

Computer Vision

IFT6758 - Data Science

Sources:

<http://www.cs.cmu.edu/~16385/>

<http://cs231n.stanford.edu/2018/syllabus.html>

<http://www.cse.psu.edu/~rtc12/CSE486/>

Announcement

- Assignment 3 is available on gradescope:
 - Algorithmic discrimination
 - NLP (has two bonus point questions)
 - CV (basics)
 - **Due: November 28**
- Grades of assignment 2 and mid-term will be published this week.

CV Pipeline

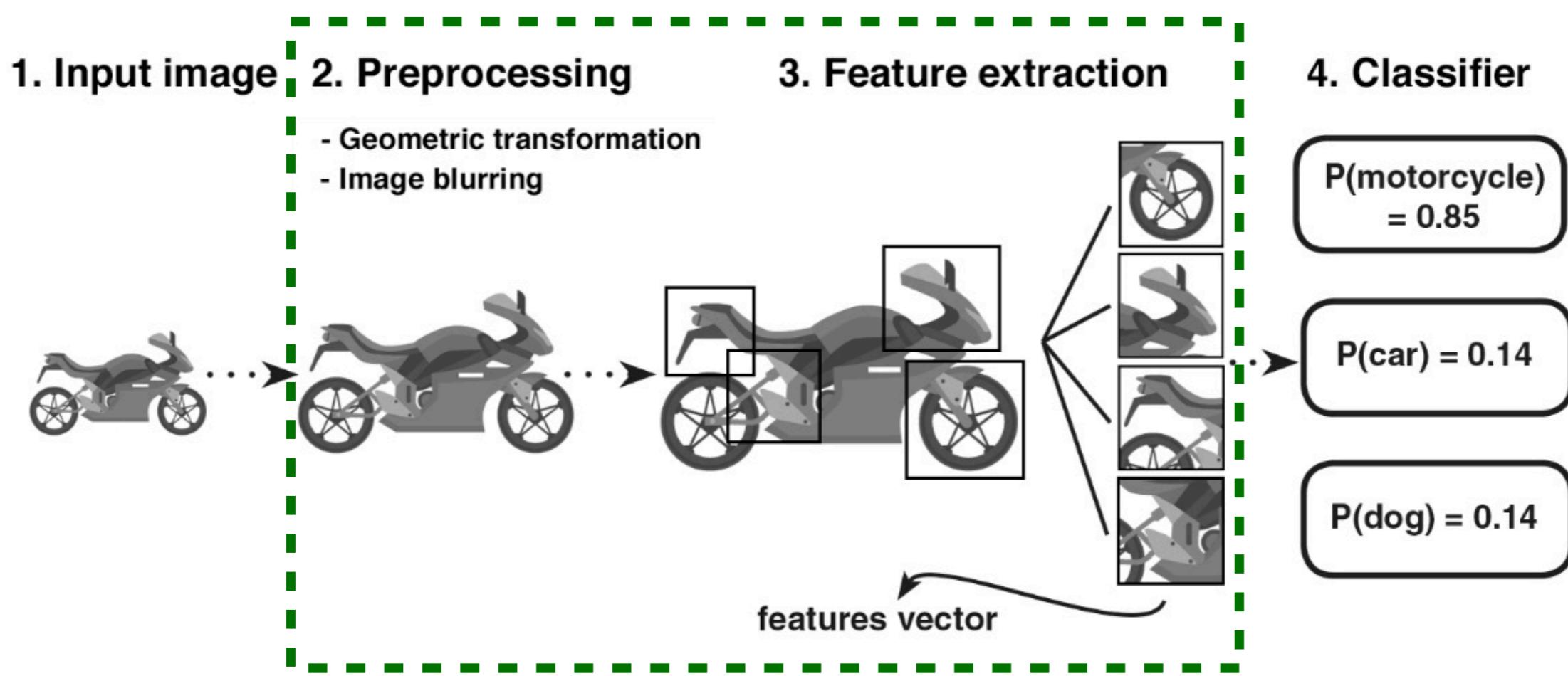
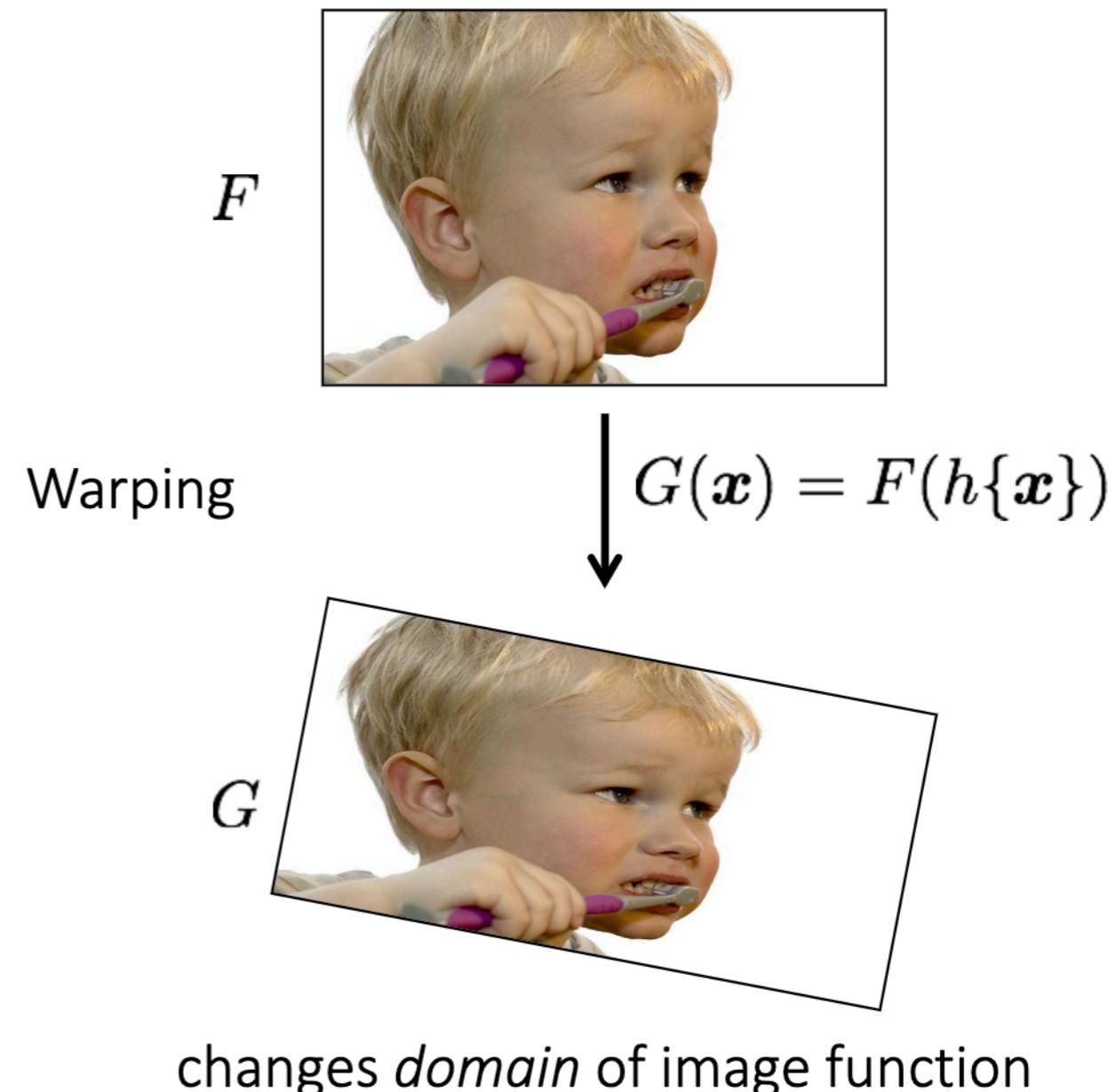
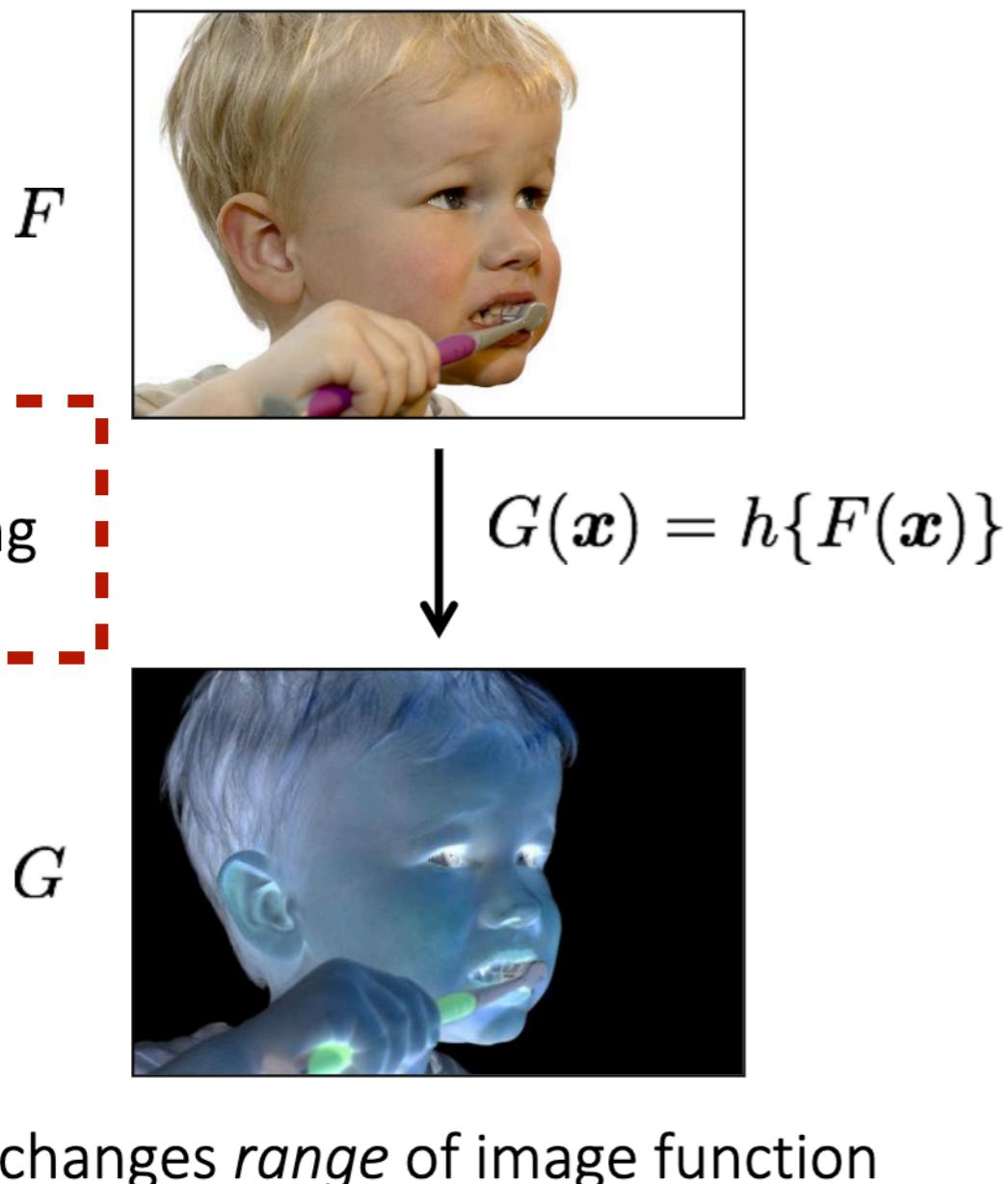


Image transformations



Recall: filtering

- **Filtering:**
 - Form a new image whose pixels are a combination original pixel values

Goals:

- Extract useful information from the images
 - Features (edges, corners, blobs...)
- Modify or enhance image properties:
 - super-resolution; in-painting; de-noising

Convolution

Commutative: $f * g = g * f$

Associative: $(f * g) * h = f * (g * h)$

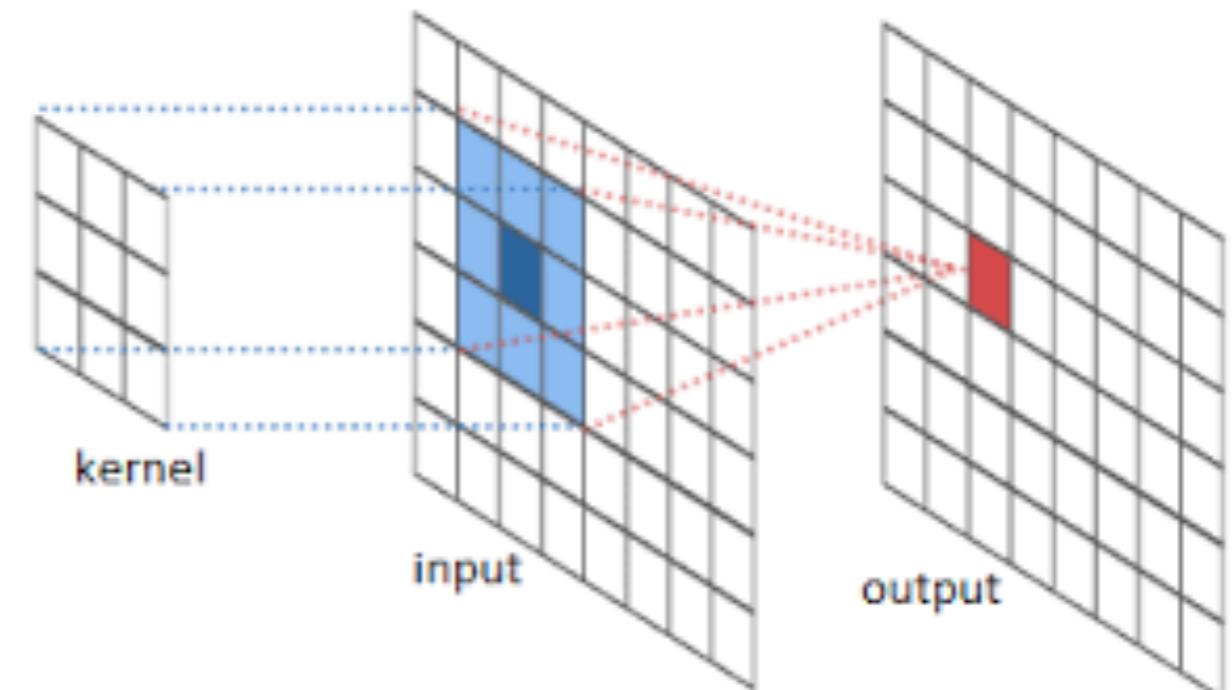
Distributive: $(f + g) * h = f * h + g * h$

Linear: $(af + bg) * h = af * h + bg * h$

Shift Invariant: $f(x+t) * h = (f * h)(x+t)$

Differentiation rule:

$$\frac{\partial}{\partial x}(f * g) = \frac{\partial f}{\partial x} * g$$



Convolution

Commutative: $f * g = g * f$

Associative: $(f * g) * h = f * (g * h)$

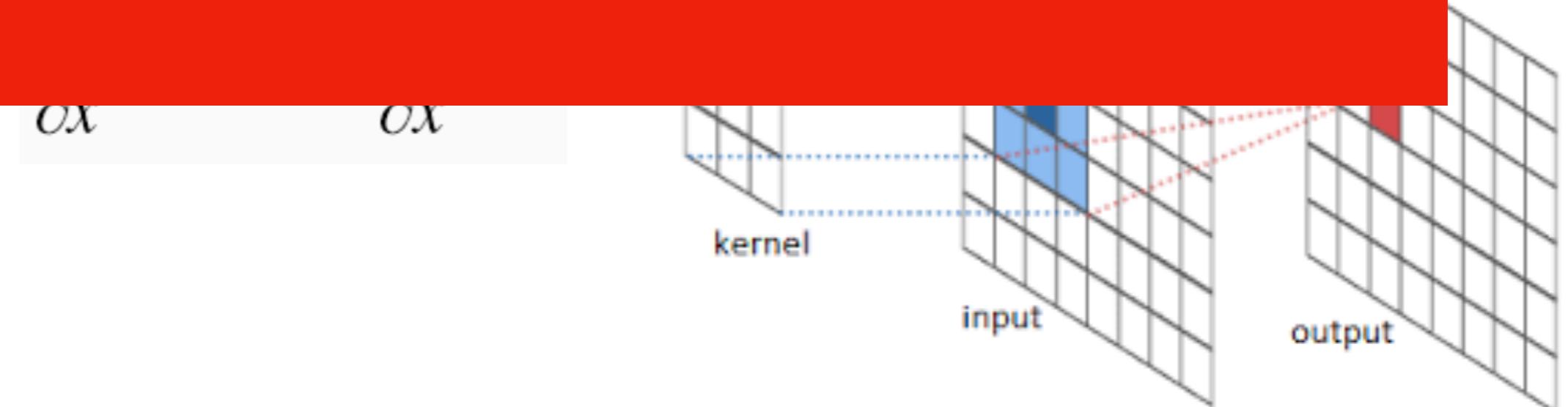
Distributive: $(f + g) * h = f * h + g * h$

Linear

Shift

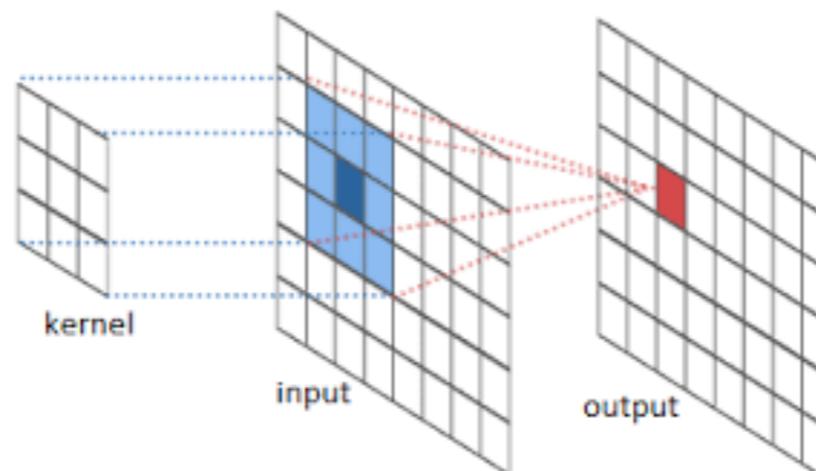
Differ

All of computer vision
is convolutions
(basically)



Convolution

- The mathematics for many filters can be expressed in a principal manner using 2D convolution, such as **smoothing** and **sharpening** images and **detecting edges**.
- Convolution in 2D operates on **two images**, with one functioning as **the input image** and the other, called the **kernel**, serving as a **filter**.
- It expresses the amount overlap of one function as it is shifted over another function, as the output image is produced by sliding the kernel over the input image.

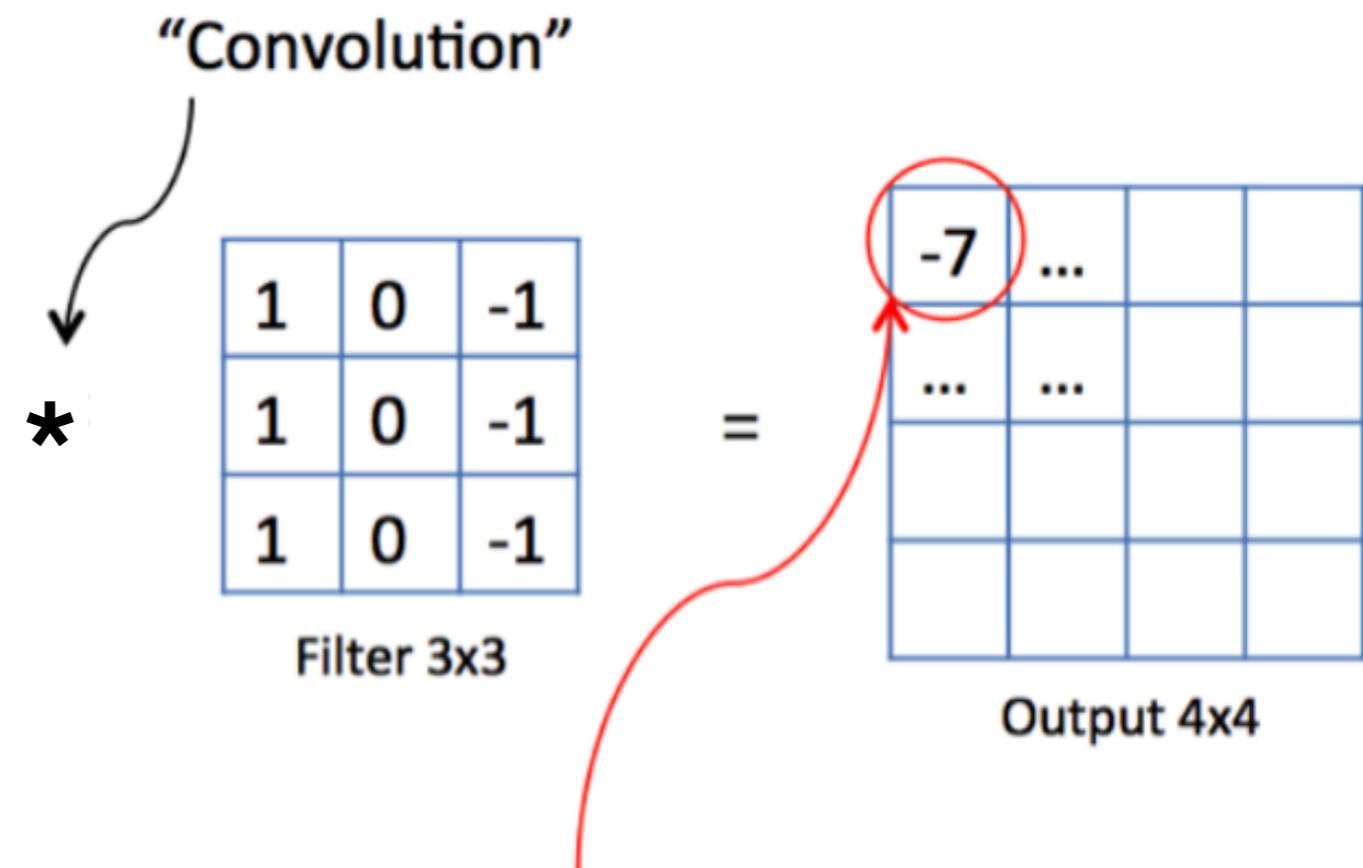


Convolution

- Convolution is the process of adding each element of the image to its local neighbors, weighted by the kernel.

3	1	1	2	8	4
1	0	7	3	2	6
2	3	5	1	1	3
1	4	1	2	6	5
3	2	1	3	7	2
9	2	6	2	5	1

Original image 6x6



Result of the element-wise product and sum of the filter matrix and the orginal image

Convolution for 2D discrete signals

Definition of filtering as convolution:

$$(f * g)(x, y) = \sum_{i,j=-\infty}^{\infty} f(i, j)I(x - i, y - j)$$

→ → →

filtered image filter input image

Convolution for 2D discrete signals

Definition of filtering as convolution:

$$(f * g)(x, y) = \sum_{i,j=-\infty}^{\infty} f(i, j)I(x - i, y - j)$$

→
filtered image filter input image

If the filter $f(i, j)$ is non-zero only within $-1 \leq i, j \leq 1$, then

$$(f * g)(x, y) = \sum_{i,j=-1}^{1} f(i, j)I(x - i, y - j)$$

The box filter kernel we saw earlier is the 3x3 matrix representation of $f(i, j)$.

Convolution Examples



Original

$$\begin{matrix} \bullet 0 & \bullet 0 & \bullet 0 \\ \bullet 0 & \bullet 1 & \bullet 0 \\ \bullet 0 & \bullet 0 & \bullet 0 \end{matrix}$$

=



Filtered
(no change)

Shifted right by one pixel:



Original

$$\begin{matrix} \bullet 0 & \bullet 0 & \bullet 0 \\ \bullet 0 & \bullet 0 & \bullet 1 \\ \bullet 0 & \bullet 0 & \bullet 0 \end{matrix}$$

=

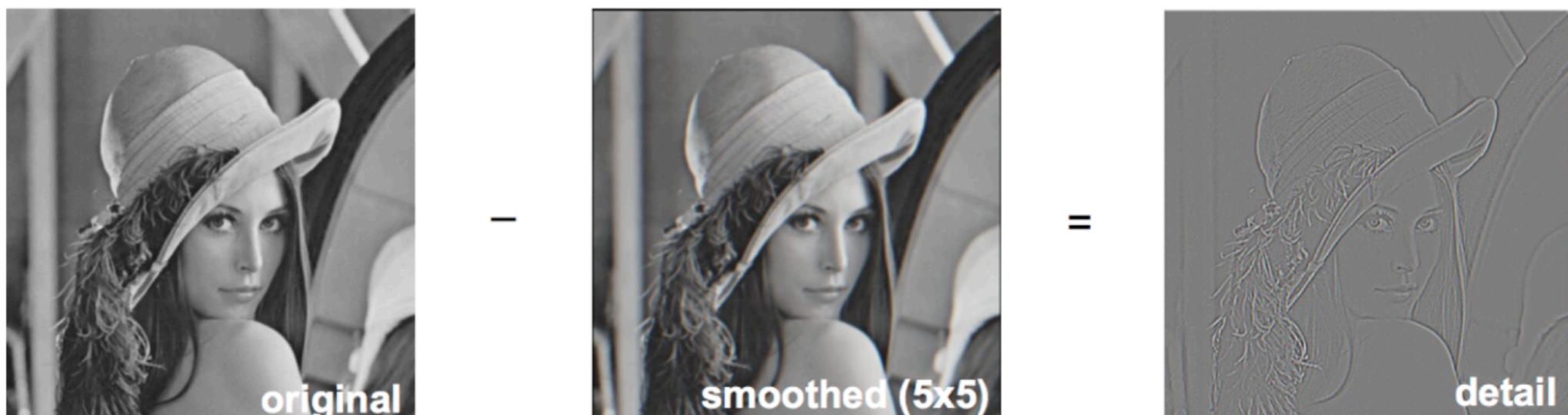


Shifted right
By 1 pixel

Convolution Examples

- A **sharpening** filter can be broken down into two steps: It takes a smoothed image, subtracts it from the original image to obtain the "details" of the image, and adds the "details" to the original image.

Step 1: Original - Smoothed = "Details"



$$\begin{matrix} \bullet 0 & \bullet 0 & \bullet 0 \\ \bullet 0 & \bullet 1 & \bullet 0 \\ \bullet 0 & \bullet 0 & \bullet 0 \end{matrix}$$

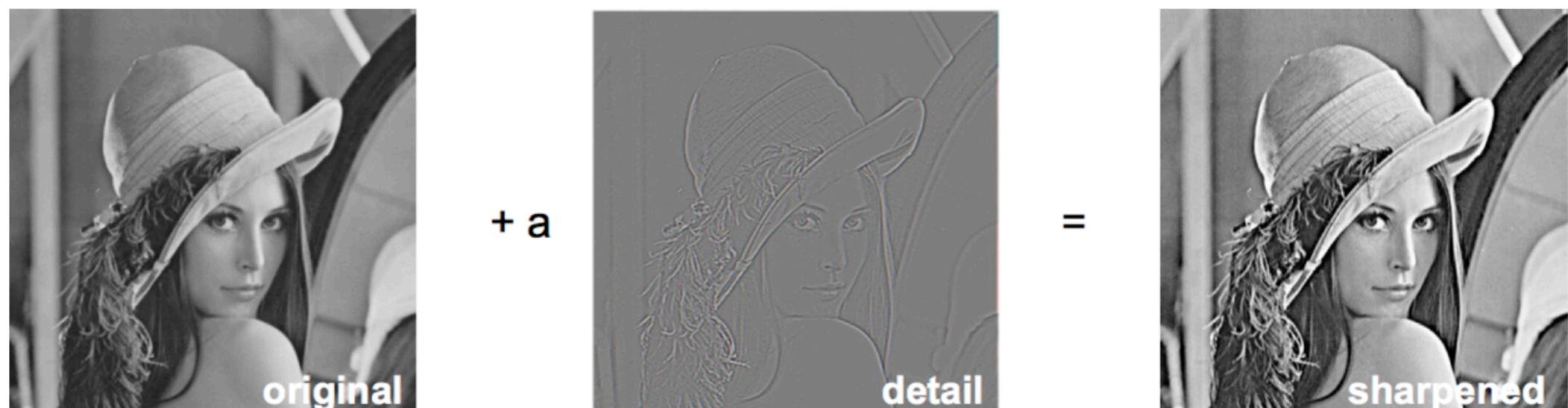
-

$$\frac{1}{9} \begin{matrix} \bullet 1 & \bullet 1 & \bullet 1 \\ \bullet 1 & \bullet 1 & \bullet 1 \\ \bullet 1 & \bullet 1 & \bullet 1 \end{matrix}$$

Convolution Examples

- A **sharpening** filter can be broken down into two steps: It takes a smoothed image, subtracts it from the original image to obtain the "details" of the image, and adds the "details" to the original image.

Step 2: Original + "Details" = Sharpened



$$\begin{array}{|c|c|c|} \hline \bullet & 0 & 0 \\ \hline 0 & \bullet & 1 \\ \hline 0 & 0 & 0 \\ \hline \end{array} + \begin{array}{|c|c|c|} \hline \bullet & 0 & 0 \\ \hline 0 & \bullet & 1 \\ \hline 0 & 0 & 0 \\ \hline \end{array} - \frac{1}{9} \begin{array}{|c|c|c|} \hline \bullet & 1 & 1 \\ \hline 1 & \bullet & 1 \\ \hline 1 & 1 & \bullet \\ \hline \end{array} = \begin{array}{|c|c|c|} \hline \bullet & 0 & 0 \\ \hline 0 & \bullet & 2 \\ \hline 0 & 0 & 0 \\ \hline \end{array} - \frac{1}{9} \begin{array}{|c|c|c|} \hline \bullet & 1 & 1 \\ \hline 1 & \bullet & 1 \\ \hline 1 & 1 & \bullet \\ \hline \end{array}$$

Convolution Examples

- **Gaussian Smoothing Filter:** is a weighted averaging filter which gives more weights to central pixels and less weights to the neighbors

$$G_{\sigma} \equiv \frac{1}{2\pi\sigma^2} \exp\left\{-\frac{x^2+y^2}{2\sigma^2}\right\}$$



original



sigma = 3



sigma = 10

Convolution vs Correlation

Definition of discrete 2D
convolution:

$$(f * g)(x, y) = \sum_{i,j=-\infty}^{\infty} f(i, j)I(x - i, y - j)$$


← notice the flip

Definition of discrete 2D
correlation:

$$(f * g)(x, y) = \sum_{i,j=-\infty}^{\infty} f(i, j)I(x + i, y + j)$$

← notice the lack of a flip

- Most of the time won't matter, because our kernels will be symmetric.

Convolution vs correlation

- **Convolution:** is an integral that expresses the amount of overlap of one function as it is shifted over another function.
 - Convolution is a **filtering** operation
- **Correlation** compares the similarity of two sets of data. Correlation computes a measure of **similarity of two input signals** as they are shifted by another.
 - The correlation reaches a maximum at the time when the two signals matches best.

Correlation application

- Correlation tells you how similar the signal is to the filter at any point. This is used for image alignment, template matching and simple image matching.



Template



Original image

Separable filters

A 2D filter is separable if it can be written as the product of a “column” and a “row”.

example:
box filter

1	1	1
1	1	1
1	1	1

$$\begin{matrix} & = & \begin{matrix} 1 \\ 1 \\ 1 \end{matrix} & * & \begin{matrix} 1 & 1 & 1 \end{matrix} \\ & & \text{column} & & \text{row} \end{matrix}$$

Separable filters

A 2D filter is separable if it can be written as the product of a “column” and a “row”.

example:
box filter

1	1	1
1	1	1
1	1	1

$$\begin{matrix} & = & \begin{matrix} 1 \\ 1 \\ 1 \end{matrix} & * & \begin{matrix} 1 & 1 & 1 \end{matrix} \\ & & \text{column} & & \text{row} \end{matrix}$$

2D convolution with a separable filter is equivalent to two 1D convolutions (with the “column” and “row” filters).

Separable filters

A 2D filter is separable if it can be written as the product of a “column” and a “row”.

example:
box filter

1	1	1
1	1	1
1	1	1

$$= \begin{matrix} 1 \\ 1 \\ 1 \end{matrix} * \begin{matrix} 1 & 1 & 1 \end{matrix}$$

column row

2D convolution with a separable filter is equivalent to two 1D convolutions (with the “column” and “row” filters).

If the image has $M \times M$ pixels and the filter kernel has size $N \times N$:

What is the cost of convolution with a non-separable filter?

Separable filters

A 2D filter is separable if it can be written as the product of a “column” and a “row”.

example:
box filter

1	1	1
1	1	1
1	1	1

$$= \begin{matrix} 1 \\ 1 \\ 1 \end{matrix} * \begin{matrix} 1 & 1 & 1 \end{matrix}$$

column

1	1	1
1	1	1
1	1	1

row

2D convolution with a separable filter is equivalent to two 1D convolutions (with the “column” and “row” filters).

If the image has $M \times M$ pixels and the filter kernel has size $N \times N$:

What is the cost of convolution with a non-separable filter?

$M^2 \times N^2$

Separable filters

A 2D filter is separable if it can be written as the product of a “column” and a “row”.

example:
box filter

1	1	1
1	1	1
1	1	1

$$= \begin{matrix} 1 \\ 1 \\ 1 \end{matrix} * \begin{matrix} 1 & 1 & 1 \end{matrix}$$

column

1	1	1
1	1	1
1	1	1

row

2D convolution with a separable filter is equivalent to two 1D convolutions (with the “column” and “row” filters).

If the image has $M \times M$ pixels and the filter kernel has size $N \times N$:

What is the cost of convolution with a non-separable filter? $M^2 \times N^2$

What is the cost of convolution with a separable filter?

Separable filters

A 2D filter is separable if it can be written as the product of a “column” and a “row”.

example:
box filter

1	1	1
1	1	1
1	1	1

$$= \begin{matrix} 1 \\ 1 \\ 1 \end{matrix} * \begin{matrix} 1 & 1 & 1 \end{matrix}$$

column

1	1	1
1	1	1
1	1	1

row

2D convolution with a separable filter is equivalent to two 1D convolutions (with the “column” and “row” filters).

If the image has $M \times M$ pixels and the filter kernel has size $N \times N$:

What is the cost of convolution with a non-separable filter? $M^2 \times N^2$

What is the cost of convolution with a separable filter? $2 \times N \times M^2$

Examples of separable filters

- **Box-filter** that is used as smoothing filter.

Simple box filter

$$\frac{1}{3} \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} * \frac{1}{3} [1 \quad 1 \quad 1] = \frac{1}{9} \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

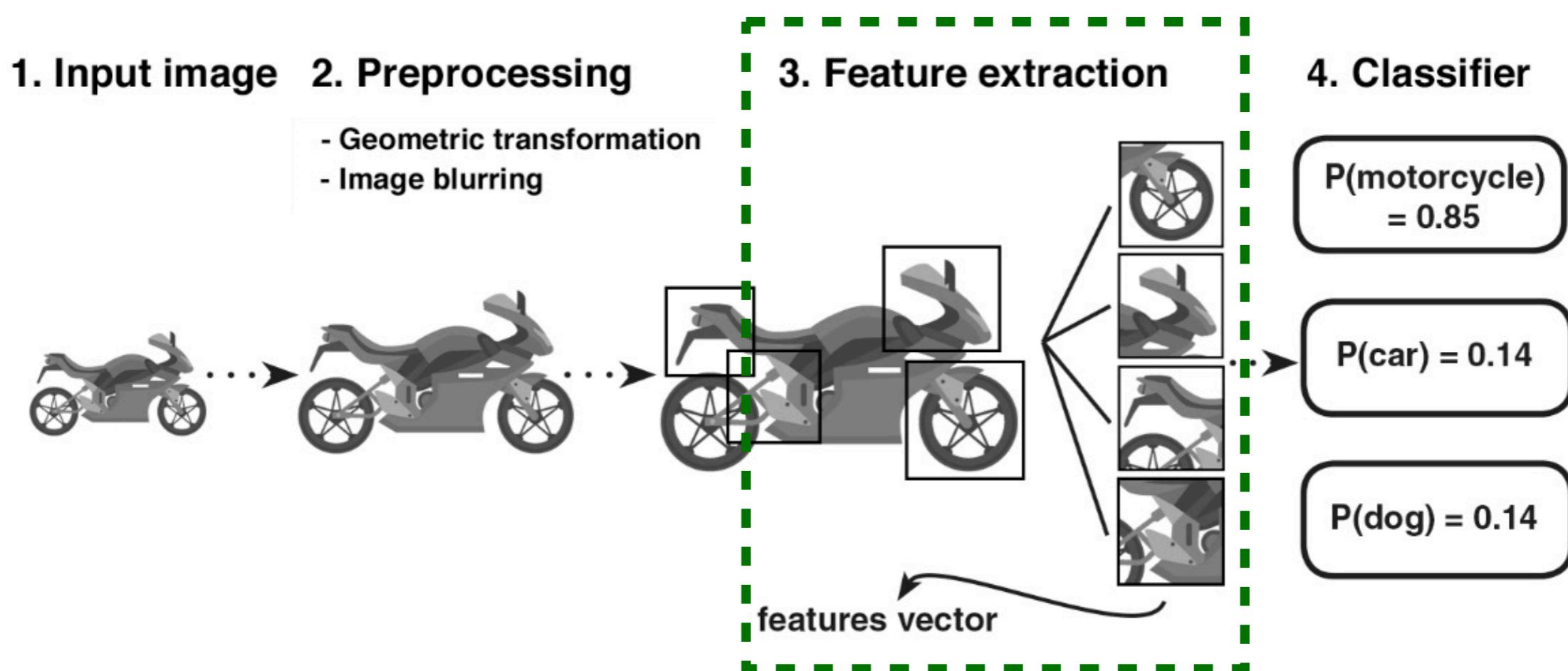
- **Sober operator** which is used commonly for edge detection.

Simple Gaussian

$$\begin{bmatrix} 1 \\ 2 \\ 1 \end{bmatrix} * [1 \quad 0 \quad -1] = \begin{bmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{bmatrix}$$

Finite diff operator

CV Pipeline

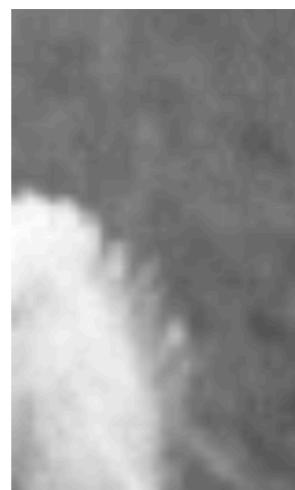
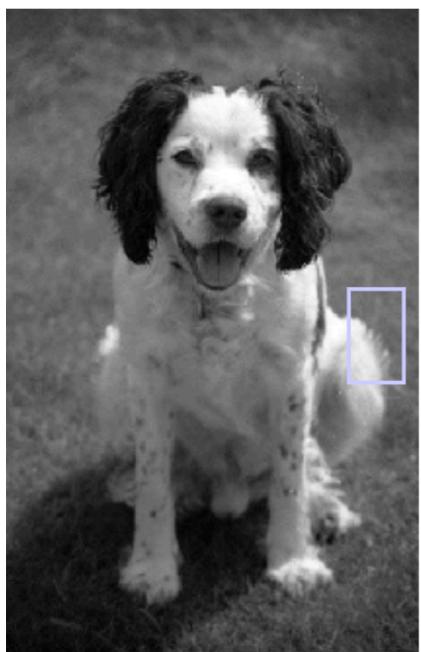


Edge detection

- Edges are the points in an image where the image brightness changes sharply or has discontinuities. Such discontinuities generally correspond to:
 - Discontinuities in depth
 - Discontinuities in surface orientation
 - Changes in material properties
 - Variations in scene illumination
- Edges are important for two main reasons.
 - 1) Most semantic and shape information can be deduced from them, so we can perform object recognition and analyze perspectives and geometry of an image.
 - 2) They are a more compact representation than pixels.

Edge detection

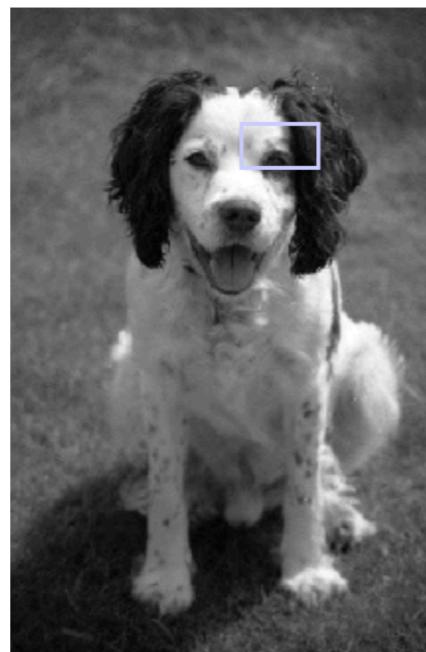
Boundaries of objects



Boundaries of Lighting

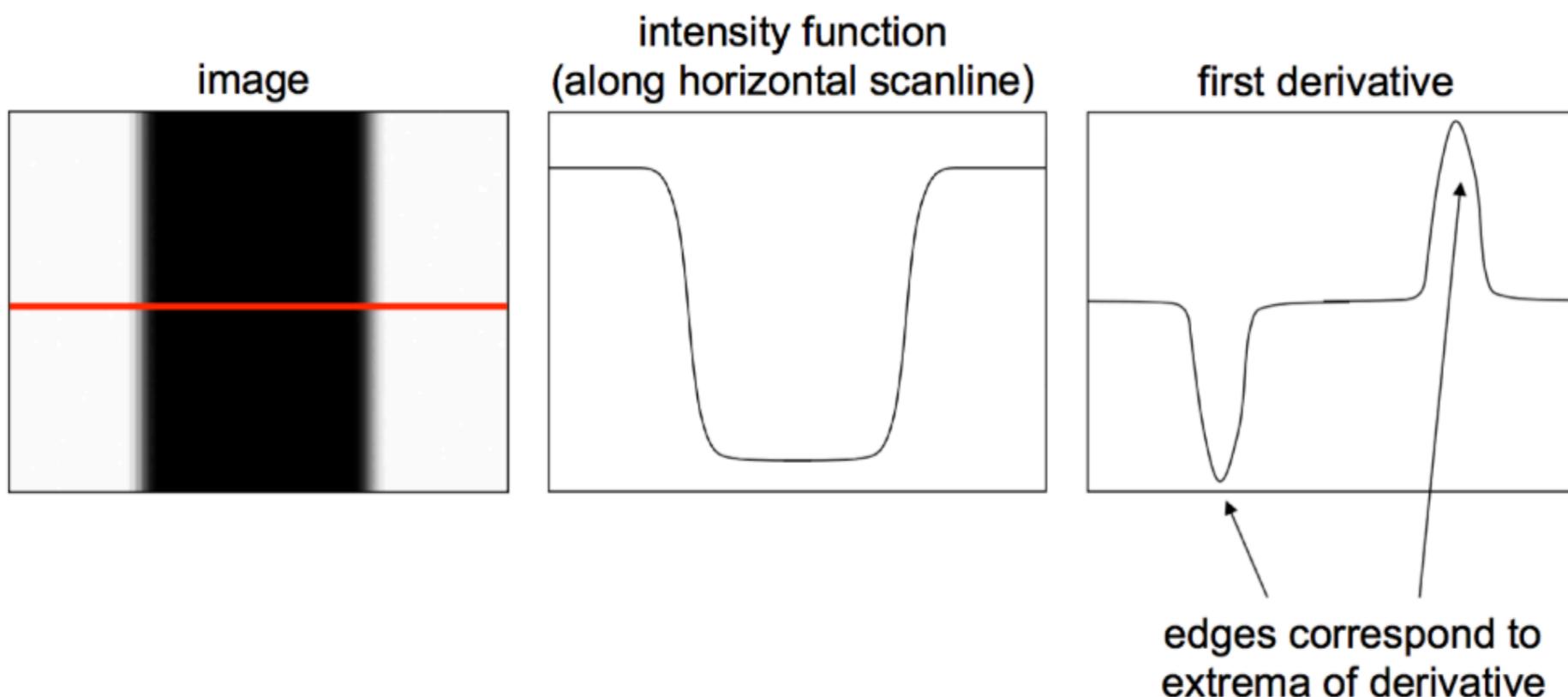


Boundaries of Material Properties



Characterizing edges

- We can pinpoint where edges occur from an image's intensity profile along a row or column of the image. Wherever there is a rapid change in the intensity function indicates an edge, as seen where the function's first derivative has a local extrema.



Partial derivatives with Convolution

For 2D function $f(x,y)$, the partial derivative is:

$$\frac{\partial f(x, y)}{\partial x} = \lim_{\varepsilon \rightarrow 0} \frac{f(x + \varepsilon, y) - f(x, y)}{\varepsilon}$$

For discrete data, we can approximate using finite differences:

$$\frac{\partial f(x, y)}{\partial x} \approx \frac{f(x+1, y) - f(x, y)}{1}$$

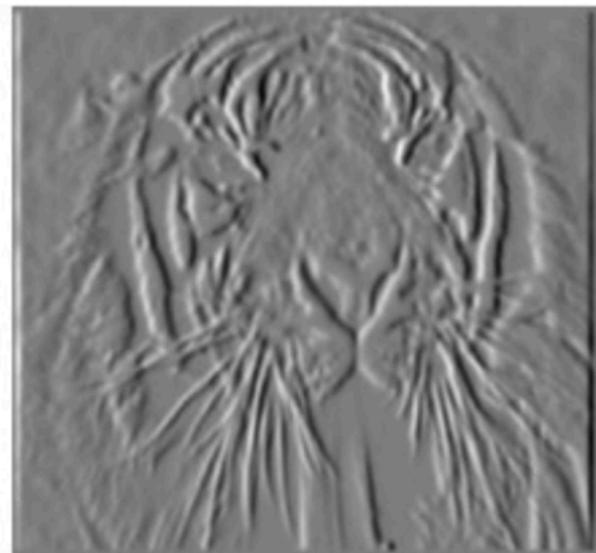
To implement above as convolution, what would be the associated filter?

Source: K. Grauman

Partial derivatives of an image

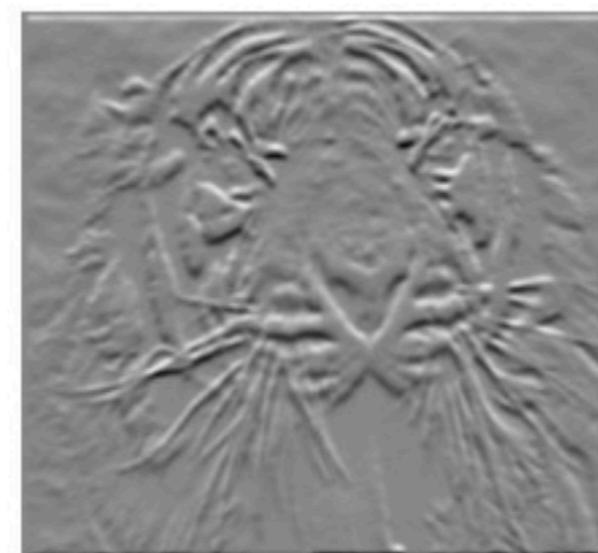
$$\begin{array}{|c|c|c|} \hline -1 & 0 & 1 \\ \hline \end{array}$$

$$\frac{\partial f(x, y)}{\partial x}$$



$$\begin{array}{|c|c|c|} \hline 1 & 0 & -1 \\ \hline \end{array}$$

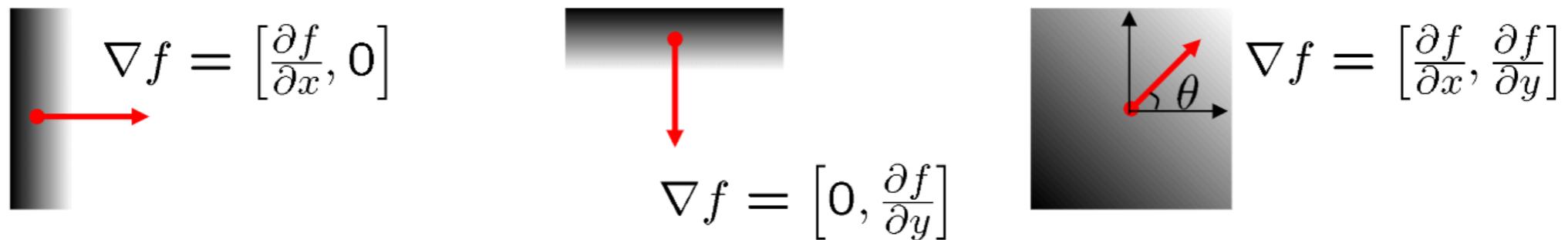
$$\frac{\partial f(x, y)}{\partial y}$$



Which shows changes with respect to x?

Image Gradient

- The gradient of an image: $\nabla f = \left[\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y} \right]$



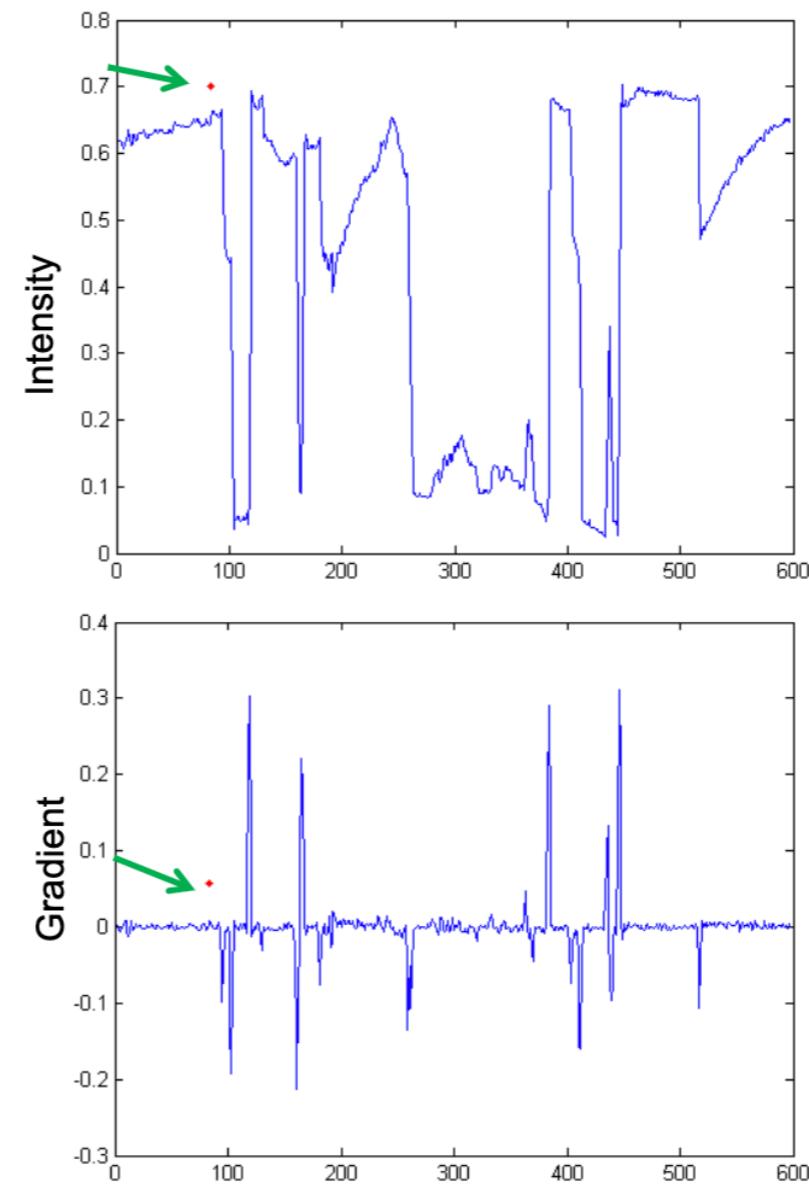
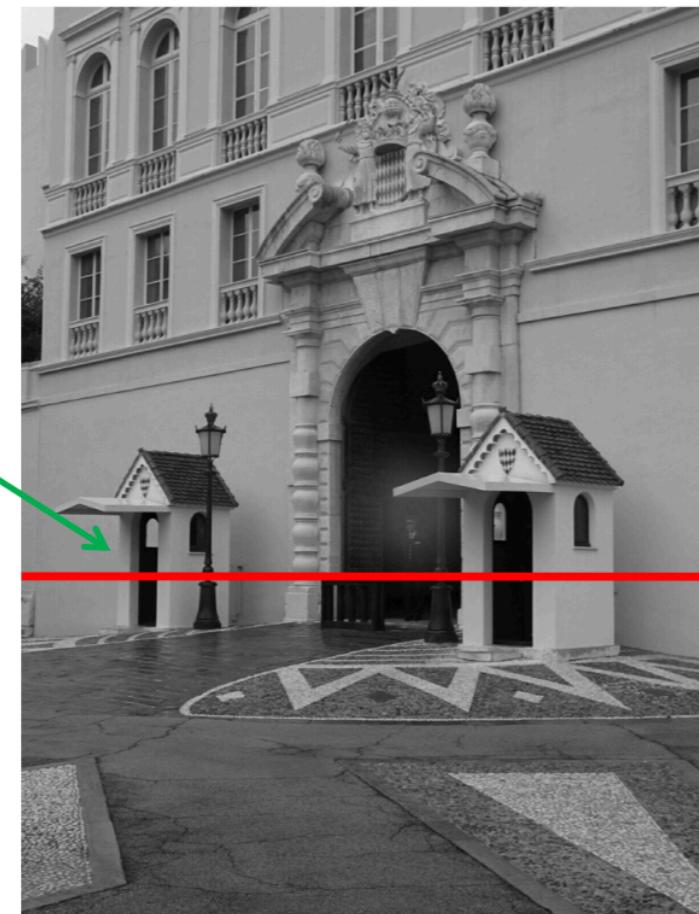
The gradient points in the direction of most rapid increase in intensity

The *edge strength* is given by the gradient magnitude

$$\|\nabla f\| = \sqrt{\left(\frac{\partial f}{\partial x}\right)^2 + \left(\frac{\partial f}{\partial y}\right)^2}$$

Source: Steve Seitz

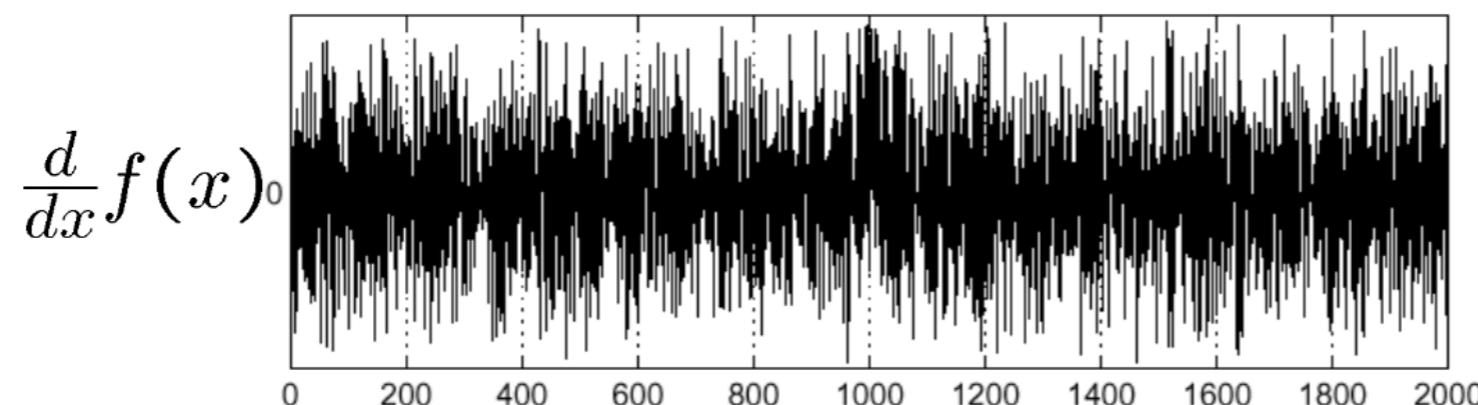
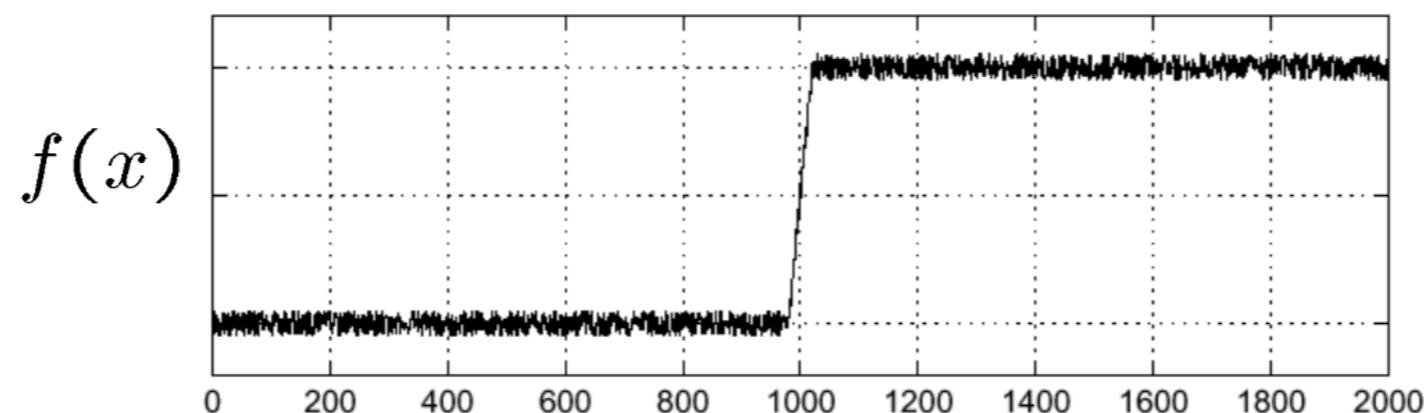
Intensity profile



Source: D. Hoiem

Effect of noise

- Consider a single row or column of the image
 - Plotting intensity as a function of position gives a signal



Where is the edge?

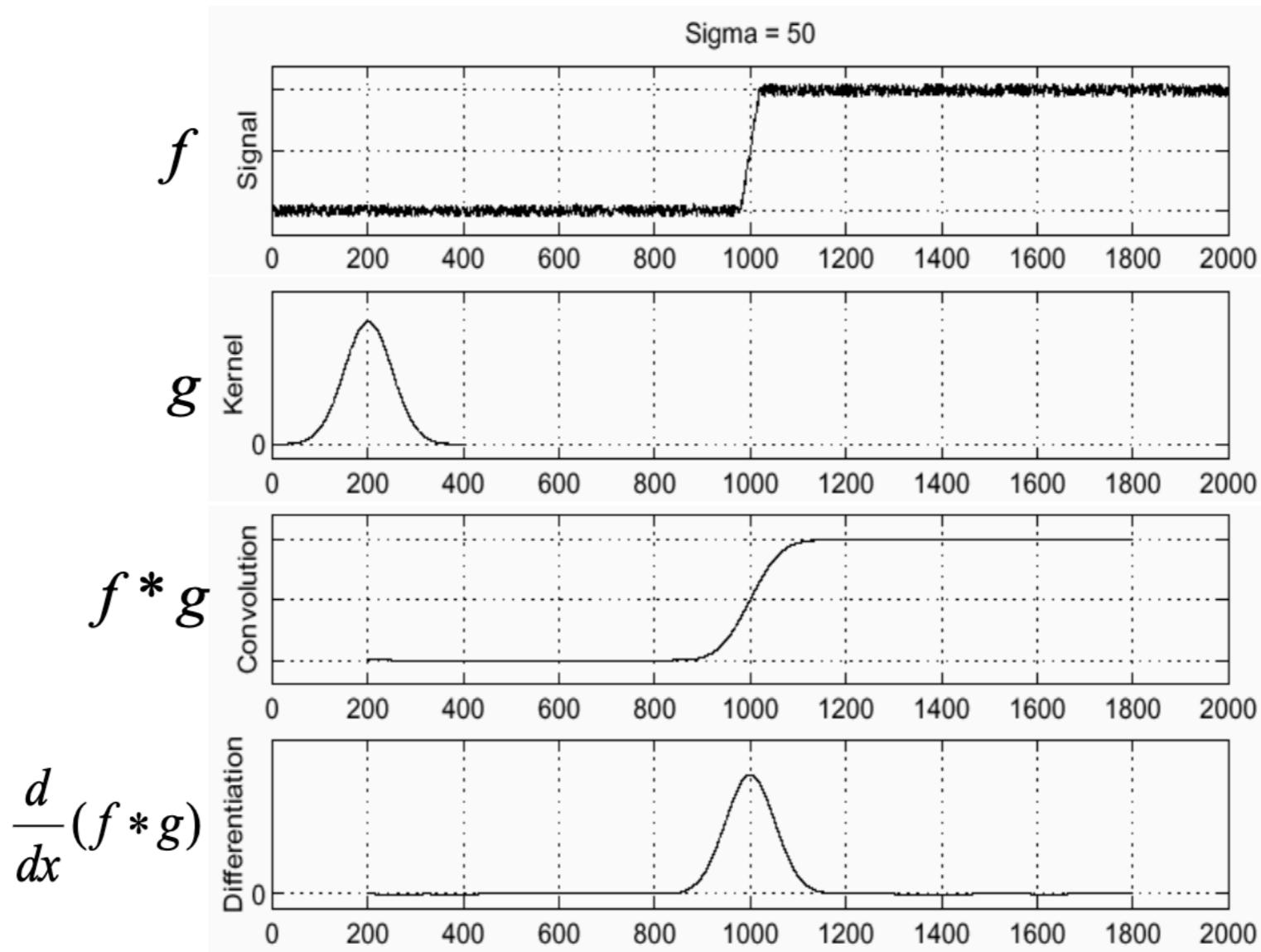
Source: S. Seitz

Effect of noise

- Finite difference filters respond strongly to noise
 - Image noise results in pixels that look very different from their neighbors
 - Generally, the larger the noise the stronger the response
- What is to be done?
 - Smoothing the image should help, by forcing pixels different to their neighbors (=noise pixels?) to look more like neighbors

Source: D. Forsyth

Solution: Smoothing



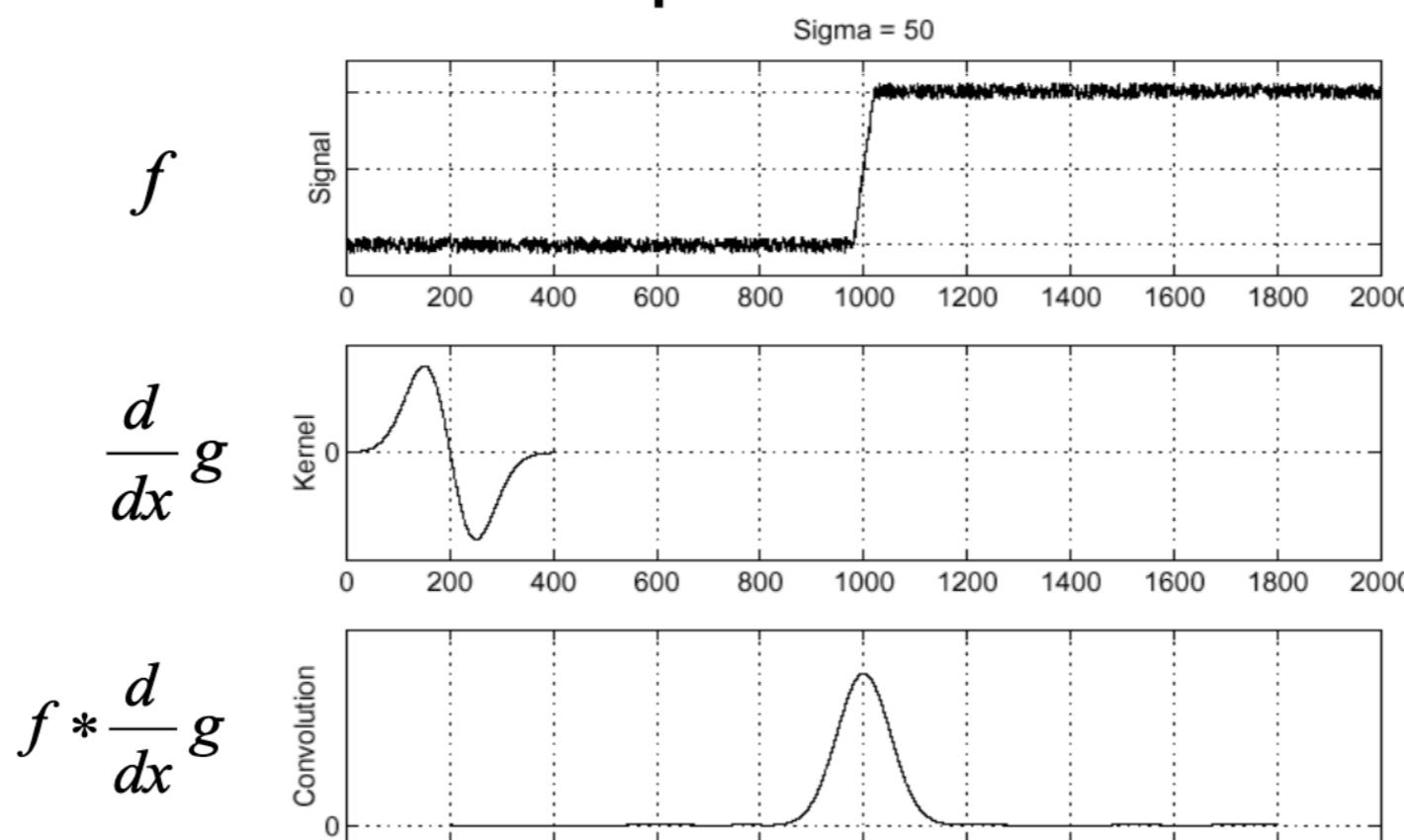
- To find edges, look for peaks in

$$\frac{d}{dx}(f * g)$$

Source: S. Seitz

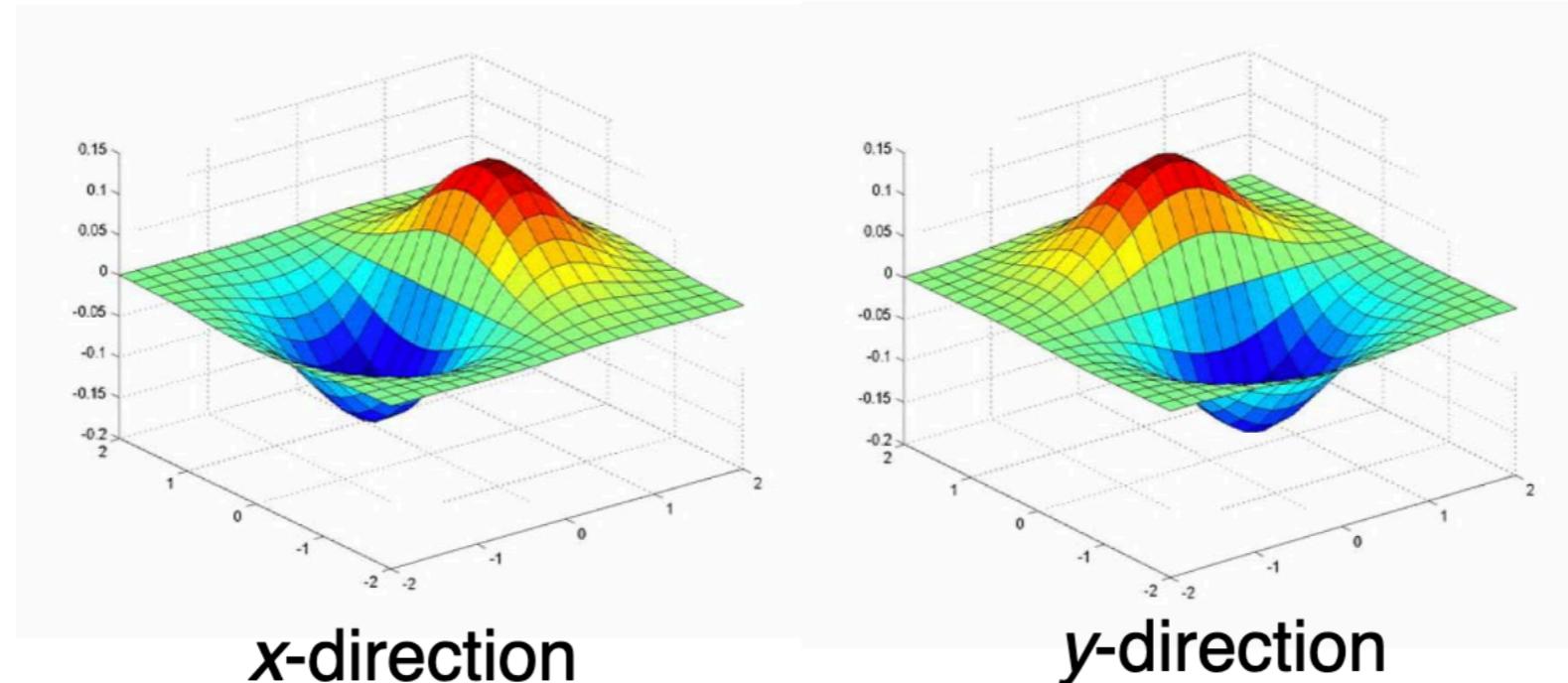
Edge detection via Convolution

- Differentiation is convolution, and convolution is associative: $\frac{d}{dx}(f * g) = f * \frac{d}{dx}g$
- This saves us one operation:



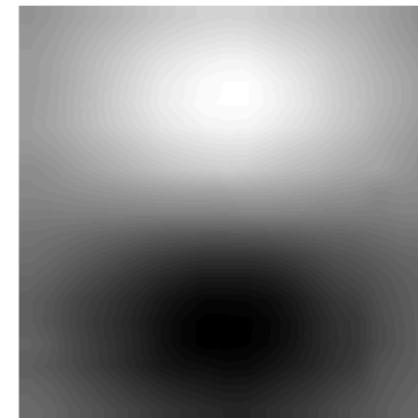
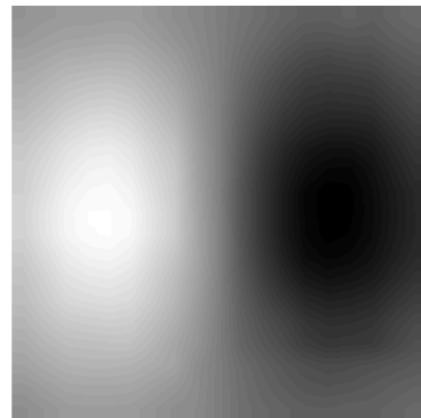
Source: S. Seitz

Derivative of Gaussian filter



x-direction

y-direction



Which one finds horizontal/vertical edges?

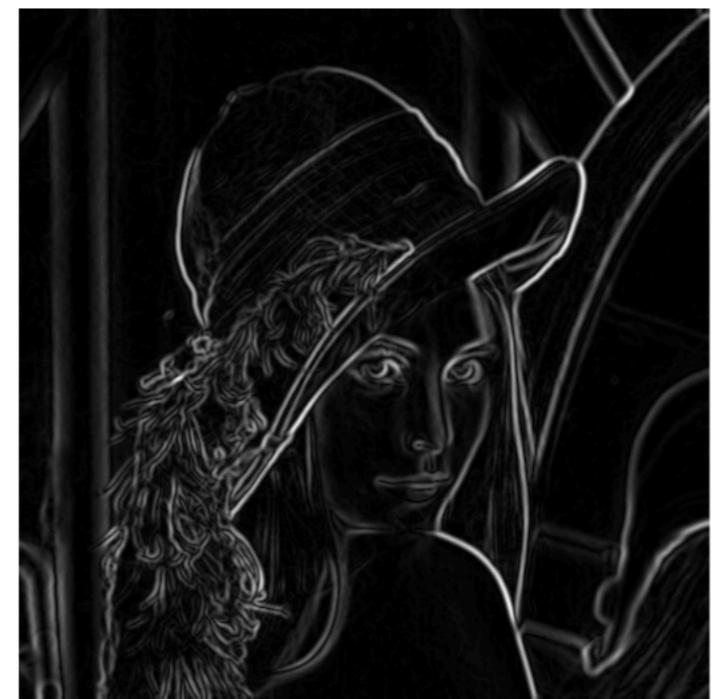
Derivatives of Gaussian filter



X-Derivative of Gaussian



Y-Derivative of Gaussian



Gradient Magnitude

Edge detectors

Note that a Prewitt operator is a box filter convolved with a derivative operator

$$\begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix} = \boxed{\text{Finite diff operator}} * \boxed{\text{Simple box filter}}$$
$$= \begin{bmatrix} -1 & 0 & 1 \end{bmatrix} * \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}$$

Also note: a Sobel operator is a $[1 \ 2 \ 1]$ filter convolved with a derivative operator.

$$\begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} = \boxed{\text{Finite diff operator}} * \boxed{\text{Simple Gaussian}}$$
$$= \begin{bmatrix} -1 & 0 & 1 \end{bmatrix} * \begin{bmatrix} 1 \\ 2 \\ 1 \end{bmatrix}$$

Canny Edge detection

```
1 #import the required libraries
2 import numpy as np
3 import cv2
4 import matplotlib.pyplot as plt
5 %matplotlib inline
6 #read the image
7 image = cv2.imread('coins.jpg')
8 #calculate the edges using Canny edge algorithm
9 edges = cv2.Canny(image,100,200)
10 #plot the edges
11 plt.imshow(edges)
```

Edge Detection.py hosted with ❤ by GitHub

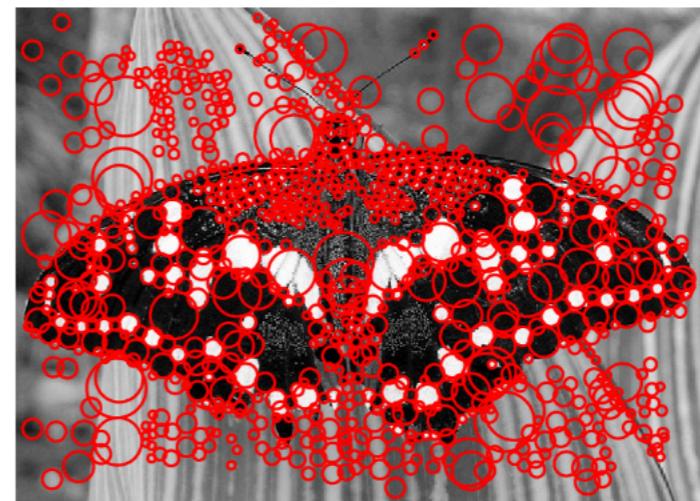
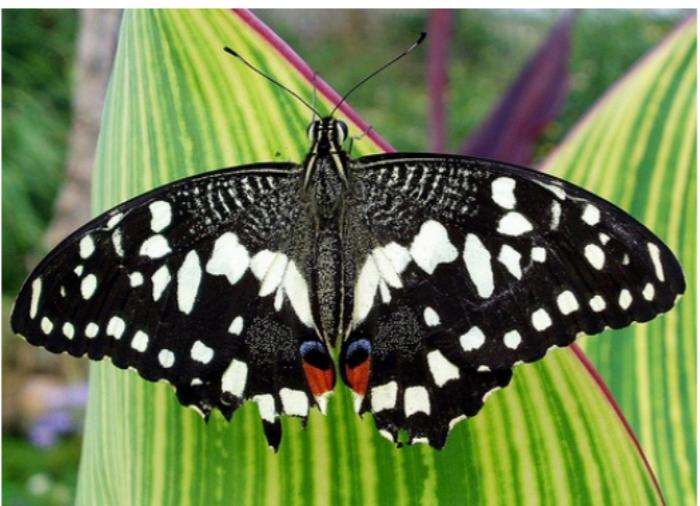
[view raw](#)



J. Canny, [*A Computational Approach To Edge Detection*](#), IEEE Trans. Pattern Analysis and Machine Intelligence, 8:679-714, 1986.

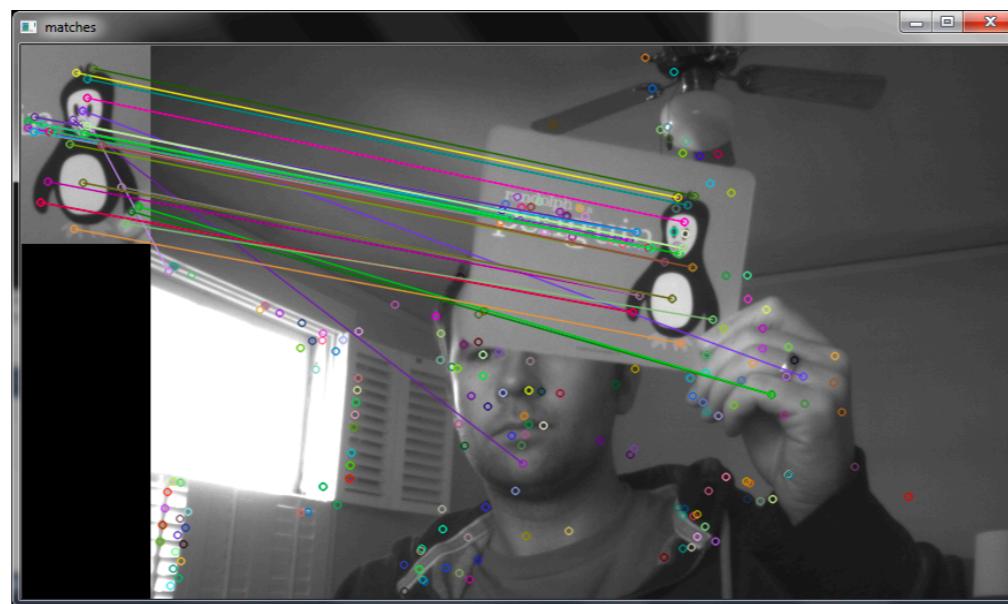
Corner/blob detectors

- Edges are useful as local features, but corners and small areas (blobs) are generally more helpful in computer vision tasks. Blob detectors can be built by extending the basic edge detector idea that we just discussed.



Scale Invariant Feature Transform (SIFT)

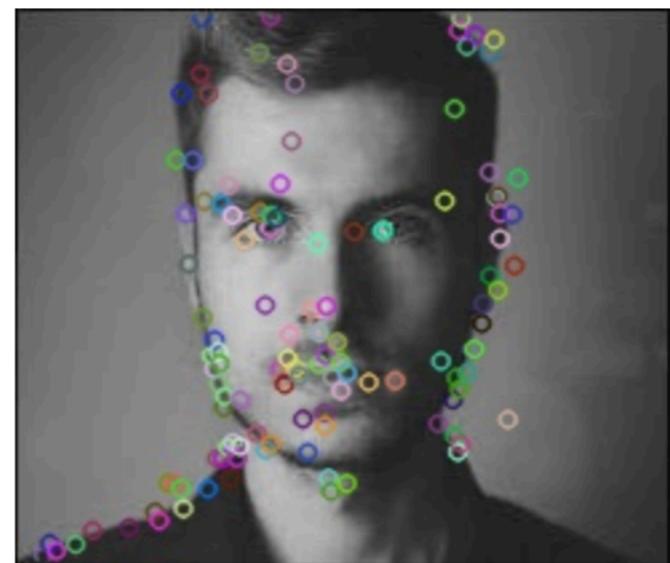
- **Keypoints** are basically the points of interest in an image. Keypoints are analogous to the features of a given image.
- They are locations that define what is interesting in the image. Keypoints are important, because no matter how the image is modified (rotation, shrinking, expanding, distortion), we will always find the same keypoints for the image.



Lowe, David G. "Distinctive image features from scale-invariant keypoints." *International journal of computer vision* 60.2 (2004): 91-110.

SIFT

```
1 #import required libraries
2 import cv2
3 import numpy as np
4 import matplotlib.pyplot as plt
5 %matplotlib inline
6 #show OpenCV version
7 print(cv2.__version__)
8 #read the iamge and convert to grayscale
9 image = cv2.imread('index.png')
10 gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
11 #create sift object
12 sift = cv2.xfeatures2d.SIFT_create()
13 #calculate keypoints and their orientation
14 keypoints,descriptors = sift.detectAndCompute(gray,None)
15 #plot keypoints on the image
16 with_keypoints = cv2.drawKeypoints(gray,keypoints)
17 #plot the image
18 plt.imshow(with_keypoints)
```



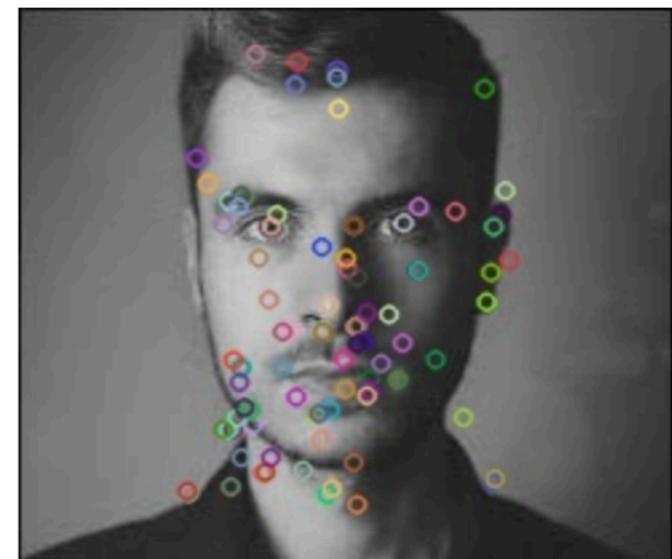
Speeded-Up Robust Features (SURF)

- Speeded-Up Robust Features (SURF) is an enhanced version of SIFT. It works much faster and is more robust to image transformations.
- In SIFT, the scale space is approximated using **Difference of Gaussians** (DoG) while in SURF they use **Laplacian of Gaussian**. The Laplacian kernel works by approximating a second derivative of the image. Hence, it is very sensitive to noise and they apply the **Gaussian kernel** to the image before Laplacian kernel thus giving it the name **Laplacian of Gaussian**.
- In SURF, the Laplacian of Gaussian is calculated using a box filter (kernel). The convolution with box filter can be done in parallel for different scales which is the underlying reason for the enhanced speed of SURF (compared to SIFT).

Bay, Herbert, Tinne Tuytelaars, and Luc Van Gool. "Surf: Speeded up robust features." *European conference on computer vision*. Springer, Berlin, Heidelberg, 2006.

SURF

```
1 #import required libraries
2 import cv2
3 import numpy as np
4 import matplotlib.pyplot as plt
5 %matplotlib inline
6 #show OpenCV version
7 print(cv2.__version__)
8 #read image and convert to grayscale
9 image = cv2.imread('index.png')
10 gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
11 #instantiate surf object
12 surf = cv2.xfeatures2d.SURF_create(400)
13 #calculate keypoints and their orientation
14 keypoints,descriptors = surf.detectAndCompute(gray,None)
15
16 with_keypoints = cv2.drawKeypoints(gray,keypoints)
17
18 plt.imshow(with_keypoints)
```



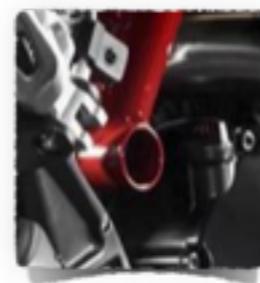
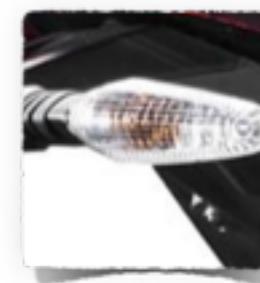
Bag of Words

Dictionary Learning:
Learn Visual Words using clustering

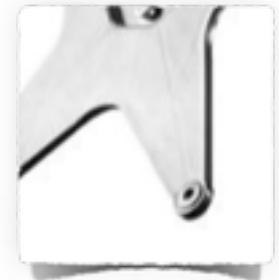
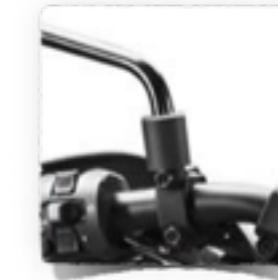
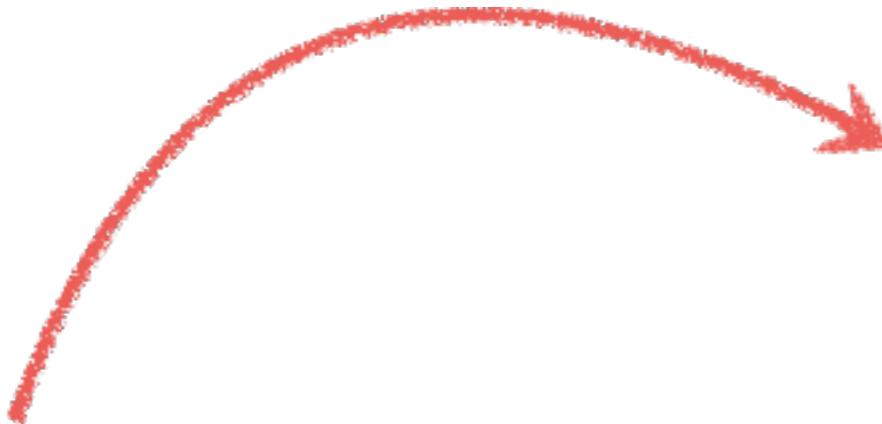
Encode:
build Bags-of-Words (BOW) vectors
for each image

Classify:
Train and test data using BOWs

Which object do these parts belong to?



Some local feature are very informative



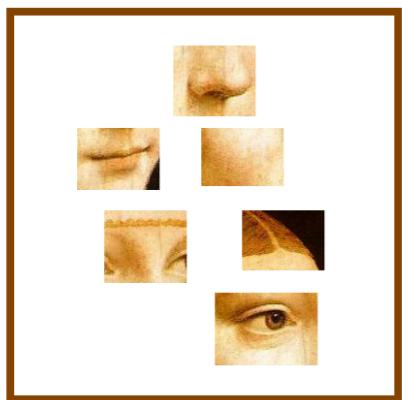
An object as

a collection of local features
(bag-of-words)

- deals well with occlusion
- scale invariant
- rotation invariant

BOW

- Extract features (e.g., SIFT)



- Learn a visual dictionary



Image Features

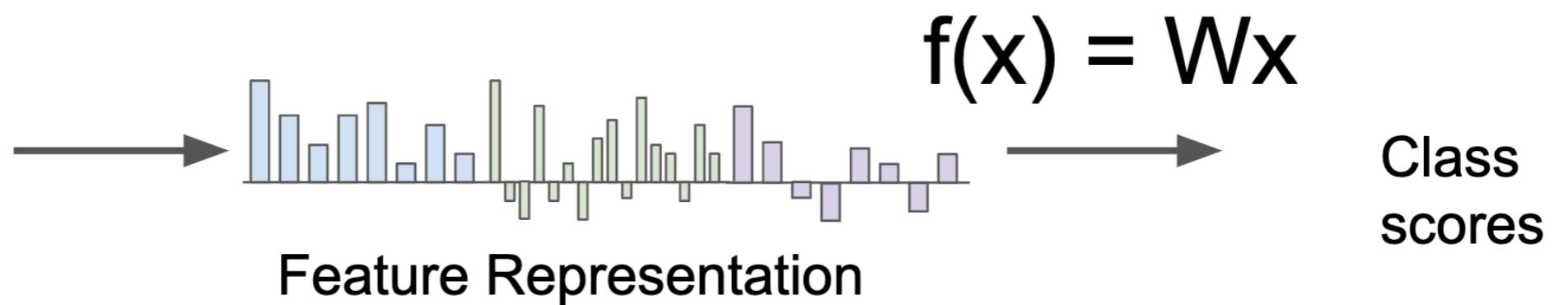
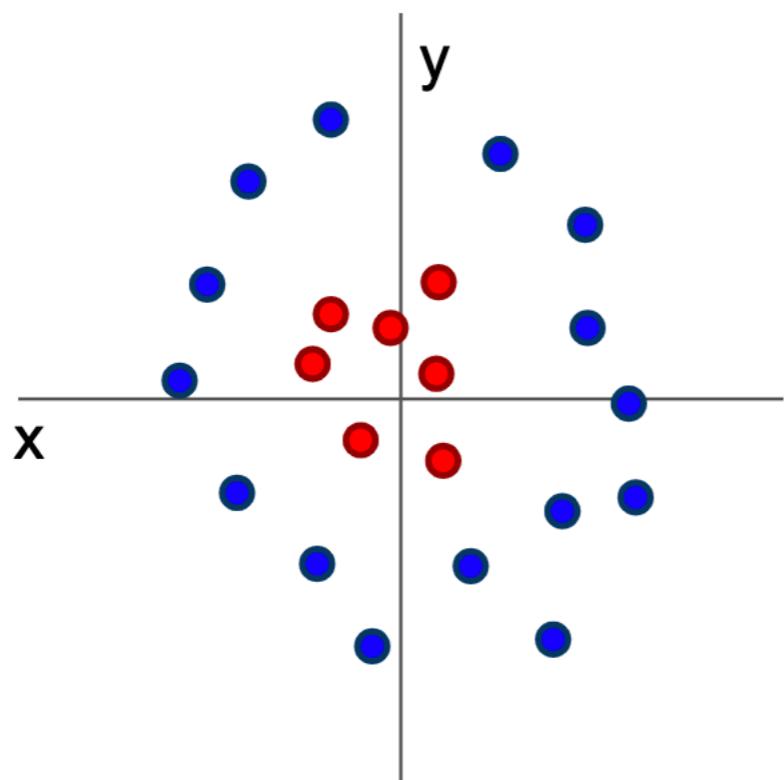
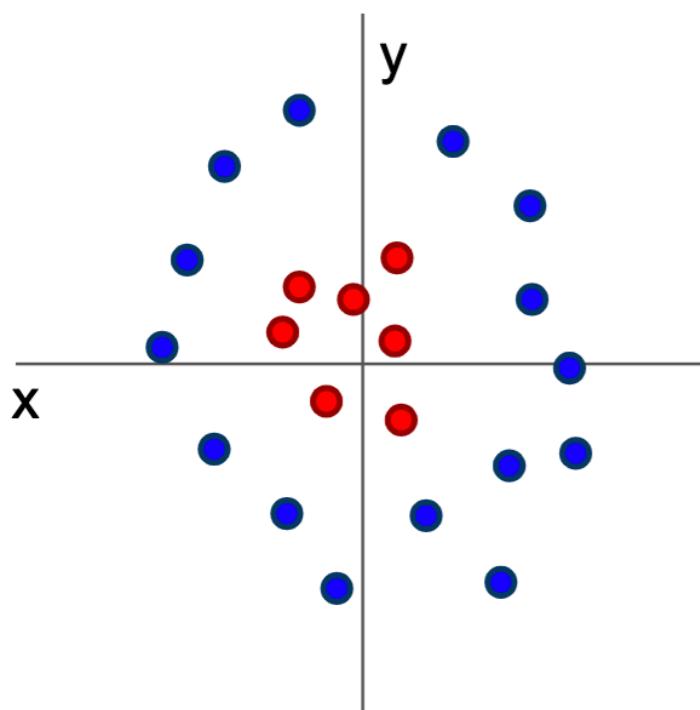


Image Features: Motivation

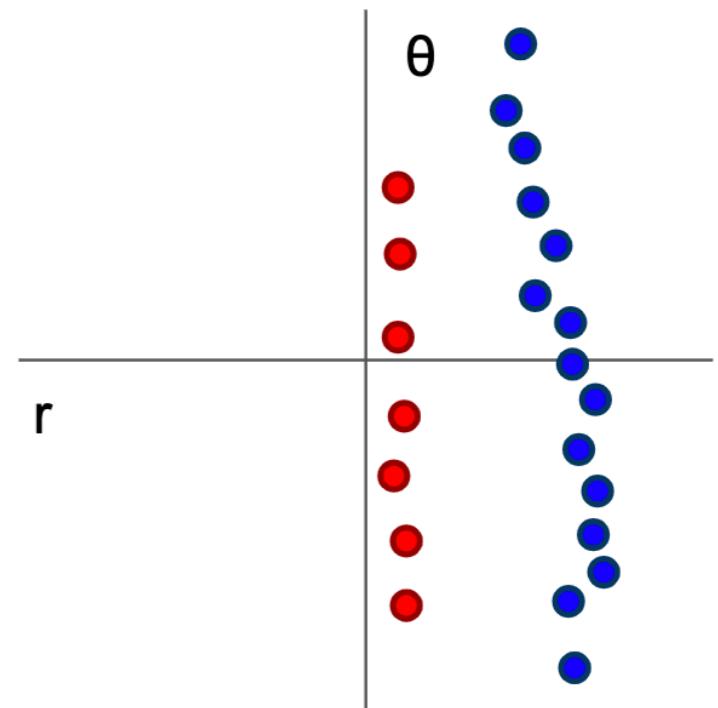


Cannot separate red
and blue points with
linear classifier

Image Features: Motivation



$$f(x, y) = (r(x, y), \theta(x, y))$$



Cannot separate red
and blue points with
linear classifier

After applying feature
transform, points can
be separated by linear
classifier

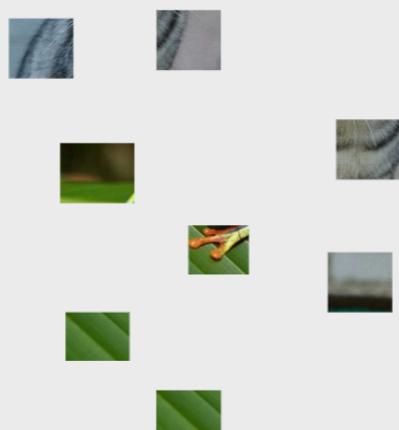
Image Features

Example: Bag of Words

Step 1: Build codebook



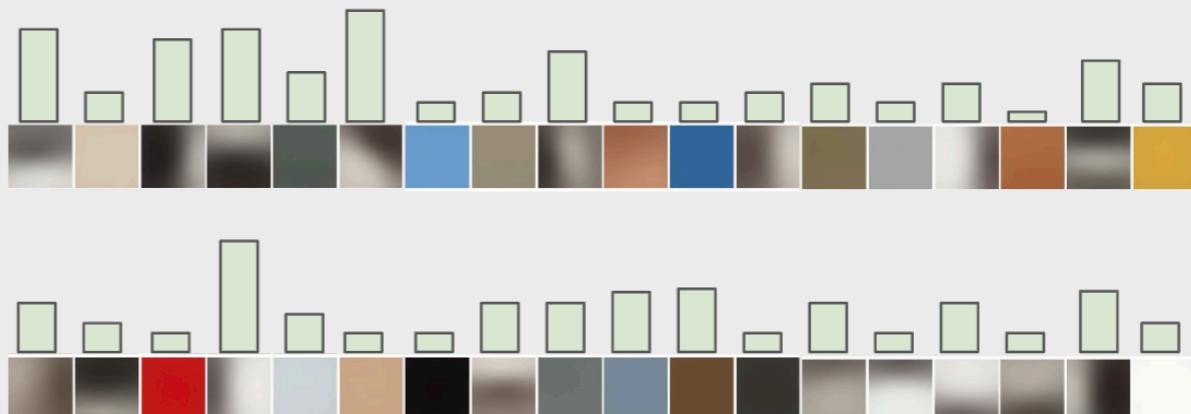
Extract random patches



Cluster patches to form “codebook” of “visual words”



Step 2: Encode images



Fei-Fei and Perona, "A bayesian hierarchical model for learning natural scene categories", CVPR 2005

Image Features

Aside: Image Features

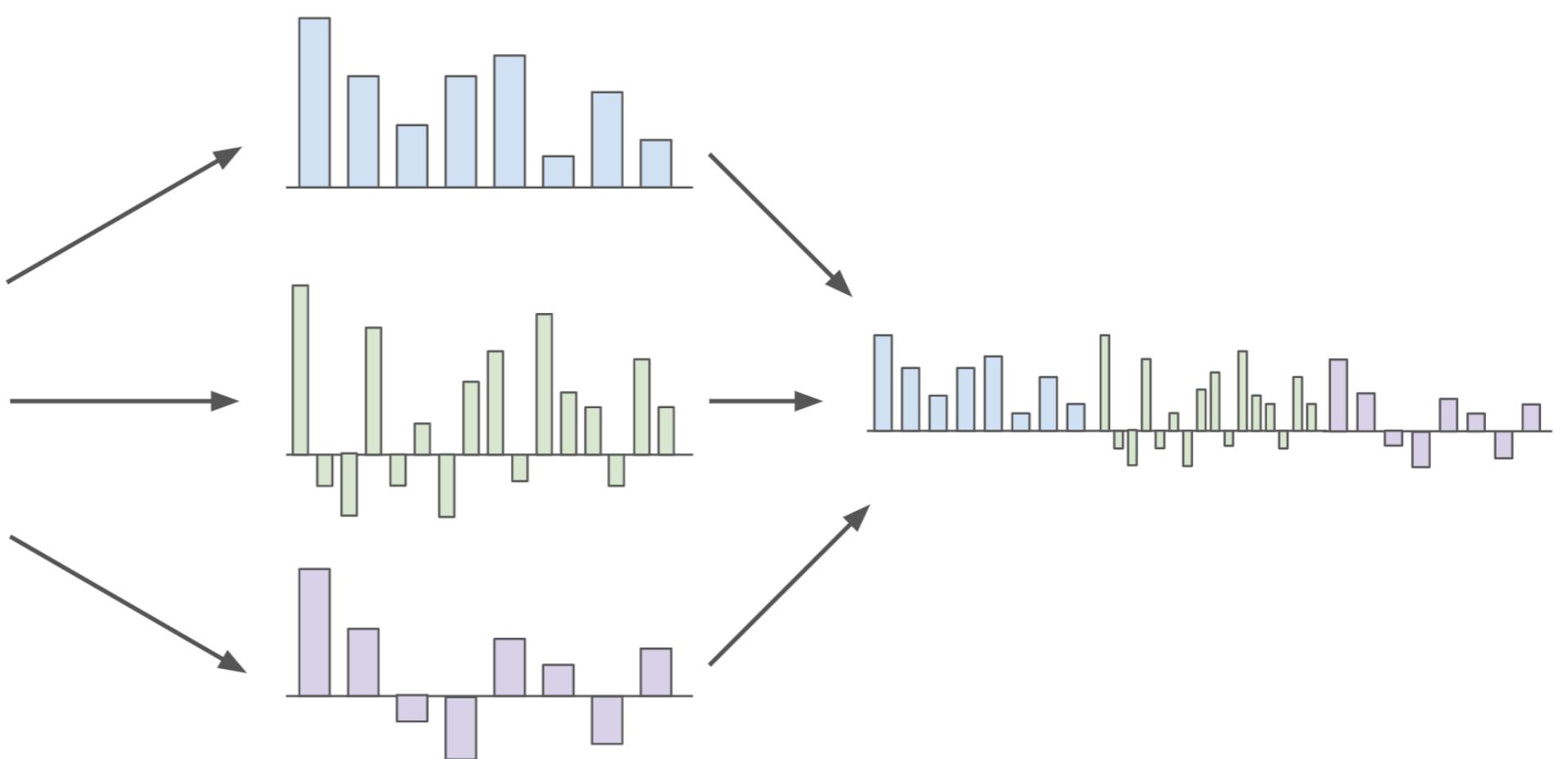
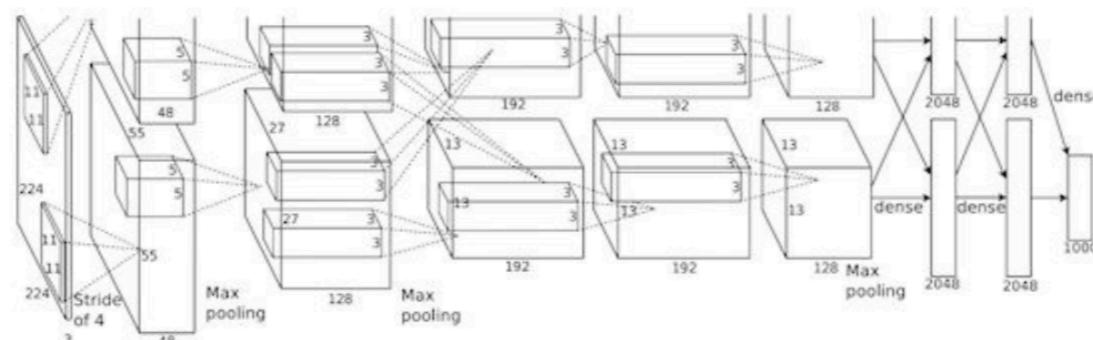


Image features vs ConvNets



training

Krizhevsky, Sutskever, and Hinton, "Imagenet classification with deep convolutional neural networks", NIPS 2012.
Figure copyright Krizhevsky, Sutskever, and Hinton, 2012.
Reproduced with permission.

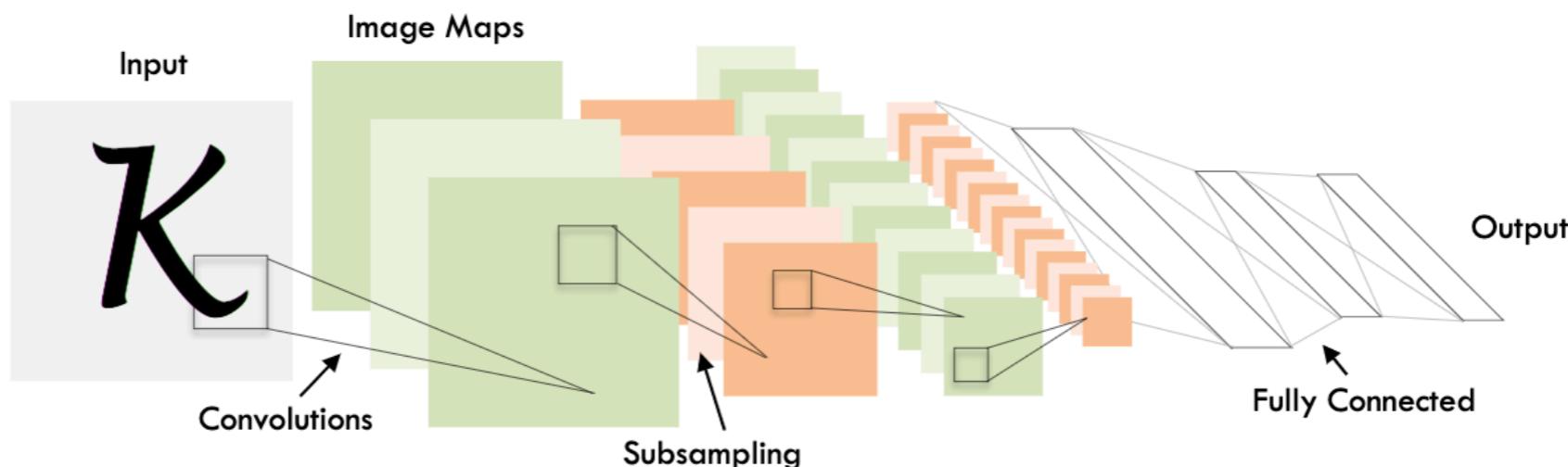


training

10 numbers giving scores for classes



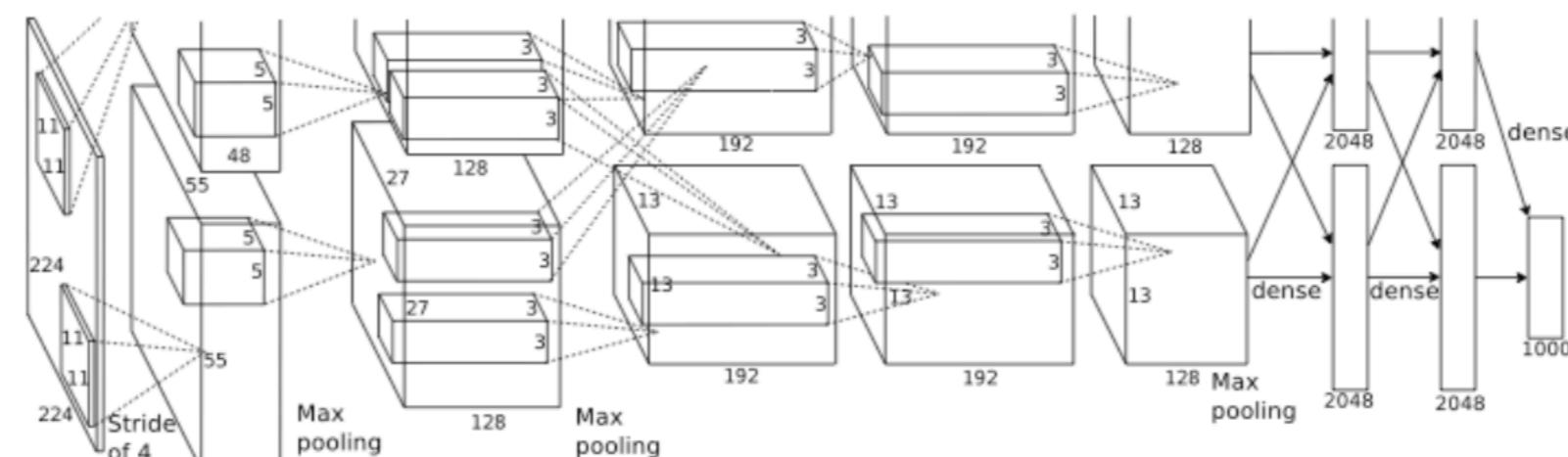
1998
LeCun et al.



of transistors
 10^6
pentium® II

of pixels used in training
 10^7 **NIST**

2012
Krizhevsky et al.



of transistors
 10^9



of pixels used in training
 10^{14} **IMAGENET**

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

ImageNet

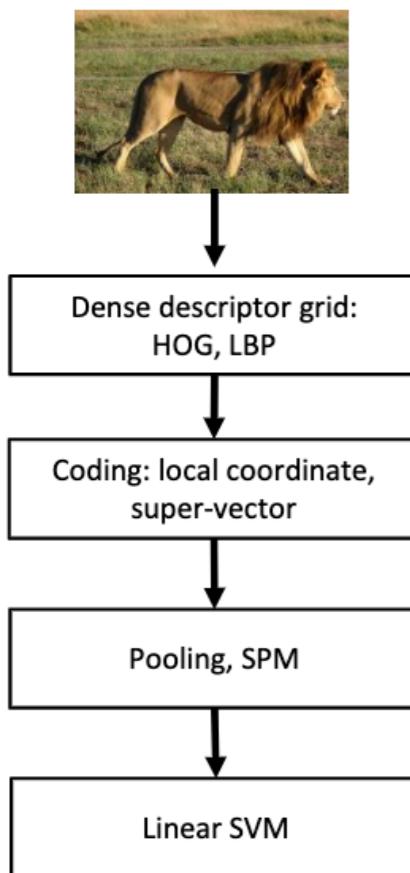
- **ImageNet** is an image database organized according to the WordNet hierarchy (currently only the nouns), in which each node of the hierarchy is depicted by hundreds and thousands of images. Currently we have an average of over five hundred images per node



IMAGENET Large Scale Visual Recognition Challenge

Year 2010

NEC-UIUC

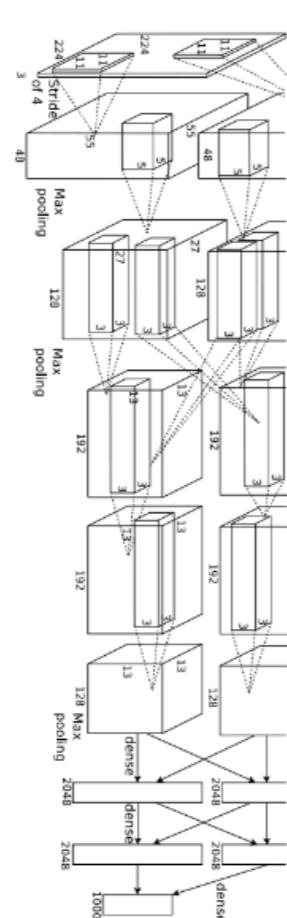


[Lin CVPR 2011]

Lion image by Swissfrog is licensed under [CC BY 3.0](https://creativecommons.org/licenses/by/3.0/)

Year 2012

SuperVision

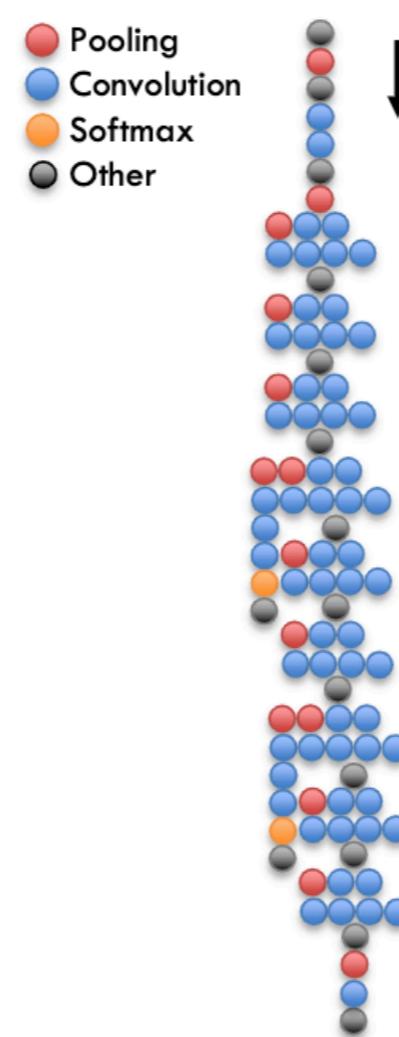


[Krizhevsky NIPS 2012]

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

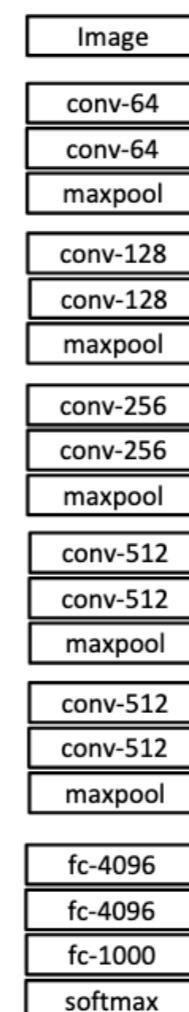
Year 2014

GoogLeNet



[Szegedy arxiv 2014]

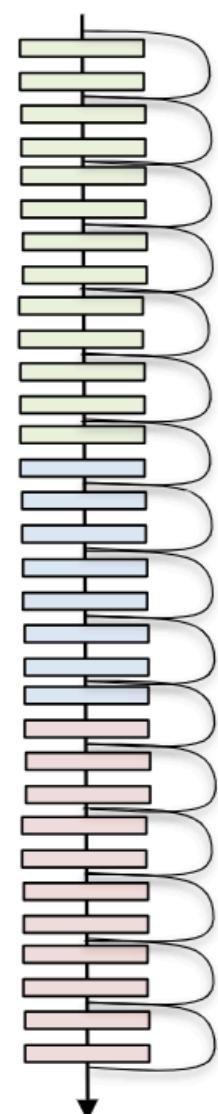
VGG



[Simonyan arxiv 2014]

Year 2015

MSRA



[He ICCV 2015]

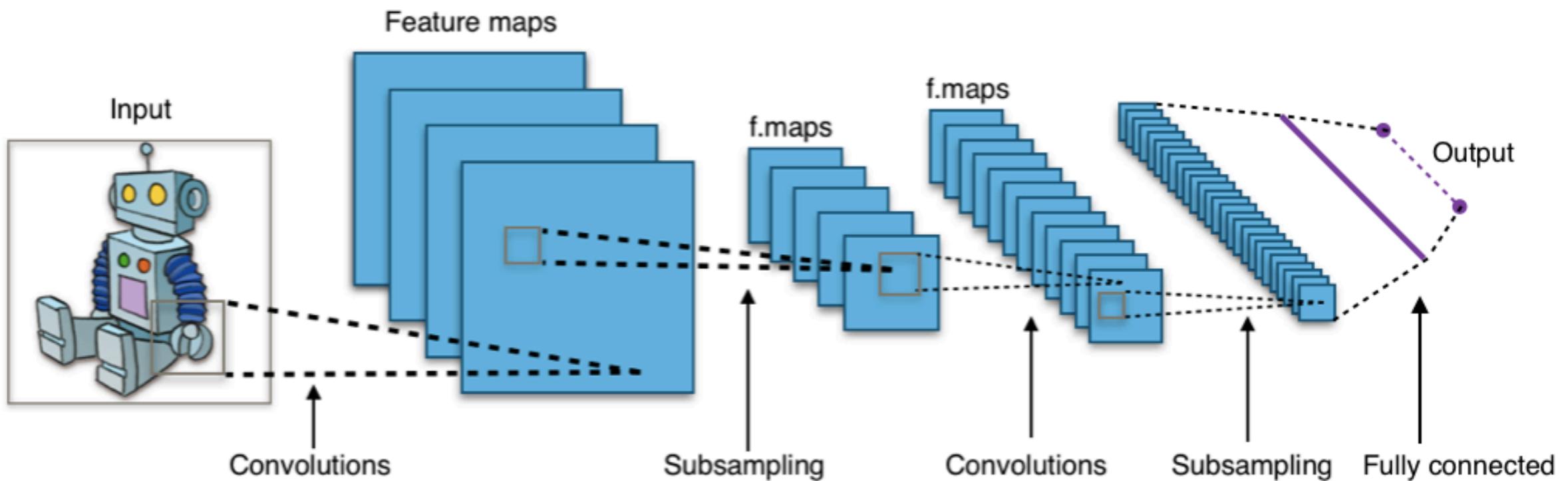
AlexNet



Figure 3: 96 convolutional kernels of size $11 \times 11 \times 3$ learned by the first convolutional layer on the $224 \times 224 \times 3$ input images. The

“AlexNet is considered one of the most influential papers published in computer vision, having spurred many more papers published employing CNNs and GPUs to accelerate deep learning.”

Convolution Neural Networks



We will learn about it on Thursday!

Conferences focusing on CV

- **CVPR** : IEEE/CVF Conference on Computer Vision and Pattern Recognition
<http://cvpr2020.thecvf.com/>
- **ICCV** : IEEE/CVF International Conference on Computer Vision
<http://iccv2019.thecvf.com/>
- **ACMMM** : ACM International Conference on Multimedia
<https://www.acmmm.org/2020/>
- CV is one of the main topics of the major machine learning and AI conferences such as:
AAAI, IJCAI, ICML, NEURIPS, ...