Training Region-based Object Detectors with Online Hard Example Mining

Abhinav Shrivastava[1] Abhinav Gupta[1] Ross Girshick[2]
[1]Carnegie Mellon University [2]Facebook AI Research
{ashrivas,abhinavg}@cs.cmu.edu rbg@fb.com

Abstract

The field of object detection has made significant advances riding on the wave of region-based ConvNets, but their training procedure still includes many heuristics and hyperparameters that are costly to tune. We present a simple yet surprisingly effective online hard example mining (OHEM) algorithm for training region-based ConvNet detectors. Our motivation is the same as it has always been – detection datasets contain an overwhelming number of easy examples and a small number of hard examples. Automatic selection of these hard examples can make training more effective and efficient. OHEM is a simple and intuitive algorithm that eliminates several heuristics and hyperparameters in common use. But more importantly, it yields consistent and significant boosts in detection performance on benchmarks like PASCAL VOC 2007 and 2012. Its effectiveness increases as datasets become larger and more difficult, as demonstrated by the results on the MS COCO dataset. Moreover, combined with complementary advances in the field, OHEM leads to state-of-the-art results of 78.9% and 76.3% mAP on PASCAL VOC 2007 and 2012 respectively.

1. Introduction

Image classification and object detection are two fundamental computer vision tasks. Object detectors are often trained through a *reduction* that converts object detection into an image classification problem. This reduction introduces a new challenge that is not found in natural image classification tasks: the training set is distinguished by a large imbalance between the number of annotated objects and the number of *background* examples (image regions not belonging to any object class of interest). In the case of sliding-window object detectors, such as the deformable parts model (DPM) [12], this imbalance may be as extreme as 100,000 background examples to every one object. The recent trend towards object-proposal-based detectors [15, 32] mitigates this issue to an extent, but the imbalance ratio may still be high (*e.g.*, 70:1). This challenge opens space for learning techniques that cope with imbalance and yield faster training, higher accuracy, or both.

Unsurprisingly, this is not a new challenge and a standard solution, originally called *bootstrapping* (and now often called *hard negative mining*), has existed for at least 20 years. Bootstrapping was introduced in the work of Sung and Poggio [30] in the mid-1990's (if not earlier) for training face detection models. Their key idea was to gradually grow, or *bootstrap*, the set of background examples by selecting those examples for which the detector triggers a false alarm. This strategy leads to an iterative training algorithm that alternates between updating the detection model given the current set of examples, and then using the updated model to find new false positives to add to the bootstrapped training set. The process typically commences with a training set consisting of all object examples and a small, random set of background examples.

Bootstrapping has seen widespread use in the intervening decades of object detection research. Dalal and Triggs [7] used it when training SVMs for pedestrian detection. Felzenszwalb *et al*. [12] later proved that a form of bootstrapping for SVMs converges to the global optimal solution defined on the entire dataset. Their algorithm is often referred to as *hard negative mining* and is frequently used when training SVMs for object detection [15, 16, 32]. Bootstrapping was also successfully applied to a variety of other learning models, including shallow neural networks [25] and boosted decision trees [9]. Even modern detection methods based on deep convolutional neural networks (ConvNets) [19, 20], such as R-CNN [15] and SPPnet [16], still employ SVMs trained with hard negative mining.

It may seem odd then that the current state-of-the-art object detectors, embodied by Fast R-CNN [14] and its descendants [24], do not use bootstrapping. The underlying reason is a technical difficulty brought on by the shift towards purely online learning algorithms, particularly in the context of deep ConvNets trained with stochastic gradient descent (SGD) on millions of examples. Bootstrapping, and its variants in the literature, rely on the aforementioned alternation template: (a) for some period of time a *fixed* model is used to find new examples to add to the active training set; (b) then, for some period of time the model is trained on the

fixed active training set. Training deep ConvNet detectors with SGD typically requires hundreds of thousands of SGD steps and freezing the model for even a few iterations at a time would dramatically slow progress. What is needed, instead, is a purely online form of hard example selection.

In this paper, we propose a novel bootstrapping technique called *online hard example mining*[1] (OHEM) for training state-of-the-art detection models based on deep ConvNets. The algorithm is a simple modification to SGD in which training examples are sampled according to a non-uniform, non-stationary distribution that depends on the current loss of each example under consideration. The method takes advantage of detection-specific problem structure in which each SGD mini-batch consists of only one or two images, but *thousands* of candidate examples. The candidate examples are subsampled according to a distribution that favors diverse, high loss instances. Gradient computation (backpropagation) is still efficient because it only uses a small subset of all candidates. We apply OHEM to the standard Fast R-CNN detection method and show three benefits compared to the baseline training algorithm:

- It removes the need for several heuristics and hyperparameters commonly used in region-based ConvNets.
- It yields a consistent and significant boosts in mean average precision.
- Its effectiveness increases as the training set becomes larger and more difficult, as demonstrated by results on the MS COCO dataset.

Moreover, the gains from OHEM are complementary to recent improvements in object detection, such as multi-scale testing [16] and iterative bounding-box regression [13]. Combined with these tricks, OHEM gives state-of-the-art results of **78.9%** and **76.3%** mAP on PASCAL VOC 2007 and 2012, respectively.

2. Related work

Object detection is one of the oldest and most fundamental problems in computer vision. The idea of dataset *bootstrapping* [25, 30], typically called *hard negative mining* in recent work [12], appears in the training of most successful object detectors [7, 9, 12, 13, 15, 16, 23, 25, 29]. Many of these approaches use SVMs as the detection scoring function, even after training a deep convolutional neural network (ConvNet) [19, 20] for feature extraction. One notable exception is the Fast R-CNN detector [14] and its descendants, such as Faster R-CNN [24]. Since these models do not use SVMs, and are trained purely online with SGD, existing hard example mining techniques cannot be immediately applied. This work addresses that problem by introducing an online hard example mining algorithm that improves optimization and detection accuracy. We briefly review hard example mining, modern ConvNet-based object detection, and relationships to concurrent works using hard example selection for training deep networks.

Hard example mining. There are two hard example mining algorithms in common use. The first is used when optimizing SVMs. In this case, the training algorithm maintains a working set of examples and alternates between training an SVM to convergence on the working set, and updating the working set by removing some examples and adding others according to a specific rule [12]. The rule removes examples that are "easy" in the sense that they are correctly classified beyond the current model's margin. Conversely, the rule adds new examples that are hard in the sense that they violate the current model's margin. Applying this rule leads to the global SVM solution. Importantly, the working set is usually a small subset of the entire training set.

The second method is used for non-SVMs and has been applied to a variety of models including shallow neural networks [25] and boosted decision trees [9]. This algorithm usually starts with a dataset of positive examples and a random set of negative examples. The machine learning model is then trained to convergence on that dataset and subsequently applied to a larger dataset to harvest false positives. The false positives are then added to the training set and then the model is trained again. This process is usually iterated only once and does not have any convergence proofs.

ConvNet-based object detection. In the last three years significant gains have been made in object detection. These improvements were made possible by the successful application of deep ConvNets [19] to ImageNet classification [8]. The R-CNN [15] and OverFeat [26] detectors lead this wave with impressive results on PASCAL VOC [11] and ImageNet detection. OverFeat is based on the sliding-window detection method, which is perhaps the most intuitive and oldest search method for detection. R-CNN, in contrast, uses region proposals [1, 2, 3, 4, 6, 10, 18, 32, 34], a method that was made popular by the selective search algorithm [32]. Since R-CNN, there has been rapid progress in region-based ConvNets, including SPPnet [16], MR-CNN [13], and Fast R-CNN [14], which our work builds on.

Hard example selection in deep learning. There is recent work [22, 27, 33] concurrent to our own that selects hard examples for training deep networks. Similar to our approach, all these methods base their selection on the current loss for each datapoint. [27] independently selects hard positive and negative example from a larger set of random examples based on their loss to learn image descriptors.

[1] We use the term hard *example* mining, rather than hard *negative* mining, because our method is applied in a multi-class setting to all classes, not just a "negative" class.

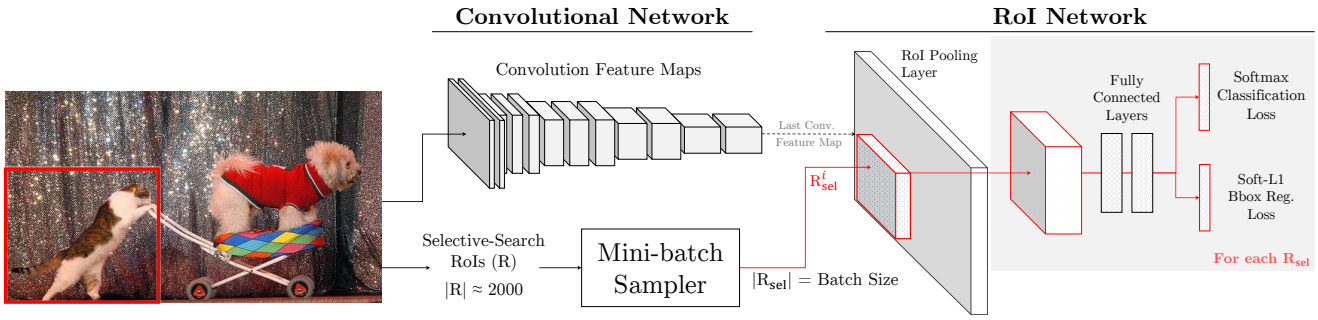

Figure 1: Architecture of the Fast R-CNN approach (see Section 3 for details).

Given a positive pair of patches, [33] finds hard negative patches from a large set using triplet loss. Akin to our approach, [22] investigates online selection of hard examples for mini-batch SGD methods. Their selection is also based on loss, but the focus is on ConvNets for image classification. Complementary to [22], we focus on online hard example selection strategy for region-based object detectors.

3. Overview of Fast R-CNN

We first summarize the Fast R-CNN [14] (FRCN) framework. FRCN takes as input an image and a set of object proposal regions of interest (RoIs). The FRCN network itself can be divided into two sequential parts: a convolutional (*conv*) network with several convolution and max-pooling layers (Figure 1, "Convolutional Network"); and an RoI network with an RoI-pooling layer, several fully-connected (*fc*) layers and two loss layers (Figure 1, "RoI Network").

During inference, the conv network is applied to the given image to produce a conv feature map, size of which depends on the input image dimensions. Then, for each object proposal, the RoI-pooling layer projects the proposal onto the conv feature map and extracts a fixed-length feature vector. Each feature vector is fed into the fc layers, which finally give two outputs: (1) a softmax probability distribution over the object classes and background; and (2) regressed coordinates for bounding-box relocalization.

There are several reasons for choosing FRCN as our base object detector, apart from it being a fast end-to-end system. Firstly, the basic two network setup (conv and RoI) is also used by other recent detectors like SPPnet and MR-CNN; therefore, our proposed algorithm is more broadly applicable. Secondly, though the basic setup is similar, FRCN also allows for training the entire conv network, as opposed to both SPPnet and MR-CNN which keep the conv network fixed. And finally, both SPPnet and MR-CNN require features from the RoI network to be cached for training a separate SVM classifier (using hard negative mining). FRCN uses the RoI network itself to train the desired classifiers. In fact, [14] shows that in the unified system using the SVM classifiers at later stages was unnecessary.

3.1. Training

Like most deep networks, FRCN is trained using stochastic gradient descent (SGD). The loss per example RoI is the sum of a classification log loss that encourages predicting the correct object (or background) label and a localization loss that encourages predicting an accurate bounding box (see [14] for details).

To share conv network computation between RoIs, SGD mini-batches are created hierarchically. For each mini-batch, N images are first sampled from the dataset, and then B/N RoIs are sampled from each image. Setting $N = 2$ and $B = 128$ works well in practice [14]. The RoI sampling procedure uses several heuristics, which we describe briefly below. One contribution of this paper is to eliminate some of these heuristics and their hyperparameters.

Foreground RoIs. For an example RoI to be labeled as foreground (`fg`), its intersection over union (IoU) overlap with a ground-truth bounding box should be at least 0.5. This is a fairly standard design choice, in part inspired by the evaluation protocol of the PASCAL VOC object detection benchmark. The same criterion is used in the SVM hard mining procedures of R-CNN, SPPnet, and MR-CNN. We use the same setting.

Background RoIs. A region is labeled background (`bg`) if its maximum IoU with ground truth is in the interval $[\texttt{bg\_lo}, 0.5)$. A lower threshold of $\texttt{bg\_lo} = 0.1$ is used by both FRCN and SPPnet, and is hypothesized in [14] to crudely approximate hard negative mining; the assumption is that regions with some overlap with the ground truth are more likely to be the confusing or hard ones. We show in Section 5.4 that although this heuristic helps convergence and detection accuracy, it is suboptimal because it ignores some infrequent, but important, difficult background regions. Our method removes the `bg_lo` threshold.

Balancing `fg`-`bg` RoIs: To handle the data imbalance described in Section 1, [14] designed heuristics to rebalance the foreground-to-background ratio in each mini-batch to a target of $1:3$ by undersampling the background patches at random, thus ensuring that 25% of a mini-batch is `fg`

RoIs. We found that this is an important design decision for the training FRCN. Removing this ratio (*i.e*. randomly sampling RoIs), or increasing it, decreases accuracy by $\sim$3 points mAP. With our proposed method, we can remove this ratio hyperparameter with no ill effect.

4. Our approach

We propose a simple yet effective online hard example mining algorithm for training Fast R-CNN (or any Fast R-CNN style object detector). We argue that the current way of creating mini-batches for SGD (Section 3.1) is inefficient and suboptimal, and we demonstrate that our approach leads to better training (lower training loss) and higher testing performance (mAP).

4.1. Online hard example mining

Recall the alternating steps that define a hard example mining algorithm: (a) for some period of time a *fixed* model is used to find new examples to add to the active training set; (b) then, for some period of time the model is trained on the *fixed* active training set. In the context of SVM-based object detectors, such as the SVMs trained in R-CNN or SPPnet, step (a) inspects a variable number of images (often 10's or 100's) until the active training set reaches a threshold size, and then in step (b) the SVM is trained to convergence on the active training set. This process repeats until the active training set contains all support vectors. Applying an analogous strategy to FRCN ConvNet training slows learning because no model updates are made while selecting examples from the 10's or 100's of images.

Our main observation is that these alternating steps can be combined with how FRCN is trained using online SGD. The key is that although each SGD iteration samples only a small number of images, each image contains *thousands* of example RoIs from which we can select the hard examples rather than a heuristically sampled subset. This strategy fits the alternation template to SGD by "freezing" the model for only one mini-batch. Thus the model is updated exactly as frequently as with the baseline SGD approach and therefore learning is not delayed.

More specifically, the online hard example mining algorithm (OHEM) proceeds as follows. For an input image at SGD iteration t, we first compute a conv feature map using the conv network. Then the RoI network uses this feature map and the **all** the input RoIs (R), instead of a sampled mini-batch [14], to do a forward pass. Recall that this step only involves RoI pooling, a few fc layers, and loss computation for each RoI. The loss represents how well the current network performs on each RoI. Hard examples are selected by sorting the input RoIs by loss and taking the B/N examples for which the current network performs worst. Most of the forward computation is shared between RoIs via the conv feature map, so the extra computation needed to forward all RoIs is relatively small. Moreover, because only a small number of RoIs are selected for updating the model, the backward pass is no more expensive than before.

However, there is a small caveat: co-located RoIs with high overlap are likely to have correlated losses. Moreover, these overlapping RoIs can project onto the same region in the conv feature map, because of resolution disparity, thus leading to loss double counting. To deal with these redundant and correlated regions, we use standard non-maximum suppression (NMS) to perform deduplication (the implementation from [14]). Given a list of RoIs and their losses, NMS works by iteratively selecting the RoI with the highest loss, and then removing all lower loss RoIs that have high overlap with the selected region. We use a relaxed IoU threshold of 0.7 to suppress only highly overlapping RoIs.

We note that the procedure described above does not need a `fg-bg` ratio for data balancing. If any class were neglected, its loss would increase until it has a high probability of being sampled. There can be images where the `fg` RoIs are easy (*e.g*. canonical view of a car), so the network is free to use only `bg` regions in a mini-batch; and vice-versa when `bg` is trivial (*e.g*. sky, grass *etc*.), the mini-batch can be entirely `fg` regions.

4.2. Implementation details

There are many ways to implement OHEM in the FRCN detector, each with different trade-offs. An obvious way is to modify the loss layers to do the hard example selection. The loss layer can compute loss for all RoIs, sort them based on this loss to select *hard* RoIs, and finally set the loss of all *non-hard* RoIs to 0. Though straightforward, this implementation is inefficient as the RoI network still allocates memory and performs backward pass for **all** RoIs, even though most RoIs have 0 loss and hence no gradient updates (a limitation of current deep learning toolboxes).

To overcome this, we propose the architecture presented in Figure 2. Our implementation maintains two copies of the RoI network, one of which is *readonly*. This implies that the readonly RoI network (Figure 2(a)) allocates memory only for forward pass of all RoIs as opposed to the standard RoI network, which allocates memory for both forward and backward passes. For an SGD iteration, given the conv feature map, the readonly RoI network performs a forward pass and computes loss for **all** input RoIs (R) (Figure 2, green arrows). Then the hard RoI sampling module uses the procedure described in Section 4.1 to select hard examples $(\mathrm{R}_{\text{hard-sel}})$, which are input to the regular RoI network (Figure 2(b), red arrows)). This network computes forward and backward passes only for $\mathrm{R}_{\text{hard-sel}}$, accumulates the gradients and passes them to the conv network. In practice, we use all RoIs from all N images as R, therefore the effective batch size for the readonly RoI network is $|\mathrm{R}|$ and for the regular RoI network is the standard B from Section 3.1.

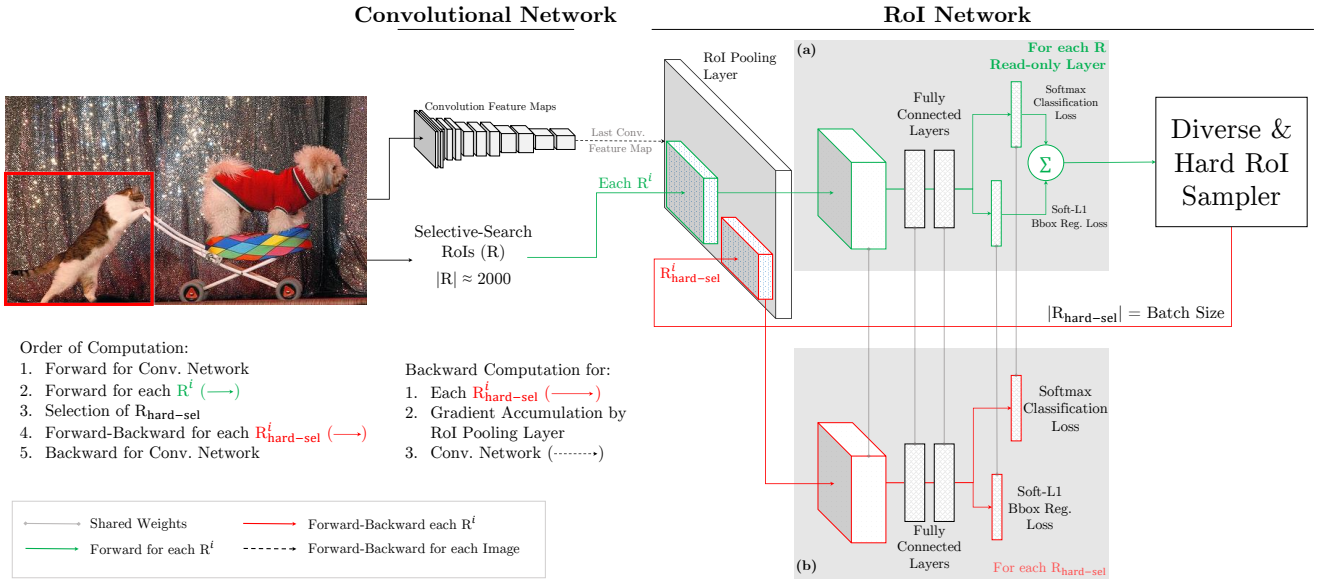

Figure 2: Architecture of the proposed training algorithm. Given an image, and selective search RoIs, the conv network computes a conv feature map. In (a), the *readonly* RoI network runs a forward pass on the feature map and all RoIs (shown in green arrows). Then the Hard RoI module uses these RoI losses to select B examples. In (b), these hard examples are used by the RoI network to compute forward and backward passes (shown in red arrows).

We implement both options described above using the Caffe [17] framework (see [14]). Our implementation uses gradient accumulation with N forward-backward passes of single image mini-batches. Following FRCN [14], we use $N = 2$ (which results in $|\mathrm{R}| \approx 4000$) and $B = 128$. Under these settings, the proposed architecture (Figure 2) has similar memory footprint as the first option, but is $> 2\times$ faster. Unless specified otherwise, the architecture and settings described above will be used throughout this paper.

5. Analyzing online hard example mining

This section compares FRCN training with online hard example mining (OHEM) to the baseline heuristic sampling approach. We also compare FRCN with OHEM to a less efficient approach that uses all available example RoIs in each mini-batch, not just the B hardest examples.

5.1. Experimental setup

We conduct experiments with two standard ConvNet architectures: VGG_CNN_M_1024 (VGGM, for short) from [5], which is a wider version of AlexNet [19], and VGG16 from [28]. All experiments in this section are performed on the PASCAL VOC07 dataset. Training is done on the trainval set and testing on the test set. Unless specified otherwise, we will use the default settings from FRCN [14]. We train all methods with SGD for 80k mini-batch iterations, with an initial learning rate of 0.001 and we decay the learning rate by 0.1 every 30k iterations. The baseline numbers reported in Table 1 (row 1-2) were reproduced using our training schedule and are slightly higher than the ones reported in [14].

5.2. OHEM vs. heuristic sampling

Standard FRCN, reported in Table 1 (rows $1-2$), uses `bg_lo` $= 0.1$ as a heuristic for hard mining (Section 3.1). To test the importance of this heuristic, we ran FRCN with `bg_lo` $= 0$. Table 1 (rows $3-4$) shows that for VGGM, mAP drops by 2.4 points, whereas for VGG16 it remains roughly the same. Now compare this to training FRCN with OHEM (rows $11-13$). OHEM improves mAP by 2.4 points compared to FRCN with the `bg_lo` $= 0.1$ heuristic for VGGM, and 4.8 points without the heuristic. This result demonstrates the sub-optimality of these heuristics and the effectiveness of our hard mining approach.

5.3. Robust gradient estimates

One concern over using only $N = 2$ images per batch is that it may cause unstable gradients and slow convergence because RoIs from an image may be highly correlated [31]. FRCN [14] reports that this was not a practical issue for their training. But this detail might raise concerns over our training procedure because we use examples with high loss from the same image and as a result they may be more highly correlated. To address this concern, we experiment with $N = 1$ in order to increase correlation in an effort to break our method. As seen in Table 1 (rows $5-6, 11$), performance of the original FRCN drops by $\sim$1 point with $N = 1$, but when using our training procedure, mAP remains approximately the same. This shows that OHEM is robust in case one needs fewer images per batch in order to reduce GPU memory usage.

Table 1: Impact of hyperparameters on FRCN training.

	Experiment	Model	N	LR	B	bg_lo	07 mAP
1	Fast R-CNN [14]	VGGM	2	0.001	128	0.1	59.6
2		VGG16					67.2
3	Removing hard mining heuristic (Section 5.2)	VGGM	2	0.001	128	**0**	57.2
4		VGG16					67.5
5	Fewer images per batch (Section 5.3)	VGG16	**1**	0.001	128	0.1	66.3
6						0	66.3
7	Bigger batch, High LR (Section 5.4)	VGGM	1	**0.004**	**2048**	0	57.7
8			2				60.4
9		VGG16	1	**0.003**	**2048**	0	67.5
10			2				68.7
11	Our Approach	VGG16	**1**	0.001	128	0	69.7
12		VGGM	2	0.001	128	0	62.0
13		VGG16					69.9

5.4. Why just hard examples, when you can use all?

Online hard example mining is based on the hypothesis that it is important to consider all RoIs in an image and then select hard examples for training. But what if we train with all the RoIs, not just the hard ones? The easy examples will have low loss, and won't contribute much to the gradient; training will automatically focus on the hard examples. To compare this option, we ran standard FRCN training with a large mini-batch size of $B = 2048$, using $\texttt{bg\_lo} = 0$, $N \in \{1, 2\}$ and with other hyperparameters fixed. Because this experiment uses a large mini-batch, it's important to tune the learning rate to adjust for this change. We found optimal results by increasing it to 0.003 for VGG16 and 0.004 for VGGM. The outcomes are reported in Table 1 (rows $7-10$). Using these settings, mAP of both VGG16 and VGGM increased by $\sim$1 point compared to $B = 128$, but the improvement from our approach is still > 1 points over using all RoIs. Moreover, because we compute gradients with a smaller mini-batch size training is faster.

5.5. Better optimization

Finally, we analyze the training loss for the various FRCN training methods discussed above. It's important to measure training loss in a way that does not depend on the sampling procedure and thus results in a valid comparison between methods. To achieve this goal, we take model snapshots from each method every 20k steps of optimization and run them over the entire VOC07 trainval set to compute the average loss over *all* RoIs. This measures the training set loss in a way that does not depend on the example sampling scheme.

Figure 3 shows the average loss per RoI for VGG16 with the various hyperparameter settings discussed above and presented in Table 1. We see that $\texttt{bg\_lo} = 0$ results in the highest training loss, while using the heuristic $\texttt{bg\_lo} = 0.1$ results in a much lower training loss. Increasing the mini-batch size to $B = 2048$ and increasing the learning rate lowers the training loss below the $\texttt{bg\_lo} = 0.1$ heuristic. Our proposed online hard example mining method achieves the lowest training loss of all methods, validating our claims that OHEM leads to better training for FRCN.

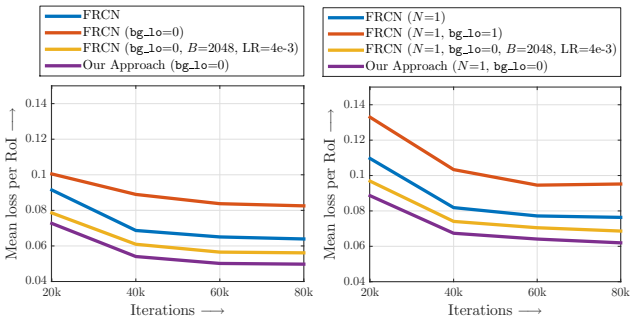

Figure 3: Training loss is computed for various training procedures using VGG16 networks discussed in Section 5. We report mean loss per RoI. These results indicate that using hard mining for training leads to lower training loss than any of the other heuristics.

Table 2: Computational statistics of training FRCN [14] and FRCN with OHEM (using an Nvidia Titan X GPU).

	VGGM		VGG16		
	FRCN	Ours	FRCN	FRCN*	Ours*
time (sec/iter)	0.13	0.22	0.60	0.57	1.00
max. memory (G)	2.6	3.6	11.2	6.4	8.7

*: uses gradient accumulation over two forward/backward passes

5.6. Computational cost

OHEM adds reasonable computational and memory overhead, as reported in Table 2. OHEM costs 0.09s per training iteration for VGGM network (0.43s for VGG16) and requires 1G more memory (2.3G for VGG16). Given that FRCN [14] is a fast detector to train, the increase in training time is likely acceptable to most users.

6. PASCAL VOC and MS COCO results

In this section, we evaluate our method on VOC 2012 [11] as well as the more challenging MS COCO [21] dataset. We demonstrate consistent and significant improvement in FRCN performance when using the proposed OHEM approach. Per-class results are also presented on VOC 2007 for comparison with prior work.

Experimental setup. We use VGG16 for all experiments. When training on VOC07 trainval, we use the SGD parameters as in Section 5 and when using extra data (07+12 and 07++12, see Table 3 and 4), we use 200k mini-batch iterations, with an initial learning rate of 0.001 and decay step size of 40k. When training on MS COCO [21], we use 240k

Table 3: **VOC 2007 test** detection average precision (%). All methods use VGG16. Training set key: **07**: VOC07 trainval, **07+12**: union of **07** and VOC12 trainval. All methods use bounding-box regression. Legend: **M**: using multi-scale for training and testing, **B**: multi-stage bbox regression. FRCN⋆ refers to FRCN [14] with our training schedule.

method	M	B	train set	mAP	aero	bike	bird	boat	bottle	bus	car	cat	chair	cow	table	dog	horse	mbike	persn	plant	sheep	sofa	train	tv
FRCN [14]			07	66.9	74.5	78.3	69.2	53.2	36.6	77.3	78.2	82.0	40.7	72.7	67.9	79.6	79.2	73.0	69.0	30.1	65.4	70.2	75.8	65.8
FRCN⋆			07	67.2	74.6	76.8	67.6	52.9	37.8	78.7	78.8	81.6	42.2	73.6	67.0	79.4	79.6	74.1	68.3	33.4	65.9	68.7	75.4	68.1
Ours			07	**69.9**	71.2	78.3	69.2	57.9	46.5	81.8	79.1	83.2	47.9	76.2	68.9	83.2	80.8	75.8	72.7	39.9	67.5	66.2	75.6	75.9
FRCN⋆	✓	✓	07	72.4	77.8	81.3	71.4	60.4	48.3	85.0	84.6	86.2	49.4	80.7	68.1	84.1	86.7	80.2	75.3	38.7	71.9	71.5	77.9	67.8
MR-CNN [13]	✓	✓	07	74.9	78.7	81.8	76.7	66.6	61.8	81.7	85.3	82.7	57.0	81.9	73.2	84.6	86.0	80.5	74.9	44.9	71.7	69.7	78.7	79.9
Ours	✓	✓	07	**75.1**	77.7	81.9	76.0	64.9	55.8	86.3	86.0	86.8	53.2	82.9	70.3	85.0	86.3	78.7	78.0	46.8	76.1	72.7	80.9	75.5
FRCN [14]			07+12	70.0	77.0	78.1	69.3	59.4	38.3	81.6	78.6	86.7	42.8	78.8	68.9	84.7	82.0	76.6	69.9	31.8	70.1	74.8	80.4	70.4
Ours			07+12	**74.6**	77.7	81.2	74.1	64.2	50.2	86.2	83.8	88.1	55.2	80.9	73.8	85.1	82.6	77.8	74.9	43.7	76.1	74.2	82.3	79.6
MR-CNN [13]	✓	✓	07+12	78.2	80.3	84.1	78.5	**70.8**	**68.5**	88.0	85.9	87.8	**60.3**	**85.2**	73.7	**87.2**	86.5	**85.0**	76.4	48.5	76.3	75.5	**85.0**	**81.0**
Ours	✓	✓	07+12	**78.9**	**80.6**	**85.7**	**79.8**	69.9	60.8	**88.3**	**87.9**	**89.6**	59.7	85.1	**76.5**	87.1	**87.3**	82.4	**78.8**	**53.7**	**80.5**	**78.7**	84.5	80.7

Table 4: **VOC 2012 test** detection average precision (%). All methods use VGG16. Training set key: **12**: VOC12 trainval, **07++12**: union of VOC07 trainval, VOC07 test, and VOC12 trainval. Legend: **M**: using multi-scale for training and testing, **B**: iterative bbox regression.

method	M	B	train set	mAP	aero	bike	bird	boat	bottle	bus	car	cat	chair	cow	table	dog	horse	mbike	persn	plant	sheep	sofa	train	tv
FRCN [14]			12	65.7	80.3	74.7	66.9	46.9	37.7	73.9	68.6	87.7	41.7	71.1	51.1	86.0	77.8	79.8	69.8	32.1	65.5	63.8	76.4	61.7
Ours[1]			12	**69.8**	81.5	78.9	69.6	52.3	46.5	77.4	72.1	88.2	48.8	73.8	58.3	86.9	79.7	81.4	75.0	43.0	69.5	64.8	78.5	68.9
MR-CNN [13]	✓	✓	12	70.7	85.0	79.6	71.5	55.3	57.7	76.0	73.9	84.6	50.5	74.3	61.7	85.5	79.9	81.7	76.4	41.0	69.0	61.2	77.7	72.1
Ours[2]	✓	✓	12	**72.9**	85.8	82.3	74.1	55.8	55.1	79.5	77.7	90.4	52.1	75.5	58.4	88.6	82.4	83.1	78.3	47.0	77.2	65.1	79.3	70.4
FRCN [14]			07++12	68.4	82.3	78.4	70.8	52.3	38.7	77.8	71.6	89.3	44.2	73.0	55.0	87.5	80.5	80.8	72.0	35.1	68.3	65.7	80.4	64.2
Ours[3]			07++12	**71.9**	83.0	81.3	72.5	55.6	49.0	78.9	74.7	89.5	52.3	75.0	61.0	87.9	80.9	82.4	76.3	47.1	72.5	67.3	80.6	71.2
MR-CNN [13]	✓	✓	07++12	73.9	85.5	82.9	76.6	57.8	**62.7**	79.4	77.2	86.6	55.0	79.1	62.2	87.0	83.4	84.7	78.9	45.3	73.4	65.8	80.3	74.0
Ours[4]	✓	✓	07++12	**76.3**	**86.3**	**85.0**	**77.0**	**60.9**	59.3	**81.9**	**81.1**	**91.9**	**55.8**	**80.6**	**63.0**	**90.8**	**85.1**	**85.3**	**80.7**	**54.9**	**78.3**	**70.8**	**82.8**	**74.9**

[1] http://host.robots.ox.ac.uk:8080/anonymous/XNDVK7.html, [2] http://host.robots.ox.ac.uk:8080/anonymous/H49PTT.html, [3] http://host.robots.ox.ac.uk:8080/anonymous/LSANTB.html, [4] http://host.robots.ox.ac.uk:8080/anonymous/R7EAMX.html

mini-batch iterations, with an initial learning rate of 0.001 and decay step size of 160k, owing to a larger epoch size.

6.1. VOC 2007 and 2012 results

Table 3 shows that on VOC07, OHEM improves the mAP of FRCN from 67.2% to 69.9% (and 70.0% to 74.6% with extra data). On VOC12, OHEM leads to an improvement of 4.1 points in mAP (from 65.7% to 69.8%). With extra data, we achieve an mAP of 71.9% as compared to 68.4% mAP of FRCN, an improvement of 3.5 points. Interestingly the improvements are not uniform across categories. Bottle, chair, and tvmonitor show larger improvements that are consistent across the different PASCAL splits. Why these classes benefit the most is an interesting and open question.

6.2. MS COCO results

To test the benefit of using OHEM on a larger and more challenging dataset, we conduct experiments on MS COCO [21] and report numbers from test-dev 2015 evaluation server (Table 5). On the standard COCO evaluation metric, FRCN [14] scores 19.7% AP, and OHEM improves it to 22.6% AP.[2] Using the VOC overlap metric of IoU ≥ 0.5, OHEM gives a 6.6 points boost in AP^{50}. It is also interesting to note that OHEM helps improve the AP of medium sized objects by 4.9 points on the strict COCO AP evaluation metric, which indicates that the proposed hard example mining approach is helpful when dealing with smaller sized objects. Note that FRCN with and without OHEM were trained on MS COCO train set.

[2] COCO AP averages over classes, recall, and IoU levels. See http://mscoco.org/dataset/#detections-eval for details.

7. Adding bells and whistles

We've demonstrated consistent gains in detection accuracy by applying OHEM to FRCN training. In this section, we show that these improvements are orthogonal to recent bells and whistles that enhance object detection accuracy. OHEM with the following two additions yields state-of-the-art results on VOC and competitive results on MS COCO.

Multi-scale (M). We adopt the multi-scale strategy from SPPnet [16] (and used by both FRCN [14] and MR-CNN [13]). Scale is defined as the size of the shortest side (s) of an image. During training, one scale is chosen at random, whereas at test time inference is run on all scales. For VGG16 networks, we use $s \in \{480, 576, 688, 864, 900\}$ for training, and $s \in \{480, 576, 688, 864, 1000\}$ during testing, with the max dimension capped at 1000. The scales and caps were chosen because of GPU memory constraints.

Table 5: **MS COCO 2015 test−dev** detection average precision (%). All methods use VGG16. Legend: **M**: using multi-scale for training and testing.

AP@IoU	area	FRCN†	**Ours**	**Ours** [+M]	**Ours*** [+M]
[0.50 : 0.95]	all	19.7	22.6	24.4	25.5
0.50	all	35.9	42.5	44.4	45.9
0.75	all	19.9	22.2	24.8	26.1
[0.50 : 0.95]	small	3.5	5.0	7.1	7.4
[0.50 : 0.95]	med.	18.8	23.7	26.4	27.7
[0.50 : 0.95]	large	34.6	37.9	38.5	40.3

†from the leaderboard, *trained on trainval set

Iterative bounding-box regression (B). We adopt the iterative localization and bounding-box (bbox) voting scheme from [13]. The network evaluates each proposal RoI to get scores and relocalized boxes R_1. High-scoring R_1 boxes are the rescored and relocalized, yielding boxes R_2. Union of R_1 and R_2 is used as the final set R_F for post-processing, where $\mathrm{R}_\mathrm{F}^\mathrm{NMS}$ is obtained using NMS on R_F with an IoU threshold of 0.3 and weighted voting is performed on each box r_i in $\mathrm{R}_\mathrm{F}^\mathrm{NMS}$ using boxes in R_F with an IoU of $\geq$0.5 with r_i (see [13] for details).

7.1. VOC 2007 and 2012 results

We report the results on VOC benchmarks in Table 3 and 4. On VOC07, FRCN with the above mentioned additions achieves 72.4% mAP and OHEM improves it to **75.1%**, which is currently the highest reported score under this setting (07 data). When using extra data (07+12), OHEM achieves **78.9%** mAP, surpassing the current state-of-the-art MR-CNN (78.2% mAP). We note that MR-CNN uses selective search and edge boxes during training, whereas we only use selective search boxes. Our multi-scale implementation is also different, using fewer scales than MR-CNN. On VOC12 (Table 4), we consistently perform better than MR-CNN. When using extra data, we achieve state-of-the-art mAP of **76.3%** (*vs*. 73.9% mAP of MR-CNN).

Ablation analysis. We now study in detail the impact of these two additions and whether OHEM is complementary to them, and report the analysis in Table 6. Baseline FRCN mAP improves from 67.2% to 68.6% when using multi-scale during both training and testing (we refer to this as **M**). However, note that there is only a marginal benefit of using it at training time. Iterative bbox regression (**B**) further improves the FRCN mAP to 72.4%. But more importantly, using OHEM improves it to **75.1%** mAP, which is state-of-the-art for methods trained on VOC07 data (see Table 3). In fact, using OHEM consistently results in higher mAP for all variants of these two additions (see Table 6).

Table 6: Impact of multi-scale and iterative bbox reg.

Multi-scale (**M**)		Iterative bbox reg. (**B**)	VOC07 mAP	
Train	Test		FRCN	**Ours**
			67.2	69.9
	✓		68.4	71.1
		✓	70.8	72.7
	✓	✓	71.9	74.1
✓			67.7	70.7
✓	✓		68.6	71.9
✓		✓	71.2	72.9
✓	✓	✓	72.4	**75.1**

7.2. MS COCO results

MS COCO [21] test-dev 2015 evaluation server results are reported in Table 5. Using multi-scale improves the performance of our method to 24.4% AP on the standard COCO metric and to 44.4% AP^{50} on the VOC metric. This again shows the complementary nature of using multi-scale and OHEM. Finally, we train our method using the entire MS COCO trainval set, which further improves performance to **25.5%** AP (and 45.9% AP^{50}). In the 2015 MS COCO Detection Challenge, a variant of this approach finished 4^{th} place overall.

8. Conclusion

We presented an online hard example mining (OHEM) algorithm, a simple and effective method to train region-based ConvNet detectors. OHEM eliminates several heuristics and hyperparameters in common use by automatically selecting hard examples, thus simplifying training. We conducted extensive experimental analysis to demonstrate the effectiveness of the proposed algorithm, which leads to better training convergence and consistent improvements in detection accuracy on standard benchmarks. We also reported state-of-the-art results on PASCAL VOC 2007 and 2012 when using OHEM with other orthogonal additions. Though we used Fast R-CNN throughout this paper, OHEM can be used for training any region-based ConvNet detector.

Our experimental analysis was based on the overall detection accuracy, however it will be an interesting future direction to study the impact of various training methodologies on individual category performance.

Acknowledgment. This project started as an intern project at Microsoft Research and continued at CMU. We thank Larry Zitnick, Ishan Misra and Sean Bell for many helpful discussions. AS was supported by the Microsoft Research PhD Fellowship. This work was also partially supported by ONR MURI N000141612007. We thank NVIDIA for donating GPUs.

References

[1] B. Alexe, T. Deselaers, and V. Ferrari. What is an object? In *CVPR*, 2010.

[2] B. Alexe, T. Deselaers, and V. Ferrari. Measuring the objectness of image windows. *TPAMI*, 2012.

[3] P. Arbeláez, J. Pont-Tuset, J. T. Barron, F. Marques, and J. Malik. Multiscale combinatorial grouping. In *CVPR*, 2014.

[4] J. Carreira and C. Sminchisescu. Constrained parametric min-cuts for automatic object segmentation. In *CVPR*, 2010.

[5] K. Chatfield, K. Simonyan, A. Vedaldi, and A. Zisserman. Return of the devil in the details: Delving deep into convolutional nets. In *BMVC*, 2014.

[6] M.-M. Cheng, Z. Zhang, W.-Y. Lin, and P. H. S. Torr. BING: Binarized normed gradients for objectness estimation at 300fps. In *CVPR*, 2014.

[7] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In *CVPR*, 2005.

[8] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. Imagenet: A large-scale hierarchical image database. In *CVPR*, 2009.

[9] P. Dollár, Z. Tu, P. Perona, and S. Belongie. Integral channel features. In *BMVC*, 2009.

[10] I. Endres and D. Hoiem. Category independent object proposals. In *ECCV*, 2010.

[11] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The pascal visual object classes (voc) challenge. *IJCV*, 2010.

[12] P. Felzenszwalb, R. Girshick, D. McAllester, and D. Ramanan. Object detection with discriminatively trained part-based models. *PAMI*, 2010.

[13] S. Gidaris and N. Komodakis. Object detection via a multi-region & semantic segmentation-aware cnn model. In *ICCV*, 2015.

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

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

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

[17] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell. Caffe: Convolutional architecture for fast feature embedding. *arXiv preprint arXiv:1408.5093*, 2014.

[18] P. Krähenbühl and V. Koltun. Geodesic object proposals. In *ECCV*. 2014.

[19] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *NIPS*, 2012.

[20] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel. Backpropagation applied to handwritten zip code recognition. *Neural computation*, 1989.

[21] T. Lin, M. Maire, S. Belongie, L. D. Bourdev, R. B. Girshick, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick. Microsoft COCO: common objects in context. *CoRR*, abs/1405.0312, 2014.

[22] I. Loshchilov and F. Hutter. Online batch selection for faster training of neural networks. *arXiv preprint arXiv:1511.06343*, 2015.

[23] T. Malisiewicz, A. Gupta, and A. A. Efros. Ensemble of exemplar-svms for object detection and beyond. In *ICCV*, 2011.

[24] S. Ren, K. He, R. Girshick, and J. Sun. Faster R-CNN: Towards real-time object detection with region proposal networks. In *Neural Information Processing Systems (NIPS)*, 2015.

[25] H. Rowley, S. Baluja, and T. Kanade. Neural network-based face detection. *IEEE PAMI*, 1998.

[26] P. Sermanet, D. Eigen, X. Zhang, M. Mathieu, R. Fergus, and Y. LeCun. Overfeat: Integrated recognition, localization and detection using convolutional networks. *CoRR*, abs/1312.6229, 2013.

[27] E. Simo-Serra, E. Trulls, L. Ferraz, I. Kokkinos, and F. Moreno-Noguer. Fracking deep convolutional image descriptors. *arXiv preprint arXiv:1412.6537*, 2014.

[28] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *CoRR*, abs/1409.1556, 2014.

[29] S. Singh, A. Gupta, and A. A. Efros. Unsupervised discovery of mid-level discriminative patches. In *European Conference on Computer Vision*, 2012.

[30] K.-K. Sung and T. Poggio. Learning and Example Selection for Object and Pattern Detection. In *MIT A.I. Memo No. 1521*, 1994.

[31] M. Takáč, A. Bijral, P. Richtárik, and N. Srebro. Mini-batch primal and dual methods for svms. *arXiv preprint arXiv:1303.2314*, 2013.

[32] J. Uijlings, K. van de Sande, T. Gevers, and A. Smeulders. Selective search for object recognition. *IJCV*, 2013.

[33] X. Wang and A. Gupta. Unsupervised learning of visual representations using videos. In *ICCV*, 2015.

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