Training Region-based Object Detectors using Online Hard Example Mining (OHEM)

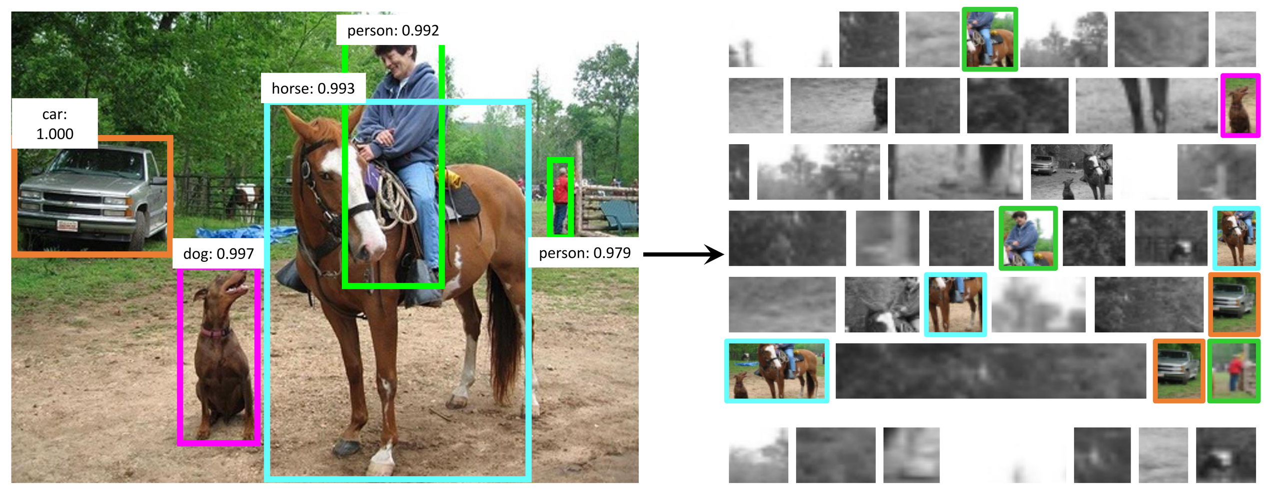
Carnegie Mellon University — The Robotics Institute

Abhinav Shrivastava Abhinav Gupta Ross Girshick

Microsoft Research Facebook AI Research

Object Detection

Generally *reduced* to Image Classification:



- This *reduction* introduces new problems unique to detection
- Huge imbalance b/w annotated Foreground (fg) objects & Background (bg) examples

Setting	fg : bg
Sliding-window (e.g., DPM)	1 : 100,000
Region-based (e.g., R-CNN)	1 : 70

Bootstrapping to the rescue!

Referred to as Hard Negative Mining

Simple, yet powerful, algorithm:
1. Fix Training Set
   Update Model
2. Freeze Model
   Find Hard Negatives
3. Iterate

- Existed for at least 20 years! [Sung and Poggio, 1994]
- Standard way to deal with fg:bg imbalance
- Widespread use since mid-1990s for object detection

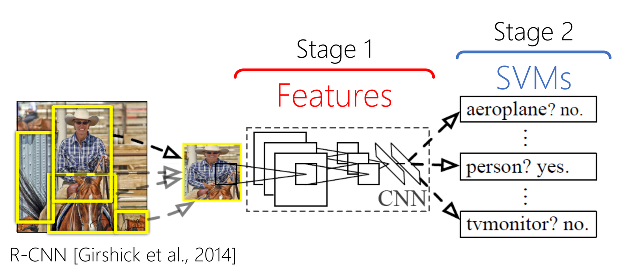
Mainstay in Object Detection for >20 years

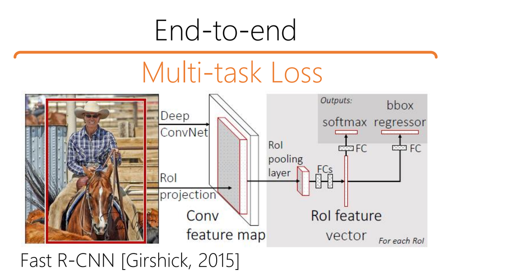
Shallow Neural Networks [Sung & Poggio, 1994] [Rowley et al., 1998]
SVMs, LSVMs [Dalal & Triggs, 2005] DPM [Felzenszwalb et al., 2010]
Boosted Decision Trees [Dollár et al., 2009]
ConvNet Features + SVMs R-CNN [Girshick et al., 2014] SPPNet [He et al., 2014]

Why don't state-of-the-art detectors use bootstrapping anymore?

Multi-stage pipelines are being replaced by end-to-end systems

Multi-stage Training: ConvNet Features + SVMs R-CNN [Girshick et al., 2014] SPPNet [He et al., 2014] → End-to-end Training: ConvNet Detectors Fast R-CNN [Girshick, 2015] Faster R-CNN [Ren et al., 2015]





Hard Negative Mining for SVMs Stochastic Gradient Descent (SGD)

Why is standard bootstrapping not trivial in SGD?

Bootstrapping:
1. Fix Training Set
   Update Model
2. Freeze Model
   Find Hard Negatives
3. Iterate

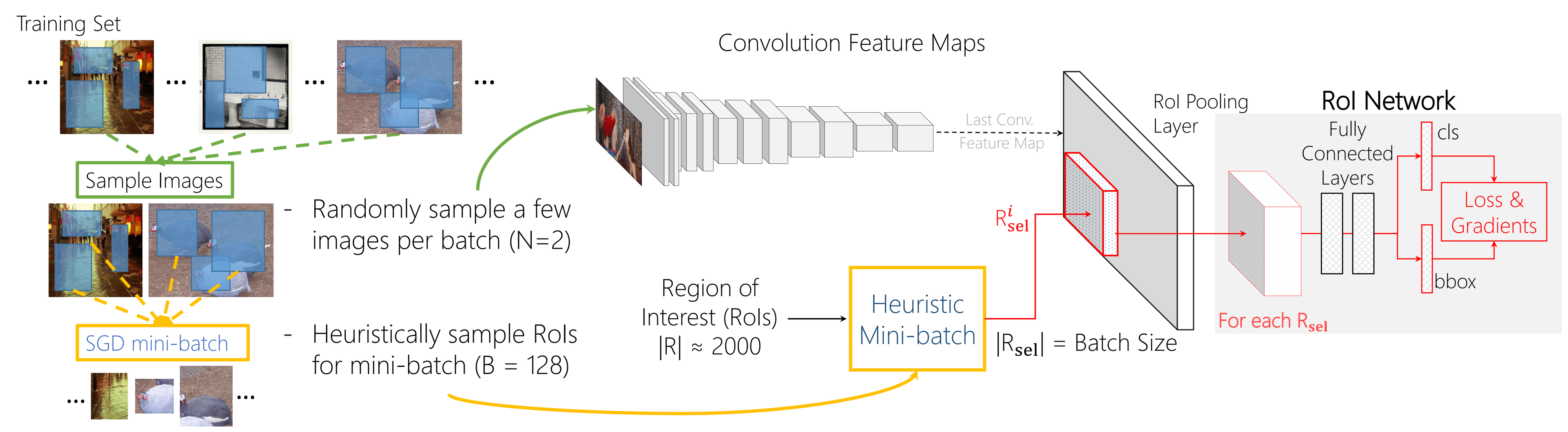
Training Object Detector:
- Trained online using SGD
- Requires 100,000s of iterations
- Freezing the model slows training
- As model becomes better, the problem become worse

Need a purely online method to select hard examples, that plays nicely with SGD.

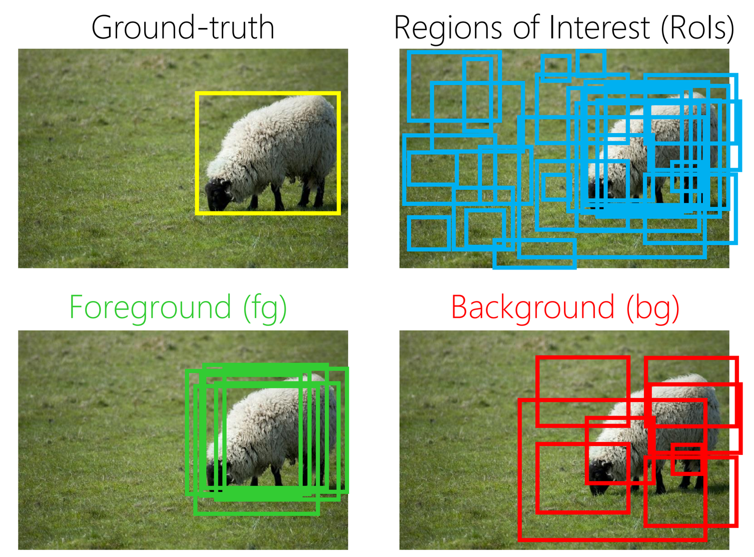
Training a ConvNet Detector (Fast R-CNN)

Stochastic Gradient Descent (SGD) version:

Generic paradigm used in most Region-based Object Detectors; e.g., R-CNN [Girshick et al., 2014], SPPNet [He et al., 2014], Fast R-CNN [Girshick, 2015], Faster R-CNN [Ren et al., 2015], MR-CNN [Gidaris & Konodakis, 2015] etc.



RoI Sampling Heuristics for SGD Mini-batch



Foreground RoIs:
- RoIs with IoU ≧ 0.5 with any GT
- Inspired by VOC eval. protocol

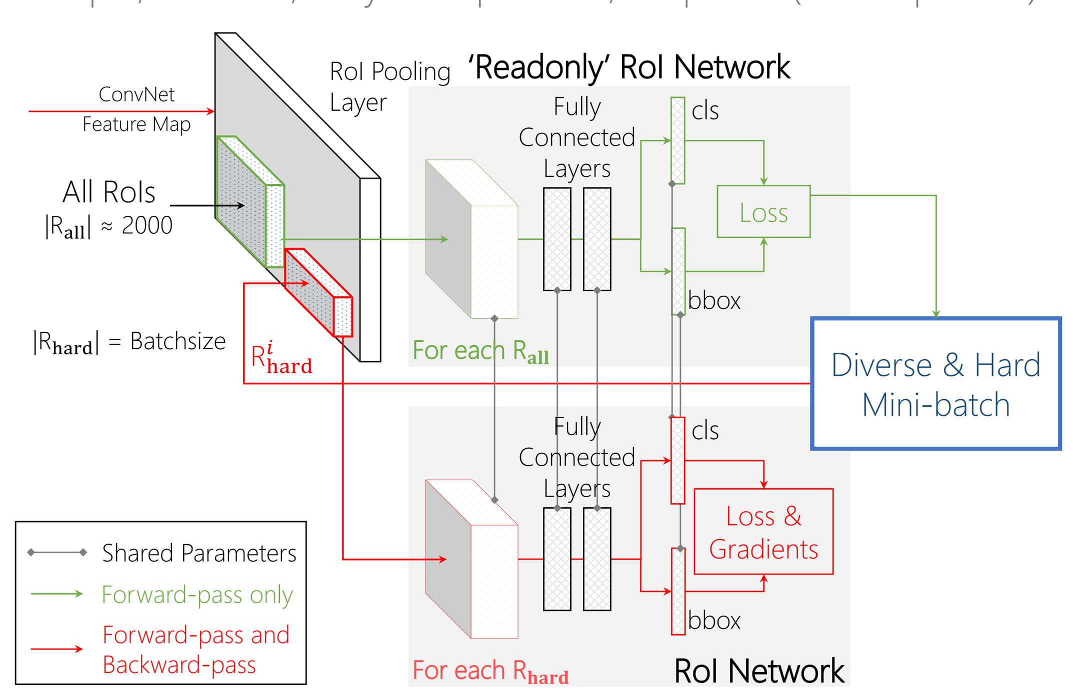
Background RoIs:
- RoIs with max IoU in [bg_lo, 0.5)
- bg_lo used to approx. hard mining
- Sub-optimal: Ignores RoIs with no GT overlap

fg-bg RoIs Ratio in mini-batch:
- To balance fg:bg RoIs

Standard Settings:
- fg:bg = 1:3 (25% fg RoIs/batch)
- bg_lo = 0.1

Online Hard Example Mining (OHEM) + SGD version:

Simple, effective, easy to implement, simplified (and improved) training, consistent and significant improvements!



Key Insights:
- Even though a few images are sampled (N=2), each image has 1000s of RoIs.
- Replace Heuristic Sampling with Hard Example Sampling

How to freeze the model efficiently to find hard examples?
- Forward-pass is already freezes the model, exactly for one SGD iteration

How is it online?
- Hard Examples sampling is performed inline with online SGD iteration

Is this efficient?
- ConvNet forward-backward pass and RoI Network backward-pass remain intact
- Only addition is RoI Network forward-pass

Diverse & Hard Mini-batch Sampler:
1. Sort RoIs based on loss
2. Do non-max suppression for de-duplication
3. Select top B (=128) RoIs

	VGGM Heuristic	VGGM OHEM	VGG16 Heuristic	VGG16 OHEM
time (sec/iter)	0.13	0.22	0.57	0.87
max. memory (G)	2.6	3.6	6.4	7.7

Using Nvidia Titan X, gradient accumulation for VGG16

Why use OHEM?

Starter code available! https://git.io/voSj9

- Simple and easy to implement
- Simplifies training: reduces costly to tune hyperarameters.
- Results in better training and higher performance.

Why non-max suppression (NMS)?
- Co-located RoIs = Co-related loss
- Res. disparity = Loss double counting

OHEM: Main Results (VOC07, VOC12, COCO)

Method	M	B	train set	07 mAP
FRCN			07	66.9
Ours			07	69.9
FRCN	✓	✓	07	72.4
MR-CNN	✓	✓	07	74.9
Ours	✓	✓	07	75.1
FRCN			07+12	70.0
Ours			07+12	74.6
MR-CNN	✓	✓	07+12	78.2
Ours	✓	✓	07+12	78.9

Method	M	B	train set	12 mAP
FRCN			12	65.7
Ours			12	69.8
MR-CNN	✓	✓	12	70.7
Ours	✓	✓	12	72.9
FRCN			07++12	68.4
Ours			07++12	71.9
MR-CNN	✓	✓	07++12	73.9
Ours	✓	✓	07++12	76.3
Ours	✓	✓	+COCO	80.1

All methods use VGG16.

Method key:
FRCN: Fast R-CNN [Girshick, 15],
MR-CNN: [Gidaris & Konodakis, 15],
Ours: FRCN+OHEM

Legend:
M: Multi-scale training & testing (from SPPNet),
B: Iterative bbox regression (from MR-CNN).
07 mAP: VOC 2007 test
12 mAP: VOC 2012 test server

Training set key: 07: VOC 2007 trainval, 12: VOC 2012 trainval, 07+12: union of 07 and 12, 07++12: union of 07, VOC 2007 test and 12, +COCO: a model trained on COCO trainval and fine-tuned on 07++12.

COCO test-dev AP@IoU	area	FRCN	Ours	Ours [+M]	Ours* [+M]
[0.50:0.95]	all	19.7	22.6	24.4	25.6
0.50	all	35.9	42.5	44.4	46.0
0.75	all	19.9	22.2	24.8	26.3
[0.50:0.95]	small	3.5	5.0	7.1	7.8
[0.50:0.95]	med.	18.8	23.7	26.4	27.9
[0.50:0.95]	large	34.6	37.9	38.5	40.5

*: trained on trainval.

- OHEM consistently & significantly improves performance
- Best amongst methods w/ VGG16 on the VOC leaderboard
- Orthogonal to other bells and whistles

Bells and Whistles: Ablative Analysis

Impact of Multi-scale & Iterative Bbox Regression

Multi-scale (M) Train	Multi-scale (M) Test	Iterative Bbox Reg. (B)	07 mAP FRCN	07 mAP Ours
			67.2	69.9
	✓		68.4	71.1
		✓	70.8	72.7
	✓	✓	71.9	74.1
✓			67.7	70.7
✓	✓		68.6	71.9
✓		✓	71.2	72.9
✓	✓	✓	72.4	75.1

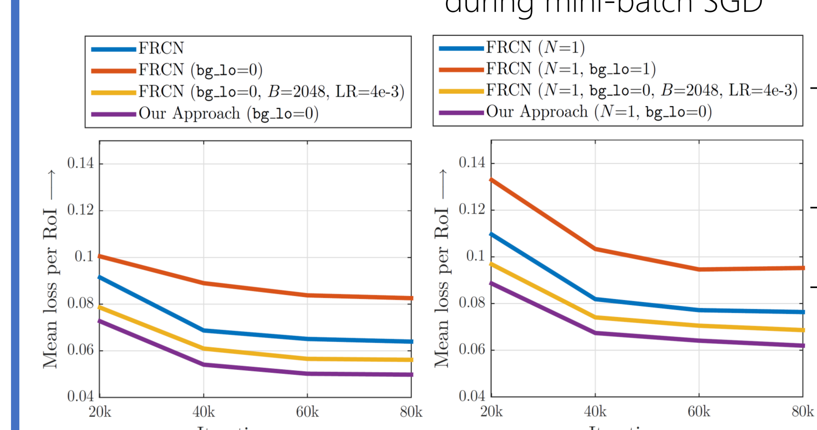
Understanding OHEM

Ablation analysis and more

Experiment	Model	N	LR	B	bg_lo	07 mAP
Fast R-CNN	VGGM	2	10^{-3}	128	0.1	59.6
	VGG16					67.2
Online Hard Example Mining	VGG16	1	10^{-3}	128	0	69.7
	VGGM	2	10^{-3}	128	0	62.0
	VGG16					69.9
Removing hard mining heuristic	VGGM	2	10^{-3}	128	0	57.2
	VGG16					67.5
Robust Gradient Estimates	VGG16	1	10^{-3}	128	0.1	66.3
					0	66.3
Bigger batch, High LR	VGGM	1	4×10^{-3}	2048	0	57.7
		2				60.4
	VGG16	1	3×10^{-3}	2048	0	67.5
		2				68.8

Better Optimization

Study the empirical training loss (mean loss per RoI) during mini-batch SGD



- Using all RoIs leads to lower training loss than other methods
- OHEM results in lowest training loss
- Interesting future work to investigate this theoretically

Online Hard Mining vs. Heuristics

Removing hard mining heuristic

- bg_lo = 0.1 used to approximate hard negative mining
  - +1 mAP for VGGM, no impact VGG16
- Sub-optimal: ignores hard RoIs (e.g., paintings)
- OHEM naturally doesn't require this heuristic & automatically selects hard examples

Why just hard when you can see all?

Bigger batch, High LR

- When using all RoIs:
  - Too many easy RoIs (∼0 loss) dilute the impact of useful (hard) RoIs
  - Need to carefully adjust the LR to account for a larger batch-size
  - +1 mAP
- OHEM outperforms this heuristic & is much faster to train w/o the need to tweak hyperparameters

Robust Gradient Estimates

- Fewer images in a mini-batch leads to correlated RoIs, unstable gradients and/or slower convergence
- For FRCN, -1 mAP for N=1 (vs. N=2)
- No impact for OHEM; demonstrates robustness

Foreground-background Ratio

- Standard fg:bg = 0.25 (fiddling leads to -3%)
- OHEM chooses the distribution based on image contents