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 NYUDv2, 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 pixels1. 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.

†Equal contribution.

1Although 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.

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-
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 hand-crafted 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 (P) and negatives (N), CE Loss is often weighted with a class balancing (CB) factor, e.g. as follows [49]:

\[
L_{CE} = \sum_{i} -\beta y_{i} \log(p_{i}) - (1 - \beta)(1 - y_{i}) \log(1 - p_{i}),
\]

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: \(\beta = |N|/|N \cup P|, (1 - \beta) = |P|/|N \cup P|\). 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]:

\[
L_{DICE} = \frac{\sum_{i} p_{i}^{2} + \sum_{i} y_{i}^{2}}{2 \sum_{i} p_{i} y_{i}},
\]

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

\[
L_{final} = \alpha L_{DICE} + \beta L_{CE},
\]

where \(\alpha\) and \(\beta\) 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.
of the negative pixels, we adapt the ranking-based loss function to rank the scores of positive edge pixels higher than those of negative ones.

3.1. Ranking Positive Edge Pixels over Negatives

Sorting Positives with Respect to their Uncertainties

Ranking Positives over Negatives ($\mathcal{L}_{\text{Rank}}$): $\mathcal{L}_{\text{Rank}}$ is a ranking-based solution with two novel components for edge detection (Figure 2):

- Ranking Positives over Negatives ($\mathcal{L}_{\text{Rank}}$): $\mathcal{L}_{\text{Rank}}$ 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. $\mathcal{L}_{\text{Sort}}$ 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]):

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

(4)

where $\text{rank}^+(i)$ is the rank of pixel $i$ among positives ($P$) defined as $\text{rank}^+(i) = \sum_{j \in P} H(x_{ij})$ 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 $\delta$-approximation around the step, as suggested by Chen et al.:

$$
H(x) = \begin{cases} 
0, & x < -\delta \\
\frac{x}{2\delta} + 0.5, & -\delta \leq x \leq \delta \\
1, & \delta < x.
\end{cases}
$$

(5)
Plugging these definitions into Eq. 4 yields:

\[ L_{\text{Rank}} = 1 - \frac{1}{|\mathcal{P}|} \sum_{i \in \mathcal{P}} \sum_{j \in \mathcal{P}} H(x_{ij}) \]

\[ = \frac{1}{|\mathcal{P}|} \sum_{i \in \mathcal{P}} \sum_{j \in \mathcal{N}} L_{ij}^{\text{Rank}}, \tag{6} \]

where \( L_{ij}^{\text{Rank}} \), coined as the primary term, is defined as

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

### 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:

\[ L_{\text{Sort}} = \frac{1}{|\mathcal{P}|} \sum_{i \in \mathcal{P}} (\ell_{\text{Sort}}(i) - \ell_{\text{Sort}}^*(i)), \tag{8} \]

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., \( H(x_{ij}) = 1 \)):

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

where \( c_j \in [0, 1] \) is the label certainty of pixel \( j \) (explained in Section 3.3), and \( \text{rank}^+(i) \) and \( H(x_{ij}) \) 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:

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

where \( |\mathcal{P}| \), the Iverson Bracket, is 1 if \( \mathcal{P} \) is True and 0 otherwise.

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

\[ L_{ij} = (\ell_{\text{Sort}}(i) - \ell_{\text{Sort}}^*(i)) \frac{H(x_{ij})[c_j \geq c_i]}{\sum_{k} H(x_{ik})[c_k \geq c_i]}, \tag{11} \]

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 \( \{y^n\}_{n=1}^N \) for one RGB image provided by \( n \) annotators: \( y^* \leftarrow \oplus_{a=1}^n y^n \). (ii) Binarize \( y^* \) using a chosen threshold \( \tau \) to obtain the training target \( y \):

\[ y_i = \begin{cases} 0, & \text{for } y^*_i < \tau, \\ 1, & \text{otherwise}. \end{cases} \]

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 \( y^* \) 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 \( c \) which preserves the level of agreement among annotators (Algorithm 1). For this, first we take the pixel-wise logical-OR of the annotations: \( \tilde{y} = \text{OR}_{a=1}^n y^a (\tilde{y}_i \text{ for pixel } i) \). 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 \( \{y^n\}_{n=1}^N \):

\[ c_i = \frac{1}{n} \sum_{a=1}^n \text{CP}(\tilde{y}_i, y^a, d), \tag{13} \]

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

**Algorithm 1** Computing the certainty map \( c \) of annotations.

**Input:** - Set of \( n \) annotations \( \{y^n\}_{n=1}^N \).
- Maximum distance tolerance for overlap: \( d \).

**Output:** Certainty map, \( c \) (for pixel \( i \)).

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

### 3.4. Overall Loss Function

We combine the ranking and sorting components as follows:

\[ L_{\text{Overall}} = L_{\text{Rank}} + \alpha L_{\text{Sort}}. \tag{14} \]

3242
where \( \alpha \) is a coefficient to control influence of sorting loss \( L_{\text{Sort}} \). In the case of a single-annotation dataset (i.e., \( n = 1 \)), we are unable to calculate an uncertainty map \( \partial L / \partial p \); therefore, we set \( \alpha = 0 \) for such datasets.

### 3.5. Optimization

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

\[
\frac{\partial L_o}{\partial p_i} = \sum_j L_{ji}^o t_{ji}^o - \sum_j L_{ij}^o t_{ij}^o, \tag{15}
\]

where \( t_{ji}^o = 1 \) if \( y_k = 1 \) and \( y_l = 0 \) for \( o = \text{Rank} \); \( t_{ij}^o = 1 \) if \( y_j = 1 \) and \( y_i = 0 \) for \( o = \text{Sort} \). 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 × 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 × 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 × 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 ~4.5× compared to iteration-based ones. While the fully-vectorized implementation requires huge GPU memory – at least 45 GB for 320×320 input resolution with the base model of Swin-Transformer [30], we also provide a semi-vectorized implementation which speeds up about 2.5× 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 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 × 320 resolution. We do not use any threshold to select
positive pixels (i.e., $\tau = 0$). We take the $\delta$ parameter in $H(\cdot)$ (Eq. 5) as 0.4 for NYUD-v2 and 0.1 for BSDS and Multi-cue datasets in $L_{\text{Rank}}$. In $L_{\text{Sen}}$, $\delta$ 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 best and second-best results are shown with bold and underlined texts, respectively. R: Ranking only.

**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 if $c_i \geq 0.2$, 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.

<table>
<thead>
<tr>
<th>Method</th>
<th>ODS</th>
<th>OIS</th>
<th>AP</th>
</tr>
</thead>
<tbody>
<tr>
<td>gPb-ucm [1]</td>
<td>(PAM11)</td>
<td>.632</td>
<td>.661</td>
</tr>
<tr>
<td>Silberman et al. [43] (ECCV’12)</td>
<td></td>
<td>.658</td>
<td>.661</td>
</tr>
<tr>
<td>gPb+NG [17]</td>
<td>(CVPR’13)</td>
<td>.687</td>
<td>.716</td>
</tr>
<tr>
<td>OEF [18]</td>
<td>(CVPR’15)</td>
<td>.651</td>
<td>.667</td>
</tr>
<tr>
<td>HED [49]</td>
<td>(ICCV’15)</td>
<td>.720</td>
<td>.734</td>
</tr>
<tr>
<td>RCF [29]</td>
<td>(CVPR’17)</td>
<td>.729</td>
<td>.742</td>
</tr>
<tr>
<td>AMH-Net [50] (NeurIPS’17)</td>
<td></td>
<td>.744</td>
<td>.758</td>
</tr>
<tr>
<td>LPCB [10]</td>
<td>(ECCV’18)</td>
<td>.739</td>
<td>.754</td>
</tr>
<tr>
<td>BDCN [20] (CVPR’19)</td>
<td></td>
<td>.748</td>
<td>.763</td>
</tr>
<tr>
<td>PiDiNet [44] (ICCV’21)</td>
<td></td>
<td>.733</td>
<td>.747</td>
</tr>
<tr>
<td>EDTER [39] (CVPR’22)</td>
<td></td>
<td>.724</td>
<td>.788</td>
</tr>
<tr>
<td>ACTD [12] (Neurocomp’23)</td>
<td></td>
<td>.762</td>
<td>.774</td>
</tr>
<tr>
<td>RANKED (R)</td>
<td></td>
<td>.780</td>
<td>.783</td>
</tr>
</tbody>
</table>

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.

**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 $c > 0.0$ 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 $c = 1.0$ (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., $c$ 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
4.3. Experiment 3: Ablation Analysis

We also conduct an experiment to show the efficiency of our certainty map on BSDS dataset. In this experiment, as shown in Table 8, we use pixel-wise averaging and use it as the certainty map ($c \leftarrow \sum_n y^n$). In this experiment, all settings are the same except for the labels used. Our proposed certainty map $c$ 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: $(1 - c_i) \cdot L(i)$ 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 Comparisons

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 $L_{\text{Rank}}$ and $L_{\text{Sen}}$ 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)
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 \( c > 0 \) contains all ground-truth edges in all labels, case \( c = 1 \) contains only ground-truth edges that are labeled in all labels. \( R \): Only ranking, \( R + S \): Ranking & Sorting.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Loss</th>
<th>ODS</th>
<th>OIS</th>
<th>AP</th>
</tr>
</thead>
<tbody>
<tr>
<td>NYUD-v2</td>
<td>CE(_C_E)</td>
<td>.775</td>
<td>.789</td>
<td>.802</td>
</tr>
<tr>
<td></td>
<td>CE + DICE</td>
<td>.779</td>
<td>.791</td>
<td>.807</td>
</tr>
<tr>
<td></td>
<td>RANKED ((H))</td>
<td>.786</td>
<td>.793</td>
<td>.826</td>
</tr>
<tr>
<td>BSIDS</td>
<td>CE(_C_E)</td>
<td>.820</td>
<td>.831</td>
<td>.871</td>
</tr>
<tr>
<td></td>
<td>CE + DICE</td>
<td>.821</td>
<td>.836</td>
<td>.872</td>
</tr>
<tr>
<td></td>
<td>RANKED ((H))</td>
<td>.822</td>
<td>.838</td>
<td>.886</td>
</tr>
<tr>
<td></td>
<td>RANKED ((H) + S)</td>
<td>.840</td>
<td>.854</td>
<td>.908</td>
</tr>
<tr>
<td>Multicue Edge</td>
<td>CE(_C_E)</td>
<td>.924±.006</td>
<td>.926±.007</td>
<td>.880±.001</td>
</tr>
<tr>
<td></td>
<td>CE + DICE</td>
<td>.900±.005</td>
<td>.931±.004</td>
<td>.883±.006</td>
</tr>
<tr>
<td></td>
<td>RANKED ((H))</td>
<td>.951±.002</td>
<td>.953±.001</td>
<td>.962±.004</td>
</tr>
<tr>
<td></td>
<td>RANKED ((H) + S)</td>
<td>.962±.003</td>
<td>.965±.003</td>
<td>.973±.006</td>
</tr>
<tr>
<td>Multicue Boundary</td>
<td>CE(_C_E)</td>
<td>.934±.004</td>
<td>.943±.003</td>
<td>.981±.003</td>
</tr>
<tr>
<td></td>
<td>CE + DICE</td>
<td>.947±.004</td>
<td>.954±.004</td>
<td>.984±.004</td>
</tr>
<tr>
<td></td>
<td>RANKED ((H))</td>
<td>.954±.004</td>
<td>.958±.005</td>
<td>.992±.002</td>
</tr>
<tr>
<td></td>
<td>RANKED ((H) + S)</td>
<td>.963±.002</td>
<td>.967±.002</td>
<td>.995±.001</td>
</tr>
</tbody>
</table>

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 Our\(_R\) represents using only ranking loss, \( R \): Ranking, \( R + S \): Ranking & Sorting. CE\(_C_E\): Class-balanced cross-entropy.

<table>
<thead>
<tr>
<th>Label Type</th>
<th>ODS</th>
<th>OIS</th>
<th>AP</th>
</tr>
</thead>
<tbody>
<tr>
<td>Label Averaging as Certainty ( c = \sum \gamma_i )</td>
<td>.818</td>
<td>.831</td>
<td>.880</td>
</tr>
<tr>
<td>Loss Weighting with Uncertainty ( 1 - c_i )</td>
<td>.789</td>
<td>.813</td>
<td>.849</td>
</tr>
<tr>
<td>RANKED</td>
<td>.824</td>
<td>.840</td>
<td>.895</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Implementations</th>
<th>Loss</th>
<th>Exec. Time (msec/img)</th>
</tr>
</thead>
<tbody>
<tr>
<td>-</td>
<td>CE</td>
<td>0.32</td>
</tr>
<tr>
<td>-</td>
<td>CE + DICE</td>
<td>0.39</td>
</tr>
<tr>
<td>For-iterations</td>
<td>RANKED ((H))</td>
<td>147.79</td>
</tr>
<tr>
<td></td>
<td>RANKED ((H) + S)</td>
<td>1530.22</td>
</tr>
<tr>
<td>Semi-vectorized</td>
<td>RANKED ((H))</td>
<td>2.90</td>
</tr>
<tr>
<td></td>
<td>RANKED ((H) + S)</td>
<td>16.73</td>
</tr>
<tr>
<td>Vectorized</td>
<td>RANKED ((H))</td>
<td>1.00</td>
</tr>
<tr>
<td></td>
<td>RANKED ((H) + S)</td>
<td>14.09</td>
</tr>
</tbody>
</table>

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

and negative (non-edge) classes and uncertainty arising from disagreement between different annotators. Our solution contains two novel loss functions: One for ranking positive edge pixels over negatives, and another for sorting positive pixels with respect to their edge certainty. 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.

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


[27] Florian Kraus and Klaus Dietmayer. Uncertainty estimation in one-stage object detection. In 2019 *ieee
[40] Qi Qian, Lei Chen, Hao Li, and Rong Jin. DR loss: Improving object detection by distributional ranking.

