Intro to Object Detection

John Sigman

Image Classification

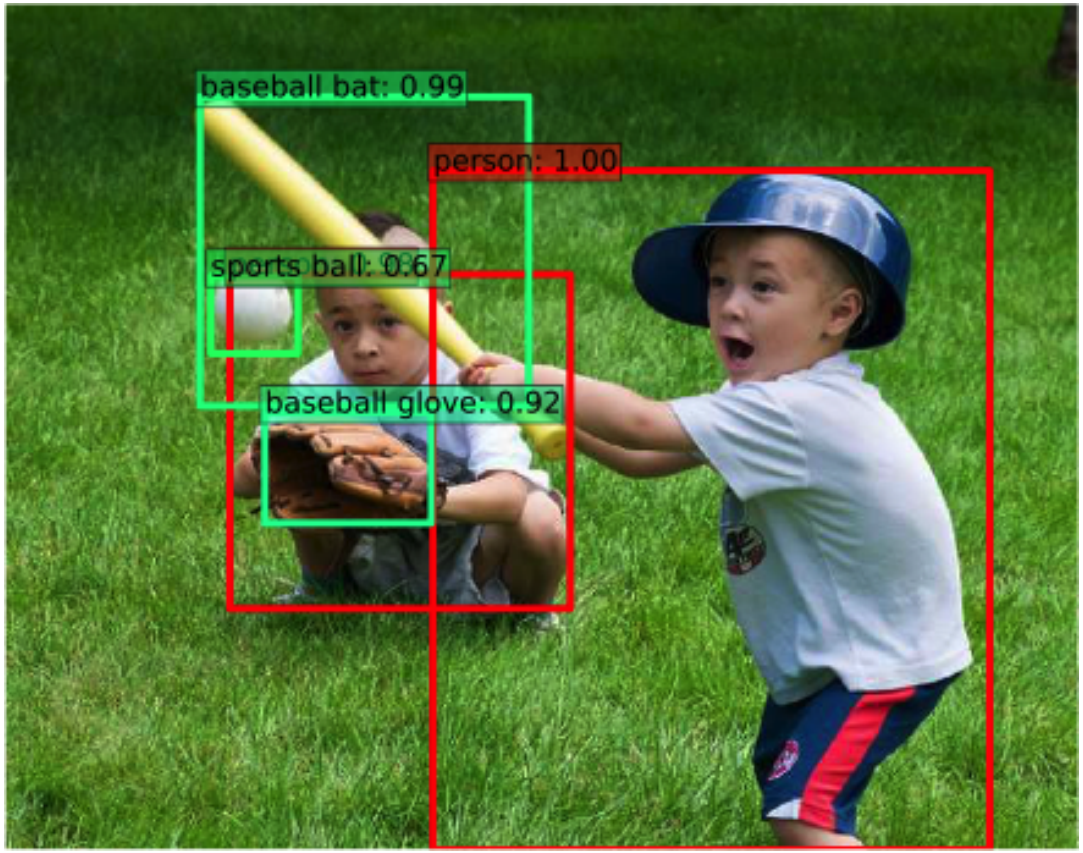
Question: “What is this an image of?”

Answer: “95% probability this is a ballplayer”

Object Detection

Question: “What are all the objects in this object and where are they?”

Answer:



Classification versus detection

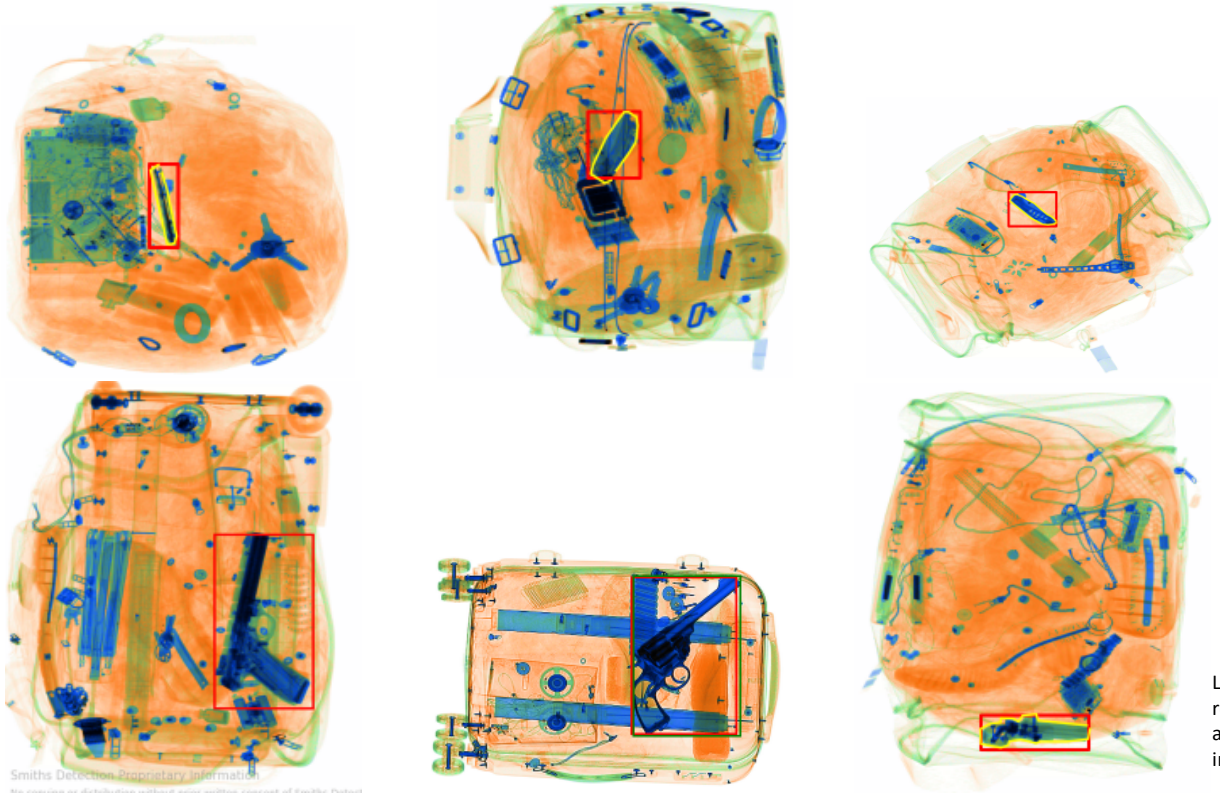
Very related, but different problems.

Many aspects will be shared between the two problems

Self-Driving Cars

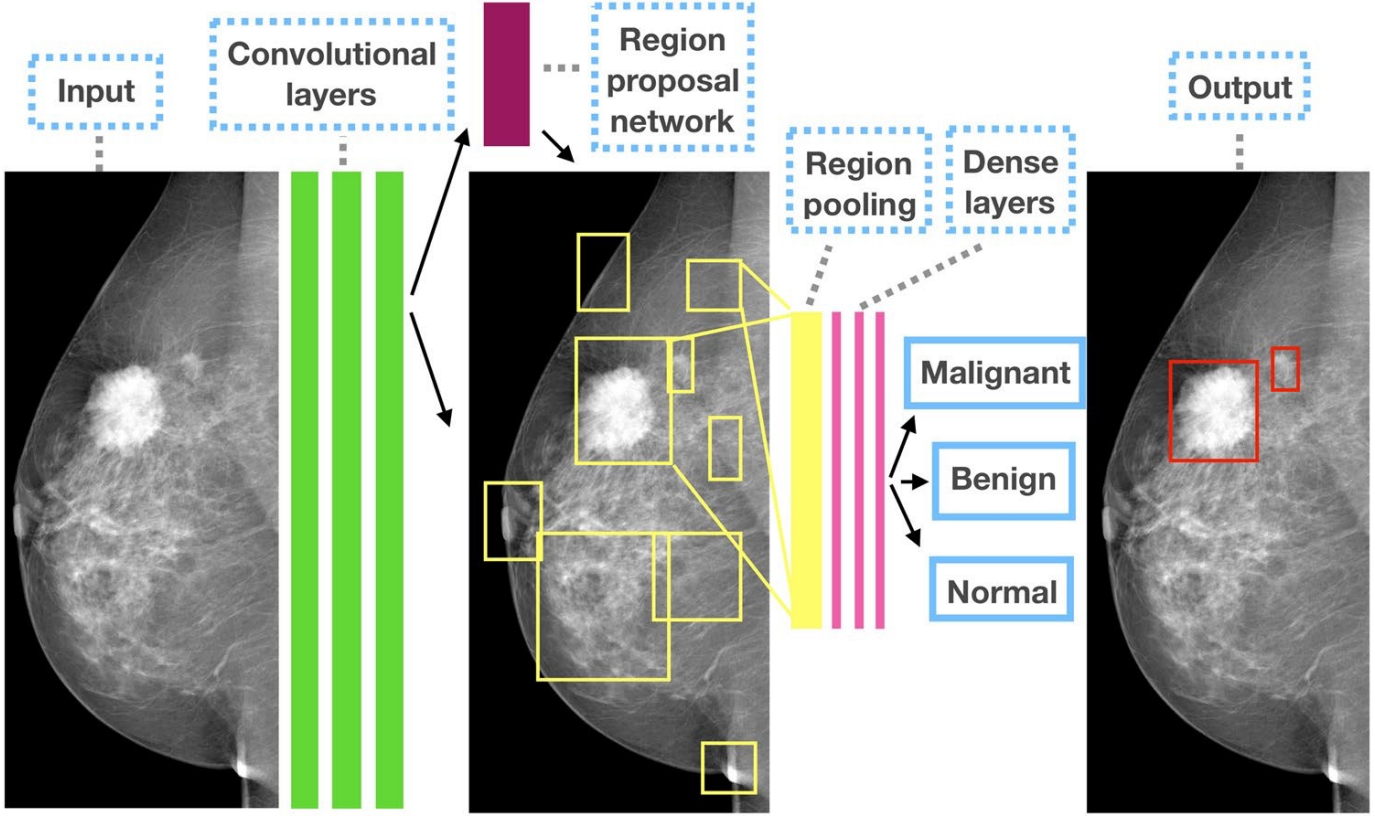
Feng, Rosenbaum, and Dietmayer, “Towards Safe Autonomous Driving: Capture Uncertainty in the Deep Neural Network For Lidar 3D Vehicle Detection”

Application: TSA



Liang et al., “Automatic threat recognition of prohibited items at aviation checkpoints with x-ray imaging: a deep learning approach”.

Application: Medical Imaging



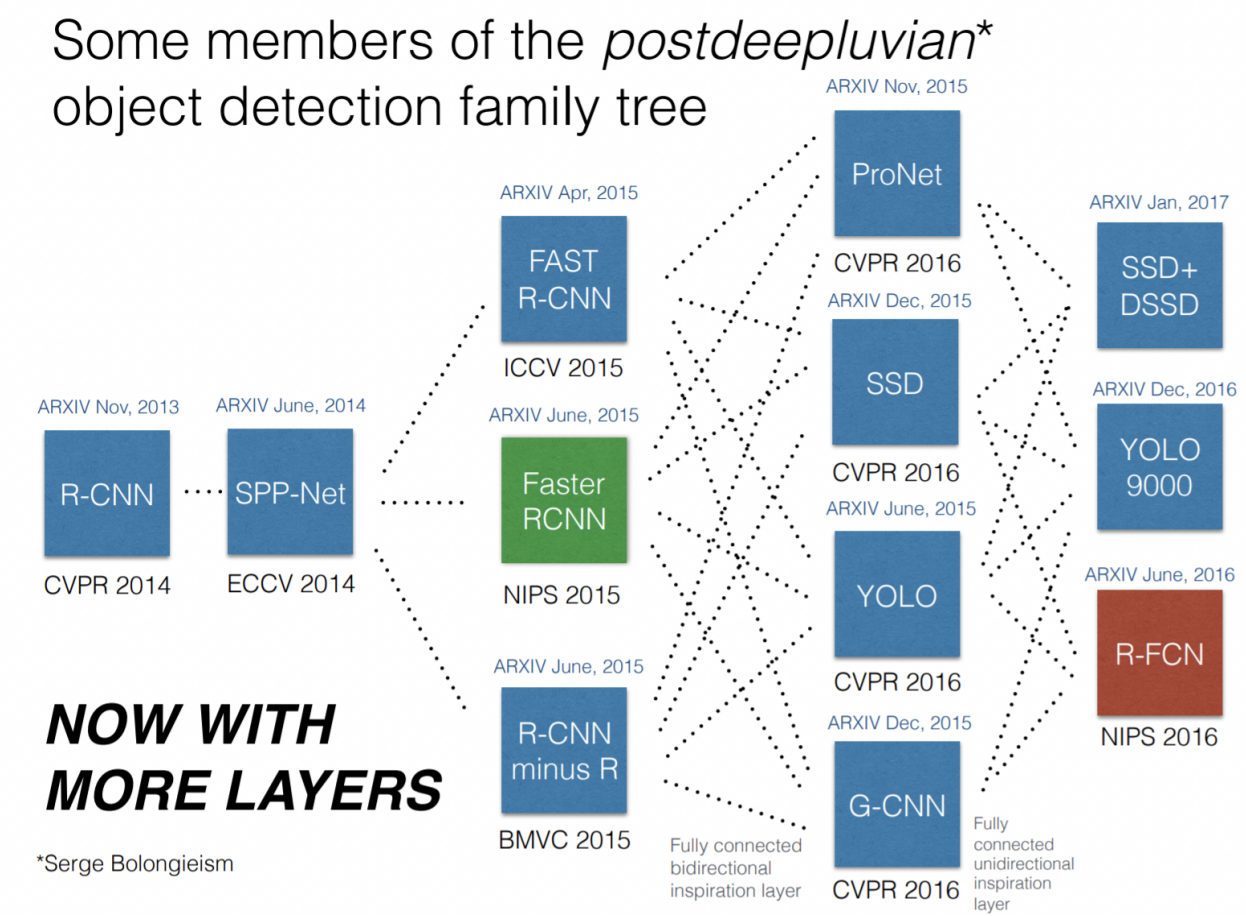
Ribli, Dezső, et al. "Detecting and classifying lesions in mammograms with deep learning." *Scientific reports* 8.1 (2018): 4165.

Pieces of a Deep Algorithm

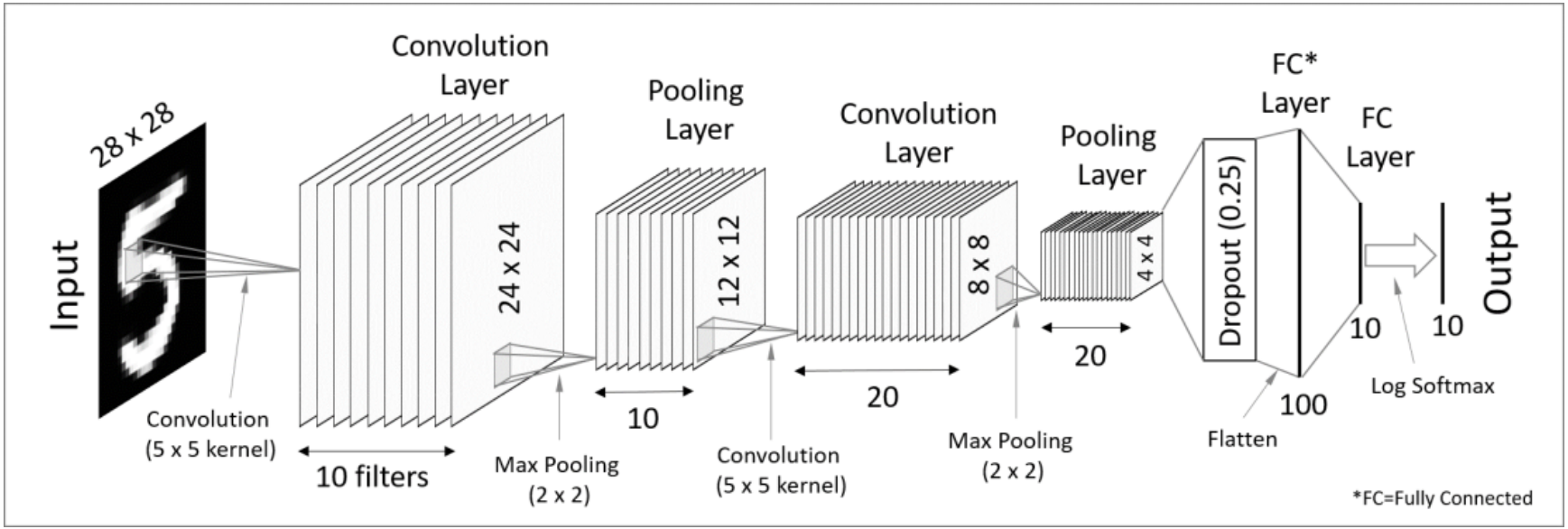
Object detection is very related to image classification. We use similar types of algorithms and related data types.

	Image Classification	Object Detection
Model	CNN Feature Extractor + Softmax Classifier	SSD, Faster R-CNN, e.g.
Data	Imagenet, CIFAR-10, e.g.	COCO, PASCAL VOC, e.g.
Scalar loss function (Objective)	CE loss + Regularization	SSD Loss, Faster R-CNN Loss, (both very complicated)
Optimization Algorithm and Hyperparameters	Stochastic Gradient Descent (SGD), Adam, Momentum, etc.	

Rapidly changing area of research



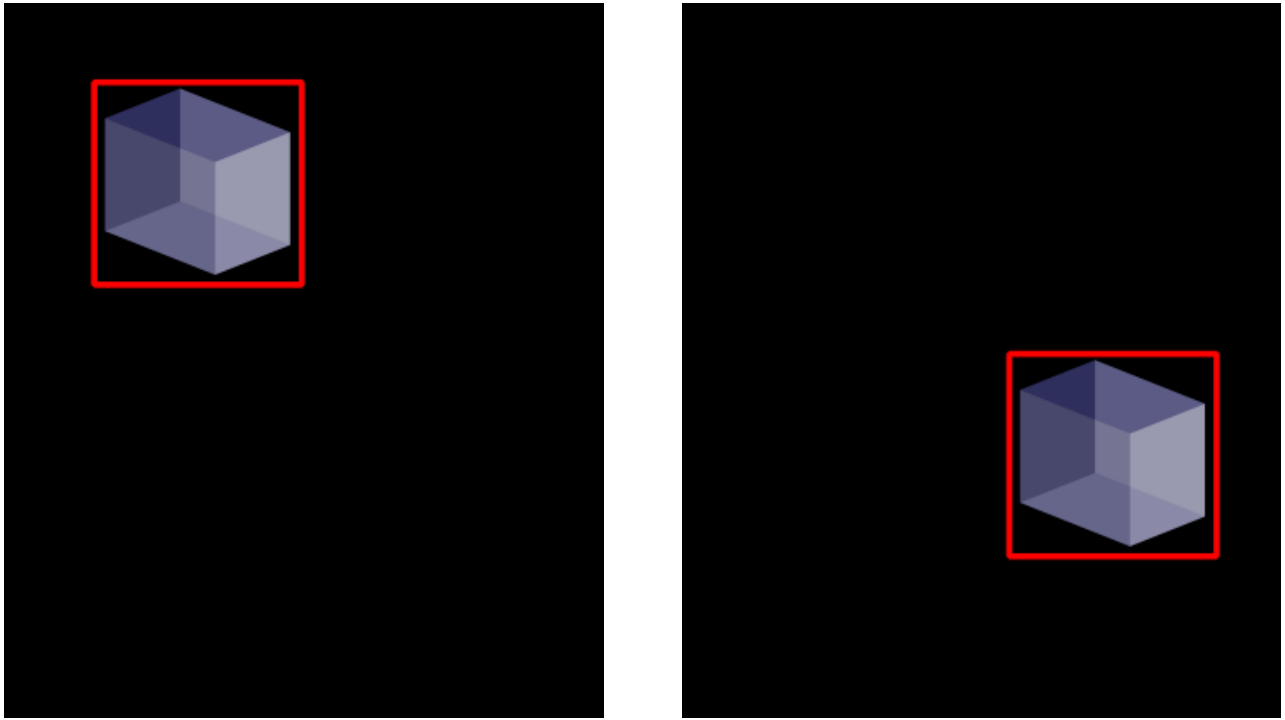
Review: Convolutional Neural Network (MNIST)



What do we want in object detection?

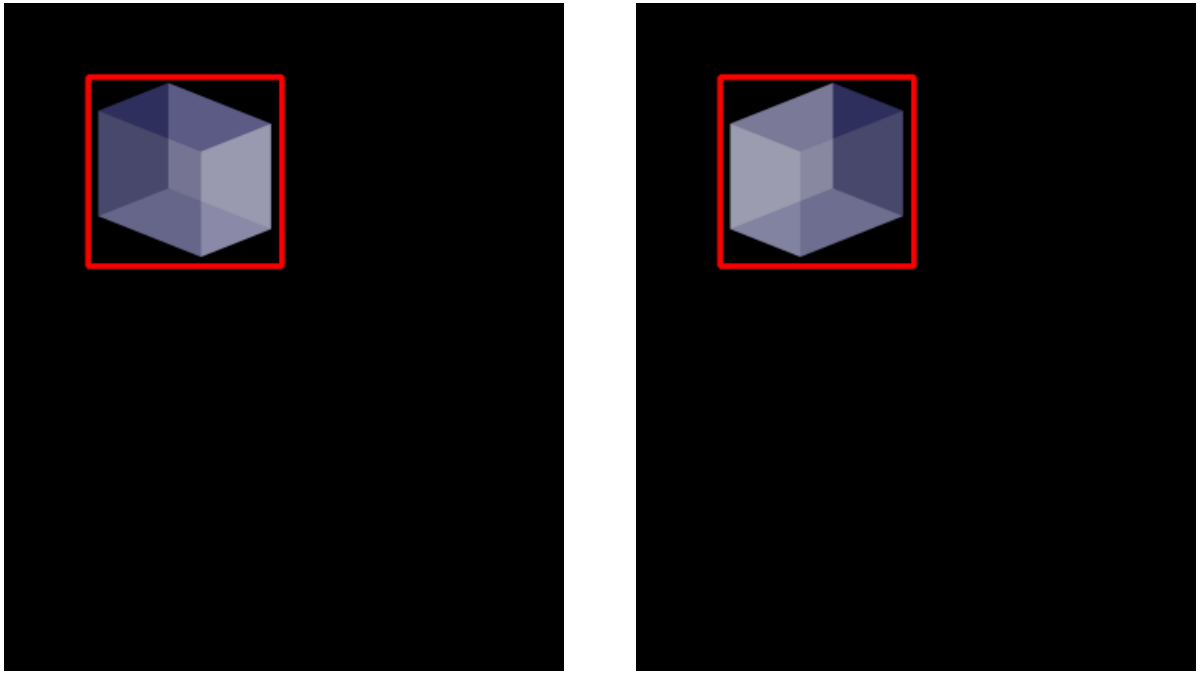
- We have some important properties that are useful in object detection
  - Translation Invariance
  - Scale Invariance
  - Rotation Invariance
- Some are built into the algorithm, and some come from the structure of the dataset

Translation Invariance



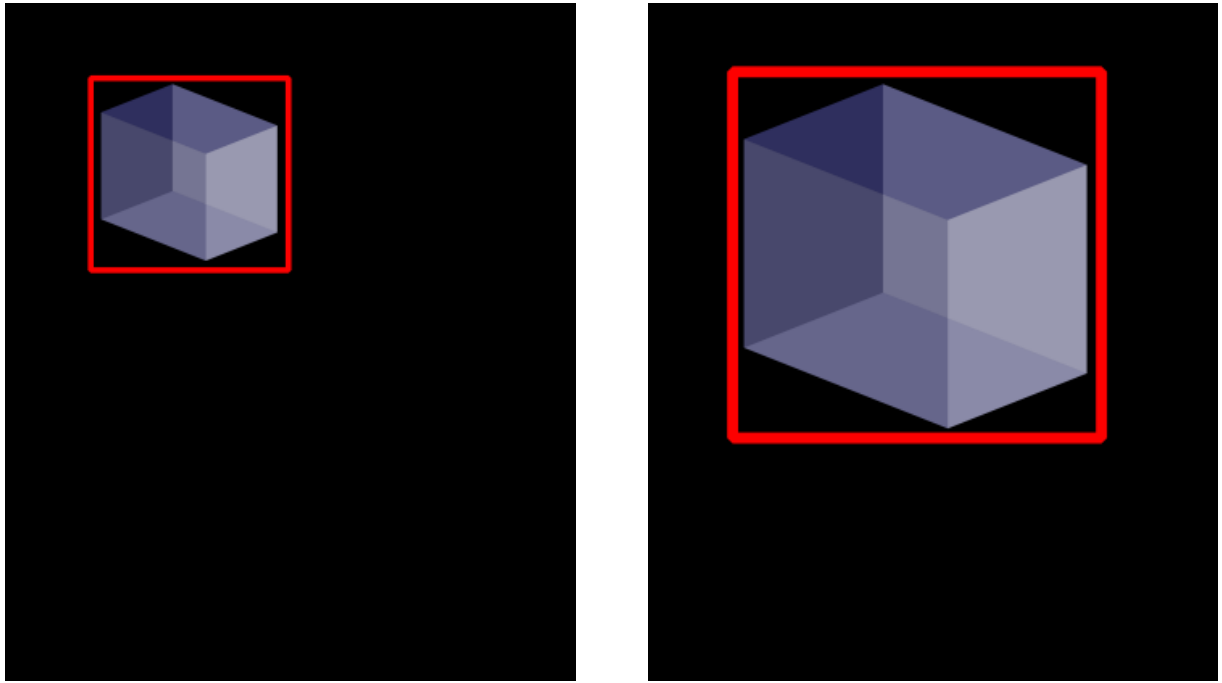
This is addressed with convolutional features…

Rotation Invariance



Will be handled by data augmentation and other techniques in the dataset.

Scale Invariance



Will by handled by data diversity and rescaling feature maps.

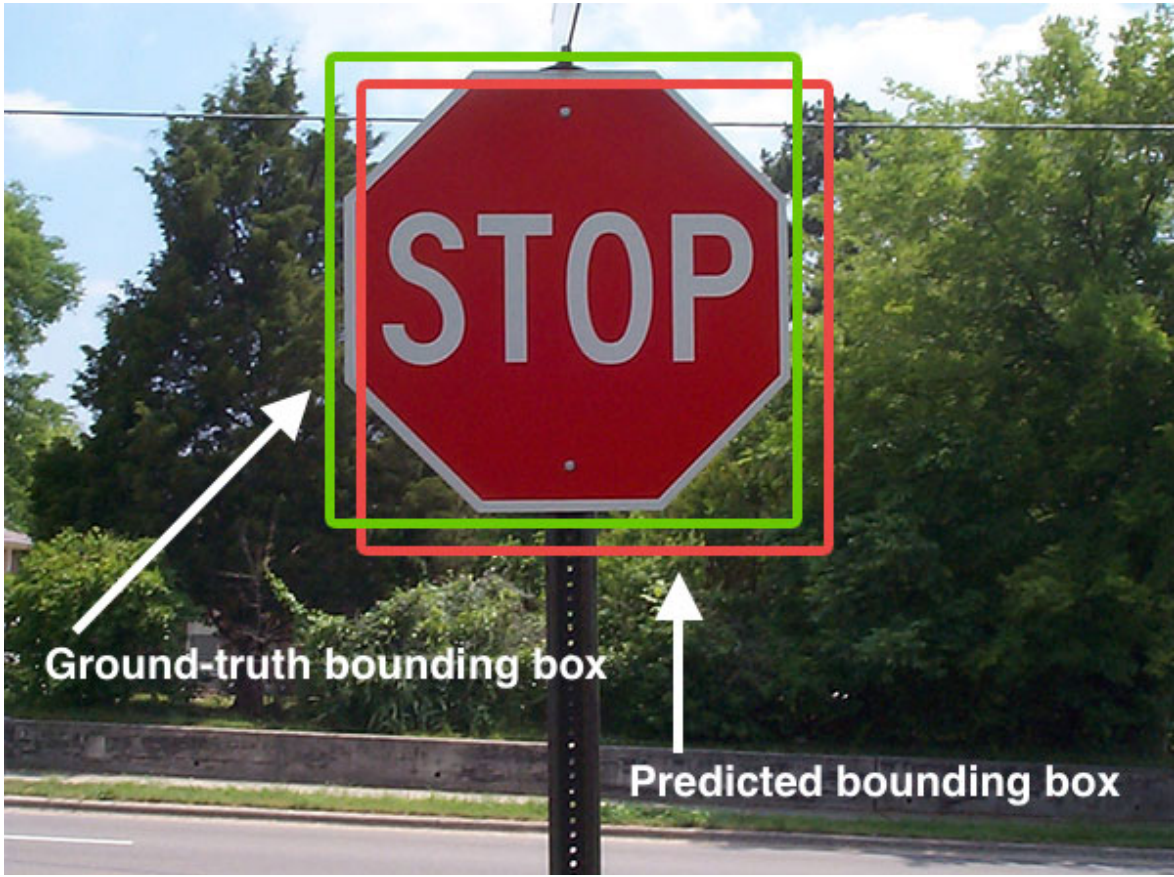
We need a metric to evaluate our model

Classification – did we choose the correct class?

Object detection – is our object correct?

How do we determine if we got the correct location?

We want the bounding box to be good enough.



Wikipedia contributors, Jaccard index — Wikipedia, The Free Encyclopedia.

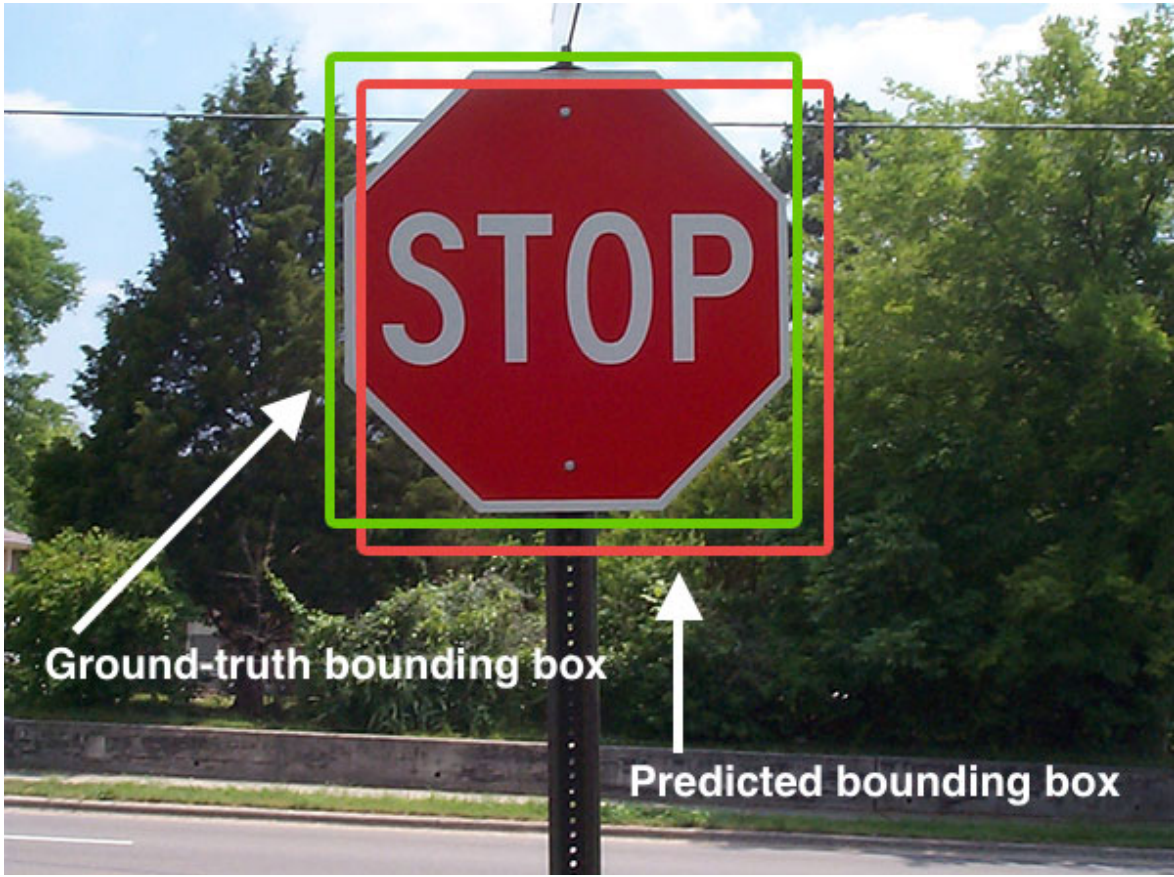
Intersection-over-Union

The intersection-over-union metric is one way of defining this.

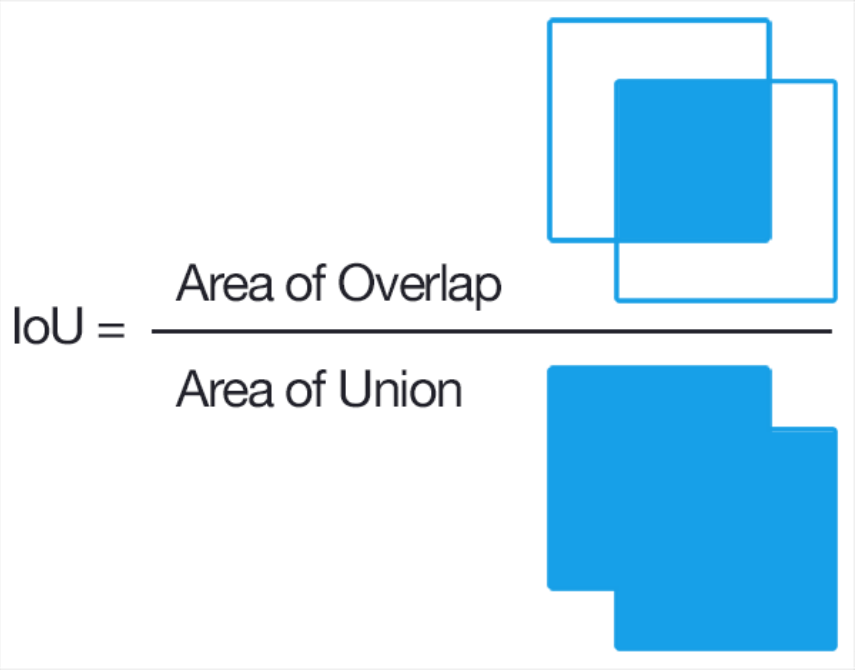
Intersection-the common area covered by the ground-truth bounding box and the predicted bounding box

Union-the total area covered by either the ground-truth bounding box and predicted bounding box.

IoU=Intersection/Union

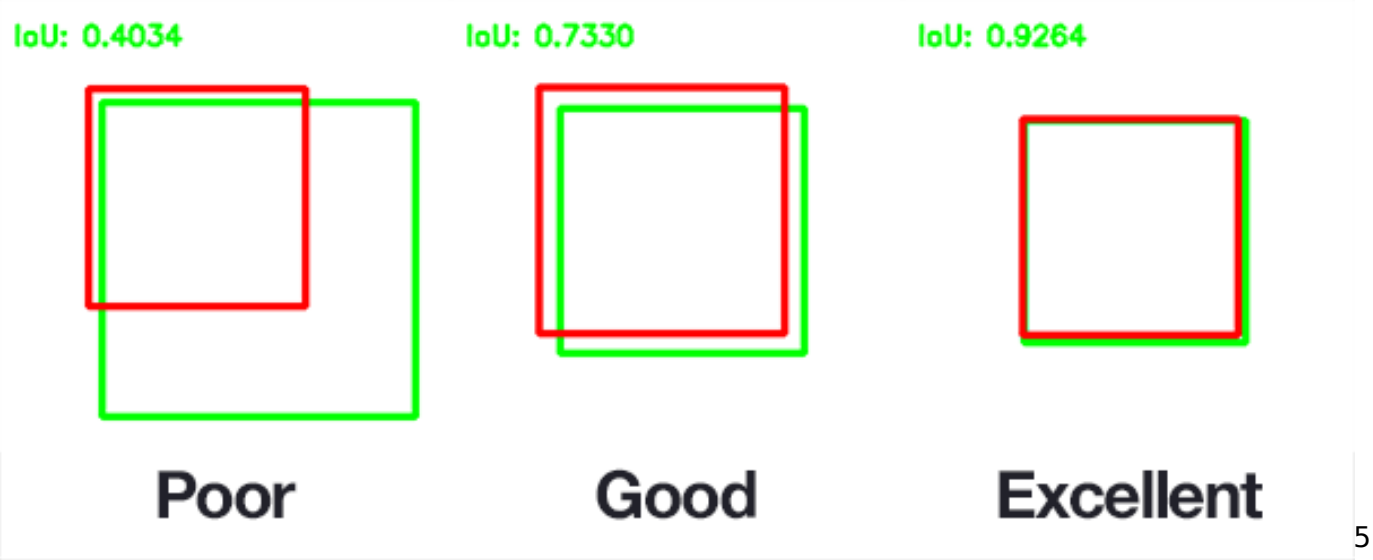


Intersection-over-Union



Wikipedia contributors, Jaccard index — Wikipedia, The Free Encyclopedia.

Intersection-over-Union



5

Wikipedia contributors, Jaccard index — Wikipedia, The Free Encyclopedia.

From Wikipedia

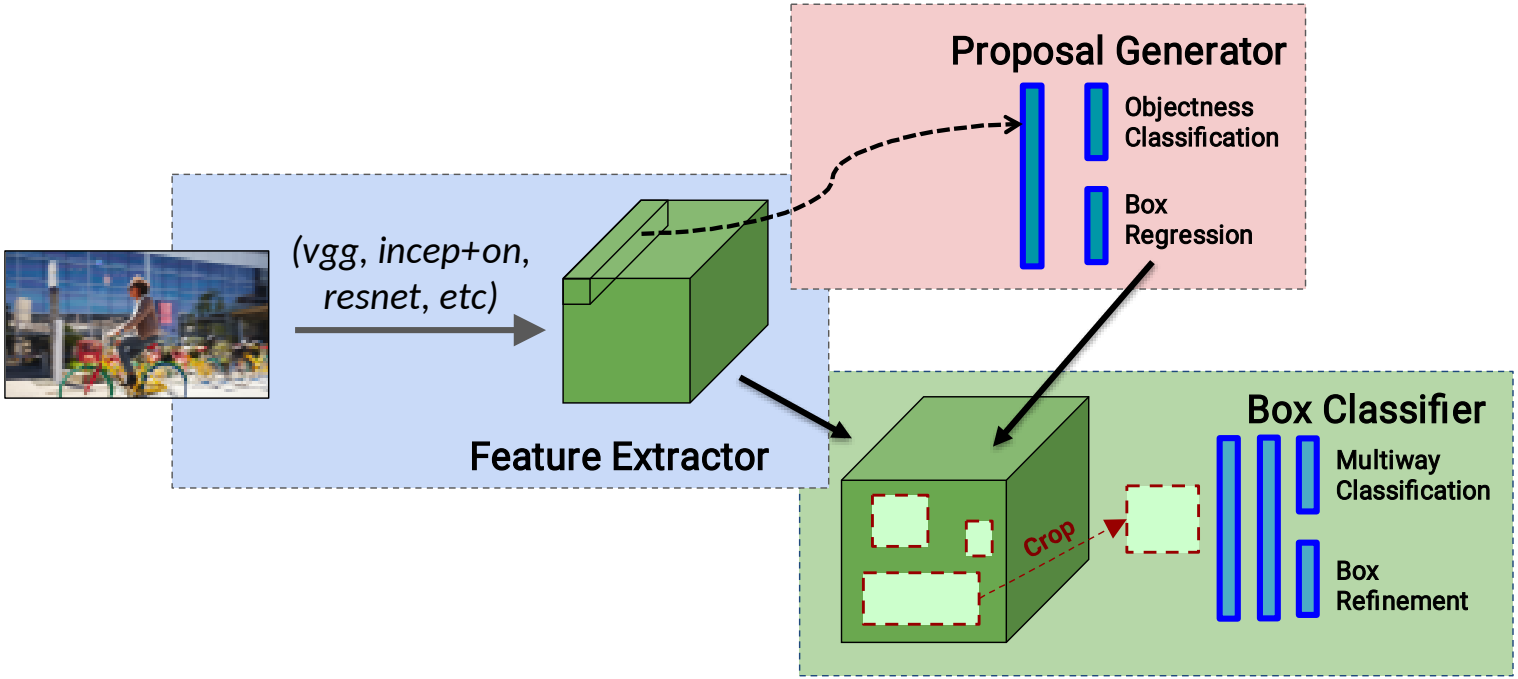
Two Types of Detectors

	One-Stage	**Two-Stage**
Examples	Single-Shot Multibox Detector (SSD), YOLO	Faster R-CNN
Speed	Faster (59 fps[6])	Slower (5 fps[7])
Accuracy	Less accurate ($\approx$.24 mAP on COCO)	More accurate ($\approx$.37 mAP on COCO)

6Liu et al., "Ssd: Single shot multibox detector".
7Ren et al., "Faster r-cnn: Towards real-time object detection with region proposal networks" .

HOW DO CONVOLUTIONAL FEATURES LEAD TO OBJECT DETECTION?

Faster RCNN



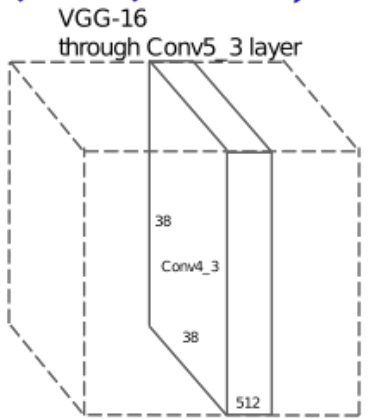
Huang et al., “Speed/accuracy trade-offs for modern convolutional object detectors”

Ren et al

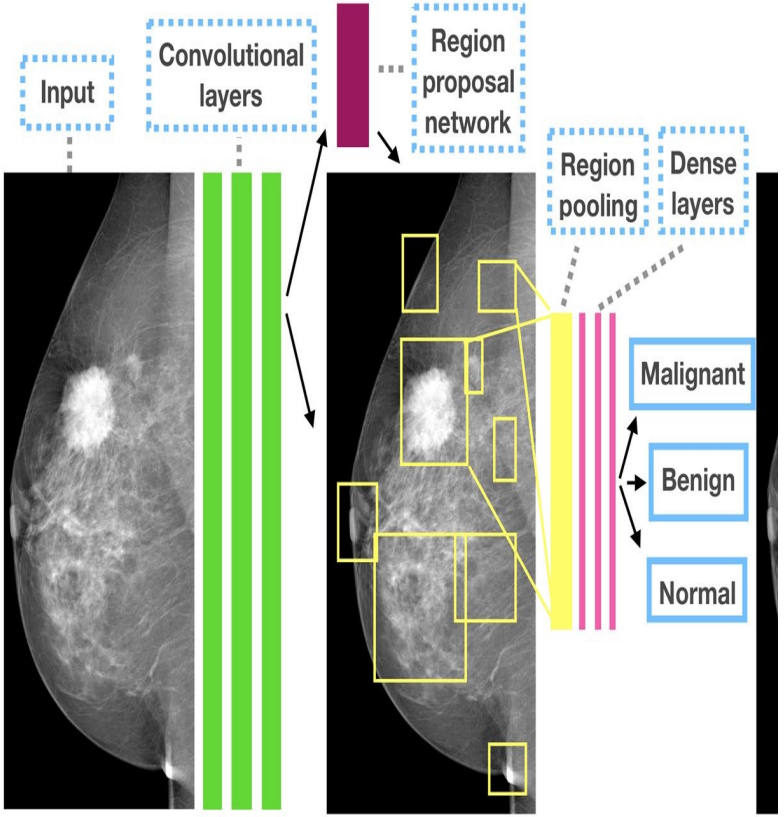
Stage 1, Step 1: Feature Extraction

Input Image
(479,479,3)

Feature Maps
(38,38,512)



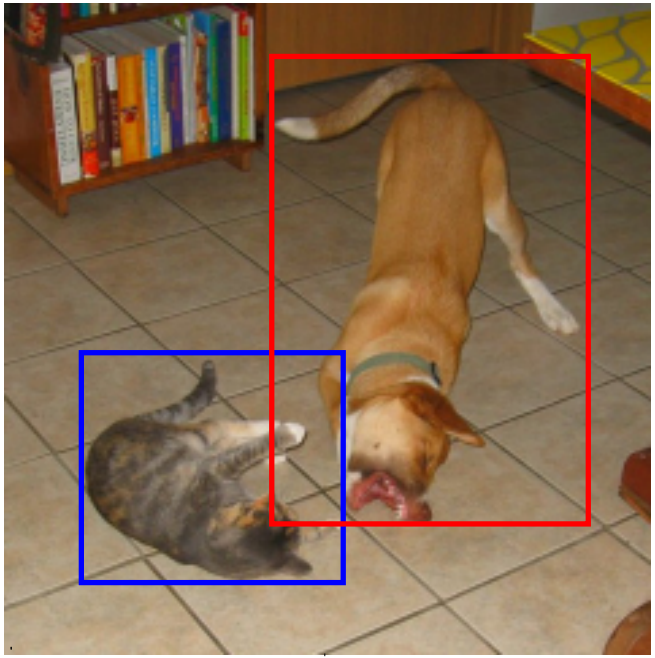
Stage 1, Step 2: Proposal Generation



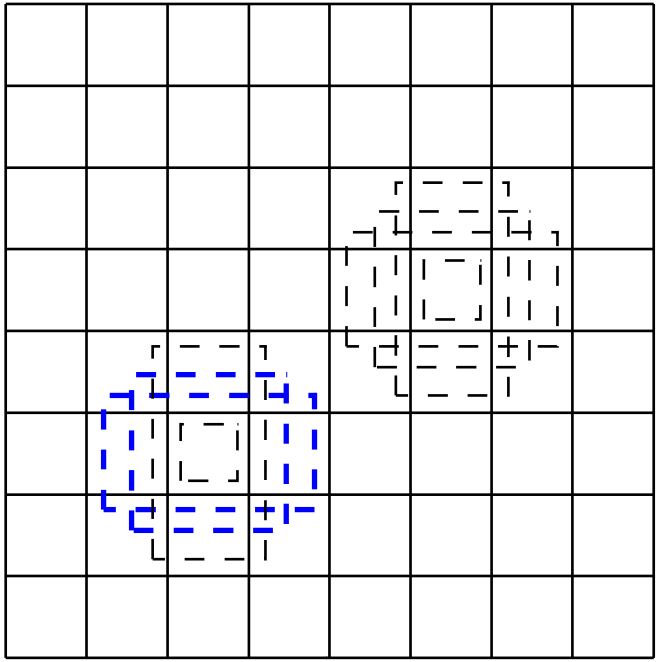
Ribli, Dezső, et al. "Detecting and classifying lesions in mammograms with deep learning." *Scientific reports* 8.1 (2018): 4165.

Anchor Boxes

Image

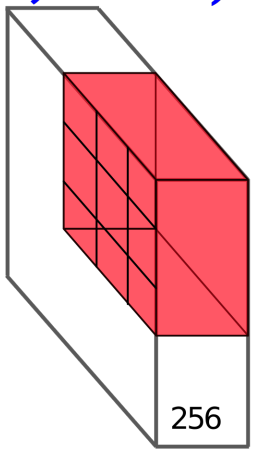


Features

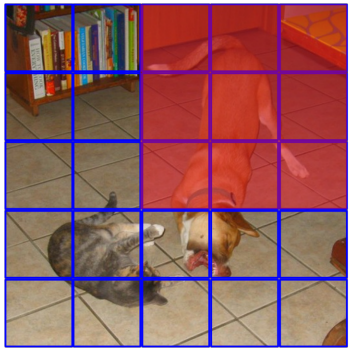


Consider only one section of an image at a time

Feature Maps (5,5,256)

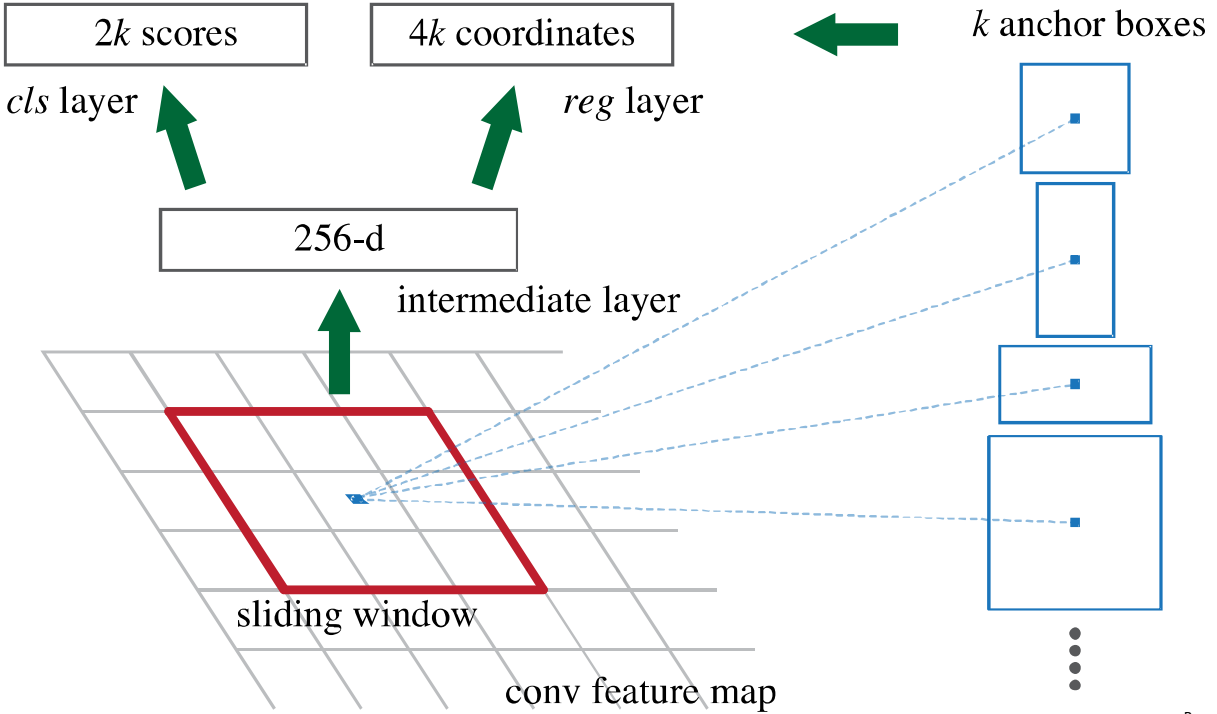


Conv: 1x1x128
Conv: 3x3x256-s2



Sliding Convolutional Kernel (3,3,256)

Region Proposal Network



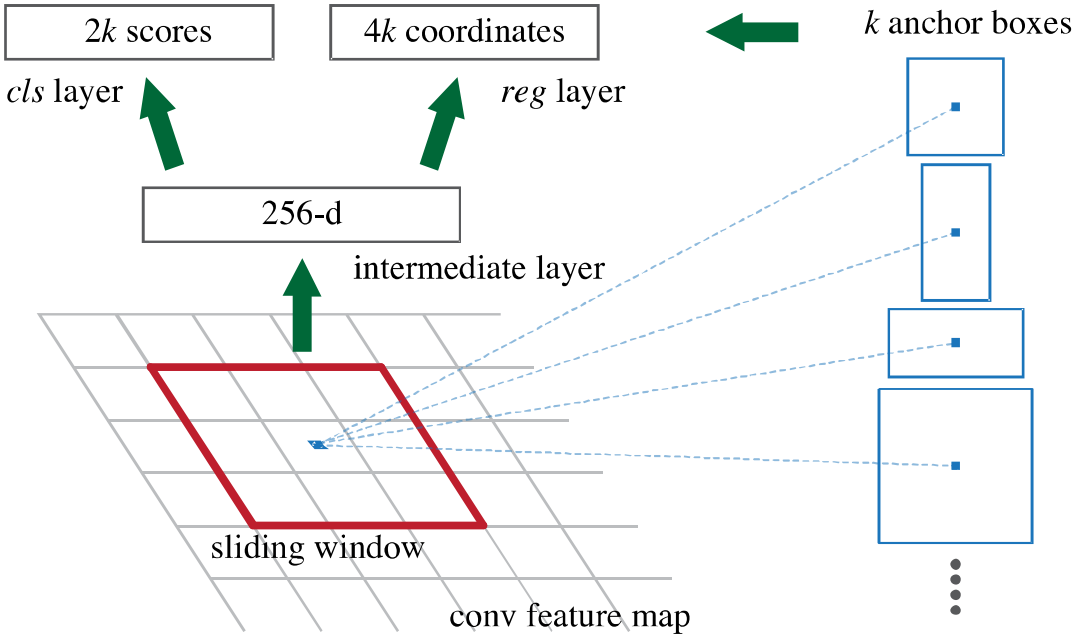
Ren et al., "Faster r-cnn: Towards real-time object detection with region proposal networks"

Region proposal network

Using a sliding window on our convolutional filters, we first produce an intermediate layer of features.

We predict $4k$ coordinates per bound box.

What does this mean?



Ren et al., "Faster r-cnn: Towards real-time object detection with region proposal networks"

How do we predict bounding boxes?

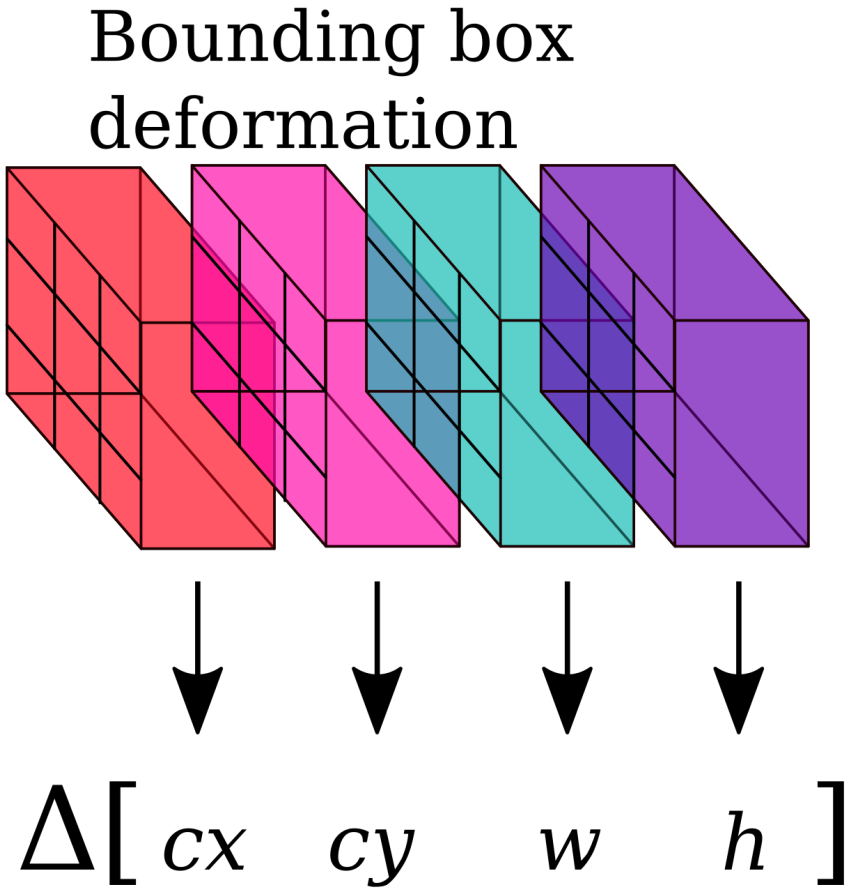
Start with a “base” bounding box.

Predict four terms:

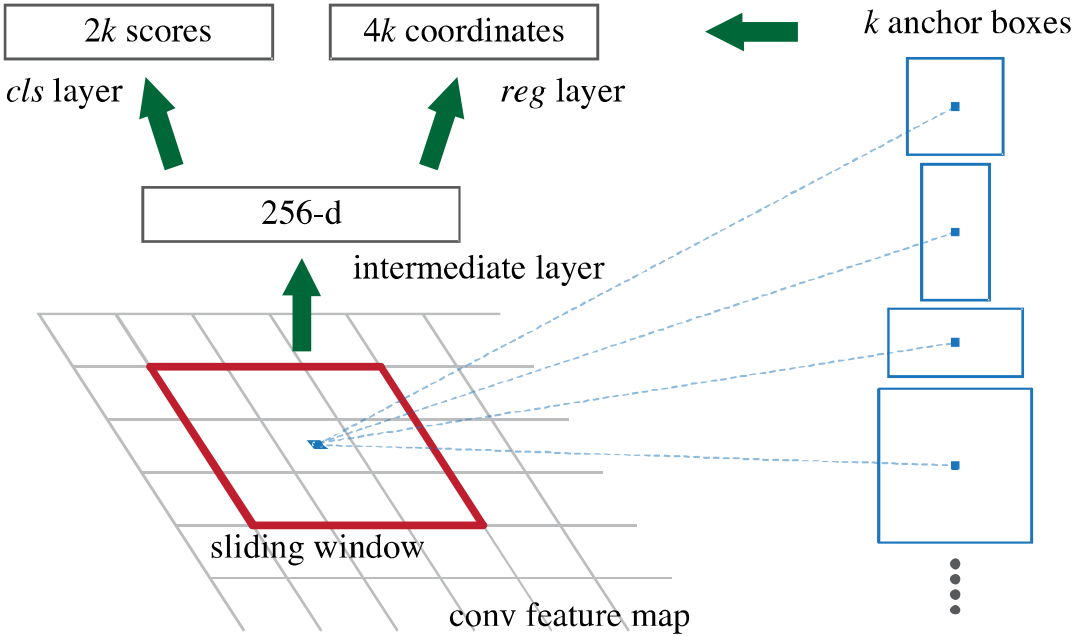
Center (x,y) of the bounding box

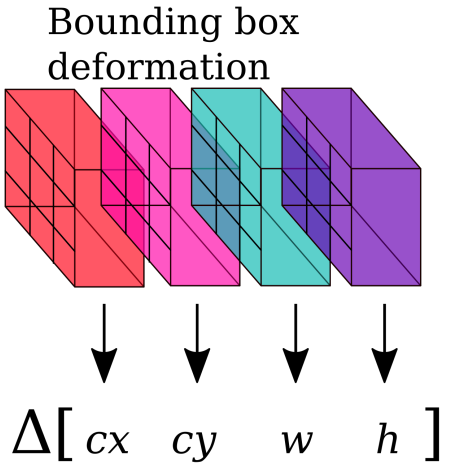
Width of the bounding box (typically a multiple, 1x is the default box)

Height of the bounding box (typically a multiple, 1x is the default box)



How do we predict bounding boxes?





Mathematically, what do we predict?

Let a denote the "anchor" box and gt denote ground truth boxes.

Then we can define our predictions as:

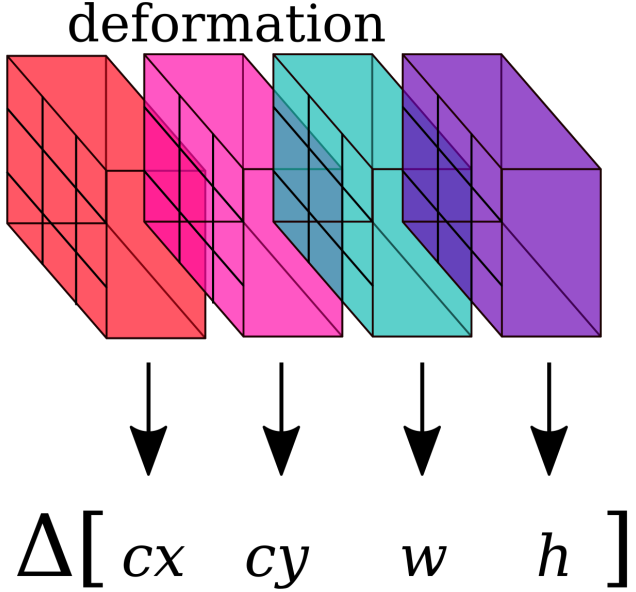
$$tx = \frac{cx - x_a}{w_a}$$

$$ty = \frac{cy - y_a}{h_a}$$

$$tw = \log\left(\frac{w}{w_a}\right)$$

$$th = \log\left(\frac{h}{h_a}\right)$$

Can calculate same quantities with respect to ground truth (necessary for creating loss functions.

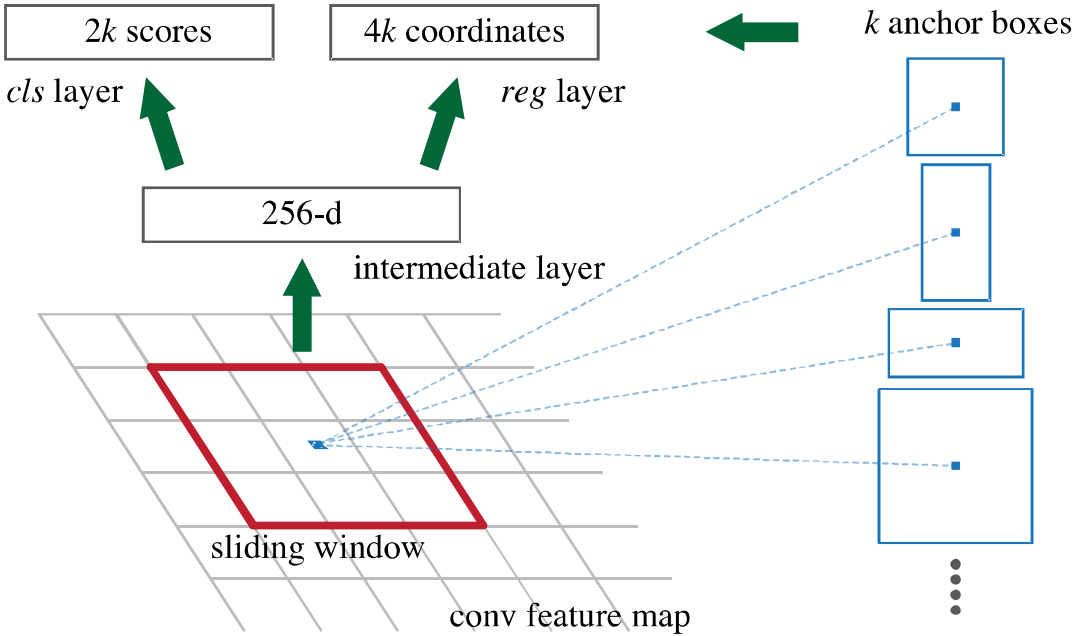


What do our scores do?

Our scores simply return probabilities of “box” or “no box.”

Box means that this is a real bounding box (i.e. we’re confident that we have a real box)

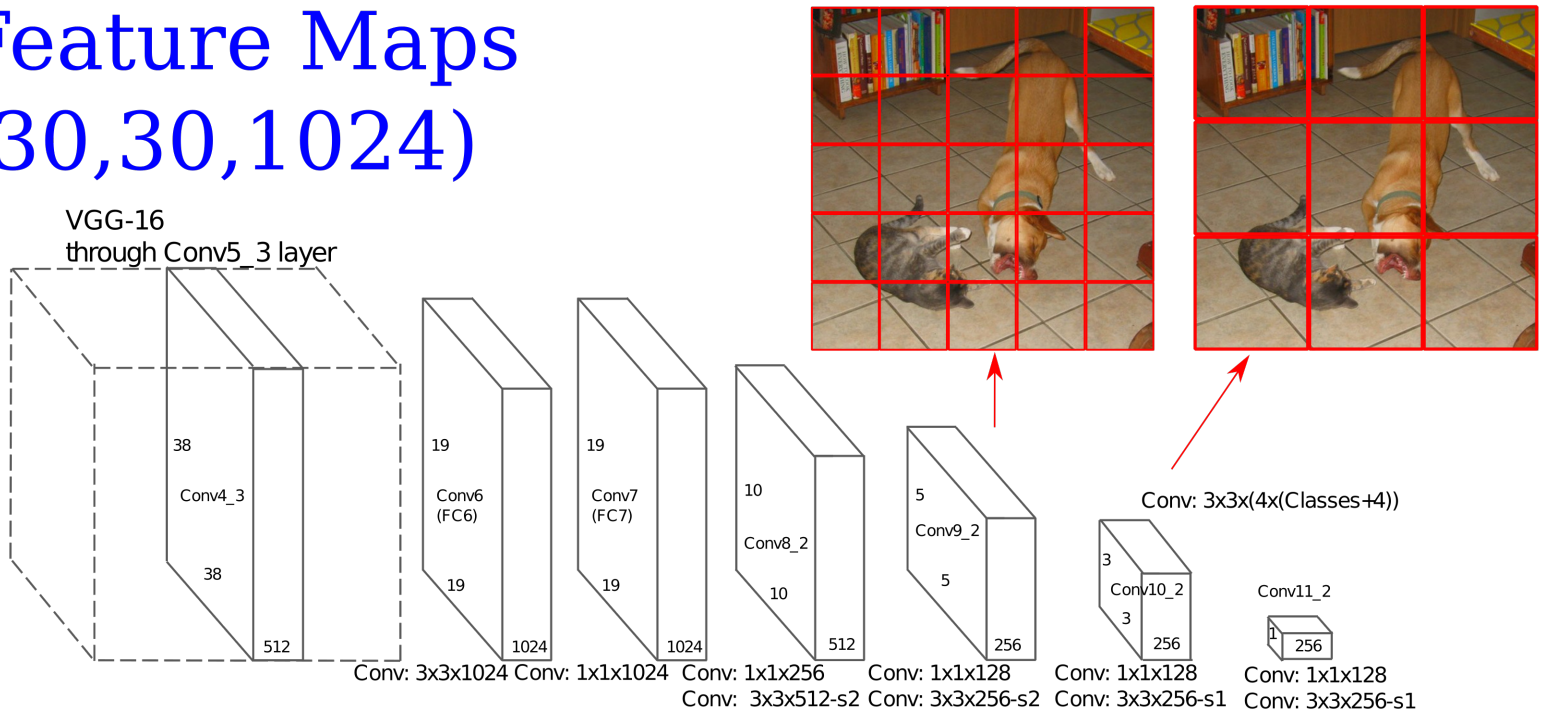
Want to ignore everything else. There are *lots* of incorrect proposals…



Ren et al., “Faster r-cnn: Towards real-time object detection with region proposal networks”

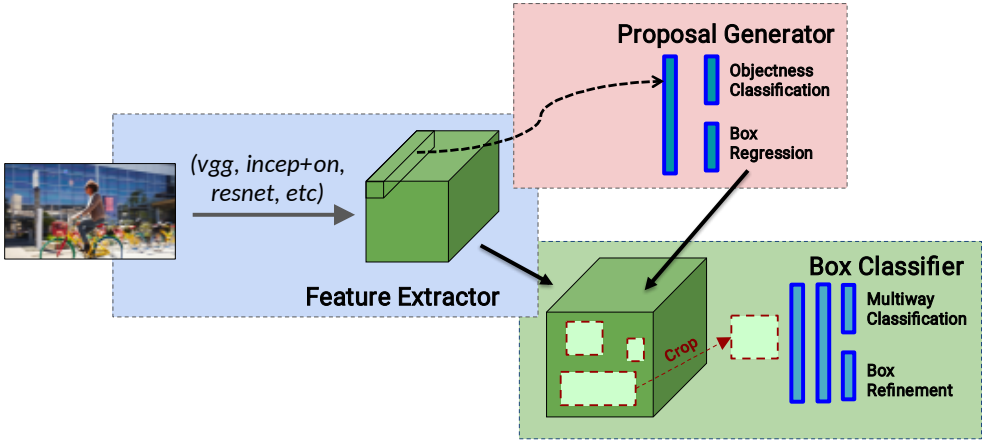
Can Consider Multiple Scales

Feature Maps (30,30,1024)

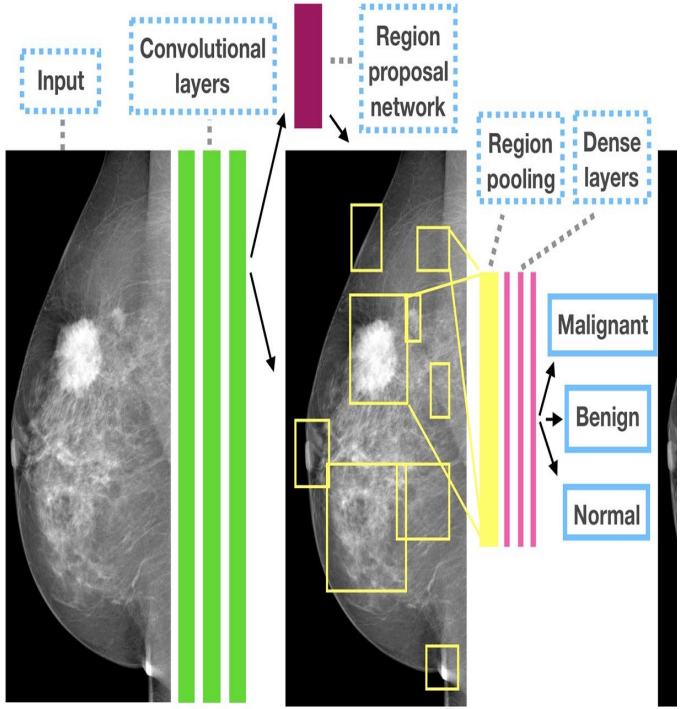


Stage 2: Classification

Second stage object classifier



CNN Features

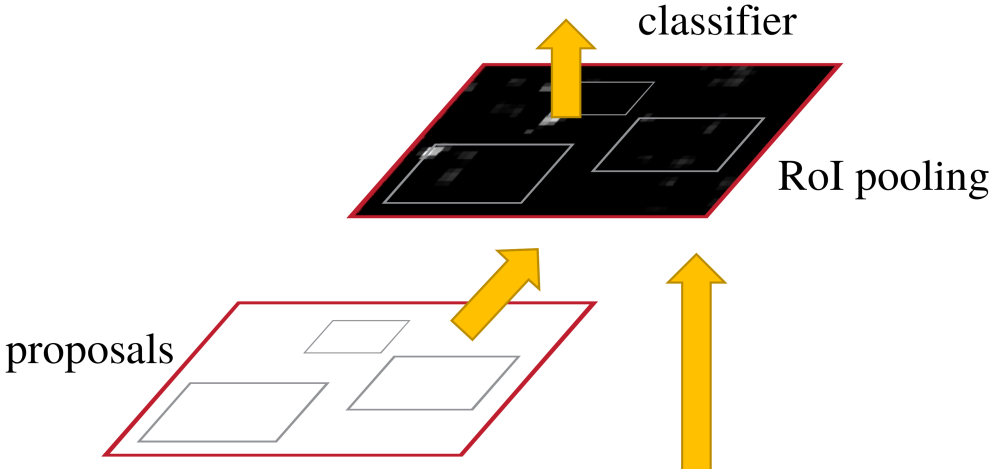


Ren et al., "Faster r-cnn: Towards real-time object detection with region proposal networks"

How to we determine what's in a box?

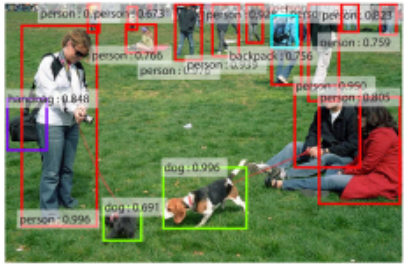
"Region-of-Interest (ROI) pooling" is a critical step to reduce complexity of later bounding boxes. It forces all boxes to have the same feature dimensionality.

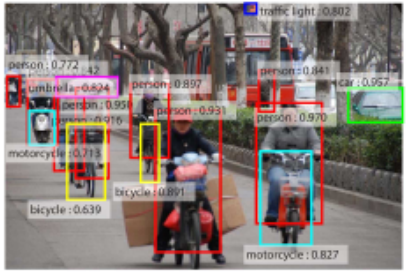
Vector of features for the box from the CNN goes through a classifier (typically an MLP) to predict one of the classes or ***background***.

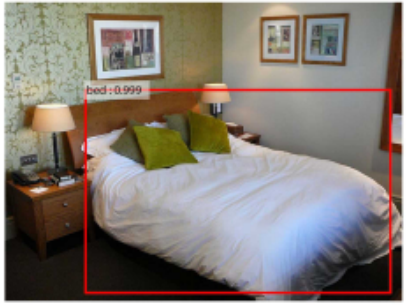


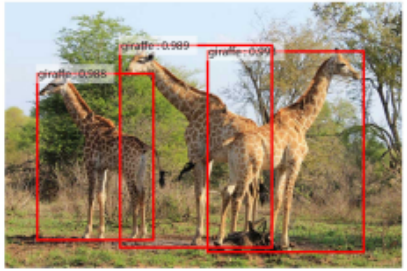
Ren et al., "Faster r-cnn: Towards real-time object detection with region proposal networks"

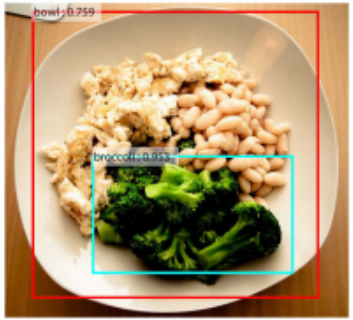
Sample Detections











How can this be learned?

- Object detection adds *a lot* of complexity compared to a CNN
- Many practical details become important for effective training
- I will go over key details; read the papers if you go to implement this yourself

Positive and Negative Boxes

- For evaluation, we consider the following a correct box:

  >=0.5 IOU: Correct

  <0.5 IOU: Incorrect

- For training:

  Positive: .7<= IOU:

  Ignore: .3 <= IOU < .7

  Negative: IOU <= .3

Training Loss for the RPN

- Define
  - p_i: probability anchor i is an object
  - p_i^*: indicator is object is positive (i.e. 0/1)
  - t_i: vector containing coordinates for bounding box i
  - t_i^*: coordinates of ground truth bounding box i
- The total loss for the model is given as:

$$L(\{p_i\}, \{t_i\}) = \frac{1}{N_{cls}} \sum_i L_{cls}(p_i, p_i^*)$$

$$+\lambda \frac{1}{N_{reg}} \sum_i p_i^* L_{reg}(t_i, t_i^*).$$

Training Loss for the RPN

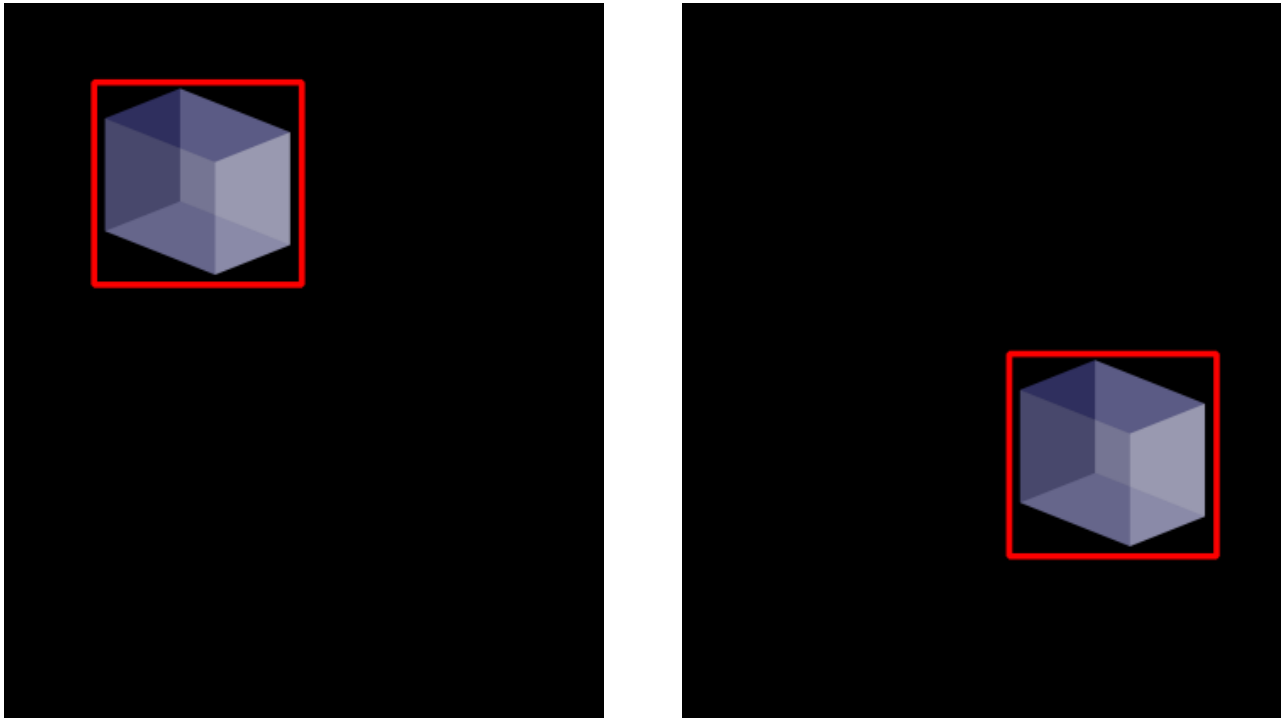
$$L(\{p_i\},\{t_i\}) = \frac{1}{N_{cls}}\sum_i L_{cls}(p_i, p_i^*)$$
$$+\lambda\frac{1}{N_{reg}}\sum_i p_i^* L_{reg}(t_i, t_i^*).$$

- The first term is a cross-entropy loss on the object predictor (i.e. object/not object)
- Second term is on the box dimensions, which is an l1 loss
- Given the boxes, it's just a multi-class classification (often treated with a separate loss given just the boxes.

Let’s go back to our goals:

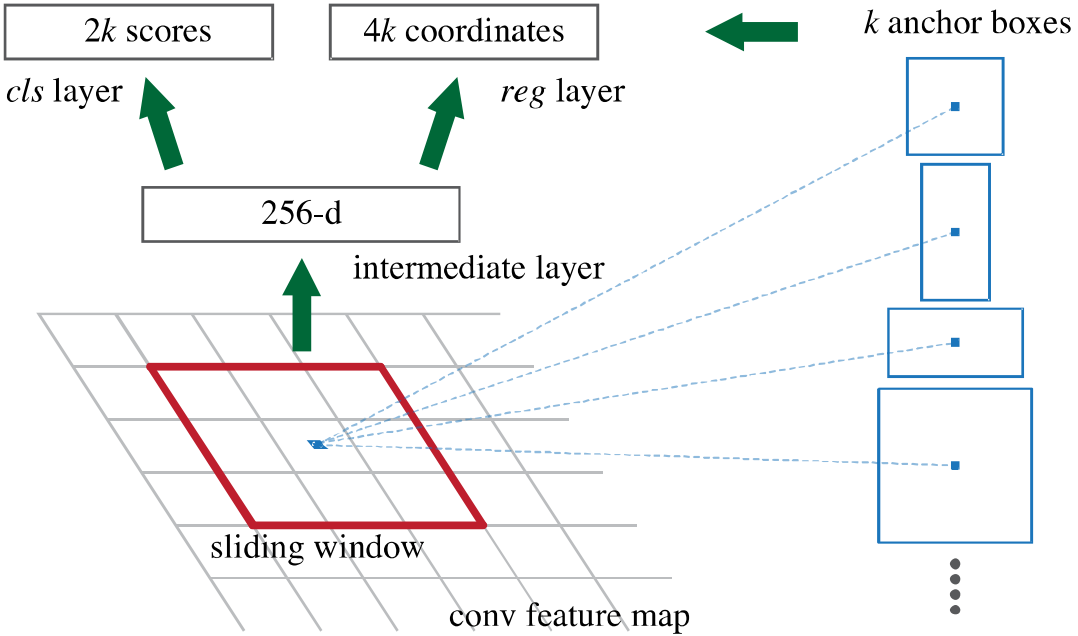
- We have some important properties that are useful in object detection
  - Translation Invariance
  - Scale Invariance
  - Rotation Invariance
- Some are built into the algorithm, and some come from the structure of the dataset

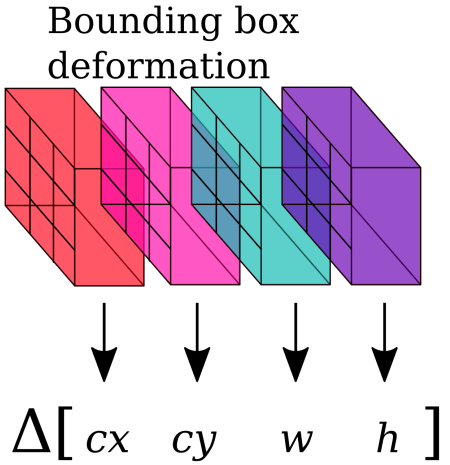
Translation Invariance



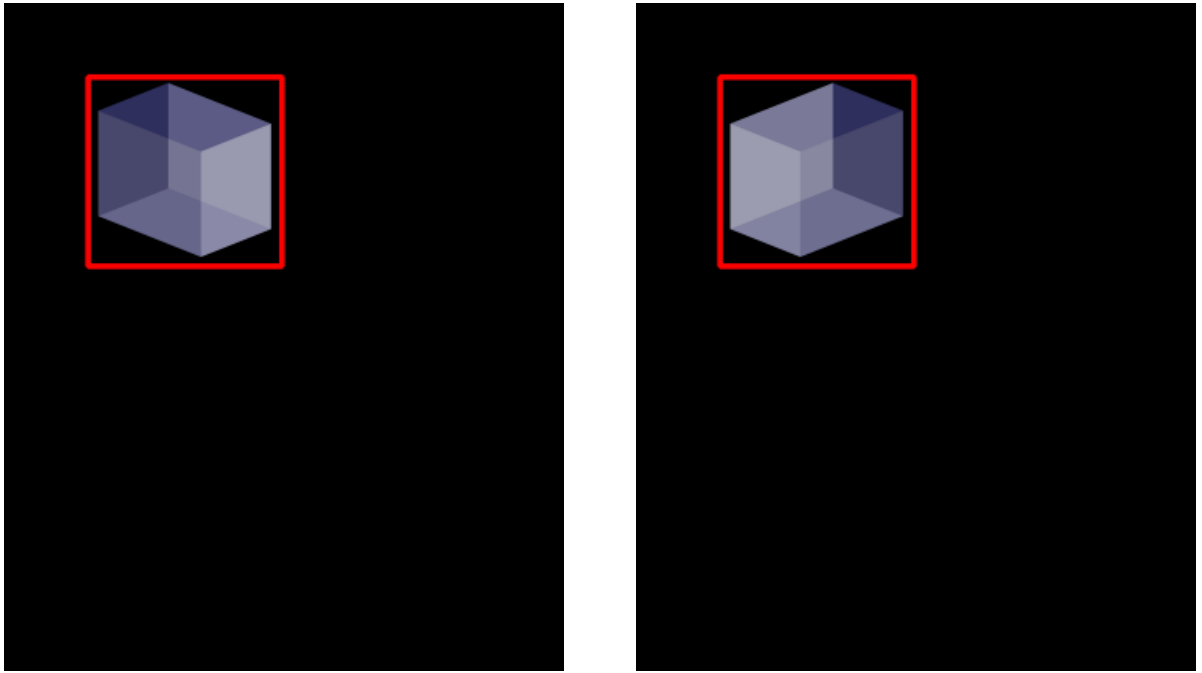
This is addressed with convolutional features…

Translation invariance comes from the sliding windows (same as conv.)



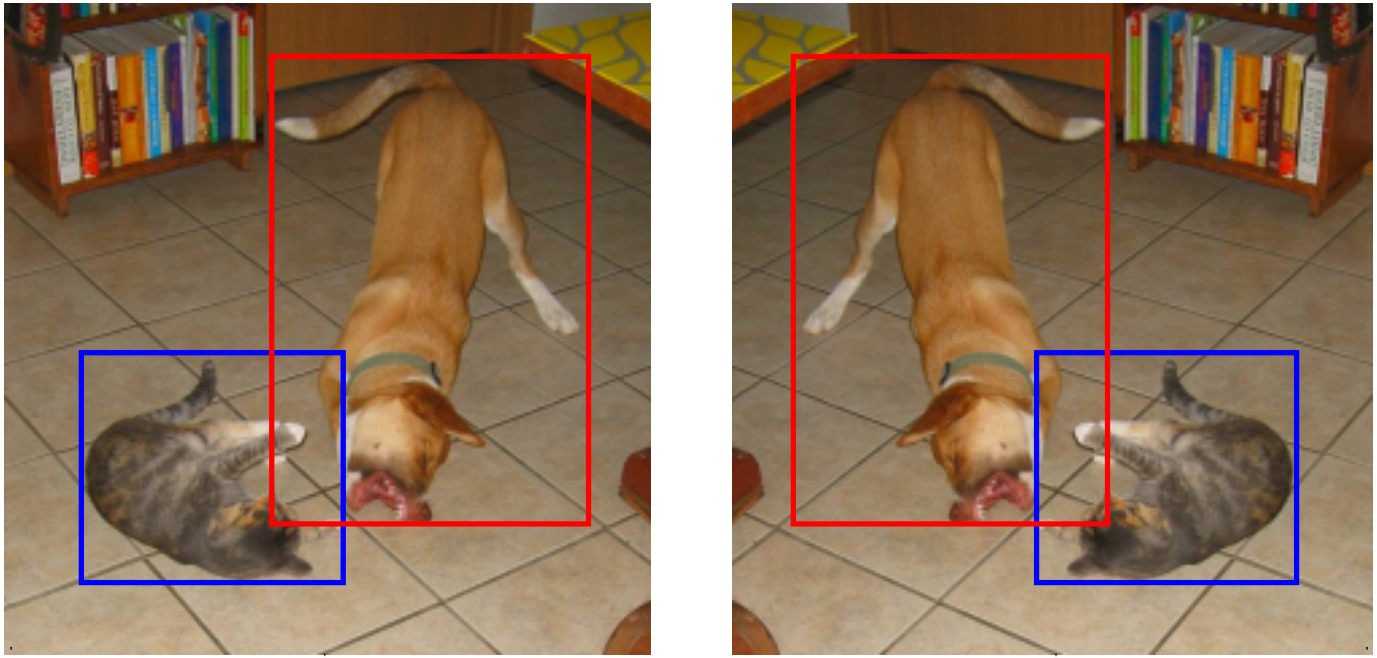


Rotation Invariance



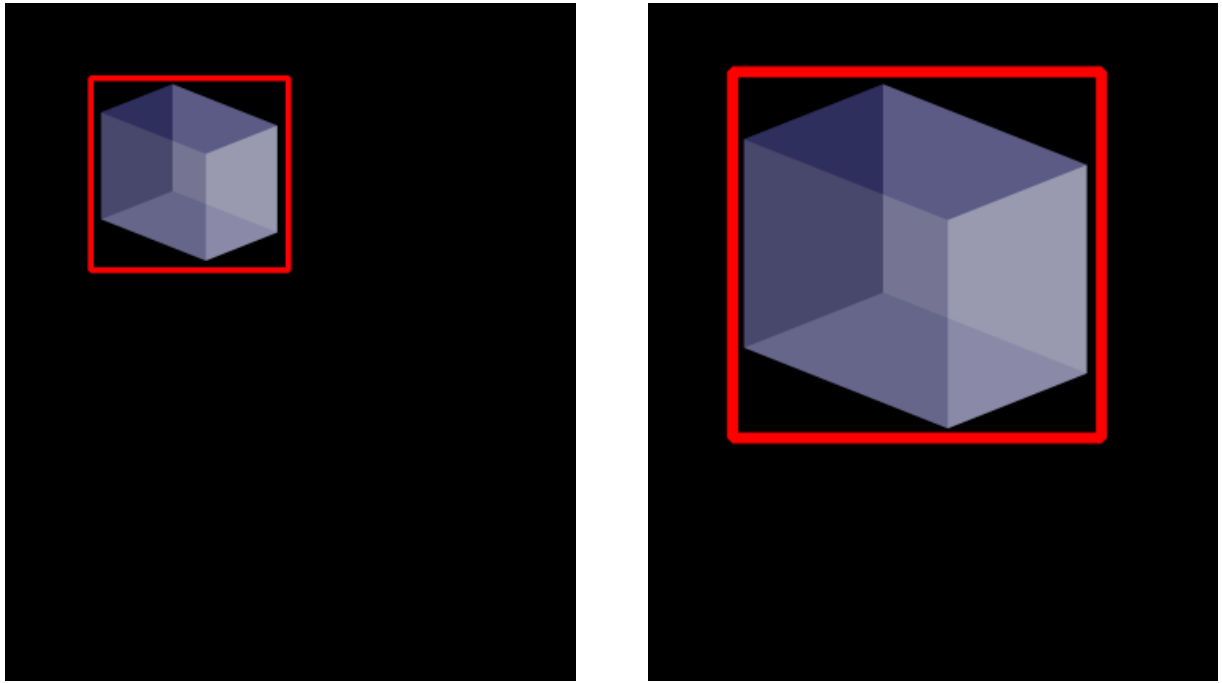
Will be handled by data augmentation and other techniques in the dataset.

Data Augmentation



Feed in mirros, rotations, shifts, etc.

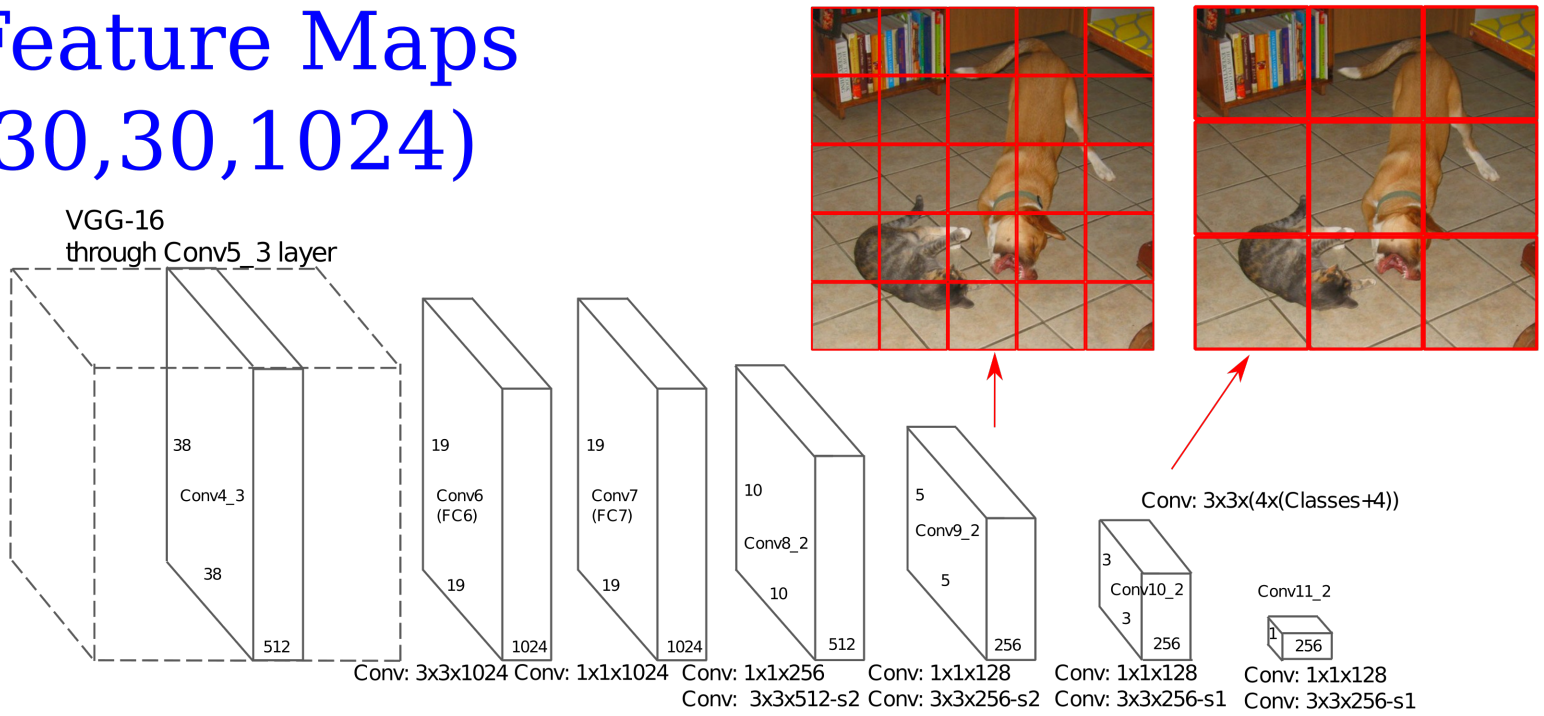
Scale Invariance



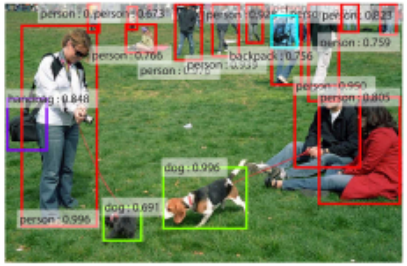
Will by handled by data diversity and rescaling feature maps.

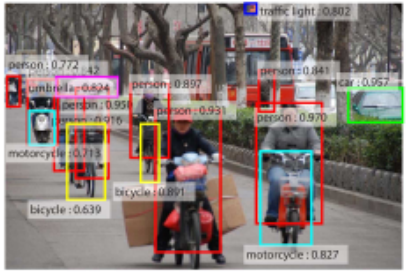
Pyramid or Multiscale

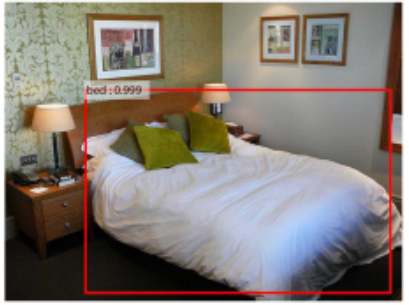
Feature Maps (30,30,1024)

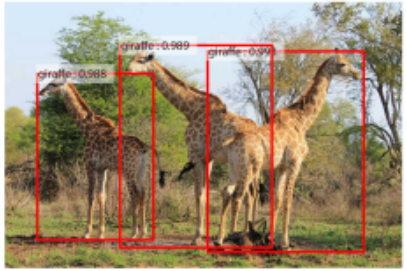


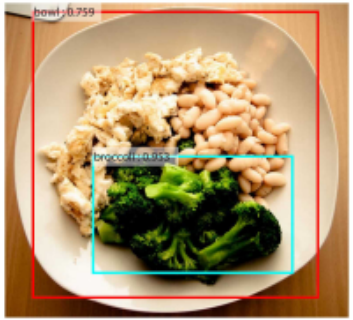
Sample Detections











How do we measure performance

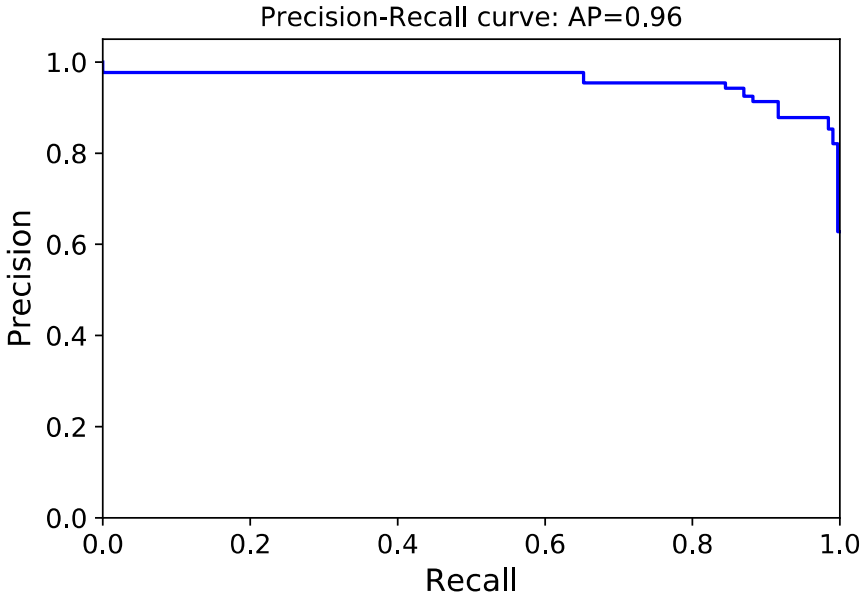
- We use mean average precision (mAP) over all of our classes
- Must use a precision-recall metric *because there are so many ways to get things wrong*
  - Classes are super imbalanced

Types of Predictions

	Predict Negative	Predict Positive
True Label is Positive	False Negatives (FN)	True Positives (TP)
True Label is Negative	True Negatives (TN)	False Positives (FP)

Precision is given by (TP)/(TP+FP).
Recall is given by (TP)/(TP+FN).

Precision-Recall



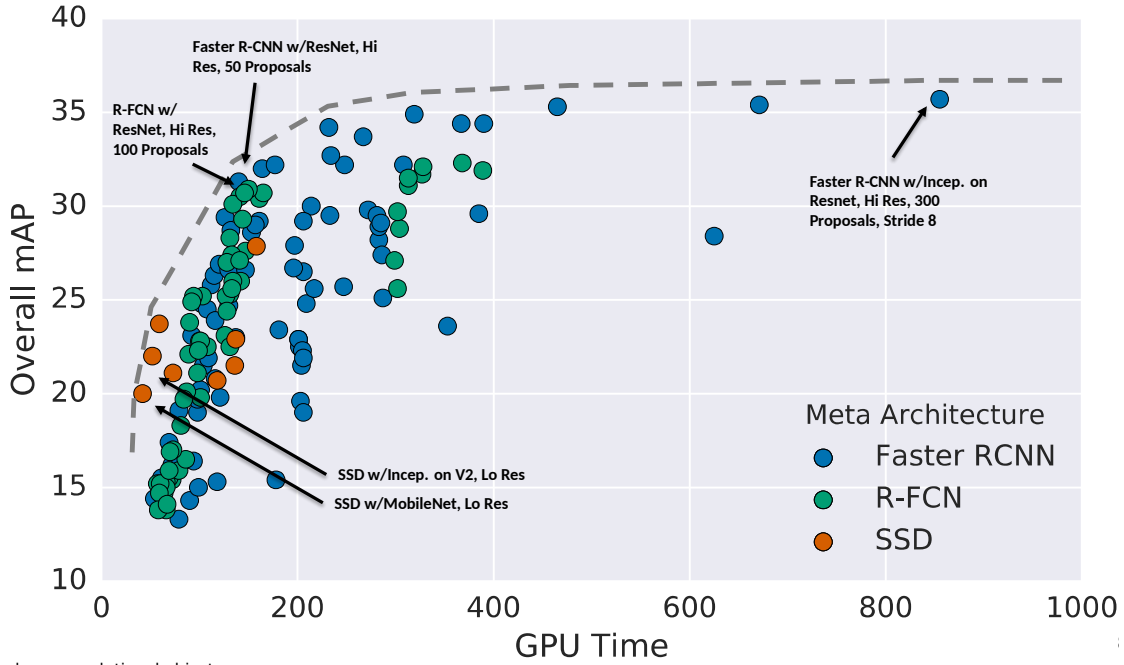
Average Precision is the integral of this curve.

Mean Average Precision

- When we have many classes or individualized responses, it can be difficult to evaluate. Accuracy is dependent on decision thresholds, which means that it can be gamed a bit and may be hard to compare across algorithms.
- Average Precision (AP) isn't perfect, but works well for a single class.
- When we have multiple classes, we can take the "Mean Average Precision," which is the average precision over multiple classes:
  mAP (or MAP)= mean (AP for each class).

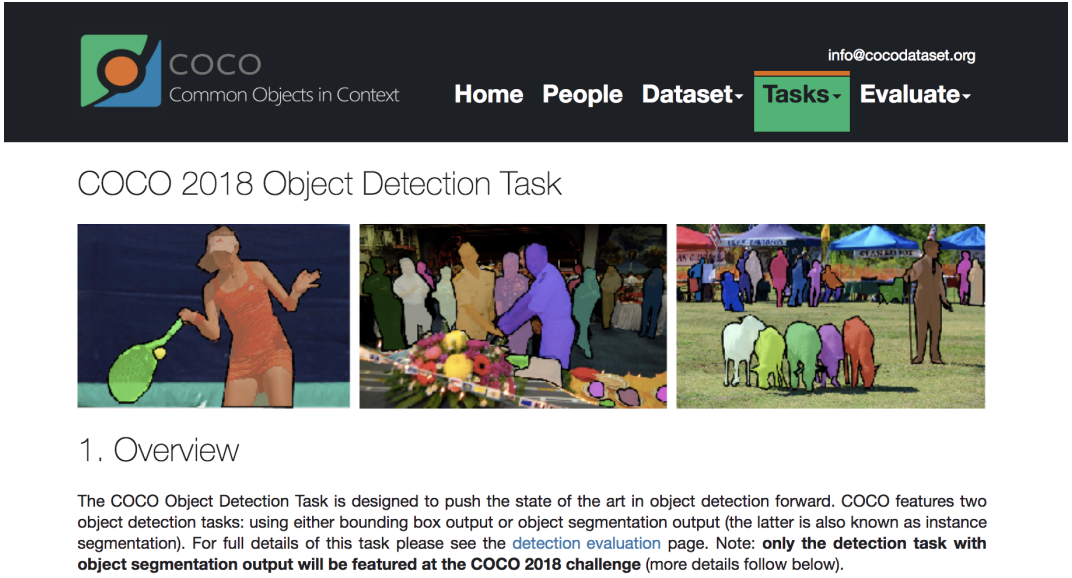
- Other common metrics include:
  F1=2 (Precision*Recall)/(Precision+Recall)
- MRR=mean reciprocal rank (useful when only one answer is appropriate out of many)

Performance Tradeoffs



Huang et al., “Speed/accuracy trade-offs for modern convolutional object detectors”

Where is object detection going?

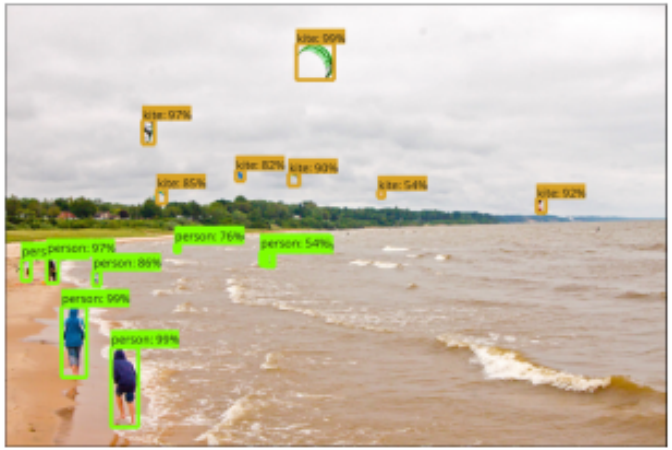


http://cocodataset.org/#detection-2018

Resources for training your own object detector

Tensorflow Object Detection API

Creating accurate machine learning models capable of localizing and identifying multiple objects in a single image remains a core challenge in computer vision. The TensorFlow Object Detection API is an open source framework built on top of TensorFlow that makes it easy to construct, train and deploy object detection models. At Google we've certainly found this codebase to be useful for our computer vision needs, and we hope that you will as well.



Contributions to the codebase are welcome and we would love to hear back from you if you find this API useful. Finally if you use the Tensorflow Object Detection API for a research publication, please consider citing:

Conclusions

- Object detection leads to a similar task as object classification, but with different goals
- Lots of technical details, but feasible in a lot of different challenges
- *Many* real-world applications