CS 3630!

Lecture 20:

Deep Learning

Many slides adapted from Stanford's CS231N by Fei-Fei Li, Justin Johnson, Serena Yeung, as well as Slides by Marc'Aurelio Ranzato (NYU), Dhruv Batra & Devi Parikh (Georgia Tech)

Topics

1. **Supervised Learning**
2. **Convolutional Neural Networks**
3. **Learning CNN Parameters**
4. **Applications in Perception**

Motivation

- Robotics:
  - Perception, thinking, acting
- Deep learning has revolutionized perception
- Getting increasingly important in thinking/acting
- This lecture:
  - High-level intro to CNNs and learning for perception
- Next lecture:
  - Applications in robotics

Image by Voyage

1. Supervised Learning

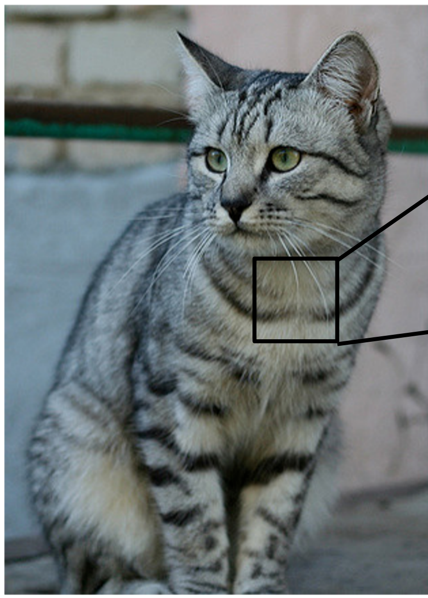
- Example: classification

This image by Nikita is licensed under

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

$\longrightarrow$ cat

The Problem: Semantic Gap




```
[[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)

An image classifier

```python
def classify_image(image):
  # Some magic here?
  return class_label
```

Unlike e.g. sorting a list of numbers,

no obvious way to hard-code the algorithm for recognizing a cat, or other classes.

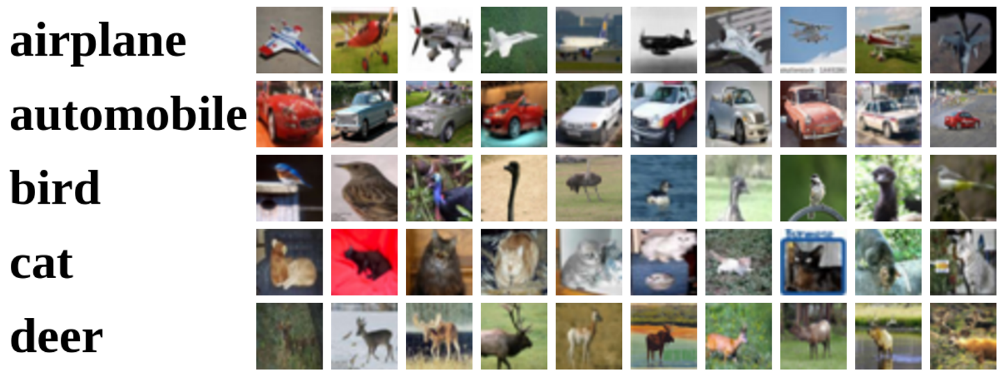
ML: A Data-Driven Approach

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

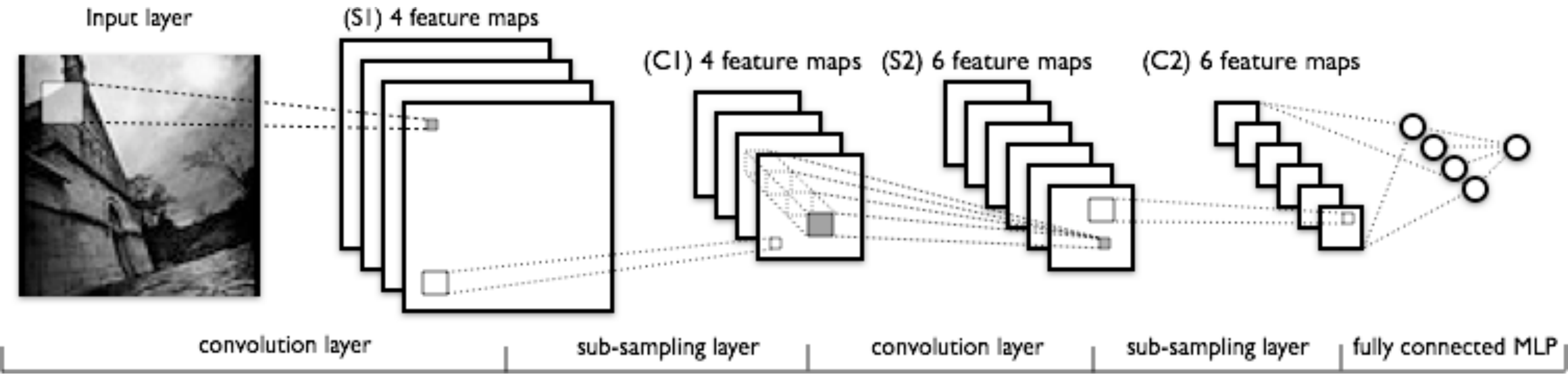
```
def train(images, labels):
  # Machine learning!
  return model
```

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

Example training set



2. Convolutional Neural Networks



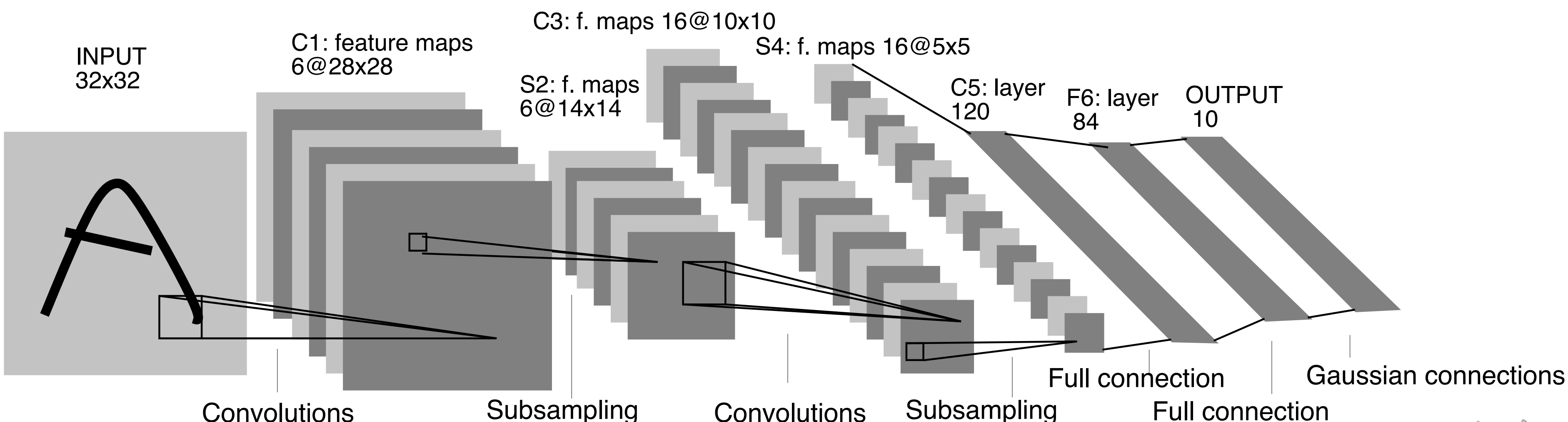
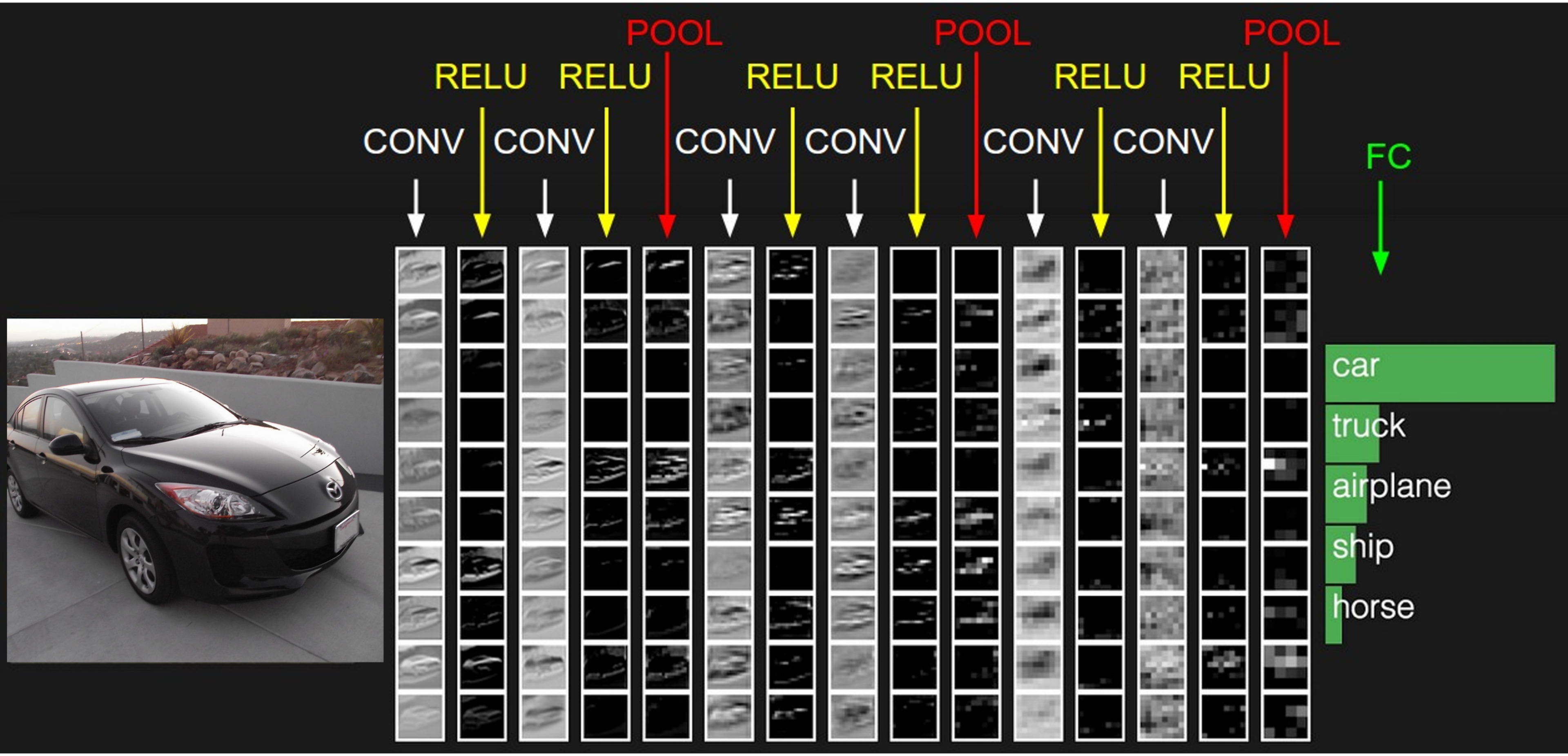


Image Credit: Yann LeCun, Kevin Murphy

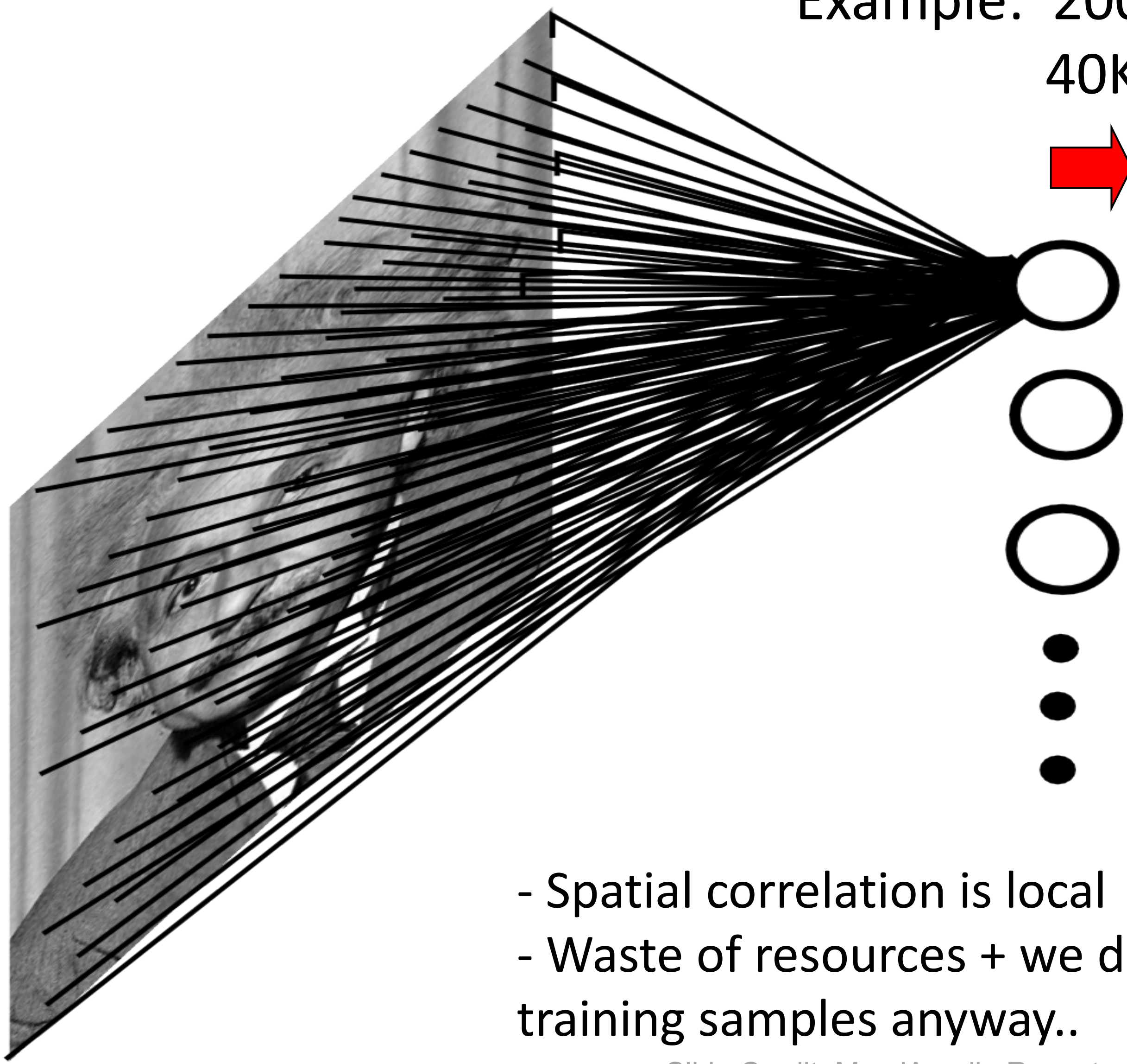
preview:



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

Fully Connected Layer

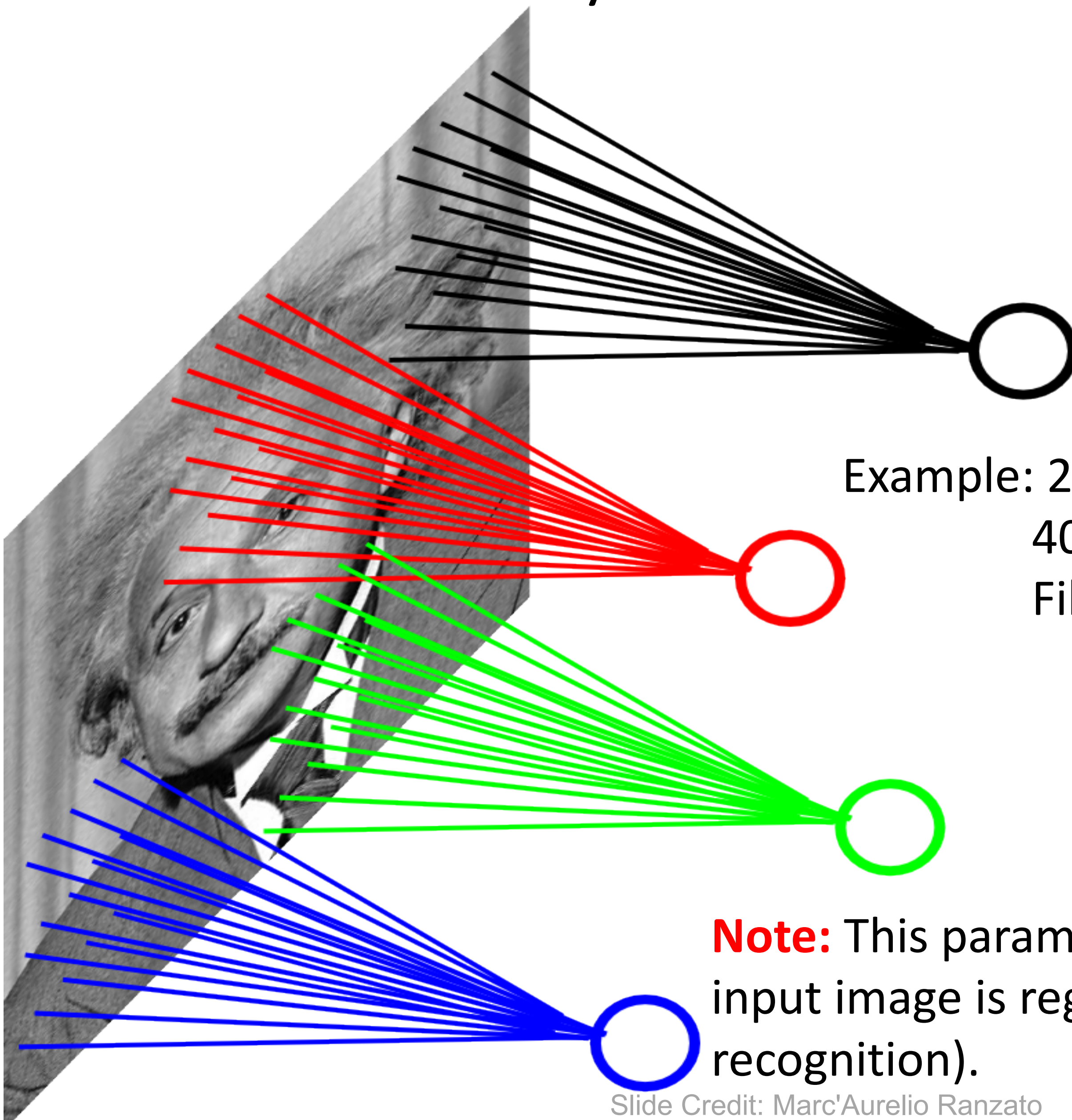
Example: 200x200 image
40K hidden units
~2.4B parameters!!!



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

Slide Credit: Marc'Aurelio Ranzato

Locally Connected Layer



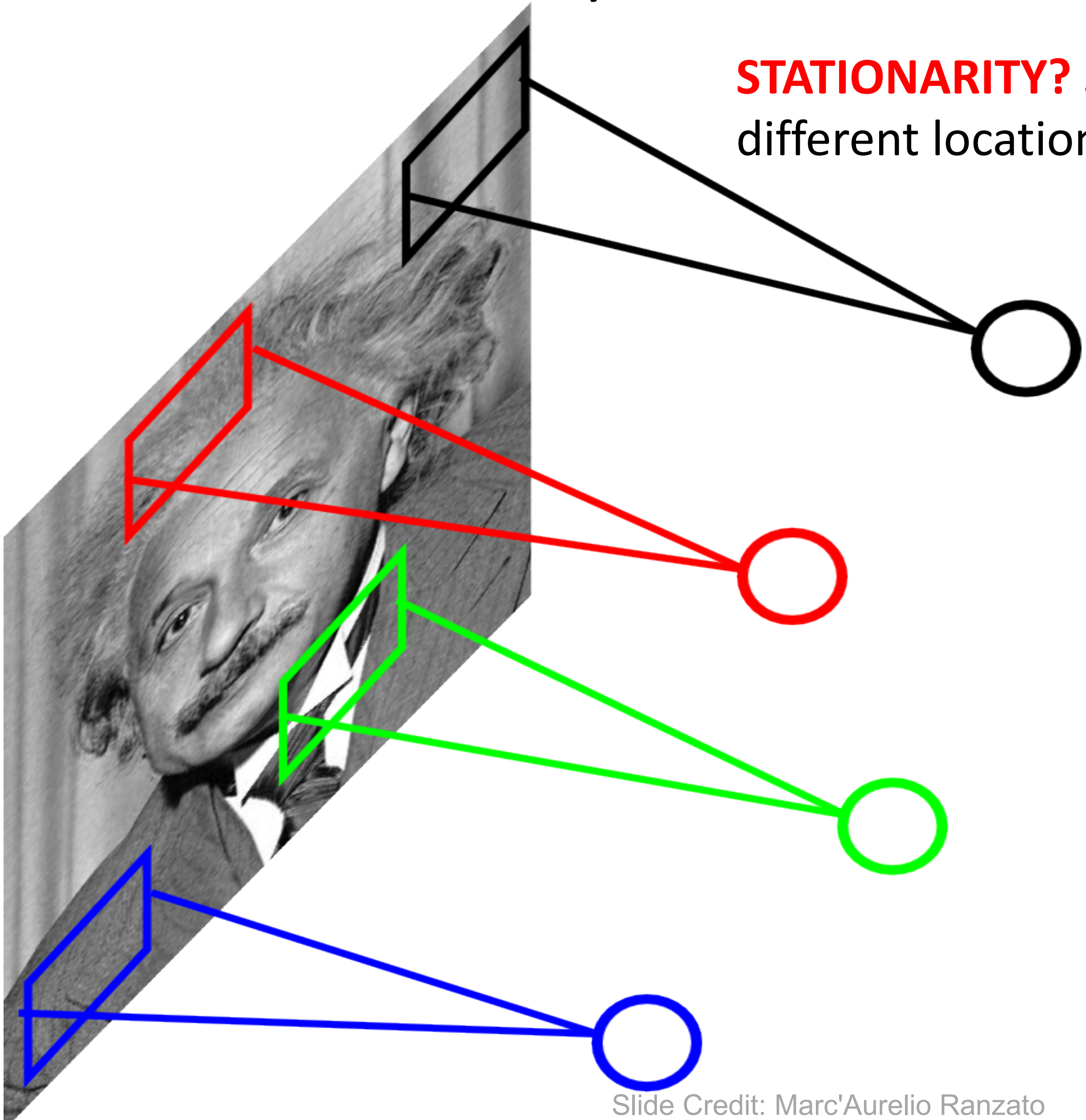
Example: 200x200 image
40K hidden units
Filter size: 10x10
4M parameters

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

Slide Credit: Marc'Aurelio Ranzato

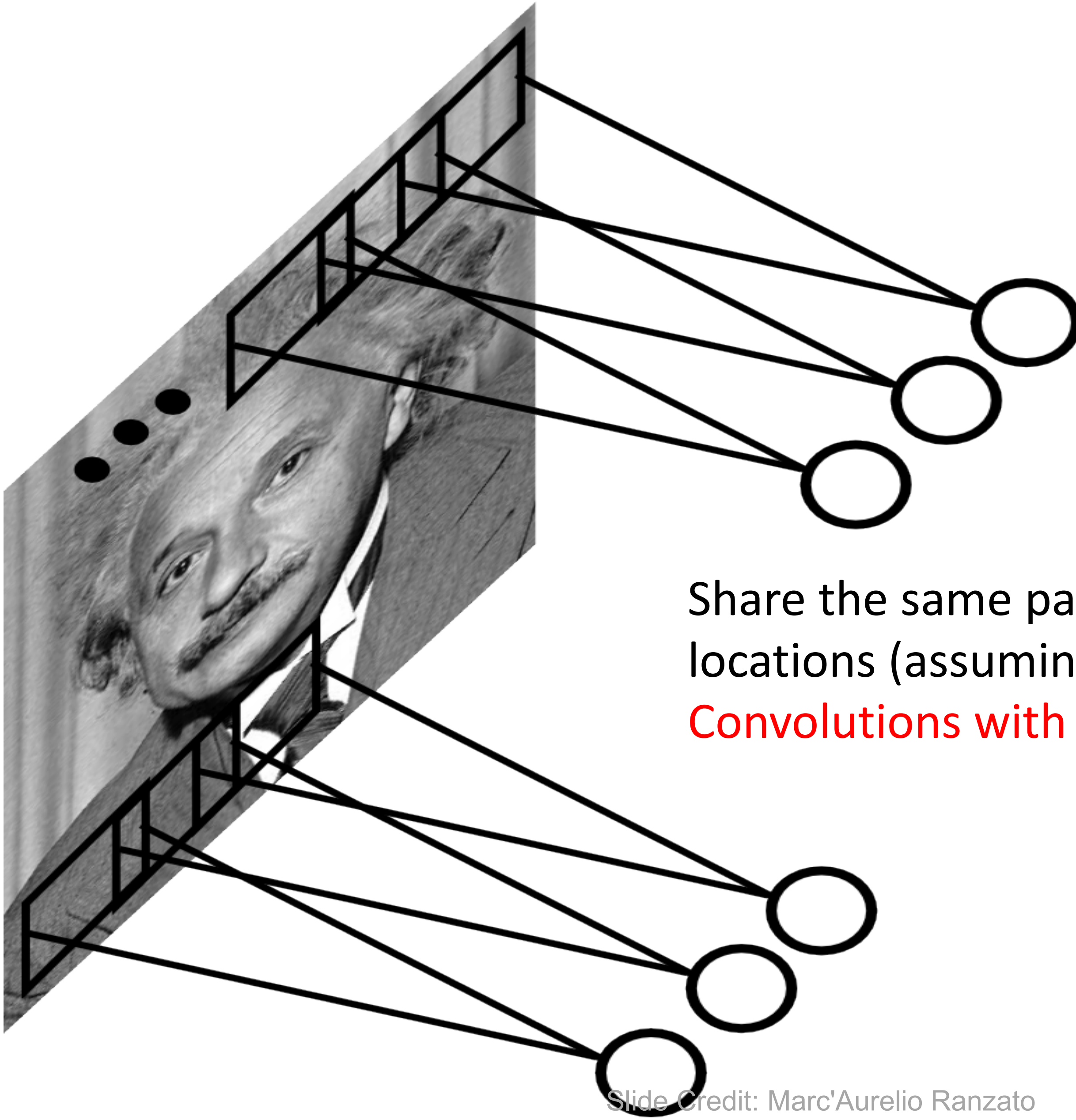
Locally Connected Layer

STATIONARITY? Statistics is similar at different locations



Slide Credit: Marc'Aurelio Ranzato

Convolutional Layer

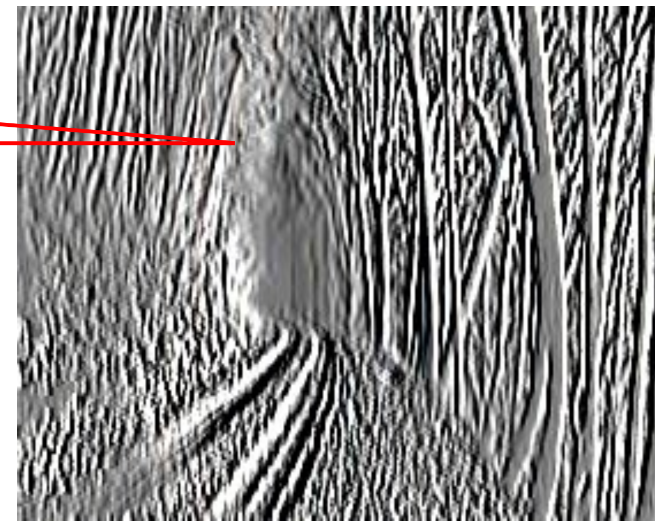


Share the same parameters across different locations (assuming input is stationary):
Convolutions with learned kernels

Slide Credit: Marc'Aurelio Ranzato

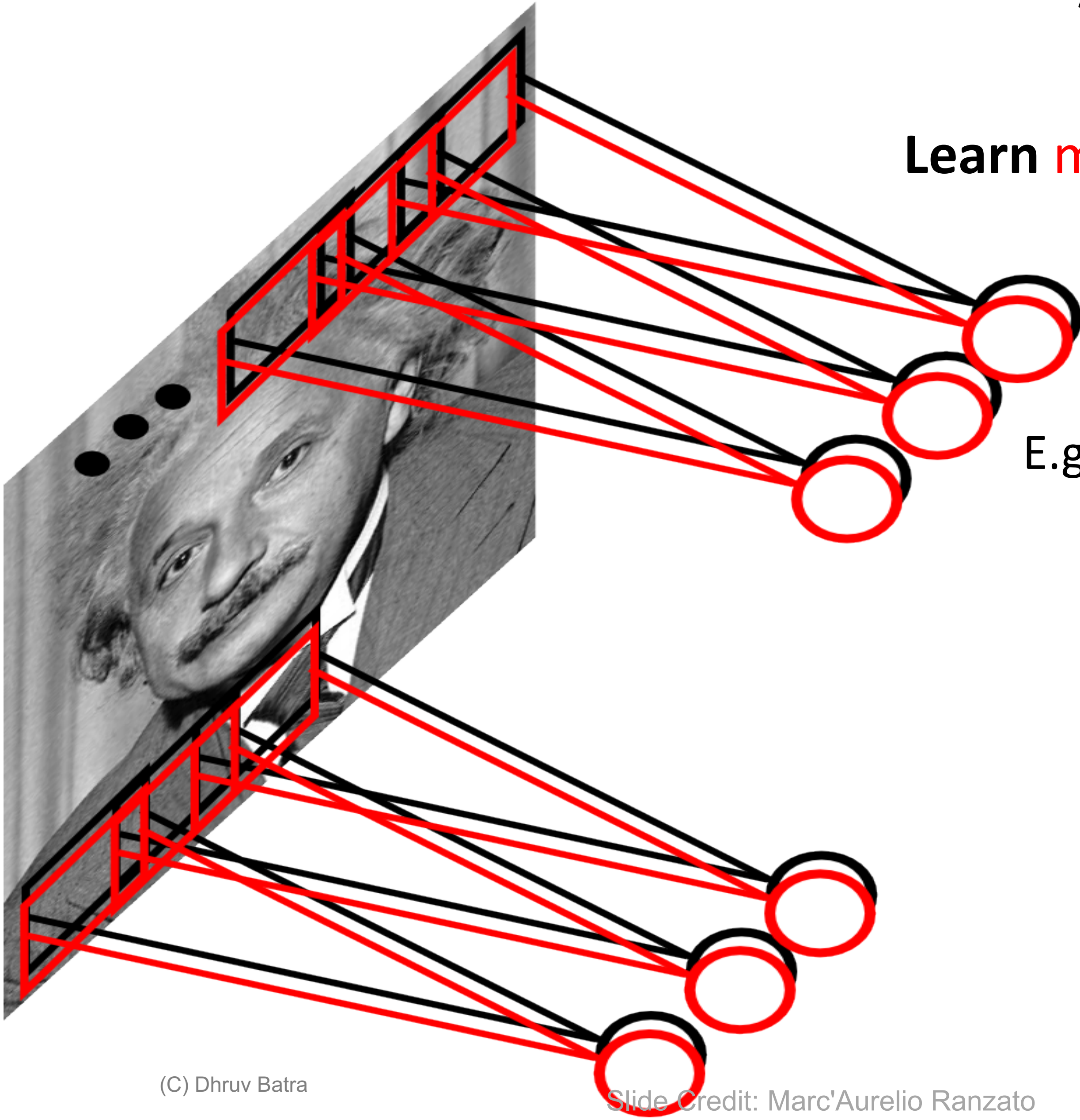
Convolutional Layer

$$* \begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix} =$$



Slide Credit: Marc'Aurelio Ranzato

Convolutional Layer

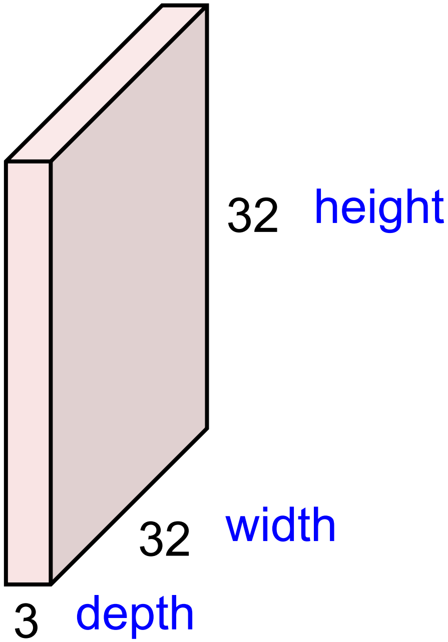


Learn multiple filters.

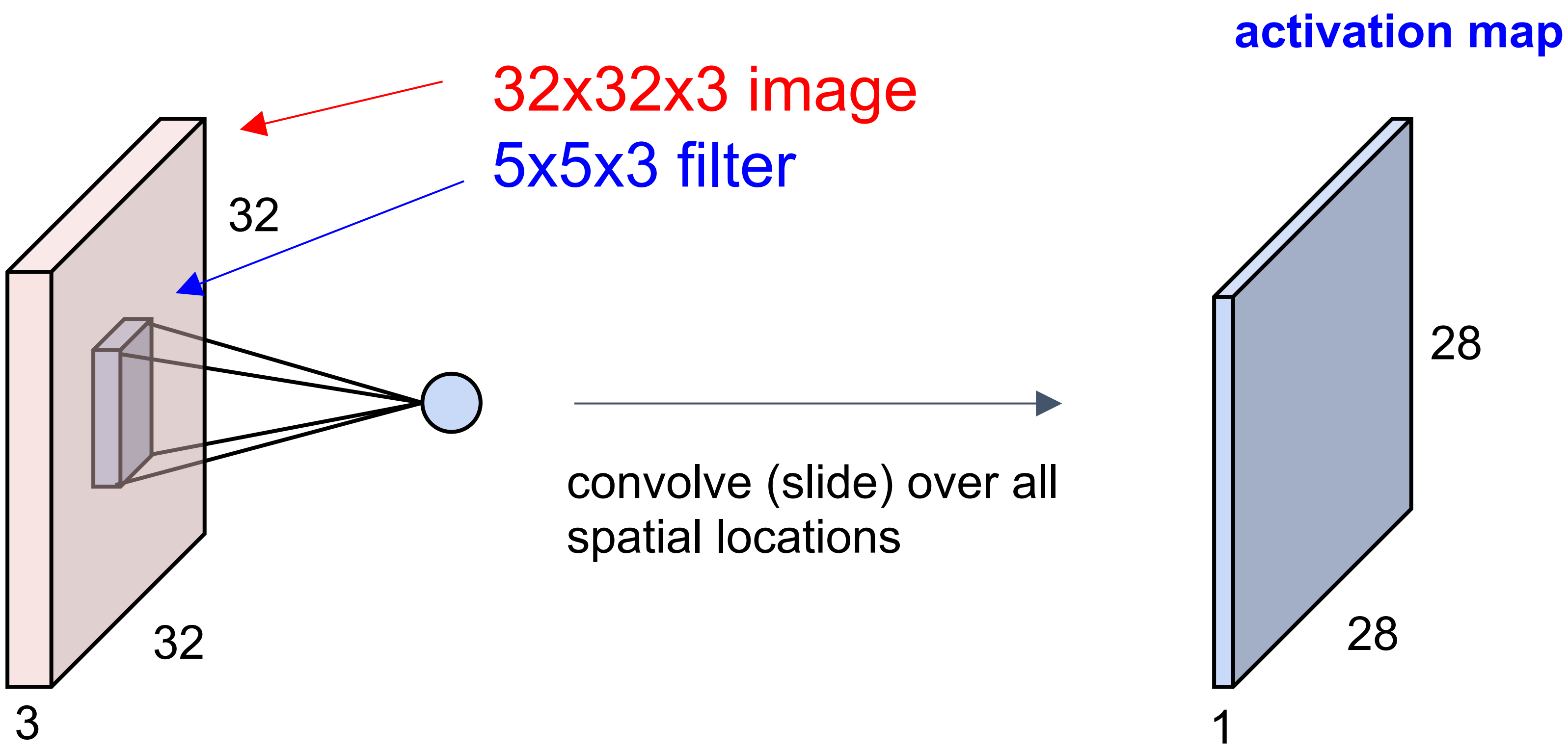
E.g.: 200x200 image
100 Filters
Filter size: 10x10
10K parameters

Convolution Layer

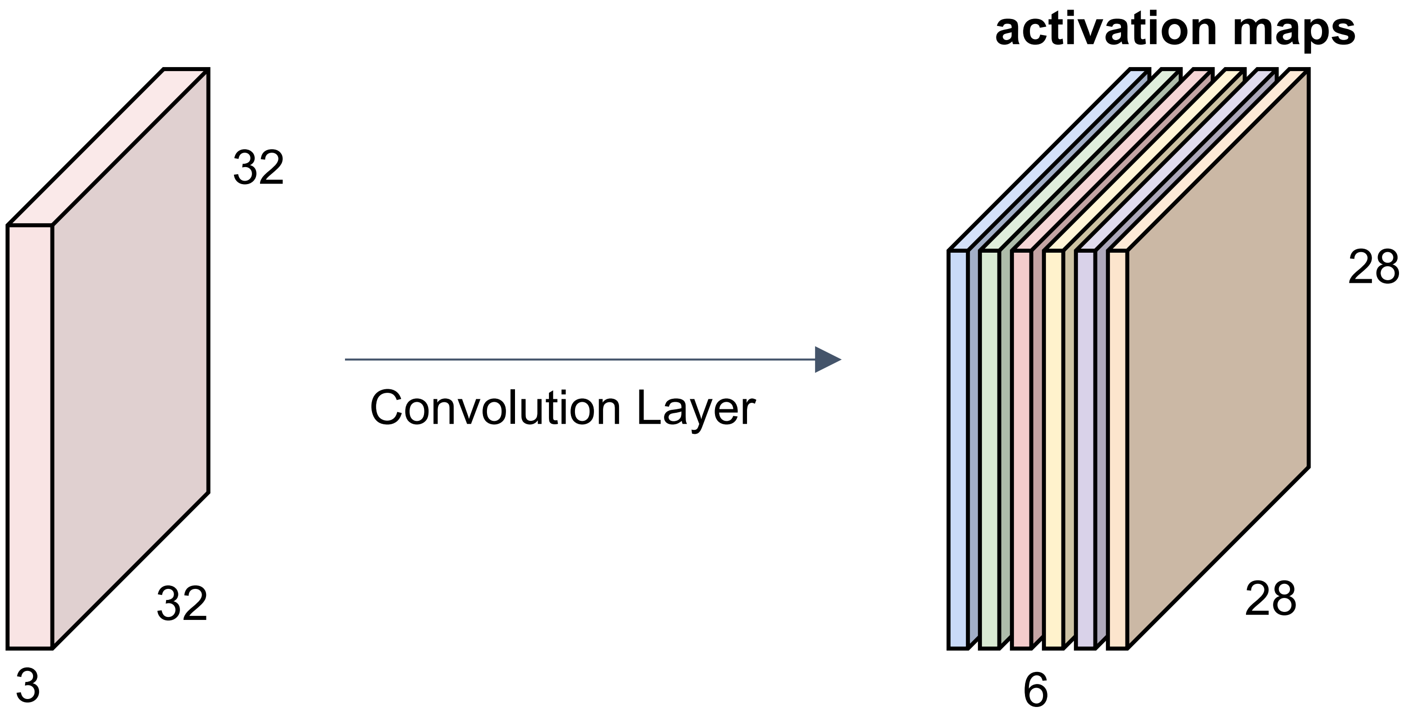
32x32x3 image -> preserve spatial structure



Convolution Layer

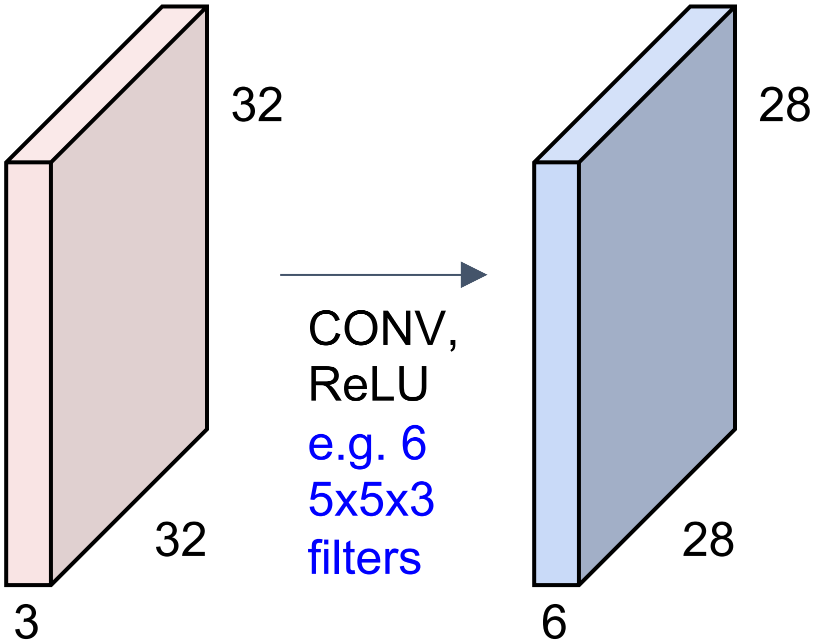


Multiple filters: if we have 6 5x5 filters, we'll get 6 separate activation maps:

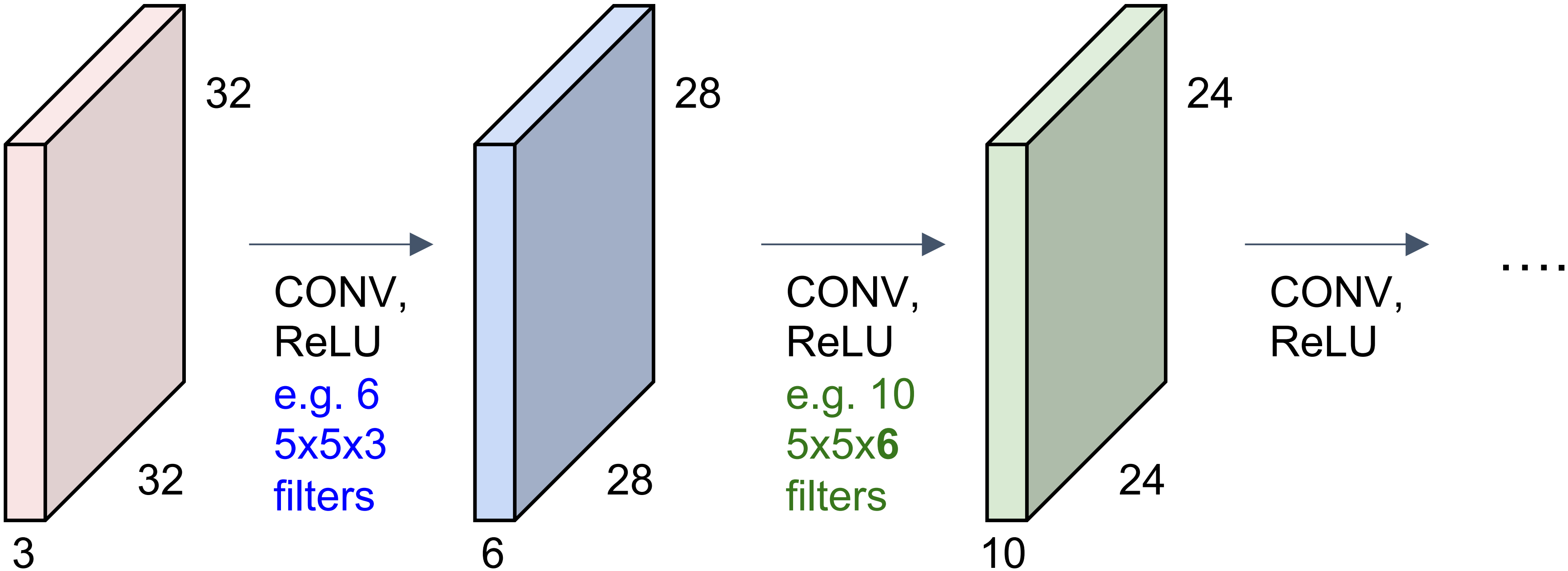


We stack these up to get a "new image" of size 28x28x6!

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



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

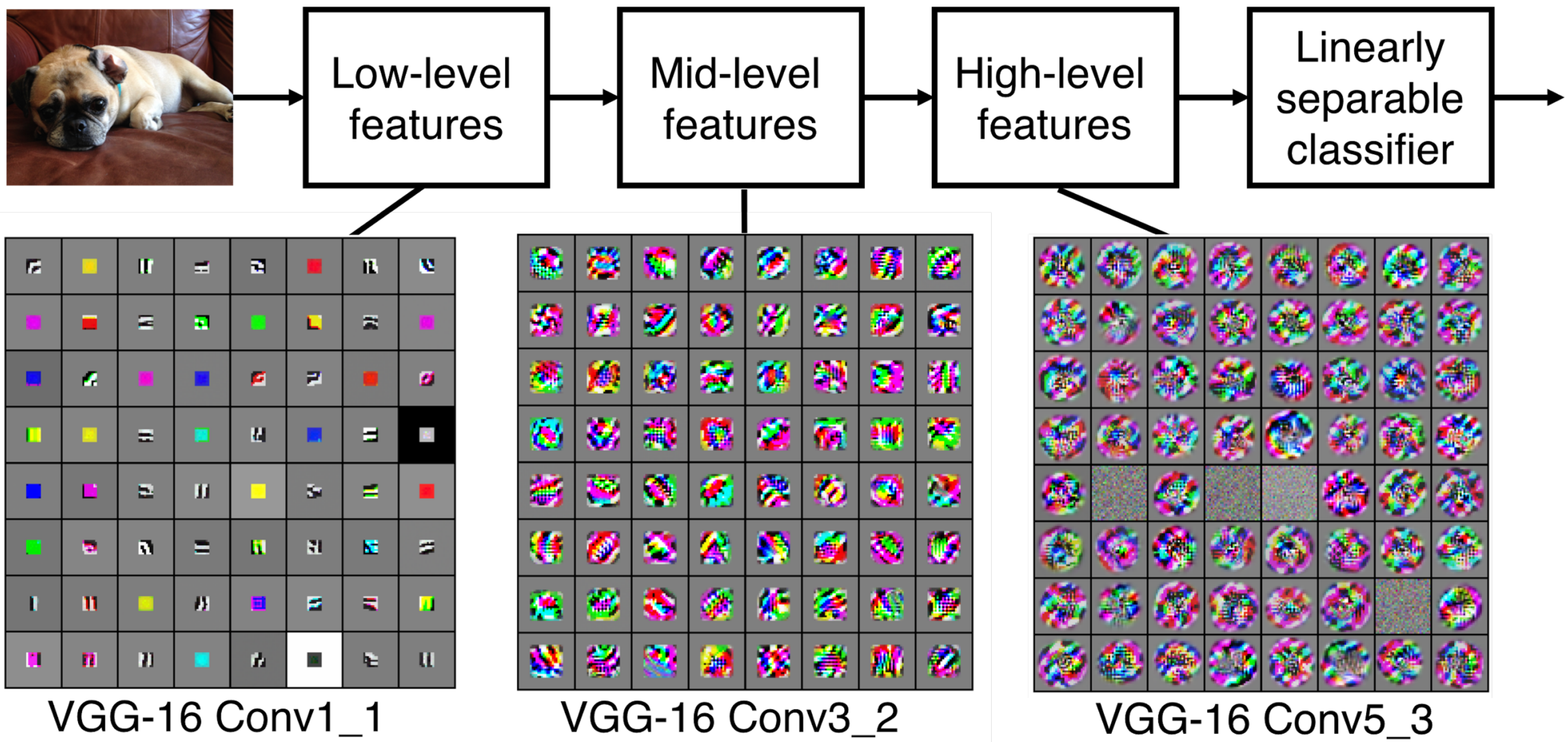


Slide Credit: Fei-Fei Li, Justin Johnson, Serena Yeung, CS 231n

Preview

[Zeiler and Fergus 2013]

Visualization of VGG-16 by Lane McIntosh. VGG-16 architecture from [Simonyan and Zisserman 2014].



Visualizing Learned Filters

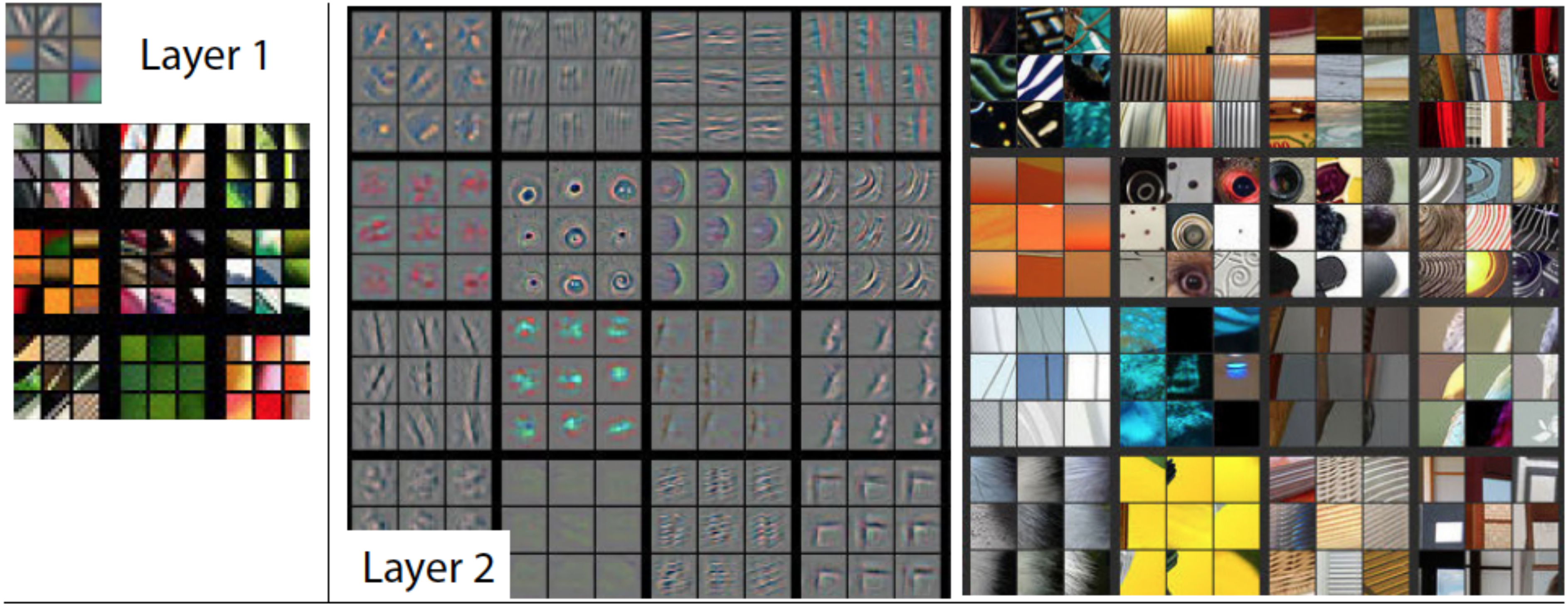


Figure Credit: [Zeiler & Fergus ECCV14]

Visualizing Learned Filters

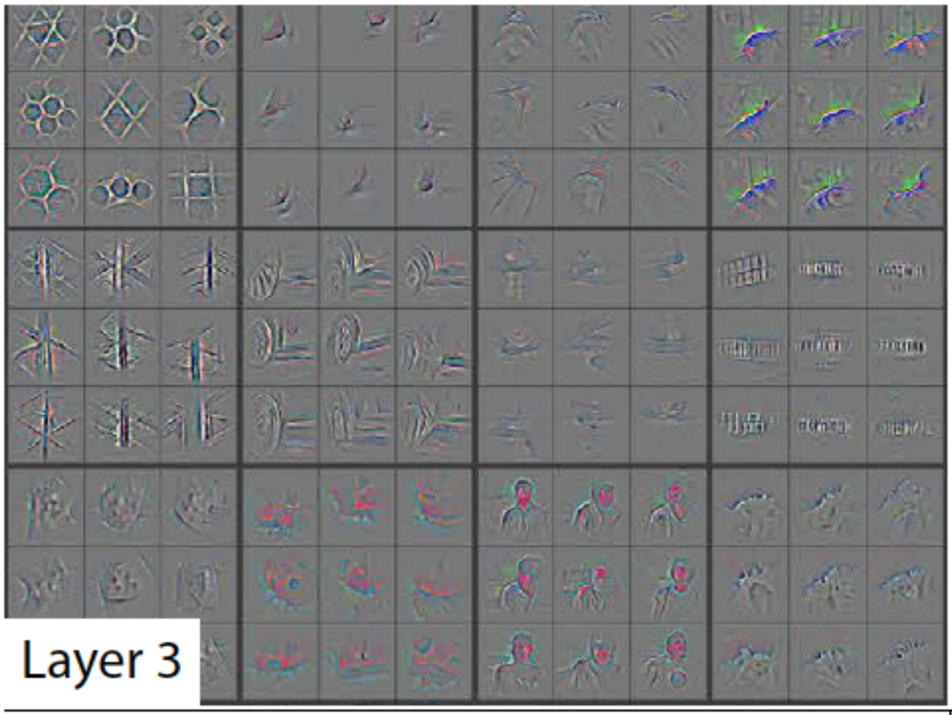


Figure Credit: [Zeiler & Fergus ECCV14]

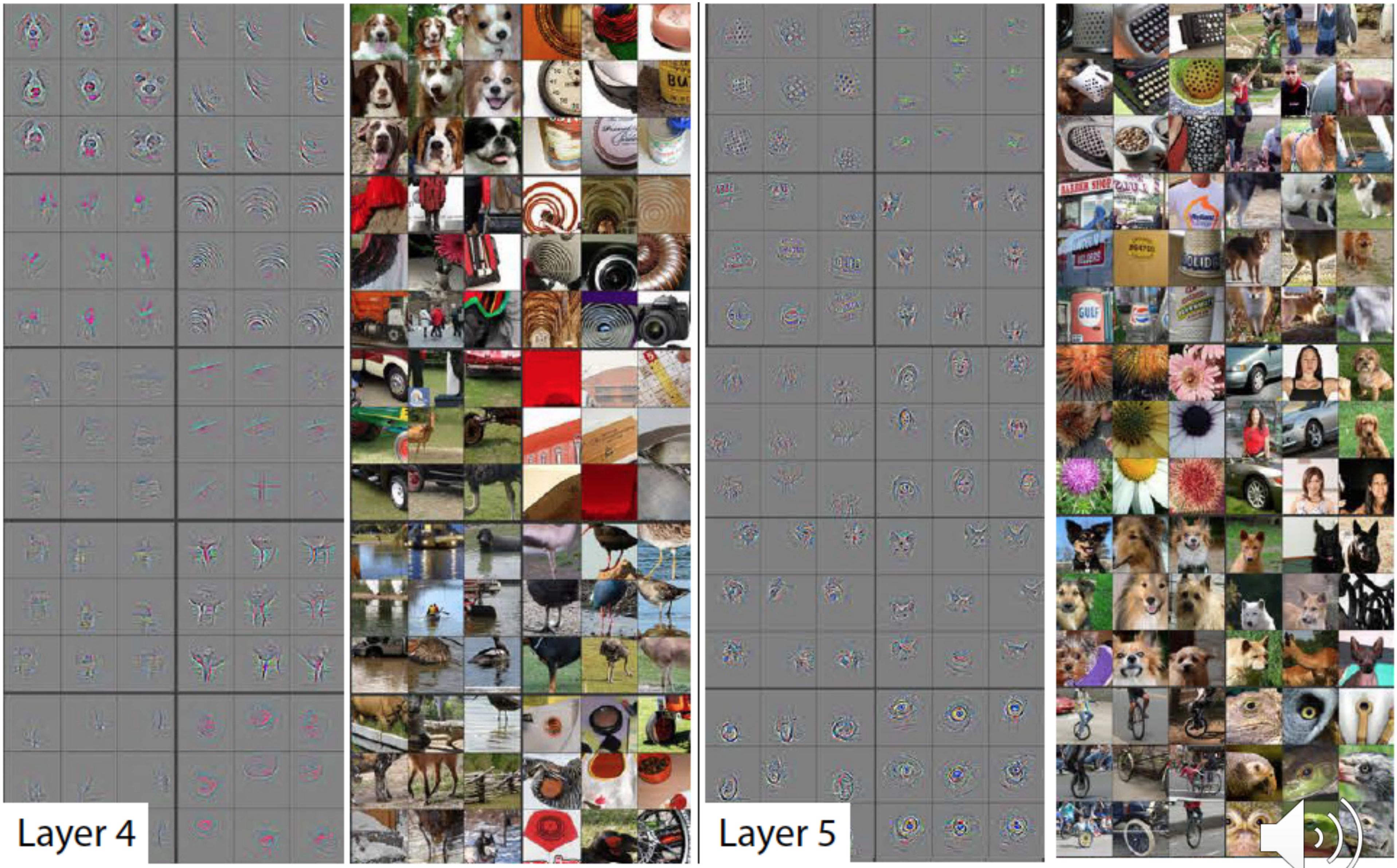
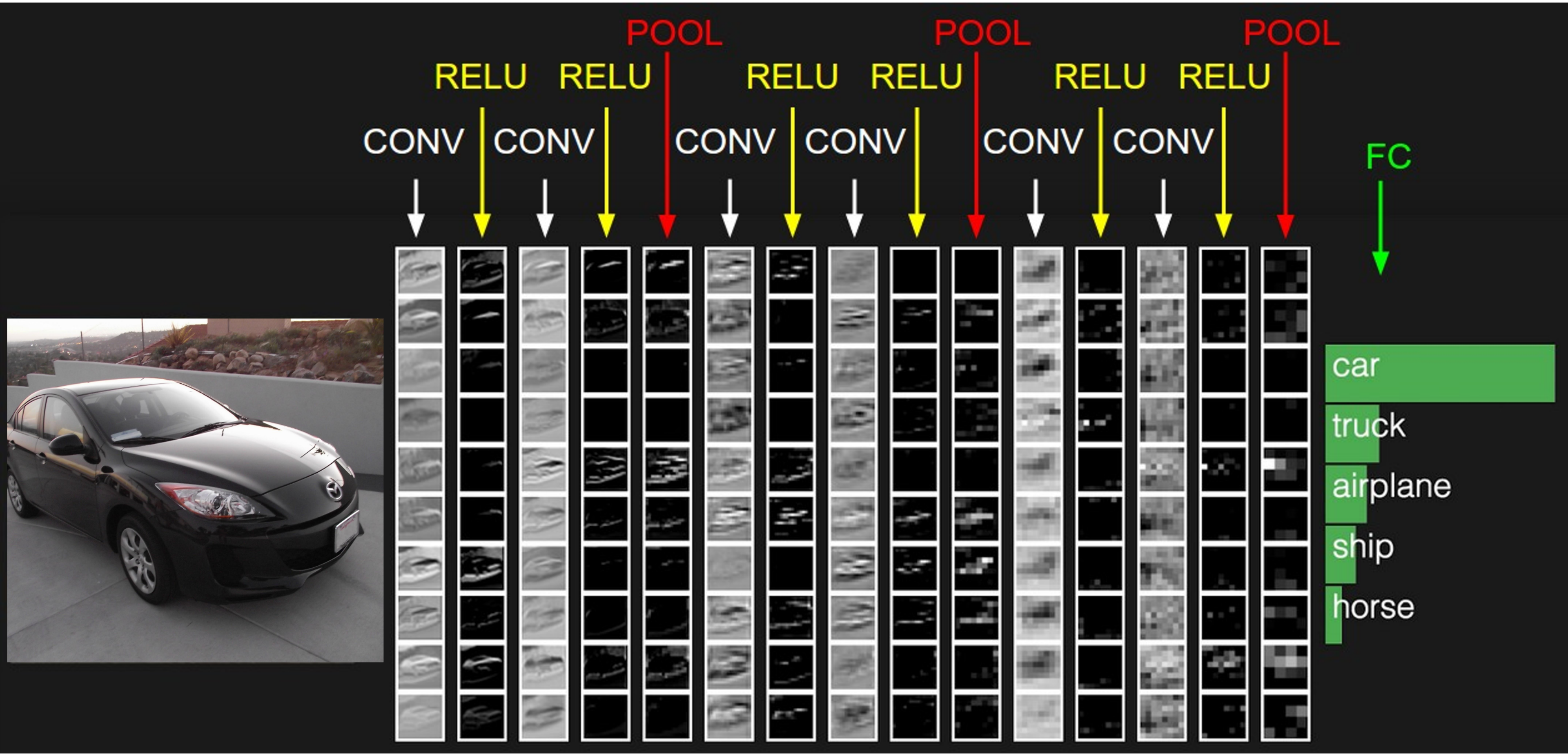
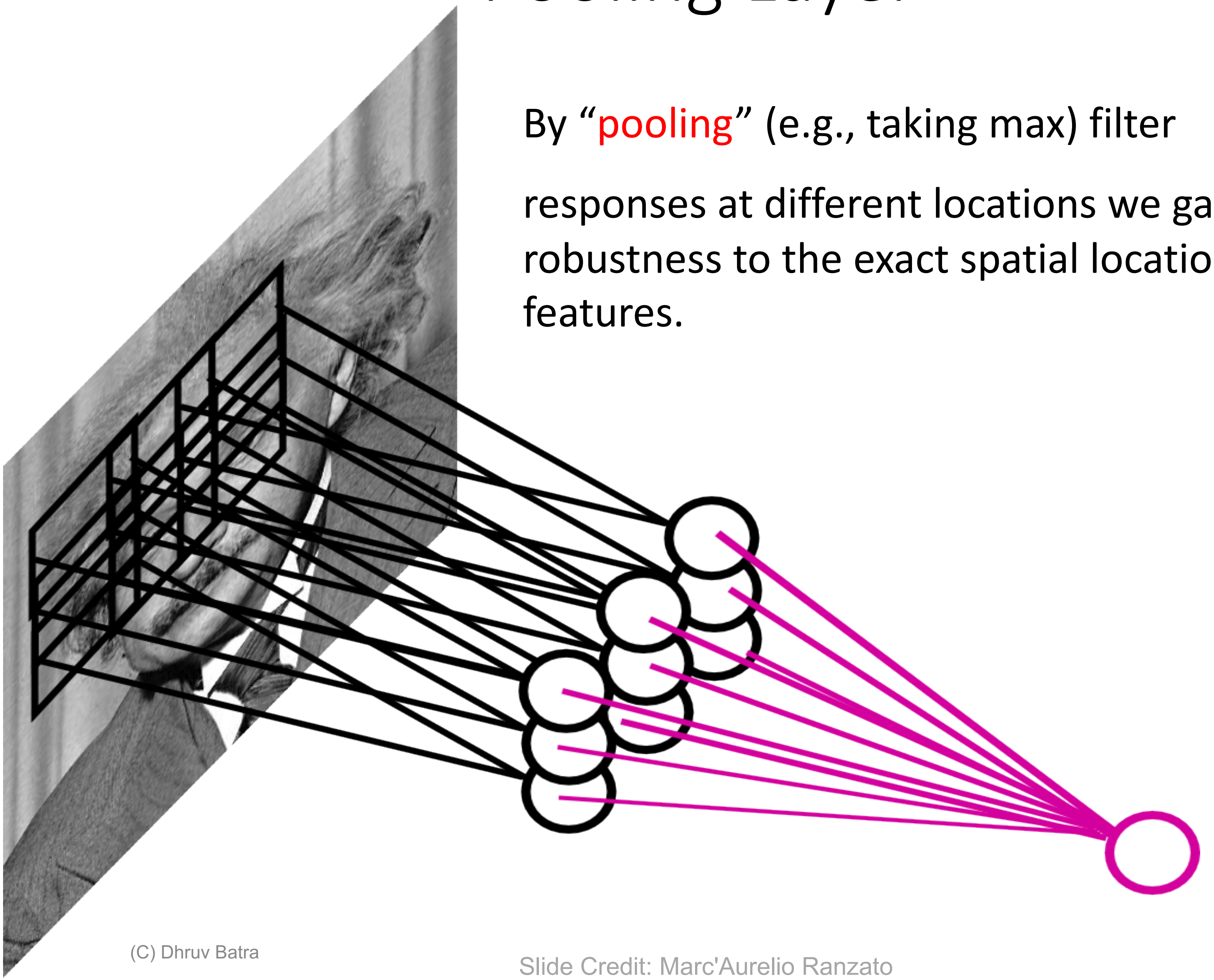


Figure Credit: [Zeiler & Fergus ECCV14]

two more layers to go: POOL/FC



Pooling Layer

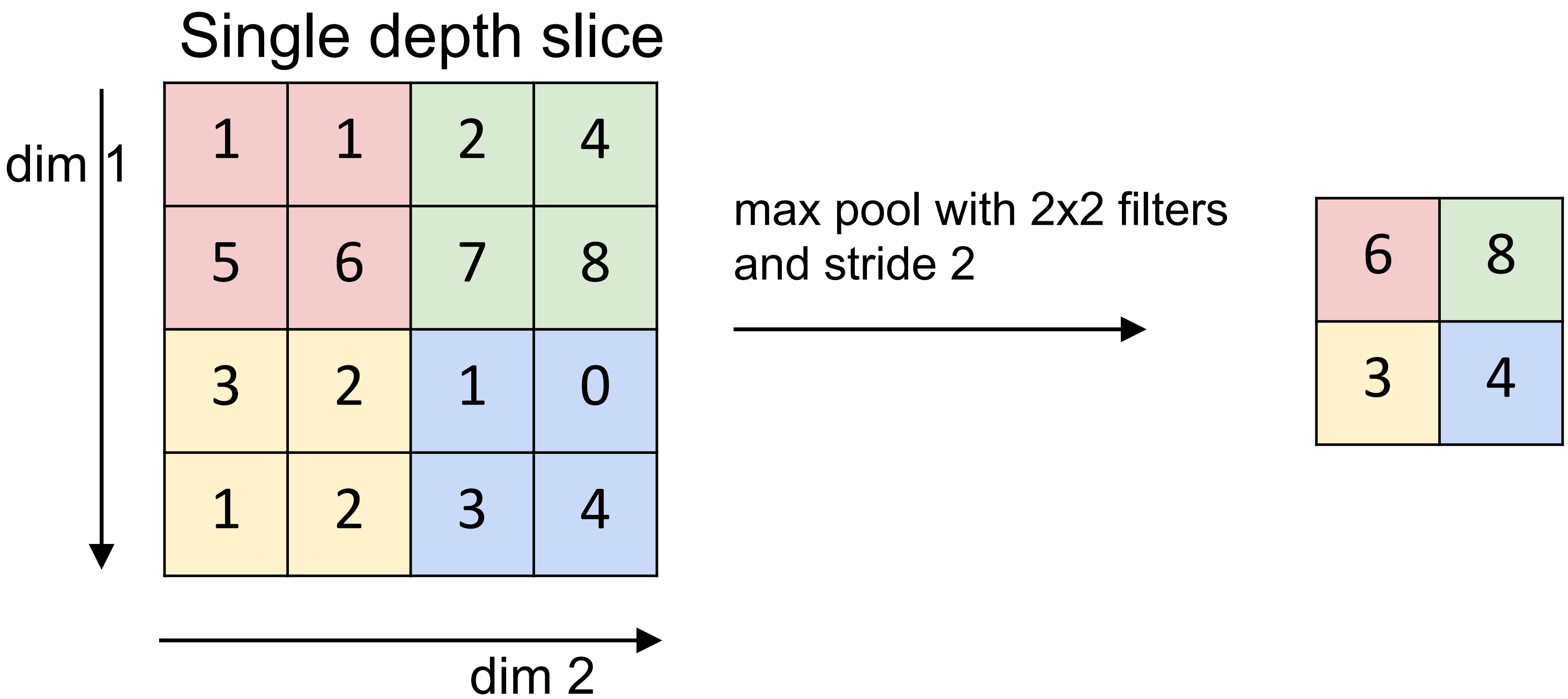


By "pooling" (e.g., taking max) filter

responses at different locations we gain robustness to the exact spatial location of features.

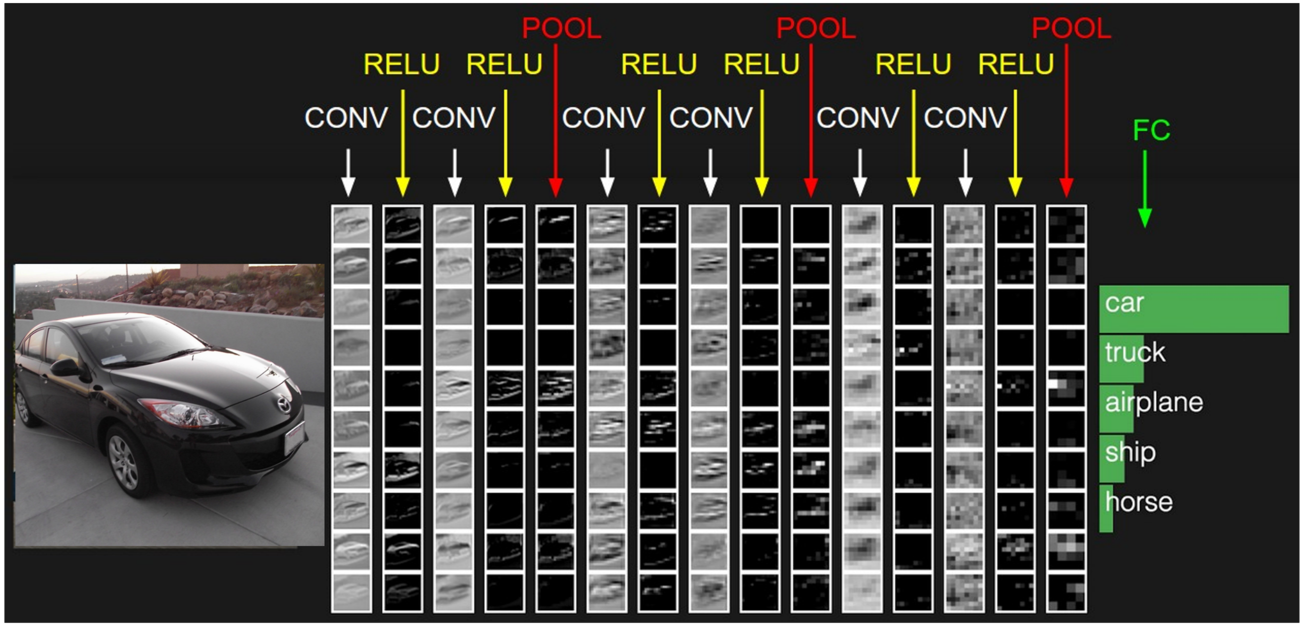
Slide Credit: Marc'Aurelio Ranzato

MAX POOLING


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

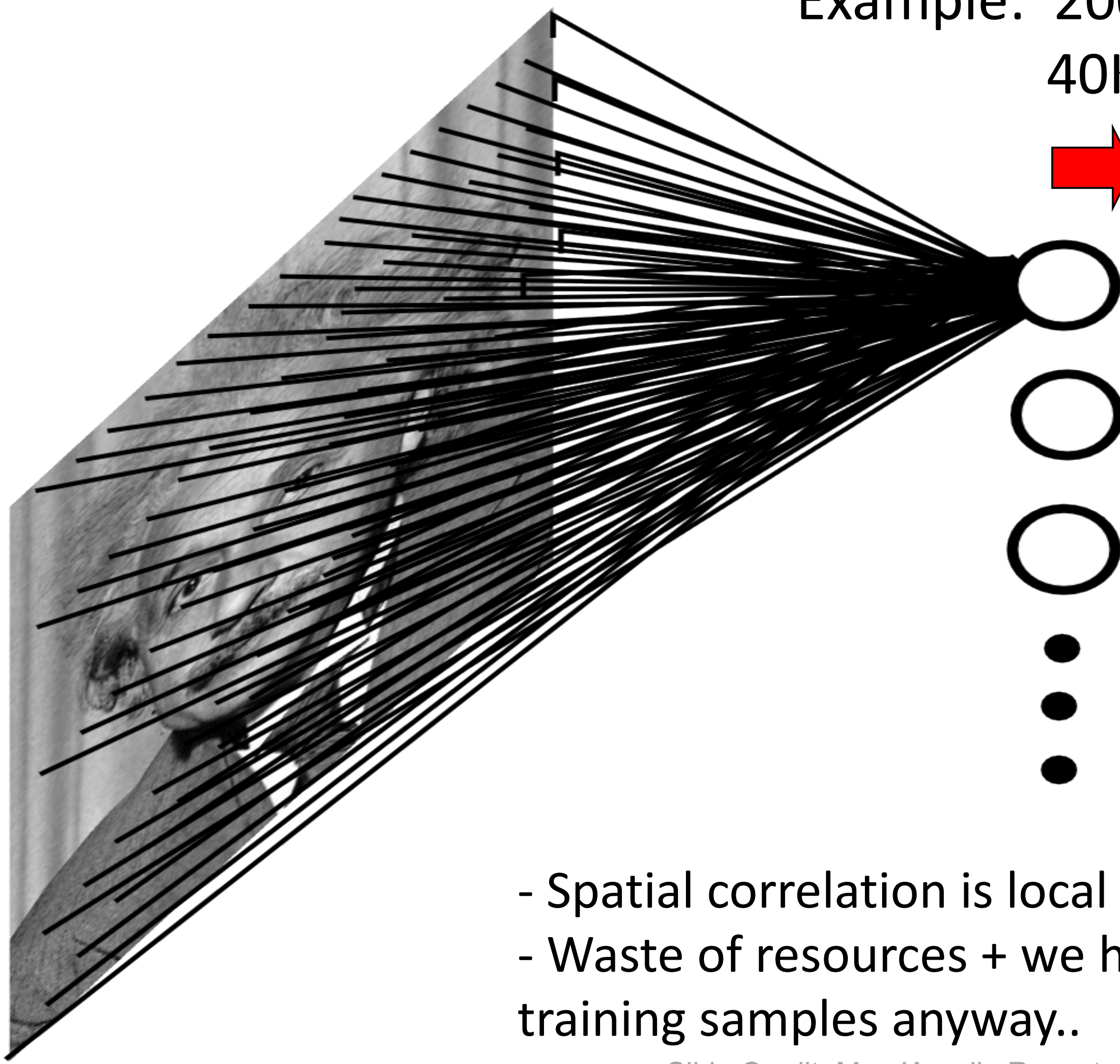
Fully Connected Layer (FC layer)

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



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

Fully Connected Layer



Example: 200x200 image
40K hidden units
→ **~2B parameters**!!!

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

Slide Credit: Marc'Aurelio Ranzato

3. Learning CNN Parameters

Suppose: 3 training examples, 3 classes.
With some W the scores $f(x, W) = Wx$ are:

cat	**3.2**	1.3	2.2
car	5.1	**4.9**	2.5
frog	-1.7	2.0	**-3.1**

A **loss function** tells how good our current classifier is

Given a dataset of examples

$$\{(x_i, y_i)\}_{i=1}^{N}$$

Where x_i is image and
y_i is (integer) label

Loss over the dataset is a sum of loss over examples:

$$L = \frac{1}{N} \sum_i L_i(f(x_i, W), y_i)$$

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

How to minimize the loss by changing the weights?
Strategy: **Follow the slope of the loss function**

Strategy: **Follow the slope**

In 1-dimension, the derivative of a function:

$$\frac{df(x)}{dx} = \lim_{h \to 0} \frac{f(x+h) - f(x)}{h}$$

In multiple dimensions, the **gradient** is the vector of (partial derivatives) along each dimension

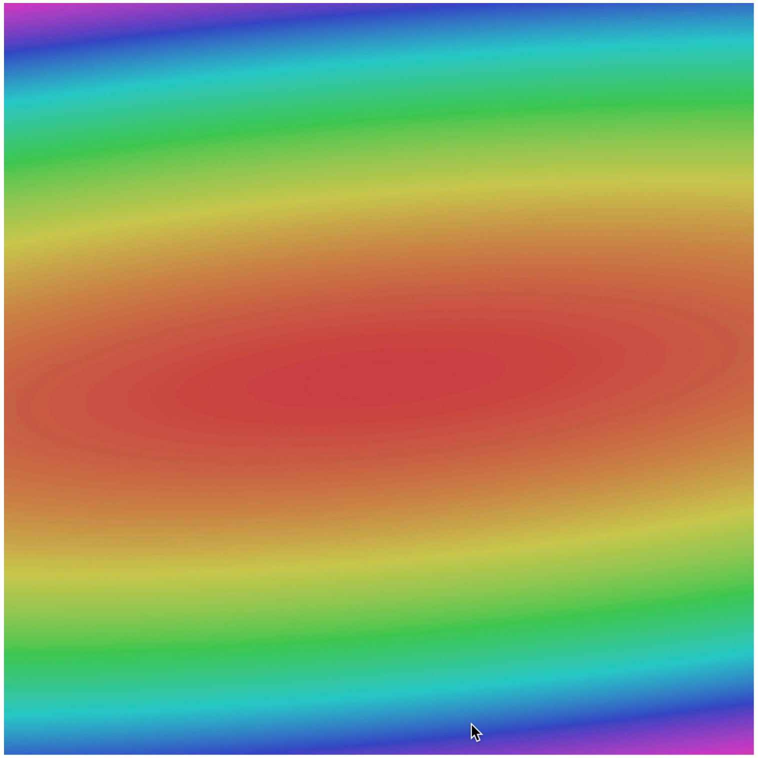
The slope in any direction is the **dot product** of the direction with the gradient

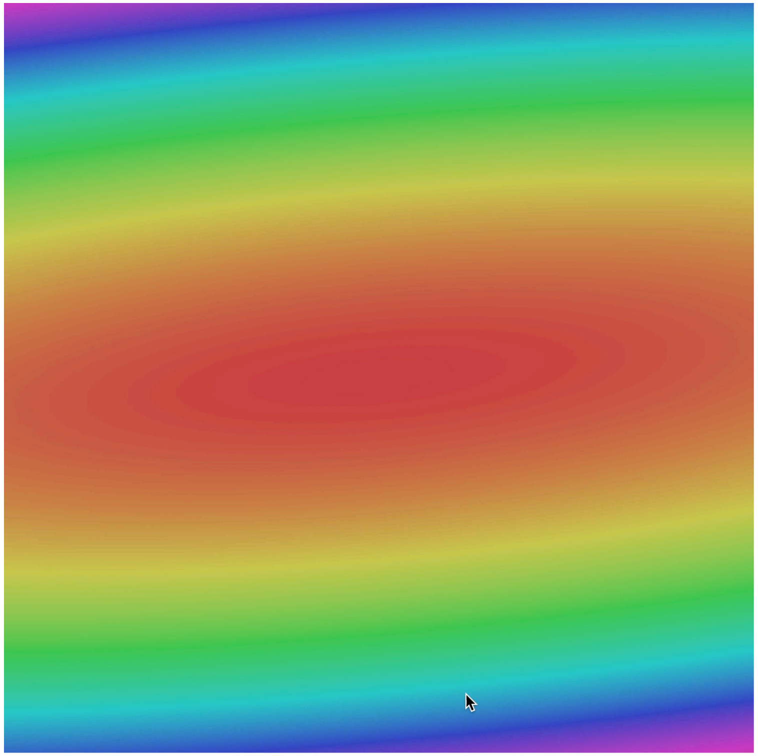
The direction of steepest descent is the **negative gradient**

Gradient Descent

```
# Vanilla Gradient Descent

while True:
  weights_grad = evaluate_gradient(loss_fun, data, weights)
  weights += - step_size * weights_grad # perform parameter update
```





Stochastic Gradient Descent (SGD)

$$L(W) = \frac{1}{N}\sum_{i=1}^{N} L_i(x_i, y_i, W) + \lambda R(W)$$

$$\nabla_W L(W) = \frac{1}{N}\sum_{i=1}^{N} \nabla_W L_i(x_i, y_i, W) + \lambda \nabla_W R(W)$$

Full sum expensive when N is large!

Approximate sum using a **minibatch** of examples
32 / 64 / 128 common

```
# Vanilla Minibatch Gradient Descent

while True:
  data_batch = sample_training_data(data, 256) # sample 256 examples
  weights_grad = evaluate_gradient(loss_fun, data_batch, weights)
  weights += - step_size * weights_grad # perform parameter update
```

How do we compute gradients?

- Analytic or “Manual” Differentiation
- Symbolic Differentiation
- Numerical Differentiation
- Automatic Differentiation!
  - Forward mode AD
  - Reverse mode AD
    - aka “backpropagation”
  - Implemented in specialized frameworks:
    - pytorch (Facebook)
    - TensorFlow (Google) frameworks
  - Main computation, mainly done on GPU (or TPU)

4. Applications in Perception

- From pixels to concepts:
    - Image processing
    - Object classification
    - Object detection
    - Pixelwise segmentation

Colorization

- Given a grayscale image, colorize the image realistically
- Zhang et al. pose colorization as classification task and use class-rebalancing to improve results
- Demonstrate higher rates of fooling humans using “colorization Turing test”

Colorful Image Colorization. Richard Zhang, Phillip Isola, Alexei A. Efros. ECCV 2016.

DeOldify

https://github.com/jantic/DeOldify

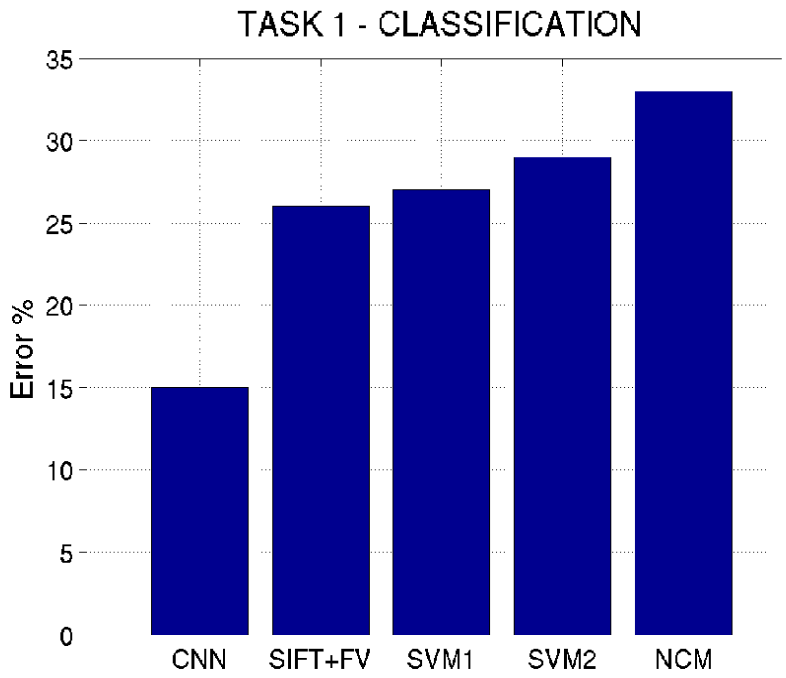
Super-Resolution

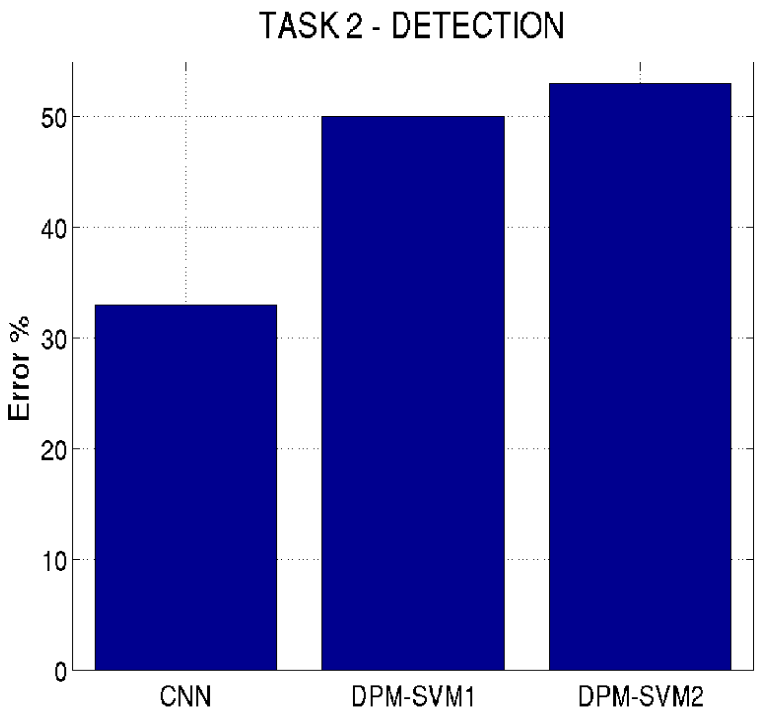
Low resolution

High resolution

Object Classification Revolution

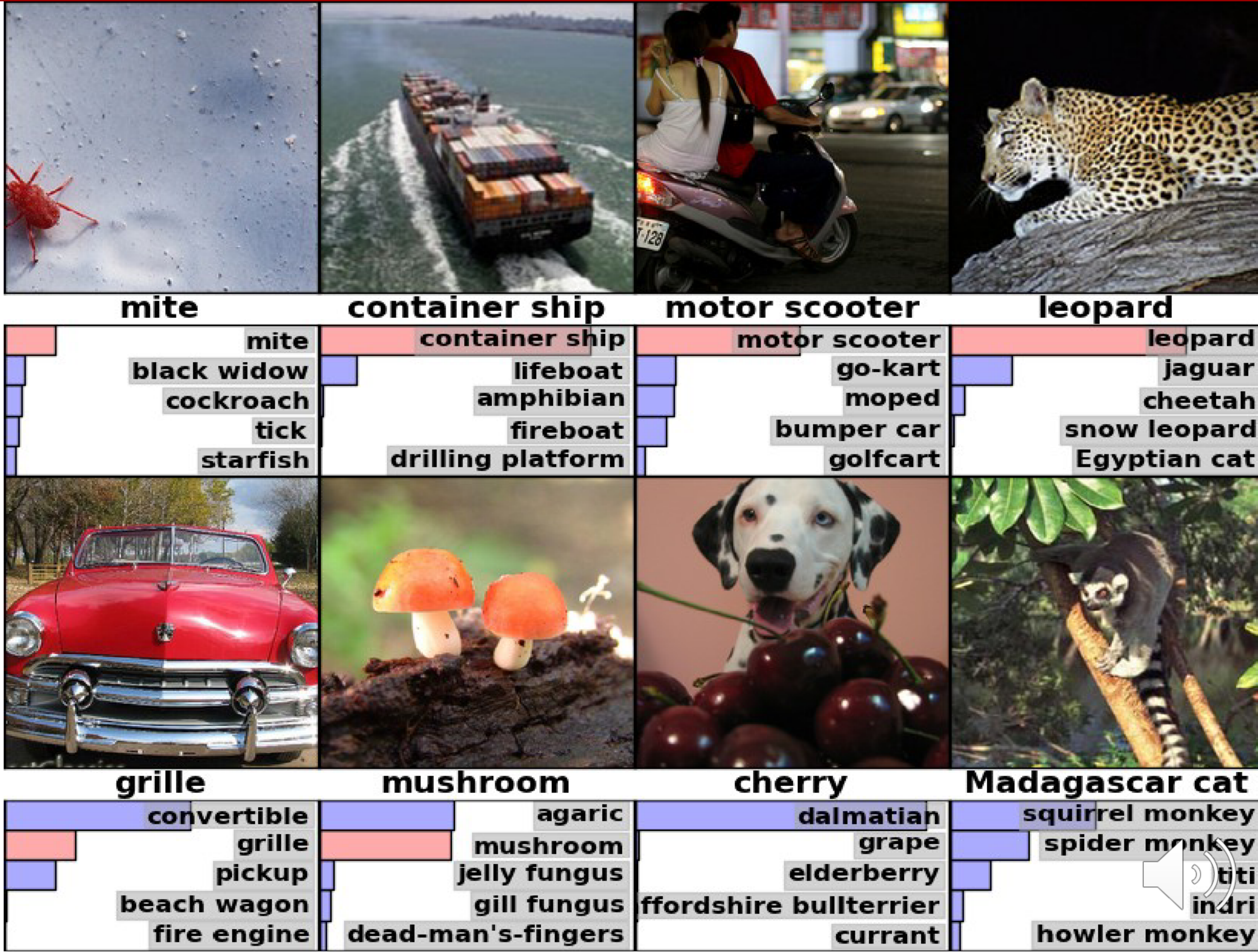
Results: ILSVRC 2012





Krizhevsky et al. "ImageNet Classification with deep CNNs" NIPS 2012

Ranzato

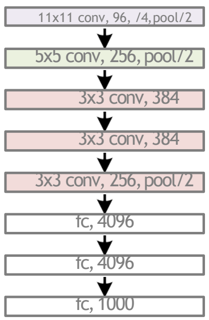


mite
mite
black widow
cockroach
tick
starfish
container ship
container ship
lifeboat
amphibian
fireboat
drilling platform
motor scooter
motor scooter
go-kart
moped
bumper car
golfcart
leopard
leopard
jaguar
cheetah
snow leopard
Egyptian cat
grille
convertible
grille
pickup
beach wagon
fire engine
mushroom
agaric
mushroom
jelly fungus
gill fungus
dead-man's-fingers
cherry
dalmatian
grape
elderberry
ffordshire bullterrier
currant
Madagascar cat
squirrel monkey
spider monkey
titi
indri
howler monkey

Revolution of Depth

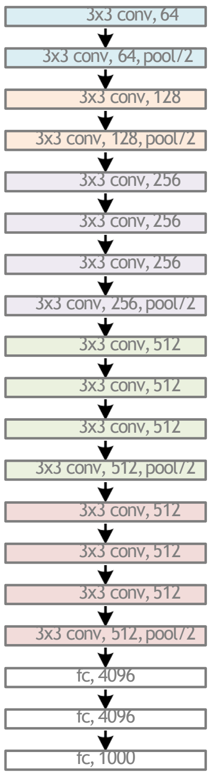
AlexNet, 8 layers

(ILSVRC 2012)



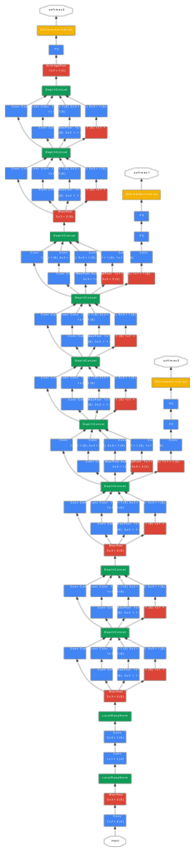
VGG, 19 layers

(ILSVRC 2014)



GoogleNet, 22 layers

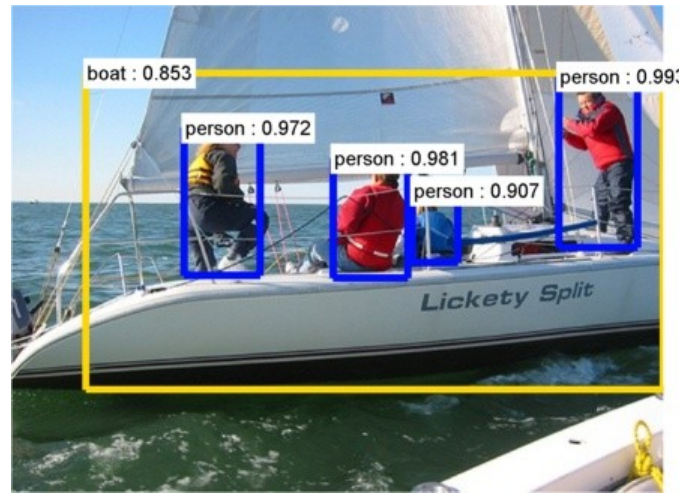
(ILSVRC 2014)



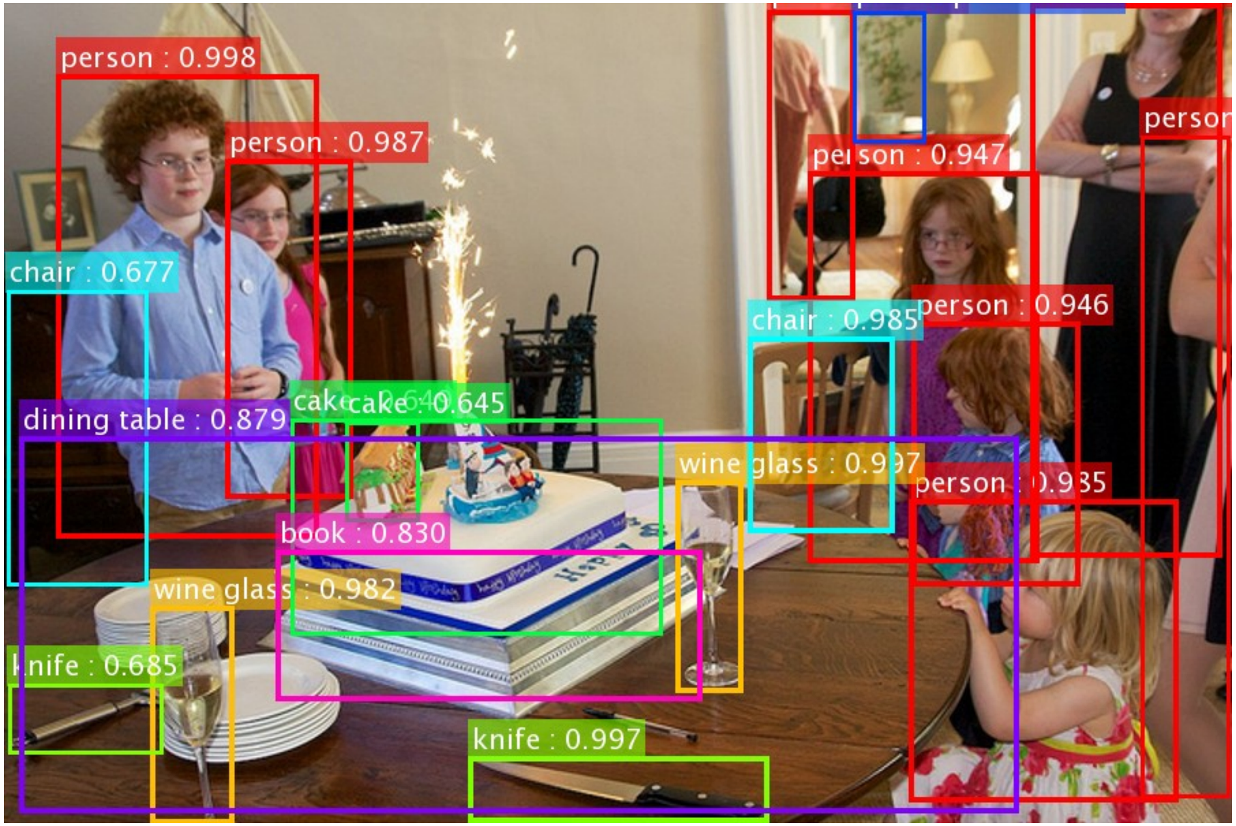
Kaiming He, Xiangyu Zhang, Shaoqing Ren, & Jian Sun. “Deep Residual Learning for Image Recognition”. CVPR 2016.

Object Detection

Image Classification (what?)



Object Detection (what + where?)



ResNet's object detection result on COCO

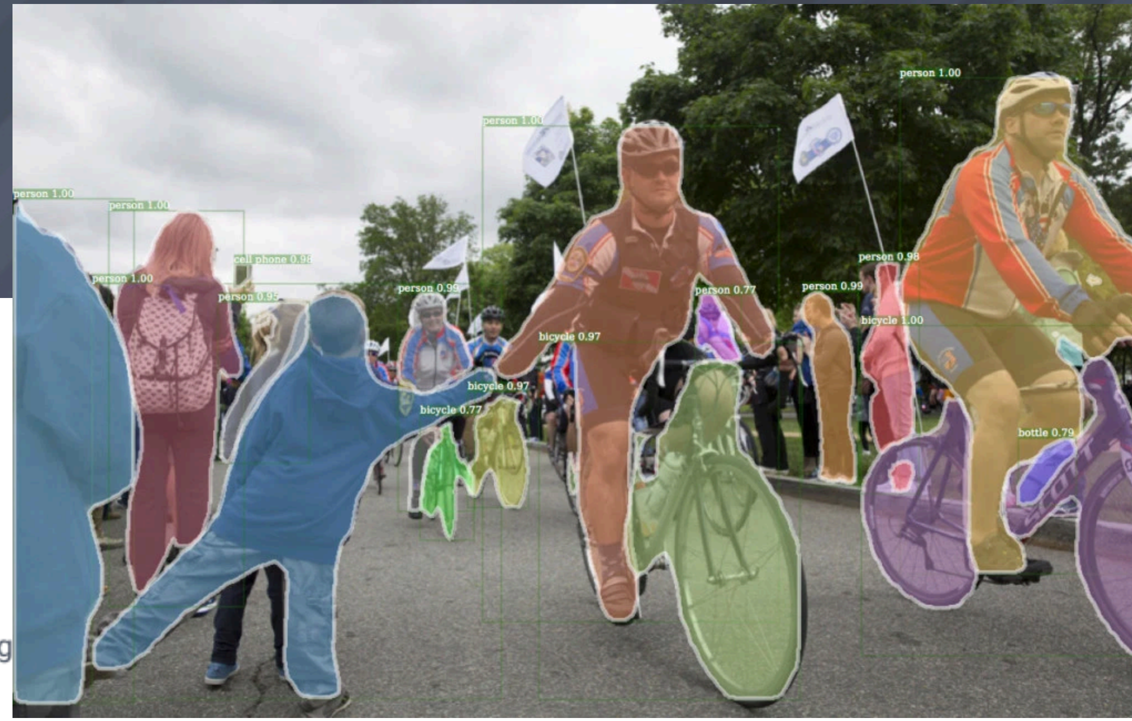
Kaiming He, Xiangyu Zhang, Shaoqing Ren, & Jian Sun. "Deep Residual Learning for Image Recognition". CVPR 2016.
Shaoqing Ren, Kaiming He, Ross Girshick, & Jian Sun. "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks". NIPS 2015.

this video is available online: https://youtu.be/WZmSMkK9VuA

Results on real video. Models trained on MS COCO (80 categories). (frame-by-frame; no temporal processing)

Kaiming He, Xiangyu Zhang, Shaoqing Ren, & Jian Sun. "Deep Residual Learning for Image Recognition". CVPR 2016.
Shaoqing Ren, Kaiming He, Ross Girshick, & Jian Sun. "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks". NIPS 2015.

Detectron



Detectron includes implementations of the following

- Mask R-CNN — *Marr Prize at ICCV 2017*
- RetinaNet — *Best Student Paper Award at ICCV 2017*
- Faster R-CNN
- RPN
- Fast R-CNN
- R-FCN

using the following backbone network architectures:

- ResNeXt{50,101,152}
- ResNet{50,101,152}
- Feature Pyramid Networks (with ResNet/ResNeXt)
- VGG16

Summary

1. Supervised Learning is where we learn from (lots) of labels
2. Convolutional Neural Networks are specialized multi-layer networks
3. Learning CNN Parameters can be done via stochastic gradient descent
4. Applications in Perception illustrate how well this works