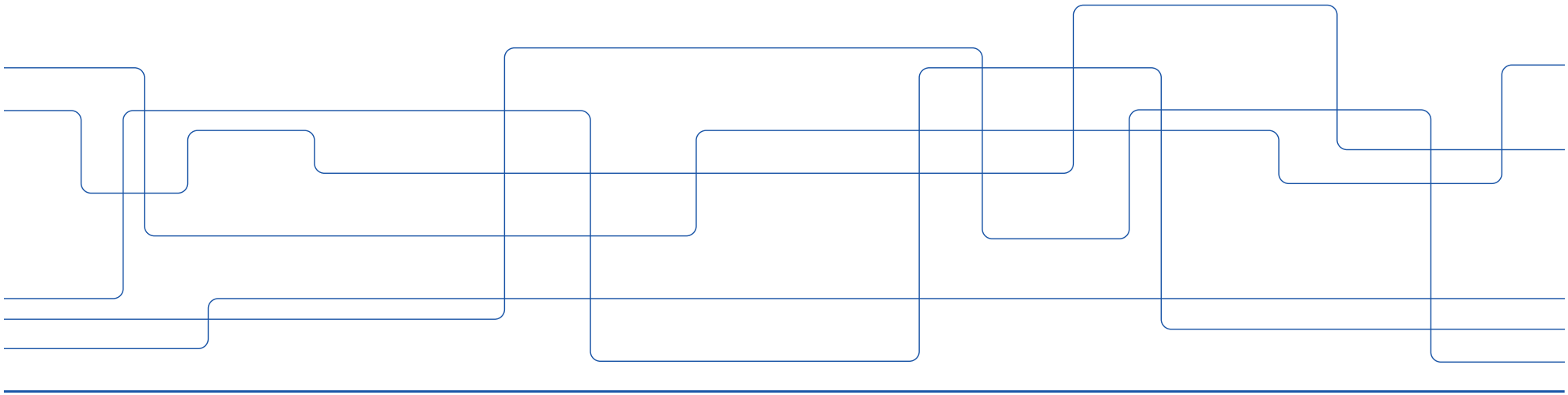
Object recognition

Mårten Björkman



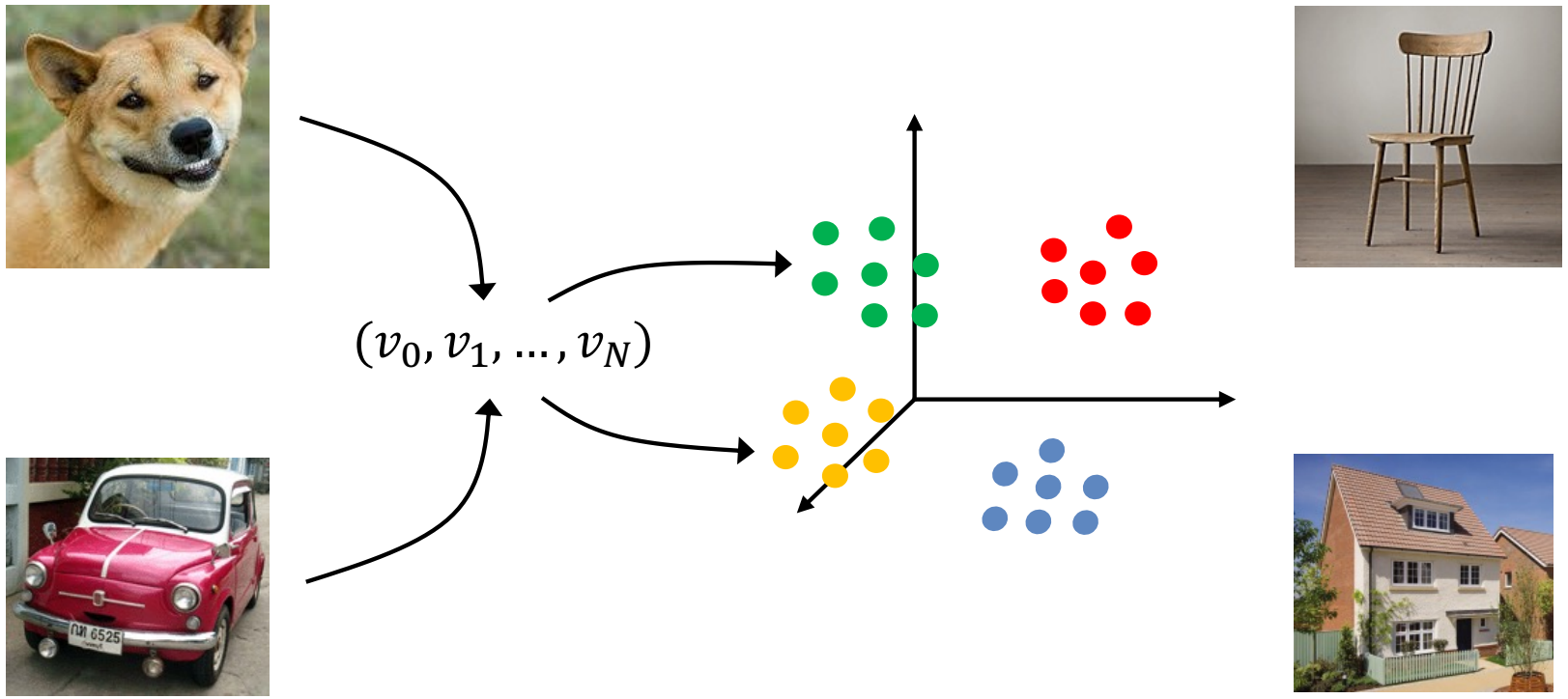
What are we talking about?

- Recognition - Is this my cup?
  - Only requires a model of one cup.
  - What makes the individual cup unique?
  - Classical matching methods often work very well.
- Classification - What is this?
  - Requires a model that captures all cups.
  - How to group objects into classes?
  - Deep learning-based methods works very well.
- Detection - If there is a cup, where is it?
  - A much harder problem.

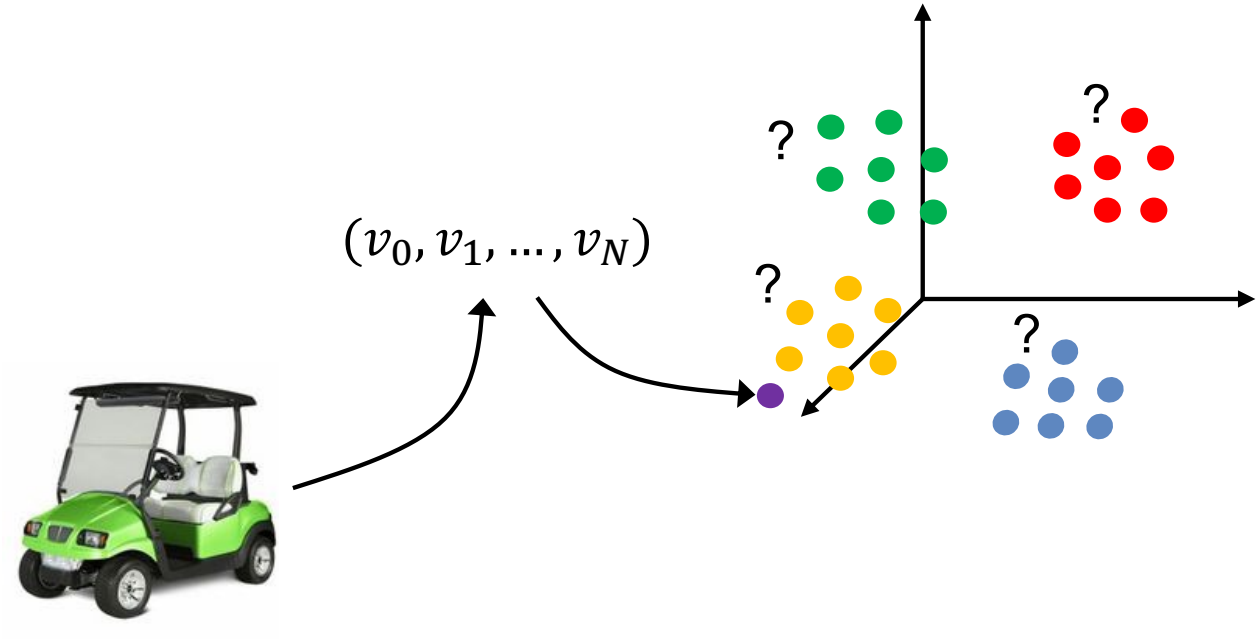
Common for all methods

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

Recognition (Step 1: representation)



Recognition (Step 2: classification)



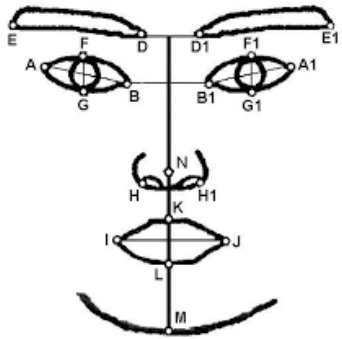
- Given a new image in feature space, determine the class it belongs to.

Differences between methods

- Supervised vs unsupervised
  - Supervised: annotated training examples exist
  - Unsupervised (clustering): no annotated training examples
  - Semi-supervised: some annotated examples, most are not
- 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



Invariances

- The appearance of an 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

What remains the same, but what does not?

Object occlusions

Many objects are covered by other objects.

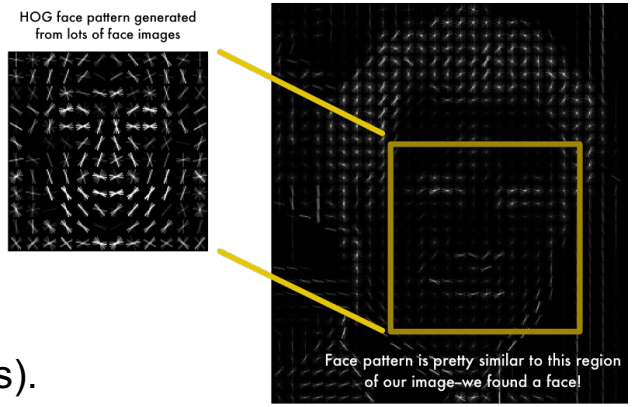
Deformations

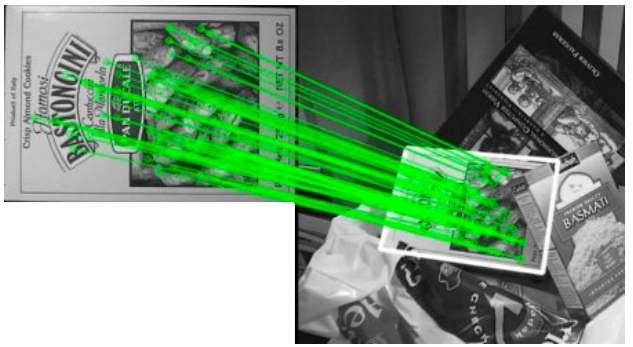
Viewpoint invariance

Some views are more common that others. Often leads to a bias.

Traditional methods of recognizing 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 can be very 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.



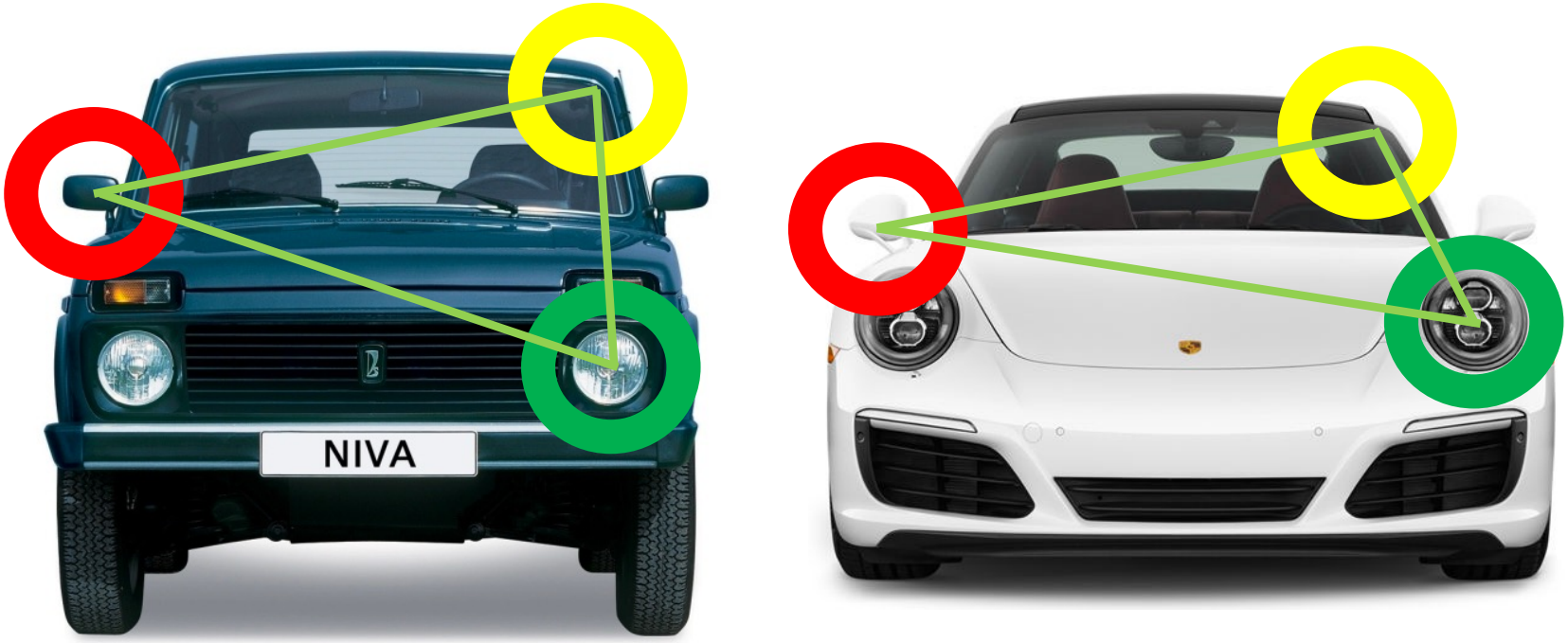


Feature based methods

Methods typically contain:

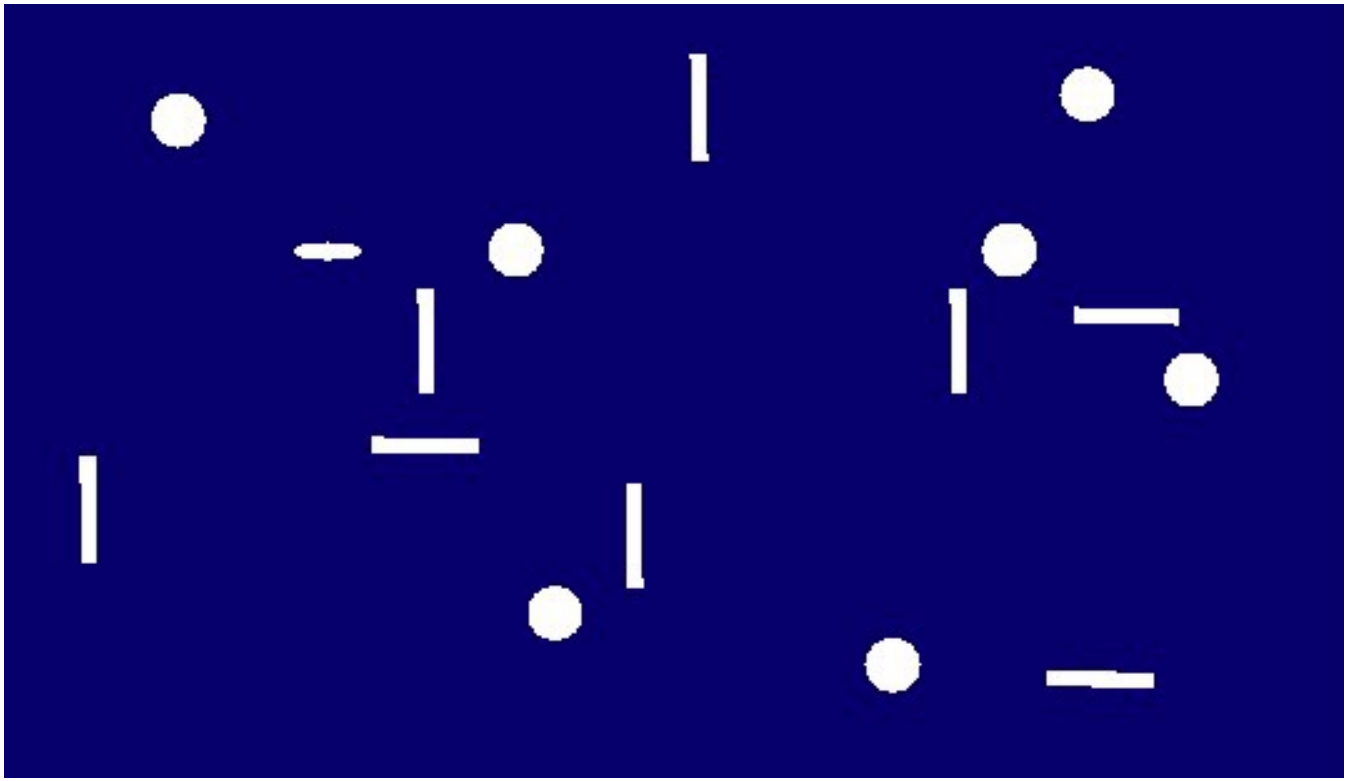
- Some feature detector
  - High curvature points (corners)
  - Scale-space blobs (Laplace, DoG)
- Some invariant descriptor(s)
  - Templates (small windows around features)
  - Statistics (edges, colours, etc.)
  - Combinations of templates and statistics (SIFT)
- Some way of combining matching features
  - Voting or similar combination measure
  - Matches often used to find a transformation (homography)

Recognition by parts



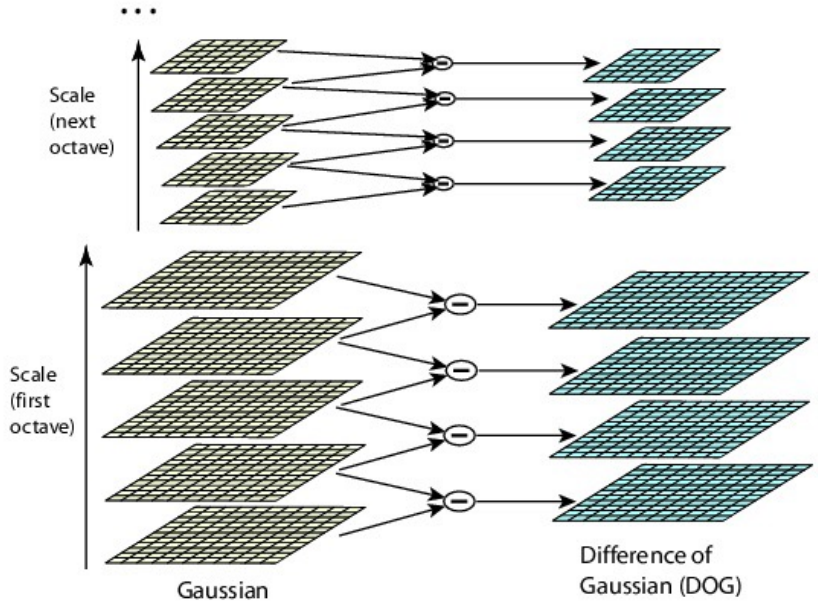
Each part might look like many other parts, but combined they might strengthen a hypothesis.

Importance of feature grouping



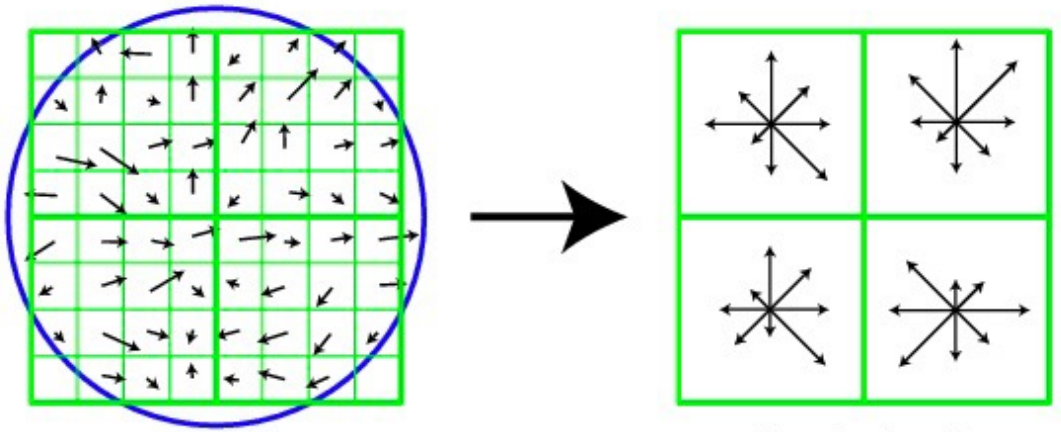
Can you see the face?

Repetition: SIFT features (detector)



- Feature positions detected as peaks in Differences of Gaussians (DoG).
  - A faster approximation of Laplacian (blob detector).
- Multiple scales allow features to be detected regardless of projection size.

Repetition: SIFT features (descriptor)

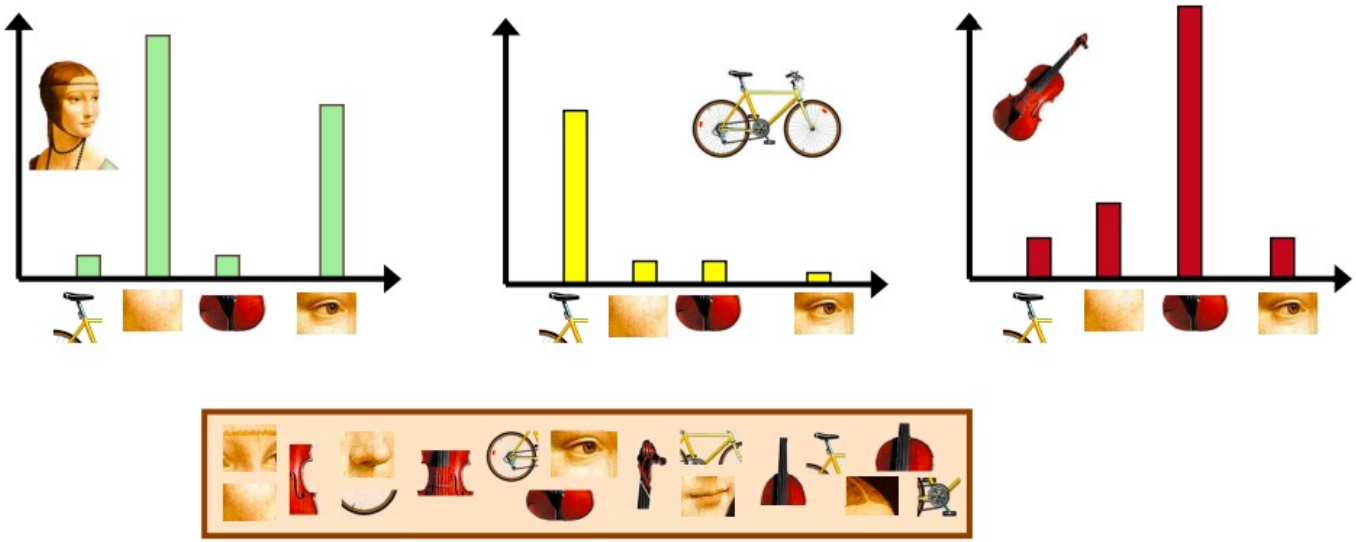


- Descriptors consists of histograms of local gradient directions.
- Dominating direction used to orient local window ⟹ rotation invariance.
- Detected scale used as the size of local window ⟹ scale invariance.
- Normalize feature vector v to $[v] = 1$ ⟹ luminance invariance.

SIFT features (matching)

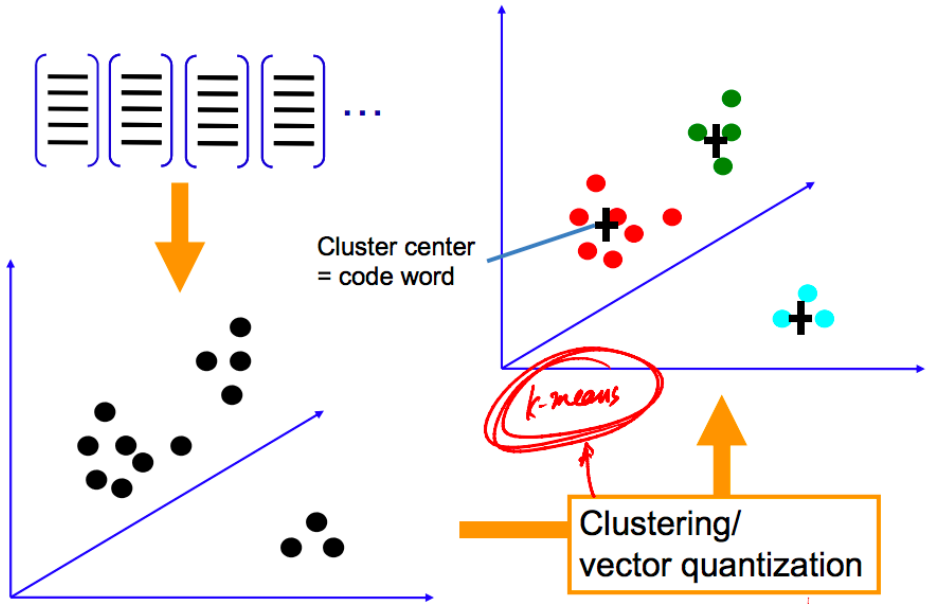
- For each feature v_i^1 find the best matching feature v_j^2 in the other image.
  - If the two feature are each others favourites, keep the match (v_i^1, v_j^2).
  - If the second best match v_k^2 is too similar to v_i^1, ignore the match (v_i^1, v_j^2).
- Fit a model to the matches to remove mismatches (e.g. homography).
  - If enough features are matched, we have a good match between the images.
- Still a good method to confirm the identity of a detected object.

Bag of Words (BoW): extension to classes



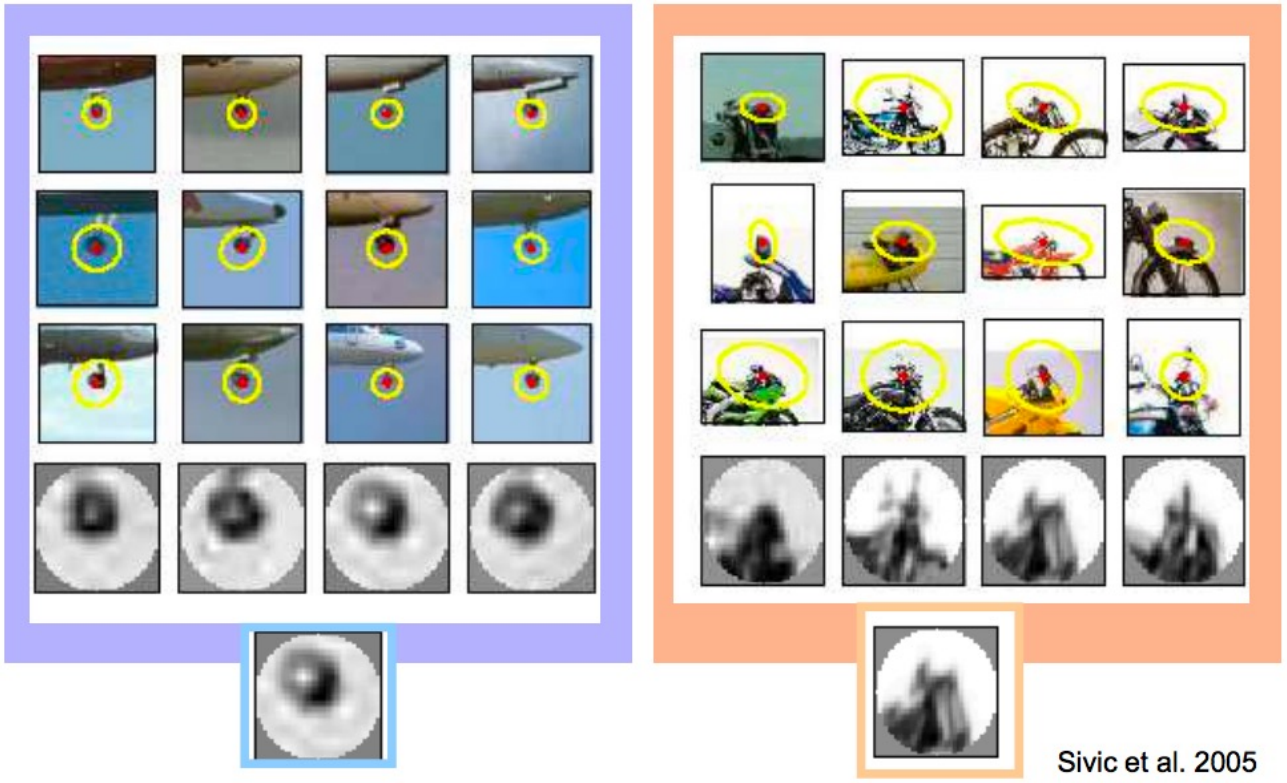
- Idea: represent each image as a histograms of features (often SIFT).
- Google: a document is represented as a histogram of words.
- Sparse histograms can be very quickly matched, even if they are large.

Bag of Words (BoW): codebook creation



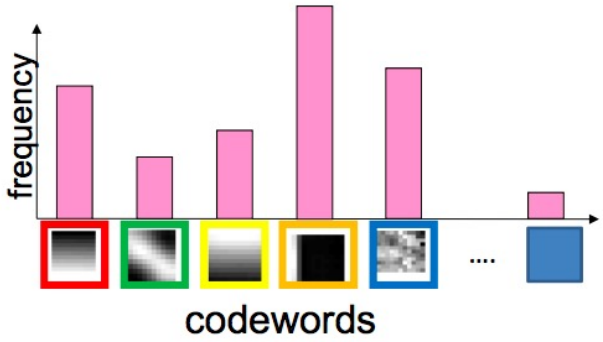
- Goal: Group (SIFT) features into clusters in feature space.
- Hierarchical K-means clustering has often been used in practice.
- Codebooks are often very large with more than 10 thousand words.

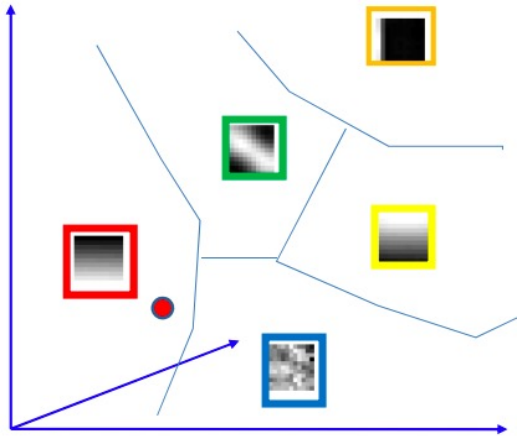
Bag of Words (BoW): codebook example



Sivic et al. 2005

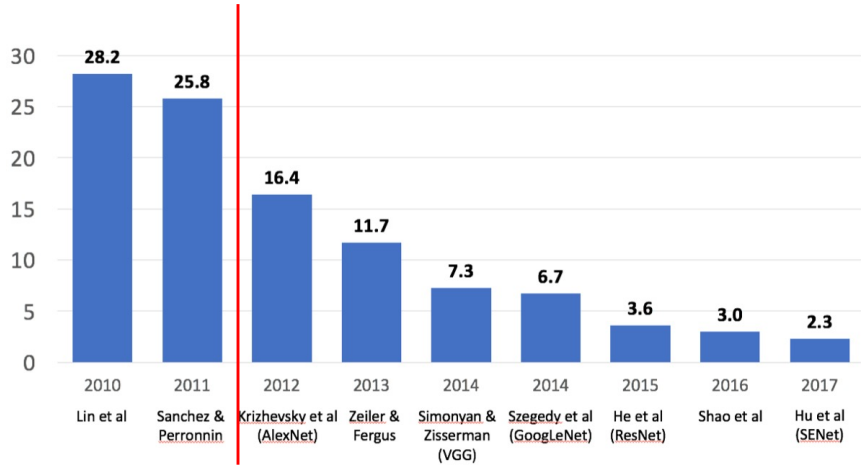
Bag of Words (BoW): Summary of steps

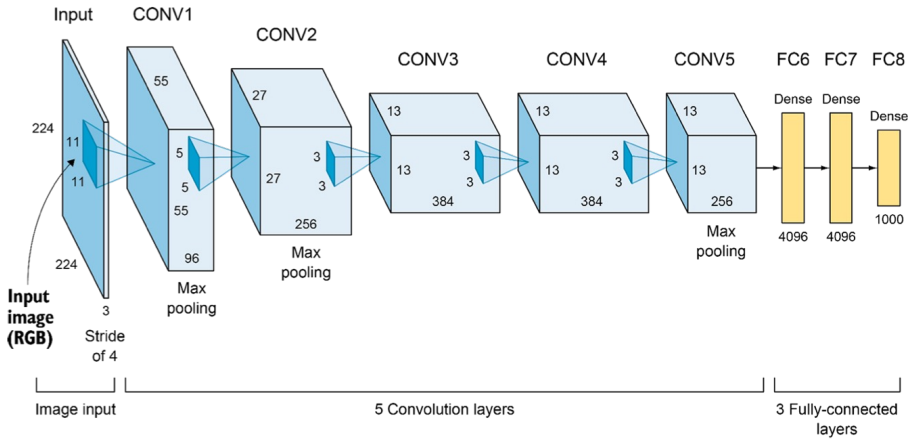




- Step 1: Extraction of image features (e.g. SIFT)
- Step 2: Clustering in feature space into codewords (e.g. K-means)
- Step 3: Collection of histograms of feature codewords
- Step 4: Classification of histograms into different object classes

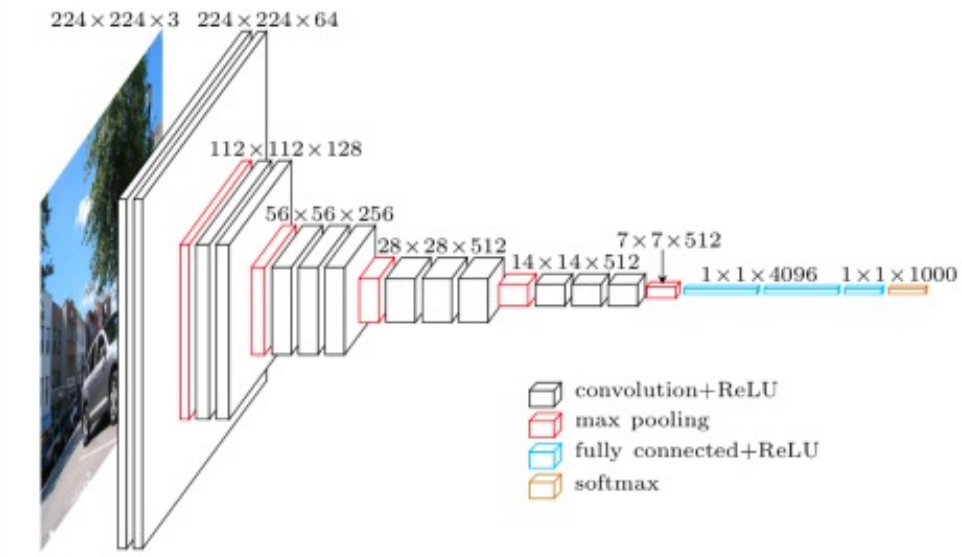
Deep learning breakthrough





- In 2012, a deep neural network (AlexNet) showed to be superior on ImageNet.
- Based on LeNet (LeCun et al, 1998), but was deeper than earlier networks.
- Breakthrough: used ReLU for activation and normalisation of activation to allow training of deeper networks.

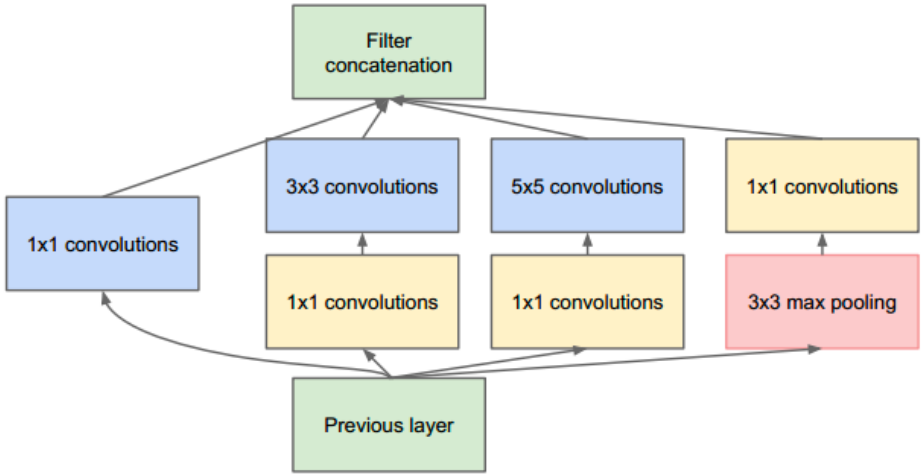
VGG (2014)



- Novelty: VGG only used small 3x3 kernels, but instead had more layers.
- Depth showed to be more important than larger kernels.
- Often used for feature learning for a new task by retraining only the last layers.

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

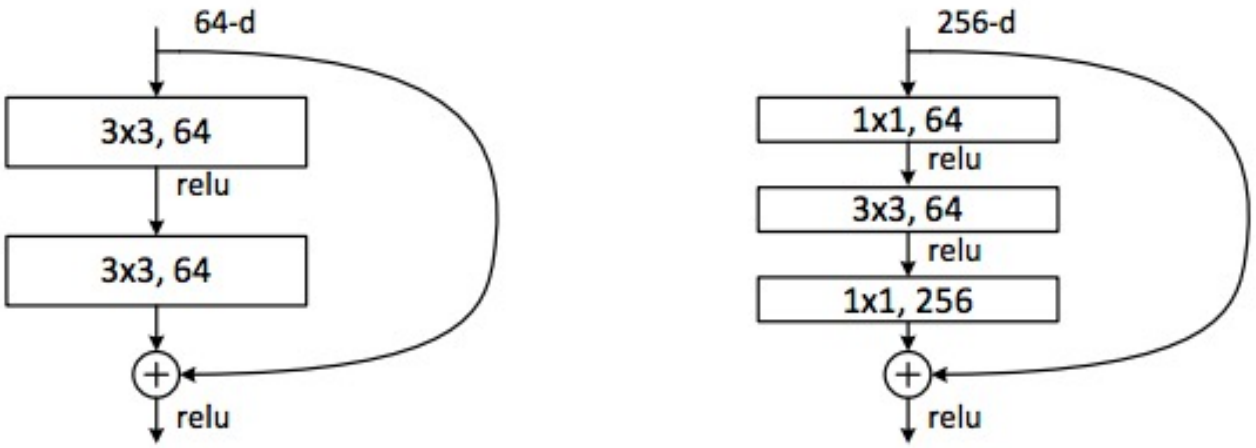
GoogLeNet (2014)



- Novelty: Used sequences of so called Inception modules.
- Instead of 3D kernels over image space and channels, 1x1 convolutions are over input channels only.
- Leads to much fewer parameters to train ⟶ faster training.

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

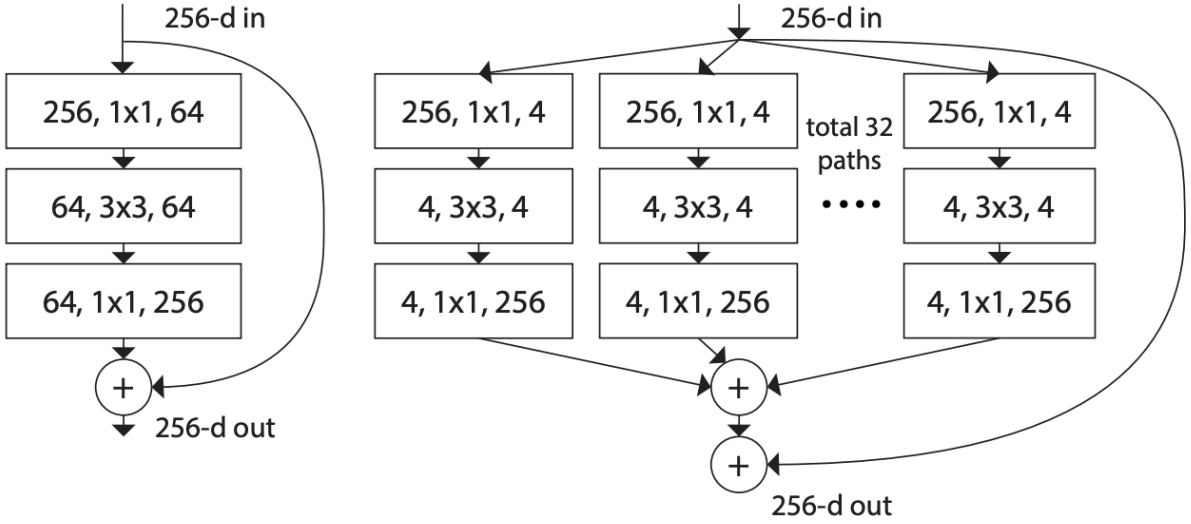
ResNet (2015)



- Novelty: Used sequences of so called Residual blocks.
- Included connections that skip layers.
- Faster backpropagation of errors during training ⟶ deeper networks.

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

ResNeXt (2017)



- Novelty: Project down to 4×32 dimensions instead of 64
- Similar number of parameters, but more ability to learn

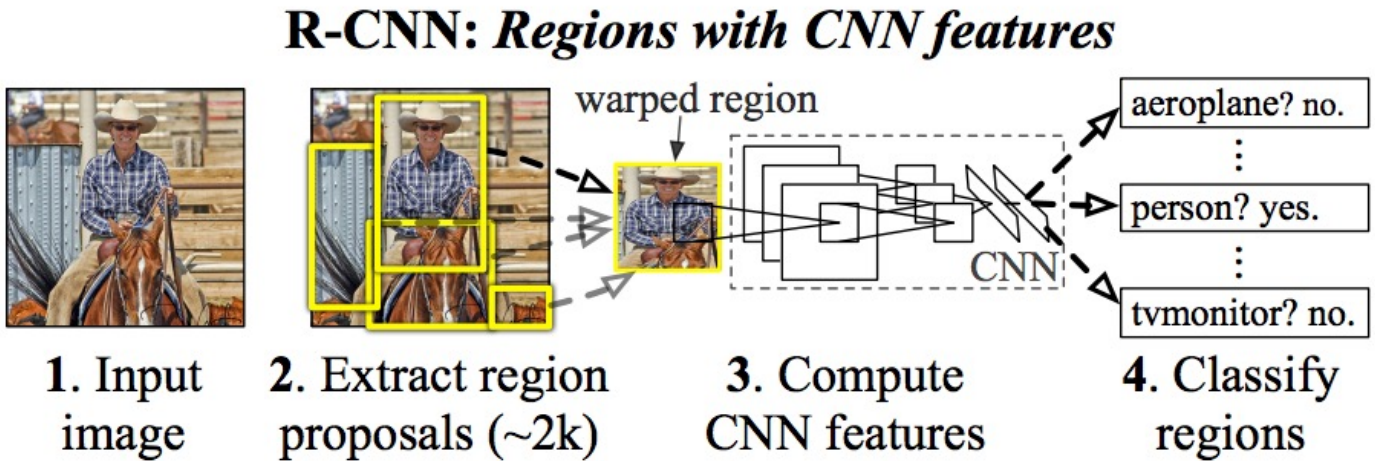
Xie et al. ”Aggregated Res. Transformations for Deep Neural Networks.”, CVPR 2017.

Results on ImageNet

Network	Layers	Top-1 error	Top-5 error	#params	Year
AlexNet	8	42.9%	16.4%	60M	2012
VGG-16	16	27.0%	8.8%	138M	2014
Inception V2	22	25.2%	7.8%	11M	2015
ResNet-152	152	21.4%	5.7%	60M	2015
ResNeXt-101 32x4	101	19.1%	4.4%	84M	2017
Meta Pseudo Labels	800+	9.8%	1.2%	480M	2021

- In recent years, most focus on refining existing models.
- The best models today require dedicated supercomputers to train.
  - Apply large scale architecture search to find the best possible hyperparameters.
  - Ensembles – train multiple classifiers and combine into a new one.
  - Semi-supervised – add billions of automatically annotated images.

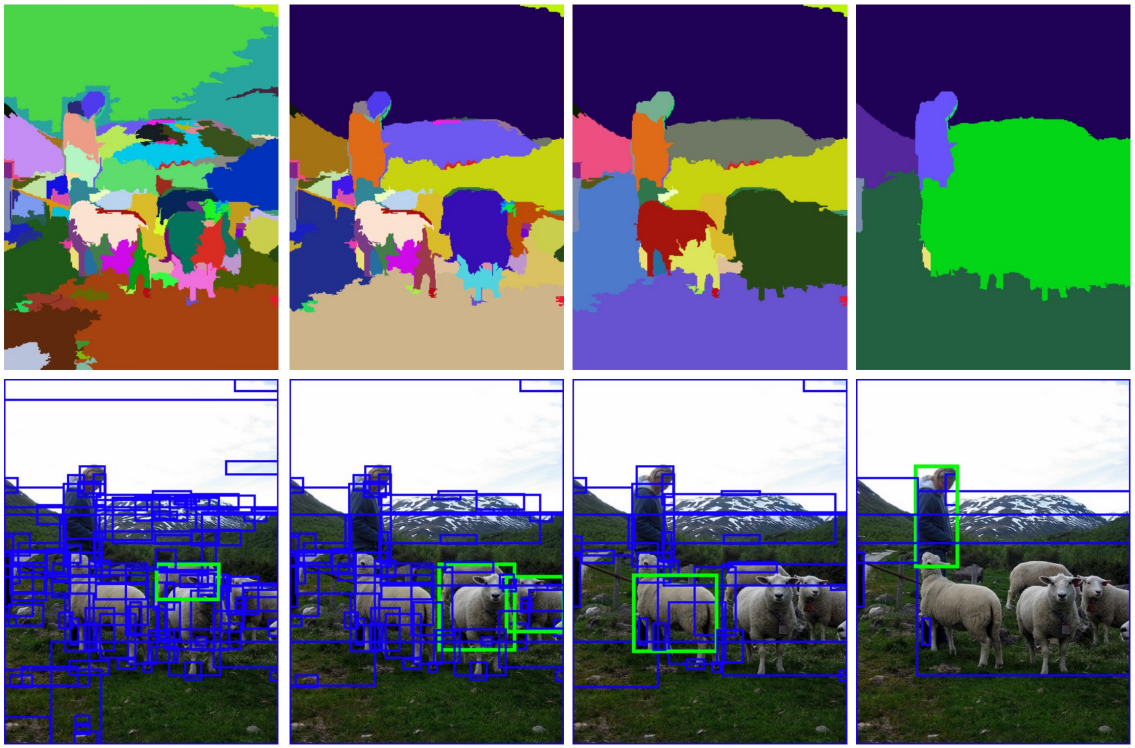
Object detection with CNNs



- Detection is a much harder problem – first you need to find object candidates.
- R-CNN is the first real deep learning-based attempt, but it is slow, since each candidate region needs to be warped and processed by a neural network.

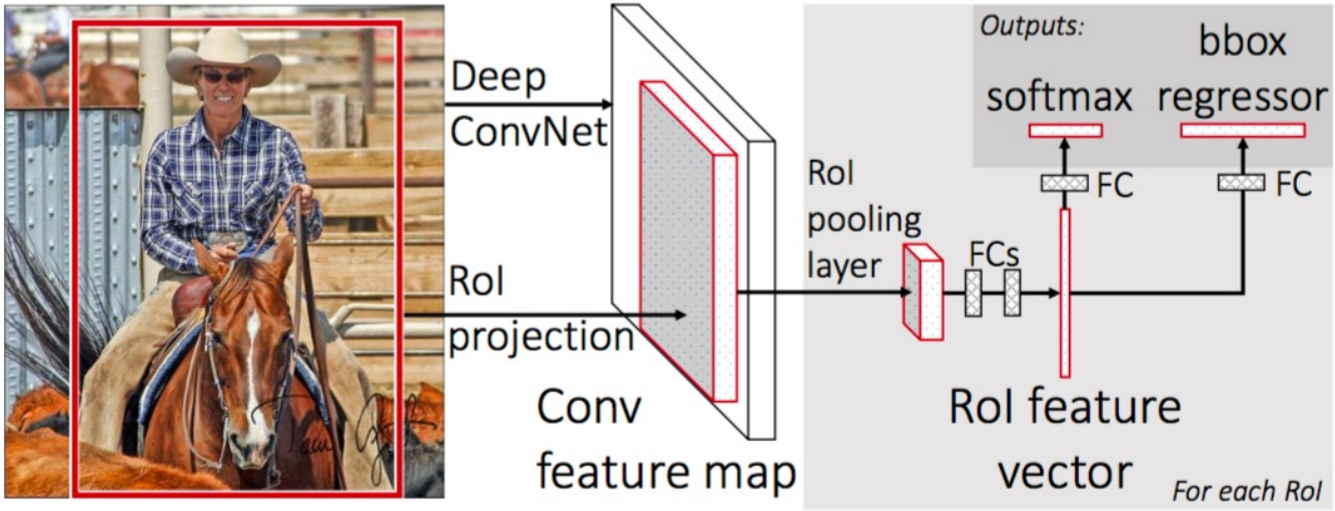
Girshick et al. ”Rich feature hierarchies for accurate object detection and semantic segmentation.”, CVPR 2014.

Selective search for R-CNN (2014)



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

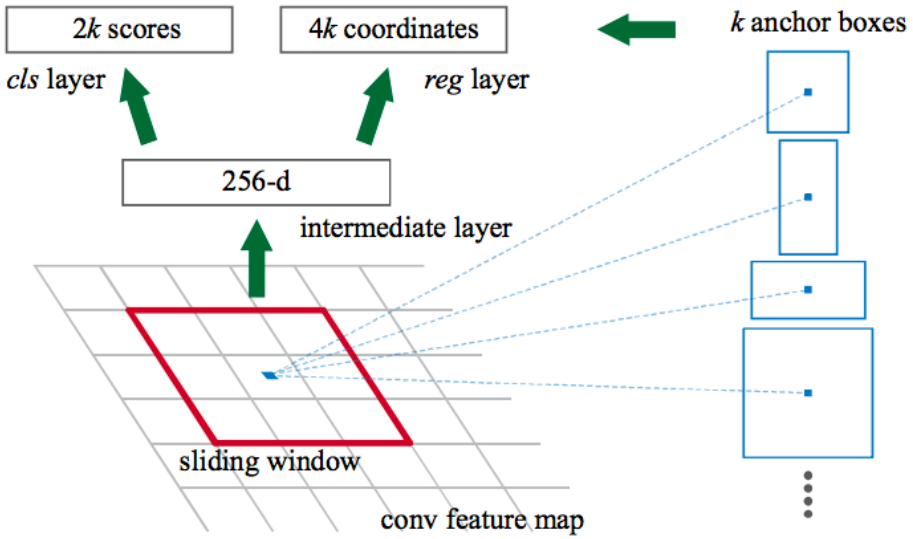
Fast R-CNN (2015)



- Novelty: apply convolutions only on the image to create feature maps.
- With pooling let region proposals read from feature maps, not from the image.
- Refine object by a bounding box regressor to get a better fit.

Girshick, ”Fast R-CNN”, ICCV 2015.

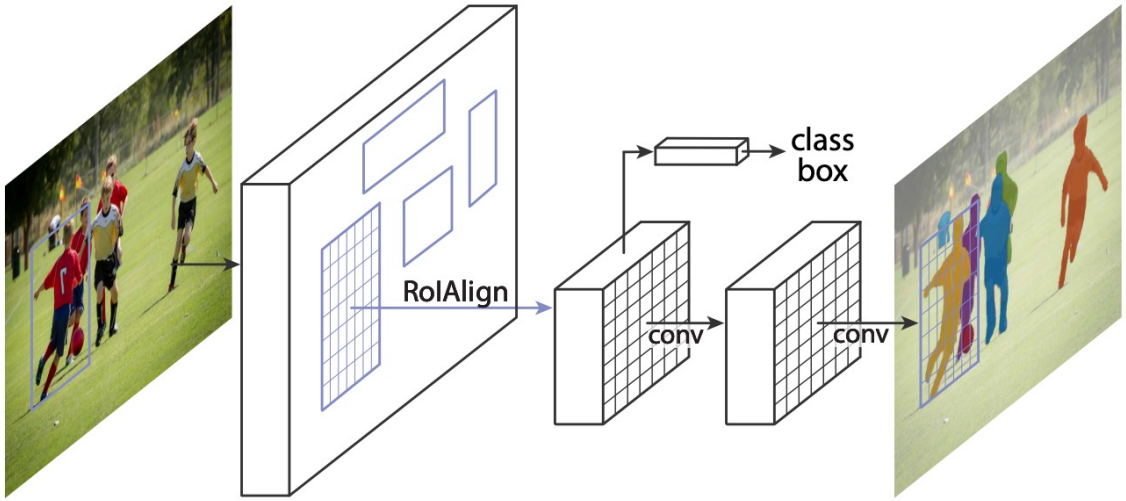
Faster R-CNN (2015)



- Novelty: replace selective search (mean-shift) with a CNN.
- First compute feature map, then test different positions and sizes.
- Each anchor outputs class scores and refined bounding boxes.

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

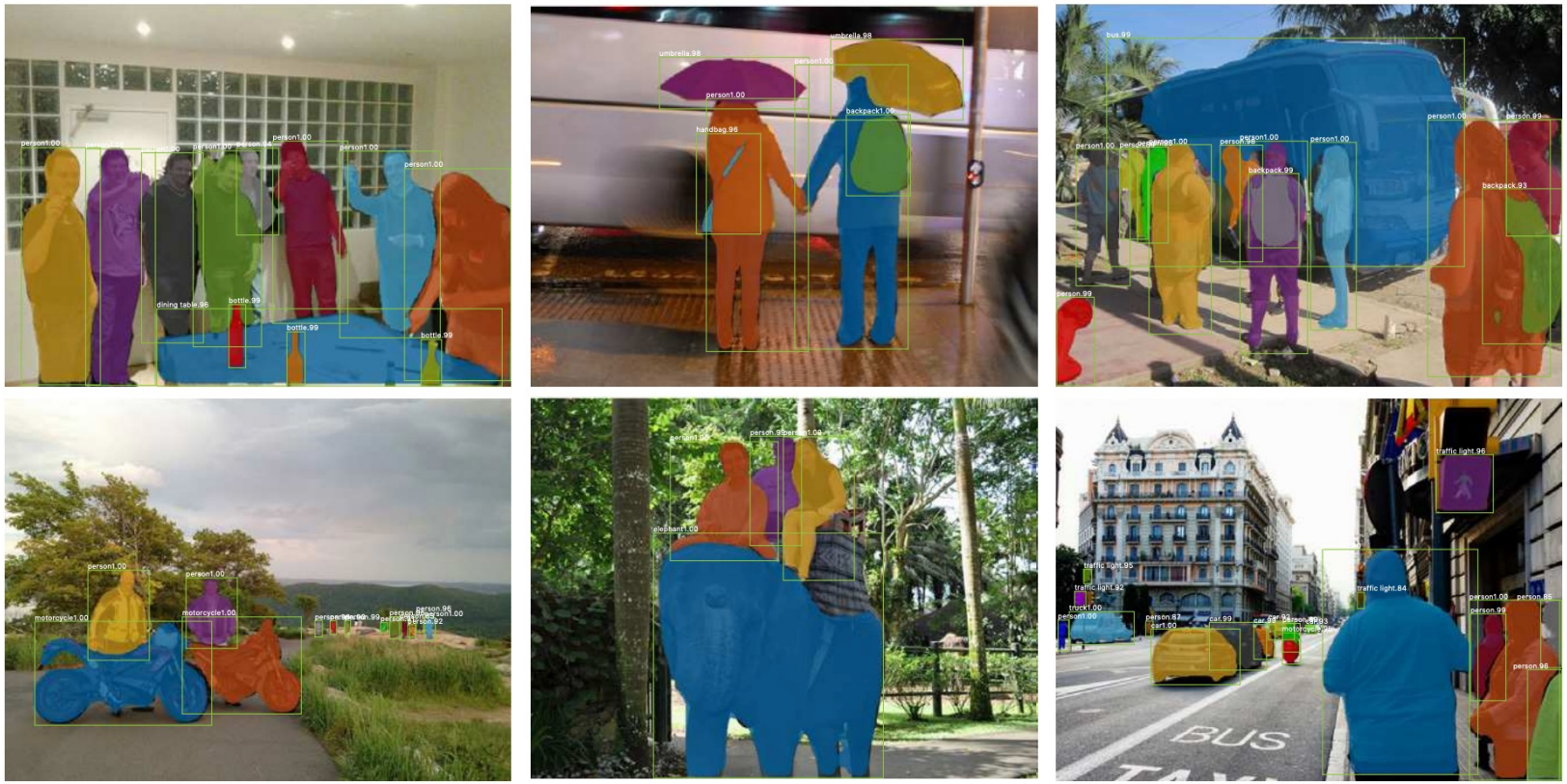
Mask R-CNN: Combined detection & segmentation



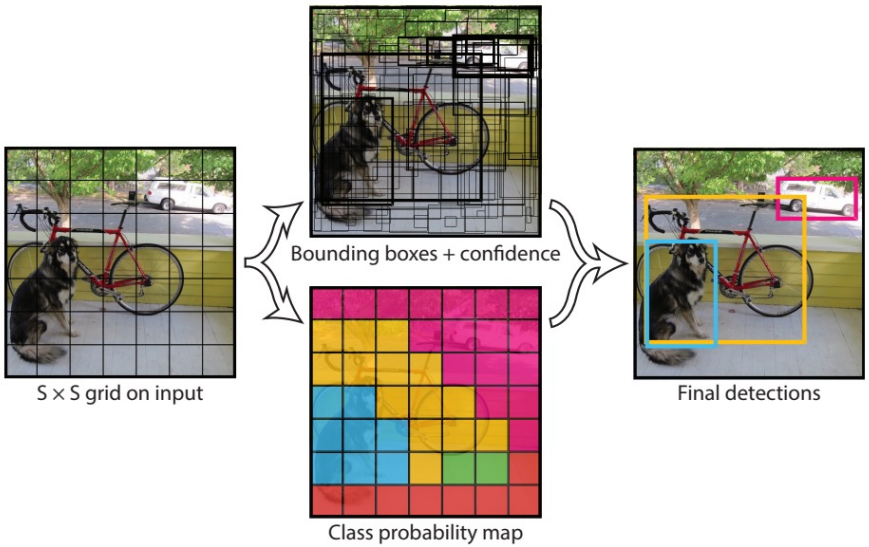
- Idea: If you know the object class, segmentation should become easier.
- Mask R-CNN uses Faster R-CNN to detect objects and find bounding boxes.
- Then attaches a fully convolutional network (FCN) for object segmentation.

He, K., et al. ”Mask R-CNN”, ICCV 2017.

Mask R-CNN examples



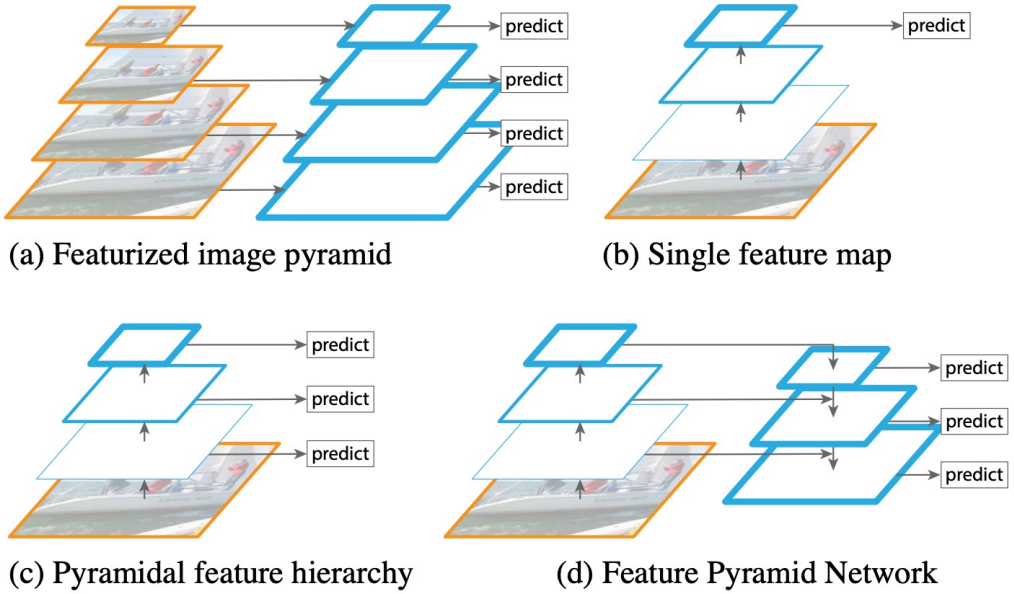
YOLO – You only look once (2016)



- Novelty: combine everything into a single pass. Much faster!
- YOLO decouples class score from bounding box prediction.
- Uses fewer anchor points, leads to problems with small objects.

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

FPN – Feature pyramid networks (2016)



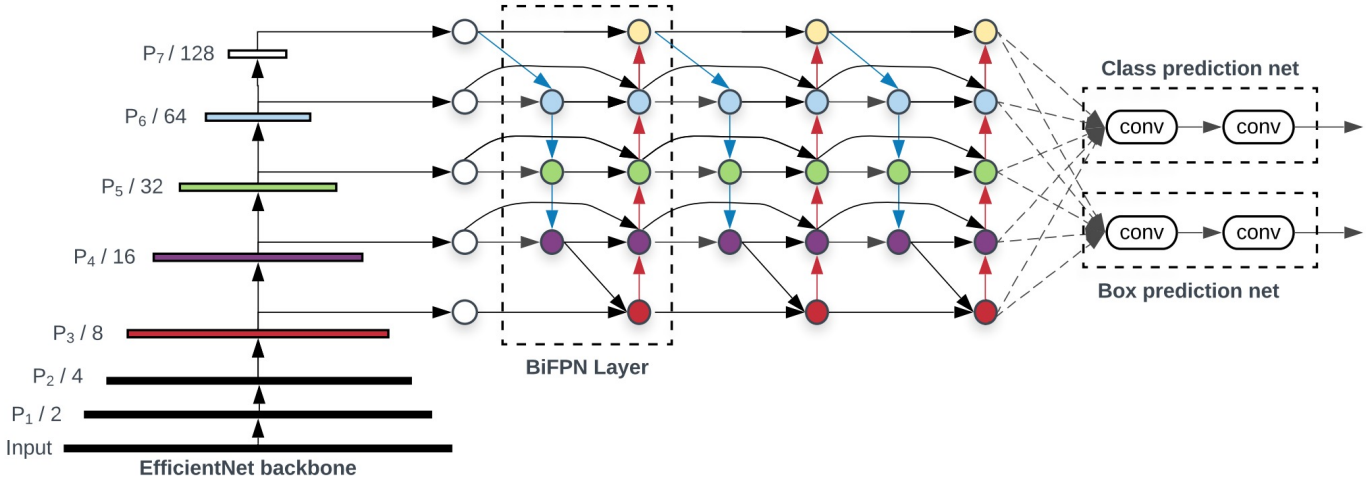
(a) Featurized image pyramid (b) Single feature map

(c) Pyramidal feature hierarchy (d) Feature Pyramid Network

- Novelty: Run Faster R-CNN at different scales in a pyramid.
- Instead of a Gaussian pyramid (top-left), use a pyramid of features (bottom-right) and combine features of different scales.

Lin et al., ”Feature pyramid networks for object detection”, CVPR 2017.

EfficientDet (2020)



- Novelty: Uses 1) EfficientNet as a base network and 2) a more complex method of combining features at different scales.
- EfficientNet: for a given time budget, the image size, number of channels and layers are optimized using large-scale architecture search.

Tan at al, ”EfficientDet: Scalable and efficient object detection”, CVPR 2020.

Object detection precision vs speed

Results on PASCAL VOC07 benchmark

Method	Accuracy (mAP)	FPS	Year
R-CNN	68.4	0.02	2014
Fast R-CNN	70.0	0.44	2015
Faster R-CNN	76.4	5	2015
YOLOv1	63.4	45	2016
YOLOv2	78.6	34	2017

Results on COCO benchmark

Method	Accuracy (box AP)	FPS	Year
Faster R-CNN	34.7	5	2015
Faster R-CNN+FPN	36.2	5	2017
EfficientDet-D7	53.7	6	2020

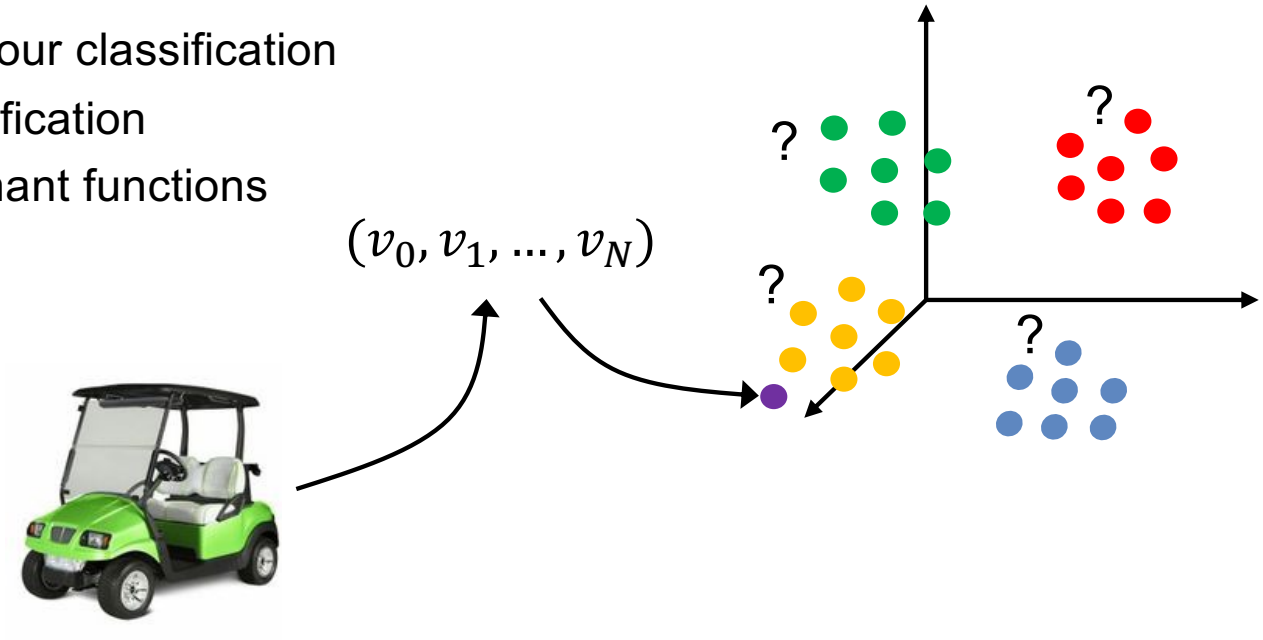
- Since the original papers, significant improvements have been made to the methods above. Optimization done with respect to either speed or accuracy.

Basic methods for classification

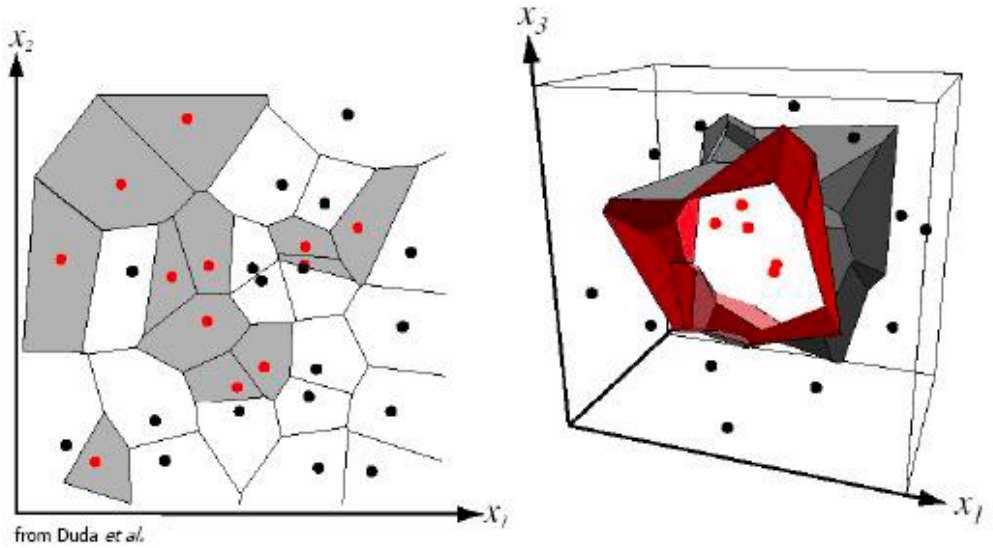
How to determine which class a feature vector belongs to?

Three basic methods:

- 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 the distances to feature vectors for a representative selection of images from known classes.
- Let the final classification be given by the class of the nearest neighbour.

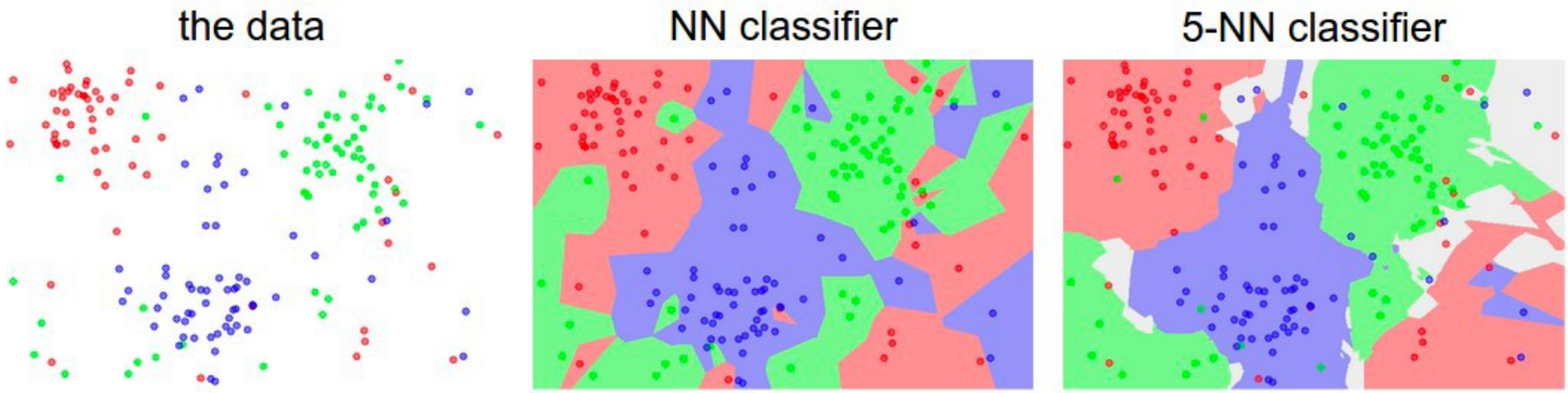
Pros:

- Very easy to implement and get some baseline results.
- Works for well separated classes.
- Can represent clusters with very complex shapes.

Cons:

- May require complex computations in higher dimensions.
- Results may depend on the choice of metric.
- Highly sensitive to outliers (no suppression).

K-Nearest neighbour classification



- Pick the class with most votes among K-nearest neighbours.
- Much more robust to individual weird samples (outliers).

Bayesian classification

- Idea: consider 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 the distribution of feature vector for class k, $p(v|k)$,
  we can compute the class probability given the feature vector, $p(k|v)$.
- Bayes' formula:

$$p(v,k) = p(v|k)p(k) = p(k|v)p(v) \Longrightarrow$$

$$p(k|v) = \frac{p(v|k)p(k)}{p(v)} = \frac{p(v|k)p(k)}{\sum_i p(v|k_i)p(k_i)}$$

Bayesian classification

Choose the class k that maximizes $p(k|v)$.

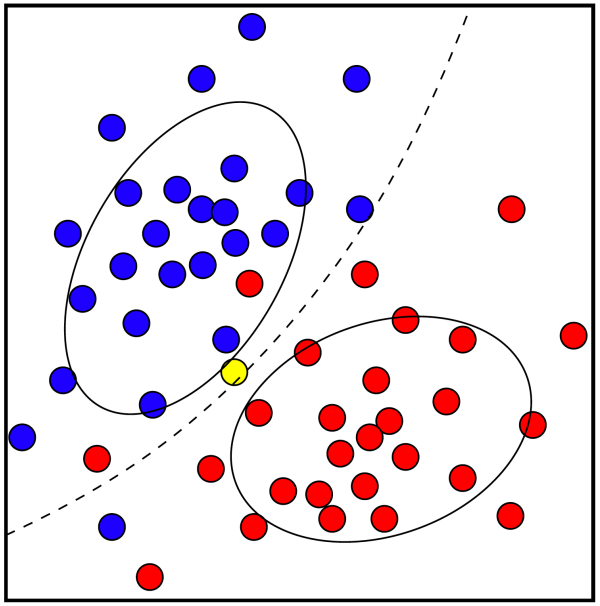
- Minimizes the probability of wrong classification, IF statistical model is correct.
- Gives most probable class, if prior probabilities $p(k)$ are known.
- Common assumption: All $p(k)$ equal $\Longrightarrow$ Maximum likelihood

Note: it is enough to compute $p(v|k)p(k)$ for each class, since $p(v)$ does not depend on the class.

$$p(k|v) = \frac{p(v|k)p(k)}{p(v)} \backsimeq p(v|k)p(k)$$

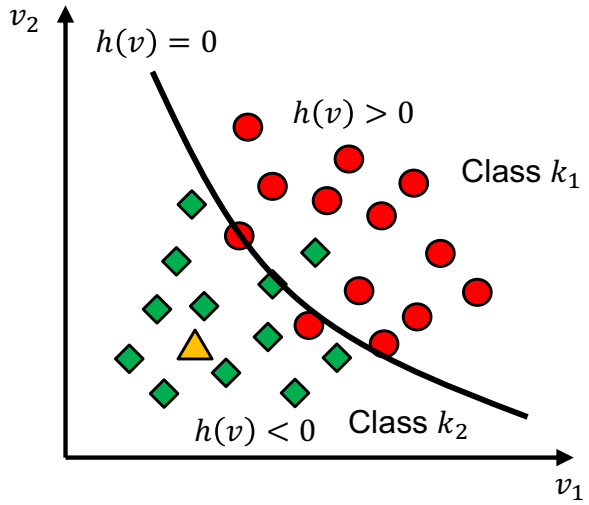
Assuming 2D Gaussian distributions

$$p(v|k_i) = \frac{1}{2\pi\sqrt{|\Sigma_i|}} \exp\left\{-\frac{1}{2}(v-\mu_i)^T \Sigma_i^{-1}(v-\mu_i)\right\}, \qquad p(k_1|v) = \frac{p(v|k_1)p(k_1)}{p(v|k_1)p(k_1) + p(v|k_2)p(k_2)}$$



Decision boundary given by $p(v|k_1)p(k_1) = p(v|k_2)p(k_2)$.

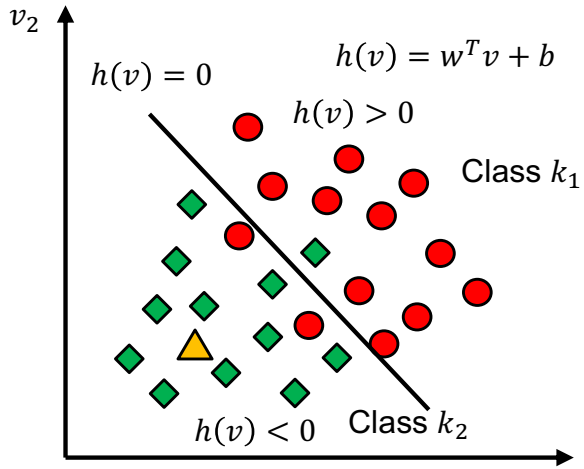
Discriminant functions



- Instead of modelling the distributions and then find a decision function, find a decision function $\mathrm{h}(v)$ 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 parameters (w, b) given training examples.
- Most modern machine learning methods (Boosting, SVM, FCN) use combinations of many linear discriminant functions.
- Note that the last fully-connected layers of a CNN can be seen 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 are the dominating methods based on deep learning?
- Why is object detection harder than object classification?
- What characteristics does a nearest neighbour classifier have?
- How do you find a decision boundary for Bayesian classification?

Recommended reading

- Gonzalez & Woods: Chapter 12.4
- Szeliski: Chapter 5.1 and 6.2-6.3