Fast R-CNN

QingHong Lin
2018/10

Abstract

• Fast R-CNN imporve training and testing speed while also increasing detection accuracy.
• Compared to **R-CNN**,Fast R-CNN trains VGG16 9x faster,tests 213x faster and higher mAP.
• Compared to **SPPnet**,Fast R-CNN trains VGG16 3x faster,tests 10x faster and more accurate.

1.Introduction

• current approaches train models in multi-stage pipelines that are slow and inelegant.
•this paper propose **a single-stage training algorithm** to classify object proposals and refine their spatial locations.

1.1.R-CNN and SPPnet

R-CNN

• R-CNN **drawbacks**
  Training is a multi-stage pipeline
  Training is expensive in space and time
  Object detection is slow
• R-CNN **slow** because it performs a ConvNet forward pass for each object proposal **without sharing computation.**

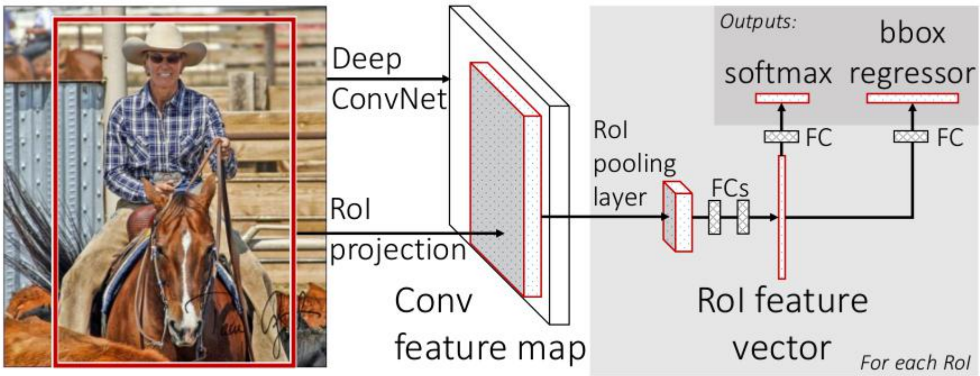
SPPnet

• SPPnets speed up R-CNN **by sharing computation**.
• Sppnets method
1)computes a convolutional feature map for the entire input image
2)classifies each object proposal using a feature vector extracted from the shared feature map.
3)Features are extracted for a proposal by max-pooling the portion of the feature map inside the proposal into a fixed-size output.
4)Multiple output sizes are pooled and then concatenated as in spatial pyramid pooling.
• SPPnet accelerates R-CNN by 10 to 100x at test time and training time is reduced by 3x.
• SPPnet **drawbacks**
  training is a multi-stage pipeline
  fixed convolutional layers limits the accuracy of deep networks

1.2.Contributions

• Fast R-CNN **advantages**
  Higher detection quality(mAP) than R-CNN,SPPnet
  Training is single-stage, using a multi-task loss
  Training can update all network layers
  No disk storage is required for feature caching

2.Fast R-CNN architecture and training



- Fast R-CNN input:image with proposals.
- Fast R-CNN process

1)image was <u>processed</u> by several **conv** and **max pooling layers** to <u>produce a conv feature map</u>.

2)for each object proposal, **RoI pooling layer** extracts <u>a fixed-length feature vector</u> from the feature map.

3)Each feature vector throught **fc layers** that finally branch into <u>two sibling output layers</u>:

-produces **softmax probability** estimates over <u>K object classes plus a “background” class</u>

-each set of **4 values** encodes <u>refined bounding-box positions</u> for one of the K classes.

2.1.The RoI pooling layer

- RoI defined by a **four-tuple(r,c,h,w)**:top-left corner(r,c) and height and width(h,w)
- RoI pooling process

1)divide the <u>hxw RoI window</u> into <u>a HxW feature map</u> of <u>sub-windows of size h/Hxw/W</u>.

2)max-pooling the values in <u>each sub-window</u>.

- RoI layer is the **special-case** of <u>the spatial pyramid pooling layer in SPPnets</u>.

2.2.Initializing from pre-trained networks

- initializes a Fast R-CNN undergoes three transformations

1)<u>the last max pooling layer</u> is replaced by a <u>RoI pooling layer.</u>

2)<u>last fully connected layer and softmax</u> are replaced by <u>two sibling layers</u>

(a fully connected layer and softmax over K+1 categories and category-specific bounding-box regressors)

3)two data **inputs**:<u>a list of images</u> and <u>a list of RoIs in those images</u>.

2.3.Fine-tuning for detection

- why SPPnet unable update weights below the spatial pyramid pooling layer?

when <u>each training sample from a different image</u>, **back-propagation** through the SPP layer is **highly inefficient** because <u>each RoI may have a very large receptive field,often spanning the entire input iamge</u>.

sampled hierarchiacally

- paper's method:feature sharing during training

SGD minibatches are **sampled hierarchiacally**,RoIs from the **same image** <u>share computation and</u>

memory in the forward and backward passes.

• may cause **slow training convergence**:RoIs from the same image are correlated.

streamlined training process with one fine-tuning stage

• optimizes a softmax classifier and bounding-box regressors rather than training three model separately.

Multi-task loss

• two output:discrete probabilty distribution *p=(p0,...,pk)* and bounding-box regression offsets *tk=(txk, tyk, twk, thk)*.

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

in which $L_{\text{cls}}(p, u) = -\log p_u$ is log loss for true class u.

• [u>=1] evaluates when u>=1 and 0 when catch-all background class hence Lloc is ignored.

$$L_{\text{loc}}(t^u, v) = \sum_{i \in \{\text{x,y,w,h}\}} \text{smooth}_{L_1}(t_i^u - v_i), \quad (2)$$

in which

$$\text{smooth}_{L_1}(x) = \begin{cases} 0.5x^2 & \text{if } |x| < 1 \\ |x| - 0.5 & \text{otherwise}, \end{cases} \quad (3)$$

• L1 loss less sensitive to outliers than L2 loss which may cause exploding gradients.

• λ controls the balance between the two task losses.All experiments use λ =1.

Mini-batch sampling

• RoIs with IoU at least 0.5 labeled with a foreground object class(**u>=1**).

• RoIs with IoU in [0.1,0.5) are labeled with background(**u=0**).

• RoIs with IoU lower 0.1 for hard example mining.

Back-propagation through RoI pooling layers

$$\frac{\partial L}{\partial x_i} = \sum_r \sum_j [i = i^*(r, j)] \frac{\partial L}{\partial y_{rj}}.$$

SGD hyper-parameters

• FC layers used for two ouput are initialized from zero-mean Gaussian distributions.

2.4.Scale invariance

• two ways of achieving scale invariant object detection

(1)**via "brute force" learning**

each image is processed at a pre-defined pixel size during both training and testing. scale-invariant from the training data.

(2)**multi-scale approach**

provides scale-invariance to the network through an image pyramid.

3.Fast R-CNN detection

• **input**:an image(pyramid) and a list of R object proposals to score.

• **output**:a class posterior probability distribution p and a set of predicted bounding-box offests relative to r.

3.1.Truncated SVD for faster detection

$$W \approx U \Sigma_t V^T$$

- Truncated SVD reduces the parameter count from uv to t(u+v).(t is much smaller than min(u,v))
- this method gives good speedups when the number of RoIs is large.

4.Main results

- Three results

1)State-of-the art **mAP** on VOC07, 2010 and 2012

2)Fast training and testing **compared** to R-CNN, SPPnet

3)**Fine-tuning** conv layers in VGG16 improves mAP

4.1.Experimental setup

- three models:
  - model S: CaffeNet;
  - model M: VGG_CNN_M_1024
  - model L: VGG16

4.2.VOC 2012 and 2012 results

- on VOC2012, Fast R-CNN is **top** with a mAP of 65.7% and two magnitude **faster** than other.
- on VOC 2010
  - SegDeepM(67.2%) better than Fast R-CNN(66.1%)
  - When enlarged 07++12 training set, Fast R-CNN increases to 68.8% surpassing SegDeep.

4.3.VOC 2007 results

- fine-tuning the conv layers improve mAP from 63.1% to 66.9%.
- Removing “difficult” examples imporves Fast R-CNN mAP to 68.1%.

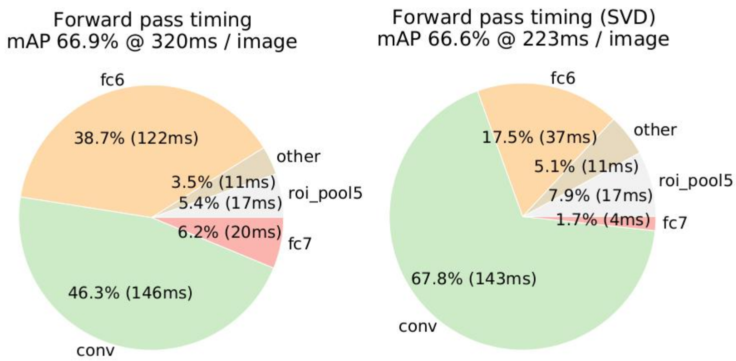
4.4.Training and testing time

- Result

	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	-	-	-	-

- Truncated SVD

- reduce detection time by more than 30%
- small drop in mAP
- without needing to perform additional fine-tuning.



4.5.Which layers to fine-tune?

• SPPnet paper idea: fine-tuning only the FC layers are good for accuracy.

• we **<u>freeze</u>** <u>the 13 conv layers</u> so that <u>only the fc layers learn</u> and we find mAP decreases from 66.9% to 61.4%.

• verifies hypothesis: <u>training through the RoI pooling layer</u> is important for **deep nets**.

Table 1: **ConvNet configurations** (shown in columns). The depth of the configurations increases from the left (A) to the right (E), as more layers are added (the added layers are shown in bold). The convolutional layer parameters are denoted as "conv⟨receptive field size⟩-⟨number of channels⟩". The ReLU activation function is not shown for brevity.

ConvNet Configuration					
A	A-LRN	B	C	D	E
11 weight layers	11 weight layers	13 weight layers	16 weight layers	16 weight layers	19 weight layers
input (224 × 224 RGB image)					
conv3-64	conv3-64 **LRN**	conv3-64 **conv3-64**	conv3-64 conv3-64	conv3-64 conv3-64	conv3-64 conv3-64
maxpool					
conv3-128	conv3-128	conv3-128 **conv3-128**	conv3-128 conv3-128	conv3-128 conv3-128	conv3-128 conv3-128
maxpool					
conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256 **conv1-256**	conv3-256 conv3-256 **conv3-256**	conv3-256 conv3-256 conv3-256 **conv3-256**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
FC-4096					
FC-4096					
FC-1000					
soft-max					

Table 2: **Number of parameters** (in millions).

Network	A,A-LRN	B	C	D	E
Number of parameters	133	133	134	138	144

• not all conv layers should be fine-tuned such as in S&M, allow conv1 to learn or not has no meaningful effect.

- for VGG16, only necessary to updata layers from conv3_1 and up which is help because:
    - updating from conv2_1 is slower
    - updating from conv1_1 over-runs GPU memory

5.Design evaluation

5.1.Does multi-task training help?

- multi-task training **improves** pure classification accuracy(+0.8 to +1.1 mAP)

	S				M				L			
multi-task training?		✓		✓		✓		✓		✓		✓
stage-wise training?			✓				✓				✓	
test-time bbox reg?			✓	✓			✓	✓			✓	✓
VOC07 mAP	52.2	53.3	54.6	**57.1**	54.7	55.5	56.6	**59.2**	62.6	63.4	64.0	**66.9**

5.2.Scale invariance: to brute force or finesse?

- single-scale detection performs as well as multi-scale detection.
- deep ConvNets are adept at directly learning scale invariance.
- single-scale processing offers the best tradeoff between speed and accuracy.

	SPPnet **ZF**		S		M		L
scales	1	5	1	5	1	5	1
test rate (s/im)	0.14	0.38	**0.10**	0.39	0.15	0.64	0.32
VOC07 mAP	58.0	59.2	57.1	58.4	59.2	60.7	**66.9**

5.3.Do we need more training data?

- Enlarging the training set improves mAP on VOC07 test from 66.9% to 70.0%

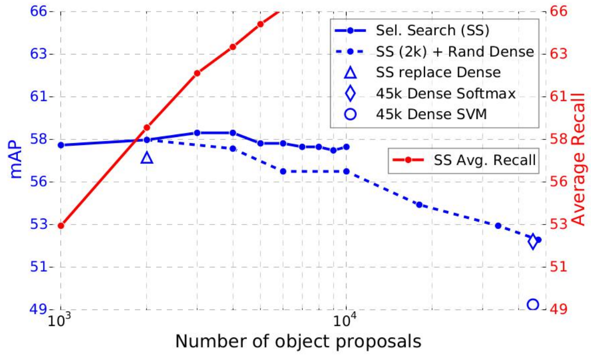
5.4.Do SVMs outperform softmax?

- softmax slightly outperforming SVM(+0.1 to +0.8 mAP).
- “one-shot”fine-tuning is **sufficient** compared to previous multi-stage training.

method	classifier	S	M	L
R-CNN [9, 10]	SVM	**58.5**	**60.2**	66.0
FRCN [ours]	SVM	56.3	58.7	66.8
FRCN [ours]	softmax	57.1	59.2	**66.9**

5.5.Are more proposals always better?

- two types of object detectors: a **sparse** set of object proposals(selective search) and a **dense** set.(DPM)
- *(solid blue line)* More proposals not help and even slightly hurt accuracy.



- *(solid red line)* **Average Recall** is the state-of-the-art for measuring object proposal quality.
  - AR does not correlate well with mAP as the number of proposals per image is varied.
- *(blue triangle)* using densely generated boxe, mAP drop only 1 point.
- *(dotted blue line)* adding a random sample of dense boxes,mAP falls more strongly than only SS boxes.
- *(blue diamond)* train and test using **only** dense boxes.
- *(blue circle)* **SVMs** with hard negative mining.

5.6.Preliminary MS COCO results

- The PASCAL-style mAP is 35.9%; the new COCO-style AP is 19.7%.

6.Conclusion

- Fast R-CNN, a update to R-CNN and SPPnet.
- sparse object proposals appear to improve detector quality.
- maybe dense boxes to perform as well as sparse proposals in furture.

My Reference

RCNN,fast RCNN,faster RCNN 比较归纳总结（一）

https://blog.csdn.net/xiaoye5606/article/details/71191429

Fast RCNN 算法详解

https://blog.csdn.net/u014696921/article/details/53767130

DPM(Deformable Parts Model)--原理(一)

https://blog.csdn.net/carson2005/article/details/22499565