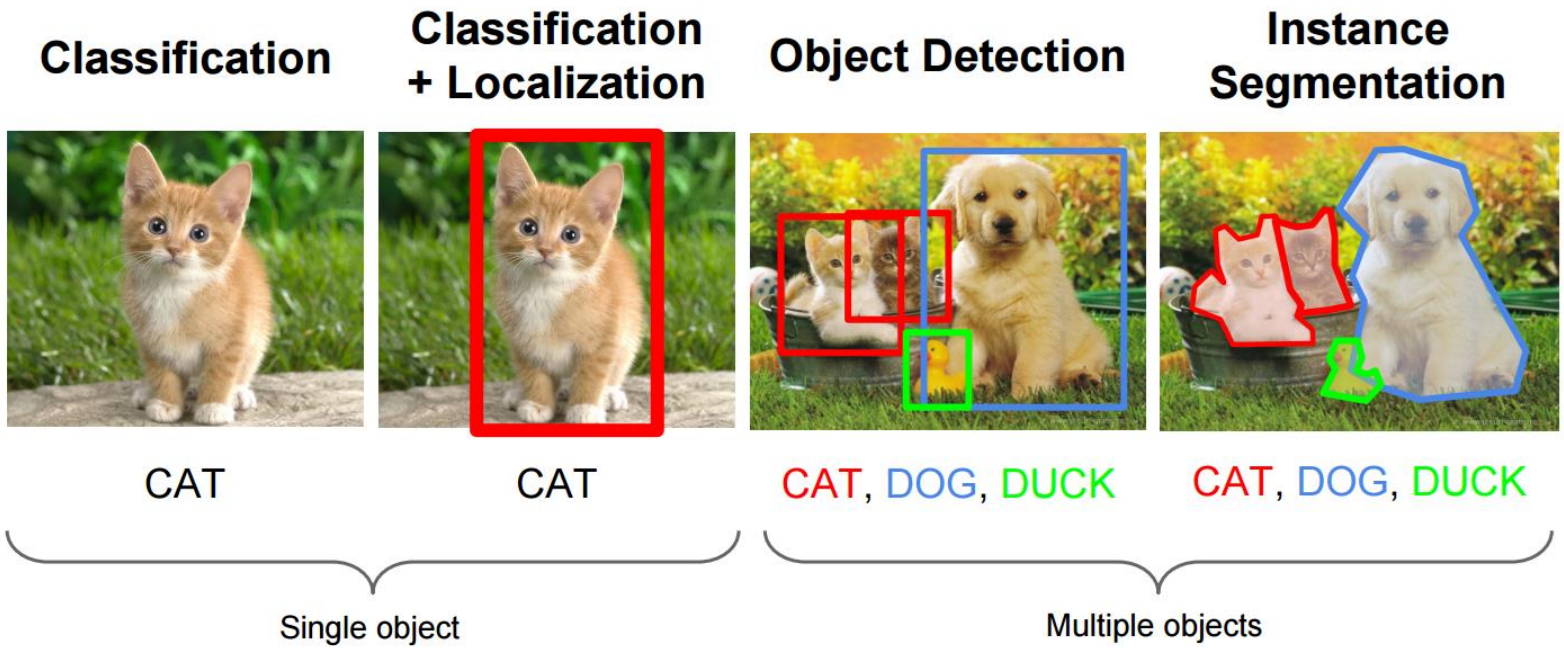
Object Detection, Semantic Segmentation & Instance Segmentation

Alex
Safety Sensing and Control Department
Mechanical and Mechatronics Systems Research Laboratories
Industrial Technology Research Institute

Overview

- Object Detection
- Segmentation

What is object detection



http://cs231n.stanford.edu/slides/winter1516_lecture8.pdf

Detection Competitions

Pascal VOC

COCO

KITTI

VOC: 20 classes

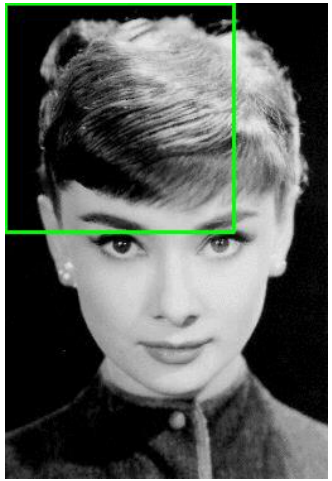
COCO: 200 classes

KITTI: 8 classes with only 3 evaluated

Before Deep Learning

Sliding windows.

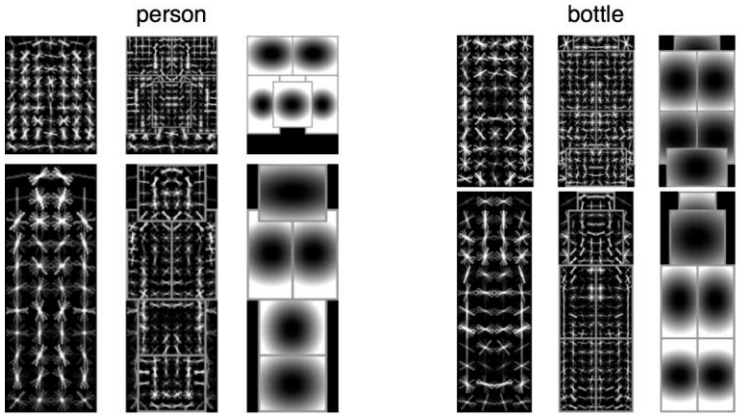
- Score every subwindow.



http://www.pyimagesearch.com/2014/11/10/histogram-oriented-gradients-object-detection/

Deformable part models (DPM)

- Uses HOG features
- Very fast



https://cs.brown.edu/~pff/papers/lsvm-pami.pdf

Selective Search

http://www.huppelen.nl/publications/selectiveSearchDraft.pdf

Two Stage Detection

RCNN : Region Proposal + CNN

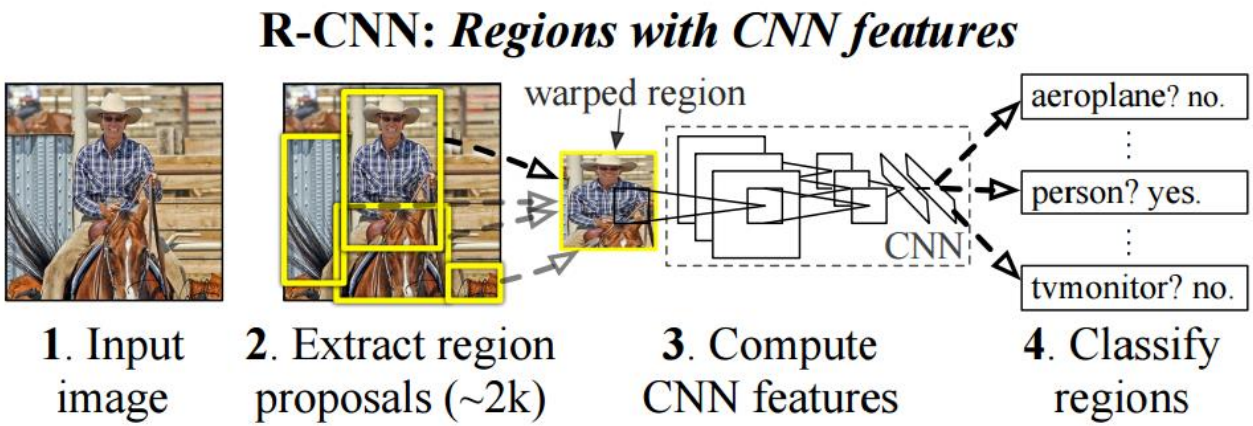
Use selective search to come up with regional proposal

First object detection method using CNN

Rich feature hierarchies for accurate object detection and semantic segmentation
Ross Girshick Jeff Donahue Trevor Darrell Jitendra Malik
Nov 2013

https://people.eecs.berkeley.edu/~rbg/papers/r-cnn-cvpr.pdf

RCNN :



https://people.eecs.berkeley.edu/~rbg/papers/r-cnn-cvpr.pdf

Training RCNN

Step1: train your own CNN model for classification (or use existing model), using ImageNet dataset.

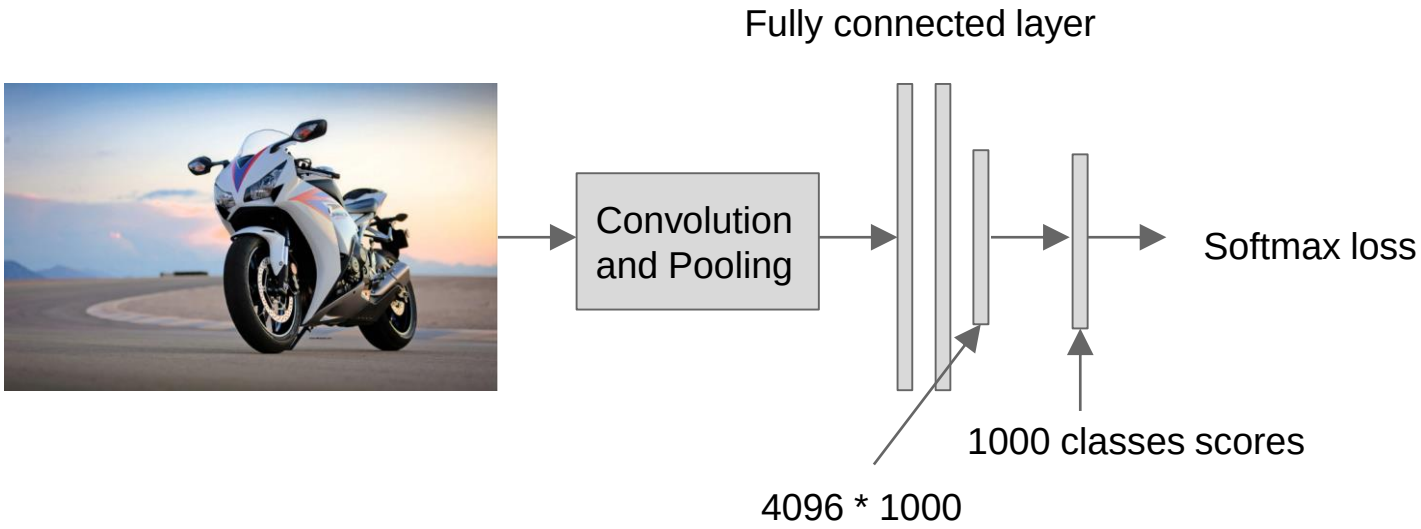
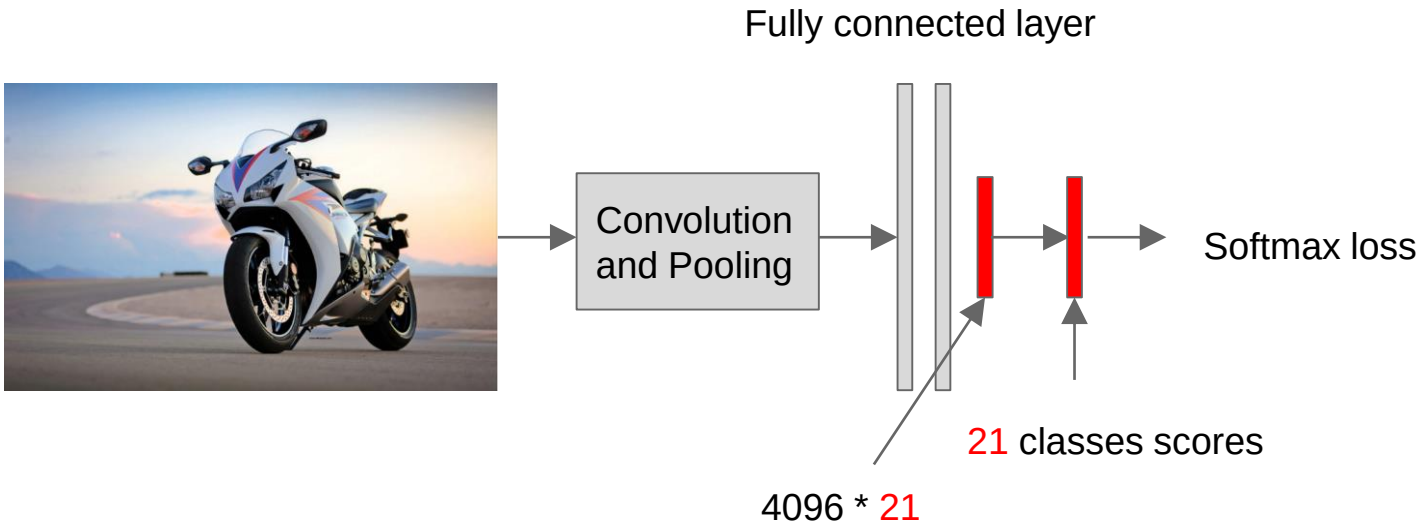


Image source : http://www.shunvmall.com/bike-pic/47539009.html

Training RCNN

Step2: focus on 20 classes + 1 background. Remove the last FC layer and replace it with a smaller layer and fine-tune the model using PASCAL VOC dataset



Training RCNN

Step3: extract feature

Crop & Warp

Convolution and Pooling

Store all the features after pool 5 layer and save to disk

It is about ~ 200G features

Training RCNN

Step4: train SVM for each class

Crop / Warp image

Features from last step

Positive samples for Motorbike

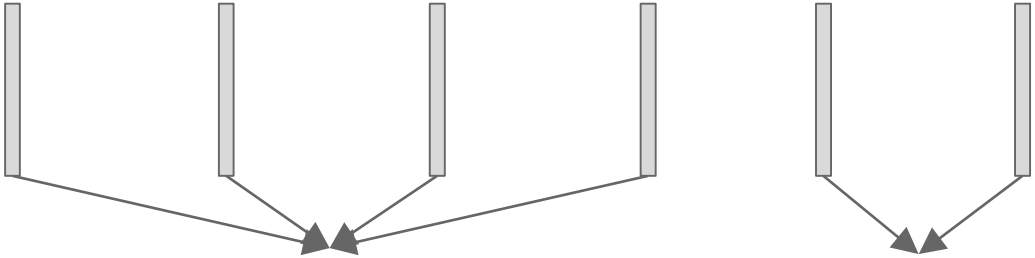
Negative samples for Motorbike

Training RCNN

Step4: train SVM for each class

Crop / Warp image

Features from last step



Negative samples for Bicycle

Positive samples for Bicycle

Ablation Studies

Without fine-tuning With fine-tuning

VOC 2007 test	aero	bike	bird	boat	bottle	bus	car	cat	chair	cow	table	dog	horse	mbike	person	plant	sheep	sofa	train	tv	mAP
R-CNN $pool_5$	51.8	60.2	36.4	27.8	23.2	52.8	60.6	49.2	18.3	47.8	44.3	40.8	56.6	58.7	42.4	23.4	46.1	36.7	51.3	55.7	44.2
R-CNN fc_6	59.3	61.8	43.1	34.0	25.1	53.1	60.6	52.8	21.7	47.8	42.7	47.8	52.5	58.5	44.6	25.6	48.3	34.0	53.1	58.0	46.2
R-CNN fc_7	57.6	57.9	38.5	31.8	23.7	51.2	58.9	51.4	20.0	50.5	40.9	46.0	51.6	55.9	43.3	23.3	48.1	35.3	51.0	57.4	44.7
R-CNN FT $pool_5$	58.2	63.3	37.9	27.6	26.1	54.1	66.9	51.4	26.7	55.5	43.4	43.1	57.7	59.0	45.8	28.1	50.8	40.6	53.1	56.4	47.3
R-CNN FT fc_6	63.5	66.0	47.9	37.7	29.9	62.5	70.2	60.2	32.0	57.9	47.0	53.5	60.1	64.2	52.2	31.3	55.0	50.0	57.7	63.0	53.1
R-CNN FT fc_7	64.2	69.7	50.0	41.9	32.0	62.6	71.0	60.7	32.7	58.5	46.5	56.1	60.6	66.8	54.2	31.5	52.8	48.9	57.9	64.7	54.2
R-CNN FT fc_7 BB	**68.1**	**72.8**	**56.8**	**43.0**	**36.8**	**66.3**	**74.2**	**67.6**	**34.4**	**63.5**	**54.5**	**61.2**	**69.1**	**68.6**	**58.7**	**33.4**	**62.9**	**51.1**	**62.5**	**64.8**	**58.5**
DPM v5 [18]	33.2	60.3	10.2	16.1	27.3	54.3	58.2	23.0	20.0	24.1	26.7	12.7	58.1	48.2	43.2	12.0	21.1	36.1	46.0	43.5	33.7
DPM ST [26]	23.8	58.2	10.5	8.5	27.1	50.4	52.0	7.3	19.2	22.8	18.1	8.0	55.9	44.8	32.4	13.3	15.9	22.8	46.2	44.9	29.1
DPM HSC [28]	32.2	58.3	11.5	16.3	30.6	49.9	54.8	23.5	21.5	27.7	34.0	13.7	58.1	51.6	39.9	12.4	23.5	34.4	47.4	45.2	34.3

Table 2: Detection average precision (%) on VOC 2007 test. Rows 1-3 show R-CNN performance without fine-tuning. Rows 4-6 show results for the CNN pre-trained on ILSVRC 2012 and then fine-tuned (FT) on VOC 2007 trainval. Row 7 includes a simple bounding box regression (BB) stage that reduces localization errors (Section 3.4). Rows 8-10 present DPM methods as a strong baseline. The first uses only HOG, while the next two use different feature learning approaches to augment or replace HOG.

Fast RCNN

Share convolution layers for proposals from the same image

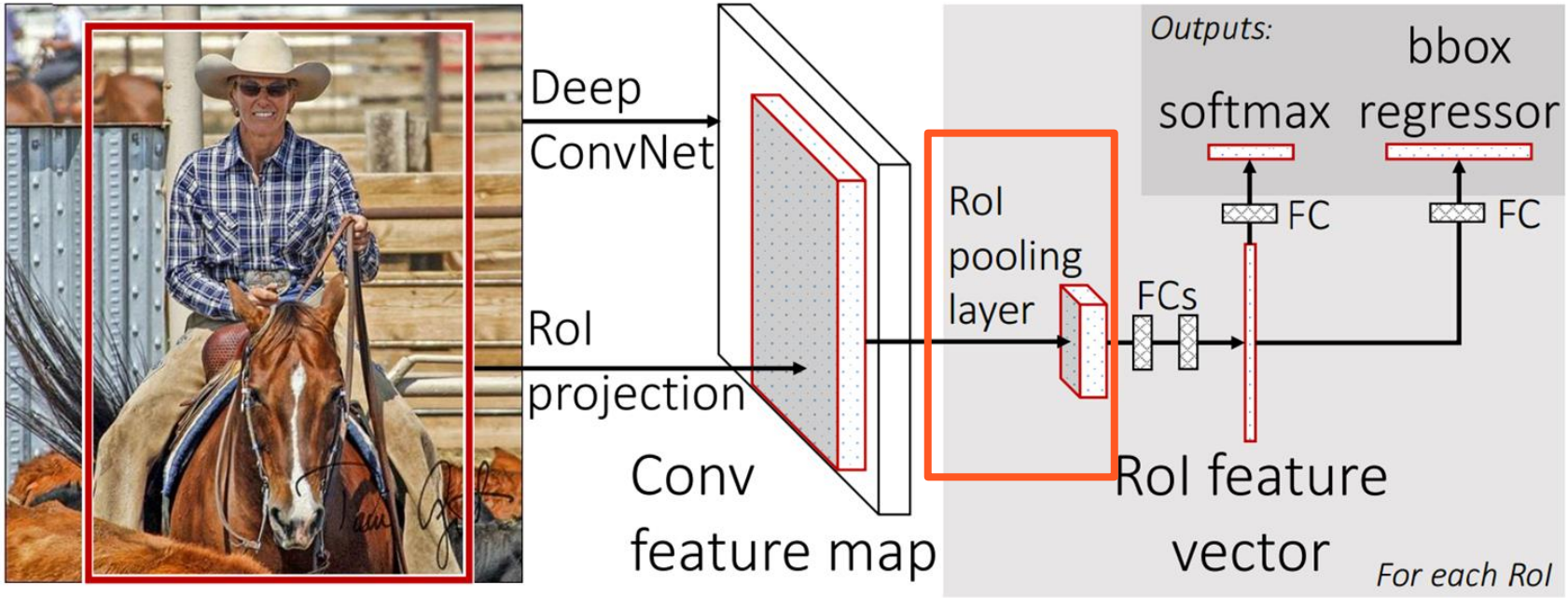
Faster and More accurate than RCNN

ROI Pooling

Fast R-CNN
Ross Girshick
Apr 2015

https://arxiv.org/pdf/1504.08083v2.pdf

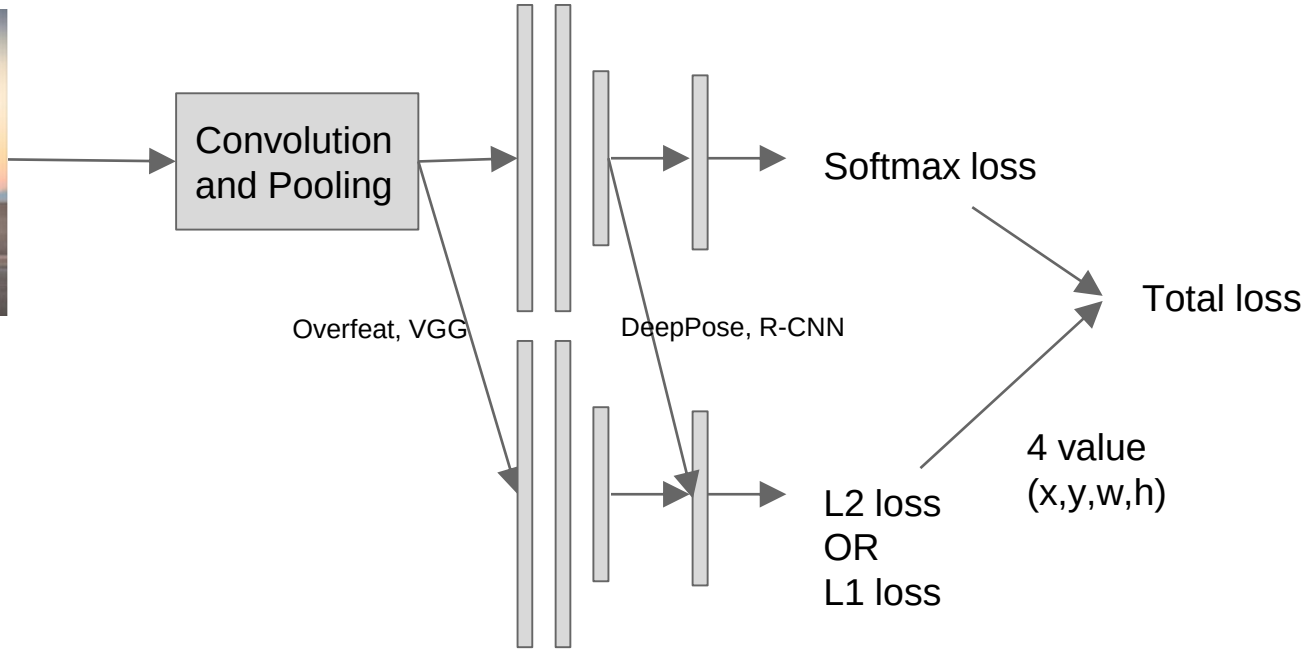
Fast RCNN



https://arxiv.org/pdf/1504.08083v2.pdf

What is bbox-regressor?

Bounding box regression



VOC 2007 results

- All methods start from the same pre-trained VGG16 and use bounding-box regression.
- Fine-tuning the conv layers is beneficial.
- Training without example marked as “difficult” improves mAP.

method	train set	aero	bike	bird	boat	bottle	bus	car	cat	chair	cow	table	dog	horse	mbike	persn	plant	sheep	sofa	train	tv	mAP
SPPnet BB [11]†	07 \ diff	73.9	72.3	62.5	51.5	44.4	74.4	73.0	74.4	42.3	73.6	57.7	70.3	74.6	74.3	54.2	34.0	56.4	56.4	67.9	73.5	63.1
R-CNN BB [10]	07	73.4	77.0	63.4	45.4	**44.6**	75.1	78.1	79.8	40.5	73.7	62.2	79.4	78.1	73.1	64.2	**35.6**	66.8	67.2	70.4	**71.1**	66.0
FRCN [ours]	07	74.5	78.3	69.2	53.2	36.6	77.3	78.2	82.0	40.7	72.7	67.9	79.6	79.2	73.0	69.0	30.1	65.4	70.2	75.8	65.8	66.9
FRCN [ours]	07 \ diff	74.6	**79.0**	68.6	57.0	39.3	79.5	**78.6**	81.9	**48.0**	74.0	67.4	80.5	80.7	74.1	69.6	31.8	67.1	68.4	75.3	65.5	68.1
FRCN [ours]	07+12	**77.0**	78.1	**69.3**	**59.4**	38.3	**81.6**	**78.6**	**86.7**	42.8	**78.8**	**68.9**	**84.7**	**82.0**	**76.6**	**69.9**	31.8	**70.1**	**74.8**	**80.4**	70.4	**70.0**

Table 1. **VOC 2007 test** detection average precision (%). All methods use VGG16. Training set key: **07**: VOC07 trainval, **07 \ diff**: **07** without “difficult” examples, **07+12**: union of **07** and VOC12 trainval. †SPPnet results were prepared by the authors of [11].

Truncated SVD for faster detection

- The learnt weightings in fully-connected layer could be represented by matrix and the forward pass could be interpreted as matrix multiplication.
- The original complexity is O(n^2) given an nxn matrix and an nx1 vector .
- However, by performing SVD on such matrix, the complexity could be reduced to O(n+n).
- Example:

$$W_{nxn} X_{nx1} \longrightarrow O(n^2)$$

$$W_{nxn} X_{nx1} = U_{nxn} D_{nxn} V^t_{nxn} X_{nx1} \longrightarrow O(n^2)$$

$$W_{nxn} X_{nx1} \approx U_{nx2} D_{2x2} (V^t)_{2xn} X_{nx1} \longrightarrow O(n)$$

Training and testing time

- S: Alexnet
- M:VGG_CNN_M-1024 FROM [3]
- L:VGG 16 from [20]

	Fast R-CNN			R-CNN			SPPnet
	S	**M**	**L**	**S**	**M**	**L**	†**L**
train time (h)	**1.2**	2.0	9.5	22	28	84	25
train speedup	**18.3**×	14.0×	8.8×	1×	1×	1×	3.4×
test rate (s/im)	0.10	0.15	0.32	9.8	12.1	47.0	2.3
▷ with SVD	**0.06**	0.08	0.22	-	-	-	-
test speedup	98×	80×	146×	1×	1×	1×	20×
▷ with SVD	169×	150×	**213**×	-	-	-	-
VOC07 mAP	57.1	59.2	**66.9**	58.5	60.2	66.0	63.1
▷ with SVD	56.5	58.7	66.6	-	-	-	-

Table 4. Runtime comparison between the same models in Fast R-CNN, R-CNN, and SPPnet. Fast R-CNN uses single-scale mode. SPPnet uses the five scales specified in [11]. †Timing provided by the authors of [11]. Times were measured on an Nvidia K40 GPU.

Truncated SVD

- Truncated SVD can reduce detection time by more than 30% with only a small drop in mAP.

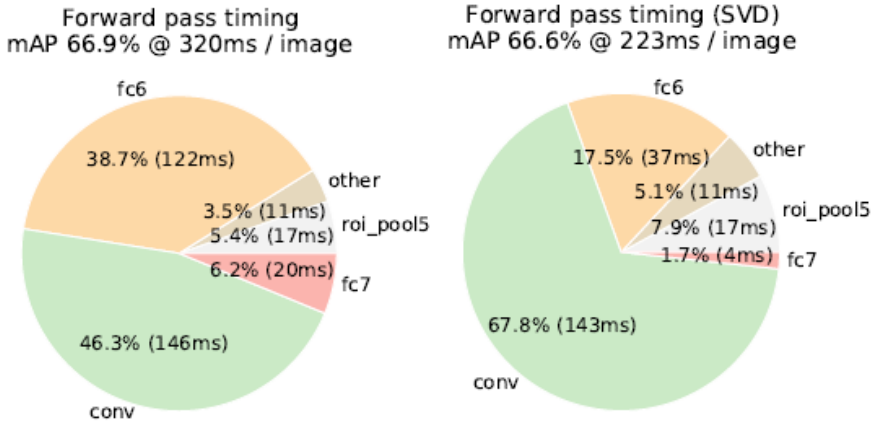


Figure 2. Timing for VGG16 before and after truncated SVD. Before SVD, fully connected layers fc6 and fc7 take 45% of the time.

Which layers to fine-tune?

- For the less deep networks considered in the SPPnet paper[11], fine-tuning only the fully connected layers appeared to be sufficient for good accuracy.
- This paper hypothesized that this result would not hold for very deep networks.
- The experiment verified our hypothesis: training through the RoI pooling layer is important for very deep nets.
- Does this mean all conv layers should be fine-tuned? No, in the smaller networks, this paper found that conv1 is generic and task independent(a well-known fact [14])
- All fast R-CNN results in this paper using VGG16 fine-tune layers conv3_1 and up; all experiments with models S and M fine-tune layers con2 and up.

	layers that are fine-tuned in model **L**			SPPnet **L**
	$\geq$ fc6	$\geq$ conv3_1	$\geq$ conv2_1	$\geq$ fc6
VOC07 mAP	61.4	66.9	**67.2**	63.1
test rate (s/im)	**0.32**	**0.32**	**0.32**	2.3

Does multi-task training help?

- Column 1 in each model: only training with L_{cls}
- Column 2 in each model: training with L_{cls}& L_{loc} with λ=1 (multi-task training without using bbox regression)
- Column 3 in each model: sequentially training with L_{cls} then L_{loc}+ bbox reg
- Column 4 in each model: multi-task training with bbox reg.

	S				M				L			
multi-task training?		✓		✓		✓		✓		✓		✓
stage-wise training?			✓				✓				✓	
test-time bbox reg?			✓	✓			✓	✓			✓	✓
VOC07 mAP	52.2	53.3	54.6	**57.1**	54.7	55.5	56.6	**59.2**	62.6	63.4	64.0	**66.9**

Table 6. Multi-task training (forth column per group) improves mAP over piecewise training (third column per group).

Scale invariance: to brute force or finesse

- This paper compared two strategies for achieving scale-invariant object detection:
  - **brute-force learning (single scale):600 pixels for shortest side**
  - **image pyramids (multi-scale): 5 scales (480, 576, 688, 864, 1200) specified in [11]**
- Observation:
  - **deep ConvNets are adept at directly learning scale invariance**
  - **In model-L, it has only single scale due to memory limitation but outperforms the result (66.0%) reported from R-CNN even though R-CNN uses "infinite" scales.**

	SPPnet **ZF**		**S**		**M**		**L**
scales	1	5	1	5	1	5	1
test rate (s/im)	0.14	0.38	**0.10**	0.39	0.15	0.64	0.32
VOC07 mAP	58.0	59.2	57.1	58.4	59.2	60.7	**66.9**

Table 7. Multi-scale vs. single scale. SPPnet **ZF** (similar to model **S**) results are from [11]. Larger networks with a single-scale offer the best speed / accuracy tradeoff. (**L** cannot use multi-scale in our implementation due to GPU memory constraints.)

Are more proposals always better?

- Observation:
  - **mAP rises and then falls slightly as the proposal count increases. (solid blue line)**
  - **Higher AR(Solid red line) doesn't imply that mAP will increase because FP will also increase**
  - **Using densely generated boxes (over scale, position and aspect ratio) at a rate of 45k boxes/image and replacing each selective search box with the closest dense box will reduce 1% (blue triangle)=>SS is better**
  - **Starting with 2k selective search boxes and adding random 1000*{2,4,6,8,10,32,45} dense box are not helpful. The mAP continues dropping.(dotted blue line)**
  - **Using only 45K dense boxes, the mAP is 52.9(blue diamond)**
  - **SVM with hard negative mining given 45k dense boxes is not helpful.(blue circle)**

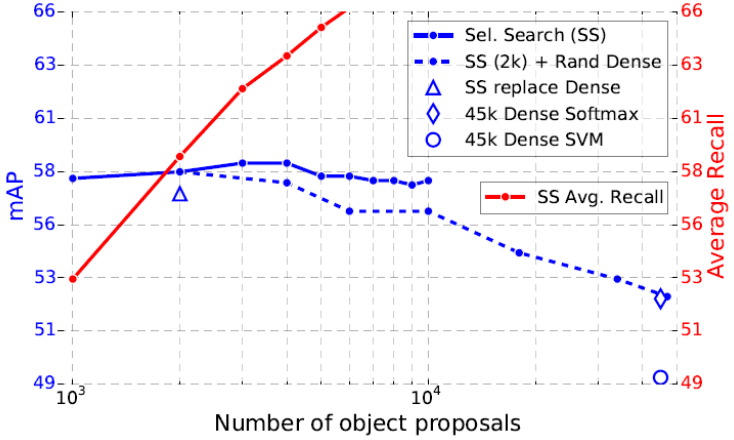


Figure 3. VOC07 test mAP and AR for various proposal schemes.

Conclusion of Fast R-CNN

- This paper proposes Fast R-CNN, a clean and fast update to R-CNN and SPPnet with advantages of
  - higher mAP then R-CNN & SPPnet
  - Single-stage training using multi-task loss
  - Updating all networks in training
  - No disk storage is required for feature caching
- A lot of new insights are given for achieving high mAP.
- Sparse object proposals appear to improve detector quality; however, undiscovered dense boxes might be beneficial.

Faster RCNN

Don’t need to have external regional proposals

RPN - Regional Proposal Network

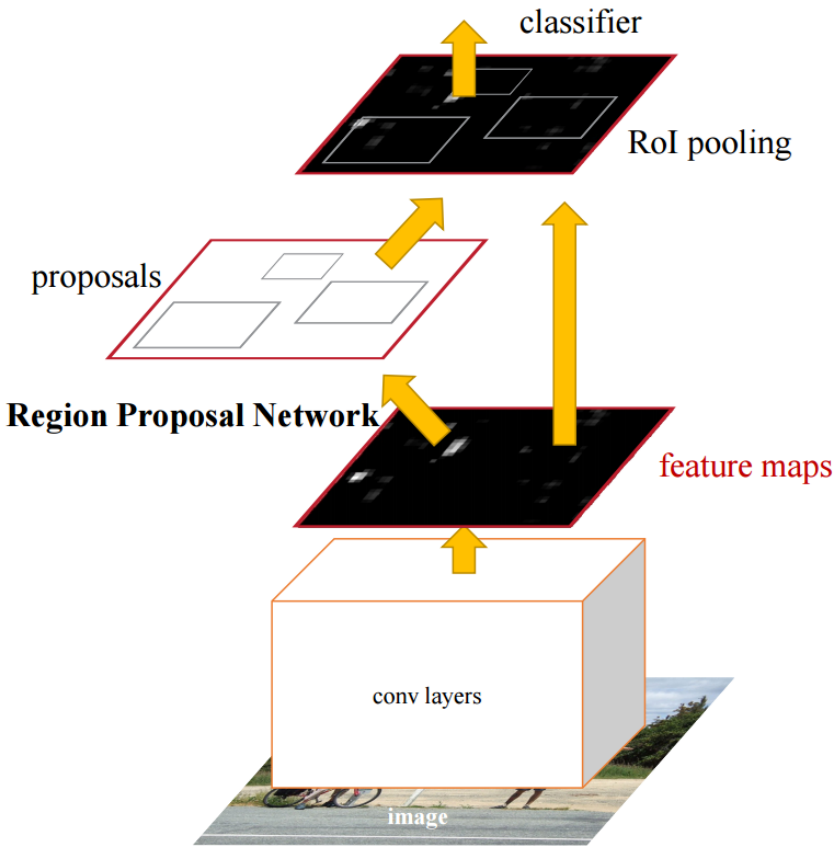
Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks

Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun

Jun 2015

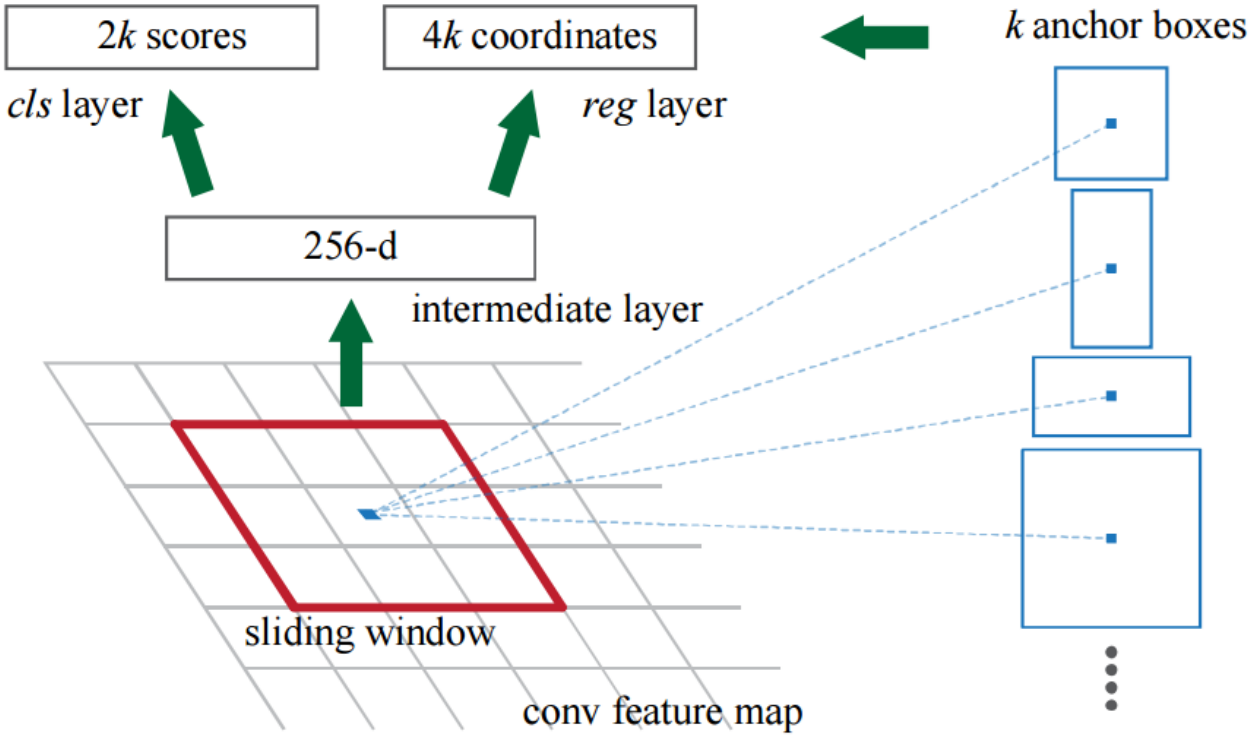
https://arxiv.org/pdf/1506.01497v3.pdf

Faster RCNN



https://arxiv.org/pdf/1506.01497v3.pdf

Faster RCNN



https://arxiv.org/pdf/1506.01497v3.pdf

Experiments on PASCAL VOC 07

Performance of VGG-16

Table 3 shows the results of VGG-16 for both proposal and detection. Using RPN+VGG, the result is 68.5% for *unshared* features, slightly higher than the SS baseline (56.8%). For the *feature-shared* variant, the result is 69.9%—better than the strong SS baseline, yet with nearly cost-free proposals.

Table 3: Detection results on **PASCAL VOC 2007 test set**. The detector is Fast R-CNN and VGG-16. Training data: "07": VOC 2007 trainval, "07+12": union set of VOC 2007 trainval and VOC 2012 trainval. For RPN, the train-time proposals for Fast R-CNN are 2000. †: this number was reported in [2]; using the repository provided by this paper, this result is higher (68.1).

method	# proposals	data	mAP (%)
SS	2000	07	66.9†
SS	2000	07+12	70.0
RPN+VGG, unshared	300	07	68.5
RPN+VGG, shared	300	07	69.9
RPN+VGG, shared	300	07+12	**73.2**
RPN+VGG, shared	300	COCO+07+12	**78.8**

Experiments on PASCAL VOC

Performance of VGG-16

Table 6: Results on PASCAL VOC 2007 test set with Fast R-CNN detectors and VGG-16. For RPN, the train-time proposals for Fast R-CNN are 2000. RPN* denotes the unsharing feature version.

method	# box	data	mAP	areo	bike	bird	boat	bottle	bus	car	cat	chair	cow	table	dog	horse	mbike	person	plant	sheep	sofa	train	tv
SS	2000	07	66.9	74.5	78.3	69.2	53.2	36.6	77.3	78.2	82.0	40.7	72.7	67.9	79.6	79.2	73.0	69.0	30.1	65.4	70.2	75.8	65.8
SS	2000	07+12	70.0	77.0	78.1	69.3	59.4	38.3	81.6	78.6	86.7	42.8	78.8	68.9	84.7	82.0	76.6	69.9	31.8	70.1	74.8	80.4	70.4
RPN*	300	07	68.5	74.1	77.2	67.7	53.9	51.0	75.1	79.2	78.9	50.7	78.0	61.1	79.1	81.9	72.2	75.9	37.2	71.4	62.5	77.4	66.4
RPN	300	07	69.9	70.0	80.6	70.1	57.3	49.9	78.2	80.4	82.0	52.2	75.3	67.2	80.3	79.8	75.0	76.3	39.1	68.3	67.3	81.1	67.6
RPN	300	07+12	73.2	76.5	79.0	70.9	65.5	52.1	83.1	84.7	86.4	52.0	81.9	65.7	84.8	84.6	77.5	76.7	38.8	73.6	73.9	83.0	72.6
RPN	300	COCO+07+12	**78.8**	**84.3**	**82.0**	**77.7**	**68.9**	**65.7**	**88.1**	**88.4**	**88.9**	**63.6**	**86.3**	**70.8**	**85.9**	**87.6**	**80.1**	**82.3**	**53.6**	**80.4**	**75.8**	**86.6**	**78.9**

Table 5: Timing (ms) on a K40 GPU, except SS proposal is evaluated in a CPU. "Region-wise" includes NMS, pooling, fully-connected, and softmax layers. See our released code for the profiling of running time.

model	system	conv	proposal	region-wise	total	rate
VGG	SS + Fast R-CNN	146	1510	174	1830	0.5 fps
VGG	RPN + Fast R-CNN	141	10	47	198	5 fps
ZF	RPN + Fast R-CNN	31	3	25	59	17 fps

Conclusions

- This paper presents RPNs for efficient and accurate region proposal generation.
- By sharing convolutional features with the down-stream detection network, the region proposal step is nearly cost-free.
- Our method enables a unified, deep-learning-based object detection system to run at near real-time frame rates.
- The learned RPN also improves region proposal quality and thus the overall object detection accuracy.

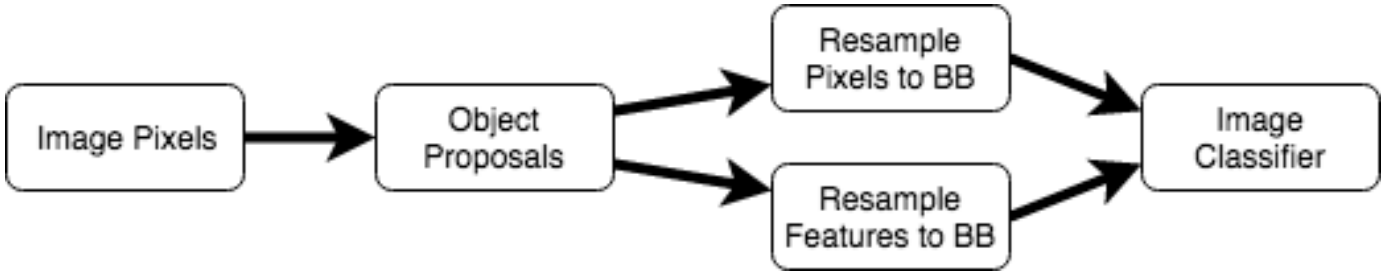
One Stage Detection

Problems with 2 step detection.

Complex Pipeline

Slow (Cannot run in real time)

Hard to optimize each component



YOLO: You Only Look Once

Consider detection a regression problem

Use a single ConvNet

Runs once on entire image. Very Fast!

You only look once: Unified, real-time object detection.
Joseph Redmon, Santosh Divvala, Ross Girshick, Ali Farhadi
June 2015

How it works

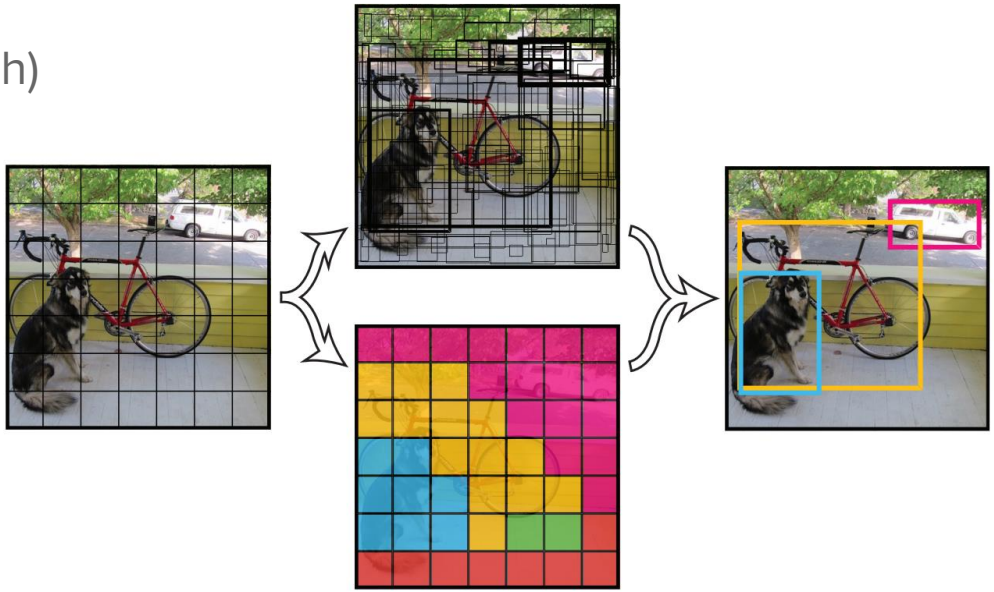
The following predictions are made for each cell in an S x S grid.

C conditional class probabilities **Pr(Class$_i$ | Obj)**

B bounding boxes (4 parameters each)

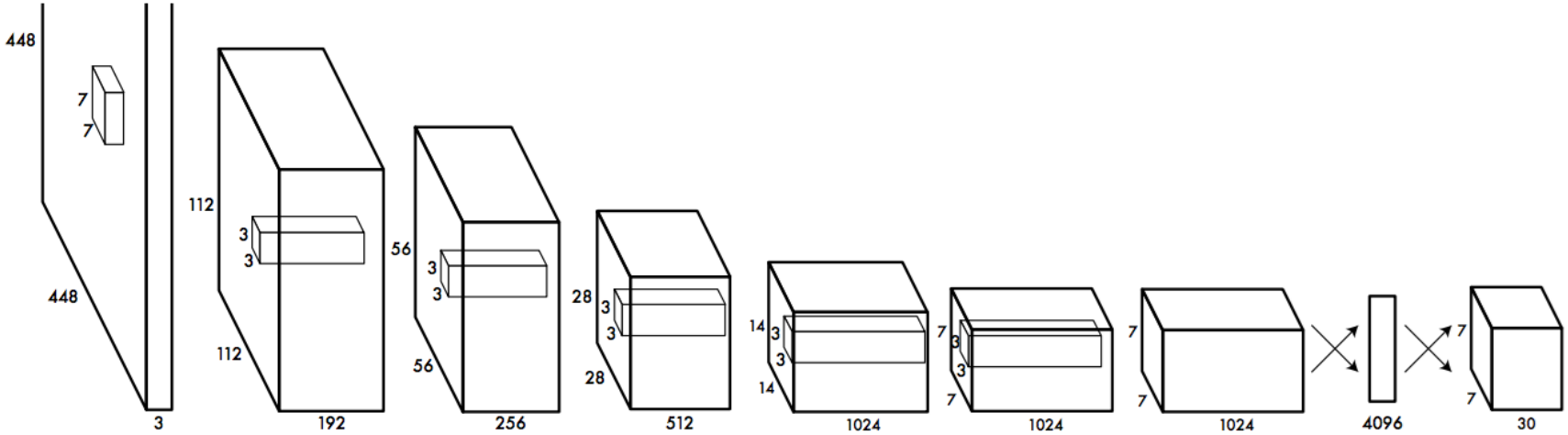
B confidence scores **Pr(Obj)*IoU**

Output is **S x S x (5B+C)** tensor



https://arxiv.org/pdf/1506.02640v5.pdf

Architecture



Conv. Layer
7x7x64-s-2
Maxpool Layer
2x2-s-2

Conv. Layer
3x3x192
Maxpool Layer
2x2-s-2

Conv. Layers
1x1x128
3x3x256
1x1x256
3x3x512
Maxpool Layer
2x2-s-2

Conv. Layers
1x1x256
3x3x512
×4
1x1x512
3x3x1024
Maxpool Layer
2x2-s-2

Conv. Layers
1x1x512
3x3x1024
×2
3x3x1024
3x3x1024-s-2

Conv. Layers
3x3x1024
3x3x1024

Conn. Layer

Conn. Layer

https://arxiv.org/pdf/1506.02640v5.pdf

Loss Function

for those bounding boxes containing objects

$$\boxed{\lambda_{\text{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left(x_i - \hat{x}_i\right)^2 + \left(y_i - \hat{y}_i\right)^2}$$

$$\boxed{+ \lambda_{\text{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left(\sqrt{w_i} - \sqrt{\hat{w}_i}\right)^2 + \left(\sqrt{h_i} - \sqrt{\hat{h}_i}\right)^2}$$

$$\boxed{+ \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left(C_i - \hat{C}_i\right)^2}$$

Pr(Object)

$$+ \lambda_{\text{noobj}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{noobj}} \left(C_i - \hat{C}_i\right)^2$$

$$\boxed{+ \sum_{i=0}^{S^2} \mathbb{1}_{i}^{\text{obj}} \sum_{c \in \text{classes}} \left(p_i(c) - \hat{p}_i(c)\right)^2} \quad (3)$$

Pr(Class$_i$|Object)

Loss Function

for those bounding boxes not containing objects

$$\lambda_{\text{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left(x_i - \hat{x}_i\right)^2 + \left(y_i - \hat{y}_i\right)^2$$

$$+ \lambda_{\text{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left(\sqrt{w_i} - \sqrt{\hat{w}_i}\right)^2 + \left(\sqrt{h_i} - \sqrt{\hat{h}_i}\right)^2$$

$$+ \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left(C_i - \hat{C}_i\right)^2$$

$$+ \boxed{\lambda_{\text{noobj}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{noobj}} \left(C_i - \hat{C}_i\right)^2}$$

Pr(Object)

$$+ \sum_{i=0}^{S^2} \mathbb{1}_{i}^{\text{obj}} \sum_{c \in \text{classes}} \left(p_i(c) - \hat{p}_i(c)\right)^2 \qquad (3)$$

Experimental Results

Draft paper

Real-Time Detectors	Train	mAP	FPS
100Hz DPM [30]	2007	16.0	100
30Hz DPM [30]	2007	26.1	30
Fast YOLO	2007+2012	52.7	**155**
YOLO	2007+2012	**63.4**	45
Less Than Real-Time			
Fastest DPM [37]	2007	30.4	15
R-CNN Minus R [20]	2007	53.5	6
Fast R-CNN [14]	2007+2012	70.0	0.5
Faster R-CNN VGG-16[27]	2007+2012	73.2	7
Faster R-CNN ZF [27]	2007+2012	62.1	18

Table 1: Real-Time Systems on PASCAL VOC 2007. Comparing the performance and speed of fast detectors. Fast YOLO is the fastest detector on record for PASCAL VOC detection and is still twice as accurate as any other real-time detector. YOLO is 10 mAP more accurate than the fast version while still well above real-time in speed.

Latest Version

Real-Time Detectors	Train	mAP	FPS
100Hz DPM [31]	2007	16.0	100
30Hz DPM [31]	2007	26.1	30
Fast YOLO	2007+2012	52.7	**155**
YOLO	2007+2012	**63.4**	45
Less Than Real-Time			
Fastest DPM [38]	2007	30.4	15
R-CNN Minus R [20]	2007	53.5	6
Fast R-CNN [14]	2007+2012	70.0	0.5
Faster R-CNN VGG-16[28]	2007+2012	73.2	7
Faster R-CNN ZF [28]	2007+2012	62.1	18
YOLO VGG-16	2007+2012	66.4	21

Table 1: Real-Time Systems on PASCAL VOC 2007. Comparing the performance and speed of fast detectors. Fast YOLO is the fastest detector on record for PASCAL VOC detection and is still twice as accurate as any other real-time detector. YOLO is 10 mAP more accurate than the fast version while still well above real-time in speed.

VOC 2007 Error Analysis

- Correct: correct class and IOU > .5
- Localization: correct class, .1 < IOU < .5
- Similar: class is similar, IOU > .1
- Other: class is wrong, IOU > .1
- Background: IOU < .1 for any object

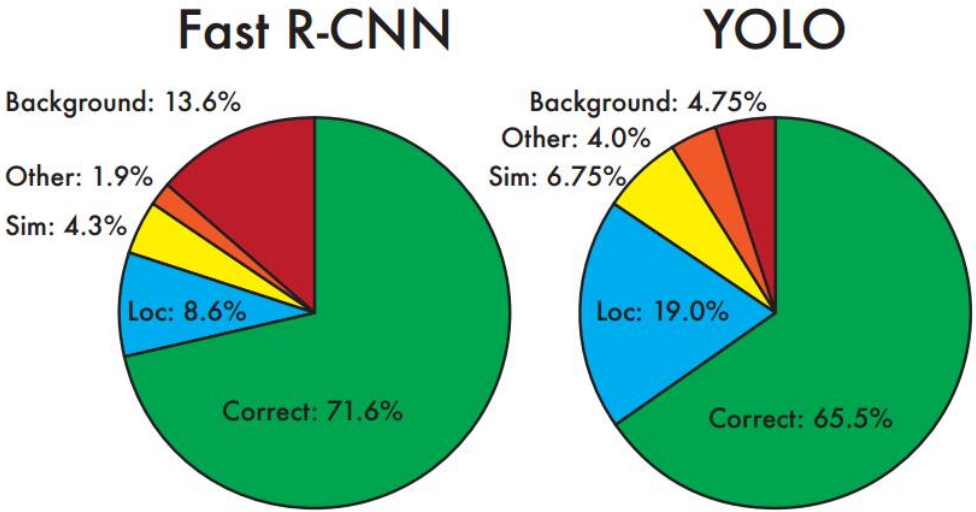


Figure 4: Error Analysis: Fast R-CNN vs. YOLO These charts show the percentage of localization and background errors in the top N detections for various categories (N = # objects in that category).

Combining Fast R-CNN and YOLO

	mAP	Combined	Gain
Fast R-CNN	-	71.8	-
Fast R-CNN (2007 data)	**66.9**	72.4	.6
Fast R-CNN (VGG-M)	59.2	72.4	.6
Fast R-CNN (CaffeNet)	57.1	72.1	.3
YOLO	63.4	**75.0**	**3.2**

Table 2: Model combination experiments on VOC 2007. We examine the effect of combining various models with the best version of Fast R-CNN. Other versions of Fast R-CNN provide only a small benefit while YOLO provides a significant performance boost.

Limitations of YOLO

Struggles with small objects

Struggles with unusual aspect ratios

Poor localization

SSD Single Shot Detector

Faster than Yolo, as accurate as Faster R-CNN

Predicts categories and box offsets

Uses small convolutional filters applied to feature maps

Makes predictions using feature maps of different scales

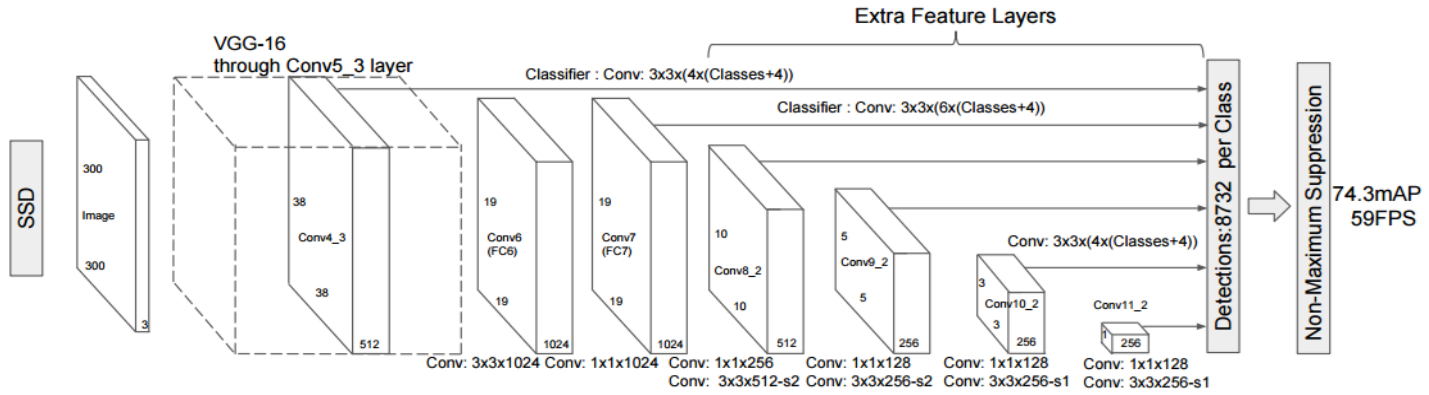
SSD: Single shot multibox detector

Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, Alexander C. Berg

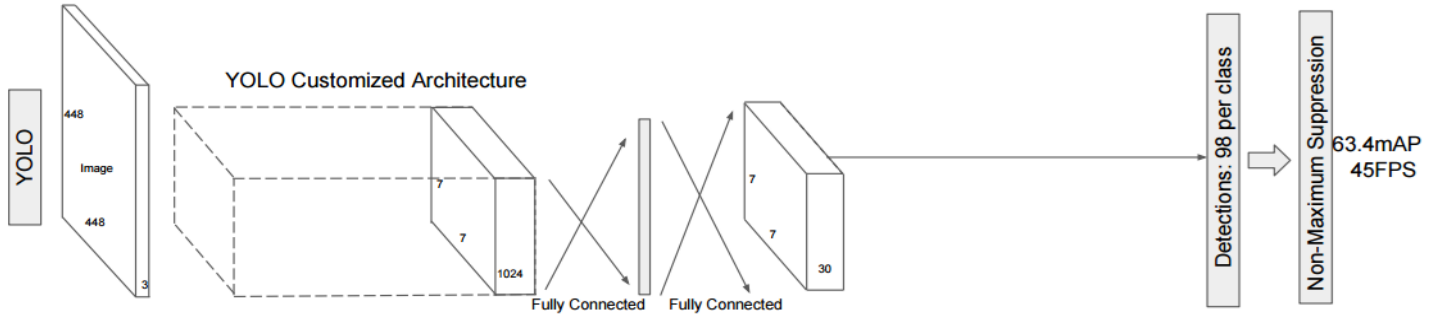
Dec 2015

Comparison to YOLO

SSD



YOLO



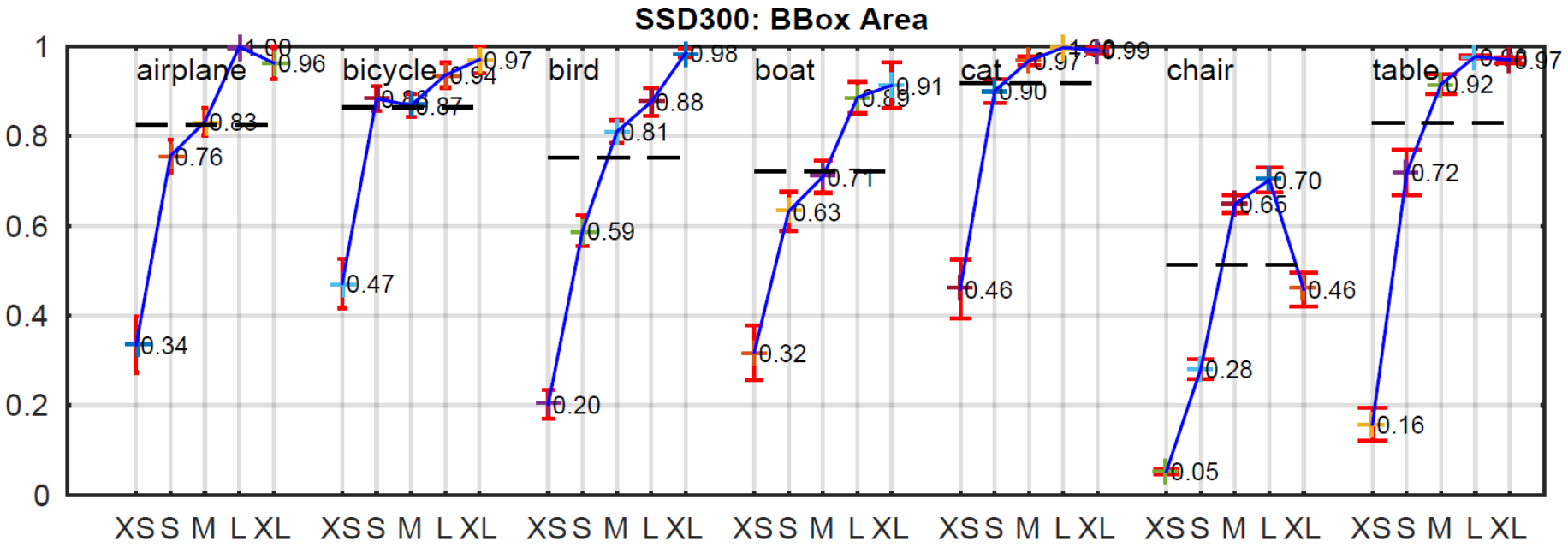
https://arxiv.org/pdf/1512.02325v5.pdf

Experimental Results: PASCAL 2007

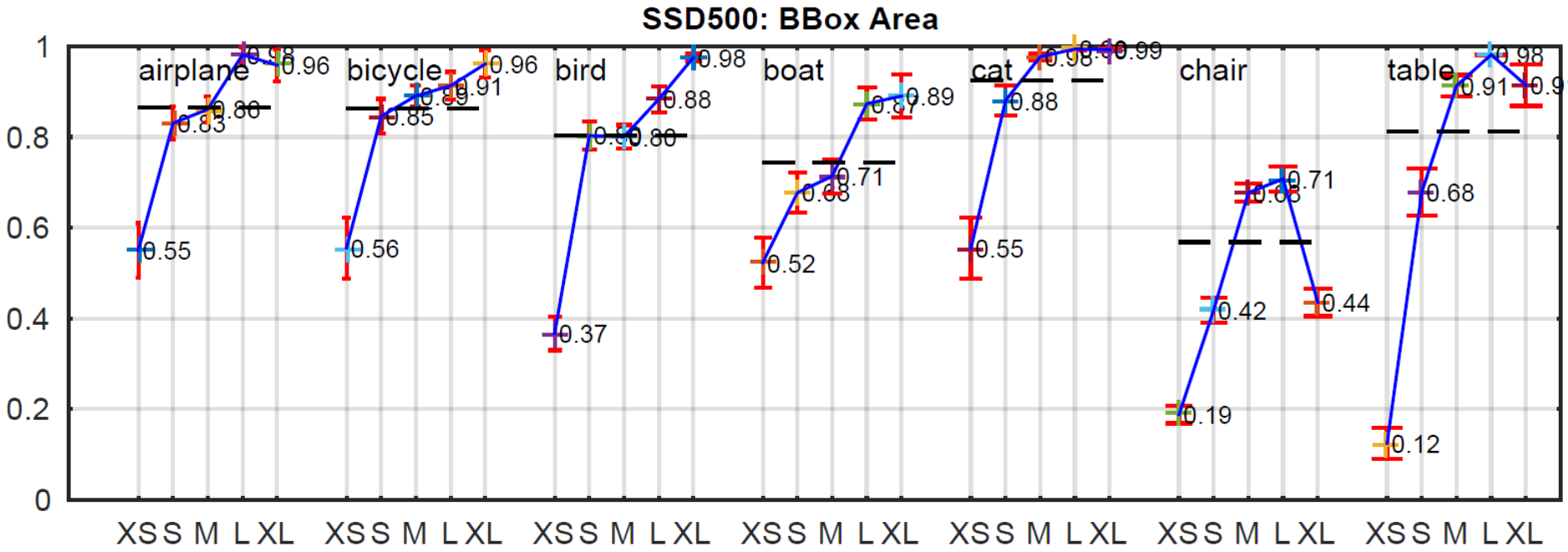
Method	mAP	aero	bike	bird	boat	bottle	bus	car	cat	chair	cow	table	dog	horse	mbike	person	plant	sheep	sofa	train	tv
Fast [6]	70.0	77.0	78.1	69.3	59.4	38.3	81.6	78.6	86.7	42.8	78.8	68.9	84.7	82.0	76.6	69.9	31.8	70.1	**74.8**	80.4	70.4
Faster [2]	73.2	76.5	79.0	70.9	**65.5**	**52.1**	83.1	84.7	86.4	52.0	**81.9**	65.7	84.8	84.6	77.5	76.7	38.8	73.6	73.9	83.0	72.6
SSD300	72.1	75.2	**79.8**	70.5	62.5	41.3	81.1	80.8	86.4	51.5	74.3	**72.3**	83.5	84.6	80.6	74.5	46.0	71.4	73.8	83.0	69.1
SSD500	**75.1**	**79.8**	79.5	**74.5**	63.4	51.9	**84.9**	**85.6**	**87.2**	**56.6**	80.1	70.0	**85.4**	**84.9**	**80.9**	**78.2**	**49.0**	**78.4**	72.4	**84.6**	**75.5**

Table 1: **PASCAL VOC2007 `test` detection results.** Both Fast and Faster R-CNN use input images whose minimum dimension is 600. The two SSD models have exactly the same settings except that they have different input sizes (300×300 vs. 500×500). It is obvious that larger input size leads to better results.

Sensitivity and impact of different object characteristics on VOC 2007 test set: SSD300 BBox Area



Sensitivity and impact of different object characteristics on VOC 2007 test set: SSD500 BBox Area



YOLOv2 from YOLO9000: Better, Faster, Stronger

- Better
  - Batch normalization
  - High resolution classifier
  - Convolution with anchor boxes
  - Dimension clusters
  - Direct location prediction
  - Fine-grained features
  - Multi-scale training
- Faster
  - Darknet-19
  - Training for classification
  - Training for detection
- Stronger
  - Hierarchical classification
  - Dataset combination with Word-tree
  - Joint classification and detection

YOLOv2

YOLO9000

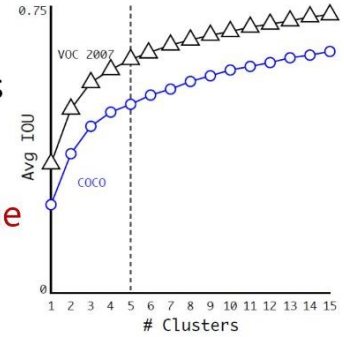
Better

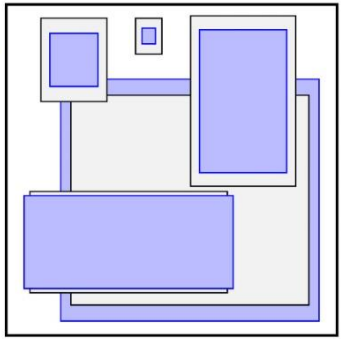
- Batch Normalization
  - Adding BN on all of conv layers in YOLO – 2% improvement in mAP
  - Removing dropout without overfitting

- High Resolution Classifier
  - YOLO trains the classifier network @224x224 and increase resolution to 448 for detection → The network has to simultaneously switch to learning object detection and adjust to the new input resolution
  - Fine tuning the classifier network @448x448 resolution for 10 epochs on ImageNet – almost 4% improvement in mAP

Better

- Dimension Clusters
  - The problem with anchor boxes is that they are hand-picked
  - Running k-means clustering on the training set b-boxes to find good priors
  - To prevent larger boxes from making more error, they use a different distance metric(not Euclidian distance)
    - d(box, centroid) = 1 – IOU(box, centroid)
  - k=5 is chosen as a good tradeoff between complexity and high recall





Box Generation	#	Avg IOU
Cluster SSE	5	58.7
Cluster IOU	5	61.0
Anchor Boxes [15]	9	60.9
Cluster IOU	9	67.2

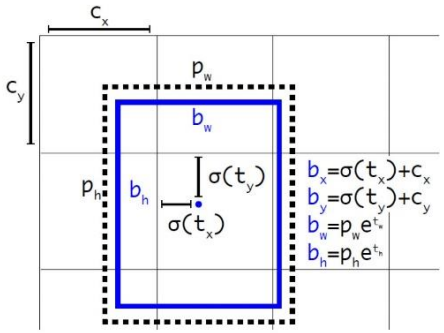
Table 1: Average IOU of boxes to closest priors on VOC 2007.

Better

- Direct Location Prediction
  - Another problem with anchor boxes is instability, in RPNs the anchor box can be anywhere in the image, regardless of what location predicted the box
  - Instead of predicting offsets, YOLOv2 predicts locations relative to the location of the grid cells
  - 5 bounding boxes for each cell, and 5 values for each bounding box → almost 5% improvement over the version with anchor boxes

RCNN's b-box regression

$$\hat{G}_x = P_w d_x(P) + P_x$$
$$\hat{G}_y = P_h d_y(P) + P_y$$
$$\hat{G}_w = P_w \exp(d_w(P))$$
$$\hat{G}_h = P_h \exp(d_h(P)).$$

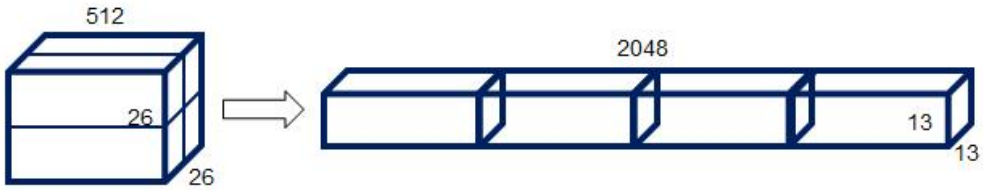


Different equations – direct prediction

$$b_x = \sigma(t_x) + c_x$$
$$b_y = \sigma(t_y) + c_y$$
$$b_w = p_w e^{t_w}$$
$$b_h = p_h e^{t_h}$$

$$Pr(\text{object}) * IOU(b, \text{object}) = \sigma(t_o)$$

Better



- Fine-Grained Features
  - Modified YOLO uses 13x13 feature maps to predict detections
    - Good at large object, finer grained features are required for small objects
  - Simply adding a passthrough layer that brings features from an earlier layer at 26x26 resolution
  - Turn 26x26x512 feature map into a 13x13x2048 feature map which can be concatenated with original features
  - Detector runs on top of this expanded features
  - 1% performance increase

Faster

- Darknet-19
  - mostly 3x3 filters (similar to VGG)
  - Following the work on Network in Network (NIN), use global average pooling to make predictions as well as 1x1 filters to compress the feature representation between 3x3 convolutions
  - Darknet-19: 19 convolutional layers & 5 max-pooling layers
  - 5.58 billion operations: 72.9% top-1 & 91.2% top-5 accuracy (30.69 billion operations & 90.0% top-5 accuracy for VGG-16)

Type	Filters	Size/Stride	Output
Convolutional	32	3×3	224×224
Maxpool		$2 \times 2/2$	112×112
Convolutional	64	3×3	112×112
Maxpool		$2 \times 2/2$	56×56
Convolutional	128	3×3	56×56
Convolutional	64	1×1	56×56
Convolutional	128	3×3	56×56
Maxpool		$2 \times 2/2$	28×28
Convolutional	256	3×3	28×28
Convolutional	128	1×1	28×28
Convolutional	256	3×3	28×28
Maxpool		$2 \times 2/2$	14×14
Convolutional	512	3×3	14×14
Convolutional	256	1×1	14×14
Convolutional	512	3×3	14×14
Convolutional	256	1×1	14×14
Convolutional	512	3×3	14×14
Maxpool		$2 \times 2/2$	7×7
Convolutional	1024	3×3	7×7
Convolutional	512	1×1	7×7
Convolutional	1024	3×3	7×7
Convolutional	512	1×1	7×7
Convolutional	1024	3×3	7×7
Convolutional	1000	1×1	7×7
Avgpool		Global	1000
Softmax			

Faster

- Training for classification
  - ImageNet 1000 classes for 160 epochs
  - Standard data augmentation: random crops, rotations, hue, saturation, and exposure shifts
  - Initial training : 224x224 → 448x448 fine-tuning for 10 epochs
    - Higher resolution achieves a top-5 accuracy of 93.3%
- Training for detection
  - Adding 3x3 conv layers with 1024 filters each followed by a final 1x1 conv layer
  - For VOC, predicting 5 boxes with 5 coordinates each and 20 classes per box, so 125 filters
  - 160 epochs with a start learning rate of 10^{-3} dividing it by 10 at 60 and 90 epochs

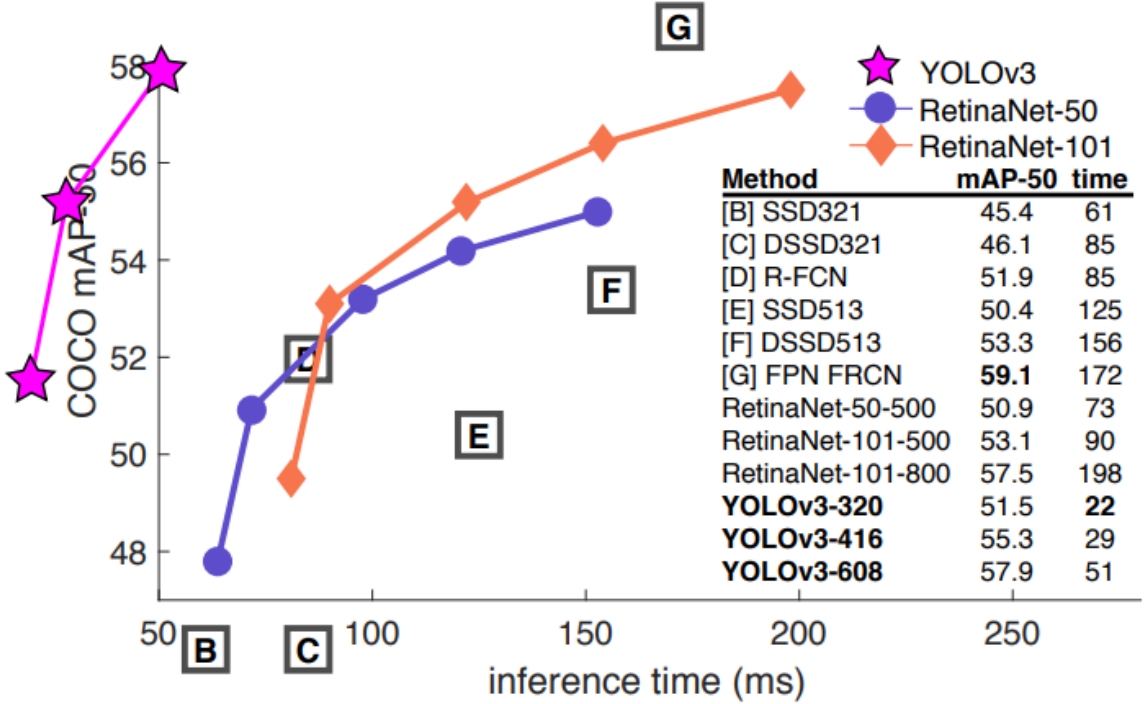
YOLO v2 Better, Faster, Stronger

Detection Frameworks	Train	mAP	FPS
Fast R-CNN [5]	2007+2012	70.0	0.5
Faster R-CNN VGG-16[15]	2007+2012	73.2	7
Faster R-CNN ResNet[6]	2007+2012	76.4	5
YOLO [14]	2007+2012	63.4	45
SSD300 [11]	2007+2012	74.3	46
SSD500 [11]	2007+2012	76.8	19
YOLOv2 288×288	2007+2012	69.0	91
YOLOv2 352×352	2007+2012	73.7	81
YOLOv2 416×416	2007+2012	76.8	67
YOLOv2 480×480	2007+2012	77.8	59
YOLOv2 544×544	2007+2012	**78.6**	40

Performance on MS COCO

Method	Data	Avg. Precision						Avg. Recall					
		IoU			Area			#Dets			Area		
		0.5:0.95	0.5	0.75	S	M	L	1	10	100	S	M	L
Fast R-CNN	train	19.7	35.9	-	-	-	-	-	-	-	-	-	-
Fast R-CNN	train	20.5	39.9	19.4	4.1	20.0	35.8	21.3	29.5	30.1	7.3	32.1	52.0
Faster R-CNN	trainval	21.9	42.7	-	-	-	-	-	-	-	-	-	-
ION	train	23.6	43.2	23.6	6.4	24.1	38.3	23.2	32.7	33.5	10.1	37.7	53.6
Faster R-CNN	trainval	24.2	45.3	23.5	7.7	26.4	37.1	23.8	34.0	34.6	12.0	38.5	54.4
SSD300	trainval35k	23.2	41.2	23.4	5.3	23.2	39.6	22.5	33.2	35.3	9.6	37.6	56.5
SSD512	trainval35k	**26.8**	**46.5**	**27.8**	**9.0**	**28.9**	**41.9**	**24.8**	**37.5**	**39.8**	**14.0**	**43.5**	**59.0**
YOLOv2	trainval35k	21.6	44.0	19.2	5.0	22.4	35.5	20.7	31.6	33.3	9.8	36.5	54.4

YOLOv3



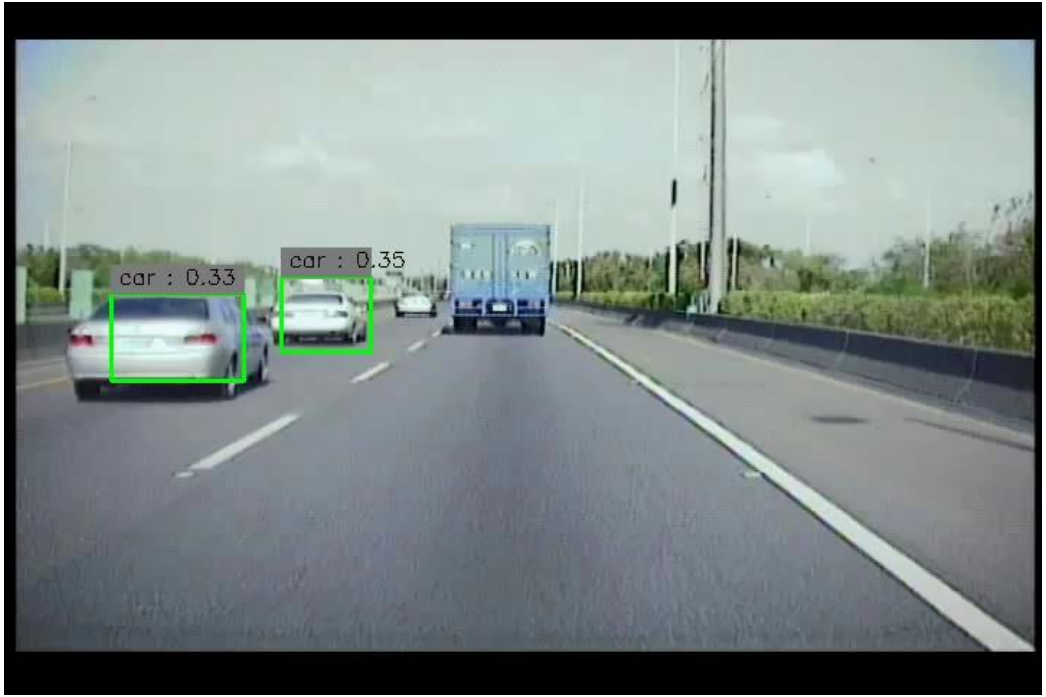
Method	mAP-50	time
[B] SSD321	45.4	61
[C] DSSD321	46.1	85
[D] R-FCN	51.9	85
[E] SSD513	50.4	125
[F] DSSD513	53.3	156
[G] FPN FRCN	**59.1**	172
RetinaNet-50-500	50.9	73
RetinaNet-101-500	53.1	90
RetinaNet-101-800	57.5	198
YOLOv3-320	51.5	**22**
YOLOv3-416	55.3	29
YOLOv3-608	57.9	51

Overall Comparison

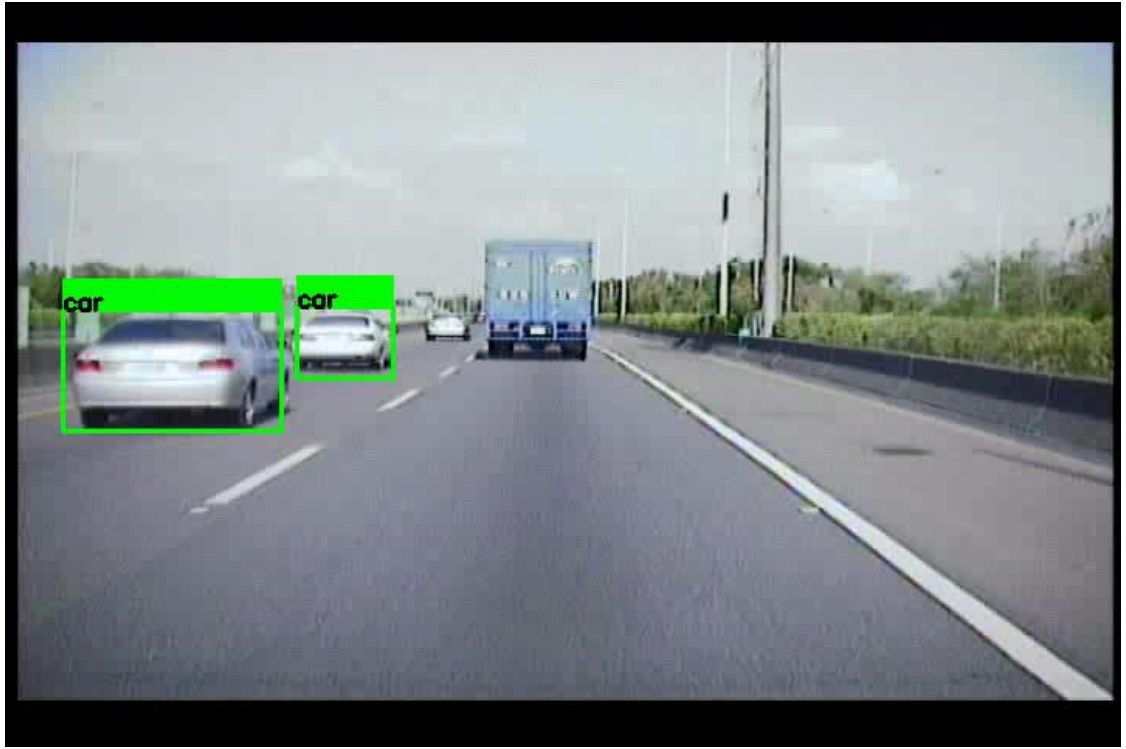
- Speed
  - single shot >> region based
- Accuracy
  - single shot >region based
- Performance
  - YOLO < Faster R-CNN (VGG-16) < SSD < Faster R-CNN (Resnet) <YOLOv2< R-FCN<DSSD<RetinaNet<YOLOv3

Vehicle Detection Result 1: SVM

Vehicle Detection Result 1: YOLO

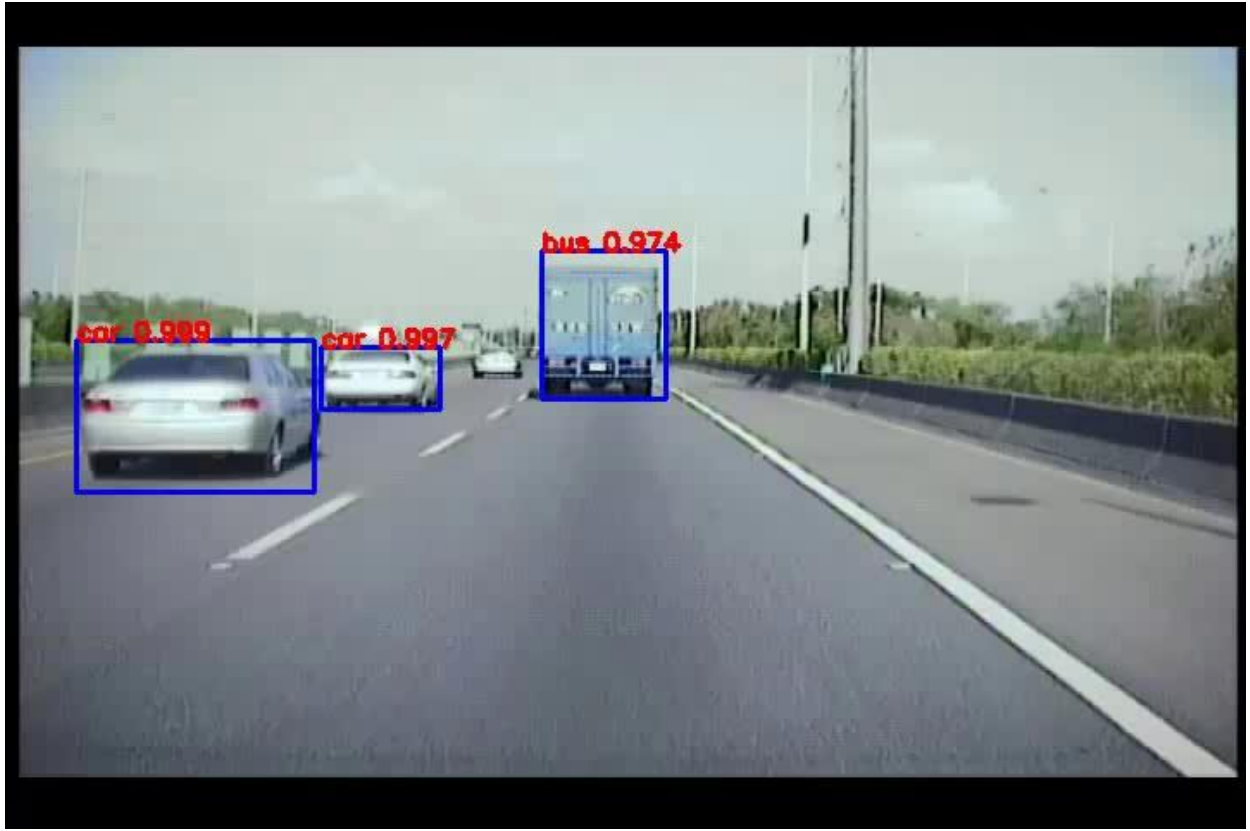


Vehicle Detection Result 1: SSD300

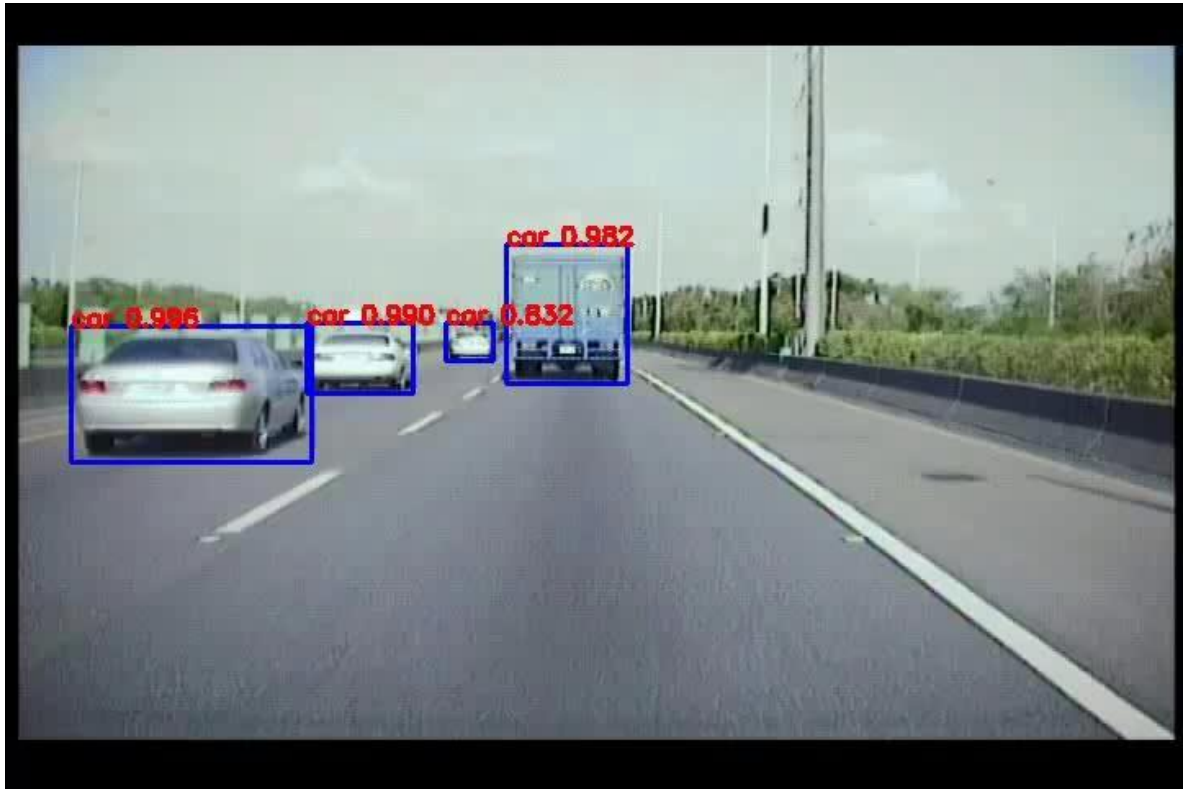


Vehicle Detection Result 1: YOLOv2

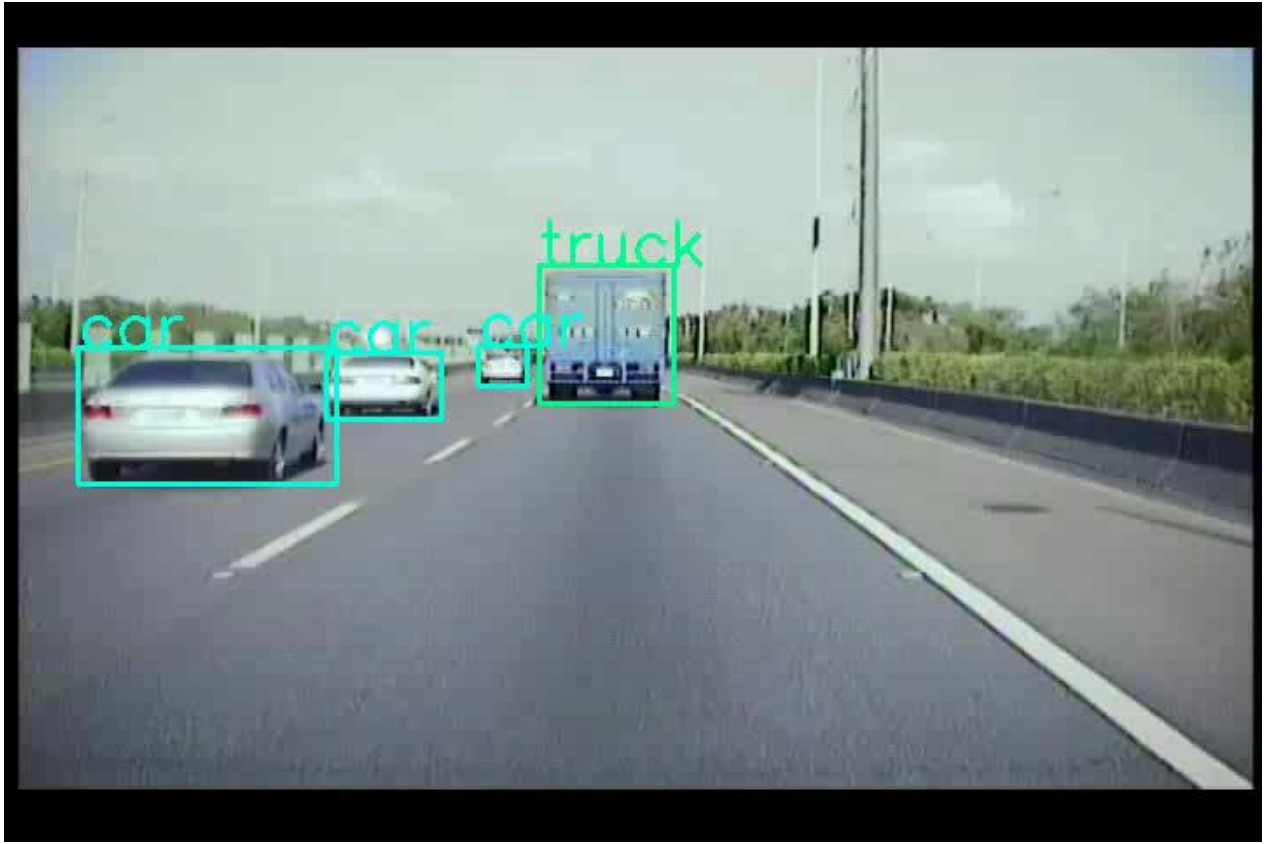
Vehicle Detection Result 1: Faster R-CNN (VGG16), original version



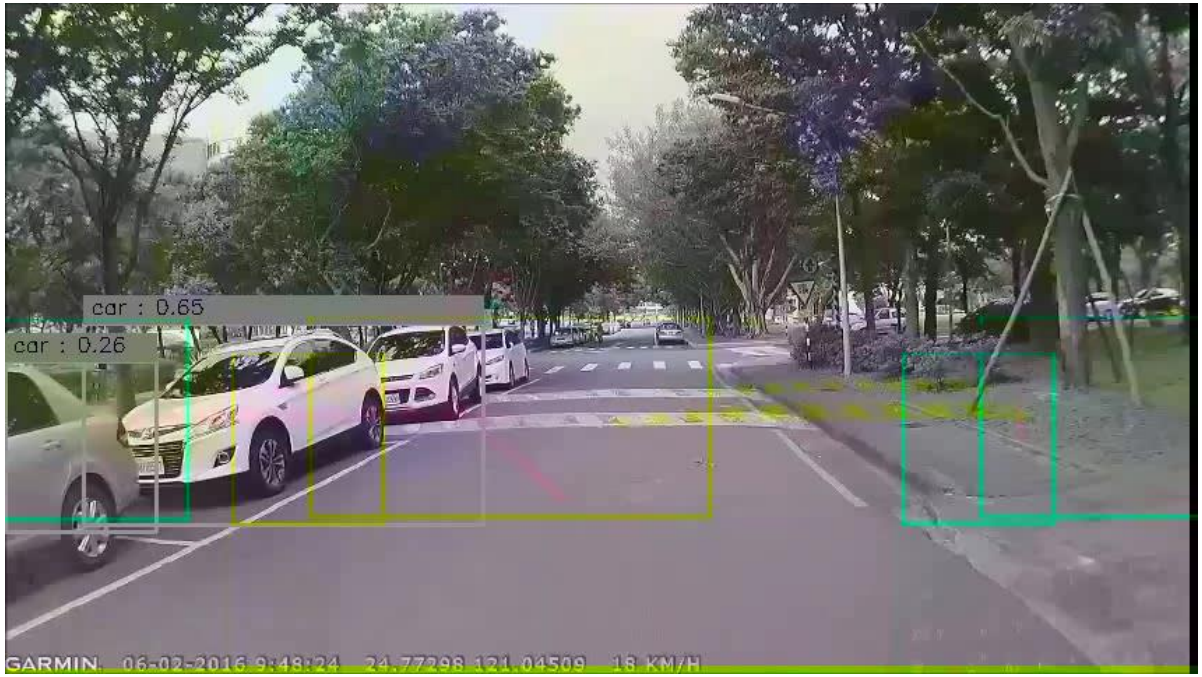
Vehicle Detection Result 1: Faster R-CNN (VGG16), fine-tuned version



Vehicle Detection Result 1: YOLOv3 (MSCOCO)



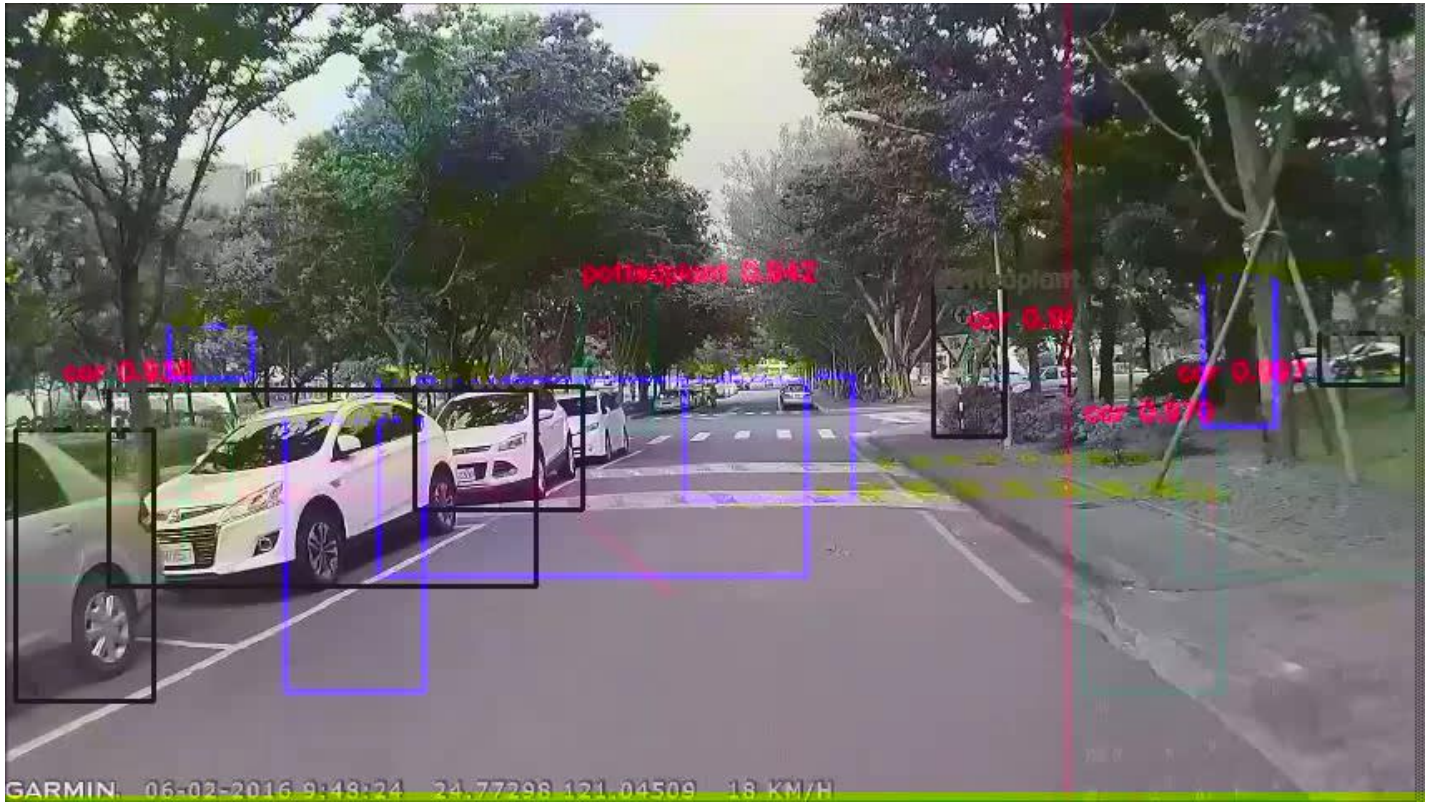
Vehicle Detection Result 2: YOLO



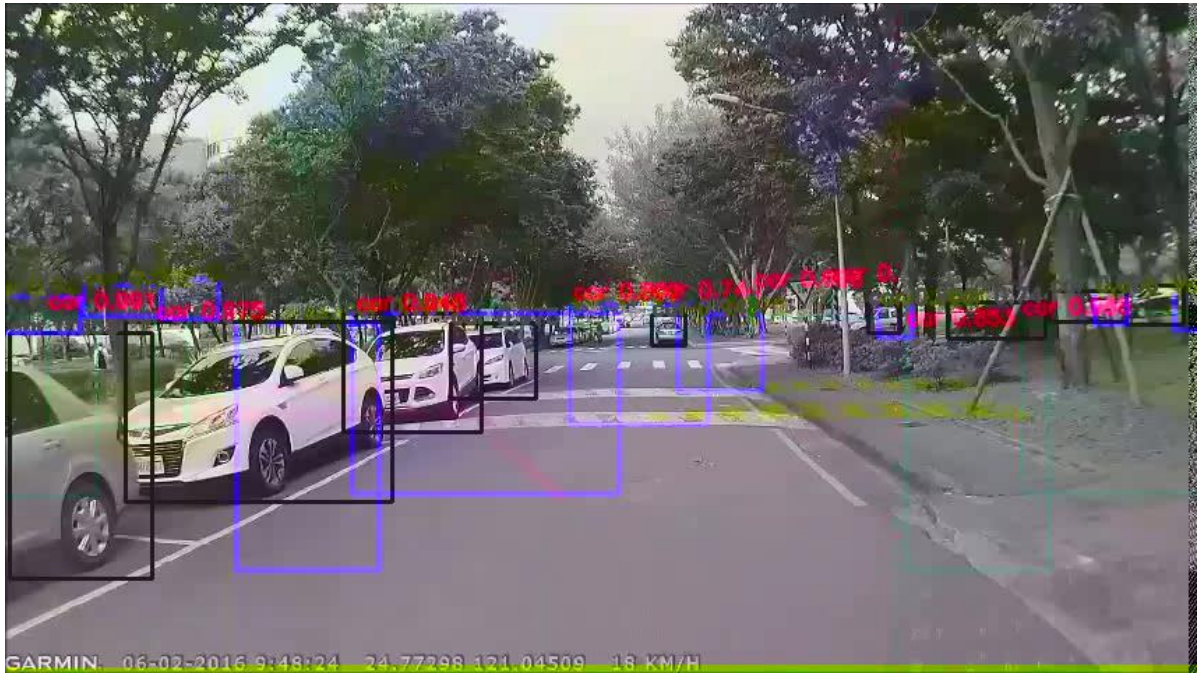
Vehicle Detection Result 2: SSD300

Vehicle Detection Result 2: YOLOv2

Vehicle Detection Result 2: Faster R-CNN (VGG16) , original version



Vehicle Detection Result 2: Faster R-CNN (VGG16)), fine-tuned version



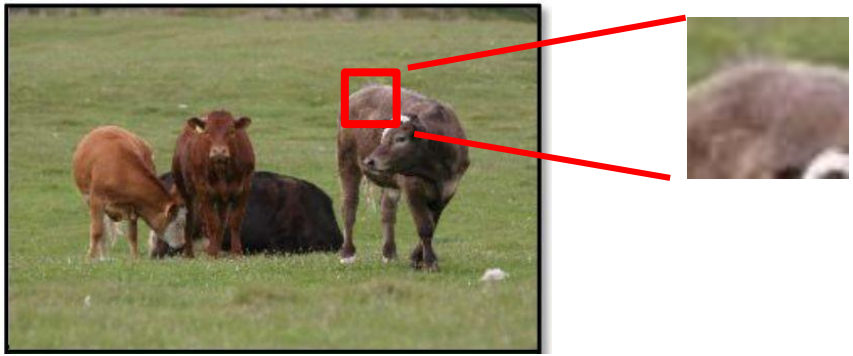
Vehicle Detection Result 2: YOLOv3

Semantic Segmentation

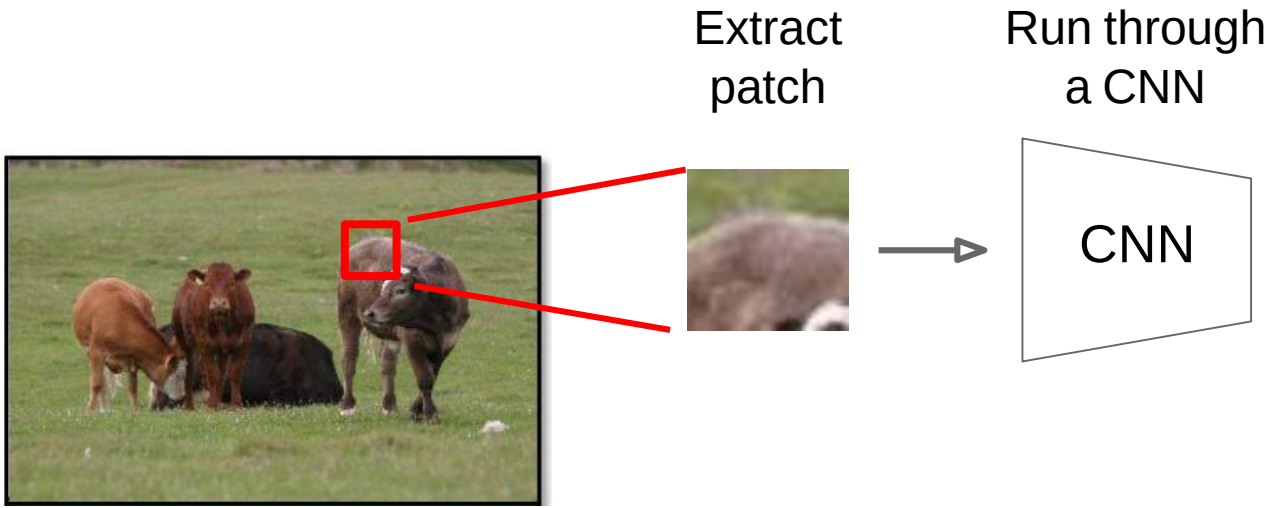
Semantic Segmentation

Semantic Segmentation

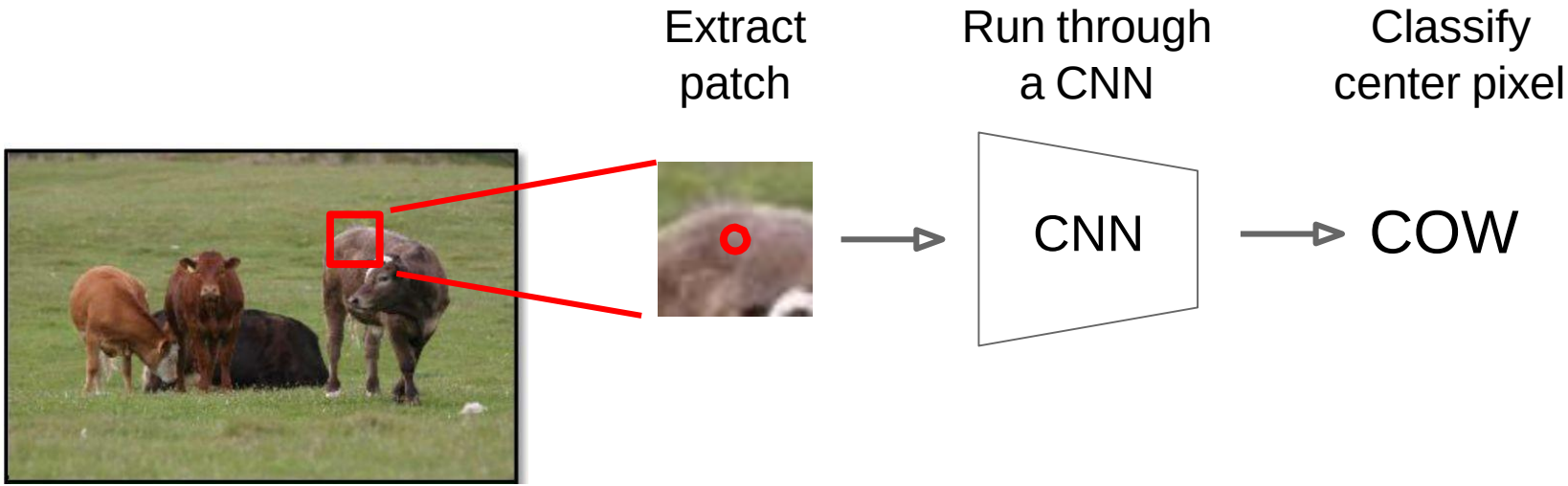
Extract patch



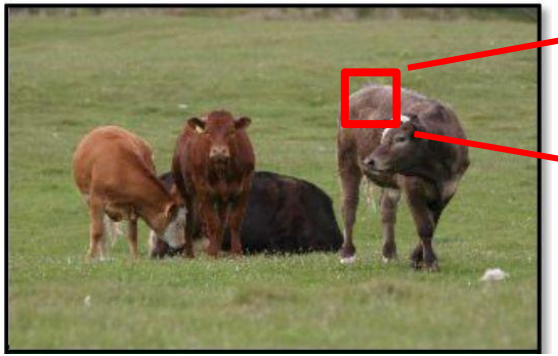
Semantic Segmentation



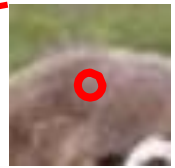
Semantic Segmentation



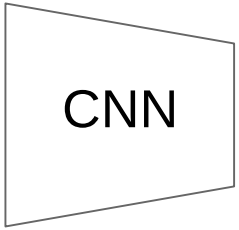
Semantic Segmentation



Extract patch



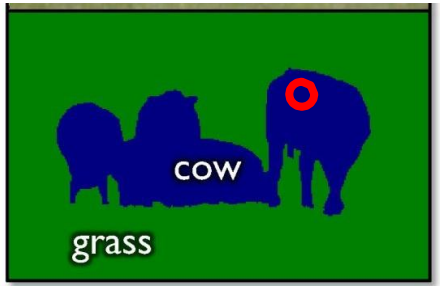
Run through a CNN



Classify center pixel

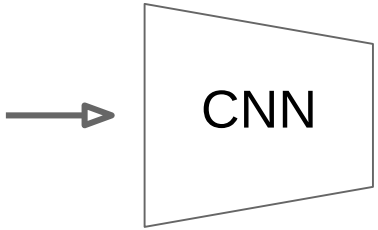
COW

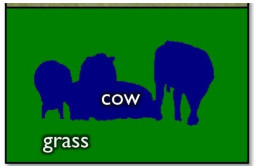
Repeat for every pixel



Semantic Segmentation

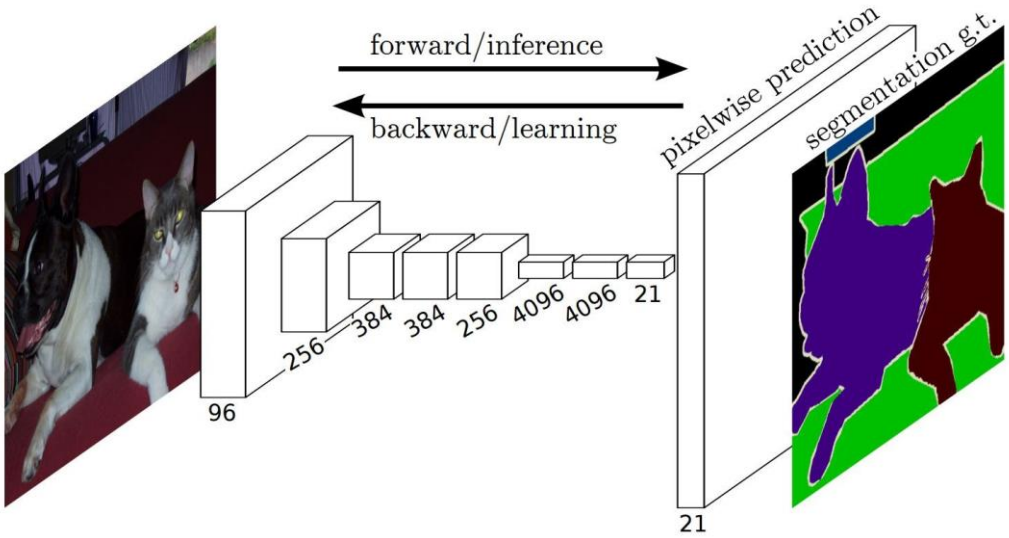
Run “fully convolutional” network
to get all pixels at once





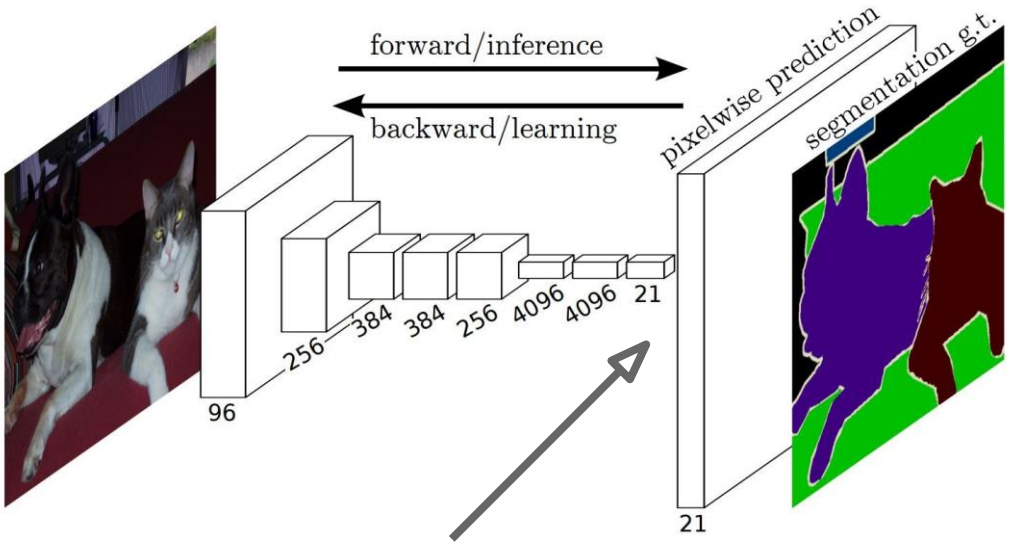
Smaller output
due to pooling

Semantic Segmentation: Upsampling



Long, Shelhamer, and Darrell, “Fully Convolutional Networks for Semantic Segmentation”, CVPR 2015

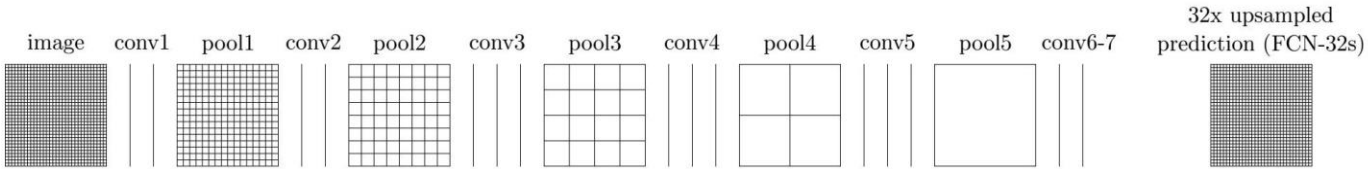
Semantic Segmentation: Upsampling



Learnable upsampling!

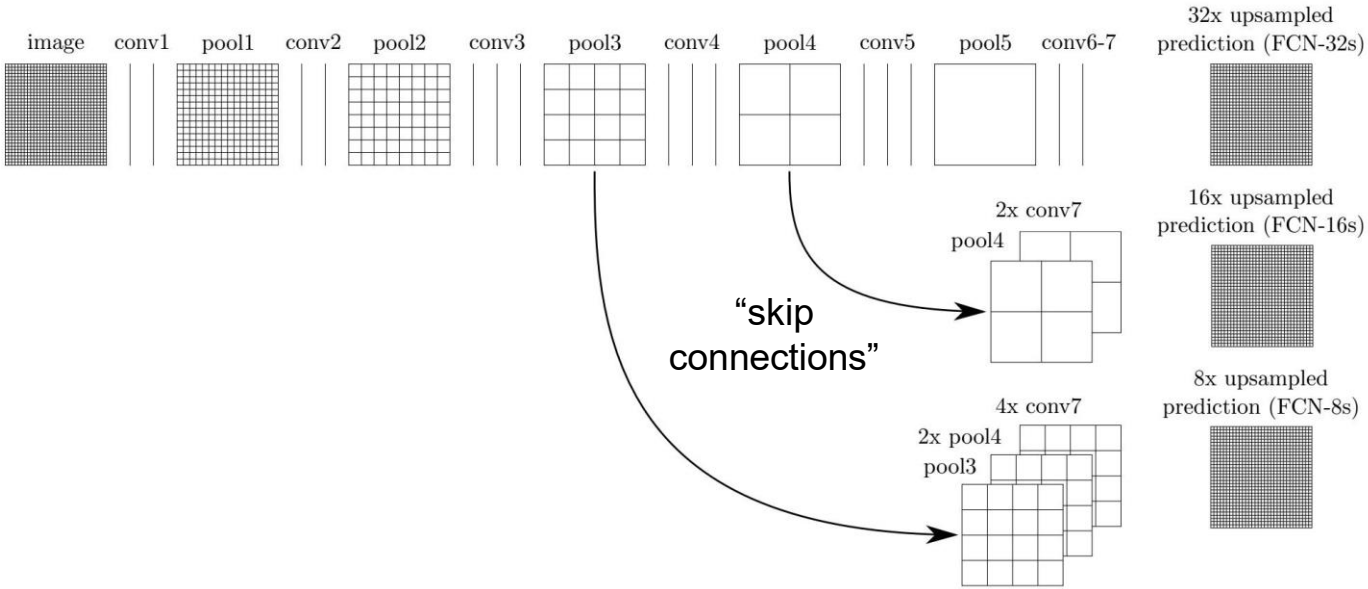
Long, Shelhamer, and Darrell, “Fully Convolutional Networks for Semantic Segmentation”, CVPR 2015

Semantic Segmentation: Upsampling



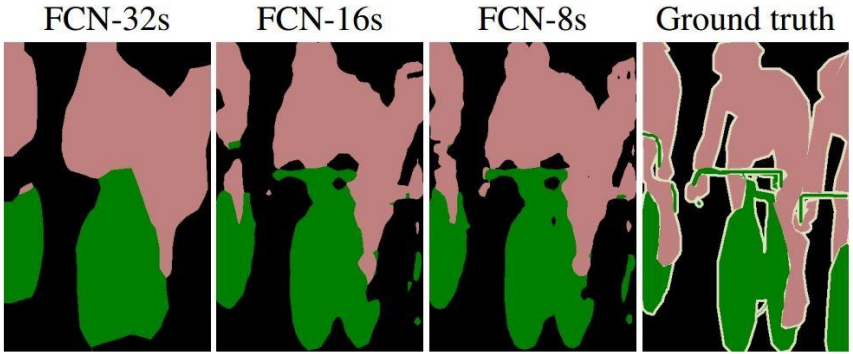
Long, Shelhamer, and Darrell, “Fully Convolutional Networks for Semantic Segmentation”, CVPR 2015

Semantic Segmentation: Upsampling

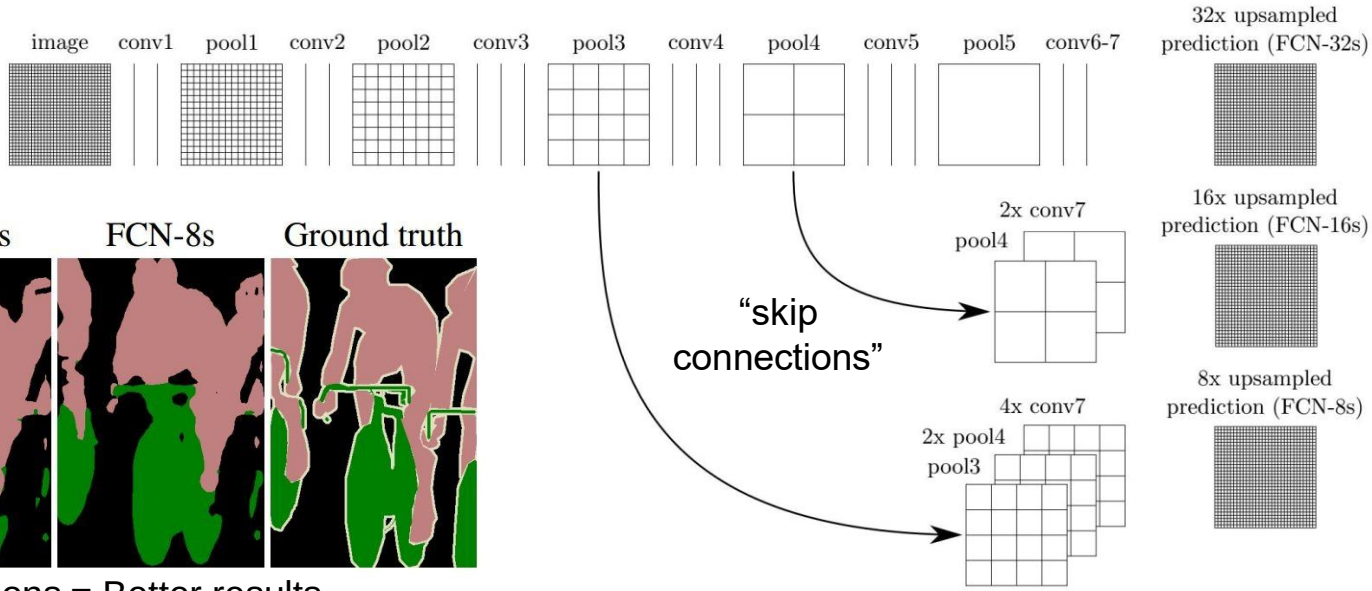


Long, Shelhamer, and Darrell, "Fully Convolutional Networks for Semantic Segmentation", CVPR 2015

Semantic Segmentation: Upsampling

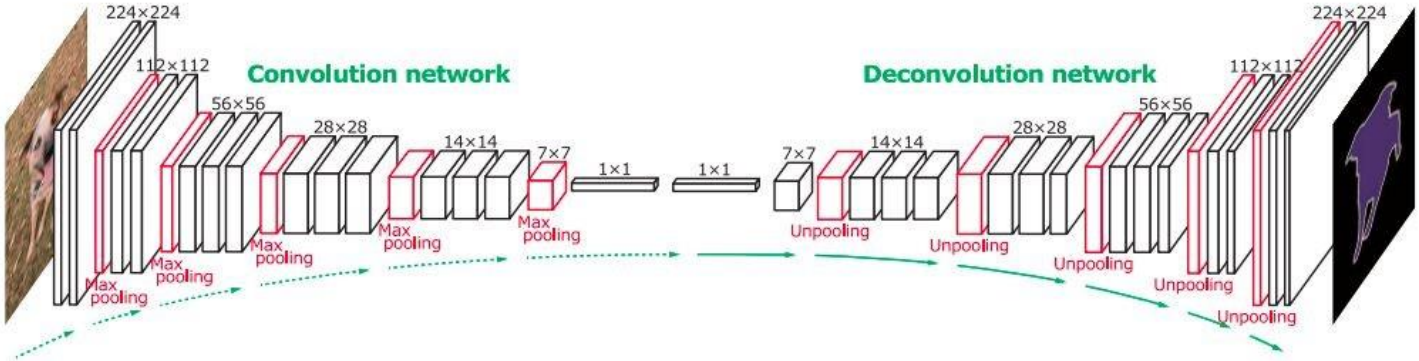


Skip connections = Better results



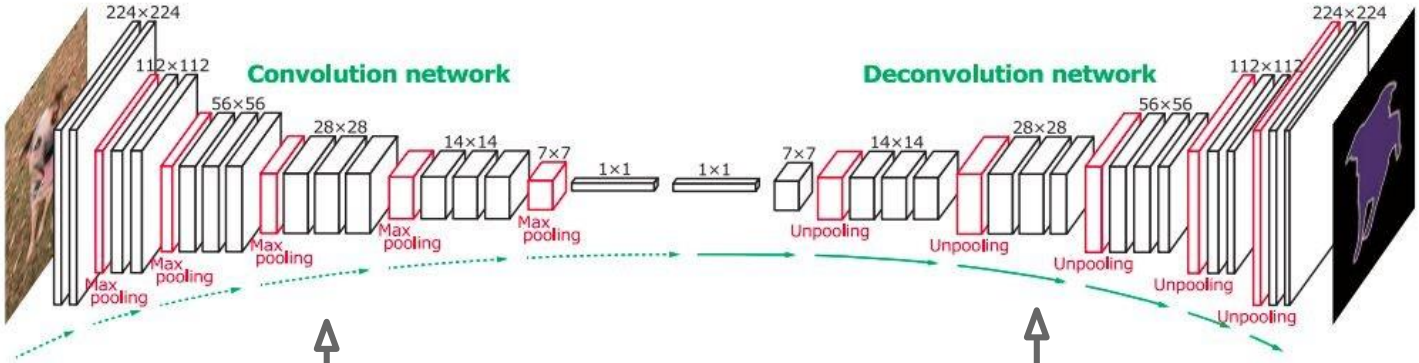
Long, Shelhamer, and Darrell, "Fully Convolutional Networks for Semantic Segmentation", CVPR 2015

Semantic Segmentation: Upsampling



Noh et al, “Learning Deconvolution Network for Semantic Segmentation”, ICCV 2015

Semantic Segmentation: Upsampling



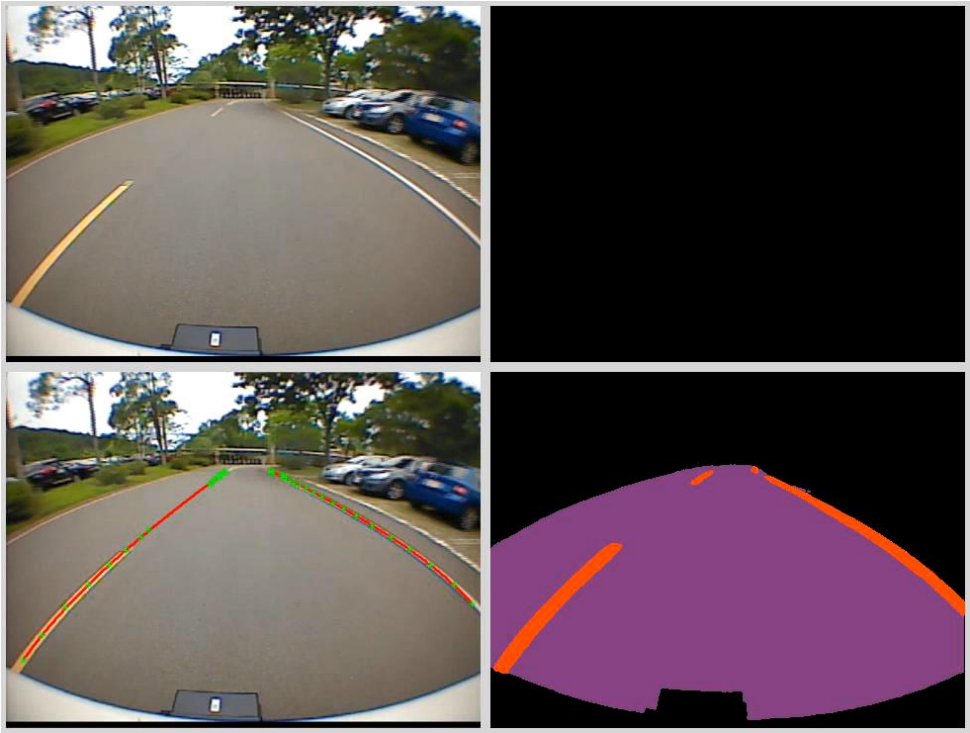
Normal VGG

“Upside down” VGG

6 days of training on Titan X…

Noh et al, “Learning Deconvolution Network for Semantic Segmentation”, ICCV 2015

Lane detection using semantic segmentation



Instance Segmentation

Instance Segmentation

Detect instances, give category, label pixels

“simultaneous detection and segmentation” (SDS)

Lots of recent work (MS-COCO)

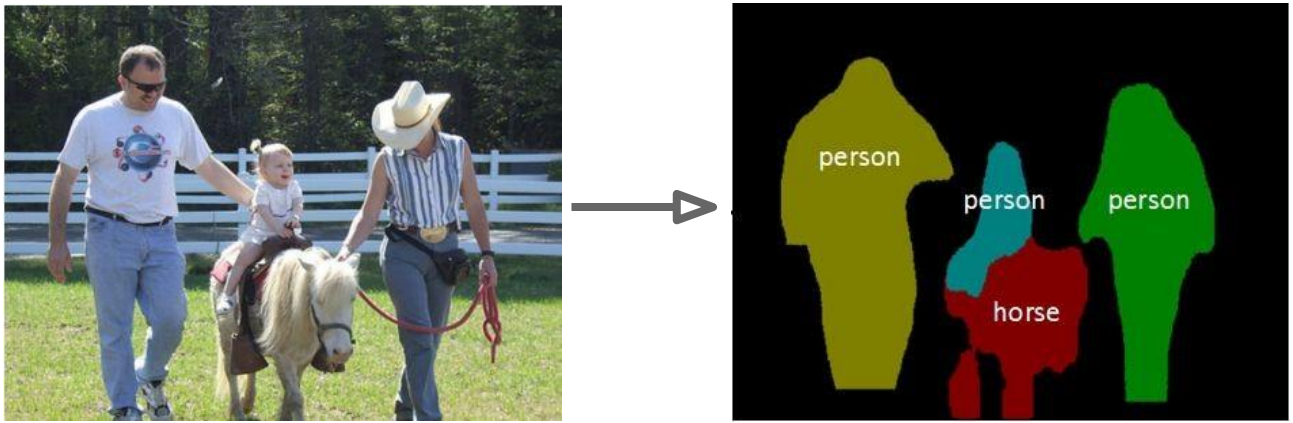


Figure credit: Dai et al, “Instance-aware Semantic Segmentation via Multi-task Network Cascades”, arXiv 2015

Instance Segmentation

Similar to R-CNN, but with segments

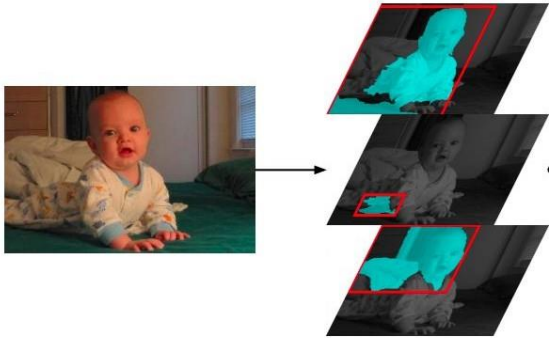
Hariharan et al, “Simultaneous Detection and Segmentation”, ECCV 2014

Instance Segmentation

Similar to R-CNN, but with segments

Proposal Generation

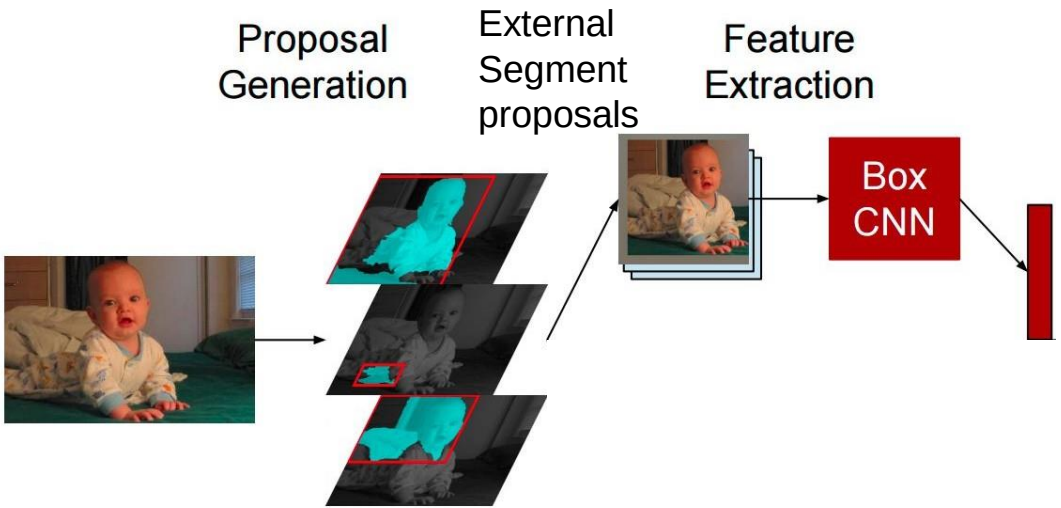
External Segment proposals



Hariharan et al, “Simultaneous Detection and Segmentation”, ECCV 2014

Instance Segmentation

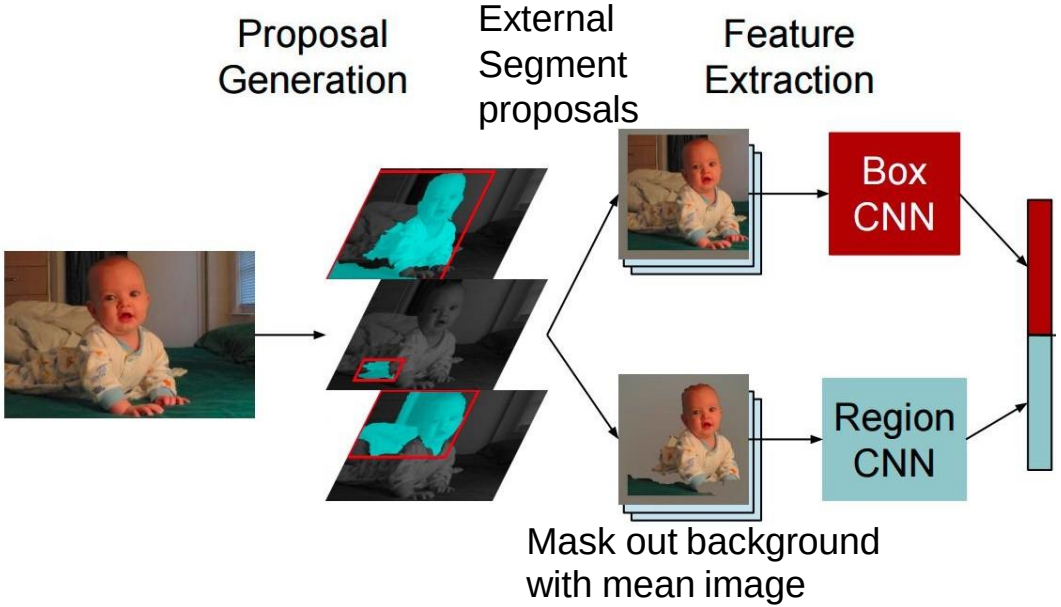
Similar to R-CNN, but with segments



Hariharan et al, “Simultaneous Detection and Segmentation”, ECCV 2014

Instance Segmentation

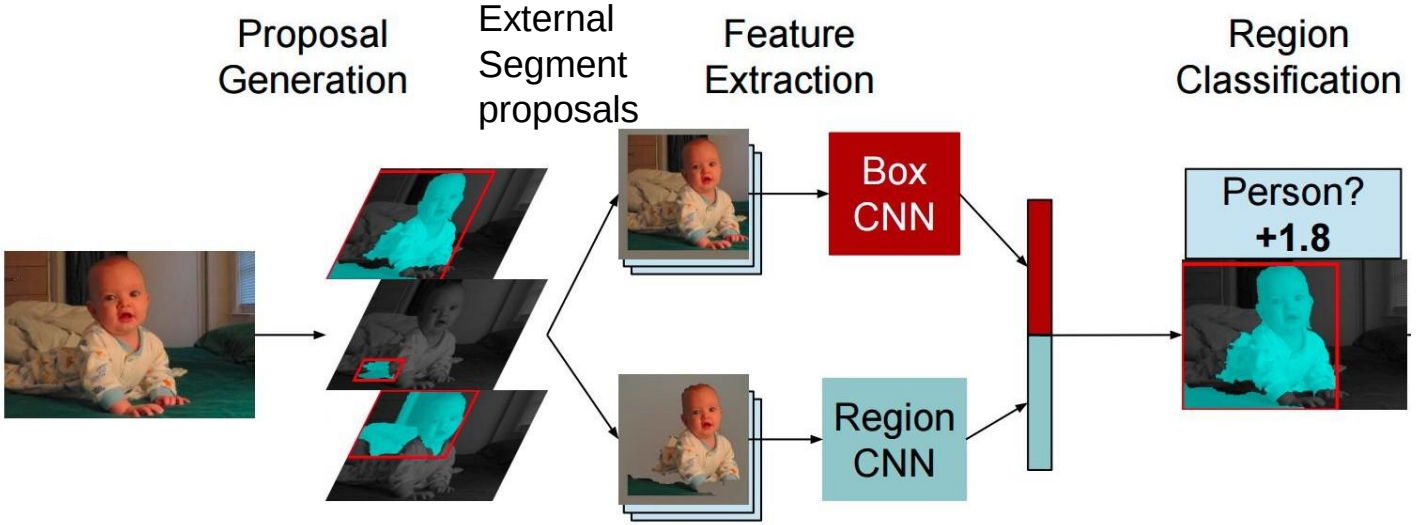
Similar to R-CNN, but with segments



Hariharan et al, “Simultaneous Detection and Segmentation”, ECCV 2014

Instance Segmentation

Similar to R-CNN, but with segments

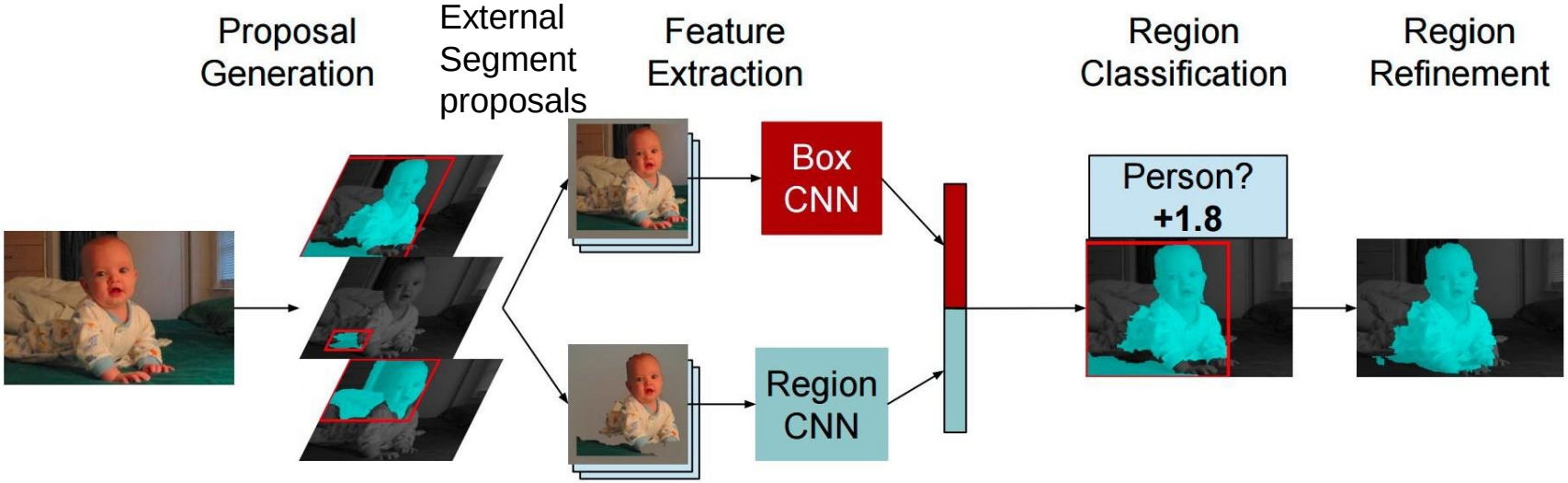


Mask out background with mean image

Hariharan et al, “Simultaneous Detection and Segmentation”, ECCV 2014

Instance Segmentation

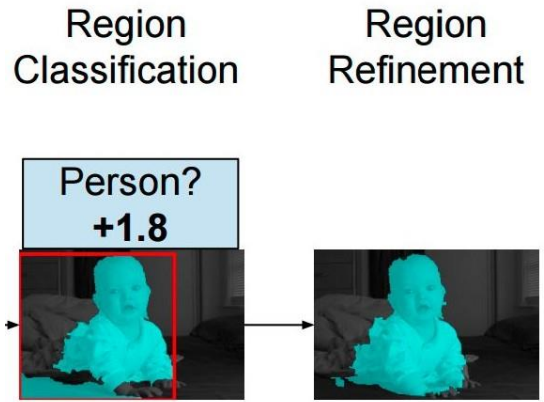
Similar to R-CNN, but with segments



Mask out background with mean image

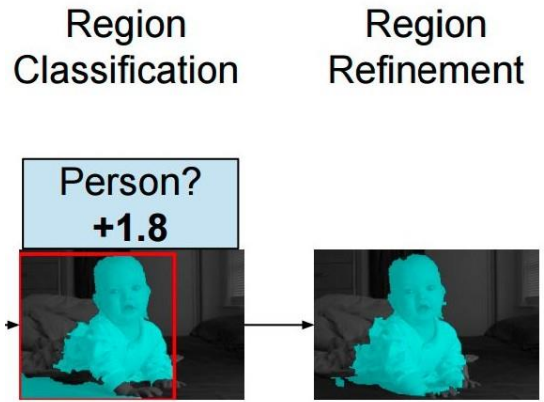
Hariharan et al, “Simultaneous Detection and Segmentation”, ECCV 2014

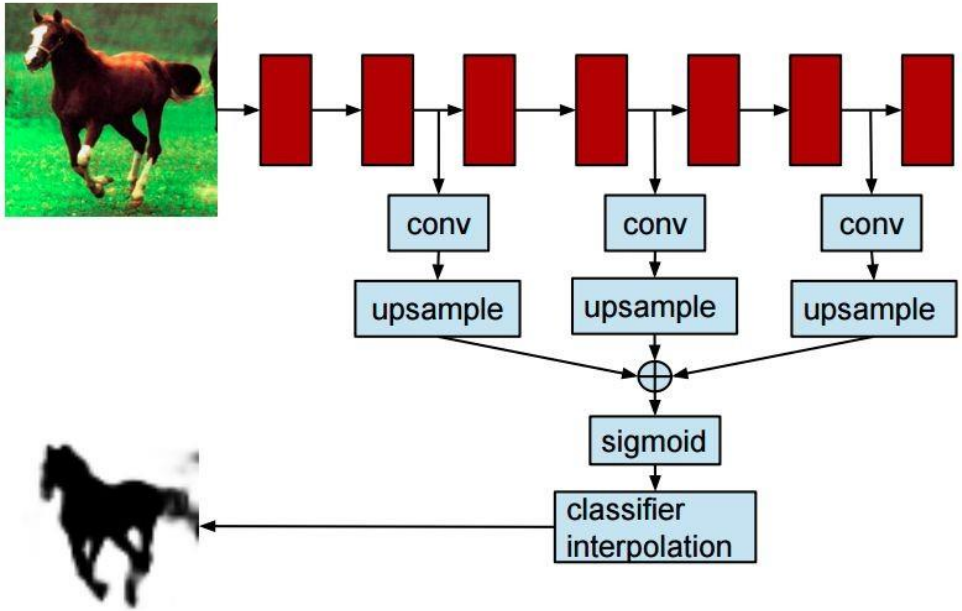
Instance Segmentation: Hypercolumns



Hariharan et al, “Hypercolumns for Object Segmentation and Fine-grained Localization”, CVPR 2015

Instance Segmentation: Hypercolumns





Hariharan et al, “Hypercolumns for Object Segmentation and Fine-grained Localization”, CVPR 2015

Instance Segmentation: Cascades

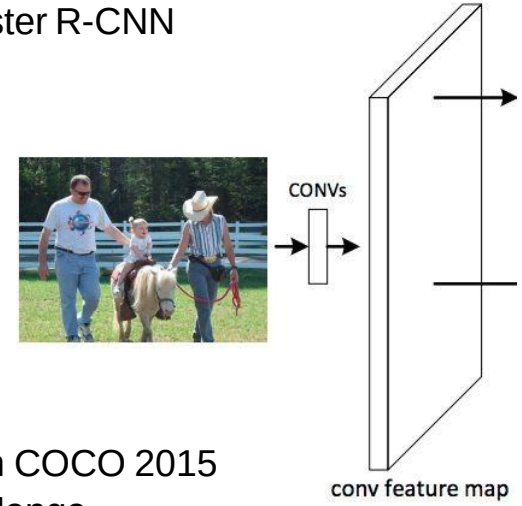
Similar to
Faster R-CNN

Won COCO 2015
challenge
(with ResNet)

Dai et al, “Instance-aware Semantic Segmentation via Multi-task Network Cascades”, arXiv 2015

Instance Segmentation: Cascades

Similar to
Faster R-CNN

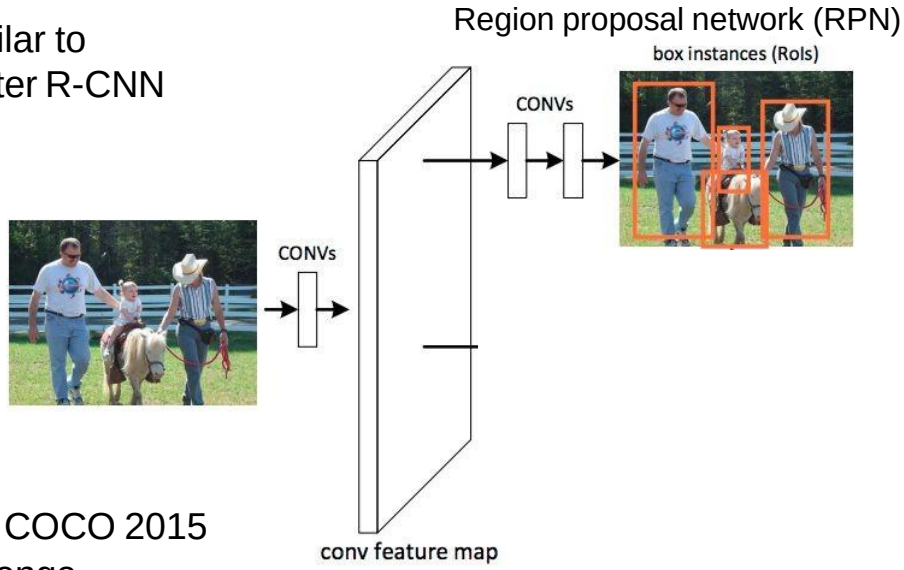


Won COCO 2015
challenge
(with ResNet)

Dai et al, “Instance-aware Semantic Segmentation via Multi-task Network Cascades”, arXiv 2015

Instance Segmentation: Cascades

Similar to
Faster R-CNN



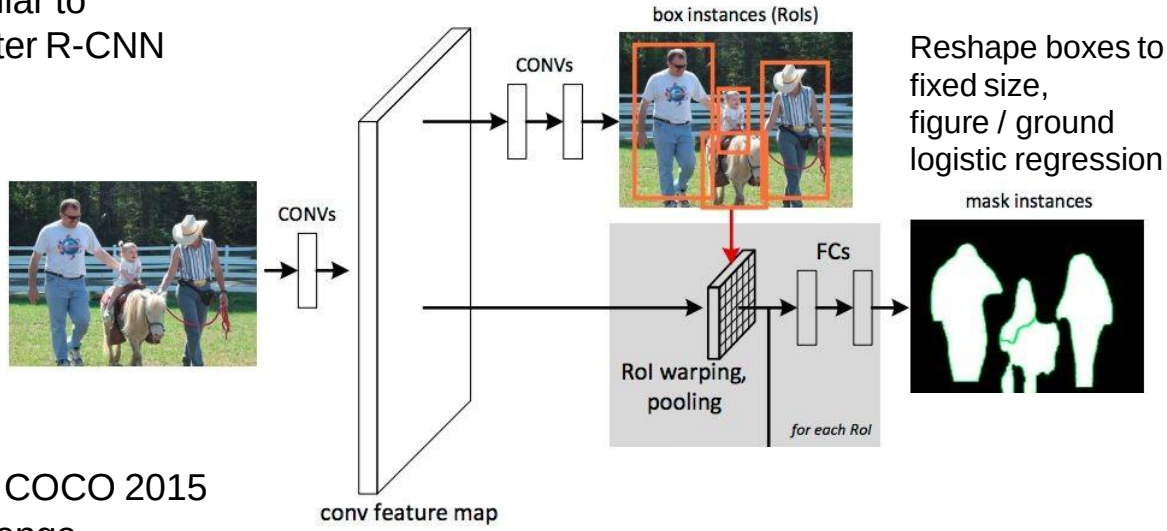
Won COCO 2015
challenge
(with ResNet)

Dai et al, “Instance-aware Semantic Segmentation via Multi-task Network Cascades”, arXiv 2015

Instance Segmentation: Cascades

Similar to
Faster R-CNN

Region proposal network (RPN)



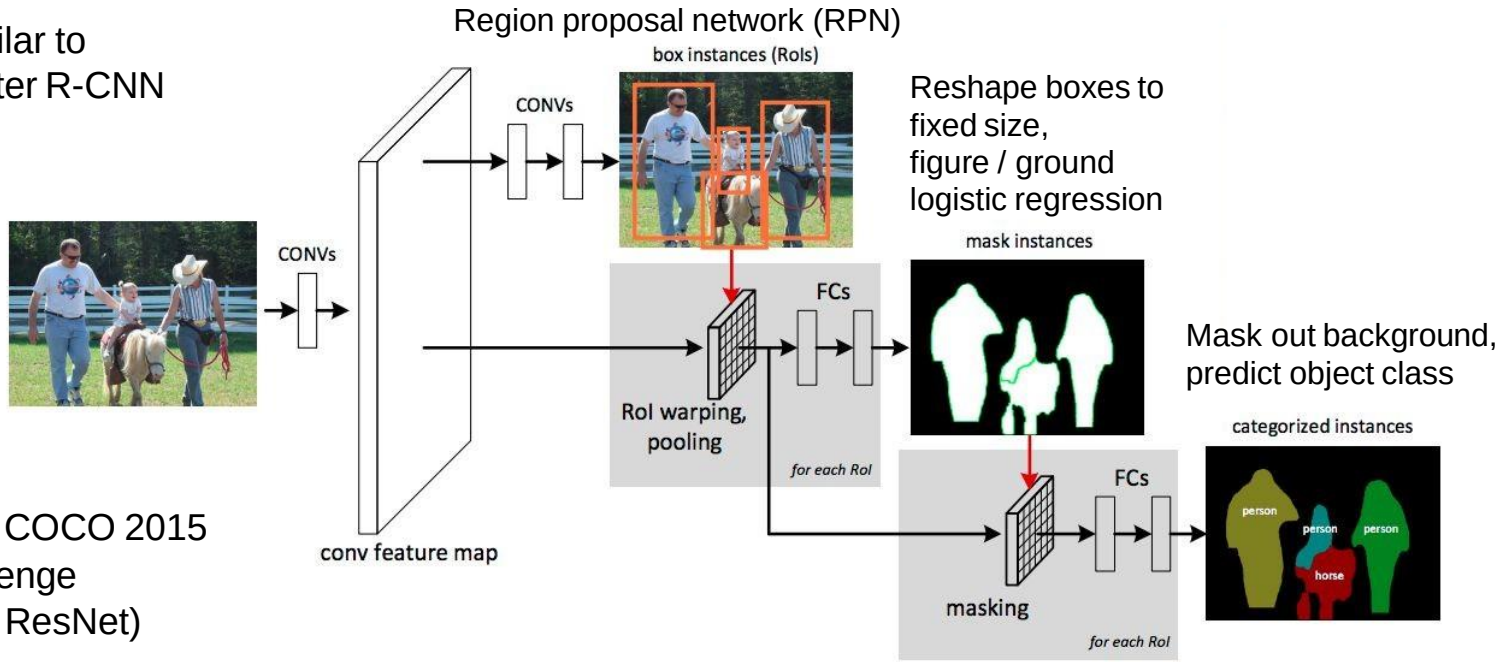
Reshape boxes to
fixed size,
figure / ground
logistic regression

Won COCO 2015
challenge
(with ResNet)

Dai et al, “Instance-aware Semantic Segmentation via Multi-task Network Cascades”, arXiv 2015

Instance Segmentation: Cascades

Similar to
Faster R-CNN



Reshape boxes to
fixed size,
figure / ground
logistic regression

Mask out background,
predict object class

Won COCO 2015
challenge
(with ResNet)

Dai et al, “Instance-aware Semantic Segmentation via Multi-task Network Cascades”, arXiv 2015

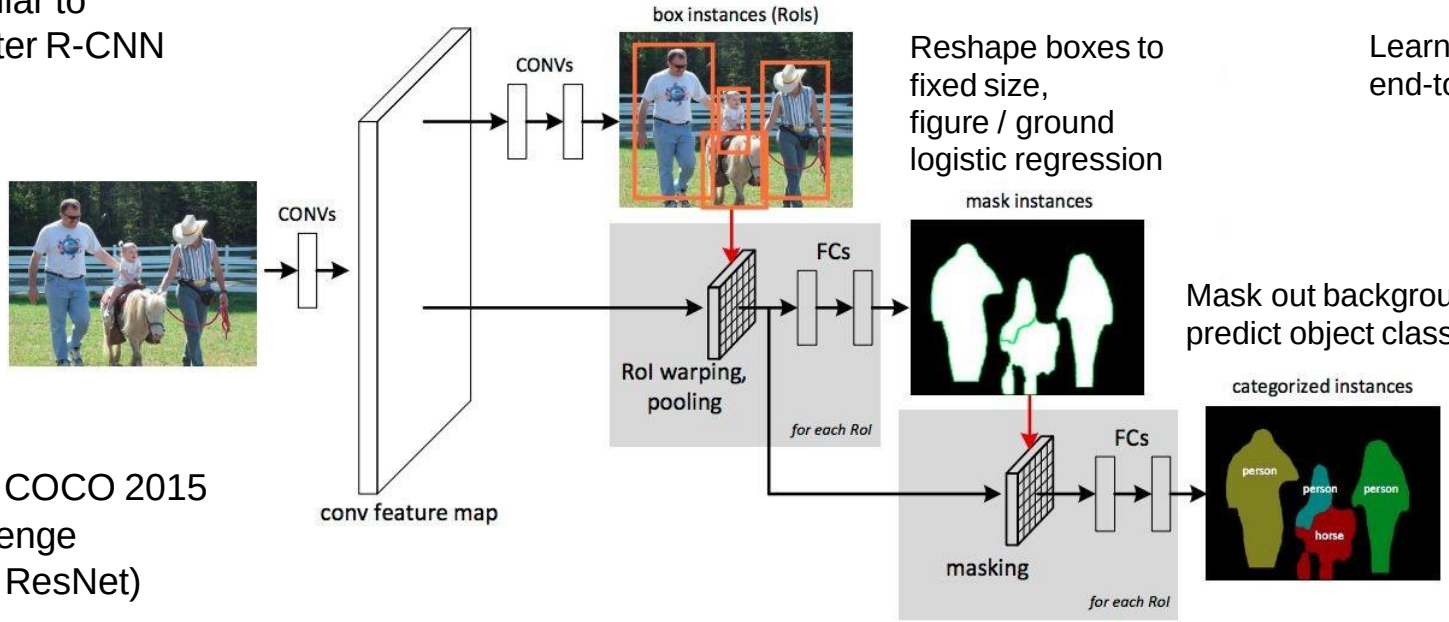
Instance Segmentation: Cascades

Similar to
Faster R-CNN

Region proposal network (RPN)

Reshape boxes to
fixed size,
figure / ground
logistic regression

Learn entire model
end-to-end!

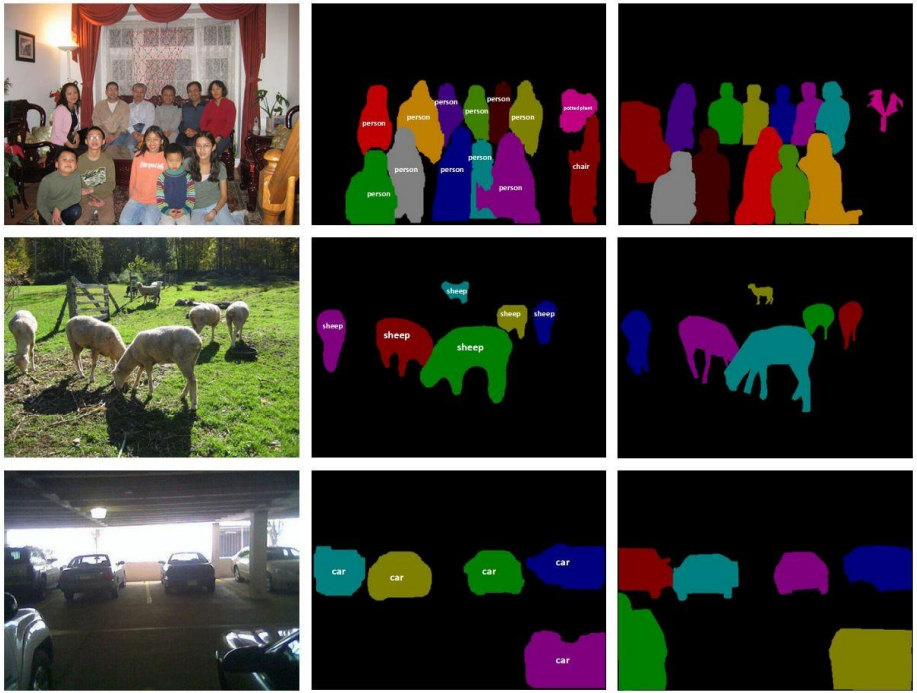


Mask out background,
predict object class

Won COCO 2015
challenge
(with ResNet)

Dai et al, “Instance-aware Semantic Segmentation via Multi-task Network Cascades”, arXiv 2015

Instance Segmentation: Cascades

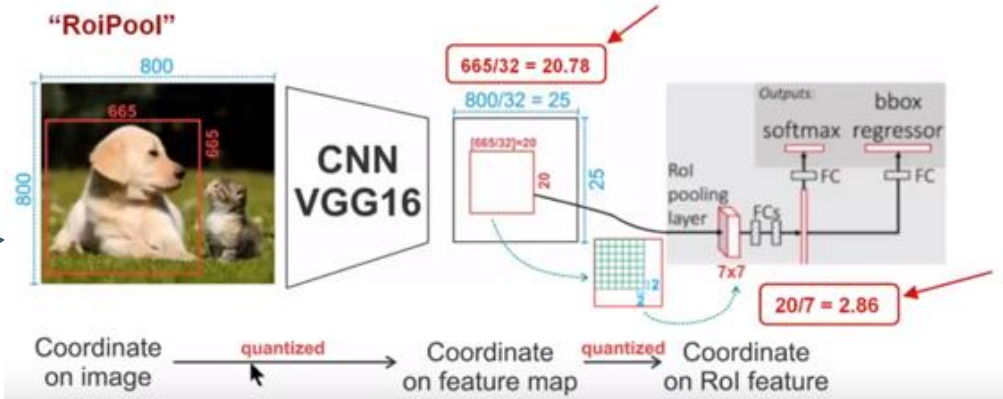


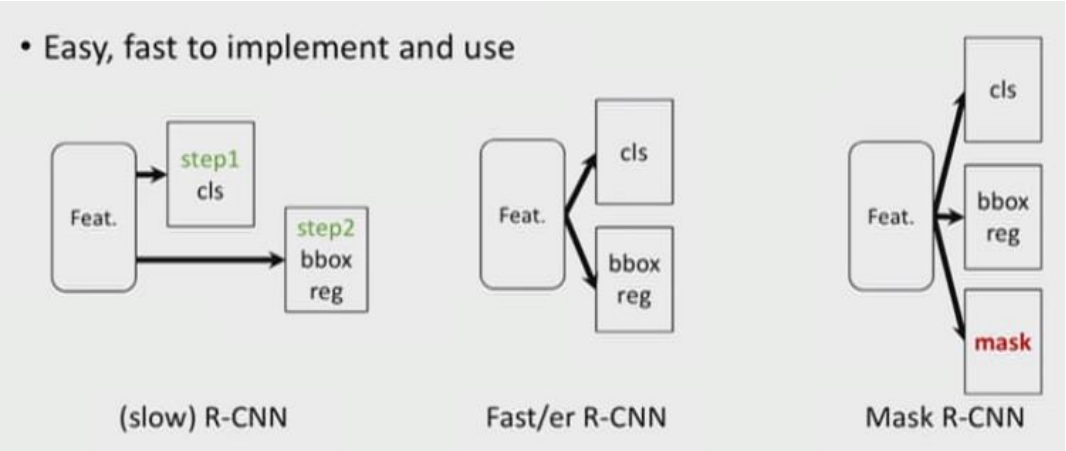
Predictions **Ground truth**

Dai et al, “Instance-aware Semantic Segmentation via Multi-task Network Cascades”, arXiv 2015

Mask R-CNN

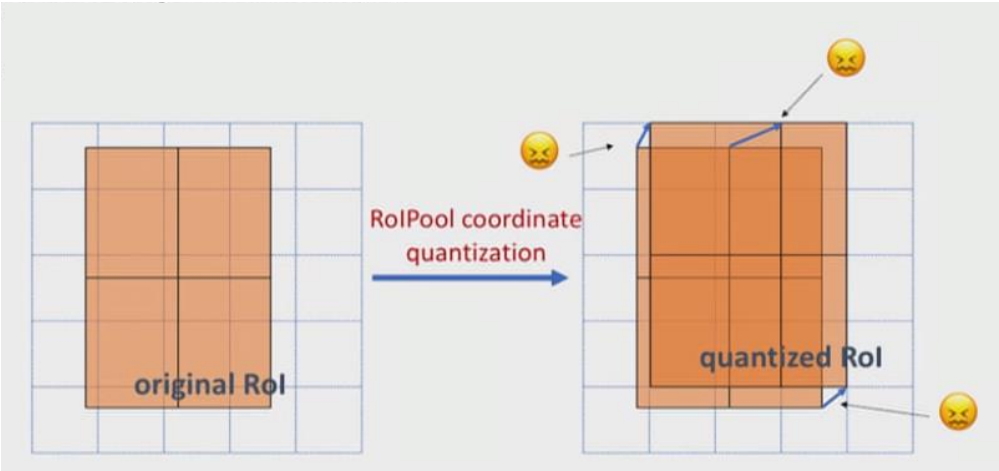
- End-to-End learning in Bbox reg, classification and masking
- Defect of ROI pooling



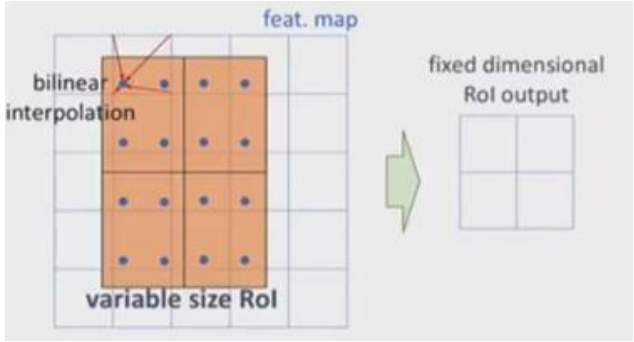


Mask R-CNN: ROI pooling vs ROI align

- ROI pooling:
  - was not for segmentation
  - breaks pixel-to-pixel alignment



ROI Pooling



ROI Align

Mask R-CNN

- Backbone: ResNet, ResNeXt, Feature Pyramid Net (FPN)
- A meta algorithm

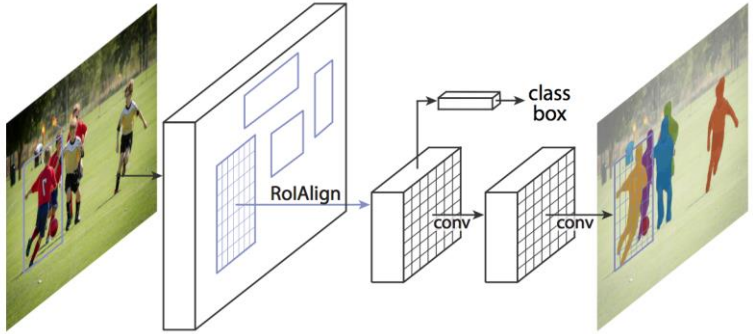
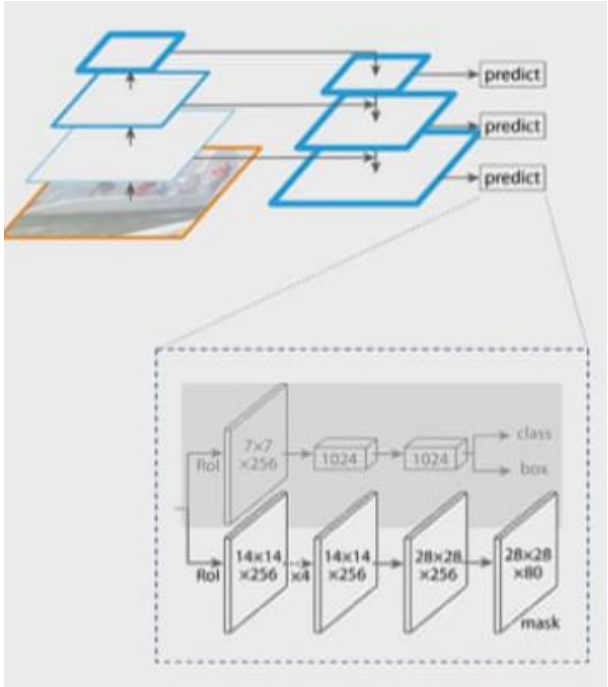
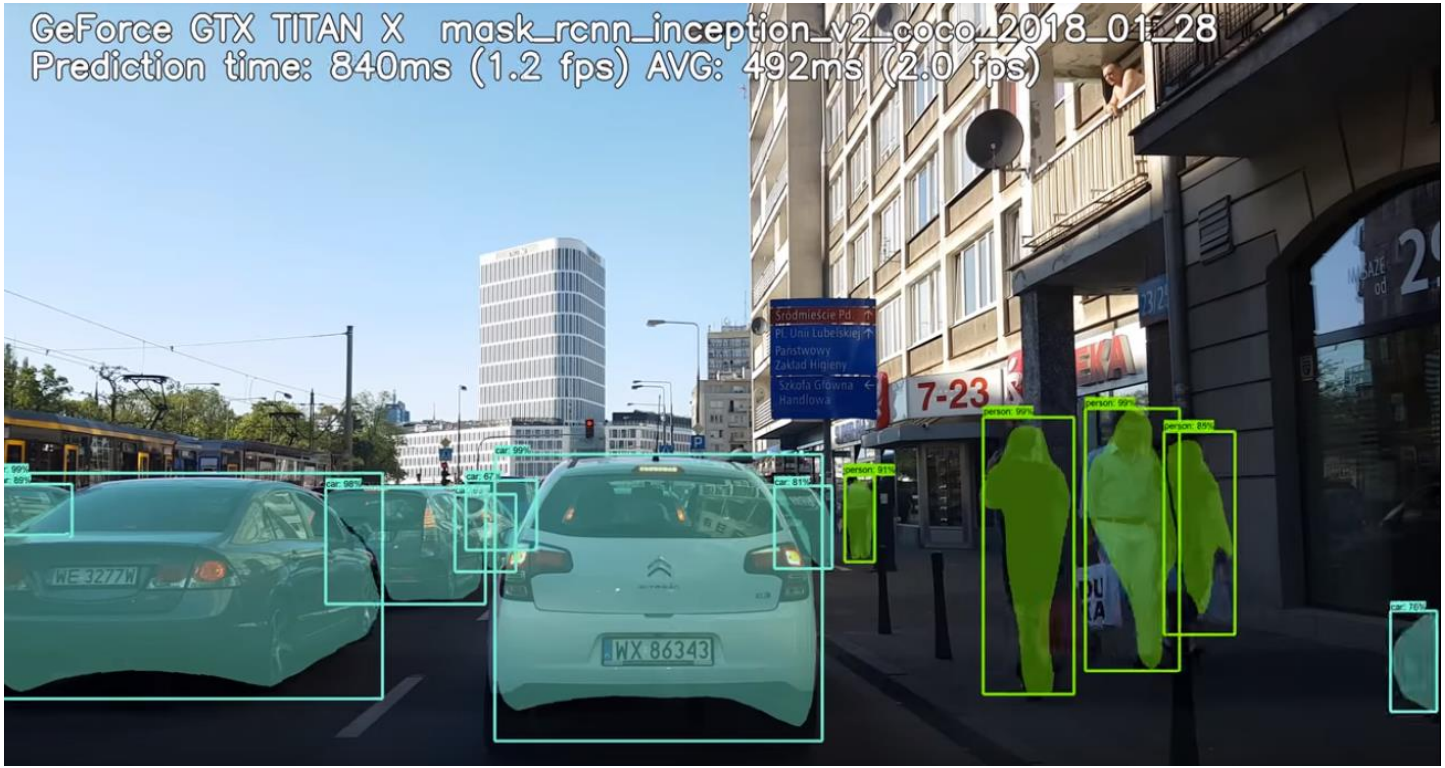


Figure 1. The **Mask R-CNN** framework for instance segmentation.



Mask R-CNN: Demo



Mask R-CNN: possible application in the future

Segmentation Overview

- Semantic segmentation
  - Classify all pixels
  - Fully convolutional models, downsample then upsample
  - Learnable upsampling: fractionally strided convolution
  - Skip connections can help
- Instance Segmentation
  - Detect instance, generate mask
  - Similar pipelines to object detection