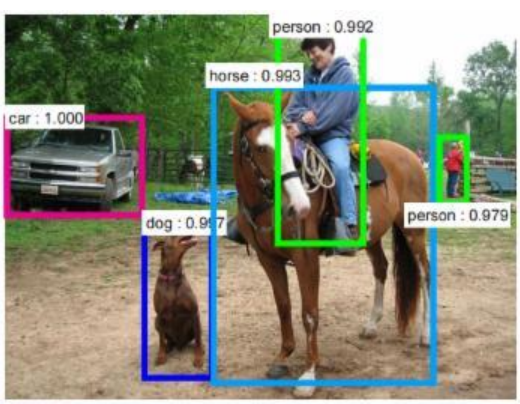
Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks

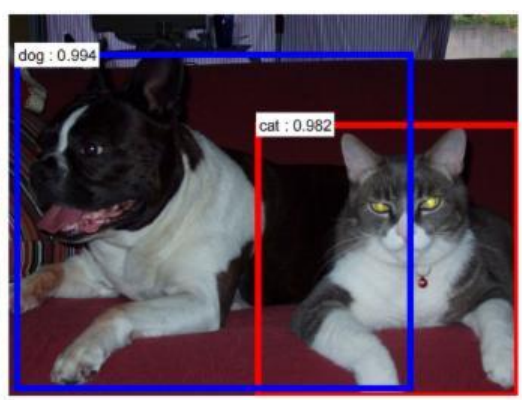
Shaoqing Ren Kaiming He
Ross Girshick Jian Sun

Present by:
Yixin Yang
Mingdong Wang

1

Object Detection



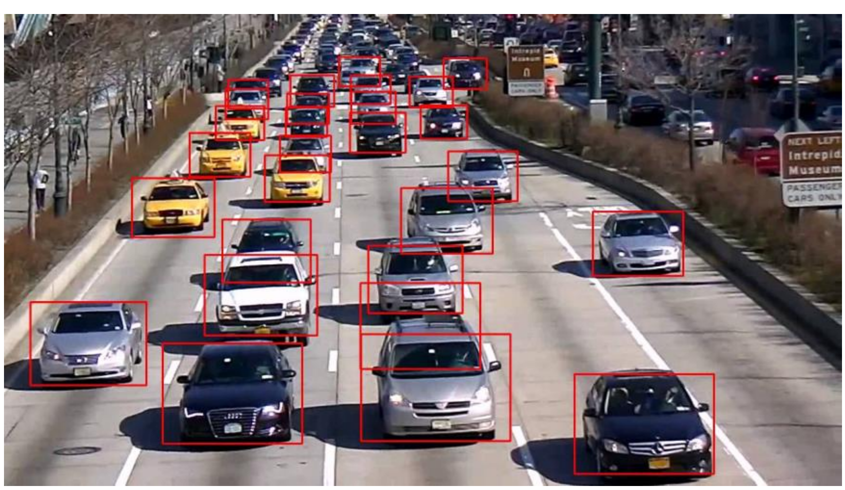


2

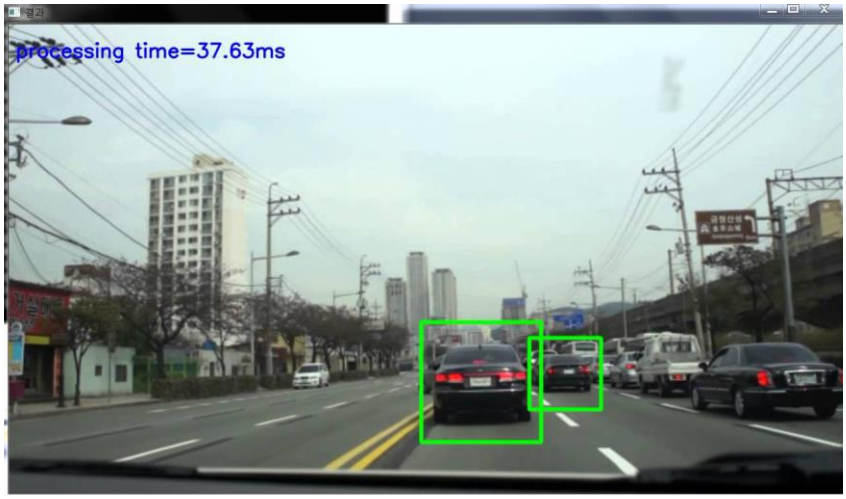
Applications

- Basic task for image understanding
  - Output only bounding boxes
- Enables many downstream applications
  - Almost everything requires detection!
  - Vehicle – traffic, accident etc.
  - Face – security, APPs
  - …

Applications

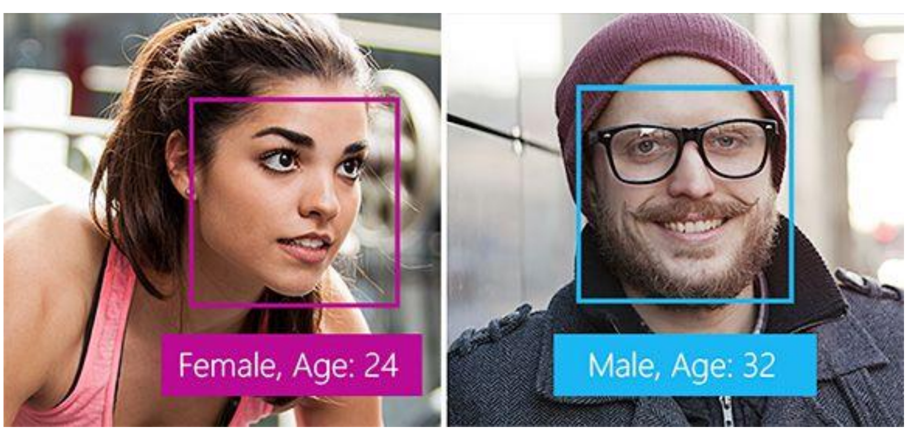


Applications



Applications

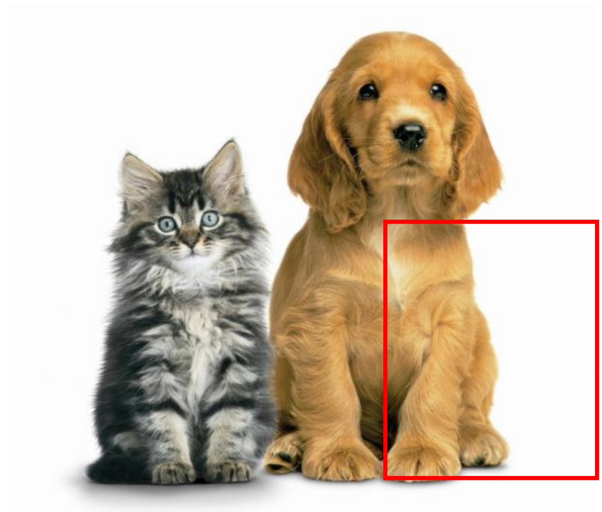
- http://how-old.net/ By Microsoft



Basic Idea: Detection as classification

- Select a region (by any means)
- Classify this region

- Example
  - Cat? No
  - Dog? No

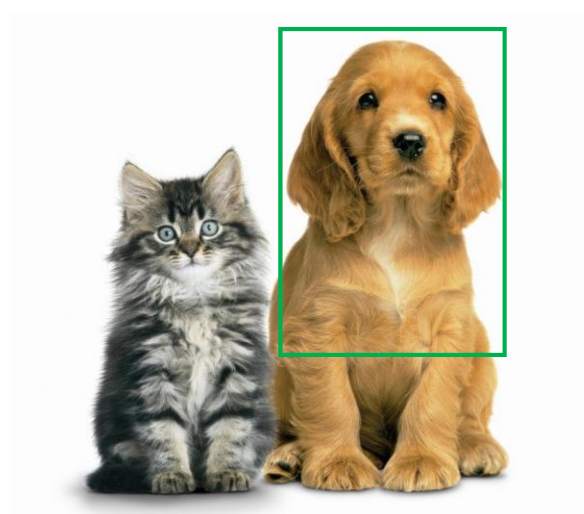


7

Basic Idea: Detection as classification

- Select a region (by any means)
- Classify this region

- Example
  - Cat? No
  - Dog? Yes

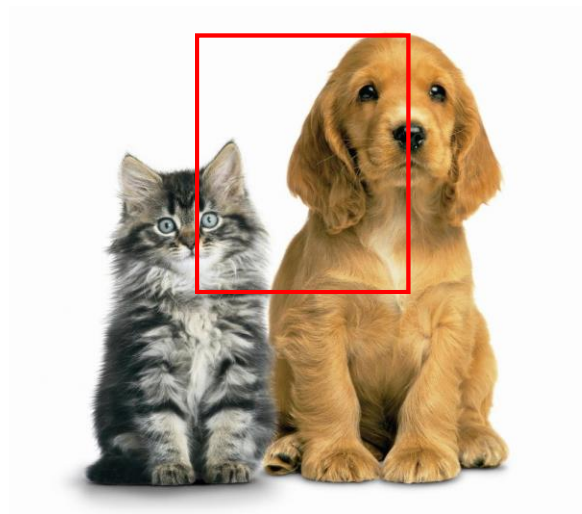


8

Basic Idea: Detection as classification

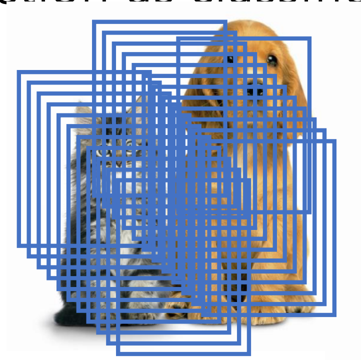
- Select a region (by any means)
- Classify this region

- Example
  - Cat? No
  - Dog? No

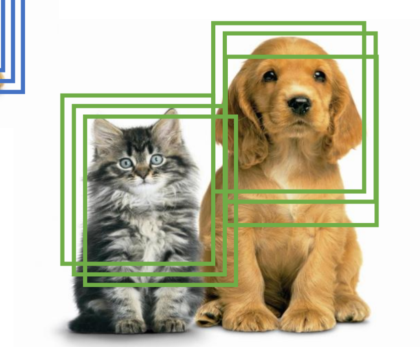


Basic Idea: Detection as classification

- Problem: Too many proposals
  - Slow
- Solution: Only look at possible regions



- Region proposal
  - Find regions that are likely to have object in it
  - Class-invariant



Region Proposal

- Selective Search
  - Feature-based
  - Bottom-up
  - Merging

Uijlings et al, “Selective Search for Object Recognition”, IJCV 2013

Region Proposal

- Many other solutions
- EdgeBoxes is the best in practice

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	·	·	·

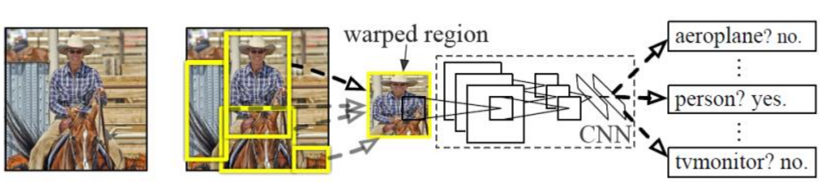
Hosang et al, “What makes for effective detection proposals?”, PAMI 2015

Outline

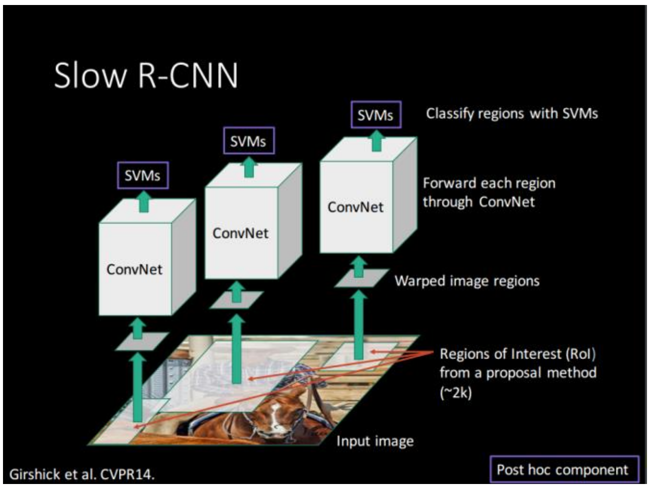
- Previous work
  - Region-based Convolutional Neural Network (R-CNN)
  - Spatial Pyramid Pooling Network (SPP-Net)
  - Fast R-CNN
- Faster R-CNN
  - Region Proposal Network (RPN)
  - Detection
- Experiments

R-CNN

- Region Proposals + CNN
- Three Steps:
  - Use Selective Search to get region proposals (~2k)
  - Warp every region proposal to 227x227, then extract feature by CNN
  - Classify: Support Vector Machine (SVM)



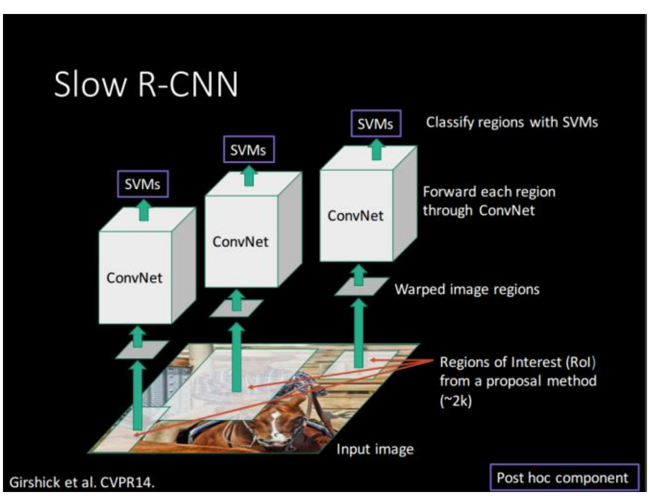
R-CNN



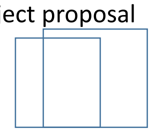
- Warp Image:
  - The inputs of CNN should be the same size
- Training:
  - Pre-train CNN for image classification
  - Fine-tune CNN for object detection
  - Train linear predictor for object detection

15

R-CNN



- What is wrong with R-CNN?
  - Training and testing is slow
  - Takes a lot of disk space
- Reason
  - a ConvNet forward pass for each object proposal

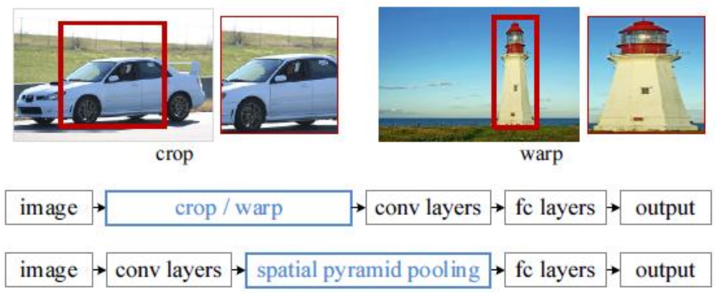


- How to solve?
  - 1 CNN for whole image -> Spatial Pyramid Pooling Net (Spp-net)

16

Spatial Pyramid Pooling Net

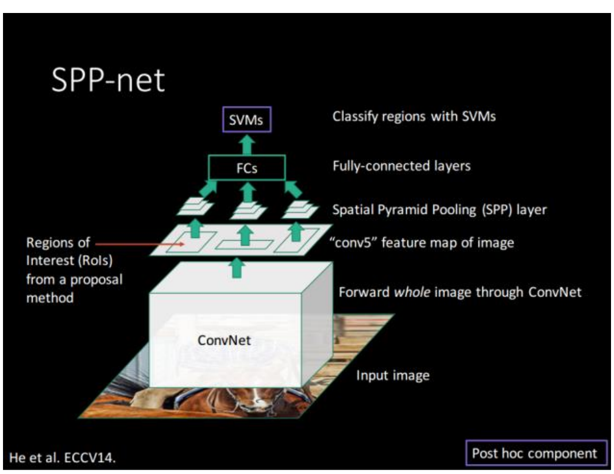
- Much similar with R-CNN, but only 1 CNN for the whole image



- In fact, it is the fully-connect layer that needs the fix-size input

17

Spatial Pyramid Pooling Net



- 1 CNN for the input image and get the feature map
- Add a SPP layer after the last convolutional layer

18

Spatial Pyramid Pooling Net

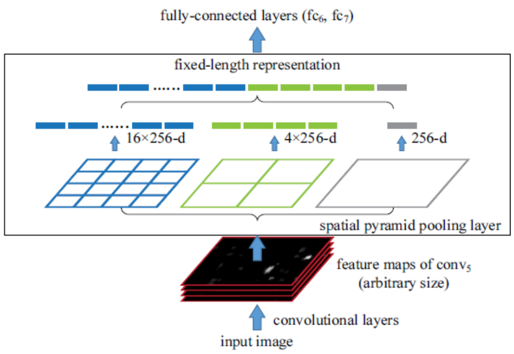


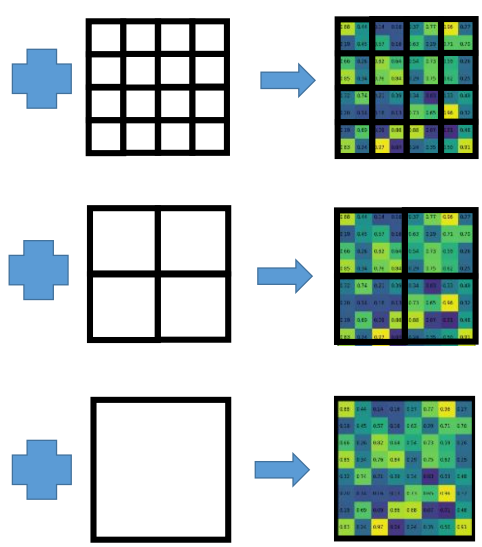
Figure 3: A network structure with a **spatial pyramid pooling layer**. Here 256 is the filter number of the $conv_5$ layer, and $conv_5$ is the last convolutional layer.

- SPP layer Example
  - The input of the spp layer can by arbitrary size
  - The output size is 21x256 no matter the input size

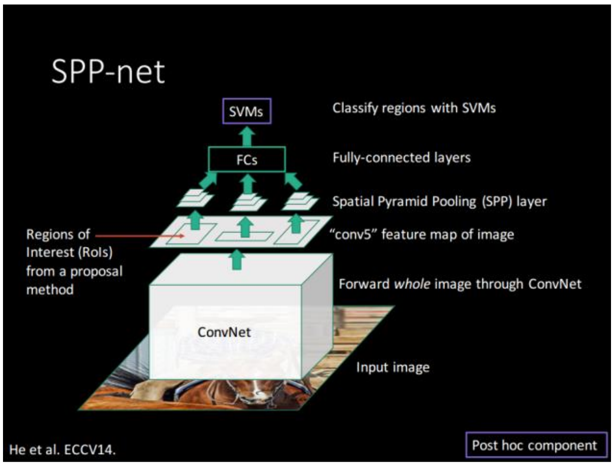
Spatial Pyramid Pooling Net

input

0.88	0.44	0.14	0.16	0.37	0.77	0.96	0.27
0.19	0.45	0.57	0.16	0.63	0.29	0.71	0.70
0.66	0.26	0.82	0.64	0.54	0.73	0.59	0.26
0.85	0.34	0.76	0.84	0.29	0.75	0.62	0.25
0.32	0.74	0.21	0.39	0.34	0.03	0.33	0.48
0.20	0.14	0.16	0.13	0.73	0.65	0.96	0.32
0.19	0.69	0.09	0.86	0.88	0.07	0.01	0.48
0.83	0.24	0.97	0.04	0.24	0.35	0.50	0.91



Spatial Pyramid Pooling Net

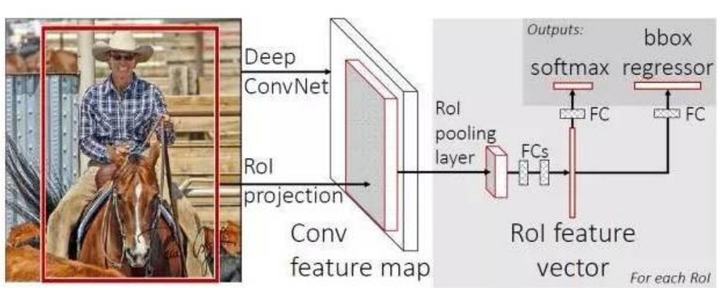


- The improvement of SPP-net
  - Makes training and testing fast
- What is wrong with SPP-net?
  - Cannot update parameters below SPP layer during training
- How to solve?
  - Use Softmax classifer instead of SVM -> Fast R-CNN

21

Fast R-CNN

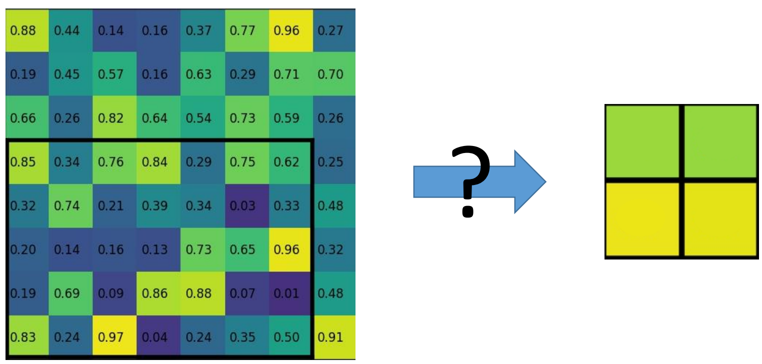
- There are two differences in fast R-CNN
  - Add a Region of Interest (RoI) pooling layer after the last convolutional layer
  - Two output vectors per RoI: softmax probabilities and bouding-box regression on offset. The architecture is trained end-to-end with a multi-task loss.



22

Fast R-CNN

- RoI pooling layer
  - Use max pooling to convert the features insides any valid region of interest into a small feature map with a fixed spatial extent of HxW

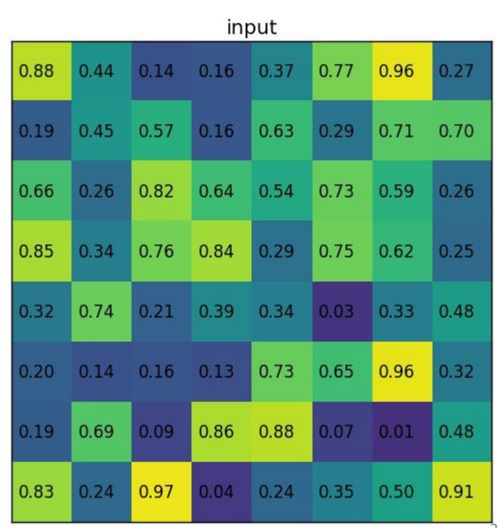


Fast R-CNN

- RoI pooling layer (example)
  - Input:

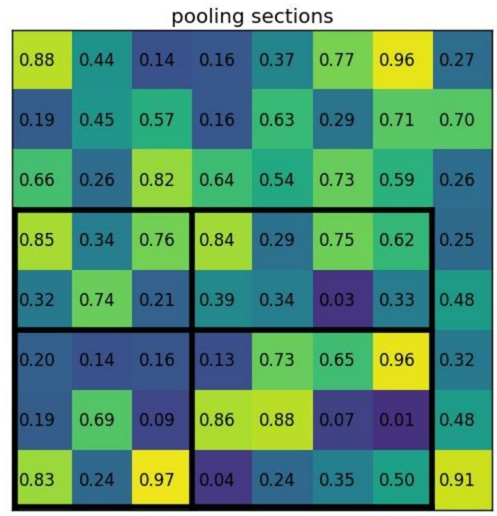
  a single 8×8 feature map and a region proposal (arbitrary size)
  - Output:

  2x2 feature map for future use



Fast R-CNN

- RoI pooling layer
  - Pooling sections
  - h/H && w/W, in this case:
    5/2 && 7/2
  - Notice that the size of the region of interest doesn’t have to be perfectly divisible by the number of pooling sections

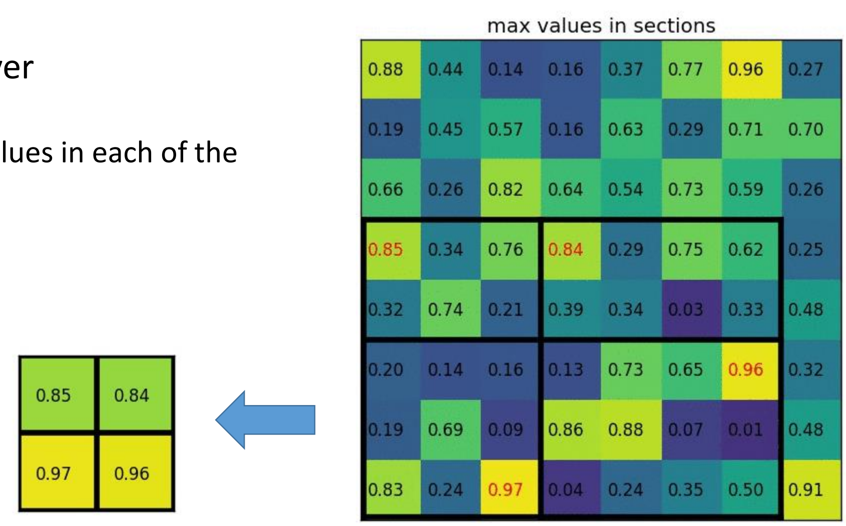


25

Fast R-CNN

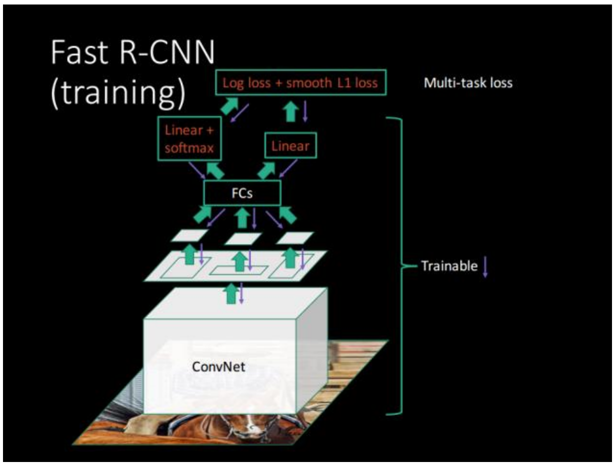
- RoI pooling layer
  - Max pooling:

  Get the max values in each of the sections



26

Fast R-CNN



- Two output layers
  - discrete probability distribution (per RoI), $p = (p_0, p_1 \ldots, p_K)$ over K+1 categories
  - bounding-box regression offsets, $t^k = (t_x^k, t_y^k, t_w^k, t_h^k)$, for each of the K object classes

27

Fast R-CNN

- Multi-task loss

$$L(p, u, t^u, v) = L_{\text{cls}}(p, u) + \lambda[u \geq 1] L_{\text{loc}}(t^u, v)$$

u is the ground-truth class

v is ground-truth bounding-box regression target

Where $L_{\text{cls}}(p, u) = -\log p_u$ is the log loss for true class u

28

Fast R-CNN

- Multi-task loss

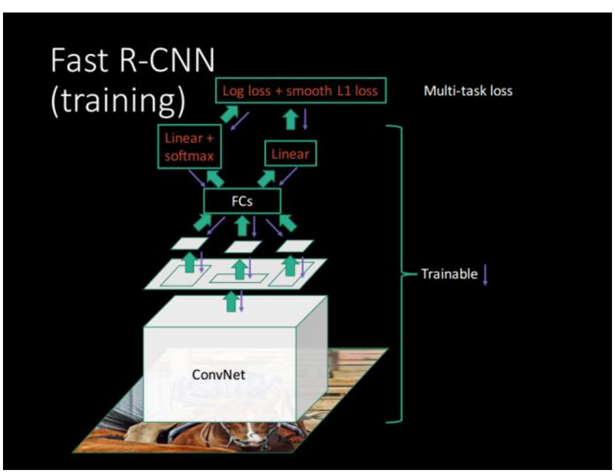
$$L(p, u, t^u, v) = L_{\text{cls}}(p, u) + \lambda[u \geq 1]L_{\text{loc}}(t^u, v)$$

Where $$L_{\text{loc}}(t^u, v) = \sum_{i \in \{\text{x,y,w,h}\}} \text{smooth}_{L_1}(t_i^u - v_i)$$

$$\text{smooth}_{L_1}(x) = \begin{cases} 0.5x^2 & \text{if } |x| < 1 \\ |x| - 0.5 & \text{otherwise} \end{cases}$$

29

Fast R-CNN



- What is wrong with Fast R-CNN
  - Use Selective Search to get the regions proposal, which is time consuming.
- How to solve?
  - Use Convolutional Neural Network to generate region proposal -> Faster R-CNN

30

Faster R-CNN

- Object proposal is the bottleneck
  - Selective search ~ 2s
  - EdgeBoxes ~ 0.2s
  - As much as the detection network
- Feature map used by detector can also used for generating proposals
- Why not also use CNN?
  - Better performance
  - Sharing computation

Region Proposal Network (RPN)

- Input
  - Image (feature map) of any size
- Output
  - A set of rectangular object proposals
  - Each with an objectness score

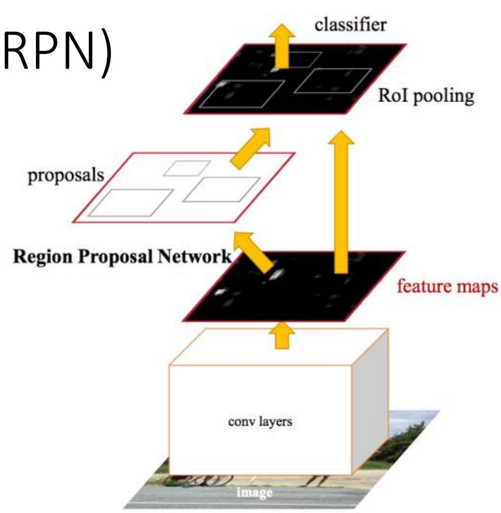


Figure 2: Faster R-CNN is a single, unified network for object detection. The RPN module serves as the 'attention' of this unified network.

RPN

- Insert the RPN after the last conv layer
- RPN will produce region proposal directly, serves as 'attention'
- After RPN, use RoI pooling and bounding-box regressor just like fast R-CNN

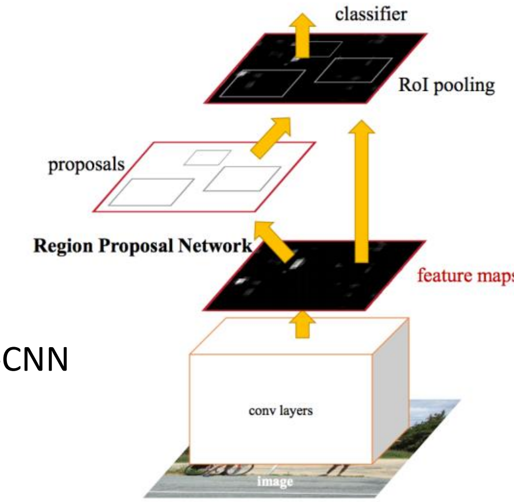
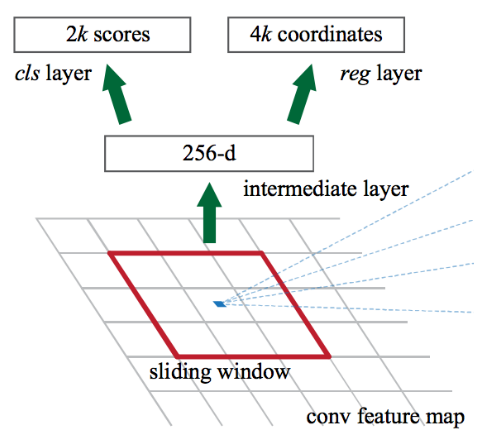


Figure 2: Faster R-CNN is a single, unified network for object detection. The RPN module serves as the 'attention' of this unified network.

33

RPN

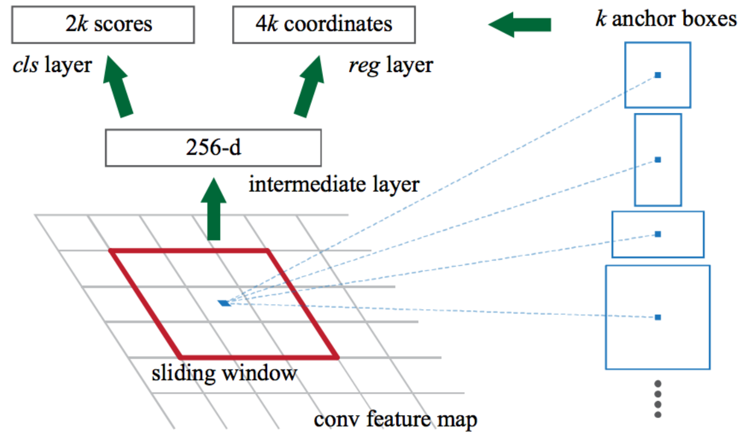
- Slide a network on the feature map
- For each nxn (n=3) window use a conv kernel to produce another feature map
- Now we have a KxKx256 feature map
- Each pixel in this feature map is an anchor



34

RPN: Anchor

- For each anchor
- Propose k anchor boxes
  - Related to this anchor
  - Prior assumption
- For each box, regress
  - Objectness score
  - Coordinates



35

Loss function

- L_cls: classification error
- L_reg: bbox coords regression error
- p_i/p_i*: predicted/ground truth classification
- t_i/t_i*: predicted/ground truth bbox coords

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

36

RPN training

- End to end trainable by SGD
  - Stochastic gradient descent
- Each mini-batch arises from a single image
- Randomly sample 256 anchors in the image
  - Try to make pos/neg ~ 1
  - Otherwise negative will dominate -> biased

37

RPN & Fast RCNN detector

- If trained separately, they will have different conv params
- Try to share conv layers
- How to write the update formula?

38

Joint training

- Exact derivatives are hard to get
  - Derivatives of RoI pooling layer w.r.t. box coords
- Alternating
  - Train RPN, then use proposals to train detector, then train RPN…
- Approximate
  - Ignore the derivatives of bbox coords as if they are fixed
- 4-step training
  - Train RPN on pretrained ImageNet network
  - Train a detector with RPN proposals but using different conv params
  - Use detector params to initialize RPN network, but fix shared layers, finetune RPN
  - Fix shared layers, finetune detector

Other details

- Anchor box area & ratio
  - Select without carefully tested
- Cross image boundaries handling
  - Ignore when training
  - Crop when testing
- RPN proposals overlapping
  - NMS to reduce proposals

Experiments

Table 2: Detection results on **PASCAL VOC 2007 test set** (trained on VOC 2007 trainval). The detectors are Fast R-CNN with ZF, but using various proposal methods for training and testing.

train-time region proposals		test-time region proposals		
method	# boxes	method	# proposals	mAP (%)
SS	2000	SS	2000	58.7
EB	2000	EB	2000	58.6
RPN+ZF, shared	2000	RPN+ZF, shared	300	**59.9**
ablation experiments follow below				
RPN+ZF, unshared	2000	RPN+ZF, unshared	300	58.7
SS	2000	RPN+ZF	100	55.1
SS	2000	RPN+ZF	300	56.8
SS	2000	RPN+ZF	1000	56.3
SS	2000	RPN+ZF (no NMS)	6000	55.2
SS	2000	RPN+ZF (no *cls*)	100	44.6
SS	2000	RPN+ZF (no *cls*)	300	51.4
SS	2000	RPN+ZF (no *cls*)	1000	55.8
SS	2000	RPN+ZF (no *reg*)	300	52.1
SS	2000	RPN+ZF (no *reg*)	1000	51.3
SS	2000	RPN+VGG	300	59.2

Experiments

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**

Table 4: Detection results on **PASCAL VOC 2012 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+test and VOC 2012 trainval. For RPN, the train-time proposals for Fast R-CNN are 2000. †: http://host.robots.ox.ac.uk:8080/anonymous/HZJTQA.html. ‡: http://host.robots.ox.ac.uk:8080/anonymous/YNPLXB.html. §: http://host.robots.ox.ac.uk:8080/anonymous/XEDH10.html.

method	# proposals	data	mAP (%)
SS	2000	12	65.7
SS	2000	07++12	68.4
RPN+VGG, shared†	300	12	67.0
RPN+VGG, shared‡	300	07++12	**70.4**
RPN+VGG, shared§	300	COCO+07++12	**75.9**

Experiments

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**

43

Experiments

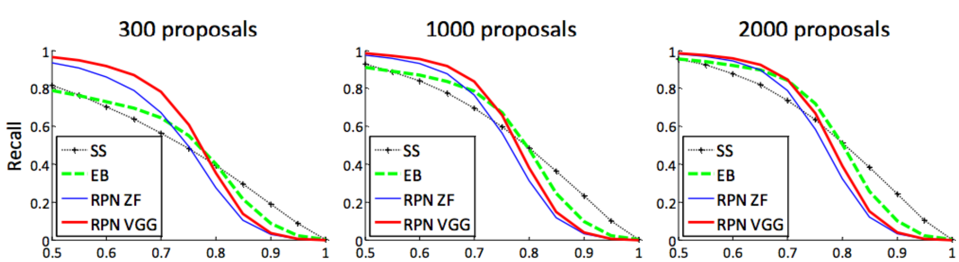


Figure 4: Recall *vs.* IoU overlap ratio on the PASCAL VOC 2007 test set.

44

Follow Up

- YOLO (You Only Look Once: Unified, Real-Time Object)
  - Instead of using region proposal + classification, doing the regression of the position and class of bounding box
  - Convert the objection detection to a Regression problem
- SDD (Single Shot MultiBox Detector)
- YOLO2
- ……

Summary

- Find a variable number of objects by classifying image regions
- R-CNN
  - Selective Search + CNN + SVM
  - ~30s / img
- Fast RCNN
  - Swap order of convolutions and region extraction
  - 2 s / img
- Faster RCNN
  - Compute region proposals within the network
  - 0.2 s / img

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

Thanks for listening
Q && A