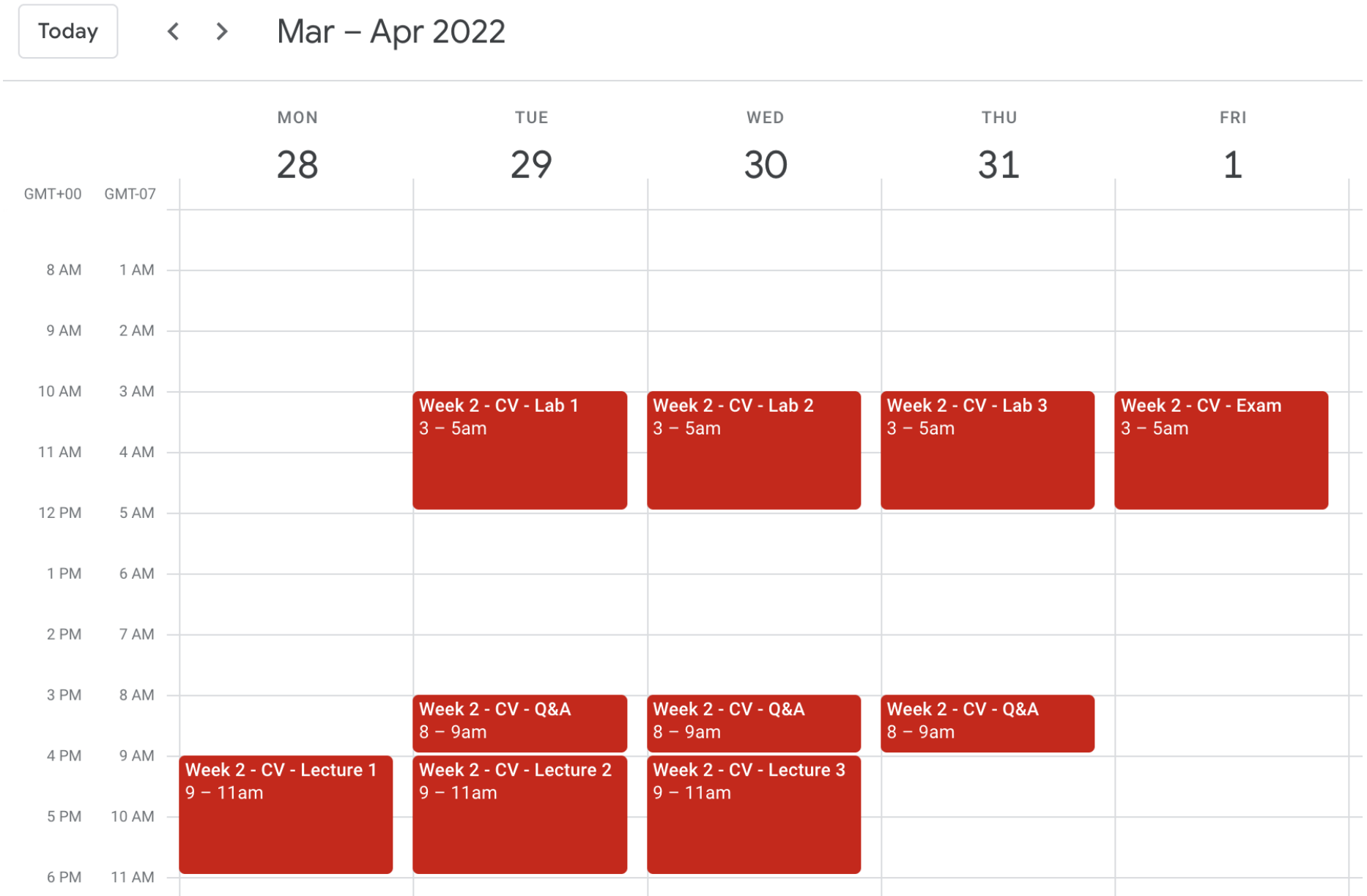
In my natural habitat, climbing mountains not far from home

Hello, nice to meet you! A bit about me, Ross Girshick (rbg)

- Undergrad in CS
- Worked for 3 years as a software engineer
- Grad school for PhD in Computer Vision
- Post-doc in Computer Vision
- Research Scientist at Microsoft Research (1 year)
- Research Scientist at FAIR (6 years)
- Starting working on object detection in winter 2008 and never stopped ☺

Schedule


Today
Mar – Apr 2022
MON 28
TUE 29
WED 30
THU 31
FRI 1
GMT+00
GMT-07
8 AM
1 AM
9 AM
2 AM
10 AM
3 AM
11 AM
4 AM
12 PM
5 AM
1 PM
6 AM
2 PM
7 AM
3 PM
8 AM
4 PM
9 AM
5 PM
10 AM
6 PM
11 AM
Week 2 - CV - Lab 1
3 – 5am
Week 2 - CV - Lab 2
3 – 5am
Week 2 - CV - Lab 3
3 – 5am
Week 2 - CV - Exam
3 – 5am
Week 2 - CV - Q&A
8 – 9am
Week 2 - CV - Q&A
8 – 9am
Week 2 - CV - Q&A
8 – 9am
Week 2 - CV - Lecture 1
9 – 11am
Week 2 - CV - Lecture 2
9 – 11am
Week 2 - CV - Lecture 3
9 – 11am

Setup

- Course information is on Campuswire
- During lectures, ask questions on Campuswire chat
- Labs: https://github.com/rbgirshick/ammi_cv_object_detection_labs
- Grading: Labs + Exam

Last week

- Image classification
- Empirical risk minimization
- Stochastic gradient descent
- Multilayer neural networks & backprop
- Convolutional neural networks
- Important network building blocks and design motifs

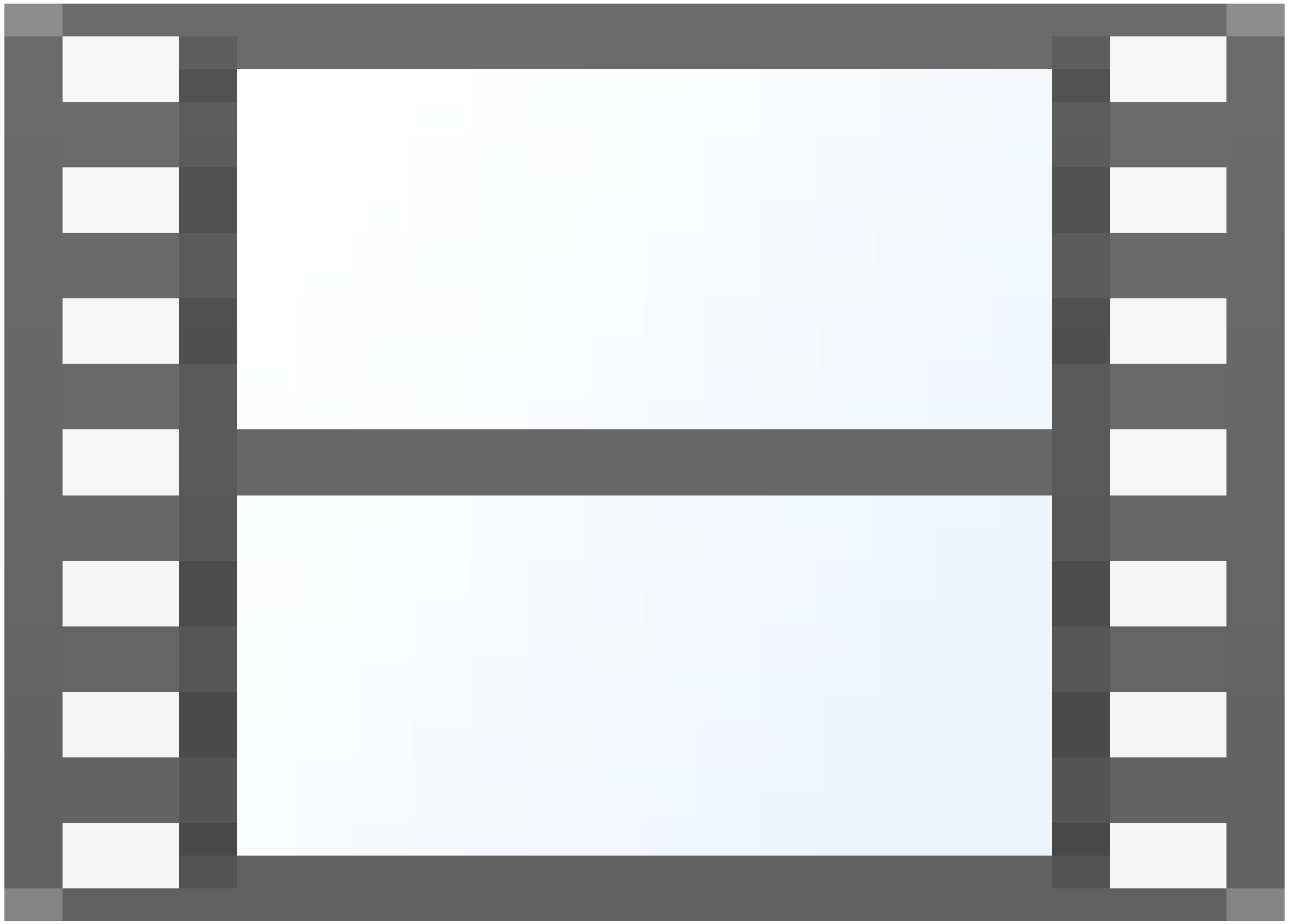
Overview of object detection lectures

- Lecture 1: Object detection – Tasks, evaluation & datasets
- Lecture 2: Object detection as an ML problem
- Lecture 3: R-CNN, Fast R-CNN, Faster R-CNN & Mask R-CNN

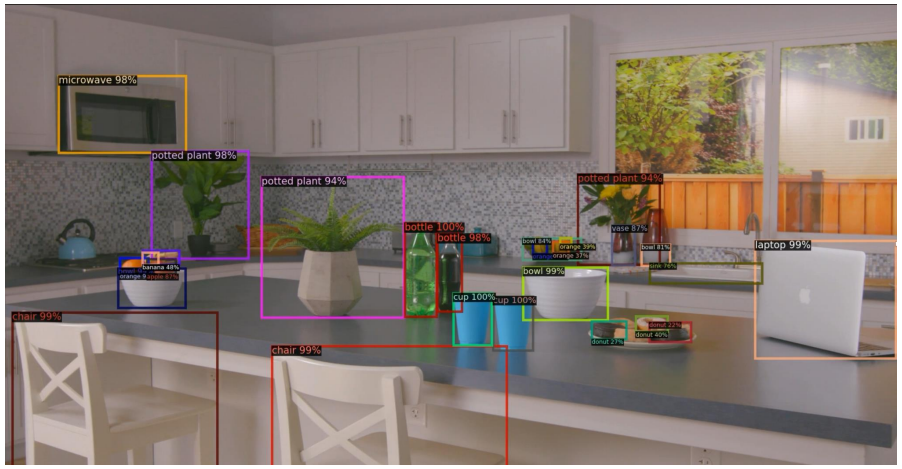
Lecture 1: Object detection – Tasks, evaluation & datasets

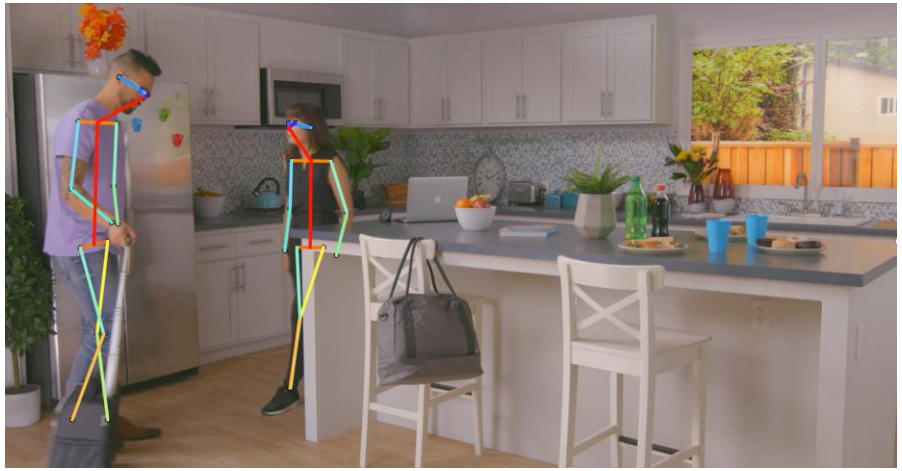
Ross Girshick (FAIR)

AMMI 2022, Computer Vision, Week 2

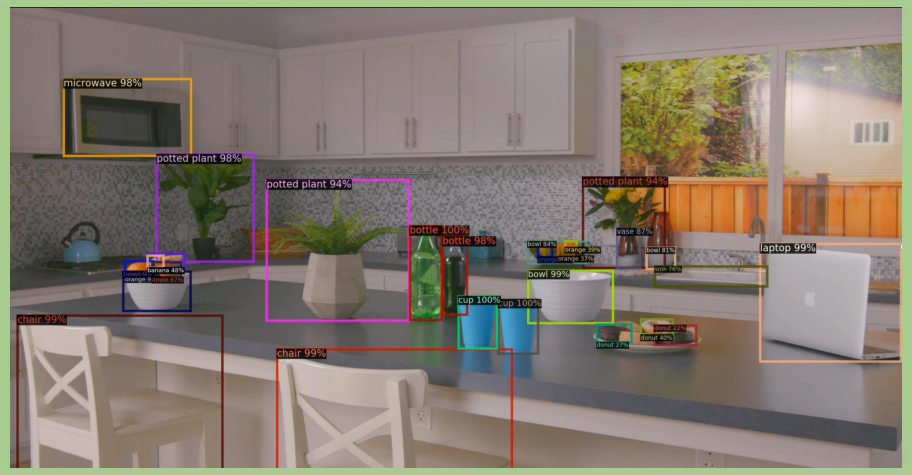


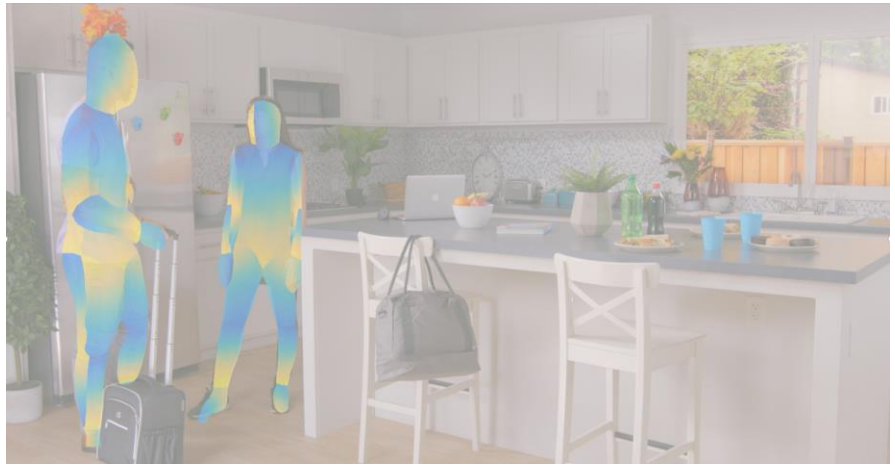
Many detection related tasks – boxes, masks, pose, meshes, …





We'll focus on classical object detection (boxes)





Learning objectives

Today:

1. What is the object detection task?
2. How is the task formalized?
3. How is the task evaluated?
4. Why is the task difficult?
5. What are the important datasets to know about?
6. Tips for reading research papers on object detection [stretch goal]

What's next? Tomorrow: *Fundamentals – Object Detection as a Machine Learning Problem*

What is the object detection task?

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

What is the object detection task?

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

Answer: What and Where?

Contrasted with image classification

Image classification

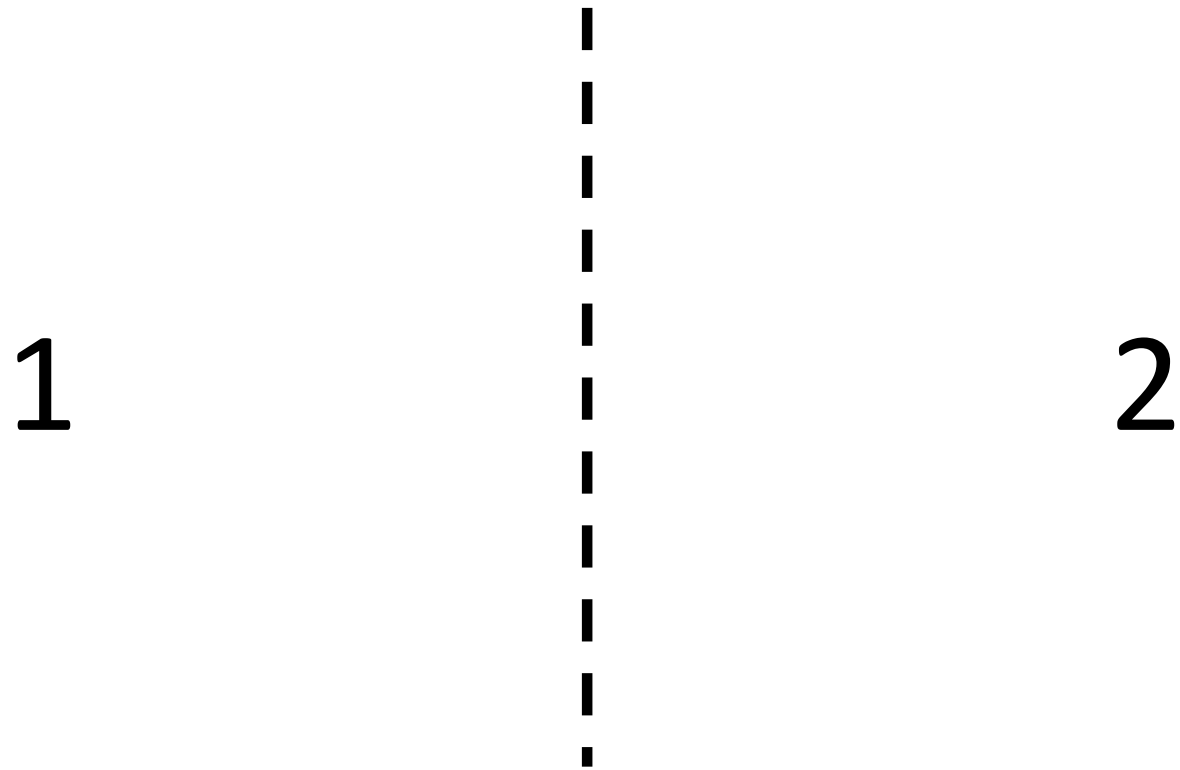
- What
- Fixed size output
- Obvious ML problem formulation
- Generic models & losses
- Easy to evaluate
- Simple training

Object detection

- What & where
- Variable size output
- Difficult to formulate as an ML problem (see: Lecture 2)
- Task-specific engineering
- Hard to evaluate
- Complex training

The two aspects of problem formulation

- Formalizing: What objects are in an image and where are they?



Aspect 1: Training

- Formalizing: <u>What</u> objects are in an image and <u>where</u> are they?

Training

- The <u>training data</u> available for fitting the ML model

2

Aspect 2: Testing

- Formalizing: What objects are in an image and where are they?

Training

- The training data available for fitting the ML model

Testing (a.k.a. Inference)

- The input data given to the model
- The output required from the model
- The evaluation algorithm & metrics

Aspect 2: Testing

- Formalizing: What objects are in an image and where are they?

Training

- The training data available for fitting the ML model

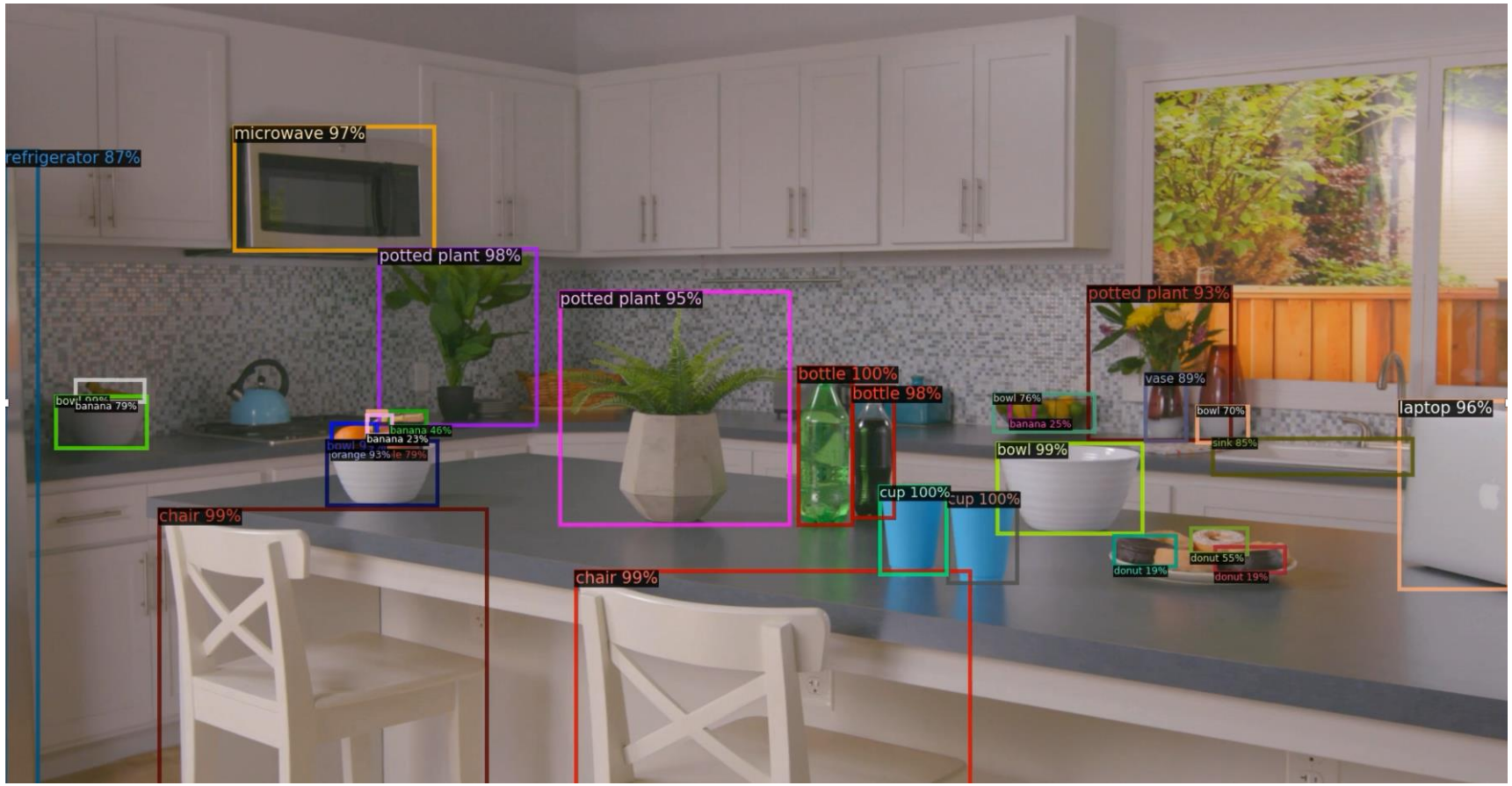
Testing (a.k.a. Inference)

- The input data given to the model
- The output required from the model
- The evaluation algorithm & metrics

Task formulation (testing) – input

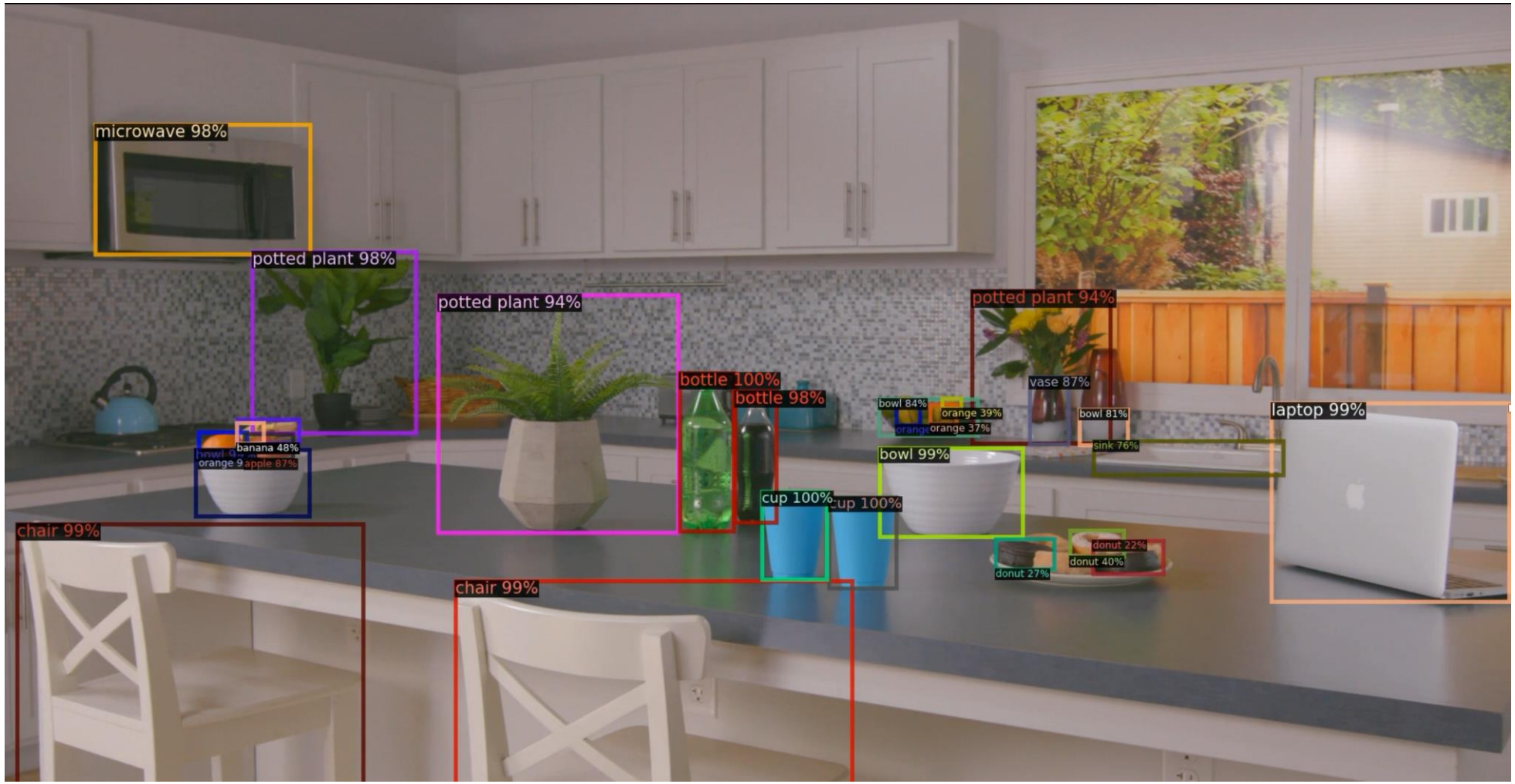
Input: an image (*not* seen during training)

Task formulation (testing) – output

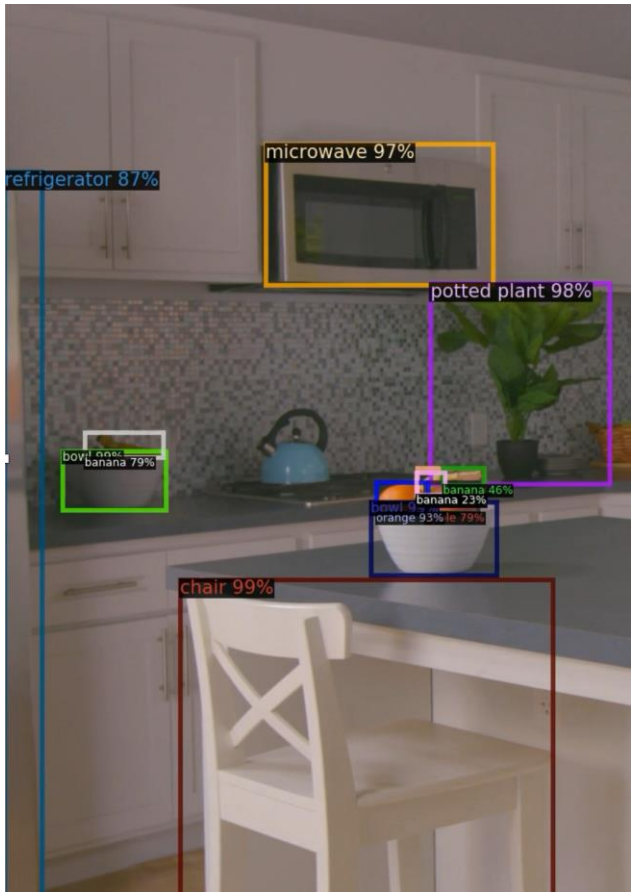


Output: boxes, categories, and confidence scores

“Where” means bounding boxes



Task formulation (testing) – output, continued



Output:

- Set of boxes – each test image may have a different number
- Category ("class") per box – comes from a pre-defined list of object categories (given by the dataset)
- Confidence score per box – a number where higher indicates the model "thinks" the output is more likely correct

Recap: Task formulation (testing)

- The test-time task formulation is deceptively simple
  - Images in; boxes, categories, and scores out!
  - (Later today we'll discuss why this problem is actually very hard)
  - That's it!

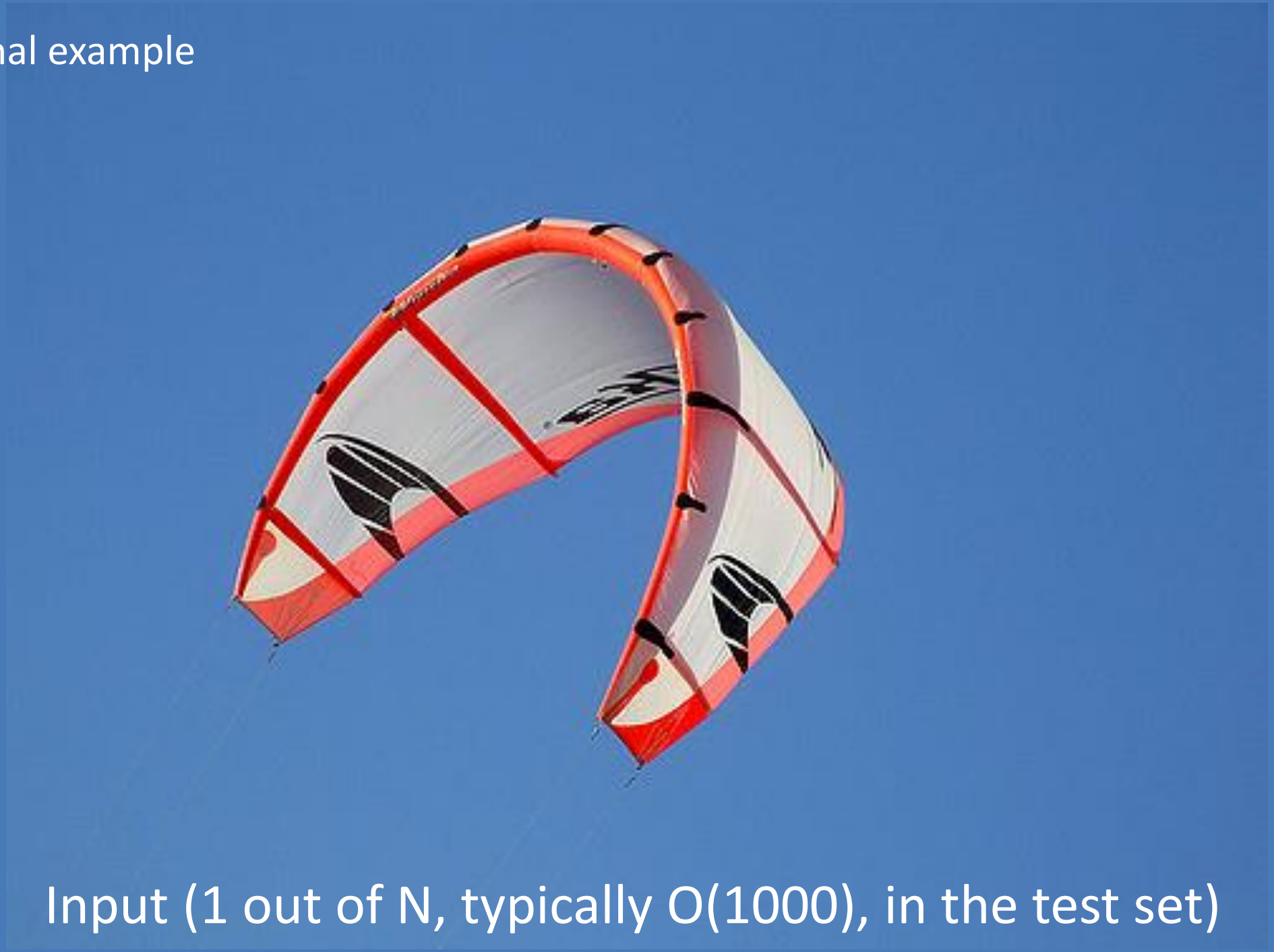
- Next up: Evaluation! (So, you have a detector; *is it any good?*)

Evaluating an object detector

- Given the inputs, how good are the outputs?

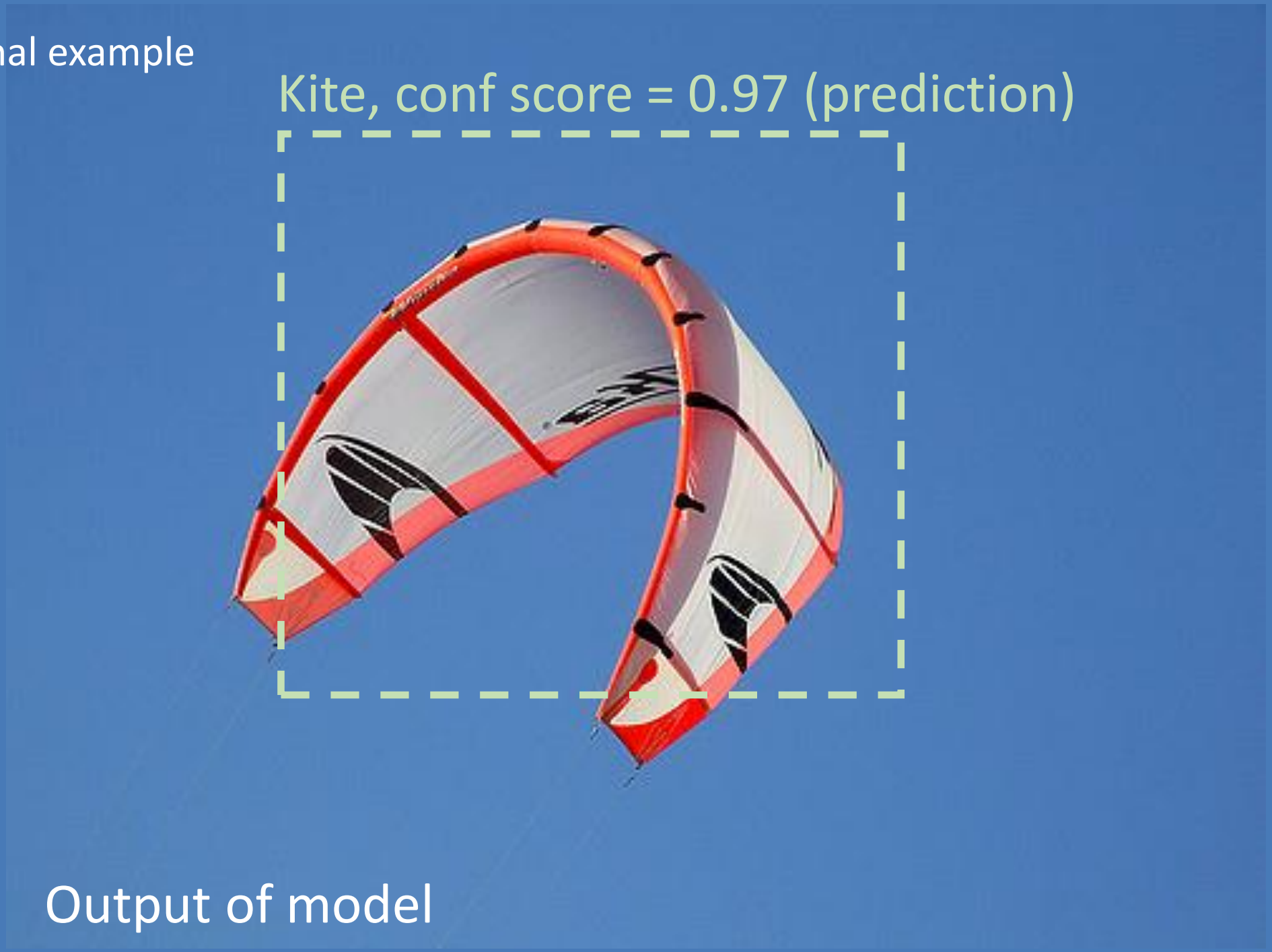
- More nuanced than image classification
  - Need to evaluate both what and where
  - Ground-truth & prediction are almost never *exactly* the same
  - Key questions: When is a prediction correct? When is it incorrect? What are we missing?

Motivational example



Input (1 out of N, typically O(1000), in the test set)

Motivational example



Output of model

Motivational example

Output and ground truth

Motivational example

Is this output good? Is it correct? Is it incorrect?

How should a good detector behave? (Intuition)

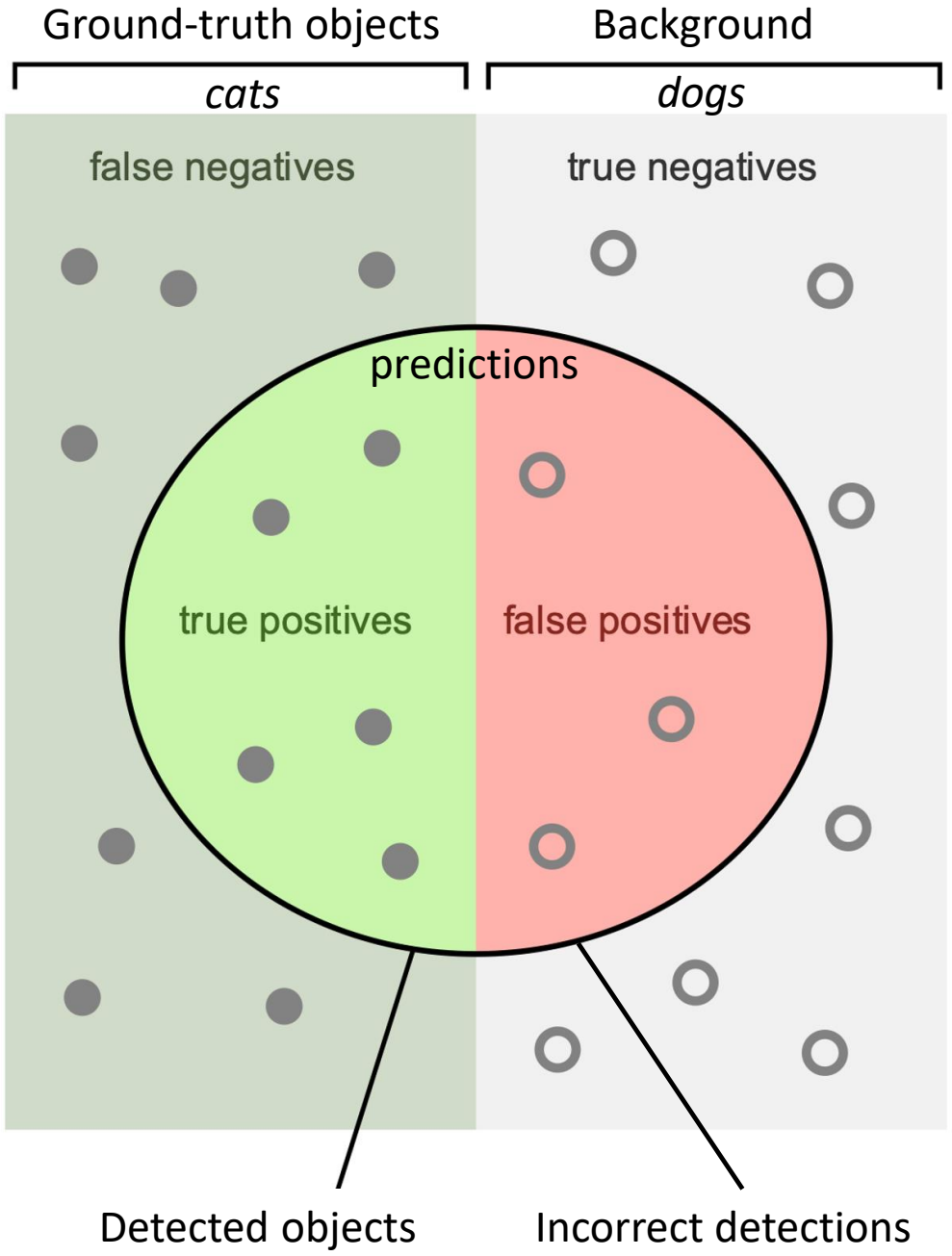
1. Perfect localization is *not* necessary for most applications
   "close is good enough"
2. Names the object correctly
   "don't call a *cat*, *dog*"
3. Doesn't find an object more than once
   "don't duplicate detections"
4. Finds all instances of each object category
   "don't miss anything"

Evaluation “oracle”

- The oracle (i.e., the evaluation code / protocol) has access to
  - Ground-truth annotations (boxes and classes) – *privileged* information
  - Predictions from a model (boxes, classes, and confidence scores)
- The oracle evaluates each category *independently*
- In particular, the oracle…
  - Sorts the predictions from highest to lowest confidence
  - Processes each prediction, one at a time in this order
  - Assigns a binary classification to each prediction:
    Either “True Positive” (TP) or “False Positive” (FP)
  - Counts the number of “False Negatives” (FN)
    I.e., objects that were *not* detected

Terminology

- True Positive (TP)
  - A prediction that is correct
- False Positive (FP)
  - A prediction that is incorrect
- False Negative (FN)
  - A ground-truth object that the detector missed
    I.e., it's not in the predictions



Modified from: https://en.wikipedia.org/wiki/Precision_and_recall

Criteria that define a true positive

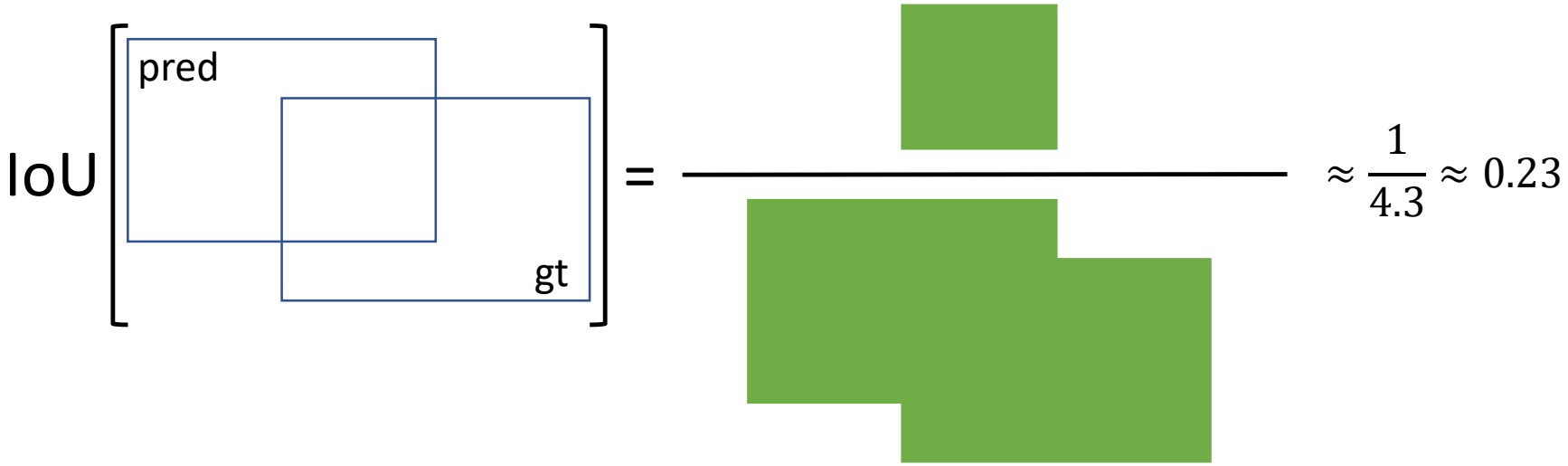
Prediction *P* is a true positive iif there exists a ground-truth *G* such that:

1. *P*'s box has sufficient similarity to *G*'s box
2. *P* and *G* have the same category label
3. *G* was not matched to a higher scoring prediction *P*'

If (1 - 3) are satisfied, then we say *P* and *G* are matched

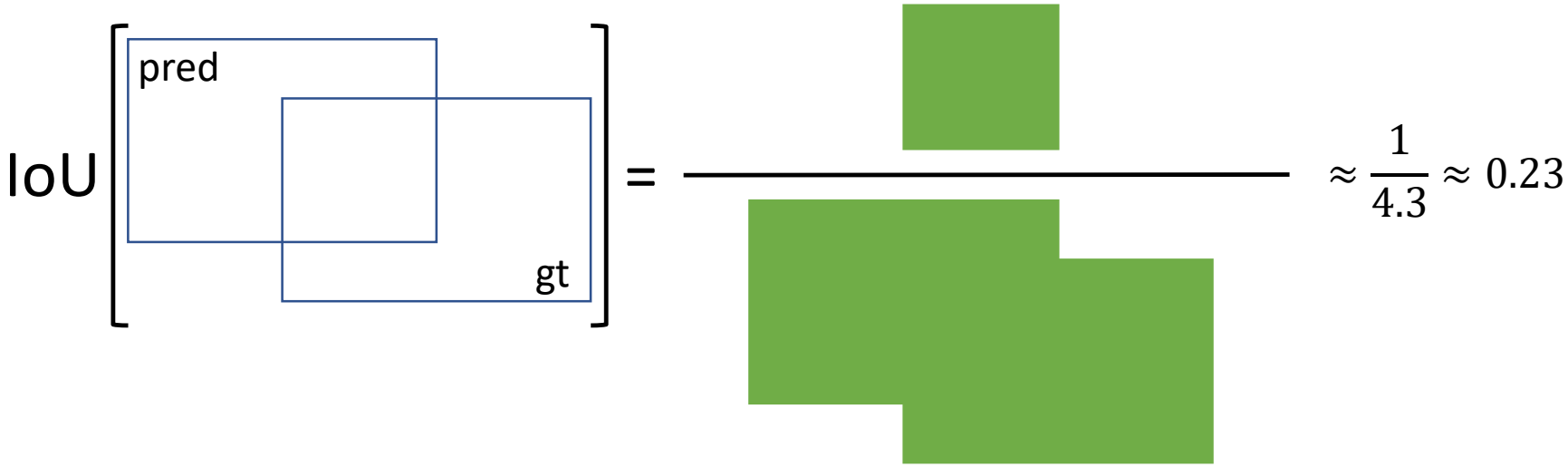
(Subtle note: Ranking by score is used to define matches; other strategies, such as optimal bipartite matching are possible, but rarely used in practice.)

Defining similarity between two boxes

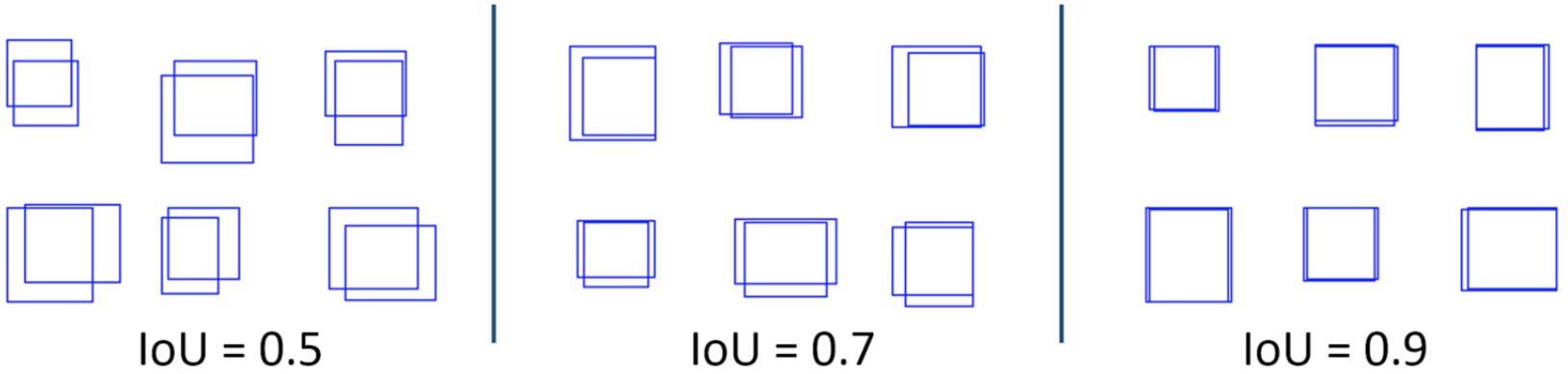


This definition of similarity is called “Intersection over Union” (IoU)
It is also called “Jaccard Similarity”

Defining similarity between two boxes

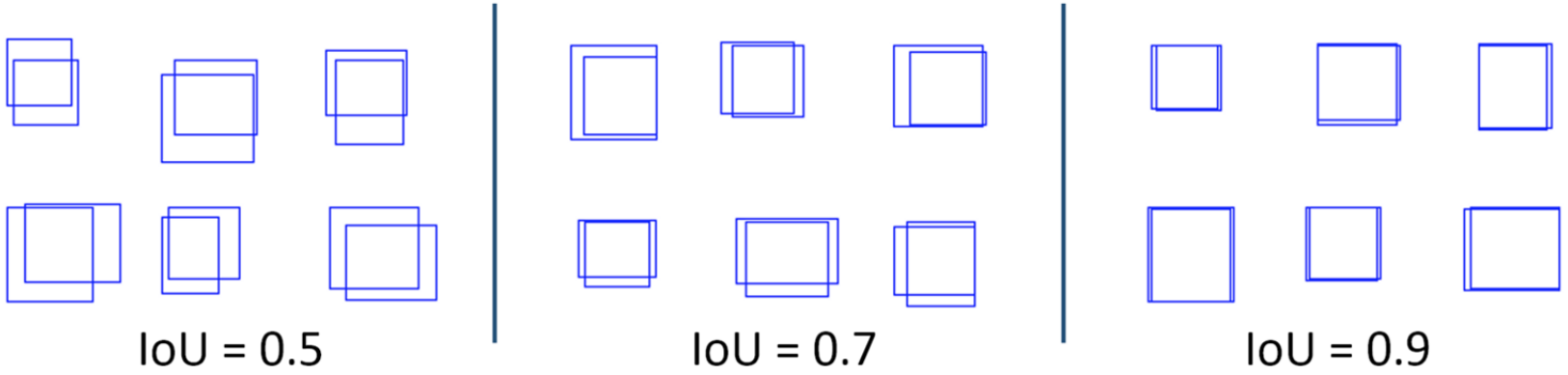


Random examples for your intuition:



Defining sufficient similarity

How much IoU is good enough (i.e., sufficient)?



Two schools of thought:

1. Pick one threshold that seems reasonable (e.g., IoU = 0.5)
2. Consider multiple thresholds (we will discuss this later)

Criterion that defines a false positive

Prediction *P* is a false positive, if *P* has no matching ground truth

Criterion that defines a false negative

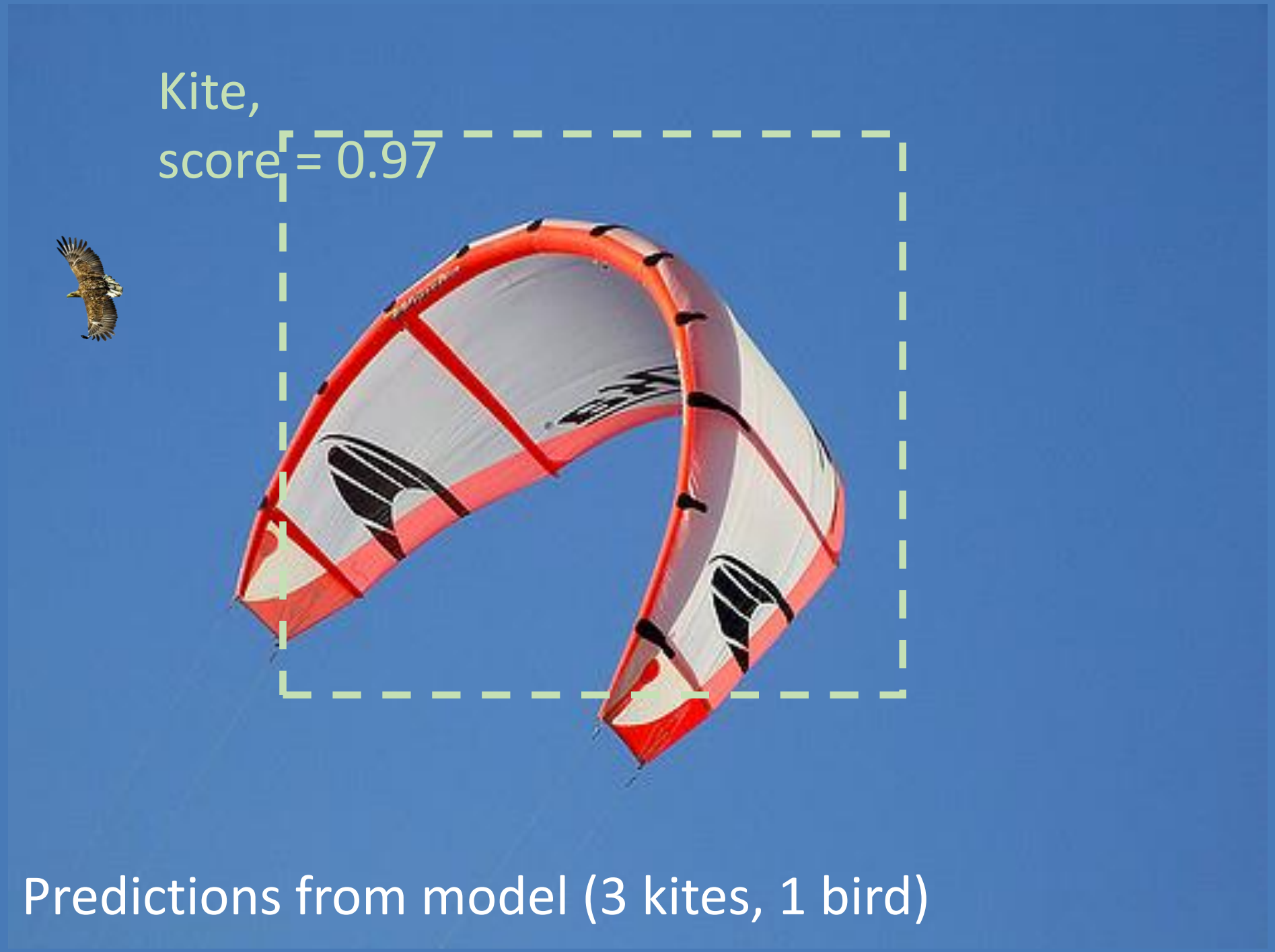
Ground-truth *G* is a false negative, if *G* has no matching prediction

Example evaluation for one image

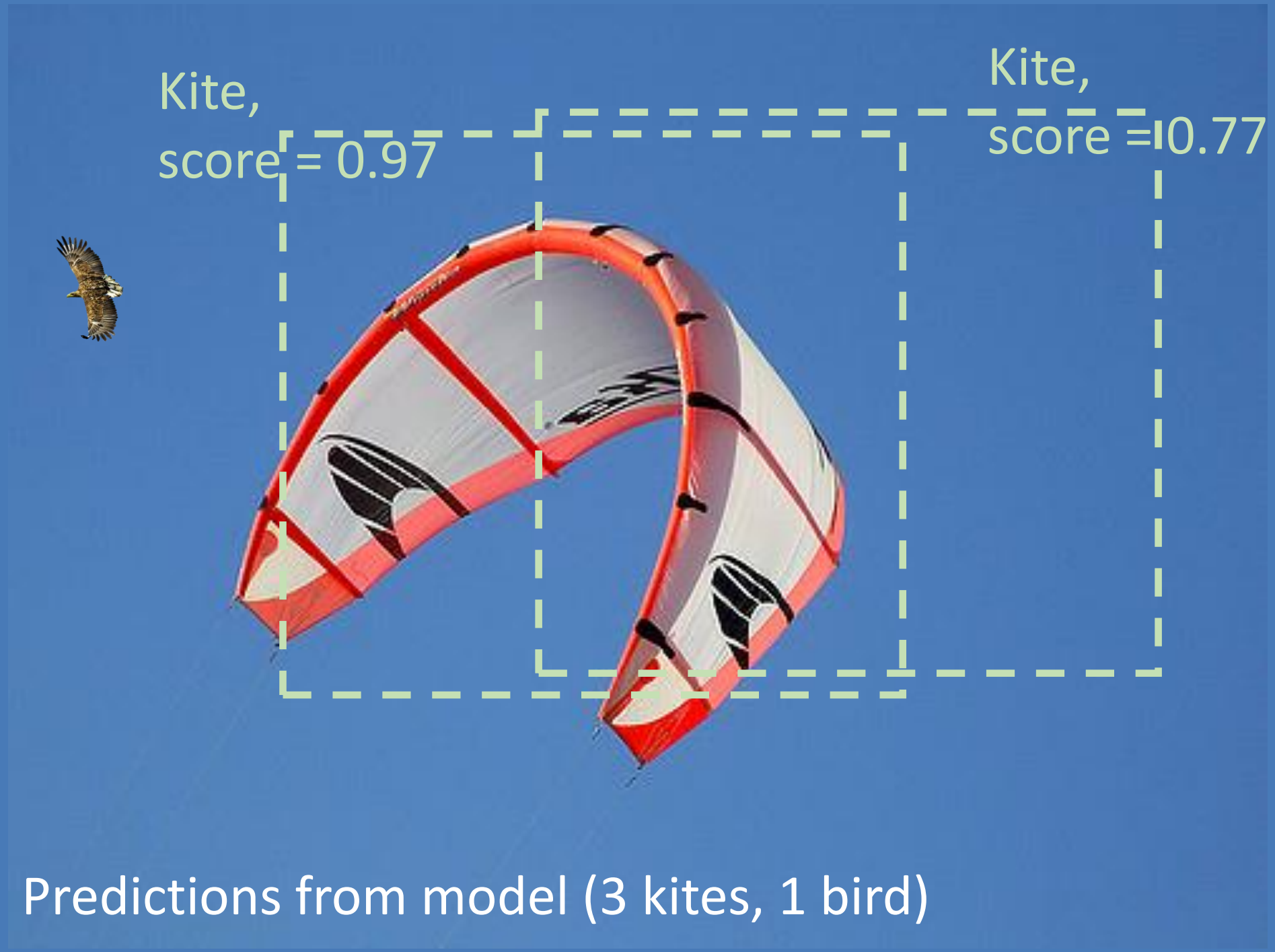
- We'll use IoU threshold 0.5
- The dataset has two categories: {bird, kite}

Testing image

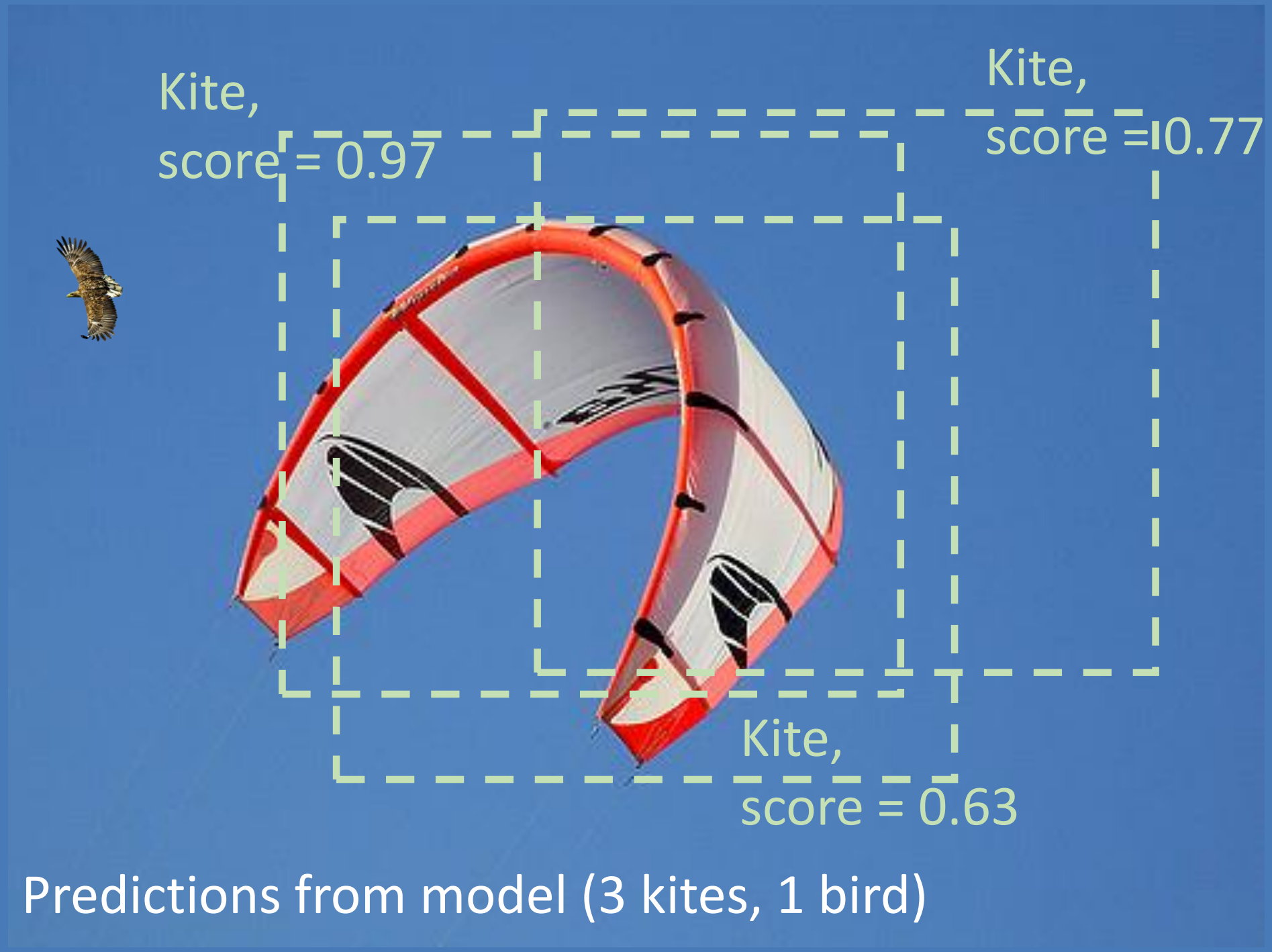
Predictions from model (3 kites, 1 bird)



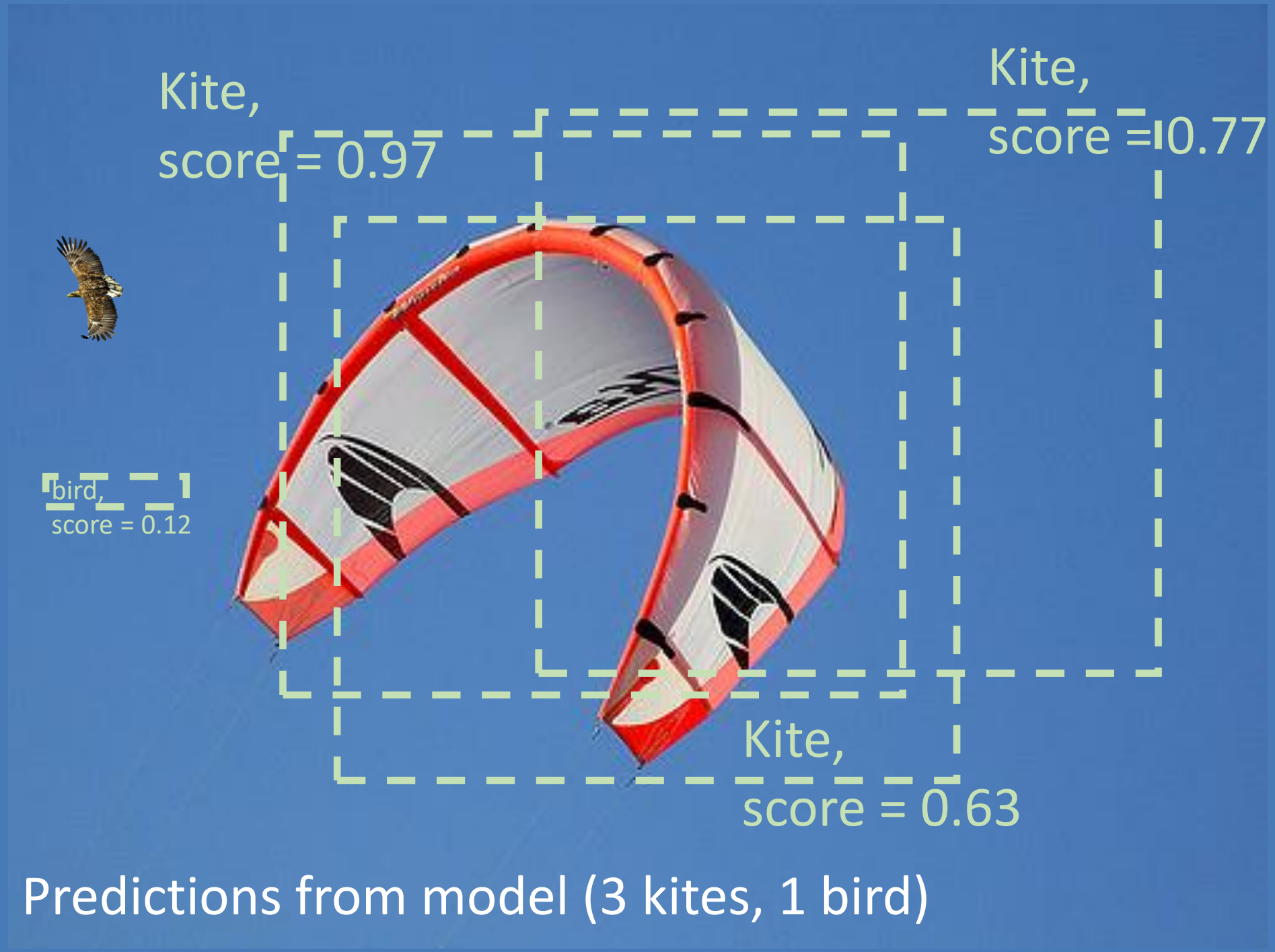
Predictions from model (3 kites, 1 bird)



Predictions from model (3 kites, 1 bird)



Predictions from model (3 kites, 1 bird)



Predictions from model (3 kites, 1 bird)

bird
(ground truth)

Ground-truth available to evaluation oracle

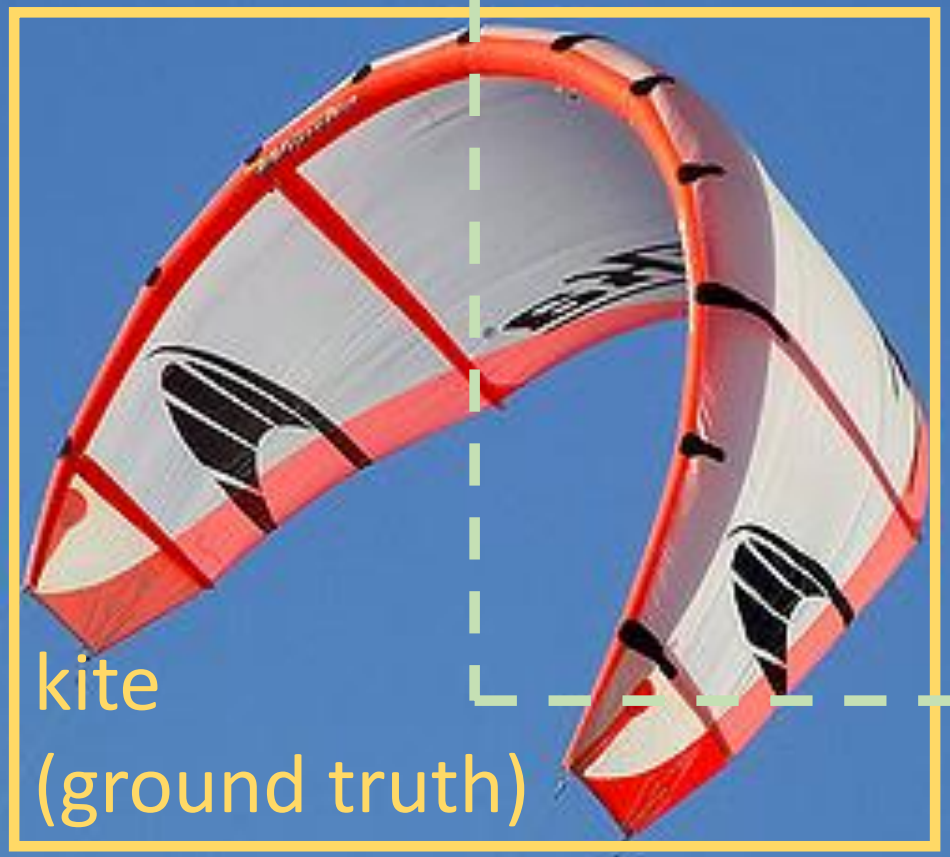
Recall what the evaluation oracle does

- The oracle evaluates each category *independently*
- The oracle
  - Sorts the predictions from highest to lowest confidence
  - Processes each prediction, one at a time in this order
  - Assigns a binary flag to each prediction:
    Either “True Positive” (TP) or “False Positive” (FP)
  - Counts the number of “False Negatives” (FN)
    I.e., objects that were *not* detected

Kite, conf score = 0.97 (prediction)

IoU 0.65 >= 0.5? (the IoU threshold)

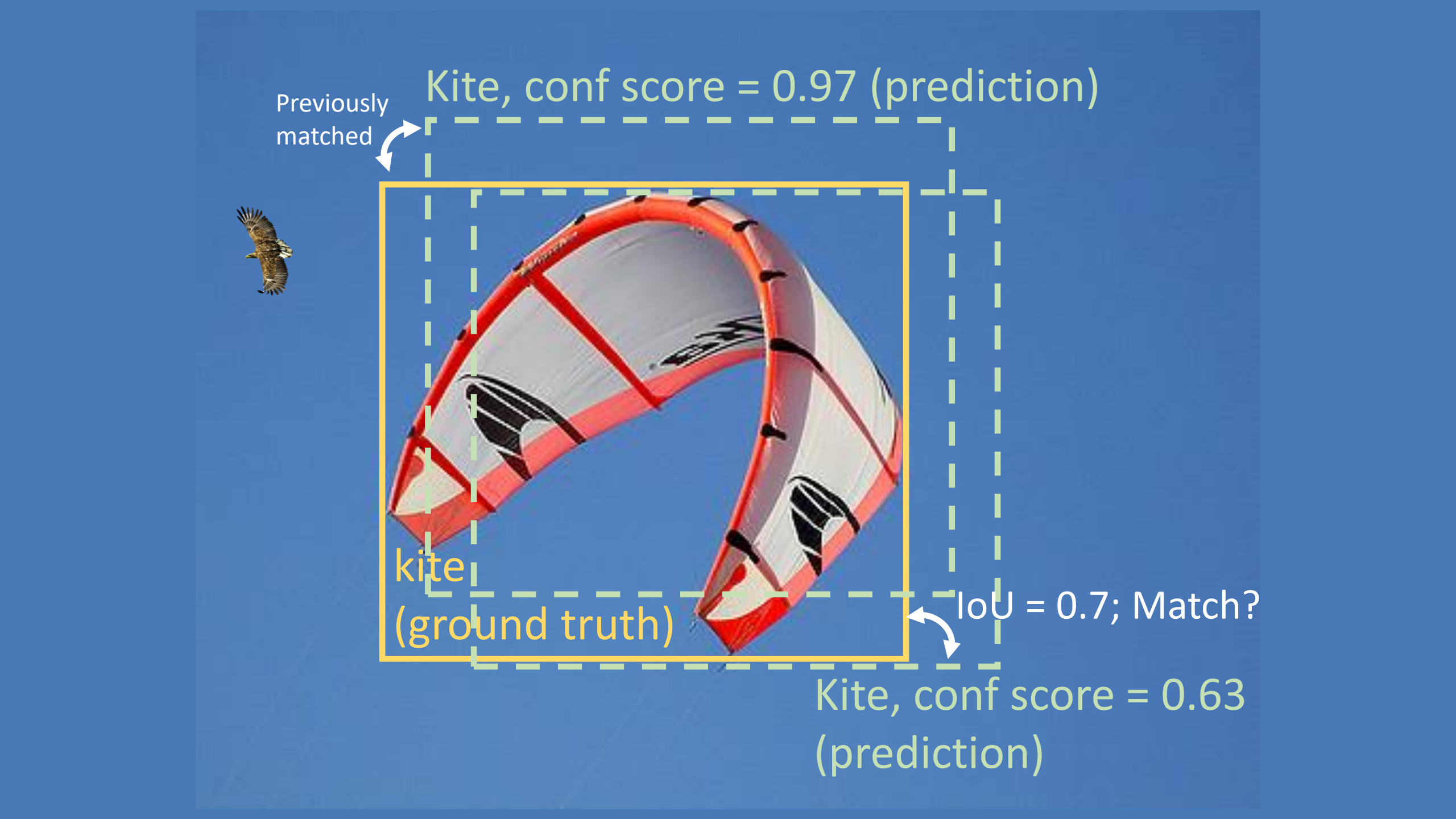
→ True positive



IoU 0.29 < 0.5?

→ False positive


kite
(ground truth)
IoU = 0.7; Match?
Kite, conf score = 0.63
(prediction)


Kite, conf score = 0.97 (prediction)
Previously matched
kite (ground truth)
IoU = 0.7; Match?
Kite, conf score = 0.63 (prediction)

Previously matched

Kite, conf score = 0.97 (prediction)

kite (ground truth)

IoU = 0.7; Match? → Nope

Duplicate detection → False positive

Kite, conf score = 0.63 (prediction)

bird
(ground truth)

IoU 0.0 < 0.5?
→ False positive

bird,
score = 0.12

No matching prediction
→ False negative

bird
(ground truth)

Recap: Evaluation oracle

- For each category C, a separate list of confidence ranked predictions
  - For each prediction of C, a decision if it is a TP or a FP
  - A set of FNs for C

How should a good detector behave? (Intuition)

1. ✓ Perfect localization is *not* necessary
   "close is good enough" → "close" is determined by IoU threshold
2. ✓ Names the object correctly
   "don't call a *cat, dog*" → exact category match
3. ✓ Doesn't find an object more than once
   "don't duplicate detections" → confidence scores define ranking
4. ✓ Finds all instances of each object category
   "don't miss anything" → penalize missed objects (FNs)

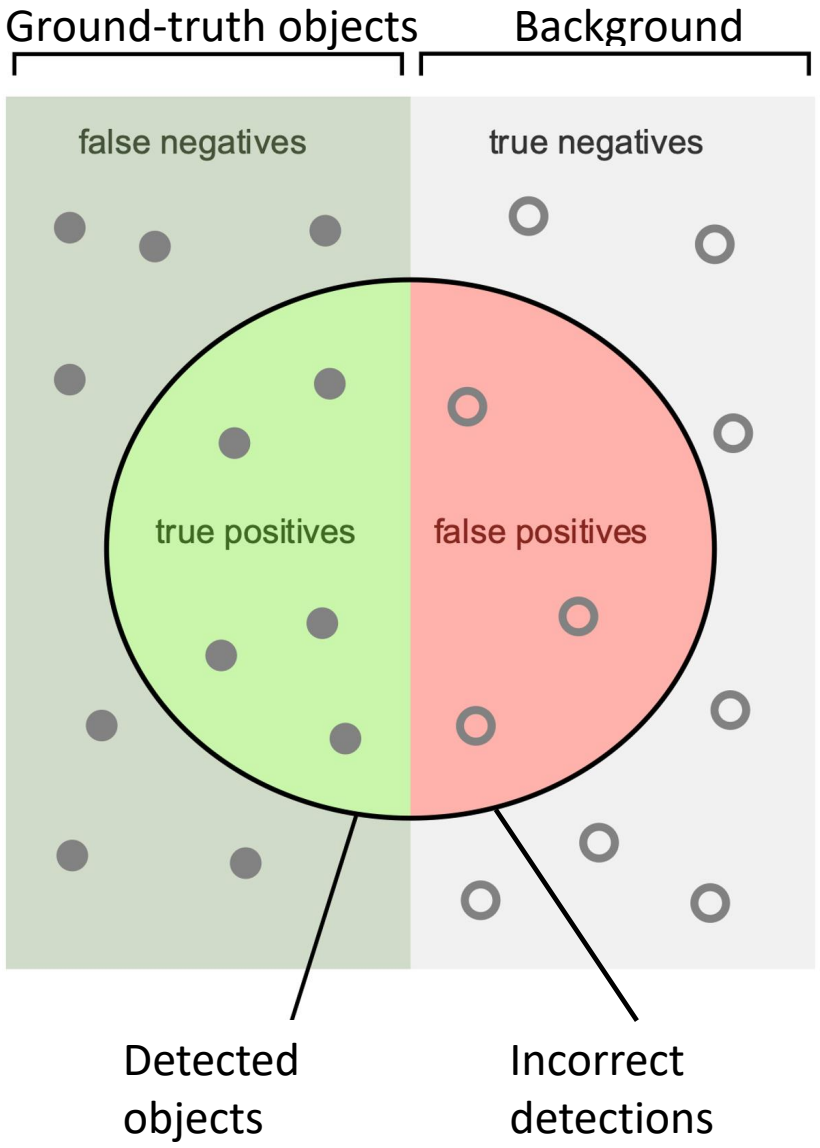
How do we *quantify* "goodness"?

- Thus far all we can do is categorize predictions (TP vs. FP) and count false negatives (FNs)

- What we want: an evaluation metric
  - A single number that *communicates* how good the model is on the dataset
  - Something like *accuracy* for image classifications

- Next, we'll build up the definition of Average Precision (AP)

Preliminaries: Precision and recall



Precision = 

"Of the detections, what fraction are correct?"

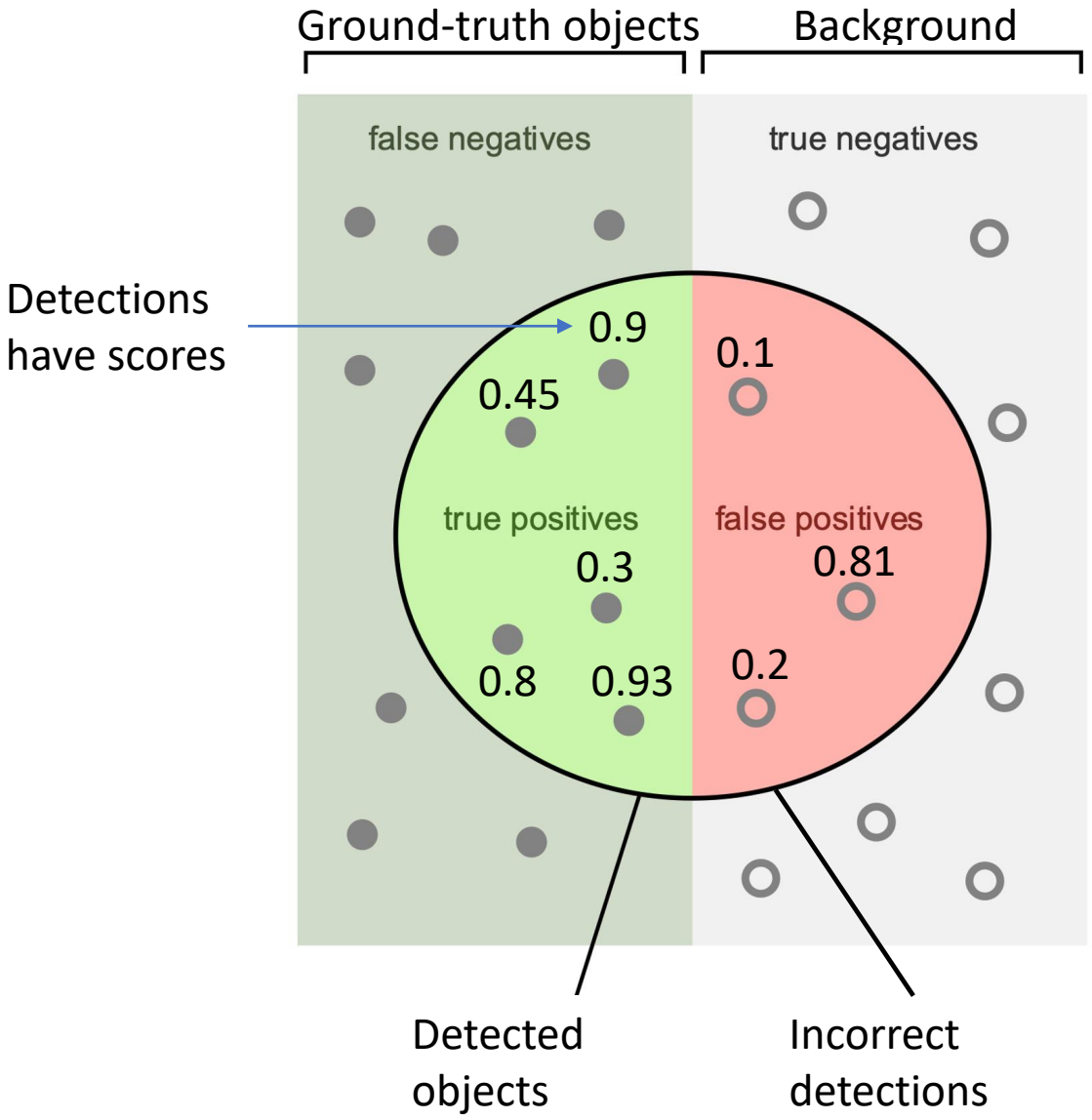
#TP / (#TP + #FP)

Recall = 

"What fraction of the ground-truth was detected?"

#TP / (#TP + #FN)

Preliminaries: Precision and recall "@s"



Precision@*s*

"Of the detections **with score >= *s***, what fraction are correct?"

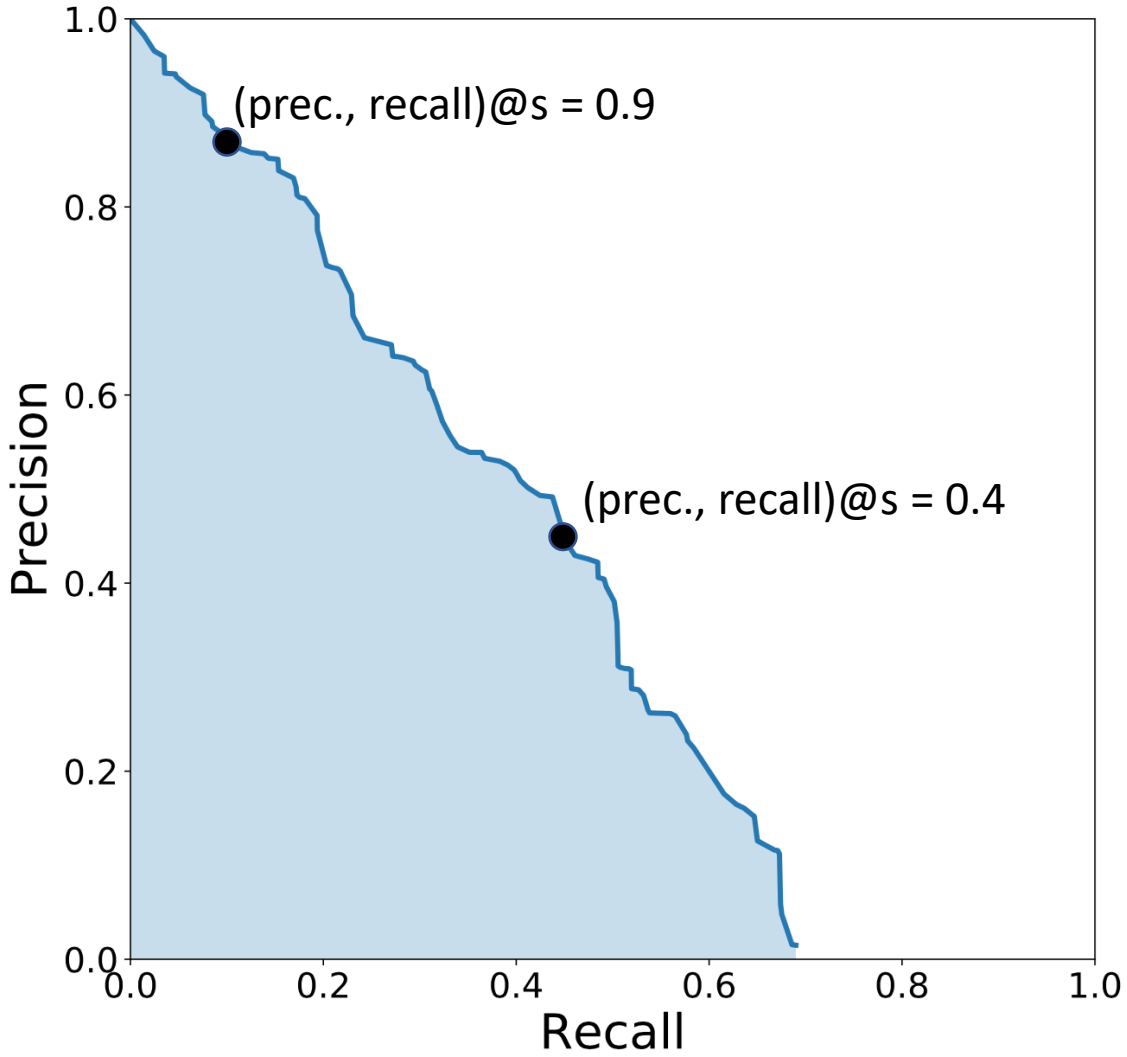
#TP@s
/ (#TP@s + #FP@s)

Recall@*s*

"What fraction of the ground-truth was detected **with score >= *s***?"

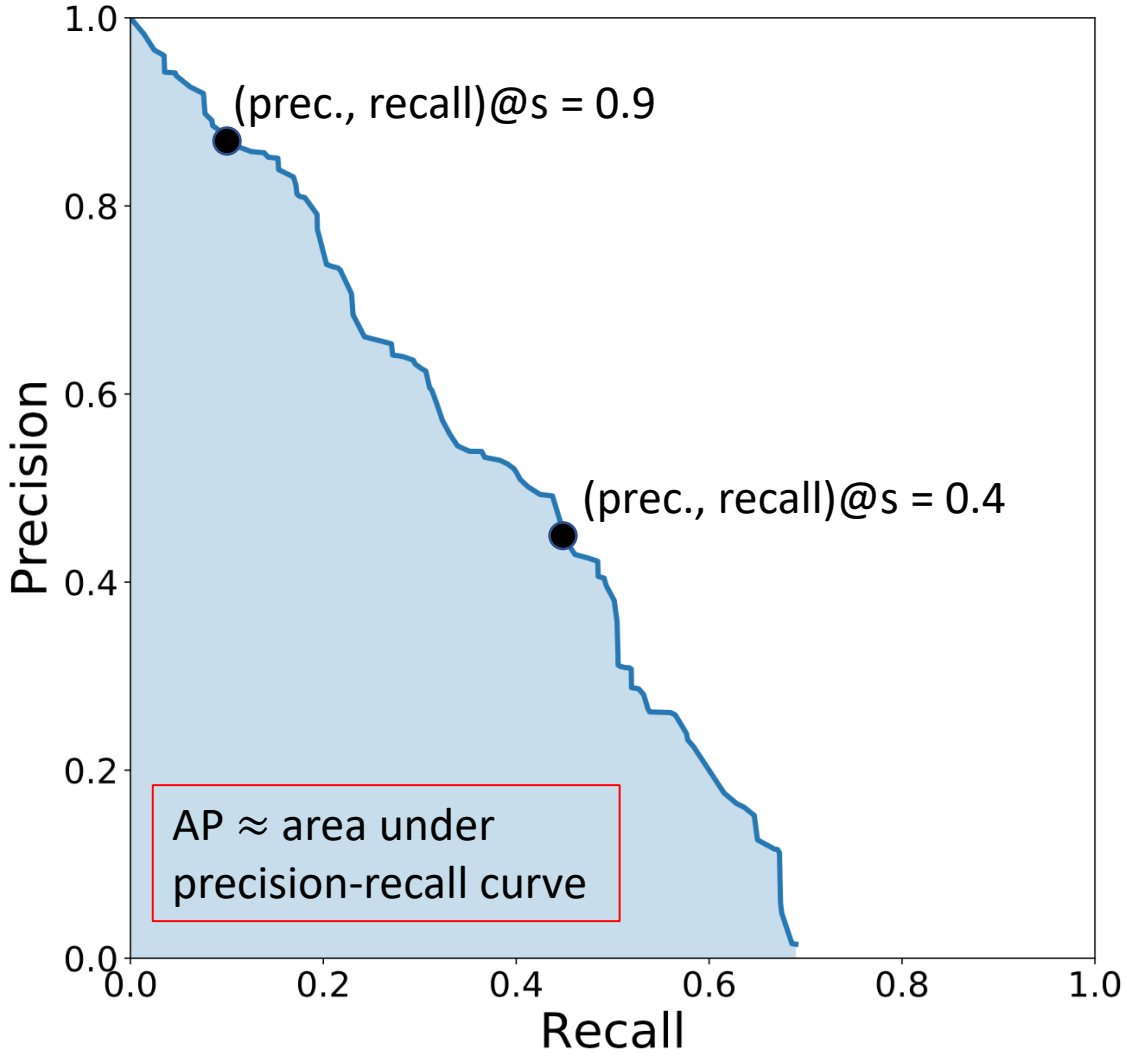
#TP@s
/ (#TP@s + #FN@s)

Precision-recall (PR) curve



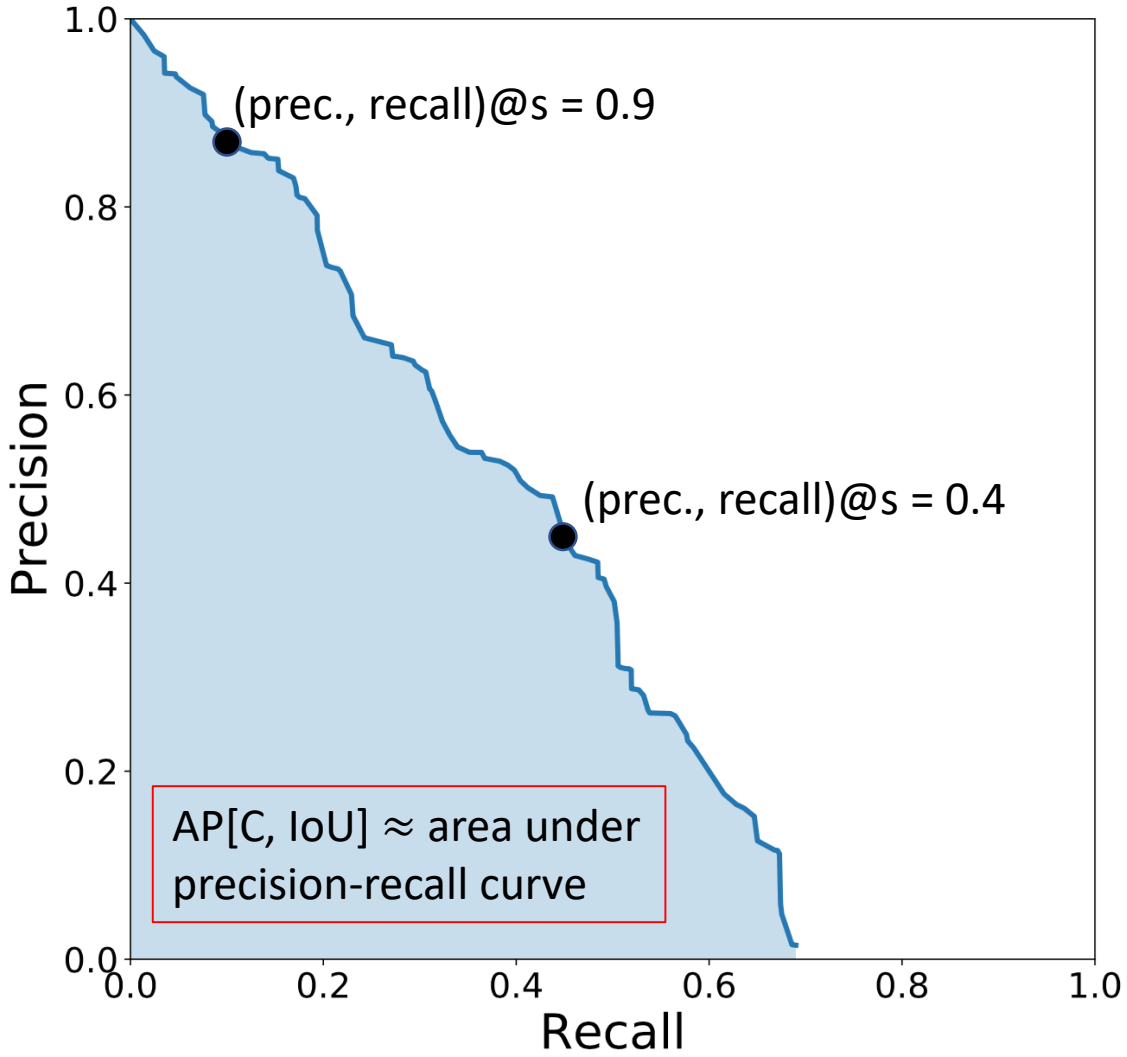
- A curve parameterized by score *s*
- Sweeping (precision, recall)@*s* from high to low *s* traces out the PR curve
  - Example points at *s* = 0.9 and *s* = 0.4
- (Note: this curve is for one category at a time; each category has a separate PR curve)

Average precision (AP) of the PR curve



- Average precision (AP) := the area under the PR curve
- Average is over all levels of recall
- AP ranges from 0 to 1
- AP of 1 means *perfect precision* and *perfect recall*
- AP is computed *numerically* given finite (prec., recall)@*s* samples
  - (Warning: datasets use different low-level details for this calculation)

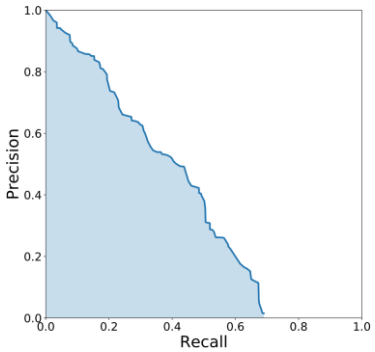
Average precision (AP) of the PR curve



Notation: AP[C, IoU]

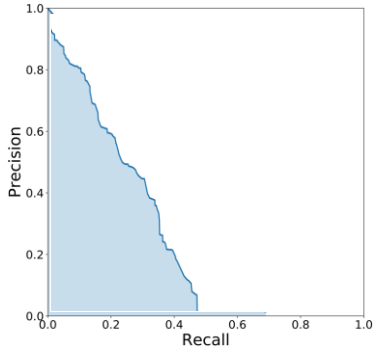
- Category C
- IoU threshold
- E.g., AP[“kite”, 0.5]

Extending AP to multiple categories



AP[“kite”, 0.5]

…



AP[“bird”, 0.5]

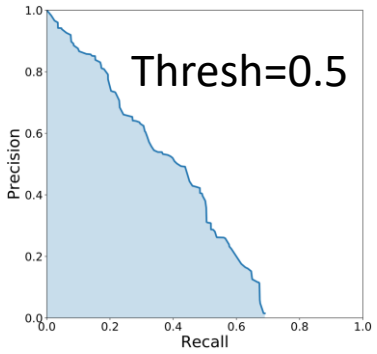
…

$$\mathbb{C} = \{\text{kite}, \text{cat}, \text{dog}, \dots, \text{bird}, \dots\}$$

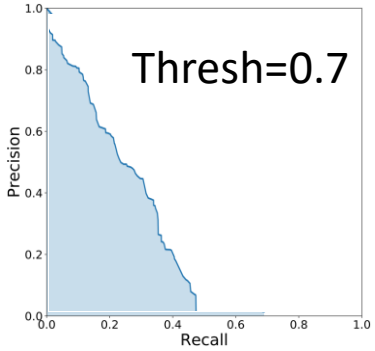
$$\text{AP}[0.5] = \frac{1}{|\mathbb{C}|} \sum_{C \in \mathbb{C}} \text{AP}[C,\ 0.5]$$

This is often called “mean average precision” or “mAP”

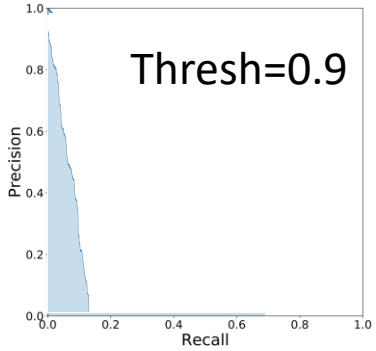
Extending AP to multiple IoU thresholds



...



...

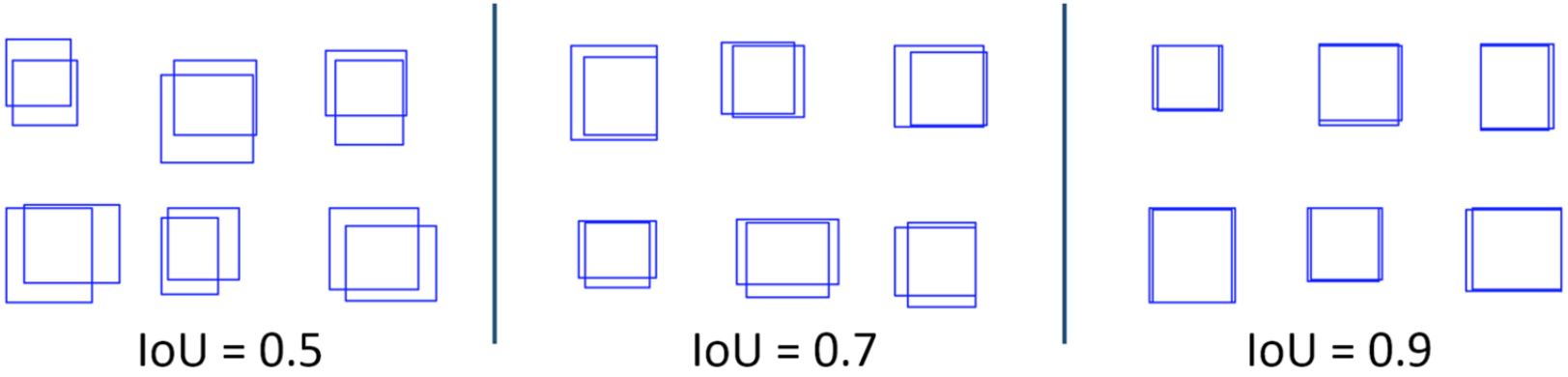


...

AP[“kite”, 0.5]

AP[“kite”, 0.7]

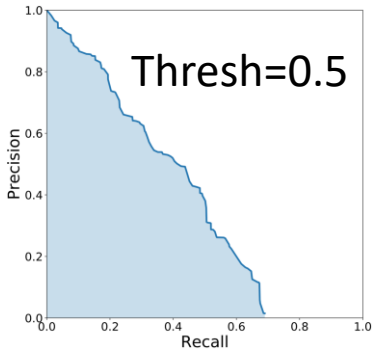
AP[“kite”, 0.9]



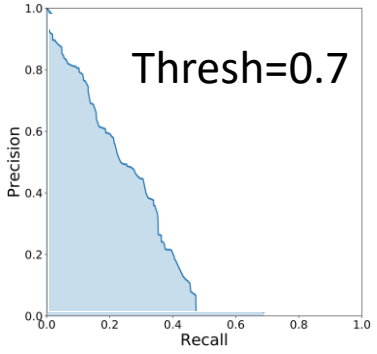
Loose

Tight

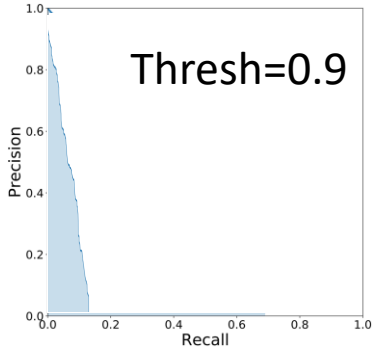
Extending AP to multiple IoU thresholds



...



...



...

AP[“kite”, 0.5]

AP[“kite”, 0.7]

AP[“kite”, 0.9]

$$\mathbb{T} = \{0.50, 0.55, 0.60, 0.65, \ldots, 0.95\}$$

$$\text{AP}[\text{“kite”}] = \frac{1}{|\mathbb{T}|} \sum_{t \in \mathbb{T}} \text{AP}[\text{kite}, \ t]$$

Multiple categories *and* IoU thresholds

$$\mathbb{C} = \{\text{kite}, \dots, \text{bird}, \dots\}$$

$$\mathbb{T} = \{0.50, 0.55, 0.60, 0.65, \dots, 0.95\}$$

$$\text{AP} = \frac{1}{|\mathbb{T}||\mathbb{C}|} \sum_{C \in \mathbb{C}, t \in \mathbb{T}} \text{AP}[C,\ t]$$

This is an average of precision over recall, categories, and IoU thresholds

("mean mean average precision"? – Most just call it "average precision")

Recap: Evaluation metric

- The most popular metric is “average precision” (AP)
  - Precision values averaged over recall ranging from 0 to 1
  - Caveat: there are *multiple* definitions of “AP”
    - Can be AP for a single category and single IoU threshold
    - AP for a single category, averaged over multiple IoU thresholds
    - AP averaged over multiple categories and multiple IoU thresholds
    - Low-level numerical computation details can differ
    - Can be difficult to disambiguate without context
    - Rule of thumb: each dataset defines a metric (e.g., “COCO-style AP”, “PASCAL VOC-style 2007 AP”, “PASCAL VOC 2010-style AP”, etc.)

Which definition of AP to use?

- Largely a philosophical question
- Real-world applications are driven by *real-world requirements*
- AP is defined *without* any grounding in a particular application
  - The metric is *generic* → doing well should be good for *many applications*
  - The goal of AP = 1 is generically good (perfect recall, perfect precision)
- Early datasets (e.g., PASCAL VOC) used IoU threshold = 0.5
  - The detection problem was new and “too hard”
- Later datasets (e.g. COCO) extended to multiple IoU thresholds
  - The detection problem is established, there’s good progress; aim higher!
- Up next: Task formulation at training time

Back to aspect 1: Training

- Formalizing: What objects are in an image and where are they?

Training

- The training data available for fitting the ML model

Testing (a.k.a. Inference)

- The input data given to the model
- The output required from the model
- The evaluation algorithm & metrics

Task formulation (training) – input

- A dataset, composed of:

Typically, 10,000 – 100,000

Car,
Person,
Football,
Bottle,
Sunglasses,
Giraffe,
…

A pre-specified set of categories
Typically, 10 – 1,000

For each image:
boxes for each category

Task formulation (training) – output

- A trained model that can be used in testing

- All other details are specific to the model and training algorithm
  - They are not part of the task formulation

Recap: Learning objectives

Today:

1. What is the object detection task?
2. How is the task formalized?
3. How is the task evaluated?
4. Why is the task difficult?
5. What are the important datasets to know about?
6. Tips for reading research papers on object detection [stretch goal]

Why is object detection difficult?

- Invariance
- Equivariance
- Imbalance

Invariance

A function f is **invariant** to a transformation g if

$$\forall x \;\; f(gx) = \;\; f(x)$$

$f(R_\theta$  $) = f($  $) = f($  $)$

$= \text{“2”}$

The “what” part of a detector (“*f*”) needs to be *invariant* to nuisance transformations – this is like image classification

Example: Photometric variation

- Bias and gain $I_2 = \alpha I_1 + \beta$

Negative bias

Fractional gain

- Photometric variations should not change the detector's output

Example: Intra-class variation

- Appearance
- Structure

ikea

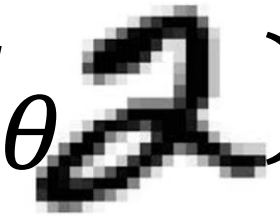
- What makes a chair a chair?
  - (Unhelpful) answer: “A chair is a chair because it looks like a chair”
- Variation within a category should not change the detector’s category prediction

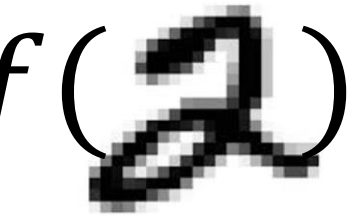
Equivariance

A function f is **equivariant** with transformation g if

$$\forall x \; \exists M_g \quad f(gx) \approx M_g f(x)$$

$$f(S_\theta \;) = f(\;) = f(\;) + \theta = \theta$$

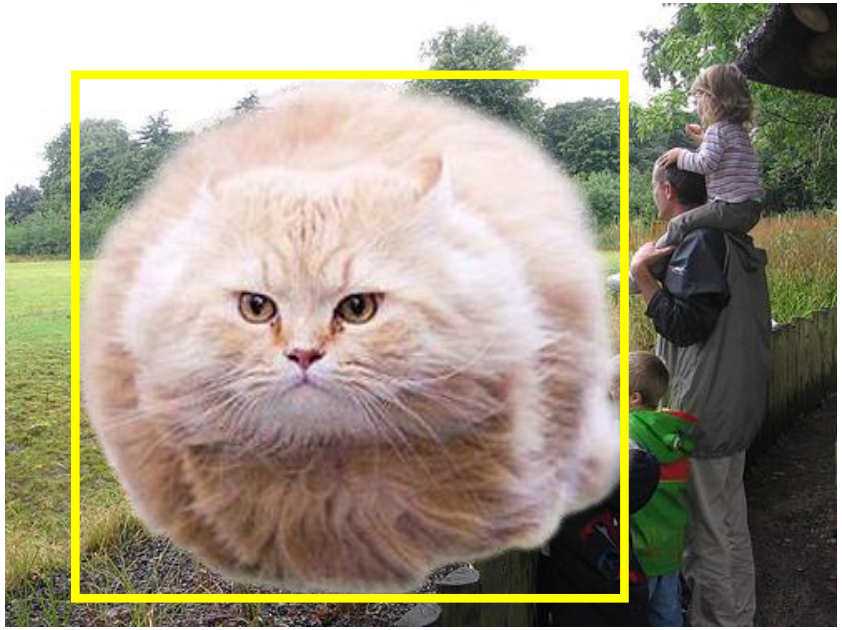




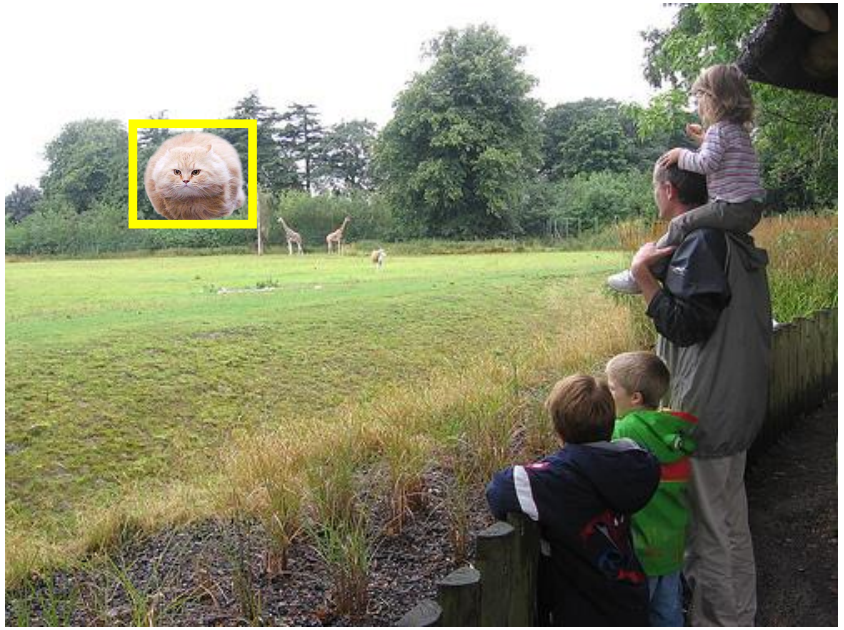
The “where” part of the detector needs to be *equivariant* with geometric transformations of objects – this is new compared to image classification

Example: Scale variation

- Detection: “where” (i.e., the box) is equivariant with scale shift
- (Contrast to classification: invariant to scale shift)



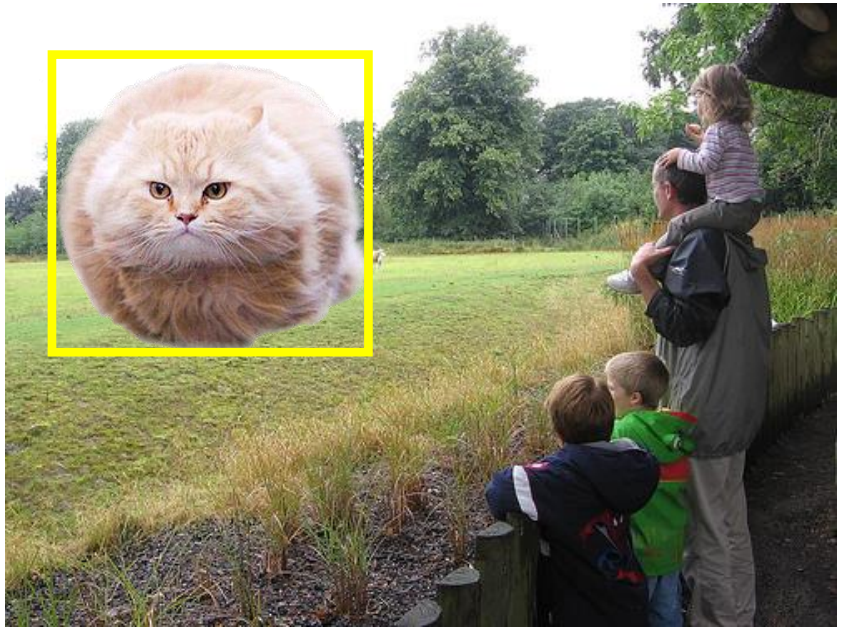
Cat



Cat

Example: Translational variation

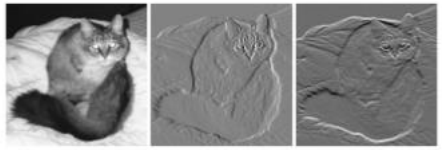
- Detection: “where” (i.e., the box) is equivariant with translation
- (Contrast to classification: invariant to translation)



Cat

Cat

Sources of variation



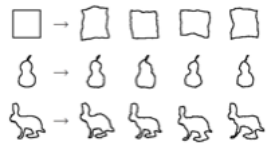
Photometric

Translation

Rotation, rigid pose

Scale

Occlusion (self, other)



Deformation (local, articulated)

Intra-class
(appearance, structural)

Context
(clutter, spurious corr.)

And more…

Imbalance

- A test set will typically only contain O(100) – O(1000) objects of each category C
- There are *vastly* more possible boxes than this, ~ O(1e15)
  - These boxes belong to an implicit “background” category
  - Classification of category C is effectively C vs. “background” (and other categories)
- From a machine learning perspective, learning with highly imbalanced data is a difficult problem
  - (More on this in tomorrow’s lecture)

Recap: Why is object detection difficult?

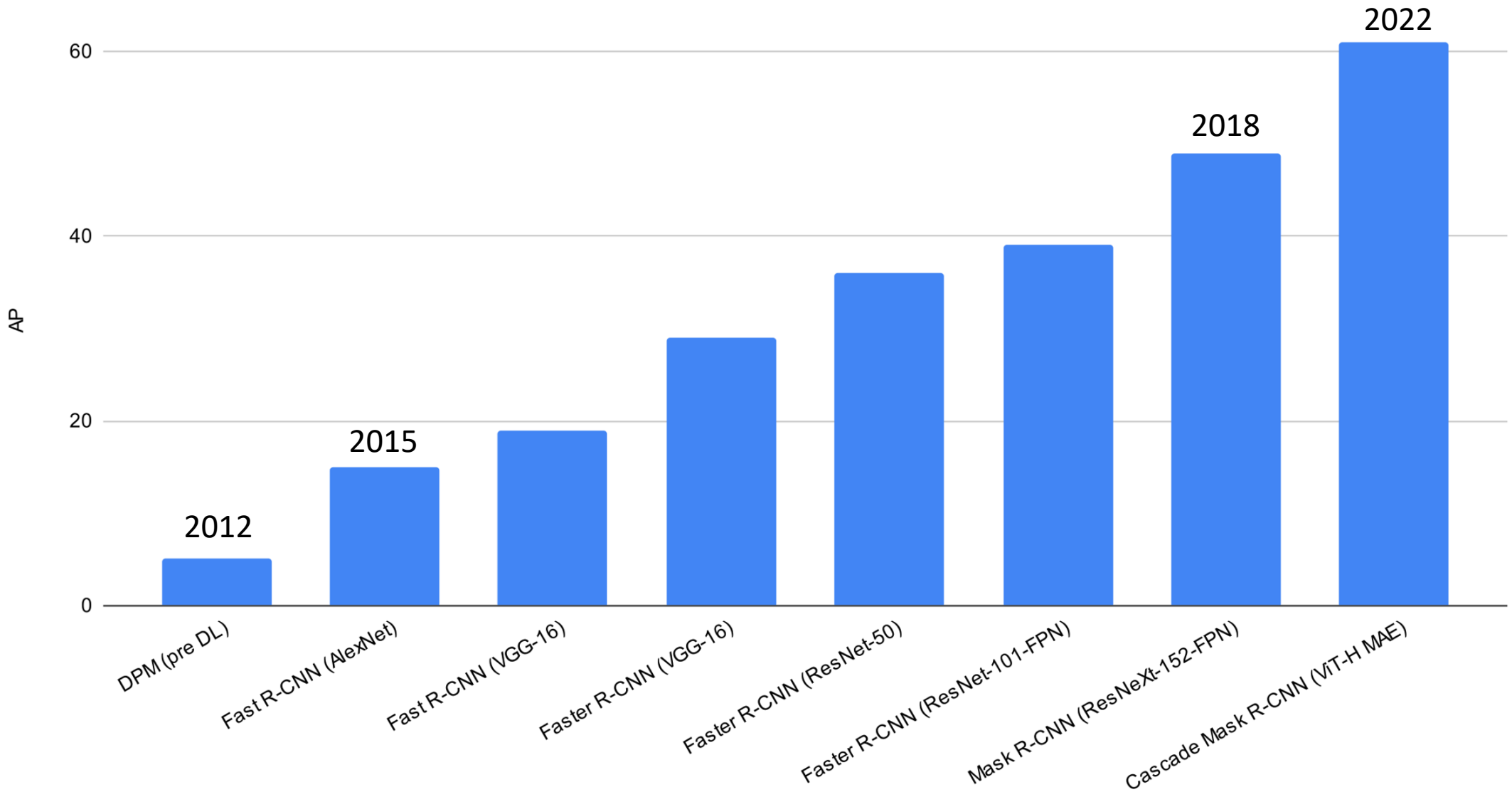
- A good detector needs a combination of invariance and equivariance properties
  - "What" needs to be *invariant* to many challenging transformations
  - "Where" needs to be *equivariant* to many challenging transformations
  - Compared to image classification, which is only about invariance

- Object detection implicitly poses an imbalanced learning problem
  - O(1e3) objects vs. O(1e15) background boxes
  - Image classification is typically balanced

Popular object detection datasets

- PASCAL VOC (retired)
  - The original high-quality, large-scale (at the time) detection dataset
  - 21 object categories
  - ~10k – 20k images
- COCO
  - Leap forward in number of image; established the instance segmentation task
  - 80 object categories
  - ~160k images
- LVIS
  - Leap forward in number of categories
  - ~1200 object categories; natural long-tailed distribution
  - ~160k images (same images as COCO)
- Open Images
- Objects365

COCO average precision (%) over time



Tips on reading papers

- For a newcomer, reading object detection papers is really, really hard!
  - A huge volume of background knowledge is assumed
  - Many implicit community norms
  - Jargon, confusing terminology
  - Misleading comparisons
  - Unscientific statements
  - …
- Many papers are bad, here's what to look for to identify good papers

What did I learn from a paper?

- A paper should be about a focused idea or question
- It should teach you something you didn't know before
- Examples:
  - A new model and training formulation, and what are it's pros and cons?
  - An interesting or surprising empirical finding
  - A simple new baseline that future papers should compare to
  - A new neural network architecture with good scaling properties
  - Showing that a previous result can be reproduced (this is not trivial!)
  - …
- "Novelty" comes in many forms, including new empirical data from existing models

What did I learn from a paper? continued

- If the paper proposes a new model:
  - Under what conditions does it work?
  - When does it *not* work?
  - If the idea has multiple components, which are the most important?
  - Which implementation details are important?
  - What are the empirical runtime and memory requirements?

How papers should show some of these

- Start from a solid baseline
- Apply the main idea to it
- Perform ablations under simple settings
- Show the simplest version that can work well
  - Avoid adding complexity for an additional small gain (e.g., +0.3 AP)

Example ablations: “One table, one message”

net-depth-features	AP	AP_{50}	AP_{75}
ResNet-50-C4	30.3	51.2	31.5
ResNet-101-C4	32.7	54.2	34.3
ResNet-50-FPN	33.6	55.2	35.3
ResNet-101-FPN	35.4	57.3	37.5
ResNeXt-101-FPN	**36.7**	**59.5**	**38.9**

(a) **Backbone Architecture**: Better backbones bring expected gains: deeper networks do better, FPN outperforms C4 features, and ResNeXt improves on ResNet.

	AP	AP_{50}	AP_{75}
softmax	24.8	44.1	25.1
sigmoid	**30.3**	**51.2**	**31.5**
	+5.5	*+7.1*	*+6.4*

(b) **Multinomial *vs.* Independent Masks** (ResNet-50-C4): *Decoupling* via per-class binary masks (sigmoid) gives large gains over multinomial masks (softmax).

	align?	bilinear?	agg.	AP	AP_{50}	AP_{75}
RoIPool [12]			max	26.9	48.8	26.4
RoIWarp [10]		✓	max	27.2	49.2	27.1
		✓	ave	27.1	48.9	27.1
RoIAlign	✓	✓	max	**30.2**	**51.0**	**31.8**
	✓	✓	ave	**30.3**	**51.2**	**31.5**

(c) **RoIAlign** (ResNet-50-C4): Mask results with various RoI layers. Our RoIAlign layer improves AP by ~3 points and AP_{75} by ~5 points. Using proper alignment is the only factor that contributes to the large gap between RoI layers.

	AP	AP_{50}	AP_{75}	AP^{bb}	AP^{bb}_{50}	AP^{bb}_{75}
RoIPool	23.6	46.5	21.6	28.2	52.7	26.9
RoIAlign	**30.9**	**51.8**	**32.1**	**34.0**	**55.3**	**36.4**
	+7.3	*+ 5.3*	*+10.5*	*+5.8*	*+2.6*	*+9.5*

(d) **RoIAlign** (ResNet-50-**C5**, *stride 32*): Mask-level and box-level AP using *large-stride* features. Misalignments are more severe than with stride-16 features (Table 2c), resulting in massive accuracy gaps.

	mask branch	AP	AP_{50}	AP_{75}
MLP	fc: 1024→1024→80·28^2	31.5	53.7	32.8
MLP	fc: 1024→1024→1024→80·28^2	31.5	54.0	32.6
FCN	conv: 256→256→256→256→256→80	**33.6**	**55.2**	**35.3**

(e) **Mask Branch** (ResNet-50-FPN): Fully convolutional networks (FCN) *vs.* multi-layer perceptrons (MLP, fully-connected) for mask prediction. FCNs improve results as they take advantage of explicitly encoding spatial layout.

Table 2. **Ablations** for Mask R-CNN. We train on `trainval35k`, test on `minival`, and report *mask* AP unless otherwise noted.

Example: Mask R-CNN paper

Example ablations: “One table, one message”

	AP	AP_{50}	AP_{75}	AP^{bb}	AP^{bb}_{50}	AP^{bb}_{75}
RoIPool	23.6	46.5	21.6	28.2	52.7	26.9
RoIAlign	**30.9**	**51.8**	**32.1**	**34.0**	**55.3**	**36.4**
	+7.3	*+ 5.3*	*+10.5*	*+5.8*	*+2.6*	*+9.5*

(d) **RoIAlign** (ResNet-50-**C5**, *stride 32*): Mask-level and box-level AP using *large-stride* features. Misalignments are more severe than with stride-16 features (Table 2c), resulting in massive accuracy gaps.

Example: *Where does RoIAlign improve over RoIPool?*
(When feature stride is large; for high IoU thresholds)

Example ablations: “One table, one message”

	align?	bilinear?	agg.	AP	AP_{50}	AP_{75}
RoIPool [12]			max	26.9	48.8	26.4
RoIWarp [10]		✓	max	27.2	49.2	27.1
		✓	ave	27.1	48.9	27.1
RoIAlign	✓	✓	max	**30.2**	**51.0**	**31.8**
	✓	✓	ave	**30.3**	**51.2**	**31.5**

(c) **RoIAlign** (ResNet-50-C4): Mask results with various RoI layers. Our RoIAlign layer improves AP by ~3 points and AP_{75} by ~5 points. Using proper alignment is the only factor that contributes to the large gap between RoI layers.

Example: *What makes RoIAlign work?*

(It’s not bilinear vs. nearest; it’s not the aggregation function; it’s avoiding quantization error due to rounding coords)

Supporting claims

- All claims should be supported
  - By an appropriate citation, or
  - By experiment

- Otherwise, statements should be *qualified;* examples:
  - "Increasing X is important for Y" vs.
    "*Intuitively*, increasing X is important for Y…"
    This statement is your intuition (not fact), others may disagree!

  - "Increasing X leads to improved Y…" vs.
    "Increasing X *may* lead to improved Y…"
    Expresses uncertainty or that some conditions may apply

Support all of Your Claims

In principle Mask R-CNN is an intuitive extension of Faster R-CNN, yet constructing the mask branch properly is critical for good results. Most importantly, Faster R-CNN was not designed for pixel-to-pixel alignment between network inputs and outputs. This is most evident in how *RoIPool* [18, 12], the *de facto* core operation for attending to instances, performs coarse spatial quantization for feature extraction. To fix the misalignment, we propose a simple, quantization-free layer, called *RoIAlign*, that faithfully preserves exact spatial locations. Despite being

Evidence

	align?	bilinear?	agg.	AP	AP_{50}	AP_{75}
RoIPool [12]			max	26.9	48.8	26.4
RoIWarp [10]		✓	max	27.2	49.2	27.1
		✓	ave	27.1	48.9	27.1
RoIAlign	✓	✓	max	**30.2**	**51.0**	**31.8**
	✓	✓	ave	**30.3**	**51.2**	**31.5**

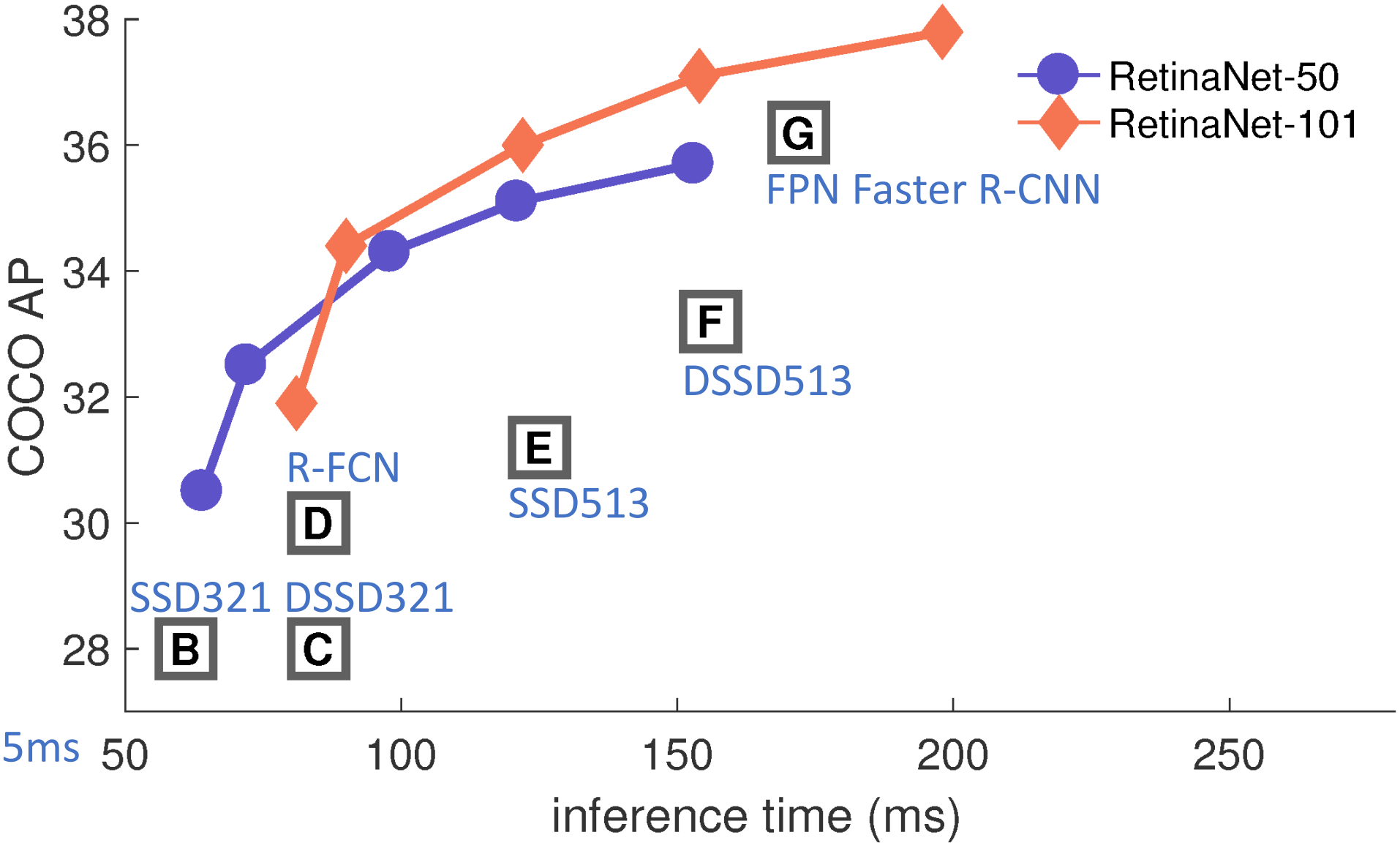
(c) **RoIAlign** (ResNet-50-C4): Mask results with various RoI layers. Our RoIAlign layer improves AP by ~3 points and AP_{75} by ~5 points. Using proper alignment is the only factor that contributes to the large gap between RoI layers.

Example: Mask R-CNN paper

Claim: alignment (defined as not rounding coords) is the key

Table: experimental evidence (bilinear is not key; max vs. ave is not key)

Beware of speed vs. accuracy claims

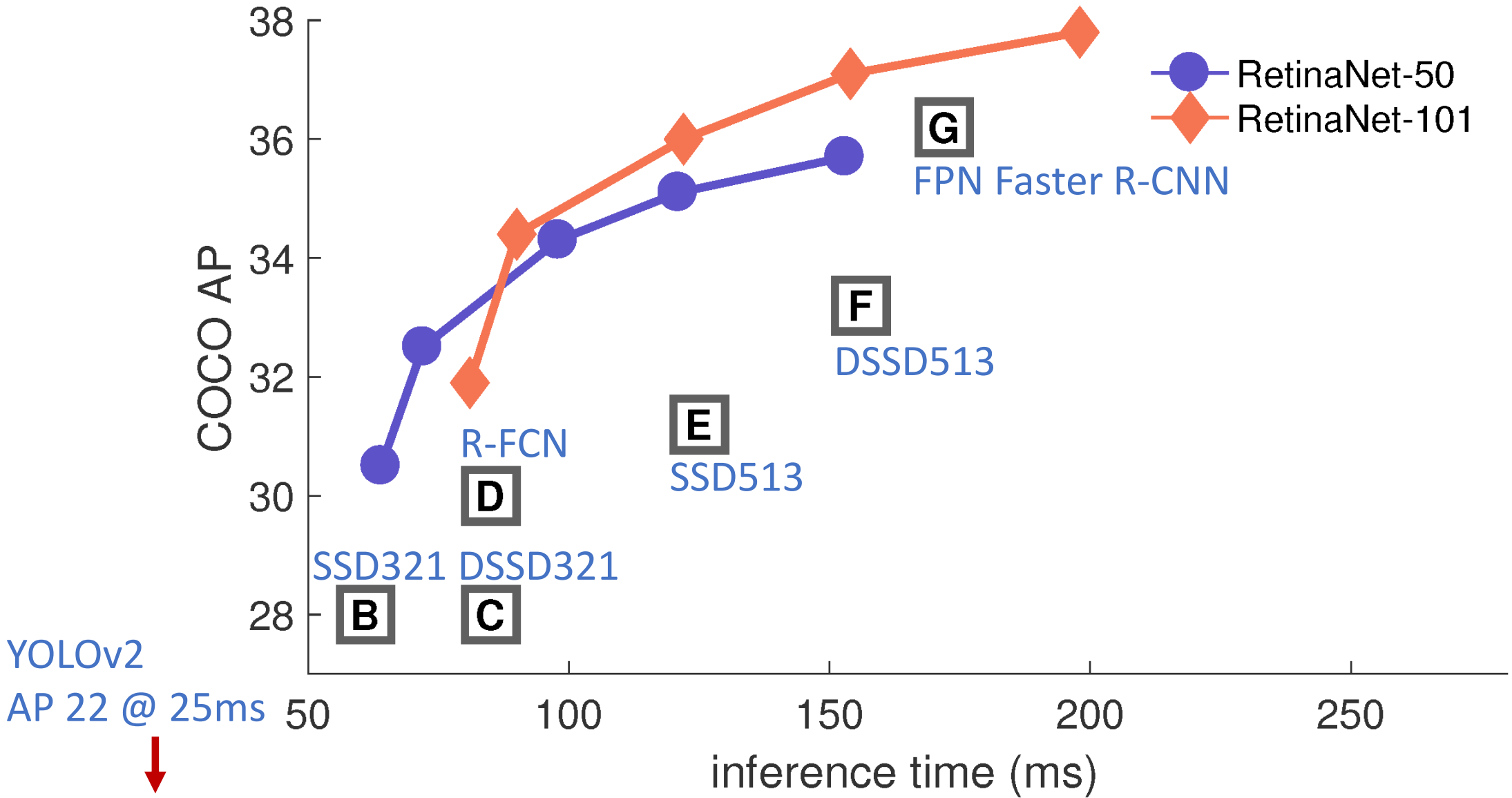


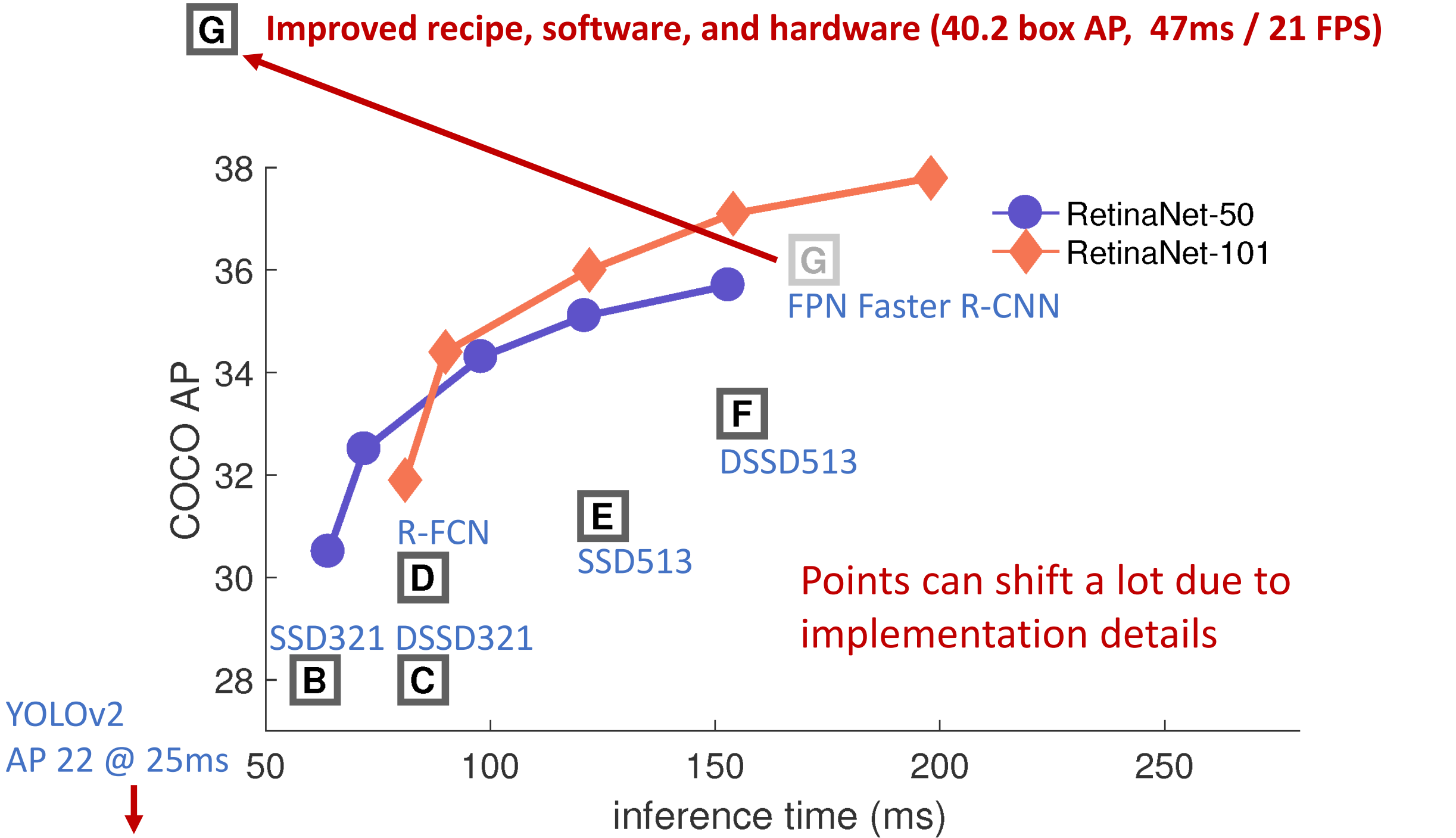
YOLOv2
AP 22 @ 25ms

Beware of speed vs. accuracy claims

Comparisons across publications are often *uncontrolled*

- Accuracy varies with hyper-parameters (often called the “recipe”)
- Speed varies with low-level optimization (perf tuning) and hardware
- Speed varies with inference details (e.g., batching during inference)
- Someone else writes fast code for a living (10 - 100x speedup)
- Therefore, speed/acc. results should be taken with a large grain of salt


38
36
34
32
30
28
COCO AP
50
100
150
200
250
inference time (ms)
RetinaNet-50
RetinaNet-101
G
FPN Faster R-CNN
F
DSSD513
E
SSD513
R-FCN
D
SSD321
DSSD321
B
C
YOLOv2
AP 22 @ 25ms


G
Improved recipe, software, and hardware (40.2 box AP, 47ms / 21 FPS)
38
36
34
32
30
28
COCO AP
50
100
150
200
250
inference time (ms)
RetinaNet-50
RetinaNet-101
G
FPN Faster R-CNN
F
DSSD513
E
SSD513
R-FCN
D
SSD321
DSSD321
B
C
Points can shift a lot due to implementation details
YOLOv2
AP 22 @ 25ms

Beware of speed vs. accuracy claims

They can be ok, if controlled by

- Making training settings as similar as possible
- Making inference settings as similar as possible
- Ensuring low-level optimization fairness
- Using the same hardware for all methods

Implement all methods in one codebase

- … if possible (not always possible)

- There are so many details that matter in detection
- E.g., COCO AP increases ~1% AP (absolute) going from detectron (v1) to detectron2 (same model)
  - A baseline in detectron (v1) is not a valid comparison to a method in detectron2
- Many good codebases now: mmdetection, detectron2
  - No excuses anymore ;)
  - Use the same codebase to the greatest extent possible

Big tables of historical comparisons: A red flag

Method	data	network	pre-train	Avg. Precision, IoU: 0.5:0.95	0.5	0.75
Faster RCNN [27]	trainval	VGGNet	✓	21.9	42.7	-
ION [1]	train	VGGNet	✓	23.6	43.2	23.6
R-FCN [19]	trainval	ResNet-101	✓	29.2	51.5	-
R-FCN$_{\text{multi-sc}}$ [19]	trainval	ResNet-101	✓	29.9	51.9	-
SSD300 (Huang et al.) [11]	< trainval35k	MobileNet	✓	18.8	-	-
SSD300 (Huang et al.) [11]	< trainval35k	Inception-v2	✓	21.6	-	-
YOLOv2 [26]	trainval35k	Darknet-19	✓	21.6	44.0	19.2
SSD300* [21]	trainval35k	VGGNet	✓	25.1	43.1	25.8
DSOD300	trainval	DS/64-192-48-1	✗	29.3	47.3	30.6

Claim: Faster R-CNN (VGG-16 backbone) has 21.9 AP on COCO

Implication: all other methods in the table outperform it

But with today's best practices…

Faster R-CNN (VGG-16 backbone): 35.6 AP

(and 35.2 AP without pre-training)

The field evolves rapidly

- “Old” papers (where old can mean, e.g., less than 10 years even) may contain seemingly weird details or bad practices
  - E.g., Why does the original R-CNN paper use SVMs?
  - E.g., Faster R-CNN was originally trained in four stages (not end-to-end)
- These are artifacts of what was *possible* at the time
  - Compute may have been limited
  - Knowledge was limited
- Never view something in a paper as “the one way” to do something
  - It may be an implementation detail that can and has changed

Recap: Tips about reading detection papers

- It’s really hard, unless you have a lot of experience!
- There are many bad papers out there that may mislead you
  - Uncontrolled comparisons to old results
  - Uncontrolled speed vs. accuracy comparisons
  - Complex methods without careful ablations
  - …
- These tips can guide you what to look for when reading a detection paper

Recap: Learning objectives

Today:

1. What is the object detection task?
2. How is the task formalized?
3. How is the task evaluated?
4. Why is the task difficult?
5. What are the important datasets to know about?
6. Tips for reading research papers on object detection

What's next? Tomorrow: *Fundamentals – Object Detection as a Machine Learning Problem*