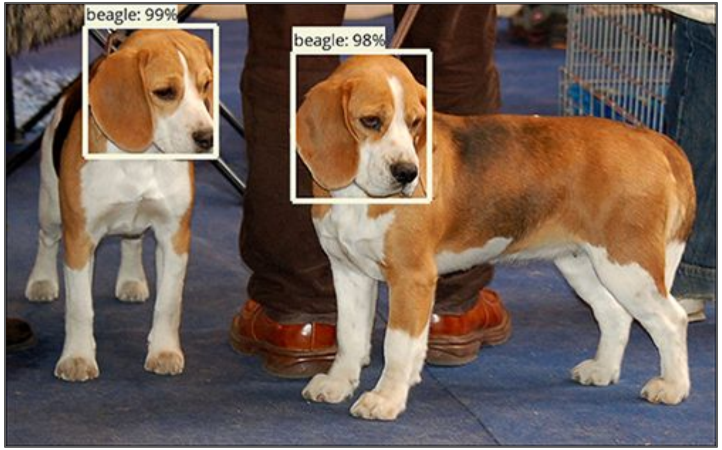
CSEP 576: Object Detection with Convolutional Networks



Jonathan Huang (jonathanhuang@google.com)

University of Washington 17 May 2018

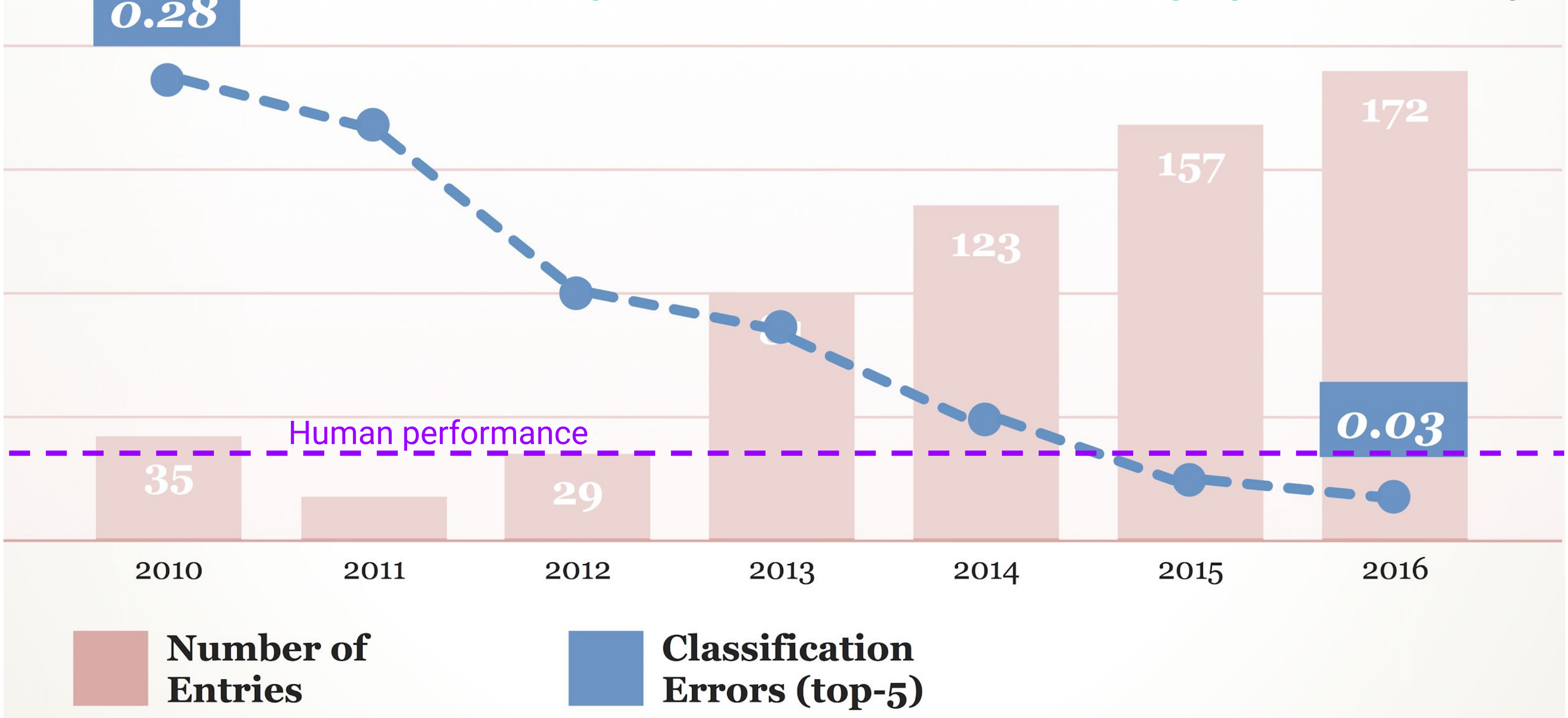
Google AI

~~Today's~~ Yesterday's Image Taggers just returned a bag of words...

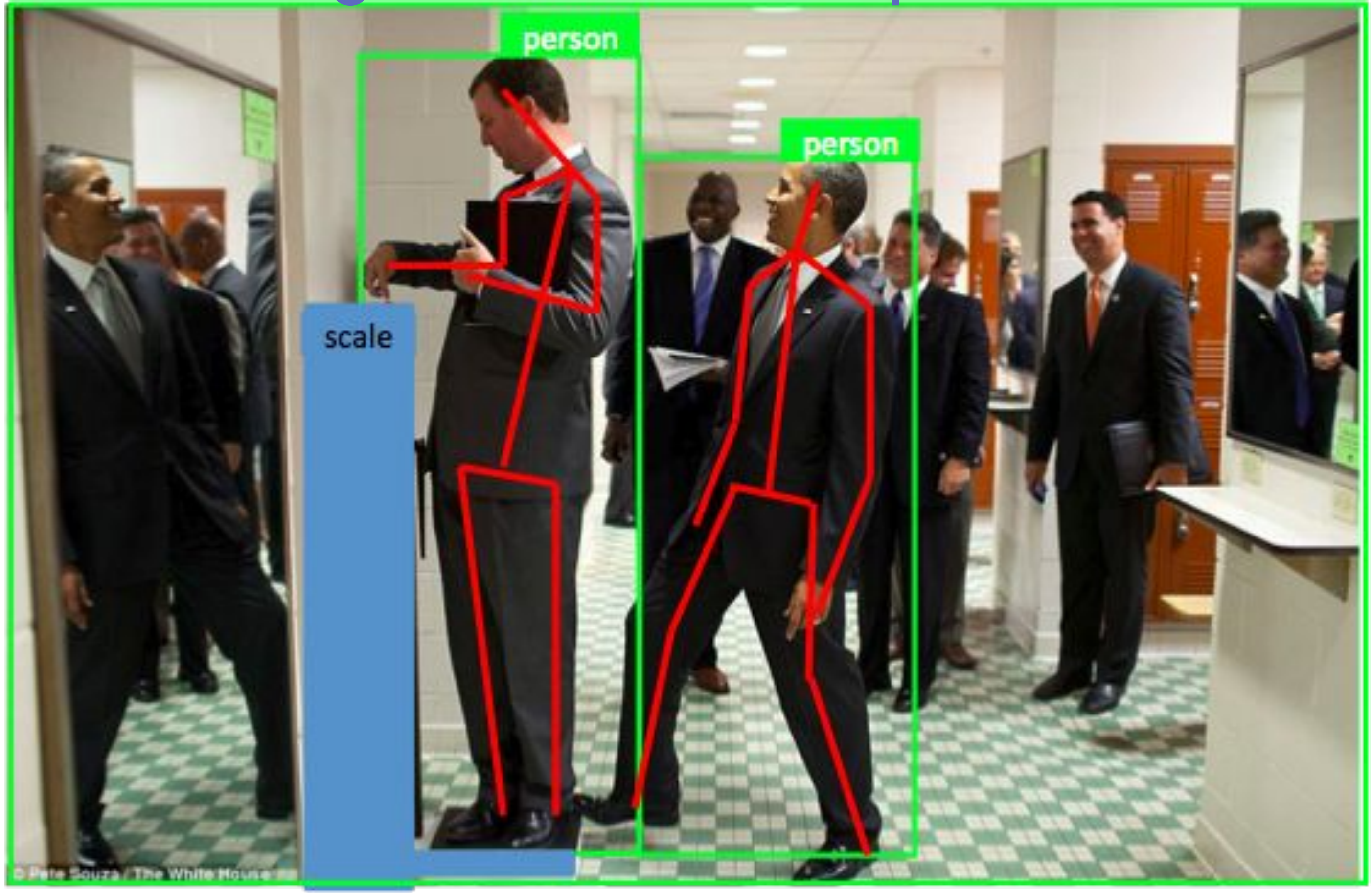
Source: http://karpathy.github.io/2012/10/22/state-of-computer-vision/

Imagenet Progress Over the Years

Source: [Imagenet: Where have we been? Where are we going?](#) *Fei Fei Li, Jia Deng*

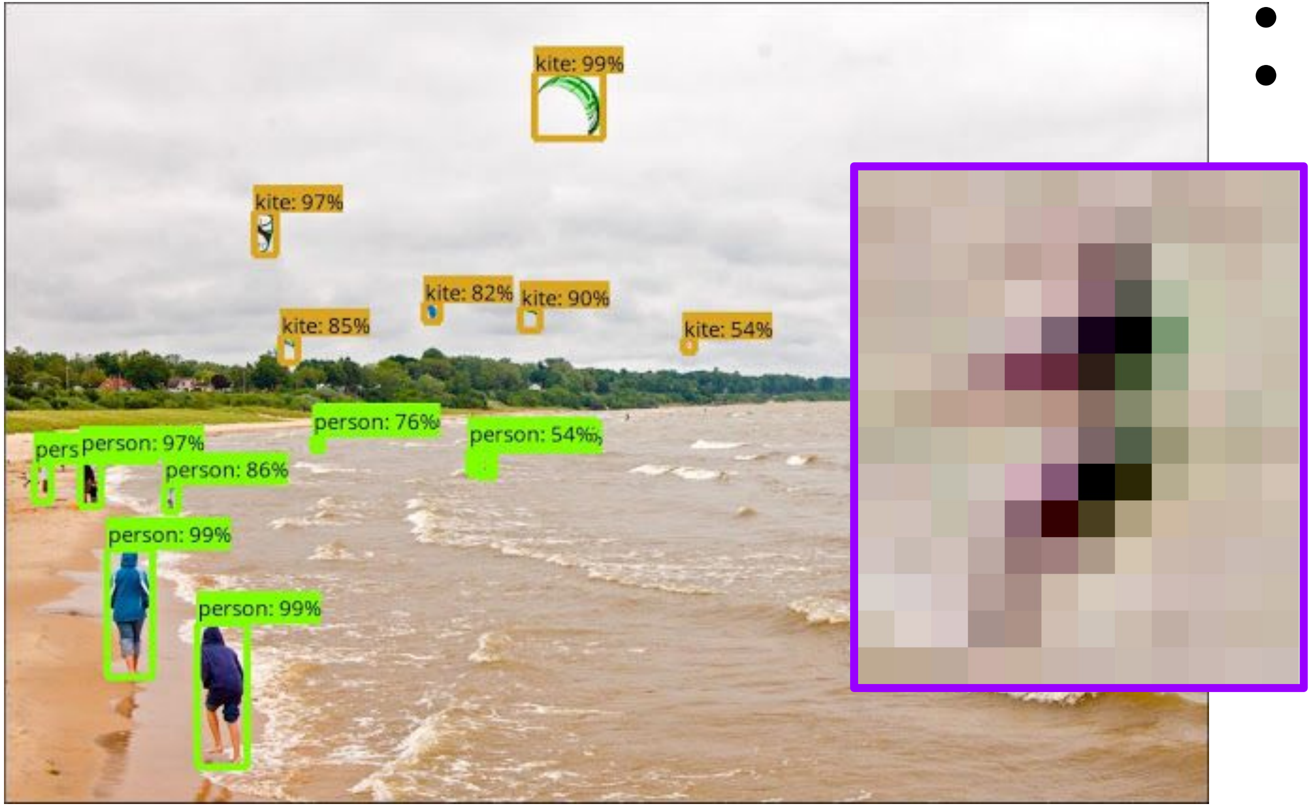


Now: boxes, segments, human pose...



Based on a figure from Jia Deng

From Classification to Detection

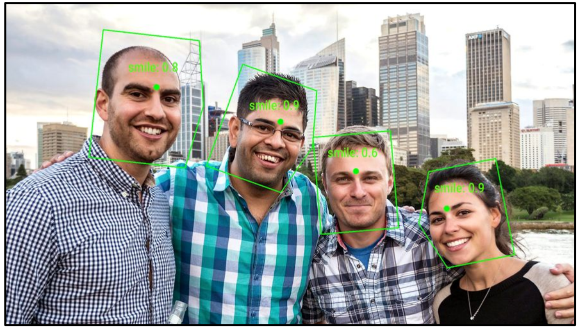


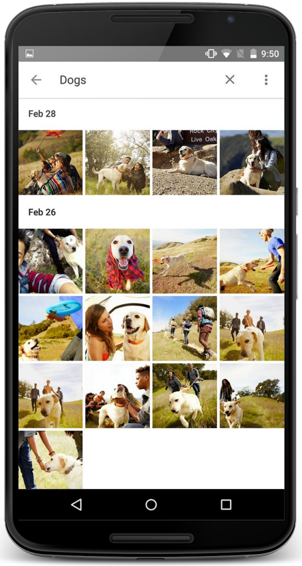
Detection = Classification + Localization

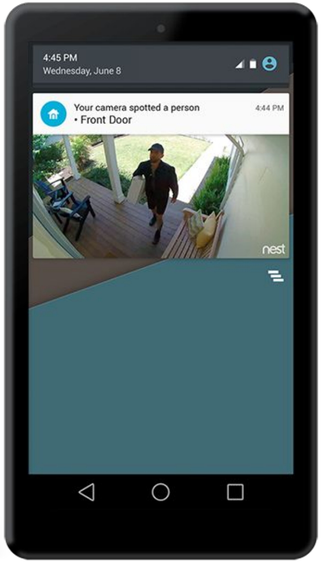
- Variable # outputs
- *Need to classify based on much fewer pixels than in Imagenet setting*; Requires context!

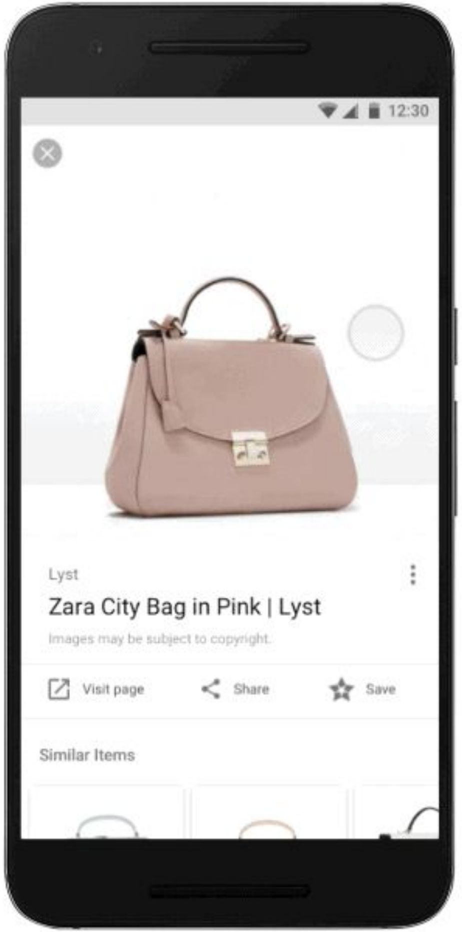
Photo credit: Michael Mina

Object Detection Applications

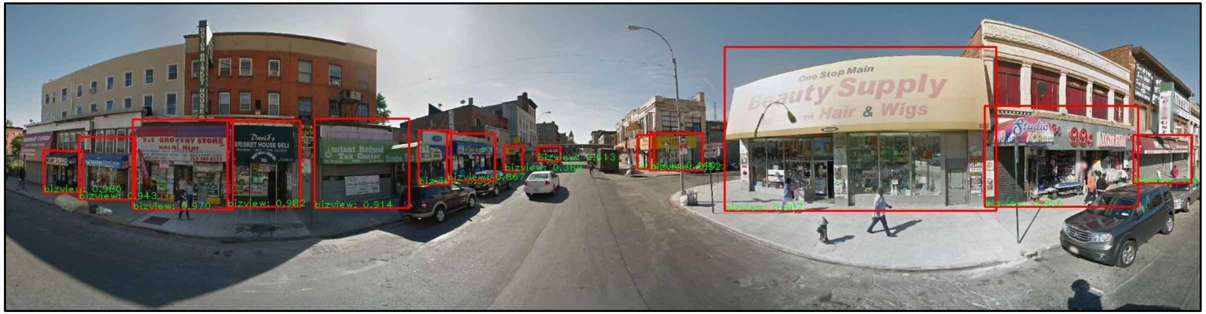








Object Detection Applications



Object Detection Applications

Bus Lane Blocked, He Trained His Computer to Catch Scofflaws

Alex Bell developed a computer program that used a traffic camera to identify how often bus and bicycle lanes were blocked by unauthorized vehicles along one block in Harlem.

Christopher Lee for The New York Times

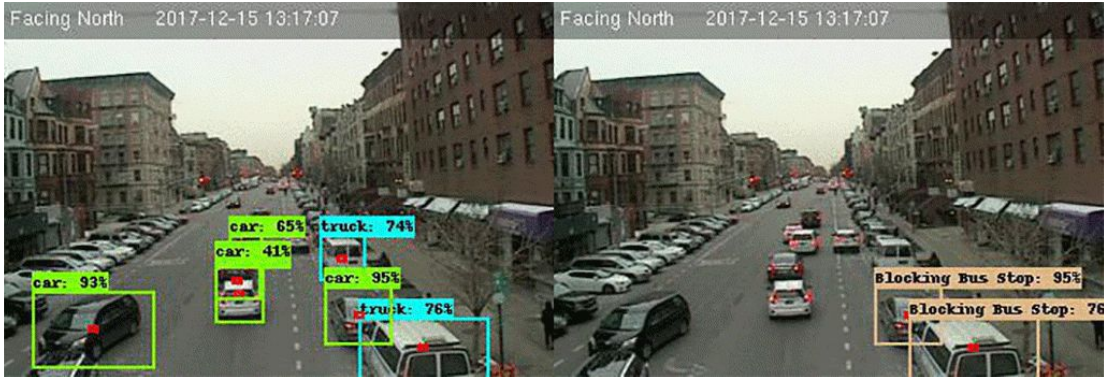


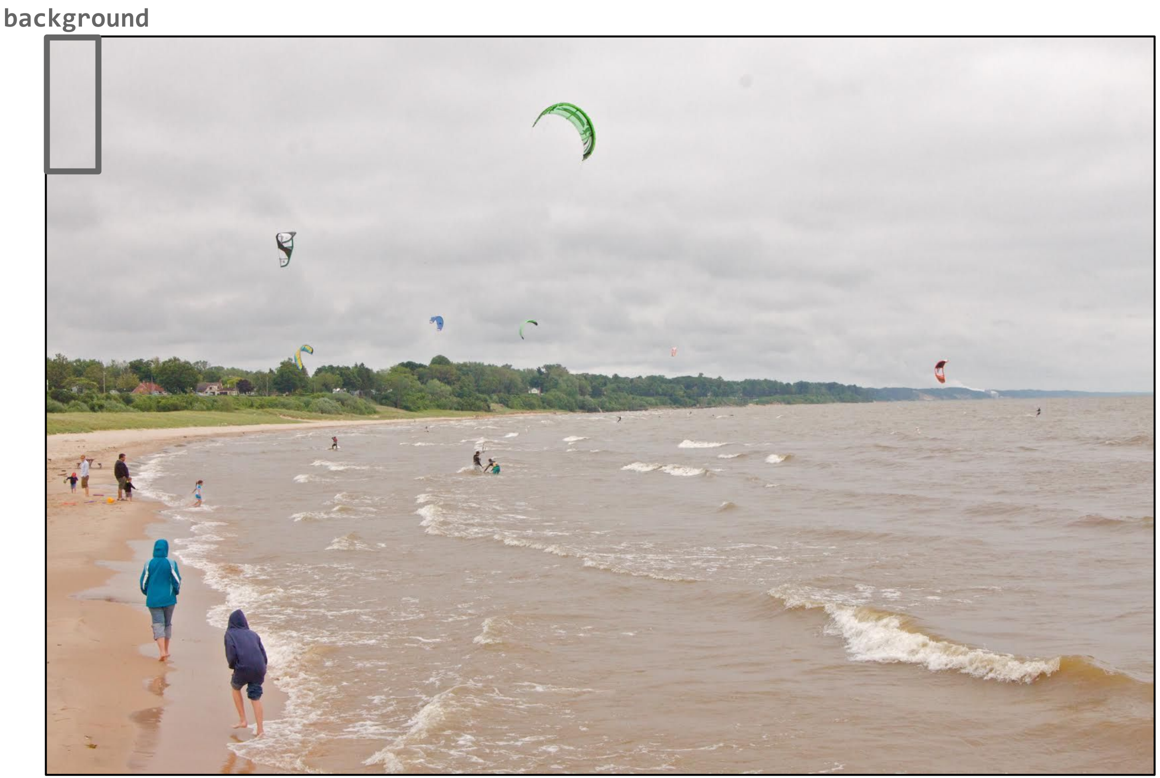
Image credit: NYTimes (author: Sarah Maslin Nir)

Object Detection Applications

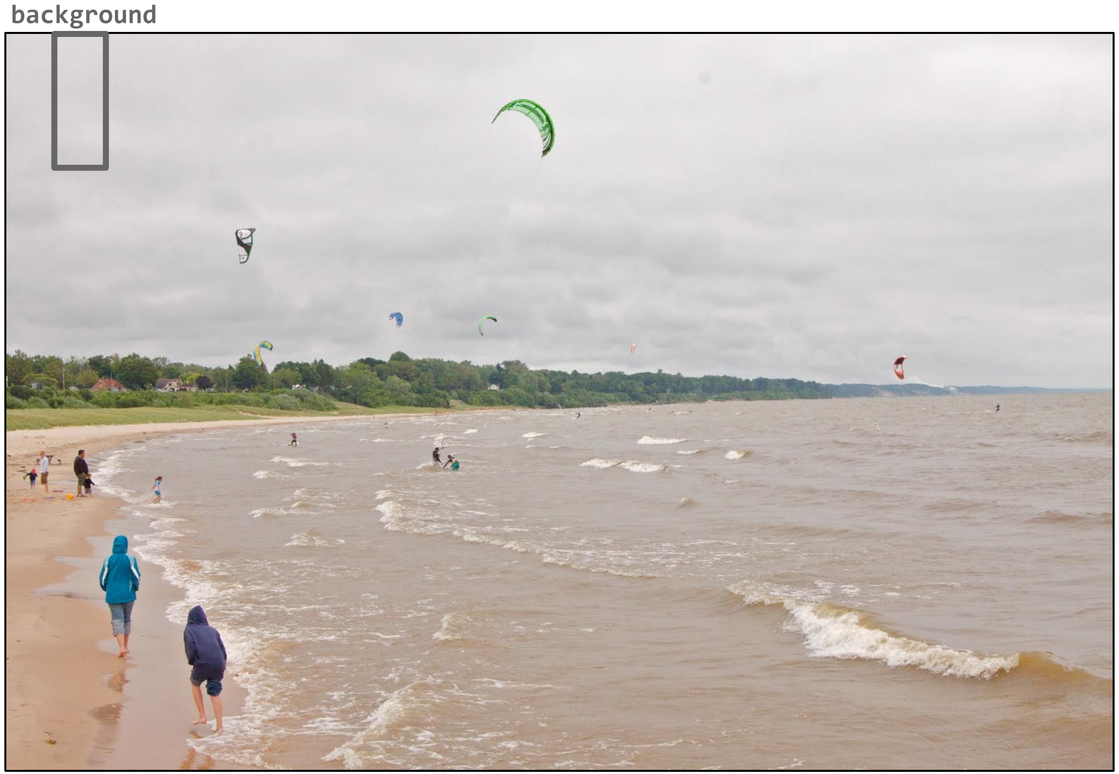
Today

- **Sliding Window Detectors**
- Detection with Convolutional Networks
- How to Evaluate a Detector
- Practical tips/tricks
- Variations on a theme (instance segmentation, keypoint detection, video detection, etc...)

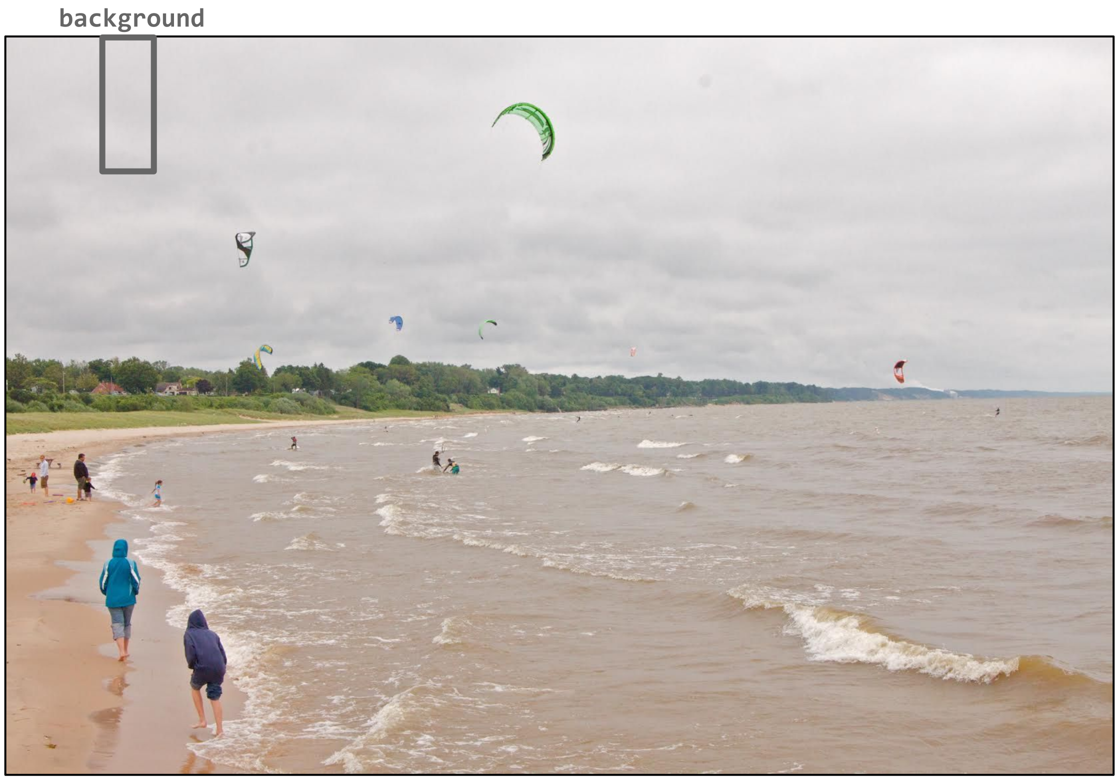
“Sliding Window” Detection



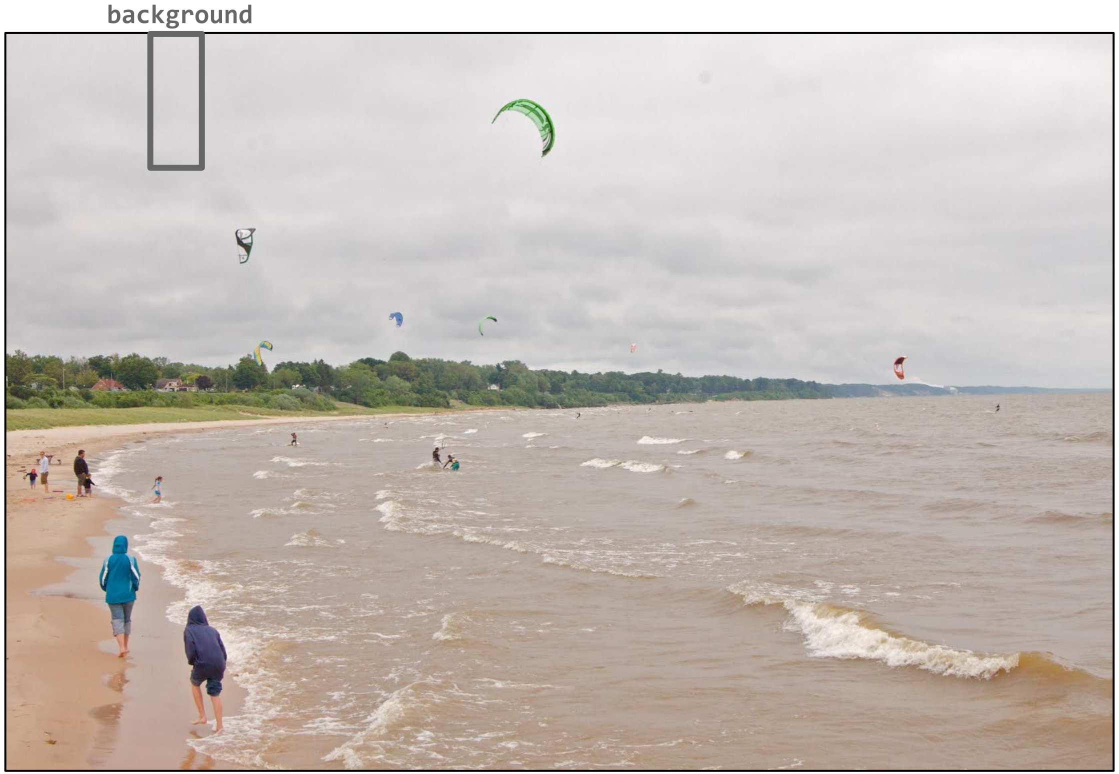
“Sliding Window” Detection



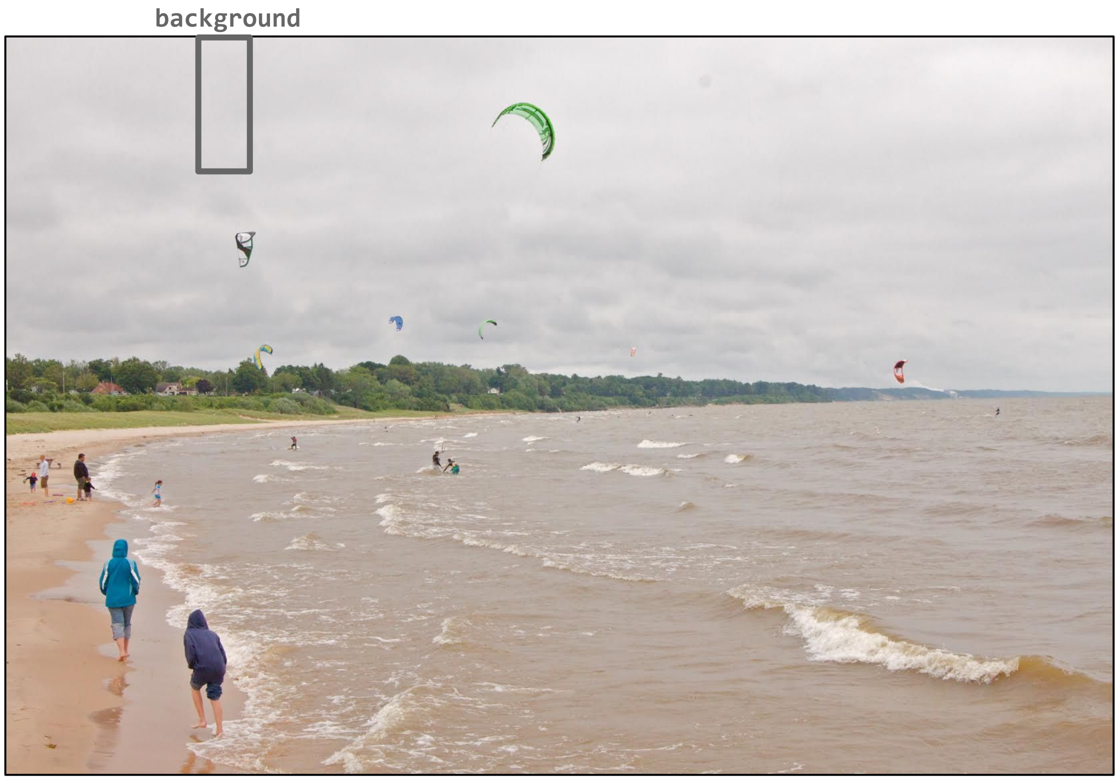
“Sliding Window” Detection



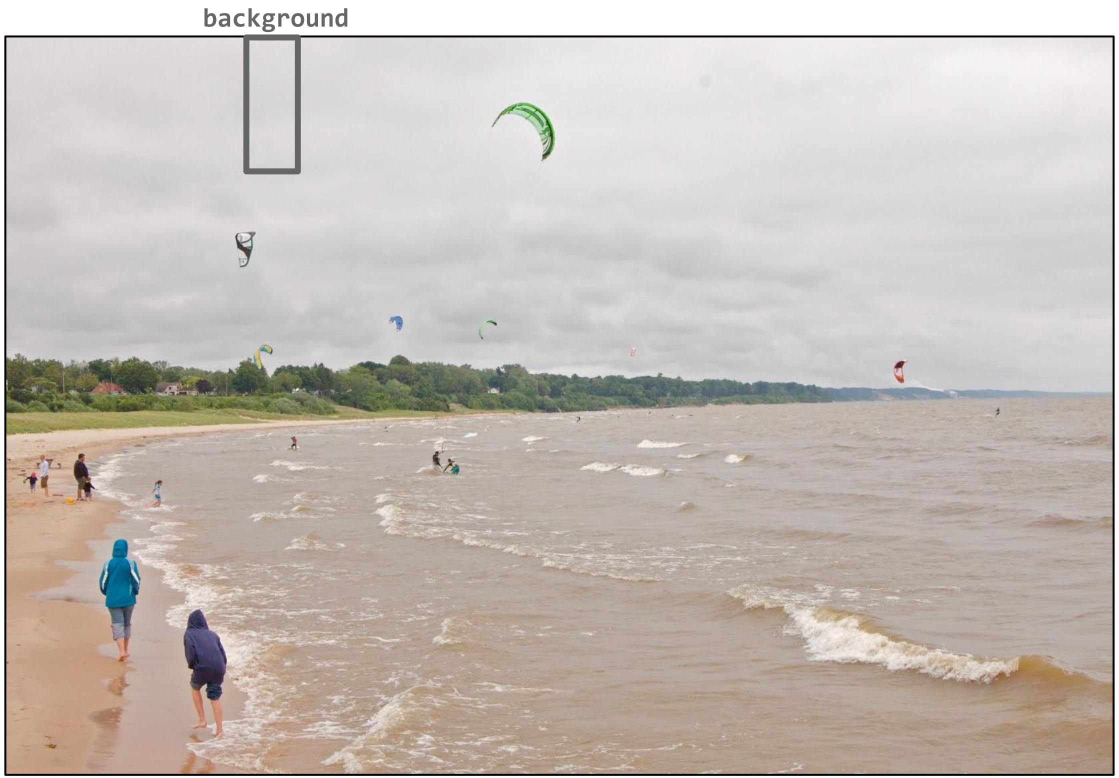
“Sliding Window” Detection



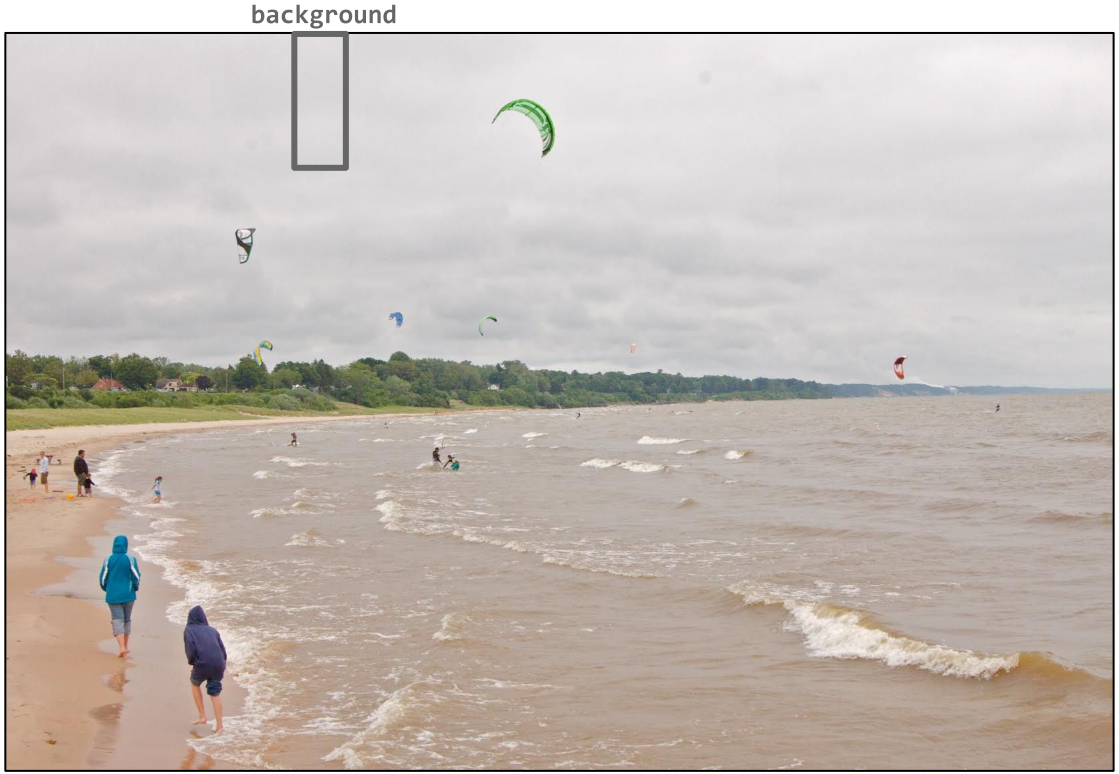
“Sliding Window” Detection



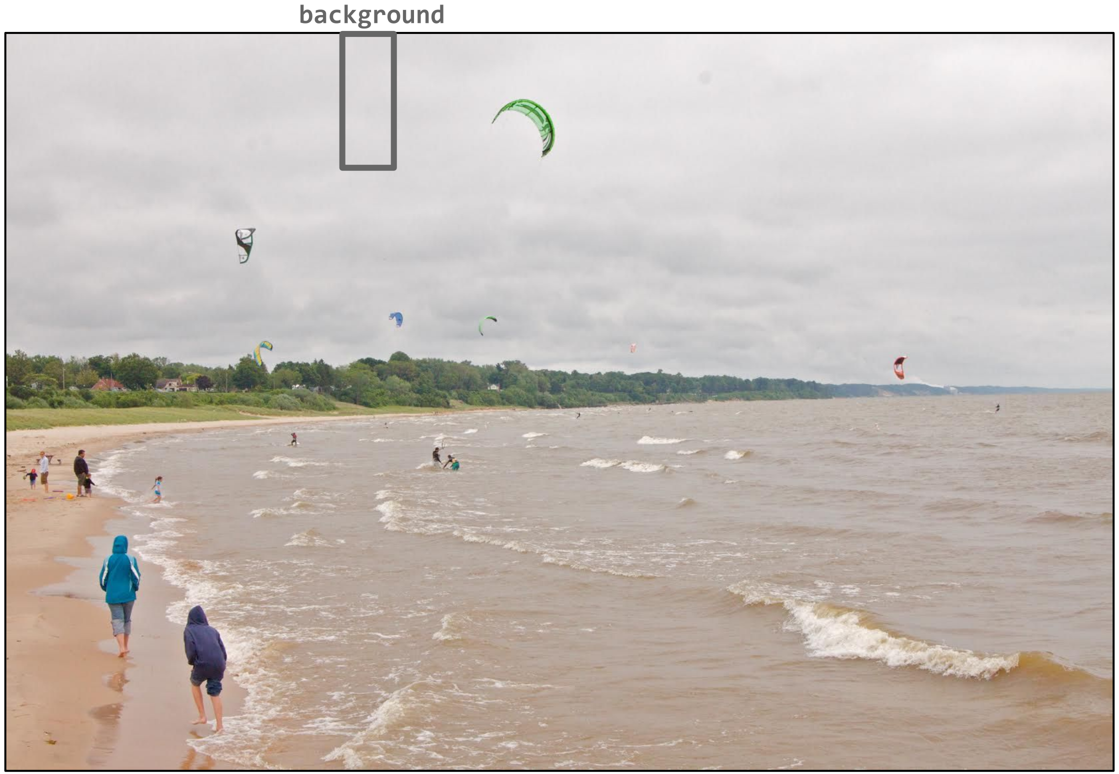
“Sliding Window” Detection



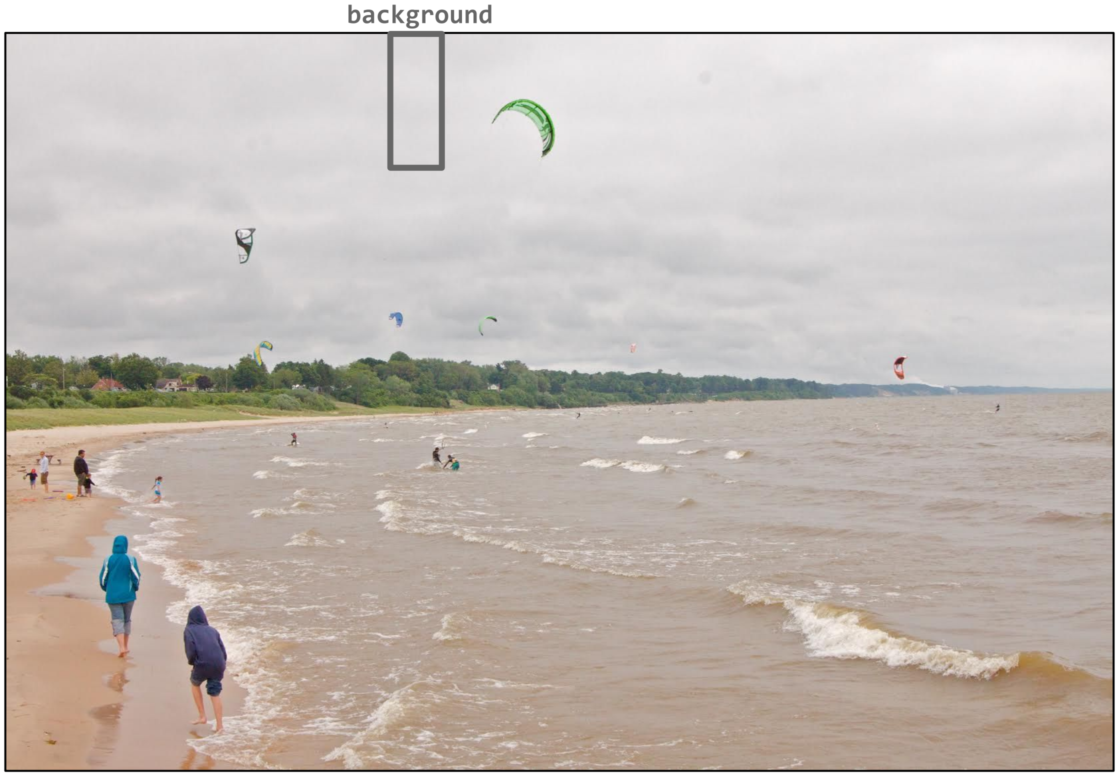
“Sliding Window” Detection



“Sliding Window” Detection

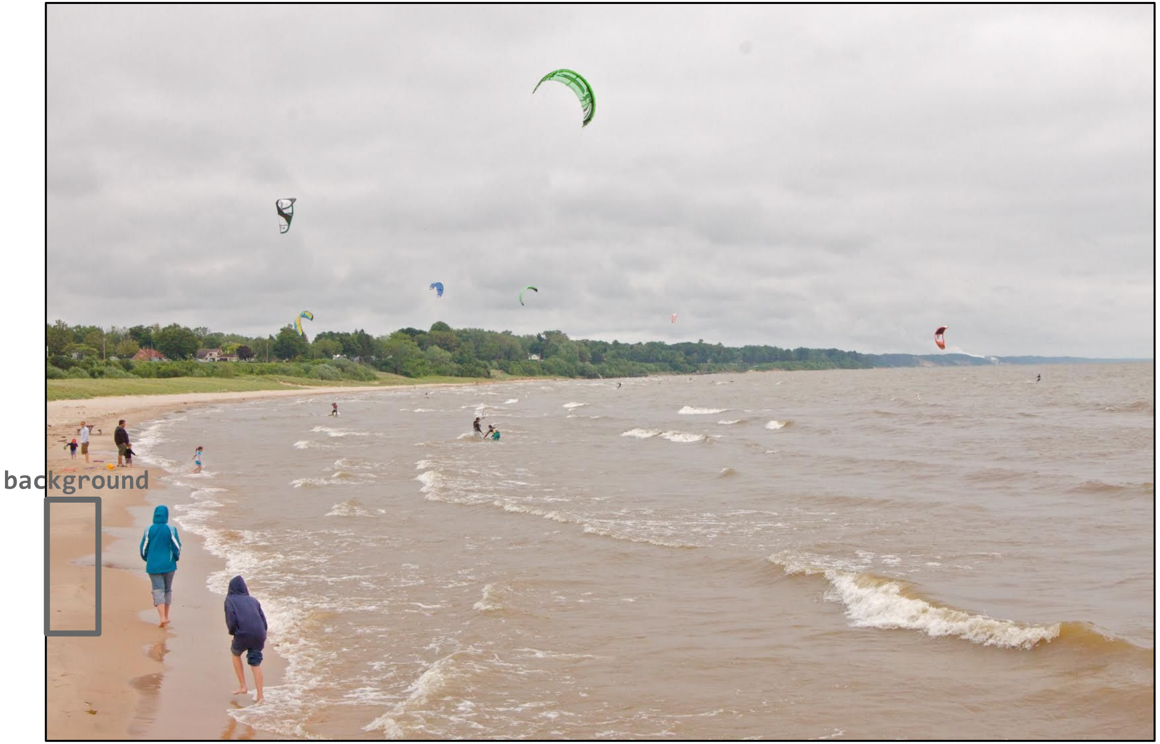


"Sliding Window" Detection



“Sliding Window” Detection

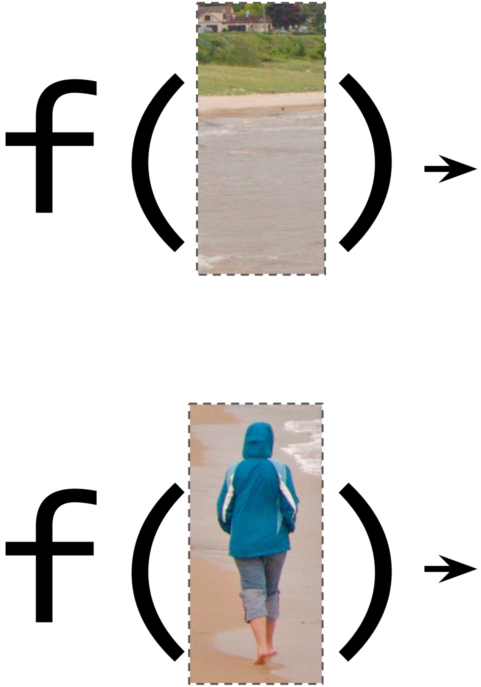
“Sliding Window” Detection



“Sliding Window” Detection

“Sliding Window” Detection

“Sliding Window” Detection



Compute within-region features, then classify

“Sliding Window” Detection

f()

f()

Typical to enlarge region to include some “context”

Sliding window placement

Slide over ***fine grid***
in x, y, scale, aspect ratio

Slow and Accurate

Slide over ***coarse grid***
in x, y, scale, aspect ratio

Fast and Not-so-accurate

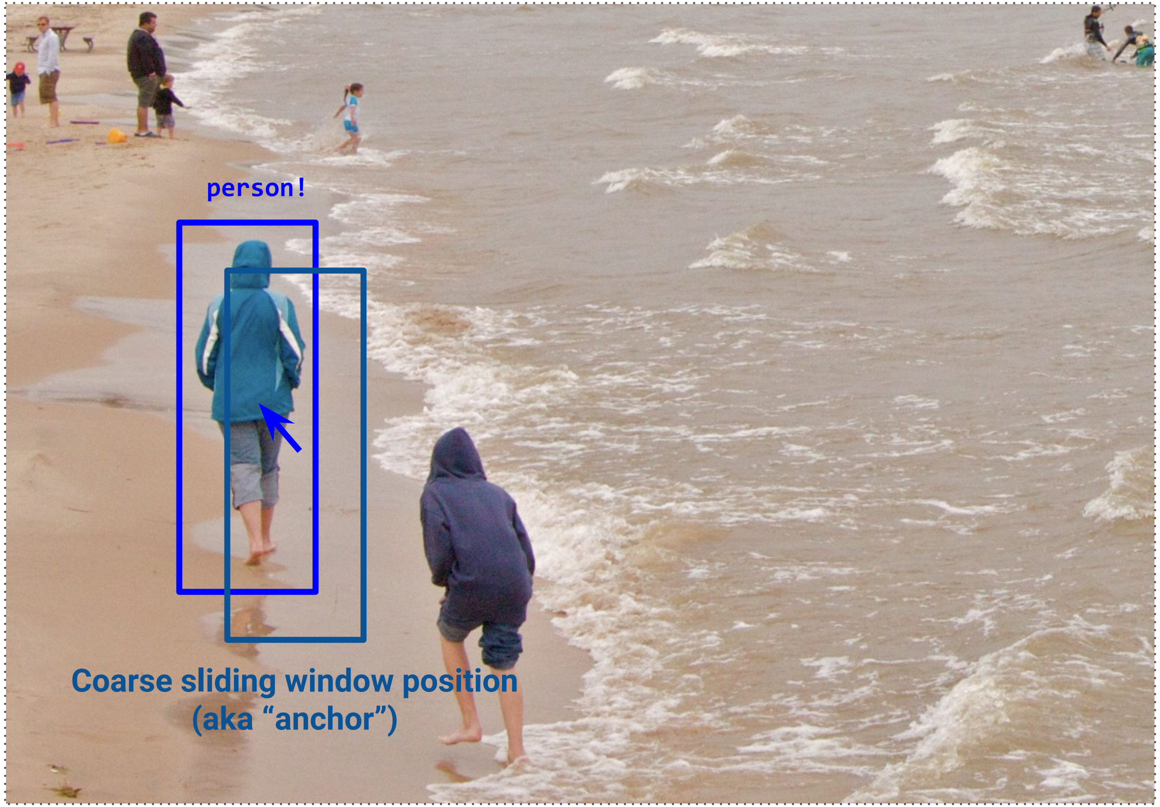
(... or can it be?)

Bounding Box Regression

Idea:

Also predict continuous offset from anchor to “snap” onto object

Bounding Box Regression



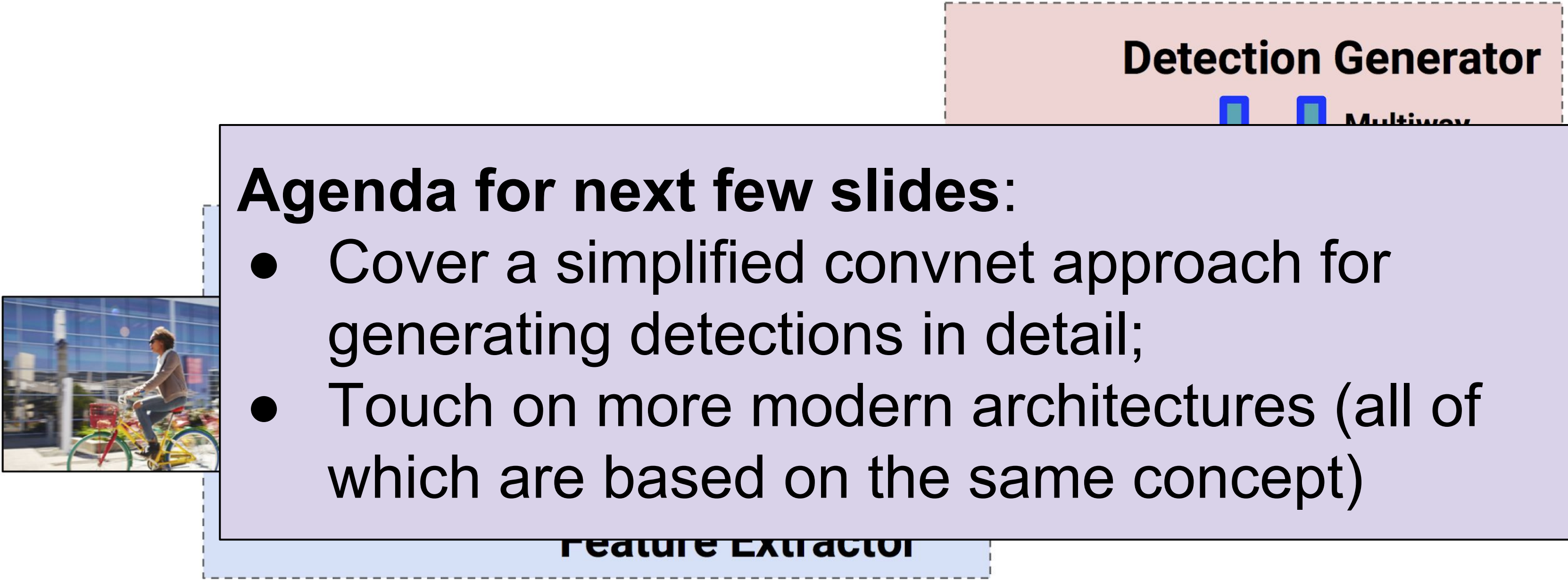
Idea:

Also predict continuous offset from anchor to “snap” onto object

Today

- Sliding Window Detectors
- **Detection with Convolutional Networks**
- How to Evaluate a Detector
- Practical tips/tricks
- Variations on a theme (instance segmentation, keypoint detection, video detection, etc...)

Using convolutional networks for detection



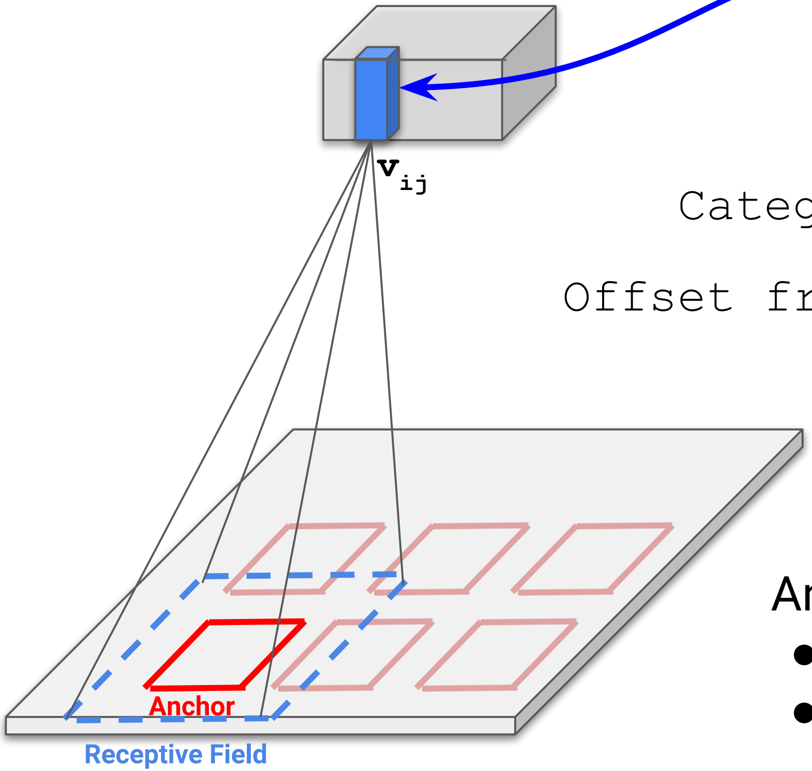
Agenda for next few slides:

- Cover a simplified convnet approach for generating detections in detail;
- Touch on more modern architectures (all of which are based on the same concept)

- Extract features at sliding window positions via convolution
- Deep networks -> large receptive fields that can account for context

A simplified convnet for detection

Think of each feature vector $\mathbf{v}_{ij}$ as corresponding to a sliding window (anchor).



$$\texttt{Category score} = \texttt{SoftMax}(W^{cls} \cdot \mathbf{v}_{ij})$$

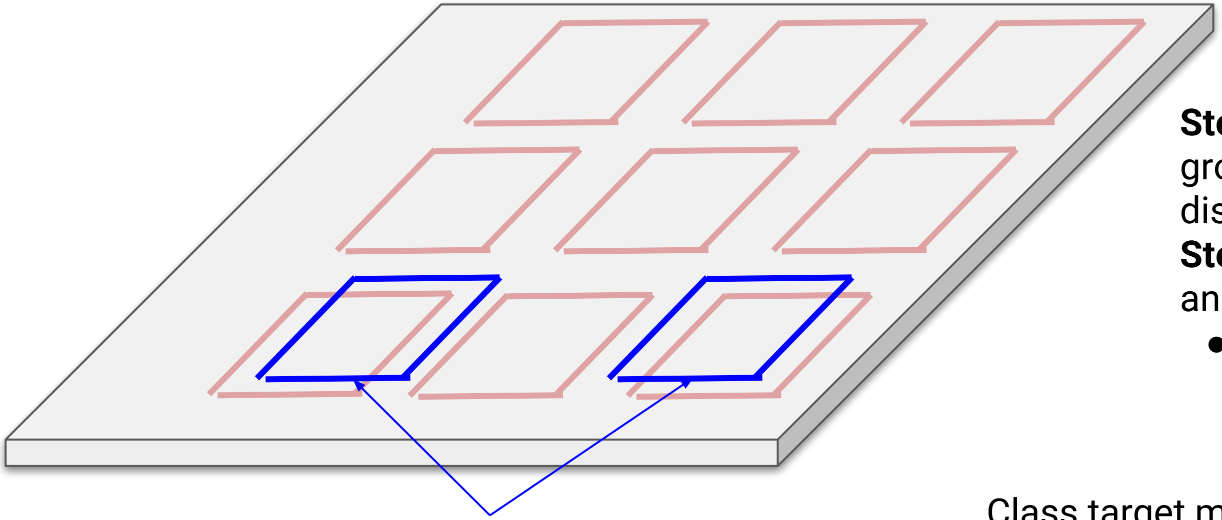
$$\texttt{Offset from anchor} = W^{loc} \cdot \mathbf{v}_{ij}$$

Use the same W^{loc} and W^{cls} for all i, j in anchor grid.

Anchors assumed to be:

- of the same shape, and
- contained and centered in receptive field

Target Assignment



groundtruth boxes (person, class 2)

Step 1: Match anchor boxes to groundtruth boxes (based on Euclidean distance or overlap area)
Step 2: Give each anchor a classification and regression target

- If anchor has no matching groundtruth, it classifies as 0 and no regression target is given

Class target matrix
(one entry per anchor)

0	0	0
0	0	0
2	0	2

Location targets
(only for matched anchors)

$$\begin{pmatrix} gt_{xmin} - anchor_{xmin} \\ gt_{ymin} - anchor_{ymin} \\ gt_{xmax} - anchor_{xmax} \\ gt_{ymax} - anchor_{ymax} \end{pmatrix}$$

Typical Training Objective

Common to use other
location losses here...

Per-anchor Loss:

$$\mathtt{L(anchor}\ \mathbf{a}\mathtt{)} = \alpha * \delta(\mathbf{a}\ \mathtt{has\ matching\ groundtruth}) * \mathtt{L}_2(\mathbf{t}^{\mathtt{loc}}, \mathtt{W}^{\mathtt{loc}} \cdot \mathbf{v}_{\mathbf{ij}})$$

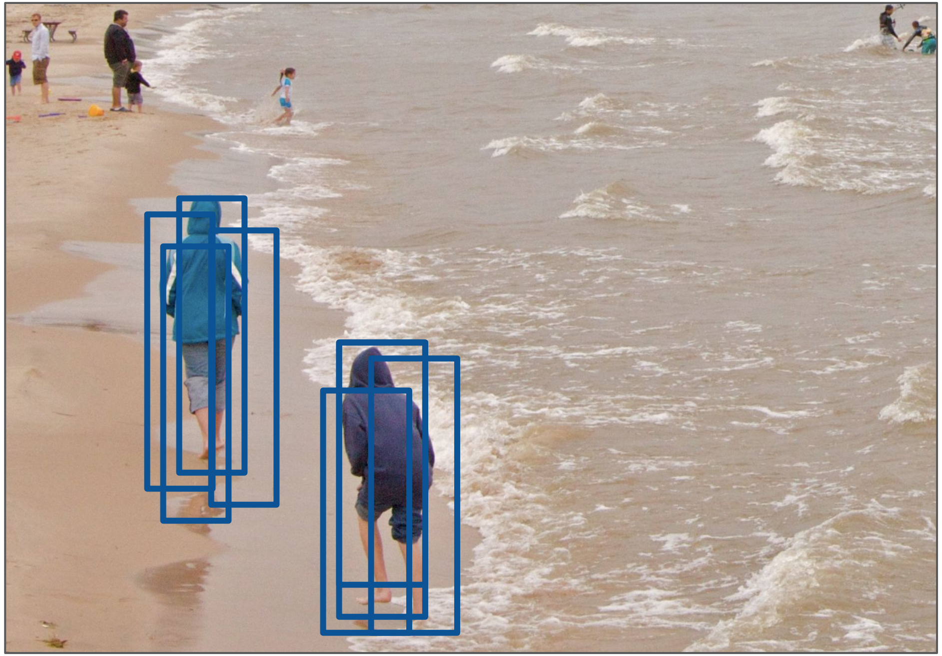
$$+ \quad \beta * \mathtt{SoftMaxCrossEntropy}(\mathbf{t}^{\mathtt{cls}}, \mathtt{W}^{\mathtt{cls}} \cdot \mathbf{v}_{\mathbf{ij}})$$

Total Loss: Average per-anchor loss over anchors

Challenge: Dealing with class imbalance (usually way more negative anchors (class 0) than positive anchors

Solutions: Subsampling negative anchors, downweighting the loss contribution of negatives, hard mining, etc...

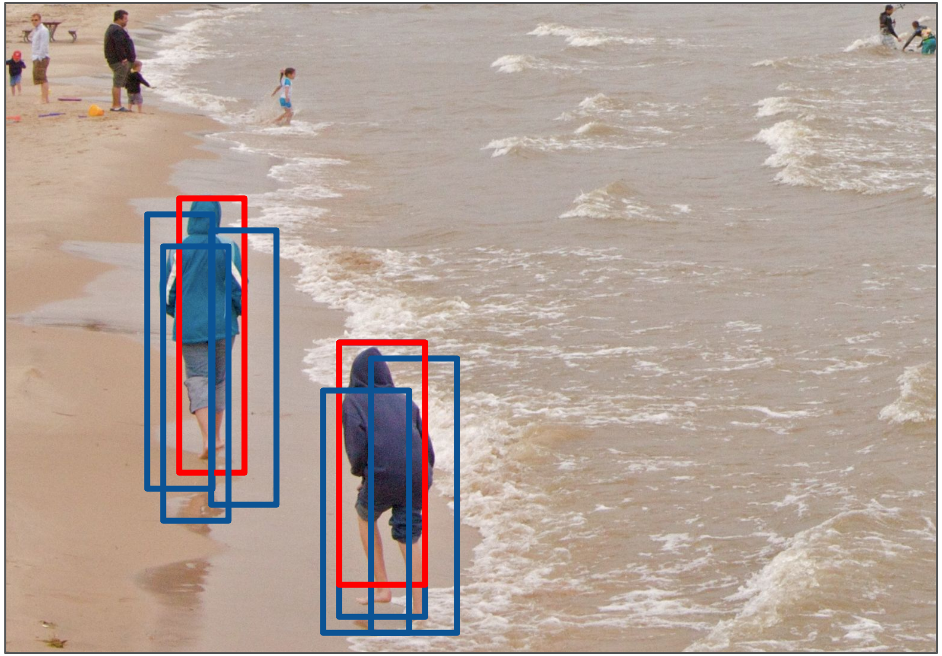
Dealing with multiple detections of the same object



Duplicate detection problem: Typically many anchors will detect the same underlying object and give slightly different boxes, with slightly different scores.

Solution: remove detections if they overlap too much with another higher scoring detection.

Non Max Suppression (NMS)



Algorithm:

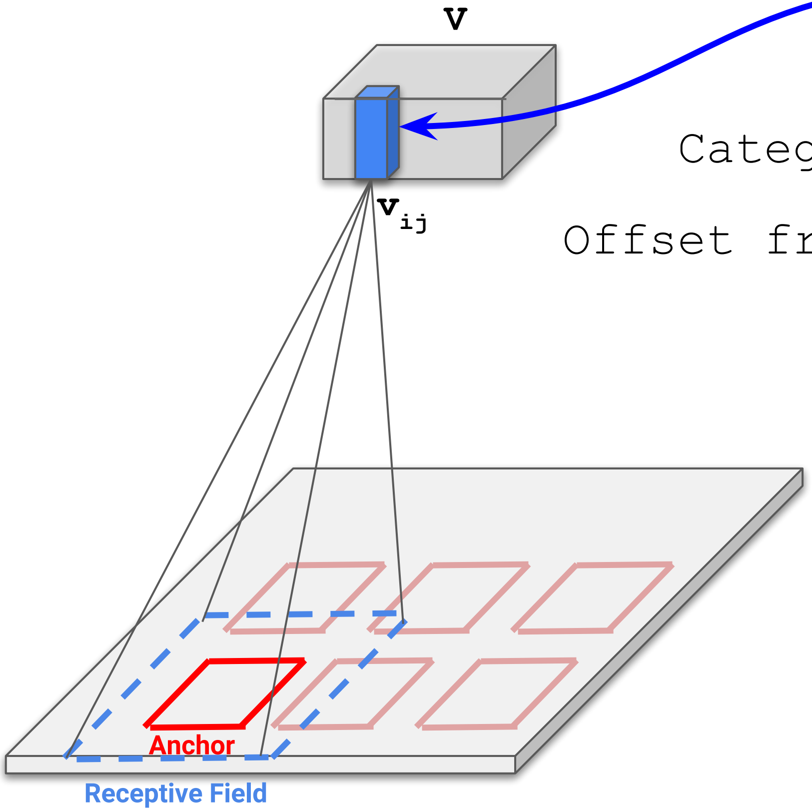
1. Sort detections in decreasing order with respect to score
2. Iterate through sorted detections:
    a. Reject a detection if it overlaps with a previous (unrejected) detection with IOU greater than some threshold
3. Return all unrejected detections

Some shortcomings of NMS to remember:

- Imposes a hard limitation on how close objects can be in order to be detected
- Similar classes do not suppress each other

A simplified convnet for detection

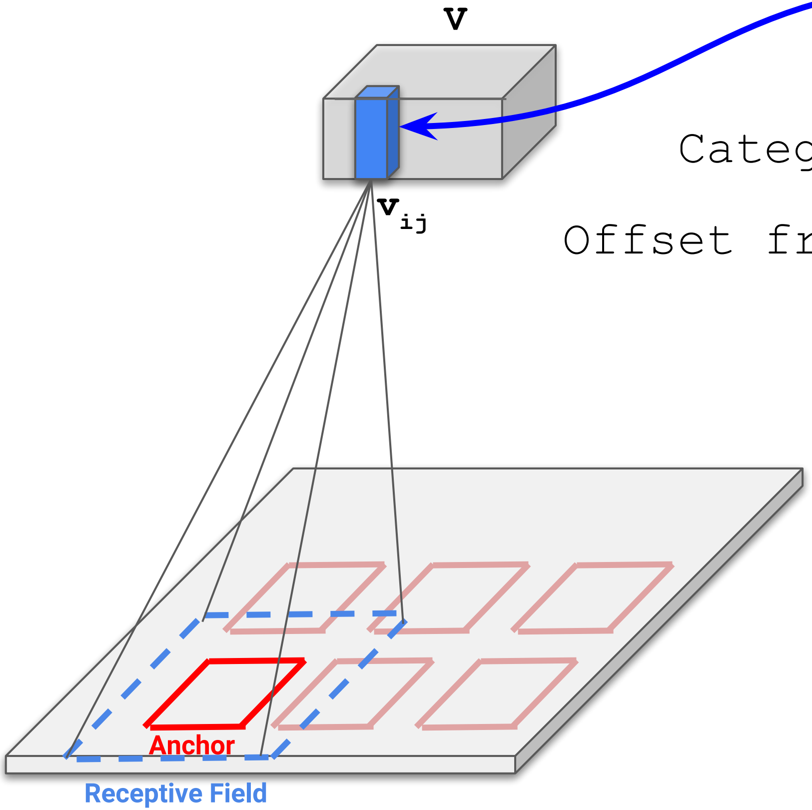
Think of each feature vector $\mathbf{v}_{ij}$ as corresponding to a sliding window (anchor).



$$\text{Category score} = \text{SoftMax}(W^{cls} \cdot \mathbf{v}_{ij})$$

$$\text{Offset from anchor} = W^{loc} \cdot \mathbf{v}_{ij}$$

A simplified convnet for detection



Think of each feature vector $\mathbf{v}_{ij}$ as corresponding to a sliding window (anchor).

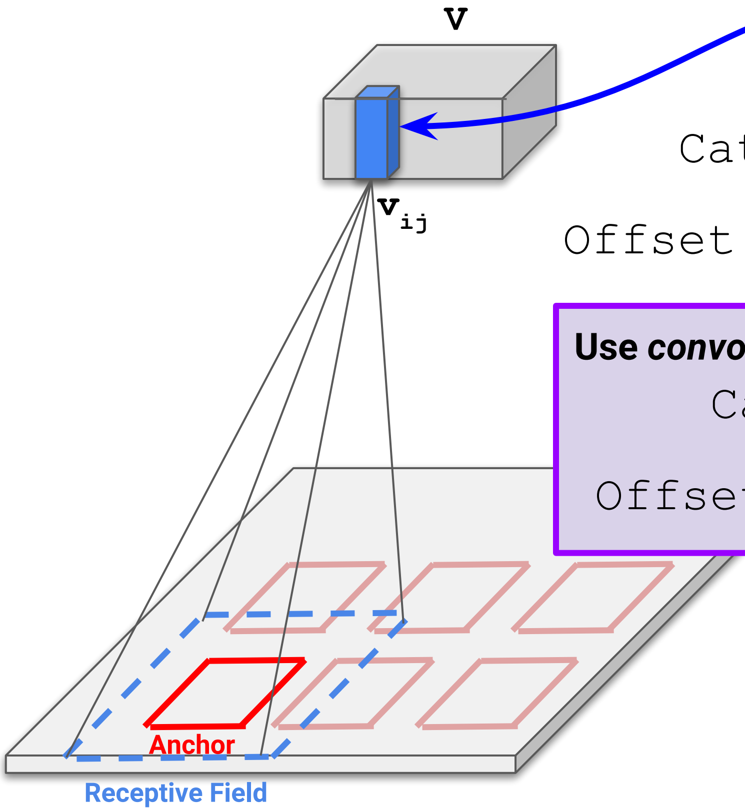
$$\text{Category score} = \text{SoftMax}(W^{cls} \cdot \mathbf{v}_{ij})$$

$$\text{Offset from anchor} = W^{loc} \cdot \mathbf{v}_{ij}$$

Use the same W^{loc} and W^{cls} for all i, j in anchor grid if anchors are:

- **of the same shape**, and
- **contained and centered in receptive field**

A simplified convnet for detection



Think of each feature vector $\mathbf{v}_{ij}$ as corresponding to a sliding window (anchor).

$$\text{Category score} = \text{SoftMax}(W^{cls} \cdot \mathbf{v}_{ij})$$

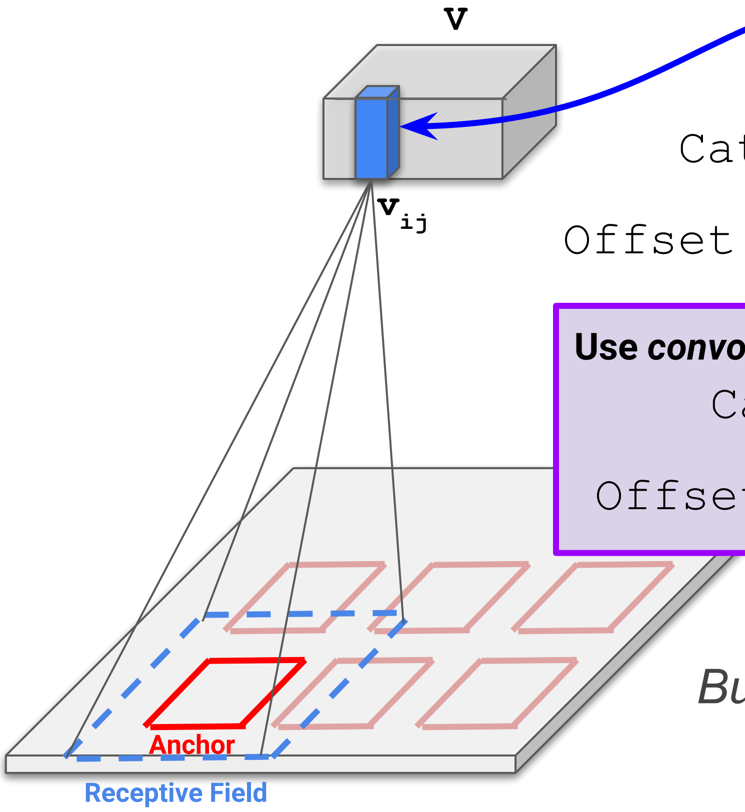
$$\text{Offset from anchor} = W^{loc} \cdot \mathbf{v}_{ij}$$

Use *convolution* to do simultaneous prediction for all anchors:

$$\text{Category score} = \text{SoftMax}(\text{Conv}(\mathbf{v};\ W^{cls}))$$

$$\text{Offset from anchor} = \text{Conv}(\mathbf{v};\ W^{loc})$$

A simplified convnet for detection



Think of each feature vector $\mathbf{v}_{ij}$ as corresponding to a sliding window (anchor).

$$\texttt{Category score} = \texttt{SoftMax}(W^{cls} \cdot \mathbf{v}_{ij})$$

$$\texttt{Offset from anchor} = W^{loc} \cdot \mathbf{v}_{ij}$$

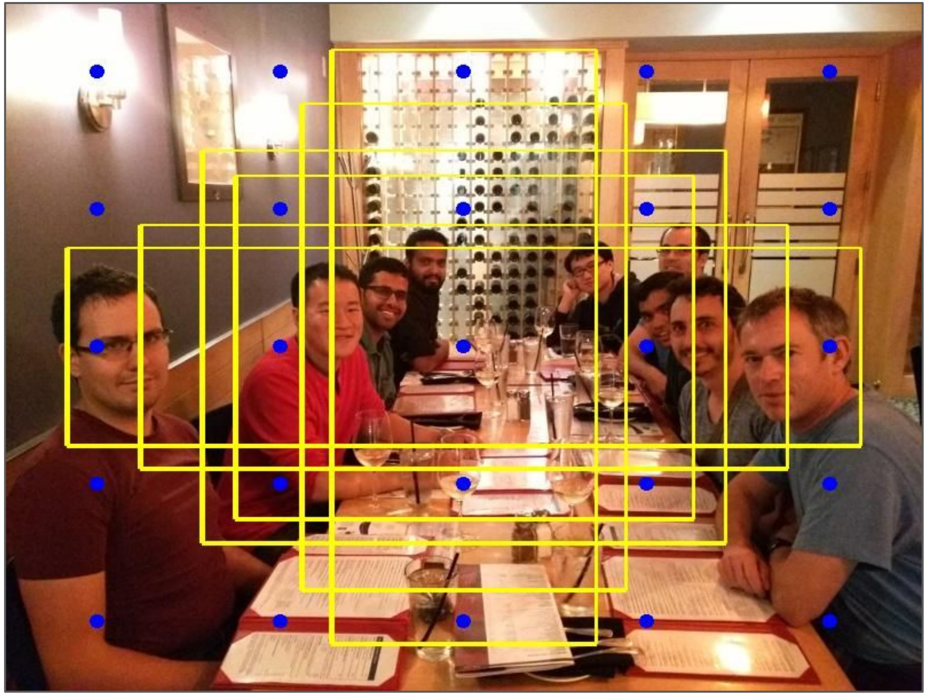
Use *convolution* to do simultaneous prediction for all anchors:

$$\texttt{Category score} = \texttt{SoftMax}(\texttt{Conv}(\mathbf{v};\ W^{cls}))$$

$$\texttt{Offset from anchor} = \texttt{Conv}(\mathbf{v};\ W^{loc})$$

But… if anchors need to be the same shape, how do we handle different scales/aspect ratios?

Solution: use multiple W^{loc} and W^{cls} (one for each aspect ratio/scale)



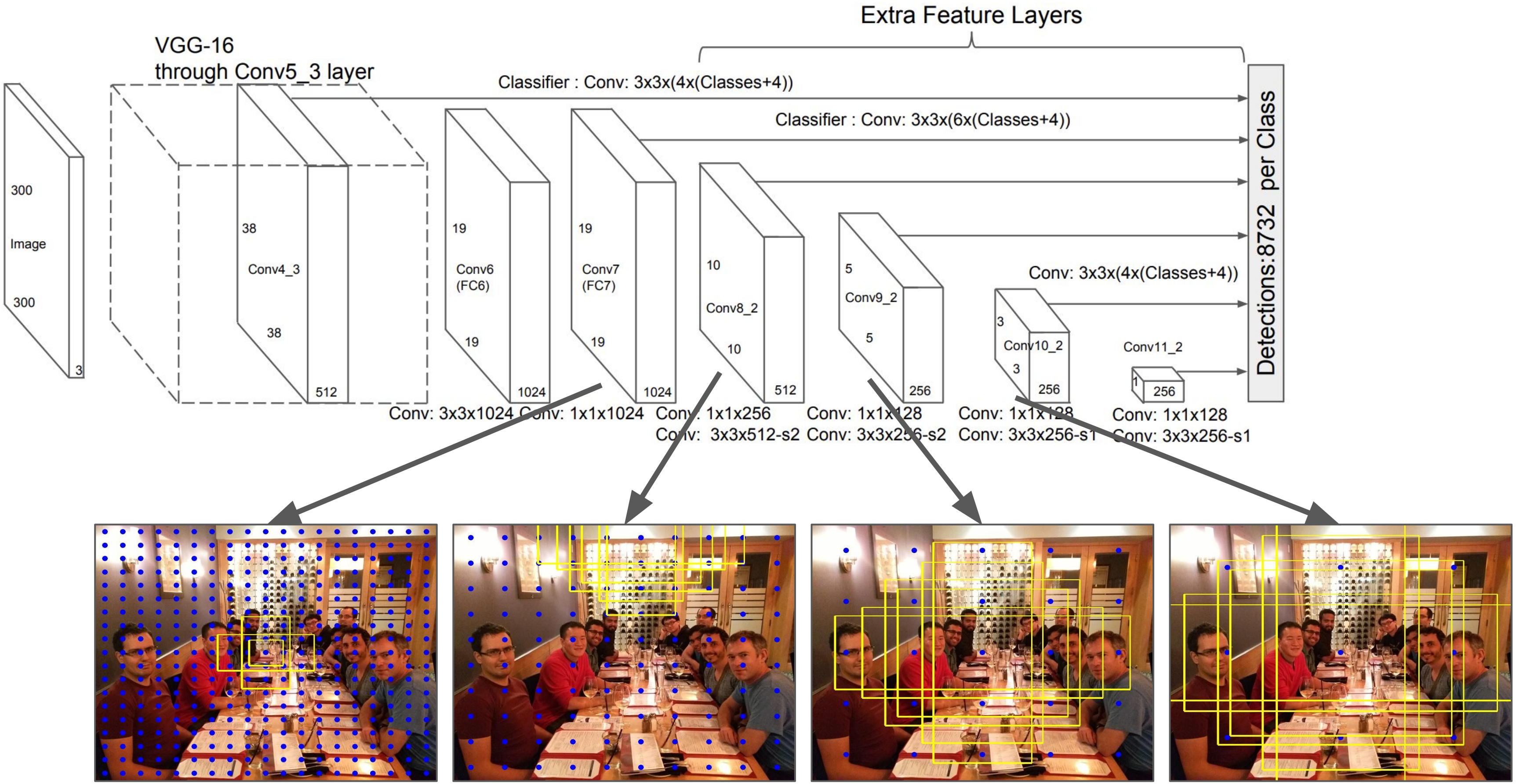
$\mathrm{SoftMax}(W^{cls,ar1} \cdot \mathbf{v}_{ij})$
$W^{loc,ar1} \cdot \mathbf{v}_{ij}$

$\mathrm{SoftMax}(W^{cls,ar2} \cdot \mathbf{v}_{ij})$
$W^{loc,ar2} \cdot \mathbf{v}_{ij}$

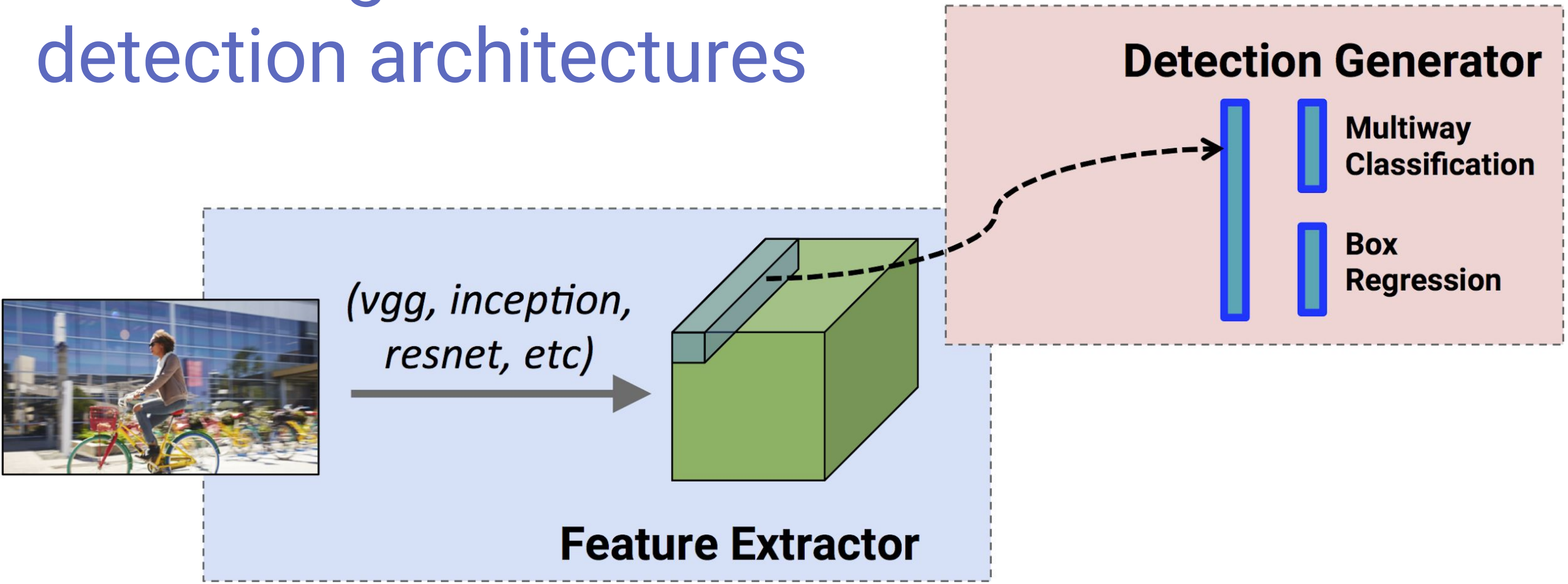
$\mathrm{SoftMax}(W^{cls,ar3} \cdot \mathbf{v}_{ij})$
$W^{loc,ar3} \cdot \mathbf{v}_{ij}$

...

Fancier Solution: use multiple anchor grid resolutions



Detection "***meta-architectures***" are a recipe for converting classification architectures into detection architectures

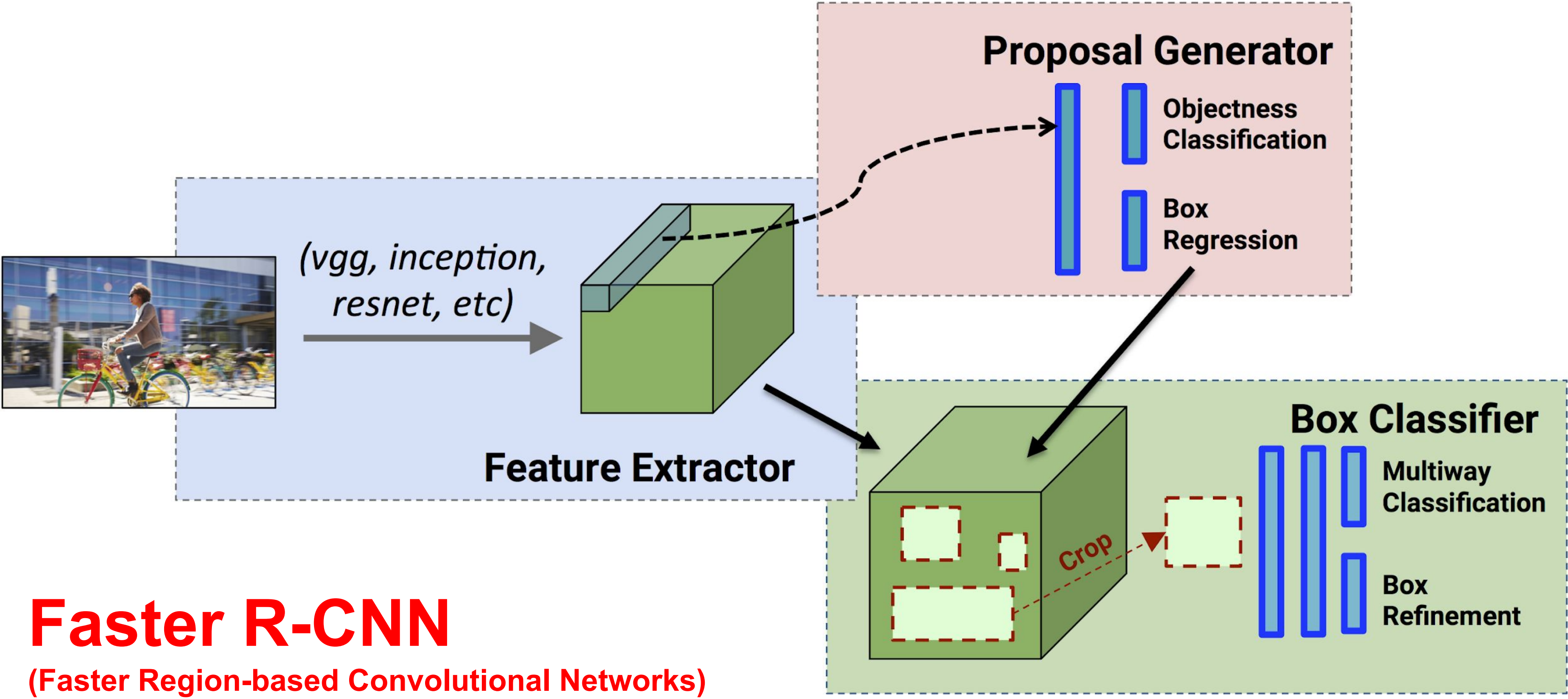


SSD

(Single Shot Detector --- encapsulates Multibox, YOLO, YOLO v2)

[Liu et al 2016]

Another popular meta-architecture

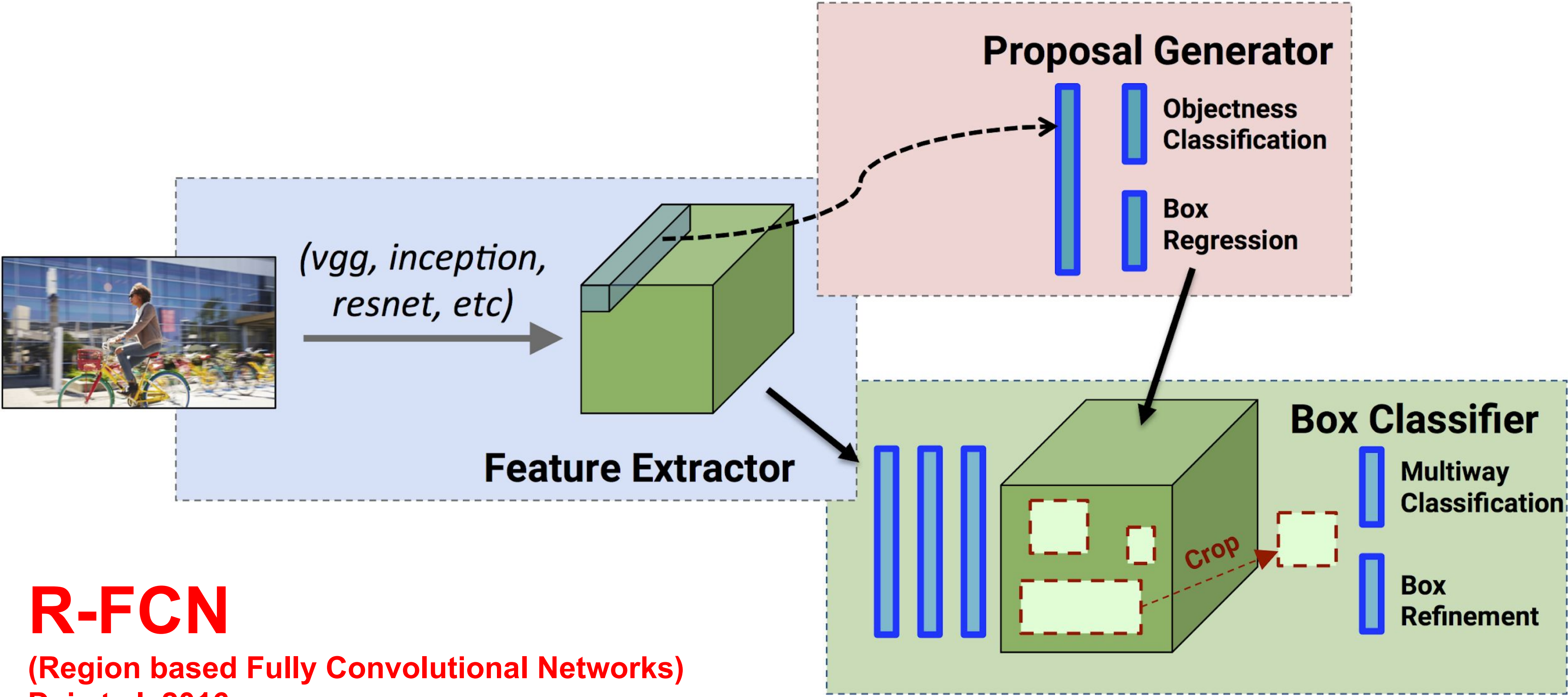


Faster R-CNN

(Faster Region-based Convolutional Networks)

[Ren et al 2015]

And yet another... but that's about it!



R-FCN

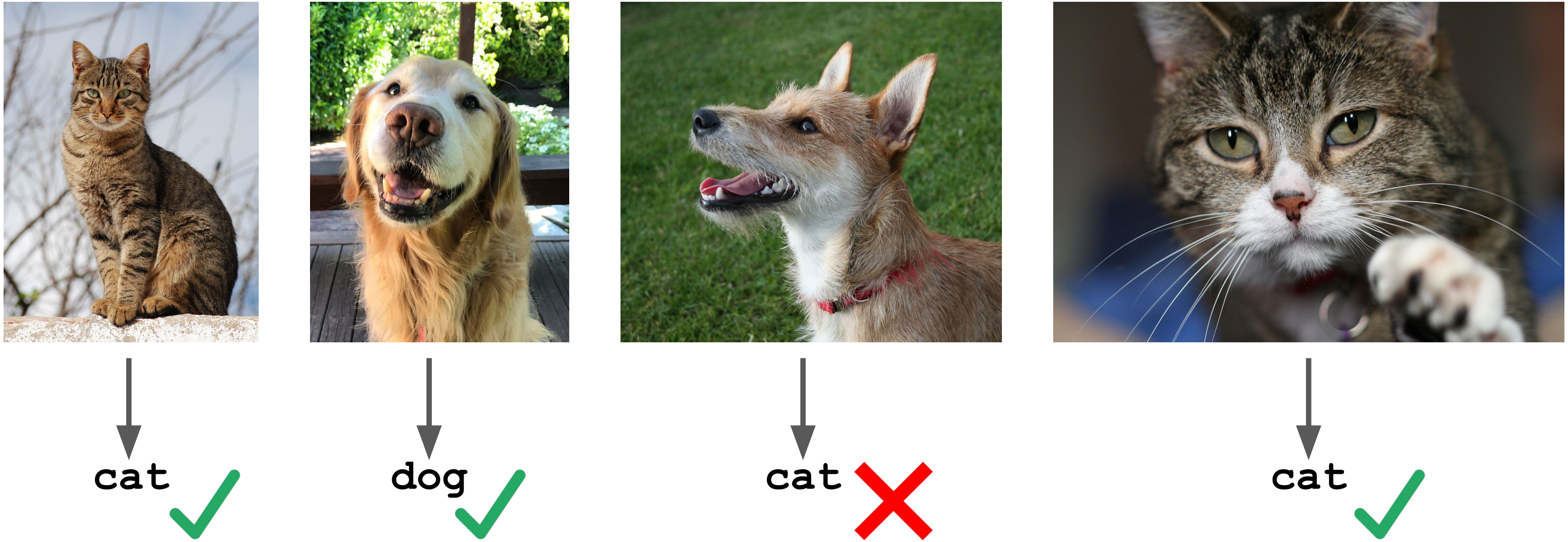
(Region based Fully Convolutional Networks)

Dai et al, 2016

Today

- Sliding Window Detectors
- Detection with Convolutional Networks
- **How to Evaluate a Detector**
- Practical tips/tricks
- Variations on a theme (instance segmentation, keypoint detection, video detection, etc...)

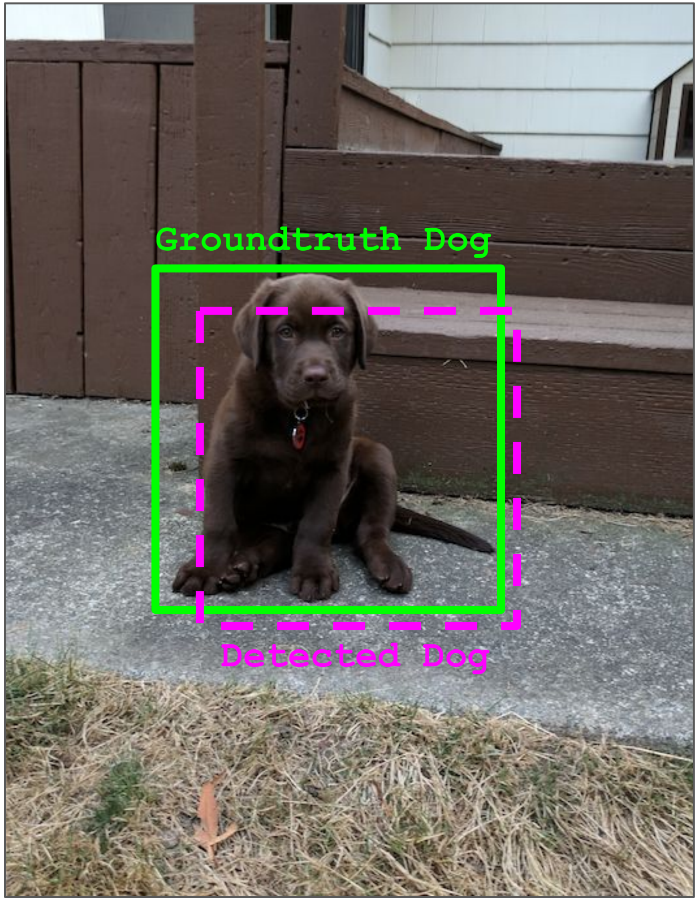
How do we know how good our model is?



Accuracy: 75%

For image classification, life is easy :)

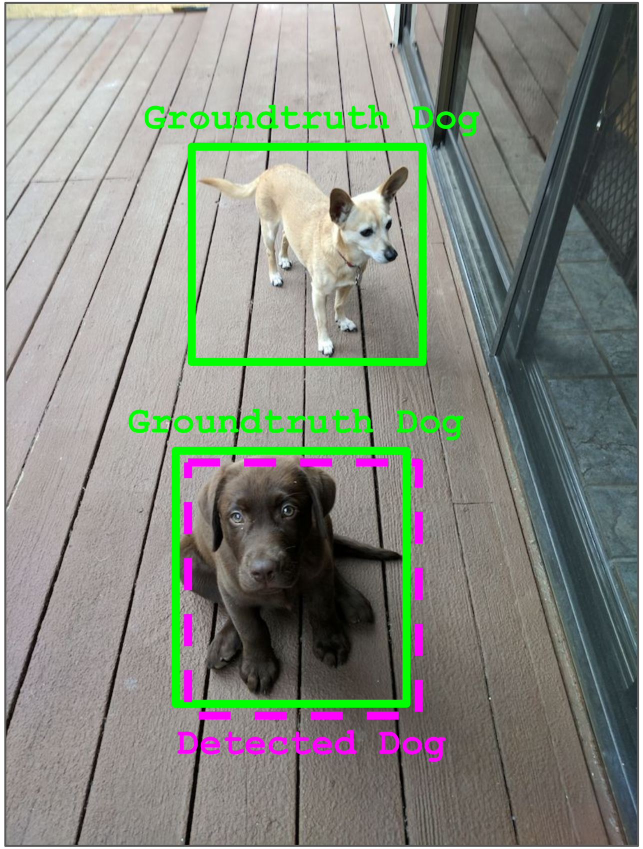
Evaluating Detectors is harder :(



Problem 1: Metrics must handle location errors

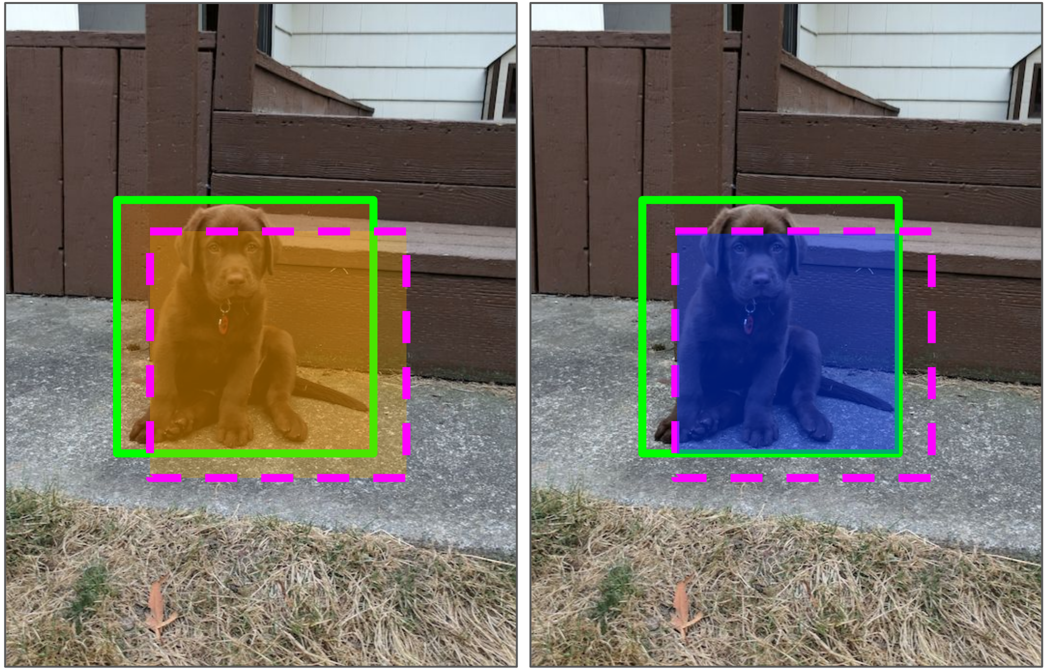
Should we consider this detection to be correct?

Evaluating Detectors is harder :(

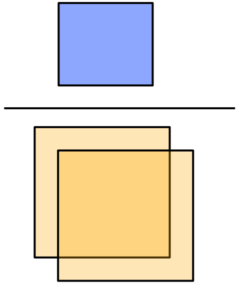


Problem 2: Metrics must account for overprediction and underprediction

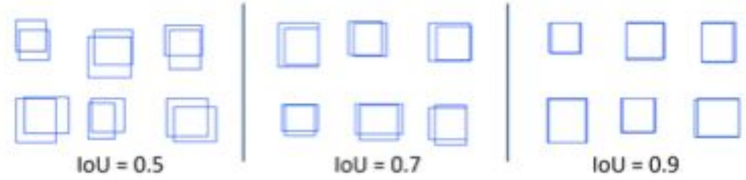
Intersection over Union (IOU)



IOU = $\frac{\text{Intersection}}{\text{Union}}$



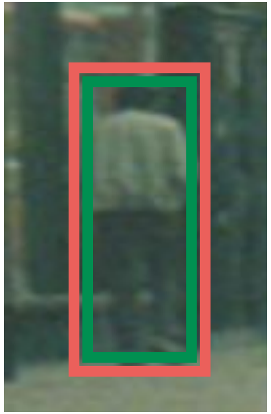
Detection is “correct” if IOU > α



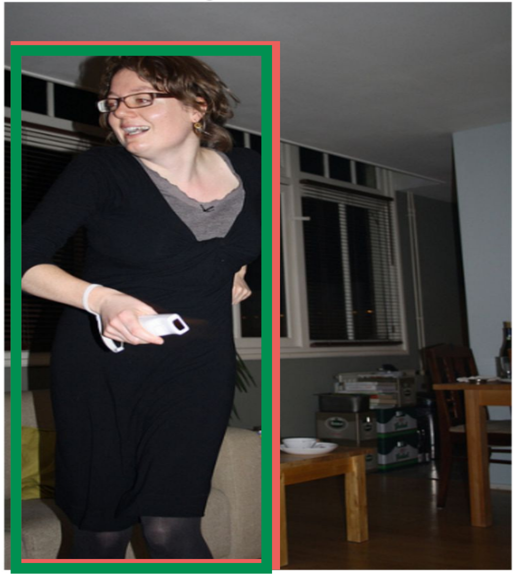
Intersection over Union (IOU)

IoU = 0.5

IoU = 0.7



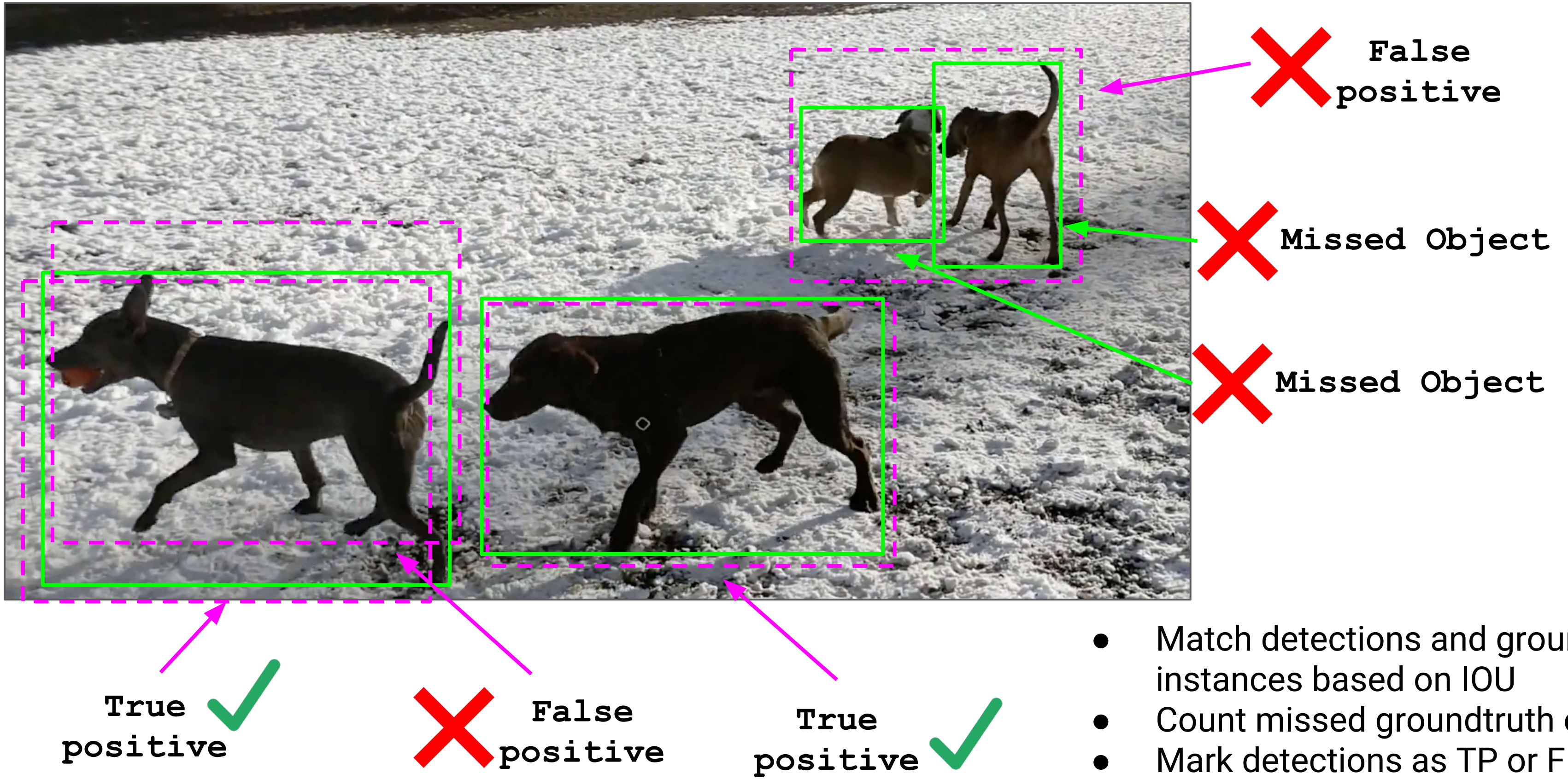
IoU = 0.95



Ground-Truth BBox   Detection BBox

Slide credit: http://image-net.org/challenges/talks/2016/ECCV2016_ilsvrc_coco_detection_segmentation.pdf

True/False Positives and Missed Objects



- Match detections and groundtruth instances based on IOU
- Count missed groundtruth objects
- Mark detections as TP or FP based on whether IOU>α

Summarizing Performance with Precision/Recall

Precision: Of the detections our model produced, how many were correct (i.e. True Positives)?

$$\text{Precision} = \frac{\#\text{TP}}{\#\text{TP} + \#\text{FP}}$$

Recall: Of the groundtruth instances in our data, what fraction of instances were correctly detected (i.e., not missed)?

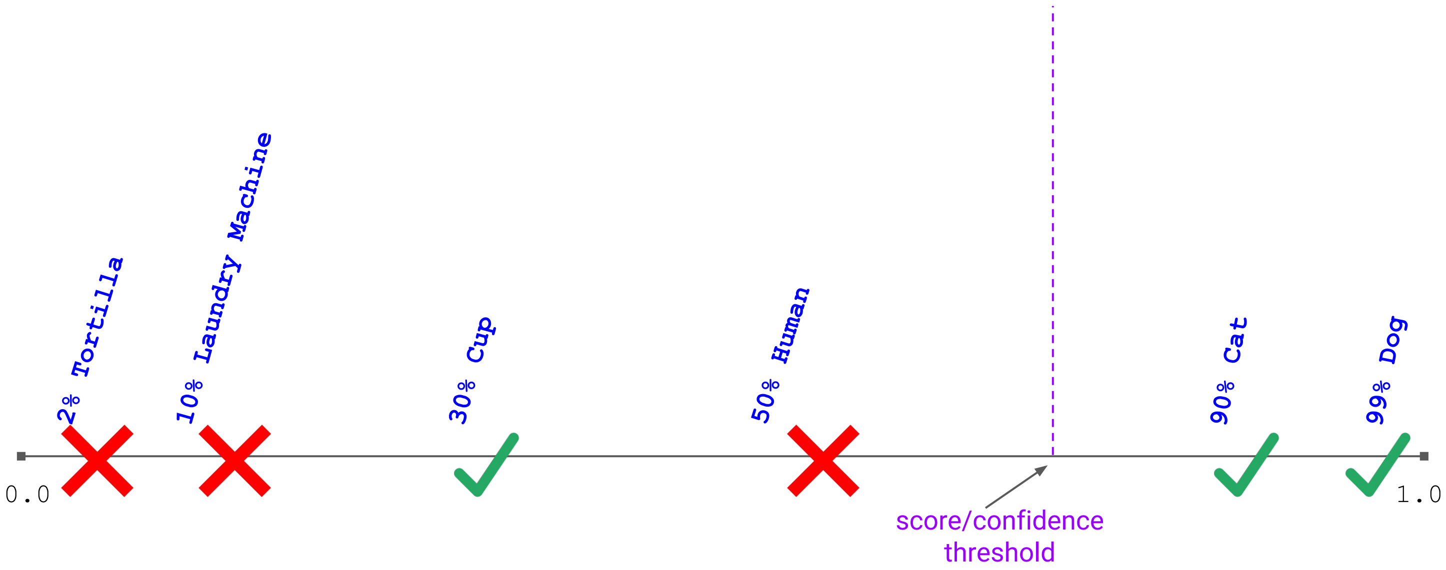
$$\text{Recall} = \frac{\#\text{TP}}{\#\text{Groundtruth Objects}}$$

Remember: Precision and Recall are in [0, 1] and higher is better.

Trading off between Precision and Recall

Detectors usually produce thousands of boxes (sliding windows), each with some score/confidence;

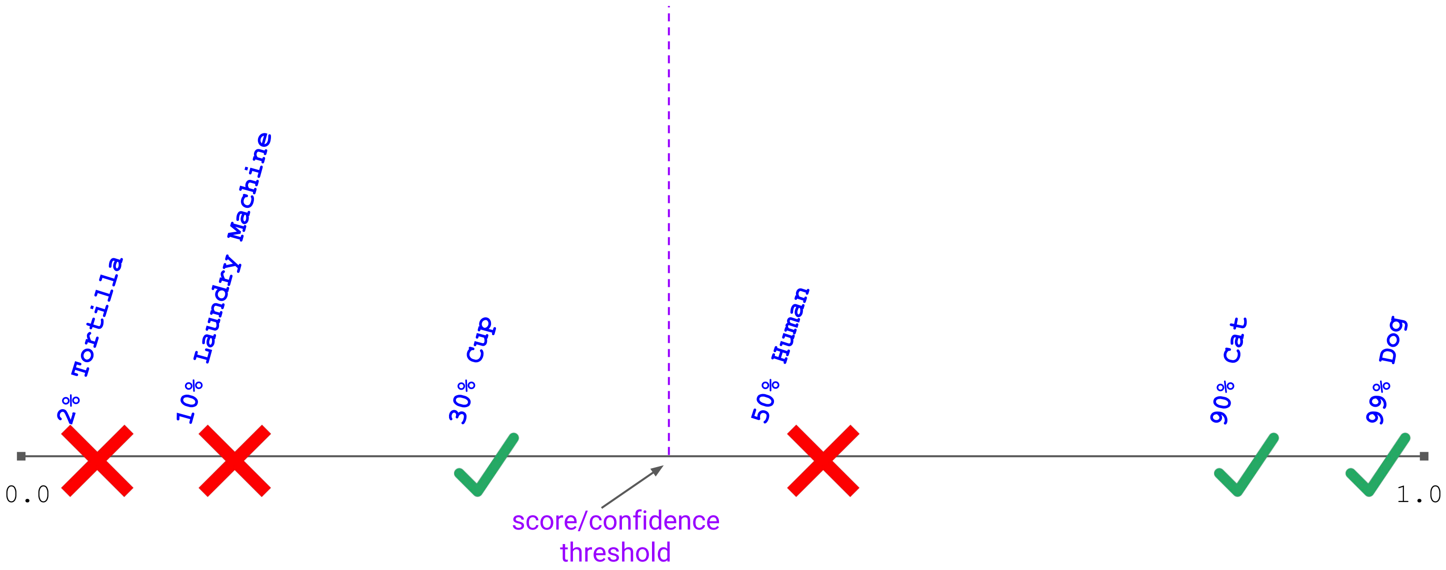
Last step of detection pipeline: *use score threshold to select final detections*



Trading off between Precision and Recall

Detectors usually produce thousands of boxes (sliding windows), each with some score/confidence;

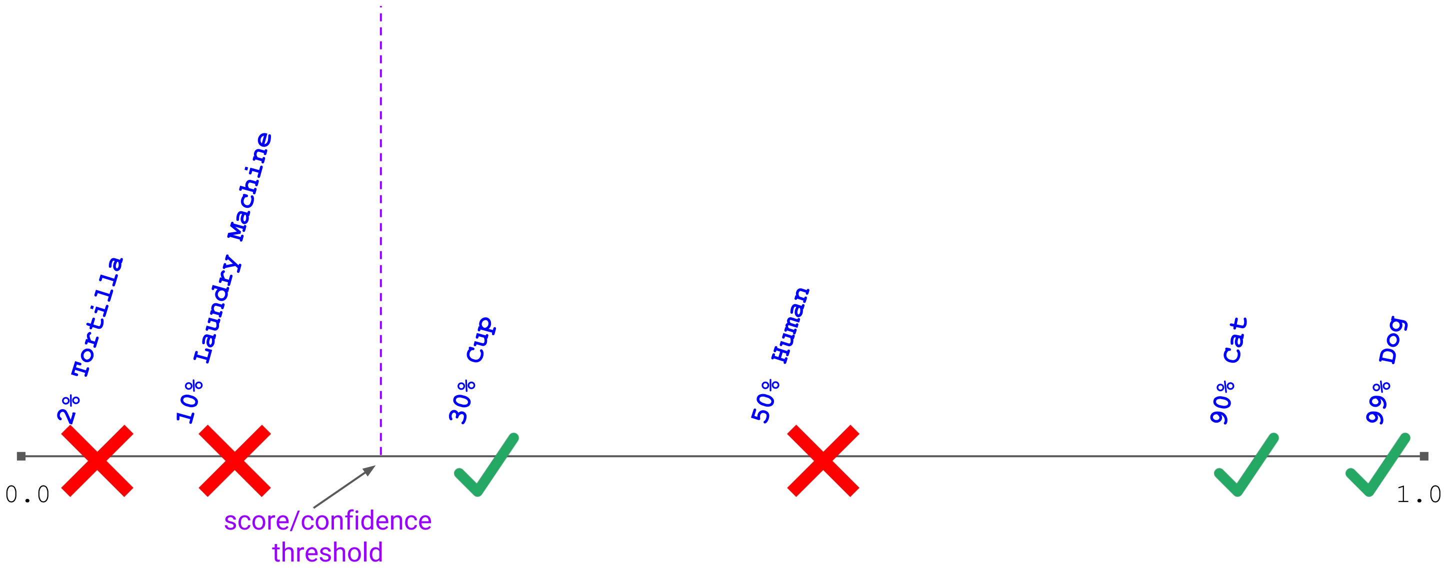
Last step of detection pipeline: *use score threshold to select final detections*



Trading off between Precision and Recall

Detectors usually produce thousands of boxes (sliding windows), each with some score/confidence;

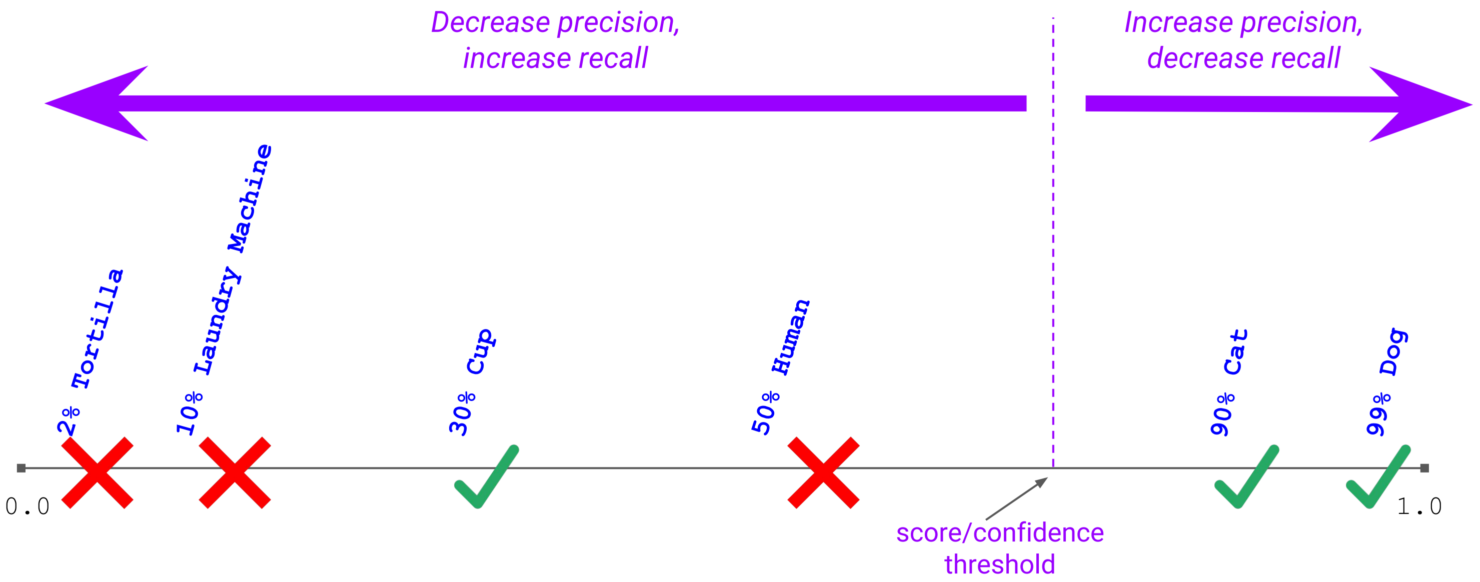
Last step of detection pipeline: *use score threshold to select final detections*



Trading off between Precision and Recall

Detectors usually produce thousands of boxes (sliding windows), each with some score/confidence;

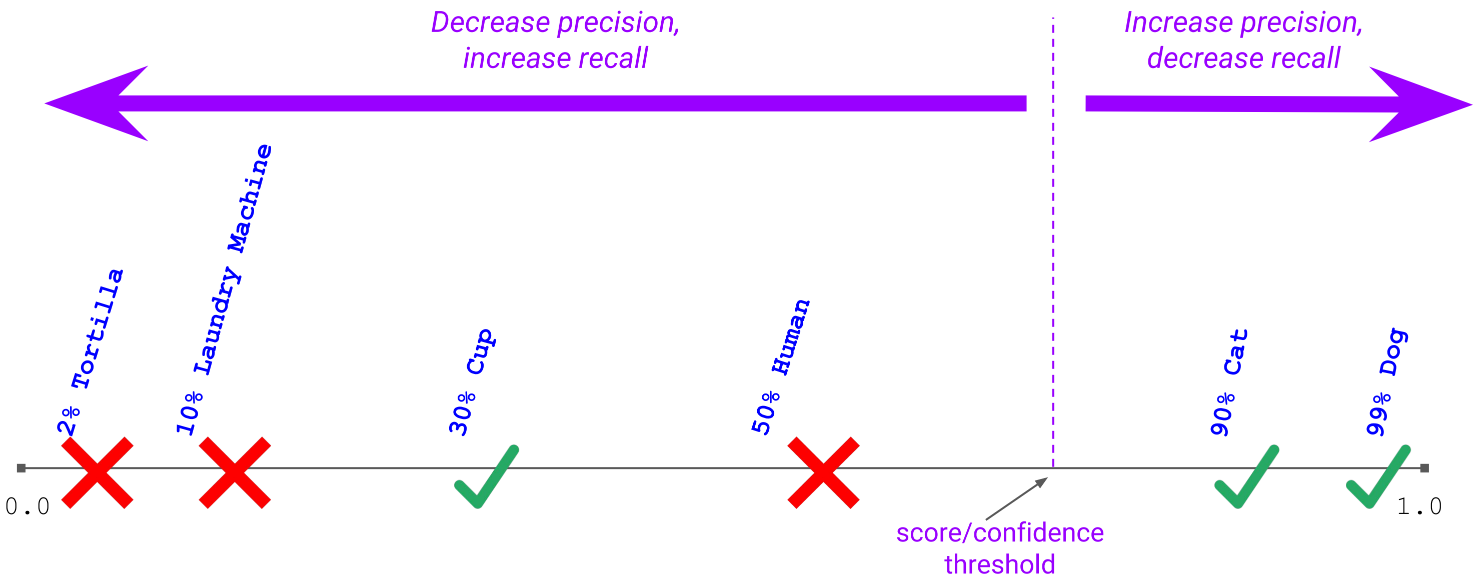
Last step of detection pipeline: *use score threshold to select final detections*



Trading off between Precision and Recall

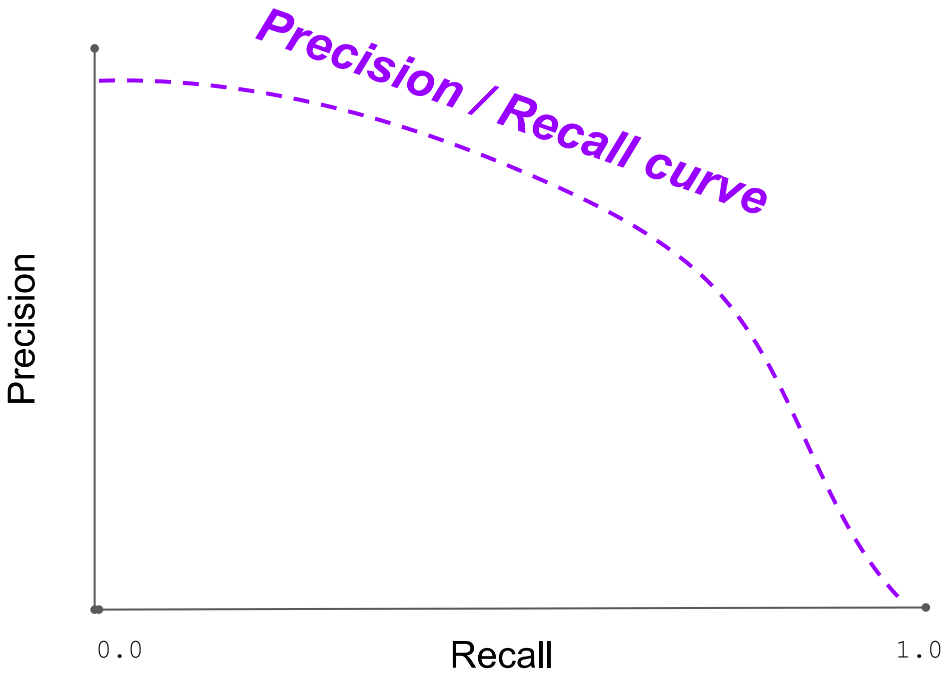
Detectors usually produce thousands of boxes (sliding windows), each with some score/confidence;

Last step of detection pipeline: *use score threshold to select final detections*

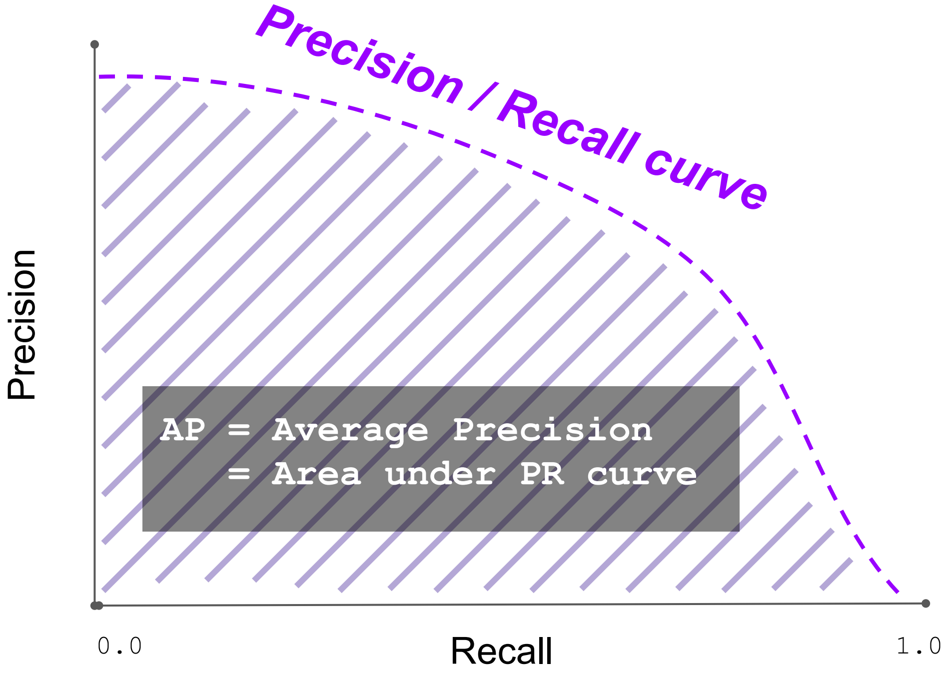


When would it be better to be on one side of this spectrum than the other?

Precision/Recall Curves and AP (Average Precision)



Precision/Recall Curves and AP (Average Precision)



Remember:

- *AP is always in [0, 1]*
- *Higher AP is better*
- *Always relative to an IOU criterion, e.g., AP@.5 IOU, AP@.75 IOU, etc...*

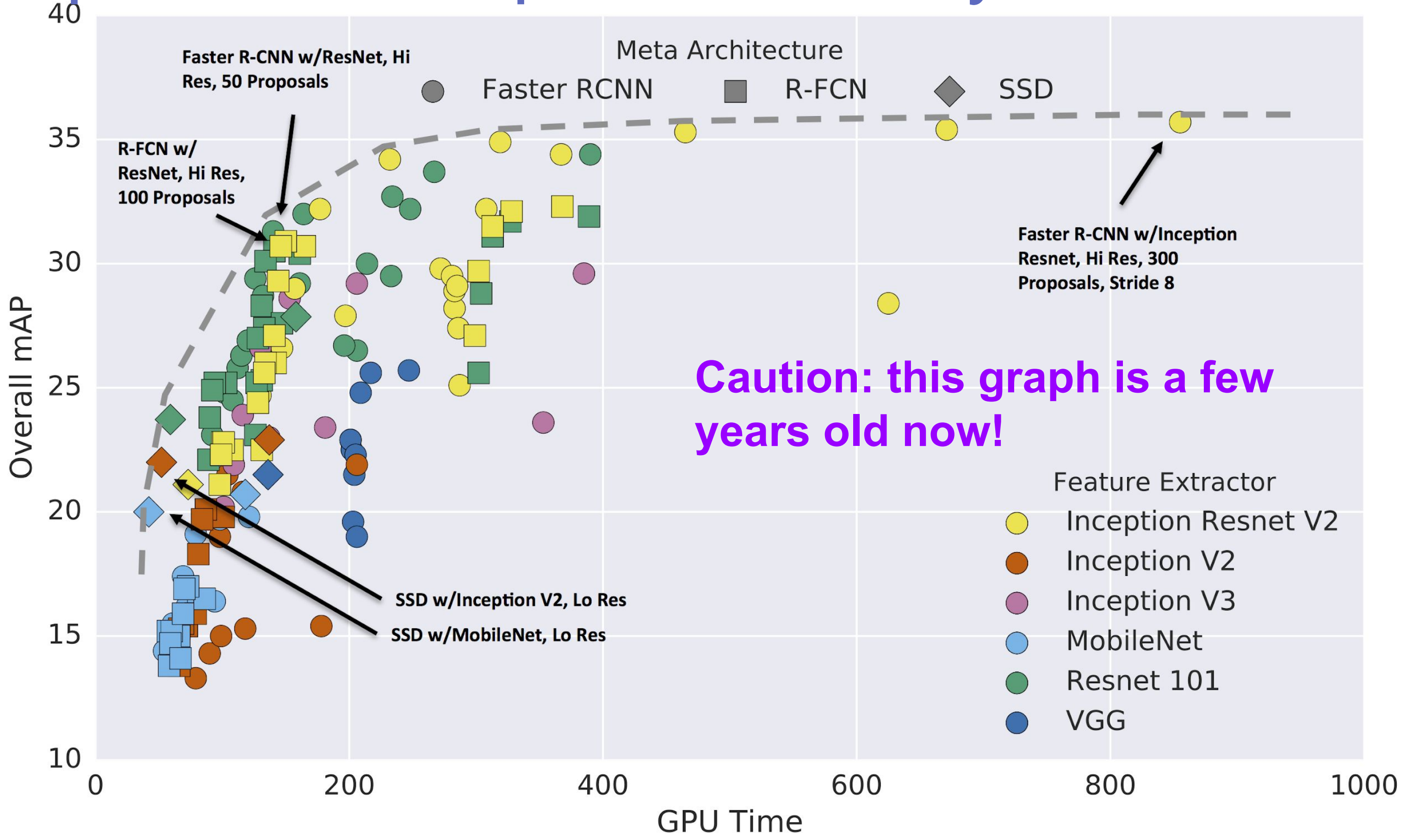
You should know:

- How to mark detections as True or False positives based on IOU
- What *Precision* and *Recall* mean
- And have some vague idea about how P-R Curves and Average Precision are computed :)

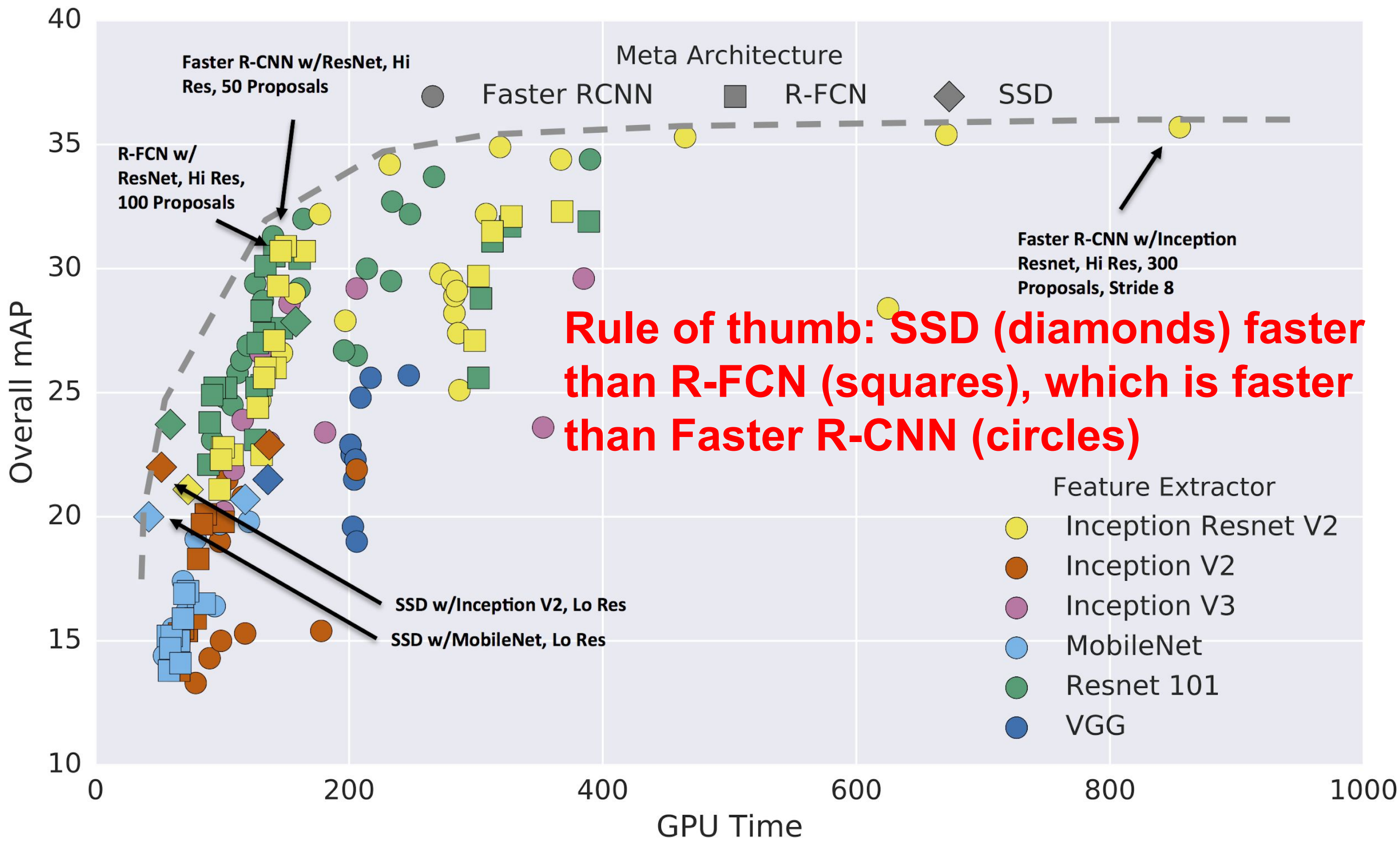
Today

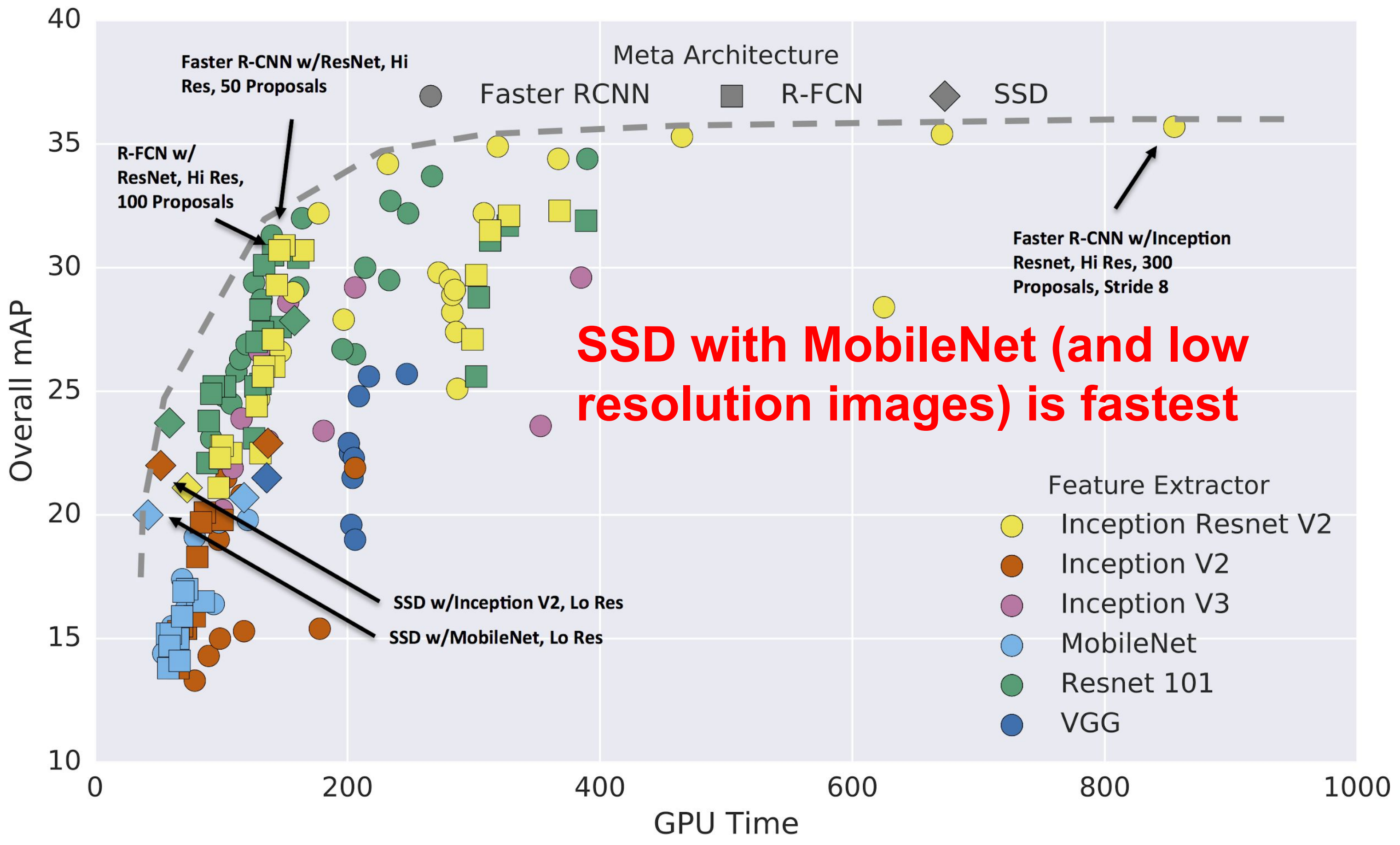
- Sliding Window Detectors
- Detection with Convolutional Networks
- How to Evaluate a Detector
- **Practical tips/tricks**
- Variations on a theme (instance segmentation, keypoint detection, video detection, etc...)

Pick a point on the speed/accuracy tradeoff curve

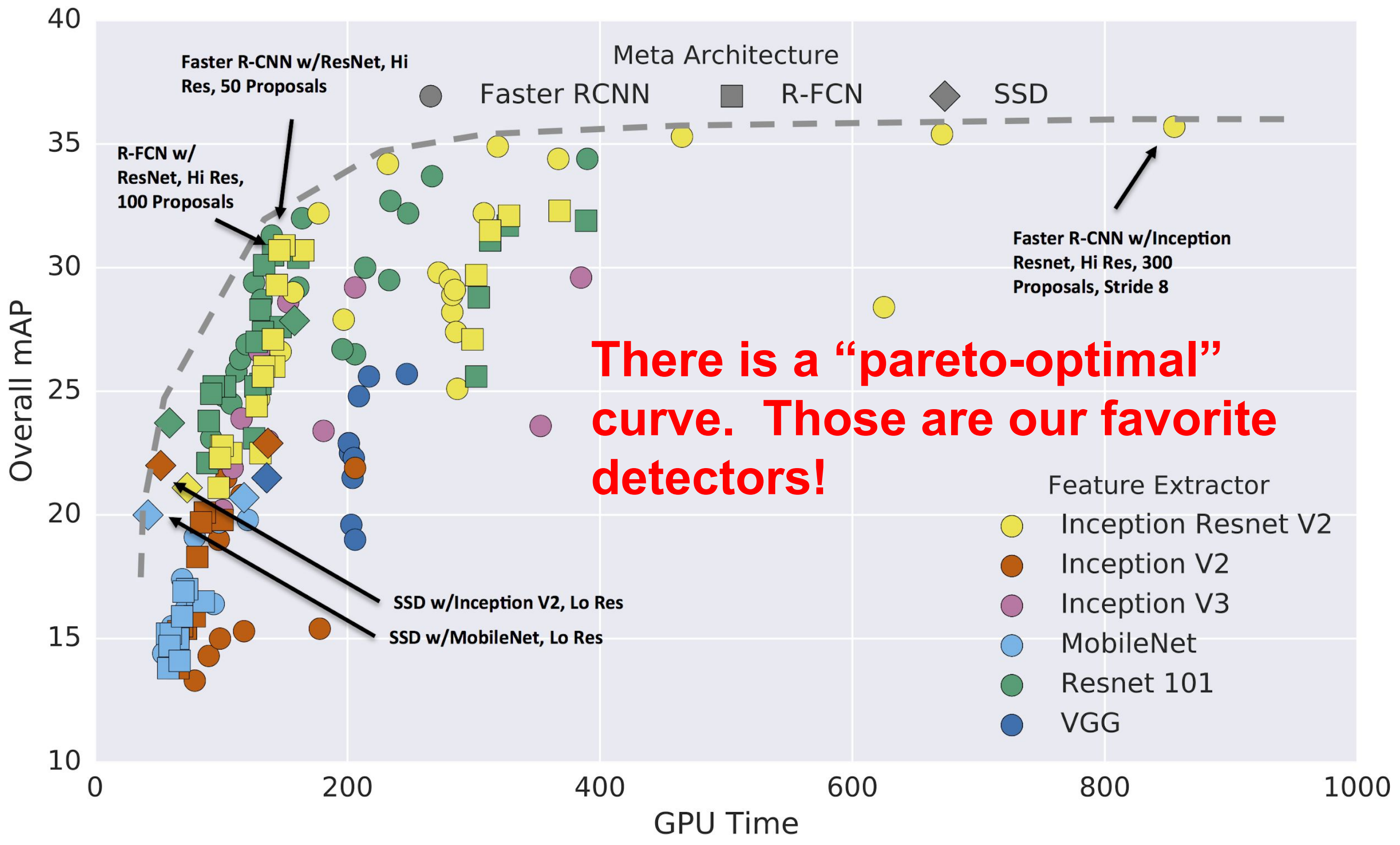


Huang, J., Rathod, V., Sun, C., Zhu, M., Korattikara, A., Fathi, A., ... Speed/accuracy trade-offs for modern convolutional object detectors. CVPR 2017


Meta Architecture
Faster RCNN
R-FCN
SSD
Faster R-CNN w/ResNet, Hi Res, 50 Proposals
R-FCN w/ ResNet, Hi Res, 100 Proposals
Faster R-CNN w/Inception Resnet, Hi Res, 300 Proposals, Stride 8
Rule of thumb: SSD (diamonds) faster than R-FCN (squares), which is faster than Faster R-CNN (circles)
SSD w/Inception V2, Lo Res
SSD w/MobileNet, Lo Res
Feature Extractor
Inception Resnet V2
Inception V2
Inception V3
MobileNet
Resnet 101
VGG
Overall mAP
GPU Time
40
35
30
25
20
15
10
0
200
400
600
800
1000

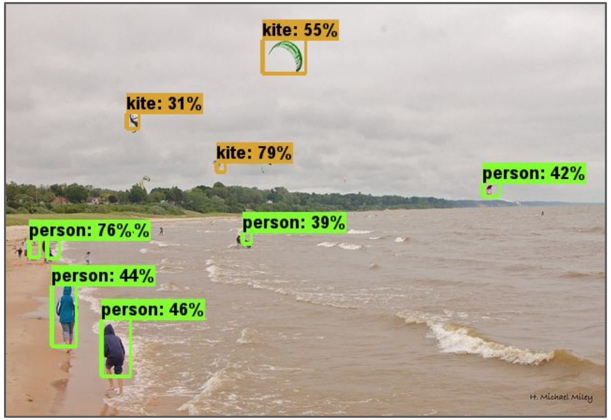


Meta Architecture
Faster RCNN
R-FCN
SSD
Faster R-CNN w/ResNet, Hi Res, 50 Proposals
R-FCN w/ ResNet, Hi Res, 100 Proposals
Faster R-CNN w/Inception Resnet, Hi Res, 300 Proposals, Stride 8
SSD with MobileNet (and low resolution images) is fastest
SSD w/Inception V2, Lo Res
SSD w/MobileNet, Lo Res
Feature Extractor
Inception Resnet V2
Inception V2
Inception V3
MobileNet
Resnet 101
VGG
Overall mAP
GPU Time
40
35
30
25
20
15
10
0
200
400
600
800
1000

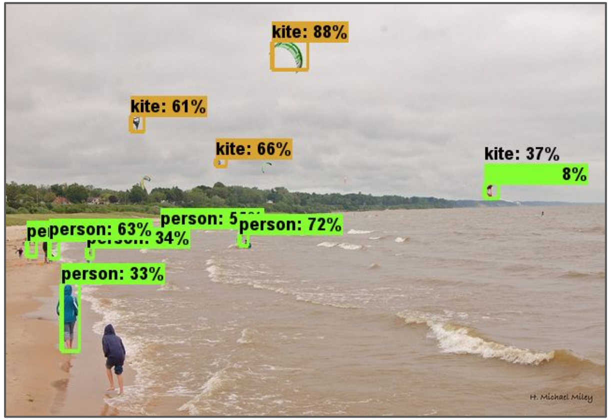


There is a "pareto-optimal" curve. Those are our favorite detectors!

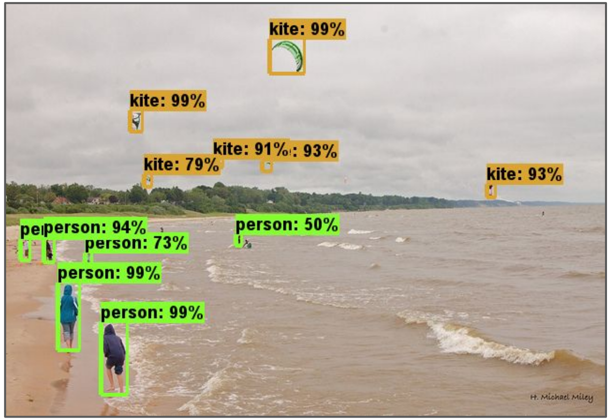
SSD w/MobileNet (Low Resolution)



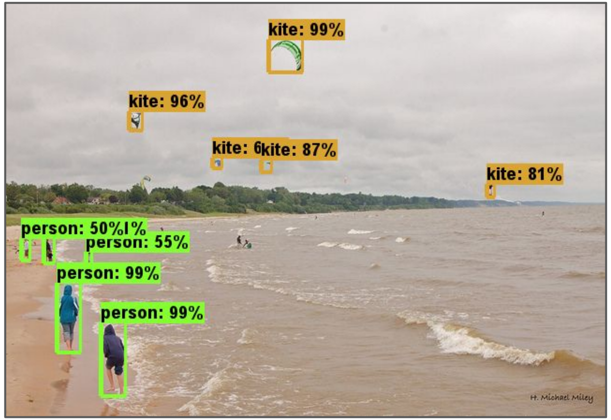
SSD w/Inception V2 (Low Resolution)



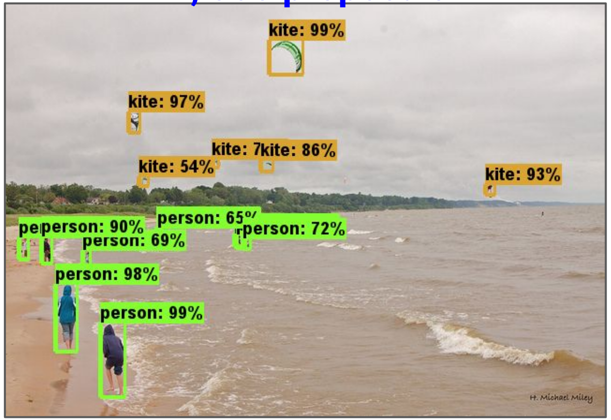
Faster R-CNN w/Resnet101, 100 proposals



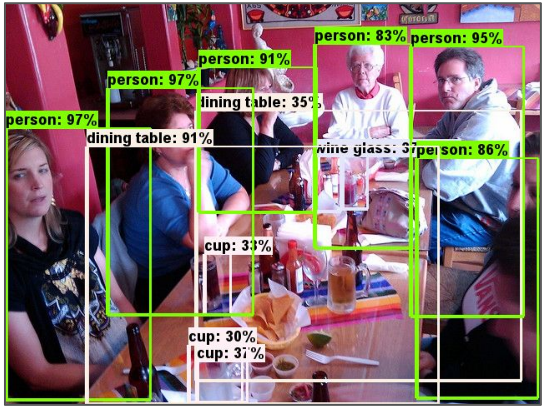
RFCN w/Resnet101, 300 proposals



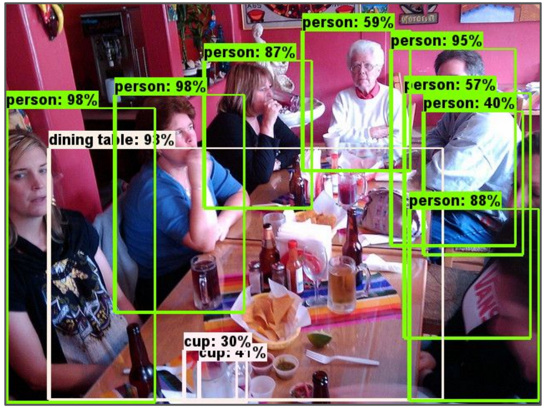
Faster R-CNN w/Inception Resnet V2, 300 proposals



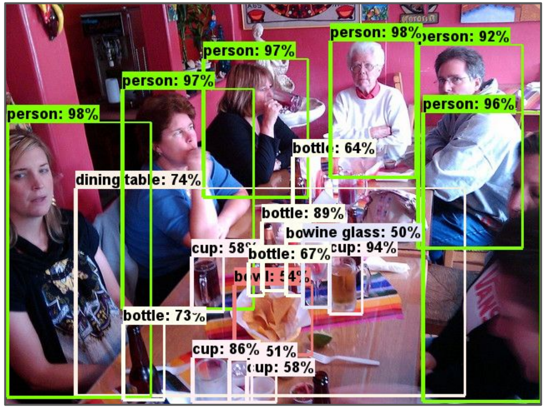
SSD w/MobileNet (Low Resolution)



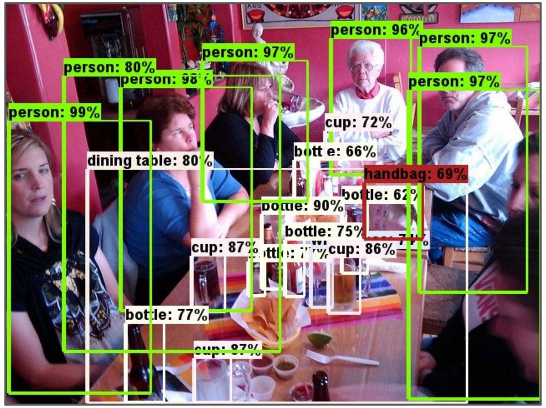
SSD w/Inception V2 (Low Resolution)



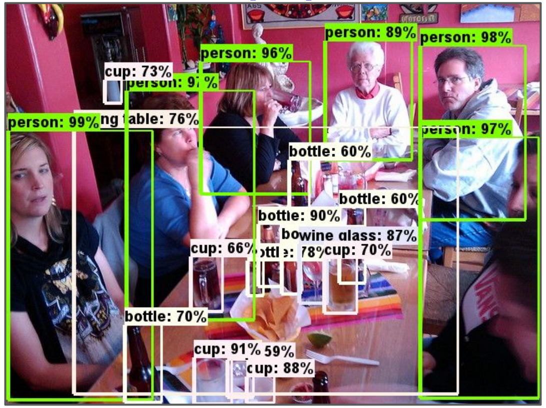
Faster R-CNN w/Resnet101, 100 proposals



RFCN w/Resnet101, 300 proposals

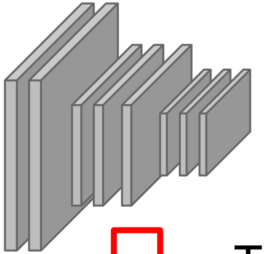


Faster R-CNN w/Inception Resnet V2, 300 proposals



Initialize from a model pre-trained to classify some other dataset (the larger the better)

JFT 300M

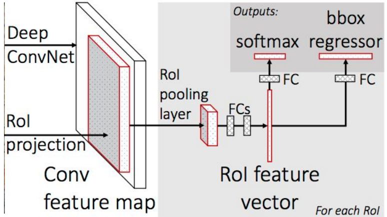


18K labels

Transfer weights

PASCAL2
Pattern Analysis, Statistical Modelling and Computational Learning

COCO
Common Objects in Context



Detections

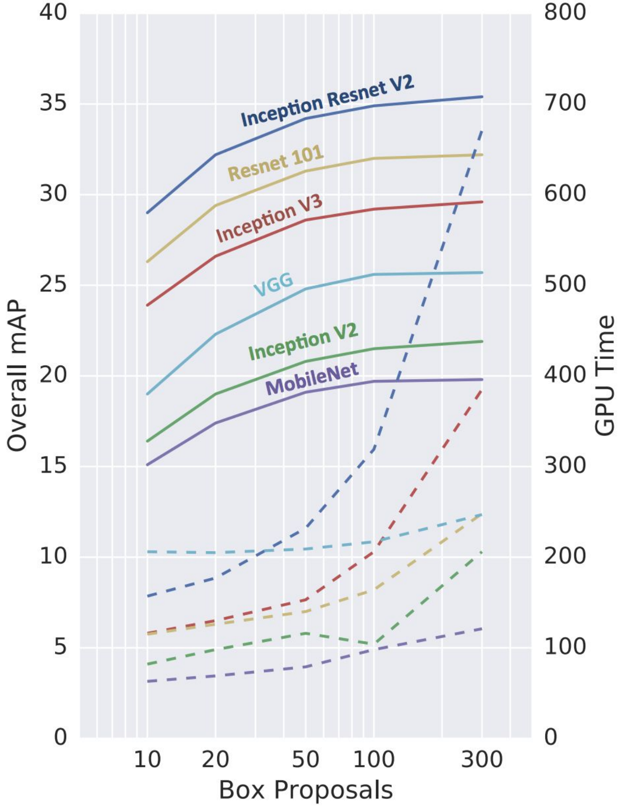
Method	mAP@0.5	mAP@[0.5,0.95]
He *et al.* [16]	53.3	32.2
ImageNet	53.6	34.3
300M	56.9	36.7
ImageNet+300M	**58.0**	**37.4**
Inception ResNet [37]	56.3	35.5

See "Revisiting Unreasonable Effectiveness of Data in Deep Learning Era" [Sun et al 2017]

Use lower resolution images for speed

Lower resolution much faster; tends to miss smaller objects

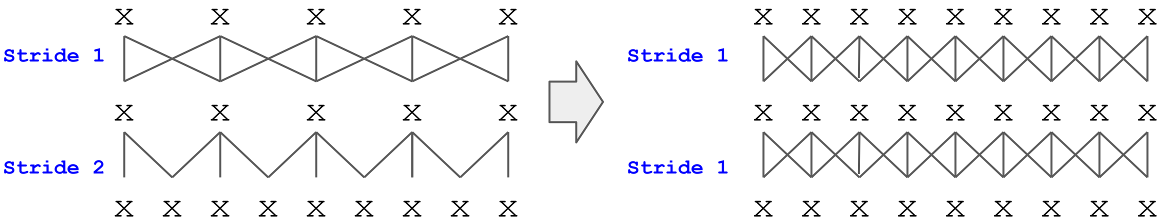
Use a small number of proposals for speed (for proposal based architectures)



Lower # of proposals much faster; sacrifices a bit of recall

Replace stride 2 convolutions with stride 1

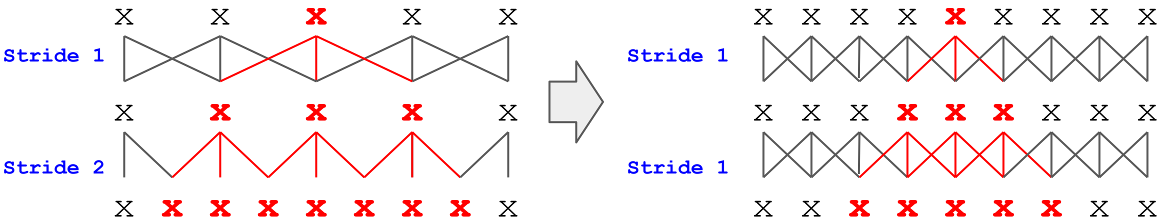
Slower, can boost performance on small objects



Problem: Doing this directly can reduce receptive field size...

Replace stride 2 convolutions with stride 1

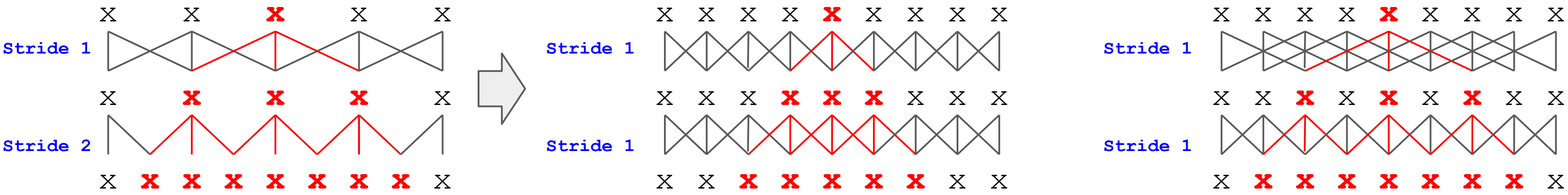
Slower, can boost performance on small objects



Problem: Doing this directly can reduce receptive field size...

Replace stride 2 convolutions with stride 1

Slower, can boost performance on small objects



Problem: Doing this directly can reduce receptive field size...

Solution: Use *atrous* convolution (convolution with holes) to compensate at the second layer.

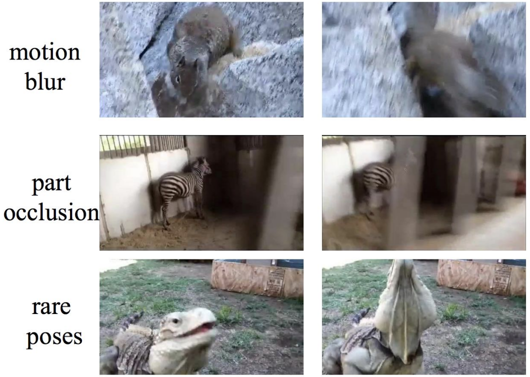
Today

- Sliding Window Detectors
- Detection with Convolutional Networks
- How to Evaluate a Detector
- Practical tips/tricks
- **Variations on a theme (instance segmentation, keypoint detection, video detection, etc...)**

Detection in Videos

Video vs static image detection:

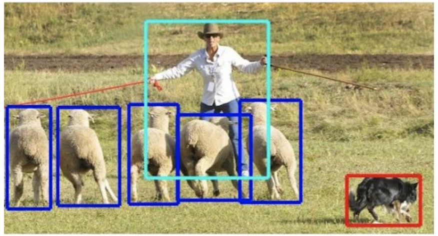
- Frames often deteriorated
- Adjacent frames are often near-identical; wasteful to run full detection every frame
- Useful to exploit motion cues



Video courtesy of Yuning Chai;
Image from Jifeng Dai

Instance Segmentation: the next step up from bounding boxes

classify



classify and regress
bounding box per object

(bounding box) detection

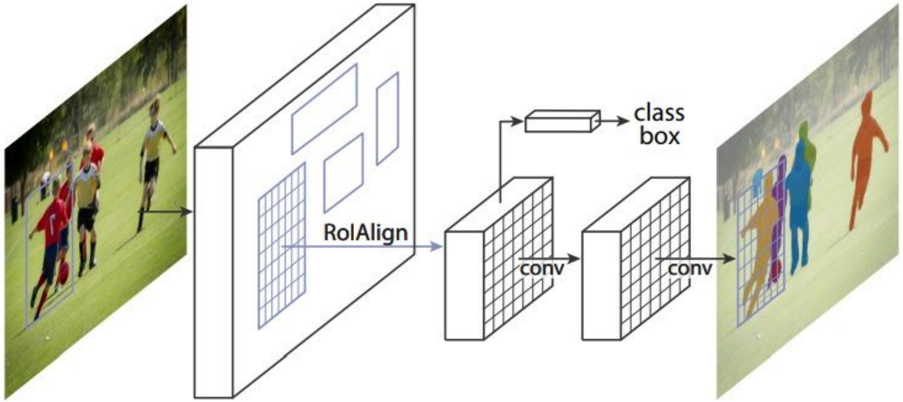
classify per pixel

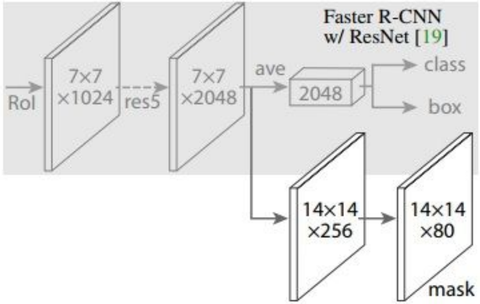
semantic segmentation

classify per pixel per object

instance segmentation

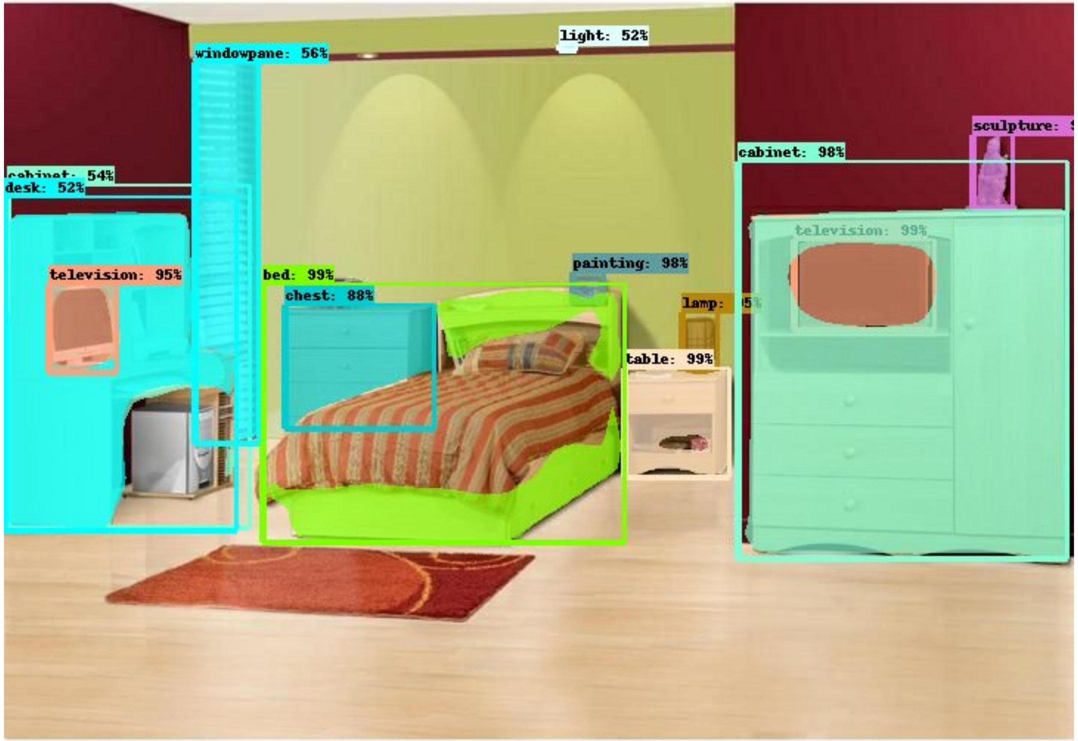
Mask R-CNN

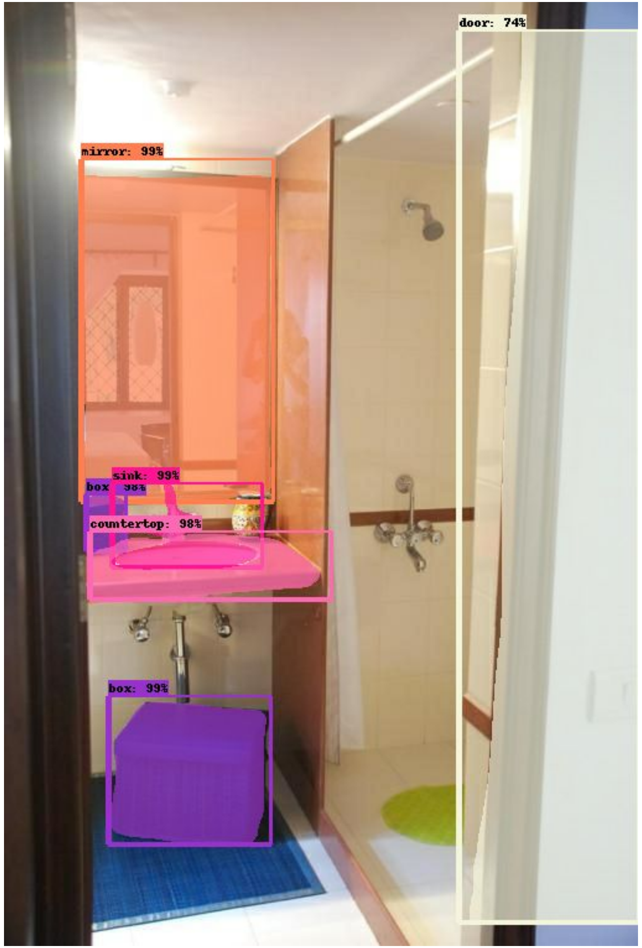




Mask R-CNN by He et al, 2017

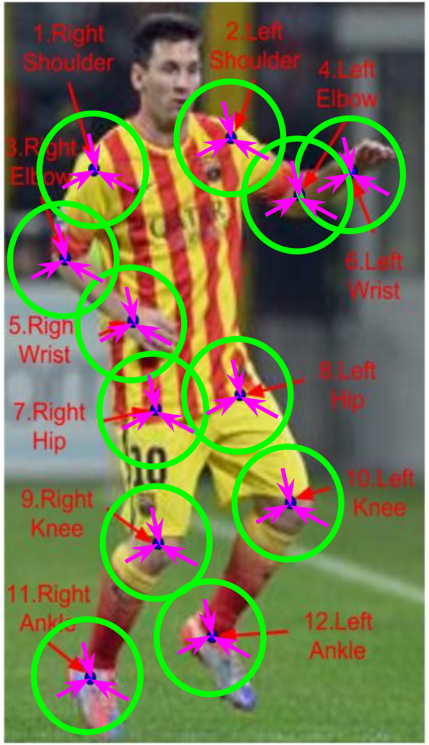
Example results from ADE20K





Slide courtesy of Alireza Fathi

Keypoint Detection



Slide courtesy of George Papandreou


imax PHOTGRAPHY
CHILE
brewers.com

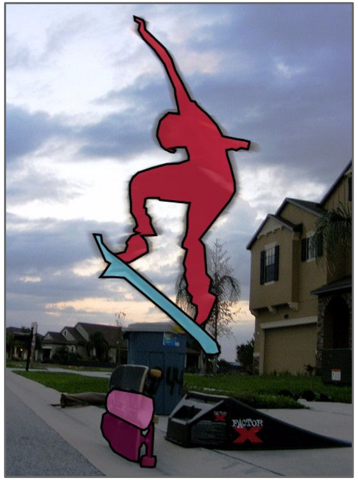
Learning with less supervision

Labeling is hard work!

COCO dataset:

- 200K labeled images
- 1.5 million object instances
- 80 object categories
- **~40 person-years of labeling time!**

Masks take ~x15 time to label compared to bounding boxes.



Can we learn to predict masks without explicit groundtruth mask annotations?

*Khoreva, Anna, et al. "Simple Does It: Weakly Supervised Instance and Semantic Segmentation." CVPR 2017

One idea: using “cut+paste” to get indirect feedback for mask predictions

Supervised question: “is this predicted mask correct?”

Weakly supervised question: “if I generate a new image by cut+pasting pixels inside the mask to a new part of the image, does it look plausible?”

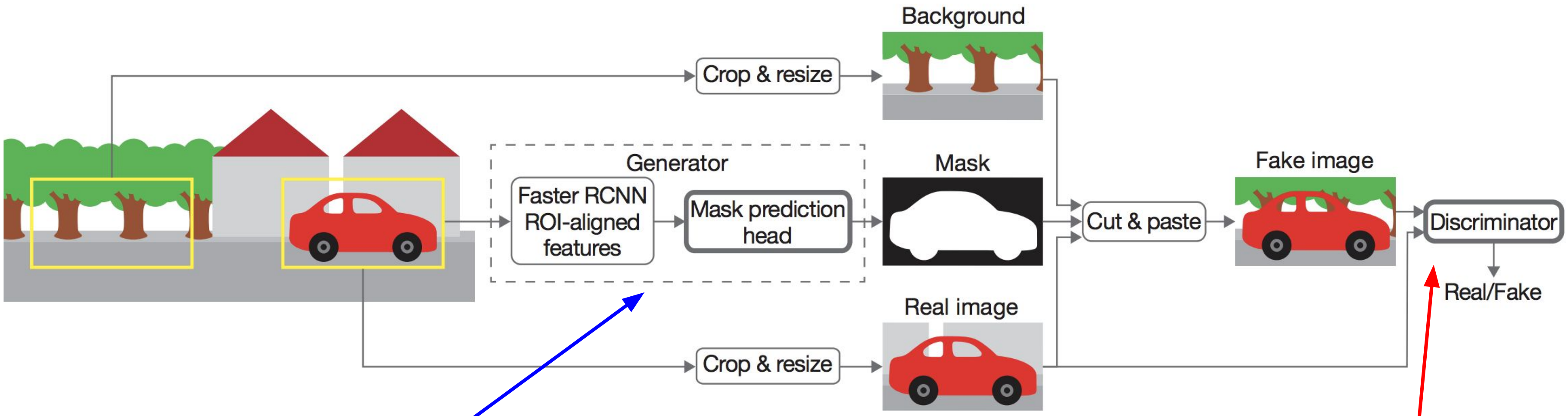
Bad mask estimate → Unrealistic Pasted Image

Good mask estimate → Realistic Pasted Image

Real Image for visual comparison

Remez, Huang, Brown. "Learning to Segment via Cut-and-Paste." (on arXiv)

Formalizing the Cut+Paste signal as a GAN (Generative Adversarial Network)

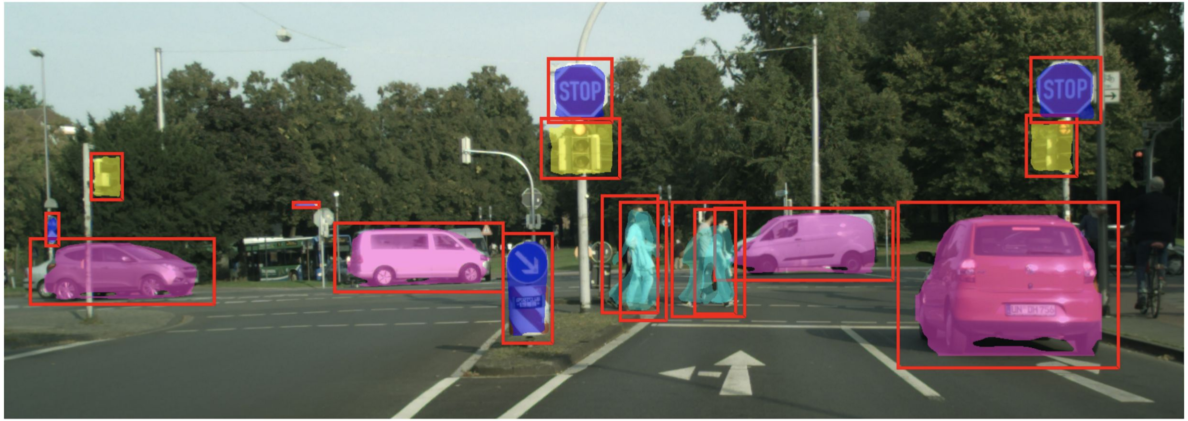


generator *receives a bounding box containing a car and predicts its mask*

discriminator *alternately sees car cut+pasted onto new background, or a real car image*

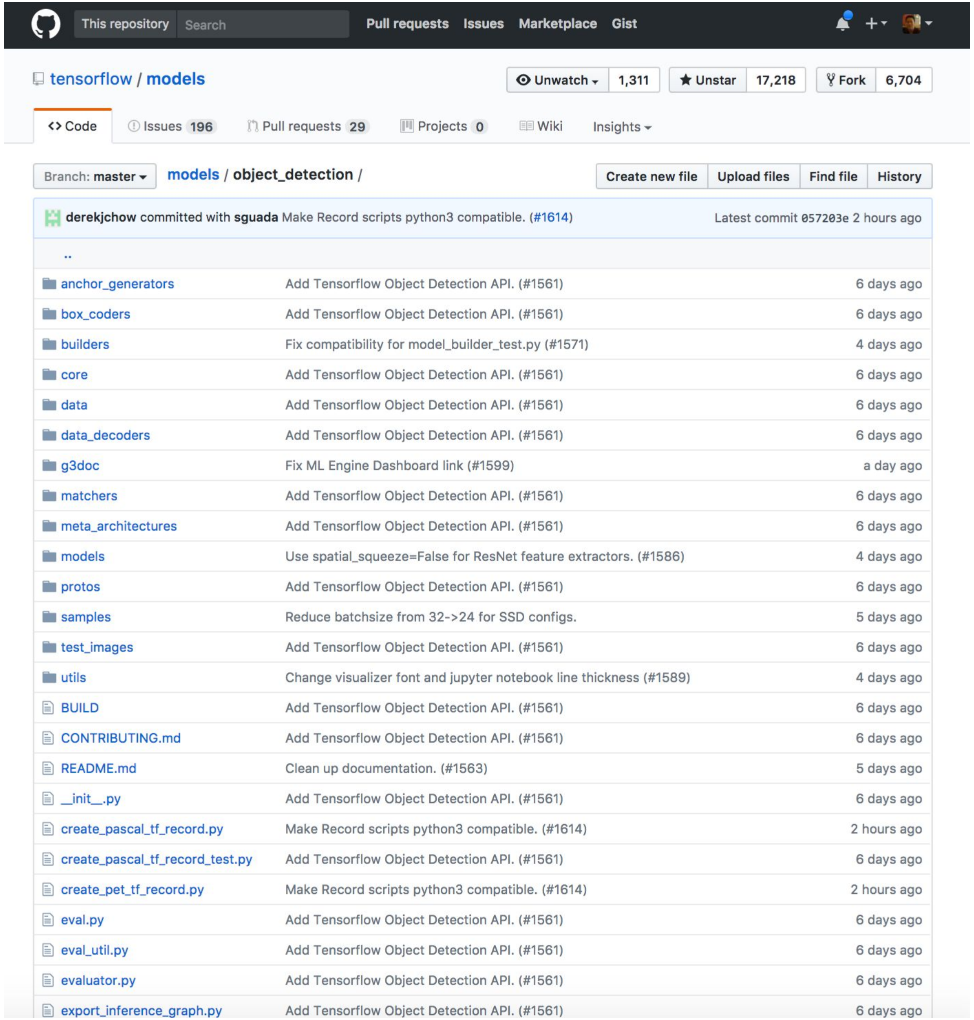
★ Both generator and discriminator are trained jointly.

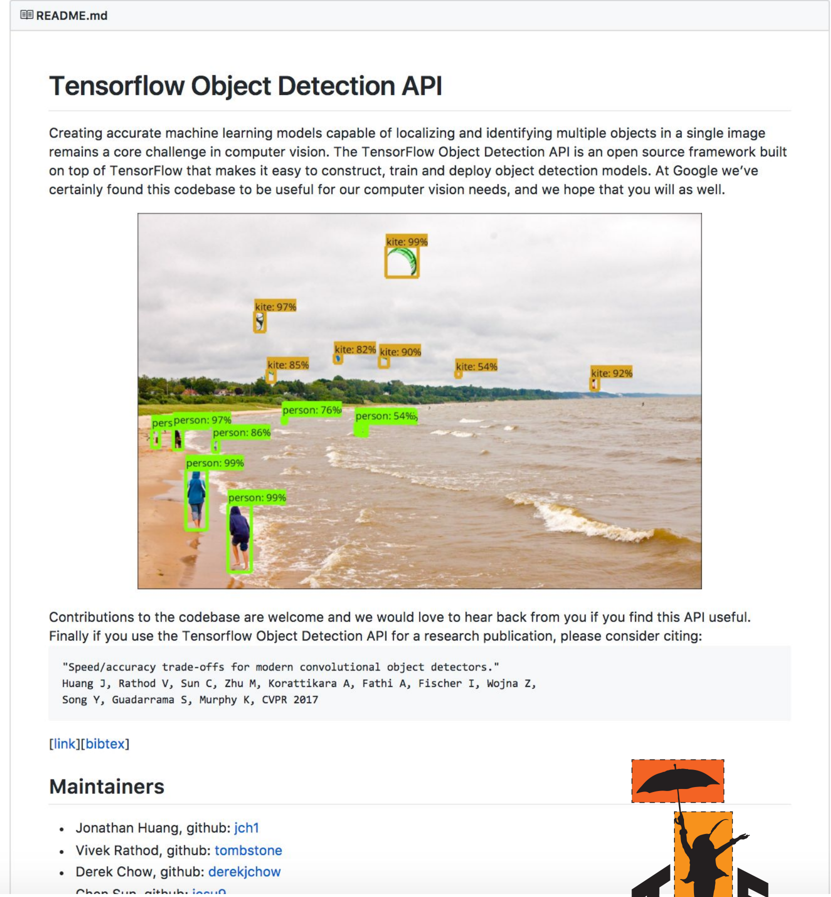
Mask R-CNN trained using Cut+Paste GAN



Summary

- Detectors are important and mature tech
- Sliding Window still the way to go
- Convnets can put the sliding in sliding window
- Detectors are evaluated with PR curves
- Bounding boxes are only the first step to complex scene understanding


This repository
Search
Pull requests
Issues
Marketplace
Gist
tensorflow / models
Unwatch 1,311
Unstar 17,218
Fork 6,704
Code
Issues 196
Pull requests 29
Projects 0
Wiki
Insights
Branch: master
models / object_detection /
Create new file
Upload files
Find file
History
derekjchow committed with sguada Make Record scripts python3 compatible. (#1614)
Latest commit 057203e 2 hours ago
anchor_generators Add Tensorflow Object Detection API. (#1561) 6 days ago
box_coders Add Tensorflow Object Detection API. (#1561) 6 days ago
builders Fix compatibility for model_builder_test.py (#1571) 4 days ago
core Add Tensorflow Object Detection API. (#1561) 6 days ago
data Add Tensorflow Object Detection API. (#1561) 6 days ago
data_decoders Add Tensorflow Object Detection API. (#1561) 6 days ago
g3doc Fix ML Engine Dashboard link (#1599) a day ago
matchers Add Tensorflow Object Detection API. (#1561) 6 days ago
meta_architectures Add Tensorflow Object Detection API. (#1561) 6 days ago
models Use spatial_squeeze=False for ResNet feature extractors. (#1586) 4 days ago
protos Add Tensorflow Object Detection API. (#1561) 6 days ago
samples Reduce batchsize from 32->24 for SSD configs. 5 days ago
test_images Add Tensorflow Object Detection API. (#1561) 6 days ago
utils Change visualizer font and jupyter notebook line thickness (#1589) 4 days ago
BUILD Add Tensorflow Object Detection API. (#1561) 6 days ago
CONTRIBUTING.md Add Tensorflow Object Detection API. (#1561) 6 days ago
README.md Clean up documentation. (#1563) 5 days ago
__init__.py Add Tensorflow Object Detection API. (#1561) 6 days ago
create_pascal_tf_record.py Make Record scripts python3 compatible. (#1614) 2 hours ago
create_pascal_tf_record_test.py Add Tensorflow Object Detection API. (#1561) 6 days ago
create_pet_tf_record.py Make Record scripts python3 compatible. (#1614) 2 hours ago
eval.py Add Tensorflow Object Detection API. (#1561) 6 days ago
eval_util.py Add Tensorflow Object Detection API. (#1561) 6 days ago
evaluator.py Add Tensorflow Object Detection API. (#1561) 6 days ago
export_inference_graph.py Add Tensorflow Object Detection API. (#1561) 6 days ago


README.md
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.
kite: 99%
kite: 97%
kite: 82%
kite: 90%
kite: 85%
kite: 54%
kite: 92%
person: 97%
person: 76%
person: 54%
person: 86%
person: 99%
person: 99%
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:
"Speed/accuracy trade-offs for modern convolutional object detectors."
Huang J, Rathod V, Sun C, Zhu M, Korattikara A, Fathi A, Fischer I, Wojna Z,
Song Y, Guadarrama S, Murphy K, CVPR 2017
[link][bibtex]
Maintainers
Jonathan Huang, github: jch1
Vivek Rathod, github: tombstone
Derek Chow, github: derekjchow

Configuring a model using the API

```
model {
 faster_rcnn {
  num_classes: 3
  image_resizer {
   keep_aspect_ratio_resizer {
    min_dimension: 600
    max_dimension: 1024
   }
  }
  feature_extractor {
   type: 'faster_rcnn_resnet101'
   first_stage_features_stride: 16
  }
...
```

{cars, people, stop signs}

high resolution input images

Faster R-CNN, Resnet 101

Configuring training using the API

```
train_config: {
 batch_size: 32
 fine_tune_checkpoint: "/home/jonathanhuang/..."
 optimizer {
  rms_prop_optimizer: {
   learning_rate: {
    exponential_decay_learning_rate {
     initial_learning_rate: 0.005
     decay_steps: 200000
     decay_factor: 0.95
    }
   }
   ...
```

pre-trained detection model (from COCO)

learning rate schedule

TF Object Detection API Model Zoo

COCO-trained models {#coco-models}

Model name	Speed (ms)	COCO mAP[^1]	Outputs
ssd_mobilenet_v1_coco	30	21	Boxes
ssd_mobilenet_v2_coco	31	22	Boxes
ssdlite_mobilenet_v2_coco	27	22	Boxes
ssd_inception_v2_coco	42	24	Boxes
faster_rcnn_inception_v2_coco	58	28	Boxes
faster_rcnn_resnet50_coco	89	30	Boxes
faster_rcnn_resnet50_lowproposals_coco	64		Boxes
rfcn_resnet101_coco	92	30	Boxes
faster_rcnn_resnet101_coco	106	32	Boxes
faster_rcnn_resnet101_lowproposals_coco	82		Boxes
faster_rcnn_inception_resnet_v2_atrous_coco	620	37	Boxes
faster_rcnn_inception_resnet_v2_atrous_lowproposals_coco	241		Boxes
faster_rcnn_nas	1833	43	Boxes
faster_rcnn_nas_lowproposals_coco	540		Boxes
mask_rcnn_inception_resnet_v2_atrous_coco	771	36	Masks
mask_rcnn_inception_v2_coco	79	25	Masks
mask_rcnn_resnet101_atrous_coco	470	33	Masks
mask_rcnn_resnet50_atrous_coco	343	29	Masks

Kitti-trained models {#kitti-models}

Model name	Speed (ms)	Pascal mAP@0.5	Outputs
faster_rcnn_resnet101_kitti	79	87	Boxes

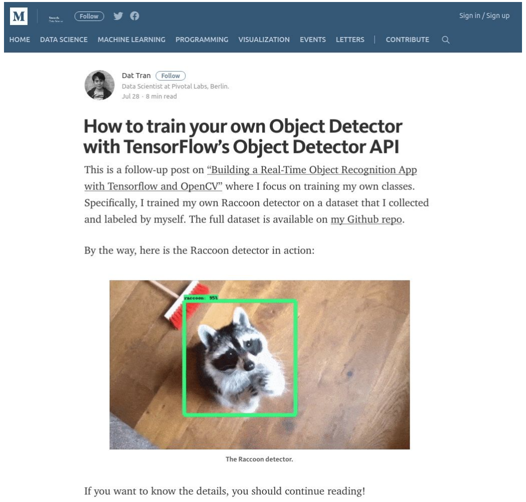
Open Images-trained models {#open-images-models}

Model name	Speed (ms)	Open Images mAP@0.5[^2]	Outputs
faster_rcnn_inception_resnet_v2_atrous_oid	727	37	Boxes
faster_rcnn_inception_resnet_v2_atrous_lowproposals_oid	347		Boxes

AVA v2.1 trained models {#ava-models}

Model name	Speed (ms)	Pascal mAP@0.5	Outputs
faster_rcnn_resnet101_ava_v2.1	93	11	Boxes

Community Creations!


HOME DATA SCIENCE MACHINE LEARNING PROGRAMMING VISUALIZATION EVENTS LETTERS CONTRIBUTE
Sign in / Sign up

Dat Tran Follow
Data Scientist at Pivotal Labs, Berlin.
Jul 28 · 8 min read

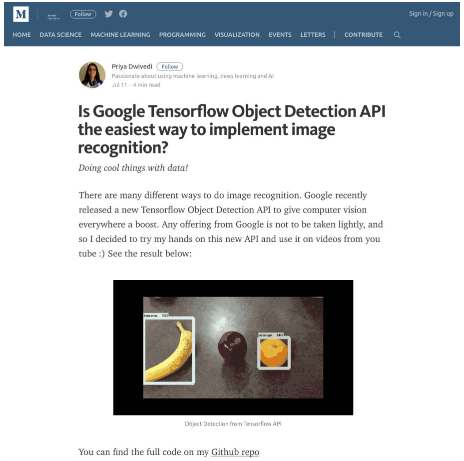
How to train your own Object Detector with TensorFlow's Object Detector API

This is a follow-up post on "Building a Real-Time Object Recognition App with Tensorflow and OpenCV" where I focus on training my own classes. Specifically, I trained my own Raccoon detector on a dataset that I collected and labeled by myself. The full dataset is available on my Github repo.

By the way, here is the Raccoon detector in action:

The Raccoon detector.

If you want to know the details, you should continue reading!


HOME DATA SCIENCE MACHINE LEARNING PROGRAMMING VISUALIZATION EVENTS LETTERS CONTRIBUTE
Sign in / Sign up

Priya Dwivedi Follow
Passionate about using machine learning, deep learning and AI
Jul 11 · 4 min read

Is Google Tensorflow Object Detection API the easiest way to implement image recognition?

Doing cool things with data!

There are many different ways to do image recognition. Google recently released a new Tensorflow Object Detection API to give computer vision everywhere a boost. Any offering from Google is not to be taken lightly, and so I decided to try my hands on this new API and use it on videos from you tube :) See the result below:

Object Detection from Tensorflow API

You can find the full code on my Github repo

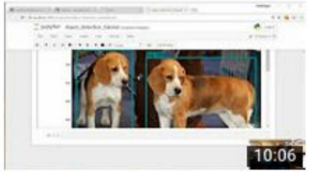
Testing Custom Object Detector - TensorFlow Object Detection A...
4,524 views • 3 days ago

Training Custom Object Detector - TensorFlow Object Detection
3,007 views • 3 days ago

Creating TFRecords - TensorFlow Object Detection API Tutorial p.4
3,145 views • 3 days ago

Tracking Custom Objects - TensorFlow Object Detection A...
4,600 views • 3 days ago

Adapting to video feed - TensorFlow Object Detection A...
15,157 views • 6 days ago



Intro - TensorFlow Object Detection API Tutorial p.1
16,571 views • 1 week ago

Thanks!