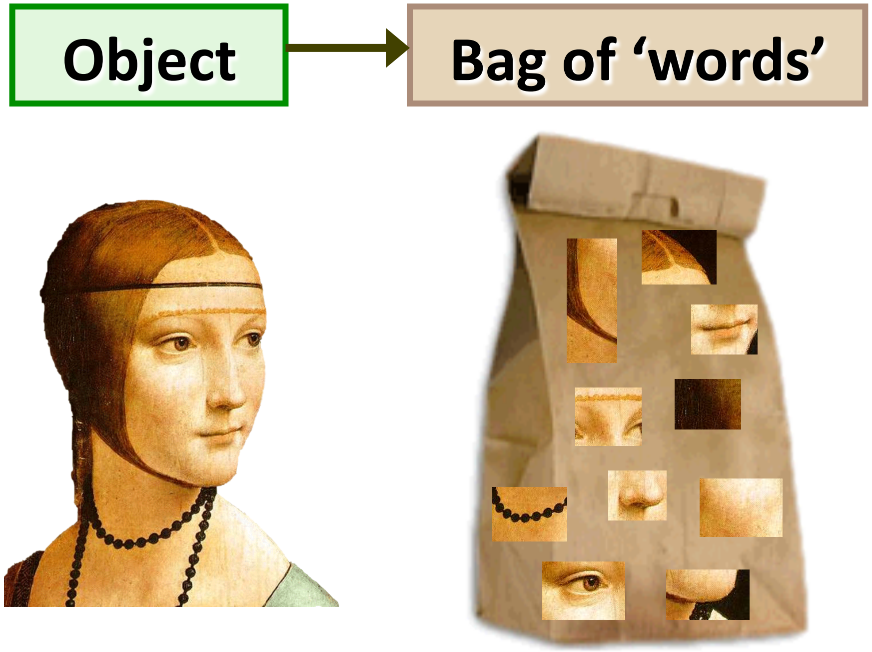
# CS4670/5670: Computer Vision

Kavita Bala

## Lecture 35: Recognition Wrapup



# ConvNets breakthroughs for visual tasks

| | Dataset | Performance | Score |
|---|---|---|---|
| **[Sermanet et al 2014]: OverFeat (fine-tuned features for each task)** (tasks are ordered by increasing difficulty) | | | |
| • image classification | ImageNet LSVRC 2013 | competitive | 13.6 % error |
| | Dogs vs Cats Kaggle challenge 2014 | **state of the art** | 98.9% |
| • object localization | ImageNet LSVRC 2013 | **state of the art** | 29.9% error |
| • object detection | ImageNet LSVRC 2013 | competitive | 24.3% mAP |
| **[Razavian et al, 2014]: public OverFeat library (no retraining) + SVM** **(simplest approach possible on purpose, no attempt at more complex classifiers)** (tasks are ordered by “distance” from classification task on which OverFeat was trained) | | | |
| • image classification | Pascal VOC 2007 | competitive | 77.2% mAP |
| • scene recognition | MIT-67 | **state of the art** | 69% mAP |
| • fine grained recognition | Caltech-UCSD Birds 200-2011 | competitive | 61.8% mAP |
| | Oxford 102 Flowers | **state of the art** | 86.8% mAP |
| • attribute detection | UIUC 64 object attributes | **state of the art** | 91.4% mAUC |
| | H3D Human Attributes | competitive | 73% mAP |
| • image retrieval (search by image similarity) | Oxford 5k buildings | **state of the art** | 68% mAP? |
| | Paris 6k buildings | **state of the art** | 79.5% mAP? |
| | Sculp6k | competitive | 42.3% mAP? |
| | Holidays | **state of the art** | 84.3% mAP? |
| | UKBench | **state of the art** | 91.1% mAP? |

Pierre Sermanet, David Eigen, Xiang Zhang, Michael Mathieu, Rob Fergus, Yann LeCun, **OverFeat: Integrated Recognition, Localization and Detection using Convolutional Networks**, http://arxiv.org/abs/1312.6229, ICLR 2014
Ali Sharif Razavian, Hossein Azizpour, Josephine Sullivan, Stefan Carlsson, **CNN Features off-the-shelf: an Astounding Baseline for Recognition**, http://arxiv.org/abs/1403.6382, DeepVision CVPR 2014 workshop

# ConvNets breakthroughs for visual tasks

| | Dataset | Performance | Score |
|---|---|---|---|
| **[Zeiler et al 2013]** | | | |
| • image classification | ImageNet LSVRC 2013 | **state of the art** | 11.2% error |
| | Caltech-101 (15, 30 samples per class) | competitive | 83.8%, 86.5% |
| | Caltech-256 (15, 60 samples per class) | **state of the art** | 65.7%, 74.2% |
| | Pascal VOC 2012 | competitive | 79% mAP |
| **[Donahue et al, 2014]: DeCAF+SVM** | | | |
| • image classification | Caltech-101 (30 classes) | **state of the art** | 86.91% |
| • domain adaptation | Amazon -> Webcam, DSLR -> Webcam | **state of the art** | 82.1%, 94.8% |
| • fine grained recognition | Caltech-UCSD Birds 200-2011 | **state of the art** | 65.0% |
| • scene recognition | SUN-397 | competitive | 40.9% |
| **[Girshick et al, 2013]** | | | |
| • image detection | Pascal VOC 2007 | **state of the art** | 48.0% mAP |
| | Pascal VOC 2010 (comp4) | **state of the art** | 43.5% mAP |
| | ImageNet LSVRC 2013 | **state of the art** | 31.4% mAP |
| • image segmentation | Pascal VOC 2011 (comp6) | **state of the art** | 47.9% mAP |
| **[Oquab et al, 2013]** | | | |
| • image classification | Pascal VOC 2007 | **state of the art** | 77.7% mAP |
| | Pascal VOC 2012 | **state of the art** | 82.8% mAP |
| | Pascal VOC 2012 (action classification) | **state of the art** | 70.2% mAP |

M.D. Zeiler, R. Fergus, **Visualizing and Understanding Convolutional Networks,** Arxiv 1311.2901 http://arxiv.org/abs/1311.2901
J. Donahue, Y. Jia, O. Vinyals, J. Hoffman, N. Zhang, E. Tzeng, and T. Darrell. **Decaf: A deep convolutional activation feature for generic visual recognition**. In ICML, 2014, http://arxiv.org/abs/1310.1531
R. B. Girshick, J. Donahue, T. Darrell, and J. Malik. **Rich feature hierarchies for accurate object detection and semantic segmentation**. arxiv:1311.2524 [cs.CV], 2013, http://arxiv.org/abs/1311.2524
M. Oquab, L. Bottou, I. Laptev, and J. Sivic. **Learning and transferring mid-level image representations using convolutional neural networks.** Technical Report HAL-00911179, INRIA, 2013. http://hal.inria.fr/hal-00911179

# ConvNets breakthroughs for visual tasks

| | Dataset | Performance | Score |
|---|---|---|---|
| **[Khan et al 2014]**<br>• shadow detection | UCF<br>CMU<br>UIUC | **state of the art**<br>**state of the art**<br>**state of the art** | 90.56%<br>88.79%<br>93.16% |
| **[Sander Dieleman, 2014]**<br>• image attributes | Kaggle Galaxy Zoo challenge | **state of the art** | 0.07492 |

S. H. Khan, M. Bennamoun, F. Sohel, R. Togneri. **Automatic Feature Learning for Robust Shadow Detection,** CVPR 2014
Sander Dieleman, Kaggle Galaxy Zoo challenge 2014 http://benanne.github.io/2014/04/05/galaxy-zoo.html

# Image Captioning

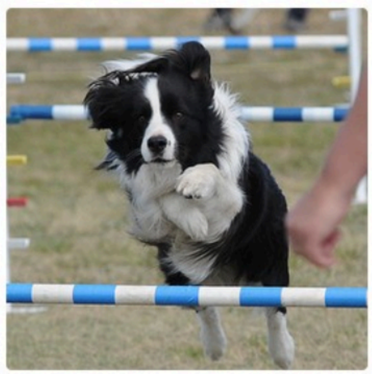
"man in black shirt is playing guitar."

"construction worker in orange safety vest is working on road."

"two young girls are playing with lego toy."

"boy is doing backflip on wakeboard."

"girl in pink dress is jumping in air."

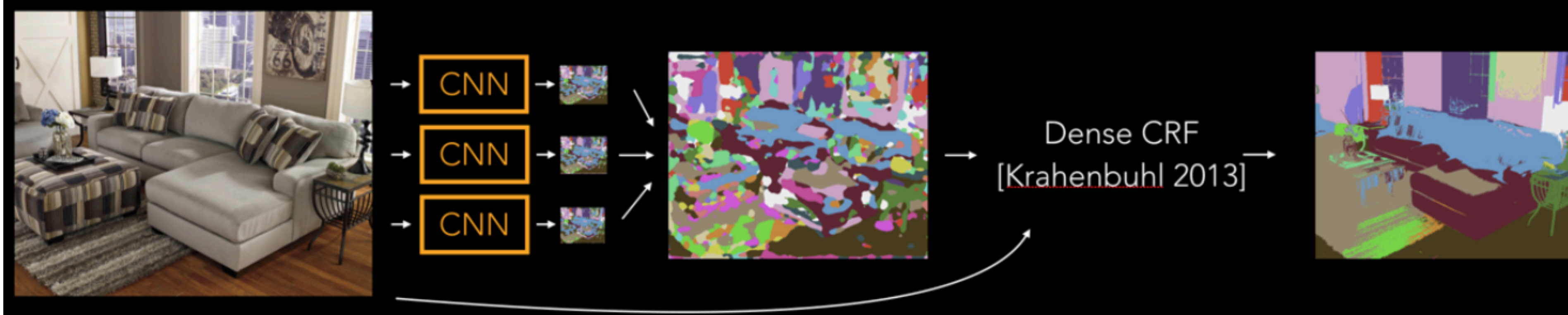


"black and white dog jumps over bar."

"young girl in pink shirt is swinging on swing."

"man in blue wetsuit is surfing on wave."

# CNNs + CRFs




*CRF Runtime: ~1s for 640x480 image*

$$E(\mathbf{x}|\mathbf{I},\boldsymbol{\theta}) = \sum_i \psi_i(x_i|\boldsymbol{\theta}) + \sum_{i<j} \psi_{ij}(x_i, x_j|\boldsymbol{\theta})$$

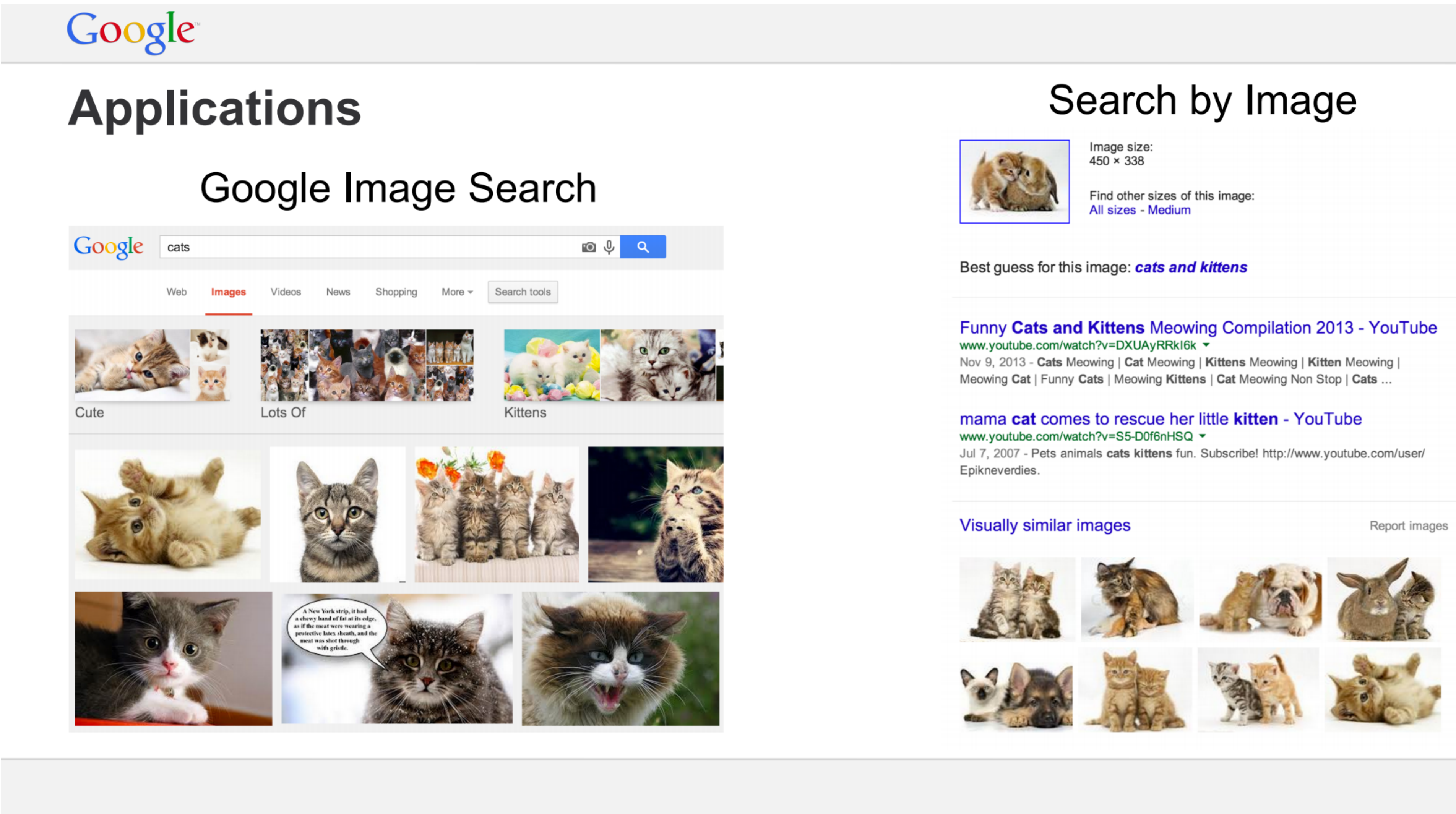
# Material Segmentation[CVPR15]


Bell, Upchurch, Snavely, Bala

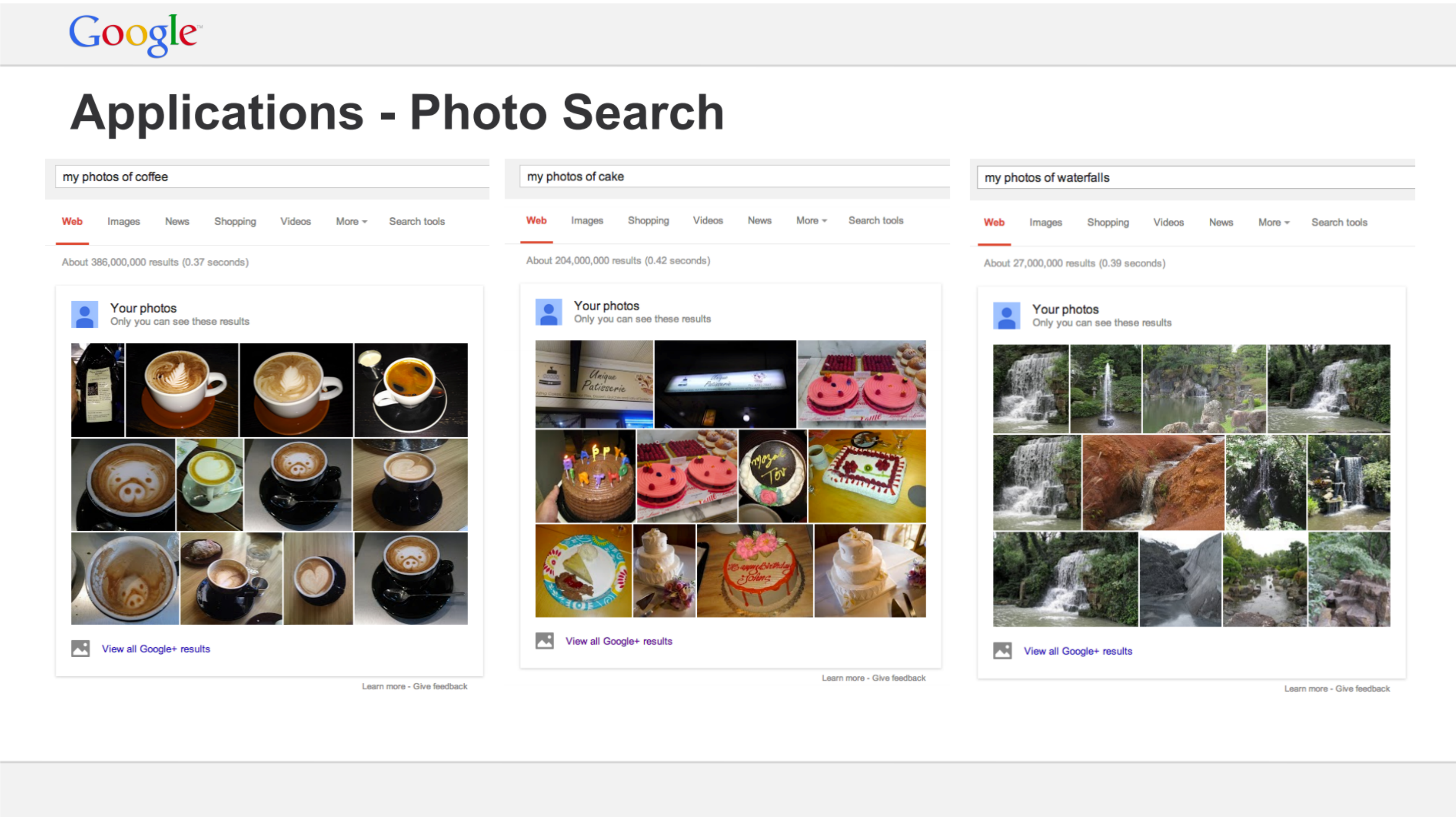
“It can be concluded that from now on, deep learning with CNN has to be considered as the primary candidate in essentially any visual recognition task.”

[Razavian 2014]

# CNNs at Google (as of 2014)



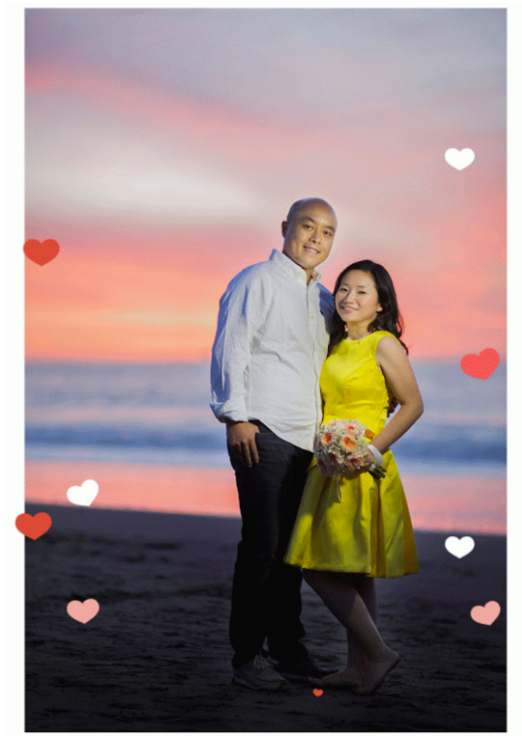
# CNNs at Google (as of 2014)



# CNNs at Google (as of 2014)

Google

## Google Photos - Auto Awesome



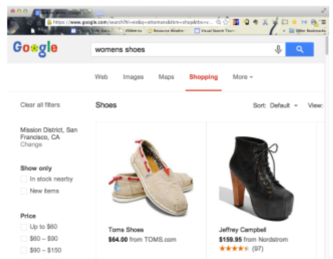
# CNNs at Google (as of 2014)

Google

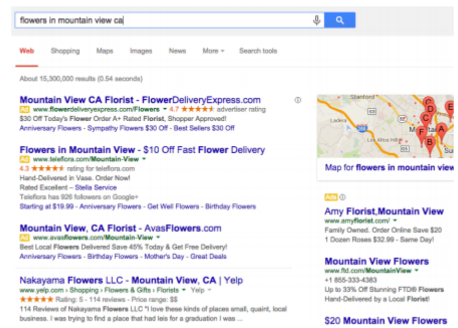
## More Image Understanding at Google

YouTube

YouTube




Google Shopping




Advertising

Much more...

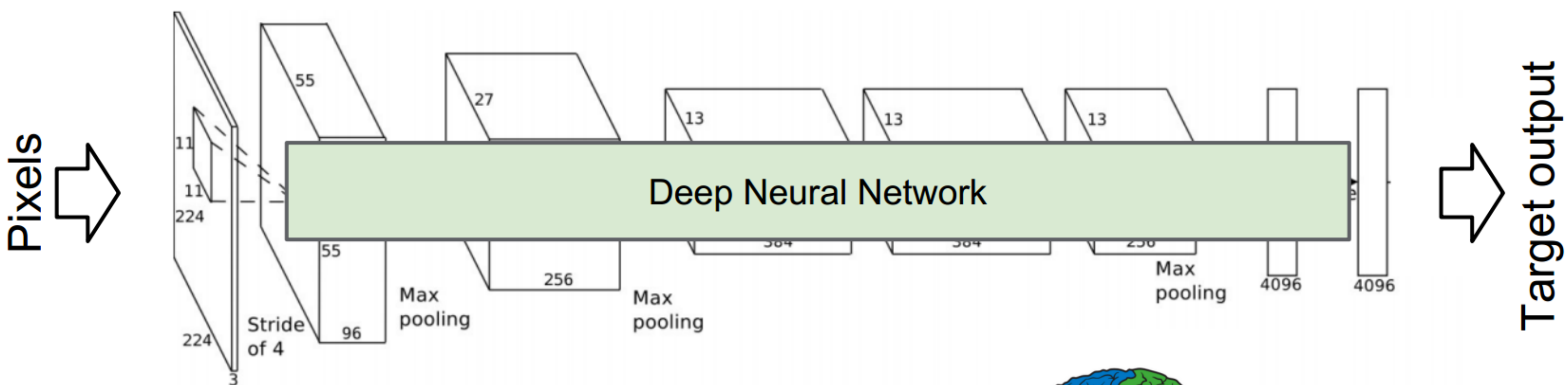
StreetView / Maps

Self-Driving Cars

Robotics

# CNNs at Google (as of 2014)

## The Deep and now Deeper Hammer




**Deep learning infrastructure by the Google Brain team**

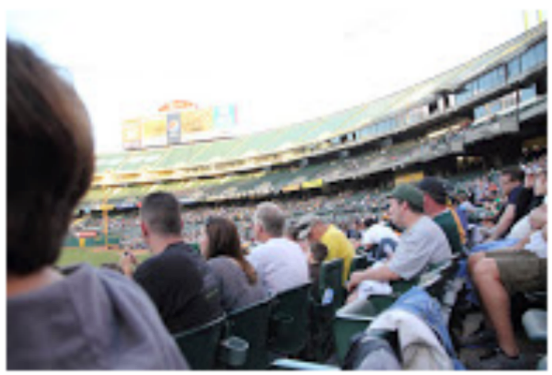
**"ImageNet Classification with Deep Convolutional Neural Networks", Krizhevsky, Sutskever, Hinton, NIPS 2012**

# CNNs at Google (as of 2014)

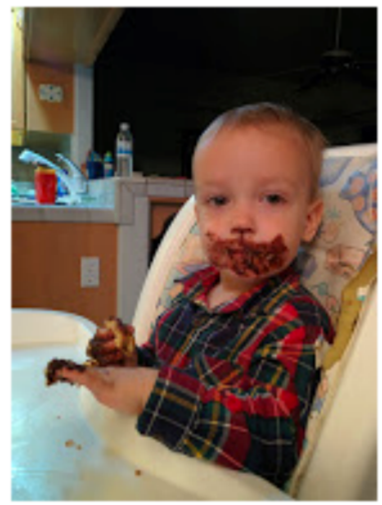
Google

Personal Photos - Example Annotations

**Christmas tree**
**Red**
**Christmas decoration**
**Christmas**



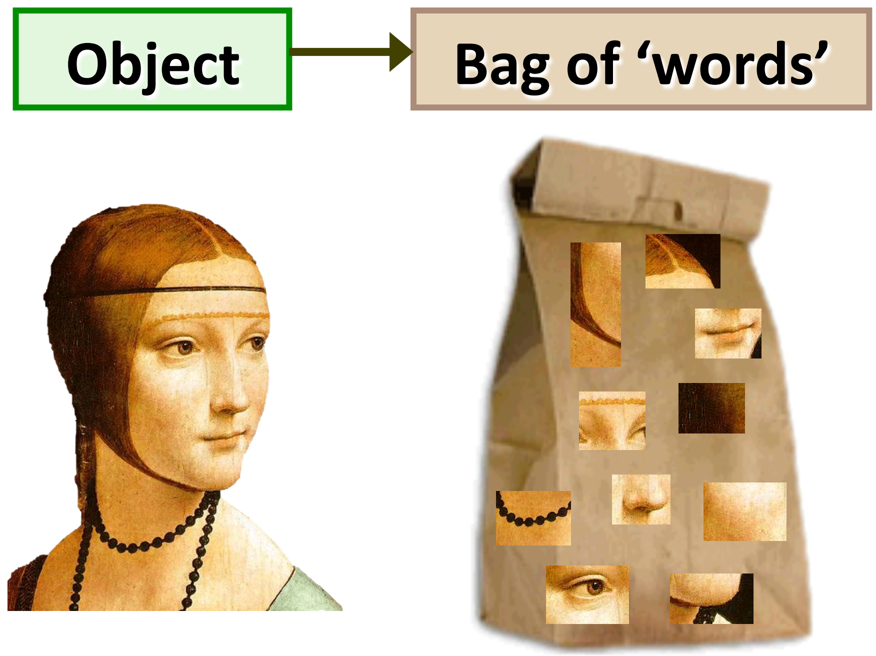
**Crowd**
**Cheering**
**People**
**Stadium**



**Play**
**Meal**
**Cake**
**Child**

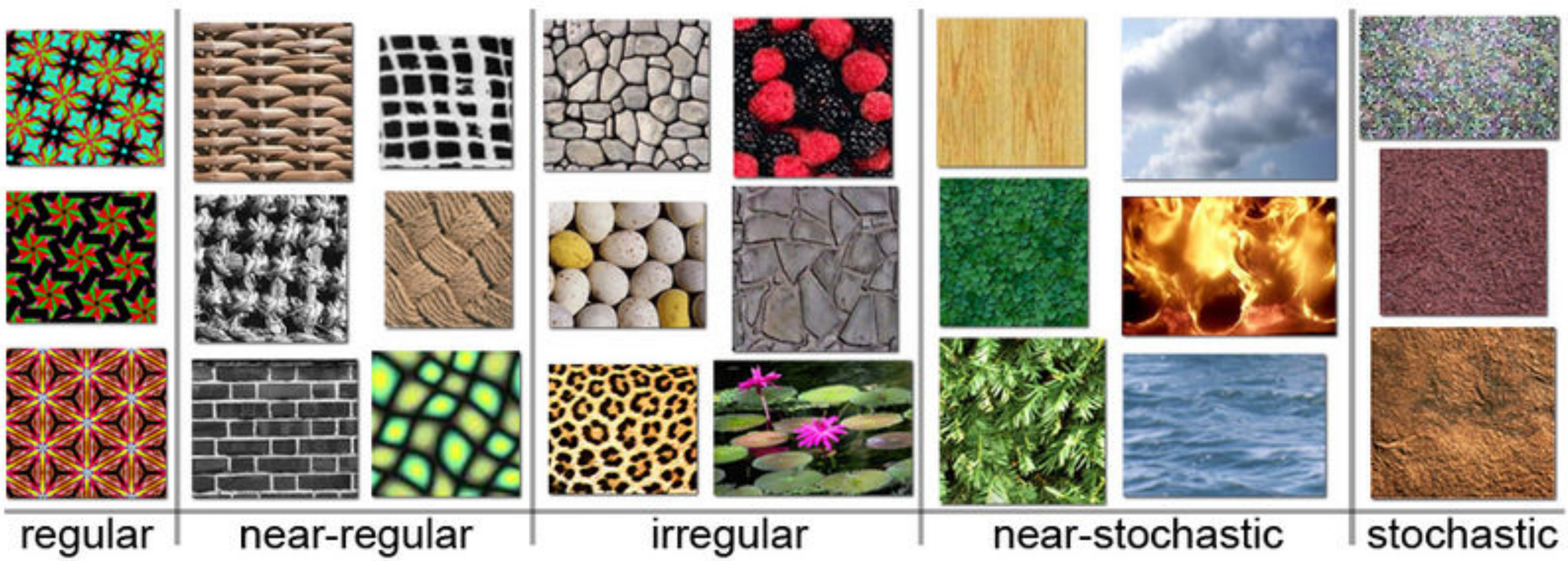
**Hummingbird**
**Macro photography**
**Reflection**
**Red**

# Before CNNs: Bag of words




Adapted from slides by Rob Fergus and Svetlana Lazebnik

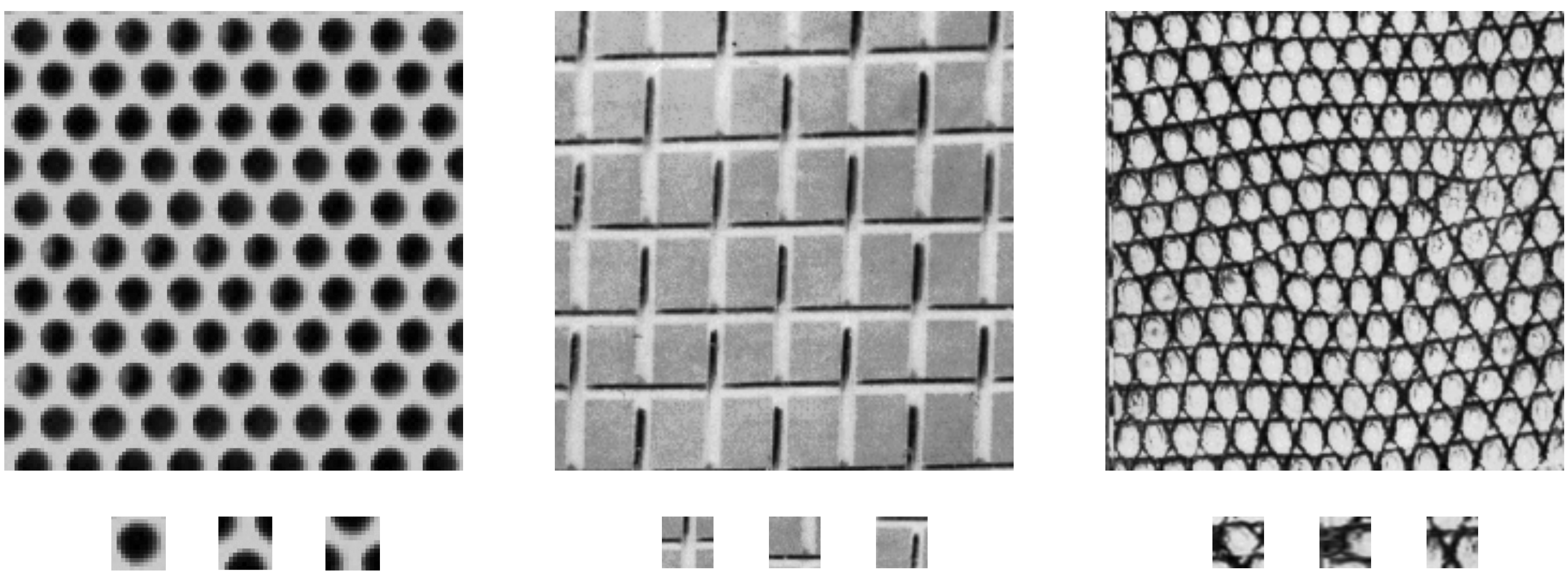
# Origin 1: Texture Recognition




Example textures (from Wikipedia)

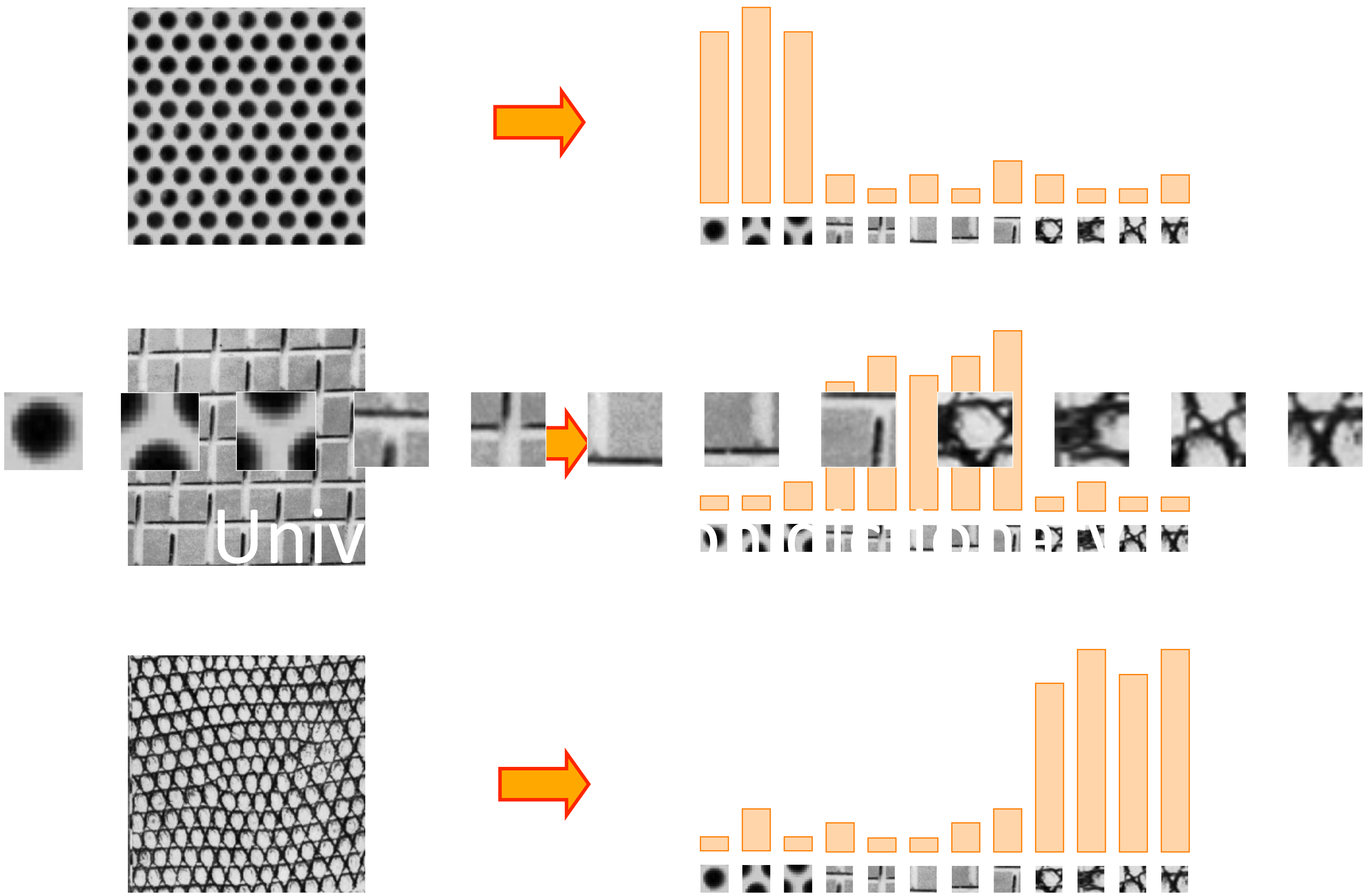
# Origin 1: Texture recognition

- Texture is characterized by the repetition of basic elements or *textons*
- For stochastic textures, the identity of the textons, not their spatial arrangement, matters



Julesz, 1981; Cula & Dana, 2001; Leung & Malik 2001; Mori, Belongie & Malik, 2001; Schmid 2001; Varma & Zisserman, 2002, 2003; Lazebnik, Schmid & Ponce, 2003

# Origin 1: Texture recognition



# Origin 2: Bag-of-words models

- Orderless document representation: frequencies of words from a dictionary Salton & McGill (1983)

# Origin 2: Bag-of-words models

- Orderless document representation: frequencies of words from a dictionary  Salton & McGill (1983)

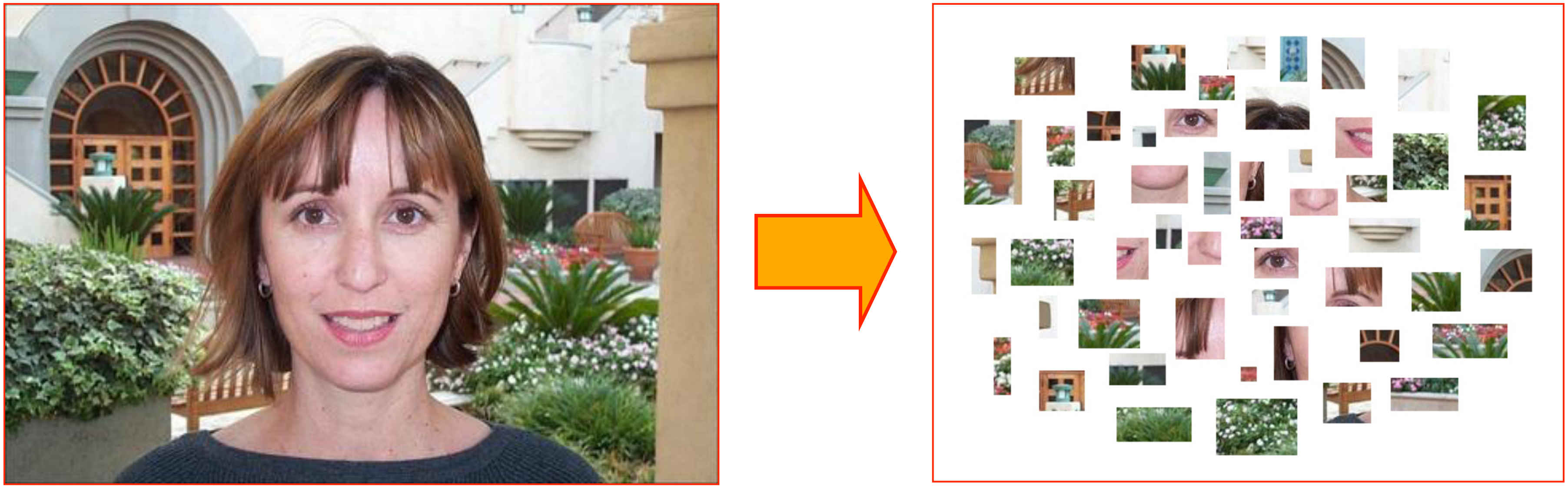
# Origin 2: Bag-of-words models

- Orderless document representation: frequencies of words from a dictionary Salton & McGill (1983)

# Origin 2: Bag-of-words models

- Orderless document representation: frequencies of words from a dictionary Salton & McGill (1983)

# Bags of features for object recognition



**face, flowers, building**

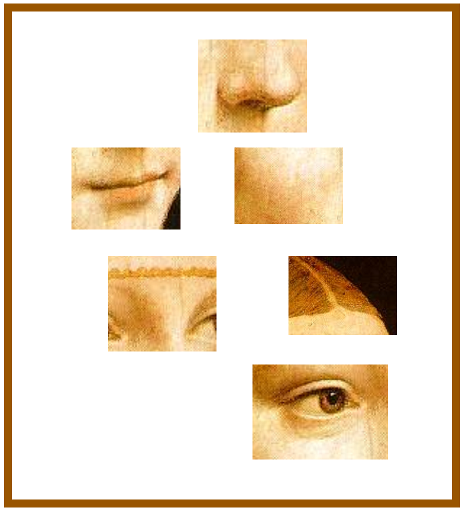
- Works pretty well for image-level classification and for recognizing object *instances*

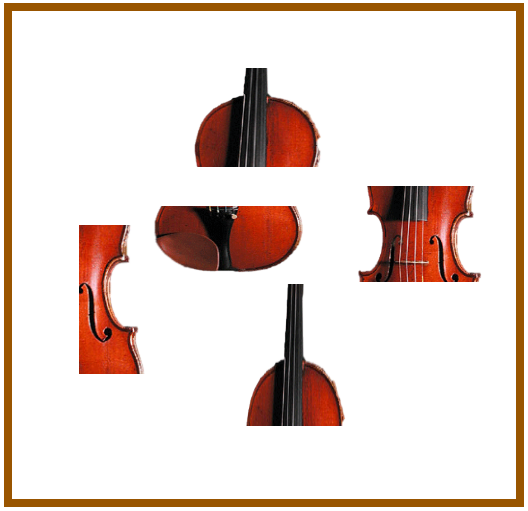
# Bag of features

- First, take a bunch of images, extract features, and build up a “dictionary” or “visual vocabulary” – a list of common features
- Given a new image, extract features and build a histogram – for each feature, find the closest visual word in the dictionary

# Bag of features: outline

1. Extract features





# Bag of features: outline

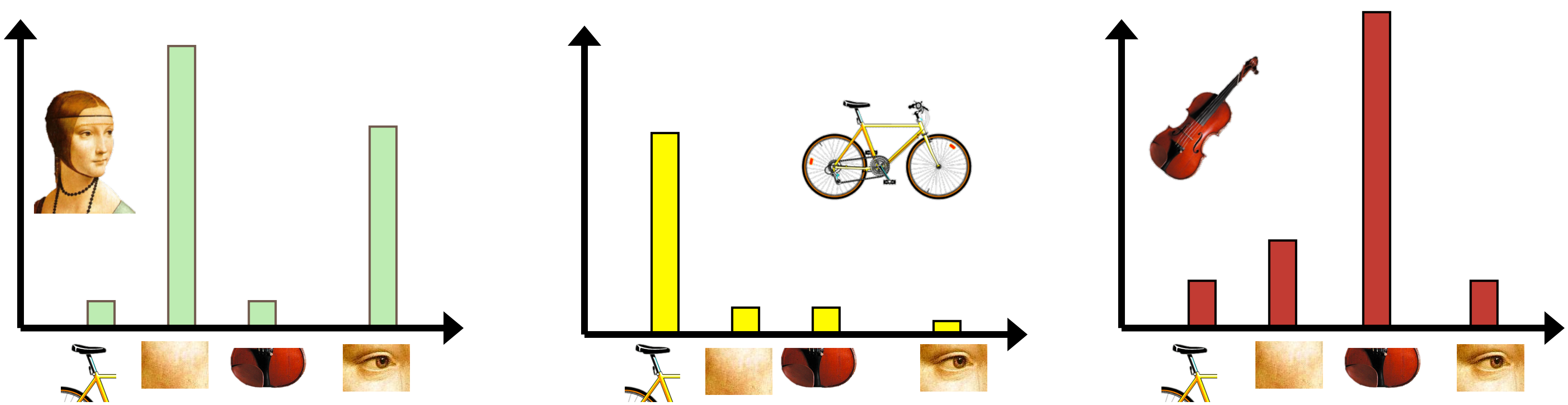
1. Extract features
2. Learn "visual vocabulary"

# Bag of features: outline

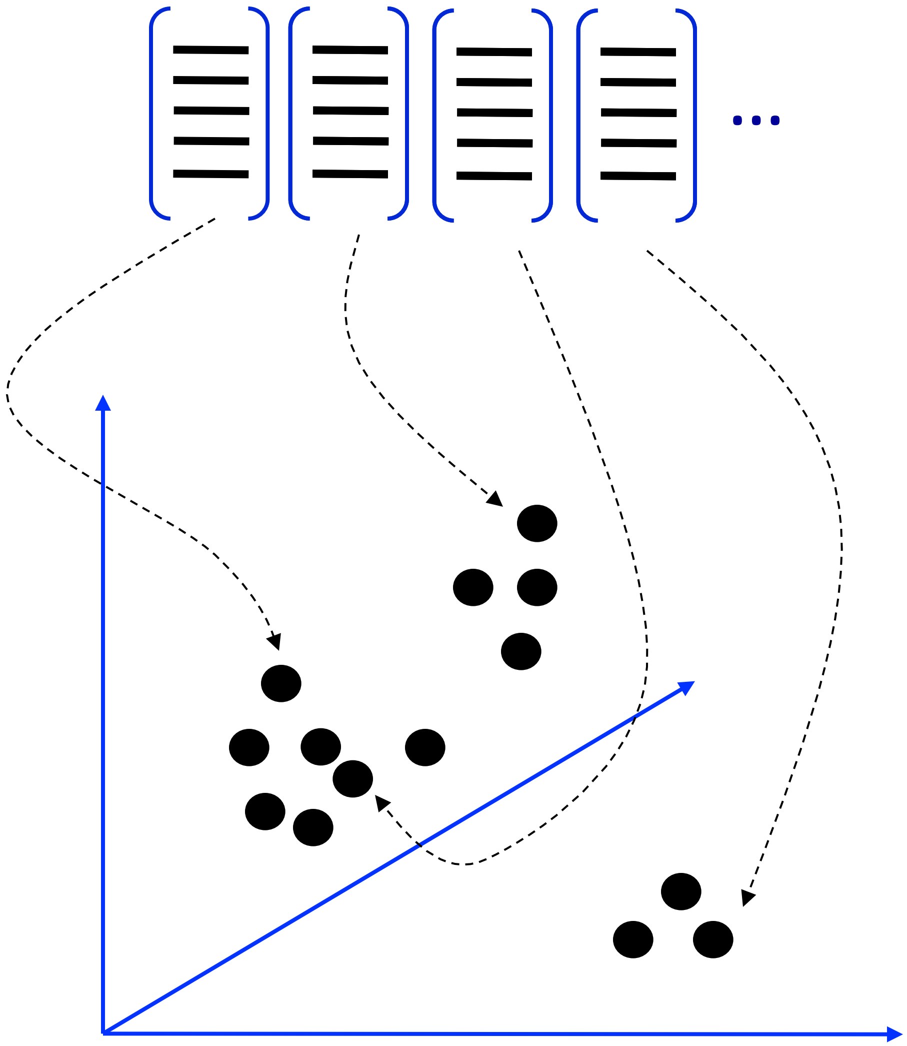
1. Extract features
2. Learn “visual vocabulary”
3. Quantize features using visual vocabulary

# Bag of features: outline

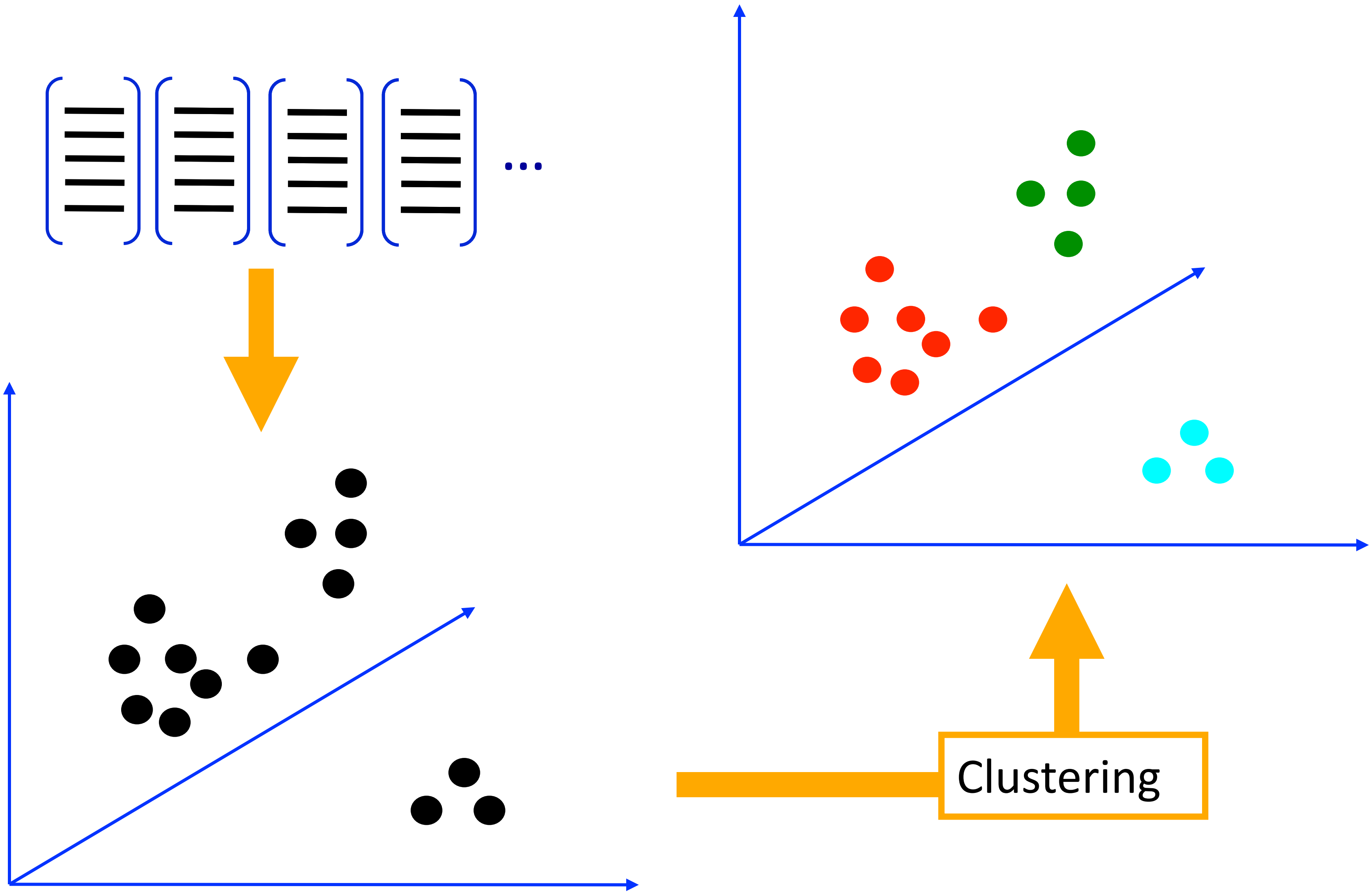
1. Extract features
2. Learn "visual vocabulary"
3. Quantize features using visual vocabulary
4. Represent images by frequencies of "visual words"



# 2. Learning the visual vocabulary

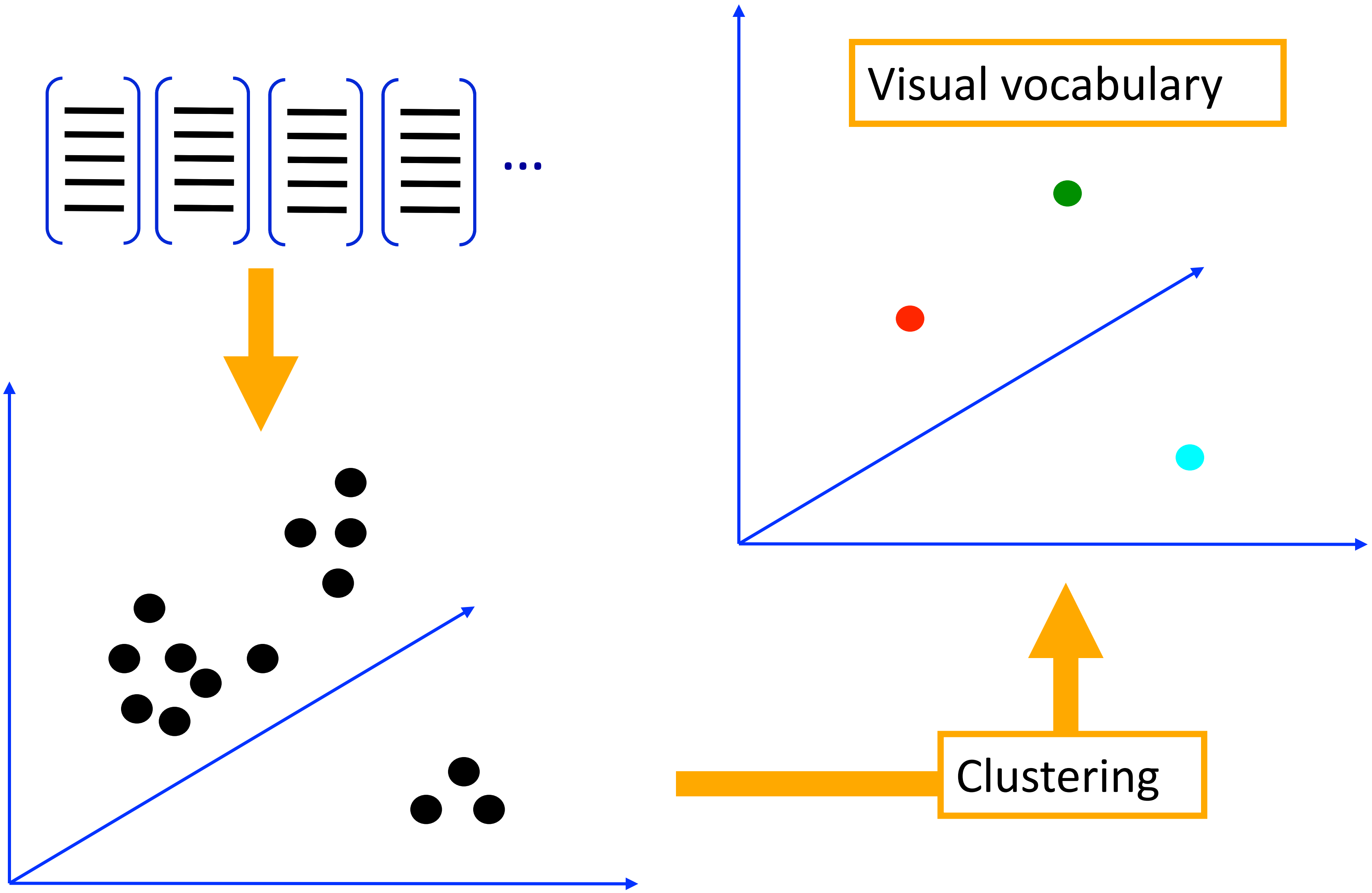


# 2. Learning the visual vocabulary




Slide credit: Josef Sivic

# 2. Learning the visual vocabulary




Slide credit: Josef Sivic

# K-means clustering

- Want to minimize sum of squared Euclidean distances between points $x_i$ and their nearest cluster centers $m_k$

$$D(X, M) = \sum_{\text{cluster } k} \sum_{\substack{\text{point } i \text{ in} \\ \text{cluster } k}} (x_i - m_k)^2$$

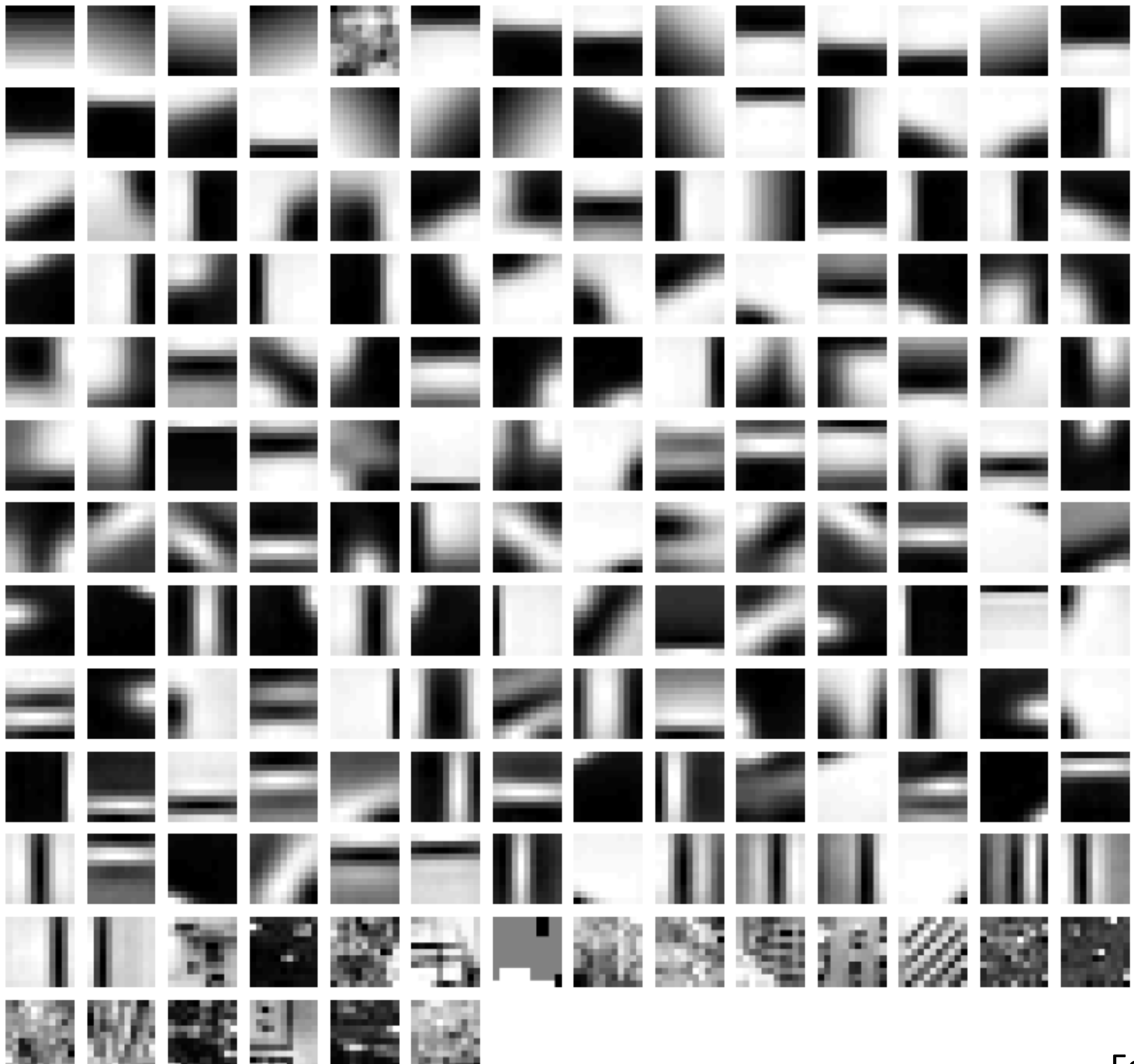
- Algorithm:
- Randomly initialize K cluster centers
- Iterate until convergence:
  - Assign each data point to the nearest center
  - Recompute each cluster center as the mean of all points assigned to it

# From clustering to vector quantization

- Clustering is a common method for learning a visual vocabulary or codebook
  - Unsupervised learning process
  - Each cluster center produced by k-means becomes a codevector
  - Provided the training set is sufficiently representative, the codebook will be "universal"

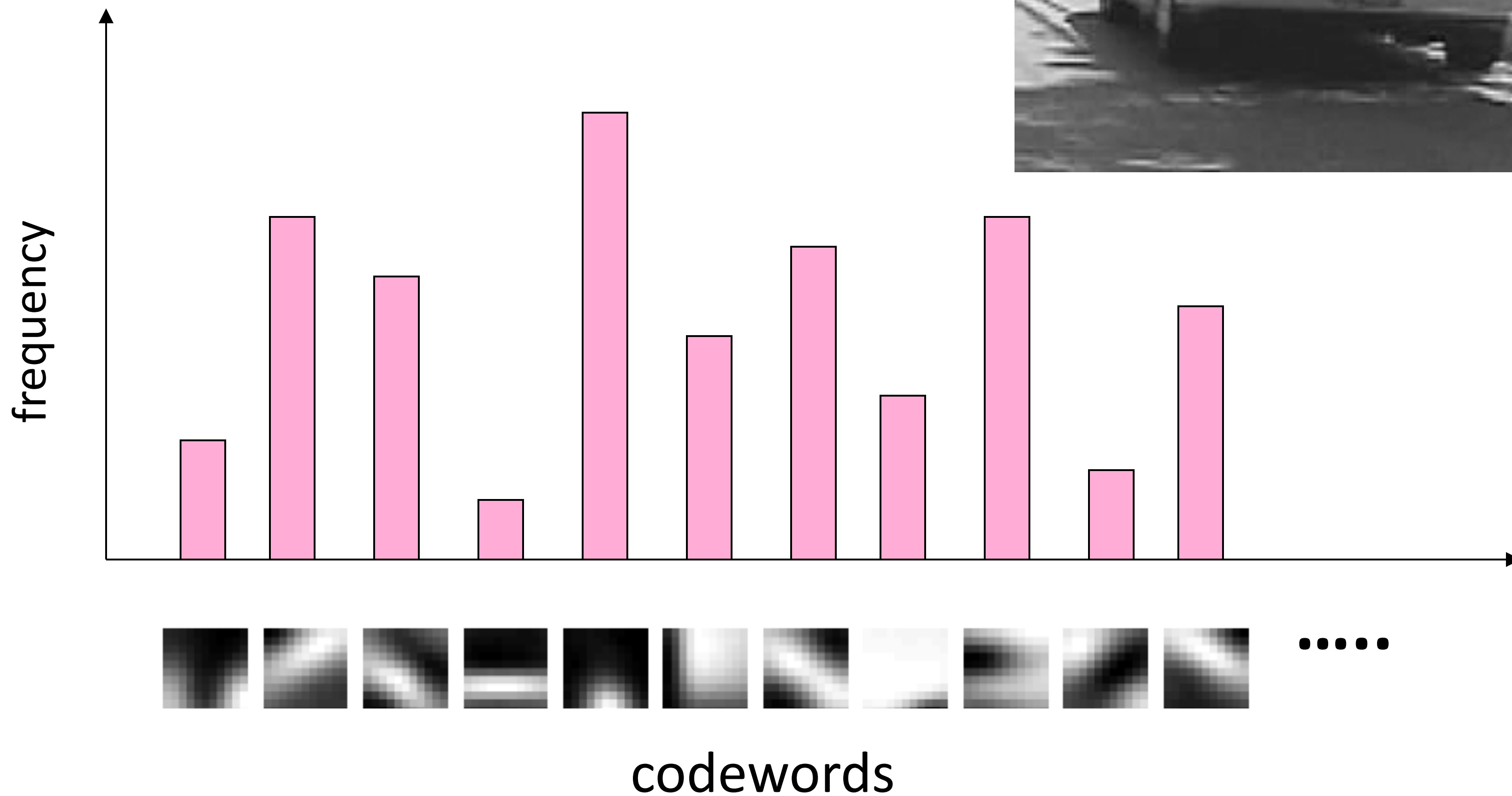
- The codebook is used for quantizing features
  - A *vector quantizer* takes a feature vector and maps it to the index of the nearest codevector in a codebook
  - Codebook = visual vocabulary
  - Codevector = visual word

# Example visual vocabulary



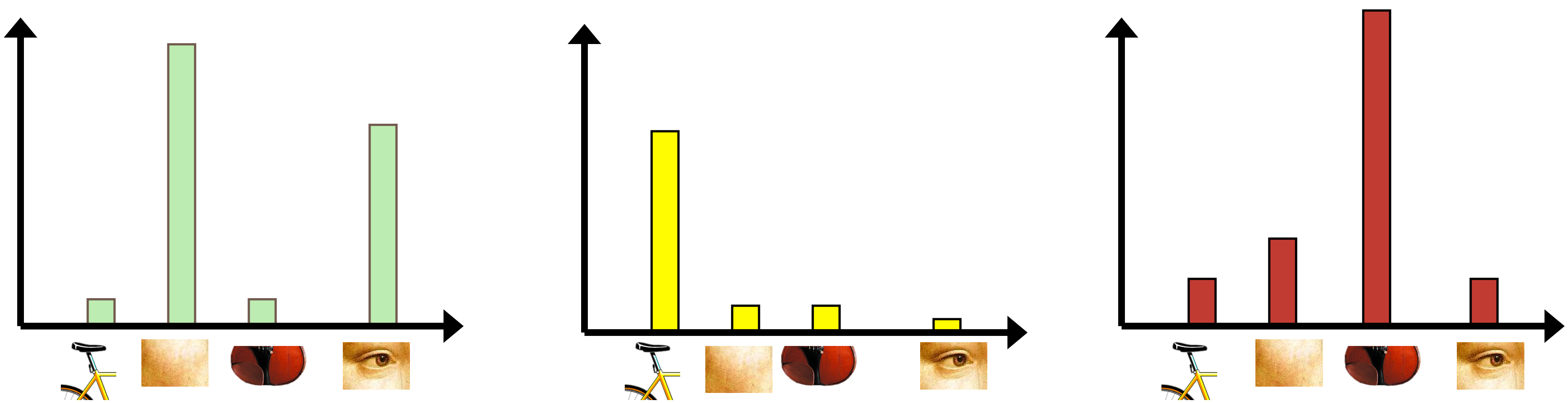
Fei-Fei et al. 2005

# 3. Image representation



# Image classification

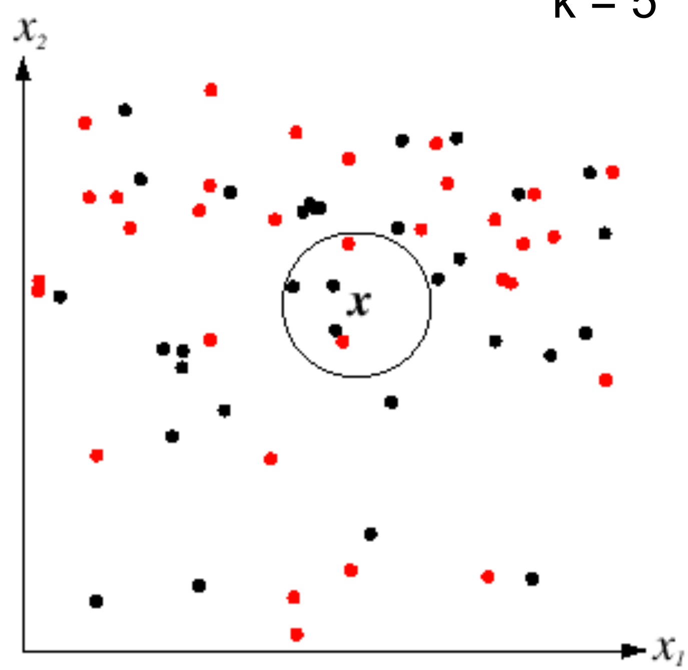
- Given the bag-of-features representations of images from different classes, classify image.



# K nearest neighbors

- For a new point, find the k closest points from training data
- Labels of the k points “vote” to classify
- Works well provided there is lots of data and the distance function is good

k = 5




Source: D. Lowe

# Uses of BoW representation

- Treat as feature vector for standard classifier
  - e.g k-nearest neighbors, support vector machine
- Cluster BoW vectors over image collection
  - Discover visual themes

# Large-scale image matching

11,400 images of game covers
(Caltech games dataset)

- Bag-of-words models have been useful in matching an image to a large database of object *instances*


how do I find this image in the database?

# Large-scale image search

- Build the database:
  - Extract features from the database images
  - Learn a vocabulary using k-means (typical k: 100,000)
  - Compute *weights* for each word
  - Create an inverted file mapping words → images

# Weighting the words

- Just as with text, some visual words are more discriminative than others

***the, and, or*** vs. ***cow, AT&T, Cher***

- the bigger fraction of the documents a word appears in, the less useful it is for matching
  - e.g., a word that appears in *all* documents is not helping us

# TF-IDF weighting

- Instead of computing a regular histogram distance, we'll weight each word by it's *inverse document frequency*

- inverse document frequency (IDF) of word $j$ =

$$\log \frac{\text{number of documents}}{\text{number of documents in which } j \text{ appears}}$$

# TF-IDF weighting

- To compute the value of bin *j* in image *I*:

*term frequency* of *j* in *I* **x** *inverse document frequency* of *j*

# Inverted file

- Each image has ~1,000 features
- We have ~100,000 visual words

  →each histogram is extremely sparse (mostly zeros)

- Inverted file
  - mapping from words to documents

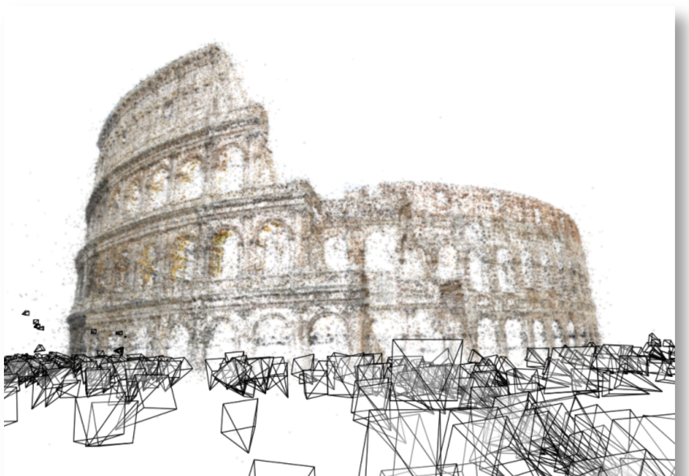
```
"a":      {2}
"banana": {2}
"is":     {0, 1, 2}
"it":     {0, 1, 2}
"what":   {0, 1}
```

# Inverted file

- Can quickly use the inverted file to compute similarity between a new image and all the images in the database
  - Only consider database images whose bins overlap the query image

# …into 3D models

St. Peter's Basilica



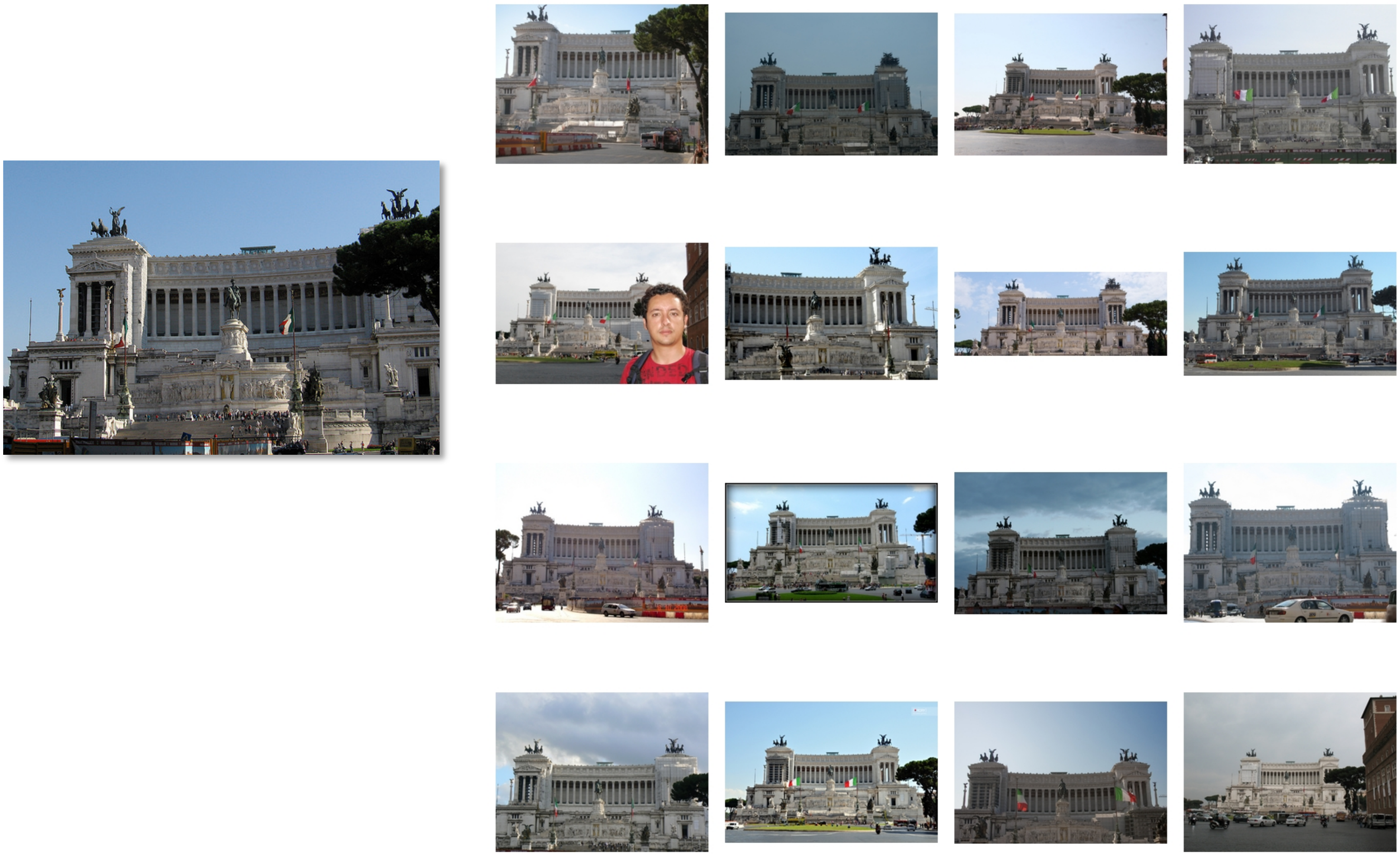
Colosseum

Trevi Fountain

# Large-scale image matching

- How can we match 1,000,000 images to each other?
- Brute force approach: 500,000,000,000 pairs
  - won’t scale
- Better approach: use bag-of-words technique to find *likely* matches
- For each image, find the top M scoring other images, do detailed SIFT matching with those

# Example bag-of-words matches



# Example bag-of-words matches

# Matching Statistics

| Dataset | Size | Matches possible | Matches Tried | Matches Found | Time |
|---|---|---|---|---|---|
| Dubrovnik | 58K | 1.6 Billion | 2.6M | 0.5M | 5 hrs |
| Rome | 150K | 11.2 Billion | 8.8M | 2.7M | 13 hrs |
| Venice | 250K | 31.2 Billion | 35.5M | 6.2M | 27 hrs |

# Quiz 4