A Unified Multi-scale Deep Convolutional Neural Network for Fast Object Detection

Zhaowei Cai[1(✉)], Quanfu Fan[2], Rogerio S. Feris[2], and Nuno Vasconcelos[1]

[1] SVCL, UC San Diego, San Diego, USA
{zwcai,nuno}@ucsd.edu
[2] IBM T. J. Watson Research, Yorktown Heights, USA
{qfan,rsferi}@us.ibm.com

Abstract. A unified deep neural network, denoted the *multi-scale CNN* (MS-CNN), is proposed for fast multi-scale object detection. The MS-CNN consists of a proposal sub-network and a detection sub-network. In the proposal sub-network, detection is performed at multiple output layers, so that receptive fields match objects of different scales. These complementary scale-specific detectors are combined to produce a strong multi-scale object detector. The unified network is learned end-to-end, by optimizing a multi-task loss. Feature upsampling by deconvolution is also explored, as an alternative to input upsampling, to reduce the memory and computation costs. State-of-the-art object detection performance, at up to 15 fps, is reported on datasets, such as KITTI and Caltech, containing a substantial number of small objects.

Keywords: Object detection · Multi-scale · Unified neural network

1 Introduction

Classical object detectors, based on the sliding window paradigm, search for objects at multiple scales and aspect ratios. While real-time detectors are available for certain classes of objects, e.g. faces or pedestrians [1,2], it has proven difficult to build detectors of multiple object classes under this paradigm. Recently, there has been interest in detectors derived from deep convolutional neural networks (CNNs) [3–7]. While these have shown much greater ability to address the multiclass problem, less progress has been made towards the detection of objects at multiple scales. The R-CNN [3] samples object proposals at multiple scales, using a preliminary attention stage [8], and then warps these proposals to the size (e.g. 224×224) supported by the CNN. This is, however, very inefficient from a computational standpoint. The development of an effective and computationally efficient region proposal mechanism is still an open problem. The more recent Faster-RCNN [9] addresses the issue with a region proposal network (RPN), which enables end-to-end training. However, the RPN generates proposals of multiple scales by sliding a fixed set of filters over a fixed set of convolutional feature maps. This creates an inconsistency between the sizes of

B. Leibe et al. (Eds.): ECCV 2016, Part IV, LNCS 9908, pp. 354–370, 2016.
DOI: 10.1007/978-3-319-46493-0_22

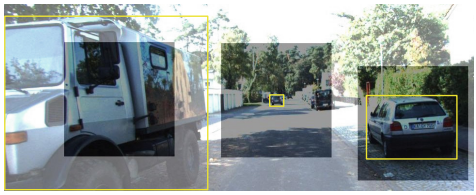


Fig. 1. In natural images, objects can appear at very different scales, as illustrated by the yellow bounding boxes. A single receptive field, such as that of the RPN [9] (shown in the shaded area), cannot match this variability.

objects, which are variable, and filter receptive fields, which are fixed. As shown in Fig. 1, a fixed receptive field cannot cover the multiple scales at which objects appear in natural scenes. This compromises detection performance, which tends to be particularly poor for small objects, like that in the center of Fig. 1. In fact, [4,5,9] handle such objects by upsampling the input image both at training and testing time. This increases the memory and computation costs of the detector.

This work proposes a unified multi-scale deep CNN, denoted the *multi-scale CNN* (MS-CNN), for fast object detection. Similar to [9], this network consists of two sub-networks: an object proposal network and an accurate detection network. Both of them are learned end-to-end and share computations. However, to ease the inconsistency between the sizes of objects and receptive fields, object detection is performed with multiple output layers, each focusing on objects within certain scale ranges (see Fig. 3). The intuition is that lower network layers, such as "conv-3," have smaller receptive fields, better matched to detect small objects. Conversely, higher layers, such as "conv-5," are best suited for the detection of large objects. The complimentary detectors at different output layers are combined to form a strong multi-scale detector. This is shown to produce accurate object proposals on detection benchmarks with large variation of scale, such as KITTI [10], achieving a recall of over 95 % for only 100 proposals.

A second contribution of this work is the use of feature upsampling as an alternative to input upsampling. This is achieved by introducing a deconvolutional layer that increases the resolution of feature maps (see Fig. 4), enabling small objects to produce larger regions of strong response. This is shown to reduce memory and computation costs. While deconvolution has been explored for segmentation [11] and edge detection [12], it is, as far as we know, for the first time used to speed up and improve detection. When combined with efficient context encoding and hard negative mining, it results in a detector that advances the state-of-the-art detection on the KITTI [10] and Caltech [13] benchmarks. Without image upsampling, the MS-CNN achieves speeds of 10 fps on KITTI (1250×375) and 15 fps on Caltech (640×480) images.

2 Related Work

One of the earliest methods to achieve real-time detection with high accuracy was the cascaded detector of [1]. This architecture has been widely used to

implement sliding window detectors for faces [1,14], pedestrians [2,15] and cars [16]. Two main streams of research have been pursued to improve its speed: fast feature extraction [1,2] and cascade learning [14,15,17]. In [1], a set of efficient Haar features was proposed with recourse to integral images. The aggregate feature channels (ACF) of [2] made it possible to compute HOG features at about 100 fps. On the learning front, [14] proposed the soft-cascade, a method to transform a classifier learned with boosting into a cascade with certain guarantees in terms of false positive and detection rate. [17] introduced a Lagrangian formulation to learn cascades that achieve the optimal trade-off between accuracy and computational complexity. [15] extended this formulation for cascades of highly heterogeneous features, ranging from ACF set to deep CNNs, with widely different complexity. The main current limitation of detector cascades is the difficulty of implementing multiclass detectors under this architecture.

In an attempt to leverage the success of deep neural networks for object classification, [3] proposed the R-CNN detector. This combines an object proposal mechanism [8] and a CNN classifier [18]. While the R-CNN surpassed previous detectors [19,20] by a large margin, its speed is limited by the need for object proposal generation and repeated CNN evaluation. [6] has shown that this could be ameliorated with recourse to spatial pyramid pooling (SPP), which allows the computation of CNN features once per image, increasing the detection speed by an order of magnitude. Building on SPP, the Fast-RCNN [4] introduced the ideas of back-propagation through the ROI pooling layer and multi-task learning of a classifier and a bounding box regressor. However, it still depends on bottom-up proposal generation. More recently, the Faster-RCNN [9] has addressed the generation of object proposals and classifier within a single neural network, leading to a significant speedup for proposal detection. Another interesting work is YOLO [21], which outputs object detections within a 7×7 grid. This network runs at $\sim$40 fps, but with some compromise of detection accuracy.

For object recognition, it has been shown beneficial to combine multiple losses, defined on intermediate layers of a single network [11,12,22,23]. GoogLeNet [22] proposed the use of three weighted classification losses, applied at layers of intermediate heights, showing that this type of regularization is useful for very deep models. The deeply supervised network architecture of [23] extended this idea to a larger number of layers. The fact that higher layers convey more semantic information motivated [11] to combine features from intermediate layers, leading to more accurate semantic segmentation. A similar idea was shown useful for edge detection in [12]. Similar to [11,12,22,23], the proposed MS-CNN is learned with losses that account for intermediate layer outputs. However, the aim is not to simply regularize the learning, as in [22,23], or provide detailed information for higher outputs, as in [11,12]. Instead, the goal is to produce a strong individual object detector at each intermediate output layer.

3 Multi-scale Object Proposal Network

In this section, we introduce the proposed network for the generation of object proposals.

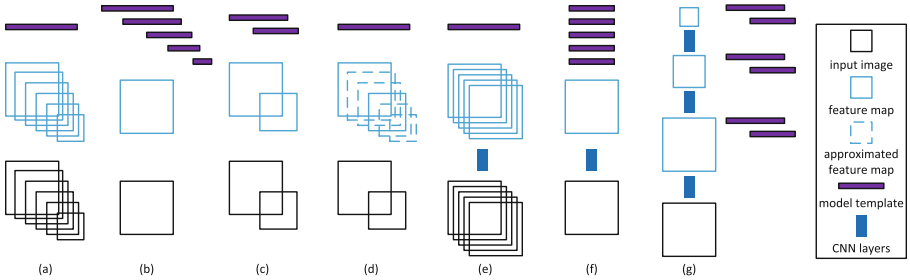


Fig. 2. Different strategies for multi-scale detection. The length of model template represents the template size.

3.1 Multi-scale Detection

The coverage of many object scales is a critical problem for object detection. Since a detector is basically a dot-product between a learned template and an image region, the template has to be matched to the spatial support of the object to recognize. There are two main strategies to achieve this goal. The first is to learn a single classifier and rescale the image multiple times, so that the classifier can match all possible object sizes. As illustrated in Fig. 2(a), this strategy requires feature computation at multiple image scales. While it usually produces the most accurate detection, it tends to be very costly. An alternative approach is to apply multiple classifiers to a single input image. This strategy, illustrated in Fig. 2(b), avoids the repeated computation of feature maps and tends to be efficient. However, it requires an individual classifier for each object scale and usually fails to produce good detectors. Several approaches have been proposed to achieve a good trade-off between accuracy and complexity. For example, the strategy of Fig. 2(c) is to rescale the input a few times and learn a small number of model templates [24]. Another possibility is the feature approximation of [2]. As shown in Fig. 2(d), this consists of rescaling the input a small number of times and interpolating the missing feature maps. This has been shown to achieve considerable speed-ups for a very modest loss of classification accuracy [2].

The implementation of multi-scale strategies on CNN-based detectors is slightly different from those discussed above, due to the complexity of CNN features. As shown in Fig. 2(e), the R-CNN of [3] simply warps object proposal patches to the natural scale of the CNN. This is somewhat similar to Fig. 2(a), but features are computed for patches rather than the entire image. The multi-scale mechanism of the RPN [9], shown in Fig. 2(f), is similar to that of Fig. 2(b). However, multiple sets of templates of the same size are applied to all feature maps. This can lead to a severe scale inconsistency for template matching. As shown in Fig. 1, the single scale of the feature maps, dictated by the (228×228) receptive field of the CNN, can be severely mismatched to small (e.g. 32×32) or large (e.g. 640×640) objects. This compromises object detection performance.

Inspired by previous evidence on the benefits of the strategy of Fig. 2(c) over that of Fig. 2(b), we propose a new multi-scale strategy, shown in Fig. 2(g).

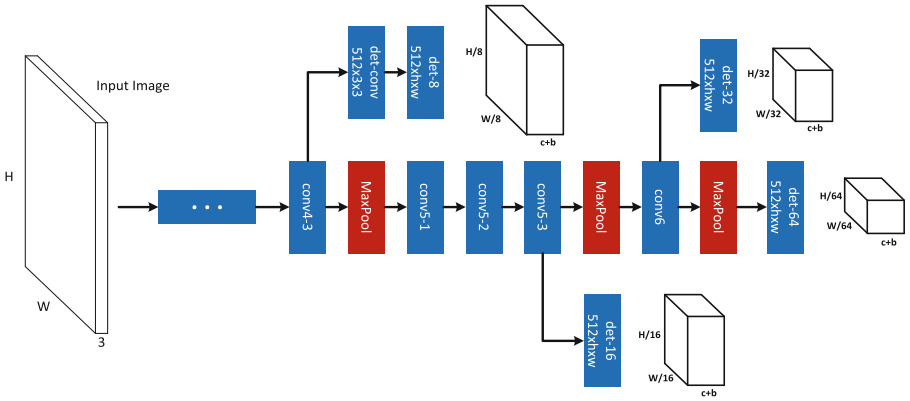


Fig. 3. Proposal sub-network of the MS-CNN. The bold cubes are the output tensors of the network. $h \times w$ is the filter size, c the number of classes, and b the number of bounding box coordinates.

This can be seen as the deep CNN extension of Fig. 2(c), but only uses a single scale of input. It differs from both Fig. 2(e) and (f) in that it exploits feature maps of several resolutions to detect objects at different scales. This is accomplished by the application of a set of templates at intermediate network layers. This results in a set of variable receptive field sizes, which can cover a large range of object sizes.

3.2 Architecture

The detailed architecture of the MS-CNN proposal network is shown in Fig. 3. The network detects objects through several detection branches. The results by all detection branches are simply declared as the final proposal detections. The network has a standard CNN trunk, depicted in the center of the figure, and a set of output branches, which emanate from different layers of the trunk. These branches consist of a single detection layer. Note that a buffer convolutional layer is introduced on the branch that emanates after layer "conv4-3". Since this branch is close to the lower layers of the trunk network, it affects their gradients more than the other detection branches. This can lead to some instability during learning. The buffer convolution prevents the gradients of the detection branch from being back-propagated directly to the trunk layers.

During training, the parameters $\mathbf{W}$ of the multi-scale proposal network are learned from a set of training samples $S = \{(X_i, Y_i)\}_{i=1}^{N}$, where X_i is a training image patch, and $Y_i = (y_i, b_i)$ the combination of its class label $y_i \in \{0, 1, 2, \cdots, K\}$ and bounding box coordinates $b_i = (b_i^x, b_i^y, b_i^w, b_i^h)$. This is achieved with a multi-task loss

$$\mathcal{L}(\mathbf{W}) = \sum_{m=1}^{M} \sum_{i \in S^m} \alpha_m l^m(X_i, Y_i | \mathbf{W}), \tag{1}$$

where M is the number of detection branches, α_m the weight of loss l^m, and $S = \{S^1, S^2, \cdots, S^M\}$, where S^m contains the examples of scale m. Note that only a subset S^m of the training samples, selected by scale, contributes to the loss of detection layer m. Inspired by the success of joint learning of classification and bounding box regression [4,9], the loss of each detection layer combines these two objectives

$$l(X, Y|\mathbf{W}) = L_{cls}(p(X), y) + \lambda[y \geq 1]L_{loc}(b, \hat{b}), \tag{2}$$

where $p(X) = (p_0(X), \cdots, p_K(X))$ is the probability distribution over classes, λ a trade-off coefficient, $L_{cls}(p(X), y) = -\log p_y(X)$ the cross-entropy loss, $\hat{b} = (\hat{b}_x, \hat{b}_y, \hat{b}_w, \hat{b}_h)$ the regressed bounding box, and

$$L_{loc}(b, \hat{b}) = \frac{1}{4} \sum_{j \in \{x,y,w,h\}} smooth_{L_1}(b_j, \hat{b}_j), \tag{3}$$

the smoothed bounding box regression loss of [4]. The bounding box loss is only used for positive samples and the optimal parameters $\mathbf{W}^* = \arg\min_{\mathbf{W}} \mathcal{L}(\mathbf{W})$ are learned by stochastic gradient descent.

3.3 Sampling

This section describes the assembly of training samples $S^m = \{S^m_+, S^m_-\}$ for each detection layer m. In what follows, the superscript m is dropped for notional simplicity. An anchor is centered at the sliding window on layer m associated with width and height corresponding to filter size. More details can be found in Table 1. A sample X of anchor bounding box b is labeled as positive if $o^* \geq 0.5$, where

$$o^* = \max_{i \in S_{gt}} IoU(b, b_i). \tag{4}$$

S_{gt} is the ground truth and IoU the intersection over union between two bounding boxes. In this case, $Y = (y_{i^*}, b_{i^*})$, where $i^* = \arg\max_{i \in S_{gt}} IoU(b, b_i)$ and (X, Y) are added to the positive set S_+. All the positive samples in $S_+ = \{(X_i, Y_i)|y_i \geq 1\}$ contribute to the loss. Samples such that $o^* < 0.2$ are assigned to a preliminary negative training pool, and the remaining samples discarded. For a natural image, the distribution of objects and non-objects is heavily asymmetric. Sampling is used to compensate for this imbalance. To collect a final set of negative samples $S_- = \{(X_i, Y_i)|y_i = 0\}$, such that $|S_-| = \gamma|S_+|$, we considered three sampling strategies: random, bootstrapping, and mixture.

Random sampling consists of randomly selecting negative samples according to a uniform distribution. Since the distribution of hard and easy negatives is heavily asymmetric too, most randomly collected samples are easy negatives. It is well known that hard negatives mining helps boost performance, since hard negatives have the largest influence on the detection accuracy. Bootstrapping accounts for this, by ranking the negative samples according to their objectness scores, and then collecting top $|S_-|$ negatives. Mixture sampling combines

the two, randomly sampling half of S_- and sampling the other half by bootstrapping. In our experiments, mixture sampling has very similar performance to bootstrapping.

To guarantee that each detection layer only detects objects in a certain range of scales, the training set for the layer consists of the subset of S that covers the corresponding scale range. For example, the samples of smallest scale are used to train the detector of "det-8" in Fig. 3. It is possible that no positive training samples are available for a detection layer, resulting in $|S_-|/|S_+| \gg \gamma$. This can make learning unstable. To address this problem, the cross-entropy terms of positives and negatives are weighted as follows

$$L_{cls} = \frac{1}{1+\gamma}\frac{1}{|S_+|}\sum_{i\in S_+} -\log p_{y_i}(X_i) + \frac{\gamma}{1+\gamma}\frac{1}{|S_-|}\sum_{i\in S_-} -\log p_0(X_i). \quad (5)$$

3.4 Implementation Details

Data Augmentation. In [4,6], it is argued that multi-scale training is not needed, since deep neural networks are adept at learning scale invariance. This, however, is not true for datasets such as Caltech [13] and KITTI [10], where object scales can span multiple octaves. In KITTI, many objects are quite small. Without rescaling, the cardinalities of the sets $S_+ = \{S_+^1, S_+^2, \cdots, S_+^M\}$ are wildly varying. In general, the set of training examples of largest object size is very small. To ease this imbalance, the original images are randomly resized to multiple scales.

Fine-Tuning. Training the Fast-RCNN [4] and RPN [9] networks requires large amounts of memory and a small mini-batch, due to the large size of the input (i.e. 1000×600). This leads to a very heavy training procedure. In fact, many background regions that are useless for training take substantially amounts of memory. Thus, we randomly crop a small patch (e.g. 448×448) around objects from the whole image. This drastically reduces the memory requirements, enabling four images to fit into the typical GPU memory of 12G.

Learning is initialized with the popular VGG-Net [25]. Since bootstrapping and the multi-task loss can make training unstable in the early iterations, a two-stage procedure is adopted. The first stage uses random sampling and a small trade-off coefficient λ (e.g. 0.05). 10,000 iterations are run with a learning rate of 0.00005. The resulting model is used to initialize the second stage, where random sampling is switched to bootstrapping and $\lambda = 1$. We set $\alpha_i = 0.9$ for "det-8" and $\alpha_i = 1$ for the other layers. Another 25,000 iterations are run with an initial learning rate of 0.00005, which decays 10 times after every 10,000 iterations. This two-stage learning procedure enables stable multi-task training.

4 Object Detection Network

Although the proposal network could work as a detector itself, it is not strong, since its sliding windows do not cover objects well. To increase detection

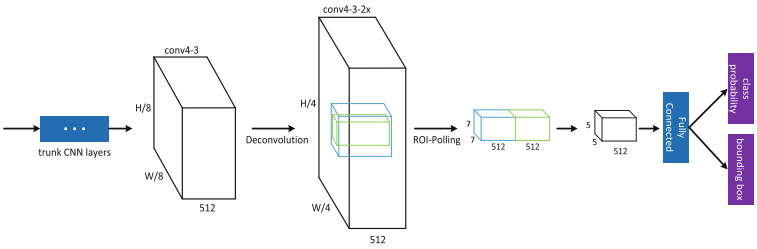


Fig. 4. Object detection sub-network of the MS-CNN. "trunk CNN layers" are shared with proposal sub-network. W and H are the width and height of the input image. The green (blue) cubes represent object (context) region pooling. "class probability" and "bounding box" are the outputs of the detection sub-network. (Color figure online)

accuracy, a detection network is added. Following [4], a ROI pooling layer is first used to extract features of a fixed dimension (e.g. $7 \times 7 \times 512$). The features are then fed to a fully connected layer and output layers, as shown in Fig. 4. A deconvolution layer, described in Sect. 4.1, is added to double the resolution of the feature maps. The multi-task loss of (1) is extended to

$$\mathcal{L}(\mathbf{W}, \mathbf{W}_d) = \sum_{m=1}^{M} \sum_{i \in S^m} \alpha_m l^m(X_i, Y_i | \mathbf{W}) + \sum_{i \in S^{M+1}} \alpha_{M+1} l^{M+1}(X_i, Y_i | \mathbf{W}, \mathbf{W}_d), \quad (6)$$

where l^{M+1} and S^{M+1} are the loss and training samples for the detection sub-network. S^{M+1} is collected as in [4]. As in (2), l^{M+1} combines a cross-entropy loss for classification and a smoothed L_1 loss for bounding box regression. The detection sub-network shares some of the proposal sub-network parameters $\mathbf{W}$ and adds some parameters $\mathbf{W}_d$. The parameters are optimized jointly, i.e. $(\mathbf{W}^*, \mathbf{W}_d^*) = \arg\min \mathcal{L}(\mathbf{W}, \mathbf{W}_d)$. In the proposed implementation, ROI pooling is applied to the top of the "conv4-3" layer, instead of the "conv5-3" layer of [4], since "conv4-3" feature maps performed better in our experiments. One possible explanation is that "conv4-3" corresponds to higher resolution and is better suited for location-aware bounding box regression.

4.1 CNN Feature Map Approximation

Input size has a critical role in CNN-based object detection accuracy. Simply forwarding object patches, at the original scale, through the CNN impairs performance (especially for small ones), since the pre-trained CNN models have a natural scale (e.g. 224×224). While the R-CNN naturally solves this problem through warping [3], it is not explicitly addressed by the Fast-RCNN [4] or Faster-RCNN [9]. To bridge the scale gap, these methods simply upsample input images (by $\sim$2 times). For datasets, such as KITTI [10], containing large amounts of small objects, this has limited effectiveness. Input upsampling also has three side effects: large memory requirements, slow training and slow testing. It should be noted that input upsampling does not enrich the image details.

Instead, it is needed because the higher convolutional layers respond very weakly to small objects. For example, a 32×32 object is mapped into a 4×4 patch of the "conv4-3" layer and a 2×2 patch of the "conv5-3" layer. This provides limited information for 7×7 ROI pooling.

To address this problem, we consider an efficient way to increase the resolution of feature maps. This consists of upsampling feature maps (instead of the input) using a deconvolution layer, as shown in Fig. 4. This strategy is similar to that of [2], shown in Fig. 2(d), where input rescaling is replaced by feature rescaling. In [2], a feature approximator is learned by least squares. In the CNN world, a better solution is to use a deconvolution layer, similar to that of [11]. Unlike input upsampling, feature upsampling does not incur in extra costs for memory and computation. Our experiments show that the addition of a deconvolution layer significantly boosts detection performance, especially for small objects. To the best of our knowledge, this is the first application of deconvolution to jointly improve the speed and accuracy of an object detector.

4.2 Context Embedding

Context has been shown useful for object detection [5,7,26] and segmentation [27]. Context information has been modeled by a recurrent neural network in [26] and acquired from multiple regions around the object location in [5,7,27]. In this work, we focus on context from multiple regions. As shown in Fig. 4, features from an object (green cube) and a context (blue cube) region are stacked together immediately after ROI pooling. The context region is 1.5 times larger than the object region. An extra convolutional layer without padding is used to reduce the number of model parameters. It helps compress redundant context and object information, without loss of accuracy, and guarantees that the number of model parameters is approximately the same.

4.3 Implementation Details

Learning is initialized with the model generated by the first learning stage of the proposal network, described in Sect. 3.4. The learning rate is set to 0.0005, and reduced by a factor of 10 times after every 10,000 iterations. Learning stops after 25,000 iterations. The joint optimization of (6) is solved by back-propagation throughout the unified network. Bootstrapping is used and $\lambda = 1$. Following [4], the parameters of layers"conv1-1" to "conv2-2" are fixed during learning, for faster training.

5 Experimental Evaluation

The performance of the MS-CNN detector was evaluated on the KITTI [10] and Caltech Pedestrian [13] benchmarks. These were chosen because, unlike VOC [28] and ImageNet [29], they contain many small objects. Typical image sizes are 1250×375 on KITTI and 640×480 on Caltech. KITTI contains three

Table 1. Parameter configurations of the different models.

		det-8		det-16		det-32		det-64	ROI	FC
car	filter	5 × 5	7 × 7	5 × 5	7 × 7	5 × 5	7 × 7	5 × 5	7 × 7	4096
	anchor	40 × 40	56 × 56	80 × 80	112 × 112	160 × 160	224 × 224	320 × 320		
ped/cyc	filter	5 × 3	7 × 5	5 × 3	7 × 5	5 × 3	7 × 5	5 × 3	7 × 5	2048
	anchor	40 × 28	56 × 36	80 × 56	112 × 72	160 × 112	224 × 144	320 × 224		
caltech	filter	5 × 3	7 × 5	5 × 3	7 × 5	5 × 3	7 × 5	5 × 3	8 × 4	2048
	anchor	40 × 20	56 × 28	80 × 40	112 × 56	160 × 80	224 × 112	320 × 160		

Table 2. Detection recall of the various detection layers on KITTI validation set (car), as a function of object hight in pixels.

	det-8	det-16	det-32	det-64	combined
25≤height<50	0.9180	0.3071	0.0003	0	0.9360
50≤height<100	0.5934	0.9660	0.4252	0	0.9814
100≤height<200	0.0007	0.5997	0.9929	0.4582	0.9964
height≥200	0	0	0.9583	0.9792	0.9583
all scales	0.6486	0.5654	0.3149	0.0863	0.9611

object classes: car, pedestrian and cyclist, and three levels of evaluation: easy, moderate and hard. The "moderate" level is the most commonly used. In total, 7,481 images are available for training/validation, and 7,518 for testing. Since no ground truth is available for the test set, we followed [5], splitting the trainval set into training and validation sets. In all ablation experiments, the training set was used for learning and the validation set for evaluation. Following [5], a model was trained for car detection and another for pedestrian/cyclist detection. One pedestrian model was learned on Caltech. The model configurations for original input size are shown in Table 1. The detector was implemented in C++ within the Caffe toolbox [30], and source code is available at https://github.com/zhaoweicai/mscnn. All times are reported for implementation on a single CPU core (2.40 GHz) of an Intel Xeon E5-2630 server with 64 GB of RAM. An NVIDIA Titan GPU was used for CNN computations.

5.1 Proposal Evaluation

We start with an evaluation of the proposal network. Following [31], oracle recall is used as performance metric. For consistency with the KITTI setup, a ground truth is recalled if its best matched proposal has IoU higher than 70 % for cars, and 50 % for pedestrians and cyclists.

The Roles of Individual Detection Layers. Table 2 shows the detection accuracy of the various detection layers as a function of object height in pixels. As expected, each layer has highest accuracy for the objects that match its scale. While the individual recall across scales is low, the combination of all detectors achieves high recall for all object scales.

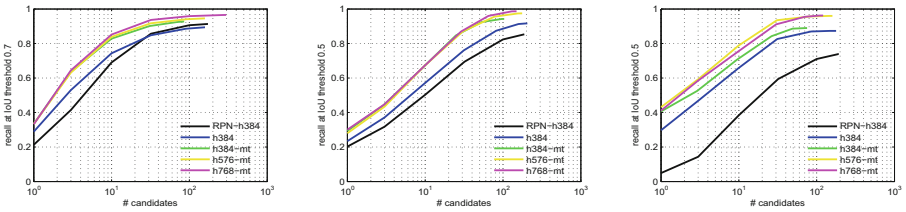


Fig. 5. Proposal recall on the KITTI validation set (moderate). "hXXX" refers to input images of height "XXX". "mt" indicates multi-task learning of proposal and detection sub-networks.

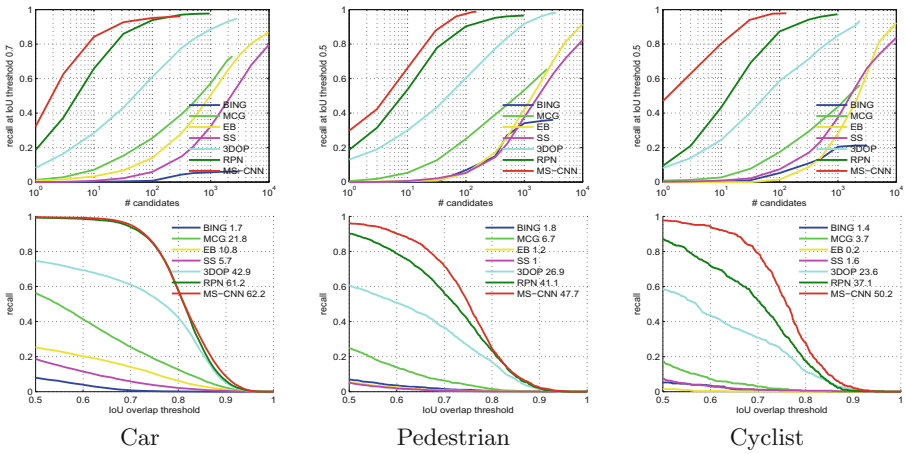
The Effect of Input Size. Fig. 5 shows that the proposal network is fairly robust to the size of input images for cars and pedestrians. For cyclist, performance increases between heights 384 and 576, but there are no gains beyond this. These results show that the network can achieve good proposal generation performance without substantial input upsampling.

Detection Sub-network Improves Proposal Sub-network. [4] has shown that multi-task learning can benefit both bounding box regression and classification. On the other hand [9] showed that, even when features are shared between the two tasks, object detection does not improve object proposals too much. Figure 5 shows that, for the MS-CNN, detection can substantially benefit proposal generation, especially for pedestrians.

Comparison with the State-of-the-art. Figure 6 compares the proposal generation network to BING [32], Selective Search [8], EdgeBoxes [33], MCG [34], 3DOP [5] and RPN [9]. The top row of the figure shows that the MS-CNN achieves a recall about 98 % with only 100 proposals. This should be compared to the ∼2,000 proposals required by 3DOP and the ∼10,000 proposals required by EdgeBoxbes. While it is not surprising that the proposed network outperforms unsupervised proposal methods, such as [8,33,34], its large gains over supervised methods [5,32], that can even use 3D information, are significant. The closest performance is achieved by RPN (input upsampled twice), which has substantially weaker performance for pedestrians and cyclists. When the input is not upsampled, RPN misses even more objects, as shown in Fig. 5. It is worth mentioning that the MS-CNN generates high quality proposals (high overlap with the ground truth) without any edge detection or segmentation. This is evidence for the effectiveness of bounding box regression networks.

5.2 Object Detection Evaluation

In this section we evaluate object detection performance. Since the performance of the cyclist detector has large variance on the validation set, due to the low number of cyclist occurrences, only car and pedestrian detection are considered in the ablation experiments.



Car Pedestrian Cyclist

Fig. 6. Proposal performance comparison on KITTI validation set (moderate). The first row is proposal recall curves and the second row is recall v.s. IoU for 100 proposals.

Table 3. Results on the KITTI validation set. "hXXX" indicates an input of height "XXX", "2x" deconvolution, "ctx" context encoding, and "c" dimensionality reduction convolution. In columns "Time" and "# params", entries before the "/" are for car model and after for pedestrian/cyclist model.

Model	Time	# params	Cars			Pedestrians		
			Easy	Mod	Hard	Easy	Mod	Hard
h384	0.11 s/0.09 s	471M/217M	90.90	80.63	68.94	73.70	68.37	60.72
h576	0.22 s/0.19 s	471M/217M	90.42	88.14	73.44	75.35	70.77	63.07
h768	0.41 s/0.36 s	471M/217M	89.84	88.88	75.78	76.38	72.26	64.08
h576-random	0.22 s/0.19 s	471M/217M	90.94	87.50	71.27	70.69	65.91	58.28
h576-mixture	0.22 s/0.19 s	471M/217M	90.33	88.12	72.90	75.09	70.49	62.43
h384-2x	0.12 s/0.10 s	471M/217M	90.55	87.93	71.90	76.01	69.53	61.57
h576-2x	0.23 s/0.20 s	471M/217M	94.08	89.12	75.54	77.74	72.49	64.43
h768-2x	0.43 s/0.38 s	471M/217M	90.96	88.83	75.19	76.33	72.71	64.31
h576-ctx	0.24 s/0.20 s	863M/357M	92.89	88.88	74.34	76.89	71.45	63.50
h576-ctx-c	0.22 s/0.19 s	297M/155M	90.49	89.13	74.85	76.82	72.13	64.14
proposal network (h576)	0.19 s/0.18 s	80M/78M	82.73	73.49	63.22	64.03	60.54	55.07

The Effect of Input Upsampling. Table 3 shows that input upsampling can be a crucial factor for detection. A significant improvement is obtained by upsampling the inputs by 1.5∼2 times, but we saw little gains beyond a factor of 2. This is smaller than the factor of 3.5 required by [5]. Larger factors lead to (exponentially) slower detectors and larger memory requirements.

Sampling Strategy. Table 3 compares sampling strategies: random ("h576-random"), bootstrapping ("h576") and mixture ("h576-mixture"). For car, these three strategies are close to each other. For pedestrian, bootstrapping and mixture are close, but random is much worse. Note that random sampling has many more false positives than the other two.

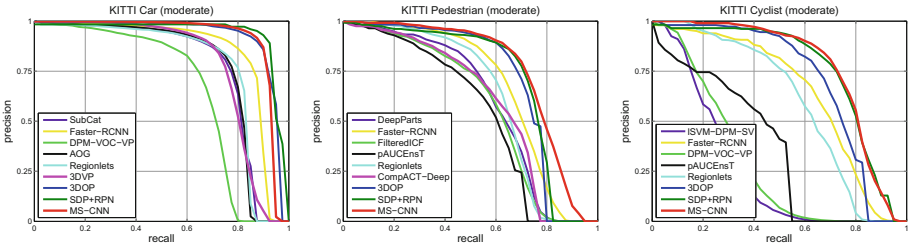


Fig. 7. Comparison to the state-of-the-art on KITTI benchmark test set (moderate).

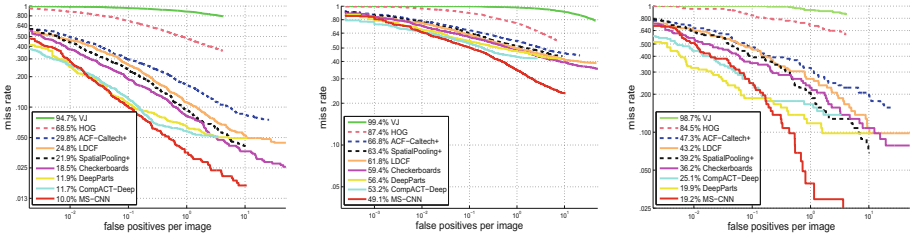
Table 4. Results on the KITTI benchmark test set (only published works shown).

Method	Time	Cars			Pedestrians			Cyclists		
		Easy	Mod	Hard	Easy	Mod	Hard	Easy	Mod	Hard
LSVM-MDPM-sv [35]	10 s	68.02	56.48	44.18	47.74	39.36	35.95	35.04	27.50	26.21
DPM-VOC-VP [36]	8 s	74.95	64.71	48.76	59.48	44.86	40.37	42.43	31.08	28.23
SubCat [16]	0.7 s	84.14	75.46	59.71	54.67	42.34	37.95	-	-	-
3DVP [37]	40 s	87.46	75.77	65.38	-	-	-	-	-	-
AOG [38]	3 s	84.80	75.94	60.70	-	-	-	-	-	-
Faster-RCNN [9]	2 s	86.71	81.84	71.12	78.86	65.90	61.18	72.26	63.35	55.90
CompACT-Deep [15]	1 s	-	-	-	70.69	58.74	52.71	-	-	-
DeepParts [39]	1 s	-	-	-	70.49	58.67	52.78	-	-	-
FilteredICF [40]	2 s	-	-	-	67.65	56.75	51.12	-	-	-
pAUCEnsT [41]	60 s	-	-	-	65.26	54.49	48.60	51.62	38.03	33.38
Regionlets [20]	1 s	84.75	76.45	59.70	73.14	61.15	55.21	70.41	58.72	51.83
3DOP [5]	3 s	**93.04**	88.64	**79.10**	81.78	67.47	64.70	78.39	68.94	61.37
SDP+RPN [42]	0.4 s	90.14	88.85	78.38	80.09	70.16	64.82	81.37	73.74	65.31
MS-CNN	0.4 s	90.03	**89.02**	76.11	**83.92**	**73.70**	**68.31**	**84.06**	**75.46**	**66.07**

CNN Feature Approximation. Three methods were attempted for learning the deconvolution layer for feature map approximation: (1) bilinearly interpolated weights; (2) weights initialized by bilinear interpolation and learned with back-propagation; (3) weights initialized with Gaussian noise and learned by back-propagation. We found the first method to work best, confirming the findings of [11,12]. As shown in Table 3, the deconvolution layer helps in most cases. The gains are larger for smaller input images, which tend to have smaller objects. Note that the feature map approximation adds trivial computation and no parameters.

Context Embedding. Table 3 shows that there is a gain in encoding context. However, the number of model parameters almost doubles. The dimensionality reduction convolution layer significantly reduces this problem, without impairment of accuracy or speed.

Object Detection by the Proposal Network. The proposal network can work as a detector, by switching the class-agnostic classification to class-specific. Table 3 shows that, although not as strong as the unified network, it achieves



(a) reasonable (b) medium (c) partial occlusion

Fig. 8. Comparison to the state-of-the-art on Caltech.

fairly good results, which are better than those of some detectors on the KITTI leaderboard[1].

Comparison to the State-of-the-art. The results of model "h768-ctx-c" were submitted to the KITTI leaderboard. A comparison to previous approaches is given in Table 4 and Fig. 7. The MS-CNN set a new record for the detection of pedestrians and cyclists. The columns "Pedestrians-Mod" and "Cyclists-Mod" show substantial gains (6 and 7 points respectively) over 3DOP [5], and much better performance than the Faster-RCNN [9], Regionlets [20], etc. We also led a nontrivial margin over the very recent SDP+RPN [42], which used scale dependent pooling. In terms of speed, the network is fairly fast. For the largest input size, the MS-CNN detector is about 8 times faster than 3DOP. On the original images (1250×375) detection speed reaches 10 fps.

Pedestrian Detection on Caltech. The MS-CNN detector was also evaluated on the Caltech pedestrian benchmark. The model "h720-ctx" was compared to methods such as DeepParts [39], CompACT-Deep [15], CheckerBoard [40], LDCF [43], ACF [2], and SpatialPooling [41] on three tasks: reasonable, medium and partial occlusion. As shown in Fig. 8, the MS-CNN has state-of-the-art performance. Figure 8(b) and (c) show that it performs very well for small and occluded objects, outperforming DeepParts [39], which explicitly addresses occlusion. Moreover, it misses a very small number of pedestrians, due to the accuracy of the proposal network. The speed is approximately 8 fps (15 fps) on upsampled 960×720 (original 640×480) Caltech images.

6 Conclusions

We have proposed a unified deep convolutional neural network, denoted the MS-CNN, for fast multi-scale object detection. The detection is preformed at various intermediate network layers, whose receptive fields match various object scales. This enables the detection of all object scales by feedforwarding a single input image through the network, which results in a very fast detector. CNN feature

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

approximation was also explored, as an alternative to input upsampling. It was shown to result in significant savings in memory and computation. Overall, the MS-CNN detector achieves high detection rates at speeds of up to 15 fps.

Acknowledgements. This work was partially funded by NSF grant IIS1208522 and a gift from KETI. We also thank NVIDIA for GPU donations through their academic program.

References

1. Viola, P.A., Jones, M.J.: Robust real-time face detection. Int. J. Comput. Vis. **57**(2), 137–154 (2004)
2. Dollár, P., Appel, R., Belongie, S.J., Perona, P.: Fast feature pyramids for object detection. IEEE Trans. Pattern Anal. Mach. Intell. **36**(8), 1532–1545 (2014)
3. Girshick, R.B., Donahue, J., Darrell, T., Malik, J.: Rich feature hierarchies for accurate object detection and semantic segmentation. In: CVPR, pp. 580–587(2014)
4. Girshick, R.B.: Fast R-CNN. In: ICCV, pp. 1440–1448(2015)
5. Chen, X., Kundu, K., Zhu, Y., Berneshawi, A., Ma, H., Fidler, S., Urtasun, R.: 3D object proposals for accurate object class detection. In: NIPS (2015)
6. He, K., Zhang, X., Ren, S., Sun, J.: Spatial pyramid pooling in deep convolutional networks for visual recognition. In: Fleet, D., Pajdla, T., Schiele, B., Tuytelaars, T. (eds.) ECCV 2014. LNCS, vol. 8691, pp. 346–361. Springer, Heidelberg (2014). doi:10.1007/978-3-319-10578-9_23
7. Gidaris, S., Komodakis, N.: Object detection via a multi-region and semantic segmentation-aware CNN model. In: ICCV, pp. 1134–1142(2015)
8. van de Sande, K.E.A., Uijlings, J.R.R., Gevers, T., Smeulders, A.W.M.: Segmentation as selective search for object recognition. In: ICCV, pp. 1879–1886(2011)
9. Ren, S., He, K., Girshick, R., Sun, J.: Faster R-CNN: Towards real-time object detection with region proposal networks. In: NIPS (2015)
10. Geiger, A., Lenz, P., Urtasun, R.: Are we ready for autonomous driving? the KITTI vision benchmark suite. In: CVPR, pp. 3354–3361(2012)
11. Long, J., Shelhamer, E., Darrell, T.: Fully convolutional networks for semantic segmentation. In: CVPR, pp. 3431–3440 (2015)
12. Xie, S., Tu, Z.: Holistically-nested edge detection. In: ICCV, pp. 1395–1403 (2015)
13. Dollár, P., Wojek, C., Schiele, B., Perona, P.: Pedestrian detection: An evaluation of the state of the art. IEEE Trans. Pattern Anal. Mach. Intell. **34**(4), 743–761 (2012)
14. Bourdev, L.D., Brandt, J.: Robust object detection via soft cascade. In: CVPR, pp. 236–243 (2005)
15. Cai, Z., Saberian, M.J., Vasconcelos, N.: Learning complexity-aware cascades for deep pedestrian detection. In: ICCV, pp. 3361–3369 (2015)
16. Ohn-Bar, E., Trivedi, M.M.: Learning to detect vehicles by clustering appearance patterns. IEEE Trans. Intell. Transp. Syst. **16**(5), 2511–2521 (2015)
17. Saberian, M.J., Vasconcelos, N.: Boosting algorithms for detector cascade learning. J. Mach. Learn. Res. **15**(1), 2569–2605 (2014)
18. Krizhevsky, A., Sutskever, I., Hinton, G.E.: Imagenet classification with deep convolutional neural networks. In: NIPS, pp. 1106–1114(2012)
19. Felzenszwalb, P.F., Girshick, R.B., McAllester, D.A., Ramanan, D.: Object detection with discriminatively trained part-based models. IEEE Trans. Pattern Anal. Mach. Intell. **32**(9), 1627–1645 (2010)

20. Wang, X., Yang, M., Zhu, S., Lin, Y.: Regionlets for generic object detection. In: ICCV, pp. 17–24 (2013)
21. Redmon, J., Divvala, S.K., Girshick, R.B., Farhadi, A.: You only look once: Unified, real-time object detection. In: CVPR (2016)
22. Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., Rabinovich, A.: Going deeper with convolutions. In: CVPR, pp. 1–9 (2015)
23. Lee, C., Xie, S., Gallagher, P.W., Zhang, Z., Tu, Z.: Deeply-supervised nets. In: AISTATS (2015)
24. Benenson, R., Mathias, M., Timofte, R., Gool, L.J.V.: Pedestrian detection at 100 frames per second. In: CVPR, pp. 2903–2910 (2012)
25. Simonyan, K., Zisserman, A.: Very deep convolutional networks for large-scale image recognition. CoRR abs/1409.1556 (2014)
26. Bell, S., Zitnick, C.L., Bala, K., Girshick, R.B.: Inside-outside net: Detecting objects in context with skip pooling and recurrent neural networks. In: CVPR (2016)
27. Zhu, Y., Urtasun, R., Salakhutdinov, R., Fidler, S.: segDeepM: Exploiting segmentation and context in deep neural networks for object detection. In: CVPR, pp. 4703–4711 (2015)
28. Everingham, M., Gool, L.J.V., Williams, C.K.I., Winn, J.M., Zisserman, A.: The pascal visual object classes (VOC) challenge. Int. J. Comput. Vis. **88**(2), 303–338 (2010)
29. Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M.S., Berg, A.C., Li, F.: Imagenet large scale visual recognition challenge. Int. J. Comput. Vis. **115**(3), 211–252 (2015)
30. Jia, Y., Shelhamer, E., Donahue, J., Karayev, S., Long, J., Girshick, R.B., Guadarrama, S., Darrell, T.: Caffe: Convolutional architecture for fast feature embedding. In: MM, pp. 675–678 (2014)
31. Hosang, J., Benenson, R., Dollár, P., Schiele, B.: What makes for effective detection proposals? PAMI **38**(4), 814–830 (2015)
32. Cheng, M., Zhang, Z., Lin, W., Torr, P.H.S.: BING: binarized normed gradients for objectness estimation at 300fps. In: CVPR, pp. 3286–3293 (2014)
33. Zitnick, C.L., Dollár, P.: Edge boxes: locating object proposals from edges. In: Fleet, D., Pajdla, T., Schiele, B., Tuytelaars, T. (eds.) ECCV 2014. LNCS, vol. 8693, pp. 391–405. Springer, Heidelberg (2014). doi:10.1007/978-3-319-10602-1_26
34. Arbeláez, P.A., Pont-Tuset, J., Barron, J.T., Marqués, F., Malik, J.: Multiscale combinatorial grouping. In: CVPR, pp. 328–335 (2014)
35. Geiger, A., Wojek, C., Urtasun, R.: Joint 3D estimation of objects and scene layout. In: NIPS, pp. 1467–1475 (2011)
36. Pepik, B., Stark, M., Gehler, P.V., Schiele, B.: Multi-view and 3D deformable part models. IEEE Trans. Pattern Anal. Mach. Intell. **37**(11), 2232–2245 (2015)
37. Xiang, Y., Choi, W., Lin, Y., Savarese, S.: Data-driven 3D voxel patterns for object category recognition. In: CVPR, pp. 1903–1911 (2015)
38. Li, B., Wu, T., Zhu, S.-C.: Integrating context and occlusion for car detection by hierarchical and-or model. In: Fleet, D., Pajdla, T., Schiele, B., Tuytelaars, T. (eds.) ECCV 2014. LNCS, vol. 8694, pp. 652–667. Springer, Heidelberg (2014). doi:10.1007/978-3-319-10599-4_42
39. Tian, Y., Luo, P., Wang, X., Tang, X.: Deep learning strong parts for pedestrian detection. In: ICCV, pp. 1904–1912 (2015)
40. Zhang, S., Benenson, R., Schiele, B.: Filtered channel features for pedestrian detection. In: CVPR, pp. 1751–1760 (2015)

41. Paisitkriangkrai, S., Shen, C., van den Hengel, A.: Pedestrian detection with spatially pooled features and structured ensemble learning. CoRR abs/1409.5209 (2014)
42. Yang, F., Choi, W., Lin, Y.: Exploit all the layers: Fast and accurate CNN object detector with scale dependent pooling and cascaded rejection classifiers. In: CVPR. (2016)
43. Nam, W., Dollár, P., Han, J.H.: Local decorrelation for improved pedestrian detection. In: NIPS, pp. 424–432 (2014)