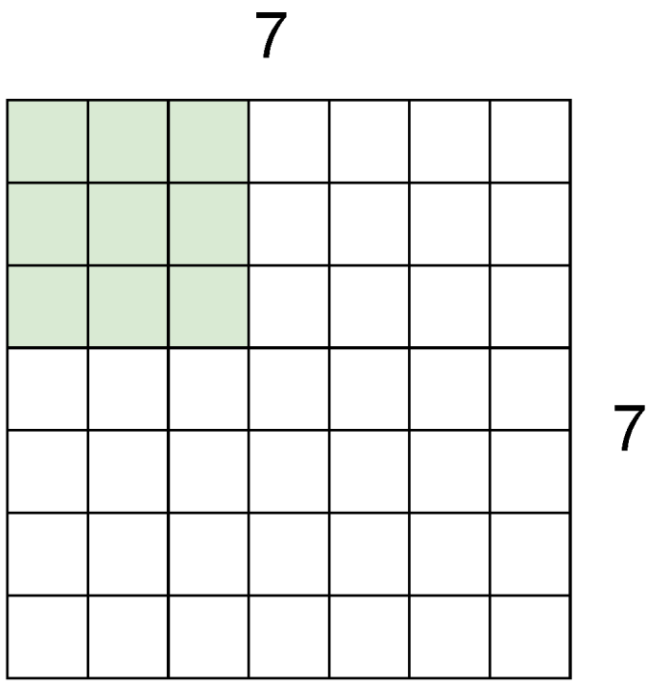
A closer look at spatial dimensions:



Slide credit Fei-Fei Li



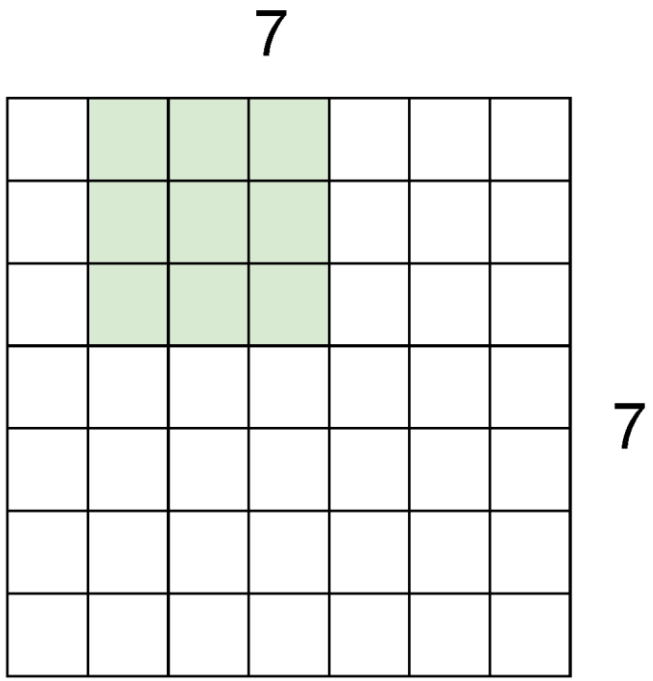
A closer look at spatial dimensions:



7x7 input (spatially)
assume 3x3 filter

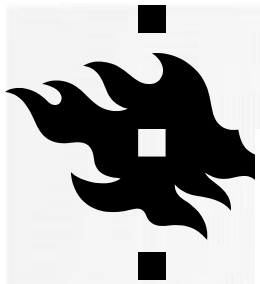
Slide credit Fei-Fei Li

A closer look at spatial dimensions:

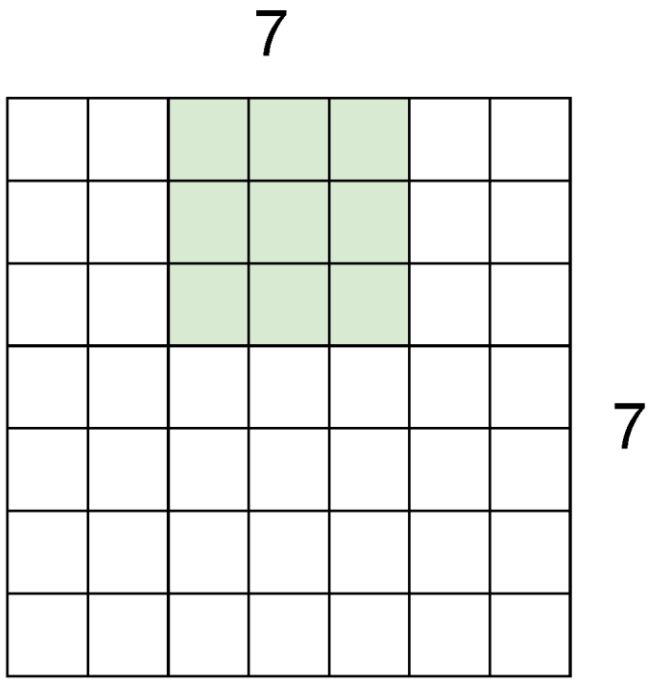


7x7 input (spatially)
assume 3x3 filter

Slide credit Fei-Fei Li



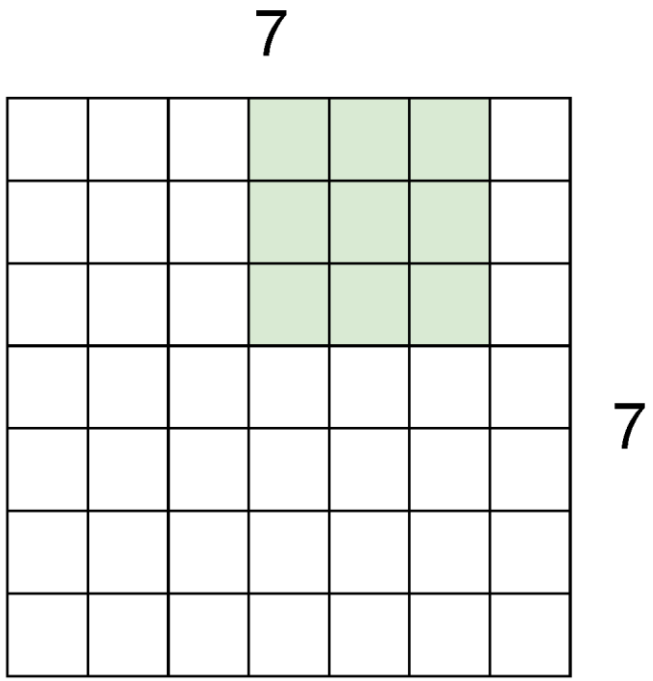
A closer look at spatial dimensions:



7x7 input (spatially)
assume 3x3 filter

Slide credit Fei-Fei Li

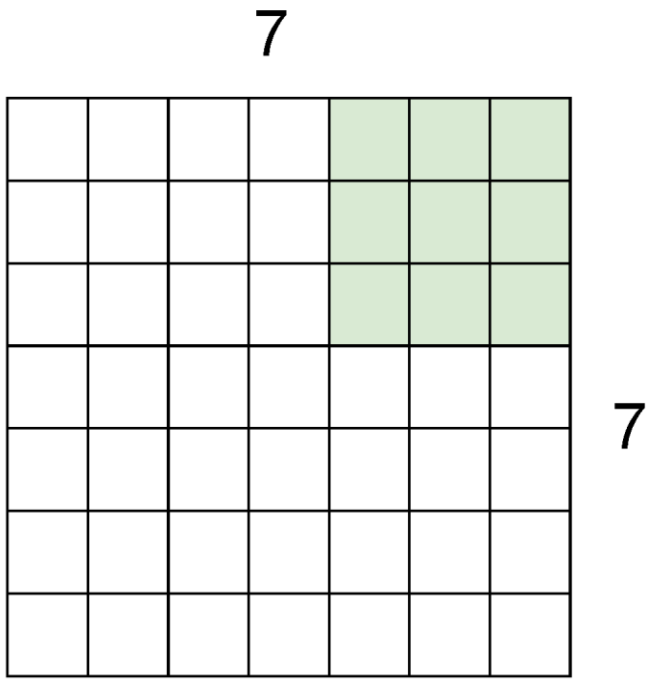
A closer look at spatial dimensions:



7x7 input (spatially)
assume 3x3 filter

Slide credit Fei-Fei Li

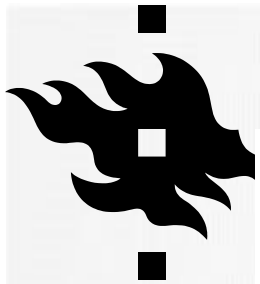
A closer look at spatial dimensions:



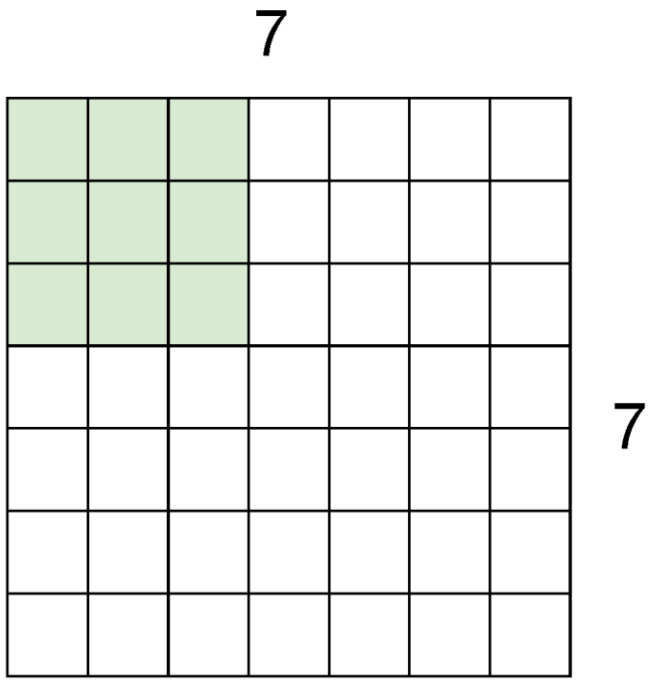
7x7 input (spatially)
assume 3x3 filter

=> 5x5 output

Slide credit Fei-Fei Li



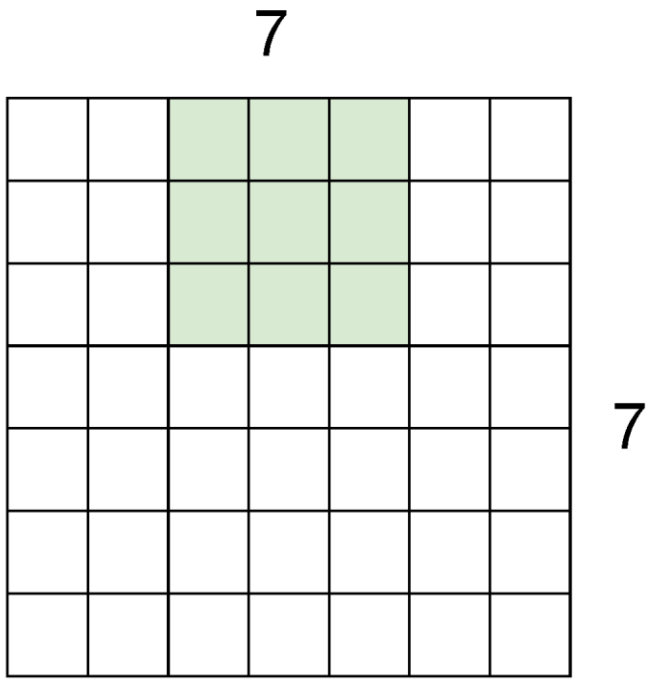
A closer look at spatial dimensions:



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

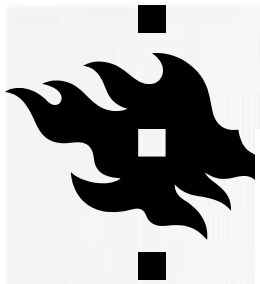
Slide credit Fei-Fei Li

A closer look at spatial dimensions:

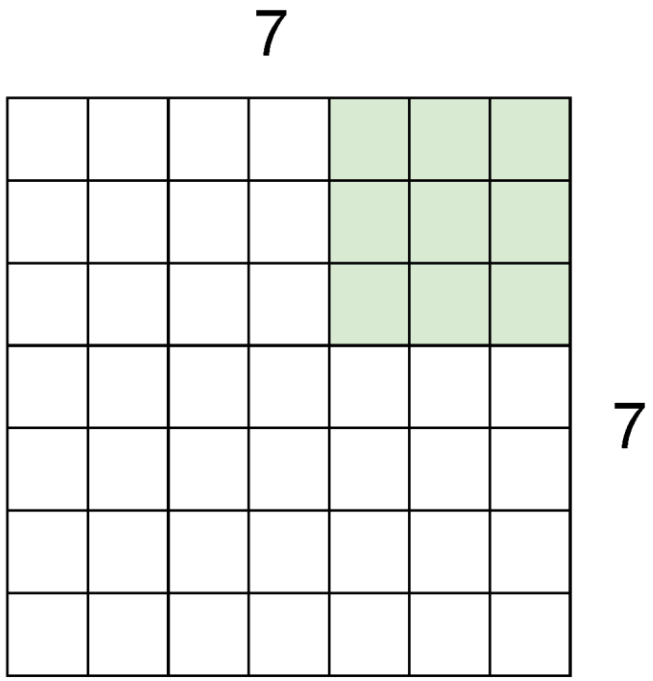


7x7 input (spatially)
assume 3x3 filter
applied **with stride 2**

Slide credit Fei-Fei Li

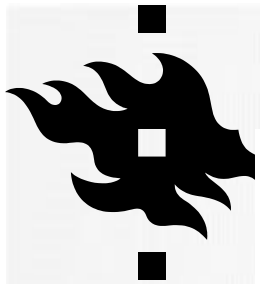


A closer look at spatial dimensions:

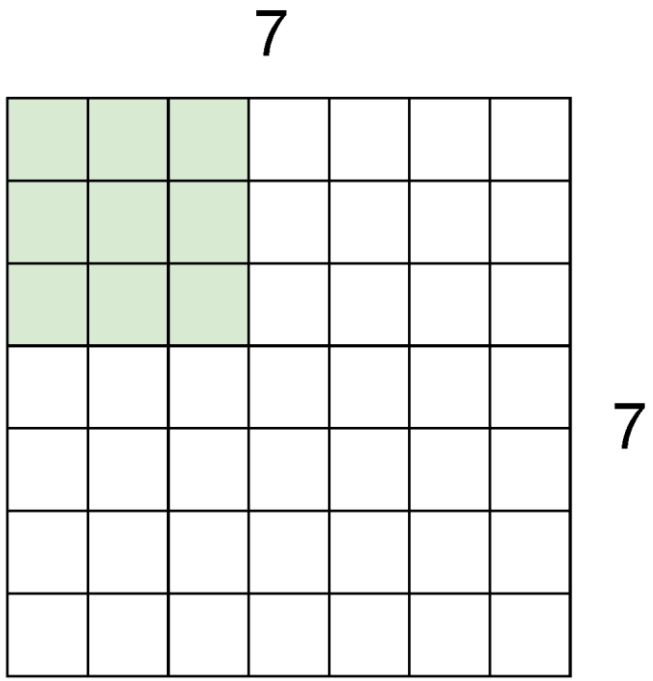


7x7 input (spatially)
assume 3x3 filter
applied **with stride 2**
=> 3x3 output!

Slide credit Fei-Fei Li



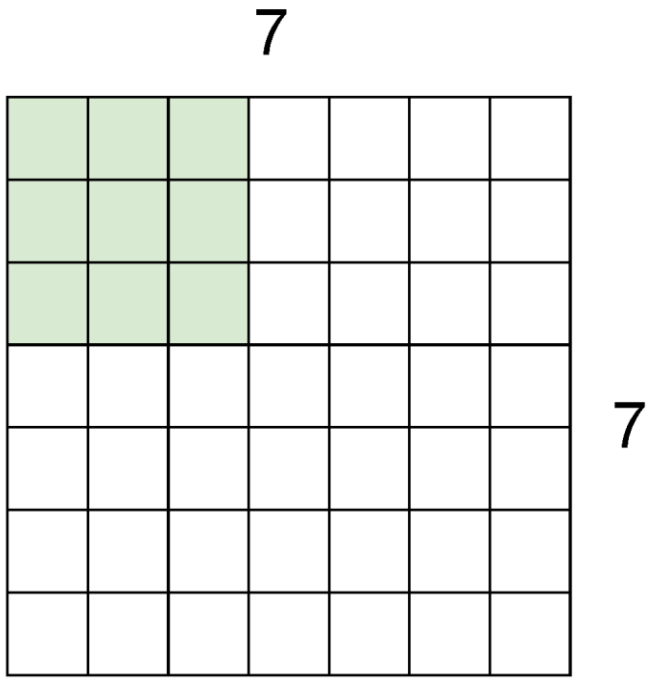
A closer look at spatial dimensions:



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

Slide credit Fei-Fei Li

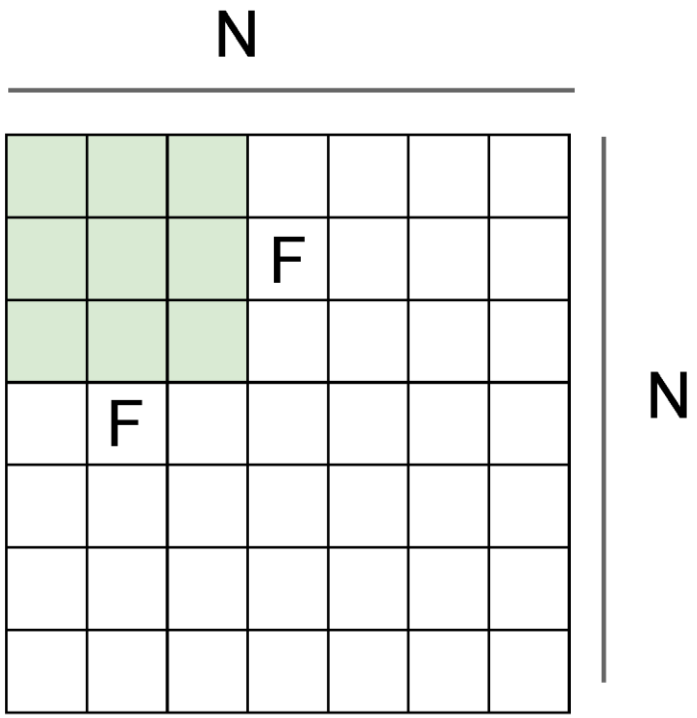
A closer look at spatial dimensions:



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

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

Slide credit Fei-Fei Li

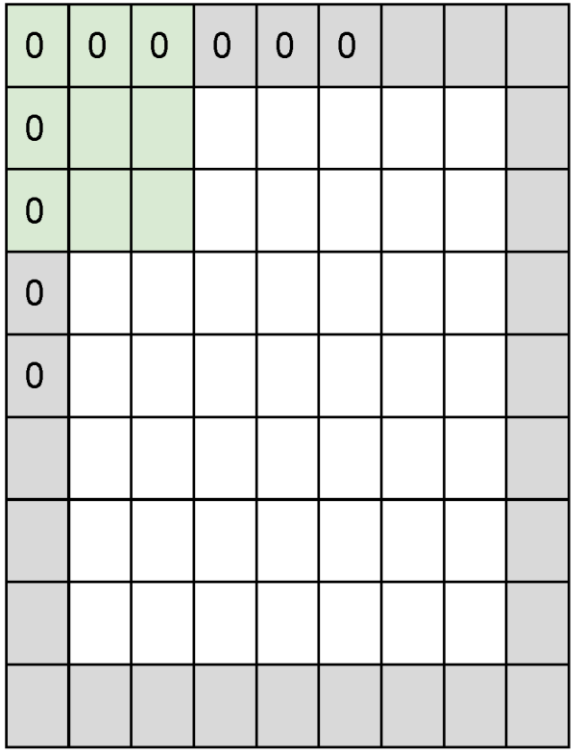


Output size:
(N - F) / stride + 1

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 :\

Slide credit Fei-Fei Li

In practice: Common to zero pad the border

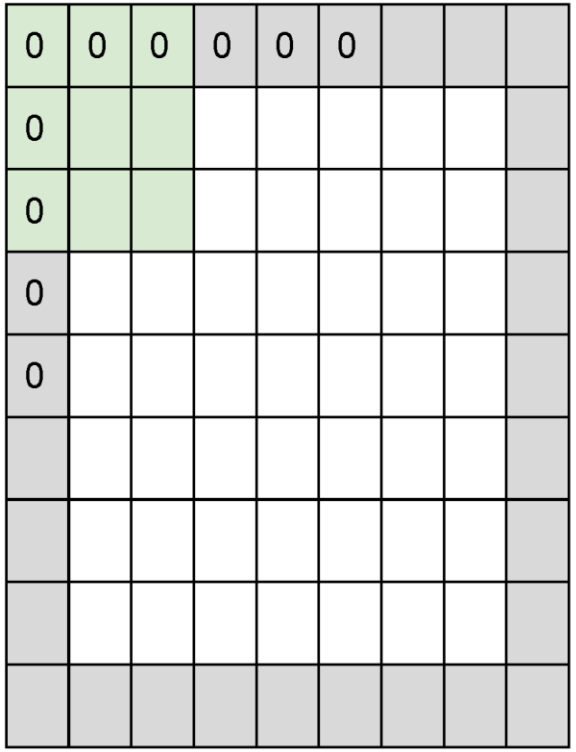


e.g. input 7x7
3x3 filter, applied with **stride 1**
pad with 1 pixel border => what is the output?

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

Slide credit Fei-Fei Li

In practice: Common to zero pad the border

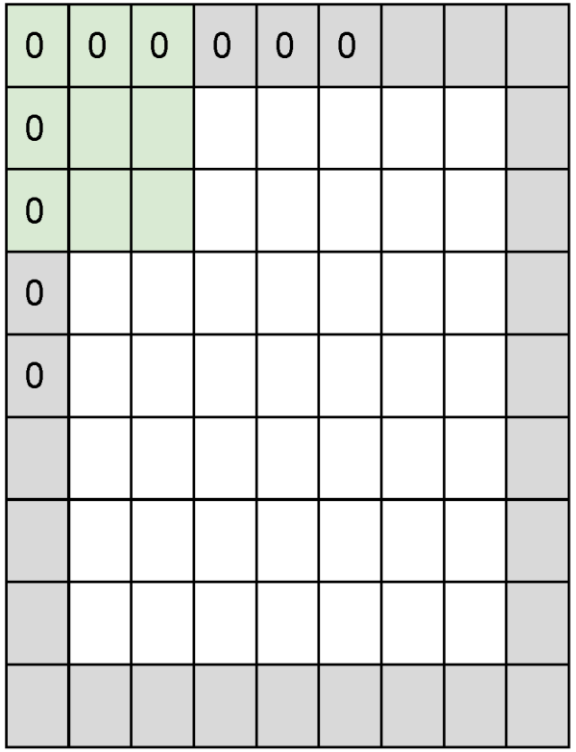


e.g. input 7x7
3x3 filter, applied with **stride 1**
pad with 1 pixel border => what is the output?

7x7 output!

Slide credit Fei-Fei Li

In practice: Common to zero pad the border



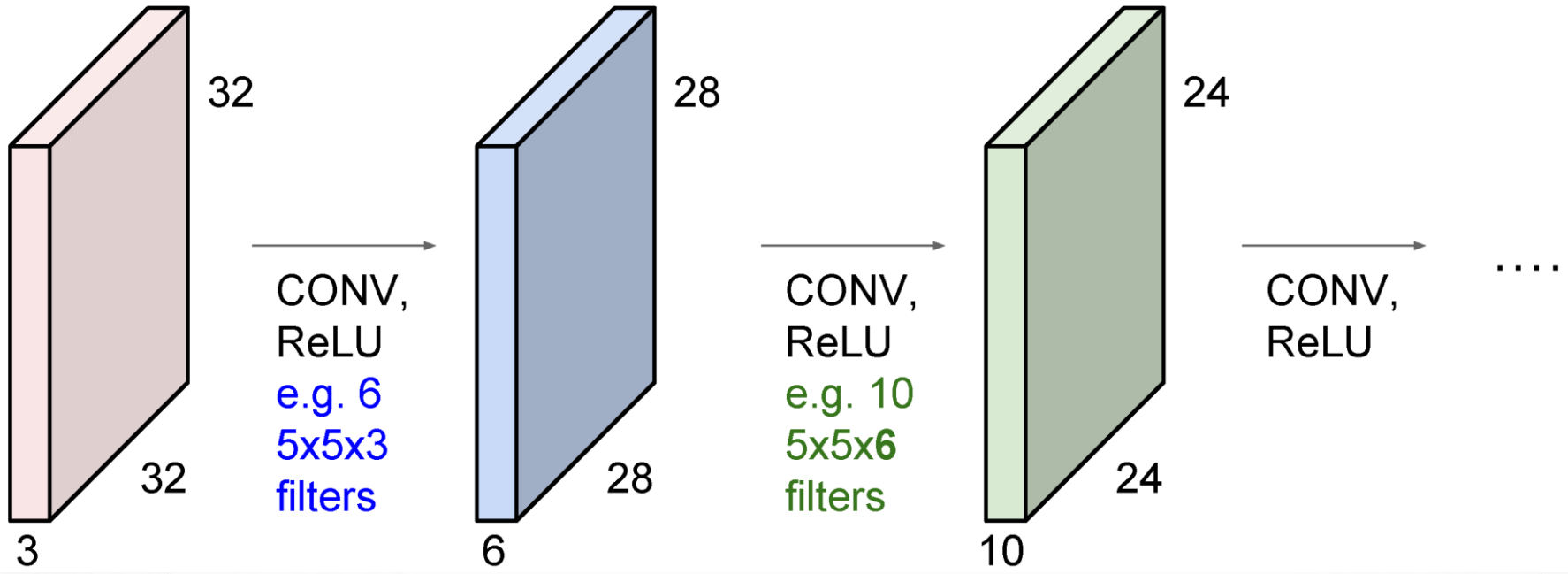
e.g. input 7x7
3x3 filter, applied with **stride 1**
pad with 1 pixel border => what is the output?

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

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.

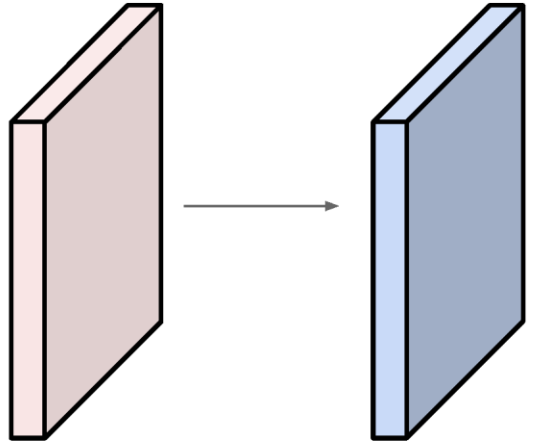


Slide credit Fei-Fei Li

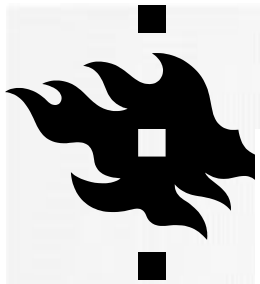
Examples time:

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

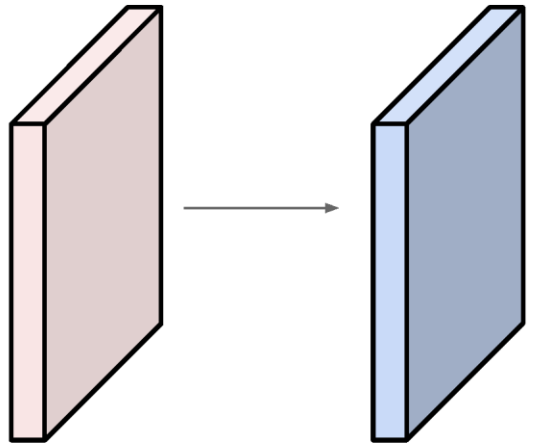
Output volume size: ?



Slide credit Fei-Fei Li



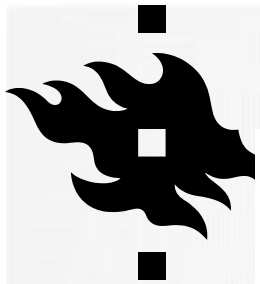
Examples time:



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

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

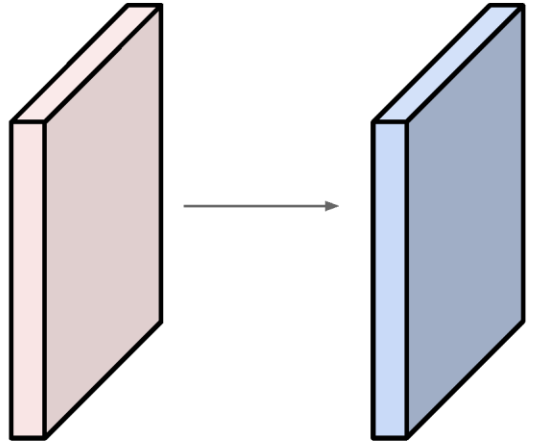
Slide credit Fei-Fei Li



Examples time:

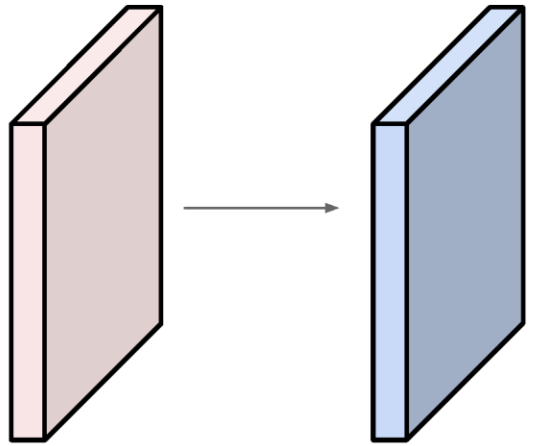
Input volume: **32x32x3**
10 5x5 filters with stride 1, pad 2

Number of parameters in this layer?



Slide credit Fei-Fei Li

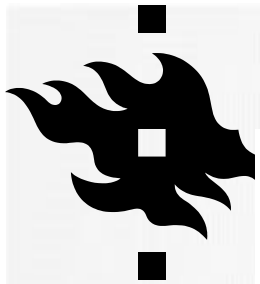
Examples time:



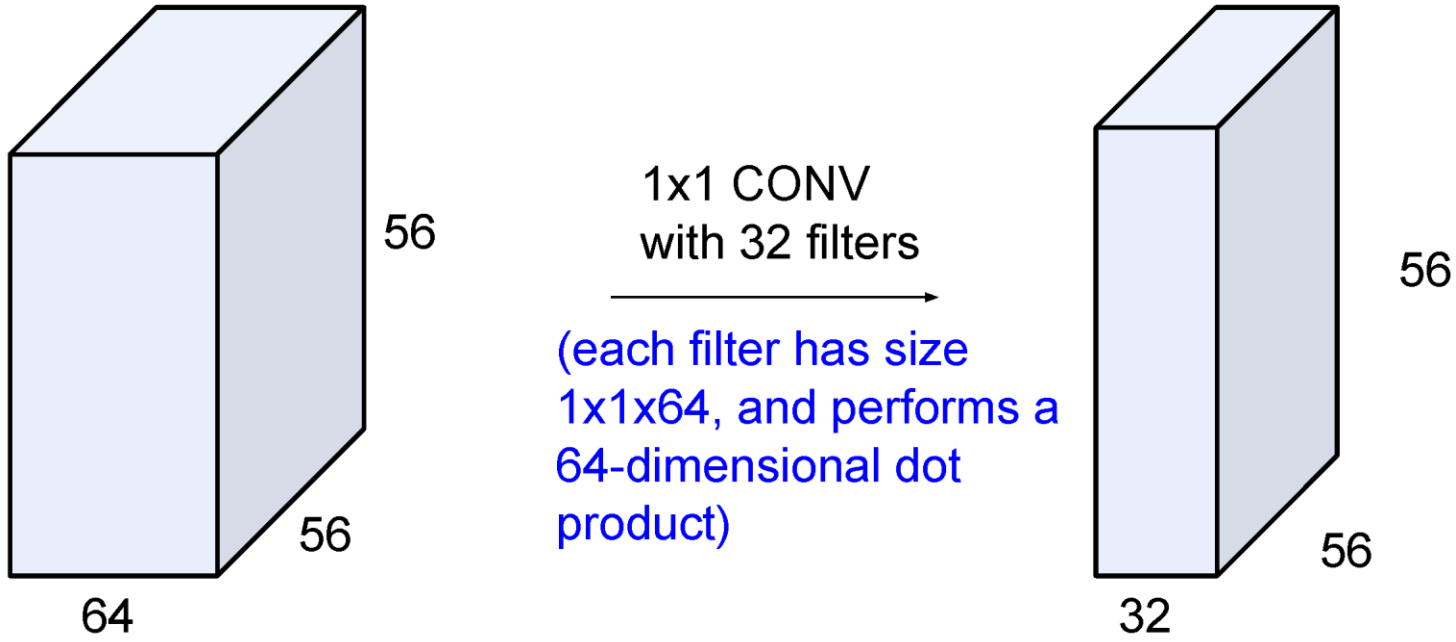
Input volume: **32x32x3**
10 5x5 filters with stride 1, pad 2

Number of parameters in this layer?
each filter has 5*5*3 + 1 = 76 params (+1 for bias)
=> 76*10 = **760**

Slide credit Fei-Fei Li



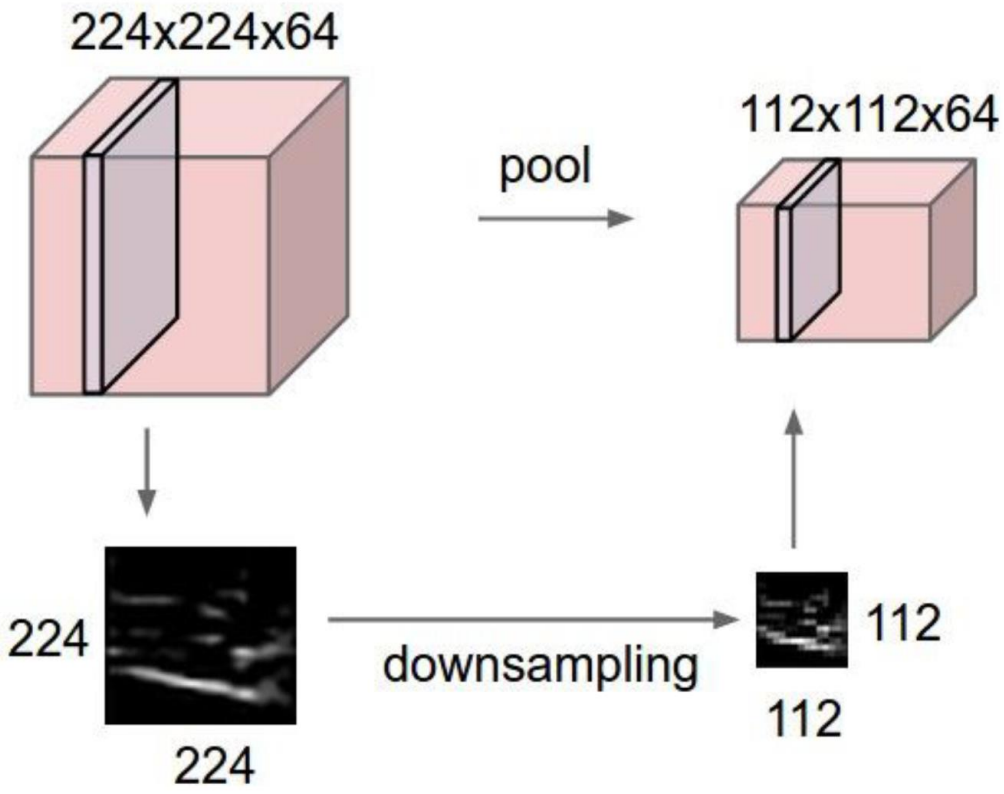
(btw, 1x1 convolution layers make perfect sense)



Slide credit Fei-Fei Li

Pooling layer

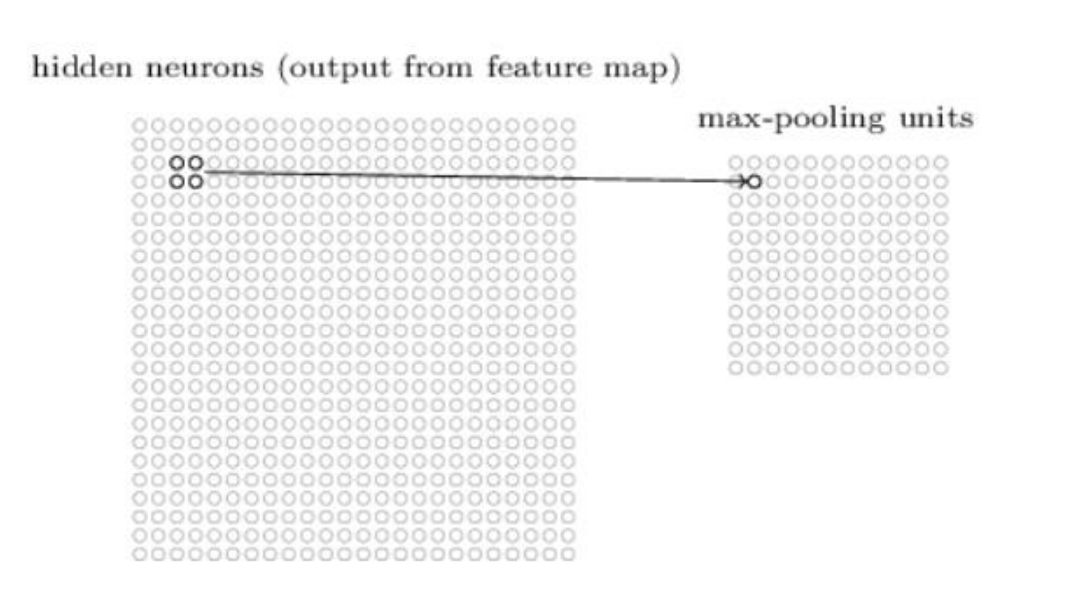
- makes the representations smaller and more manageable
- operates over each activation map independently:



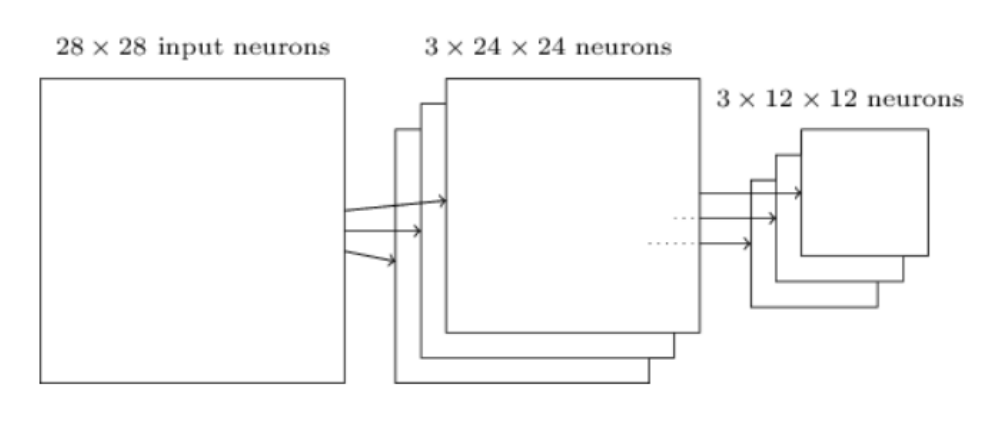
Slide credit Fei-Fei Li

Pooling

- Simplify the output from the convolutional layer

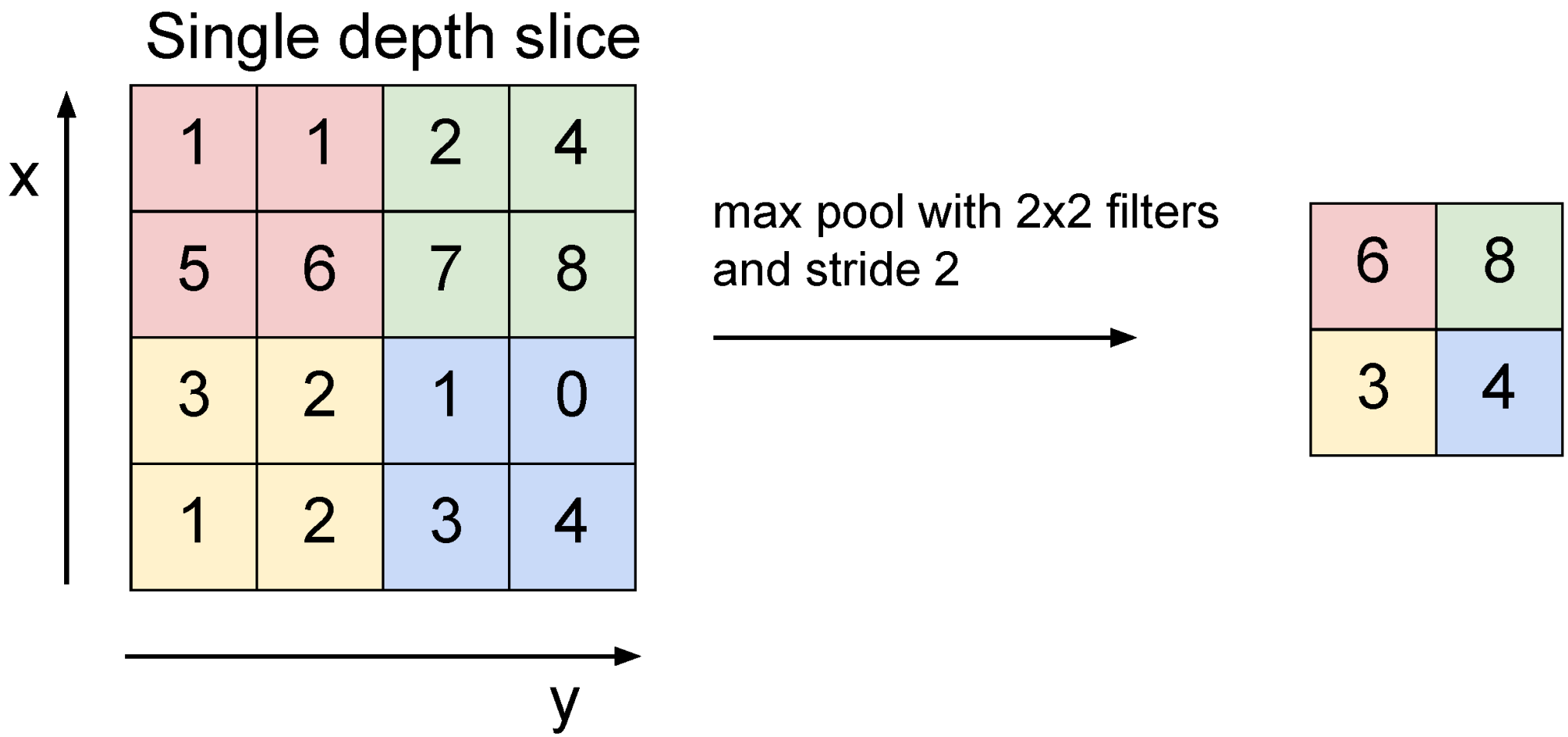


Max pooling



Once the feature was found, pooling throws away the exact position of the feature, keeps its position wrt other features

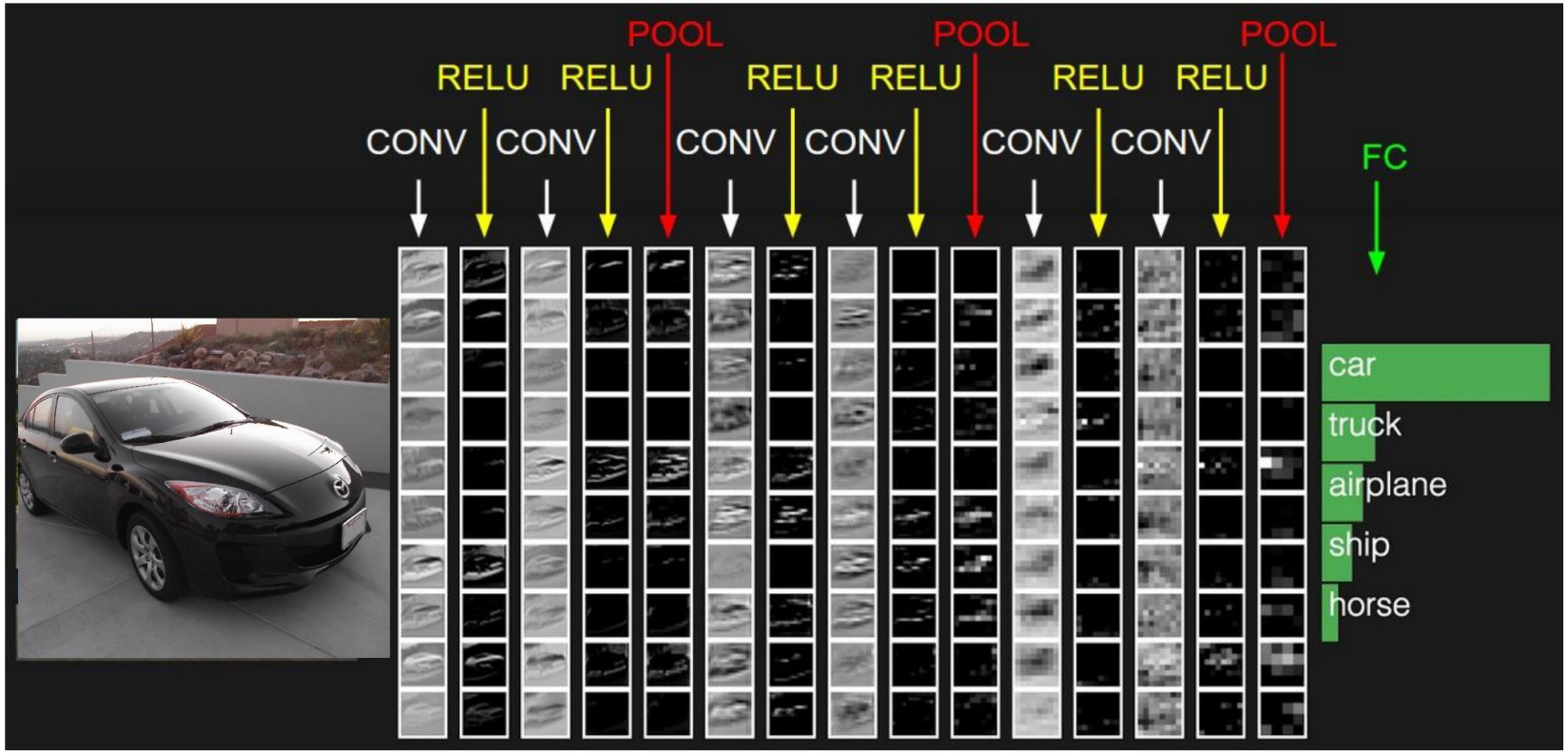
MAX POOLING



Slide credit Fei-Fei Li

Fully Connected Layer (FC layer)

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



Improving CNN performance

- Expanding the size of the fully-connected layer
  - 300 to 1000 neurons, no significant impact
- Adding an extra fully-connected layer
  - No significant impact
- Improving training
  - Dropout => removing individual activations at random while training the network
  - Makes the model more robust to the loss of individual pieces of evidence => less likely to rely on particular idiosyncracies of the training data
  - Good results

Improving CNN performance 2

- Dropout is not done for convolutional layers
  - There's no need: the convolutional layers have considerable inbuilt resistance to overfitting
  - The reason is that the shared weights mean that convolutional filters are forced to learn from across the entire image
  - This makes them less likely to pick up on local idiosyncracies in the training data.
  - And so there is less need to apply other regularizers, such as dropout.

Improving CNN performance 3

- Ensembling = creating several neural networks, getting them to vote to determine the best classification
  - E.g. training 5 different neural networks, with each achieving accuracies near to 99.6 percent
  - Even though the networks would all have similar accuracies, they might well make different errors, due to the different random initializations
  - Taking a vote amongst 5 networks might yield a classification better than any individual network

Expanding the training data

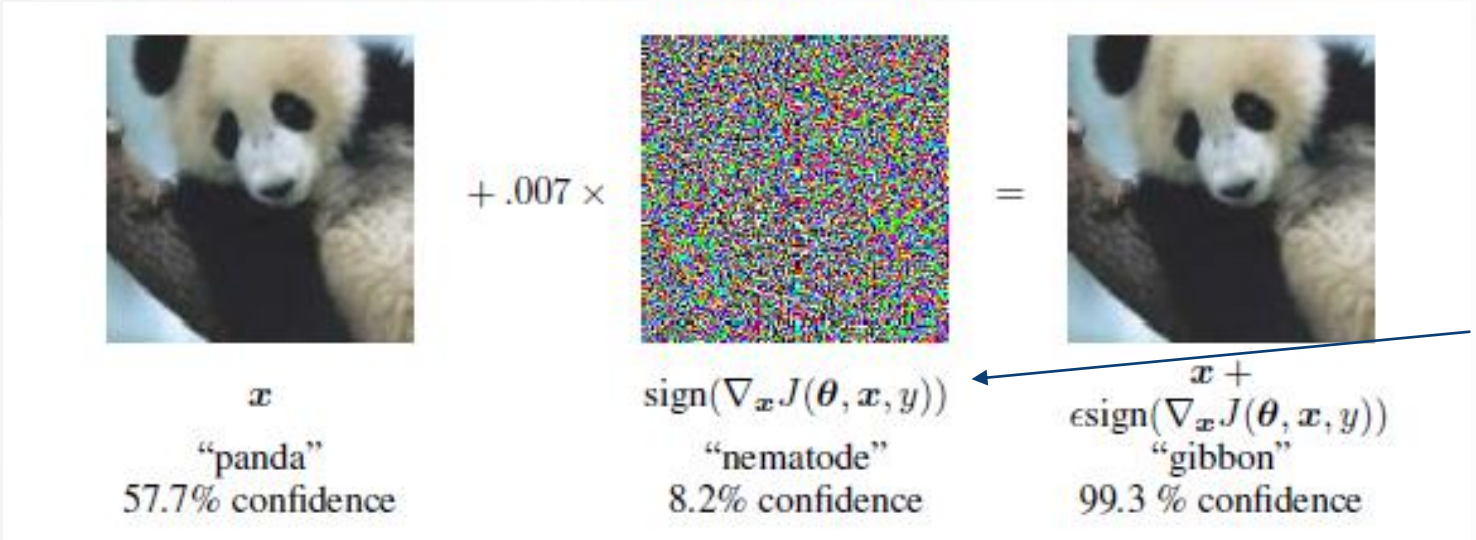
- Expanding the training data
  - Improving results by algorithmically expanding the training data. A simple way of expanding the training data is to displace each training image by a single pixel, either up one pixel, down one pixel, left one pixel, or right one pixel
- E.g. take the 50,000 MNIST training images, and prepare an expanded training set, with 250,000 training images
- Use training images to train the network
- Number of training epochs may be reduced, since training is done with 5 times as much data
- Expanding the data reduces the effect of overfitting

GENERATIVE ADVERSIAL NETWORKS (GANS)

- Adversarial examples : examples that are only slightly different from correctly classified examples drawn from the data distribution
- Many models trained on different subsets of the training data misclassify the same adversarial example
- Adversarial examples can identify critical blind spots in training algorithms
- In practice they are inputs to machine learning models that an attacker has intentionally designed to cause the model to make an error, like optical illusions for machines

Goodfellow et al. (2015) Explaining and harnessing Adversarial examples. In Proceedings of ICLR.

GENERATIVE ADVERSIAL NETWORKS (GANS)



adding a very small amount of *carefully constructed* noise

Fast gradient sign method: J(; x; y) cost used to train the neural network, linearizing the cost function around the current value of θ => an optimal max-norm constrained pertubation of ϵ sign($\nabla_x J(\theta, x, y)$)

Goodfellow et al. (2015) Explaining and harnessing Adversarial examples. In Proceedings of ICLR.

COUNTERMEASURES

- **Adversarial training**: generating many adversarial examples and training the model not to be fooled by each of them (cleverhans)
- **Defensive distillation**: Training the model to output probabilities of different classes, creating a model whose surface is smoothed in the directions an adversary will typically try to exploit
- These algorithms can still be bypassed by giving more computational power to the attacker

Fast-forward to today: ConvNets are everywhere

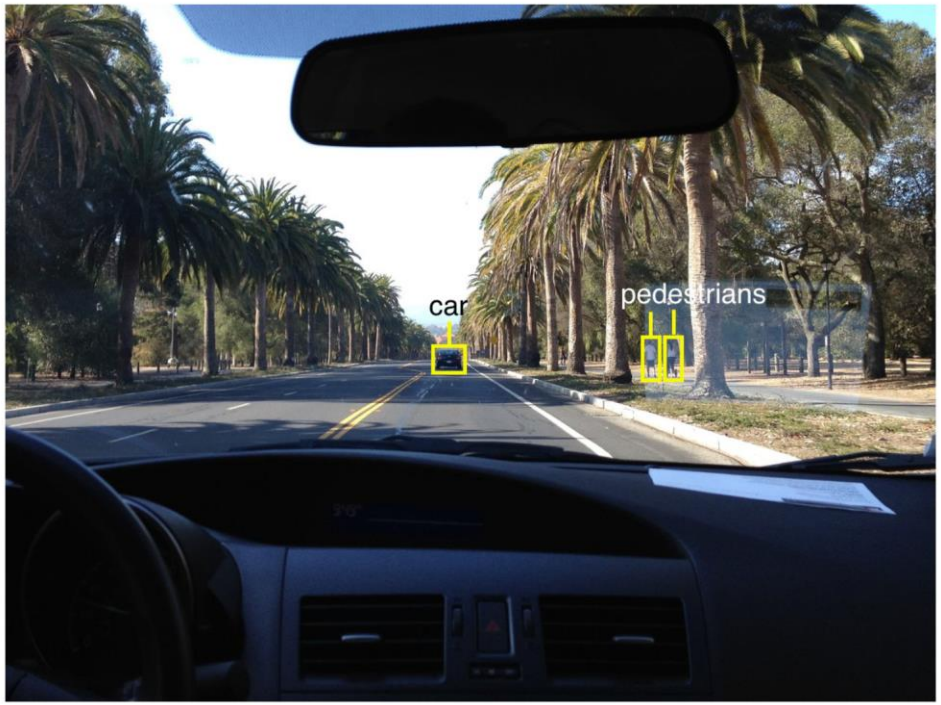


Photo by Lane McIntosh. Copyright CS231n 2017.

self-driving cars

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

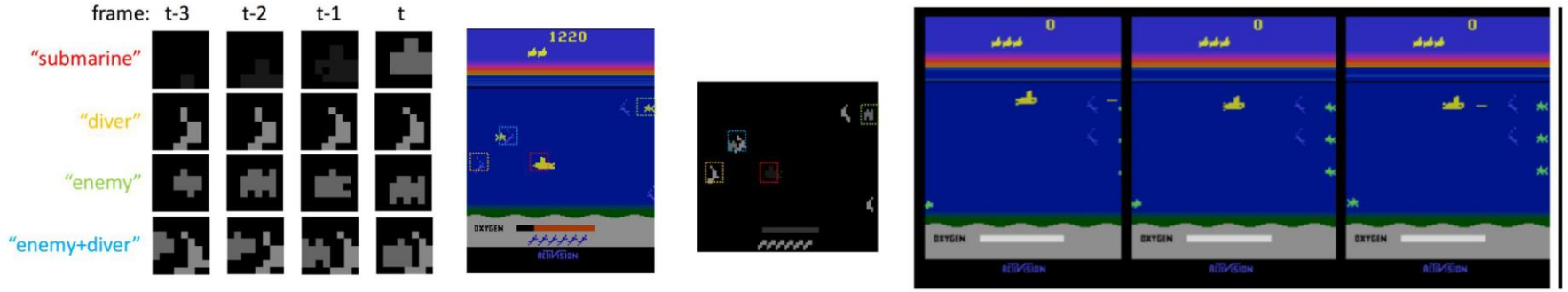
Note that for embedded systems a typical setup would involve NVIDIA Tegras, with integrated GPU and ARM-based CPU cores.

Slide credit Fei-Fei Li

Fast-forward to today: ConvNets are everywhere

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

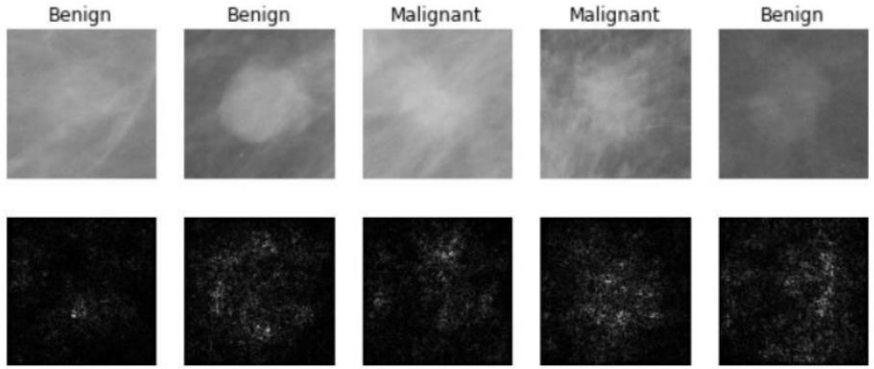
[Toshev, Szegedy 2014]



[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]

[Dieleman et al. 2014]

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

Slide credit Fei-Fei Li

Image Captioning

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

No errors

A white teddy bear sitting in the grass

A man riding a wave on top of a surfboard

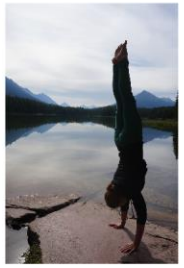
Minor errors

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

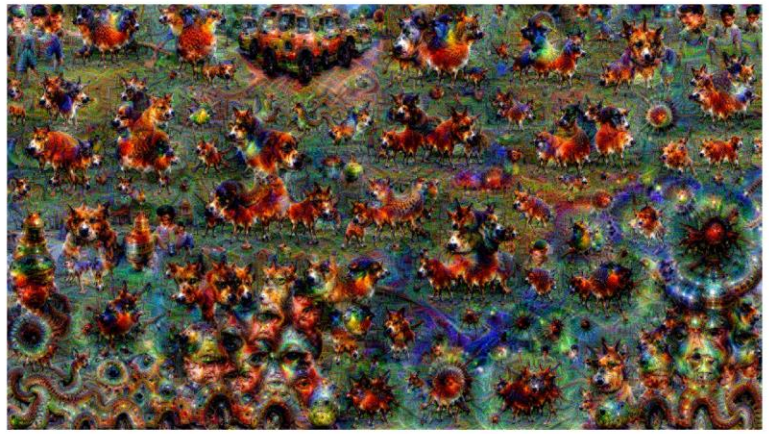


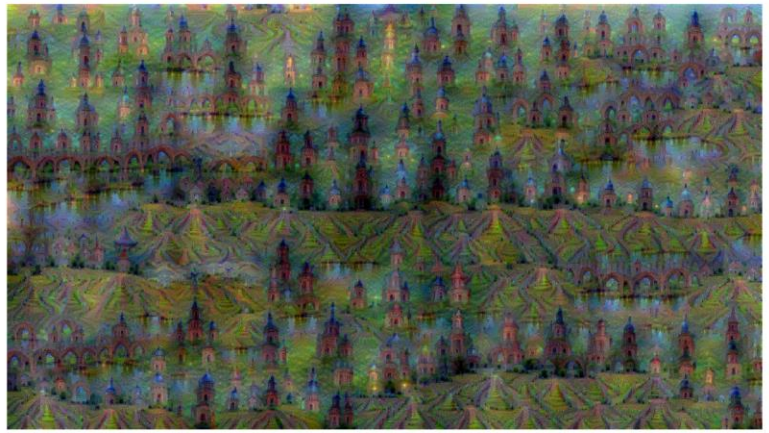
A woman standing on a beach holding a surfboard

All images are CC0 Public domain:
https://pixabay.com/en/luggage-antique-cat-1643010/
https://pixabay.com/en/teddy-plush-bears-cute-teddy-bear-1623436/
https://pixabay.com/en/surf-wave-summer-sport-litoral-1668716/
https://pixabay.com/en/woman-female-model-portrait-adult-983967/
https://pixabay.com/en/handstand-lake-meditation-496008/
https://pixabay.com/en/baseball-player-shortstop-infield-1045263/

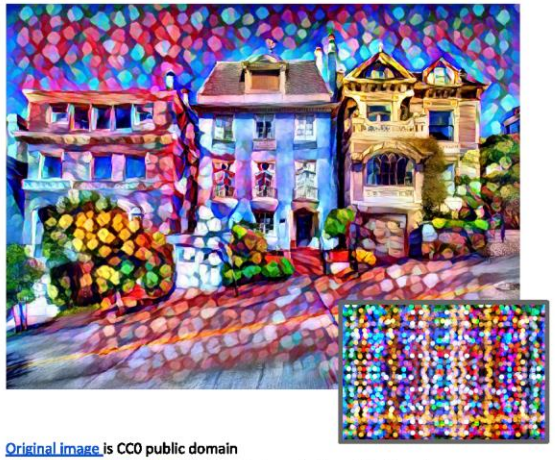
Captions generated by Justin Johnson using Neuraltalk2

Slide credit Fei-Fei Li

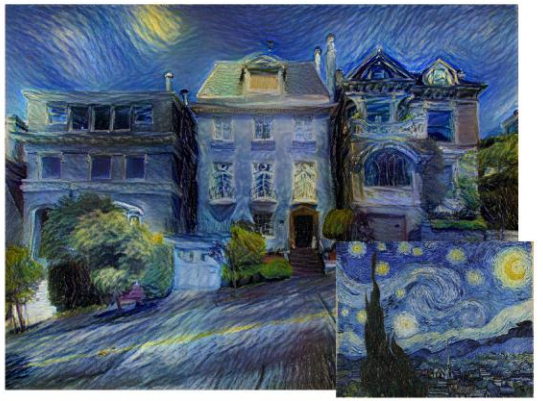


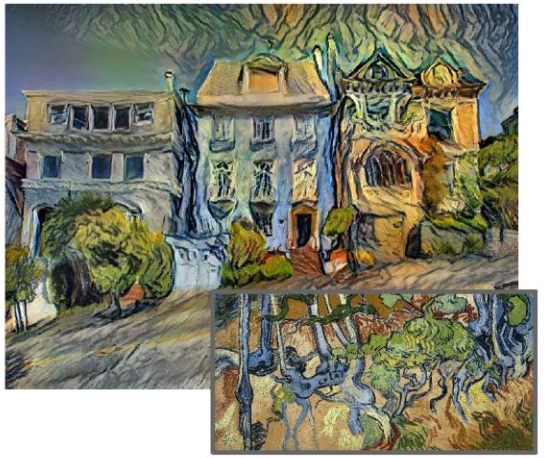


Figures copyright Justin Johnson, 2015. Reproduced with permission. Generated using the Inceptionism approach from a blog post by Google Research.



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

Slide credit Fei-Fei Li


60˚ 10 1.2 N, 24˚ 57 18 E