Fast and Faster R-CNNs for Object Detection

Trufanova Evgeniia
National Reseach University
Higher School of Economics
Moscow, Russian Federation
Email: foggyjandjane@gmail.com

***Abstract*—In this paper I investigate into two methods of object detection: Fast R-CNN and Region Proposal Network. Fast R-CNN is able to classify object proposals efficiently and accurately using deep convolutional networks. Its implementation gives up to 10x speed-up in training compared to previous generation networks (R-CNN and SPPnet). RPN is a fully-convolutional network that serves for identifying regions of interest on the image. It can share convolutional features with Fast R-CNN which allows us to get region proposals almost for free. The combination of these two methods gives a significant advance in both training and testing time and brings us close to real-time object detection.**

Keywords—Deep Learning, R-CNN, Convolutional Networks, Object Detection, Image Classification

I. Introduction

Recent advances in region-based convolution networks (R-CNNs) for object detection achieved a noticeable reduction of computational time while preserving high accuracy rates. Fast R-CNN [1] lowered network training time by 10 times compared with R-CNN [2] and SPPnet [3]. Its ground-breaking feature was sharing convolutional layers across all the region proposals (regions of interest).

However, detection of these regions of interest was still time- and resource-demanding. For this purpose, Region Proposal Network (RPN) was designed. It is a fully-convolutional network that generates regions of interest, which can be used by Fast R-CNN for high-quality image classification. Faster R-CNN [4] combines RPN and Fast R-CNN so that Fast R-CNN also shares its convolutional layers with the detection network. As a result, region detection happens at the same time and becomes almost cost-free.

II. Main Part

A. Fast R-CNN

Previous incarnations of region-based convolutional networks showed excellent results in image classification. The main problem was their expensiveness in computational time and resources. There were several reasons for this:

- **multi-stage training** – bounding-box regressors and softmax classifiers were trained in separate stages
- **no sharing computation –** R-CNN performs convolutional layers forward pass separately for each region of interest
- **storing on the disk –** features for each image are saved to the disk, which makes training and testing very expensive in terms of space

Thus, Fast R-CNN was proposed.

Fast R-CNN receives a full image and several region proposals.

First, it processes the image with several convolutional layers and performs max-polling. This stage produces image feature map.

Then, for each region of interest a feature vector is extracted from this feature map using a pooling layer. It is processed with several fully-connected layers.

The output of Fast R-CNN for each region proposal is a probability vector of length K+1 (where K is number classes and the last number stands for probability of belonging to "background" class) and for real numbers x, y, h, w which encode a rectangle (x and y are top-left corner coordinates, h and w are height and width).

Max-pooling is performed separately for each region of interest. Layer has hyperparameters H and W. Suppose i-th region of interest has shape (h, w). Then the region will be divided into HW rectangles of approximate shape $(h / H, w / W)$ and max-pooling will be applied to each of the rectangles.

The architecture of Fast R-CNN enables updating all the network weights in one back-propagation step (which could not be achieved with SPPnet).

It also takes advantage of sharing weights across regions of interest when training: SGD minibatches are sampled from the same images (clearly, for Fast R-CNN making forward and backward pass for 100 regions from the same image is much faster than for 100 regions from 100 different images since they share conv layers step). Seemingly, this method could influence the convergence since regions in one image may correlate and this causes lack of representation diversity in each minibatch. However, in practice it turned out to be out of concern.

Fast R-CNN fixed the issue of multi-staging training using multi-task loss:

$$L(p, c, t^c, v^c) = L_{cls}(p, c) + Q\, Ind_{u>0}\, L_{loc}(t^c, v^c) \quad (1)$$

$L_{cls}(p, c)$ is the classification loss, where c is a ground-truth class for the sample and p is the predicted probability of sample belonging to the class. $L_{cls}(p, c) = -\log p_c$

$L_{loc}(t^c, v)$ penalizes model for bounding-box prediction. Here v is bounding-box prediction for class c and t is ground-truth bounding-box for class c. It is a simple smooth L1 loss.

$Ind_{u>0}$ in needed to disable penalizing model for "background" class prediction (which is labelled this $c=0$).

Q is a hyperparameter to balance the two losses.

Fast R-CNN's performance was compared to SPPnet and R-CNN. The experiment was conducted on PASCAL VOC 07 dataset (this dataset consists of about 5k trainval images and

5k test images of over 20 object categories). It showed that detection average precision (mAP) raised from 63.1% for SPPnet and 66% for R-CNN to 68.1%.

When it comes to training and testing times, Fast R-CNN reduces training time from 84 and 25.5 hours respectively to 9.5 hours. Testing also went approximately 10 times faster.

B. Region Proposal Network

Whilst Fast R-CNN drastically reduces computational cost of object detection, it assumes potential regions of interest as an input and ignores the cost of their detection. Thus, identifying these regions is still a bottleneck in image classification.

Most of the current solutions are compromises between high quality of region proposal and computational time. For instance, Selective Search [5] is based on a greedy algorithm and its performance rates are not high enough (2 seconds per image). This is a real drawback since modern object detection networks (like Fast R-CNN) are now able to process one image much faster.

New approach to region of interest detection shares convolutional layers with Fast R-CNN. Thus, a result of 10 milliseconds per image is achieved.

Fast R-CNN conv layers step processes an entire image and generates image feature map, as described in the previous paragraph. Then this feature map is fed to a small neural network, the architecture of which is described below.

First, a small window slides over the convolutional feature map. Each sliding window is mapped to a lower-dimensional vector (n x n convolutional layer), to which ReLU activation function [6] is applied. The vector is later fed into two sibling fully-connected convolutional layers (1 x 1). One of these layers is a bounding-box regression layer and the other is a bounding-box classification layer.

For each window K regions proposals are generated (where K is number of classes) and the classification layers outputs one vs. all probabilities. As a result, outputs contain 2K and 4K values respectively (bounding-box regressor outputs 4 numbers for each box).

When generating a training dataset, for each image several ground-truth boxes (i.e. objects) are identified. Then, each box is assigned positive, negative or neutral label. For each box its intersection over union (IoU) with ground-truth boxes is computed. Positive labels are assigned to all the boxes that have their intersection over union higher that 0.7 and, in addition, to several boxes with the highest intersection over union. Negative labels are assigned to all the boxes with intersection over union lower than 0.3. The rest of the boxes have neutral labels. Neutral labels do not contribute to model loss anyhow.

Loss is calculated as

$$L(p, l, t, v) = L_{cls}(p, l) + Q\, l\, L_{loc}(t, v) \qquad (2)$$

Here l is a ground-truth labels of the box and p is p is the predicted probability of belonging to this class. $L_{cls}(p, l)$ is the log classification loss.

$L_{loc}(t, v)$ penalizes model for bounding-box prediction. Here v is predicted box (encoded with 4 values x, y, h, w) and t is a ground-truth box. It is a simple smooth L1 loss.

l term is needed to set box loss to zero if ground-truth label is negative.

Q is a hyperparameter to balance the two losses.

C. Sharing convolutional layers between Fast R-CNN and RPN

So far, we have described building and training Region Proposal Network. Now we will consider integrating it with a region-based convolutional neural network that will use RPN's proposals for object detection.

The main idea of Region Proposal Network was sharing convolutional features between Fast R-CNN and RPN instead of training two separate networks (in this case they would have different convolutional layers). The training of the two nets will happens in 4 steps:

1. Train RPN end-to-end independently of Fast R-CNN.
2. Train Fast R-CNN end-to-end independently of RPN convolutional layers, but using region of interest proposals generated by RPN on the first step.
3. Initialize RPN with Fast R-CNN from the second step and train it with fixed shared convolutional layers (thus only RPN's own layers are updated and now the two networks share convolutional layers).
4. Train Fast R-CNN with fixed convolutional layers (thus only its fully-connected layers are fine-tuned).

Experiment were conducted to compare Fast C-RNN performance on PASCAL VOC 07 dataset with different region of interest proposals.

The network that shared convolutional layers with RPN (network trained in 4 stages described above) showed average accuracy of 59.9%, which is higher than accuracy of Fast R-CNN with proposals from Selective Search [5] (58.7%) and EdgeBoxes [7] (58.6%).

The experiments also show that is only the first two steps of training are kept, it reduces the mAP just slightly (to 58.7%). Also, if we add the third step – fine-tuning RPN with Fast R-CNN's weights, thus keeping the first three steps of training, the quality of region proposals is improved.

TABLE I. EXPERIMENT RESULTS

Method	mAP (%)
SS	58.7%
EB	58.6%
RPN (shared conv layers)	59.9%

Fig. 1. Comparing Fast R-CNN perfomace with different region propasal methods

CONCLUSION

Fast R-CNN gives significant improvements in both computational cost and accuracy of object detection. RPN not only improves the quality of regions of interest proposal, but

also allows to reduce performance time considerably by sharing convolutional stage with Fast R-CNN. Sharing convolutional layers between Fast R-CNN and RPN makes region proposal nearly cost-free and brings us closer to real-time object detection.

REFERENCES

[1] R. Girshick. Fast R-CNN. arXiv:1504.08083, 2015.

[2] R. Girshick, J. Donahue, T. Darrell, and J. Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In CVPR, 2014.

[3] K. He, X. Zhang, S. Ren, and J. Sun. Spatial pyramid pooling in deep convolutional networks for visual recognition. In ECCV, 2014.

[4] Shaoqing Ren, Kaiming He, Ross Girshick, Jian Sun. Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks.

[5] J. R. Uijlings, K. E. van de Sande, T. Gevers, and A. W. Smeulders. Selective search for object recognition. IJCV, 2013.

[6] V. Nair and G. E. Hinton. Rectified linear units improve restricted boltzmann machines. In ICML, 2010.

[7] C. L. Zitnick and P. Dollar. Edge boxes: Locating object proposals from edges. In ´ ECCV, 2014.