Exploit All the Layers: Fast and Accurate CNN Object Detector with Scale Dependent Pooling and Cascaded Rejection Classifiers

Fan Yang*[1,2], Wongun Choi[2], and Yuanqing Lin[2]

[1]Department of Computer Science, University of Maryland College Park
[2]NEC Laboratories America
fyang@cs.umd.edu, {wongun, ylin}@nec-labs.com

Abstract

In this paper, we investigate two new strategies to detect objects accurately and efficiently using deep convolutional neural network: 1) scale-dependent pooling and 2) layer-wise cascaded rejection classifiers. The scale-dependent pooling (SDP) improves detection accuracy by exploiting appropriate convolutional features depending on the scale of candidate object proposals. The cascaded rejection classifiers (CRC) effectively utilize convolutional features and eliminate negative object proposals in a cascaded manner, which greatly speeds up the detection while maintaining high accuracy. In combination of the two, our method achieves significantly better accuracy compared to other state-of-the-arts in three challenging datasets, PASCAL object detection challenge, KITTI object detection benchmark and newly collected Inner-city dataset, while being more efficient.

1. Introduction

In recent years, deep convolutional neural network (CNN) [18, 19] contributed much to various computer vision problems including image classification, object detection, semantic segmentation, etc, thanks to its capability to learn discriminative features (or representations) at different levels of granularities. A number of recent studies [41, 43] suggest that high level visual semantics (such as motif, parts, or objects) are appearing in the middle of deep architecture which in turn provide strong cues to recognize complex visual concepts. Leveraging on the representational power of CNN, a number of methods are proposed to detect objects in natural images using CNN [13, 14, 16, 7, 42].

R-CNN [14] has been proposed for object detection and achieves promising results, where a pre-trained network is fine-tuned to classify object proposals. However, both training and testing suffer from low efficiency since the network performs a forward pass on every single object proposal independently. Convolutional filters are repeatedly applied to thousands of object proposals that are redundant and expensive. In order to reduce the computational cost, recent CNN based object detectors, such as Fast RCNN [13] and spatial pyramid pooling networks (SPPnet) [16], share the features generated by the convolutional layers and apply a multi-class classifier for each proposal. In Fast RCNN, convolutional operations are done only once on the whole image, while the features for object proposals are pooled from the last convolutional layer and fed into fully-connected (*fc*) layers to evaluate the likelihood of object categories. Compared to R-CNN [14], these methods improve the efficiency in the order of magnitude via shared convolutional layers. For instance, Fast RCNN achieves $3\times$ and $10 \sim 100\times$

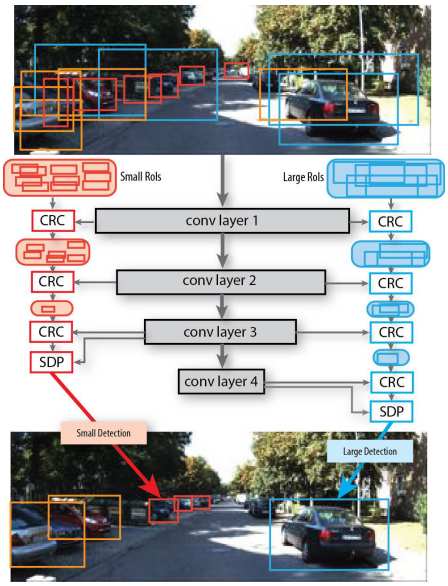

Figure 1. We present a fast and accurate object detection method using the convolutional neural network. Our method exploits the convolutional features in all layers to reject easy negatives via *cascaded rejection classifiers* and evaluate surviving proposals using our *scale-dependent pooling* method.

*This work was done at NEC Laboratories America.

speedup at training and test stage, respectively. In order to deal with scale variation, multi-scale image inputs are often used where one set of convolutional features are obtained per image scale. Despite its success, these approaches have certain drawbacks that make them less flexible. First, Fast RCNN does not handle small objects well. Since the bounding boxes are pooled directly from the last convolutional layer rather than being warped into a canonical size, they may not contain enough information for decision if the boxes are too small. Multi-scale input scheme fundamentally limits the applicability of very deep architecture like [32] due to memory constraints and additional computational burden. In addition, pooling a huge number of candidate bounding boxes and feeding them into high-dimensional *fc* layers can be extremely time-consuming.

In this work, we attempt to address aforementioned drawbacks and propose a new CNN architecture for an accurate and efficient object detection in images. The first contribution is that, unlike previous works, our method produces only one set of convolutional features for an image while handling the scale variation via multiple scale-dependent classifier models. Our intuition is that visual semantic concepts of an object can emerge in different convolutional layers depending on the size of the target objects, if proper supervision is provided in the training process. For instance, if a target object is small, we may observe a strong activation of convolutional neurons in earlier layers (e.g. *conv3*) that encode specific parts of an object. On the other hand, if a target object is large, the same part concept will emerge in much later layers (e.g. *conv5*). Based on this intuition, we represent a candidate bounding box using the convolutional features pooled from a layer corresponding to its scale (scale-dependent pooling (SDP)). The features are fed into multiple scale-dependent object classifiers to evaluate the likelihood of object categories. As for the second contribution, we present a novel cascaded rejection classifier (CRC) where the cascading direction is defined over the convolutional layers in the CNN. We treat the convolutional features in early layers as a weak classifier in the spirit of boosting classifiers [10]. Although the features from earlier convolutional layers might be too weak to make a strong evaluation of an object category, they are still useful to quickly reject easy negatives. Combining the two strategies, we can explicitly utilize the convolutional features at all layers instead of using only the last one as previous works do. Our method is illustrated in Figure 1.

We evaluate our model using three object detection datasets, PASCAL object detection challenge [8], KITTI object detection benchmark [11] and newly collected Inner-city dataset. In all experiments, we observe that our method can achieve significantly higher detection accuracy compared to the other methods with much higher computational efficiency.

2. Related Works

CNN for object detection. Before the emergence of CNN, deformable part model (DPM) [9] has been the state-of-the-art object detector for years. With the exceptional power on image classification, CNN has been applied to object detection and achieves promising results [34, 14, 7, 44, 13, 27]. In [34], detection is treated as a regression problem to object bounding box masks. A deep neural network is learned to generate and localize object boxes. Erhan *et al*. [7] design a deep network to propose class-agnostic bounding boxes for generic object detection. Sermanet *et al*. [31] use a regression network pre-trained for classification tasks to predict object bounding boxes in an exhaustive and computationally expensive way. Each bounding box is associated with a confidence score indicating the presence of an object class. Recently, Girshick *et al*. [14] propose the R-CNN framework that uses object proposals generated by selective search to fine-tune a pre-trained network for detection tasks. Zhang *et al*. [42] extend R-CNN by gradually generating bounding boxes within a search region and imposing a structured loss to penalize localization inaccuracy in network fine-tuning. To reduce the cost of forward pass for each proposal in R-CNN, Fast RCNN [13] has been proposed by sharing convolutional features and pooling object proposals only from the last convolutional layer. More recently, Faster RCNN [29] replaces the object proposals generated by selective search by a region proposal network (RPN) and achieves further speed-up.

Neural network cascades. The Viola-Jones cascaded face detector [36] and its extensions [6, 24] have been widely used. The idea of eliminating candidates by combining a series of simple features is also applied to CNNs. Sun *et al*. [33] present an ensemble of networks by combining networks focusing on different facial parts for facial point detection. Li *et al*. [21] use a shallow detection network with small scale input images to first reject easy non-face samples, and then apply two deeper networks to eliminate more negatives while maintaining a high recall. A calibration network is appended after each detection network for bounding box calibration. More recently, Angelova *et al*. [1] combine a tiny deep network and a modified AlexNet to achieve real-time pedestrian detection. The tiny deep network removes a large number of candidates and leaves a manageable size of candidates for the large network to evaluate. Our approach is significantly different from prior methods in that we consider cascaded classifiers by utilizing features from different convolutional layers within a single network, that does not introduce any additional computation.

Using convolutional features. A few works exploit features from different convolutional layers, either by concatenating them or by other popular encoding techniques. One of the most representative works is [15], where neuron activations at a pixel of different feature maps are concatenated

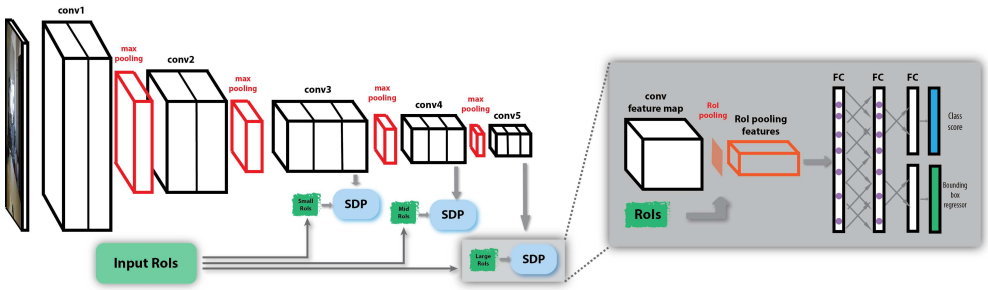

Figure 2. Details of our Scale-dependent Pooling (SDP) model on 16-layer VGG net. For better illustration, we show the groups of convolutional filters between max pooling layers as a cube, where filters are arranged side-by-side, separated by lines.

as a vector as a pixel descriptor for localization and segmentation. Similarly, in the fully convolutional network [23], feature maps from intermediate level and high level convolutional layers are combined to provide both finer details and higher-level semantics for better image segmentation. Xu *et al*. [40] extract convolutional features in the same way and encode these feature vectors by VLAD and Fisher vector for efficient video event detection. DeepProposal [12] is used to generate object proposals in a coarse-to-fine manner. Proposals are first generated in higher level convolutional layers that preserve more semantic information, and are gradually refined in lower layers that provide better localization. Similarly, Karianakis *et al*. [17] use lower-level convolutional features to generate object proposals by sliding window and remove background proposals, while refining them using higher-level convolutional features in a hierarchical way. For edge detection, Bertasius *et al*. [3] extract a sub-volume from every convolutional layers, perform three types of pooling and again concatenate these values into a single vector, which is further fed into *fc* layers. Recently, Xie and Tu [39] propose a holistically-nested edge detection scheme inspired by [20]. In the network, side-outputs are added after several early convolutional layers to provide deep supervision for predicting edges at multiple scales. A weighted-fusion layer combines all the side-outputs where the combination weights are learned during network training. In contrast to these works, our approach does not explicitly combine convolutional features, but learns classifiers separately.

3. Scale-Dependent Pooling

Scale variation is a fundamental challenge in visual recognition. Previous works [9, 5] often adopted a sliding window technique with image pyramids to handle the scale variation of target objects. Similar techniques are applied to recent CNN based object recognition methods: they treat the last convolutional layer's outputs (*conv5* of AlexNet) as the features to describe an object and apply a classifier (*fc* layers) on top of the extracted features. R-CNN [14] warps an image patch within a proposal that produces fixed dimensional feature output for classification. The independent warping process prohibits us to share any convolutional operations across proposals in the same image, which fundamentally limits the efficiency. In contrast, SPPnet [16] and Fast RCNN [13] share the convolutional features in an image and pool the features at the last convolutional layer to describe an object. In these methods, the scale variation is tackled either via image pyramid inputs or brute-force learning that directly learns the scale variation via the convolutional filters. However, the image pyramid introduces additional computational burden and requires large amount of GPU memories, and brute-force learning via convolutional filters is difficult.

As for a new contribution, we introduce a *scale-dependent pooling* (SDP) technique (illustrated in the Figure 2) to effectively handle the scale variation in object detection problem. Our method is built based on the recent Fast RCNN [13] method that pools the features for each bounding box proposal from the last convolutional layer of CNN. The region inside of each proposal is divided into a spatial grid (7×7 or 6×6) and features are pooled using max-pooling over each grid. Our SDP method examines the scale (height) of each object proposal and pools the features from a corresponding convolutional layer depending on the height. For instance, if an object proposal has a height between 0 to 64 pixels, the features are pooled from the 3^{rd} convolutional layer of CNN (SDP_3). On the other hand, if an object proposal has a height larger than 128 pixels, we pool the features from the last convolutional layer (SDP_5) (see Figure 2). The *fc* layers attached to SDPs have their own set of parameters so as to learn scale-specific classification models from different sets of feature inputs.

The main benefit of SDP is that we can effectively tackle the scale variation of target objects while computing the convolutional features only once per image. Instead of artificially resizing the input images in order to obtain a proper feature description, the SDP selects a proper feature layer to describe an object proposal. It reduces computational cost and memory overhead caused by redundant convolutional operations. Another benefit is that the SDP results in a compact and consistent representation of object proposals. Since the brute-force approach of Fast RCNN [13] pools the features for object proposals from the last convolutional layer, often the same features are repeated over

the spatial grid if an object proposal is very small. The max-pooling or multiple pixel stride in convolutional layers progressively reduces the spatial resolution of the convolutional features over layers. Thus, at the *conv5* layer, there is only one feature for large number of pixels (16 pixels for both AlexNet [18] and VGG16 [32]). In the extreme case, if the object proposal is as small as 16×16 pixels, all the grid features may be filled with a repeating single feature. Learning from such an irregular description of object examples may prohibit us from learning a strong classification model. Since the SDPs distribute the proposals depending on the scale, we can provide more consistent signal through the learning process, leading to a better detection model.

The idea of using intermediate convolutional layers to complement high level convolutional features has also been exploited for image classification and segmentation [22, 15, 23], video event detection [40], image retrieval [2, 25] and edge detection [39]. We note that our approach differs from previous works that directly concatenate or combine outputs of multiple convolutional layers. We explicitly associate classification models (*fc* layers) with different convolutional layers, which provides strong supervision to enforce more discriminative convolutional filters. Unlike [39], we do not fuse multiple side-outputs into a single output, so that each side-output is not affected by other outputs, and can be adjusted independently to handle a specific scale of objects, making the training more flexible and robust.

We present our SDP model based on VGG16 [32] in Figure 2. This SDP model has 3 branches after *conv3*, *conv4* and *conv5*. Each branch consists of a *RoI pooling* layer connected to 2 successive *fc* layers with *ReLU* activations and *dropout* layers for calculating class scores and bounding box regressors, similarly to [13]. We initialize the model parameters of convolutional layers and the *fc* layers in the SDP_5 with the Image-Net pre-trained model of VGG16 [32]. The *fc* layers in the SDP_3 and SDP_4 are randomly initialized. During fine-tuning, input object proposals are first distributed into 3 groups based on their height and then fed into corresponding *RoI pooling* layers to pool the features from corresponding convolutional outputs. Gradients are back-propagated from 3 branches to update corresponding *fc* layers and convolutional filters. By providing supervision about the scale of input object proposals, we explicitly enforce neurons to learn for different scales of objects, so that the convolutional layers are able to discover small objects at early stage. The experiments demonstrate that this simple modification effectively improves detection accuracy on small objects by a large margin (see Sec. 5).

4. Cascaded Rejection Classifiers

One major computational bottleneck in our SDP method and Fast RCNN [13] is on the evaluation of individual object proposals using high dimensional *fc* layers. When there

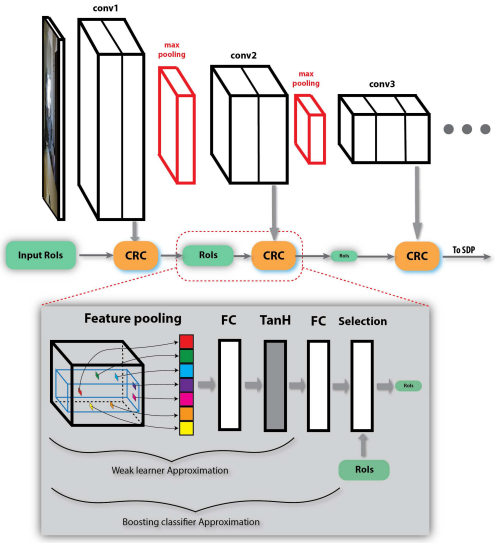

Figure 3. Structure of the rejection classifier approximation by network layers. Blue cuboid corresponds to a proposal on the feature maps. Color squares are feature points that need to be pooled out to form the feature vector.

are tens of thousands of proposals, time spent for the per-proposal evaluation dominates in the entire detection process (see Table 4). As for the second contribution, we introduce a novel cascaded rejection classifier (CRC) scheme that requires minimal amount of additional computation. Cascaded detection framework has been widely adopted in visual detection problems that includes [36, 6, 24]. The core idea is to use as little as possible computations to reduce the object proposals quickly and use complex and time-consuming features for only few highly likely candidate locations. Recently, a few methods [33, 21, 1] are proposed to use cascaded detection framework with CNN, but most of them employ another simpler network to "preprocess" object proposals and use a deeper architecture to evaluate surviving candidates. Unlike the others, we exploit the convolutional features progressively in a single network to build the cascaded rejection classifiers. Since cascaded classifiers gradually reject negatives using stronger features, they resemble the behaviour of deep networks where more semantic and constrained features gradually emerge to help discriminate objects. Therefore, our choice of boosting classifiers is reasonable and consistent with the network characteristics. Our model does not require any additional convolutional feature computation.

We adopt the popular discrete AdaBoost [10] algorithm to learn CRCs after each convolutional layer. Following the intuition of our SDP models, we learn separate rejection classifiers per scale-group ($\mathcal{R}_s^l$ where s and l represent a scale-group and the convolutional layer) in order to keep the classifiers compact while effective (see Figure 3). In following paragraphs, we assume that we have a CNN model trained with SDPs without loss of generality.

Learning CRCs. Let us first define necessary notations to

learn a CRC $\mathcal{R}_s^l$. Suppose we have N proposals that belong to a scale group s, $\mathcal{B} = [B_1, B_2, ..., B_N]$ and corresponding foreground label $y_i, i = 1, ..., N$. $y_i = 1$ if it contains a foreground object and $y_i = 0$, otherwise. We pool the corresponding features x_i for $B_i \in \mathcal{B}$ from the convolutional layer l using the CNN model trained with our SDPs. In our experiments, we use the $RoIPooling$ scheme of [13], which gives $m \times m \times c$ dimensional features, where $m = 7$ and c is the number of channels in the convolutional layer. Through this process, we obtain a training dataset of $\mathbf{X} = [\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_N] \in \mathbb{R}^{m^2 c \times N}$, and $\mathbf{Y} = \{0, 1\} \in \mathbb{R}^N$.

Given the training dataset, we learn a linear boosting classifier $\mathcal{R}_s^l$ with [10] that aggregates a set of weak-learners' response outputs, $\mathcal{R}_s^l(\mathbf{x}) = \sum_{t=1}^{T} w_t h_t(\mathbf{x})$, where h_t is a weak learner, w_t is the corresponding weight and the output is the classification score. In this work, a weak learner h_t is a decision stump that outputs 1 if the value x_v at the v^{th} feature dimension is greater than a decision threshold δ_v and -1 otherwise, that can be written as $h_t(\mathbf{x}) = sign(x_v - \delta_v)$. We learn 50 weak-learners per $\mathcal{R}_s^l$. After learning the boosting classifier, we train the rejection threshold that keeps 99% of positive training examples. All surviving training examples are passed to train the next rejection classifier $\mathcal{R}_s^{l+1}$. In order to learn progressively stronger rejection classifiers without additional computational cost, the weak-learners used in the previous $\mathcal{R}_s^l$ are used to initialize the boosting classifier training in the next $\mathcal{R}_s^{l+1}$.

CRCs in Testing. Since we know which features must be pooled after training the CRCs, we pool only the necessary features in the testing time (the *feature pooling* layer in Figure 3). Given the 50 dimensional pool of weak-learners, we approximate the boosting classifier with 2 *fc* layers and a hyperbolic tangent $tanh$ layer, so as to utilize the computational modules in the CNN framework. The first *fc* layer applies the translation of the features with δ_v, which is followed by the $tanh$ layer that approximates the $sign$ function. Finally, all the weak-learners are aggregated via the last *fc* layer to produce the final boosting classification score using w. If available ($l > 1$), the previous rejection classifier $\mathcal{R}_s^{l-1}$ score is added before rejecting an object proposal. The detailed structure of the CRC is illustrated in Figure 3. We observe that the cascaded rejection classifiers achieves about 3.2× speedup for the proposal evaluation (4.6× when combined with truncated SVD [13], see Table 4) with a marginal loss of accuracy. Instead of simply connecting boosting classifiers to the network, we show that a boosting classifier can be decomposed into a composition of network layers, which provides new insights to converting traditional classifiers into deep networks.

5. Experiments

Datasets. We evaluate our model with SDP and CRC on 3 datasets: KITTI detection benchmark [11] , PASCAL VOC2007 [8] and newly collected Inner-city dataset. The KITTI dataset is composed of 7,481 images for training, and 7,518 images for testing. The training dataset contains 28,742, 4,487, and 1,627 number of car, pedestrian and cyclist annotations. Since the ground-truth annotation of testing set is not publicly available, we use the training/validation split of [38] for the analysis. For more thorough analysis, we collected a new dataset (Inner-city). The dataset is composed of 24,509 images which are collected using a camera mounted on a car. The dataset is composed of 16,028 training and 8,481 testing images which contains 60,658, 36,547, 16,842, and 14,414 numbers of car, person, bike and truck instances, respectively. The images are sub-sampled 15 frames apart from 47 number of video sequences to avoid having highly correlated images.

Networks. Our CNN model is initialized with a deep network architecture (VGG16 [32]) trained on the ImageNet classification dataset [30]. Rather than having SDP branches for all convolutional layers, we add 3 SDP branches after 3 convolutional layers before max pooling, which are *conv3_3* (SDP_3), *conv4_3* (SDP_4) and *conv5_3* (SDP_5) of VGG16, to ensure the features are discriminative enough. We use scale groups of height between $[0, 64)$ for SDP_3, $[64, 128)$ for SDP_4, and $[128, \infty)$ for SDP_5. The *fc* layers in the SDP_5 are initialized with the pre-trained model parameters, while the *fc* layers in the SDP_3 and SDP_4 are randomly initialized. All the *fc* layers have 4096 dimensional outputs. After fine-tuning, we train rejection classifiers for each scale group using the convolutional features from *conv1_2*, *conv2_2*, *conv3_3*, *conv4_3* and *conv5_3*, resulting in 12 rejection classifiers. Due to limited amount of training samples of VOC2007, we only use two SDP branches after *conv3_3* ($[0, 64)$ for SDP_3) and *conv5_3* ($[64, \infty)$ for SDP_5). We use 2048-dim *fc* layers for SDP_3 which give better detection accuracy than that by 1024-dim and 4096-dim *fc* layers.

Training Parameters. Following the procedure introduced in [13], we randomly sample two images, from which we randomly sample 128 positive and negative object proposals per scale group as a minibatch. The negative object proposals are sampled from all the proposals that have less than 0.5 overlap with any positive ground-truth annotation. For all the experiments, we use initial learning rate of 0.0005 and decrease it by 0.1 after every 30K iterations. We use the momentum 0.9 and the weight decay 0.0005. The final model is obtained after 90K iterations. We found that using smaller dropout ratio helps to improve the detection accuracy on KITTI and Inner-city in our experiments, so we use a dropout ratio 0.25 after *fc* layers, while 0.5 for VOC2007.

Box Proposals. We obtain the bounding box proposals using Edgebox [45] and augment them with ACF [5] detection outputs trained for Car and Person categories. We observe

that using only generic box proposal methods often misses small target objects, which leads to poor detection accuracy. We use the same proposals provided by [13] for VOC2007.

5.1. Detection Accuracy Analysis

We first discuss the detection accuracy on the KITTI train/validation dataset, Inner-city dataset, and VOC2007 dataset. We mainly compare our model against two baseline methods using Fast RCNN models [13] with AlexNet [18] and VGG16 [32] architectures. For the KITTI train/validation experiment, all the training and testing images are rescaled to 500 pixel height which gives the best accuracy given GPU (K40/K80) memory constraints. We use multi-scale image inputs of $400, 800, 1200, 1600$ pixel heights for the AlexNet baseline and $400, 800$ pixel heights for the VGG16 baseline to handle the scale variation as well as possible. We also compare the VGG16 baseline and our SDP using other single scale inputs (400 and 800). In Inner-city experiments, we keep the original size of the image (420 pixel height) and use $420, 840, 1260, 1680$ for the AlexNet baseline. We use both single scale input (600) and multi-scale inputs ($300, 400, 500, 600$ for training and $500, 600$ for testing) for SDP models in VOC2007 experiments. In order to highlight the challenges posed by the scale variation, we present the accuracy comparison over different size groups in Table 1. Following KITTI [11], we use 0.7 overlap ratio for the Car category and 0.5 for the others in the evaluation. In the VOC2007 and Inner-city evaluation, we use 0.5 overlap ratio across all categories.

Results by SDP. Table 1 shows that the multi-scale image input baselines achieves similar detection accuracy across different scale groups, since features are pooled at appropriate scales. On the other hand, deeper architecture with a single image input baseline achieves higher accuracy on larger objects exploiting the rich semantic features in the deep architecture, but performs relatively poorly on small objects. Even using only 400 pixel heights, SDP already achieves better overall accuracy than VGG16 using 2 scales. Our SDP model with the same VGG16 architecture achieves highest accuracy on almost all scale groups over all the categories. Given that the multi-scale setting takes more time and occupies more GPU memory, our SDP is more efficient and practical in detecting various scales of objects. More importantly, we greatly improve the detection accuracy on the smallest scale group by $5 \sim 20\%$ thanks to the SDP branches for the intermediate convolultional layers, which confirms our hypothesis that small objects can be better recognized at lower layers if proper supervision is provided. Another important observation is that we achieve larger improvement on the Car category which has the largest number of training examples. Since our model has additional parameters to be trained (*fc* layers in SDP_3 and SDP_4), we expect that our model will improve even more when more training examples are provided. This is demonstrated in the Inner-city experiments presented in Table 1 that contains larger number of training examples. A few qualitative results are presented in Figure 4. On VOC2007, we improve the AP on several categories and obtain the overall mAP 68.2% and 69.4% using single scale and multi-scale setting, respectively, which are 1.3% higher than [13] using the same configuration. Especially, we observe significant improvements on small objects like bottle and plants. It should be noted that the number of training samples, especially those containing small objects, are much less than that on KITTI dataset. Since we train the *fc* layers connected to the intermediate layers from scratch, insufficient samples may negatively affect the performance.

Table 2. Detection AP (%) of the other state-of-the-art approaches and our method on KITTI test set. Following KITTI protocol, results are grouped into three levels of difficulties: Easy (E), Moderate (M) and Hard (H).

	Car			*Pedestrian*			*Cyclist*		
Method	E	M	H	E	M	H	E	M	H
Regionlet [37]	84.75	76.45	59.70	73.14	61.15	55.21	70.41	58.72	51.83
DPM-VOC+VP [28]	74.95	64.71	48.76	59.48	44.86	40.37	42.43	31.08	28.23
3DVP [38]	87.46	75.77	65.38	-	-	-	-	-	-
SubCat [26]	84.14	75.46	59.71	-	-	-	-	-	-
CompACT-Deep [4]	-	-	-	70.69	58.74	52.71	-	-	-
DeepParts [35]	-	-	-	70.49	58.67	52.78	-	-	-
FRCN [13]+VGG16	85.98	72.32	60.16	75.50	62.53	58.14	68.82	54.21	47.98
SDP	88.34	81.69	69.72	76.89	**64.44**	**59.72**	70.13	60.08	52.93
SDP+CRC	88.33	81.17	70.00	76.28	63.12	58.30	71.06	60.24	53.17
SDP+CRC *ft*	**90.33**	**83.53**	**71.13**	**77.74**	64.19	59.27	**74.08**	**61.31**	**53.97**

Results by CRC. Next, we evaluate the performance of our CRCs. As described in Sec. 4, we reject object proposals through our CRCs throughout the convolutional layers. With the CRC modules (denoted as SDP+CRC in Table 1), the performance decreases very marginally, indicating that CRCs successfully eliminate negative proposals while maintaining a high recall rate for positives (see Table 3 for details), even though we only use 50 feature dimensions at each convolutional layer. The results demonstrate that the intermediate convolutional layers can be exploited in a hierarchical way.

Fine-tuning with CRC. We train the network with the CRC modules in the training process. The CRC modules can serve as a hard-negative mining process to learn better classification model in the network, since many easy negatives are rejected before reaching the SDP modules. Instead of randomly sampling 128 proposals in the training process, we sample 128 proposals among the survived proposals after using all the CRCs. We run the fine-tuning for additional 50K iterations with initial learning rate 0.0001 with step size 20K iterations. We freeze the learning rate of convolutional layers to avoid CRC parameters being invalid after the fine-tuning. We observe that the additional fine-tuning (SDP+CRC *ft*) helps to improve the accuracy over the SDP+CRC marginally. In KITTI testing result (Table 2), we observe larger improvement with the additional

Table 1. Detection AP (%) of baselines and our models on KITTI validation set and Inner-city dataset, divided by size groups, and VOC2007 dataset. S_1, S_2, S_3, S_4 and S indicate the size group of $[0, 64)$, $[64, 128)$, $[128, 256)$, $[256, \infty)$ and $[0, \infty)$. We use 4 scale image pyramid for FRCN [13]+AlexNet, and 1 and 2 scale image input for the others. 07 \ diff: training without difficult examples. *ms*: multi-scale training and testing. Best numbers are bold-faced.

KITTI Train/Validation

		Car					*Pedestrian*					*Cyclist*					mAP
Methods	Inputs	S_1	S_2	S_3	S_4	S	S_1	S_2	S_3	S_4	S	S_1	S_2	S_3	S_4	S	S
FRCN [13]+AlexNet	4	52.8	60.7	75.8	55.5	61.6	**19.7**	47.5	88.4	24.1	61.4	**42.0**	51.6	44.9	0.0	46.5	56.5
FRCN [13]+VGG16	1 (400)	33.9	68.3	82.8	68.8	57.3	7.9	50.4	**95.3**	55.8	64.6	19.0	63.8	66.6	0.0	42.3	54.7
	1 (500)	42.2	70.0	85.1	65.9	62.3	12.6	55.9	94.6	44.9	66.8	29.1	63.8	68.7	0.0	48.8	59.3
	1 (800)	47.6	70.0	84.8	60.5	64.5	14.7	54.5	94.5	47.2	66.4	34.9	61.2	67.4	0.0	50.4	60.4
	2	47.4	70.2	83.1	54.5	64.1	14.9	55.2	94.5	63.1	66.5	35.8	61.2	65.9	0.0	50.4	60.3
SDP	1 (400)	59.1	73.8	84.7	**73.6**	70.7	12.6	54.8	94.9	**70.7**	65.7	29.3	65.6	**71.7**	0.0	49.4	61.9
	1 (500)	64.2	**74.4**	**86.0**	68.4	73.7	17.3	**58.4**	94.9	44.8	**66.9**	37.5	**67.3**	68.6	0.0	**54.6**	**65.1**
	1 (800)	**65.2**	73.5	**86.0**	61.0	**73.8**	16.9	57.1	94.3	44.1	65.5	36.5	61.5	61.9	0.0	49.9	63.1
SDP+CRC	1 (500)	63.9	74.3	85.8	68.2	73.5	17.5	52.0	93.7	45.9	65.5	35.1	65.7	69.2	0.0	52.9	64.0
SDP+CRC *ft*	1 (500)	63.9	74.2	85.5	62.9	73.7	17.6	50.0	93.4	61.0	65.9	35.8	66.5	67.6	0.0	53.1	64.2

Inner-city Dataset

		Car					*Pedestrian*					*Bike*					*Truck*					mAP
Methods	Inputs	S_1	S_2	S_3	S_4	S	S_1	S_2	S_3	S_4	S	S_1	S_2	S_3	S_4	S	S_1	S_2	S_3	S_4	S	S
FRCN [13]+AlexNet	4	74.6	78.9	82.9	94.9	82.4	43.9	69.1	77.8	75.4	63.7	26.2	42.3	45.9	2.2	36.3	28.7	51.5	60.0	67.0	48.7	55.0
FRCN [13]+VGG16	1	63.9	80.0	86.4	93.7	80.5	35.2	71.3	**83.3**	77.3	64.3	28.2	57.5	**68.7**	0.5	50.6	26.0	62.1	70.0	54.0	53.6	62.2
SDP	1	**76.2**	**84.2**	86.9	95.2	**85.5**	**51.1**	**78.0**	83.0	**81.5**	**73.9**	40.3	**65.4**	65.2	43.2	57.9	44.1	67.0	**71.5**	75.1	65.6	**70.7**
SDP+CRC	1	75.7	83.8	86.5	95.0	85.0	50.9	75.9	80.2	78.3	71.7	38.4	61.5	63.7	41.5	55.1	43.9	66.8	71.0	**75.6**	65.5	69.3
SDP+CRC *ft*	1	75.0	84.1	**87.2**	**95.6**	84.9	**51.1**	76.7	80.2	77.8	72.2	**41.6**	64.6	64.7	**46.9**	**58.2**	**45.8**	**69.1**	69.9	74.2	**66.4**	70.4

VOC2007 Dataset

Methods	train set	aero	bike	bird	boat	bottle	bus	car	cat	chair	cow	table	dog	horse	mbike	persn	plant	sheep	sofa	train	tv	mAP
SPPnet BB [16]	07 \ diff	73.9	72.3	62.5	51.5	44.4	74.4	73.0	74.4	42.3	73.6	57.7	70.3	74.6	74.3	54.2	34.0	56.4	56.4	67.9	73.5	63.1
R-CNN BB [14]	07	73.4	77.0	63.4	45.4	44.6	75.1	78.1	79.8	40.5	73.7	62.2	79.4	78.1	73.1	64.2	35.6	66.8	67.2	70.4	71.1	66.0
FRCN [13]	07	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	66.9
FRCN [13] *ms*	07	74.5	78.5	66.8	**57.8**	38.0	**80.1**	**78.7**	**83.5**	43.3	**74.5**	67.4	**81.5**	**82.8**	72.6	66.8	32.4	**67.1**	**70.6**	75.9	68.9	68.1
SDP	07	73.2	78.1	68.4	50.3	47.4	78.1	75.0	80.9	45.5	68.5	**68.3**	79.2	80.8	76.5	74.2	38.6	65.2	66.1	76.3	73.0	68.2
SDP *ms*	07	74.9	77.4	68.1	55.0	**49.8**	**80.1**	76.4	82.2	**47.3**	70.4	67.1	80.2	82.7	75.8	75.3	**40.6**	66.8	69.7	75.2	**73.8**	**69.4**
SDP+CRC	07	74.4	75.5	66.8	49.5	43.9	77.0	75.5	79.6	43.8	68.6	65.3	76.7	79.9	76.5	75.6	36.5	63.1	62.8	77.1	72.1	67.1
SDP+CRC *ms*	07	74.8	77.6	66.8	51.7	47.1	76.0	77.7	80.1	45.5	69.8	63.2	76.7	79.4	75.0	76.4	39.3	63.3	65.7	76.2	71.9	67.7
SDP+CRC *ft*	07	75.4	77.3	68.6	51.3	44.0	77.3	76.7	80.3	45.6	71.7	65.8	77.4	81.2	**77.0**	76.8	36.8	65.1	63.2	77.1	72.8	68.1
SDP+CRC *ms ft*	07	**76.1**	**79.4**	68.2	52.6	46.0	78.4	78.4	81.0	46.7	73.5	65.3	78.6	81.0	76.7	**77.3**	39.0	65.1	67.2	**77.5**	70.3	68.9

fine-tuning. We believe that it will achieve more improvement if all the model parameters including CRC modules are trained. We plan to explore this as a future direction.

Test set evaluation. To compare with existing approaches on KITTI test set, we train our SDP and CRC model on the full training set, and evaluate it on the test set. The results are shown in Table 2. We use the same configuration and learning parameters as in the previous experiments. Without using any stereo information, our approach outperforms all compared methods on all levels of difficulties and achieves the best results. In particular, Our method using SDP again outperforms the Fast RCNN baseline by 9% on average, verifying the effectiveness of SDP module. Notably, our method improves AP by 16.7% over Fast RCNN baseline on Hard case of Car category, where most samples are of small size or occluded. This is a clear evidence showing the discriminative power of our SDP module.

5.2. Discussions

Rejection ratio. By using CRCs, we aim to improve the efficiency for the proposal evaluation by progressively reducing the number of proposals. In Table 3, we analyze the percentage of surviving proposals with respect to the initial number of input proposals after applying CRCs, as well as the corresponding recall rate of positives after each CRC. The table shows that our CRCs successfully reject a large number of input proposals while keeping a high recall for the true objects. For each scale group, CRCs can remove over $70 \sim 80\%$ input proposals, so that only around $20 \sim 30\%$ proposals go through *fc* layers.

Table 3. Percentage (%) of surviving proposals after applying CRC, and the corresponding recall rate (%) on KITTI validation set. $\mathcal{R}_{[n_1,n_2)}$ refers to the rejection classifier for the scale group $[n_1, n_2)$.

	$\mathcal{R}_{[0,64)}$		$\mathcal{R}_{[64,128)}$		$\mathcal{R}_{[128,\infty)}$		Overall	
Layer	ratio	recall	ratio	recall	ratio	recall	ratio	recall
conv1_2	66.2	97.6	83.9	98.1	94.8	100	81.6	98.6
conv2_2	44.2	95.5	59.2	96.2	92.9	99.7	65.4	97.1
conv3_3	16.7	92.1	25.1	93.4	72.3	96.5	38.0	94.0
conv4_3	-	-	12.6	90.3	48.6	92.0	30.6	91.2
conv5_3	-	-	-	-	28.8	89.9	28.8	89.9

Run-time efficiency. We investigate the efficiency gain in-

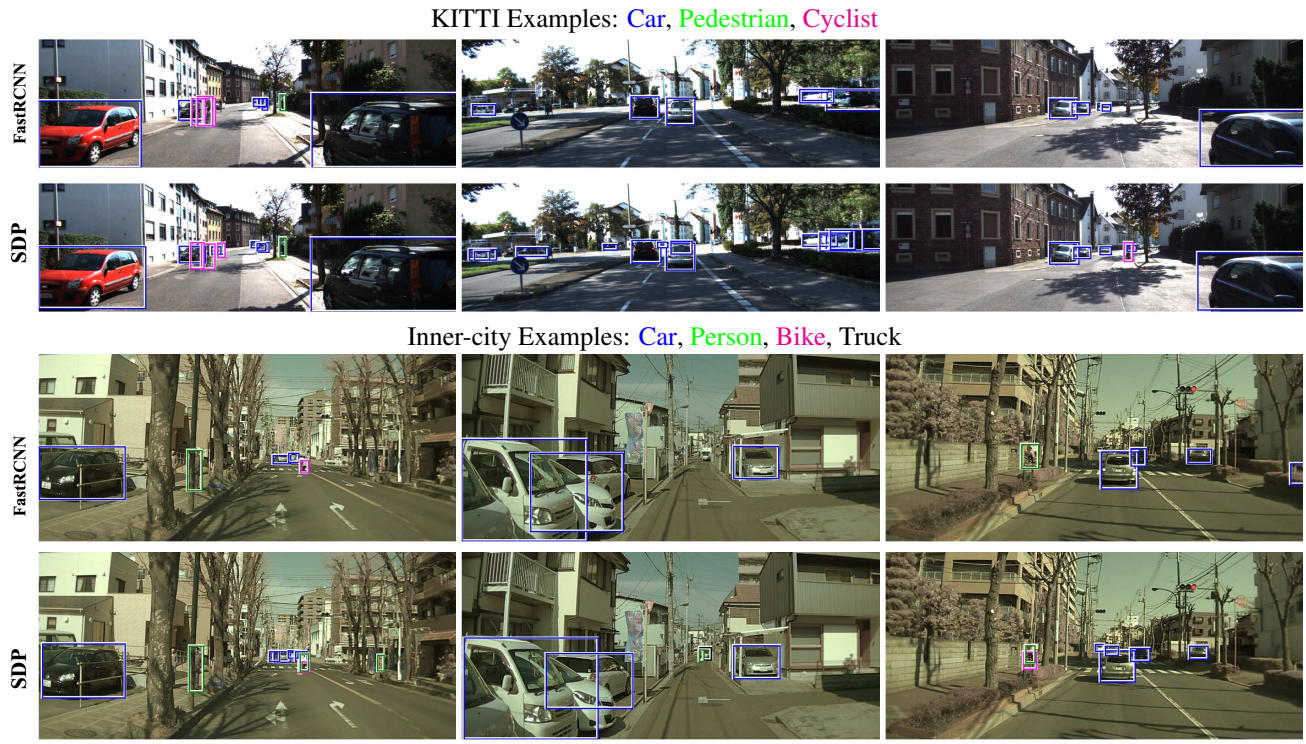

Figure 4. Qualitative results on KITTI validation set and Inner-city dataset using FRCN [13]+VGG16 baseline and our SDP model. We obtain the detection threshold for visualization at the precision 0.8. The color of bounding boxes means the types of objects. Notice that our method with SDP detects small objects much better than the baseline method. The figure is best shown in color.

troduced by CRCs. Table 4 analyzes detailed computational break-down of various methods. We measure the time spent in each component of the network, such as convolutional operations, *fc* layer computations, pre- and post-processing, etc. We compare our CRCs with the truncated SVD approach [13] that aims to reduce the dimensionality of the *fc* layers. We follow the strategy in [13] to keep the top 1024 singular values from the 1^{st} *fc* layer and the top 256 singular values from the 2^{nd} *fc* with respect to each SDP branch. In addition, we combine CRC and SVD, *i.e.*, using CRC to eliminate proposals and SVD to compress *fc* layers in SDPs, to achieve further speed-up. We include the baseline methods without SVD as a reference. The truncated SVD approach alone achieves about $2.3\times$ gain in proposal evaluations. The CRC modules alone obtain $3.2\times$ speed up for the same operation. We gain $4 \sim 5\times$ for each SDP by rejecting $70 \sim 80\%$ of proposals, but the additional computation introduced by CRC reduces the overall gain slightly. When combined with the SVD and CRC, we obtain $4.6\times$ efficiency gain in proposal evaluations and $2.4\times$ in total (including *conv* operations).

Table 4. Run-time comparison (ms per image) among the baseline methods, our method with truncated SVD [13], our method with CRC and SVD+CRC on KITTI dataset. fc_S, fc_M, and fc_L refer to the SDP classifiers for the scale group [0, 64), [64, 128), [128, ∞), respectively. "box eval." represents the time spent for individual box evaluation including *fc* layers and CRC rejections. The times were measured using an Nvidia K40 GPU under the same experimental environment.

Component	*conv*	*fc*	fc_S	fc_M	fc_L	rej.	box eval.	misc.	total
[13]+AlexNet	799	512	0	0	0	0	512	164	1476
[13]+VGG16	282	719	0	0	0	0	719	21	1022
SDP	286	0	204	254	283	0	741	90	1117
SVD	285	0	97	116	114	0	327	95	707
speedup	1.0	-	2.10	2.19	2.48	-	2.27	0.95	1.58
CRC	282	0	44	46	63	79	232	27	541
speedup	1.0	-	4.64	5.52	4.49	-	3.19	3.33	2.06
SVD+CRC	283	0	24	25	31	81	161	27	471
speedup	1.0	-	8.50	10.16	9.13	-	4.60	3.33	2.37

6. Conclusion

In this paper, we investigate two new strategies to detect objects efficiently using deep convolutional neural network, 1) scale-dependent pooling and 2) layer-wise cascaded rejection classifiers. The scale-dependent pooling (SDP) improves detection accuracy especially on small objects by fine-tuning a network with scale-specific branches attached after several convolutional layers. The cascaded rejection classifiers (CRC) effectively utilize convolutional features and eliminate negative object proposals in a cascaded manner, which greatly speeds up the detection while maintaining high accuracy. Our experimental evaluation clearly demonstrates the benefits of SDP and CRC in CNN based object detection.

References

[1] A. Angelova, A. Krizhevsky, V. Vanhoucke, A. Ogale, and D. Ferguson. Real-time pedestrian detection with deep network cascades. In *BMVC*, 2015. 2, 4

[2] H. Azizpour, A. S. Razavian, J. Sullivan, A. Maki, and S. Carlsson. From generic to specific deep representations for visual recognition. *CoRR*, abs/1406.5774, 2014. 4

[3] G. Bertasius, J. Shi, and L. Torresani. Deepedge: A multi-scale bifurcated deep network for top-down contour detection. In *CVPR*, pages 4380–4389, 2015. 3

[4] Z. Cai, M. Saberian, and N. Vasconcelos. Learning complexity-aware cascades for deep pedestrian detection. In *ICCV*, 2015. 6

[5] P. Dollár, R. Appel, S. Belongie, and P. Perona. Fast feature pyramids for object detection. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 36(8):1532–1545, 2014. 3, 5

[6] P. Dollár, R. Appel, and W. Kienzle. Crosstalk cascades for frame-rate pedestrian detection. In *ECCV*, pages 645–659, 2012. 2, 4

[7] D. Erhan, C. Szegedy, A. Toshev, and D. Anguelov. Scalable object detection using deep neural networks. In *CVPR*, pages 2155–2162, 2014. 1, 2

[8] M. Everingham, L. J. V. Gool, C. K. I. Williams, J. M. Winn, and A. Zisserman. The pascal visual object classes (VOC) challenge. *International Journal of Computer Vision*, 88(2):303–338, 2010. 2, 5

[9] P. F. Felzenszwalb, R. B. Girshick, and D. A. McAllester. Cascade object detection with deformable part models. In *CVPR*, pages 2241–2248, 2010. 2, 3

[10] Y. Freund, R. Schapire, and N. Abe. A short introduction to boosting. *Journal-Japanese Society For Artificial Intelligence*, 14(771-780):1612, 1999. 2, 4, 5

[11] A. Geiger, P. Lenz, and R. Urtasun. Are we ready for autonomous driving? the KITTI vision benchmark suite. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2012. 2, 5, 6

[12] A. Ghodrati, A. Diba, M. Pedersoli, T. Tuytelaars, and L. Van Gool. DeepProposal: Hunting objects by cascading deep convolutional layers. In *ICCV*, 2015. 3

[13] R. Girshick. Fast R-CNN. *arXiv preprint arXiv:1504.08083*, 2015. 1, 2, 3, 4, 5, 6, 7, 8

[14] R. Girshick, J. Donahue, T. Darrell, and J. Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *CVPR*, pages 580–587, 2014. 1, 2, 3, 7

[15] B. Hariharan, P. A. Arbeláez, R. B. Girshick, and J. Malik. Hypercolumns for object segmentation and fine-grained localization. In *CVPR*, pages 447–456, 2015. 2, 4

[16] K. He, X. Zhang, S. Ren, and J. Sun. Spatial pyramid pooling in deep convolutional networks for visual recognition. In *ECCV*, pages 346–361. 2014. 1, 3, 7

[17] N. Karianakis, T. J. Fuchs, and S. Soatto. Boosting convolutional features for robust object proposals. *CoRR*, abs/1503.06350, 2015. 3

[18] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012. 1, 4, 6

[19] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998. 1

[20] C. Lee, S. Xie, P. W. Gallagher, Z. Zhang, and Z. Tu. Deeply-supervised nets. *CoRR*, abs/1409.5185, 2014. 3

[21] H. Li, Z. Lin, X. Shen, J. Brandt, and G. Hua. A convolutional neural network cascade for face detection. In *CVPR*, pages 5325–5334, 2015. 2, 4

[22] L. Liu, C. Shen, and A. van den Hengel. The treasure beneath convolutional layers: Cross-convolutional-layer pooling for image classification. In *CVPR*, pages 4749–4757, 2015. 4

[23] J. Long, E. Shelhamer, and T. Darrell. Fully convolutional networks for semantic segmentation. In *CVPR*, pages 3431–3440, 2015. 3, 4

[24] M. Mathias, R. Benenson, R. Timofte, and L. J. V. Gool. Handling occlusions with franken-classifiers. In *ICCV*, pages 1505–1512, 2013. 2, 4

[25] J. Y. Ng, F. Yang, and L. S. Davis. Exploiting local features from deep networks for image retrieval. *CoRR*, abs/1504.05133, 2015. 4

[26] E. Ohn-Bar and M. M. Trivedi. Learning to detect vehicles by clustering appearance patterns. *T-ITS*, 2015. 6

[27] W. Ouyang, X. Wang, X. Zeng, S. Qiu, P. Luo, Y. Tian, H. Li, S. Yang, Z. Wang, C. C. Loy, and X. Tang. Deepid-net: Deformable deep convolutional neural networks for object detection. *CoRR*, abs/1412.5661, 2014. 2

[28] B. Pepik, M. Stark, P. V. Gehler, and B. Schiele. Multi-view and 3D deformable part models. *IEEE Trans. Pattern Anal. Mach. Intell.*, 37(11):2232–2245, 2015. 6

[29] S. Ren, K. He, R. B. Girshick, and J. Sun. Faster R-CNN: towards real-time object detection with region proposal networks. *CoRR*, abs/1506.01497, 2015. 2

[30] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, pages 1–42, April 2015. 5

[31] 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. 2

[32] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014. 2, 4, 5, 6

[33] Y. Sun, X. Wang, and X. Tang. Deep convolutional network cascade for facial point detection. In *CVPR*, pages 3476–3483, 2013. 2, 4

[34] C. Szegedy, A. Toshev, and D. Erhan. Deep neural networks for object detection. In *NIPS*, pages 2553–2561, 2013. 2

[35] Y. Tian, P. Luo, X. Wang, and X. Tang. Deep learning strong parts for pedestrian detection. In *ICCV*, 2015. 6

[36] P. A. Viola and M. J. Jones. Rapid object detection using a boosted cascade of simple features. In *CVPR*, pages 511–518, 2001. 2, 4

[37] X. Wang, M. Yang, S. Zhu, and Y. Lin. Regionlets for generic object detection. In *ICCV*, pages 17–24, 2013. 6

[38] Y. Xiang, W. Choi, Y. Lin, and S. Savarese. Data-driven 3D voxel patterns for object category recognition. In *CVPR*, 2015. 5, 6

[39] S. Xie and Z. Tu. Holistically-nested edge detection. *CoRR*, abs/1504.06375, 2015. 3, 4

[40] Z. Xu, Y. Yang, and A. G. Hauptmann. A discriminative CNN video representation for event detection. In *CVPR*, pages 1798–1807, 2015. 3, 4

[41] M. D. Zeiler and R. Fergus. Visualizing and understanding convolutional networks. In *ECCV*, pages 818–833. 2014. 1

[42] Y. Zhang, K. Sohn, R. Villegas, G. Pan, and H. Lee. Improving object detection with deep convolutional networks via bayesian optimization and structured prediction. In *CVPR*, pages 249–258, 2015. 1, 2

[43] B. Zhou, A. Khosla, A. Lapedriza, A. Oliva, and A. Torralba. Object detectors emerge in deep scene cnns. In *ICLR*, 2015. 1

[44] Y. Zhu, R. Urtasun, R. Salakhutdinov, and S. Fidler. segDeepM: Exploiting segmentation and context in deep neural networks for object detection. In *CVPR*, pages 4703–4711, 2015. 2

[45] C. L. Zitnick and P. Dollár. Edge boxes: Locating object proposals from edges. In *ECCV*, pages 391–405. 2014. 5