Exploring images with deep learning for classification, retrieval and synthesis

Liu, Y.

Chapter 3

Recognizing Image Edges

In the previous chapter, we have shown the generalization power of deep neural networks for pixel-level classification. In this chapter, we focus on how we can develop diverse supervision in CNNs for edge detection (RQ 2).

To improve the robustness of edge detection, we build hierarchical supervisory signals with additional relaxed labels and adapt the signals to consider the diversities in hierarchial layers. Specifically, we begin by capturing the relaxed labels from simple detectors (*e.g.* Canny). These relaxed labels can be seen as some false positives that are difficult to be classified. Then we merge them with the general ground-truth to generate the relaxed deep supervision (RDS). We can employ the RDS to supervise the edge detection network in a coarse-to-fine paradigm. Moreover, we compensate for the lack of training images by capturing coarse edge annotations from a segmentation dataset. We pre-train the model with coarse annotations and then fine-tune it with fine annotations. Extensive experiments demonstrate that our approach achieves superior performance on the BSDS500 dataset (ODS F-score of .792) and promising cross-dataset results on the NYUD dataset.

Keywords
Edge detection, Fully convolutional networks, Deep supervision, Pre-training

3.1 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 other high-level vision applications, for example, object detection [147] and segmentation [126]. Most of the traditional edge detection approaches [109, 110, 111, 112, 148, 149, 150, 151] extract discriminative local features with color and gradient clues, such as gPb [126], Sketch tokens [152] and Structured Edges (SE) [127].

Recently, edge detection has achieved significant advances due to the developments of deep features. Figure 3.1 displays the basic pipeline of current edge detection systems based on deep learning. Based on different levels in predicting edges, 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 [113] developed a convolutional RBM to learn pixel-level features. Hwang and Liu [153] stacked pixel features in a multi-scale CNN model and then fed them to an SVM classifier. Bertasius *et al.* [29] built four CNN models to learn multi-scale features to detect edge points. Then they improved their network structure with less computational cost [154].

(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 [155] 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.* [30] 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 [31] 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 of end-to-end fully convolutional networks (FCNs) [26], and additional deep supervision of deeply supervised nets (DSN) [125].

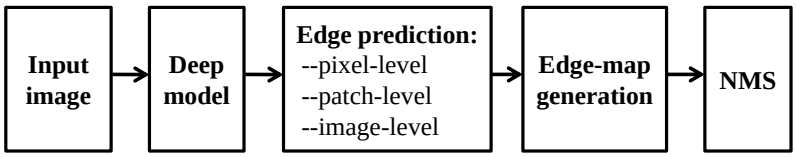

Figure 3.1: Pipeline of deep learning based edge detection. NMS is short for non-maximal suppression.

One difficulty in edge detection is attributed to *false positives*: many non-edge pixels are incorrectly predicted as edges compared with the human annotated ground-truth. To alleviate this issue, HED [31] imposed additional supervision (*i.e.* the annotated ground-truth) on the intermediate layers while training the deep model, and therefore the false positives could be corrected earlier. However, using only a general supervision for all the layers is inconsistent with the diverse representations of hierarchical layers. In addition, the general supervision can not be well-suited to all intermediate layers. Driven by this issue, in this chapter we pose a new research question **RQ 2: How can we explore diverse supervision that can adapt to different intermediate layers in deep neural networks for robust edge detection?**

To this end, we propose 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), having additional relaxed labels, in addition to the positive labels (*i.e.* edge points) and negative labels (*i.e.* non-edge points). The relaxed labels are used to adapt to the diversities of intermediate layers. To be specific, we capture the relaxed labels from simple and efficient off-the-shelf detectors, for instance, Canny [109] or SE [127]. Then, we insert the extracted relaxed labels into the original ground-truth to generate RDS. In contrast to using a fixed general supervision, RDS can guide intermediate layers in a coarse-to-fine paradigm and process the false positives using a "delayed strategy". In this way, the loss cost of the relaxed labels are ignored in current supervision, and will be reconsidered in the next supervision. Therefore, more discriminative layers are assigned to process more false positives (difficult points). RDS can incorporate network diversities to improve the performance of edge detection.

Another problem about edge detection is that it requires more expensive human annotations, than other vision tasks like image classification and object detection. In addition, the frequently benchmarked BSDS500 dataset [126] has only 200 training images that limits the learning ability of various edge detectors based on deep learning. To alleviate this deficiency, we propose to generate coarse edge annotations (CEA) from a large collection of segmentation annotations such as the PASCAL Context dataset [156]. We pre-train the model with CEA and then fine-tune it with the target dataset, BSDS500.

The contributions of this work are as follows:

- 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 efforts.
- 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 can achieve considerable improvements.

- Despite the apparent simplicity of RDS, our approach achieves competitive accuracy (ODS=.792) on the well-known benchmark BSDS500. In addition, our approach shows promising generalization between different datasets.

The rest of this chapter is structured as follows. Section 3.2 presents the proposed relaxed deep supervision for edge detection. The pre-training procedure with CEA is introduced in Section 3.3. In 3.4, we describe the implementation details in Section and report the experimental results. Finally, Section 3.5 summarizes the conclusions and future work.

3.2 Relaxed Deep Supervision

In this section, we present the proposed network with relaxed deep supervision for edge detection and formulate the algorithm.

3.2.1 Network details

Model Architecture

Our edge detection architecture is built on top of HED network [31], which is trimmed from the VGG-16 net [7] (Figure 3.2). 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 approach stems from the fact that hierarchical layers can represent specific abstracts of the input image [157, 158]. In Figure 3.2, 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)

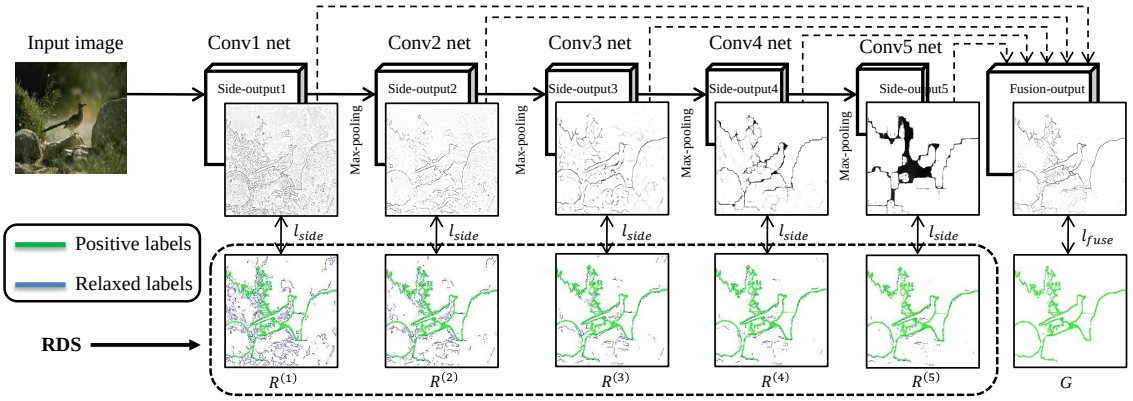

Figure 3.2: The network architecture with RDS (best viewed in color and zoom-in). The proposed RDS, including positive labels (green color), negative labels (white color for clear visualization), relaxed labels (blue color), is used to supervise the corresponding side-output prediction. The last fusion-output 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}.

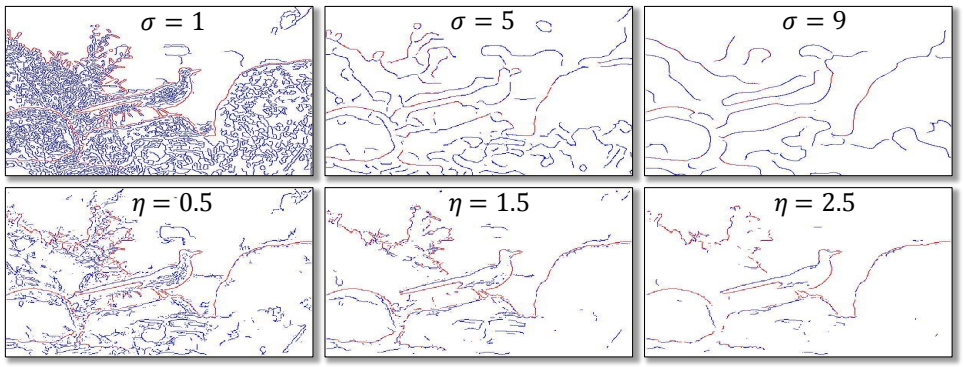

Figure 3.3: Illustration of extracting relaxed labels (blue color). The first and second rows display three edge responses from Canny [109] and SE [127], respectively.

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 by introducing the relaxed labels. In the following, we present two simple and efficient ways to capture the relaxed labels based on off-the-shelf edge detectors, including Canny [109] and SE [127].

Relaxed labels based on Canny detector

The Canny algorithm [109] 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 adjust different scales ($\sigma \in \{1, 3, 5, 7, 9\}$) to obtain various edge responses for five side-output predictions. 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 k-th side-output

prediction, 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." The relaxed labels can present the complementary clues that are not in the ground-truth. Therefore, the set of relaxed labels can be computed as follows:

$$D^{(k)} = \mathrm{H}(C^{(k)} - C^{(k)} \cap G), \tag{3.1}$$

where the function H is used to collect the set of positive labels from the input binary map. As shown in Figure 3.3, the first row gives three scales of Canny edge responses (both red and blue color) when $\sigma = 1, 5, 9$. 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 Figure 3.2.

Relaxed labels based on SE detector

To demonstrate the generalization of our method, we also employ another edge detector: Structured Edges (SE) [127]. 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 the 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 can have five binary edge responses, denoted as $\{S^{(k)}\}_{k=1}^{5}$. Similar to the above definition of relaxed labels, we compute the set of relaxed labels based on SE

$$D^{(k)} = \mathrm{H}(S^{(k)} - S^{(k)} \cap G). \tag{3.2}$$

The second row in Figure 3.3 displays the edge responses from SE and their relaxed labels (blue color). We can observe that the relaxed labels from SE detector are visually sparser than those from Canny detector.

RDS generation

It can be observed that various relaxed labels are well-suited to our needs of highlighting hierarchical diversities within the supervision. In the next stage, we need to insert the set of relaxed labels into the original ground-truth. This merging operation is used to generate the desirable RDS, which is an union of positive, negative, and relaxed labels. We denote five different RDS by $\{R^{(k)}\}_{k=1}^{5}$. The construction step can be seen in Figure 3.4, where the set $D^{(3)}$ is extracted based on $S^{(3)}$. 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 correspond to the non-edge points in G. These non-edge points can be viewed as some false positives that are difficult to classify.

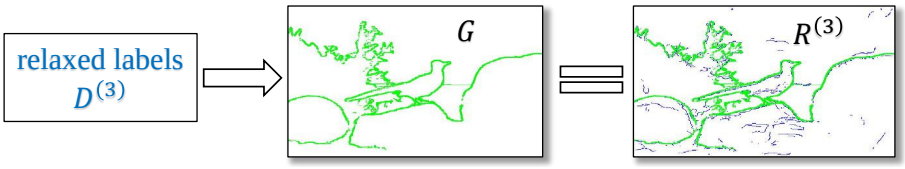

Figure 3.4: Illustration of generating the RDS (best viewed in zoom-in). $R^{(3)}$ is merged by the set $D^{(3)}$ and G.

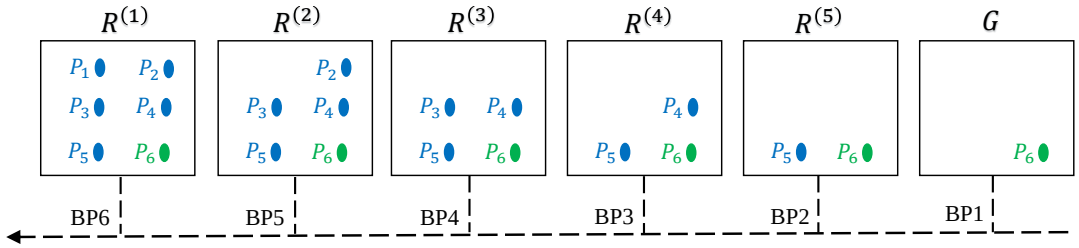

Figure 3.5: RDS employs a coarse-to-fine supervision strategy. The blue points indicate the relaxed labels, and the green point is one positive label.

3.2.2 Loss formulation

For a training dataset containing N images: $\{I_i, G_i\}_{i=1}^{N}$, I_i is the i-th input image and G_i is its edge 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 $\{R_i^{(k)}\}_{k=1}^{K}$, and $K = 5$ in the 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 (Figure 3.2). In addition, early supervision (*e.g.* $R^{(1)}$ and $R^{(2)}$) has more relaxed labels than late supervision (*e.g.* $R^{(4)}$ and $R^{(5)}$). This is consistent with the hierarchical characteristics of CNN models. Finally, the total loss function L_{RDS} is expressed with

$$\sum_{i=1}^{N}\sum_{j=1}^{|I_i|}\left(\sum_{k=1}^{K} l_{side}(\widehat{G}_{i,j}^{(k)}, R_{i,j}^{(k)}) + l_{fuse}(\widehat{G}_{i,j}^{fuse}, G_{i,j})\right), \tag{3.3}$$

where $|I_i|$ is the total number of 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 to 2, different from the positive labels (set to 1) and negative labels (set to 0). Therefore, we compute l_{side} based on the types of pixel labels

$$l_{side}(\widehat{G}_{i,j}^{(k)}, R_{i,j}^{(k)}) = \begin{cases} \alpha \cdot \log \mathrm{P}(\widehat{G}_{i,j}^{(k)}), & R_{i,j}^{(k)} = 1 \\ \beta \cdot \log(1 - \mathrm{P}(\widehat{G}_{i,j}^{(k)})), & R_{i,j}^{(k)} = 0 \\ 0, & R_{i,j}^{(k)} = 2 \end{cases} \tag{3.4}$$

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 $\beta = 1$. Notice that, 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. (3.4), because there are no relaxed labels in G_i. Next, we consider the back propagation (BP). We can deduce the partial derivatives of l_{side} *w.r.t.* $\widehat{G}_{i,j}^{(k)}$ by

$$\bigtriangledown \frac{l_{side}}{\widehat{G}_{i,j}^{(k)}} = \begin{cases} \alpha \cdot (\mathrm{sigmoid}(\widehat{G}_{i,j}^{(k)}) - 1), & R_{i,j}^{(k)} = 1 \\ \beta \cdot \mathrm{sigmoid}(\widehat{G}_{i,j}^{(k)}), & R_{i,j}^{(k)} = 0 \\ 0, & R_{i,j}^{(k)} = 2 \end{cases} \tag{3.5}$$

We follow the chain rule [159] to update the network parameters using stochastic gradient descent (SGD) with a mini-batch size [157].

Discussion. Training with RDS can maintain the strong supervision from the ground-truth, and incorporate hierarchical diversities. Here we will discuss 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 classify. 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, but can be reconsidered in the next supervision. In this way, top layers are responsible for classifying the ambiguous false positives due to their high discriminative power. This paradigm is similar to hierarchical object classification [160], in which difficult classes are classified from coarse-category prediction to fine-category prediction.

We further demonstrate the paradigm in Figure 3.5. 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 of the supervision and assign more false positives to more high-level layers. Moreover, the network can run in a *coarse-to-fine* BP procedure. First, the whole network can be updated with *coarse* supervision G; Then, *fine* supervision $R^{(k)}$ (with specific relaxed labels) is used to fine-tune their local nets. For example, P_6 can be updated by all BPs (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 help reduce the total loss in the forward pass stage and facilitate efficient updates in the back propagation stage.

(a) Fine edge annotations

(b) Coarse edge annotations

Figure 3.6: Comparison between fine and coarse edge annotations. (a) displays three images and their ground-truth from BSDS500 [126]. (b) shows the images, segmentations from Pascal Context [156] in the first and second row, and coarse edge annotations (CEA) in the third row.

3.3 Pre-training Procedure

Generally, collecting more training data can develop the learning ability of CNNs. For many visual recognition tasks such as image classification and object detection, large-scale datasets are often available, *e.g.* ImageNet [5], MSCOCO [117] and PASCAL VOC [129]. However, the BSDS500 dataset [126] contains only 200 training images for learning edge detectors. This small training set limits current edge detection algorithms in improving the performance. In addition, fine edge annotations (FEA) require more expensive human effort than image classification.

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

Algorithm 1: RDS: training and testing procedure

Input: Training dataset; VGG-16 net; training iterations T_1,T_2
Initializing: network parameters $\mathbf{W}$ using VGG model
Preparation: for one image I_i, extract the set of relaxed labels $\{D_i^{(k)}\}_{k=1}^5$ and generate RDS $\{R_i^{(k)}\}_{k=1}^5$.
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. (3.6);
    Backward propagate to get gradients $\Delta\mathbf{W}$, like Eq. (3.5);
    Update $\mathbf{W}^t = \mathbf{W}^{t-1} - \lambda_t \Delta\mathbf{W}$ with SGD;
  end while
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. (3.3);
    Backward propagate to get gradients $\Delta\mathbf{W}$, like Eq. (3.5);
    Update $\mathbf{W}^t = \mathbf{W}^{t-1} - \lambda_t \triangle \mathbf{W}$ with SGD;
  end while
Testing: feed one image into the learned network with parameters $\mathbf{W}$ and output edge map E_i
Post-processing: non-max suppression on E_i
Output: final edge map $E_i^{'}$

dataset [156], which provides full-scene segmentations for more than 400 classes, and has 10,103 train and validation images in total. Thus we extract the edges alongside the segmentations. In contrast to FEA, CEA only provides the outside boundaries of objects (See the car, people and building in Figure 3.6), but it can facilitate the network learning due to a large number of images. Notably, there are no overlap images between Pascal Context and BSDS500, which are from Flickr and Corel, respectively. During training with CEA, we simply compute the fusion-output loss function and exclude the intermediate supervision by

$$L_{CEA} = \sum_{i=1}^{N} \sum_{j=1}^{|I_i|} \left(l_{fuse}(\widehat{G}_{i,j}^{fuse}, G_{i,j}) \right). \tag{3.6}$$

In summary, we pre-train the model with the Pascal Context dataset and its CEA according to Eq. (3.6), and then fine-tune the model with the BSDS500 dataset as Eq. (3.3). We show the whole algorithm procedure in Algorithm 1, including the training and testing stages.

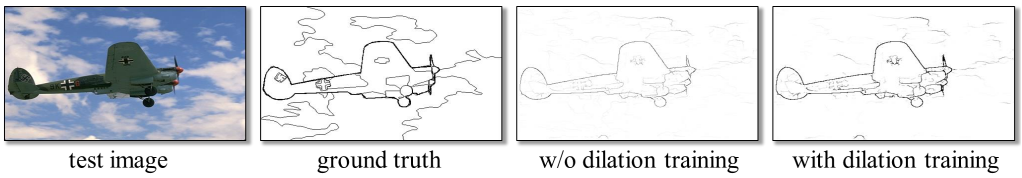

Figure 3.7: Qualitative comparison of edge detection results between without and with ground-truth dilation.

3.4 Experiments

3.4.1 Implementation details

Training details. We implemented our approach using the publicly available Caffe framework [130] and HED implementation [31]. 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. (3.4). We utilize the fixed class-balanced parameters ($\alpha = 9, \beta = 1$) for all images.

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 have rather thick boundaries. To tackle this inconsistency, we dilate the positive labels in the ground-truth of a train set using a traditional morphologic dilation operator. Figure 3.7 compares the detection results between with and without 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 the ODS accuracy about .02 on the BSDS500 test set. Hence, the ground-truth dilation is a simple and efficient step for improving the performance on edge detection. Note that we do not dilate the test set. In addition, the postprocessing non-maximal suppression (NMS) [109] can be used to thin the predicted edges.

3.4.2 Ablation study on BSDS500

Dataset. The BSDS500 dataset [126] consists of 200 training, 100 validation, and 200 testing images. The validation set is used to fine-tune the hyper-parameters. 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

Table 3.1: 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.

	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

the performance. As mentioned in Section 2.4.5, we use the fixed contour threshold (ODS), the per-image best threshold (OIS) and the average precision (AP).

Baseline methods. To experimentally evaluate the effectiveness of RDS, we 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 3.1, the Baseline 1 achieves ODS=.762 on BSDS500. Relatively, the Baseline 2 improves the accuracy to ODS=.780. This verifies the benefit of using additional intermediate supervision. The performance gap with/without intermediate supervision in HED [31] 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.

Component analysis

Table 3.1 reports the results of our approach. To give more insights, we discuss them from three aspects.

(1) RDS yields considerable improvements over the general supervision approach (Baseline 2). This verifies the advantage of RDS for incorporating hierarchical diversities. The result of RDS with relaxed labels from Canny, denoted as RDS(Canny), achieves ODS=.785. The RDS(SE) result reaches 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 [126], to capture the relaxed labels. Similarly, its result (ODS=.786) is consistent with RDS(Canny) and RDS(SE). Thus we have not invested too much effort in optimizing various relaxed labels now.

Table 3.2: 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

(3) Pre-training with CEA demonstrate further gains for both RDS(Canny) and RDS(SE), reaching ODS=.790 and .792, respectively. In addition, we also evaluate the model pre-trained with CEA (without fine-tuning on the BSDS500 training set), which achieves ODS=.765. These results show the necessity and advantage of using a large-scale dataset.

Early supervision and late supervision

We have known the advantage of additional deep supervision. This experiment aims to examine whether all the intermediate supervision has the same importance or not. We employ the RDS(SE) method in an attempt to resolve this question. As shown in Table 3.2, 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 other approaches

Here we compare our RDS(SE)+CEA result against other leading methods on BSDS500. These methods can be categorized into non deep-learning and deep learning approaches, as seen in the upper part and lower part in Table 3.3). Precision/recall curves are illustrated in Figure 3.8. As far as we know, the recent work, MES [112], shows superior results for the non deep-learning approaches. On the other hand, HED [31], as an edge detector based on deep learning, leads the 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

Table 3.3: Edge detection results on the BSDS500 dataset. Our approach is competitive with other state-of-the-art approaches. Note that, HED-multiscale augments the training images with three scales.

	ODS	OIS	AP
gPb-owt-ucm [126]	.726	.757	.696
Sketch Tokens [152]	.727	.746	.780
SCG [110]	.739	.758	.773
MS [150]	.74	.77	.78
SE-Var [127]	.746	.767	.803
OEF [151]	.749	.772	.817
MES [112]	.756	.776	.756
DeepNet [113]	.738	.759	.758
N4-Fields [155]	.753	.769	.784
DeepEdge [29]	.753	.772	.807
MSC [161]	.756	.776	.787
CSCNN [153]	.756	775	.798
DeepContour [30]	.757	.776	.790
HFL [154]	.767	.788	.795
HED-latemerge [31]	.782	.804	**.833**
HED-multiscale [31]	.790	.808	.811
RDS (ours)	**.792**	**.810**	.818

still competitive. In addition to the above quantitative results, we further show some qualitative image examples. In Figure 3.9, we illustrate some examples of our results.

3.4.3 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 [30, 127], the NYUD dataset (v2) [162] is used as the cross dataset. With the model trained on BSDS500 training set, we evaluate the BSDS500 models on the NYUD dataset with its 654 testing images. Since these models are trained with color images, we only test the color images in the NYUD dataset. In Table 3.4, we compare our ODS results with SE [127, 163] and DeepContour [30]. To compensate for the relatively inexact ground truth in NYUD dataset, we increase the maximum tolerance (maxDist) allowed for correct matches of edge predictions to ground truth from .0075 to .011 [127]. We can see that, RDS achieves better cross-dataset generalization results, no matter what the maximum tolerance is.

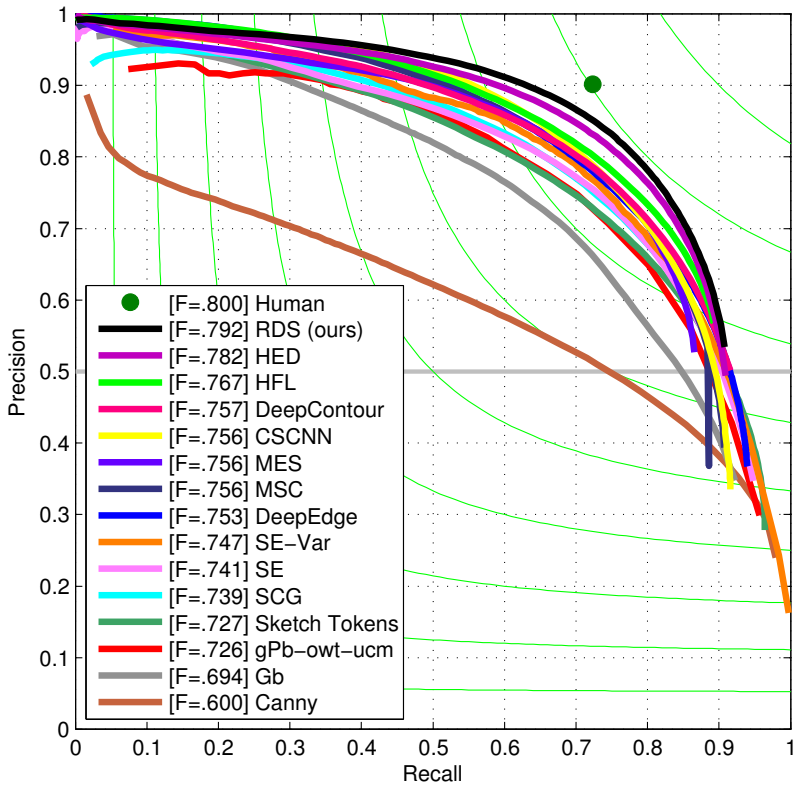

Figure 3.8: Precision and recall curves on BSDS500 test dataset. These methods are ranked according to their best F-score (ODS). Our method achieves superior result as compared with other top-tier performance.

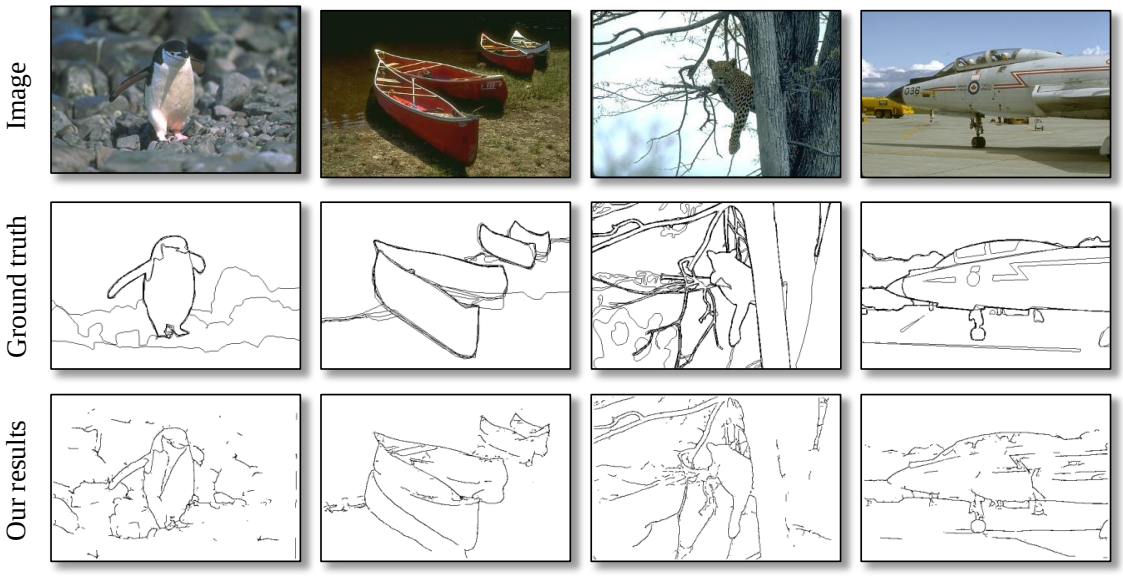

Figure 3.9: Illustration of five edge detection results. Our method can detect meaningful edges, even though they still have some differences from the ground-truth annotations.

Table 3.4: Cross-dataset generalization results (ODS F-score). The model trained on BSDS500 is used to evaluate the NYUD test set.

	maxDist=.0075	maxDist=.011
DeepContour [30]	.550	-
SE [127, 163]	.550	.64
RDS(SE)	.611	.627
RDS(SE) + CEA	**.655**	**.674**

3.4.4 Computational cost

Moreover, we report the computational cost of the proposed RDS method, including the training and testing stages. The experimental environment is Intel i7 CPU 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 and 2.5 FPS (frames per second), respectively. Next, we use the CEA data to pre-train the network with for 10,000 iterations, which takes about 10 hours on one K40 GPU. Finally, it spends less than one hour to train the model on the BSDS500 training set (200 images) for 25 epoches. (2) Testing stage: apart from computing the relaxed labels, our method takes about 500ms to predict the fusion-output edge map. Similar to HED, RDS has the similar order of magnitude in terms of computational speed.

3.5 Chapter Conclusions

In this chapter, we developed an edge detection method influenced by relaxed deep supervision (RDS) to guide the training of deep neural networks. Compared with the general deep supervision, RDS generated diverse supervisory signals to guide different intermediate layers. It can make the network have more focus on the false positives. Consequently, our method achieved considerable improvements, meanwhile retaining high efficiency. In addition, we proposed to pre-trained the model with coarse edge annotations (CEA) extracted from a large collection of segmentation annotations. This pre-training step can alleviate the lack of expensive edge annotations. Our results on the BSDS500 dataset demonstarted competitive performance (ODS=.792) with the state-of-the-art approaches. Another cross-dataset test indicated the promising generalization power of our method.

Future work. The work in this chapter has provided promising insights into efficiently exploiting diverse deep supervision to guide the network. Therefore, it is feasible to apply this relaxation strategy to other visual recognition tasks, such as object recognition and image segmentation. In addition, we will study theoretical analysis to provide more insights into diverse deep supervision.