Object Detection

Joe Yeh, M.D.

Popular CNN for Object Detection

- One-stage
  - YOLO, YOLO v2, YOLO v3
  - SSD
  - RetinaNet (Top performer as of May 2018)
- Two-stage
  - R-CNN, Fast R-CNN, Faster R-CNN

Computer Science > Computer Vision and Pattern Recognition

Rich feature hierarchies for accurate object detection and semantic segmentation

Ross Girshick, Jeff Donahue, Trevor Darrell, Jitendra Malik

(Submitted on 11 Nov 2013 (v1), last revised 22 Oct 2014 (this version, v5))

Object detection performance, as measured on the canonical PASCAL VOC dataset, has plateaued in the last few years. The best-performing methods are complex ensemble systems that typically combine multiple low-level image features with high-level context. In this paper, we propose a simple and scalable detection algorithm that improves mean average precision (mAP) by more than 30% relative to the previous best result on VOC 2012---achieving a mAP of 53.3%. Our approach combines two key insights: (1) one can apply high-capacity convolutional neural networks (CNNs) to bottom-up region proposals in order to localize and segment objects and (2) when labeled training data is scarce, supervised pre-training for an auxiliary task, followed by domain-specific fine-tuning, yields a significant performance boost. Since we combine region proposals with CNNs, we call our method R-CNN: Regions with CNN features. We also compare R-CNN to OverFeat, a recently proposed sliding-window detector based on a similar CNN architecture. We find that R-CNN outperforms OverFeat by a large margin on the 200-class ILSVRC2013 detection dataset. Source code for the complete system is available at this http URL

Comments: Extended version of our CVPR 2014 paper; latest update (v5) includes results using deeper networks (see Appendix G. Changelog)
Subjects: **Computer Vision and Pattern Recognition (cs.CV)**
Cite as: **arXiv:1311.2524 [cs.CV]**
(or **arXiv:1311.2524v5 [cs.CV]** for this version)

Submission history

From: Ross Girshick [view email]
[v1] Mon, 11 Nov 2013 18:43:49 GMT (3704kb,D)
[v2] Tue, 15 Apr 2014 01:44:31 GMT (16729kb,D)
[v3] Wed, 7 May 2014 17:09:23 GMT (6644kb,D)
[v4] Mon, 9 Jun 2014 22:07:33 GMT (6644kb,D)
[v5] Wed, 22 Oct 2014 17:23:20 GMT (6660kb,D)

R-CNN : Two Key Insights

- R-CNN = Region proposal + CNN
- (1) one can apply high-capacity convolutional neural networks (CNNs) to bottom-up region proposals in order to localize and segment objects
- (2) when labeled training data is scarce, supervised pre-training for an auxiliary task, followed by domain-specific fine-tuning, yields a significant performance boost

Introduction

- During 2000-2010, progress in object detection was largely made by SIFT and HOG.
- SIFT and HOG are block-wise orientation histogram, which act like lower level neuron in primate visual system.
- Recognition, however, occurs at several stages downstream.

How to achieve object localization?

- Approach it as an image regression problem → didn't fare well
- Sliding window detector → neurons higher up has large receptive field, localization is not precise
- R-CNN : generate category independent region proposal, warp image to same size and extract feature vector then classify

R-CNN : Module Design

- Region proposal
  - Objectness, Selective search, constrained parametric min-cuts, etc.
- Feature Extraction Using CNN
  - Caffe implementation of AlexNet
- Classification with SVM
- Bounding box regression

Why SVM after CNN at all??

- Training set (ILSVRC) was not exhaustively labeled.
- Expected negative samples from random sampling the input could actually be false negatives.
- The validation set is exhaustively labeled. Hard negative mining was done using val set in the case of SVM.
- The author didn't train CNN on the hard negatives (perhaps because dataset is rather small for CNN). Softmax of CNN performs worse than SVM.

Selective Search

- Step 1 : Generate sub-segmentation
  - Efficient Graph Based Image Segmentation https://www.cs.cornell.edu/~dph/papers/seg-ijcv.pdf
- Step 2 : Greedy algorithm that iterative combines small similar regions into one, repeat until only one region remains. This generates a hierarchy.

Algorithm 1: Hierarchical Grouping Algorithm

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

Obtain initial regions $R = \{r_1, \cdots, r_n\}$ using [13]
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 et al. Selective Search for Object Recognition. IJCV 2013, https://ivi.fnwi.uva.nl/isis/publications/2013/UijlingsIJCV2013/

R-CNN : Training

- Supervised pre-training
  - CNN was trained on ImageNet dataset
- Domain-specific fine-tuning
  - Using warped region proposal (generated by selective search) for fine-tuning
  - SGD, learning rate = 0.001
  - all region proposals with ≥ 0.5 IoU overlap with a ground-truth box as positives
- Features extracted by CNN were used to train a set of class-specific SVMs.
- A bounding box regressor was trained to get finer localization

Caveat of R-CNN

- It's not computation efficient
  - One forward pass through CNN is done for every region proposal

Fast R-CNN

Ross Girshick

(Submitted on 30 Apr 2015 (v1), last revised 27 Sep 2015 (this version, v2))

This paper proposes a Fast Region-based Convolutional Network method (Fast R-CNN) for object detection. Fast R-CNN builds on previous work to efficiently classify object proposals using deep convolutional networks. Compared to previous work, Fast R-CNN employs several innovations to improve training and testing speed while also increasing detection accuracy. Fast R-CNN trains the very deep VGG16 network 9x faster than R-CNN, is 213x faster at test-time, and achieves a higher mAP on PASCAL VOC 2012. Compared to SPPnet, Fast R-CNN trains VGG16 3x faster, tests 10x faster, and is more accurate. Fast R-CNN is implemented in Python and C++ (using Caffe) and is available under the open-source MIT License at this https URL

Comments: To appear in ICCV 2015
Subjects: **Computer Vision and Pattern Recognition (cs.CV)**
Cite as: **arXiv:1504.08083 [cs.CV]**
(or **arXiv:1504.08083v2 [cs.CV]** for this version)

Drawbacks of RCNN

- Training is a multi-stage pipeline
- Training is expensive in space and time
- Object detection is slow

Fast RCNN architecture

- Region proposal (using selective search)
- CNN processes entire input and generate feature maps
- For each object proposal a region of interest (*RoI*) pooling layer extracts a fixed-length feature vector from the feature map
- Each feature vector is fed into fully connected layers that has two outputs : one for classification, one for bounding box coordinates

RoI Pooling Layer

- RoI max pooling works by dividing the $h \times w$ RoI window into an $H \times W$ grid of sub-windows of approximate size $h/H \times w/W$ and then max-pooling the values in each sub-window into the corresponding output grid cell

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

Shaoqing Ren, Kaiming He, Ross Girshick, Jian Sun

(Submitted on 4 Jun 2015 (v1), last revised 6 Jan 2016 (this version, v3))

State-of-the-art object detection networks depend on region proposal algorithms to hypothesize object locations. Advances like SPPnet and Fast R-CNN have reduced the running time of these detection networks, exposing region proposal computation as a bottleneck. In this work, we introduce a Region Proposal Network (RPN) that shares full-image convolutional features with the detection network, thus enabling nearly cost-free region proposals. An RPN is a fully convolutional network that simultaneously predicts object bounds and objectness scores at each position. The RPN is trained end-to-end to generate high-quality region proposals, which are used by Fast R-CNN for detection. We further merge RPN and Fast R-CNN into a single network by sharing their convolutional features---using the recently popular terminology of neural networks with 'attention' mechanisms, the RPN component tells the unified network where to look. For the very deep VGG-16 model, our detection system has a frame rate of 5fps (including all steps) on a GPU, while achieving state-of-the-art object detection accuracy on PASCAL VOC 2007, 2012, and MS COCO datasets with only 300 proposals per image. In ILSVRC and COCO 2015 competitions, Faster R-CNN and RPN are the foundations of the 1st-place winning entries in several tracks. Code has been made publicly available.

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

Faster R-CNN Module Design

- Fully convolutional network that proposes regions
- Fast R-CNN detector that uses proposed regions

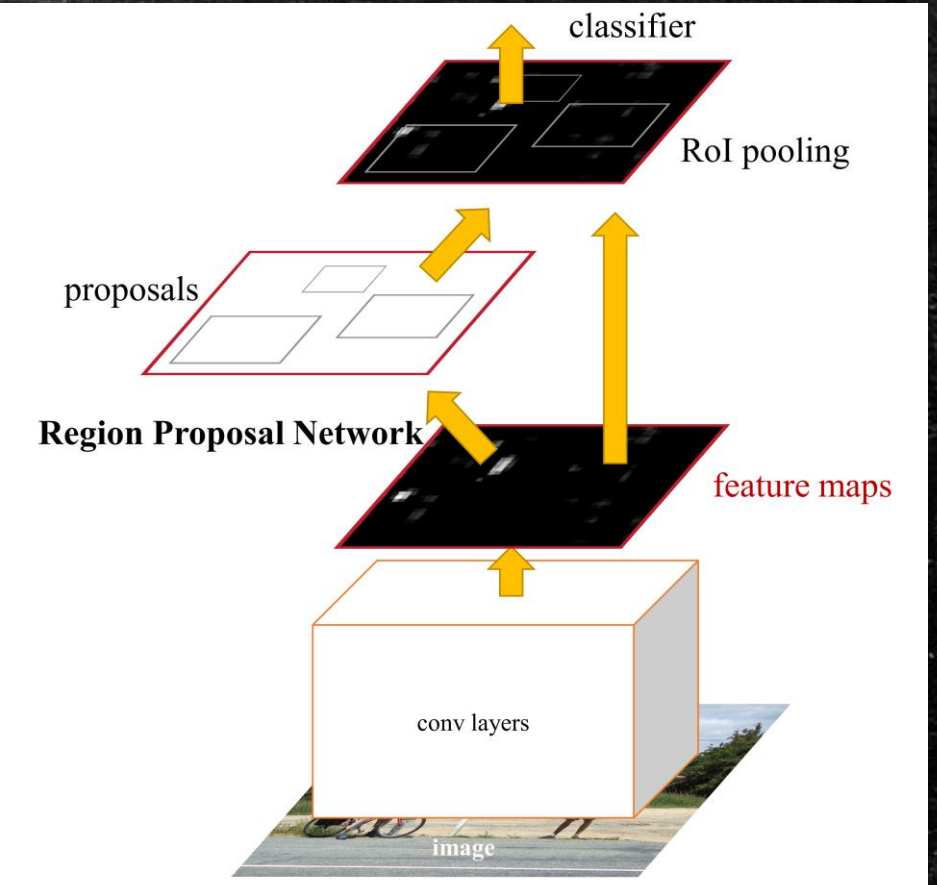
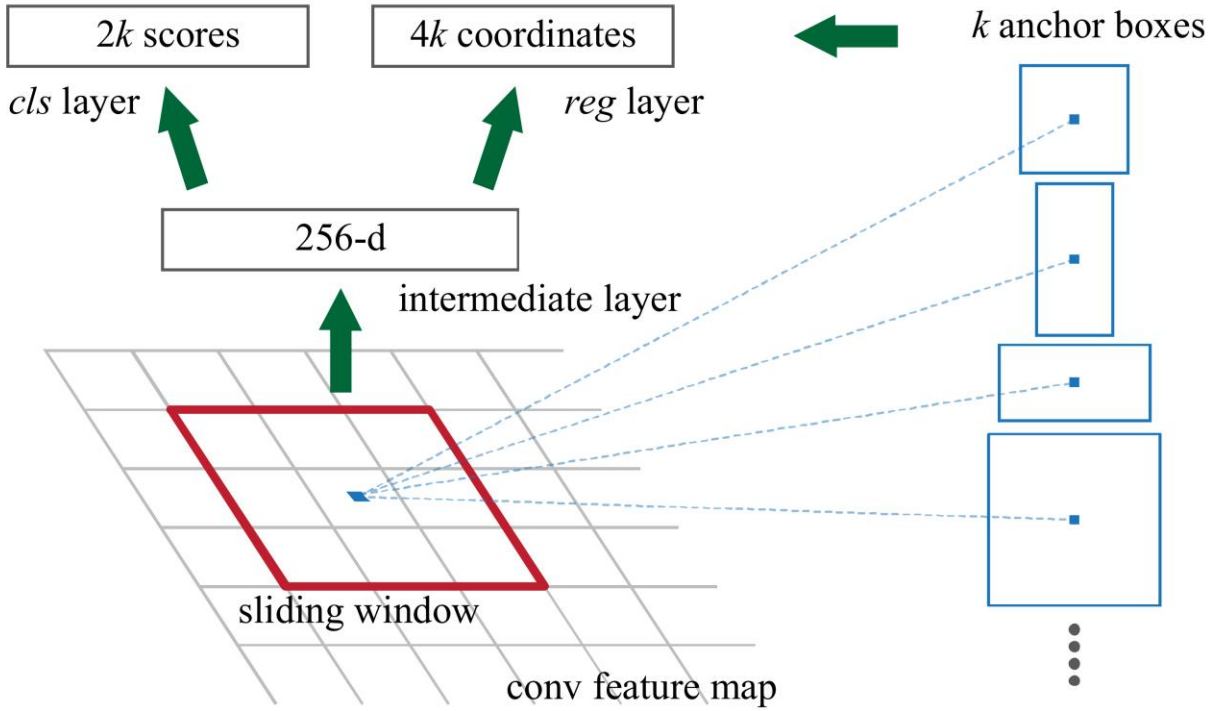


Figure 2: Faster R-CNN is a single, unified network for object detection. The RPN module serves as the 'attention' of this unified network.

Region Proposal Network



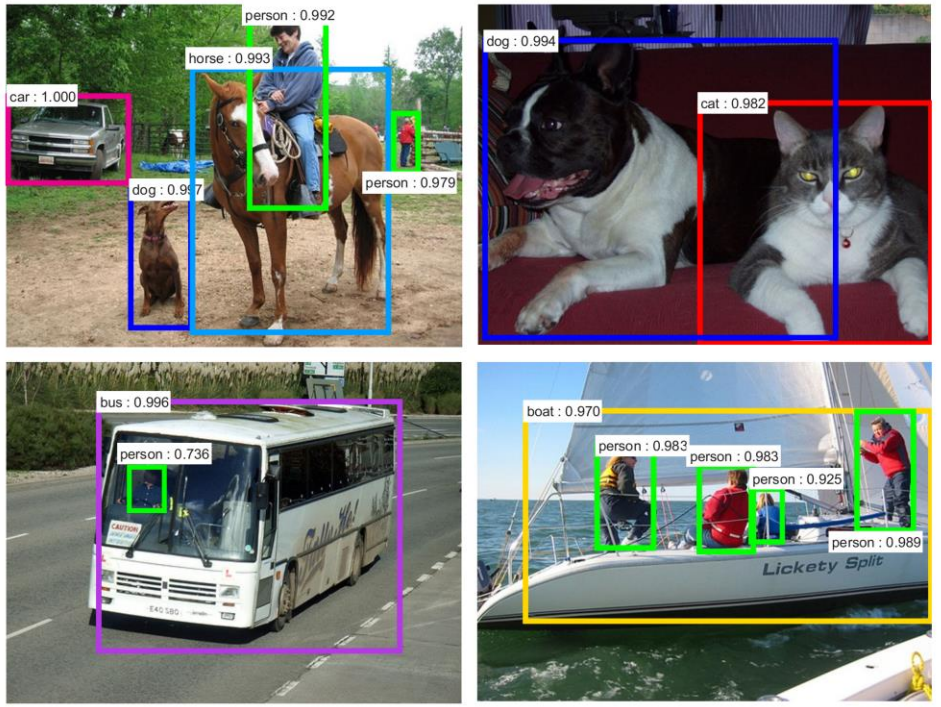


Figure 3: **Left**: Region Proposal Network (RPN). **Right**: Example detections using RPN proposals on PASCAL VOC 2007 test. Our method detects objects in a wide range of scales and aspect ratios.

Anchor boxes have multiple scales and multiple aspect ratios

Multi-task Loss for Region Proposal Network

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

- *i* is the index of an anchor in a mini-batch and *pi* is the predicted probability of anchor *i* being an object.
- t_i is a vector representing the 4 parameterized coordinates of the predicted bounding box, and t_i^* is that of the ground-truth box associated with a positive anchor
- The classification loss L_{cls} is log loss over two classes (object *vs*. not object).
- Regression loss is $reg(t_i, t_i^*) = R(t_i - t_i^*)$ where *R* is the robust loss function (smooth L1)

Sharing Features for RPN and Fast RCNN

- Both RPN and Fast R-CNN, trained independently, will modify their convolutional layers in different ways.
- Three ways for training networks with features shared:
  - *Alternating training*
  - *Approximate joint training :* Backward propagated signals from both the RPN loss and the Fast R-CNN loss are combined
  - *Non-approximate joint training* : A theoretically valid backpropagation solver that involves gradients w.r.t. the box coordinates. This is non-trivial.