Is Faster R-CNN Doing Well for Pedestrian Detection?

Liliang Zhang[1] Liang Lin[1][⋆] Xiaodan Liang[1] Kaiming He[2]

[1]School of Data and Computer Science, Sun Yat-sen University
zhangll.level0@gmail.com; linliang@ieee.org; xdliang328@gmail.com
[2]Microsoft Research
kahe@microsoft.com

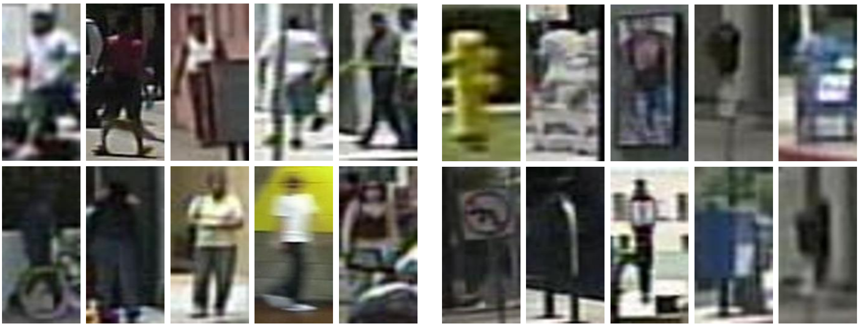
Abstract. Detecting pedestrian has been arguably addressed as a special topic beyond general object detection. Although recent deep learning object detectors such as Fast/Faster R-CNN [1,2] have shown excellent performance for general object detection, they have limited success for detecting pedestrian, and previous leading pedestrian detectors were in general hybrid methods combining hand-crafted and deep convolutional features. In this paper, we investigate issues involving Faster R-CNN [2] for pedestrian detection. We discover that the Region Proposal Network (RPN) in Faster R-CNN indeed performs well as a stand-alone pedestrian detector, but surprisingly, the downstream classifier degrades the results. We argue that two reasons account for the unsatisfactory accuracy: (i) insufficient resolution of feature maps for handling small instances, and (ii) lack of any bootstrapping strategy for mining hard negative examples. Driven by these observations, we propose a very simple but effective baseline for pedestrian detection, using an RPN followed by boosted forests on shared, high-resolution convolutional feature maps. We comprehensively evaluate this method on several benchmarks (Caltech, INRIA, ETH, and KITTI), presenting competitive accuracy and good speed. Code will be made publicly available.

Keywords: Pedestrian Detection, Convolutional Neural Networks, Boosted Forests, Hard-negative Mining

1 Introduction

Pedestrian detection, as a key component of real-world applications such as automatic driving and intelligent surveillance, has attracted special attention beyond general object detection. Despite the prevalent success of deeply learned features in computer vision, current leading pedestrian detectors (*e.g.*, [3,4,5,6]) are in general *hybrid* methods that combines traditional, hand-crafted features [7,8] and deep convolutional features [9,10]. For example, in [3] a stand-alone pedestrian detector [11] (that uses Squares Channel Features) is adopted as a highly selective proposer (<3 regions per image), followed by R-CNN [12]

[⋆] The corresponding author is Liang Lin.



(a) Small positive instances

(b) Hard negatives

Fig. 1: Two challenges for Fast/Faster R-CNN in pedestrian detection. (a) Small objects that may fail RoI pooling on low-resolution feature maps. (b) Hard negative examples that receive no careful attention in Fast/Faster R-CNN.

for classification. Hand-crafted features appear to be of critical importance for state-of-the-art pedestrian detection.

On the other hand, Faster R-CNN [2] is a particularly successful method for general object detection. It consists of two components: a fully convolutional Region Proposal Network (RPN) for proposing candidate regions, followed by a downstream Fast R-CNN [1] classifier. The Faster R-CNN system is thus a purely CNN-based method without using hand-crafted features (*e.g.*, Selective Search [13] that is based on low-level features). Despite its leading accuracy on several multi-category benchmarks, Faster R-CNN has not presented competitive results on popular pedestrian detection datasets (*e.g.*, the Caltech set [14]).

In this paper, we investigate the issues involving Faster R-CNN as a pedestrian detector. Interestingly, we find that an RPN specially tailored for pedestrian detection achieves competitive results as a stand-alone pedestrian detector. But surprisingly, the accuracy is degraded after feeding these proposals into the Fast R-CNN classifier. We argue that such unsatisfactory performance is attributed to two reasons as follows.

First, the convolutional feature maps of the Fast R-CNN classifier are of low solution for detecting small objects. Typical scenarios of pedestrian detection, such as automatic driving and intellegent surveillance, generally present pedestrian instances of small sizes (*e.g.*, 28×70 for Caltech [14]). On small objects (Fig. 1(a)), the Region-of-Interest (RoI) pooling layer [15,1] performed on a low-resolution feature map (usually with a stride of 16 pixels) can lead to "plain" features caused by collapsing bins. These features are not discriminative on small regions, and thus degrade the downstream classifier. We note that this is in contrast to hand-crafted features that have finer resolutions. We address this problem by pooling features from shallower but higher-resolution layers, and by the hole algorithm (namely, "à trous" [16] or filter rarefaction [17]) that increases feature map size.

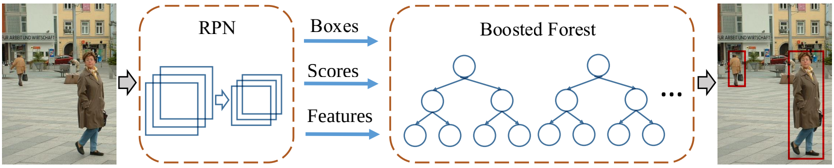


Fig. 2: Our pipeline. RPN is used to compute candidate bounding boxes, scores, and convolutional feature maps. The candidate boxes are fed into cascaded Boosted Forests (BF) for classification, using the features pooled from the convolutional feature maps computed by RPN.

Second, in pedestrian detection the false predictions are dominantly caused by confusions of hard *background* instances (Fig. 1(b)). This is in contrast to general object detection where a main source of confusion is from *multiple categories*. To address hard negative examples, we adopt cascaded Boosted Forest (BF) [18,19], which performs effective hard negative mining (bootstrapping) and sample re-weighting, to classify the RPN proposals. Unlike previous methods that use hand-crafted features to train the forest, in our method the BF *reuses* the deep convolutional features of RPN. This strategy not only reduces the computational cost of the classifier by sharing features, but also exploits the deeply learned features.

As such, we present a surprisingly simple but effective baseline for pedestrian detection based on RPN and BF. Our method overcomes two limitations of Faster R-CNN for pedestrian detection and gets rid of traditional hand-crafted features. We present compelling results on several benchmarks, including Caltech [14], INRIA [20], ETH [21], and KITTI [22]. Remarkably, our method has substantially better localization accuracy and shows a relative improvement of 40% on the Caltech dataset under an Intersection-over-Union (IoU) threshold of 0.7 for evaluation. Meanwhile, our method has a test-time speed of 0.5 second per image, which is competitive with previous leading methods.

In addition, our paper reveals that traditional pedestrian detectors have been inherited in recent methods at least for two reasons. First, the higher resolution of hand-crafted features (such as [7,8]) and their pyramids is good for detecting small objects. Second, effective bootstrapping is performed for mining hard negative examples. These key factors, however, when appropriately handled in a deep learning system, lead to excellent results.

2 Related Work

The Integrate Channel Features (ICF) detector [7], which extends the Viola-Jones framework [23], is among the most popular pedestrian detectors without using deep learning features. The ICF detector involves channel feature pyramids and boosted classifiers. The feature representations of ICF have been improved in several ways, including ACF [8], LDCF [24], SCF [11], and many others, but the boosting algorithm remains a key building block for pedestrian detection.

Driven by the success of ("slow") R-CNN [12] for general object detection, a recent series of methods [11,4,5] adopt a two-stage pipeline for pedestrian detection. In [3], the SCF pedestrian detector [11] is used to propose regions, followed by an R-CNN for classification; TA-CNN [4] employs the ACF detector [8] to generate proposals, and trains an R-CNN-style network to jointly optimize pedestrian detection with semantic tasks; the DeepParts method [5] applies the LDCF detector [24] to generate proposals and learns a set of complementary parts by neural networks. We note that these proposers are stand-alone pedestrian detectors consisting of hand-crafted features and boosted classifiers.

Unlike the above R-CNN-based methods, the CompACT method [6] learns boosted classifiers on top of hybrid hand-crafted and deep convolutional features. Most closely related to our work, the CCF detector [25] is boosted classifiers on pyramids of deep convolutional features, but uses no region proposals. Our method has no pyramid, and is much faster and more accurate than [25].

3 Approach

Our approach consists of two components (illustrated in Fig. 2): an RPN that generates candidate boxes as well as convolutional feature maps, and a Boosted Forest that classifies these proposals using these convolutional features.

3.1 Region Proposal Network for Pedestrian Detection

The RPN in Faster R-CNN [2] was developed as a class-agnostic detector (proposer) in the scenario of multi-category object detection. For single-category detection, RPN is naturally a detector for the only category concerned. We specially tailor the RPN for pedestrian detection, as introduced in the following.

We adopt anchors (reference boxes) [2] of a single aspect ratio of 0.41 (width to height). This is the average aspect ratio of pedestrians as indicated in [14]. This is unlike the original RPN [2] that has anchors of multiple aspect ratios. Anchors of inappropriate aspect ratios are associated with few examples, so are noisy and harmful for detection accuracy. In addition, we use anchors of 9 different scales, starting from 40 pixels height with a scaling stride of 1.3×. This spans a wider range of scales than [2]. The usage of multi-scale anchors waives the requirement of using feature pyramids to detect multi-scale objects.

Following [2], we adopt the VGG-16 net [10] pre-trained on the ImageNet dataset [26] as the backbone network. The RPN is built on top of the Conv5_3 layer, which is followed by an intermediate 3×3 convolutional layer and two sibling 1×1 convolutional layers for classification and bounding box regression (more details in [2]). In this way, RPN regresses boxes with a stride of 16 pixels (Conv5_3). The classification layer provides confidence scores of the predicted boxes, which can be used as the initial scores of the Boosted Forest cascade that follows.

It is noteworthy that although we will use the "à trous" [16] trick in the following section to increase resolution and reduce stride, we keep using the same RPN with a stride of 16 pixels. The à trous trick is only exploited when extracting features (as introduced next), but not for fine-tuning.

3.2 Feature Extraction

With the proposals generated by RPN, we adopt RoI pooling [1] to extract fixed-length features from regions. These features will be used to train BF as introduced in the next section. Unlike Faster R-CNN which requires to feed these features into the *original fully-connected* (fc) layers and thus limits their dimensions, the BF classifier imposes no constraint on the dimensions of features. For example, we can extract features from RoIs on Conv3_3 (of a stride = 4 pixels) and Conv4_3 (of a stride = 8 pixels). We pool the features into a fixed resolution of 7×7. These features from different layers are simply concatenated without normalization, thanks to the flexibility of the BF classifier; on the contrast, feature normalization needs to be carefully addressed [27] for deep classifiers when concatenating features.

Remarkably, as there is no constraint imposed to feature dimensions, it is flexible for us to use features of increased resolution. In particular, given the fine-tuned layers from RPN (stride = 4 on Conv3, 8 on Conv4, and 16 on Conv5), we can use the à trous trick [16] to compute convolutional feature maps of higher resolution. For example, we can set the stride of Pool3 as 1 and dilate all Conv4 filters by 2, which reduces the stride of Conv4 from 8 to 4. Unlike previous methods [17,16] that fine-tune the dilated filters, in our method we only use them for feature extraction, without fine-tuning a new RPN.

Though we adopt the same RoI resolution (7×7) as Faster R-CNN [2], these RoIs are on higher-resolution feature maps (*e.g.*, Conv3_3, Conv4_3, or Conv4_3 à trous) than Fast R-CNN (Conv5_3). If an RoI's input resolution is smaller than output (*i.e.*, $< 7 \times 7$), the pooling bins collapse and the features become "flat" and not discriminative. This problem is alleviated in our method, as it is not constrained to use features of Conv5_3 in our downstream classifier.

3.3 Boosted Forest

The RPN has generated the region proposals, confidence scores, and features, all of which are used to train a cascaded Boosted Forest classifier. We adopt the RealBoost algorithm [18], and mainly follow the hyper-parameters in [6]. Formally, we bootstrap the training by 6 times, and the forest in each stage has $\{64, 128, 256, 512, 1024, 1536\}$ trees. Initially, the training set consists of all positive examples (∼50k on the Caltech set) and the same number of randomly sampled negative examples from the proposals. After each stage, additional hard negative examples (whose number is 10% of the positives, ∼5k on Caltech) are mined and added into the training set. Finally, a forest of 2048 trees is trained after all bootstrapping stages. This final forest classifier is used for inference. Our implementation is based on [28].

We note that it is not necessary to handle the initial proposals equally, because our proposals have initial confidence scores computed by RPN. In other words, the RPN can be considered as the stage-0 classifier f_0, and we set $f_0 = \frac{1}{2} \log \frac{s}{1-s}$ following the RealBoost form where s is the score of a proposal region (f_0 is a constant in standard boosting). The other stages are as in standard RealBoost.

3.4 Implementation Details

We adopt single-scale training and testing as in [15,1,2], without using feature pyramids. An image is resized such that its shorter edge has N pixels (N=720 pixels on Caltech, 600 on INRIA, 810 on ETH, and 500 on KITTI). For RPN training, an anchor is considered as a positive example if it has an Intersection-over-Union (IoU) ratio greater than 0.5 with one ground truth box, and otherwise negative. We adopt the image-centric training scheme [1,2], and each mini-batch consists of 1 image and 120 randomly sampled anchors for computing the loss. The ratio of positive and negative samples is 1:5 in a mini-batch. Other hyper-parameters of RPN are as in [2], and we adopt the publicly available code of [2] to fine-tune the RPN. We note that in [2] the cross-boundary anchors are ignored during fine-tuning, whereas in our implementation we preserve the cross-boundary negative anchors during fine-tuning, which empirically improves accuracy on these datasets.

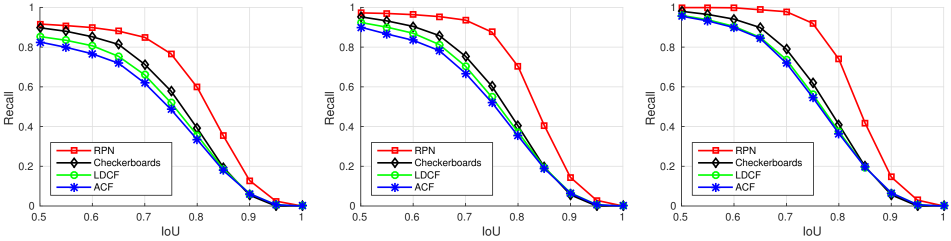
With the fine-tuned RPN, we adopt non-maximum suppression (NMS) with a threshold of 0.7 to filter the proposal regions. Then the proposal regions are ranked by their scores. For BF training, we construct the training set by selecting the top-ranked 1000 proposals (and ground truths) of each image. The tree depth is set as 5 for the Caltech and KITTI set, and 2 for the INRIA and ETH set, which are empirically determined according to the different sizes of the data sets. At test time, we only use the top-ranked 100 proposals in an image, which are classified by the BF.

4 Experiments and Analysis

4.1 Datasets

We comprehensively evaluate on 4 benchmarks: Caltech [14], INRIA [20], ETH [21] and KITTI [22]. By default an IoU threshold of 0.5 is used for determining True Positives in these datasets.

On Caltech [14], the training data is augmented by 10 folds (42782 images) following [3]. 4024 images in the standard test set are used for evaluation on the original annotations under the "reasonable" setting (pedestrians that are at least 50 pixels tall and at least 65% visible) [14]. The evaluation metric is log-average Miss Rate on False Positive Per Image (FPPI) in $[10^{-2}, 10^{0}]$ (denoted as MR_{-2} following [29], or in short MR). In addition, we also test our model on the new annotations provided by [29], which correct the errors in the original annotations. This set is denoted as "Caltech-New". The evaluation metrics in Caltech-New are MR_{-2} and MR_{-4}, corresponding to the log-average Miss Rate on FPPI ranges of $[10^{-2}, 10^{0}]$ and $[10^{-4}, 10^{0}]$, following [29].



(a) 1 proposal (b) 4 proposals (c) 100 proposals

Fig. 3: Comparison of RPN and three existing methods in terms of proposal quality (recall *vs.* IoU) on the Caltech set, with on average 1, 4 or 100 proposals per image are evaluated.

The INRIA [20] and ETH [21] datasets are often used for verifying the generalization capability of the models. Following the settings in [30], our model is trained on the 614 positive and 1218 negative images in the INRIA training set. The models are evaluated on the 288 testing images in INRIA and 1804 images in ETH, evaluated by MR_{-2}.

The KITTI dataset [22] consists of images with stereo data available. We perform training on the 7481 images of the left camera, and evaluate on the standard 7518 test images. KITTI evaluates the PASCAL-style mean Average Precision (mAP) under three difficulty levels: “Easy”, “Moderate”, and “Hard”[1].

4.2 Ablation Experiments

In this subsection, we conduct ablation experiments on the Caltech dataset.

Is RPN good for pedestrian detection?

In Fig. 3 we investigate RPN in terms of *proposal quality*, evaluated by the recall rates under different IoU thresholds. We evaluate on average 1, 4, or 100 proposals per image[2]. Fig. 3 shows that in general RPN performs better than three leading methods that are based on traditional features: SCF [11], LDCF [24] and Checkerboards [31]. With 100 proposals per image, our RPN achieves >95% recall at an IoU of 0.7.

More importantly, RPN as a *stand-alone pedestrian detector* achieves an MR of **14.9%** (Table 1). *This result is competitive and is better than all but two state-of-the-art competitors on the Caltech dataset* (Fig. 4). We note that unlike RoI pooling that may suffer from small regions, RPN is essentially based on fixed-size sliding windows (in a fully convolutional fashion) and thus avoids collapsing bins. RPN predicts small objects by using small anchors.

[1] http://www.cvlibs.net/datasets/kitti/eval_object.php

[2] To be precise, “on average k proposals per image” means that for a dataset with M images, the top-ranked kM proposals are taken to evaluate the recall.

method	RoI features	MR (%)
RPN stand-alone	-	14.9
RPN + R-CNN	raw pixels	13.1
RPN + Fast R-CNN	Conv5_3	20.2
RPN + Fast R-CNN	Conv5_3, à trous	16.2
RPN + BF	Conv5_3	18.2
RPN + BF	Conv4_3	**12.6**
RPN + BF	Conv5_3, à trous	13.7

Table 1: Comparisons of different classifiers and features on the Caltech set. All methods are based on VGG-16 (including R-CNN). The same set of RPN proposals are used for all entries.

RoI features	time/img	MR (%)
Conv2_2	0.37s	15.9
Conv3_3	0.37s	**12.4**
Conv4_3	0.37s	12.6
Conv5_3	0.37s	18.2
Conv3_3, Conv4_3	0.37s	**11.5**
Conv3_3, Conv4_3, Conv5_3	0.37s	11.9
Conv3_3, (Conv4_3, à trous)	0.51s	**9.6**

Table 2: Comparisons of different features in our RPN+BF method on the Caltech set. All entries are based on VGG-16 and the same set of RPN proposals.

How important is feature resolution?

We first report the accuracy of ("slow") R-CNN [12]. For fair comparisons, we fine-tune R-CNN using the VGG-16 network, and the proposals are from the same RPN as above. This method has an MR of 13.1% (Table 1), better than its proposals (stand-alone RPN, 14.9%). R-CNN crops raw pixels from images and warps to a fixed size (224×224), so suffers less from small objects. This result suggests that if reliable features (*e.g.*, from a fine resolution of 224×224) can be extracted, the downstream classifier is able to improve the accuracy.

Surprisingly, training a Fast R-CNN classifier on the same set of RPN proposals actually *degrades* the results: the MR is considerably increased to 20.2% (*vs.* RPN's 14.9%, Table 1). Even though R-CNN performs well on this task, Fast R-CNN presents a much worse result.

This problem is partially because of the low-resolution features. To show this, we train a Fast R-CNN (on the same set of RPN proposals as above) with the à trous trick adopted on Conv5, reducing the stride from 16 pixels to 8. The problem is alleviated (16.2%, Table 1), demonstrating that higher resolution can

method	RoI features	bootstrapped?	MR (%)
RPN + Fast R-CNN	Conv5_3, à trous		16.2
RPN + Fast R-CNN	Conv5_3, à trous	✓	14.3
RPN + BF	Conv5_3, à trous	✓	13.7

Table 3: Comparisons of with/without bootstrapping on the Caltech set.

be helpful. Yet, this result still lags far behind the stand-alone RPN or R-CNN (Table 1).

The effects of low-resolution features are also observed in our Boosted Forest classifiers. BF using Conv5_3 features has an MR of 18.2% (Table 1), lower than the stand-alone RPN. Using the à trous trick on Conv5 when extracting features (Sec. 3.2), BF has a much better MR of 13.7%.

But the BF classifier is more flexible and is able to take advantage of features of various resolutions. Table 2 shows the results of using different features in our method. Conv3_3 or Conv4_3 alone yields good results (12.4% and 12.6%), showing the effects of higher resolution features. Conv2_2 starts to show degradation (15.9%), which can be explained by the weaker representation of the shallower layers. BF on the concatenation of Conv3_3 and Conv4_3 features reduces the MR to 11.5%. The combination of features in this way is nearly cost-free. Moreover, unlike previous usage of skip connections [27], it is not necessary to normalize features in a decision forest classifier.

Finally, combining Conv3_3 with the à trous version of Conv4_3, we achieve the best result of **9.6%** MR. We note that this is at the cost of extra computation (Table 2), because it requires to re-compute the Conv4 features maps with the à trous trick. Nevertheless, the speed of our method is still competitive (Table 4).

How important is bootstrapping?

To verify that the bootstrapping scheme in BF is of central importance (instead of the tree structure of the BF classifiers), we replace the last-stage BF classifier with a Fast R-CNN classifier. The results are in Table 3. Formally, after the 6 stages of bootstrapping, the bootstrapped training set is used to train a Fast R-CNN classifier (instead of the final BF with 2048 trees). We perform this comparison using RoI features on Conv5_3 (à trous). The bootstrapped Fast R-CNN has an MR of 14.3%, which is closer to the BF counterpart of 13.7%, and better than the non-bootstrapped Fast R-CNN's 16.2%. This comparison indicates that the major improvement of BF over Fast R-CNN is because of bootstrapping, whereas the shapes of classifiers (forest *vs.* MLP) are less important.

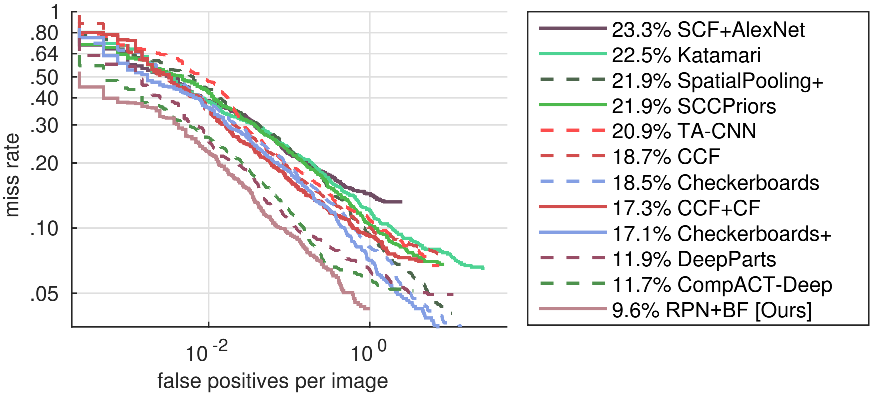


Fig. 4: Comparisons on the **Caltech** set (legends indicate MR).

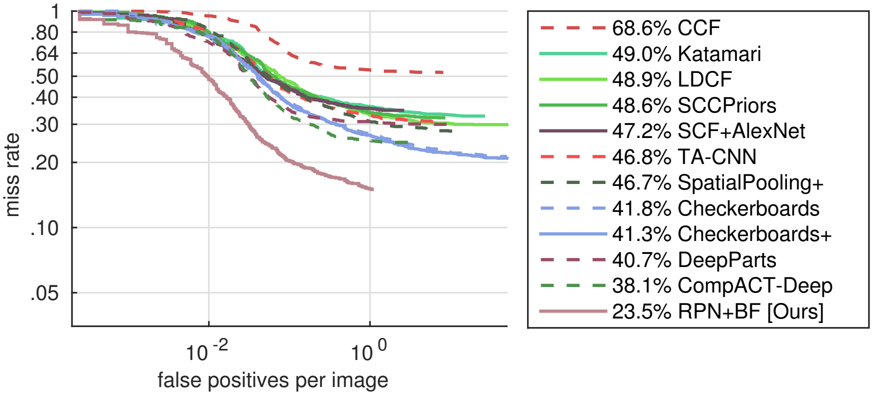


Fig. 5: Comparisons on the **Caltech** set using an IoU threshold of 0.7 to determine True Positives (legends indicate MR).

4.3 Comparisons with State-of-the-art Methods

Caltech Fig. 4 and 6 show the results on Caltech. In the case of using original annotations (Fig. 4), our method has an MR of **9.6%**, which is over 2 points better than the closest competitor (11.7% of CompactACT-Deep [6]). In the case of using the corrected annotations (Fig. 6), our method has an MR_{-2} of 7.3% and MR_{-4} of 16.8%, both being 2 points better than the previous best methods.

In addition, expect for CCF (MR 18.7%) [25], ours (MR 9.6%) is the only method that *uses no hand-crafted features*. Our results suggest that hand-crafted features are not essential for good accuracy on the Caltech dataset; rather, high-resolution features and bootstrapping are the key to good accuracy, both of which are missing in the original Fast R-CNN detector.

Fig. 5 shows the results on Caltech where an IoU threshold of 0.7 is used to determine True Positives (instead of 0.5 by default). With this more challeng-

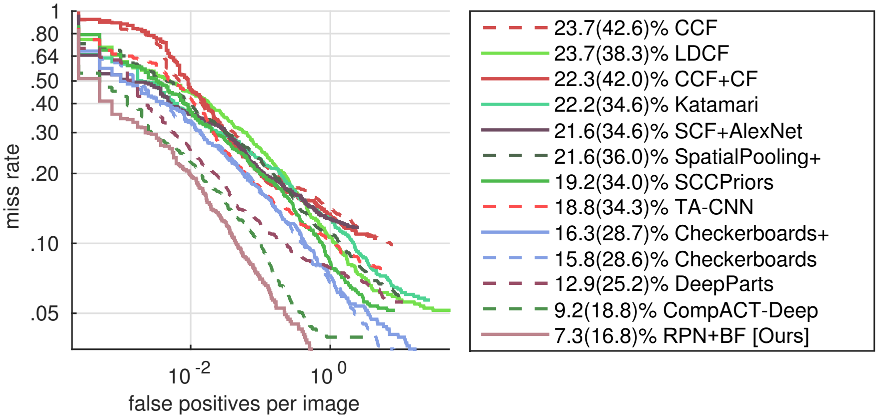


Fig. 6: Comparisons on the **Caltech-New** set (legends indicate MR_{-2} (MR_{-4})).

method	hardware	time/img (s)	MR (%)
LDCF [24]	CPU	0.6	24.8
CCF [25]	Titan Z GPU	13	17.3
CompACT-Deep [6]	Tesla K40 GPU	**0.5**	11.7
RPN+BF [ours]	Tesla K40 GPU	**0.5**	**9.6**

Table 4: Comparisons of running time on the Caltech set. The time of LDCF and CCF is reported in [25], and that of CompactACT-Deep is reported in [6].

ing metric, most methods exhibit dramatic performance drops, *e.g.*, the MR of CompactACT-Deep [6]/DeepParts [5] increase from 11.7%/11.9% to 38.1%/40.7%. Our method has an MR of 23.5%, which is **a relative improvement of ∼40%** over the closest competitors. This comparison demonstrates that our method has a substantially better **localization** accuracy. It also indicates that there is much room to improve localization performance on this widely evaluated dataset.

Table 4 compares the running time on Caltech. Our method is as fast as CompACT-Deep [6], and is much faster than CCF [25] that adopts feature pyramids. Our method shares feature between RPN and BF, and achieves a good balance between speed and accuracy.

INRIA and ETH Fig. 7 and 8 show the results on the INRIA and ETH datasets. On the INRIA set, our method achieves an MR of 6.9%, considerably better than the best available competitor's 11.2%. On the ETH set, our result (30.2%) is better than the previous leading method (TA-CNN [4]) by 5 points.

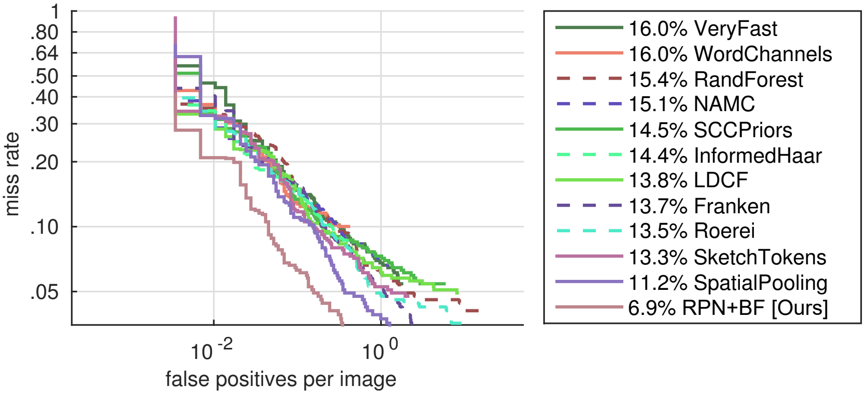


Fig. 7: Comparisons on the **INRIA** dataset (legends indicate MR).

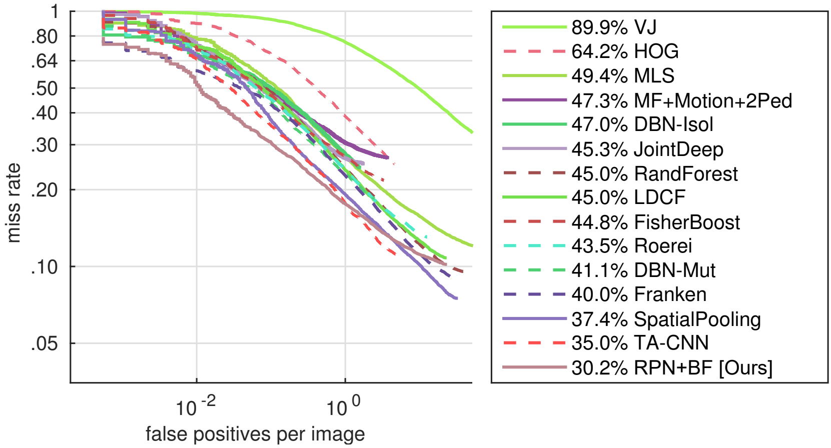


Fig. 8: Comparisons on the **ETH** dataset (legends indicate MR).

KITTI Table 5 shows the performance comparisons on KITTI. Our method has competitive accuracy and fast speed.

5 Conclusion and Discussion

In this paper, we present a very simple but effective baseline that uses RPN and BF for pedestrian detection. On top of the RPN proposals and features, the BF classifier is flexible for (i) combining features of arbitrary resolutions from any layers, without being limited by the classifier structure of the pre-trained network; and (ii) incorporating effective bootstrapping for mining hard negatives. These nice properties overcome two limitations of the Faster R-CNN

method	mAP on Easy	mAP on Moderate	mAP on Hard	Times (s)
R-CNN	61.61	50.13	44.79	4
pAUCEnstT	65.26	54.49	48.60	60
FilteredICF	67.65	56.75	51.12	2
DeepPart	70.49	58.67	52.78	1
CompACT-Deep	70.69	58.74	52.71	1
Regionlets	73.14	**61.15**	**55.21**	$1^{\dagger}$
RPN+BF [ours]	**77.12**	**61.15**	**55.12**	0.6

Table 5: Comparisons on the **KITTI** dataset collected at the time of submission (Feb 2016). The timing records are collected from the KITTI leaderboard. †: region proposal running time ignored (estimated 2s).

system for pedestrian detection. Our method is a self-contained solution and does not resort to hybrid features.

Interestingly, we show that *bootstrapping* is a key component, even with the advance of deep neural networks. Using the same bootstrapping strategy and the same RoI features, both the tree-structured BF classifier and the region-wise MLP classifier (Fast R-CNN) are able to achieve similar results (Table 3). Concurrent with this work, an independently developed method called Online Hard Example Mining (OHEM) [32] is developed for training Fast R-CNN for general object detection. It is interesting to investigate this end-to-end, online mining fashion *vs.* the multi-stage, cascaded bootstrapping one.

Acknowledgement This work was supported in part by State Key Development Program under Grant 2016YFB1001000, in part by Guangdong Natural Science Foundation under Grant S2013050014548. This work was also supported by Special Program for Applied Research on Super Computation of the NSFC-Guangdong Joint Fund (the second phase). We thank the anonymous reviewers for their constructive comments on improving this paper.

References

1. Ross Girshick. Fast R-CNN. In *IEEE International Conference on Computer Vision (ICCV)*, 2015.
2. Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster R-CNN: Towards real-time object detection with region proposal networks. In *Neural Information Processing Systems (NIPS)*, 2015.
3. Jan Hosang, Mohamed Omran, Rodrigo Benenson, and Bernt Schiele. Taking a deeper look at pedestrians. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015.

4. Yonglong Tian, Ping Luo, Xiaogang Wang, and Xiaoou Tang. Pedestrian detection aided by deep learning semantic tasks. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015.
5. Yonglong Tian, Ping Luo, Xiaogang Wang, and Xiaoou Tang. Deep learning strong parts for pedestrian detection. In *IEEE International Conference on Computer Vision (ICCV)*, 2015.
6. Zhaowei Cai, Mohammad Saberian, and Nuno Vasconcelos. Learning complexity-aware cascades for deep pedestrian detection. In *IEEE International Conference on Computer Vision (ICCV)*, 2015.
7. Piotr Dollár, Zhuowen Tu, Pietro Perona, and Serge Belongie. Integral channel features. In *British Machine Vision Conference (BMVC)*, 2009.
8. Piotr Dollár, Ron Appel, Serge Belongie, and Pietro Perona. Fast feature pyramids for object detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 2014.
9. Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Neural Information Processing Systems (NIPS)*, 2012.
10. Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv:1409.1556*, 2014.
11. Rodrigo Benenson, Mohamed Omran, Jan Hosang, and Bernt Schiele. Ten years of pedestrian detection, what have we learned? In *ECCV workshop*, 2014.
12. Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2014.
13. Jasper RR Uijlings, Koen EA van de Sande, Theo Gevers, and Arnold WM Smeulders. Selective search for object recognition. *International Journal of Computer Vision (IJCV)*, 2013.
14. Piotr Dollár, Christian Wojek, Bernt Schiele, and Pietro Perona. Pedestrian detection: An evaluation of the state of the art. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 2012.
15. Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Spatial pyramid pooling in deep convolutional networks for visual recognition. In *European Conference on Computer Vision (ECCV)*, 2014.
16. Liang-Chieh Chen, George Papandreou, Iasonas Kokkinos, Kevin Murphy, and Alan L Yuille. Semantic image segmentation with deep convolutional nets and fully connected crfs. *arXiv:1412.7062*, 2014.
17. Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015.
18. Jerome Friedman, Trevor Hastie, Robert Tibshirani, et al. Additive logistic regression: a statistical view of boosting (with discussion and a rejoinder by the authors). *The annals of statistics*, 2000.
19. Ron Appel, Thomas Fuchs, Piotr Dollár, and Pietro Perona. Quickly boosting decision trees-pruning underachieving features early. In *International Conference on Machine Learning (ICML)*, 2013.
20. Navneet Dalal and Bill Triggs. Histograms of oriented gradients for human detection. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2005.
21. Andreas Ess, Bastian Leibe, and Luc Van Gool. Depth and appearance for mobile scene analysis. In *IEEE International Conference on Computer Vision (ICCV)*, 2007.

22. Andreas Geiger, Philip Lenz, and Raquel Urtasun. Are we ready for autonomous driving? the kitti vision benchmark suite. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2012.
23. Paul Viola and Michael J Jones. Robust real-time face detection. *International Journal of Computer Vision (IJCV)*, 2004.
24. Woonhyun Nam, Piotr Dollár, and Joon Hee Han. Local decorrelation for improved pedestrian detection. In *Neural Information Processing Systems (NIPS)*, 2014.
25. Bin Yang, Junjie Yan, Zhen Lei, and Stan Z. Li. Convolutional channel features. In *IEEE International Conference on Computer Vision (ICCV)*, 2015.
26. Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 2015.
27. Wei Liu, Andrew Rabinovich, and Alexander C Berg. Parsenet: Looking wider to see better. *arXiv:1506.04579*, 2015.
28. Piotr Dollár. Piotr's Computer Vision Matlab Toolbox (PMT). `https://github.com/pdollar/toolbox`.
29. Shanshan Zhang, Rodrigo Benenson, Mohamed Omran, Jan Hosang, and Bernt Schiele. How far are we from solving pedestrian detection? In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.
30. Sakrapee Paisitkriangkrai, Chunhua Shen, and Anton van den Hengel. Strengthening the effectiveness of pedestrian detection with spatially pooled features. In *European Conference on Computer Vision (ECCV)*. 2014.
31. Shanshan Zhang, Rodrigo Benenson, and Bernt Schiele. Filtered channel features for pedestrian detection. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015.
32. Abhinav Shrivastava, Abhinav Gupta, and Ross Girshick. Training region-based object detectors with online hard example mining. *arXiv:1604.03540*, 2016.