Learning Relaxed Deep Supervision for Better Edge Detection

Yu Liu and Michael S. Lew
LIACS Media Lab, Leiden University, The Netherlands
{y.liu, m.s.lew}@liacs.leidenuniv.nl

Abstract

*We propose using relaxed deep supervision (RDS) within convolutional neural networks for edge detection. The conventional deep supervision utilizes the general ground-truth to guide intermediate predictions. Instead, we build hierarchical supervisory signals with additional relaxed labels to consider the diversities in deep neural networks. We begin by capturing the relaxed labels from simple detectors (*e.g. *Canny). Then we merge them with the general ground-truth to generate the RDS. Finally we employ the RDS to supervise the edge network following a coarse-to-fine paradigm. These relaxed labels can be seen as some false positives that are difficult to be classified. We consider these false positives in the supervision, and are able to achieve high performance for better edge detection. We compensate for the lack of training images by capturing coarse edge annotations from a large dataset of image segmentations to pretrain the model. Extensive experiments demonstrate that our approach achieves state-of-the-art performance on the well-known BSDS500 dataset (ODS F-score of .792) and obtains superior cross-dataset generalization results on NYUD dataset.*

1. Introduction

Edge detection, which aims to extract the important edges from images, has served as a fundamental task in the computer vision community for several decades. Typically, edge detection is considered as a low-level problem, and it is frequently used for high-level vision applications, such as object detection [12] and segmentation [1]. Most of the traditional edge detection approaches [4, 11, 6, 38, 23, 18, 15, 34] extract discriminative local features with color and gradient clues, such as gPb [1], Sketch tokens [24], and Structured Edges (SE) [8].

Although learning various low-level features does well in detecting edges, many researchers have begun to take advantage of object-level or high-level features to investigate how humans perceive edges [3]. Inspired by the substantial successes from deep learning and convolutional neural networks (CNNs), such as image-level classification [21, 33, 37], object-level detection [14, 16], pixel-level labeling [10, 27], and other tasks [36, 26], more and more works tend to transfer deep features from high-level vision tasks to low-level problems. Specifically, recent developments in the design of edge features are moving from carefully-engineered descriptors to hierarchical deep features. This line of works includes DeepNet [20], CSCNN [17], N4-Fields [13], DeepEdge [2], HFL [3], DeepContour [32], HED [39] and so on.

One difficulty in edge detection is attributed to the false positives, that is, many non-edge pixels are incorrectly predicted as edges as compared with the human annotated ground-truth. To alleviate this issue, HED [39] imposed general supervision (i.e. the annotated ground-truth) in the intermediate layers, and therefore the false positives could be corrected earlier. However, using only one general supervision ignores the network *diversities*: diverse representations of hierarchical layers. In addition, the general supervision can not be well-suited to all intermediate layers. Hence, how to explore diverse supervision that can adapt to all of the intermediate layers or hierarchical diversities for edge detection?

In this paper, we propose using diverse deep supervision that can vary from coarse level to fine level as deep features become more discriminative. Our diverse supervision is called relaxed deep supervision (RDS). RDS consists of additional relaxed labels, apart from the positive labels (edge points) and negative labels (non-edge points). These relaxed labels are used to adapt to the diversities of intermediate layers. Briefly speaking, we capture the relaxed labels from simple yet efficient off-the-shelf detectors, for example Canny [4] or SE [8]. Then, we insert the extracted relaxed labels into the original ground-truth to generate RDS. Finally, RDS can guide intermediate layers in a coarse-to-fine paradigm. That is, RDS can process the false positives using a "delayed strategy", in which the loss cost of the relaxed labels are ignored in current supervision, and then will be reconsidered in the next supervision. As a result, more discriminative layers are assigned to process more false positives (difficult points). In summary, RDS can

not only maintain high performance, but also incorporate network diversities for better edge detection.

Another problem about edge detection is that it requires expensive human annotations, compared to other tasks (*e.g.* image recognition and segmentation). For example the frequently benchmarked BSDS500 dataset [1] has only 200 training images. This small dataset limits the learning ability of various algorithms based on deep learning. To alleviate this deficiency, we generate coarse edge annotations (CEA) from a large collection of segmentation annotations such as the PASCAL Context dataset [29]. Thereby, we pretrain the model with CEA and then fine-tune it with the target dataset (*e.g.* BSDS500).

Our contributions are summarized as follows: (1) We propose relaxed deep supervision to guide the intermediate predictions. Compared with traditional deep supervision, RDS can adapt to the hierarchical diversities with minimal manual effort. (2) We show that pre-training the model with a large collection of CEA is an efficient way to enhance the learning ability of CNNs and thus yields considerable improvements. (3) Despite the apparent simplicity of RDS, our approach achieves state-of-the-art accuracy (ODS=.792) on the well-known benchmark BSDS500 [1]. In addition, our implementation can maintain fast convergence and low time complexity.

2. Related work

Deep learning for edge detection. Recently, edge detection has had significant advances due to the developments of deep features. Figure 1 displays the basic pipeline of current edge detection systems based on deep learning. Based on how to predict edges with deep model, we broadly divide them into three categories.

(1) *Pixel-level prediction*: extract deep feature per pixel and classify it to edge or non-edge class. Early work such as [20] developed a convolutional RBM to learn pixel-level features. Hwang and Liu [17] stacked pixel features in a multiscale CNN model and then fed them to a SVM classifier. Bertasius *et al.* [2] built four CNN models to describe multi-scale features for candidate edge points. Then they improved their network structure with less computational complexity [3].

(2) *Patch-level prediction*: estimate edge maps for the input patches and then integrate them for the whole edge map. For example, the N4-Fields [13] extracted patch features from a pre-trained CNN model, and then mapped them to the nearest neighbor annotation from a pre-built dictionary. Shen *et al.* [32] clustered contour patches for mid-level shape classes and solved the model using a positive-sharing loss function.

(3) *Image-level prediction*: predict the whole edge map end-to-end given one input image. Considering the inefficiency of the above two categories, Xie and Tu [39] proposed a holistically-nested edge detection (HED) approach that was the first attempt to perform holistic image training and prediction for edge detection. Their work took advantage of the high efficiency from end-to-end fully convolutional networks (FCNs) [27], and additional deep supervision from deeply supervised nets (DSN) [22].

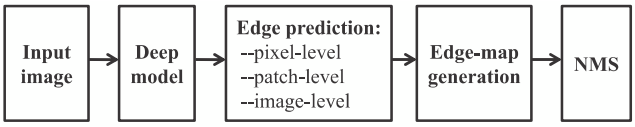


Figure 1: Pipeline of deep learning based edge detection.

Additional deep supervision. Lee *et al.* [22] firstly proposed to train deep neural networks with hidden-layer supervision. They proposed imposing additional supervision to intermediate layers in order to improve the directness and transparency of learning network. Similarly, GoogleNet [37] created two additional side branches and supervised them with a general ground-truth. Sun *et al.* [36] built a robust face recognition network learned with multiple identification-verification supervision.

3. Our approach

Inspired by the advantage of additional deep supervision [22] and its application in edge detection [39], we aim to explore diverse deep supervision to guide CNNs for better edge detection.

3.1. Relaxed deep supervision

Model. Our edge detection architecture is built on top of HED network [39], which is trimmed from the VGG-16 net [33] (See Fig. 2). The network architecture contains five convolutional nets connected with the max-pooling layers. Each convolutional net has several convolutional layers. In order to add deep supervision to guide the intermediate layers, five side-output layers (from side-output 1 to side-output 5) are inserted behind the intermediate layers. Due to the deconvolutional operation, the side-output predictions keep the same spatial size as the input image. In order to integrate multi-scale predictions, one weighted-fusion layer followed by fusion-output prediction is concatenated with five side-output predictions. Notably, HED utilizes the original ground-truth G as a general supervisory signal to guide the whole network, including five side-output predictions and the last fusion-output prediction.

Although the fusion-output prediction in HED is integrated with multi-scale predictions, their general supervision fails to present hierarchical diversities. Instead, our main aim is to explicitly make use of diverse supervision associated with different intermediate layers. To this end, we propose to integrate additional *relaxed labels* into the general supervision, and generate hierarchical and specific supervision, called relaxed deep supervision (RDS). Our

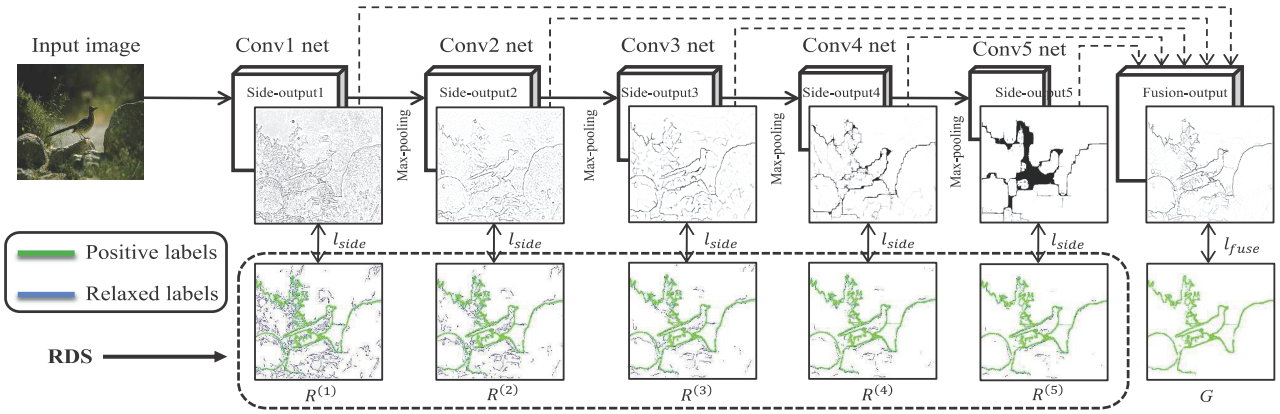


Figure 2: The network architecture with RDS (best viewed in color and zoom-in). The proposed RDS, including positive labels (green color), negative labels (white color for clear visualization), relaxed labels (blue color), is used to supervise the corresponding side-output prediction. The last fusion-output is still supervised by the original ground-truth G. The total loss cost in the network is the sum of all l_{side} and l_{fuse}.

approach stems from the fact that hierarchical layers can represent specific abstracts of the input image [21, 28]. For example in Fig. 2, the bottom side-output predictions (*e.g.* side-output 1, 2) easily detect a large number of small edges and noise. In contrast, the top predictions(*e.g.* side-output 4, 5) can fire stronger responses around the positive labels. However, the general supervision can not be well-suited to all side-output predictions. In contrast, our RDS can not only preserve the strong supervision from the ground-truth, but also allow specific diversities due to the relaxed labels. In the following, we present two simple yet efficient ways to capture the relaxed labels based on off-the-shelf edge detectors, including Canny [4] and SE [8].

Relaxed labels based on Canny detector. The Canny algorithm [4] can detect different scales of edge responses based on the parameter σ, which is the standard deviation of the Gaussian filter. The aforementioned relaxed labels can be extracted from Canny edge responses. First, we need to adjust different scales ($\sigma \in \{1, 3, 5, 7, 9\}$) to obtain various edge responses for five side-output predictions. We do not perform complicated learning algorithm to keep their strong consistency because we intend to maintain high efficiency of the whole approach. We denote these binary edge responses with $\{C^{(k)}\}_{k=1}^{5}$. For example $C^{(3)}$ is the edge response when $\sigma = 5$. Second, for the side-output prediction k, we define its *relaxed labels*: "*belong to the positive labels of $C^{(k)}$, but are not included in the positive labels of the original ground-truth G.*" Thus the relaxed labels can present the complementary clues apart from the ground-truth. Finally, the set of relaxed labels can be computed as follows

$$D^{(k)} = \mathrm{H}(C^{(k)} - C^{(k)} \cap G), \tag{1}$$

where the function H is used to collect the set of positive labels from the input binary map. As shown in Fig. 3, the first row gives three scales of Canny edge responses (both red and blue color) with $\sigma = 1, 5, 9$. We highlight the relaxed labels in blue color, and the red points indicate the overlap edges between $C^{(k)}$ and G. The ground-truth G can be seen in Fig. 2.

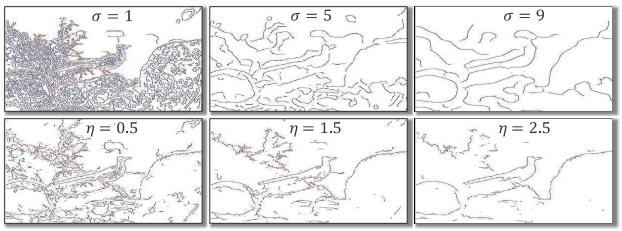


Figure 3: Illustration of extracting relaxed labels (blue color). The first and second rows display three edge responses from Canny [4] and SE [8], respectively.

Relaxed labels based on SE detector. To demonstrate the generalization of our method, we also employ another edge detector based on Structured Edges (SE) [8]. Briefly speaking, SE outputs one edge map with pixel-wise probabilities ranging from 0 to 1. Similarly, we need to create five binary edge responses from the SE edge map. We begin by computing the mean value of edge probabilities in the SE edge map, denoted as v. Then we adjust a threshold t to binarize SE edge map by $t = \eta \cdot v$, where $\eta \in \{0.5, 1.0, 1.5, 2.0, 2.5\}$. As a result, we obtain five binary edge responses, denoted as $\{S^{(k)}\}_{k=1}^{5}$. Following the definition of relaxed labels, we compute the set of relaxed labels based on SE by

$$D^{(k)} = \mathrm{H}(S^{(k)} - S^{(k)} \cap G). \tag{2}$$

The second row in Fig. 3 displays the edge responses from SE and their relaxed labels (blue color). One can observe that the relaxed labels from SE detector are visually sparser than those from Canny detector. We will give more comparison in Sec. 4.2.

RDS generation. It can be observed that various re-

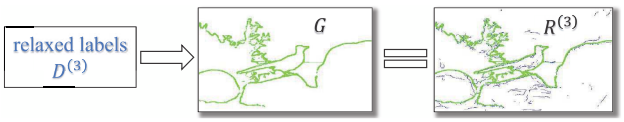


Figure 4: Illustration of generating the RDS (best viewed in zoom-in). $R^{(3)}$ is merged by the set $D^{(3)}$ and G.

laxed labels are well-suited to our needs of highlighting hierarchical diversities within the supervision. In the next stage, we insert the set of relaxed labels into the original ground-truth. This merging operation is used to generate RDS, which is an union of positive, negative, and relaxed labels. Mathematically, we denote five different RDS by $\{R^{(k)}\}_{k=1}^{5}$. The construction step can be seen in Fig. 4. For example the set $D^{(3)}$ is extracted based on $S^{(3)}$. Consequently, the generated $R^{(3)}$ can not only preserve the positive labels in the ground-truth G, but also contain specific relaxed labels. Notably, the relaxed labels are corresponded to the non-edge points in G. These non-edge points are seen as some false positives that are difficult to be predicted. Hence, we intend to use RDS to solve them.

3.2. Training with RDS

In this section we will give the training details with RDS. Assume a training dataset which contains N images: $\{I_i, G_i\}_{i=1}^{N}$, where I_i is the i-th input image and G_i is its binary ground-truth. $I_{i,j}$ denotes the j-th raw pixel over the spatial dimensions of I_i. Assume that we use the relaxed labels derived from SE detector $\left(\{D^{(k)}\}_{k=1}^{K}\right)$. The corresponding RDS are denoted as $\{R_i^{(k)}\}_{k=1}^{K}$, where $K = 5$ in this network. Five different side-output predictions are separately supervised with the corresponding RDS, and the fusion-output prediction is still supervised with the original ground-truth (See Fig. 2). Notably, early supervision (*e.g.* $R^{(1)}, R^{(2)}$) has more relaxed labels than late supervision (*e.g.* $R^{(4)}, R^{(5)}$). This meets the hierarchical characteristics of CNN models. Finally, the total loss function L_{RDS} is formulated as

$$\sum_{i=1}^{N}\sum_{j=1}^{|I_i|}\left(\sum_{k=1}^{K} l_{side}(\widehat{G}_{i,j}^{(k)}, R_{i,j}^{(k)}) + l_{fuse}(\widehat{G}_{i,j}^{fuse}, G_{i,j})\right), \quad (3)$$

where $|I_i|$ is the total number of all pixels in I_i. l_{side} and l_{fuse} represent the loss cost per pixel, from the side-output and fusion-output, respectively. $\widehat{G}_{i,j}^{(k)}$ and $\widehat{G}_{i,j}^{fuse}$ indicates the j-th pixel prediction from the k-th side-output and the fusion-output, respectively. For notational simplicity, the network parameters, such as weights and bias, are not included in the equation. In $R_i^{(k)}$, the relaxed labels are set with 2, different from the positive labels (with 1) and negative labels (with 0). Therefore, we compute l_{side} based on the types of pixel labels as follows

$$l_{side}(\widehat{G}_{i,j}^{(k)}, R_{i,j}^{(k)}) = \begin{cases} \alpha \cdot \log \mathrm{P}(\widehat{G}_{i,j}^{(k)}), & R_{i,j}^{(k)} = 1 \\ \beta \cdot \log(1 - \mathrm{P}(\widehat{G}_{i,j}^{(k)})), & R_{i,j}^{(k)} = 0 \\ 0, & R_{i,j}^{(k)} = 2 \end{cases} \quad (4)$$

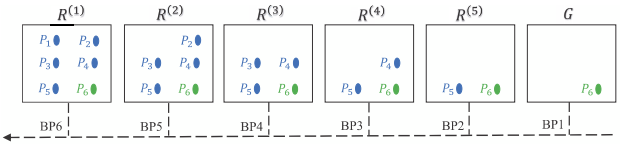


Figure 5: RDS employs a coarse-to-fine supervision strategy. The blue points indicate the relaxed labels, and the green point is one positive label.

where $\mathrm{P}(\widehat{G}_{i,j}^{(k)})$, using sigmoid function, indicates the probability of current pixel being an edge point; α and β are used to balance the biased distribution between edge and non-edge pixels. Since about 90% pixels belong to non-edge class, we set $\alpha = 9\beta$ to enhance the edge class, for instance $\alpha = 9$ and $\beta = 1$. In summary, we compute l_{side} when the pixel has positive or negative label. However, when the pixel has a relaxed label ($R_{i,j}^{(k)} = 2$), we do not compute its loss cost and set $l_{side} = 0$. On the other hand, the computation of l_{fuse} excludes the third term in Eq. (4), because there is no relaxed labels in G_i. Next, we consider the backward propagation (BP). We can deduce the partial derivations of l_{side} w.r.t. $\widehat{G}_{i,j}^{(k)}$ by

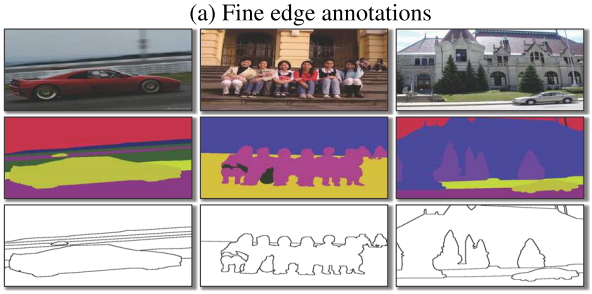
$$\bigtriangledown = \begin{cases} \alpha \cdot (\mathrm{sigmoid}(\widehat{G}_{i,j}^{(k)}) - 1), & R_{i,j}^{(k)} = 1 \\ \beta \cdot \mathrm{sigmoid}(\widehat{G}_{i,j}^{(k)}), & R_{i,j}^{(k)} = 0 \\ 0, & R_{i,j}^{(k)} = 2 \end{cases} \quad (5)$$

For each backward propagation with $R^{(k)}$ or G, we utilize the chain rule to compute partial derivations [5] and update the network parameters based on stochastic gradient descent (SGD) with a mini-batch size [21].

Explanation. Here we will clarify the procedure about how RDS improves edge detection. As mentioned before, one difficult issue in edge detection is attributed to the false positives. The relaxed labels based on Canny/SE actually correspond to some false positives that are difficult to be classified. RDS processes these false positives using a *coarse-to-fine* paradigm: the false positives (with relaxed labels) in current supervision are ignored without computing their loss cost (Equation 4), and then will be reconsidered in the next supervision. Consequently, top layers are assigned to process more false positives due to their high discriminatory power. This procedure is similar with hierarchical object classification [40], in which difficult classes are classified from coarse-category prediction to fine-category prediction.

We clarify this paradigm in Figure 5. In $R^{(1)}$, P_1 serves as a relaxed label that is difficult to be predicted in the side-output 1. Thus we do not compute the loss cost of P_1 and delay its prediction until in $R^{(2)}$. In $R^{(2)}$, P_1 is converted to be a negative label (no-edge), so this provides evidence that the side-output 2 associated with stronger discrimination is able to predict P_1. Similarly, $R^{(5)}$ is able to recognize most relaxed labels except for P_5. Therefore, RDS can incrementally improve the strength

(a) Fine edge annotations



(b) Coarse edge annotations

Figure 6: Comparison between fine and coarse edge annotations. (a) displays three images and their ground-truth from BSDS500 [1]. (b) shows the images, segmentations from Pascal Context [29] in the first and second row. The CEA is extracted in the third row.

of the supervision and assign more false positives to more high-level layers. Moreover, the network is also involved in a *coarse-to-fine* BP procedure: First, the whole network parameters are updated with *coarse* supervision G; Then *fine* supervision $R^{(k)}$ (with specific relaxed labels) is used to fine-tune their local nets. For example the pixel with P_6 is updated by all BP (six times), and P_4 will be updated twice (by G and $R^{(5)}$).

In a nutshell, RDS can benefit the whole training for edge detection. It can reduce the total loss cost in the forward propagation stage and facilitate efficient updates in the backward propagation stage.

3.3. Pretraining with CEA

Generally, collecting more training data can develop the learning ability of CNNs. Many visual recognition tasks have access to large-scale datasets, like ImageNet [31], MS COCO [25] and PASCAL VOC [9]. But fine edge annotations (FEA) require rather expensive human effort. Thus the well-known BSDS500 dataset [1] collects only 200 training images. This small training set limits current edge detection algorithms in improving the performance.

To alleviate this issue, we attempt to extract coarse edge annotations (CEA) from a large collection of segmentation annotations. Here, we utilize the Pascal Context dataset [29], which provides full-scene segmentations for more than 400 classes, and has 10103 train and val images in total. Thus we extract the edges alongside the segmentations. Different from FEA, CEA only provides the outside boundaries of objects (See the car, people and building in Fig. 6), but it can facilitate the network learning due to a large number of images. Notably, there are no overlap

Algorithm 1 RDS: training and testing procedure

1: **Input:** Training dataset; VGG-16 net; training iterations T_1,T_2
2: **Initializing:** network parameters $\mathbf{W}$ using VGG model
3: **Preparation:** for one image I_i, extract the set of relaxed labels $\{D_i^{(k)}\}_{k=1}^5$ and generate RDS $\{R_i^{(k)}\}_{k=1}^5$, Sec. 3.1.
4: **Pre-training:** use Pascal Context data and its CEA, $t=0$
  while $t < T_1$ **do**
    $t \leftarrow t+1$
    Forward propagate to compute L_{CEA} in Eq. (6);
    Backward propagate to get gradients $\Delta\mathbf{W}$, like Eq. (5);
    Update $\mathbf{W}^t = \mathbf{W}^{t-1} - \lambda_t \Delta\mathbf{W}$ with SGD;
  end while
5: **Training:** use the target training data set (*e.g.* BSDS500), $t=0$
  while $t < T_2$ **do**
    $t \leftarrow t+1$
    Forward propagate to compute L_{RDS} in Eq. (3);
    Backward propagate to get gradients $\Delta\mathbf{W}$, like Eq. (5);
    Update $\mathbf{W}^t = \mathbf{W}^{t-1} - \lambda_t \triangle \mathbf{W}$ with SGD;
  end while
6: **Testing:** feed one image into the learned network with parameters $\mathbf{W}$ and output edge map E_i
7: **Post-processing:** non-max suppression on E_i
8: **Output:** final edge map $E_i^{'}$

images between Pascal Context and BSDS500, which are from Flickr and Corel, respectively. While training with CEA, we simply compute the fusion-output loss function and exclude the intermediate supervision by

$$L_{CEA} = \sum_{i=1}^{N}\sum_{j=1}^{|I_i|}\left(l_{fuse}(\hat{G}_{i,j}^{fuse}, G_{i,j})\right). \quad (6)$$

In practice, we pretrain the model with Pascal Context and its CEA according to Eq. (6), and then fine-tune it with the BSDS500 dataset as Eq. (3). In summary, we show the whole algorithm procedure in Algorithm 1, including the training and testing stages.

3.4. Implementation

We implemented our approach with the publicly available Caffe framework [19] and HED implementation [39]. Likewise, the whole network is initialized with the VGG-16 net [33] pretrained on ImageNet dataset [31].

Parameters. We refer to some basic parameters as HED net, including momentum (0.9), weight decay (0.0002), initialization of the side-output filters (0), and initialization of fusion-output filter (0.2). The training images are resized to 400×400 and the batch size is 8. More importantly, we present some different parameters in our experiments. For example, the learning rate is fixed with 1e-9. This learning rate is quite efficient and reducing it during training iterations has no remarkable improvement. The training will be terminated after 25 epoches. Another difference is the class-balanced parameters α and β in Eq. (4). We utilize the fixed class-balanced parameters ($\alpha = 9, \beta = 1$) for all images. On the other hand, we also try to tune σ and η used for computing the relaxed labels in Sec. 3.1. But our

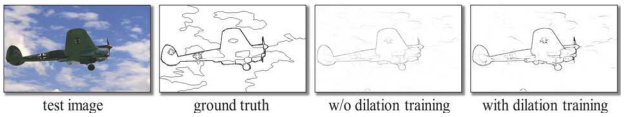


Figure 7: Comparison of edge detection results between without and with ground-truth dilation.

algorithm is relatively insensitive to them.

Ground-truth dilation. Frequently, human subjects annotate the ground-truth edges with thin boundaries (e.g. one pixel width). However, the predicted edges from deep models has rather thick boundaries. To tackle this inconsistency, we dilate the positive labels in the ground-truth of train set using traditional morphologic dilation operator. Figure 7 compares the detection results between without and with dilation training. It can be seen that training with the dilated ground-truth contributes to predicting stronger edge maps. Quantitatively, the dilation process can increase about .02 accuracy (ODS) on BSDS500 test set. Hence, the ground-truth dilation is a quite useful step for edge detection. Note that we do not dilate the test set. In addition, the postprocessing non-maximal suppression (NMS) [4] can be used to thin the predicted edges.

Fast convergence. We also evaluate the convergence of our implementation by analyzing the training loss w.r.t. iterations. It is observed that during the first five epoches, the loss cost reduces quickly. Afterwards, it tends to keep steady until almost convergence. Thereby, our implementation is beneficial for fast convergence.

4. Results

We conduct the majority of experiments on BSDS500 dataset [1], and then verify its cross-dataset generalization on NYUDv2 dataset [30].

4.1. Baselines

To experimentally evaluate the effectiveness and advantage of RDS, we also implemented two baseline methods. (1) *Baseline 1*: only supervises the fusion-output prediction with the general supervision (i.e. original ground-truth). (2) *Baseline 2*: imposes the general supervision to not only the fusion-output prediction, but also five side-output predictions. In Table 1, the Baseline 1 achieves ODS=.762 on BSDS500. Relatively, the Baseline 2 improves the accuracy to ODS=.780. This provides evidence of the benefits of additional intermediate supervision. The performance gap with/without intermediate supervision in HED is less than that of our Baseline1 and Baseline2. The reason is that we do not perform data augmentation (e.g. rotation and flip) that has been employed in HED. Although the data augmentation may decrease the improvement of intermediate supervision, we believe that it should not remove our awareness of its importance.

Table 1: Results on BSDS500 testing set. RDS(Canny) and RDS(SE) derive the relaxed labels from Canny and SE. CEA uses the extra data from Pascal Context dataset.

	ODS	OIS	AP
Baseline 1	.762	.782	.766
Baseline 2	.780	.802	.786
RDS(Canny)	.785	.803	.813
RDS(SE)	.787	.804	.817
RDS(gPb)	.786	.803	.814
CEA	.765	.785	.724
RDS(Canny) + CEA	.790	.809	**.819**
RDS(SE) + CEA	**.792**	**.810**	.818

4.2. BSDS500 results

The BSDS500 dataset [1] consists of 200 training, 100 validation, and 200 testing images. The validation set is used to fine-tune the hyperparameters. Each image is manually annotated by five human annotators on average. For training images, we just preserve their positive labels annotated by at least three human annotators. In testing stage, we extract the fusion-output prediction to evaluate the performance, based on three measures: fixed contour threshold (ODS), per-image best threshold (OIS) and average precision (AP).

Results discussion. Table 1 reports the results regarding our approach. We discuss them from the following aspects.

(1) RDS yields considerable improvements over the general supervision approach (Baseline 2). This verifies the advantage of RDS for incorporating hierarchical diversities. In details, the result of RDS with relaxed labels from Canny, denoted as RDS(Canny), achieves ODS=.785. Furthermore, the RDS(SE) result reaches to ODS=.787.

(2) RDS is relatively insensitive to different choices of relaxed labels. First, we can see that RDS can obtain similar results with Canny and SE. In addition, we use another detector, gPb [1], to capture the relaxed labels. Similarly, its result (ODS=.786) keeps consistent with RDS(Canny) and RDS(SE). Thus we have not invested too much effort in optimizing various relaxed labels now.

(3) Pretraining with CEA demonstrate further gains for both RDS(Canny) and RDS(SE), reaching to ODS=.790 and .792, respectively. In addition, we also evaluate the model pretrained by CEA on the BSDS500 test set (without finetuning on BSDS500 training set). Significantly, it can still achieve ODS=.765. These results evidences the necessity and advantage of a large-scale fruitful dataset.

Table 2: Comparing the importance of early and late supervision on BSDS500 testing dataset.

$R^{(1)}$	$R^{(2)}$	$R^{(3)}$	$R^{(4)}$	$R^{(5)}$	G	ODS	OIS	AP
					√	.762	.782	.766
√	√	√			√	.770	.795	.778
			√	√	√	.780	.801	.785
√	√	√	√	√	√	.787	.804	.817

Table 3: Edge detection results on BSDS500 dataset.

	ODS	OIS	AP
Human	.80	.80	-
gPb-owt-ucm [1]	.726	.757	.696
Sketch Tokens [24]	.727	.746	.780
SCG [38]	.739	.758	.773
MS [18]	.74	.77	.78
SE-Var [8]	.746	.767	.803
OEF [15]	.749	.772	.817
MES [34]	.756	.776	.756
DeepNet [20]	.738	.759	.758
N4-Fields [13]	.753	.769	.784
DeepEdge [2]	.753	.772	.807
MSC [35]	.756	.776	.787
CSCNN [17]	.756	775	.798
DeepContour [32]	.757	.776	.790
HFL [3]	.767	.788	.795
HED-latemerge [39]	.782	.804	**.833**
HED-multiscale [39][1]	.790	.808	.811
RDS (ours)	**.792**	**.810**	.818

Early supervision and late supervision. Since we know the advantage of additional deep supervision, but whether all the intermediate supervision has the same importance or not. We employ the RDS(SE) method to evaluate this test (See Table 2). Here we briefly divide two groups: early supervision and late supervision. The early supervision consists of $R^{(1)}$, $R^{(2)}$, and $R^{(3)}$, and the late supervision includes $R^{(4)}$ and $R^{(5)}$. In addition, the fuse-output supervision with G is necessary all the time. We train the model with early and late supervision separately and compare their effects. We can see that (1) compared with no intermediate supervision, using the late supervision achieves more boosts than the early supervision; (2) training with both early and late supervision outperforms any single way. These results show that all intermediate supervision provides useful and complementary information.

Comparisons with state-of-the-art. Here we compare our RDS(SE)+CEA result against other leading methods on BSDS500 dataset in Table 3. Precision/recall curves are illustrated in Fig. 8. These methods can be categorized into non deep-learning and deep learning groups (See the upper part and lower part in the table). As far as we know, the recent work, MES [34], shows the superior result in the non deep-learning group. On the other hand, HED [39], as an edge detector based on deep learning, leads other methods, meanwhile retaining high efficiency. Our method, RDS, improves the ODS by 1 point and OIS by 0.6 point as compared with HED-latemerge. It is worth mentioning that HED has better average precision (AP), due to its late-merging step. However, we do not perform this optional late-merging step. Besides, HED further presents better results using multi-scale augmentation. Nevertheless, our results are still competitive. In Fig. 9, we illustrate some examples of edge detection results and visually compare with MES [34] and HED [39]. In addition, Figure 10 shows

[1] Augmenting the training images with three scales.

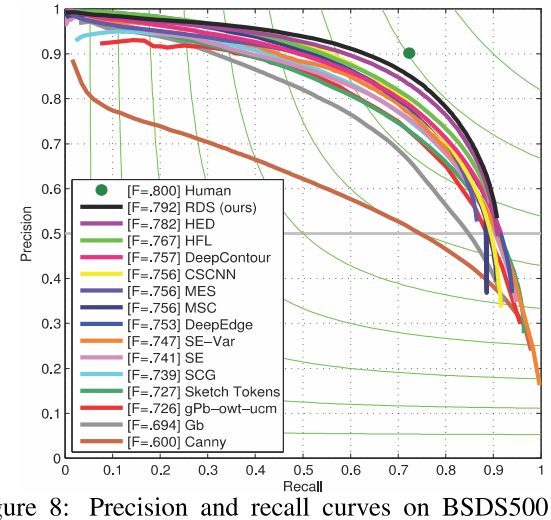


Figure 8: Precision and recall curves on BSDS500 test dataset. These methods are ranked according to their best F-score (ODS). Our method achieves superior result as compared with other top-tier performance.

an example of a zoomed-in region in one image. RDS can extract meaningful edges in a small region.

4.3. Cross dataset generalization

To investigate the generalization of one edge detector, it is necessary to conduct experiments on another dataset. Following the experimental setup in [8, 32], the NYUD dataset (v2) [30] is used as the cross dataset. With the model trained on BSDS500 training set, we evaluate the NYUD performance with its 654 testing images. In Table 4, we compare our ODS results with SE [7, 8] and DeepContour [32]. Notably, to compensate for the relatively inexact ground truth in NYUD dataset, the maximum tolerance (maxDist) allowed for correct matches of edge predictions to ground truth increases from .0075 to .011 [8]. In summary, RDS achieves better cross-dataset generalization results, no matter what the maximum tolerance is.

4.4. Computational cost

In this section, we evaluate the computational cost of the proposed RDS method, including training and testing stages. The experimental environment is CPU i7 with 64GB RAM and NVIDIA K40 GPU. (1) Training stage: we need to extract the relaxed labels using off-the-shelf Canny or SE. They are both quite efficient detectors with about 15

Table 4: Cross-dataset generalization results. The model trained on BSDS500 is used to evaluate NYU test set.

	maxDist=.0075	maxDist=.011
DeepContour [32]	.55	-
SE [7, 8]	.55	.64
RDS(SE)	.611	.627
RDS(SE) + CEA	**.655**	**.674**

Figure 9: Illustration of five edge detection examples. The first and second rows list the original image and its ground truth. The third and forth rows display the results from MES [34] (Threshold=.407) and HED [39] (Threshold=.320). The proposed RDS(SE) (Threshold=.390) and RDS(SE)+CEA (Threshold=.347) methods are shown in the fifth and sixth rows. Compared with HED [39], RDS can reduce the false positives, such as the stones in the first image. Meanwhile, our method can obtain more true positives, for example our closed boundaries of the boats in the second image.

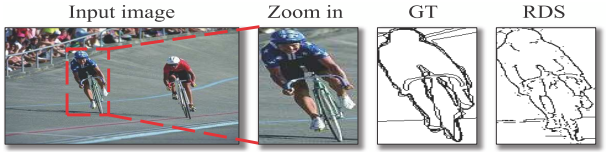


Figure 10: Result of a zoomed in region in one image.

and 2.5 FPS (frames per second), respectively. Next, we use the CEA data to pretrain the network with about 10K iterations, which takes about 10 hours on one K40 GPU. Finally, it just spends less than one hour to train the model on the BSDS500 training set (200 images) with 25 epoches. (2) Testing stage: apart from computing the relaxed labels, our method takes about about 500ms to predict the fusion-output edge map. Thus RDS has the similar magnitude of computational speed as HED.

5. Conclusion

In this paper, we developed an edge detection method influenced by relaxed deep supervision (RDS) to guide the CNN model. Compared with the general deep supervision, RDS takes advantage of all of the intermediate layers or hierarchical diversities within the network, and focuses on the false positives. Consequently, our method achieves considerable improvements, meanwhile retaining high efficiency. In addition, we pretrained the model with coarse edge annotations (CEA) extracted from a large collection of segmentation annotations. This pretraining step obtains further gains. Our results on BSDS500 dataset show state-of-the-art accuracy (ODS=.792). Another cross-dataset test indicates the promising generalization properties of our method. Our work can provide promising insights into efficiently exploiting diverse deep supervision to guide the network. In addition, it is feasible to apply this relaxation strategy to other visual recognition tasks, such as object recognition and image segmentation.

Acknowledgments This work was supported mainly by the LIACS Media Lab at Leiden University and in part by the China Scholarship Council. We are also grateful to the support of NVIDIA with the donation of one Tesla K40 GPU card.

References

[1] P. Arbelaez, M. Maire, C. Fowlkes, and J. Malik. Contour detection and hierarchical image segmentation. *IEEE Trans. Pattern Anal. Mach. Intell.*, 33(5):898–916, 2011. 1, 2, 5, 6, 7

[2] G. Bertasius, J. Shi, and L. Torresani. Deepedge: A multi-scale bifurcated deep network for top-down contour detection. In *CVPR*, 2015. 1, 2, 7

[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. 1, 2, 7

[4] J. Canny. A computational approach to edge detection. *IEEE Trans. Pattern Anal. Mach. Intell.*, 8(6):679–698, 1986. 1, 3, 6

[5] L. Cun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel. Handwritten digit recognition with a back-propagation network. In *NIPS*, 1990. 4

[6] P. Dollár, Z. Tu, and S. Belongie. Supervised learning of edges and object boundaries. In *CVPR*, 2006. 1

[7] P. Dollár and C. L. Zitnick. Structured forests for fast edge detection. In *ICCV*, 2013. 7

[8] P. Dollár and C. L. Zitnick. Fast edge detection using structured forests. *IEEE Trans. Pattern Anal. Mach. Intell.*, 37(8):1558–1570, 2015. 1, 3, 7

[9] M. Everingham, S. M. A. Eslami, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The pascal visual object classes challenge: A retrospective. *IJCV*, 111(1):98–136, 2015. 5

[10] C. Farabet, C. Couprie, L. Najman, and Y. LeCun. Learning hierarchical features for scene labeling. *IEEE Trans. Pattern Anal. Mach. Intell.*, 35(8):1915–1929, 2013. 1

[11] P. F. Felzenszwalb and D. P. Huttenlocher. Efficient graph-based image segmentation. *Int. J. Comput. Vision*, 59(2):167–181, 2004. 1

[12] V. Ferrari, L. Fevrier, F. Jurie, C. Schmid, and S. Member. Groups of adjacent contour segments for object detection. *IEEE Trans. Pattern Anal. Mach. Intell.*, 30(1):36–51, 2008. 1

[13] Y. Ganin and V. S. Lempitsky. N^4 -fields: Neural network nearest neighbor fields for image transforms. In *ACCV*, 2014. 1, 2, 7

[14] R. Girshick, J. Donahue, T. Darrell, and J. Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *CVPR*, 2014. 1

[15] S. Hallman and C. C. Fowlkes. Oriented edge forests for boundary detection. In *CVPR*, 2015. 1, 7

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

[17] J. Hwang and T. Liu. Pixel-wise deep learning for contour detection. In *ICLR*, 2015. 1, 2, 7

[18] P. Isola, D. Zoran, D. Krishnan, and E. H. Adelson. Crisp boundary detection using pointwise mutual information. In *ECCV*, 2014. 1, 7

[19] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell. Caffe: Convolutional Architecture for Fast Feature Embedding. In *ACM Multimedia*, 2014. 5

[20] J. J. Kivinen, C. K. I. Williams, and N. Heess. Visual boundary prediction: A deep neural prediction network and quality dissection. In *AISTATS*, 2014. 1, 2, 7

[21] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *NIPS*, 2012. 1, 3, 4

[22] C. Lee, S. Xie, P. Gallagher, Z. Zhang, and Z. Tu. Deeply supervised nets. In *AISTATS*, 2015. 2

[23] M. Leordeanu, R. Sukthankar, and C. Sminchisescu. Generalized boundaries from multiple image interpretations. *IEEE Trans. Pattern Anal. Mach. Intell.*, 36(7):1312–1324, 2014. 1

[24] J. Lim, C. L. Zitnick, and P. Dollár. Sketch tokens: A learned mid-level representation for contour and object detection. In *CVPR*, 2013. 1, 7

[25] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick. Microsoft COCO: Common objects in context. In *ECCV*, 2014. 5

[26] Y. Liu, Y. Guo, S. Wu, and M. S. Lew. Deepindex for accurate and efficient image retrieval. In *ICMR*, 2015. 1

[27] J. Long, E. Shelhamer, and T. Darrell. Fully convolutional networks for semantic segmentation. In *CVPR*, 2015. 1, 2

[28] A. Mahendran and A. Vedaldi. Understanding deep image representations by inverting them. In *CVPR*, 2015. 3

[29] 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. 2, 5

[30] P. K. Nathan Silberman, Derek Hoiem and R. Fergus. Indoor segmentation and support inference from rgbd images. In *ECCV*, 2012. 6, 7

[31] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla,

M. Bernstein, A. C. Berg, and L. Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *IJCV*, pages 1–42, 2015. 5

[32] 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. 1, 2, 7

[33] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. In *ICLR*, 2015. 1, 2, 5

[34] A. Sironi, V. Lepetit, and P. Fua. Projection onto the manifold of elongated structures for accurate extraction. In *ICCV*, 2015. 1, 7, 8

[35] A. Sironi, E. Tretken, V. Lepetit, and P. Fua. Multiscale centerline detection. *IEEE Trans. Pattern Anal. Mach. Intell.*, 2015. 7

[36] Y. Sun, X. Wang, and X. Tang. Deeply learned face representations are sparse, seletive, and robust. In *CVPR*, 2015. 1, 2

[37] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. In *CVPR*, 2015. 1, 2

[38] R. Xiaofeng and L. Bo. Discriminatively trained sparse code gradients for contour detection. In *NIPS*, 2012. 1, 7

[39] S. Xie and Z. Tu. Holistically-nested edge detection. In *ICCV*, 2015. 1, 2, 5, 7, 8

[40] Z. Yan, H. Zhang, R. Piramuthu, V. Jagadeesh, D. DeCoste, W. Di, and Y. Yu. Hd-cnn: Hierarchical deep convolutional neural network for large scale visual recognition. In *ICCV*, 2015. 4