Fast R-CNN

Hongzhi Liu

July 19, 2018

Abstract

Recently, deep ConvNets have significantly improved image classification and object detection accuracy. Fast R-CNN builds on previous work to efficiently classify object proposals using deep convolutional networks. During preparation for URPC2018, I read a thesis written by Ross Girshick who is from Microsoft Research. This paper proposes a Fast Region-based Convolutional Network method (Fast R-CNN) for object detection. 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 9× faster than R-CNN, is 213× faster at test-time, and achieves a higher mAP on PASCAL VOC 2012. Compared to SPPnet, Fast R-CNN trains VGG16 3× faster, tests 10× faster, and is more accurate. Fast R-CNN is implemented in Python and is available under the open-source MIT License.

1. Overview of Fast R-CNN

Compared to image classification, object detection is a more challenging task that requires more complex methods to solve. Due to this complexity, current approaches [4, 5] train models in multi-stage pipelines that are slow and inelegant. Complexity arises because detection requires the accurate localization of objects, creating two primary challenges. First, numerous candidate object locations must be processed. Second, these candidates provide only rough localization that must be refined to achieve precise localization. Solutions to these problems often compromise speed, accuracy or simplicity.

In this paper, Ross Girshick streamlines the training process for state-of-the-art ConvNet-based object detectors [3]. Besides, he proposes a single-stage training algorithm that jointly learns to classify object proposals and refine their spatial locations. The resulting method can train a very deep detection network VGG16 [6] 9× faster than R-CNN [4] and 3× faster than SPPnet [5]. At runtime, the detection network processes images in 0.3s while achieving top accuracy on PASCAL VOC 2012 [2] with a mAP of 66% ans 62% for R-CNN.

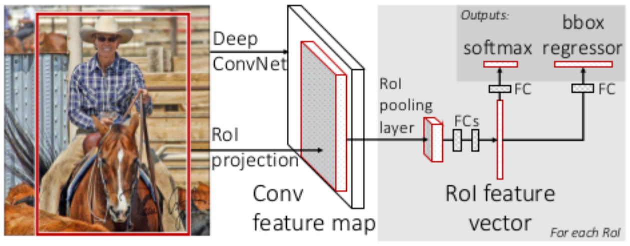

Figure 1. Fast R-CNN architecture. An input image and multiple regions of interest (RoIs) are input into a fully convolutional network. Each RoI is pooled into a fixed-size feature map and then mapped to a feature vector by fully connected layers (FCs). The network has two output vectors per RoI: softmax probabilities and per-class bounding-box regression offsets. The architecture is trained end-to-end with a multi-task loss.

2. Fast R-CNN Architecture and Training

Fig. 1 illustrates the Fast R-CNN architecture. A Fast R-CNN network takes as input an entire image and a set of object proposals. The network first processes the whole image with several convolutional *(conv)* and max pooling layers to produce a conv feature map. Then, 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 a sequence of fully connected *(fc)* layers that finally branch into two sibling output layers: one that produces softmax probability estimates over K object classes plus a catch-all "background" class and another layer that outputs four real-valued numbers for each of the K object classes. Each set of 4 values encodes refined bounding-box positions for one of the K classes.

2.1. The RoI Pooling Layer

The RoI pooling layer uses max pooling to convert the features inside any valid region of interest into a small feature map with a fixed spatial extent of H× W, where H and W are layer hyper-parameters that are independent of any particular RoI. In this paper, an RoI is a rectangular window into a conv feature map. Each RoI is defined by a four-tuple (r, c, h, w) that specifies its top-left corner (r, c) and its height and width (h, w).

2.2. Initializing from Pre-trained Networks

Ross Girshick and his team experiment with three pre-trained ImageNet [1] networks, each with five max pooling layers and between five and thirteen conv layers. When a pre-trained network initializes a Fast R-CNN network, it undergoes three transformations.

First, the last max pooling layer is replaced by a RoI pooling layer that is configured by setting H and W to be compatible with the net's first fully connected layer. Second, the network's last fully connected layer and softmax are replaced with the two sibling layers described earlier. Third, the network is modified to take two data inputs that are a list of images and a list of RoIs in those images.

2.3. Fine-tuning for Detection

Training all network weights with back-propagation is an important capability of Fast R-CNN. Girshick and his team propose a more efficient training method that takes advantage of feature sharing during training. In Fast R-CNN training, stochastic gradient descent (SGD) minibatches are sampled hierarchically, first by sampling N images and then by sampling R/N RoIs from each image. Critically, RoIs from the same image share computation and memory in the forward and backward passes. Making N small decreases minibatch computation. For example, when using $N = 2$ and $R = 128$, the proposed training scheme is roughly 64× faster than sampling one RoI from 128 different images. One concern over this strategy is it may cause slow training convergence because RoIs from the same image are correlated. This concern does not appear to be a practical issue and they achieve good results with $N = 2$ and $R = 128$ using fewer SGD iterations than R-CNN. In addition to hierarchical sampling, Fast R-CNN uses a streamlined training process with one fine-tuning stage that jointly optimizes a softmax classifier and bounding-box regressors.

3. Fast R-CNN Detection

Once a Fast R-CNN network is fine-tuned, detection amounts to little more than running a forward pass. The network takes as input an image and a list of R object proposals to score. At test-time, R is typically around 2000, although they will consider cases in which it is larger. When using animage pyramid, each RoI is assigned to the scale such that the scaled RoI is closest to 224^2 pixels in area [5]. For each test RoI r, the forward pass outputs a class posterior probability distribution p and a set of predicted bounding-box offsets relative to r. They assign a detection confidence to r for each object class k using the estimated probability $P_r(class = k|r) \triangleq p_k$. Then they perform non-maximum suppression independently for each class using the algorithm and settings from R-CNN [4].

For whole-image classification, the time spent computing the fully connected layers is small compared to the conv layers. On the contrary, for detection the number of RoIs to process is large and nearly half of the forward pass time is spent computing the fully connected layers (see Fig. 2). Large fully connected layers are easily accelerated by compressing them with truncated SVD. In this technique, a layer parameterized by the u× v weight matrix W is approximately factorized as Eq. 1:

$$W \approx U \sum_t V^T \tag{1}$$

In this factorization, U is a u× t matrix comprising the first t left-singular vectors of W , $\sum_t$ is a t× t diagonal matrix containing the top t singular values of W , and V is v× t matrix comprising the first t right-singular vectors of W. Truncated SVD reduces the parameter count from uv to t(u + v), which can be significant if t is much smaller than $\min(u, v)$. To compress a network, the single fully connected layer corresponding to W is replaced by two fully connected layers, without a non-linearity between them. The first of these layers uses the weight matrix $\sum_t V^T$ and the second uses U. This simple compression method gives good speedups when the number of RoIs is large.

4. Experiment Results of Model

Fast training and testing times are Ross Girshick's main result. Tab. 1 compares training time, testing rate, and mAP on VOC07 between Fast RCNN, R-CNN, and SPPnet. For VGG16, Fast R-CNN processes images 146× faster than R-CNN without truncated SVD and 213× faster with it. Training time is reduced by 9×, from 84 hours to 9.5. Compared to SPPnet, Fast RCNN trains VGG16 2.7× faster and tests 7× faster without truncated SVD or 10× faster with it.

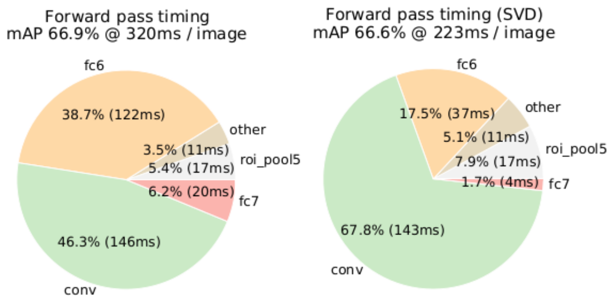

Figure 2. Timing for VGG16 before and after truncated SVD. Before SVD, fully connected layers fc6 and fc7 take 45% of the time.

Table 1. Number of dataset images for each tag.

	Fast R-CNN			R-CNN			SPPnet
	S	M	L	S	M	L	L
train time (h)	1.2	2.0	9.5	22	28	84	25
train speedup	18.3×	14.0×	8.8×	1×	1×	1×	3.4×
test rate (s/im)	0.10	0.15	0.32	9.8	12.1	47.0	2.3
with SVD	0.06	0.08	0.22	-	-	-	-
test speedup	98×	80×	146×	1×	1×	1×	20×
with SVD	169×	150×	213×	-	-	-	-
VOC07 mAP	57.1	59.2	66.9	58.5	60.2	66.0	63.1
with SVD	56.5	58.7	66.6	-	-	-	-

Fast R-CNN also eliminates hundreds of gigabytes of disk storage, because it does not cache features.

References

[1] J. Deng, W. Dong, R. Socher, L. J. Li, K. Li, and F. F. Li. ImageNet: A large-scale hierarchical image database. In *CVPR*, 2009. 2

[2] M. Everingham, L. V. Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The PASCAL visual object classes (VOC) challenge. *IJCV*, 2010. 1

[3] R. Girshick. Fast R-CNN. In *ICCV*, 2015. 1

[4] R. Girshick, J. Donahue, T. Darrell, and J. Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *CVPR*, 2014. 1, 2

[5] K. He, X. Zhang, S. Ren, and J. Sun. Spatial pyramid pooling in deep convolutional networks for visual recognition. In *ECCV*, 2014. 1, 2

[6] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *Computer Science*, 2014. 1