RankED: Addressing Imbalance and Uncertainty in Edge Detection Using Ranking-based Losses

Bedrettin Cetinkaya, Sinan Kalkan[†], Emre Akbas[†]
Department of Computer Engineering and METU ROMER Robotics Center
Middle East Technical University, Ankara, Turkey
{bckaya, skalkan, eakbas}@metu.edu.tr

Abstract

Detecting edges in images suffers from the problems of (P1) heavy imbalance between positive and negative classes as well as (P2) label uncertainty owing to disagreement between different annotators. Existing solutions address P1 using class-balanced cross-entropy loss and dice loss and P2 by only predicting edges agreed upon by most annotators. In this paper, we propose RankED, a unified ranking-based approach that addresses both the imbalance problem (P1) and the uncertainty problem (P2). RankED tackles these two problems with two components: One component which ranks positive pixels over negative pixels, and the second which promotes high confidence edge pixels to have more label certainty. We show that RankED outperforms previous studies and sets a new state-of-the-art on NYUD-v2, BSDS500 and Multi-cue datasets. Code is available at https://ranked-cvpr24.github.io.

1. Introduction

Detecting contours of objects in a given image is a fundamental problem in Computer Vision. It has been approached as a machine learning problem since the introduction of the influential BSDS dataset [1]. As with any learning-based approach, characteristics of the training data affects performance. One striking issue regarding ground-truth contour data is that contours are rare events. For example, in the BSDS dataset, only 7% of all pixels within an image are marked as edge pixels[1]. This creates a significant imbalance between the positive (edge) and negative (non-edge) classes, which hinders the training of machine learning models. Another important issue observed in edge data is the uncertainty regarding the ground-truth annotations.

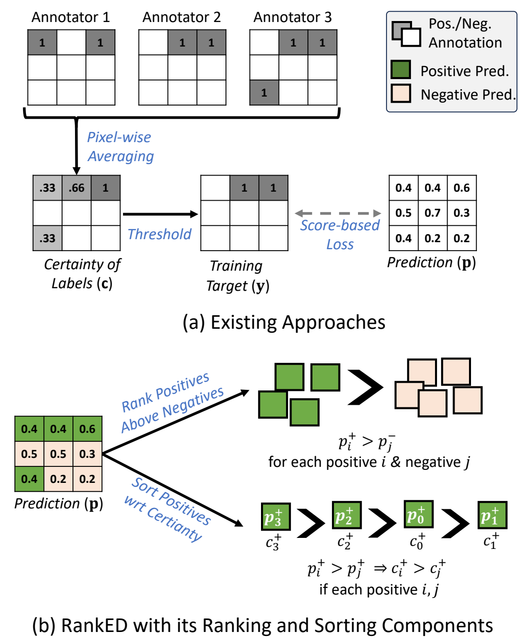

Figure 1. (a) Current approaches threshold label certainties and class-balanced cross-entropy loss for training edge detectors. (b) With RankED, we propose a unified approach which ranks positives over negatives to handle the imbalance problem and sorts positives with respect to their certainties.

There exist a non-trivial amount of variation between the annotations produced by different human annotators, which essentially creates noise in the supervisory signal.

These two issues of edge ground-truth data, namely the imbalance and uncertainty, have long been known, however, efforts to address them have remained limited. For the imbalance problem, although there is a vast literature on long-tailed and imbalance learning (see, e.g., [46, 53]), researchers have only explored using Dice Loss [10–12] and weighted cross-entropy loss [20, 29, 39, 44, 49], which mitigate the problem to a certain extent. However, as we show in this paper, they are far from finding the optimal solu-

[†]Equal contribution.

[1]Although some studies call such high-level, semantic edges as "contour" and low-level edges as "edge", we follow the recent literature [12, 15, 39, 44, 55] and use the term "edge" for contours in the rest of the paper.

tion. The label uncertainty has been tackled using edge pixels agreed by most annotators [12, 20, 29, 39, 49] (Figure 1(a)). A recent exception is the study by Zhou et al. [55] who proposed jointly learning the mean and variances of labels using multi-variate Gaussian distributions. To the best of our knowledge, this is the only work in literature to attempt the uncertainty problem.

In this paper, we address both the imbalance and the uncertainty problems encountered in edge detection using ranking-based losses. We draw inspiration from the recent success of ranking-based losses in object detection [6, 36, 40, 41] and instance segmentation [36]. Similar to edge detection, the data in these problems also manifest high imbalance between classes [35]. In particular, AP Loss [6] and RS Loss [36] have shown to outperform focal loss and weighted cross-entropy in these problems. Their success stems from the ability to naturally balance the gradients for different classes [34]. Inspired by these works, we propose RANKED, which basically uses a ranking-based loss function to learn from data (Figure 1(b)). When a training image has only one annotation, then RANKED tries to rank edge pixels above non-edge pixels. When there are multiple annotations per training image, RANKED not only tries to rank edge pixels above non-edge pixels but also to sort edge pixels with respect to their annotation certainty.

Contributions. Our main contributions are as follows:

- We propose RANKED, a new ranking-based loss function for edge detection.
- RANKED simultaneously addresses the imbalance and uncertainty issues commonly encountered in edge detection datasets.
- Our experiments on three edge detection datasets (BSDS, NYUDv2, MultiCue) show that when integrated with Swin-Transformer, RANKED consistently outperforms all SOTA models in average precision (AP).

2. Related Work

Edge Detection. Before the rise of deep learning, the conventional edge detection methods were based on handcrafted filters such as Sobel [25], Canny [4], and Laplacian of Gaussian [32]. Deep learning has enabled detecting edges directly from data and has provided high quality results [3, 10, 20, 23, 26, 31, 38, 39, 42, 44, 49, 50, 55].

Earlier deep learning based methods [2, 3, 42] used features extracted from fixed patches. In this approach, CNNs detect edges using extracted patches around candidate contour points. Following studies extended this patch-based approach with end-to-end learning. These studies focused on detecting edges at multiple scales [49] by combining features at different layers [29] and in cascades [15, 20] and making these networks faster [44]. Recently, attention-based modules and transformers have been shown to provide significant improvements [12, 39].

Loss Functions for Edge Detection. Edge detection is generally tackled using score-based classification losses, e.g., Cross Entropy (CE) Loss [20, 39, 44, 49]. However, owing to the severe imbalance problem between positives ($\mathcal{P}$) and negatives ($\mathcal{N}$), CE Loss is often weighted with a class balancing (CB) factor, e.g. as follows [49]:

$$\mathcal{L}_{CB} = \sum_i -\beta y_i \log(p_i) - (1-\beta)(1-y_i)\log(1-p_i), \quad (1)$$

where p_i is the edge prediction probability for pixel i and y_i is its target. The terms are multiplied with weights to mitigate the imbalance issue: $\beta = |\mathcal{N}|/|\mathcal{N} \cup \mathcal{P}|$, $(1-\beta) = |\mathcal{P}|/|\mathcal{N} \cup \mathcal{P}|$. With these weights, class-balanced loss function applies a higher penalty for edge pixels as they are rare.

A more common approach is to use an adaptation of the Dice Loss [45] for edge detection [10–12]:

$$\mathcal{L}_{DICE} = \frac{\sum_i p_i^2 + \sum_i y_i^2}{2\sum_i p_i y_i}, \quad (2)$$

where i runs over all pixels. Dice Loss is often combined with CE Loss [10–12]:

$$\mathcal{L}_{final} = \alpha \mathcal{L}_{DICE} + \beta \mathcal{L}_{CE}, \quad (3)$$

where α and β balance the two loss functions.

Although there are studies that propose regularization terms to improve edge detection quality (e.g., sharpness [12]); Cross Entropy, Class Balanced Cross Entropy and Dice are the de facto losses used for training deep edge detectors. To the best of our knowledge, there are no ranking-based losses proposed as an alternative.

Using Uncertainty in Edge Detection. Although uncertainty has been shown to be a useful measure for various Computer Vision problems such as classification [24], object detection [19, 27], depth estimation [21, 37], and semantic segmentation [22, 54], there is only one study [55] employing uncertainty in edge detection. Zhou et al. [55] propose an uncertainty-aware edge detector (UAED) that exploits the multi-label nature of the edge detection problem. Their method jointly learns the mean and variance of given inputs and constructs multi-variate Gaussian distribution using predicted mean and variance. Moreover, their method gives more importance to the loss of pixels with higher uncertainty.

Comparative Summary We note from the literature reviewed above and their summary in Table 1 that there are no studies that adopt a ranking approach to edge detection. There is only one study [55] utilizing label uncertainty in edge detection, which, however, does not use a ranking-based approach.

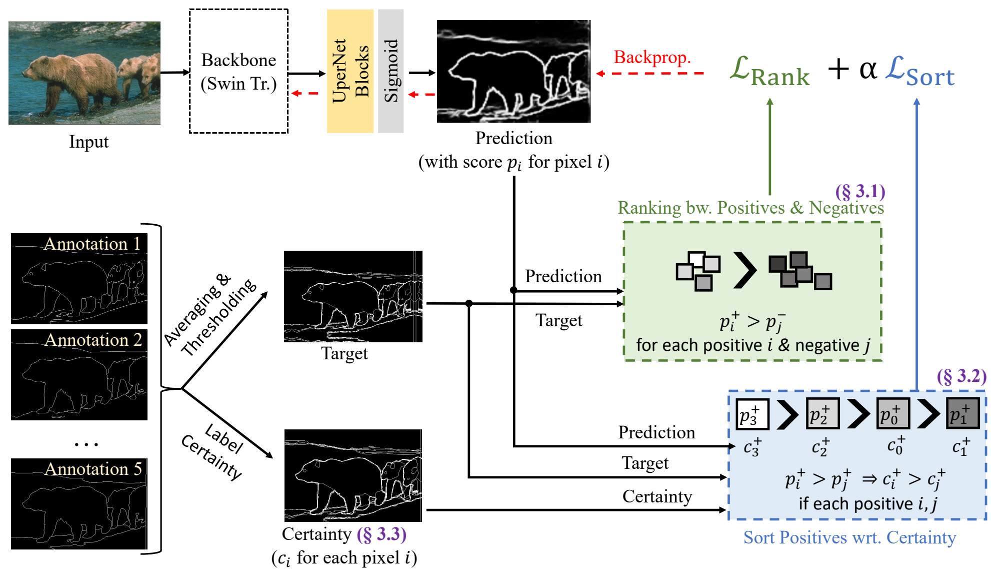

Figure 2. An overview of RANKED. RANKED introduces two novel loss functions for edge detection: $\mathcal{L}_{\text{Rank}}$ (§ 3.1) for ranking positive pixels over background pixels, and $\mathcal{L}_{\text{Sort}}$ (§ 3.2) for sorting positive pixels with respect to their (un)certainties (§ 3.3).

Method		Ranking-based Loss Function?	Uncertainty?
BDCN [20]	(CVPR '19)	✗	✗
PiDiNet [44]	(ICCV '21)	✗	✗
EDTER [39]	(CVPR '22)	✗	✗
FCL-Net [51]	(NN '22)	✗	✗
ACTD [12]	(NeuroCom. '23)	✗	✗
CHRNet [15]	(Pat.Rec. '23)	✗	✗
UAED [55]	(CVPR '23)	✗	✓
RANKED	(This paper)	✓	✓

Table 1. A comparative summary with recent studies.

3. Methodology: RANKED

RANKED is a ranking-based solution with two novel components for edge detection (Figure 2):

- Ranking Positives over Negatives ($\mathcal{L}_{\text{Rank}}$): RANKED is supervised with a loss function that ranks the classification scores of positives (edge pixels) over negatives (non-edge pixels). For this, we use a differentiable approximation of Average Precision as a loss function, following its successful applications in object detection [5].
- Sorting Positives with Respect to their Uncertainties ($\mathcal{L}_{\text{Sort}}$): Positive pixels tend to have different uncertainties, which can be used to adjust their supervision. RANKED gives more priority for positive pixels with higher certainty. For this, we adapt the loss function proposed by Oksuz et al. [36] for sorting positives with respect to their localization qualities in object detection.

3.1. Ranking Positive Edge Pixels over Negatives

To rank the scores of positive edge pixels higher than those of the negative pixels, we adapt the ranking-based loss function proposed by Chen et al. [5] for object detection. Chen et al. approximated Average Precision as a differentiable ranking based loss function, which was later shown by Oksuz et al. [34] to be very robust to extreme positive-negative imbalance. As edge detection exhibits a severe imbalance between positive and negative classes (positive pixels are 1%, 3%, 7% of all pixels for Multicue-boundary, Multi-cue edge, and BSDS datasets, respectively), a ranking-based loss maximizing Average Precision (AP) can be beneficial for edge detection.

We define $\mathcal{L}_{\text{Rank}}$, the loss for ranking positive edge pixels above negatives in terms of Average Precision by counting positives and negatives as follows (following the notation introduced in Section 2 and Chen et al. [6]):

$$\mathcal{L}_{\text{Rank}} = 1 - \text{AP} = 1 - \frac{1}{|\mathcal{P}|} \sum_{i \in \mathcal{P}} \frac{\text{rank}^+(i)}{\text{rank}(i)}, \tag{4}$$

where $\text{rank}^+(i)$ is the rank of pixel i among positives ($\mathcal{P}$) defined as $\text{rank}^+(i) = \sum_{j\in\mathcal{P}} H(x_{ij})$ with $x_{ij} = p_j - p_i$. Similarly, $\text{rank}(i)$ is the rank of pixel i among all predictions: $\text{rank}(i) = \sum_{j\in\mathcal{P}\cup\mathcal{N}} H(x_{ij})$. These rank definitions rely on a step function, $H(\cdot)$, with a δ-approximation around the step, as suggested by Chen et al.:

$$H(x) = \begin{cases} 0\,, & x < -\delta \\ \dfrac{x}{2\delta} + 0.5\,, & -\delta \le x \le \delta \\ 1\,, & \delta < x. \end{cases} \tag{5}$$

Plugging these definitions into Eq. 4 yields:

$$\begin{aligned}\mathcal{L}_{\text{Rank}} &= 1 - \frac{1}{|\mathcal{P}|}\sum_{i\in\mathcal{P}} \frac{\sum_{j\in\mathcal{P}} H\left(x_{ij}\right)}{\sum_{j\in\mathcal{P}\cup\mathcal{N}} H\left(x_{ij}\right)}, \\ &= \frac{1}{|\mathcal{P}|}\sum_{i\in\mathcal{P}}\sum_{j\in\mathcal{N}} L_{ij}^{\text{Rank}},\end{aligned} \tag{6}$$

where L_{ij}^{Rank}, coined as the primary term, is defined as

$$L_{ij}^{\text{Rank}} = \frac{H\left(x_{ij}\right)}{\sum_{k\in\mathcal{P}\cup\mathcal{N}} H\left(x_{ik}\right)}. \tag{7}$$

3.2. Sorting Positives with their Uncertainties

Edge detection datasets contain annotations from multiple annotators, leading to (aleatoric) uncertainty in ground truth labels. In RankED, we favor accurate prediction of positive edge pixels with high certainty over those with low certainty. To this end, we use the sorting objective introduced by RS Loss [36], where object detection hypotheses were sorted by their localization qualities.

Following RS Loss [36], we define our sorting loss as:

$$\mathcal{L}_{\text{Sort}} = \frac{1}{|\mathcal{P}|}\sum_{i\in\mathcal{P}} \left(\ell_{\text{Sort}}(i) - \ell^*_{\text{Sort}}(i)\right), \tag{8}$$

where $\ell_{\text{Sort}}(i)$ and $\ell^*_{\text{Sort}}(i)$ are a summation of current sorting error and target sorting error, respectively. We define the sorting error $\ell_{\text{Sort}}(i)$ as the average uncertainty of pixels with higher confidence than pixel i (i.e., $H(x_{ij}) = 1$):

$$\ell_{\text{Sort}}(i) = \frac{1}{\text{rank}^+(i)} \sum_{j\in\mathcal{P}} H\left(x_{ij}\right)(1 - c_j), \tag{9}$$

where $c_j \in [0, 1]$ is the label certainty of pixel j (explained in Section 3.3), and $\text{rank}^+(i)$ and $\mathrm{H}\left(x_{ij}\right)$ are as defined in Section 3.1.

The target error $\ell^*_{\text{Sort}}(i)$ is defined as the average uncertainty $(1 - c_j)$ over each positive pixel $j \in \mathcal{P}$ with higher confidence $(H(x_{ij}) = 1)$ and higher certainty $(c_j \geq c_i)$, calculated as:

$$\ell^*_{\text{Sort}}(i) = \frac{\sum_{j\in\mathcal{P}} \mathrm{H}\left(x_{ij}\right)\left[c_j \geq c_i\right](1 - c_j)}{\sum_{j\in\mathcal{P}} \mathrm{H}\left(x_{ij}\right)\left[c_j \geq c_i\right]}, \tag{10}$$

where $[P]$, the Iverson Bracket, is 1 if P is True and 0 otherwise.

We plug the definitions of $\ell_{\text{Sort}}(i)$ and $\ell^*_{\text{Sort}}(i)$ into $\mathcal{L}_{\text{Sort}}$ to obtain the primary terms as in AP Loss as follows:

$$L_{ij}^{\text{Sort}} = \left(\ell_{\text{Sort}}(i) - \ell^*_{\text{Sort}}(i)\right)\frac{H(x_{ij})[c_j > c_i]}{\sum_k H(x_{ik})[c_k > c_i]}, \tag{11}$$

which is zero if $i, j \notin \mathcal{P}$.

3.3. Computing Pixel Uncertainties

Previous studies create labels for the multi-label datasets as follows: (i) Merge n annotations $\{\mathbf{y}^a\}_{a=1}^n$ for one RGB image provided by n annotators: $\mathbf{y}^* \leftarrow \oplus_{a=1}^n \mathbf{y}^a$. (ii) Binarize $\mathbf{y}^*$ using a chosen threshold τ to obtain the training target $\mathbf{y}$:

$$y_i = \begin{cases} 0, & \text{for } y_i^* < \tau, \\ 1, & \text{otherwise}, \end{cases} \tag{12}$$

where $i \in \mathcal{P} \cup \mathcal{N}$.

This approach neglects both pixel-wise and label-wise uncertainties. Precise labeling of edges by hand is difficult. Most edge pixels annotated by humans do not overlap with low-level edges in RGB images. Hence, the evaluation procedure of edge detection tolerates localization error to match edges in predicted and ground-truth results [14]. Therefore, a simple merging operation ignores this pixel-wise uncertainty in training. Moreover, binarizing $\mathbf{y}^*$ not only leads to a loss of information about how many times edge pixels are labeled by multiple annotators but also causes a loss of edges that are rarely labeled among these annotations.

To this end, we propose calculating a certainty map $\mathbf{c}$ which preserves the level of agreement among annotators (Algorithm 1). For this, first we take the pixel-wise logical-OR of the annotations: $\tilde{\mathbf{y}} = \mathrm{OR}_{a=1}^n \mathbf{y}^a$ ($\tilde{y}_i$ for pixel i). Then, we define the certainty c_i for pixel i as an average of how many annotations match an edge pixel in its d-vicinity in $\{\mathbf{y}^a\}_{a=1}^n$:

$$c_i = \frac{1}{n}\sum_{a=1}^{n} \mathrm{CP}(\tilde{y}_i, \mathbf{y}^a, d), \tag{13}$$

where $\mathrm{CP}(\tilde{y}_i, \mathbf{y}^a, d)$ is a commonly used function in the edge detection benchmarks [1, 14] to find match predictions ($\tilde{y}_i \in \tilde{\mathbf{y}}$) with the ground truth ($\mathbf{y}^a$) within a d-Manhattan distance (CP: Correspond Pixels).

Algorithm 1 Computing the certainty map $\mathbf{c}$ of annotations.

Input: - Set of n annotations $\{\mathbf{y}^a\}_{a=1}^n$.
- Maximum distance tolerance for overlap: d.

Output: Certainty map, $\mathbf{c}$ (c_i: for pixel i).

1: $\tilde{\mathbf{y}} \leftarrow \mathbf{y}^1 \,\mathrm{OR}\, \mathbf{y}^2 \,\mathrm{OR} \ldots \mathrm{OR}\, \mathbf{y}^n$. ▷ Combine annotations.
2: **for** each edge pixel i in $\tilde{\mathbf{y}}$ **do**
3: $\quad c_i \leftarrow \frac{1}{n}\sum_{a=1}^n \mathrm{CP}(\tilde{y}_i, \mathbf{y}^a, d)$ ▷ Eq. 13.
4: **end for**

3.4. Overall Loss Function

We combine the ranking and sorting components as follows:

$$\mathcal{L}_{\text{Overall}} = \mathcal{L}_{\text{Rank}} + \alpha\mathcal{L}_{\text{Sort}}, \tag{14}$$

where α is a coefficient to control influence of sorting loss $\mathcal{L}_{\text{Sort}}$. In the case of a single-annotation dataset (i.e., $n = 1$), we are unable to calculate an uncertainty map / the sorting loss $\mathcal{L}_{\text{Sort}}$; therefore, we set α to 0 for such datasets.

3.5. Optimization

The counting-based definitions of $\mathcal{L}_{\text{Rank}}$ and $\mathcal{L}_{\text{Sort}}$ prohibit an autograd-based calculation of $\partial\mathcal{L}_o/\partial p_i$ where $o \in$ {Rank, Sort}. To address this issue, we follow the error-driven update mechanism of Chen et al. [6] that approximates such a gradient as follows:

$$\frac{\partial \mathcal{L}_o}{\partial p_i} = \sum_j L^o_{ji} t^o_{ji} - \sum_j L^o_{ij} t^o_{ij}, \tag{15}$$

where $t^o_{kl} = 1$ if $y_k = 1$ and $y_l = 0$ for $o =$ Rank; $t^o_{kl} = 1$ if $y_k = 1$ and $y_l = 1$ for $o =$ Sort. See Chen et al. [6] for the derivation details.

4. Experiments

Datasets. To evaluate our method, we carried out comprehensive experiments on three commonly used datasets:

BSDS500 [1] is one of the most popular datasets for evaluating edge detection. It officially contains 200 images for training, 100 images for validation, and 200 images for testing. Each image has 321 $\times$ 481 resolution and is manually labeled by at least five different annotators, which creates aleatoric uncertainty in the ground-truth labels.

NYUDv2 [43] contains 1449 paired RGB and Depth images with 576 $\times$ 448 resolution. Edge detection labels are extracted using the boundaries of segmented objects. It has a single ground-truth edge map for each RGB image. It officially contains 381 images for training, 414 for validation, and 654 images for testing.

Multi-cue [33] contains 100 natural images with 1280 $\times$ 720 resolution. Each image has six edge and five boundary labels. The dataset does not provide training and testing splits. We randomly allocated 80 images for training, and 20 images for testing following previous work [39, 55].

Implementation and Training Details. We use the MMSegmentation toolbox [7] to implement our method RankED. The official implementations of AP Loss [5] and RS Loss [36] are based on for-iterations, which lead to long training times. Therefore, we developed fully-vectorized implementations, boosting training $\sim$4.5$\times$ compared to iteration-based ones. While the fully-vectorized implementation requires huge GPU memory – at least 45 GB for 320$\times$320 input resolution with the base model of Swin-Transformer [30], we also provide a semi-vectorized implementation which speeds up about 2.5$\times$ and requires 20 GB of memory for the same input and model settings.

For the NYUD-v2 dataset, we use the official training and validation sets for training. We apply the following data

Method		SS			MS		
		ODS	OIS	AP	ODS	OIS	AP
Canny [4]	(PAMI'86)	.611	.676	.520	-	-	-
gPb-UCM [1]	(PAMI'10)	.729	.755	.745	-	-	-
SCG [48]	(NeurIPS'12)	.739	.758	.773	-	-	-
SE [14]	(PAMI'14)	.743	.764	.800	-	-	-
OEF [18]	(CVPR'15)	.746	.770	.815	-	-	-
DeepEdge [2]	(CVPR'15)	.753	.772	.807	-	-	-
DeepContour [42]	(CVPR'15)	.757	.776	.790	-	-	-
HED [49]	(ICCV'15)	.788	.808	.840	-	-	-
DeepBoundary [26]	(ICLR'15)	.789	.811	.789	.803	.820	.848
CEDN [52]	(CVPR'16)	.788	.804	-	-	-	-
RDS [28]	(CVPR'16)	.792	.810	.818	-	-	-
COB [31]	(ECCV'16)	.793	.820	.859	-	-	-
AMH-Net [50]	(NeurIPS'17)	.798	.829	.869	-	-	-
RCF [29]	(CVPR'17)	.798	.815	-	-	-	-
CED [47]	(CVPR'17)	.803	.820	.871	-	-	-
LPCB [10]	(ECCV'18)	.800	.816	-	-	-	-
BDCN [20]	(CVPR'19)	.806	.826	.847	-	-	-
DSCD [9]	(ACMMM'20)	.802	.817	-	-	-	-
FCL-Net [51]	(NN'22)	.807	.822	-	.816	.833	-
EDTER [39]	(CVPR'22)	$\underline{.824}$	$\underline{.841}$	.880	**.840**	**.858**	.896
UAED [55]	(CVPR'23)	**.829**	**.847**	$\underline{.892}$	$\underline{.837}$	$\underline{.855}$	.897
ACTD [12]	(Neurocomp.'23)	.817	.836	.839	-	-	-
RankED (Ranking Only)		.822	.838	.886	.829	.850	$\underline{.900}$
RankED (Ranking & Sorting)		$\underline{.824}$	.840	**.895**	$\underline{.837}$	$\underline{.855}$	**.911**

Table 2. Quantitative results on BSDS dataset [1]. SS and MS represent single-scale and multi-scale, respectively. Bold: Best result. Underline: Second-best results.

Method		SS+VOC			MS+VOC		
		ODS	OIS	AP	ODS	OIS	AP
LDC [11] (	ACMMM'21)	.812	.826	.857	.819	.834	.860
PiDiNet [44]	(ICCV'21)	.807	.823	-	-	-	-
FCL-Net [51]	(NN'22)	.815	.834	-	.826	.845	-
EDTER [39]	(CVPR'22)	.832	.847	.886	**.848**	**.865**	.903
UAED [55]	(CVPR'23)	**.838**	**.855**	**.902**	$\underline{.844}$	$\underline{.864}$	$\underline{.905}$
CHRNet [15]	(Pat. Rec.'23)	.787	.788	.801	.830	.853	.870
ACTD [12]	(Neurocomp.'23)	.821	.837	.850	.826	.842	.854
RankED (Ranking Only)		$\underline{.833}$	$\underline{.848}$	$\underline{.901}$	$\underline{.844}$	.860	**.916**

Table 3. Quantitative results on BSDS dataset [1] using extra Pascal Context Data [16] in training. SS and MS represent single-scale and multi-scale, respectively. The best and second-best results are shown with bold and underlined texts, respectively.

augmentations like in previous studies [39, 49]: (i) Scaling with 0.5 and 1.5, (ii) horizontal flip, and (iii) rotation with degrees 90, 180, and 270.

For the BSDS500 dataset, we use the official training and validation sets for training. By following the literature [20, 39], we apply horizontal flip, rotation with 16 different angles between [0,360], and scaling with 0.5 and 1.5. For experiments on using additional training data, we use the Pascal Context Dataset [16].

For the Multi-cue dataset, we randomly choose 80 images for training, and the remaining images for testing. We repeat this procedure 3 times and report their average with standard deviation. We apply horizontal flip and rotation with 90, 180, and 360 degrees as the data augmentation.

For all datasets, we randomly crop RGB images to 320 $\times$ 320 resolution. We do not use any threshold to select positive pixels (i.e., $\tau = 0$). We take the δ parameter in $H(\cdot)$ (Eq. 5) as 0.4 for NYUD-v2 and 0.1 for BSDS and Multi-cue datasets in $\mathcal{L}_{\text{Rank}}$. In $\mathcal{L}_{\text{Sort}}$, δ is used as 0.1 for BSDS and Multi-cue datasets. The learning rate is 1e-6 for BSDS and 1e-5 for NYUD-v2 & Multicue datasets.

The number of iterations is 200k for NYUD-v2 and BSDS datasets and 140k for Multicue dataset because of the risk of overfitting. Batch size is 1. The remaining settings of the optimizer are the same as the base model of Swin-Transformer [30] for the segmentation task. Also, we use pre-trained weights of ImageNet-22k [8] on 384x384 images. The sorting loss coefficient α (Eq. 14) is used as 1 and 2 for Multicue and BSDS datasets, respectively.

For testing, we followed the standard edge detection evaluation protocol [13] which includes non-maximum suppression (NMS) and skeleton-based edge thinning, respectively. Also, we set the maximum allowed distance (d in Eq. 13) as 0.011 for NYUD-v2, 0.0075 for BSDS500 & Multicue datasets similar to previous studies [20, 39, 49].

Performance Measures. We use the following three commonly used measures: Optimal Dataset Scale (ODS), Optimal Image Scale (OIS), and Average Prevision (AP). To calculate ODS, one threshold is used for binarizing the predicted edge maps in the dataset, whereas different thresholds are used for each image in OIS. Also, ODS and OIS are essentially F_1 scores, showing the best result at only one point on the precision-recall curve. On the other hand, AP gives an insight into the general performance of the model.

Unlike previous studies, we present uncertainty-aware results (UaR) using these measures for multi-label datasets. In this type of result, we separately report these three measures for different values of certainty (e.g., AP @ $c_i \geq 0.2$, which means AP if c_i is thresholded at 0.2).

4.1. Experiment 1: Comparison with SOTA

This section compares RANKED with the state-of-the-art (SOTA) methods including hand-crafted and deep-learning-based methods. We report the results of SOTA methods from their publications.

On BSDS500. We compare the performance of our model with SOTA methods, as shown in Tables 2 and 3, for different input and training data settings: Single Scale (SS), Multi-scale (MS), with additional Pascal-VOC dataset in Single Scale (SS+VOC), and with additional Pascal-VOC dataset in Multi-Scale (MS+VOC). The results suggest that our method provides the best AP for all input and training data settings – with only one exception at SS+VOC, where RANKED is 0.1 AP points behind UAED [55]. RANKED's ODS and OIS performances are on par with others. Since the Pascal VOC dataset is not a multi-label dataset (i.e. no uncertainty), we cannot apply the sorting loss on it.

On NYUD-v2. We provide results for RGB images in Table 4. Our RANKED achieves the best results in all three measures: 78.0% ODS, 79.3% OIS, 82.6% AP. HHA and RGB-HHA results are reported in the supplementary material.

Method		ODS	OIS	AP
gPb-ucm [1]	(PAMI'11)	.632	.661	.562
Silberman et al. [43]	(ECCV'12)	.658	.661	-
gPb+NG [17]	(CVPR'13)	.687	.716	.629
OEF [18]	(CVPR'15)	.651	.667	-
HED [49]	(ICCV'15)	.720	.734	.734
RCF [29]	(CVPR'17)	.729	.742	-
AMH-Net [50]	(NeurIPS'17)	.744	.758	.765
LPCB [10]	(ECCV'18)	.739	.754	-
BDCN [20]	(CVPR'19)	.748	.763	.770
PiDiNet [44]	(ICCV'21)	.733	.747	.715
EDTER [39]	(CVPR'22)	<u>.774</u>	<u>.789</u>	<u>.797</u>
ACTD [12]	(Neurocomp.'23)	.762	.774	-
RANKED (R)		**.780**	**.793**	**.826**

Table 4. Quantitative comparisons on NYUD-v2 [43] for RGB images. The best and second-best results are shown with bold and underlined texts, respectively. R: Ranking only.

On Multicue. We compare our method with SOTA deep learning-based methods using both edge and boundary labels of Multi-cue dataset, as shown in Table 5. RANKED clearly outperforms all SOTA methods for edge and boundary detection in all metrics. On edge detection, it yields +5.5, +4.3, and +2.3 percentage point improvements over the second-best results for ODS, OIS, and AP metrics, respectively. Similarly, on boundary, it gives +9.9, +9.5, and +6.8 percentage point improvements over the second-best results in ODS, OIS, and AP metrics, respectively.

4.2. Experiment 2: Uncertainty-aware Evaluation

This section provides uncertainty-aware evaluation (UaR) of RANKED and SOTA methods which published their weights on the BSDS dataset as shown in Table 6. As most studies published their models for only the BSDS dataset, we could not do a similar analysis on the Multi-cue dataset.

For the BSDS dataset, we define six uncertainty levels. For example, level $\mathbf{c} > 0.0$ contains all ground-truth edges (high + low uncertainties) in all labels. Therefore, this case is equivalent to the standard evaluation of edge detectors. On the other hand, level $\mathbf{c} = 1.0$ (lowest uncertainty) contains only ground-truth edges which are labeled by all annotators.

The results in Table 6 suggest a similar behavior for all models: They are more successful on low-uncertainty edges than high-uncertainty ones. While the uncertainty level is increased (i.e., $\mathbf{c}$ goes from 1.0 to 0.0), the scores of ODS, OIS, and AP measures decrease.

Our method RANKED gives better results when the uncertainty level is decreased. For example, our method gives

Method		ODS	OIS	AP
Human [33]	(VR'16)	.750±.024	-	-
Multicue [33]	(VR'16)	.830±.002	-	-
HED [49]	(ICCV'15)	.851±.014	.864±.011	-
RCF [29]	(CVPR'17)	.857±.004	.862±.004	-
BDCN [20]	(CVPR'19)	.891±.001	.898±.002	.935±.002
DSCD [9]	(ACMMM'20)	.871±.007	.876±.002	-
LDC [11]	(ACMMM'21)	.881±.012	.893±.011	-
PiDiNet [44]	(ICCV'21)	.855±.007	.860±.005	-
FCL-Net [51]	(NN'22)	.875±.005	.880±.005	-
EDTER [39]	(CVPR'22)	.894±.005	.900±.003	.944±.002
UAED [55]	(CVPR'23)	.895±.002	.902±.001	.949±.002
CHRNet [15]	(Pat. Rec.'23)	.907	.922	-
ACTD [12]	(Neurocomp.'23)	.890±.011	.905±.009	-
RANKED (R)		<u>.951±.002</u>	<u>.953±.001</u>	<u>.962±.004</u>
RANKED ($R+S$)		**.962±.003**	**.965±.003**	**.973±.006**

(a) Edge

Method		ODS	OIS	AP
Human [33]	(VR'16)	.760±.017	-	-
Multicue [33]	(VR'16)	.720±.014	-	-
HED [49]	(ICCV'15)	.814±.011	.822±.008	.869±.015
RCF [29]	(CVPR'17)	.817±.004	.825±.005	-
BDCN [20]	(CVPR'19)	.836±.001	.846±.003	.893±.001
DSCD [9]	(ACMMM'20)	.828±.003	.835±.004	-
LDC [11]	(ACMMM'21)	.839±.012	.853±.006	-
PiDiNet [44]	(ICCV'21)	.818±.003	.830±.005	-
FCL-Net [51]	(NN'22)	.834±.016	.840±.016	-
EDTER [39]	(CVPR'22)	.861±.003	.870±.004	.919±.003
UAED [55]	(CVPR'23)	.864±.004	.872±.006	.927±.006
CHRNet [15]	(Pat. Rec.'23)	.859	.863	-
ACTD [12]	(Neurocomp.'23)	.852 ±.004	.863±.008	-
RANKED (R)		<u>.954 ±.004</u>	<u>.958± .005</u>	<u>.992 ±.002</u>
RANKED ($R+S$)		**.963 ±.002**	**.967±.002**	**.995±.001**

(b) Boundary

Table 5. Quantitative results on Multicue Dataset [33]. The best and second-best results are shown with bold and underlined texts, respectively. R: Ranking only. $R+S$: Ranking & Sorting.

nearly the same performance with EDTER [39] for the ODS and OIS measures in level $\mathbf{c} > 0.0$, and the performance of our model increases, as the uncertainty level decreases. This can be explained by our sorting task because it assigns higher importance to lower uncertainty pixels.

This set of experiments and our uncertainty-aware results (UaR) reveal that edge detection models give a better performance when the uncertainty levels are decreased and most of the performance loss is caused by higher uncertainty pixels, especially labeled by only one annotator. Therefore, focusing on those pixels can provide more performance gain for the future study. UaR results of Multi-cue dataset are reported in the supplementary material.

4.3. Experiment 3: Ablation Analysis

Different loss functions. To show the effectiveness of our method, we perform step-by-step comparisons by including class-balanced cross-entropy and the combination of cross-entropy and dice losses [10] with the same model and input settings on NYUD-v2, BSDS, and Multi-cue datasets. The results in Table 7 suggest that ranking systematically improves the results over the commonly used loss functions in the literature. Moreover, we see that, whenever feasible, using the sorting component provides gains compared to only using ranking.

Our certainty map computation vs. label averaging. We also conduct an experiment to show the efficiency of our certainty map $\mathbf{c}$ on BSDS dataset. In this experiment, as shown in Table 8, we use pixel-wise averaging and use it as the certainty map ($\mathbf{c} \leftarrow \sum_a \mathbf{y}^a$). In this experiment, all settings are the same except for the labels used. Our proposed certainty map $\mathbf{c}$ gives better performance than the standard labels obtained by pixel-wise averaging in all metrics because we consider pixel-wise uncertainty using a dataset-specific distance toleration like an evaluation protocol of edge detection. In this way, we overlap edges in multi-label that do not overlap in case of pixel-wise averaging and these overlaps provide labels that are more informative to the sorting task in terms of uncertainty.

Prioritize Uncertain Pixels. Our sorting loss prioritizes low-uncertainty pixels while Zhou et al. [55] give higher priority to high-uncertainty edges. To validate our approach, we integrate their uncertainty weighting scheme into our loss function as follows: $(1 - c_i) \cdot \mathcal{L}(i)$ for pixel i. The results in Table 8 suggest that this significantly drops the performance of our model. Therefore, giving more importance to low-uncertainty edges provides a better performance in RANKED.

4.4. Experiment 4: Qualitative Comparison

We also provide a qualitative comparison between SOTA models and ours on BSDS dataset, as shown in Figure 3. These results are presented after the post-processing step using thresholds of OIS measure. While our model detects prominent edges better, it may miss low-level details. See the Supp. Mat. for more results.

4.5. Experiment 5: Running Time Comparisions

Although ranking-based loss functions give superior performance [6, 36], they suffer from long training time due to for-loops in their official implementations. Our vectorized implementations solve this problem as shown in Table 9. For a fair comparison, we test their execution time using loss functions with random tensors. Our vectorized and semi-vectorized implementations of $\mathcal{L}_{\text{Rank}}$ and $\mathcal{L}_{\text{Sort}}$ are almost 100 and 90 times, respectively, faster than their official implementations. However, when they are integrated into models, we observe less performance increase due to other bottlenecks such as forward/backward in models.

5. Conclusion

We presented a ranking-based solution for object contour detection, or as widely referred to as edge detection, to address both the imbalance problem between positive (edge)

Method	Low Certainty (*High Uncertainty*) ⟵															⟶ High Certainty (*Low Uncertainty*)		
	$\mathbf{c} > 0.0$			$\mathbf{c} \geq 0.2$			$\mathbf{c} \geq 0.4$			$\mathbf{c} \geq 0.6$			$\mathbf{c} \geq 0.8$			$\mathbf{c} = 1.0$		
	ODS	OIS	AP	ODS	OIS	AP	ODS	OIS	AP	ODS	OIS	AP	ODS	OIS	AP	ODS	OIS	AP
HED [49] (ICCV'15)	.788	.804	.840	.794	.810	.846	.812	.829	.864	.833	.850	.887	.852	.868	.902	.880	.893	.926
BDCN [20] (CVPR'19)	.815	.830	.872	.821	.837	.880	.842	.859	.899	.864	.881	.919	.884	.900	.933	.914	.926	.956
PiDiNet [44] (ICCV'21)	.807	.823	.856	.810	.826	.864	.829	.846	.883	.851	.867	.904	.871	.887	.923	.899	.911	.947
EDTER [39] (CVPR'22)	**.824**	**.841**	.880	.826	.842	.881	.848	.864	.899	.870	.885	.918	.890	.905	.936	.920	**.932**	.962
RankED (R)	.822	.838	.886	.828	.844	.890	.849	.866	.909	.870	.886	.924	.889	.904	.939	.916	.923	.962
RankED ($R+S$)	**.824**	.840	**.895**	**.831**	**.847**	**.900**	**.854**	**.870**	**.918**	**.877**	**.892**	**.936**	**.897**	**.911**	**.951**	**.926**	.929	**.970**

Table 6. Uncertainty-aware results on BSDS dataset. All SOTA results are computed using official weights. Also, c represents the certainty map mentioned in Algorithm 1. While case $c > 0.0$ contains all ground-truth edges in all labels, case $c = 1.0$ contains only ground-truth edges that are labeled in all labels. R: Only ranking, $R+S$: Ranking & Sorting.

Dataset	Loss	ODS	OIS	AP
NYUD-v2	CE_{CB}	.775	.789	.802
	CE + DICE	.779	.791	.807
	RankED (R)	**.780**	**.793**	**.826**
BSDS	CE_{CB}	.820	.831	.871
	CE + DICE	.821	.836	.872
	RankED (R)	.822	.838	.886
	RankED ($R+S$)	**.824**	**.840**	**.895**
Multicue Edge	CE_{CB}	.926±.006	.926±.007	.880±.001
	CE + DICE	.930±.005	.931±.004	.883±.006
	RankED (R)	.951±.002	.953±.001	.962±.004
	RankED ($R+S$)	**.962±.003**	**.965±.003**	**.973±.006**
Multicue Boundary	CE_{CB}	.934±.004	.943±.003	.981±.003
	CE + DICE	.947±.004	.954±.004	.984±.004
	RankED (R)	.954 ±.004	.958± .005	.992 ±.002
	RankED ($R+S$)	**.963 ±.002**	**.967±.002**	**.995±.001**

Table 7. Quantitative comparison between our method and commonly used loss functions in edge detection. Bold numbers show the best results for corresponding columns and datasets. While Ours_R represents using only ranking loss, R: Ranking. $R+S$: Ranking & Sorting. CE_CB: Class-balanced cross-entropy.

Label Type	ODS	OIS	AP
Label Averaging as Certainty ($\mathbf{c} \leftarrow \sum_a \mathbf{y}^a$)	0.818	0.831	0.880
Loss Weighting with Uncertainty $(1 - c_i)$	0.789	0.813	0.849
RankED	**0.824**	**0.840**	**0.895**

Table 8. Comparison with pixel-wise averaging of labels as certainty, and uncertainty-weighted loss.

Implementations	Loss	Exec. Time (msec/img)
-	CE	0.32
-	CE+DICE	0.39
For-iterations	RankED (R)	142.79
	RankED ($R+S$)	1530.22
Semi-vectorized	RankED (R)	2.90
	RankED ($R+S$)	16.73
Vectorized	RankED (R)	1.00
	RankED ($R+S$)	14.09

Table 9. Execution time comparisons (avg. of 100 runs).

and negative (non-edge) classes and uncertainty arising from disagreement between different annotators. Our solution contained two novel loss functions: One for ranking positive edge pixels over negatives, and another for sorting positive pixels with respect to their edge certainties. Our extensive experiments show the efficiency of our proposal on NYUD-v2, BSDS, and Multicue datasets. RankED can train an edge detector maximize AP (via its the ranking loss). Our paper paves the way for optimizing other objectives and making all post processing steps as part of the training process.

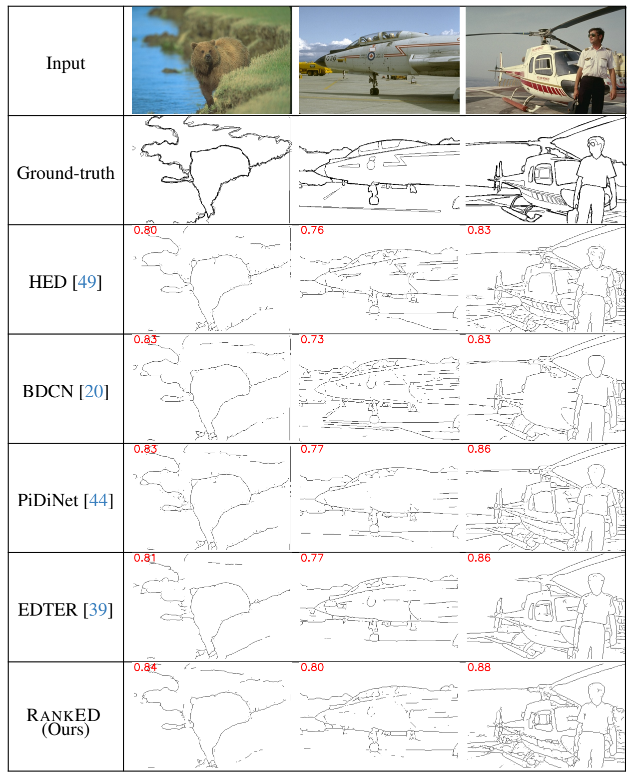

Figure 3. Qualitative results on BSDS dataset. All outputs are obtained after the post-processing step. Red: OIS scores.

Acknowledgements

We gratefully acknowledge the computational resources provided by METU-ROMER, Center for Robotics and Artificial Intelligence, Middle East Technical University. Dr. Akbas is supported by the "Young Scientist Awards Program (BAGEP)" of Science Academy, Türkiye.

References

[1] Pablo Arbelaez, Michael Maire, Charless Fowlkes, and Jitendra Malik. Contour detection and hierarchical image segmentation. *IEEE Trans. Pattern Anal. Mach. Intell.*, 33(5):898–916, 2010.

[2] Gedas Bertasius, Jianbo Shi, and Lorenzo Torresani. Deepedge: A multi-scale bifurcated deep network for top-down contour detection. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 4380–4389, 2015.

[3] Gedas Bertasius, Jianbo Shi, and Lorenzo Torresani. High-for-low and low-for-high: Efficient boundary detection from deep object features and its applications to high-level vision. In *Int. Conf. Comput. Vis.*, pages 504–512, 2015.

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

[5] Kean Chen, Jianguo Li, Weiyao Lin, John See, Ji Wang, Lingyu Duan, Zhibo Chen, Changwei He, and Junni Zou. Towards accurate one-stage object detection with ap-loss. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 5119–5127, 2019.

[6] Kean Chen, Weiyao Lin, Jianguo Li, John See, Ji Wang, and Junni Zou. AP-loss for accurate one-stage object detection. *IEEE Trans. Pattern Anal. Mach. Intell.*, 43(11):3782–3798, 2020.

[7] MMSegmentation Contributors. MMSegmentation: Openmmlab semantic segmentation toolbox and benchmark. https://github.com/open-mmlab/mmsegmentation, 2020.

[8] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 248–255. IEEE, 2009.

[9] Ruoxi Deng and Shengjun Liu. Deep structural contour detection. In *Proceedings of the 28th ACM international conference on multimedia*, pages 304–312, 2020.

[10] Ruoxi Deng, Chunhua Shen, Shengjun Liu, Huibing Wang, and Xinru Liu. Learning to predict crisp boundaries. In *Eur. Conf. Comput. Vis.*, pages 562–578, 2018.

[11] Ruoxi Deng, Shengjun Liu, Jinxin Wang, Huibing Wang, Hanli Zhao, and Xiaoqin Zhang. Learning to decode contextual information for efficient contour detection. In *Proceedings of the 29th ACM International Conference on Multimedia*, pages 4435–4443, 2021.

[12] Ruoxi Deng, Zhao-Min Chen, Huiling Chen, and Jie Hu. Learning to refine object boundaries. *Neurocomputing*, 557:126742, 2023.

[13] Piotr Dollár and C. Lawrence Zitnick. Structured forests for fast edge detection. In *Int. Conf. Comput. Vis.*, 2013.

[14] Piotr Dollár and C Lawrence Zitnick. Fast edge detection using structured forests. *IEEE Trans. Pattern Anal. Mach. Intell.*, 37(8):1558–1570, 2014.

[15] Omar Elharrouss, Youssef Hmamouche, Assia Kamal Idrissi, Btissam El Khamlichi, and Amal El Fallah-Seghrouchni. Refined edge detection with cascaded and high-resolution convolutional network. *Pattern Recognition*, 138:109361, 2023.

[16] Mark Everingham, Luc Van Gool, Christopher KI Williams, John Winn, and Andrew Zisserman. The pascal visual object classes (voc) challenge. *Int. J. Comput. Vis.*, 88:303–338, 2010.

[17] Saurabh Gupta, Pablo Arbelaez, and Jitendra Malik. Perceptual organization and recognition of indoor scenes from rgb-d images. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 564–571, 2013.

[18] Sam Hallman and Charless C Fowlkes. Oriented edge forests for boundary detection. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 1732–1740, 2015.

[19] Ali Harakeh, Michael Smart, and Steven L Waslander. Bayesod: A bayesian approach for uncertainty estimation in deep object detectors. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 87–93. IEEE, 2020.

[20] Jianzhong He, Shiliang Zhang, Ming Yang, Yanhu Shan, and Tiejun Huang. Bi-directional cascade network for perceptual edge detection. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 3828–3837, 2019.

[21] Julia Hornauer and Vasileios Belagiannis. Gradient-based uncertainty for monocular depth estimation. In *Eur. Conf. Comput. Vis.*, pages 613–630, 2022.

[22] Po-Yu Huang, Wan-Ting Hsu, Chun-Yueh Chiu, Ting-Fan Wu, and Min Sun. Efficient uncertainty estimation for semantic segmentation in videos. In *Eur. Conf. Comput. Vis.*, pages 520–535, 2018.

[23] André Peter Kelm, Vijesh Soorya Rao, and Udo Zölzer. Object contour and edge detection with refinecontournet. In *Computer Analysis of Images and Patterns: 18th International Conference*, pages 246–258, 2019.

[24] Alex Kendall and Yarin Gal. What uncertainties do we need in bayesian deep learning for computer vision? *Adv. Neural Inform. Process. Syst.*, 30, 2017.

[25] Josef Kittler. On the accuracy of the sobel edge detector. *Image and Vision Computing*, 1(1):37–42, 1983.

[26] Iasonas Kokkinos. Pushing the boundaries of boundary detection using deep learning. *arXiv preprint arXiv:1511.07386*, 2015.

[27] Florian Kraus and Klaus Dietmayer. Uncertainty estimation in one-stage object detection. In *2019 ieee*

intelligent transportation systems conference (itsc), pages 53–60. IEEE, 2019.

[28] Yu Liu and Michael S Lew. Learning relaxed deep supervision for better edge detection. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 231–240, 2016.

[29] Yun Liu, Ming-Ming Cheng, Xiaowei Hu, Kai Wang, and Xiang Bai. Richer convolutional features for edge detection. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 3000–3009, 2017.

[30] Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin transformer: Hierarchical vision transformer using shifted windows. In *Int. Conf. Comput. Vis.*, pages 10012–10022, 2021.

[31] Kevis-Kokitsi Maninis, Jordi Pont-Tuset, Pablo Arbeláez, and Luc Van Gool. Convolutional oriented boundaries. In *Eur. Conf. Comput. Vis.*, pages 580–596, 2016.

[32] David Marr and Ellen Hildreth. Theory of edge detection. *Proceedings of the Royal Society of London. Series B. Biological Sciences*, 207(1167):187–217, 1980.

[33] David A Mély, Junkyung Kim, Mason McGill, Yuliang Guo, and Thomas Serre. A systematic comparison between visual cues for boundary detection. *Vision Research*, 120:93–107, 2016.

[34] Kemal Oksuz, Baris Can Cam, Emre Akbas, and Sinan Kalkan. A ranking-based, balanced loss function unifying classification and localisation in object detection. *Adv. Neural Inform. Process. Syst.*, 33: 15534–15545, 2020.

[35] Kemal Oksuz, Baris Can Cam, Sinan Kalkan, and Emre Akbas. Imbalance problems in object detection: A review. *IEEE Trans. Pattern Anal. Mach. Intell.*, 43 (10):3388–3415, 2020.

[36] Kemal Oksuz, Baris Can Cam, Emre Akbas, and Sinan Kalkan. Rank & sort loss for object detection and instance segmentation. In *Int. Conf. Comput. Vis.*, pages 3009–3018, 2021.

[37] Matteo Poggi, Filippo Aleotti, Fabio Tosi, and Stefano Mattoccia. On the uncertainty of self-supervised monocular depth estimation. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 3227–3237, 2020.

[38] Mengyang Pu, Yaping Huang, Qingji Guan, and Haibin Ling. Rindnet: Edge detection for discontinuity in reflectance, illumination, normal and depth. In *Int. Conf. Comput. Vis.*, pages 6879–6888, 2021.

[39] Mengyang Pu, Yaping Huang, Yuming Liu, Qingji Guan, and Haibin Ling. Edter: Edge detection with transformer. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 1402–1412, 2022.

[40] Qi Qian, Lei Chen, Hao Li, and Rong Jin. DR loss: Improving object detection by distributional ranking. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 12164–12172, 2020.

[41] Michal Rolínek, Vít Musil, Anselm Paulus, Marin Vlastelica, Claudio Michaelis, and Georg Martius. Optimizing rank-based metrics with blackbox differentiation. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 7620–7630, 2020.

[42] Wei Shen, Xinggang Wang, Yan Wang, Xiang Bai, and Zhijiang Zhang. Deepcontour: A deep convolutional feature learned by positive-sharing loss for contour detection. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 3982–3991, 2015.

[43] Nathan Silberman, Derek Hoiem, Pushmeet Kohli, and Rob Fergus. Indoor segmentation and support inference from rgbd images. In *Eur. Conf. Comput. Vis.*, pages 746–760. Springer, 2012.

[44] Zhuo Su, Wenzhe Liu, Zitong Yu, Dewen Hu, Qing Liao, Qi Tian, Matti Pietikäinen, and Li Liu. Pixel difference networks for efficient edge detection. In *Int. Conf. Comput. Vis.*, pages 5117–5127, 2021.

[45] Carole H Sudre, Wenqi Li, Tom Vercauteren, Sebastien Ourselin, and M Jorge Cardoso. Generalised dice overlap as a deep learning loss function for highly unbalanced segmentations. In *Deep Learning in Medical Image Analysis and Multimodal Learning for Clinical Decision Support: Third International Workshop, DLMIA 2017, and 7th International Workshop, ML-CDS 2017, Held in Conjunction with MICCAI 2017, Québec City, QC, Canada, September 14, Proceedings 3*, pages 240–248. Springer, 2017.

[46] Le Wang, Meng Han, Xiaojuan Li, Ni Zhang, and Haodong Cheng. Review of classification methods on unbalanced data sets. *IEEE Access*, 9:64606–64628, 2021.

[47] Yupei Wang, Xin Zhao, and Kaiqi Huang. Deep crisp boundaries. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 3892–3900, 2017.

[48] Ren Xiaofeng and Liefeng Bo. Discriminatively trained sparse code gradients for contour detection. *Adv. Neural Inform. Process. Syst.*, 25, 2012.

[49] Saining Xie and Zhuowen Tu. Holistically-nested edge detection. In *Int. Conf. Comput. Vis.*, pages 1395–1403, 2015.

[50] Dan Xu, Wanli Ouyang, Xavier Alameda-Pineda, Elisa Ricci, Xiaogang Wang, and Nicu Sebe. Learning deep structured multi-scale features using attention-gated crfs for contour prediction. *Adv. Neural Inform. Process. Syst.*, 30, 2017.

[51] Wenjie Xuan, Shaoli Huang, Juhua Liu, and Bo Du. Fcl-net: Towards accurate edge detection via fine-scale corrective learning. *Neural Networks*, 145:248–259, 2022.

[52] Jimei Yang, Brian Price, Scott Cohen, Honglak Lee, and Ming-Hsuan Yang. Object contour detection with a fully convolutional encoder-decoder network. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 193–202, 2016.

[53] Yifan Zhang, Bingyi Kang, Bryan Hooi, Shuicheng Yan, and Jiashi Feng. Deep long-tailed learning: A survey. *IEEE Trans. Pattern Anal. Mach. Intell.*, 2023.

[54] Zhedong Zheng and Yi Yang. Rectifying pseudo label learning via uncertainty estimation for domain adaptive semantic segmentation. *Int. Conf. Comput. Vis.*, 129(4):1106–1120, 2021.

[55] Caixia Zhou, Yaping Huang, Mengyang Pu, Qingji Guan, Li Huang, and Haibin Ling. The treasure beneath multiple annotations: An uncertainty-aware edge detector. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 15507–15517, 2023.