# Object Recognition

## DD2423 Image Analysis and Computer Vision

Mårten Björkman

Robotics, Perception and Learning Division
School of Electrical Engineering and Computer Science

November 30, 2020

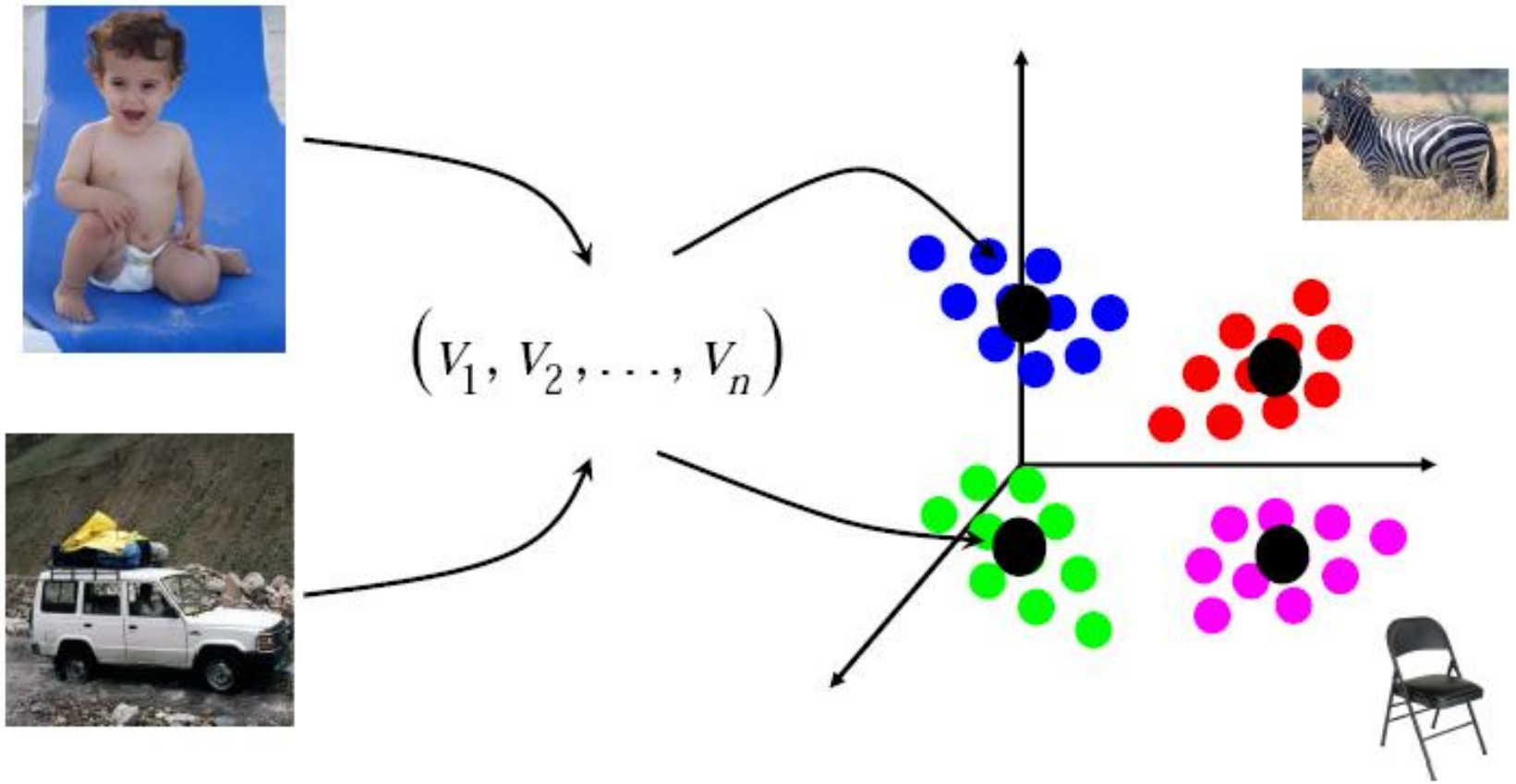
# Goals

- Recognition - Is this my cup?
  - Only requires a model of one cup.
  - Enough for many applications!
- Classification - What is this?
  - What do all cups have in common?
  - How to group objects into classes?
- Detection - If there is a cup, where is it?
  - An even harder problem.

# Common for all methods

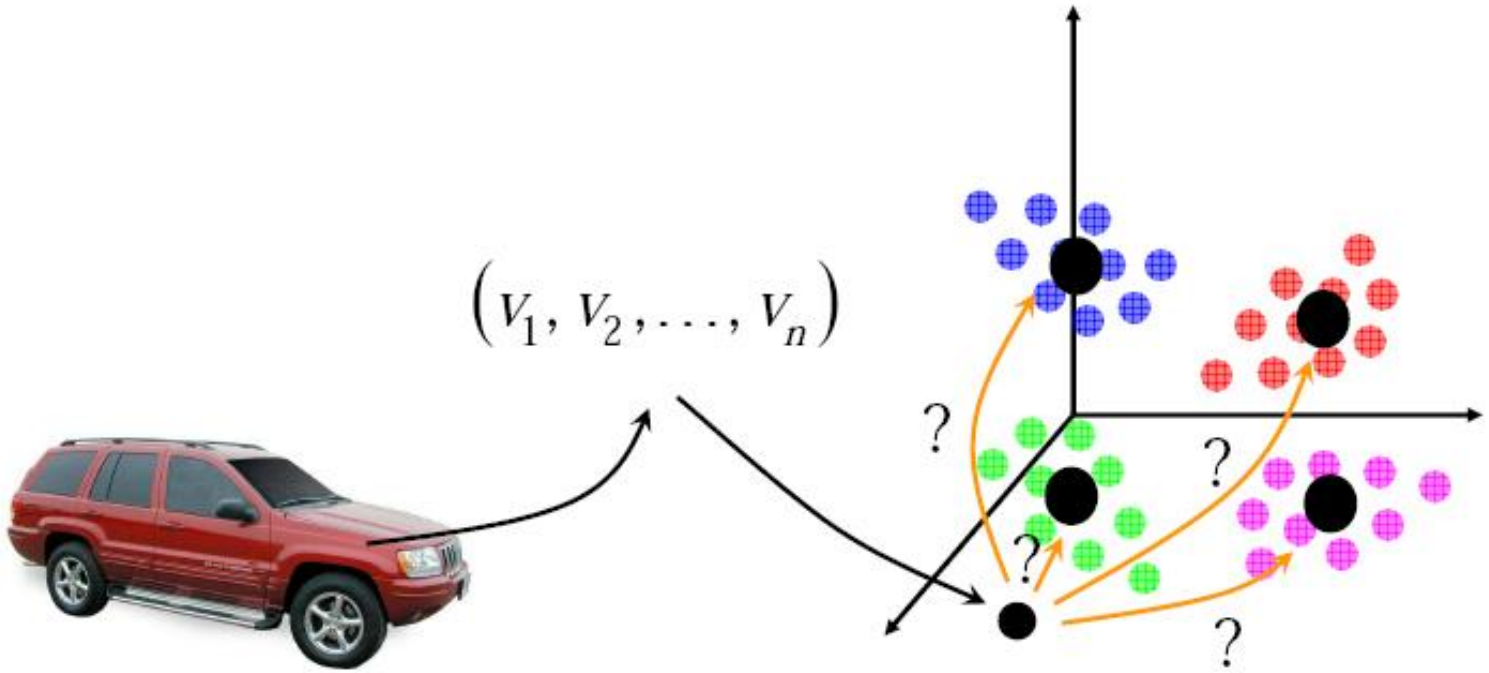
- Two or more hypotheses.
  - Is this a cup or not?
  - Is this a book, cup, monkey, pen or ...?
- Need for representation.
  - As small as possible, otherwise redundant.
  - A database of objects can be very large.
- Need for metric, such that representations of
  - ... the same class are close.
  - ... different classes are distant.

# Recognition (Step 1: representation)




Take an image (region) and represent it in some feature space.

# Recognition (Step 2: classification)




Given a new image in feature space, determine class it belongs to.

# Differences between methods

- Supervised vs unsupervised
  - Supervised: annotated training examples exist
  - Unsupervised (clustering): no annotated training examples
- Generative vs discriminative
  - Generative: models how things look like
    ex. face = 2 eyes + 1 nose + 1 mouth
  - Discriminative: models the difference between things
    ex. face vs non-face

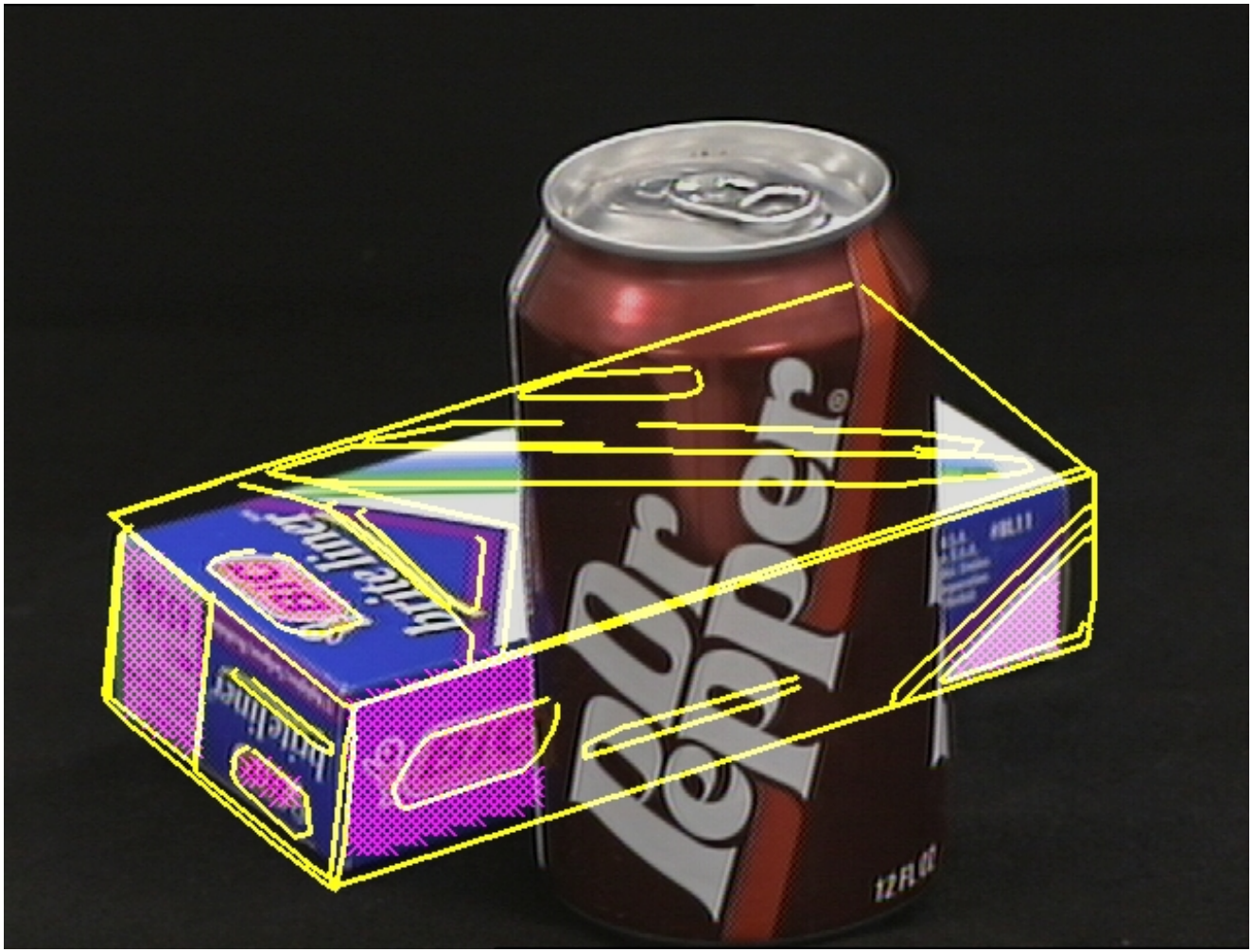
# Invariance

- Appearance of object may vary due to different:
  - Lighting conditions (shadows, colours)
  - Poses (viewing directions)
  - Deformations (changes in shape)
  - Clutter (occlusions)
  - Projective sizes (distances)
  - Cameras (projections, distortions)
- Thus... matching has to be (more or less) invariant to these.

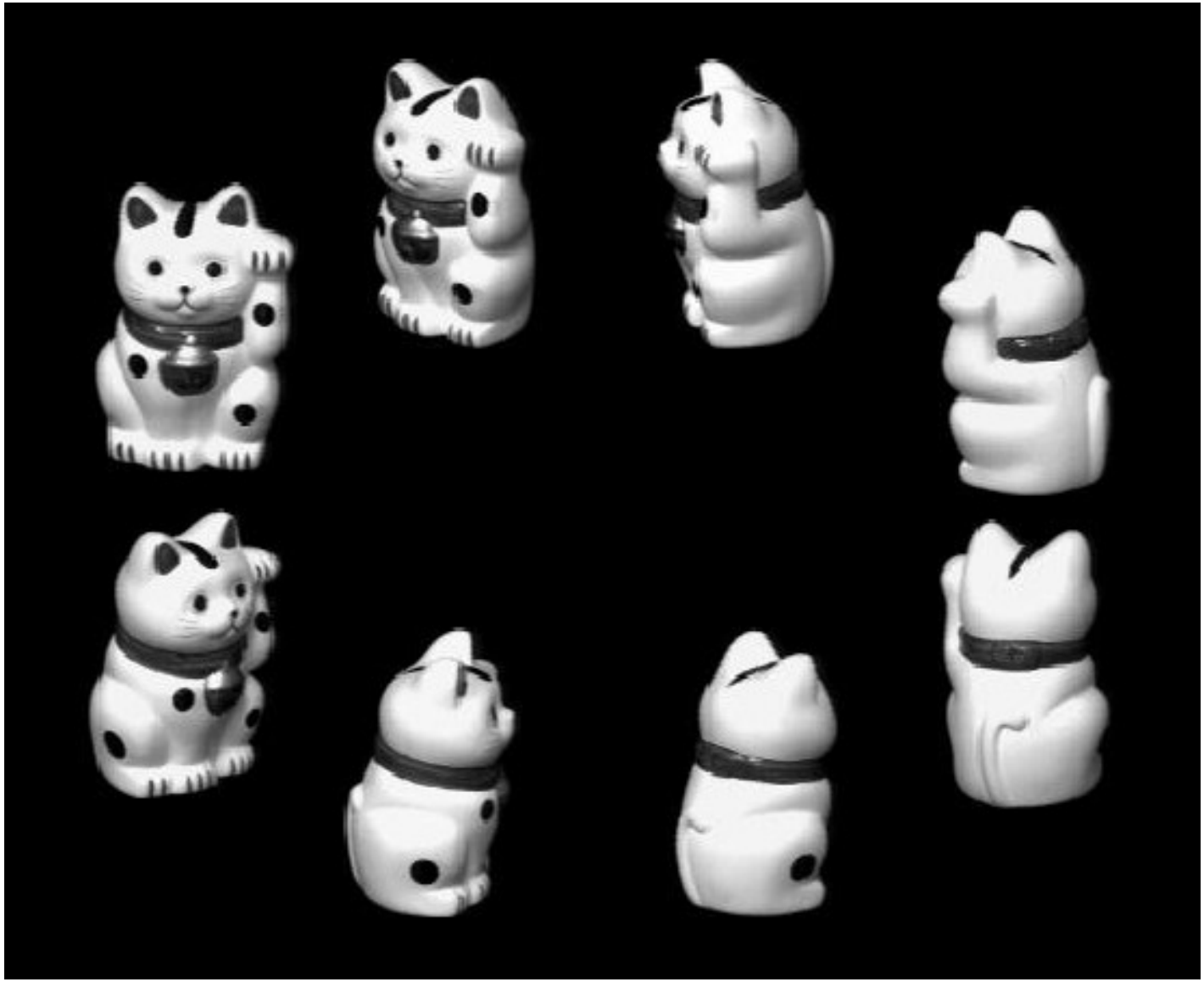
# Illumination Invariance

# Shadows

# Occlusion



# View-point Invariance



# Methods of representing objects

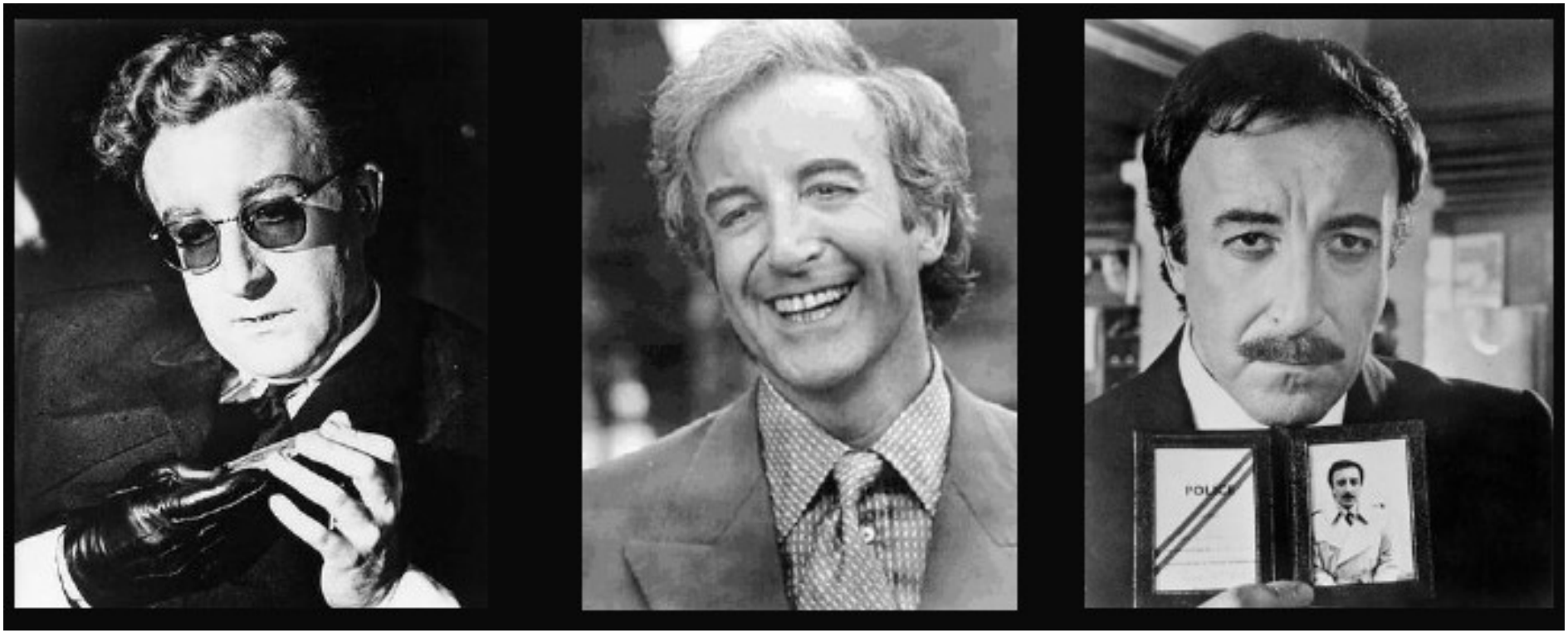
- Template based (dense)
  - Store some “image” of object (characters, faces).
  - Relative positions very important.
  - Normalized (lighting, scales) for invariance.
- Feature based (sparse)
  - Store sets of characteristic features (corners, contours).
  - Relative positions flexible.
  - Only keeps information that can be made invariant.
- Statistics based (usually dense)
  - Store histograms of image data (edges, colours).
  - Positions are disregarded (textures).
  - Each kind of data matched invariantly.

# Feature based methods

Methods typically contain:

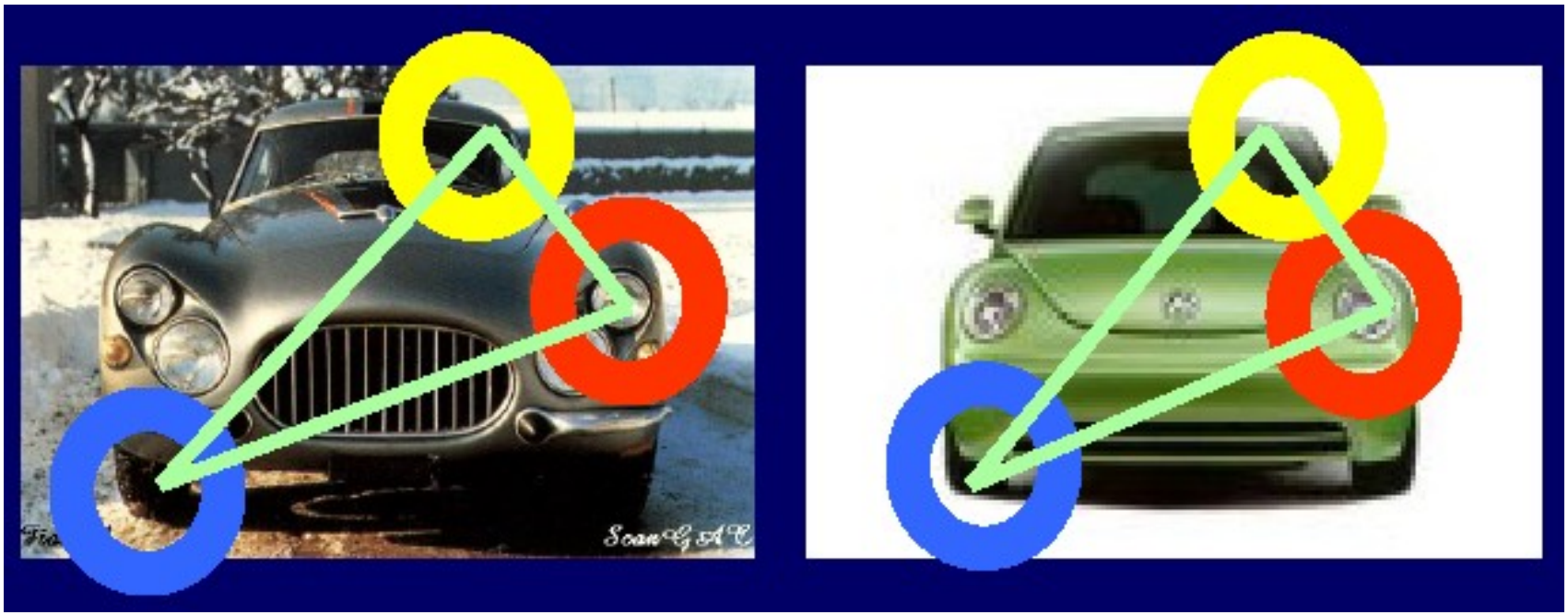
- Some feature detector
  - High curvature points (corners)
  - Scale-space blobs
- Some invariant descriptor(s)
  - Templates (small windows around features)
  - Statistics (edges, colours, etc.)
  - Combinations of templates and statistics
- Some way of combining features
  - Voting or similar combination measure
  - Often using relative position statistics

# Presence / Absence of features



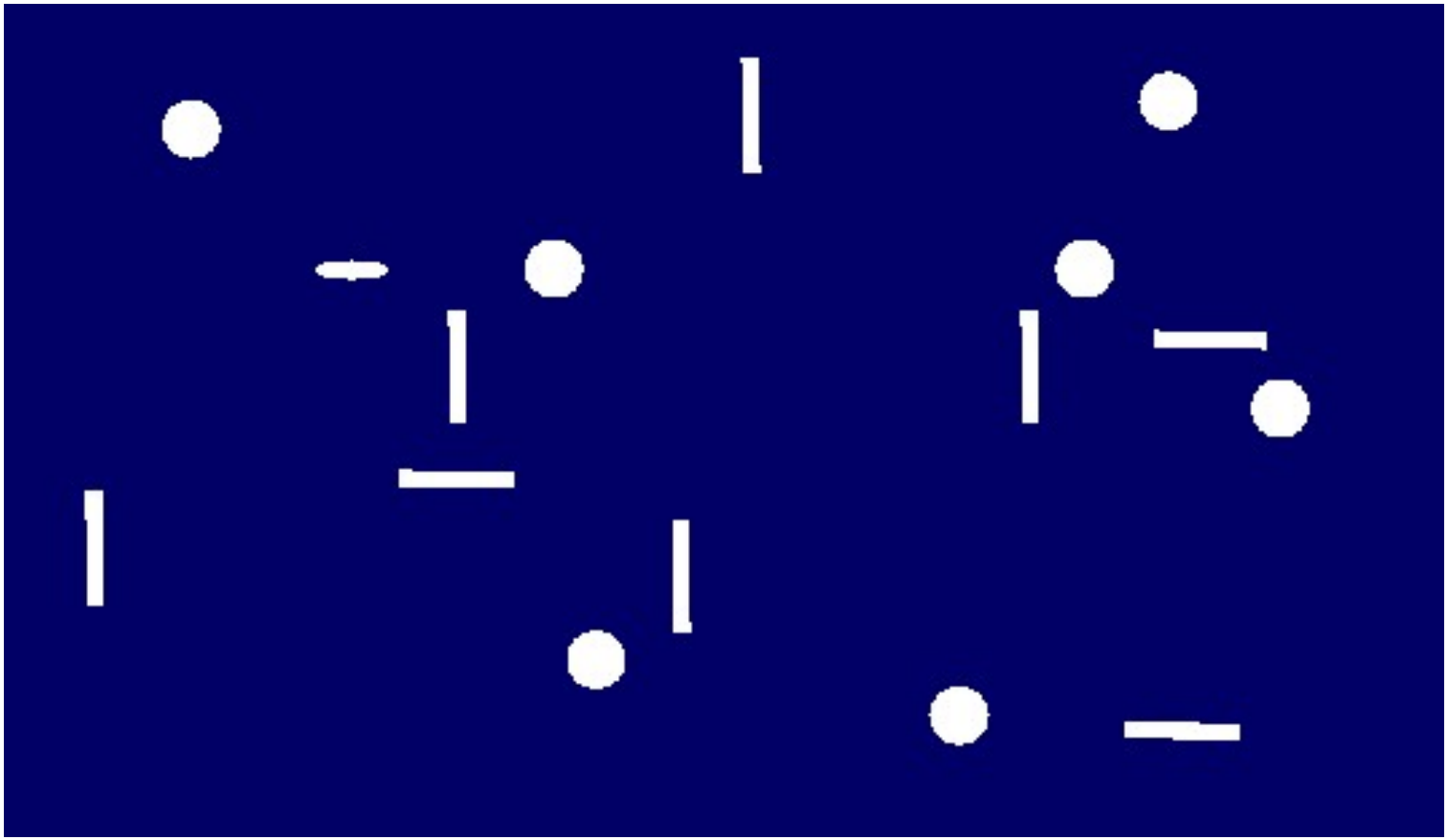
What kind of features exist in most training examples?

# Part similarity



Each part might look like many other parts, but combined they might strengthen a hypothesis.

# Importance of feature grouping



Can you see the face?

# Corner detection revisited

- Corner features are frequently used for recognition.
- Second Moment matrix:

$$M(\mathbf{x}; s, t) = g(\mathbf{x}; s) * \begin{pmatrix} L_x^2(\mathbf{x}; t) & L_x L_y(\mathbf{x}; t) \\ L_y L_x(\mathbf{x}; t) & L_y^2(\mathbf{x}; t) \end{pmatrix}$$

- Local measure of shape
  - Type 2: Textured areas ($\lambda_{min} \gg 0, \lambda_{max} \gg 0$)
  - Type 1: Line segment ($\lambda_{min} \approx 0, \lambda_{max} \gg 0$)
  - Type 0: Texture-less ($\lambda_{min} \approx 0, \lambda_{max} \approx 0$)

# Second Moment matrix

Only Type 2 features have accurate positions in 2D.

Type 0 Type 1 Type 2

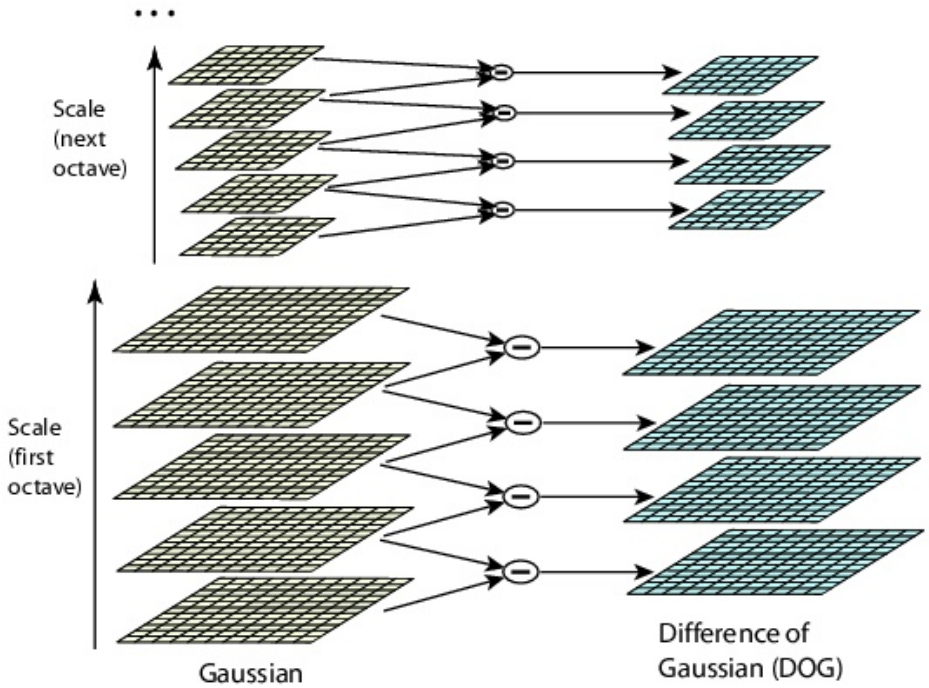
# Corner features (Harris & Stephens '88)

- Quick approach of finding Type 2 features.
- Most popular feature detector for many years.
- Features detected a local maxima of

$$\begin{aligned} C &= \det(M) - K \operatorname{trace}^2(M) \\ &= \lambda_{max}\lambda_{min} - K\,(\lambda_{max} + \lambda_{min})^2 \end{aligned}$$

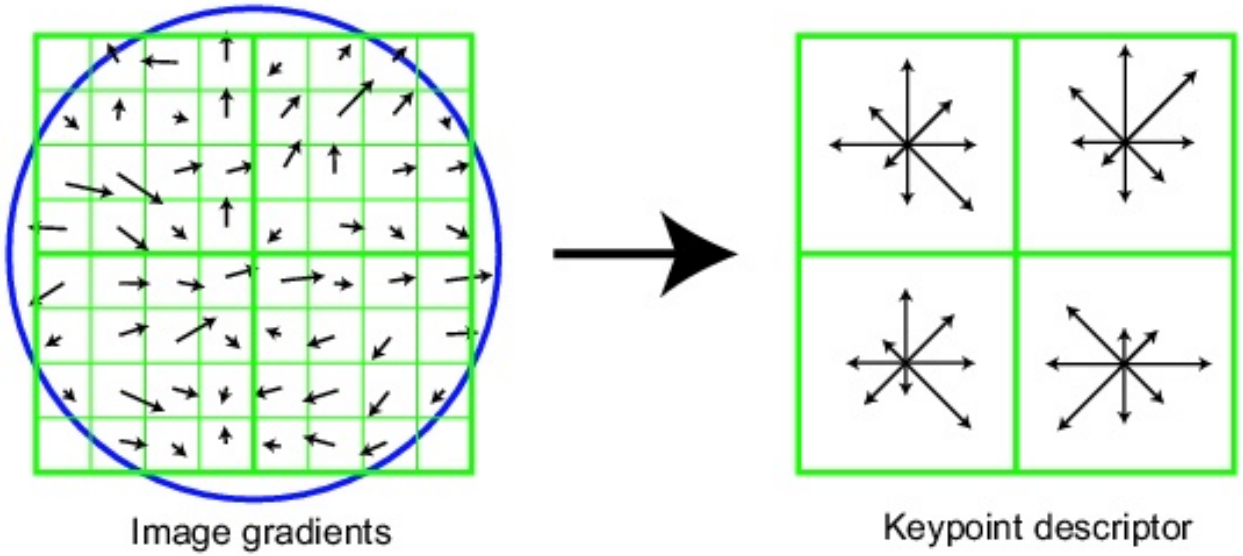
  if $C$ is above a predefined threshold $C_0$.
- Typically $K \approx 0.05$ and $C_0$ depends on image quality.

# SIFT features (detector) [Lowe '99, Lindeberg '94]




- Detected as peaks in Differences of Gaussians.
  - Approximation of Laplacian (blob detector).
- Multiple scales for scale invariance.

# SIFT features (descriptor)

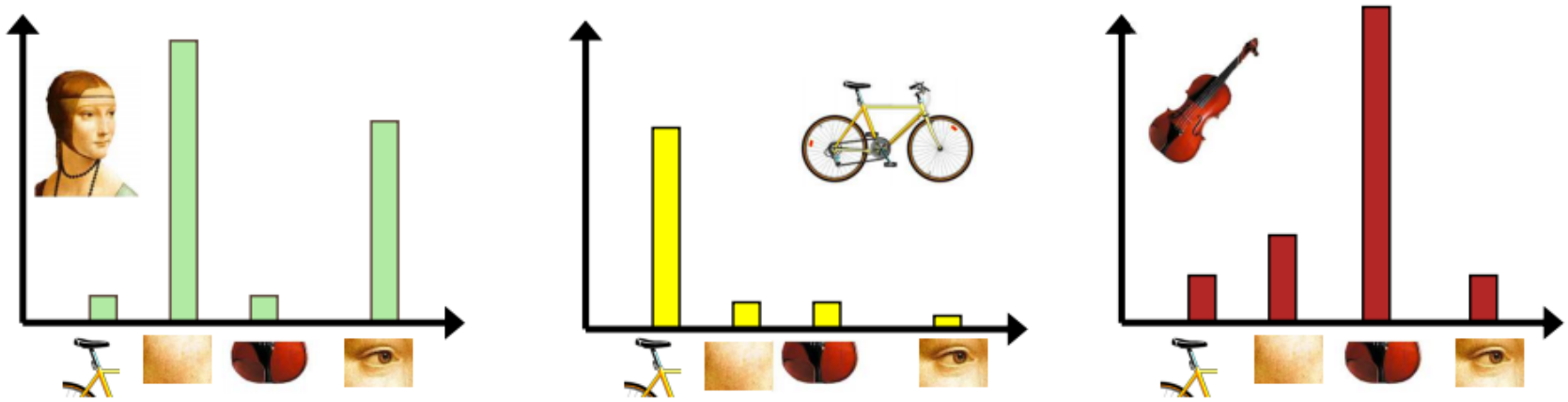



- Descriptors consists of local gradient direction histograms.
- Dominating direction as reference $\Rightarrow$ rotation invariance.
- Window size from scale-space detector $\Rightarrow$ scale invariance.
- Normalize feature vector $v$ to $|v| = 1 \Rightarrow$ luminance invariance.

# SIFT features (matching)


- Match features between image and model.
- Fit a model to remove mismatches (e.g. homography).
- Is there a frog or locomotive in the scene?
  - If enough features are matched the question is Yes.

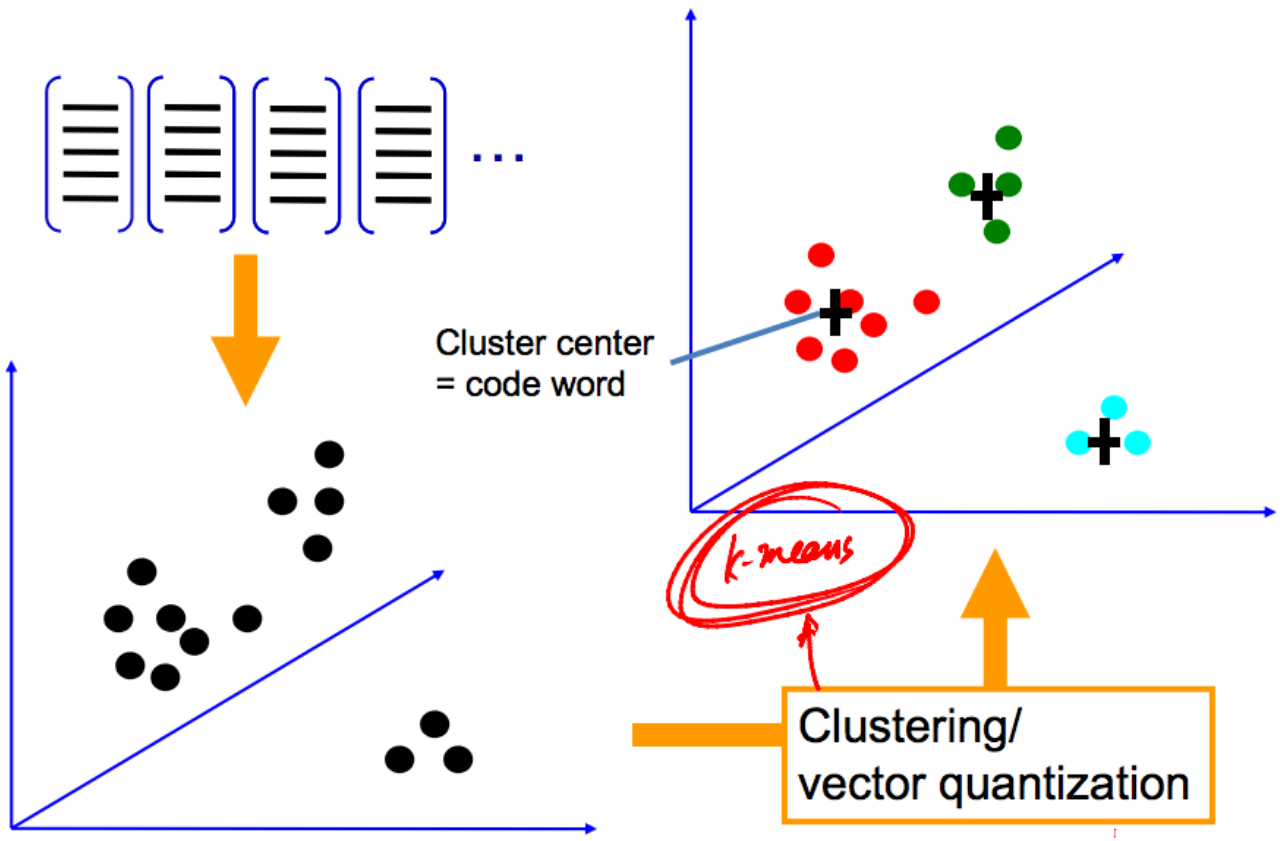
# Bag of Words (BoW): How to go to millions of images?



codewords dictionary

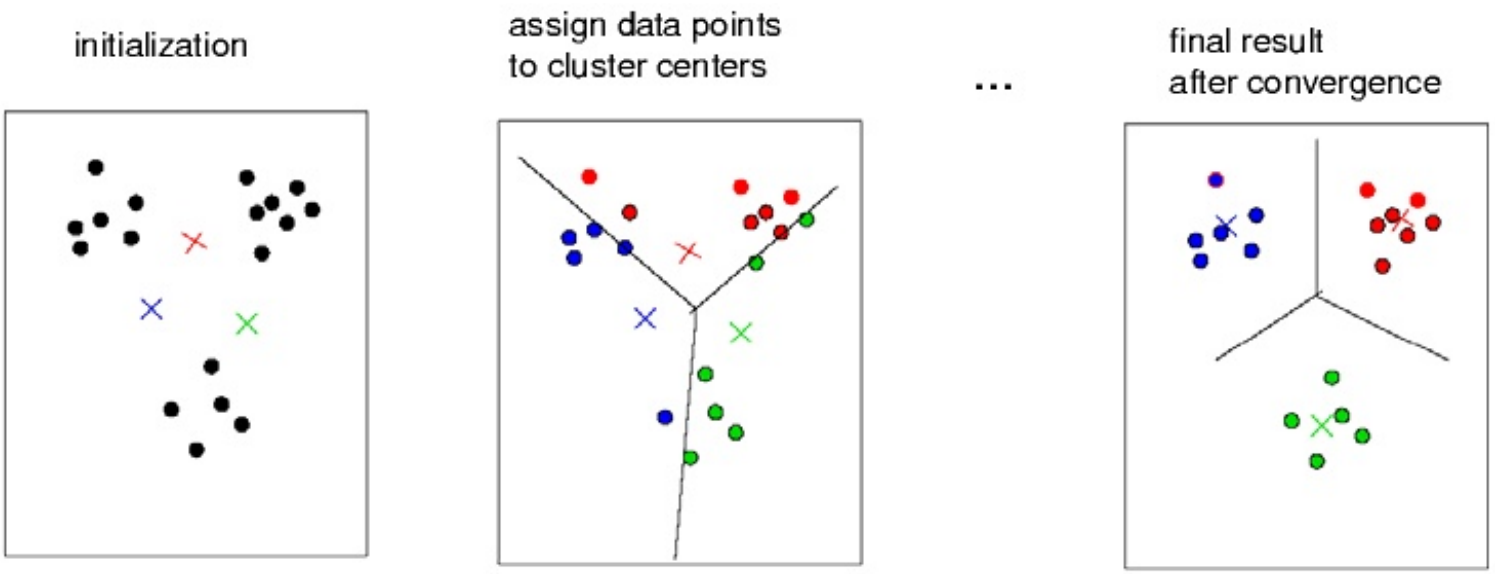
- Idea: represent images as a histograms of features (often SIFT).
- Google: a document is represented as a histogram of words.

# Bag of Words (BoW): Codebook creation



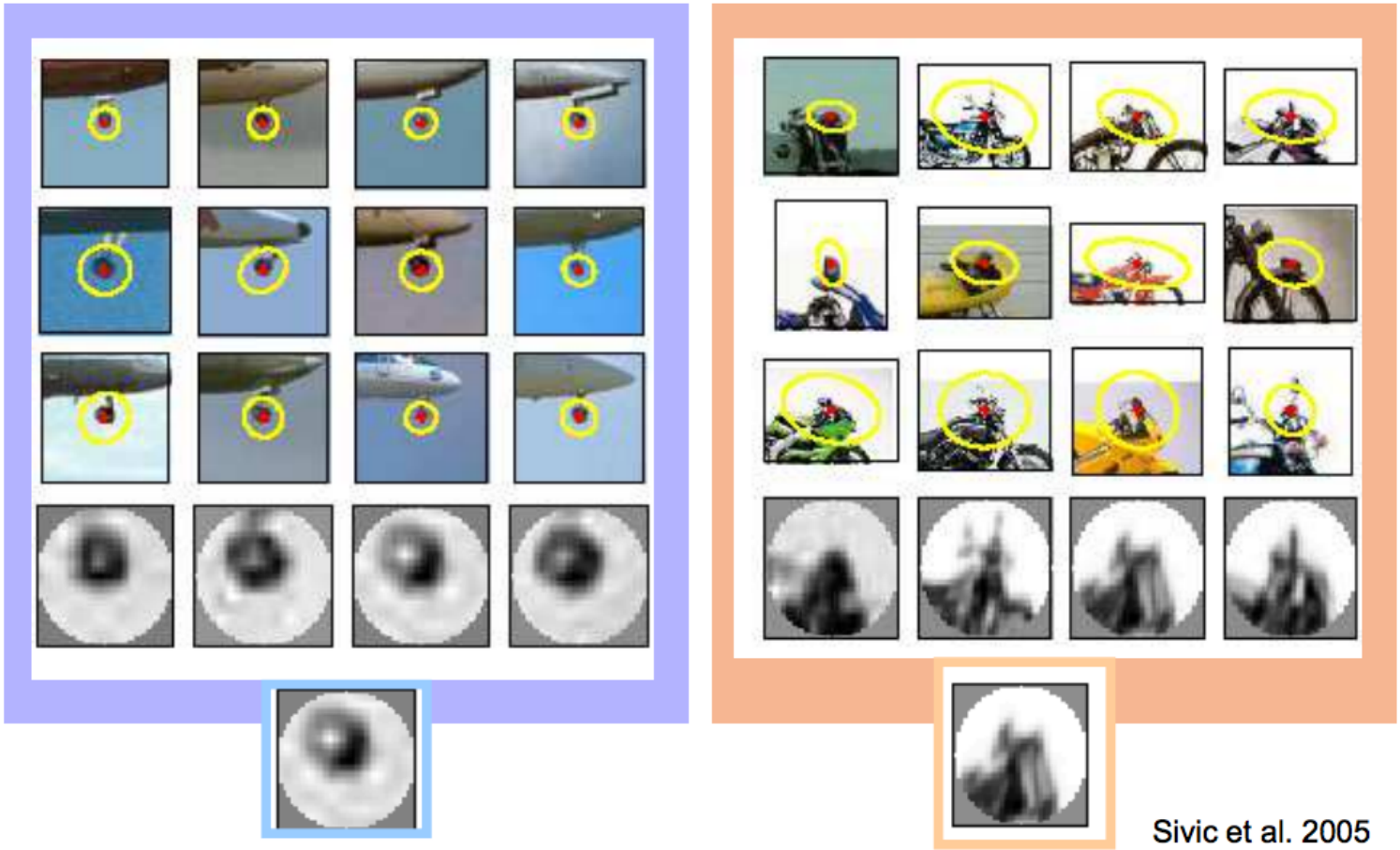

- Goal: Group (SIFT) features into clusters in feature space.
- K-means clustering algorithm is often used in practice:

# K-means clustering (example)



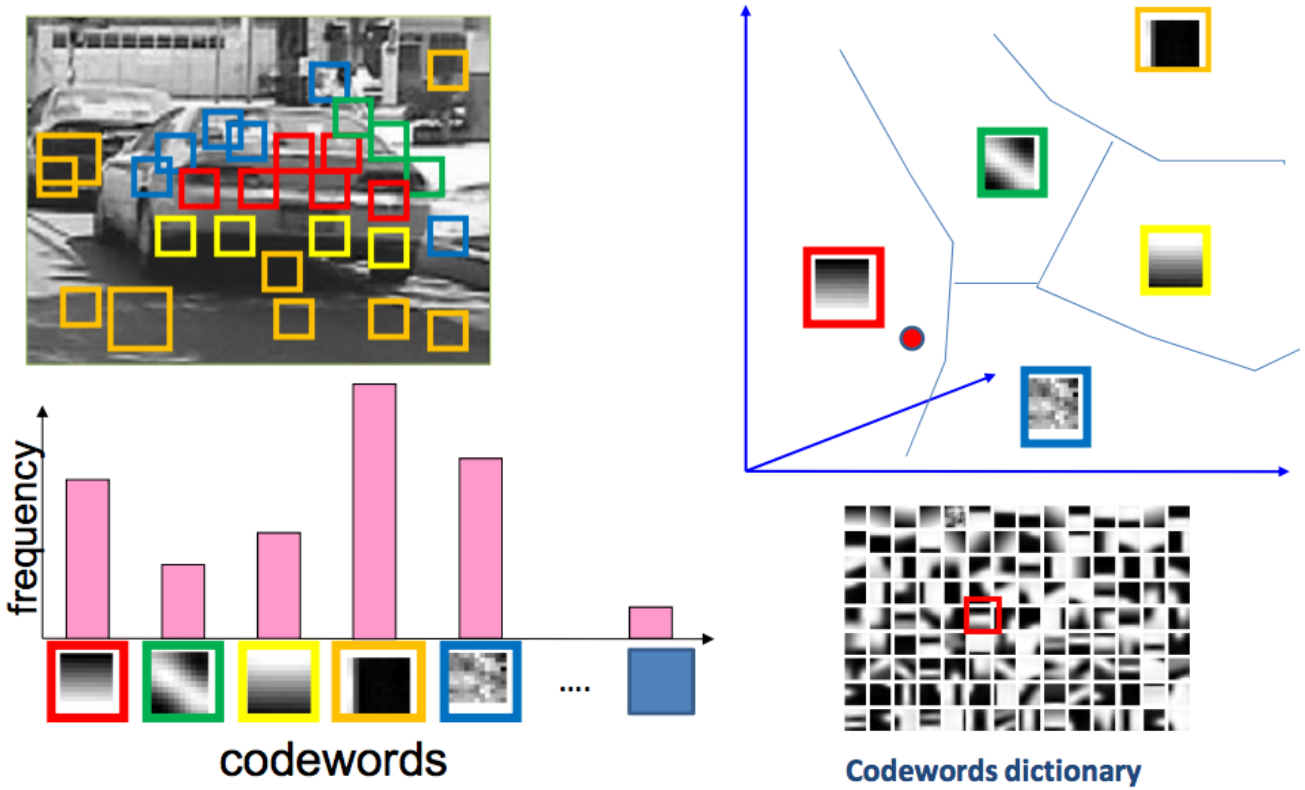

- Problem: Some clusters may have very few features.
- Common solution: Remove clusters that are too small and randomly add new ones.

# Codeword examples




Similar features are matched to the same codeword.

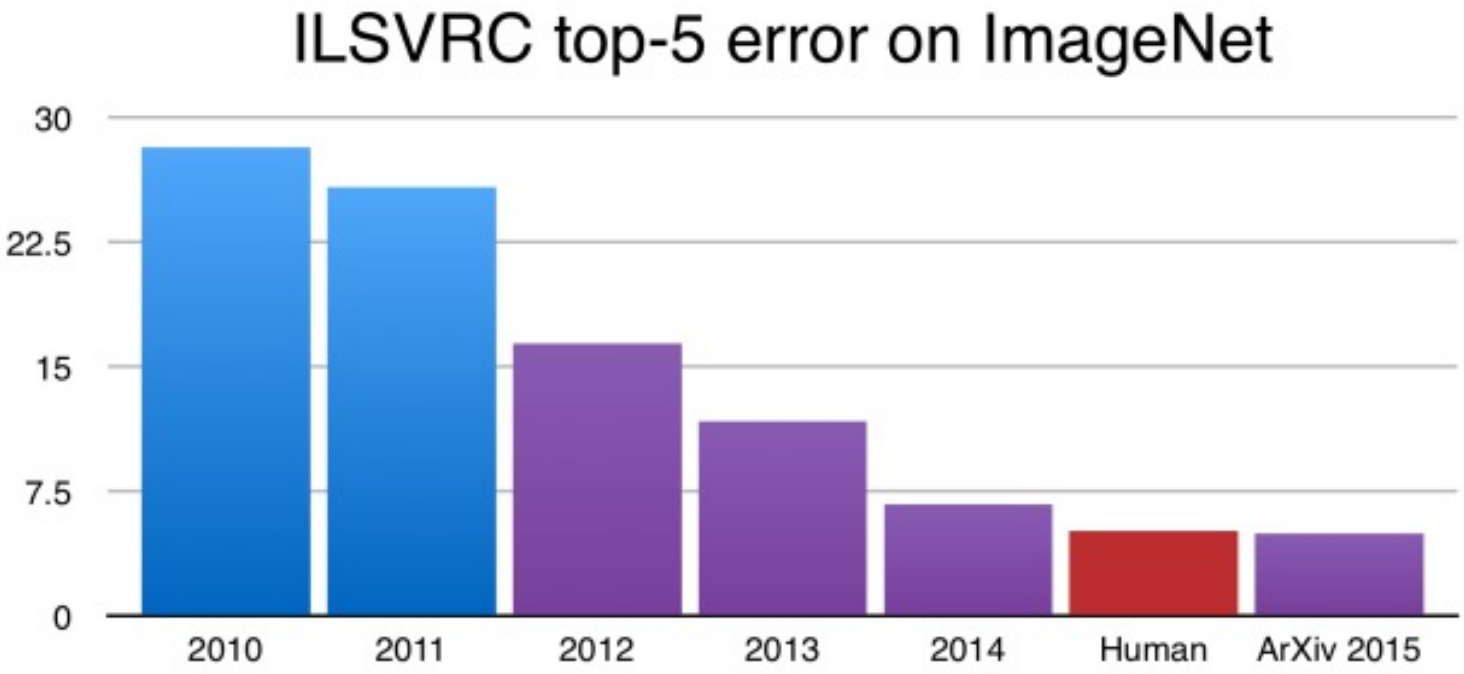
# Bag of Words (BoW): Summary of steps




- Step 1: Feature extraction (e.g. SIFT)
- Step 2: Clustering in feature space (e.g. K-means)
- Step 3: Collect histograms of feature numbers
- Step 4: Classify histograms into different object classes

# Deep Networks for object recognition

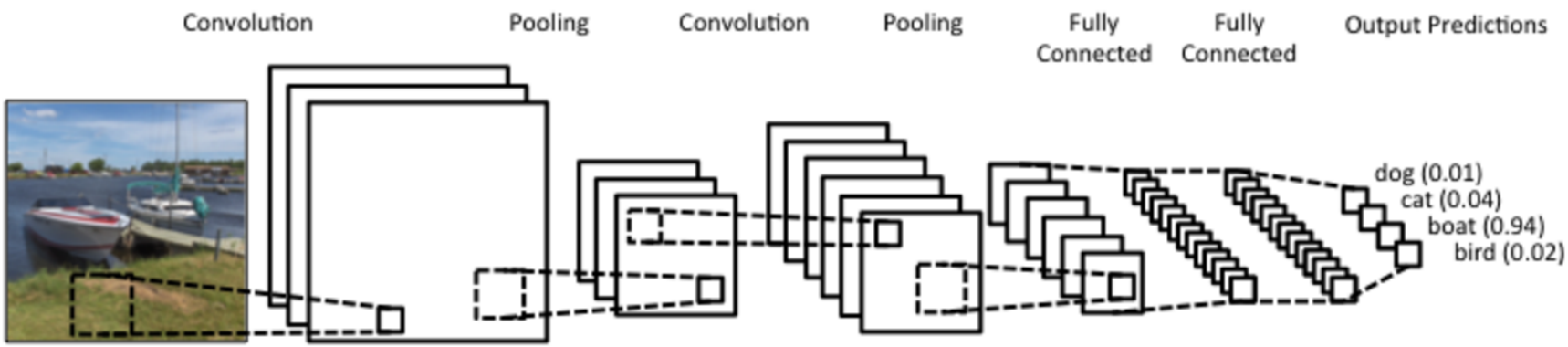
- Neural networks were long forgotten in computer vision.
- Recently, deep neural networks have become state-of-the-art.



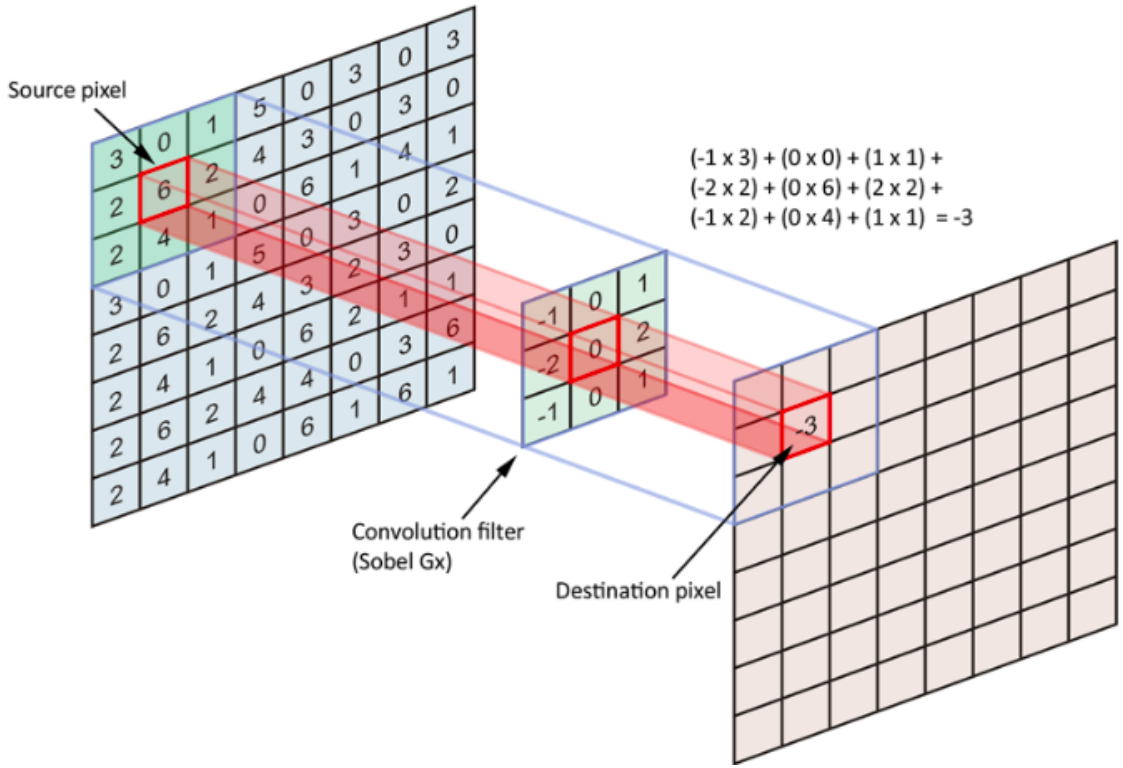

The best methods today have a top-5 error of about 3%.

# Repetition: Convolutional Neural Networks (CNN)

- Idea: have a very simple structure, but learn everything.
- Each layer includes four steps:
  1. Convolutions (normal filtering operations)
  2. Non-linear operator (e.g. ReLU: set negative values to zero)
  3. Pooling (e.g. find local maximum and subsample)
- Last layers are fully connected.

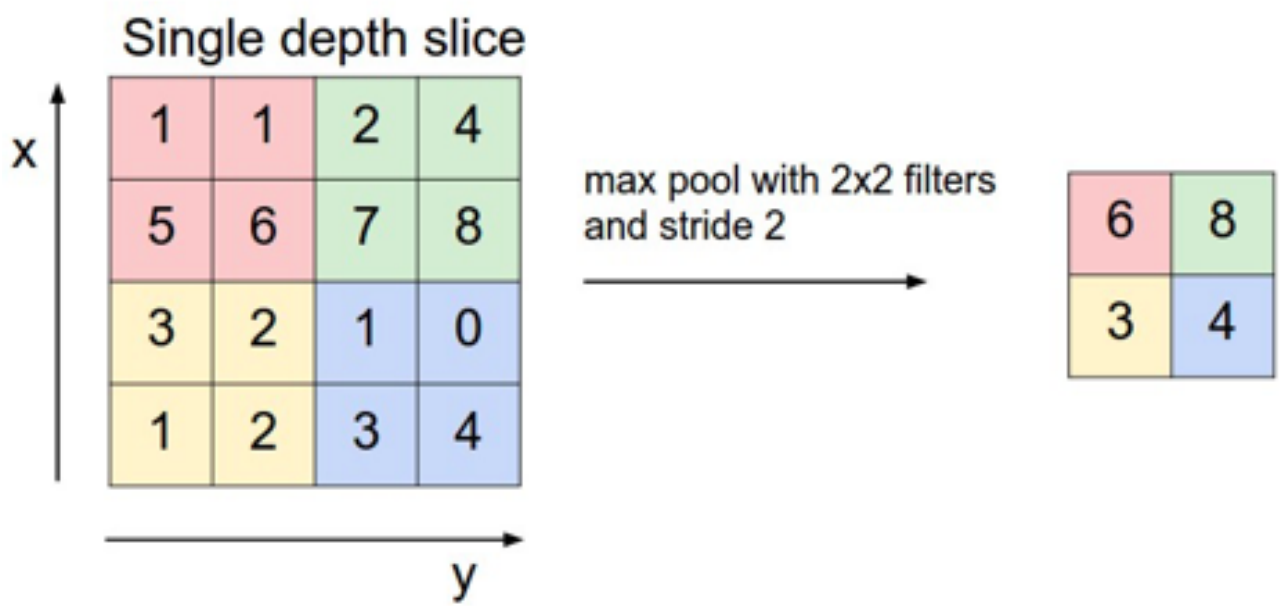


# CNN: convolutional layers



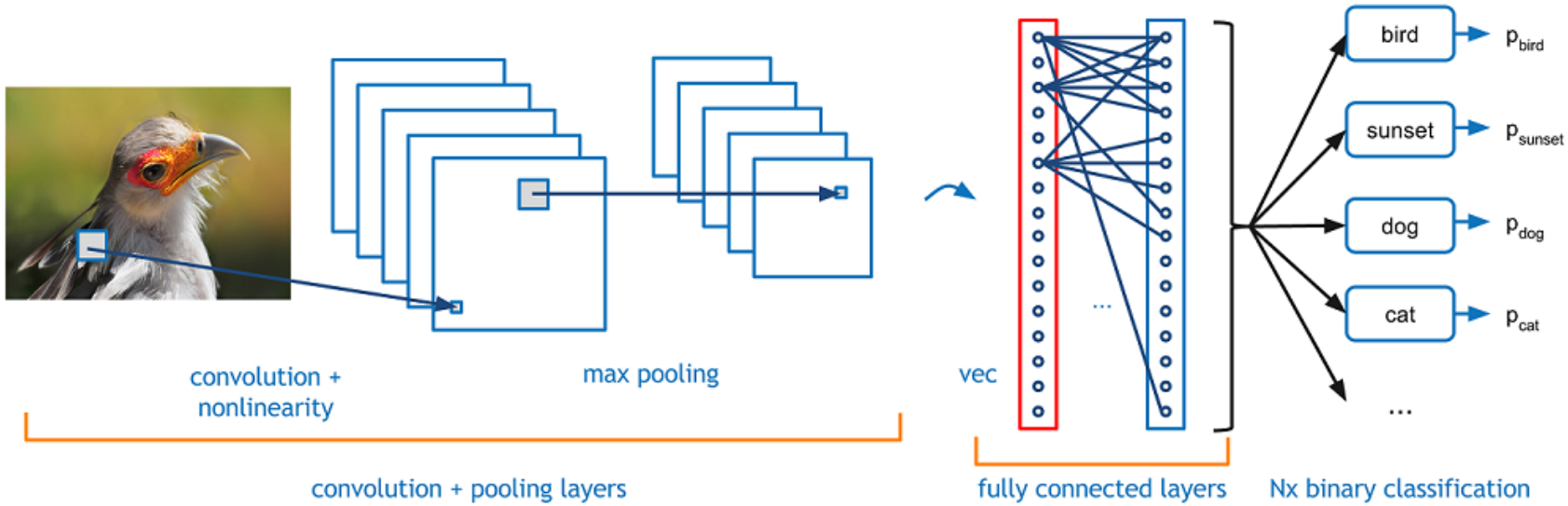

- Similar to the image filters we have seen before.
- However, filters are learned from large amounts of data, and kernels are typically in 3D over multiple input channels.
- Followed by non-linear activation function.

# CNN: pooling layers



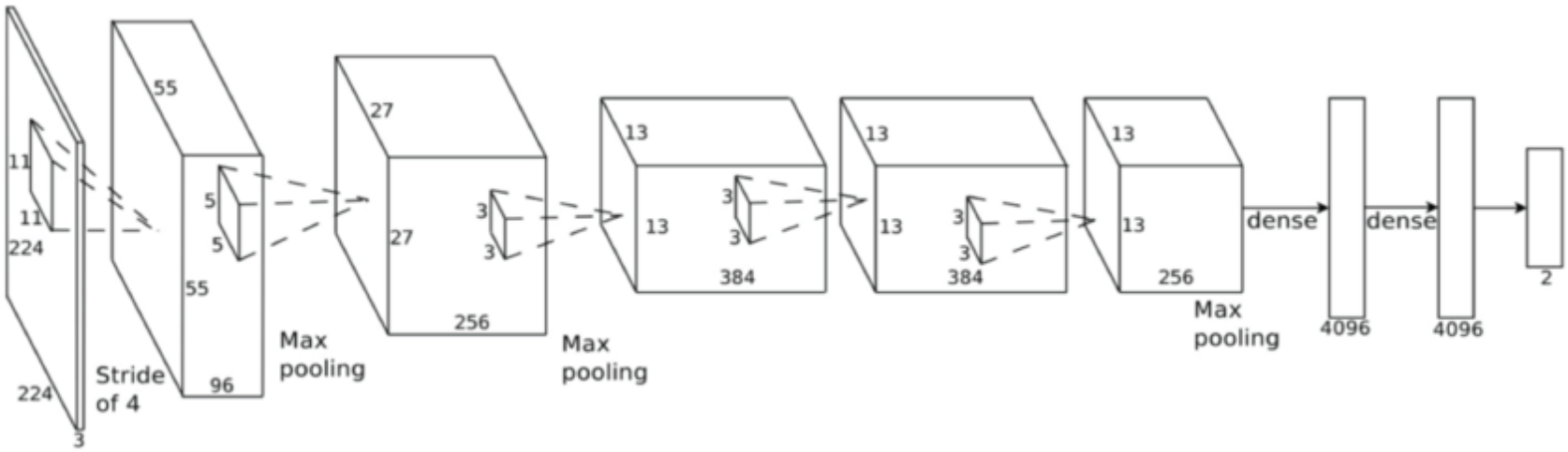

- Pooling reduces image sizes by local maximisation or averaging.
- Gradually makes result more translationally invariant.

# CNN: fully-connected layers




- After flattening data, last layers are fully connected.
- It is where the actual classification is done.
- Feature learning: Train for classification, but use results just before the fully-connected layers for something else.

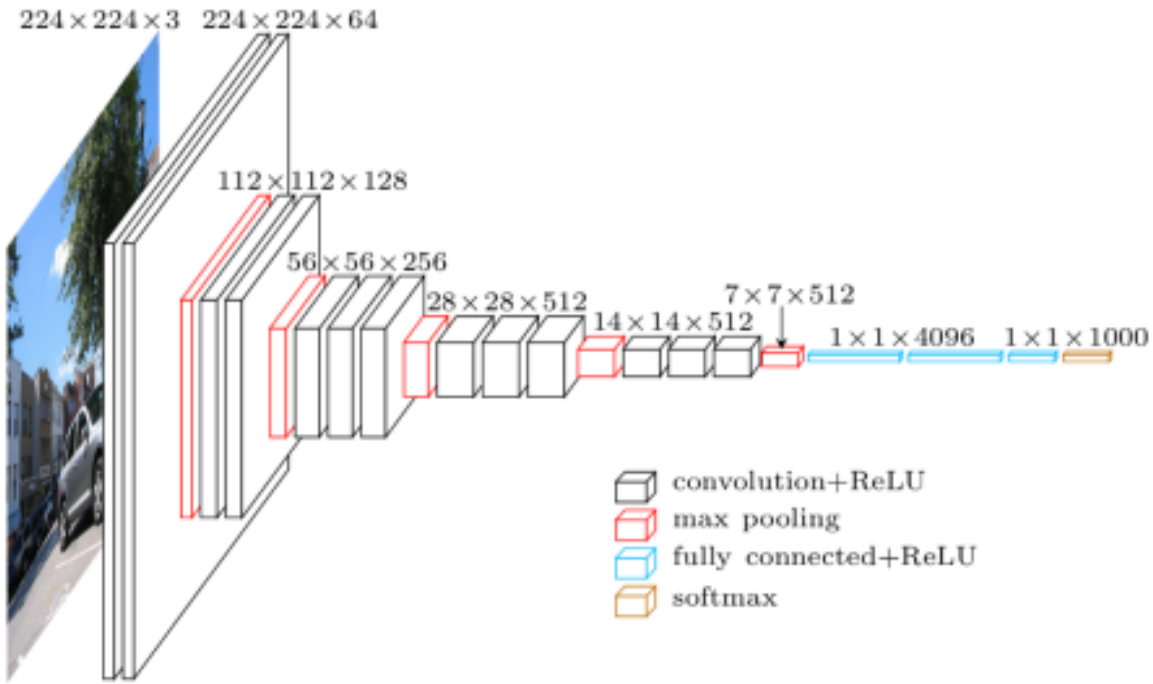
# AlexNet (2012)




- Based on LeNet (LeCun et al, 1998), but revolutionised image classification.
- Break-through: used ReLU for activation and normalization of activation to allow training of deeper networks.

Krizhevsky et al., “ImageNet Classification with Deep Convolutional Neural Networks”, NIPS 2012.

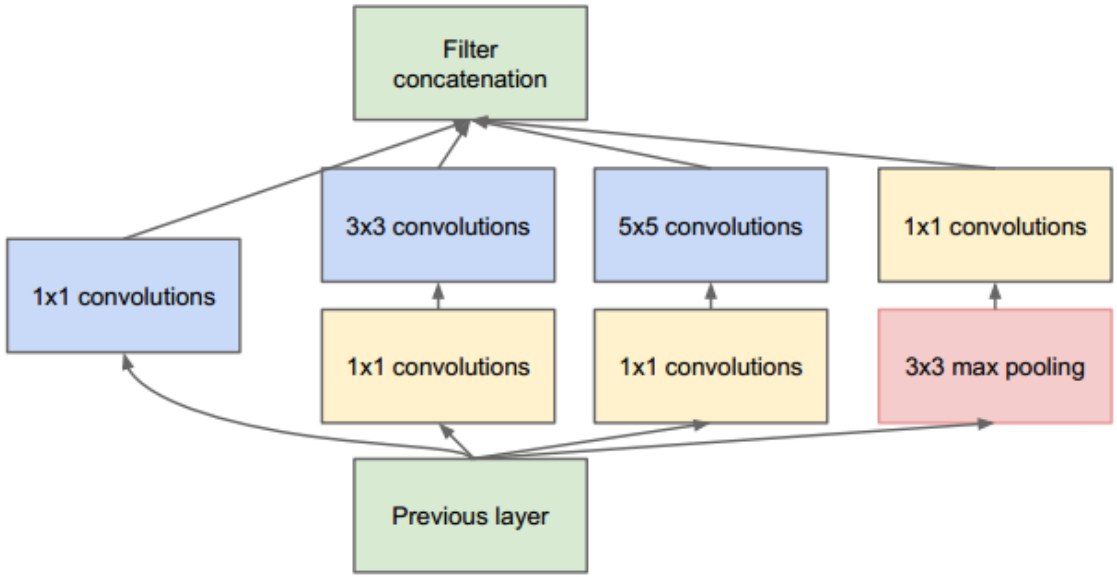
# VGG (2014)




- Novelty: only applied small 3x3 filters, but more layers.
- This is better than fewer layers with larger kernels.
- Often used for feature learning, without training your own network.

Simonyan and Zisserman, “Very Deep Convolutional Networks for Large-Scale Image Recognition”, arXiv 2014.

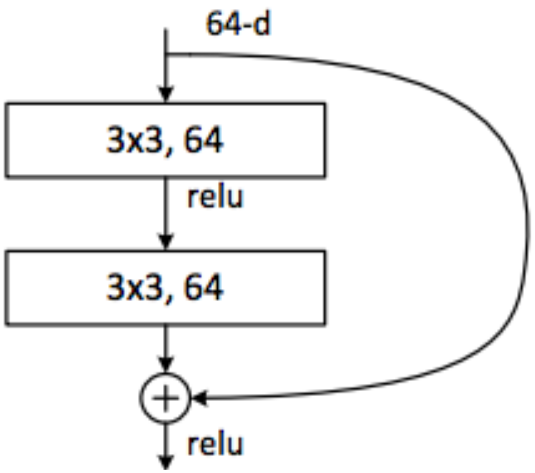
# GoogleNet (2014)

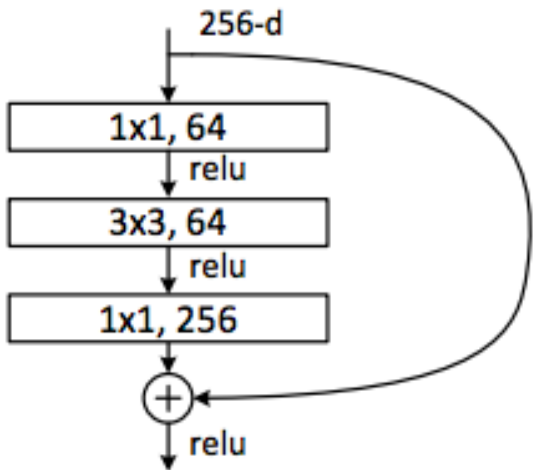



- Novelty: Used sequencies of so called Inception modules.
- 1x1 convolutions are over input channels only, instead of 3D kernels over image space and channels.
- Leads to much fewer parameters to train $\rightarrow$ faster.

Szegedy et al., “Going Deeper with Convolutions”, CVPR 2015.

# ResNet (2015)




- Novelty: Used sequencies of so called Residual blocks.
- Faster propagation of errors during training $\rightarrow$ deeper networks.

He et al., “Deep Residual Learning for Image Recognition”, CVPR 2016.

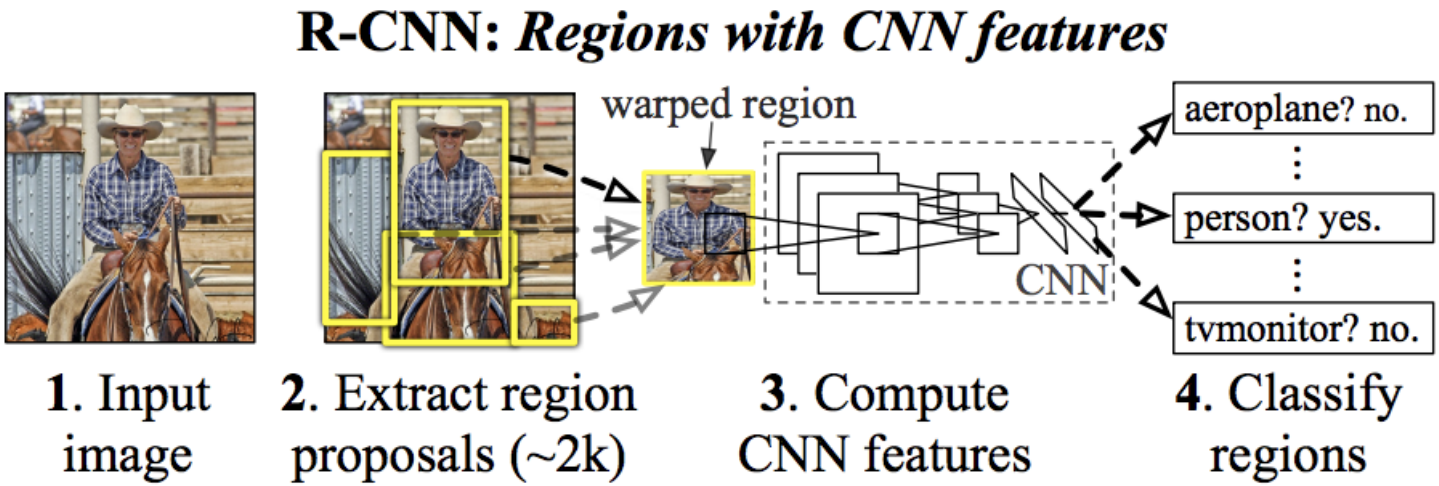
# Results on ImageNet

| Network | Layers | Top-1 error | Top-5 error | Speed | Year |
|---|---|---|---|---|---|
| AlexNet | 8 | 42.9% | 19.8% | 15 ms | 2012 |
| VGG-16 | 16 | 27.0% | 8.8% | 129 ms | 2014 |
| GoogleNet-V1 | 22 | 30.2% | 10.1% | 39 ms | 2014 |
| ResNet-50 | 50 | 24.0% | 7.0% | 104 ms | 2015 |
| ResNet-152 | 152 | 22.2% | 6.2% | 218 ms | 2015 |

In recent years, the focus has been more on object localisation and making networks as small and fast as possible.

# Object detection with CNNs

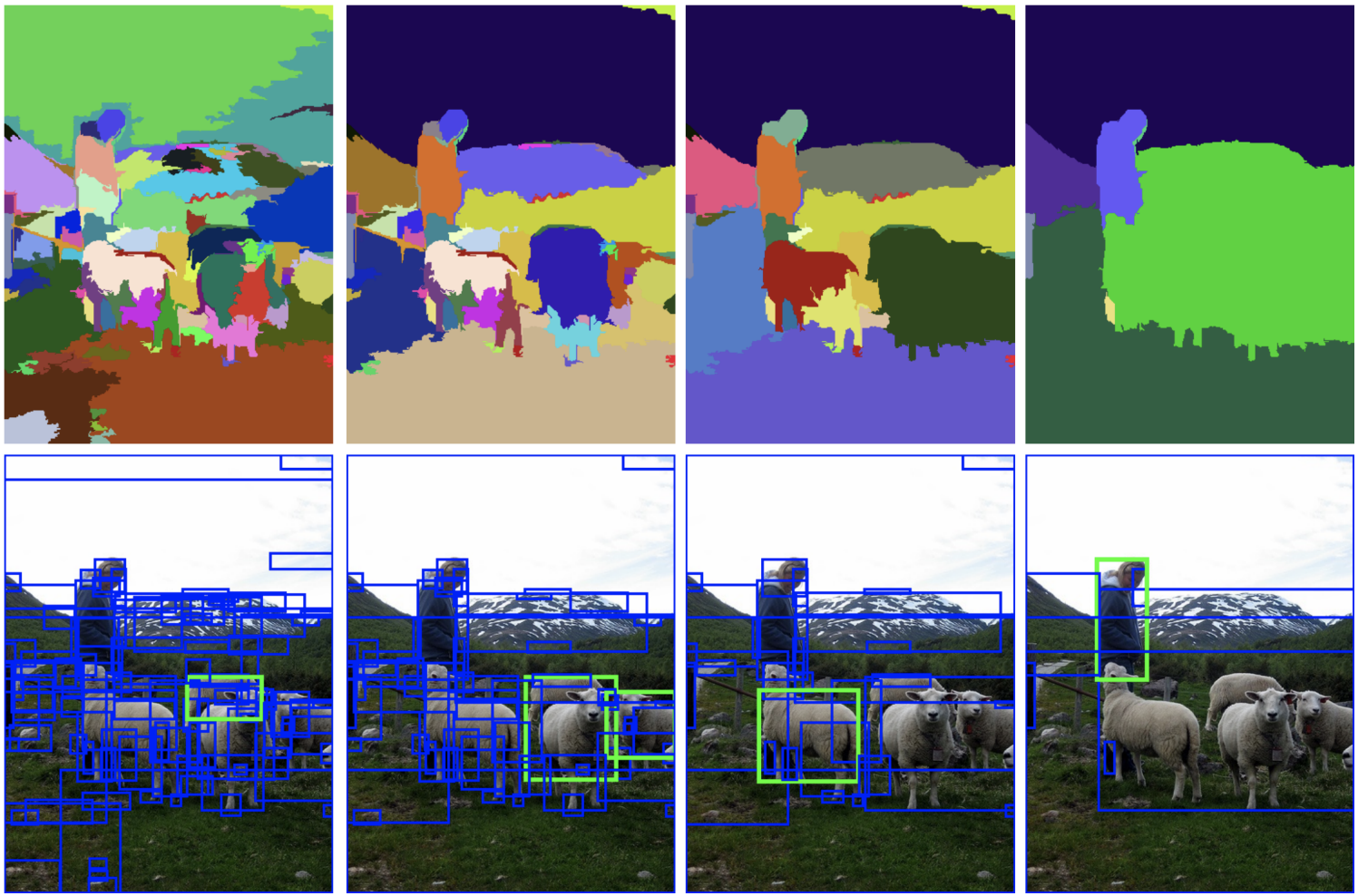
- Much harder problem – first need to find object candidates.




- Slow, since each region is warped and processed by network.

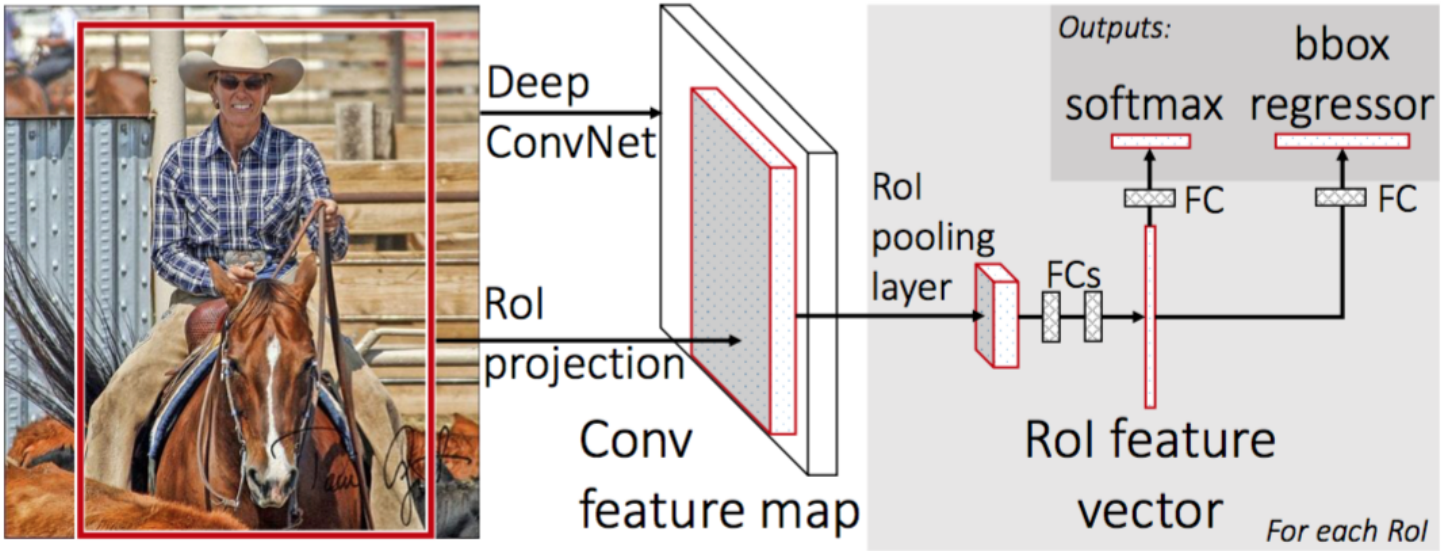
Girshick et al. ”Rich feature hierarchies for accurate object detection and semantic segmentation.”, CVPR 2014.

# Selective search for R-CNN (2014)



Apply Mean-Shift segmentation at different scales and then hierarchically merge regions to get object candidates.

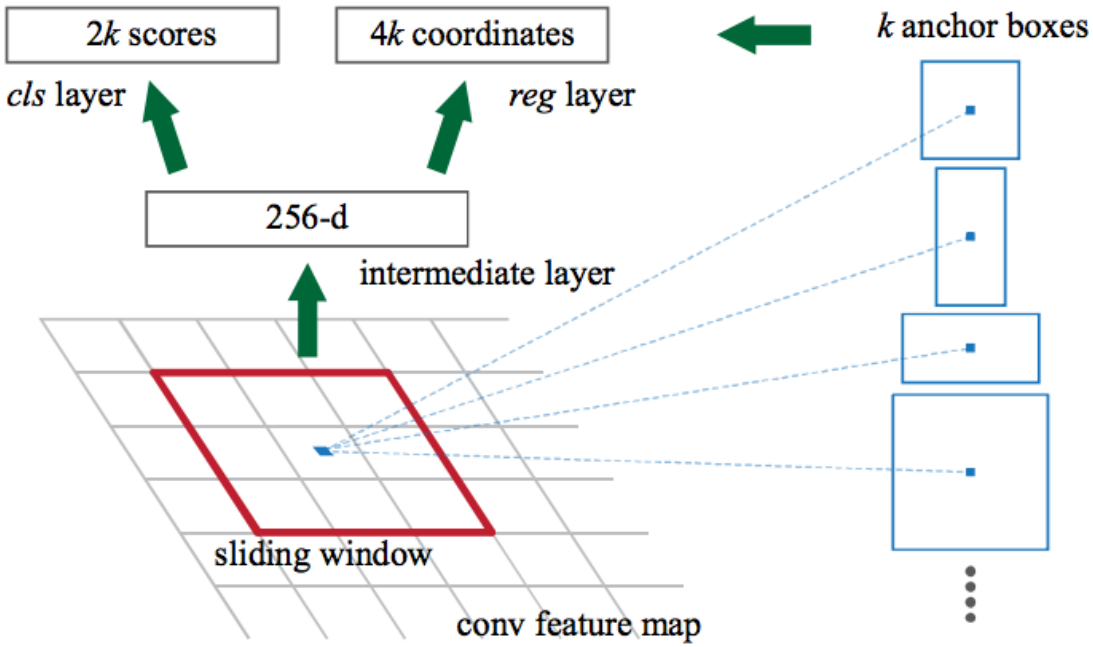
# Fast R-CNN (2015)




- Novelty: do convolutions only on the full image.
- With pooling, apply region proposals directly to feature space.
- Refine object by a bounding box regressor.

Girshick, ”Fast R-CNN”, ICCV 2015.

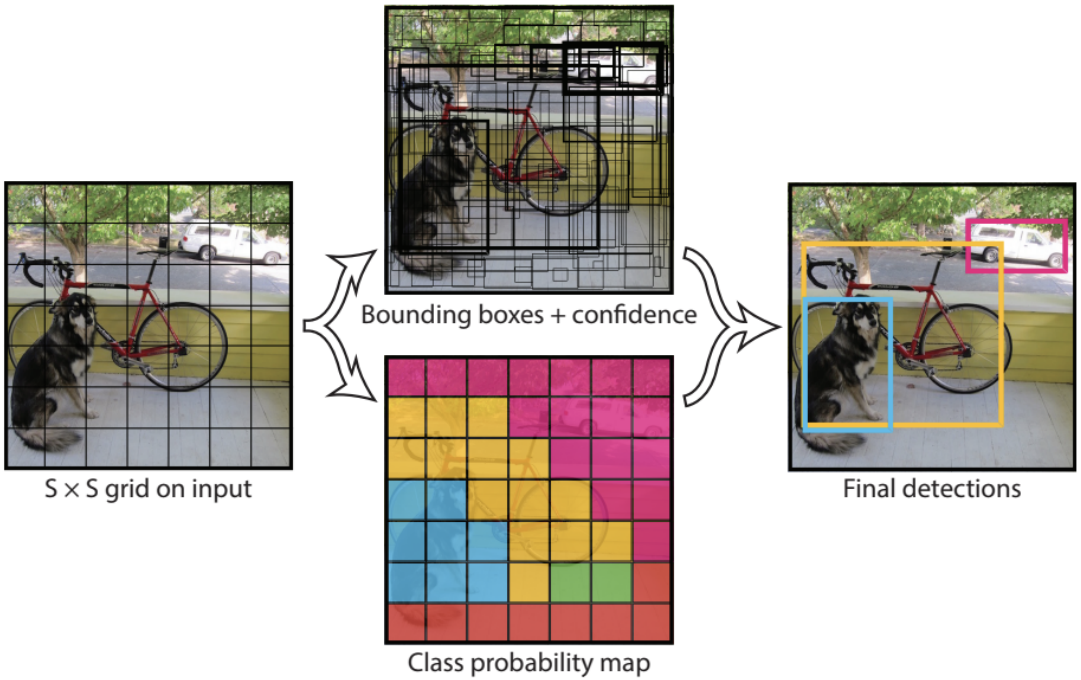
# Faster R-CNN (2015)




- Novelty: replace selective search (Mean-shift) with region proposal network (CNN).
- Use anchor boxes of different shapes and sizes.
- Each anchor outputs: class scores + refined bounding box shape.

Ren et al. ”Faster R-CNN: Towards real-time object detection with region proposal networks.”, NIPS 2015.

# YOLO – You only look once (2016)




- Novelty: decouple class score from bounding box prediction.
- Combine generation and verification in one pass $\rightarrow$ much faster
- Uses fewer anchor points, leads to problems with small objects.

Redmon et al. ”YOLO: Unified, real-time object detection”, CVPR 2016.

# Object detection precision and speed

Results from one particular benchmark

| Method | Accuracy (mAP) | Speed (s) | Year |
|---|---|---|---|
| R-CNN | 68.4 | 49 | 2014 |
| Fast R-CNN | 70.0 | 2.3 | 2015 |
| Faster R-CNN | 76.4 | 0.20 | 2015 |
| YOLO | 63.4 | 0.022 | 2016 |

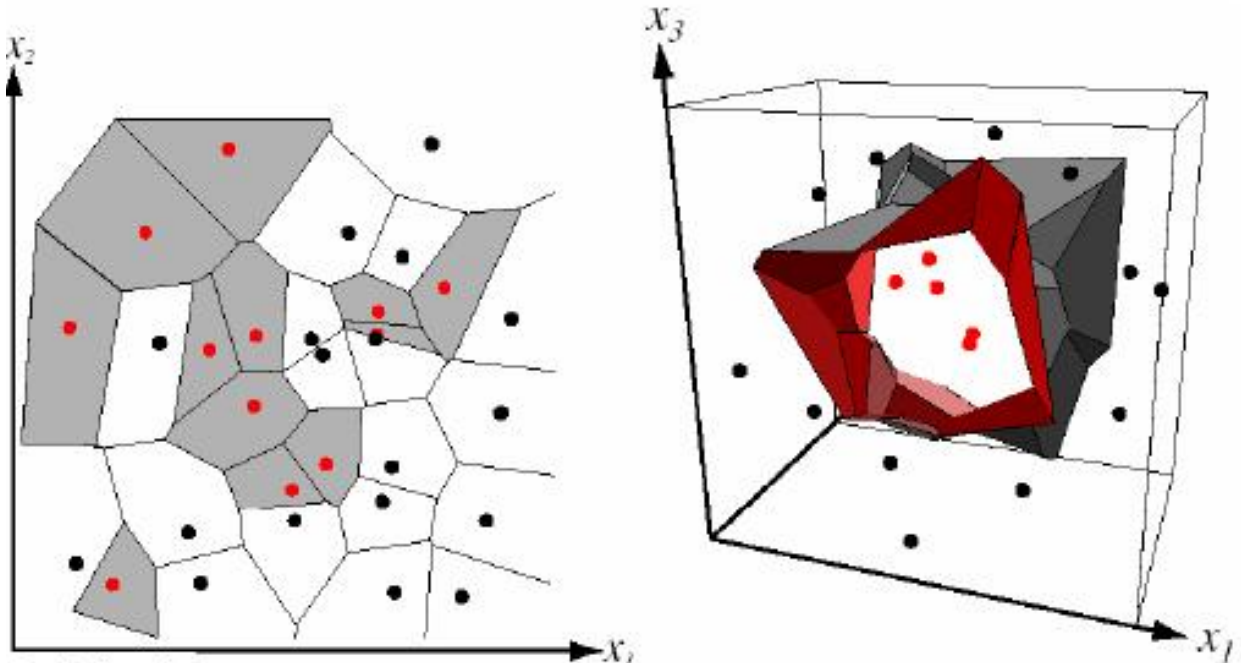
In recent years, YOLO and similar methods (SSD, RetinaNet, etc.) have improved considerably in mAP (mean average precision).

# Basic methods for classification

How to determine which class a feature vector belongs to?

- Nearest neighbour classification
- Bayesian classification
- Linear discriminant functions

# Nearest neighbour classifier




- Straight-forward (model-less) method: given a test image, find the nearest neighbour among all training examples.
- Problem: Number of training examples can be very large and search for nearest neighbour too costly.

# Nearest neighbour classification

- Measure feature vectors for a representative selection of images from known classes.
- Given a feature vector, let the classification be the class index of the nearest representative in the scatter diagram.

\+ Works for well separated classes.

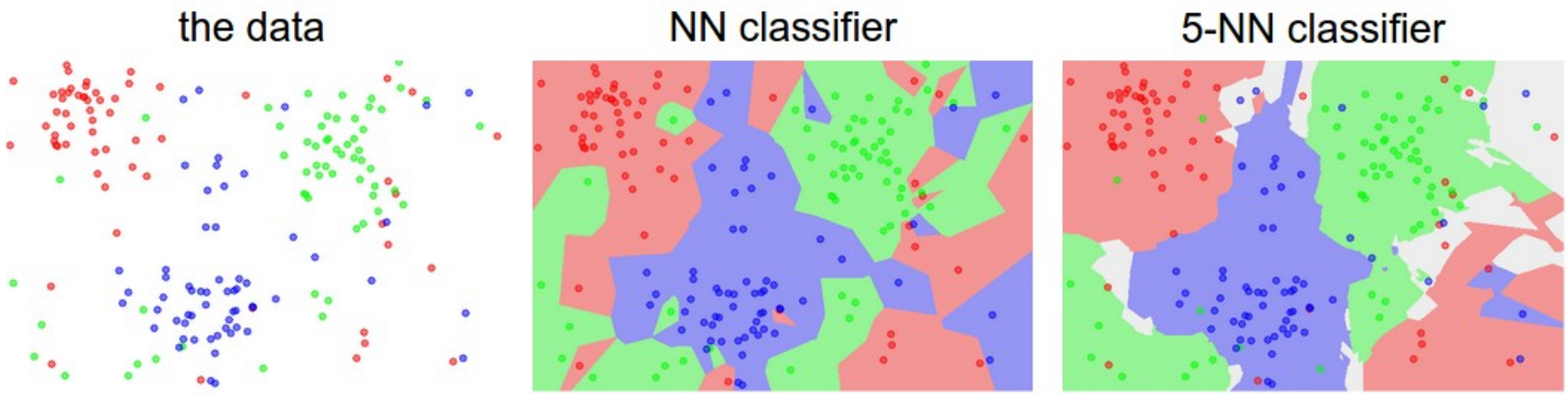
\+ Can represent clusters with complex shape.

\- May require complex computations in higher dimensions.

\- Depends on choice of metrics.

\- Highly sensitive to outliers (no suppression).

# K-Nearest neighbour classification




- Pick the class with most votes among K-nearest neighbours.
- More robust to invididual weird samples (outliers).

# Bayesian classification

- Idea: consider the class assignment and feature vectors as stochastic variables. Determine a classification function that minimizes the classification error.
- If we know the prior probability of the class, $p(k)$,
  and distribution of feature vector for class $k$, $p(z \mid k)$,
  we know class probability given the feature vector, $p(k \mid z)$.

$$p(z,k) = p(z \mid k)\, p(k) = p(k \mid z)\, p(z) \Rightarrow$$

$$p(k \mid z) = \frac{p(z \mid k)\, p(k)}{p(z)} = \frac{p(z \mid k)\, p(k)}{\sum_i p(z \mid k_i)\, p(k_i)} \quad \text{(Bayes' formula)}$$

# Bayesian classification (continue)

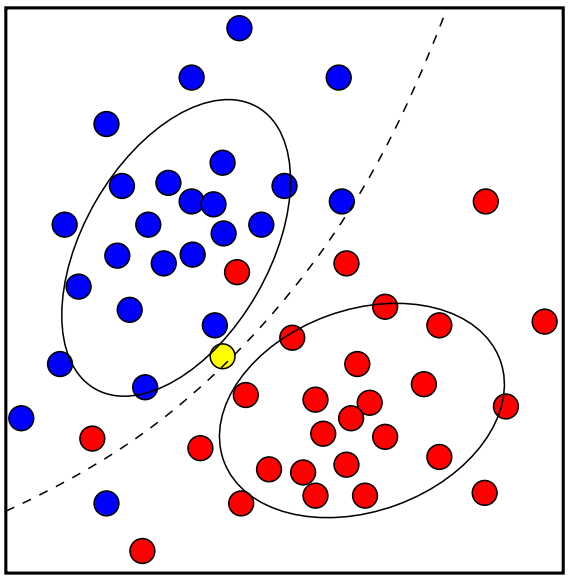
Choose the class $k$ that maximizes $p(k \mid z)$.

- Minimizes the probability of wrong classification, IF statistical model is correct.
- Given most probable class, if prior probabilities $p(k)$ are known.
- Common assumption: All $p(k)$ equal $\Rightarrow$ Maximum likelihood

# Assuming 2D Gaussian distributions

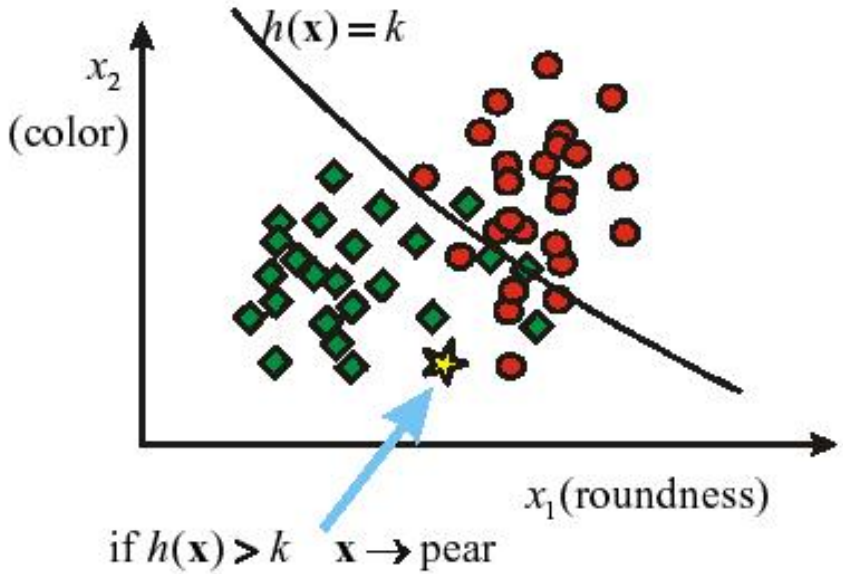
$$p(\mathbf{z} \mid k_i) = \frac{1}{(2\pi)^{D/2} \mid \Sigma_i \mid^{1/2}} \exp\left\{ -\frac{1}{2}(\mathbf{z} - \mu_\mathbf{i})^\top \Sigma_i^{-1} (\mathbf{z} - \mu_\mathbf{i}) \right\}$$

$$p(k_1 \mid \mathbf{z}) = \frac{p(\mathbf{z} \mid k_1)\, p(k_1)}{p(\mathbf{z} \mid k_1)\, p(k_1) + p(\mathbf{z} \mid k_2)\, p(k_2)}$$



Decision boundary: $p(\mathbf{z} \mid k_1)\, p(k_1) = p(\mathbf{z} \mid k_2)\, p(k_2)$

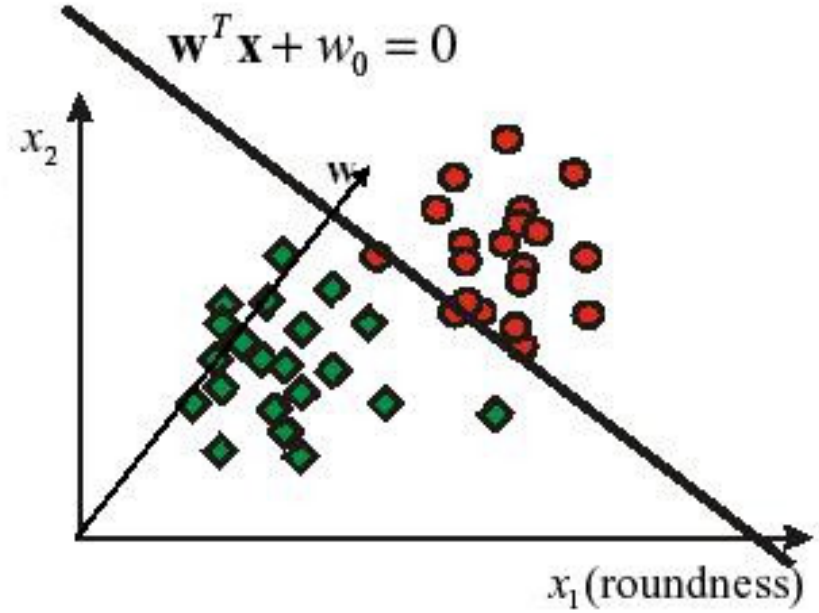
# Discriminant functions




Instead of modelling the distributions and then find a decision function, find a decision function and its boundary directly.

1. Choose class of decision function.
2. Estimate parameters of this function from training data.
3. Classify a new point based on the decision function.

# Linear Discriminant Functions




- Problem: Finding the best $\mathbf{v} = (\mathbf{w}, w_o)$ given training examples.
- Most modern machine learning methods (Boosting, SVM, etc) use combinations of many linear discriminant functions.
- In fact, the last fully-connected layers of a CNN can be seens as a combination of linear discriminant functions.

# Summary of good questions

- What is the difference between recognition and classification?
- What makes a good feature space for recognition?
- What kind of invariances do you often want in recognition?
- What classes of recognition methods exist and what are their differences?
- What does a typical feature based method consist of?
- What steps does a Bag of Words approach include?
- What characteristics does a nearest neighbour classifier have?
- How do you find a decision boundary for Bayesian classification?

# Readings

- Gonzalez & Woods: Chapter 12.1 – 12.2.2
- Szeliski: Chapter 14