Lec4 Detectors and Descriptors

·AIPKU·

人工智能引论实践课 计算机视觉小班

主讲人：刘家瑛

部分讲义来源与致谢

1. University of Washington CSE455: Computer Vision
2. Georgia Tech CS 4495: Computer Vision
3. Tinne Tuytelaars, Krystian Mikolajczyk. Local Invariant Feature Detectors: A Survey. Foundations and Trends in Computer Graphics and Vision 2007
4. Fei-Fei Li, Justin Johnson, Serena Yeung. Stanford University CS231n: Deep Learning for Computer Vision
5. Tinne Tuytelaars, Krystian Mikolajczyk. Local Invariant Feature Detectors: A Survey. 2008
6. Koen E. A. van de Sande, Theo Gevers, Cees G. M. Snoek. Evaluating Color Descriptors for Object and Scene Recognition. IEEE TPAMI 2010
7. Mohammad Khairul Islam, Farah Jahan, Joong-Hwan Baek. Object Cataloging Using Heterogeneous Local Features for Image Retrieval. KSII Transactions on Internet and Information Systems 2015
8. Toturial: SIFT detector and descriptor. https://www.vlfeat.org/overview/sift.html

Outline

- Properties of Ideal Local Feature
- Edge detector
- Blob detector
- SIFT descriptor
- HoG descriptor
- LBP descriptor
- Color descriptor

Outline

- Properties of Ideal Local Feature
- Edge detector
- Blob detector
- SIFT descriptor
- HoG descriptor
- LBP descriptor
- Color descriptor

How to Describe Objects?

Visual Descriptor

The visual components of color, form, line, shape, space, texture, and value.

LINE

A **line** is defined by a point moving in space. Lines may be two or three-dimensional, descriptive, implied, or abstract.

SHAPE

A **shape** is two-dimensional, flat, or limited to height and width.

VALUE

Value is the the lightness or darkness of tones or colors. White is the lightest value; black is the darkest. The value halfway between these extremes is called middle gray.

FORM

A **form** is a three-dimensional shape and encloses volume; includes height, width and depth. Cubes, spheres, pyramids, or cylinders are all examples of forms.

SPACE

Space refers to distances or areas around, between, or within components of a work of art to achieve a sense of depth.

COLOR

Colors are made up of three properties: hue, value, and intensity.

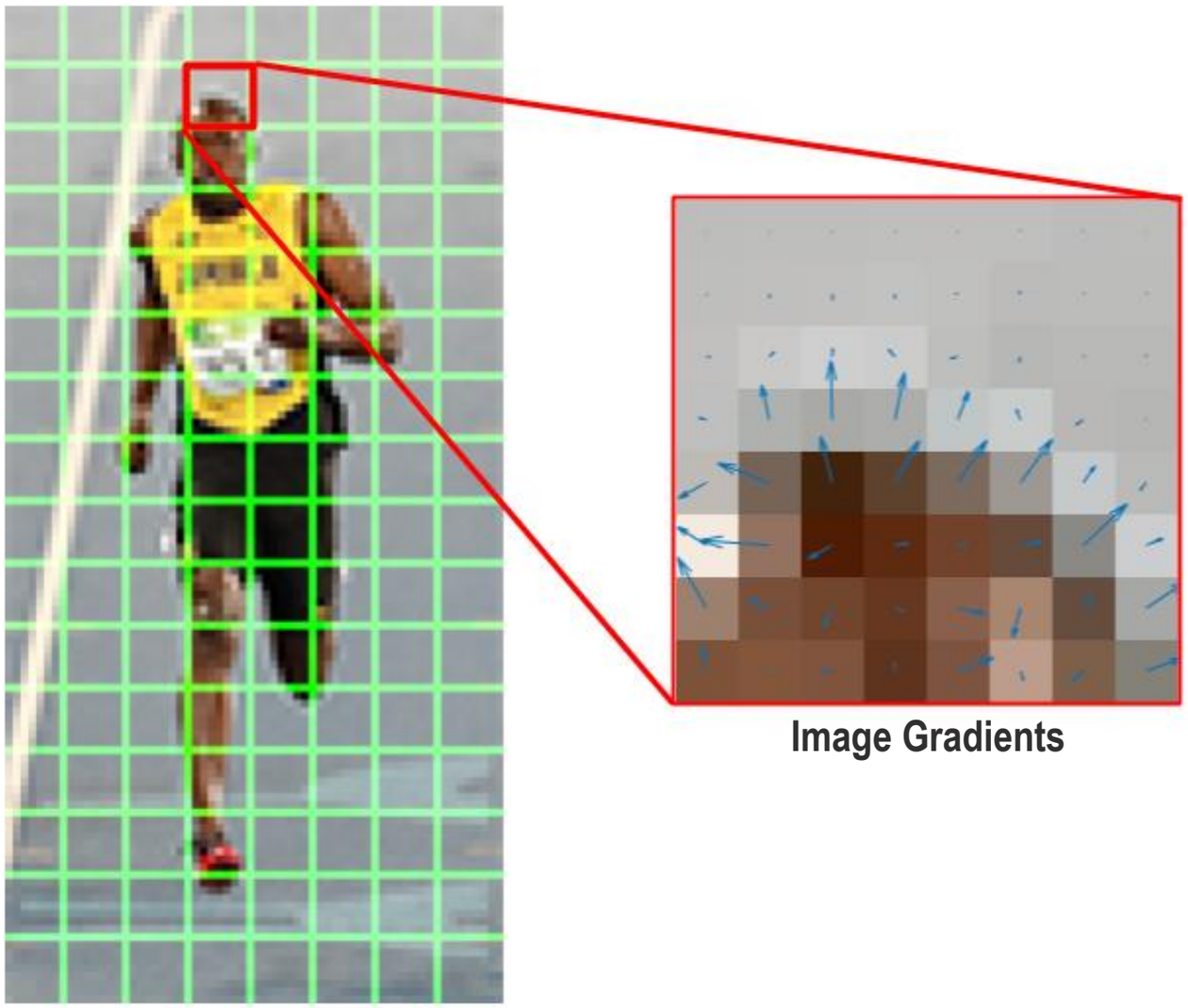
TEXTURE

Texture refers to the way things feel or look as if they might feel if touched — like feeling the surface of a football.

Source: https://www.khanacademy.org/partner-content/49ers-steam/49ers-art/49ers-elements-art-design/a/art-in-football

How to Represent Images?

Detector



Dense Sampling

Image Gradients

Descriptor

2	3	4	4	3	4	2	2
5	11	17	13	7	9	3	4
11	21	23	27	22	17	4	6
23	99	165	135	85	32	26	2
91	155	133	136	144	152	57	28
98	196	76	38	26	60	170	51
165	60	60	27	77	85	43	136
71	13	34	23	108	27	48	110

Gradient Magnitude

80	36	5	10	0	64	90	73
37	9	9	179	78	27	169	166
87	136	173	39	102	163	152	176
76	13	1	168	159	22	125	143
120	70	14	150	145	144	145	143
58	86	119	98	100	101	133	113
30	65	157	75	78	165	145	124
11	170	91	4	110	17	133	110

Gradient Direction

Applications

- Matching
- Estimation
- Indexing
- Detection
- Tracking

Source: https://www.learnopencv.com/histogram-of-oriented-gradients/

Example: Image Stitching

Image 1

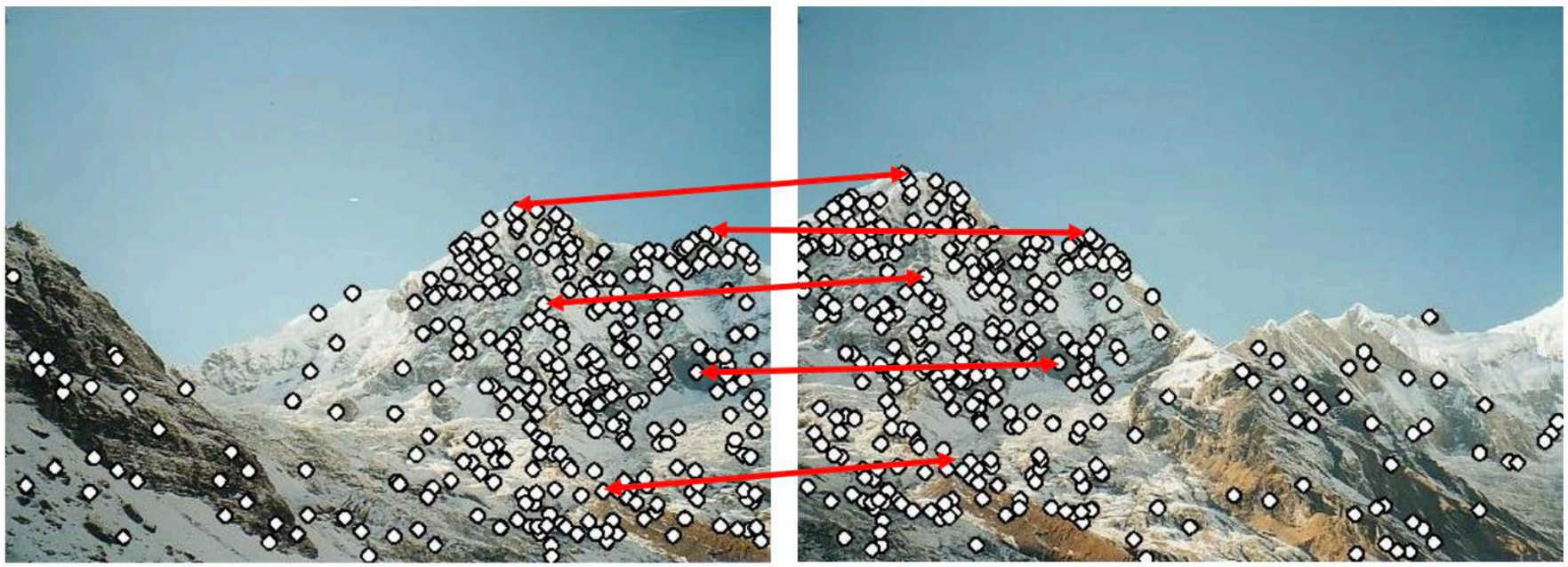
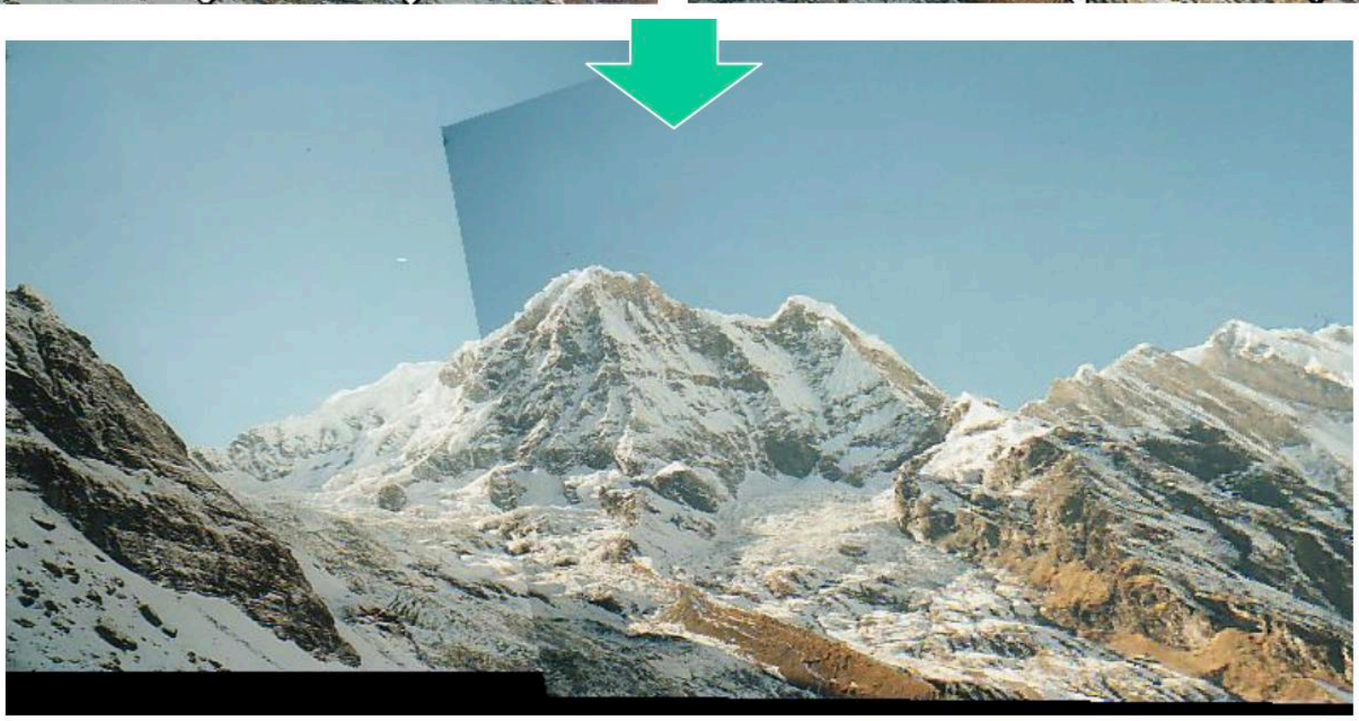


Image 2

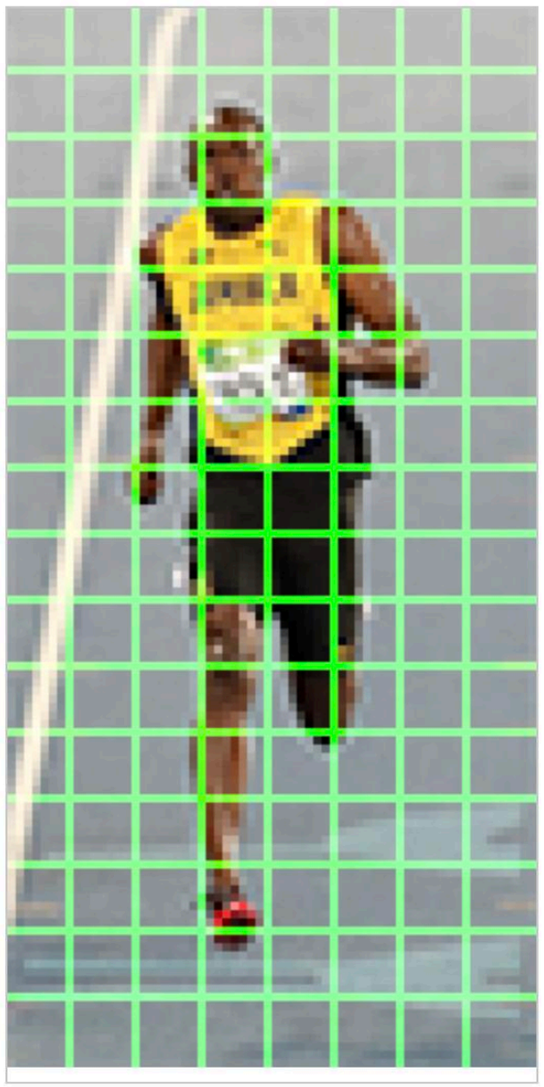


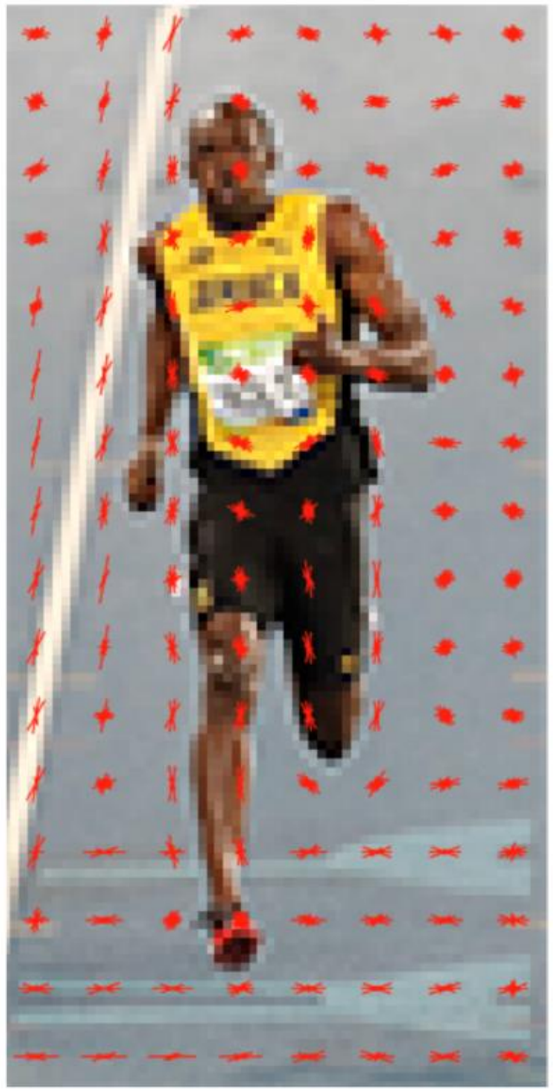
Source: https://courses.cs.washington.edu/courses/cse455/09wi/Lects/lect6.pdf

Feature Descriptor

Definition

- A representation of an image / an image patch that describes the image by **extracting useful information**





Global Feature Vs. Local Feature

Global Features

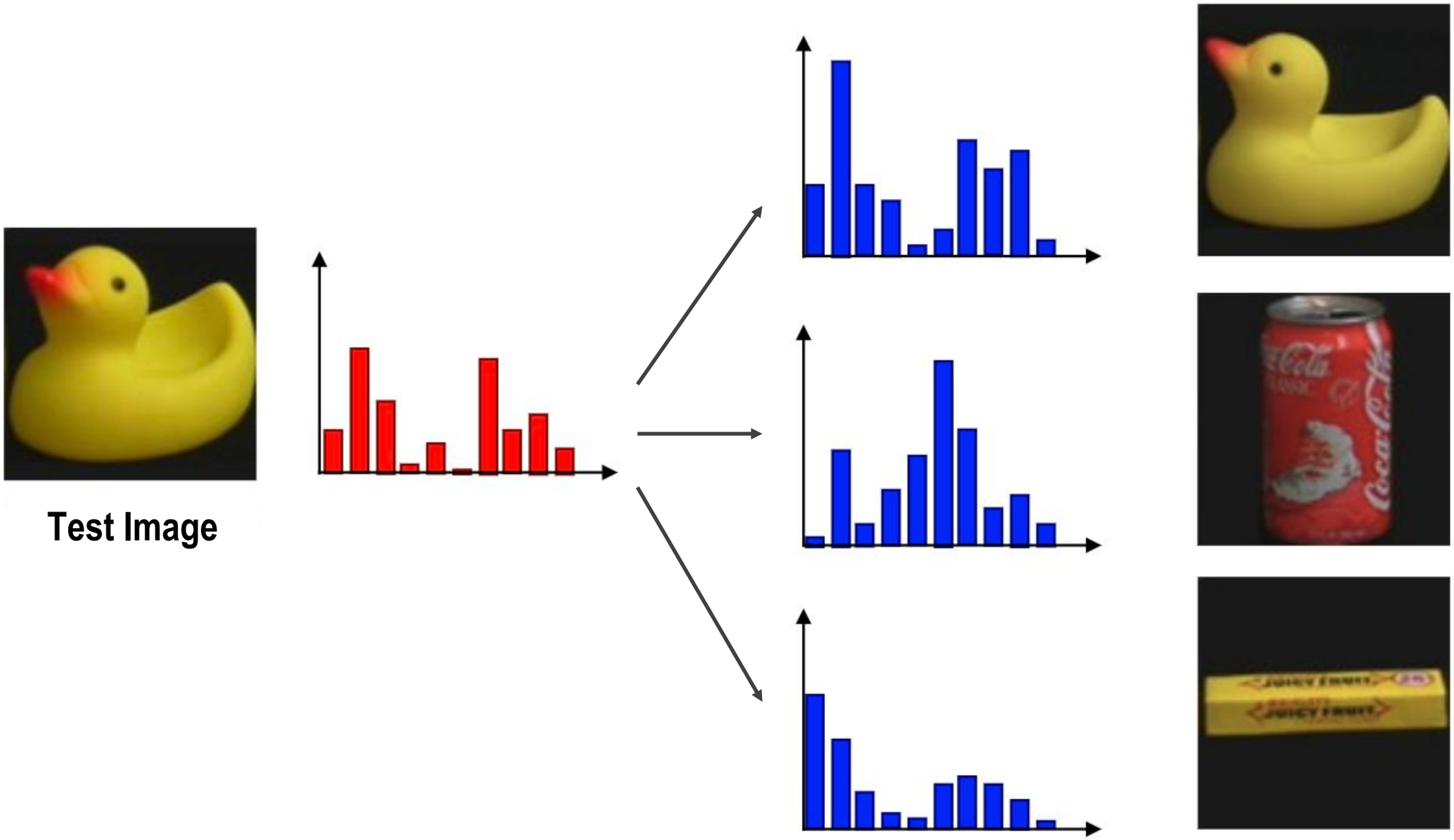
- Describe an image as a whole
- Ability to generalize an entire object with a single vector

Local Features

- Represent image patches
- Computed at multiple points in the image
- Require to handle cases ← →

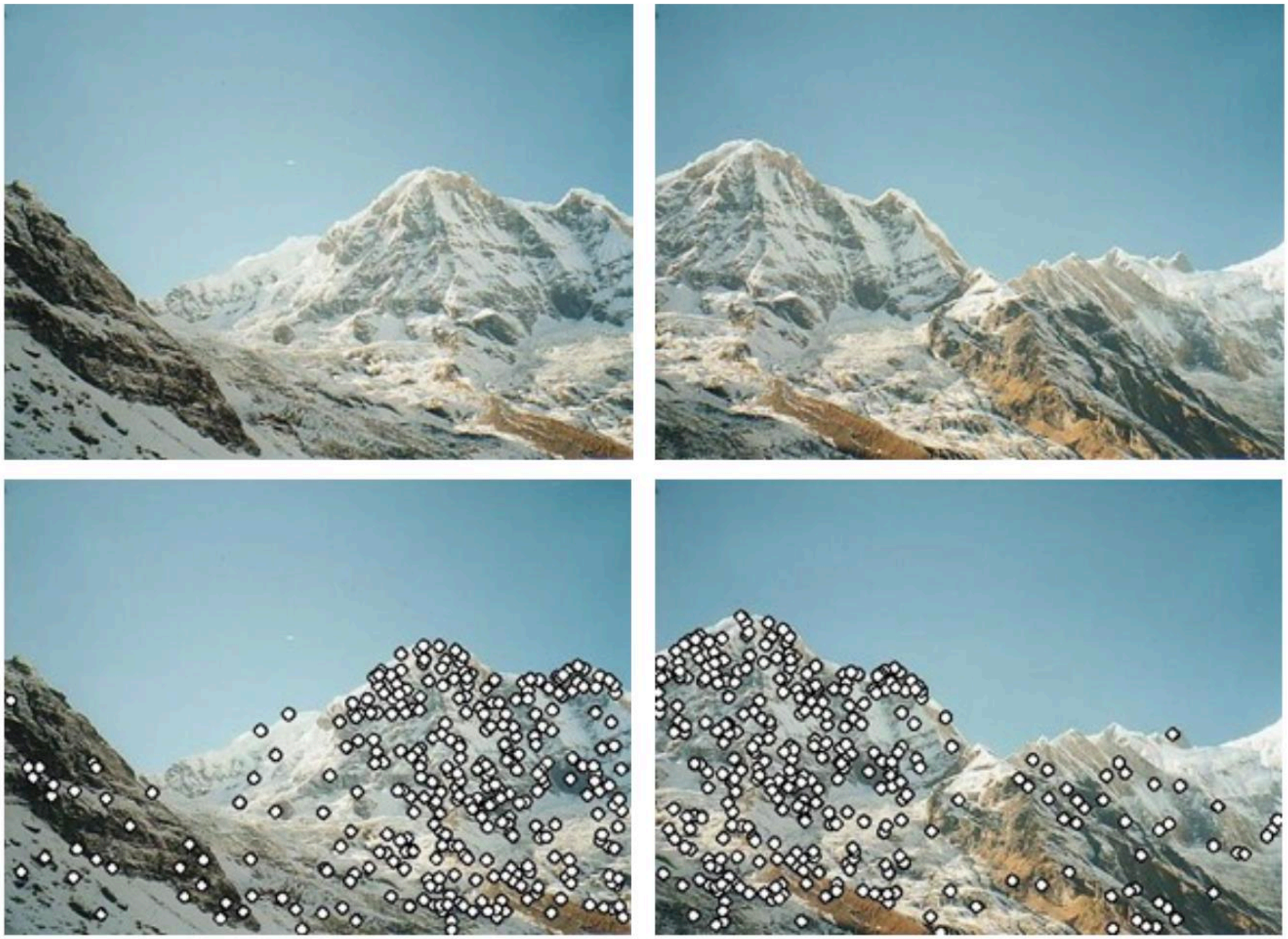
  a variable number of feature vectors per image

Recognition with Global Features



Test Image

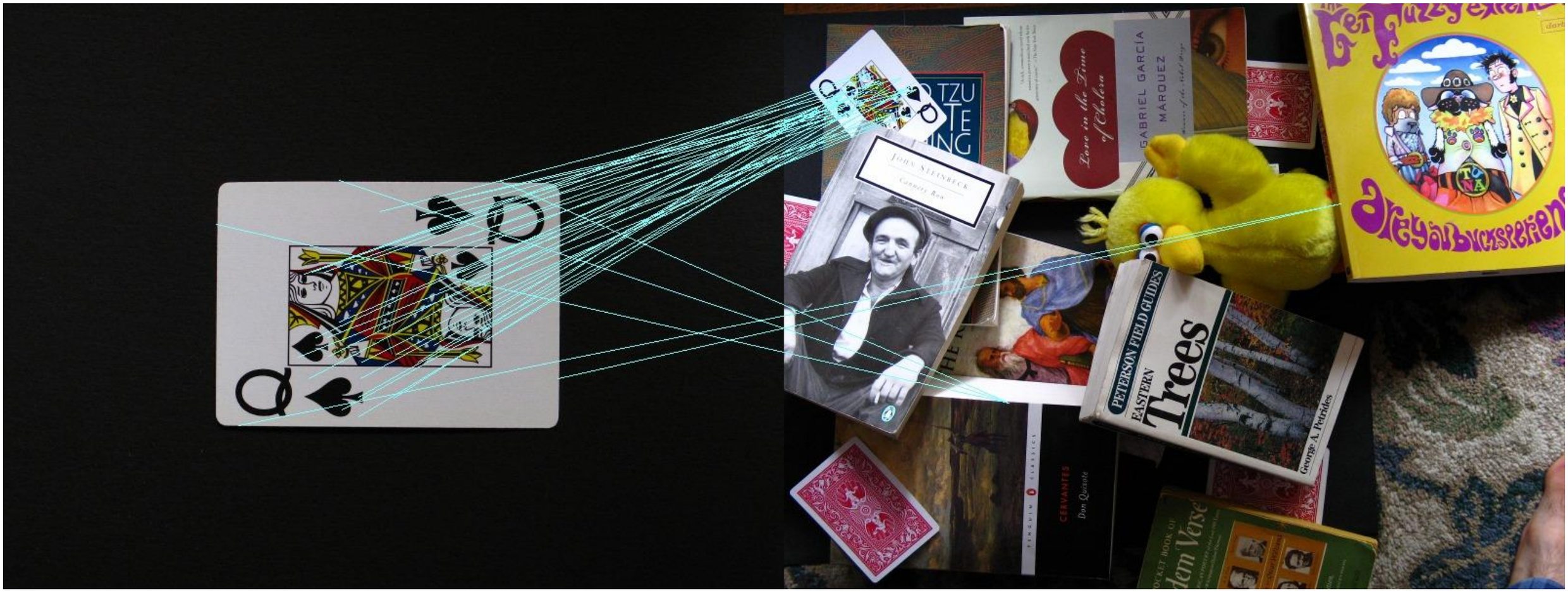
Recognition with Local Features



Recognition with Local Features

In comparison to global features,

local features are more robust to occlusion and clutter



Properties of Ideal Local Feature

(1) Repeatability

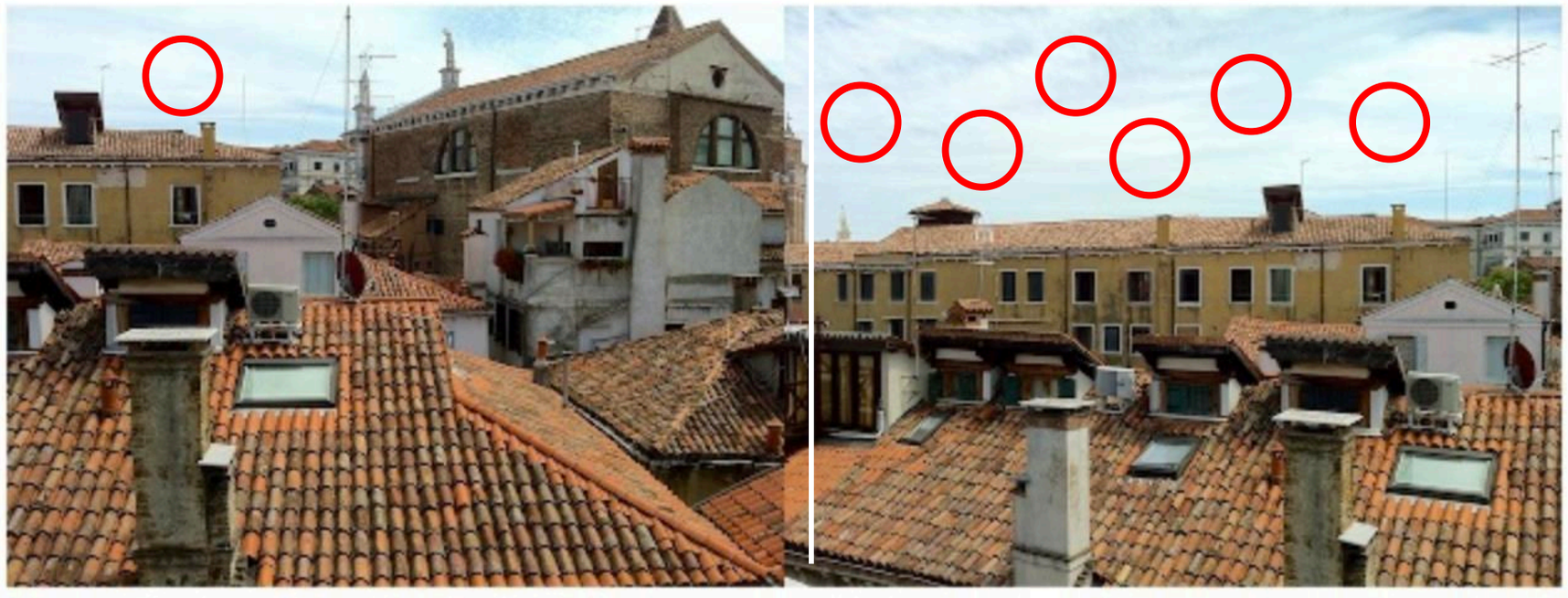
- Given two images of the same object / scene, taken under different viewing conditions
- High percentage of features detected on the scene part visible in both images should be found in both images

Source: http://homes.esat.kuleuven.be/~tuytelaa/FT_survey_interestpoints08.pdf

Properties of Ideal Local Feature

(2) Distinctiveness / Informativeness

- The intensity patterns underlying the detected features should show a lot of variation
- Such that features can be distinguished and matched



Properties of Ideal Local Feature

(3) Locality

- Features should be local, to reduce the probability of occlusion
- allow simple model approximations of geometric & photometric deformations
- between two images taken under different viewing conditions

Properties of Ideal Local Feature

(4) Quantity

- The number of detected features should be sufficiently large
- such that a reasonable number of features are detected even on small objects

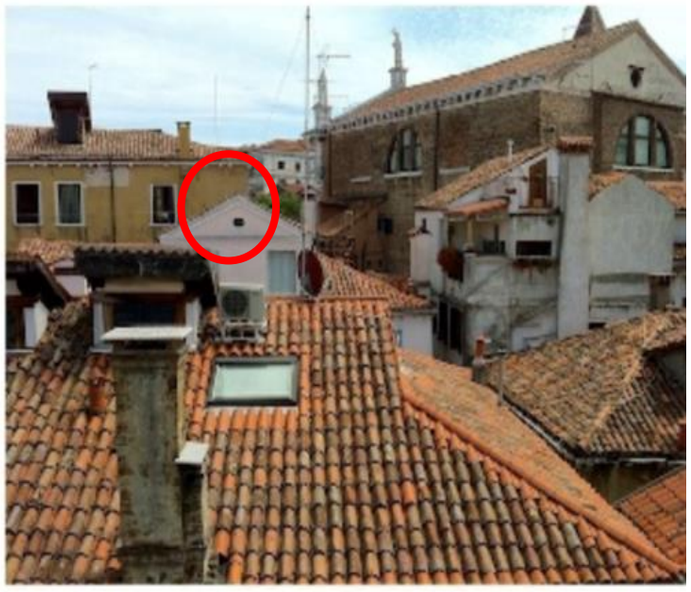
Feature A

Feature B

Properties of Ideal Local Feature

(5) Accuracy

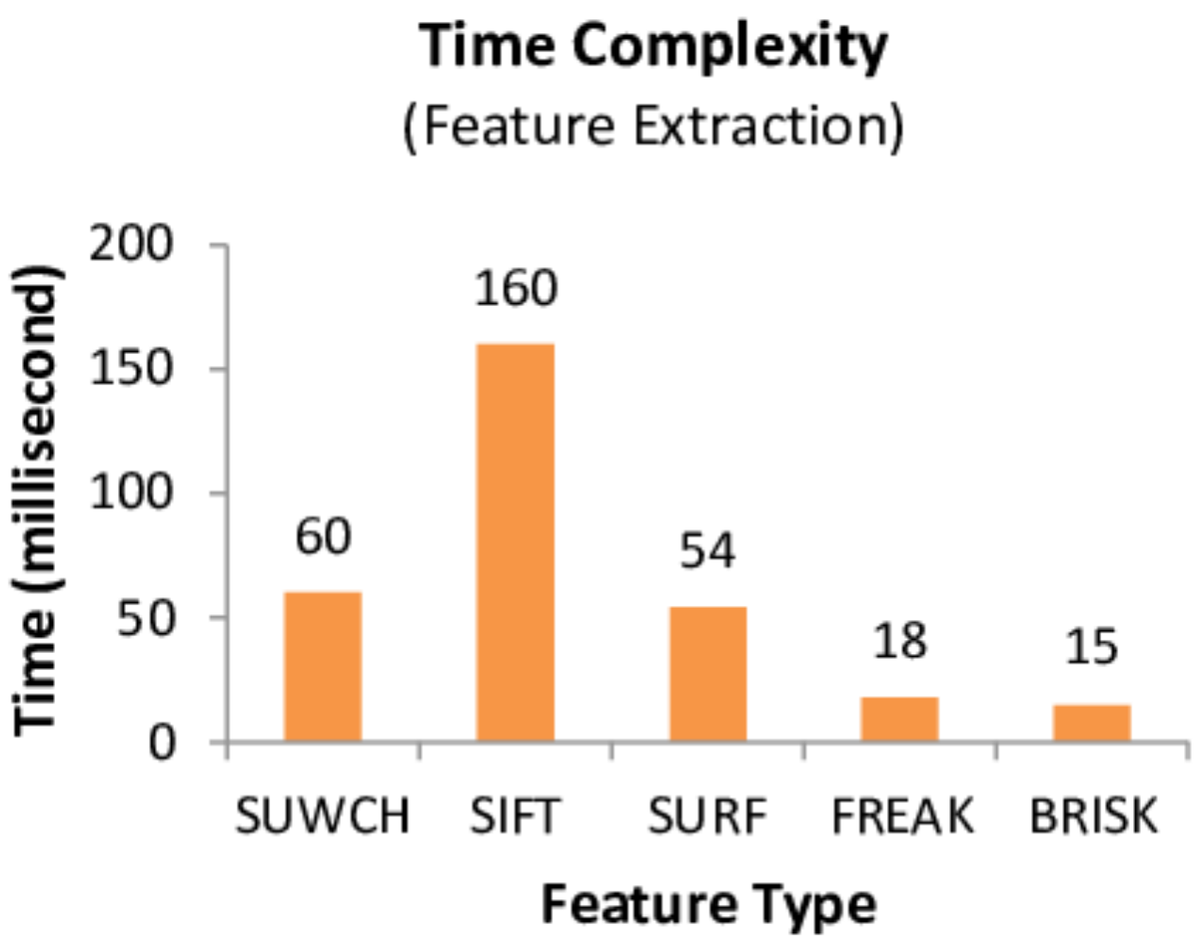
- The detected features should be accurately localized, both in image location
- as with respect to scale and possibly shape



Properties of Ideal Local Feature

(6) Efficiency

- Preferably, the detection of features in a new image should allow for time-critical applications



Elapsed time required for extraction of various types of features.

M.K. Islam et al., “Object cataloging using heterogeneous local features for image retrieval,” 2015.

Edge Detection: Example

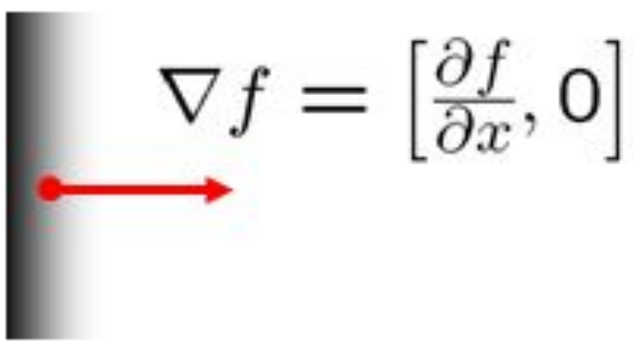
- **Edge:** Significant local changes in an image

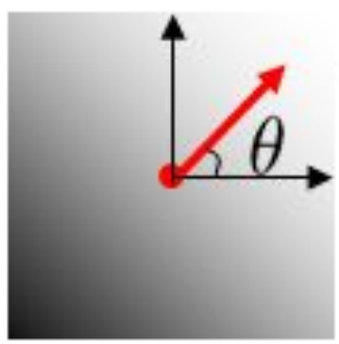
Image Gradients

- 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 change in intensity**





Source: https://slideplayer.com/slide/8048909/

Image Gradients

- The **gradient direction** is given by:

$$\theta = \tan^{-1}\left(\frac{\partial f}{\partial y} / \frac{\partial f}{\partial x}\right)$$

- 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: https://slideplayer.com/slide/8048909/

First-Order Methods

- The simplest approach is to use **central differences**:

$$L_x(x,y) = -\frac{1}{2}L(x-1,y) + 0 \cdot L(x,y) + \frac{1}{2} \cdot L(x+1,y)$$

$$L_y(x,y) = -\frac{1}{2}L(x,y-1) + 0 \cdot L(x,y) + \frac{1}{2} \cdot L(x,y+1)$$

corresponding to the application of the following **filter masks** to the image data

$$L_x = \begin{bmatrix} +1/2 & 0 & -1/2 \end{bmatrix} L \quad \text{and} \quad L_y = \begin{bmatrix} +1/2 \\ 0 \\ -1/2 \end{bmatrix} L.$$

The well-know **Sobel Operator** is based on the following filters:

$$L_x = \begin{bmatrix} +1 & 0 & -1 \\ +2 & 0 & -2 \\ +1 & 0 & -1 \end{bmatrix} L \quad \text{and} \quad L_y = \begin{bmatrix} +1 & +2 & +1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix} L$$

Designing an Edge Detector

Two Factors:

- Use derivatives (in x and y direction) to define **a location with high gradient**
- Need **smoothing** to reduce noise prior to take derivative

Source: http://web.stanford.edu/class/cs231a/lectures/lecture9_detector_descriptors.pdf

Edge Detector in 2D

- **Smoothing**

$$I' = g(x,y) * I \qquad \text{where} \quad g(x,y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}}$$

- **Derivative**

$$\mathrm{S} = \nabla(g * I) = \nabla(g) * I = \begin{bmatrix} g_x \\ g_y \end{bmatrix} * I = \begin{bmatrix} g_x * I \\ g_y * I \end{bmatrix} = [S_x \; S_y]$$

gradient vector

$$\text{where} \quad \nabla g = \begin{bmatrix} \frac{\partial g}{\partial x} \\ \frac{\partial g}{\partial y} \end{bmatrix} = \begin{bmatrix} g_x \\ g_y \end{bmatrix}$$

Canny Edge Detection

original

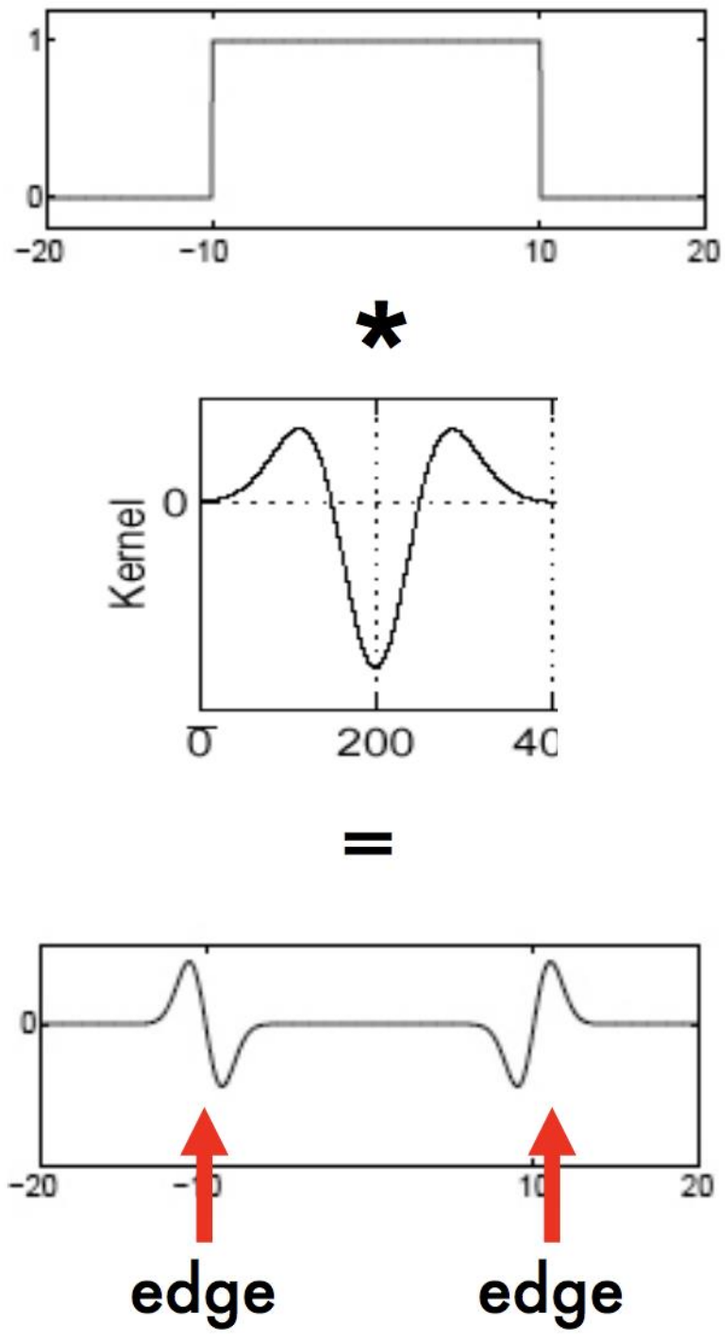
Canny with $\sigma = 1$

Canny with $\sigma = 2$

- The choice of σ depends on desired behavior
  - large σ detects large scale edges
  - small σ detects fine features

Source: http://web.stanford.edu/class/cs231a/lectures/lecture9_detector_descriptors.pdf

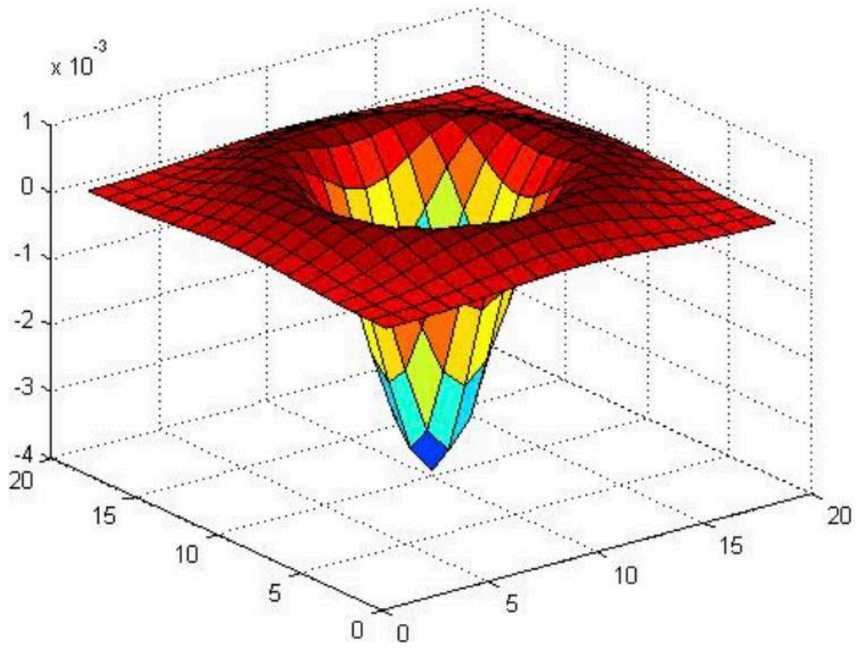
Edge Detection as Zero Crossing

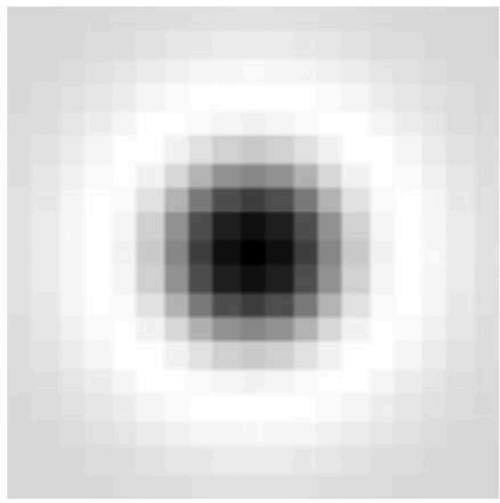


Blob Detection in 2D

- **Laplacian of Gaussian**

Circularly symmetric operator for blob detection in 2D

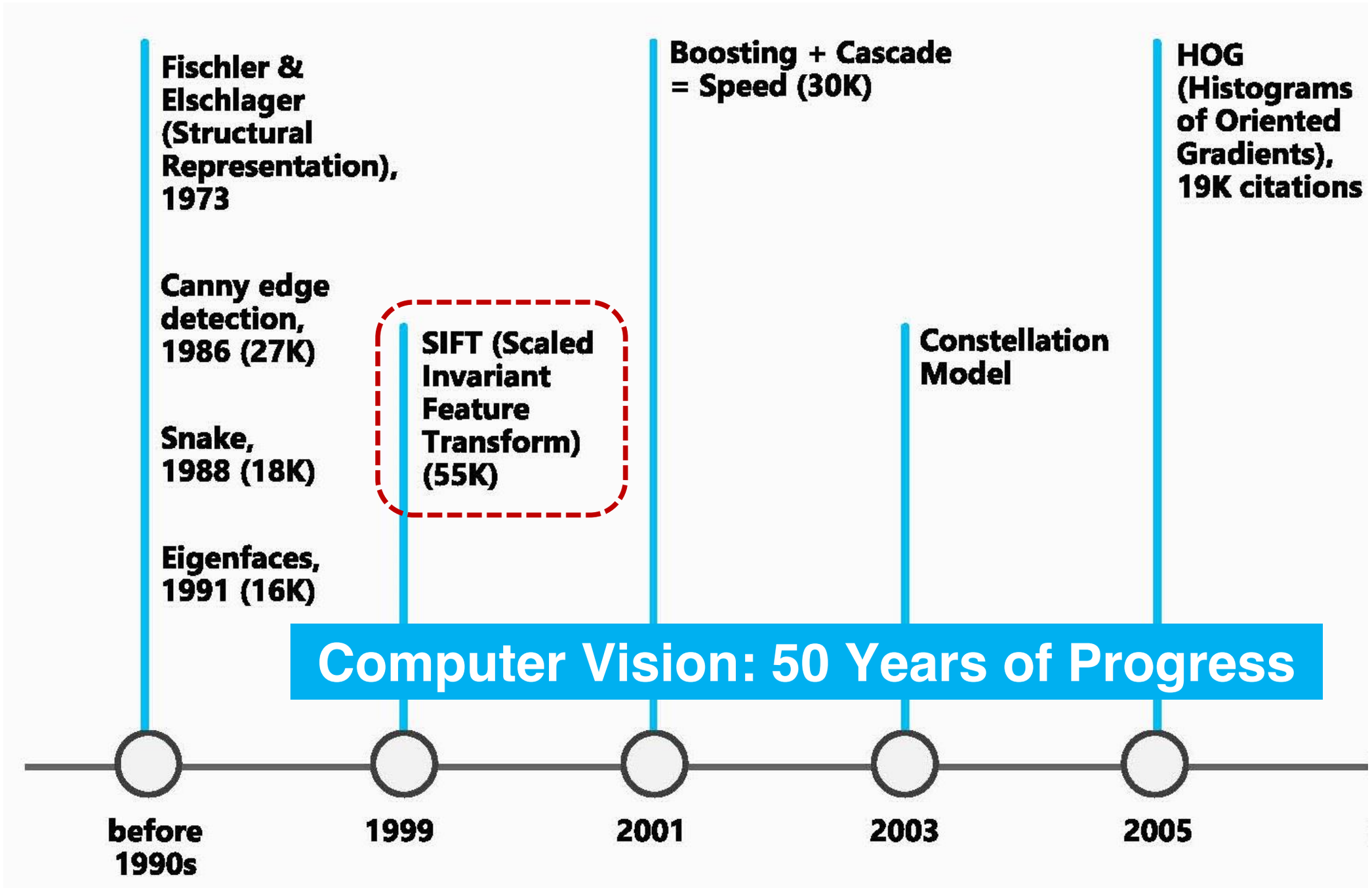




Scale-normalized: $\nabla^2_{\text{norm}} g = \sigma^2 \left(\frac{\partial^2 g}{\partial x^2} + \frac{\partial^2 g}{\partial y^2} \right) = \sigma^2 (g_{xx} + g_{yy})$

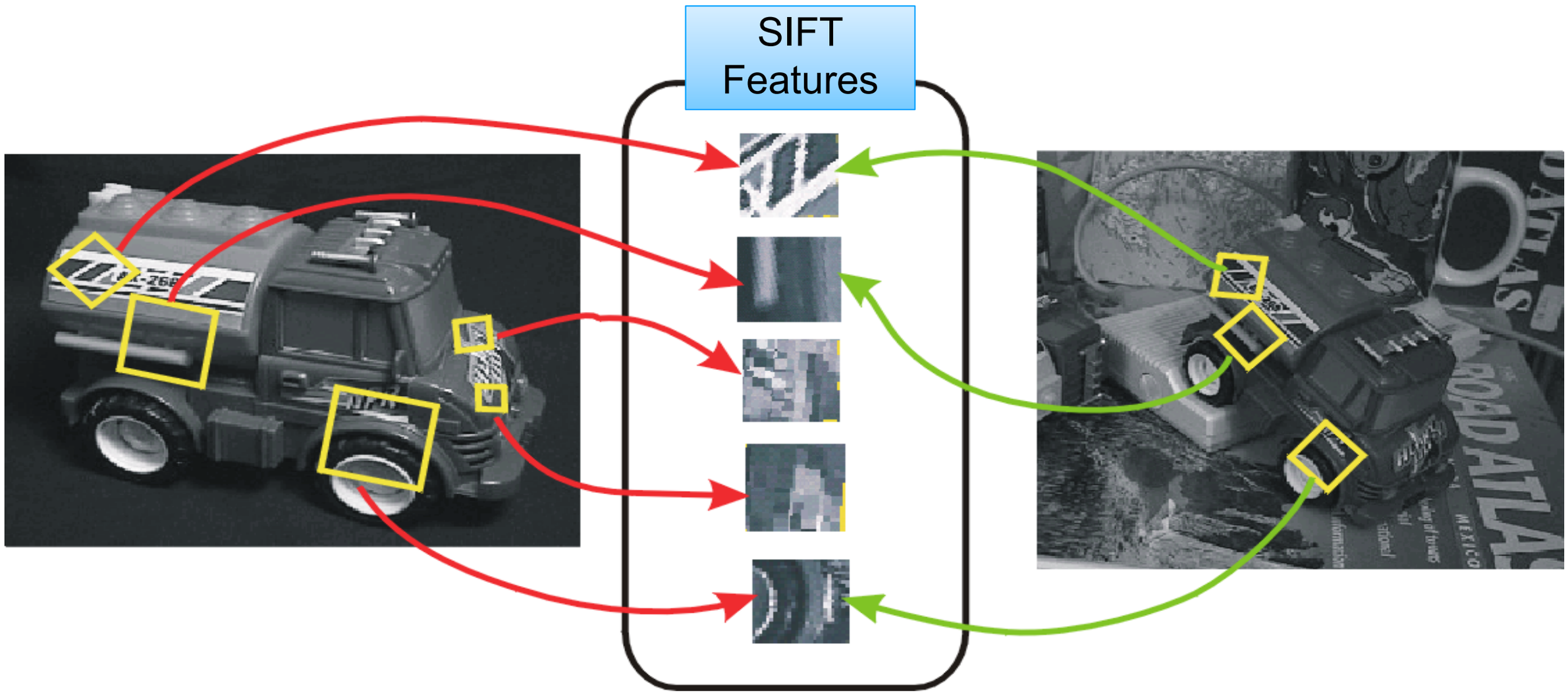
Scale-Space Blob Detector: Example

Scale Invariant Feature Transform (SIFT)

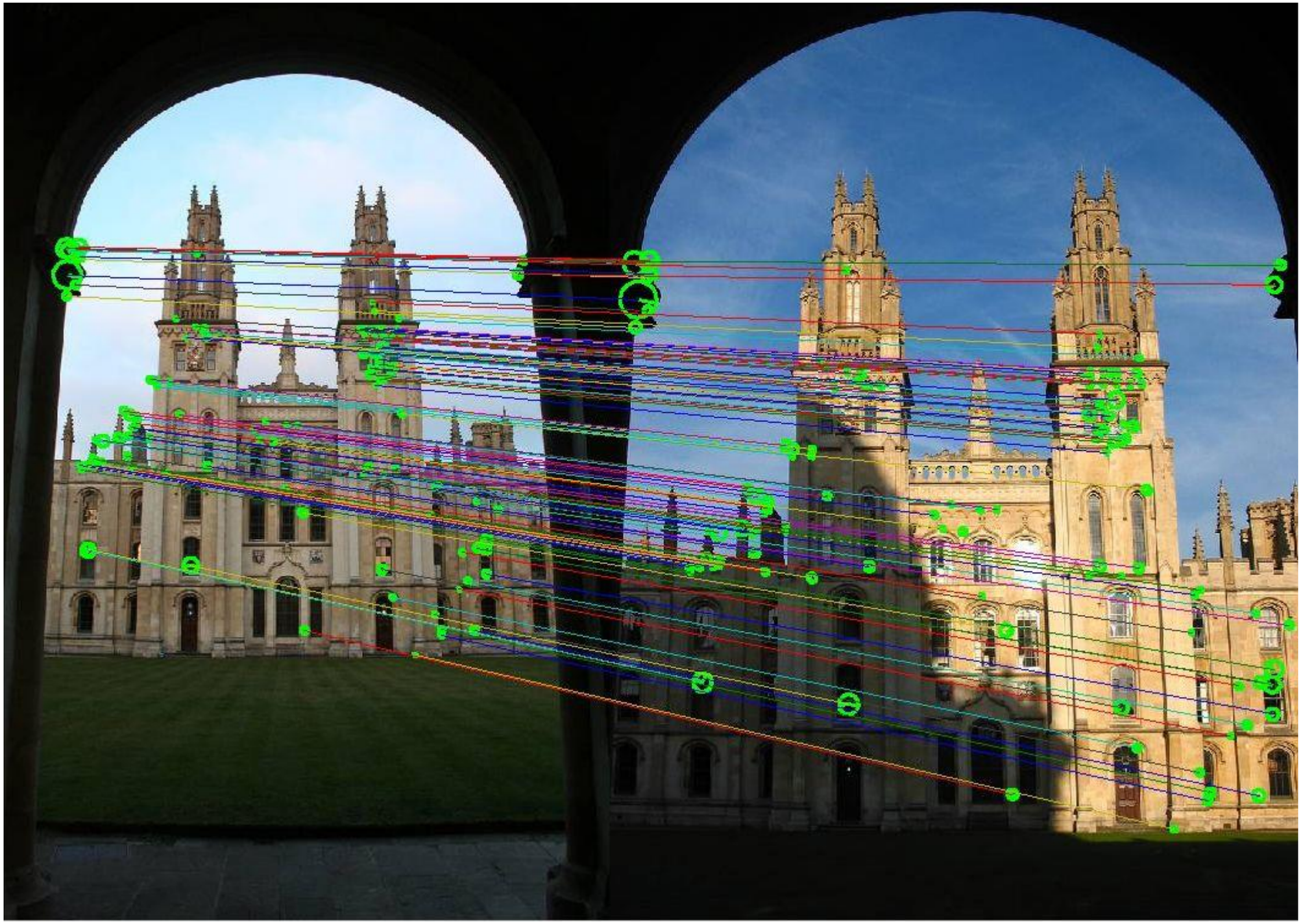


SIFT Feature

Image content is transformed into local feature coordinates that are invariant to translation, rotation, scale, and other imaging parameters



SIFT Feature



SIFT Feature

- David G Lowe, Distinctive Image Features from Scale-invariant Keypoints. International journal of computer vision (IJCV), Vol.60, No.2, pp.91-110, 2004.

Google Scholar

David Lowe

FOLLOW

Professor Emeritus, Computer Science Dept., University of British Columbia
Verified email at cs.ubc.ca - Homepage

Computer Vision Object Recognition

TITLE	CITED BY	YEAR
Distinctive image features from scale-invariant keypoints DG Lowe International journal of computer vision 60 (2), 91-110	66413	2004
Object recognition from local scale-invariant features DG Lowe International Conference on Computer Vision, 1999, 1150-1157	22256	1999
Fast Approximate Nearest Neighbors with Automatic Algorithm Configuration. M Muja, DG Lowe VISAPP (1) 2, 331-340	3639	2009
Automatic panoramic image stitching using invariant features M Brown, DG Lowe International Journal of Computer Vision 74 (1), 59-73	3133	2007

US6711293B1
United States

Download PDF Find Prior Art Similar

Inventor: David G. Lowe

Current Assignee : University of British Columbia

Worldwide applications

2000 • US

Application US09/519,893 events

1999-03-08 • Priority to US12336999P

2000-03-06 • Application filed by University of British Columbia

2004-03-23 • Publication of US6711293B1

2004-03-23 • Application granted

2020-03-06 • Anticipated expiration

2020-03-15 • Application status is Active

SIFT: Motivation

- The **Harris operator** is not invariant to scale and correlation is not invariant to rotation

- For better image matching, Lowe's goal was to develop an interest operator that is invariant to scale and rotation.

- Also, Lowe aimed to create a descriptor that was robust to the variations corresponding to typical viewing conditions. The descriptor is the most-used part of SIFT.

SIFT: Overall Procedure at a High Level

1. Scale-space extrema detection

Search over multiple scales and image locations.

2. Keypoint localization

Fit a model to determine location and scale.
Select key points based on a measure of stability.

3. Orientation assignment

Compute best orientation(s) for each keypoint region.

4. Keypoint description

Use local image gradients at selected scale and rotation to describe each keypoint region.

SIFT: Overall Procedure at a High Level

1. Scale-space extrema detection

Search over multiple scales and image locations.

2. Keypoint localization

Fit a model to determine location and scale.
Select key points based on a measure of stability.

detector

3. Orientation assignment

descriptor

Compute best orientation(s) for each keypoint region.

4. Keypoint description

Use local image gradients at selected scale and rotation to describe each keypoint region.

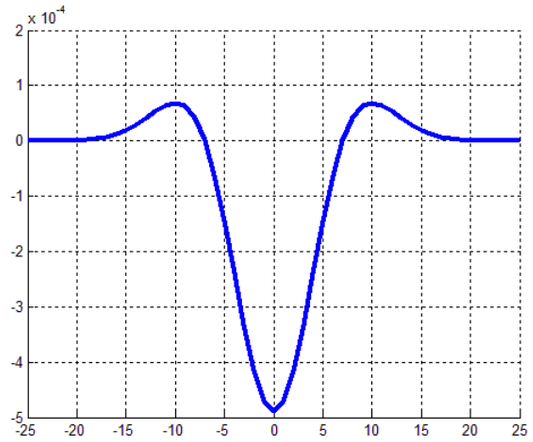
1. Scale-Space Extrema Detection

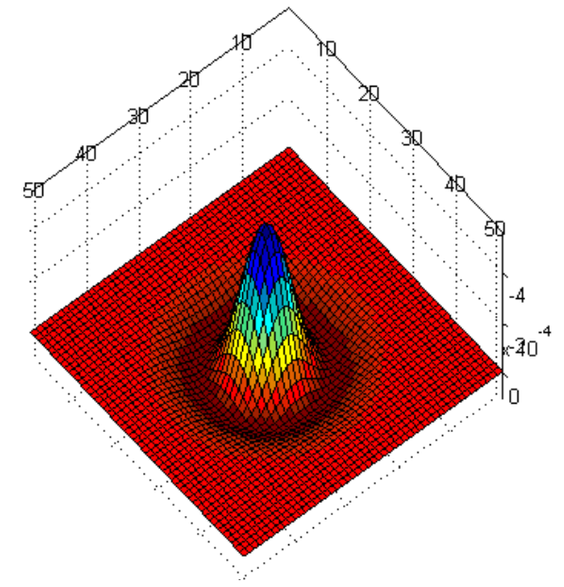
- **Goal:** Identify locations and scales that can be repeatable assigned under different views of the same scene or object.
- **Method:** search for stable features across multiple scales using a continuous function of scale.

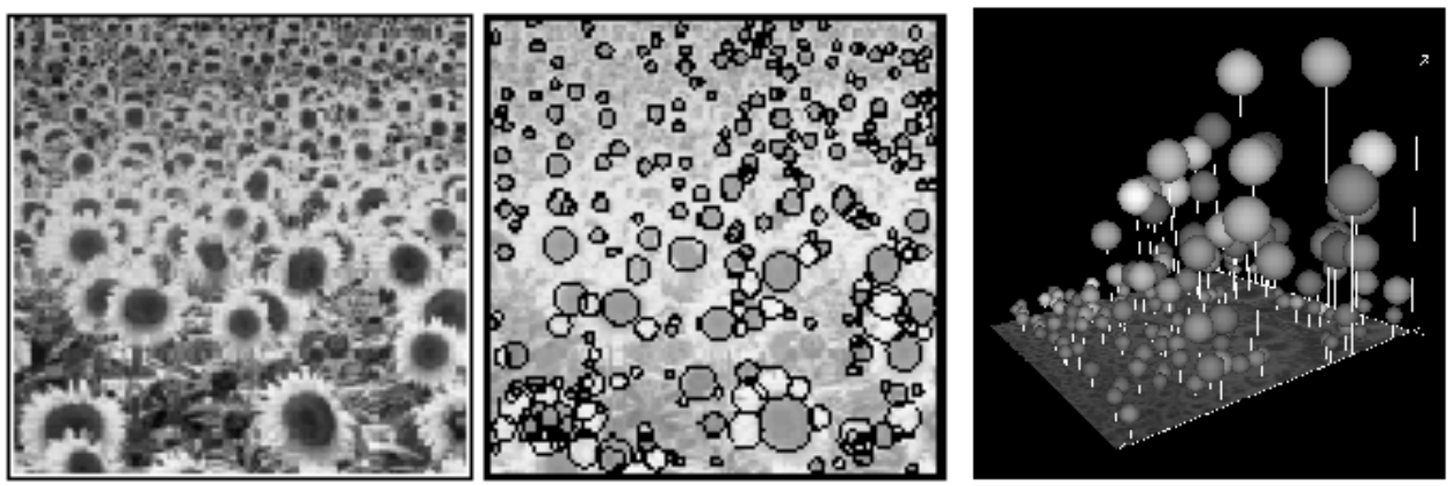
- **The scale space of an image is a function** $L(x, y, \sigma)$ that is produced from the convolution of a Gaussian kernel (at different scales) with the input image.

Scale-Space Interest Points

- **Laplacian of Gaussian** kernel
  - Scale normalized (x by scale^2)
  - Proposed by Lindeberg
- Scale-space detection
  - Find local maxima across scale/space
  - A good “blob” detector







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

$$\nabla^2 G(x, y, \sigma) = \frac{\partial^2 G}{\partial x^2} + \frac{\partial^2 G}{\partial y^2}$$

[T. Lindeberg IJCV 1998]

Difference of Gaussians (DoG)

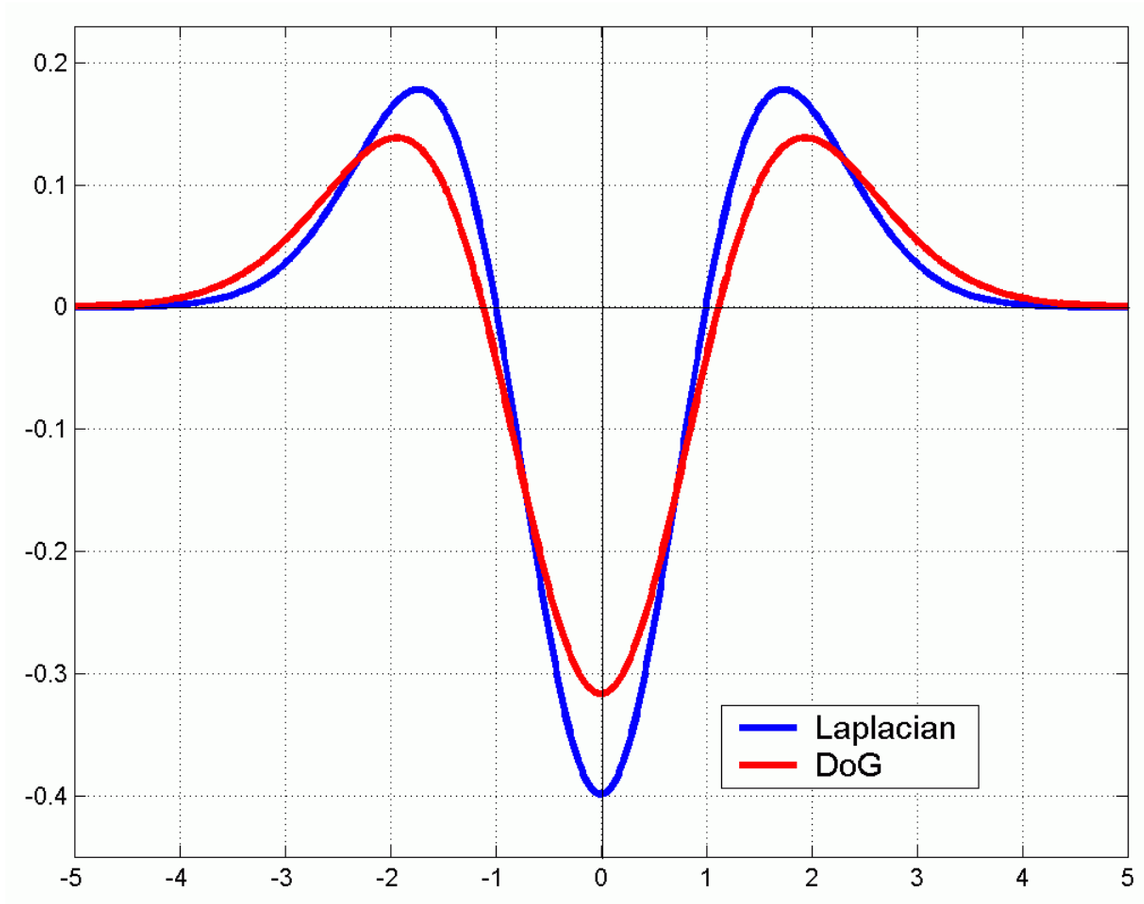
- Approximating the Laplacian with a difference of Gaussians:

$$L = \sigma^2 \left(g_{xx}(x,y,\sigma) + g_{yy}(x,y,\sigma) \right)$$

(Laplacian)

$$DoG = g(x,y,2\sigma) - g(x,y,\sigma)$$

Difference of gaussian with scales 2 σ and σ

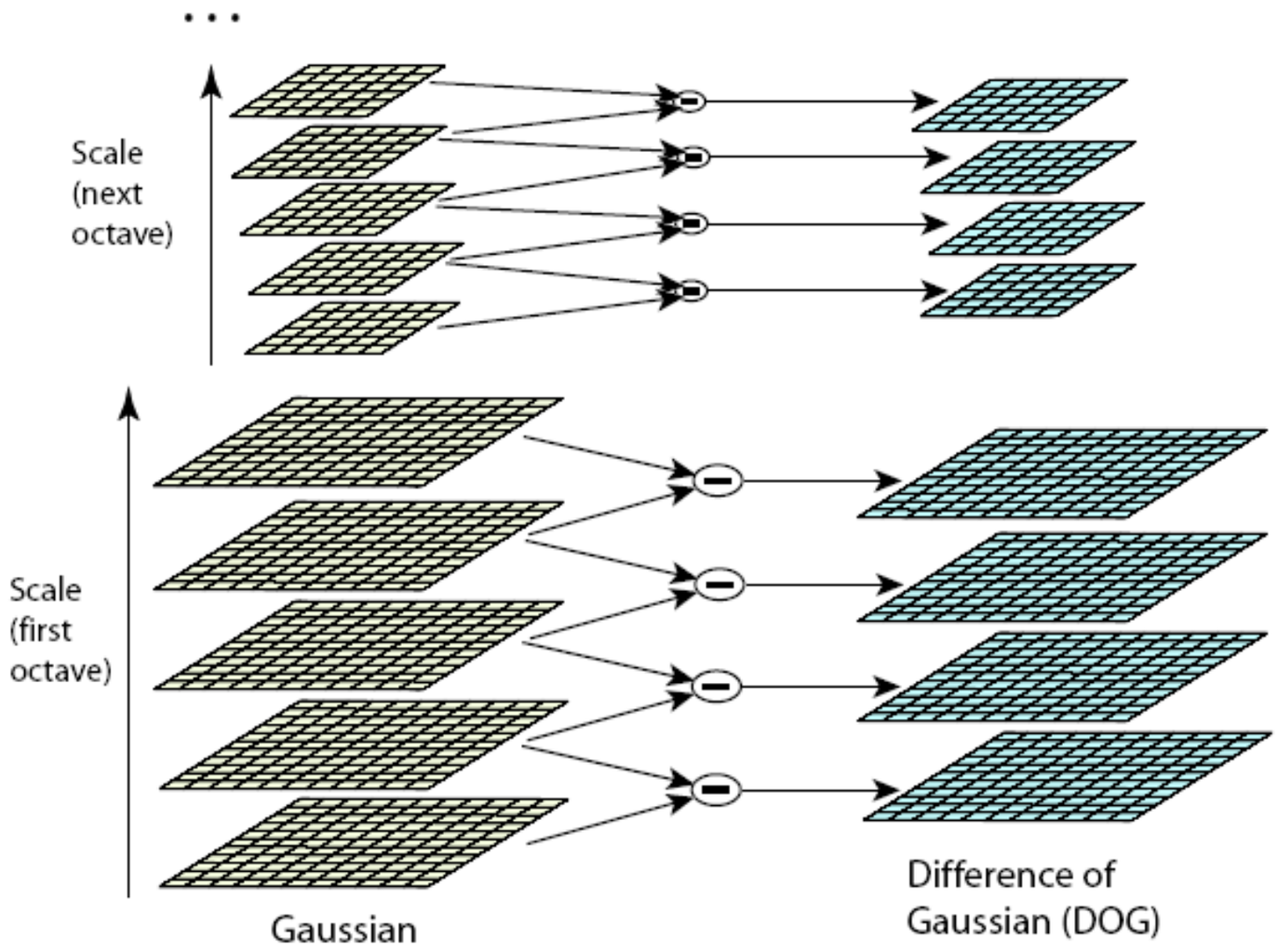


In general:

$$DoG = g(x,y,k\sigma) - g(x,y,\sigma) \approx (k-1)\sigma^2 \mathbf{L}$$

Lowe's Pyramid Scheme

- **Difference of Gaussian** is obtained as the difference of Gaussian blurring of an image with two different σ
- let it be σ and $k\sigma$. This process is done for different octaves (组) of the image in Gaussian Pyramid.



Gaussian Pyramid:

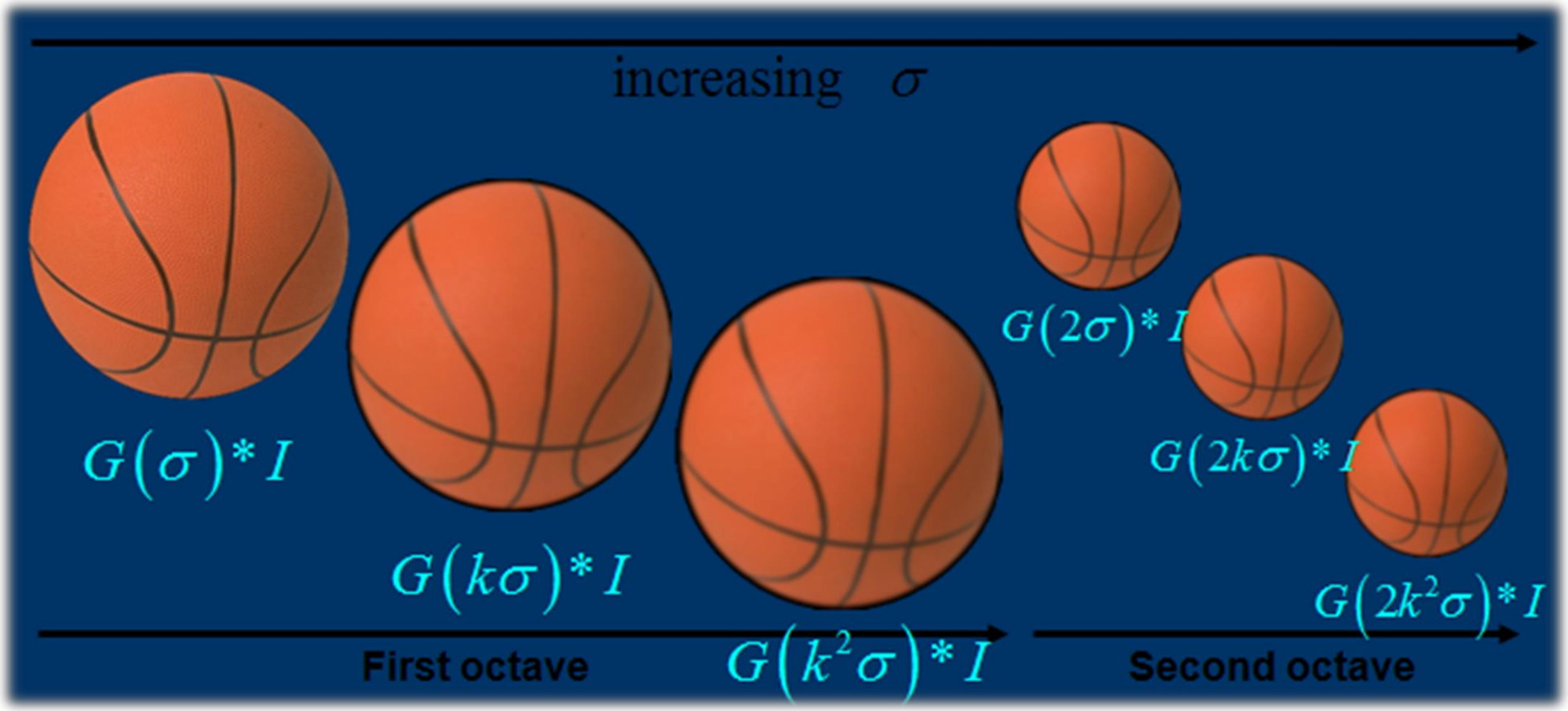
$$2^{i-1}(\sigma, k\sigma, k^2\sigma, \cdots, k^{n-1}\sigma)$$

$$k = 2^{1/s}$$

$\boldsymbol{S}$: # of scale levels in the i-th octave

Lowe's Pyramid Scheme

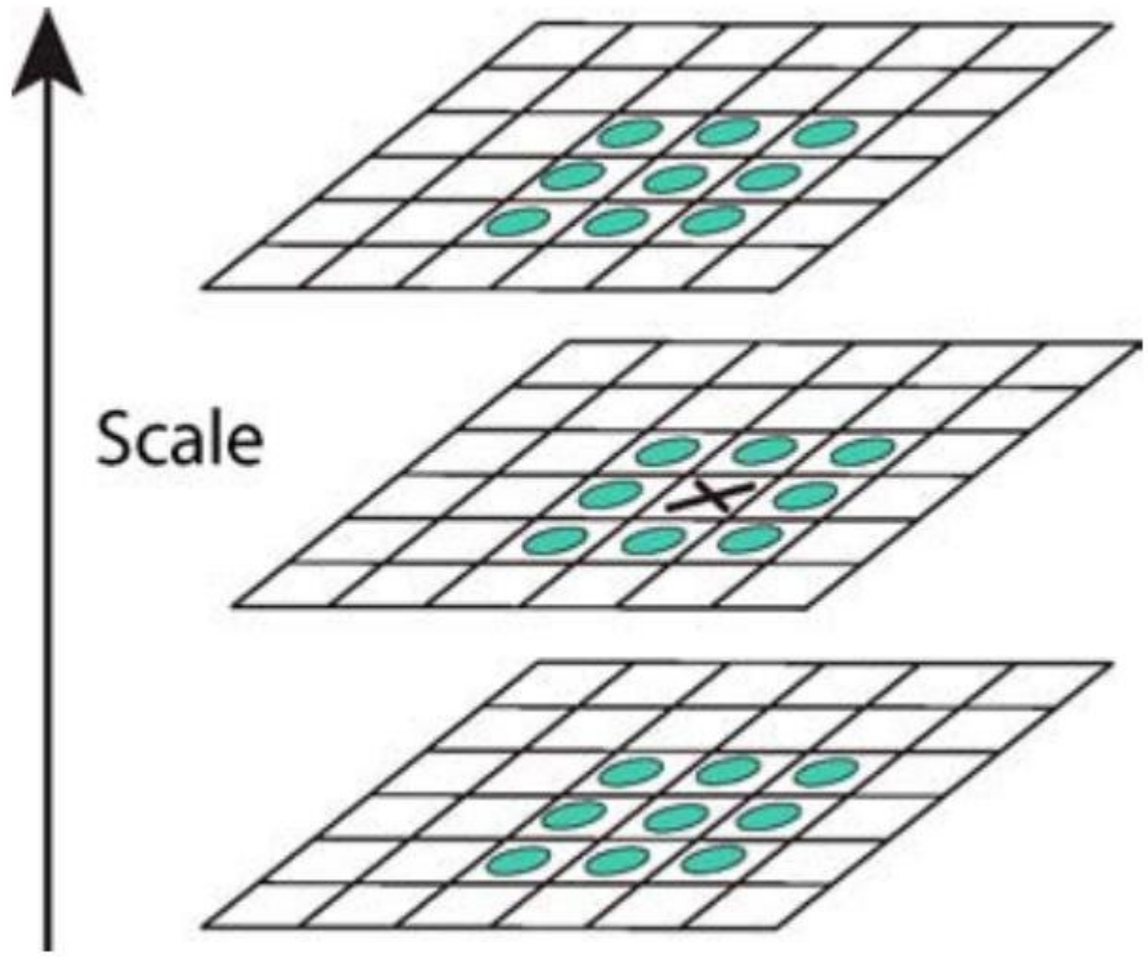
- **Difference of Gaussian** is obtained as the difference of Gaussian blurring of an image with two different σ
- let it be σ and $k\sigma$. This process is done for different octaves of the image in Gaussian Pyramid.



2. Keypoint Localization

- Detect Max and Min of DoG in scale space

- Each point is compared to its 8 neighbors in the current image, and 9 neighbors each in the scales above and below



For each max or min found, output is the location and the scale.

Keypoint Localization

- Once a keypoint candidate is found, perform a detailed fit to nearby data to determine
  - location, scale, and ratio of principal curvatures
- In initial work keypoints were found at location and scale of a central sample point
- In newer work, they fit a 3D quadratic function to improve interpolation accuracy
- The Hessian matrix was used to eliminate edge responses

$$D(\mathbf{x}) = D + \frac{\partial D}{\partial \mathbf{x}}^T \mathbf{x} + \frac{1}{2}\mathbf{x}^\mathbf{T} \frac{\partial^2 D}{\partial \mathbf{x}^2}\mathbf{x}$$

$$\Rightarrow \quad \hat{\mathbf{x}} = -\frac{\partial^2 D}{\partial \mathbf{x}^2}^{-1} \frac{\partial D}{\partial \mathbf{x}}$$

$\mathbf{x}$: the sample point:

$\hat{\mathbf{x}}$: location of the extremum

Eliminating the Edge Response

- Reject flats: $D(\hat{\mathbf{x}}) < 0.03$
- Reject edges: $r < 10$

Let α be the eigenvalue with the largest magnitude and β be the smaller one.

2x2 Hessian matrix: $\mathbf{H} = \begin{bmatrix} D_{xx} & D_{xy} \\ D_{xy} & D_{yy} \end{bmatrix}$

Let $r = \alpha/\beta$, So $\alpha = r\beta$

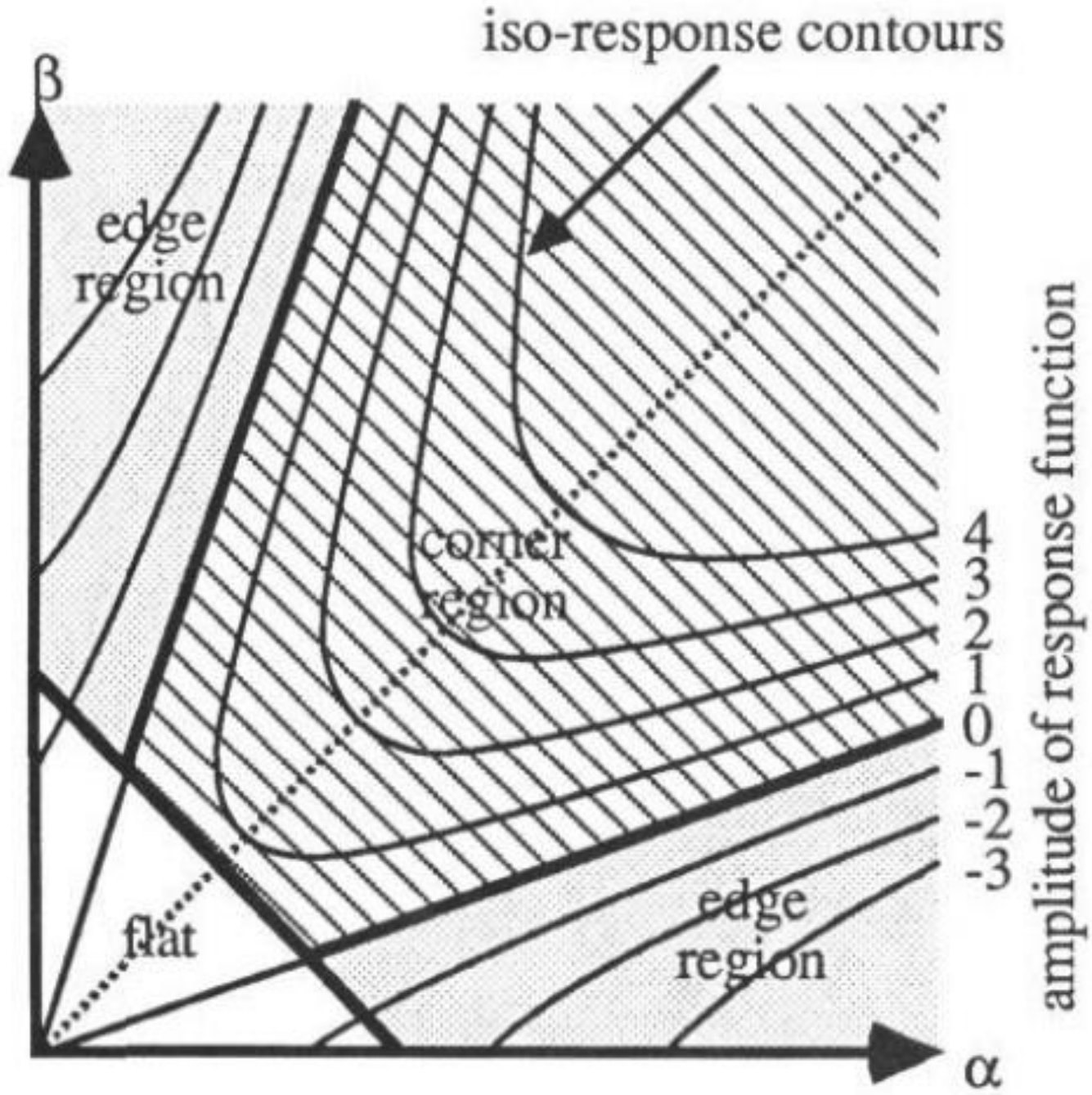
$$\mathrm{Tr}(\mathbf{H}) = D_{xx} + D_{yy} = \alpha + \beta,$$

$$\mathrm{Det}(\mathbf{H}) = D_{xx}D_{yy} - (D_{xy})^2 = \alpha\beta.$$

$$\frac{\mathrm{Tr}(\mathbf{H})^2}{\mathrm{Det}(\mathbf{H})} = \frac{(\alpha+\beta)^2}{\alpha\beta} = \frac{(r\beta+\beta)^2}{r\beta^2} = \frac{(r+1)^2}{r},$$

- What does this look like?

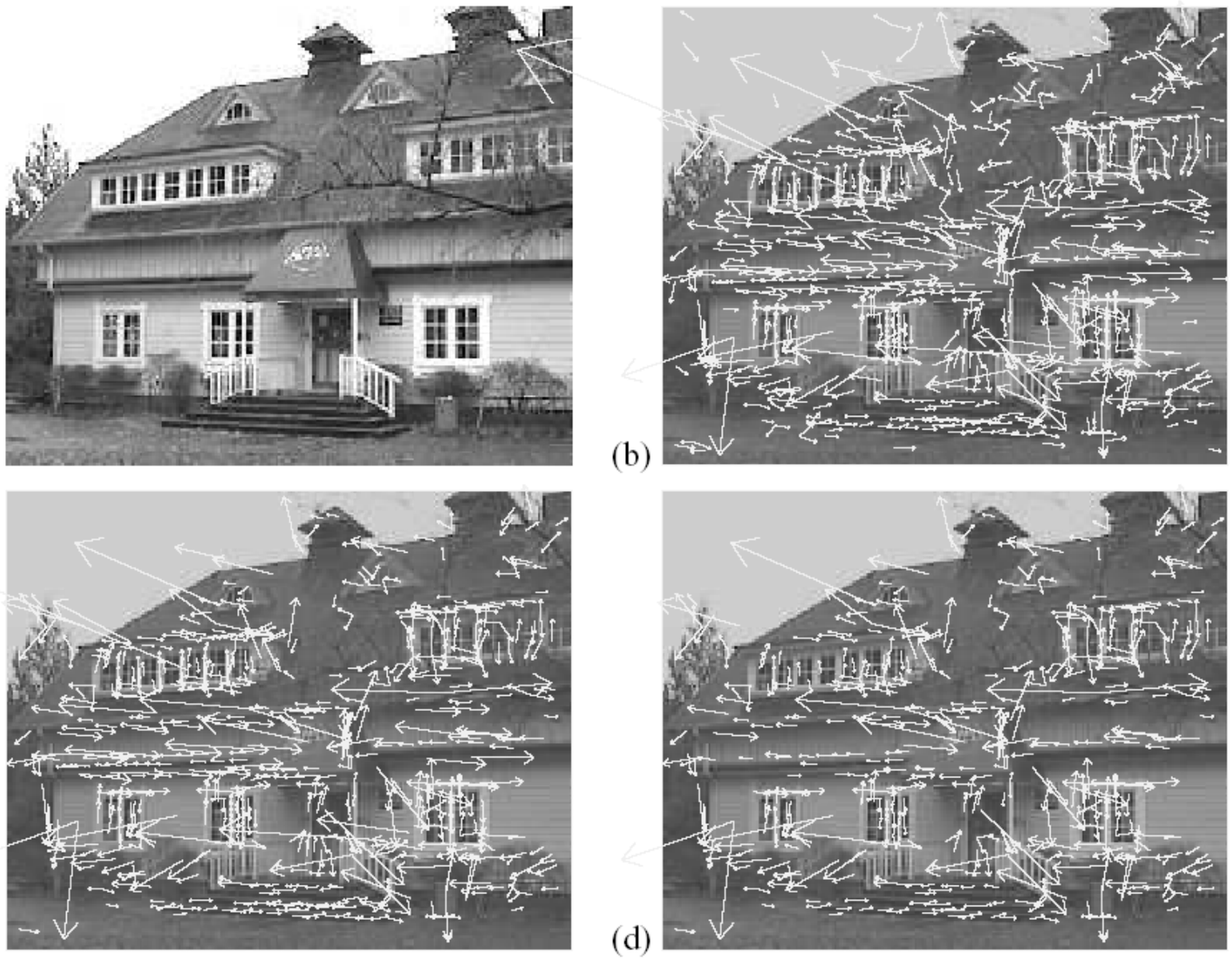
Corner / Edge / Flat Classification



Source: C. Harris and M. Stephens, “A combined corner and edge detector,” 1988.

Keypoint Localization with Orientation

233x189



832

initial keypoints

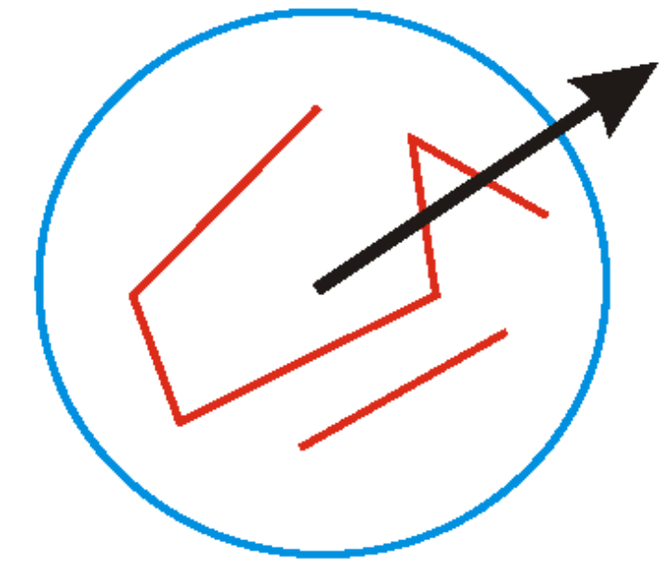
729

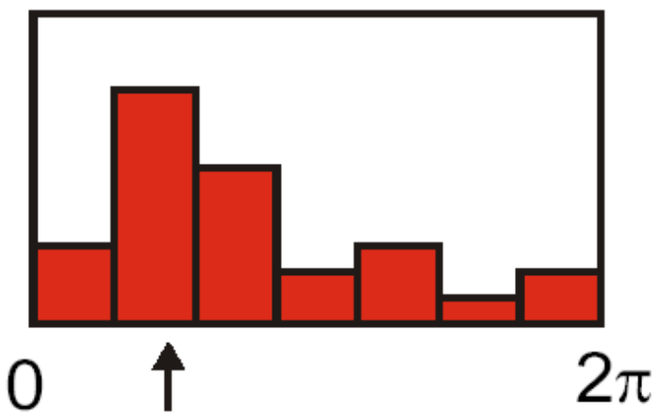
keypoints after gradient threshold

536

keypoints after ratio threshold

3. Orientation Assignment





- For each image sample, $L(x, y)$, at this scale:
  - **Gradient magnitude:**

$$m(x,y) = \sqrt{(L(x+1,y) - L(x-1,y))^2 + (L(x,y+1) - L(x,y-1))^2}$$

  - **Orientation:**

$$\theta(x,y) = \tan^{-1}((L(x,y+1) - L(x,y-1))/(L(x+1,y) - L(x-1,y)))$$

- Create histogram of local gradient directions at selected scale
- Assign canonical orientation at peak of smoothed histogram
- Each key specifies stable 2D coordinates (x, y, scale, orientation)

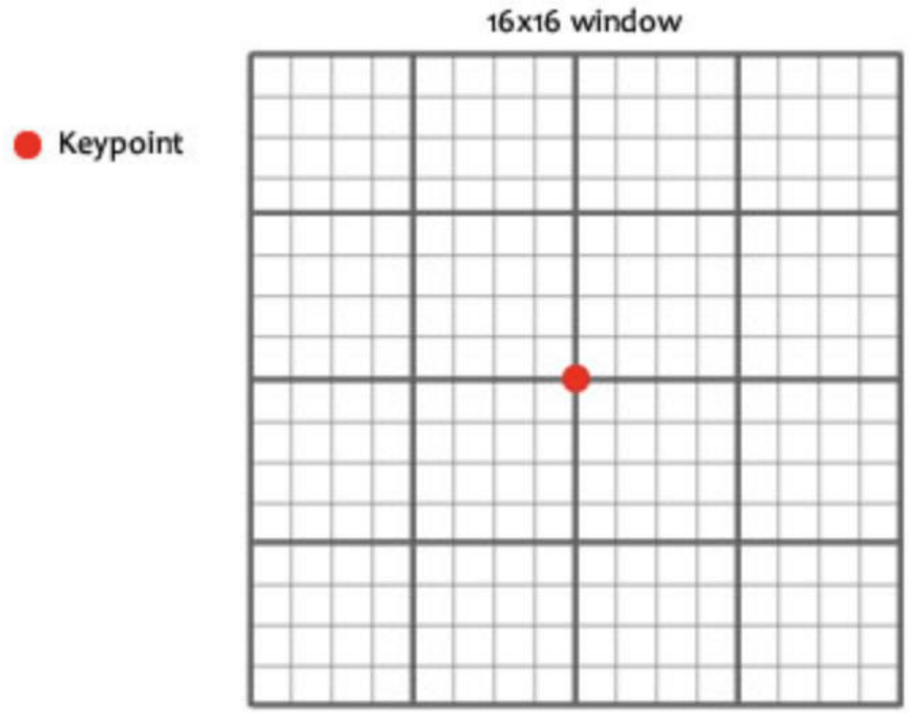
Keypoint Localization with Orientation

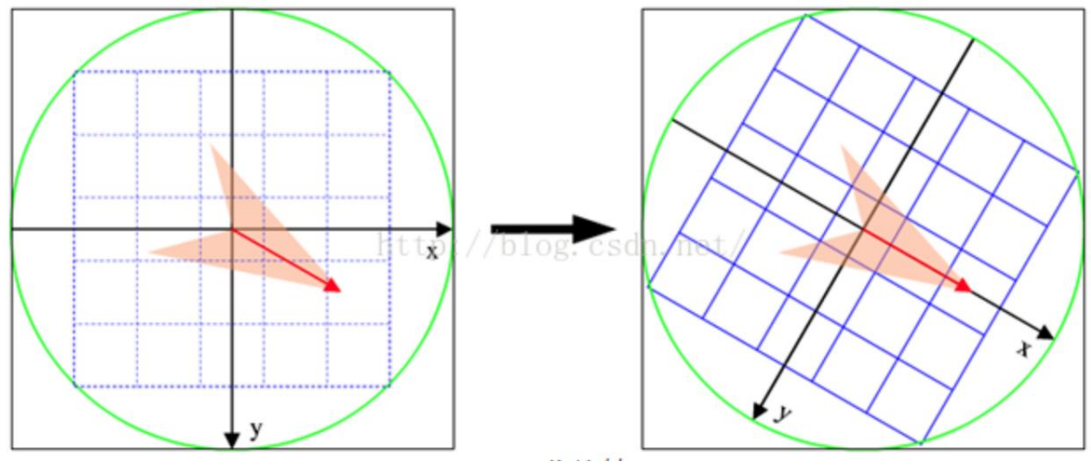
- At this point, each keypoint has
  - location
  - scale
  - orientation
- Next is to compute a descriptor for the local image region about each keypoint that is
  - highly distinctive
  - invariant as possible to variations

  such as changes in viewpoint and illumination

SIFT Descriptor

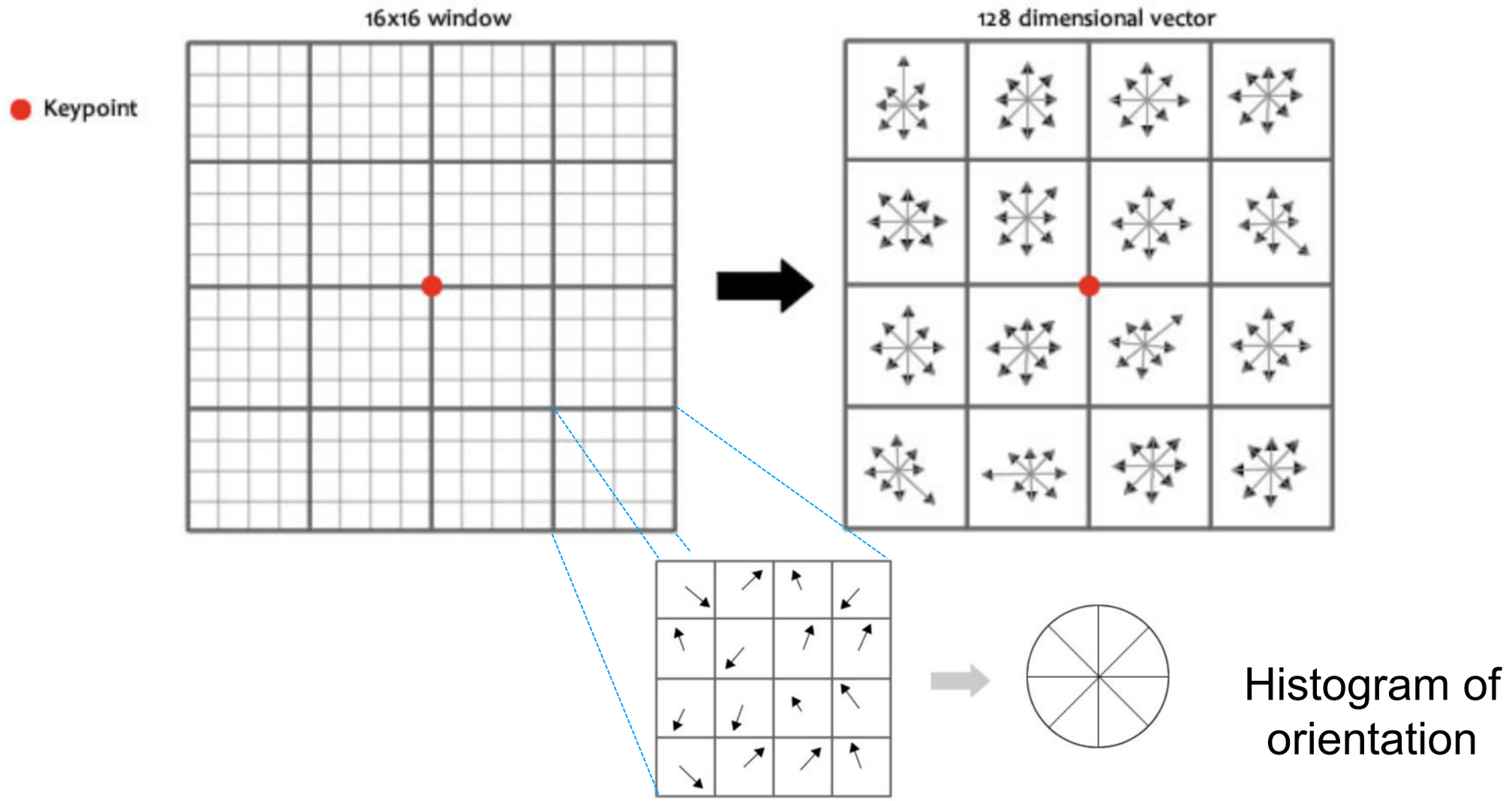
- Divide the 16x16 window into a 4x4 grid of cells





SIFT Descriptor

- Compute an orientation histogram for each cell

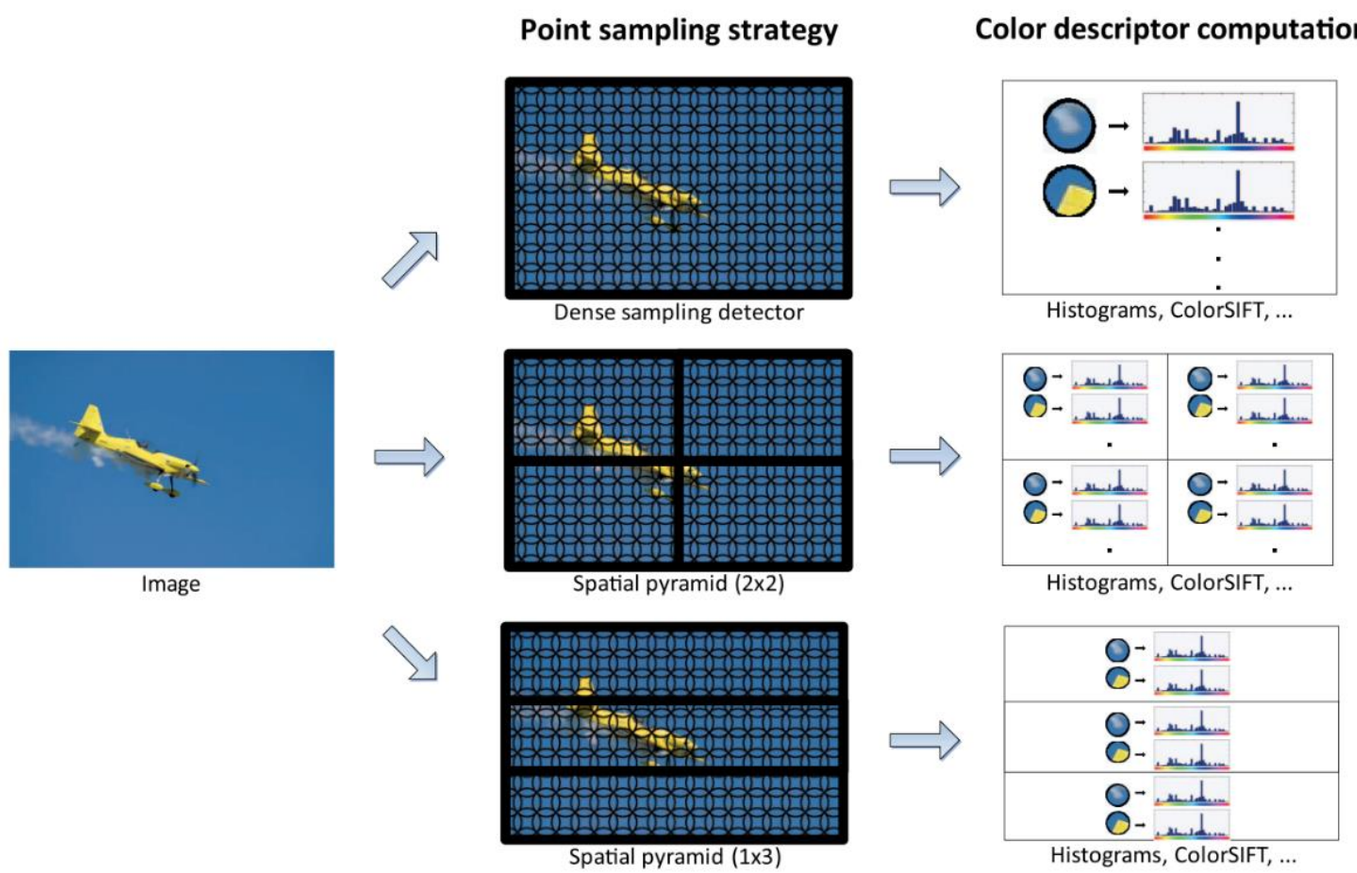


- 16 cells * 8 orientations = 128 dimensional descriptor

SIFT Descriptor: Example

Remark

- Local features are a popular tool for image description nowadays, with a variety of applications, such as image matching, 3D reconstruction, motion tracking, object recognition, robot navigation.



Source: Koen E.A. van de Sande et al., “Evaluating Color Descriptors for Object and Scene Recognition,” IEEE PAMI, 2010.

Lec4 Machine Learning

·AIPKU·

人工智能引论实践课 计算机视觉小班

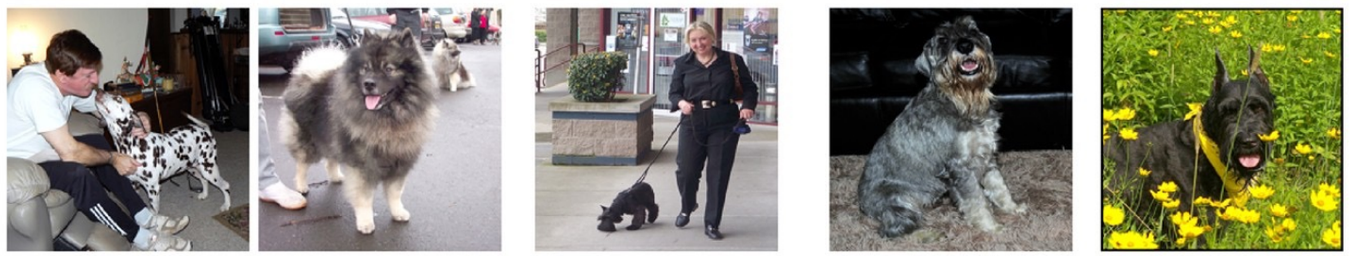
主讲人：刘家瑛

为什么使用机器学习算法

- 人工设置的规则很难应对复杂的实际分类需求
  - 分类类别数量大
  - 复杂的背景分布
  - 遮挡
  - 部分内容缺失
  - 对象大小变化
  - .....

Cat

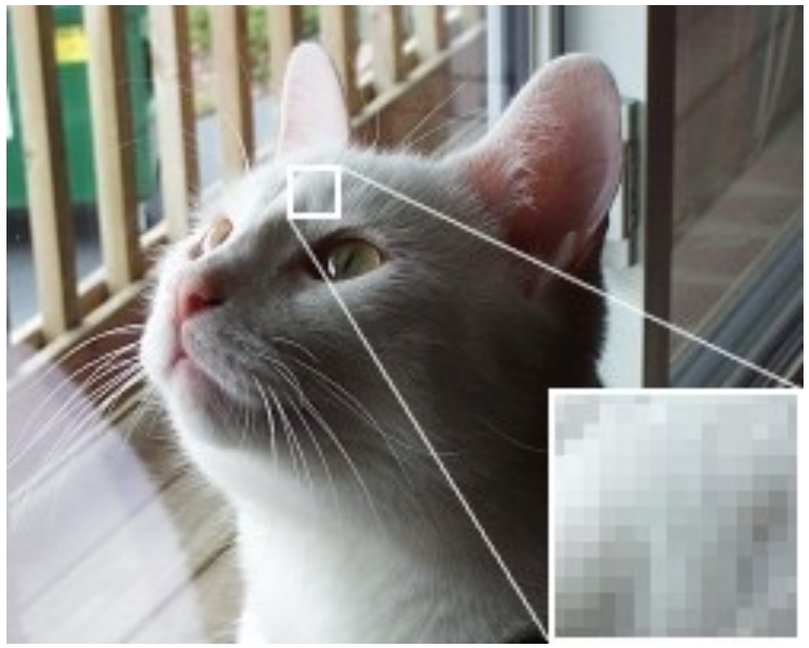
Dog



[ILSVRC, Deng et al., CVPR'09]

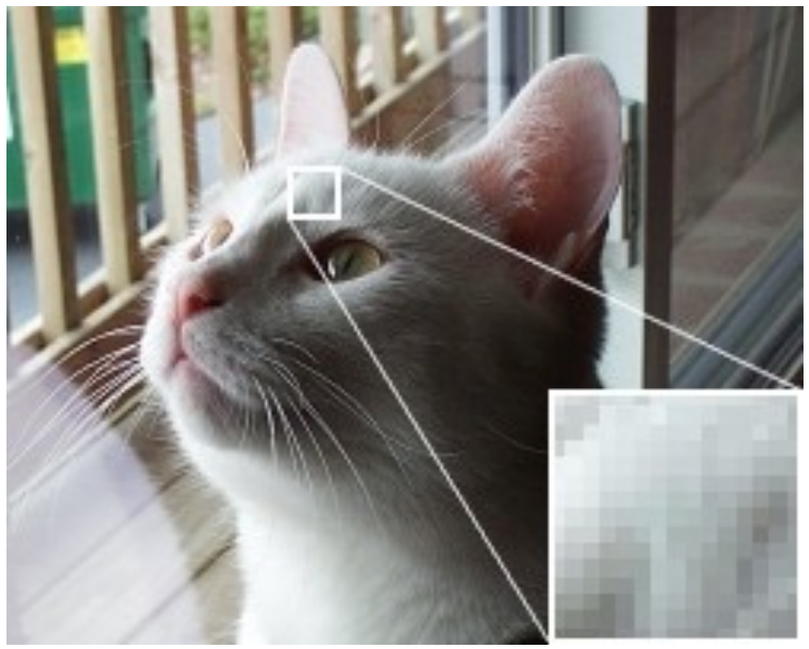
如何表达图像?

- 原始像素?



如何表达图像？

- ~~原始像素~~



- 使用特征
  - HoG
  - SIFT
  - 卷积神经网络

图像表达 + 分类器

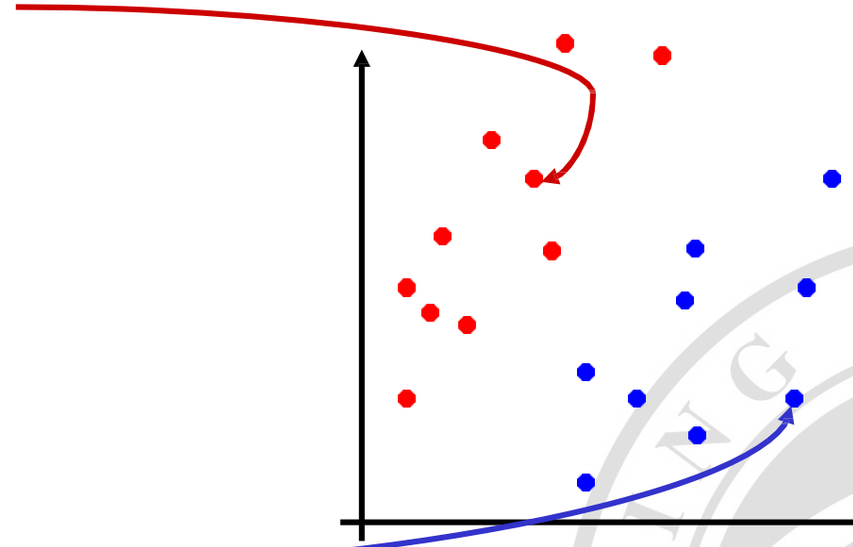
- 将图像映射到高维空间中的一点

特征提取

$$\begin{pmatrix} 4.8 \\ 1.9 \\ -2 \\ 6.3 \end{pmatrix}$$

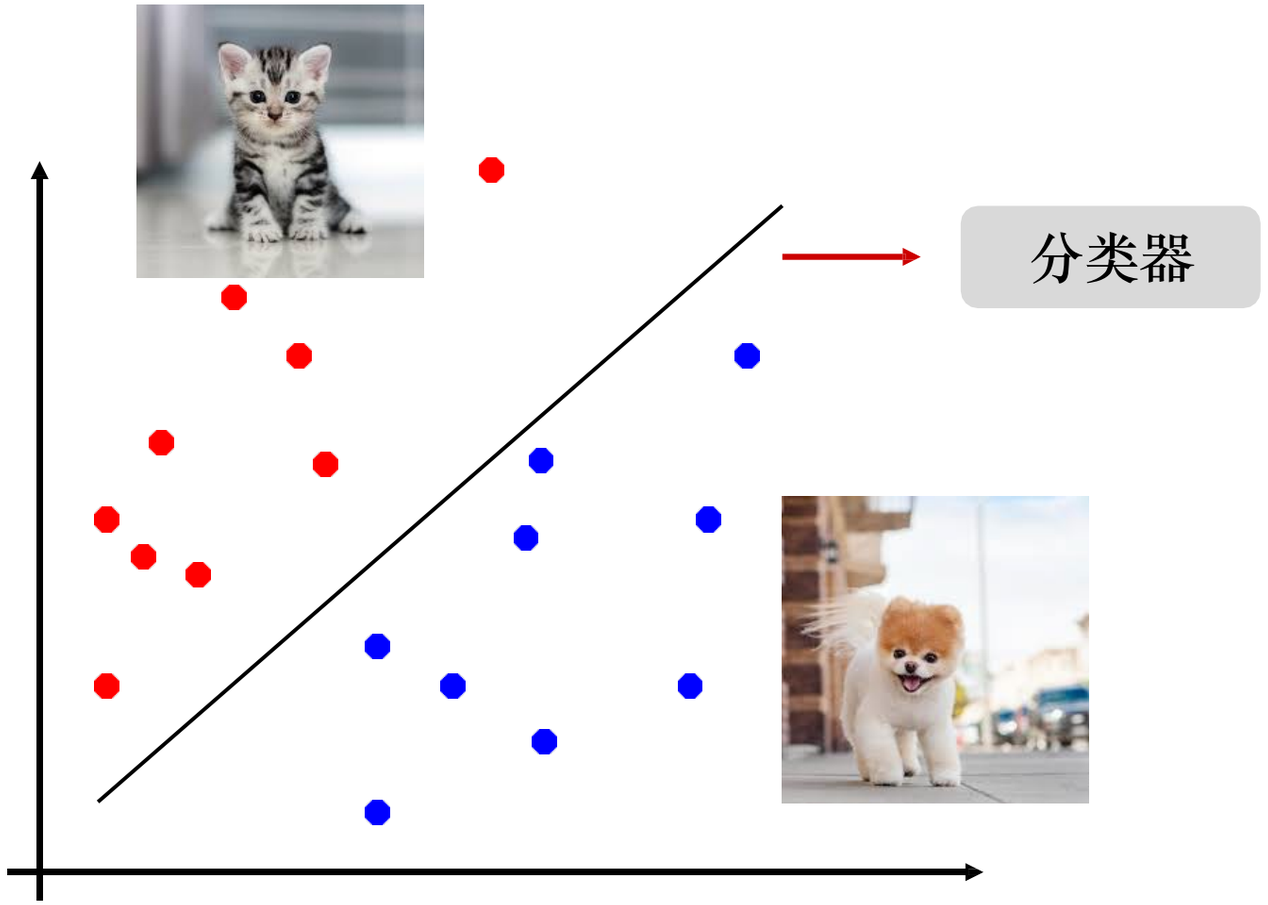
特征提取

$$\begin{pmatrix} 1.3 \\ -4 \\ -9 \\ 5.8 \end{pmatrix}$$



图像表达 + 分类器

- 使用分类器获得该点所属的类别



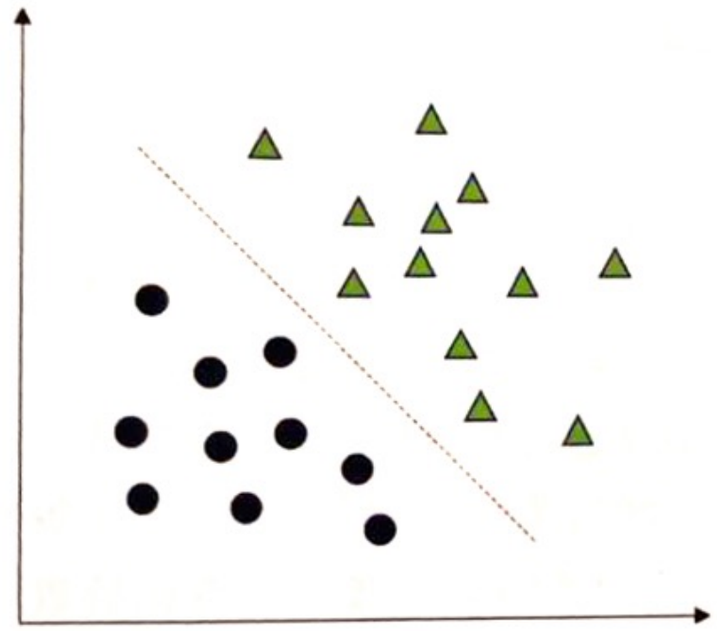
- 常用分类器
  - **支持向量机** (Support Vector Machine, SVM)
  - **朴素贝叶斯**、决策树、随机森林、神经网络等

模式识别与机器学习简介

- 感知机
- 支持向量机 (SVM)

感知机 (Perceptron)

- 求出将训练数据进行线性划分的分离超平面



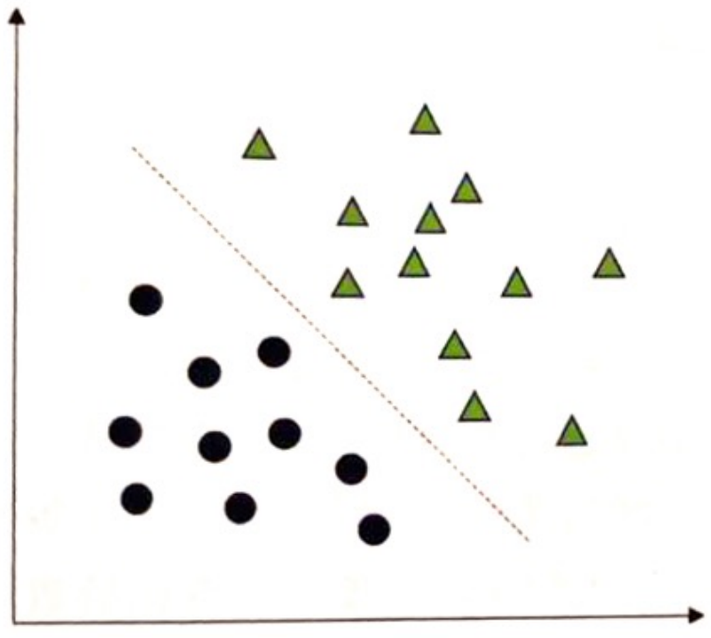
感知机 (Perceptron)

- 求出将训练数据进行线性划分的分离超平面

- 感知机模型 (w 和 b 为模型参数):

$$f(\boldsymbol{x}) = \text{sign}\,(\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x} + b)$$

$$\text{sign}\,(x) = \begin{cases} -1 & x < 0 \\ 1 & x \geq 0 \end{cases}$$



- $\boldsymbol{w}$ 为法向量，决定超平面的方向
- b 为位移项，决定超平面与原点之间的距离

感知机 (Perceptron)

- 感知机模型 (w 和 b 为模型参数):

$$f(\boldsymbol{x}) = \text{sign}(\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x} + b)$$

$$\text{sign}(x) = \begin{cases} -1 & x < 0 \\ 1 & x \geq 0 \end{cases}$$

- 给定训练集 $D = \{(\boldsymbol{x}_1, y_1), (\boldsymbol{x}_2, y_2), \dots, (\boldsymbol{x}_m, y_m)\}$, $y_i \in \{1, -1\}$
- 如果 $(\boldsymbol{x}_i, y_i)$ 分类正确: $-y_i(\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}_i + b) < 0$
- 如果 $(\boldsymbol{x}_i, y_i)$ 分类错误: $-y_i(\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}_i + b) \geq 0$

感知机 (Perceptron)

- 感知机模型 (w 和 b 为模型参数):

$$f(\boldsymbol{x}) = \text{sign}(\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x} + b)$$

$$\text{sign}(x) = \begin{cases} -1 & x < 0 \\ 1 & x \geq 0 \end{cases}$$

- 给定训练集 $D = \{(\boldsymbol{x}_1, y_1), (\boldsymbol{x}_2, y_2), \cdots, (\boldsymbol{x}_m, y_m)\}$, $y_i \in \{1, -1\}$

- 训练损失函数 (最小化 $L(\boldsymbol{w}, b)$):

$$L(\boldsymbol{w}, b) = -\sum_{x_i \in M} y_i(\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}_i + b)$$

$x_i \in M$

M 为误分类点

随机梯度下降求解感知机

- 训练输入：训练数据集 $D = \{(\boldsymbol{x}_1, \mathrm{y}_1), (\boldsymbol{x}_2, y_2), \dots, (\boldsymbol{x}_m, y_m)\}$, $y_i \in \{1, -1\}$

  学习率 $\eta \in (0, 1]$

- 训练输出：感知机模型 $f(x) = \mathrm{sign}(\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x} + b)$

1) 选取初值 $\boldsymbol{w}_0$ 和 b_0

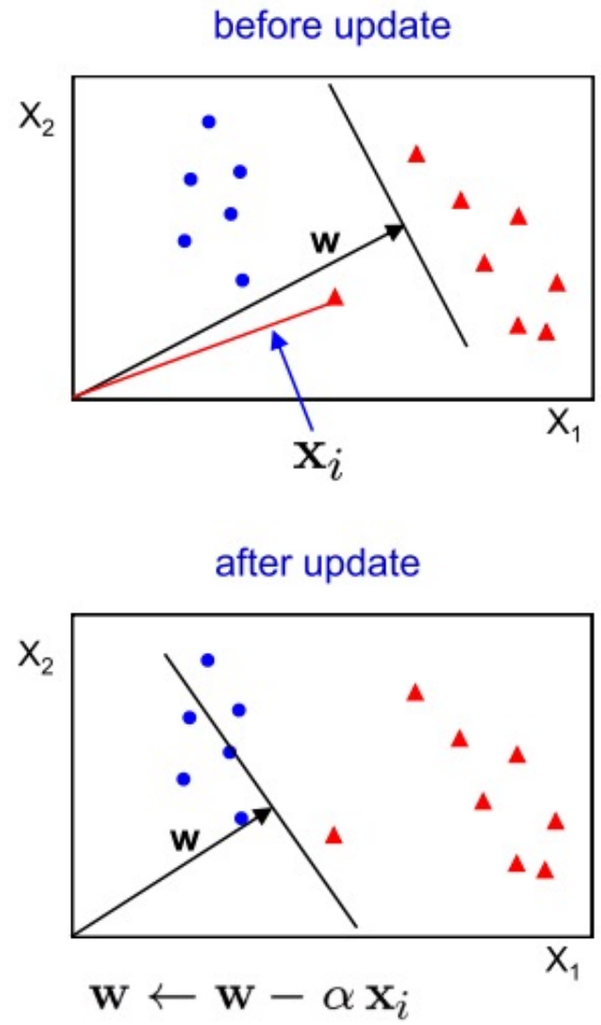
2) 从训练集中随机选取数据 $(\boldsymbol{x}_i, y_i)$

3) 如果分类错误 $-y_i(\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}_i + b) \geq 0$ 则更新权重：

$$\boldsymbol{w} \leftarrow \boldsymbol{w} + \eta y_i \boldsymbol{x}_i$$

$$b \leftarrow b + \eta y_i$$

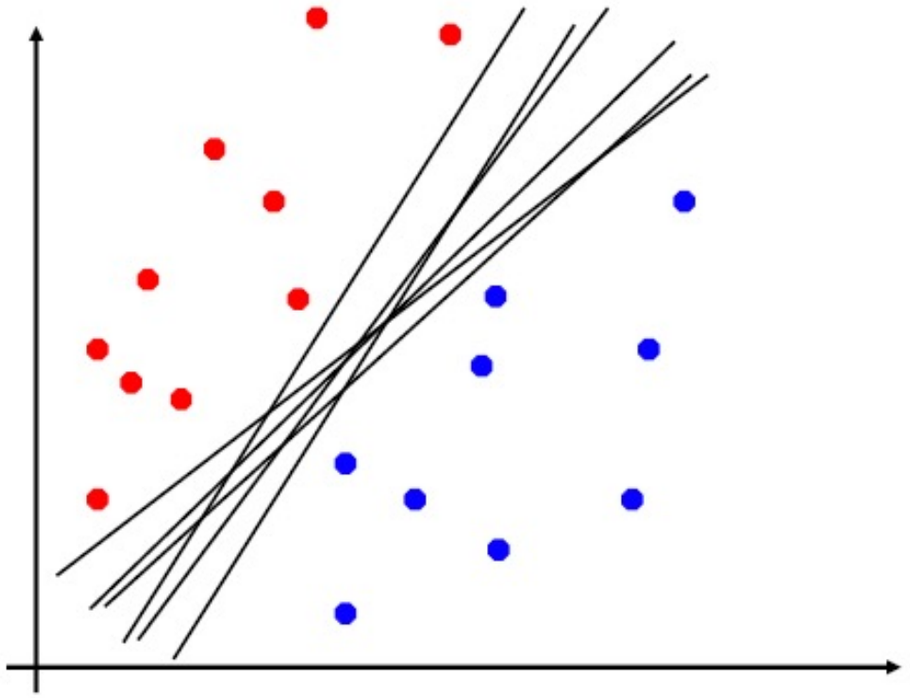
4) 重复 (2-3) 直到无误分类点



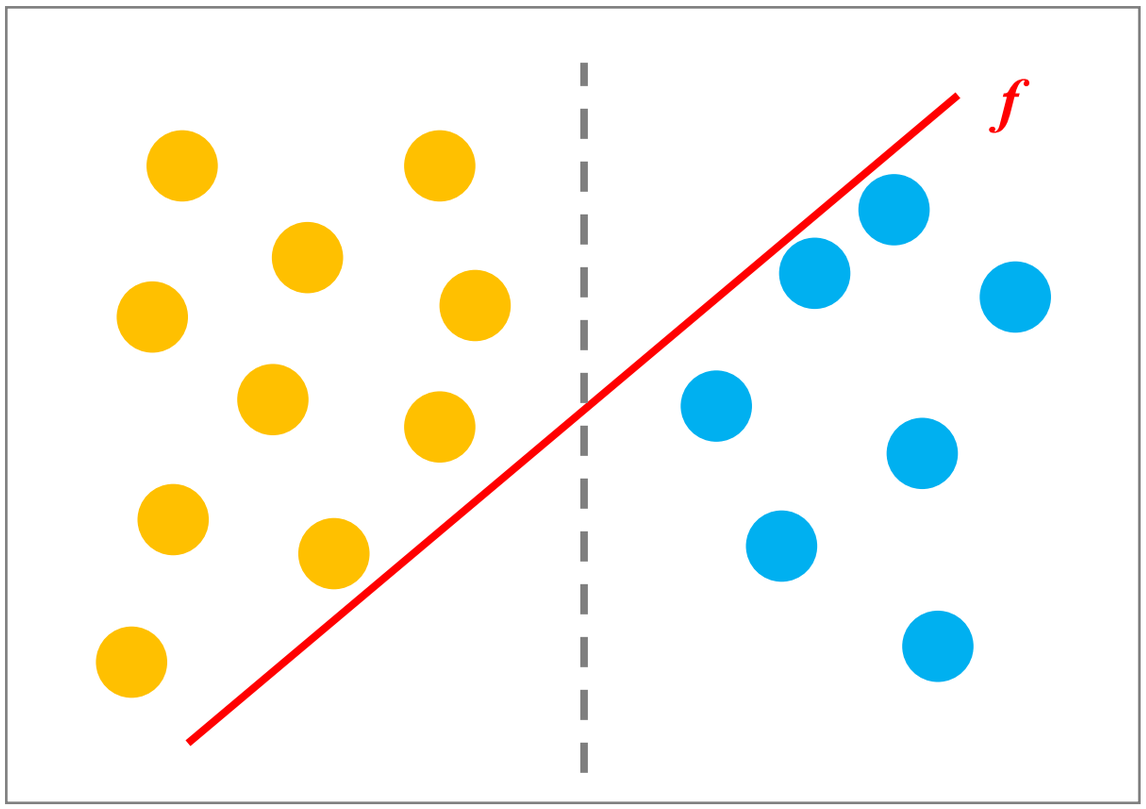
可证明：对于线性可分的数据，感知机在有限步内有解（证明略）

感知机

- 无法处理线性不可分数据
- 求解所得的超平面并不唯一
- 容易对训练数据过拟合

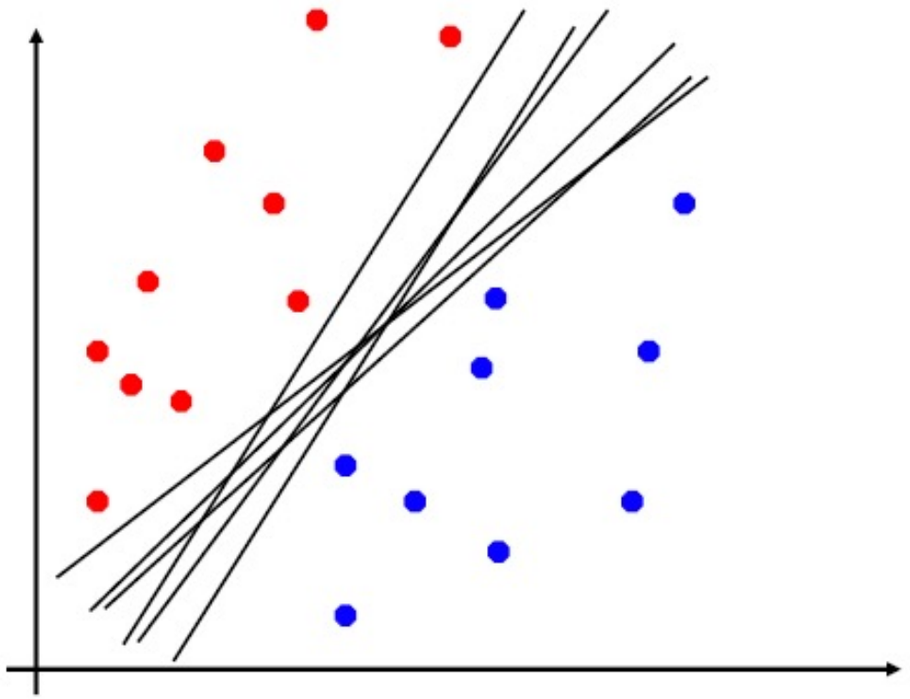


应该找哪一个呢？

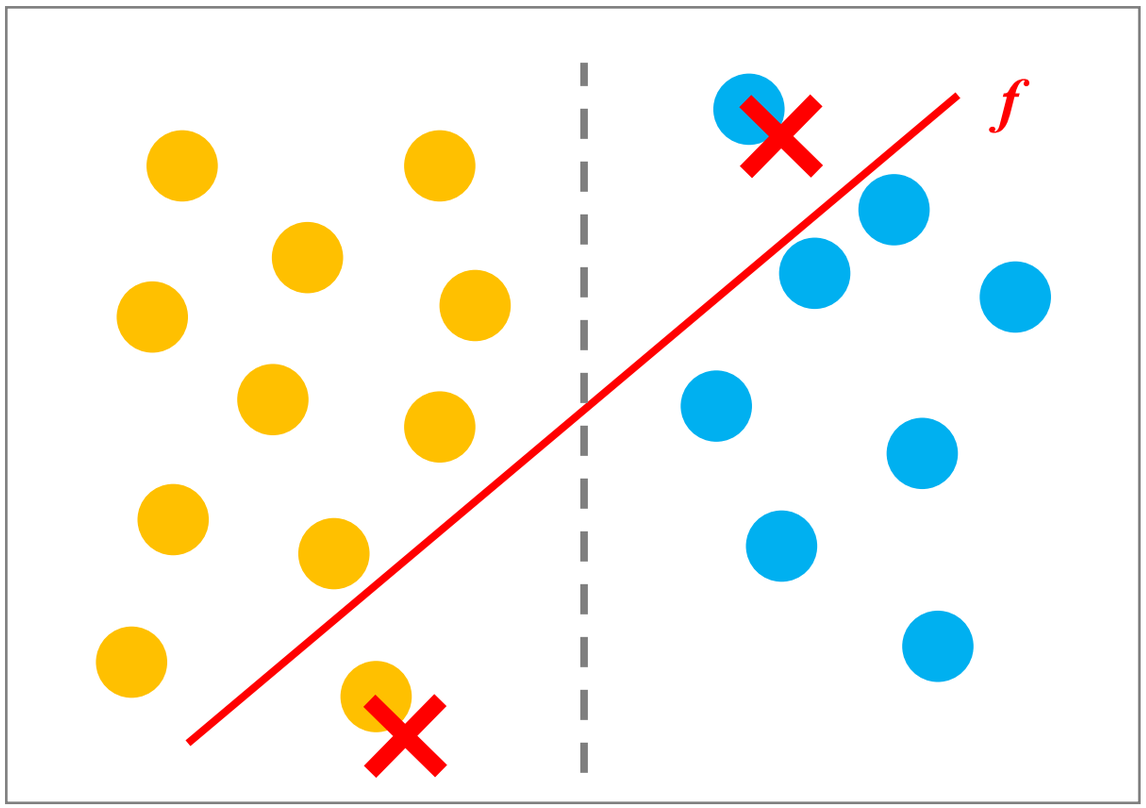


感知机

- 无法处理线性不可分数据
- 求解所得的超平面并不唯一
- 容易对训练数据过拟合



应该找哪一个呢？



模式识别与机器学习简介

- 感知机
- 支持向量机 (SVM)

支持向量机

- **最大间隔** 分类超平面
- 样本空间任意点 x 到超平面 $(\boldsymbol{w}, b)$ 的距离 $r = \dfrac{|\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x} + b|}{||\boldsymbol{w}||}$
- 如果所有的点都能分对： $y_i(\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}_i + b) \geq 1$

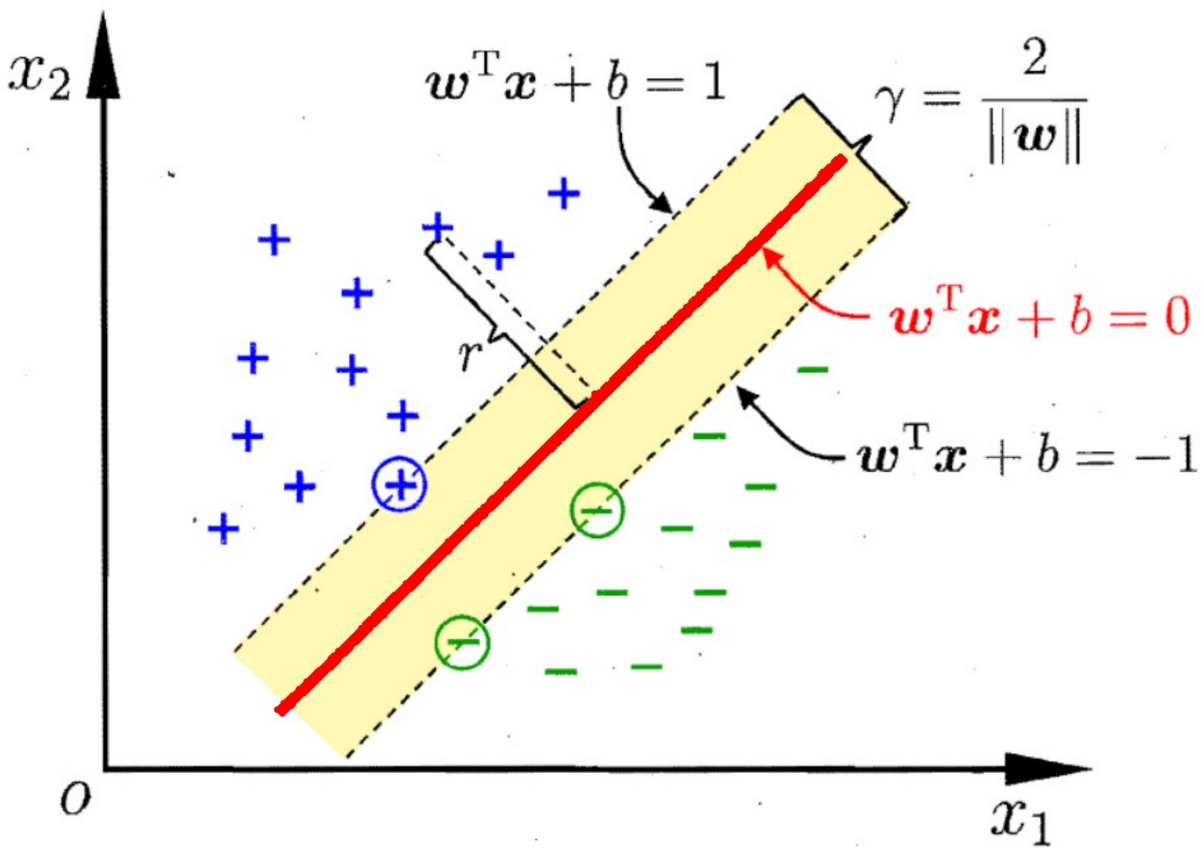
- 哪些数据使得上式取等号?

  距离超平面最近的训练样本点

- 这些点被称为 **支持向量**

- 两类支持向量之间的间隔：

$$\rho = 2 / \|\boldsymbol{w}\|$$



支持向量机

- 线性可分支持向量机最优化问题

$$\max_{\boldsymbol{w},b} \quad \frac{2}{||\boldsymbol{w}||}$$

$$\text{s.t.} \quad y_i(\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}_i + b) \geqslant 1, \quad i = 1, 2, \ldots, m.$$

- 等价于

$$\min_{\boldsymbol{w},b} \quad \frac{1}{2} \|\boldsymbol{w}\|^2$$

$$\text{s.t.} \quad y_i(\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}_i + b) \geqslant 1, \quad i = 1, 2, \ldots, m.$$

支持向量机的基本型

求解支持向量机

使用拉格朗日乘子法，得到对偶问题：

- 对约束项添加拉格朗日乘子 $\alpha_i \geq 0$，得到该问题的拉格朗日函数

$$L(\boldsymbol{w}, b, \boldsymbol{\alpha}) = \frac{1}{2} \|\boldsymbol{w}\|^2 + \sum_{i=1}^{m} \alpha_i \left(1 - y_i(\boldsymbol{w}^{\mathrm{T}} \boldsymbol{x}_i + b)\right)$$

- 令 $L(\boldsymbol{w}, b, \boldsymbol{\alpha})$ 对 $\boldsymbol{w}$ 和 b 的偏导为零

$$\boldsymbol{w} = \sum_{i=1}^{m} \alpha_i y_i \boldsymbol{x}_i \ , \qquad 0 = \sum_{i=1}^{m} \alpha_i y_i$$

- 上式代入 $L(\boldsymbol{w}, b, \boldsymbol{\alpha})$，即可消去 $\boldsymbol{w}$ 和 b

求解支持向量机

- 得到对偶问题

$$\max_{\boldsymbol{\alpha}} \quad \sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i=1}^{m} \sum_{j=1}^{m} \alpha_i \alpha_j y_i y_j \boldsymbol{x}_i^{\mathrm{T}} \boldsymbol{x}_j$$

$$\text{s.t.} \quad \sum_{i=1}^{m} \alpha_i y_i = 0 \,,$$

$$\alpha_i \geqslant 0 \,, \quad i = 1, 2, \ldots, m \,.$$

- 求解出上式的 α 后，可代入得到 $f(\boldsymbol{x}) = \sum_{i=1}^{m} \alpha_i y_i \boldsymbol{x}_i^{\mathrm{T}} \boldsymbol{x} + b$

求解支持向量机

- 现在有 $f(\boldsymbol{x}) = \sum_{i=1}^{m} \alpha_i y_i \boldsymbol{x}_i^{\mathrm{T}} \boldsymbol{x} + b$

- 原问题有不等式约束，因此上述过程需满足KKT条件

$$\begin{cases} \alpha_i \geqslant 0 \\ y_i f(\boldsymbol{x}_i) - 1 \geqslant 0 \\ \alpha_i \left(y_i f(\boldsymbol{x}_i) - 1\right) = 0 \end{cases}$$

求解支持向量机

- 现在有 $f(\boldsymbol{x}) = \sum_{i=1}^{m} \alpha_i y_i \boldsymbol{x}_i^{\mathrm{T}} \boldsymbol{x} + b$

- 原问题有不等式约束，因此上述过程需满足KKT条件

$$\begin{cases} \alpha_i \geqslant 0 \\ y_i f(\boldsymbol{x}_i) - 1 \geqslant 0 \\ \alpha_i \left(y_i f(\boldsymbol{x}_i) - 1\right) = 0 \end{cases}$$

- 对于任意的样本 $(\boldsymbol{x}_i, y_i)$，总有 $\alpha_{\mathrm{i}} = 0$ 或 $y_i f(\boldsymbol{x}_i) = 1$
- 若 $\alpha_{\mathrm{i}} = 0$：样本不在 $f(\boldsymbol{x})$ 中出现
- 若 $\alpha_{\mathrm{i}} > 0$：$y_i f(\boldsymbol{x}_i) = 1$

求解支持向量机

- 现在有 $f(\boldsymbol{x}) = \sum_{i=1}^{m} \alpha_i y_i \boldsymbol{x}_i^{\mathrm{T}} \boldsymbol{x} + b$

- 原问题有不等式约束，因此上述过程需满足KKT条件

$$\begin{cases} \alpha_i \geqslant 0 \\ y_i f(\boldsymbol{x}_i) - 1 \geqslant 0 \\ \alpha_i \left(y_i f(\boldsymbol{x}_i) - 1\right) = 0 \end{cases}$$

- 对于任意的样本 $(\boldsymbol{x}_i, y_i)$，总有 $\alpha_{\mathrm{i}} = 0$ 或 $y_i f(\boldsymbol{x}_i) = 1$
- 若 $\alpha_{\mathrm{i}} = 0$：样本不在 $f(\boldsymbol{x})$ 中出现
- 若 $\alpha_{\mathrm{i}} > 0$：$y_i f(\boldsymbol{x}_i) = 1$

求解支持向量机

- 现在有 $f(\boldsymbol{x}) = \sum_{i=1}^{m} \alpha_i y_i \boldsymbol{x}_i^{\mathrm{T}} \boldsymbol{x} + b$

- 原问题有不等式约束，因此上述过程需满足KKT条件

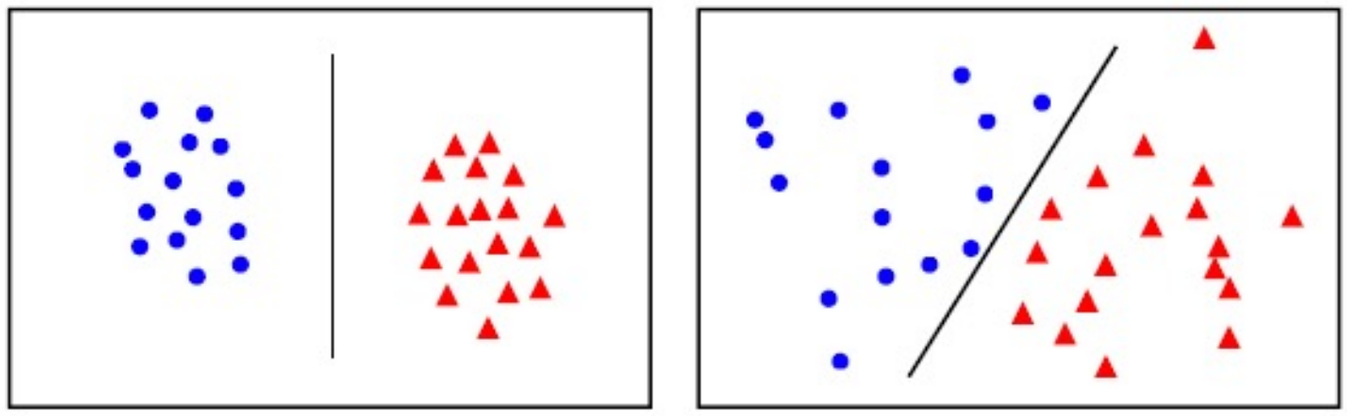
$$\begin{cases} \alpha_i \geqslant 0 \\ y_i f(\boldsymbol{x}_i) - 1 \geqslant 0 \\ \alpha_i \left(y_i f(\boldsymbol{x}_i) - 1\right) = 0 \end{cases}$$

- 对于任意的样本 $(\boldsymbol{x}_i, y_i)$，总有 $\alpha_{\mathrm{i}} = 0$ 或 $y_i f(\boldsymbol{x}_i) = 1$
- 若 $\alpha_{\mathrm{i}} = 0$：样本不在 $f(\boldsymbol{x})$ 中出现
- 若 $\alpha_{\mathrm{i}} > 0$：$y_i f(\boldsymbol{x}_i) = 1$ → **样本为支持向量**

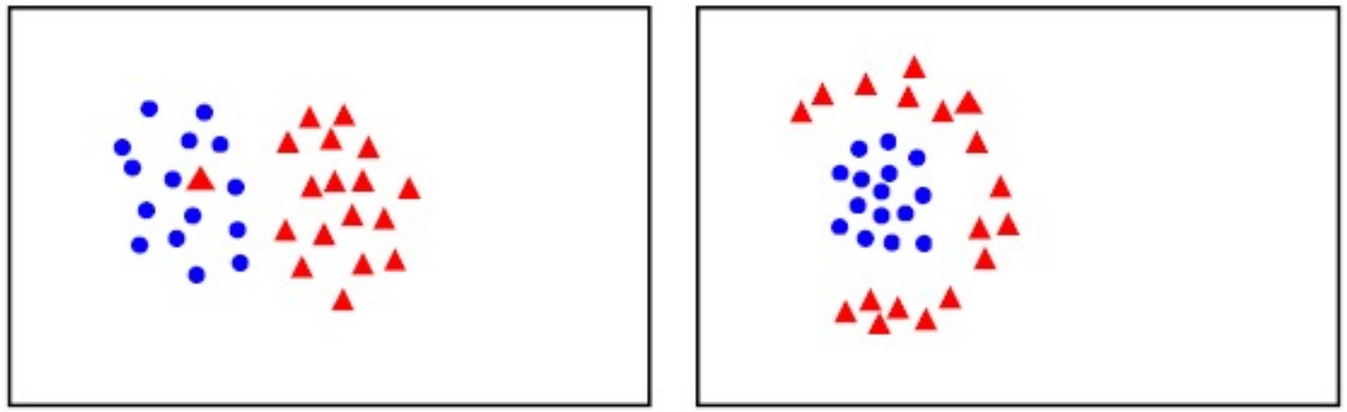
w 实际上由距离超平面最近的点决定，最终模型仅与支持向量有关

支持向量机 —— 核函数

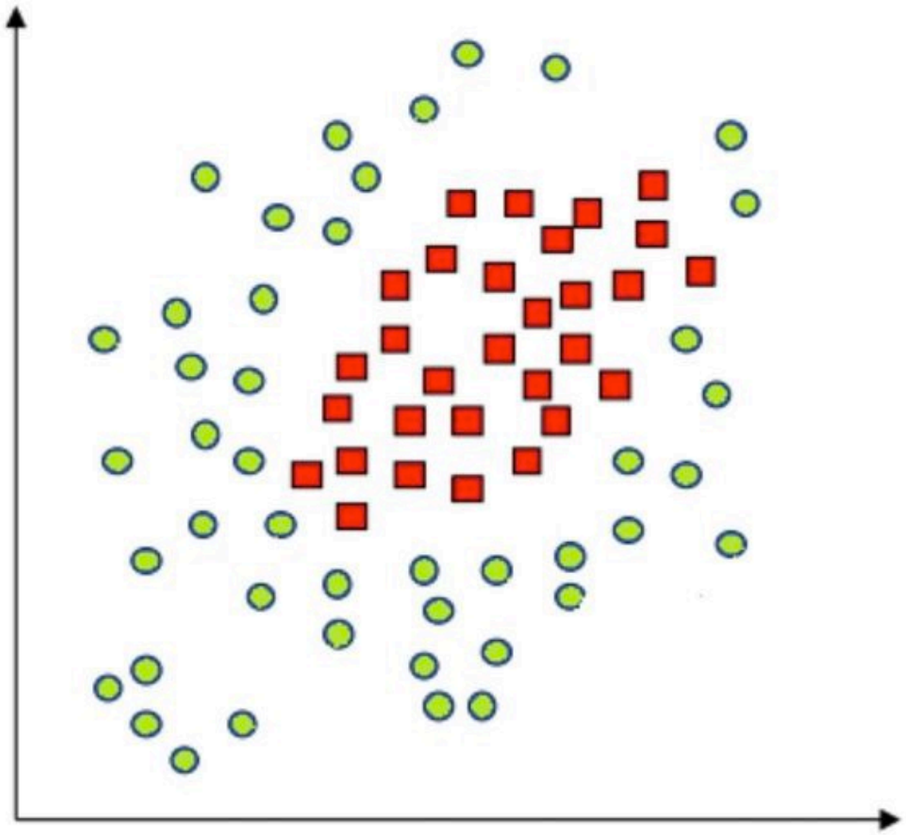
线性可分



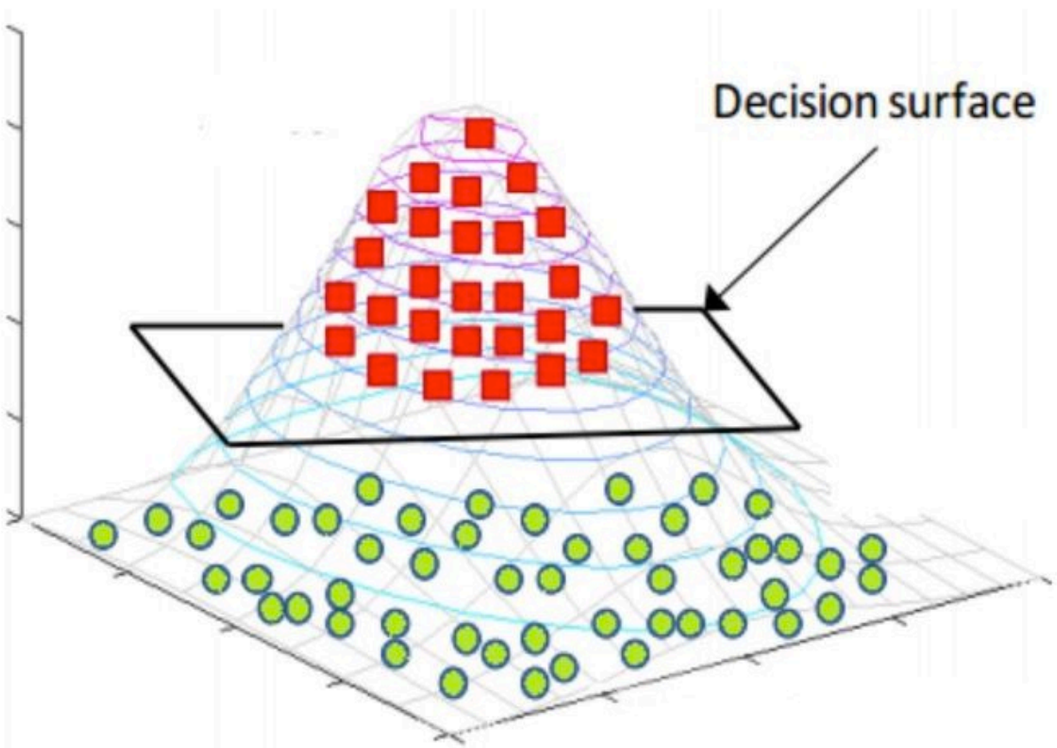
非
线性可分



支持向量机 —— 核函数



kernel



支持向量机 —— 核函数

- 模型变为：$f(\boldsymbol{x}) = \boldsymbol{w}^{\mathrm{T}}\phi(\boldsymbol{x}) + b$

- 类似地，求解得到：$f(\boldsymbol{x}) = \boldsymbol{w}^{\mathrm{T}}\phi(\boldsymbol{x}) + b$

$$= \sum_{i=1}^{m} \alpha_i y_i \phi(\boldsymbol{x}_i)^{\mathrm{T}} \phi(\boldsymbol{x}) + b$$

$$= \sum_{i=1}^{m} \alpha_i y_i \underline{\kappa(\boldsymbol{x}, \boldsymbol{x}_i)} + b \ .$$

核函数

- 即使不知道 Φ 的形式，也可以定义核函数

支持向量机 —— 核函数

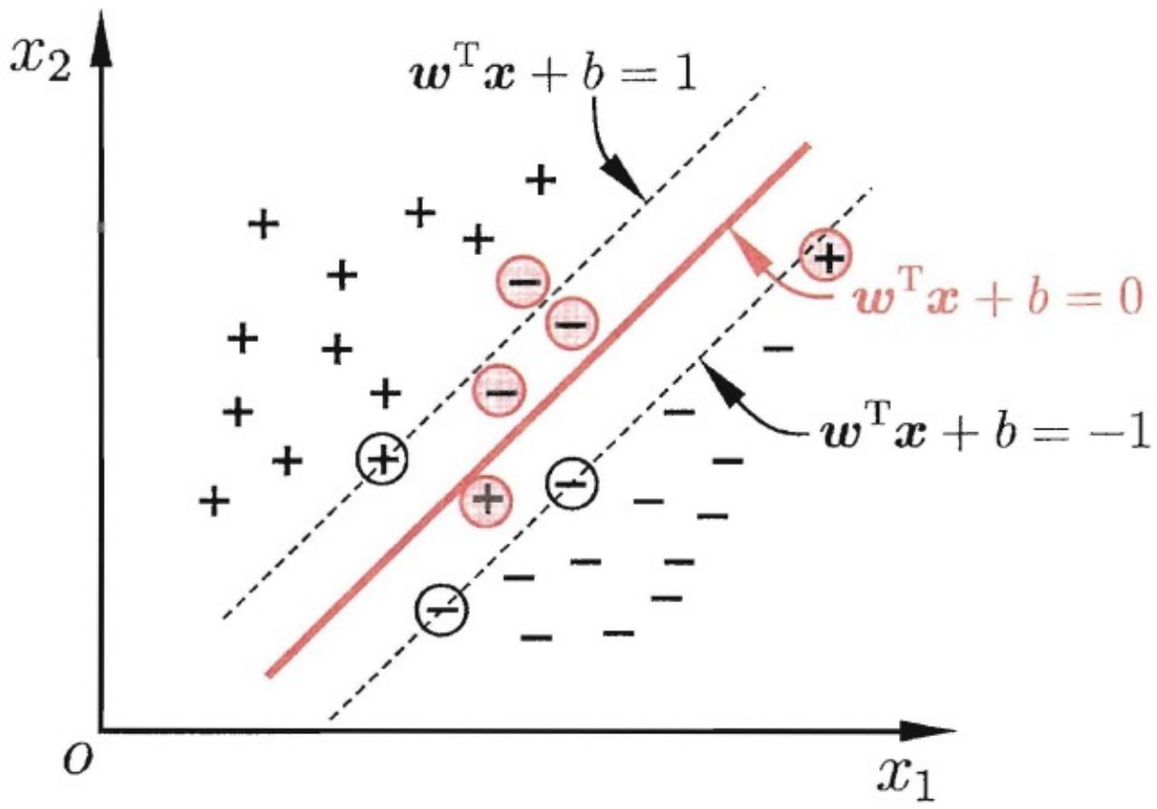
- 常用核函数

名称	表达式	参数
线性核	$\kappa(\boldsymbol{x}_i, \boldsymbol{x}_j) = \boldsymbol{x}_i^{\mathrm{T}} \boldsymbol{x}_j$	
多项式核	$\kappa(\boldsymbol{x}_i, \boldsymbol{x}_j) = (\boldsymbol{x}_i^{\mathrm{T}} \boldsymbol{x}_j)^d$	$d \geqslant 1$ 为多项式的次数
高斯核	$\kappa(\boldsymbol{x}_i, \boldsymbol{x}_j) = \exp\left(-\frac{\|\boldsymbol{x}_i - \boldsymbol{x}_j\|^2}{2\sigma^2}\right)$	$\sigma > 0$ 为高斯核的带宽(width)
拉普拉斯核	$\kappa(\boldsymbol{x}_i, \boldsymbol{x}_j) = \exp\left(-\frac{\|\boldsymbol{x}_i - \boldsymbol{x}_j\|}{\sigma}\right)$	$\sigma > 0$
Sigmoid 核	$\kappa(\boldsymbol{x}_i, \boldsymbol{x}_j) = \tanh(\beta \boldsymbol{x}_i^{\mathrm{T}} \boldsymbol{x}_j + \theta)$	tanh 为双曲正切函数, $\beta > 0, \theta < 0$

- 此外，还可通过函数组合得到

软间隔支持向量机

- 希望允许部分点更靠近分类超平面，甚至被错误分类



- 所有样本都必须划分正确：硬间隔
- 允许某些样本不满足约束：软间隔

软间隔支持向量机

- 软间隔支持向量机优化目标：

$$\min_{\boldsymbol{w},b} \ \frac{1}{2}\|\boldsymbol{w}\|^2 + C\sum_{i=1}^{m}\ell_{0/1}\left(y_i\left(\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}_i + b\right) - 1\right)$$

其中 $\ell_{0/1}(z) = \begin{cases} 1, & \text{if } z < 0; \\ 0, & \text{otherwise.} \end{cases}$

- 参数 $C > 0$ 控制了惩罚力度
  - 更小的 C：注重泛化性能
  - 更大的 C：注重拟合性能 （$C = +\infty$ 时变为硬间隔）

软间隔支持向量机

- 软间隔支持向量机优化目标：

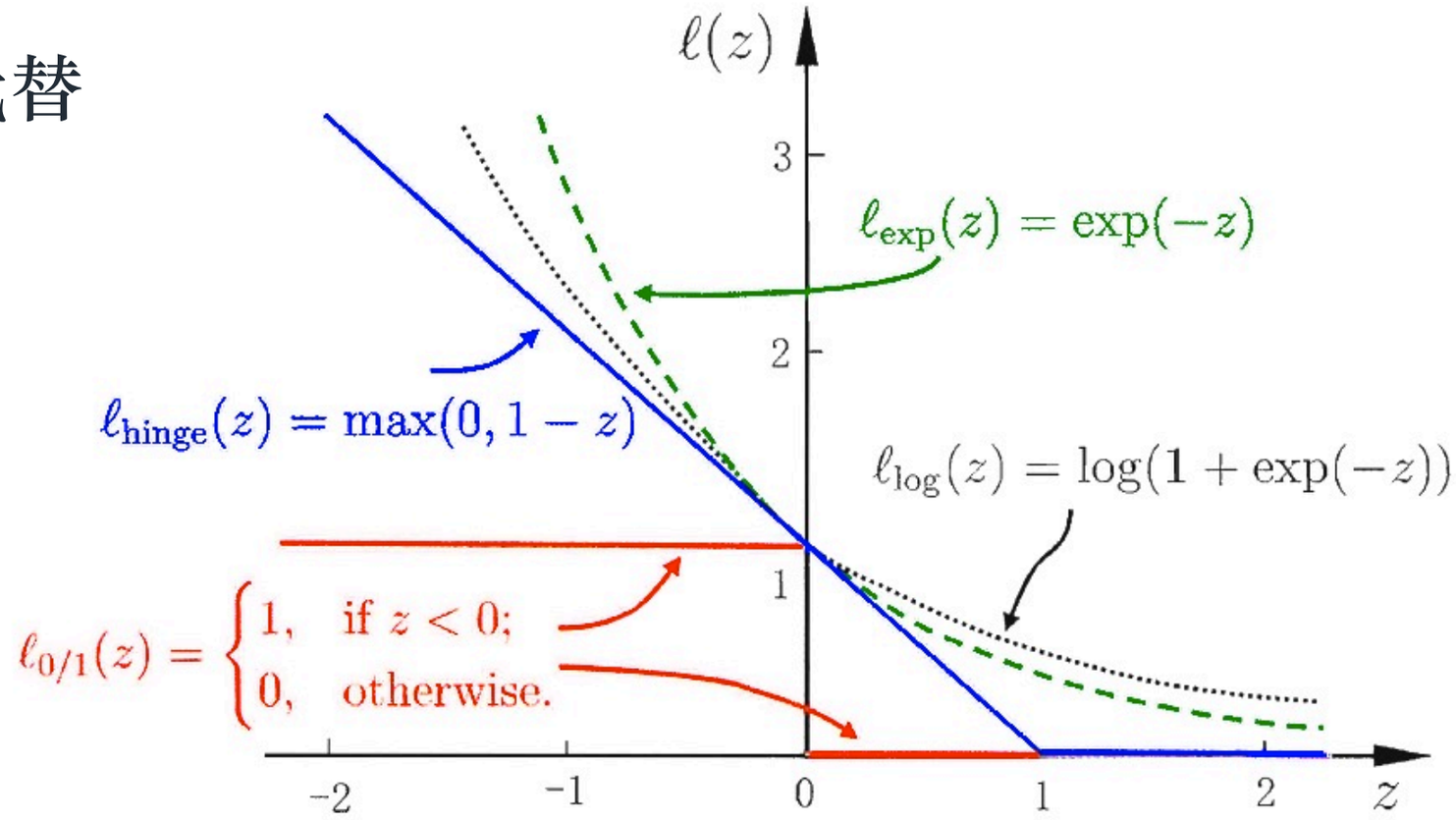
$$\min_{\boldsymbol{w},b} \ \frac{1}{2}\|\boldsymbol{w}\|^2 + C\sum_{i=1}^{m}\ell_{0/1}\left(y_i\left(\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}_i + b\right) - 1\right)$$

其中 $\ell_{0/1}(z) = \begin{cases} 1, & \text{if } z < 0; \\ 0, & \text{otherwise.} \end{cases}$

- 参数 $C > 0$ 控制了惩罚力度
- 优化问题实际上包含两个 tradeoff 部分
  - 最小化 $\boldsymbol{w}$ 的模长
  - 最小化 **误分类点** 的分类误差

软间隔支持向量机

- $\ell_{0/1}(z)$ 数学性质并不好，非凸、非连续
- 实际常用其他函数作为代替



hinge 损失: $\ell_{hinge}(z) = \max(0, 1-z)$;

指数损失(exponential loss): $\ell_{exp}(z) = \exp(-z)$;

对率损失(logistic loss): $\ell_{log}(z) = \log(1+\exp(-z))$.

软间隔支持向量机

- 若采用 hinge 损失，则软间隔支持向量机优化目标变为

$$\min_{\boldsymbol{w},b} \ \frac{1}{2}\|\boldsymbol{w}\|^2 + C\sum_{i=1}^{m}\max\left(0, 1 - y_i\left(\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}_i + b\right)\right)$$

- 引入松弛变量 $\xi_i \geq 0$，对函数进行重写

$$\begin{aligned} \min_{\boldsymbol{w},b,\xi_i} \quad & \frac{1}{2}\|\boldsymbol{w}\|^2 + C\sum_{i=1}^{m}\xi_i \\ \text{s.t.} \quad & y_i(\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}_i + b) \geqslant 1 - \xi_i \\ & \xi_i \geqslant 0 \ , \ i = 1, 2, \ldots, m \end{aligned}$$

常用的
软间隔支持向量机

支持向量机小结

- 支持向量机是一个最大间隔分类超平面
  - 因此支持向量机有更好的预测能力
- 支持向量机的学习可以表达为一个二次优化问题
  - 该问题可以使用拉格朗日乘子法有效求解
- 使用核函数寻找其他高维特征空间
- 针对存在少量噪声的数据，设计软间隔支持向量机
  - 允许部分样本离分类超平面近一些甚至被误分类
  - 损失函数同时惩罚 w 的模长和误分类的程度

SVM 实践：

1. LIBSVM

https://www.csie.ntu.edu.tw/~cjlin/libsvm/

2. LIBLINEAR

https://www.csie.ntu.edu.tw/~cjlin/liblinear/

3. [推荐] scikit-learn

```
pip install scikit-learn
```

```
import sklearn
```

SVM 实践：线性SVM

SVM Inplementation in Scikit-Learn

```
sklearn.svm.LinearSVC(
    penalty='l2', loss='hinge', dual=True,
    tol=0.0001, C=1.0, multi_class='ovr',
    fit_intercept=True, intercept_scaling=1,
    class_weight=None, verbose=0, random_state=None,
    max_iter=1000
)
```

$$\min_{\boldsymbol{w},b} \quad \|\boldsymbol{w}\|^2$$

$$\text{s.t.} \quad y_i(\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}_i + b) \geqslant 1, \quad i = 1, 2, \ldots, m.$$

SVM 实践：线性SVM

SVM Inplementation in Scikit-Learn

```
sklearn.svm.LinearSVC(
    penalty='l2', loss='hinge', dual=True,
    tol=0.0001, C=1.0, multi_class='ovr',
    fit_intercept=True, intercept_scaling=1,
    class_weight=None, verbose=0, random_state=None,
    max_iter=1000
)
```

Dual:拉格朗日乘子法求解对偶问题

tol: 迭代求解算法结束允许的误差

SVM 实践：线性SVM

SVM Inplementation in Scikit-Learn

```
sklearn.svm.LinearSVC(
        penalty='l2', loss='hinge', dual=True,
        tol=0.0001, C=1.0, multi_class='ovr',
        fit_intercept=True, intercept_scaling=1,
       class_weight=None, verbose=0, random_state=None,
        max_iter=1000
)
```

$$\min_{\boldsymbol{w},b,\xi_i} \quad \frac{1}{2}\|\boldsymbol{w}\|^2 + C\sum_{i=1}^{m}\xi_i$$

$$\text{s.t.} \quad y_i(\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}_i + b) \geqslant 1 - \xi_i$$

$$\xi_i \geqslant 0 \ , \ i = 1, 2, \ldots, m$$

SVM 实践：线性SVM

SVM Inplementation in Scikit-Learn

```
sklearn.svm.LinearSVC(
        penalty='l2', loss='hinge', dual=True,
        tol=0.0001, C=1.0, multi_class='ovr',
        fit_intercept=True, intercept_scaling=1,
       class_weight=None, verbose=0, random_state=None,
        max_iter=1000
)
```

SVM 解决二分类问题，多分类问题需要被表达成二分类问题

'ovr' 即 one Vs rest，训练 n_classes 个一对多分类器

'crammer_singer'优化所有类别的联合目标（通常代价太高）

SVM 实践：线性SVM

SVM Inplementation in Scikit-Learn

```
sklearn.svm.LinearSVC(
        penalty='l2', loss='hinge', dual=True,
        tol=0.0001, C=1.0, multi_class='ovr',
        fit_intercept=True, intercept_scaling=1,
        class_weight=None, verbose=0, random_state=None,
        max_iter=1000
)
```

class_weight：对于不均匀的类别分布（例如某种类别比例特别高），可以对每种类别指定不一样的C来平衡不均匀性

verbose：是否输出中间结果到屏幕

max_iter：最大迭代次数

SVM 实践：非线性 SVM

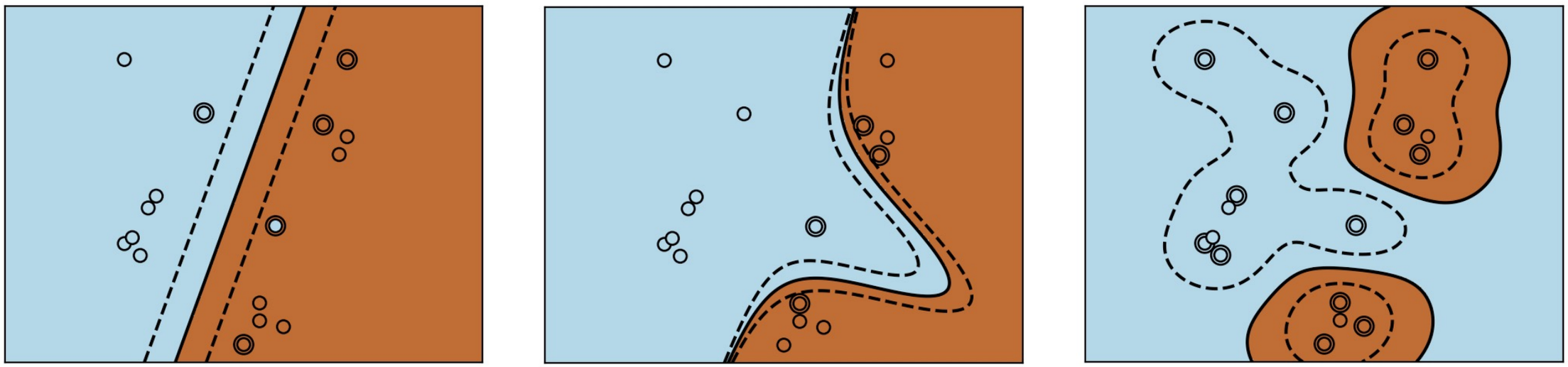
Scikit-Learn 中的 非线性SVM

```
sklearn.svm.SVC(
        C=1.0, kernel='rbf', degree=3,
        gamma='auto_deprecated', coef0=0.0,
        shrinking=True, probability=False, tol=0.001,
        cache_size=200, class_weight=None,
        verbose=False, max_iter=-1,
        decision_function_shape='ovr',
        random_state=None
)
```

对样本空间做变换，使得原来可能线性不可分的数据也能够被SVM成功分类。一般任务使用线性 SVM 即可，速度也比较快。

SVM 实践：非线性 SVM

- kernel 可视化样例



'linear' 'poly' 'rbf'

- LinearSVC ≠ SVC(kernel='linear')

SVM 实践

```python
# 合成一份数据集
from sklearn.datasets import make_classification
X, y = make_classification(n_features=4, random_state=0)

# SVM训练
from sklearn.svm import LinearSVC
clf = LinearSVC()
clf.fit(X, y)
```

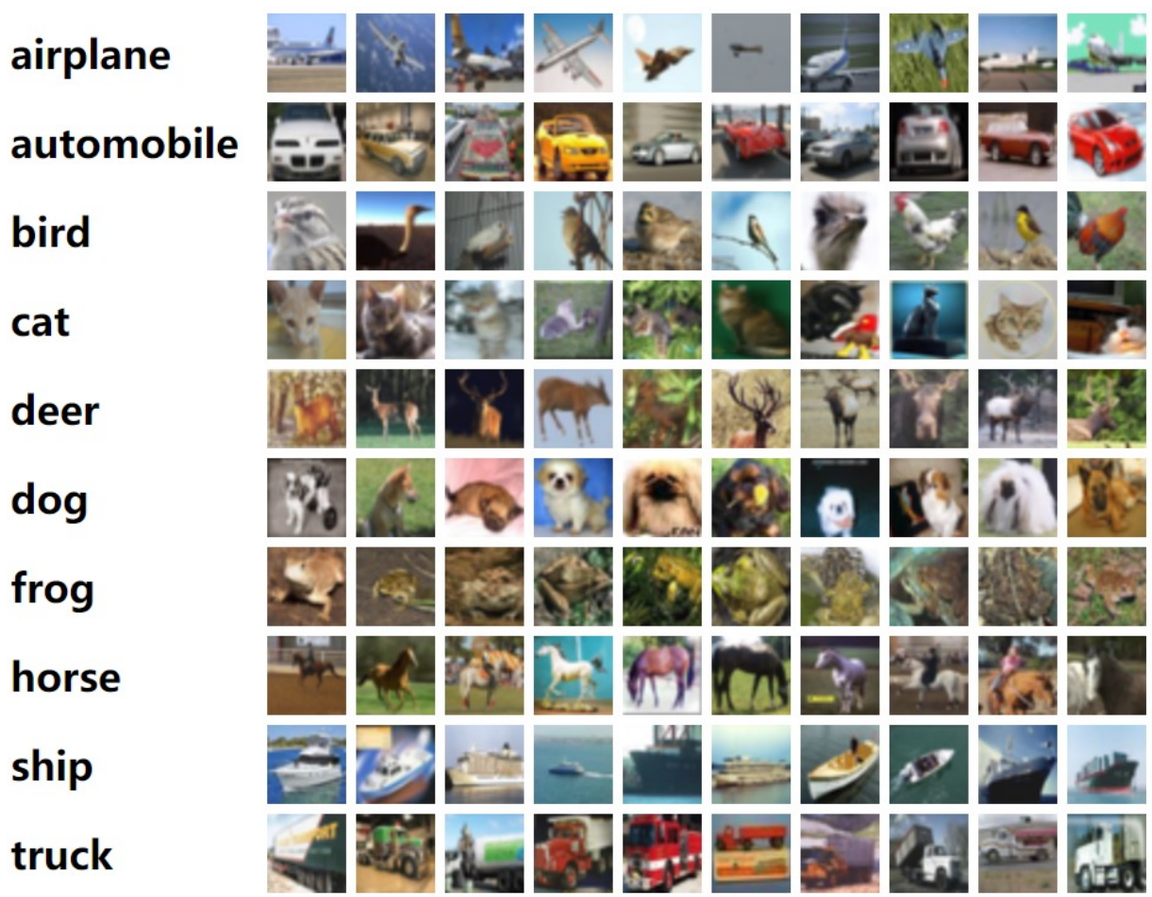
SVM 实践

```
# SVM测试1: 预测新数据
new_X = [[ 0, 0,  0, 0], [-1, 0, -1, 0]]
pred_y = clf.predict(new_X)
# [1 0]
```

```
# SVM测试2: 测试集上的平均准确率
test_X, test_y = make_classification(n_features=4,
                                     random_state=0)
score = clf.score(test_X, test_y)
# 0.93
```

SVM 实践进阶：图像分类

- Cifar10
  - Tiny image recognition datasets
  - 60,000 32×32 color images in 10 different classes



Step 1: 数据准备

HOG-SVM.py — UNREGISTERED

HOG-SVM.py

```python
import numpy as np
import torch
import torchvision
import torchvision.transforms as transforms

# 数据集准备
transform = transforms.Compose(
    [transforms.ToTensor(),
     transforms.Normalize((0.5, 0.5, 0.5), (0.5, 0.5, 0.5))])

trainset = torchvision.datasets.CIFAR10(root='./data', train=True,
                                        download=True, transform=transform)
testset = torchvision.datasets.CIFAR10(root='./data', train=False,
                                       download=True, transform=transform)

train_x = trainset.data
train_y = np.array(trainset.targets)
test_x = testset.data
test_y = np.array(testset.targets)

# type(train_x): <class 'numpy.ndarray'>
# type(train_y): <class 'numpy.ndarray'>
# train_x.shape: (50000, 32, 32, 3)
# train_y.shape: (50000,)

```

Line 52, Column 1 — Spaces: 4 — Python

Step 2: 特征提取

- 提取 HOG 特征
  - 我们选择使用 Scikit-Image 包
    - 大部分按照Python风格设计
    - 核心代码用C编写以提高效率

HOG-SVM.py — UNREGISTERED

HOG-SVM.py

```
29  # 提取HOG特征
30  from skimage.feature import hog
31
32  def get_hog(data):
33      X = []
34      for img in data:
35          fd = hog(img, pixels_per_cell=(4,4), cells_per_block=(2,2),
36                   multichannel=True, feature_vector=True)
37          X.append(fd)
38      return np.array(X)
39
```

Line 1, Column 1 Spaces: 4 Python

Step 3: 训练和测试分类器

HOG-SVM.py — UNREGISTERED

HOG-SVM.py

```
41
42  from sklearn.svm import LinearSVC
43  clf = LinearSVC()
44
45  # SVM训练
46  train_fx = get_hog(train_x)  # shape: (50000, 1764)
47  clf.fit(train_fx, train_y)
48
49  # SVM测试
50  test_fx = get_hog(test_x)
51  print("HOG+SVM:", clf.score(test_fx, test_y))
52
53  # HOG+SVM: 0.5442
54
55
```

Line 1, Column 1 Spaces: 4 Python

小结

图像识别任务的基本步骤

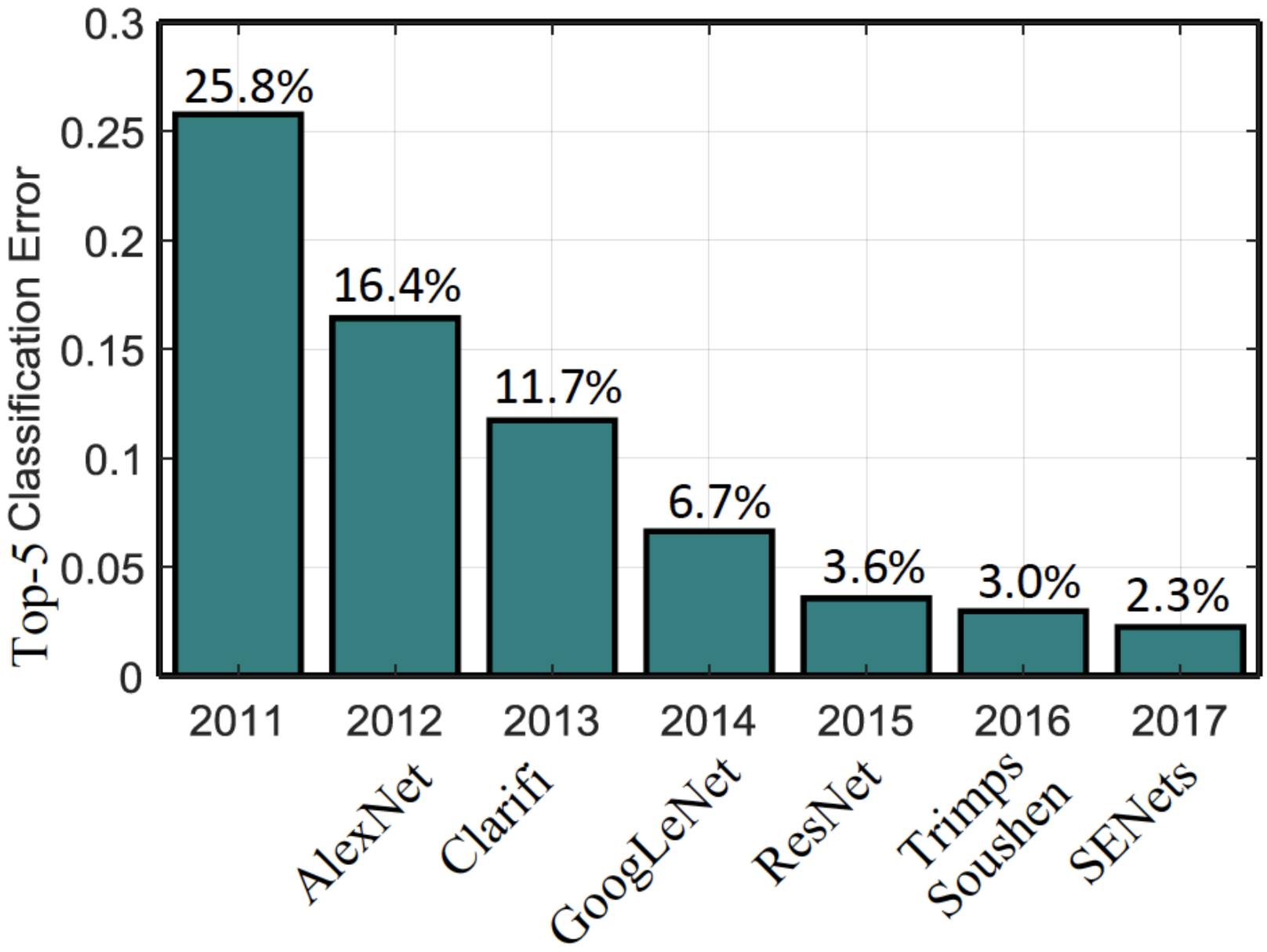
1. 准备数据集（可能包括增强和预处理）
2. 提取特征（可能需要对特征进行向量编码）
3. 选择、训练分类器对特征向量进行分类

神经网络图像分类方法

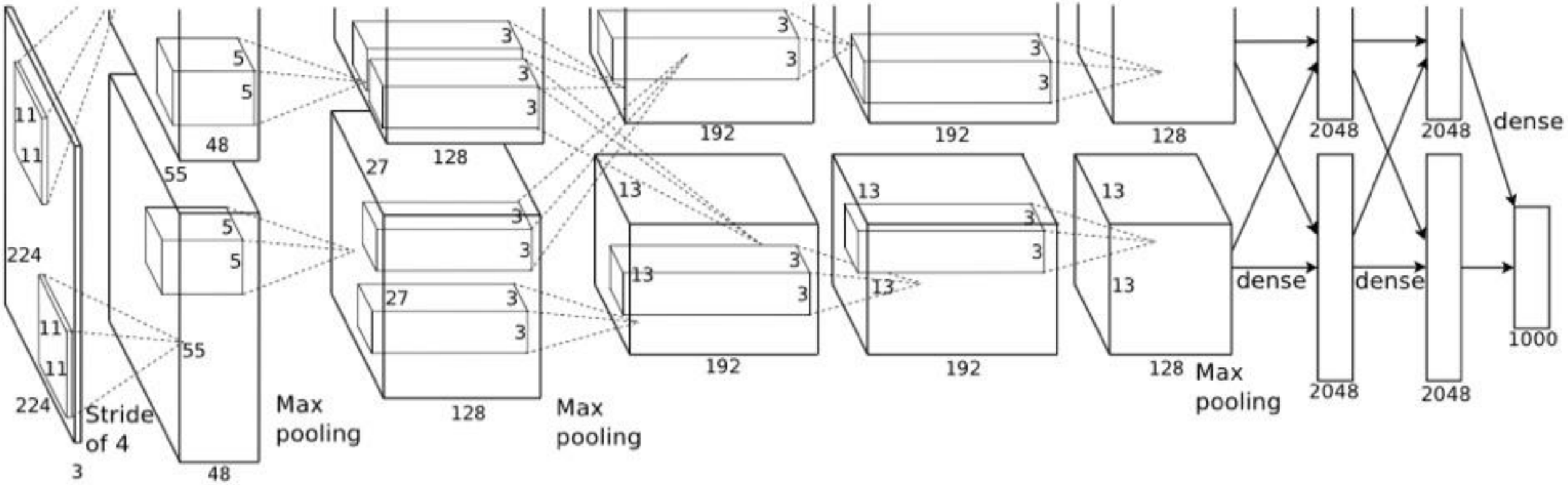
Deep-based Image Classification

Top Image Classification Competition Results

Top Image Classification Competition Results at ILSVRC



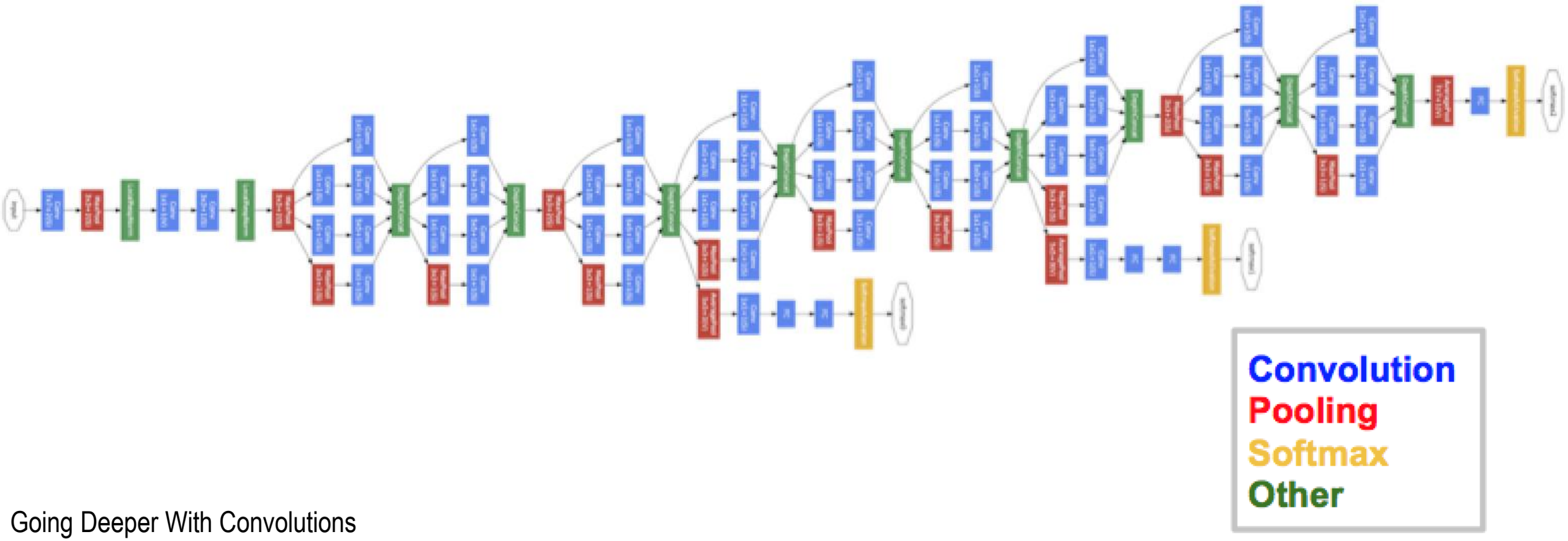
AlexNet



Krizhevsky, Alex, Sutskever, Ilya, and Hinton, Geoffrey E. Imagenet classification with deep convolutional neural networks. In Advances in neural information processing systems (NIPS), pp.1097–1105, 2012.

GoogleNet / Inception

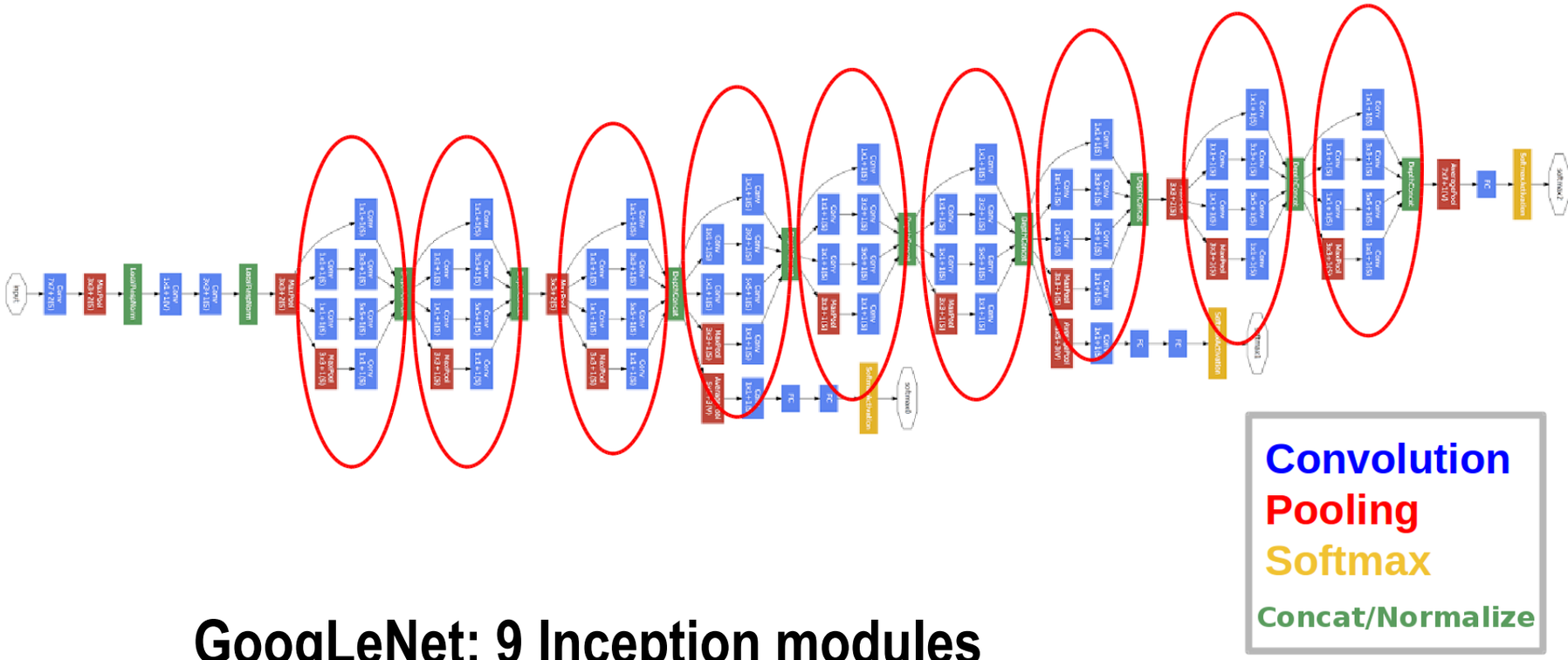
- GoogleNet architecture consists of a 22 layer deep CNN
- But reduced the number of parameters from 60 million (AlexNet) to 4 million



Going Deeper With Convolutions
Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, Andrew Rabinovich; The IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2015, pp. 1-9

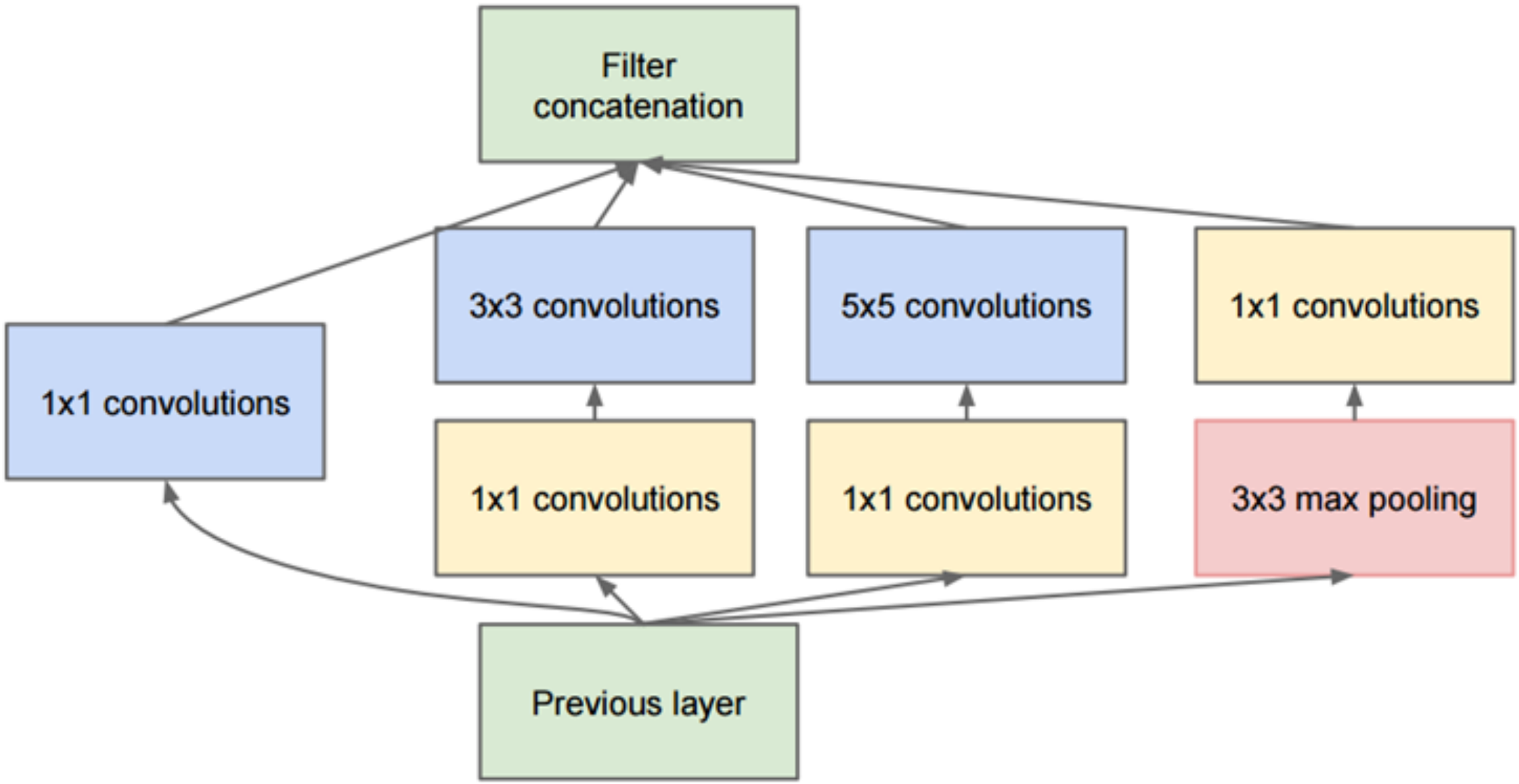
GoogleNet: CNN Micro-architecture

- With the trend of designing very deep CNNs,
- it becomes cumbersome to manually select filter dimensions for each layer
  - To add this, various higher level building blocks, or modules, have been proposed



GoogLeNet: 9 Inception modules

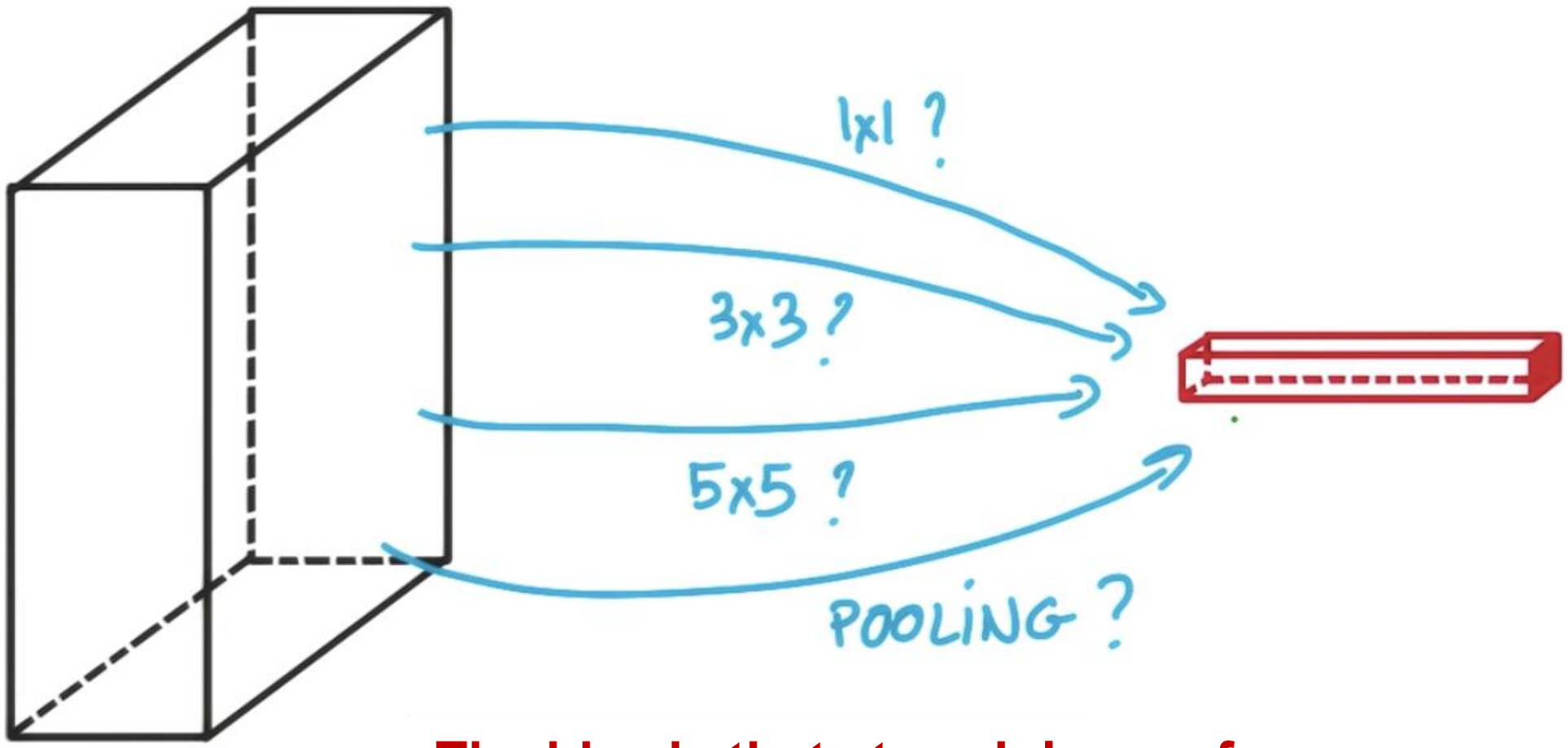
Inception Module



Full Inception Module

Intuitive Behind Inception Module

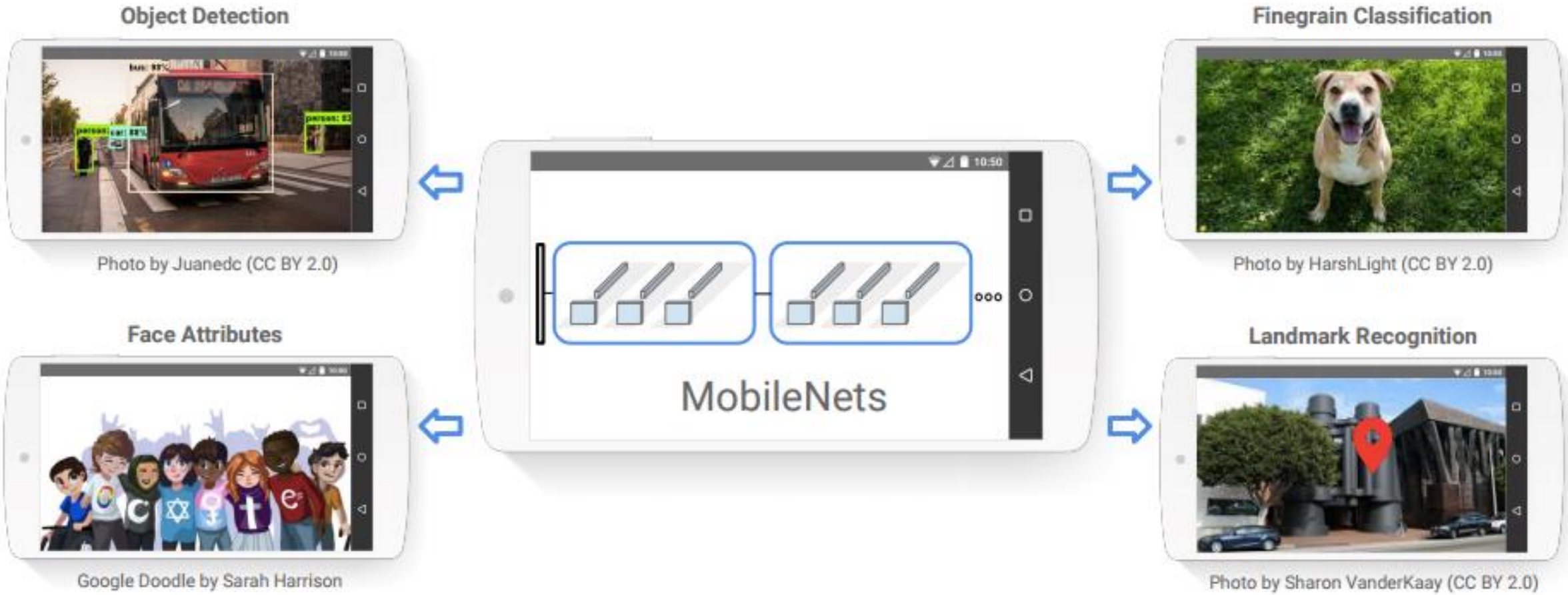
- In the lower layers, there are high correlations in image patches that are **local** and **near-local**



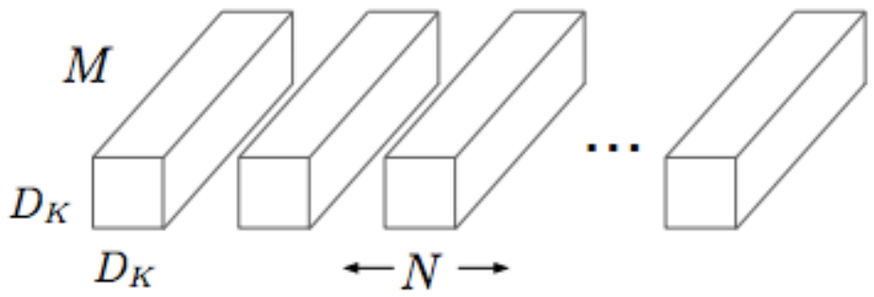
The idea is that at each layer of your convnets you can make a choice.

MobileNets

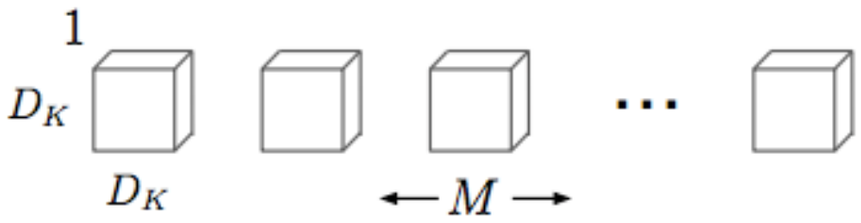
- Efficient Convolutional Neural Networks for Mobile Vision Applications



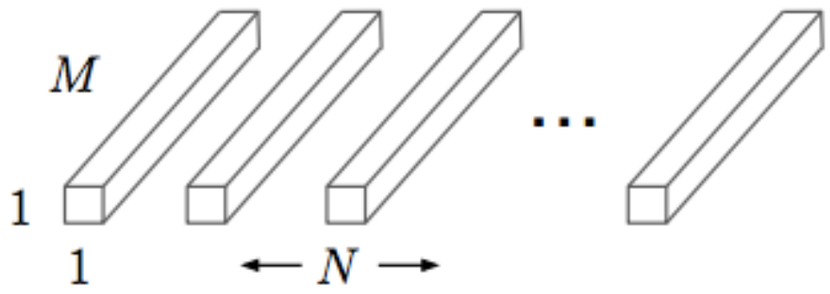
MobileNets



(a) Standard Convolution Filters



(b) Depthwise Convolutional Filters



(c) 1×1 Convolutional Filters called Pointwise Convolution in the context of Depthwise Separable Convolution

Figure 2. The standard convolutional filters in (a) are replaced by two layers: depthwise convolution in (b) and pointwise convolution in (c) to build a depthwise separable filter.

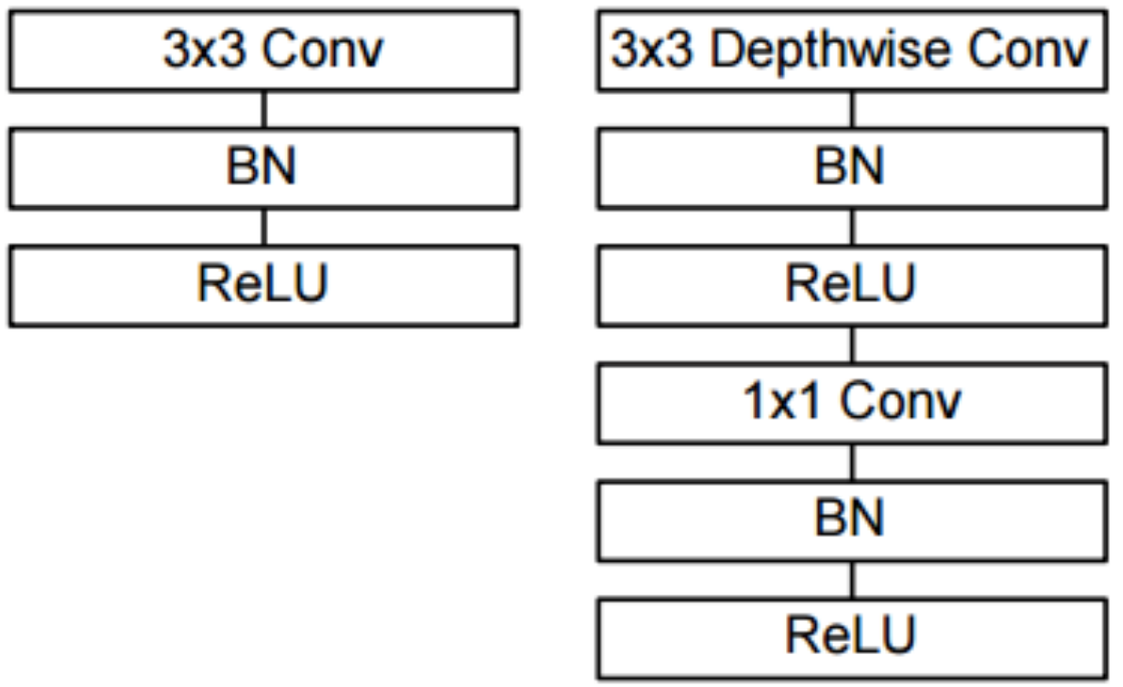


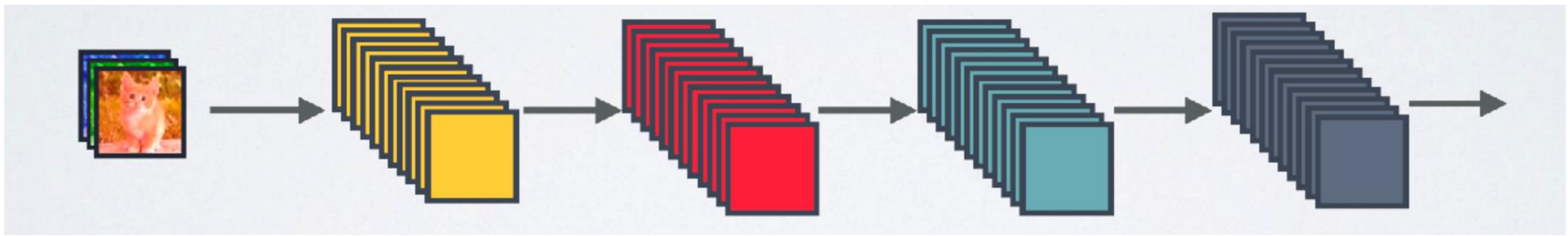
Figure 3. Left: Standard convolutional layer with batchnorm and ReLU. Right: Depthwise Separable convolutions with Depthwise and Pointwise layers followed by batchnorm and ReLU.

MobileNet Architecture

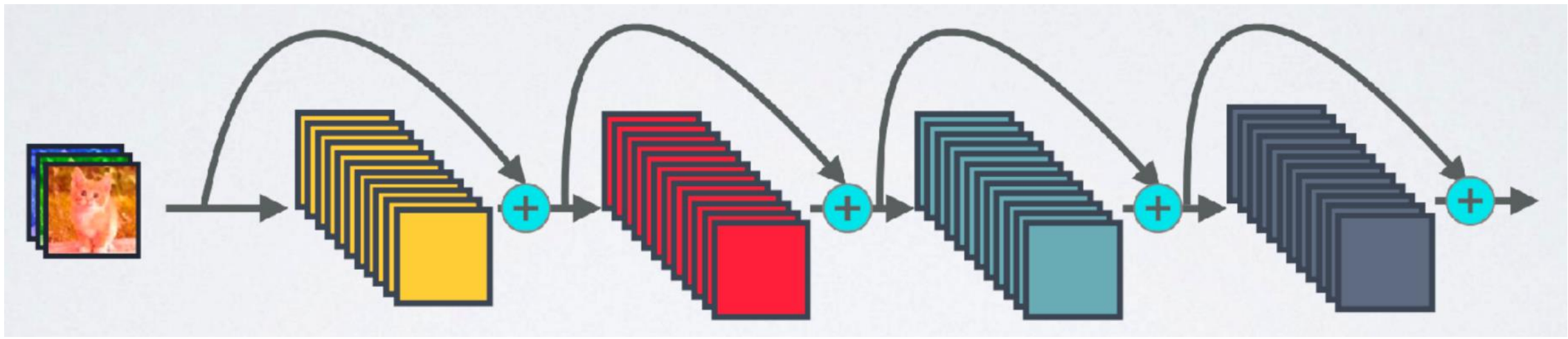
Type / Stride	Filter Shape	Input Size
Conv / s2	$3 \times 3 \times 3 \times 32$	$224 \times 224 \times 3$
Conv dw / s1	$3 \times 3 \times 32$ dw	$112 \times 112 \times 32$
Conv / s1	$1 \times 1 \times 32 \times 64$	$112 \times 112 \times 32$
Conv dw / s2	$3 \times 3 \times 64$ dw	$112 \times 112 \times 64$
Conv / s1	$1 \times 1 \times 64 \times 128$	$56 \times 56 \times 64$
Conv dw / s1	$3 \times 3 \times 128$ dw	$56 \times 56 \times 128$
Conv / s1	$1 \times 1 \times 128 \times 128$	$56 \times 56 \times 128$
Conv dw / s2	$3 \times 3 \times 128$ dw	$56 \times 56 \times 128$
Conv / s1	$1 \times 1 \times 128 \times 256$	$28 \times 28 \times 128$
Conv dw / s1	$3 \times 3 \times 256$ dw	$28 \times 28 \times 256$
Conv / s1	$1 \times 1 \times 256 \times 256$	$28 \times 28 \times 256$
Conv dw / s2	$3 \times 3 \times 256$ dw	$28 \times 28 \times 256$
Conv / s1	$1 \times 1 \times 256 \times 512$	$14 \times 14 \times 256$
$5\times$ Conv dw / s1	$3 \times 3 \times 512$ dw	$14 \times 14 \times 512$
$5\times$ Conv / s1	$1 \times 1 \times 512 \times 512$	$14 \times 14 \times 512$
Conv dw / s2	$3 \times 3 \times 512$ dw	$14 \times 14 \times 512$
Conv / s1	$1 \times 1 \times 512 \times 1024$	$7 \times 7 \times 512$
Conv dw / s2	$3 \times 3 \times 1024$ dw	$7 \times 7 \times 1024$
Conv / s1	$1 \times 1 \times 1024 \times 1024$	$7 \times 7 \times 1024$
Avg Pool / s1	Pool 7×7	$7 \times 7 \times 1024$
FC / s1	1024×1000	$1 \times 1 \times 1024$
Softmax / s1	Classifier	$1 \times 1 \times 1000$

ResNet

- Standard connectivity
  - Each layer receives input from previous layer generate feature for the next layer

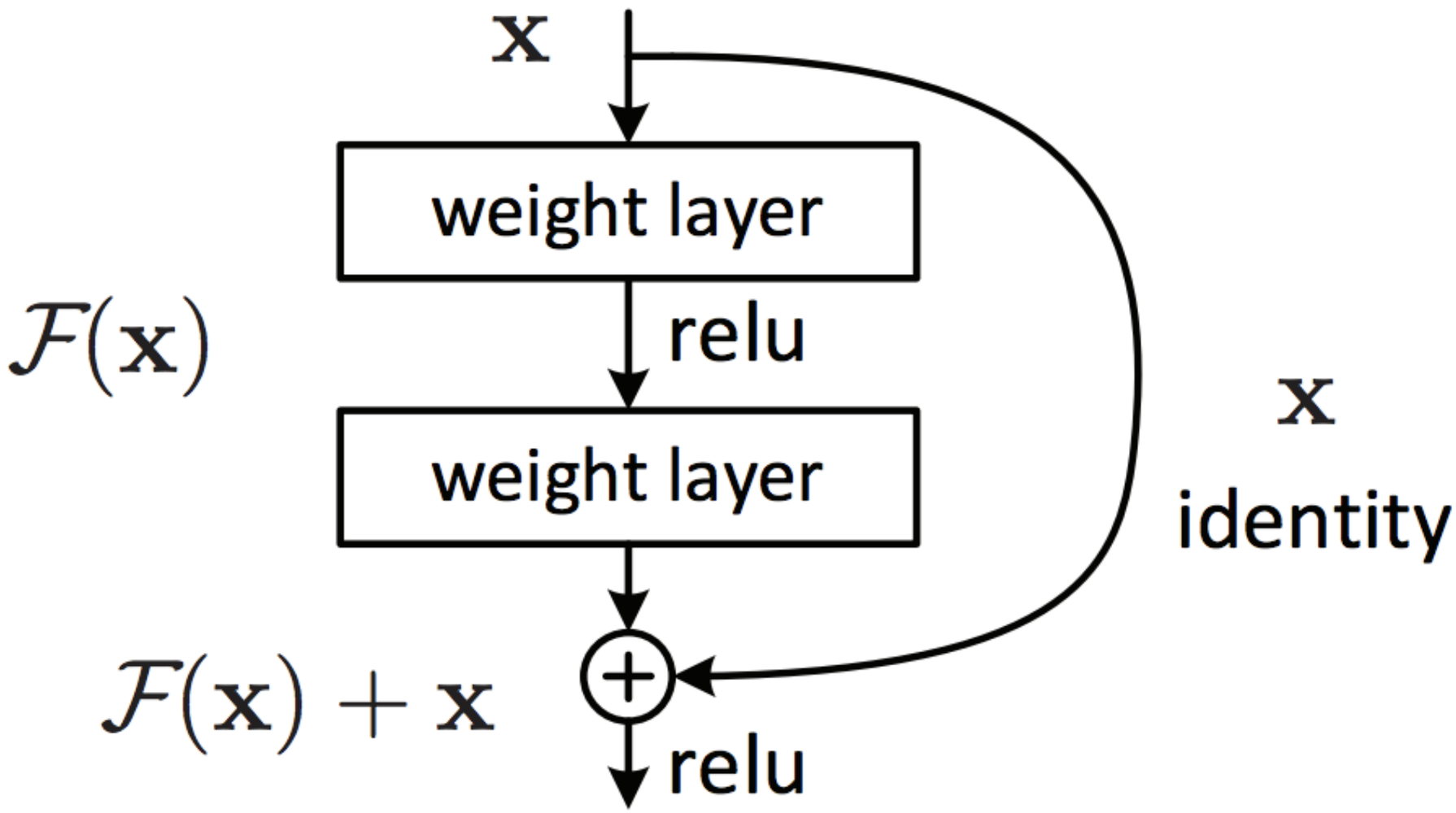


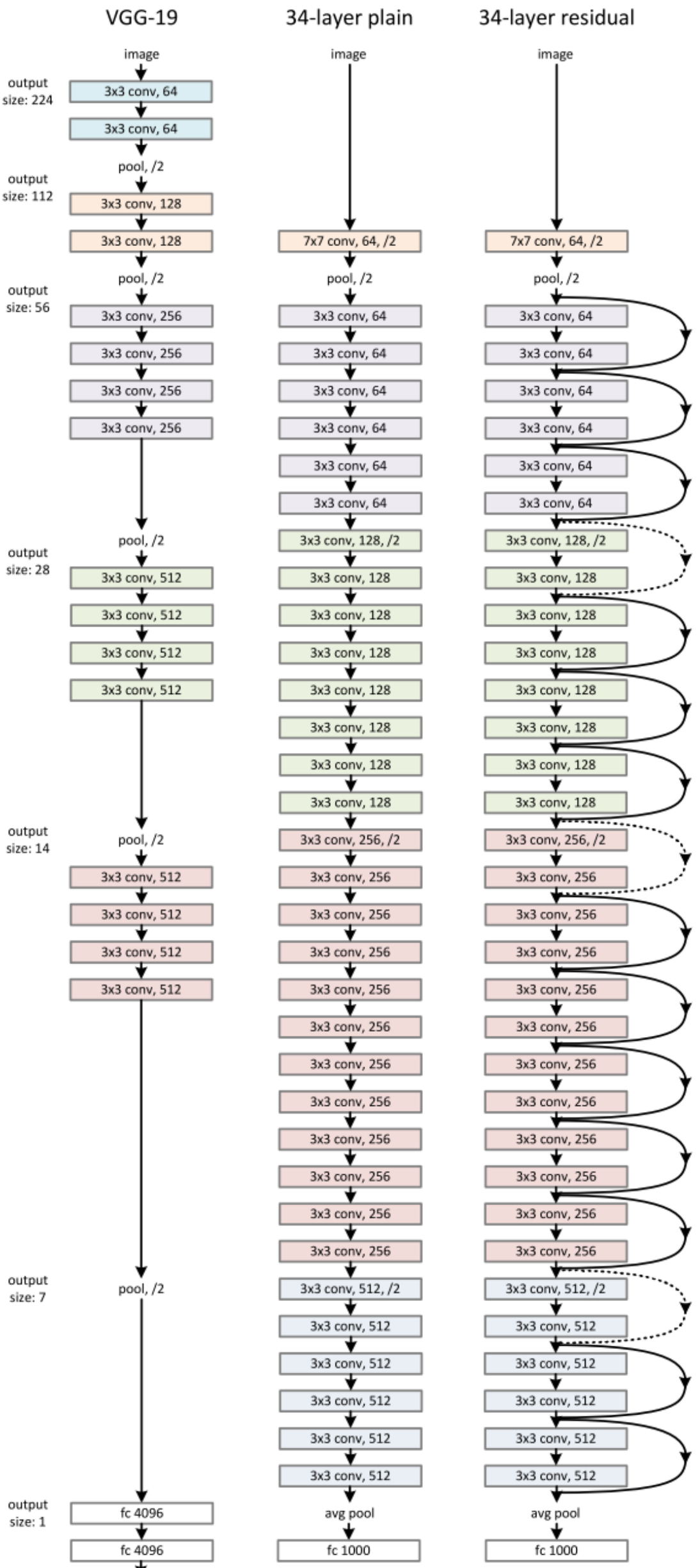
- ResNet connectivity
  - Identity mappings promote gradient propagation “skip connections”



ResNet

- Residual learning: a building block



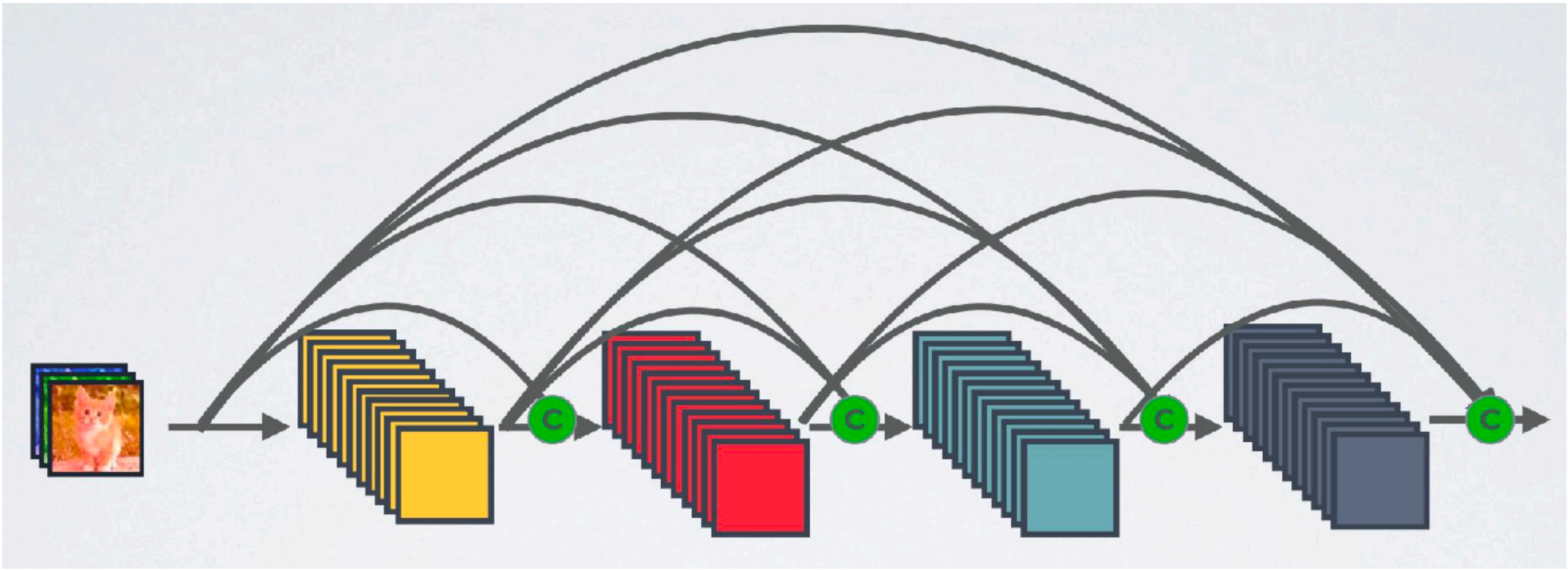


AIPKU

VGG-19 Vs. ResNet-34

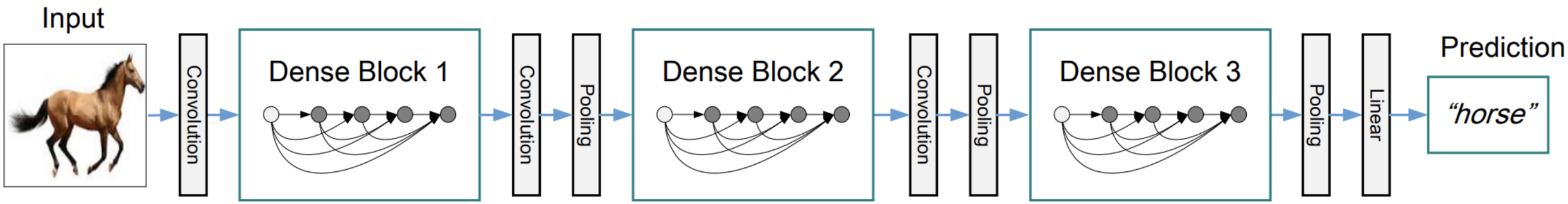
DenseNet (CVPR'17)

- Connect every two layers in the network
- Each layer receives signals from all its preceding layers



DenseNet Variants

- A deep DenseNet with three dense blocks



ResNet Vs. DenseNet