max planck institut informatik

SIC Saarland Informatics Campus

Elements of Data Science and Artificial Intelligence

Machine Learning Intro — Part 3

Bernt Schiele - schiele@mpi-inf.mpg.de

Overview Today's Lecture

- "Standard" Neural Networks
  - from linear classifiers to multi-layer neural networks (e.g. perceptrons)

- Convolutional Neural Networks (CNNs)
  - motivation
  - convolution layer + activation function + pooling + fully connected layer
  - visualizing layers
  - depth matters

max planck institut
informatik

SIC Saarland Informatics Campus

“Standard” Neural Networks

Classification Problem

- E.g.: Image classification:

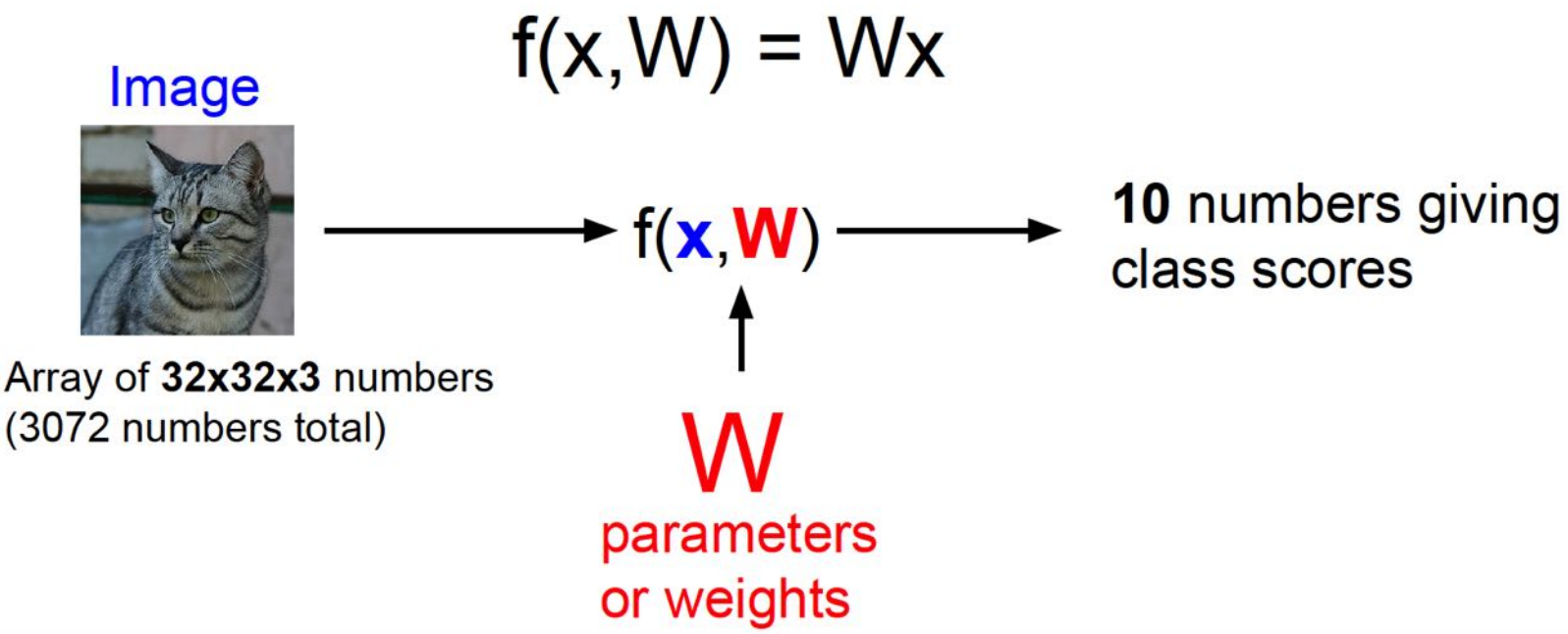
This image by Nikita is licensed under CC-BY 2.0

(assume given set of discrete labels)
{dog, cat, truck, plane, ...}

→ cat

slide credit: Fei-Fei, Justin Johnson, Serena Yeung

Parametric Approach: Linear Classifier

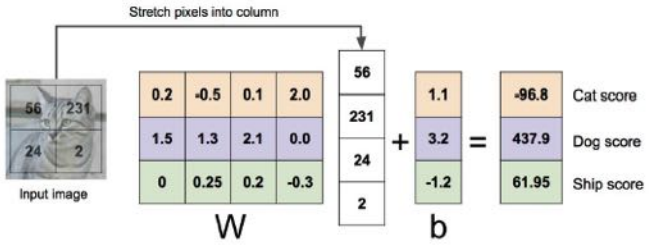


slide credit: Fei-Fei, Justin Johnson, Serena Yeung

Linear Classifier: Three Viewpoints

Algebraic Viewpoint

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

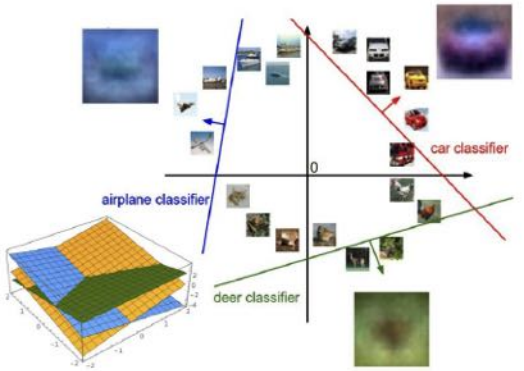


Visual Viewpoint

One template per class

Geometric Viewpoint

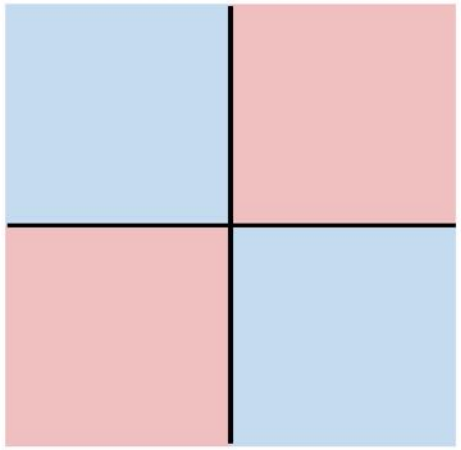
Hyperplanes cutting up space



Hard Cases for a Linear Classifier

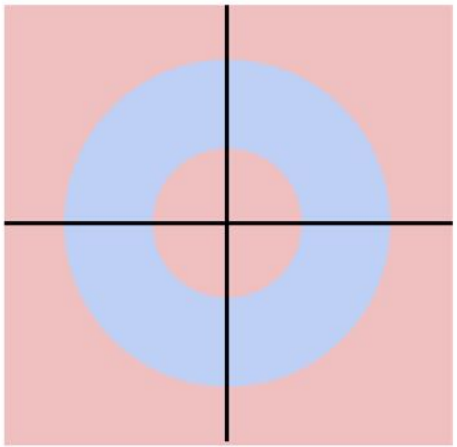
Class 1:
First and third quadrants

Class 2:
Second and fourth quadrants



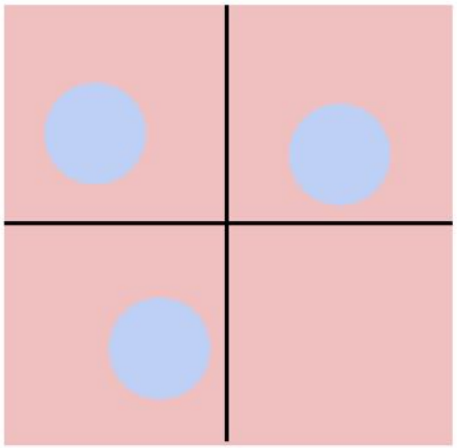
Class 1:
1 <= L2 norm <= 2

Class 2:
Everything else

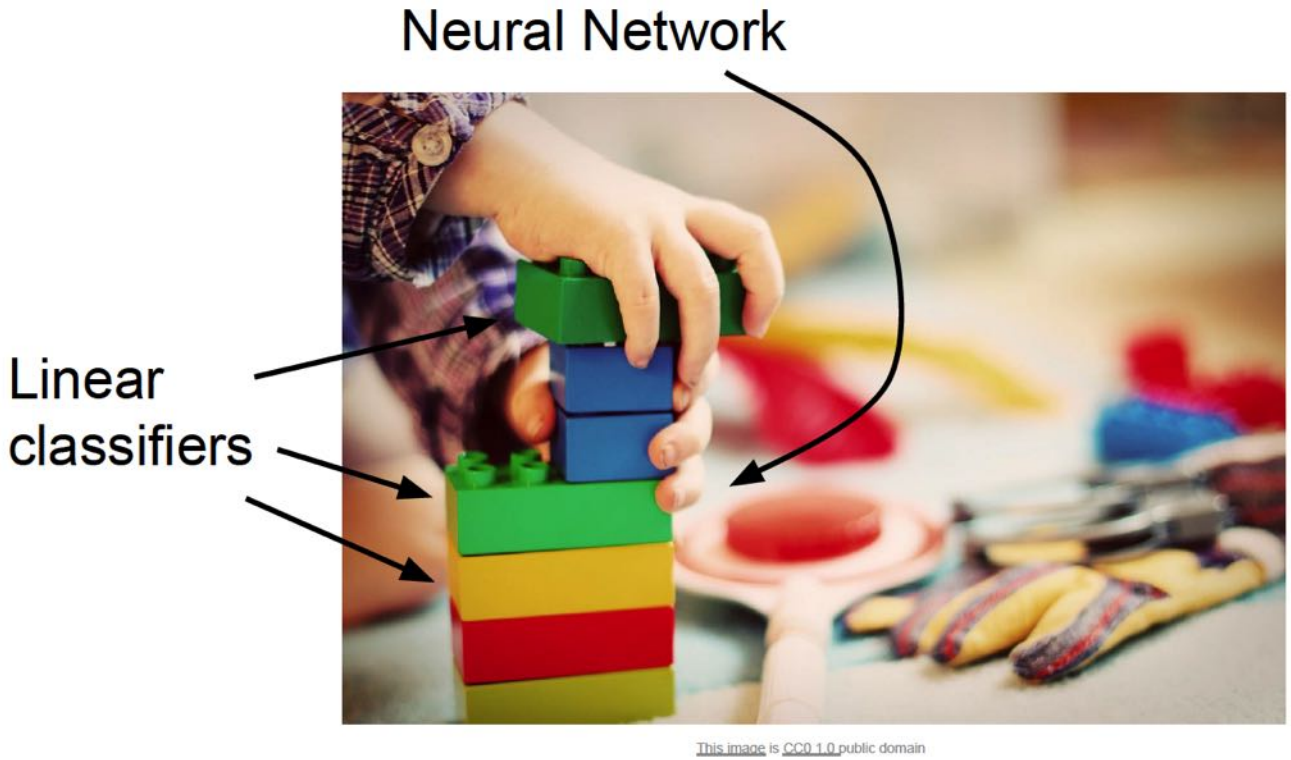


Class 1:
Three modes

Class 2:
Everything else



Linear Classification



slide credit: Fei-Fei, Justin Johnson, Serena Yeung

“Standard” Neural Networks

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

slide credit: Fei-Fei, Justin Johnson, Serena Yeung

“Standard” Neural Networks

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

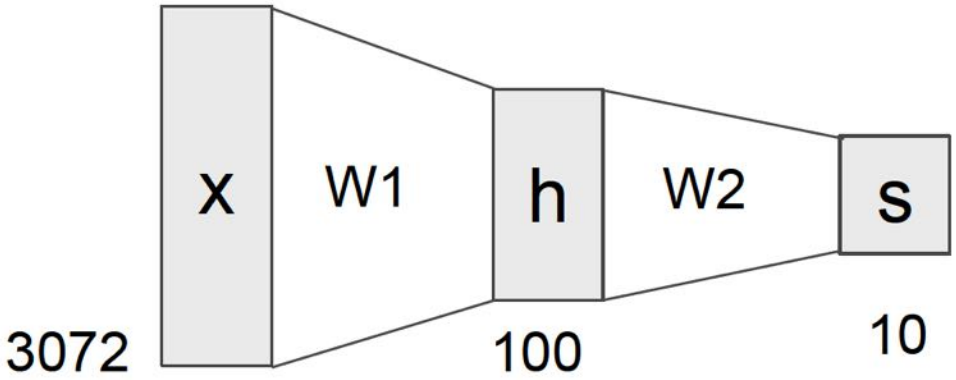
(**Now**) 2-layer Neural Network $f = W_2 \max(0, W_1 x)$

slide credit: Fei-Fei, Justin Johnson, Serena Yeung

“Standard” Neural Networks

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

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

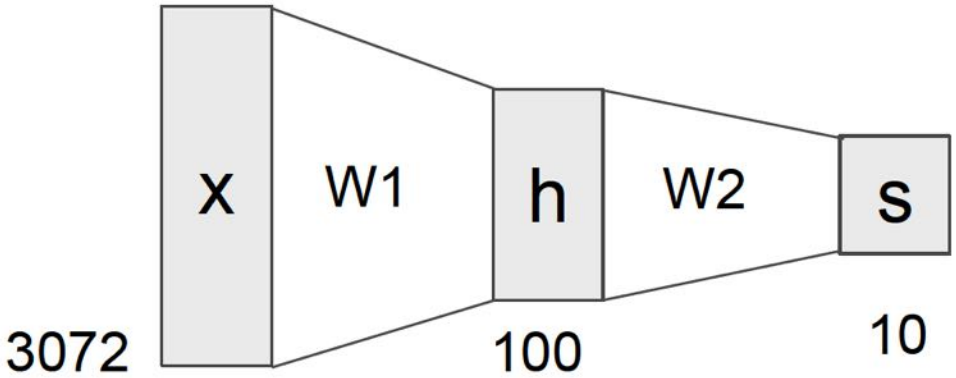


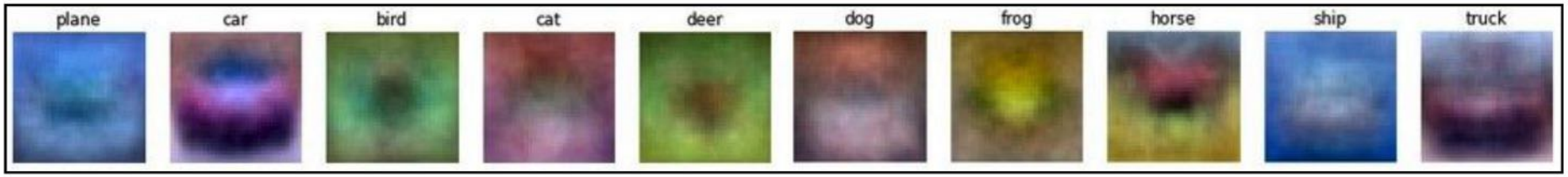
slide credit: Fei-Fei, Justin Johnson, Serena Yeung

“Standard” 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, Justin Johnson, Serena Yeung

“Standard” Neural Networks

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

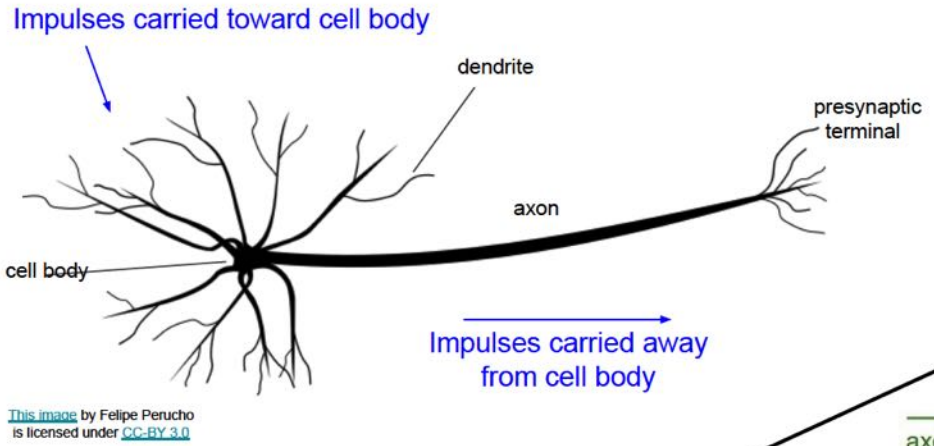
(**Now**) 2-layer Neural Network $f = W_2 \max(0, W_1 x)$

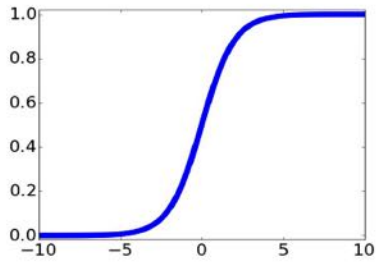
or 3-layer Neural Network

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

slide credit: Fei-Fei, Justin Johnson, Serena Yeung

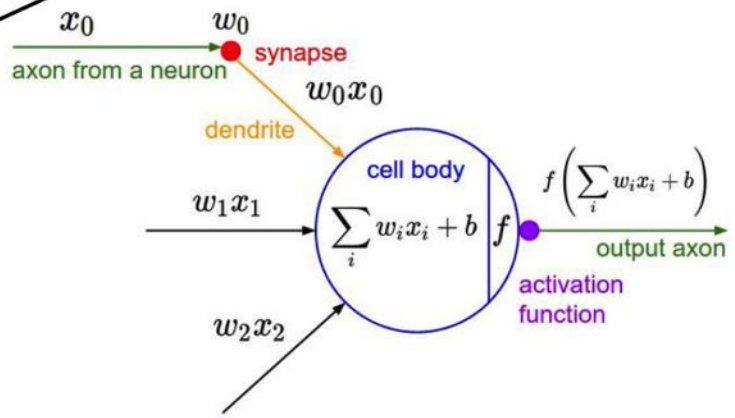
Some Inspiration from the (Human) Brain





sigmoid activation function

$$\frac{1}{1+e^{-x}}$$



slide credit: Fei-Fei, Justin Johnson, Serena Yeung

Some Inspiration from the (Human) Brain

- Be careful with your brain analogies…

- Biological neurons:
  - many different types
  - dendrites can perform complex non-linear computations
  - synapses are n ot a single weight but a complex non-linear dynamical system
  - e.g. [Dendritic Computation, London & Hauser]

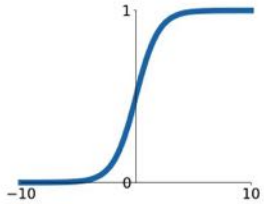
slide credit: Fei-Fei, Justin Johnson, Serena Yeung

Activation Functions

Activation functions

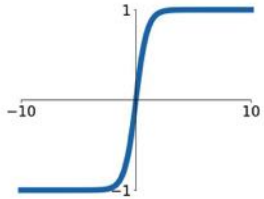
Sigmoid

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



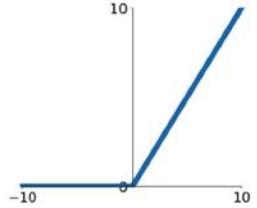
tanh

$$\tanh(x)$$



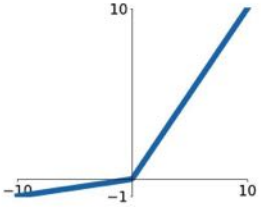
ReLU

$$\max(0, x)$$



Leaky ReLU

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

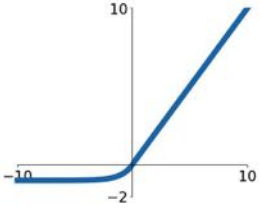


Maxout

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

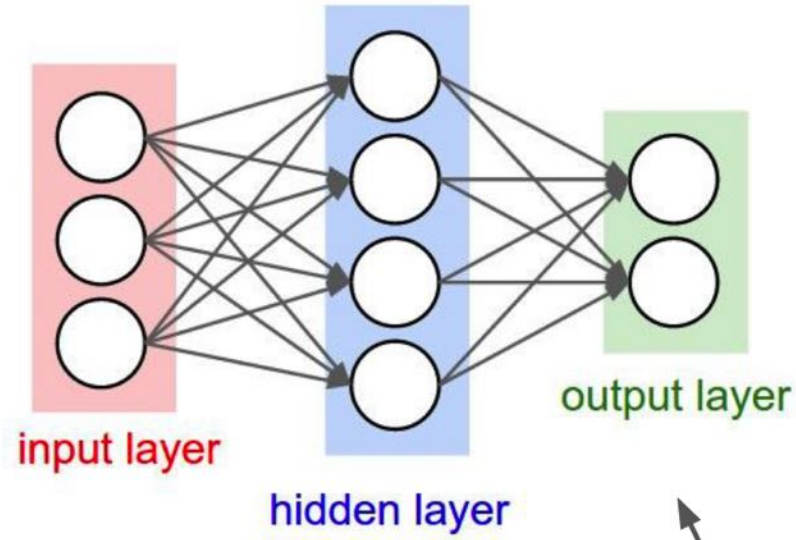
ELU

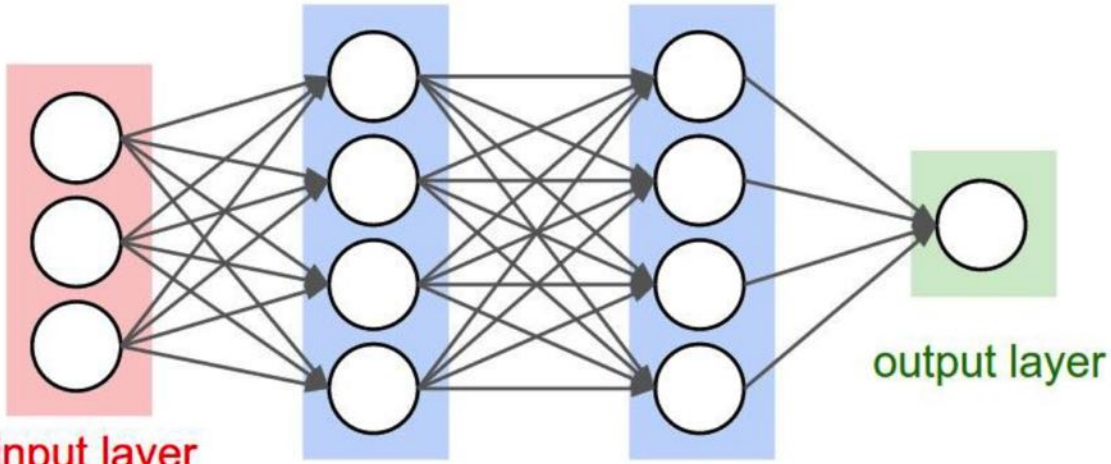
$$\begin{cases} x & x \geq 0 \\ \alpha(e^x - 1) & x < 0 \end{cases}$$



slide credit: Fei-Fei, Justin Johnson, Serena Yeung

Neural networks: Architectures





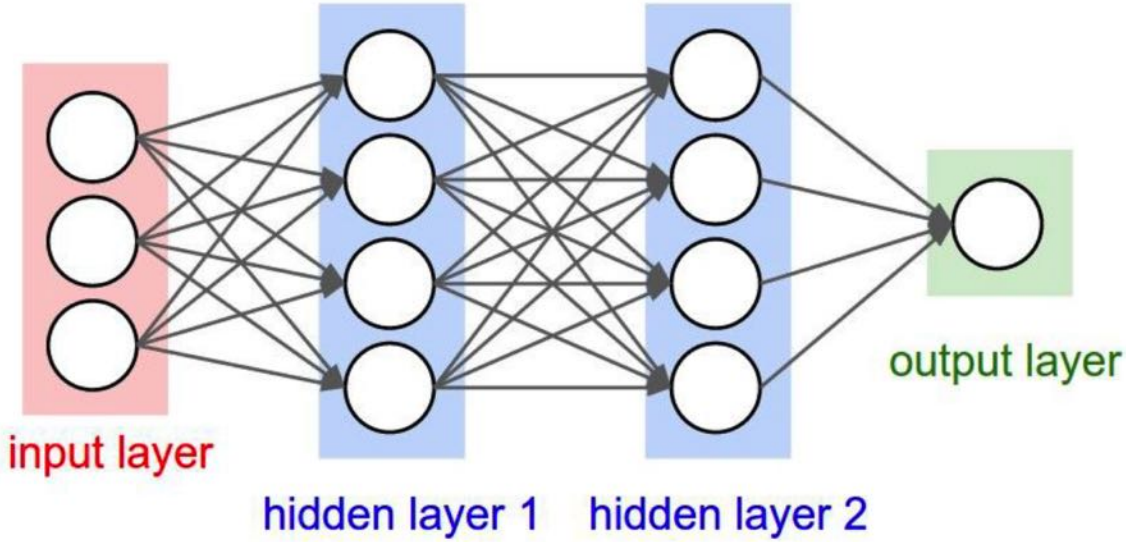
"2-layer Neural Net", or
"1-hidden-layer Neural Net"

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

"Fully-connected" layers

slide credit: Fei-Fei, Justin Johnson, Serena Yeung

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, Justin Johnson, Serena Yeung

Summary

- We arrange neurons in fully-connected layers
- The abstraction of a layer has the nice property that it allows us to use efficient vectorized code (e.g. matrix multipliers — se previous slide)
- Neural networks are not really neural :)

- Next: Convolutional Neural networks

slide credit: Fei-Fei, Justin Johnson, Serena Yeung

max planck institut
informatik

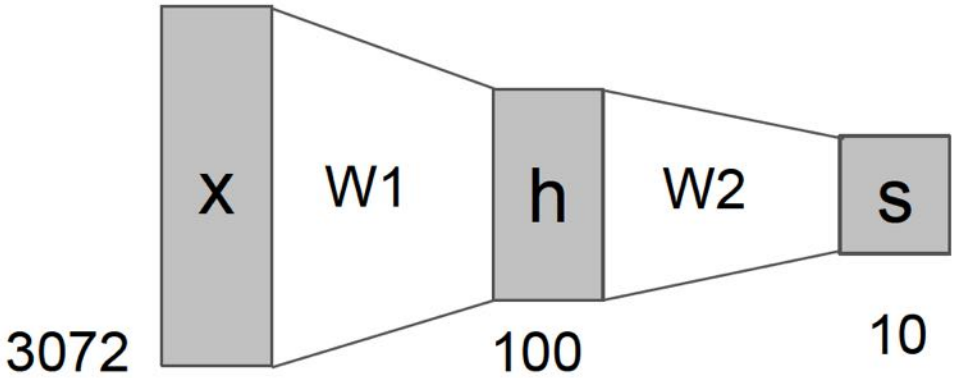
SIC Saarland Informatics Campus

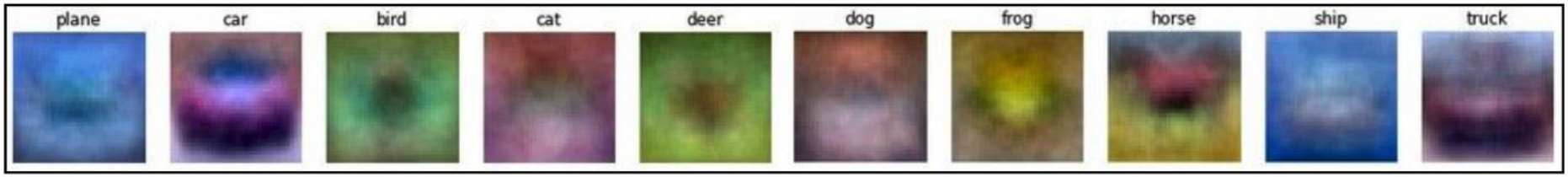
Convolutional Neural Networks (CNNs)

So far: Neural Networks

Linear score function: $f = Wx$

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





slide credit: Fei-Fei, Justin Johnson, Serena Yeung

Now: Convolutional Neural Networks

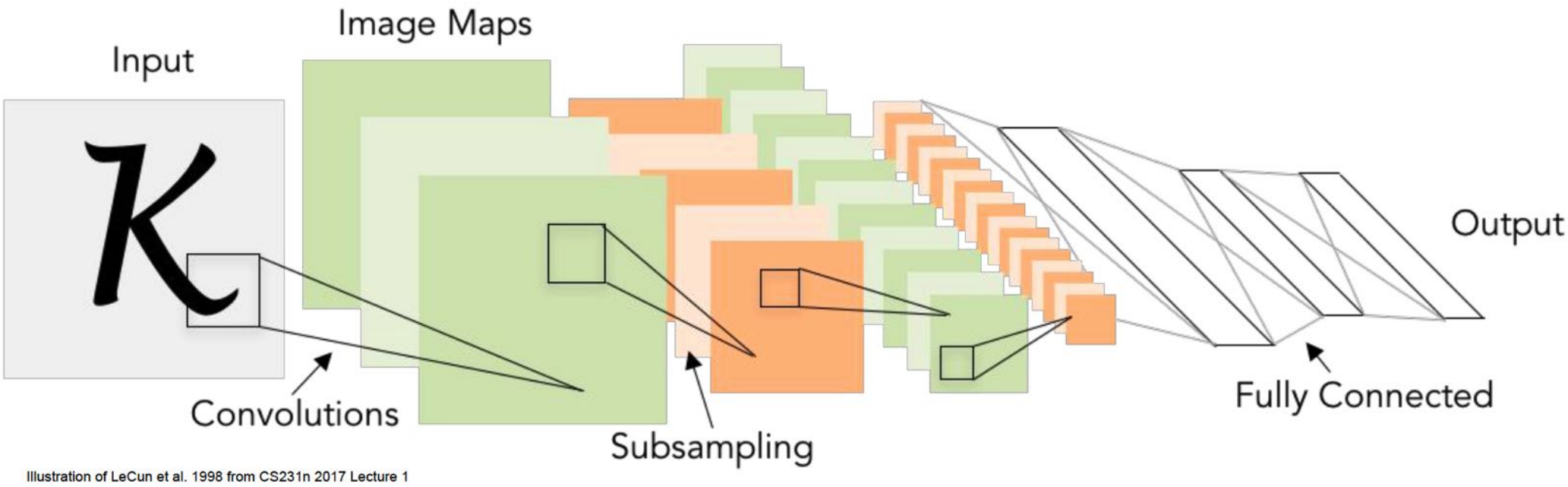


Illustration of LeCun et al. 1998 from CS231n 2017 Lecture 1

slide credit: Fei-Fei, Justin Johnson, Serena Yeung

A bit of history...

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

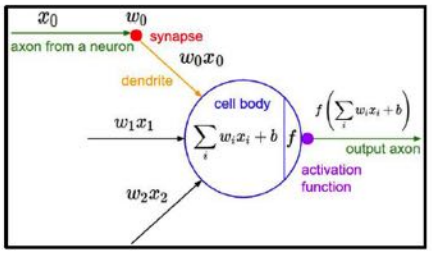
The machine was connected to a camera that used 20×20 cadmium sulfide photocells to produce a 400-pixel image.

recognized
letters of the alphabet

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

update rule:

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

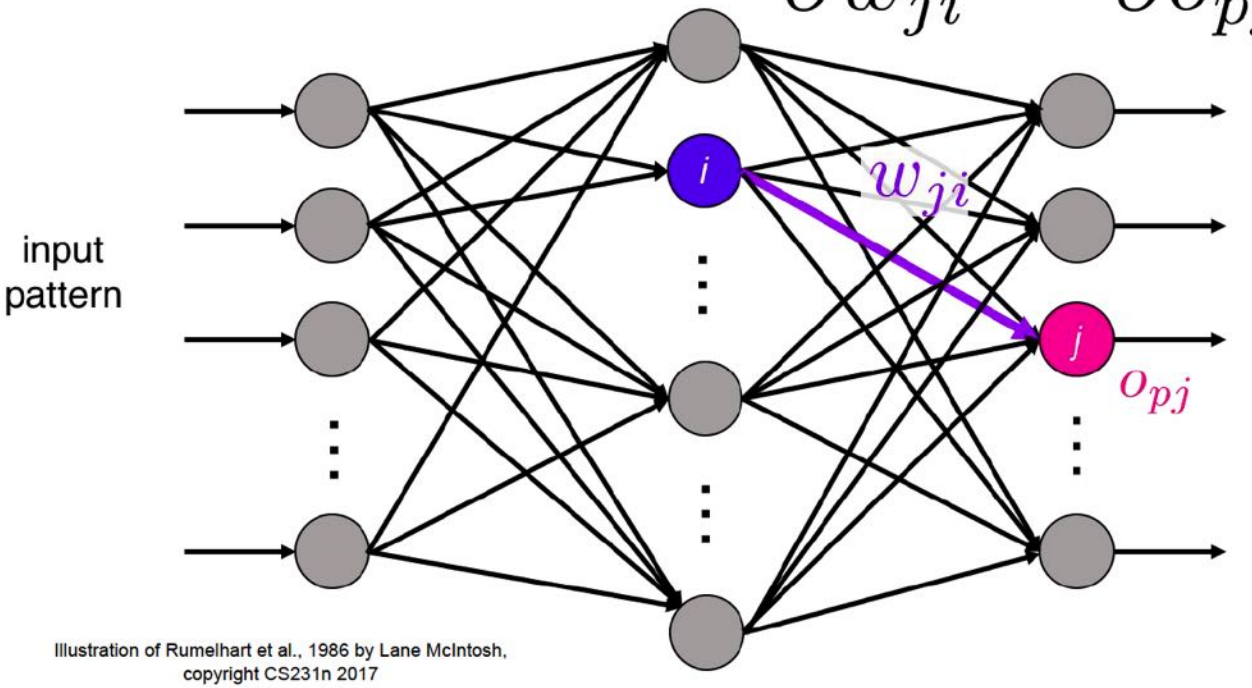


Frank Rosenblatt, ~1957: Perceptron

slide credit: Fei-Fei, Justin Johnson, Serena Yeung

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

slide credit: Fei-Fei, Justin Johnson, Serena Yeung

A bit of history...

[Hinton and Salakhutdinov 2006]

Reinvigorated research in Deep Learning

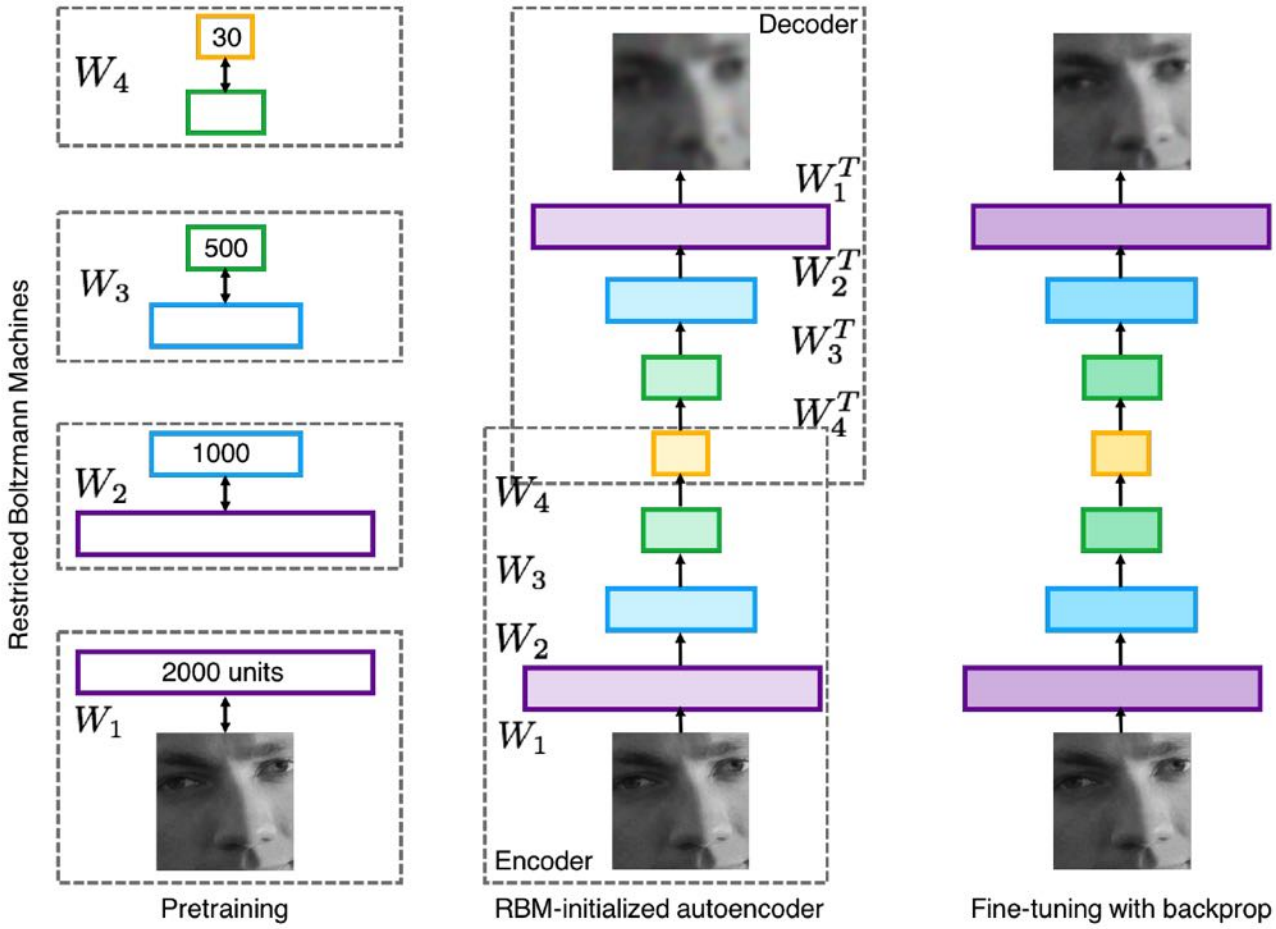


Illustration of Hinton and Salakhutdinov 2006 by Lane McIntosh, copyright CS231n 2017

slide credit: Fei-Fei, Justin Johnson, Serena Yeung

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

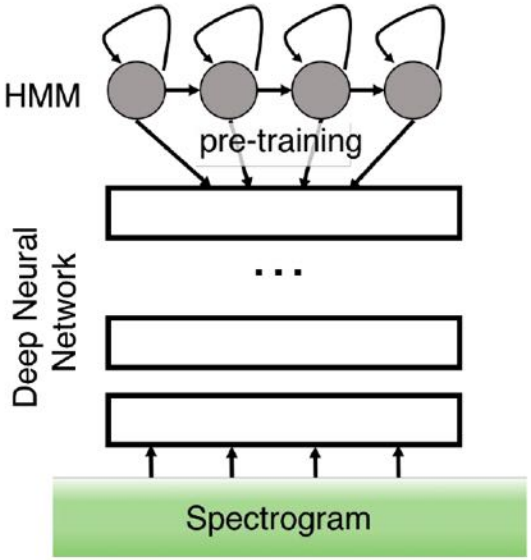
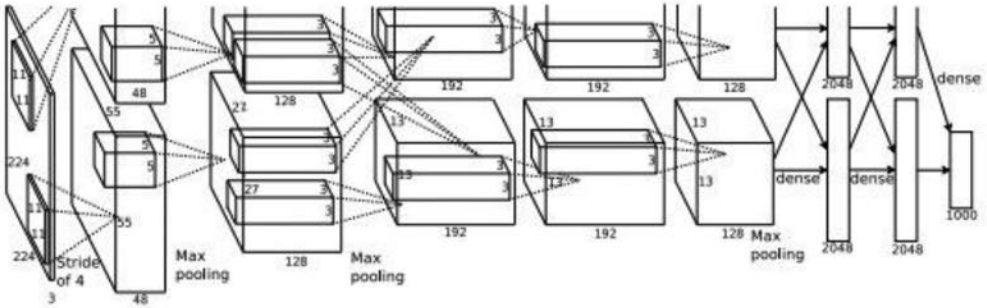
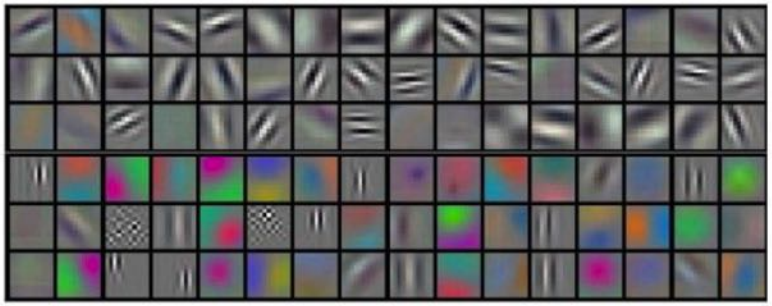


Illustration of Dahl et al. 2012 by Lane McIntosh, copyright CS231n 2017

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, Justin Johnson, Serena Yeung

A bit of history:

Hubel & Wiesel,

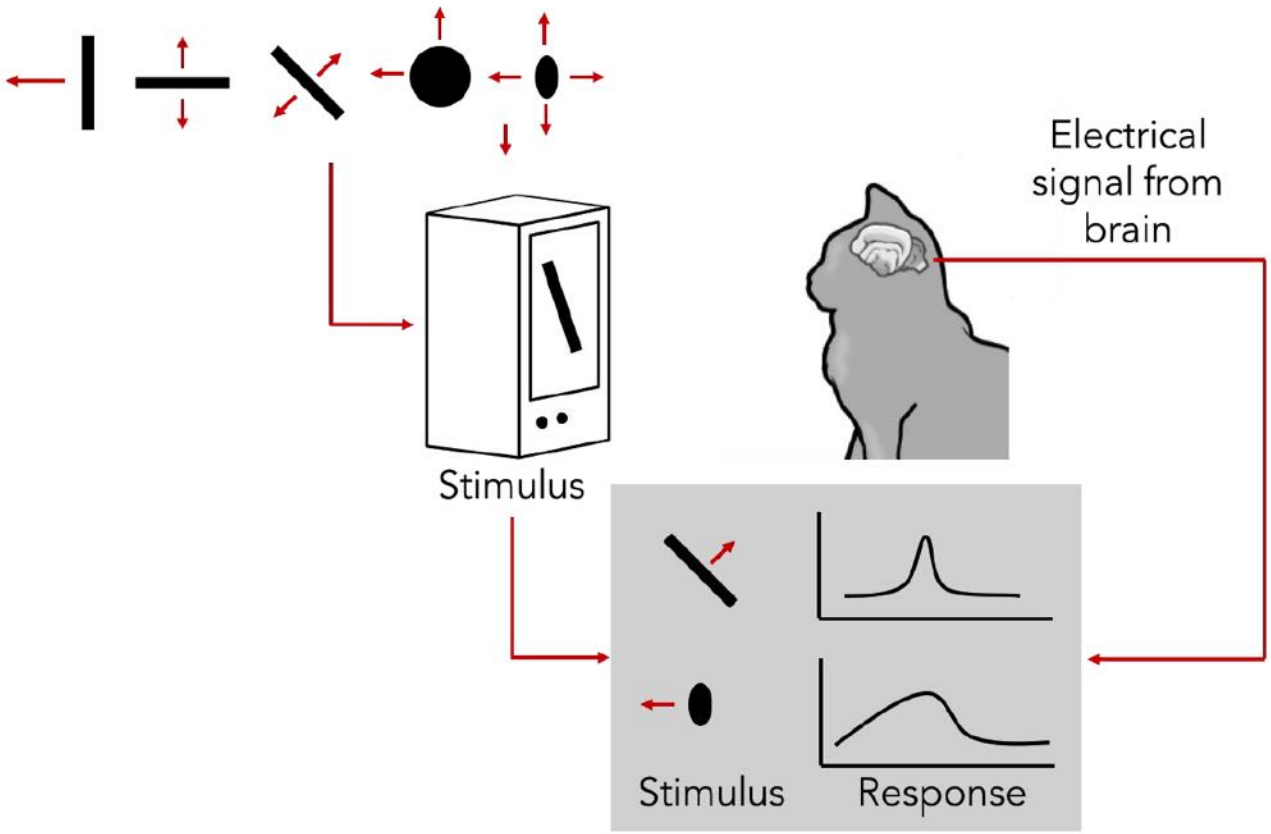
1959

RECEPTIVE FIELDS OF SINGLE NEURONES IN THE CAT'S STRIATE CORTEX

1962

RECEPTIVE FIELDS, BINOCULAR INTERACTION AND FUNCTIONAL ARCHITECTURE IN THE CAT'S VISUAL CORTEX

1968…



slide credit: Fei-Fei, Justin Johnson, Serena Yeung

Hierarchical organization

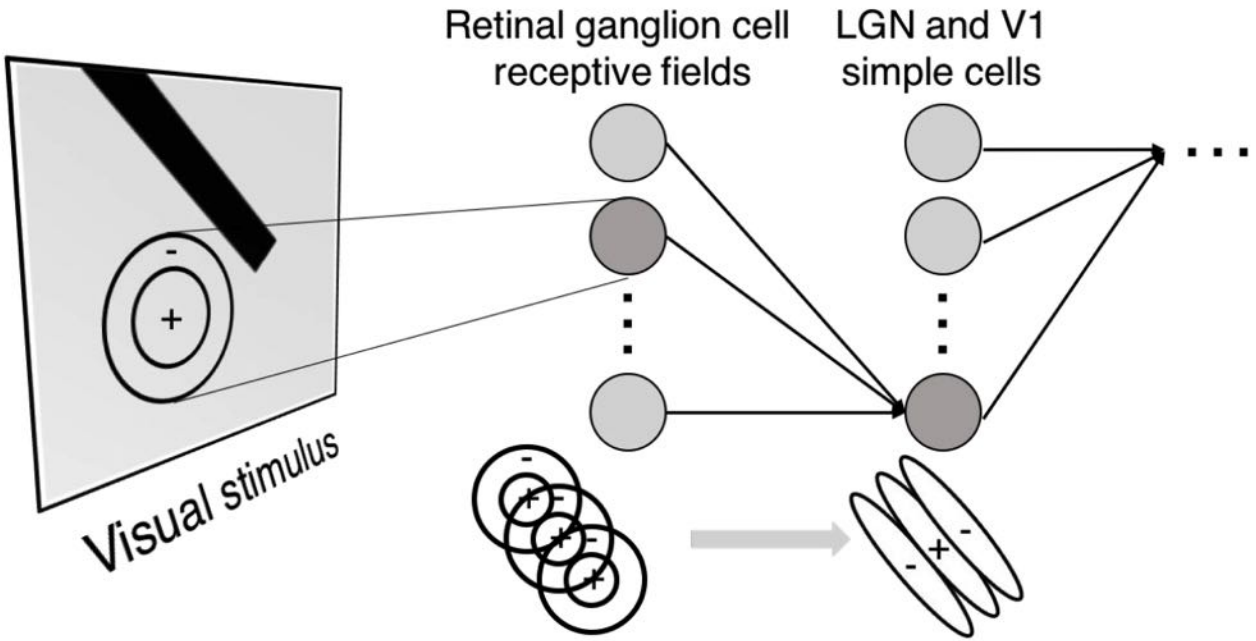
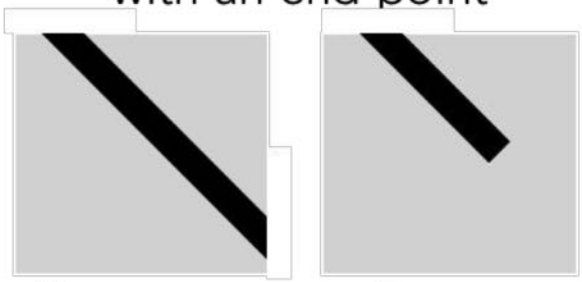


Illustration of hierarchical organization in early visual pathways by Lane McIntosh, copyright CS231n 2017

Simple cells:
Response to light orientation

Complex cells:
Response to light orientation and movement

Hypercomplex cells:
response to movement with an end point



No response

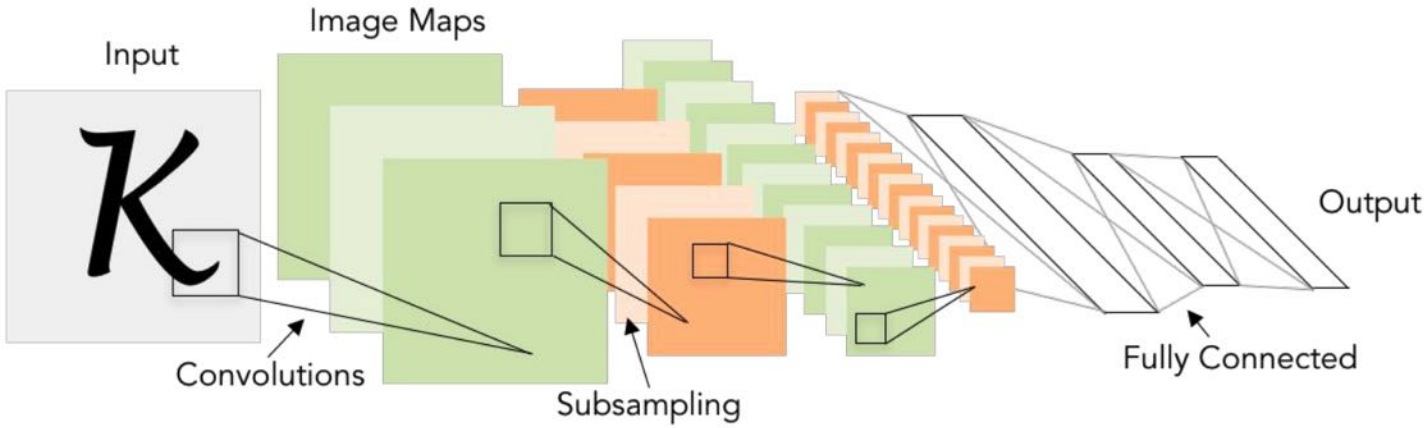
Response (end point)

slide credit: Fei-Fei, Justin Johnson, Serena Yeung

A bit of history:

Gradient-based learning applied to document recognition

[LeCun, Bottou, Bengio, Haffner 1998]



LeNet-5

slide credit: Fei-Fei, Justin Johnson, Serena Yeung

A bit of history:

ImageNet Classification with Deep Convolutional Neural Networks

[Krizhevsky, Sutskever, Hinton, 2012]

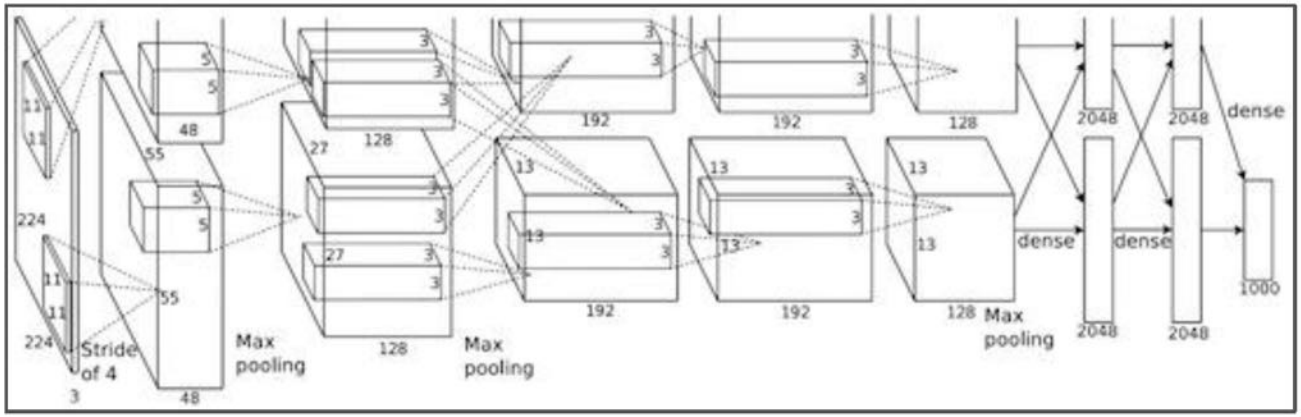
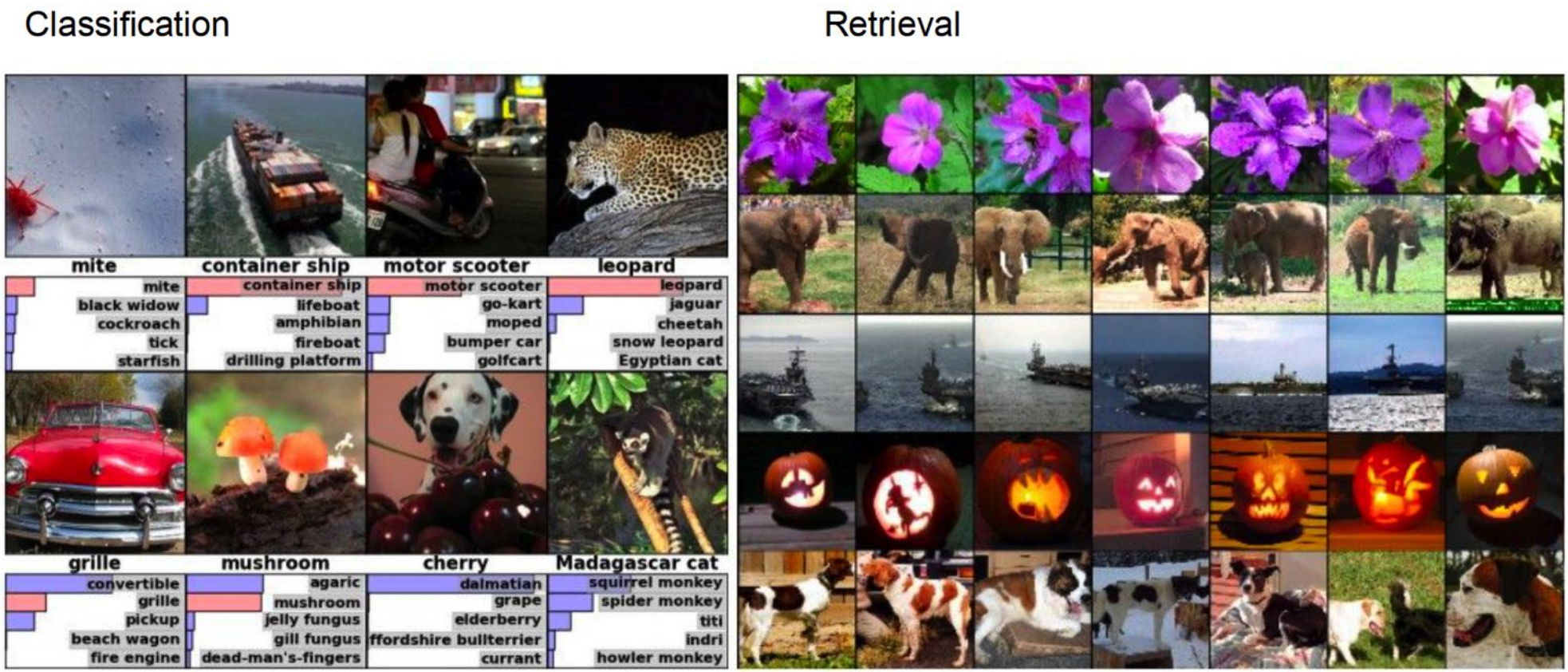


Figure copyright Alex Krizhevsky, Ilya Sutskever, and Geoffrey Hinton, 2012. Reproduced with permission.

"AlexNet"

slide credit: Fei-Fei, Justin Johnson, Serena Yeung

Fast-forward to today: ConvNets are everywhere

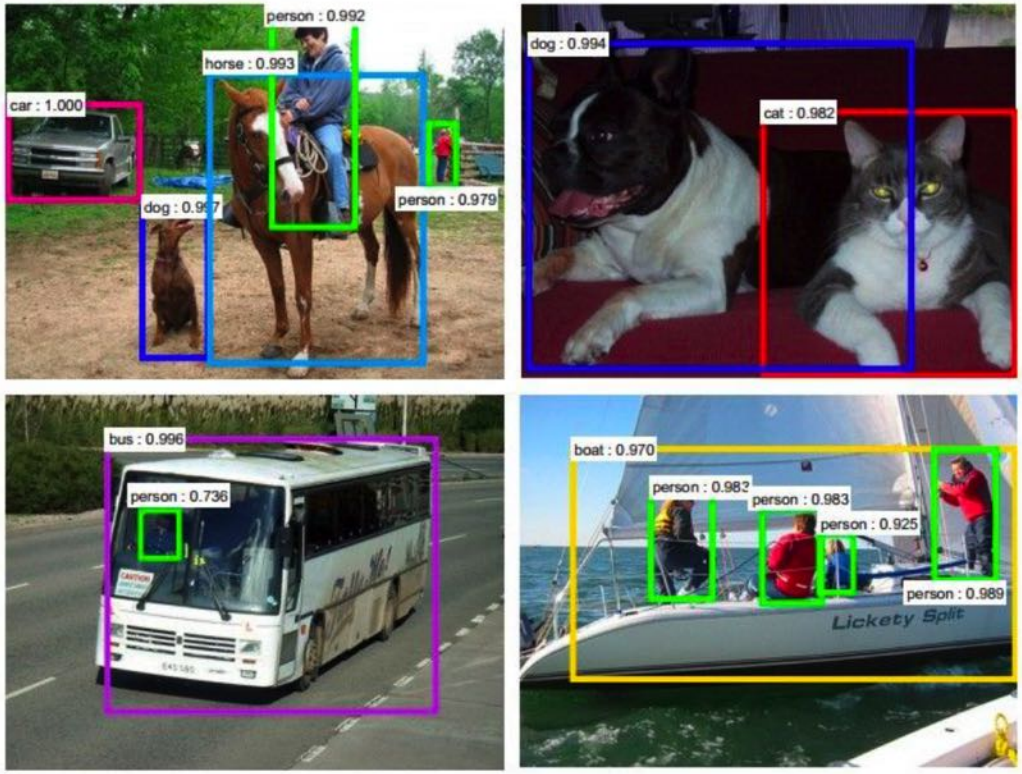


Figures copyright Alex Krizhevsky, Ilya Sutskever, and Geoffrey Hinton, 2012. Reproduced with permission.

slide credit: Fei-Fei, Justin Johnson, Serena Yeung

Fast-forward to today: ConvNets are everywhere

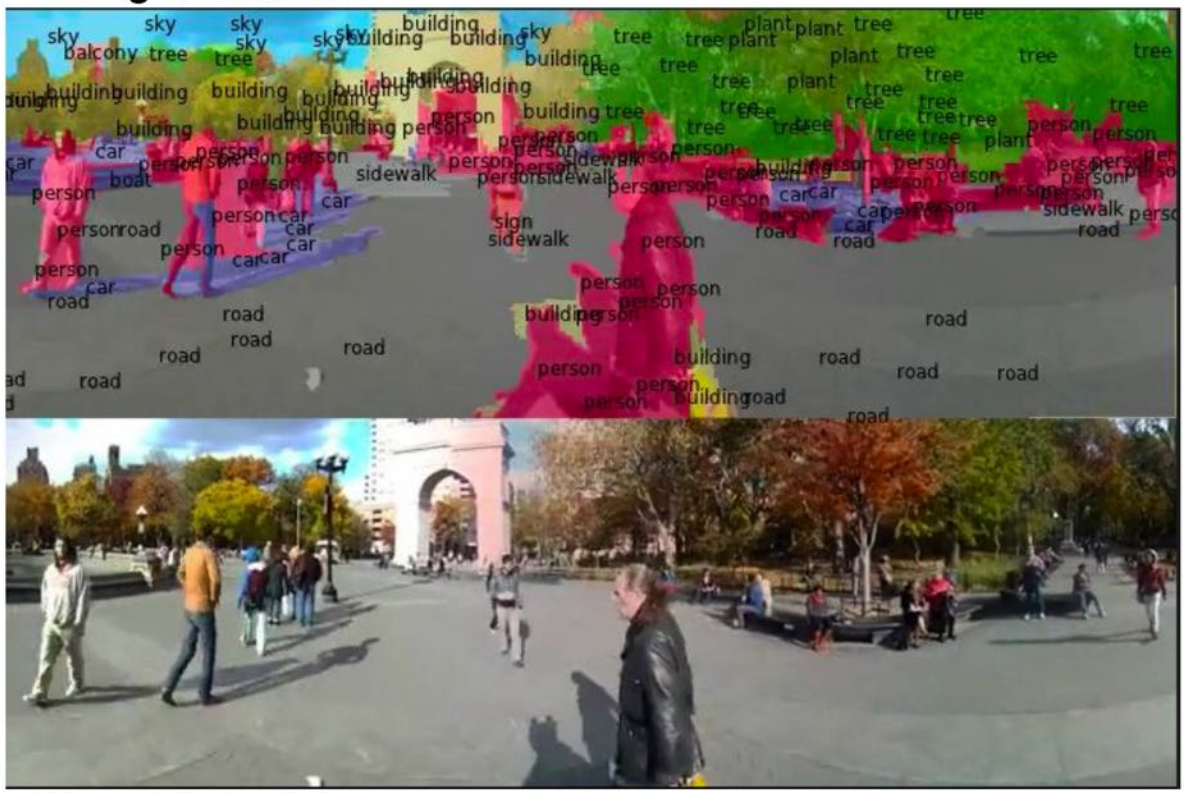
Detection



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, Justin Johnson, Serena Yeung

Fast-forward to today: ConvNets are everywhere

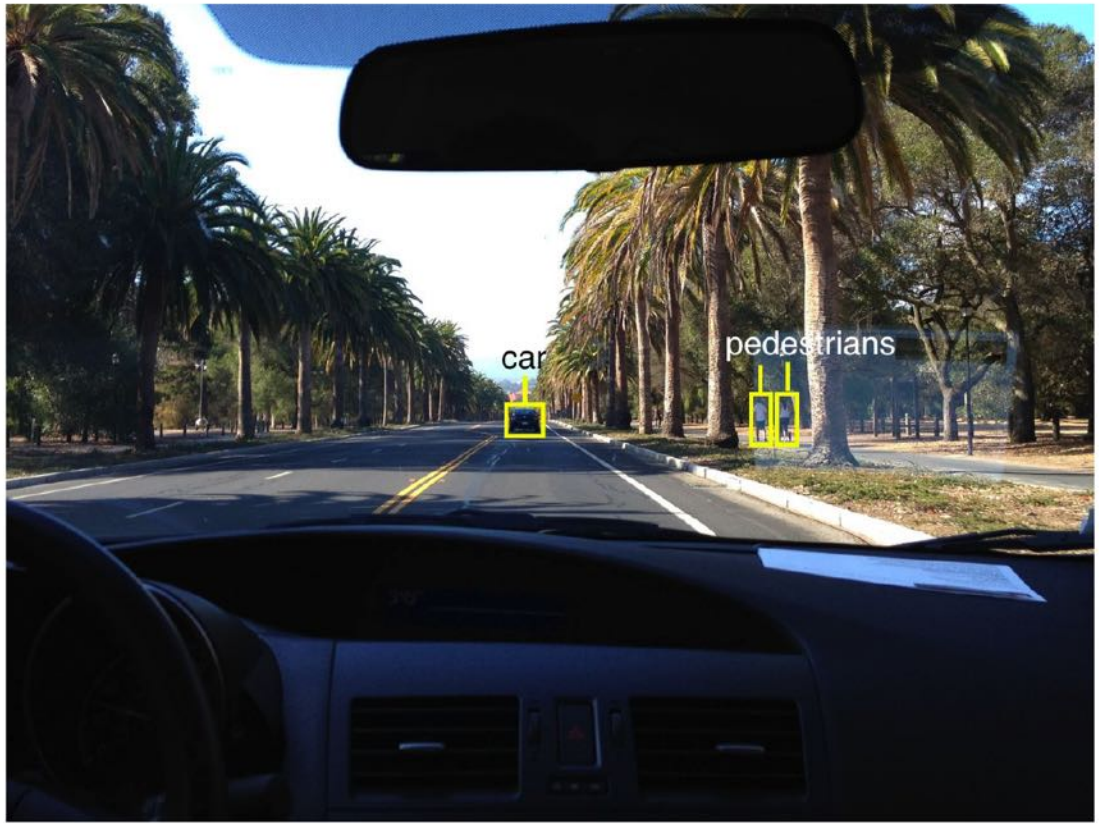


Photo by Lane McIntosh. Copyright CS231n 2017.

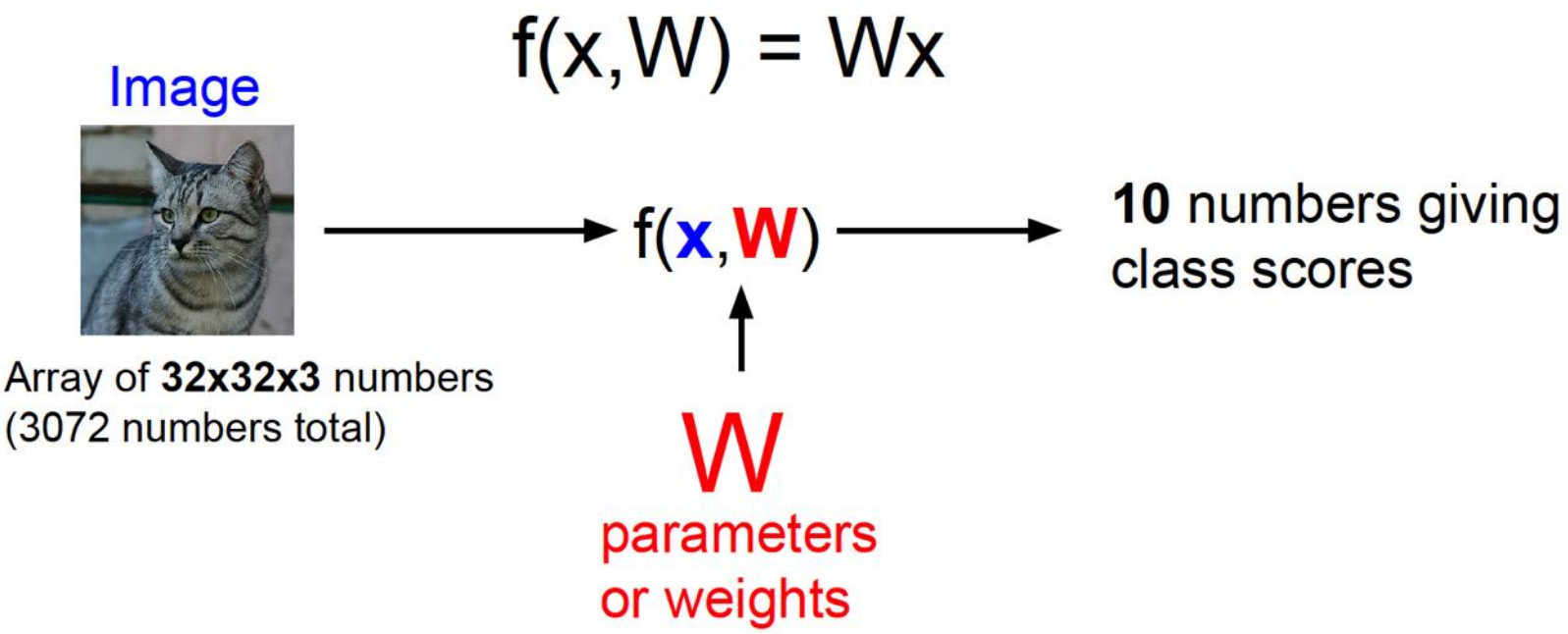
self-driving cars

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, Justin Johnson, Serena Yeung

Parametric Approach: Linear Classifier



slide credit: Fei-Fei, Justin Johnson, Serena Yeung

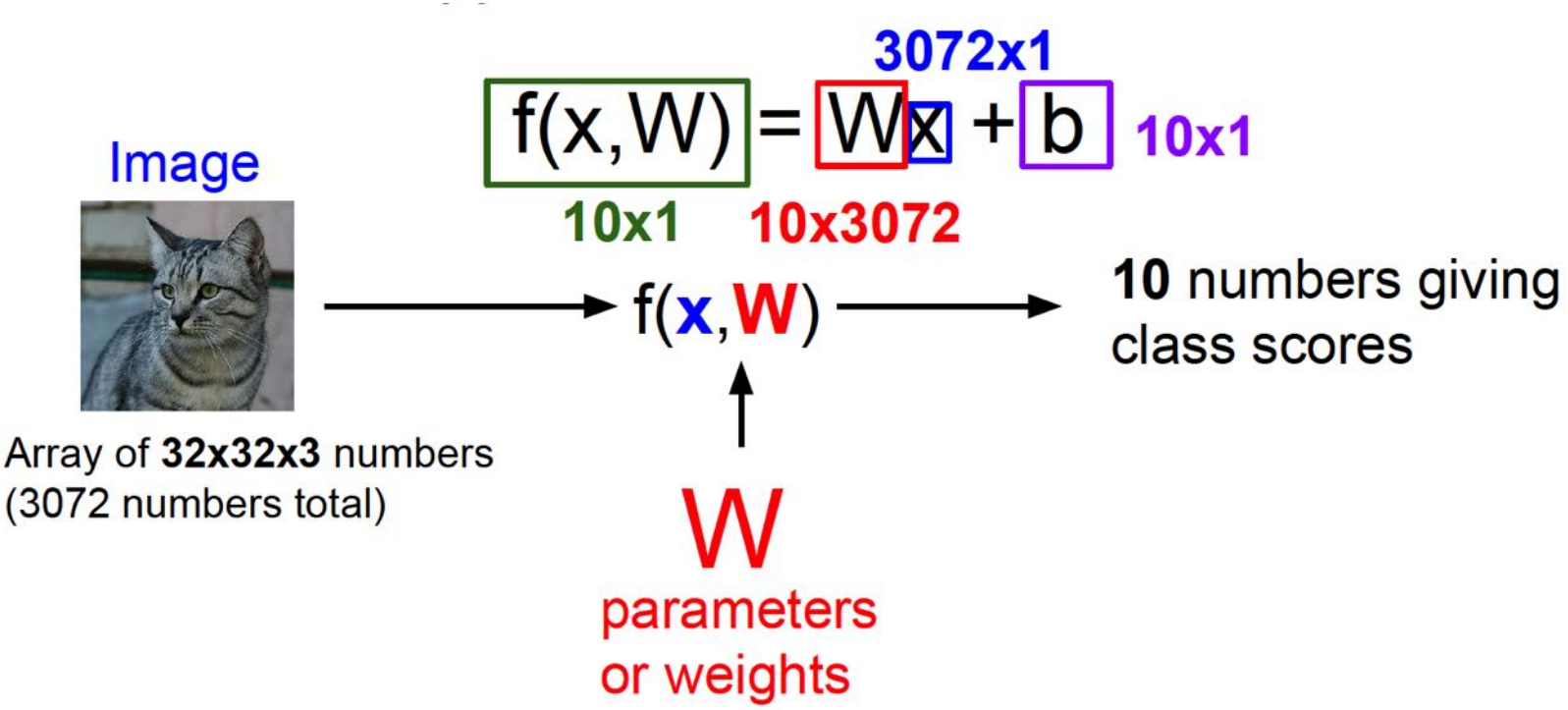
Recall CIFAR10

50,000 training images
each image is **32x32x3**

10,000 test images.

slide credit: Fei-Fei, Justin Johnson, Serena Yeung

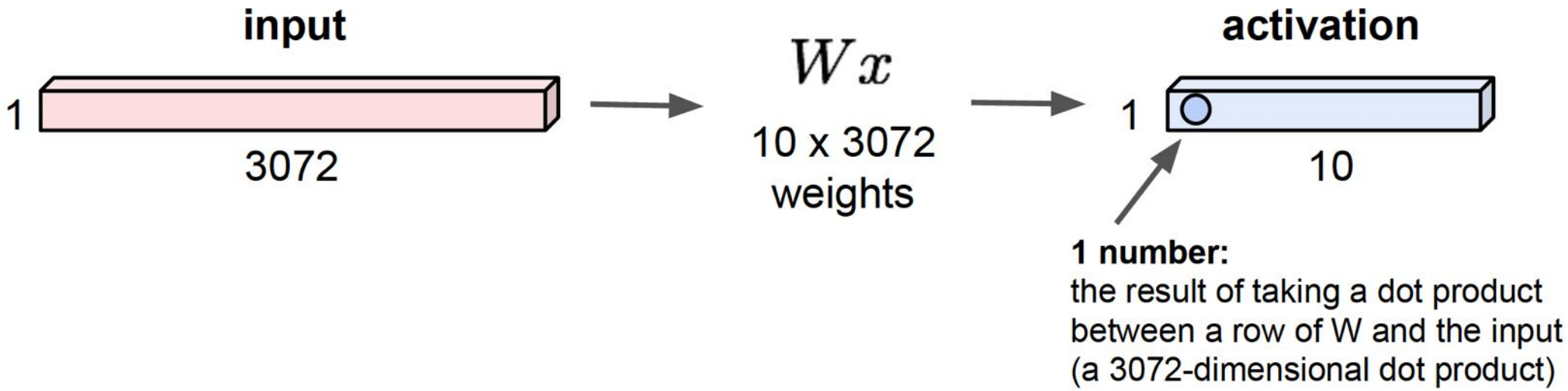
Parametric Approach: Linear Classifier



slide credit: Fei-Fei, Justin Johnson, Serena Yeung

Fully Connected Layer

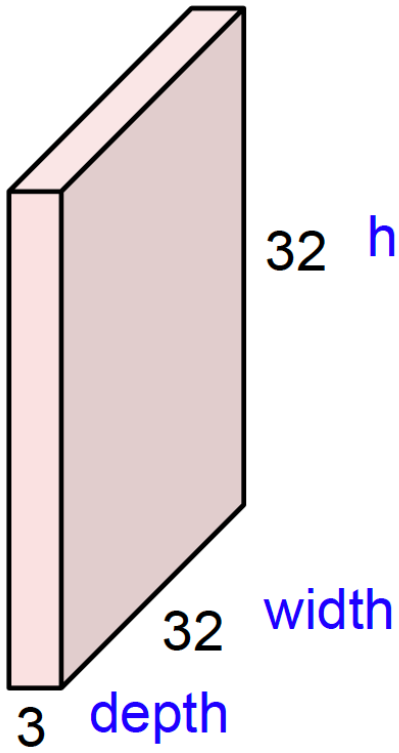
32x32x3 image -> stretch to 3072 x 1



slide credit: Fei-Fei, Justin Johnson, Serena Yeung

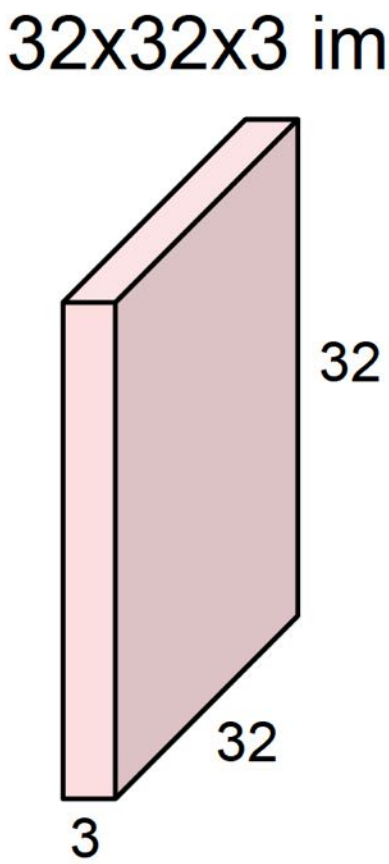
Convolution Layer

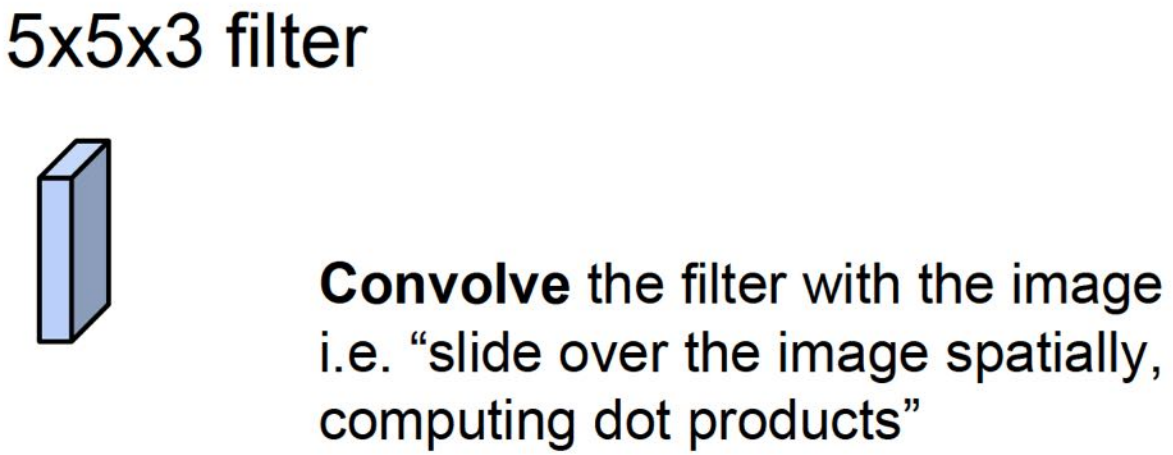
32x32x3 image -> preserve spatial structure



slide credit: Fei-Fei, Justin Johnson, Serena Yeung

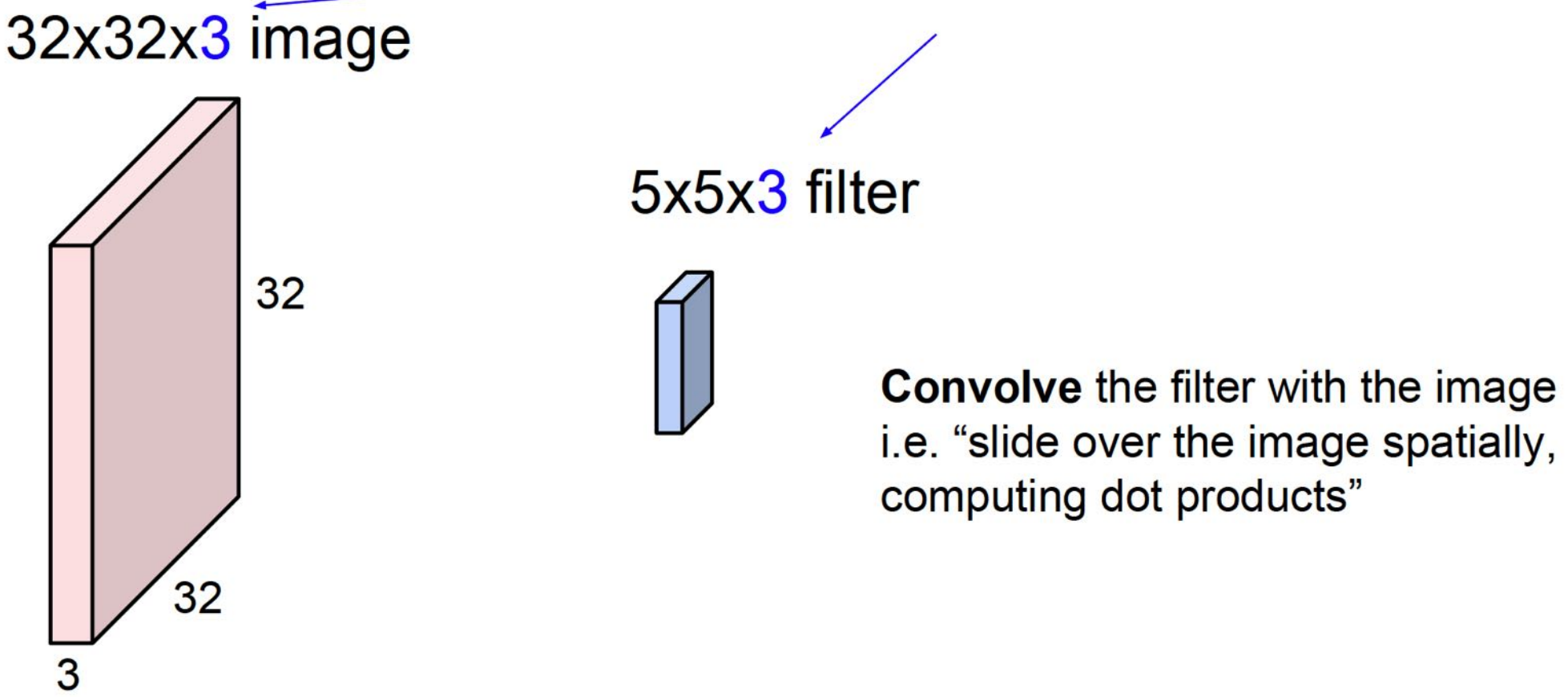
Convolution Layer





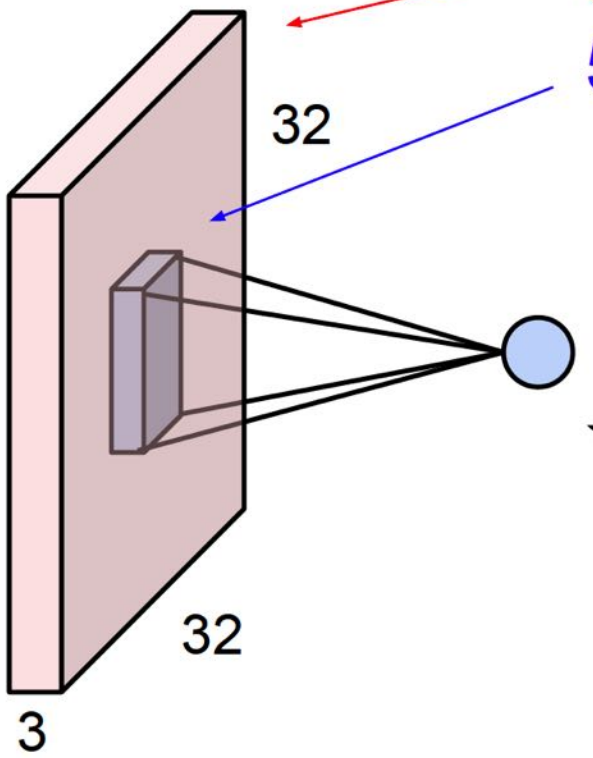
slide credit: Fei-Fei, Justin Johnson, Serena Yeung

Convolution Layer



slide credit: Fei-Fei, Justin Johnson, Serena Yeung

Convolution Layer



32x32x3 image

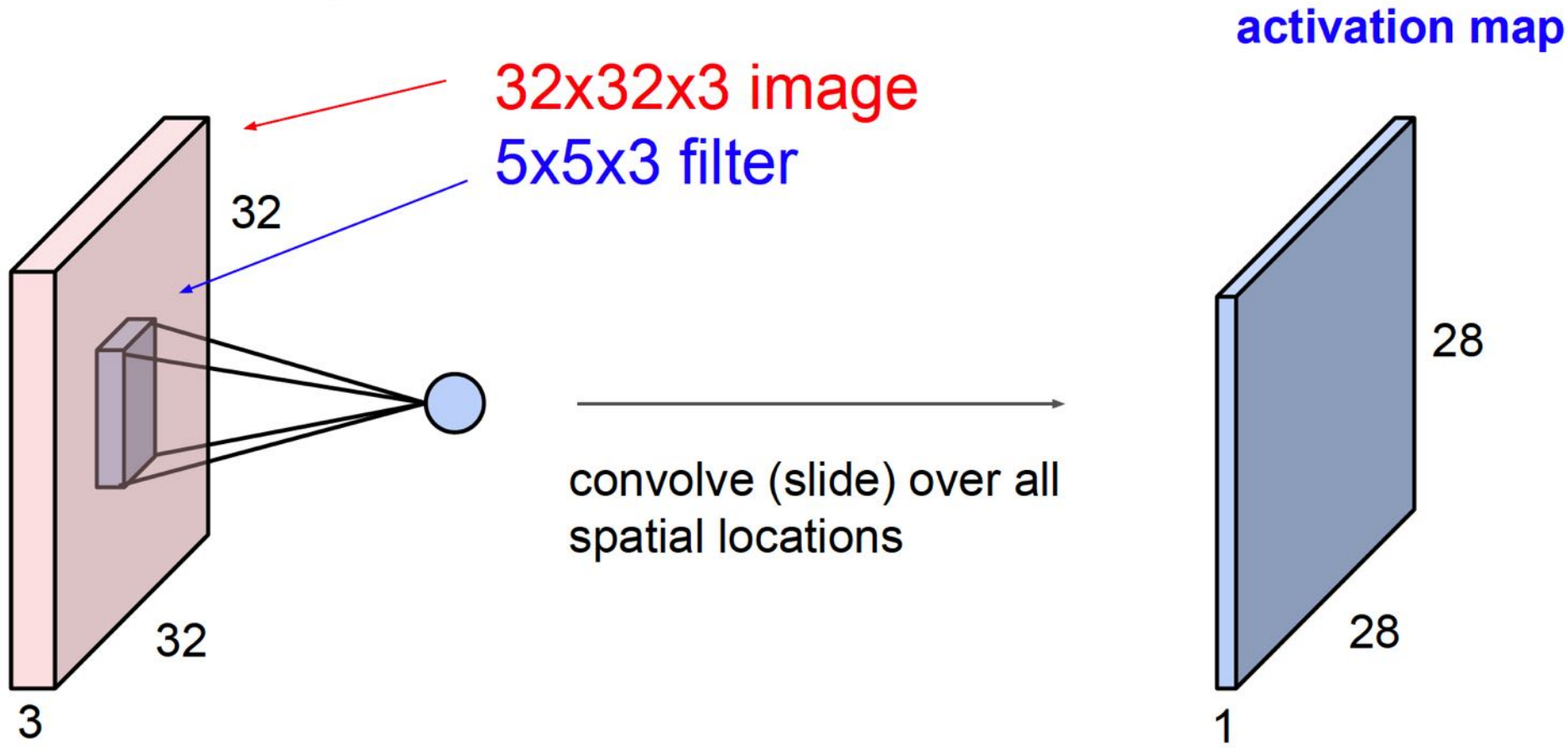
5x5x3 filter w

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, Justin Johnson, Serena Yeung

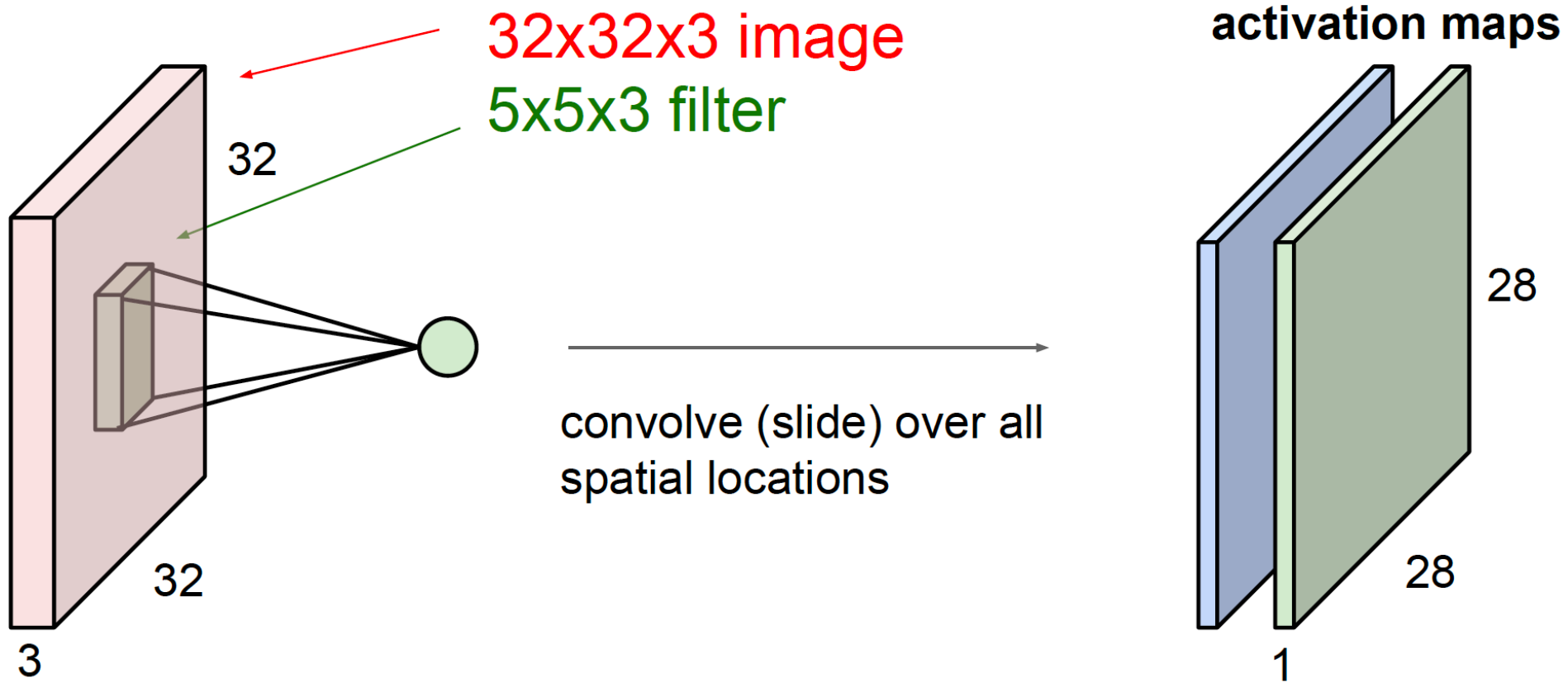
Convolution Layer



slide credit: Fei-Fei, Justin Johnson, Serena Yeung

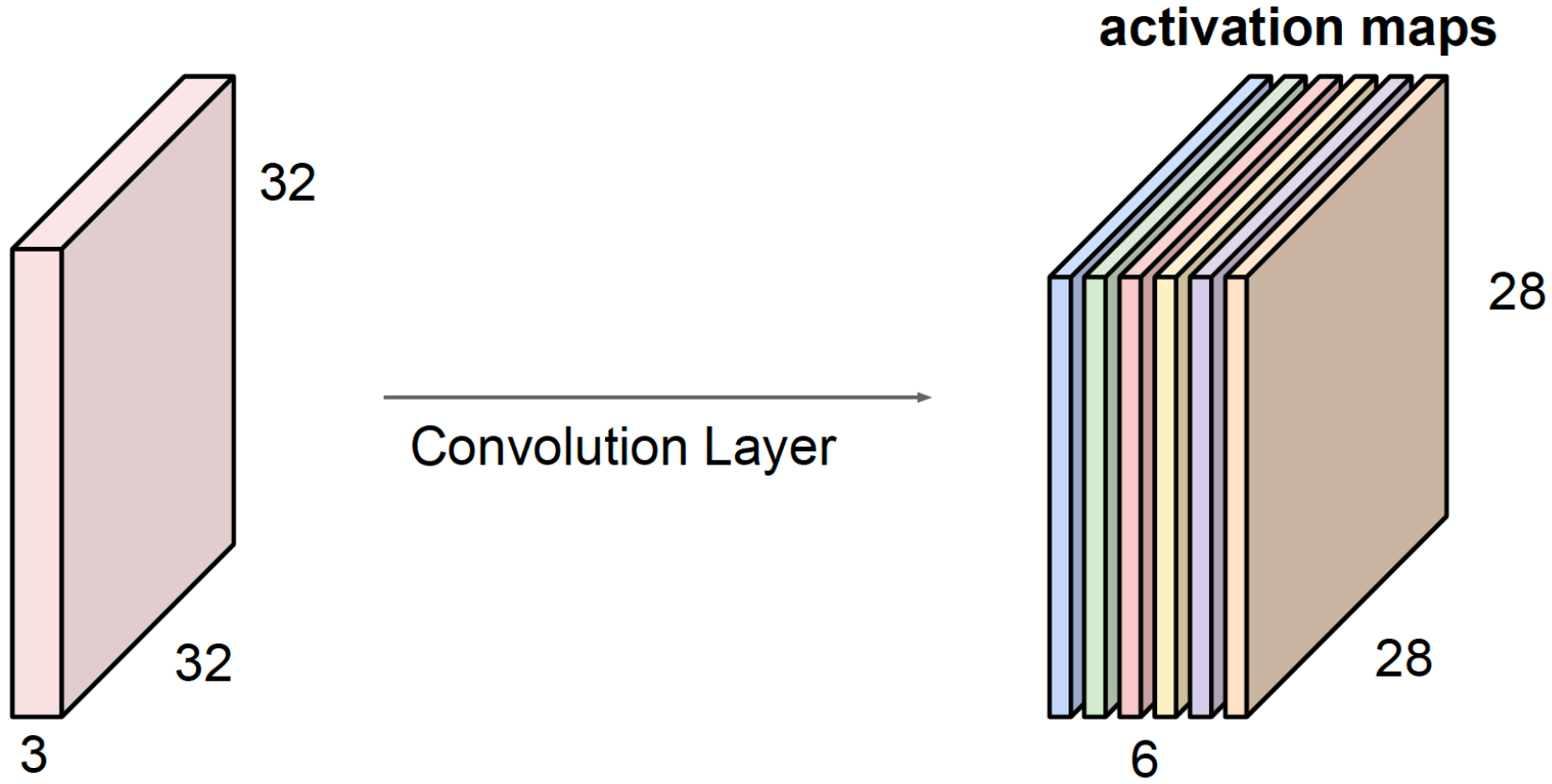
Convolution Layer

consider a second, green filter



slide credit: Fei-Fei, Justin Johnson, Serena Yeung

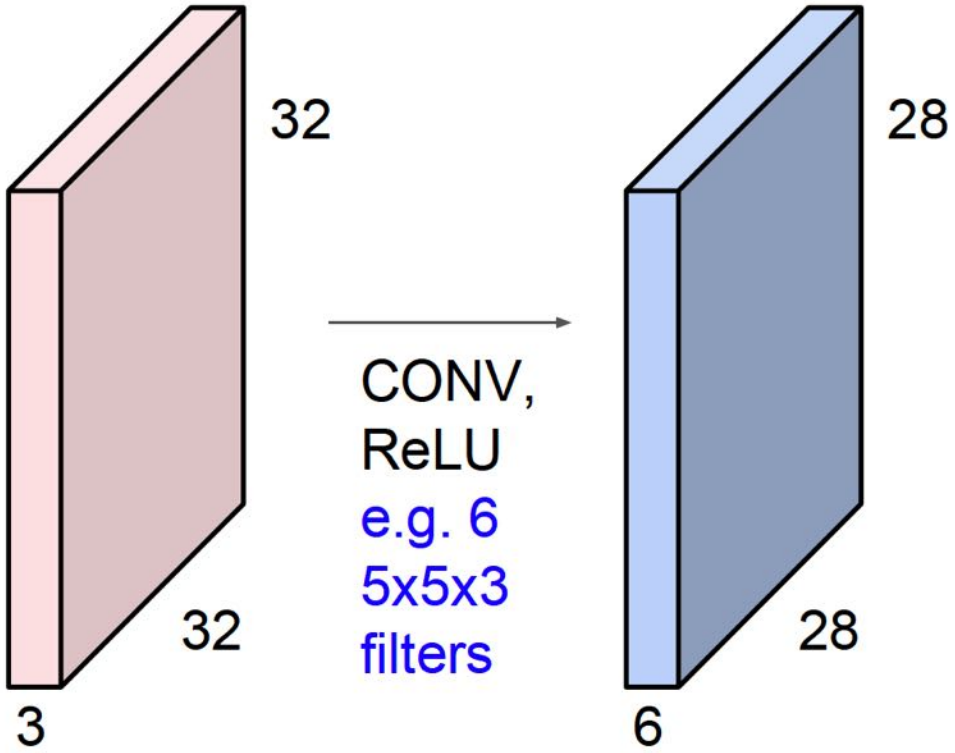
For example, if we had 6 5x5 filters, we'll get 6 separate activation maps:



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

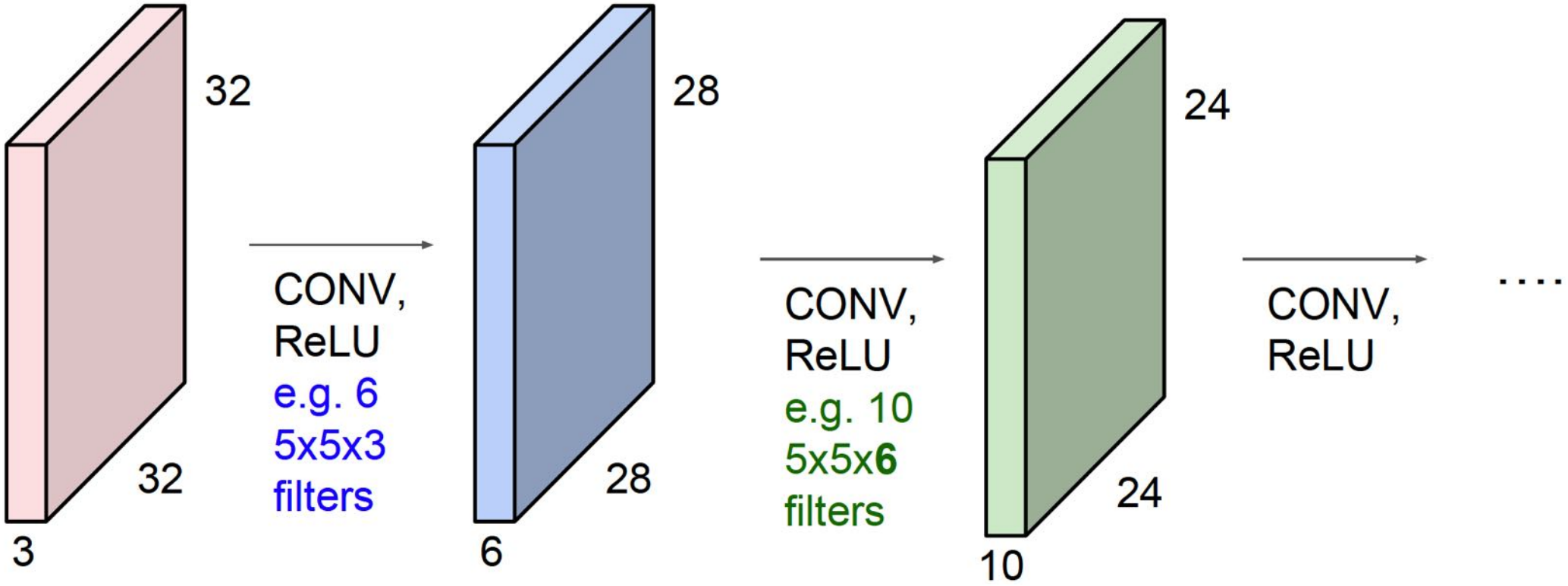
slide credit: Fei-Fei, Justin Johnson, Serena Yeung

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



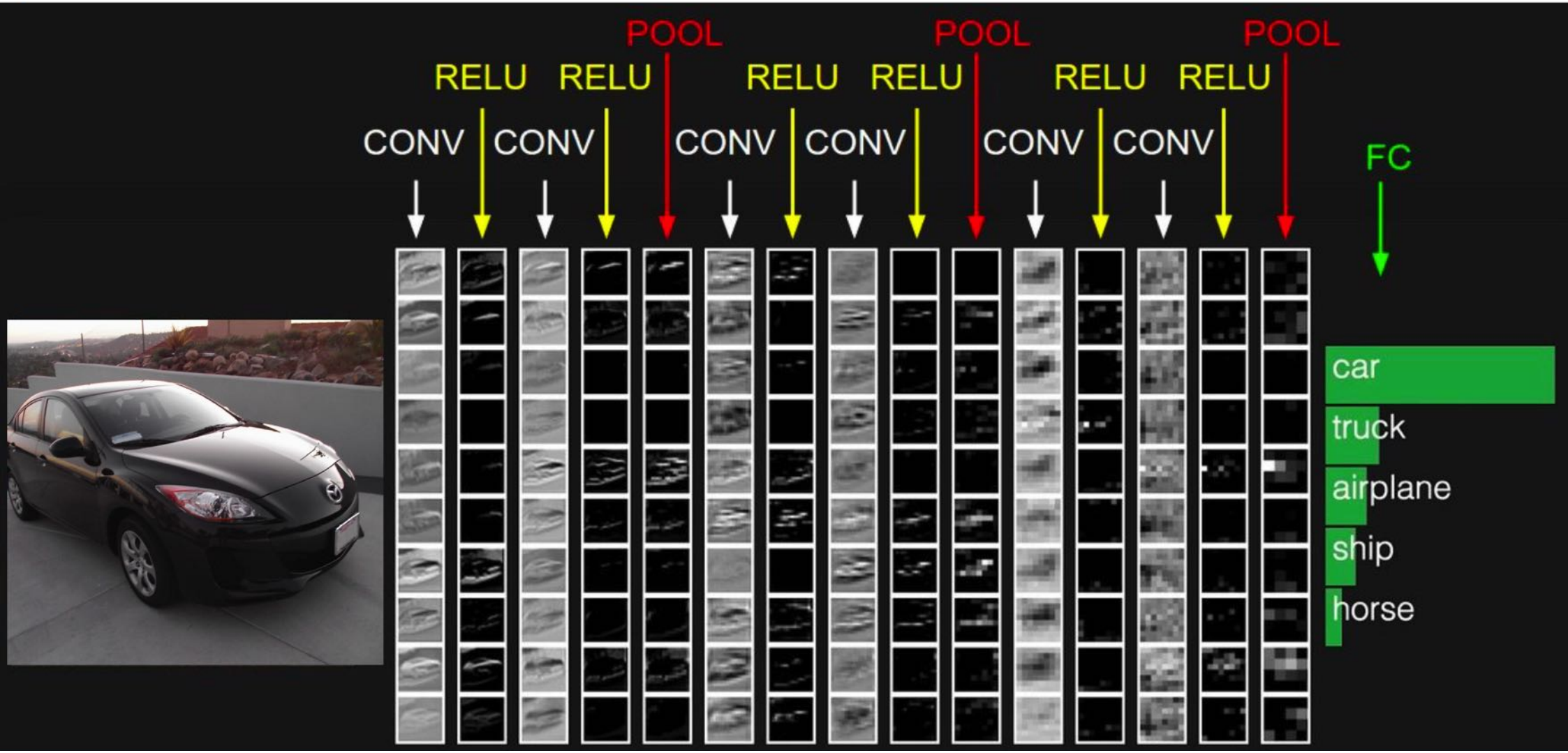
slide credit: Fei-Fei, Justin Johnson, Serena Yeung

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



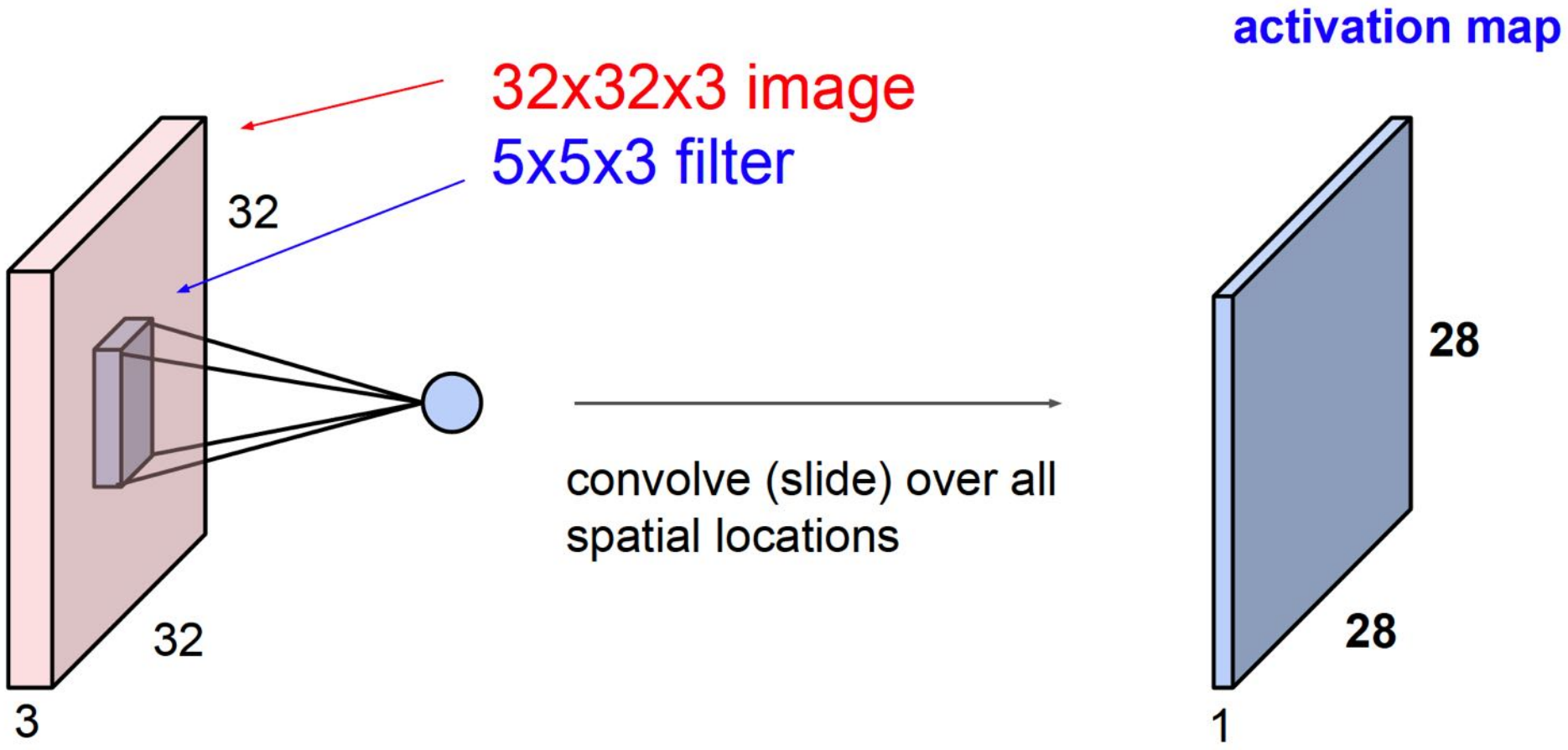
slide credit: Fei-Fei, Justin Johnson, Serena Yeung

preview:



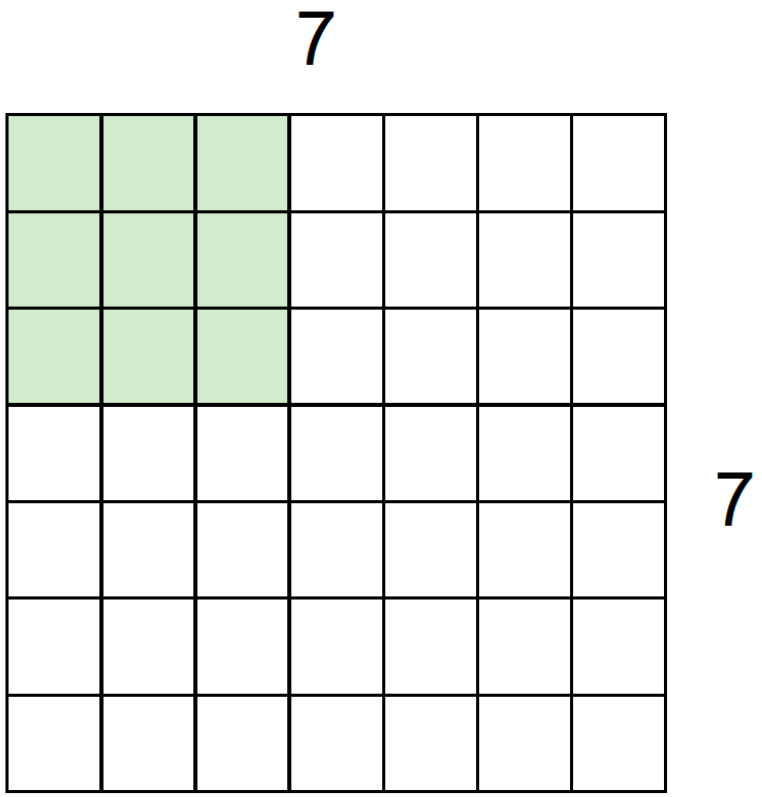
slide credit: Fei-Fei, Justin Johnson, Serena Yeung

A closer look at spatial dimensions:



slide credit: Fei-Fei, Justin Johnson, Serena Yeung

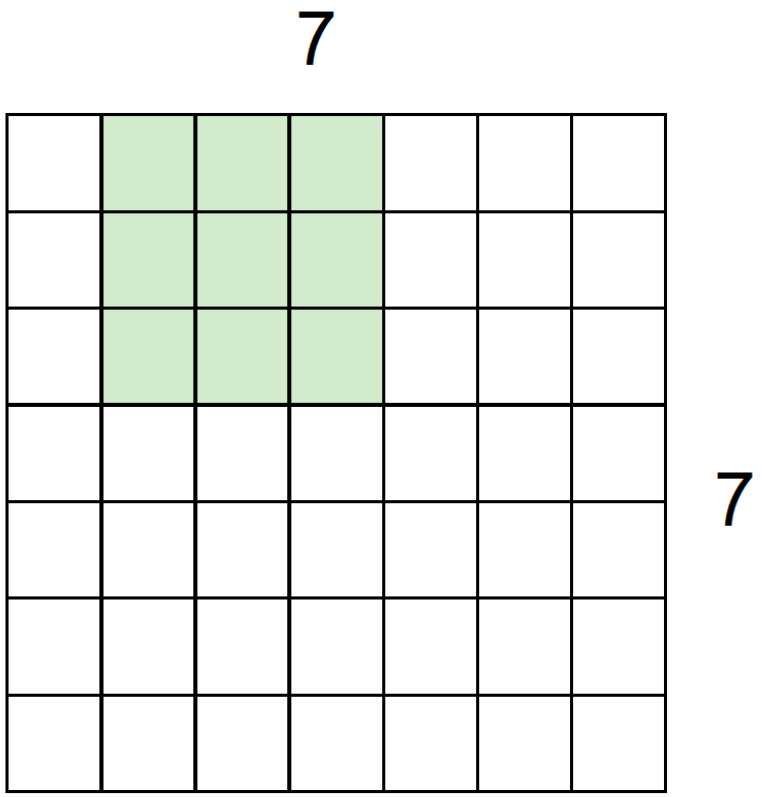
A closer look at spatial dimensions:



7x7 input (spatially)
assume 3x3 filter

slide credit: Fei-Fei, Justin Johnson, Serena Yeung

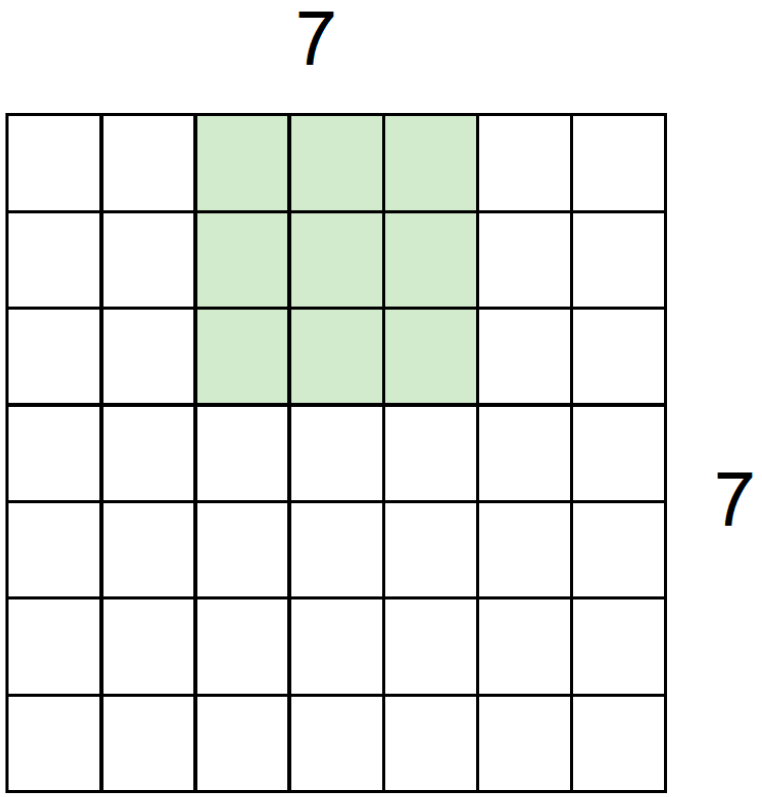
A closer look at spatial dimensions:



7x7 input (spatially)
assume 3x3 filter

slide credit: Fei-Fei, Justin Johnson, Serena Yeung

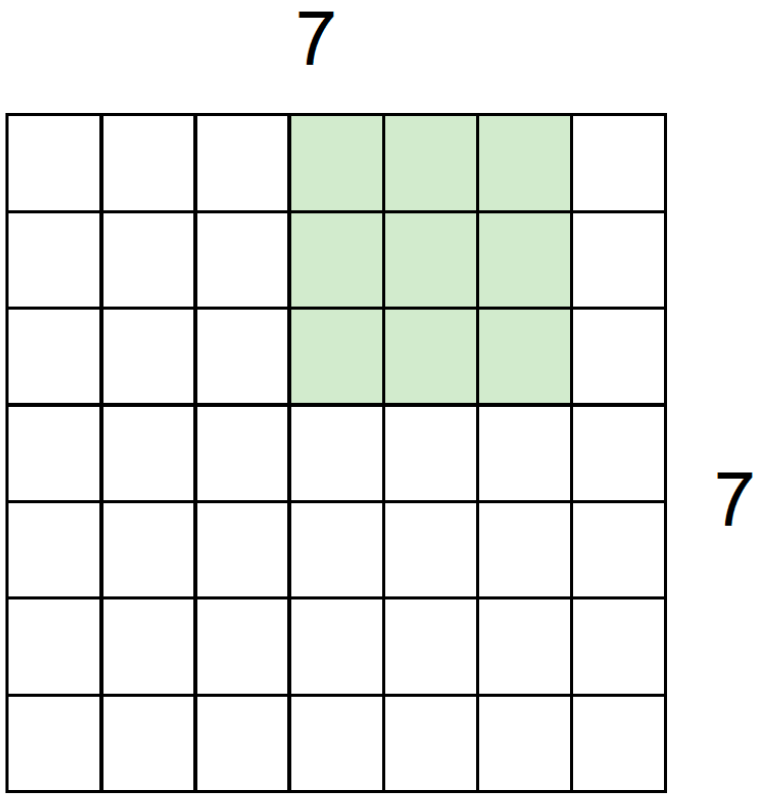
A closer look at spatial dimensions:



7x7 input (spatially)
assume 3x3 filter

slide credit: Fei-Fei, Justin Johnson, Serena Yeung

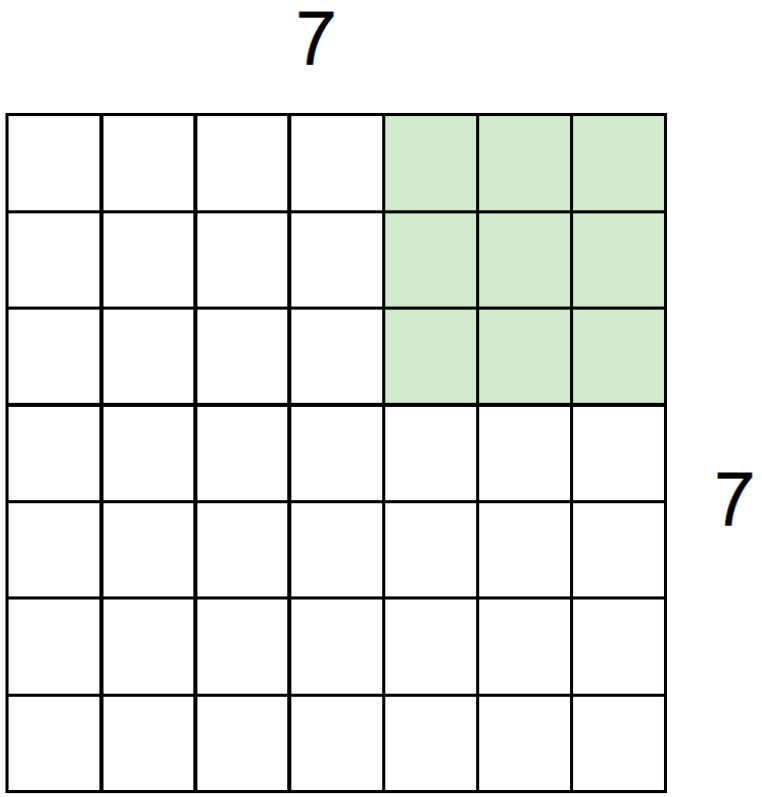
A closer look at spatial dimensions:



7x7 input (spatially)
assume 3x3 filter

slide credit: Fei-Fei, Justin Johnson, Serena Yeung

A closer look at spatial dimensions:

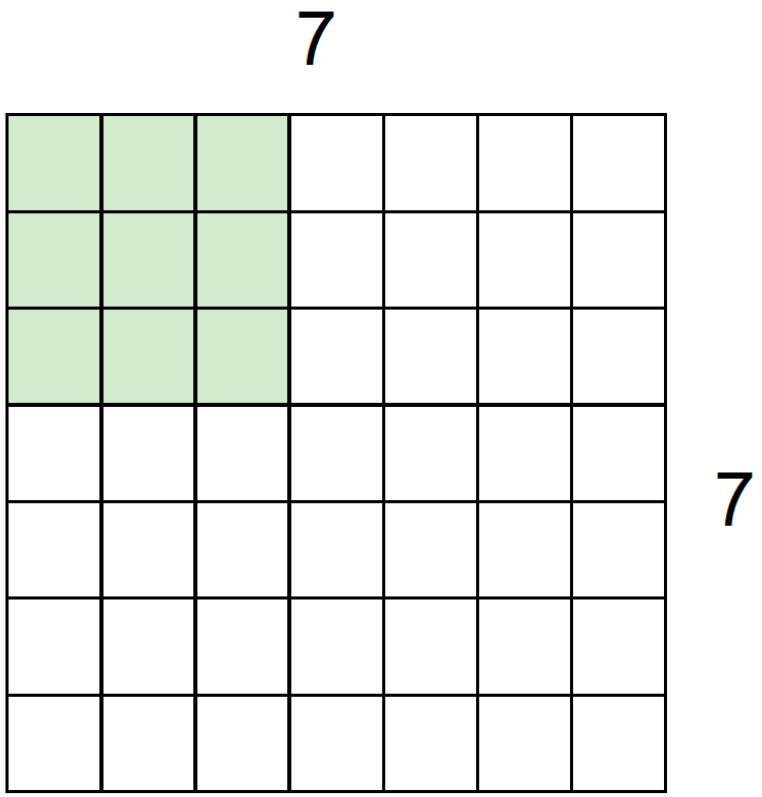


7x7 input (spatially)
assume 3x3 filter

=> 5x5 output

slide credit: Fei-Fei, Justin Johnson, Serena Yeung

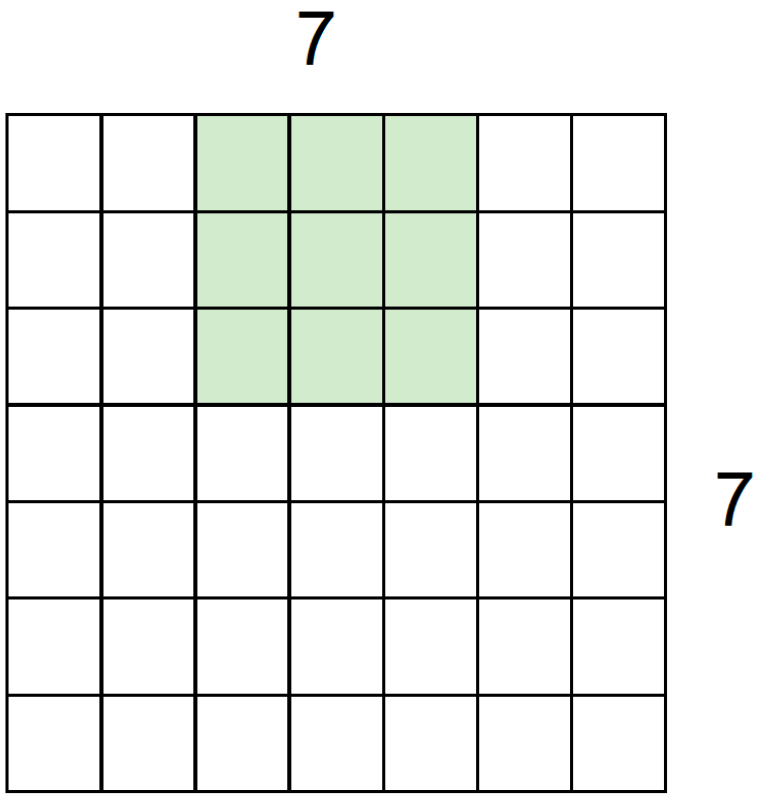
A closer look at spatial dimensions:



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

slide credit: Fei-Fei, Justin Johnson, Serena Yeung

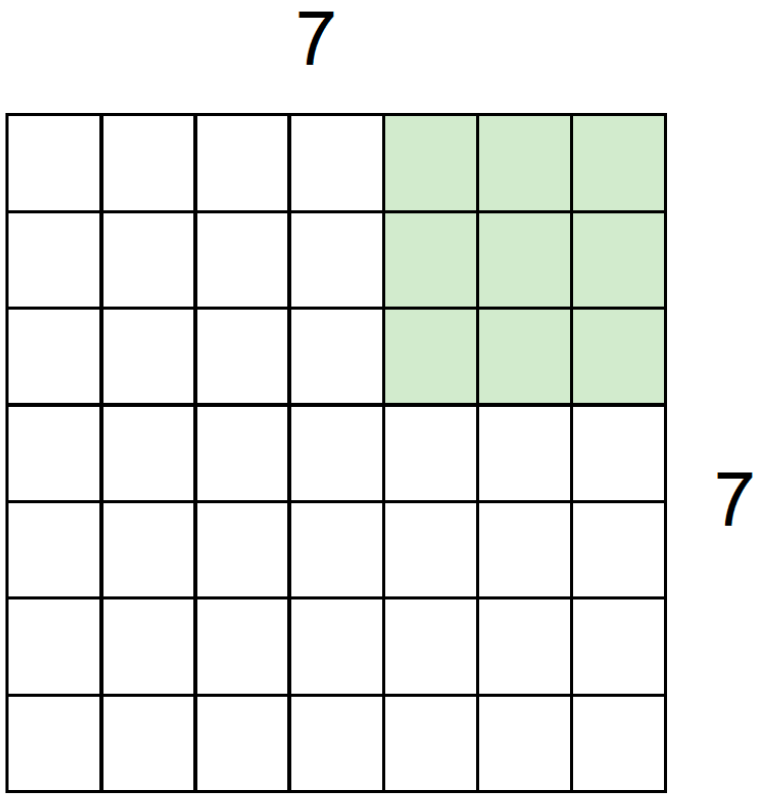
A closer look at spatial dimensions:



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

slide credit: Fei-Fei, Justin Johnson, Serena Yeung

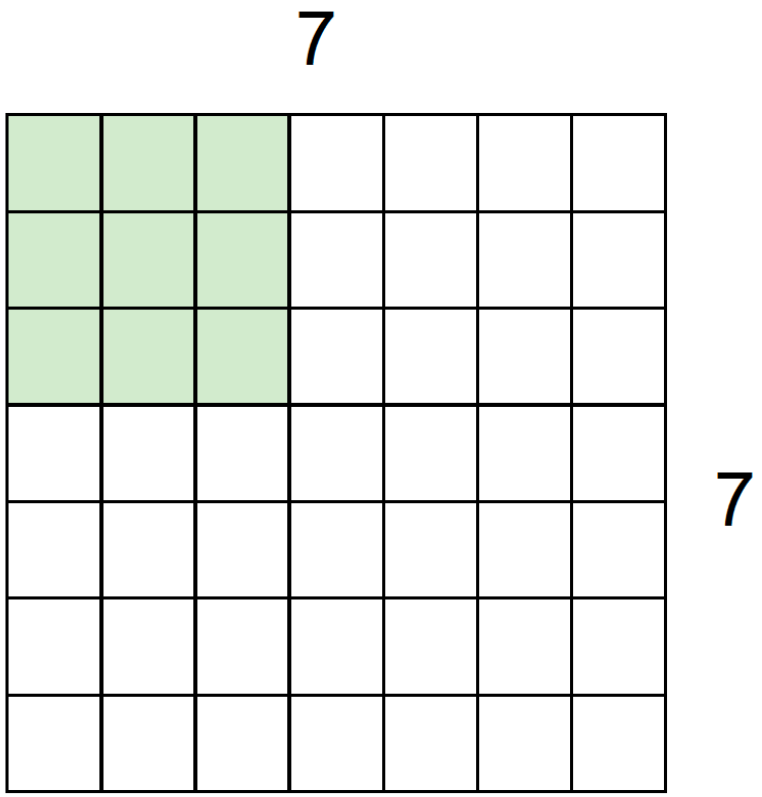
A closer look at spatial dimensions:



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

slide credit: Fei-Fei, Justin Johnson, Serena Yeung

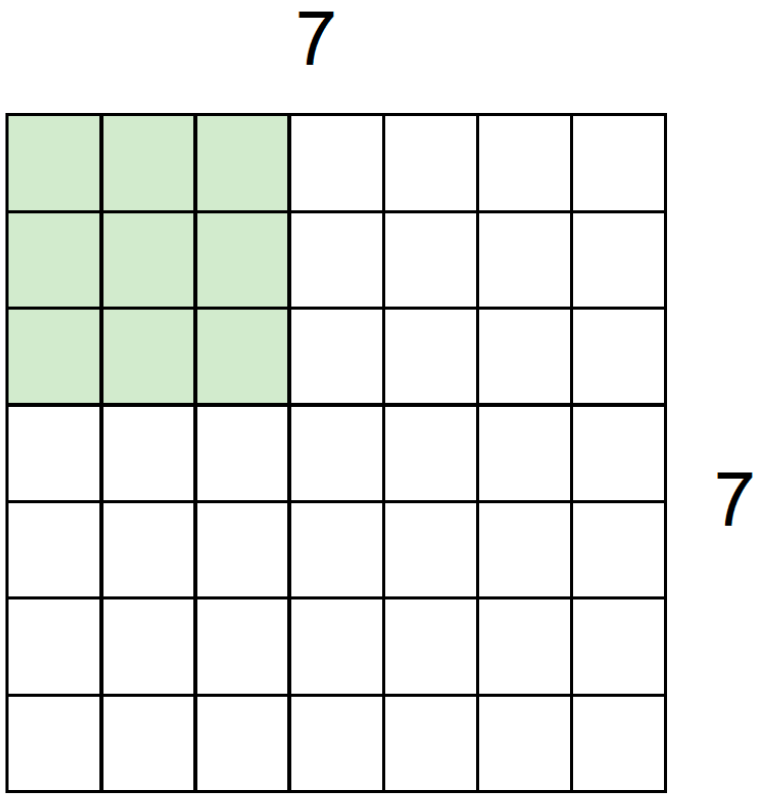
A closer look at spatial dimensions:



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

slide credit: Fei-Fei, Justin Johnson, Serena Yeung

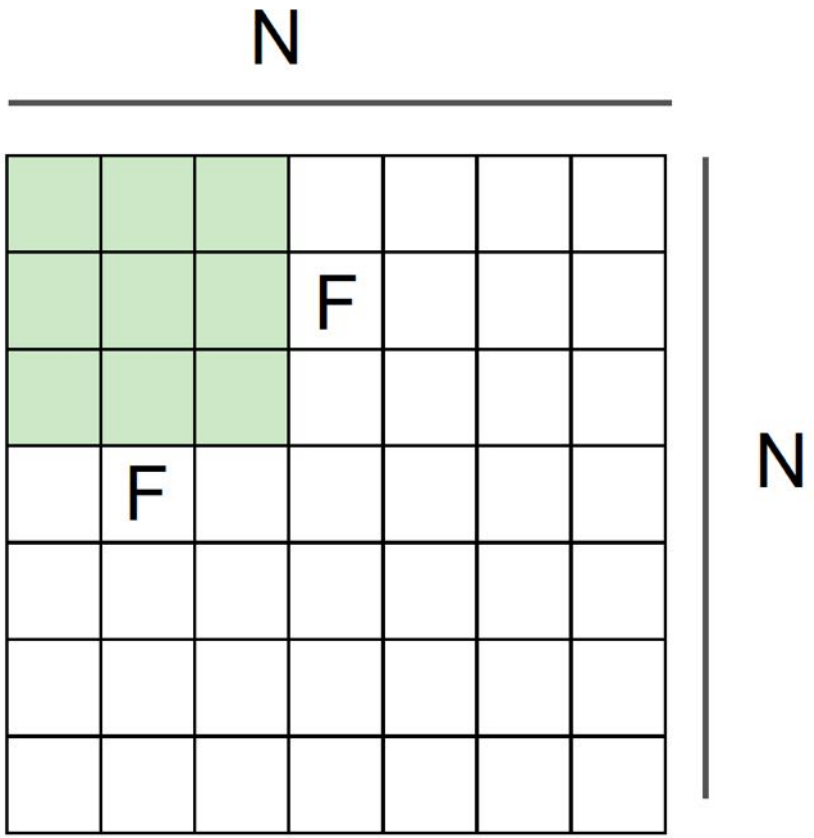
A closer look at spatial dimensions:



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, Justin Johnson, Serena Yeung

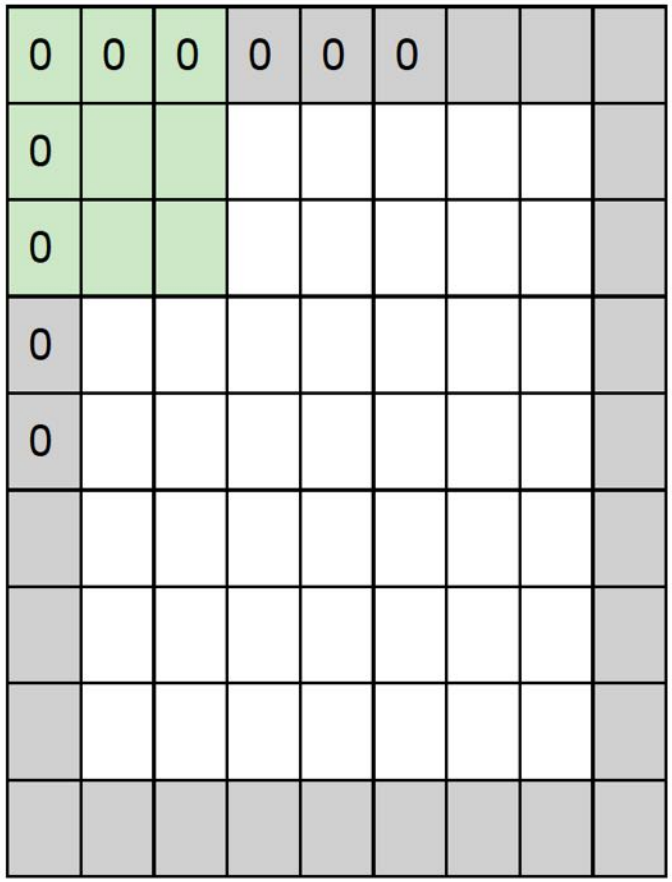


Output size:
(N - F) / stride + 1

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, Justin Johnson, Serena Yeung

In practice: Common to zero pad the border

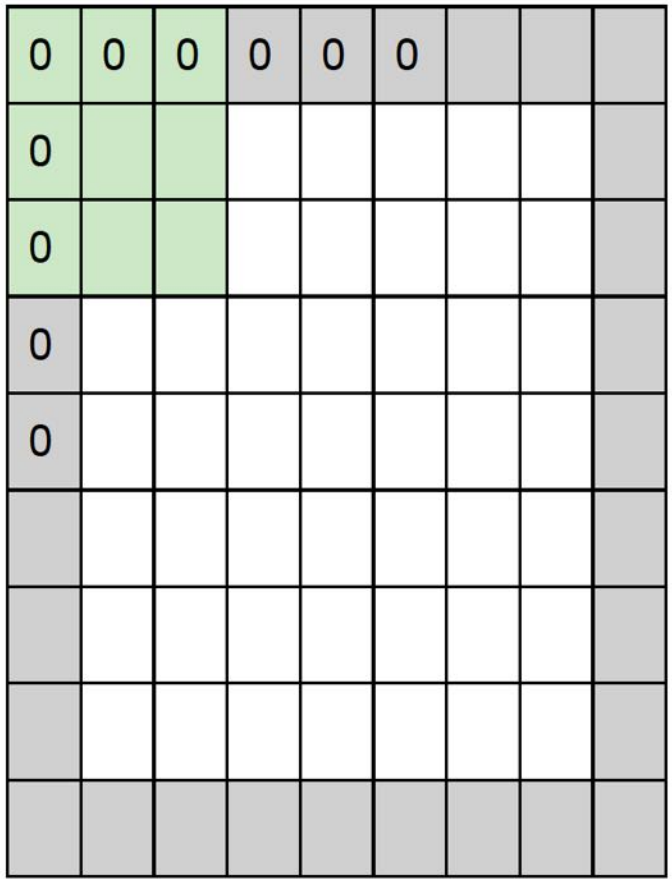


e.g. input 7x7
3x3 filter, applied with **stride 1**
pad with 1 pixel border => what is the output?

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

slide credit: Fei-Fei, Justin Johnson, Serena Yeung

In practice: Common to zero pad the border

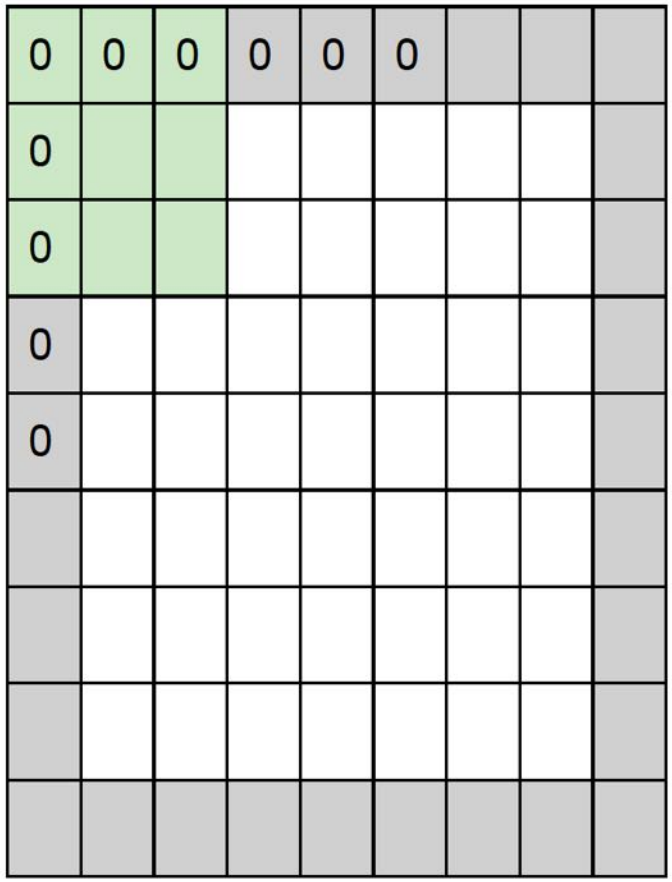


e.g. input 7x7
3x3 filter, applied with **stride 1**
pad with 1 pixel border => what is the output?

7x7 output!

slide credit: Fei-Fei, Justin Johnson, Serena Yeung

In practice: Common to zero pad the border



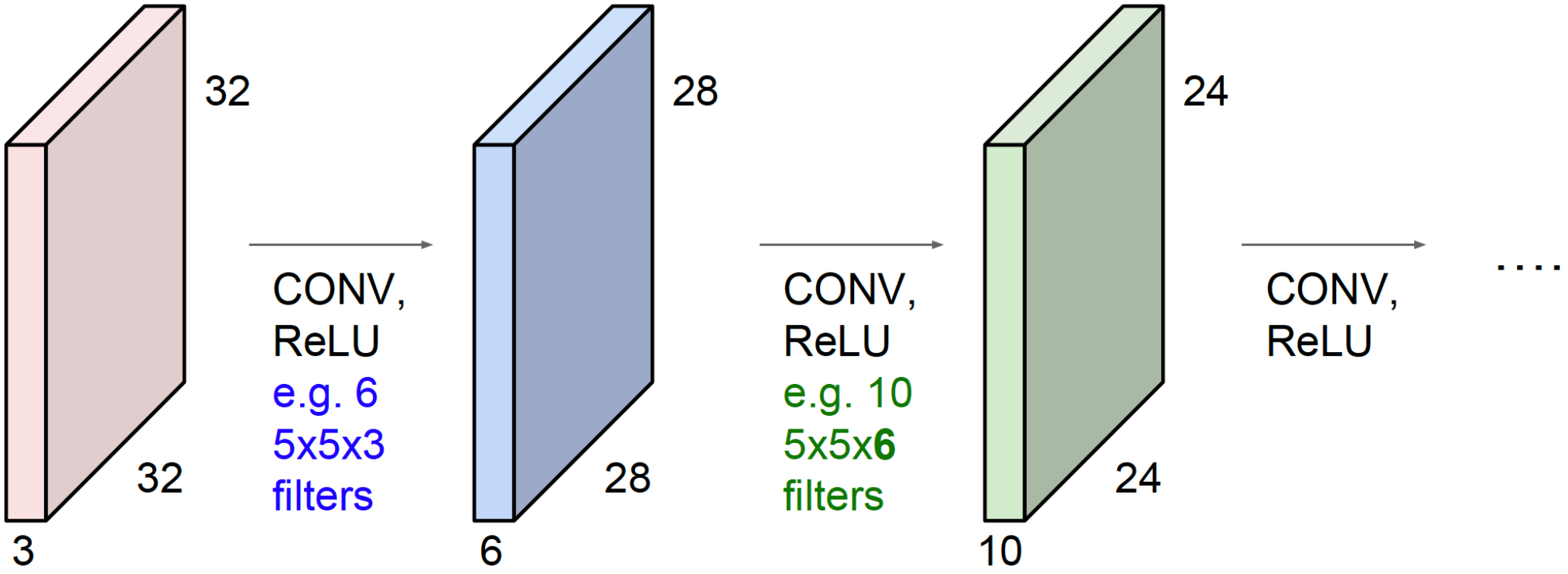
e.g. input 7x7
3x3 filter, applied with **stride 1**
pad with 1 pixel border => what is the output?

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, Justin Johnson, Serena Yeung

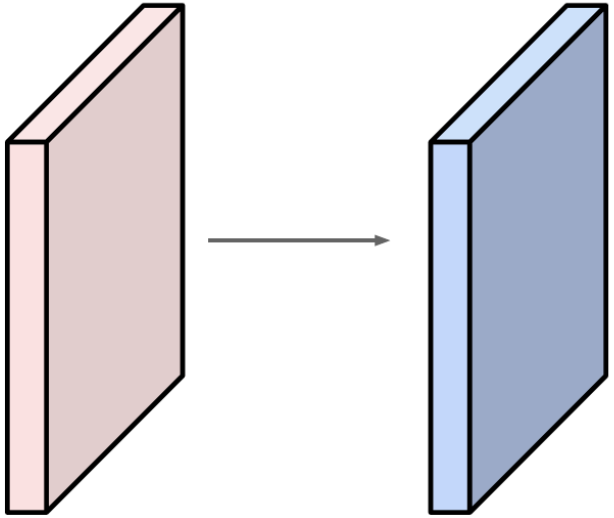
Remember back to…

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, Justin Johnson, Serena Yeung

Examples time:

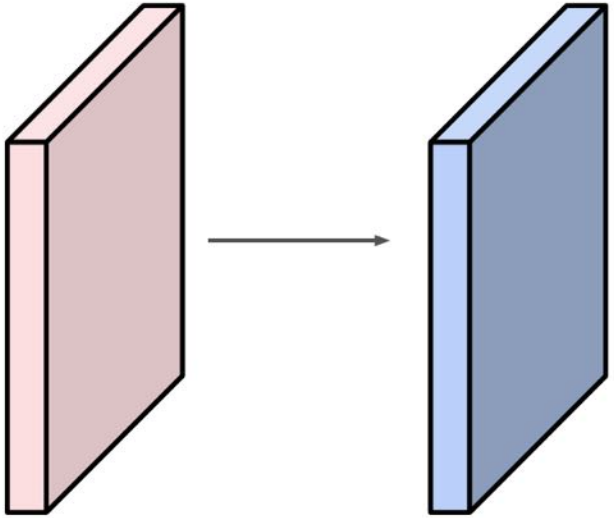


Input volume: **32x32x3**
10 5x5 filters with stride 1, pad 2

Output volume size: ?

slide credit: Fei-Fei, Justin Johnson, Serena Yeung

Examples time:

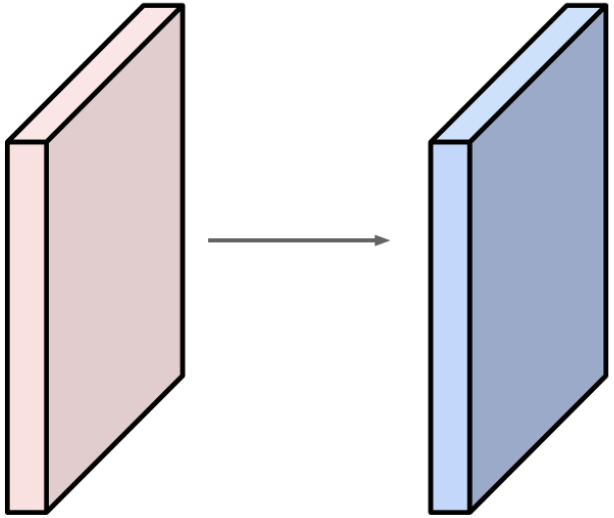


Input volume: **32x32x3**
10 5x5 filters with stride 1, pad 2

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

slide credit: Fei-Fei, Justin Johnson, Serena Yeung

Examples time:

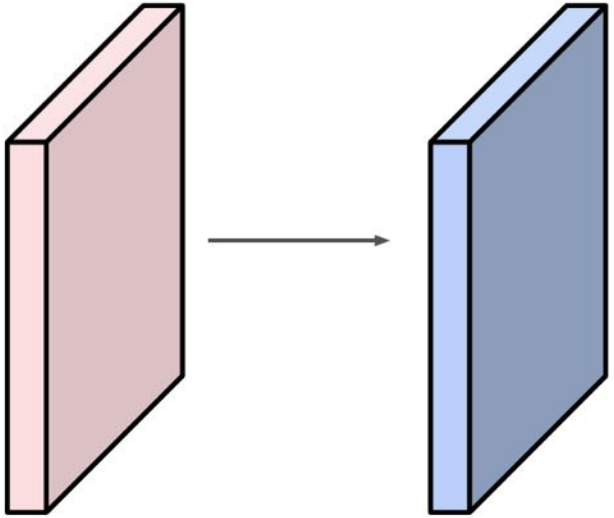


Input volume: **32x32x3**
10 5x5 filters with stride 1, pad 2

Number of parameters in this layer?

slide credit: Fei-Fei, Justin Johnson, Serena Yeung

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, Justin Johnson, Serena Yeung

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.

slide credit: Fei-Fei, Justin Johnson, Serena Yeung

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.

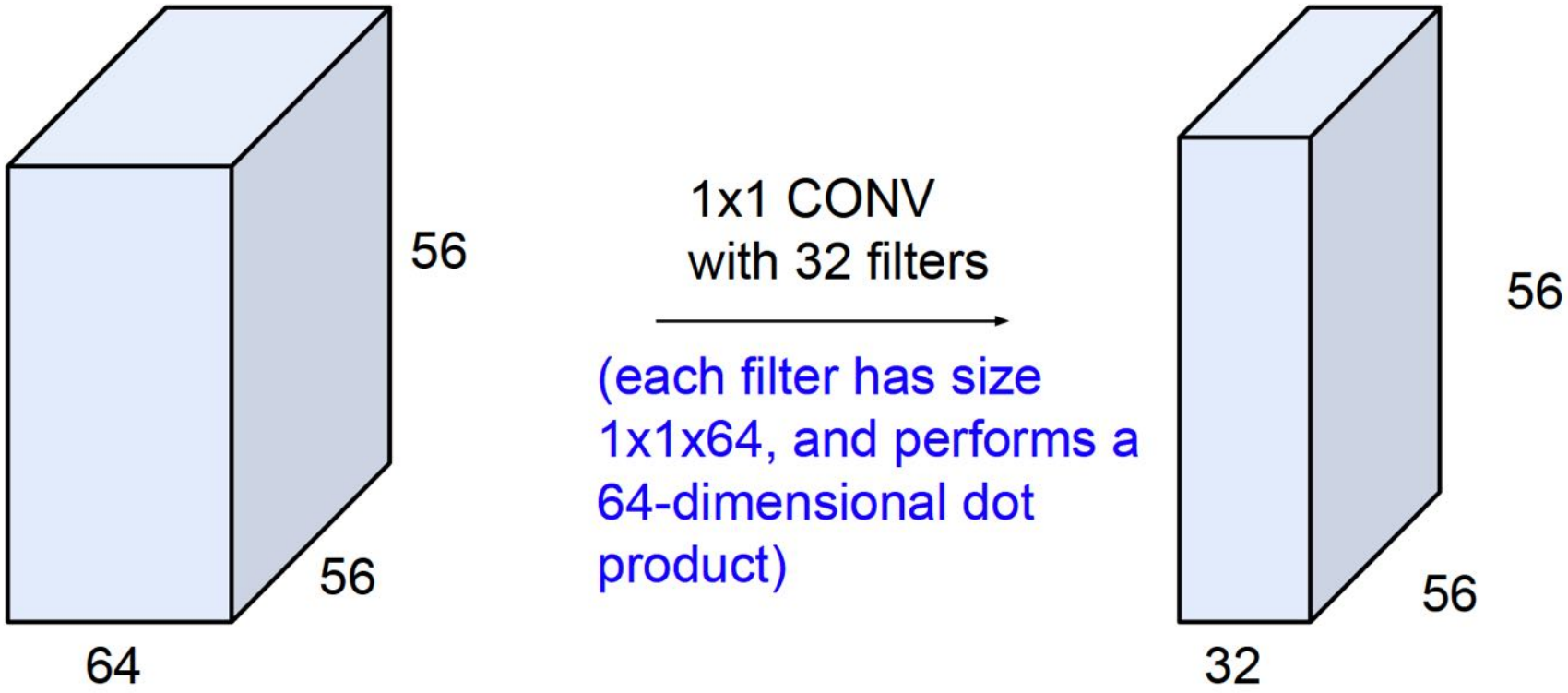
Common settings:

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

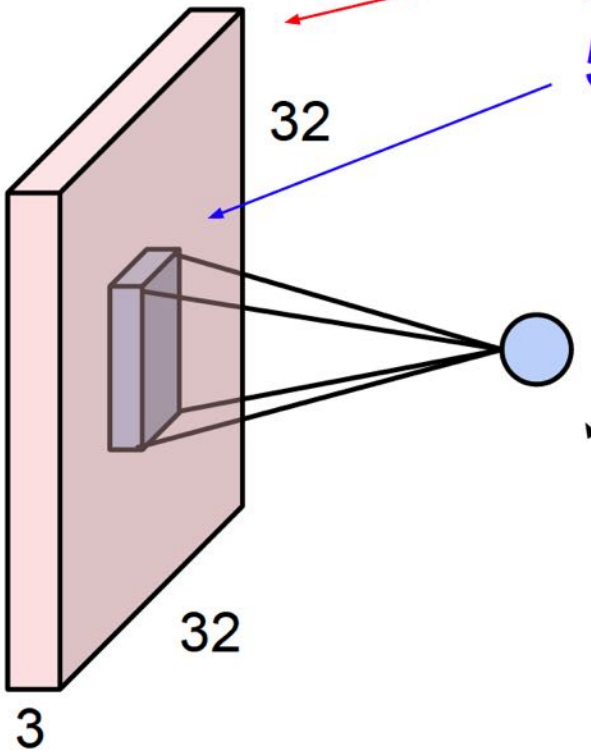
slide credit: Fei-Fei, Justin Johnson, Serena Yeung

(btw, 1x1 convolution layers make perfect sense)



slide credit: Fei-Fei, Justin Johnson, Serena Yeung

The brain/neuron view of CONV Layer



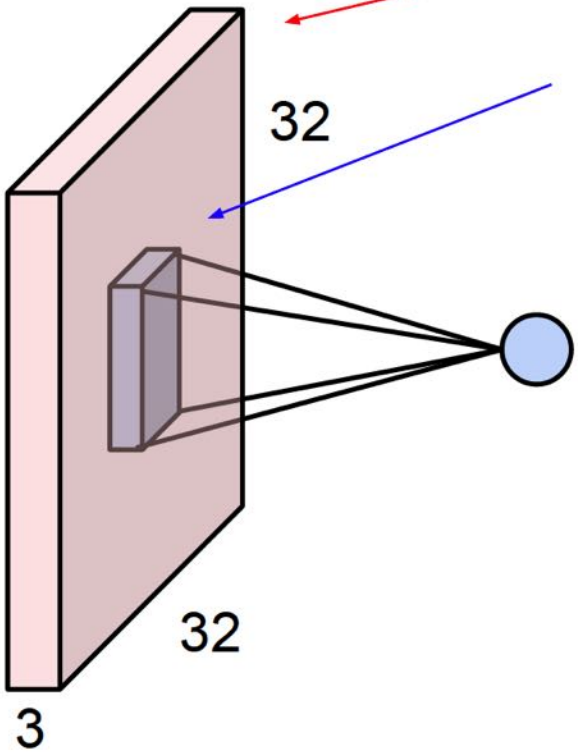
32x32x3 image

5x5x3 filter

1 number:
the result of taking a dot product between
the filter and this part of the image
(i.e. 5*5*3 = 75-dimensional dot product)

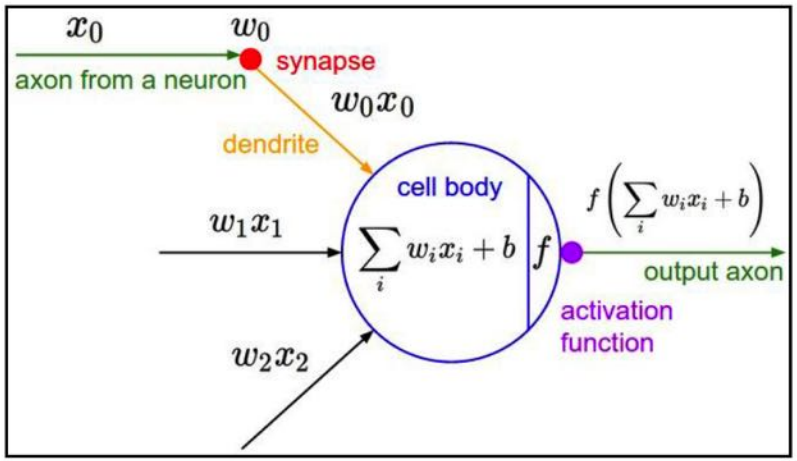
slide credit: Fei-Fei, Justin Johnson, Serena Yeung

The brain/neuron view of CONV Layer



32x32x3 image

5x5x3 filter

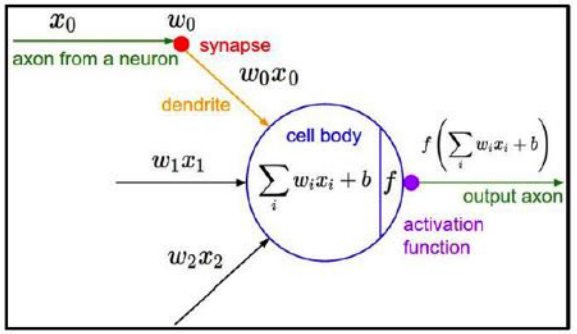


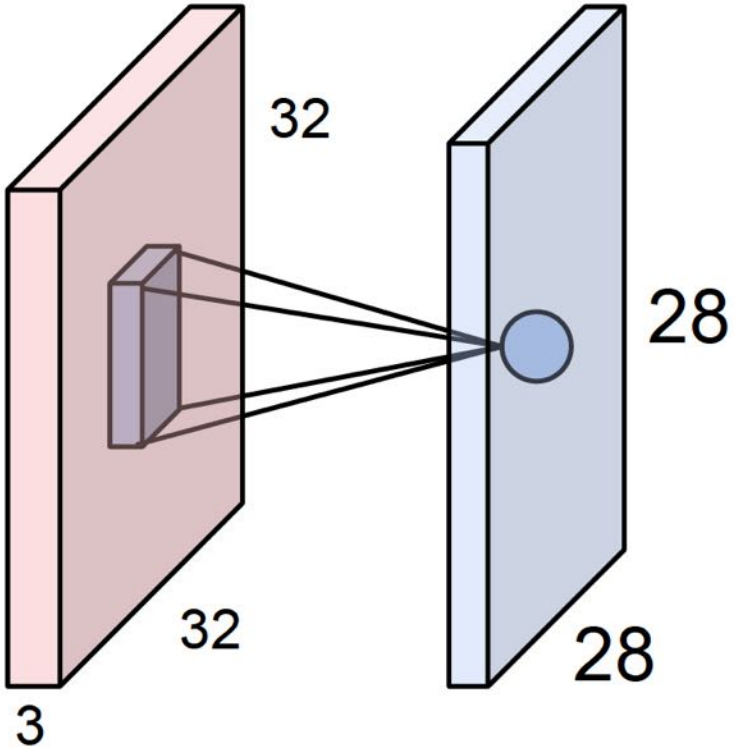
It's just a neuron with local connectivity...

1 number:
the result of taking a dot product between the filter and this part of the image
(i.e. 5*5*3 = 75-dimensional dot product)

slide credit: Fei-Fei, Justin Johnson, Serena Yeung

The brain/neuron view of CONV Layer





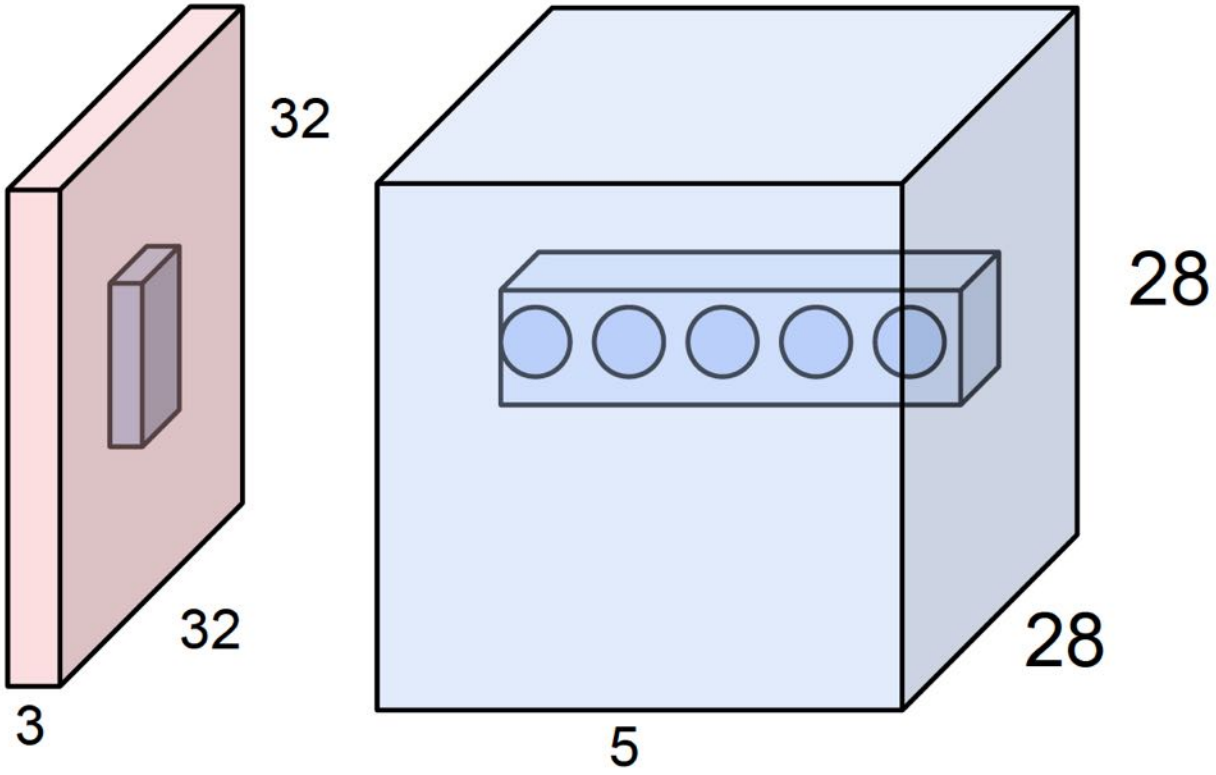
An activation map is a 28x28 sheet of neuron outputs:

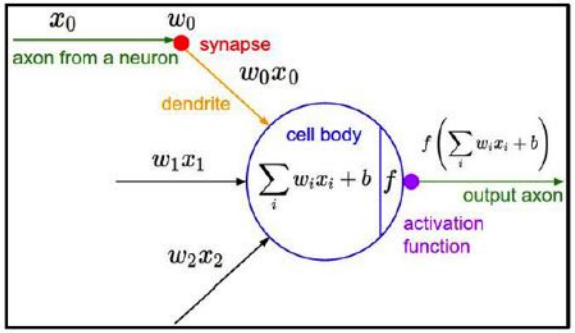
1. Each is connected to a small region in the input
2. All of them share parameters

"5x5 filter" -> "5x5 receptive field for each neuron"

slide credit: Fei-Fei, Justin Johnson, Serena Yeung

The brain/neuron view of CONV Layer



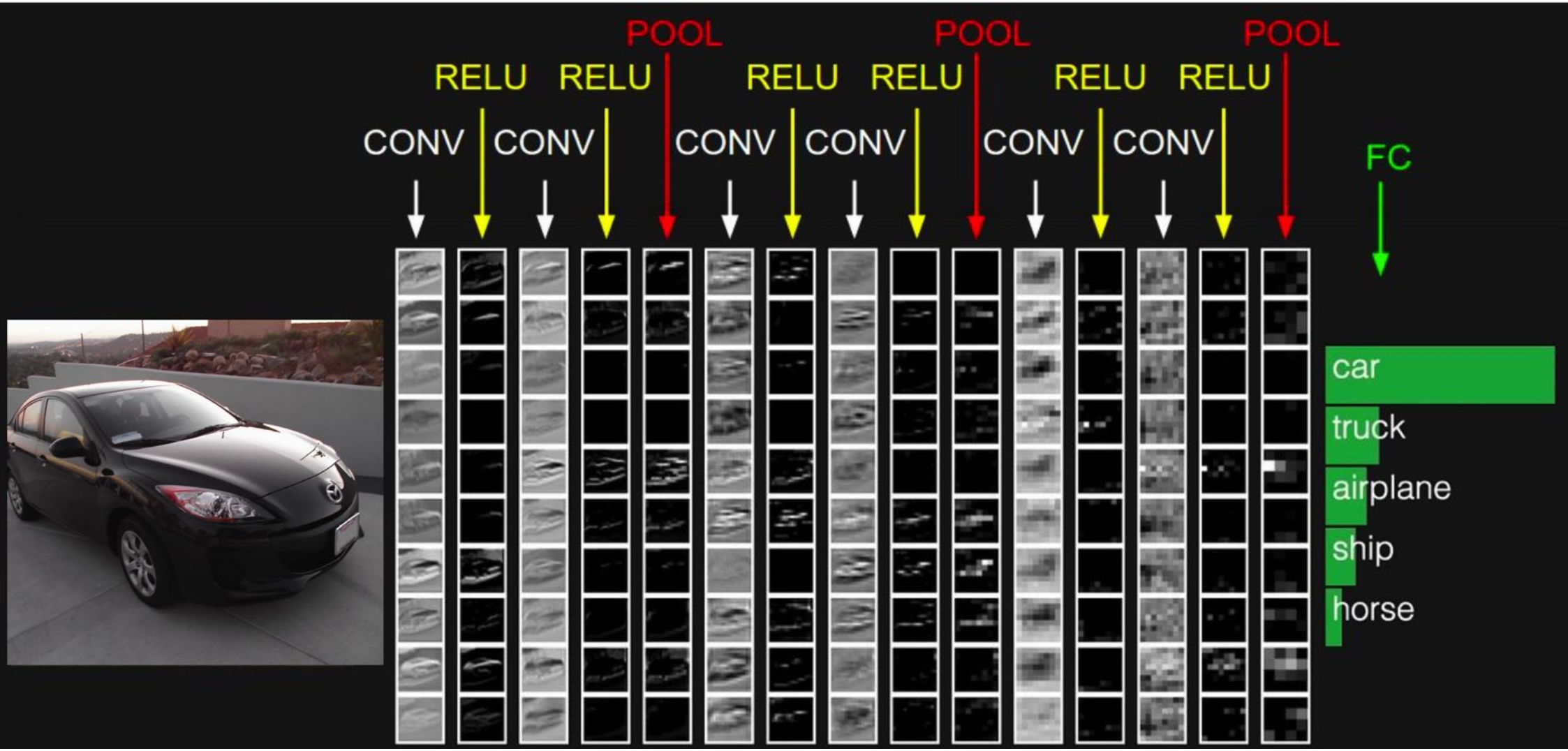


E.g. with 5 filters,
CONV layer consists of neurons arranged in a 3D grid (28x28x5)

There will be 5 different neurons all looking at the same region in the input volume

slide credit: Fei-Fei, Justin Johnson, Serena Yeung

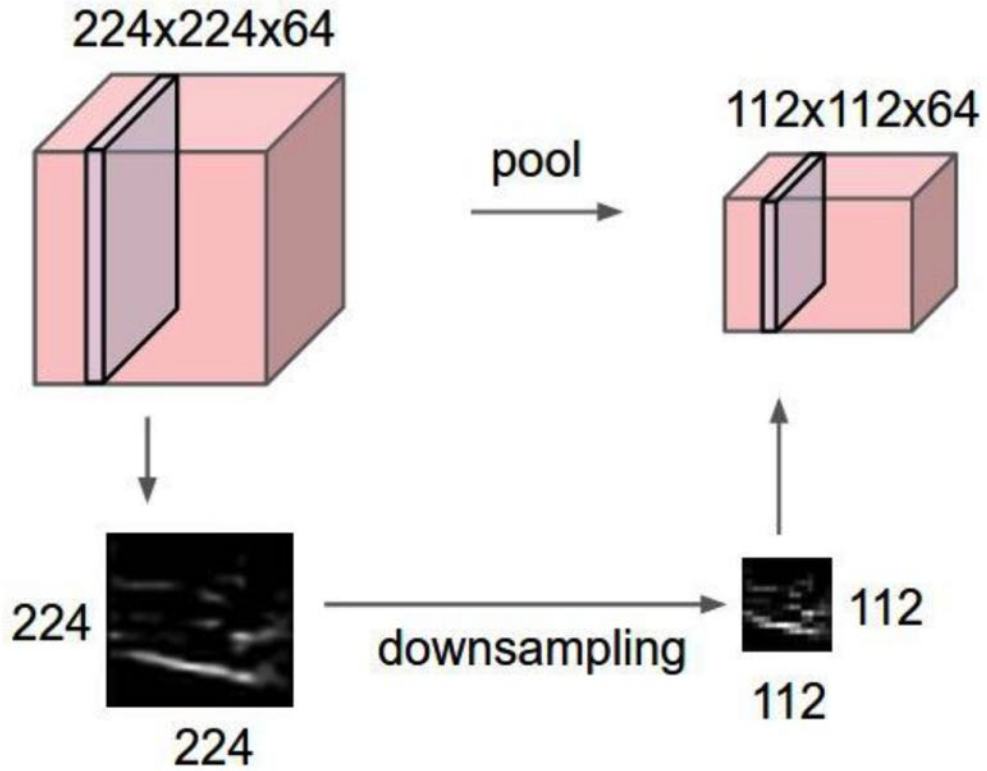
two more layers to go: POOL/FC



slide credit: Fei-Fei, Justin Johnson, Serena Yeung

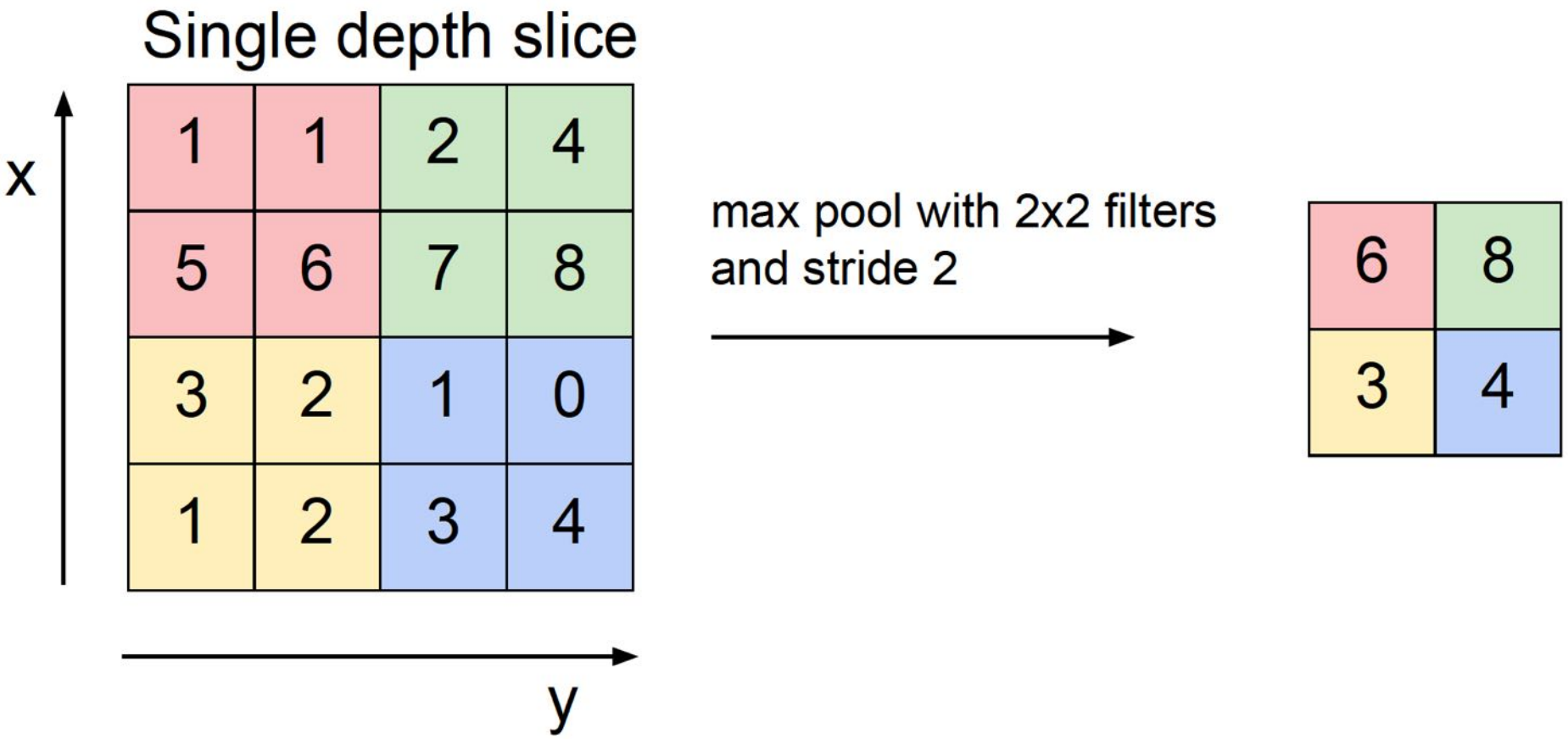
Pooling layer

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



slide credit: Fei-Fei, Justin Johnson, Serena Yeung

MAX POOLING



slide credit: Fei-Fei, Justin Johnson, Serena Yeung

(Max) Pooling Layer

- Accepts a volume of size $W_1 \times H_1 \times D_1$
- Requires three hyperparameters:
  - their spatial extent F,
  - the stride S,
- Produces a volume of size $W_2 \times H_2 \times D_2$ where:
  - $W_2 = (W_1 - F)/S + 1$
  - $H_2 = (H_1 - F)/S + 1$
  - $D_2 = D_1$
- Introduces zero parameters since it computes a fixed function of the input
- Note that it is not common to use zero-padding for Pooling layers

slide credit: Fei-Fei, Justin Johnson, Serena Yeung

(Max) Pooling Layer

- Accepts a volume of size $W_1 \times H_1 \times D_1$
- Requires three hyperparameters:
  - their spatial extent F,
  - the stride S,
- Produces a volume of size $W_2 \times H_2 \times D_2$ where:
  - $W_2 = (W_1 - F)/S + 1$
  - $H_2 = (H_1 - F)/S + 1$
  - $D_2 = D_1$
- Introduces zero parameters since it computes a fixed function of the input
- Note that it is not common to use zero-padding for Pooling layers

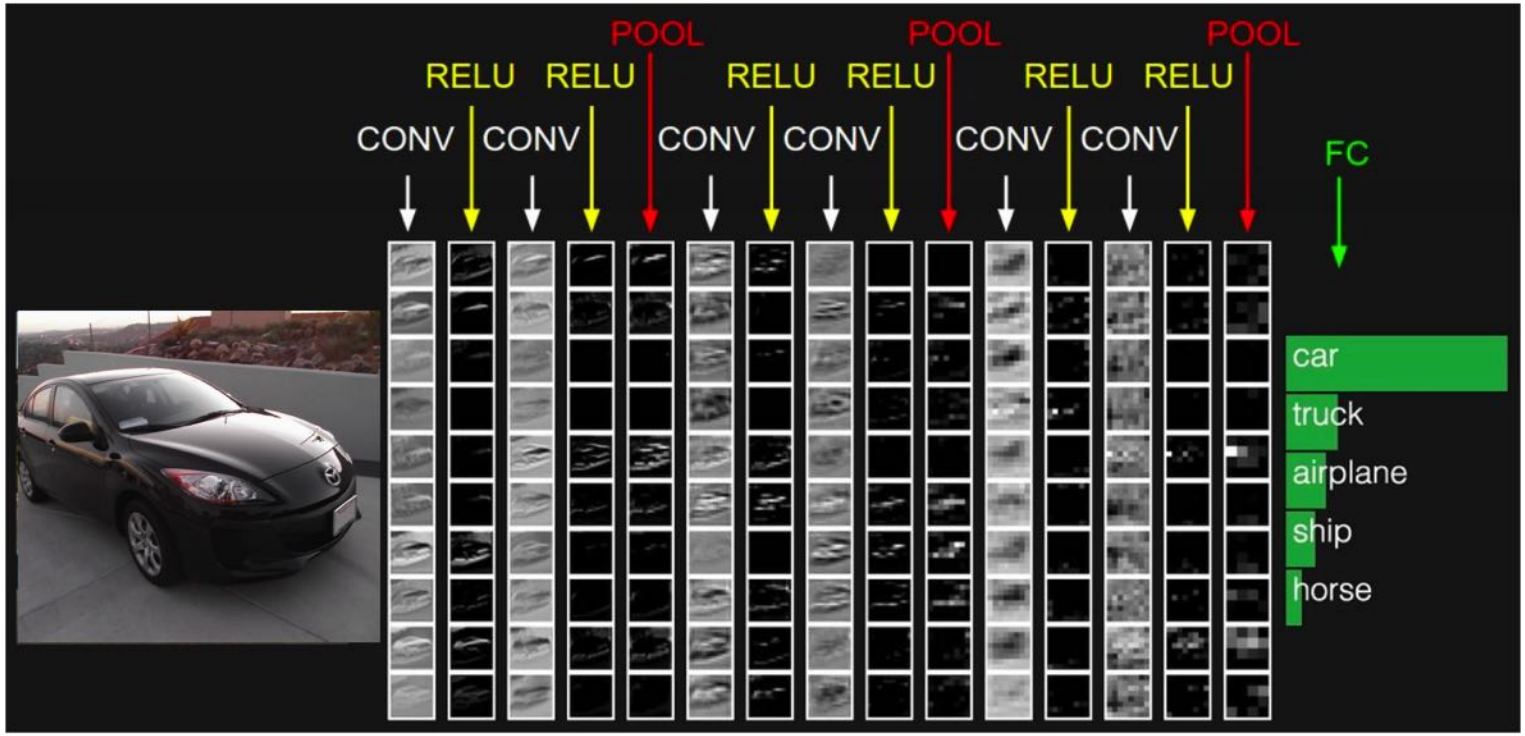
Common settings:

F = 2, S = 2
F = 3, S = 2

slide credit: Fei-Fei, Justin Johnson, Serena Yeung

Fully Connected Layer (FC layer)

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



slide credit: Fei-Fei, Justin Johnson, Serena Yeung

Convolutional Neural Networks

- CNNs stack CONV, POOL, FC layers
  - Trend towards smaller filters and deeper architectures
  - Trend towards getting rid of POOL / FC layers (just CONV)

- Typical Architecture:
  - { (CONF+RELU) * N + POOL } * M + {FC+RELU} * K + SOFTMAX

  - softmax function:

$$\sigma(z_i) = \frac{e^{(-z_i)}}{\sum_{j=1}^{C} e^{(-z_j)}} = \frac{\exp(-z_i)}{\sum_{j=1}^{C} \exp(-z_j)}$$

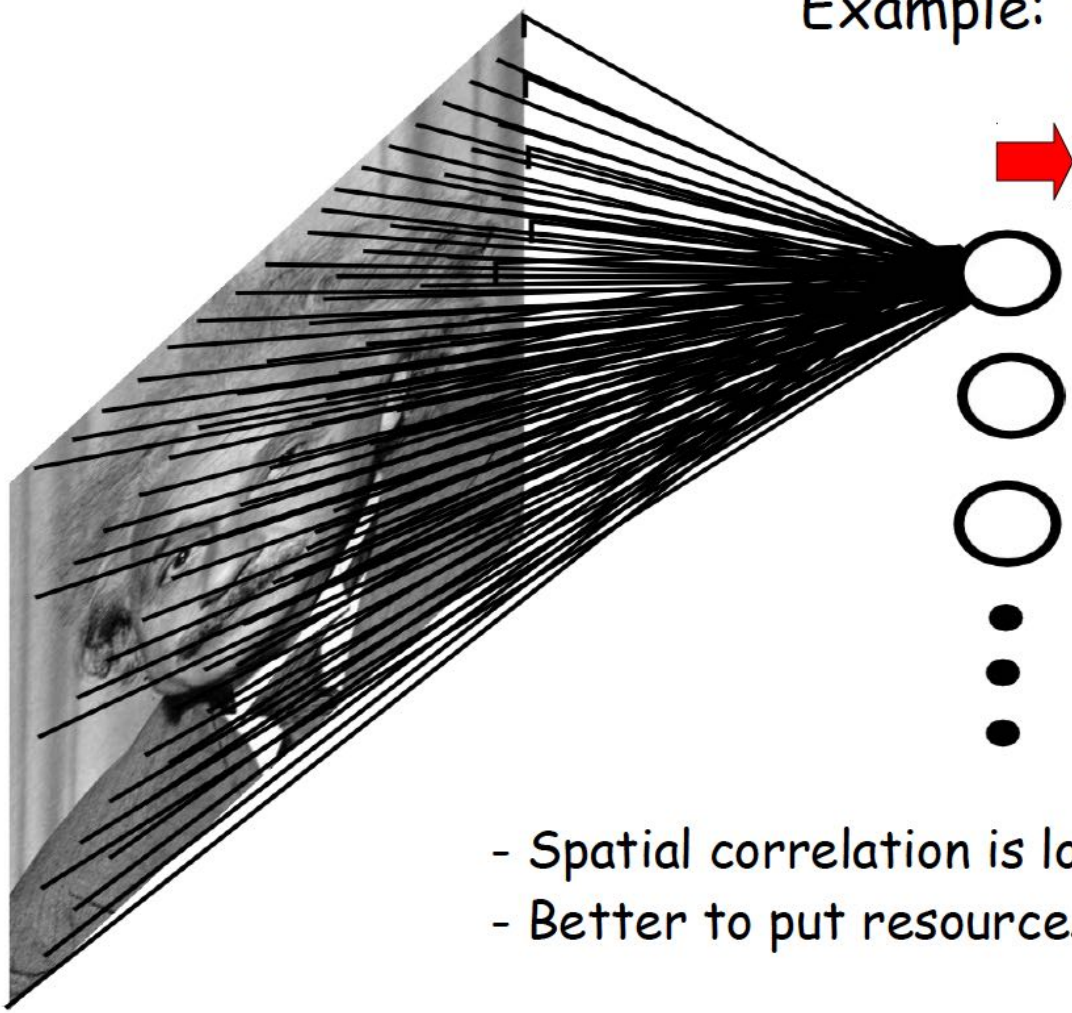
slide credit: Fei-Fei, Justin Johnson, Serena Yeung

max planck institut
informatik

SIC Saarland Informatics Campus

Convolutional Neural Networks
vs.
Fully Connected Neural Networks

FULLY CONNECTED NEURAL NET

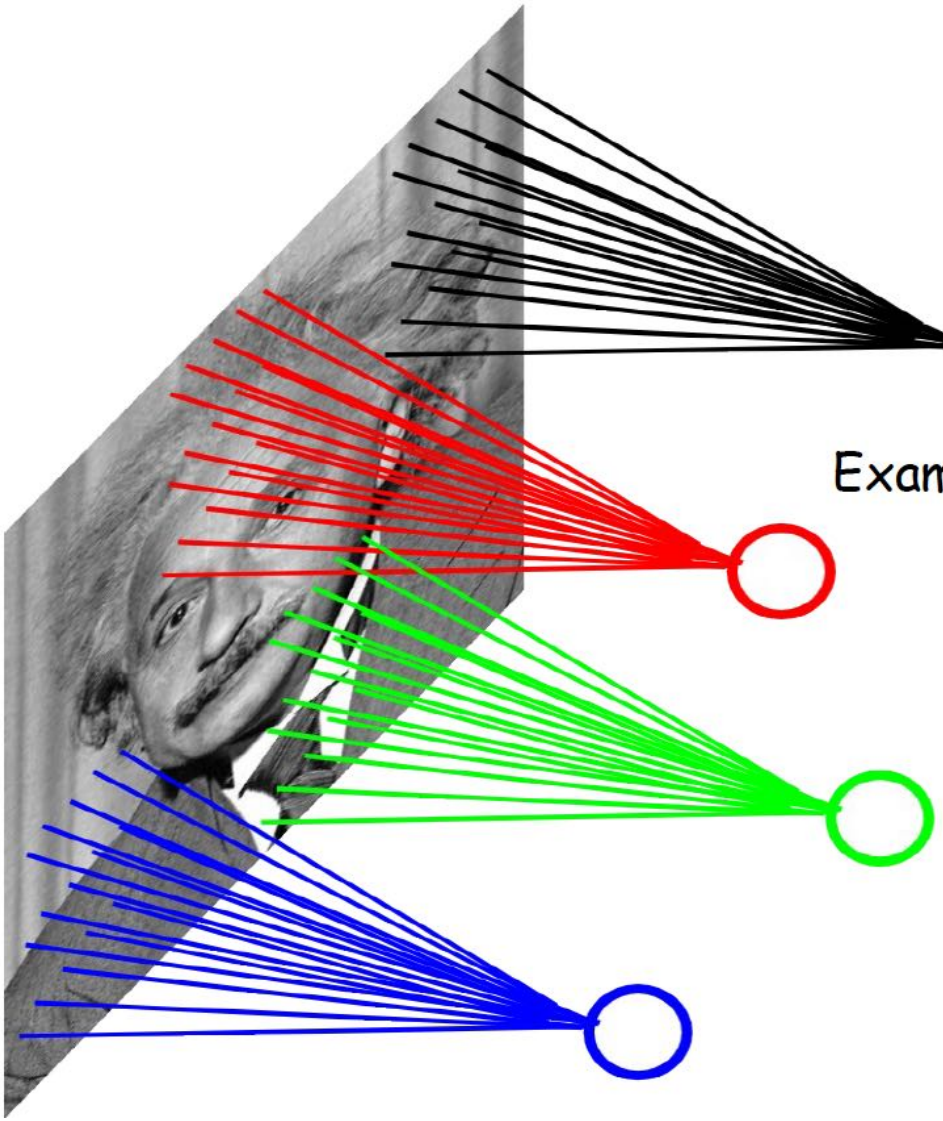


Example: 1000x1000 image
1M hidden units
10^12 parameters!!!

- Spatial correlation is local
- Better to put resources elsewhere!

slide credit: Ranzato

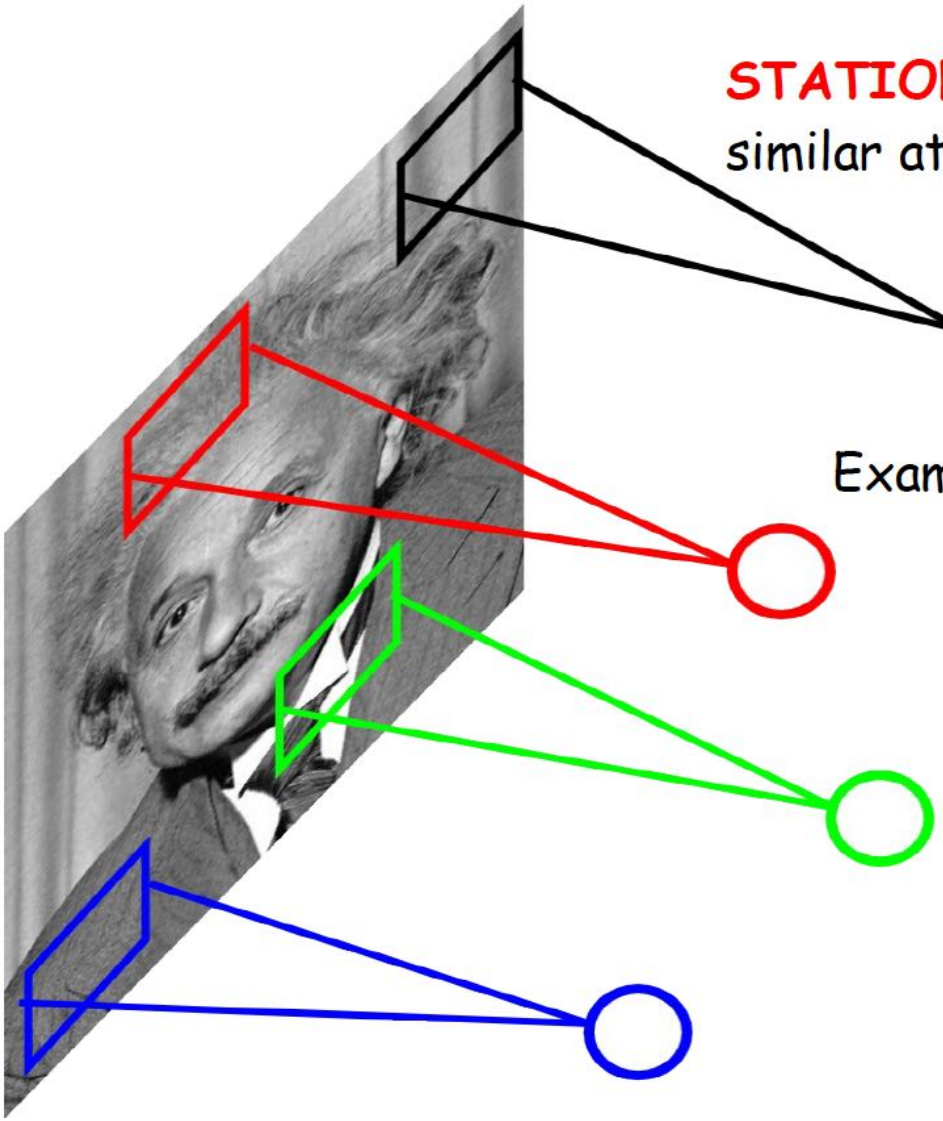
LOCALLY CONNECTED NEURAL NET



Example: 1000x1000 image
1M hidden units
Filter size: 10x10
100M parameters

slide credit: Ranzato

LOCALLY CONNECTED NEURAL NET

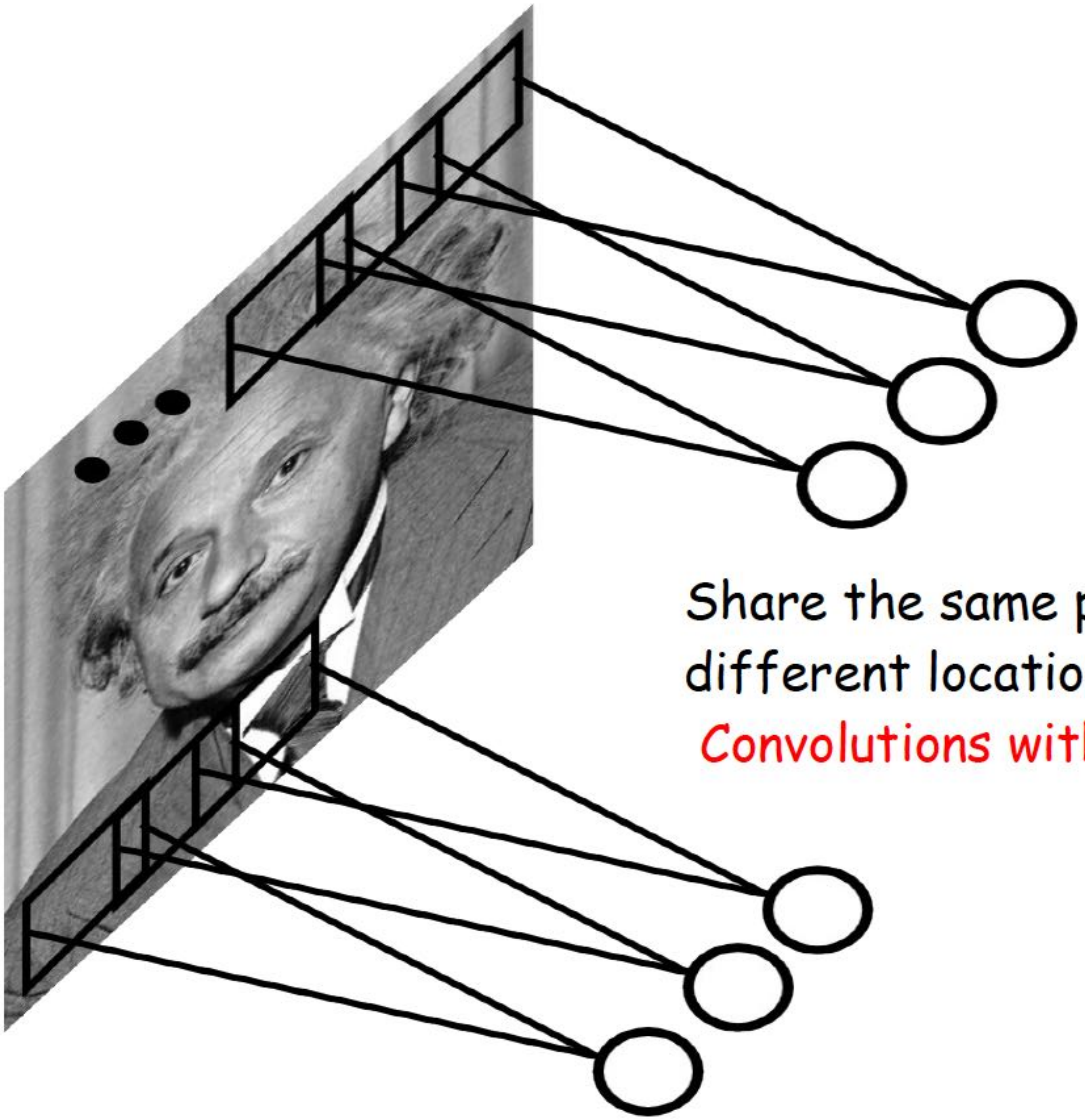


STATIONARITY? Statistics is similar at different locations

Example: 1000x1000 image
1M hidden units
Filter size: 10x10
100M parameters

slide credit: Ranzato

CONVOLUTIONAL NET

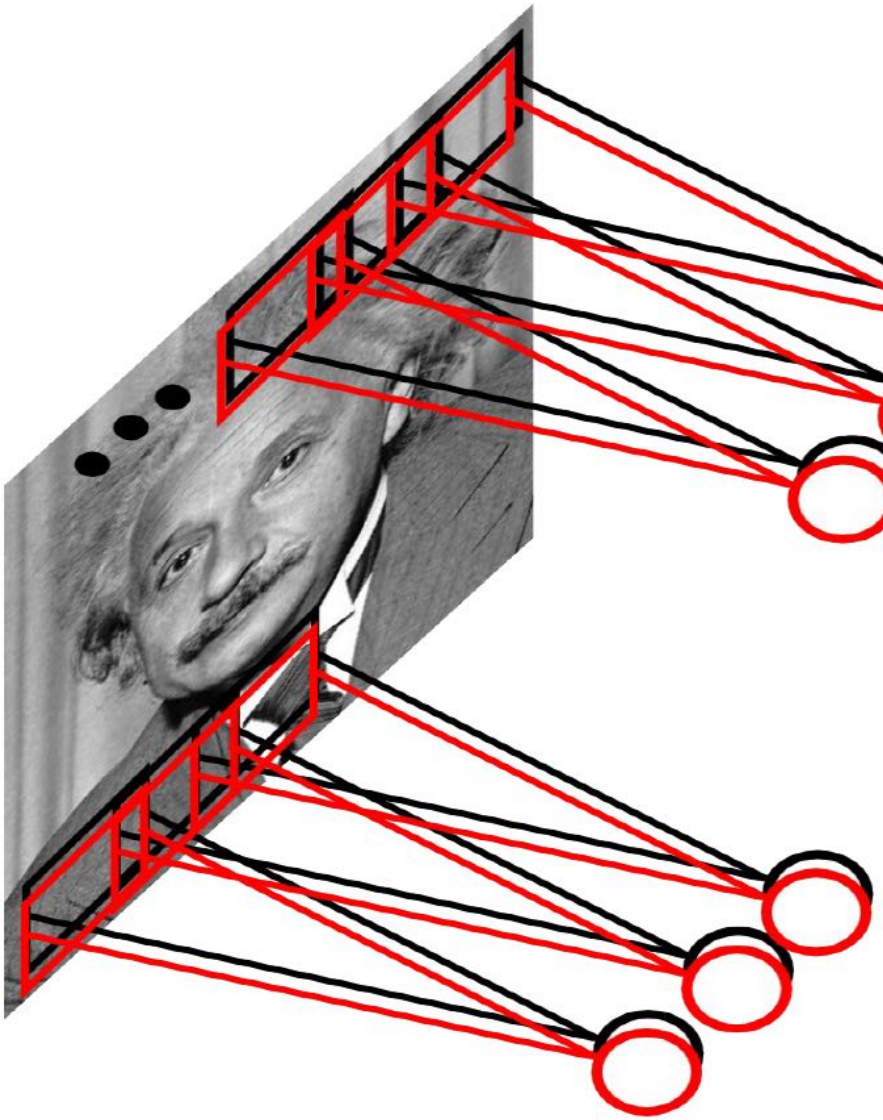


Share the same parameters across different locations:

Convolutions with learned kernels

slide credit: Ranzato

CONVOLUTIONAL NET



Learn multiple filters.

E.g.: 1000x1000 image
100 Filters
Filter size: 10x10
10K parameters

slide credit: Ranzato

NEURAL NETS FOR VISION

A standard neural net applied to images:

- scales quadratically with the size of the input
- does not leverage stationarity

Solution:

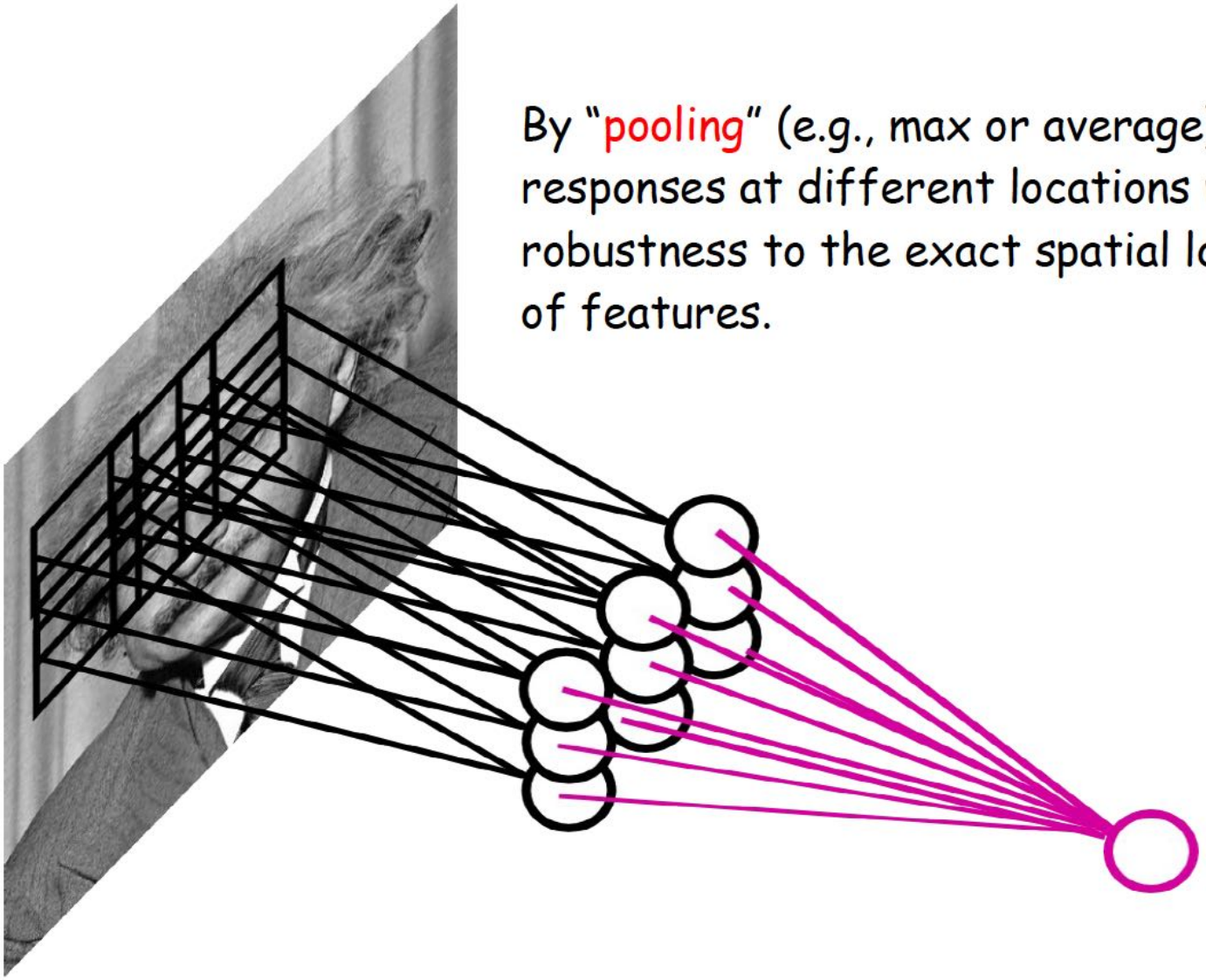
- connect each hidden unit to a small patch of the input
- share the weight across hidden units

This is called: **convolutional network.**

LeCun et al. "Gradient-based learning applied to document recognition" IEEE 1998

slide credit: Ranzato

CONVOLUTIONAL NET



By "pooling" (e.g., max or average) filter responses at different locations we gain robustness to the exact spatial location of features.

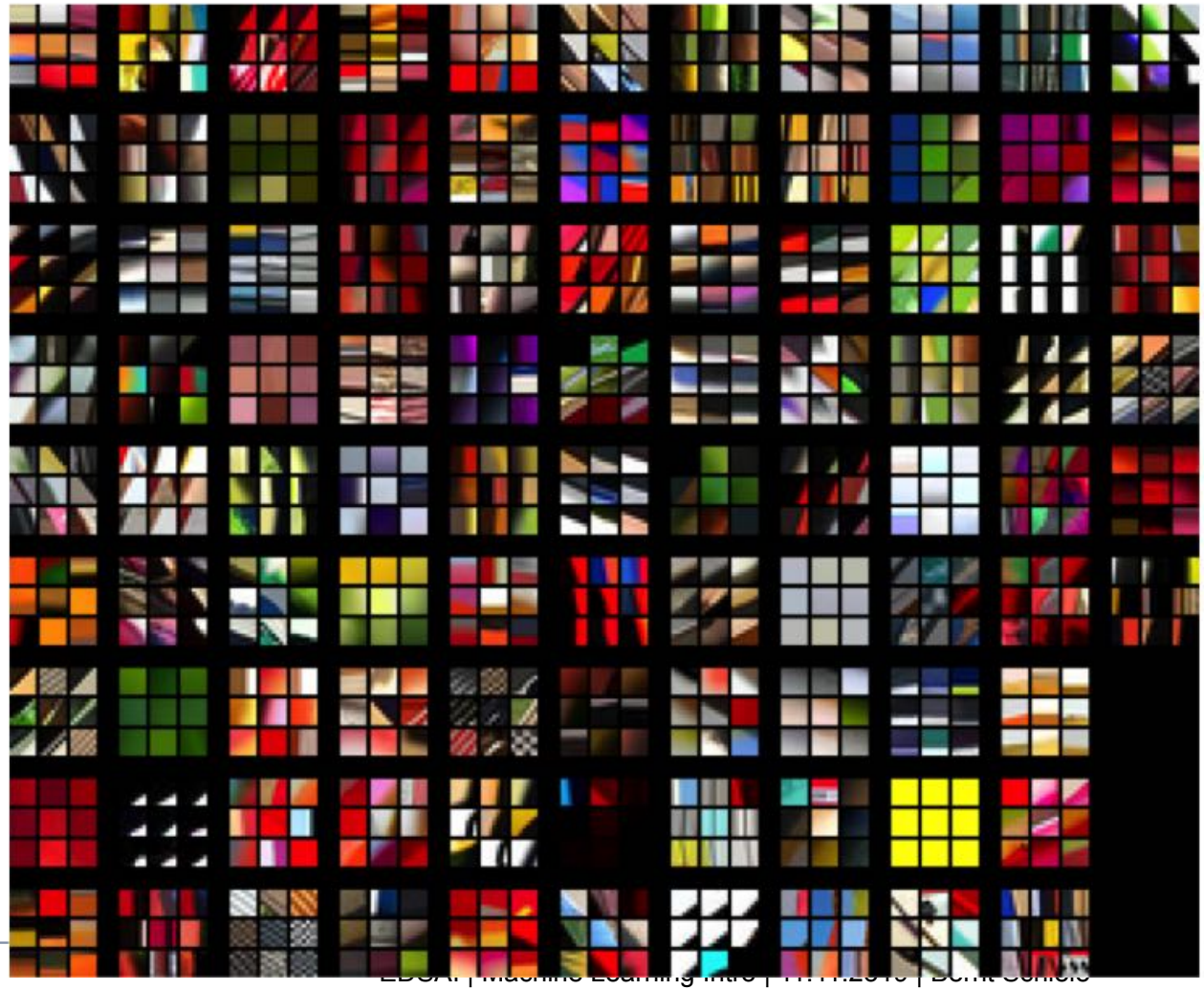
max planck institut
informatik

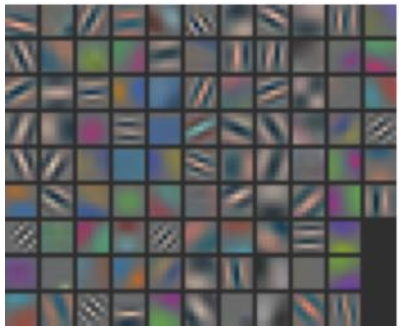
SIC Saarland Informatics Campus

Convolutional Neural Networks

Visualizing of Hierarchical Structure of Layers (here: Alexnet)

Layer 1: Top-9 Patches
(from validation images that give maximal activation for a given feature map)

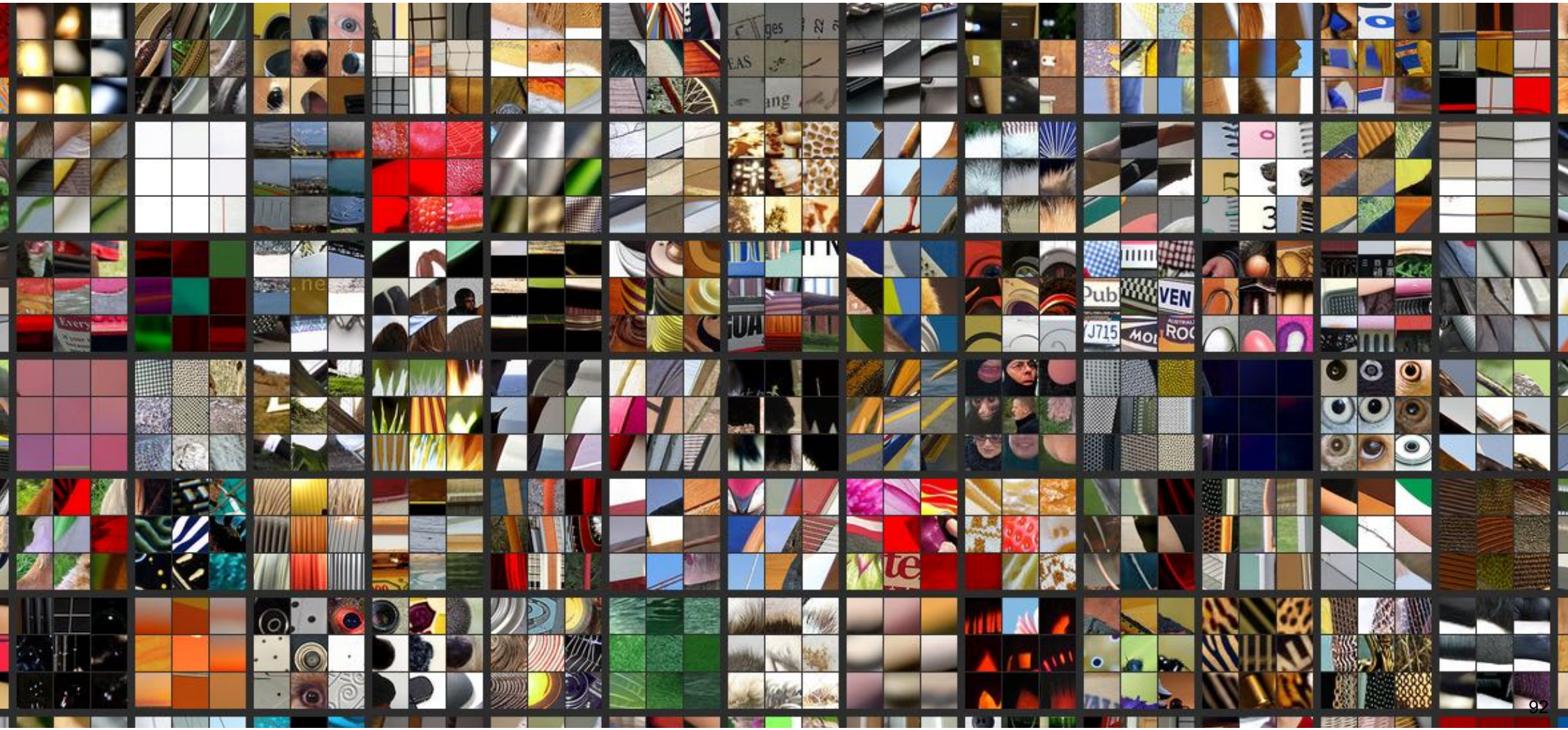




slide credit: Fergus

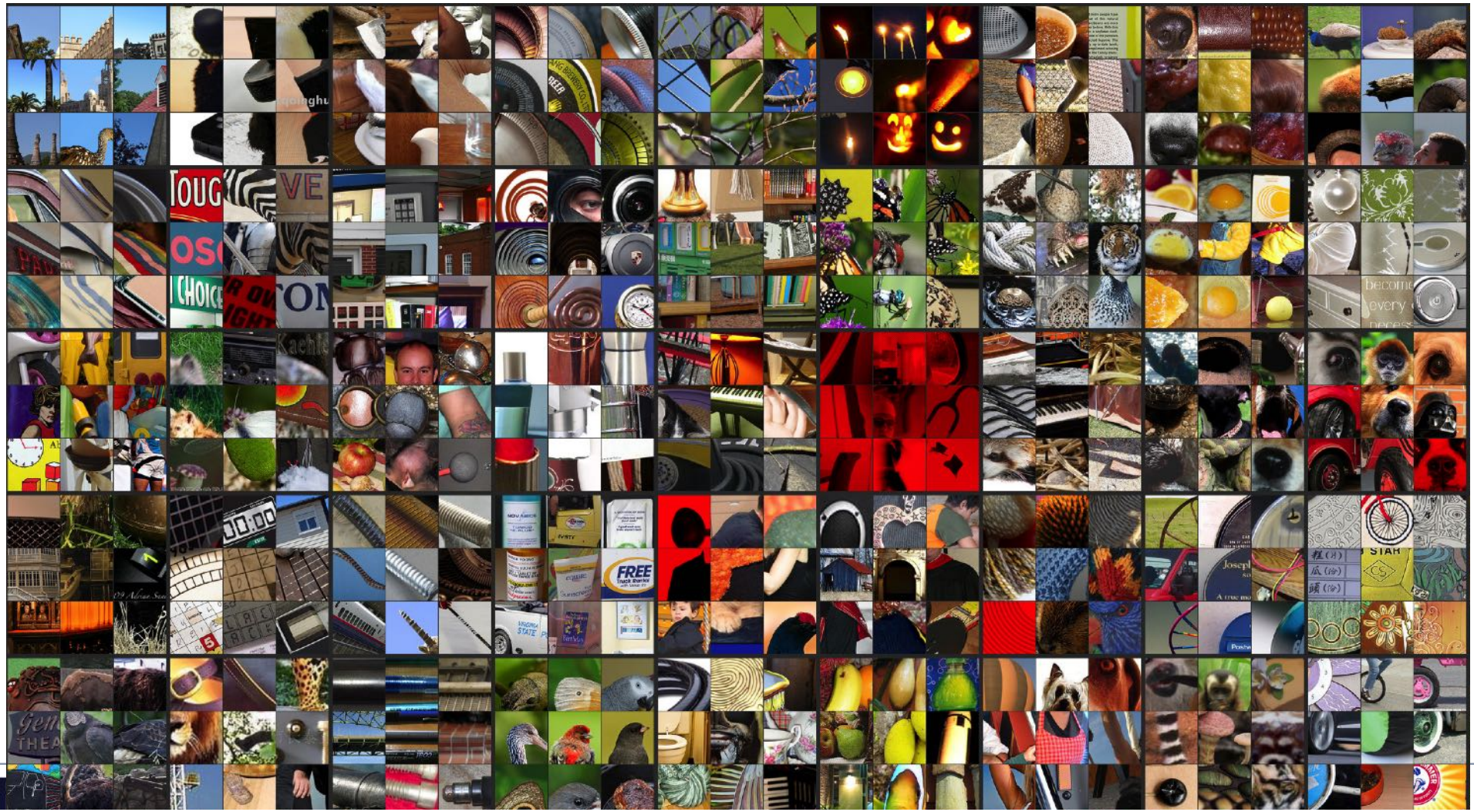
slide credit: Fergus

Layer 2: Top-9 Patches
(from validation images that give maximal activation for a given feature map)



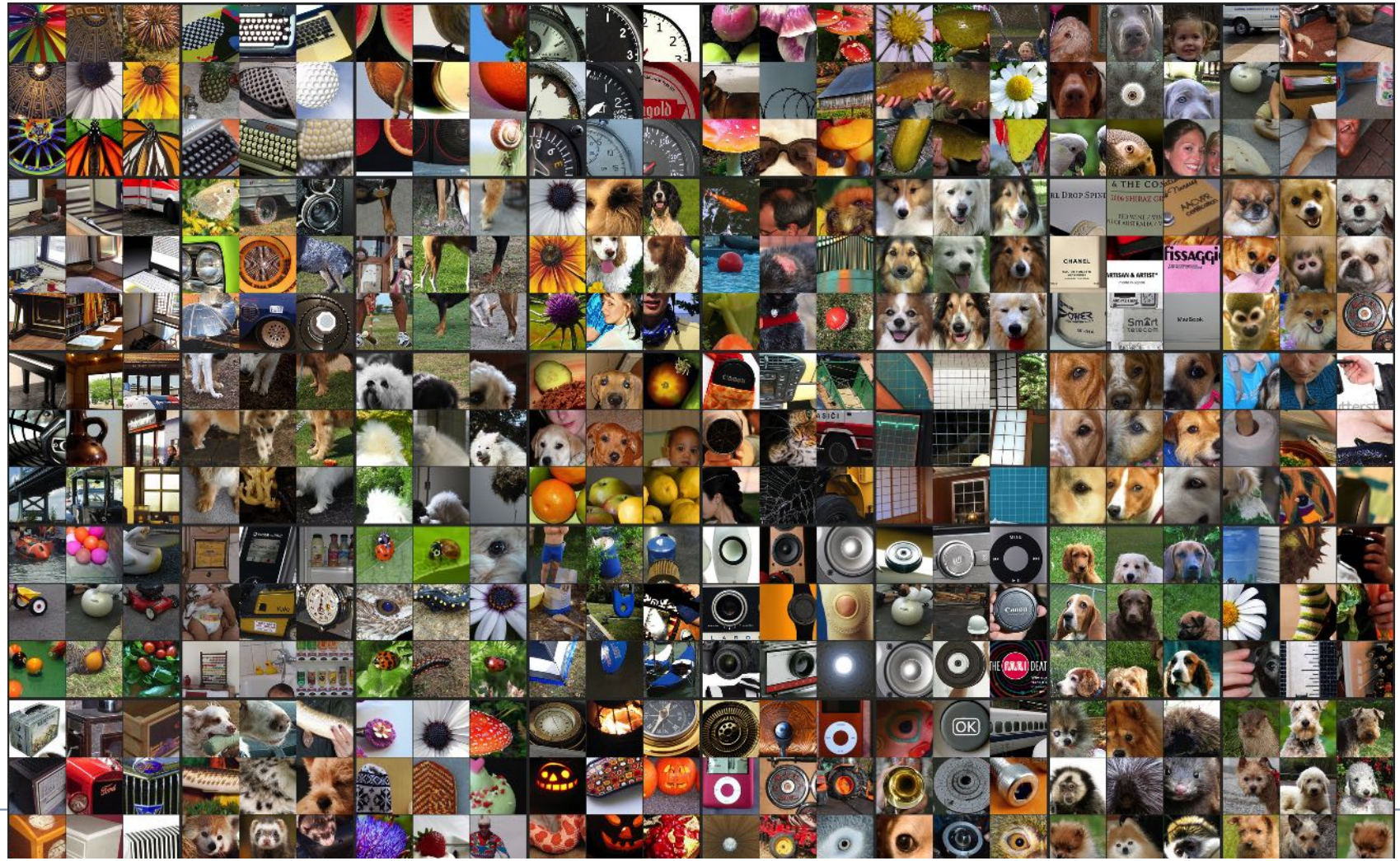
slide credit: Fergus

Layer 3: Top-9 Patches
(from validation images that give maximal activation for a given feature map)



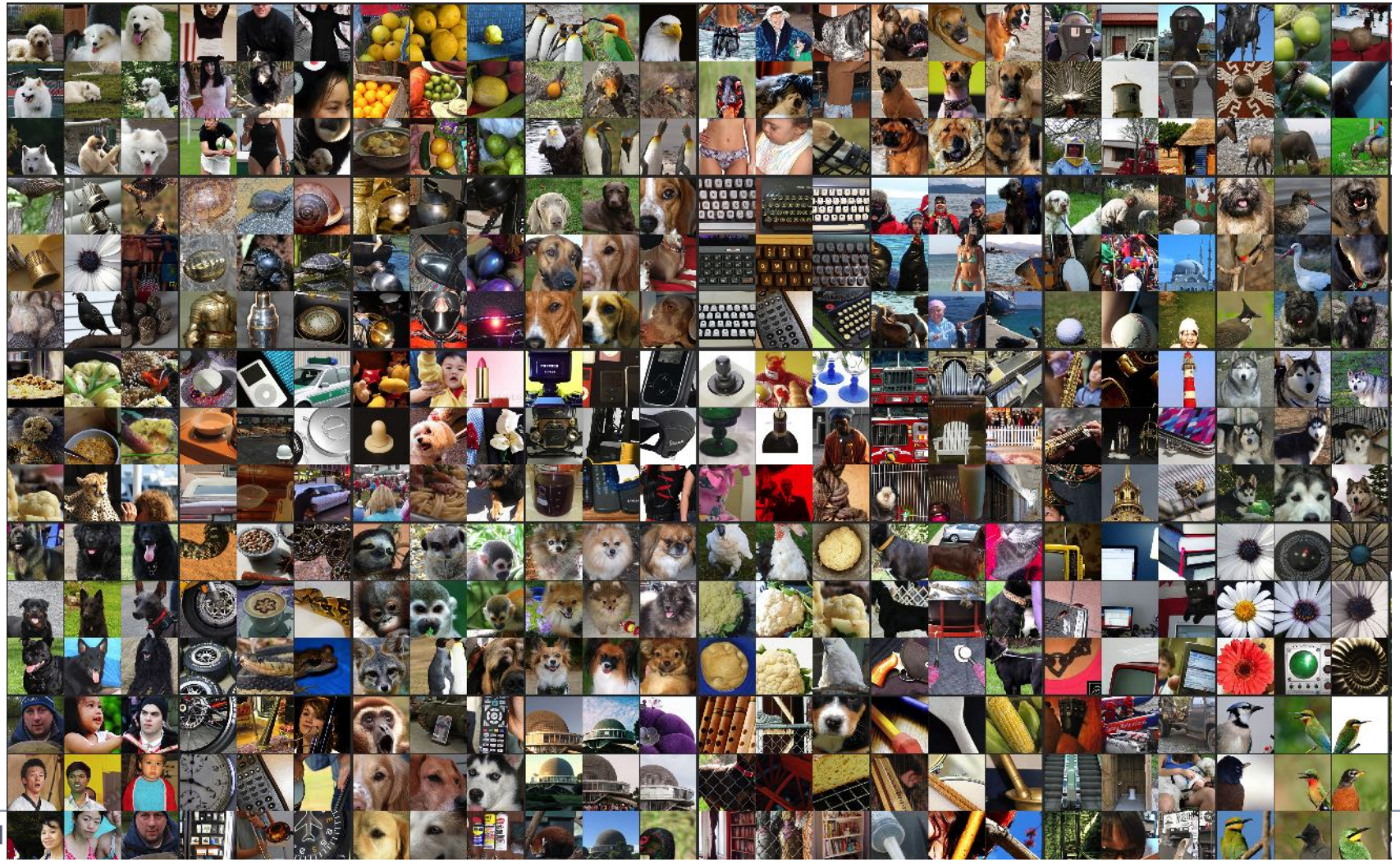
slide credit: Fergus

Layer 4: Top-9 Patches
(from validation images that give maximal activation for a given feature map)



slide credit: Fergus

Layer 5: Top-9 Patches
(from validation images that give maximal activation for a given feature map)



max planck institut
informatik

SIC Saarland Informatics Campus

Convolutional Neural Networks

Depth Matters !

slide credit: Fergus

Architecture of Krizhevsky et al. (Alexnet)

- 8 layers total

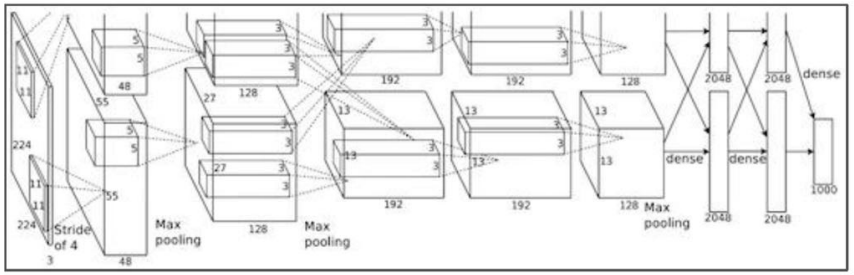
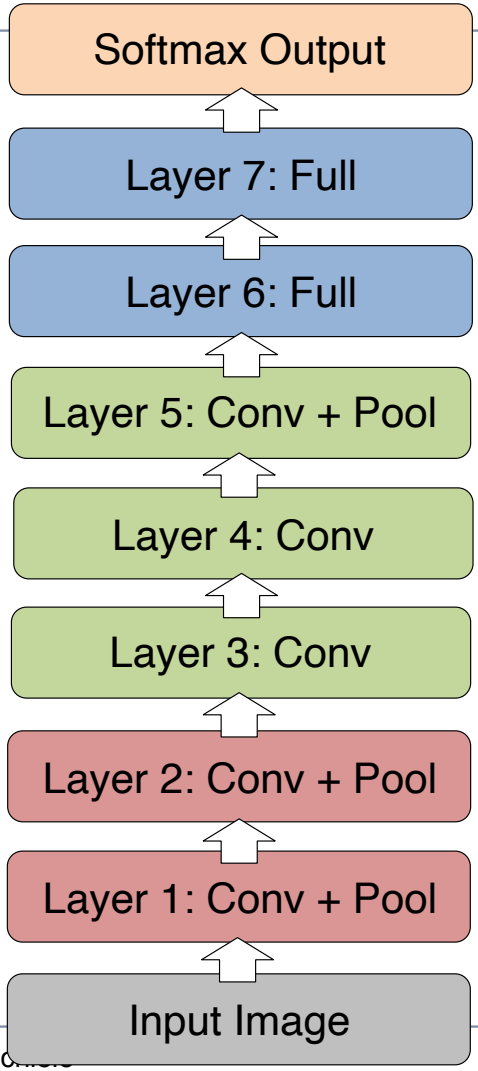


Figure copyright Alex Krizhevsky, Ilya Sutskever, and Geoffrey Hinton, 2012. Reproduced with permission.

"AlexNet"

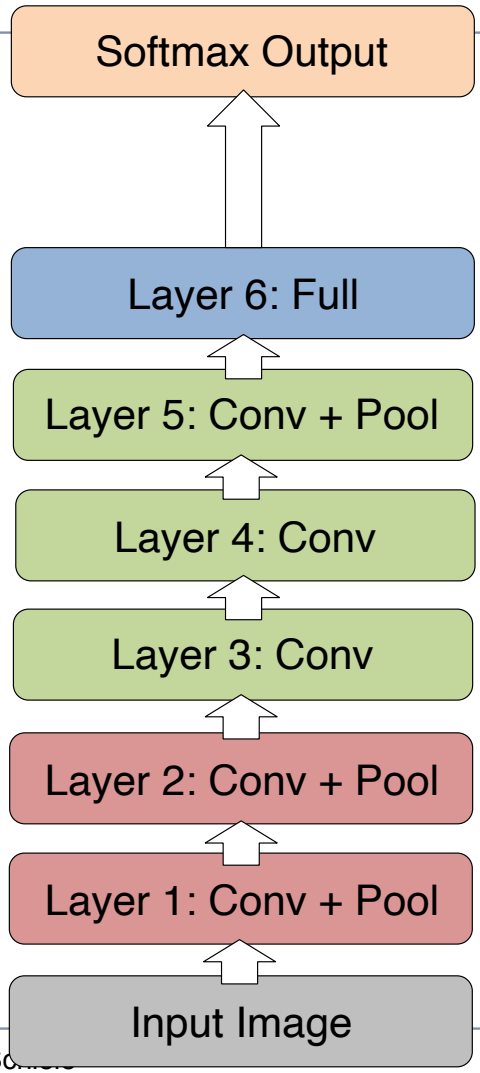
- Trained on Imagenet dataset [Deng et al. CVPR'09]
- 18.1% top-5 error



slide credit: Fergus

Architecture of Krizhevsky et al. (Alexnet)

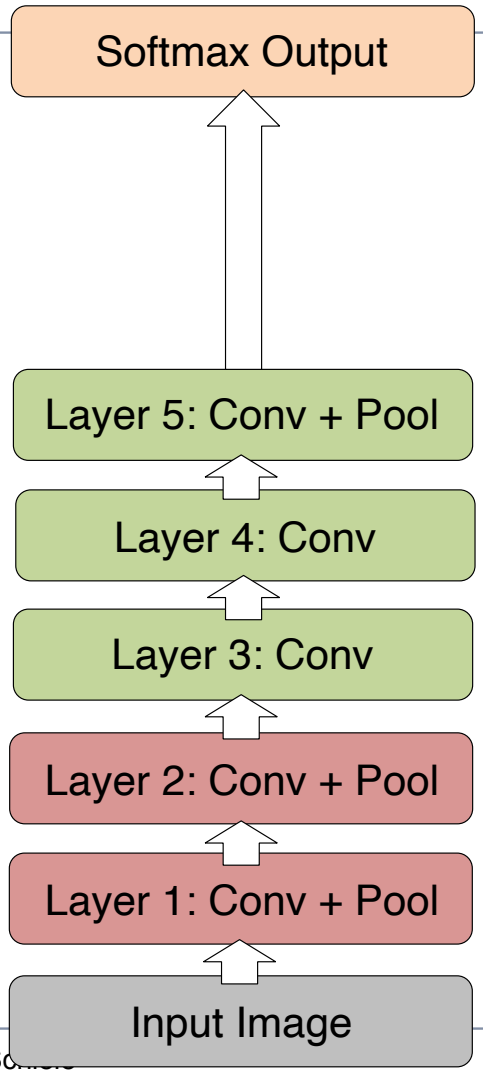
- Remove top fully connected layer
  - Layer 7
- Drop 16 million parameters
- Only 1.1% drop in performance!



slide credit: Fergus

Architecture of Krizhevsky et al. (Alexnet)

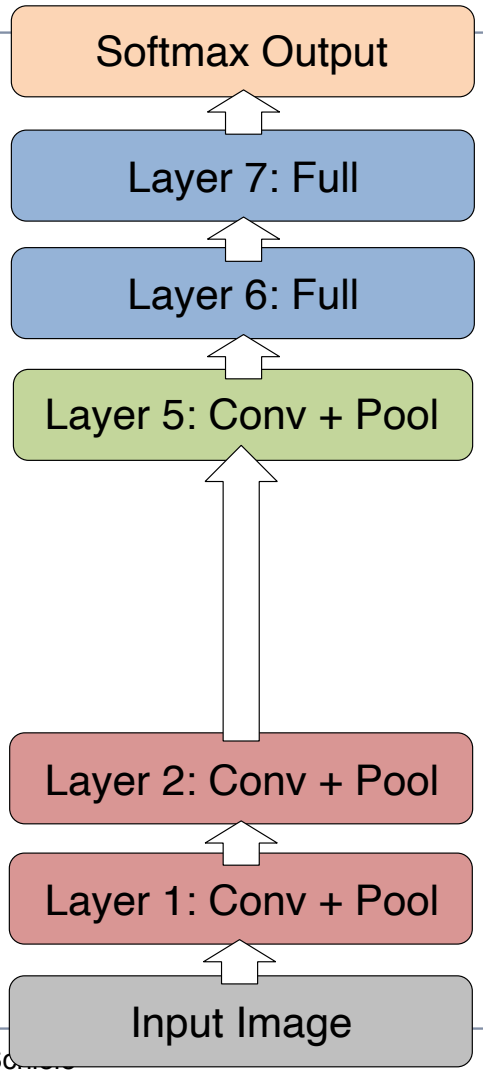
- Remove both fully connected layers
  - Layer 6 & 7
- Drop ~50 million parameters
- 5.7% drop in performance



slide credit: Fergus

Architecture of Krizhevsky et al. (Alexnet)

- Now try removing upper feature extractor layers:
  - Layers 3 & 4
- Drop ~1 million parameters
- 3.0% drop in performance

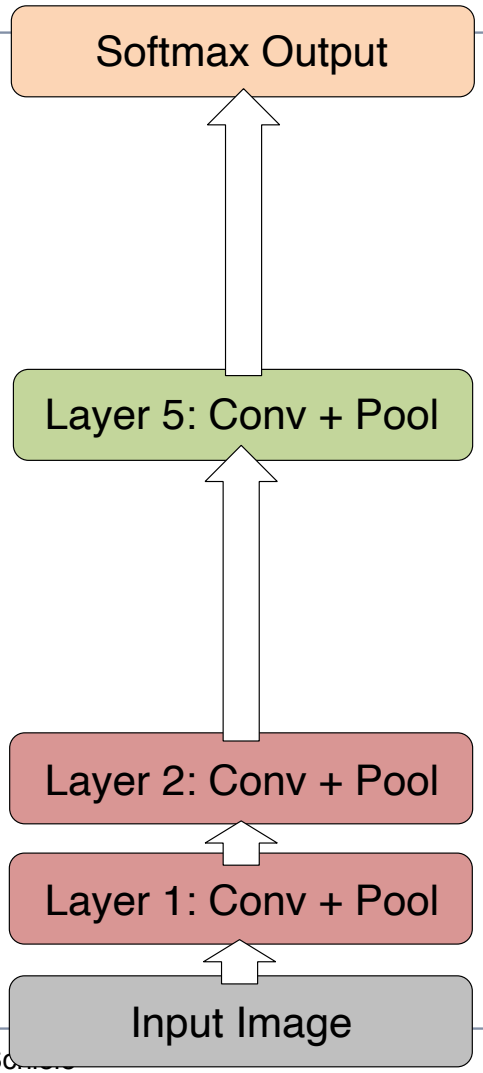


slide credit: Fergus

Architecture of Krizhevsky et al. (Alexnet)

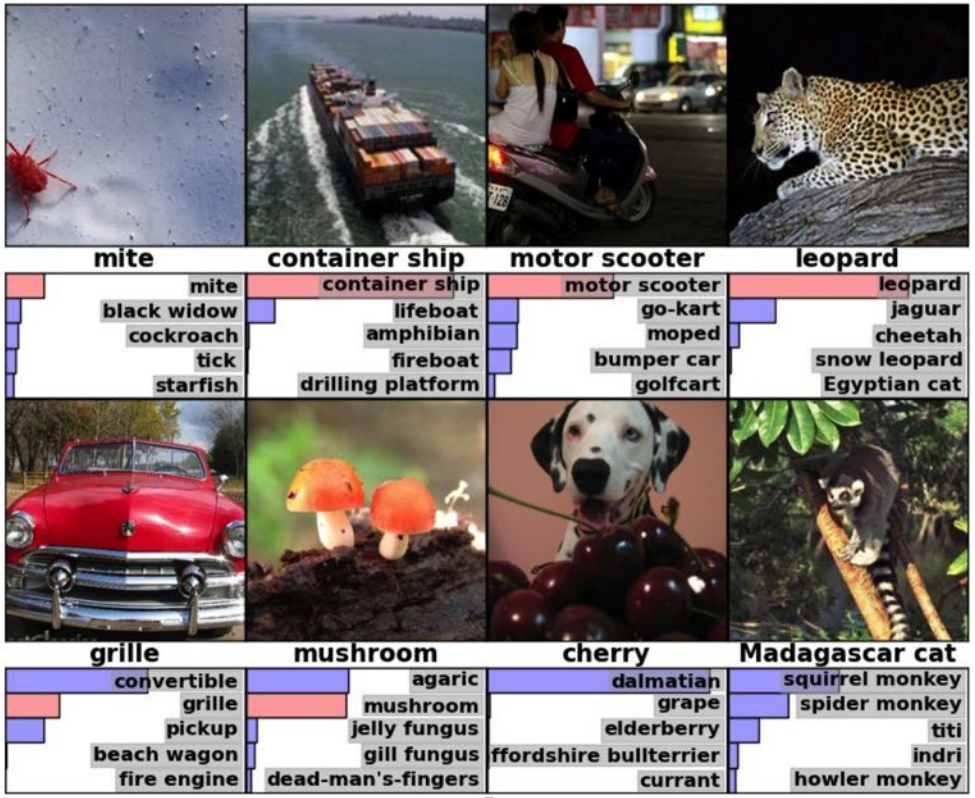
- Now try removing upper feature extractor layers & fully connected:
  - Layers 3, 4, 6 ,7
- Now only 4 layers
- 33.5% drop in performance

→Depth of network is key

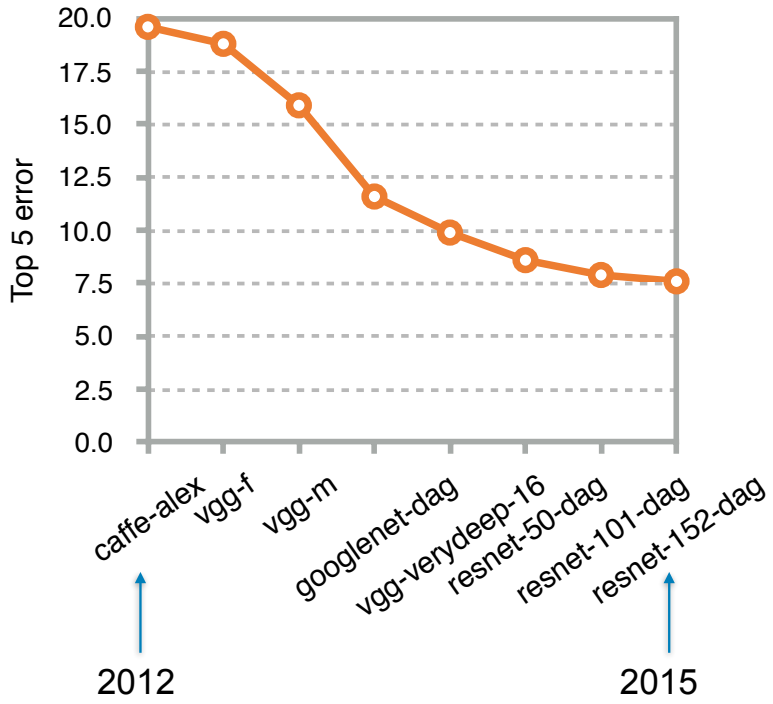


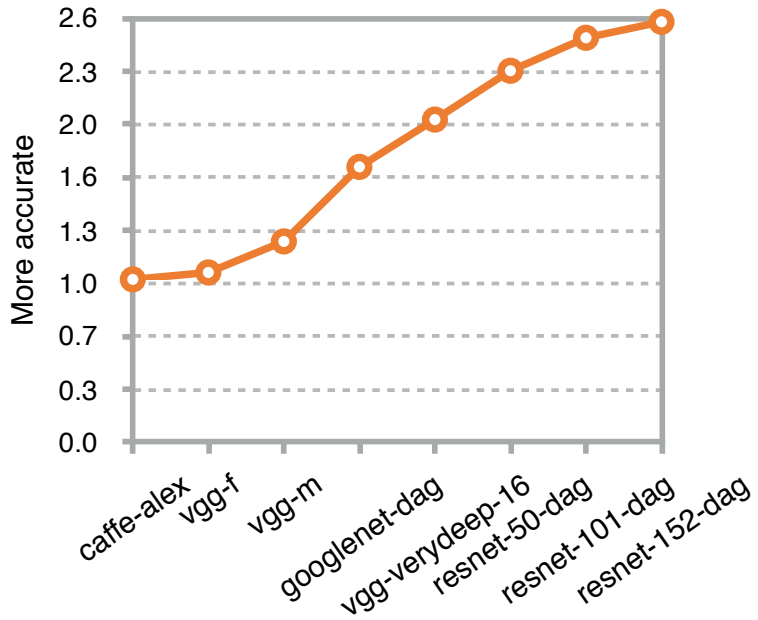
Computer Vision: ImageNet Image Classification

- 1000 classes — around 1000 images for each class



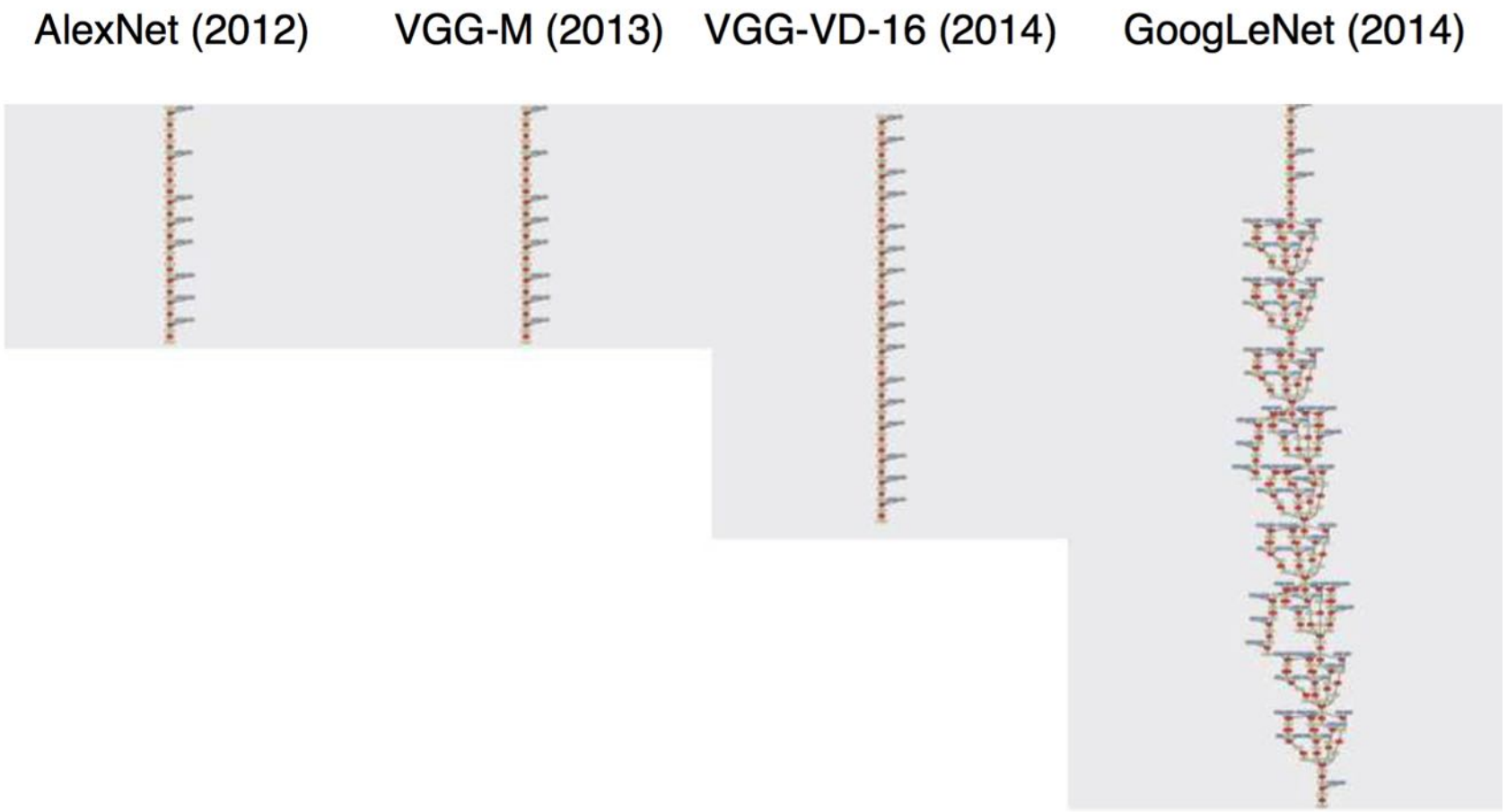
Architectures get deeper



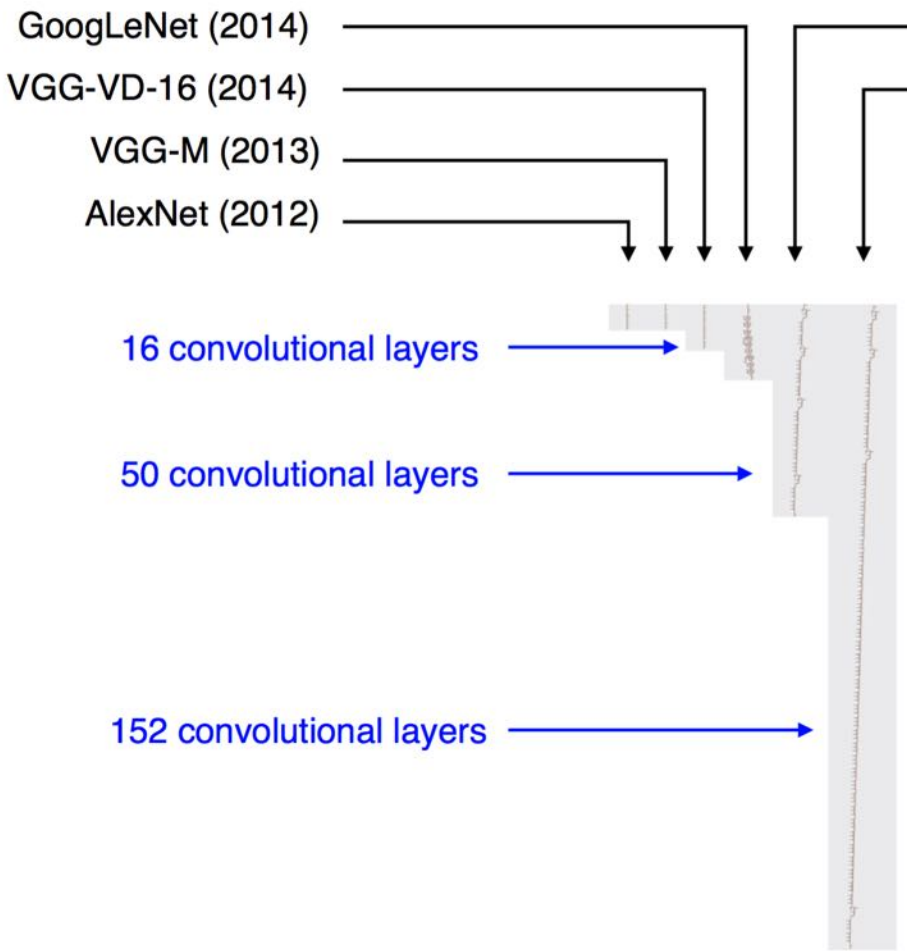


~ 2.6x improvement in 3 years

Architectures get deeper



Architectures get deeper



Krizhevsky, I. Sutskever, and G. E. Hinton. *ImageNet classification with deep convolutional neural networks.* In Proc. NIPS, 2012.

C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. *Going deeper with convolutions.* In Proc. CVPR, 2015.

K. Simonyan and A. Zisserman. *Very deep convolutional networks for large-scale image recognition.* In Proc. ICLR, 2015.

K. He, X. Zhang, S. Ren, and J. Sun. *Deep residual learning for image recognition.* In Proc. CVPR, 2016.