Lecture 2: Object detection as a machine learning problem

Ross Girshick (FAIR)

AMMI 2022, Computer Vision, Week 2

Learning objectives

- Modern object detection systems are (very) complex
  - It can be difficult to understand how we got here
- What is the underlying logic behind all these design choices?
  - Goal: establish a foundation for understanding object detection research
- Thinking in terms of different levels of problem representation
  - Abstract, mathematical, computational

Recap: The object detection problem

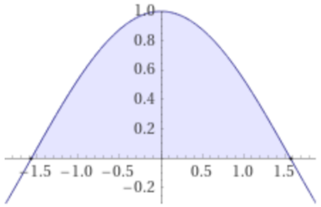
What *objects are in an image and* *where* *are they?*

Levels of problem representation

- Abstract problem formulation
  - “What objects are in an image and where are they?”
- Mathematical problem formulation
  - “How can we describe the above using mathematical language?”
- Computational problem formulation
  - “How can we compute the mathematical model?”
  - Often (not always) involves some approximations (i.e., solving a <u>proxy problem</u>)

Example: Levels of problem representation

- Abstract problem formulation
  - "What is the area under a curve?"



- Mathematical problem formulation
  - "The definite integral from calculus can solve for the area under a curve"
  - E.g., $\int_{-\pi/2}^{\pi/2} \cos(x)\, dx = \sin(x)\big]_{-\pi/2}^{\pi/2} = 2$

- Computational problem formulation
  - "Algorithm: rectangle rule for numerical integration" (an approximation via a proxy problem)
    - Alternative: symbolic system like Wolfram Mathematica

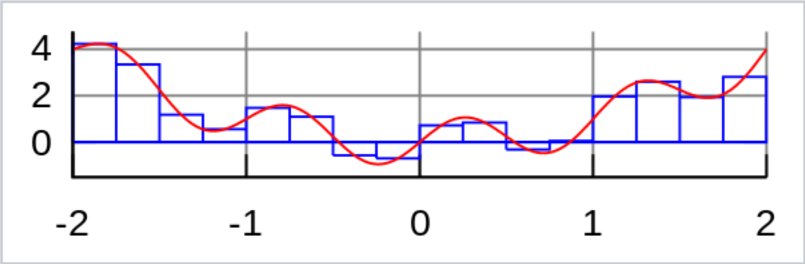


Illustration of the rectangle rule.

https://en.wikipedia.org/wiki/Numerical_integration

From abstract to mathematical problem

- We want to express object detection using the mathematical language of machine learning (ML)

Modeling object detection as an ML problem

- We’ll start with a mathematical formulation

- Input
  - An image $I \in \mathbb{R}^{3 \times H \times W}$

- Output
  - Any finite subset of the (infinitely many) possible boxes in an image
  - For each box in this subset: its category label and confidence score

- Opening our ML toolbox, what do we know how to do?
  - Classification, regression, clustering, …

Modeling object detection as an ML problem

Detection $\coloneqq$ the classification of boxes

Btw, what do we mean by “modeling”?

Detection $\coloneqq$ the classification of boxes

- Object detection is *not* <u>intrinsically</u> “the classification of boxes”
- “Modeling”: We frame the problem in a particular way by choice
  - Some framings may be very natural; others less so
- Other modeling choices are possible!
  - Future detection systems may do things differently – open research directions

Modeling object detection as an ML problem

Detection $\coloneqq$ the classification of boxes

This classic strategy has been used since time immemorial …

H. A. Rowley, S. Baluja, and T. Kanade. Neural networkbased face detection. TPAMI, 1998
R. Vaillant, C. Monrocq, and Y. LeCun. Original approach for the localisation of objects in images. IEE Proc on Vision, Image, and Signal Processing, 1994.
N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In CVPR, 2005
P. Felzenszwalb, R. Girshick, D. McAllester, and D. Ramanan. Object detection with discriminatively trained part based models. TPAMI, 2010
J. Uijlings, K. van de Sande, T. Gevers, and A. Smeulders. Selective search for object recognition. IJCV, 2013.
R. Girshick, J. Donahue, T. Darrell, and J. Malik, Rich feature hierarchies for accurate object detection and semantic segmentation. In CVPR, 2014
R. Girshick, Fast R-CNN. In ICCV, 2015.
… *literally **thousands** of papers* …

Detection ≔ the classification of boxes

- What are the labels?
  - One catch-all "background" class: 0; C object ("foreground") classes: 1, …, C
  - The set of boxes $\mathbb{B}$ in an image $I \in \mathbb{I}$ is infinite (assuming real numbers)

- Classification rule: each box in an image belongs to a class in $\{0, \ldots, C\}$
  - The true mapping is $f\colon \mathbb{I} \longmapsto \{0, \ldots, C\}^{|\mathbb{B}|}$
  - If a box is in the ground-truth (g.t.) set, its class is the g.t. label in $\{1, \ldots, C\}$
  - Otherwise, the box's class is 0 (background)

- Train a classifier to learn this classification rule (*i.e.*, to classify every box)

From mathematical to computational problem

- We want to run something on a computer, not just have a mathematical description on paper
- How can we compute this mathematical model?
- Recall the definite integral example
  - Infinites and infinitesimals – the enemies of computers!
  - Prepare yourself for some approximations

How do we compute this problem?

1. “Too many” boxes (infinite)

A cascade of challenges emerges…

1. “Too many” boxes (infinite) → switch to a proxy problem

A cascade of challenges emerges…

1. “Too many” boxes (infinite) → switch to a proxy problem

2. Broken classification rule

A cascade of challenges emerges…

1. "Too many" boxes (infinite) → switch to a proxy problem

2. Broken classification rule → design label assignment heuristics

A cascade of challenges emerges…

1. "Too many" boxes (infinite) → switch to a proxy problem

2. Broken classification rule → design label assignment heuristics

3. Duplicate detections

A cascade of challenges emerges…

1. "Too many" boxes (infinite) → switch to a proxy problem
2. Broken classification rule → design label assignment heuristics
3. Duplicate detections → cluster outputs into instances

A cascade of challenges emerges…

1. “Too many” boxes (infinite) → switch to a proxy problem
2. Broken classification rule → design label assignment heuristics
3. Duplicate detections → cluster outputs into instances
4. Foreground-background imbalance (intrinsic)

A cascade of challenges emerges…

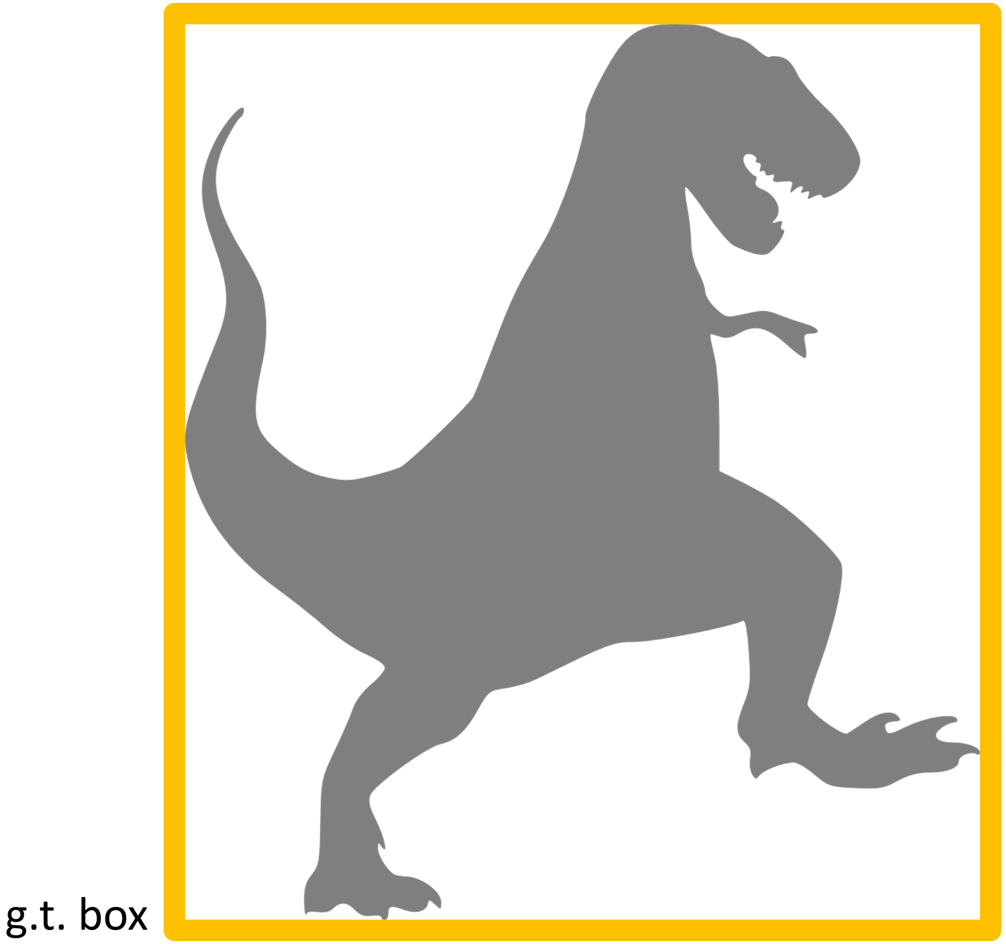
1. “Too many” boxes (infinite) → switch to a proxy problem
2. Broken classification rule → design label assignment heuristics
3. Duplicate detections → cluster outputs into instances
4. Foreground-background imbalance (intrinsic) → balanced sampling, novel losses, cascades, …

Recap: From abstract to computational

- We started with an abstract problem of object detection
- We formulated it as a mathematical ML problem
- Translating this to a computational problem introduces challenges
  - We will go through them one by one now

Challenge 1: Too many boxes

- Detection output space is infinite

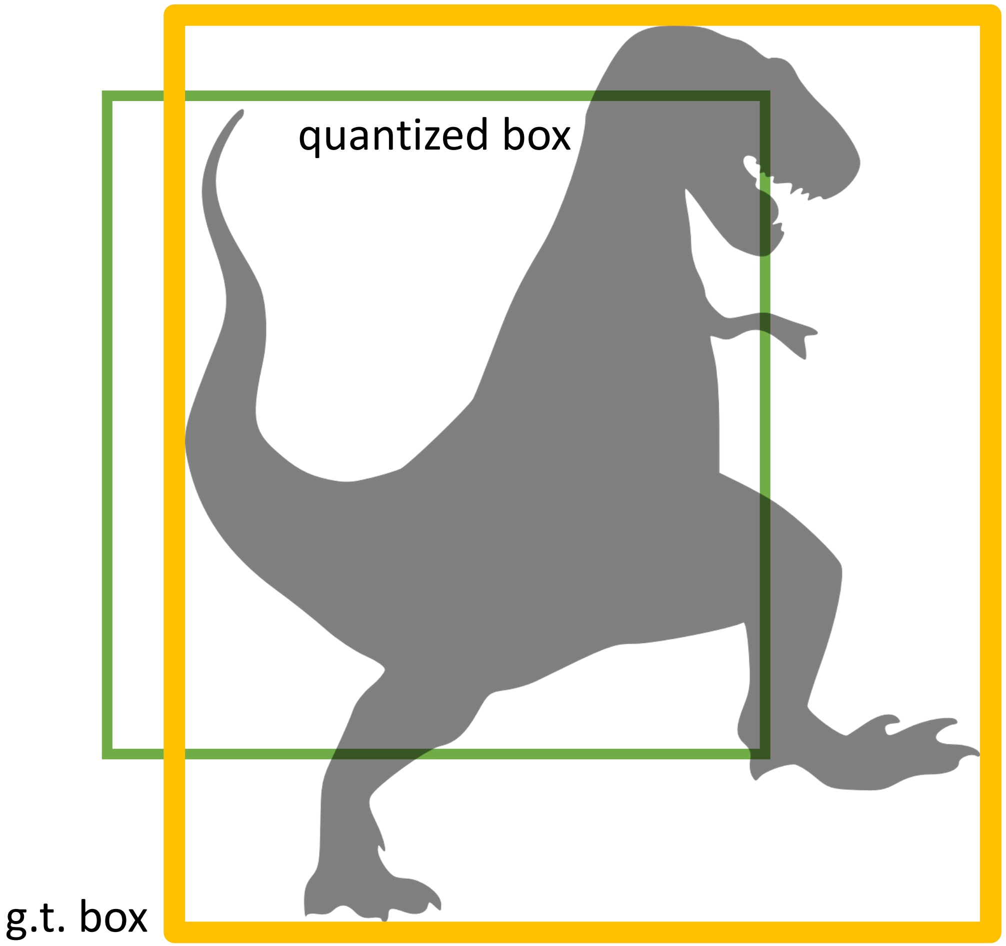


Challenge 1: Too many boxes

Standard solution

- Detection output space is infinite

- Approximate the infinite set of all possible boxes with a finite set of boxes (i.e., *quantize* the set)

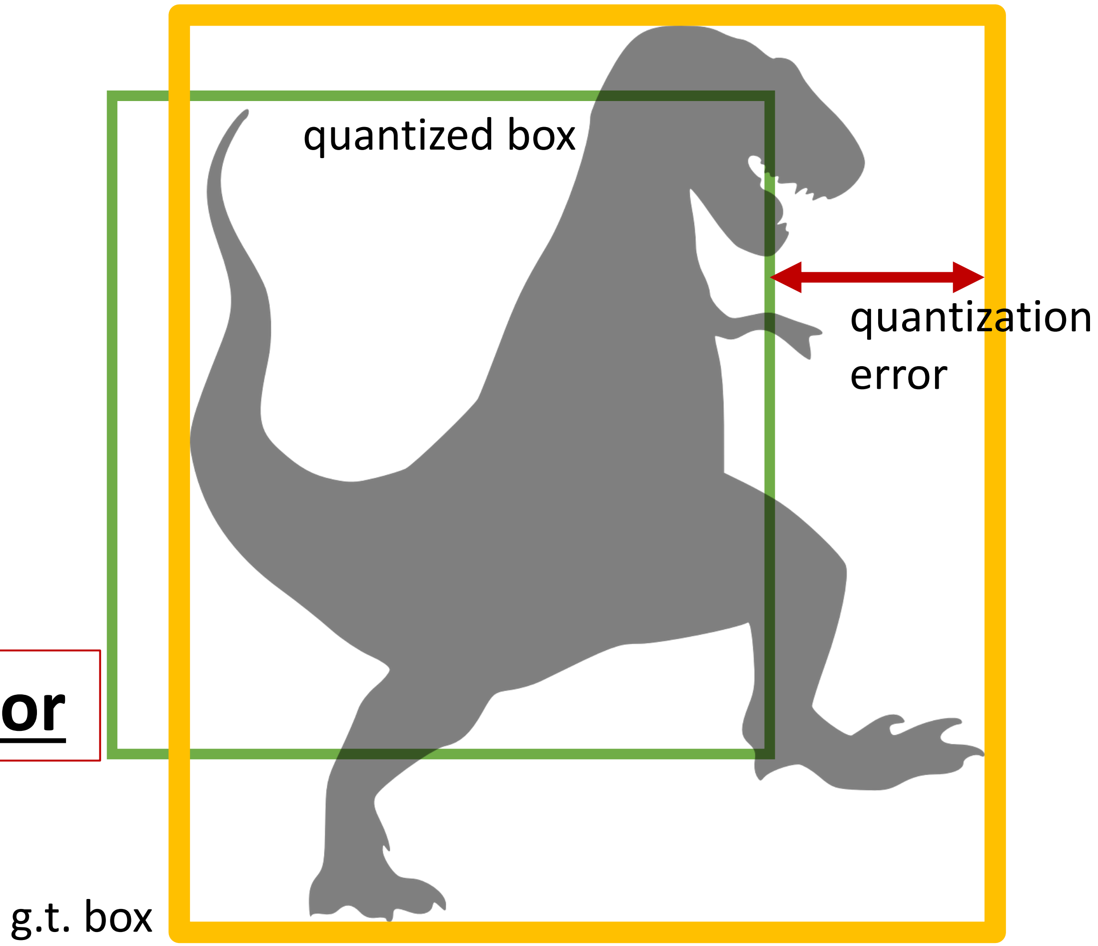


Challenge 1: Too many boxes

Standard solution

- Detection output space is infinite
- Approximate the infinite set of all possible boxes with a finite set of boxes (i.e., *quantize* the set)

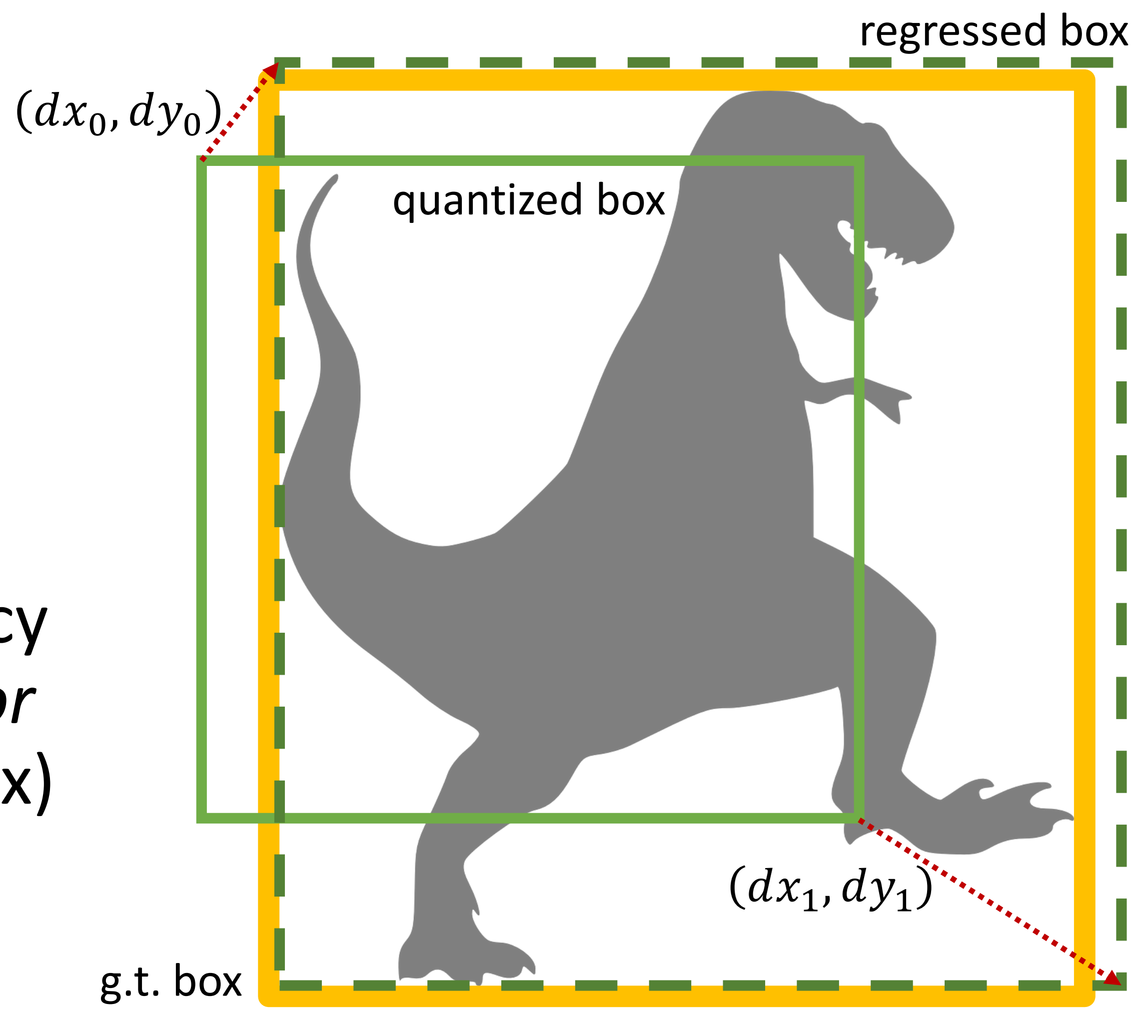
This approximation creates quantization error



Challenge 1: Too many boxes

Standard solution

- Detection output space is infinite
- Approximate the infinite set of all possible boxes with a finite set of boxes (i.e., *quantize* the set)
- Recover loss of localization accuracy by predicting the *quantization error* to cancel it (i.e., *regress* the g.t. box)

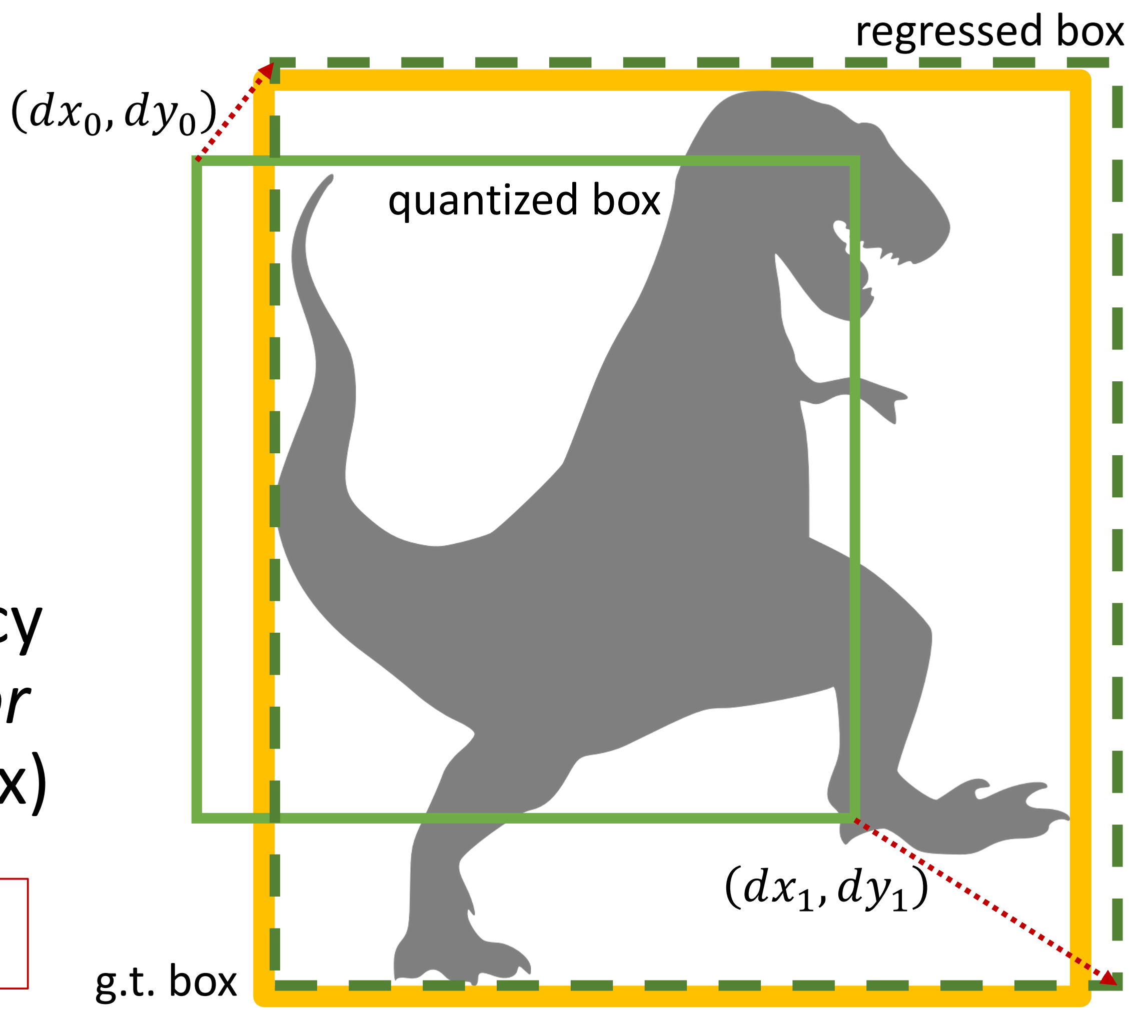


Challenge 1: Too many boxes

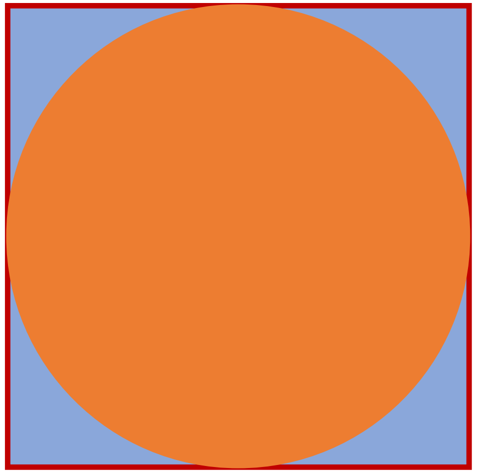
Standard solution

- Detection output space is infinite
- Approximate the infinite set of all possible boxes with a finite set of boxes (i.e., *quantize* the set)
- Recover loss of localization accuracy by predicting the *quantization error* to cancel it (i.e., *regress* the g.t. box)

We have switched to a <u>proxy problem</u>

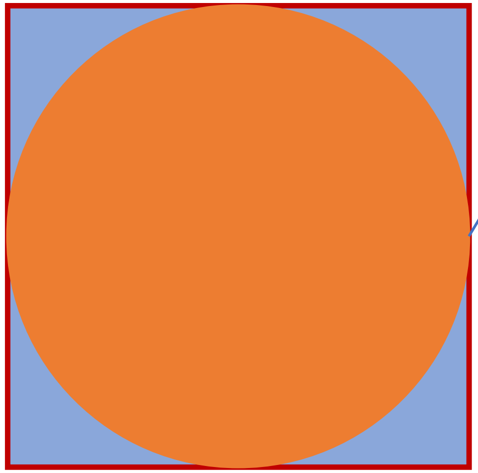


Foundational concept: Sliding-window detector



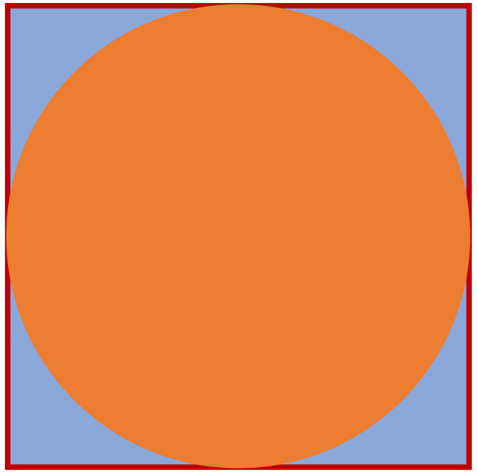
Template
for “pumpkin”
category

Foundational concept: Sliding-window detector



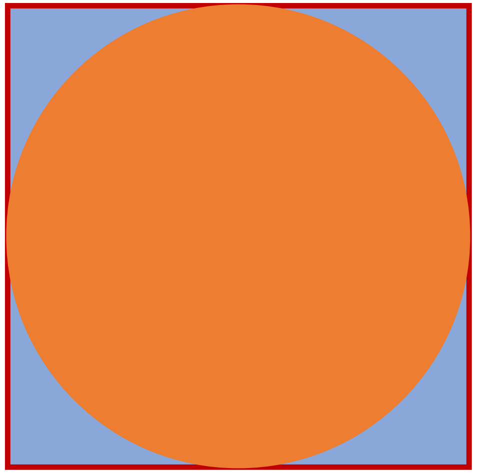
Template
for “pumpkin”
category

Foundational concept: Sliding-window detector

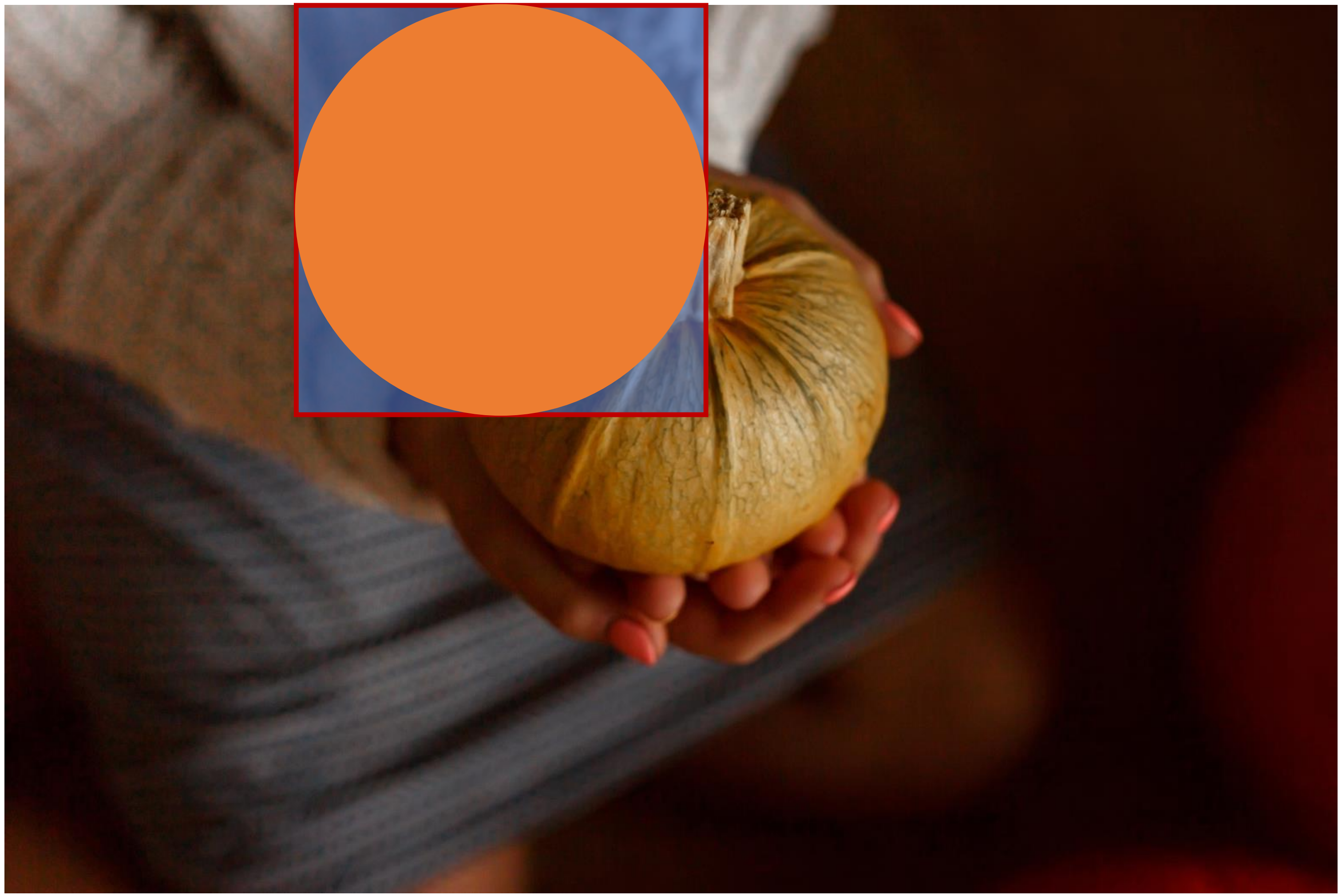


Template
for “pumpkin”
category

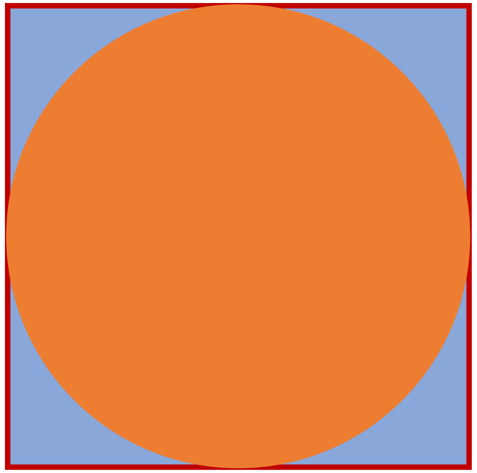
Foundational concept: Sliding-window detector



Template
for “pumpkin”
category

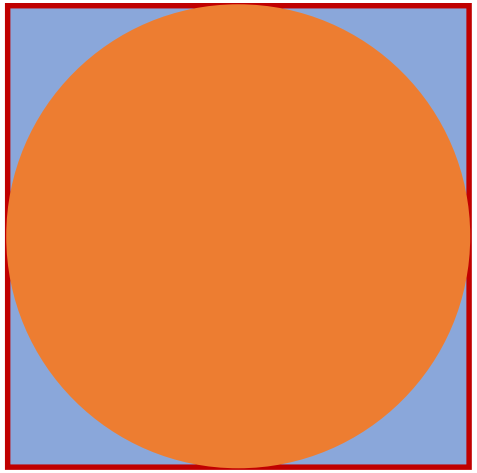


Foundational concept: Sliding-window detector



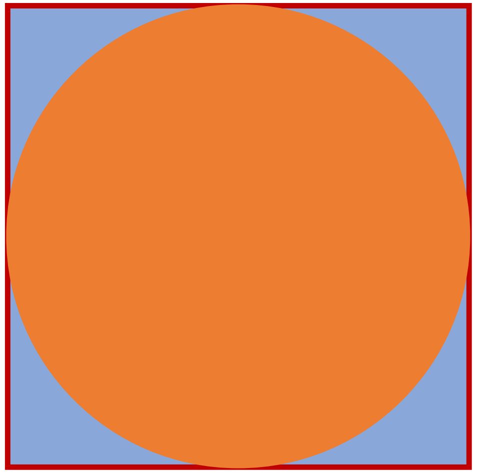
Template
for “pumpkin”
category

Foundational concept: Sliding-window detector



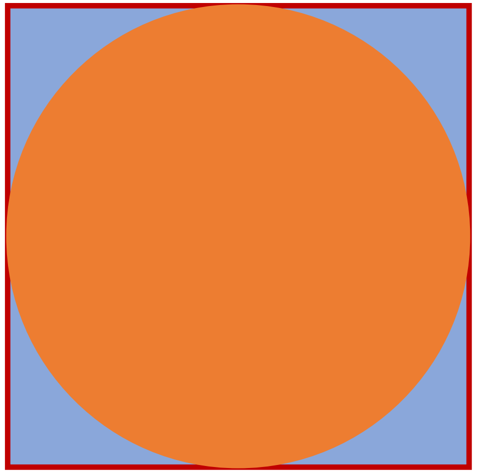
Template
for “pumpkin”
category

Foundational concept: Sliding-window detector

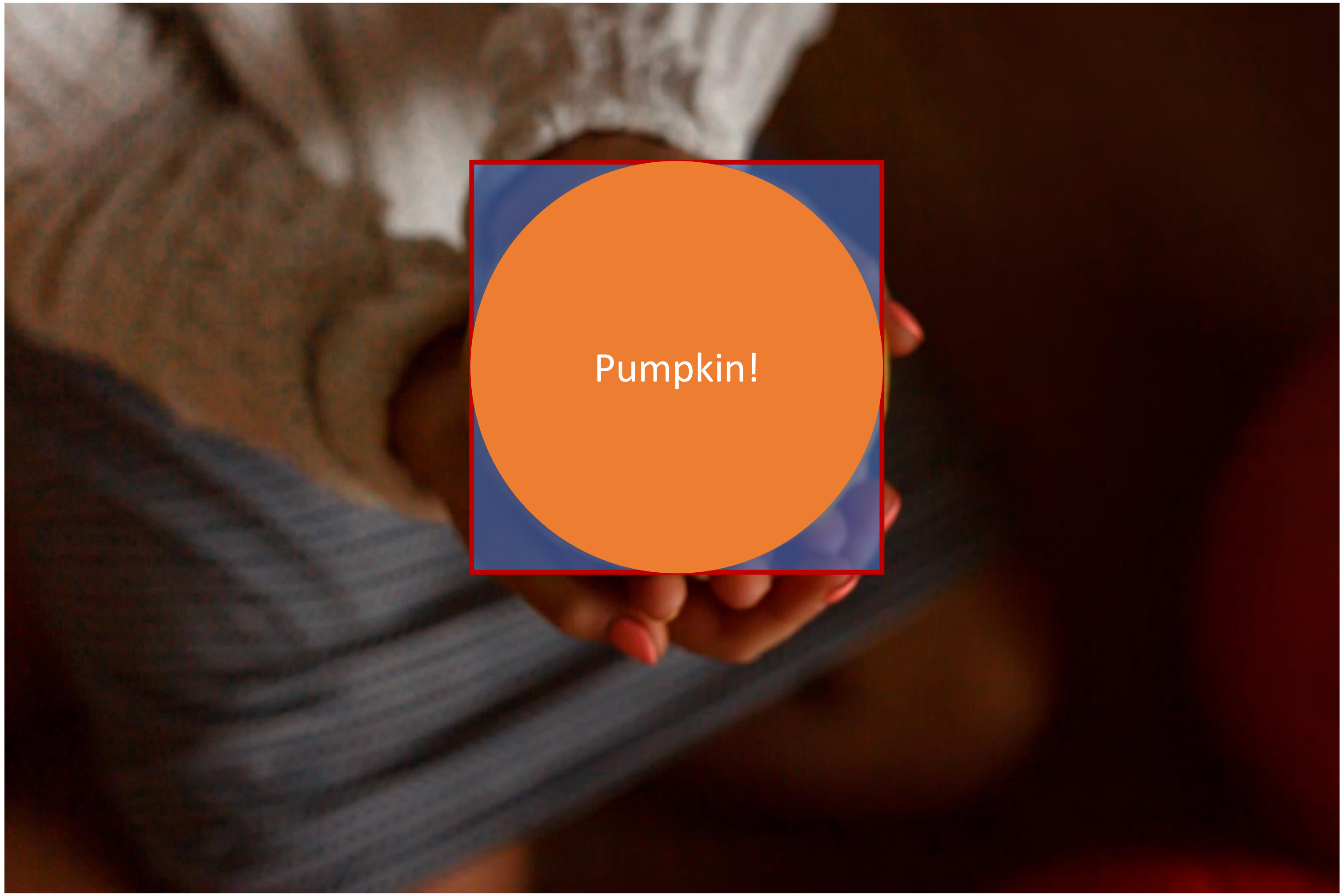


Template
for “pumpkin”
category

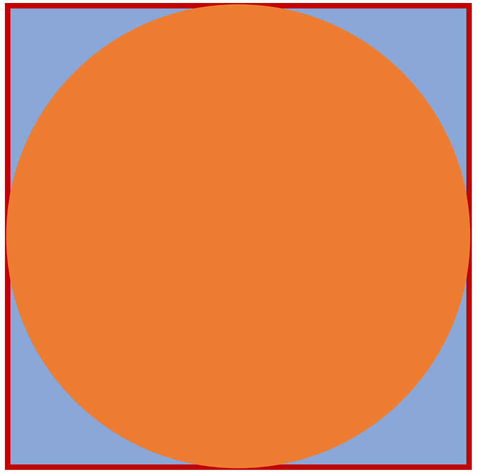
Foundational concept: Sliding-window detector



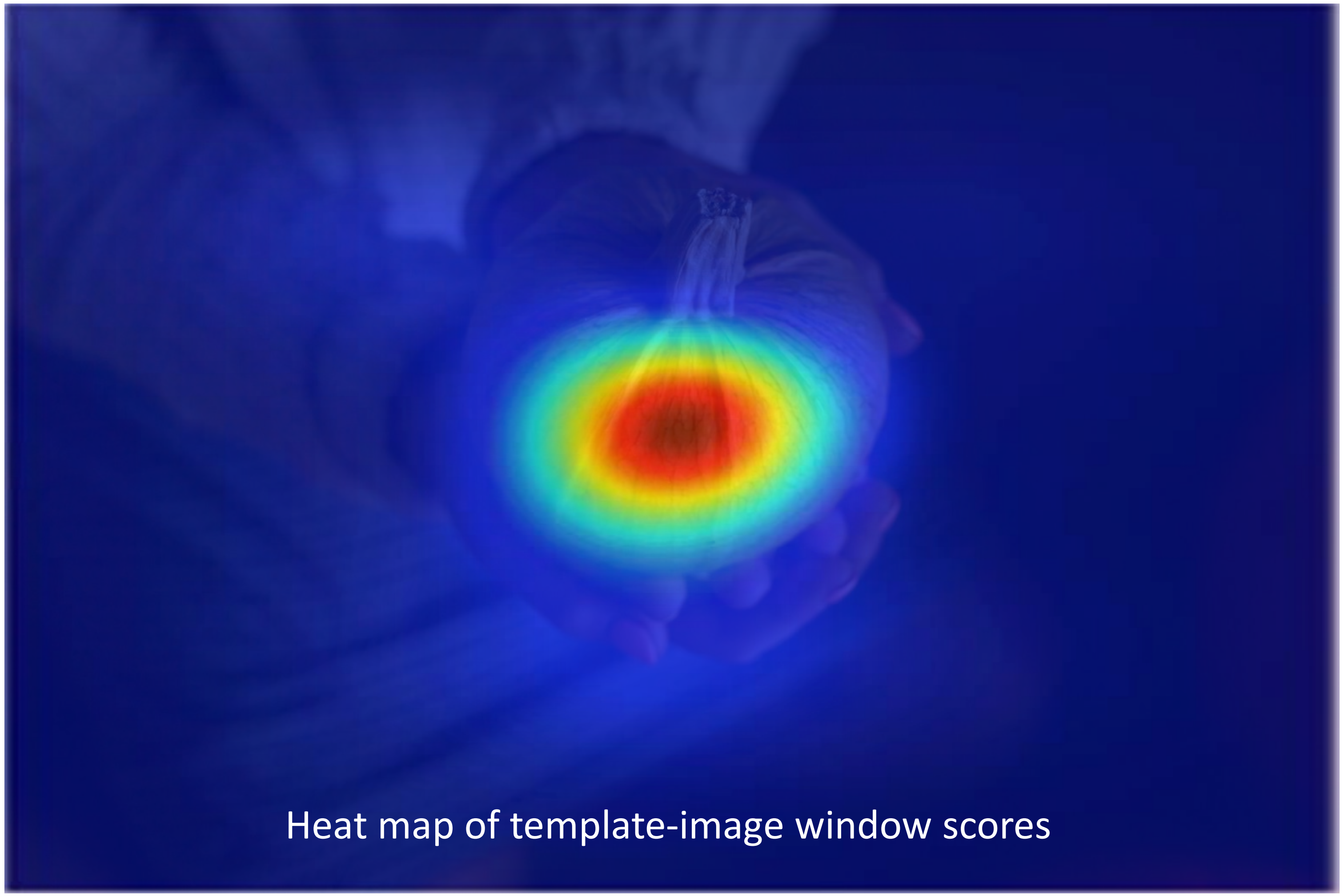
Template
for “pumpkin”
category



Foundational concept: Sliding-window detector

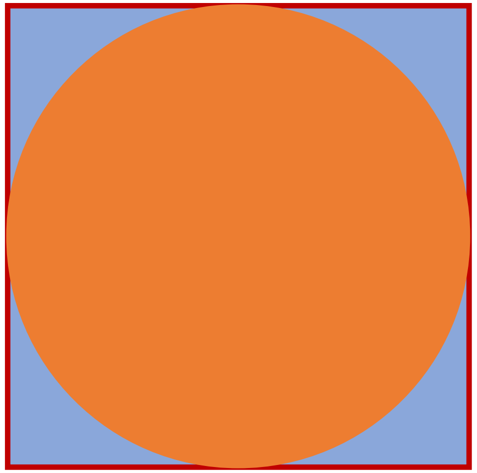


Template
for "pumpkin"
category



Heat map of template-image window scores

Foundational concept: Sliding-window detector



Template
for “pumpkin”
category

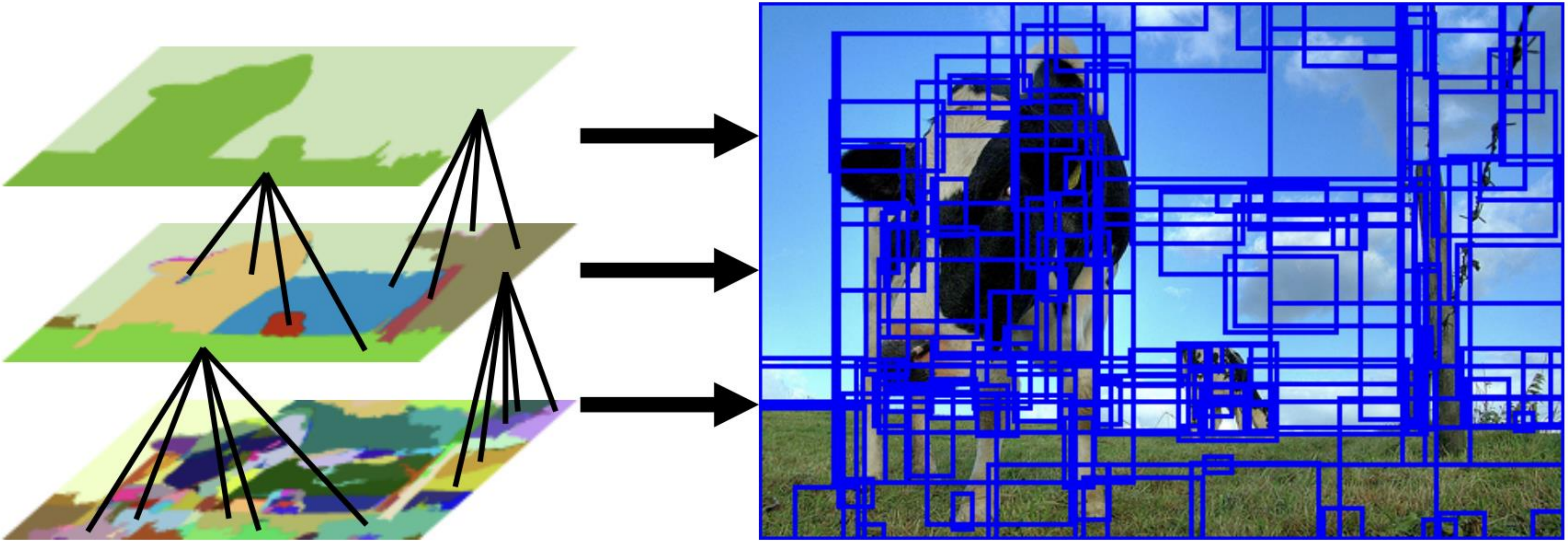
Where does the template come from?

- It’s an ML model, trained on data
- Simple case: it’s a linear filter
  - This is a convolution! (Technically cross-correlation)
- More sophisticated: the “template” is a neural network
- ConvNets are built from many layers of sliding-window feature detectors

Foundational concept: Region proposals

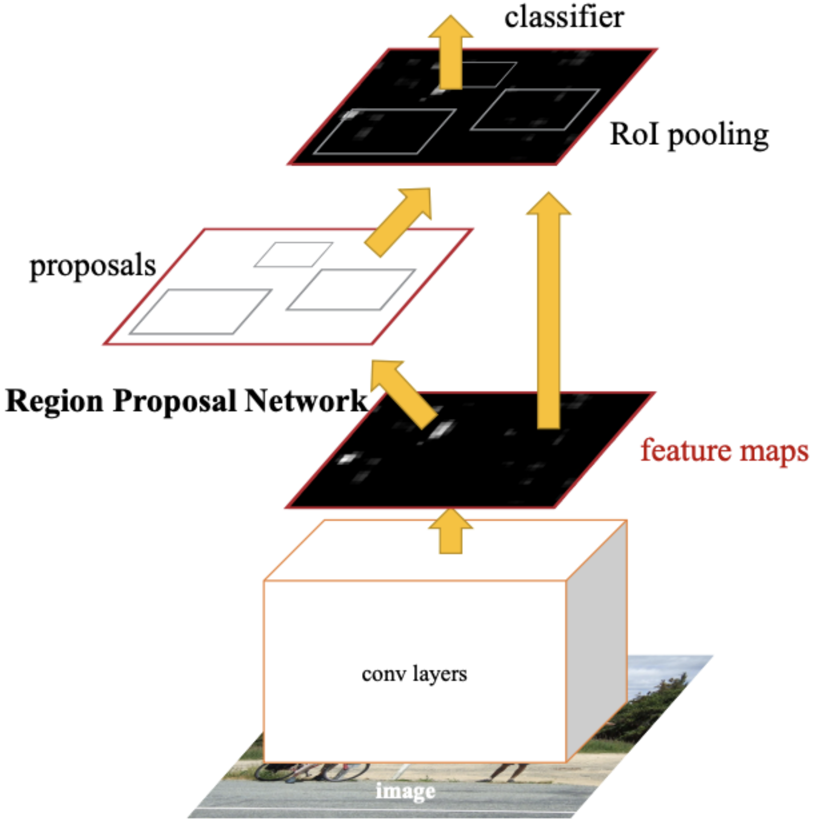
- Sparse, irregular set of boxes (called “regions”)
- Each one is a proposed candidate object location
- A downstream classifier will classify each proposal
- Proposals need to have high recall, while not being too numerous

Foundational concept: Region proposals



Strategy 1: Bottom-up region proposals from the Selective Search algorithm [Uijlings et al. 2012]

Foundational concept: Region proposals



- A generic object vs. not-object sliding window detector is trained
- It's high-scoring output are taken as proposals for use in the downstream classifier

Strategy 2: Top-down objectness classifier from Region Proposal Network [Ren et al. 2015]

Consequence of the proxy problem

[Recall:

- Classification rule: each box in an image belongs to a class in $\{0, \ldots, C\}$
  - The true mapping is $f\colon \mathbb{I} \longmapsto \{0, \ldots, C\}^{|\mathbb{B}|}$
  - If a box is in the ground-truth (g.t.) set, its class is the g.t. label in $\{1, \ldots, C\}$
  - Otherwise, the box's class is 0 (background)

]

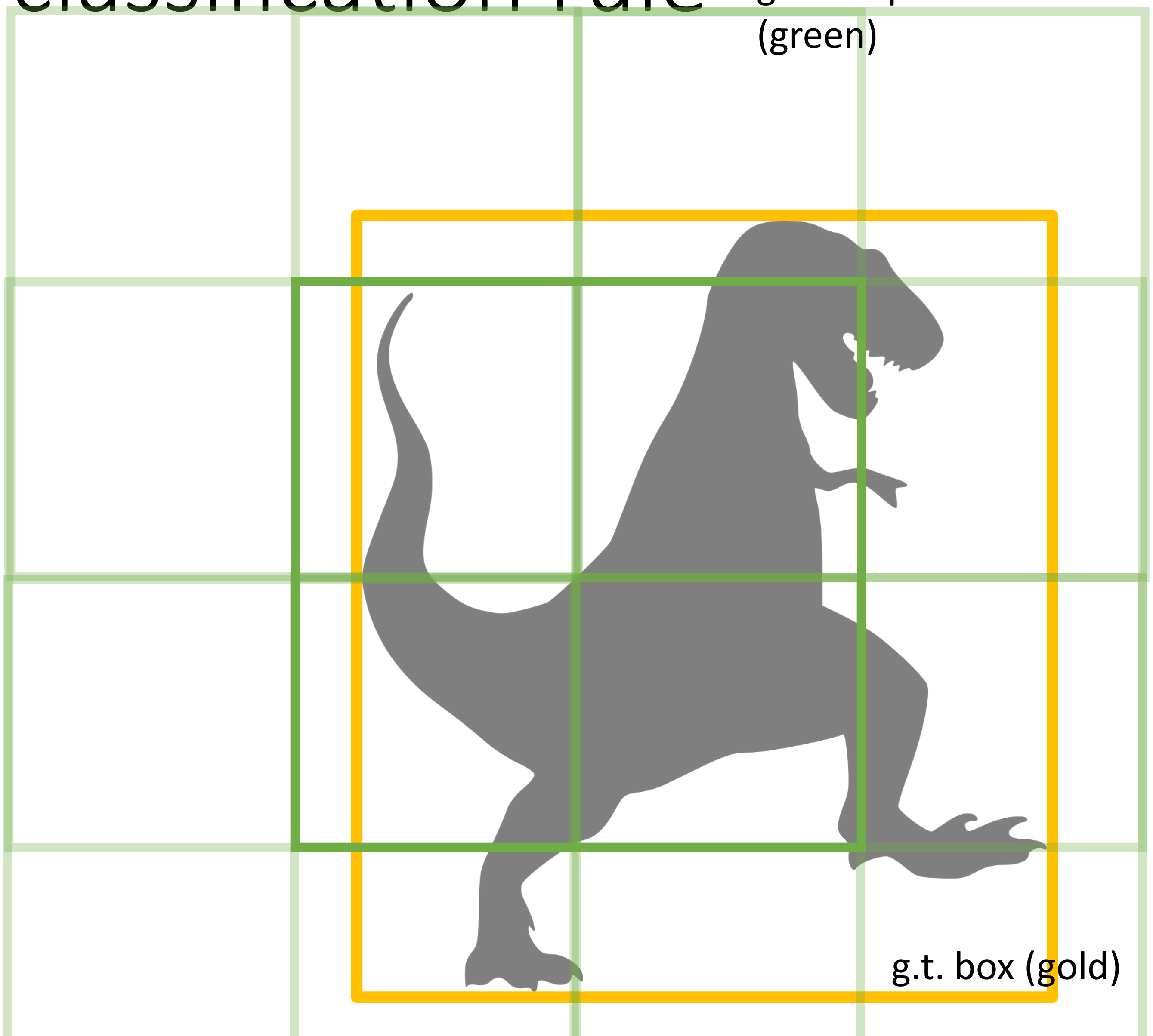
- This classification rule is no longer meaningful for solving detection
  - It's "broken" by the introduction of quantized boxes
  - Why?

Challenge 2: Broken classification rule

- Why? Ground-truth boxes are not (likely) in the set of quantized boxes

Challenge 2: Broken classification rule

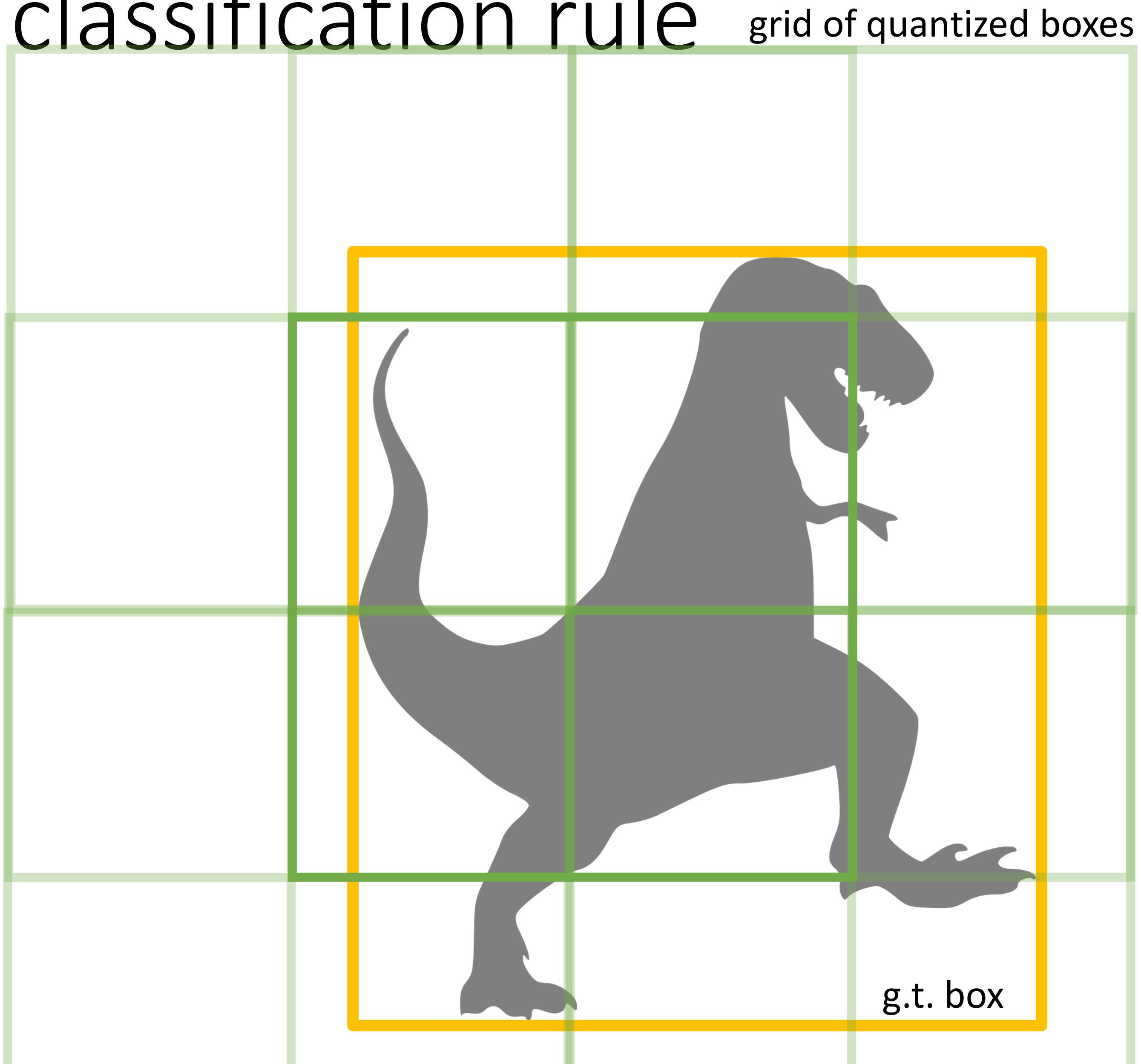
- Why? Ground-truth boxes are not (likely) in the set of quantized boxes



Challenge 2: Broken classification rule

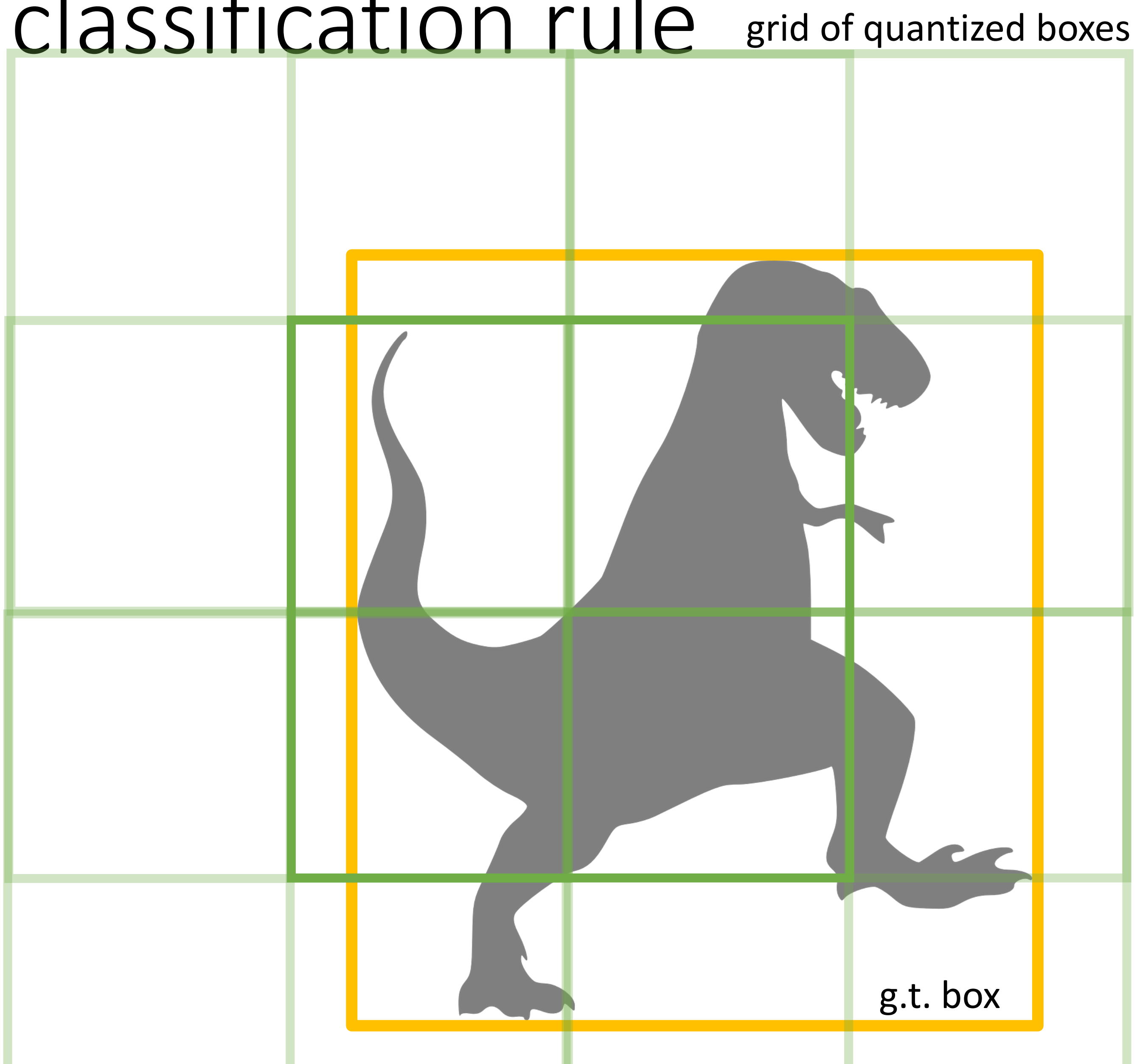
- Why? Ground-truth boxes are not (likely) in the set of quantized boxes

- $f(I) \rightarrow \{0, \ldots, 0\}$ almost always – not useful!



Challenge 2: Broken classification rule

- Why? Ground-truth boxes are not (likely) in the set of quantized boxes
- $f(I) \rightarrow \{0, \ldots, 0\}$ almost always – not useful!
- We need to redefine f, *i.e.*, specify $\hat{f}$ via a g.t. label assignment heuristic



Challenge 2: Broken classification rule

Standard solution

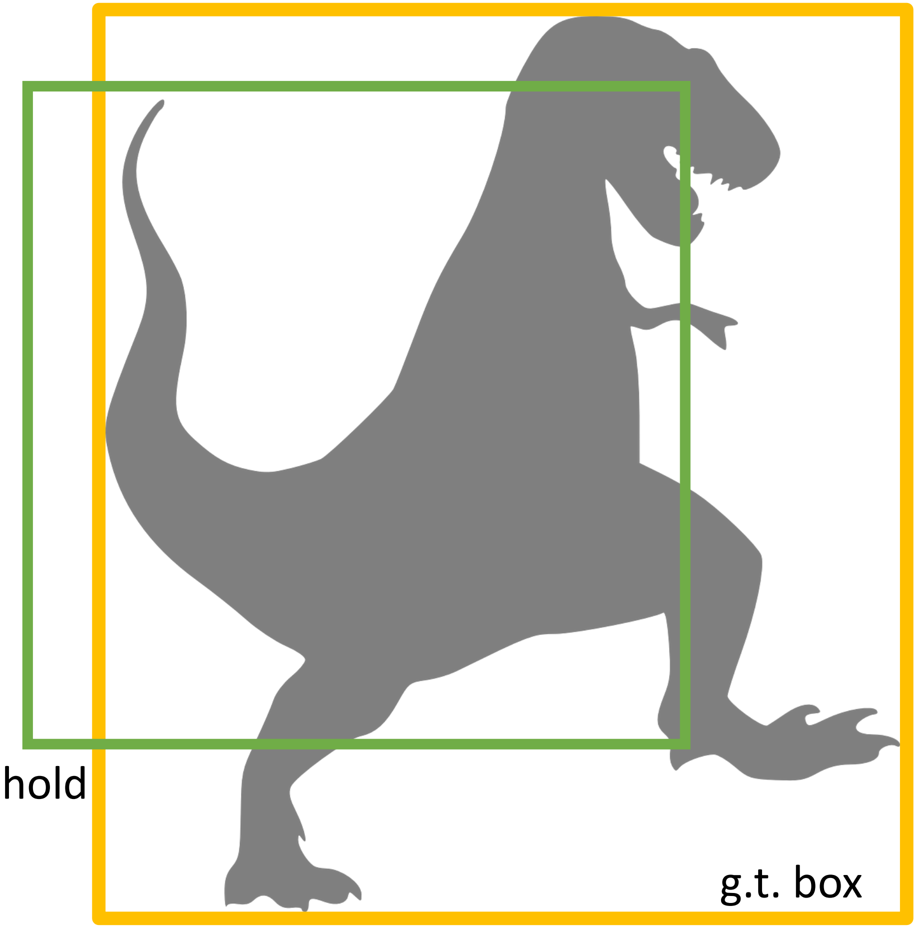
- Assign each quantized box to zero or one g.t. boxes
- Using a "labeling heuristic", e.g.:
  - IoU thresholds,
  - centeredness,
  - *etc.*

Challenge 2: Broken classification rule

Standard solution

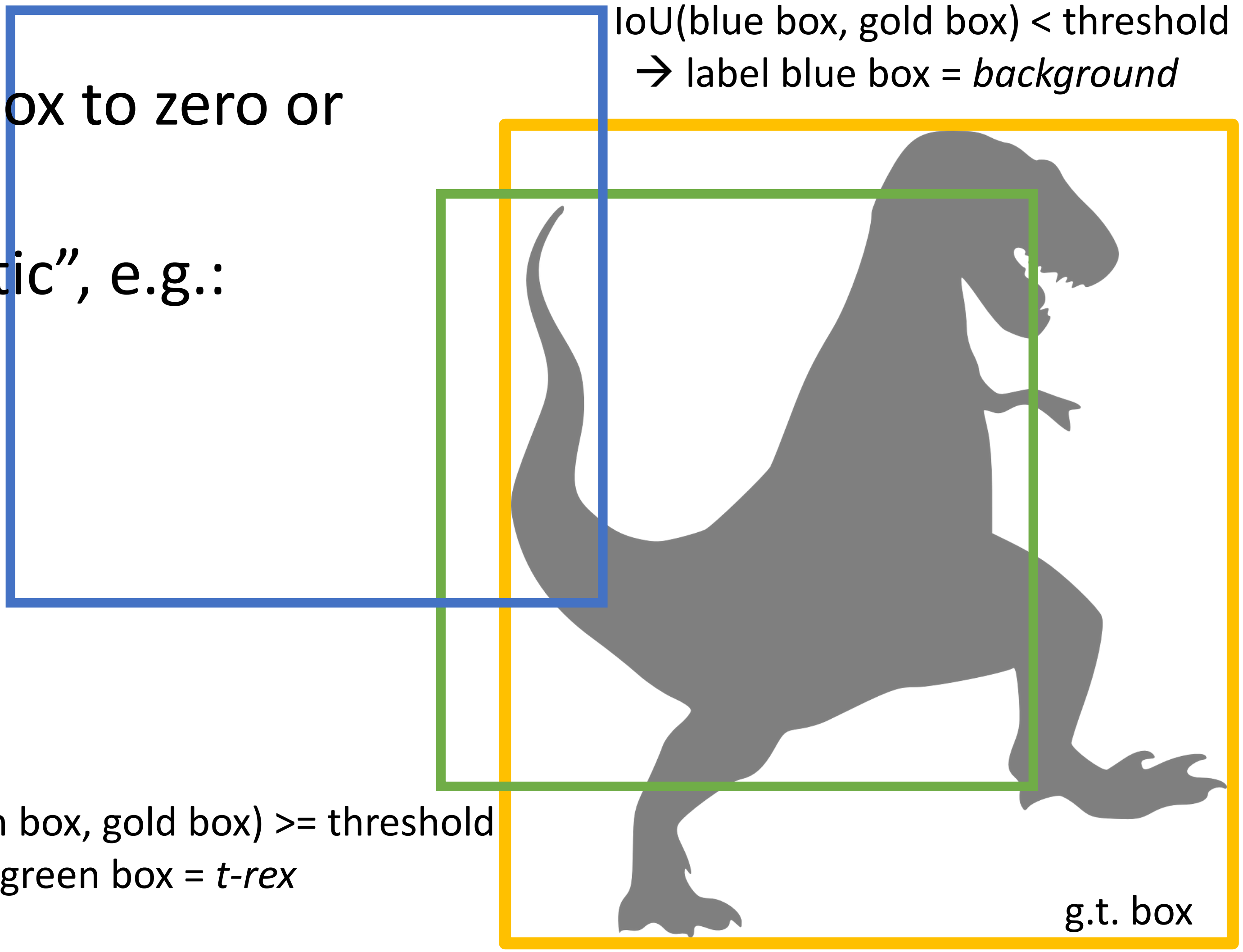
- Assign each quantized box to zero or one g.t. boxes
- Using a "labeling heuristic", e.g.:
  - IoU thresholds,
  - centeredness,
  - *etc.*

IoU(green box, gold box) >= threshold
→ label green box = *t-rex*



Challenge 2: Broken classification rule Standard solution

- Assign each quantized box to zero or one g.t. boxes
- Using a “labeling heuristic”, e.g.:
  - <u>IoU thresholds</u>,
  - centeredness,
  - *etc.*



Challenge 2: Broken classification rule

Standard solution

- Proxy classification rule: each quantized box in an image belongs to a class in $\{0, \ldots, C\}$
  - The proxy mapping is $\hat{f}\colon \mathbb{I} \longmapsto \{0, \ldots, C\}^{|\widehat{\mathbb{B}}|}$ ($\widehat{\mathbb{B}}$ is the set of quantized boxes)
  - If a quantized box B was assigned to a g.t. box G, the g.t. label for B is taken from G
  - Otherwise (i.e., no assignment), B's class is 0 (background)

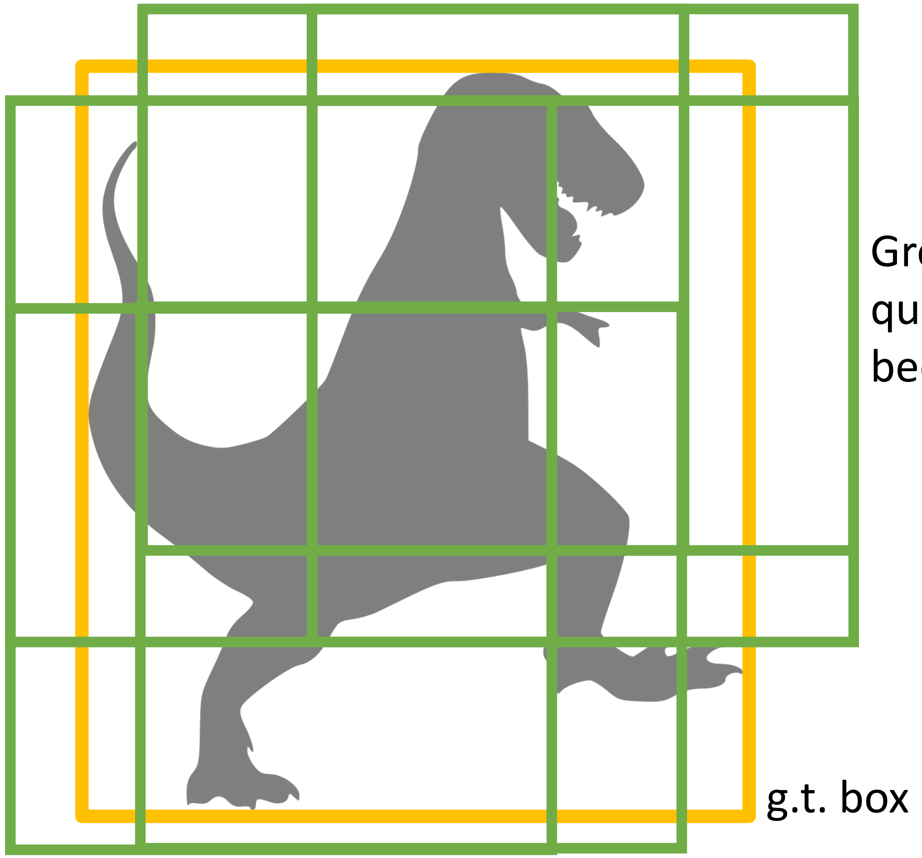
- Train a classifier to learn the proxy classification rule $\hat{f}$
- Train a regressor to learn the regression function from B to G

Consequence of the proxy classification rule

- No <u>single</u> quantized box is correct

Consequence of the proxy classification rule

- No <u>single</u> quantized box is correct
  - Each g.t. box is <u>spatially spread</u> to nearby quantized boxes



Green boxes: just a few of the quantized boxes labeled *t-rex* because of <u>only one</u> g.t. box

Consequence of the proxy classification rule

- No <u>single</u> quantized box is correct
  - Each g.t. box is <u>spatially spread</u> to nearby quantized boxes

- The proxy classification rule explicitly asks for <u>duplicate detections</u>

Consequence of the proxy classification rule

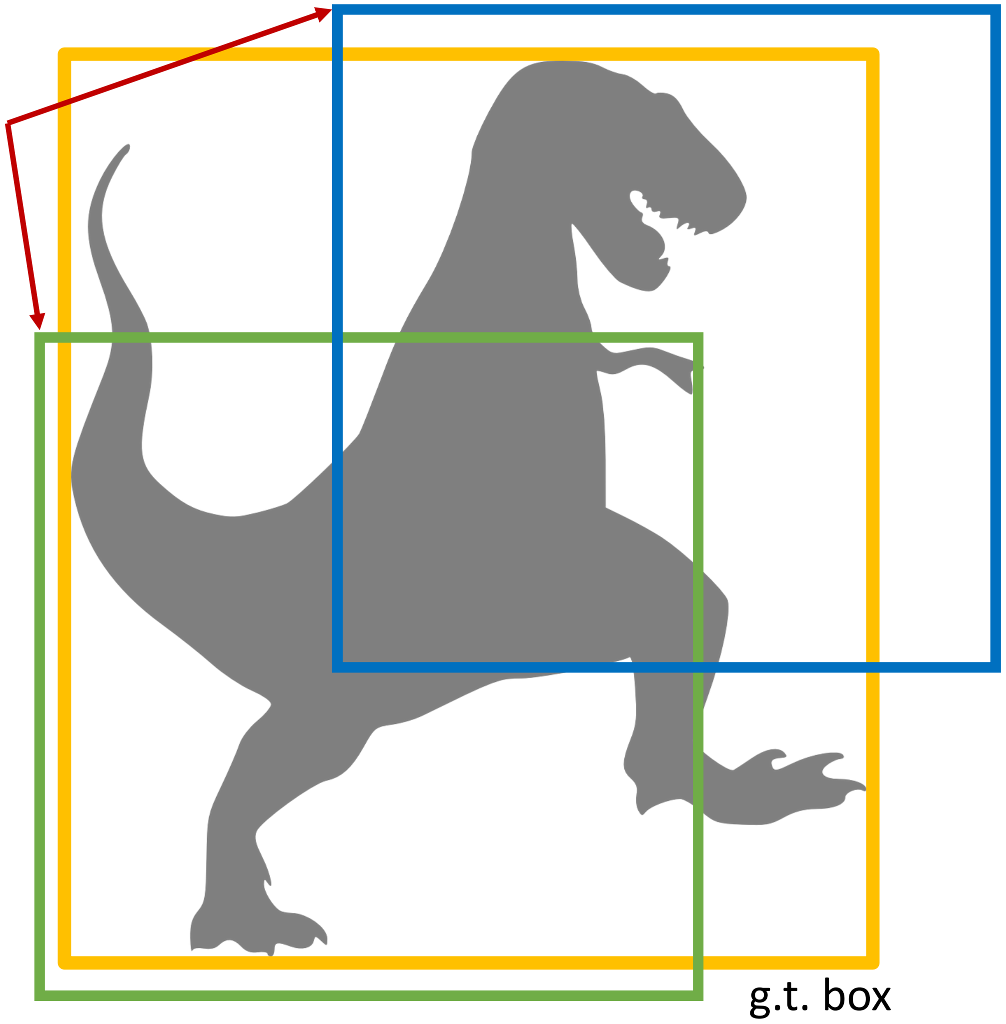
- No single quantized box is correct
  - Each g.t. box is spatially spread to nearby quantized boxes
- The proxy classification rule explicitly asks for duplicate detections
  - Recall: duplicates are undesirable and punished by AP!

Consequence of the proxy classification rule

- No single quantized box is correct
  - Each g.t. box is spatially spread to nearby quantized boxes
- The proxy classification rule explicitly asks for duplicate detections
  - Recall: duplicates are undesirable and punished by AP!
- Removing duplicates requires a post-processing (“clean up”) step to fix the the proxy classification rule

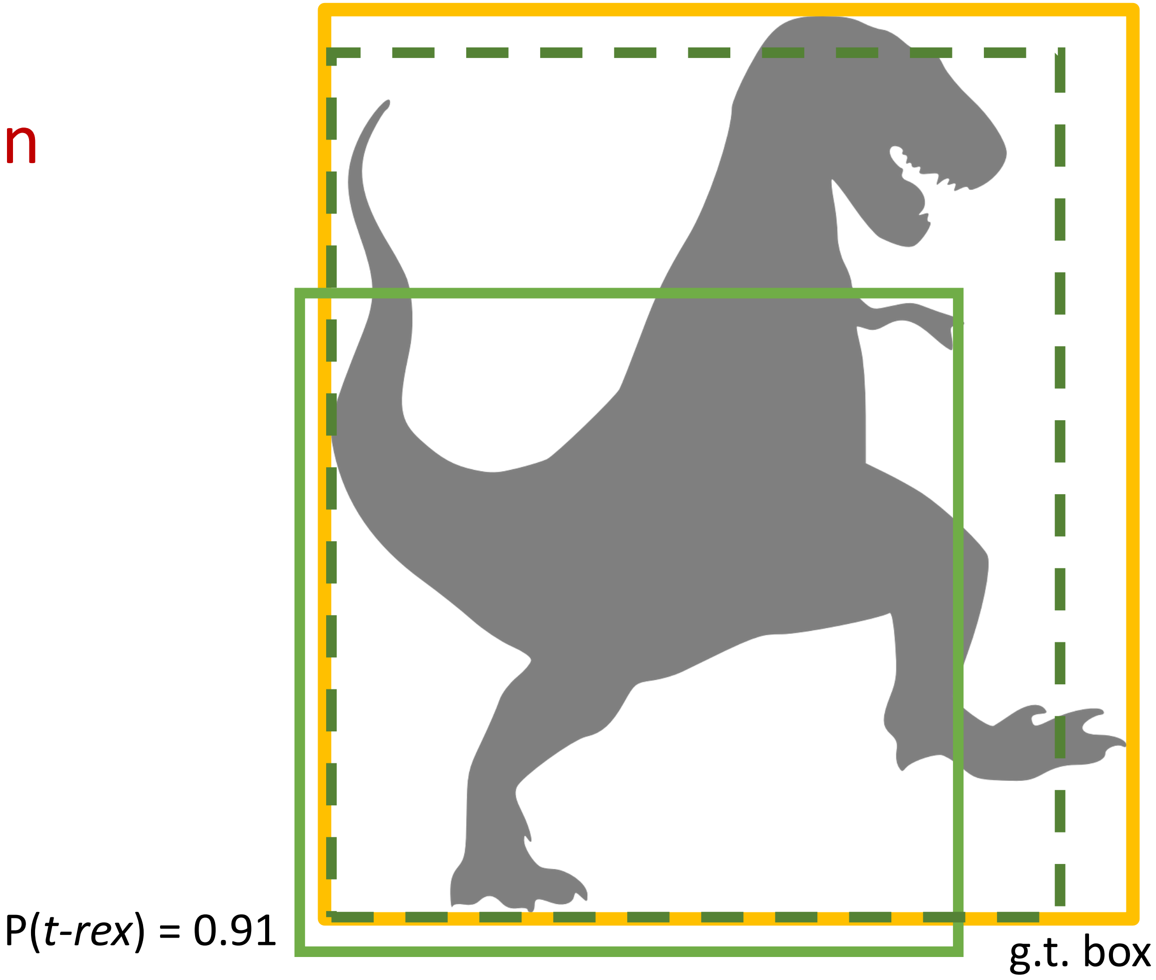
Challenge 3: Duplicate detections

Two (quantized) boxes,
both assigned label *t-rex*
because of the <u>same</u> g.t. box



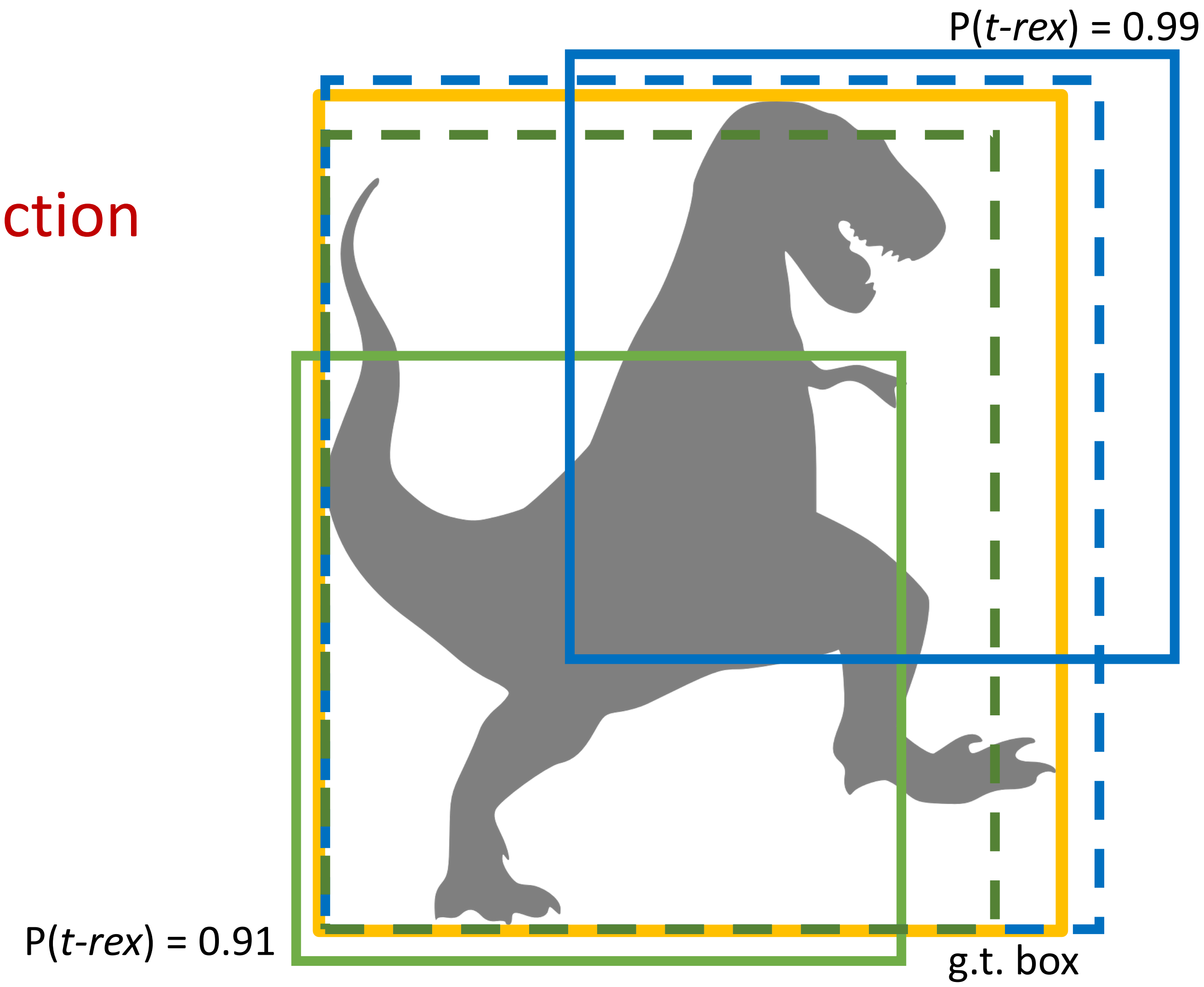
Challenge 3: Duplicate detections

First prediction

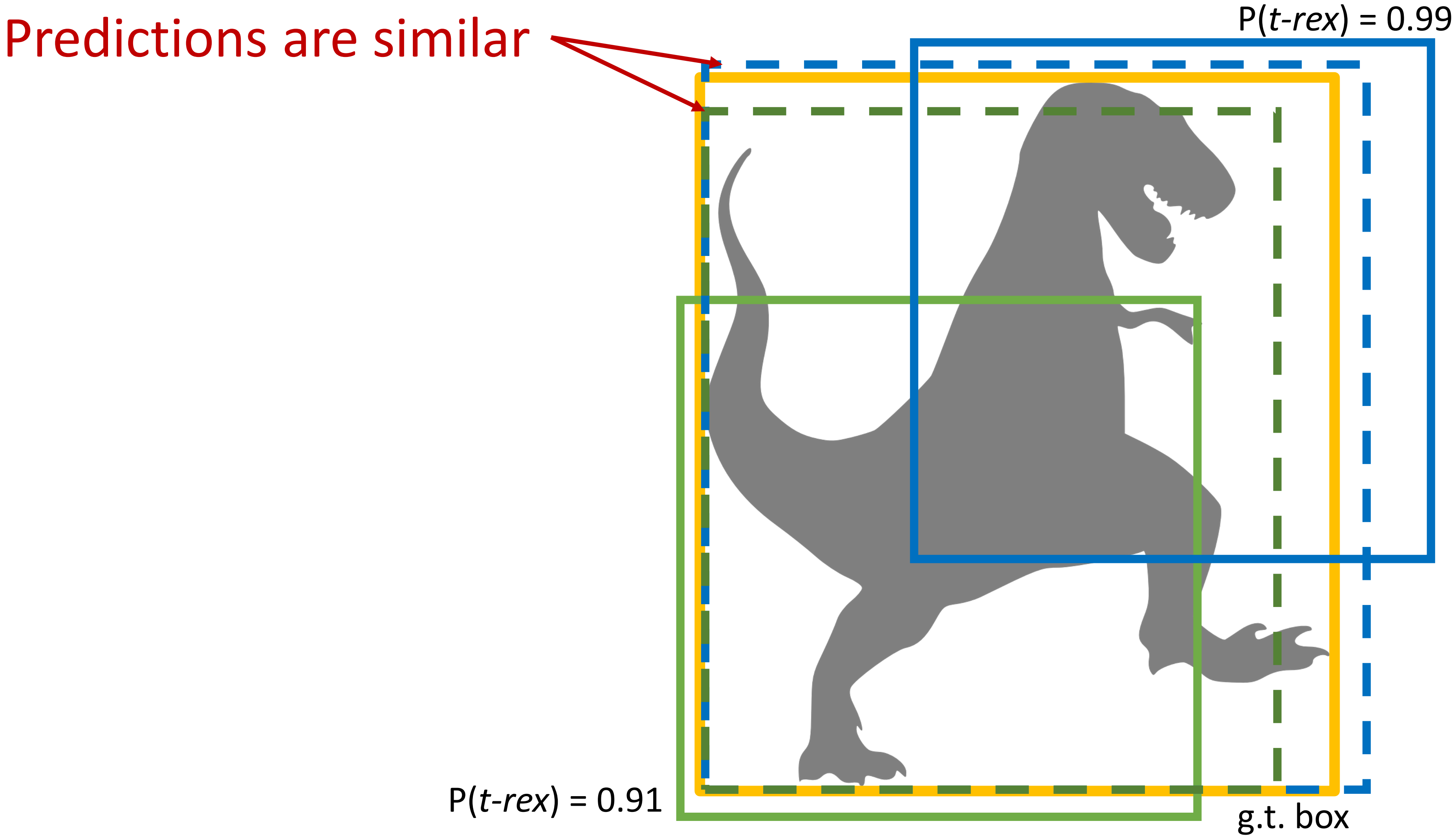


Challenge 3: Duplicate detections

Second prediction



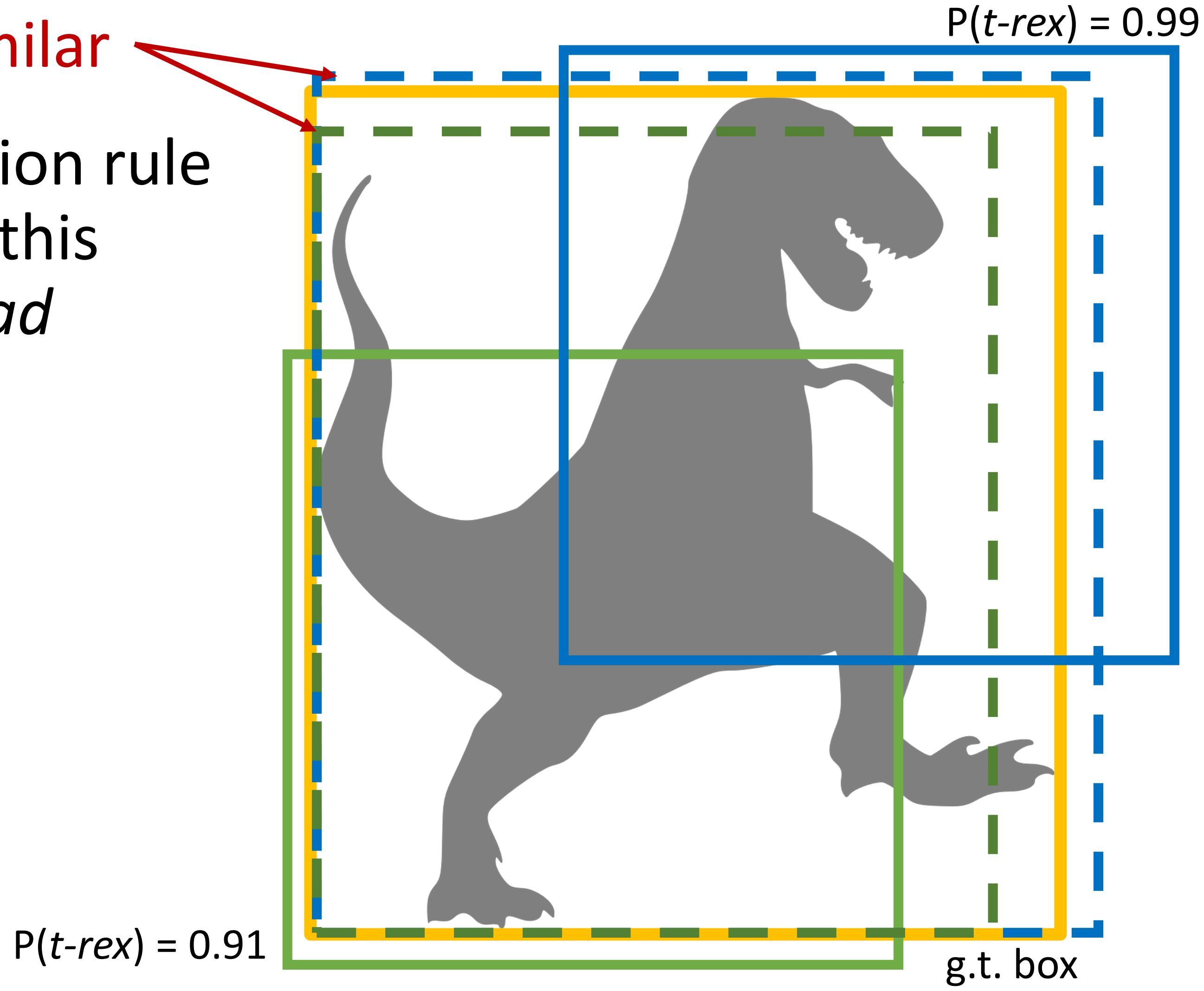
Challenge 3: Duplicate detections



Challenge 3: Duplicate detections

Predictions are similar

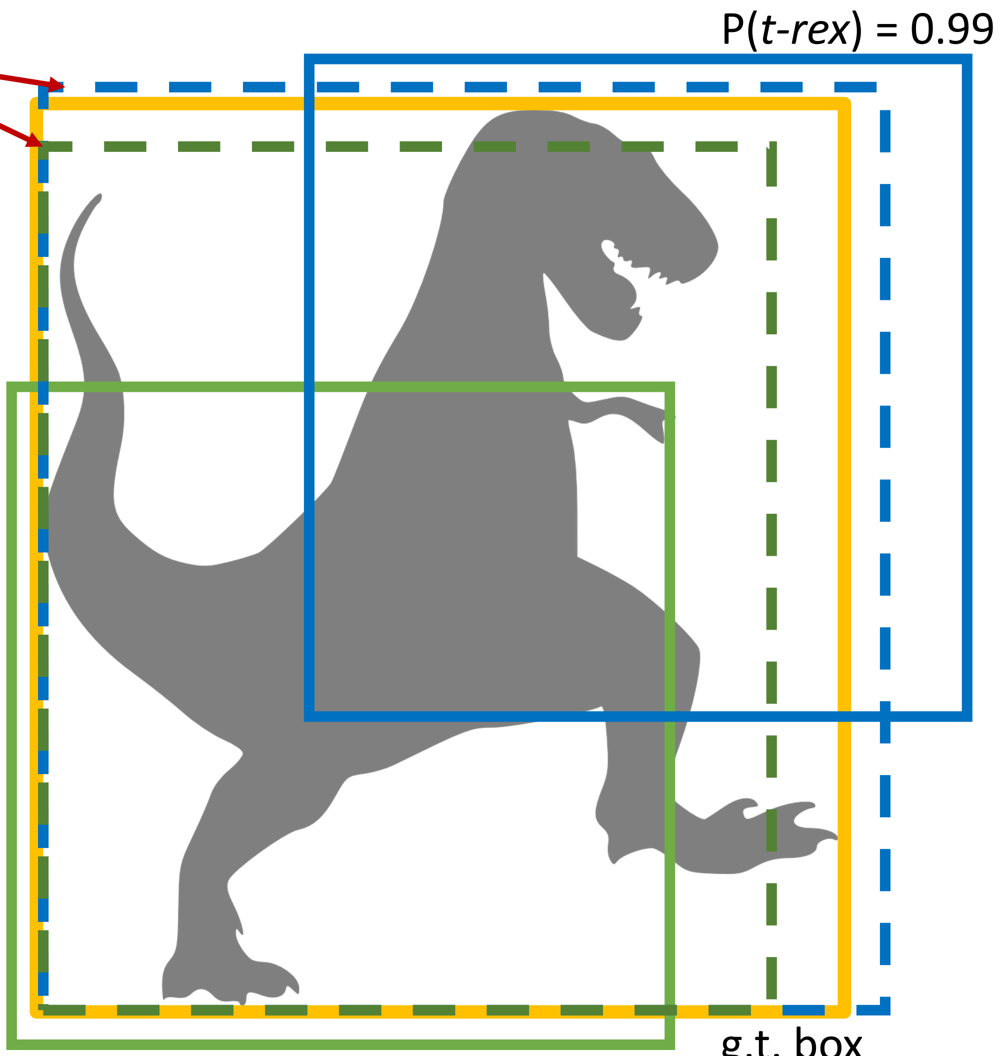
- Proxy classification rule asks for exactly this (labels are *spread* spatially)



Challenge 3: Duplicate detections

Predictions are similar

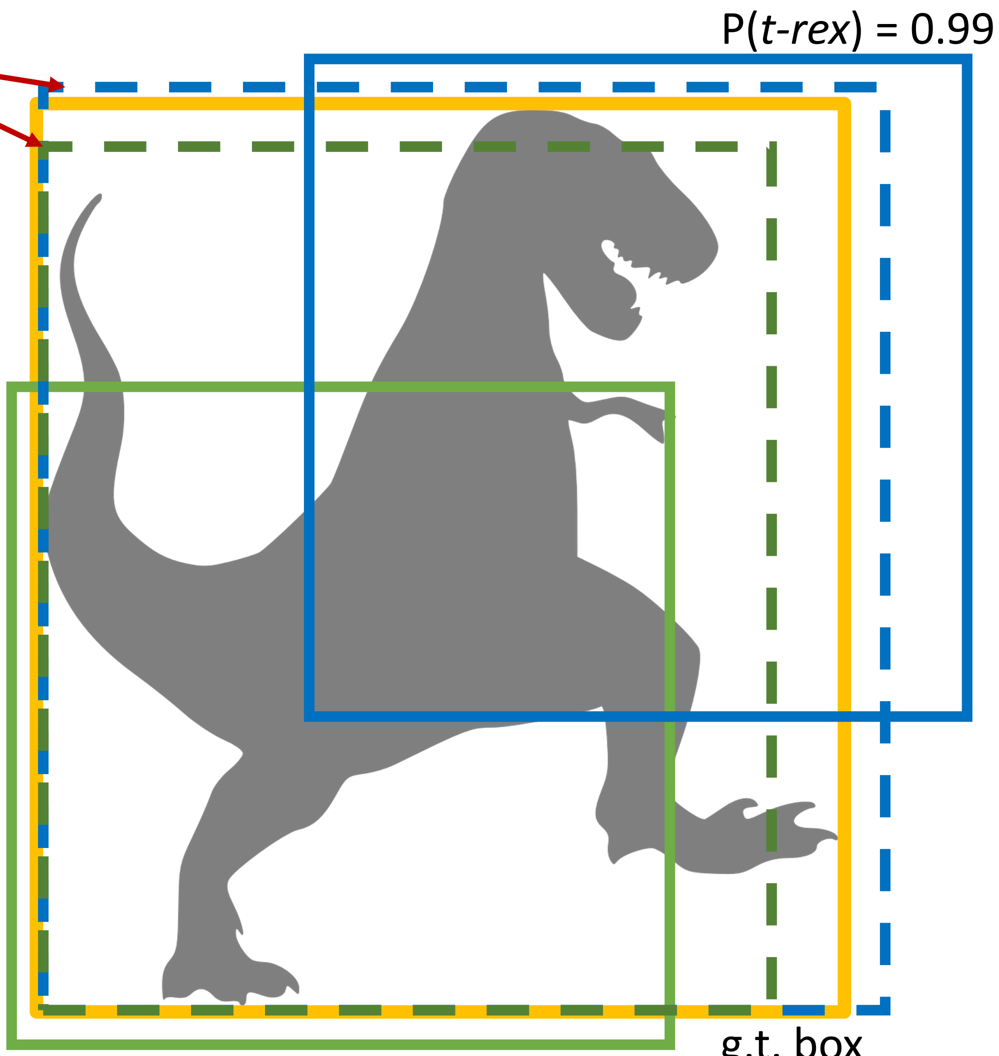
- Proxy classification rule asks for exactly this (labels are *spread* spatially)
- Could define a different proxy classification rule that does not spread?



Challenge 3: Duplicate detections

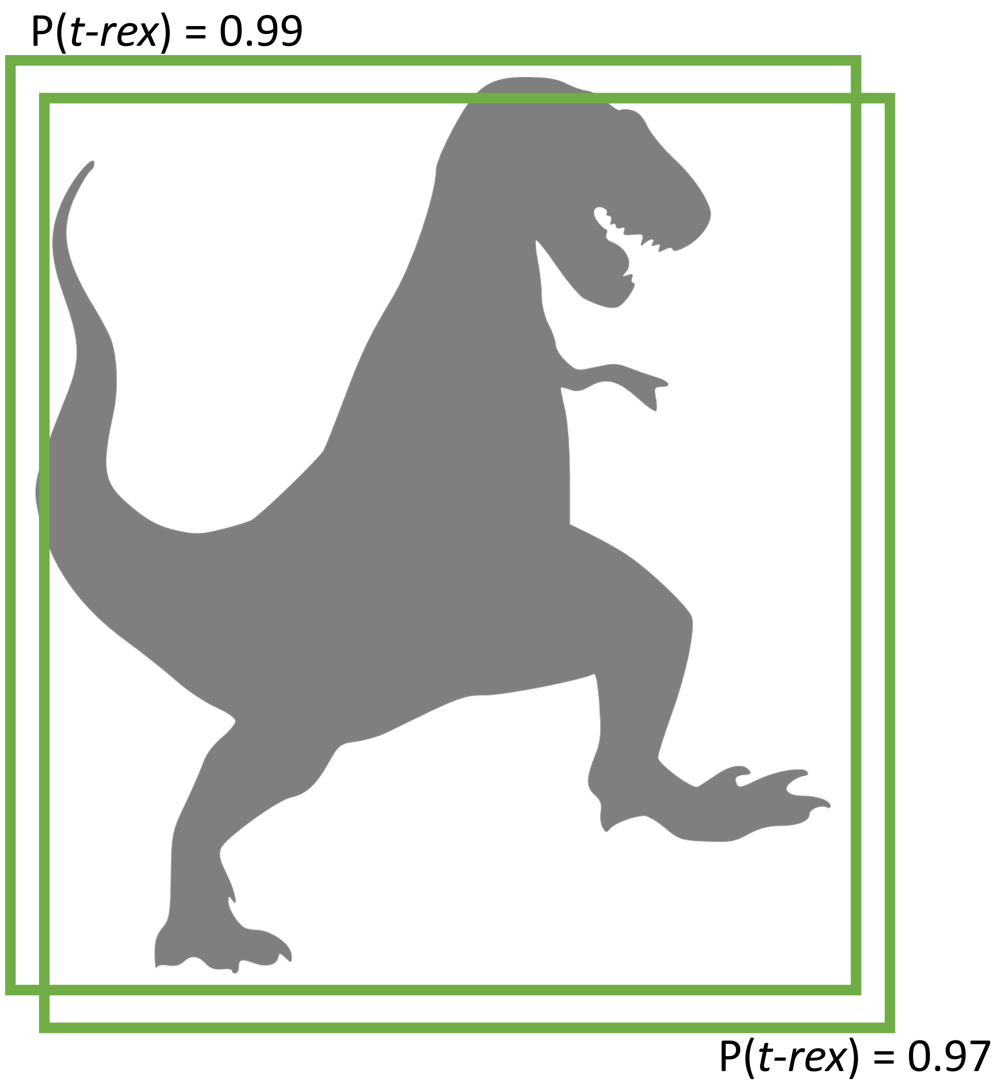
Predictions are similar

- Proxy classification rule asks for exactly this (labels are *spread* spatially)
- Could define a different proxy classification rule that does not spread?
  - Empirical works poorly (likely due to symmetry)



Challenge 3: Duplicate detections

Standard solution

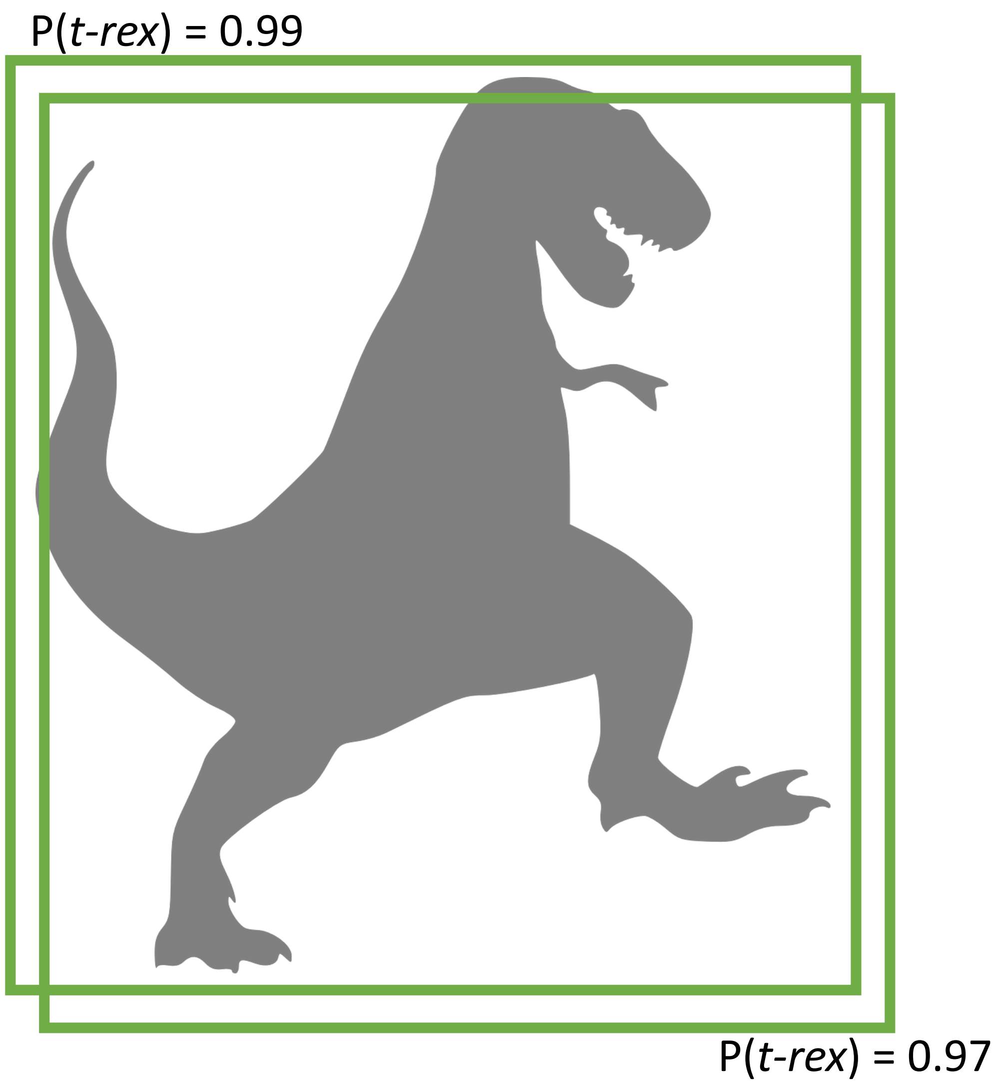


Strategy

- Don't try to learn a sharp classification rule

Challenge 3: Duplicate detections

Standard solution



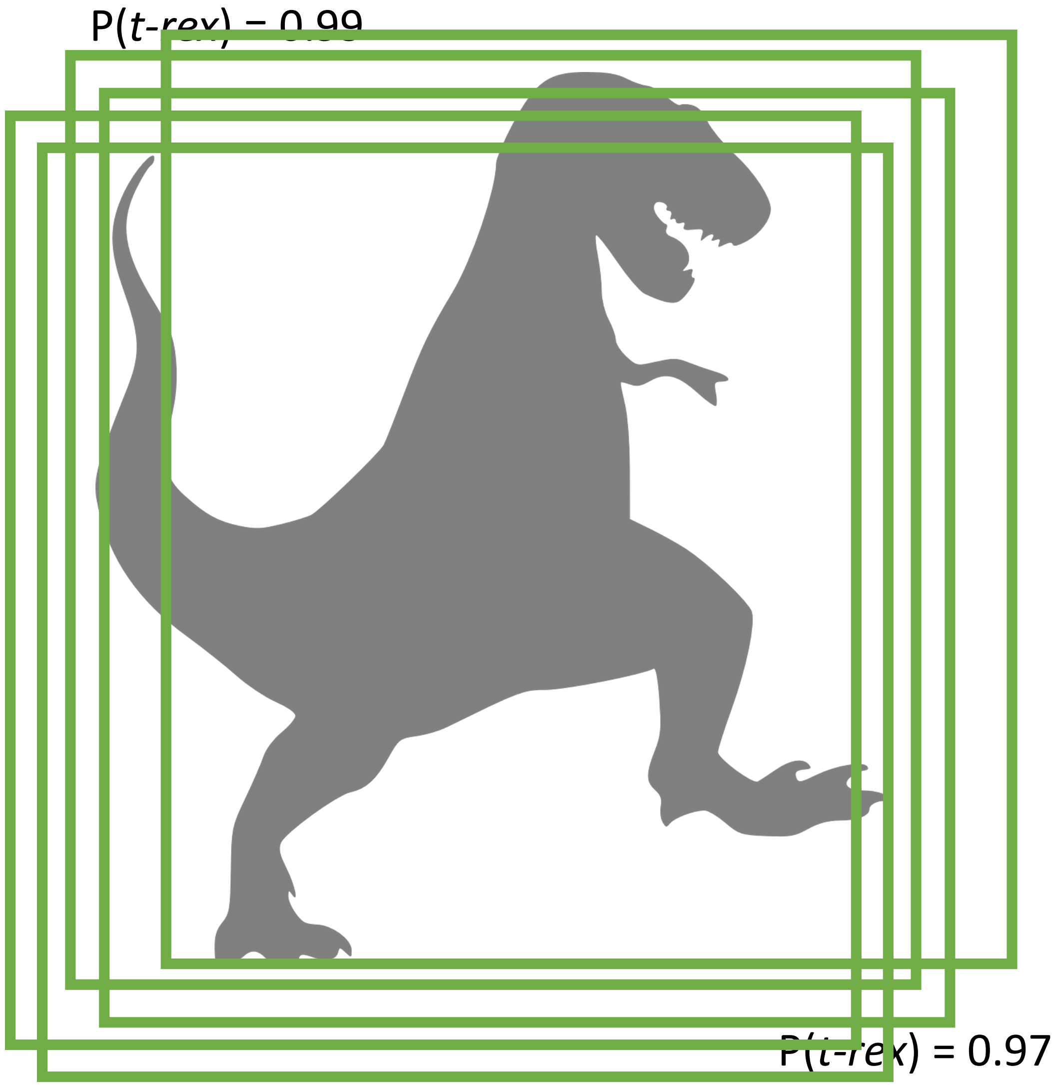
Strategy

- Don't try to learn a sharp classification rule

- Instead, clean up duplicates in a post-processing step

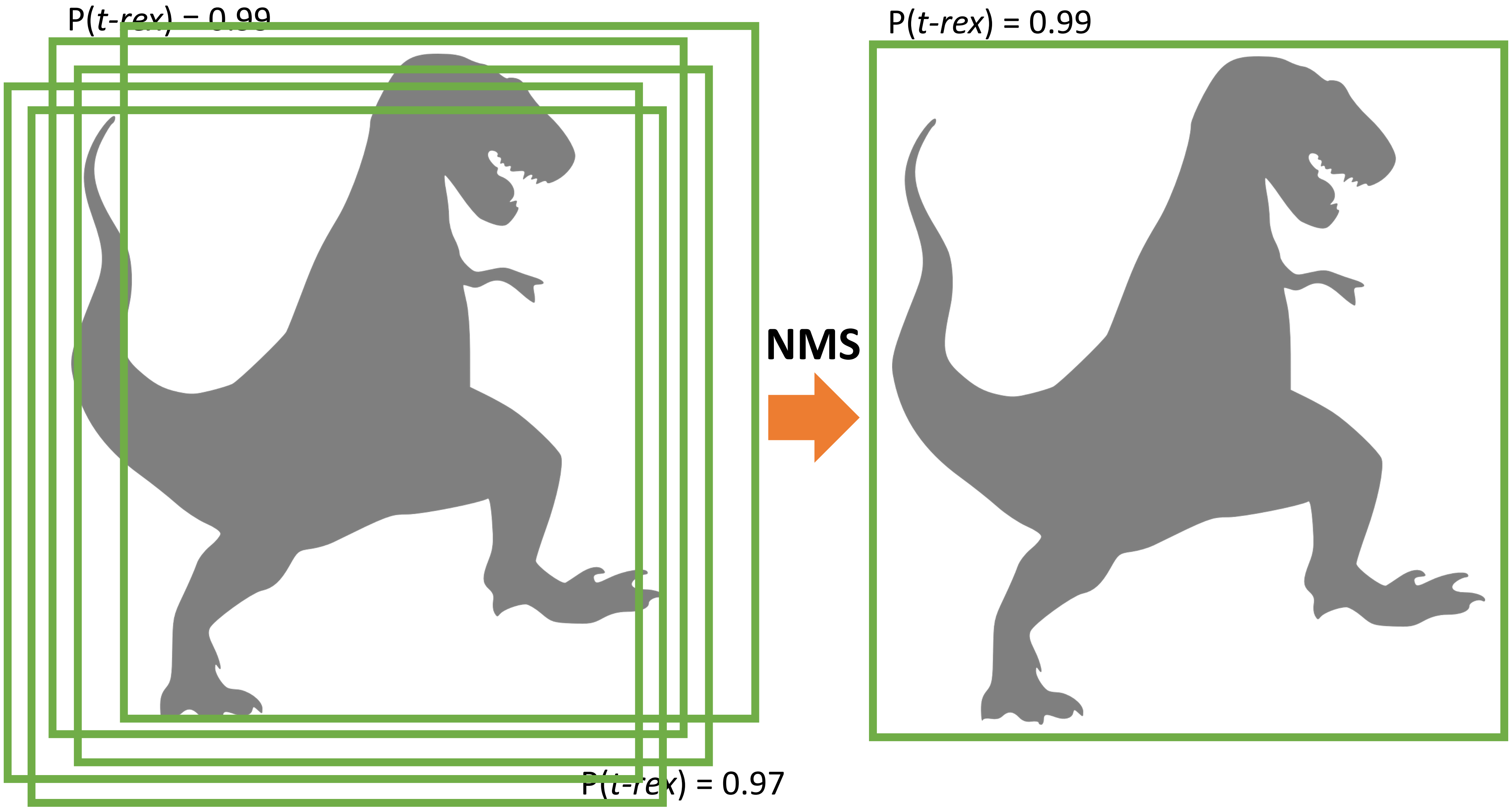
Challenge 3: Duplicate detections

Standard solution



Challenge 3: Duplicate detections

Standard solution



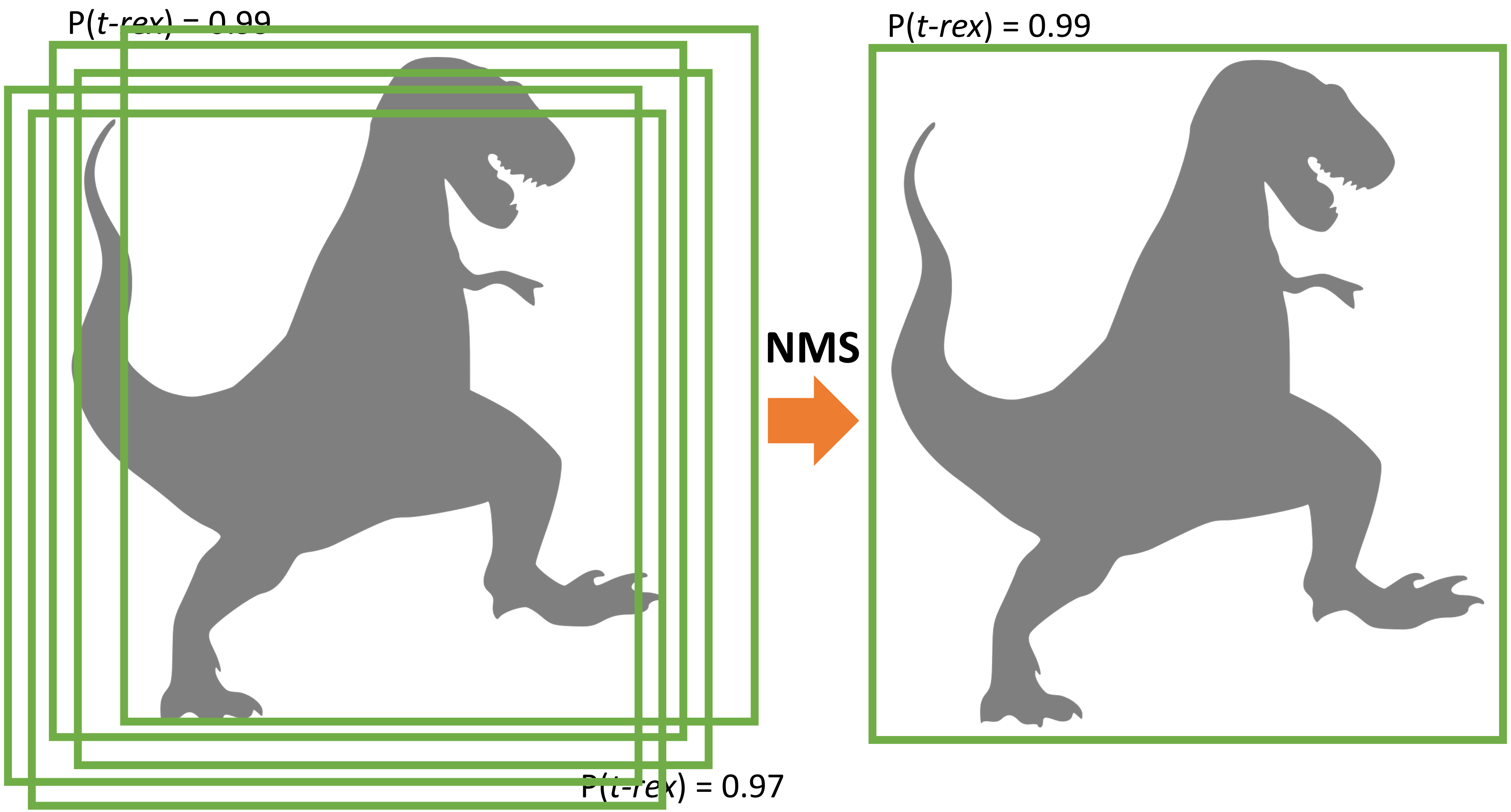
Found. concept: Non-maximum suppression

- General concept beyond detection: suppress values that are not maximal

- 1D signal example:

```
NMS([0.0, 0.0, 0.1, 0.4, 0.9, 0.5, 0.1, 0.01, 0.0]) →
    [0.0, 0.0, 0.0, 0.0, 0.9, 0.0, 0.0, 0.00, 0.0]
```

Found. concept: Non-maximum suppression



Found. concept: Non-maximum suppression

- Many possible algorithms for boxes

- Most common: greedy selection
    1. Sort detections by score
    2. Keep the highest scoring unsuppressed box B
    3. Find all lower scoring boxes B’ with IoU(B, B’) > nms_iou_threshold
    4. Suppress these boxes B’
    5. Go to step 2

- Can also view this as a clustering problem

Challenge 4: Foreground-background imbalance

- Before quantization
    - Infinitely imbalanced! (An intrinsic problem)

Challenge 4: Foreground-background imbalance

- Before quantization
  - Infinitely imbalanced! (An intrinsic problem)
- After quantization
  - Typically, ~100k classification decisions per image (better than infinite!)
  - But only 0.01 to 0.1% are assigned foreground labels (imbalanced!)

Challenge 4: Foreground-background imbalance

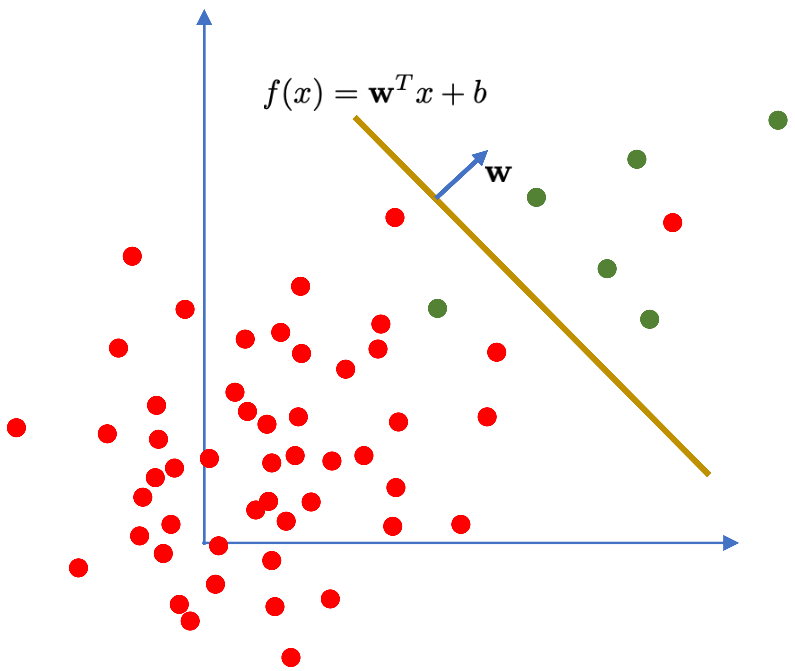
- Before quantization
  - Infinitely imbalanced! (An intrinsic problem)

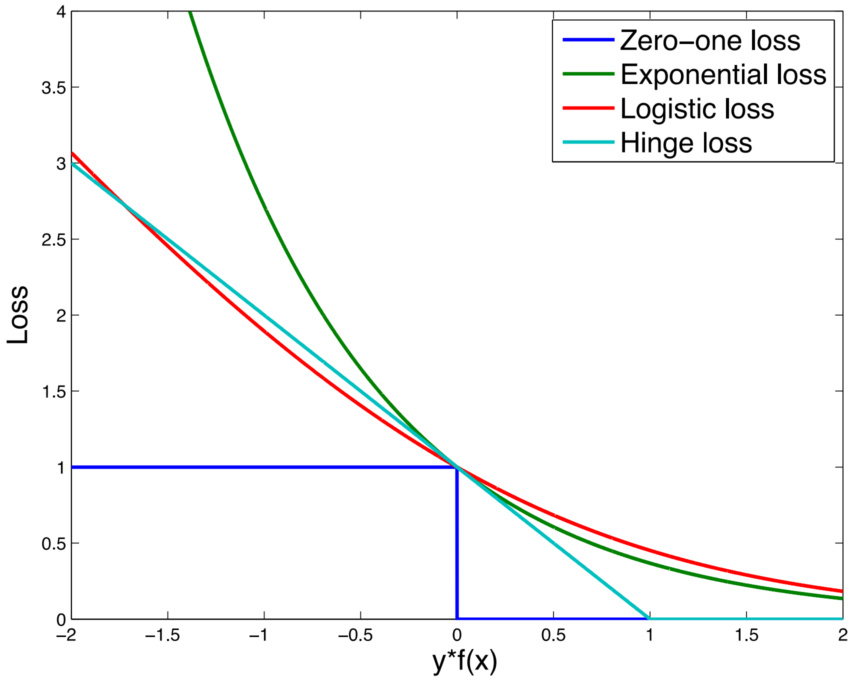
- After quantization
  - Typically, ~100k classification decisions per image (better than infinite!)
  - But only 0.01 to 0.1% are assigned foreground labels (imbalanced!)

- Two issues
  - Learning from imbalanced data is difficult (open research area)
    (e.g., ignore the minority class → ~100% classification accuracy*)
  - Processing speed

*This is why we use AP!

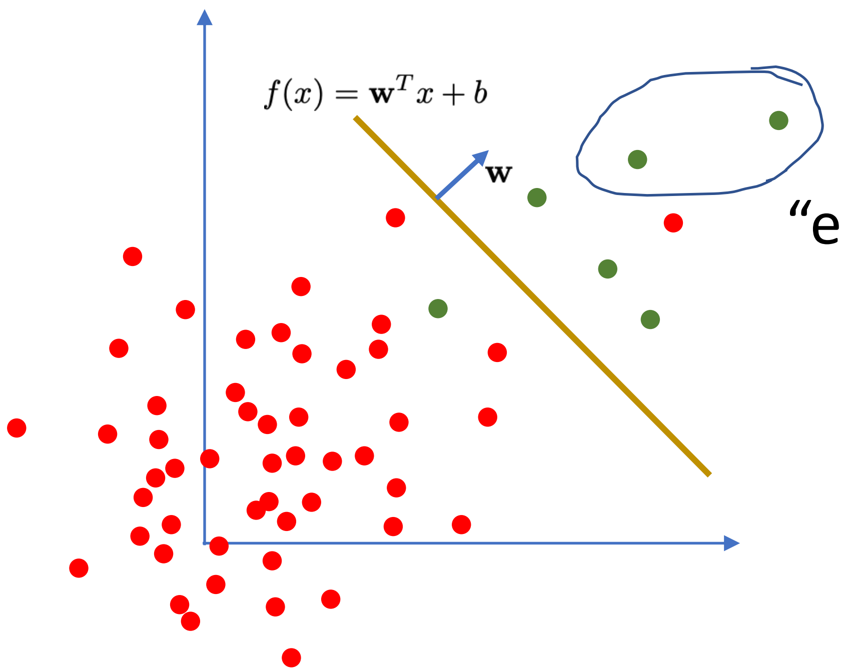
Found. concept: Loss functions; easy/hard data



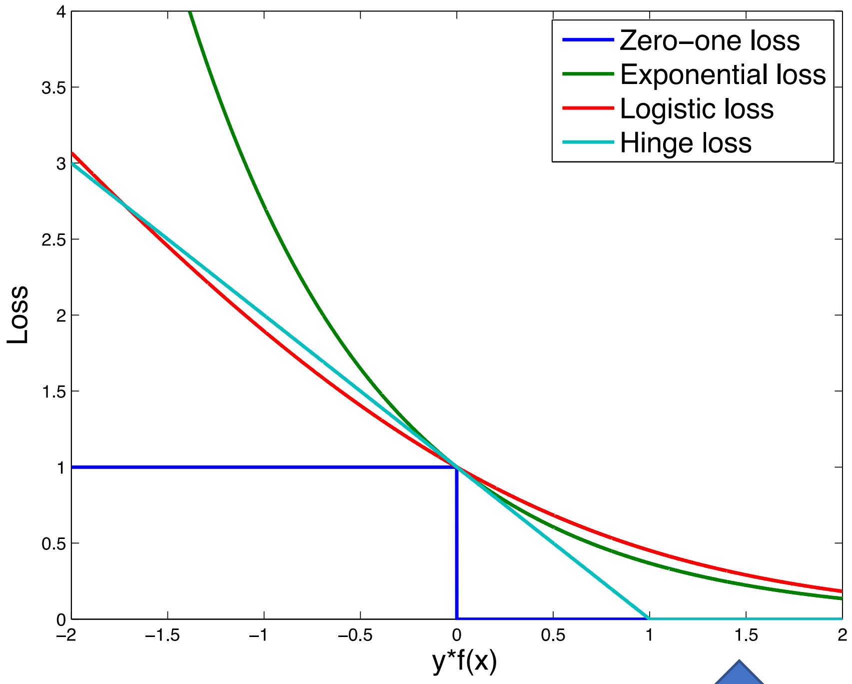


(margin)

Found. concept: Loss functions; easy/hard data



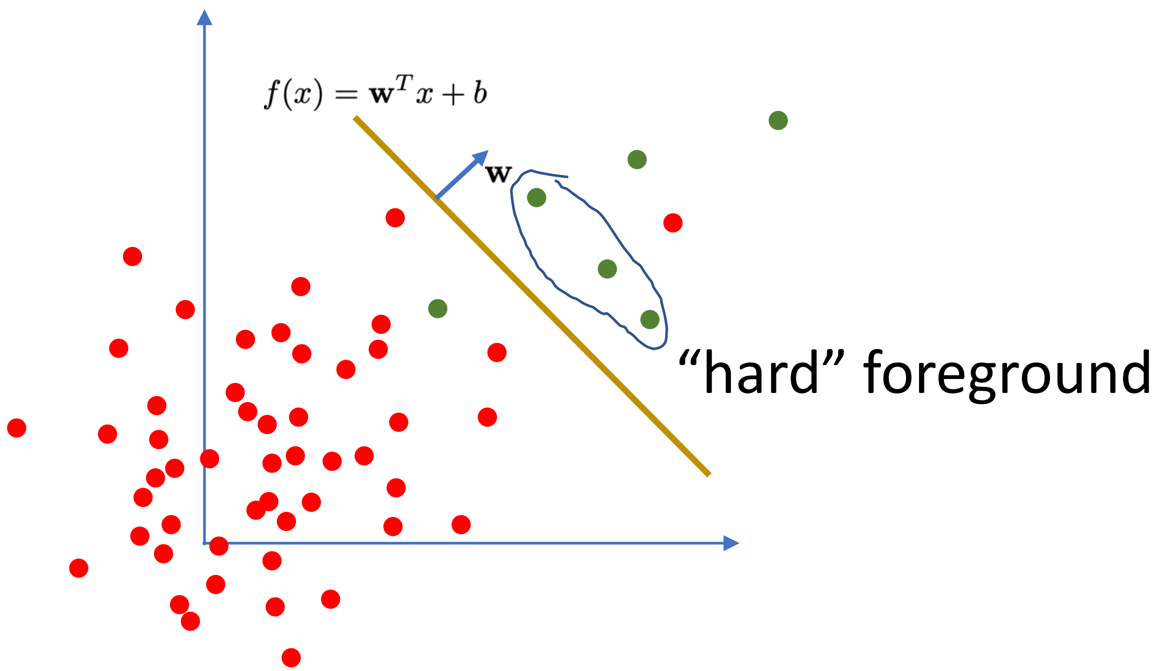
“easy” foreground

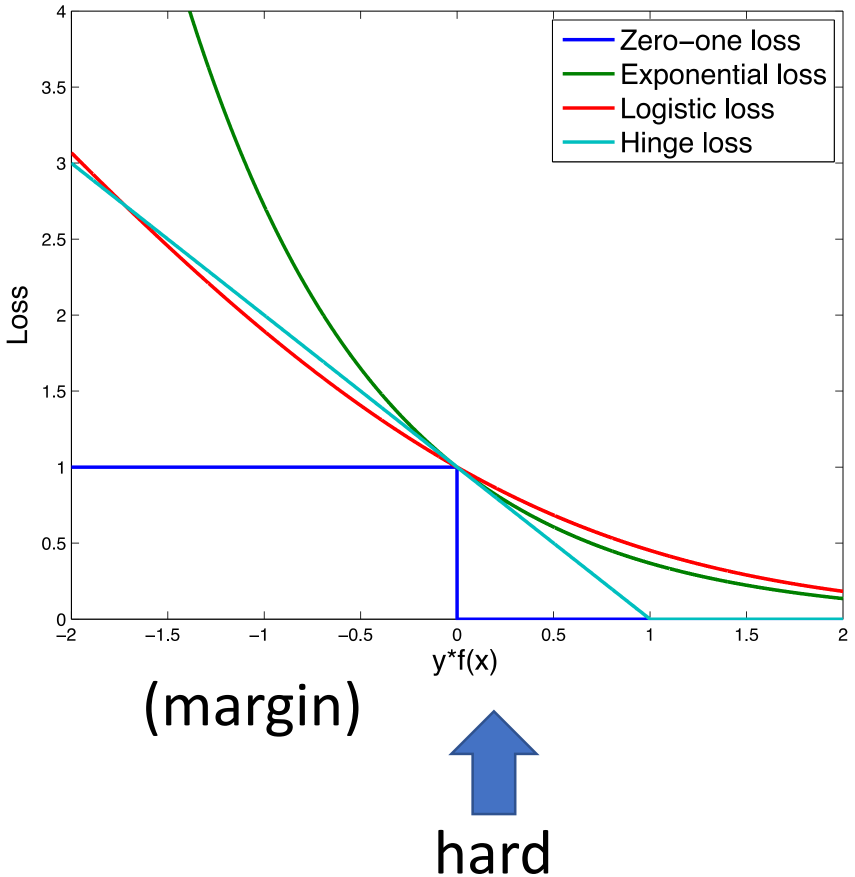


(margin)

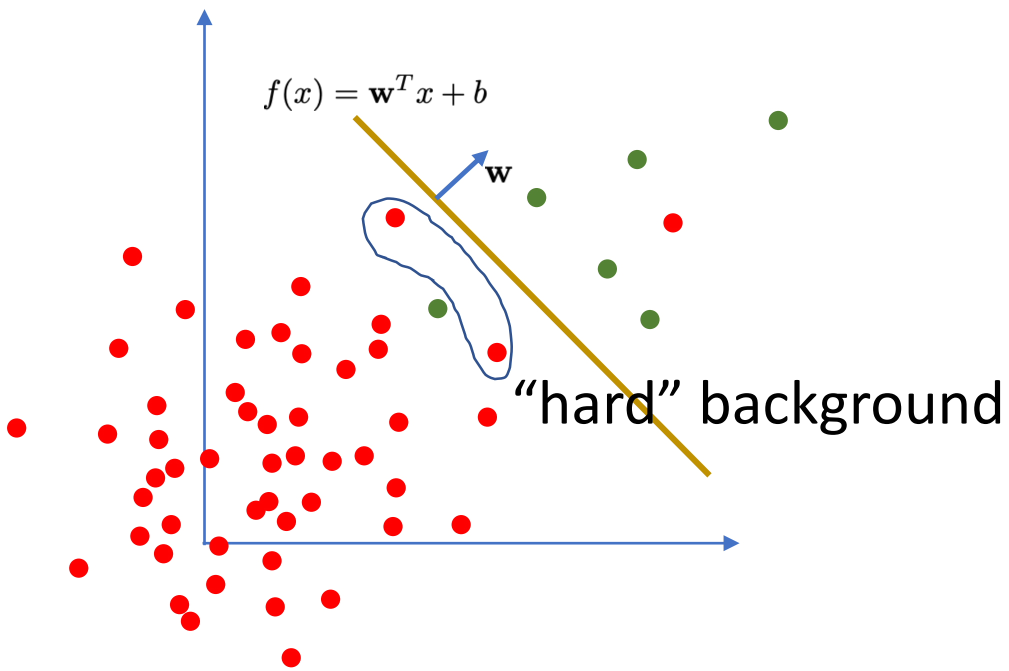
easy

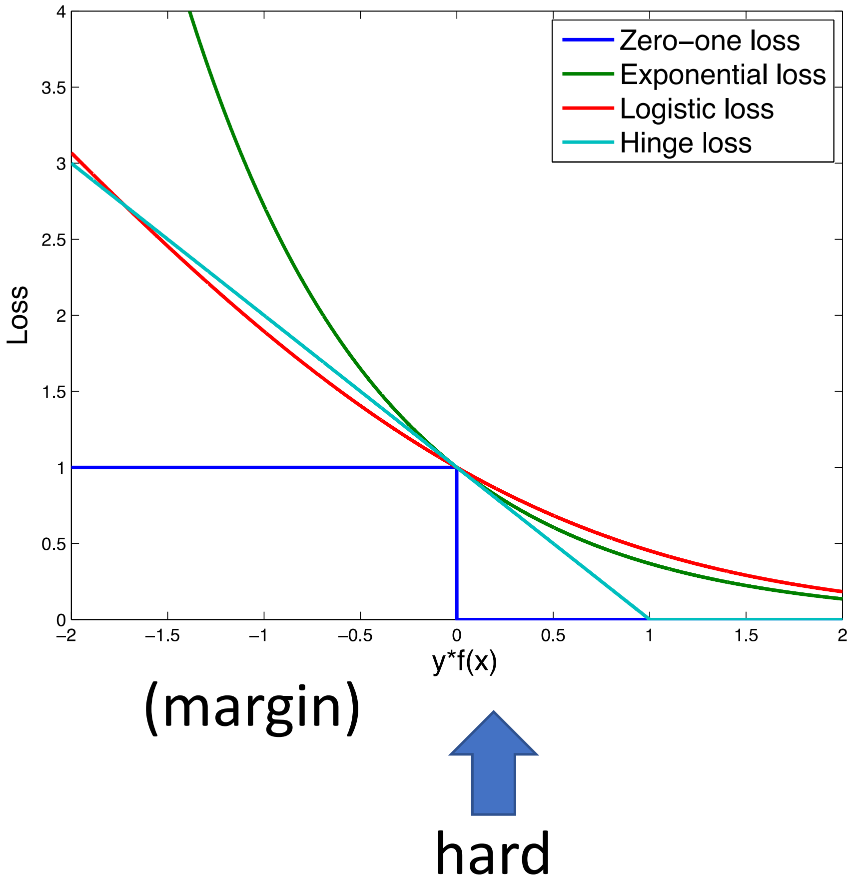
Found. concept: Loss functions; easy/hard data



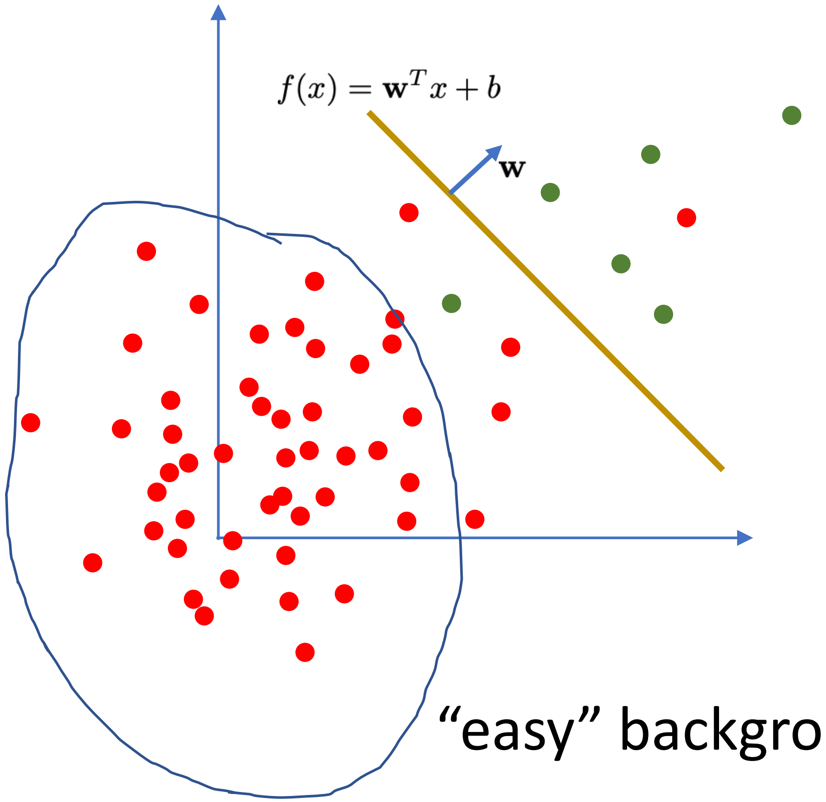


Found. concept: Loss functions; easy/hard data

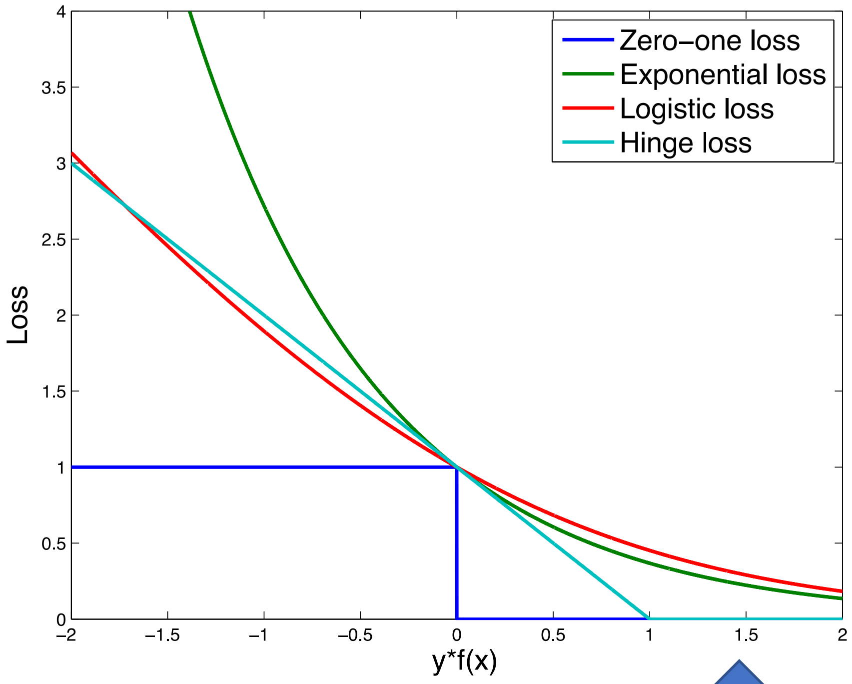




Found. concept: Loss functions; easy/hard data



"easy" background – there are a *lot* of these!

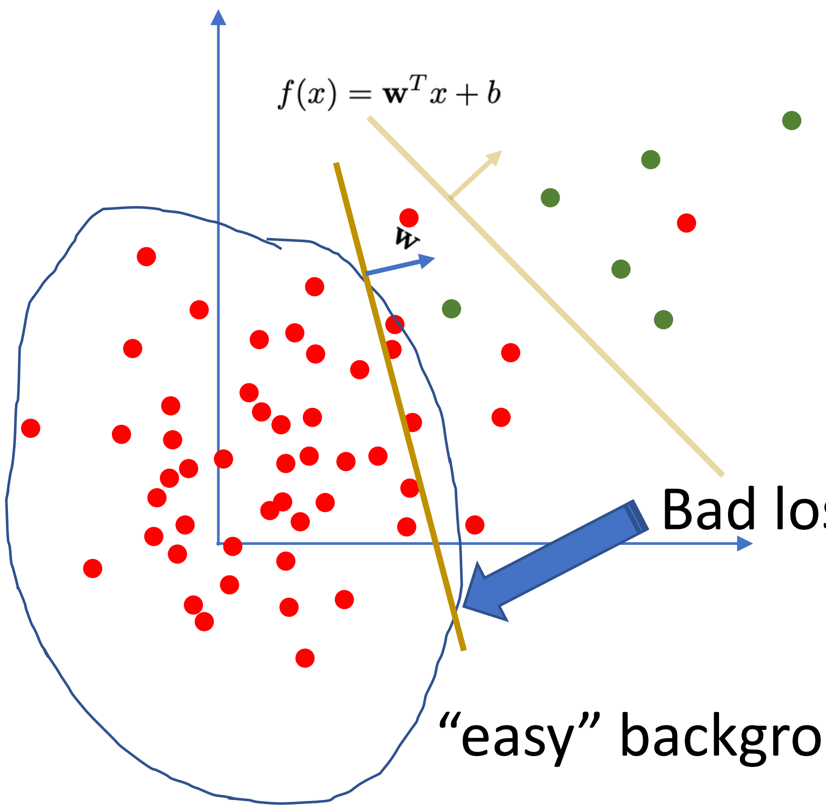


(margin)

easy

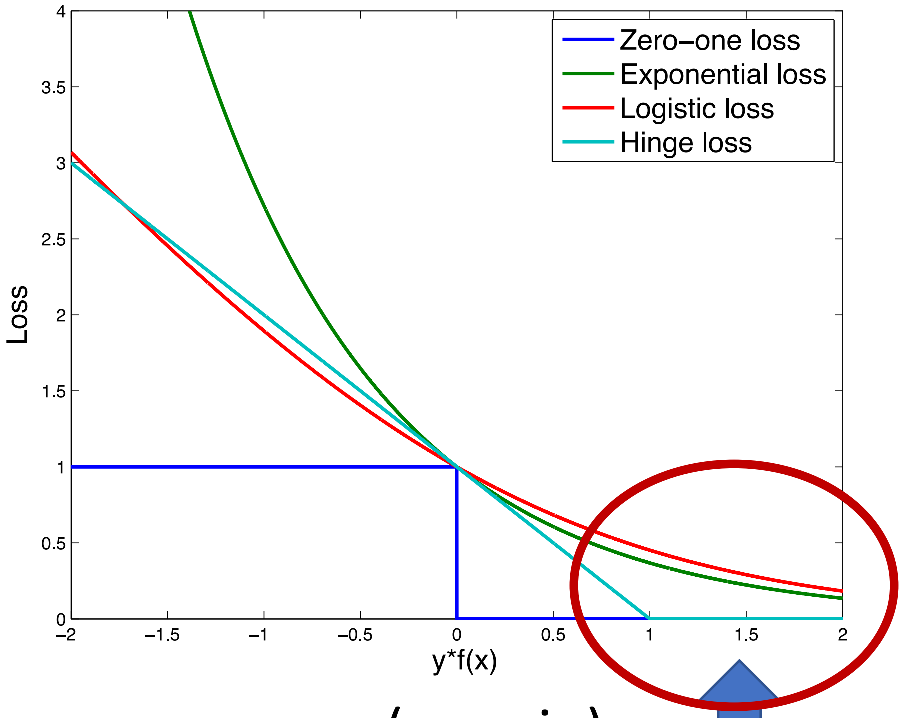
Found. concept: Loss functions; easy/hard data

Common loss functions (CE / logistic) are sensitive to lots of easy background



Bad loss function, bad AP

"easy" background – there are a *lot* of these!

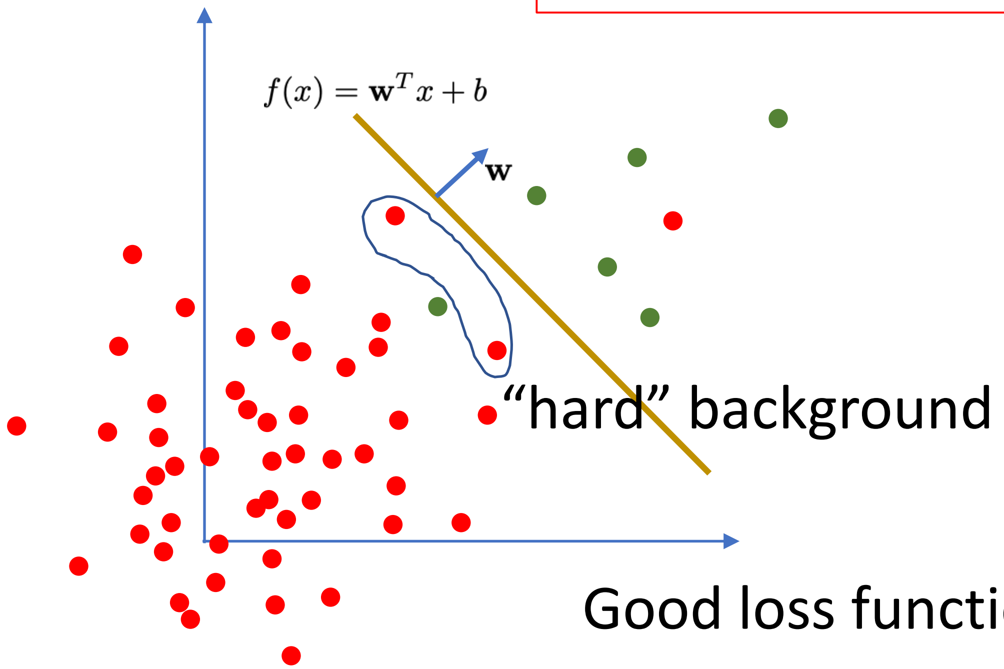


(margin)

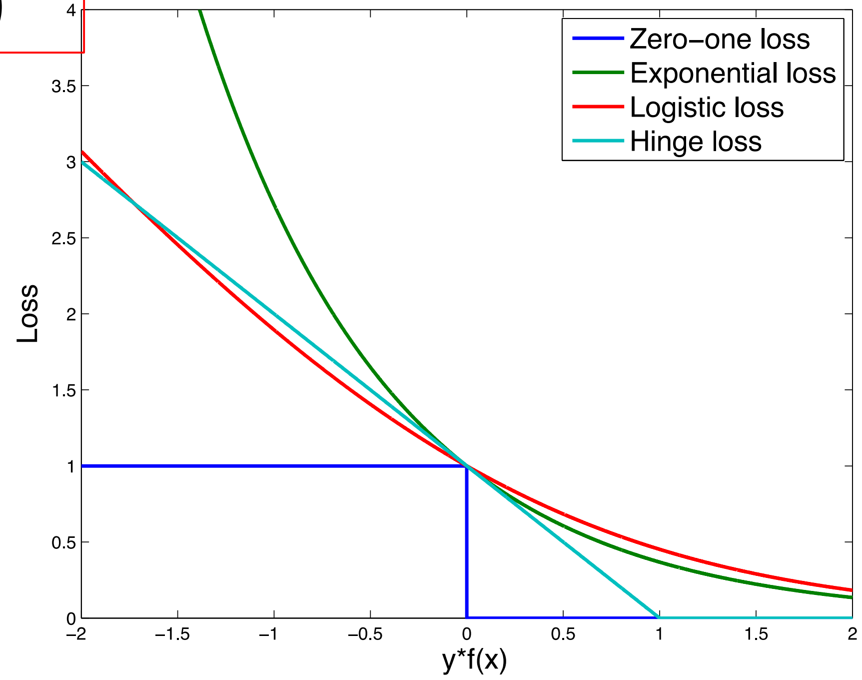
easy

Found. concept: Loss functions; easy/hard data

A good loss function for detection pays more attention to hard examples
(often called "hard example mining")



Good loss function, good AP

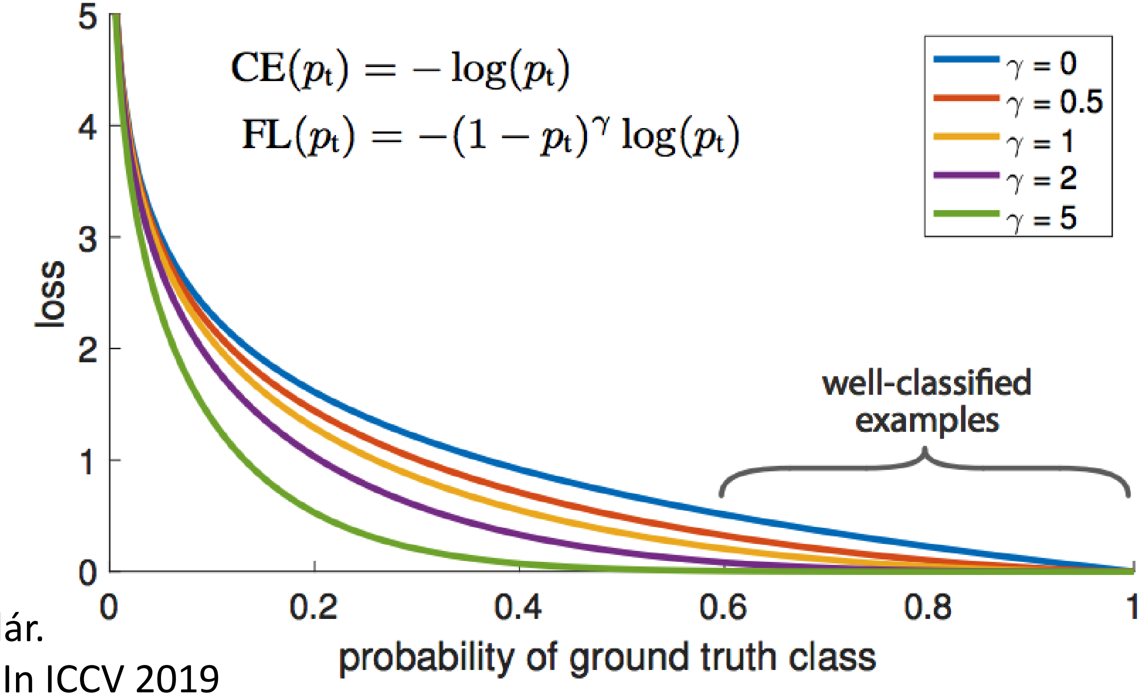


(margin)

Found. concept: Loss functions; easy/hard data

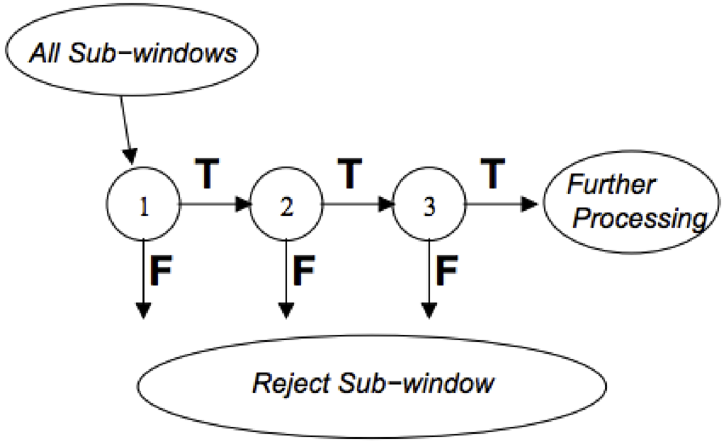
A good loss function for detection pays more attention to hard examples
(often called "hard example mining")

Example: focal loss



T-Y Lin, P Goyal, R Girshick, K He, P Dollár.
Focal Loss for Dense Object Detection. In ICCV 2019

Foundational concept: Cascade

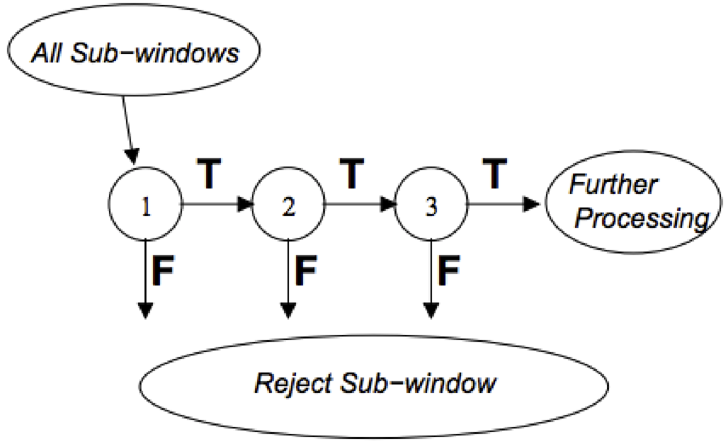


Classification cascade

[Viola & Jones 2001]

- Each classifier stage is inexpensive (implies fast, but weak)

Foundational concept: Cascade

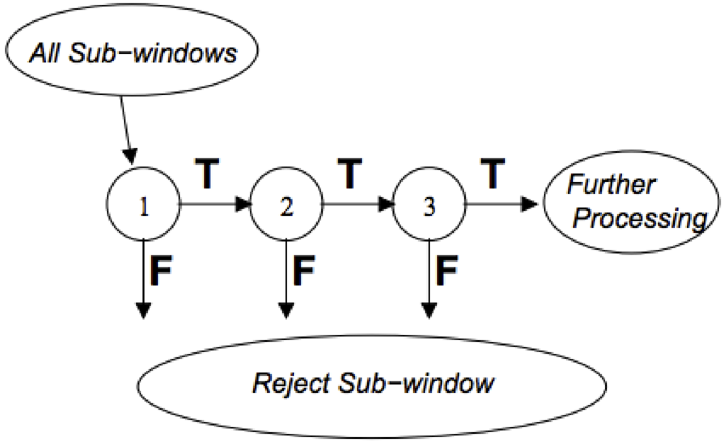


Classification cascade

[Viola & Jones 2001]

- Each classifier stage is inexpensive (implies fast, but weak)
- Early stages: tuned for high recall
  - Winnow down false positives, retain true positives
  - Gradually alleviates imbalance

Foundational concept: Cascade

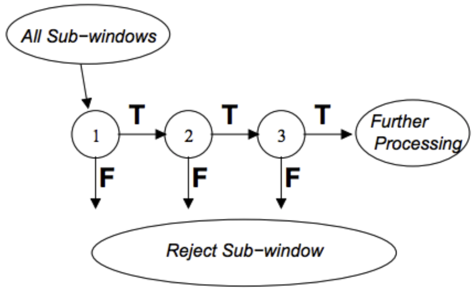


Classification cascade

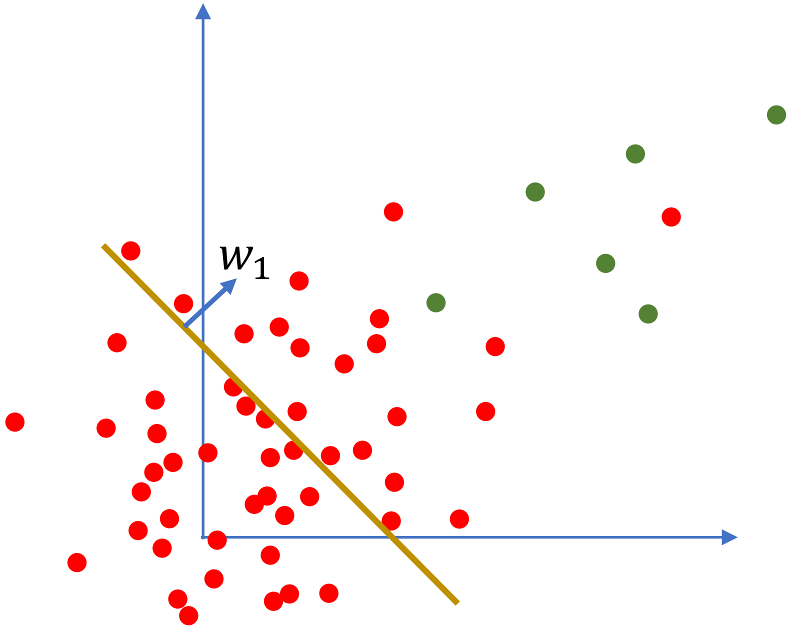
[Viola & Jones 2001]

- Each classifier stage is inexpensive (implies fast, but weak)
- Early stages: tuned for high recall
  - Winnow down false positives, retain true positives
  - Gradually alleviates imbalance
- Later stages see (more) balanced data

Foundational concept: Cascade

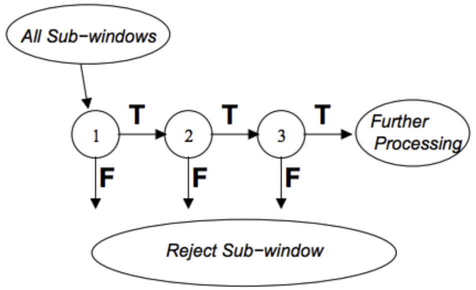


eliminate easy background step-by-step, leading to a …

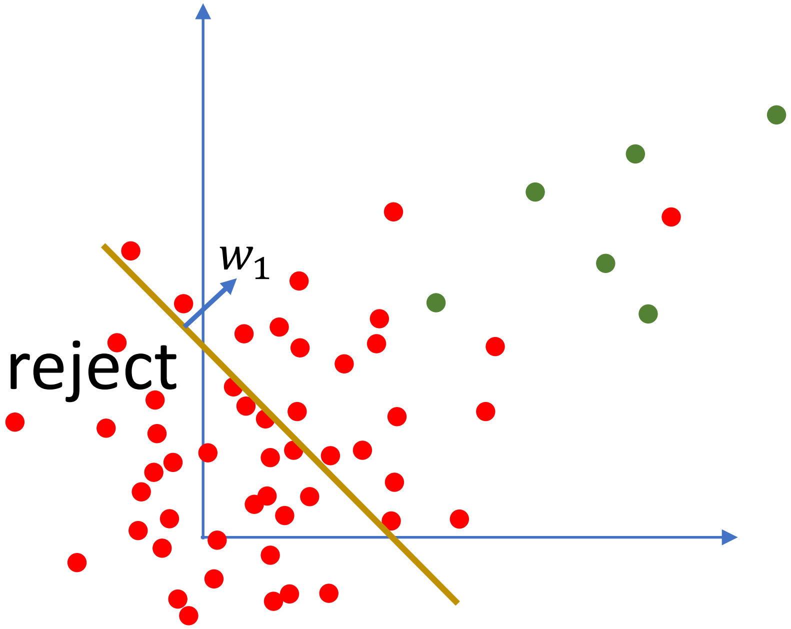


Stage 1

Foundational concept: Cascade

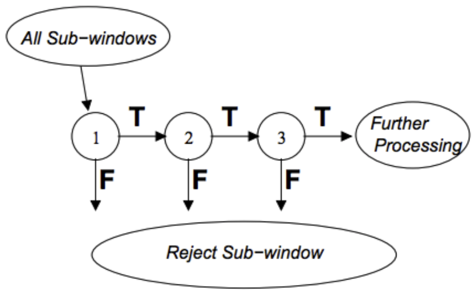


eliminate easy background step-by-step, leading to a …

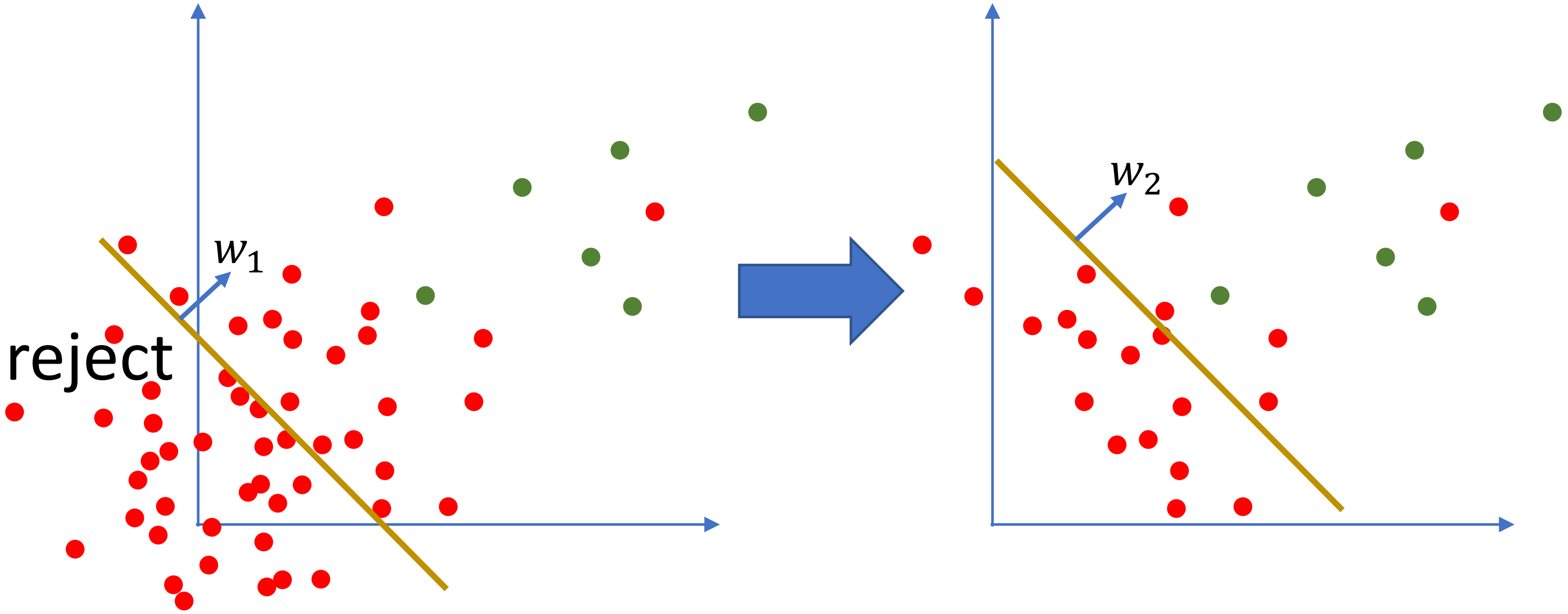


Stage 1

Foundational concept: Cascade

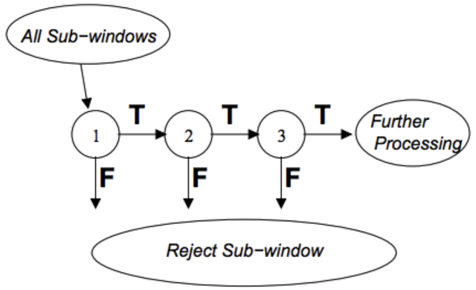


eliminate easy background step-by-step, leading to a …

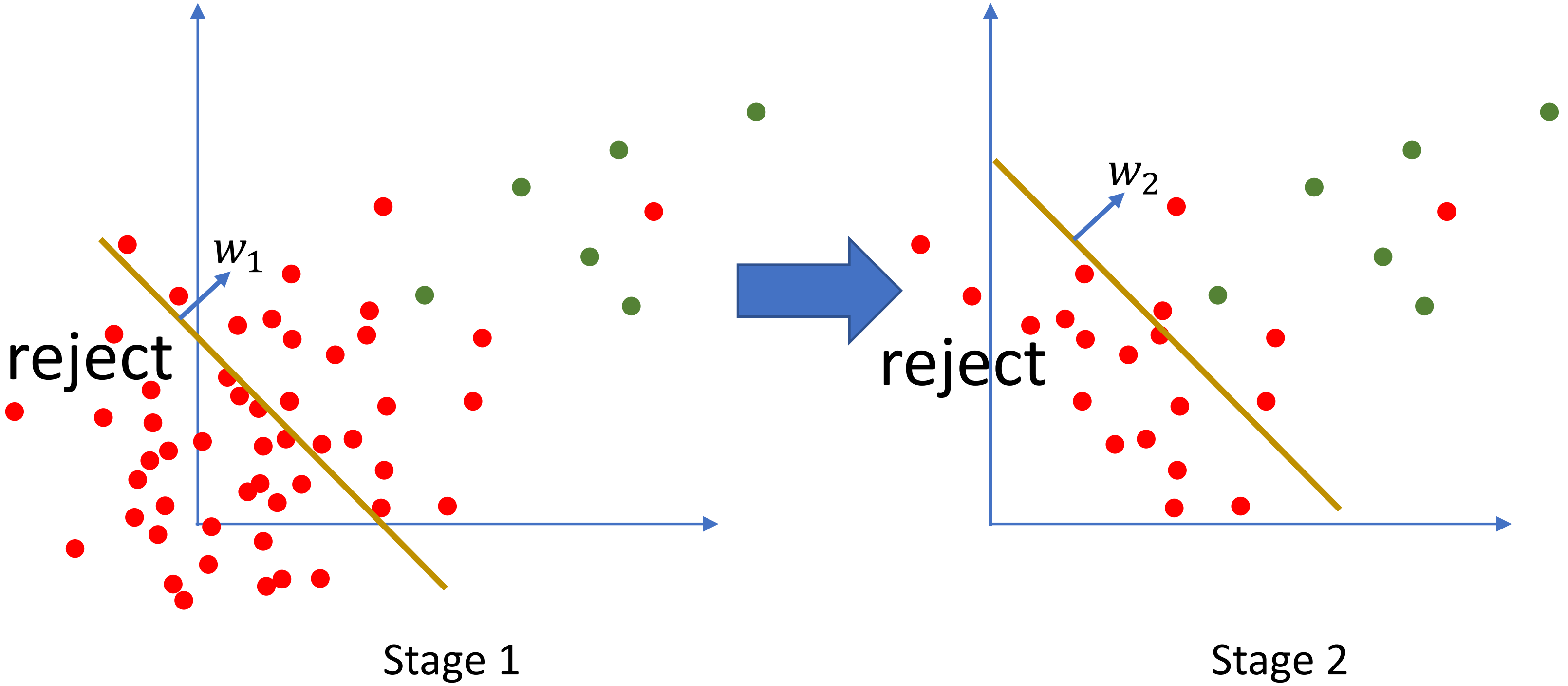


Stage 1 Stage 2

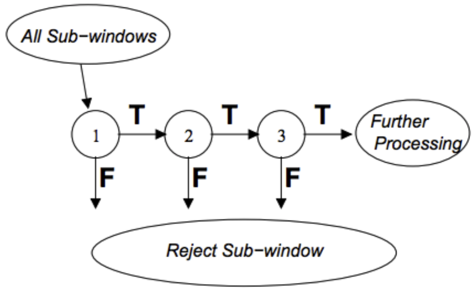
Foundational concept: Cascade



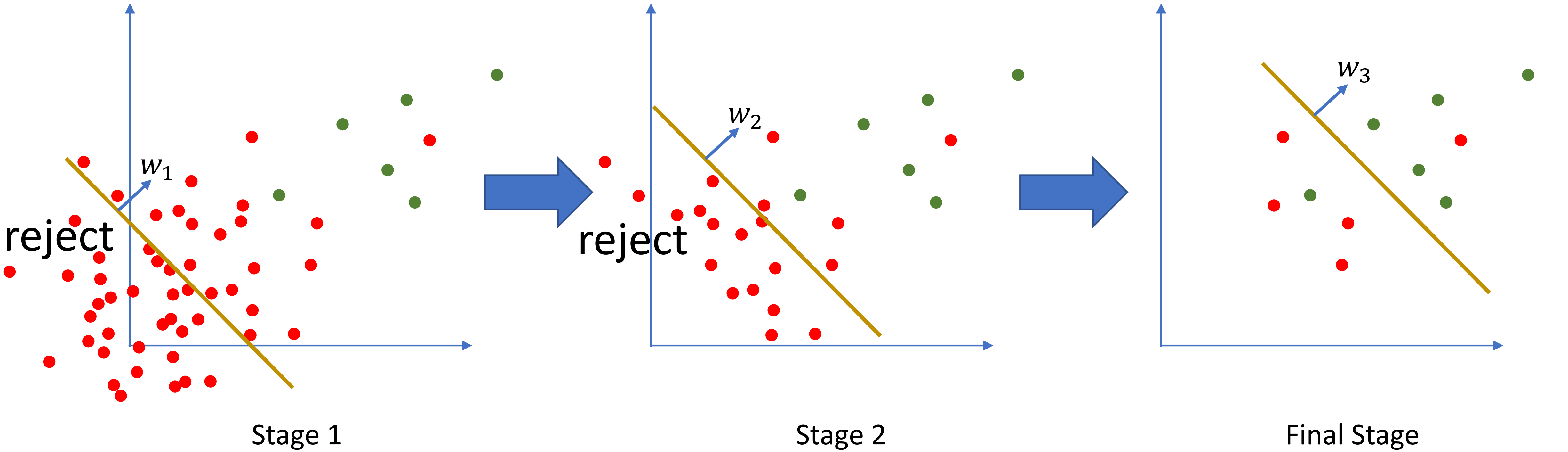
eliminate easy background step-by-step, leading to a …



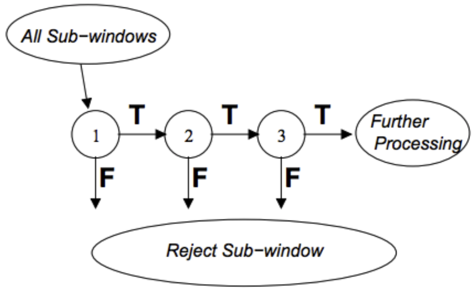
Foundational concept: Cascade



eliminate easy background step-by-step, leading to a … … more balanced problem



Foundational concept: Cascade



Examples in modern use, R-CNN family:

- R-CNN: selective search → convnet
- Fast R-CNN: selective search → Fast R-CNN head
- Faster R-CNN: RPN → Fast R-CNN head
- Cascade R-CNN: RPN → Fast R-CNN head 1 → … → Fast R-CNN head *N*

Recap of today's lecture

- Object detection is a challenging input → output mapping
  - Output space: all finite subsets of all infinitely many possible boxes + category labels
- Abstract → mathematical → computational
  - The computation ML problem is a proxy, introducing assumptions and approximations
- The modeling choice has consequences that introduce new problems
  - *Box quantization, label assignment, redundant outputs, fg-bg imbalance*
- Most research on object detection focuses on these new problems
  - If we change the ML-modeling, do we get different, more tractable problems?