CSE616 Neural Networks and Their Applications
Project 2 Submission
Fast R-CNN

Ayman Wagih Mohsen (2000728)
Robeir Samir George (1901823)

June 2, 2022

1 Introduction

Fast R-CNN is an object detection model that improves in its predecessor the R-CNN which is called regional-based convolutional neural network, in a number of ways. Instead of extracting CNN features independently for each region of interest, Fast R-CNN aggregates them into a single forward pass over the image; i.e. regions of interest from the same image share computation and memory in the forward and backward passes.

1.1 R-CNN

The region-based CNNs has a good accuracy in object detection, but it has some disadvantages such as: the training is a multi-stage pipeline, as well as the problem that the training is expensive in space and time, and finally the detection at test time is very slow (reaches about 47 seconds per image). Like R-CNN we have SPPnet, which also has a great drawback, similar to R-CNN, training is a multi-stage pipeline that involves extracting features, fine-tuning a network with log loss, training SVMs, and fitting boundary-box regressors.

1.2 Contributions

The paper introduces a new training algorithm, trying to fix the drawbacks of both the R-CNN and SPPnet. The algorithm is called Fast R-CNN, with advantages such as: the training is a single-stage, training updates all layers of the network, and no disk storage is required for feature caching.

2 Fast R-CNN architecture and training

The paper uses Selective Search to generate a number of regions of interest (RoIs) (the specific method for generating RoIs is not the focus of this paper).

Each RoI is then evaluated for whether there's an object there, and if so what its bounding box is. The innovation here is to share expensive early-layer computation with the help of an RoI pooling layer, which also means that the heavy use of caching features to disk is no longer necessary. The architecture takes in an image and a set of RoIs. For each RoI we output softmax probabilities over K classes and x, y, w, h adjustments to the region that determine a bounding box prediction — the exact parameterization of the bounding box numbers is somewhat complicated and is laid out in the earlier R-CNN paper. For the loss function, the paper follows the now-standard strategy of putting as much as possible into one network and combining multiple terms into a single loss. In this case there are two terms: a class loss (a log loss) and a location loss (a smoothed L1 loss summed over the 4 outputs — they went with this because L2 loss caused exploding gradients). Regions that are classified as background don't include a location loss, which makes sense considering that backgrounds don't really have bounding boxes.

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

2.0.1 RoI Pooling:

This is what the paper is all about, but it's actually quite simple and is equivalent to what is now referred to as adaptive pooling. The idea is to have a max pooling layer that takes an arbitrary input region and maps it down to a fixed H×W feature map (e.g. 7×7 — this is a hyperparameter). You just divide the region into H×W subregions and do max pooling.

2.0.2 Scale Invariance:

In the object detection problems, it is common to try achieving scale invariant detection, here we have two approaches: 1. Via the brute force learning and 2. Via using image pyramids. In the brute force approach, each image is processed at a pre-defined pixel size during both training and testing. The other approach provides approximate scale-invariance to the network through an image pyramid, where at the testing time, the image pyramid is used to approximately scale-normalize each object proposal.

3 Fast R-CNN Detection

• The network takes an image as an input, and a list of R objects proposals to score. R is around 2000 at testing time.
• When using an image pyramid, each RoI is assigned to the scale such that the scaled RoI is closer to 224^2 pixels in area.
• For each test RoI, the forward pass outputs a class and a set of predicted bounding-box offsets relative to the RoI.

3.1 Truncated SVD for faster detection

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.

4 Main Results

4.1 Experimental Results

Fast R-CNN uses 3 pretrained ImageNet Models:
1. The small (S) size CaffeNet, which is based on AlexNet.
2. The medium (M) size VGG_CNN_M_1024.
3. The large (L) size VGG16.
Training and testing are single sale and the length of the shortest image side s=600.

4.2 VOC 2007, 2010 and 2012 results

The PASCAL Virtual Object Classes (VOC) project is an object classification competition that had a series of challenges spanning from 2005 till 2012.
The mean average precision (mAP) of Fast R-CNN on VOC 2012 was the highest at its time (65.7%). It was also the fastest contestant on that dataset.
Fast R-CNN was surpassed only by SegDeepM on VOC 2010.
As for VOC 2007 Fast R-CNN is a large improvement over the other contestants.

4.3 Training and testing time

Truncated singular value decomposition (SVD) helps in fully connected (FC) layers. For example if the fc6 layer has weights in the shape of a 25088×4096 matrix, then by using SVD we can use the largest 1024 singular values only. Saving test time at the cost of a little precision.

4.4 Which layers to fine-tune?

With very deep networks it is not enough to fine tune just the fully connected layers. In the fast R-CNN study using the large VGG16 net, it was revealed that it is best to fine-tune the last 9 conv layers out of the total 13 ($conv3_1$ and up). And this increased mAP from 61.4% to 66.9%. Fine-tuning starting from earlier layers has more impact on train time than on mAP.
As for medium and small nets, it was shown that fine-tuning conv layers has too little of an impact on the mAP. So fine-tuning the FC layers only is enough.

Table 1: Fast R-CNN train and test time compared to R-CNN

	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	-	-	-	-

5 Design evaluation

5.1 Does multi-task training help?

Multi-task learning is when a network is trained for multiple tasks simultaneously to save redundant calculations at test time, espacially in the early layers. The loss function becomes a weighted sum of sub-task loss functions. From the fast R-CNN study, it turns out that multi-task training for tasks A and B sometimes helps the network get better at task A than just training for task A. When the network was jointly trained for object classification and bounding box regression, it got better by 0.8 to 1.1 mAP at classification than training just for classification. This positive effect is consistent assuming the tasks are somehow related.

5.2 Scale invariance: to brute force or finesse?

Image scale is the length of its shortest side. Brute-force learning here means that all images are in single scale (of s=600). The aspect ration is kept the same. While multi-scale learning uses image pyramids at scales of s=480, 576, 688, 864, 1200. The fast R-CNN concludes that there is no need to train at multiple scales, as the gain in mAP is negligible compared to the increase in training memory requirements. This applies especially to the deep models like VGG16.

5.3 Do we need more training data?

When the number of images in VOC 2007 dataset was tripled to 16500 by augmenting with VOC 2012 dataset, the mAP improved from 66.9% to only 70.0%.

When the VOC 2010 dataset was augmented with VOC 2007 to 21500 images and the learning rate is lowered by $0.1\times$ each 40k iterations, mAP improved more significantly from 66.1% to 68.8%. The same was done with VOC 2012 dataset and mAP improved from 65.7% to 68.4%.

5.4 Do SVMs outperform softmax?

R-CNN uses softmax, while fast R-CNN was tested with softmax and SVM. Softmax slightly outperforms support vector machines (SVMs) by 0.1 to 0.8 mAP.

5.5 Are more proposals always better?

A proposal here means a candidade region of interest (RoI). Object detectors can use a sparse set of object proposals (e.g. selective search) or a dense set (e.g. DPM). For sparse set, as the proposal count increases, mAP rises then falls slightly, peaking around 3000 or 4000 proposals. With densely generated boxes is was concluded that mAP drops with larger proposal count.

5.6 Preliminary MS COCO results

The large VGG16 net based fast R-CNN was also trained on Microsoft COCO dataset with 80k images for 240k iterations, giving mAP of 35.9%.

6 Conclusion

An improvement on Region-based CNN (R-CNN) was proposed called Fast R-CNN. The new object detection technique is orders of magnitude faster at training and testing, allowing for more experimentation to study the effects of various hyperparameters on mean average precision (mAP). The speedup was achieved as follows: instead of passing each region of interest (RoI) again and again through the CNN, the CNN is run on the whole image followed by a novel stage called ROIPooling. This eliminates redundant calculations.

References

[1] Ross Girshick (ICCV 2015) ”Fast R-CNN” https://arxiv.org/abs/1504.08083