Deep Object Detection

Kaustav Kundu

University of Toronto

February 9, 2016

Outline

- Object Detection Task

Outline

- Object Detection Task
- Object Detection Pipeline

Outline

- Object Detection Task
- Object Detection Pipeline
- Using Deep Networks
    - RCNN

Outline

- Object Detection Task
- Object Detection Pipeline
- Using Deep Networks
  - RCNN
  - Fast RCNN

Outline

- Object Detection Task
- Object Detection Pipeline
- Using Deep Networks
    - RCNN
    - Fast RCNN
    - Faster RCNN

Object Detection

Input Image

Object Detection

Input Image

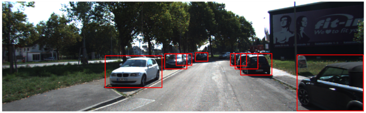
Question: Where are the cars in the image?

Object Detection

Input Image

Question: Where are the cars in the image?

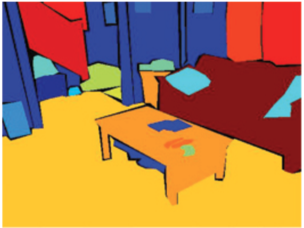
Answer:



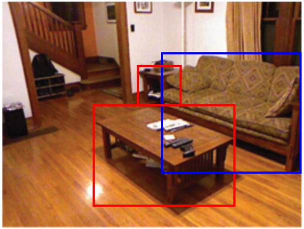
Object Detection Approach: Recognition + Localization

Object Segmentation vs Detection

Input Image

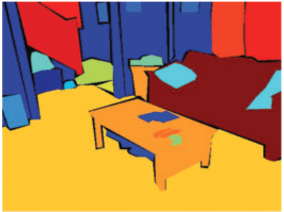


Object Segmentation

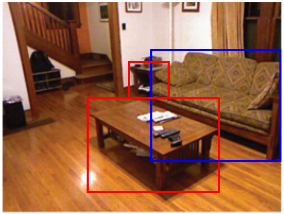


Object Detection

Object Segmentation vs Detection



Object Segmentation



Object Detection

	Object Segmentation	Object Detection
Dense Labeling	✓	✗
Instance Level	✗	✓
Stuff Category	✓	✗
Metric	IoU	AP at IoU=0.5[1]
Annotations	Difficult	Easier

[1] Modifications used sometimes, *e.g.* KITTI, MS COCO

Typical Object Detection Pipeline

Input Image

Typical Object Detection Pipeline

Input Image

- Candidate Box Selection

Typical Object Detection Pipeline

Input Image

$$\begin{bmatrix} \mathbf{x} \end{bmatrix}$$

Feature Extraction

- Candidate Box Selection
- Feature Extraction

Typical Object Detection Pipeline

Input Image

$$\begin{bmatrix} \mathbf{x} \end{bmatrix}$$

Feature Extraction

$f_c(\mathbf{x})$

- Candidate Box Selection
- Feature Extraction
- Classification

Typical Object Detection Pipeline

Input Image

$$\begin{bmatrix} \mathbf{x} \end{bmatrix}$$

Feature Extraction

$f_c(\mathbf{x})$

Classification

- Candidate Box Selection
- Feature Extraction
- Classification
- Post processing

Typical Object Detection Pipeline

	Candidate Box Selection	**Feature Extraction**	**Classification**
Pre Deep Era	Exhaustive	Hand-crafted (*e.g.* HOG)	Linear
RCNN			
Fast RCNN			
Faster RCNN			

Typical Object Detection Pipeline

	Candidate Box Selection	Feature Extraction	Classification
Pre Deep Era	Exhaustive	Hand-crafted (*e.g.* HOG)	Linear
RCNN	Region Proposal	Deep	Linear
Fast RCNN			
Faster RCNN			

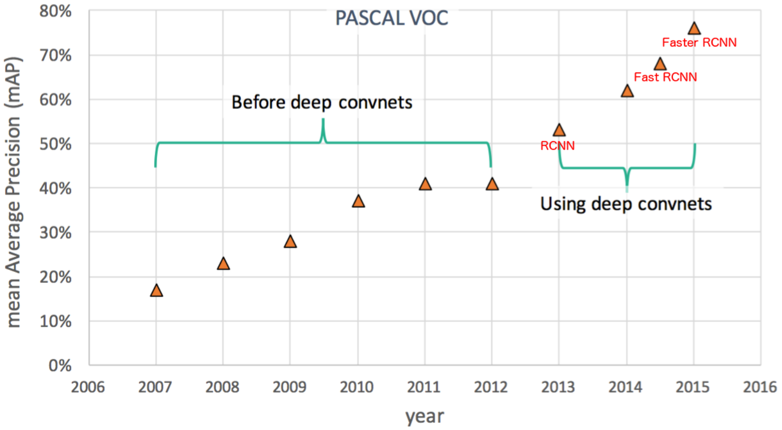
Typical Object Detection Pipeline

	Candidate Box Selection	**Feature Extraction**	**Classification**
Pre Deep Era	Exhaustive	Hand-crafted (*e.g.* HOG)	Linear
RCNN	Region Proposal	Deep	Linear
Fast RCNN	Region Proposal	Deep	
Faster RCNN			

Typical Object Detection Pipeline

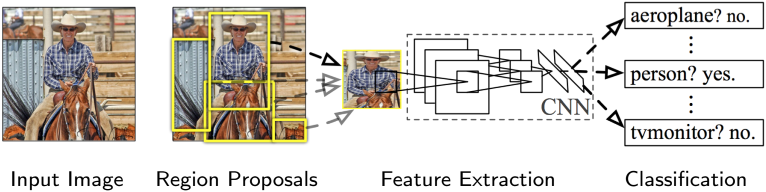
	Candidate Box Selection	**Feature Extraction**	**Classification**
Pre Deep Era	Exhaustive	Hand-crafted (*e.g.* HOG)	Linear
RCNN	Region Proposal	Deep	Linear
Fast RCNN	Region Proposal	Deep	
Faster RCNN	Deep	Deep	

Object Detection performance



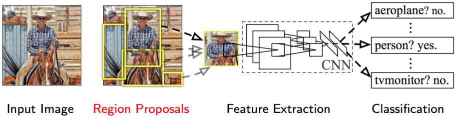
Source: *Ross Girshick*

Region CNN (RCNN)



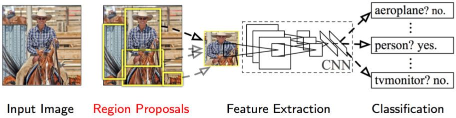
- Region Proposals: Selective Search
- Feature Network: Classification Networks
- Classifier: Linear Model

Region Proposals

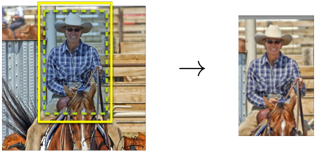


- Selective Search: Hierarchical grouping based on color, texture, size

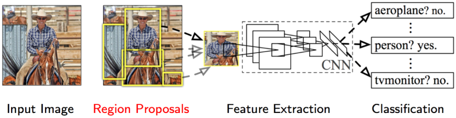
Region Proposals



- Selective Search: Hierarchical grouping based on color, texture, size
- Crop

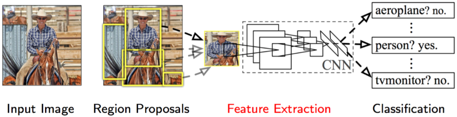


Region Proposals

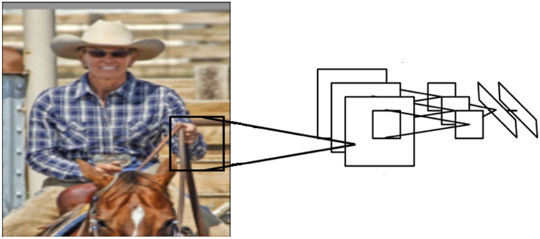


- Selective Search: Hierarchical grouping based on color, texture, size
- Crop
- Scale to a fixed size

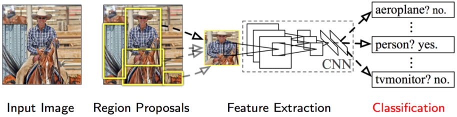
Feature Extraction



- Classification networks such as AlexNet/VGG-Net have been used
- Outputs from fc7 layer are taken as features corresponding to each proposal



Classification



- Linear Model with class dependent weights.

$$f_c\left(\mathbf{x}_{fc7}\right) = \mathbf{w}_c^\top \mathbf{x}_{fc7}$$

where,

$$\mathbf{x}_{fc7} = \text{fc7 features from the network}$$
$$c = \text{object class}$$

Bounding Box Regression

- Prediction of the 2D box, defined by its 2D location, (x, y) and dimensions, width (w) and height (h)
- For regression targets, x^*, y^*, w^*, h^*, we have

$$\frac{x^* - x}{w} = \mathbf{w}_{c,x}^\top \mathbf{x}_{pool5}$$
$$\frac{y^* - y}{w} = \mathbf{w}_{c,y}^\top \mathbf{x}_{pool5}$$
$$ln\left(\frac{w^*}{w}\right) = \mathbf{w}_{c,w}^\top \mathbf{x}_{pool5}$$
$$ln\left(\frac{h^*}{h}\right) = \mathbf{w}_{c,h}^\top \mathbf{x}_{pool5}$$

where, $\mathbf{x}_{pool5}$ are the features from the pool5 layer of the network.

Training

- Deep Network: Fine-tune classification networks with log loss

Training

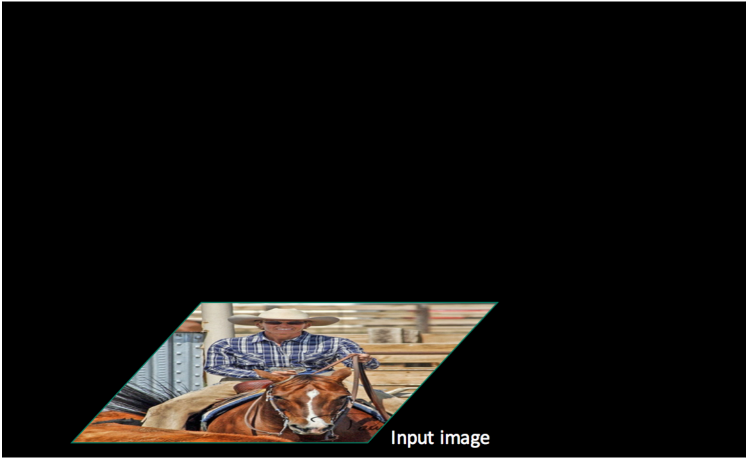
- Deep Network: Fine-tune classification networks with log loss
- Linear classification weights: Trained using hinge loss

Training

- Deep Network: Fine-tune classification networks with log loss
- Linear classification weights: Trained using hinge loss
- Regression weights: Trained using ridge regression

Inference Time = PropTime + NumProps*ConvTime + NumProps*fcTime

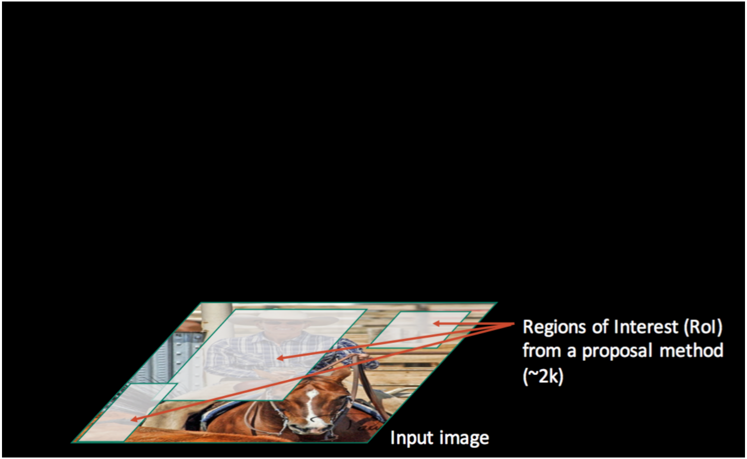
Inference Time = PropTime + NumProps*ConvTime + NumProps*fcTime



Source: *Ross Girshick*

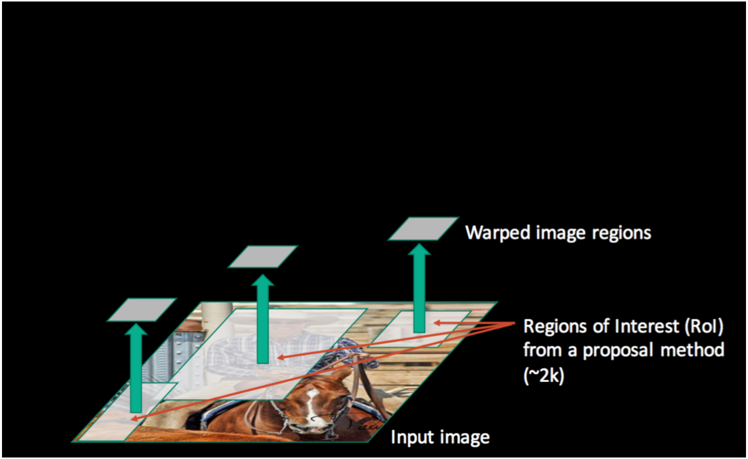
RCNN Review

Inference Time = PropTime + NumProps*ConvTime + NumProps*fcTime



Source: *Ross Girshick*

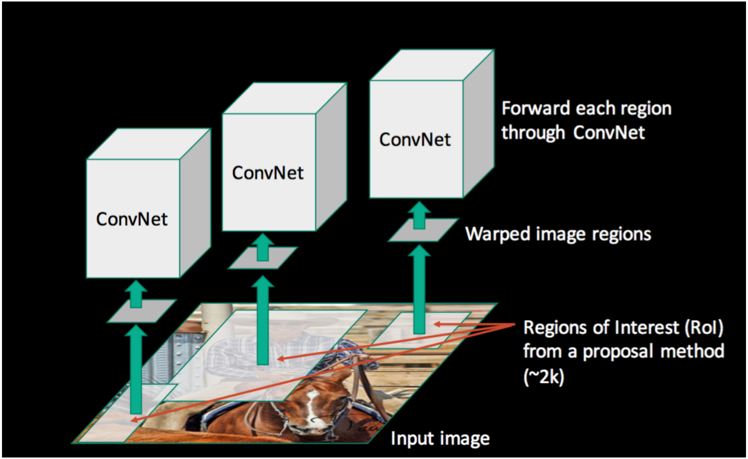
Inference Time = PropTime + NumProps*ConvTime + NumProps*fcTime



Source: *Ross Girshick*

RCNN Review

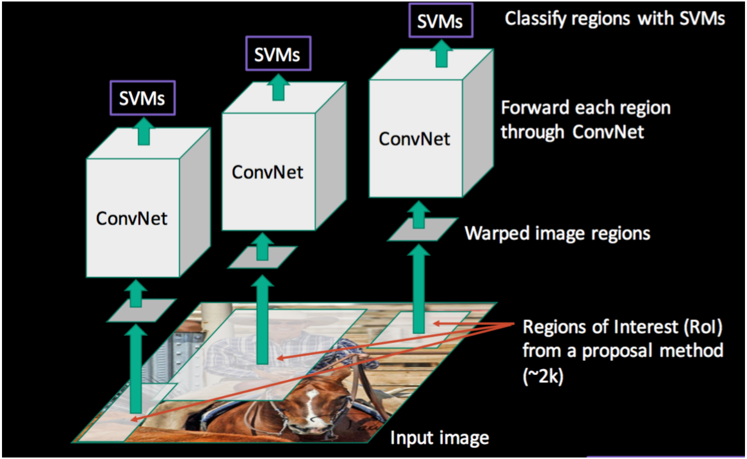
Inference Time = PropTime + NumProps*ConvTime + NumProps*fcTime



Source: *Ross Girshick*

RCNN Review

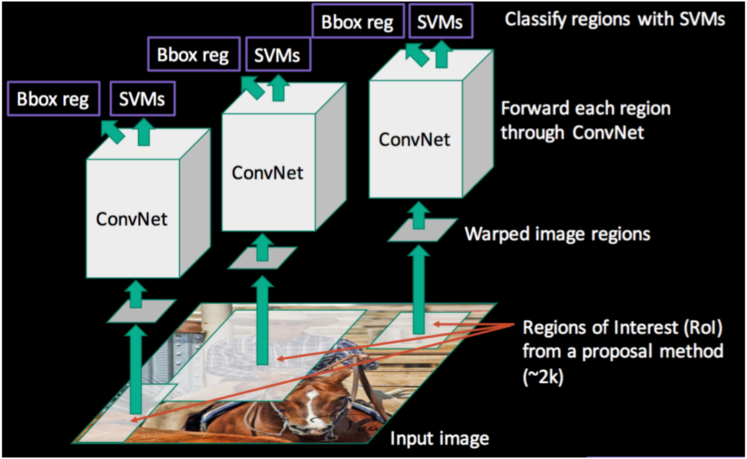
Inference Time = PropTime + NumProps*ConvTime + NumProps*fcTime



Source: *Ross Girshick*

RCNN Review

Inference Time = PropTime + NumProps*ConvTime + NumProps*fcTime



Source: *Ross Girshick*

Problems with RCNN

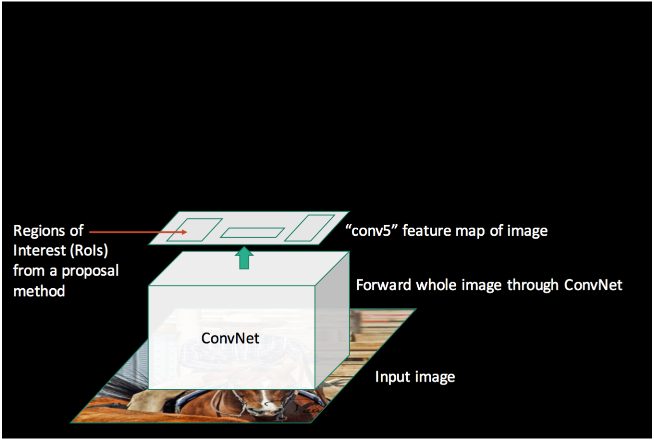
- Ad hoc training objectives
  - Fine tune network with softmax classifier (log loss)
  - Train post-hoc linear SVMs (hinge loss)
  - Train post-hoc bounding-box regressors (squared loss)
- Training ($\approx$3 days) and testing (47s per image) is slow[2].
- Takes a lot of disk space

Source: *Ross Girshick*

[2] Using VGG-Net

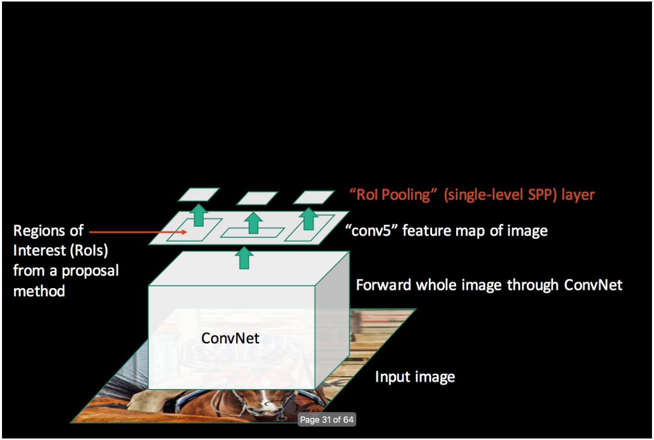
Fast RCNN

Forward Pass:



Source: *Ross Girshick*

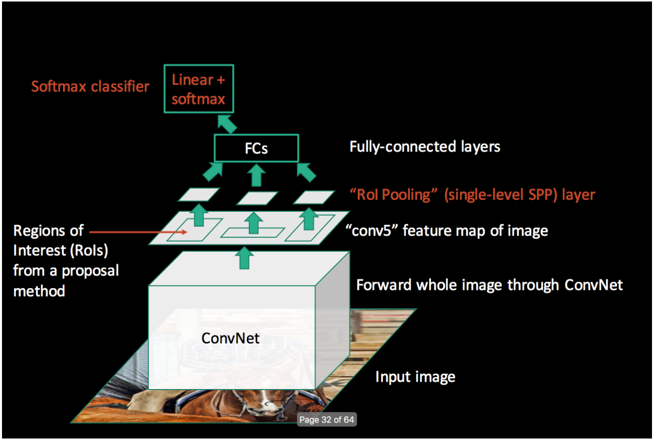
Forward Pass:



Source: *Ross Girshick*

Fast RCNN

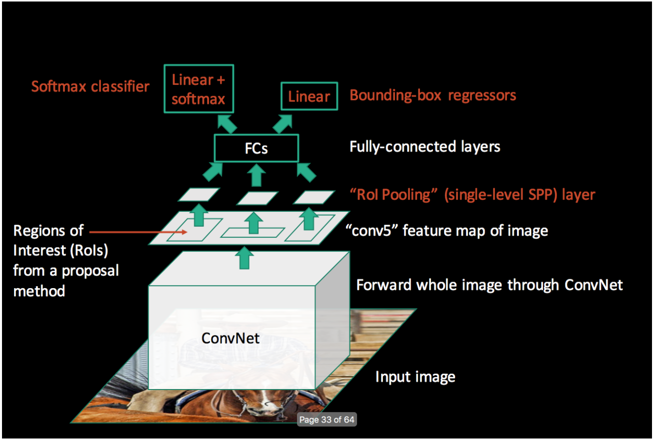
Forward Pass:



Source: *Ross Girshick*

Fast RCNN

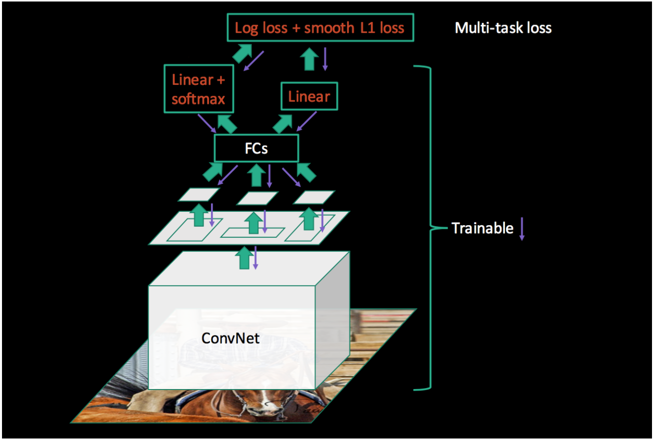
Forward Pass:



Source: *Ross Girshick*

Fast RCNN

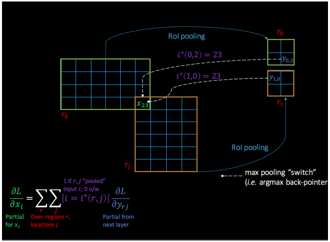
Backward Pass:



Source: *Ross Girshick*

Fast RCNN: Training

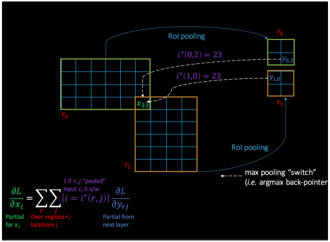
- Computing gradients for RoI pooling layer



Source: *Ross Girshick*

Fast RCNN: Training

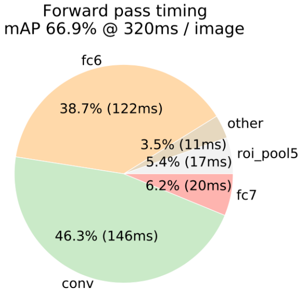
- Computing gradients for RoI pooling layer



Source: *Ross Girshick*

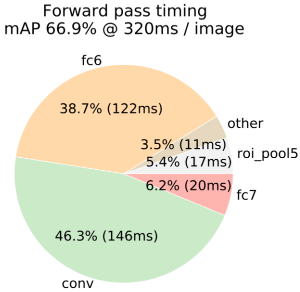
- Selecting mini-batches
    - Taking boxes from different images will lead to similar training time as RCNN
    - Instead take more boxes from a limited number of images.

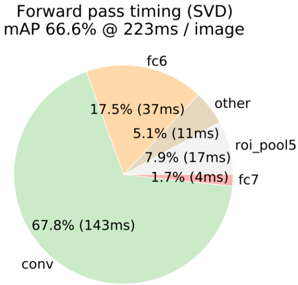
Fast RCNN: More Speedup



Source: *Ross Girshick*

Fast RCNN: More Speedup





Source: *Ross Girshick*

Fast RCNN: Main Results

Approach	Time
RCNN	PropTime + NumProp*ConvTime + NumProp*fcTime
Fast RCNN	PropTime + 1*ConvTime + NumProp*fcTime

[3] PASCAL VOC 07 test set

Fast RCNN: Main Results

Approach	Time
RCNN	PropTime + NumProp*ConvTime + NumProp*fcTime
Fast RCNN	PropTime + 1*ConvTime + NumProp*fcTime

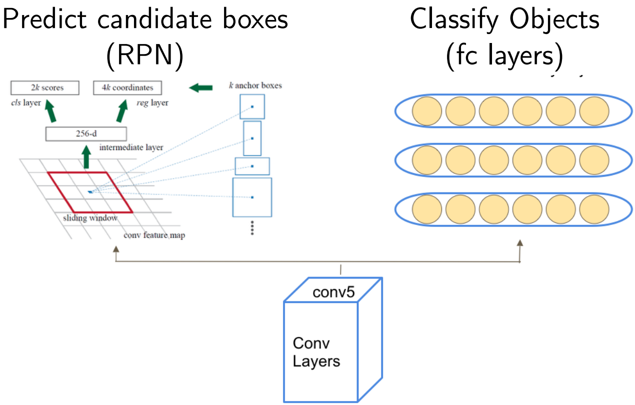
		RCNN	Fast RCNN (w/o SVD)	Fast RCNN (with SVD)
Training	Time (in hours)	84	9.5	9.5
	Speedup	1x	8.8x	8.8x
Testing	Time (in s/image)	47	0.32	0.22
	Speedup	1x	146x	214x
Performance[3]	AP	66.0 %	66.9%	66.6%

Testing time does not include time to compute region proposals.

- Selective Search: $\approx$ 2s
- Edge Boxes: 0.25s

[3] PASCAL VOC 07 test set

Faster RCNN

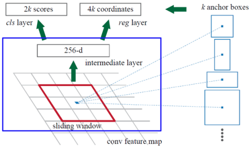


Source: *Andy Tsai*

RPN

After conv5 layer:

- Convolution layer to produce 256 dim vector for each anchor at each location

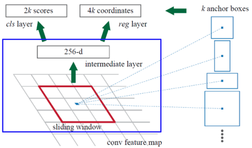


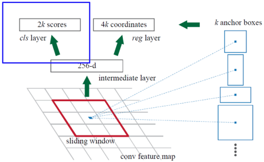
Source: *Andy Tsai*

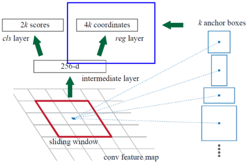
RPN

After conv5 layer:

- Convolution layer to produce 256 dim vector for each anchor at each location
- Convolution layer to produce objectness score and region bounds of anchors.

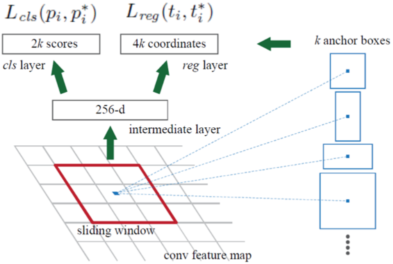






Source: *Andy Tsai*

RPN: Training



Source: *Andy Tsai*

$$L\left(\{p_i\}, \{t_i\}\right) = \frac{1}{N_{cls}} \sum_i L_{cls}\left(p_i, p_i^*\right) + \lambda \frac{1}{N_{reg}} \sum_i p_i * L_{cls}\left(t_i, t_i^*\right)$$

Faster RCNN: Training

- Finetune RPN from pre-trained ImageNet network.

Faster RCNN: Training

- Finetune RPN from pre-trained ImageNet network.
- Finetune fast RCNN from pre-trained ImageNet network using bounding boxes from step 1.

Faster RCNN: Training

- Finetune RPN from pre-trained ImageNet network.
- Finetune fast RCNN from pre-trained ImageNet network using bounding boxes from step 1.
- Keeping common convolutional layer parameters fixed from step 2, finetune RPN (post conv5 layers)

Faster RCNN: Training

- Finetune RPN from pre-trained ImageNet network.
- Finetune fast RCNN from pre-trained ImageNet network using bounding boxes from step 1.
- Keeping common convolutional layer parameters fixed from step 2, finetune RPN (post conv5 layers)
- Keeping common convolution layer parameters fixes from step 2, fine-tune fast RCNN fc layers.

Faster RCNN: Time comparisons

Approach	Time
RCNN	PropTime + NumProp*ConvTime + NumProp*fcTime
Fast RCNN	PropTime + 1*ConvTime + NumProp*fcTime
Faster RCNN	1*ConvTime + NumProp*fcTime

Results

	RCNN	Fast RCNN (w/o SVD)	Fast RCNN (with SVD)	Faster RCNN (w/o SVD)
Time (in s/image)	48.5	1.82	1.72	0.20
Speedup	1x	27x	28x	243x
AP[4]	66.0 %	66.9%	66.6%	69.9%
Num. Proposals	2500	2500	2500	300

[4] PASCAL VOC 07 test set

Approach	Data	mAP (in %)
Fast RCNN	12	65.7
	07+12	68.4
Faster RCNN	12	67.0
	07+12	70.4
	COCO+07+12	75.9
Faster RCNN (ResNet)	COCO+07+12	**83.8**

PASCAL VOC 2012 Test Set

Other Datasets

- MS COCO

Approach	**mAP** (in %)
Faster RCNN + ResNet	**58.8**
ION	52.9
FAIRCNN	51.9

Other Datasets

- MS COCO

Approach	mAP (in %)
Faster RCNN + ResNet	**58.8**
ION	52.9
FAIRCNN	51.9

- 3D Object Detection
  - Datasets: KITTI, NYUv2, SUN3D
  - Metric: AP at 3D IoU=0.25

Approach	mAP (in %)
Deep Sliding Shapes	72.3
RCNN3D	58.5

NYUv2

Object Detection for Autonomous Driving

Approach	Car (in %)	Pedestrian (in %)	Cyclist (in %)
3DOP[5]	**88.64**	**67.47**	**68.94**
3DOP-Monocular	88.09	66.34	67.03
Faster RCNN	81.84	65.90	63.35

KITTI

- IoU threshold for Cars = 0.7

[5] Chen *et al.*, 3D Object Proposals for Accurate Object Class Detection, 2015.

Object Detection for Autonomous Driving

Approach	**Car (in %)**	**Pedestrian (in %)**	**Cyclist (in %)**
3DOP[5]	**88.64**	**67.47**	**68.94**
3DOP-Monocular	88.09	66.34	67.03
Faster RCNN	81.84	65.90	63.35

KITTI

- IoU threshold for Cars = 0.7

[5] Chen *et al.*, 3D Object Proposals for Accurate Object Class Detection, 2015.

3DOP

Left Image

Right Image

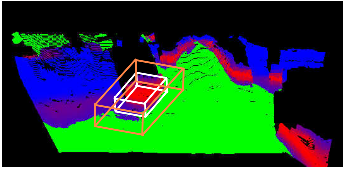
Stereo

Left Image

Right Image

Stereo

- Proposal Generation



■ : Occupancy

■ : Free Space

■ : Road plane

Blue → Red: Increasing height

3DOP

Left Image

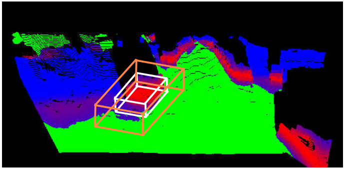
Right Image

Stereo

- Proposal Generation

: Occupancy

: Free Space



: Road plane

Blue $\rightarrow$ Red: Increasing height

- Object Detection: Fast RCNN Network

References

- Pre-Deep Era:
  - Felzenszwalb *et al.*, Discriminatively trained deformable part models, 2010.
  - Fidler *et al.*, Bottom-up Segmentation for Top-down Detection, 2013.
- RCNN and related ideas:
  - Girshick *et al.*, Region-based Convolutional Networks for Accurate Object Detection and Semantic Segmentation, 2014.
  - Zhu *et al.*, segDeepM: Exploiting Segmentation and Context in Deep Neural Networks for Object Detection, 2015.
- Fast RCNN and related ideas:
  - He *et al.*, Spatial Pyramid Pooling in Deep Convolutional Networks for Visual Recognition, 2014.
  - Girshick, Ross, Fast R-CNN, 2015.
- Faster RCNN and related ideas:
  - Szegedy *et al.*, Scalable, High-Quality Object Detection, 2014.
  - Ren *et al.*, Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks, 2015.
- Bell *et al.*, Inside-Outside Net: Detecting Objects in Context with Skip Pooling and Recurrent Neural Networks, 2015.

References

- 3D Object Detection
  - Gupta *et al.*, Learning Rich Features from RGB-D Images for Object Detection and Segmentation, 2014.
  - Gupta *et al.*, Inferring 3D Object Pose in RGB-D Images, 2015.
  - Chen *et al.*, 3D Object Proposals for Accurate Object Class Detection, 2015.
  - Song and Xiao, Deep Sliding Shapes for Amodal 3D Object Detection in RGB-D Images, 2015.
- Joint Detection and Segmentation
  - Hariharan *et al.*, Simultaneous Detection and Segmentation, 2014.
  - Hariharan *et al.*, Hypercolumns for Object Segmentation and Fine-grained Localization, 2014 .
  - Dai *et al.*, Instance-aware Semantic Segmentation via Multi-task Network Cascades, 2015.
- Segmentation Aware Detection
  - Zhu *et al.*, segDeepM: Exploiting Segmentation and Context in Deep Neural Networks for Object Detection, 2015.
  - Gidaris *et al.*, Object detection via a multi-region & semantic segmentation-aware CNN model, 2015.