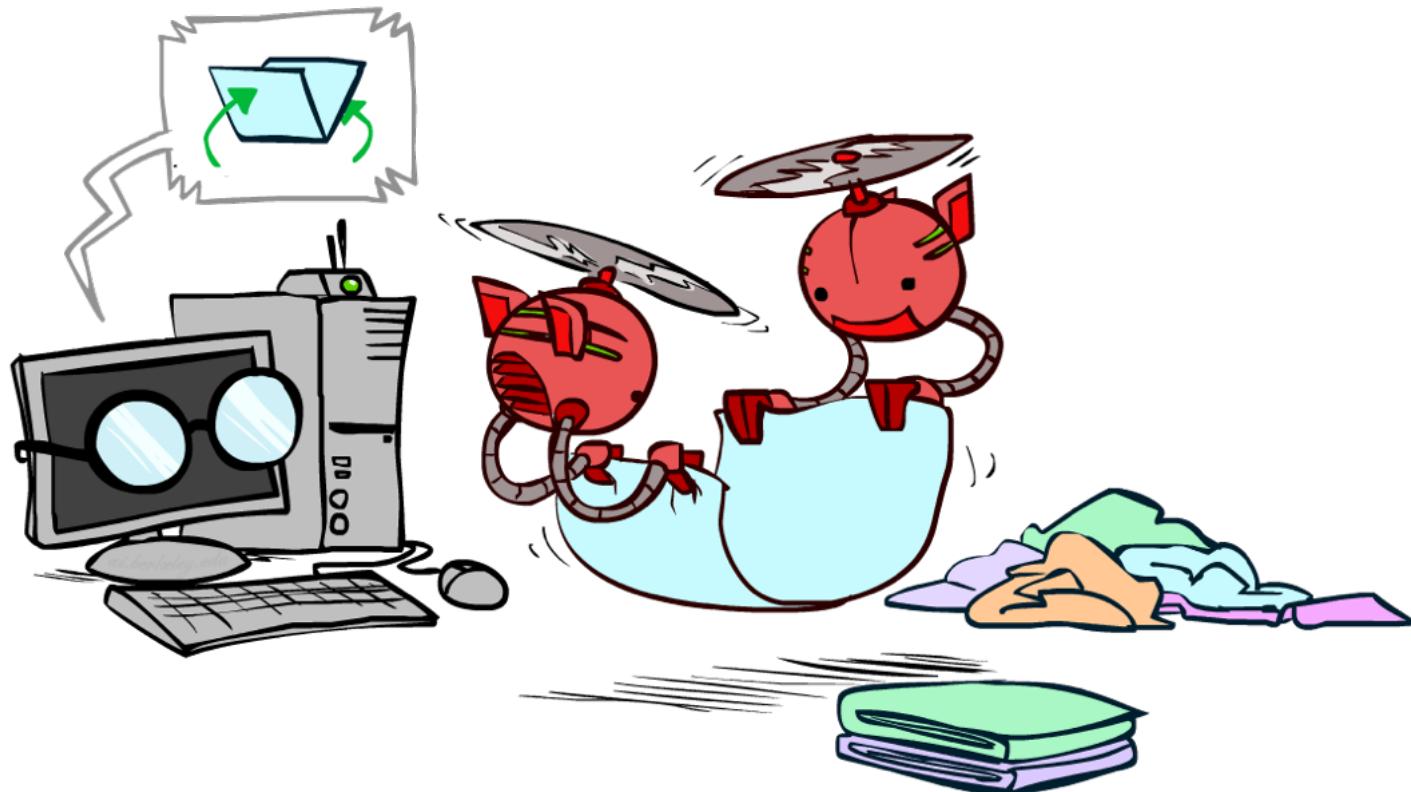


CSE 5522: Artificial Intelligence II

Advanced Applications: Computer Vision *

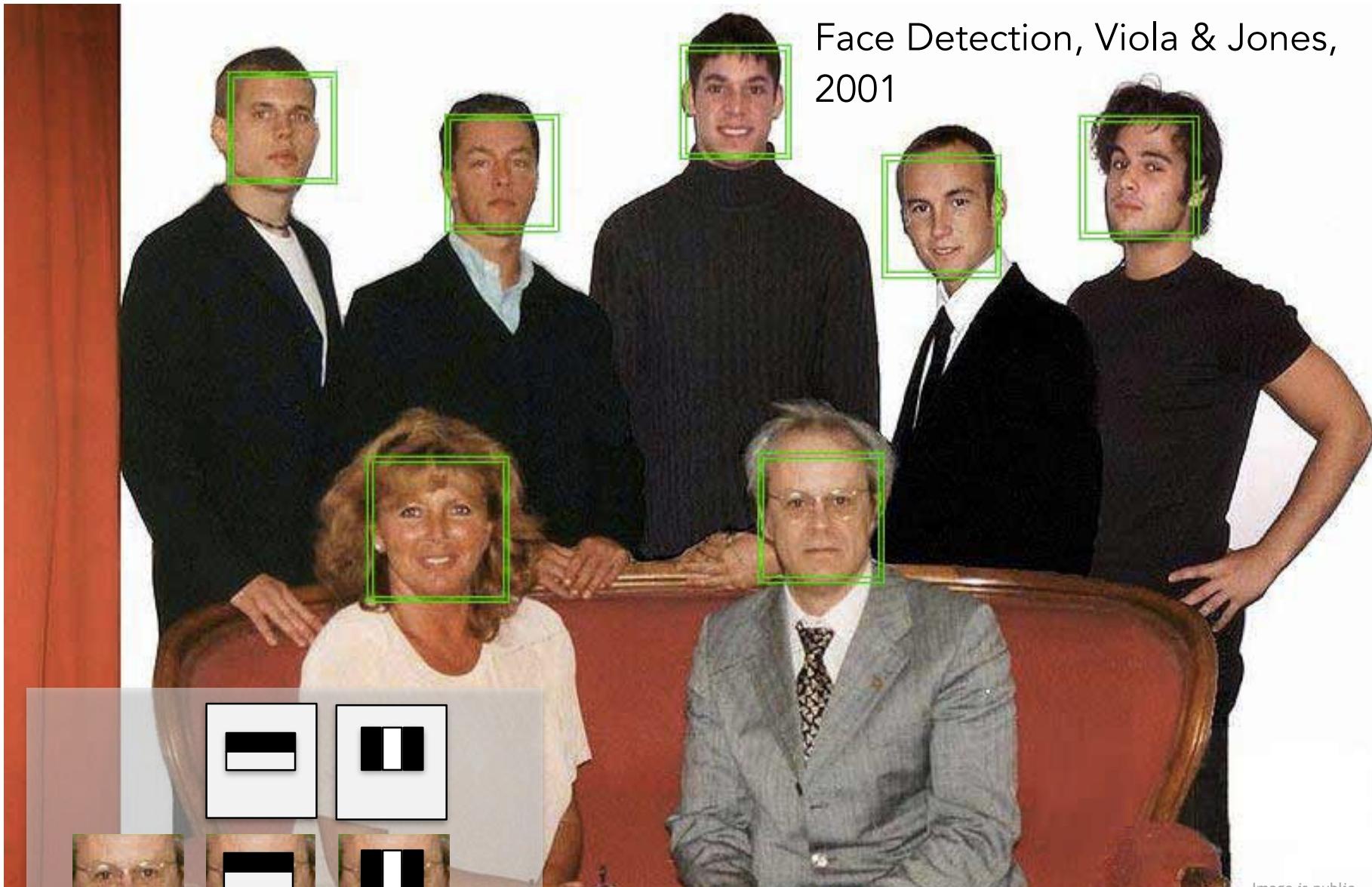


Instructor: Wei Xu

Ohio State University

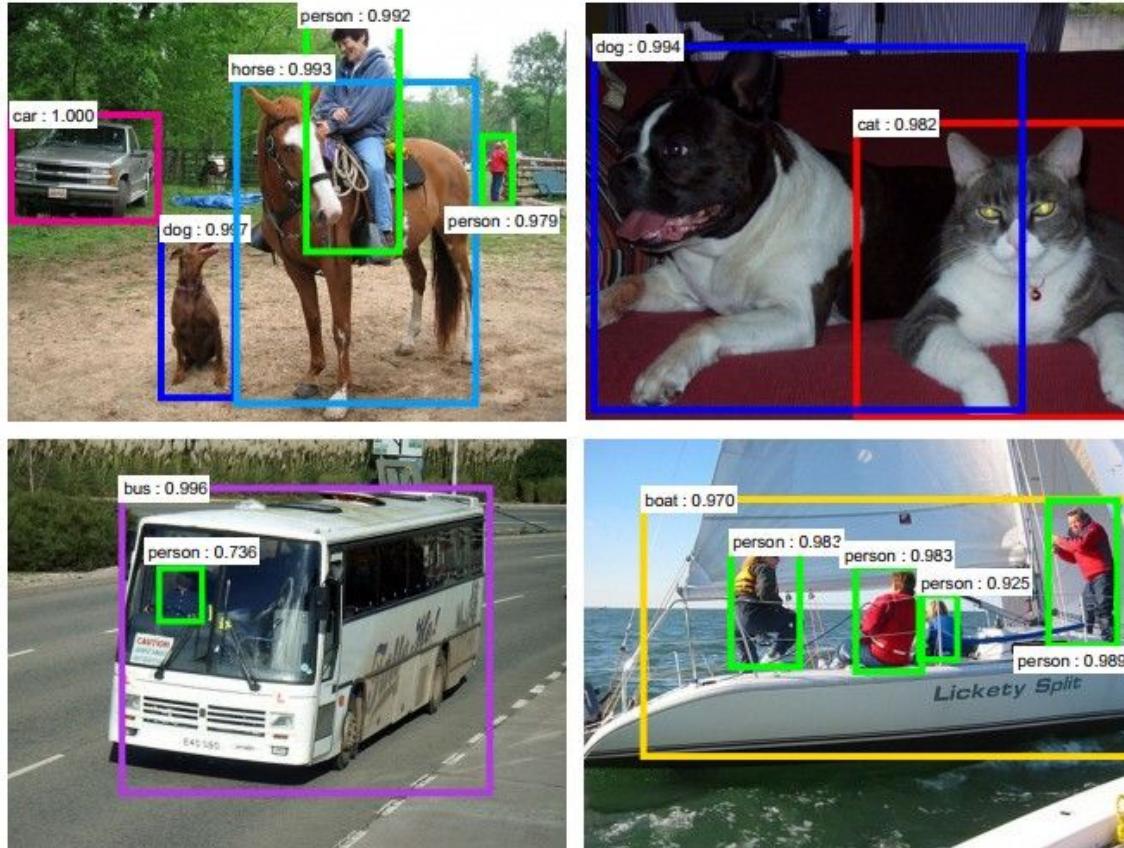
[These slides were adapted from CS231n Computer Vision at Stanford and CS188 Intro to AI at UC Berkeley]

Face Detection



Object Detection & Image Segmentation

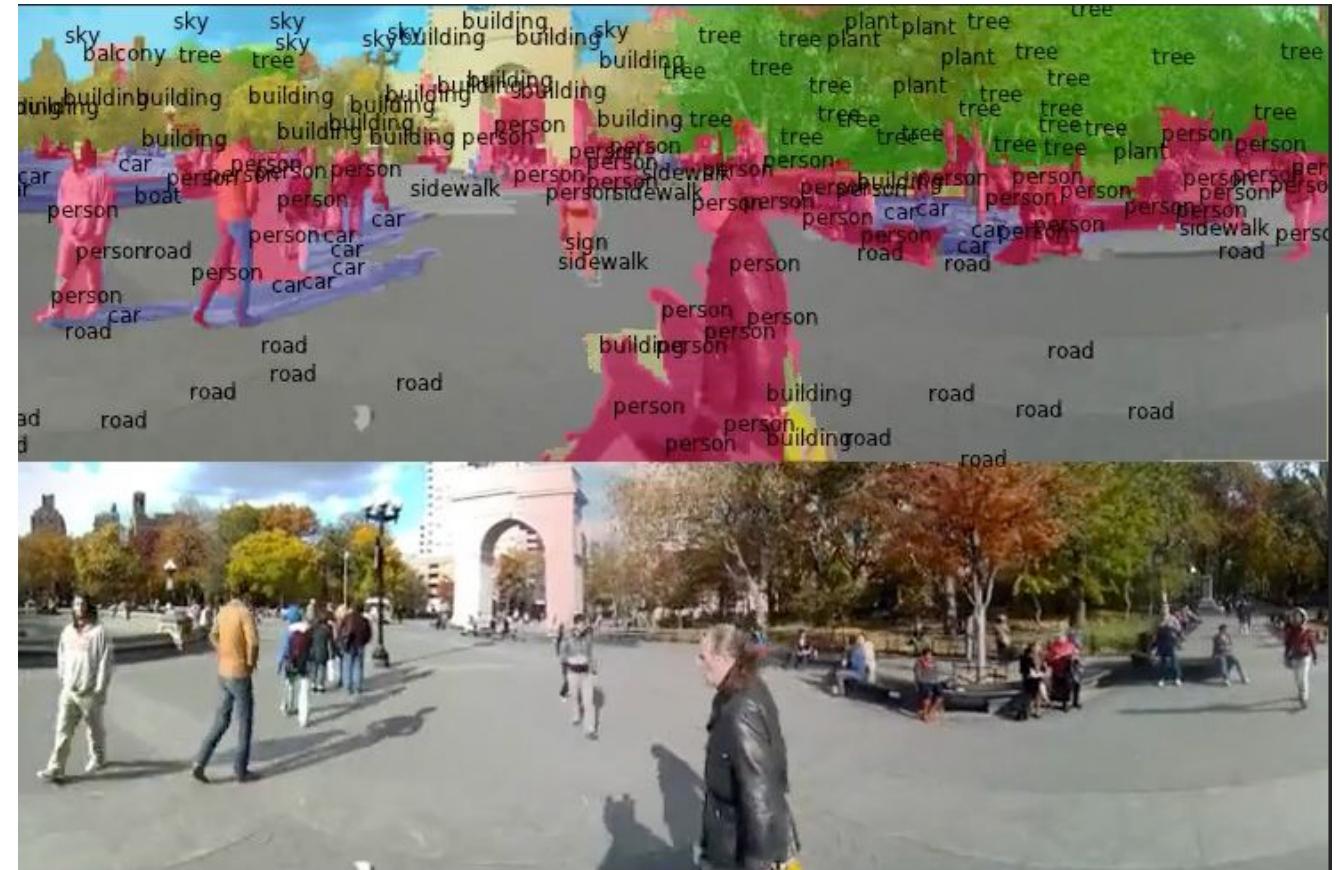
Detection



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

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

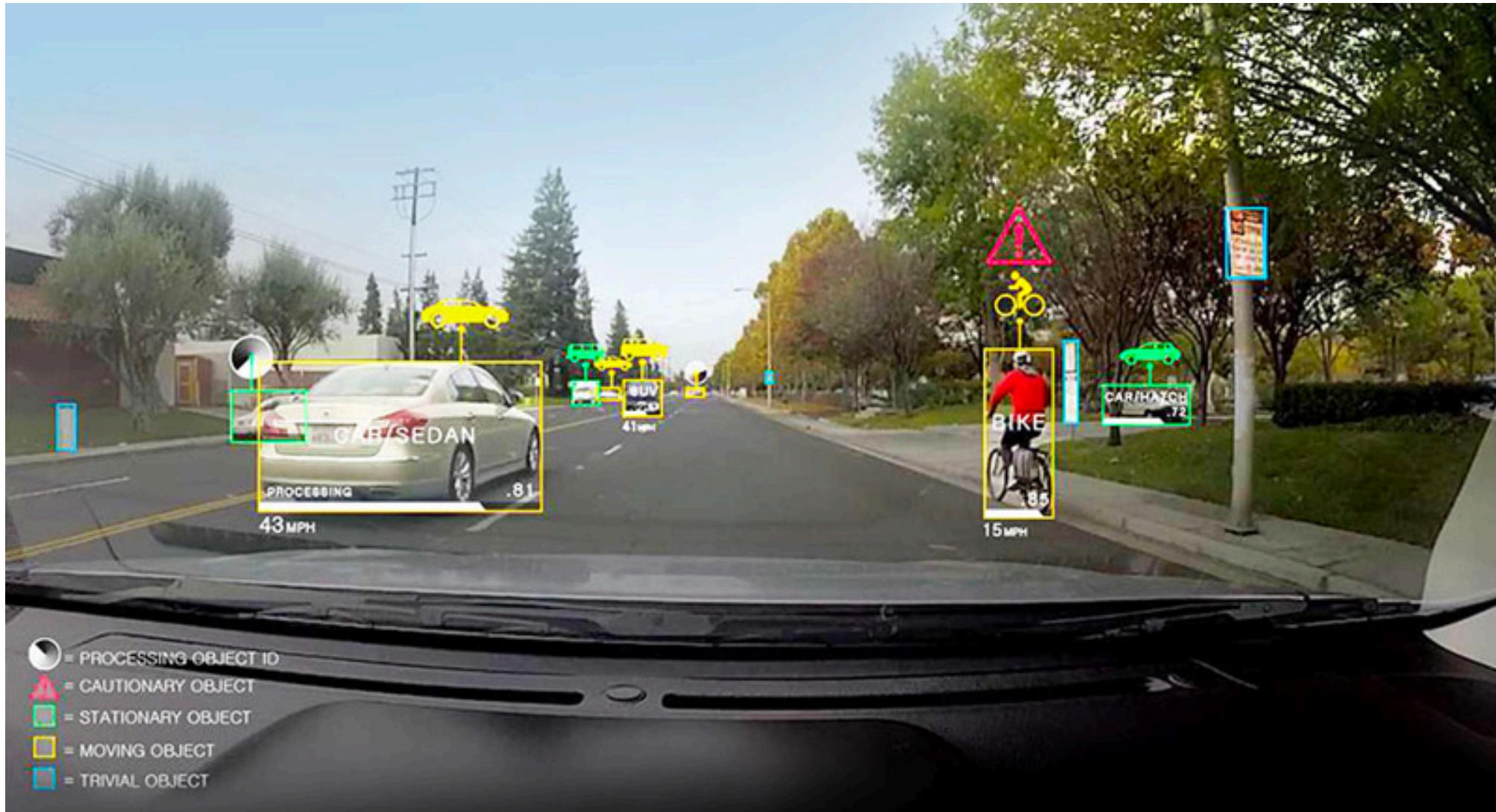
Segmentation



Figures copyright Clement Farabet, 2012.
Reproduced with permission.

[*Farabet et al., 2012*]

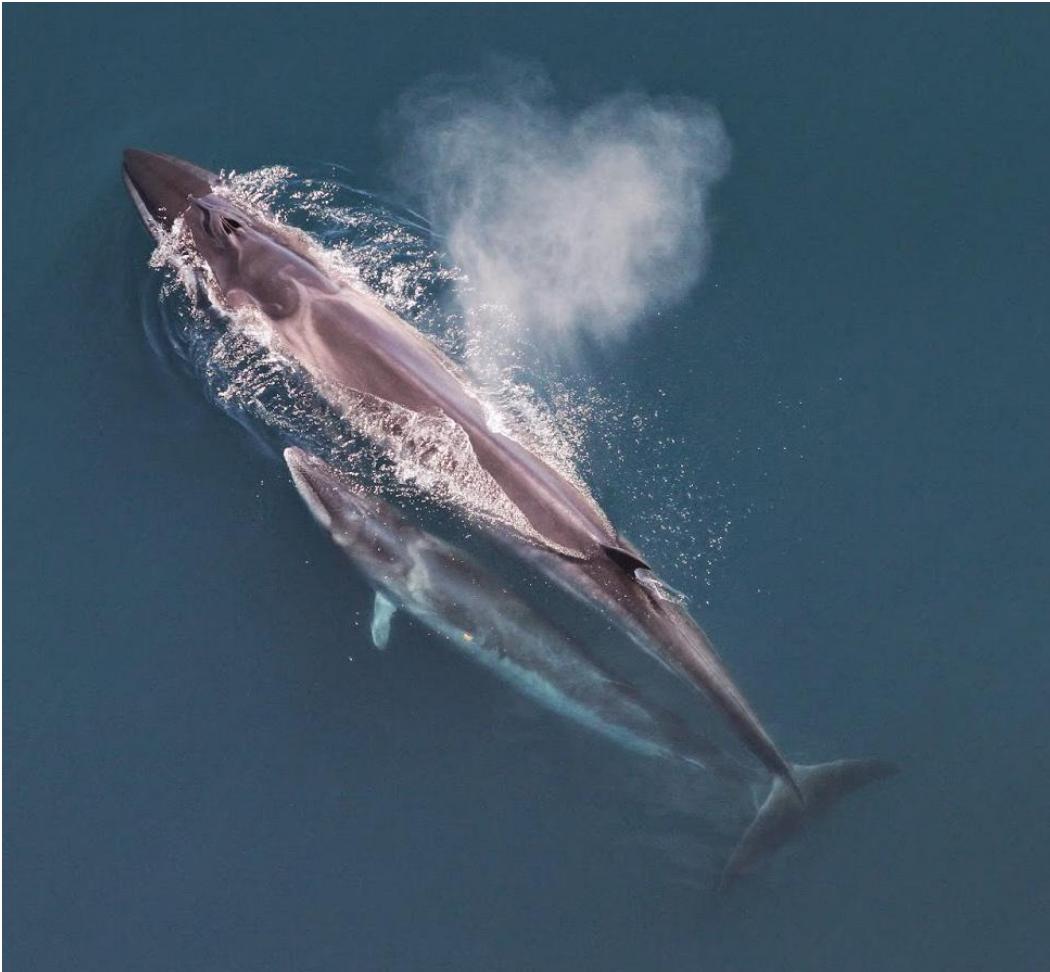
Self-driving Cars



NVIDIA DRIVE PX

Whale and Road Recognition

[This image](#) by Christin Khan is in the public domain and originally came from the U.S. NOAA.



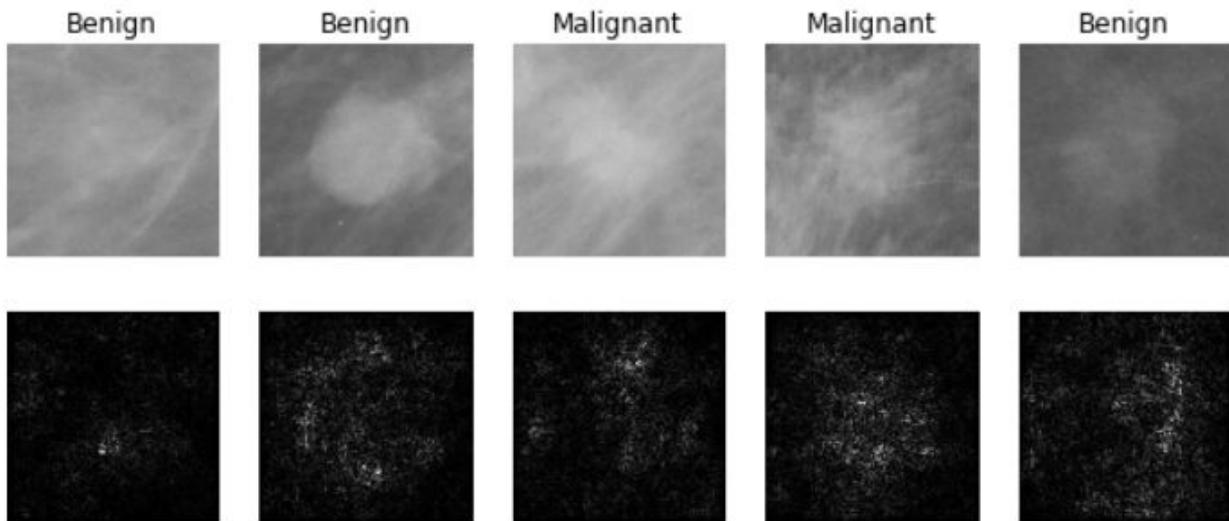
Whale recognition, Kaggle Challenge



Mnih and Hinton, 2010

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

And Many More ...



[Levy et al. 2016]

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



[Dieleman et al. 2014]



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

Photos by Lane McIntosh.
Copyright CS231n 2017.

Galaxy Morphology Prediction

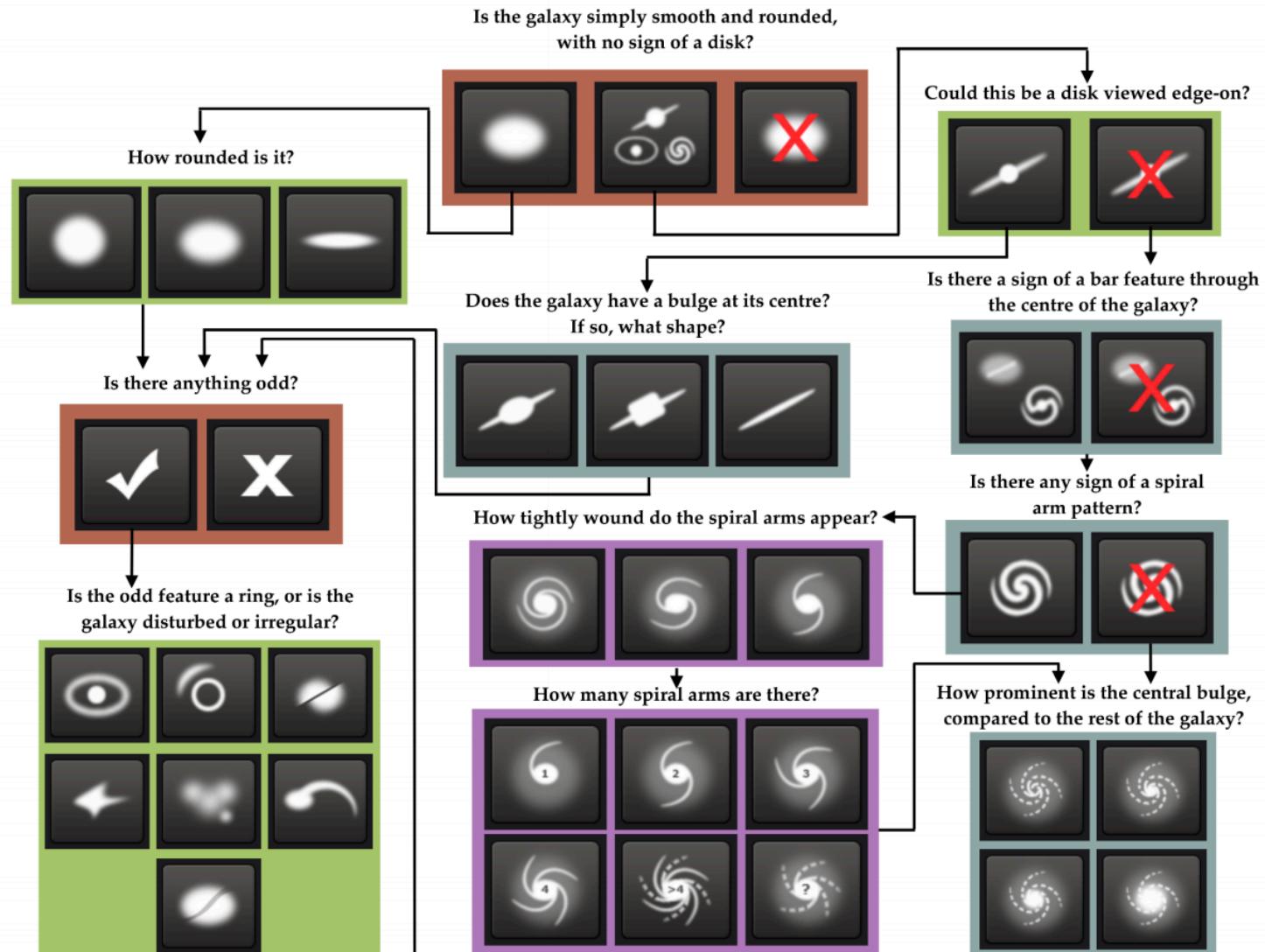


Figure 1. The Galaxy Zoo 2 decision tree. Reproduced from Figure 1 in Willett et al. (2013).

Image and Video Captioning

No errors



A white teddy bear sitting in the grass



A man riding a wave on top of a surfboard

Minor errors



A man in a baseball uniform throwing a ball



A cat sitting on a suitcase on the floor

Somewhat related



A woman is holding a cat in her hand



A woman standing on a beach holding a surfboard

Image Captioning

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

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/>

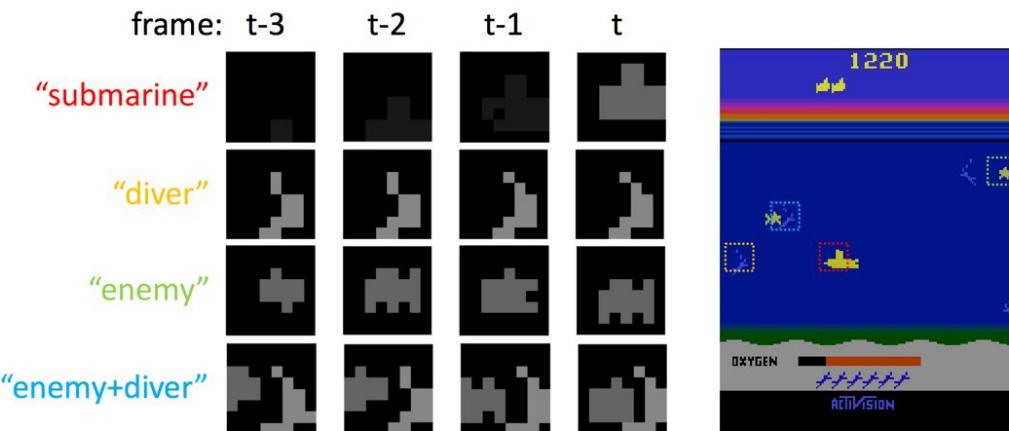
Captions generated by Justin Johnson using [Neuraltalk2](#)

Pose Estimation & Atari Games

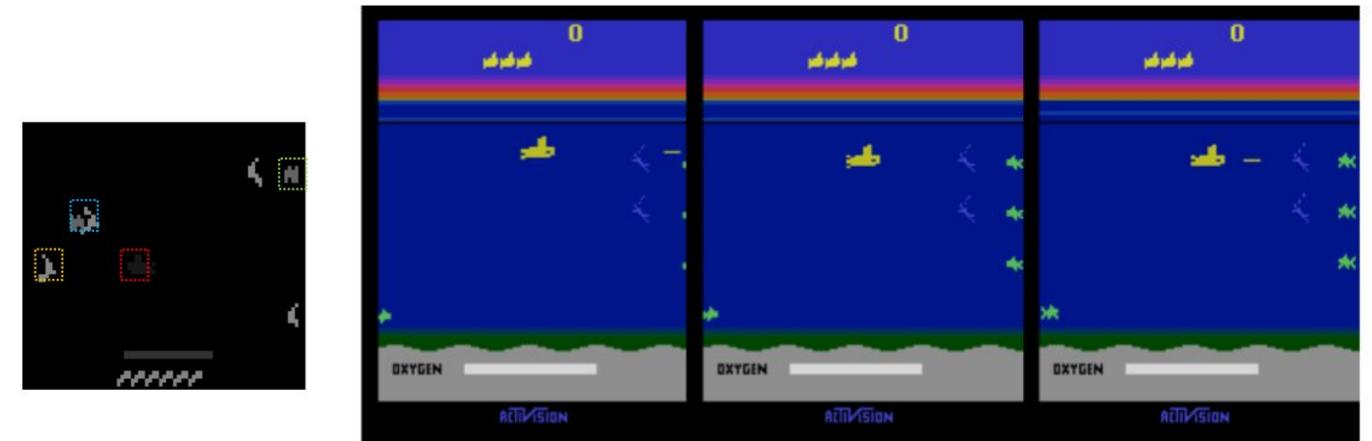


Images are examples of pose estimation, not actually from Toshev & Szegedy 2014. Copyright Lane McIntosh.

[Toshev, Szegedy 2014]

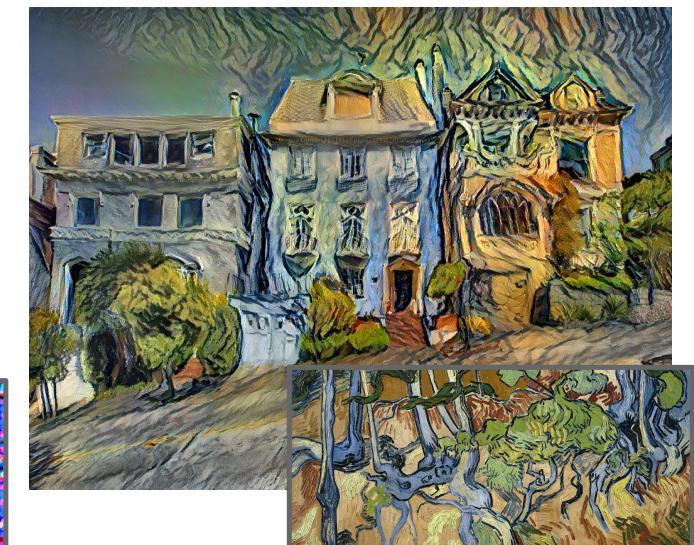
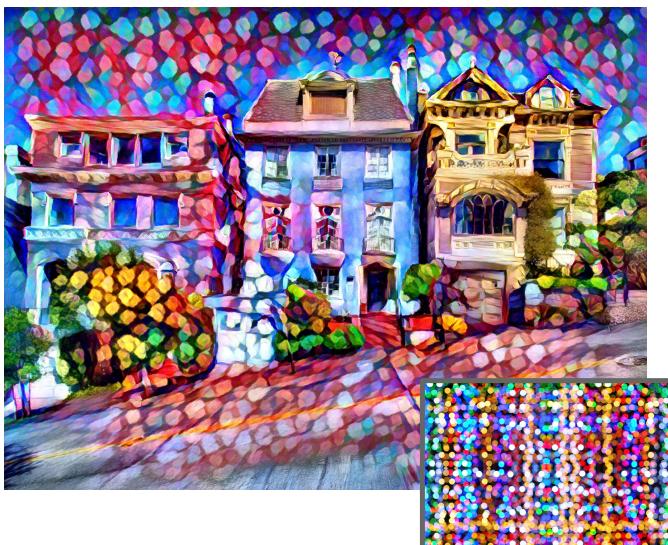
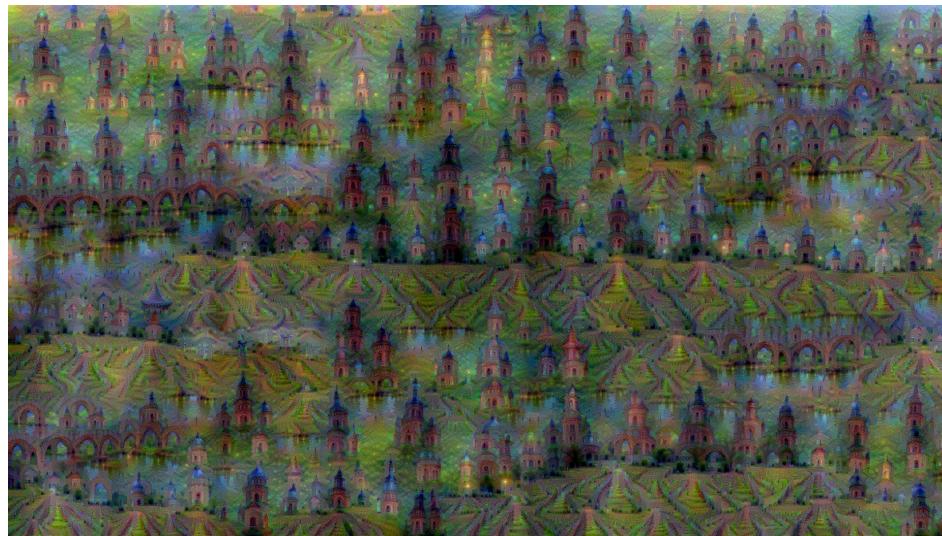
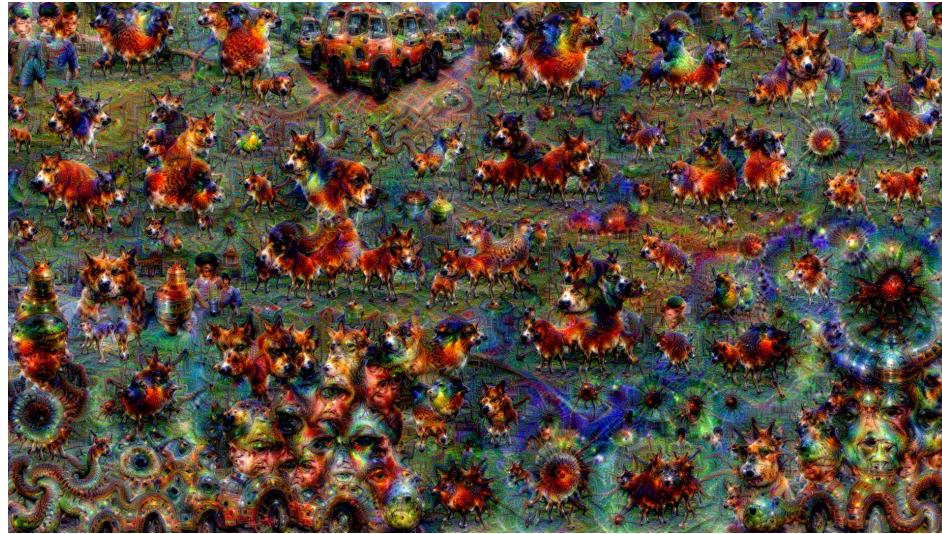


[Guo et al. 2014]



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

DeepDream and Style Transfer



[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

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

Gatys et al, "Image Style Transfer using Convolutional Neural Networks", CVPR 2016

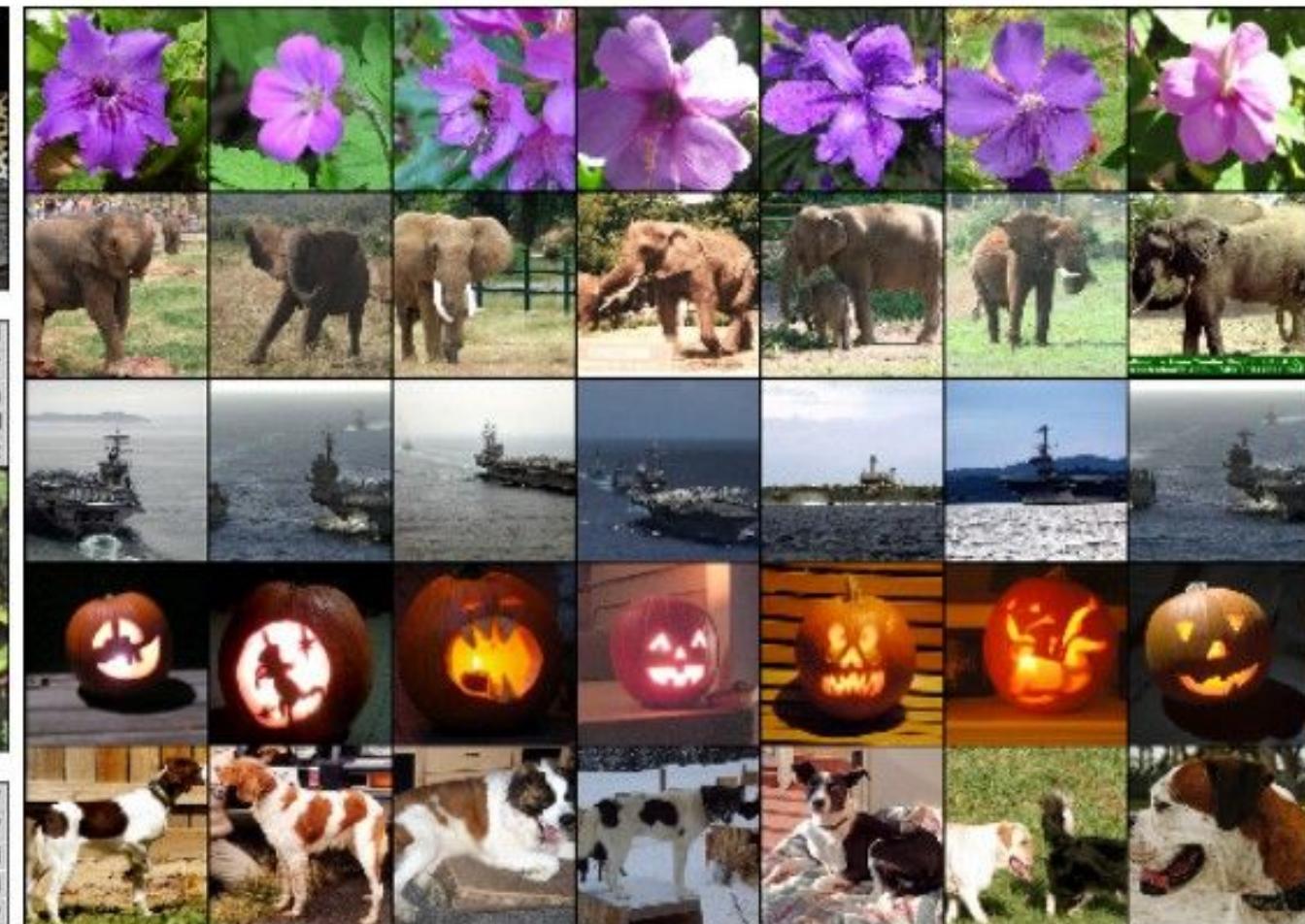
Gatys et al, "Controlling Perceptual Factors in Neural Style Transfer", CVPR 2017

Image Classification & Retrieval

Classification



Retrieval



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

PASCAL Visual Object Challenge

(20 object categories)

[Everingham et al. 2006-2012]

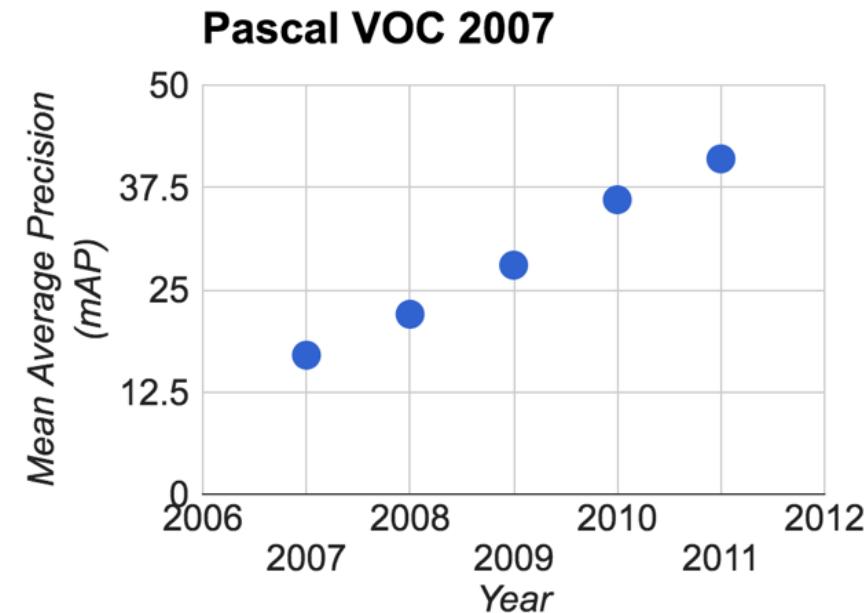
[Image](#) is CC BY-SA 3.0



[Image](#) is CC0 1.0 public domain



[This image](#) is licensed under
CC BY-SA 2.0; changes made



ImageNet



www.image-net.org

22K categories and **14M** images

- Animals
 - Bird
 - Fish
 - Mammal
 - Invertebrate
- Plants
 - Tree
 - Flower
 - Food
 - Materials
- Structures
 - Artifact
 - Tools
 - Appliances
 - Structures
- Person
- Scenes
 - Indoor
 - Geological Formations
- Sport Activities



ImageNet Large Scale Visual Recognition Challenge

Steel drum

The Image Classification Challenge:

1,000 object classes

1,431,167 images



Output:

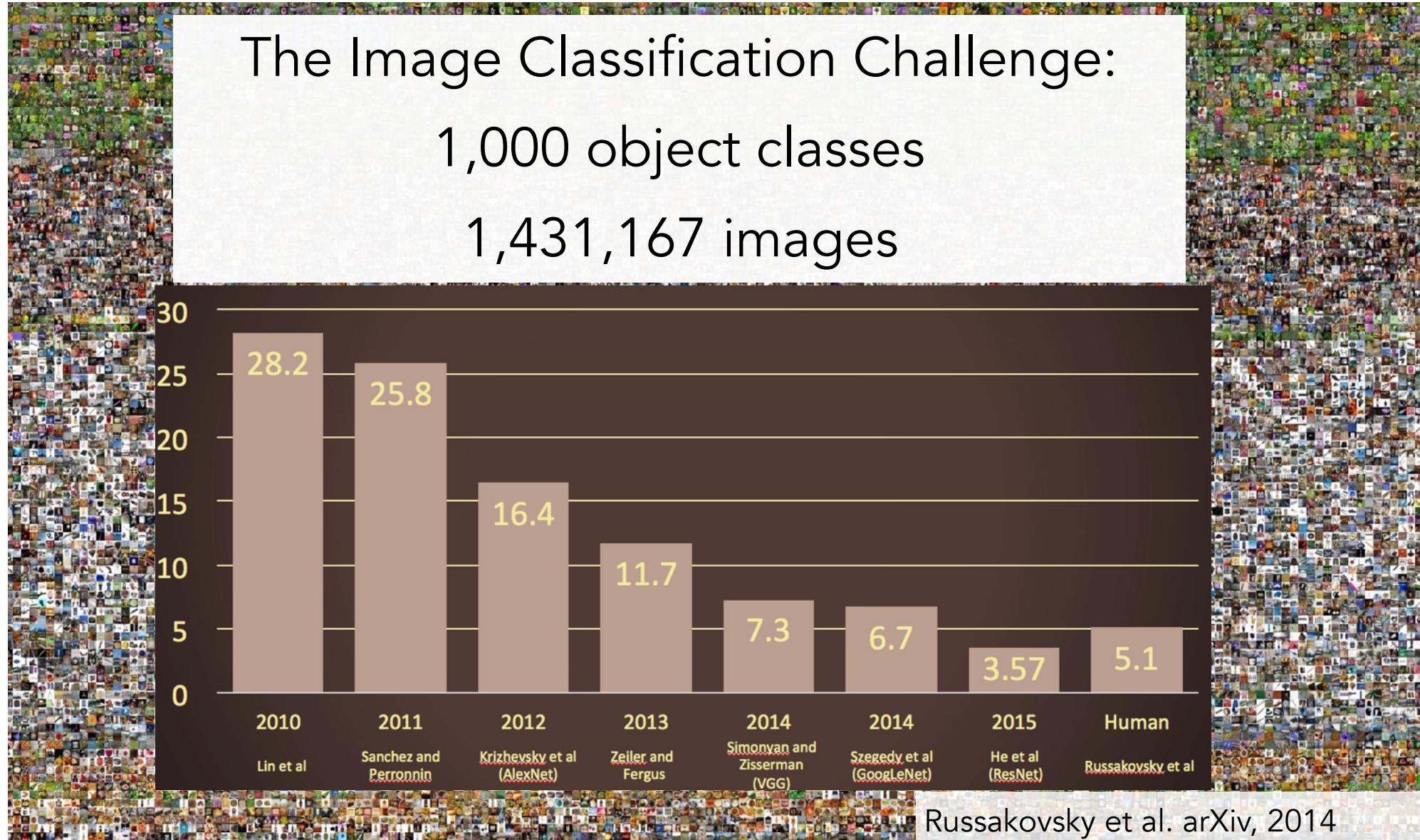
- Scale
- T-shirt
- Steel drum
- Drumstick
- Mud turtle

Output:

- Scale
- T-shirt
- Giant panda
- Drumstick
- Mud turtle

Russakovsky et al. arXiv, 2014

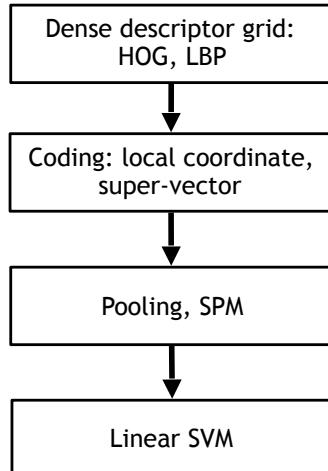
ImageNet Large Scale Visual Recognition Challenge



ImageNet Large Scale Visual Recognition Challenge

Year 2010

NEC-UIUC

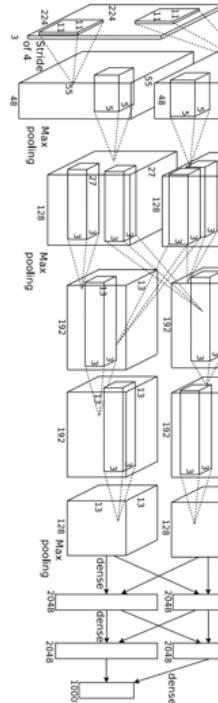


[Lin CVPR 2011]

[Lion image](#) by Swissfrog is licensed under [CC BY 3.0](#)

Year 2012

SuperVision



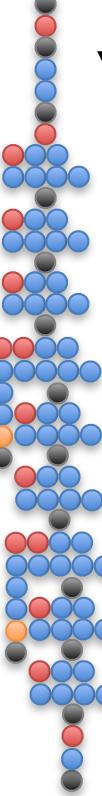
[Krizhevsky NIPS 2012]

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

Year 2014

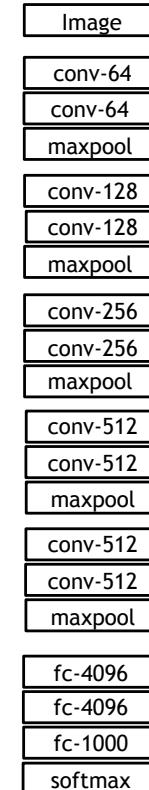
GoogLeNet

- Pooling
- Convolution
- Softmax
- Other



[Szegedy arxiv 2014]

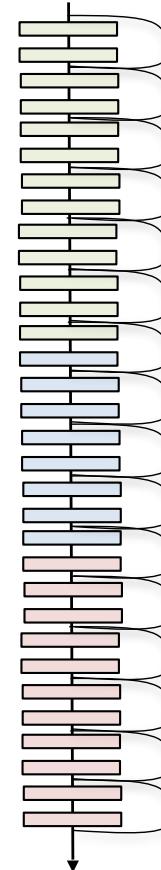
VGG



[Simonyan arxiv 2014]

Year 2015

MSRA

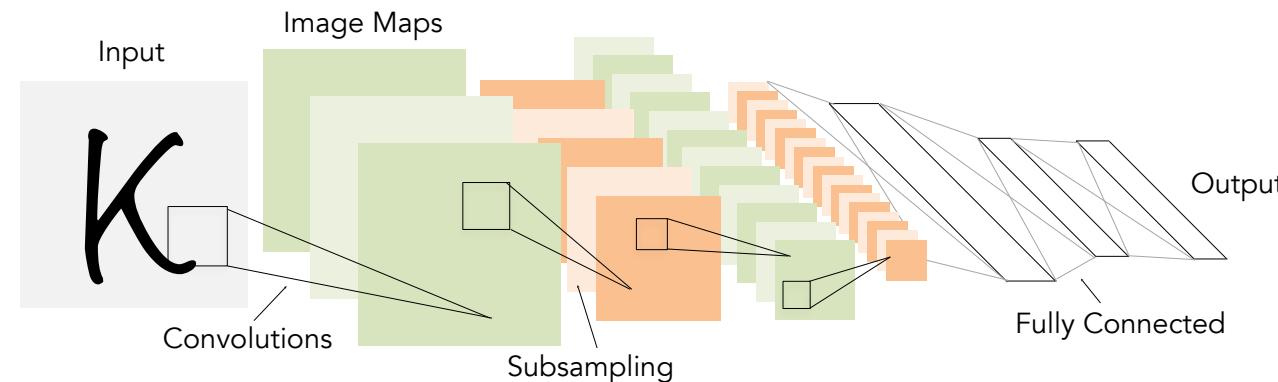


[He ICCV 2015]

Convolutional Neural Networks (CNN)

1998

LeCun et al.



of transistors



10^6

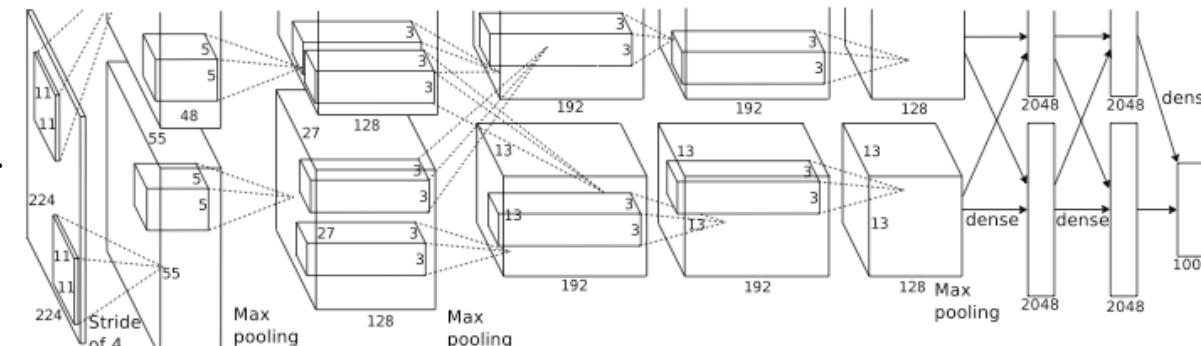
pentium[®] II

of pixels used in training

10^7 **NIST**

2012

Krizhevsky et al.



of transistors



10^9

GPUs



of pixels used in training

10^{14} **IMAGENET**

Image Classification



This image by Nikita is
licensed under CC-BY 2.0

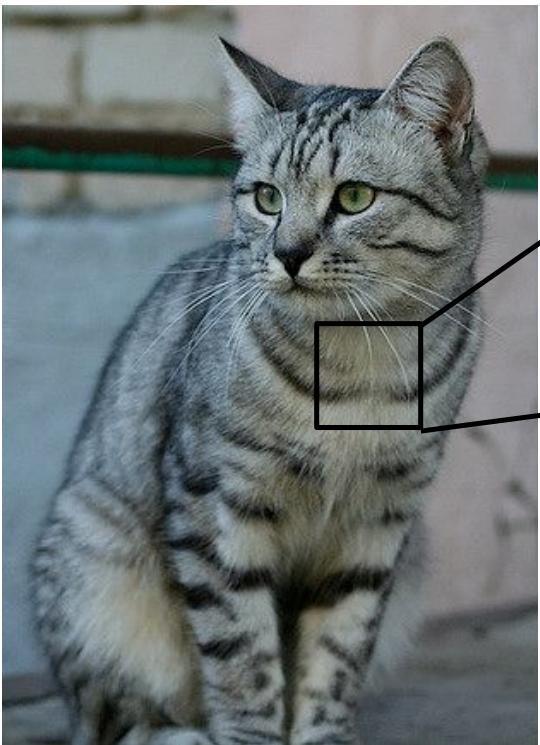
(assume given set of discrete labels)
{dog, cat, truck, plane, ...}



cat

Image Representation

The Problem: Semantic Gap



This image by Nikita is
licensed under [CC-BY 2.0](#)

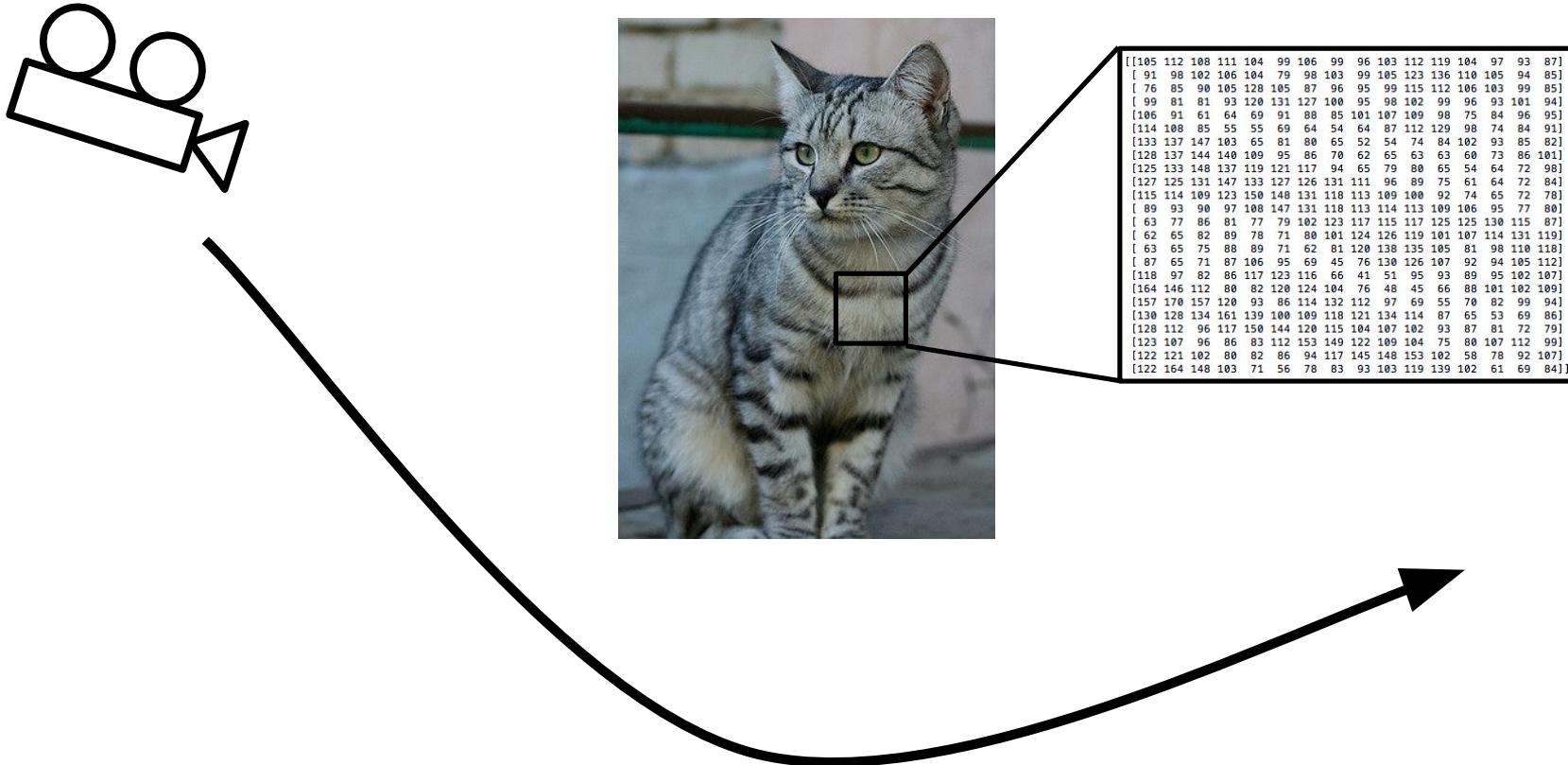
[105 112 108 111 104 99 106 99 96 103 112 119 104 97 93 87]
[91 98 102 106 104 79 98 103 99 105 123 136 110 105 94 85]
[76 85 90 105 128 105 87 96 95 99 115 112 106 103 99 85]
[99 81 81 93 120 131 127 100 95 98 102 99 96 93 101 94]
[106 91 61 64 69 91 88 85 101 107 109 98 75 84 96 95]
[114 108 85 55 55 69 64 54 64 87 112 129 98 74 84 91]
[133 137 147 103 65 81 80 65 52 54 74 84 102 93 85 82]
[128 137 144 140 109 95 86 70 62 65 63 63 60 73 86 101]
[125 133 148 137 119 121 117 94 65 79 80 65 54 64 72 98]
[127 125 131 147 133 127 126 131 111 96 89 75 61 64 72 84]
[115 114 109 123 150 148 131 118 113 109 100 92 74 65 72 78]
[89 93 90 97 108 147 131 118 113 114 113 109 106 95 77 80]
[63 77 86 81 77 79 102 123 117 115 117 125 125 130 115 87]
[62 65 82 89 78 71 80 101 124 126 119 101 107 114 131 119]
[63 65 75 88 89 71 62 81 120 138 135 105 81 98 110 118]
[87 65 71 87 106 95 69 45 76 130 126 107 92 94 105 112]
[118 97 82 86 117 123 116 66 41 51 95 93 89 95 102 107]
[164 146 112 80 82 120 124 104 76 48 45 66 88 101 102 109]
[157 170 157 120 93 86 114 132 112 97 69 55 70 82 99 94]
[130 128 134 161 139 100 109 118 121 134 114 87 65 53 69 86]
[128 112 96 117 150 144 120 115 104 107 102 93 87 81 72 79]
[123 107 96 86 83 112 153 149 122 109 104 75 80 107 112 99]
[122 121 102 80 82 86 94 117 145 148 153 102 58 78 92 107]
[122 164 148 103 71 56 78 83 93 103 119 139 102 61 69 84]

What the computer sees

An image is just a big grid of numbers between [0, 255]:

e.g. 800 x 600 x 3
(3 channels RGB)

Challenges: Viewpoint variation



Challenges: Illumination



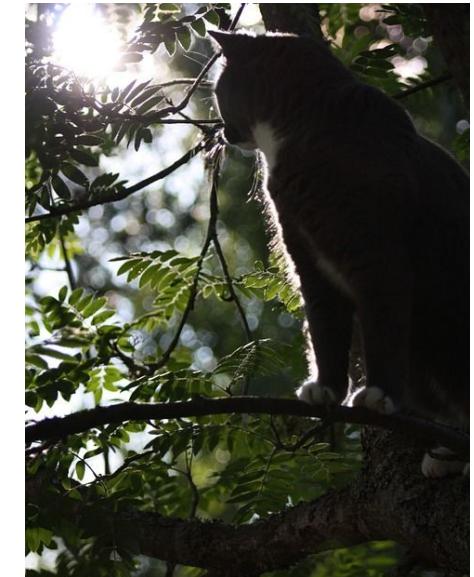
[This image is CC0 1.0 public domain](#)



[This image is CC0 1.0 public domain](#)



[This image is CC0 1.0 public domain](#)

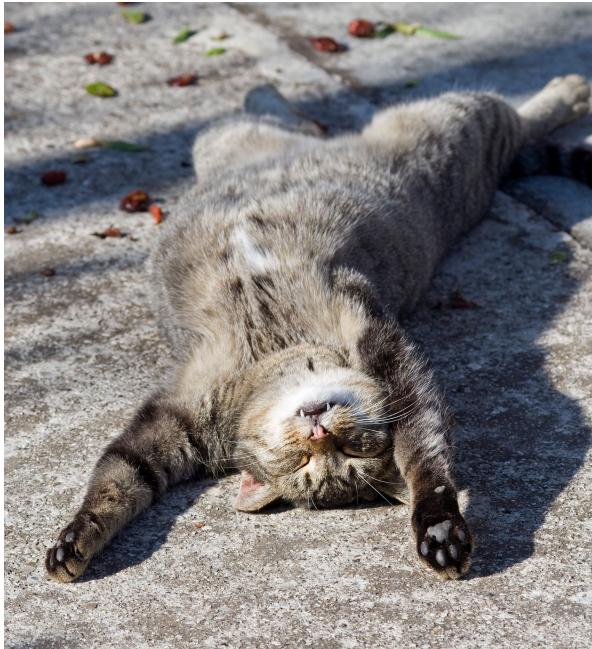


[This image is CC0 1.0 public domain](#)

Challenges: Deformation



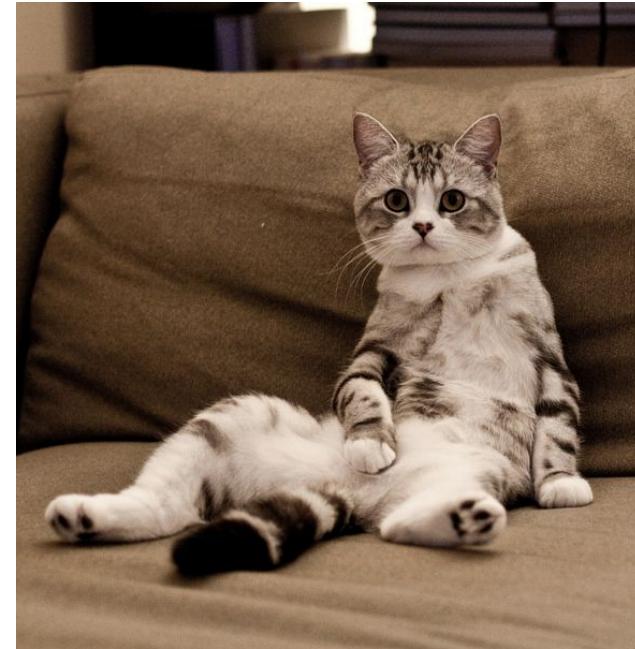
[This image by Umberto Salvagnin](#)
is licensed under [CC-BY 2.0](#)



[This image by Umberto Salvagnin](#)
is licensed under [CC-BY 2.0](#)



[This image by sare bear](#) is
licensed under [CC-BY 2.0](#)



[This image by Tom Thai](#) is
licensed under [CC-BY 2.0](#)

Challenges: Occlusion



This image is [CC0 1.0 public domain](#)



This image is [CC0 1.0 public domain](#)



This image by [jonsson](#) is licensed
under [CC-BY 2.0](#)

Challenges: Background clutter



This image is [CC0 1.0 public domain](#)



This image is [CC0 1.0 public domain](#)

Challenges: Intraclass variation



This image is CC0 1.0 public domain



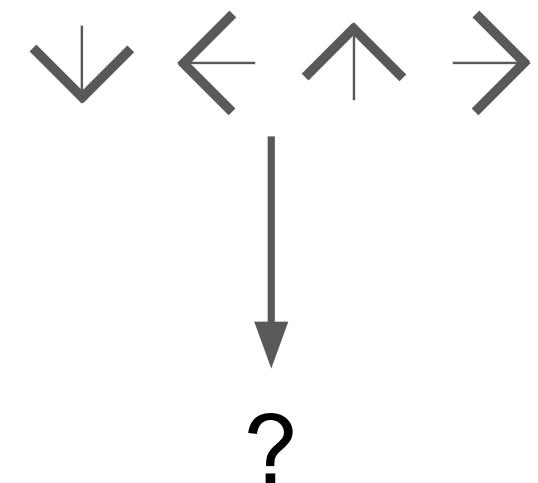
How do we do this?



Find edges



Find corners



Machine Learning

Data-Driven Approach

1. Collect a dataset of images and labels
2. Use Machine Learning to train a classifier
3. Evaluate the classifier on new images

Example training set

```
def train(images, labels):
    # Machine learning!
    return model
```

```
def predict(model, test_images):
    # Use model to predict labels
    return test_labels
```

airplane



automobile



bird



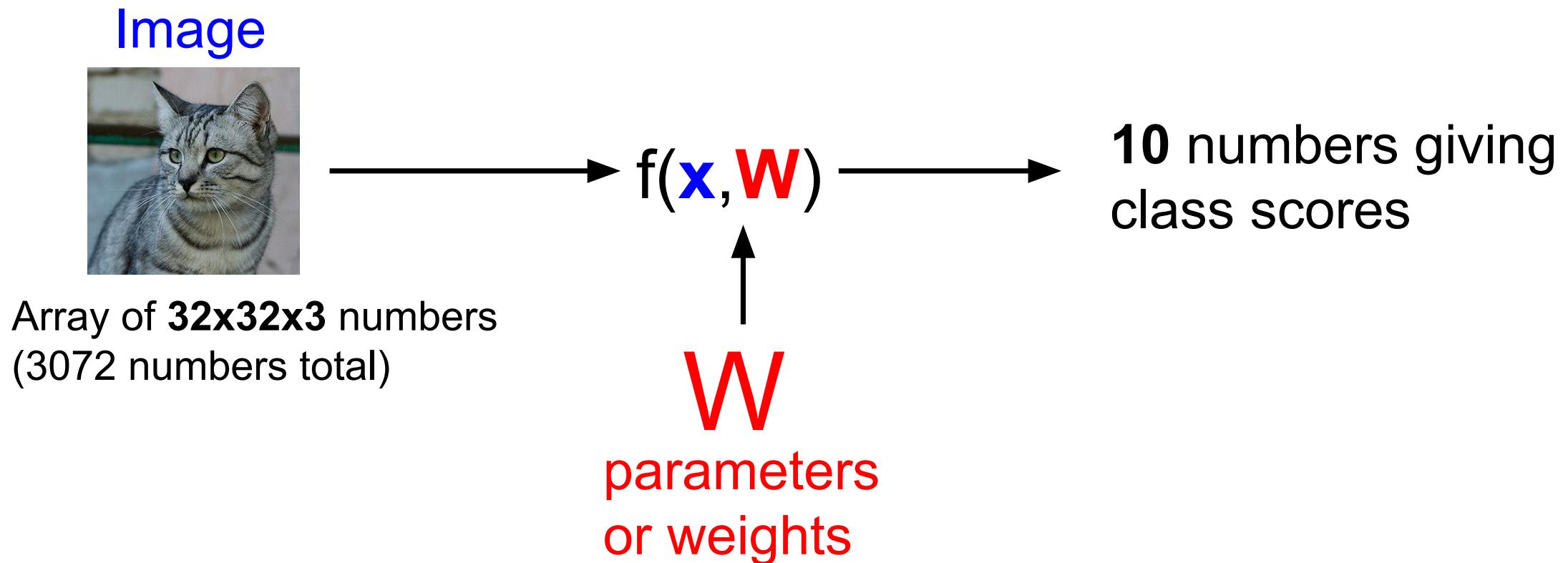
cat



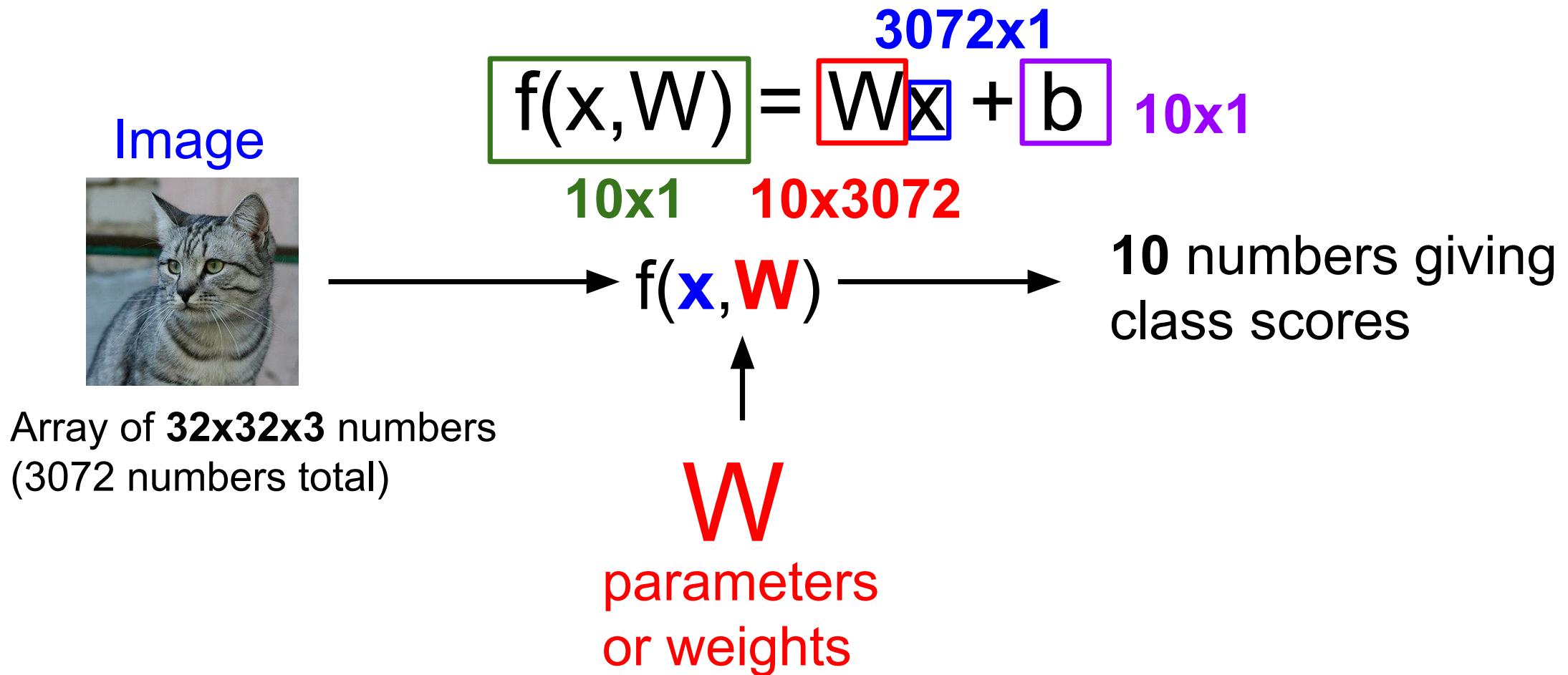
deer



Linear Classifier

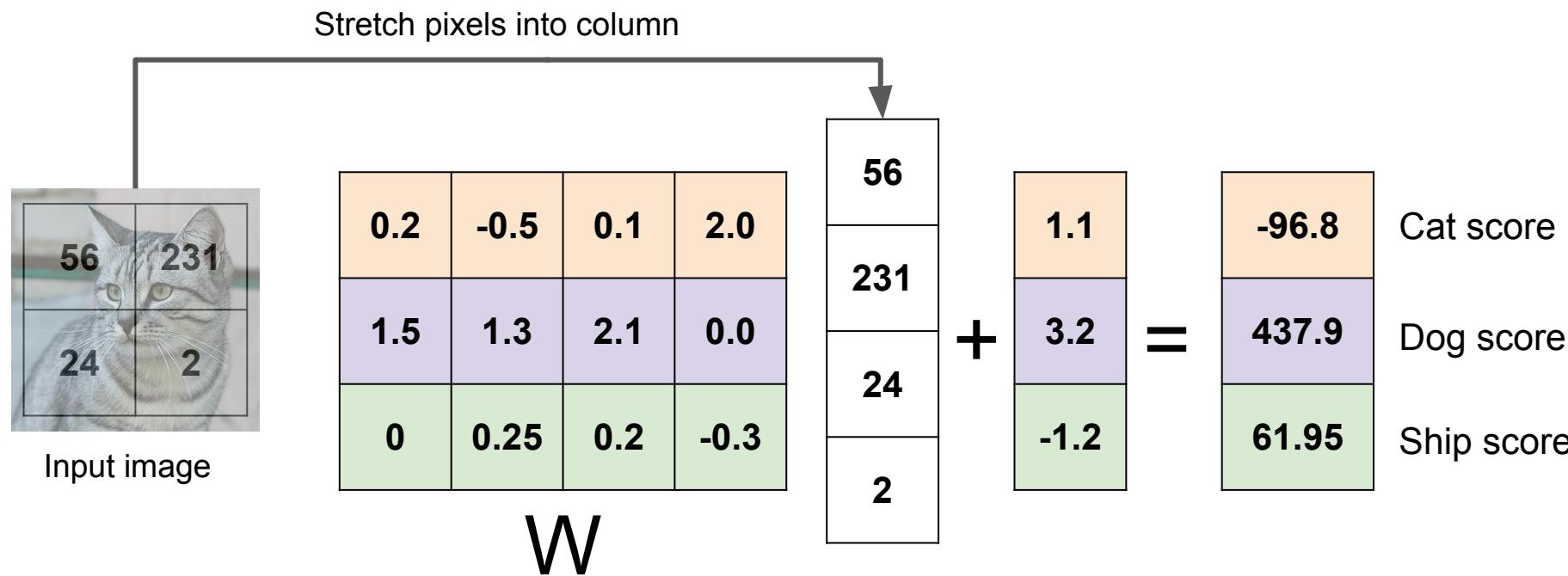


Linear Classifier



Linear Classifier

Example with an image with 4 pixels, and 3 classes (cat/dog/ship)



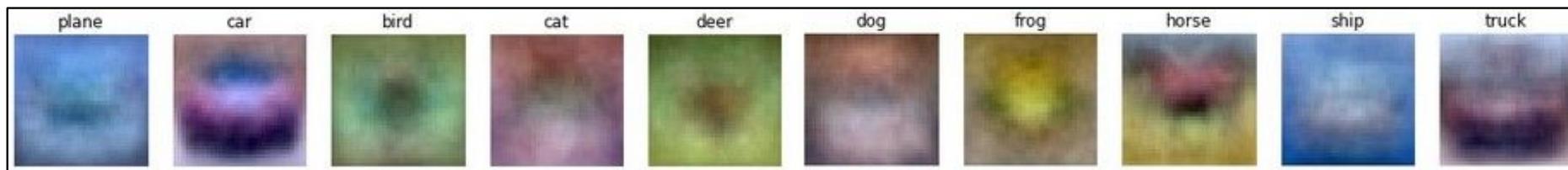
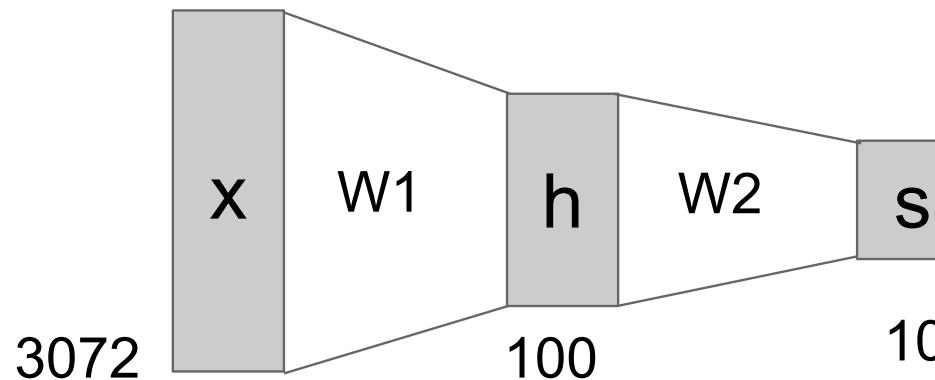
Neural Networks

Linear score function:

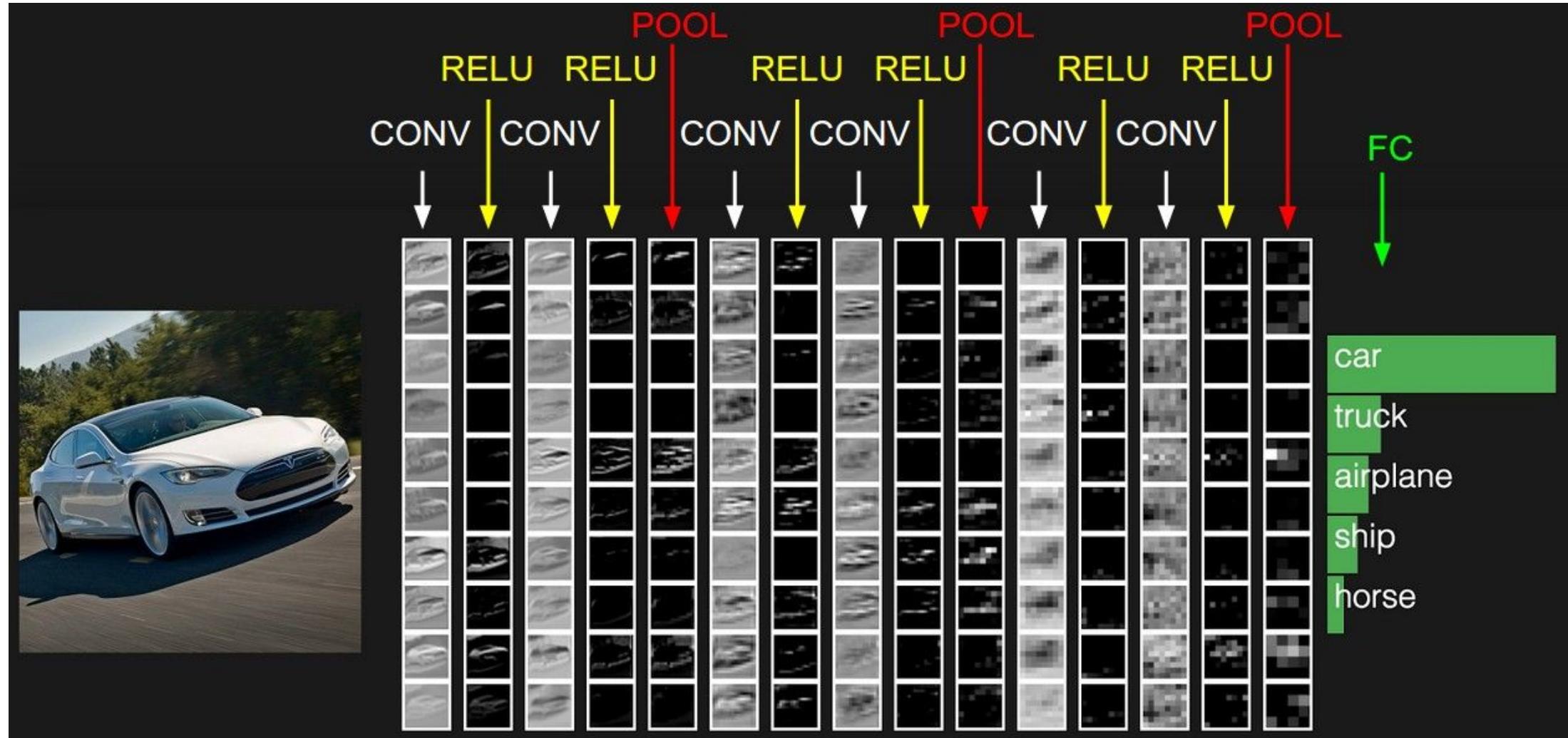
$$f = Wx$$

2-layer Neural Network

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

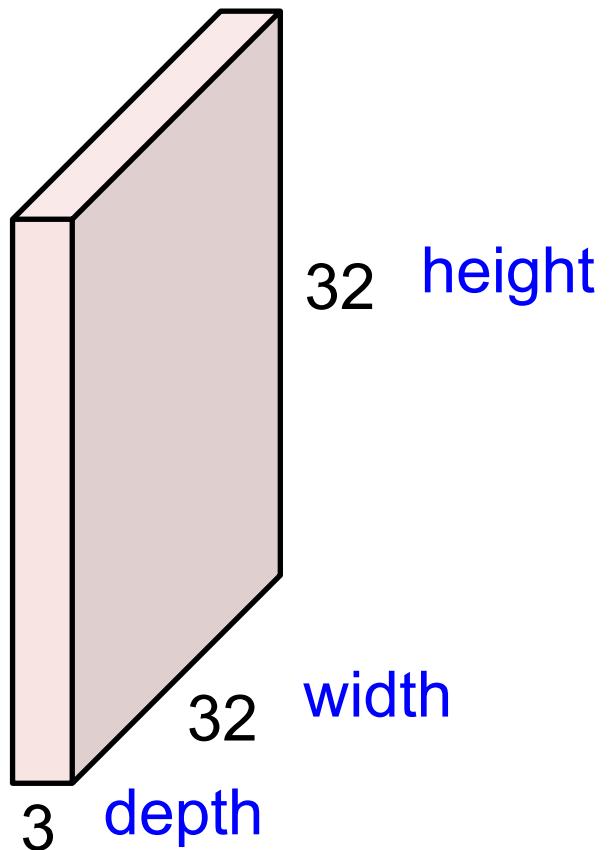


Convolutional Neural Networks (CNN)



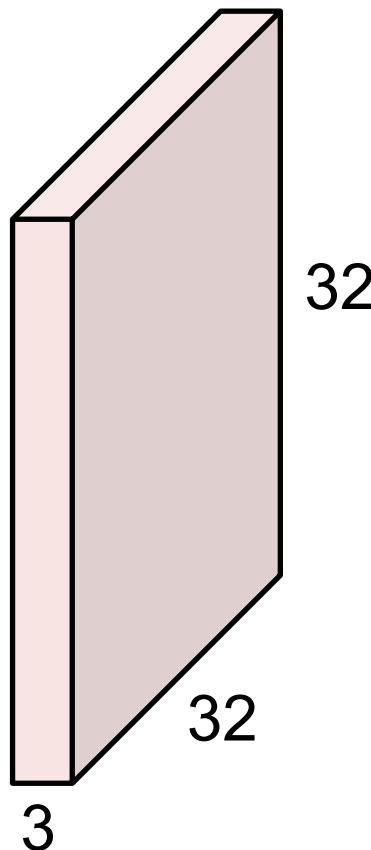
Convolution Layer

32x32x3 image

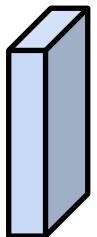


Convolution Layer

32x32x3 image

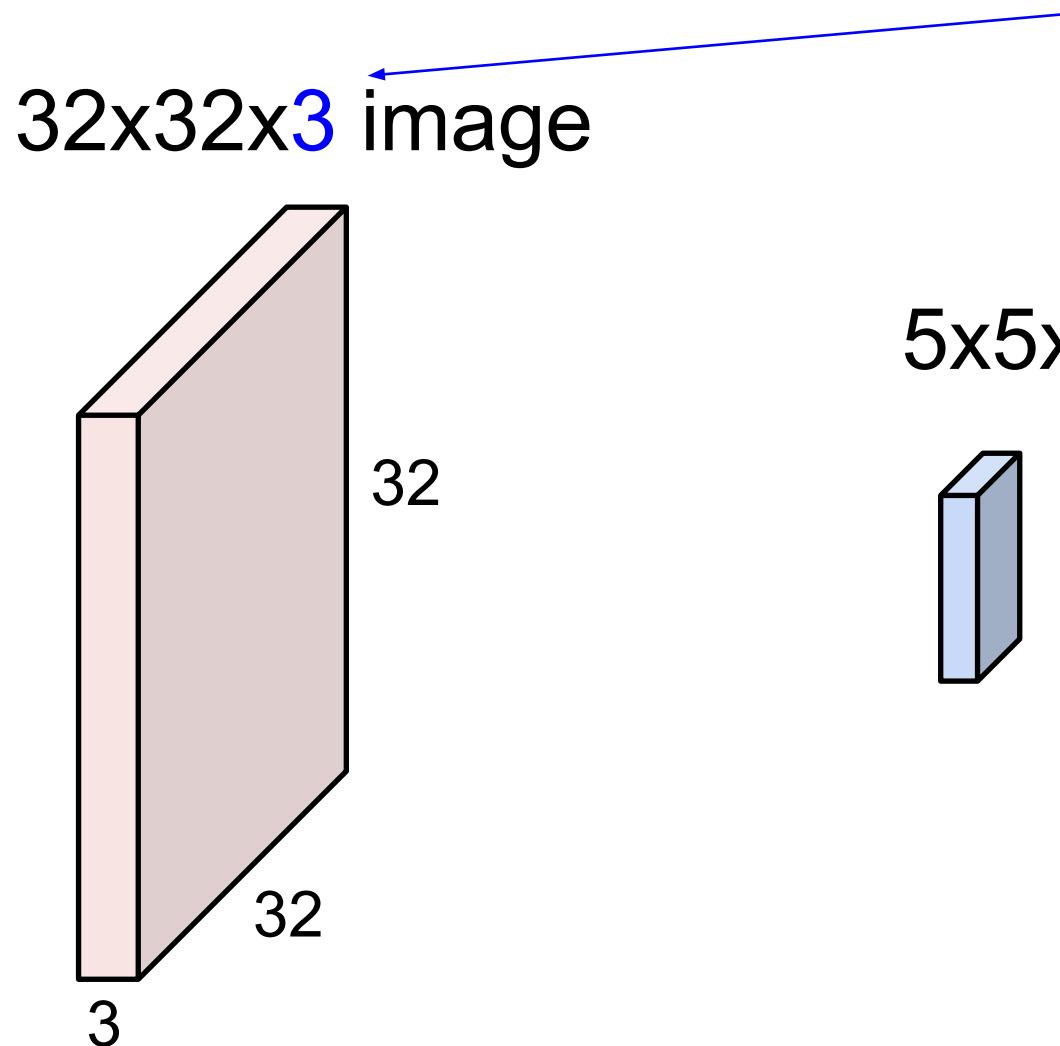


5x5x3 filter



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

Convolution Layer

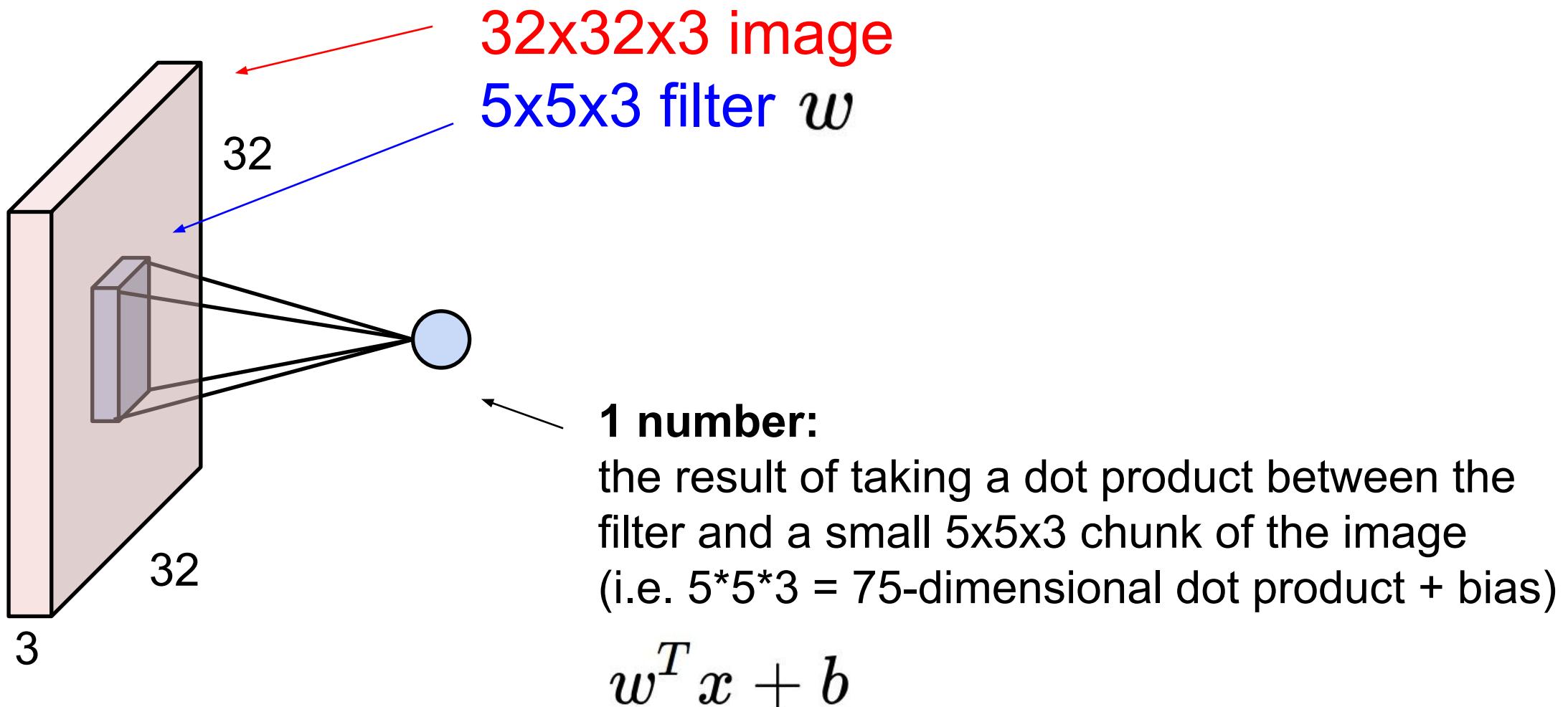


Filters always extend the full depth of the input volume

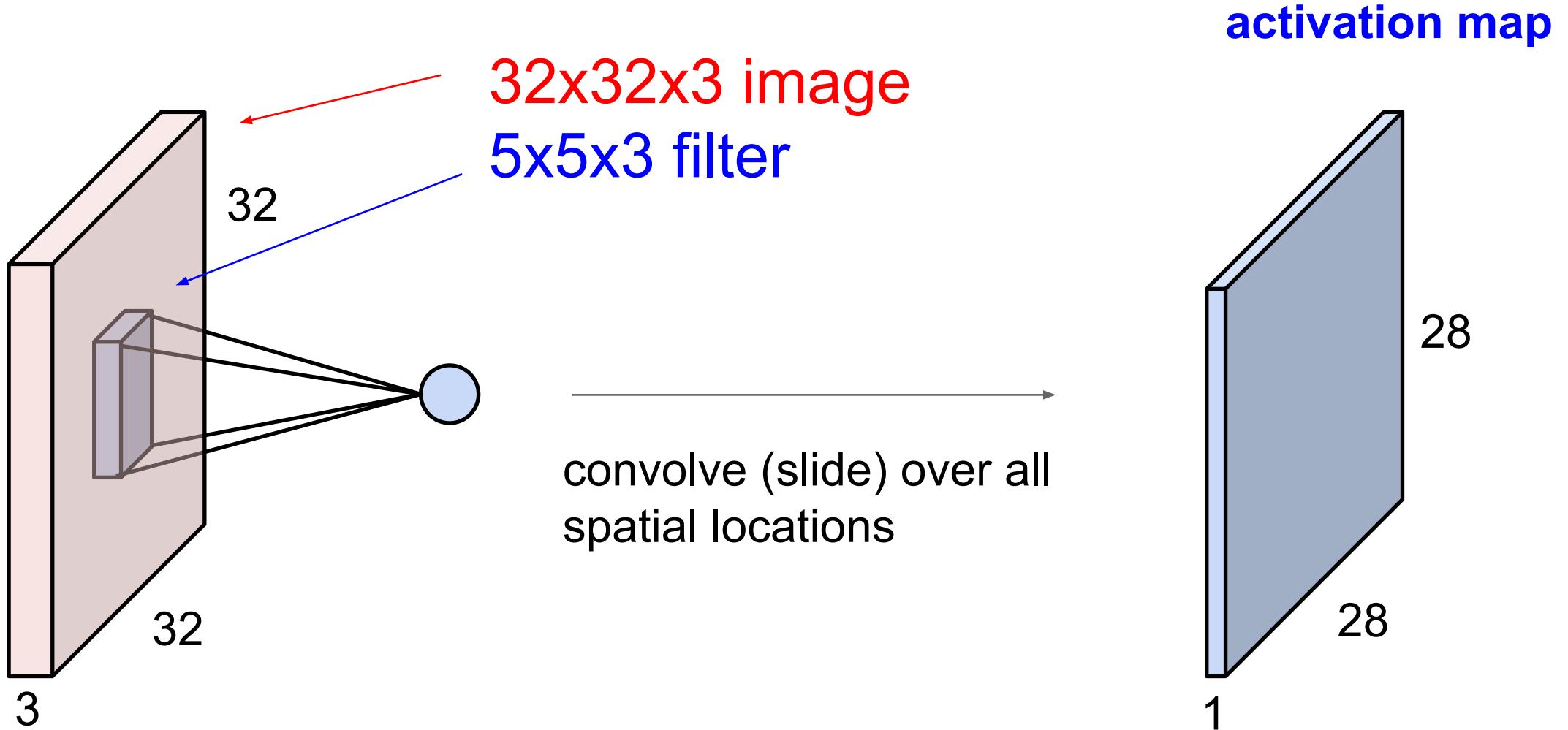
5x5x3 filter

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

Convolution Layer



Convolution Layer

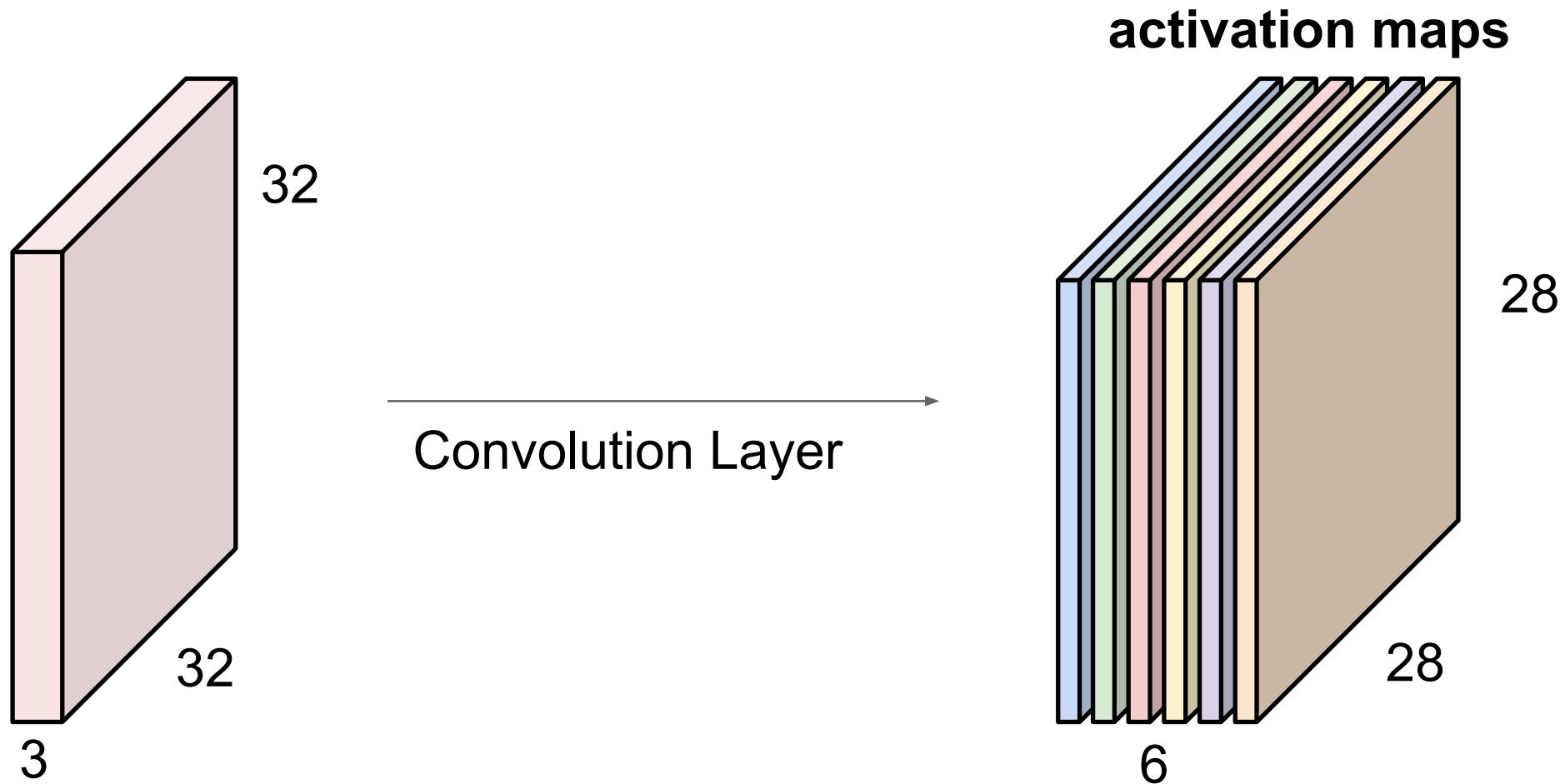


Convolution Layer

consider a second, green filter

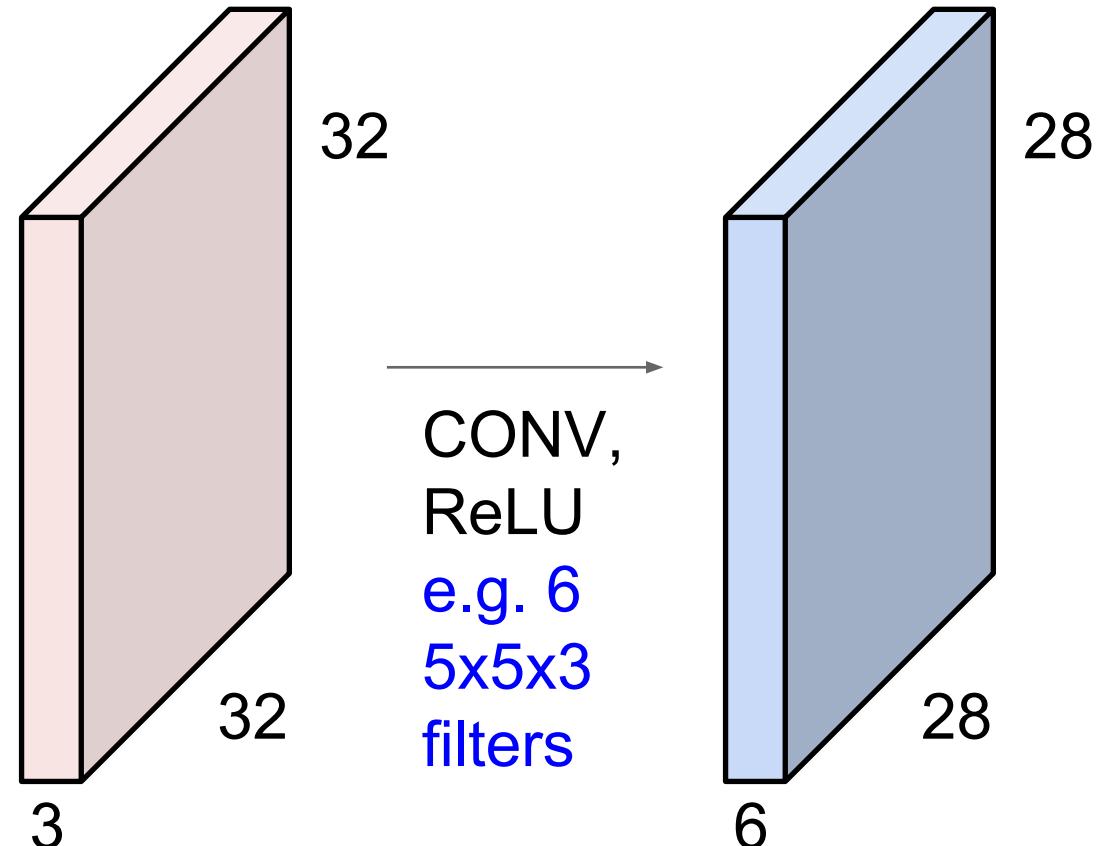


For example, if we had 6 5x5 filters, we'll get 6 separate activation maps:

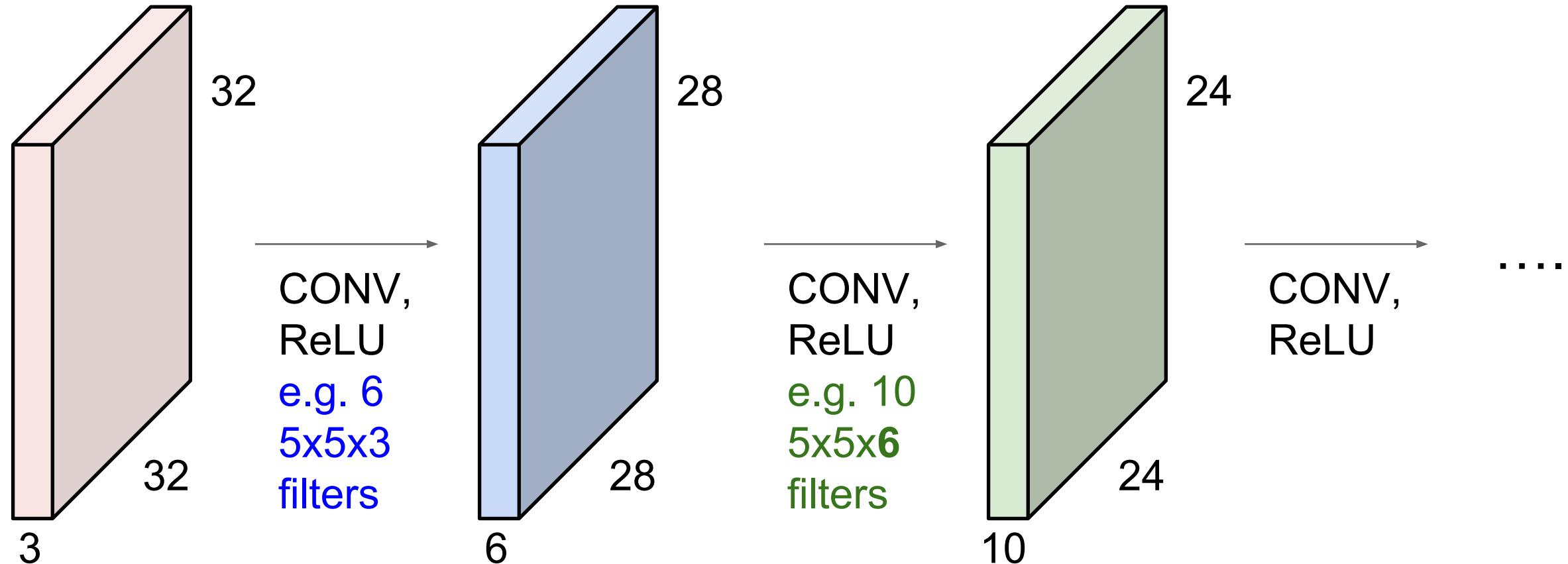


We stack these up to get a “new image” of size $28 \times 28 \times 6$!

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

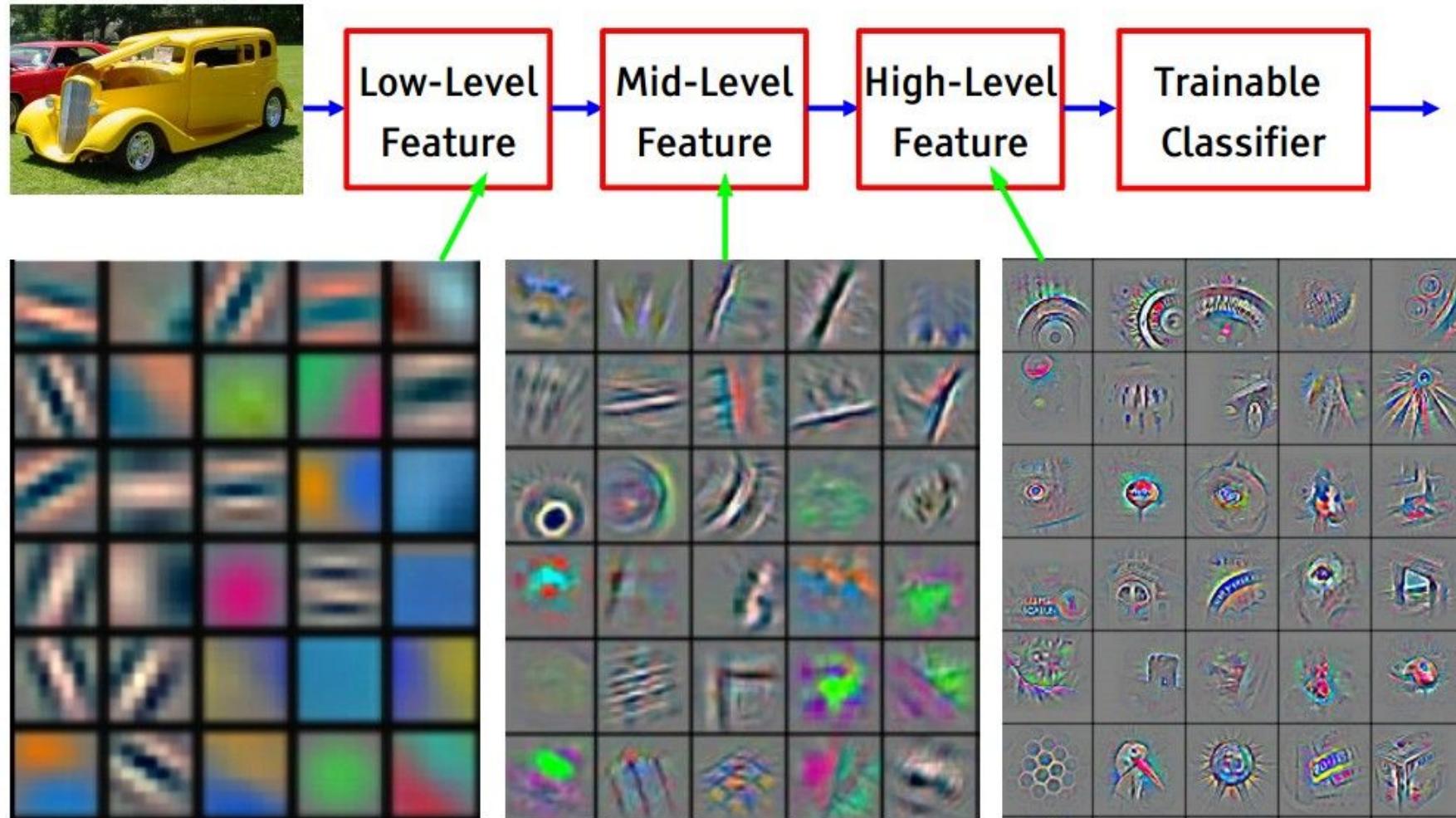


Preview: ConvNet is a sequence of Convolutional Layers, interspersed with activation functions



Preview

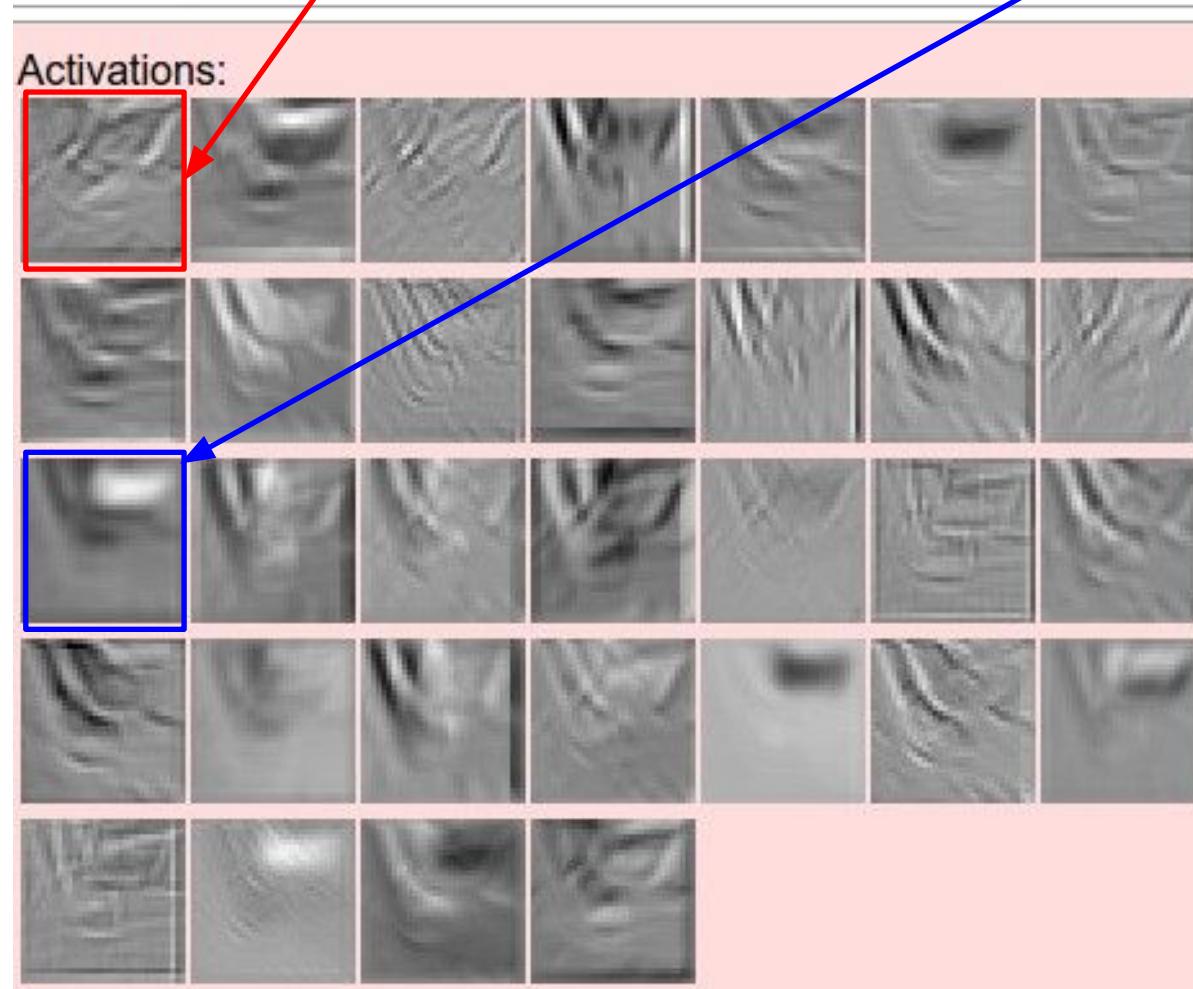
[From recent Yann LeCun slides]



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



one filter =>
one activation map



example 5x5 filters
(32 total)

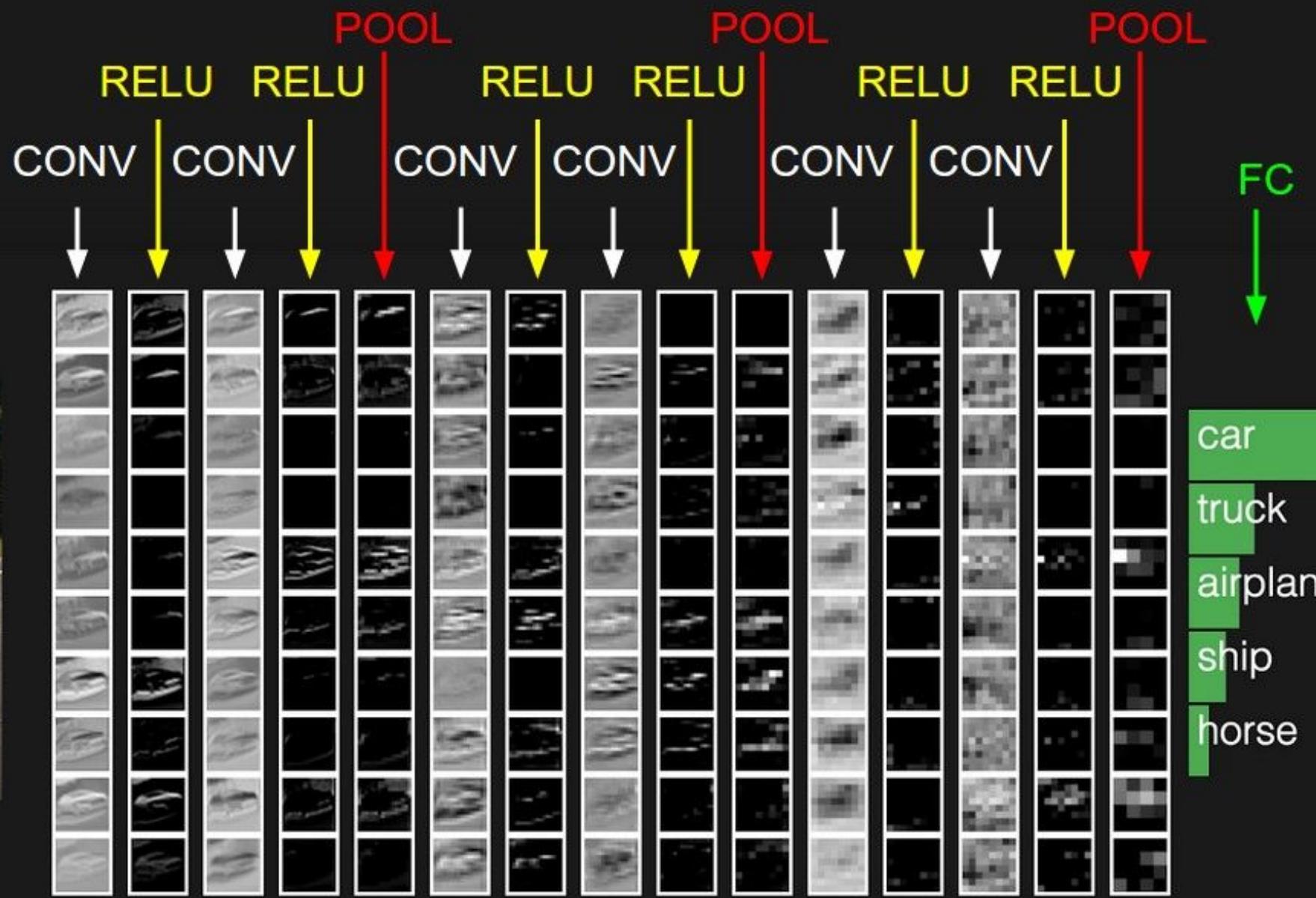
We call the layer convolutional
because it is related to convolution
of two signals:

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

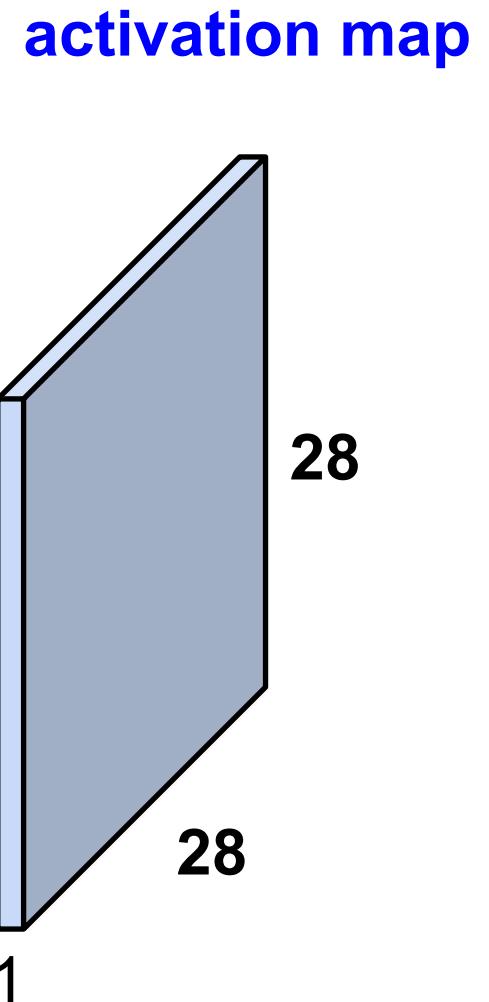
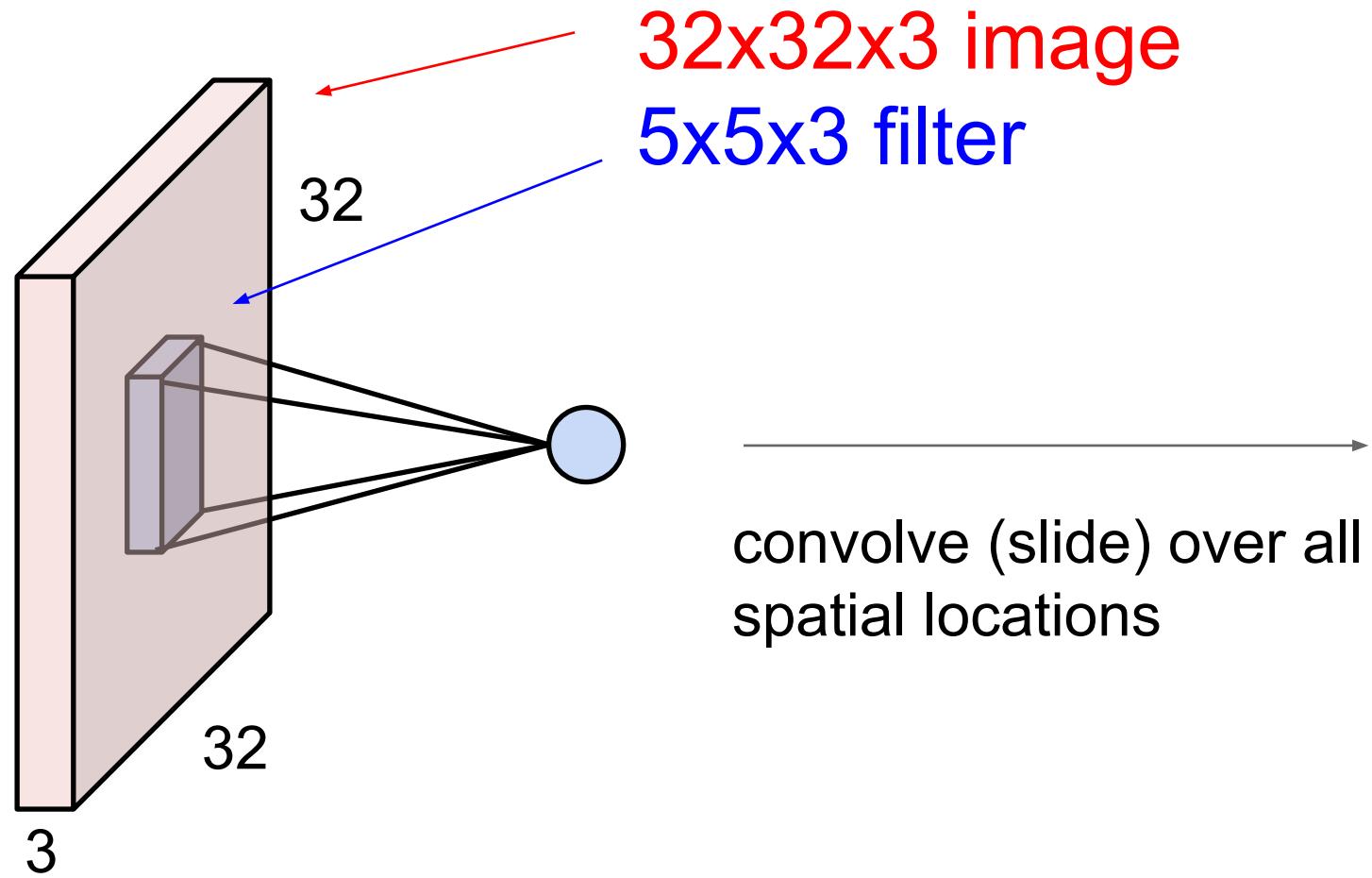


elementwise multiplication and sum of
a filter and the signal (image)

preview:

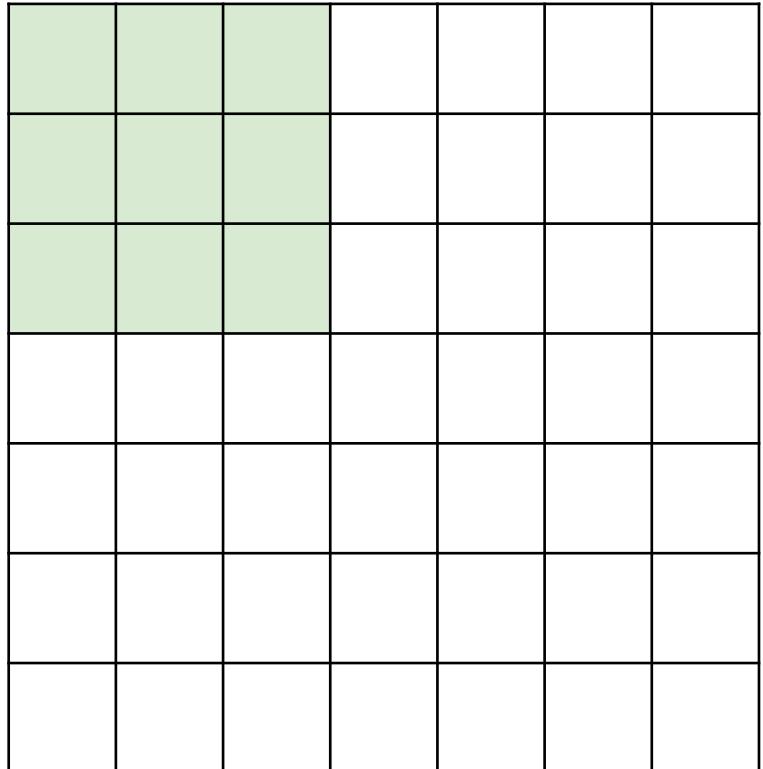


A closer look at spatial dimensions:



A closer look at spatial dimensions:

7



7x7 input (spatially)
assume 3x3 filter

7

A closer look at spatial dimensions:

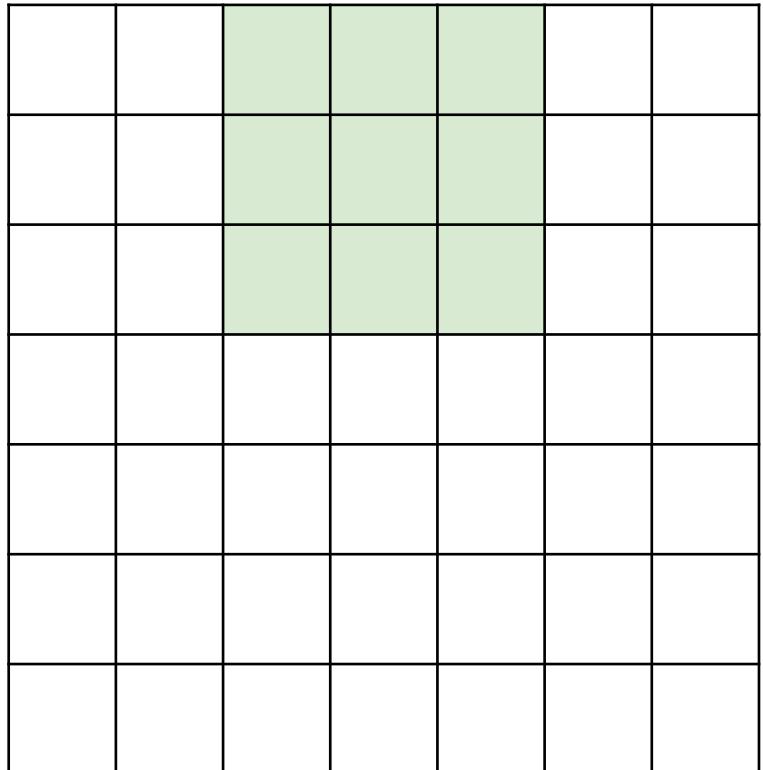
7

7x7 input (spatially)
assume 3x3 filter

7

A closer look at spatial dimensions:

7

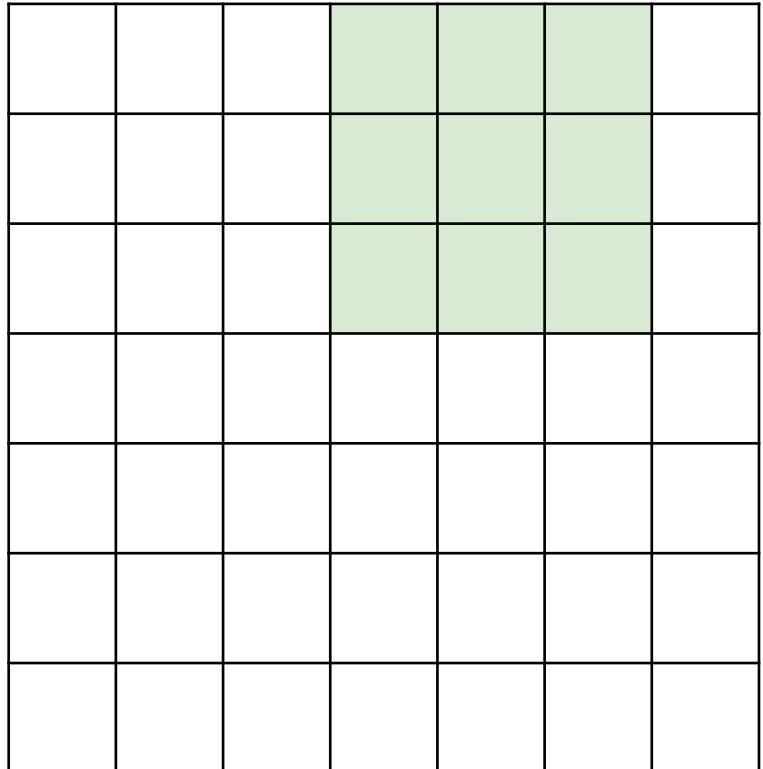


7x7 input (spatially)
assume 3x3 filter

7

A closer look at spatial dimensions:

7

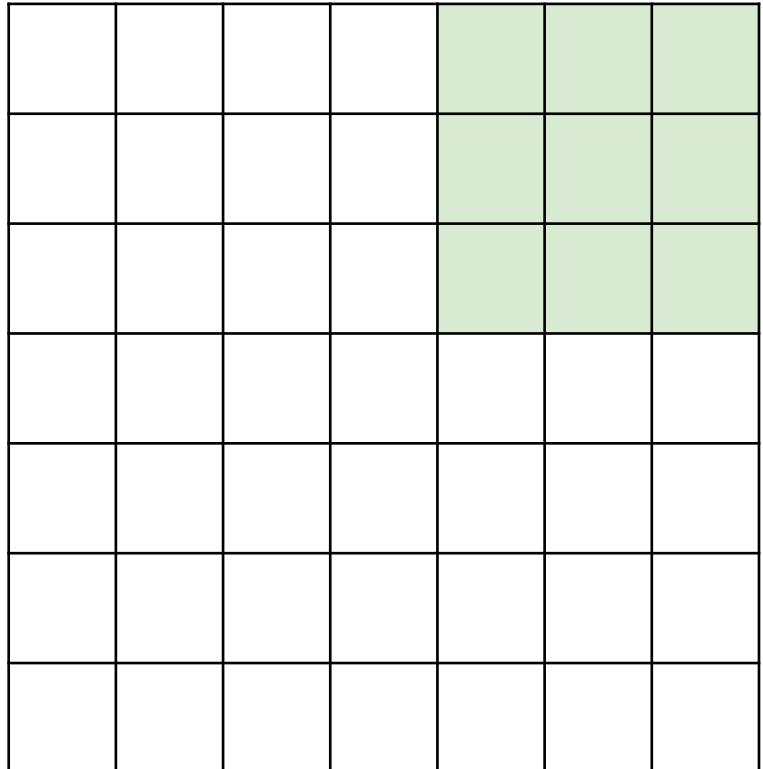


7x7 input (spatially)
assume 3x3 filter

7

A closer look at spatial dimensions:

7

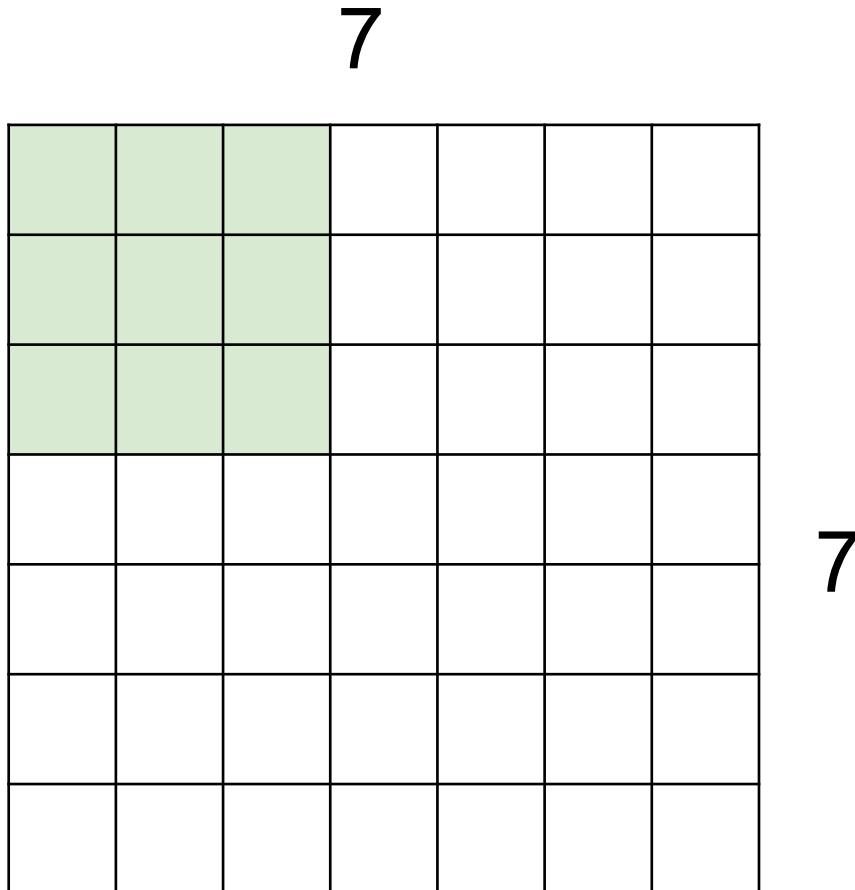


7x7 input (spatially)
assume 3x3 filter

=> 5x5 output

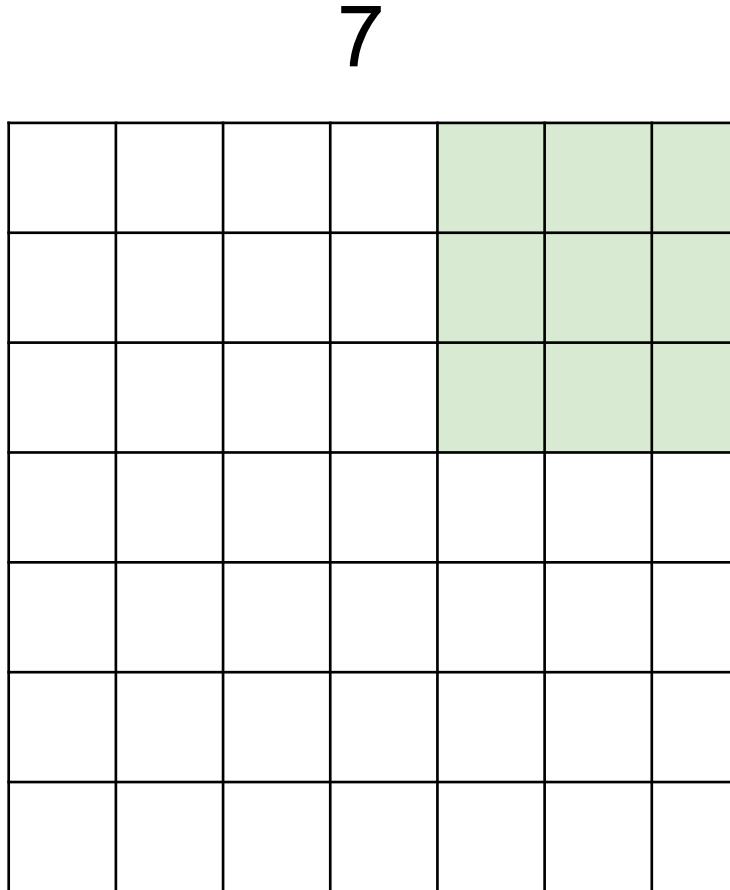
7

A closer look at spatial dimensions:



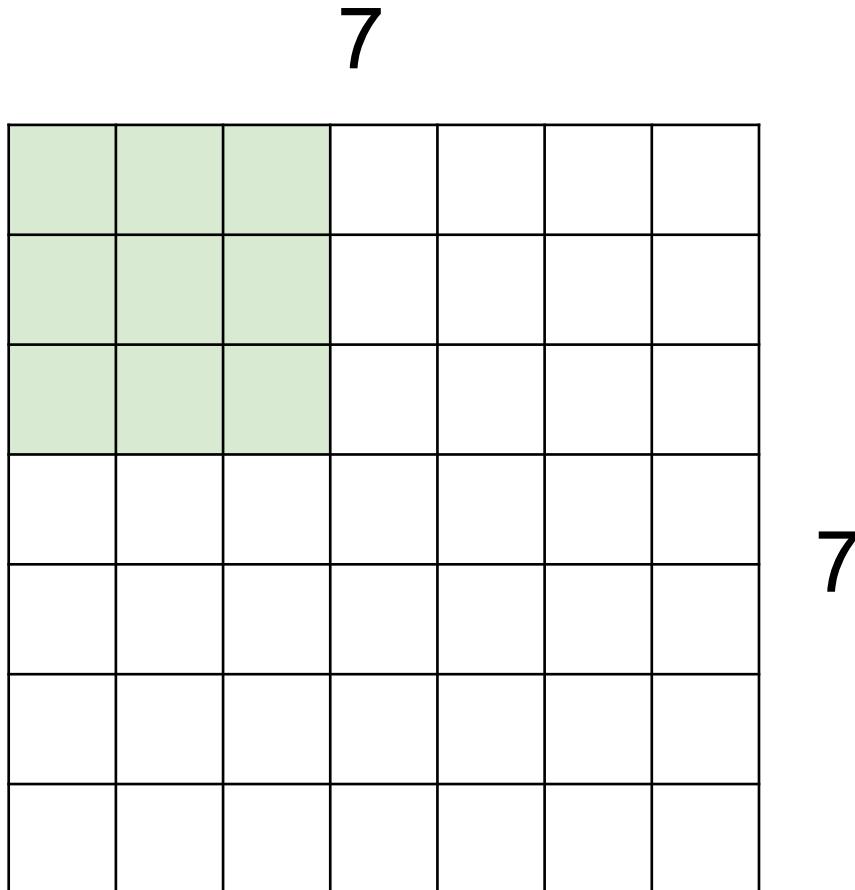
7x7 input (spatially)
assume 3x3 filter
applied **with stride 2**

A closer look at spatial dimensions:



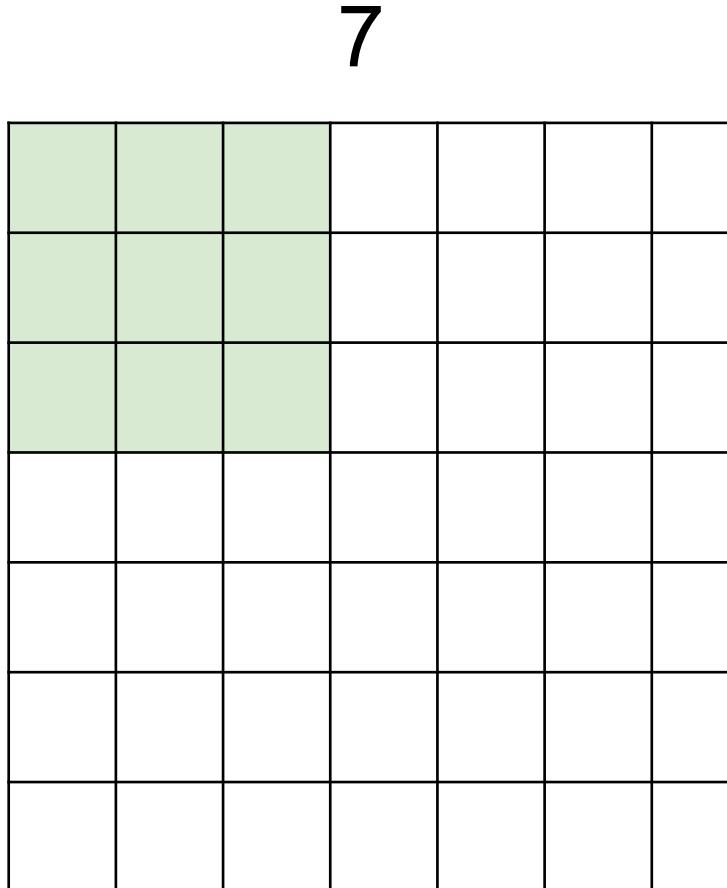
7x7 input (spatially)
assume 3x3 filter
applied **with stride 2**
=> 3x3 output!

A closer look at spatial dimensions:



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

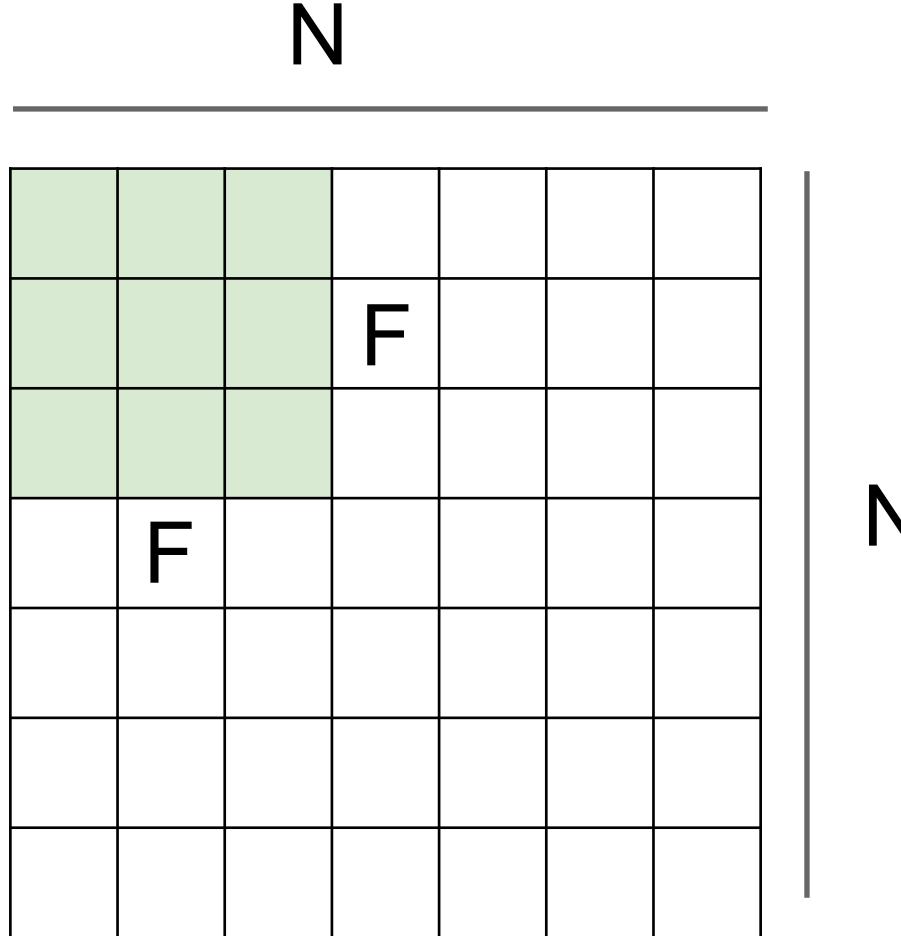
A closer look at spatial dimensions:



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

7

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



Output size:
 $(N - F) / \text{stride} + 1$

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

In practice: Common to zero pad the border

0	0	0	0	0	0			
0								
0								
0								
0								

e.g. input 7x7

3x3 filter, applied with stride 1

pad with 1 pixel border => what is the output?

(recall:)

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

In practice: Common to zero pad the border

0	0	0	0	0	0			
0								
0								
0								
0								

e.g. input 7x7

3x3 filter, applied with stride 1

pad with 1 pixel border => what is the output?

7x7 output!

In practice: Common to zero pad the border

0	0	0	0	0	0			
0								
0								
0								
0								

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)

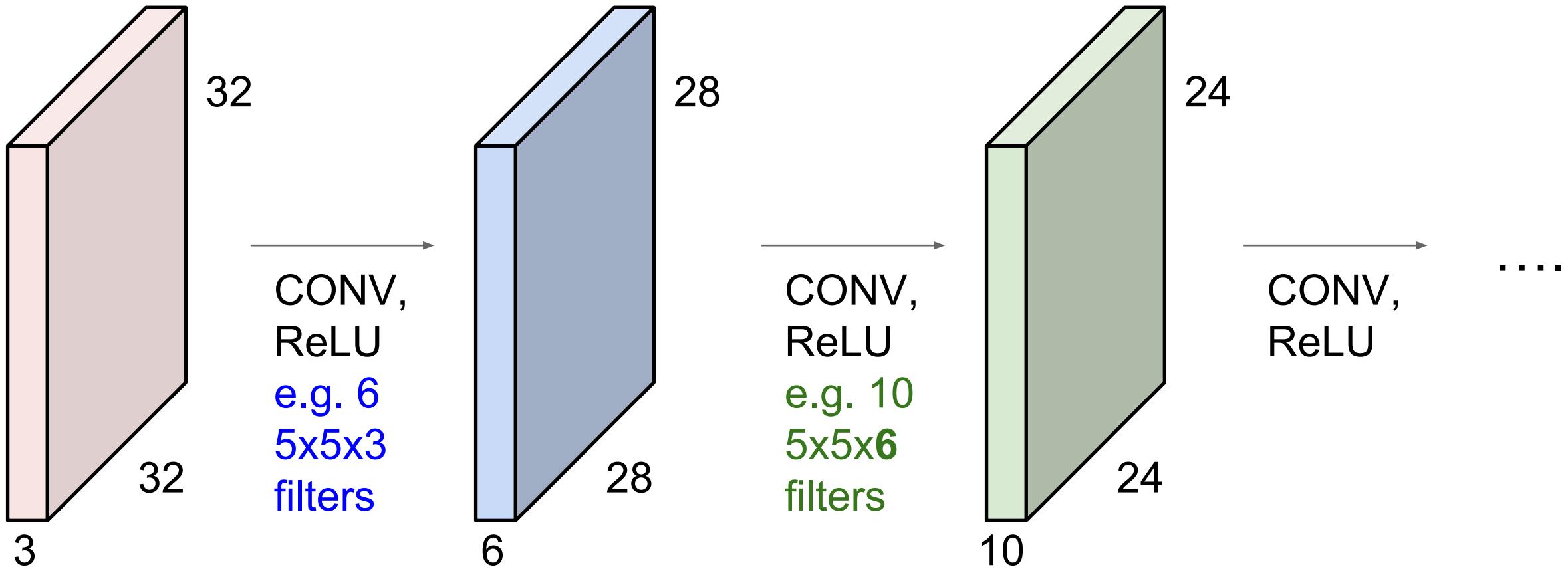
e.g. $F = 3 \Rightarrow$ zero pad with 1

$F = 5 \Rightarrow$ zero pad with 2

$F = 7 \Rightarrow$ zero pad with 3

Remember back to...

E.g. 32x32 input convolved repeatedly with 5x5 filters shrinks volumes spatially!
(32 -> 28 -> 24 ...). Shrinking too fast is not good, doesn't work well.

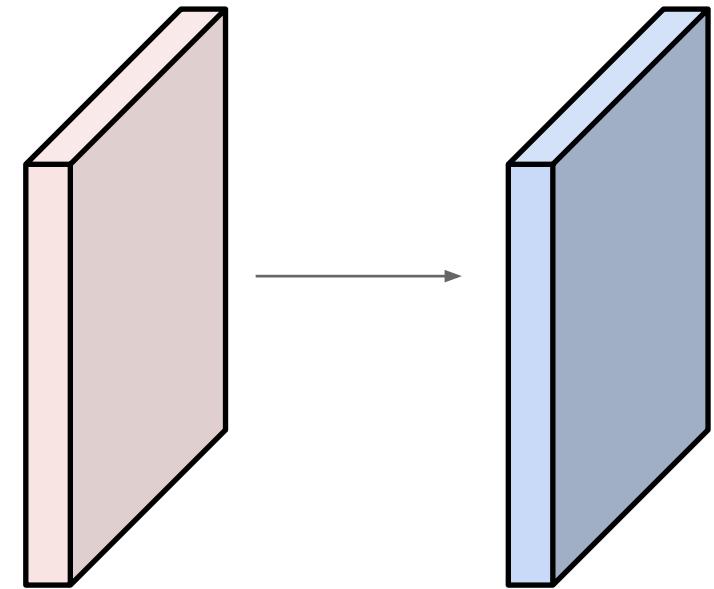


Examples time:

Input volume: **32x32x3**

10 5x5 filters with stride 1, pad 2

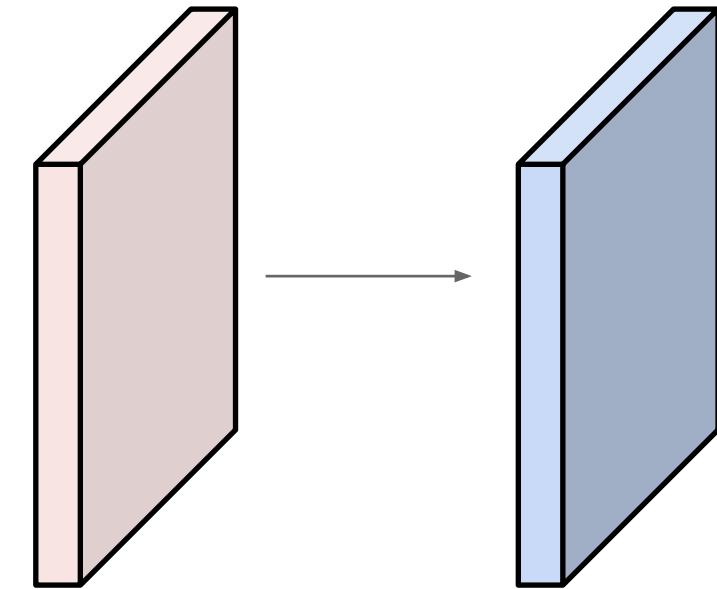
Output volume size: ?



Examples time:

Input volume: **32x32x3**

10 **5x5** filters with stride 1, pad 2



Output volume size:

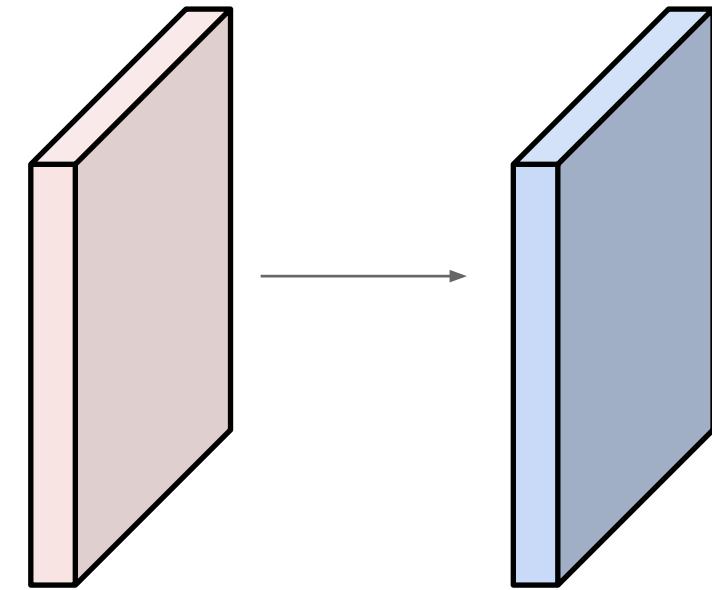
$(32+2*2-5)/1+1 = 32$ spatially, so

32x32x10

Examples time:

Input volume: **32x32x3**

10 5x5 filters with stride 1, pad 2

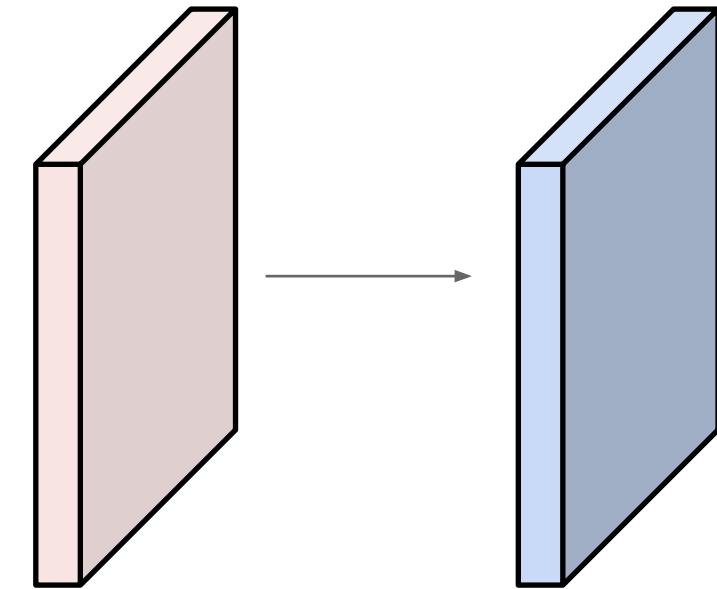


Number of parameters in this layer?

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)

$$\Rightarrow 76 * 10 = 760$$

Summary. To summarize, the Conv Layer:

- Accepts a volume of size $W_1 \times H_1 \times D_1$
- Requires four hyperparameters:
 - Number of filters K ,
 - their spatial extent F ,
 - the stride S ,
 - the amount of zero padding P .
- Produces a volume of size $W_2 \times H_2 \times D_2$ where:
 - $W_2 = (W_1 - F + 2P)/S + 1$
 - $H_2 = (H_1 - F + 2P)/S + 1$ (i.e. width and height are computed equally by symmetry)
 - $D_2 = K$
- With parameter sharing, it introduces $F \cdot F \cdot D_1$ weights per filter, for a total of $(F \cdot F \cdot D_1) \cdot K$ weights and K biases.
- In the output volume, the d -th depth slice (of size $W_2 \times H_2$) is the result of performing a valid convolution of the d -th filter over the input volume with a stride of S , and then offset by d -th bias.

Common settings:

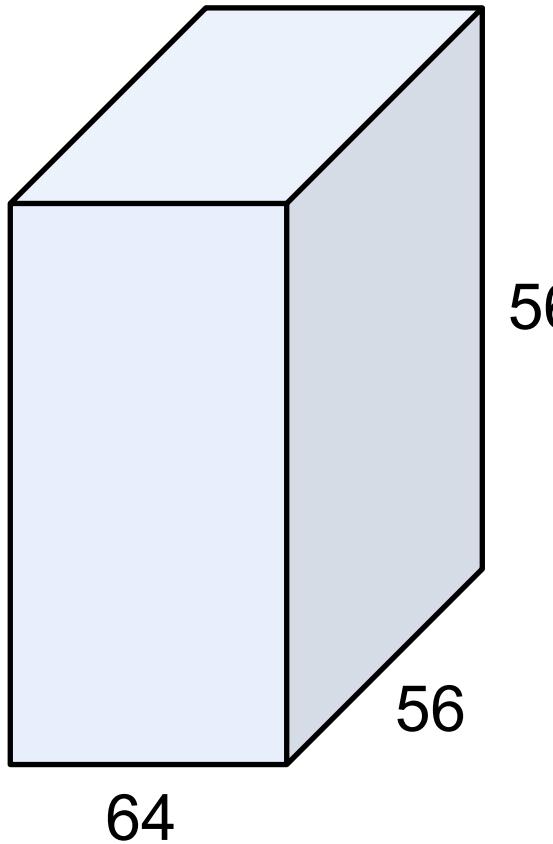
Summary. To summarize, the Conv Layer:

- 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.

$K = (\text{powers of 2, e.g. } 32, 64, 128, 512)$

- $F = 3, S = 1, P = 1$
- $F = 5, S = 1, P = 2$
- $F = 5, S = 2, P = ? \text{ (whatever fits)}$
- $F = 1, S = 1, P = 0$

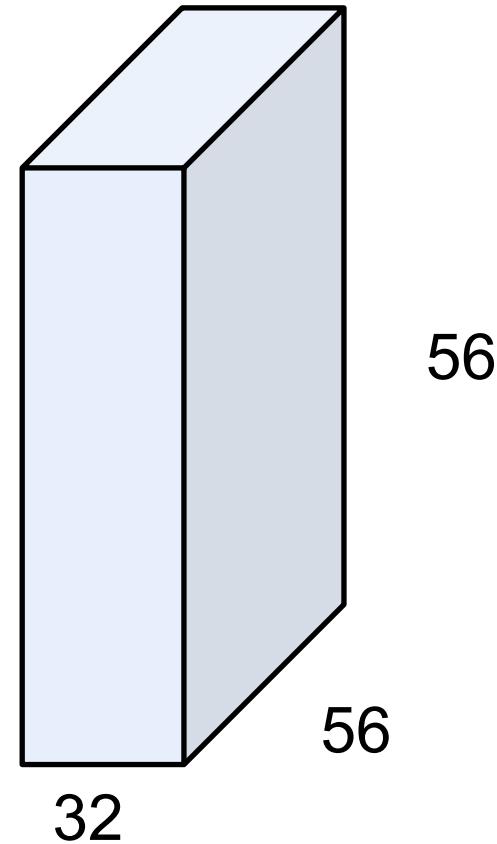
(btw, 1x1 convolution layers make perfect sense)



1x1 CONV
with 32 filters

→

(each filter has size
1x1x64, and performs a
64-dimensional dot
product)

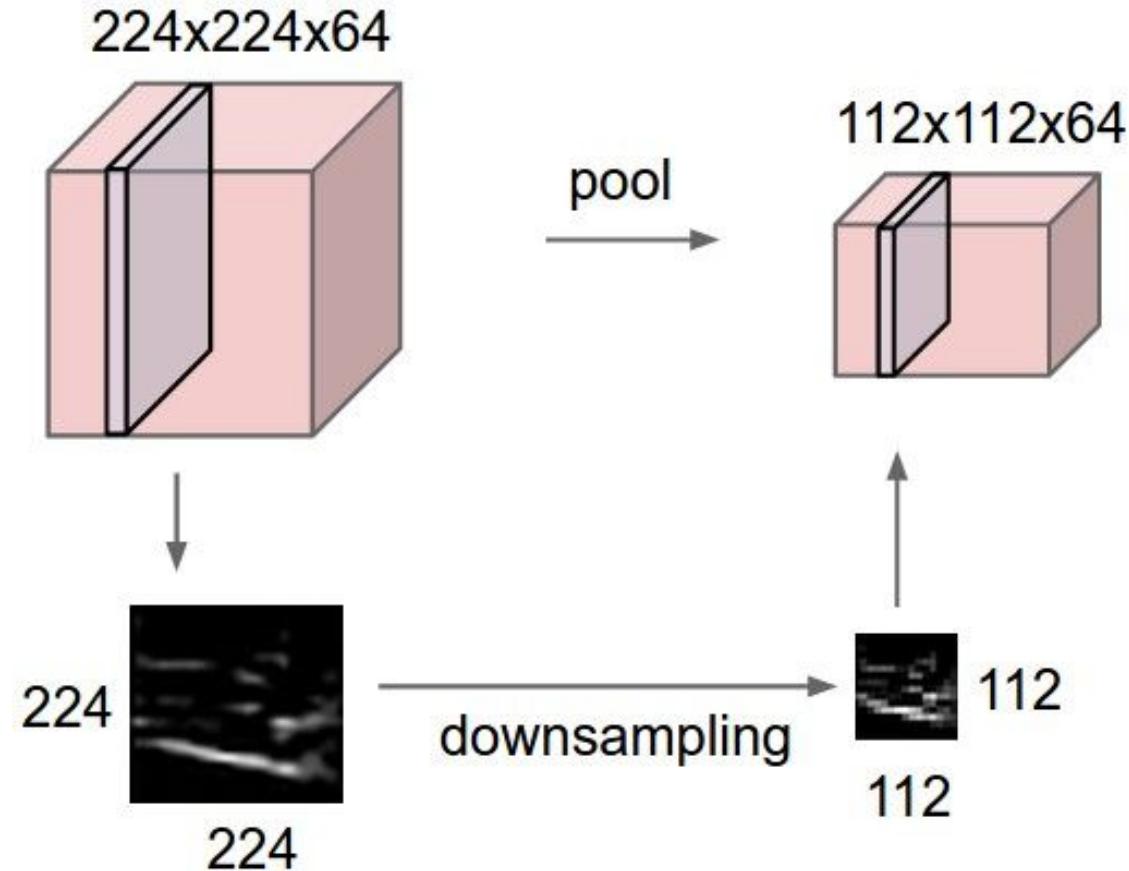


two more layers to go: POOL/FC



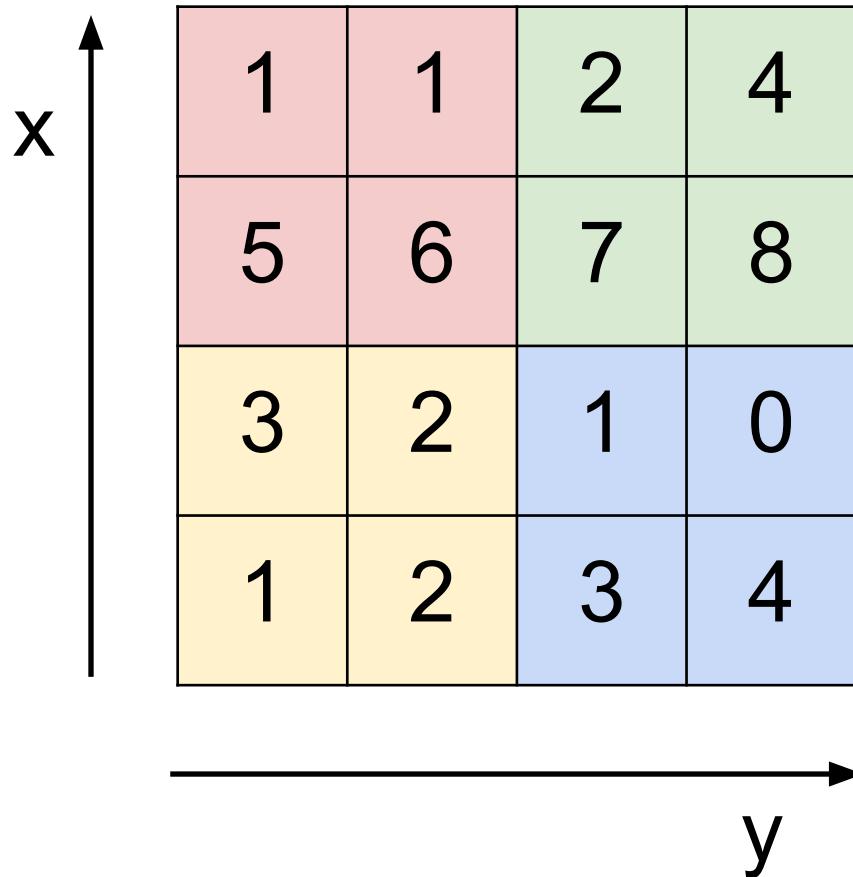
Pooling layer

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



MAX POOLING

Single depth slice



max pool with 2x2 filters
and stride 2

A 2x2 grid representing the output of the max pooling operation. It contains the maximum values from the 2x2 receptive fields of each output unit:

6	8
3	4

- 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:

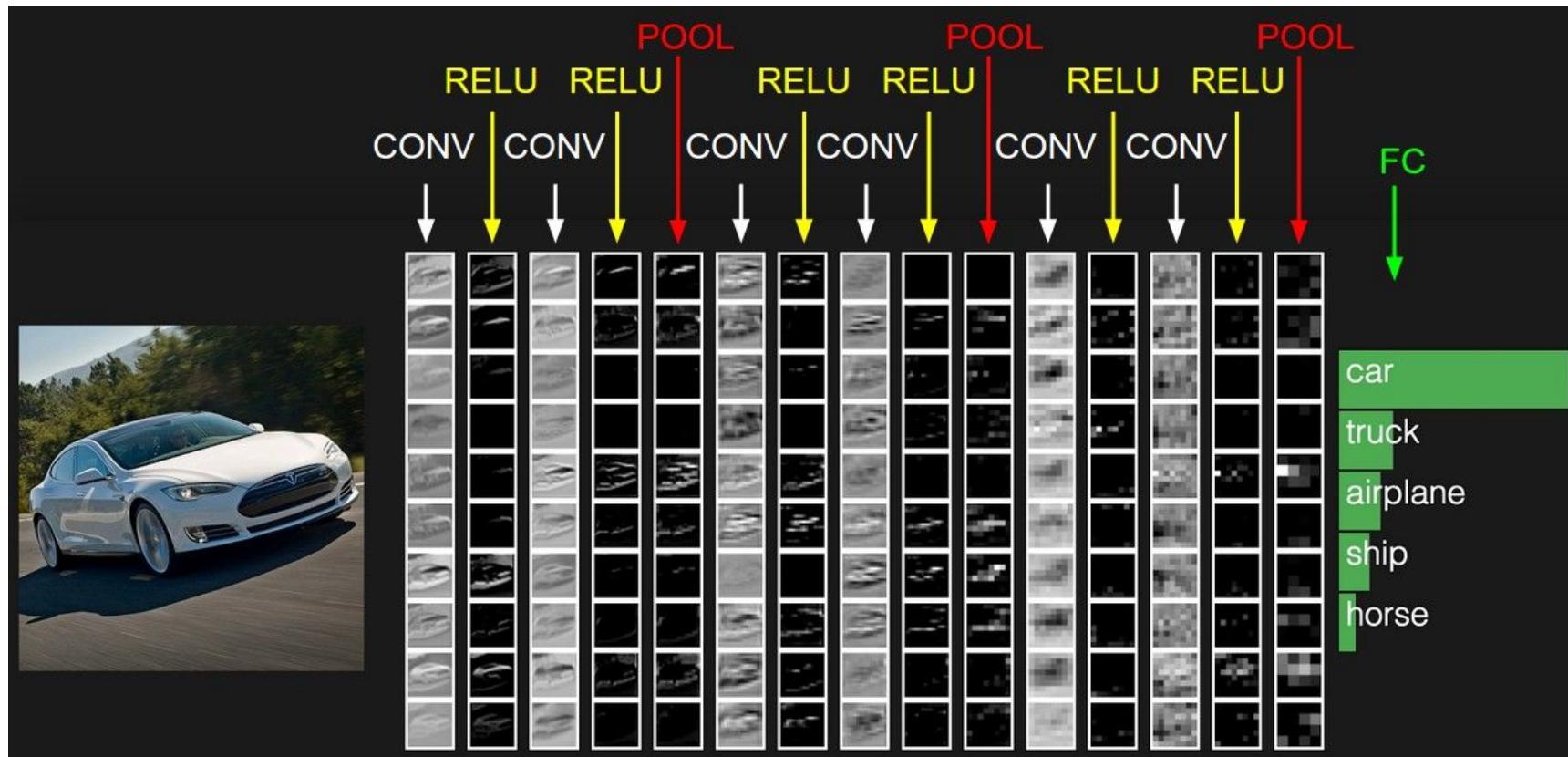
- Accepts a volume of size $W_1 \times H_1 \times D_1$
- Requires three hyperparameters:
 - their spatial extent F ,
 - the stride S ,
- Produces a volume of size $W_2 \times H_2 \times D_2$ where:
 - $W_2 = (W_1 - F)/S + 1$
 - $H_2 = (H_1 - F)/S + 1$
 - $D_2 = D_1$
- Introduces zero parameters since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

$F = 2, S = 2$

$F = 3, S = 2$

Fully Connected Layer (FC layer)

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



Example: CONV layer in Torch

Summary. To summarize, the Conv Layer:

- Accepts a volume of size $W_1 \times H_1 \times D_1$
- Requires four hyperparameters:
 - Number of filters K ,
 - their spatial extent F ,
 - the stride S ,
 - the amount of zero padding P .

SpatialConvolution

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

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

The parameters are the following:

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

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

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

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

Summary of ConvNets

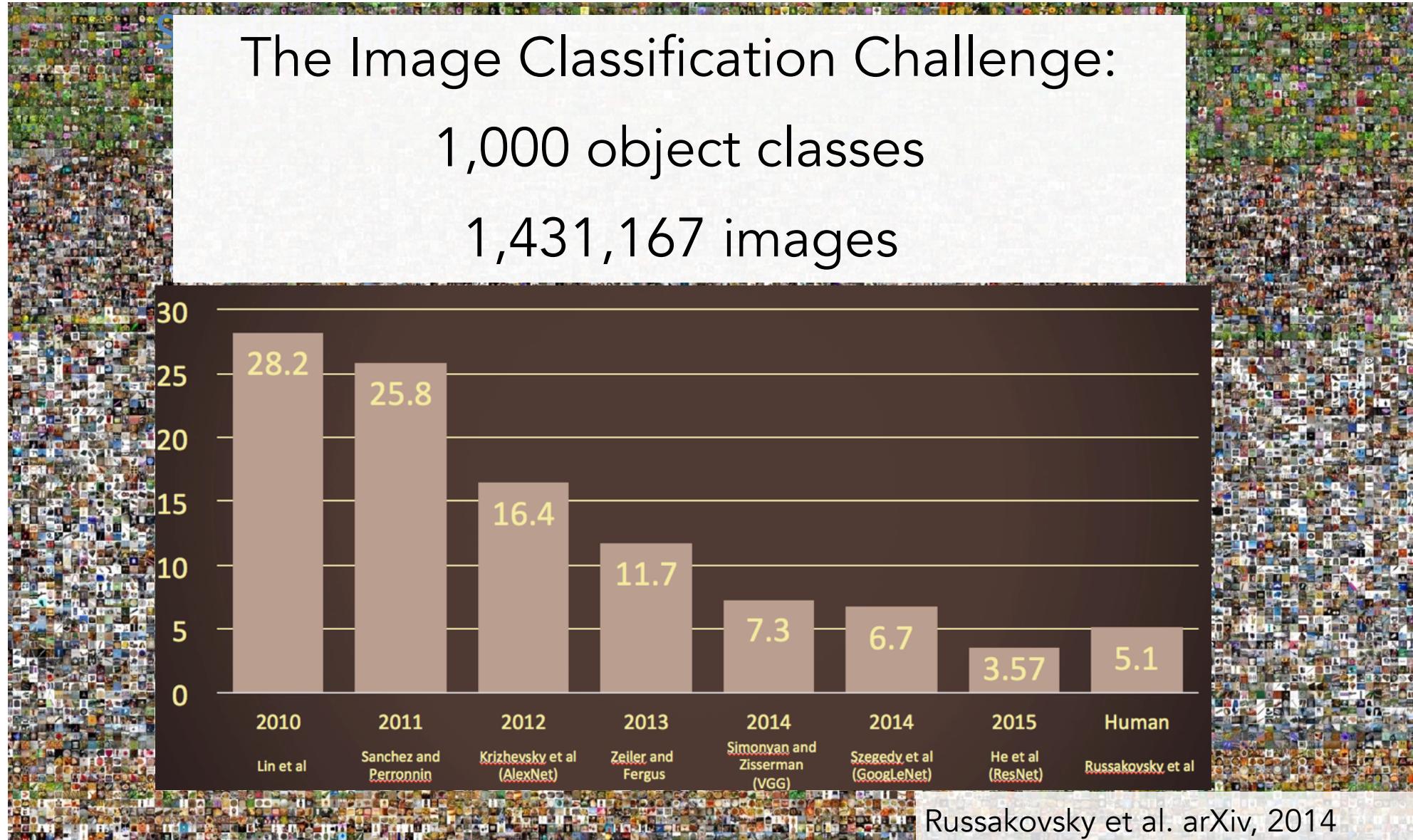
- ConvNets stack CONV,POOL,FC layers
- Trend towards smaller filters and deeper architectures
- Trend towards getting rid of POOL/FC layers (just CONV)
- Typical architectures look like

$[(\text{CONV-RELU})^*N - \text{POOL?}]^*M - (\text{FC-RELU})^*K, \text{SOFTMAX}$

where N is usually up to ~5, M is large, $0 \leq K \leq 2$.

- but recent advances such as ResNet/GoogLeNet challenge this paradigm

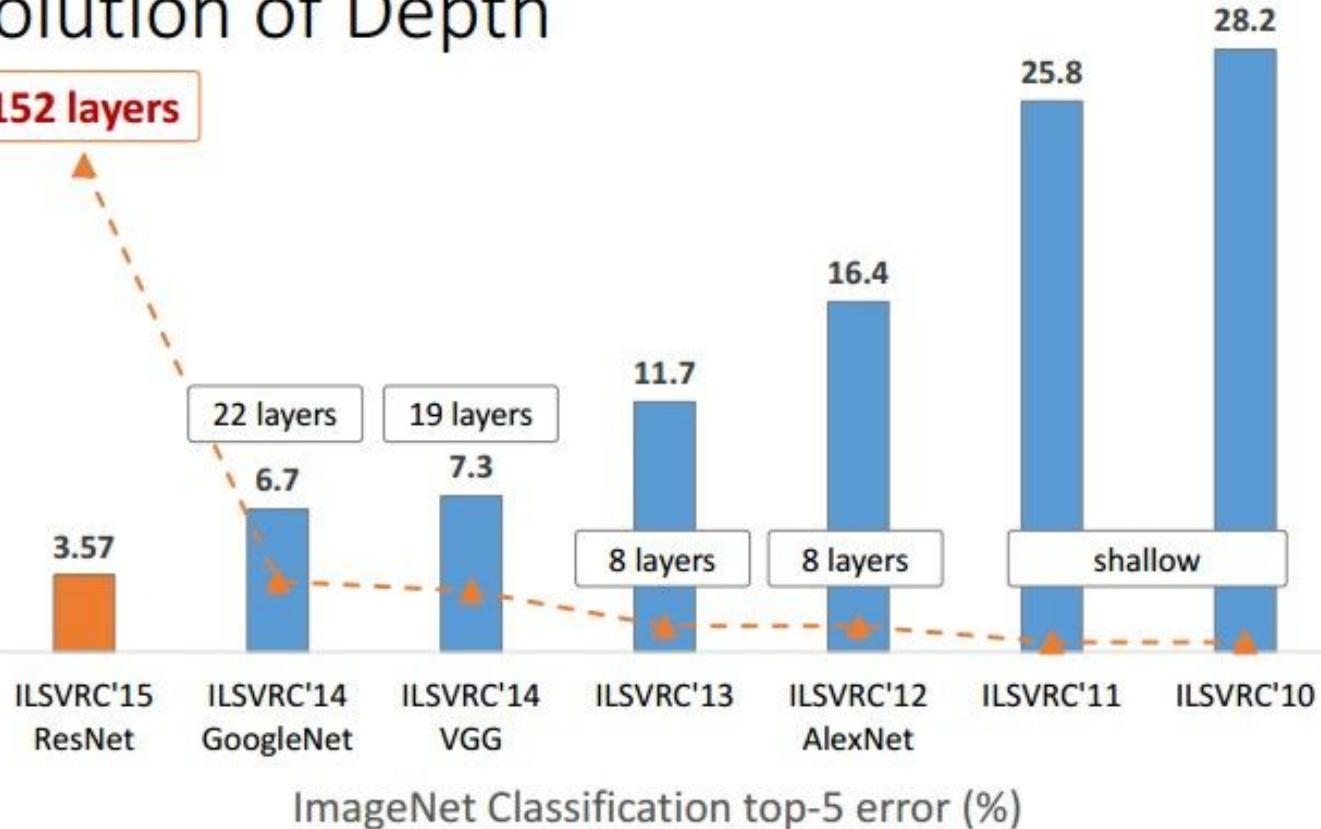
ImageNet Large Scale Visual Recognition Challenge



Deep Learning

Microsoft
Research

Revolution of Depth



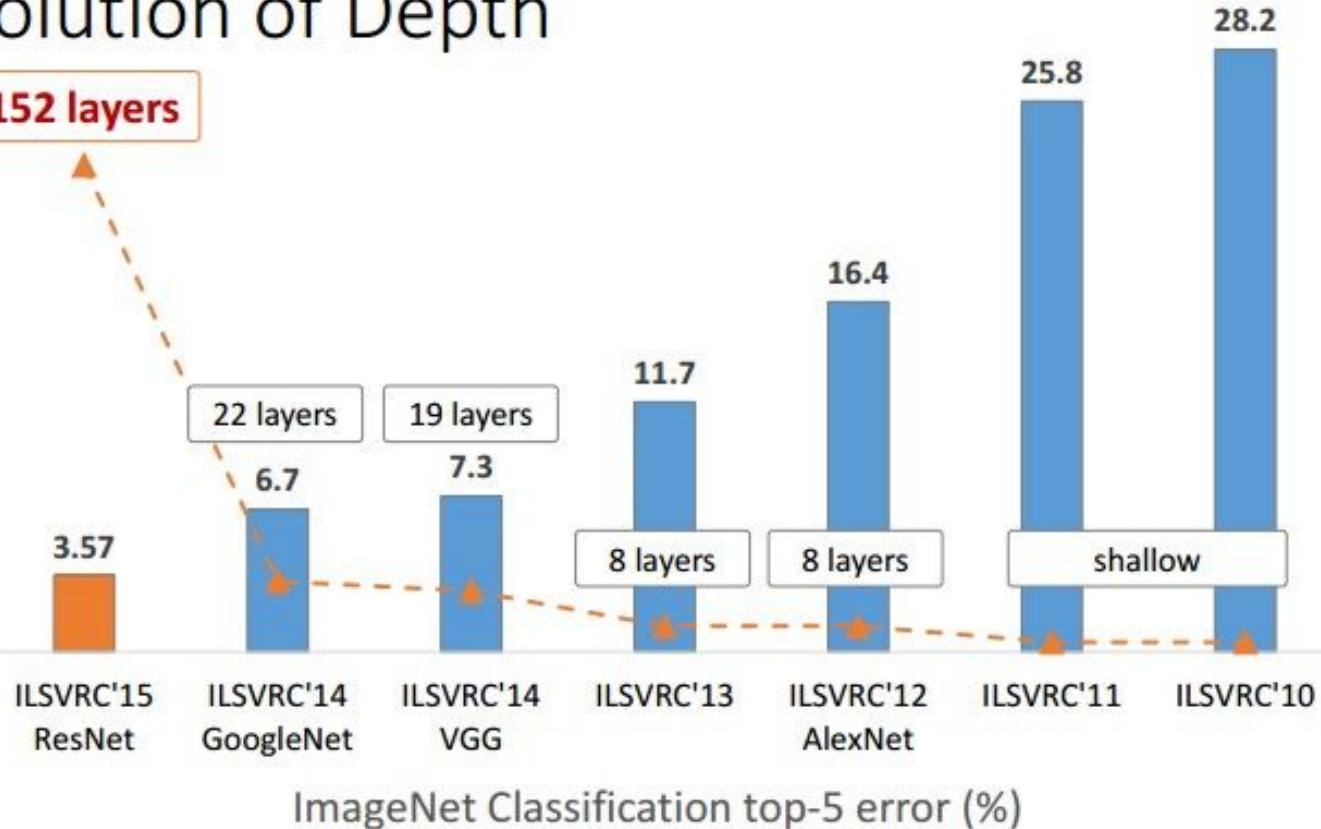
Kaiming He, Xiangyu Zhang, Shaoqing Ren, & Jian Sun. "Deep Residual Learning for Image Recognition". arXiv 2015.

(slide from Kaiming He's recent presentation)

Deep Learning

Microsoft
Research

Revolution of Depth



Kaiming He, Xiangyu Zhang, Shaoqing Ren, & Jian Sun. "Deep Residual Learning for Image Recognition". arXiv 2015.

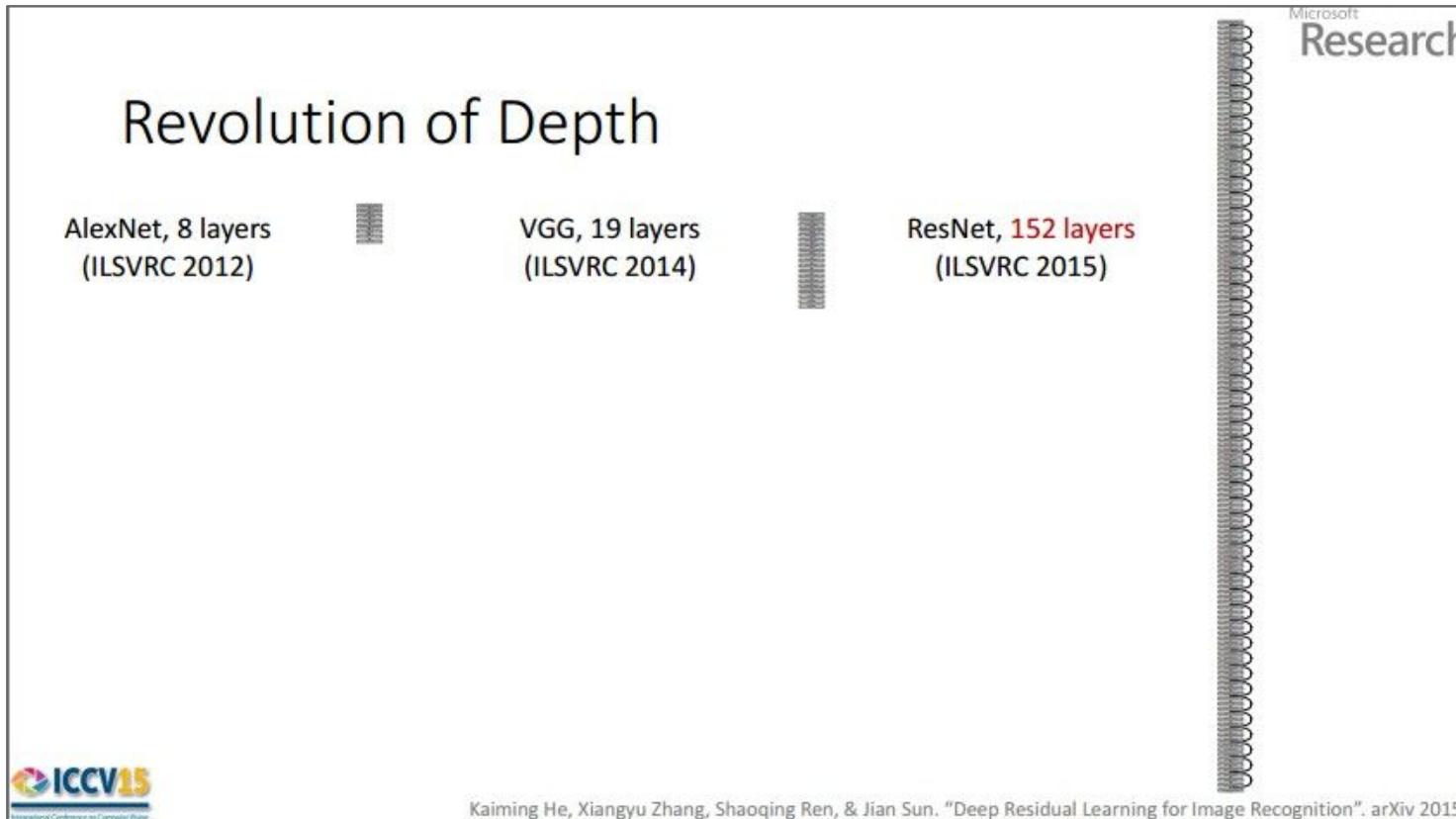
(slide from Kaiming He's recent presentation)

Deep Learning

Case Study: ResNet

[He et al., 2015]

ILSVRC 2015 winner (3.6% top 5 error)



2-3 weeks of training
on 8 GPU machine

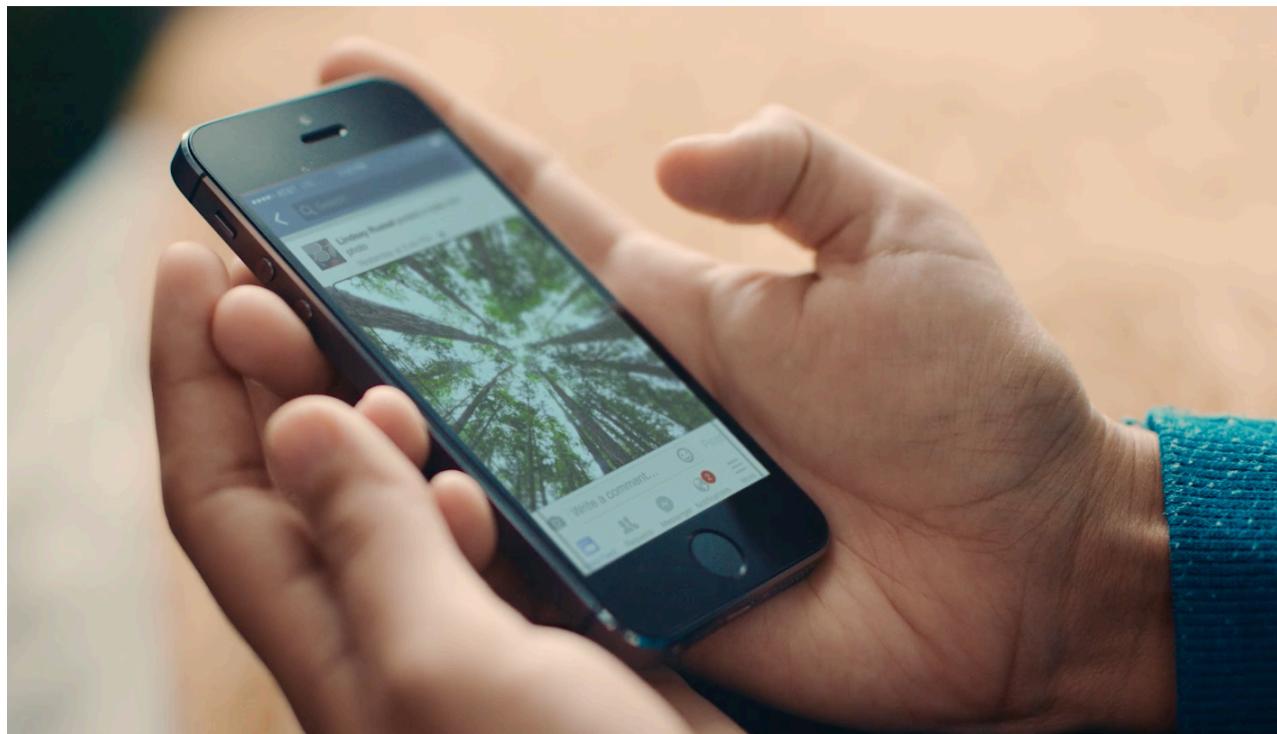
at runtime: faster
than a VGGNet!
(even though it has
8x more layers)

(slide from Kaiming He's recent presentation)

Facebook's Blind Users

April 4, 2016

Using Artificial Intelligence to Help Blind People 'See' Facebook



By [Shaomei Wu](#), Software Engineer and [Hermes Pique](#), Software Engineer on iOS and [Jeffrey Wieland](#), Head of Accessibility

<https://www.youtube.com/watch?v=5fZLch5DjZc>

<https://newsroom.fb.com/news/2016/04/using-artificial-intelligence-to-help-blind-people-see-facebook/>