Learning to Predict Crisp Boundaries

Ruoxi Deng[1,⋆], Chunhua Shen[2], Shengjun Liu[1], Huibing Wang[3,⋆], Xinru Liu[1]

[1]Central South University, China; [2]The University of Adelaide, Australia; [3]Dalian University of Technology, China[⋆]

Abstract. Recent methods for boundary or edge detection built on Deep Convolutional Neural Networks (CNNs) typically suffer from the issue of predicted edges being thick and need post-processing to obtain crisp boundaries. Highly imbalanced categories of boundary versus background in training data is one of main reasons for the above problem. In this work, the aim is to make CNNs produce sharp boundaries without post-processing. We introduce a novel loss for boundary detection, which is very effective for classifying imbalanced data and allows CNNs to produce crisp boundaries. Moreover, we propose an end-to-end network which adopts the bottom-up/top-down architecture to tackle the task. The proposed network effectively leverages hierarchical features and produces pixel-accurate boundary mask, which is critical to reconstruct the edge map. Our experiments illustrate that directly making crisp prediction not only promotes the visual results of CNNs, but also achieves better results against the state-of-the-art on the BSDS500 dataset (ODS F-score of .815) and the NYU Depth dataset (ODS F-score of .762).

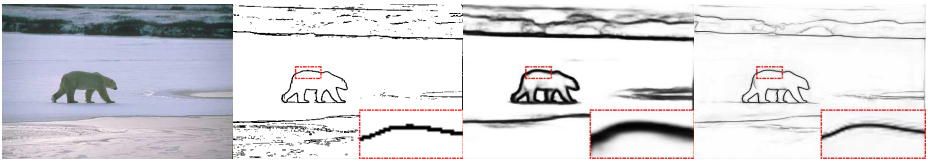
Keywords: Edge detection, contour detection, convolutional neural networks

1 Introduction

Edge detection is a long-standing task in computer vision [1, 2]. In early years, the objective is defined as to find sudden changes of discontinuities in intensity images [3]. Nowadays, it is expected to localize semantically meaningful objects boundaries, which play a fundamental and significant role in many computer vision tasks such as image segmentation [4–7] and optical flow [8, 9]. In the past few years, deep convolutional neural networks (CNNs) have dominated the research on edge detection. CNN based methods, such as DeepEdge [10], DeepContour [11], HED [12] and RCF [13], take advantage of its remarkable ability of hierarchical feature learning and have demonstrated state-of-the-art F-score performance on the benchmarks such as BSDS500 [5] and NYUDv2 [14].

Although CNN-based methods are good at producing semantically meaningful contours, we observe a common behavior that their prediction is much thicker than the result of classic methods. For example, in Figure 1 we show two prediction examples from the Sobel detector [15] and the HED detector, respectively.

⋆ Part of this work was done when R. Deng and H. Wang were visiting The University of Adelaide.



(a) (b) (c) (d)

Fig. 1: (a) is an example image from the BSDS500 dataset [5]. (b) is the result of the Sobel detector [15]. Here (c) is the output of the HED detector [12]. (d) is the output of our proposed method. All the predictions do not apply post-processing.

The edge of the polar bear that we highlight in the dotted rectangle on the HED result is roughly 10 pixels wide, which is two times wider than the Sobel result (roughly 4 pixels). Note that, the behavior of thick prediction is not only on the result of HED but also can be found in many recent representative works such as RCF [13], Casenet [16] and CEDN [17].

Existing works in the literature seldom discuss this issue of predicted boundaries being overly thick. One possible reason is that edge detection methods typically apply edge thinning post-processing to obtain one-pixel wide results after generating an initial prediction. Therefore it seems no difference that how wide the initial prediction is. However, this behavior attracts our attention and we believe it is worth finding out the reason behind it which in turn improves the quality of prediction. The work in [45] addressed this problem by proposing a refinement architecture (encoder-decoder) for achieving crips edges. As we show in our experiments, it only slightly improves the result of HED. Instead of modifying the convolutional network for boundary detection, we address this problem by investigating the loss function.

In this work, we explore and solve the thickness issue of CNN-based boundary predictions. We present an end-to-end fully convolutional network which is accurate, fast and convenient to perform image-to-boundary prediction. Our method consists of two key components, which are a fully convolutional neural network of the bottom-up/top-town architecture and a simple yet effective loss function. The method can automatically learn rich hierarchical features, resolve ambiguity in prediction and predict crisp results without postprocessing. Figure 1 gives an example of the improvement of edge quality between our method and the HED detector. More examples can be found in Section 4. We demonstrate that tackling the issue of thickness is critical for CNNs performing crisp edge detection, which improves the visual result as well as promotes the performance in terms of boundary detection evaluation metrics. We achieve the state-of-the-art performance on the BSDS500 dataset with the ODS F-score of 0.815 and the fast version of our method achieves ODS F-score of 0.808 at the speed of 30 FPS.

2 Related work

Edge detection has been studied for over forty years. There are plenty of related works and here we only highlight a few representative works. Early edge detectors focus on computing the image gradients to obtain the edges [18–21]. For example, the Sobel detector [18] slides a 3×3 filter on a gray image to compute the image gradient for the response to the edge pixel. The Canny detector [19] goes a step further by removing the noise on the output map and employing non-maximum suppression to extract one-pixel wide contour. These traditional methods are often used as one of the fundamental features in many computer vision applications [4, 22, 23]. Learning-based methods [24, 5, 25, 26] often ensemble different low-level features and train a classifier to generate object-level contours. Although these methods achieved great performance compared to traditional methods, they rely on hand-crafted features which limit their room for improvement.

Recent state-of-the-art methods for edge detection [12, 13, 27] are built on deep convolutional neural networks [28, 29]. DeepEdge [10] extracts multiple patches surrounding an edge candidate point (extracted by the Canny detector) and feeds these patches into a multi-scale CNN to decide if it is an edge pixel. DeepContour [11] is also a patch-based approach which first divides an image into many patches then put these patches into the network to detect if the patch has a contour. Differing from these works, the HED detector [12] is an end-to-end fully convolutional neural network which takes an image as input and directly outputs the prediction. It proposes a weighted cross entropy loss and takes the skip-layer structure to make independent predictions from each block of the pre-trained VGG model [30] and average the results. RCF [13] also utilizes the skip-layer structure and the similar loss with HED, yet it makes independent predictions from each convolutional layer of the VGG model. CEDN [17] employs an encoder-decoder network and train the network on the extra data of Pascal VOC dataset. CASENet [16] proposes a novel task which is to assign each edge pixel to one or more semantic classes and solve the task by utilizing an end-to-end system similar to HED.

Summarizing the development of deep learning based methods, we find that the HED detector is very popular and has enlightened many subsequent methods such as RCF, CASENet and the works mentioned in the paper [31]. However, we observe empirically that the weighted cross entropy loss employed by the HED detector may have contributed to the resulted edges being thick. We verify this in the next section.

Contributions In this work, we develop an end-to-end edge detection method. Our main contributions are as follows. We aim to detect crisp boundaries in images using deep learning. We explore the issue of predicted edges being overly thick, which can be found in almost all recent CNNs based methods. We propose a method that manages to tackle the thickness issue. It allows CNN-based methods to predict crisp edge without resorting to post-processing. Furthermore, our experiments show that our method outperforms previous state-of-the-art methods on the BSDS500 and NYUDv2 datasets.

(a) (b) (c)

Fig. 2: A simple test on the class-balance weight β. From left to right: (a) is the original image from the BSDS500 dataset. (b) is the result of using the standard cross-entropy loss, i.e., $\beta = 0.5$. (c) is the result of using the weighted cross-entropy loss.

3 The proposed method

In this section, we describe the details of the proposed method. Loss function is the most important component in an end-to-end dense prediction system since the quality of prediction is most affected by its loss. Thus we first revisit the weighted cross entropy loss used in previous state-of-the-art methods. We then propose our edge detection system, including the loss based on image similarity and the network of bottom-up/top-down structure.

3.1 Revisiting the weighted cross-entropy loss for edge detection

Previously fully convolutional networks (FCN) based edge-detection methods often employ the weighted cross entropy loss as adopted by the HED detector. It is well known that the cross-entropy loss is used to solve binary classification. However, the edge/non-edge pixels are of a highly imbalanced distribution (the majority of pixels are non-edge) thus the direct use of the cross-entropy loss would fail to train the network. To tackle the issue, HED uses the weighted cross entropy loss, which writes

$$\mathcal{L}(W, w) = -\beta \sum_{j \in Y_+} \log \Pr(y_j = 1 | X; W, w) - (1 - \beta) \sum_{j \in Y_-} \log \Pr(y_j = 0 | X; W, w), \tag{1}$$

where Y_+ and Y_- denotes the edge pixel set and non-edge pixel sets, respectively. $\beta = |Y_-|/|Y|$ and $1 - \beta = |Y_+|/|Y|$. X is the input image and $\Pr(y_j | X; W, w)$ is computed using softmax on the classification scores at pixel y_j.

The class-balance weights β and $1 - \beta$ are used to preserve and suppress the losses from the class of edge pixels and non-edge pixels, respectively. This simple solution helps CNN manage to better train the network. We perform a comparison test that uses the standard cross-entropy loss and the weighted loss on the same HED network structure, demonstrating the effectiveness of the weight β. The result of the test is shown in Figure 2. As we can see, the standard loss fails to train the network since the result (Figure 2b) is not an edge map but

an 'embossed' image. However, if we carefully look at its details, we are able to find the contour of rhinoceros that has reasonable thickness and is thinner than the results of the weighted loss (Figure 2c). It is likely to indicate that, although the class-balance weights β and $1-\beta$ manage to make CNNs successfully trained, they cause the 'thickness' issue. This finding explains why recent methods such as RCF and Casenet tend to output overly thick edges. These two methods have employed the cross-entropy loss with the same strategy, i.e., setting weights on edge/non-edge pixels to balance the loss. To make the network trainable and output a crisp prediction at the same time, we will need alternative solutions.

3.2 The proposed loss function for edge detection

We have shown that a distinct characteristic of edge map is that the data is highly biased because the vast majority of the pixels are non-edges. This highly biased issue would cause the learning to fail to find the crisp edges which are the 'rare events'.

Similar to our task, many applications such as fraud detection, medical image processing, and text classification are dealing with class imbalance data and there are corresponding solutions to these tasks [32–36]. Inspired by the work of [37] using the Dice coefficient [38] to solve the class-imbalance problem, we propose to use the Dice coefficient for edge detection.

Given an input image I and the ground-truth G, the activation map M is the input image I processed by a fully convolutional network F. Our objective is to obtain a prediction P. Our loss function L is given by

$$L(P, G) = Dist(P, G) = \frac{\sum_i^N p_i^2 + \sum_i^N g_i^2}{2\sum_i^N p_i g_i}, \tag{2}$$

where p_i, g_i denote the value of i-th pixel on the prediction map P and the ground-truth G, respectively. The prediction map P is computed from the activation map M by the sigmoid function. The loss function L is the reciprocal of the Dice coefficient. Since the Dice coefficient is a measure of similarity of two sets. Our loss is to compare the similarity of two sets P, G and minimizes their distance on the training data. We do not need to consider the issue of balancing the loss of edge/non-edge pixels by using the proposed loss and are able to achieve our objective—make the network trainable and predict crisp edges at the same time.

We should emphasize the way of computing our total loss in a mini-batch. Given a mini-batch of training samples and their corresponding ground-truth, our total loss is given by

$$L(MP, MG) = \sum_i^M Dist(MP_i, MG_i), \tag{3}$$

where MP and MG denote a mini-batch of predictions and their ground-truth, respectively. M is the total number of training samples in the mini-batch. Since

our loss function is based on the similarity of per image-ground-truth pair, our total loss of a mini-batch is the sum of the total distances over all pairs.

To achieve better performance, we propose to combine the cross-entropy loss and the proposed Dice loss. The Dice loss may be thought of as image-level that focuses on the similarity of two sets of image pixels. The cross-entropy loss concentrates on the pixel-level difference, since it is the sum of the distance of every corresponding pixel-pair between prediction and the ground-truth. Therefore the combined loss is able to hierarchically minimize the distance from image-level to pixel-level.

Our final loss function is given by:

$$L_{\text{final}}(P, G) = \alpha L(P, G) + \beta L_c(P, G), \tag{4}$$

where $L(P, G)$ is Equation 2; $L_c(P, G)$ is the normal cross-entropy loss which is $L_c(P, G) = -\sum_j^N (g_j \log p_j + (1 - g_j)(1 - \log p_j))$. N is the total pixel number of an image. α and β are the parameters to control the influence of two losses. In experiments we set $\alpha = 1$ and $\beta = 0.001$. We also tried to use the weighted cross-entropy loss (Equation 1) instead of L_c, and no improvement is observed. To compute the total loss in a mini-batch, we use Equation 3 where $Dist(P, G)$ is replaced by $L_{\text{final}}(P, G)$. We emphasize that the proposed Dice loss $L(P, G)$ is the cornerstone for generating crisp edges. Using only the proposed Dice loss, we achieve an ODS F-score of .805 on the BSDS500 dataset.

The formulation (4) can be differentiated yielding the gradient

$$\frac{\partial L_{\text{final}}}{\partial p_k} = \alpha \frac{2p_k \sum_{i=1}^N p_i g_i - g_k(\sum_{i=1}^N p_i^2 + \sum_{i=1}^N g_i^2)}{2(\sum_{i=1}^N p_i g_i)^2} - \beta \frac{2g_k - 1}{p_k} \tag{5}$$

computed with respect to the k-th pixel of the prediction.

In the next subsection, we describe our network structure.

3.3 Network architecture

We attempt to select the network structure which has multiple stages to efficiently capture hierarchical features and is able to fuse the features of different levels, so as to generate semantically meaningful contours. The success of HED shows the great importance of a carefully designed structure. In this paper, we look at another advanced structure, that is the bottom-up/top-down architecture [39] for inspiration to make better use of hierarchical features. The method of [39] achieves improved accuracy of object segmentation by proposing a novel top-down refinement approach. We hypothesize that this structure may also work for edge detection well since our task is related to object segmentation.

We follow the setting of the network [39] to apply the VGG-16 model [30] as the backbone and stack its 'refactored' structure of the refinement module to recover the resolution of features. However, we have the following modifications at the refinement module to make it suitable for edge detection: (i) to better extract side feature from each stage of VGG-16, we use the *ResNeXt* [40] block

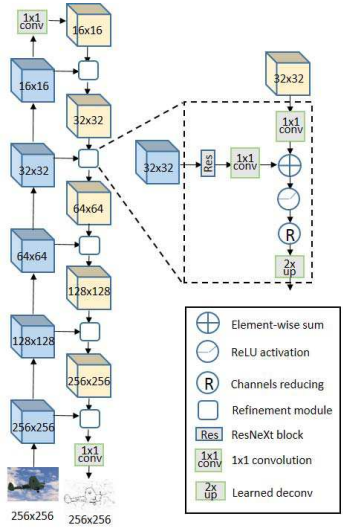


Fig. 3: Overview of the proposed network. Blue cubics indicate the features on the bottom-up path while yellow cubics indicate the mask-encoding on the top-down path. The backbone of our network is the VGG-16 model in which the last pooling layer and all the fully connected layers are removed. The mask-encoding from conv5_3 layer repeatedly goes through the proposed refinement module to recover its resolution. In a refinement module, the mask-encoding is fused with the side-output features and then reduces its channels by a factor of 2 and double its resolution to prepare for the fusion in the next refinement module.

to connect each side output layer, respectively conv1_2, conv2_2, conv3_3, conv4_3 and conv5_3. Thus, the feature from each side output first goes through a *ResNeXt* block then is fused with the mask-encoding from the top-down path; (ii) we use 1×1 *conv* layer to replace original 3×3 *conv* layers of the module. By doing so, we find the performance is improved with the decrease of model complexity; (iii) we use the learned *deconv* layer to double the resolution of fused features. Especially, the *deconv* layer is grouped. The group number equals to the channel number of the fused features. The grouped *deconv* layer allows our model to keep the performance with less model complexity. The modified refinement module is fully back-propable. We show the overall structure in Figure 3 and our refinement module in the dotted rectangle.

Our network is simple yet very effective for edge detection. We highlight that it is vital for an edge-detection network to increase the ability of feature extraction with the decrease of model complexity. Compared to the original structure, our network has the advantage of using fewer parameters to achieve better performance. To be more specific, our network has 15.69M parameters and achieves an ODS of .808 on the BSDS500 dataset. Without the modifications

described in (ii) and (iii), the parameter number increases to 22.64M but the performance decreases to ODS of .802.

The reason behind this phenomenon might be due to overfitting, as the dataset for edge detection has limited number of training samples (for example, the BSDS500 dataset only has only 200 training images). In experiments, we tried a few more sophisticated bottom-up/top-down networks such as Refinenet [41], but failed to achieve better performance possibly because of limited training data. Using the *ResNeXt* block is for the same reason. It groups the inside *conv* layers to decrease the model complexity. We also test the *ResNet* block [42] to extract the side features, which is used to compare the performance against the *ResNeXt* block. We find that they are both helpful to boost the performance while the *ResNeXT* block performs slightly better with roughly 50% complexity of the *ResNet* block.

4 Experiments

In this section, we first present the implementation details as well as a brief description of the datasets. Our experiments start with an ablation study of the proposed method. We then conduct a comparative study on HED to demonstrate the effectiveness of the proposed loss on the crispness of prediction. We further compare our method with the state-of-the-art edge detectors and demonstrate the advantages.

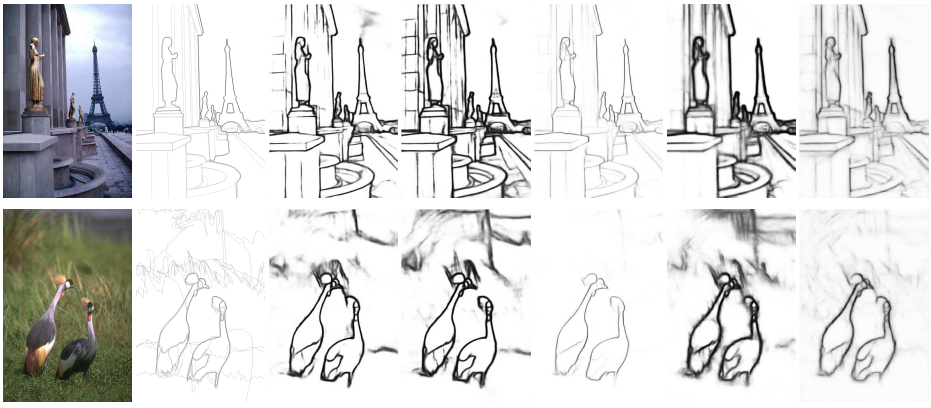
4.1 Implementation details

We implement our method using Pytorch [43]. We evaluate edge detectors on Berkeley Segmentation Dataset (BSDS 500) and NYU depth dataset (NYUD), which are widely used in previous works [10–13, 17]. The hyper-parameters of our model include: mini-batch size (36), input image resolution (480×320), weight-decay (1e−4), training epochs (30). We use the ADAM solver [44] for optimization.

Beside the hyper-parameters, the following several key issues are worth mentioning:

Data augmentation Data augmentation is an effective way to boost performance when the amount of training data is limited. We first randomly scale the image-label pairs (0.7 to 1.3). We then rotate the pairs to 16 different angles and crop the largest rectangle in the rotated angle. We finally flip the cropped images, which leads to an augmented training set from 200 images to more than 100k images.

Up-sampling method We employ learned deconvolution in the backward-refining pathway to progressively increase the resolution of feature maps. Although bilinear upsampling was demonstrated useful in HED, it is abandoned in our method. We observe in experiments that bilinear upsampling may make the prediction discontinuous at a number of locations and cause a slight decrease in performance.



(a) (b) (c) (d) (e) (f) (g)

Fig. 4: Illustration of the qualitative results of the ablation study as well as applying the proposed loss with HED. From left to right: (a) input images in the BSDS500 dataset; (b) ground-truth; (c),(d),(e) are the predictions of the methods *Ours-w/o-rN-w/o-FL, Ours-w/o-FL* and *Ours* in the ablation study, respectively; (f), (g) are the predictions of the methods *HED-BL* and *HED-FL* in the comparative study, respectively. Our method, especially the proposed loss, shows a clear advantage in generating sharp boundaries.

Multi-scale edge detection Inspired by the works [13, 45], during testing we make use of multi-scale edge detection to further improve performance. We first resize an input image to three different resolutions ($0.5\times$, $1.0\times$ and $1.5\times$ of the original size), which are fed into the network. We then resize the outputs back to the original size and average them to obtain the final prediction.

4.2 BSDS500 dataset

Berkeley Segmentation Dataset (BSDS 500) [5] contains 200 training images, 100 validation images, and 200 testing images. Each image is annotated by multiple users. We use the train set (200 training images) for training and employ all the ground-truth labels to prepare the training data. That is, if an image has five annotations, we first create five copies of the image. Each copy is corresponding to one of the annotations, respectively. We then apply these five image-annotation pairs for data augmentation. This would introduce ambiguity in the ground-truth pairs because different annotators may disagree with each other for a small number of pixels. However, in this case, we are able to get more annotations for data augmentation. In the meantime, we observed that introducing certain ambiguity prevented the training from overfitting.

Ablation study We first conduct a series of ablation studies to evaluate the importance of each component in the proposed method. Our first experiment is

Table 1: Ablation studies of the proposed method on the BSDS500 dataset. NMS stands for non-maximum suppression. 'Ours-w/o-FL' refers to our method without the fusion loss. 'w/o-rN' refers to without all the ResNeXt blocks in the backward-refining path.

Method	ODS (after/before NMS)	OIS (after/before NMS)
Ours-w/o-rN-w/o-FL	.797 / .671	.815 / .678
Ours-w/o-FL	.798 / .674	.815 / .679
Ours	.800 /.693	.816 /.700

Table 2: Comparative studies of HED. HED-BL refers to HED trained by the balanced cross-entropy loss. HED-FL refers to HED trained via the proposed fusion loss.

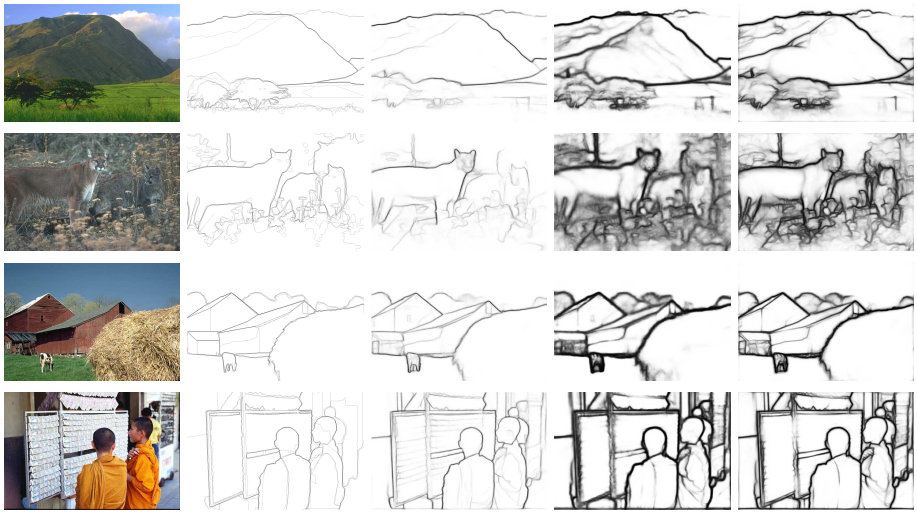
Method	ODS (after/before NMS)	OIS (after/before NMS)
HED-BL	.781 / .583	.798/ .598
HED-FL	.783 / .635	.802 / .644

to examine the effectiveness of the basic encoder-decoder network (Ours-w/o-rN-w/o-FL) for the task. To this end, our baseline model is the proposed network removing all the ResNeXt blocks in the backward-refining path. We train this baseline using the balanced cross-entropy loss. Moreover, we trained two versions of the proposed network via the balanced cross-entropy loss (Ours-w/o-FL) and the proposed fusion loss (Ours), respectively.

The accuracy of prediction is evaluated via two standard measures: fixed contour threshold (ODS) and per-image best threshold (OIS).

Previous works tend to only examine the correctness of prediction since they apply a standard non-maximal suppression (NMS) to predicted edge maps before evaluation. While in this study and the following comparative study, we would like to evaluate each model twice (before and after NMS). By doing so, we can examine both the correctness and the sharpness since low-crispness prediction is prone to achieve low ODS scores, without the aid of NMS. We are aware that CED [45] and PMI [46] apply a different way to benchmark the crispness of predictions by varying a matching distance parameter. However, we consider that directly evaluating non-NMS results is simpler yet effective for the same purpose.

The quantitative results are listed in Table 1 and two qualitative examples are shown in Figure 4(c), (d) and (e). From the results, we observe three findings. Firstly, each component is able to improve performance; Secondly, a convolutional encoder-decoder network may be more competent for the task, compared to the network of HED. We can see that the baseline (Ours-w/o-rN-w/o-FL) achieves an ODS score .797, which significantly outperforms HED (.790 on the BSDS500 dataset). Lastly, both the quantitative and the qualitative results have demonstrated the effectiveness of the proposed fusion loss. By simply using the proposed fusion loss, the ODS f-score (before NMS) of our network is increased



(a) Input Image (b) GT (c) Ours (d) RCF [13] (e) CED [45]

Fig. 5: State-of-the-art comparisons on BSDS500. From left to right: (a) the original images, (b) ground-truth, (c) the predictions of the proposed method, (d) the results of the RCF detector, (e) the results of the CED detector. Note that all the predictions are ent-to-end outputs and not postprocessed.

from .674 to .693 and the improvement of boundary sharpness can be also observed in Figure 4(d) and (e).

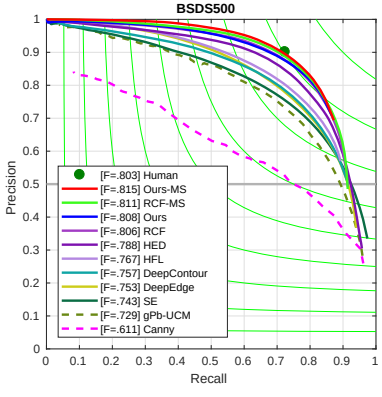


Fig. 6: Precision/recall curves on the BSDS500 dataset. Our method ahieves the best result (ODS=.815).

Table 3: Results on the BSDS500 dataset. MS refers to the multi-scale testing. VOC-aug refers to training with extra PASCAL VOC context data. † refers to GPU time.

Method	ODS	OIS	FPS
Canny [19]	.611	.676	28
gPb-UCM [5]	.729	.755	1/240
SE [26]	.743	.763	2.5
DeepContour [11]	.757	.776	1/30†
DeepEdge [10]	.753	.772	1/1000†
HFL [47]	.767	.788	5/6†
HED [12]	.788	.808	30†
CEDN [17]	.788	.804	10
MIL+G-DSN+MS+NCuts [27]	.813	.831	1
RCF-VOC-aug [13]	.806	.823	30†
RCF-MS-VOC-aug [13]	.811	.830	10†
CED [45]	.794	.811	30†
CED-MS [45]	.803	.820	10†
CED-MS-VOC-aug [45]	**.815**	.833	10†
Ours	.800	.816	30†
Ours-VOC-aug	.808	.824	30†
Ours-MS-VOC-aug	**.815**	**.834**	10†

Improving the crispness of HED As mentioned in Section 3.2, the proposed fusion loss plays a key role in our method in terms of generating sharp bound-

aries, which was demonstrated in the ablation study. One may ask a question: *Does the fusion loss only work on the convolutional encoder-decoder network? Could it also allow different methods such as HED to improve crispness?* To answer this question, we perform a comparative study on the HED edge detector. Similar to the ablation experiments, we evaluate two versions of HED: one is trained by means of the proposed fusion loss, the other is applying the balanced cross-entropy loss. Both methods are trained using deep supervision. Note that our training data of BSDS500 is generated in a different way, compared to HED [31], thus the performance of the re-implemented HED is slightly different from the original paper. We summarize the quantitative results in Table 2 and show two qualitative examples in Figure 4(f) and (g).

The results are consistent with those of the ablation experiments. With the use of the proposed loss, *HED-FL* improves the non-NMS results over *HED-BL* by almost 9%, which is a significant increase at boundary crispness.

State-of-the-art comparisons In this subsection, we further compare against the top performing edge detectors. The methods to be evaluated are composed of two classes: the first class is deep-learning based, which includes HED [12], RCF [13], DeepContour [11], DeepEdge [10], CED [45], HFL [47], CEDN [17], MIL+G-DSN+MS+NCuts [27] and our method; the second class contains SE [26], gPb-UCM [5] and the Canny detector [19]. We also follow the works of [17, 13, 27, 45] to employ the extra training data from PASCAL VOC Context dataset [48]. The results are shown in Figure 5, Figure 6 and Table 3.

We first look at the qualitative result in Figure 5. RCF and CED are the leading edge detectors at present. Especially, CED shares the same aim with our method, which is to solve the issue of boundary crispness. Comparing to the other methods, our approach shows a clear advantage in the quality of edge maps which are cleaner and sharper. Consider the 'cow' in the third example. Our method is able to precisely match its contour, whereas RCF and CED incur much more blurry and noisy edges. The qualitative comparisons suggest that our method generates sharper boundaries.

The quantitative results are summarized in Table 3. Figure 6 shows Precision-Recall curves of all methods. Note that, all the results have been post processed (using NMS) before evaluation. Without extra training data and the multi-scale testing, our method already outperforms most of the state-of-the-art edge detectors. By means of extra training data, our single-scale model achieves a significant improvement on ODS f-score from .800 to .808. With the multi-scale testing, our method achieves the same top performance with CED. However, CED adopted both the train and validation set for training while we only use the train set.

In addition to this, we evaluate the non-NMS results of CED (single-scale, without extra training data) and obtain the performance of ODS f-score of .655, OIS f-score of .662. The result is far behind our single-scale non-NMS performance (ODS f-score of .693). Another advantage of our method is that our detector is able to run in real time. The single scale detector can operate at 30F-

(a) Input Image (b) GT (c) Ours (d) RCF [13]

Fig. 7: State-of-the-art comparisons on NYUDv2. From left to right: (a) is the original image, (b) is the groundtruth, (c) is the prediction of the proposed method, (d) is the result of the RCF detector. Note that the predictions of RCF and the proposed method are trained only on the RGB data. No postprocessing is applied.

PS on a GTX980 GPU. Since our method is simple, effective and very fast, it is easy to be used along with high-level vision tasks such as image segmentation.

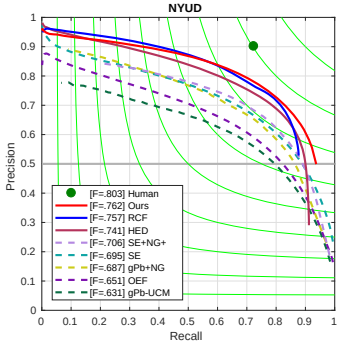


Fig. 8: Precision/recall curves on the NYUDv2 dataset. Our method trained on the RGB data and the H-HA feature achieves the best result (ODS=.762).

Table 4: Results on the NYUDv2 dataset. † means GPU time.

Method	ODS	OIS	FPS
OEF [49]	.651	.667	1/2
gPb-UCM [5]	.631	.661	1/360
gPb+NG [50]	.687	.716	1/375
SE [26]	.695	.708	5
SE+NG+ [51]	.706	.734	1/15
HED-RGB [12]	.720	.734	20†
HED-HHA [12]	.682	.695	20†
HED-RGB-HHA [12]	.746	.761	10†
RCF-RGB [13]	.729	.742	20†
RCF-HHA [13]	.705	.715	20†
RCF-RGB-HHA [13]	.757	.771	10†
Ours-RGB	.739	.754	30†
Ours-HHA	.707	.719	30†
Ours-RGB-HHA	**.762**	**.778**	15†

4.3 NYUDv2 dataset

The NYU depth dataset [14] is a large depth benchmark for indoor scenes, which is collected by a Microsoft Kinect sensor. It has a densely labeled dataset (every pixel has a depth annotation) which has 1449 pairs of aligned RGB and depth images. Gupta *et al.* [50] processed the data to generate edge annotation and split the dataset into 381 training images, 414 validation images, and 654 testing images. We follow their data-split setting and change several hyper-parameters of our method for training: mini-batch size (26), image resolution (480×480). The maximum tolerance allowed for correct matches of edge prediction in evaluation is increased from .0075 to .011, as used in [31, 13, 26]. We compare against the state-of-the-art methods which include OEF [49], gPb-UCM [5], gPb+NG [50], SE [26], SE+NG+ [51], HED [12] and RCF [13].

Motivated by the previous works [12, 13], we leverage the depth information to improve performance. We employ the HHA feature [51] in which the depth information is encoded into three channels: horizontal disparity, height above ground, and angle with gravity. The way of employing the HHA feature is s-traightforward. We simply train two versions of the proposed network, one on the RGB data, another on HHA feature images. The final prediction is generated by directly averaging the output of the RGB model and HHA model.

We show the quantitative results in Table 4 and the precision-recall curve in Figure 8. Our method achieves the best performance of ODS F-score .762. The qualitative results in Figure 7 show consistent performance with those of the experiments on BSDS 500. Our prediction produces sharper boundaries against the leading competitor RCF, which demonstrates the effectiveness of our method.

5 Conclusions

In this work, we have presented a simple yet effective method for edge detection which achieves state-of-the-art results. We have shown that it is possible to achieve excellent boundary detection results using a carefully designed loss function and a simple convolutional encoder-decoder network.

In future work, we plan to extend the use of the edge detector to the tasks like object detection and optical flow which have the requirement of boundary sharpness and a fast processing speed.

Acknowledgement

This work is funded by the China Scholarship Council (Grant No.201506370087), the National Natural Science Foundation of China (Grant No.61572527, Grant No.61628211, Grant No.61602524).

References

1. Marr, D., Hildreth, E.: Theory of edge detection. Proceedings of the Royal Society of London B: Biological Sciences **207**(1167) (1980) 187–217

2. Gonzalez, R.C., Wood, R.E.: Digital image processing, 2nd edtn
3. Torre, V., Poggio, T.A.: On edge detection. IEEE Transactions on Pattern Analysis and Machine Intelligence (2) (1986) 147–163
4. Senthilkumaran, N., Rajesh, R.: Edge detection techniques for image segmentation–a survey of soft computing approaches. International journal of recent trends in engineering **1**(2) (2009) 250–254
5. Arbelaez, P., Maire, M., Fowlkes, C., Malik, J.: Contour detection and hierarchical image segmentation. IEEE transactions on pattern analysis and machine intelligence **33**(5) (2011) 898–916
6. Chen, L.C., Barron, J.T., Papandreou, G., Murphy, K., Yuille, A.L.: Semantic image segmentation with task-specific edge detection using cnns and a discriminatively trained domain transform. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. (2016) 4545–4554
7. Bertasius, G., Shi, J., Torresani, L.: Semantic segmentation with boundary neural fields. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. (2016) 3602–3610
8. Ren, X.: Local grouping for optical flow. In: Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on, IEEE (2008) 1–8
9. Revaud, J., Weinzaepfel, P., Harchaoui, Z., Schmid, C.: Epicflow: Edge-preserving interpolation of correspondences for optical flow. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. (2015) 1164–1172
10. Bertasius, G., Shi, J., Torresani, L.: Deepedge: A multi-scale bifurcated deep network for top-down contour detection. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. (2015) 4380–4389
11. Shen, W., Wang, X., Wang, Y., Bai, X., Zhang, Z.: Deepcontour: A deep convolutional feature learned by positive-sharing loss for contour detection. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. (2015) 3982–3991
12. Xie, S., Tu, Z.: Holistically-nested edge detection. In: Proceedings of the IEEE international conference on computer vision. (2015) 1395–1403
13. Liu, Y., Cheng, M.M., Hu, X., Wang, K., Bai, X.: Richer convolutional features for edge detection. arXiv preprint arXiv:1612.02103 (2016)
14. Nathan Silberman, Derek Hoiem, P.K., Fergus, R.: Indoor segmentation and support inference from rgbd images. In: ECCV. (2012)
15. Sobel, I.: Camera models and machine perception. Technical report, Stanford Univ Calif Dept of Computer Science (1970)
16. Yu, Z., Feng, C., Liu, M.Y., Ramalingam, S.: Casenet: Deep category-aware semantic edge detection. ArXiv e-prints (2017)
17. Yang, J., Price, B., Cohen, S., Lee, H., Yang, M.H.: Object contour detection with a fully convolutional encoder-decoder network. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. (2016) 193–202
18. Kittler, J.: On the accuracy of the sobel edge detector. Image and Vision Computing **1**(1) (1983) 37–42
19. Canny, J.: A computational approach to edge detection. IEEE Transactions on pattern analysis and machine intelligence (6) (1986) 679–698
20. Fram, J.R., Deutsch, E.S.: On the quantitative evaluation of edge detection schemes and their comparison with human performance. Computers IEEE Transactions on **C-24**(6) (1975) 616–628
21. Perona, P., Malik, J.: Scale-space and edge detection using anisotropic diffusion. IEEE Transactions on pattern analysis and machine intelligence **12**(7) (1990) 629–639

22. Lowe, D.G.: Distinctive image features from scale-invariant keypoints. International Journal of Computer Vision **60**(2) (2004) 91–110
23. Siddiqui, M., Medioni, G.: Human pose estimation from a single view point, real-time range sensor. In: Computer Vision and Pattern Recognition Workshops (CVPRW), 2010 IEEE Computer Society Conference on, IEEE (2010) 1–8
24. Martin, D.R., Fowlkes, C.C., Malik, J.: Learning to detect natural image boundaries using local brightness, color, and texture cues. IEEE transactions on pattern analysis and machine intelligence **26**(5) (2004) 530–549
25. Dollar, P., Tu, Z., Belongie, S.: Supervised learning of edges and object boundaries. In: Computer Vision and Pattern Recognition, 2006 IEEE Computer Society Conference on. Volume 2., IEEE (2006) 1964–1971
26. Dollár, P., Zitnick, C.L.: Fast edge detection using structured forests. IEEE transactions on pattern analysis and machine intelligence **37**(8) (2015) 1558–1570
27. Kokkinos, I.: Pushing the boundaries of boundary detection using deep learning. arXiv preprint arXiv:1511.07386 (2015)
28. Krizhevsky, A., Sutskever, I., Hinton, G.E.: Imagenet classification with deep convolutional neural networks. In: Advances in neural information processing systems. (2012) 1097–1105
29. LeCun, Y., Boser, B.E., Denker, J.S., Henderson, D., Howard, R.E., Hubbard, W.E., Jackel, L.D.: Handwritten digit recognition with a back-propagation network. In: Advances in neural information processing systems. (1990) 396–404
30. Simonyan, K., Zisserman, A.: Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556 (2014)
31. Xie, S., Tu, Z.: Holistically-nested edge detection. International Journal of Computer Vision **125**(1-3) (2017) 3–18
32. Gu, J., Zhou, Y., Zuo, X.: Making class bias useful: A strategy of learning from imbalanced data. Intelligent Data Engineering and Automated Learning-IDEAL 2007 (2007) 287–295
33. Tang, L., Liu, H.: Bias analysis in text classification for highly skewed data. In: Data Mining, Fifth IEEE International Conference on, IEEE (2005) 4–pp
34. Lusa, L., et al.: Class prediction for high-dimensional class-imbalanced data. BMC bioinformatics **11**(1) (2010) 523
35. Haider, A.H., Schneider, E.B., Sriram, N., Dossick, D.S., Scott, V.K., Swoboda, S.M., Losonczy, L., Haut, E.R., Efron, D.T., Pronovost, P.J., et al.: Unconscious race and class bias: its association with decision making by trauma and acute care surgeons. Journal of Trauma and Acute Care Surgery **77**(3) (2014) 409–416
36. Phillips, S.J., Dudík, M.: Generative and discriminative learning with unknown labeling bias. In: Advances in Neural information Processing Systems. (2009) 401–408
37. Milletari, F., Navab, N., Ahmadi, S.A.: V-net: Fully convolutional neural networks for volumetric medical image segmentation. In: 3D Vision (3DV), 2016 Fourth International Conference on, IEEE (2016) 565–571
38. Dice, L.R.: Measures of the amount of ecologic association between species. Ecology **26**(3) (1945) 297–302
39. Pinheiro, P.O., Lin, T.Y., Collobert, R., Dollár, P.: Learning to refine object segments. In: European Conference on Computer Vision, Springer (2016) 75–91
40. Xie, S., Girshick, R., Dollr, P., Tu, Z., He, K.: Aggregated residual transformations for deep neural networks. (2016)
41. Lin, G., Milan, A., Shen, C., Reid, I.: Refinenet: Multi-path refinement networks for high-resolution semantic segmentation. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR). (2017)

42. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: Proceedings of the IEEE conference on computer vision and pattern recognition. (2016) 770–778
43. Adam Paszke, Sam Gross, S.C., Chanan, G.: Pytorch (2017)
44. Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization. CoRR **abs/1412.6980** (2014)
45. Wang, Y., Zhao, X., Huang, K.: Deep crisp boundaries. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. (2017) 3892–3900
46. Isola, P., Zoran, D., Krishnan, D., Adelson, E.H.: Crisp boundary detection using pointwise mutual information. In: European Conference on Computer Vision, Springer (2014) 799–814
47. Bertasius, G., Shi, J., Torresani, L.: High-for-low and low-for-high: Efficient boundary detection from deep object features and its applications to high-level vision. In: Proceedings of the IEEE International Conference on Computer Vision. (2015) 504–512
48. Mottaghi, R., Chen, X., Liu, X., Cho, N.G., Lee, S.W., Fidler, S., Urtasun, R., Yuille, A.: The role of context for object detection and semantic segmentation in the wild. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. (2014) 891–898
49. Hallman, S., Fowlkes, C.C.: Oriented edge forests for boundary detection. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. (2015) 1732–1740
50. Gupta, S., Arbelaez, P., Malik, J.: Perceptual organization and recognition of indoor scenes from rgb-d images. In: Computer Vision and Pattern Recognition (CVPR), 2013 IEEE Conference on, IEEE (2013) 564–571
51. Gupta, S., Girshick, R., Arbeláez, P., Malik, J.: Learning rich features from rgb-d images for object detection and segmentation. In: European Conference on Computer Vision, Springer (2014) 345–360