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/

Mila

Université de Montréal

Announcement

- Assignment 3 is available on grade scope:
    - 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

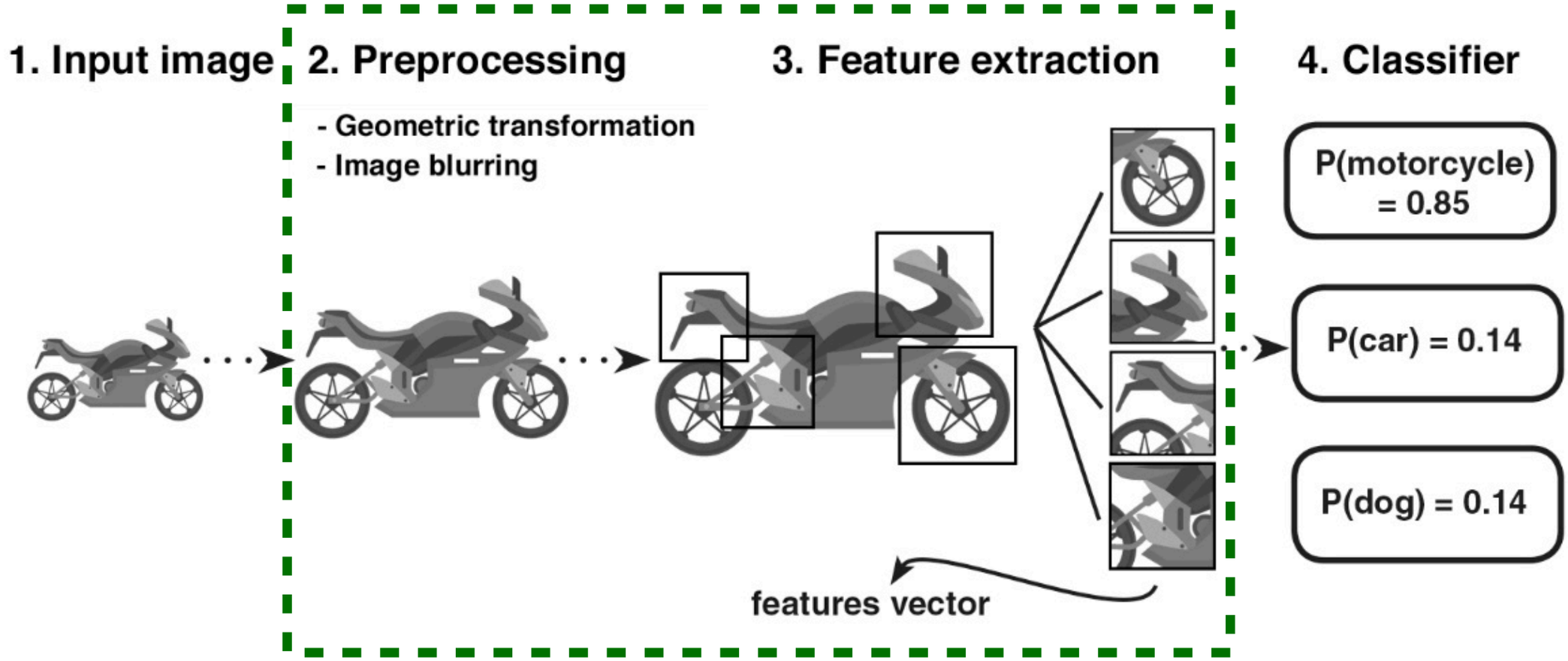
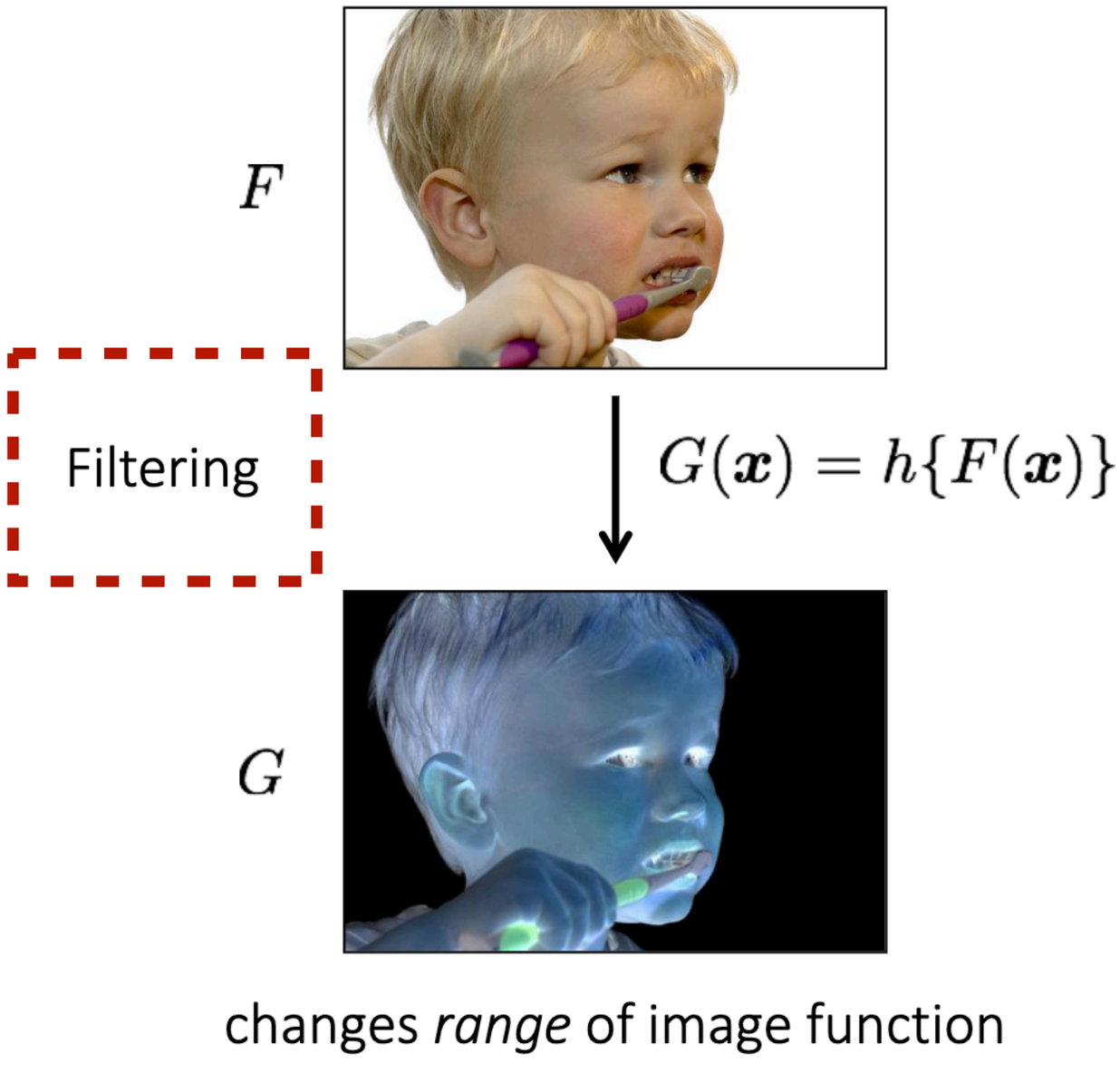
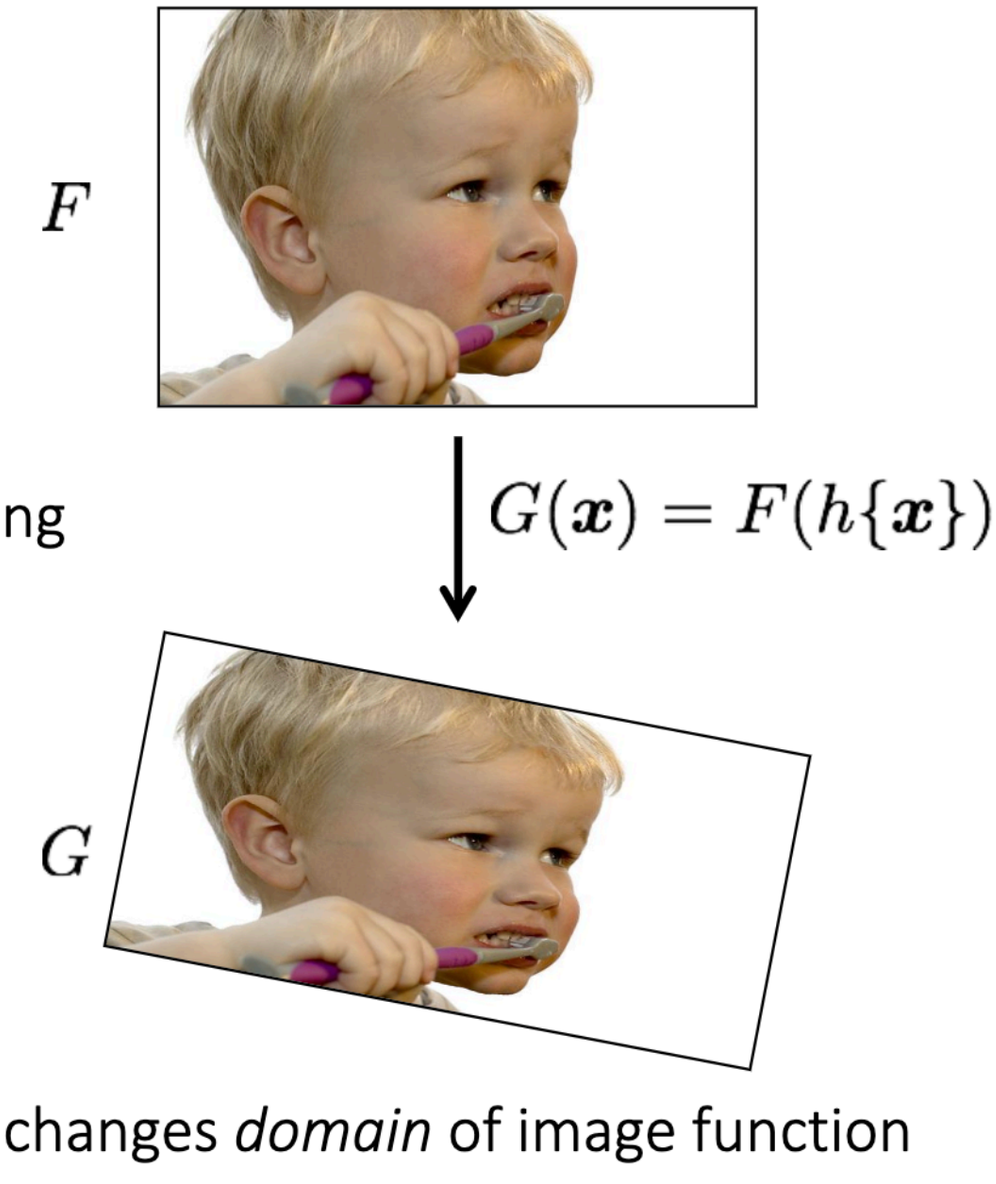

1. Input image
2. Preprocessing
- Geometric transformation
- Image blurring
3. Feature extraction
features vector
4. Classifier
P(motorcycle) = 0.85
P(car) = 0.14
P(dog) = 0.14

Image transformations



changes *range* of image function



changes *domain* of image function

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

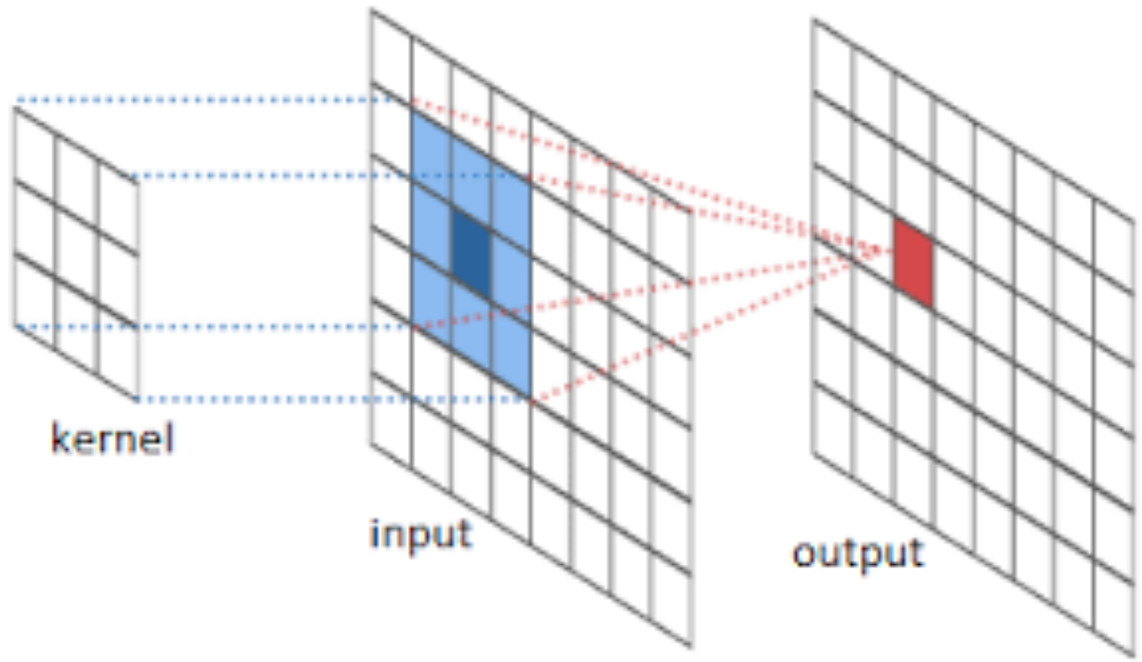
Distributive: $(f + g) * h = f * h + g * h$

Linear: $(a\,f + b\,g) * h = a\,f * h + b\,g * h$

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

Linea

Shift

Differ

$\frac{\partial}{\partial x}$ $\frac{\partial}{\partial x}$

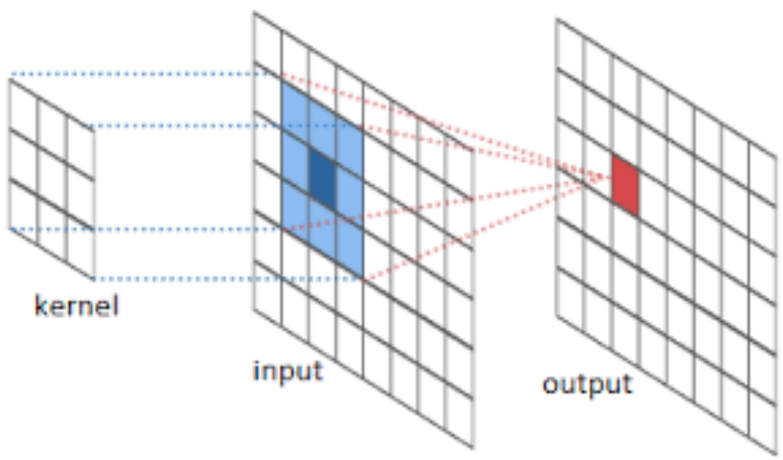
All of computer vision is convolutions (basically)

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**.

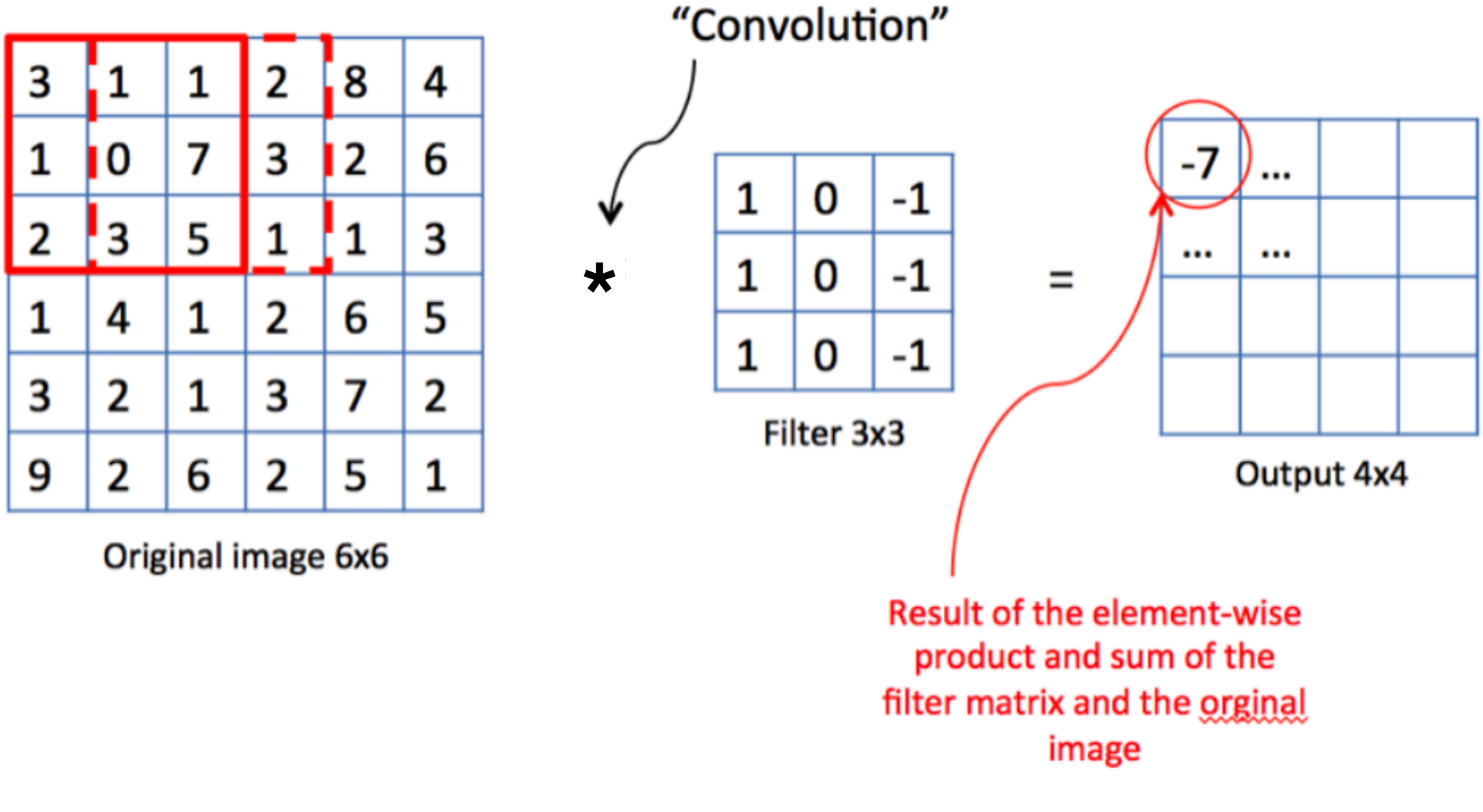
- Convolution in 2D operates on **two images**, with one functioning as **the input image** and the other, called the **kernel**, serving as a **filter.**

- 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.



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 ← $(f * g)(x, y)$

filter ← $f(i, j)$

input image ← $I(x - i, y - j)$

Convolution for 2D discrete signals

Definition of filtering as convolution:

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

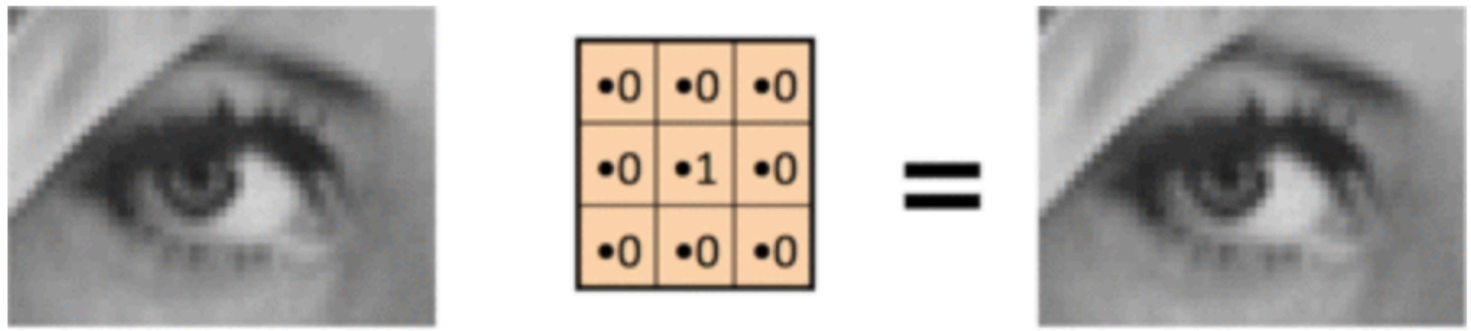
filtered image ↗ | filter ↖ | input image ↖

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

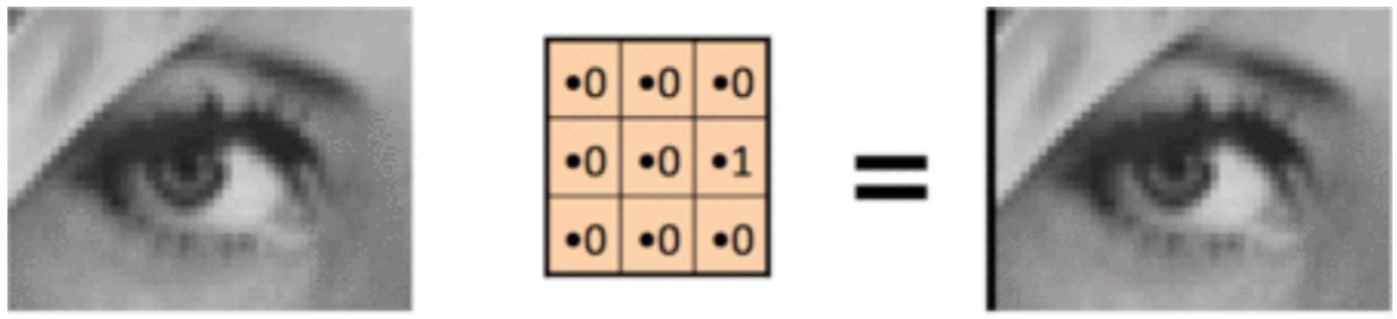
Convolution Examples



Original

Filtered
(no change)

Shifted right by one pixel:



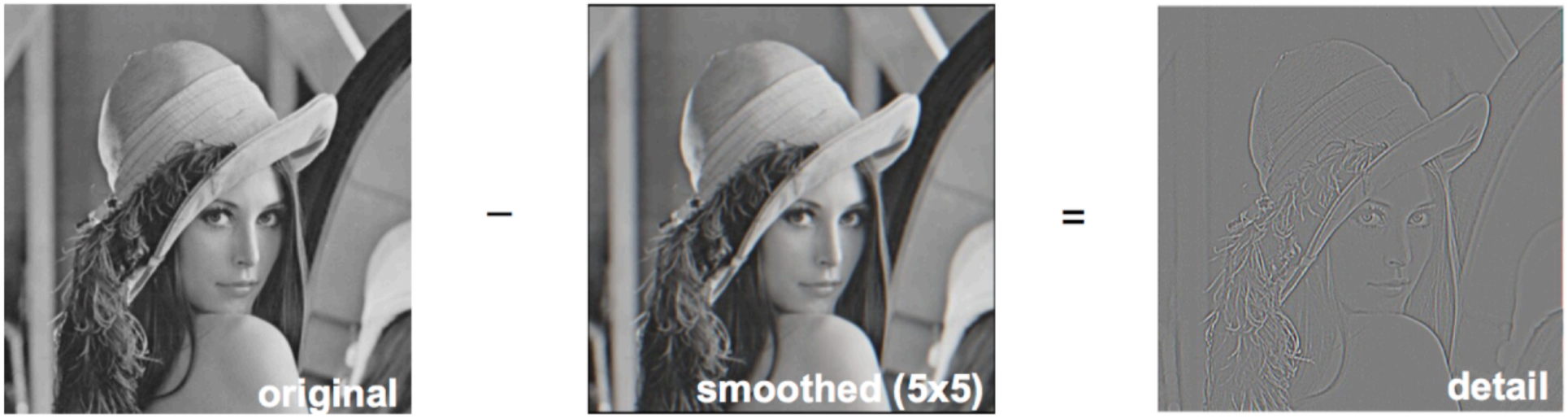
Original

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"

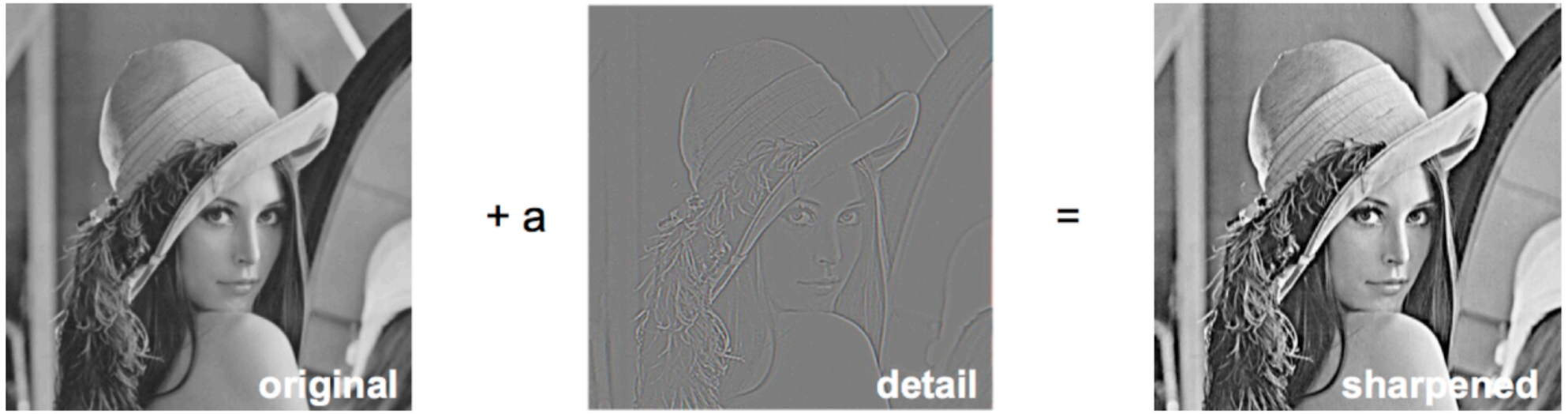


$$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix} - \frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

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



$$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix} + \begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix} - \frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 0 \end{bmatrix} - \frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

Convolution Examples

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

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

original **sigma = 3** **sigma = 10**

Convolution vs Correlation

Definition of discrete 2D **convolution**:

notice the flip

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

Definition of discrete 2D **correlation**:

notice the lack of a flip

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

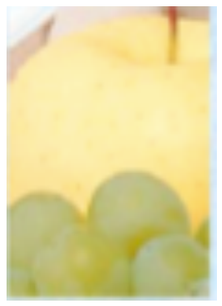
- Most of the time won't matter, because our kernels will be symmetric.

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 the 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.



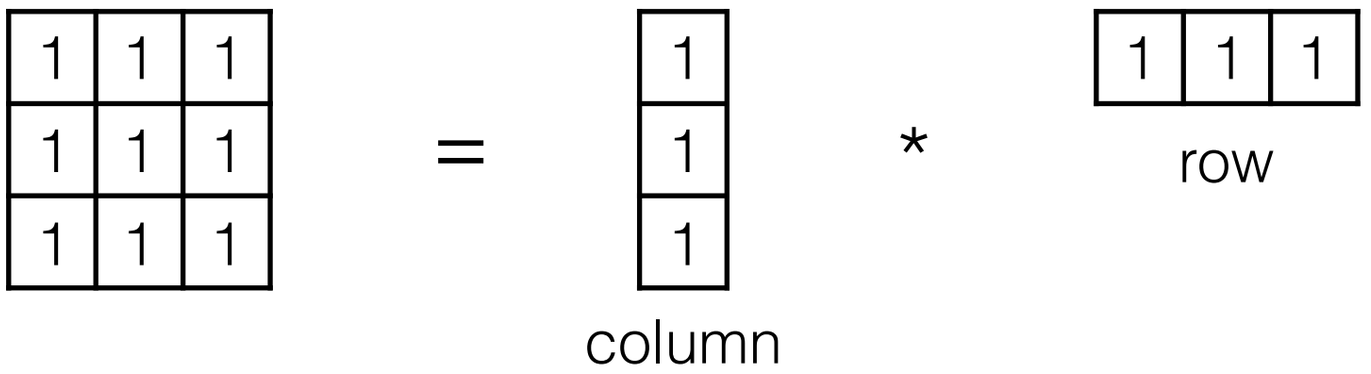
Template

Original image

Separable filters

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

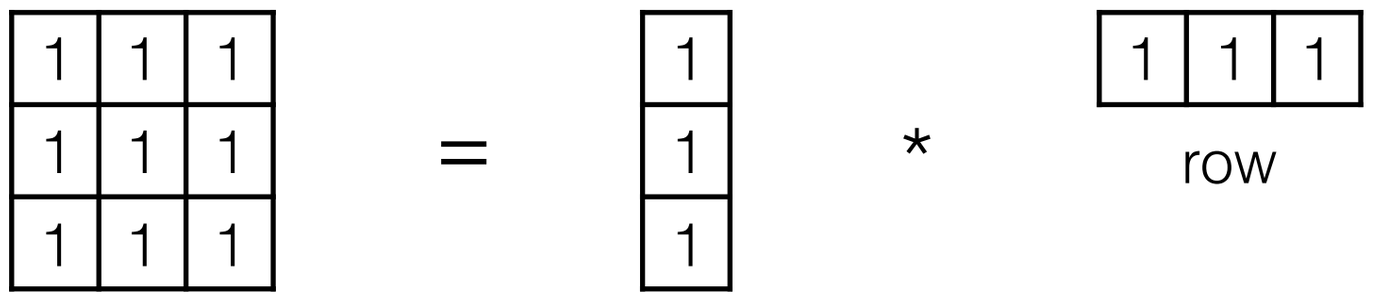
example:
box filter



Separable filters

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

example:
box filter

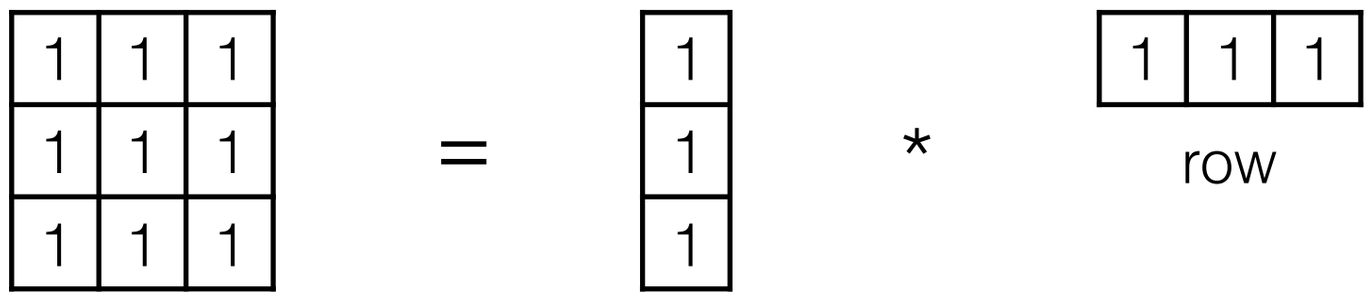


2D convolution with a separable filter is equivalent to two 1D convolutions (with the “column” and “row” filters).

Separable filters

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

example:
box filter



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

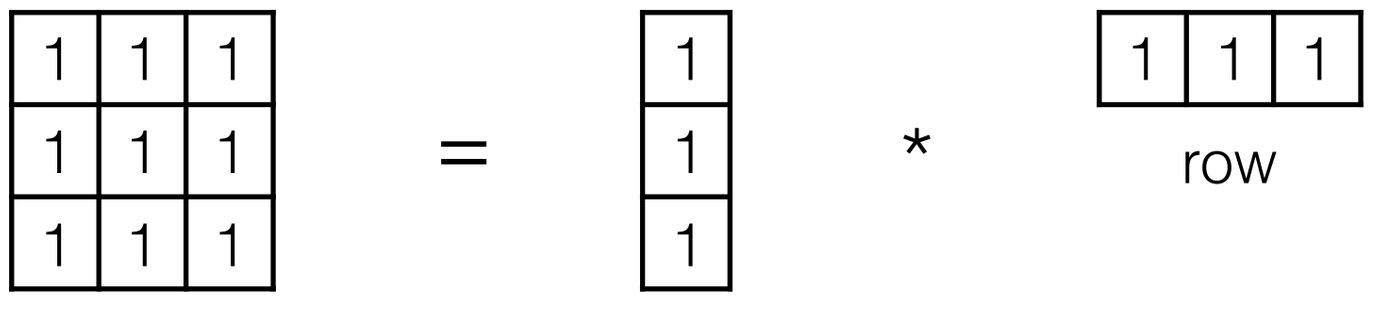
If the image has M x M pixels and the filter kernel has size N x N:

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



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

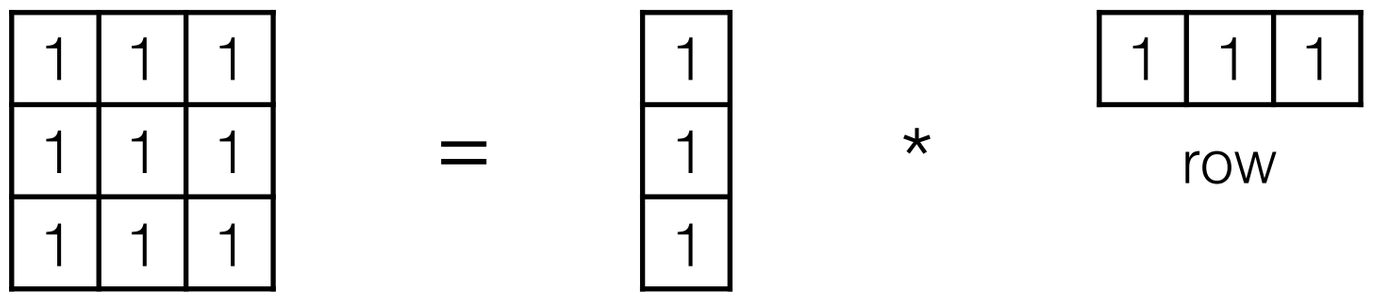
If the image has M x M pixels and the filter kernel has size N x N:

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



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

If the image has M x M pixels and the filter kernel has size N x N:

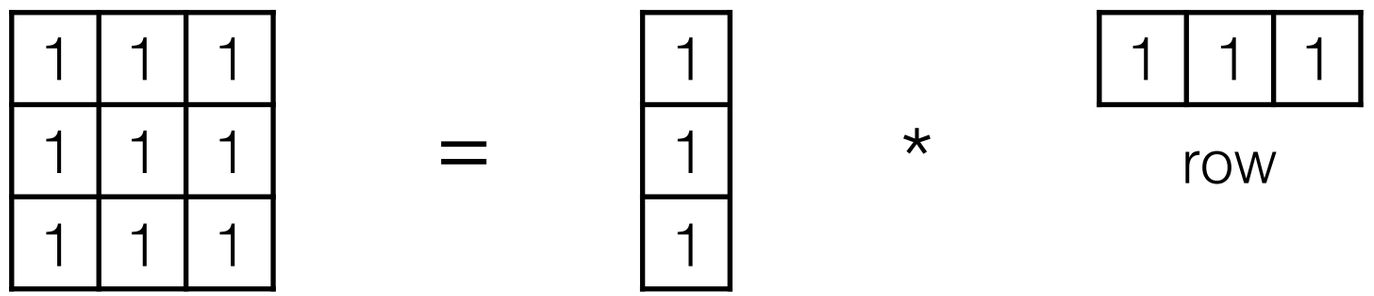
What is the cost of convolution with a non-separable filter? **$M^2 \times N^2$**

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



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

If the image has M x M pixels and the filter kernel has size N x 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

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

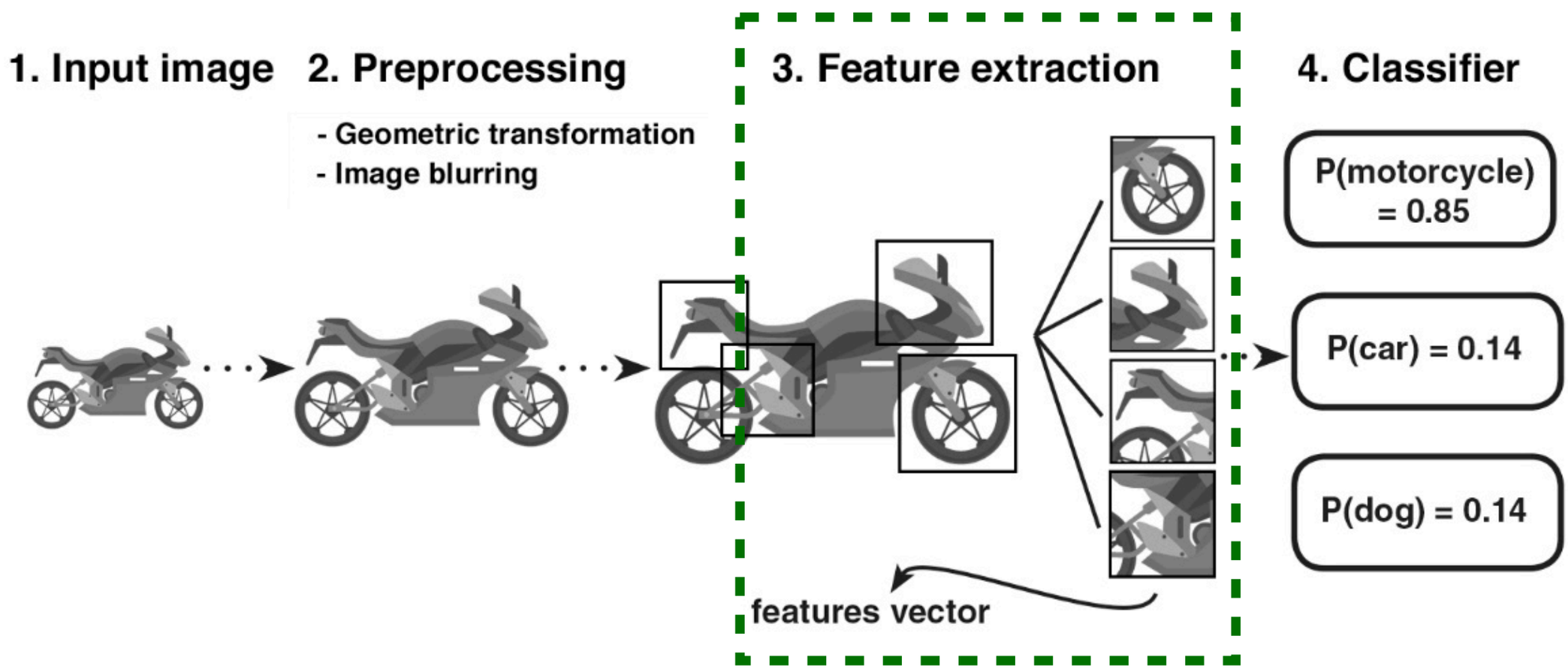
- **Sober operator** which is used commonly for edge detection.

Simple Gaussian

Finite diff operator

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

CV Pipeline

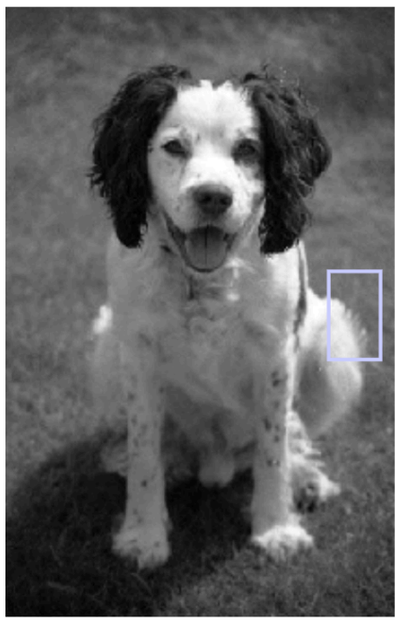

1. Input image
2. Preprocessing
- Geometric transformation
- Image blurring
3. Feature extraction
4. Classifier
P(motorcycle) = 0.85
P(car) = 0.14
P(dog) = 0.14
features vector

Edge detection

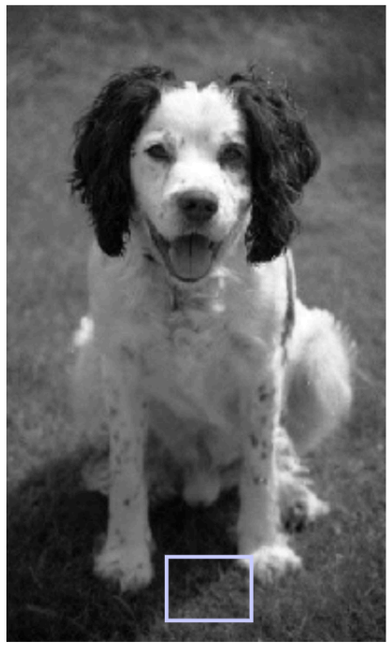
- Edges are the points in an image where the image brightness changes sharply or has discontinuities. Such discontinuities generally correspond to:
    - Discontinuities in depth
    - Discontinuities in surface orientation
    - Changes in material properties
    - Variations in scene illumination
- 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.

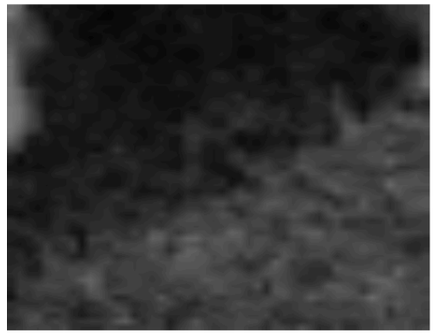
Edge detection

Boundaries of objects

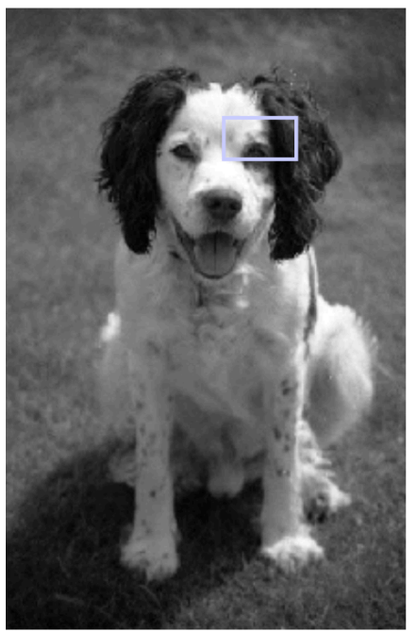


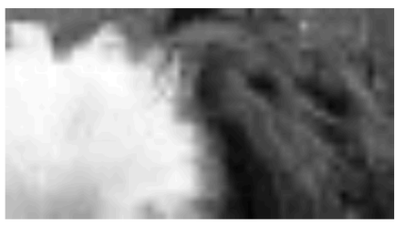
Boundaries of Lighting





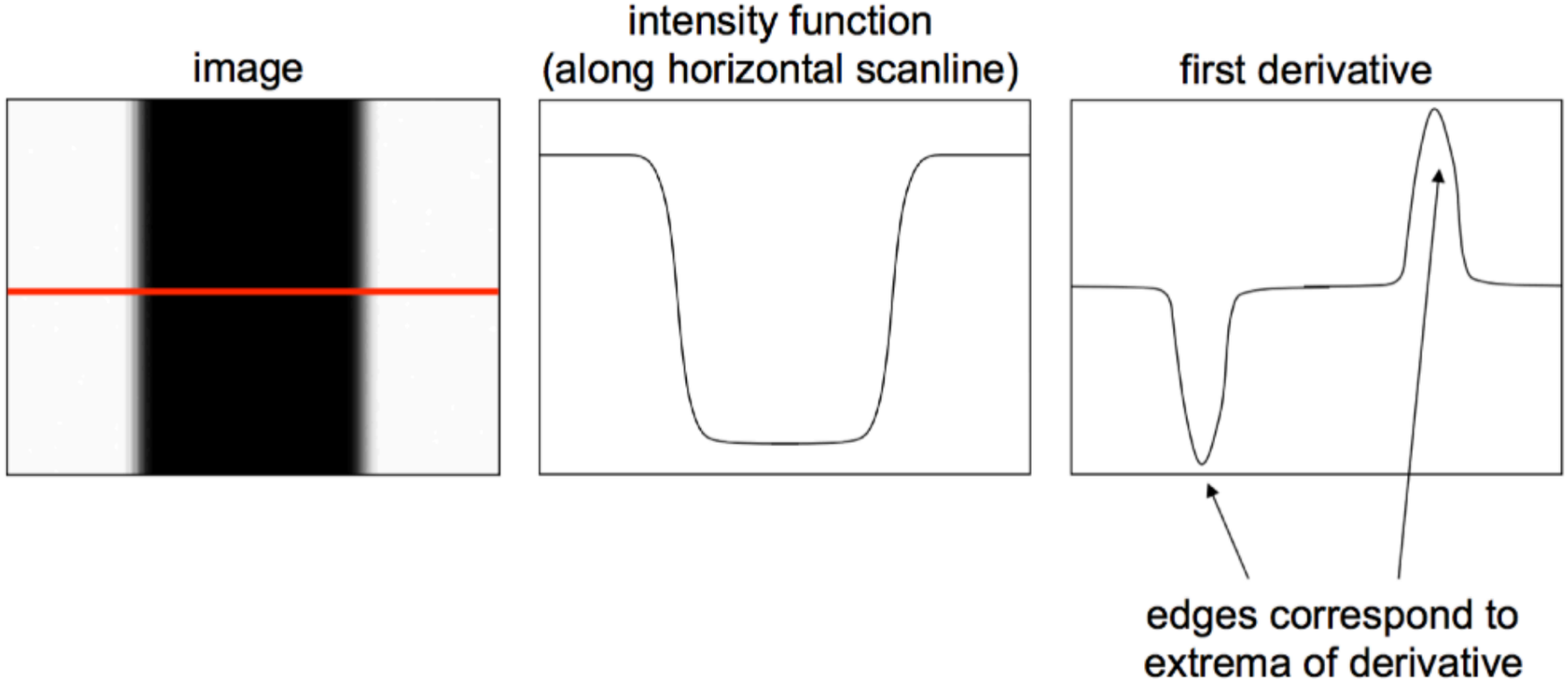
Boundaries of Material Properties





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:

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

For discrete data, we can approximate using finite differences:

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

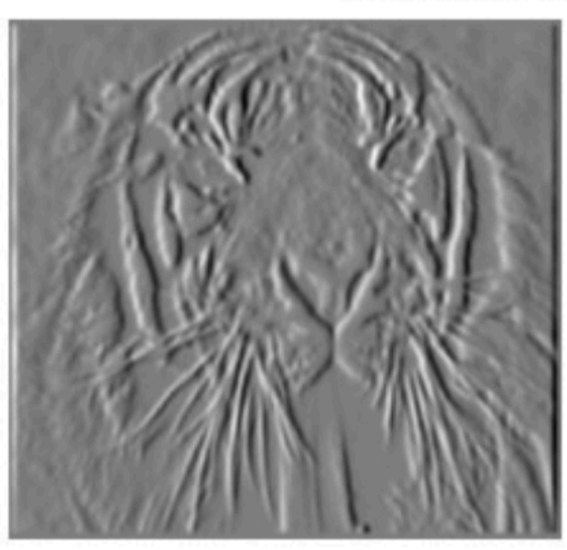
To implement above as convolution, what would be the associated filter?

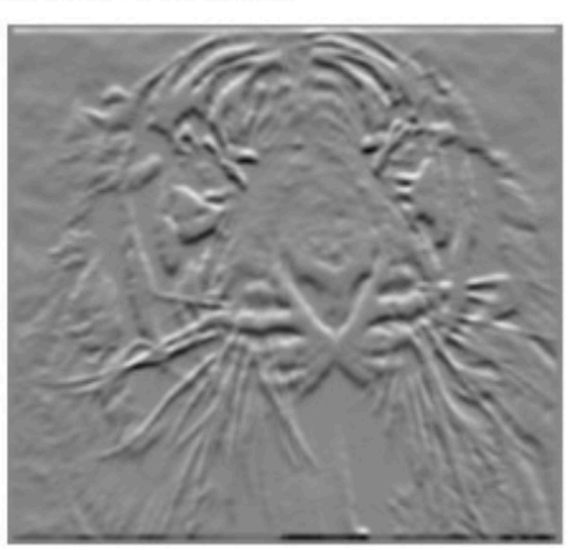
Source: K. Grauman

Partial derivatives of an image

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

-1	1





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

-1
1

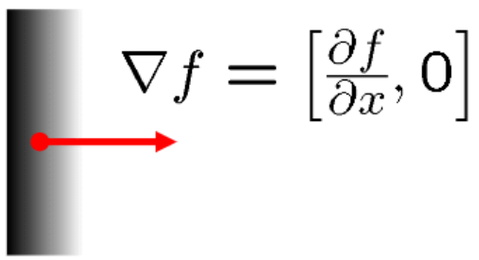
or

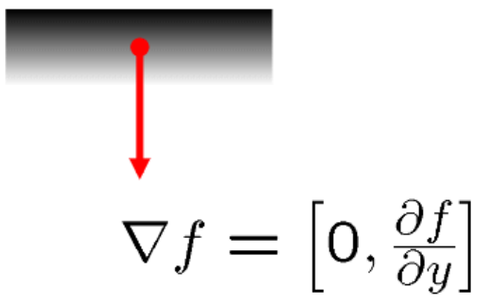
1
-1

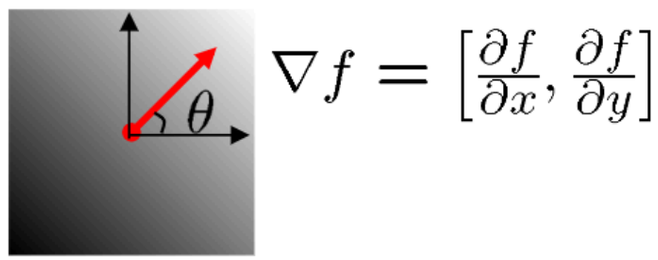
Which shows changes with respect to x?

Image Gradient

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







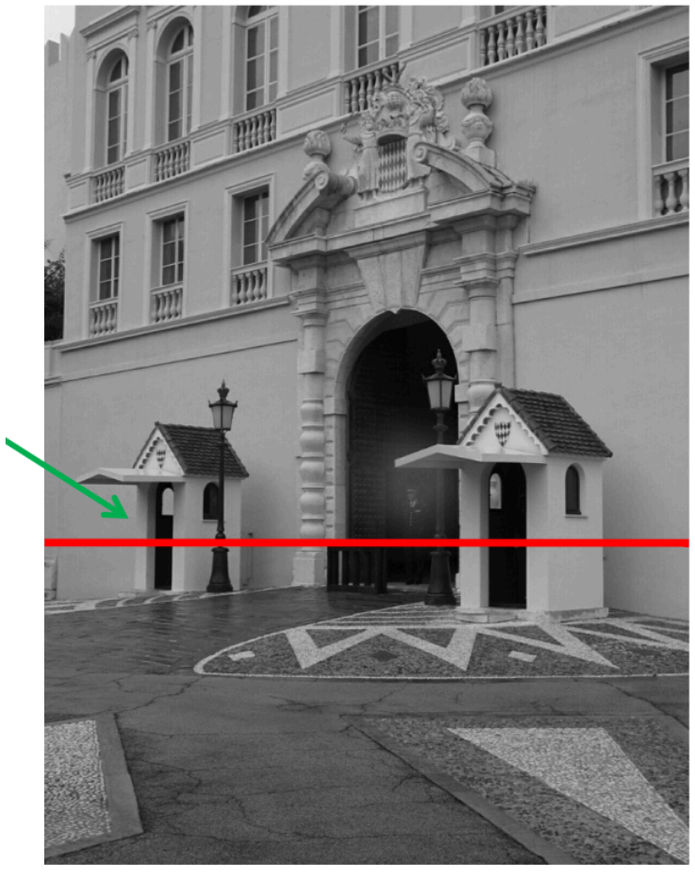
The gradient points in the direction of most rapid increase in intensity

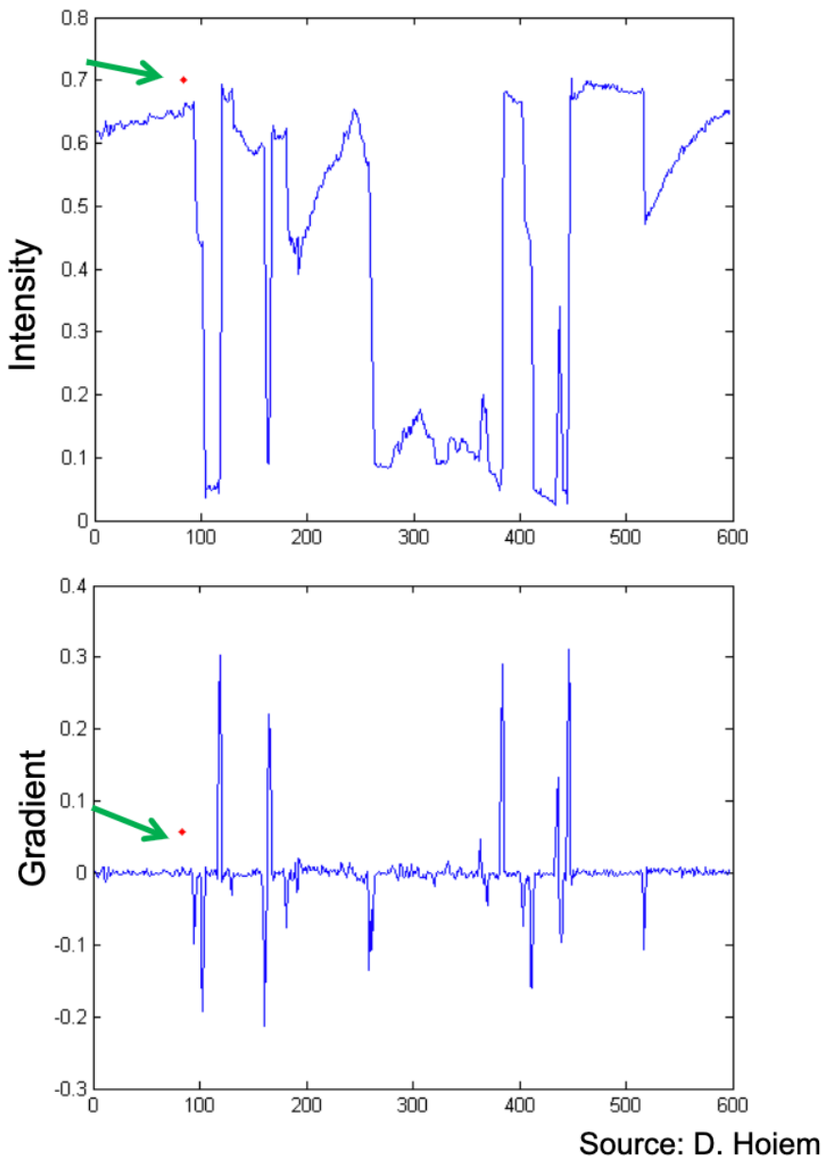
The *edge strength* is given by the gradient magnitude

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

Source: Steve Seitz

Intensity profile

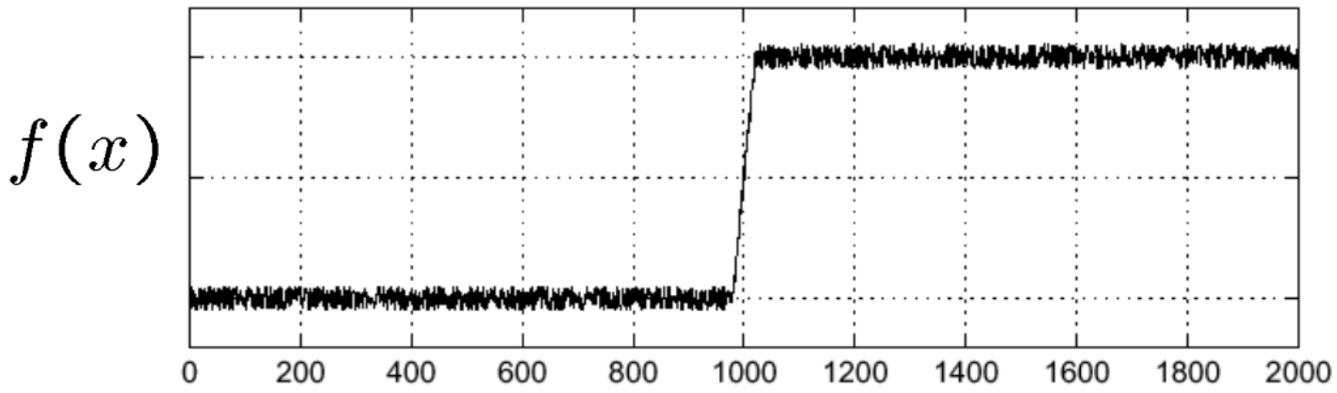


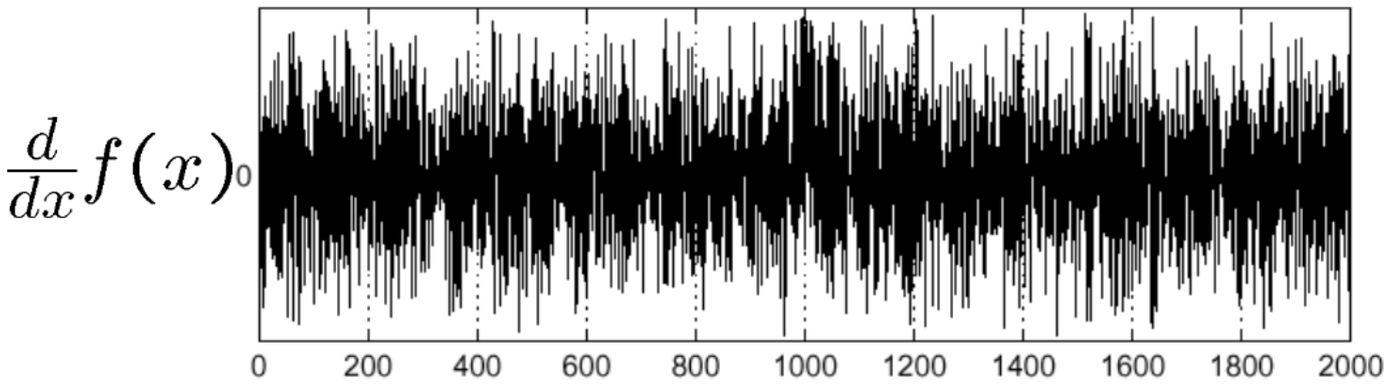


Source: D. Hoiem

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?

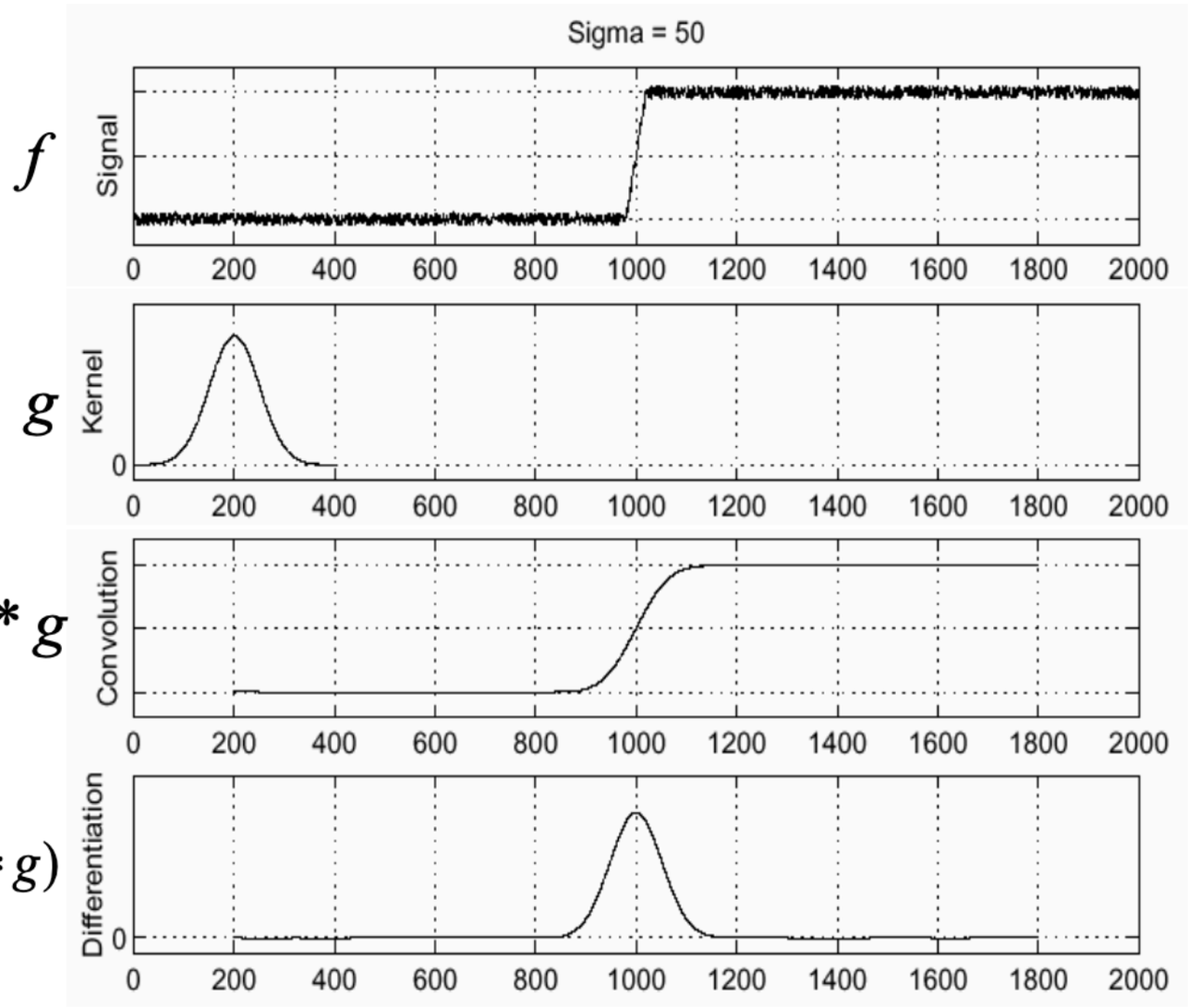
Source: S. Seitz

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

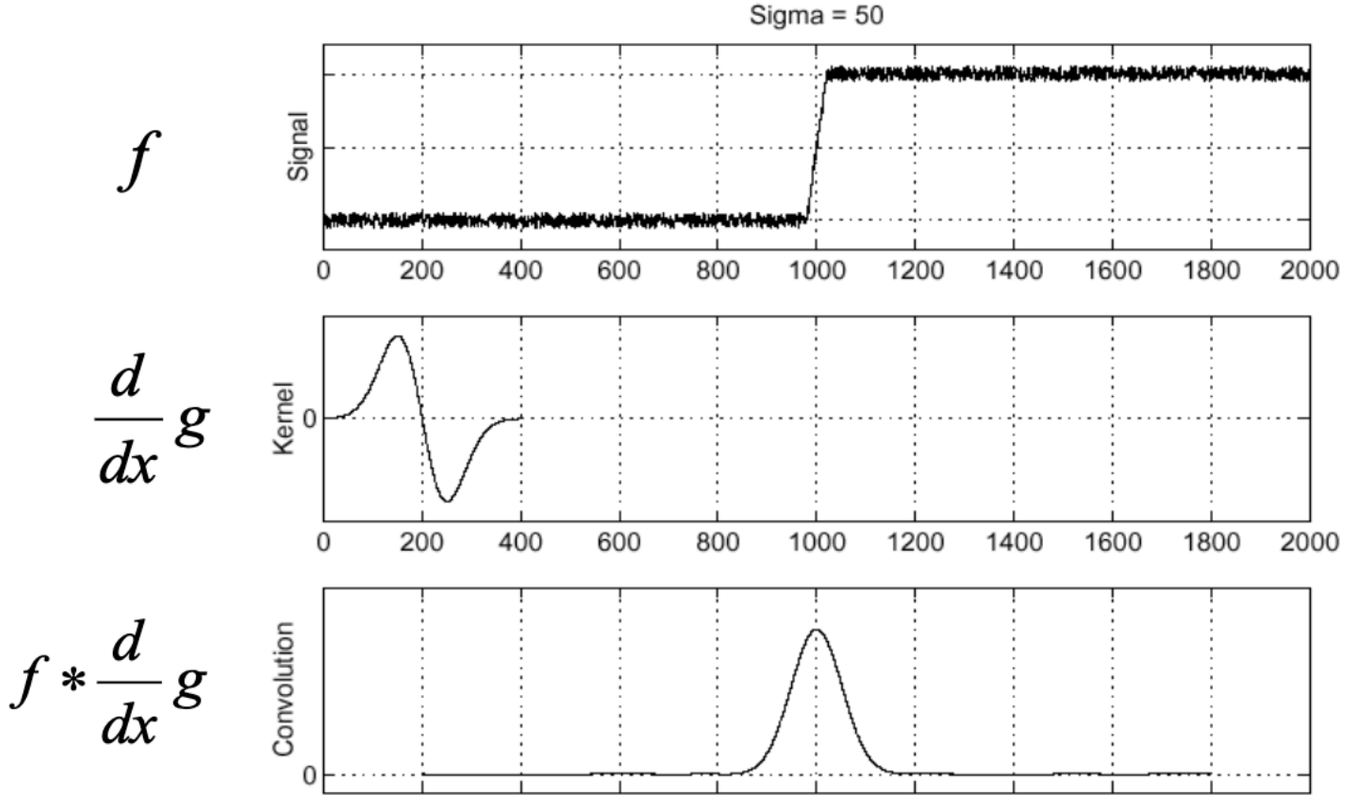


- To find edges, look for peaks in $\frac{d}{dx}(f * g)$

Source: S. Seitz

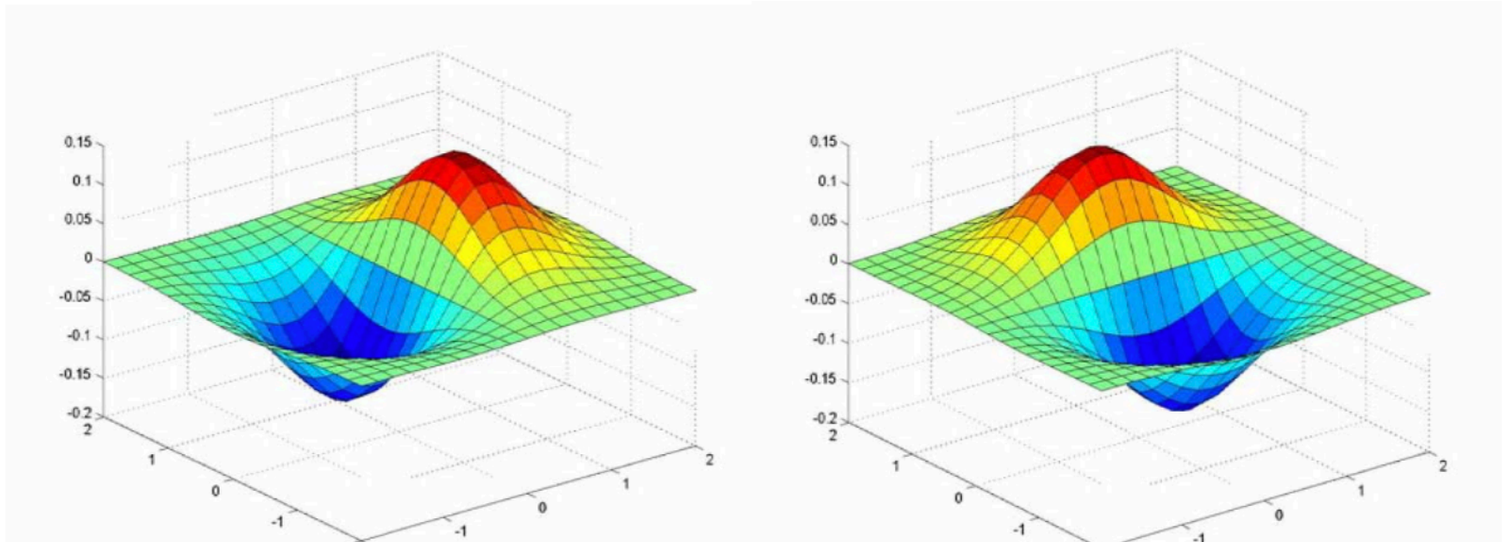
Edge detection via Convolution

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

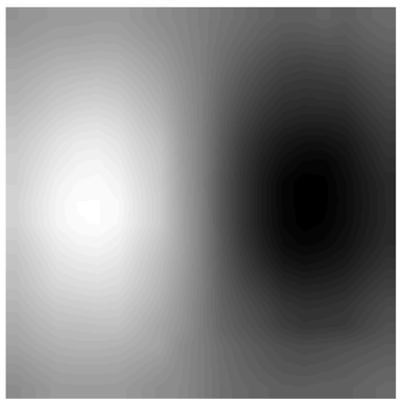


Source: S. Seitz

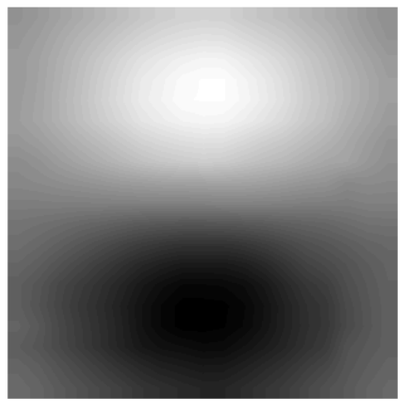
Derivative of Gaussian filter



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

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

Finite diff operator

Simple box filter

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

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

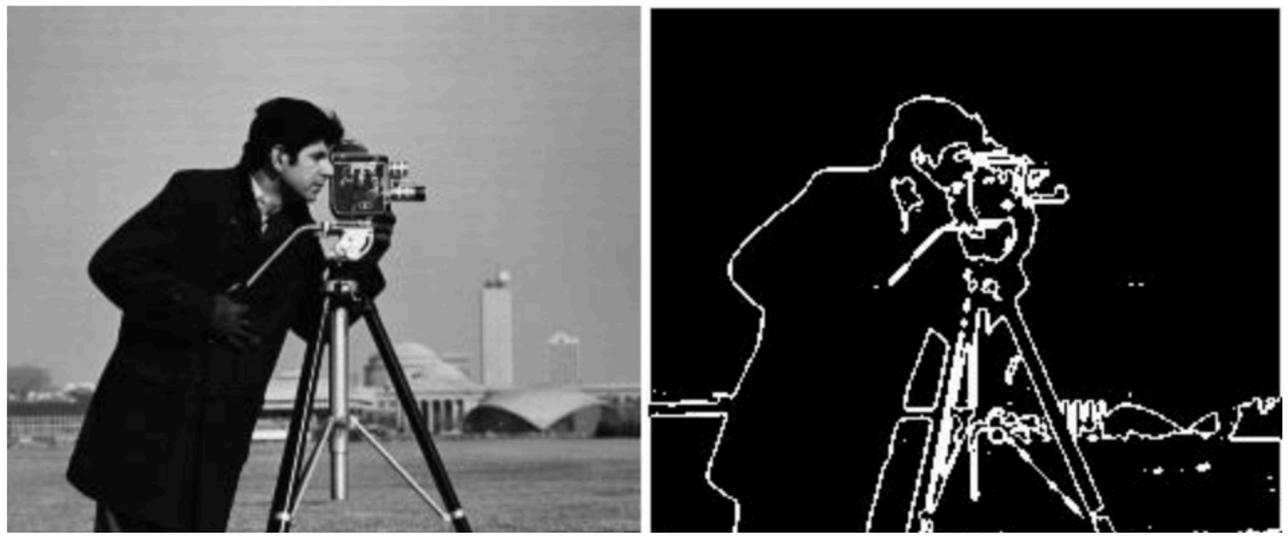
Finite diff operator

Simple Gaussian

Canny Edge detection

```python
#import the required libraries
import numpy as np
import cv2
import matplotlib.pyplot as plt
%matplotlib inline
#read the image
image = cv2.imread('coins.jpg')
#calculate the edges using Canny edge algorithm
edges = cv2.Canny(image,100,200)
#plot the edges
plt.imshow(edges)
```

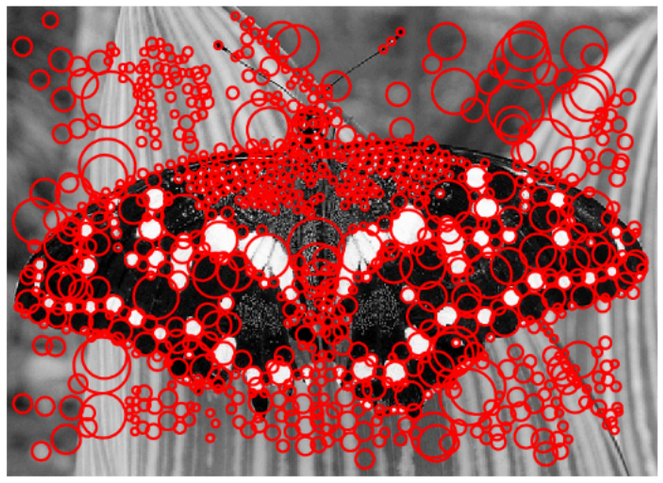
Edge Detection.py hosted with ❤ by GitHub view raw



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

Corner/blob detectors

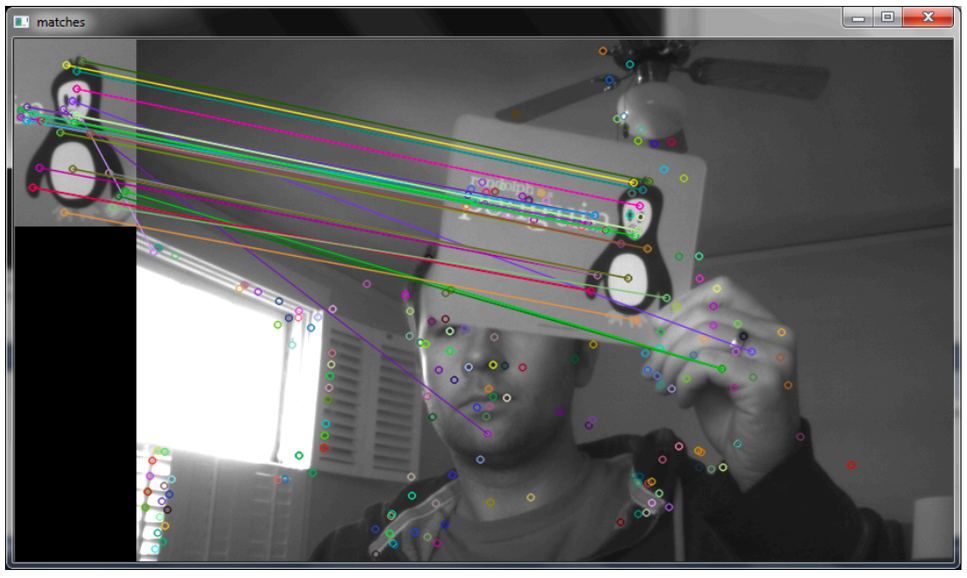
- Edges are useful as local features, but corners and small areas (blobs) are generally more helpful in computer vision tasks. Blob detectors can be built by extending the basic edge detector idea that we just discussed.



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.

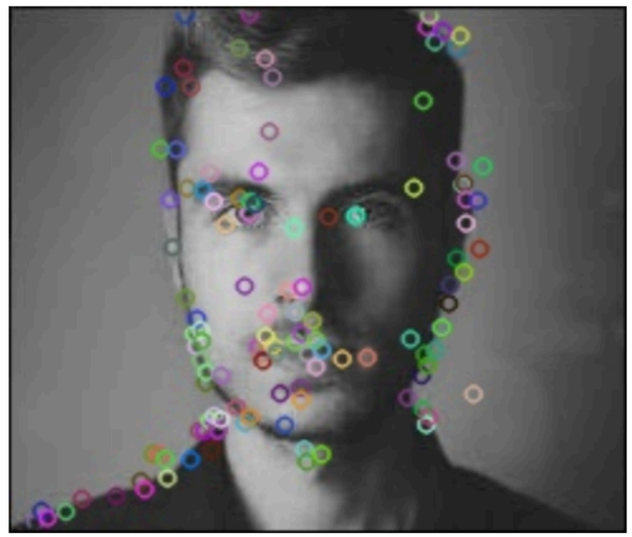
- They are locations that define what is interesting in the image. Keypoints are important, because no matter how the image is modified (rotation, shrinking, expanding, distortion), we will always find the same keypoints for the image.



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

SIFT

```python
#import required libraries
import cv2
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
#show OpenCV version
print(cv2.__version__)
#read the iamge and convert to grayscale
image = cv2.imread('index.png')
gray = cv2.cvtColor(image,cv2.COLOR_BGR2GRAY)
#create sift object
sift  = cv2.xfeatures2d.SIFT_create()
#calculate keypoints and their orientation
keypoints,descriptors = sift.detectAndCompute(gray,None)
#plot keypoints on the image
with_keypoints = cv2.drawKeypoints(gray,keypoints)
#plot the image
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).

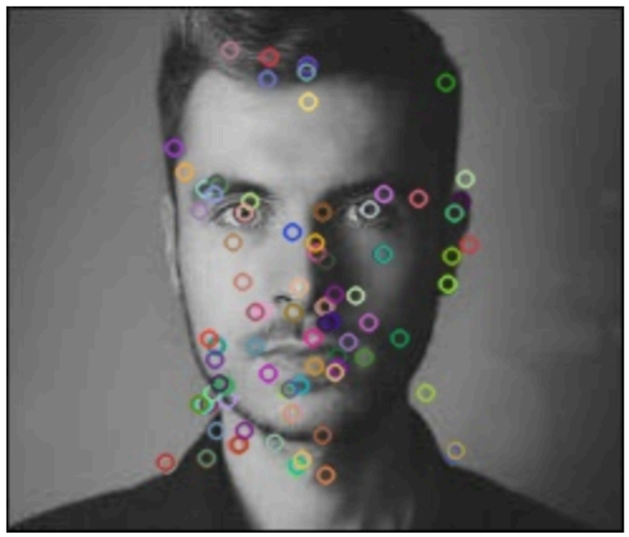
Bay, Herbert, Tinne Tuytelaars, and Luc Van Gool. "Surf: Speeded up robust features." *European conference on computer vision*. Springer, Berlin, Heidelberg, 2006.

SURF

```
#import required libraries
import cv2
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
#show OpenCV version
print(cv2.__version__)
#read image and convert to grayscale
image = cv2.imread('index.png')
gray = cv2.cvtColor(image,cv2.COLOR_BGR2GRAY)
#instantiate surf object
surf  = cv2.xfeatures2d.SURF_create(400)
#calculate keypoints and their orientation
keypoints,descriptors = surf.detectAndCompute(gray,None)

with_keypoints = cv2.drawKeypoints(gray,keypoints)

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?





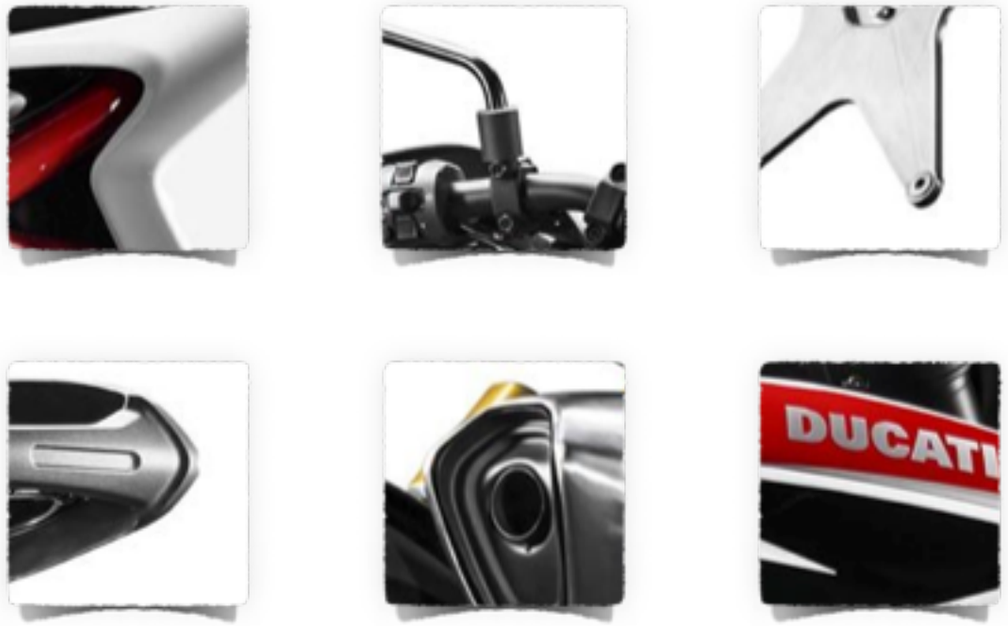









Some local feature are very informative



An object as

a collection of local features

(bag-of-features)

- deals well with occlusion
- scale invariant
- rotation invariant

Université de Montréal

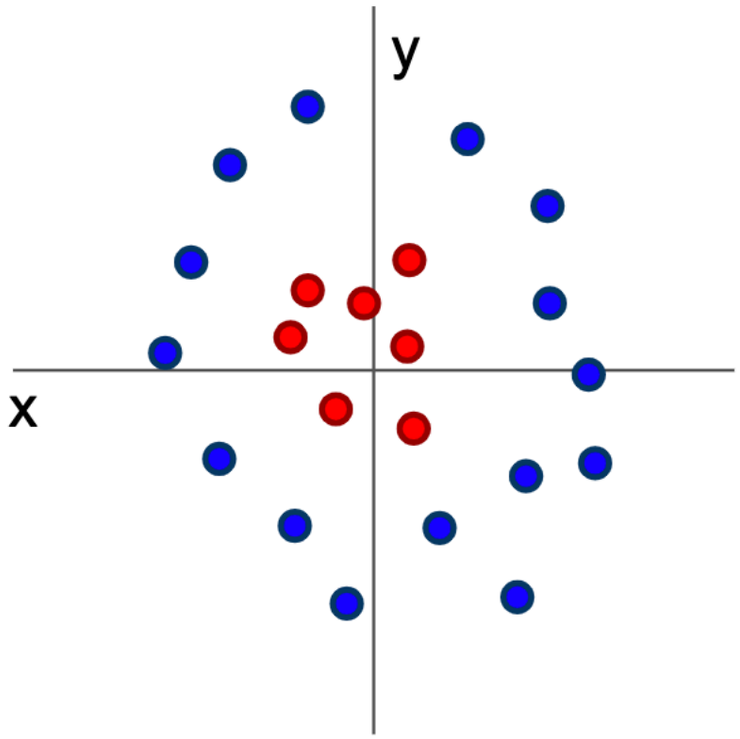
BOW

- Extract features (e.g., SIFT)

- Learn a visual dictionary

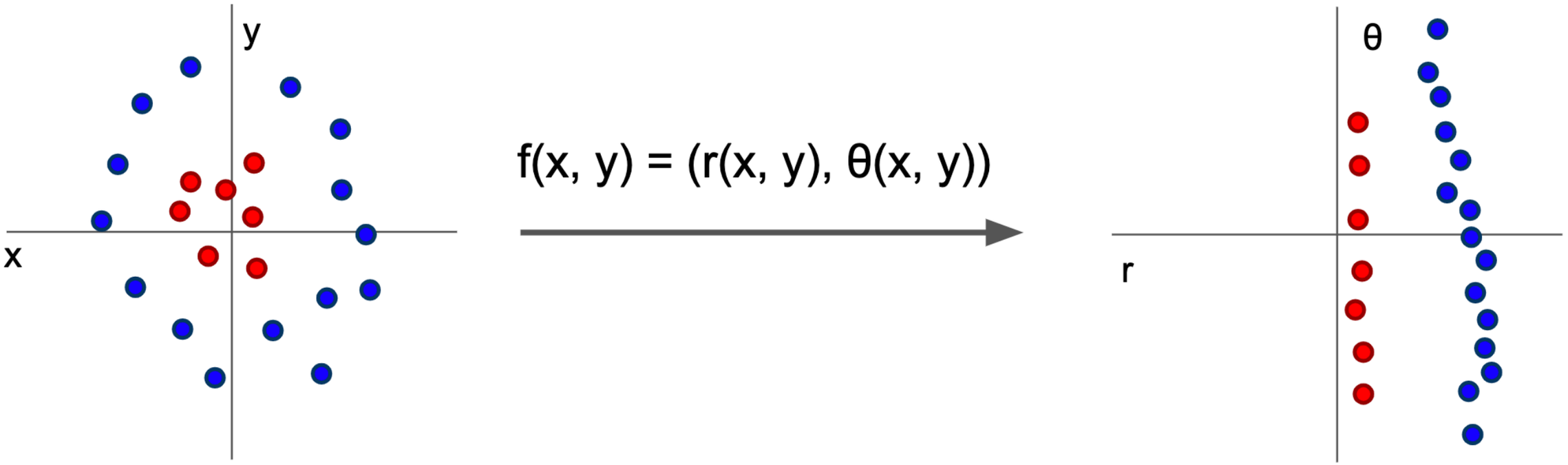
Image Features

Image Features: Motivation



Cannot separate red and blue points with linear classifier

Image Features: Motivation



Cannot separate red and blue points with linear classifier

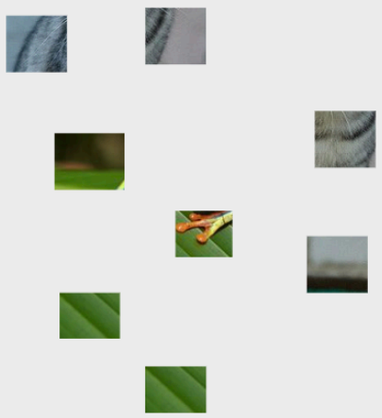
After applying feature transform, points can be separated by linear classifier

Image Features

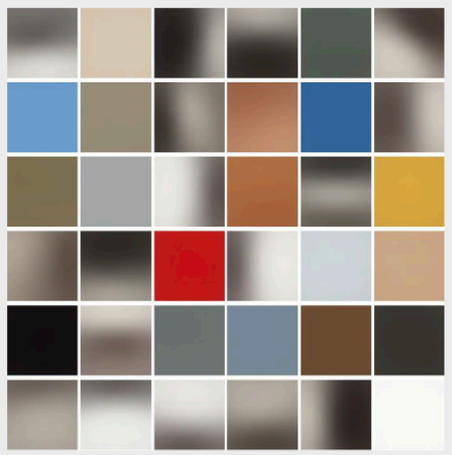
Example: Bag of Words

Step 1: Build codebook

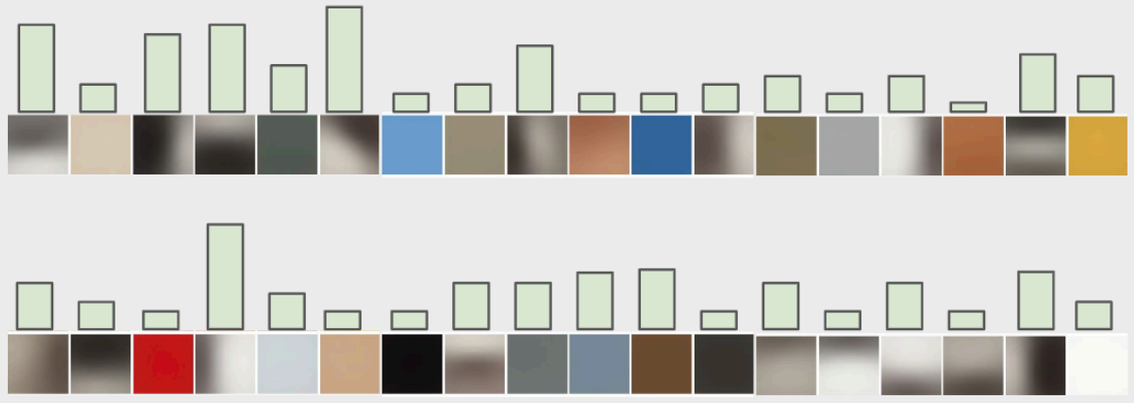
Extract random patches



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

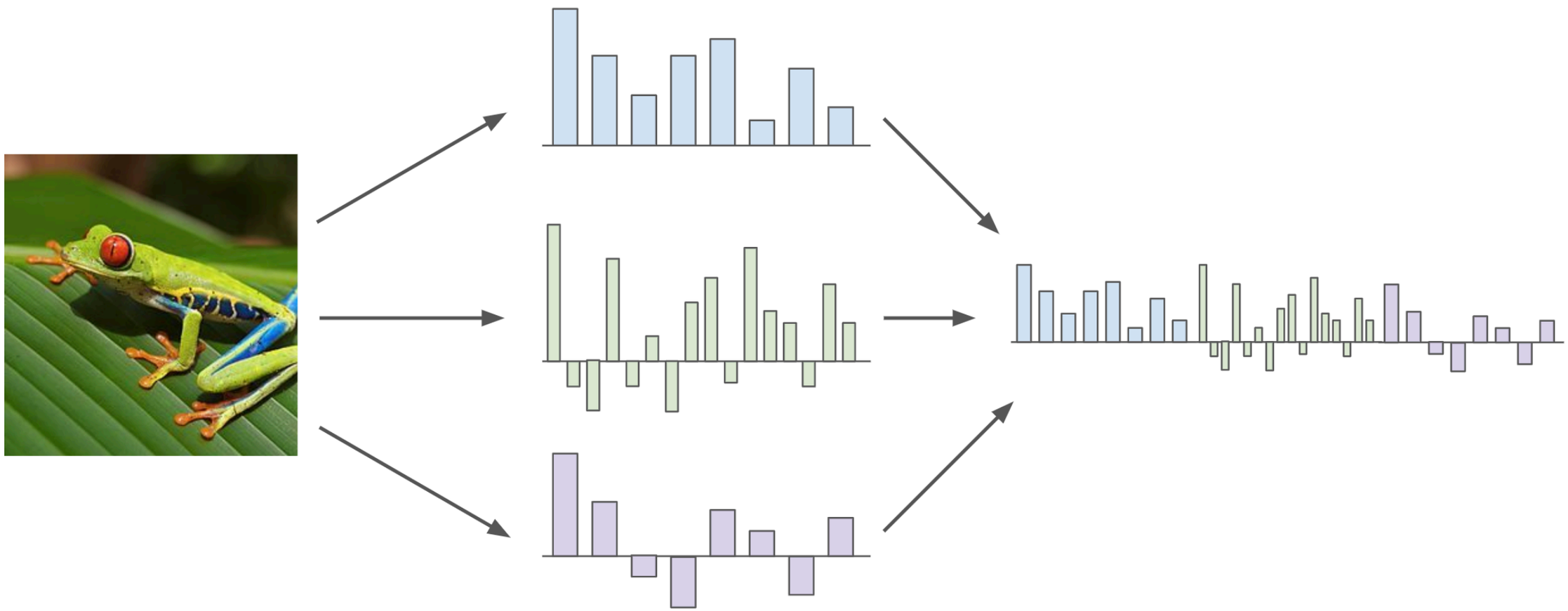


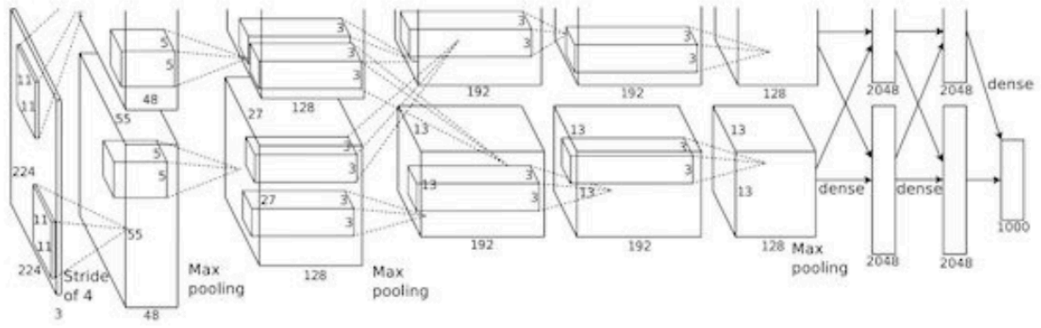
Image features vs ConvNets

Feature Extraction

f

10 numbers giving scores for classes

training



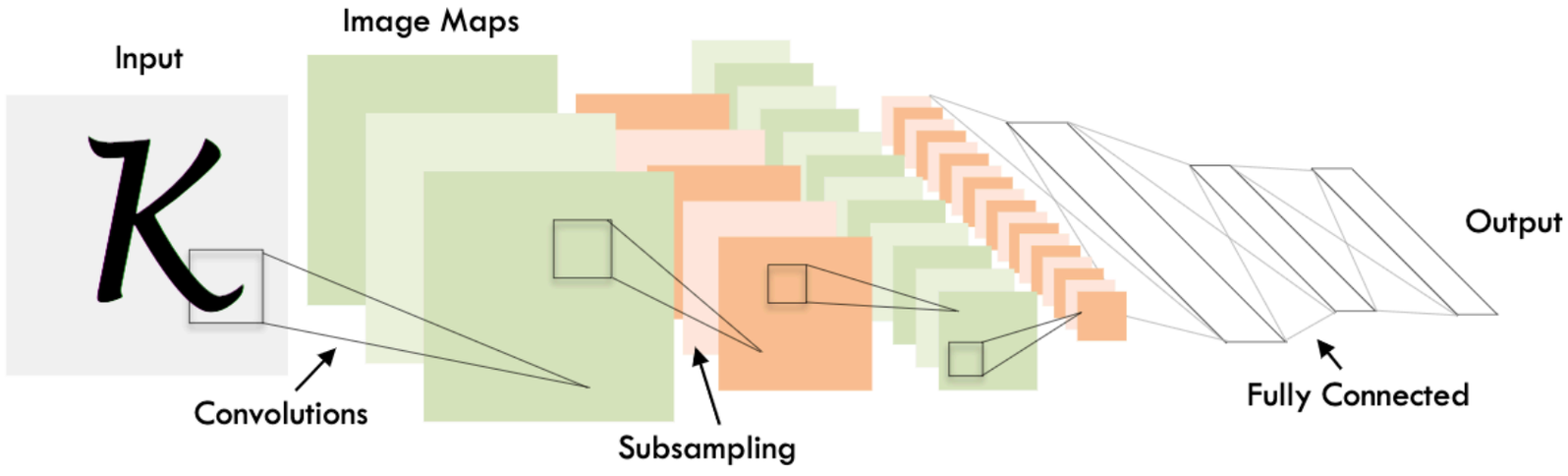
Krizhevsky, Sutskever, and Hinton, "Imagenet classification with deep convolutional neural networks", NIPS 2012. Figure copyright Krizhevsky, Sutskever, and Hinton, 2012. Reproduced with permission.

10 numbers giving scores for classes

training

1998

LeCun et al.



# of transistors

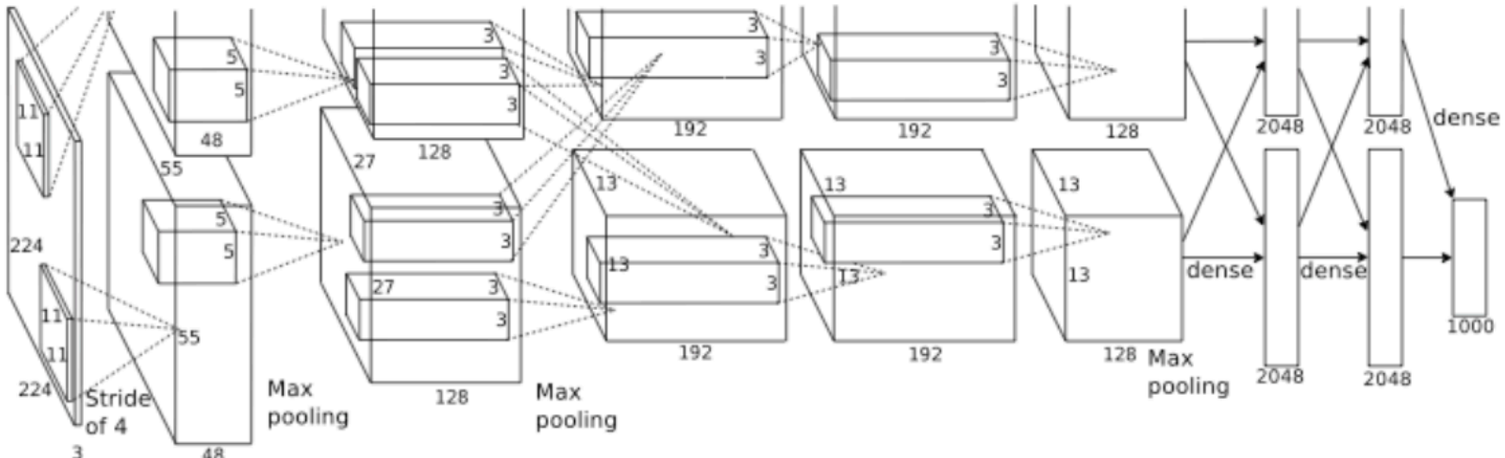
10^6

# of pixels used in training

10^7 NIST

2012

Krizhevsky et al.



# of transistors

10^9

GPUs

# of pixels used in training

10^{14} IMAGENET

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

Dense descriptor grid: HOG, LBP

Coding: local coordinate, super-vector

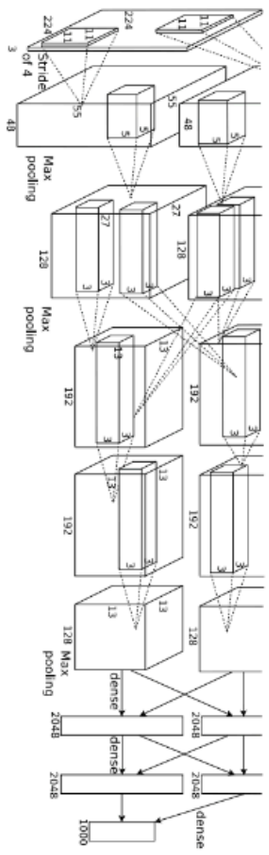
Pooling, SPM

Linear SVM

[Lin CVPR 2011]

Year 2012

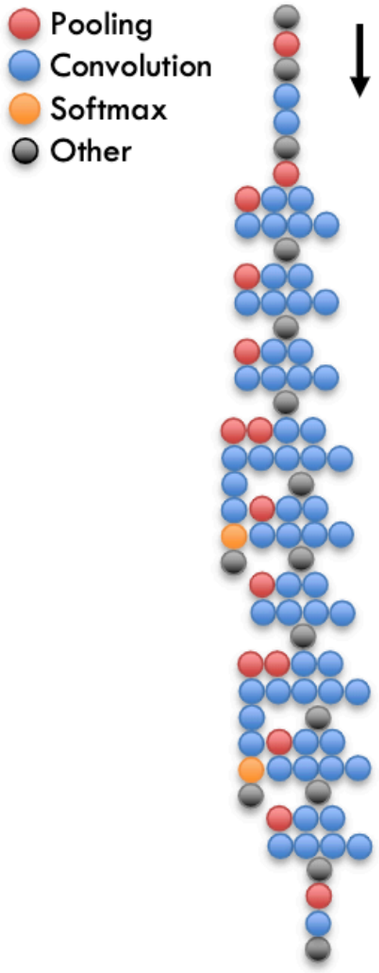
SuperVision



[Krizhevsky NIPS 2012]

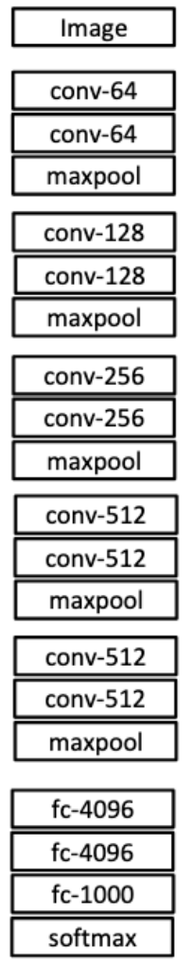
Year 2014

GoogLeNet



[Szegedy arxiv 2014]

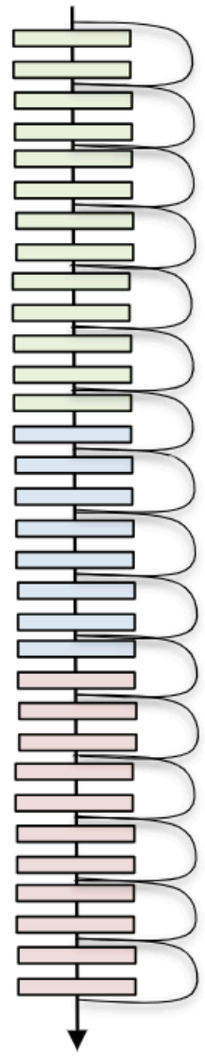
VGG



[Simonyan arxiv 2014]

Year 2015

MSRA



[He ICCV 2015]

AlexNet

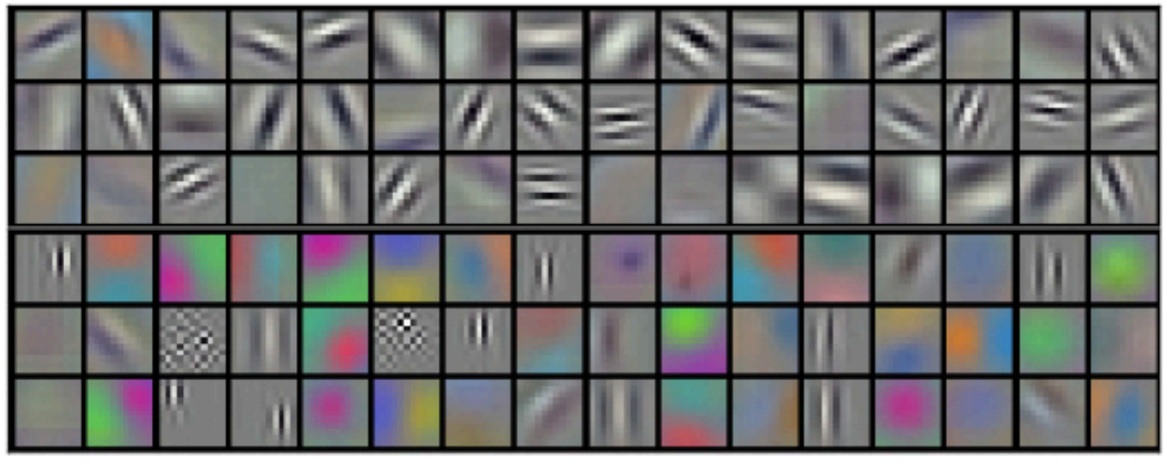
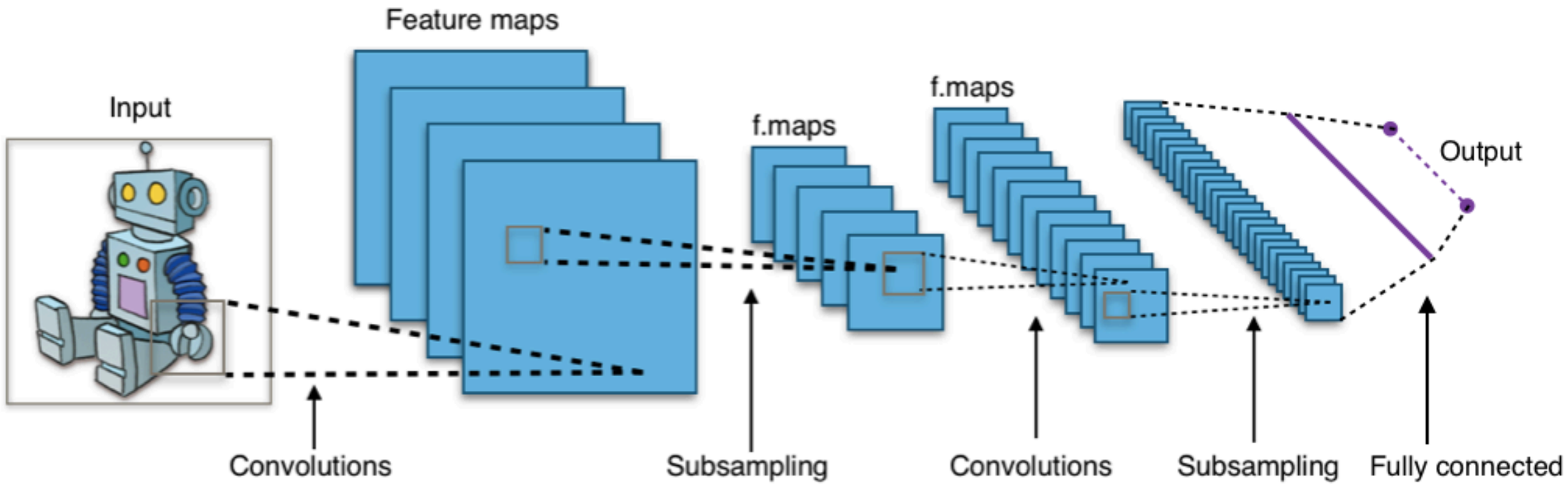


Figure 3: 96 convolutional kernels of size $11 \times 11 \times 3$ learned by the first convolutional layer on the $224 \times 224 \times 3$ input images. The

"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, …