

# Deep Crisp Boundaries

Yupei Wang<sup>1,2</sup>, Xin Zhao<sup>1,2</sup>, Kaiqi Huang<sup>1,2,3</sup>

<sup>1</sup>CRIPAC & NLPR, CASIA <sup>2</sup>University of Chinese Academy of Sciences

<sup>3</sup>CAS Center for Excellence in Brain Science and Intelligence Technology

wangyupei2014@ia.ac.cn; xzhao@nlpr.ia.ac.cn; kaiqi.huang@nlpr.ia.ac.cn

## Abstract

Edge detection has made significant progress with the help of deep Convolutional Networks (ConvNet). ConvNet based edge detectors approached human level performance on standard benchmarks. We provide a systematical study of these detector outputs, and show that they failed to accurately localize edges, which can be adversarial for tasks that require crisp edge inputs. In addition, we propose a novel refinement architecture to address the challenging problem of learning a crisp edge detector using ConvNet. Our method leverages a top-down backward refinement pathway, and progressively increases the resolution of feature maps to generate crisp edges. Our results achieve promising performance on BSDS500, surpassing human accuracy when using standard criteria, and largely outperforming state-of-the-art methods when using more strict criteria. We further demonstrate the benefit of crisp edge maps for estimating optical flow and generating object proposals.

## 1. Introduction

Edge detection is a well-established problem in computer vision. Finding perceptually salient edges in natural images is important for mid-level vision [27]. Moreover, edge detection outputs, in terms of boundary maps, are often used for other vision tasks, including optical flow [30], object proposals [2] and object recognition [6]. We have witnessed a significant progress on edge detection, ever since our community embraced a learning based approach [7]. In particular, state-of-the-art methods [39, 18] such as Holistic Edge Detector [39] (HED), leveraging deep ConvNet for detecting edges, achieved human level performance on standard datasets such as BSDS500 [1].

*Is edge detection a solved problem?* In Figure 1(a), we show a visualization of human labeled edges, in comparison to outputs from HED (the current state-of-the-art) and PMI (designed for accurately localizing edges). While the HED result has a higher score, the quality of the edge map is

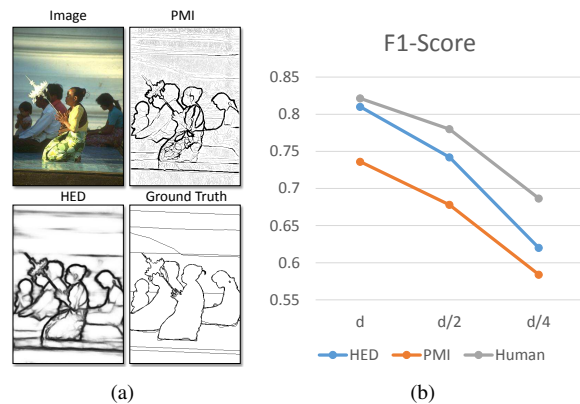


Figure 1. (a) Visualization of edge maps from PMI [14] and HED [39] with input images and ground-truth edges; (b) Performance (on the left image) drops with decreased matching distance. With a tighter distance, the gap between PMI and HED decreases and the gap between HED and human increases. These results suggest that edges from HED are not well aligned with image boundaries. We seek to improve the localization ability of ConvNet based edge detector in this paper.

less satisfactory—edges are blurry and do not stick to actual image boundaries. An accurate edge detector has to balance between “correctness” of an edge (distinguishing between edge and non-edge pixels) and “crispness” of the boundary (precisely localizing edge pixels) [14]. We can capture the “crispness” by decreasing the maximal permissible distance when matching ground-truth edges during benchmark. The F1 score drops dramatically when we tighten the evaluation criteria (see Figure 1(b)).

Both qualitative and quantitative results suggest that edge maps from a ConvNet are highly “correct” yet less “crisp”—edges are not well localized. This issue is deeply rooted in modern ConvNet architecture [19]. First, spatial resolution of features is drastically reduced in more discriminative top layers, leading to blurred output of edges. Second, fully convolutional architecture encourages similar responses of neighboring pixels, and thus may fail to produce a thin edge map. Such a thick and blurred edge map

can be adversarial for other vision tasks [14]. For example, recent optical flow methods [29, 30] require accurate and crisp edge inputs to interpolate sparse matching results, and thus may have sub-optimal performance with blurry edges.

We address this challenging problem of learning a crisp edge detector using ConvNet, and seek to improve the localization ability of HED. To this end, we propose a novel refinement architecture, inspired by the recent advance in dense image labeling [28, 32]. Our method equips an edge detection network with a top-down backward-refining pathway, which progressively increases the resolution of feature maps using efficient sub-pixel convolution [32]. The refinement pathway adds additional non-linearity to the network, further reducing the correlation between edge responses within neighboring pixels. Our method achieves promising results on BSDS500, surpassing human performance when using standard criteria, and largely outperforming state-of-the-art methods when using more strict evaluation criteria. We also demonstrate the benefit of crisp edges for optical flow and object proposals.

Our contributions are thus summarized into three parts.

- We provide a systematical study of edge maps from ConvNet. We show that ConvNet is good at classifying edge pixels yet has poor localization ability.
- We combine the refinement scheme [28] with sub-pixel convolution [7] into a novel architecture, which is specifically designed for learning crisp edge detector.
- Our results on BSDS500 outperform state-of-the-art methods on all matching distances. We also show that crisp edge maps can improve optical flow estimation and object proposals generation.

We organize our paper as follows. Section 2 reviews related work on edge detection. Section 3 presents our study of edge maps from ConvNet. Section 4 details our method. Finally, Section 5 demonstrates experimental results.

## 2. Related Work

There is a vast literature on the classical problem of edge detection. A complete survey is out of scope for this paper. We only review a subset of relevant works in this section.

Early edge detectors are manually designed to find discontinuities in intensity and color [11, 5, 12]. Martin et al. [27] found that adding texture gradients significantly improves the performance. Most recent works explore learning based approaches for edge detection. Dollár et al. [7] proposed a data-driven, supervised edge detector, where detection is posed as a dense binary labeling problem with features collected in local patches. Many modern edge detectors have followed this paradigm by using more sophisticated learning methods. For example, Lim et

al. [22] proposed to cluster human generated contours into so called Sketch Tokens. They then learn a random forest that maps a local patch to these tokens, which is used to re-assemble local edges. This idea was further extended by Dollár and Zitnick. They proposed structured random forest that simultaneously learns the clustering and mapping, and directly outputs a local edge patch. Ren and Bao [38] combined features learned from sparse coding and Support Vector Machine (SVM) for edge detection. Their method can be considered as a two-layer neural network.

The recent success of deep ConvNet has greatly advanced the performance of edge detection. Xie and Tu [39] proposed to combine fully convolutional networks [24] with deep supervision [20]. Their method leverages features from different scales using skip-layer connections and has achieved a superior performance (within 2% gap to human level). Kokkinos [18] further extended HED by adding multi-instance learning, more training samples and a global grouping step. Their results had surpassed human performance on BSDS500, although with significantly more training images. In addition, Maninis et al. [25] proposed to model the orientation of edges. Li et al. [21] presented an unsupervised learning pipeline for edge detection. However, these results tend to emphasis on the “correctness” of edges by selecting an optimistic matching distance,<sup>1</sup> and overlook the “crispness” of edges. In fact, the performance drops dramatically when the evaluation criteria is tightened. In contrast, our method builds on HED and seeks to improve its localization ability.

Our method is motivated by Isola et al. [14]. They proposed an affinity measure based on point-wise mutual information using distributions of hand-crafted local features. Edges are then detected using this affinity with spectral clustering. We share the same goal of designing a crisp edge detector yet our method and setting are completely different. More precisely, we pursue a learning based approach using ConvNet for crisp edges. Finally, our method is inspired by Pinheiro et al. [28], where a refinement architecture is proposed for segmenting objects. Our method adopts the top-down pathway of [28] to label the sparse binary signals of edges. We also replace the bilinear interpolation (deconvolution) with sub-pixel convolution [32], which is critical for generating better-localized, sharp edge output.

## 3. Thick Boundaries from ConvNet

We start by looking into the output edge maps of HED [39], a recent successful edge detector using ConvNet. HED predicts edge confidence at different layers of the network, leading to a set of edge maps. These maps are down-sampled due to successive pooling operations in the network. Thus, they are further up-sampled to fit the input

<sup>1</sup>4.3 pixels in a resolution of  $321 \times 481$

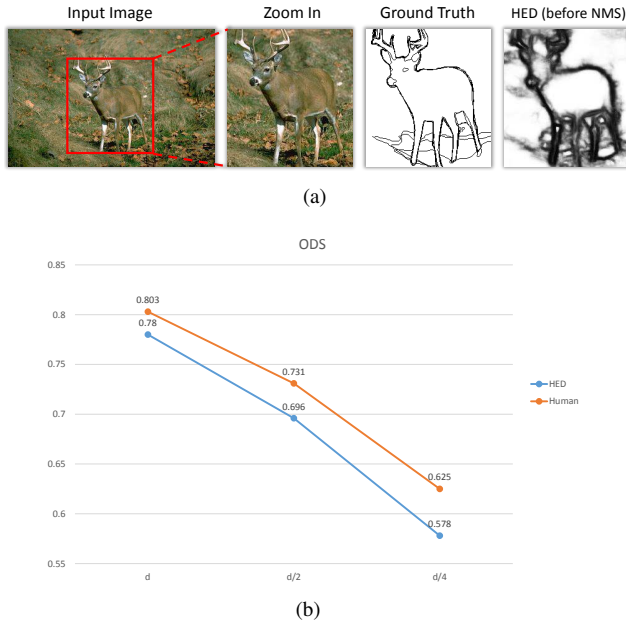


Figure 2. (a) Thick and noisy edge map generated with HED [39] before non-maximal suppression(NMS); (b) Optimal Dataset Score (ODS) for both HED and human drop with decreased matching distance on the BSDS500 test set. However, the performance gap between HED and human increases from 2.3% to 4.7% as the distance decreases from  $d$  to  $d/4$ .

resolution by bilinear interpolation and averaged to produce the final edge map. We show an example of the edge map in Figure 2(a). Although the detector achieved a ODS of 0.78 on BSDS, the visual quality of the edge map is not satisfying. Edges look blurred and visually defective.

Why would such a blurry edge map reach a high score in benchmark? The standard evaluation [1] iterates over all confidence thresholds and uses bipartite graph matching to match between a binarized edge map to ground-truth edges. The matching is controlled by a maximal permissible distance  $d$ . A misaligned edge pixel is still considered correct as long as its distance to the nearest ground-truth is smaller than  $d$  pixels. With a optimistic  $d$ , we can achieve a good score even if edges are slightly shifted.

In fact, edge detection has to balance between “correctness” of an edge (distinguishing between edge and non-edge pixels) and “crispness” of the boundary (precisely localizing edge pixels) [14]. Crisp edges may be critical for other vision tasks, such as optical flow or image segmentation. “Crispness” can be measured by decreasing  $d$  in the benchmark. Human performance gradually decreases with smaller  $d$ , as we show in Figure 2(b). However, HED outputs show a more drastic drop, indicating that HED edges are not well aligned to actual image boundaries. This is in accordance with our visual inspection of the edge map.

## 4. Make Convolutional Boundaries Crisp

How can we make a crisp edge map from ConvNet? We start by analyzing the architecture of HED. Like modern ConvNets, spatial resolution of more discriminative top layers is significantly reduced due to pooling operations. HED further attaches a linear classifier on layers with different resolution, and uses bilinear interpolation (realized as deconvolution) to up-sample their outputs to the original resolution. This design has two major issues. First, linear classifiers within the fully convolution architecture produce similar responses at neighboring pixels. It is thus difficult to distinguish an edge pixel from its neighbors. More importantly, up-sampling further blurs the edge map.

Architecture modifications are thus required for generating a crisp edge map. In this section, we address the challenging problem of designing a Crisp Edge Detector (CED) by proposing a novel architecture. Our method supplements HED network with a backward-refining pathway, which progressively up-samples features using efficient sub-pixel convolution [32]. CED is able to generate an edge map that is better aligned with image boundaries. We present details of CED and explain our design choices.

### 4.1. Architecture Overview

Figure 3 shows an overview of CED with two major components: the forward-propagating pathway and backward-refining pathway. The forward-propagating pathway is similar to HED. It generates a high-dimensional low-resolution feature map with rich semantic information. The backward-refining pathway fuses the feature map with intermediate features along the forward-propagating pathway. This refinement is done multiple times by a refinement module. Each time we increase the feature resolution by a small factor (2x) using sub-pixel convolution, eventually reaching the input resolution. Details of our network are elaborated in following subsections.

### 4.2. Refinement Module

The skip-layer connection provides HED the important ability to use features at different layers for finding edges [39]. HED simply averages independent predictions from all side-output layers. We argue that this is not a good design as it does not explore the hierarchical feature representations of ConvNet. To get a better fusion of multi-layer features, we introduce the backward-refining pathway with refinement modules, similar to [28]. Note that our task of detecting sparse edges is significantly different from segmenting objects in [28]. Thus, directly applying the same module in [28] leads to sub-optimal performance.

The refinement module is repeated several times to progressively increase the resolution of feature maps. The key idea is to aggregate evidences of edges across the path using

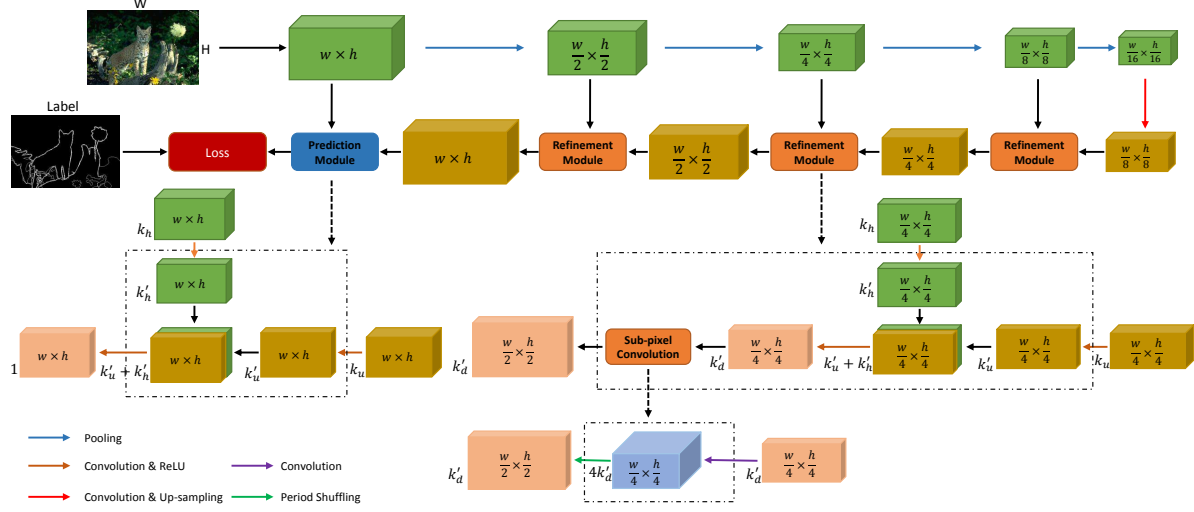


Figure 3. Our method of Crisp Edge Detector (CED). We add a backward-refining pathway, which progressively increase the resolution of feature maps. Our refinement module fuses a top-down feature map with feature maps on the forward pass, and up-samples the map using sub-pixel convolution. This architecture is specially designed for generating edge maps that are well-aligned to image boundaries.

intermediate feature maps. Detailed structure of the module is shown in the bottom part of 3. Each module fuses a top-down feature map from the backward pathway with the feature map from current layer in the forward pathway, and further up-samples the map by a small factor (2x), which is then passed down the pathway. There are two core components in this module, namely **fusion** and **up-sampling**.

**Fusion:** A straightforward strategy of fusion is to directly concatenate two feature maps. However, this is problematic since they have different number of feature channels. Directly concatenating the features risks drowning out the lower dimensional signal. Similar to [28], we match the number of feature channels between the two maps through dimension reduction. This is done by reducing the dimension of both feature maps. We then concatenate two low-dimensional feature maps with equal channels.

We denote the number of channels of the input forward pathway feature map as  $k_h$ . After the convolutional and ReLU operations, the channels are reduced to  $k'_h$ , which is much less than  $k_h$ . The same operations are conducted to the feature map from the previous refinement module to produce  $k'_u$  from  $k_u$ . We concatenate the above feature maps into a new feature map with  $k'_u + k'_h$  channels and reduce it to a feature map with  $k'_d$  channels by a  $3 \times 3$  convolutional layer as well. Thus, the overall computational cost is reduced and the two input feature maps are balanced.

**Up-sampling:** After fusion, our refinement module will also expand the resolution of feature maps. We up-sample the fused feature map with a sub-pixel convolution [33]. The sub-pixel convolution, different from the popular deconvolution for up-sampling [40, 10, 36], is a

standard convolution followed by additional rearrangement of feature values, termed phase shift. It helps to remove the block artifact in image super-resolution task and maintain a low computational cost. We found that using sub-pixel convolution is important for better localization of edges.

Supposed we have  $i$  input channels and  $o$  desired output channels, the kernel size of a convolutional layer is denoted as  $(o, i, r, c)$ , where  $r$  and  $c$  stand for the kernel width and kernel height respectively. Considering output feature map with  $k$  times larger resolution than the input one, the traditional deconvolutional layer would employ the kernel size to be  $(o, i, k \times r, k \times c)$ . Instead of directly output enlarged feature map through a single deconvolutional layer, the sub-pixel convolution consists of one convolutional layer and one following phase shift layer. The kernel size of the convolutional layer is  $(o \times k^2, i, r, c)$ , thus generating feature map with  $o \times k^2$  feature channels with identical resolution. Then we apply the phase shift to assemble the output feature map to the feature map with  $o$  feature channels but  $k$  times larger resolution in a fixed order.

**Relationship to [39] and [28]:** CED subsumes HED [39] as a special case, where  $3 \times 3$  convolutions and ReLUs are replaced by linear classifiers and progressive up-sampling is used. Our method is different from [28] as we replace bilinear interpolation with sub-pixel convolution. This enables a more expressive model with a small number of extra parameters. Our task of edge detection is also different from object segmentation in [28].

### 4.3. Implementation Details

Our implementation builds on the publicly available code of HED [39], using Caffe as backend [15]. For train-

ing, we initialize the forward-propagating pathway with the pre-trained HED model. The other layers are initialized with Gaussian random distribution with fixed mean (0.0) and variance(0.01). The hyper parameters, including the initial learning rate, weight decay and momentum, are set to  $1e-5$ ,  $2e-4$  and 0.99, respectively.

For backward-refining pathway, the number of convolutional kernels is set to 256 for the top layer. This number is decreased by half along the path. For example, the first, second, and third top-down refinement module will have 128, 64 and 32 feature channels, respectively. Since the resolution of feature maps decreases by a factor of 2 after every pooling operation, the sub-pixel convolution up-samples the input feature map by 2x in each refinement module.

## 5. Experiments

We conduct extensive experiments to benchmark CED. We first present our datasets and evaluation criteria. Our experiments start with an ablation study of network architecture. We further compare our best performing method with state-of-the-art methods on edge detection. Finally, we plug in our method into optical flow estimation and object proposals generation, and evaluate its benefit for each task.

### 5.1. Datasets and Benchmarks

We evaluate edge detectors on the widely-used Berkeley Segmentation Dataset and Benchmark (BSDS500) dataset [27, 1]. It consists of 200 images for training, 100 for validation, and 200 for testing. Each image is annotated by multiple annotators. We use the train and validation set for training (similar to [39]) and report results on the test set. The performance is measured by the precision/recall curve that captures the trade-off between accuracy and noise [27]. In addition, three standard metrics are reported: fixed contour threshold (ODS), per-image best threshold (OIS) and average precision (AP).

We benchmark optical flow estimation and object proposals generation method by applying our edge detector. The optical flow estimation results are reported on MPI Sintel dataset [4], a challenging optical flow evaluation benchmark obtained from animated sequences, and we use the final version with photo-realistic rendering. Similar to [30, 21], we report Average Endpoint Error (AEE) on the training set for optical flow estimation.

We benchmark object proposals generation on Pascal VOC 2012 validation set (VOC12 val set) [26]. The mean Jaccard index (mean best overlap) at instance level and class level are reported for the evaluation, as the same metrics in [2, 16]. More precisely, the Jaccard index at instance level ( $J_i$ ) is the mean best overlap (intersection over union) for all the ground-truth instances in the dataset. The Jaccard index at class level ( $J_c$ ) is the mean over the ground-truth instances within class  $c$ .

Method	ODS	OIS	AP
HED	.780	.797	.829
CED-w/o-Subpixel-w/o-Multi	.793	.811	.838
CED-w/o-Multi	.794	.811	.847
CED-w/o-Subpixel	.800	.819	.859
CED	<b>.803</b>	<b>.820</b>	<b>.871</b>

Table 1. Results on BSDS500 with different network architecture settings. CED-w/o-Multi refers to CED without multi-scale testing, similar for CED-w/o-Subpixel-w/o-Multi.

### 5.2. Ablation Study

Our first experiment is to test different network architectures of CED. We use the original HED [39] network as our baseline. We trained different versions of CED with or without sub-pixel convolution (CED-w/o-Subpixel by using deconvolution instead). This is carried out to prove the effectiveness of sub-pixel convolution. It is worth noting that both CED and CED-w/o-Subpixel are initialized by the same model. Moreover, we tested the multi-scale fusion strategy for the evaluation. In this case, a testing image is resized to three different resolutions (1/2x, 1x, 2x), which are fed into the same network independently. Finally, we resize the three output edge maps to the original resolution, and average them to generate the final edge map.

In training, we adopt a modified version of consensus sampling strategy [39] to prevent the problematic convergence behavior. A pixel is assigned positive label if it is labeled as edge by at least three annotators. Pixels have not been labeled by any annotators are treated as negative. The rest of the pixels are ignored during training (by blocking their gradients). We also augment the data by rotating, cropping and multi-scale resizing. Our results are summarized in Table 1. Our refinement module improves over the baseline HED by 1%, sub-pixel convolution further boosts the performance by 1.4% and multi-scale testing adds another 0.9%. Our full model improves the ODS from 0.780 to 0.803, slightly higher than human performance of 0.8027. These results demonstrate the effectiveness of CED.

### 5.3. Boundary Detection

We further compare the best performing version of CED to state-of-the-art methods in Table 2. Fig. 4 shows Precision-Recall curves of all methods for comparison. Without multi-scale testing, CED already achieves better results than the top-performing method [23] in all 3 metrics. Integrated with the multi-scale testing, CED achieves a further improvement, enhancing the ODS by 1.1%, OIS by 1.0% and AP by 5.3% in comparison to [23]. This result also surpasses the human benchmark on the BSDS500 dataset with ODS 0.8027. We note that the current record is from DeepBounaries [18], which used extra training sam-



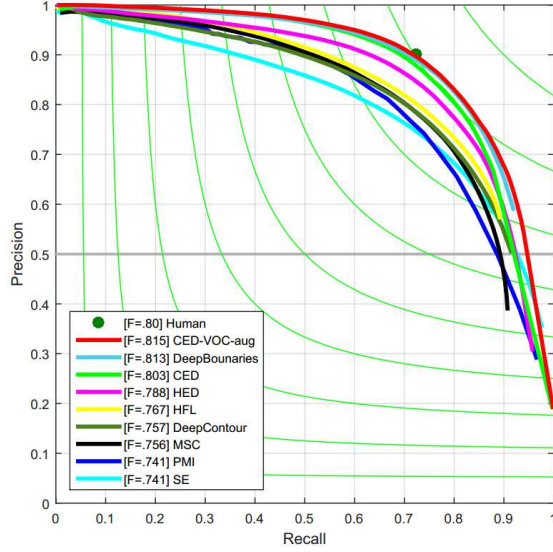


Figure 4. Precision/Recall curves of different methods on BSDS500 dataset using standard evaluation criteria. CED is close to the best record by DeepBoundaries, which uses extra training data and post-processing steps. Simply augmenting training data as DeepBoundaries, without any post-processing, CED-VOC-aug achieves the best result.

Method	ODS	OIS	AP
Human	.8027	.8027	-
gPb-owt-ucm[1]	.726	.757	.696
SE-Var[9]	.746	.767	.803
PMI[14]	.741	.769	.799
MES[34]	.756	.776	.756
DeepEdge [17]	.753	.769	.784
MSC [35]	.756	.776	.787
CSCNN [13]	.756	.775	.798
DeepContour [31]	.757	.776	.790
HFL [3]	.767	.788	.795
HED [39]	.788	.808	.840
RDS [23]	.792	.810	.818
DeepBoundaries [18]	.813	.831	.866
CED-w/o-Multi	.794	.811	.847
CED	<b>.803</b>	<b>.820</b>	<b>.871</b>
CED-VOC-aug	<b>.815</b>	<b>.833</b>	<b>.889</b>

Table 2. Comparison to the state-of-arts on BSDS500 dataset.

ples ( $> 10K$  images in VOC) and post-processing steps of global grouping. After simply augmenting the standard BSDS500 training dataset with VOC images as [18], without any post-processing steps, CED gives better results with ODS 0.815. The new form of CED is denoted as CED-VOC-aug.

We further benchmark the “crispness” of edges from CED. We report quantitative evaluation results by varying

the matching distance  $d$ . The selected evaluation method should indicate whether the object contours can be precisely localized by tightened criteria. We evaluate CED on the following settings of  $d$ :  $d_0$ ,  $d_0/2$ , and  $d_0/4$ , where  $d_0 = 4.3$  pixels. We compare the results with HED [39] and PMI [14], which also aims for generating crisp edges. The results are reported in the plot in Figure 6.

The performance of all methods decreases when  $d$  decreases. The gap between HED and PMI is getting closer with a smaller  $d$ . In contrast, the gap between CED and the two baselines stays fairly consistent. In fact, the ODS gap between CED and HED increases from 2.3% to 2.8%, the OIS gap increases from 2.3% to 2.9% and the AP gap increases from 4.2% to 9.1%. In addition, CED achieves ODS=0.606 at the setting of  $d_0/4$ , approaching human level performance (0.625), and outperforming the methods in [14, 39] by a large margin. The results suggest that CED produces a crisp edge map.

Finally, Figure 5 shows a comparison of edge maps from PMI, HED and CED, before non-maximal suppression (NMS). Even without the standard non-maximal suppression (NMS), our method eliminates most blurry and noisy boundaries significantly. We observe that CED can produce cleaner, thinner and crisper image boundaries.

#### 5.4. Optical Flow with Crisp Boundaries

To analyze the benefit of crisp edges, we plug in CED results for optical flow estimation. In this case, we train CED on BSDS500 and test it on Sintel. We choose EpicFlow [30] as our optical flow method. EpicFlow computes geodesic distance using an edge map, which is further used to interpolate sparse matches from [37]. Thus, an accurate edge map is important for good flow results. We compare CED results with HED. The AEE on Sintel training set using CED is 3.570 while HED gives 3.588. This result illustrates that the optical flow can benefit from a better localized edge map. Figure 8 shows visualizations of sample flow maps from Sintel. Again, CED get slightly more accurate flow results than HED.

#### 5.5. Object Proposals with Crisp Boundaries

We also demonstrate the benefit of the crisp edge map for object proposals generation—another important mid-level vision task. We choose the Multi-scale Combinatorial Grouping (MCG) and its single scale version (SCG) in [2] to generate object proposals. With an input edge map, MCG builds a hierarchical grouping of contours to generate object proposals. The original MCG adopts the Structured Edge (SE) [8] as the default edge detector. We simply replace the edge detector with HED [39] and CED. Note that HED and CED are both trained only on the BSDS500 dataset. We benchmark the combination of three edge detectors (SE, HED, CED) with both MCG and SCG.

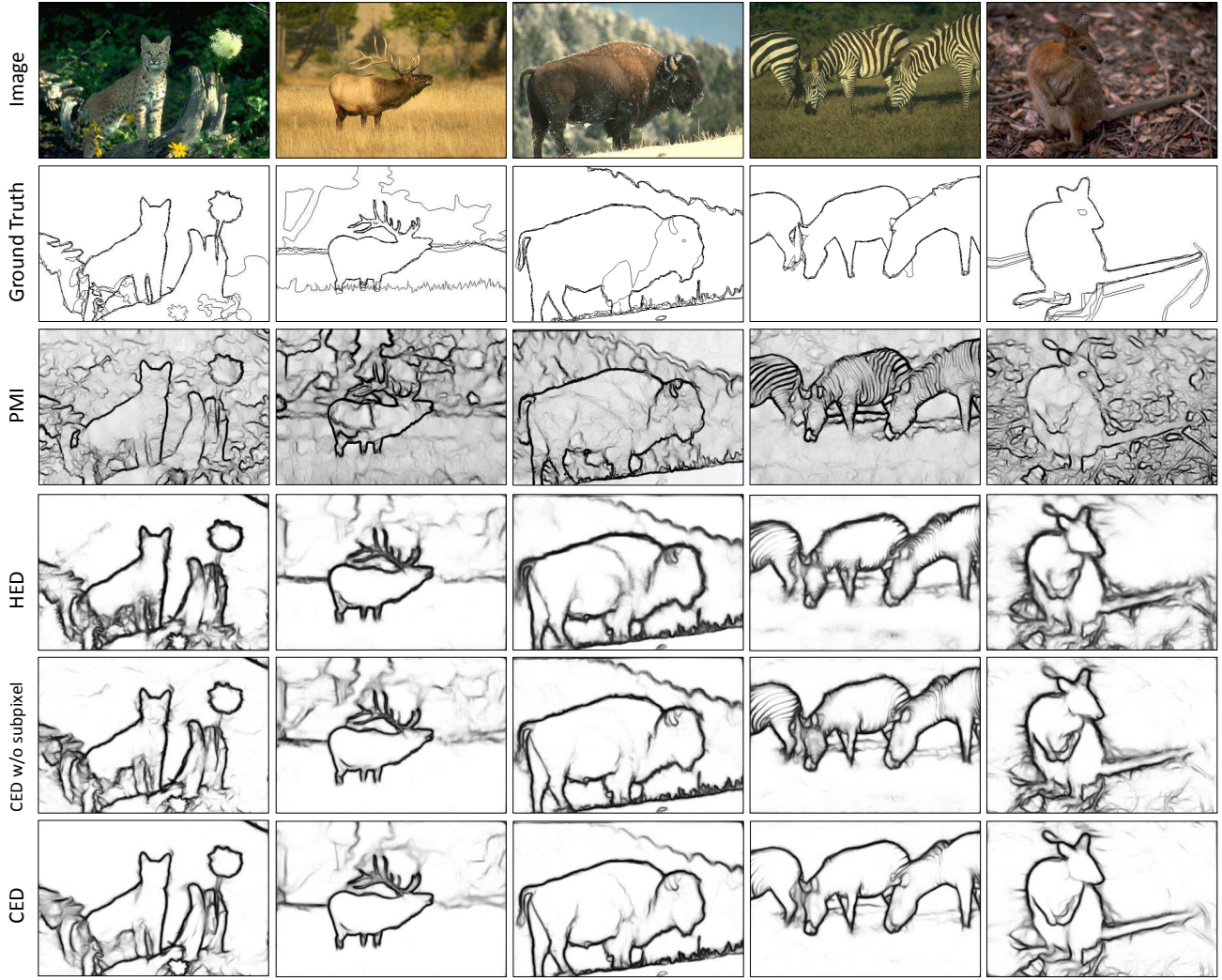


Figure 5. Visualization of edge detection results from different methods. First two rows show the original images and ground-truths edges. The next four rows include the raw edge maps (before NMS) of PMI, HED, CED-w/o-Subpixel and CED, respectively. Edge maps from CED is sharper than HED and cleaner than PMI.

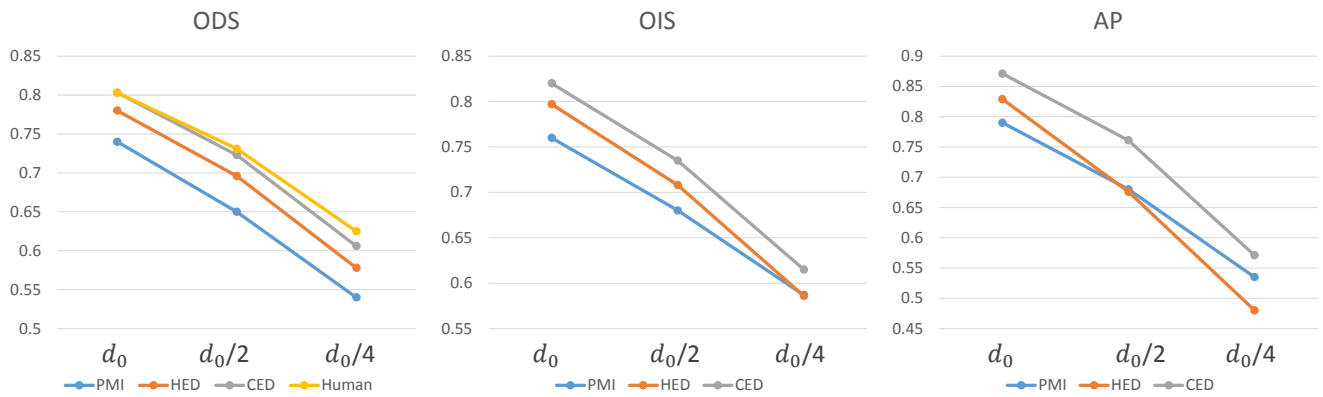


Figure 6. “Crispness” of edges. We report the performance (ODS, OIS and AP) as a function of the maximal permissible distance  $d$ .

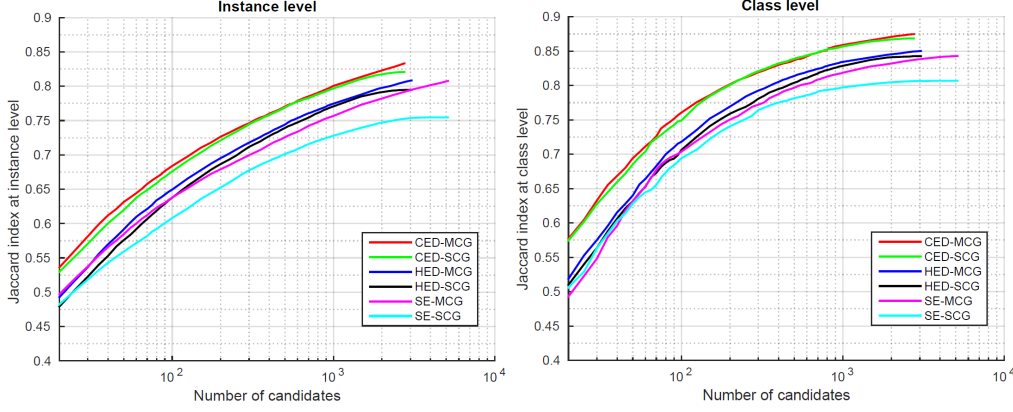


Figure 7. Object proposals performance on VOC12 val set at instance level (left) and class level (right). We evaluate three different edge detectors (SE, HED, CED) with two grouping methods (MCG, SCG), and report curve of the mean Jaccard index with respect to all number of proposals. CED-MCG achieves the best results.

	Nc	Plane	Bicycle	Bird	Boat	Bottle	Bus	Car	Cat	Chair	Cow	Table	Dog	Horse	MBike	Person	Plant	Sheep	Sofa	Train	TV	Global
SE-MCG	100	71.2	38.6	74.7	64.5	53.7	70.6	48.9	82.7	55.4	79.1	66.7	78.8	70.6	65.4	60.4	49.9	71.9	73.7	71.3	76.2	63.8
HED-MCG	100	73.0	39.3	79.9	70.0	56.2	68.8	50.7	86.9	55.1	81.6	62.7	85.0	73.7	63.9	59.2	56.5	76.2	72.6	68.6	73.6	64.9
CED-MCG	100	78.2	42.2	85.1	72.3	55.2	76.4	56.0	90.5	56.2	83.6	68.2	88.2	76.9	70.2	62.8	55.7	78.8	77.4	78.2	75.3	<b>68.4</b>
SE-MCG	5138	83.0	51.2	86.4	79.7	78.1	81.8	77.4	90.8	74.5	88.7	84.2	88.1	81.4	78.6	79.7	77.5	86.7	87.8	81.9	90.3	80.8
HED-MCG	3051	83.3	52.2	87.8	81.5	77.3	80.4	78.3	94.2	74.2	90.8	82.6	92.2	84.0	76.9	78.4	75.5	90.0	87.6	80.0	87.9	80.8
CED-MCG	2757	86.8	54.7	90.3	82.7	77.5	86.9	82.2	95.1	77.1	92.0	84.6	93.2	86.2	80.8	80.8	78.8	91.0	89.4	86.3	89.6	<b>83.3</b>

Table 3. Object proposals performance on VOC12 val set with top-100 and all proposals. We report per-class and mean Jaccard index (mean best overlap) at instance level. Our CED clearly outperforms all other edge detectors.

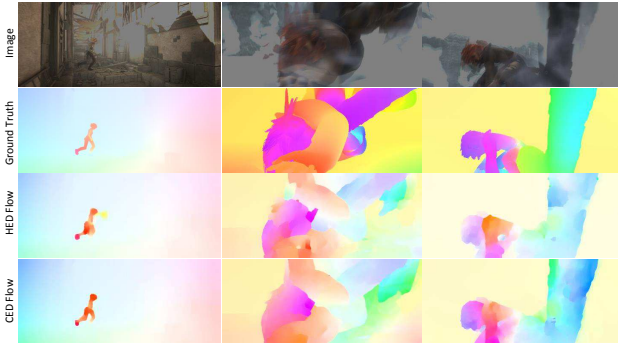


Figure 8. Visualization of optical flow estimation results with different edge maps. From top to bottom: mean of two consecutive images, ground-truth flow, and optical flow estimation results using edge maps generated with HED and CED. CED produces better motion details, such as the leg of the girl in the first image.

We report the mean Jaccard index at both instance level and class level with respect to the number of proposals in Figure 7. CED-MCG achieves the best results at both metrics for all number of proposals. Table 3 further compares the Jaccard index at instance level for each class at two operation points of  $N_c$ , namely the top-100 proposals and all proposals. Our method outperforms HED by 4.6% with top-100 proposals, and by 2.7% with all proposals. These results further demonstrate the benefits of crisp edges.

## 6. Conclusion

We showed that ConvNet based edge detector tends to generate edge maps which are not well aligned with image boundaries. We discussed the reason behind the issue and proposed a novel architecture that largely improved its localization ability. Our detector achieved promising performance on BSDS500, outperforming the state-of-the-art methods when using strict maximum tolerance setting. We verified the benefit of crisp edge map for optical flow estimation and object proposals generation. We also plan to apply CED to other vision tasks, such as semantic segmentation.

Our work looked into edge detection, a classical problem in computer vision. We hope that it will provide a reflection of the recent victory of ConvNet in computer vision. While we are getting better results from standard quantitative evaluations, the fundamental vision problem still remains open. It is probably the right time to revisit our evaluation criteria.

## Acknowledgement

This work is funded by the National Key Research and Development Program of China (2016YFB1001005), the National Natural Science Foundation of China (Grant No.61673375, Grant No.61602485), and the Projects of Chinese Academy of Science, (Grant No.QYZDB-SSW-JSC006, Grant No.173211KYSB20160008).



## References

- [1] P. Arbeláez, M. Maire, C. Fowlkes, and J. Malik. Contour detection and hierarchical image segmentation. *TPAMI*, 2011.
- [2] P. Arbeláez, J. Pont-Tuset, J. Barron, F. Marques, and J. Malik. Multiscale combinatorial grouping. In *CVPR*, 2014.
- [3] 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.
- [4] D. J. Butler, J. Wulff, G. B. Stanley, and M. J. Black. A naturalistic open source movie for optical flow evaluation. In *ECCV*, 2012.
- [5] J. Canny. A computational approach to edge detection. *TPAMI*, (6):679–698, 1986.
- [6] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In *CVPR*, volume 1, pages 886–893. IEEE, 2005.
- [7] P. Dollár, Z. Tu, and S. Belongie. Supervised learning of edges and object boundaries. In *CVPR*, volume 2, pages 1964–1971. IEEE, 2006.
- [8] P. Dollár and C. L. Zitnick. Structured forests for fast edge detection. In *ICCV*, 2013.
- [9] P. Dollár and C. L. Zitnick. Fast edge detection using structured forests. *TPAMI*, 2015.
- [10] V. Dumoulin and F. Visin. A guide to convolution arithmetic for deep learning. *arXiv preprint arXiv:1603.07285*, 2016.
- [11] J. R. Fram and E. S. Deutsch. On the quantitative evaluation of edge detection schemes and their comparison with human performance. *IEEE Transactions on Computers*, 100(6):616–628, 1975.
- [12] W. T. Freeman and E. H. Adelson. The design and use of steerable filters. *TPAMI*, 13(9):891–906, 1991.
- [13] J.-J. Hwang and T.-L. Liu. Pixel-wise deep learning for contour detection. *arXiv preprint arXiv:1504.01989*, 2015.
- [14] P. Isola, D. Zoran, D. Krishnan, and E. H. Adelson. Crisp boundary detection using pointwise mutual information. In *ECCV*, 2014.
- [15] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell. Caffe: Convolutional architecture for fast feature embedding. *arXiv preprint arXiv:1408.5093*, 2014.
- [16] Y. Jimei, P. Brian, C. Scott, L. Honglak, and Y. Ming-Hsuan. Object contour detection with a fully convolutional encoder-decoder network. In *CVPR*, 2016.
- [17] J. J. Kivinen, C. K. Williams, N. Heess, and D. Technologies. Visual boundary prediction: A deep neural prediction network and quality dissection. In *AISTATS*, 2014.
- [18] I. Kokkinos. Pushing the boundaries of boundary detection using deep learning. *ICLR*, 2016.
- [19] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *NIPS*, pages 1097–1105, 2012.
- [20] C.-Y. Lee, S. Xie, P. Gallagher, Z. Zhang, and Z. Tu. Deeply supervised nets. In *AISTATS*, volume 2, page 6, 2015.
- [21] Y. Li, M. Paluri, J. M. Rehg, and P. Dollár. Unsupervised learning of edges. In *CVPR*, 2016.
- [22] J. J. Lim, C. L. Zitnick, and P. Dollár. Sketch tokens: A learned mid-level representation for contour and object detection. In *CVPR*, pages 3158–3165, 2013.
- [23] Y. Liu and M. S. Lew. Learning relaxed deep supervision for better edge detection. In *CVPR*, 2016.
- [24] J. Long, E. Shelhamer, and T. Darrell. Fully convolutional networks for semantic segmentation. In *CVPR*, pages 3431–3440, 2015.
- [25] K.-K. Maninis, J. Pont-Tuset, P. Arbeláez, and L. Van Gool. Convolutional oriented boundaries. In *ECCV*, 2016.
- [26] E. Mark, G. Luc, Van, W. Chris, W. John, and Z. Andrew. The pascal visual object classes (voc) challenge. *IJCV*, 88(2):303–338, 2010.
- [27] D. R. Martin, C. C. Fowlkes, and J. Malik. Learning to detect natural image boundaries using local brightness, color, and texture cues. *TPAMI*, 2004.
- [28] P. O. Pinheiro, T.-Y. Lin, R. Collobert, and P. Dollár. Learning to refine object segments. In *ECCV*, 2016.
- [29] X. Ren. Local grouping for optical flow. In *CVPR*, pages 1–8. IEEE, 2008.
- [30] J. Revaud, P. Weinzaepfel, Z. Harchaoui, and C. Schmid. Epicflow: Edge-preserving interpolation of correspondences for optical flow. In *CVPR*, 2015.
- [31] W. Shen, X. Wang, Y. Wang, X. Bai, and Z. Zhang. Deep-contour: A deep convolutional feature learned by positive-sharing loss for contour detection. In *CVPR*, 2015.
- [32] W. Shi, J. Caballero, F. Huszár, J. Totz, A. P. Aitken, R. Bishop, D. Rueckert, and Z. Wang. Real-time single image and video super-resolution using an efficient sub-pixel convolutional neural network. In *CVPR*, 2016.
- [33] W. Shi, J. Caballero, F. Huszár, J. Totz, A. P. Aitken, R. Bishop, D. Rueckert, and Z. Wang. Real-time single image and video super-resolution using an efficient sub-pixel convolutional neural network. In *CVPR*, 2016.
- [34] A. Sironi, V. Lepetit, and P. Fua. Projection onto the manifold of elongated structures for accurate extraction. In *ICCV*, 2015.
- [35] A. Sironi, E. Türetken, V. Lepetit, and P. Fua. Multiscale centerline detection. *TPAMI*, 2016.
- [36] A. Vedaldi and K. Lenc. Matconvnet: Convolutional neural networks for matlab. In *ACMMM*, 2015.
- [37] P. Weinzaepfel, J. Revaud, Z. Harchaoui, and C. Schmid. DeepFlow: Large displacement optical flow with deep matching. In *ICCV*, Sydney, Australia, Dec. 2013.
- [38] R. Xiaofeng and L. Bo. Discriminatively trained sparse code gradients for contour detection. In *NIPS*, pages 584–592, 2012.
- [39] S. Xie and Z. Tu. Holistically-nested edge detection. In *ICCV*, 2015.
- [40] M. D. Zeiler, G. W. Taylor, and R. Fergus. Adaptive deconvolutional networks for mid and high level feature learning. In *ICCV*, 2011.