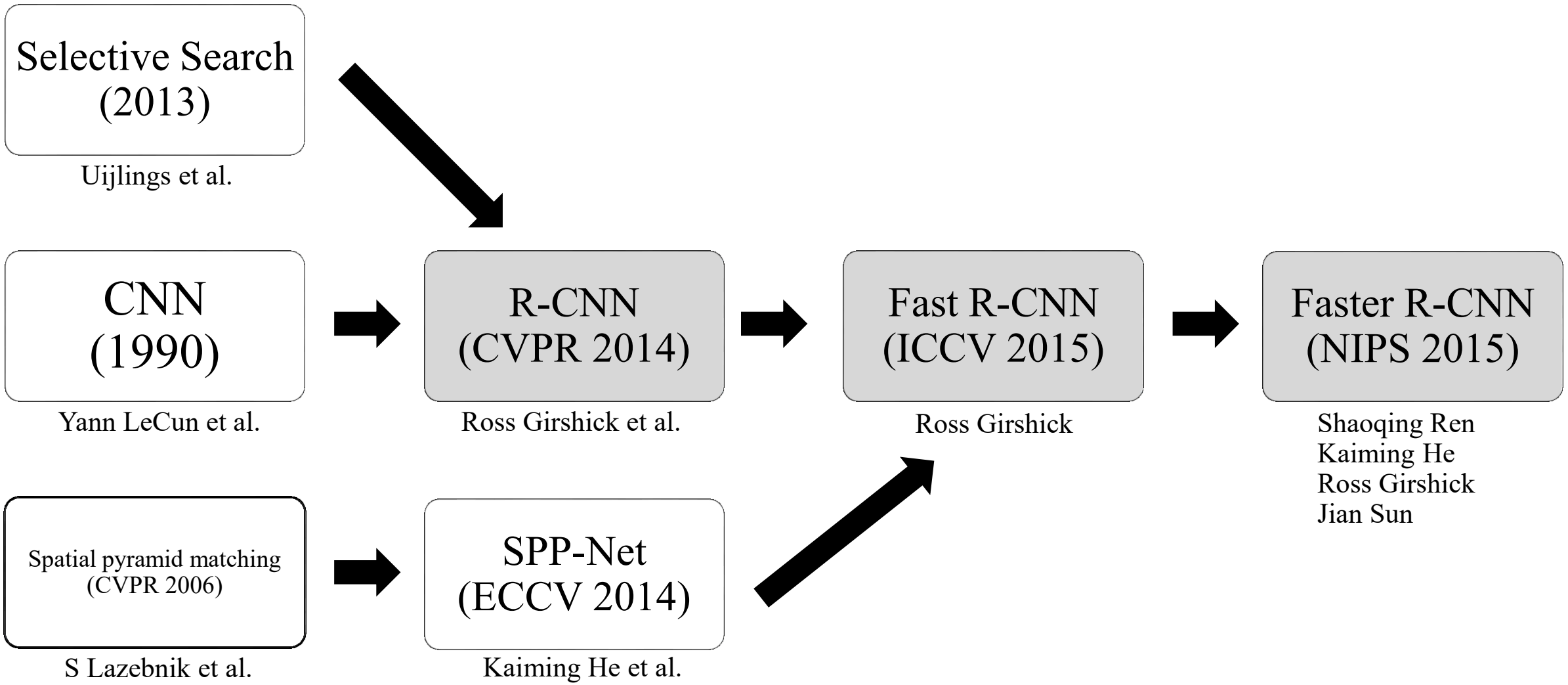
Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks (NIPS 2015)

Shaoqing Ren, Kaiming He, Ross Girshick, Jian Sun

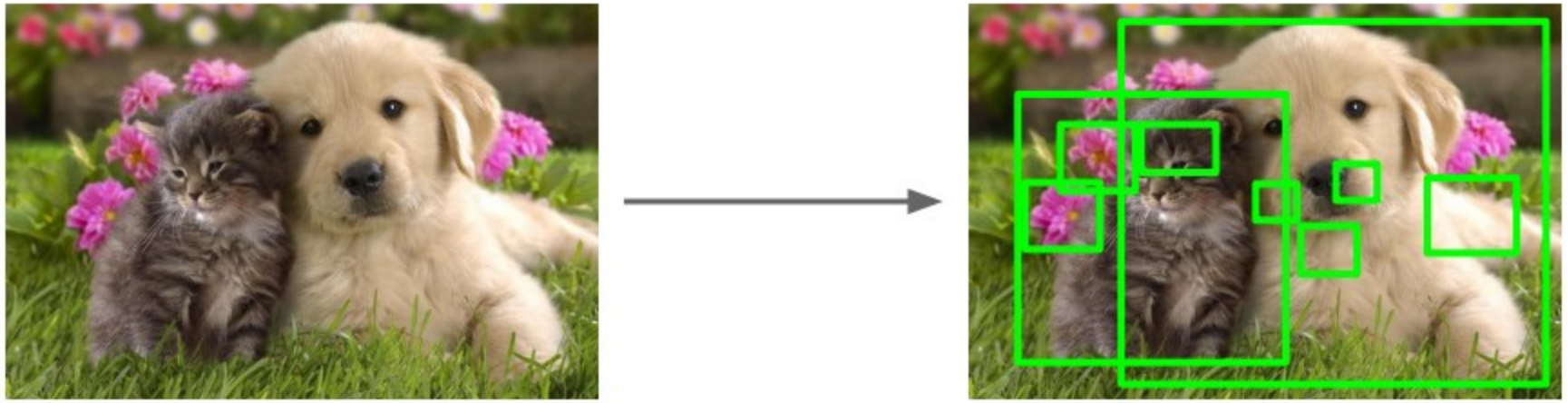
Microsoft Research

CNN-based object detection papers



Region proposals

- Find “blobby” image regions that are likely to contain objects
- Relatively fast to run; e.g. Selective Search gives 1000 region proposals in a few seconds on CPU



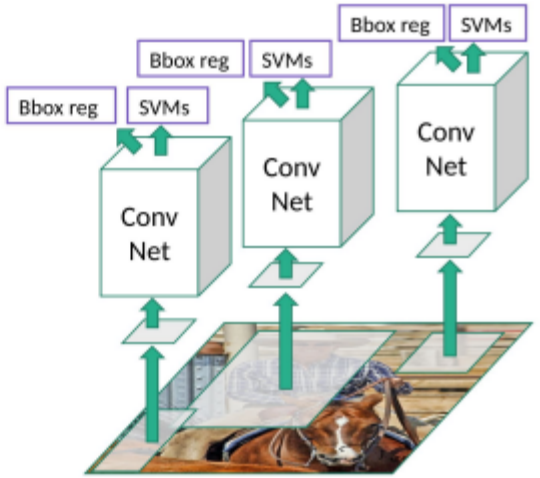
Alexe et al, “Measuring the objectness of image windows”, TPAMI 2012
Uijlings et al, “Selective Search for Object Recognition”, IJCV 2013
Cheng et al, “BING: Binarized normed gradients for objectness estimation at 300fps”, CVPR 2014
Zitnick and Dollar, “Edge boxes: Locating object proposals from edges”, ECCV 2014

http://cs231n.stanford.edu/slides/2017/cs231n_2017_lecture11.pdf

R-CNN

R-CNN: Problems

- Ad hoc training objectives
  - Fine-tune network with softmax classifier (log loss)
  - Train post-hoc linear SVMs (hinge loss)
  - Train post-hoc bounding-box regressions (least squares)
- Training is slow (84h), takes a lot of disk space
- Inference (detection) is slow
  - 47s / image with VGG16 [Simonyan & Zisserman. ICLR15]
  - Fixed by SPP-net [He et al. ECCV14]



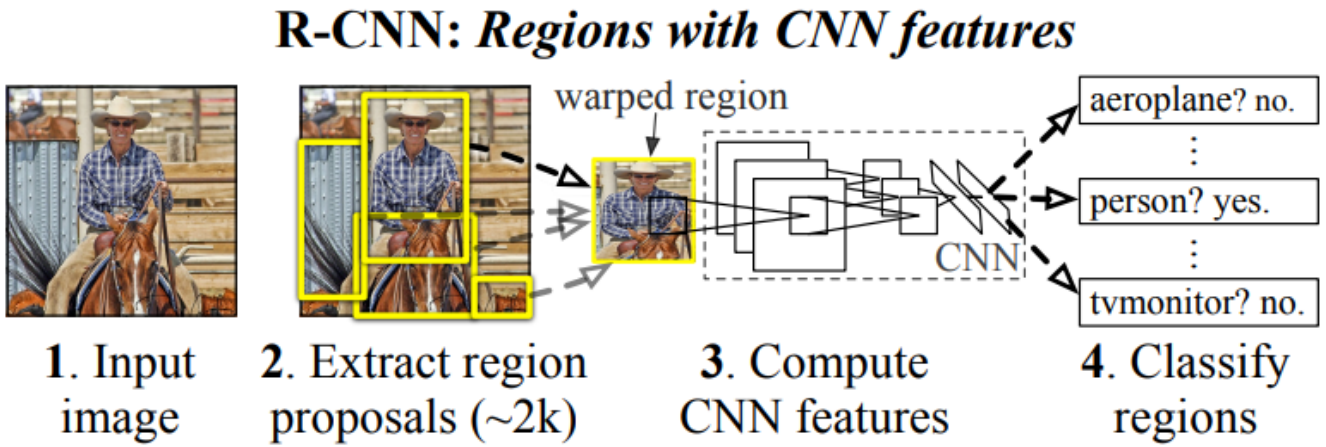
Girshick et al, "Rich feature hierarchies for accurate object detection and semantic segmentation", CVPR 2014.
Slide copyright Ross Girshick, 2015; source. Reproduced with permission.

http://cs231n.stanford.edu/slides/2017/cs231n_2017_lecture11.pdf

R-CNN vs. Fast R-CNN

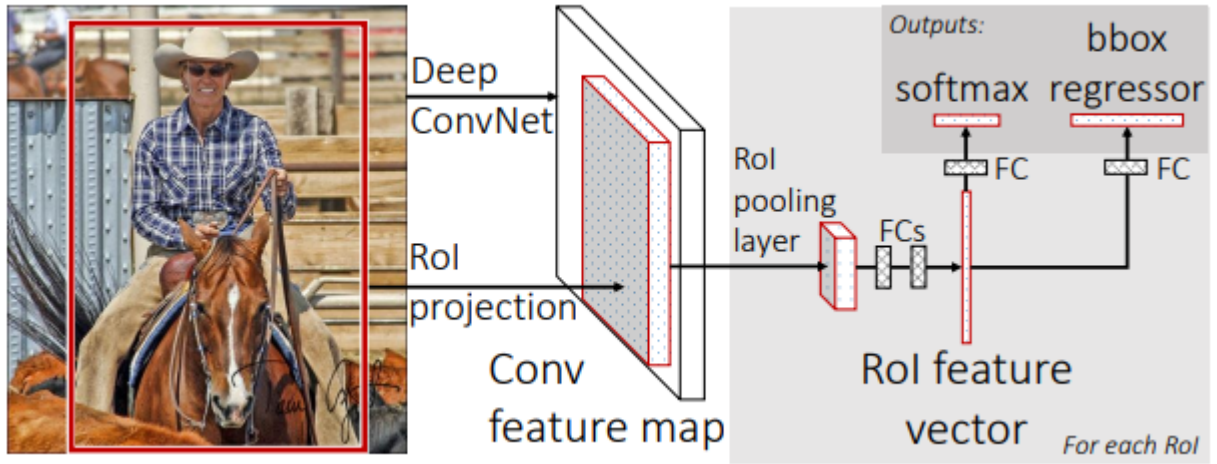
R-CNN
(CVPR 2014)

Ross Girshick et al.



Fast R-CNN
(ICCV 2015)

Ross Girshick

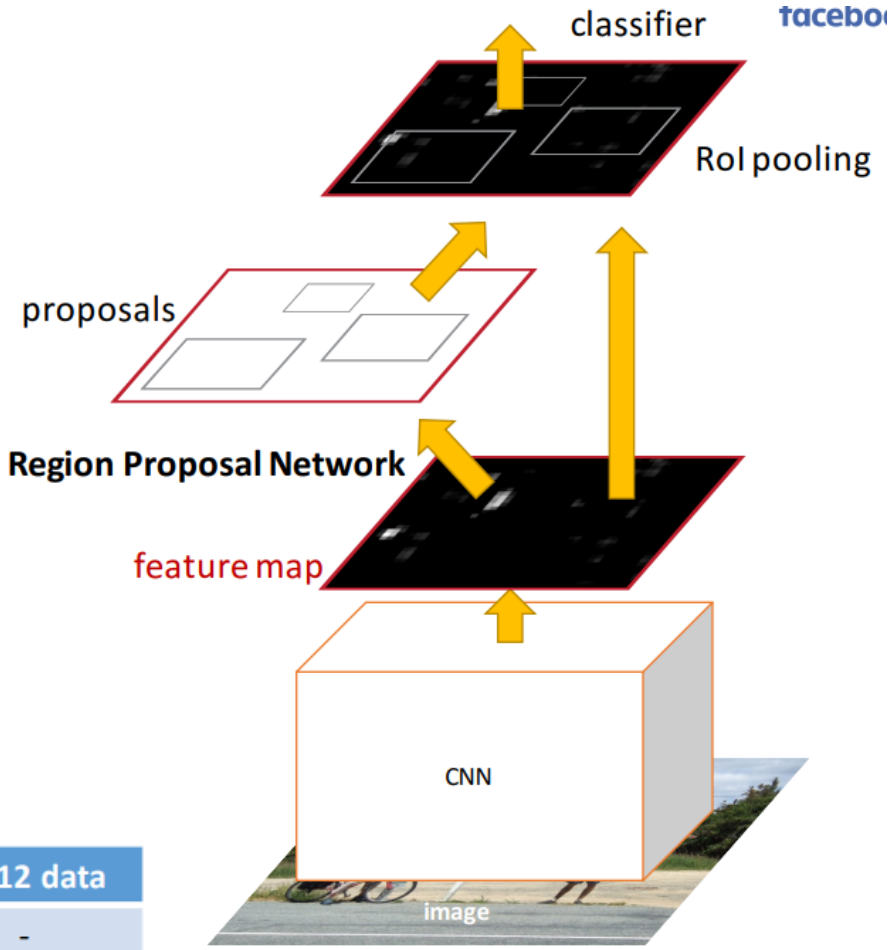


http://mp7.watson.ibm.com/ICCV2015/slides/iccv15_tutorial_training_rbg.pdf

Faster R-CNN

What is Faster R-CNN?

- Presented in Kaiming's section
- Review:
  Faster R-CNN = Fast R-CNN + Region Proposal Networks
  - Does *not* depend on an external region proposal algorithm
  - Does object detection in a single forward pass



system	time	07 data	07+12 data
R-CNN	~50s	66.0	-
Fast R-CNN	~2s	66.9	70.0
Faster R-CNN	198ms	**69.9**	**73.2**

detection mAP on PASCAL VOC 2007, with VGG-16 pre-trained on ImageNet

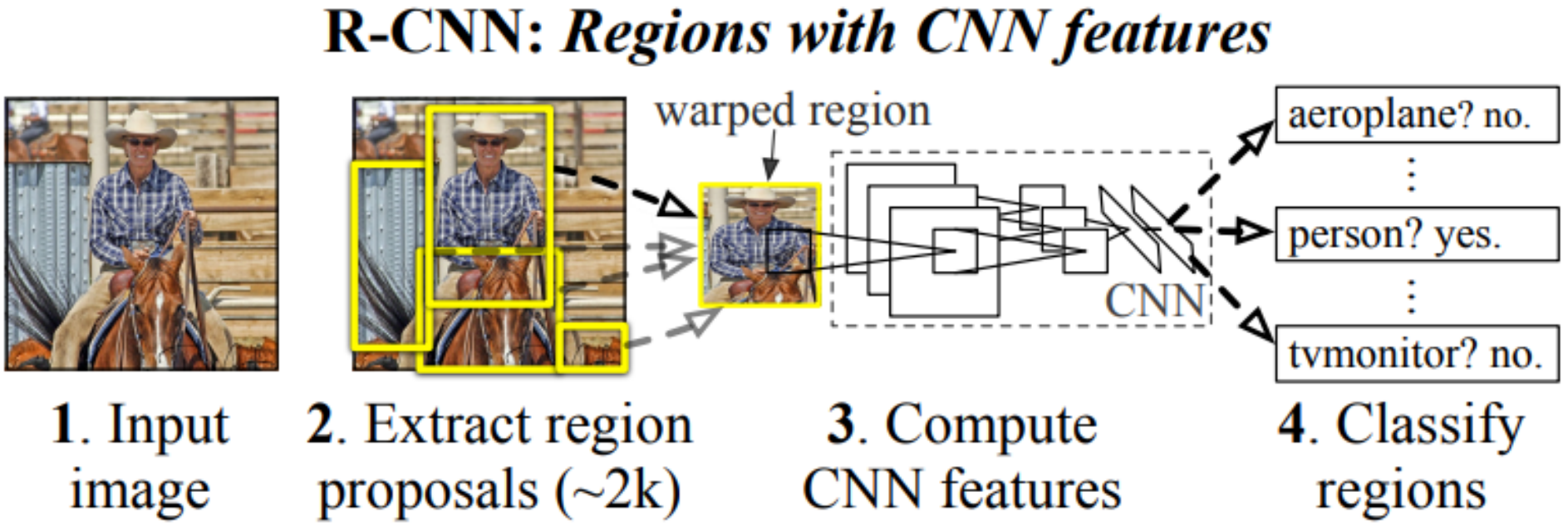
http://mp7.watson.ibm.com/ICCV2015/slides/iccv15_tutorial_training_rbg.pdf

1. Introduction

- R-CNNs were expensive → SPP-Net, Fast R-CNN reduced cost by sharing convolution across proposals
- Fast R-CNN achieves near real-time rates using very deep networks [19]
  when ignoring the time spent on region proposals → ***proposals are the computational bottleneck***
- Region proposal methods typically rely on inexpensive features and economical inference schemes
  1. compared to Fast R-CNN, ***SS*** is too slow (2s/image in a CPU)
  2. ***EB*** provides the best trade-off between proposal quality and speed (0.2s/image)
     but, the region proposal step still consumes as much running time as the detection network
  3. fast R-CNNs use GPU →but the region proposals still running on the CPU

SS (Selective Search): Uijlings, J. R., Van De Sande, K. E., Gevers, T., & Smeulders, A. W. (2013). Selective search for object recognition. *International journal of computer vision*, *104*(2), 154-171.
EB (Edge Boxes): Zitnick, C. Lawrence, and Piotr Dollár. "Edge boxes: Locating object proposals from edges." *European conference on computer vision*. Springer, Cham, 2014.

R-CNN



R. Girshick, J. Donahue, T. Darrell, & J. Malik. “Rich feature hierarchies for accurate object detection and semantic segmentation”. CVPR 2014.

Selective search

Fig. 2 Two examples of our selective search showing the necessity of different scales. On the *left* we find many objects at different scales. On the *right* we necessarily find the objects at different scales as the girl is contained by the tv

Algorithm 1: Hierarchical Grouping Algorithm

DontPrintSemicolon **Input**: (colour) image
Output: Set of object location hypotheses L

Obtain initial regions $R = \{r_1, \cdots, r_n\}$ using Felzenszwalb and Huttenlocher (2004) Initialise similarity set $S = \emptyset$;
foreach *Neighbouring region pair* (r_i, r_j) **do**
  Calculate similarity $s(r_i, r_j)$;
  $S = S \cup s(r_i, r_j)$;

while $S \neq \emptyset$ **do**
  Get highest similarity $s(r_i, r_j) = \max(S)$;
  Merge corresponding regions $r_t = r_i \cup r_j$;
  Remove similarities regarding r_i : $S = S \setminus s(r_i, r_*)$;
  Remove similarities regarding r_j : $S = S \setminus s(r_*, r_j)$;
  Calculate similarity set S_t between r_t and its neighbours;
  $S = S \cup S_t$;
  $R = R \cup r_t$;
Extract object location boxes L from all regions in R;

Uijlings, J. R., Van De Sande, K. E., Gevers, T., & Smeulders, A. W. (2013). Selective search for object recognition. *International journal of computer vision*, *104*(2), 154-171.

Edge Boxes

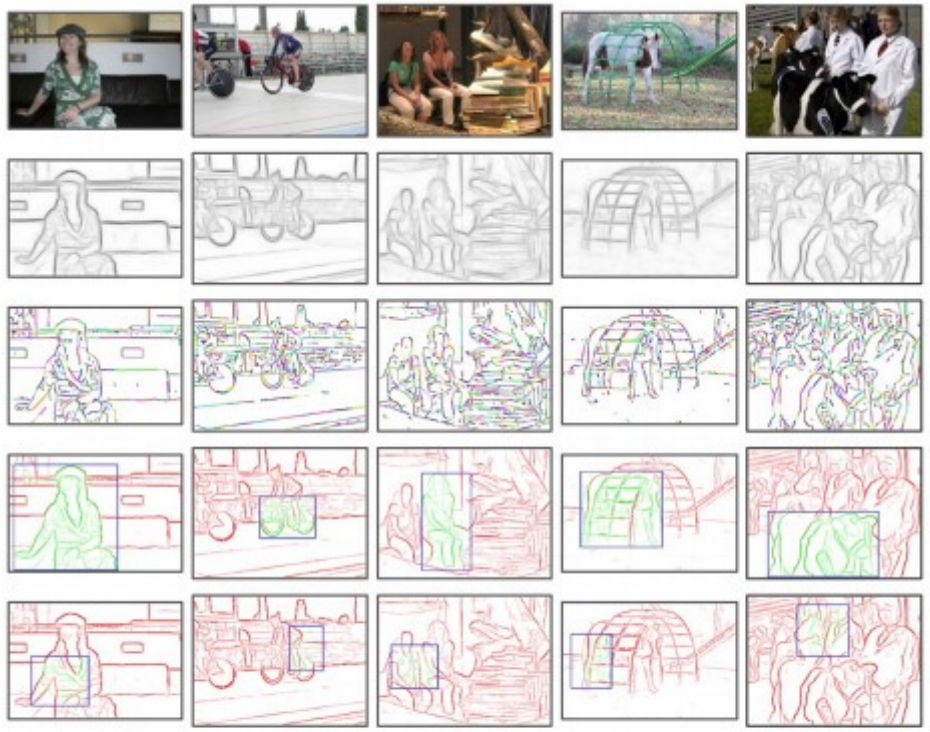


Fig. 1. Illustrative examples showing from top to bottom (first row) original image, (second row) Structured Edges [16], (third row) edge groups, (fourth row) example correct bounding box and edge labeling, and (fifth row) example incorrect boxes and edge labeling. Green edges are predicted to be part of the object in the box ($w_b(s_i) = 1$), while red edges are not ($w_b(s_i) = 0$). Scoring a candidate box based solely on the number of contours it *wholly encloses* creates a surprisingly effective object proposal measure. The edges in rows 3-5 are thresholded and widened to increase visibility.

Table 2. Results for our approach, Edge Boxes 70, compared to other methods for IoU threshold of 0.7. Methods are sorted by increasing Area Under the Curve (AUC). Additional metrics include the number of proposals needed to achieve 25%, 50% and 75% recall and the maximum recall using 5000 boxes. Edge Boxes is best or near best under every metric. All method runtimes were obtained from [26].

	AUC	N@25%	N@50%	N@75%	Recall	Time
BING [11]	.20	292	–	–	29%	**.2s**
Rantalankila [10]	.23	184	584	–	68%	10s
Objectness [4]	.27	27	–	–	39%	3s
Rand. Prim's [8]	.35	42	349	3023	80%	1s
Rahtu [7]	.37	29	307	–	70%	3s
Selective Search [5]	.40	28	199	1434	**87%**	10s
CPMC [6]	.41	15	111	–	65%	250s
Edge boxes 70	**.46**	**12**	**108**	**800**	**87%**	**.25s**

Zitnick, C. Lawrence, and Piotr Dollár. "Edge boxes: Locating object proposals from edges." *European conference on computer vision*. Springer, Cham, 2014.

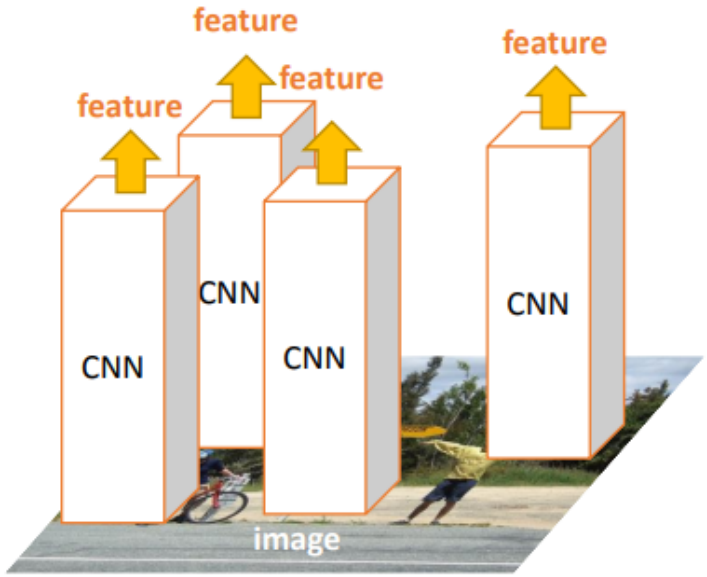
1. Introduction

- ***An obvious way to accelerate proposal computation is to re-implement it for the GPU***
- This may be an effective engineering solution, but re-implementation ignores the down-stream detection network and therefore misses important opportunities for sharing computation
- computing proposals with a deep net leads to an elegant and effective solution, and where nearly cost-free
- we introduce novel RPNs → share conv layers with SPP-Net and Fast R-CNN
- By sharing conv at test-time, the marginal cost for computing proposals is small
- the conv feature maps used by region-based detectors can also be used for generating region proposals.
- On top of these conv features, we construct RPNs by adding 2 additional conv layers:
  1. one encodes each conv map position into a short (e.g., 256-d) feature vector
  2. The other, at each conv map position, outputs an objectness score and regressed bounds for k-region proposals relative to various scales and aspect ratios at that location

1. Introduction

- RPNs can be trained end-to-end specifically for the task for generating detection proposals
- To unify RPNs with Fast R-CNN, we propose a simple training scheme that alternates between fine-tuning for the region proposal task and then fine-tuning for object detection, while keeping the proposals fixed
- This scheme converges quickly and produces a unified network with conv features that are shared between both tasks
- We evaluate our method on the PASCAL VOC detection benchmarks [4], where RPNs with Fast R-CNNs produce detection accuracy better than the strong baseline of SS with Fast R-CNNs.
- our method waives nearly all computational burdens of SS at test-time—the effective running time for proposals is just 10 milliseconds.
- Using the VGG-16 model, Fast R-CNN has a frame rate of 5fps on a GPU

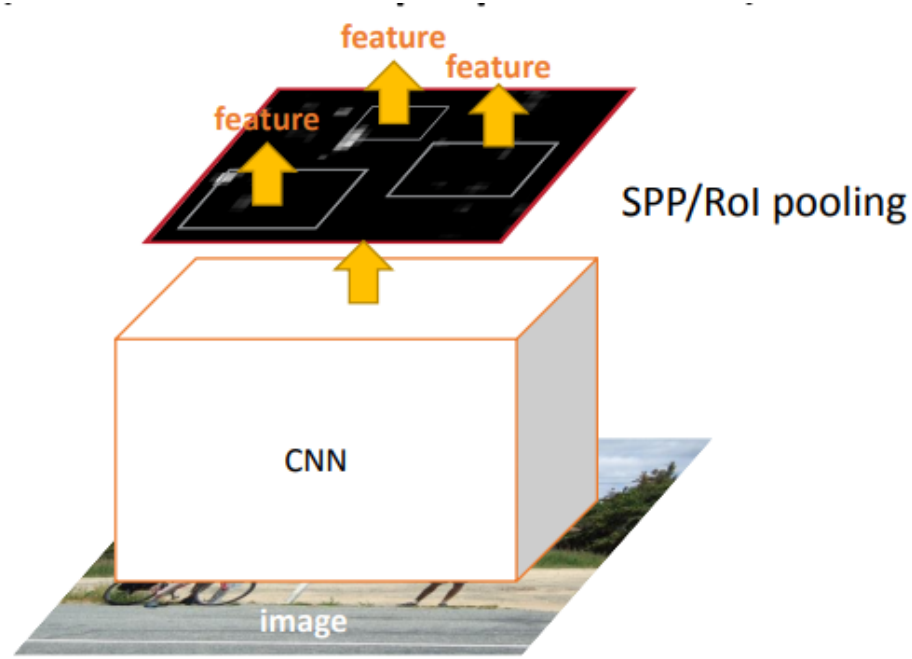
R-CNN vs. Fast R-CNN



R-CNN

- Complexity: ~224 × 224 × 2000

- Extract image regions
- 1 CNN per region (2000 CNNs)
- Classify region-based features



SPP-net & Fast R-CNN (the same forward pipeline)

- Complexity: ~600 × 1000 × **1**
- ~160**x faster** than R-CNN

- 1 CNN on the entire image
- Extract features from feature map regions
- Classify region-based features

http://mp7.watson.ibm.com/ICCV2015/slides/iccv15_tutorial_training_rbg.pdf

Faster R-CNN

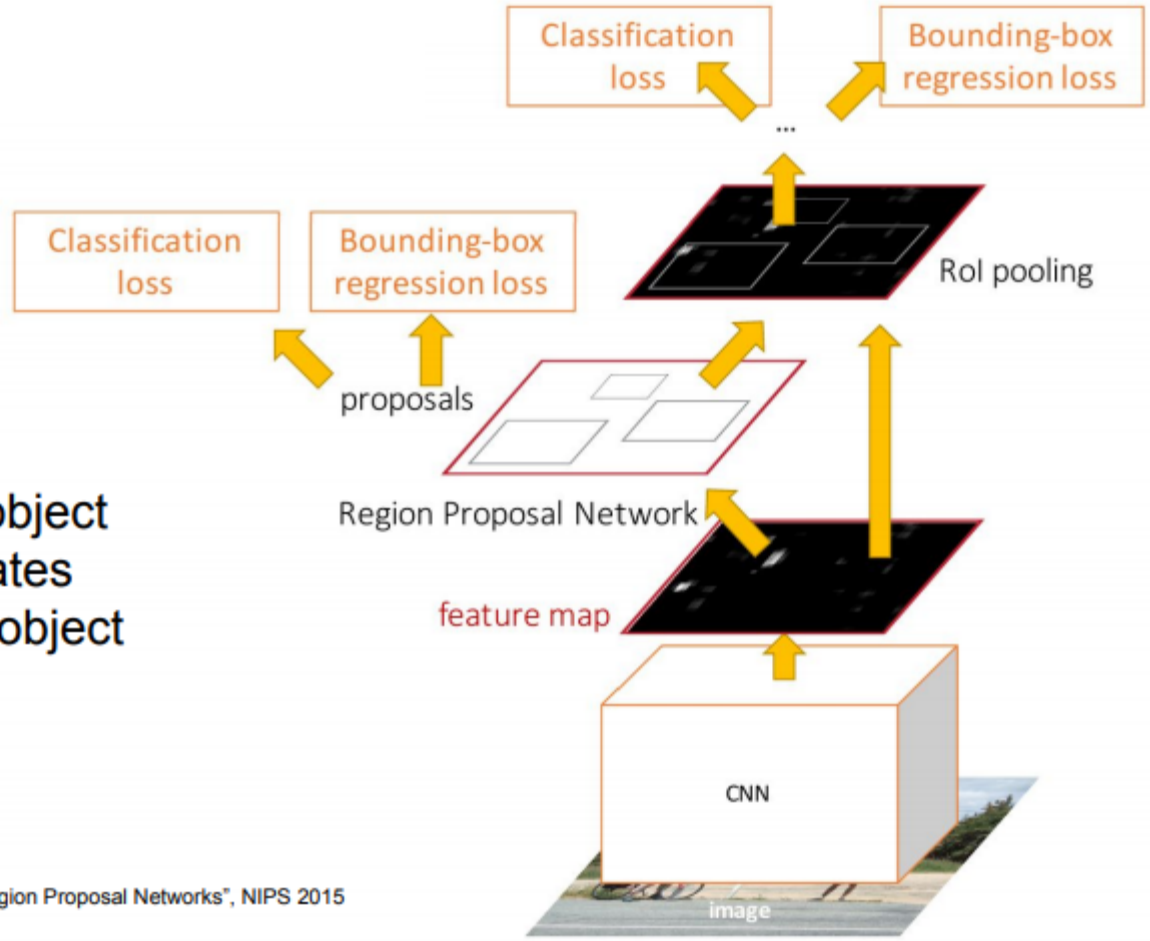
Faster R-CNN:

Make CNN do proposals!

Insert **Region Proposal Network (RPN)** to predict proposals from features

Jointly train with 4 losses:

1. RPN classify object / not object
2. RPN regress box coordinates
3. Final classification score (object classes)
4. Final box coordinates



Ren et al, "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks", NIPS 2015
Figure copyright 2015, Ross Girshick; reproduced with permission

http://cs231n.stanford.edu/slides/2017/cs231n_2017_lecture11.pdf

2. Related Work

- **using deep network to locate class-specific or class-agnostic bounding boxes**
  - ✓ OverFeat [18]: a FC layer is trained to predict the box coordinates for the localization task that assumes a single object. The FC layer is then turned into a conv layer for detecting multiple class-specific objects
  - ✓ MultiBox [3, 20]: generate region proposals from a network whose last FC layer simultaneously predicts multiple (e.g., 800) boxes, which are used for R-CNN. Their proposal network is applied to a single image or multiple large image crops (e.g., 224×224) [20].
- **Shared computation of convolutions for efficient, yet accurate, visual recognition**
  - ✓ OverFeat computes conv features from an image pyramid for classification, localization, and detection
  - ✓ Adaptively-sized pooling (SPP) on shared conv feature maps is proposed for efficient region-based object detection and semantic segmentation
  - ✓ Fast R-CNN enables end-to-end detector training on shared conv features and shows compelling accuracy and speed

3. Region Proposal Networks

- A RPN takes an image of any size as input and outputs a set of rectangular object proposals, each with an objectness score. We model this process with a FC network
- Because our ultimate goal is to share computation with a Fast R-CNN, we assume that both nets share a common set of conv layers
- In our experiments, we investigate the ZF model, which has 5 shareable conv layers and the Simonyan and Zisserman model (VGG), which has 13 shareable conv layers.
- To generate region proposals, we slide a small network over the conv feature map output by the last shared conv layer.
- This network is fully connected to an $n \times n$ spatial window of the input conv feature map. Each sliding window is mapped to a lower-dimensional vector (256-d for ZF and 512-d for VGG).
- This vector is fed into 2 sibling FC layers → a box-regression layer and a box-classification layer
- We use n = 3 in this paper, noting that the effective receptive field on the input image is large (171 and 228 pixels for ZF and VGG, respectively).

3. Region Proposal Networks

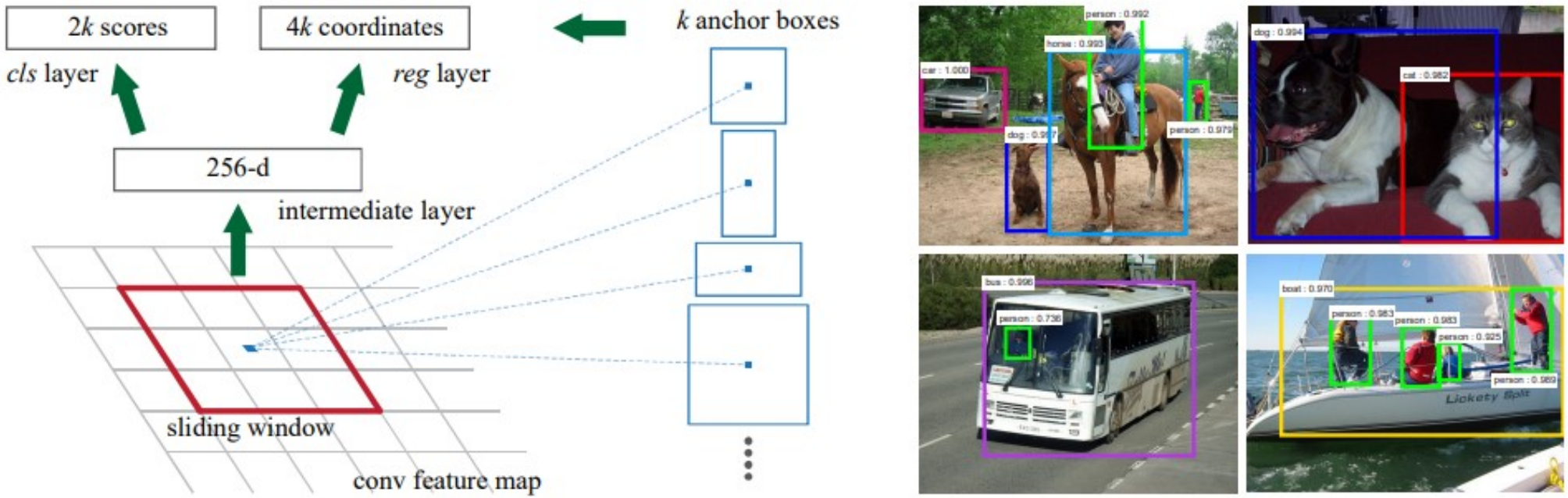


Figure 1: **Left**: Region Proposal Network (RPN). **Right**: Example detections using RPN proposals on PASCAL VOC 2007 test. Our method detects objects in a wide range of scales and aspect ratios.

3.1 Translation-Invariant Anchors

- At each sliding-window location, we simultaneously predict k region proposals, so the *reg* layer has $4k$ outputs encoding the coordinates of k boxes.
- The *cls* layer outputs $2k$ scores that estimate the probability of object / not-object for each proposal.
- The k proposals are parameterized *relative* to k reference boxes, called *anchors*.
- Each anchor is centered at the sliding window in question and is associated with a scale and aspect ratio. We use 3 scales and 3 aspect ratios, yielding $k = 9$ anchors at each sliding position.
- For a conv feature map of a size $W \times H$ (typically $\sim$2,400), there are W x H x k anchors
- This is *translation invariant*, both in terms of the anchors and the functions that compute proposals relative to the anchors
- As a comparison, the MultiBox uses k-means to generate 800 anchors, which are not translation invariant.
- If one translates an object in an image, the proposal should translate and the same function should be able to predict the proposal in either location.
- the MultiBox anchors are not translation invariant, it requires a $(4+1)\times 800$-dimensional output layer, whereas our method requires a $(4+2)\times 9$-dimensional output layer.
- RPN proposal layers have an order of magnitude fewer parameters (27M for MultiBox using GoogLeNet vs. 2.4M for RPN using VGG-16) → less risk of overfitting on small datasets

3.2 A Loss Function for Learning Region Proposals

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

$$t_x = (x - x_a)/w_a, \quad t_y = (y - y_a)/h_a, \quad t_w = \log(w/w_a), \quad t_h = \log(h/h_a),$$
$$t_x^* = (x^* - x_a)/w_a, \quad t_y^* = (y^* - y_a)/h_a, \quad t_w^* = \log(w^*/w_a), \quad t_h^* = \log(h^*/h_a),$$

where x, y, w, and h denote the two coordinates of the box center, width, and height. Variables x, x_a, and x^* are for the predicted box, anchor box, and ground-truth box respectively (likewise for y, w, h). This can be thought of as bounding-box regression from an anchor box to a nearby ground-truth box.

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

in which

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

3.3 Optimization

- trained end-to-end by back-propagation and SGD
- randomly sample 256 anchors in an image to compute the loss function of a mini-batch, where the sampled positive and negative anchors have a ratio of up to 1:1.
- If there are fewer than 128 positive samples in an image, we pad the mini-batch with negative ones.
- randomly initialize all new layers by drawing weights from a zero-mean Gaussian distribution with a standard deviation of 0.01.
- All other layers are initialized by pre-training a model for ImageNet classification
- tune all layers of the ZF net, and conv3 1 and up for the VGG net to conserve memory
- learning rate of 0.001 for 60k mini-batches
- learning rate of 0.0001 for next 20k mini-batches (on the PASCAL dataset)
- momentum of 0.9 and
- weight decay of 0.0005

3.4 Sharing Conv Features for Region Proposal and Object Detection

- While this joint optimizing is an interesting question for future work, we develop a pragmatic 4-step training algorithm to learn shared features via alternating optimization.
  1. train the RPN as described above. This network is initialized with an ImageNet pre-trained model and fine-tuned end-to-end for the region proposal task.
  2. train a separate detection network by Fast R-CNN using the proposals generated by the step-1 RPN. This detection network is also initialized by the ImageNet-pre-trained model. At this point, the two networks do not share conv layers.
  3. use the detector network to initialize RPN training, but we fix the shared conv layers and only fine-tune the layers unique to RPN. Now the two networks share conv layers
  4. keeping the shared conv layers fixed, we fine-tune the FC layers of the Fast R-CNN. As such, both networks share the same conv layers and form a unified network

3.5 Implementation Details

- train and test both region proposal and object detection networks on single-scale images
- re-scale the images such that their shorter side is s = 600 pixels.
- Multi-scale feature extraction may improve accuracy but does not exhibit a good speed-accuracy trade-off
- We also note that for ZF and VGG nets, the total stride on the last conv layer is 16 pixels on the re-scaled image, and thus is ~10 pixels on a typical PASCAL image (~500×375). Even such a large stride provides good results, though accuracy may be further improved with a smaller stride.
- For anchors, use 3 scales with box areas of 128^2, 256^2, and 512^2 pixels, and 3 aspect ratios of 1:1, 1:2, and 2:1.
- We note that our algorithm allows the use of anchor boxes that are larger than the underlying receptive field when predicting large proposals.
- Such predictions are not impossible—one may still roughly infer the extent of an object if only the middle of the object is visible.
- With this design, our solution does not need multi-scale features or multi-scale sliding windows to predict large regions, saving considerable running time. Fig. 1 (right) shows the capability of our method for a wide range of scales and aspect ratios. The table below shows the learned average proposal size for each anchor using the ZF net (numbers for s = 600).

3.5 Implementation Details

anchor	128^2, 2:1	128^2, 1:1	128^2, 1:2	256^2, 2:1	256^2, 1:1	256^2, 1:2	512^2, 2:1	512^2, 1:1	512^2, 1:2
proposal	188×111	113×114	70×92	416×229	261×284	174×332	768×437	499×501	355×715

- The anchor boxes that cross image boundaries need to be handled with care. During training, we ignore all cross-boundary anchors so they do not contribute to the loss. For a typical 1000 × 600 image, there will be roughly 20k (≈ 60 × 40 × 9) anchors in total.
- With the cross-boundary anchors ignored, there are about 6k anchors per image for training.
- If the boundary-crossing outliers are not ignored in training, they introduce large, difficult to correct error terms in the objective, and training does not converge.
- During testing, however, we still apply the fully-convolutional RPN to the entire image. This may generate cross-boundary proposal boxes, which we clip to the image boundary.
- Some RPN proposals highly overlap with each other. To reduce redundancy, we adopt ***NMS*** on the proposal regions based on their cls scores.
- We fix the IoU threshold for NMS at 0.7, which leaves us about 2k proposal regions per image.
- NMS does not harm the ultimate detection accuracy but substantially reduces the number of proposals.
- After NMS, we use the top-N ranked proposal regions for detection. In the following, we train Fast R-CNN using 2k RPN proposals, but evaluate different numbers of proposals at test-time.

NMS (non-maximum suppression):

4. Experiments

- PASCAL VOC 2007 detection benchmark (5k train-val images & 5k test images, 20 categories)
- ImageNet pre-trained network:
  1. “fast” version of ZF net (5 conv layers + 3 fc layers)
  2. VGG-16 (13 conv layers + 3 fc layers)
- Performance evaluation: mAP

mAP (mean Average Precision)

4. Experiments

Table 1: 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	2k	SS	2k	58.7
EB	2k	EB	2k	58.6
RPN+ZF, shared	2k	RPN+ZF, shared	300	**59.9**
ablation experiments follow below				
RPN+ZF, unshared	2k	RPN+ZF, unshared	300	58.7
SS	2k	RPN+ZF	100	55.1
SS	2k	RPN+ZF	300	56.8
SS	2k	RPN+ZF	1k	56.3
SS	2k	RPN+ZF (no NMS)	6k	55.2
SS	2k	RPN+ZF (no *cls*)	100	44.6
SS	2k	RPN+ZF (no *cls*)	300	51.4
SS	2k	RPN+ZF (no *cls*)	1k	55.8
SS	2k	RPN+ZF (no *reg*)	300	52.1
SS	2k	RPN+ZF (no *reg*)	1k	51.3
SS	2k	RPN+VGG	300	59.2

4.1 Ablation Experiments

Table 2: 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 2k. †: this was reported in [5]; using the repository provided by this paper, this number is higher (68.0±0.3 in six runs).

method	# proposals	data	mAP (%)	time (ms)
SS	2k	07	66.9†	1830
SS	2k	07+12	70.0	1830
RPN+VGG, unshared	300	07	68.5	342
RPN+VGG, shared	300	07	69.9	**198**
RPN+VGG, shared	300	07+12	**73.2**	**198**

Table 3: 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 2k. †: `http://host.robots.ox.ac.uk:8080/anonymous/HZJTQA.html`. ‡: `http://host.robots.ox.ac.uk:8080/anonymous/YNPLXB.html`

method	# proposals	data	mAP (%)
SS	2k	12	65.7
SS	2k	07++12	68.4
RPN+VGG, shared†	300	12	67.0
RPN+VGG, shared‡	300	07++12	**70.4**

Table 4: **Timing** (ms) on a K40 GPU, except SS proposal is evaluated in a CPU. "Region-wise" includes NMS, pooling, fc, and softmax. 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**

4.2 Detection Accuracy and Running Time of VGG-16

Table 2: 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 2k. †: this was reported in [5]; using the repository provided by this paper, this number is higher (68.0±0.3 in six runs).

method	# proposals	data	mAP (%)	time (ms)
SS	2k	07	66.9†	1830
SS	2k	07+12	70.0	1830
RPN+VGG, unshared	300	07	68.5	342
RPN+VGG, shared	300	07	69.9	**198**
RPN+VGG, shared	300	07+12	**73.2**	**198**

Table 3: 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 2k. †: `http://host.robots.ox.ac.uk:8080/anonymous/HZJTQA.html`. ‡: `http://host.robots.ox.ac.uk:8080/anonymous/YNPLXB.html`

method	# proposals	data	mAP (%)
SS	2k	12	65.7
SS	2k	07++12	68.4
RPN+VGG, shared†	300	12	67.0
RPN+VGG, shared‡	300	07++12	**70.4**

Table 4: **Timing** (ms) on a K40 GPU, except SS proposal is evaluated in a CPU. "Region-wise" includes NMS, pooling, fc, and softmax. 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**

4.3 Analysis of Recall-to-IoU

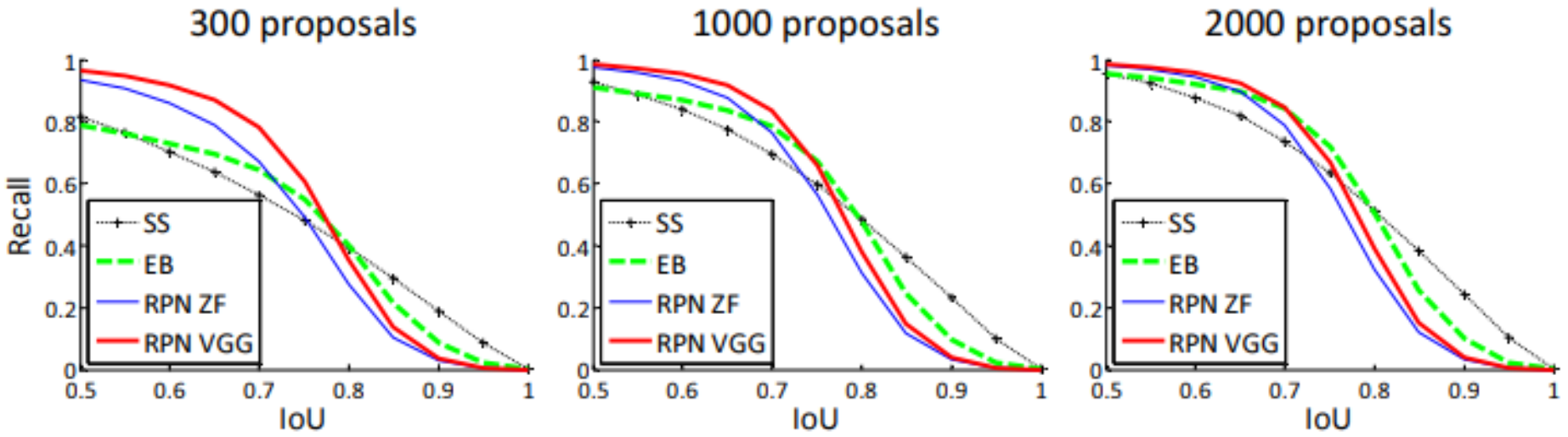


Figure 2: Recall *vs.* IoU overlap ratio on the PASCAL VOC 2007 test set.

Table 5: **One-Stage Detection *vs.* Two-Stage Proposal + Detection**. Detection results are on the PASCAL VOC 2007 test set using the ZF model and Fast R-CNN. RPN uses unshared features.

	regions		detector	mAP (%)
Two-Stage	RPN + ZF, unshared	300	Fast R-CNN + ZF, 1 scale	58.7
One-Stage	dense, 3 scales, 3 asp. ratios	20k	Fast R-CNN + ZF, 1 scale	53.8
One-Stage	dense, 3 scales, 3 asp. ratios	20k	Fast R-CNN + ZF, 5 scales	53.9

4.4 1-Stage Detection vs. 2-Stage Proposal + Detection