HA NOI UNIVERSITY OF SCIENCE AND TECHNOLOGY
SCHOOL OF INFORMATION AND COMMUNICATION TECHNOLOGY

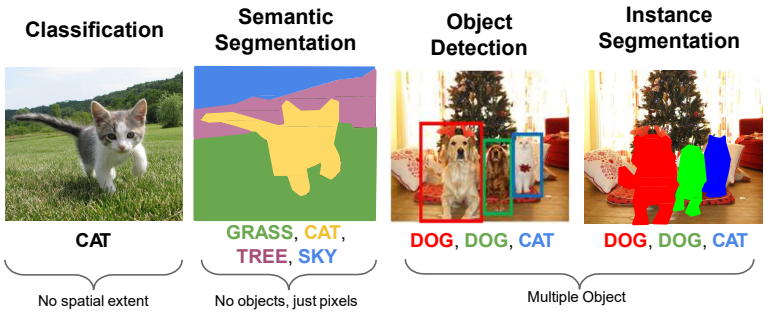
Computer Vision

Chapter 7 (part 3):
Deep Learning for CV

Deep learning for CV

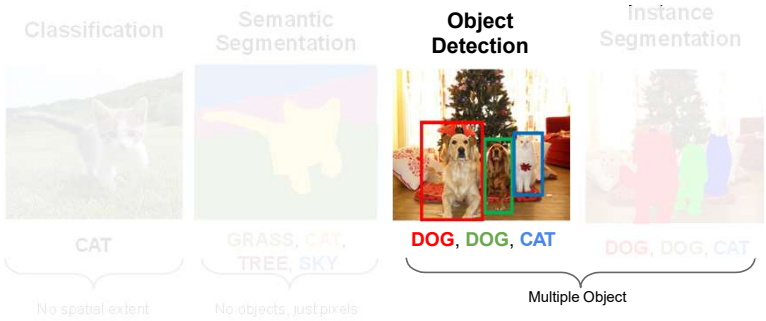
Contents

Computer Vision Tasks

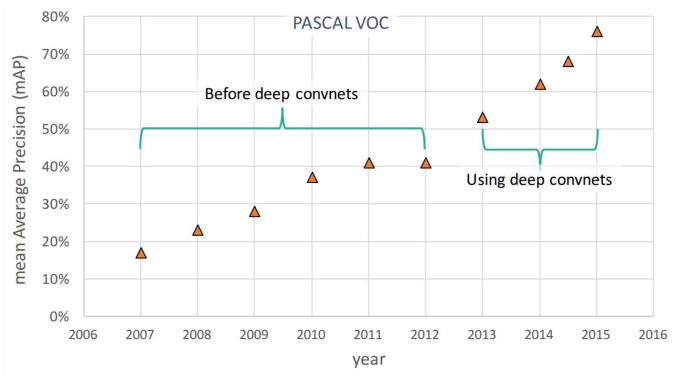


Object detection: sliding windows

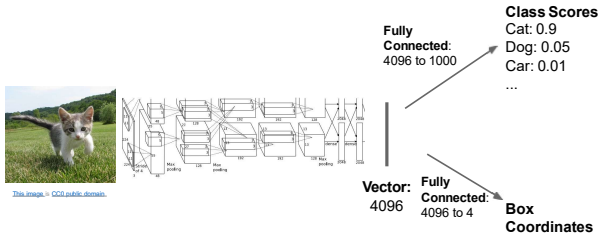
Object Detection



Object Detection: Impact of Deep Learning

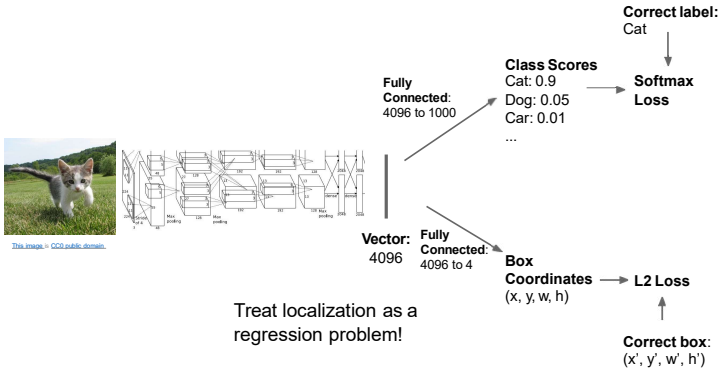


Object Detection: Single Object (Classification + Localization)

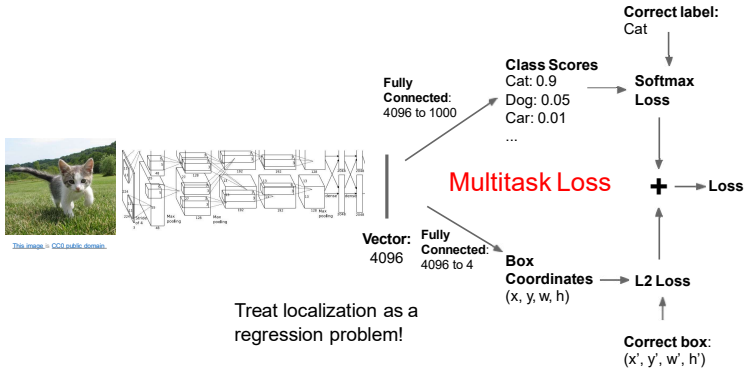


Treat localization as a regression problem!

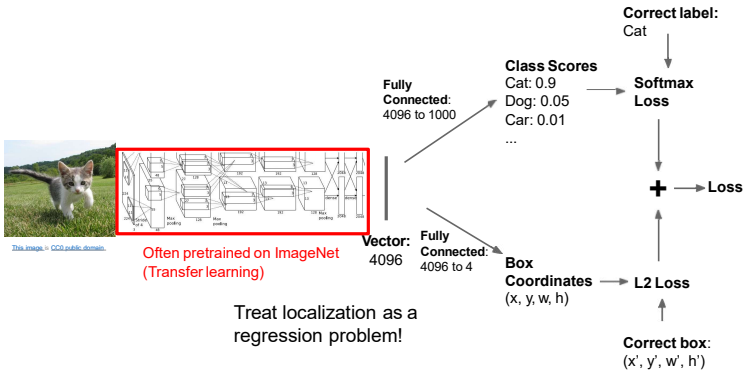
Object Detection: Single Object (Classification + Localization)



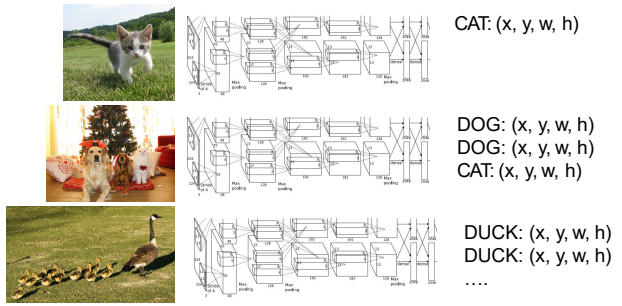
Object Detection: Single Object (Classification + Localization)



Object Detection: Single Object (Classification + Localization)

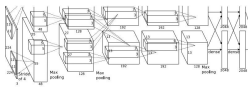


Object Detection: Multiple Objects

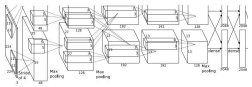


Object Detection: Multiple Objects

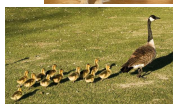
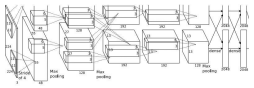
Each image needs a different number of outputs!

CAT: (x, y, w, h) — 4 numbers

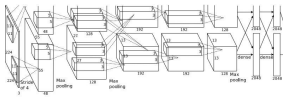
DOG: (x, y, w, h)
DOG: (x, y, w, h)
CAT: (x, y, w, h) — 16 numbers

DUCK: (x, y, w, h)
DUCK: (x, y, w, h)
…. — Many numbers!

Object Detection: Multiple Objects

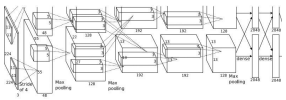
Apply a CNN to many different crops of the image, CNN classifies each crop as object or background

Dog? NO
Cat? NO
Background? YES

Object Detection: Multiple Objects

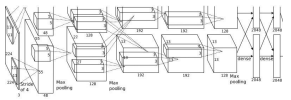
Apply a CNN to many different crops of the image, CNN classifies each crop as object or background

Dog? YES
Cat? NO
Background? NO

Object Detection: Multiple Objects

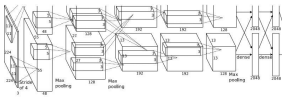
Apply a CNN to many different crops of the image, CNN classifies each crop as object or background

Dog? YES
Cat? NO
Background? NO

Object Detection: Multiple Objects

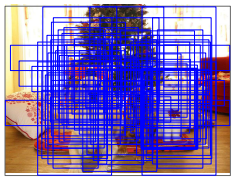
Apply a CNN to many different crops of the image, CNN classifies each crop as object or background

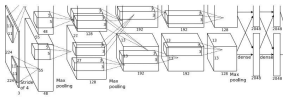


Dog? NO
Cat? YES
Background? NO

Object Detection: Multiple Objects

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





Dog? NO
Cat? YES
Background? NO

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

Object Detection

Two Stages

- Propose “objects”
- Classify each candidate

One-Stage

- Sliding window to classify all candidates

Object Detection

Two Stages

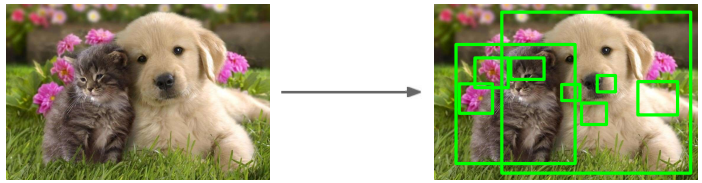
- Propose “objects”
- Classify each candidate

One-Stage

- Sliding window to classify all candidates

Region Proposals: Selective Search

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



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

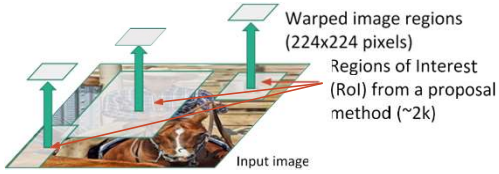
R-CNN

Girshick et al, “Rich feature hierarchies for accurate object detection and semantic segmentation”, CVPR 2014.

R-CNN

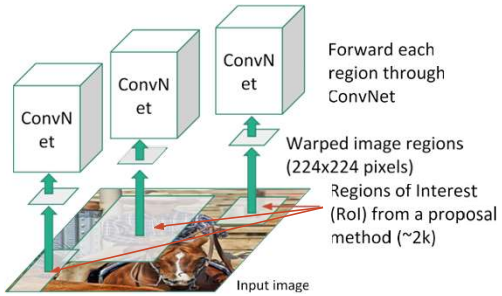
Girshick et al, “Rich feature hierarchies for accurate object detection and semantic segmentation”, CVPR 2014.

R-CNN



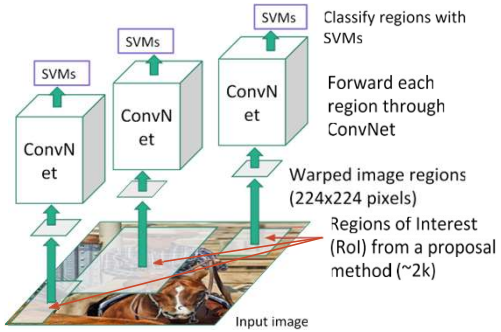
Girshick et al, “Rich feature hierarchies for accurate object detection and semantic segmentation”, CVPR 2014.

R-CNN



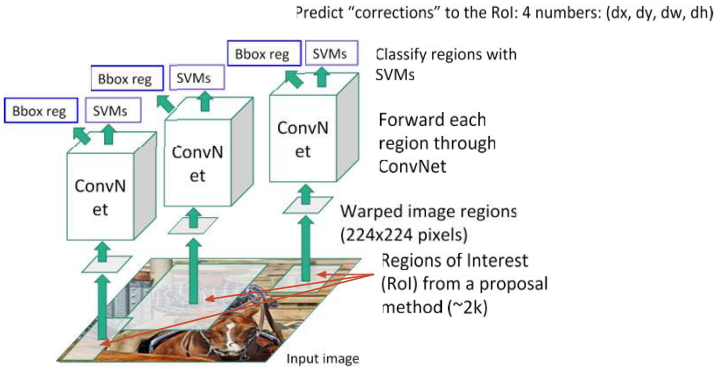
Girshick et al, "Rich feature hierarchies for accurate object detection and semantic segmentation", CVPR 2014.

R-CNN



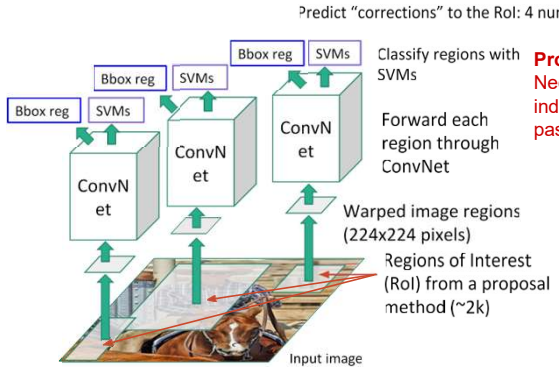
Girshick et al, "Rich feature hierarchies for accurate object detection and semantic segmentation", CVPR 2014.

R-CNN



Girshick et al, "Rich feature hierarchies for accurate object detection and semantic segmentation", CVPR 2014.

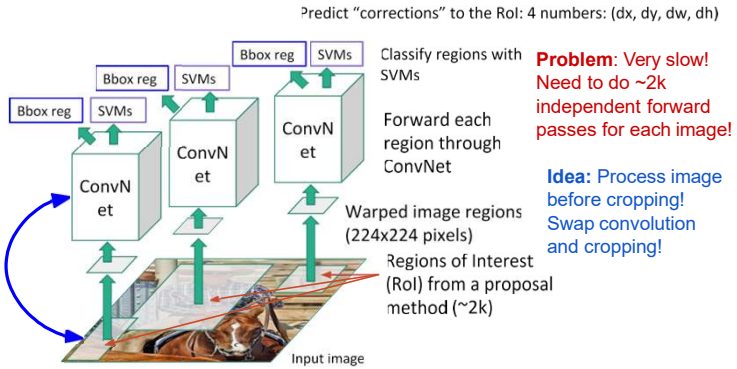
R-CNN



Problem: Very slow! Need to do ~2k independent forward passes for each image!

Girshick et al, "Rich feature hierarchies for accurate object detection and semantic segmentation", CVPR 2014.

"Slow" R-CNN



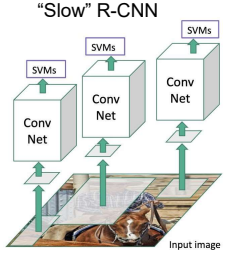
Problem: Very slow! Need to do ~2k independent forward passes for each image!

Idea: Process image before cropping! Swap convolution and cropping!

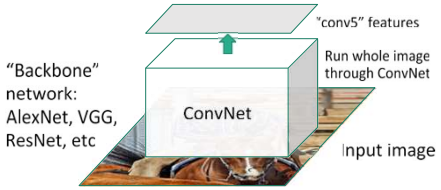
Girshick et al, "Rich feature hierarchies for accurate object detection and semantic segmentation", CVPR 2014.

Fast R-CNN

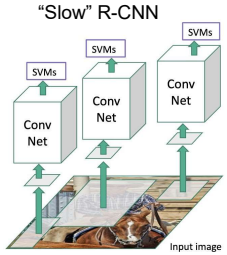
"Slow" R-CNN



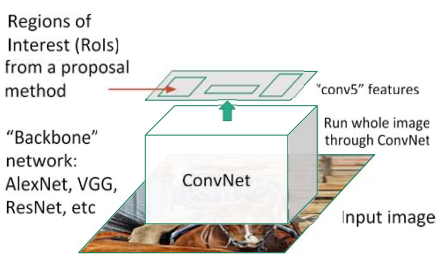
Fast R-CNN



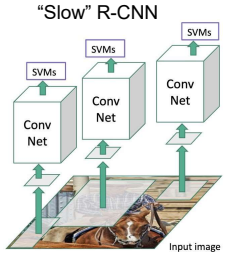
"Slow" R-CNN



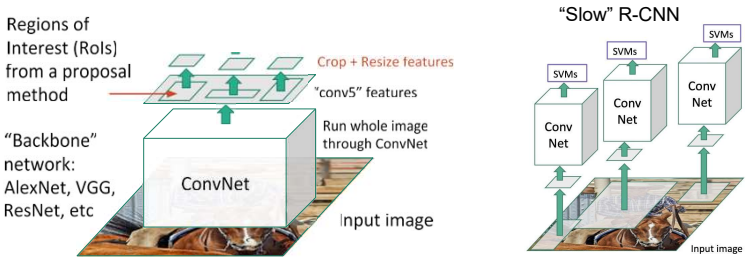
Fast R-CNN



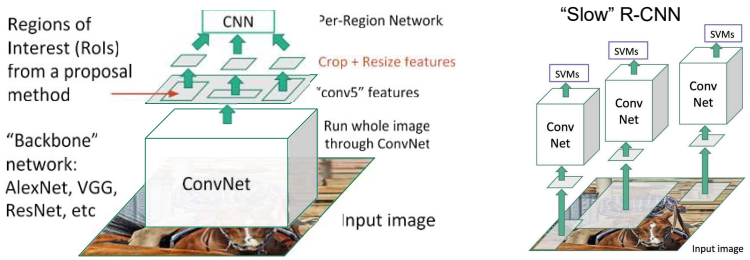
"Slow" R-CNN



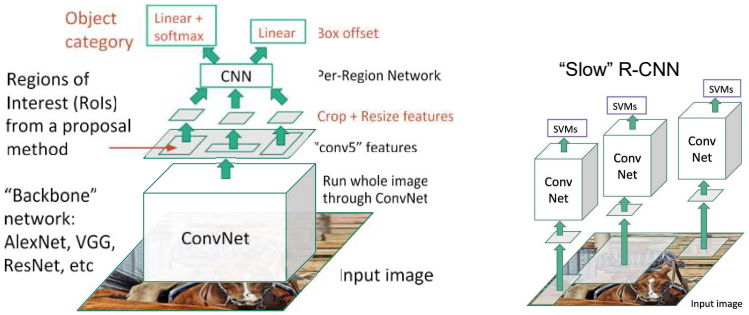
Fast R-CNN



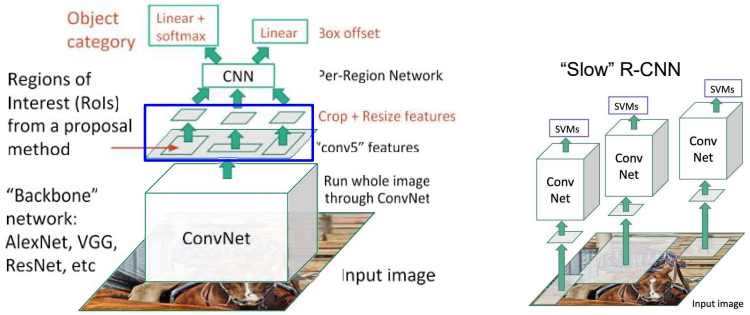
Fast R-CNN



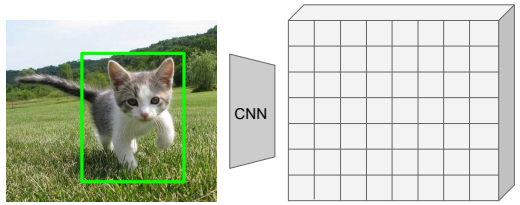
Fast R-CNN



Fast R-CNN

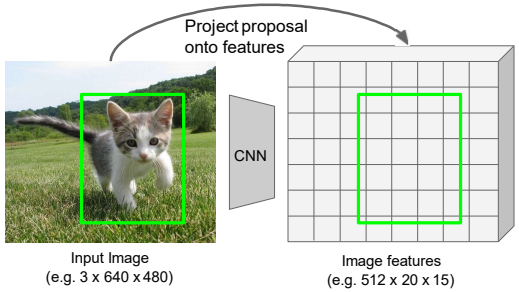


Cropping Features: RoI Pool



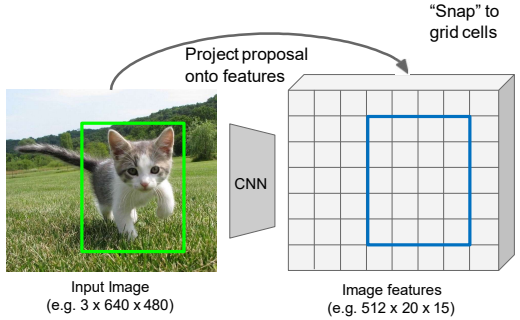
Girshick, "Fast R-CNN", ICCV2015.

Cropping Features: RoI Pool



Girshick, "Fast R-CNN", ICCV2015.

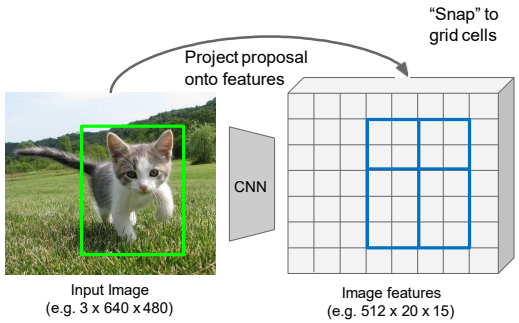
Cropping Features: RoI Pool



Girshick, "Fast R-CNN", ICCV2015.

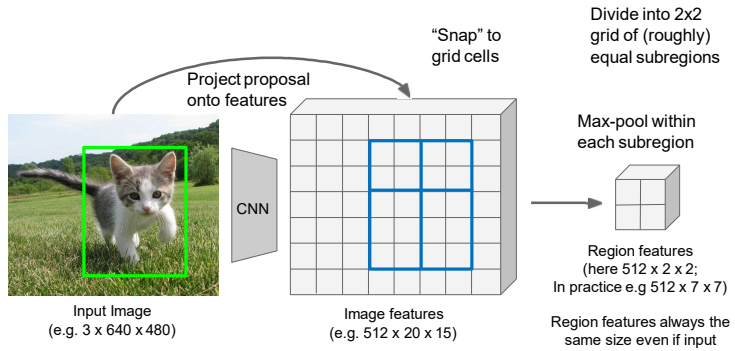
Cropping Features: RoI Pool

Divide into 2x2
grid of (roughly)
equal subregions



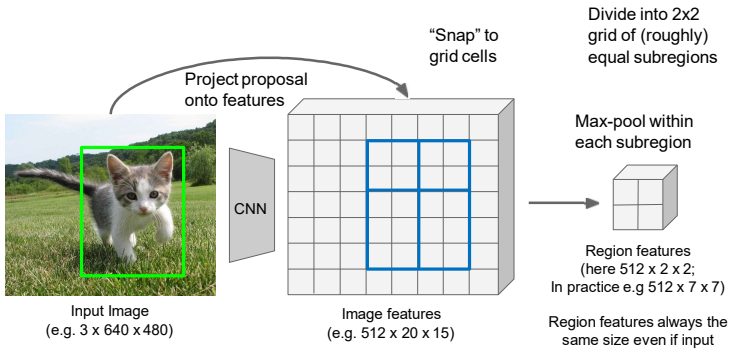
Girshick, "Fast R-CNN", ICCV2015.

Cropping Features: RoI Pool



Girshick, "Fast R-CNN", ICCV2015.

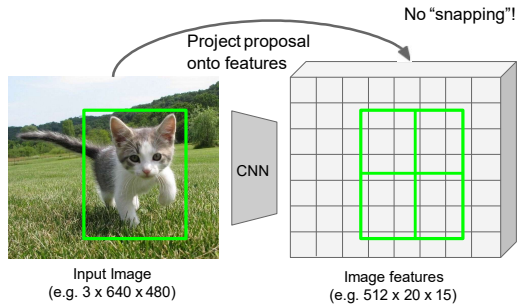
Cropping Features: RoI Pool



Girshick, "Fast R-CNN", ICCV2015.

Problem: Region features slightly misaligned

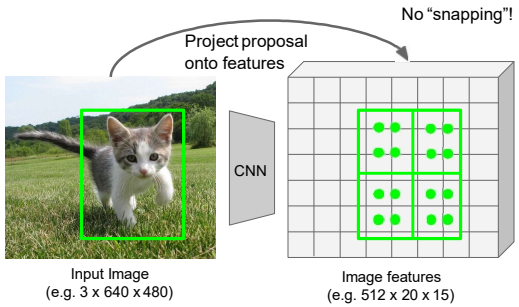
Cropping Features: RoI Align



Girshick, "Fast R-CNN", ICCV2015.

Cropping Features: RoI Align

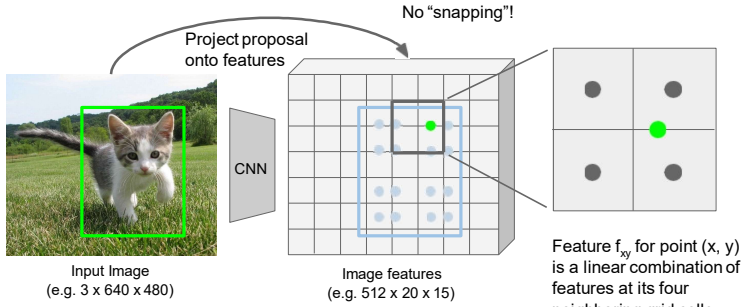
Sample at regular points in each subregion using bilinear interpolation



Girshick, "Fast R-CNN", ICCV2015.

Cropping Features: RoI Align

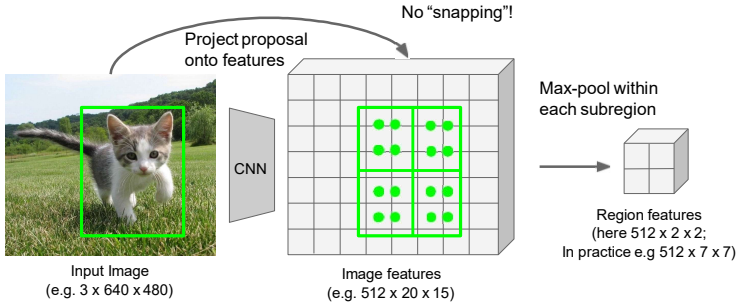
Sample at regular points in each subregion using bilinear interpolation



Girshick, "Fast R-CNN", ICCV 2015.

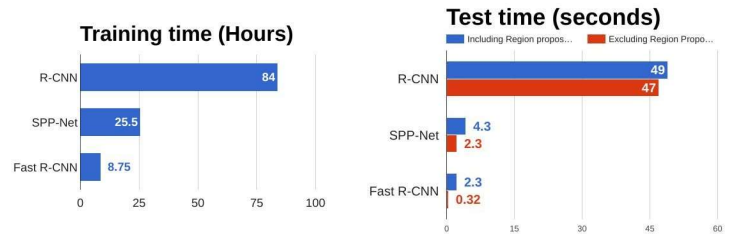
Cropping Features: RoI Align

Sample at regular points in each subregion using bilinear interpolation



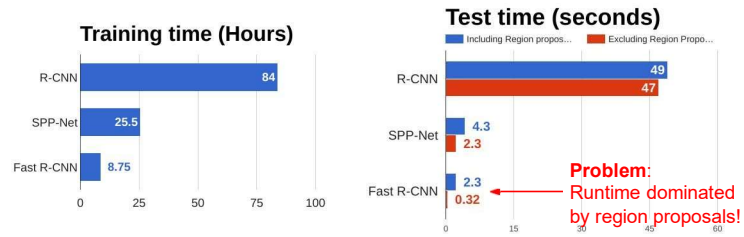
Girshick, "Fast R-CNN", ICCV 2015.

R-CNN vs Fast R-CNN



Girshick et al, "Rich feature hierarchies for accurate object detection and semantic segmentation", CVPR 2014.
He et al, "Spatial pyramid pooling in deep convolutional networks for visual recognition", ECCV 2014
Girshick, "Fast R-CNN", ICCV 2015

R-CNN vs Fast R-CNN

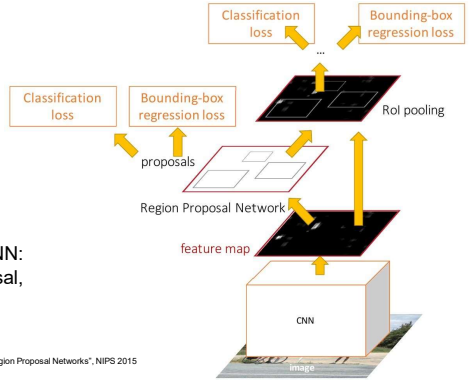


Girshick et al, "Rich feature hierarchies for accurate object detection and semantic segmentation", CVPR 2014.
He et al, "Spatial pyramid pooling in deep convolutional networks for visual recognition", ECCV 2014
Girshick, "Fast R-CNN", ICCV 2015

Faster R-CNN: Make CNN do proposals!

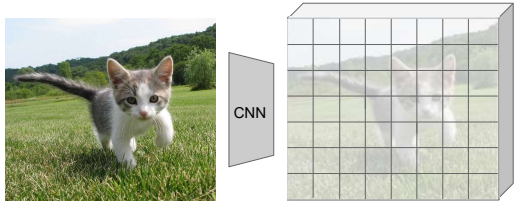
Insert **Region Proposal Network (RPN)** to predict proposals from features

Otherwise same as Fast R-CNN: Crop features for each proposal, classify each one



Ren et al, "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks", NIPS 2015
Figure copyright 2015, Ross Girshick; reproduced with permission

Region Proposal Network

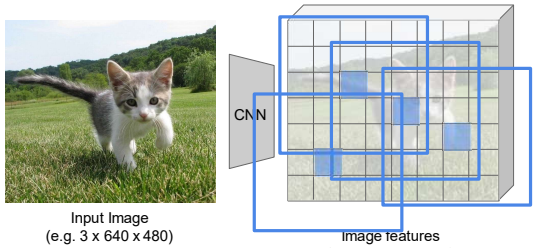


Input Image
(e.g. 3 x 640 x 480)

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

Region Proposal Network

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

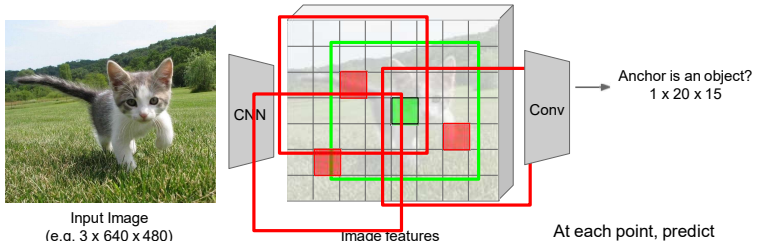


Input Image
(e.g. 3 x 640 x 480)

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

Region Proposal Network

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

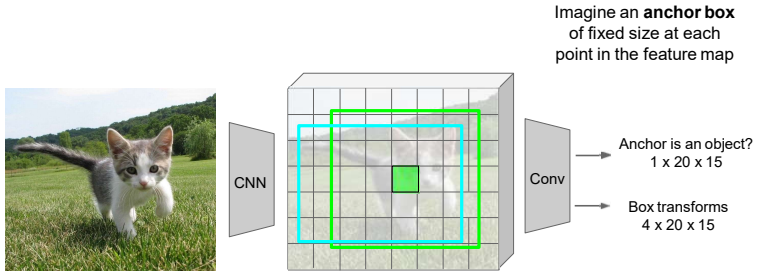


Input Image
(e.g. 3 x 640 x 480)

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

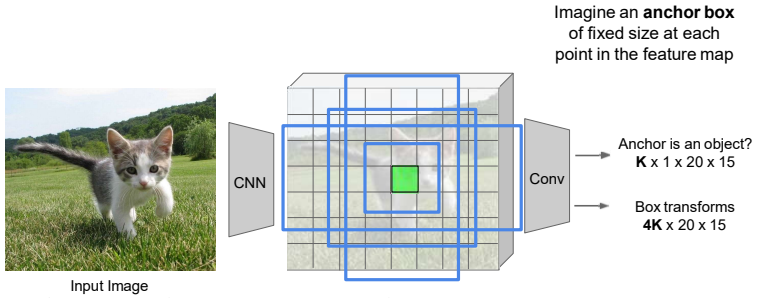
At each point, predict whether the corresponding anchor contains an object (per-pixel logistic regression)

Region Proposal Network

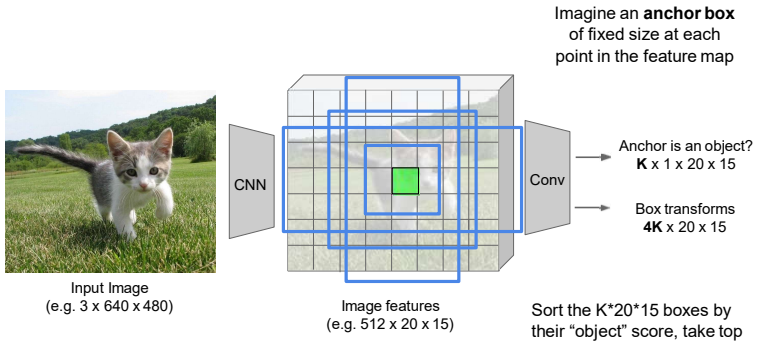


For positive boxes, also predict a transformation from the anchor to the ground-truth box (regress 4 numbers per pixel)

Region Proposal Network



Region Proposal Network

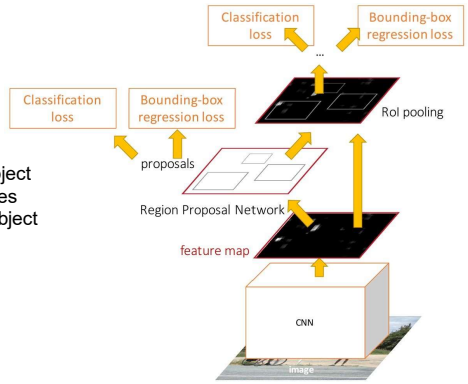


Sort the K*20*15 boxes by their "object" score, take top ~300 as our proposals

Faster R-CNN: Make CNN do proposals!

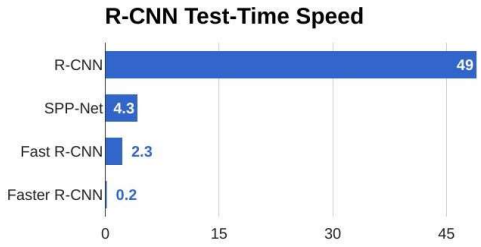
Jointly train with 4 losses:

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



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

Faster R-CNN: Make CNN do proposals!

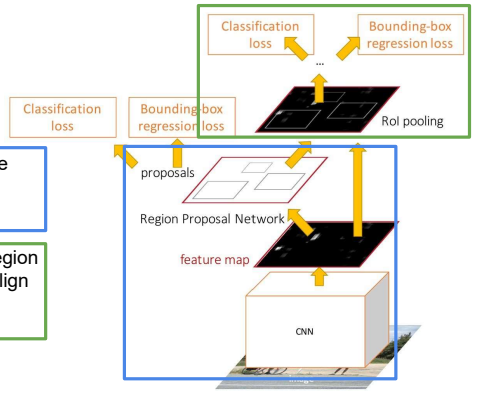


Faster R-CNN: Make CNN do proposals!

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

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

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



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

One-stage Object detection

Anchor-based

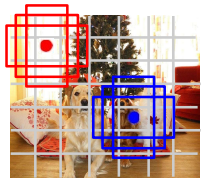
Object Detection

Two Stages
- Propose "objects"
- Classify each candidate

One-Stage
- Sliding window to classify all candidates

Single-Stage Object Detectors: YOLO / SSD / RetinaNet

Input image
3 x H x W



Divide image into grid
7 x 7

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

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

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

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

Imbalance

Number of "negative" anchors ~O(10K)
Number of "positive" anchors ~O(10)

What happens to CE loss in this case?

$$\mathcal{L} = -\frac{1}{N}\sum_i y_i \log(p_i), \qquad \frac{\partial \mathcal{L}}{\partial p_i} = \begin{cases} p_{i,l} & \text{if } y_i \neq l \\ p_{i,l} - 1 & \text{if } y_i = l \end{cases}$$

Loss and gradient are dominated by correctly classified negative examples

Training outcome → constant "negative" prediction.

Imbalance – Focal Loss

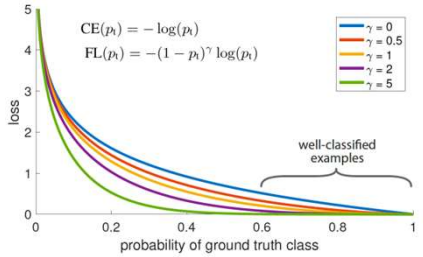
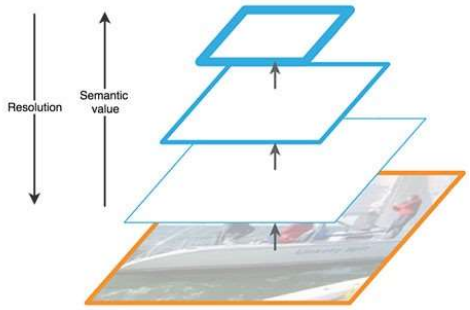


Figure 1: We propose a novel loss we term the *Focal Loss* that adds a factor $(1 - p_t)^\gamma$ to the standard cross entropy criterion. Setting $\gamma > 0$ reduces the relative loss for well-classified examples ($p_t > .5$), putting more focus on hard, misclassified examples. As our experiments will demonstrate, the proposed focal loss enables training highly accurate dense object detectors in the presence of vast numbers of easy background examples.

Lin,Goyal, Girshick, He, and Dollár
Focal loss for dense object detection (PAMI 2018)

Feature Pyramid Network (FPN)

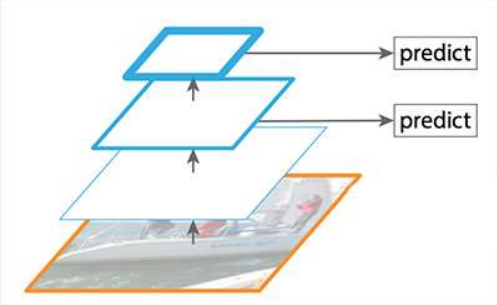
- How to handle multiscale predictions?



Tsung-Yi, Dollár, Girshick, He, Hariharan and Belongie. Feature Pyramid Networks for Object Detection (CVPR 2017)

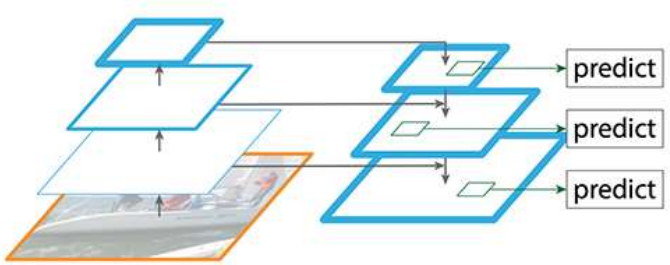
Feature Pyramid Network (FPN)

- How to handle multiscale predictions?

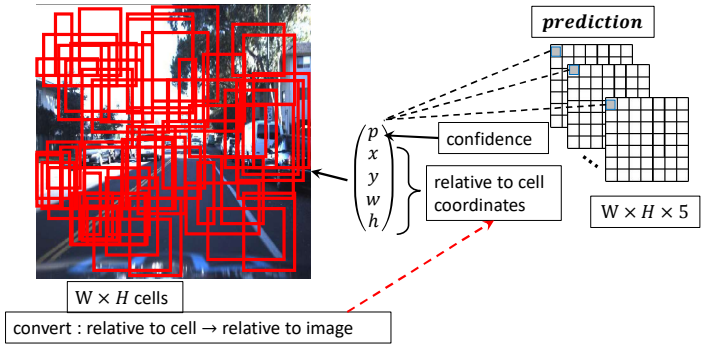


Feature Pyramid Network (FPN)

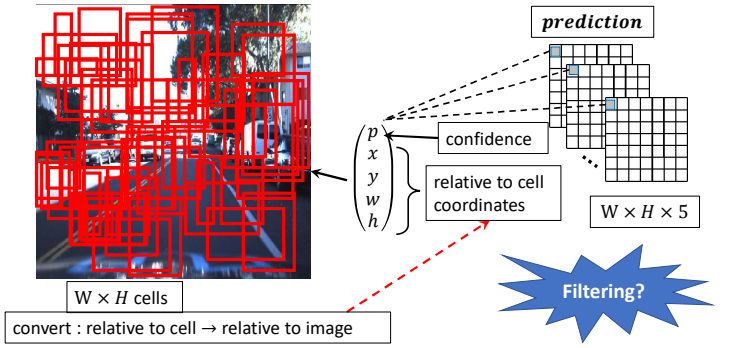
- How to handle multiscale predictions?



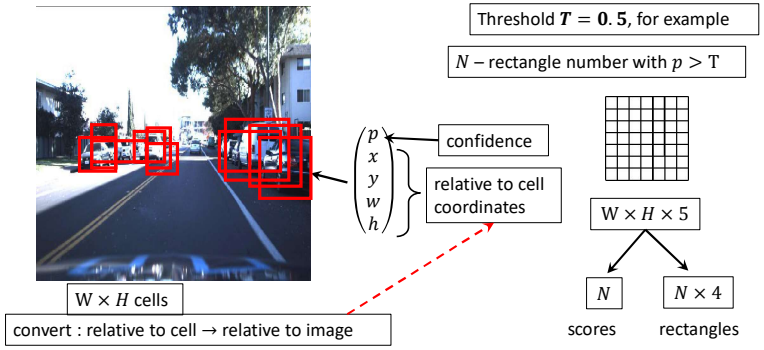
Postprocessing: NMS



Postprocessing: NMS

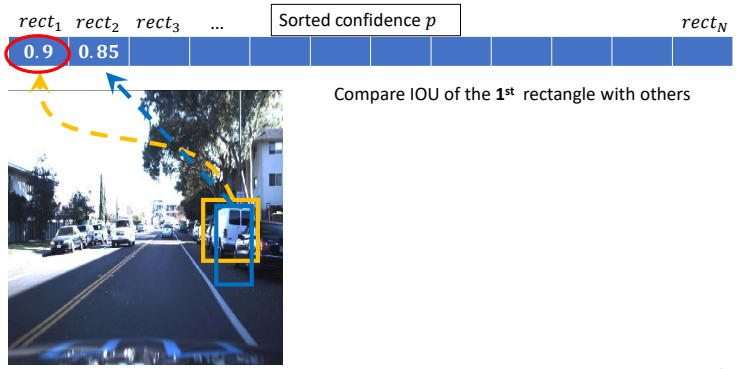


Postprocessing: NMS

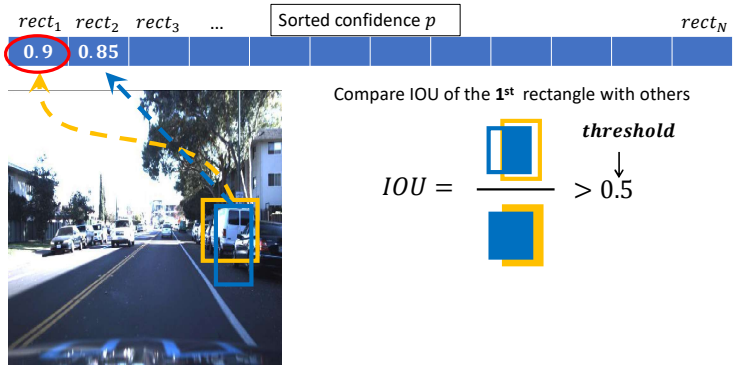


Postprocessing: NMS

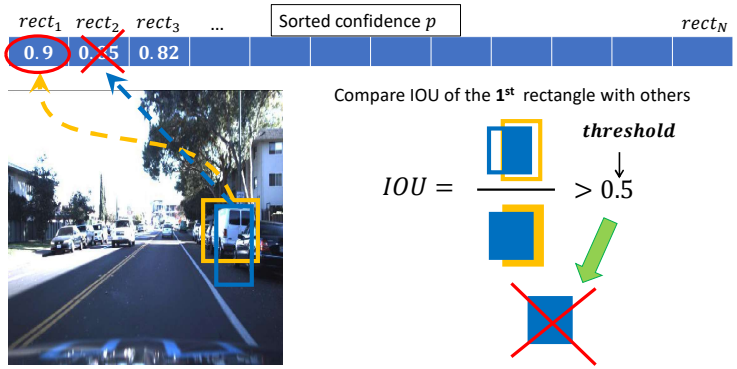
Postprocessing: NMS



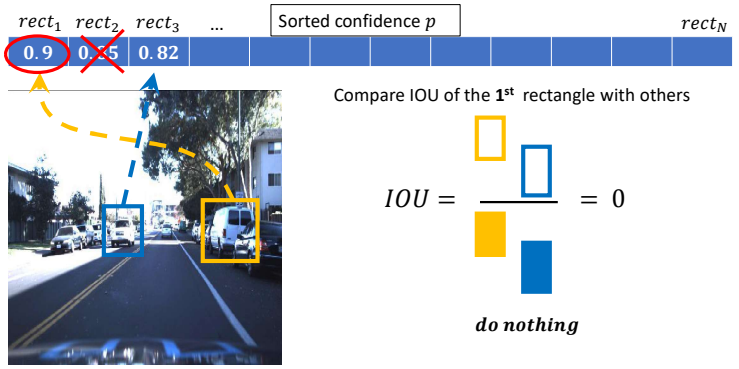
Postprocessing: NMS



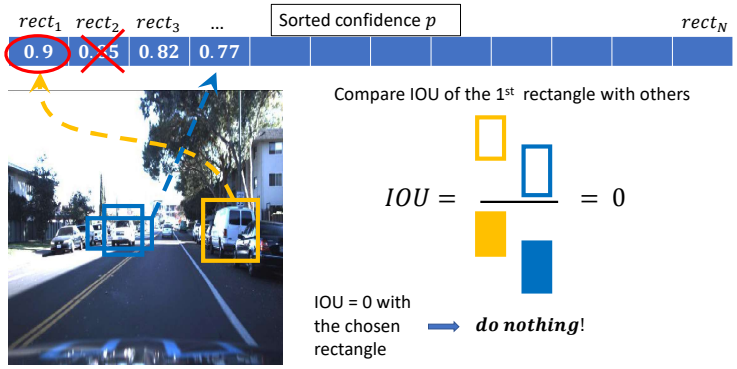
Postprocessing: NMS



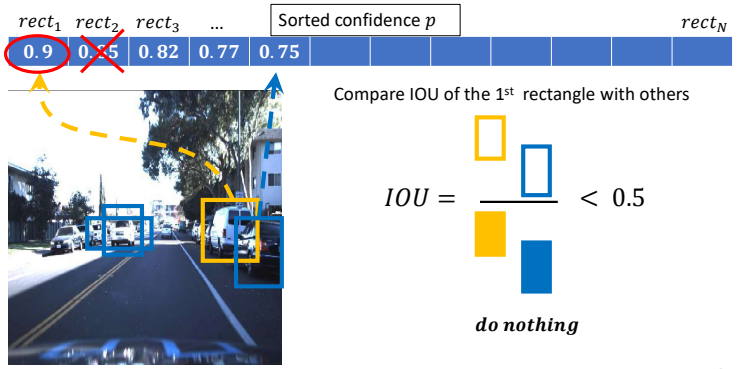
Postprocessing: NMS



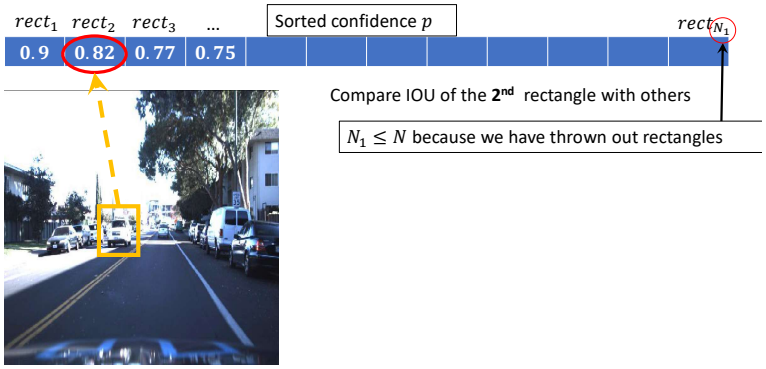
Postprocessing: NMS



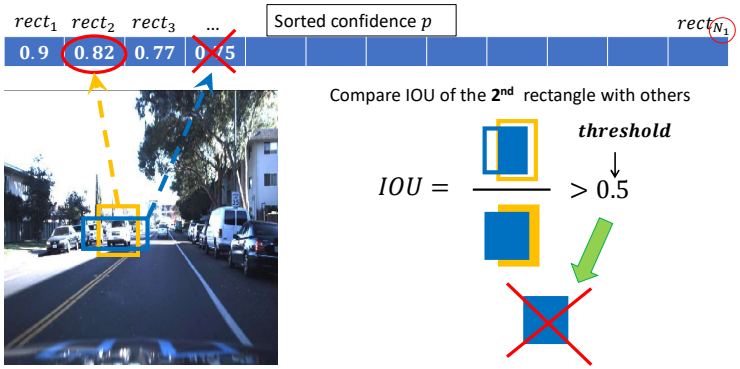
Postprocessing: NMS



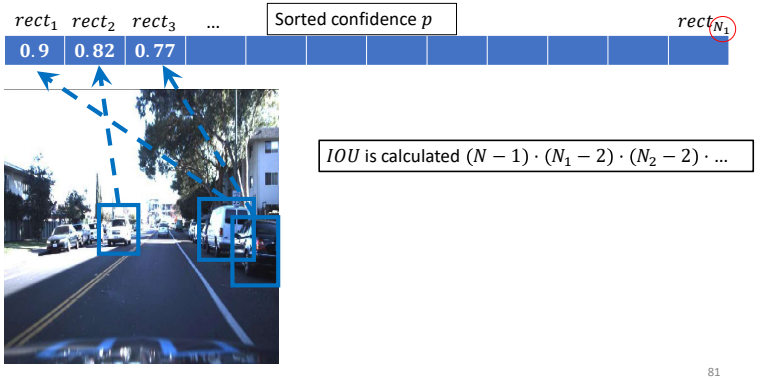
Postprocessing: NMS



Postprocessing: NMS



Postprocessing: NMS



One-stage Object detection

Anchor-free

Drawbacks of Anchor Boxes

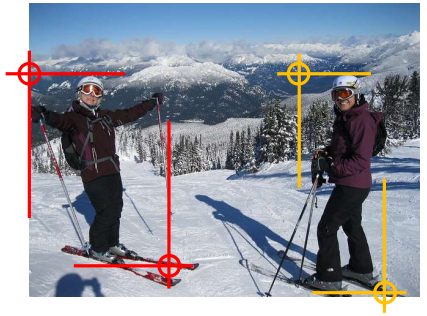
1. Need a large number of anchors

→ A tiny fraction of anchors are positive examples

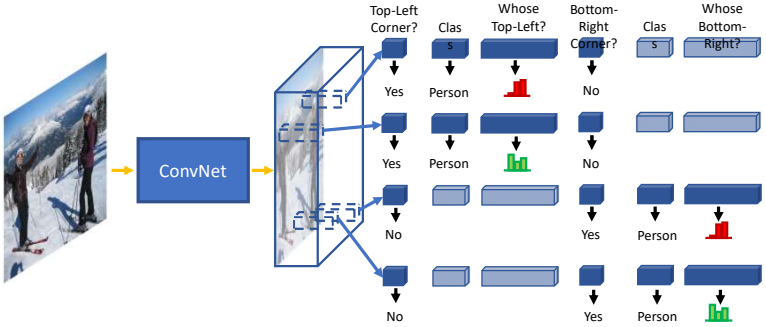
→ Slow down training [Lin et al. ICCV'17]

2. Extra hyperparameters – sizes and aspect ratios

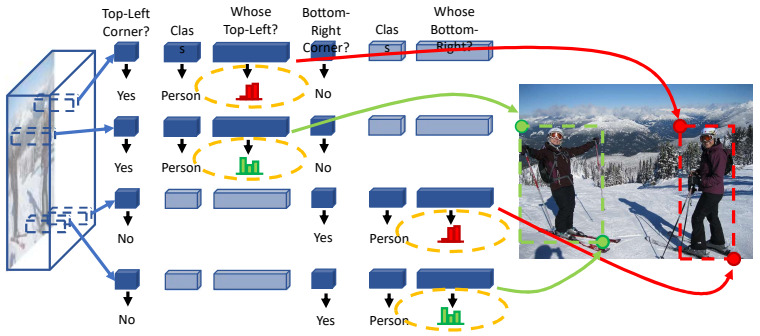
CornerNet: Detecting Objects as Paired Keypoints



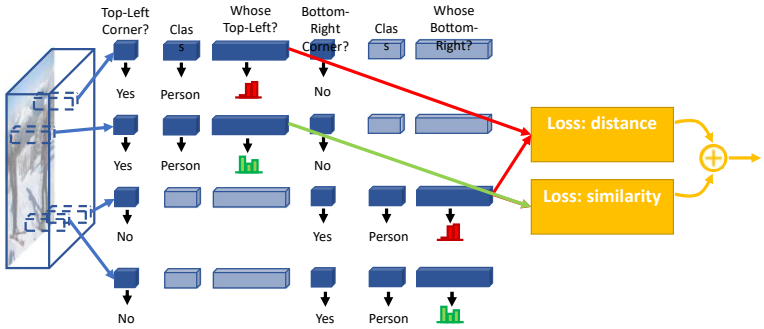
CornerNet: Detecting Objects as Paired Keypoints



CornerNet: Detecting Objects as Paired Keypoints

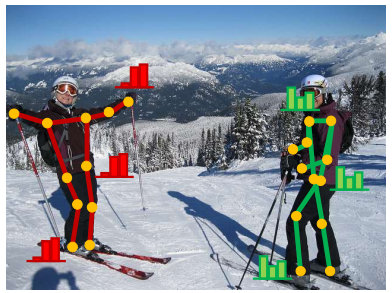


CornerNet: Detecting Objects as Paired Keypoints



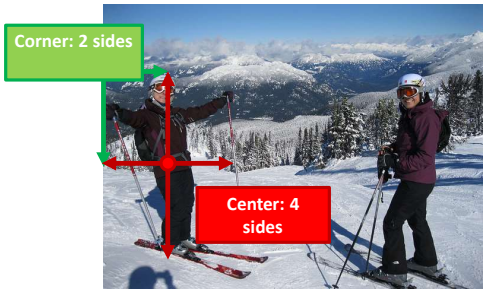
Associative Embedding [Newell et al. NIPS'17]

- CornerNet is inspired by the idea of associative embedding from work by Newell et al. [NIPS'17]



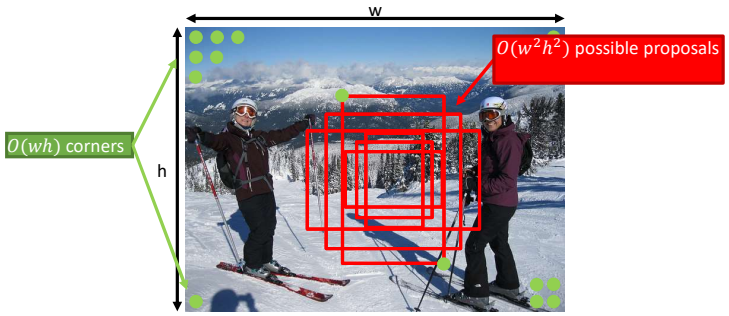
https://proceedings.neurips.cc/paper/2017/file/8edd72158ccd2a879f79cb2538568fdc

Advantages of Detecting Corners



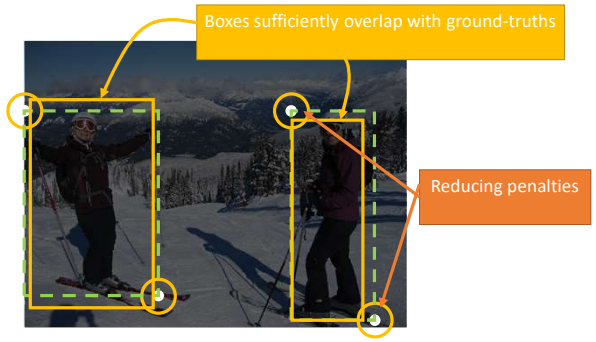
Detecting corner is easier than detecting center

Advantages of Detecting Corner

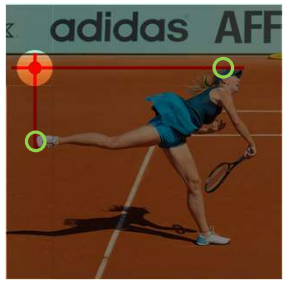


Represent $O(w^2h^2)$ possible proposals using only $O(wh)$ corners

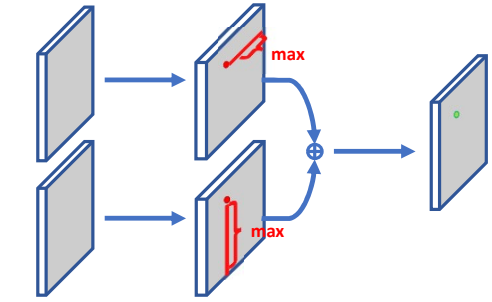
Supervising Corner Detection



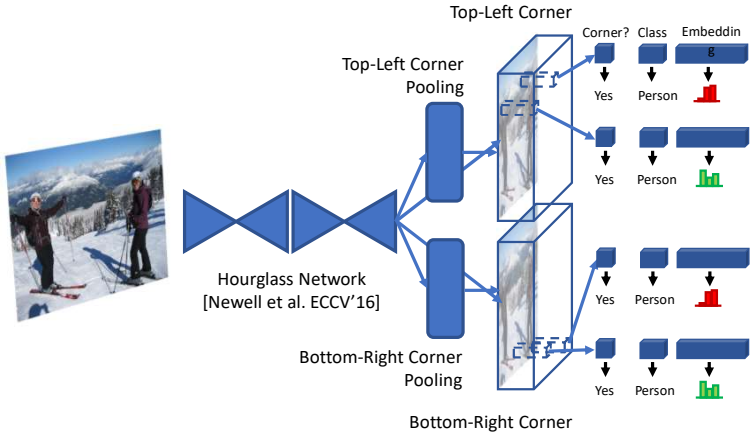
Corner Pooling



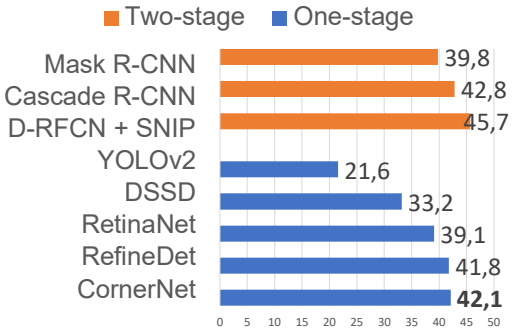
Top-Left Corner Pooling



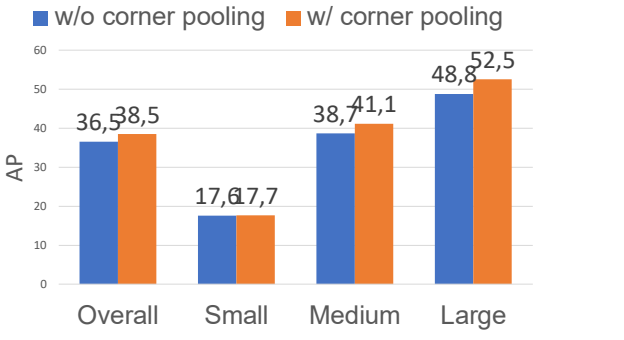
CornerNet

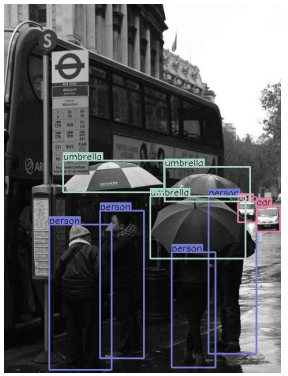


Experiment: CornerNet versus Others

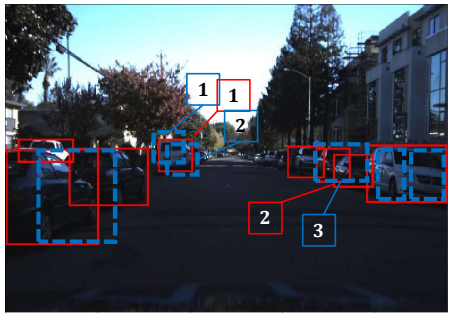


Experiment: Corner Pooling





Accuracy evaluation



n predicted

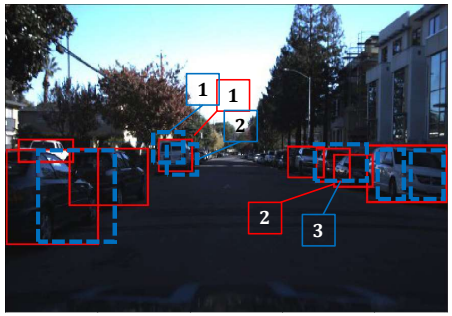
m Ground Truth

$$IoU = \frac{A \cap B}{A \cup B}$$

A – ground truth
B – detector result

Correspondence between GT and predicted?

Accuracy evaluation



evaluator.py ->
get_single_image_results(gt_boxes, pred_boxes, iou_thr)

Sorted array of $IOU > T$ between GT and predicted (max length = $n \cdot m$)

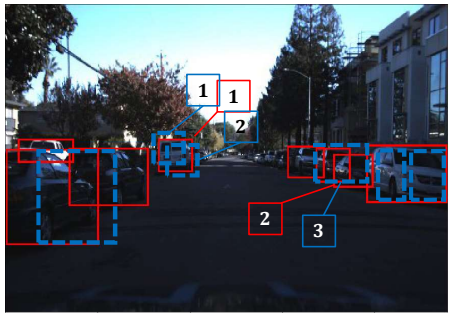
$(iou_1, iou_2, iou_3, \ldots)$

GT	2	1	1
pred	3	1	2

if appear firstly

matched GT	2	1
matched pred	3	1

Accuracy evaluation



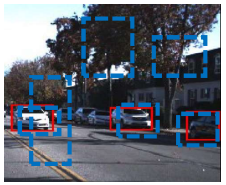
evaluator.py ->
get_single_image_results(gt_boxes, pred_boxes, iou_thr)

Sorted array of $IOU > T$ between GT and predicted (max length = $n \cdot m$)

True predicted

matched GT	2	1
matched pred	3	1

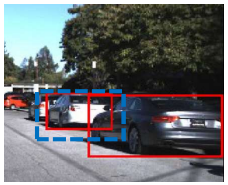
Accuracy evaluation: precision, recall



$T = 0.5$

• • •

getTruePredicted



- predicted = 4
- ground truth = 3
- **true predicted = 1**

$$precision = \frac{true\ predicted}{predicted}$$

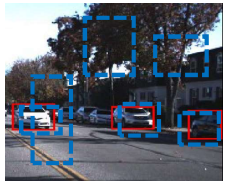
What part of predicted is true

- predicted = 1
- ground truth = 2
- **true predicted = 1**

$$recall = \frac{true\ predicted}{ground\ truth}$$

What part of true predicted from all GT objects

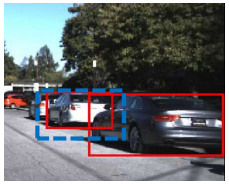
Accuracy evaluation: precision, recall



$T = 0.5$

• • •

getTruePredicted



- predicted = 4
- ground truth = 3
- **true predicted = 1**

$$precision = \frac{true\ predicted}{predicted}$$

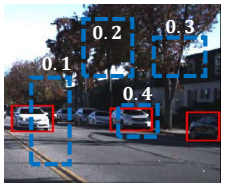
- predicted = 1
- ground truth = 2
- **true predicted = 1**

$$recall = \frac{true\ predicted}{ground\ truth}$$

evaluator.py -> calc_precision_recall(img_results)

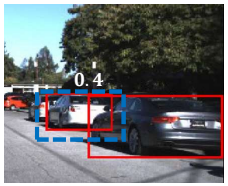
Accuracy evaluation: precision, recall

sorted box confidence ↓ ↑ $(p_1, p_2, \dots p_i, \dots, p_{N-1}, p_N)$



• • •

Each predicted box has confidence $\boldsymbol{p}$



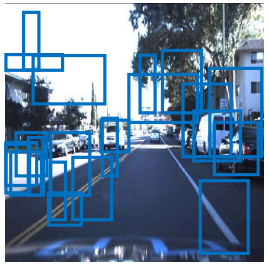
Accuracy evaluation: precision, recall

sorted box confidence ↓ ↑ $(p_1, p_2, \dots p_i, \dots, p_{N-1}, p_N)$

I. Get boxes with $p \geq p_1$, i.e. all boxes

calcPrecisionRecall

score	prec	recall
p_1	0.1	0.9
p_2	0.2	0.8
...	...	...



For all images together!

Accuracy evaluation: precision, recall

sorted box confidence ↓ ↑ $(p_1, p_2, \dots p_i, \dots, p_{N-1}, p_N)$

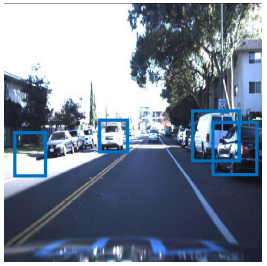
I. Get boxes with $p \geq p_1$, i.e. all boxes

calcPrecisionRecall

score	prec	recall
p_1	0.1	0.9
p_2	0.2	0.8
...	...	...
p_N	0.9	0.1

High precision

low recall



For all images together!

Accuracy evaluation: precision, recall

sorted box confidence

$\downarrow \quad \uparrow$

$(p_1, p_2, \dots p_i, \dots, p_{N-1}, p_N)$

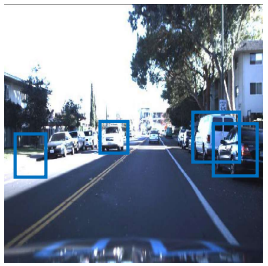
I. Get boxes with $p \geq p_1$, i.e. all boxes

calcPrecisionRecall

score	prec	recall
p_1	**0.1**	**0.9**
p_2	**0.2**	**0.8**
...	...	...
p_N	**0.9**	**0.1**

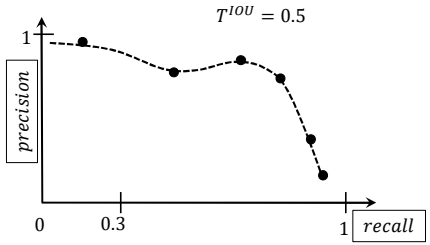
High precision

low recall



get_thr_prec_rec(...)

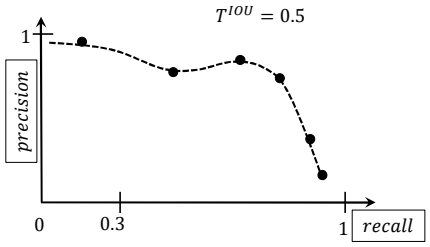
Accuracy evaluation: mAP



hundreds of values for real data sets

score	prec	recall
p_1	**0.1**	**0.9**
p_2	**0.2**	**0.8**
...	...	...
p_N	**0.9**	**0.1**

Accuracy evaluation: mAP



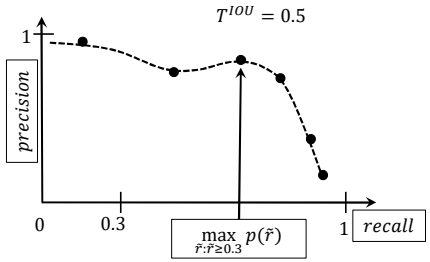
hundreds of values for real data sets

score	prec	recall
p_1	**0.1**	**0.9**
p_2	**0.2**	**0.8**
...	...	...
p_N	**0.9**	**0.1**

11 recall thresholds

$$AP^T = \sum_{r \in \{0, 0.1, \dots, 1\}} \max_{\tilde{r}: \tilde{r} \geq r} p(\tilde{r})$$

Accuracy evaluation: mAP



hundreds of values for real data sets

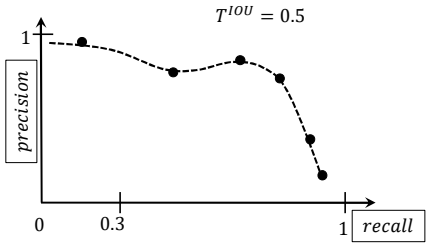
score	prec	recall
p_1	**0.1**	**0.9**
p_2	**0.2**	**0.8**
...	...	...
p_N	**0.9**	**0.1**

A lot of mAP tutorials has misunderstanding : "area under curve" (AUC) is not the same!

11 recall thresholds

$$AP^T = \sum_{r \in \{0, 0.1, \dots, 1\}} \max_{\tilde{r}: \tilde{r} \geq r} p(\tilde{r})$$

Accuracy evaluation: mAP



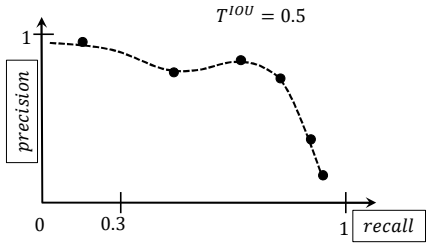
hundreds of values for real data sets

score	prec	recall
p_1	0.1	0.9
p_2	0.2	0.8
...	...	...
p_N	0.9	0.1

$$AP^T = \sum_{r \in \{0, 0.1, \ldots, 1\}} \max_{\tilde{r}: \tilde{r} \geq r} p(\tilde{r})$$

Pascal VOC metric is $AP^{0.5}$

Accuracy evaluation: mAP



hundreds of values for real data sets

score	prec	recall
p_1	0.1	0.9
p_2	0.2	0.8
...	...	...
p_N	0.9	0.1

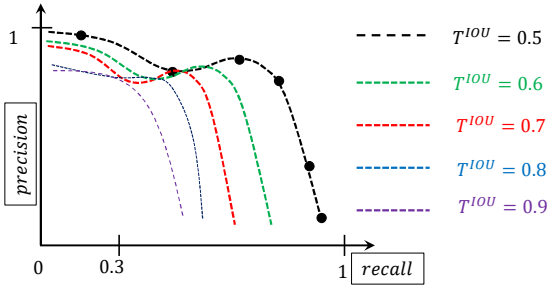
$$AP = \frac{1}{10} \cdot \sum_{T \in \{0.5, 0.55, \ldots, 0.95\}} AP^T$$

MS COCO has metrics $AP^{0.5}, AP^{0.75}, AP$

Accuracy evaluation: mAP

$$AP = \frac{1}{10} \cdot \sum_{T \in \{0.5, 0.55, \ldots, 0.95\}} AP^T$$

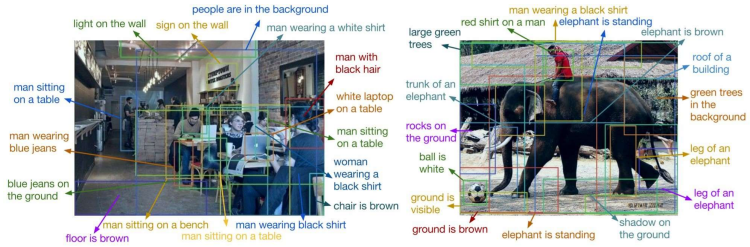
MS COCO uses 10 IOU thresholds 0.5, 0.55, ... 0.95



Computer Vision Tasks

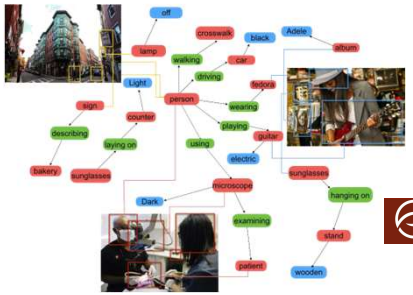
Beyond 2D Object Detection...

Object Detection + Captioning = Dense Captioning



Johnson, Karpathy, and Fei-Fei, "DenseCap: Fully Convolutional Localization Networks for Dense Captioning", CVPR 2016
Figure copyright IEEE, 2016. Reproduced for educational purposes.

Objects + Relationships = Scene Graphs

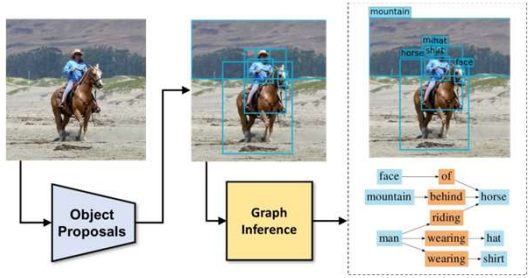


108,077 Images
5.4 Million Region Descriptions
1.7 Million Visual Question Answers
3.8 Million Object Instances
2.8 Million Attributes
2.3 Million Relationships
Everything Mapped to Wordnet Synsets

VISUALGENOME

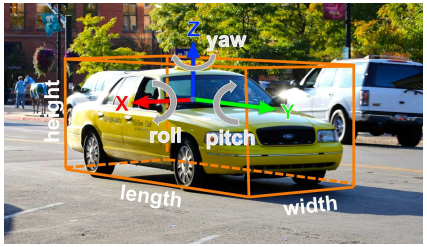
Krishna, Ranjay, Yuke Zhu, Oliver Groth, Justin Johnson, Kenji Hata, Joshua Kravitz, Stephanie Chen et al. "Visual genome: Connecting language and vision using crowdsourced dense image annotations." International Journal of Computer Vision 123, no. 1 (2017): 32-73.

Scene Graph Prediction



Xu, Zhu, Choy, and Fei-Fei, "Scene Graph Generation by Iterative Message Passing", CVPR 2017
Figure copyright IEEE, 2018. Reproduced for educational purposes.

3D Object Detection



2D Object Detection:
2D bounding box
(x, y, w, h)

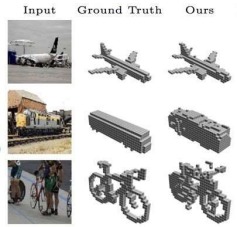
3D Object Detection:
3D oriented bounding box
(x, y, z, w, h, l, r, p, y)

Simplified bbox: no roll & pitch

Much harder problem than 2D object detection!

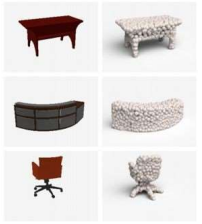
3D Shape Prediction

Voxel:
D x D x D binary



Choy et al, "3D-R2N2: A Unified Approach for Single and Multi-view 3D Object Reconstruction", ECCV 2016

Pointcloud:
V x 3 float



Fan et al, "A Point Set Generation Network for 3D Object Reconstruction from a Single Image", CVPR 2017

Mesh:
V x 3 float, F x 3 int

Wang et al, "Pixel2Mesh: Generating 3D Mesh Models from Single RGB Images", ECCV 2018

Open Source Frameworks

- Lots of good implementations on GitHub!
- TensorFlow Detection API:
- https://github.com/tensorflow/models/tree/master/research/object detection

Faster RCNN, SSD, RFCN, Mask R-CNN

- Caffe2 Detectron:
- https://github.com/facebookresearch/Detectron
- Mask R-CNN, RetinaNet, Faster R-CNN, RPN, Fast R-CNN, R-FCN
- Finetune on your own dataset with pre-trained models

References

1. CS231n: Convolutional Neural Networks for Visual Recognition
   - http://cs231n.stanford.edu/
2. Object Detection Creation from Scratch. Samsung R&D Institute Ukraine. Vitaliy Bulygin https://aiukraine.com/wp-content/uploads/2018/08/Vitalij-Buly-gin-.pptx
3. CornerNet: Detecting Objects as. Paired Keypoints https://pvl.cs.princeton.edu/assets/CornerNet.pptx


25
SOICT
VIỆN CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG
SCHOOL OF INFORMATION AND COMMUNICATION TECHNOLOGY
Thank you!
soict.hust.edu.vn/
fb.com/groups/soict
B1