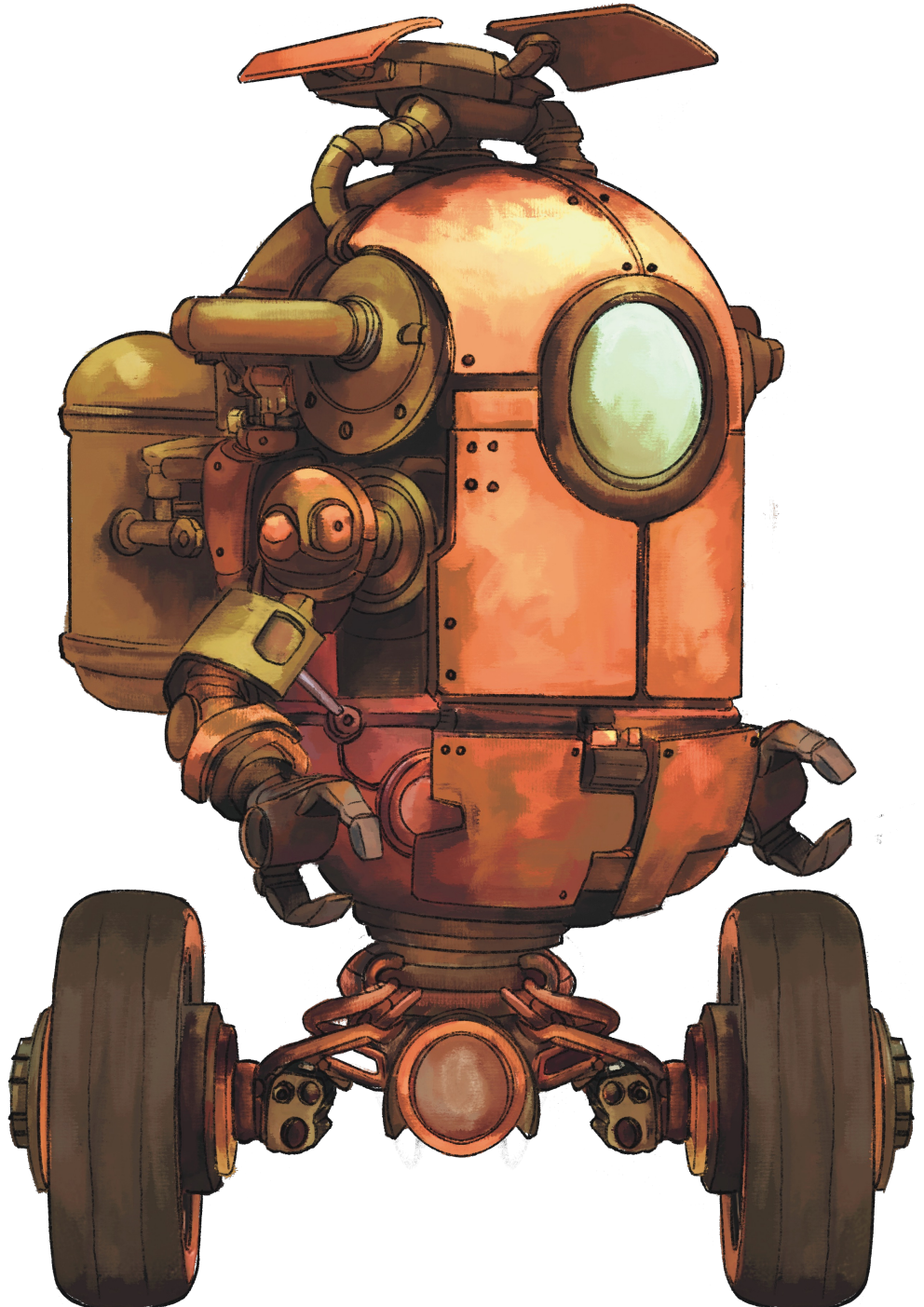
CS 3630, Fall 2026

Lecture 19 (Section 5.4):

Computer Vision and Convolutional Neural Networks



Motivation

- Robotics:
  - Perception, thinking, acting
- Deep learning has revolutionized perception
- Getting increasingly important in thinking/acting
- This lecture:
  - Basic Image Filtering
  - High-level intro to CNNs
  - Applications in robotics
- More resources:
  - Computer vision courses at Georgia Tech
  - Deep Learning courses at Georgia Tech
  - Great online resource:



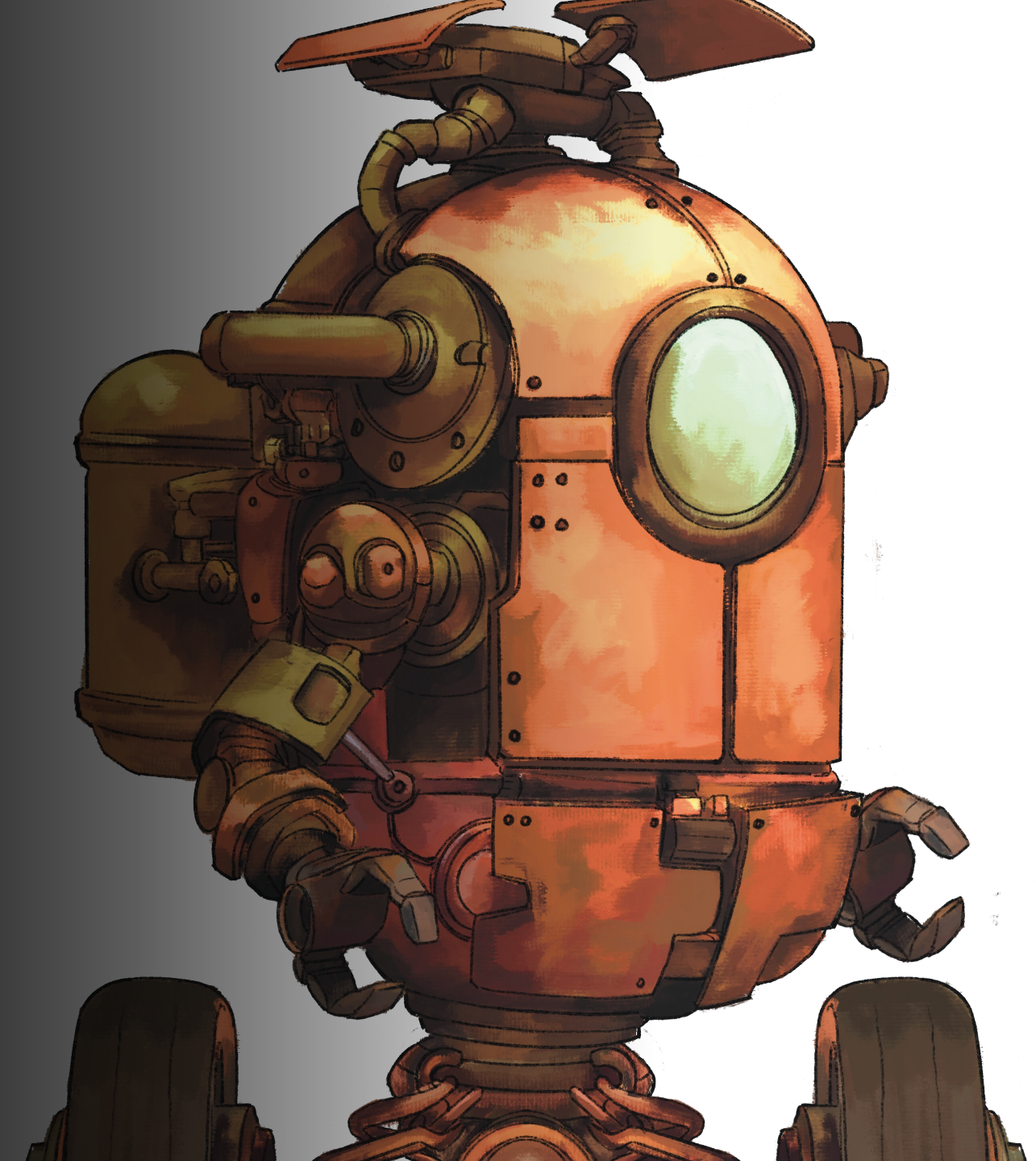
Image by Voyage



https://d2l.ai

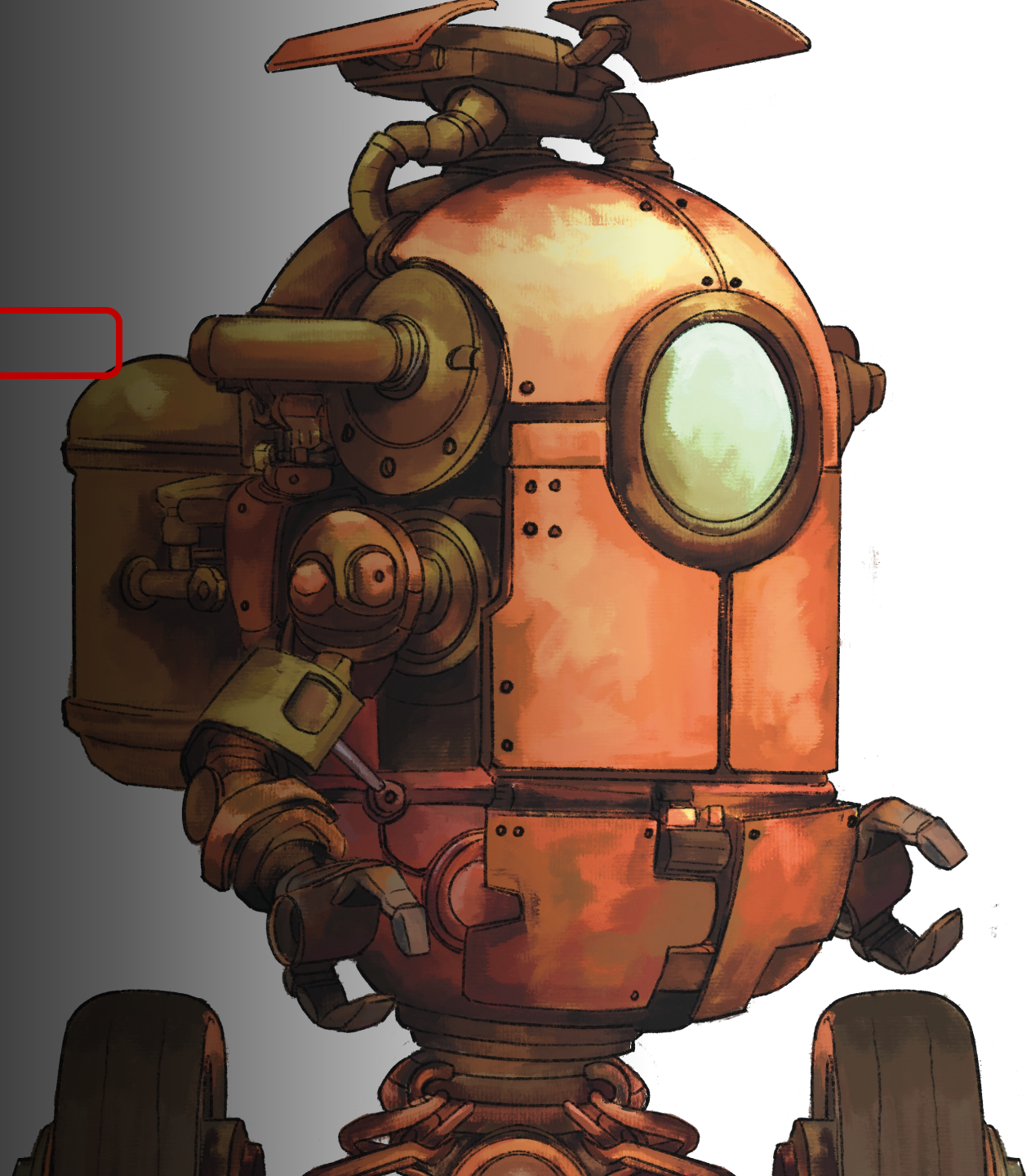
Outline

1. Digital Image Acquisition
2. Image Filtering
3. Supervised Learning
4. Multi-layer Perceptrons
5. Convolutional Neural Nets
6. Visualizing Learned Filters
7. Pooling and FC Layers
8. Applications in Vision
9. Object Classification
10. Applications in Robotics

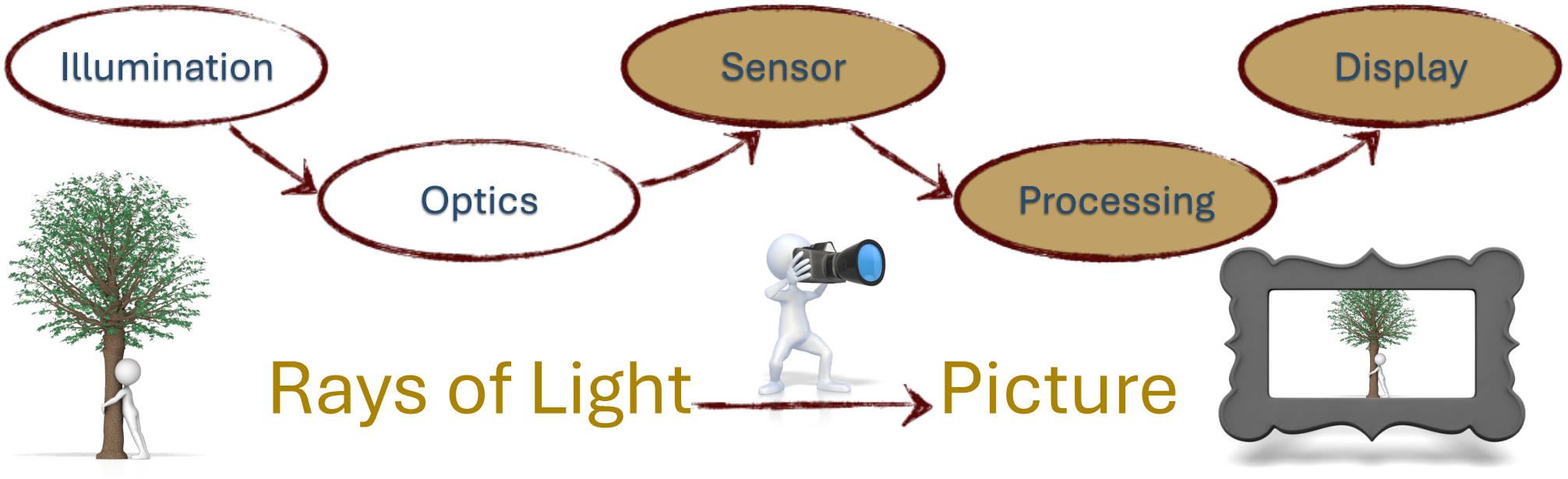


Outline

1. Digital Image Acquisition
2. Image Filtering
3. Supervised Learning
4. Multi-layer Perceptrons
5. Convolutional Neural Nets
6. Visualizing Learned Filters
7. Pooling and FC Layers
8. Applications in Vision
9. Object Classification
10. Applications in Robotics

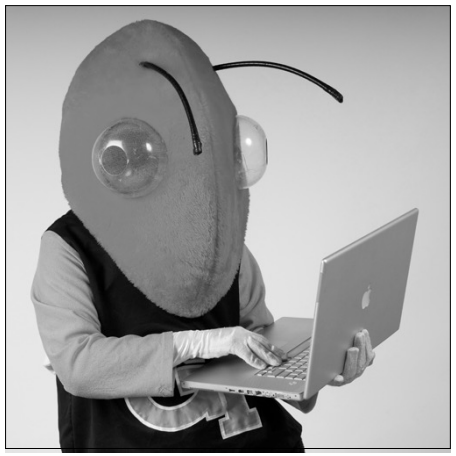


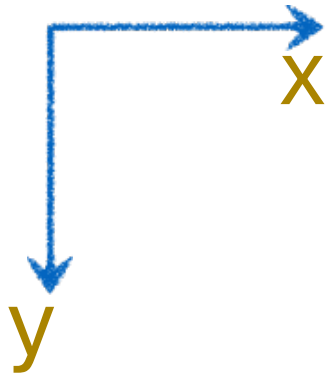
1. Digital Image Acquisition

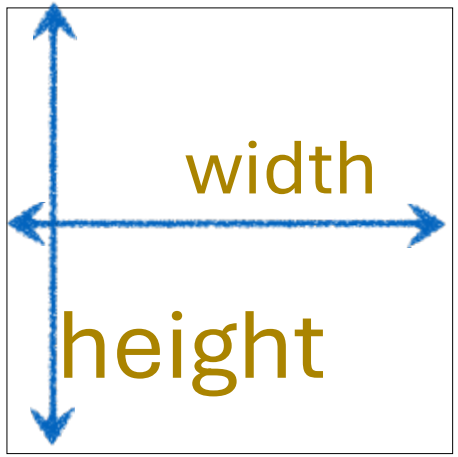


* Analog (incoming light) to digital (pixels)

A Digital Image!



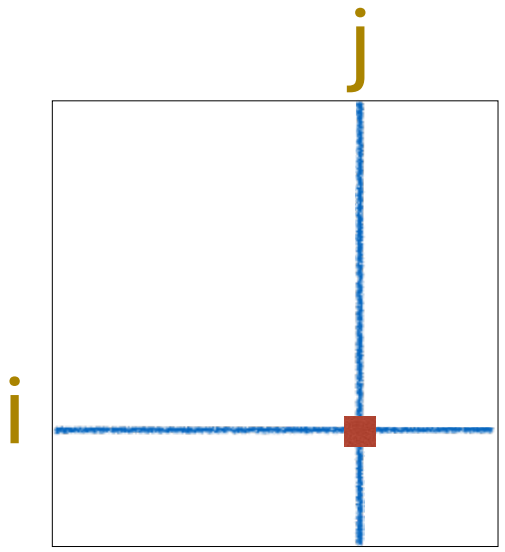




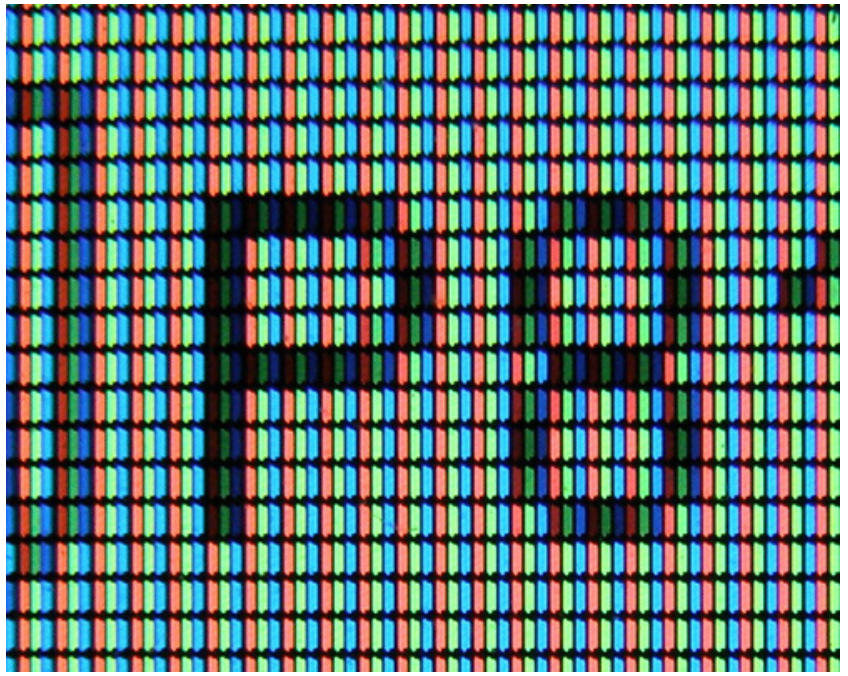
- Numeric representation in 2-D (x and y)
- Referred to as *I(x,y)* in continuous function form, *I(i,j)* in discrete
- Image Resolution: expressed in terms of Width and Height of the image

Pixel

A “picture element” that contains the light intensity at some location *(i,j)* in the image

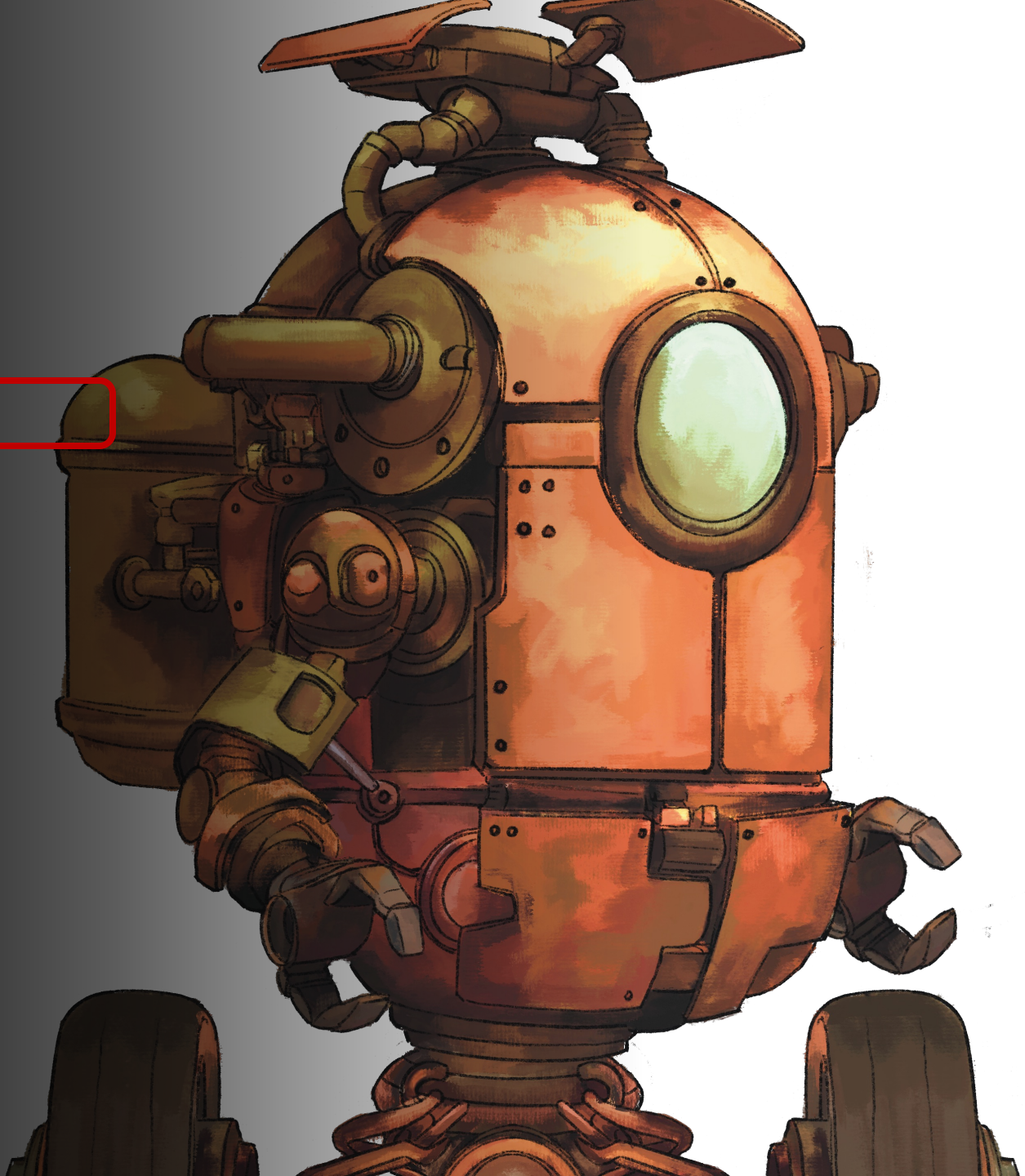


I(i,j) = Some Numeric Value



Outline

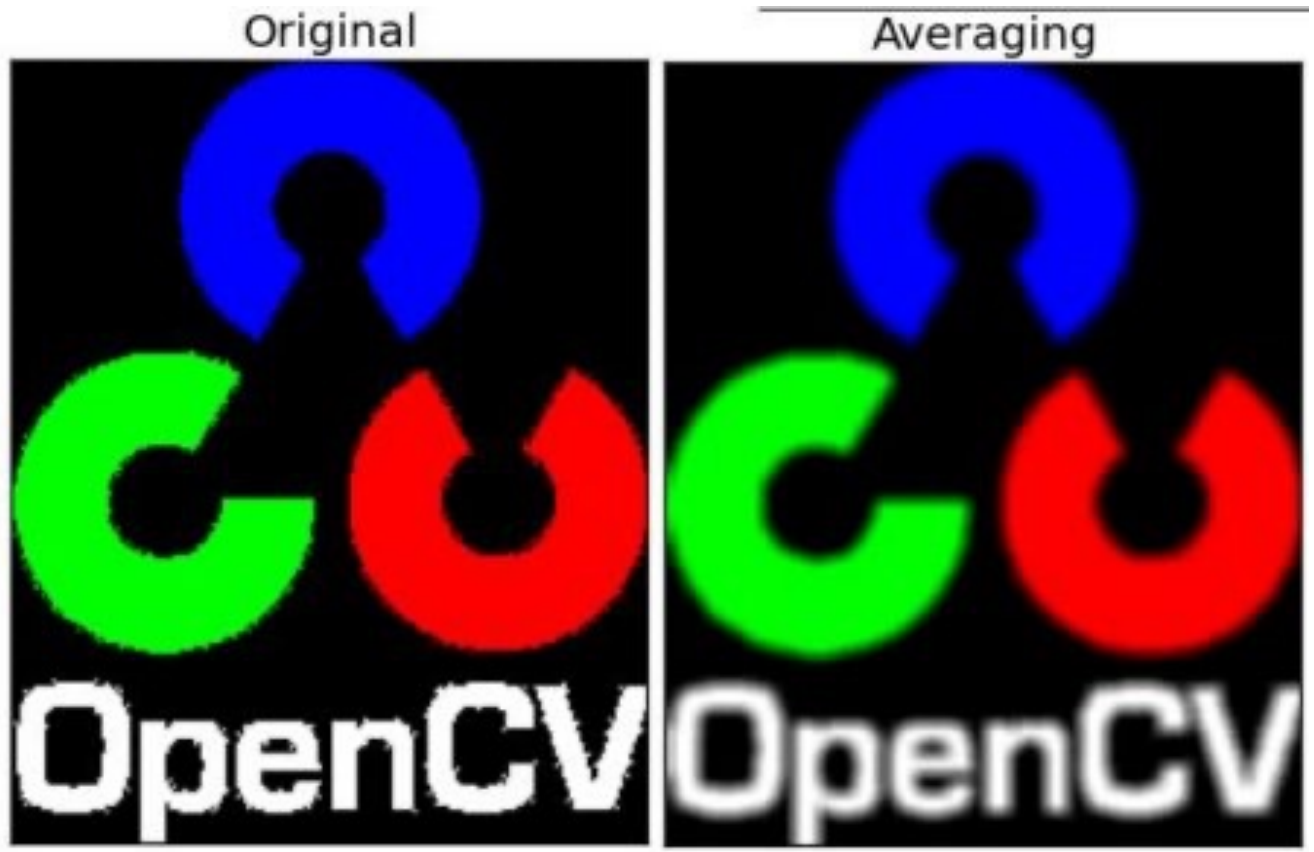
1. Digital Image Acquisition
2. Image Filtering
3. Supervised Learning
4. Multi-layer Perceptrons
5. Convolutional Neural Nets
6. Visualizing Learned Filters
7. Pooling and FC Layers
8. Applications in Vision
9. Object Classification
10. Applications in Robotics



2. Image Filtering

- Image filtering: compute function of local neighborhood at each position

- Very important!
  - Enhance images
    - Denoise, resize, increase contrast, etc.
  - Extract information from images
    - Texture, edges, distinctive points, etc.
  - Detect patterns
    - Template matching
  - Deep Convolutional Networks



Example: box filter

$$g[\cdot,\cdot]$$

$$\frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

Slide credit: David Lowe (UBC)

Image filtering

$g[\cdot,\cdot]$ $\frac{1}{9}$

1	1	1
1	1	1
1	1	1

$f[.,.]$

0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	0	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	0	0	0	0	0	0	0
0	0	90	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

$h[.,.]$

$$h[m,n] = \sum_{k,l} g[k,l]\, f[m+k,n+l]$$

Credit: S. Seitz

Image filtering

$g[\cdot,\cdot]$ $\frac{1}{9}$

1	1	1
1	1	1
1	1	1

$f[\cdot,\cdot]$

0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	0	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	0	0	0	0	0	0	0
0	0	90	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

$h[\cdot,\cdot]$

	0	10							

$$h[m,n] = \sum_{k,l} g[k,l]\, f[m+k,n+l]$$

Credit: S. Seitz

Image filtering

$g[\cdot,\cdot]$ $\frac{1}{9}$

1	1	1
1	1	1
1	1	1

$f[.,.]$

0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	0	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	0	0	0	0	0	0	0
0	0	90	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

$h[.,.]$

	0	10	20						

$$h[m,n] = \sum_{k,l} g[k,l]\, f[m+k,n+l]$$

Credit: S. Seitz

Image filtering

$g[\cdot,\cdot]$ $\frac{1}{9}$

1	1	1
1	1	1
1	1	1

$f[\cdot,\cdot]$

0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	0	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	0	0	0	0	0	0	0
0	0	90	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

$h[\cdot,\cdot]$

	0	10	20	30					

$$h[m,n] = \sum_{k,l} g[k,l]\, f[m+k,n+l]$$

Credit: S. Seitz

Image filtering

$g[\cdot,\cdot]$ $\frac{1}{9}$

1	1	1
1	1	1
1	1	1

$f[\cdot,\cdot]$

0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	0	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	0	0	0	0	0	0	0
0	0	90	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

$h[\cdot,\cdot]$

	0	10	20	30	30				

$$h[m,n] = \sum_{k,l} g[k,l]\, f[m+k,n+l]$$

Credit: S. Seitz

Image filtering

$g[\cdot,\cdot]$ $\frac{1}{9}$

1	1	1
1	1	1
1	1	1

$f[.,.]$

0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	0	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	0	0	0	0	0	0	0
0	0	90	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

$h[.,.]$

	0	10	20	30	30				
				?					

$$h[m,n] = \sum_{k,l} g[k,l]\, f[m+k,n+l]$$

Credit: S. Seitz

Image filtering

$g[\cdot,\cdot]$ $\frac{1}{9}$

1	1	1
1	1	1
1	1	1

$f[\cdot,\cdot]$

0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	0	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	0	0	0	0	0	0	0
0	0	90	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

$h[\cdot,\cdot]$

	0	10	20	30	30				
						?			
				50					

$$h[m,n] = \sum_{k,l} g[k,l]\, f[m+k,n+l]$$

Credit: S. Seitz

Image filtering

$g[\cdot,\cdot]$ $\frac{1}{9}$

1	1	1
1	1	1
1	1	1

$f[\cdot,\cdot]$

0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	0	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	0	0	0	0	0	0	0
0	0	90	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

$h[\cdot,\cdot]$

	0	10	20	30	30	30	20	10	
	0	20	40	60	60	60	40	20	
	0	30	60	90	90	90	60	30	
	0	30	50	80	80	90	60	30	
	0	30	50	80	80	90	60	30	
	0	20	30	50	50	60	40	20	
	10	20	30	30	30	30	20	10	
	10	10	10	0	0	0	0	0	

$$h[m,n] = \sum_{k,l} g[k,l]\, f[m+k,n+l]$$

Credit: S. Seitz

Box Filter

What does it do?

- Replaces each pixel with an average of its neighborhood

- Achieve smoothing effect (remove sharp features)

$g[\cdot,\cdot]$

$$\frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

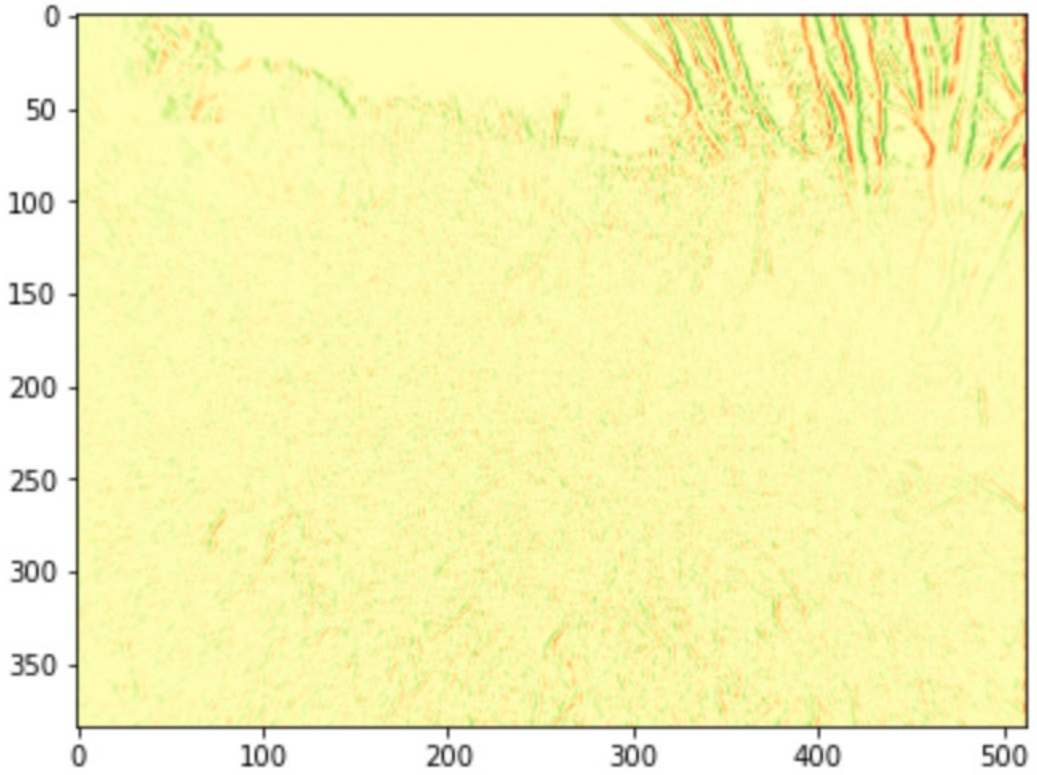
Slide credit: David Lowe (UBC)

Smoothing with box filter



Sobel filter [-1 0 1]: horizontal gradient

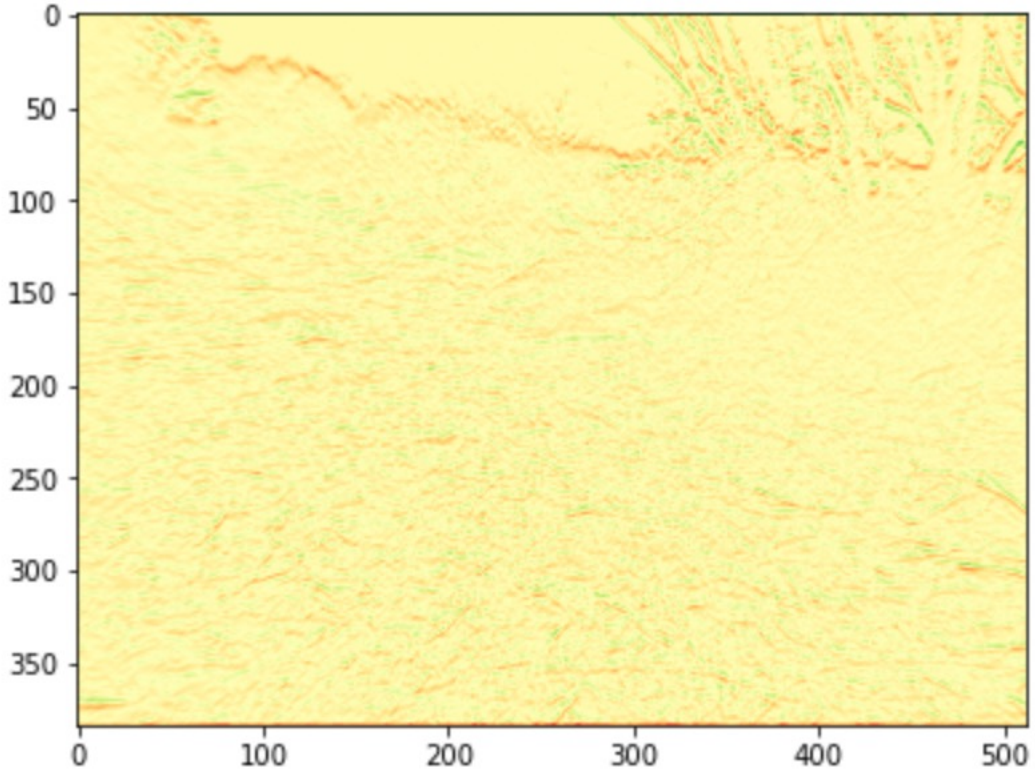





```
sobel_u = torch.tensor([[-1, 0, 1]], dtype=float)
I_u = diffdrive.conv2(grayscale, sobel_u)
```

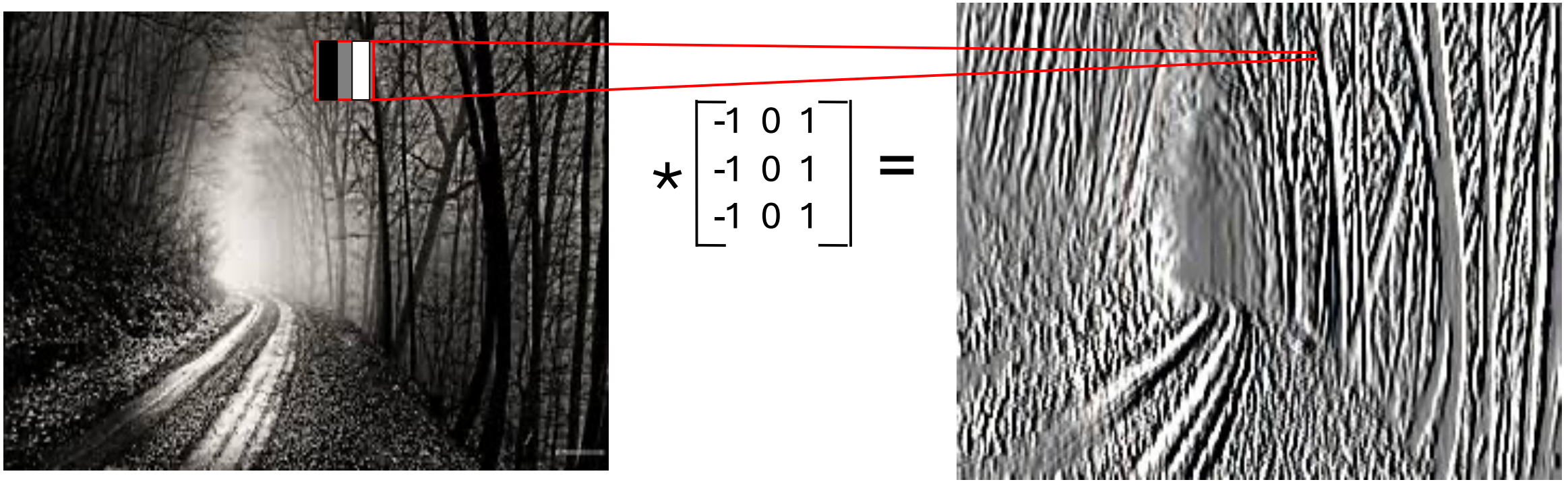
Sobel filter $[-1\ 0\ 1]^T$: vertical gradient






```
sobel_v = torch.tensor([[-1], [0], [1]], dtype=float)
I_v = diffdrive.conv2(grayscale, sobel_v)
```

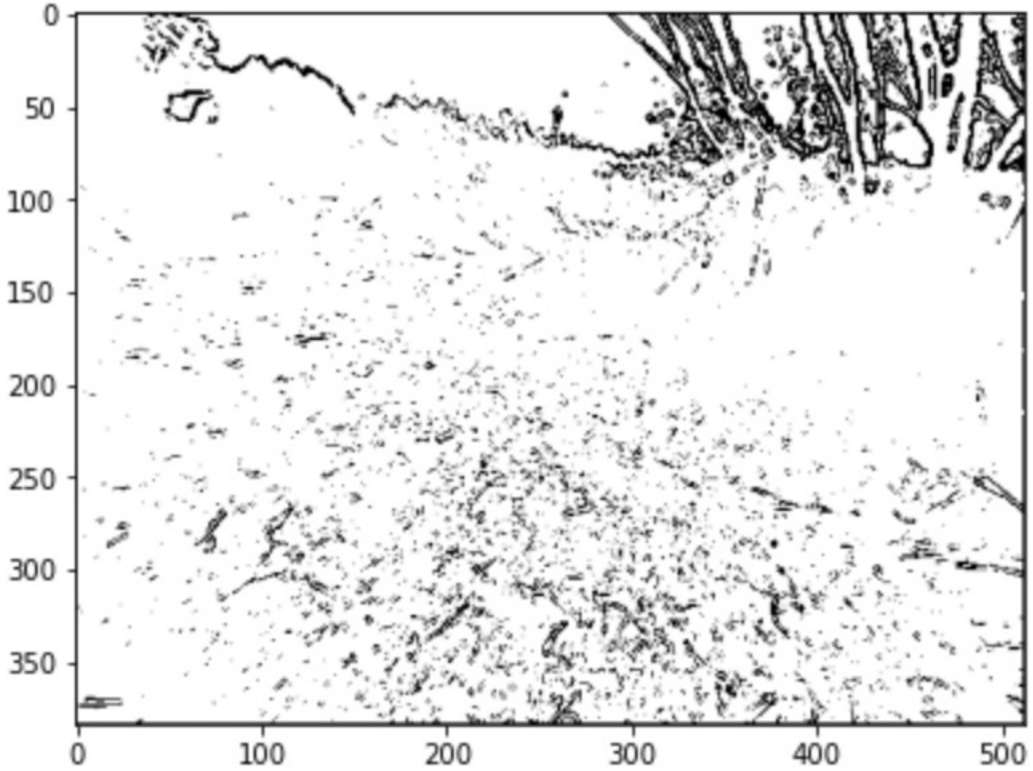
Sobel and box-filter can be combined to yield edge-like “features”!



Slide Credit: Marc'Aurelio Ranzato

However! Edge Detection is more than Sobel Gradients: it is a per-pixel classification problem

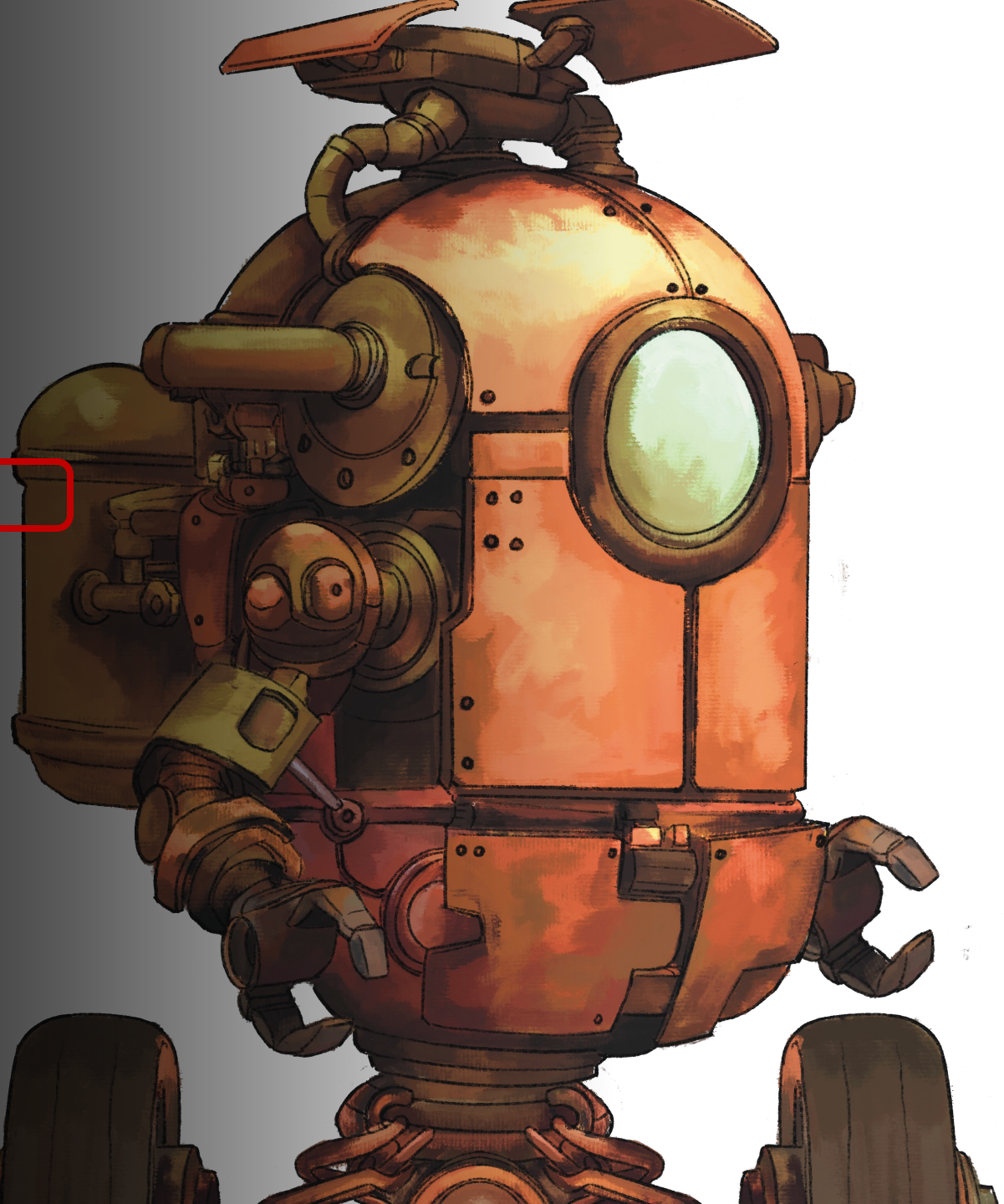





```
I_m = torch.sqrt(torch.square(I_u)+torch.square(I_v))
edges = torch.threshold(I_m,50,0)>0
```

Outline

1. Digital Image Acquisition
2. Image Filtering
3. Supervised Learning
4. Multi-layer Perceptrons
5. Convolutional Neural Nets
6. Visualizing Learned Filters
7. Pooling and FC Layers
8. Applications in Vision
9. Object Classification
10. Applications in Robotics



3. Supervised Learning

- Example: classification



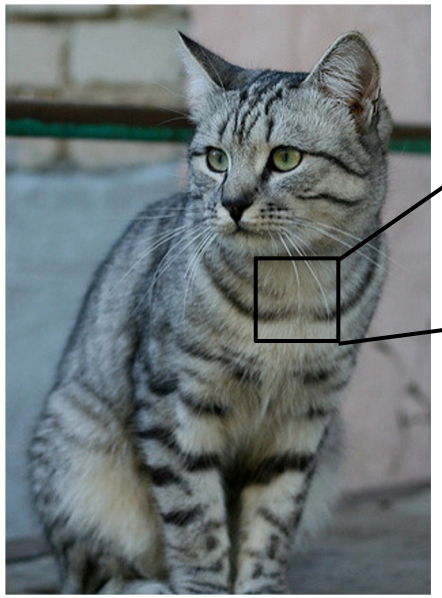
This image by Nikita is licensed under CC-BY 2.0

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

cat

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

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)

An image (or pixel) classifier

```
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.

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

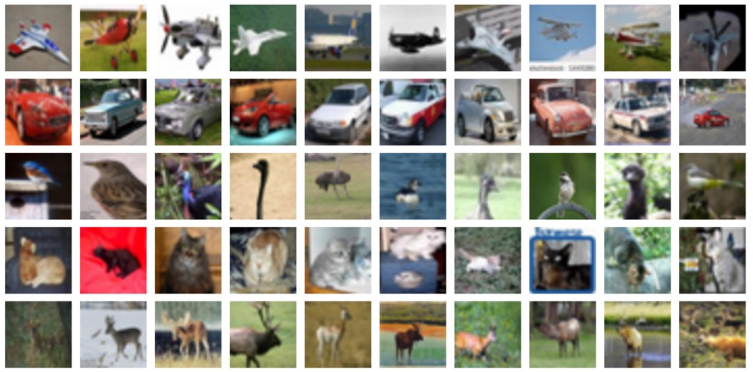
ML: A Data-Driven Approach

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
```

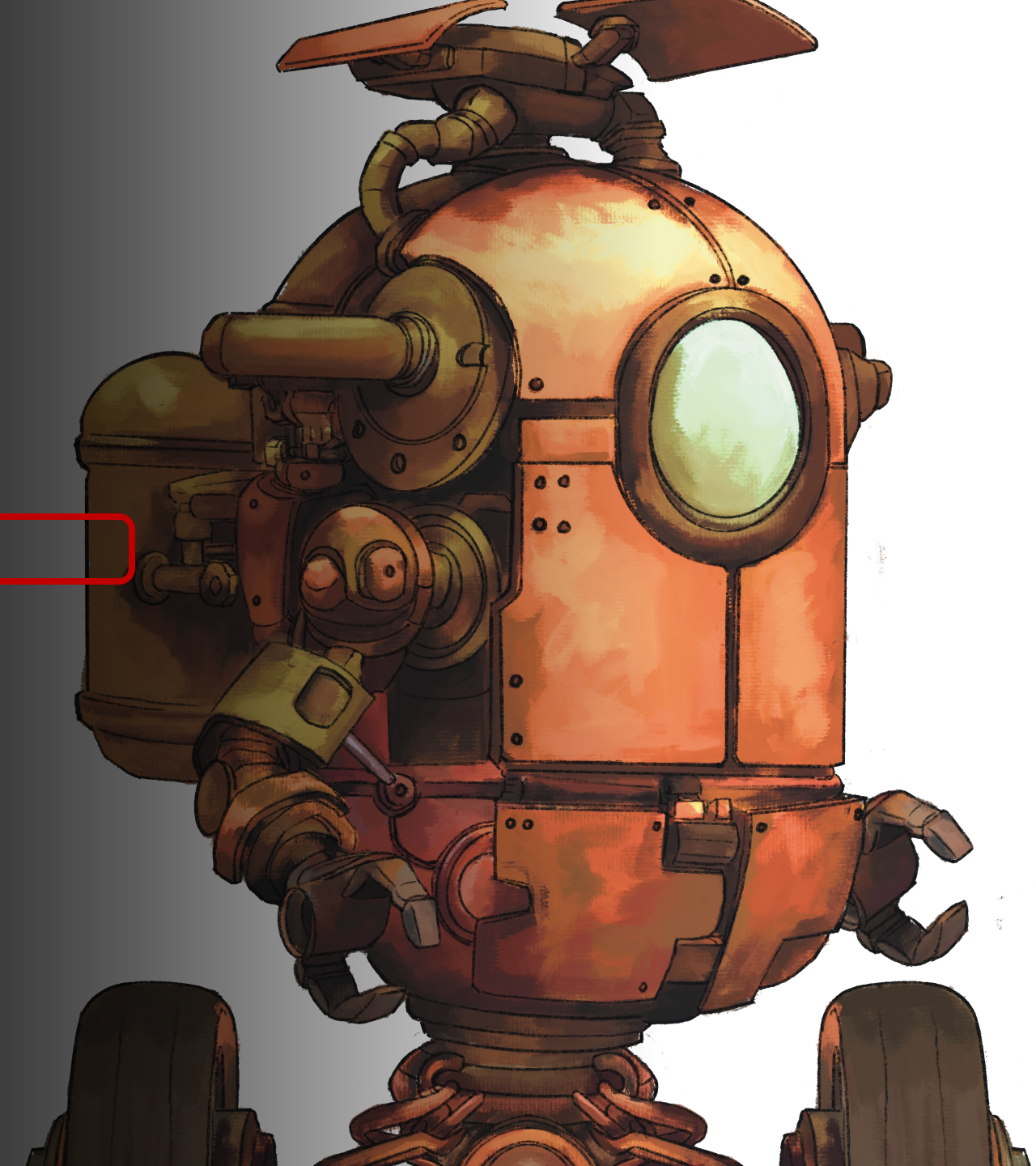
Example training set



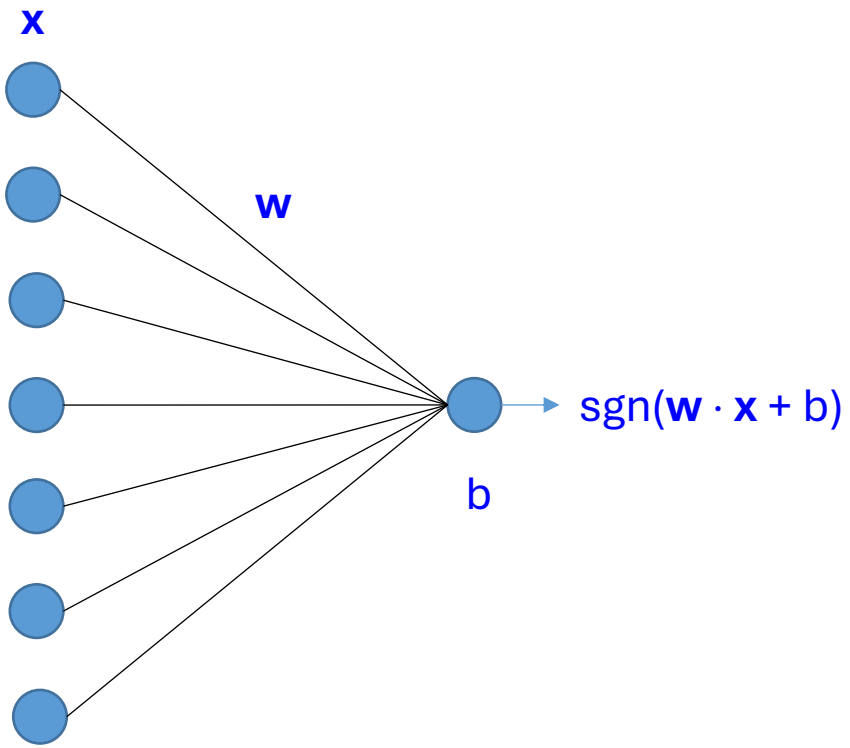
Slide Credit: Fei-Fei Li, Justin Johnson, Serena Yeung, CS 231n

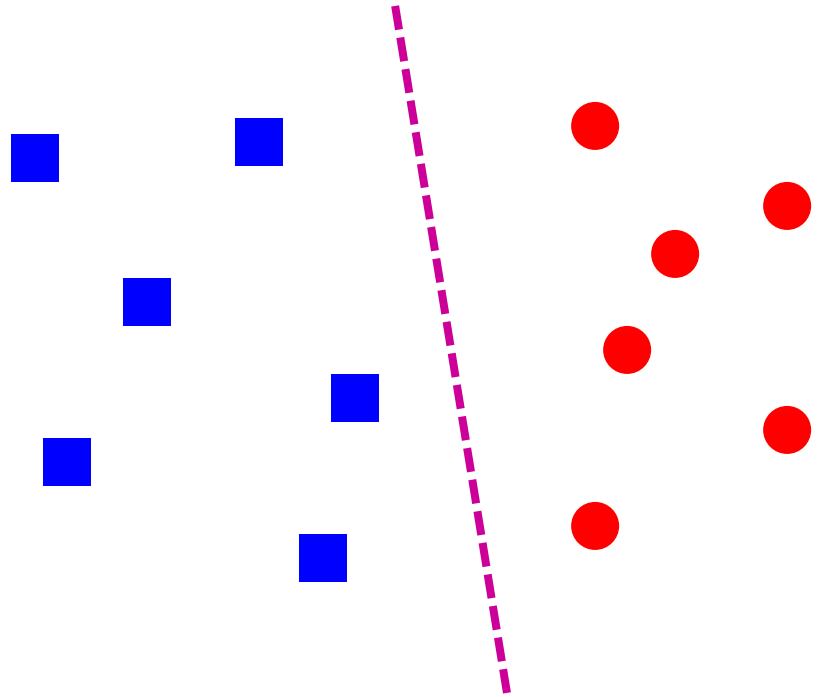
Outline

1. Digital Image Acquisition
2. Image Filtering
3. Supervised Learning
4. Multi-layer Perceptrons
5. Convolutional Neural Nets
6. Visualizing Learned Filters
7. Pooling and FC Layers
8. Applications in Vision
9. Object Classification
10. Applications in Robotics



4. Linear classifiers aka Perceptrons

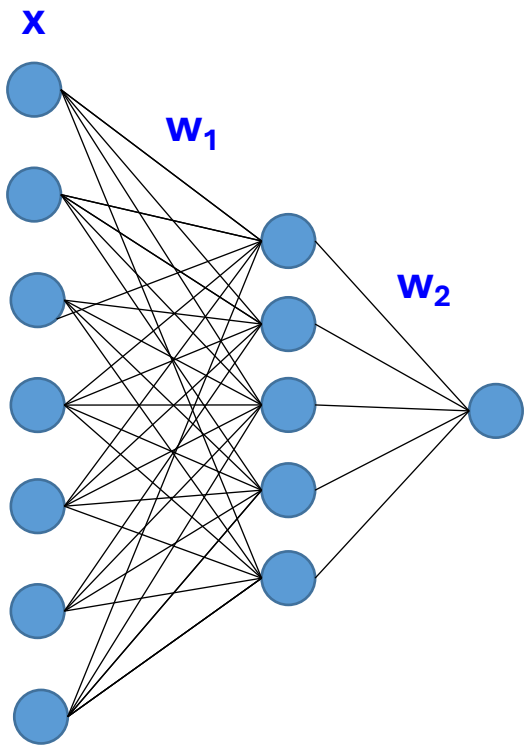


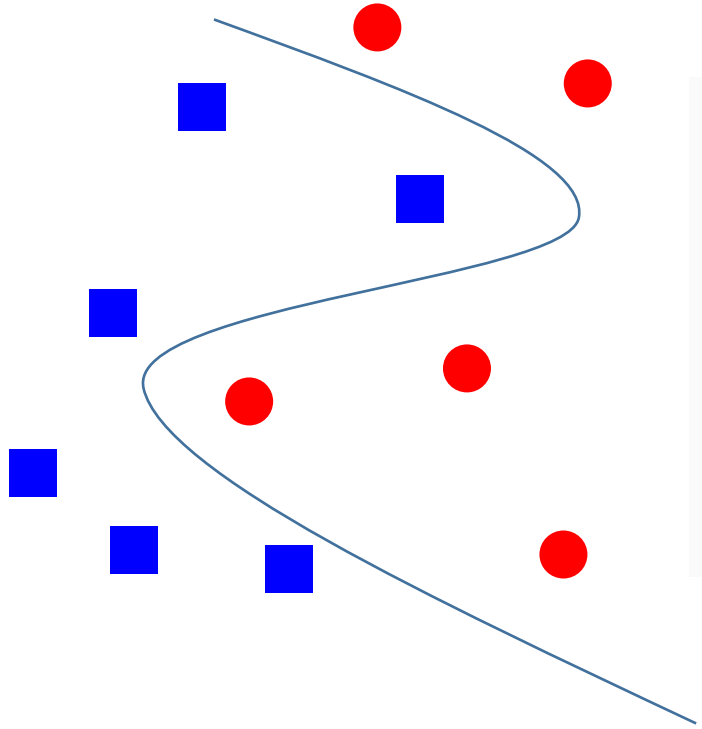


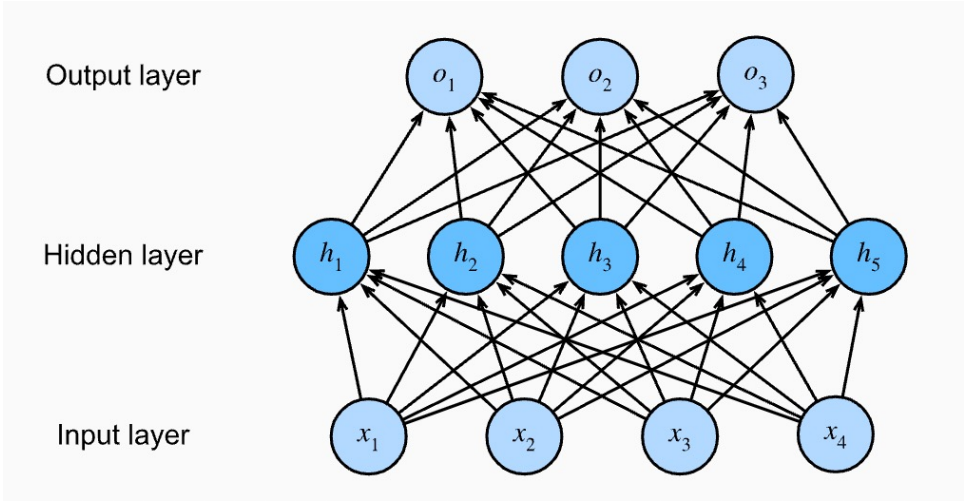
- Find a *linear function* to separate the classes:

$$f(\mathbf{x}) = \mathrm{sgn}(\mathbf{w} \cdot \mathbf{x} + b)$$

Multi-Layer Perceptrons (MLP)





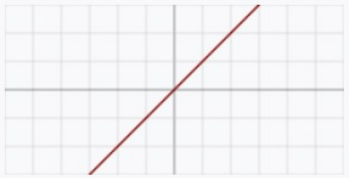
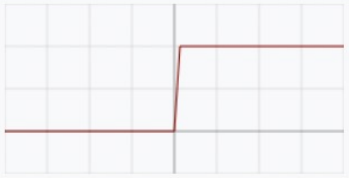
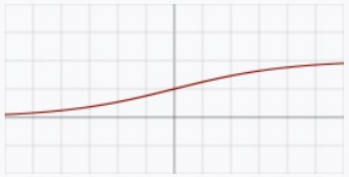
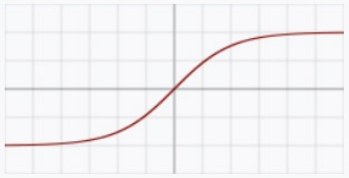
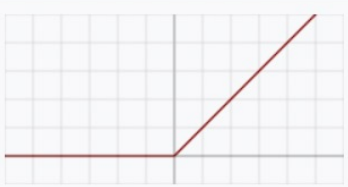


DIVE INTO DEEP LEARNING

- Find a *nonlinear function* to separate the classes:

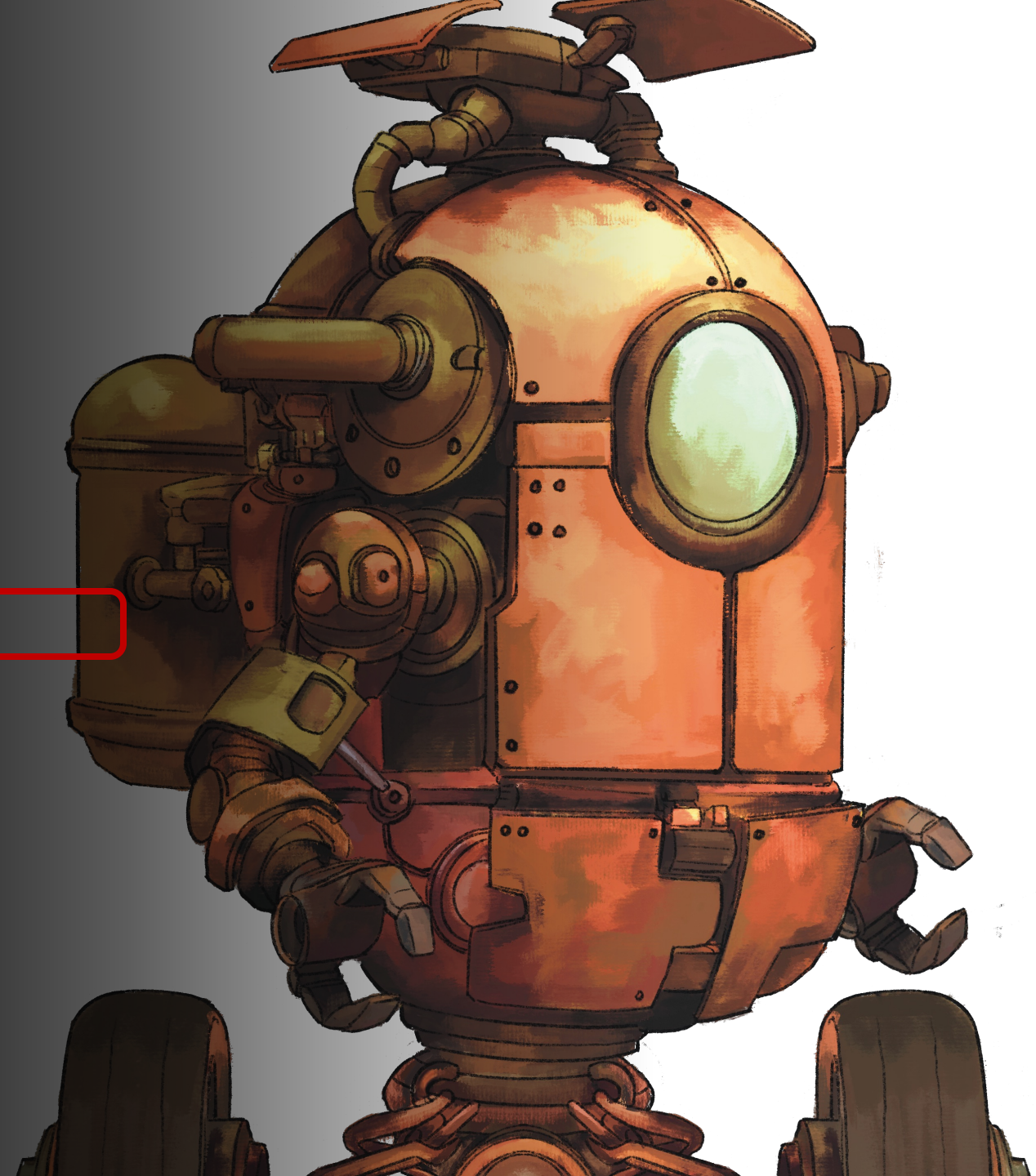
$$f(\mathbf{x}) = \mathrm{sgn}(\mathbf{w_2} \cdot \mathrm{sgn}(\mathbf{w_1} \cdot \mathbf{x} + b))$$

Activation Functions

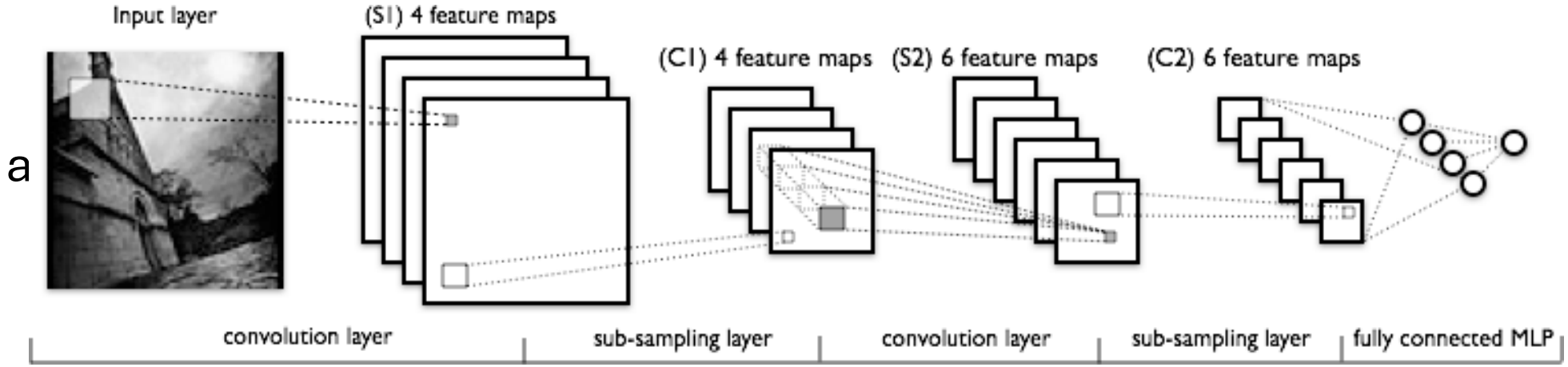
Name	Plot	Function, $g(x)$
Identity		x
Binary step		$\begin{cases} 0 & \text{if } x < 0 \\ 1 & \text{if } x \geq 0 \end{cases}$
Logistic, sigmoid, or soft step		$\sigma(x) = \frac{1}{1 + e^{-x}}$
Hyperbolic tangent (tanh)		$\tanh(x) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}}$
Rectified linear unit (ReLU)[7]		$\begin{cases} 0 & \text{if } x \leq 0 \\ x & \text{if } x > 0 \end{cases}$ $= \max\{0, x\} = x\mathbf{1}_{x>0}$

Outline

1. Digital Image Acquisition
2. Image Filtering
3. Supervised Learning
4. Multi-layer Perceptrons
5. Convolutional Neural Nets
6. Visualizing Learned Filters
7. Pooling and FC Layers
8. Applications in Vision
9. Object Classification
10. Applications in Robotics



5. Convolutional Neural Networks



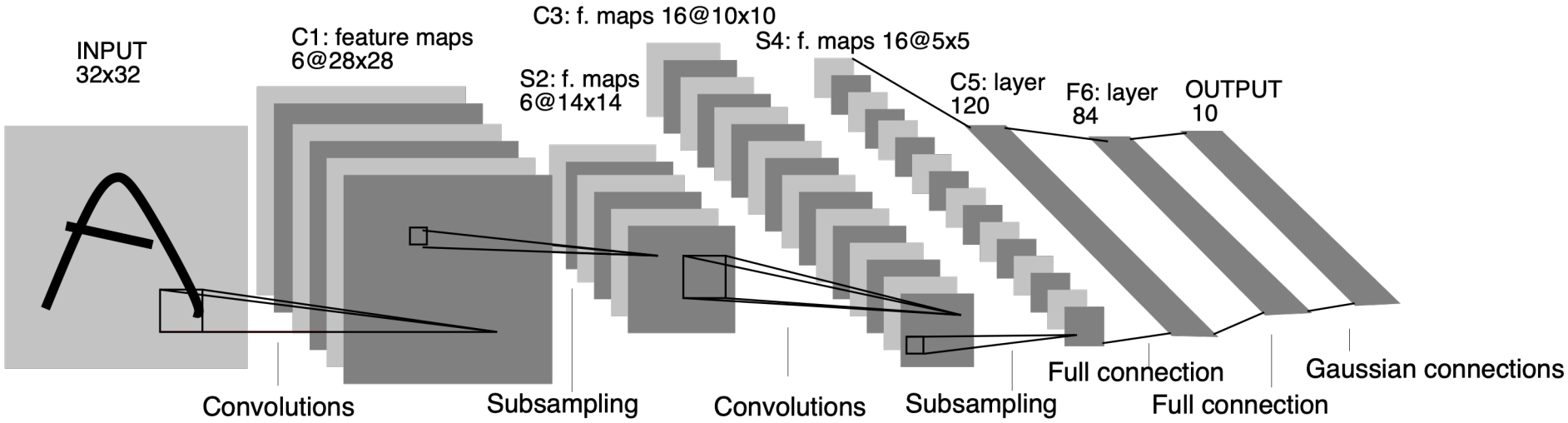
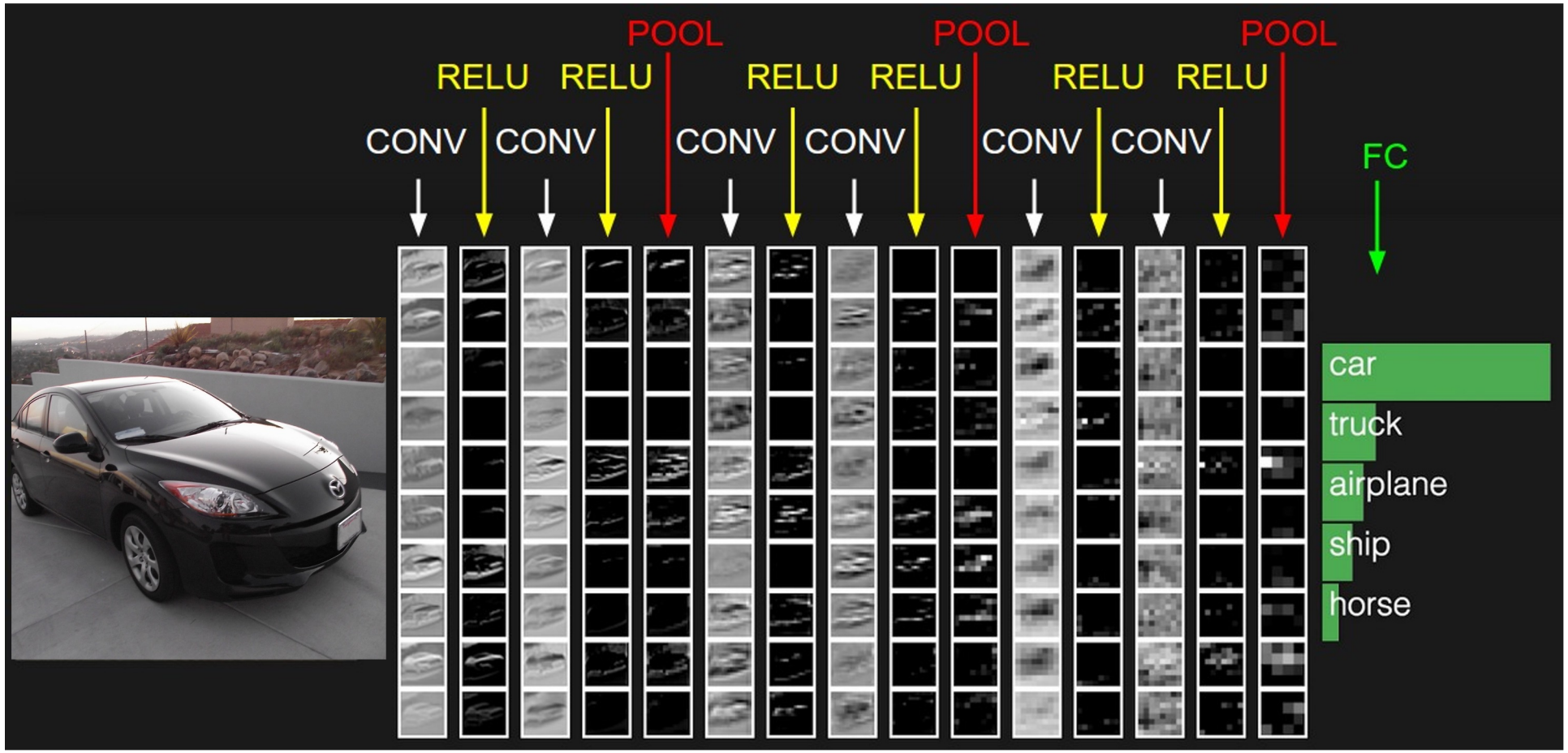


Image Credit: Yann LeCun, Kevin Murphy

preview:

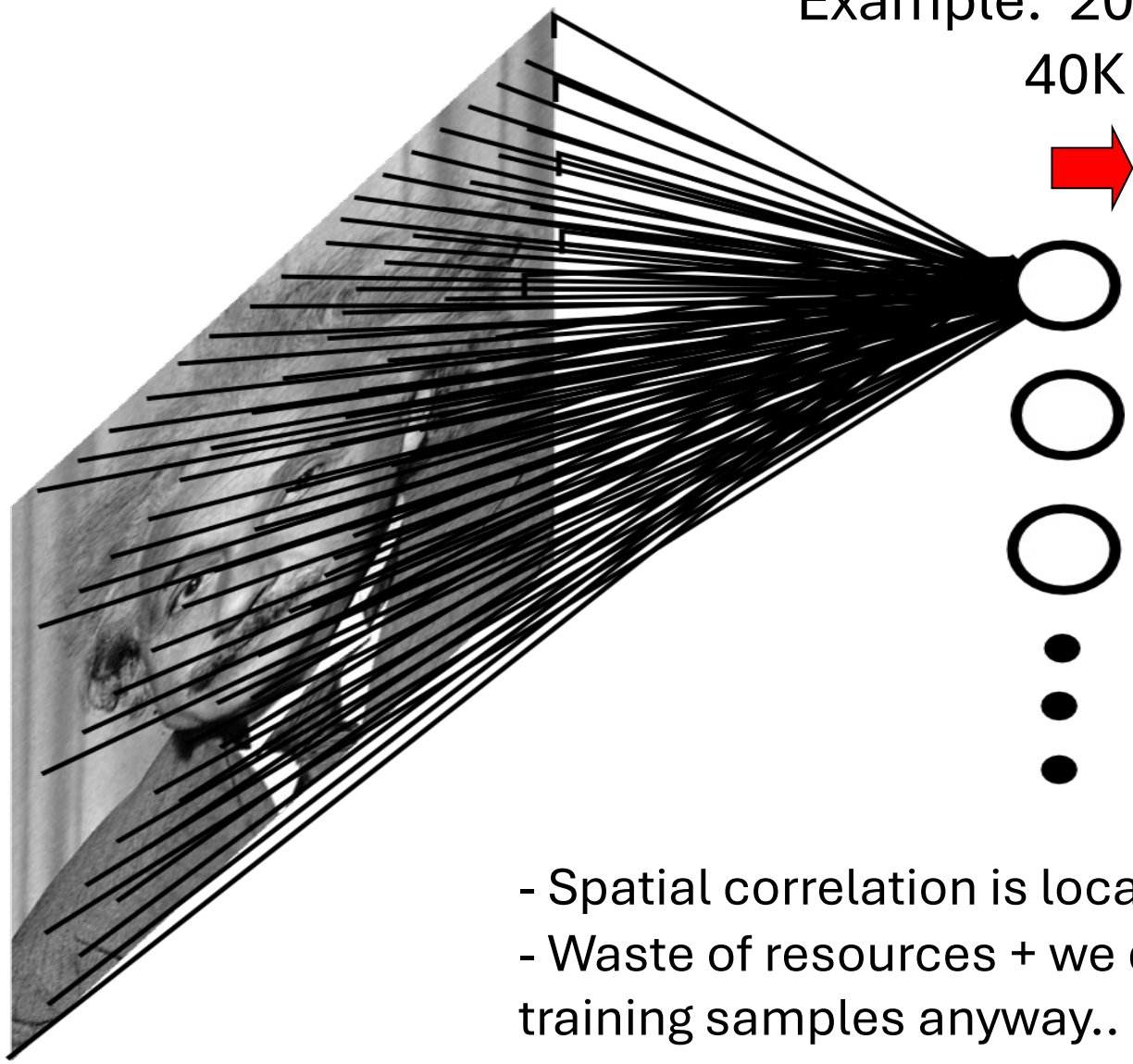


Slide Credit: Fei-Fei Li, Justin Johnson, Serena Yeung, CS 231n

Fully Connected Layer

Example: 200x200 image
40K hidden units

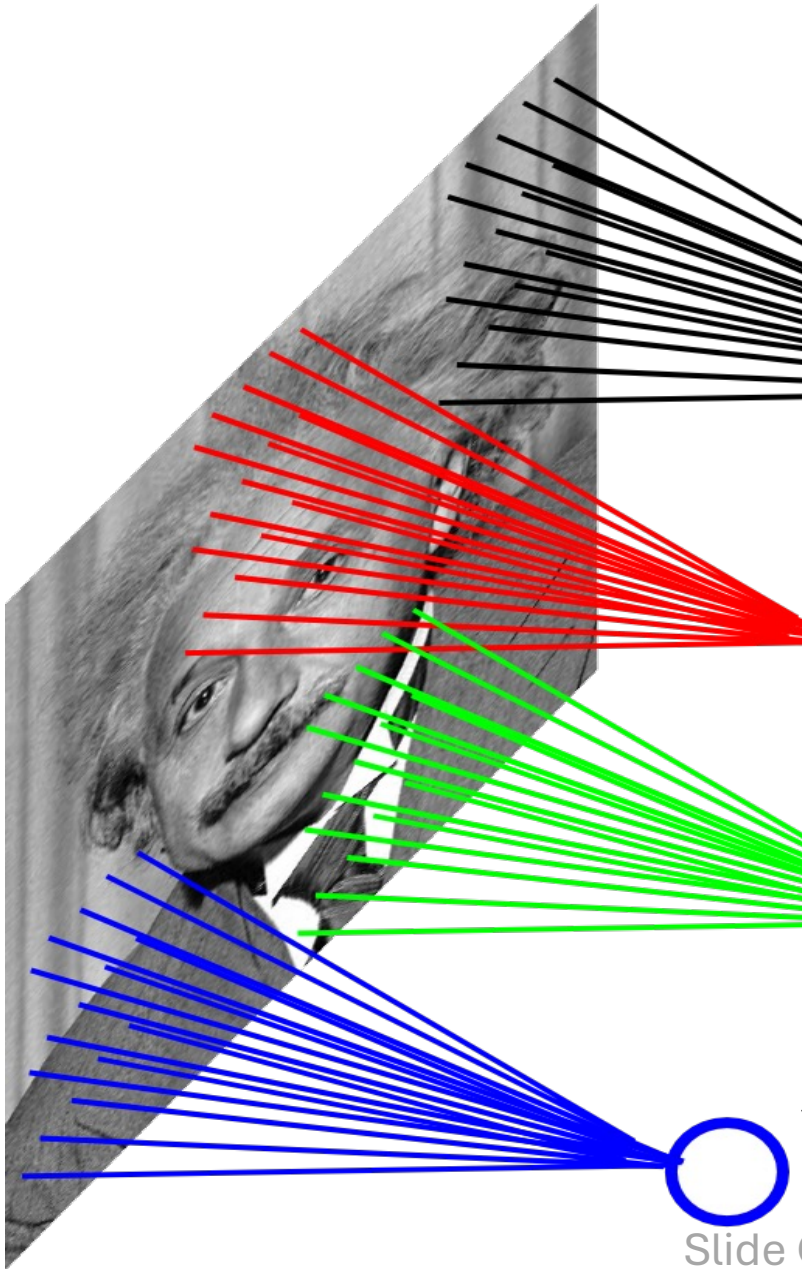
~1.6B parameters!!!



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

Slide Credit: Marc'Aurelio Ranzato

Locally Connected Layer

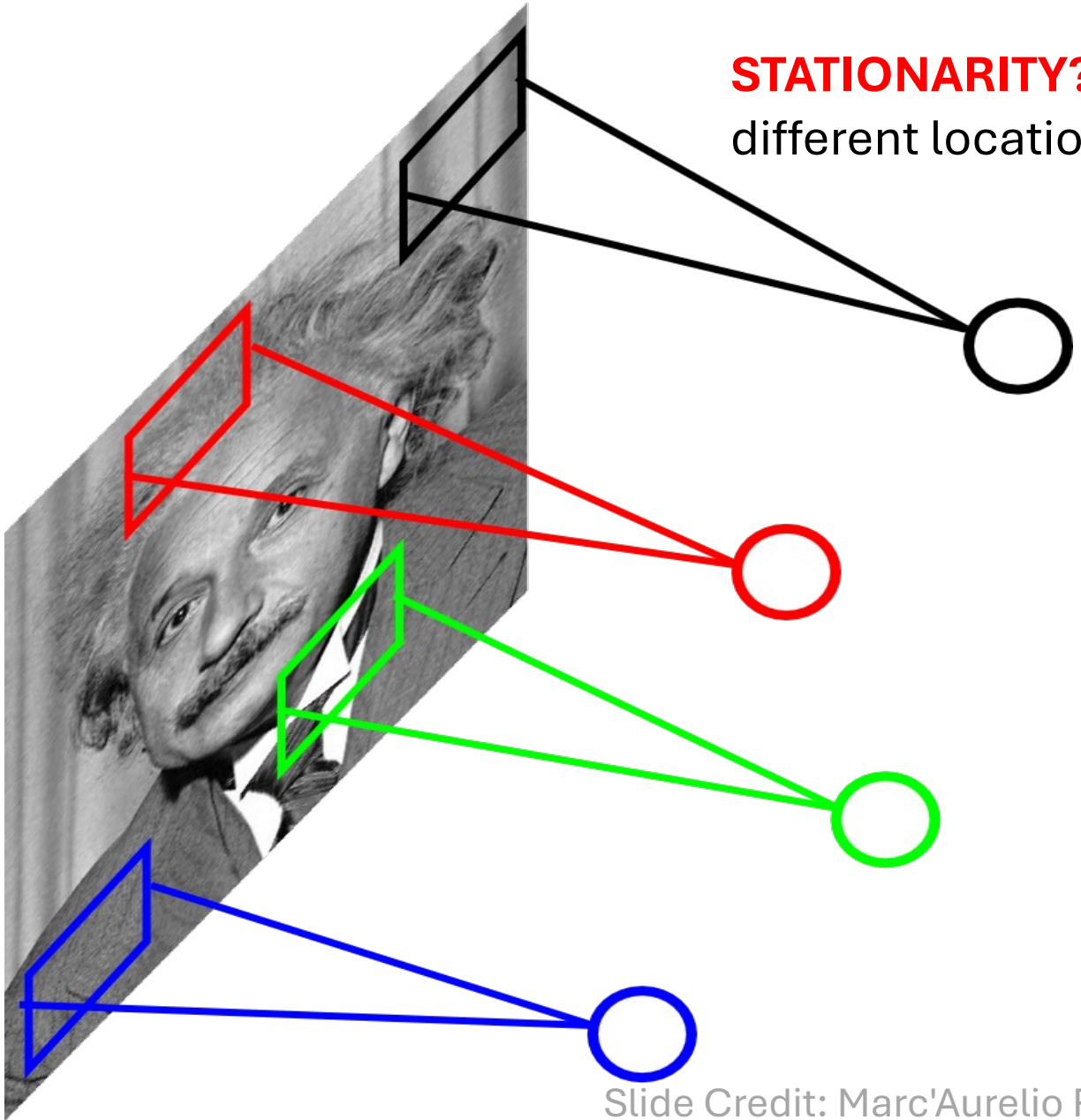


Example: 200x200 image
40K hidden units
Filter size: 10x10
4M parameters

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

Slide Credit: Marc'Aurelio Ranzato

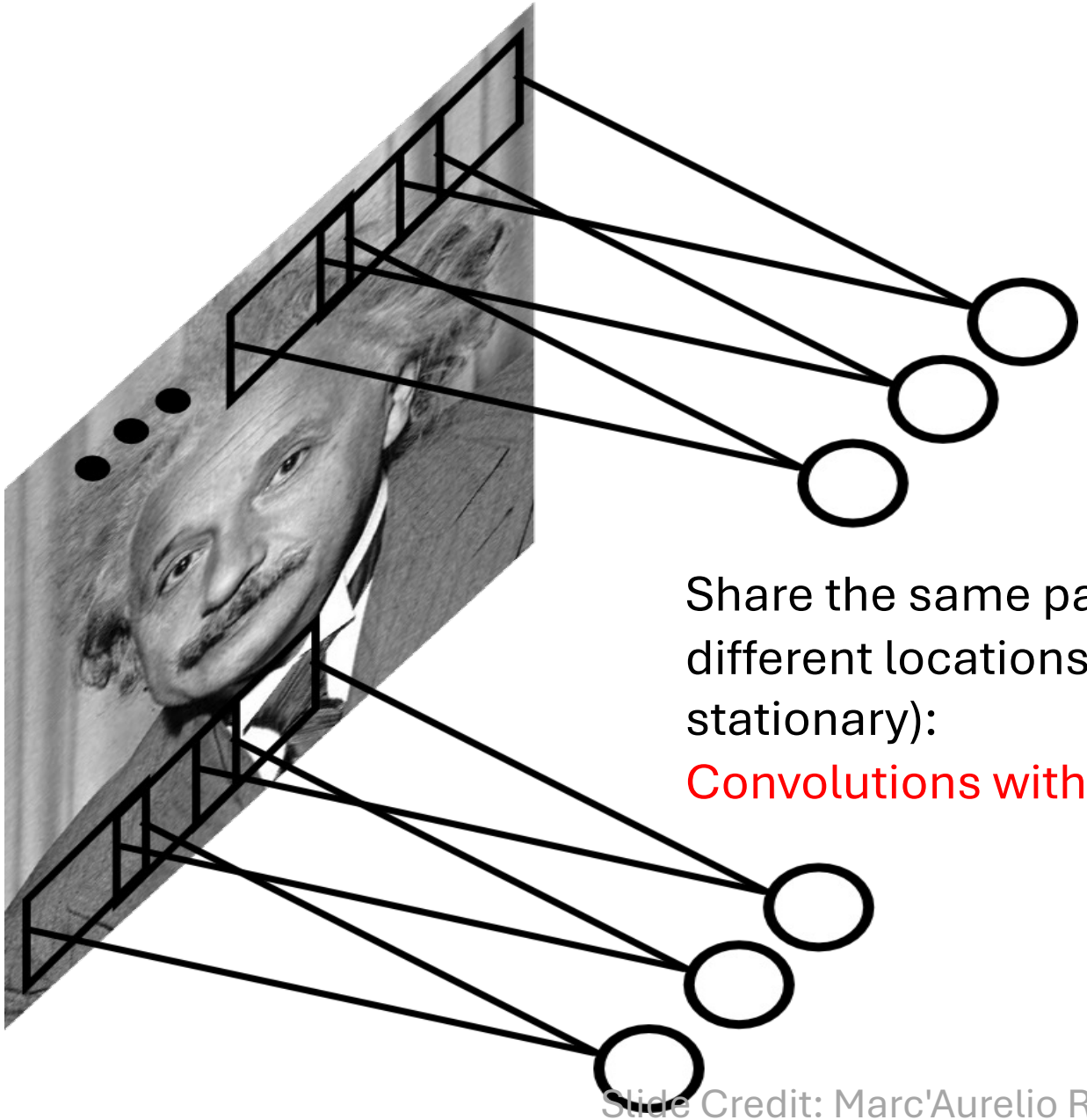
Locally Connected Layer



STATIONARITY? Statistics is similar at different locations

Slide Credit: Marc'Aurelio Ranzato

Convolutional Layer

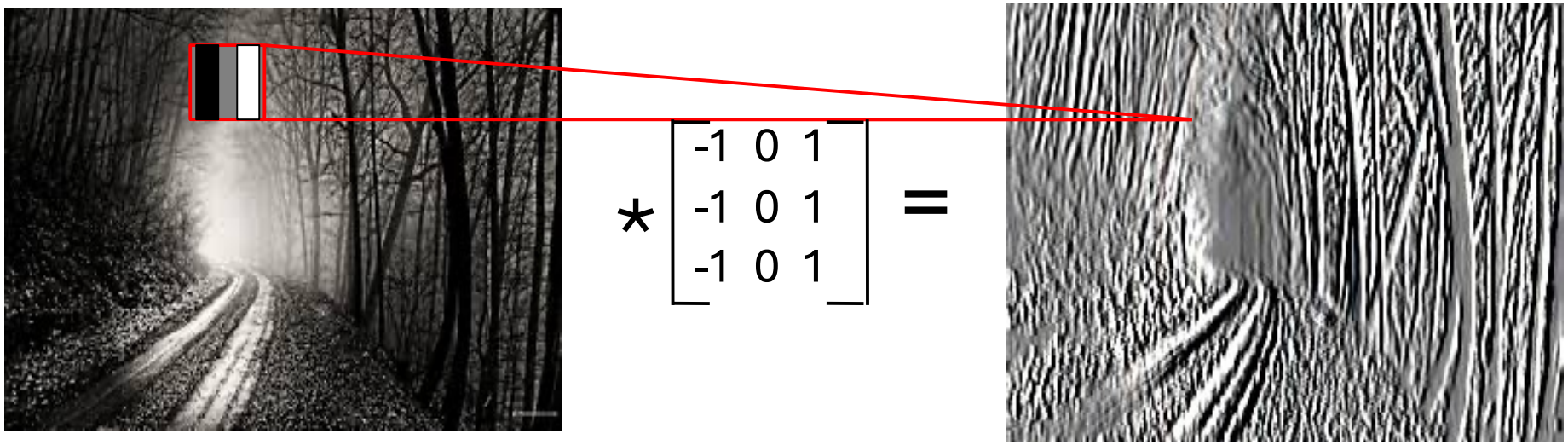


Share the same parameters across different locations (assuming input is stationary):

Convolutions with learned kernels

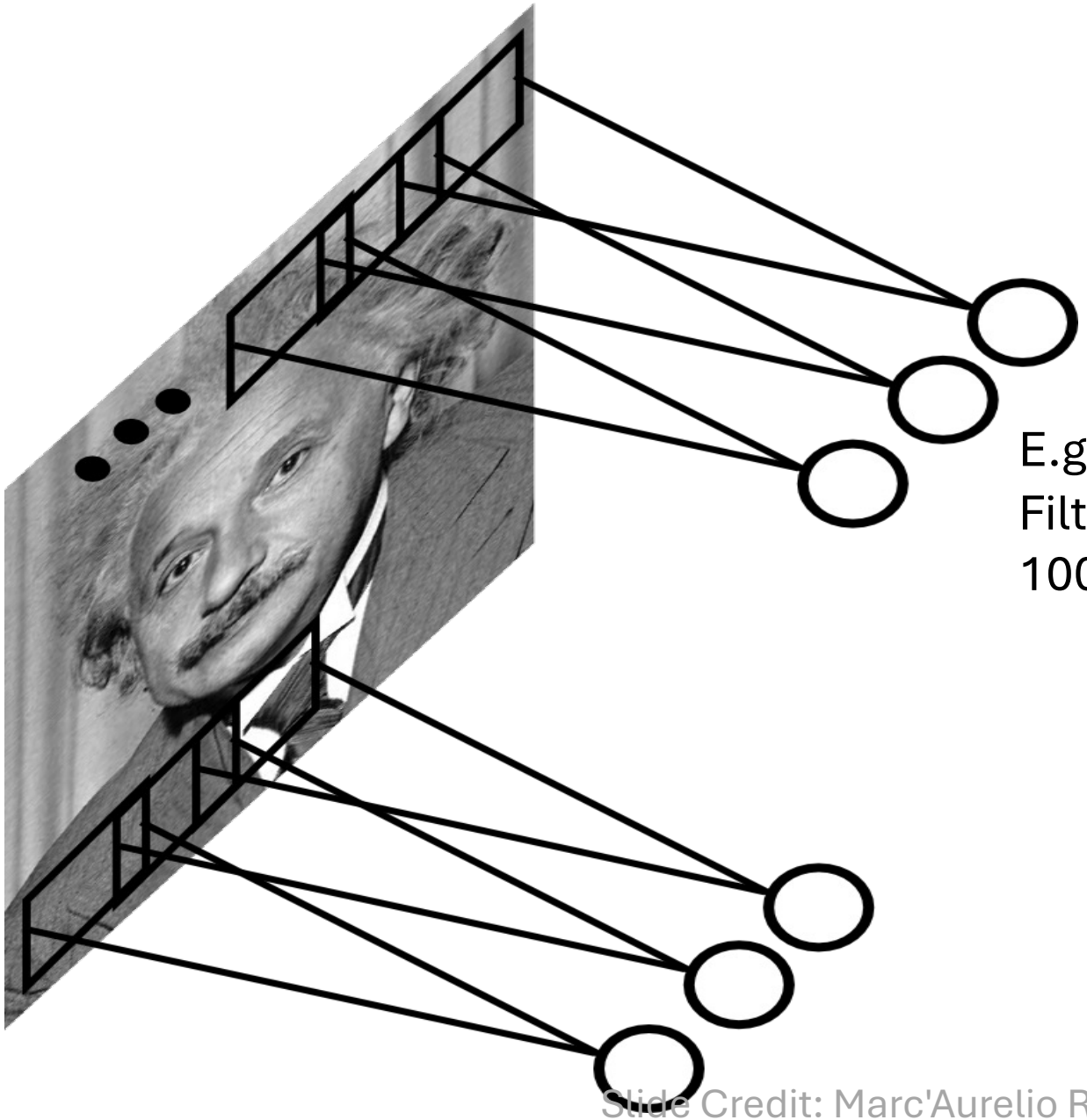
Slide Credit: Marc'Aurelio Ranzato

Convolutional Layer



Slide Credit: Marc'Aurelio Ranzato

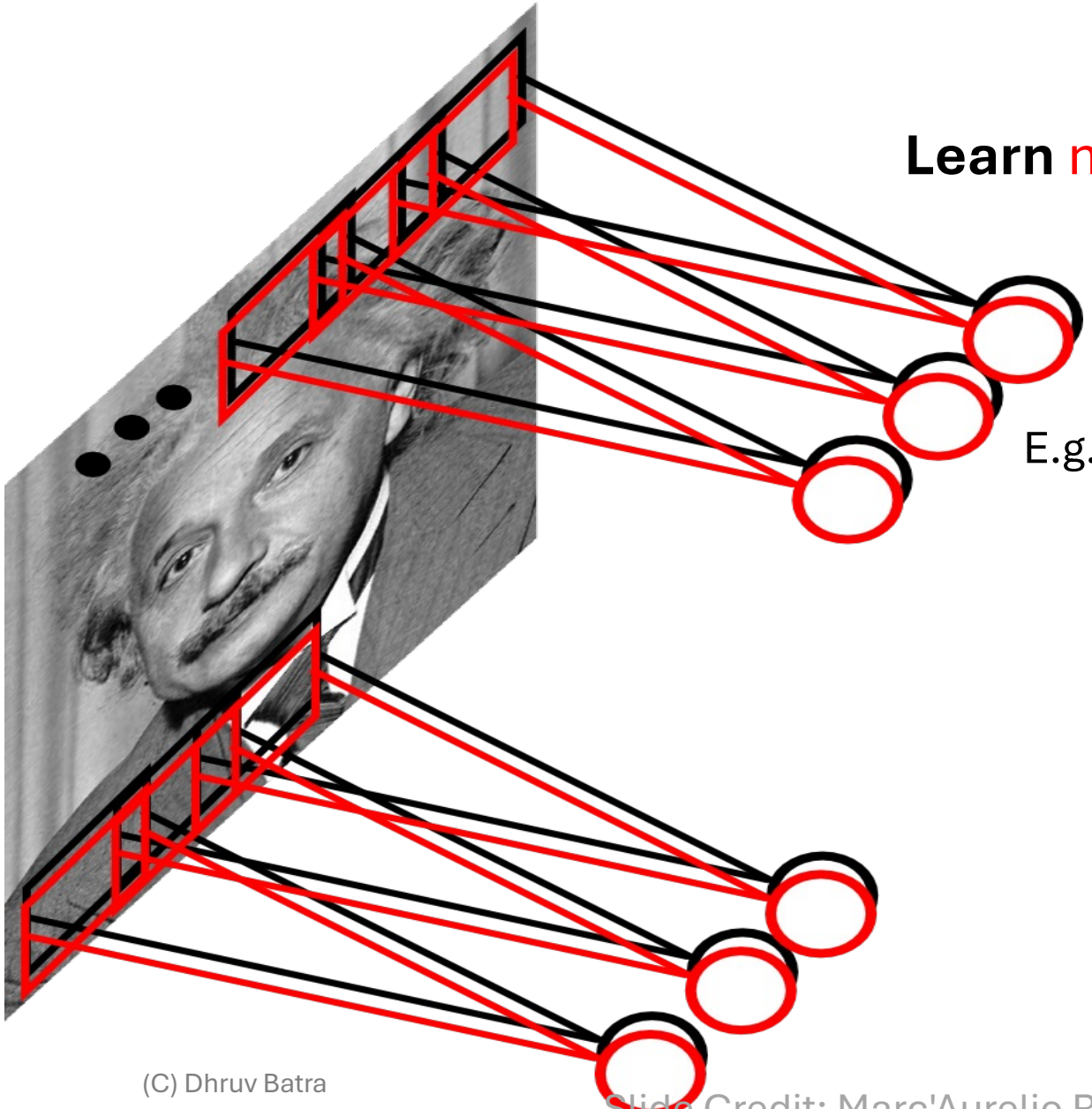
Convolutional Layer



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

Slide Credit: Marc'Aurelio Ranzato

Multi-channel Convolutional Layer



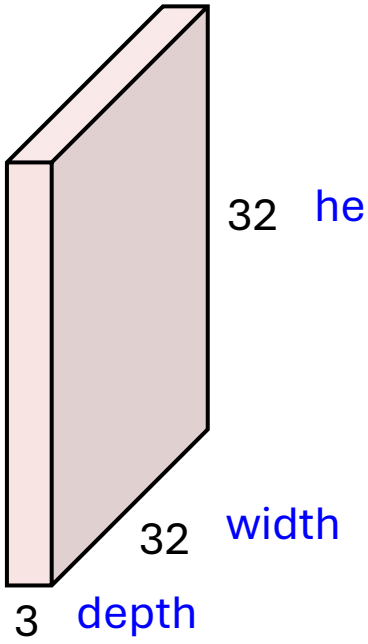
Learn multiple filters.

E.g.: 200x200 image
50 Filters
Filter size: 10x10
5K parameters: still tiny << 1.6B

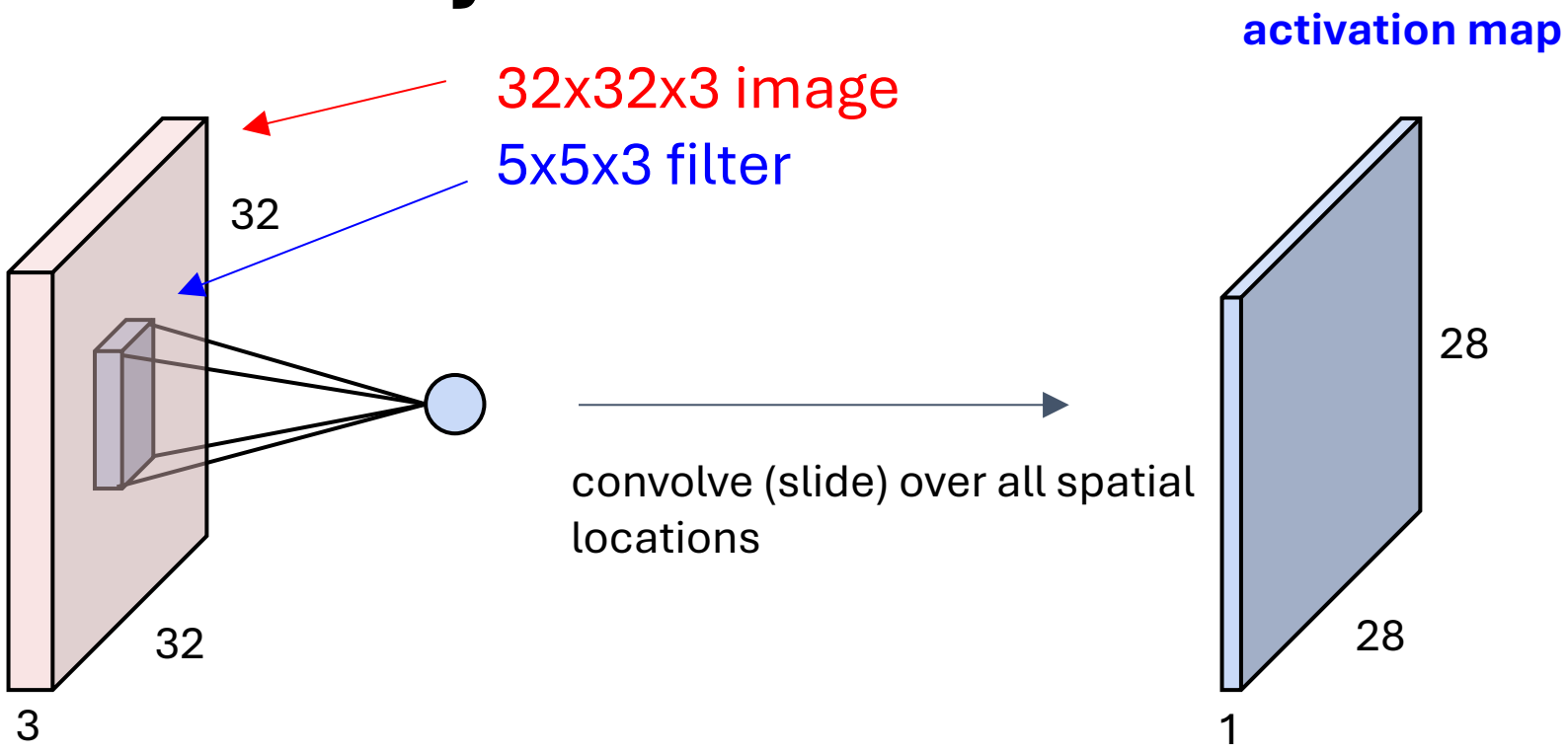
Slide Credit: Marc'Aurelio Ranzato

Convolution Layer

32x32x3 image -> preserve spatial structure



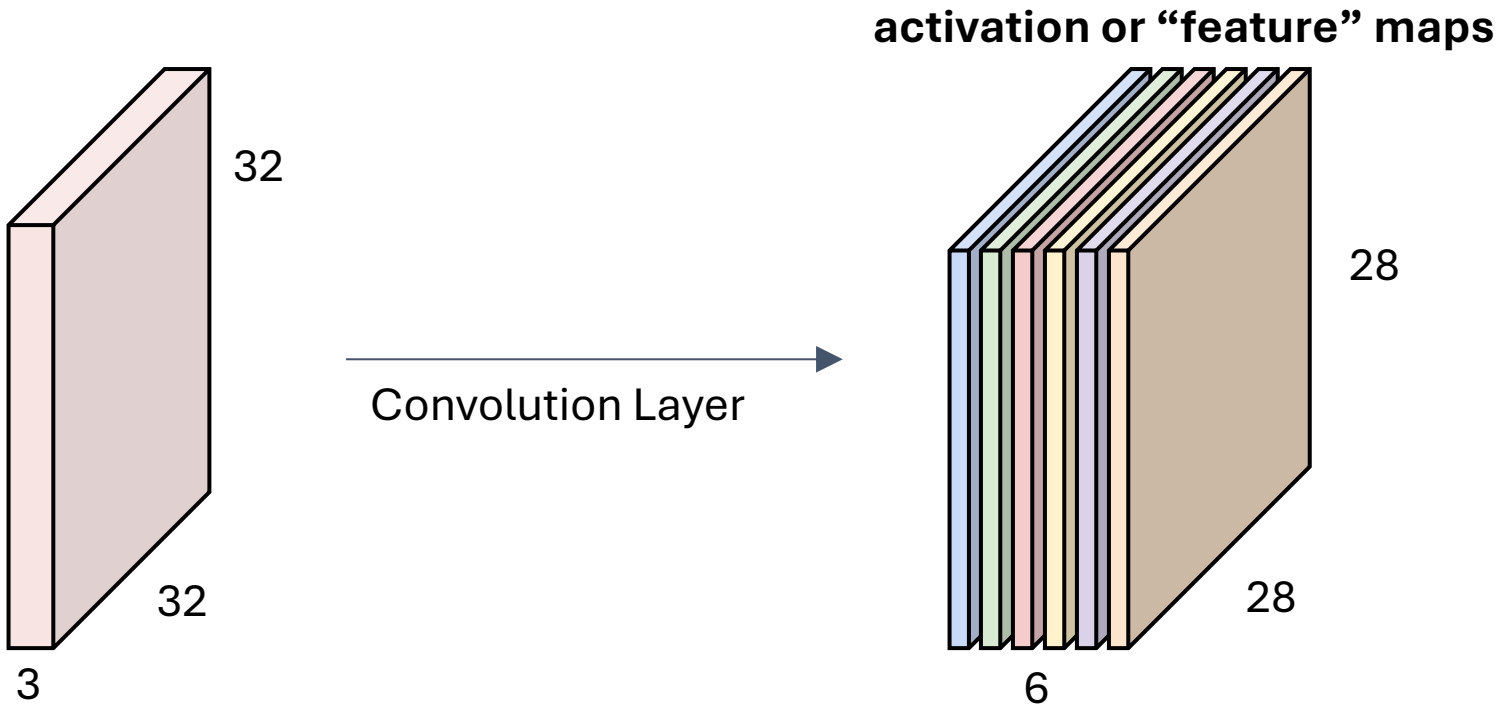
Convolution Layer



Number of parameters here?

5x5x3 = 75 !

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

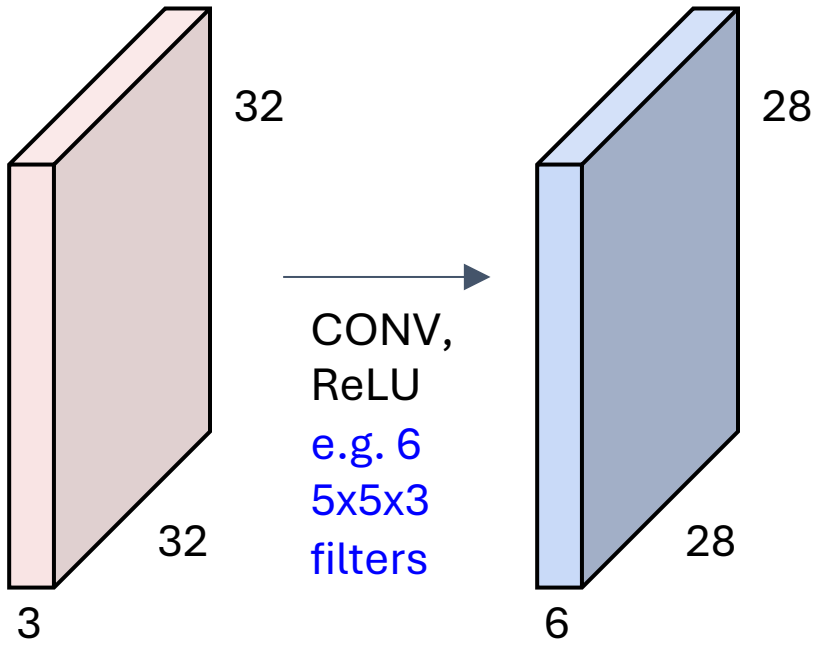


We stack these up to get a "new image" of size 28x28x6!

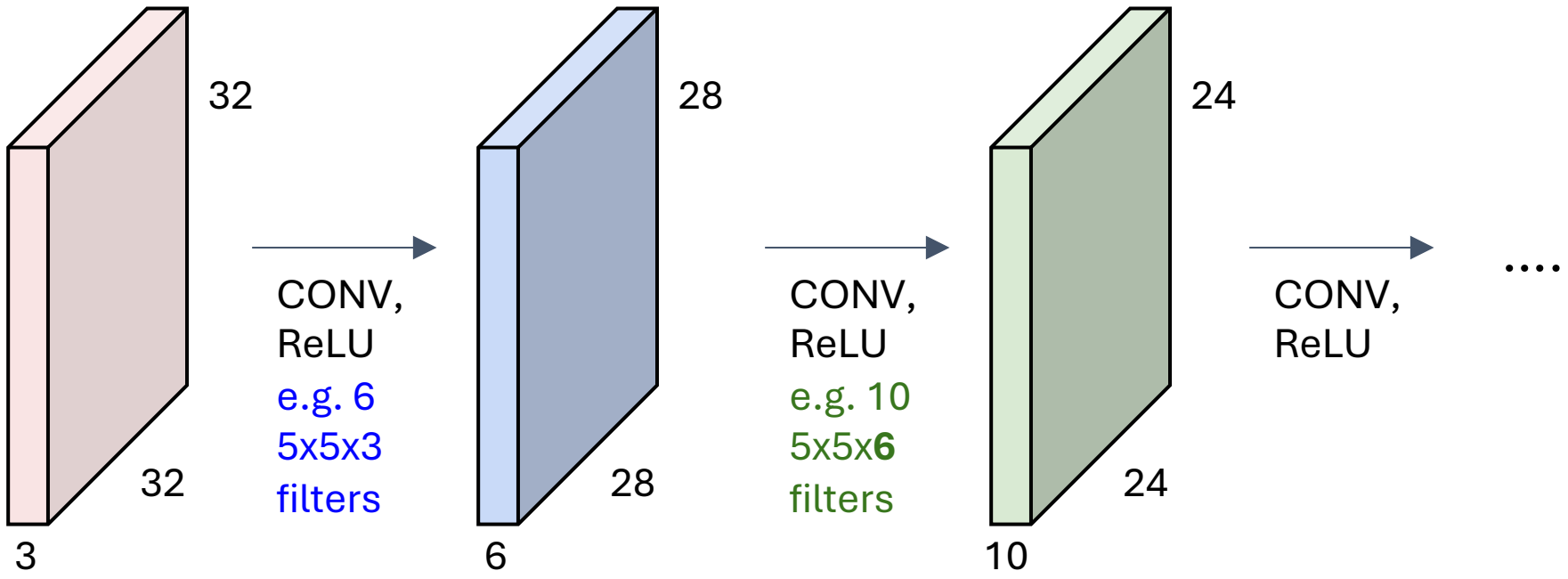
Number of parameters here?

6 filters * 5x5x3 = 6 x 75 = 450 !

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

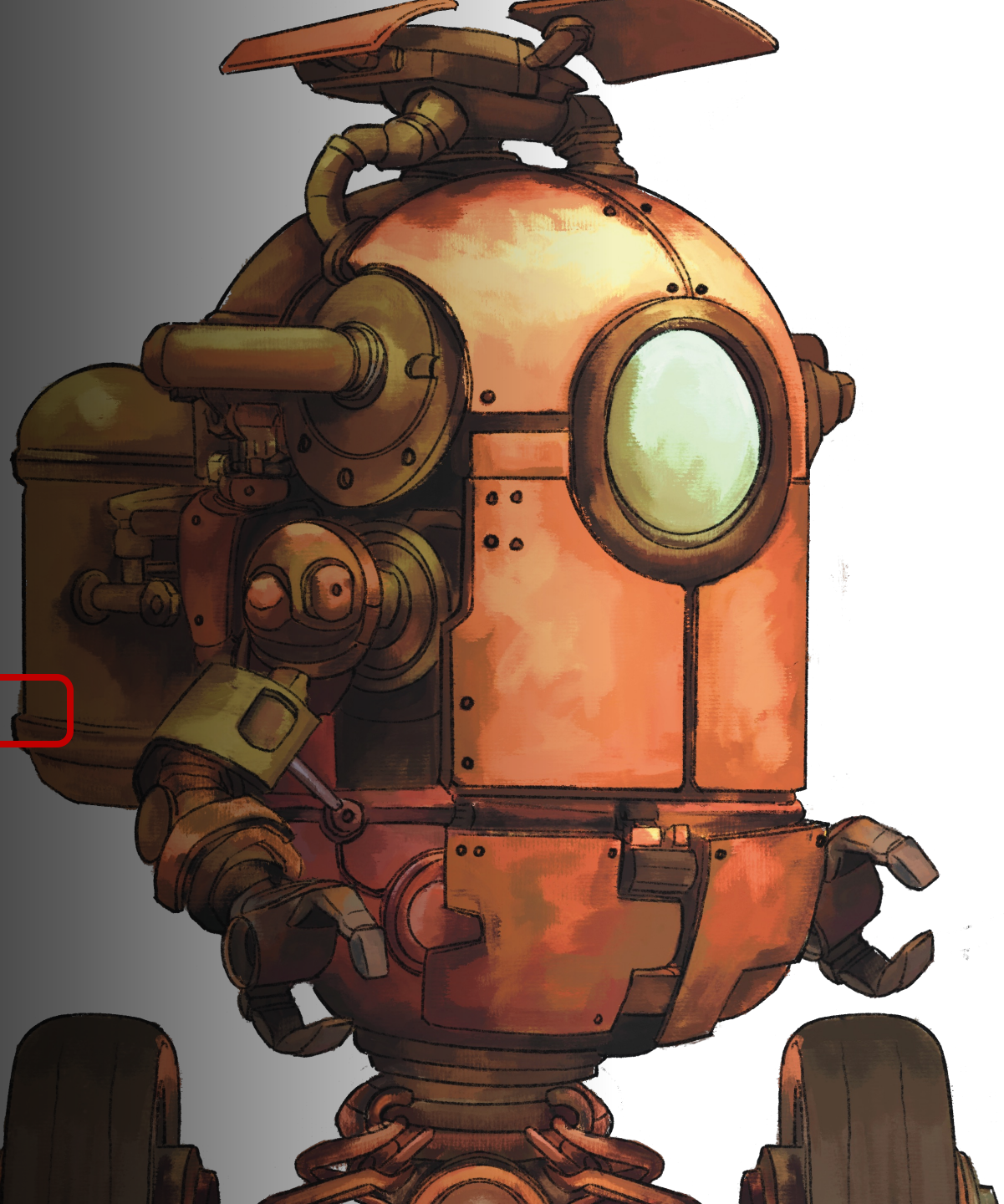


Preview: ConvNet is a sequence of Convolutional Layers, interspersed with **activation functions**



Outline

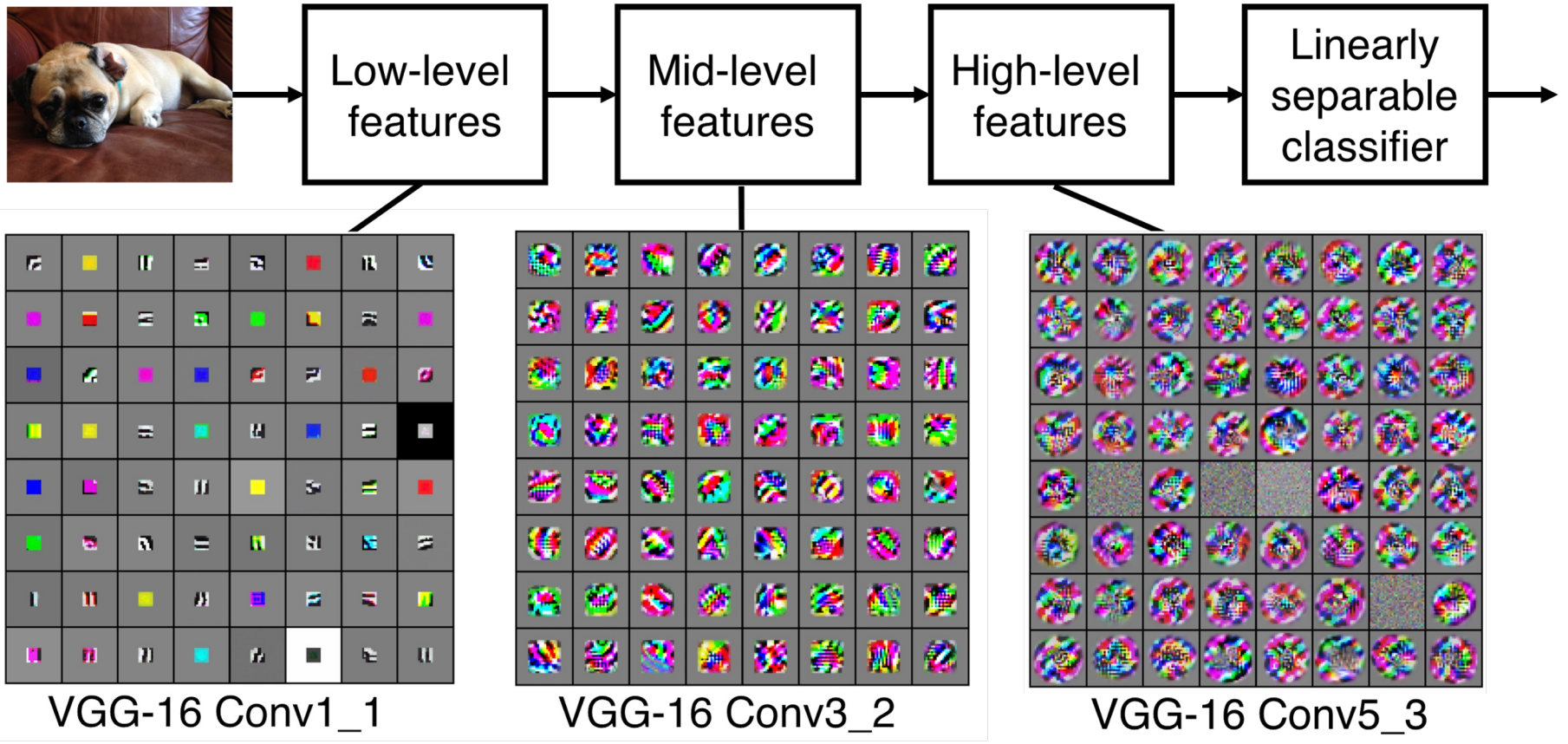
1. Digital Image Acquisition
2. Image Filtering
3. Supervised Learning
4. Multi-layer Perceptrons
5. Convolutional Neural Nets
6. Visualizing Learned Filters
7. Pooling and FC Layers
8. Applications in Vision
9. Object Classification
10. Applications in Robotics



6. Visualizing Learned Filters

[Zeiler and Fergus 2013]

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



Layers 1,2

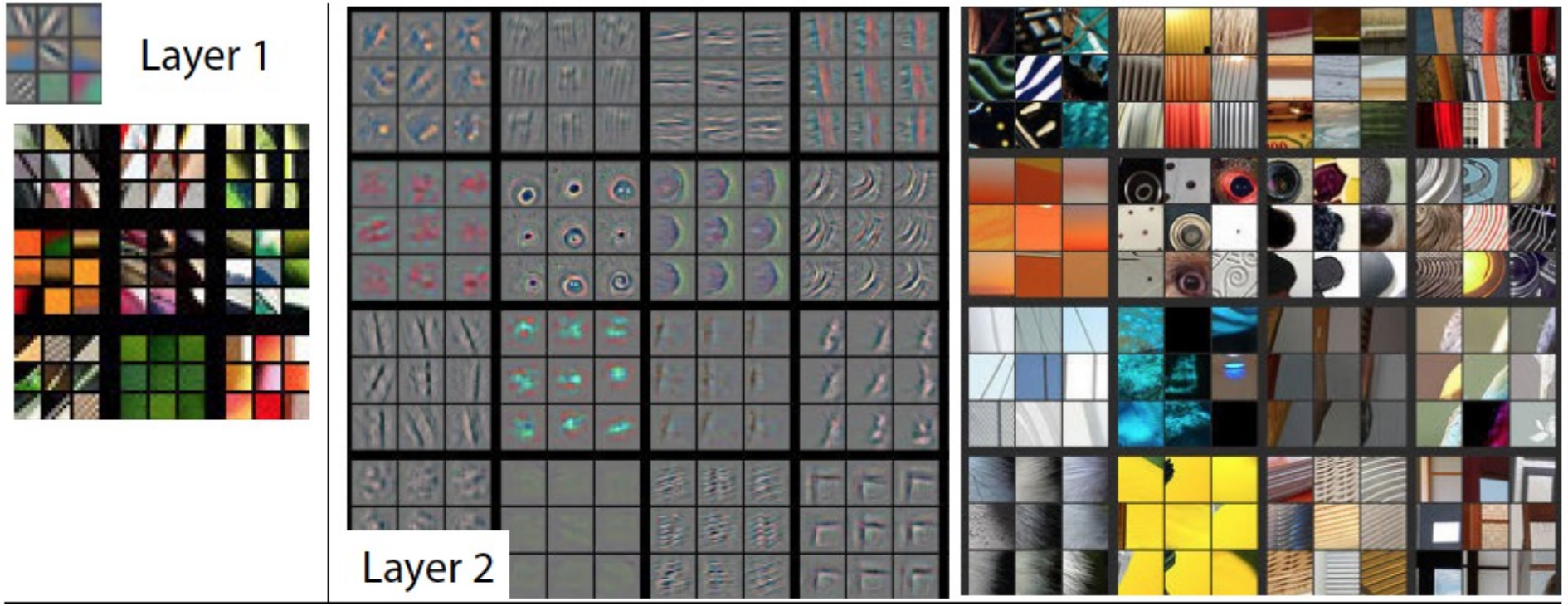


Figure Credit: [Zeiler & Fergus ECCV14]

Layer 3

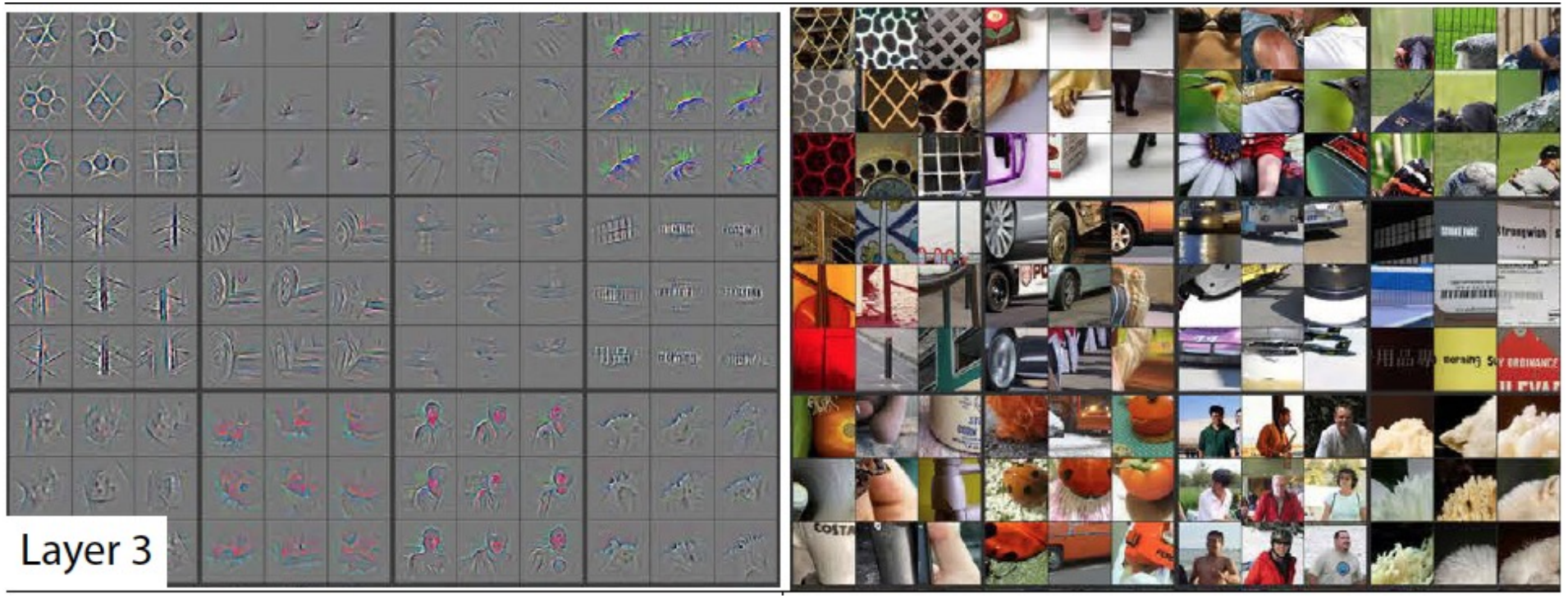


Figure Credit: [Zeiler & Fergus ECCV14]

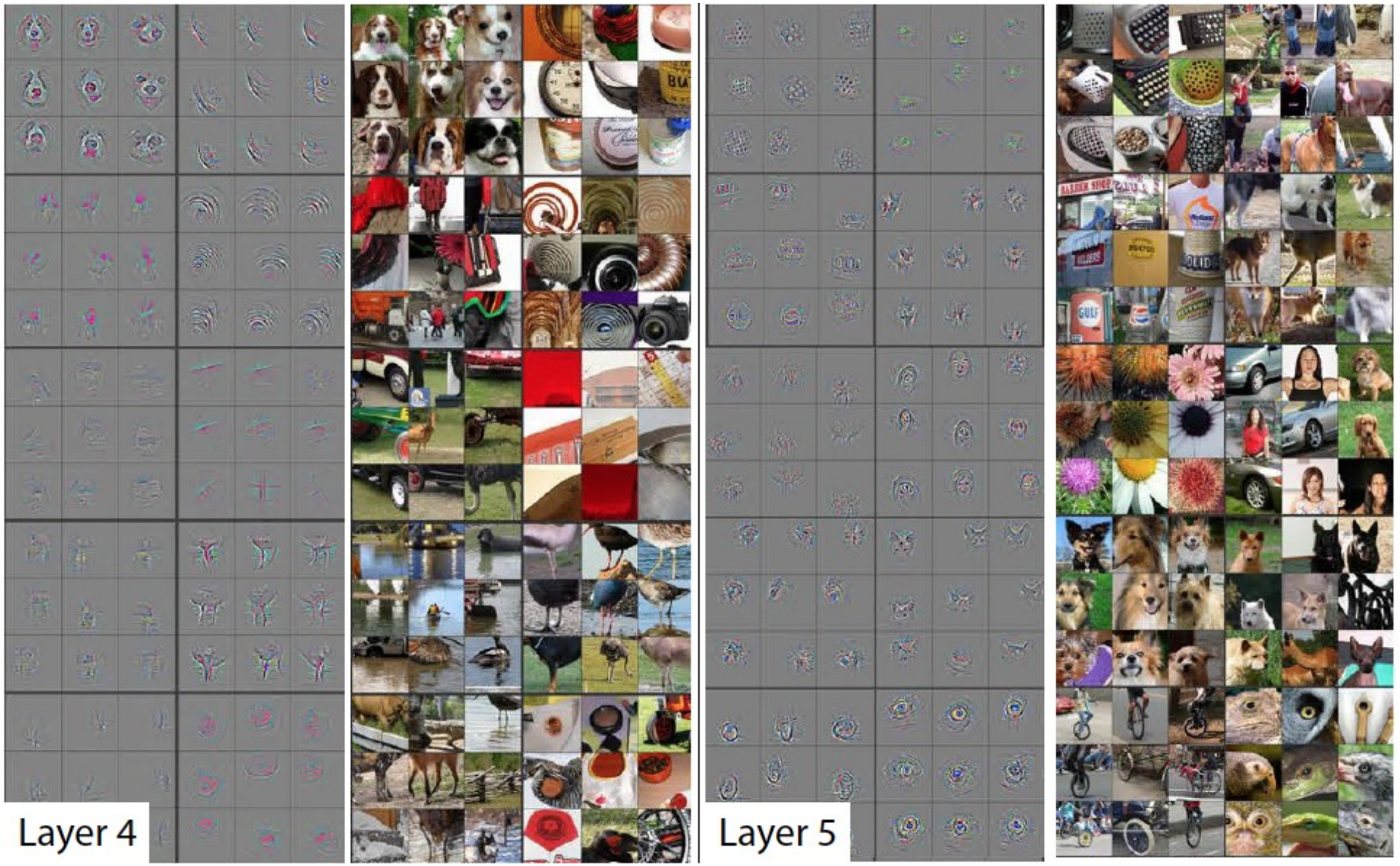
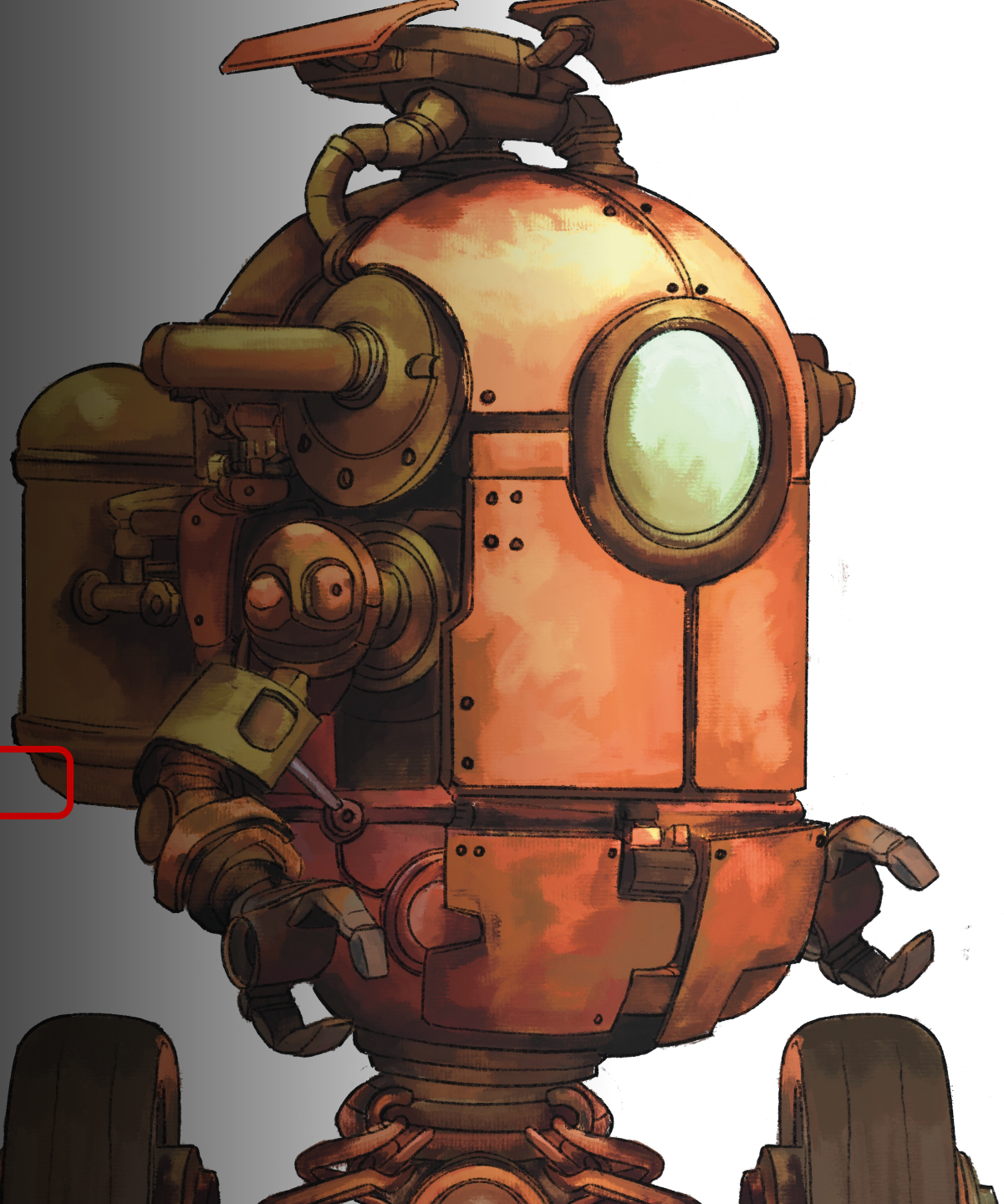


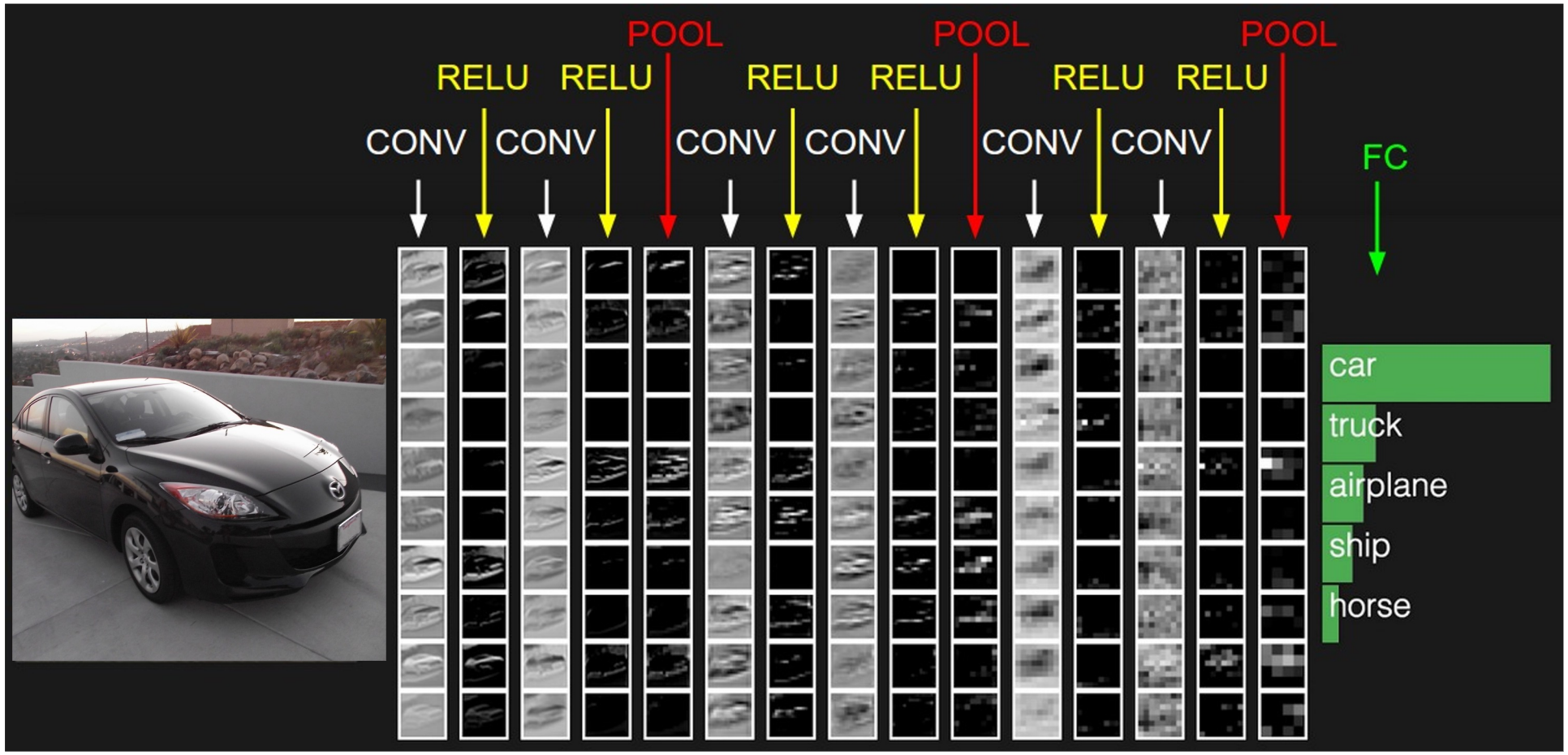
Figure Credit: [Zeiler & Fergus ECCV14]

Outline

1. Digital Image Acquisition
2. Image Filtering
3. Supervised Learning
4. Multi-layer Perceptrons
5. Convolutional Neural Nets
6. Visualizing Learned Filters
7. Pooling and FC Layers
8. Applications in Vision
9. Object Classification
10. Applications in Robotics

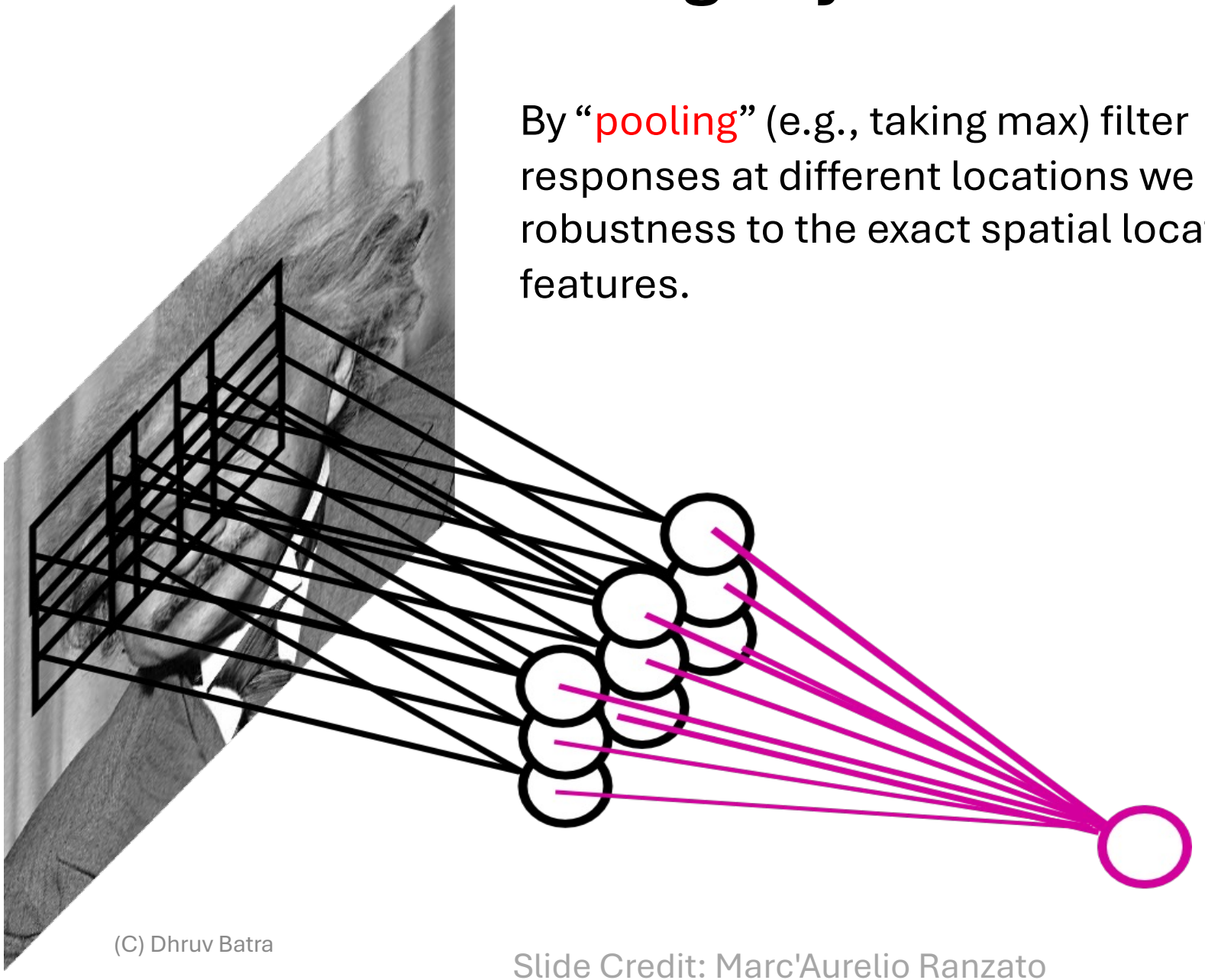


7. Pooling and Fully Convolutional Classifier



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

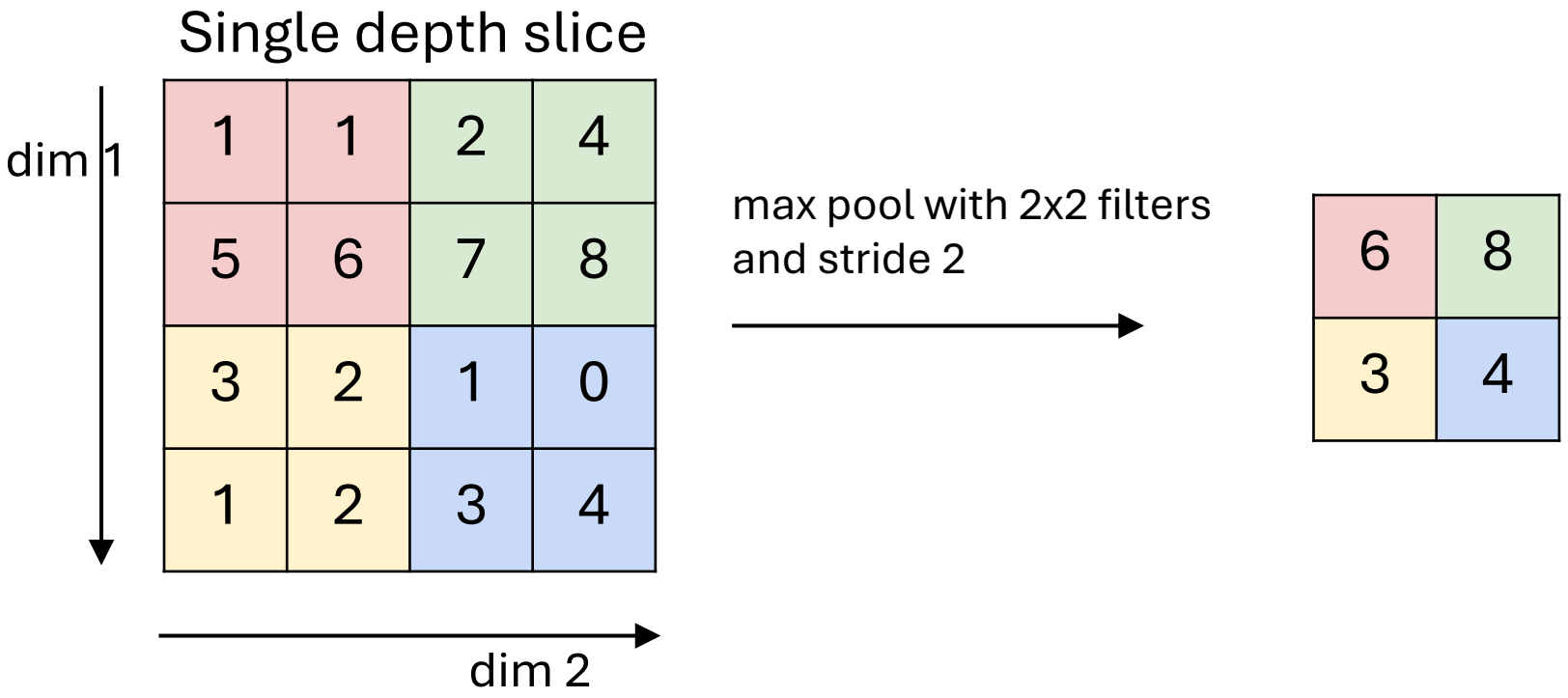
Pooling Layer



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

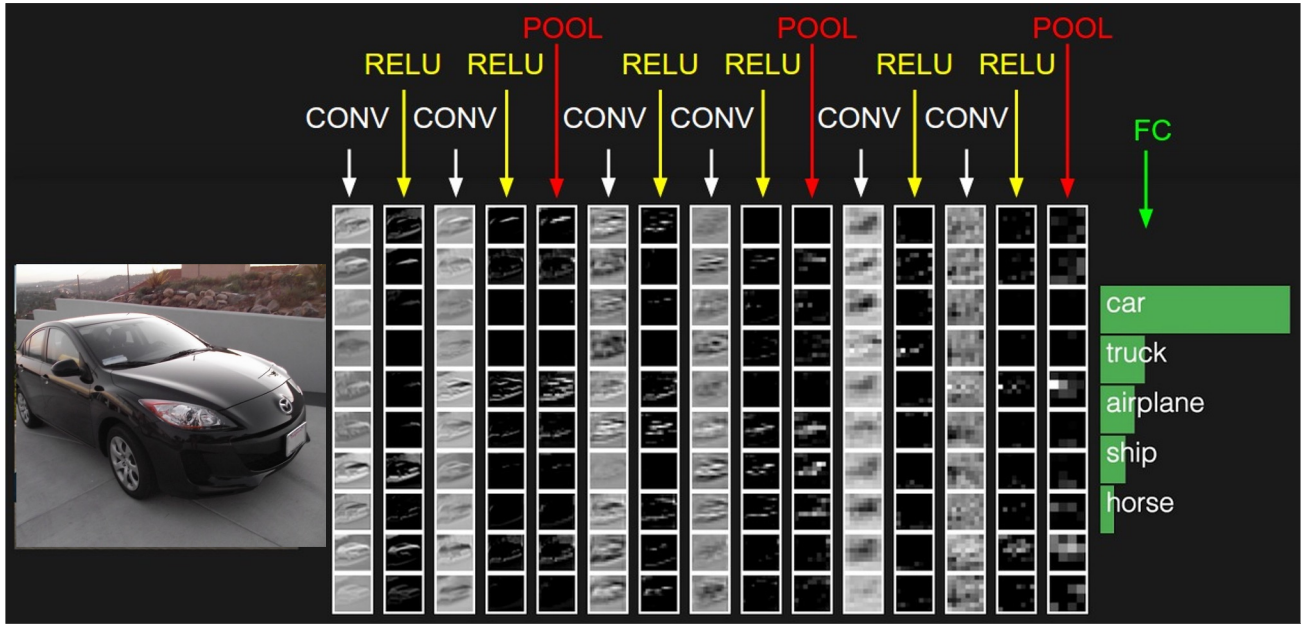
Slide Credit: Marc'Aurelio Ranzato

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)

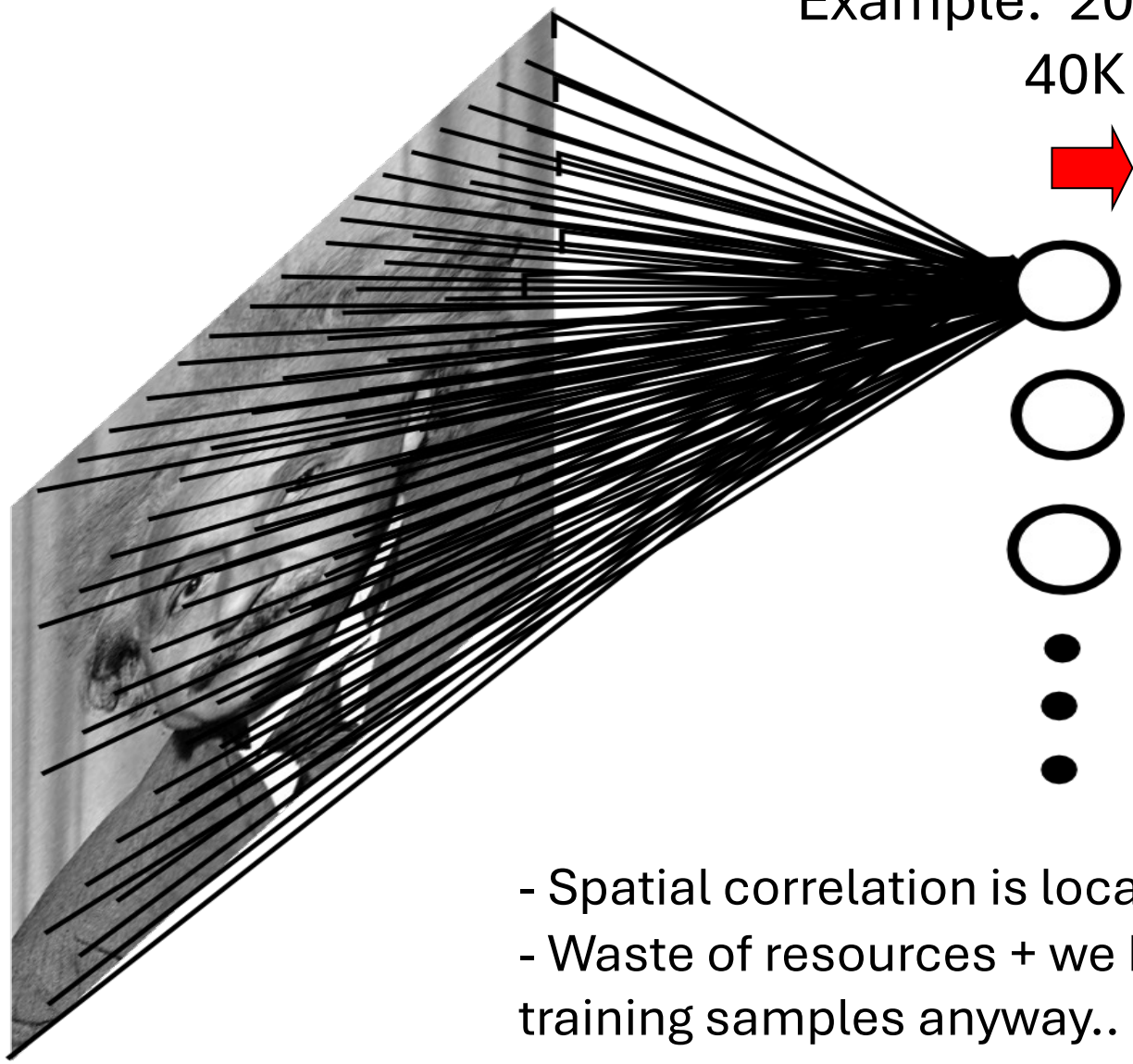
- Contains neurons that connect to the entire input volume, as in multi-layer perceptrons



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

Fully Connected Layer

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

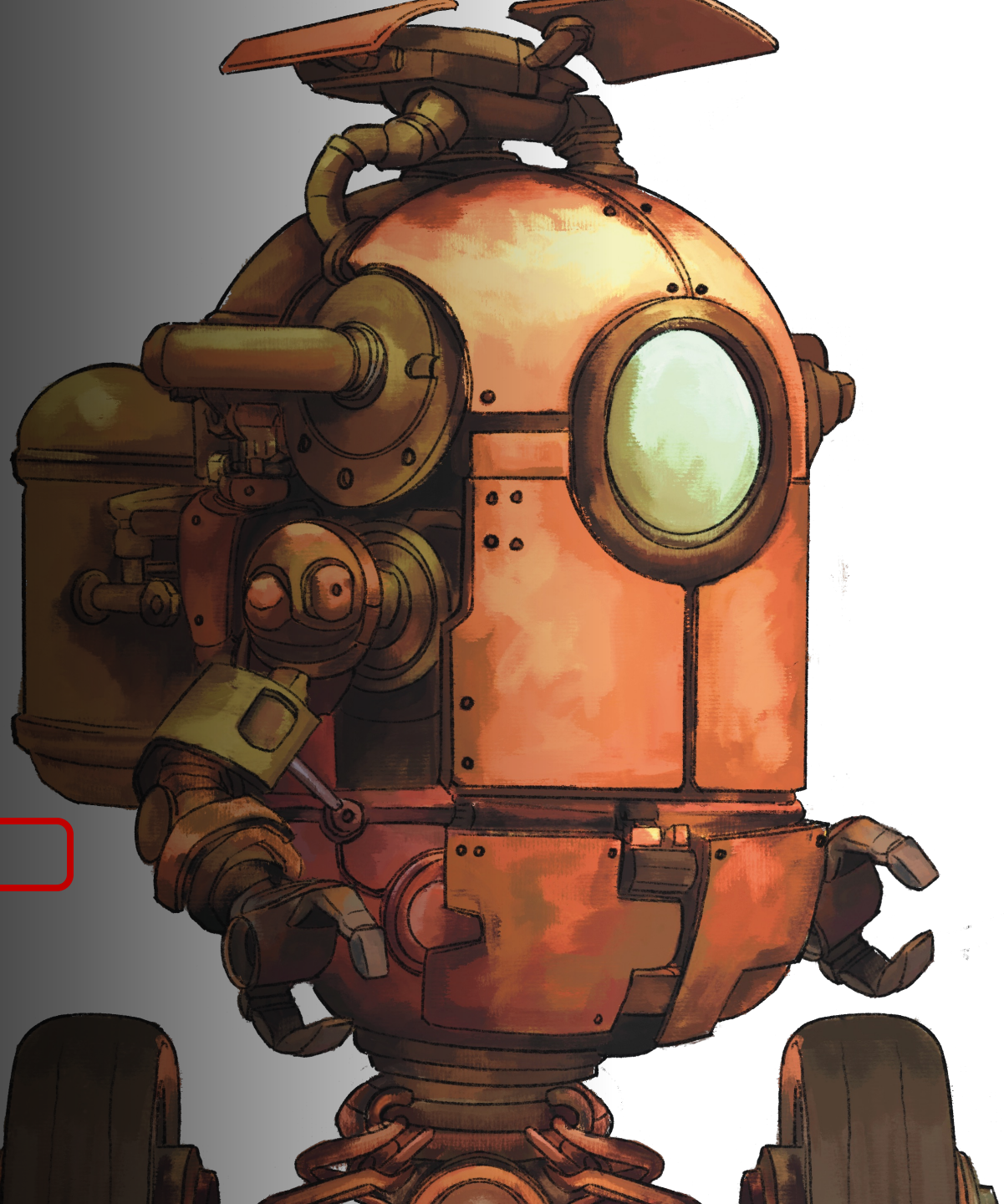


- Spatial correlation is local
- Waste of resources + we have not enough training samples anyway..

Slide Credit: Marc'Aurelio Ranzato

Outline

1. Digital Image Acquisition
2. Image Filtering
3. Supervised Learning
4. Multi-layer Perceptrons
5. Convolutional Neural Nets
6. Visualizing Learned Filters
7. Pooling and FC Layers
8. Applications in Vision
9. Object Classification
10. Applications in Robotics



8. Applications in Vision

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

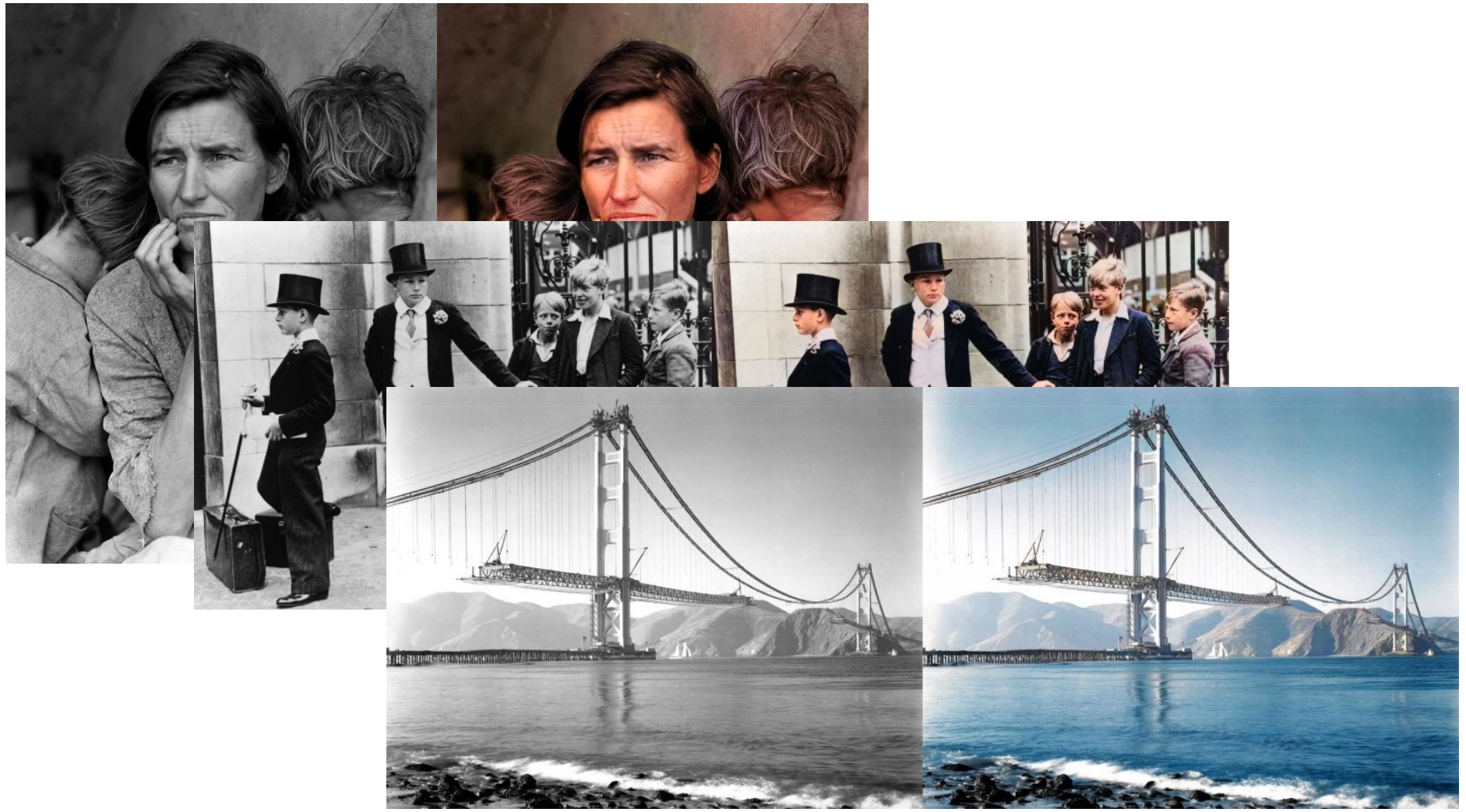
Colorization

- 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





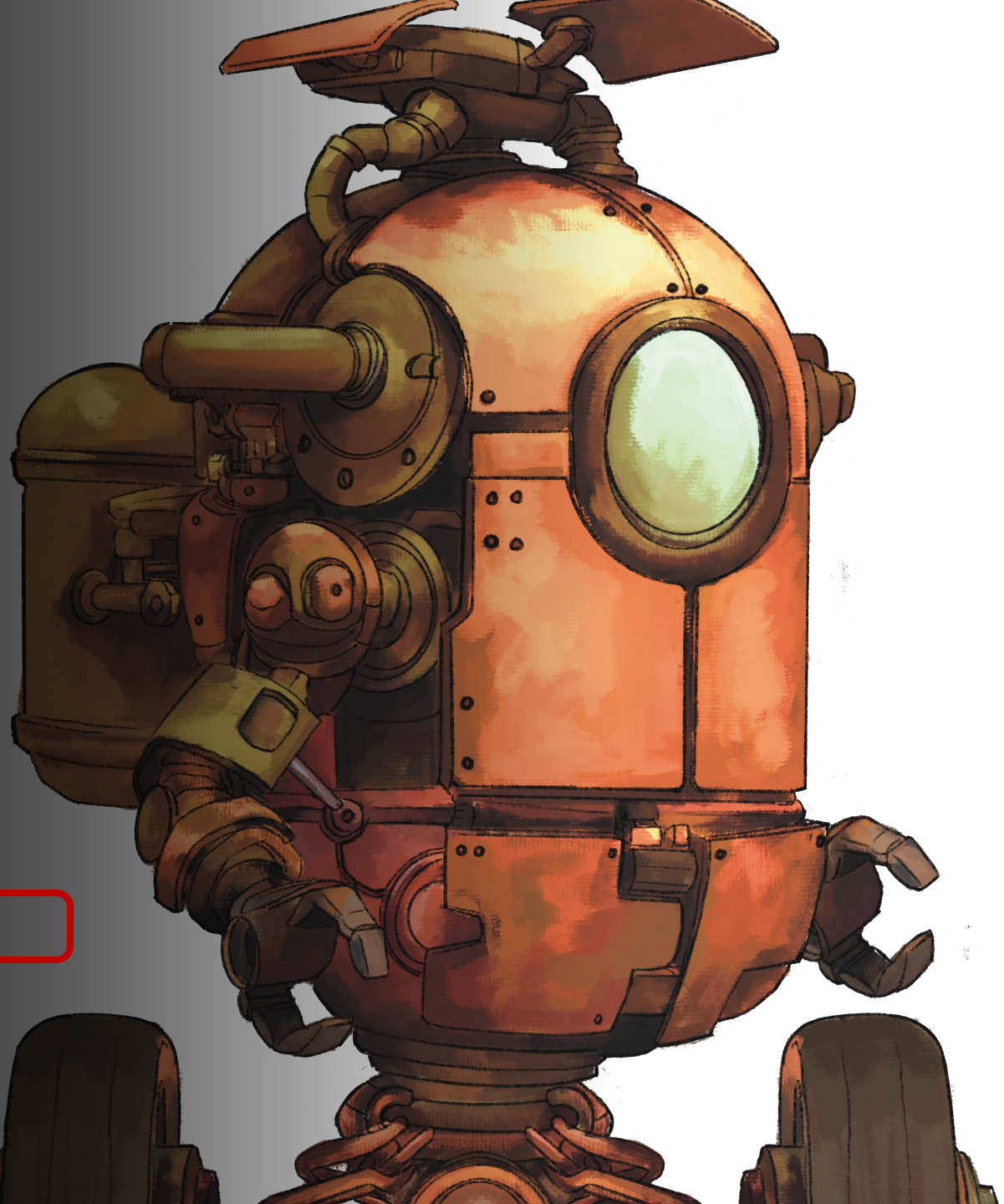
High resolution





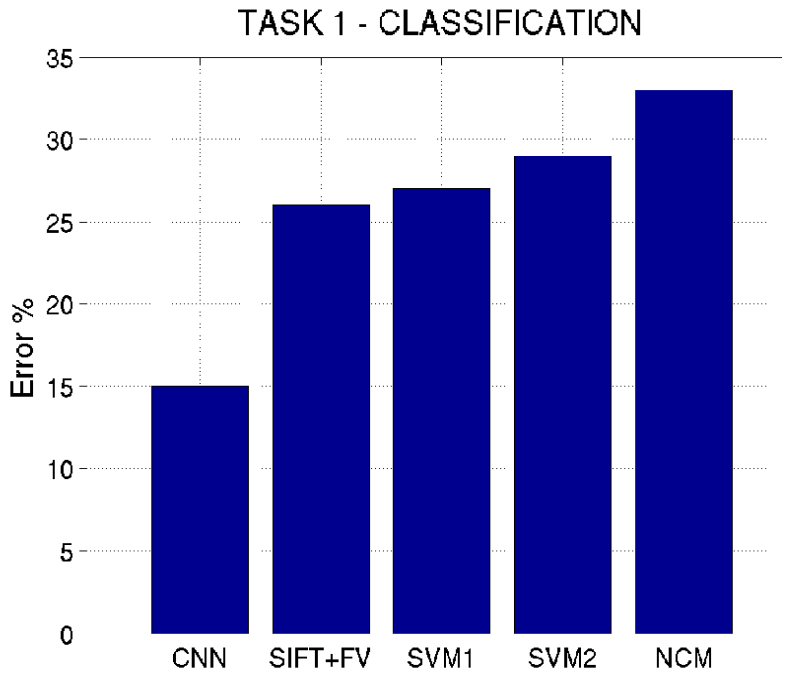
Outline

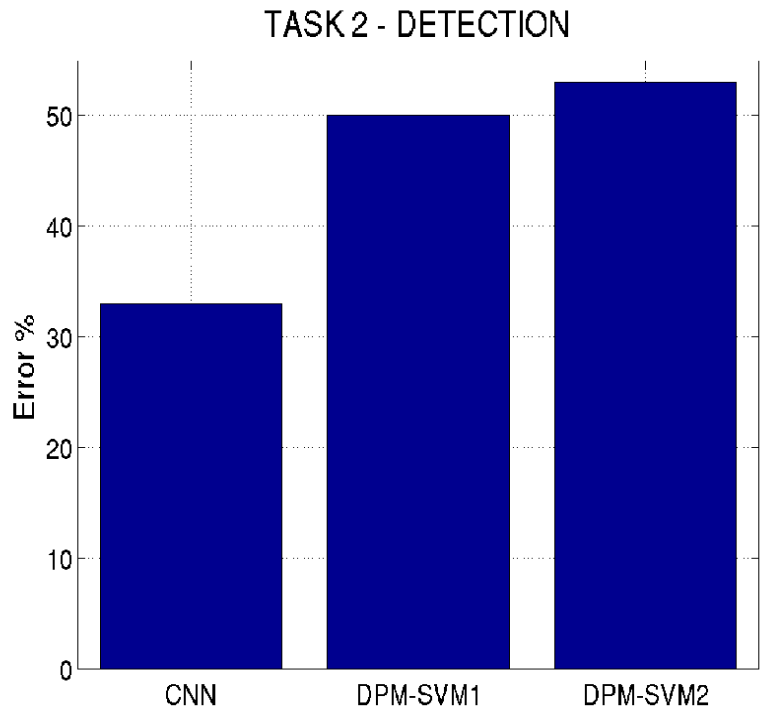
1. Digital Image Acquisition
2. Image Filtering
3. Supervised Learning
4. Multi-layer Perceptrons
5. Convolutional Neural Nets
6. Visualizing Learned Filters
7. Pooling and FC Layers
8. Applications in Vision
9. Object Classification
10. Applications in Robotics



9. Object Classification Revolution

Results: ILSVRC 2012





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



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



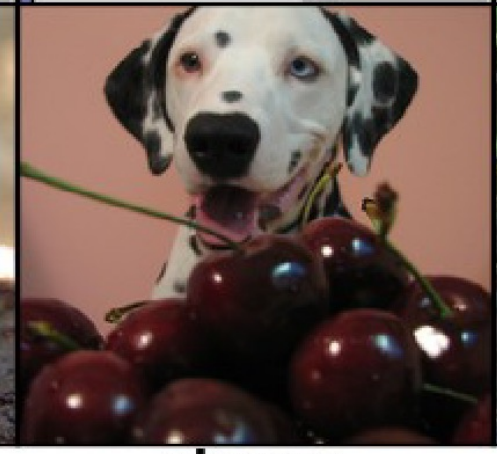
grille

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

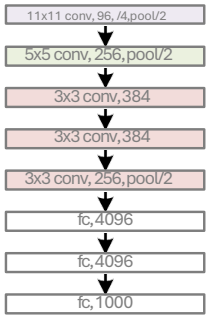


Madagascar cat

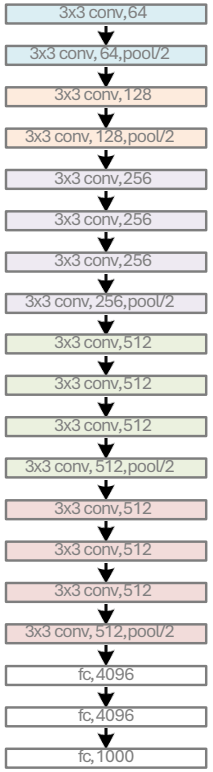
squirrel monkey
spider monkey
titi
indri
howler monkey

Revolution of Depth

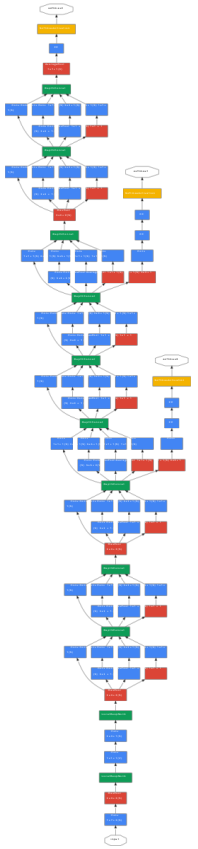
AlexNet, 8 layers
(ILSVRC 2012)



VGG, 19 layers
(ILSVRC 2014)



GoogleNet, 22 layers
(ILSVRC 2014)

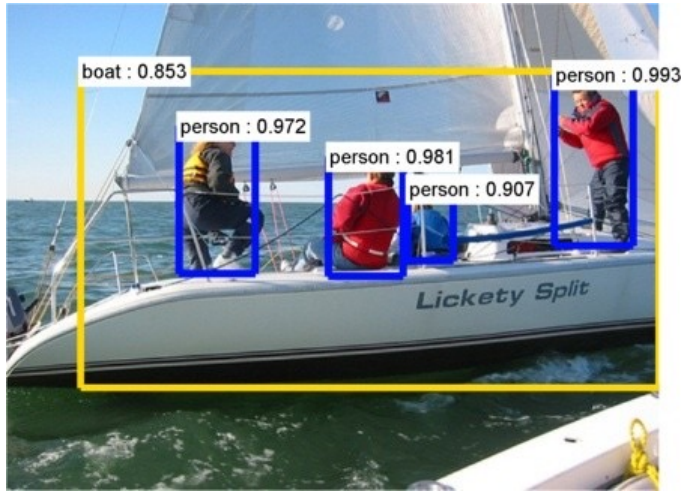


Kaiming He, Xiangyu Zhang, Shaoqing Ren, & Jian Sun. "Deep Residual Learning for Image Recognition". CVPR 2016.

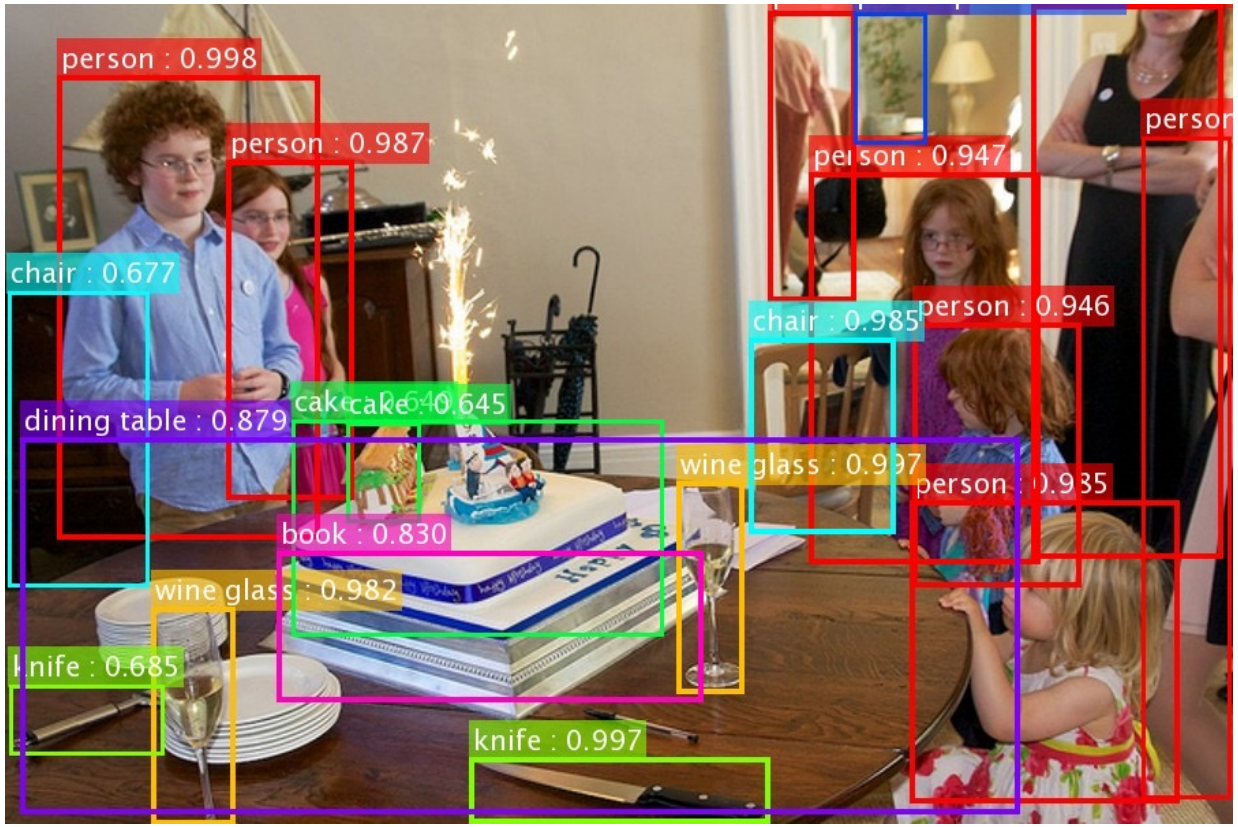
Object Detection



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.

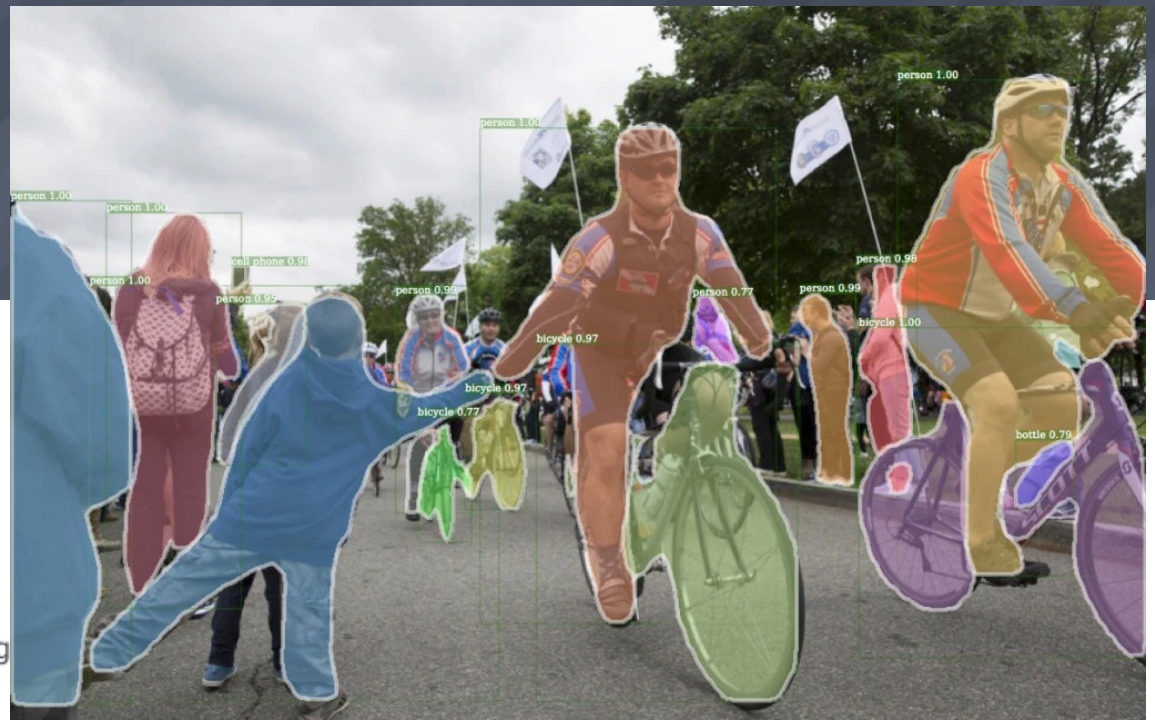


this video is available online: https://youtu.be/WZmSMkK9VuA

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

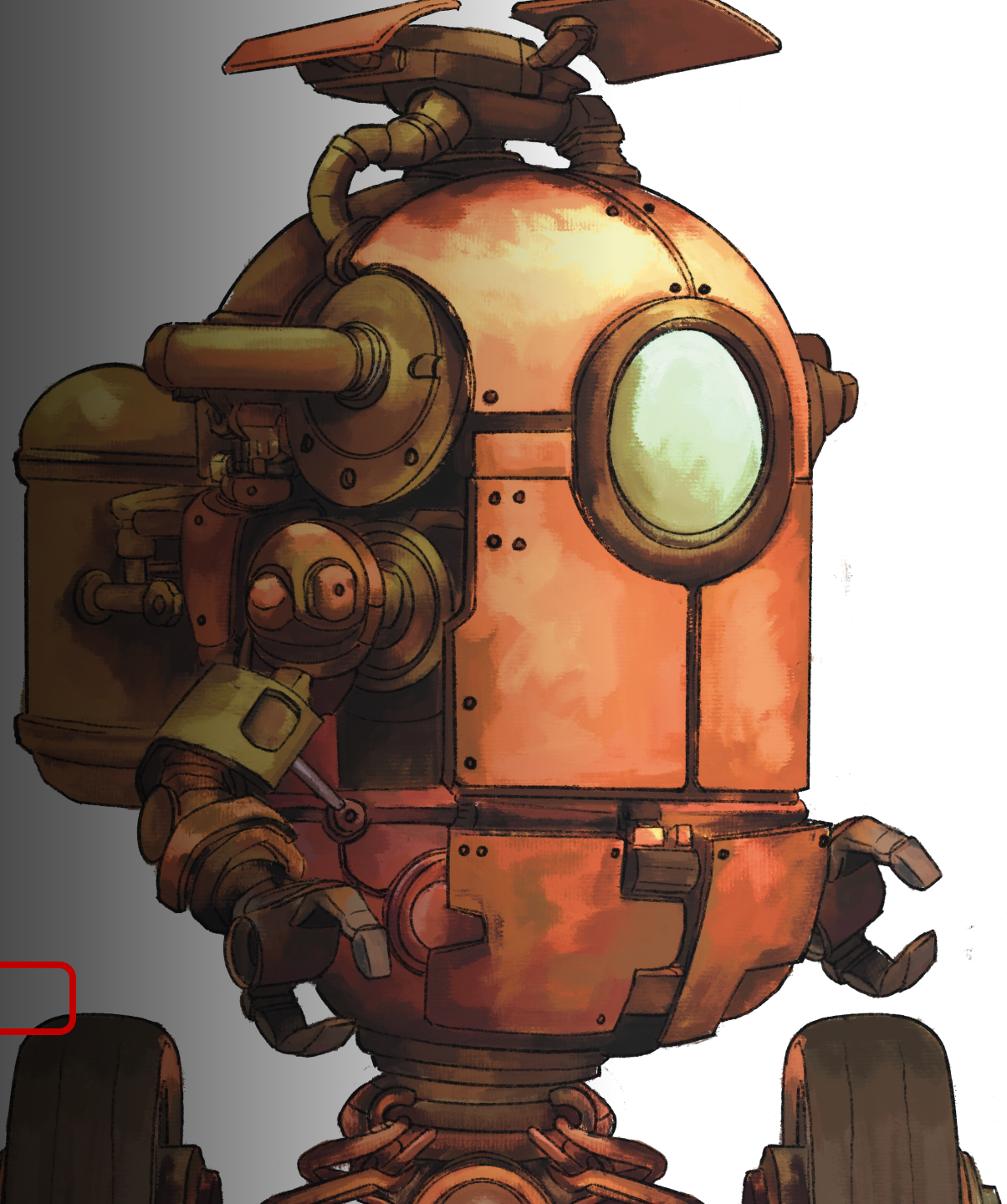
- Mask R-CNN — *Marr Prize at ICCV 2017*
- RetinaNet — *Best Student Paper Award at ICCV 2017*
- Faster R-CNN
- RPN
- Fast R-CNN
- R-FCN

using the following backbone network architectures:

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

Outline

1. Digital Image Acquisition
2. Image Filtering
3. Supervised Learning
4. Multi-layer Perceptrons
5. Convolutional Neural Nets
6. Visualizing Learned Filters
7. Pooling and FC Layers
8. Applications in Vision
9. Object Classification
10. Applications in Robotics



10. Applications in Robotics

- From pixels to action:
  - Object Recognition
  - Depth from a Single Image
  - Deep Stereo
  - 3D Foundation Models

- Deep Reinforcement Learning: for another day!



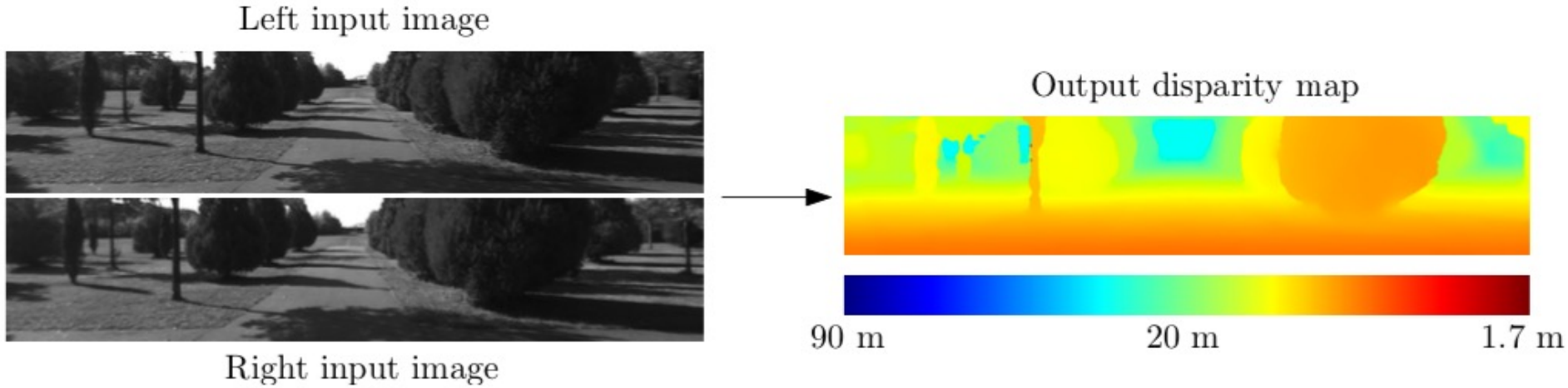
Verdant Robotics: Don’t’ shoot the Broccoli !

Deep Stereo

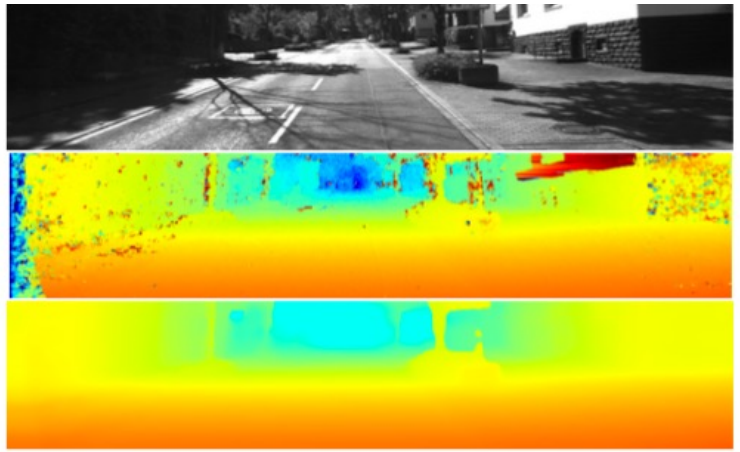
Computing the Stereo Matching Cost with a Convolutional Neural Network

Jure Žbontar
University of Ljubljana
jure.zbontar@fri.uni-lj.si

Yann LeCun
New York University
yann@cs.nyu.edu



- Learns cost function

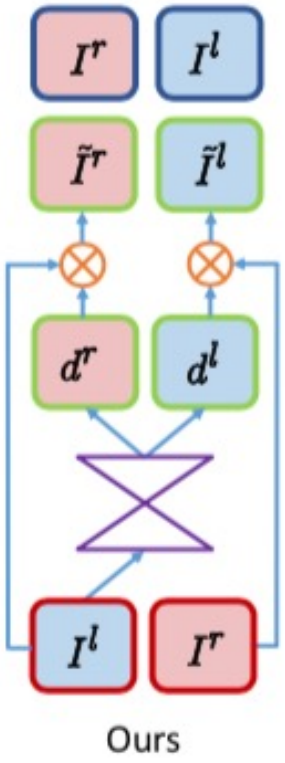


Winner-take all

Smoothed

Monocular Depth

- Can we learn depth from a *single* image?
- Train on stereo, but test on mono!
- Learn to warp left to right and vice versa



Ours

Unsupervised Monocular Depth Estimation with Left-Right Consistency

Clément Godard Oisin Mac Aodha Gabriel J. Brostow
University College London
http://visual.cs.ucl.ac.uk/pubs/monoDepth/

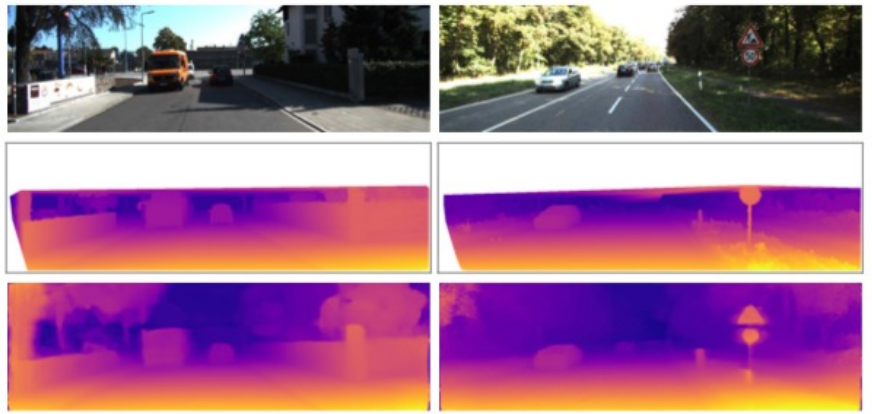


Figure 1. Our depth prediction results on KITTI 2015. Top to bottom: input image, ground truth disparities, and our result. Our method is able to estimate depth for thin structures such as street signs and poles.

Stereo Datasets

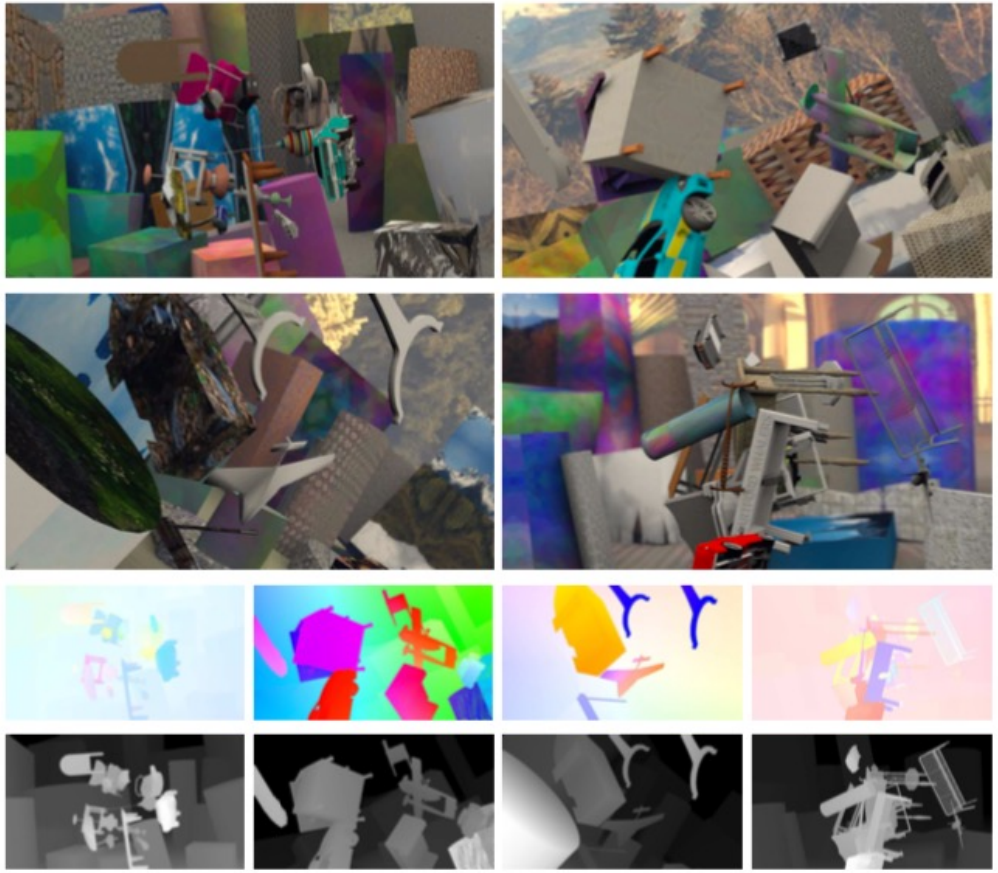
- *FlyingThings3D -> DispNet*

A Large Dataset to Train Convolutional Networks for Disparity, Optical Flow, and Scene Flow Estimation

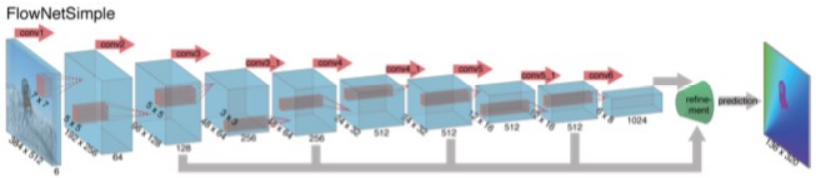
Nikolaus Mayer*[1], Eddy Ilg*[1], Philip Häusser*[2], Philipp Fischer*[1†]
[1]University of Freiburg [2]Technical University of Munich
[1]{mayern,ilg,fischer}@cs.uni-freiburg.de [2]haeusser@cs.tum.edu

Daniel Cremers
Technical University of Munich
cremers@tum.de

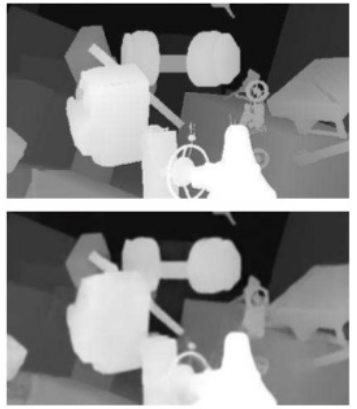
Alexey Dosovitskiy, Thomas Brox
University of Freiburg
{dosovits,brox}@cs.uni-freiburg.de



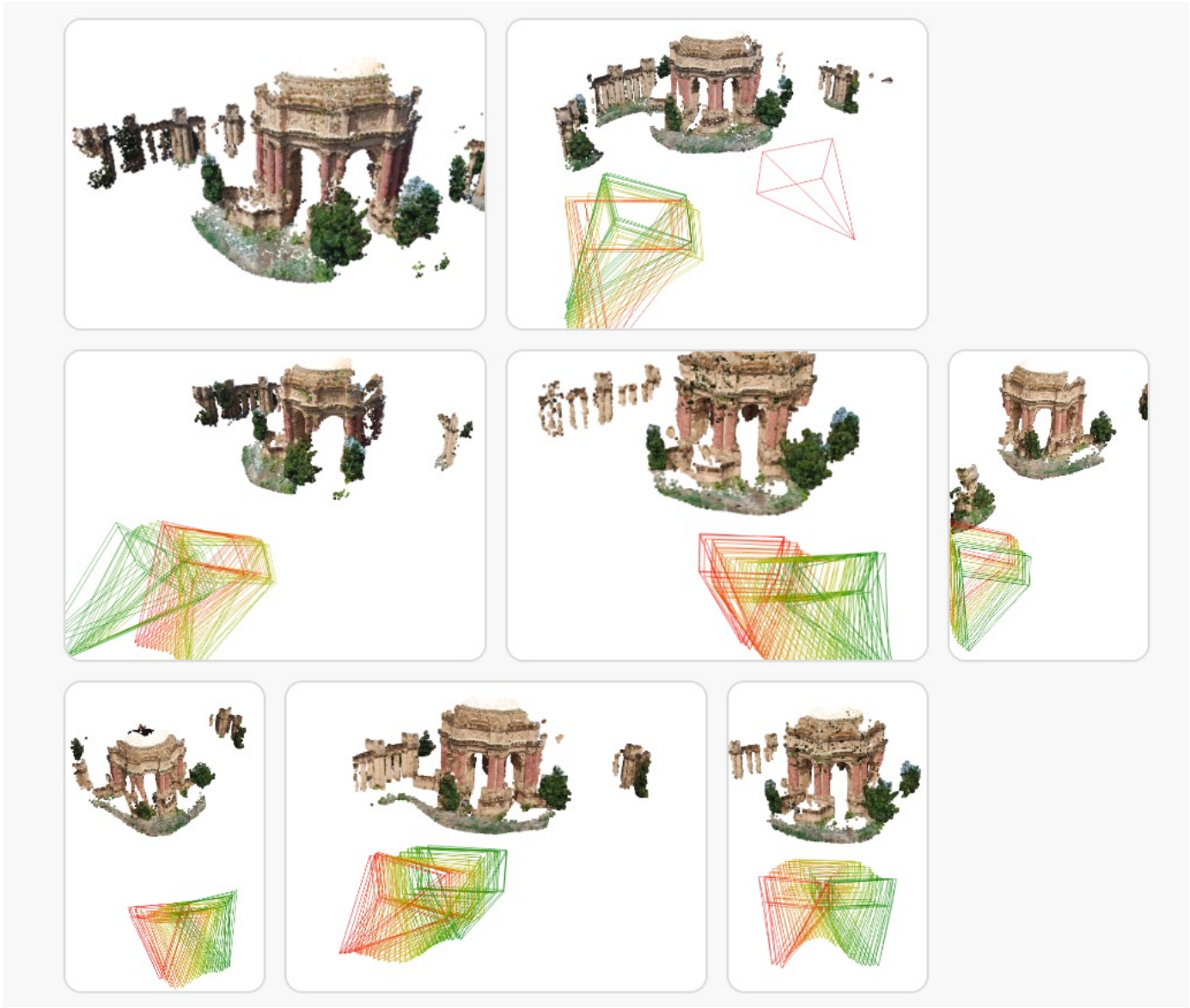




disp GT / prediction



3D Reconstruction VGGT: Best paper at CVPR 2025



Summary

1. Digital Image Acquisition
2. Image Filtering
3. Supervised Learning
4. Multi-layer Perceptrons
5. Convolutional Neural Nets
6. Visualizing Learned Filters
7. Pooling and FC Layers
8. Applications in Vision
9. Object Classification
10. Applications in Robotics

