NYU TANDON SCHOOL OF ENGINEERING

AI4CE

Robot Vision

Object Detection

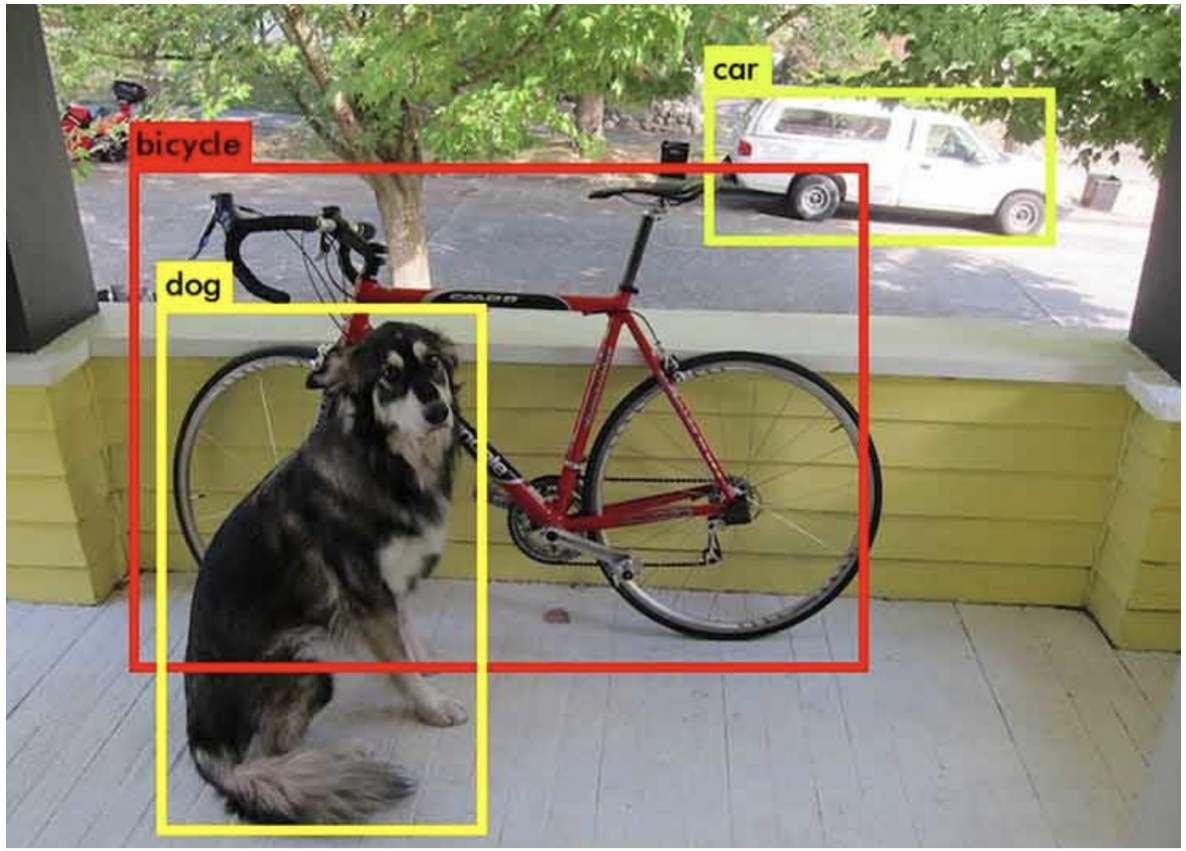
Dr. Chen Feng

cfeng@nyu.edu

ROB-UY 3203, Spring 2024

Overview

- Before Deep Learning
  - AdaBoost
  - HOG
- Deep Learning for Object Detection
  - Faster R-CNN
  - Single-stage Detector: YOLO
- Evaluation for Object Detection



Source: Blog by Matthijs Hollemans

References

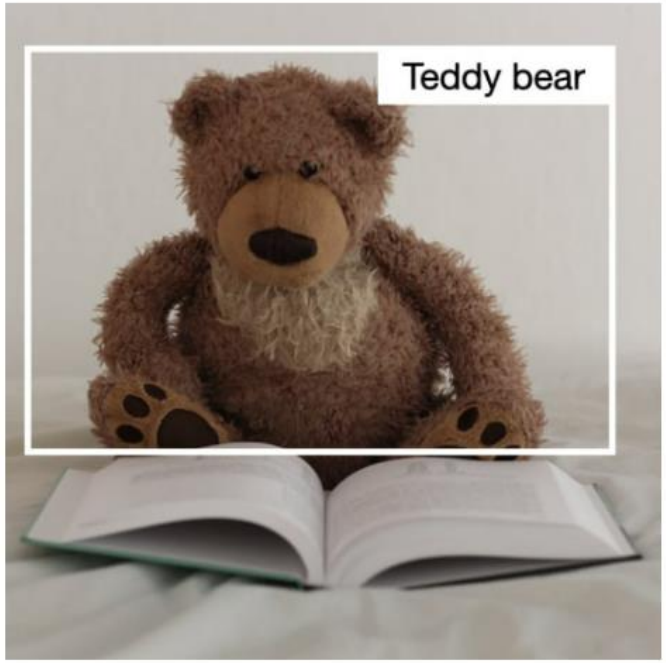
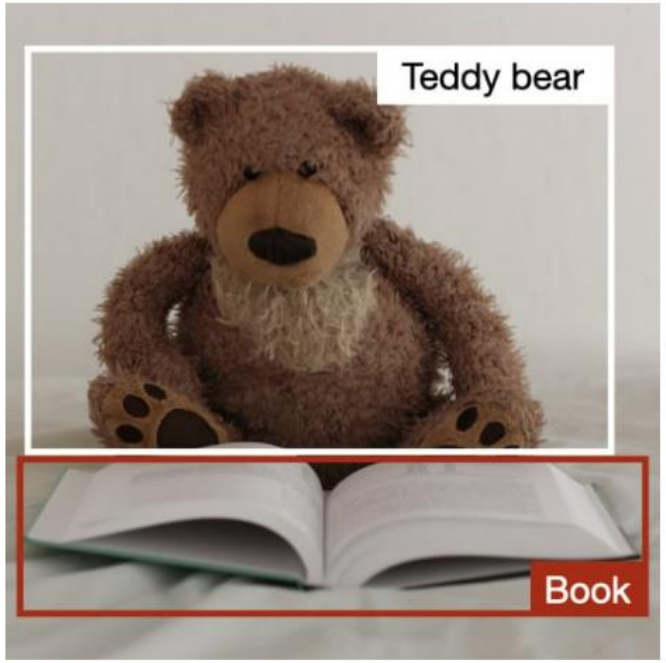
- Szeliski 2022
  - Section 6.3

- Forsyth & Ponce 2011
  - Section 15.1, Chapter 17

- https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.AdaBoostClassifier.html

- Ren, Shaoqing, Kaiming He, Ross Girshick, and Jian Sun. "Faster r-cnn: Towards real-time object detection with region proposal networks." arXiv preprint arXiv:1506.01497 (2015).
- Redmon, Joseph, Santosh Divvala, Ross Girshick, and Ali Farhadi. "You only look once: Unified, real-time object detection." In Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 779-788. 2016.

Different Types of Object Recognition

Image classification	Classification w. localization	Detection
		
• Classifies a picture • Predicts probability of object	• Detects an object in a picture • Predicts probability of object and where it is located	• Detects up to several objects in a picture • Predicts probabilities of objects and where they are located

Classification

- Classification algorithms are supervised algorithms to predict categorical labels

- Differs from regression which is a supervised technique to predict real-valued labels

Formal problem statement:

- **Produce a function that maps**

  $C : \mathcal{X} \rightarrow \mathcal{Y}$

- **Given a training set**

  $\{(\mathbf{x_1}, y_1), \ldots, (\mathbf{x_n}, y_n)\}$

  $y \in \mathcal{Y}$ label

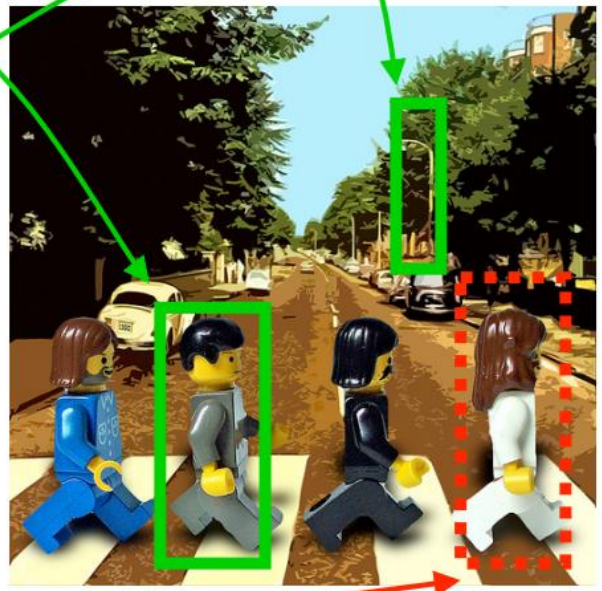
  $\mathbf{x} \in \mathcal{X}$ training sample

Slides from Arras et. al.

Classification

Error types

Predicted value of the classifier	True value: T	True value: N
T'	True Positive	False Positive
N'	False Negative	True Negative



- **Precision** = TP / (TP + FP)
- **Recall** = TP / (TP + FN)

Many more measures...

Slides from Arras et. al.

Boosting

- An **ensemble** technique (a.k.a. committee method)
- Supervised learning: given <samples x, labels y>

- Learns an accurate **strong classifier** by combining an ensemble of inaccurate "rules of thumb"
- **Inaccurate** rule $h_i(x)$: "weak" classifier, weak learner, basis classifier, feature
- **Accurate** rule H(x): "strong" classifier, final classifier

- Other ensemble techniques exist: Bootstrap Aggregation (Bagging), Voting, Mixture of Experts, etc.

Slides from Arras et. al.

AdaBoost (Adaptive Boosting)

- Most popular algorithm: **AdaBoost**

  [Freund et al. 95], [Schapire et al. 99]

- Given an ensemble of weak classifiers $h(x_i)$, the combined strong classifier $H(x_i)$ is obtained by a **weighted majority voting scheme**

$$f(x_i) = \sum_{t=1}^{T} \alpha_t h_t(x_i) \qquad H(x_i) = \text{sgn}\big(f(x_i) \big)$$

- AdaBoost in Robotics:

  [Viola et al. 01], [Treptow et al. 04], [Martínez-Mozos et al. 05], [Rottmann et al. 05] , [Monteiro et al. 06] , [Arras et al. 07]

Slides from Arras et. al.

Why is AdaBoost interesting?

- 1. It tells you what the **best "features"** are
- 2. What the **best thresholds** are, and
- 3. How to **combine them to a classifier**

- AdaBoost can be seen as a **principled feature selection strategy**
- Classifier design becomes **science**, not art

- AdaBoost is a **non-linear** classifier
- Has good generalization properties: can be proven to maximize the margin
- Quite **robust** to overfitting
- Very **simple** to implement

Slides from Arras et. al.

Weak Classifiers

- Prerequisite: weak classifier must be better than chance
  - error < 0.5 in a binary classification problem

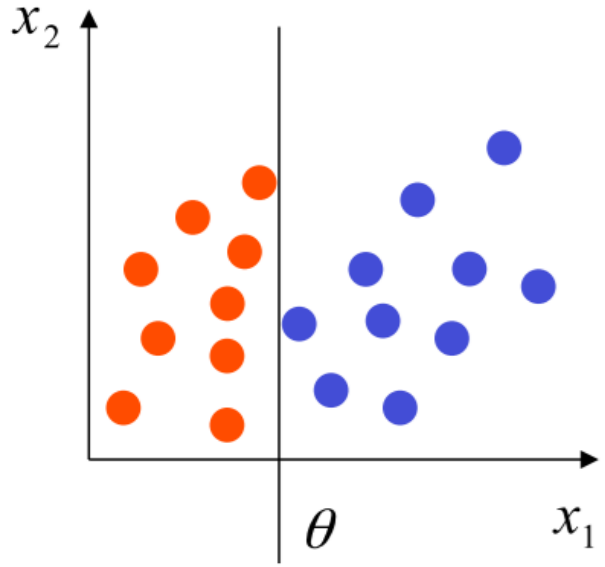
- Possible Weak Classifiers:
  - Decision stump: Single axis-parallel partition of space
  - Decision tree: Hierarchical partition of space
  - Multi-layer perceptron: General non-linear function approximators
  - Support Vector Machines (SVM): Linear classifier with RBF Kernel

- Trade-off between diversity among weak learners versus their accuracy.
- **Decision stumps are a popular choice**

Decision stump

- Simple-most type of **decision tree**
- Equivalent to linear classifier defined by affine hyperplane
- Hyperplane is orthogonal to axis with which it intersects in threshold θ
- Commonly not used on its own
- Formally,

$$h(x; j, \theta) = \begin{cases} +1 & x_j > \theta \\ -1 & \text{else} \end{cases}$$



where x is (d-dim.) training sample, j is dimension

Slides from Arras et. al.

AdaBoost

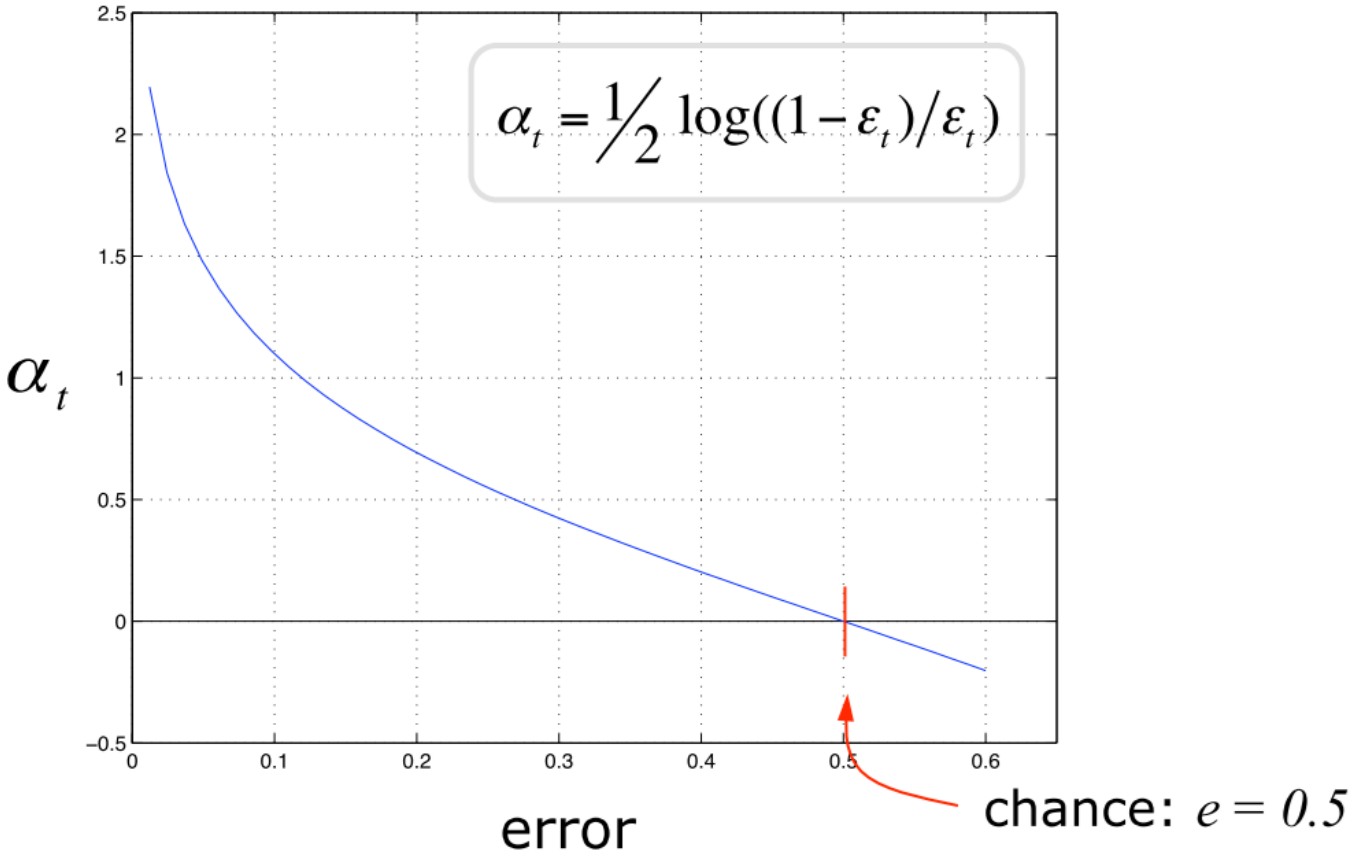
Given the **training data** $\{(\mathbf{x_1}, y_1), \ldots, (\mathbf{x_n}, y_n)\}$ $\mathbf{x} \in \mathcal{X}$ $y \in \mathcal{Y}$

1. Initialize weights $w_t(i) = 1/n$
2. For $t = 1,\ldots,T$
   - Train a **weak classifier** $h_t(x)$ on weighted training data minimizing the error
     $$\varepsilon_t = \sum_{i=1}^{n} w_t(i)\, \mathrm{I}(y_i \neq h_t(x_i))$$
   - Compute voting weight of $h_t(x)$:
     $$\alpha_t = \tfrac{1}{2} \log((1-\varepsilon_t)/\varepsilon_t)$$
   - Recompute weights:
     $$w_{t+1}(i) = w_t(i) \exp\{-\alpha_t\, y_i\, h_t(x_i)\} / Z_t$$
3. Make predictions using the final **strong classifier**

Slides from Arras et. al.

Voting Weight

- Computing the **voting weight** α_t of a weak classifier
- α_t measures the **importance** assigned to $h_t(x_i)$



Slides from Arras et. al.

Weight Update

- Looking at the weight update step:

$$w_{t+1}(i) = w_t(i)\exp\{-\alpha_t y_i h_t(x_i)\} / Z_t$$

Z_t : **Normalizer** such that w_{t+1} is a prob. distribution

$$\exp\{-\alpha_t y_i h_t(x_i)\} = \begin{cases} <1, & y_i = h_t(x_i) \\ >1, & y_i \neq h_t(x_i) \end{cases}$$

➔ Weights of misclassified training samples are **increased**

➔ Weights of correctly classified samples are **decreased**

- Algorithm generates weak classifier by training the next learner on the **mistakes of the previous one**
- Now we understand the name: **AdaBoost** comes from **adaptive Boosting**

Slides from Arras et. al.

Strong Classifier

- **Training is completed…**

  The weak classifiers $h_{1\ldots T}(x)$ and their voting weight $\alpha_{1\ldots T}$ are now fix

- **The resulting strong classifier is**

$$H(x_i) = \operatorname{sgn}\left(\sum_{t=1}^{T} \alpha_t h_t(x_i)\right) \longrightarrow \text{Class Result } \{+1, -1\}$$

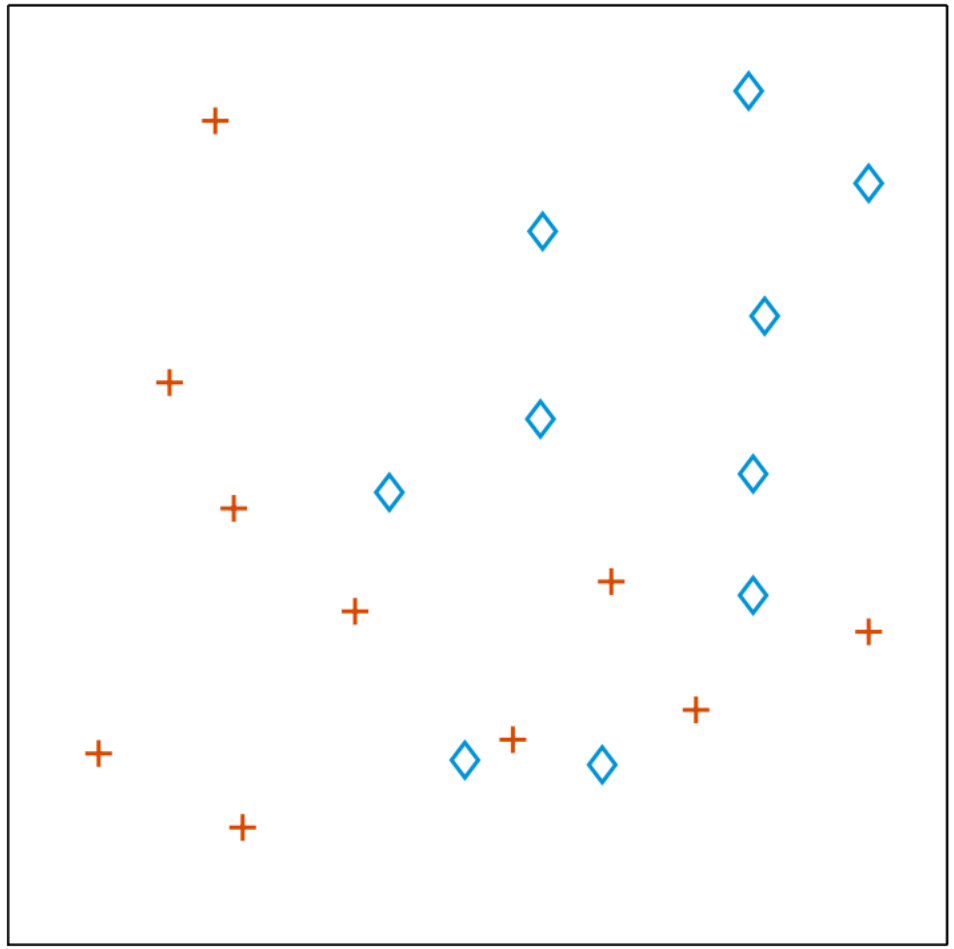
Put your data here

Weighted majority voting scheme

Slides from Arras et. al.

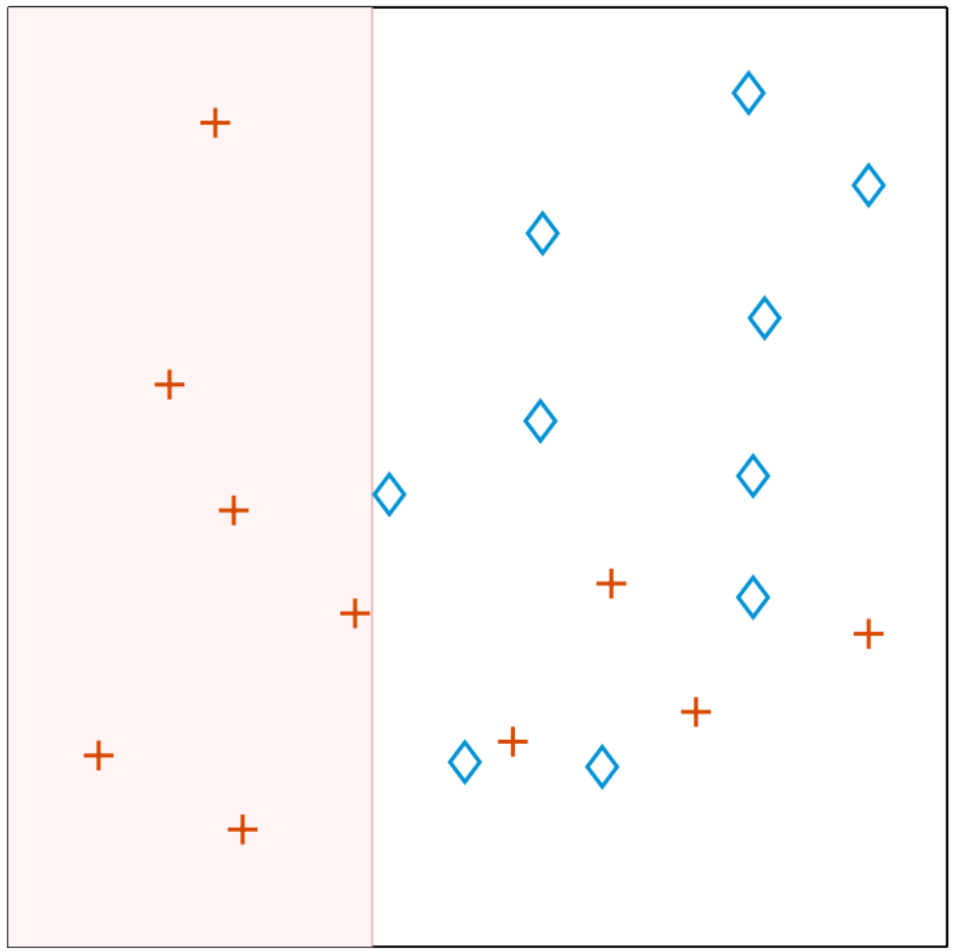
AdaBoost: Step-By-Step

- **Training data**



AdaBoost: Step-By-Step

- **Iteration 1, train weak classifier 1**



Threshold
$\theta^* = 0.37$

Dimension
$j^* = 1$

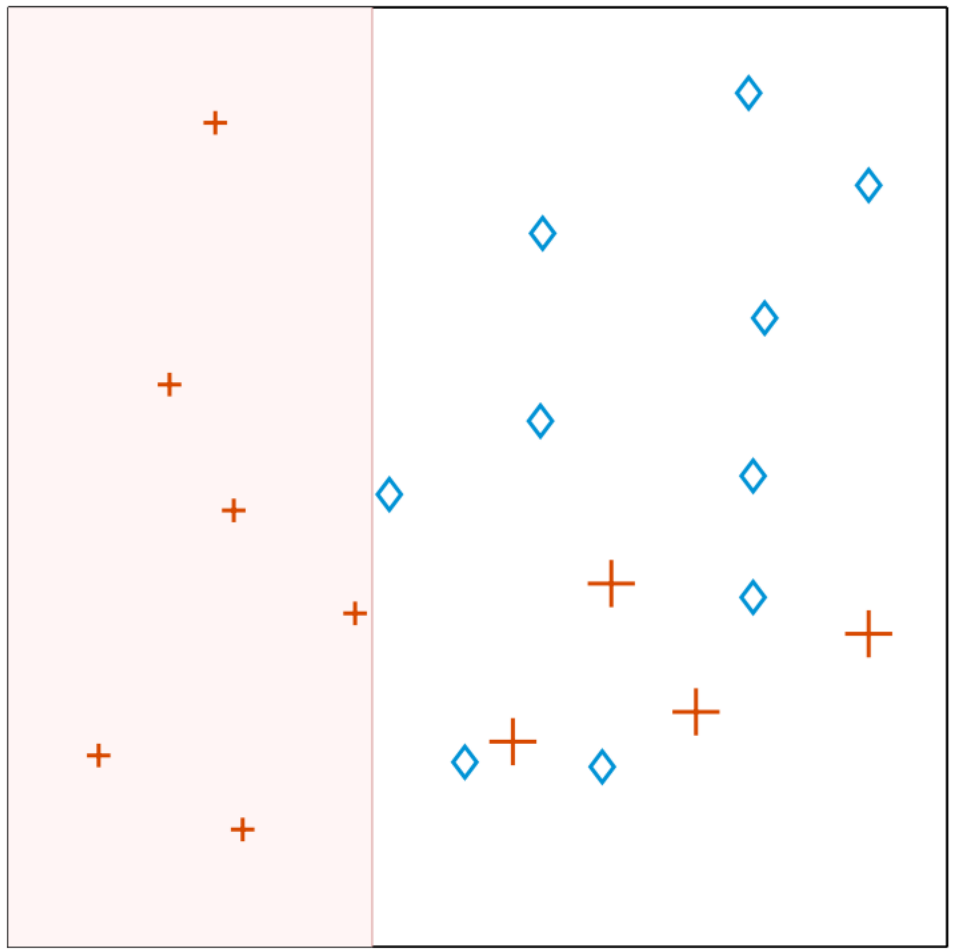
Weighted error
$e_t = 0.2$

Voting weight
$\alpha_t = 1.39$

Total error = 4

AdaBoost: Step-By-Step

- **Iteration 1, recompute weights**



Threshold
$\theta^* = 0.37$

Dimension
$j^* = 1$

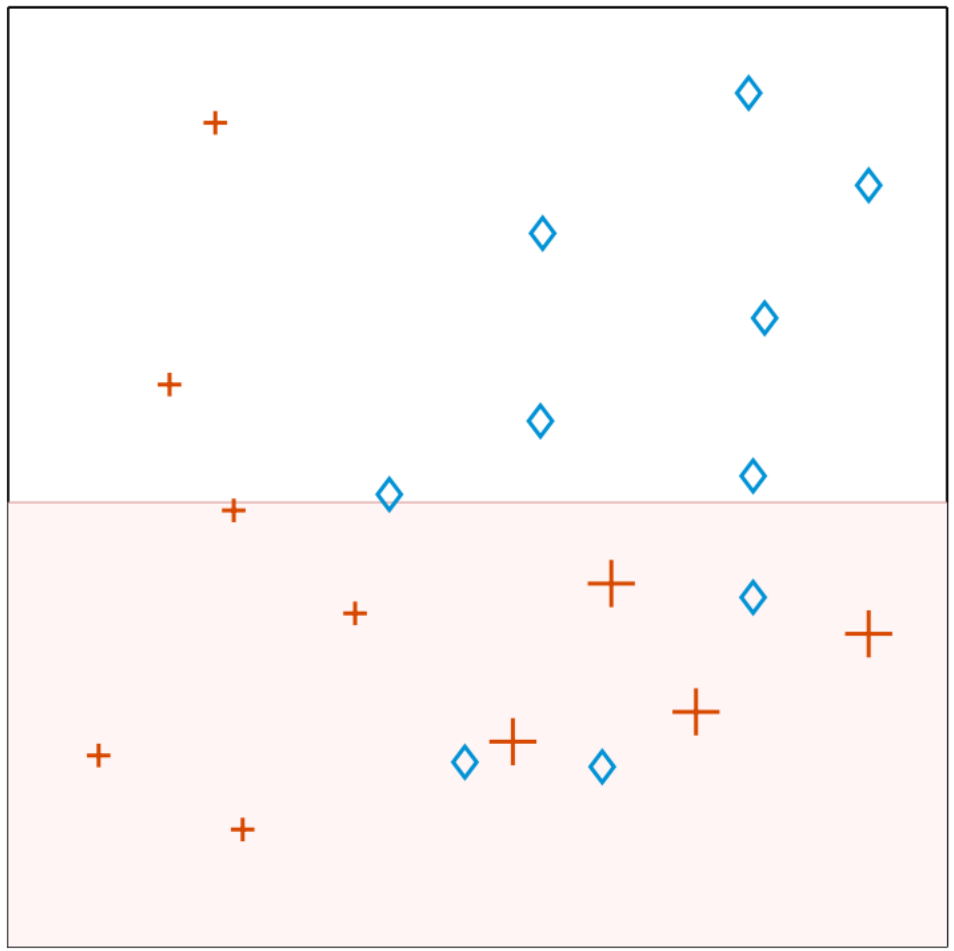
Weighted error
$e_t = 0.2$

Voting weight
$\alpha_t = 1.39$

Total error = 4

AdaBoost: Step-By-Step

- **Iteration 2, train weak classifier 2**



Threshold
$\theta^* = 0.47$

Dimension
$j^* = 2$

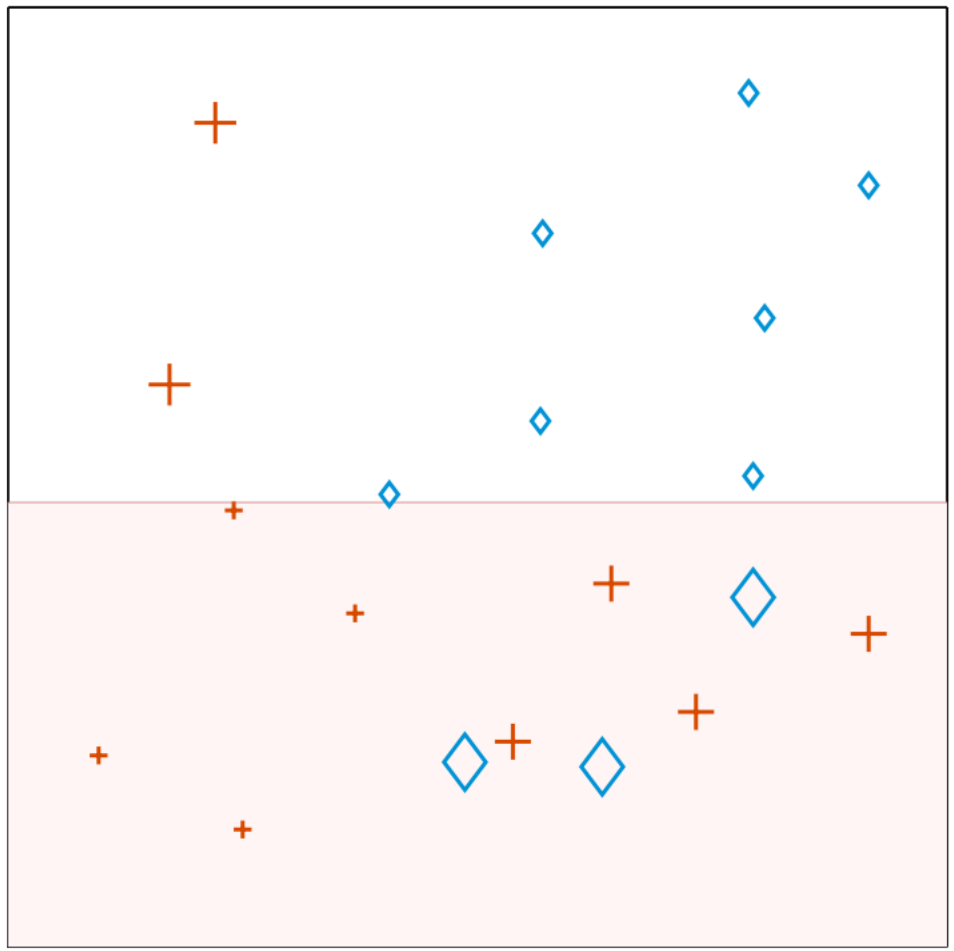
Weighted error
$e_t = 0.16$

Voting weight
$\alpha_t = 1.69$

Total error = 5

AdaBoost: Step-By-Step

- **Iteration 2, recompute weights**



Threshold
$\theta^* = 0.47$

Dimension
$j^* = 2$

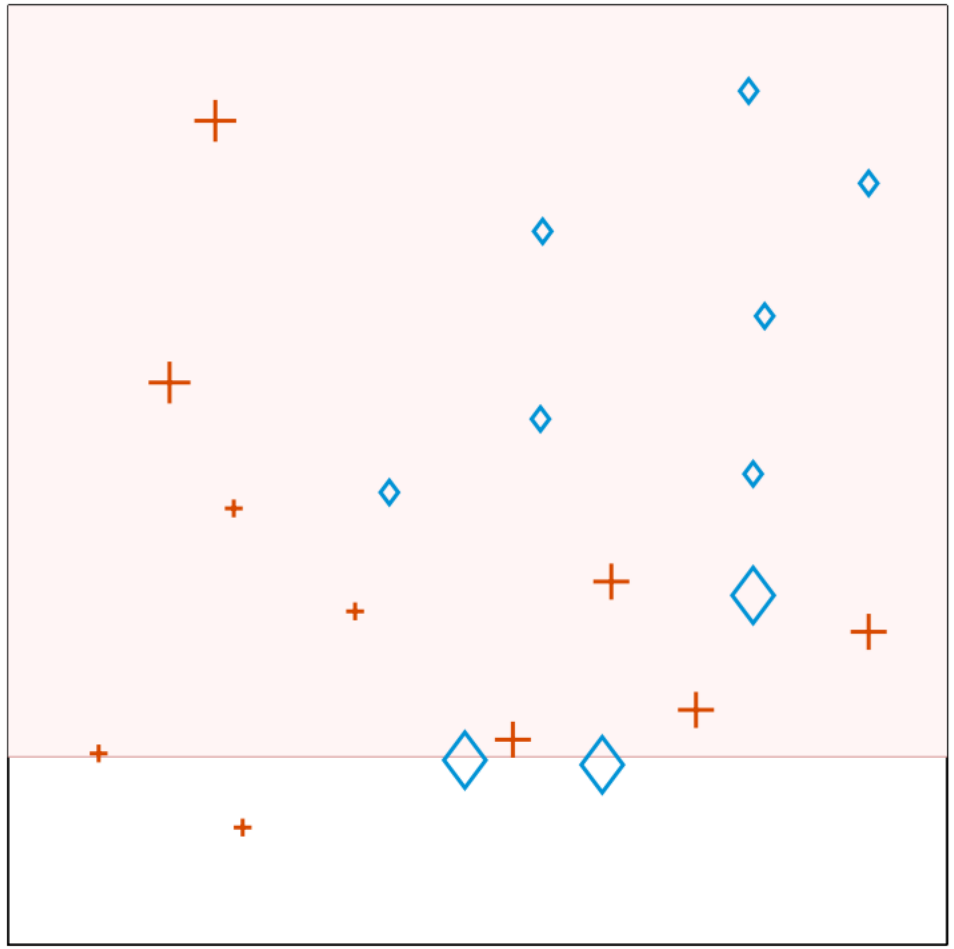
Weighted error
$e_t = 0.16$

Voting weight
$\alpha_t = 1.69$

Total error = 5

AdaBoost: Step-By-Step

- **Iteration 3, train weak classifier 3**



Threshold
$\theta^* = 0.14$

Dimension, sign
$j^* = 2$, neg

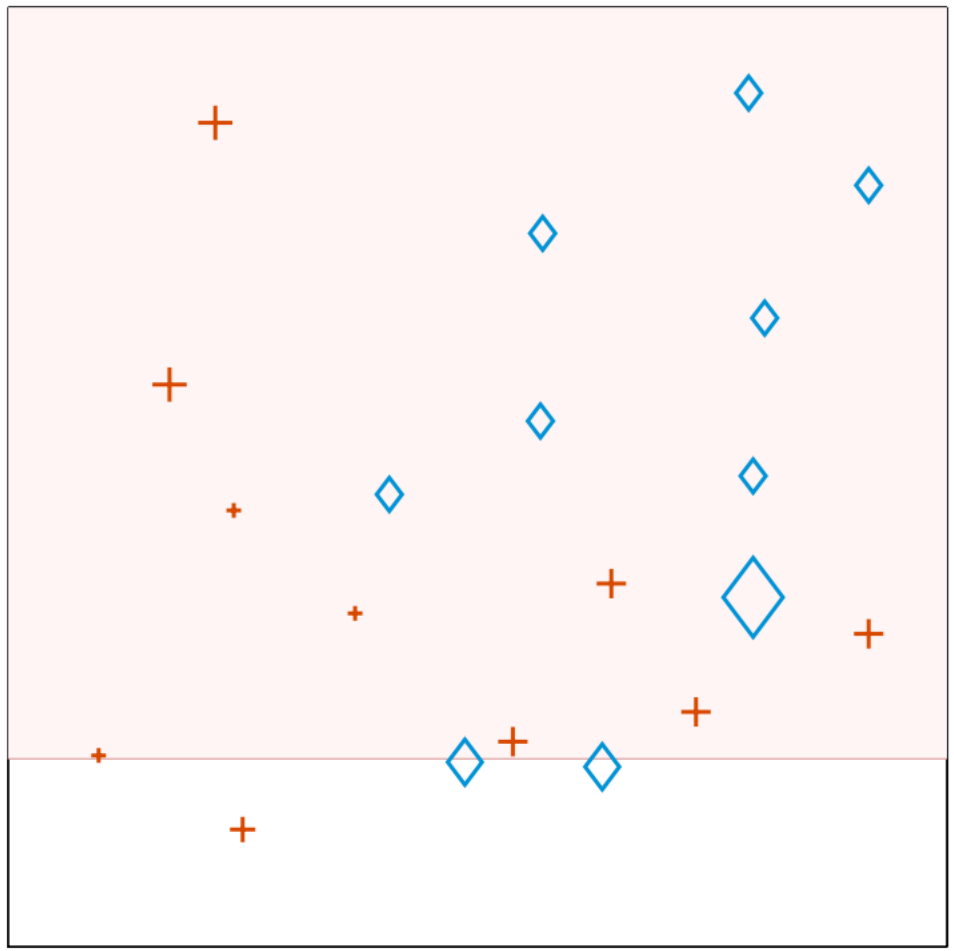
Weighted error
$e_t = 0.25$

Voting weight
$\alpha_t = 1.11$

Total error = 1

AdaBoost: Step-By-Step

- **Iteration 3, recompute weights**



Threshold
$\theta^* = 0.14$

Dimension, sign
$j^* = 2$, neg

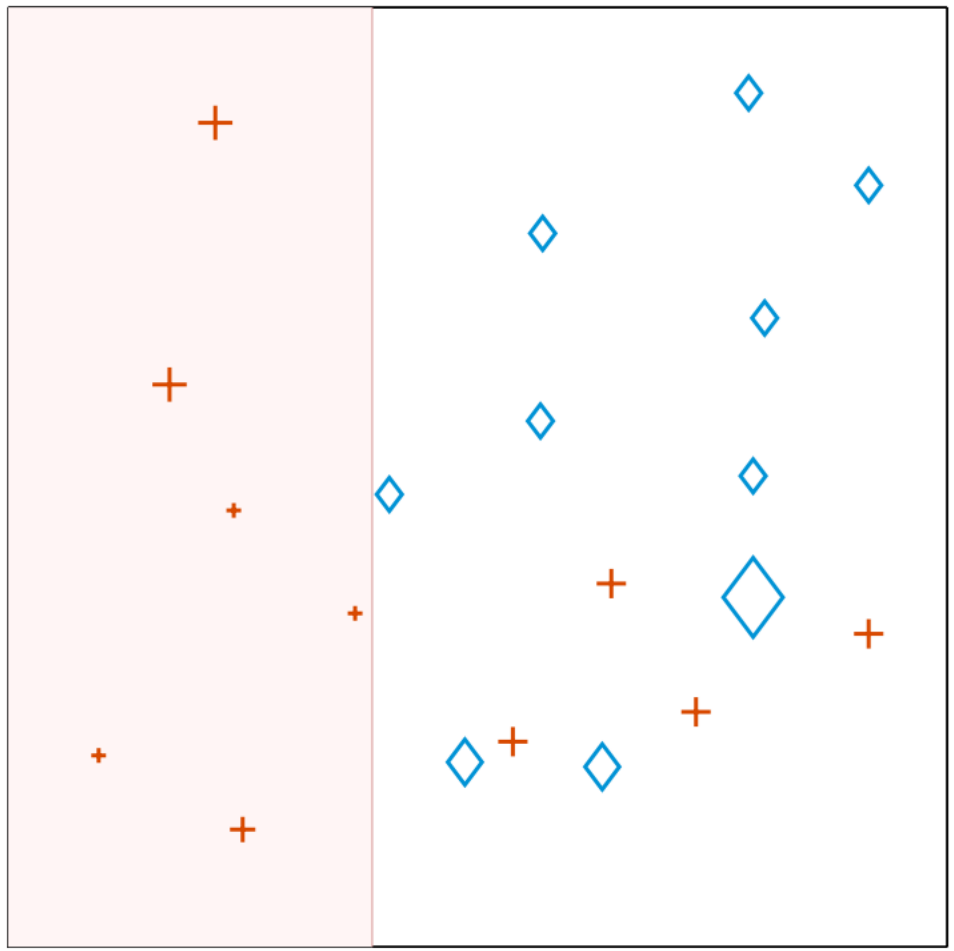
Weighted error
$e_t = 0.25$

Voting weight
$\alpha_t = 1.11$

Total error = 1

AdaBoost: Step-By-Step

- **Iteration 4, train weak classifier 4**



Threshold
$\theta^* = 0.37$

Dimension
$j^* = 1$

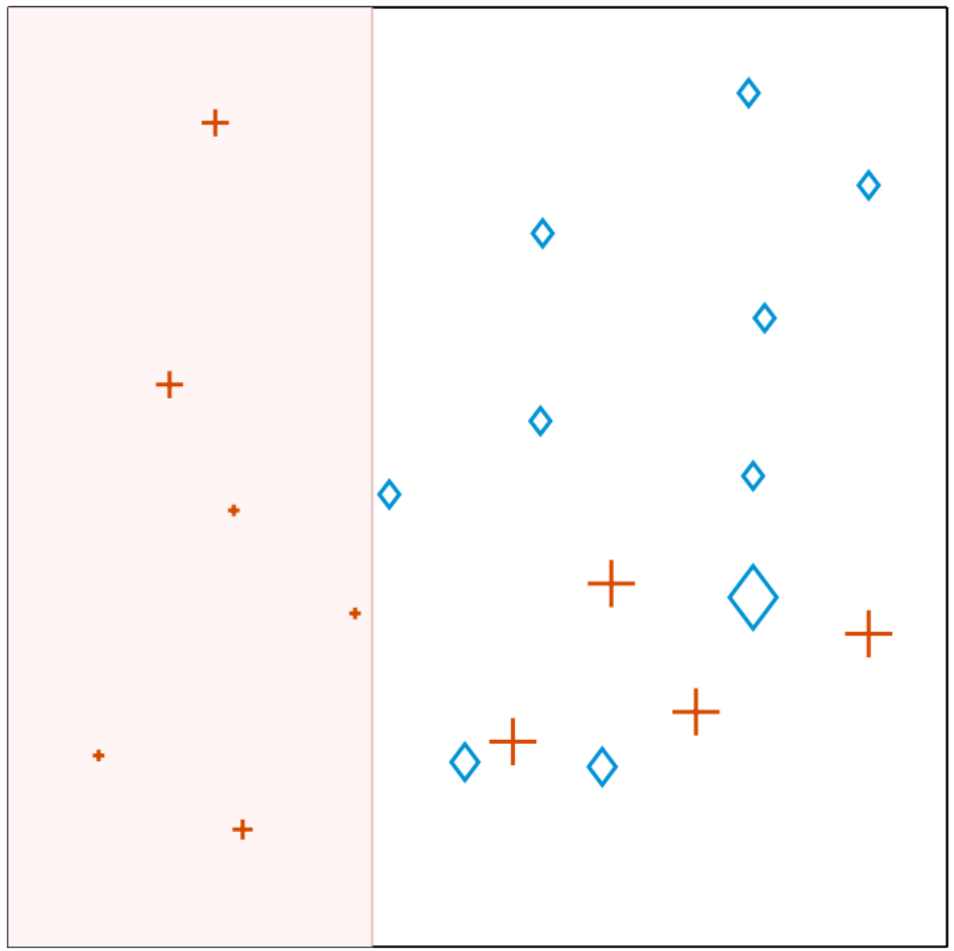
Weighted error
$e_t = 0.20$

Voting weight
$\alpha_t = 1.40$

Total error = 1

AdaBoost: Step-By-Step

- **Iteration 4, recompute weights**



Threshold
$\theta^* = 0.37$

Dimension
$j^* = 1$

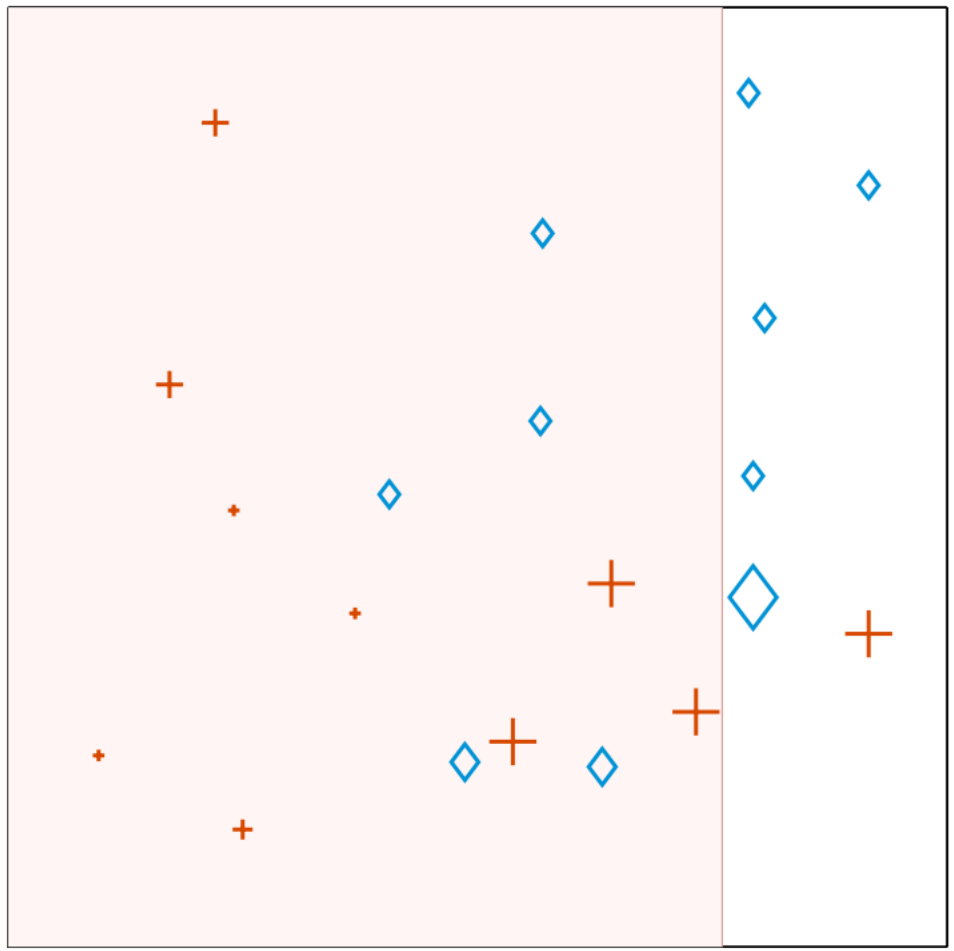
Weighted error
$e_t = 0.20$

Voting weight
$\alpha_t = 1.40$

Total error = 1

AdaBoost: Step-By-Step

- **Iteration 5, train weak classifier 5**



Threshold
$\theta^* = 0.81$

Dimension
$j^* = 1$

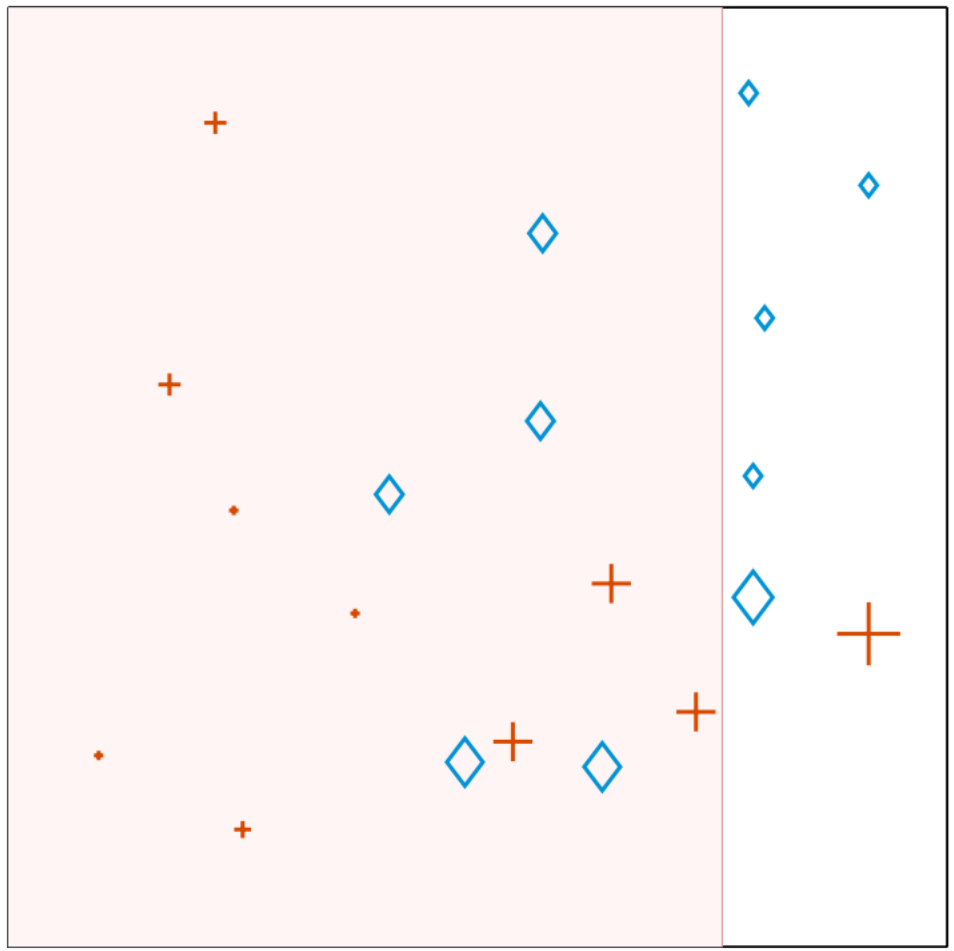
Weighted error
$e_t = 0.28$

Voting weight
$\alpha_t = 0.96$

Total error = 1

AdaBoost: Step-By-Step

- **Iteration 5, recompute weights**



Threshold
$\theta^* = 0.81$

Dimension
$j^* = 1$

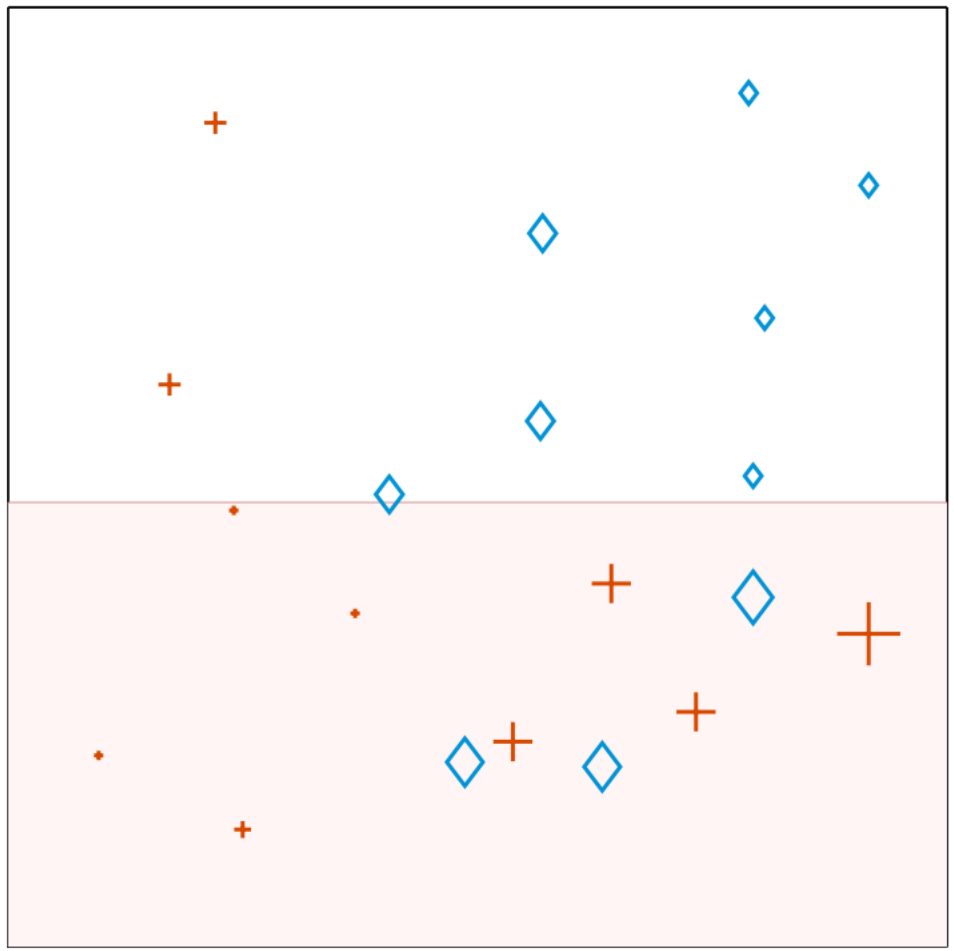
Weighted error
$e_t = 0.28$

Voting weight
$\alpha_t = 0.96$

Total error = 1

AdaBoost: Step-By-Step

- **Iteration 6, train weak classifier 6**



Threshold
$\theta^* = 0.47$

Dimension
$j^* = 2$

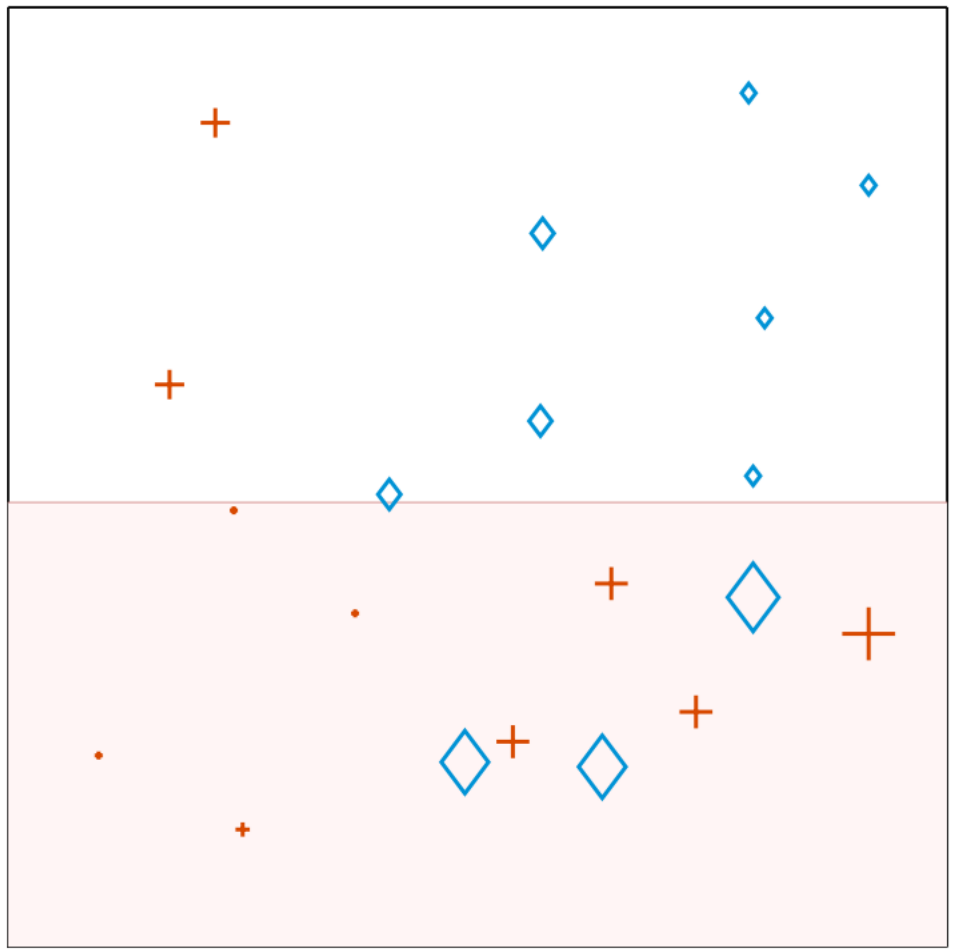
Weighted error
$e_t = 0.29$

Voting weight
$\alpha_t = 0.88$

Total error = 1

AdaBoost: Step-By-Step

- **Iteration 6, recompute weights**



Threshold
$\theta^* = 0.47$

Dimension
$j^* = 2$

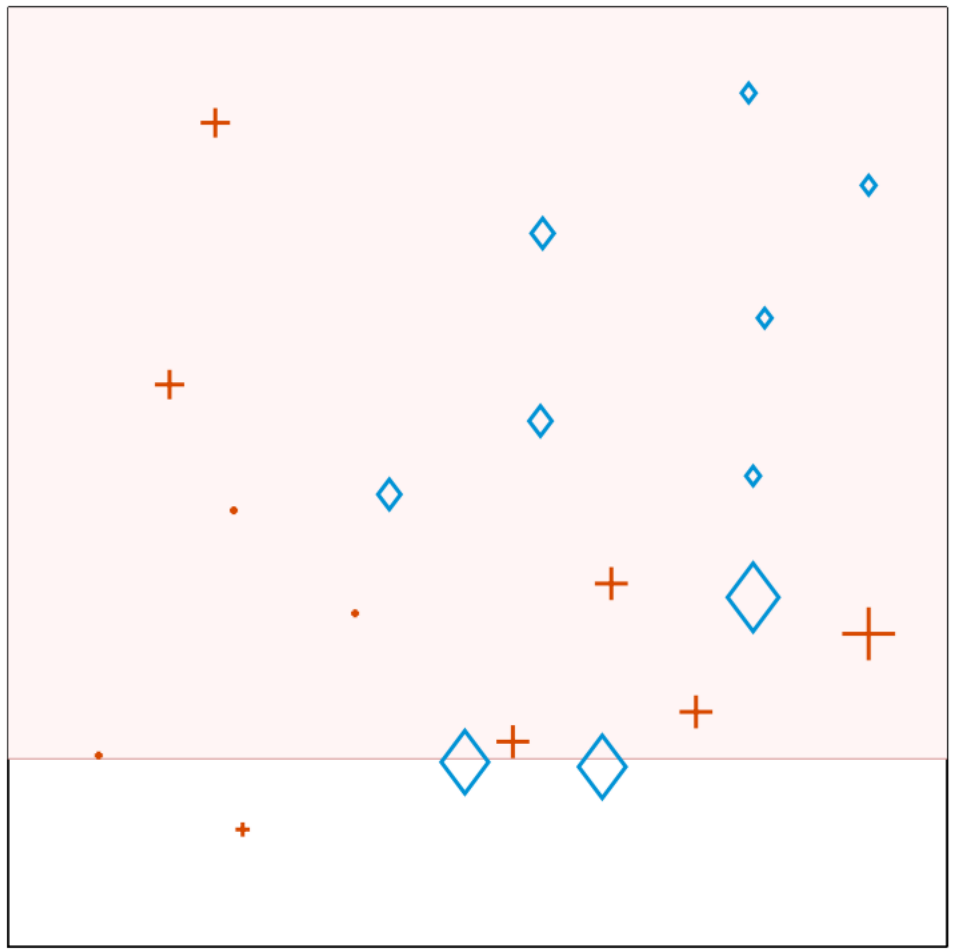
Weighted error
$e_t = 0.29$

Voting weight
$\alpha_t = 0.88$

Total error = 1

AdaBoost: Step-By-Step

- **Iteration 7, train weak classifier 7**



Threshold
$\theta^* = 0.14$

Dimension, sign
$j^* = 2$, neg

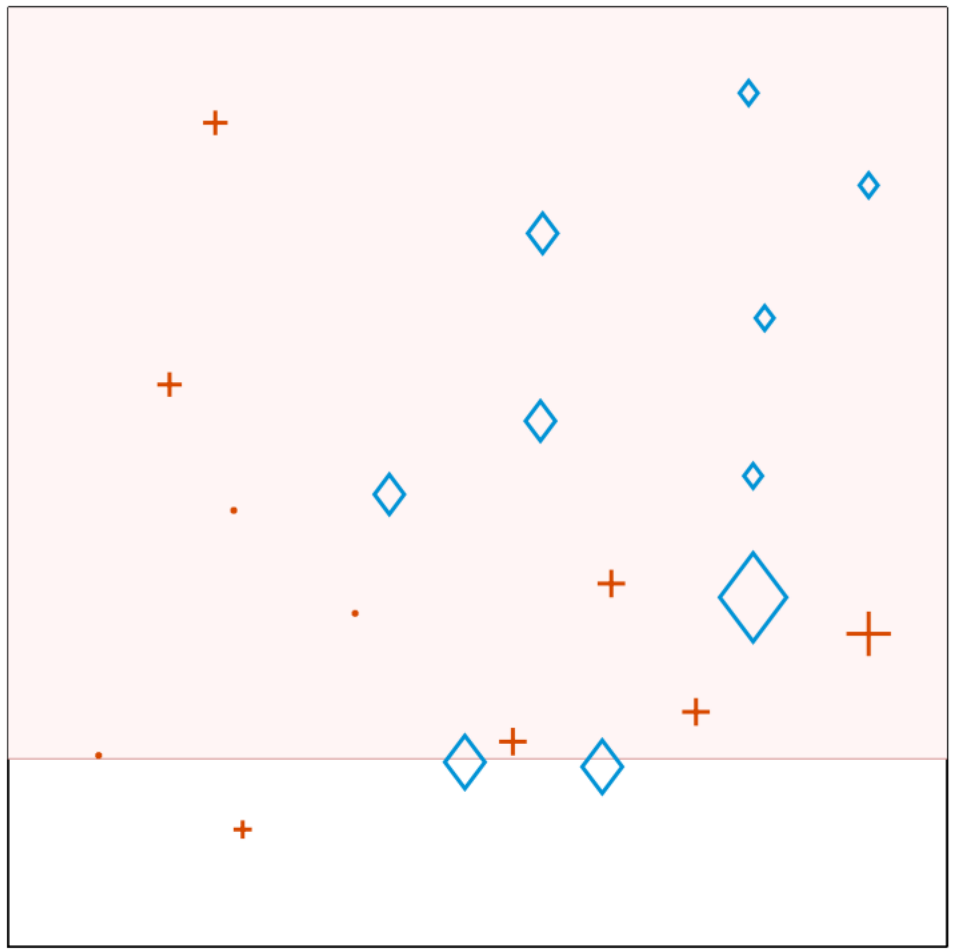
Weighted error
$e_t = 0.29$

Voting weight
$\alpha_t = 0.88$

Total error = 1

AdaBoost: Step-By-Step

- **Iteration 7, recompute weights**



Threshold
$\theta^* = 0.14$

Dimension, sign
$j^* = 2$, neg

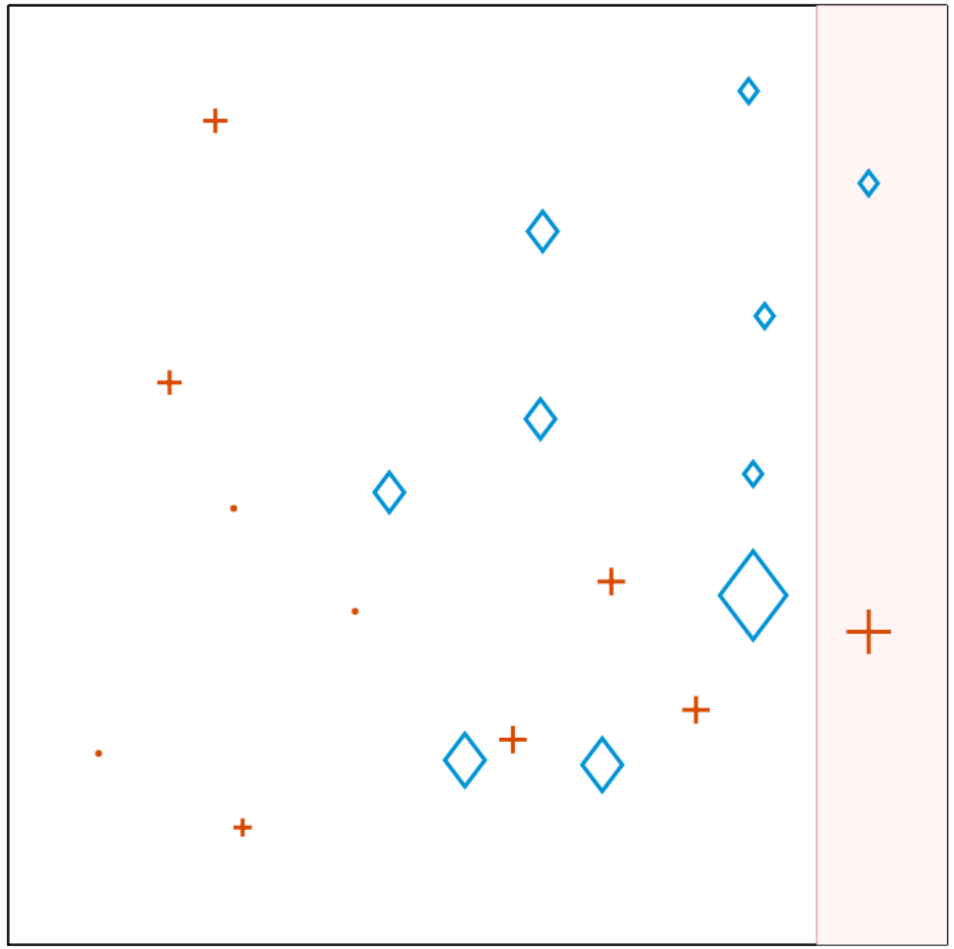
Weighted error
$e_t = 0.29$

Voting weight
$\alpha_t = 0.88$

Total error = 1

AdaBoost: Step-By-Step

- **Iteration 8, train weak classifier 8**



Threshold
$\theta^* = 0.93$

Dimension, sign
$j^* = 1$, neg

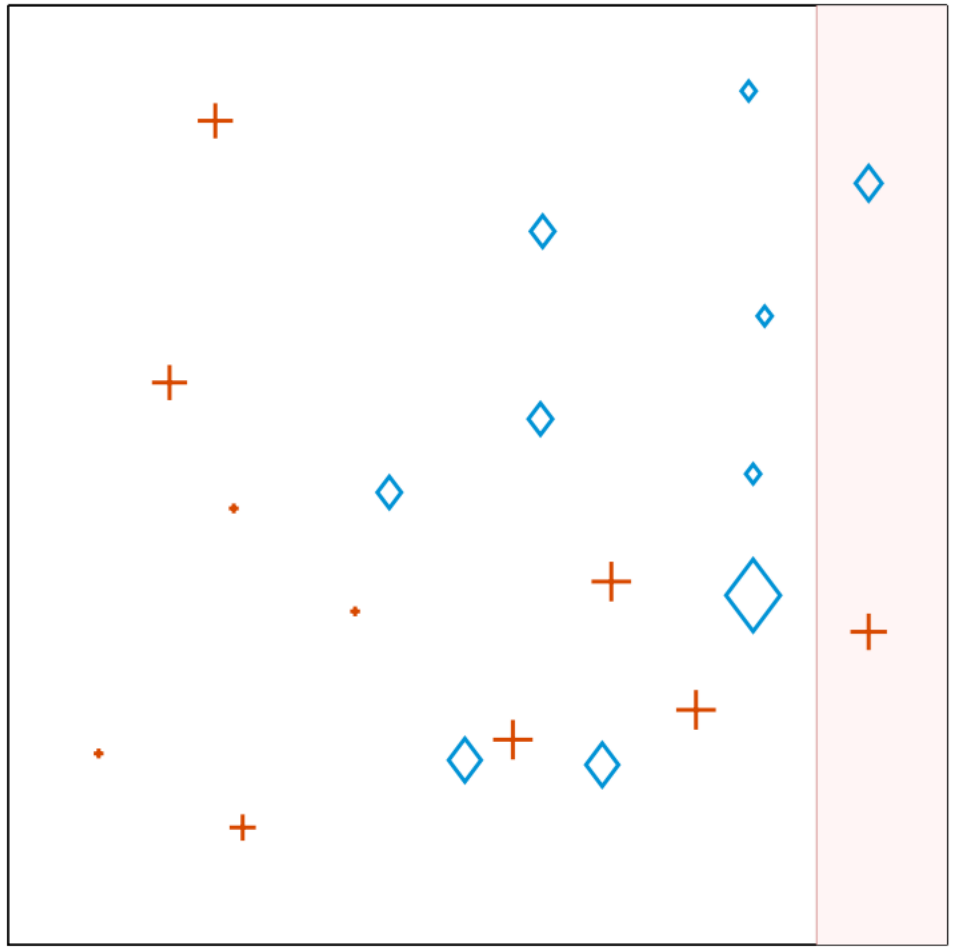
Weighted error
$e_t = 0.25$

Voting weight
$\alpha_t = 1.12$

Total error = 0

AdaBoost: Step-By-Step

- **Iteration 8, recompute weights**



Threshold
$\theta^* = 0.93$

Dimension, sign
$j^* = 1$, neg

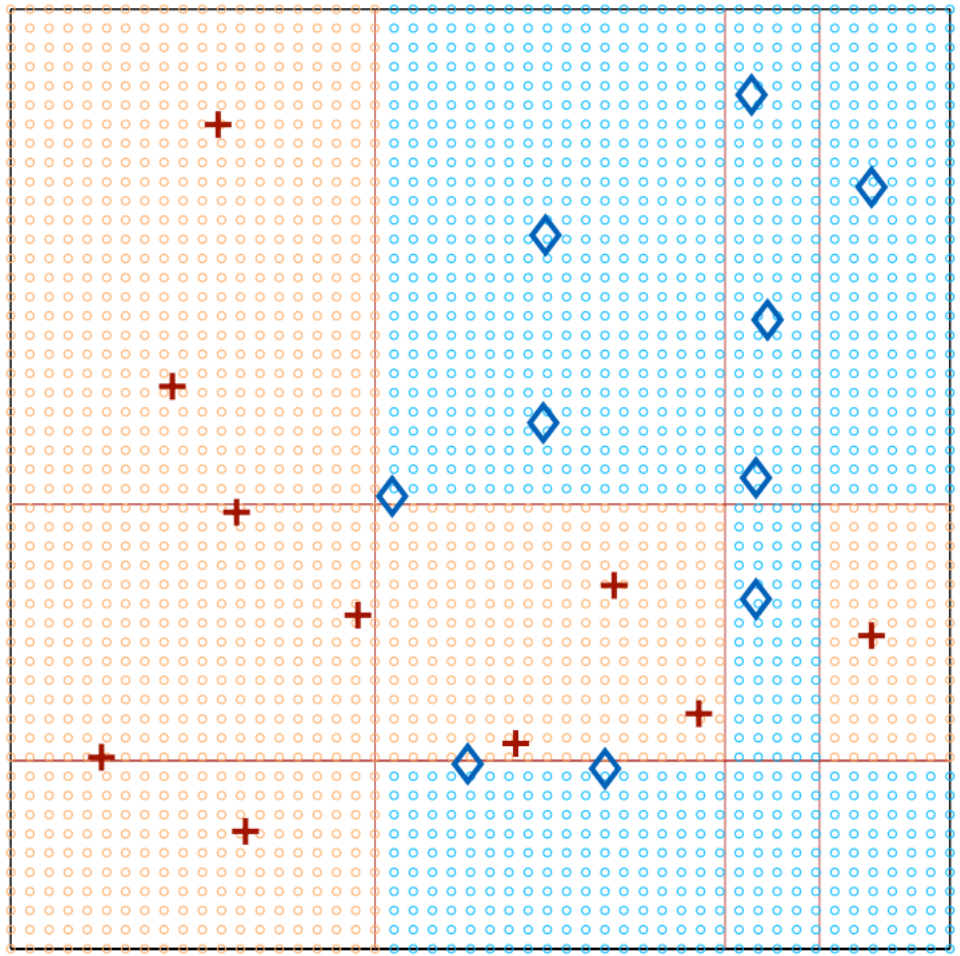
Weighted error
$e_t = 0.25$

Voting weight
$\alpha_t = 1.12$

Total error = 0

AdaBoost: Step-By-Step

- **Final Strong Classifier**



Total training error = 0

(Rare in practice)

AdaBoost Summary

- **Misclassified samples receive higher weight**. The higher the weight the "more attention" a training sample receives
- Algorithm generates weak classifier by training the next learner **on the mistakes** of the previous one
- AdaBoost minimizes **the upper bound of the training error** by properly choosing the optimal weak classifier and voting weight. AdaBoost can further be shown to maximize the margin
- **Large impact** on ML community and beyond

Slides from Arras et. al.

AdaBoost Example in Robotics

- People detection and tracking is a key component for many vision systems and for all robots in human environments

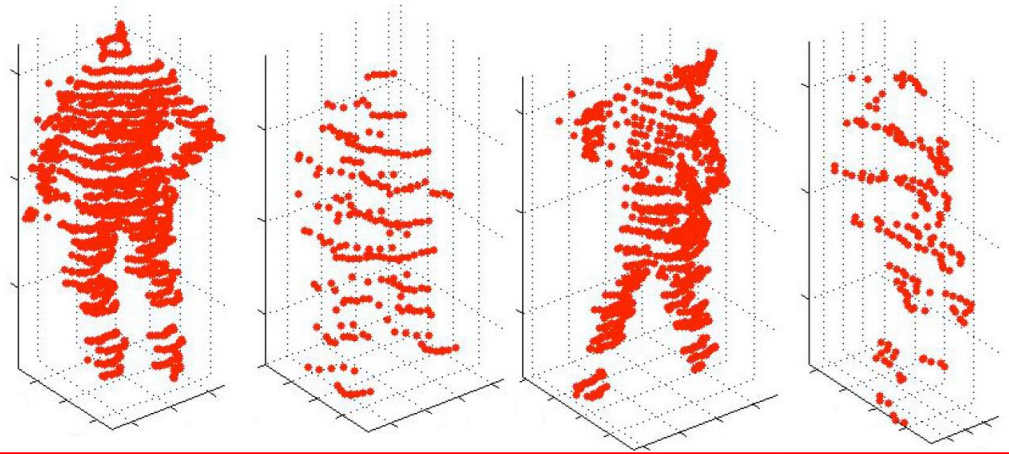
- **Where are the people?**
- **Why is it hard?**

  Range data contain little information on people

  Hard in cluttered environments

- **2D range data** from a SICK laser scanner

- Appearance of humans in **3D range data** (Velodyne scanner)



Slides from Arras et. al.

AdaBoost-based People Detection in 2D Range Scans

- Can we find **robust features** for people, legs and groups of people in 2D range data?
- What are the **best features** for people detection?
- Can we find people that **do not move**?

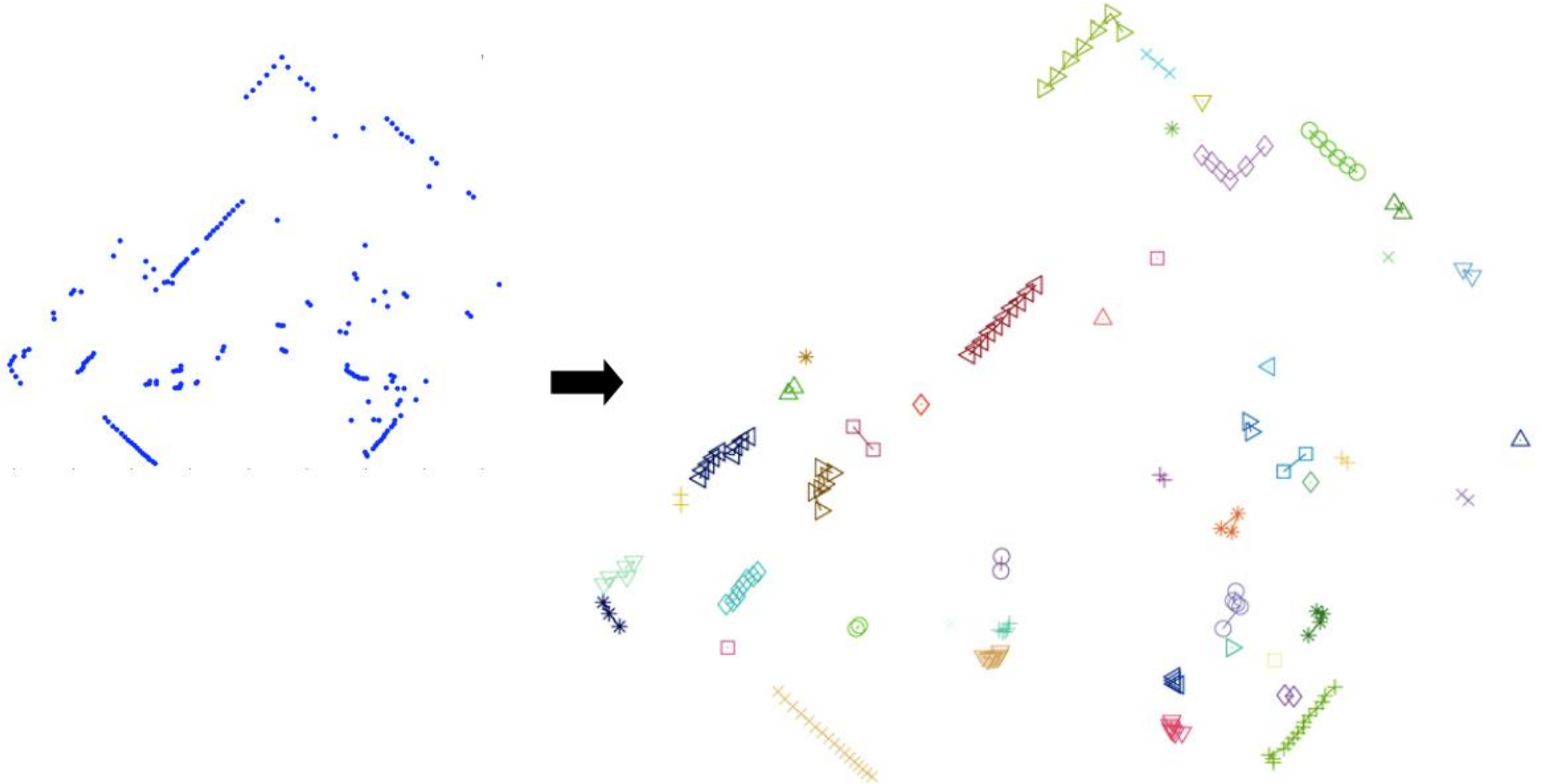
Approach:

- Classifying **groups of adjacent** beams (segments)
- Computing a set of scalar features on these groups
- **Boosting the features**

Slides from Arras et. al.

Segmentation

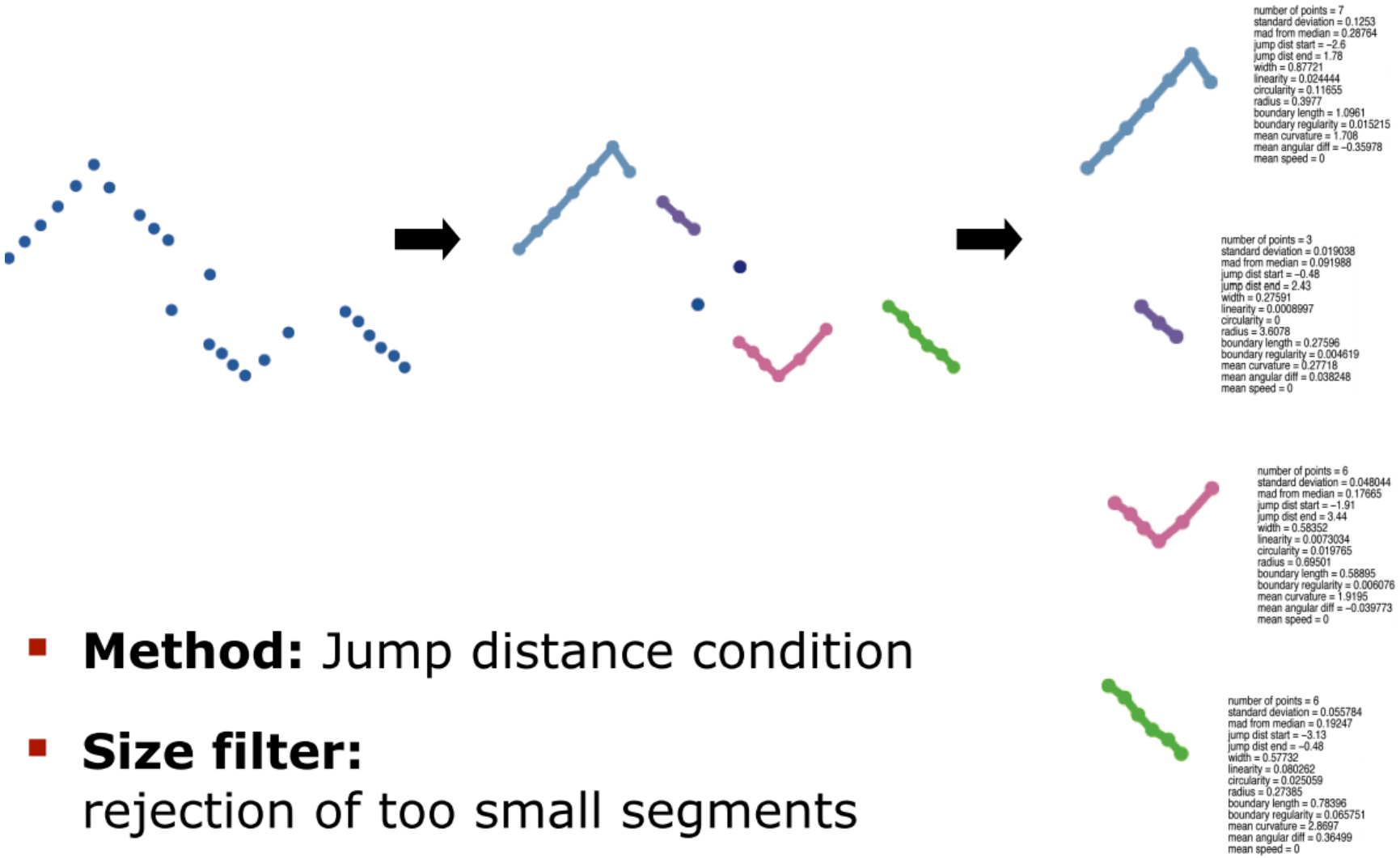
- Divide the scan into segments



Range image segmentation

Slides from Arras et. al.

Segmentation

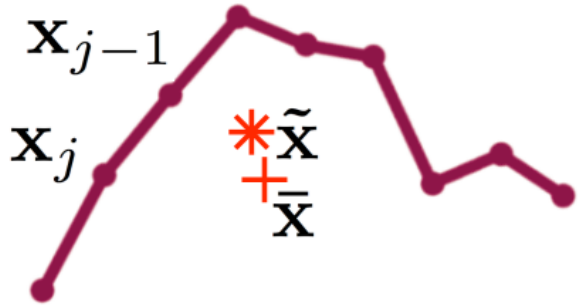


- **Method:** Jump distance condition
- **Size filter:**
  rejection of too small segments

Slides from Arras et. al.

Features per Segment

Segment S_i

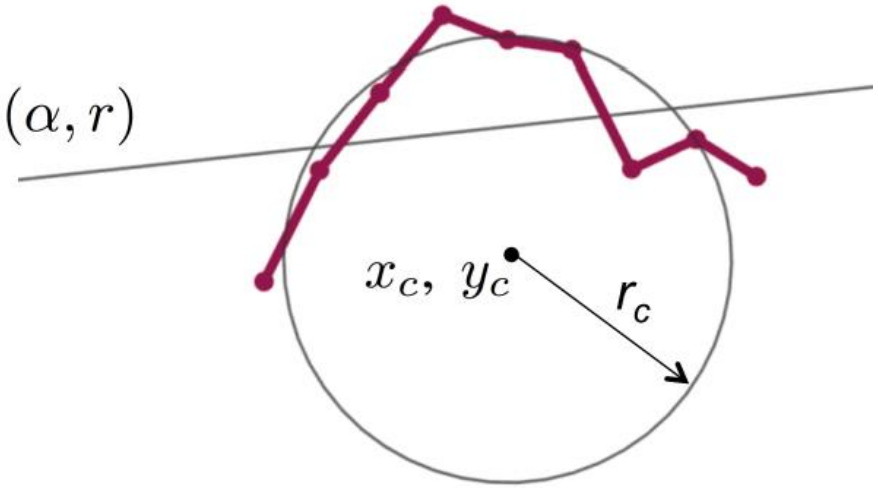


1. Number of points $n = |S_i|$
2. Standard Deviation $\sigma = \sqrt{\frac{1}{n-1}\sum ||\mathbf{x}_j - \bar{\mathbf{x}}||^2}$
3. Mean avg. deviation from median $\varsigma = \frac{1}{n}\sum ||\mathbf{x}_j - \tilde{\mathbf{x}}||$
4. Jump dist. to preceding segment $\delta_{j-1,j}$
5. Jump dist. to succeeding segment $\delta_{j,j+1}$
6. Width $w_i = ||\mathbf{x}_1 - \mathbf{x}_n||$

Slides from Arras et. al.

Features per Segment

Segment S_i



7. Linearity $s_l = \sum (x_j cos(\alpha) + y_j sin(\alpha) - r)^2$

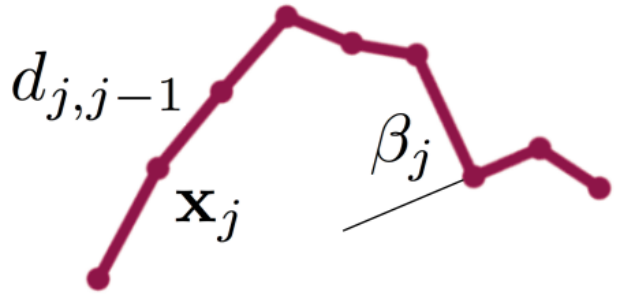
8. Circularity $s_c = \sum (r_c - \sqrt{(x_c - x_i)^2 + (y_c - y_i)^2})^2$

9. Radius r_c

Slides from Arras et. al.

Features per Segment

Segment S_i



10. Boundary Length $l = \sum_j d_{j,j-1}$

11. Boundary Regularity $\sigma_d = \sqrt{\frac{1}{n-1}\sum (d_{j,j-1} - \bar{d})^2}$

12. Mean curvature $\bar{k} = \frac{1}{n}\sum \hat{k}_j$

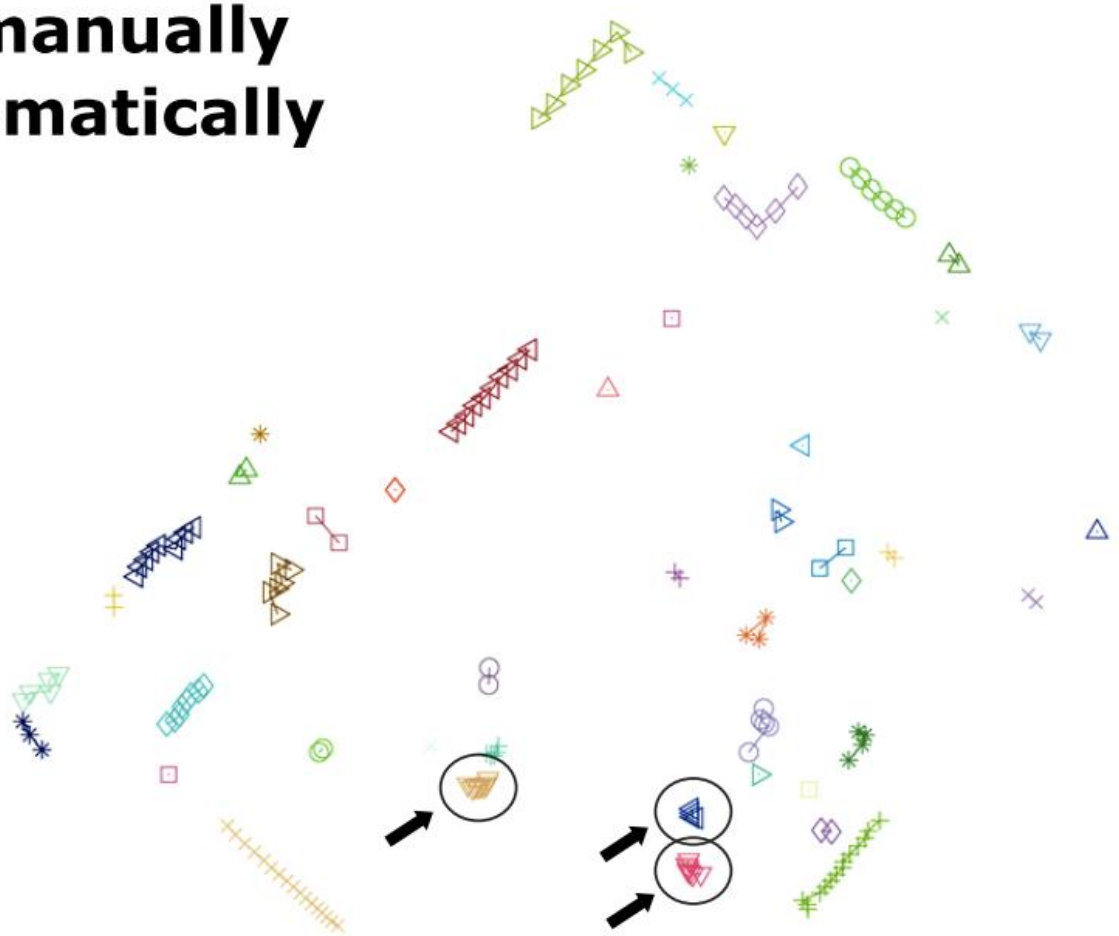
13. Mean angular difference $\bar{\beta} = \frac{1}{n}\sum \beta_j$

14. Mean speed $\bar{v} = \frac{1}{n}\sum \frac{\rho_j^{k+1} - \rho_j^k}{\Delta T}$

Slides from Arras et. al.

Data Labeling

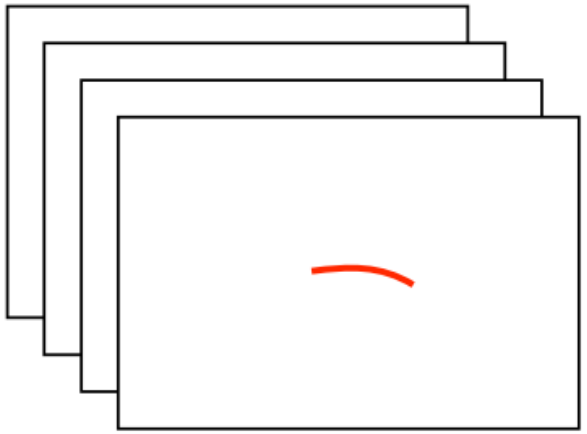
- **Mark segments** that correspond to people
- Either **manually** or **automatically**



Slides from Arras et. al.

Training

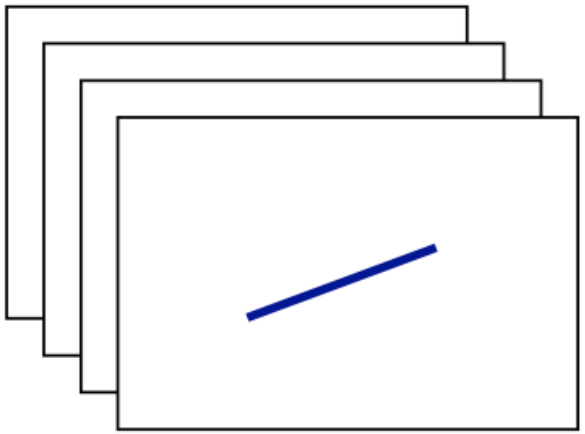
- Resulting **Training Set**



+1

Segments corresponding to people

(foreground segments)



-1

Segments corresponding to other objects

(background segments)

Slides from Arras et. al.

Evaluation

Env. 1: Corridor, no clutter

	Detected Label		
True Label	Person	No Person	Total
Person	**239** (99.58%)	**1** (0.42%)	**240**
No Person	**27** (1.03%)	**2589** (98.97%)	**2616**

Env. 2: Office, very cluttered

	Detected Label		
True Label	Person	No Person	Total
Person	**497** (97.45%)	**13** (2.55%)	**510**
No Person	**171** (2.73%)	**6073** (96.26%)	**6244**

Slides from Arras et. al.

Evaluation

Env. 1 & 2: Corridor and Office

	Detected Label		
True Label	Person	No Person	Total
Person	**722** (96.27%)	**28** (3.73%)	**750**
No Person	**225** (2.54%)	**8649** (99.88%)	**8860**

Env. 1→2: Cross-evaluation
Trained in corridor, applied in office

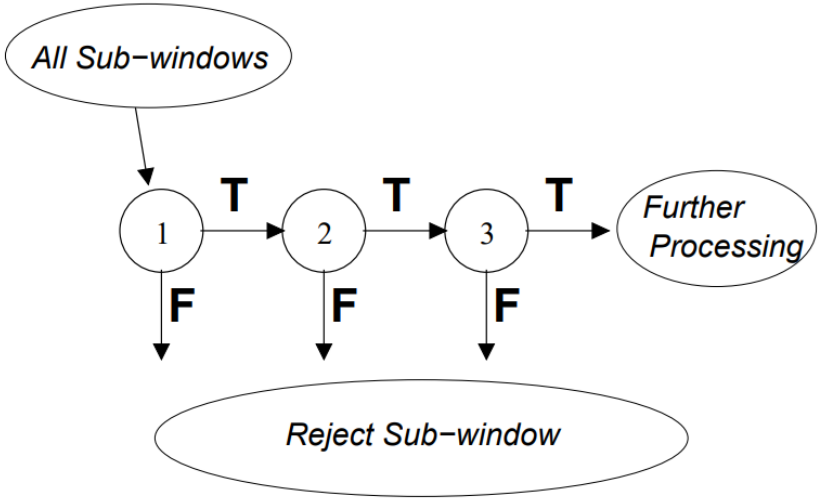
	Detected Label		
True Label	Person	No Person	Total
Person	**217** (90.42%)	**23** (9.58%)	**240**
No Person	**112** (4.28%)	**2504** (95.72%)	**2616**

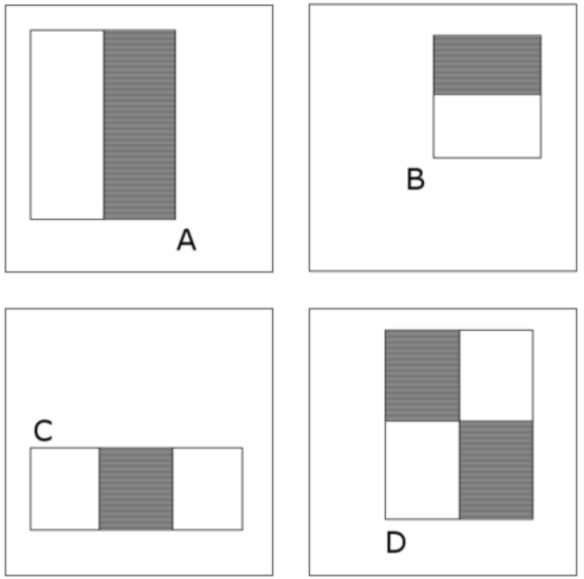
Slides from Arras et. al.

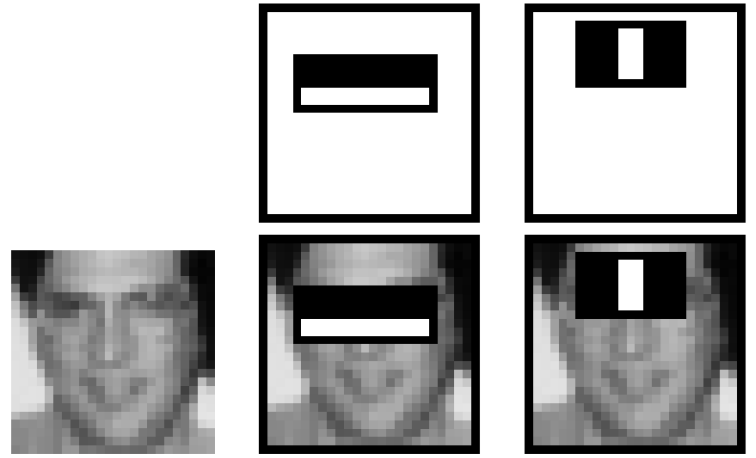
AdaBoost Example: Viola-Jones Object Detector

- **Haar Feature** Selection
- Creating an Integral Image
- AdaBoost Training

- Cascading Classifiers



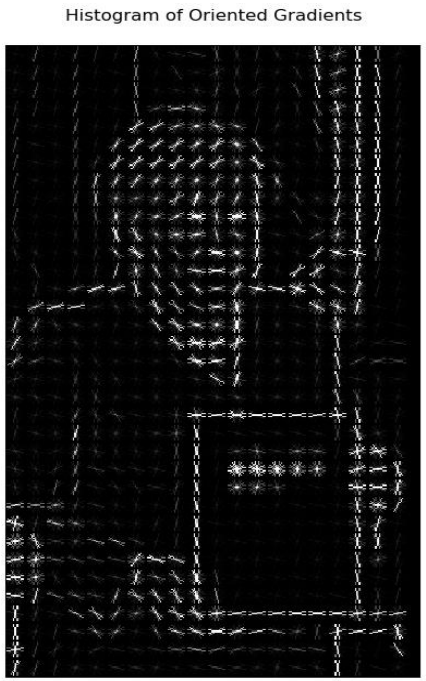




Viola, Paul, and Michael Jones. "Rapid object detection using a boosted cascade of simple features." *CVPR (1)* 1, no. 511-518 (2001): 3.

Histogram of Oriented Gradients (HOG)

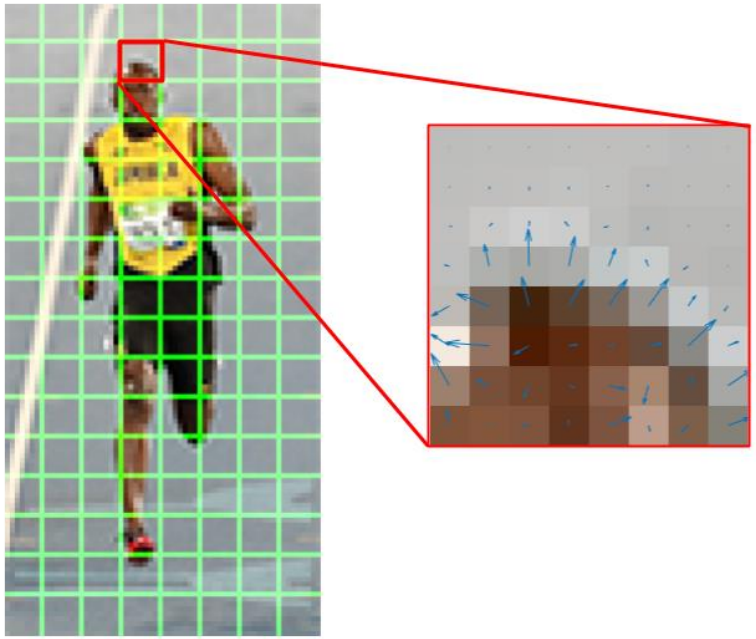
- Proposed by Dalal & Triggs in CVPR 2005 for detecting human in 2D images
- One of the most widely used feature descriptor for detecting human in 2D images



https://de.wikipedia.org/wiki/Histogram_of_oriented_gradients

Histogram of Oriented Gradients (HOG) - Algorithm

- Slide a detection window across different locations of an image.
- Divide detection window into small non-overlapping regions called cells
- Compute intensity gradient orientation histograms in each cell
- Human shapes can be described by local gradient orientations; local histograms are insensitive to small variations in translations and pose.



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

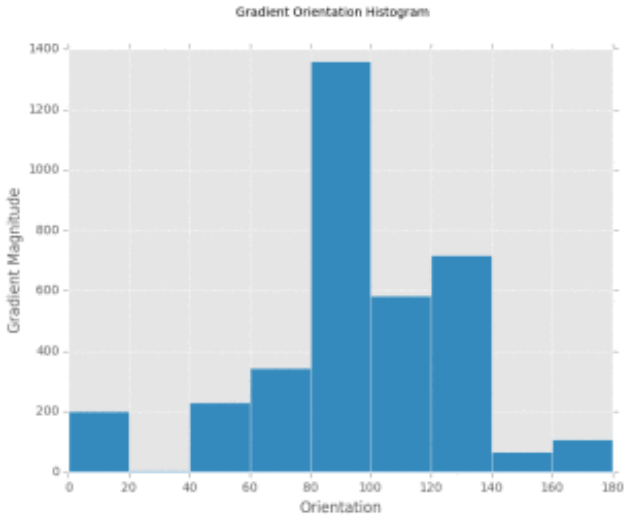
80	36	5	10	0	64	90	73
37	9	9	179	78	27	169	166
87	136	173	39	102	163	152	176
76	13	1	168	159	22	125	143
120	70	14	150	145	144	145	143
58	86	119	98	100	101	133	113
30	65	157	75	78	165	145	124
11	170	91	4	110	17	133	110

Gradient Direction

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

Histogram of Oriented Gradients (HOG) - Algorithm

- Normalize local gradient orientation histograms over larger local regions called blocks
  - For handling variations in illuminations and image contrast
  - Blocks are made up of cells and overlapping blocks are typically used in HOG



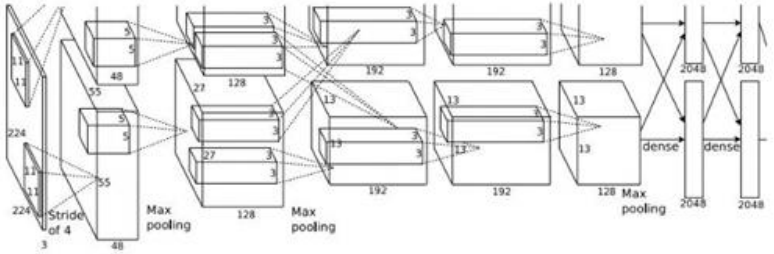
- The normalized histograms from the individual blocks are concatenated together to form the final HoG descriptor (feature vector)

https://gurus.pyimagesearch.com/lesson-sample-histogram-of-oriented-gradients-and-car-logo-recognition/

Object Detection using Deep Learning

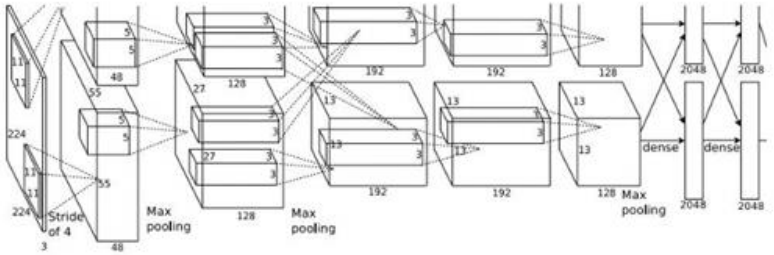
Object Detection by CNN – Challenges

Each image needs a different number of outputs



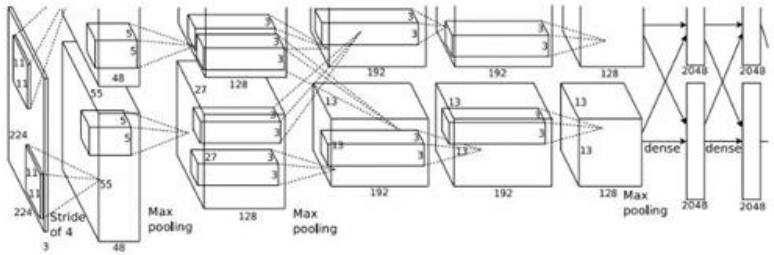
CAT: (x, y, w, h)

4 numbers



DOG: (x, y, w, h)
DOG: (x, y, w, h)
CAT: (x, y, w, h)

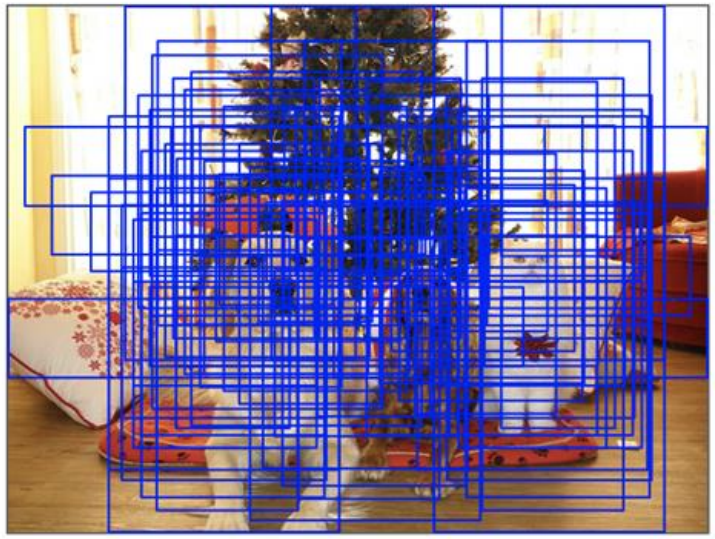
12 numbers



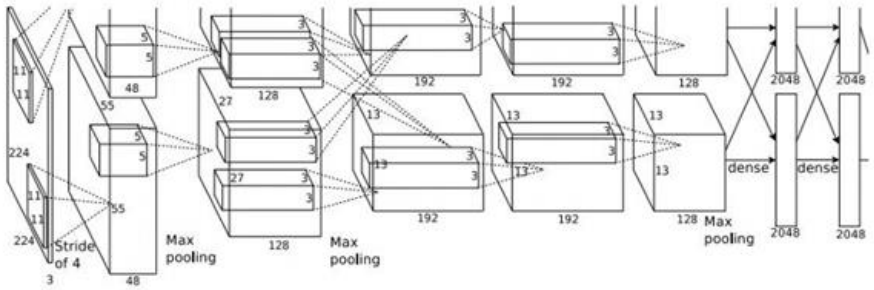
DUCK: (x, y, w, h)
DUCK: (x, y, w, h)
....

Many numbers!

Object Detection by CNN – Challenges



Apply a CNN to many different crops of the image, CNN classifies each crop as object or background

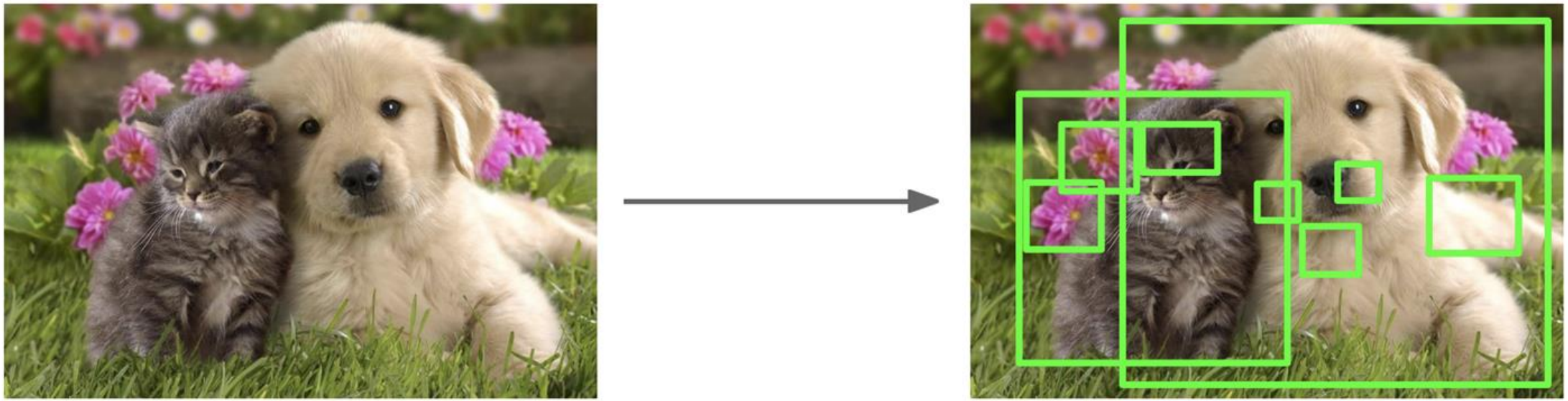


Dog? NO
Cat? YES
Background? NO

Problem: Need to apply CNN to huge number of locations, scales, and aspect ratios, very computationally expensive!

Region Proposals Can Help: Selective Search

- Find "blobby" image regions that are likely to contain objects
- Relatively fast to run; e.g. Selective Search gives 2000 region proposals in a few seconds on CPU



Alexe et al, "Measuring the objectness of image windows", TPAMI 2012
Uijlings et al, "Selective Search for Object Recognition", IJCV 2013
Cheng et al, "BING: Binarized normed gradients for objectness estimation at 300fps", CVPR 2014
Zitnick and Dollar, "Edge boxes: Locating object proposals from edges", ECCV 2014

Selective Search

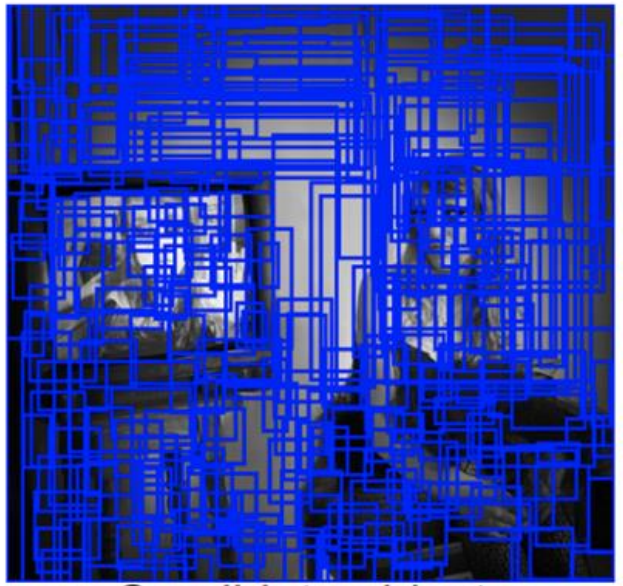
Step 1: Generate initial sub-segmentation

Goal: Generate many regions, each of which belongs to at most one object.

Using the method described by Felzenszwalb et al.

Input Image

Segmentation

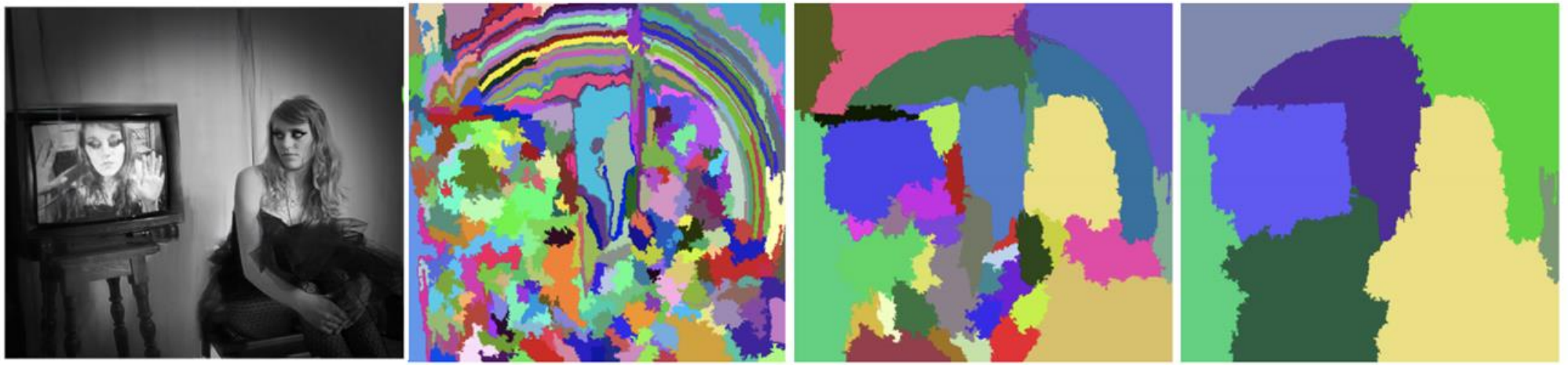


Candidate objects

[P. F. Felzenszwalb and D. P. Huttenlocher. “Efficient Graph-Based Image Segmentation.” IJCV, 59:167–181, 2004.]

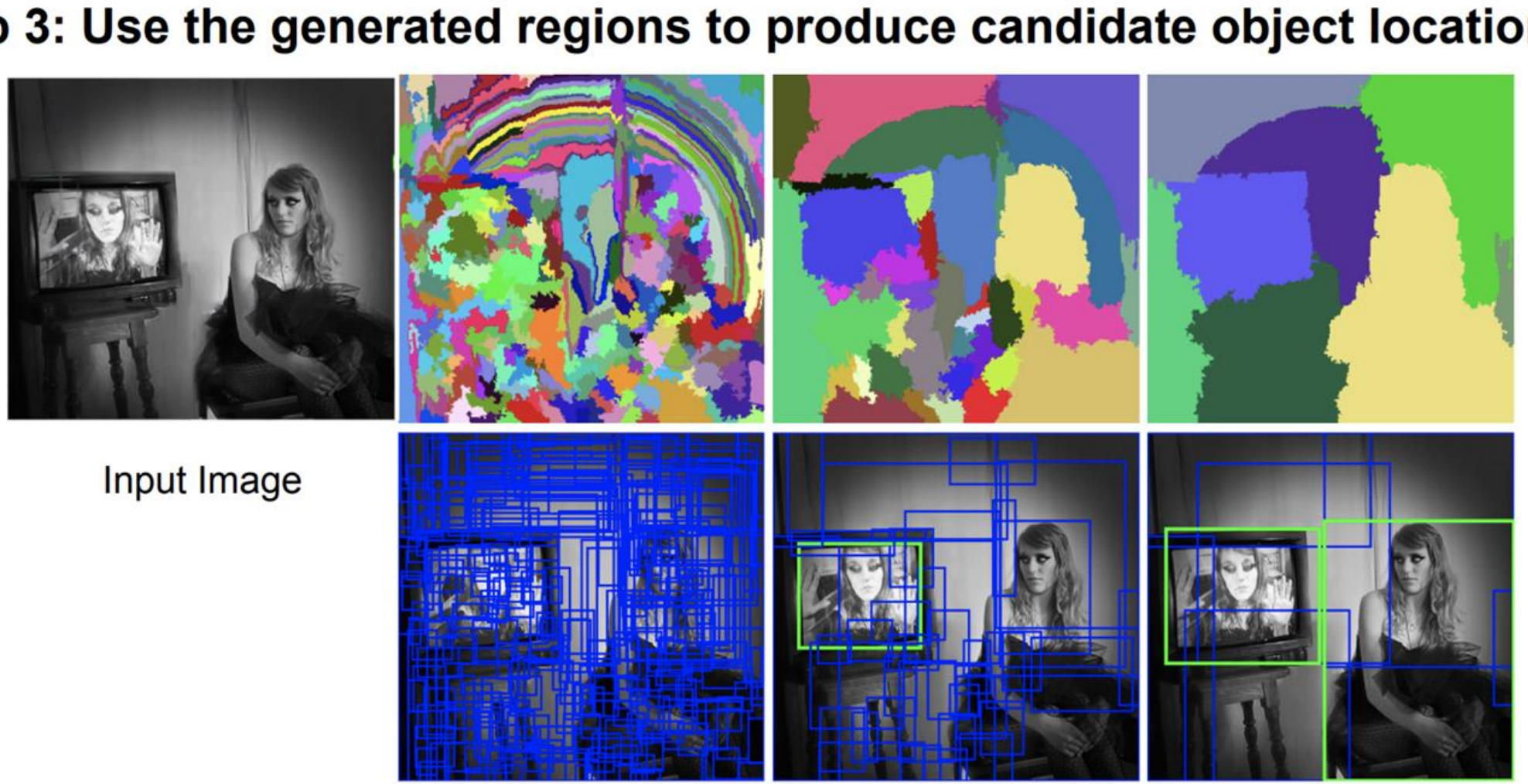
Selective Search

Step 2: Recursively combine similar regions into larger ones.



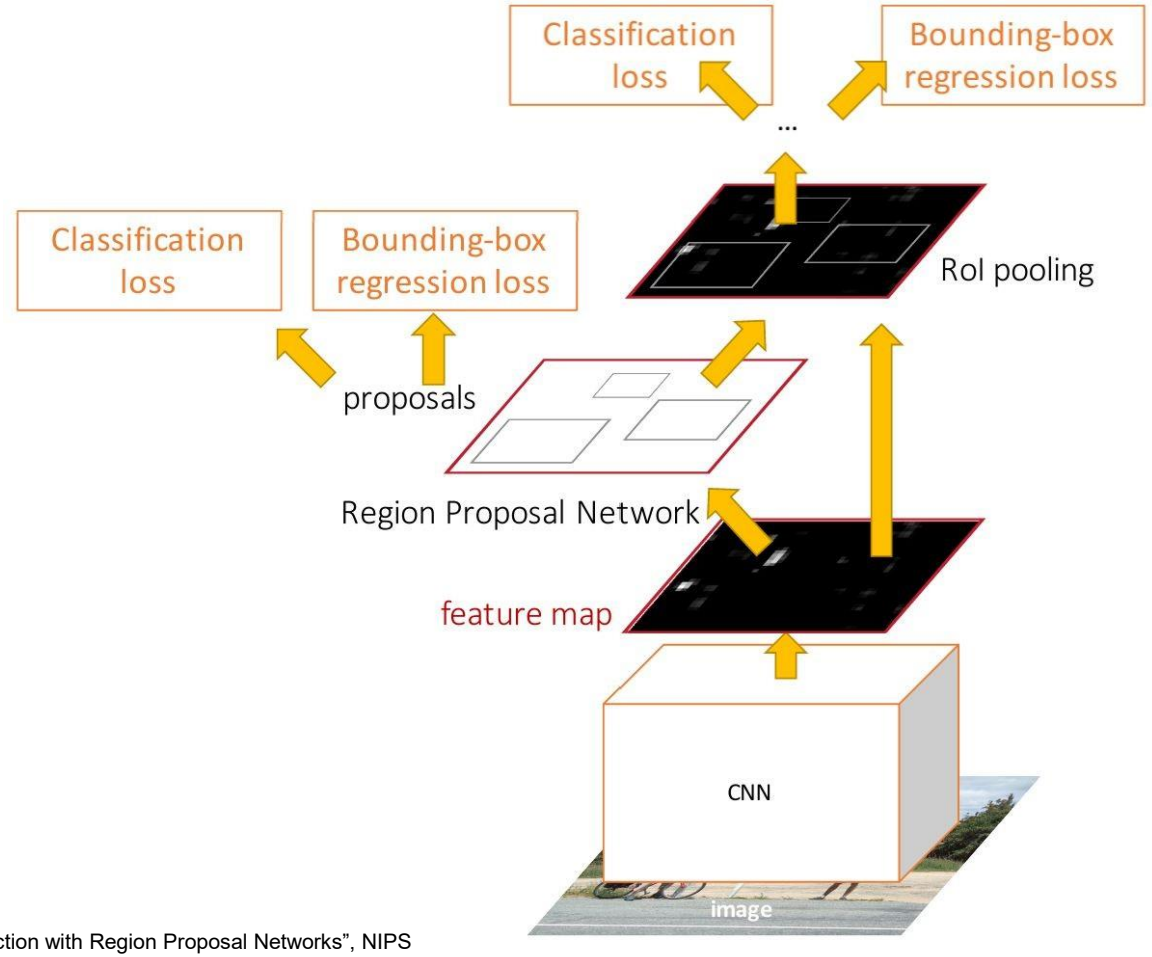
Input Image | Initial Segmentation | After some iterations | After more iterations

Selective Search



Faster R-CNN

- Make CNN do region proposal.



Ren et al, “Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks”, NIPS 2015

Region Proposal Network (RPN)

Input Image
(e.g. 3 x 640 x 480)

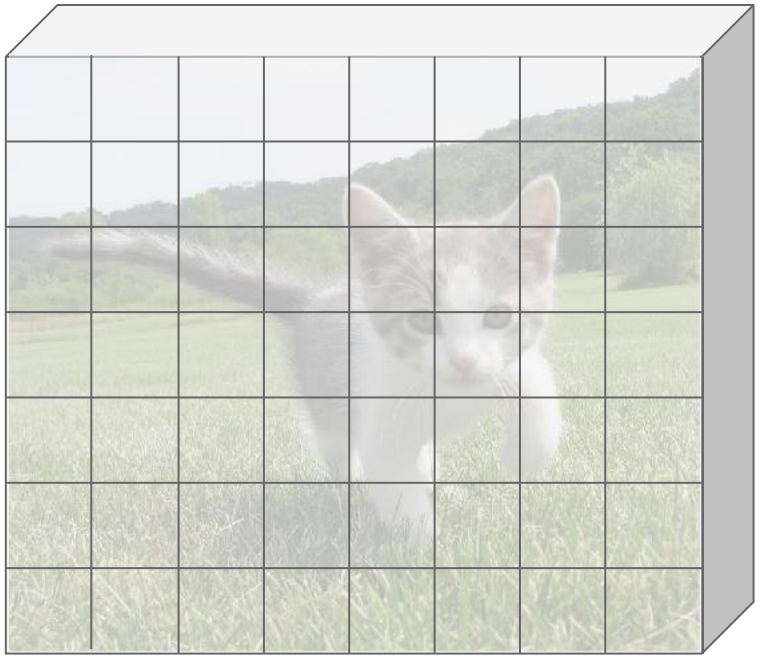


Image features
(e.g. 512 x 20 x 15)

Region Proposal Network: Anchor

Imagine an **anchor box** of fixed size at each point in the feature map

Input Image
(e.g. 3 x 640 x 480)

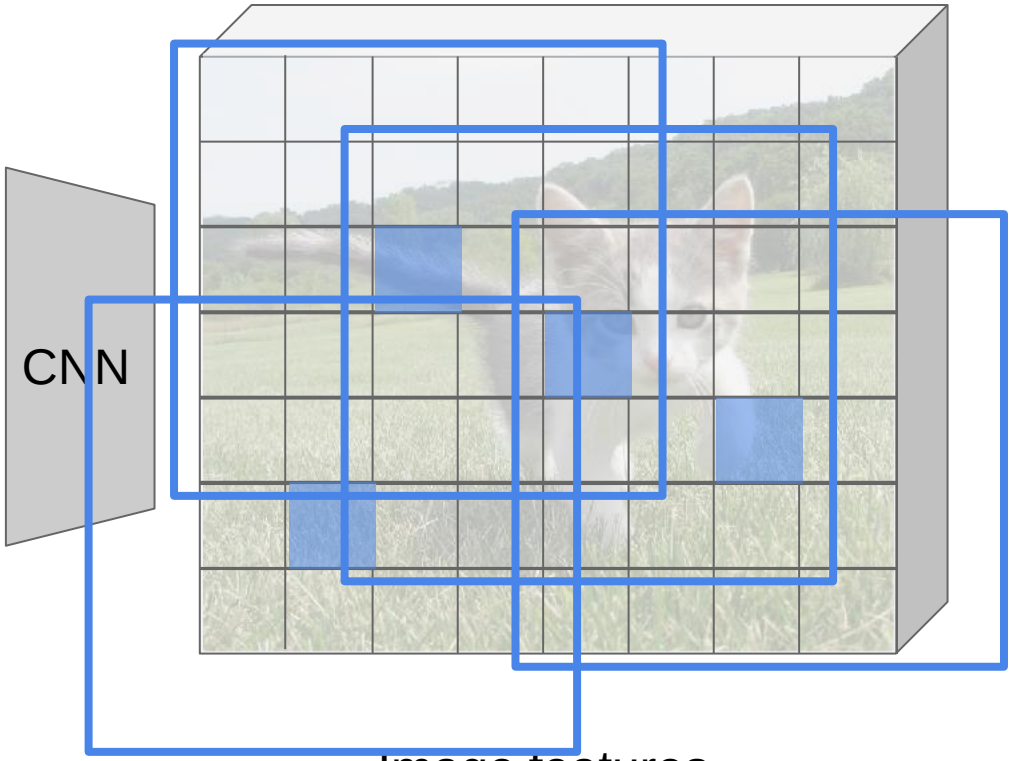


Image features
(e.g. 512 x 20 x 15)

Region Proposal Network: Anchor

Imagine an **anchor box** of fixed size at each point in the feature map

Input Image
(e.g. 3 x 640 x 480)

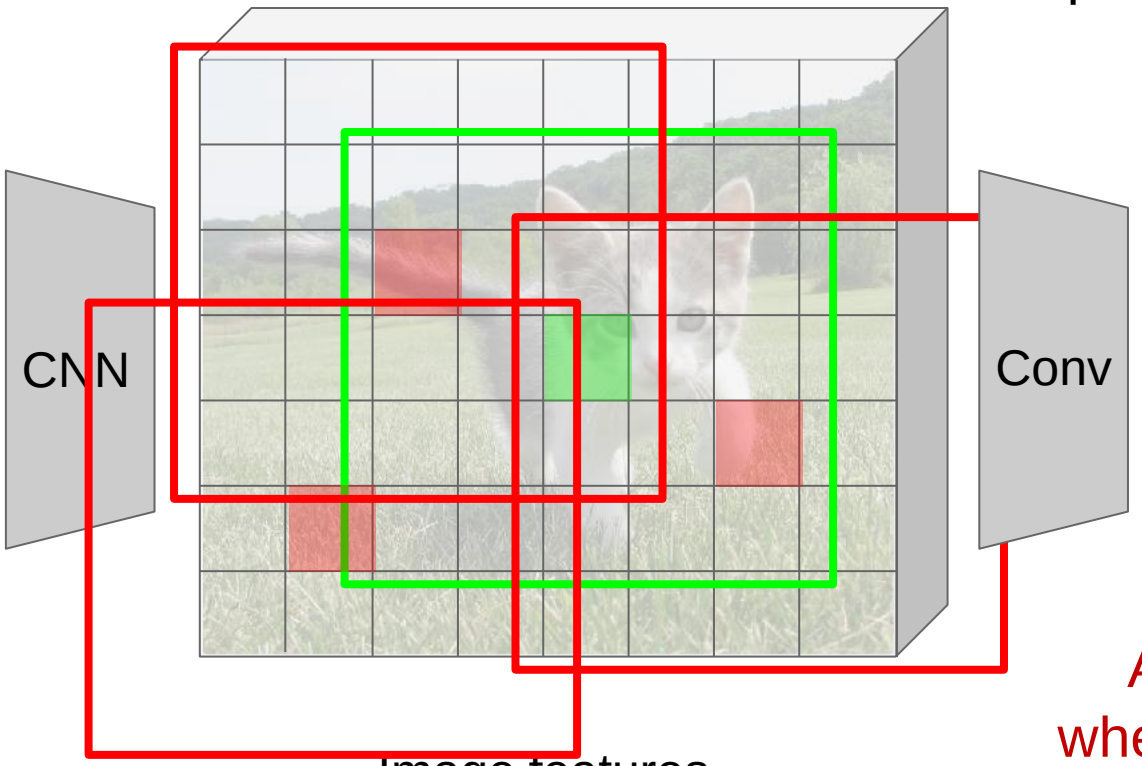


Image features
(e.g. 512 x 20 x 15)

Anchor is an object?
1 x 20 x 15

At each point, predict whether the corresponding anchor contains an object (binary classification)

Region Proposal Network: Anchor

Imagine an **anchor box** of fixed size at each point in the feature map

Input Image
(e.g. 3 x 640 x 480)

CNN

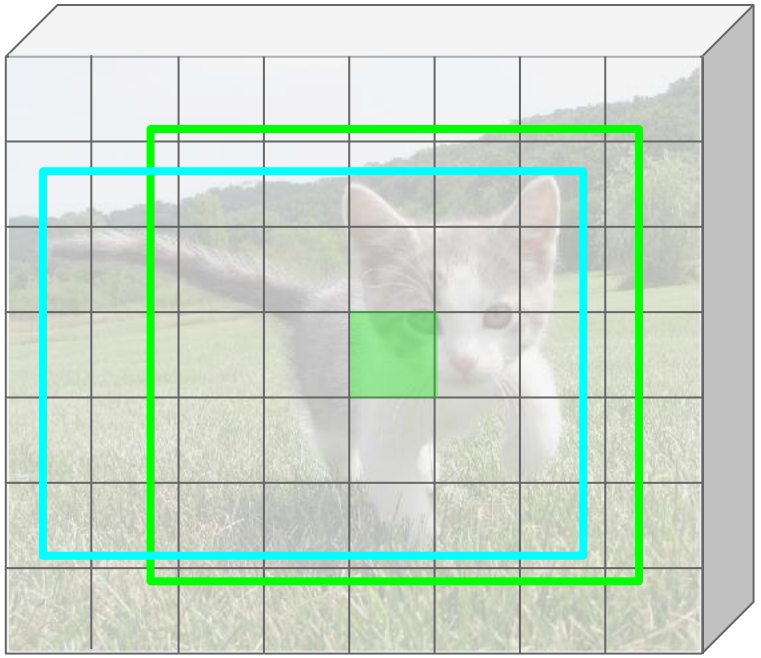


Image features
(e.g. 512 x 20 x 15)

Conv

Anchor is an object?
1 x 20 x 15

Box corrections
4 x 20 x 15

For positive boxes, also predict a corrections from the anchor to the ground-truth box (regress 4 numbers per pixel)

Region Proposal Network: Anchor

In practice use K different anchor boxes of different size / scale at each point

Input Image
(e.g. 3 x 640 x 480)

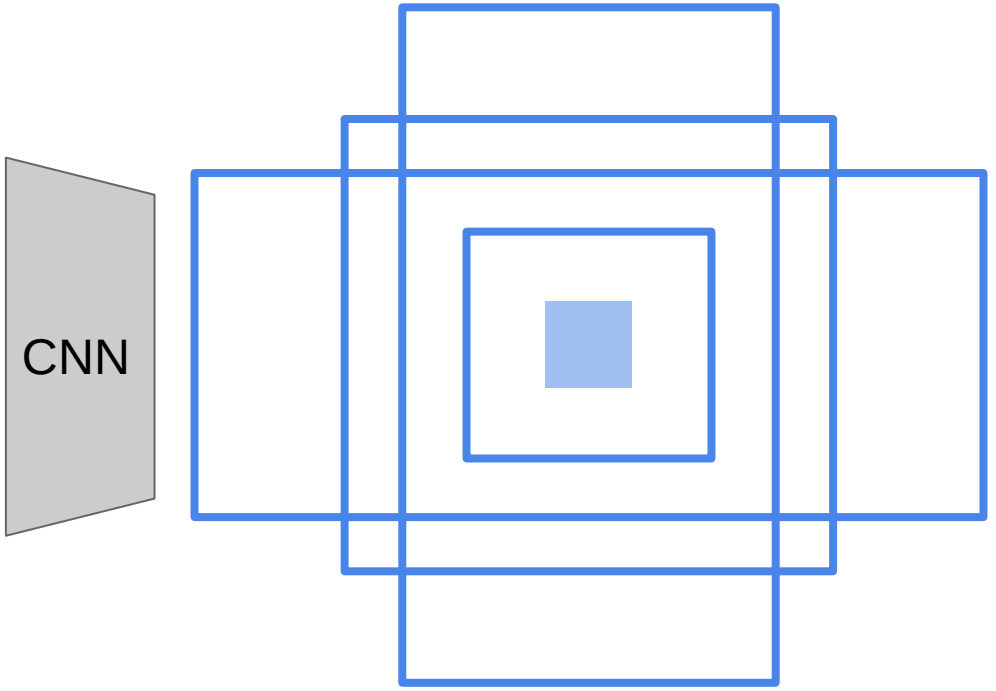


Image features
(e.g. 512 x 20 x 15)

Conv

→ Anchor is an object?
K x 20 x 15

→ Box transforms
4K x 20 x 15

Region Proposal Network: Anchor

In practice use K different anchor boxes of different size / scale at each point

Input Image
(e.g. 3 x 640 x 480)

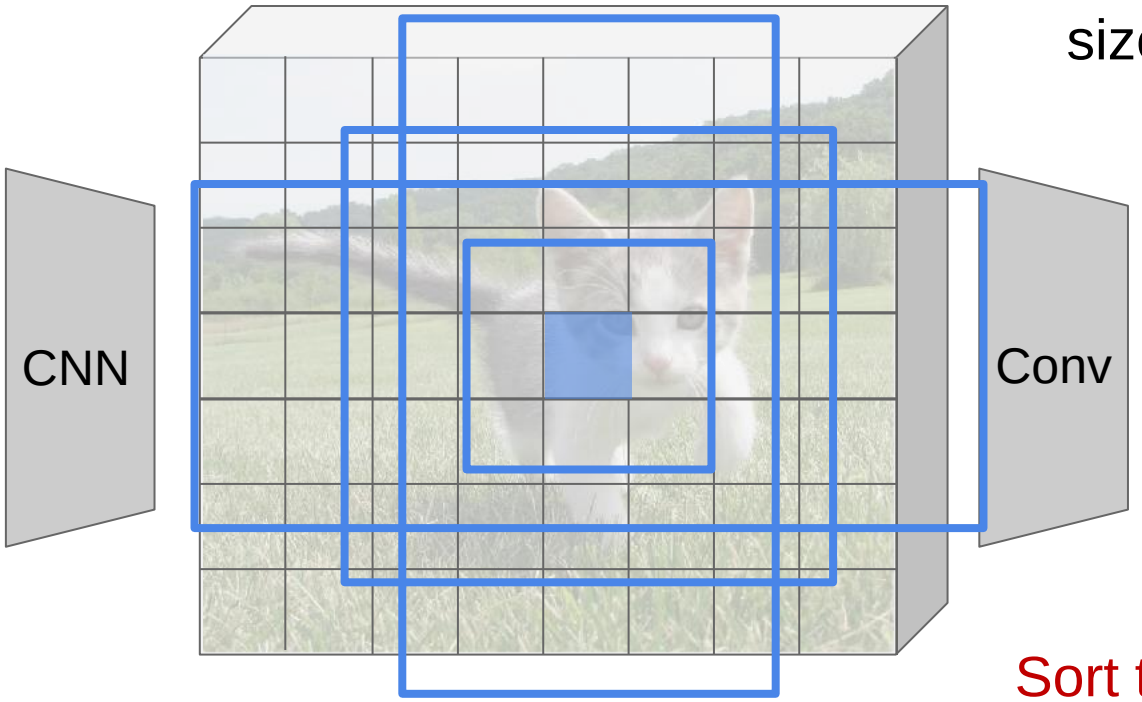


Image features
(e.g. 512 x 20 x 15)

Anchor is an object?
K x 20 x 15

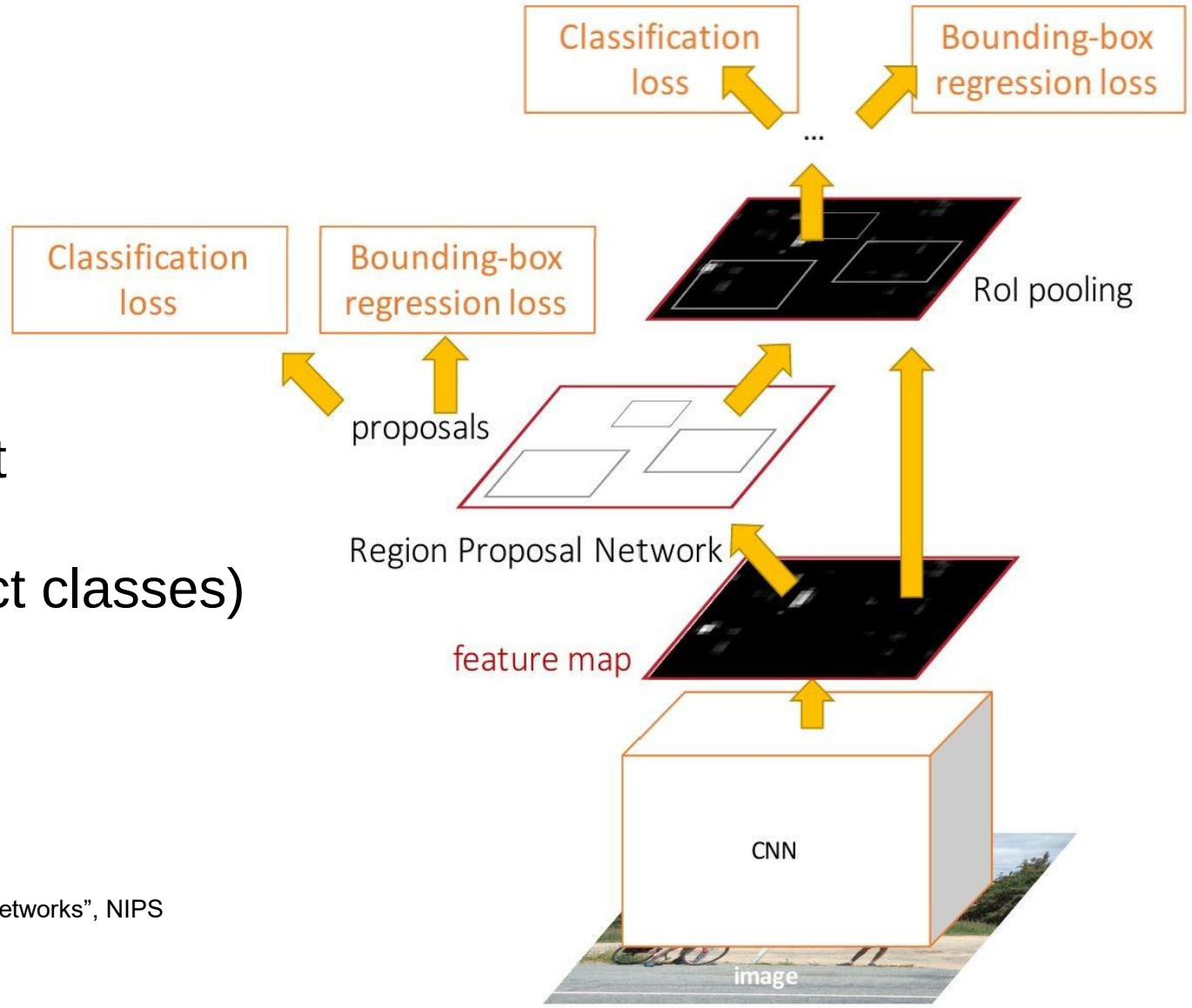
Box transforms
4K x 20 x 15

Sort the K*20*15 boxes by their “objectness” score, take top ~300 as our proposals

Faster R-CNN

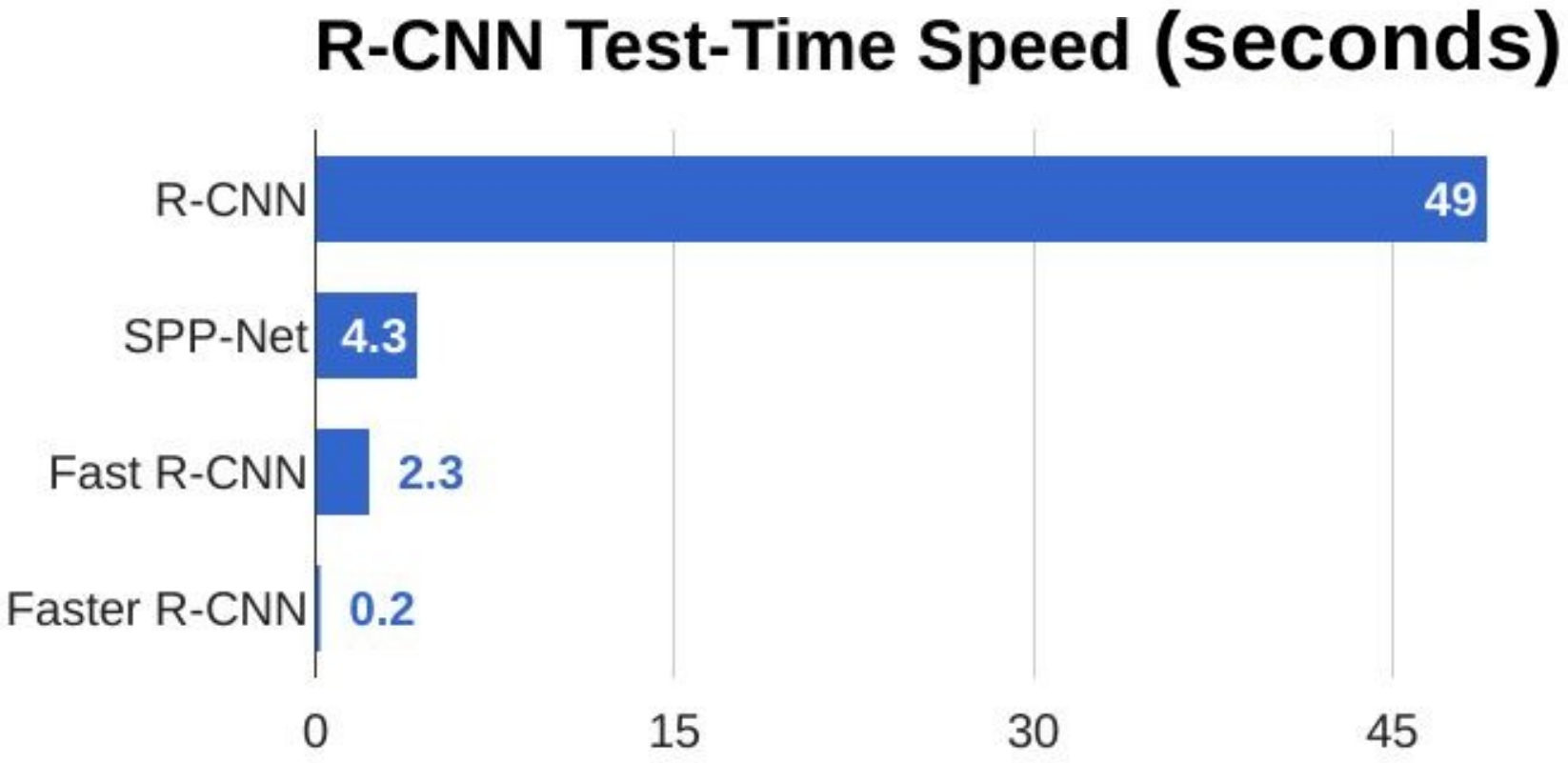
Jointly train with 4 losses:

1. RPN classify object / not object
2. RPN regress box coordinates
3. Final classification score (object classes)
4. Final box coordinates



Ren et al, “Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks”, NIPS 2015

Faster R-CNN

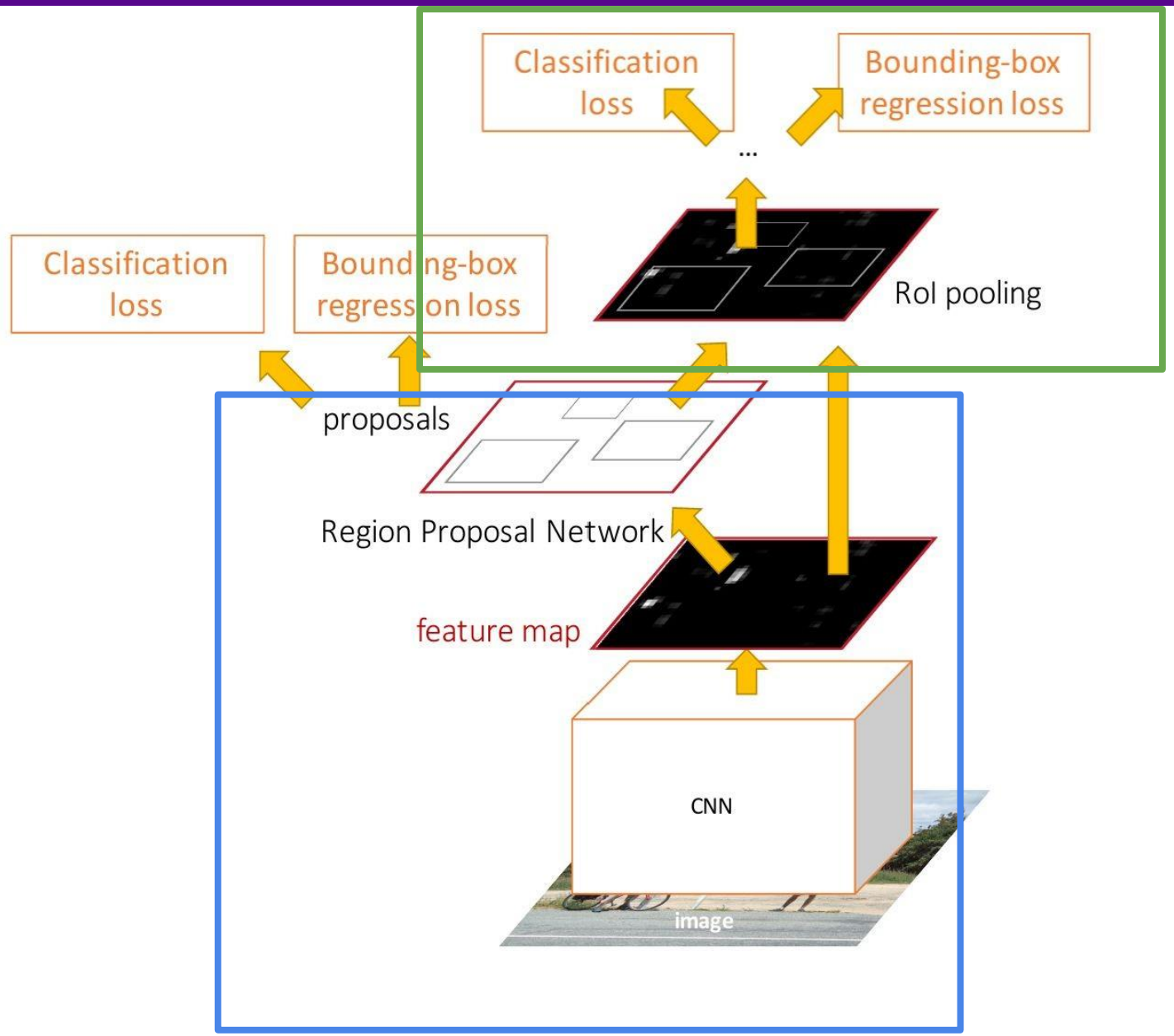

R-CNN Test-Time Speed (seconds)
R-CNN
49
SPP-Net
4.3
Fast R-CNN
2.3
Faster R-CNN
0.2
0
15
30
45

Faster R-CNN

Faster R-CNN is a **Two-stage object detector**

First stage: Run once per image
- Backbone network
- Region proposal network

Second stage: Run once per region
- Crop features: RoI pool / align
- Predict object class
- Prediction bbox offset

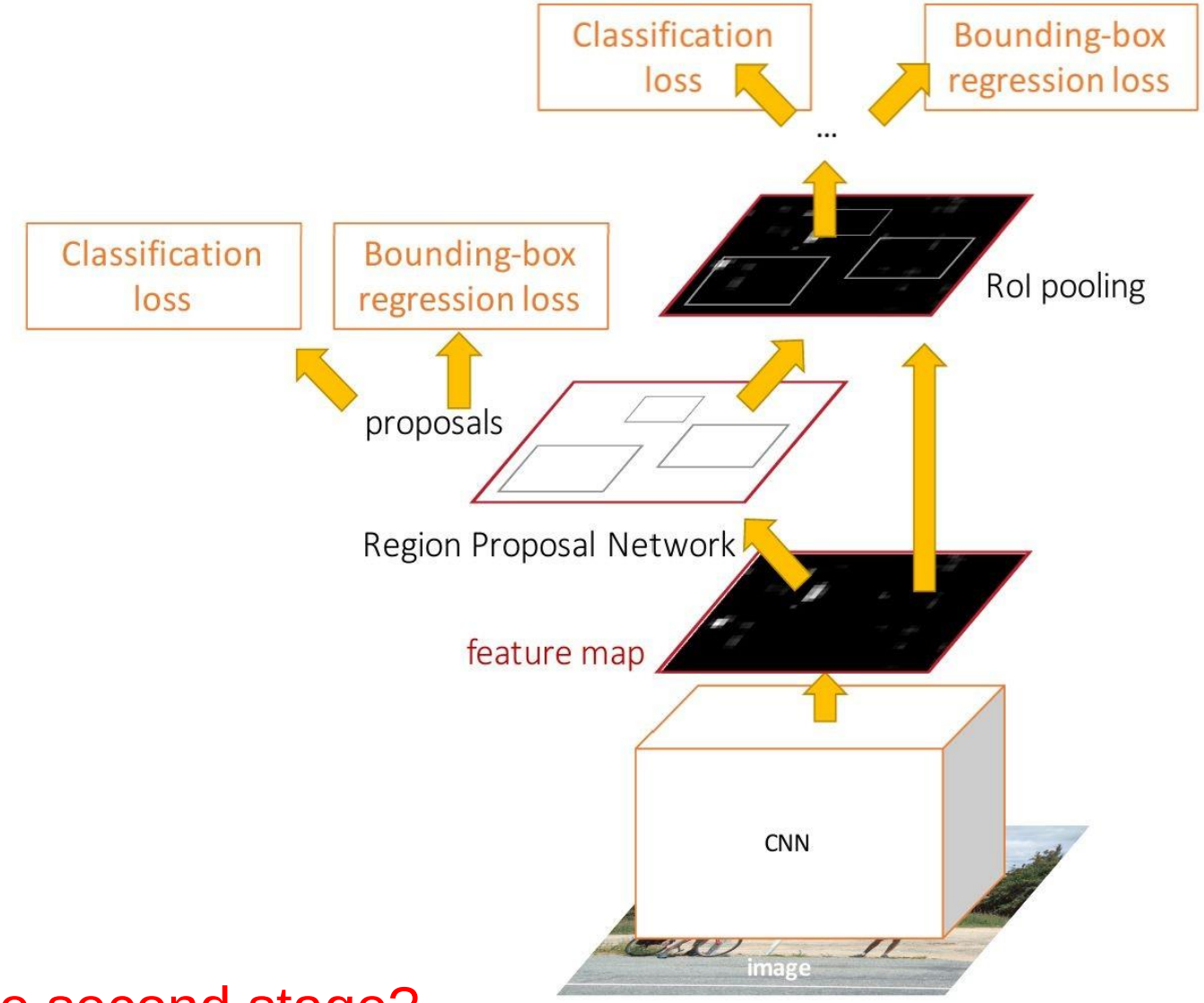


Faster R-CNN

Faster R-CNN is a **Two-stage object detector**

First stage: Run once per image
- Backbone network
- Region proposal network

Second stage: Run once per region
- Crop features: RoI pool / align
- Predict object class
- Prediction bbox offset

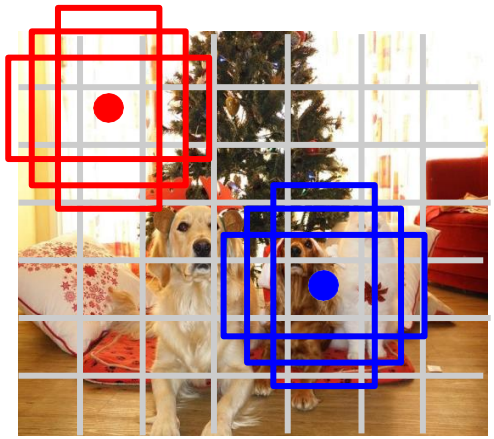


Do we really need the second stage?

Single Stage Object Detection

- A one-stage detector requires only a single pass through the neural network and predicts all the bounding boxes in one go.
- Unlike two-stage detectors, these methods are faster and don't have to deal with a large number of design choices.

Input image
3 x H x W



Divide image into grid 7 x 7

Imagine a set of **base boxes** centered at each grid cell
Here B = 3

Within each grid cell:

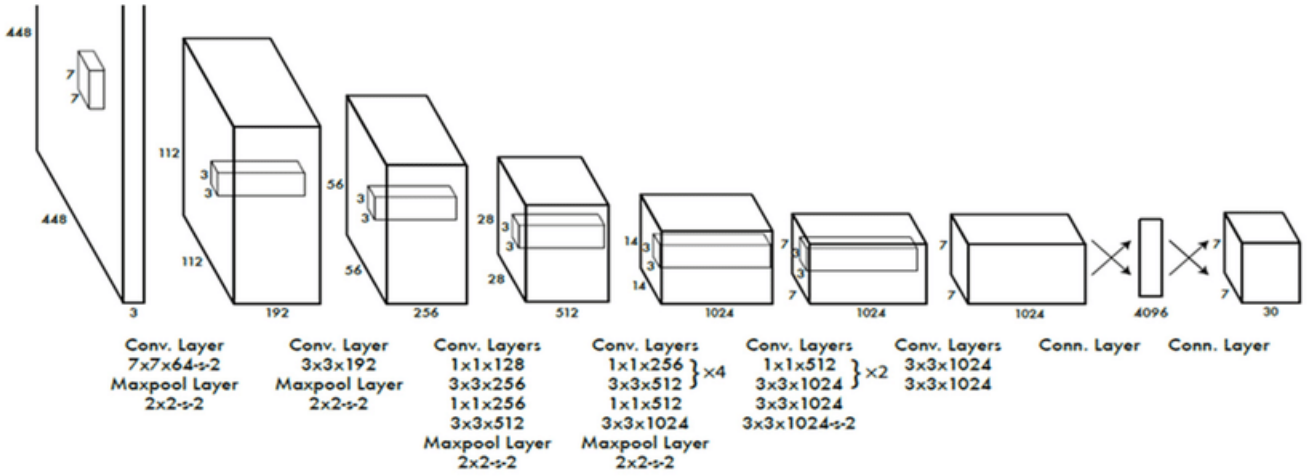
- Regress from each of the B base boxes to a final box with 5 numbers: (dx, dy, dh, dw, confidence)
- Predict scores for each of C classes (including background as a class)
- Looks a lot like RPN, but category-specific!

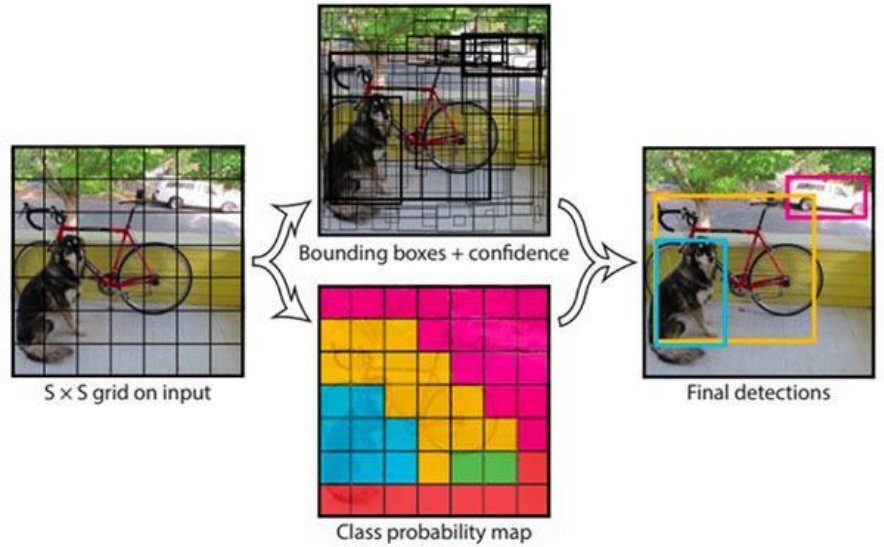
Output:
7 x 7 x (5 * B + C)

Redmon et al, "You Only Look Once: Unified, Real-Time Object Detection", CVPR 2016
Liu et al, "SSD: Single-Shot MultiBox Detector", ECCV 2016
Lin et al, "Focal Loss for Dense Object Detection", ICCV 2017

Example : YOLO (You Only Look Once)

- YOLO divides up the image into a grid of **13 by 13** cells: Each of these cells is responsible for predicting **5 bounding boxes**. A bounding box describes the rectangle that encloses an object.
- YOLO also outputs a confidence score that tells us how certain it is that the predicted bounding box actually encloses an object.





Summary – Deep Learning based Object Detection

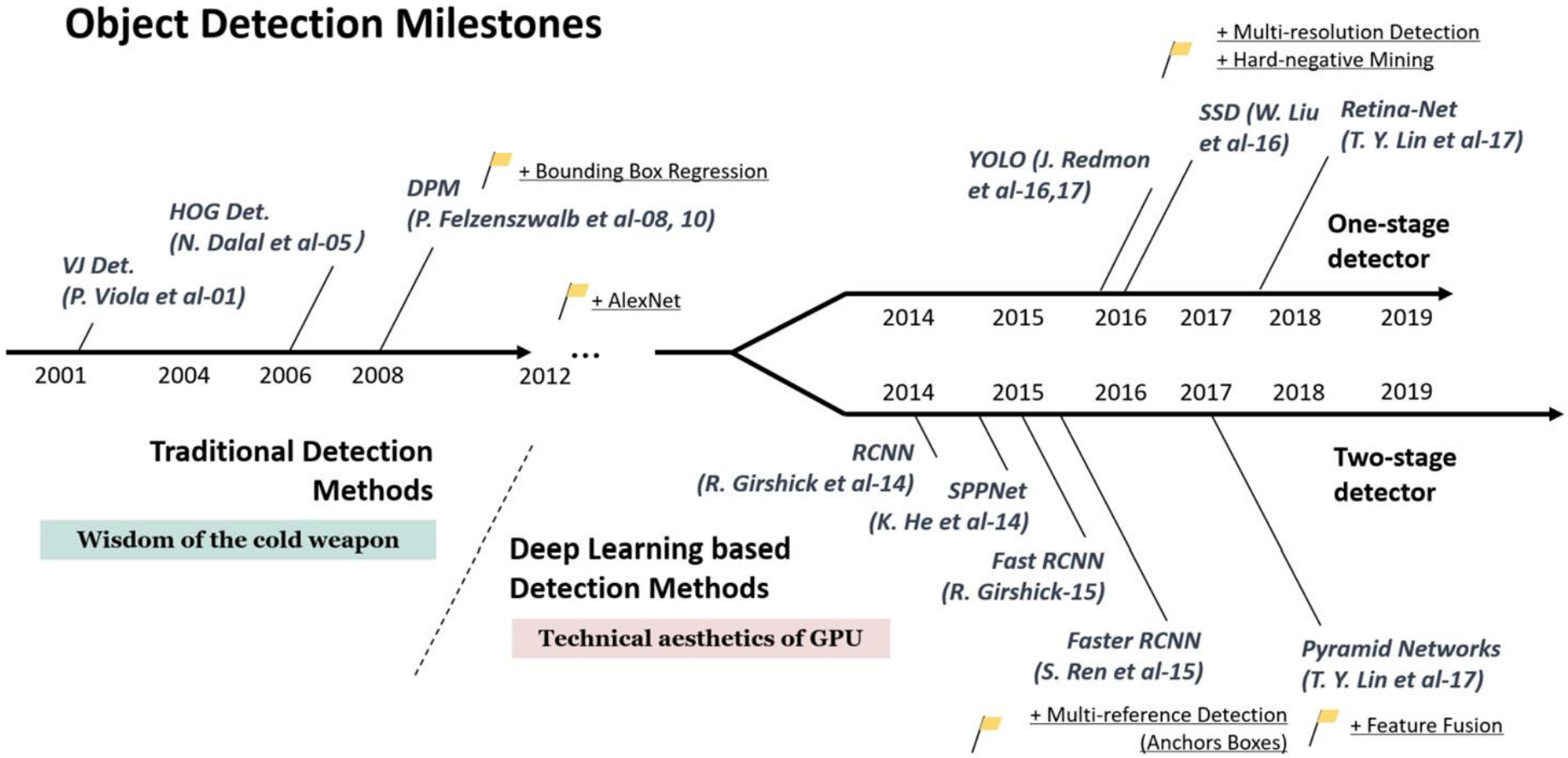


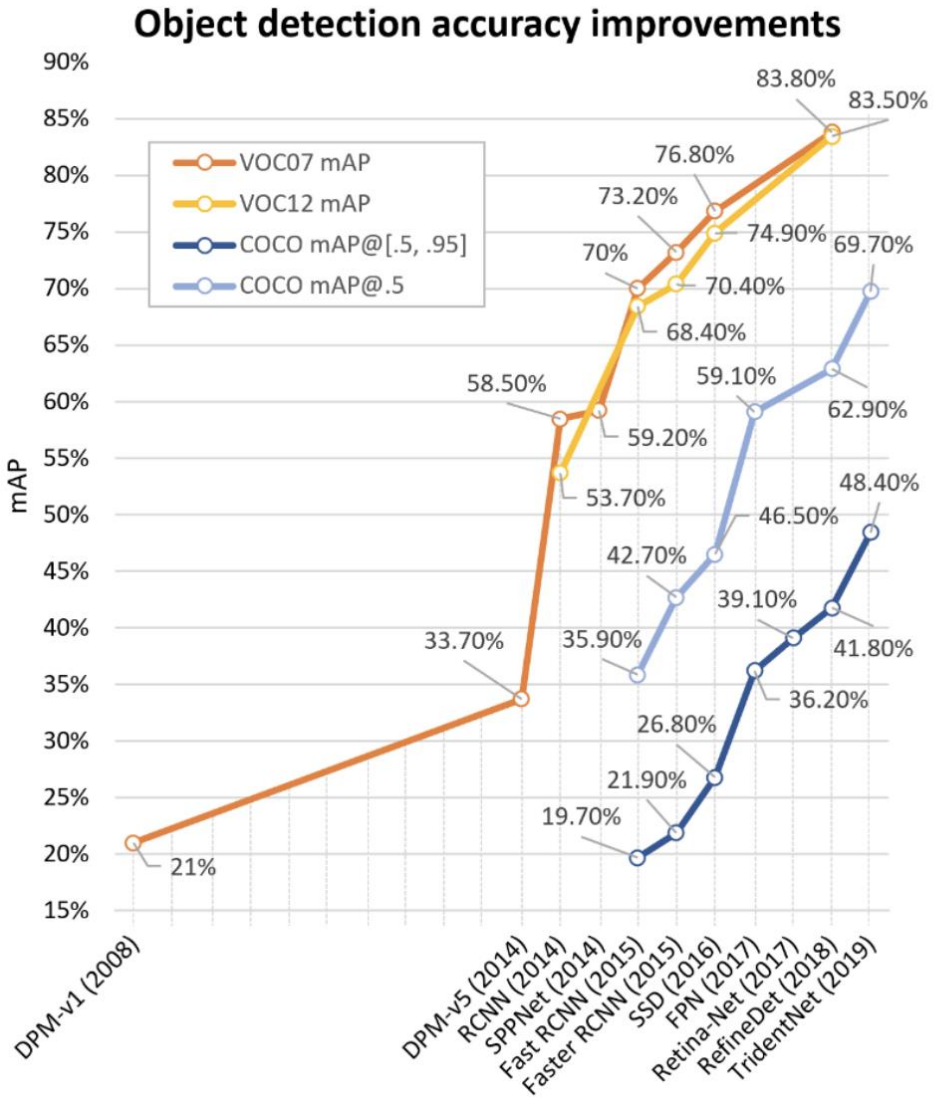
Fig. 2. A road map of object detection. Milestone detectors in this figure: VJ Det. [10, 11], HOG Det. [12], DPM [13–15], RCNN [16], SPPNet [17], Fast RCNN [18], Faster RCNN [19], YOLO [20], SSD [21], Pyramid Networks [22], Retina-Net [23].

https://arxiv.org/pdf/1905.05055.pdf

Summary – Deep Learning based Object Detection

- RetinaNet

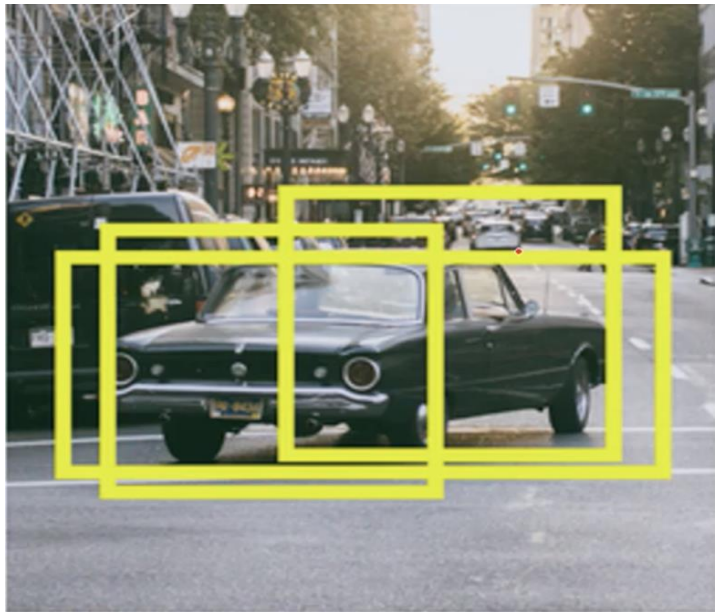
In despite of its high speed and simplicity, the one-stage detectors have trailed the accuracy of two-stage detectors for years. T.-Y. Lin *et al.* have discovered the reasons behind and proposed RetinaNet in 2017 [23]. They claimed that the extreme foreground-background class imbalance encountered during training of dense detectors is the central cause. To this end, a new loss function named "focal loss" has been introduced in RetinaNet by reshaping the standard cross entropy loss so that detector will put more focus on hard, misclassified examples during training. Focal Loss enables the one-stage detectors to achieve comparable accuracy of two-stage detectors while maintaining very high detection speed. (COCO mAP@.5=59.1%, mAP@[.5, .95]=39.1%).



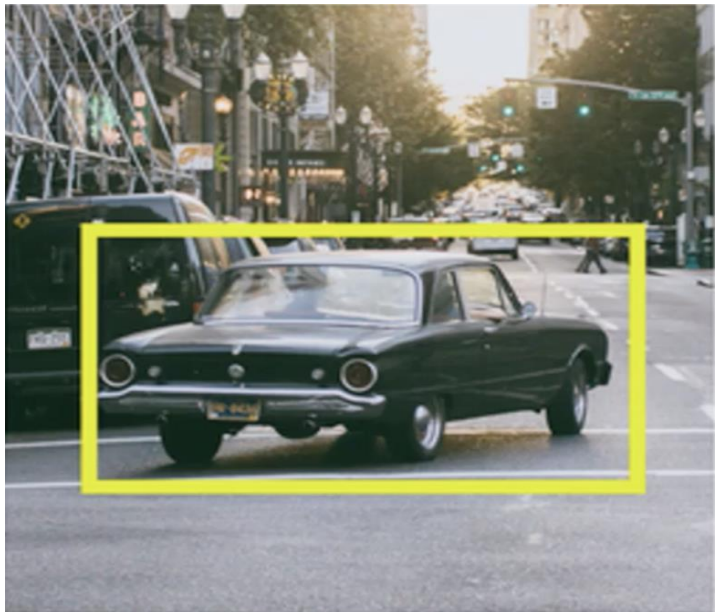
https://arxiv.org/pdf/1905.05055.pdf

Overlapping Boxes

Problem: Object detectors often output many overlapping detections:

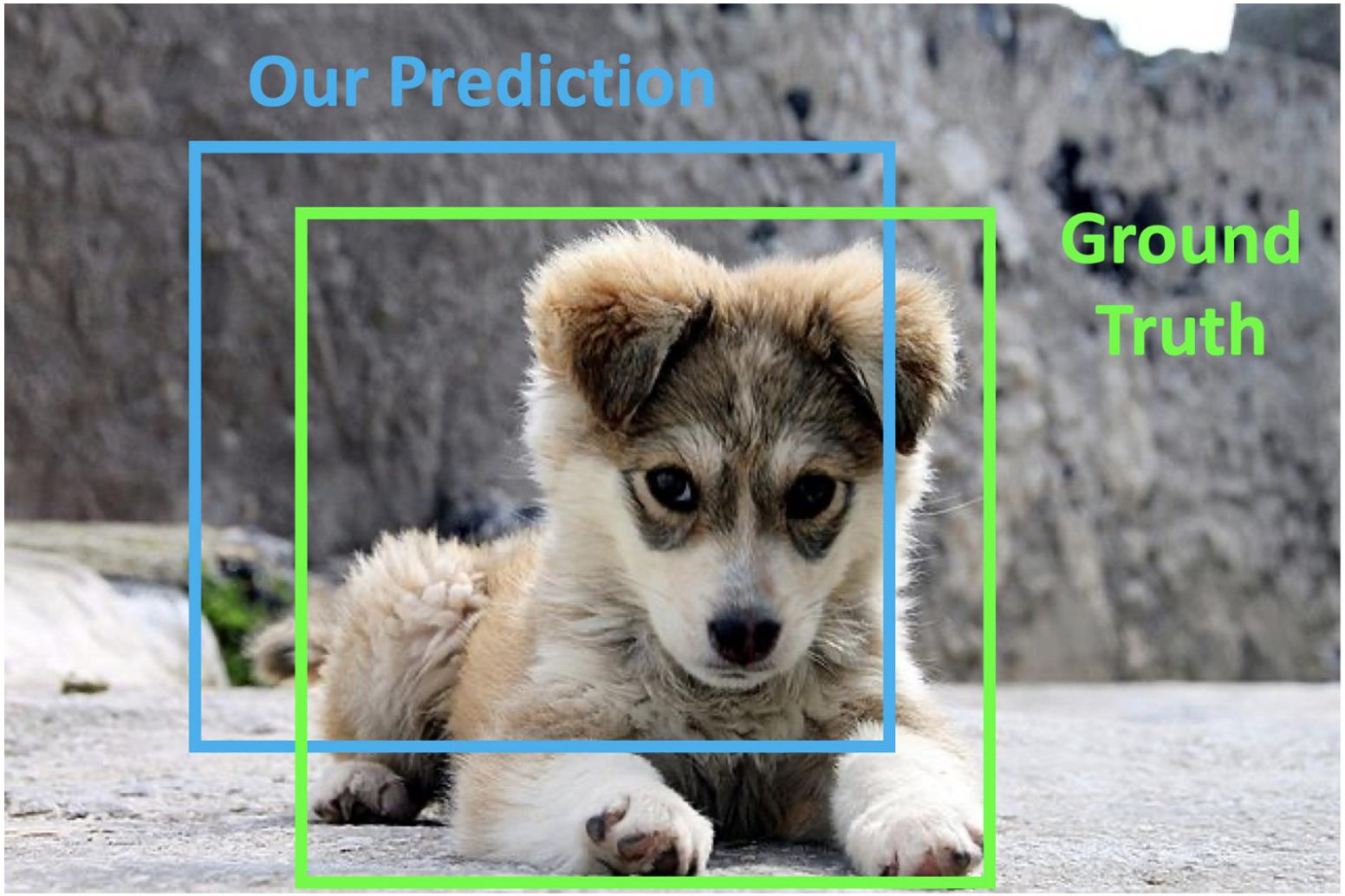


Suppression



Comparing Boxes: Intersection over Union (IoU)

How can we compare our prediction to the ground-truth box?

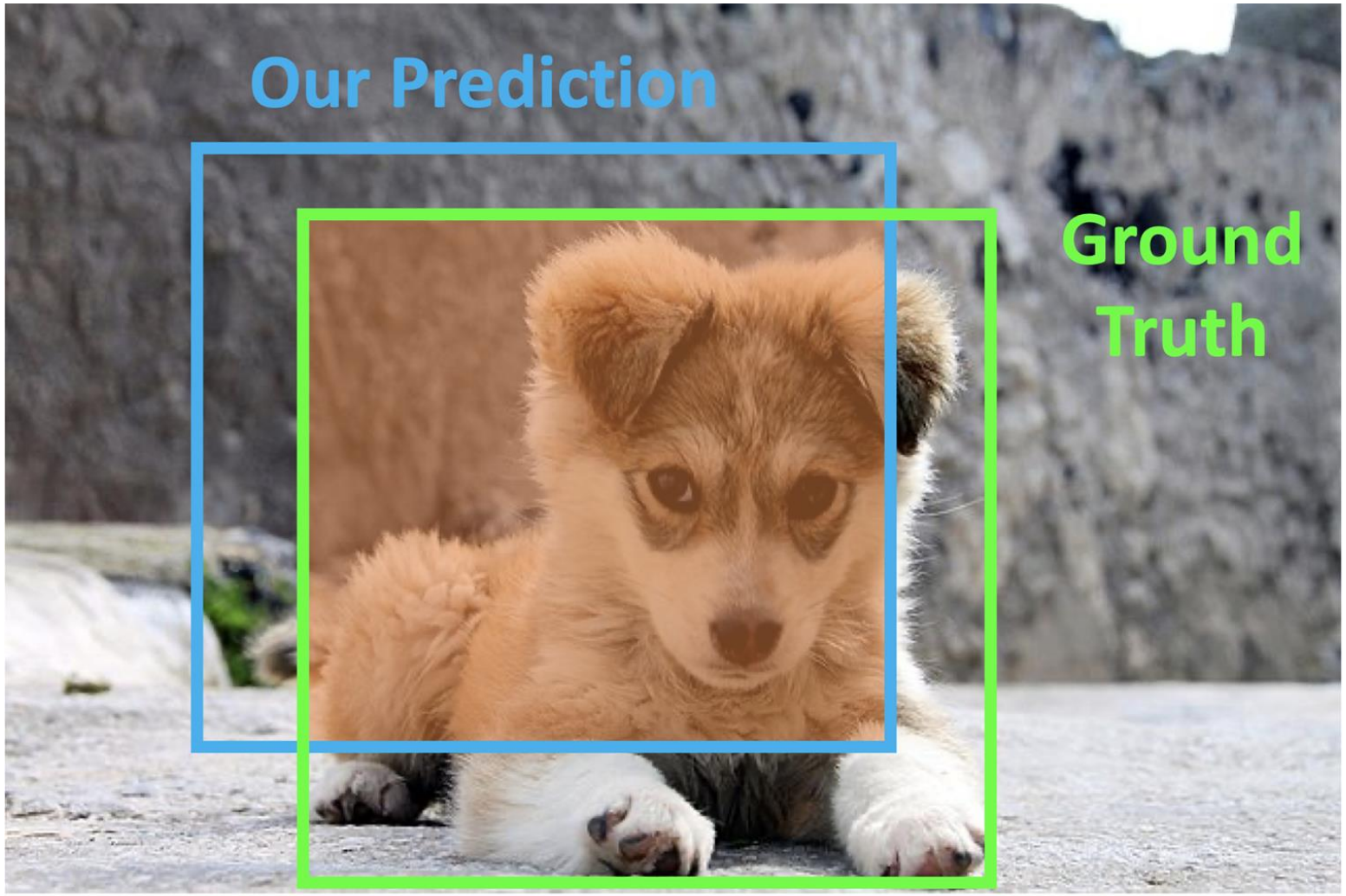


Comparing Boxes: Intersection over Union (IoU)

How can we compare our prediction to the ground-truth box?

Intersection over Union (IoU) (Also called "Jaccard similarity" or "Jaccard index"):

$$\frac{\textit{Area of Intersection}}{\textit{Area of Union}}$$



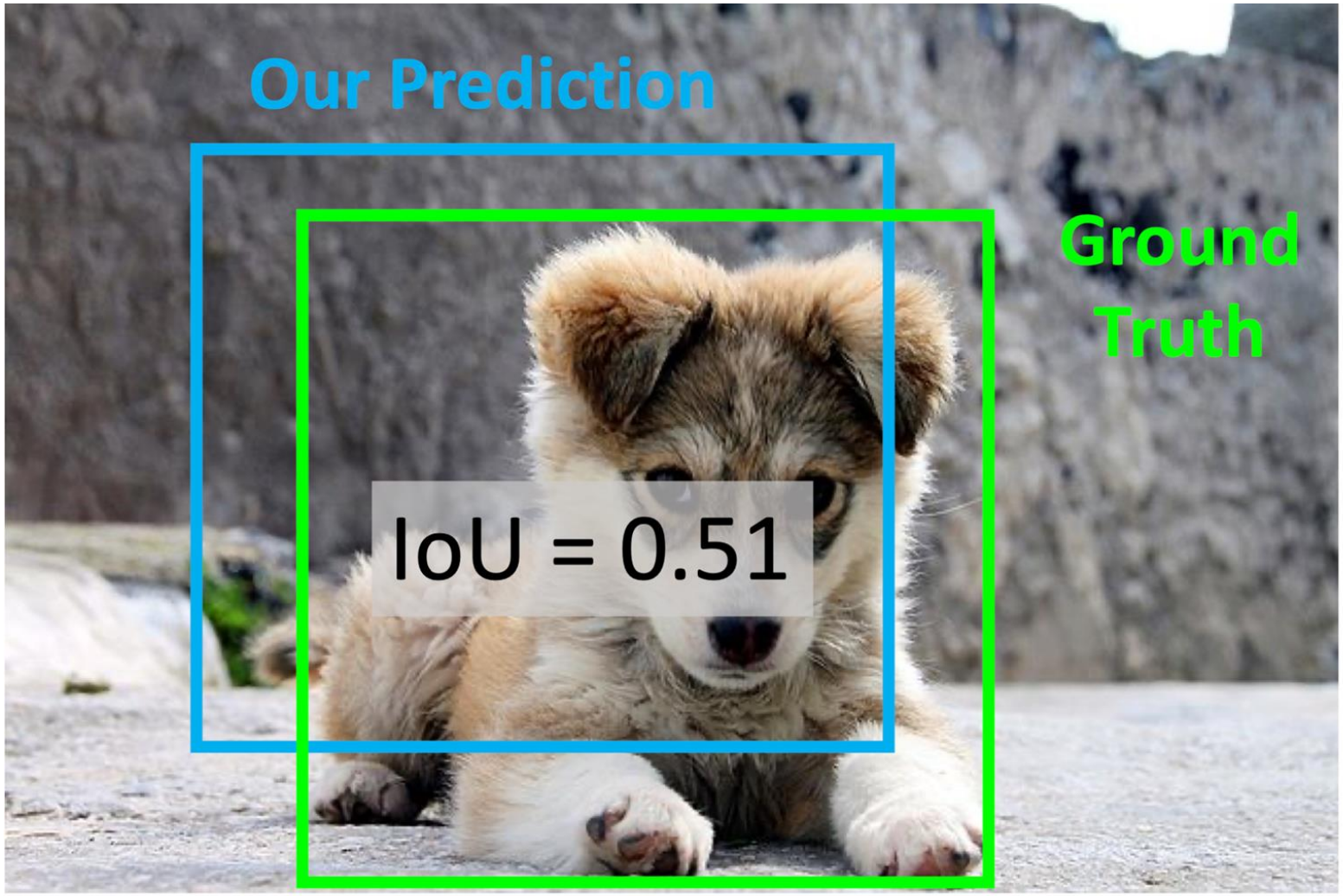
Comparing Boxes: Intersection over Union (IoU)

How can we compare our prediction to the ground-truth box?

Intersection over Union (IoU) (Also called “Jaccard similarity” or “Jaccard index”):

$$\frac{\textit{Area of Intersection}}{\textit{Area of Union}}$$

IoU > 0.5 is “decent”



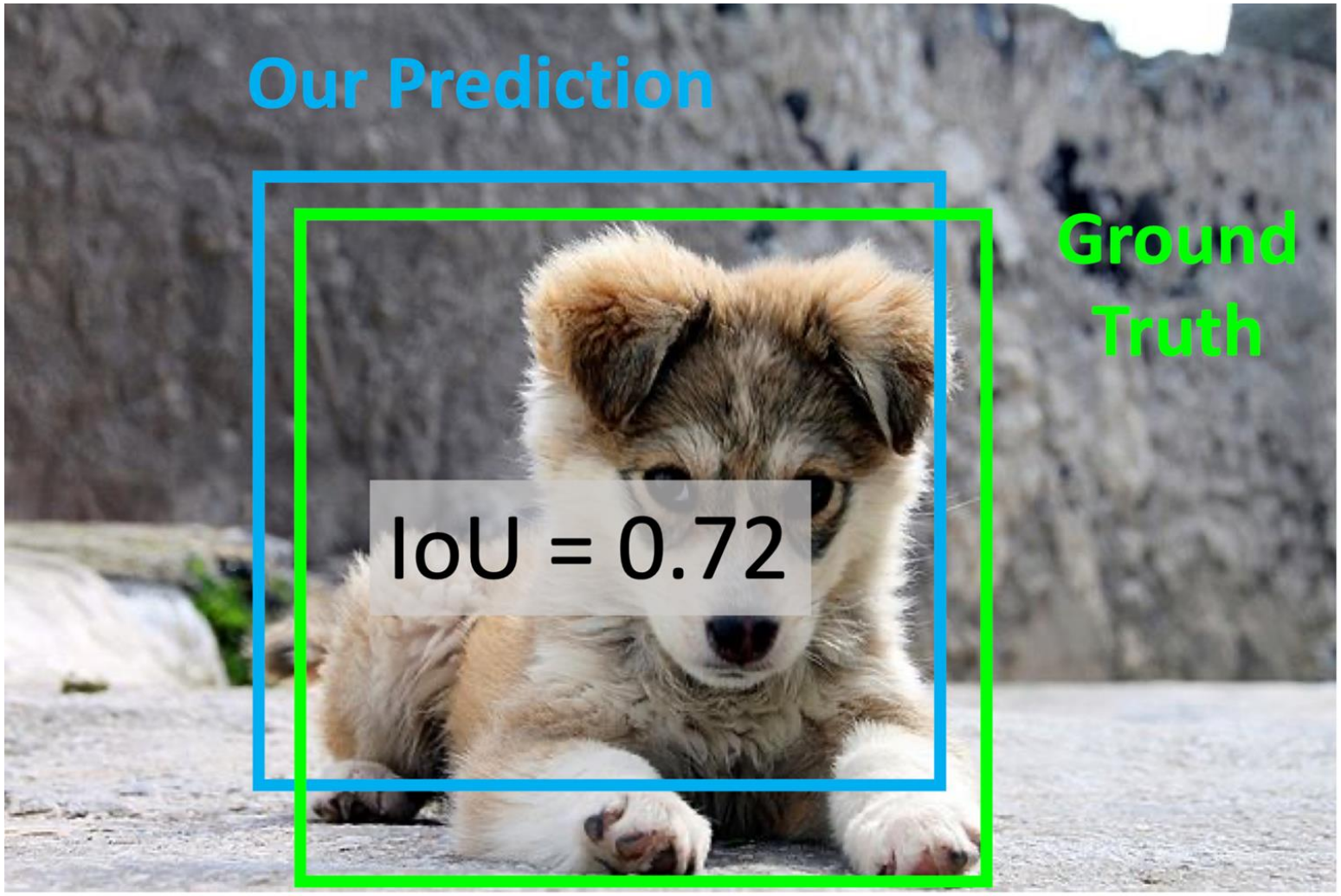
Comparing Boxes: Intersection over Union (IoU)

How can we compare our prediction to the ground-truth box?

Intersection over Union (IoU) (Also called "Jaccard similarity" or "Jaccard index"):

$$\frac{\textit{Area of Intersection}}{\textit{Area of Union}}$$

IoU > 0.7 is "pretty good"



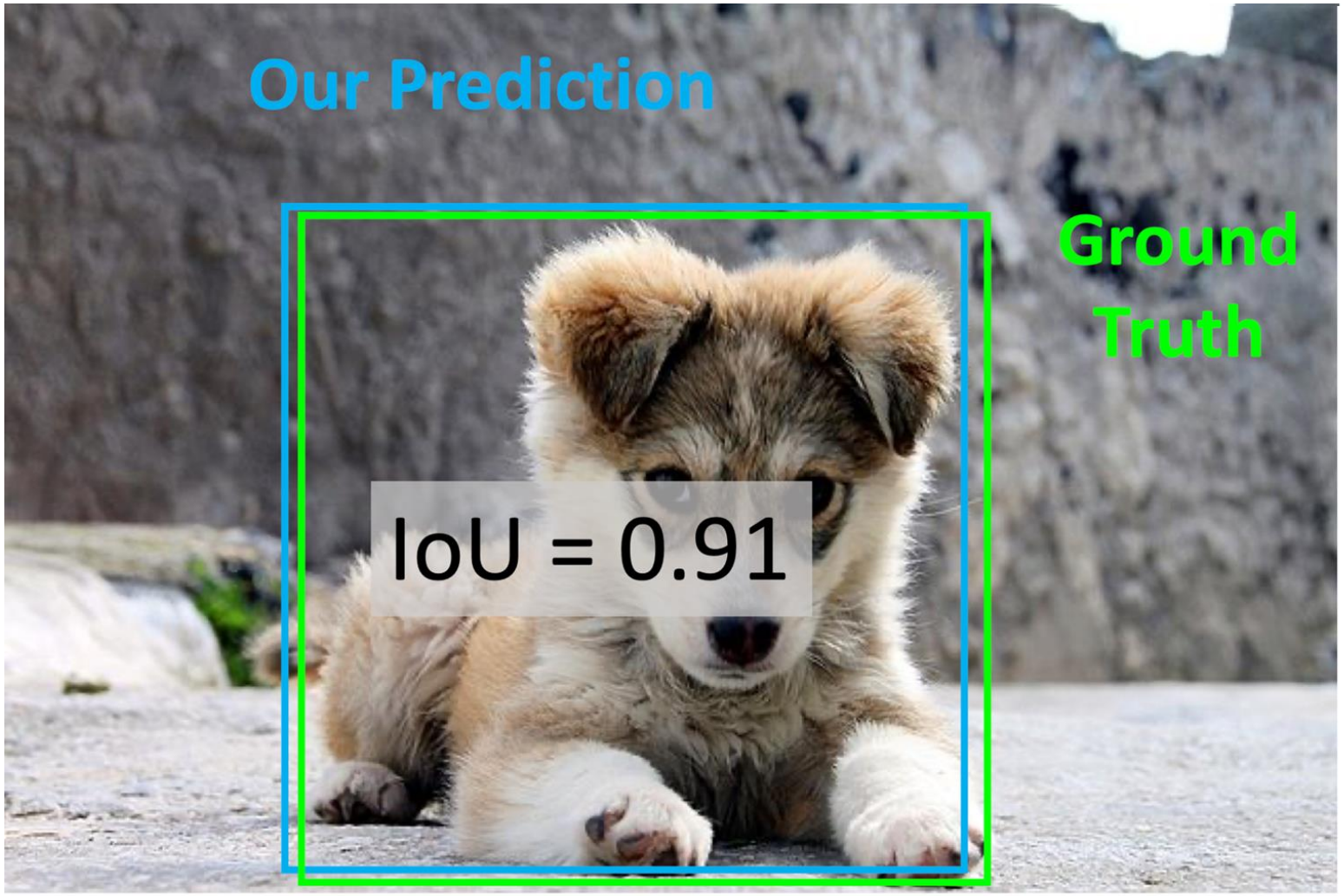
Comparing Boxes: Intersection over Union (IoU)

How can we compare our prediction to the ground-truth box?

Intersection over Union (IoU) (Also called "Jaccard similarity" or "Jaccard index"):

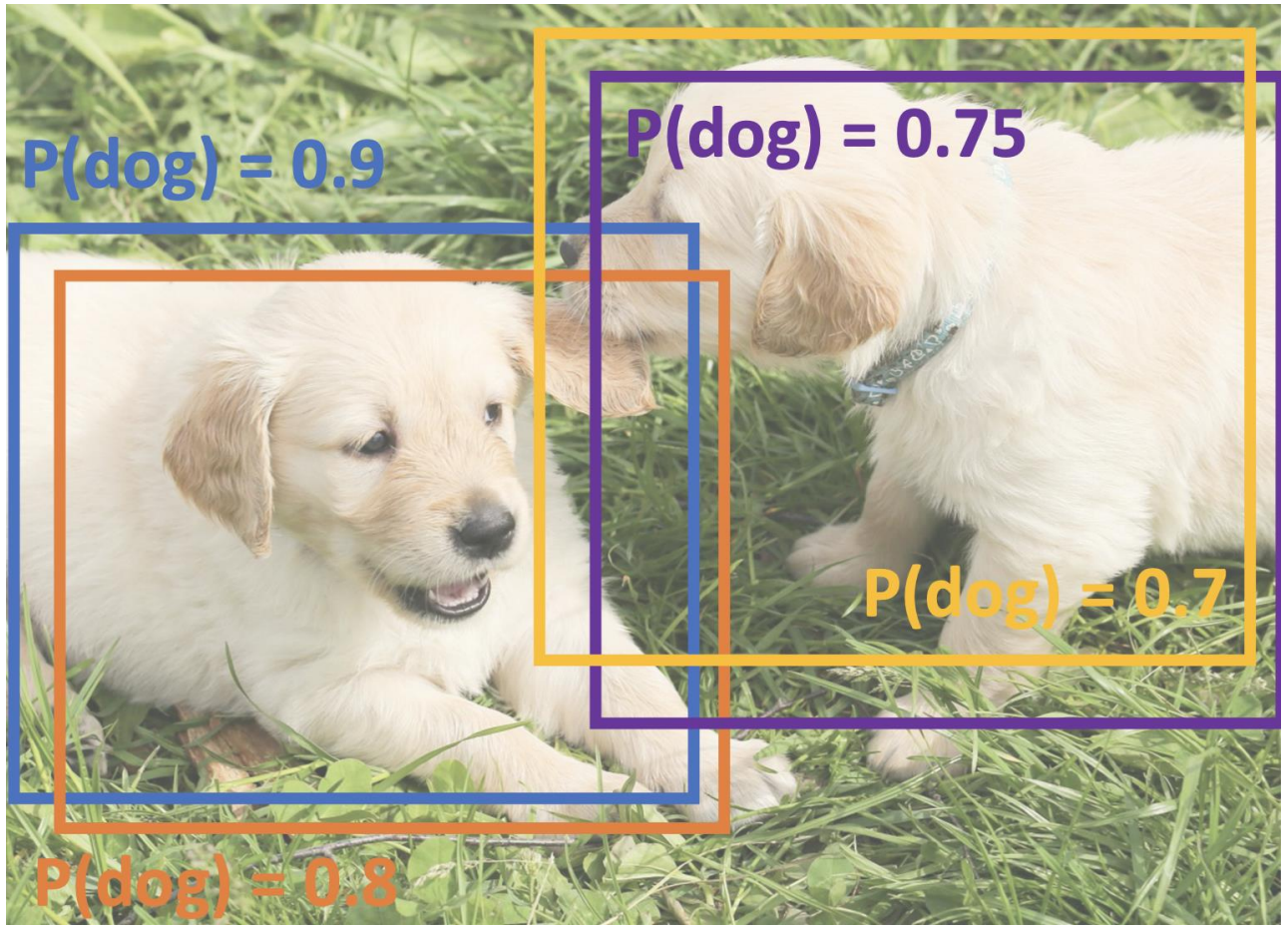
$$\frac{\textit{Area of Intersection}}{\textit{Area of Union}}$$

IoU > 0.9 is "almost perfect"



Overlapping Boxes

Problem: Object detectors often output many overlapping detections:

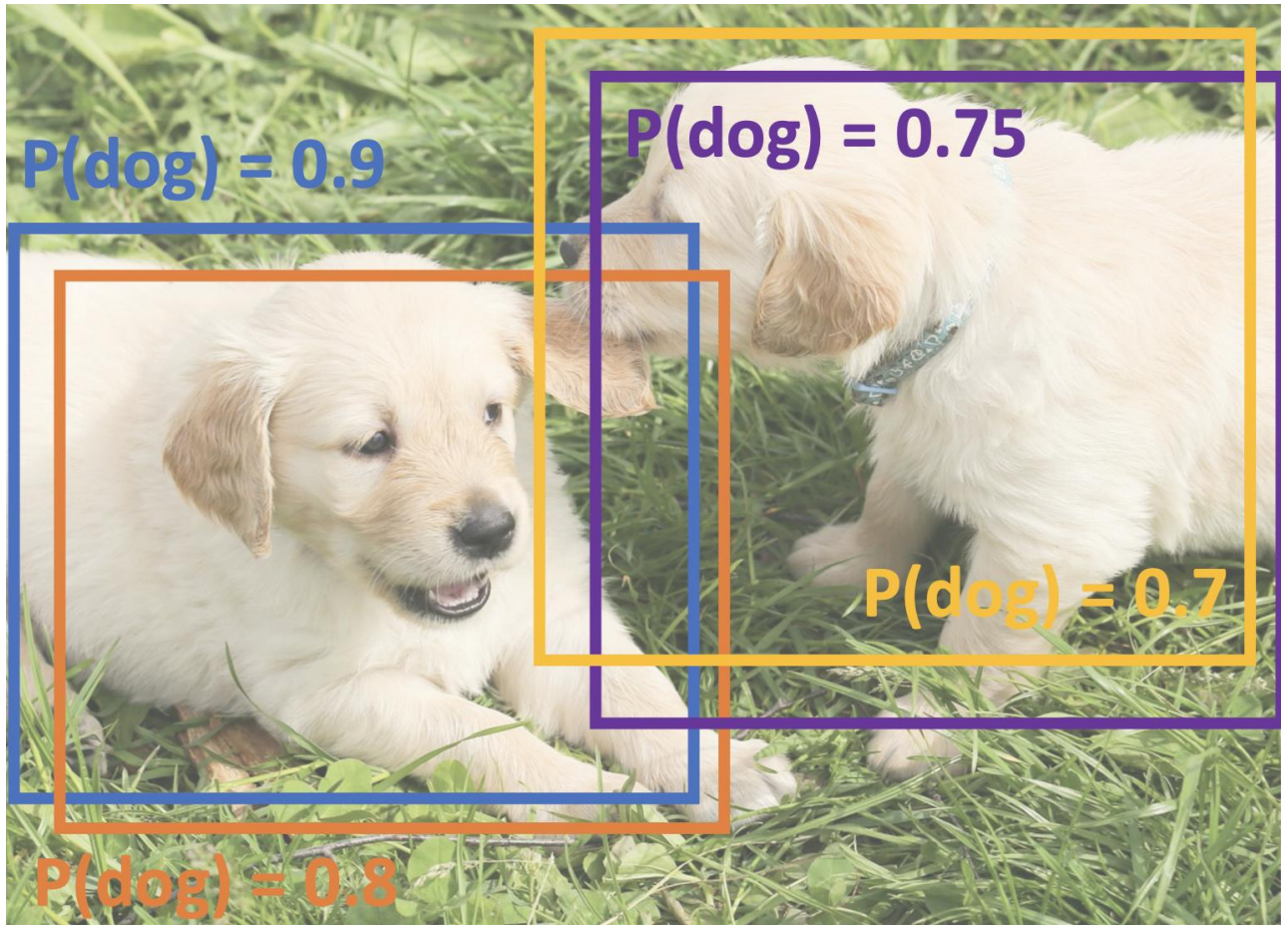


Overlapping Boxes

Problem: Object detectors often output many overlapping detections:

Solution: Post-process raw detections using **Non-Max Suppression (NMS)**

1. Select next highest-scoring box
2. Eliminate lower-scoring boxes with IoU < threshold (e.g. 0.7)
3. If any boxes remain, GOTO 1



Overlapping Boxes

Problem: Object detectors often output many overlapping detections:

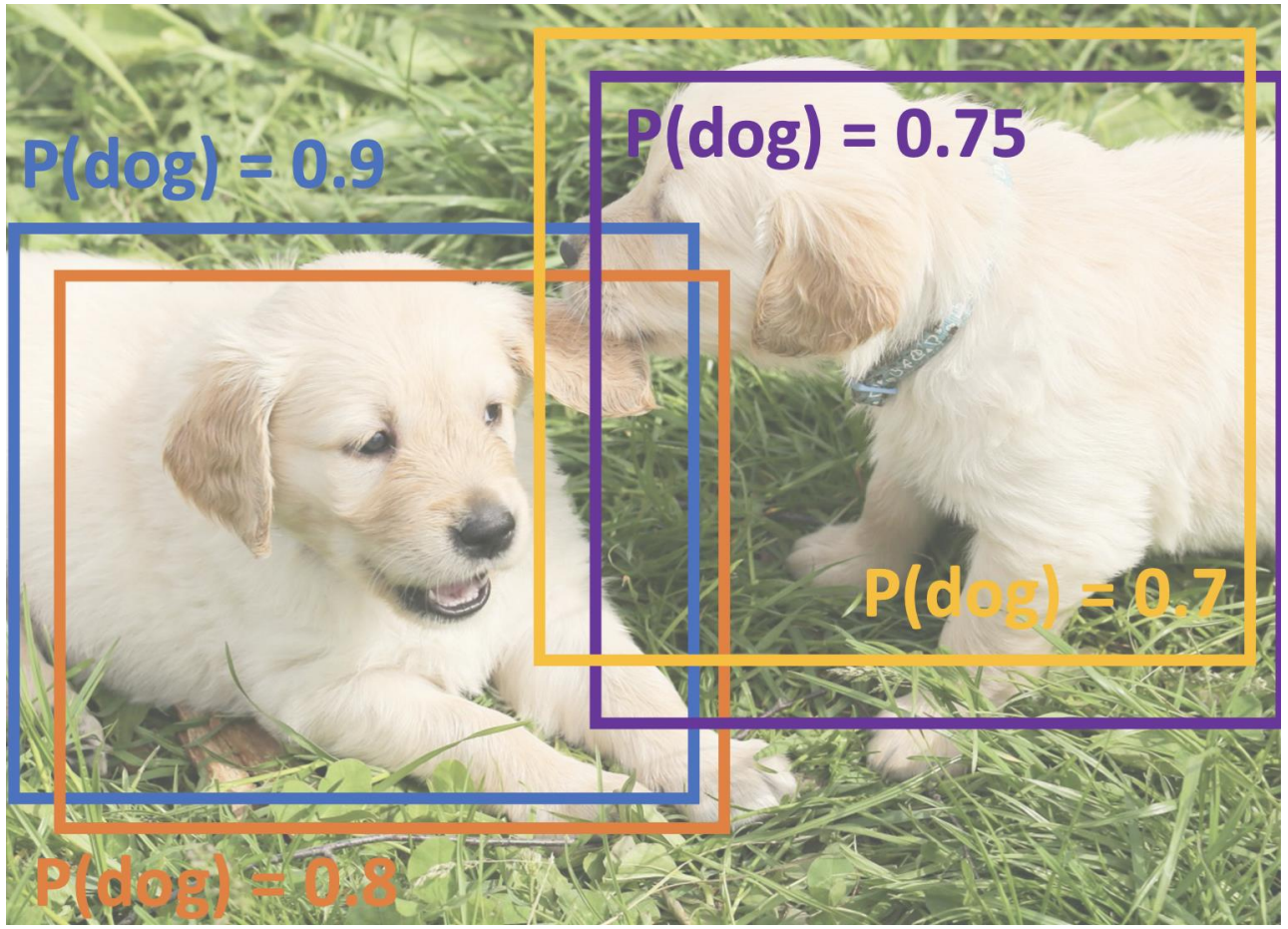
Solution: Post-process raw detections using **Non-Max Suppression (NMS)**

1. Select next highest-scoring box
2. Eliminate lower-scoring boxes with IoU < threshold (e.g. 0.7)
3. If any boxes remain, GOTO 1

IoU(■, ■) = **0.78**

IoU(■, ■) = 0.05

IoU(■, ■) = 0.07



Overlapping Boxes

Problem: Object detectors often output many overlapping detections:

Solution: Post-process raw detections using **Non-Max Suppression (NMS)**

1. Select next highest-scoring box
2. Eliminate lower-scoring boxes with IoU < threshold (e.g. 0.7)
3. If any boxes remain, GOTO 1

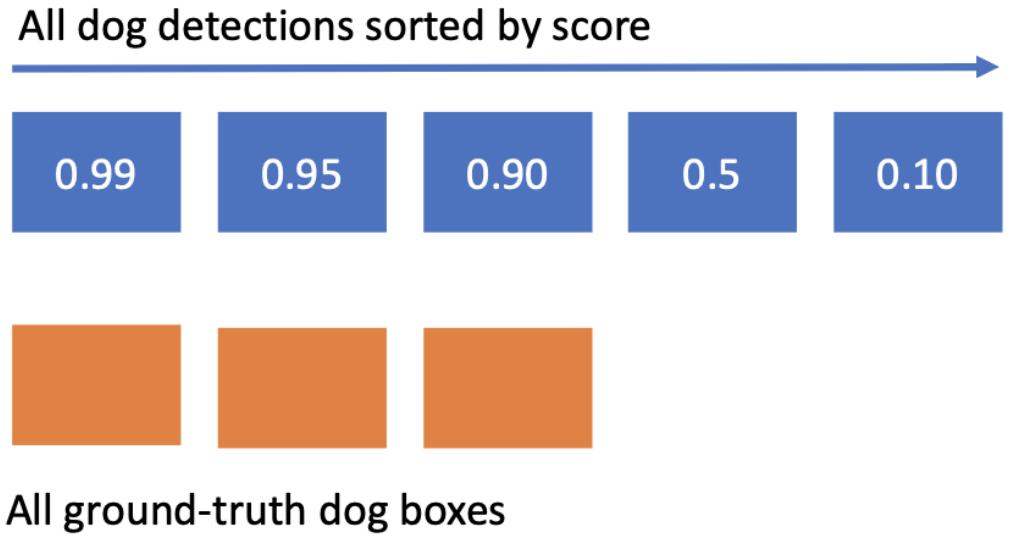
Problem: NMS may eliminate ”good” boxes when objects are highly overlapping... no good solution =(

Evaluating Object Detectors: Mean Average Precision (mAP)

1. Run object detector on all test images (with NMS)
2. For each category, compute Average Precision (AP) = area under Precision vs Recall Curve

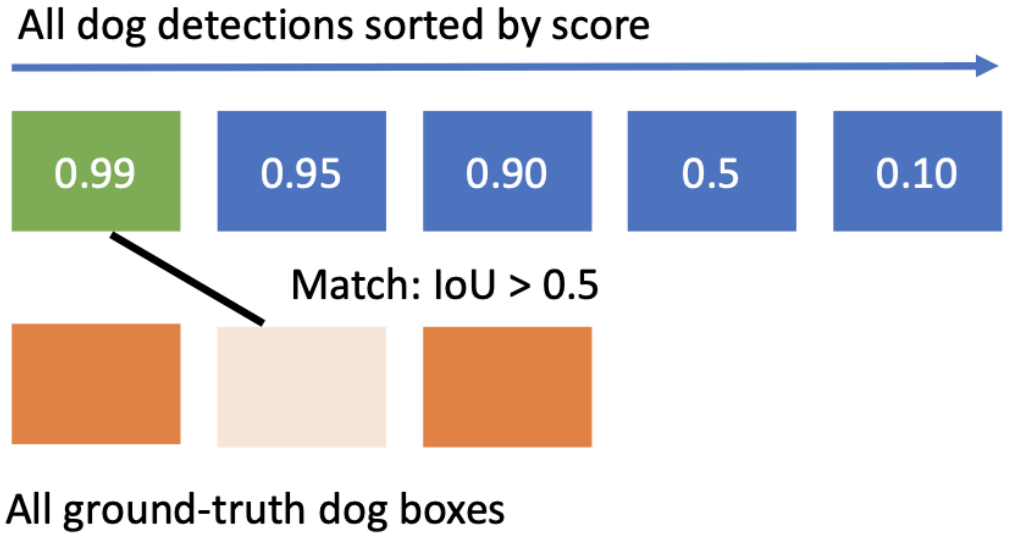
Evaluating Object Detectors: Mean Average Precision (mAP)

1. Run object detector on all test images (with NMS)
2. For each category, compute Average Precision (AP) = area under Precision vs Recall Curve



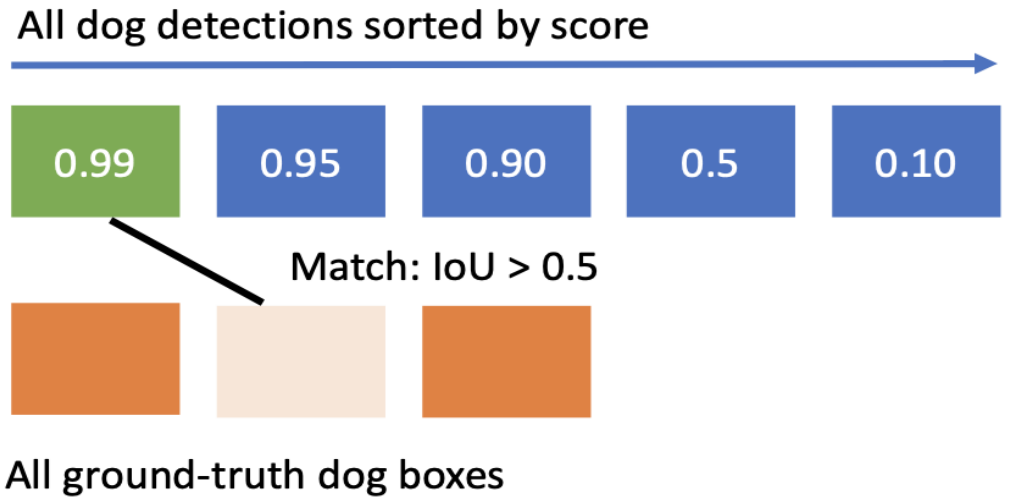
Evaluating Object Detectors: Mean Average Precision (mAP)

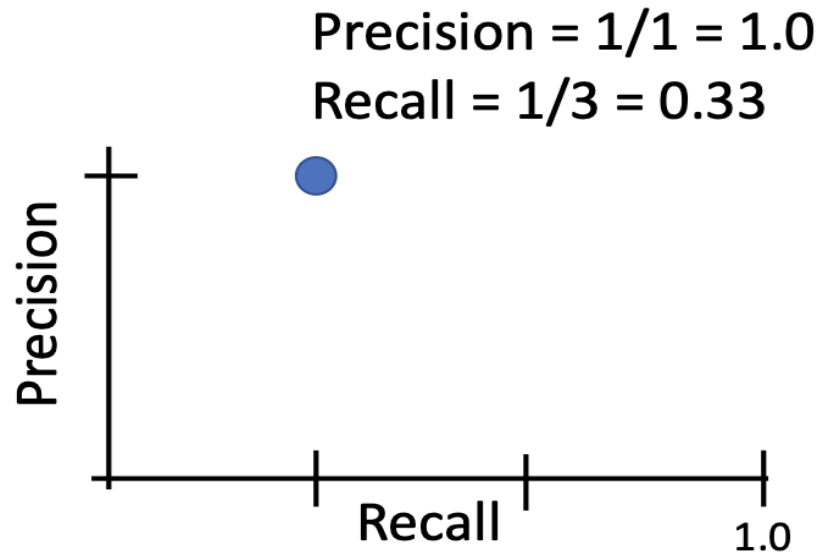
1. Run object detector on all test images (with NMS)
2. For each category, compute Average Precision (AP) = area under Precision vs Recall Curve
    a. For each detection (highest score to lowest score)
        i. If it matches some GT box with IoU > 0.5, mark it as positive and eliminate the GT
        ii. Otherwise mark it as negative



Evaluating Object Detectors: Mean Average Precision (mAP)

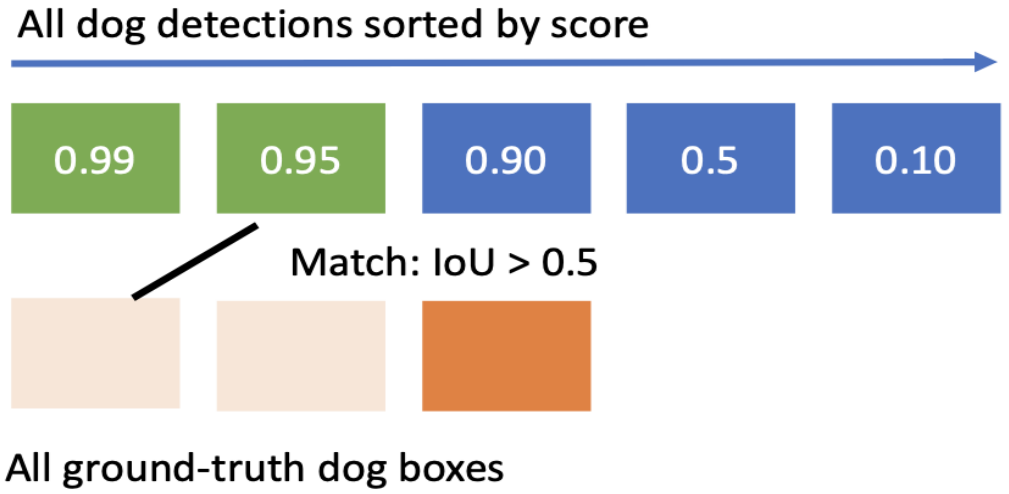
1. Run object detector on all test images (with NMS)
2. For each category, compute Average Precision (AP) = area under Precision vs Recall Curve
   a. For each detection (highest score to lowest score)
      i. If it matches some GT box with IoU > 0.5, mark it as positive and eliminate the GT
      ii. Otherwise mark it as negative
      iii. Plot a point on PR Curve

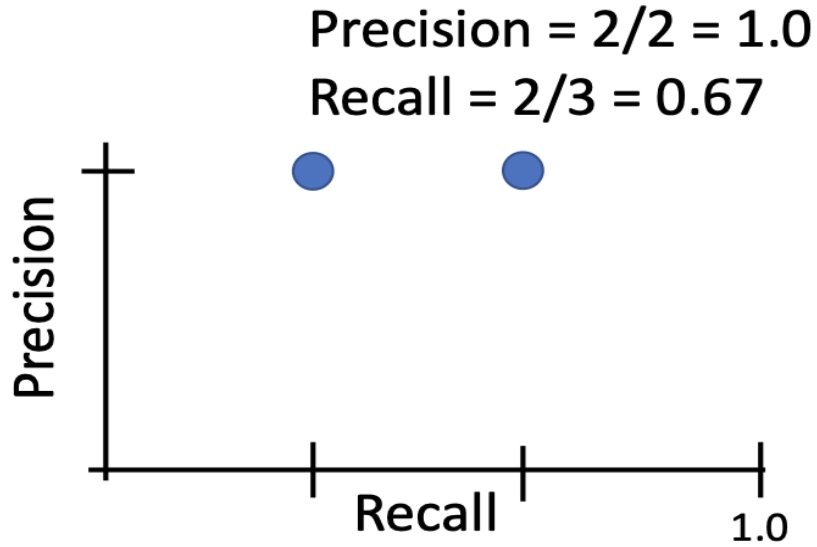




Evaluating Object Detectors: Mean Average Precision (mAP)

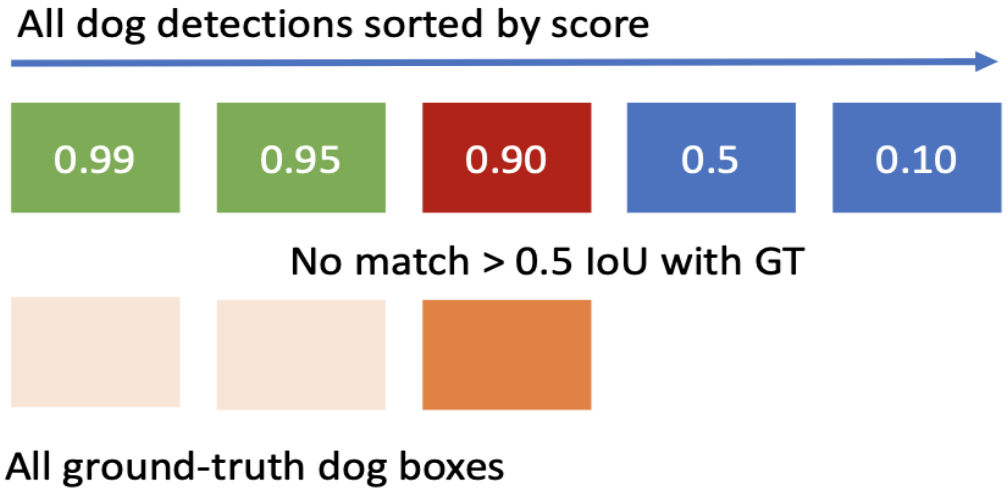
1. Run object detector on all test images (with NMS)
2. For each category, compute Average Precision (AP) = area under Precision vs Recall Curve
   a. For each detection (highest score to lowest score)
      i. If it matches some GT box with IoU > 0.5, mark it as positive and eliminate the GT
      ii. Otherwise mark it as negative
      iii. Plot a point on PR Curve

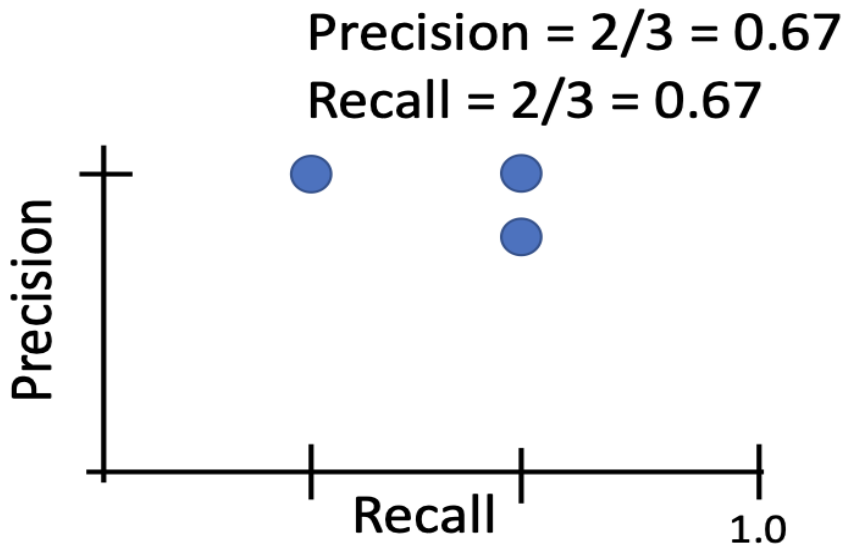




Evaluating Object Detectors: Mean Average Precision (mAP)

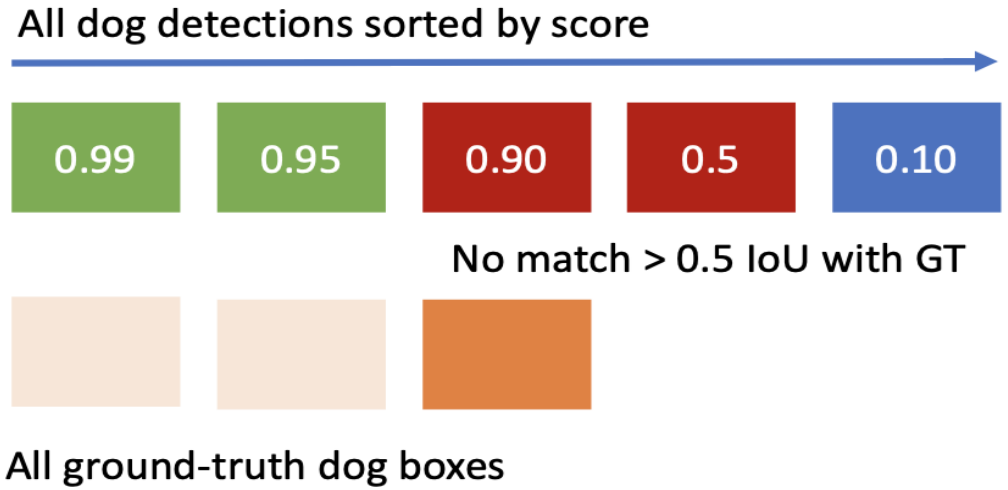
1. Run object detector on all test images (with NMS)
2. For each category, compute Average Precision (AP) = area under Precision vs Recall Curve
   a. For each detection (highest score to lowest score)
      i. If it matches some GT box with IoU > 0.5, mark it as positive and eliminate the GT
      ii. Otherwise mark it as negative
      iii. Plot a point on PR Curve

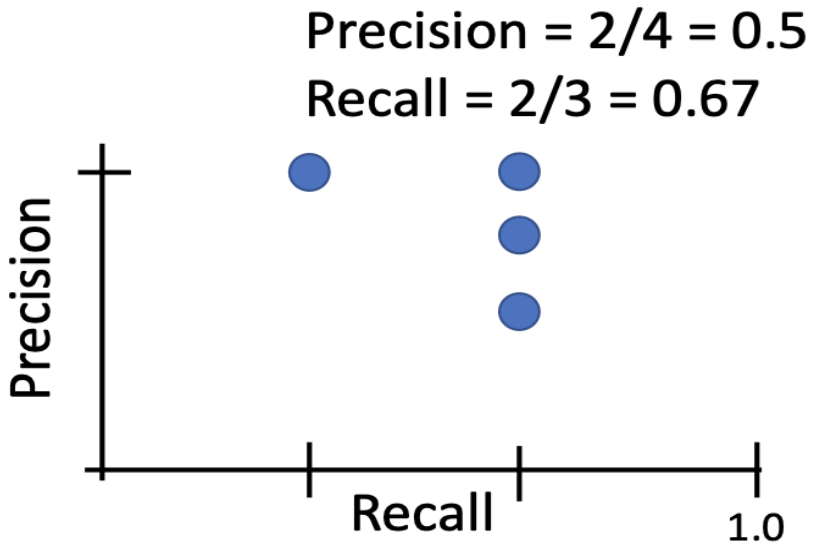




Evaluating Object Detectors: Mean Average Precision (mAP)

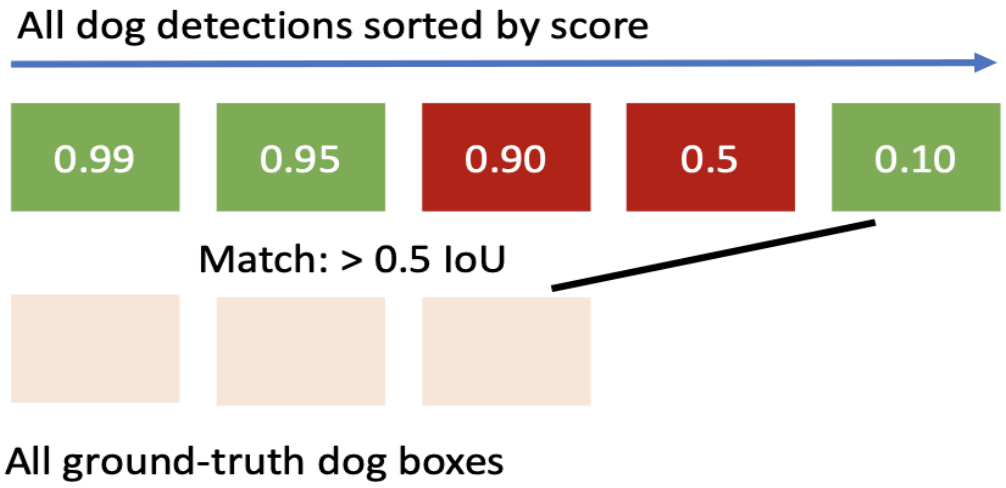
1. Run object detector on all test images (with NMS)
2. For each category, compute Average Precision (AP) = area under Precision vs Recall Curve
   a. For each detection (highest score to lowest score)
      i. If it matches some GT box with IoU > 0.5, mark it as positive and eliminate the GT
      ii. Otherwise mark it as negative
      iii. Plot a point on PR Curve

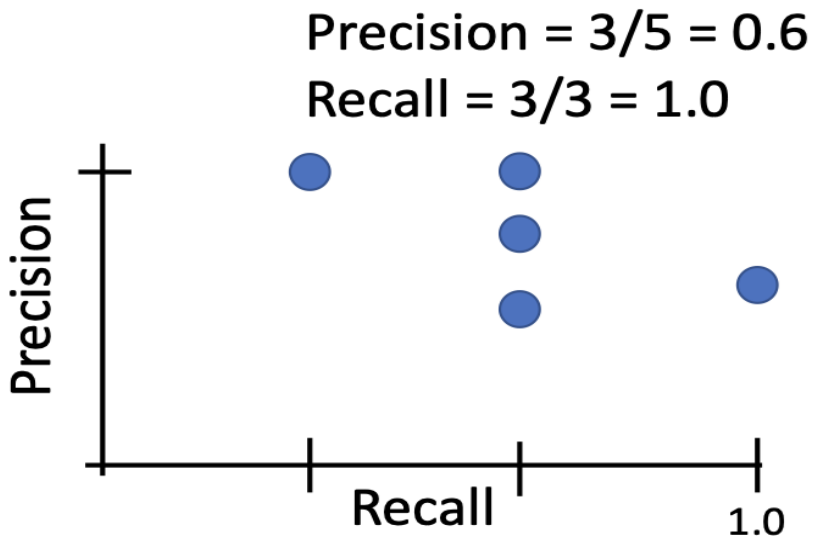




Evaluating Object Detectors: Mean Average Precision (mAP)

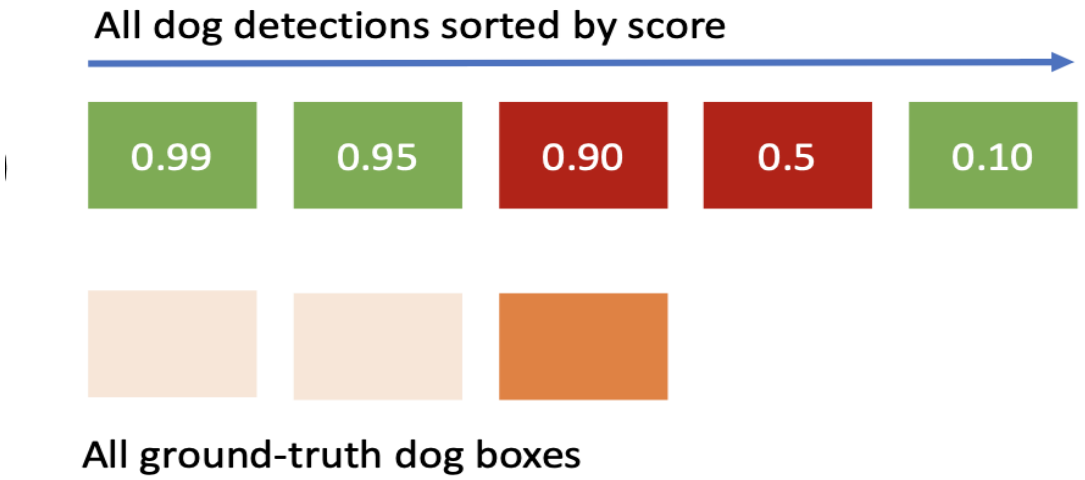
1. Run object detector on all test images (with NMS)
2. For each category, compute Average Precision (AP) = area under Precision vs Recall Curve
   a. For each detection (highest score to lowest score)
      i. If it matches some GT box with IoU > 0.5, mark it as positive and eliminate the GT
      ii. Otherwise mark it as negative
      iii. Plot a point on PR Curve

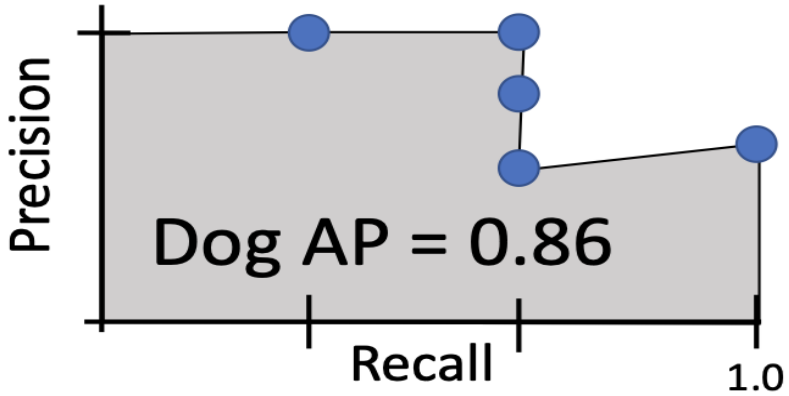




Evaluating Object Detectors: Mean Average Precision (mAP)

1. Run object detector on all test images (with NMS)
2. For each category, compute Average Precision (AP) = area under Precision vs Recall Curve
   a. For each detection (highest score to lowest score)
      i. If it matches some GT box with IoU > 0.5, mark it as positive and eliminate the GT
      ii. Otherwise mark it as negative
      iii. Plot a point on PR Curve

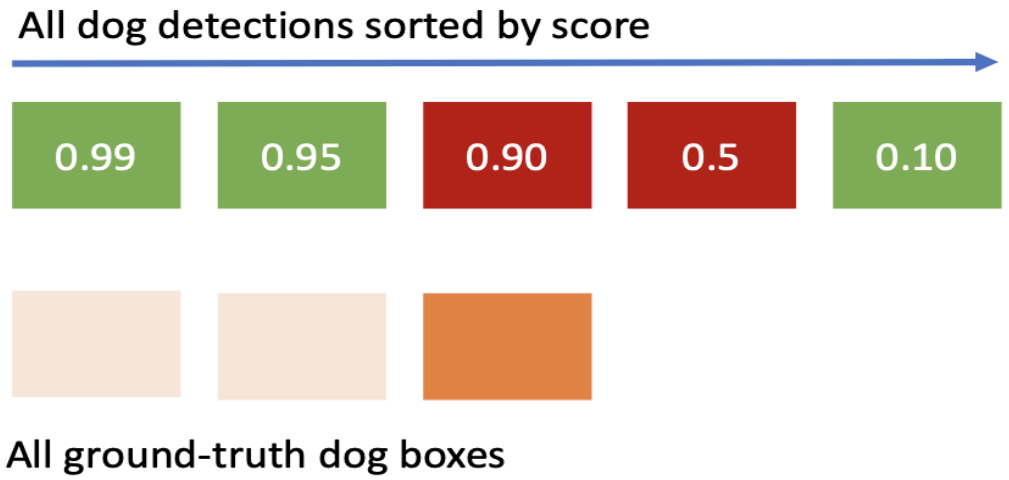


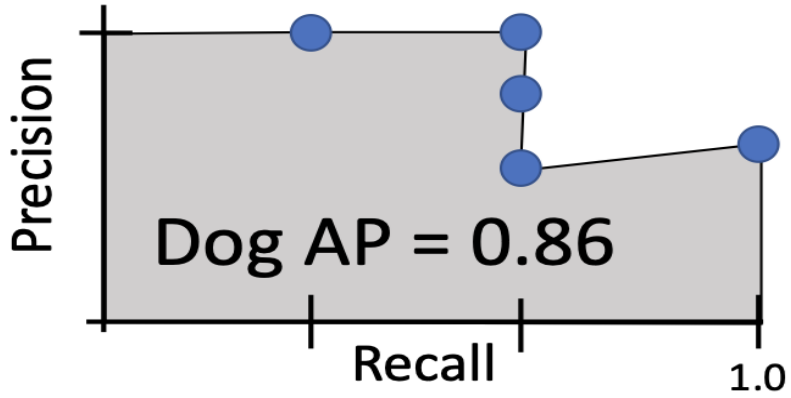


Evaluating Object Detectors: Mean Average Precision (mAP)

1. Run object detector on all test images (with NMS)
2. For each category, compute Average Precision (AP) = area under Precision vs Recall Curve
   a. For each detection (highest score to lowest score)
      i. If it matches some GT box with IoU > 0.5, mark it as positive and eliminate the GT
      ii. Otherwise mark it as negative
      iii. Plot a point on PR Curve
   b. Average Precision (AP) = area under PR curve

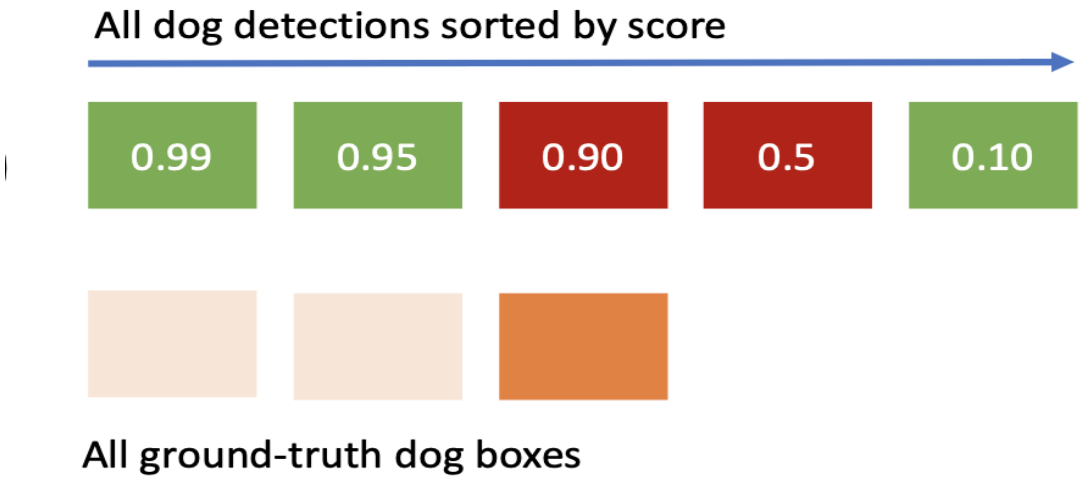
How to get AP = 1.0: Hit all GT boxes with IoU > 0.5, and have no "false positive" detections ranked above any "true positives"

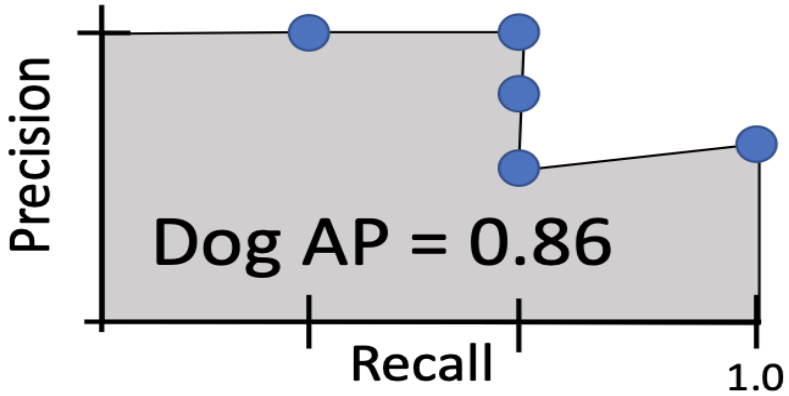




Evaluating Object Detectors: Mean Average Precision (mAP)

1. Run object detector on all test images (with NMS)
2. For each category, compute Average Precision (AP) = area under Precision vs Recall Curve
   a. For each detection (highest score to lowest score)
      i. If it matches some GT box with IoU > 0.5, mark it as positive and eliminate the GT
      ii. Otherwise mark it as negative
      iii. Plot a point on PR Curve
   b. Average Precision (AP) = area under PR curve
   c. Mean Average Precision (mAP) = average of AP for each category





References

- Szileski 2022
  - Section 9.3, 9.4.4
- Forsyth & Ponce 2011
  - Chapter 11
- Baker, Simon, and Iain Matthews. "Lucas-kanade 20 years on: A unifying framework." *International journal of computer vision*56.3 (2004): 221-255.
- Comaniciu, Dorin, and Peter Meer. "Mean shift: A robust approach toward feature space analysis." *IEEE Transactions on Pattern Analysis & Machine Intelligence* 5 (2002): 603-619.
- Comaniciu, Dorin, Visvanathan Ramesh, and Peter Meer. "Real-time tracking of non-rigid objects using mean shift." In Proceedings IEEE Conference on Computer Vision and Pattern Recognition, 2000.
- Bertinetto, Luca, Jack Valmadre, Joao F. Henriques, Andrea Vedaldi, and Philip HS Torr. "Fully-convolutional siamese networks for object tracking." In European conference on computer vision, pp. 850-865. Springer, Cham, 2016.