BDCN: Bi-Directional Cascade Network for Perceptual Edge Detection

Jianzhong He, Shiliang Zhang, *Member IEEE,* Ming Yang, *Member IEEE,* Yanhu Shan, and Tiejun Huang, *Senior Member IEEE*

Abstract—Exploiting multi-scale representations is critical to improve edge detection for objects at different scales. To extract edges at dramatically different scales, we propose a Bi-Directional Cascade Network (BDCN) architecture, where an individual layer is supervised by labeled edges at its specific scale, rather than directly applying the same supervision to different layers. Furthermore, to enrich multi-scale representations learned by each layer of BDCN, we introduce a Scale Enhancement Module (SEM), which utilizes dilated convolution to generate multi-scale features, instead of using deeper CNNs. These new approaches encourage the learning of multi-scale representations in different layers and detect edges that are well delineated by their scales. Learning scale dedicated layers also results in a compact network with a fraction of parameters. We evaluate our method on three datasets, *i.e.*, *BSDS500*, *NYUDv2*, and *Multicue*, and achieve ODS F-measure of 0.832, 2.7% higher than current state-of-the-art on the *BSDS500* dataset. We also applied our edge detection result to other vision tasks. Experimental results show that, our method further boosts the performance of image segmentation, optical flow estimation, and object proposal generation.

Index Terms—Edge Detection, Bi-Directional Cascade Network, Scale Enhancement, Convolutional Neural Network.

1 Introduction

Edge detection targets on extracting object boundaries and perceptually salient edges from natural images, which preserve the gist of an image and ignore unintended details. Thus, it is important to a variety of mid- and high-level vision tasks, such as image segmentation [1], [2], object detection [3], [4], optical flow estimation [5] and image recognition [3], *etc*. In the past several decades, edge detection has attracted constant attention. Thanks to research efforts ranging from exploiting low-level visual cues with hand-crafted features [1], [4], [6], [7], [8] to recent deep models [9], [10], [11], [12], the accuracy of edge detection has been significantly boosted. For example, on the Berkeley Segmentation Data Set and Benchmarks 500 (*BSDS500*) [1], the detection performance has been improved from 0.598 [13] to 0.815 [12] in ODS F-measure.

Nevertheless, there remain several open issues worthy of studying. As shown in Fig. 1, edges in one image stem from both object-level boundaries and meaningful local details, *e.g.*, the contour of the koala and the boundaries of its eyes. Edges also vary considerably in scales, *e.g.*, the boundaries of two persons in one image. The variety of scale of edges makes it crucial to exploit multi-scale representations for edge detection. Recent neural network based methods [14], [15], [16] utilize hierarchical features learned by Convolutional Neural Networks (CNN) to obtain multi-scale representations. To generate more powerful multi-scale representation, some researchers adopt very deep CNNs, like ResNet50 [17], as the backbone of the edge detector. Another way is to build an image pyramid and fuse multi-level features. Deeper CNNs generally involve more parameters, making the network hard to train and expensive to infer. Image pyramid computation may involve redundant computations. Therefore, it is appealing to design a lightweight CNN to exploit multi-scale representation and achieve a comparable or even better performance.

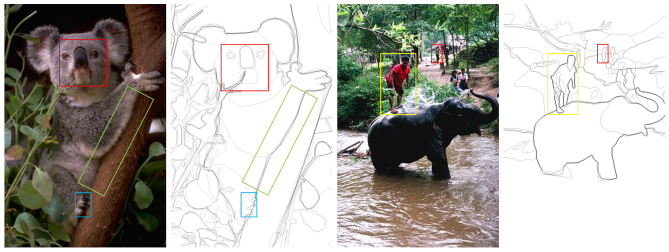


Fig. 1. Sample images and their ground truth edge maps in *BSDS500* dataset. The scale of edges in one image varies considerably, like the boundaries of two persons annotated with yellow and red bounding boxes, respectively.

Another issue is about the CNN training strategy for edge detection. Most of previous works supervise predictions of different network layers by one general ground truth edge map [10], [16]. For instance, HED [16], [18] and RCF [10] compute edge prediction on each intermediate CNN output to spot edges at different scales, *i.e.*, the lower layers are expected to detect local image patterns while higher layers capture object-level information with larger receptive fields. Since different network layers attend to depict patterns at different scales, it is not optimal to train those layers with the same supervision. In another word, existing works [10], [16], [18] enforce each layer of CNN

- *J. He, S. Zhang, and T. Huang are with the Department of EECS, Peking University, Beijing, China, 100871. email: {jianzhonghe, slzhang.jdl, tjhuang}@pku.edu.cn*
- *M. Yang and Y. Shan are with Horizon Robotics, Inc., Beijing, China, 100871. email: m-yang4@u.northwestern.edu and yanhu.shan@gmail.com*

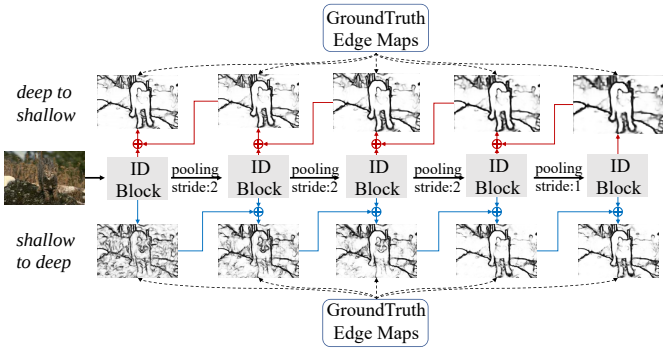


Fig. 2. The overall architecture of BDCN. ID Block denotes the Incremental Detection Block, which is the basic component of BDCN. Each ID Block is trained by a layer-specific supervision inferred by the bi-directional cascade structure. This structure trains each ID Block to spot edges at a proper scale. The predictions of ID Blocks are fused as the final result.

to predict edges at all scales and ignore that, one specific intermediate layer could be suited to delineate edges at certain scales. To obtain more appropriate supervision for intermediate predictions, Liu *et al.* [19] propose to relax the supervisions on intermediate layers using Canny [6] detectors with different scales. However, it is hard to decide layer-specific scales through human intervention.

Aiming to exploit multi-scale cues with a shallow CNN, we introduce a Scale Enhancement Module (SEM) which consists of multiple parallel convolutions with different dilation rates. As shown in image segmentation [20], dilated convolution effectively increases the size of receptive fields of network neurons. By involving multiple dilated convolutions, SEM captures multi-scale spatial contexts. Compared with previous strategies, *i.e.*, introducing deeper networks and explicitly fusing multiple edge detections, SEM does not significantly increase network parameters and avoids the repetitive edge detection on image pyramids.

To address the second issue, each layer in CNN shall be trained by a proper layer-specific supervision, *e.g.*, the shallow layers are trained to focus on meaningful details and deep layers should depict object-level boundaries. We propose a Bi-Directional Cascade Network (BDCN) architecture to generate the layer-specific supervision. For each layer in BDCN, its layer-specific supervision is inferred by a bi-directional cascade structure, which propagates the outputs from its adjacent higher and lower layers, as shown in Fig. 2. In another word, each layer in BDCN predicts edges in an incremental way *w.r.t* scale. We hence call the basic block in BDCN as the Incremental Detection Block (ID Block), which is constructed by inserting the SEM into convolution blocks of existing CNNs. This bi-directional cascade structure enforces each layer to focus on a specific scale, allowing for a more effective training procedure.

By combining SEM and BDCN, our method achieves competitive performance on three widely used datasets, *i.e.*, *BSDS500*, *NYUDv2*, and *Multicue*. Using VGG16 as the backbone, it achieves ODS F-measure of 0.828, 1.3% higher than current state-of-the art CED [12] on *BSDS500*. Only using the trainval data of *BSDS500* for training, it still achieves ODS F-measure of 0.806 and outperforms the human perception, *i.e.*, 0.803. To our best knowledge, we are the first that outperforms human perception by training only on trainval data of *BSDS500*. Using deeper network as the backbone, our method achieves better performance, *e.g.*, it boosts the ODS F-measure from 0.828 to 0.832 on the *BSDS500* with the ResNet50 backbone.

Compared with existing methods relying on deeper models, we also achieve a better trade-off between model compactness and accuracy. With a shallow CNN structure, we obtain comparable performance with some well-known methods [9], [14], [15]. For example, we outperform the well-known HED [16] using 1/6 of its parameters. Besides the above experiments, we further applied our edge detection results to three vision tasks, *i.e.*, optical flow estimation, semantic segmentation, and object proposal generation. Combined with existing algorithms like Epicflow [5], BNF [21], and MCG [22], our method further boosts the performance of those vision tasks.

Contributions of this work can be summarized in to following aspects.

- The bi-directional cascade architecture is proposed to automatically infer layer-specific supervision. With this strategy, an individual layer is supervised by labeled edges at its specific scale, which is more rational than applying the same supervision to different layers.
- We propose an efficient scale enhancement module, which enriches the multi-scale representations learned by shallow CNNs.
- Based on the BDCN and SEM, our method achieves competitive performance with a lightweight CNN design. This differs with previous works, which tend to introduce deeper and heavier CNNs to achieve better performance.

The reminder of this paper is organized as follows. Section 2 reviews related works on edge detection, multi-scale representation learning, network architecture design, and the application of edge detection in vision tasks. Section 3 and Section 4 present our formulation and solution to BDCN. Section 5 describes and analyzes experimental results, followed by the conclusions in Section 6.

2 Related Work

This work is related to edge detection, multi-scale representation learning, network cascade structure design, as well as edge based vision tasks. We briefly review these related works, respectively.

Edge Detection: Most of edge detection methods can be categorized into three groups, *i.e.*, traditional edge operators, learning based methods, and the recent CNN-based methods, respectively. Traditional edge operators [6], [7], [23], [24] detect edges by finding sudden changes in intensity, color, texture, *etc*. Since low-level cues are hard to capture semantic information and are noise-sensitive, researchers turn to design new methods by involving sophisticated learning paradigms. For example, Dollár *et al.* [8] propose the structured edge which jointly learns the clustering of groundtruth edges and the mapping of image patch to clustered token. The first two groups of methods stem from the

hand-crafted features which are not discriminative enough to capture edges at different scales.

With the strong learning capability of CNNs, a lot of CNN-based algorithms have been proposed in recent years. Ganin *et al.* [25] propose the N4-Fields to combine CNNs with the nearest neighbor search for edge detection. Bertasius *et al.* [14] employ CNNs to generate features for candidate contour points. Xie *et al.* [16] propose an end-to-end detection model that leverages the outputs from different intermediate layers with lateral connections. Liu *et al.* [19] propose the Relaxing Deep Supervision (RDS), which applies different supervisions to different layers by ignoring edge pixels detected by canny detectors at different scales. Liu *et al.* [10] fuse features derived from all convolutional layers to enrich the learned deep representations. Xu *et al.* [26] introduce a hierarchical deep model to extract multi-scale features and use a gated conditional random field to fuse them. Zheng *et al.* [27] propose to use the generative adversarial network with a differential evolution mode to produce edges. Some recent works also use CNN for semantic edge detection. For instance, Yu *et al.* [28] propose the CASENet to fuse multi-scale cues learned by different CNN layers for semantic edge detection. CASENet fuses several side output features with the shared concatenation to generate the final prediction.

Multi-Scale Representation Learning: Extraction and fusion of multi-scale features are fundamental and critical for many vision tasks, *e.g.*, [29], [30], [31]. Multi-scale representations can be constructed from multiple re-scaled images [32], [33], [34], *i.e.*, an image pyramid, either by computing features independently at each scale [32] or using the output from one scale as the input to the next scale [33], [34]. Recently, innovative works DeepLab [20] and PSPNet [35] use dilated convolutions and pooling to achieve multi-scale feature learning in image segmentation. Chen *et al.* [31] propose an attention mechanism to softly weight the multi-scale features at each pixel location.

Like other image patterns, edges vary dramatically in scales. Ren *et al.* [36] show that considering multi-scale cues does improve the performance of edge detection. Multiple scale cues are also used in many approaches [10], [18], [24], [26], [36], [37], [38]. Most of those approaches explore the scale-space of edges, *e.g.*, using Gaussian smoothing at multiple scales [37] or extracting features from different scaled images [1]. Recent deep network based methods employ image pyramid and hierarchical features. For example, Liu *et al.* [10] forward multiple re-scaled images to a CNN independently, then fuse their outputs.

Network Cascade: Network cascade [39], [40], [41], [42], [43] is an effective scheme for many vision applications like classification [40], detection [41], pose estimation [42] and semantic segmentation [43]. For example, Murthy *et al.* [40] treat easy and hard samples with different networks to improve the classification accuracy. Yuan *et al.* [44] ensemble a set of models with different complexities to process samples with different difficulties. Li*et al.* [43] propose to classify easy regions in a shallow network and train deeper networks to deal with hard regions. Lin *et al.* [45] propose a top-down architecture with lateral connections to propagate deep semantic features to shallow layers. They use the features of deep layers to enhance the shallow layers and build a Feature Pyramid Network (FPN) to capture different scales of objects. Recently, FPN has been adopted in many other vision tasks, such as instance/semantic segmentation. Another related work is the CASENet [28], which also enhances the feature with multiple CNN outputs. Different from existing works, our cascade network does not fuse the outputs from different layers with simple feature concatenation. The designed bi-directional cascade structure propagates outputs from other layers to infer the layer-specific training label. This new approach encourages the learning of multi-scale representations in different layers and detects edges that are well delineated by their scales. As shown in our experiments, this design brings substantial performance gains to edge detection.

Edges for Vision Tasks: As a fundamental vision task, edge detection commonly serves as a key component in many mid- and high-level vision tasks. For example, because edges provide a sparse yet informative image representation, Zitnick *et al.* [46] propose EdgeBox to generate object bounding box proposals using edges. As edges indicate boundaries of different objects, Gong *et al.* [47] propose PGN to refine human parsing based on edge detection result. Maninis *et al.* [48] exploit edge orientation map by CNN and employ it to assist higher level vision tasks including object proposal generation, image segmentation, object detection, *etc*. Bertasius *et al.* [21] propose BNF to optimize the initial semantic segmentation by referring to edges in images. Revaud *et al.* [5] propose EpicFlow, which leverages edges to interpolate matches between adjacent video frames to improve the optical flow computation. Arbeláez *et al.* [22] propose a unified approach for bottom-up segmentation and object candidate generation, which is based on edges detected from images.

Our approach also leverages CNN for edge detection. It differs with previous CNN based methods in that, it builds SEM to learn multi-scale representations in an efficient way, which avoids repetitive computation on multiple input images. Different from previous network cascade design, BDCN is a bi-directional pseudo-cascade structure, which allows for an innovative way to supervise each layer individually for layer-specific edge detection. To our best knowledge, this is an early and original attempt to adopt a cascade architecture in edge detection. We also applied our edge detection results to three vision tasks. As shown in our experiments, our edge detection algorithm further boosts the performance of those tasks.

3 Problem Formulation

Edge detection is to obtain a probabilistic edge map P from image X to approximate the groundtruth edge map Y. Let (X, Y) denote one sample in the training set $\mathbb{T}$, where $X = \{x_j, j = 1, \cdots, |X|\}$ is a raw input image and $Y = \{y_j, j = 1, \cdots, |Y|\}, y_j \in \{0, 1\}$ is the corresponding groundtruth edge map. Considering the scale of edges may vary considerably in one image, we aim to train an edge detector $\mathrm{D}(\cdot)$ capable to detect edges at different scales. Note that, we assume the scale of edges is in proportion to the size of their depicted objects.

Because pooling layer effectively enlarges the receptive field in adjacent convolutional layers, they progressively

depict image patterns at larger scales. A natural way to implement $\mathrm{D}(\cdot)$ is thus to design a deep CNN. Specifically, we can build a CNN with S convolutional layers, where each layer has a side-output to detect edges at a specific scale. The final result could be acquired by fusing those intermediate detections to cover edges at different scales.

Let P denote the final fusion result and P_s denote the intermediate detection computed by side-output of layer s, $1 \leq s \leq S$. This CNN can be trained with S supervisions on the side-outputs and one uniform supervision on the fused result. To make different layers depict edges with different scales, each side-output is expected to be trained with a layer-specific groundtruth edge map. For instance, groundtruth edge maps for low layer should contain small-scale edges and those for higher layers should contain edges with larger scales. We denote the training loss of this network as,

$$\mathcal{L}_{CNN} = \sum_{s=1}^{S} w_{side}\mathcal{L}(P_s, Y_s) + w_{fuse}\mathcal{L}(P, Y), \tag{1}$$

where $\mathcal{L}(\cdot)$ is the loss function computed with the predicted edge map and groundtruth edge map. w_{side} and w_{fuse} are fusion weights for detected edges by layer s and the fused result, respectively. Y_s is the layer-specific groundtruth edge map for the layer s.

Recent CNN based edge detectors [10], [16] train side-outputs from different layers with the same supervision, *i.e.*, the original groundtruth Y. This strategy enforces each layer to depict edges at all scales. We acquire labeled edges at different scales to chase a more reasonable training solution. Edges in Y can be decomposed into S binary edge maps according to the scale of their depicted objects. We conceptually denote the decomposition as,

$$Y = \sum_{s=1}^{S} Y_s, \tag{2}$$

where the subscript s denotes the scale, and Y_s contains annotated edges corresponding to the scale s.

It is not easy to decompose the groundtruth edge map Y manually into different scales, making it hard to compute the layer-specific supervision Y_s with Eq. (2). A possible solution is to employ an approximation schema to obtain Y_s. For example, Y_s can be inferred based on ground truth label Y and edges predicted at other layers, *i.e.*,

$$Y_s \sim Y - \sum_{i \neq s} P_i. \tag{3}$$

Eq. (3) is easy to compute, but is not an appropriate approximation to Y_s. The following discussions briefly explain the reason.

According to Eq. (3), predicted edge map P_s at layer s is trained to approximate Y_s, *i.e.*, $P_s \sim Y - \sum_{i \neq s} P_i$. In other words, we can pass the other layers' predictions to layer s for training, resulting in an equivalent formulation, *i.e.*, $Y \sim \sum_i P_i$. The training loss $\mathcal{L}_s$ on the side-output of layer s can be denoted as

$$\mathcal{L}_s = \mathcal{L}(\hat{Y}, Y), \hat{Y} = \sum_i P_i. \tag{4}$$

The gradient *w.r.t* the prediction P_s of layer s is

$$\frac{\partial \mathcal{L}_s}{\partial P_s} = \frac{\partial \mathcal{L}(\hat{Y}, Y)}{\partial P_s} = \frac{\partial \mathcal{L}(\hat{Y}, Y)}{\partial \hat{Y}} \cdot \frac{\partial \hat{Y}}{\partial P_s}. \tag{5}$$

According to Eq. (5), for edge predictions P_s, P_i at any two layers s and i with $s \neq i$, their loss gradients are equal because $\partial \hat{Y}/\partial P_s = \partial \hat{Y}/\partial P_i = 1$. This implies that, with Y_s computed in Eq. (3), the training process does not necessarily differentiate the scales depicted by different layers, making it not appropriate for our layer-specific scale learning task.

To address the above issue, we approximate Y_s with two complementary supervisions. One ignores the edges with scales smaller than s, and the other ignores the edges with larger scales. We define those two supervisions at layer s as

$$\begin{aligned} Y_s^{s2d} &= Y - \sum_{i<s} P_i^{s2d}, \\ Y_s^{d2s} &= Y - \sum_{i>s} P_i^{d2s}, \end{aligned} \tag{6}$$

where the superscript $s2d$ denotes information propagation from shallow layers to deeper layers, and $d2s$ denotes the prorogation from deep layers to shallower layers.

For layer s, the predicted edges ${P_s}^{s2d}$ and ${P_s}^{d2s}$ approximate to ${Y_s}^{s2d}$ and ${Y_s}^{d2s}$, respectively. Their combination is a reasonable approximation to Y_s, *i.e.*,

$${P_s}^{s2d} + {P_s}^{d2s} \sim 2Y - \sum_{i<s} {P_i}^{s2d} - \sum_{i>s} {P_i}^{d2s}, \tag{7}$$

where the edges predicted at scales $i \neq s$ are depressed. Therefore, we use ${P_s}^{s2d} + {P_s}^{d2s}$ to interpolate the edge prediction at scale s.

The corresponding training loss for the side-output of layer s can be denoted as,

$$\mathcal{L}_s = \mathcal{L}\left(P_s^{d2s}, Y - \sum_{i>s} {P_i}^{d2s}\right) + \mathcal{L}\left(P_s^{s2d}, Y - \sum_{i<s} {P_i}^{s2d}\right) \tag{8}$$

$$\doteq \mathcal{L}\left(Y, \sum_{i \geq s} {P_i}^{d2s}\right) + \mathcal{L}\left(Y, \sum_{i \leq s} {P_i}^{s2d}\right). \tag{9}$$

The gradient *w.r.t* the predictions of layer s can be denoted as,

$$\frac{\partial \mathcal{L}\left(Y, \sum\limits_{i \geq s} {P_i}^{d2s}\right)}{\partial {P_s}^{d2s}} + \frac{\partial \mathcal{L}\left(Y, \sum\limits_{i \leq s} {P_i}^{s2d}\right)}{\partial {P_s}^{s2d}}. \tag{10}$$

It is easy to infer that, the loss gradient in Eq. (10) differs *w.r.t* layer ID s to allow for the layer-specific network training. After training, the final edge prediction P can be computed by fusing intermediate predictions $P_s^{s2d}, P_s^{d2s}, s = 1...S$ with a 1x1 convolutional layer. In the following section, we proceed to introduce the design of our network architecture, which can be trained with Eq. (8).

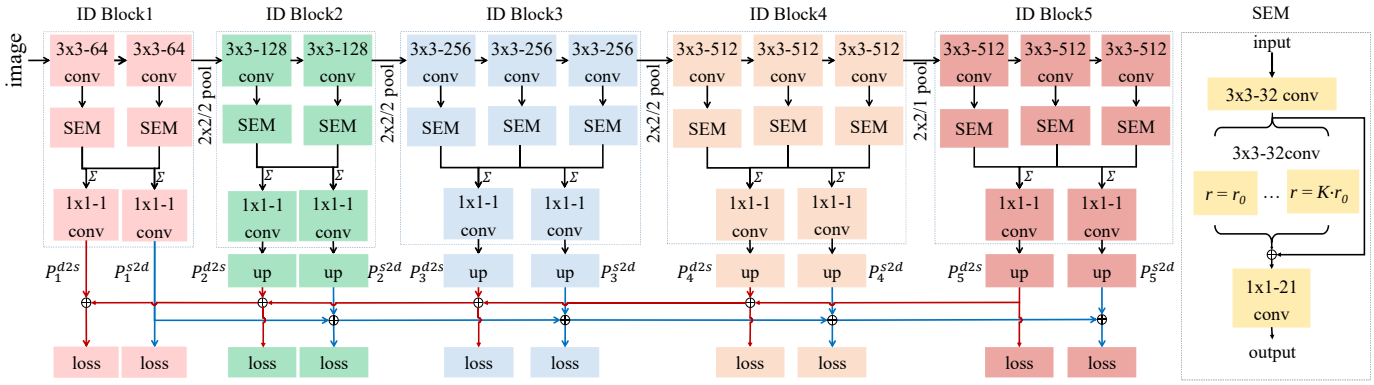


Fig. 3. The detailed architecture of BDCN and SEM. The illustrated BDCN is modified based on VGG16 [49]. The number of ID Blocks in BDCN can be flexibly set from 2 to 5 (see Fig. 6) to achieve a better trade-off between efficiency and accuracy.

4 Proposed Methods

Network Architecture: Eq. (6) passes predictions from other layers to layer s to infer the layer-specific supervision. Based on this intuition, we propose a Bi-Directional Cascade Network (BDCN) architecture to implement the layer-specific training. The overall architecture is shown in Fig. 2, where our network is composed of multiple ID Blocks, each produces two side-outputs trained with different supervisions. Supervisions for each layer are inferred based on the groundtruth edge map and edge predictions from its higher and lower layers, respectively.

We illustrate the detailed architecture of BDCN in Fig. 3. Specifically, the network in Fig. 3 is constructed based on the VGG16 [49] by removing its three fully connected layers and the last pooling layer. The 13 convolutional layers in VGG16 are divided into 5 blocks, each follows a pooling layer to progressively enlarge the receptive fields in the next block. The VGG blocks evolve into ID Blocks by inserting several Scale Enhancement Modules (SEMs).

ID Block is the basic component of our network. Each ID block produces two edge predictions. As shown in Fig. 3, an ID Block consists of several convolutional layers, each is followed by a SEM. The outputs of multiple SEMs are fused and fed into two 1×1 convolutional layers to generate two edges predictions P^{d2s} and P^{s2d}, respectively. The bi-directional cascade structure in Fig. 3 propagates edge predictions in two directions. For the s-th block, P_s^{s2d} is trained with supervision Y_s^{s2d} computed in Eq. (6). P_s^{d2s} is trained in a similar way. The final edge prediction is computed by fusing those intermediate edge predictions in a fusion layer using 1×1 convolution.

Scale Enhancement Module: Since different convolutional layers depict different scales, the depth of a neural network determines the range of scales it could model. A shallow network may not be capable to detect edges at different scales. However, a large number of convolutional layers involves too many parameters and makes the training difficult. To achieve a better trade-off between efficiency and accuracy, we enhance the multi-scale representations learned in each convolutional layer with the SEM.

SEM is inserted into each ID Block to enrich its multi-scale representations. SEM is inspired by the dilated convolution proposed by Chen *et al.* [20] for image segmentation. For an input feature map $\mathbf{x} \in \mathcal{R}^{H\times W}$ with a convolution filter $\mathbf{w} \in \mathcal{R}^{h\times w}$, the output $\mathbf{y} \in \mathcal{R}^{H'\times W'}$ of dilated convolution at location (i, j) is computed by

$$\mathbf{y}_{ij} = \sum_{m,n}^{h,w} \mathbf{x}_{[i+r\cdot m, j+r\cdot n]} \cdot \mathbf{w}_{[m,n]}, \tag{11}$$

where r is the dilation rate, indicating the stride for sampling input feature map. Standard convolution can be treated as a special case with $r = 1$. Eq. (11) shows that dilated convolution enlarges the receptive field of neurons without reducing the resolution of feature maps or increasing the parameters.

As shown on the right side of Fig. 3, for each SEM we apply K dilated convolutions with different dilation rates. For the k-th dilated convolution, we set its dilation rate as $r_k = max(1, r_0 \times k)$, which involves two parameters in SEM: the dilation rate factor r_0 and the number of convolution layers K. They are evaluated in Sec. 5.3.

Network Training: Each ID Block in our network is trained with two side supervisions. Besides that, we fuse the intermediate edge predictions with a fusion layer as the final result. Therefore, BDCN is trained with two types of loss. We formulate the overall loss $\mathcal{L}_{BDCN}$ as,

$$\mathcal{L}_{BDCN} = \sum_{s=1}^{S} w_{side} \cdot \mathcal{L}_s + w_{fuse} \cdot \mathcal{L}(P, Y), \tag{12}$$

where $\mathcal{L}_s$ denotes the loss on side-outputs computed with Eq. (8) and P denotes the final edge prediction.

The function $\mathcal{L}(\cdot)$ is computed at each pixel with respect to its edge annotation. Because the distribution of edge/non-edge pixels is heavily biased, we employ a class-balanced cross-entropy loss as $\mathcal{L}(\cdot)$. Because of the inconsistency of edge annotations among different annotators, we also introduce a threshold γ for loss computation. For a groudtruth $Y = (y_j, j = 1, ..., |Y|), y_j \in (0, 1)$, we define $Y_+ = \{y_j, y_j > \gamma\}$ and $Y_- = \{y_j, y_j = 0\}$. Only pixels corresponding to Y_+ and Y_- are considered in loss computation. We hence define $\mathcal{L}(\cdot)$ as

$$\mathcal{L}(P, Y) = -\alpha \sum_{j\in Y_-} log(1 - p_j) - \beta \sum_{j\in Y_+} log(p_j), \tag{13}$$

where $P = (p_j, j = 1, ..., |P|), p_j \in (0, 1)$ denotes a predicted edge map, $\alpha = \lambda \cdot |Y_+|/(|Y_+| + |Y_-|), \beta =$

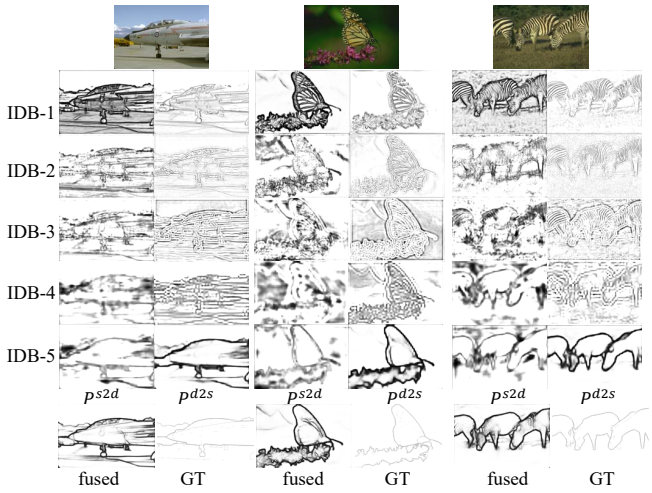


Fig. 4. Examples of edges detected by different ID Blocks (IDB for short). Each ID Block generates two edge predictions, P^{s2d} and P^{d2s}, respectively."fused" denotes the final prediction and "GT" denotes the groundtruth edge map.

$|Y_-|/(|Y_+| + |Y_-|)$ balance the edge/non-edge pixels. λ controls the weight of positive over negative samples.

Fig. 4 shows edges detected by different ID blocks. We observe that, edges detected by different ID Blocks correspond to different scales. The shallow ID Blocks produce strong responses on local details and deeper ID Blocks are more sensitive to edges at larger scale. For instance, detailed edges on the body of zebra and butterfly can be detected by shallow ID Block, but are depressed by deeper ID Block. This figure also shows that, it is reasonable to fuse layer-wise predictions to generate the final edge prediction. For instance, some detailed edges in the groundtruth edge map can not be captured by the IDB-5 layer, but can be detected by shallow layers. The following section tests the validity of BDCN and SEM.

5 Experiment

5.1 Datasets

We evaluate the proposed approach on the three public datasets: *BSDS500* [1], *NYUDv2* [50], and *Multicue* [51] for edge detection. We also conduct experiments on Pascal Context [52] for image segmentation, on Pascal VOC2012 validation set (VOC2012 val) for object proposal generation, and on MPI-Sintel [53] for optical flow estimation to further demonstrate the validity of BDCN.

BSDS500 contains 200 images for training, 100 images for validation, and 200 images for testing. Each image is manually annotated by multiple annotators. The final groundtruth is the averaged annotations by the annotators. We also utilize the strategies in [10], [12], [16] to augment training and validation sets by randomly flipping, scaling and rotating images. Following those works, we also adopt the PASCAL Context dataset [52] as our training set.

NYUDv2 consists of 1449 pairs of aligned RGB and depth images. It is split into 381 training, 414 validation, and 654 testing images. *NYUDv2* is initially used for scene understanding, hence is also used for edge detection in previous works [10], [16], [54], [55]. Following those works, we augment the training set by randomly flipping, scaling, and rotating training images.

Multicue [51] contains 100 challenging natural scenes. Each scene has two frame sequences taken from left and right view, respectively. The last frame of left-view sequence is annotated with edges and boundaries. Following [10], [18], [51], we randomly split 100 annotated frames into 80 and 20 images for training and testing, respectively. We also augment the training data with the same way in [16].

Pascal VOC2012 [56] includes 4369 labeled images, including 1464 images for training, 1449 for validation and 1456 for testing. Following the settings in previous works [18], [57], [58], we also use the Pascal VOC2012 validation set to test the performance of object proposal generation. The Average Recall (AR) with respect to the number of proposals is reported and compared.

Pascal Context [52] is constructed based on PASCAL VOC2010 for image segmentation. It includes 10103 images and is split into 4998 for training and 5015 for validation. Following setting in previous works [18], [57], we evaluate on the most frequent 59 classes and background, *i.e.*, 60 classes are considered. We report the Pixel Accuracy (PA), Mean Pixel Accuracy (MPA), Mean Intersection Over Union (Mean IOU) as evaluation metrics for fair comparison.

MPI Sintel [53] has 23 and 12 naturalistic video sequences for training and testing optical flow estimation algorithms, respectively. Following the settings in previous work [12], we also use the final version of this dataset with photo-realistic rendering and report Average Endpoint Error (AEE) on the training set.

5.2 Implementation Details

We implement our network using PyTorch. The VGG16 [49] pretrained on ImageNet [59] is used to initialize the backbone. The threshold γ used for loss computation is set as 0.3 for *BSDS500*. γ is set as 0.3 and 0.4 for *Multicue boundary* and *edges* datasets, respectively. *NYUDv2* provides binary annotations, thus does not need to set γ for loss computation. Following [10], we set the parameter λ as 1.1 for *BSDS500* and *Multicue*, set λ as 1.2 for *NYUDv2*. A deeper ResNet50 [17] pretrained on ImageNet [59] is also used as the backbone of BDCN. We treat the 5 stages of ResNet50 as 5 convolution blocks and evolve them to ID Block with similar way for VGG16. [1]

SGD optimizer is adopted to train our network. On *BSDS500* and *NYUDv2*, we set the batch size to 10 for all the experiments. The initial learning rate, momentum and weight decay are set to $1e$-6, 0.9, and 2e-4 respectively. The learning rate decreases by 10 times after every 10k iterations. We train 40k iterations for *BSDS500* and 8k for *NYUDv2*, 2k and 4k iterations for *Multicue boundary* and *edge*, respectively. w_{side} and w_{fuse} are set as 0.5, and 1.1, respectively. Since *Multicue* dataset includes high resolution images, we randomly crop 500×500 patches from each image in training. All the experiments are conducted on a NVIDIA GeForce1080Ti GPU with 11GB memory.

1. The code and pre-trained BCDN models based on VGG16 and ResNet50 can be downloaded from: https://drive.google.com/file/d/1CmDMypSlLM6EAvOt5yjwUQ7O5w-xCm1n/view?usp=sharing

TABLE 1
Impact of SEM parameters to the edge detection performance on *BSDS500* validation set. (a) shows the impact of K with r_0=4. (b) shows the impact of r_0 with K=3.

(a)

K	ODS	OIS	AP
0	.7728	.7881	.8093
1	.7733	.7845	.8139
2	.7738	.7876	.8169
3	**.7748**	**.7894**	**.8170**
4	.7745	.7896	.8166

(b)

r_0	rate	ODS	OIS	AP
0	1,1,1	.7720	.7881	.8116
1	1,2,3	.7721	.7882	.8124
2	2,4,6	.7725	.7875	.8132
4	4,8,12	**.7748**	**.7894**	**.8170**
8	8,16,24	.7742	.7889	.8169

TABLE 2
Validity of components in BDCN on *BSDS500* validation set with the VGG16 backbone. (a) tests different cascade architectures. (b) shows the validity of SEM and the bi-directional cascade architecture.

(a)

Architecture	ODS	OIS	AP
baseline	.7681	.7751	.7912
S2D	.7683	.7802	.7978
D2S	.7710	.7816	**.8049**
S2D+D2S (BDCN w/o SEM)	**.7762**	**.7872**	.8013

(b)

Method	ODS	OIS	AP
baseline	.7681	.7751	.7912
SEM	.7748	**.7894**	**.8170**
S2D+D2S (BDCN w/o SEM)	.7762	.7872	.8013
BDCN	**.7765**	.7882	.8091

TABLE 3
Validity of components in BDCN on *BSDS500* validation set with the ResNet50 backbone. (a) tests different cascade architectures. (b) shows the validity of SEM and the bi-directional cascade architecture.

(a)

Architecture	ODS	OIS	AP
baseline	.7703	.7806	.8012
S2D	.7710	.7819	.8021
D2S	.7715	.7827	.8032
S2D+D2S (BDCN w/o SEM)	**.7782**	**.7913**	**.8045**

(b)

Method	ODS	OIS	AP
baseline	.7703	.7806	.8012
SEM	.7761	7907	**.8201**
S2D+D2S (BDCN w/o SEM)	.7782	.7913	.8065
BDCN	**.7793**	**.7927**	.8124

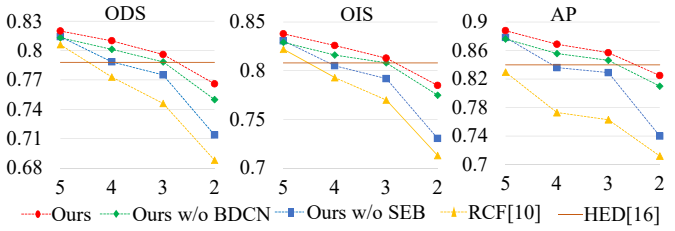


Fig. 5. Comparison of edge detection accuracy as we decrease the number of ID Blocks from 5 to 2. HED learned with VGG16 is denoted as the solid line for comparison.

We follow previous works [10], [12], [16], [26], and perform standard Non-Maximum Suppression (NMS) to produce the final edge maps. For a fair comparison with other work, we report our edge detection performance with commonly used evaluation metrics, including Average Precision (AP), as well as F-measure at both Optimal Dataset Scale (ODS) and Optimal Image Scale (OIS). The maximum tolerance allowed for correct matches between edge predictions and groundtruth annotations is set to 0.0075 for *BSDS500* and *Multicue* dataset, and is set to 0.011 for *NYUDv2* dataset as in previous works [10], [18], [51].

5.3 Ablation Study

In this section, we conduct experiments on *BSDS500* to study the impact of parameters and verify each component in our network.

Validity of SEM: We train the network on the *BSDS500* training set and evaluate on the validation set. Firstly, we test the impact of the parameters in SEM, *i.e.*, the number of dilated convolutions K and the dilation rate factor r_0. Experimental results are summarized in Table 1.

Table 1 (a) shows the impact of K with r_0=4. Note that, K=0 means directly copying the input as output. The results demonstrate that setting K larger than 1 substantially improves the performance. However, too large K does not constantly boost the performance. The reason might be that, large K produces high dimensional outputs and makes edge extraction from such high dimensional data difficult. Table 1 (b) also shows that larger r_0 improves the performance. But the performance starts to drop with too large r_0, *e.g.*, r_0=8. In our following experiments, we fix K=3 and r_0=4.

Validity of BDCN architecture: Table 2 (a) shows the comparison among different cascade architectures, *i.e.*, single direction cascade from shallow to deep layers (S2D), from deep to shallow layers (D2S), and the bi-directional cascade (S2D+D2S), *i.e.*, the BDCN w/o SEM. Note that, we use the VGG16 network with 5 side-outputs, *i.e.*, the HED [16] as the baseline. It can be observed that, both S2D and D2S structures outperform the baseline. This shows the validity of the cascade structure in network training. The combination of these two cascade structures, *i.e.*, S2D+D2S, results in the best performance. We further test the performance of combining SEM and S2D+D2S and summarize the results in Table 2 (b), which shows that SEM and bi-directional cascade structure consistently improve the performance of baseline, *e.g.*, improve the ODS F-measure by 0.7% and 0.8% respectively. Combining SEM and S2D+D2S results in the best performance. Different from ODS, OIS and AP are computed with the best threshold on each individual image. Therefore, ODS is commonly regarded as a more important metric for edge detection. It is also commonly observed that better ODS may harm OIS and AP [11], [60].

We further repeat the above experiments using the ResNet50 [17] with 5 side-outputs as the baseline and summarize experimental results in Table 3. It can be observed that, deeper network boosts the baseline performance, *e.g.*, from 76.81% to 77.03% in ODS F-measure. It is also clear that, our method works with strong baseline and consistently improves the baseline performance, *e.g.*, SEM and the bi-directional cascade structure improve the baseline ODS F-measure by 0.8% and 0.9% respectively. We can conclude that, each component introduced in our method is valid in boosting the edge detection performance.

Performance with shallow network: We proceed to test the capability of our method in learning multi-scale representations with shallow networks. We test our approach and RCF with different depth of networks, *i.e.*, using different numbers of convolutional block to construct the edge detection model. Fig. 5 presents results on the *BSDS500* testset. As shown in Fig. 5, the performance of RCF [10] drops more substantially than our method as we decrease the depth of

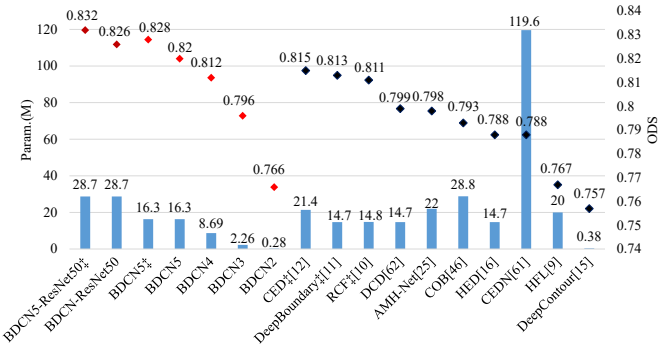


Fig. 6. Comparison of parameters and performance with other methods. The number behind "BDCN" indicates the number of ID Block. ‡means the multiscale results.

networks. This verifies that our approach is more effective in detecting edges with shallow networks. We also show the performance of our approach without the SEM and the BDCN structure. These ablations show that removing either BDCN or SEM degrades the performance. It is also interesting to observe that, without SEM, the performance of our method drops substantially. This hence verifies the importance of SEM to multi-scale representation learning in shallow networks.

Fig. 6 further shows the comparison of parameters *vs*. performance of our method with other deep net based methods on *BSDS500*. With 5 convolutional blocks in VGG16, HED [16], RCF [10], and our method use similar number of parameters, *i.e.*, about 16M. As we decrease the number of ID Blocks from 5 to 2, our number of parameters decreases dramatically, drops to 8.69M, 2.26M, and 0.28M, respectively. Our method still achieves F-measure ODS of 0.766 using only two ID Blocks with 0.28M parameters. It also outperforms HED and RCF with a shallower network, *i.e.*, with 3 and 4 ID Blocks respectively. For example, it outperforms HED by 0.8% with 3 ID Blocks and just 1/6 parameters of HED. We thus conclude that, our method achieves reasonably good edge detection accuracy even with a lightweight network. With 5 ID Blocks, our method runs at about 22fps for edge detection, on par with most CNN-based methods. With 4, 3 and 2 ID Blocks, it accelerates to 29 fps, 33fps, and 37fps, respectively. As shown in the Fig. 6, our work achieves better performance with a stronger backbone. ResNet50-BDCN has similar model size with COB, but achieves significantly better performance, *i.e.*, 0.832 *vs*. 0.793.

Performance of individual layers: To further show the advantage of our method, we evaluate the performance of edge predictions by different intermediate layers, and show the comparison with HED [16] and RCF [10] in Table 4. It can be observed that, the intermediate predictions of our network also consistently outperform the ones from HED and RCF, respectively. The detected edges by each intermediate layer are visualized in Fig. 4.

5.4 Comparison with Recent Works

Performance on BSDS500: We compare our approach with recent deep learning based methods including CED [12], RCF [10], DeepBoundary [11], DCD [64], COB [48], HED [16], HFL [9], DeepEdge [14] and DeepContour [15], and traditional edge detection methods, including SCG [55], PMI [61] and OEF [62]. The comparison on *BSDS500* is summarized in Table 5 and Fig. 8, respectively.

As shown in Table 5, our method obtains the F-measure ODS of 0.820 using single scale input, and achieves 0.828 with multi-scale inputs, both outperform all these competing methods. Using a single-scale input, our method still outperforms the recent CED [12] and DeepBoundary [11] that use multi-scale inputs. Our method also outperforms the human perception by 2.5% in F-measure ODS. The F-measure OIS and AP of our approach are also higher than the ones of the other methods. Using ResNet50 as the backbone, our performance is further boosted, *e.g.*, achieves F-measure ODS of 0.826 using the single scale input, and achieves 0.832 with multi-scale inputs. Fig. 7 compares sample edge detection results generated by our approach and several recent ones on *BSDS500*.

Performance on NYUDv2: *NYUDv2* has three types of inputs, *i.e.*, RGB, HHA, and RGB-HHA, respectively. Following previous works [10], [16], we perform experiments on all of them. The results of RGB-HHA are obtained by

TABLE 4
The performance (ODS) of each layer in BDCN, RCF [10], and HED [16] on *BSDS500* test set.

Layer ID.	HED [16]	RCF [10]	BDCN
1	0.595	0.595	**0.727**
2	0.697	0.710	**0.762**
3	0.750	0.766	**0.771**
4	0.748	0.761	**0.802**
5	0.637	0.758	**0.815**
fuse	0.790	0.805	**0.820**

TABLE 5
Comparison with other methods on *BSDS500* test set. †indicates trained with additional PASCAL-Context data. ‡indicates the fused result of multi-scale images.

Method	ODS	OIS	AP
Human	.803	.803	–
SCG [55]	.739	.758	.773
PMI [61]	.741	.769	.799
OEF [62]	.746	.770	.820
DeepContour [15]	.757	.776	.800
HFL [9]	.767	.788	.795
HED [16]	.788	.808	.840
CEDN [63] †	.788	.804	–
ResNet50-COB [48]	.793	.820	.859
DCD [64]	.799	.817	.849
ResNet50-AMHNet [26]	.798	.829	.869
RCF [10]	.798	.815	–
RCF [10] †	.806	.823	–
RCF [10] ‡	.811	.830	–
Deep Boundary [11]	.789	.811	.789
Deep Boundary [11] ‡	.809	.827	.861
Deep Boundary [11] ‡ + Grouping	.813	.831	.866
CED [12]	.794	.811	.847
CED [12] ‡	.815	.833	.889
LPCB [60]	.800	.806	–
LPCB [60] †	.808	.824	–
LPCB [60] ‡	.815	.834	–
BDCN	**.806**	**.826**	**.847**
BDCN †	**.820**	**.838**	**.888**
BDCN ‡	**.828**	**.844**	**.890**
ResNet50-BDCN	**.809**	**.828**	**.850**
ResNet50-BDCN †	**.826**	**.840**	**.862**
ResNet50-BDCN ‡	**.832**	**.847**	**.872**

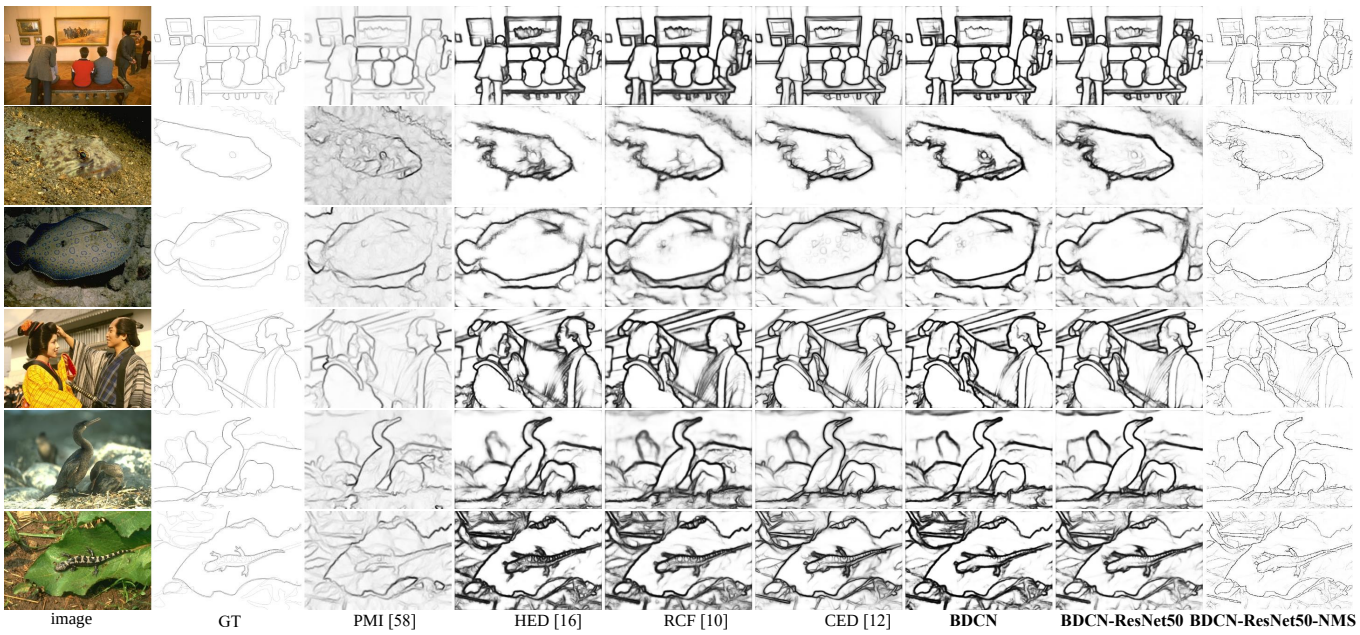


Fig. 7. Comparison of edge detection results on *BSDS500* test set. GT denotes the groundtruth edge map. All the results except for the last row are raw edge maps computed with a single scale input before Non-Maximum Suppression.

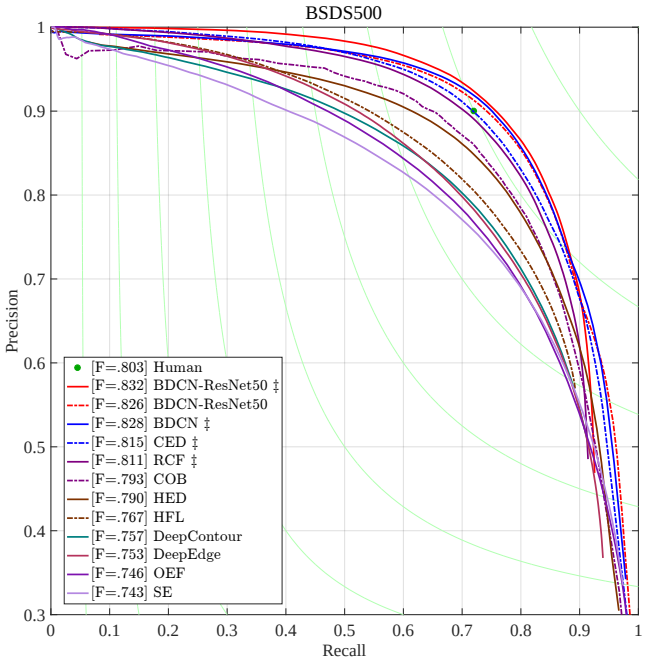


Fig. 8. The precision-recall curves of our method and other works on *BSDS500* test set.

TABLE 6
Comparison with recent works on *NYUDv2* dataset.

Method		ODS	OIS	AP
gPb-UCM [1]	RGB	.632	.661	.562
gPb+NG [54]		.687	.716	.629
OEF [62]		.651	.667	–
SE [8]		.695	.708	.679
SE+NG+ [65]		.706	.734	.738
HED [16]	RGB	.720	.734	.734
	HHA	.682	.695	.702
	RGB-HHA	.746	.761	.786
RCF [10]	RGB	.729	.742	–
	HHA	.705	.715	–
	RGB-HHA	.757	.771	–
ResNet50-RCF-RGB-HHA [66]		.781	.793	–
ResNet50-AMH [26]	RGB	.744	.758	.765
	HHA	.716	.729	.734
	RGB-HHA	.771	.786	.802
LPCB [60]	RGB	.739	.754	–
	HHA	.707	.719	–
	RGB-HHA	.762	.778	–
COB-ResNet50 [58]	RGB-HHA	.784	.805	825
BDCN	RGB	**.748**	**.763**	**.770**
	HHA	.708	.720	.733
	RGB-HHA	.767	.783	.783
ResNet50-BDCN	RGB	**.760**	**.773**	**.782**
	HHA	**.717**	**.730**	**.733**
	RGB-HHA	**.788**	**.802**	**.810**

averaging the edges detected on RGB and HHA. Table 6 shows the comparison of our method with several recent approaches, including gPb-ucm [1], OEF [62], gPb+NG [54], SE+NG+ [65], SE [8], HED [16], RCF [10] and AMH-Net [26]. Fig. 9 shows the precision-recall curves of our method and other competitors. All the evaluation results are based on the single scale input.

As shown in Table 6 and Fig. 9, our performance is competitive, *i.e.*, it outperforms most of the compared works except AMH-Net [26]. Note that, AMH-Net applies the deeper ResNet50 to construct the edge detector. With a shallower VGG16 backbone, our method still outperforms AMH-Net on the RGB image, *i.e.*, our 0.748 vs. 0.744 of AMH-Net in F-measure ODS. Compared with previous works, our improvement over existing works is actually more substantial, *e.g.*, on *NYUDv2* our gains over RCF [10] and HED [16] are 0.019 and 0.028 in ODS, higher than the 0.009 gain of RCF [10] over HED [16]. Table 6 also shows our performance achieved with the deeper ResNet50 backbone. It is clear that, with stronger backbone, BDCN achieves the best performance, *e.g.*, it achieves F-measure ODS of 0.760, substantially better than the 0.748 of VGG16 backbone and

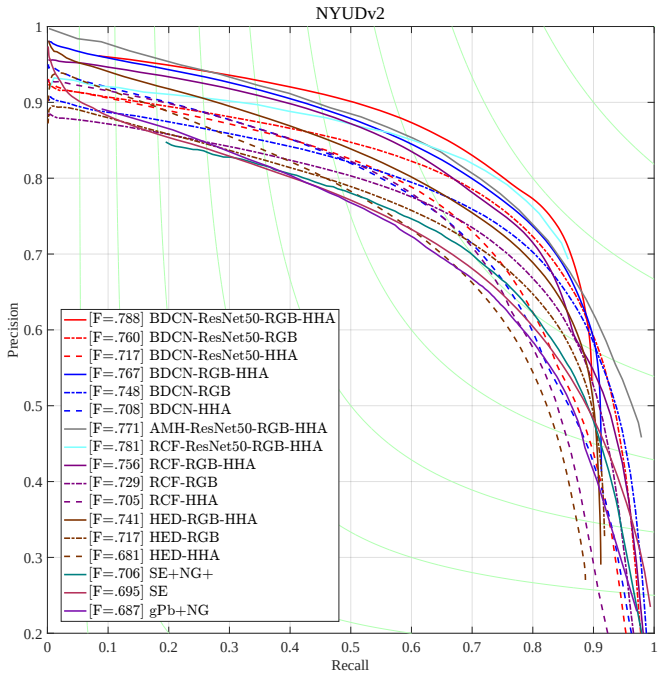


Fig. 9. The precision-recall curves of our method and other works on the *NYUDv2* dataset.

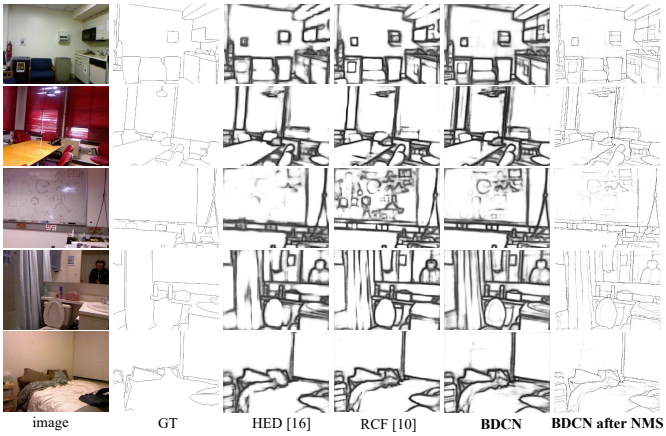


Fig. 10. Sample edge detection results obtained by BDCN and several recent approaches on *NYUDv2*.

the 0.744 of AMH-Net, which is also implemented based on the ResNet50. Fig. 10 visualizes edge detection results by BDCN and several other recent works on *NYUDv2*.

Performance on Multicue: *Multicue* consists of two sub datasets, *i.e.*, *Multicue boundary* and *Multicue edge*. As done in RCF [10] and the recent version of HED [18], we average the scores of three independent experiments as the final result. We show the comparison with recent works in Table 7, where our method achieves substantially higher performance than RCF [10] and HED [16]. For boundary detection task, we outperform RCF and HED by 1.3% and 2.4%, respectively in F-measure ODS. For edge detection task, our performance is 3.4% and 4.3% higher than the ones of RCF and HED. Moreover, the performance fluctuation of our method is considerably smaller than those two methods, which means our method delivers more stable results. Some edge detection results generated by our approach on *Multicue* are presented in Fig. 11.

TABLE 7
Comparison with recent works on *Multicue*. ‡indicates the fused result of multi-scale images.

Cat.	Method	ODS	OIS	AP
Boundary	Human [51]	.760 (0.017)	–	–
	Multicue [51]	.720 (0.014)	–	–
	HED [18]	.814 (0.011)	.822 (0.008)	.869(0.015)
	RCF [10]	.817 (0.004)	.825 (0.005)	–
	RCF [10] ‡	.825 (0.008)	.836 (0.007)	–
	BDCN	.836 (**0.001**)	.846(**0.003**)	.893(**0.001**)
	BDCN ‡	**.838**(0.004)	**.853**(0.009)	**.906**(0.005)
Edge	Human [51]	.750 (0.024)	–	–
	Multicue [51]	.830 (0.002)	–	–
	HED [18]	.851 (0.014)	.864 (0.011)	–
	RCF [10]	.857 (0.004)	.862 (0.004)	–
	RCF [10] ‡	.860 (0.005)	.864 (0.004)	–
	BDCN	.891 (**0.001**)	.898 (**0.002**)	.935(**0.002**)
	BDCN ‡	**.894**(0.002)	**.901**(0.004)	**.941**(0.005)

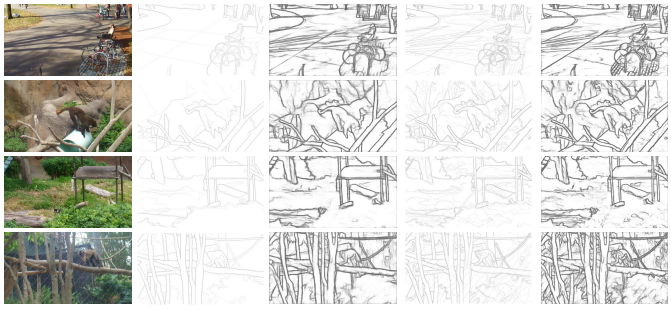


Fig. 11. Examples of our edge detection results before Non-Maximum Suppression on *Multicue* dataset.

Discussions: The above experiments show that our approach achieves promising performance. It is interesting to discuss the reason for the performance gains. The above experiments show that, both the BDCN structure and SEM are important for our performance gains. As shown in Table 5, the combination of BDCN and SEM achieves the 0.806 ODS on the *BSDS500* test set, already outperforming other well-known methods such as LPCB [60], RCF [10], *etc.*, under the same experimental setting. Table 2 shows that with 5 ID Blocks, BDCN, *i.e.*, the S2D+D2S, achieves better ODS than SEM. In Fig. 5, SEM is more important than BDCN when the network is shallow, *e.g.*, as the network becomes shallower, the performance without SEM drops more substantially than the case without BDCN. The reason is that, shallow networks lack multi-scale cues learning capability, making the SEM more important. We hence conclude that, SEM is the key to achieve the good performance with a shallow network. Combining BDCN and SEM constantly boosts the ODS performance, which is commonly regarded as a more important performance metric for edge detection. Besides proposed methods, some training tricks also boost the performance. For instance in Table 5, additional training data improves the ODS from 0.806 to 0.820 and multi-scale test further improves the ODS from 0.820 to 0.828. Stronger backbone, *i.e.*, ResNet50, boosts the ODS to 0.832.

Most of existing works on edge detection focus on s-

TABLE 8
Optimal flow estimation performance achieved with HED, CED, and BDCN by the Epicflow [5] on *Sintel* dataset.

methods	AEE	methods	AEE
baseline [5]	3.686	–	–
HED [18]	3.588	ResNet50-HED [18]	3.573
CED [12]	3.570	ResNet50-CED [12]	3.549
BDCN	**3.546**	ResNet50-BDCN	**3.515**

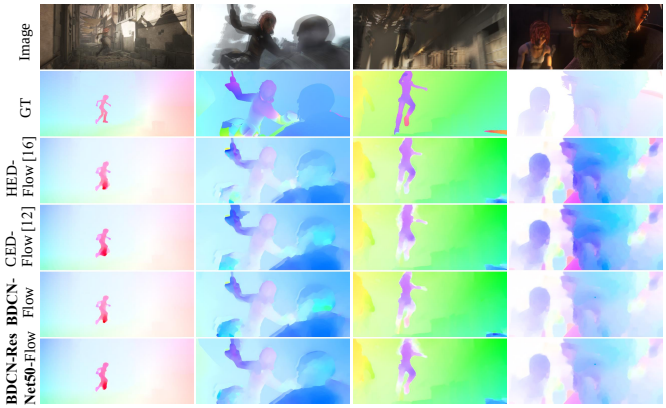


Fig. 12. Illustration of sample optical flow estimation results achieved with HED, CED, and BDCN, respectively.

tudying more reasonable network architectures and training strategies. A recent work [27] studies another way. It uses Differential Evolution and Generative Adversarial Networks (GANs) and achieves promising edge detection performance. Compared with this work, our work does not involve evolutionary computation and GAN module, thus could be more compact in size and easier to train. For the future work, it would be interesting to study strategies like evolutionary algorithm, network architecture search, and adversarial learning for edge detection.

5.5 BDCN for Other Vision Tasks

Optical flow estimation is to estimate the motion cues between consecutive video frames. As an important vision task, optical flow estimation could be sensitive to occlusions and noises. To conquer those issues, Revaud *et al.* [5] propose the EpicFlow algorithm. EpicFlow leverages detected edges to compute geodesic distance and then obtains dense matches of two neighbouring frames based on sparse matches from DeepMatching [67]. Edge detection hence plays an important role in EpicFlow algorithm. To show the advantage of our edge detection algorithm, we choose the EpicFlow as the optical flow estimation method, and apply our edge detection results to EpicFlow.

We conduct experiments on the *Sintel* dataset and compare with edge detectors HED and CED. Following the settings in [57], we train edge detectors on the *BSDS500* for fair comparison. Experimental results are summarized in Table 8. As illustrated in Table. 8, using VGG16 as the backbone, BDCN achieves better AEE performance than HED and CED. With the deeper ResNet50 baseline, BDCN further drops the AEE to 3.515, which also outperforms the ones achieved with HED and CED. Fig. 12 illustrates several optical estimation results achieved with HED, CED, and BDCN, respectively.

TABLE 9
The comparison of results for segmentation on the Pascal Context validation set.

methods	PA	MPA	mIOU
FCN-8s [68]	67.0	50.7	37.8
UoA-Context+CRF [69]	71.5	53.9	43.3
IFCN-8s [70]	74.5	57.7	45.0
RefineNet-ResNet101 [71]	–	–	47.1
DeepLab [20]	70.5	54.6	42.5
HED-BNF [16]	71.1	54.8	42.8
CED-BNF [12]	71.4	55.2	43.1
BDCN-BNF	**71.6**	**55.3**	**43.4**
ResNet50-BDCN-BNF	**71.9**	**55.6**	**43.5**
EncNet [72]	78.3	60.0	49.5
BDCN-BNF	**79.2**	**60.4**	**50.4**
ResNet50-BDCN-BNF	**79.4**	**60.6**	**50.6**

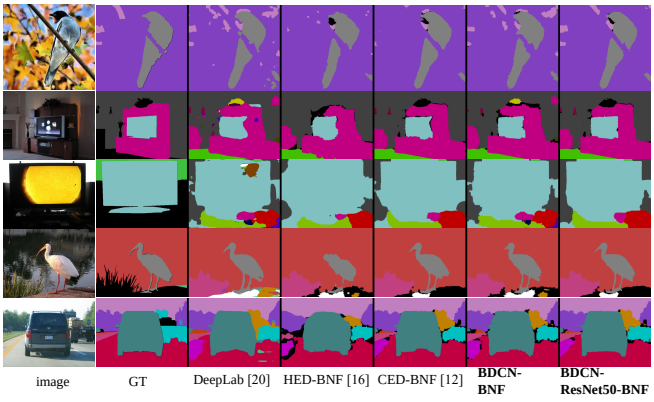


Fig. 13. Illustration of image segmentation results achieved by BNF with different edge detectors. The initial segmentation is computed by the DeepLab. Each color represents a semantic category.

Semantic segmentation aims at labeling each pixel of an image with the semantic category. In recent years, researchers have significantly boosted the performance of semantic segmentation [72], [73]. However, there still remain several challenging issues. For instance, the successive pooling layers reduce feature map resolution and cause inaccurate segmentation near object boundaries. To address this issue, Bertasius *et al.* [21] proposed Boundary Neural Field (BNF), which optimizes the initial segmentation with edge map. An accurate edge map is hence important for BNF. This experiment applies our edge detection to BNF to compute the semantic segmentation.

To make comparison with existing edge detection algorithms, we first apply DeepLab [20] to generate the initial segmentation, then apply our edge detection algorithm and BNF to refine the initial result. Experimental results are summarized in Table 9. As shown in the table, our approach boosts the performance of DeepLab, *e.g.*, improves the mIOU from 42.5% to 43.4%, which also outperforms the ones achieved with other edge detectors. We further apply our approach with a recent segmentation algorithm EncNet [72]. As shown in Table 9, our approach further boosts the performance of EncNet, *e.g.*, BDCN with ResNet50 boosts the mIOU from 49.5% to 50.6%, which also outperforms the four recent semantic segmentation algorithms compared in Table 9. Fig. 13 illustrates sample semantic segmentation results achieved by different algorithms.

Object proposal generation targets to generate object

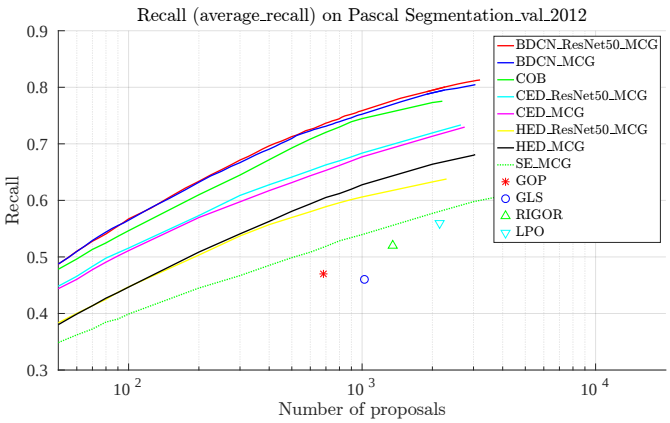


Fig. 14. Comparison with other works in object proposal generation task on Pascal VOC2012 validation set.

bounding box proposals indicating object localizations. It commonly serves as the initial step for object detection. Recently, Arbeláez *et al.* [22] propose the Multi-scale Combinatorial Grouping (MCG) algorithm for object proposal generation. Given an input image, MCG first uses an edge detector [8] to produce edge probability map, then builds hierarchical contours to generate object proposals. This experiment applies the BDCN to MCG to generate object proposals.

Following the settings in [57], [58], we report the Average Recall (AR) *w.r.t* the number of proposals in Fig. 14. Note that, all of the edge detectors are trained on the *BSDS500* dataset for fair comparison. Fig. 14 also shows the performance achieved with edge detectors COB [58], CED [57], HED [18], SE [8], as well as proposal generation algorithms GOP [74], GLS [75], RIGOR [76], and LPO [77]. As shown in the figure, BDCN-MCG achieves AR of 0.81 with ∼3000 proposals per image, substantially better than the AR of 0.75 with ∼3000 proposals per image achieved by CED-MCG. BDCN implemented with ResNet50 further boosts the AR to 0.83 with ∼3000 proposals per image. It is clear that, our BDCN generates more accurate edge detection results and effectively boosts the performance of different vision tasks.

6 Conclusions

This paper proposes a Bi-Directional Cascade Network for edge detection. By introducing a bi-directional cascade structure to enforce each layer to focus on a specific scale, BDCN trains each network layer with a layer-specific supervision. To enrich the multi-scale representations learned with a shallow network, we further introduce a Scale Enhancement Module (SEM). Our method compares favorably with about 20 edge detection methods on three datasets, achieving ODS F-measure of 0.832, 2.7% higher than current state-of-art on *BSDS500*. Our experiments also show that learning scale dedicated layers results in compact networks with a fraction of parameters, *e.g.*, our approach outperforms HED [16] with only 1/6 of its parameters. Our approach also effectively boosts the performance of three vision tasks, *i.e.*, optical flow estimation, semantic segmentation, and object proposal generation, respectively.

Acknowledgments: This work is supported in part by The National Key Research and Development Program of China under Grant No. 2018YFE0118400, in part by Beijing Natural Science Foundation under Grant No. JQ18012, in part by Natural Science Foundation of China under Grant No. 61936011, 61425025, 61620106009, 61572050, 91538111.

References

[1] P. Arbelaez, M. Maire, C. Fowlkes, and J. Malik, "Contour detection and hierarchical image segmentation," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 33(5), pp. 898–916, 2011.
[2] C. Rother, V. Kolmogorov, and A. Blake, "Grabcut: Interactive foreground extraction using iterated graph cuts," *ACM Trans. on Graph*, vol. 23, pp. 309–314, 2004.
[3] V. Ferrari, L. Fevrier, F. Jurie, and C. Schmid, "Groups of adjacent contour segments for object detection," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 30(1), pp. 36–51, 2008.
[4] J. J. Lim, C. L. Zitnick, and P. Dollár, "Sketch tokens: A learned mid-level representation for contour and object detection," in *CVPR*, 2013.
[5] J. Revaud, P. Weinzaepfel, Z. Harchaoui, and C. Schmid, "Epicflow: Edge-preserving interpolation of correspondences for optical flow," in *CVPR*, 2015.
[6] J. Canny, "A computational approach to edge detection," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 8, no. 6, pp. 679–698, Jun. 1986.
[7] J. Kittler, "On the accuracy of the sobel edge detector," *Image and Vision Computing*, vol. 1(1), pp. 37–42, 1983.
[8] P. Dollár and C. L. Zitnick, "Fast edge detection using structured forests," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 37(8), pp. 1558–1570, 2015.
[9] G. Bertasius, J. Shi, and L. Torresani, "High-for-low and low-for-high: Efficient boundary detection from deep object features and its applications to high-level vision," in *ICCV*, 2015.
[10] Y. Liu, M.-M. Cheng, X. Hu, K. Wang, and X. Bai, "Richer convolutional features for edge detection," in *CVPR*, 2017.
[11] I. Kokkinos, "Pushing the boundaries of boundary detection using deep learning," in *ICLR*, 2016.
[12] Y. Wang, X. Zhao, and K. Huang, "Deep crisp boundaries," in *CVPR*, 2017.
[13] D. Comaniciu and P. Meer, "Mean shift: A robust approach toward feature space analysis," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 24(5), pp. 603–619, 2002.
[14] G. Bertasius, J. Shi, and L. Torresani, "Deepedge: A multi-scale bifurcated deep network for top-down contour detection," in *CVPR*, 2015.
[15] W. Shen, X. Wang, Y. Wang, X. Bai, and Z. Zhang, "Deepcontour: A deep convolutional feature learned by positive-sharing loss for contour detection," in *CVPR*, 2015.
[16] S. Xie and Z. Tu, "Holistically-nested edge detection," in *ICCV*, 2015.
[17] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *CVPR*, 2016.
[18] S. Xie and Z. Tu, "Holistically-nested edge detection," *International Journal of Computer Vision*, vol. 125, no. 1, pp. 3–18, 2017.
[19] Y. Liu and M. S. Lew, "Learning relaxed deep supervision for better edge detection," in *CVPR*, 2016.
[20] L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille, "Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs," *arXiv preprint arXiv:1606.00915*, 2016.
[21] G. Bertasius, J. Shi, and L. Torresani, "Semantic segmentation with boundary neural fields," in *CVPR*, 2016.
[22] P. Arbeláez, J. Pont-Tuset, J. T. Barron, F. Marques, and J. Malik, "Multiscale combinatorial grouping," in *CVPR*, 2014.
[23] V. Torre and T. A. Poggio, "On edge detection," *IEEE Trans. Pattern Anal. Mach. Intell.*, no. 2, pp. 147–163, 1986.
[24] D. R. Martin, C. C. Fowlkes, and J. Malik, "Learning to detect natural image boundaries using local brightness, color, and texture cues," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 26(5), pp. 530–549, 2004.
[25] Y. Ganin and V. Lempitsky, "N4-fields: Neural network nearest neighbor fields for image transforms," in *ACCV*. Springer, 2014.

[26] D. Xu, W. Ouyang, X. Alameda-Pineda, E. Ricci, X. Wang, and N. Sebe, "Learning deep structured multi-scale features using attention-gated crfs for contour prediction," in *NIPS*, 2017.

[27] W. Zheng, C. Gou, L. Yan, and F.-Y. Wang, "Differential-evolution-based generative adversarial networks for edge detection," in *CVPR Workshops*, 2019.

[28] Z. Yu, C. Feng, M.-Y. Liu, and S. Ramalingam, "Casenet: Deep category-aware semantic edge detection," in *CVPR*, 2017.

[29] G. Huang, D. Chen, T. Li, F. Wu, L. van der Maaten, and K. Q. Weinberger, "Multi-scale dense convolutional networks for efficient prediction," *arXiv preprint arXiv:1703.09844*, 2017.

[30] S. Yan, J. S. Smith, W. Lu, and B. Zhang, "Hierarchical multi-scale attention networks for action recognition," *Signal Processing: Image Communication*, vol. 61, pp. 73–84, 2018.

[31] L.-C. Chen, Y. Yang, J. Wang, W. Xu, and A. L. Yuille, "Attention to scale: Scale-aware semantic image segmentation," in *CVPR*, 2016.

[32] C. Farabet, C. Couprie, L. Najman, and Y. LeCun, "Learning hierarchical features for scene labeling," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 35, no. 8, pp. 1915–1929, 2013.

[33] P. Pinheiro and R. Collobert, "Recurrent convolutional neural networks for scene labeling," in *ICML*, 2014.

[34] D. Eigen and R. Fergus, "Predicting depth, surface normals and semantic labels with a common multi-scale convolutional architecture," in *ICCV*, 2015.

[35] H. Zhao, J. Shi, X. Qi, X. Wang, and J. Jia, "Pyramid scene parsing network," in *CVPR*, 2017.

[36] X. Ren, "Multi-scale improves boundary detection in natural images," in *ECCV*, 2008.

[37] A. P. Witkin, "Scale-space filtering," *Readings in Computer Vision*, pp. 329–332, 1987.

[38] S. Konishi, A. L. Yuille, J. M. Coughlan, and S. C. Zhu, "Statistical edge detection: Learning and evaluating edge cues," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 25(1), pp. 57–74, 2003.

[39] W. Ke, J. Chen, J. Jiao, G. Zhao, and Q. Ye, "Srn: Side-output residual network for object symmetry detection in the wild," in *CVPR*, 2017.

[40] V. N. Murthy, V. Singh, T. Chen, R. Manmatha, and D. Comaniciu, "Deep decision network for multi-class image classification," in *CVPR*, 2016.

[41] H. Li, Z. Lin, X. Shen, J. Brandt, and G. Hua, "A convolutional neural network cascade for face detection," in *CVPR*, 2015.

[42] A. Toshev and C. Szegedy, "Deeppose: Human pose estimation via deep neural networks," in *CVPR*, 2014.

[43] X. Li, Z. Liu, P. Luo, C. Change Loy, and X. Tang, "Not all pixels are equal: Difficulty-aware semantic segmentation via deep layer cascade," in *CVPR*, 2017.

[44] Y. Yuan, K. Yang, and C. Zhang, "Hard-aware deeply cascaded embedding," *CoRR, abs/1611.05720*, 2016.

[45] T.-Y. Lin, P. Dollár, R. Girshick, K. He, B. Hariharan, and S. Belongie, "Feature pyramid networks for object detection," in *CVPR*, 2017.

[46] C. L. Zitnick and P. Dollár, "Edge boxes: Locating object proposals from edges," in *ECCV*, 2014.

[47] K. Gong, X. Liang, Y. Li, Y. Chen, M. Yang, and L. Lin, "Instance-level human parsing via part grouping network," in *ECCV*, 2018.

[48] K. Maninis, J. Pont-Tuset, P. Arbeláez, and L. V. Gool, "Convolutional oriented boundaries," in *ECCV*, 2016.

[49] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[50] N. Silberman, D. Hoiem, P. Kohli, and R. Fergus, "Indoor segmentation and support inference from rgbd images," in *ECCV*, 2012.

[51] D. A. Mély, J. Kim, M. McGill, Y. Guo, and T. Serre, "A systematic comparison between visual cues for boundary detection," *Vision Research*, vol. 120, pp. 93–107, 2016.

[52] R. Mottaghi, X. Chen, X. Liu, N.-G. Cho, S.-W. Lee, S. Fidler, R. Urtasun, and A. Yuille, "The role of context for object detection and semantic segmentation in the wild," in *CVPR*, 2014.

[53] D. J. Butler, J. Wulff, G. B. Stanley, and M. J. Black, "A naturalistic open source movie for optical flow evaluation," in *ECCV*, 2012.

[54] S. Gupta, P. Arbelaez, and J. Malik, "Perceptual organization and recognition of indoor scenes from rgb-d images," in *CVPR*, 2013.

[55] X. Ren and L. Bo, "Discriminatively trained sparse code gradients for contour detection," in *NIPS*, 2012.

[56] M. Everingham, L. Van Gool, C. K. Williams, J. Winn, and A. Zisserman, "The pascal visual object classes (voc) challenge," *International Journal of Computer Vision*, vol. 88, no. 2, pp. 303–338, 2010.

[57] Y. Wang, X. Zhao, Y. Li, and K. Huang, "Deep crisp boundaries: From boundaries to higher-level tasks," *IEEE Transactions on Image Processing*, vol. 28, no. 3, pp. 1285–1298, 2019.

[58] K.-K. Maninis, J. Pont-Tuset, P. Arbeláez, and L. Van Gool, "Convolutional oriented boundaries: From image segmentation to high-level tasks," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 40, no. 4, pp. 819–833, 2018.

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

[60] R. Deng, C. Shen, S. Liu, H. Wang, and X. Liu, "Learning to predict crisp boundaries," in *ECCV*, 2018.

[61] P. Isola, D. Zoran, D. Krishnan, and E. H. Adelson, "Crisp boundary detection using pointwise mutual information," in *ECCV*, 2014.

[62] S. Hallman and C. C. Fowlkes, "Oriented edge forests for boundary detection," in *CVPR*, 2015.

[63] J. Yang, B. Price, S. Cohen, H. Lee, and M.-H. Yang, "Object contour detection with a fully convolutional encoder-decoder network," in *CVPR*, 2016.

[64] Y. Liao, S. Fu, X. Lu, C. Zhang, and Z. Tang, "Deep-learning-based object-level contour detection with ccg and crf optimization," in *ICME*, 2017.

[65] S. Gupta, R. Girshick, P. Arbeláez, and J. Malik, "Learning rich features from rgb-d images for object detection and segmentation," in *ECCV*, 2014.

[66] Y. Liu, M.-M. Cheng, X. Hu, J.-W. Bian, L. Zhang, X. Bai, and J. Tang, "Richer convolutional features for edge detection," *IEEE Trans. Pattern Anal. Mach. Intell.*, 2019.

[67] J. Revaud, P. Weinzaepfel, Z. Harchaoui, and C. Schmid, "Deepmatching: Hierarchical deformable dense matching," *International Journal of Computer Vision*, vol. 120, no. 3, pp. 300–323, 2016.

[68] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 39, no. 4, pp. 640–651, 2017.

[69] G. Lin, C. Shen, A. Van Den Hengel, and I. Reid, "Efficient piecewise training of deep structured models for semantic segmentation," in *CVPR*, 2016.

[70] B. Shuai, T. Liu, and G. Wang, "Improving fully convolution network for semantic segmentation," *arXiv preprint arXiv:1611.08986*, 2016.

[71] G. Lin, A. Milan, C. Shen, and I. Reid, "Refinenet: Multi-path refinement networks for high-resolution semantic segmentation," in *CVPR*, 2017.

[72] H. Zhang, K. Dana, J. Shi, Z. Zhang, X. Wang, A. Tyagi, and A. Agrawal, "Context encoding for semantic segmentation," in *CVPR*, 2018.

[73] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," in *CVPR*, 2015.

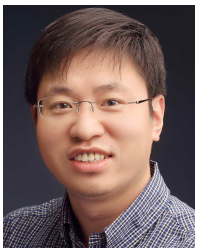
[74] P. Krähenbühl and V. Koltun, "Geodesic object proposals," in *ECCV*, 2014.

[75] P. Rantalankila, J. Kannala, and E. Rahtu, "Generating object segmentation proposals using global and local search," in *CVPR*, 2014.

[76] A. Humayun, F. Li, and J. M. Rehg, "Rigor: Reusing inference in graph cuts for generating object regions," in *CVPR*, 2014.

[77] P. Krahenbuhl and V. Koltun, "Learning to propose objects," in *CVPR*, 2015.

Jianzhong He received the B.S. degree in Computer Science from College of Computer Science of Sichuan University, Chengdu, China, in 2016. He is currently pursuing the Master degree with Peking University, Beijing, China. His current research interests are computer vision and deep learning, with focus on edge detection and image segmentation.



Shiliang Zhang received the Ph.D. degree in computer science from the Institute of Computing Technology, Chinese Academy of Sciences, in 2012. He was a Post-Doctoral Scientist with NEC Laboratories America and a Post-Doctoral Research Fellow with The University of Texas at San Antonio. He is currently a tenure-track Assistant Professor with the School of Electronic Engineering and Computer Science, Peking University.

He has authored or co-authored over 70 papers in journals and conferences, including IEEE TRANSACTIONS ON PATTERN ANALYSIS AND MACHINE INTELLIGENCE, IEEE TRANSACTIONS ON IMAGE PROCESSING, IEEE TRANSACTIONS ON MULTIMEDIA, ACM Multimedia, ICCV, and ECCV. His research interests include large-scale image retrieval and computer vision. He was a recipient of the Distinguished Young Scholar Fund of Beijing Natural Science Foundation, National 1000 Youth Talents Plan of China, Outstanding Doctoral Dissertation Awards from the Chinese Academy of Sciences and Chinese Computer Federation, the President Scholarship from the Chinese Academy of Sciences, the NEC Laboratories America Spot Recognition Award, and the Microsoft Research Fellowship. He was a recipient of the Top 10% Paper Award at the IEEE MMSP 2011. He serves as Associate Editor of IET Computer Vision, reviewer for 20+ Journals including ACM Computing Survey, IJCV, T-PAMI, T-IP, and TPC member for 10+ conferences including ICCV, CVPR, ECCV, ACM MM, IJCAI.

Ming Yang (M'08) received the B.E. and M.E. degrees in electronic engineering from Tsinghua University, Beijing, China, in 2001 and 2004, respectively, and the Ph.D. degree in electrical and computer engineering from Northwestern University, Evanston, IL, USA, in 2008. From 2004 to 2008, he was a Research Assistant with the Computer Vision Group, Northwestern University. After his graduation, he joined NEC Laboratories America, Cupertino, CA, USA, where he was a Senior Researcher. He was a Research Scientist in AI research at Facebook from 2013 to 2015. He is currently the Co-Founder and the VP of software at Horizon Robotics, Inc. He has authored over 40 peer-reviewed publications in prestigious international journals and conferences, which have been cited over 8000 times. His research interests include computer vision, machine learning, face recognition, large scale image retrieval, and intelligent multimedia content analysis.

Yanhu Shan received the B.S. degree from Beijing Information Science and Technology University, Beijing, China, in 2009 and the Ph.D. degree in pattern recognition and intelligent systems from the National Laboratory of Pattern Recognition, Institute of Automation, Chinese Academy of Sciences, Beijing, China, in 2015. He is a researcher at Horizon Robotics from 2017. He was formerly with Samsung R&D Institute after his graduation. His research interests include computer vision and deep learning.

Tiejun Huang (M'01−SM'12) is currently a Professor with the School of Electronic Engineering and Computer Science, Peking University, Beijing, China, where he is also the Director of the Institute for Digital Media Technology. He received the Ph.D. degree in pattern recognition and intelligent system from Huazhong (Central China) University of Science and Technology, Wuhan, China, in 1998, and the masters and bachelor's degree in computer science from the Wuhan University of Technology, Wuhan, in 1995 and 1992, respectively. His research area includes video coding, image understanding, digital right management, and digital library. He has authored or co-authored over 100 peer-reviewed papers and three books. He is a member of the Board of Director for Digital Media Project, the Advisory Board of the IEEE Computing Society, and the Board of the Chinese Institute of Electronics.