

CV overview

Vladislav Goncharenko
Spring 2024
MIPT, Moscow

Test

1. Почему при хранении картинки мы используем 3 канала?
2. Какая проблема есть у объектива с одной линзой?
3. Назовите три важнейших цветовых пространства
4. Назовите характерные проблемы Rolling Shutter камер
5. Назовите три библиотеки для работы с изображениями в Пионе (не нейросетевые).

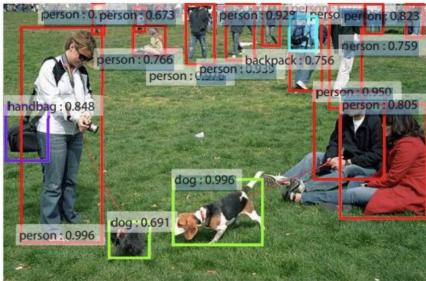
Outline

- Computer Vision tasks
 - Classification
 - Localisation
 - Detection
 - Semantic Segmentation
 - Instance Segmentation
- Metrics
- Datasets
- R-CNN
- Fast R-CNN
- Faster R-CNN

Computer Vision tasks



Computer Vision tasks



- Classification
- Localisation
- Detection
- Semantic Segmentation
- Instance Segmentation
- Style Transfer
- Adversarial Attacks
- Human Pose Estimation
- Person Re-Identification
- Object Tracking
-

Classification



Cat

Input image ($C \times H \times W$)

Output

class label (int)

Localisation



Cat

Input image ($C \times H \times W$)

Output

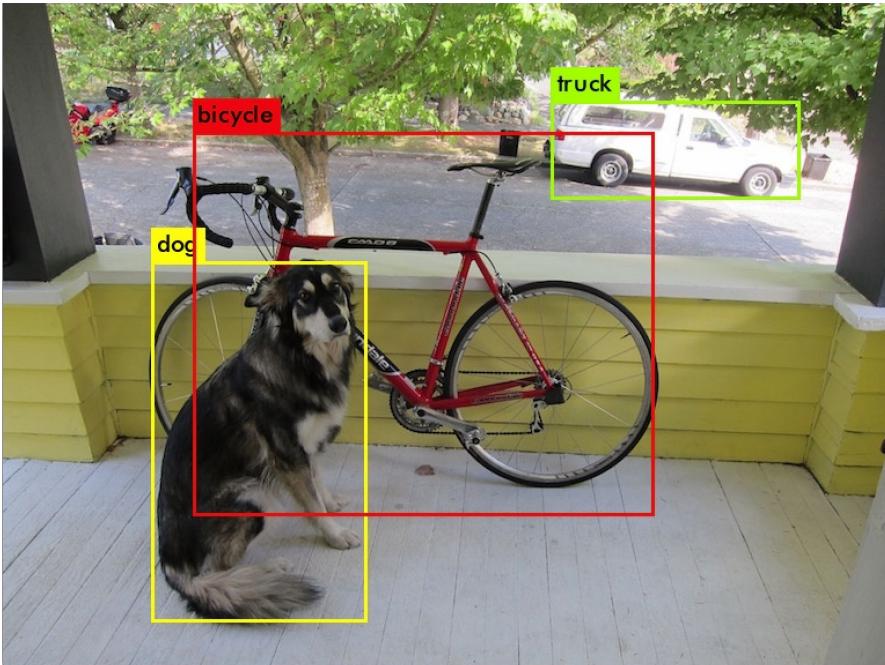
class label (int), BB* coordinates (4 ints)

Assumption

one object per image

* BB - bounding box

Detection



Input image ($C \times H \times W$)

Output

class label (int), confidence (0-1 float), BB (4 ints)

.....

class label (int), confidence (0-1 float), BB (4 ints)

Assumption

multiple objects per image

Semantic Segmentation



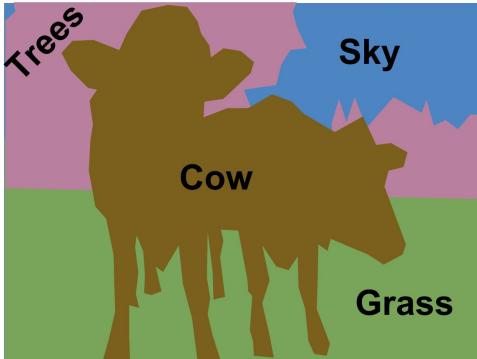
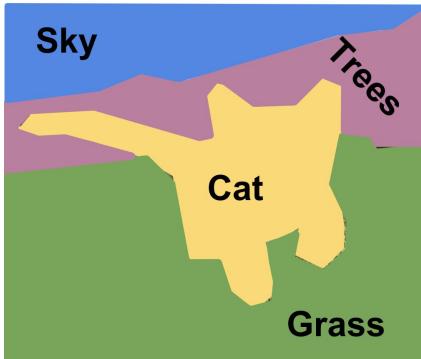
Input image ($C \times H \times W$)

Output

mask of classes ($1 \times H \times W$)

Assumption

just class label for each pixel



Instance Segmentation



Input image (C x H x W)

Output

mask of class instances (detection + 1 x H x W)

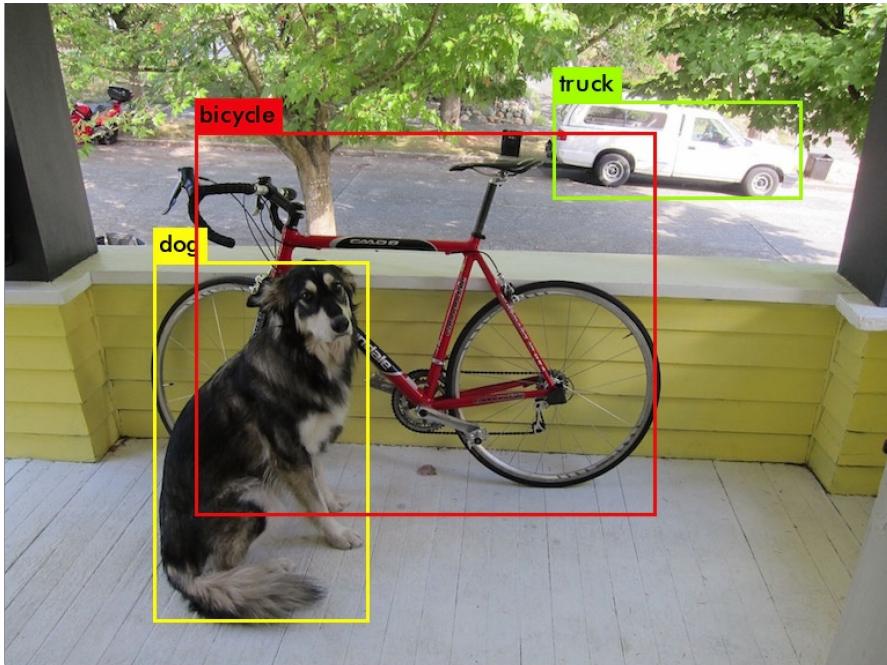
Assumption

separated masks for each distinct object

Detection metrics



Detection



Input image ($C \times H \times W$)

Output

class label (int), confidence (0-1 float), BB (4 ints)

.....

class label (int), confidence (0-1 float), BB (4 ints)

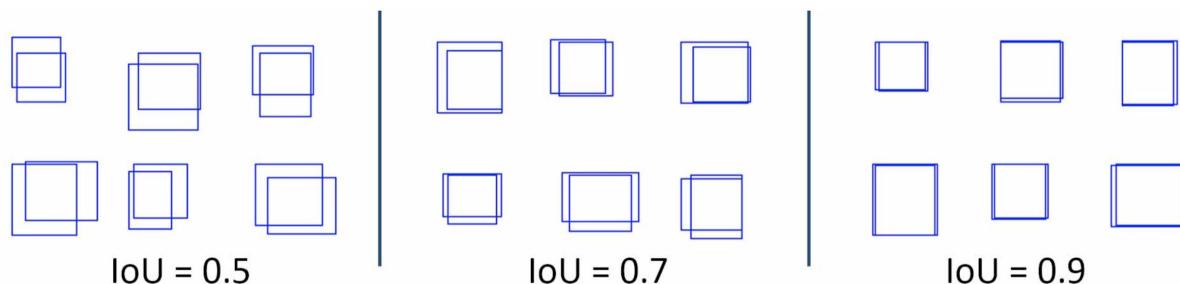
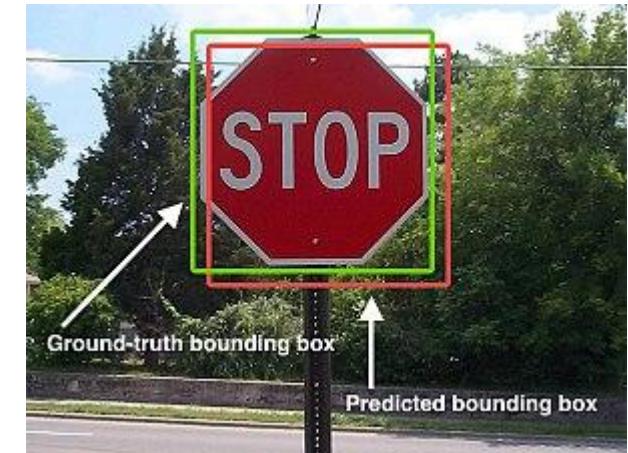
Assumption

multiple objects per image

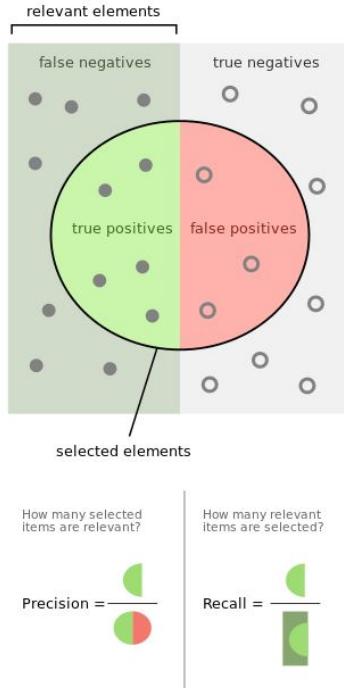
Intersection over Union (IoU)

aka Jaccard index

$$\text{IoU} = \frac{\text{Area of Overlap}}{\text{Area of Union}}$$



mean Average Precision (mAP)

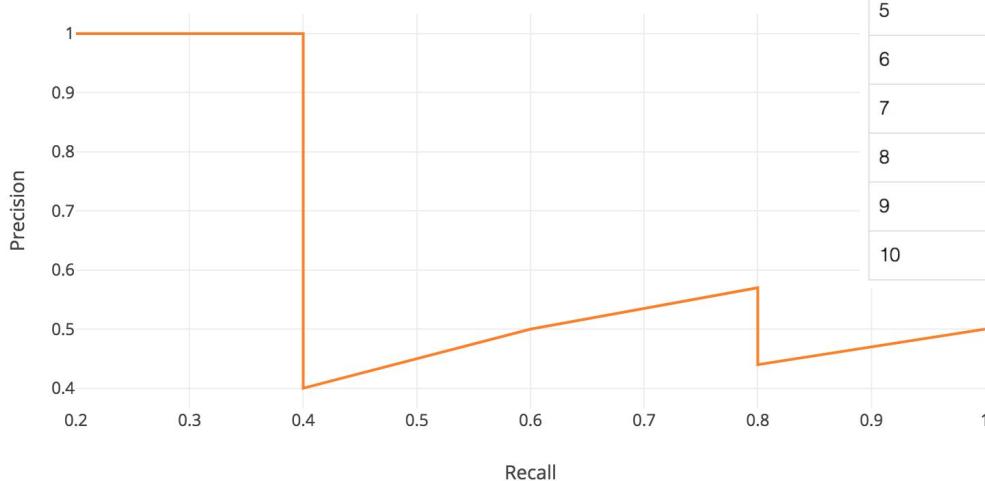


Consider 5 objects and 10 detections

Rank	Correct?	Precision	Recall
1	True	1.0	0.2
2	True	1.0	0.4
3	False	0.67	0.4
4	False	0.5	0.4
5	False	0.4	0.4
6	True	0.5	0.6
7	True	0.57	0.8
8	False	0.5	0.8
9	False	0.44	0.8
10	True	0.5	1.0

mean Average Precision (mAP)

The intention in interpolating the precision/recall curve in this way is to reduce the impact of the “wiggles” in the precision/recall curve, caused by small variations in the ranking of examples.

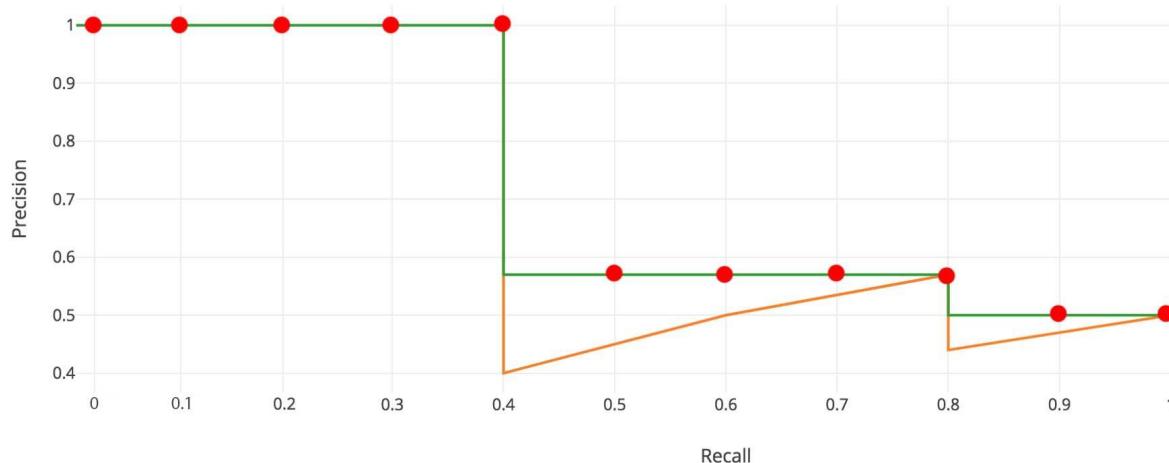


Rank	Correct?	Precision	Recall
1	True	1.0	0.2
2	True	1.0	0.4
3	False	0.67	0.4
4	False	0.5	0.4
5	False	0.4	0.4
6	True	0.5	0.6
7	True	0.57	0.8
8	False	0.5	0.8
9	False	0.44	0.8
10	True	0.5	1.0

mean Average Precision (mAP)

mAP is mean of AP over all classes

$$AP = \frac{1}{11} \times (AP_r(0) + AP_r(0.1) + \dots + AP_r(1.0))$$



COCO mAP

Average Precision (AP):

AP % AP at IoU=.50:.05:.95 (primary challenge metric)
 $AP^{IoU=.50}$ % AP at IoU=.50 (PASCAL VOC metric)
 $AP^{IoU=.75}$ % AP at IoU=.75 (strict metric)

AP Across Scales:

AP^{small} % AP for small objects: area < 32^2
 AP^{medium} % AP for medium objects: $32^2 < \text{area} < 96^2$
 AP^{large} % AP for large objects: area > 96^2

Average Recall (AR):

$AR^{max=1}$ % AR given 1 detection per image
 $AR^{max=10}$ % AR given 10 detections per image
 $AR^{max=100}$ % AR given 100 detections per image

AR Across Scales:

AR^{small} % AR for small objects: area < 32^2
 AR^{medium} % AR for medium objects: $32^2 < \text{area} < 96^2$
 AR^{large} % AR for large objects: area > 96^2

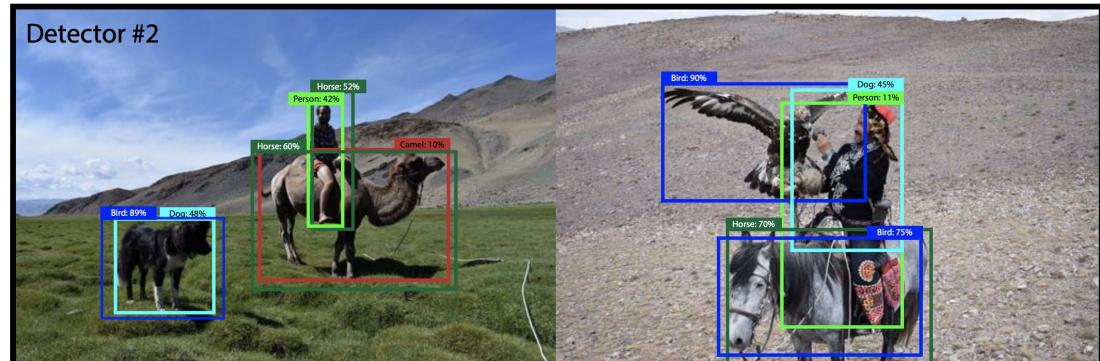
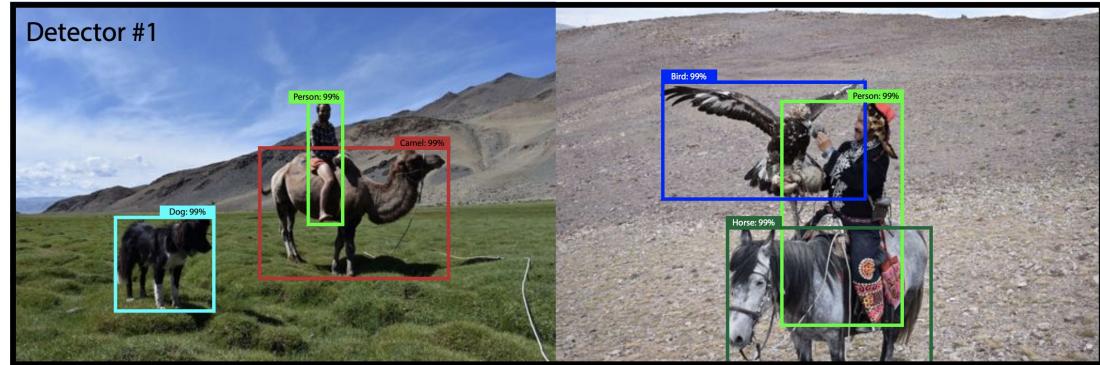
See https://medium.com/@jonathan_hui/map-mean-average-precision-for-object-detection-45c121a31173

mean Average Precision (mAP)

Problems

These two hypothetical detectors are perfect according to mAP over these two images.
They are both perfect. Totally equal

([from YOLOv3 article](#))

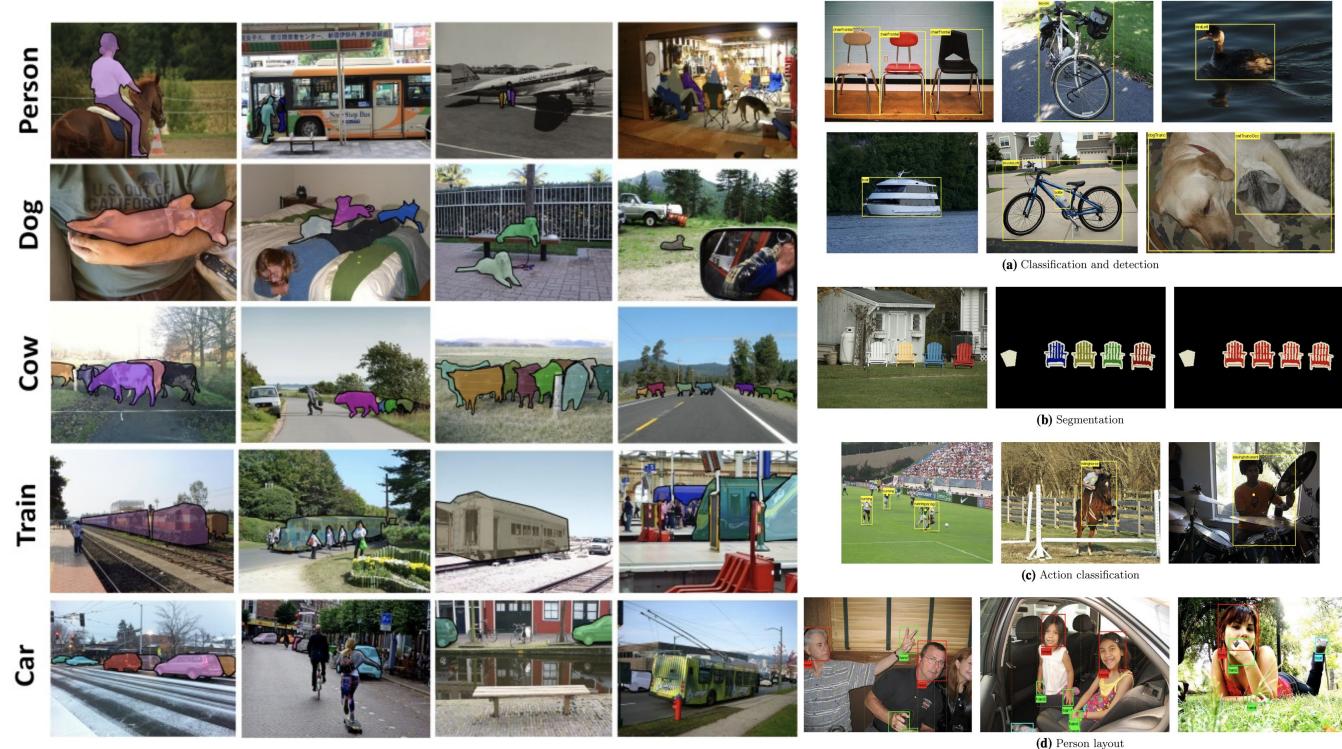


Datasets



Detection datasets

1. PASCAL VOC 2012
2. ImageNet
3. COCO 2017
4. Open Images



Thanks to Ilya Zakharkin
for materials

PASCAL VOC 2012



Table 1 The VOC classes

Vehicles	Household	Animals	Other
Aeroplane	Bottle	Bird	Person
Bicycle	Chair	Cat	
Boat	Dining table	Cow	
Bus	Potted plant	Dog	
Car	Sofa	Horse	
Motorbike	TV/Monitor	Sheep	
Train			

The classes can be considered in a notional taxonomy

Relevant dataset overview:

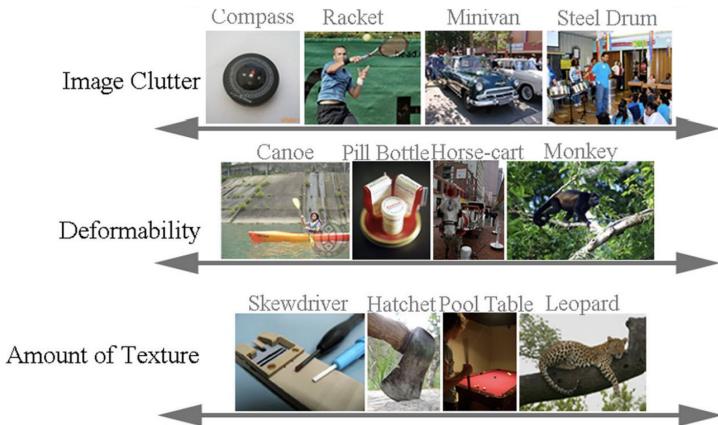
<http://host.robots.ox.ac.uk/pascal/VOC/pubs/everingham15.pdf>

- PASCAL Visual Object Classes
- developed since 2005 and 4 classes
- 11,5k images
- 20 classes
- 31,5k detections
- 7k segmentations
- Images came from flickr.com

ImageNet

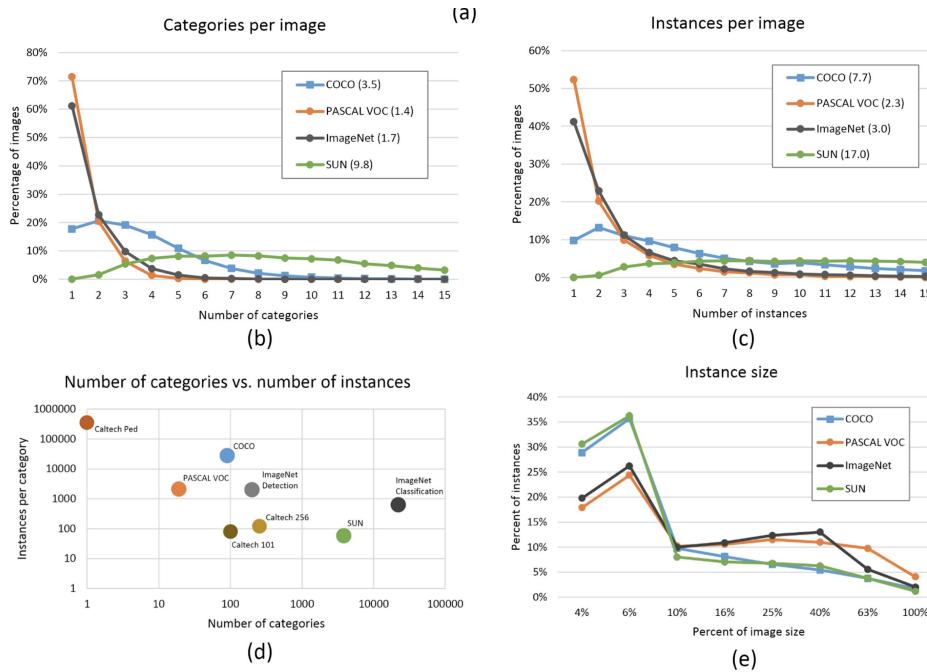
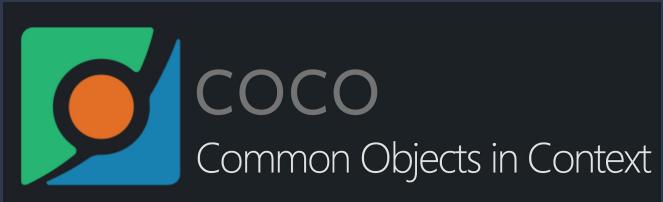


- organized according to the WordNet hierarchy
- 14.2m images
- 1000 classes
- 1m images with BBs
- more fine graded classes than PASCAL
- more diverse objects properties



<http://image-net.org/explore>

MS COCO 2017



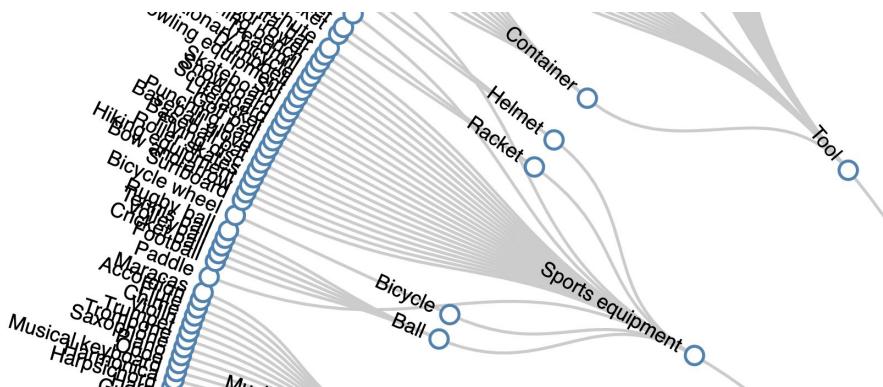
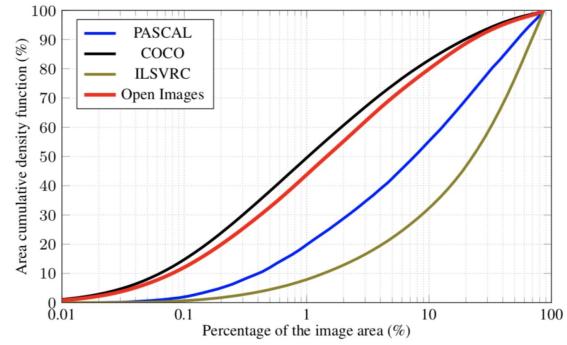
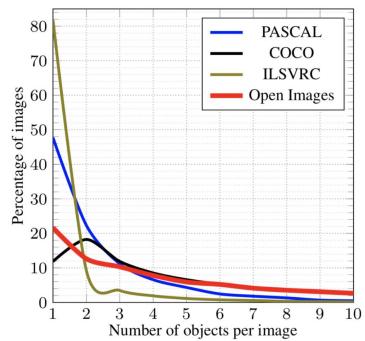
- MicroSoft Common Objects in COnText
- 328k images
- 91 object categories
- 82 having more than 5,000 labeled instances
- 2,5m labeled instances in total
- 250k people with keypoints
- online exploration tool
<http://cocodataset.org/#explore>
- overview <https://arxiv.org/pdf/1405.0312.pdf>

Open Images

- Currently relevant challenge
- 15m boxes on 600 categories
- 2.8m instance segmentations on 350 categories
- 36.5m image-level labels on 20k categories
- 391k relationship annotations of 329 relationships
- Extension - 478k crowdsourced images with 6k+ categories



<https://storage.googleapis.com/openimages/web/factsfigures.html>



R-CNN



Detection as Classification



CAT? NO

DOG? NO

Detection as Classification

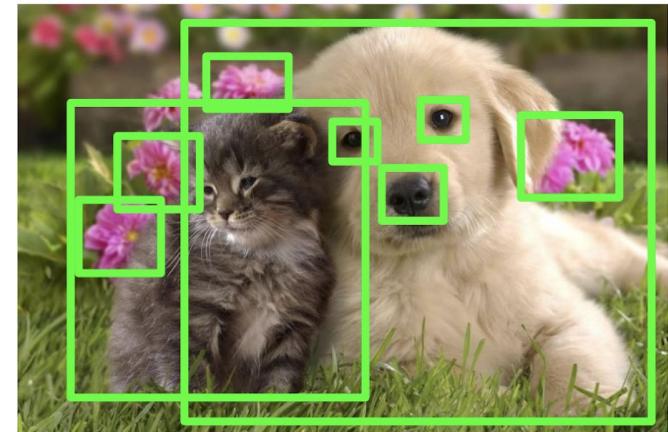


CAT? YES!

DOG? NO

Region Proposals

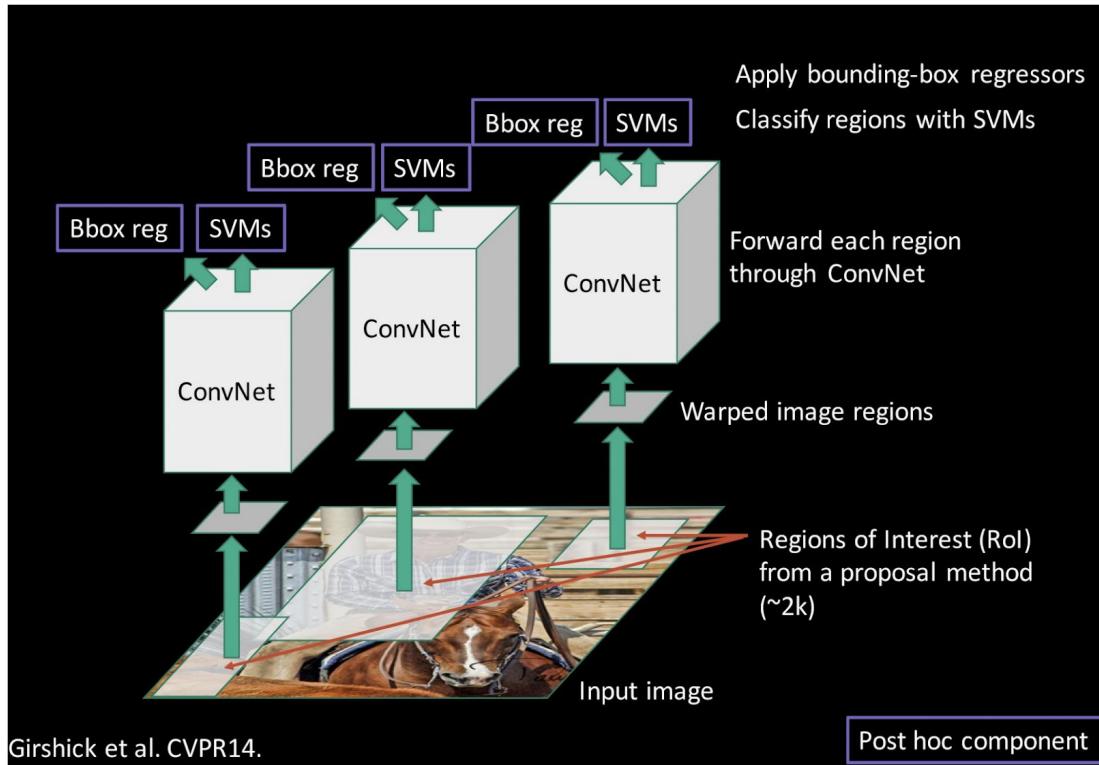
- Find “blobby” image regions that are likely to contain objects
- “Class-agnostic” object detector
- Look for “blob-like” regions



Region Proposals: Many other choices

Method	Approach	Outputs Segments	Outputs Score	Control #proposals	Time (sec.)	Repea- tability	Recall Results	Detection Results
Bing [18]	Window scoring		✓	✓	0.2	***	*	.
CPMC [19]	Grouping	✓	✓	✓	250	-	**	*
EdgeBoxes [20]	Window scoring		✓	✓	0.3	**	***	***
Endres [21]	Grouping	✓	✓	✓	100	-	***	**
Geodesic [22]	Grouping	✓		✓	1	*	***	**
MCG [23]	Grouping	✓	✓	✓	30	*	***	***
Objectness [24]	Window scoring		✓	✓	3	.	*	.
Rahtu [25]	Window scoring		✓	✓	3	.	.	*
RandomizedPrim's [26]	Grouping	✓		✓	1	*	*	**
Rantalankila [27]	Grouping	✓		✓	10	**	.	**
Rigor [28]	Grouping	✓		✓	10	*	**	**
SelectiveSearch [29]	Grouping	✓	✓	✓	10	**	***	***
Gaussian				✓	0	.	.	*
SlidingWindow				✓	0	***	.	.
Superpixels		✓			1	*	.	.
Uniform				✓	0	.	.	.

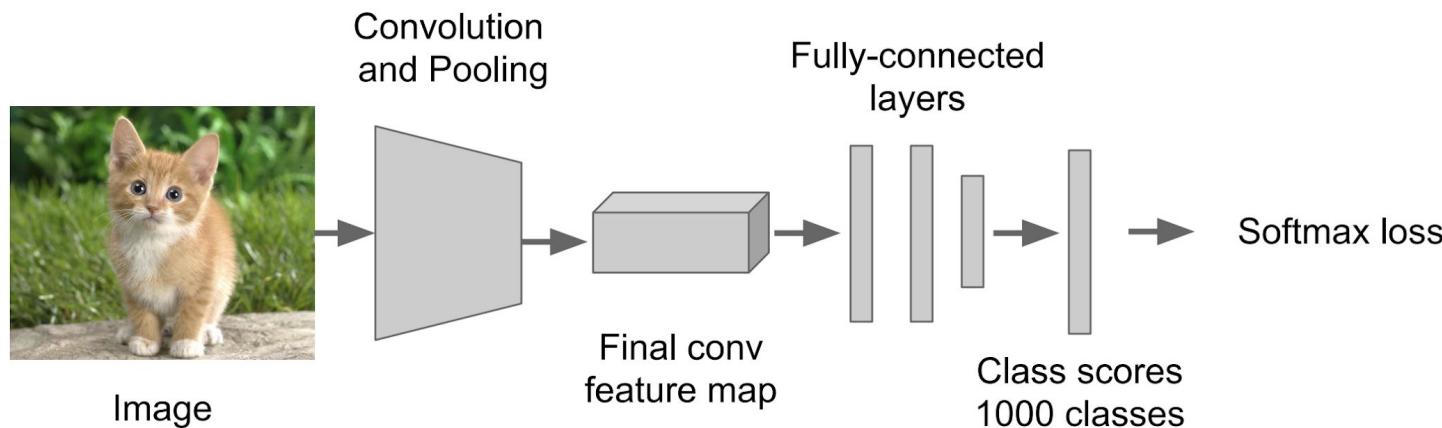
Putting it together: R-CNN



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

R-CNN Training

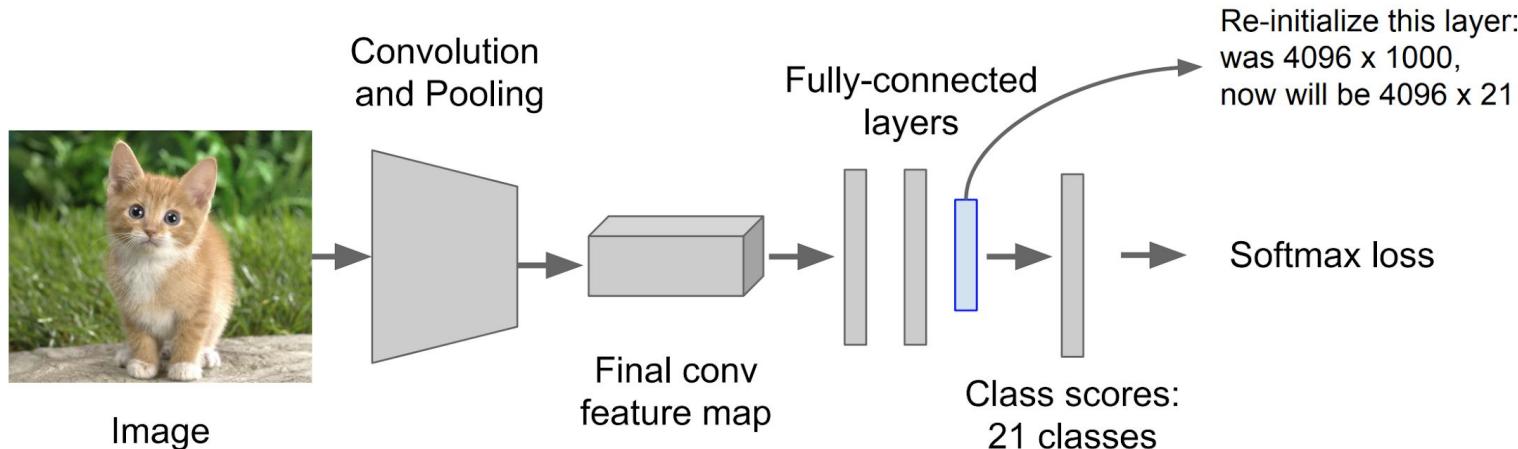
Step 1: Train (or download) a classification model for ImageNet (AlexNet)



R-CNN Training

Step 2: Fine-tune model for detection

- Instead of 1000 ImageNet classes, want 20 object classes + background
- Throw away final fully-connected layer, reinitialize from scratch
- Keep training model using positive / negative regions from detection images



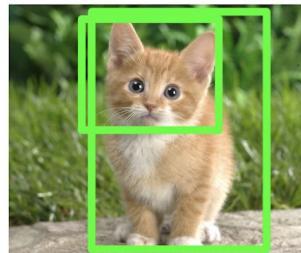
R-CNN Training

Step 3: Extract features

- Extract region proposals for all images
- For each region: warp to CNN input size, run forward through CNN, save pool5 features to disk
- Have a big hard drive: features are ~200GB for PASCAL dataset!



Image

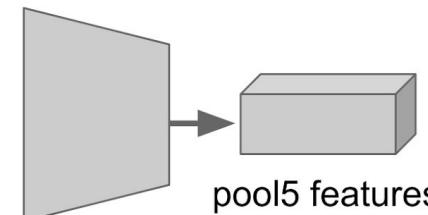


Region Proposals

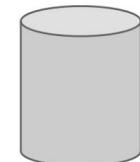


Crop + Warp

Convolution
and Pooling



Forward pass



Save to disk

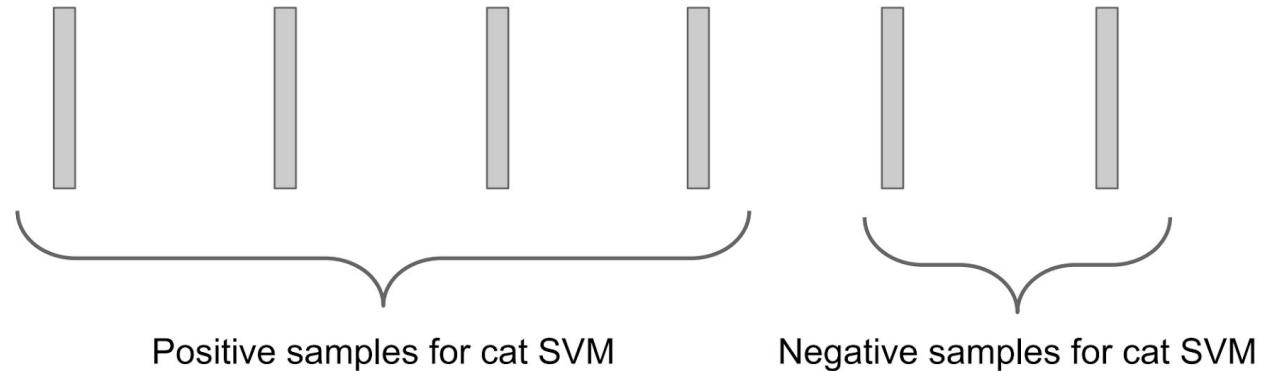
R-CNN Training

Step 4: Train one binary SVM per class to classify region features

Training image regions



Cached region features



R-CNN Training

Step 5 (bbox regression): For each class, train a linear regression model to map from cached features to offsets to GT boxes to make up for “slightly wrong” proposals

Training image regions



Cached region features



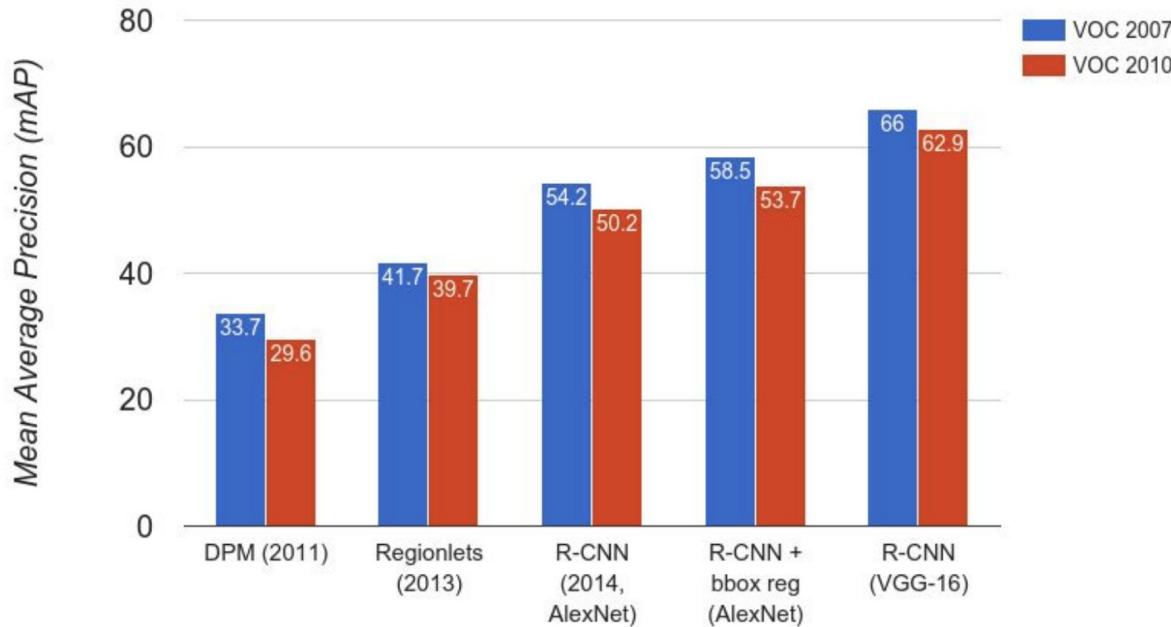
Regression targets
(dx , dy , dw , dh)
Normalized coordinates

$(0, 0, 0, 0)$
Proposal is good

$(.25, 0, 0, 0)$
Proposal too far to left

$(0, 0, -0.125, 0)$
Proposal too wide

R-CNN Results

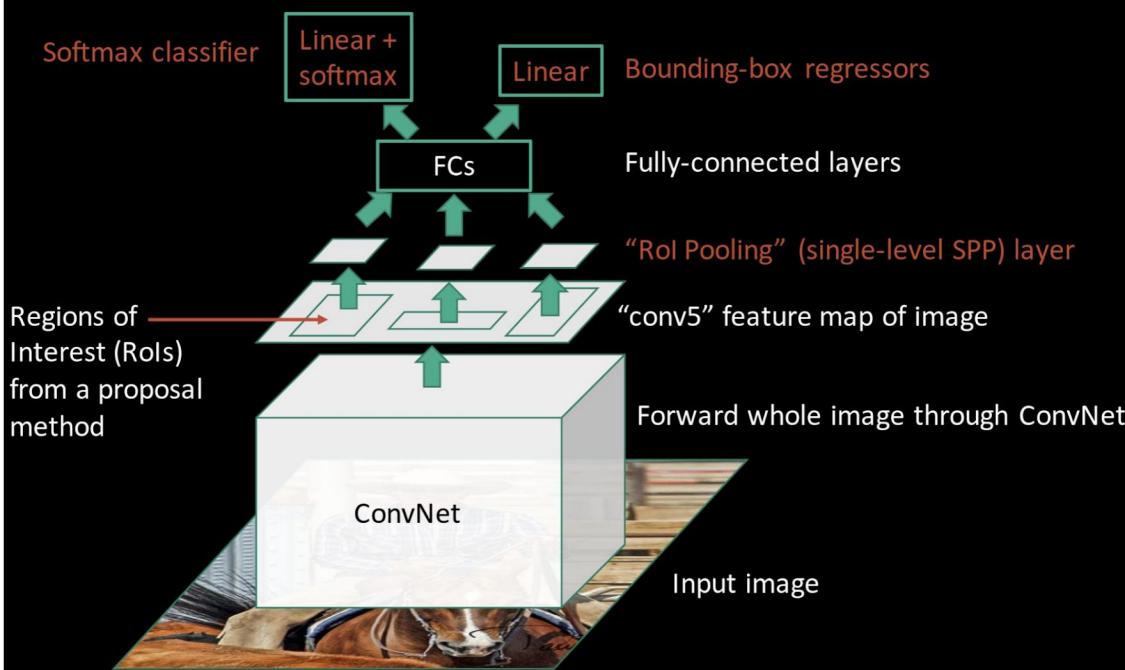


R-CNN Problems

1. Slow at test-time: need to run full forward pass of CNN for each region proposal
2. SVMs and regressors are post-hoc: CNN features not updated in response to SVMs and regressors
3. Complex multistage training pipeline

Fast R-CNN

Fast R-CNN (test time)

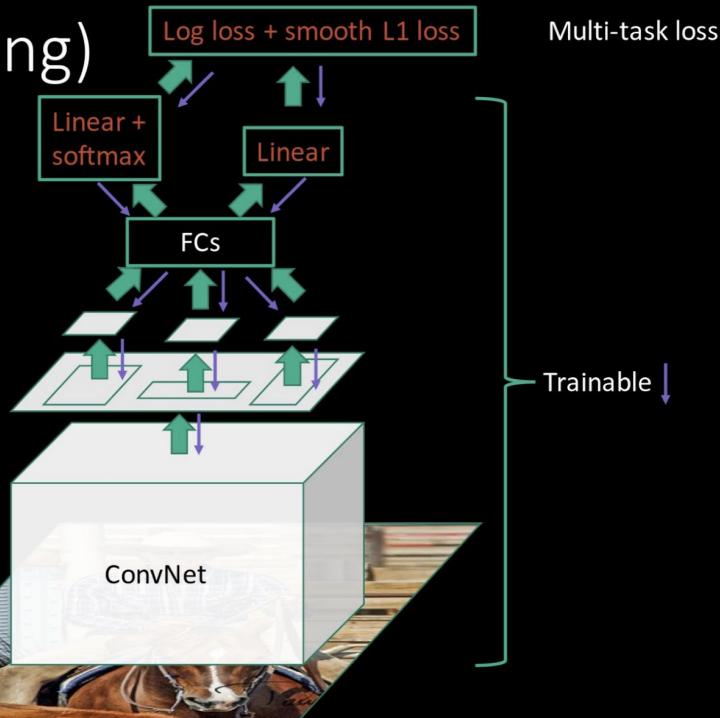


R-CNN Problem #1:
Slow at test-time due to
independent forward
passes of the CNN

Solution:
Share computation
of convolutional
layers between
proposals for an
image

Fast R-CNN

Fast R-CNN
(training)



R-CNN Problem #2:

Post-hoc training: CNN not updated in response to final classifiers and regressors

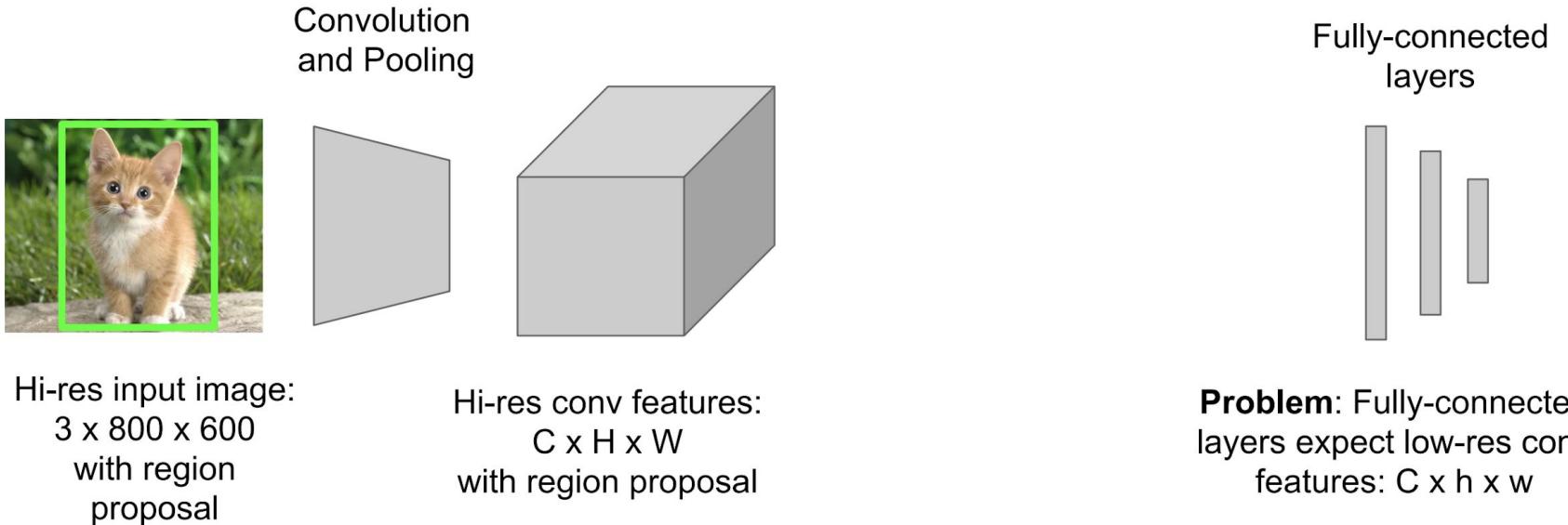
R-CNN Problem #3:

Complex training pipeline

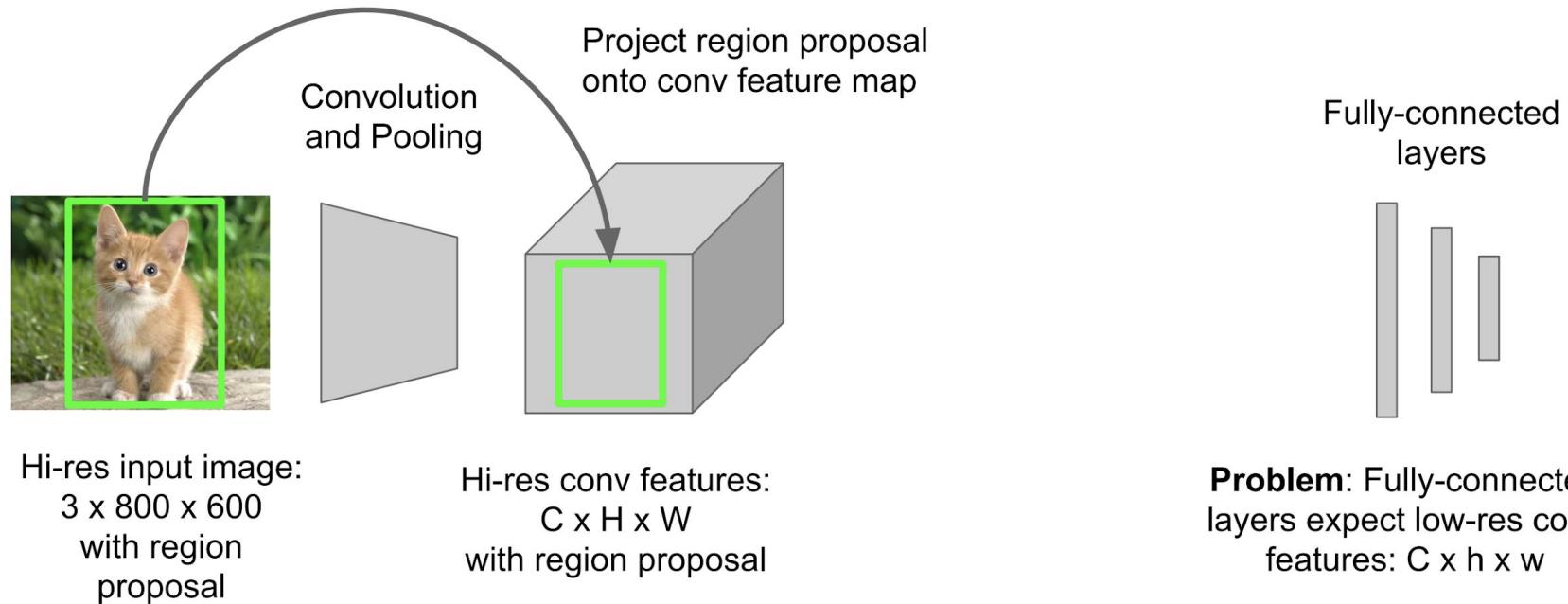
Solution:

Just train the whole system end-to-end all at once!

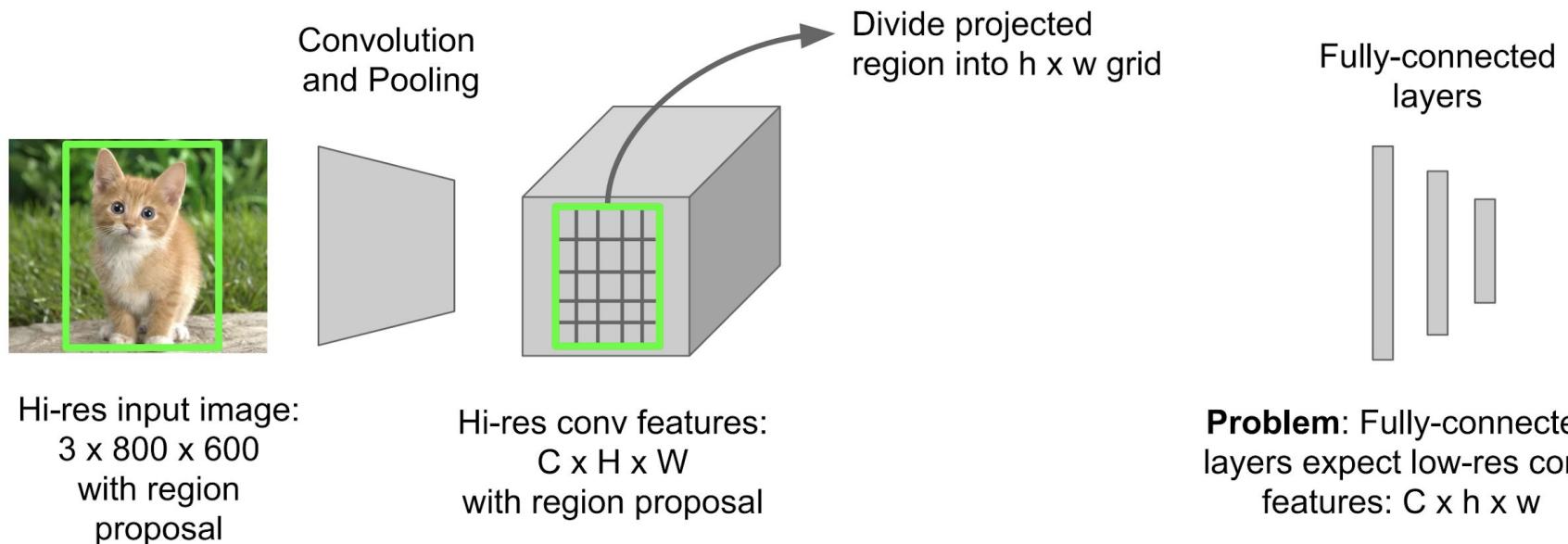
Fast R-CNN: Region of Interest Pooling



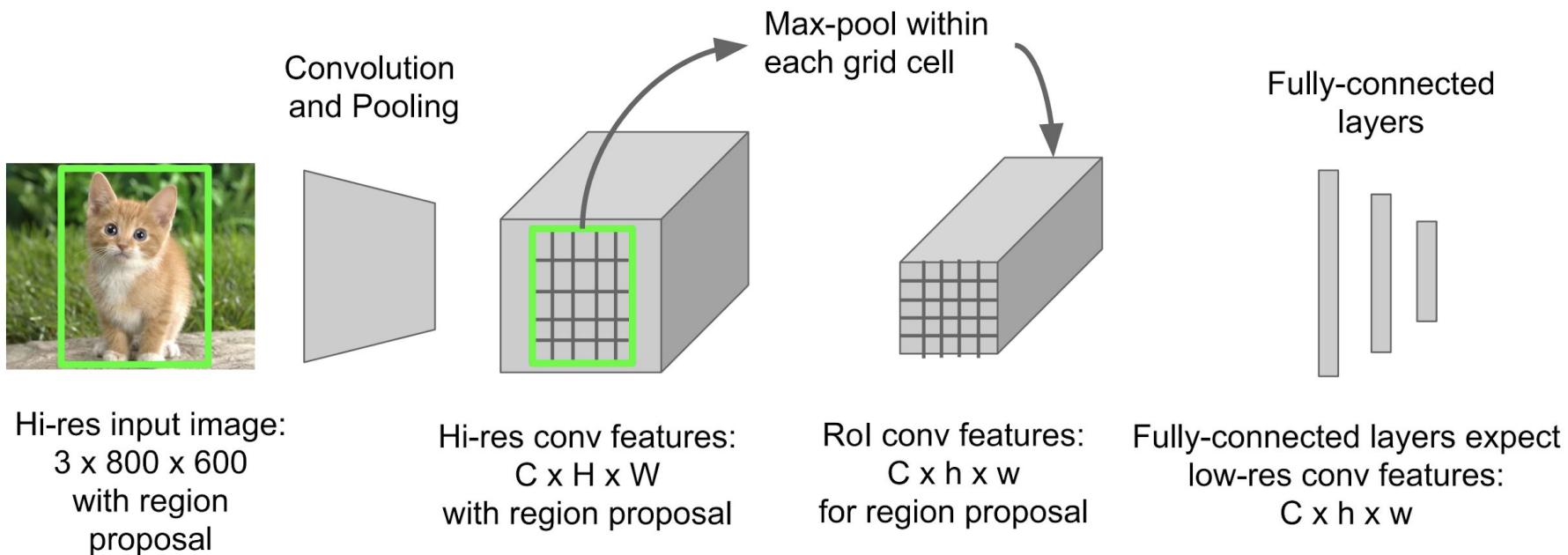
Fast R-CNN: Region of Interest Pooling



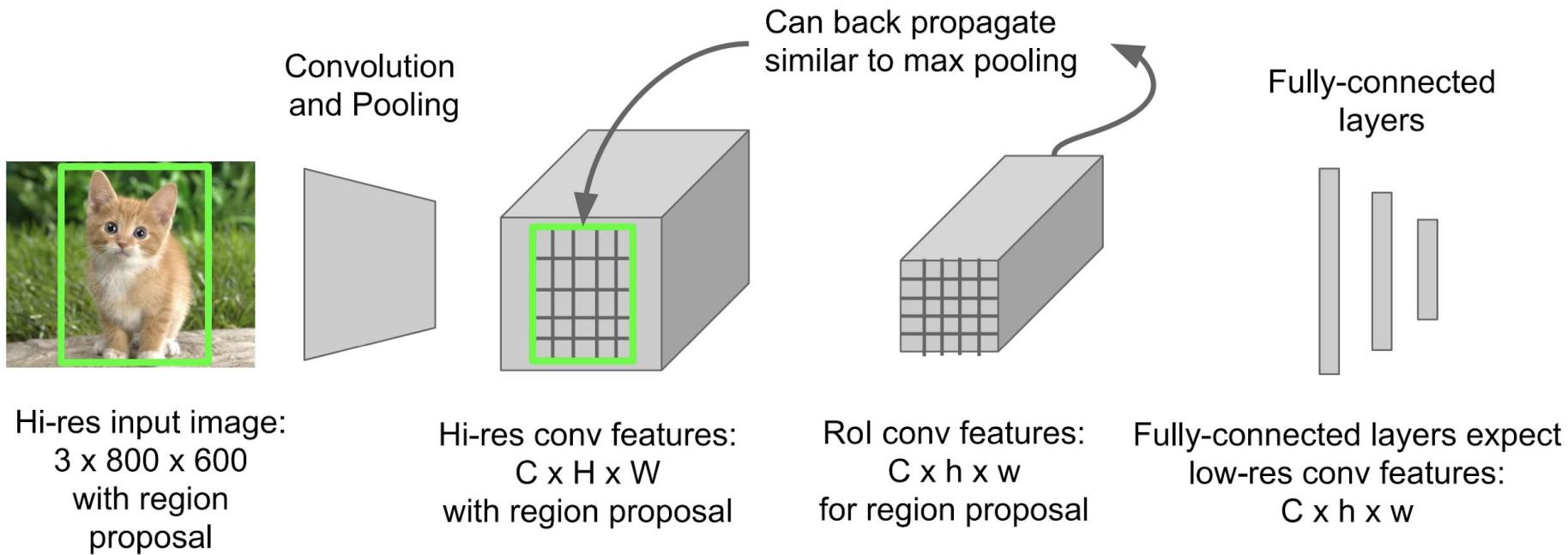
Fast R-CNN: Region of Interest Pooling



Fast R-CNN: Region of Interest Pooling



Fast R-CNN: Region of Interest Pooling



Fast R-CNN Results

Faster!

FASTER!

Better!

	R-CNN	Fast R-CNN
Training Time:	84 hours	9.5 hours
(Speedup)	1x	8.8x
Test time per image	47 seconds	0.32 seconds
(Speedup)	1x	146x
mAP (VOC 2007)	66.0	66.9

Using VGG-16 CNN on Pascal VOC 2007 dataset

Fast R-CNN Problem

Test-time speeds don't include region proposals

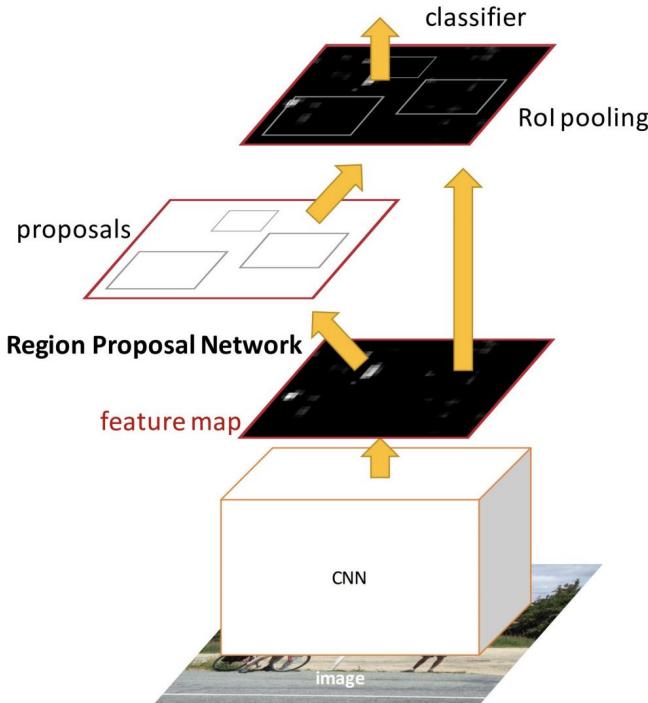
	R-CNN	Fast R-CNN
Test time per image	47 seconds	0.32 seconds
(Speedup)	1x	146x
Test time per image with Selective Search	50 seconds	2 seconds
(Speedup)	1x	25x

Fast R-CNN Problem Solution

Test-time speeds don't include region proposals
Just make the CNN do region proposals too!

	R-CNN	Fast R-CNN
Test time per image	47 seconds	0.32 seconds
(Speedup)	1x	146x
Test time per image with Selective Search	50 seconds	2 seconds
(Speedup)	1x	25x

Faster R-CNN



Insert a **Region Proposal Network (RPN)** after the last convolutional layer

RPN trained to produce region proposals directly; no need for external region proposals!

After RPN, use RoI Pooling and an upstream classifier and bbox regressor just like Fast R-CNN

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

Slide credit: Ross Girshick

Faster R-CNN: Region Proposal Network

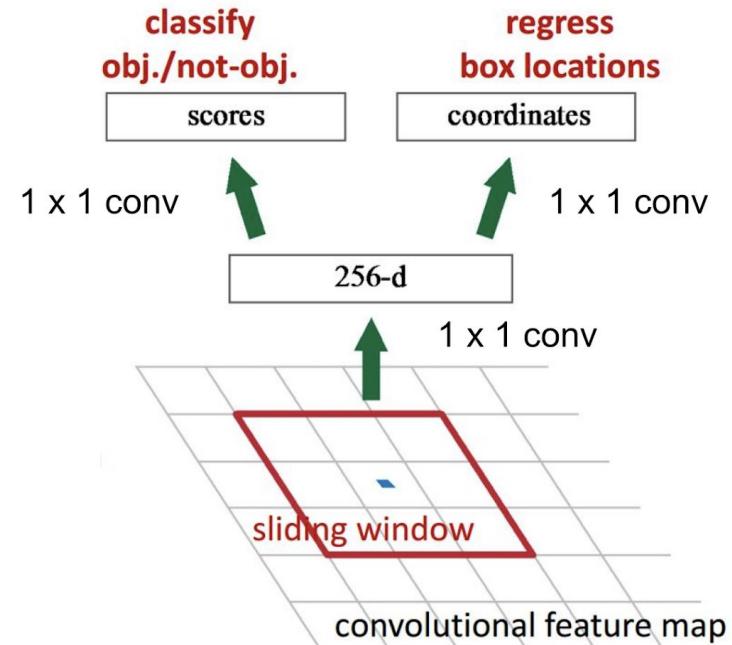
Slide a small window on the feature map

Build a small network for:

- classifying object or not-object, and
- regressing bbox locations

Position of the sliding window provides localization information with reference to the image

Box regression provides finer localization information with reference to this sliding window



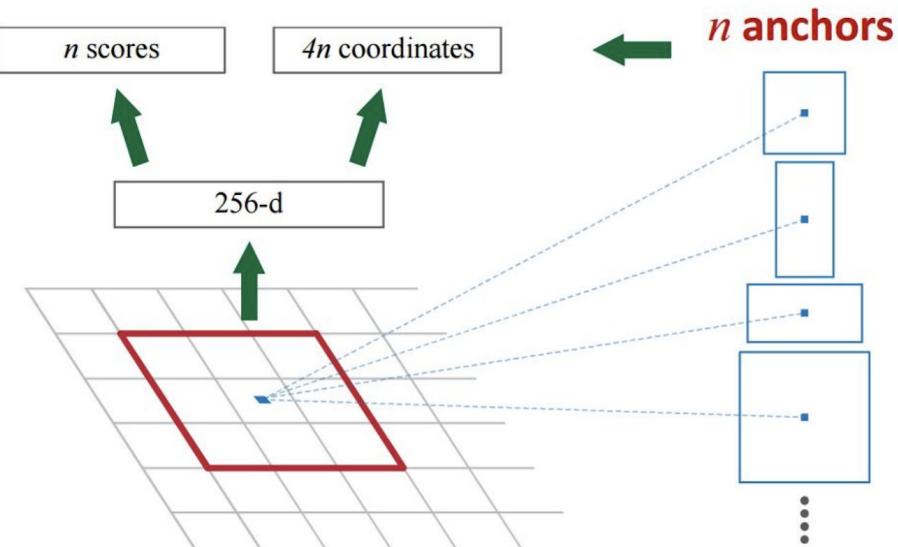
Faster R-CNN: Region Proposal Network

Use **N anchor boxes** at each location

Anchors are **translation invariant**: use the same ones at every location

Regression gives offsets from anchor boxes

Classification gives the probability that each (regressed) anchor shows an object



Faster R-CNN: Training

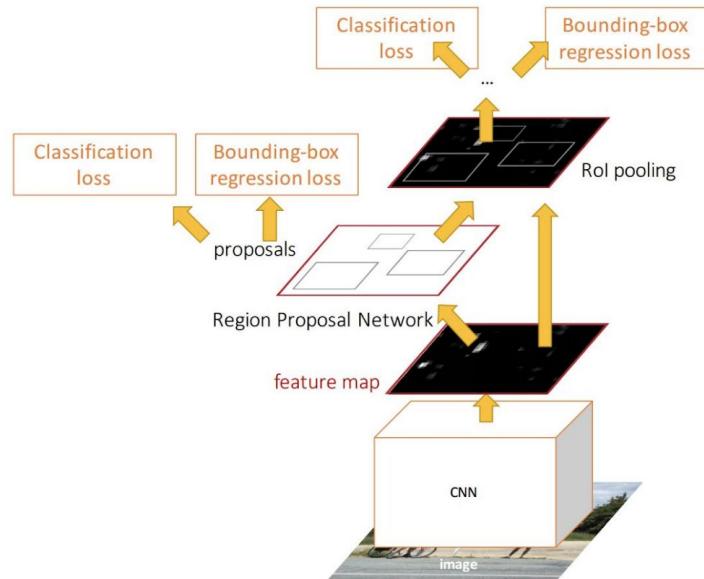
In the paper: Ugly pipeline

- Use alternating optimization to train RPN, then Fast R-CNN with RPN proposals, etc.
- More complex than it has to be

Since publication: Joint training!

One network, four losses

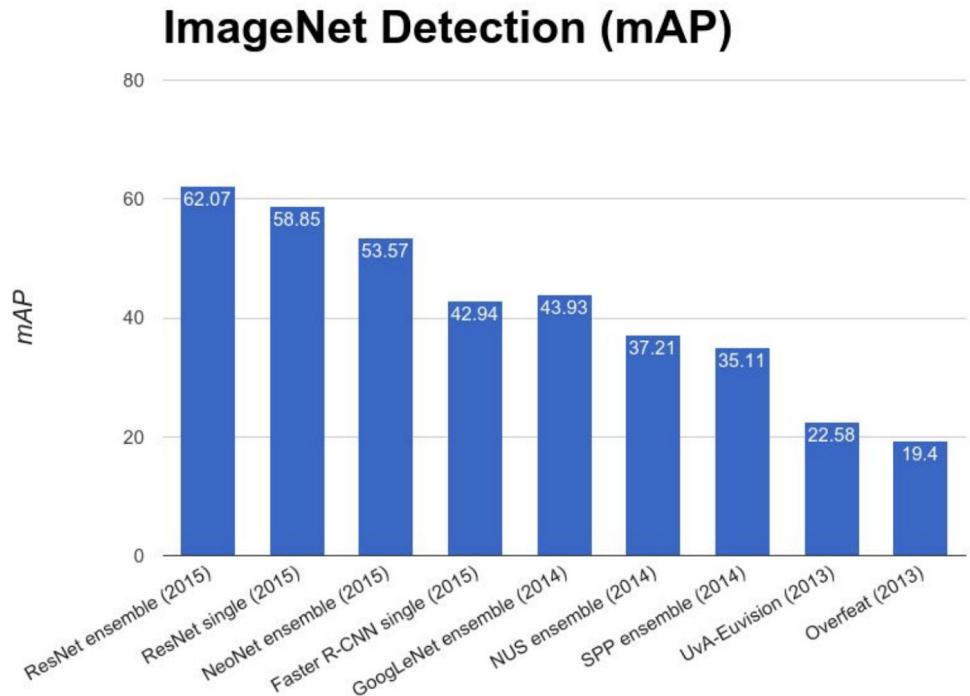
- RPN classification (anchor good / bad)
- RPN regression (anchor \rightarrow proposal)
- Fast R-CNN classification (over classes)
- Fast R-CNN regression (proposal \rightarrow box)



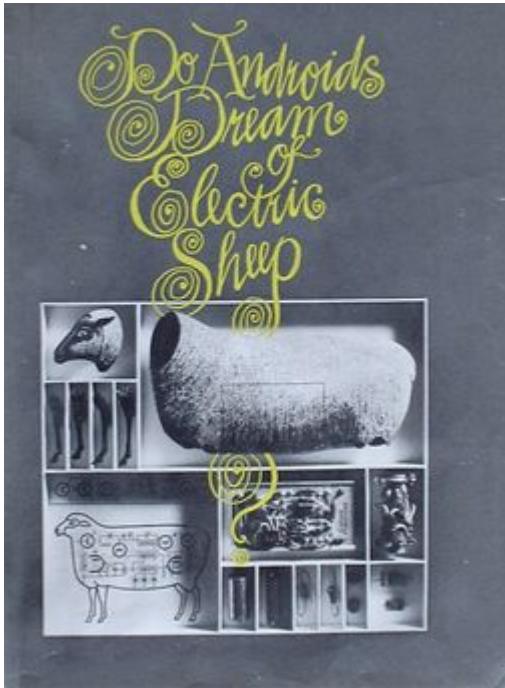
Faster R-CNN: Results

	R-CNN	Fast R-CNN	Faster R-CNN
Test time per image (with proposals)	50 seconds	2 seconds	0.2 seconds
(Speedup)	1x	25x	250x
mAP (VOC 2007)	66.0	66.9	66.9

ImageNet Detection 2013 - 2015



More CV to come...



Next time will:

- try to Only Look Once at an image
- know Do Androids Dream of Electric Sheep?