Fast R-CNN

Introduction:

We summarize 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 increasing detection accuracy.

Recently, our deep ConvNet algorithms have been able to get better predictions on image classification problems, but object detection is a more complicated and challenging task. Complexity arises because detection requires the accurate localization of objects, creating two primary challenges. First, we must process numerous candidate object locations (often called “proposals”). Second, we must refine the rough localization given by these candidates to achieve precise localization.

So, we propose a single-stage training algorithm that jointly learns to classify object proposals and refine their spatial locations. But why do we need a new model? Let's see the drawbacks of the earlier used models(i.e., R-CNN, SPPnets).

- ➔ **<u>Training is a multi-stage pipeline</u>**: R-CNN first finetunes a ConvNet on object proposals using log loss. Then, it fits SVMs to ConvNet features. These SVMs act as object detectors, replacing the softmax classifier learned by fine-tuning. In the third training stage, bounding-box regressors are learned.
- ➔ **<u>Training is expensive in space and time:</u>** For SVM and bounding-box regressor training, features are extracted from each object proposal in each image and written to disk.

- ➔ **<u>Object detection is slow</u>**: We extract features from each object proposal in each test image at test time.

<u>Why Fast R-CNN?:</u>

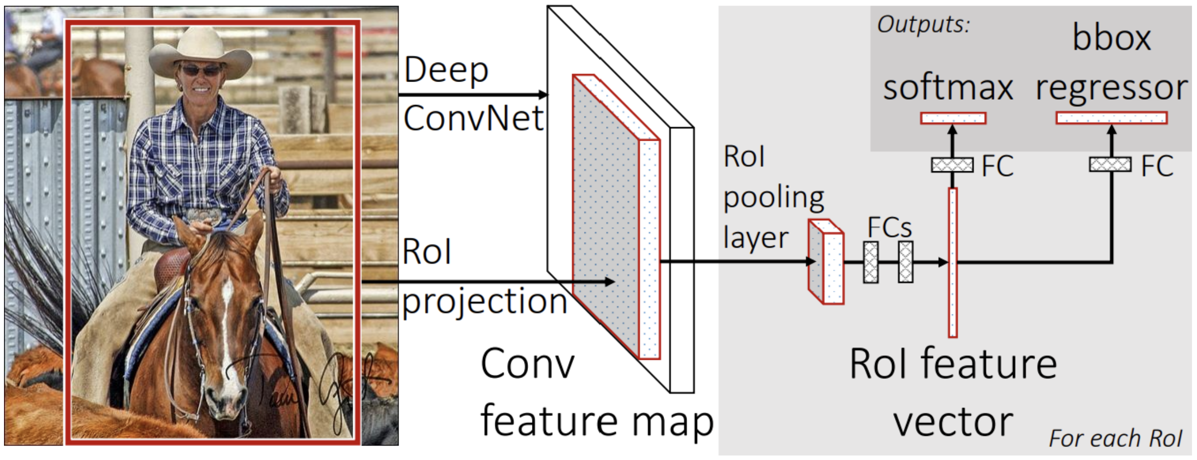
We propose a new training algorithm that fixes the disadvantages of R-CNN and SPPnet while improving their speed and accuracy. We call this method Fast R-CNN because it’s comparatively fast to train and test. The Fast RCNN method has several advantages:

1. Higher detection quality (mAP) than R-CNN, SPPnet.
2. Training is single-stage, using a multi-task loss.
3. Training can update all network layers.
4. No disk storage is required for feature caching.

<u>Architecture:</u>

This network takes as input an entire input image and a set of object proposals. 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 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,
- Another that gives output four real-valued numbers for each of the K object classes.



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 $H \times W$ (e.g., 7×7), where H and W are layer hyper-parameters that are independent of any particular RoI. Here RoI is a rectangular window into a convolutional 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). 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. Pooling is applied to each feature map as in standard max pooling.

Initialization from pre-trained networks:

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 (e.g., *H* = *W* = 7 for VGG16).

Second, the network's last fully connected layer and softmax (which were trained for 1000-way ImageNet classification) are replaced with the two sibling layers described earlier (a fully connected layer and softmax over K + 1 categories and category-specific bounding-box regressors).

Third, the network is modified to take two data inputs: a list of images and a list of RoIs in those images.

Fine-tuning for detection:

We propose a more efficient training method that takes advantage of feature sharing during training. In Fast RCNN training, stochastic gradient descent (SGD) mini batches 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.

Fast R-CNN uses a streamlined training process with one fine-tuning stage that jointly optimizes a softmax classifier and bounding-box regressors rather than training a softmax classifier, SVMs, and regressors in three separate steps.

Multi-task loss:

Fast R-CNN network has two sibling output layers. The first outputs a discrete probability distribution (per RoI), $p = (p_0, \ldots, p_K)$, over $K + 1$ categories. As usual, p is computed by a softmax over the $K+1$ outputs of a fully connected layer. The second sibling layer results in bounding-box regression offsets, $\boldsymbol{t^k = (t^k_x, t^k_y, t^k_w, t^k_h)}$ for each of the K object classes, indexed by k.

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

Where:

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

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

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

- $\boldsymbol{u}$ - a ground-truth class.
- $\boldsymbol{v}$ - a ground-truth bounding-box regression target.
- The Iverson bracket indicator function $\boldsymbol{[u \geq 1]}$ evaluates to 1 when u ≥ 1 and 0 otherwise.

Back-propagation through RoI pooling layers:

Let $x_i \in R$ be the *i-th* activation input into the RoI pooling layer and let y_{rj} be the layer's *j-th* output from the *rth* RoI. The RoI pooling layer computes $\boldsymbol{y_{rj} = x_{i*(r,j)}}$, in which $\boldsymbol{i*(r, j) = \mathrm{argmax}_{i' \in R(r,j)}\, x_{i'}}$. $\boldsymbol{R(r, j)}$ is the index set of inputs in the sub-window over which the output unit y_{rj} max pools. A single xi may be assigned to several different outputs y_{rj}. The RoI pooling layer's backward function computes the partial derivative of the loss function with respect to each input variable xi by following the *arg max* switches:

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

<u>Results:</u>

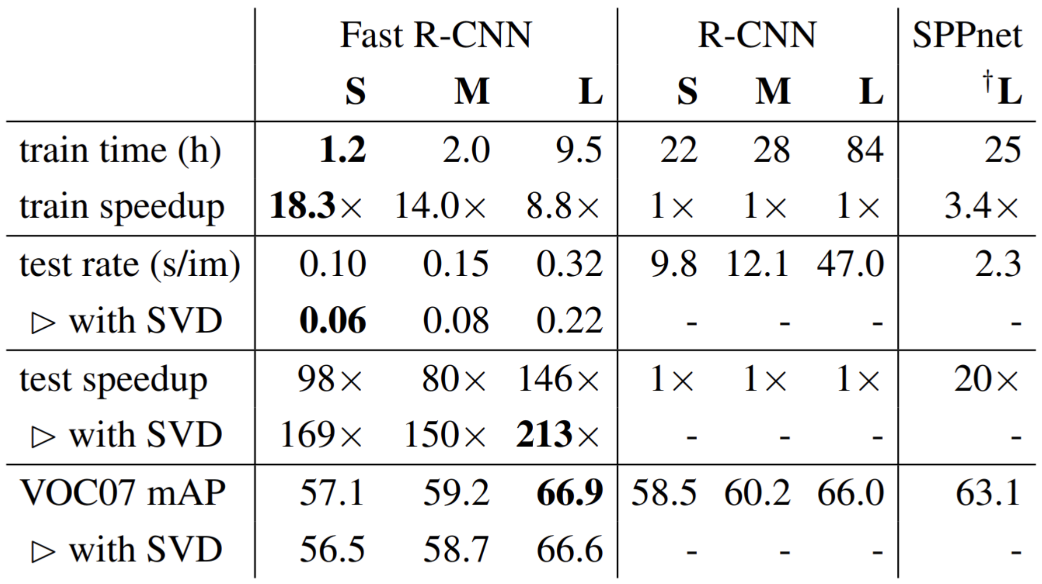
<u>VOC 2010 and 2012 results:</u>

Fast R-CNN achieves the top result on VOC12 with a mAP of 65.7% (and 68.4% with extra data).
On VOC10, it achieved a result with a mAP of 66.1%.

<u>VOC 2007 results:</u>

The improvement of Fast R-CNN over SPPnet illustrates that even though Fast R-CNN uses single-scale training and testing, fine-tuning the convolutional layers greatly improves mAP (from 63.1% to 66.9%).

Training and testing time:

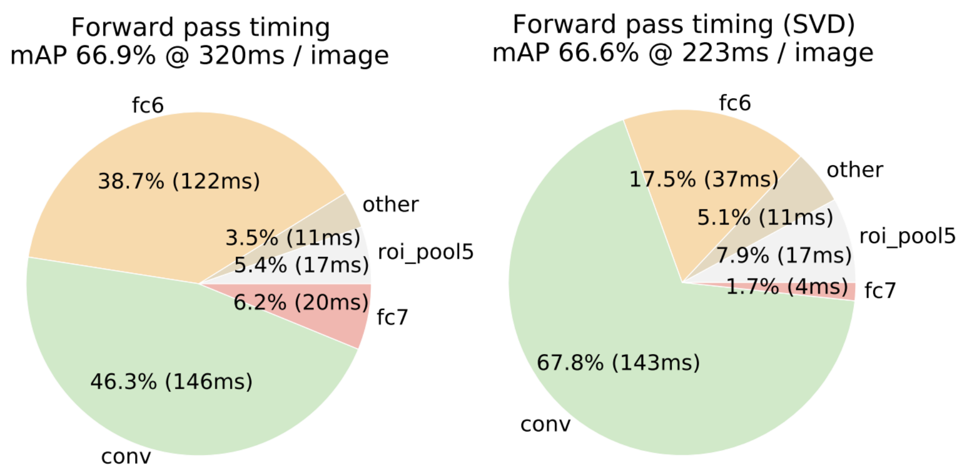


	Fast R-CNN			R-CNN			SPPnet
	S	**M**	**L**	**S**	**M**	**L**	[†]**L**
train time (h)	**1.2**	2.0	9.5	22	28	84	25
train speedup	**18.3**×	14.0×	8.8×	1×	1×	1×	3.4×
test rate (s/im)	0.10	0.15	0.32	9.8	12.1	47.0	2.3
▷ with SVD	**0.06**	0.08	0.22	-	-	-	-
test speedup	98×	80×	146×	1×	1×	1×	20×
▷ with SVD	169×	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	-	-	-	-

Runtime comparison between the Fast RCNN, R-CNN, SPPnets.

Truncated SVD:

Truncated SVD can reduce detection time by more than 30% with only a small (0.3 percentage point) drop in mAP without needing additional fine-tuning after model compression.



In this technique, a layer parameterized by the $u \times v$ weight matrix W is approximately factorized as

$$\mathbf{W} \approx \mathbf{U}\mathbf{\Sigma}_t\mathbf{V}^T$$

using SVD. Truncated SVD reduces the parameter count from $u*v$ to $t*(u+v)$, which can be significant if t is much smaller than $min(u, v)$.

Which layers to fine-tune?

For the less deep networks considered in the SPPnet paper, fine-tuning only the fully connected layers appeared to be sufficient for good accuracy.

To validate that fine-tuning the convolutional layers is important for VGG16, we use Fast R-CNN to fine-tune but freeze the thirteen convolutional layers so that only the fully connected layers learn. *This experiment verifies our hypothesis: training through the RoI pooling layer is important for very deep nets.*

Does multi-task training help?

Multi-task training is convenient because it avoids managing a pipeline of sequentially-trained tasks. But it also has the potential to improve results because the tasks influence each other through a shared representation (the ConvNet).

	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**

Across all three networks, we observe that multi-task training improves pure classification accuracy relative to training for classification alone.

<u>Do SVMs outperform softmax?</u>

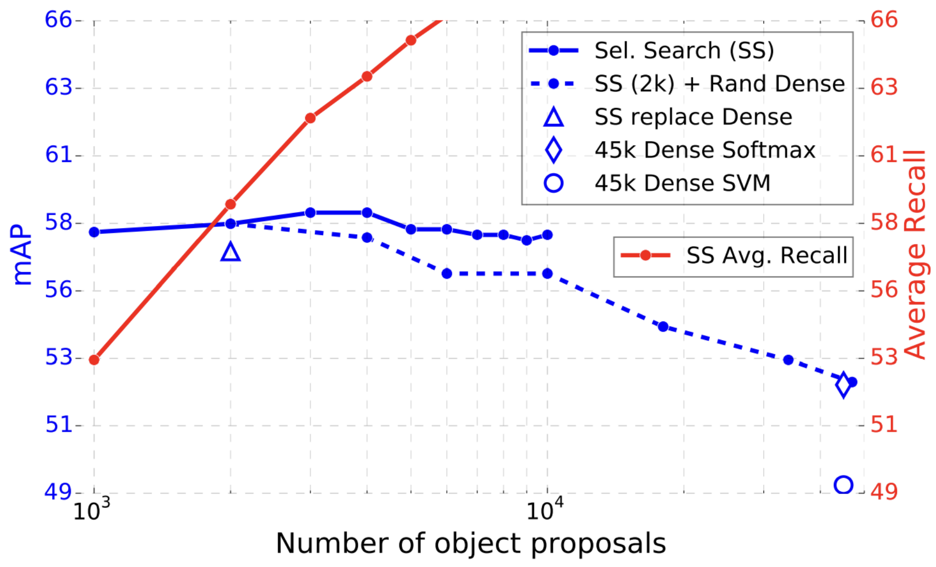
Fast R-CNN uses the softmax classifier learned during fine-tuning instead of training one-vs-rest linear SVMs post-hoc, as was done in R-CNN and SPPnet.

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**

This effect is small, but it demonstrates that "one-shot" fine-tuning is sufficient compared to previous multi-stage training approaches.

<u>Are more proposals always better?</u>

Using selective search's quality mode, we sweep from 1k to 10k proposals per image, each time re-training and retesting model M. If proposals serve a purely computational role, increasing the number of proposals per image should not harm mAP.



We find that mAP rises and then falls slightly as the proposal count increases (Fig. 3, solid blue line).The state-of-the-art for measuring object proposal quality is *Average Recall (AR)*. AR correlates well with mAP for several proposal methods using R-CNN, when using a fixed number of proposals per image.

Preliminary MS COCO result:

We applied Fast R-CNN (with VGG16) to the MS COCO dataset to establish a preliminary baseline. The PASCAL-style mAP is 35.9%; the new COCO-style AP, which also averages over IoU thresholds, is 19.7%.

Conclusion:

This paper proposes Fast R-CNN, a clean and fast update to R-CNN and SPPnet. Of particular note, sparse object proposals appear to improve detector quality.