Deep Learning in Image Processing

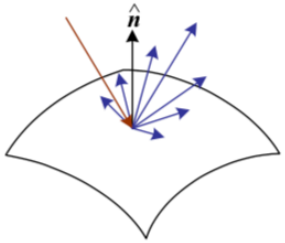
Topics:

- Image Filtering 101
- CNNs 101
- Image Processing Pipelines

Frank Dellaert

CS 4476 Computer Vision

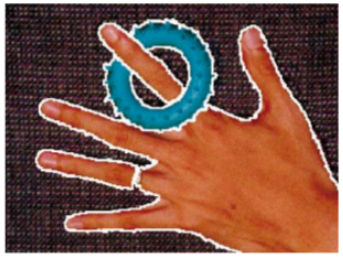
Many slides from Stanford's CS231N by Fei-Fei Li, Justin Johnson, Serena Yeung, as well as some slides on filtering from Devi Parikh and Kristen Grauman, who may in turn have borrowed some from others



2. Image Formation

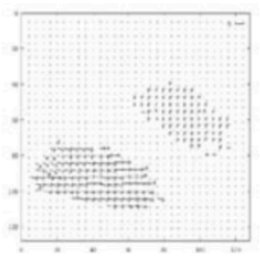
3. Image Processing

4. Features

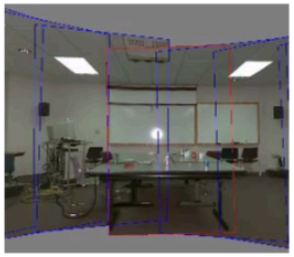


5. Segmentation

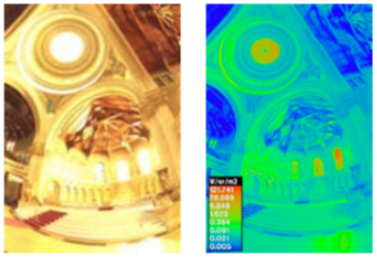
6-7. Structure from Motion



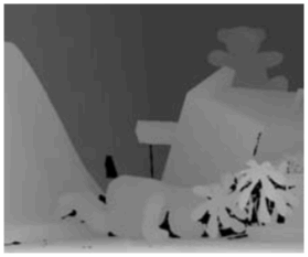
8. Motion



9. Stitching

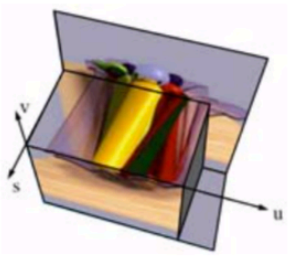


10. Computational Photography



11. Stereo

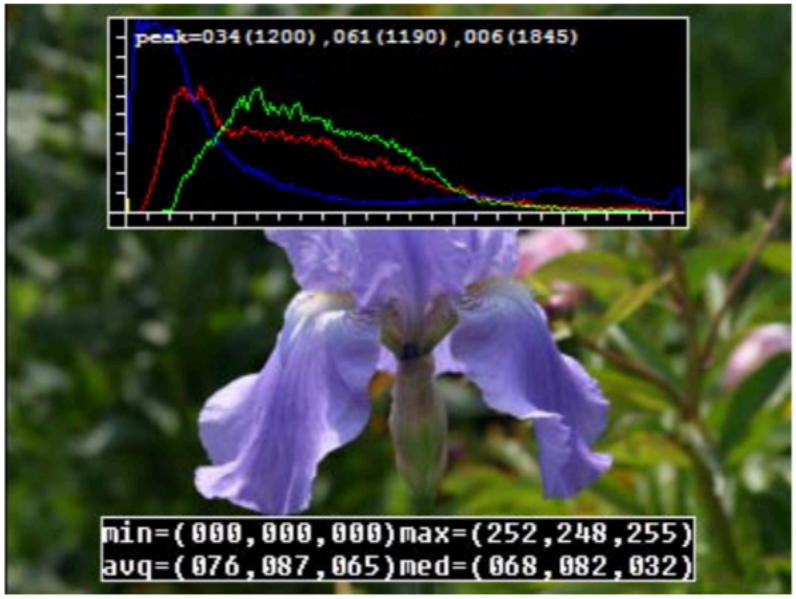
12. 3D Shape

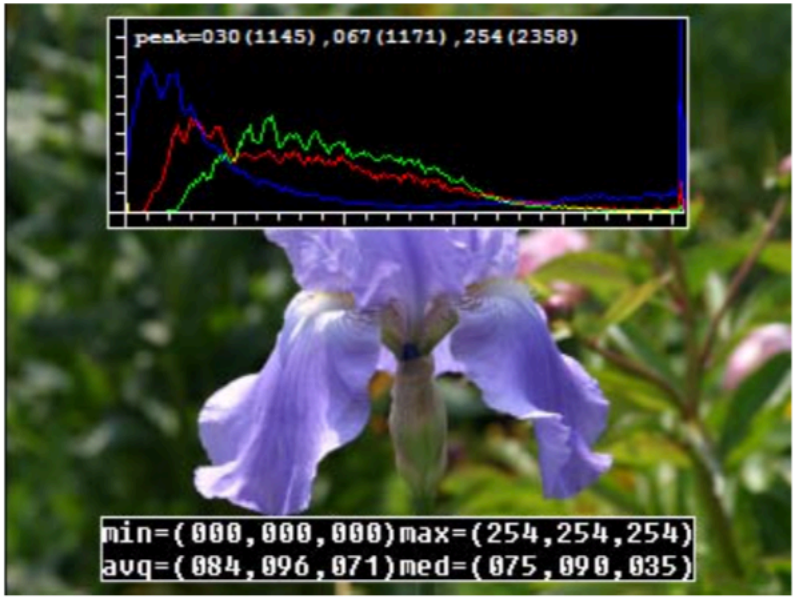


13. Image-based Rendering

14. Recognition

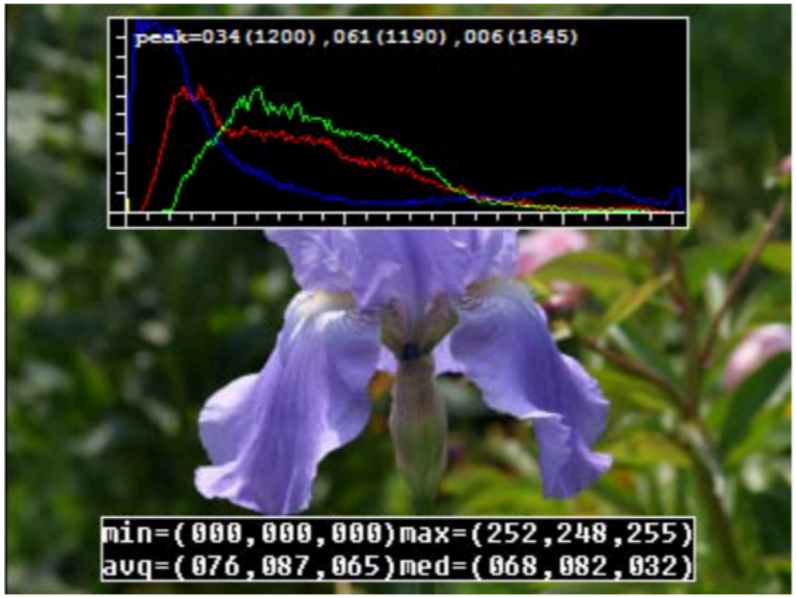
Contrast

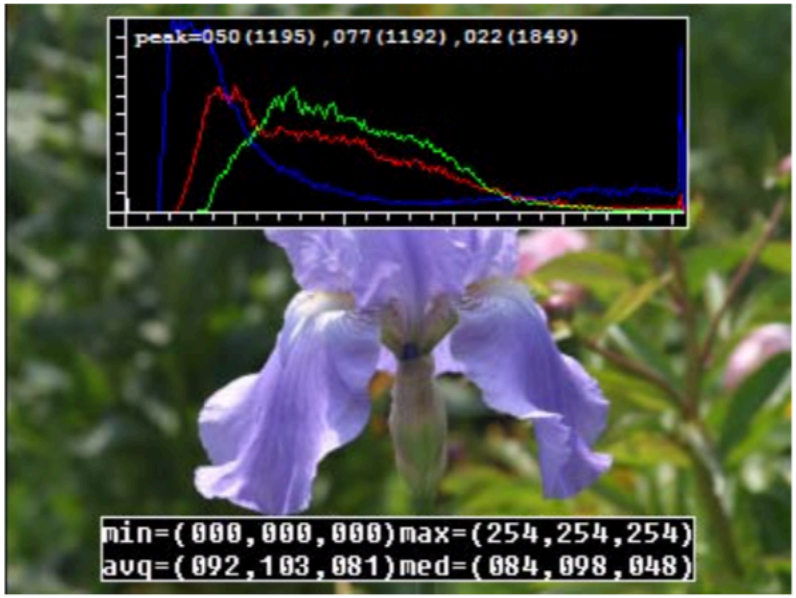




- g(x) = a f(x), a=1.1

Brightness





- g(x) = f(x) + b, b=16

Image filtering

- Compute a function of the local neighborhood at each pixel in the image
  - Function specified by a “filter” or mask saying how to combine values from neighbors.

- Uses of filtering:
  - Enhance an image (denoise, resize, etc)
  - Extract information (texture, edges, etc)
  - Detect patterns (template matching)

Slide credit: Kristen Grauman, Adapted from Derek Hoiem

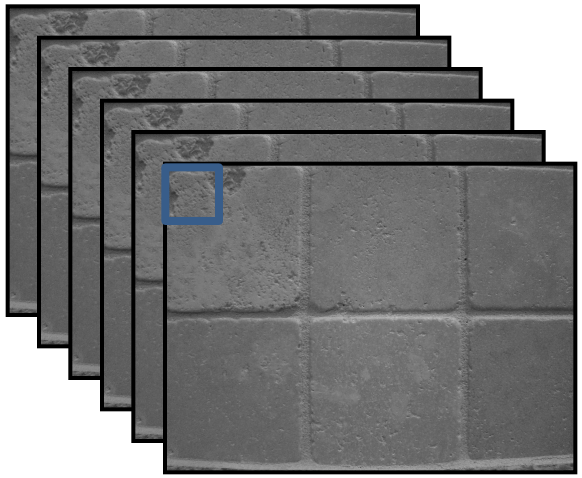
Image filtering

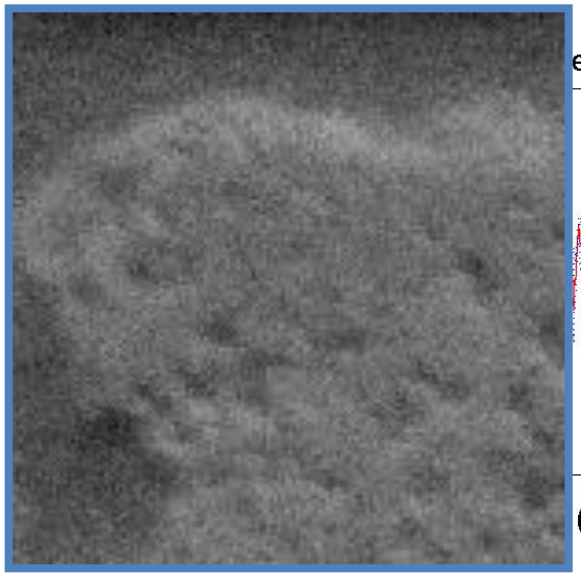
- Compute a function of the local neighborhood at each pixel in the image
  - Function specified by a “filter” or mask saying how to combine values from neighbors.

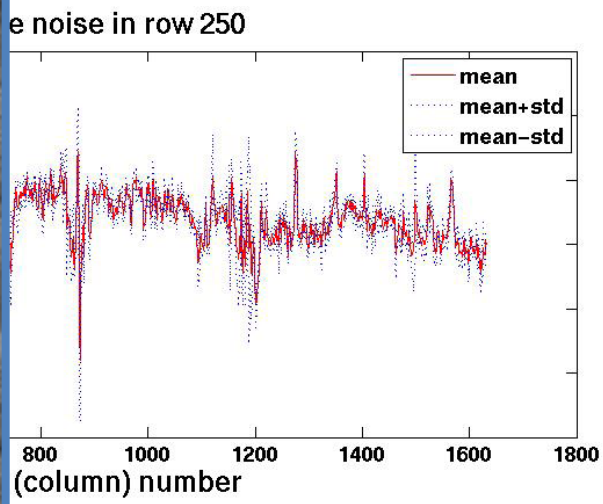
- Uses of filtering:
  - Enhance an image (denoise, resize, etc)
  - Extract information (texture, edges, etc)
  - Detect patterns (template matching)

Slide credit: Kristen Grauman, Adapted from Derek Hoiem

Motivation: noise reduction







- Even multiple images of the **same static scene** will not be identical.

Slide credit: Adapted from Kristen Grauman

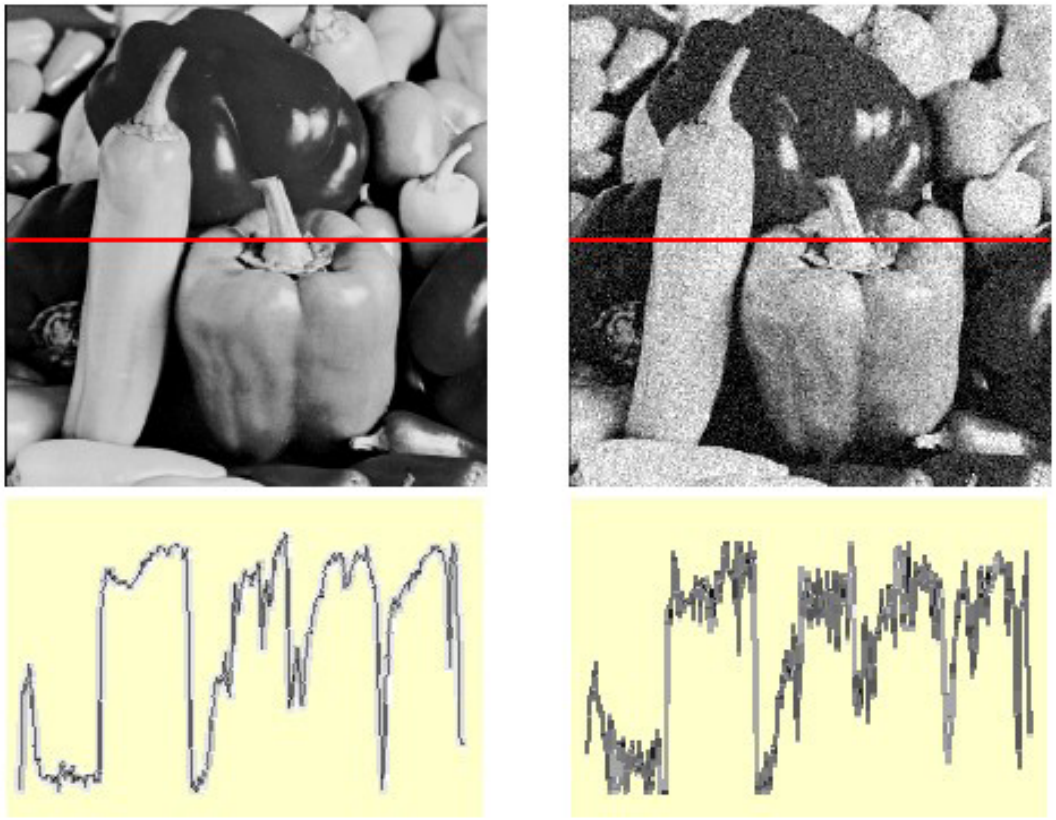
Common types of noise

- **Salt and pepper noise**: random occurrences of black and white pixels
- **Impulse noise:** random occurrences of white pixels
- **Gaussian noise**: variations in intensity drawn from a Gaussian normal distribution

Original

Slide credit: Steve Seitz

Gaussian noise



$$f(x,y) = \overbrace{\bar{f}(x,y)}^{\text{Ideal Image}} + \overbrace{\eta(x,y)}^{\text{Noise process}}$$

Gaussian i.i.d. ("white") noise:
$\eta(x,y) \sim \mathcal{N}(\mu,\sigma)$

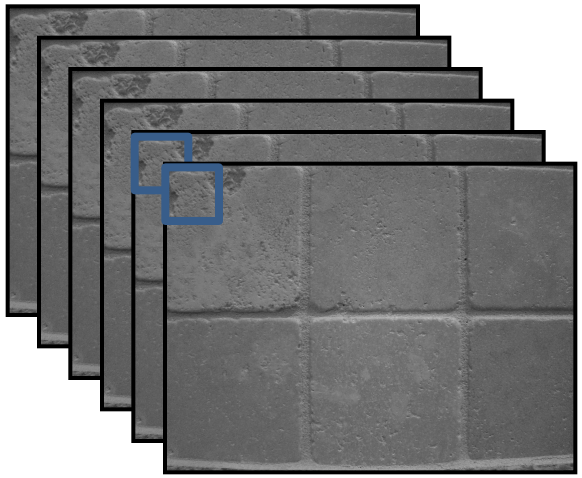
```
>> noise = randn(size(im)).*sigma;

>> output = im + noise;
```

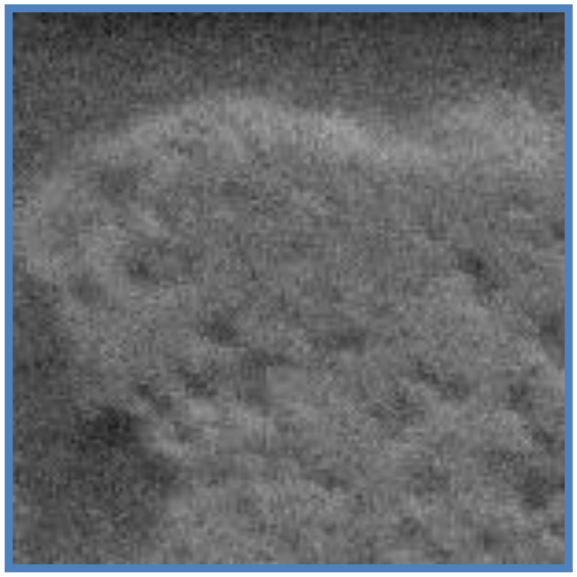
What is impact of the sigma?

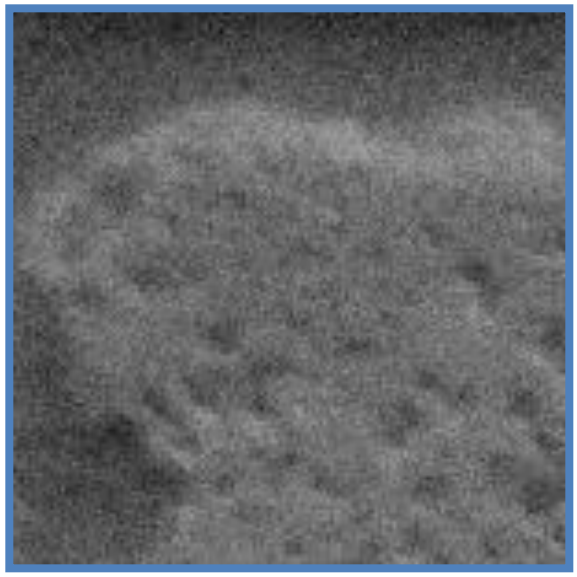
Slide credit: Kristen Grauman

Figure from Martial Hebert

Motivation: noise reduction







- Even multiple images of the same static scene will not be identical.
- How could we reduce the noise, i.e., give an estimate of the true intensities?
- **What if there's only one image?**

Slide credit: Kristen Grauman

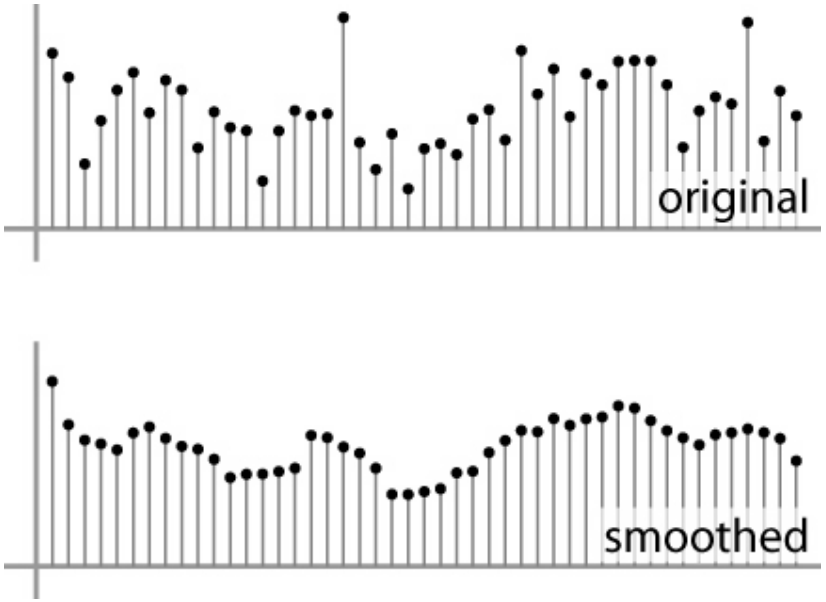
First attempt at a solution

- Let's replace each pixel with an average of all the values in its neighborhood
- Assumptions:
  - Expect pixels to be like their neighbors
  - Expect noise processes to be independent from pixel to pixel

Slide credit: Kristen Grauman

First attempt at a solution

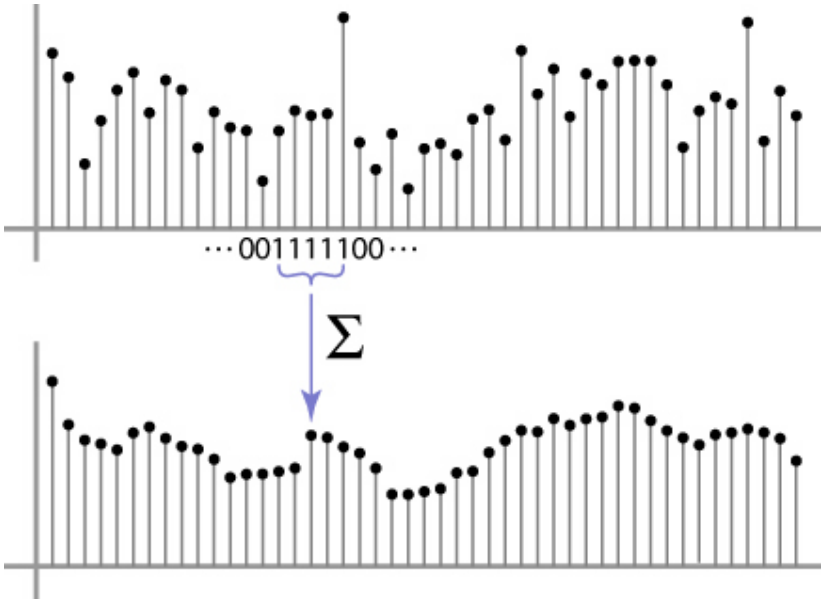
- Let's replace each pixel with an average of all the values in its neighborhood
- Moving average in 1D:



Slide credit: S. Marschner

Weighted Moving Average

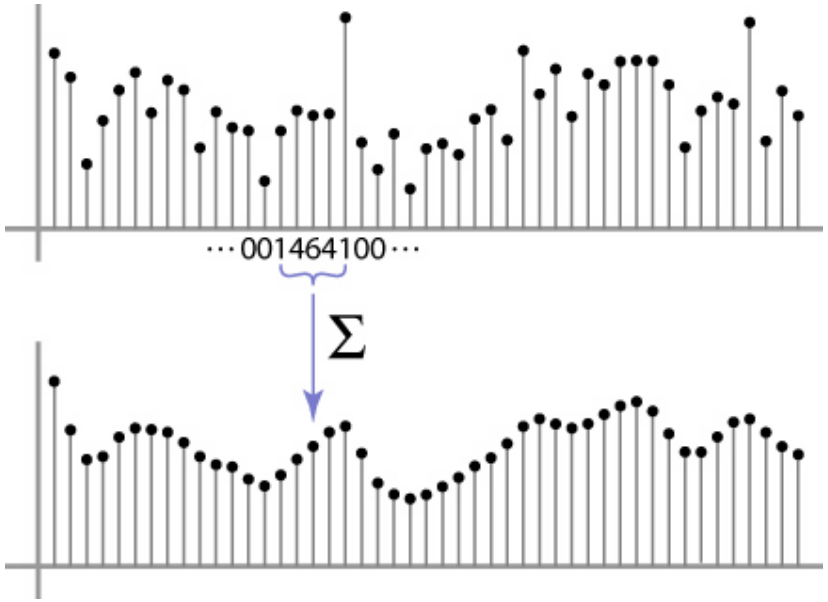
- Can add weights to our moving average
- *Weights* [1, 1, 1, 1, 1] / 5



Slide credit: S. Marschner

Weighted Moving Average

- Non-uniform weights [1, 4, 6, 4, 1] / 16



Slide credit: S. Marschner

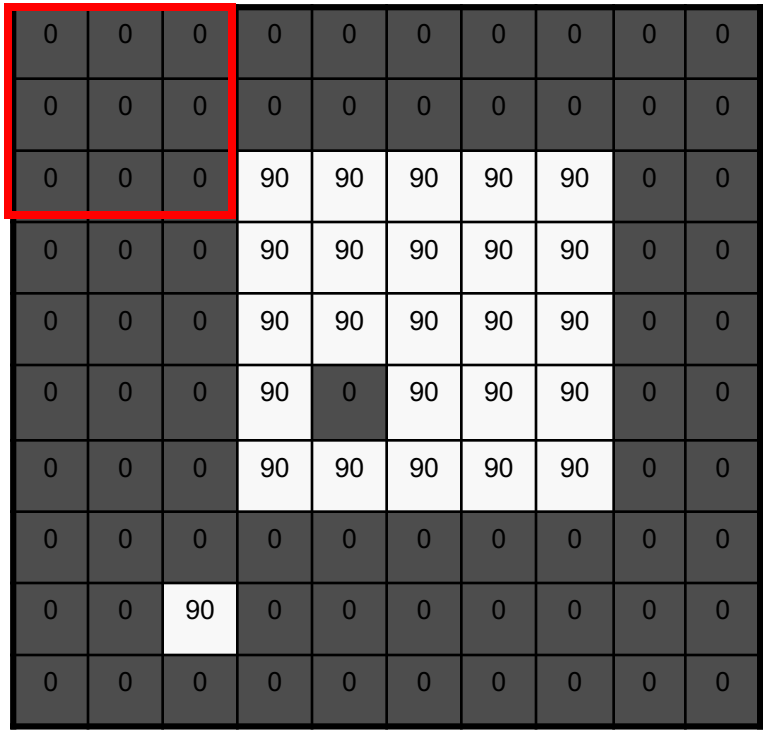
Image filtering

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

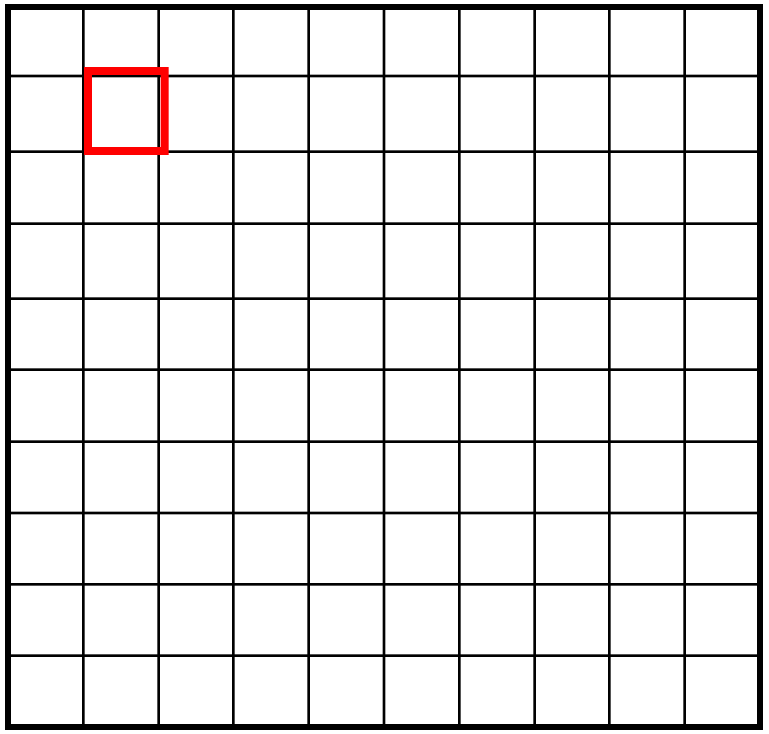
- Really important!
  - Enhance images
    - Denoise, resize, increase contrast, etc.
  - Extract information from images
    - Texture, edges, distinctive points, etc.
  - Detect patterns
    - Template matching
  - Deep Convolutional Networks

Moving Average In 2D

$F[x, y]$



$G[x, y]$



Slide credit: Steve Seitz

Moving Average In 2D

$F[x, y]$

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

$G[x, y]$

	0								

Slide credit: Steve Seitz

Moving Average In 2D

$F[x, y]$

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

$G[x, y]$

	0								

Slide credit: Steve Seitz

Moving Average In 2D

$F[x, y]$

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

$G[x, y]$

	0	10							

Slide credit: Steve Seitz

Moving Average In 2D

$F[x, y]$

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

$G[x, y]$

	0	10							

Slide credit: Steve Seitz

Moving Average In 2D

$F[x, y]$

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

$G[x, y]$

	0	10	20						

Slide credit: Steve Seitz

Moving Average In 2D

$F[x, y]$

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

$G[x, y]$

	0	10	20						

Slide credit: Steve Seitz

Moving Average In 2D

$F[x, y]$

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

$G[x, y]$

	0	10	20	30					

Slide credit: Steve Seitz

Moving Average In 2D

$F[x, y]$

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

$G[x, y]$

	0	10	20	30					

Slide credit: Steve Seitz

Moving Average In 2D

$F[x, y]$

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

$G[x, y]$

	0	10	20	30	30				

Slide credit: Steve Seitz

Moving Average In 2D

$F[x, y]$

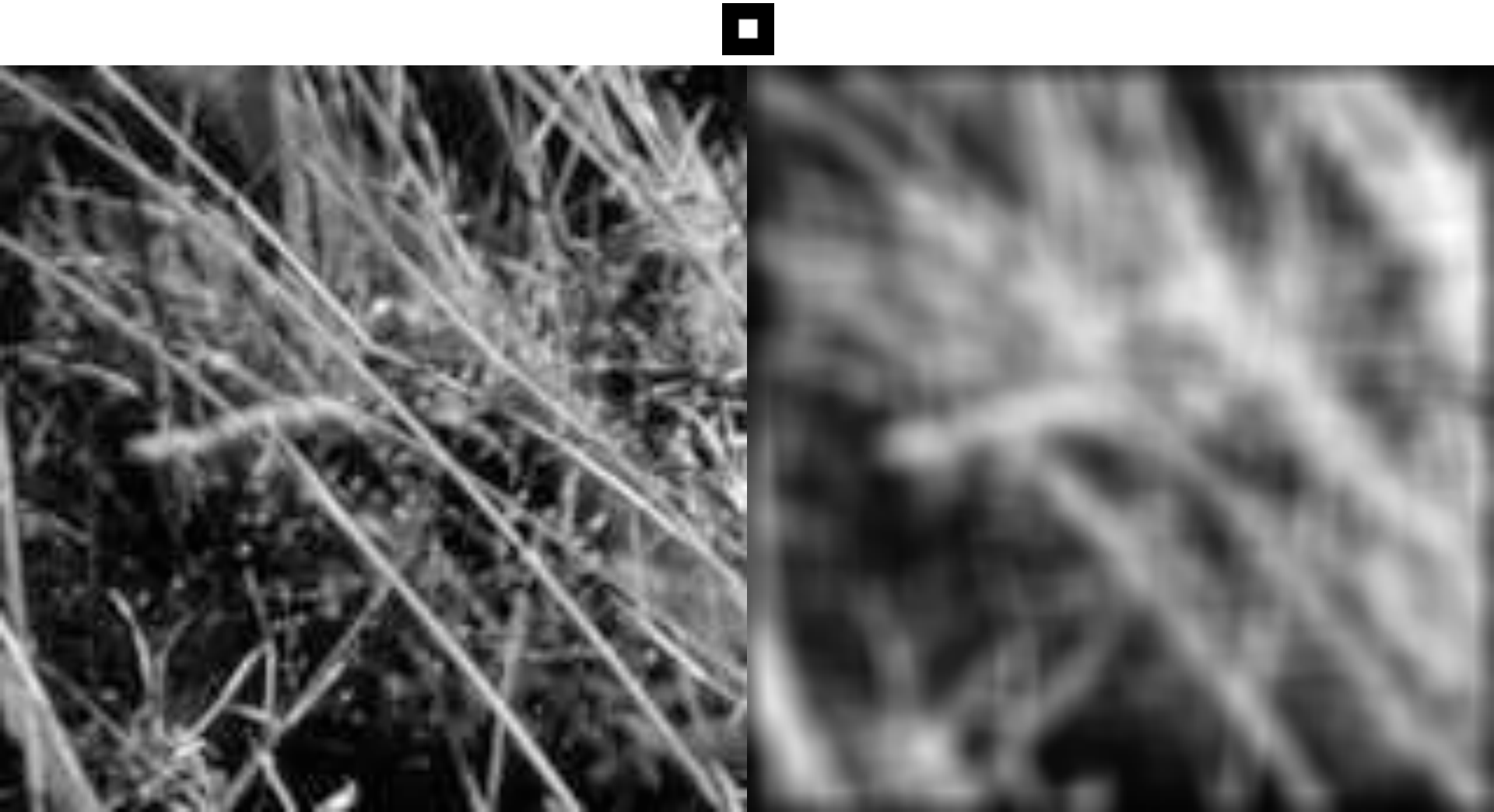
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	0	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	0	0	0	0	0	0	0
0	0	90	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

$G[x, y]$

	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	

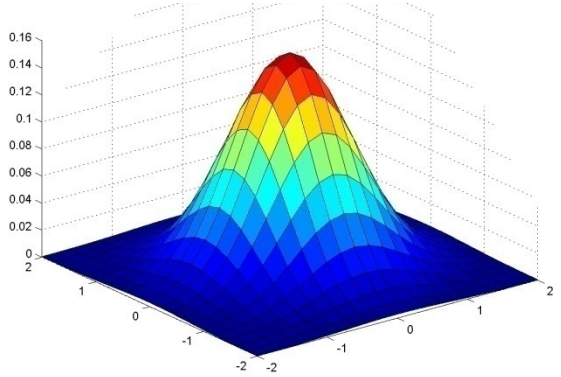
Slide credit: Steve Seitz

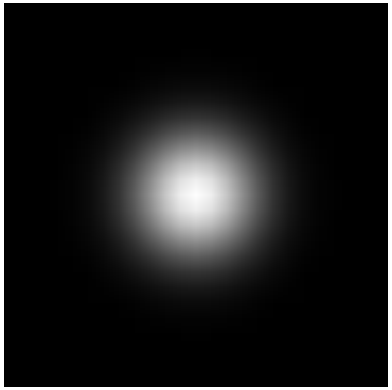
Smoothing with box filter



Important filter: Gaussian

- Weight contributions of neighboring pixels by nearness





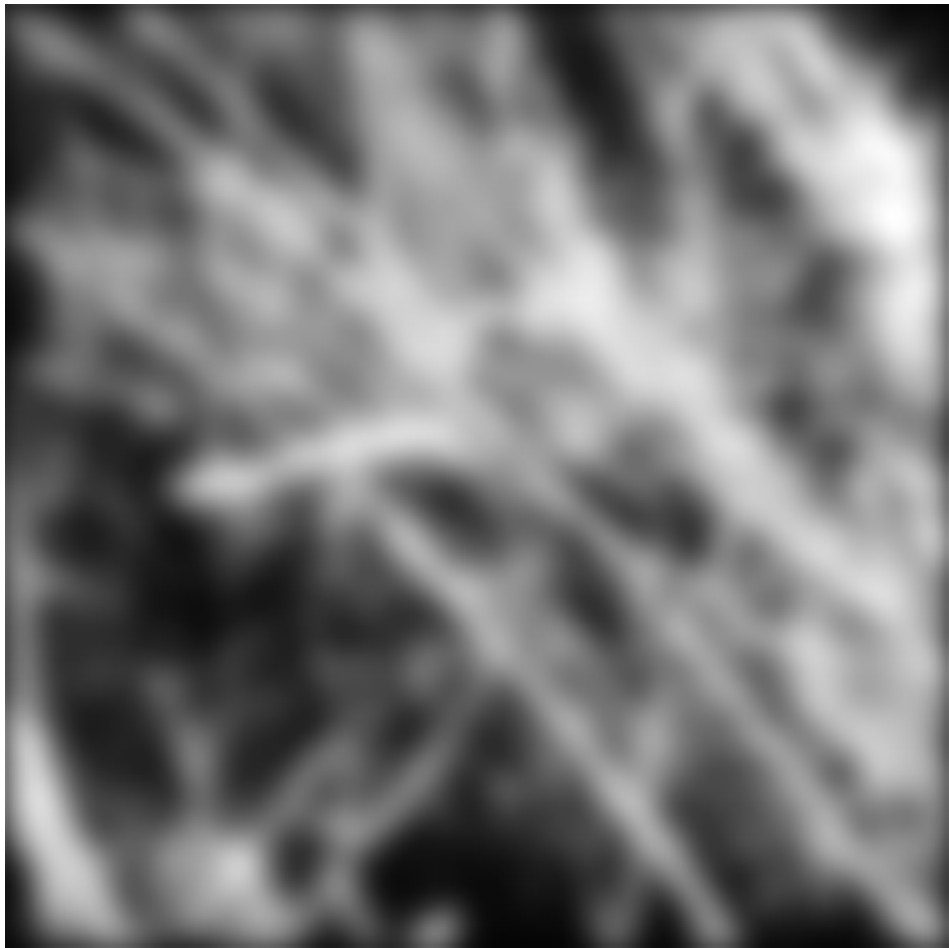
0.003	0.013	0.022	0.013	0.003
0.013	0.059	0.097	0.059	0.013
0.022	0.097	0.159	0.097	0.022
0.013	0.059	0.097	0.059	0.013
0.003	0.013	0.022	0.013	0.003

5 x 5, $\sigma = 1$

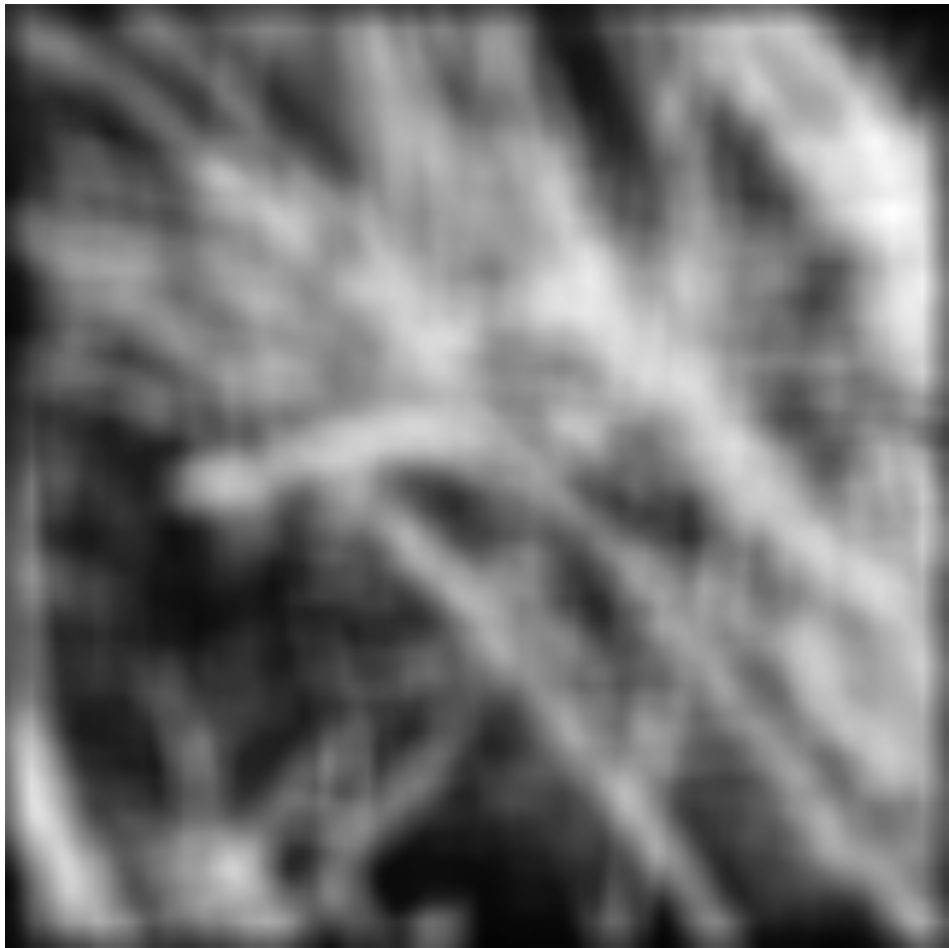
$$G_\sigma = \frac{1}{2\pi\sigma^2} e^{-\frac{(x^2+y^2)}{2\sigma^2}}$$

Slide credit: Christopher Rasmussen

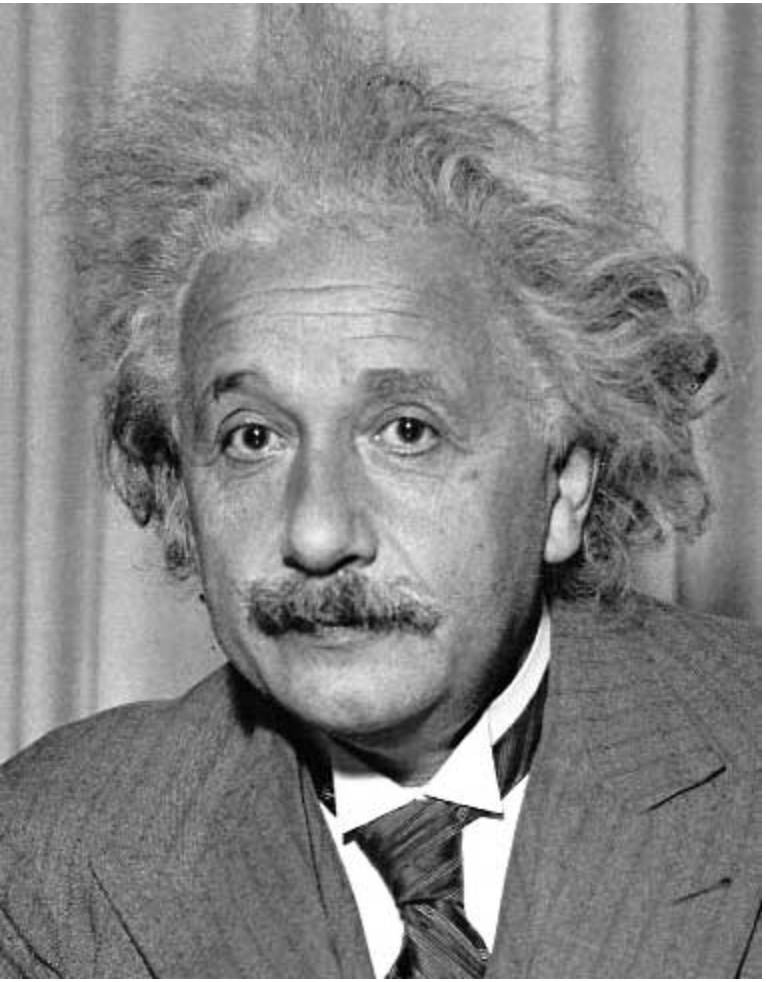
Smoothing with Gaussian filter



Smoothing with box filter

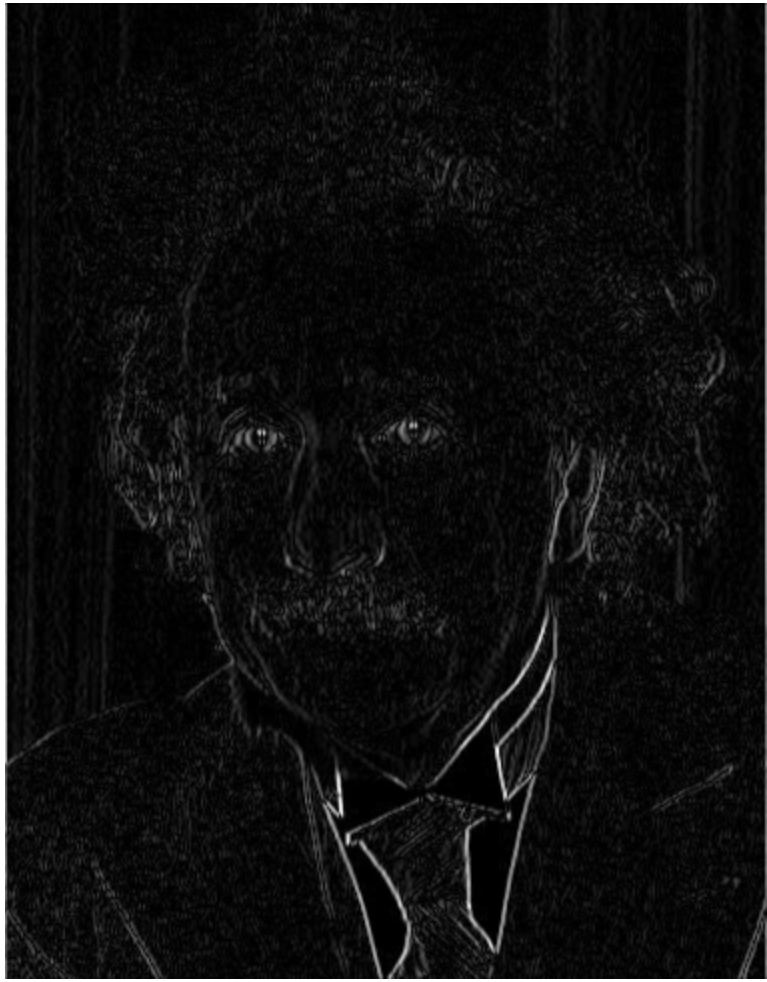


Other filters



1	0	-1
2	0	-2
1	0	-1

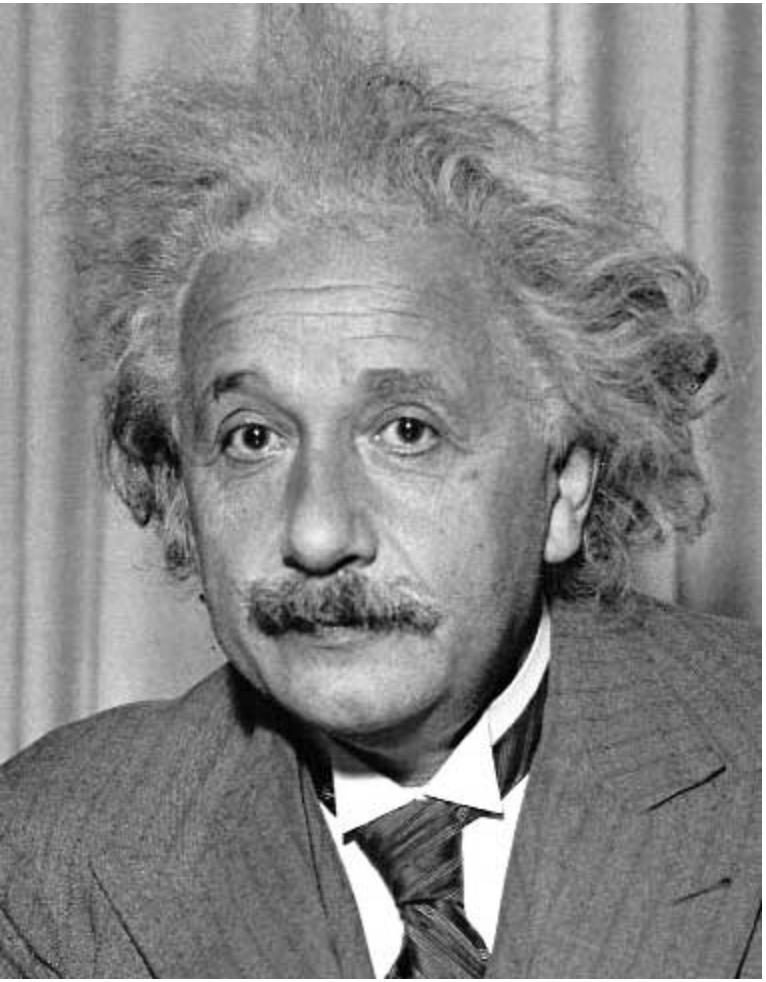
Sobel



Vertical Edge

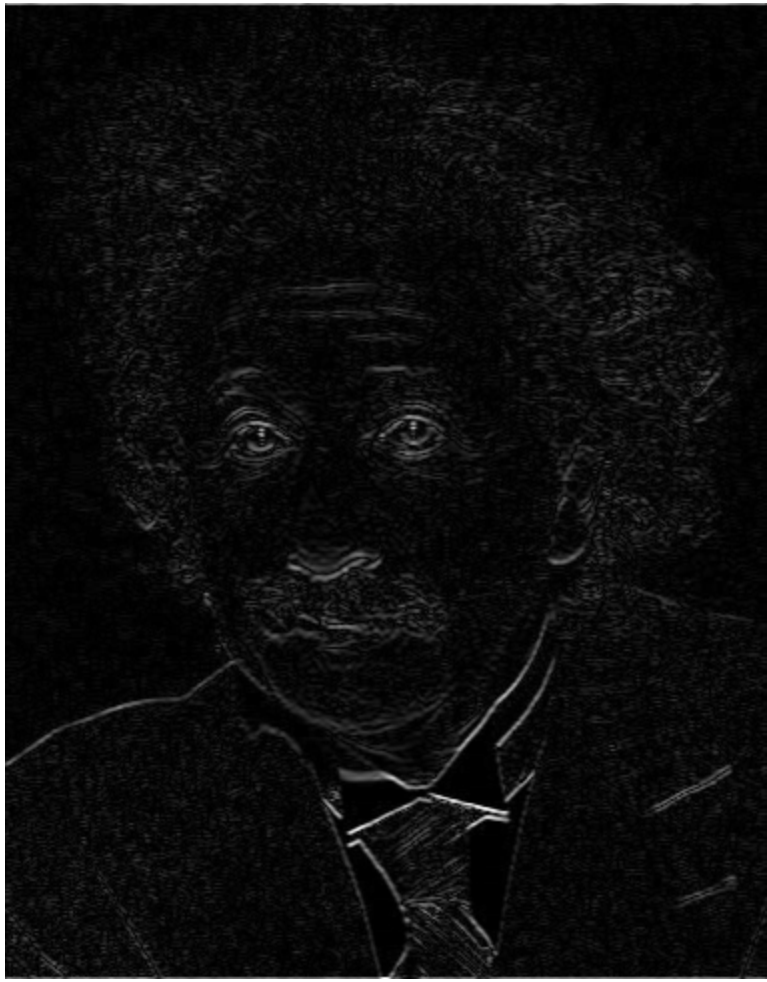
(absolute value)

Other filters



1	2	1
0	0	0
-1	-2	-1

Sobel



Horizontal Edge

(absolute value)

Gaussian filters

- Remove “high-frequency” components from the image (low-pass filter)
  - Images become more smooth
- Convolution with self is another Gaussian
  - So can smooth with small-width kernel, repeat, and get same result as larger-width kernel would have
  - Convolving two times with Gaussian kernel of width σ is same as convolving once with kernel of width $\sigma\sqrt{2}$
- *Separable* kernel
  - Factors into product of two 1D Gaussians

Source: K. Grauman

Separability of the Gaussian filter

$$G_\sigma(x,y) = \frac{1}{2\pi\sigma^2}\exp^{-\frac{x^2+y^2}{2\sigma^2}}$$

$$= \left(\frac{1}{\sqrt{2\pi}\sigma}\exp^{-\frac{x^2}{2\sigma^2}}\right)\left(\frac{1}{\sqrt{2\pi}\sigma}\exp^{-\frac{y^2}{2\sigma^2}}\right)$$

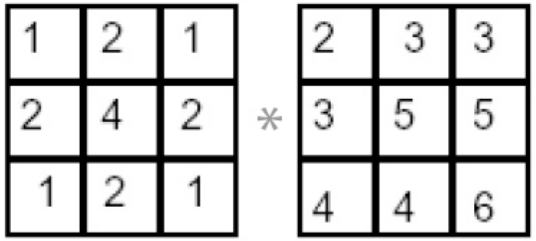
The 2D Gaussian can be expressed as the product of two functions, one a function of x and the other a function of y

In this case, the two functions are the (identical) 1D Gaussian

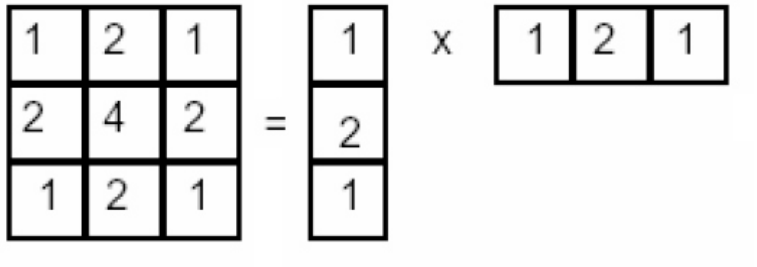
Source: D. Lowe

Separability example

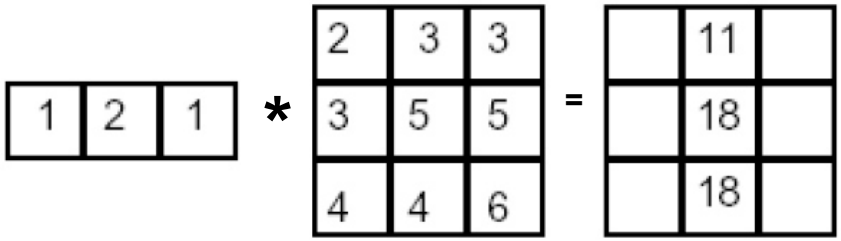
2D convolution
(center location only)



The filter factors
into a product of 1D
filters:

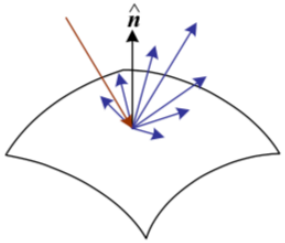


Perform convolution
along rows:

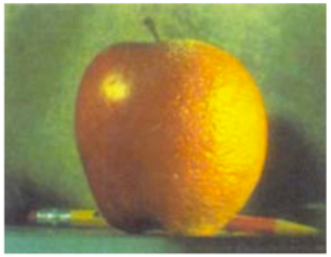


Followed by convolution
along the remaining column:

Source: K. Grauman

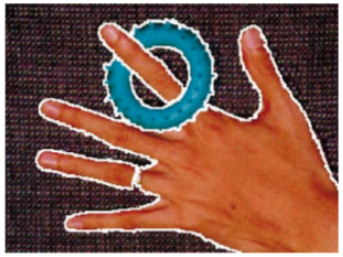


2. Image Formation



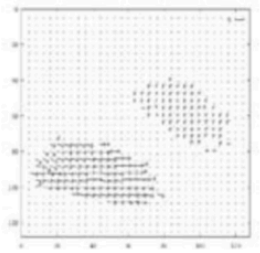
3. Image Processing

4. Features

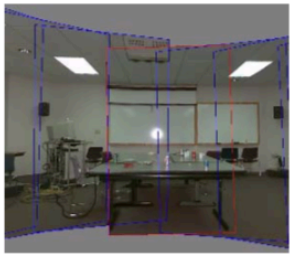


5. Segmentation

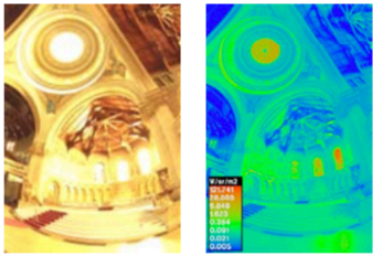
6-7. Structure from Motion



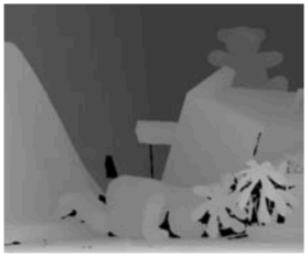
8. Motion



9. Stitching

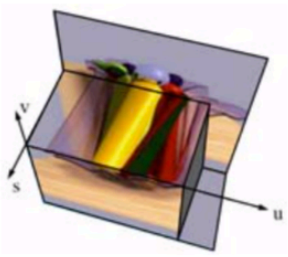


10. Computational Photography



11. Stereo

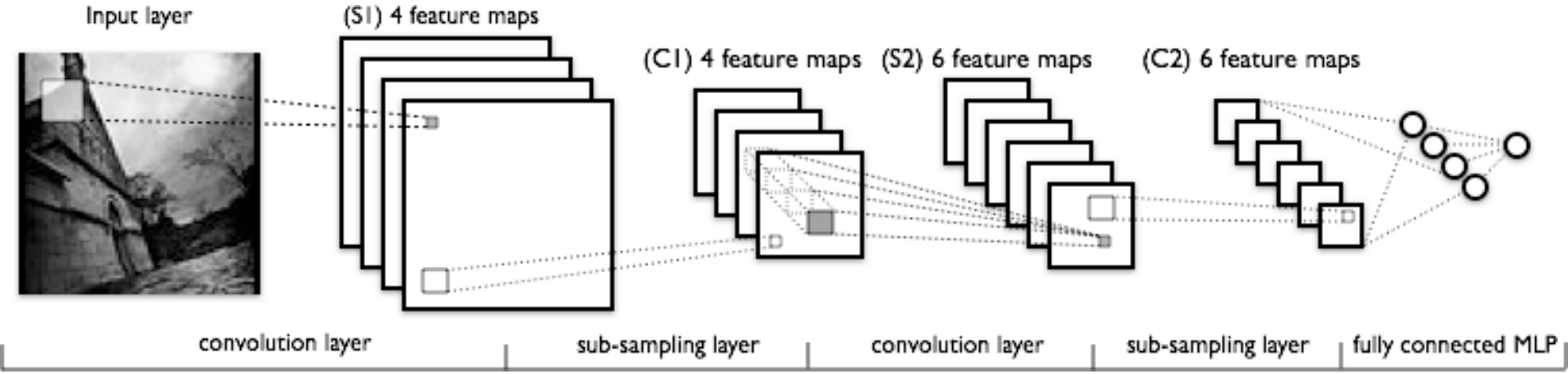
12. 3D Shape



13. Image-based Rendering

14. Recognition

Convolutional Neural Networks



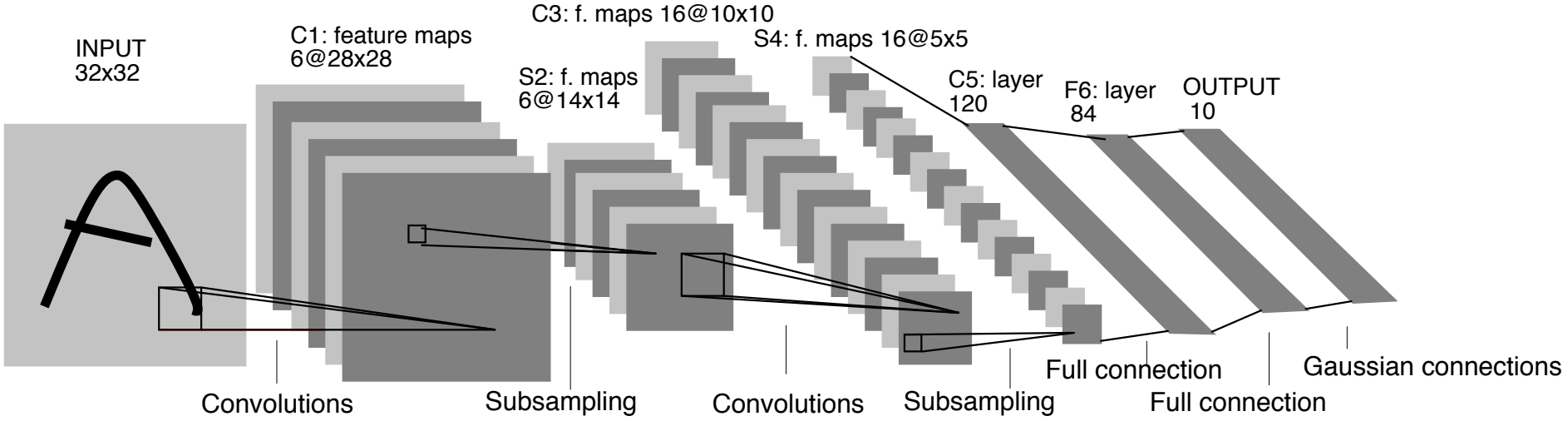
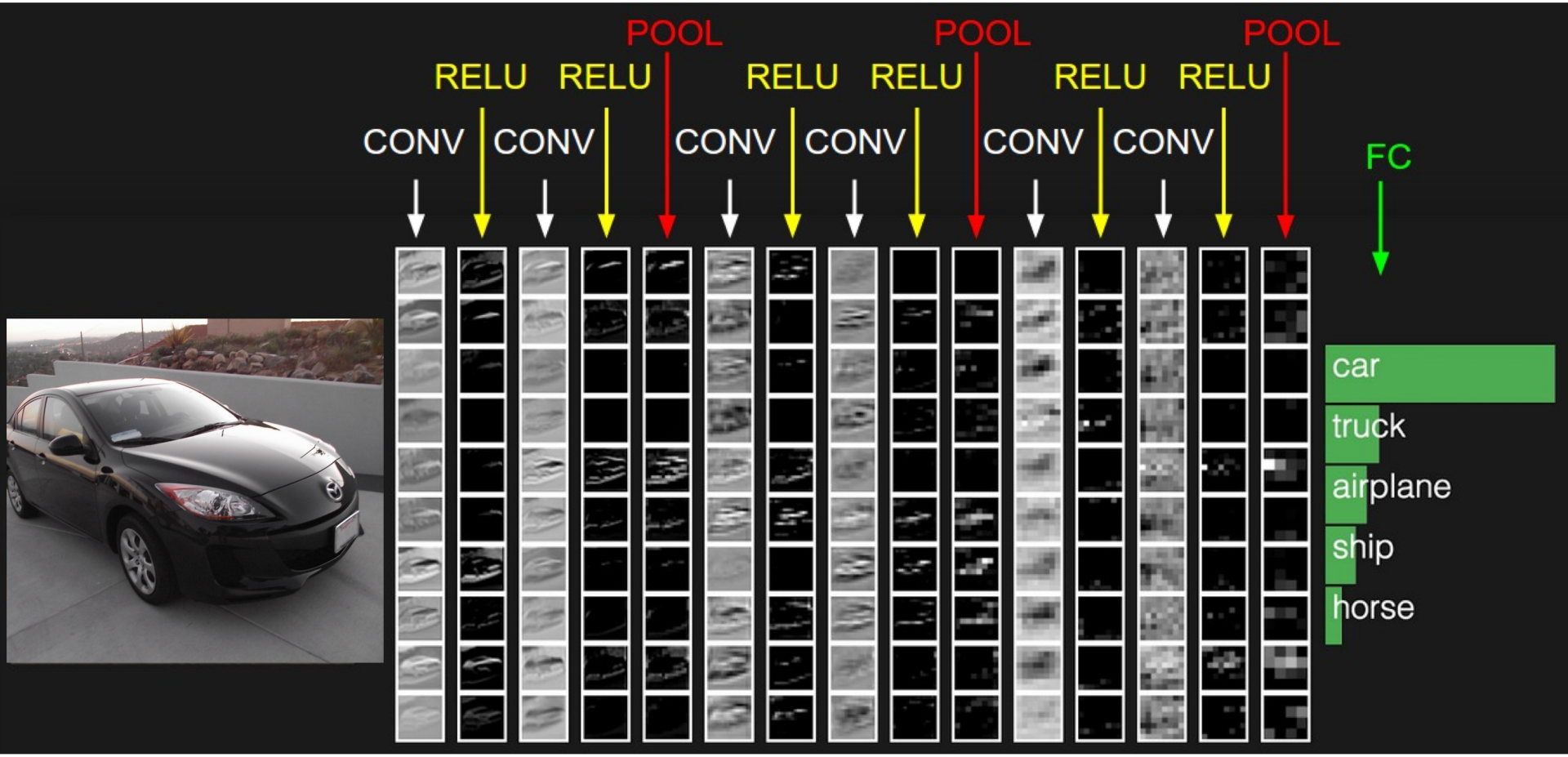


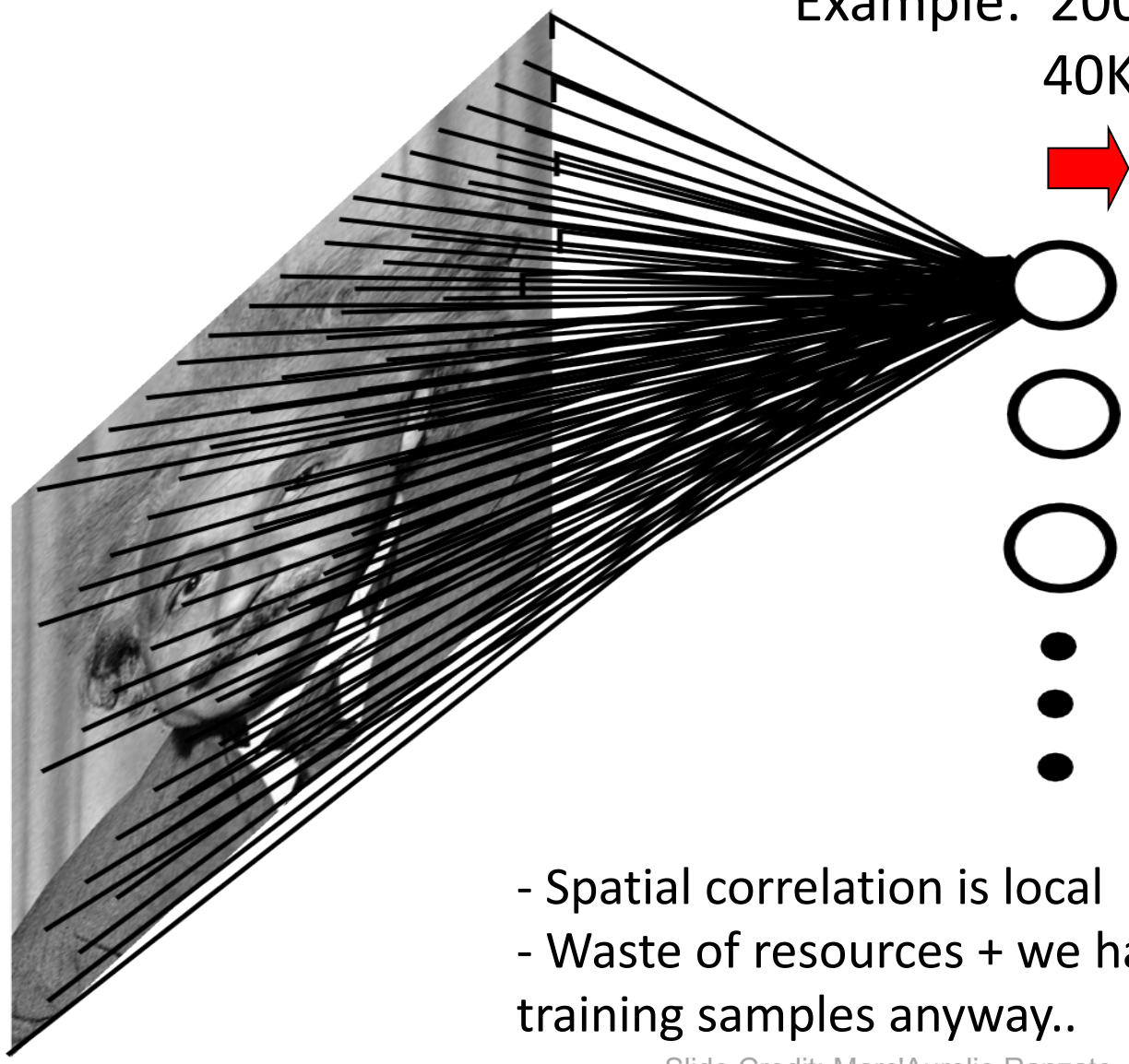
Image Credit: Yann LeCun, Kevin Murphy

preview:



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

Fully Connected Layer



Example: 200x200 image

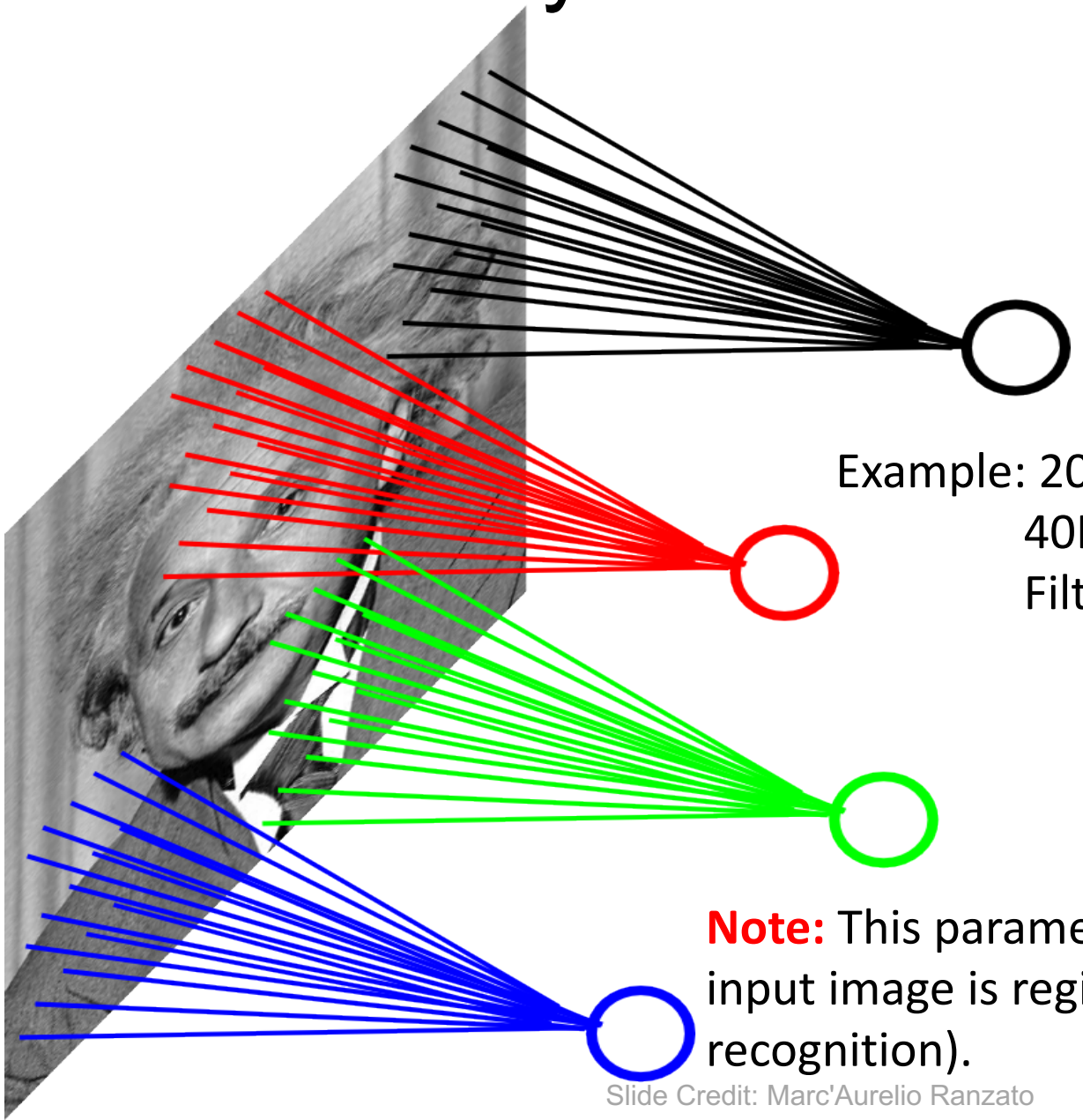
40K hidden units

~2B parameters!!!

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

Slide Credit: Marc'Aurelio Ranzato

Locally Connected Layer

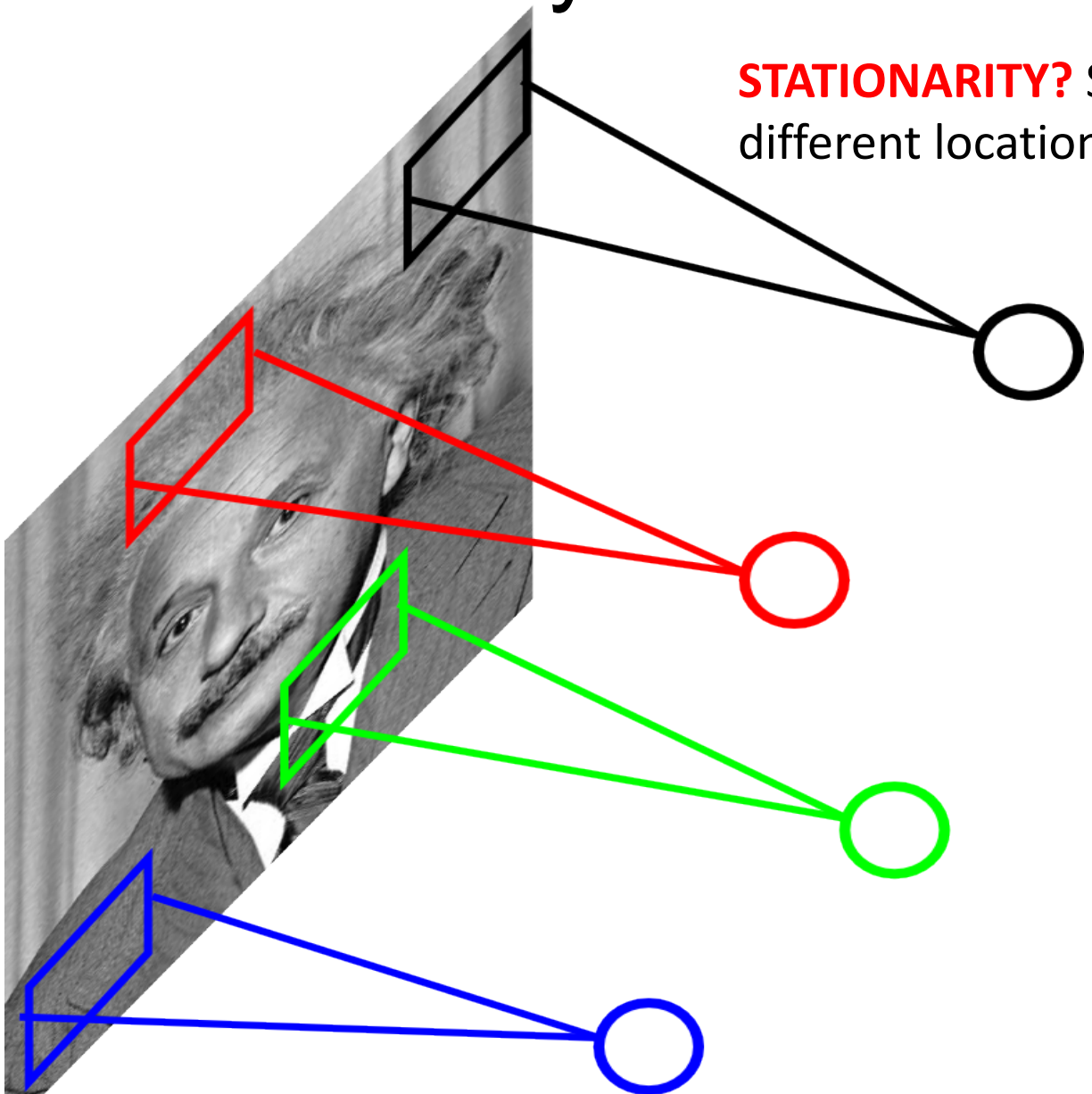


Example: 200x200 image
40K hidden units
Filter size: 10x10
4M parameters

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

Slide Credit: Marc'Aurelio Ranzato

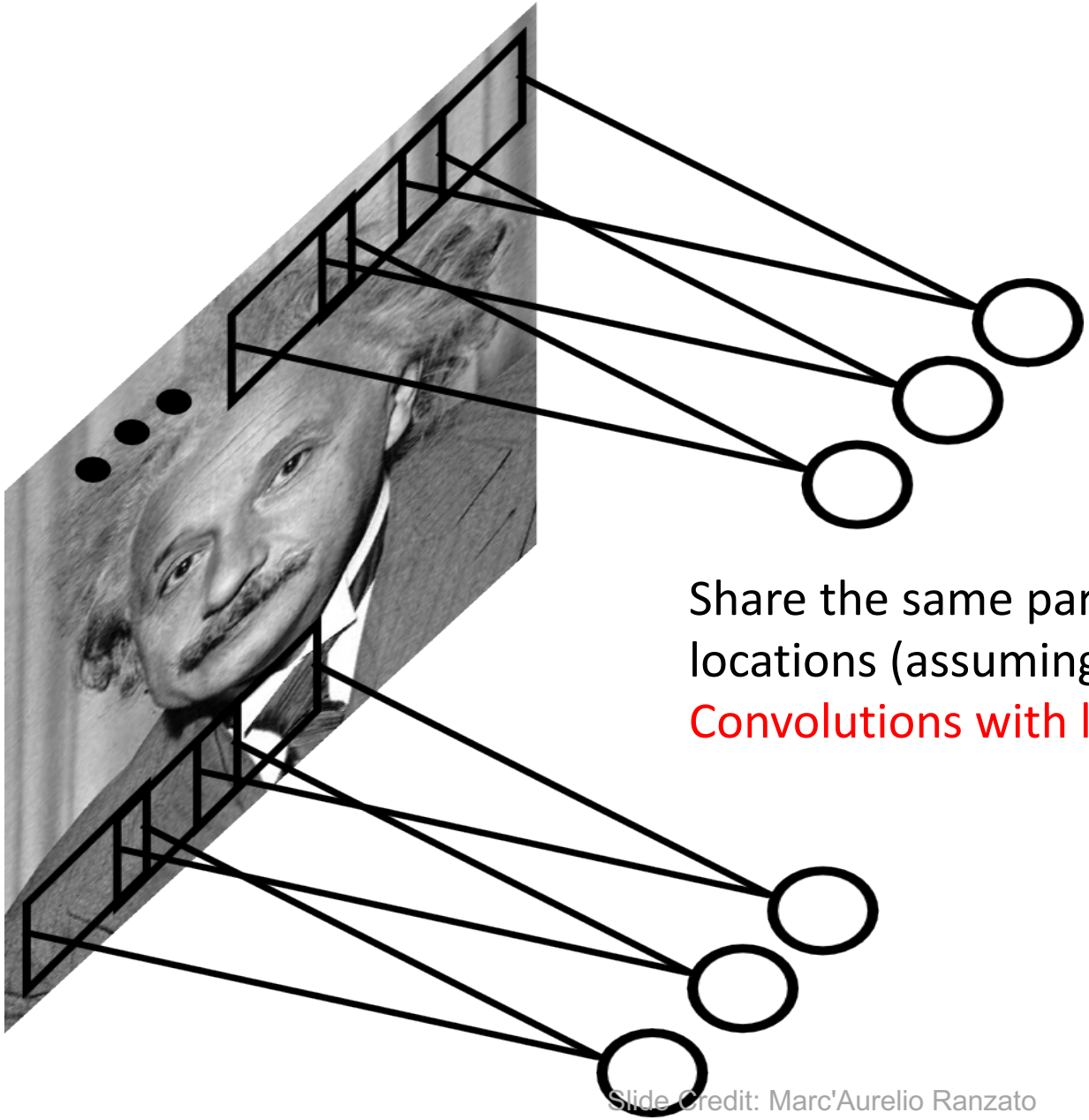
Locally Connected Layer



STATIONARITY? Statistics is similar at different locations

Slide Credit: Marc'Aurelio Ranzato

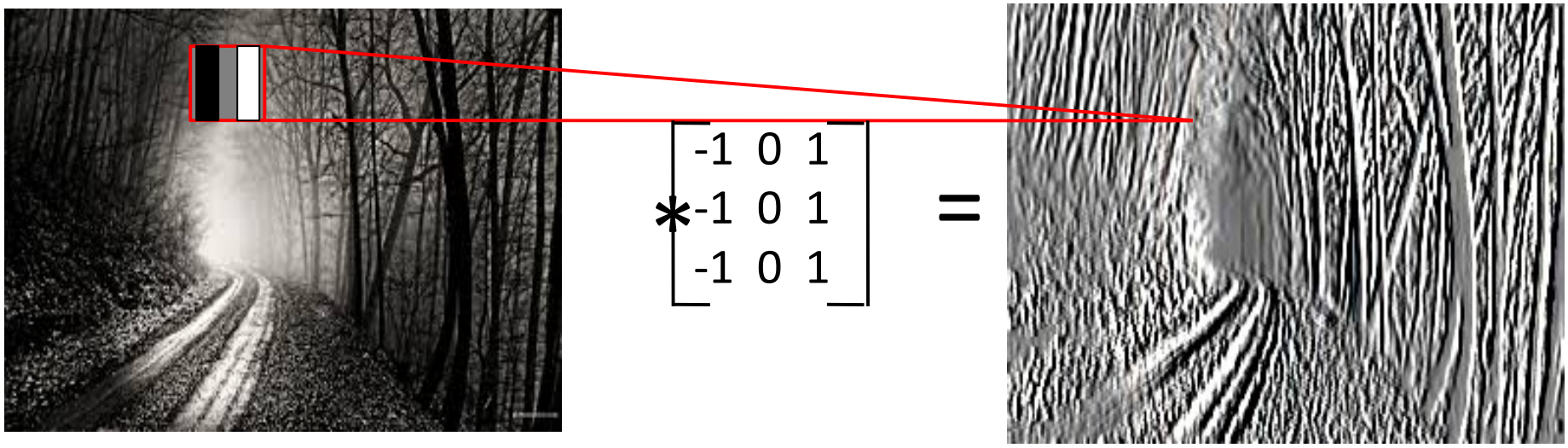
Convolutional Layer



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

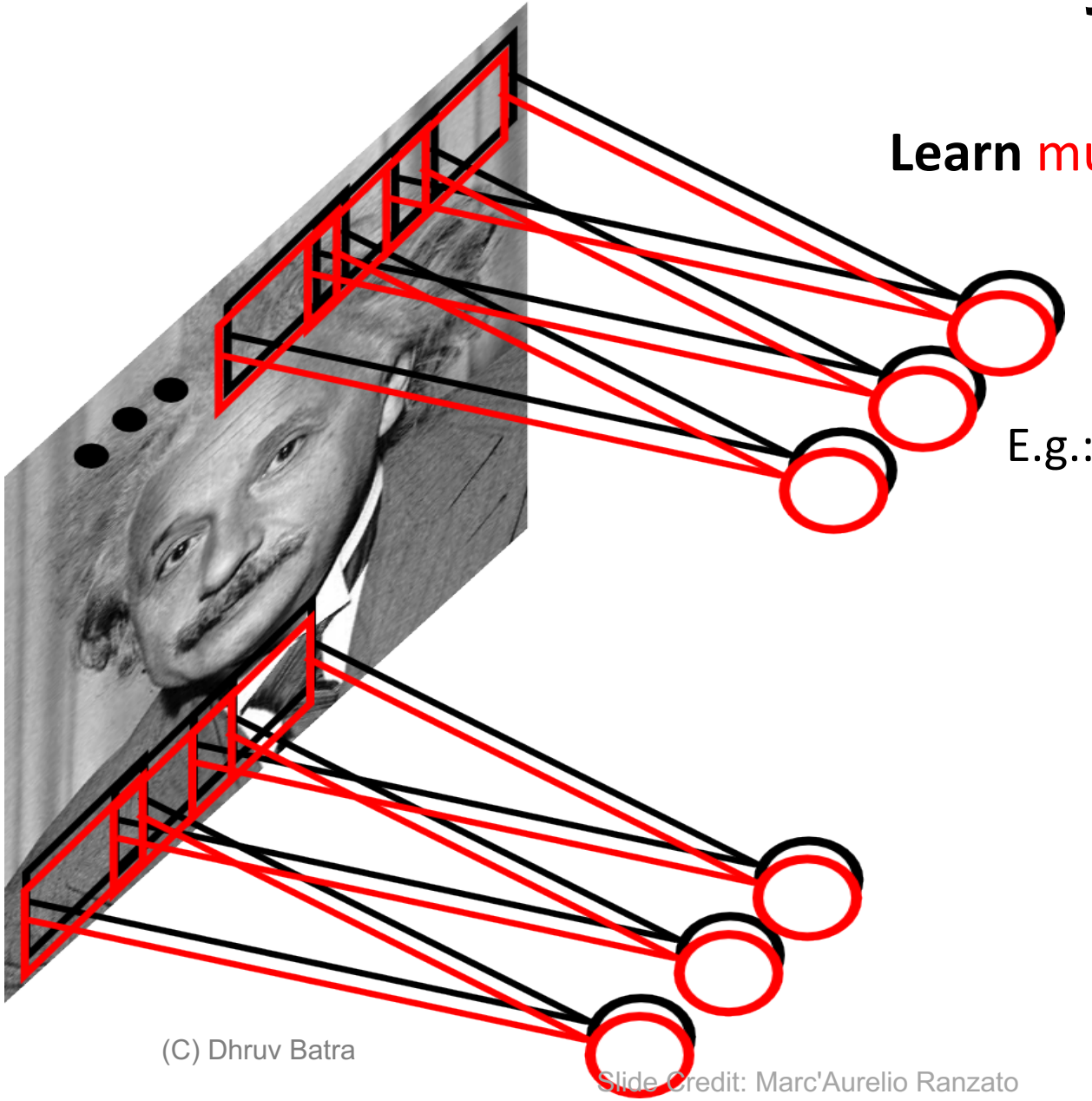
Slide Credit: Marc'Aurelio Ranzato

Convolutional Layer



Slide Credit: Marc'Aurelio Ranzato

Convolutional Layer



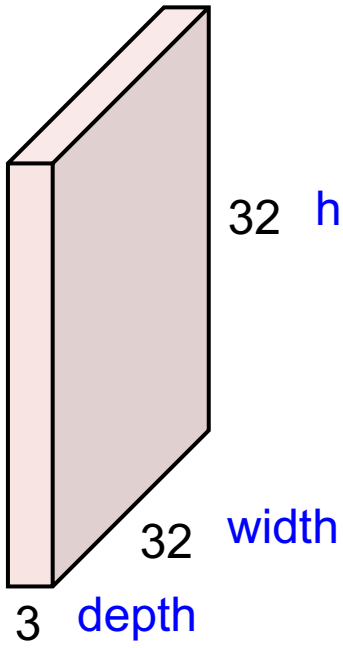
Learn multiple filters.

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

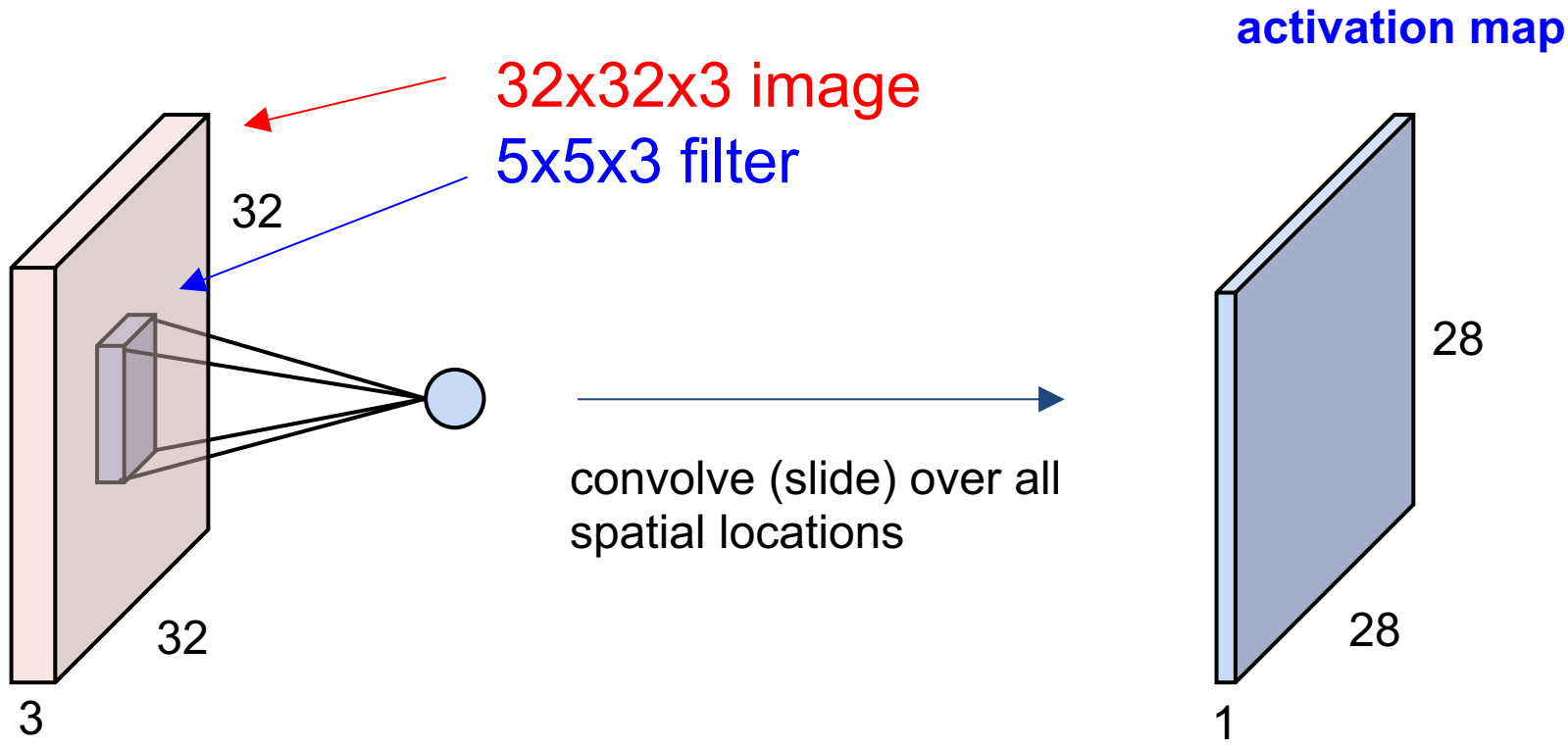
Slide Credit: Marc'Aurelio Ranzato

Convolution Layer

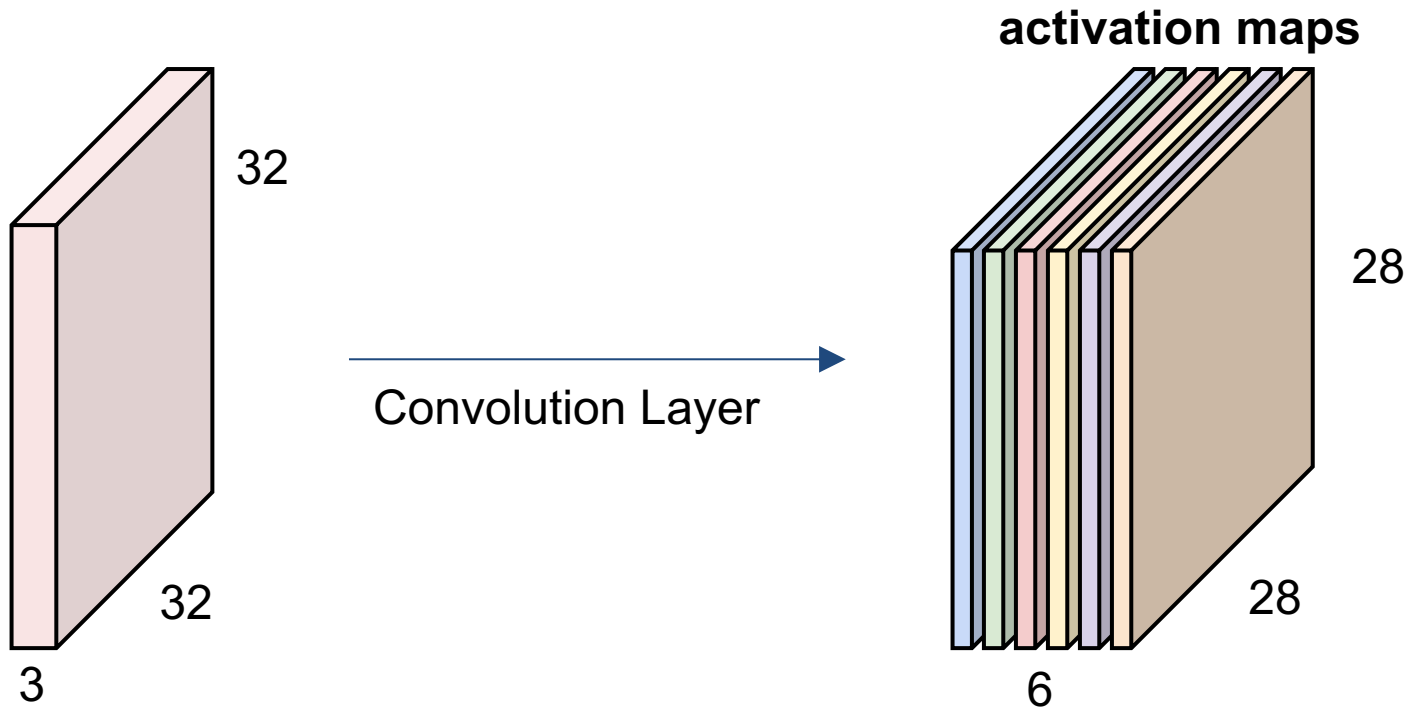
32x32x3 image -> preserve spatial structure



Convolution Layer

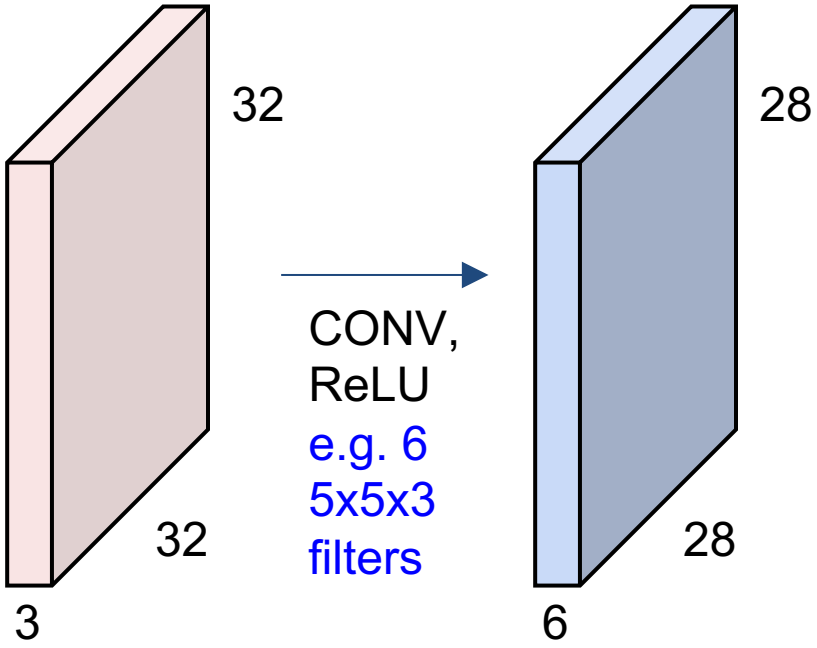


Multiple filters: if we have 6 5x5 filters, we'll get 6 separate activation maps:

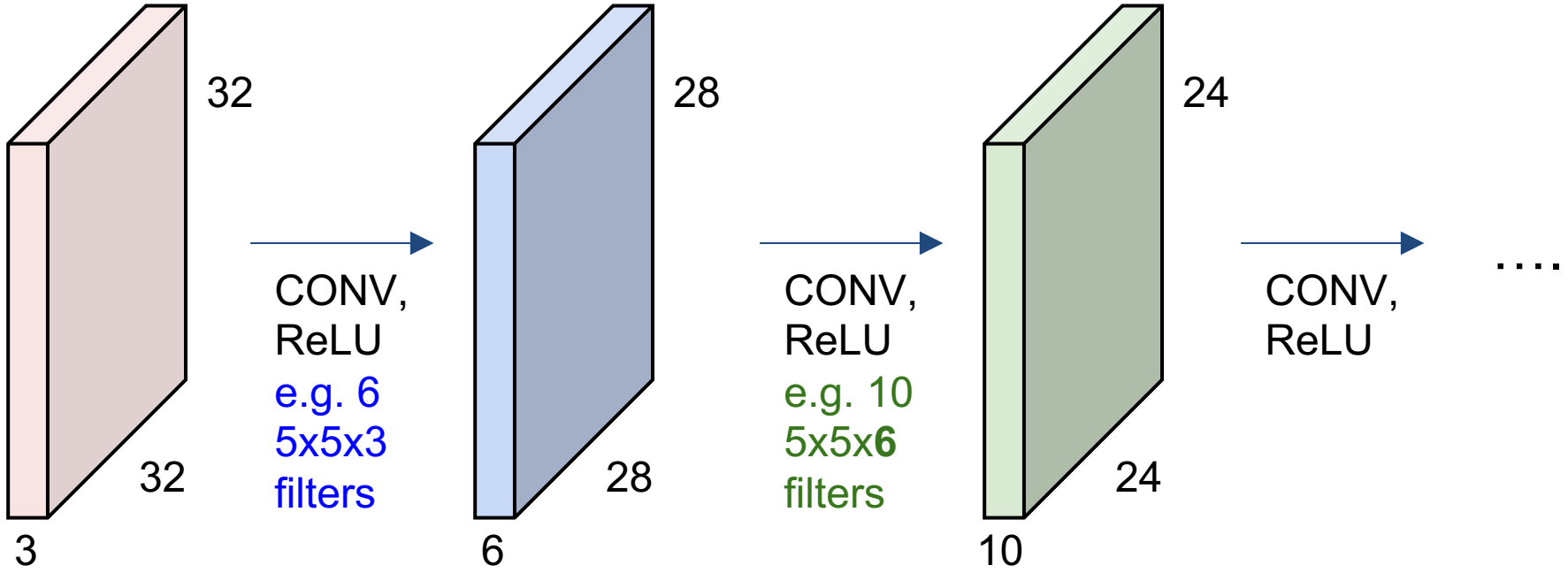


We stack these up to get a "new image" of size 28x28x6!

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



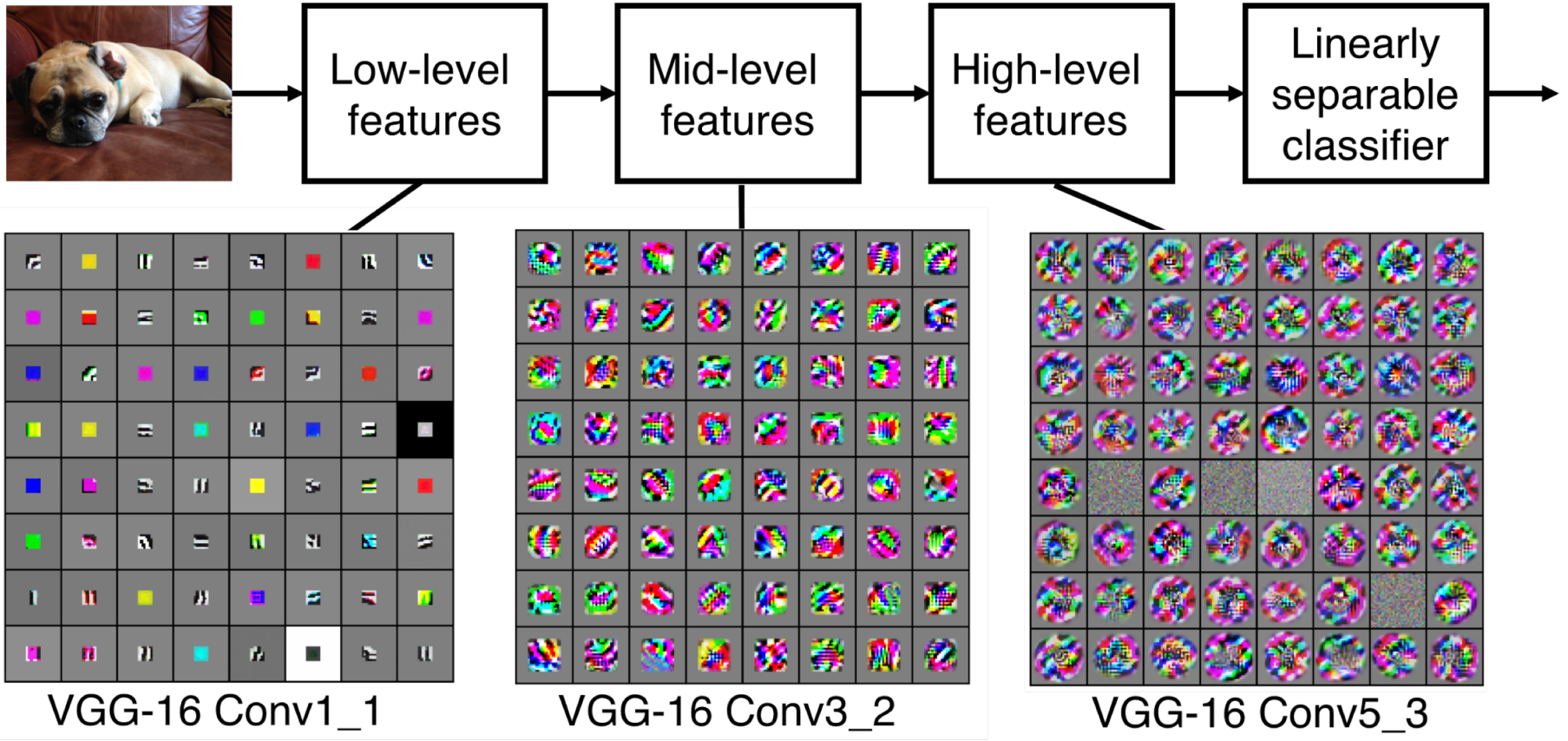
Preview: ConvNet is a sequence of Convolutional Layers, interspersed with activation functions



Preview

[Zeiler and Fergus 2013]

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



Visualizing Learned Filters

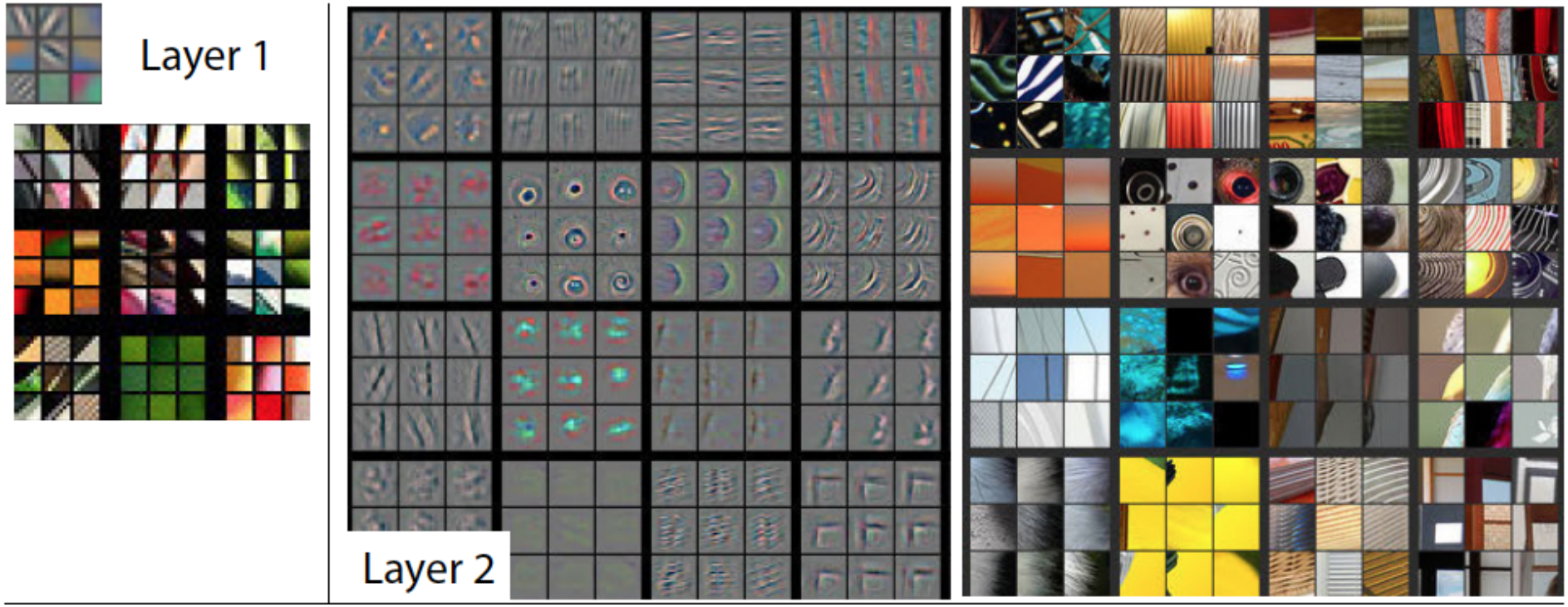
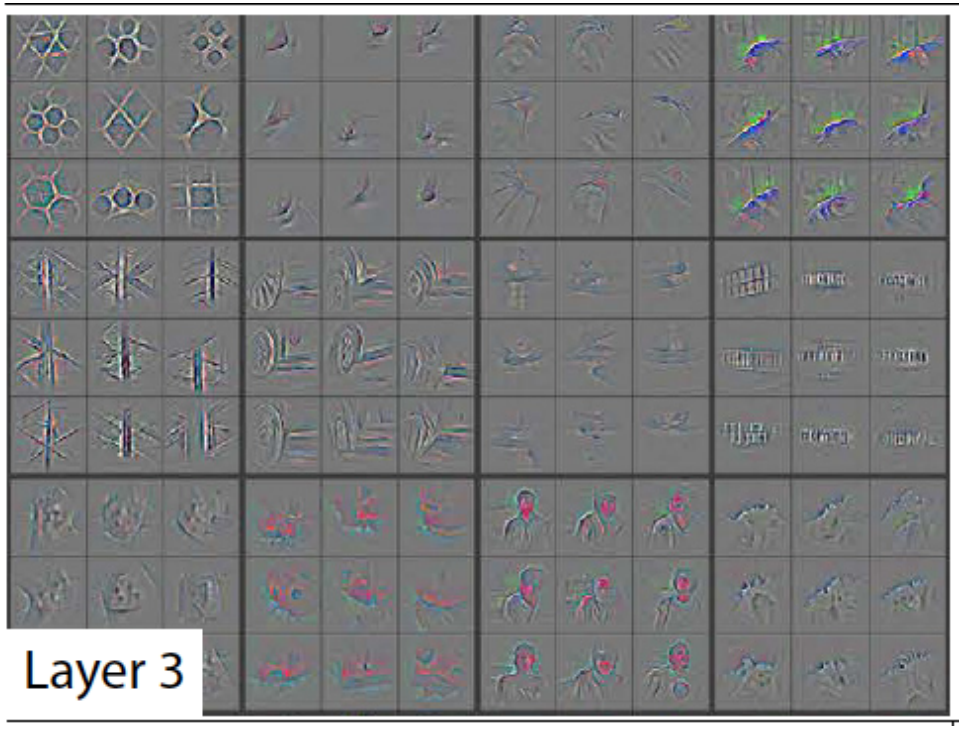


Figure Credit: [Zeiler & Fergus ECCV14]

Visualizing Learned Filters



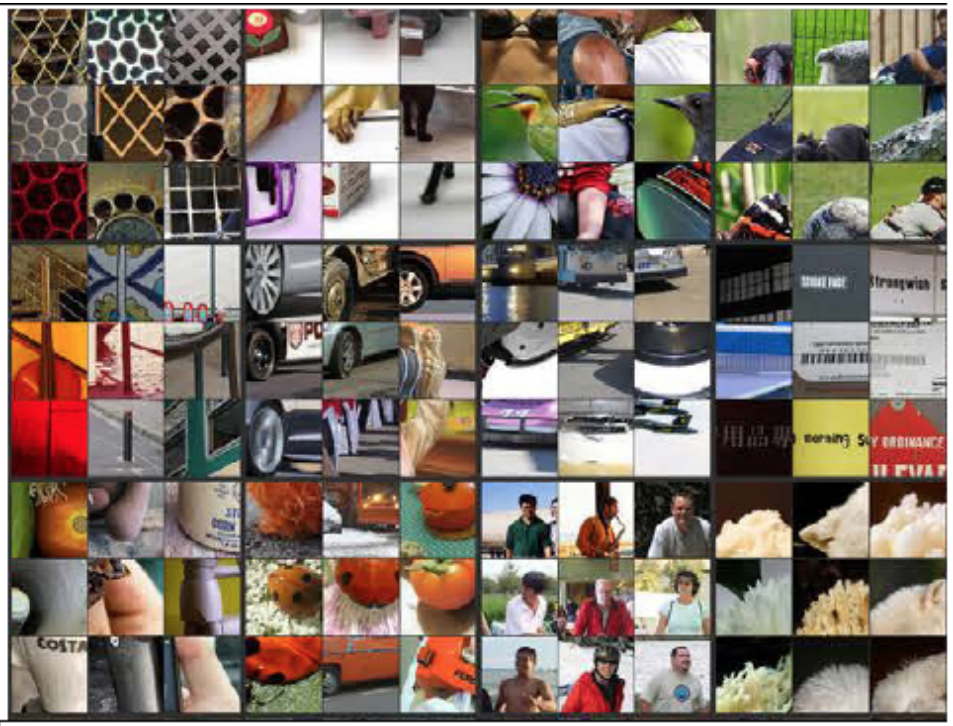
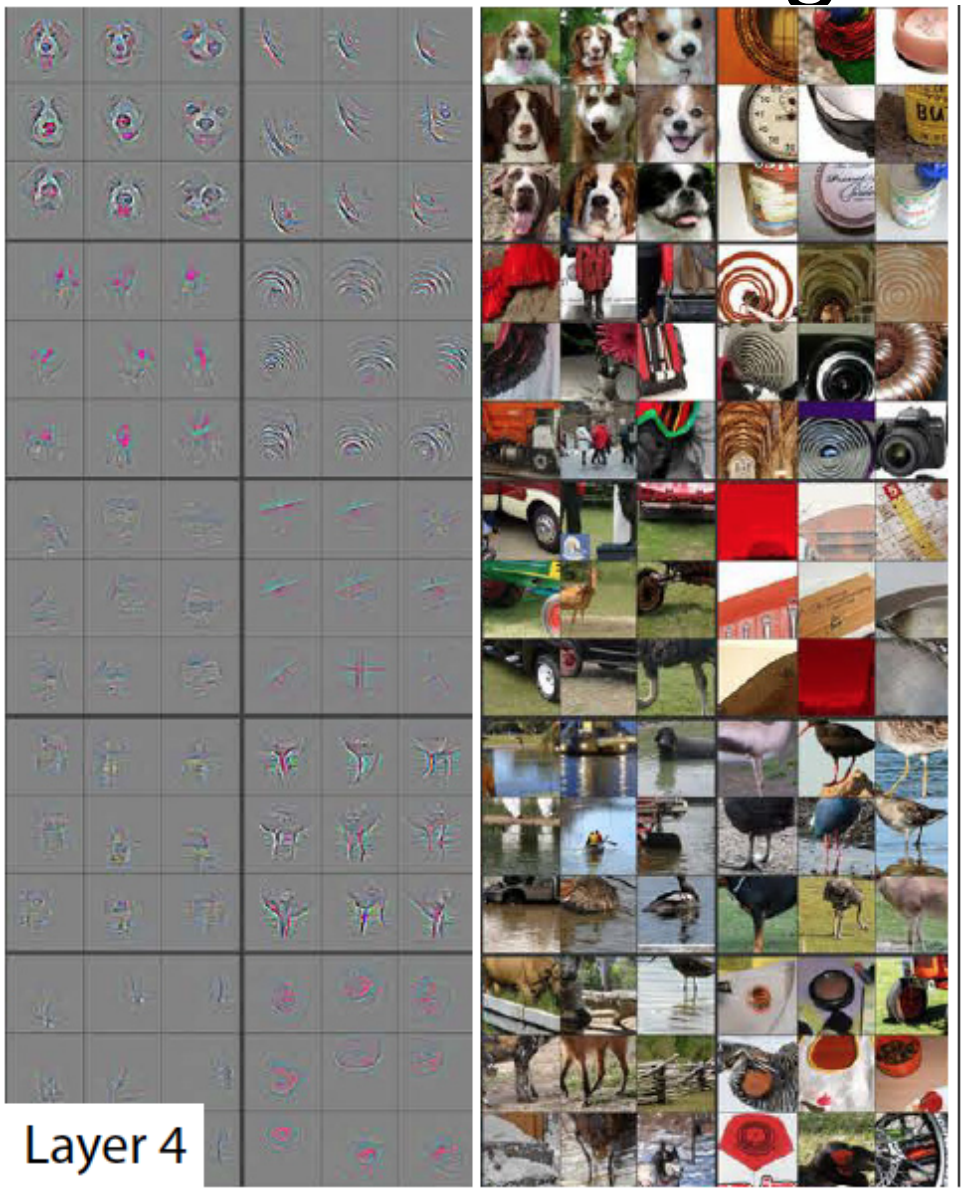


Figure Credit: [Zeiler & Fergus ECCV14]

Visualizing Learned Filters



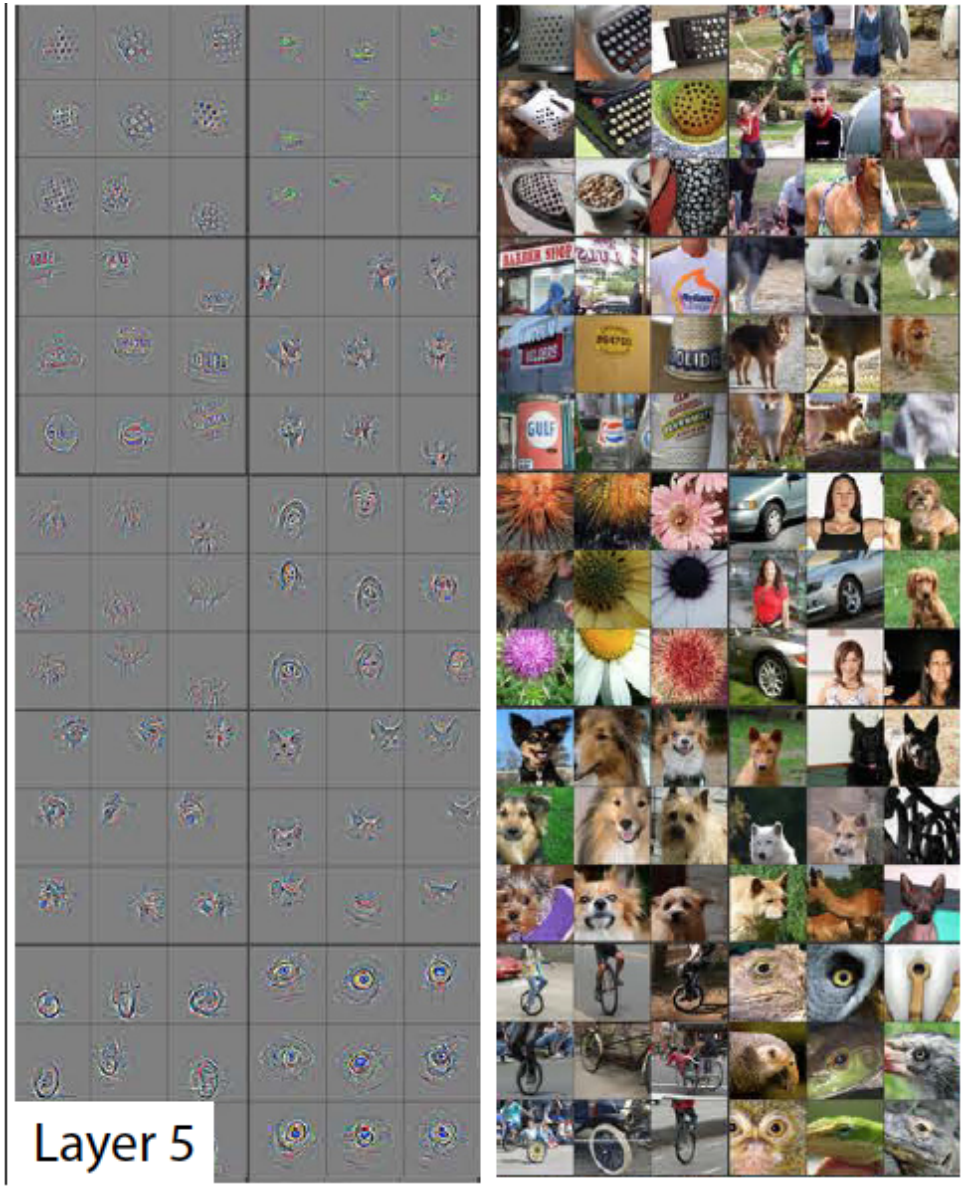
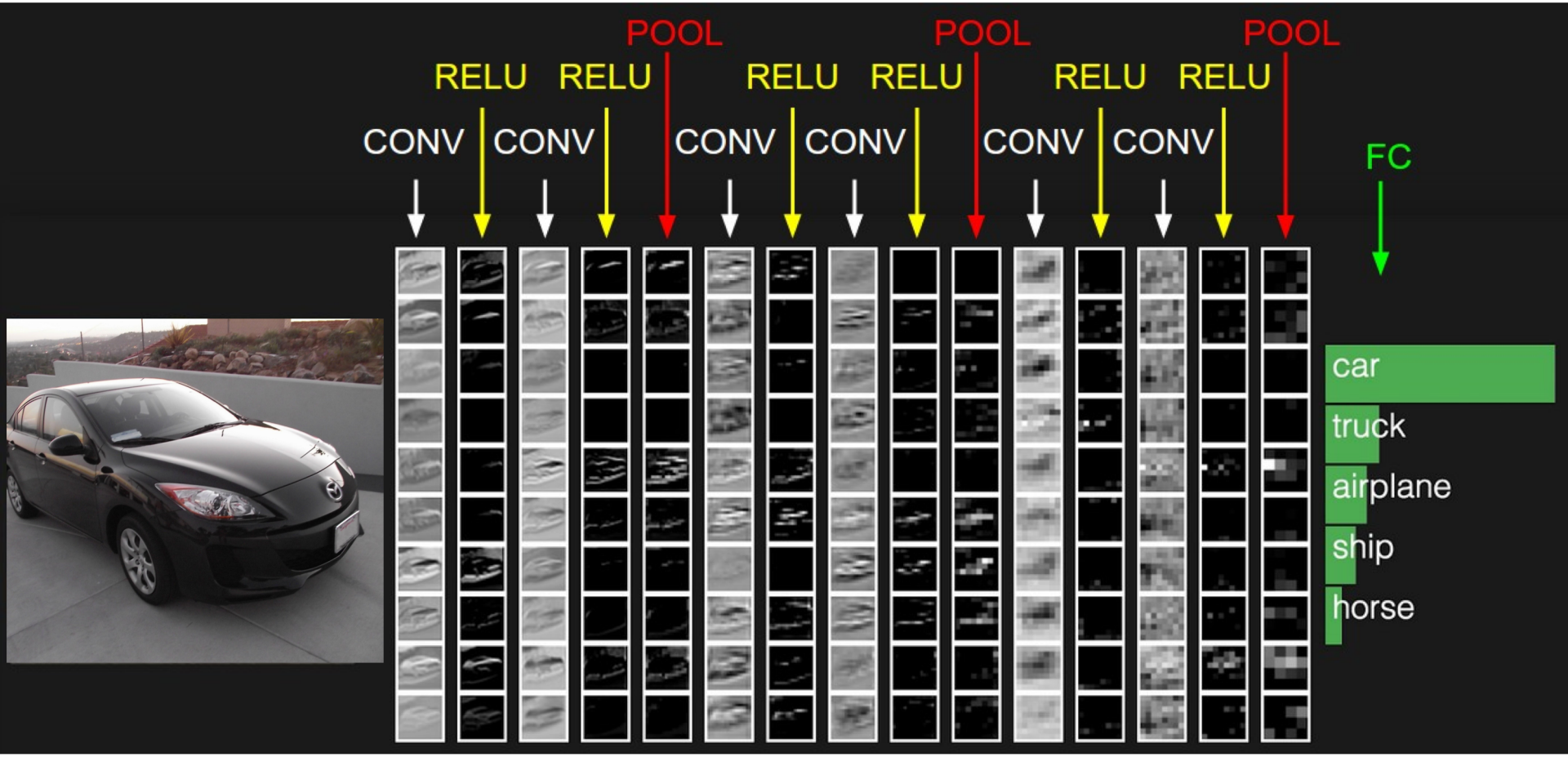


Figure Credit: [Zeiler & Fergus ECCV14]

two more layers to go: POOL/FC

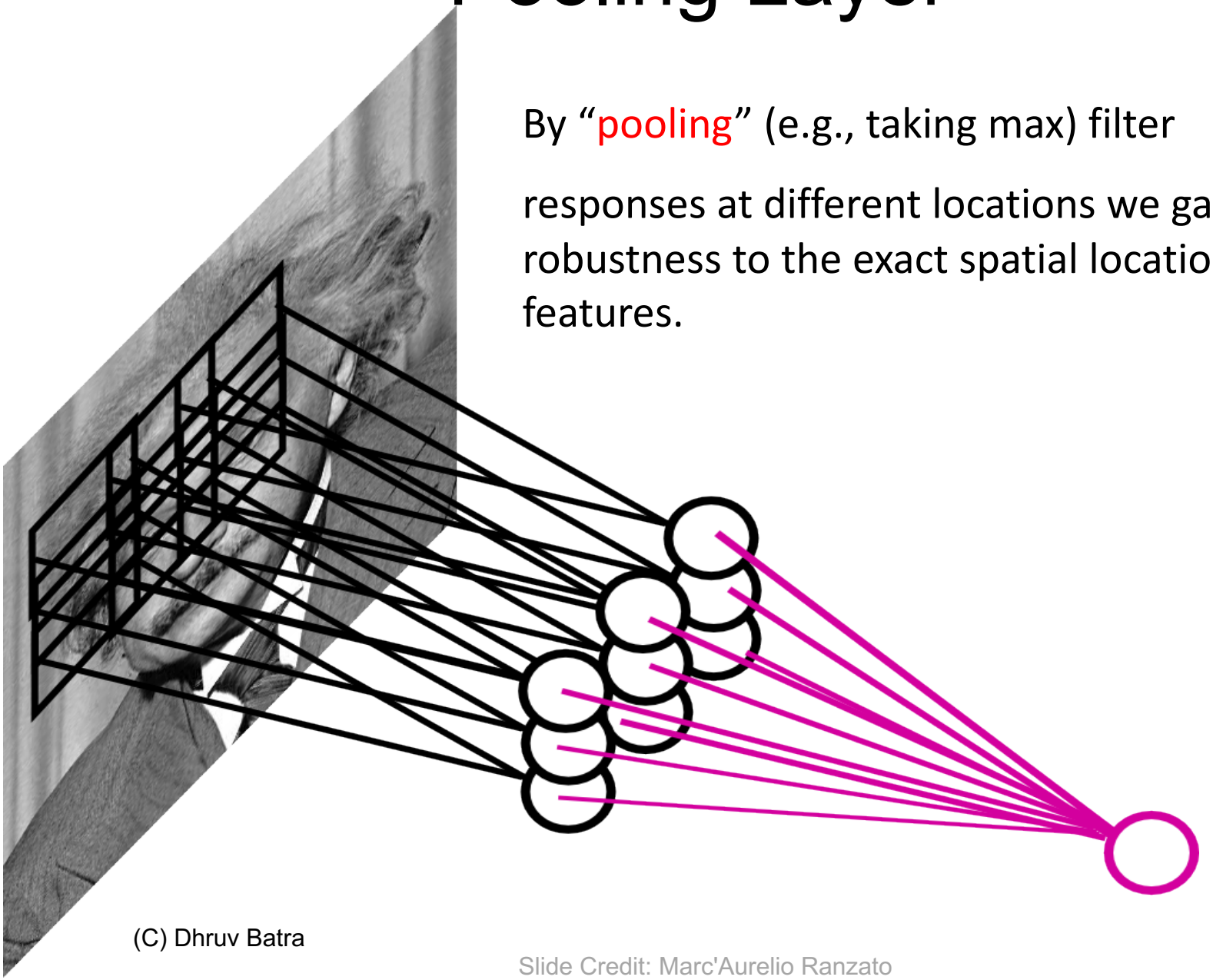


Slide Credit: Fei-Fei Li, Justin Johnson, Serena Yeung, CS 231n

Pooling Layer

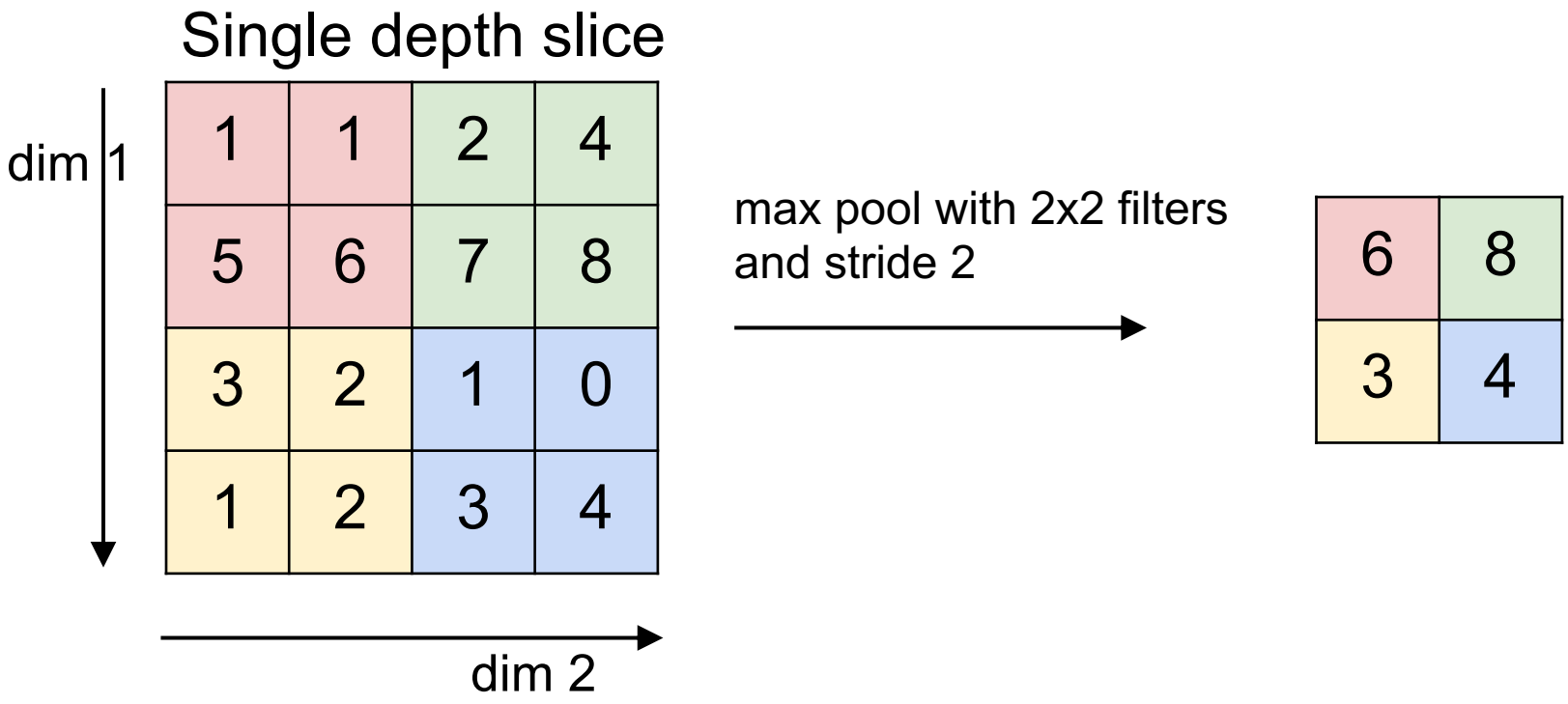
By “pooling” (e.g., taking max) filter

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



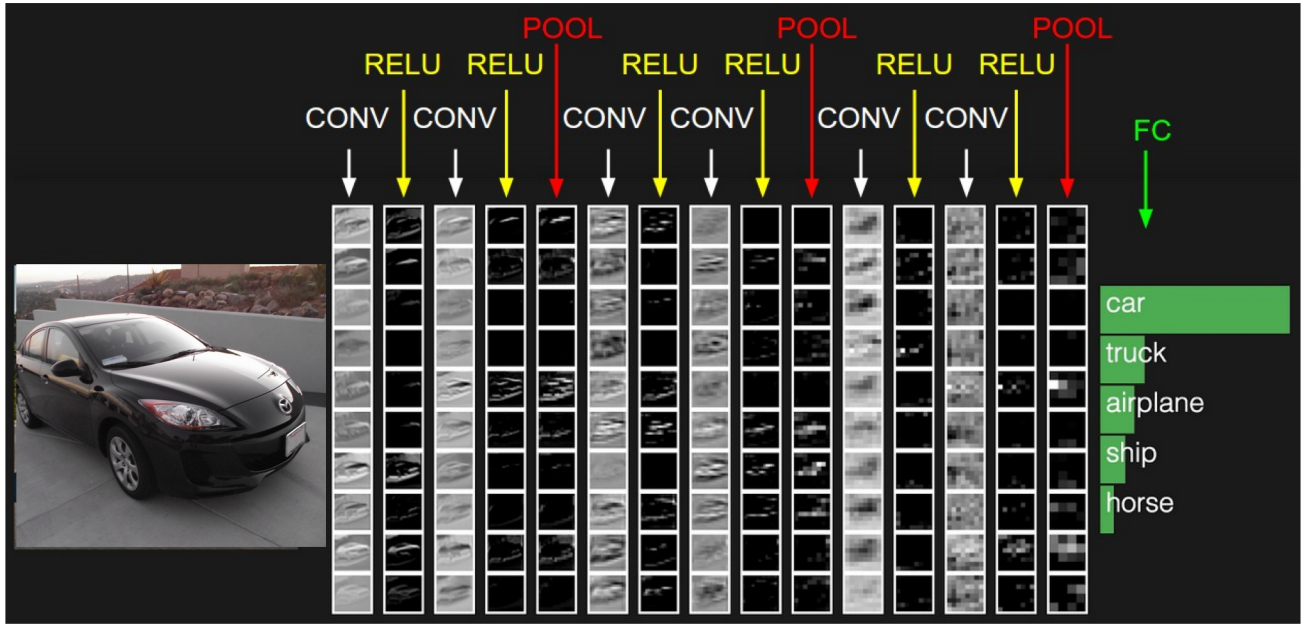
Slide Credit: Marc'Aurelio Ranzato

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 ordinary Neural Networks



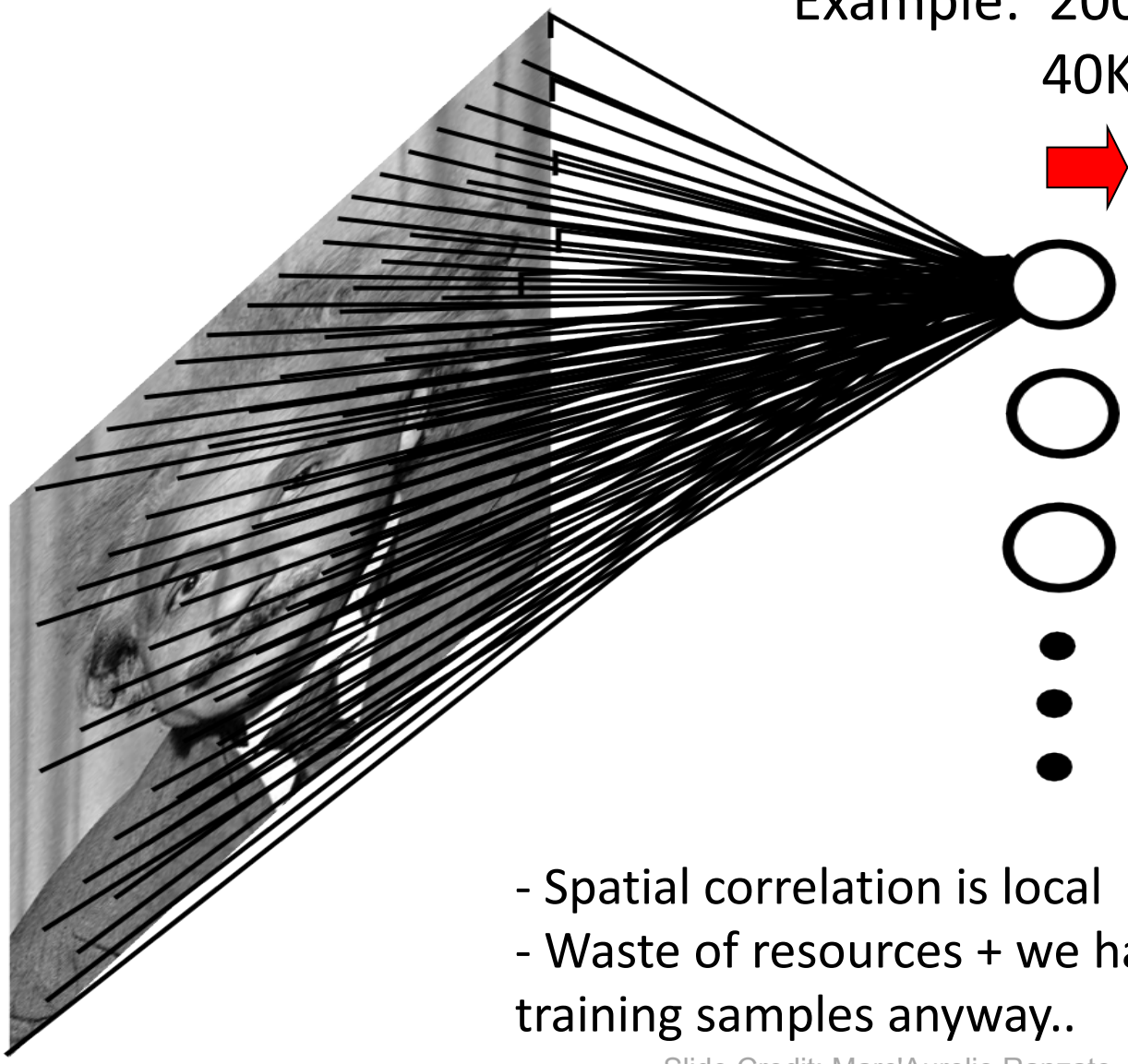
Slide Credit: Fei-Fei Li, Justin Johnson, Serena Yeung, CS 231n

Fully Connected Layer

Example: 200x200 image

40K hidden units

~2B parameters!!!



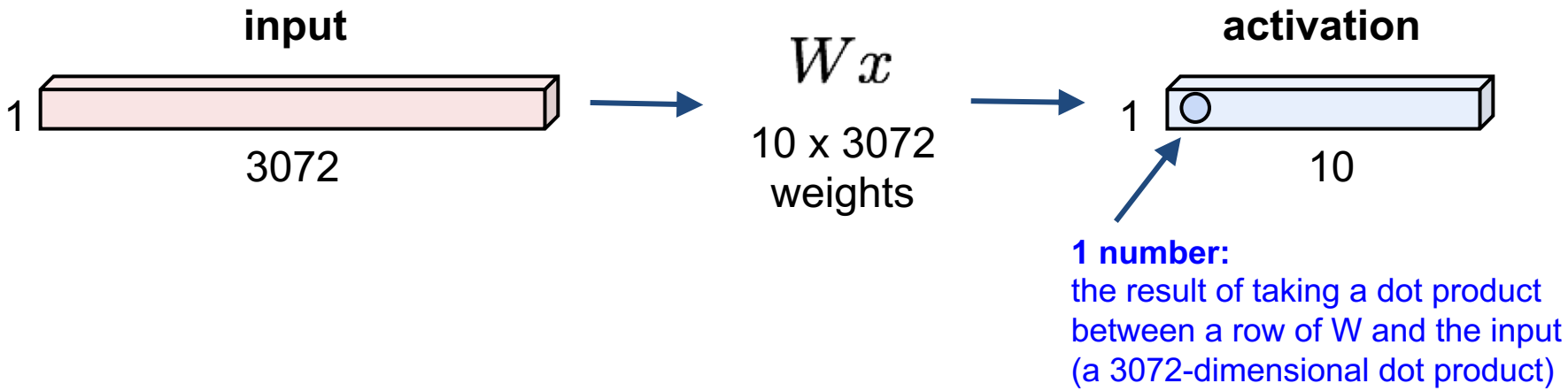
- Spatial correlation is local
- Waste of resources + we have not enough training samples anyway..

Slide Credit: Marc'Aurelio Ranzato

Fully Connected Layer

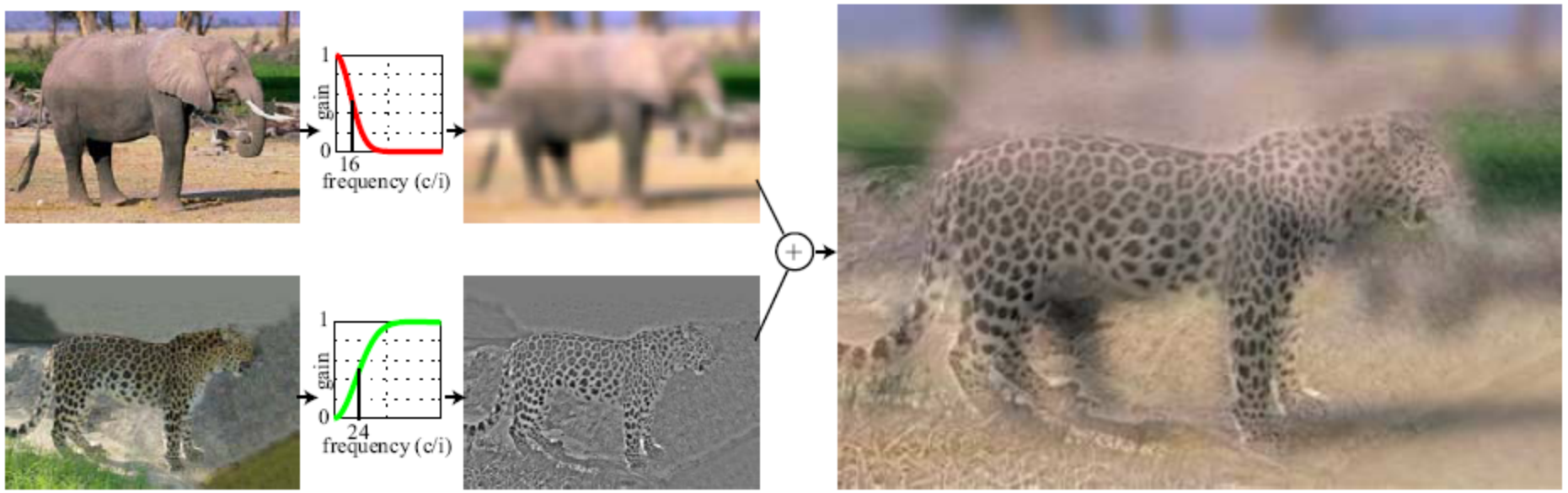
32x32x3 image -> stretch to 3072 x 1

Each neuron looks at the full input volume



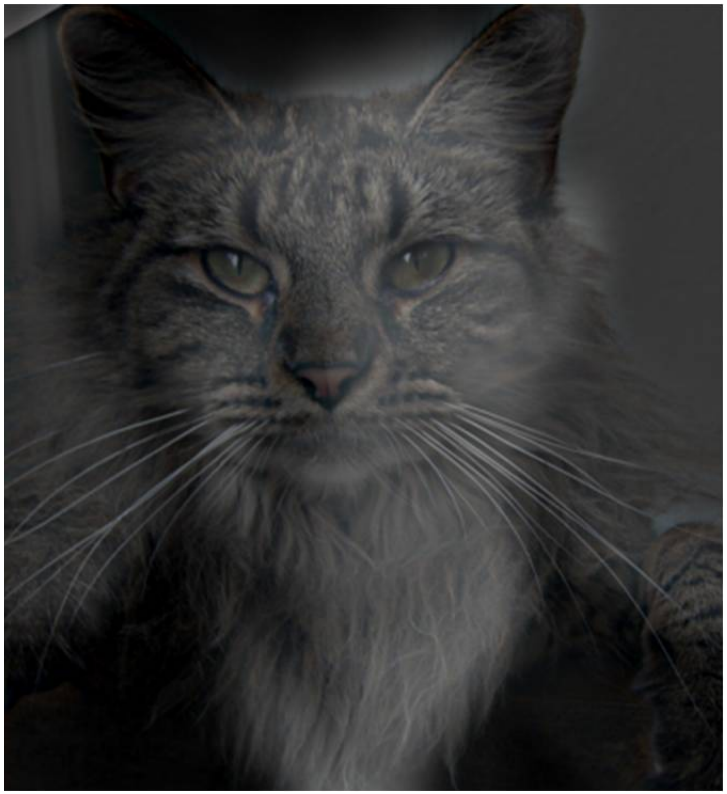
CNNs for Image Processing

Hybrid Images



- A. Oliva, A. Torralba, P.G. Schyns, "Hybrid Images," SIGGRAPH 2006

Why do we get different, distance-dependent interpretations of hybrid images?



?

Slide: Hoiem





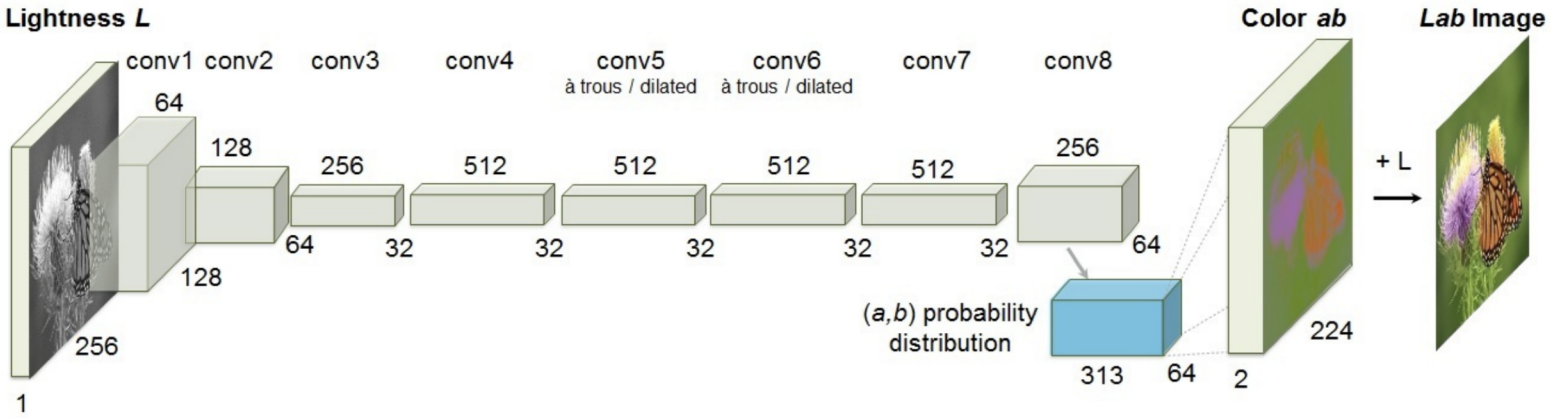
Colorization

- 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.

Colorization

- Training data: decompose any RGB image into L*a*b color space
  - *L*: grayscale input (lightness channel)
  - *ab*: color channels
- Train CNN with ***one million color images*** and a new objective function to incorporate more diverse colors. Many possible correct colorizations!



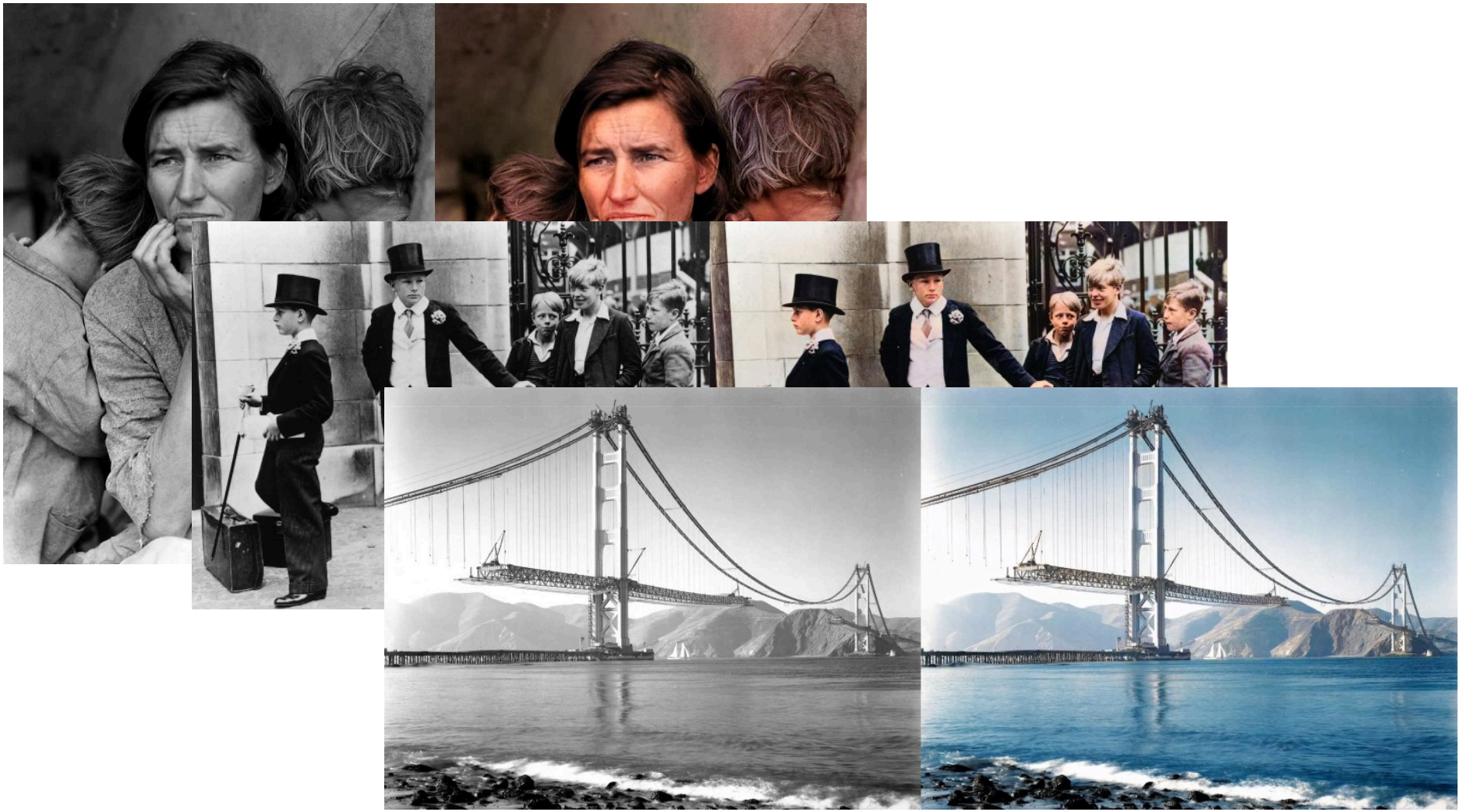
Colorful Image Colorization. Richard Zhang, Phillip Isola, Alexei A. Efros. ECCV 2016.

How to convert the inferred distribution to an image?

- 313-way classification over discretized ab color bins
- Network will output a distribution *z* over colors at each pixel. Need to convert to a single pixel value
  - Mode: vibrant but sometimes spatially inconsistent (e.g., the red splotches on the bus)
  - Mean: produces spatially consistent but desaturated results, exhibiting an unnatural sepia tone

$$\mathcal{H}(\mathbf{Z}_{h,w}) = \mathbb{E}\big[f_T(\mathbf{Z}_{h,w})\big], \quad f_T(\mathbf{z}) = \frac{\exp(\log(\mathbf{z})/T)}{\sum_q \exp(\log(\mathbf{z}_q)/T)}$$

DeOldify



https://github.com/jantic/DeOldify

Super-Resolution

Low resolution

High resolution

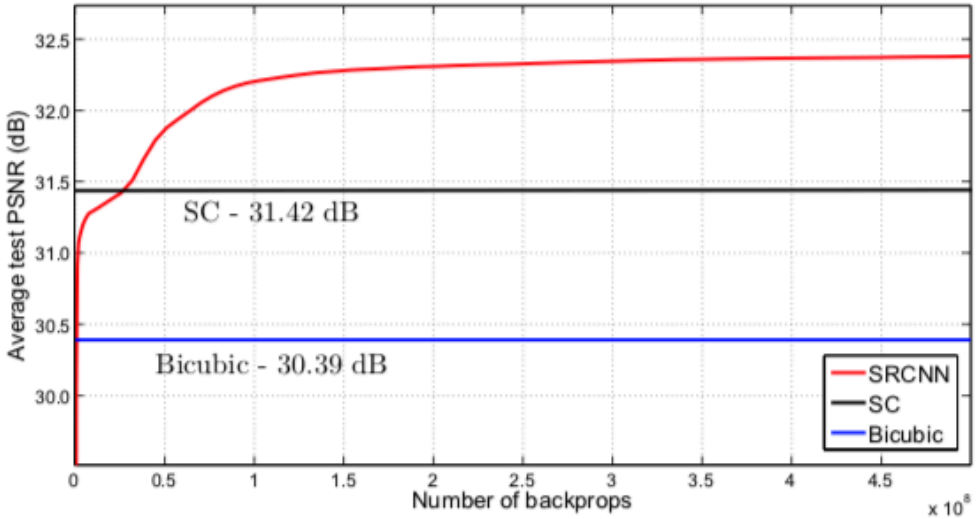
Super-Resolution as a task

- Quality-degrading factors / sources of noise:
  - Camera shake, shadows, motion blur, radial distortion from fisheye/GoPro type cameras, poor contrast, poor lighting, lossy compression, transmission defects, dust, haze, smoke, and mist, motion of the camera sensor platform, moving objects captured within the observed scene, e.g. people and vehicles.
- How to measure super-resolution?
  - Peak signal-to-noise ratio (PSNR), higher is better. Relies upon the Mean Square Error (MSE) error metric to evaluate image compression quality between two images:

$$MSE = \frac{1}{MN} \sum_{M} \sum_{N} [I_1(m,n) - I_2(m,n)]^2 = \|I_1 - I_2\|_F$$

$$PSNR = 10 \log_{10}(\frac{R^2}{MSE})$$

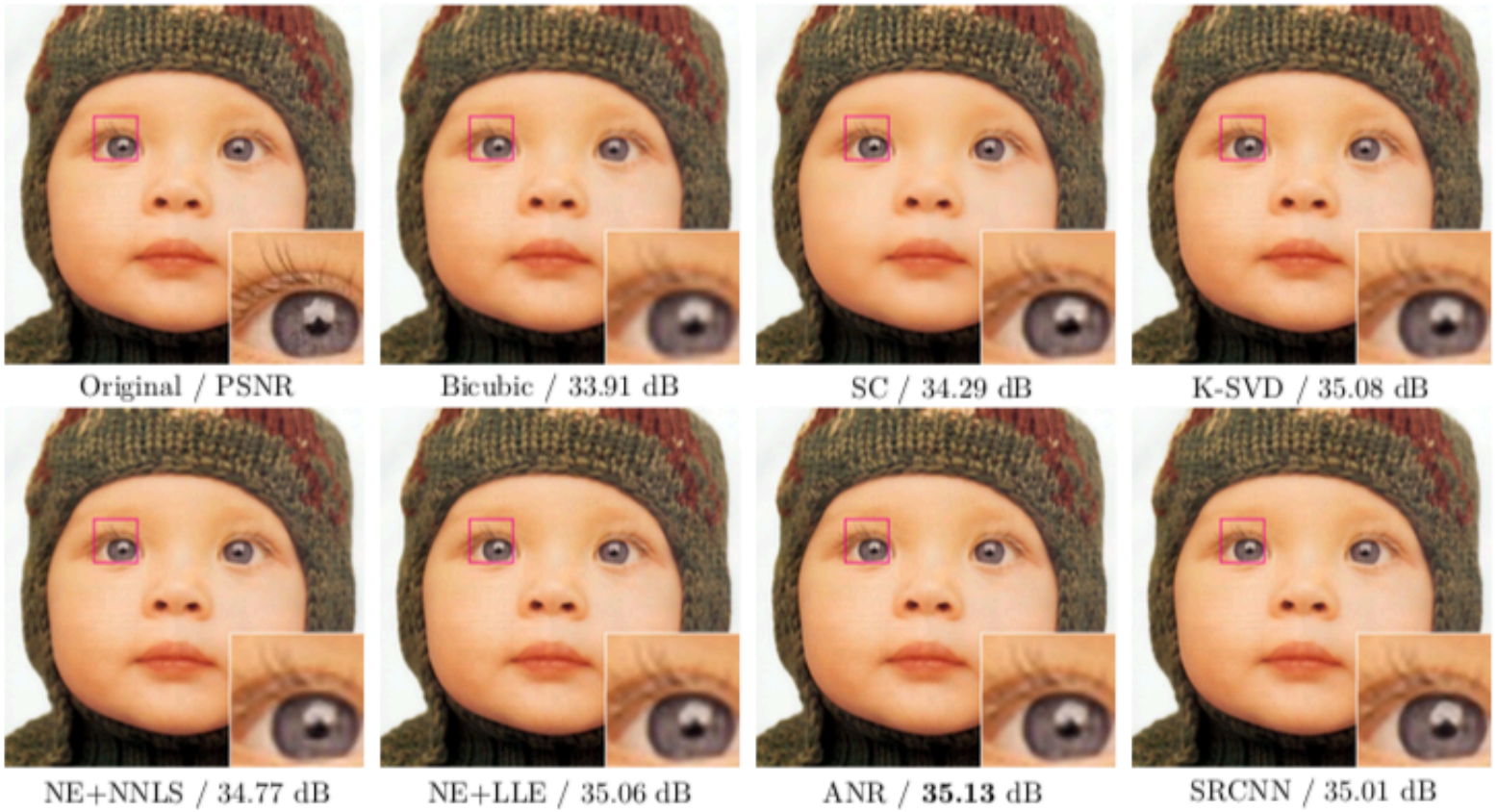
An early CNN paper (2016)



Original / PSNR | Bicubic / 24.04 dB | SC / 25.58 dB | SRCNN / 27.58 dB

Dong, Chao, et al. "Learning a deep convolutional network for image super-resolution." *European conference on computer vision*. Springer, Cham, 2014.

An early CNN paper (2016)



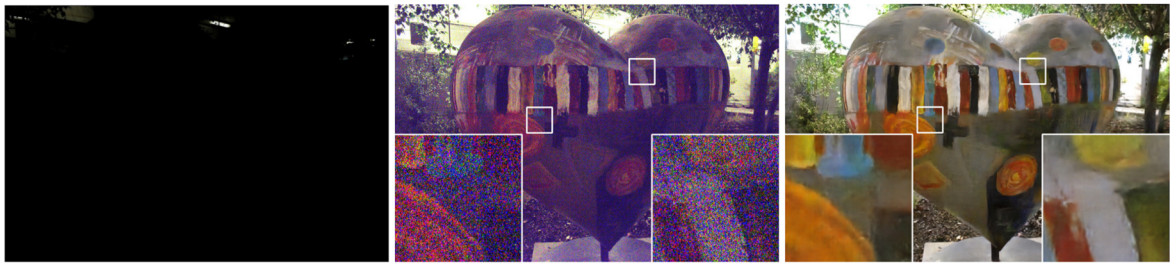
Upscaling factor of 3 !

Dong, Chao, et al. "Learning a deep convolutional network for image super-resolution." *European conference on computer vision*. Springer, Cham, 2014.

Underexposed Photo Enhancement

- Goal: enhance extreme low-light imaging with severely limited illumination (e.g., moonlight) and short exposure (exposure time is set to 1/30 second)
- The less light there is, the more ISO you need
  - High ISO can be used to increase brightness, but amplifies noise
  - Leads to low signal-to-noise ratio (SNR) due to low photon counts

(a) Camera output with ISO 8,000 (b) Camera output with ISO 409,600 (c) Our result from the raw data of (a)



(a) JPEG image produced by camera (b) Raw data via traditional pipeline (c) Our result

Learning to See in the Dark. Qifeng Chen, Vladlen Koltun. CVPR 2018.

Solution? Collect dataset and train a deep network

- See-in-the-Dark (SID) dataset contains 5094 raw short exposure images, each with a corresponding long-exposure reference image
- Corresponding reference (ground truth) images captured with 100-300x longer exposure (i.e. 10 to 30 seconds)
- Overcome low photon counts!
- Train deep neural networks to learn the image processing pipeline w/ L1 loss.

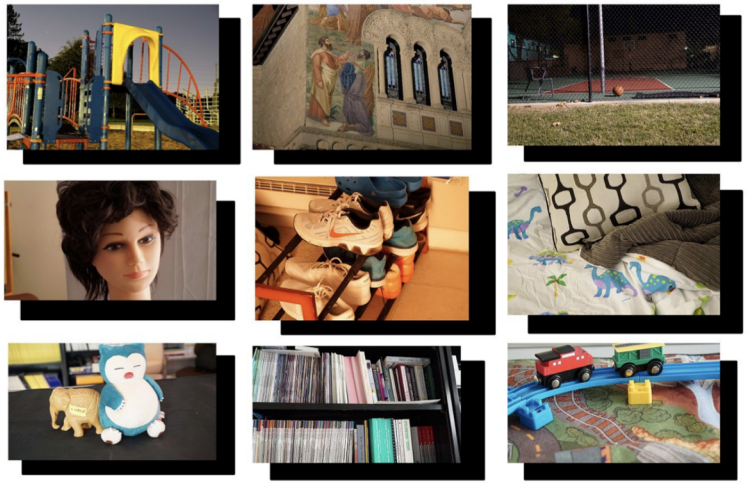
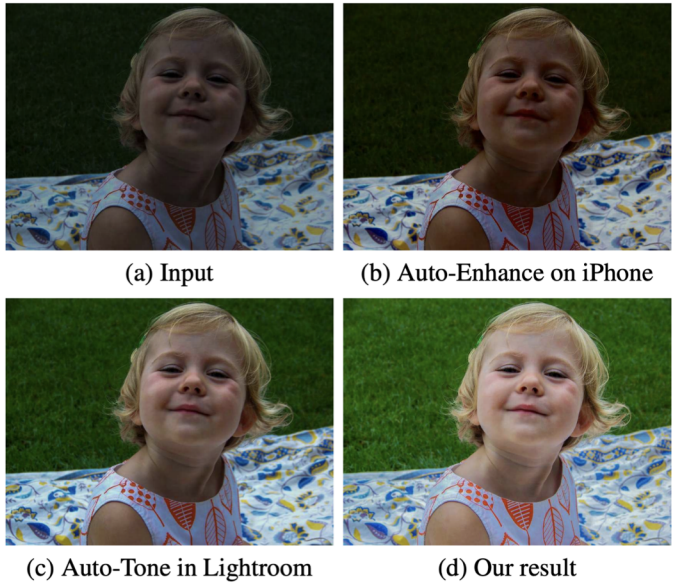


Figure 2. Example images in the SID dataset. Outdoor images in the top two rows, indoor images in the bottom rows. Long-exposure reference (ground truth) images are shown in front. Short-exposure input images (essentially black) are shown in the back. The illuminance at the camera is generally between 0.2 and 5 lux outdoors and between 0.03 and 0.3 lux indoors.

Learning to See in the Dark. Qifeng Chen, Vladlen Koltun. CVPR 2018.

Underexposed Photo Enhancement

- Learn image-to-image mapping? Too hard!
- Instead estimate an image-to-illumination mapping (model varying-lighting conditions)
  - Illumination maps for natural images typically have relatively simple forms with known priors
- Then take illumination map to light up the underexposed photo.
- Minimize (reconstruction loss + smoothness loss + color loss)



(a) Input (b) Auto-Enhance on iPhone

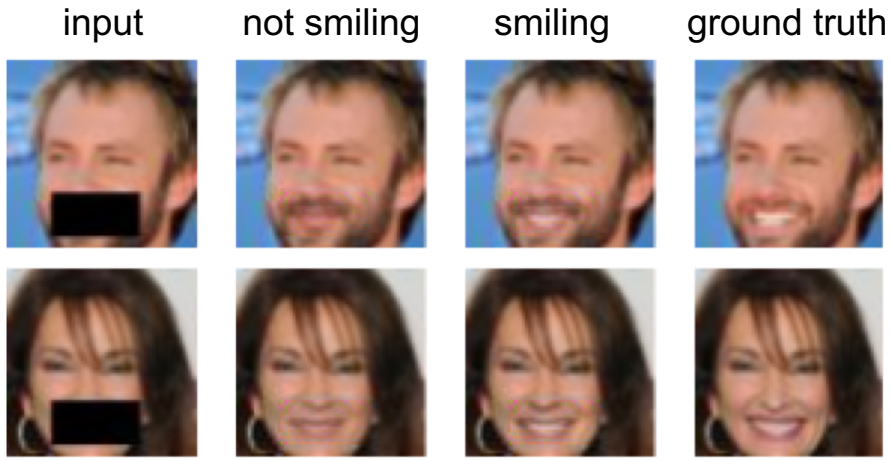
(c) Auto-Tone in Lightroom (d) Our result

Figure 1: A challenging underexposed photo (a) enhanced by various tools (b)-(d). Our result contains more details, distinct contrast, and more natural color.

Underexposed Photo Enhancement Using Deep Illumination Estimation. Wang et al. CVPR 2019.

Image Inpainting

- Perceptual loss is added to ELBO, the typical objective function used in variational autoencoders, to increase the sharpness and overall quality of inpainted images
- Demonstrate results on attribute-guided image completion



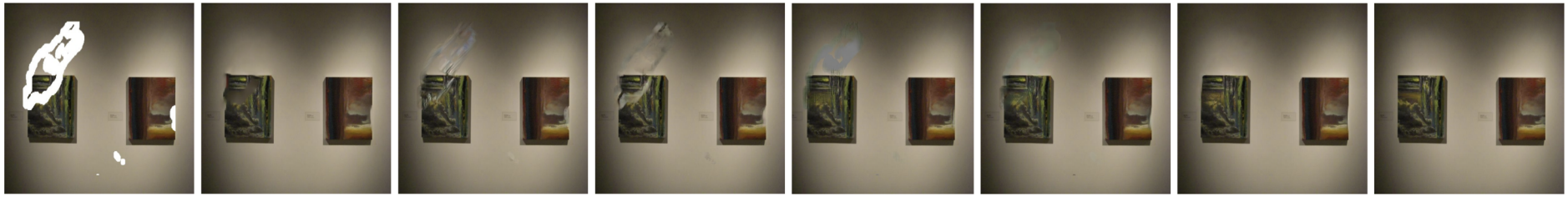
$$\mathcal{L}_{recon} = \|x_{gen} - x_{gt}\|^2 + \sum_l \lambda_l \|\eta_l(x_{gen}) - \eta_l(x_{gt})\|^2$$

x_{gen} : generated image
x_{gt} : ground truth image
η_l : activation of the l^{th} layer of a pre-trained VGG

Variational Image Inpainting. Cusuh Ham*, Amit Raj*, Vincent Cartillier*, Irfan Essa. NeuRIPS 2018 Workshop.

Image Inpainting

- Proposes partial convolutions, comprised of a masked & re-normalized convolution operator
- Updates mask automatically after partial convolutions, removing any masking where partial convolution was able to operate on unmasked value



(a) Image with hole (b) PatchMatch (c) Iizuka et al.[10] (d) Yu et al.[38] (e) Hole=127.5 (f) Hole=IN_Mean (g) Partial Conv (h) Ground Truth

Image Inpainting for Irregular Holes Using Partial Convolutions. Liu et al. ECCV 2018.