Recursive Contour-Saliency Blending Network for Accurate Salient Object Detection

Yun Yi Ke
Computer Vision & AI Technology Lab
Open8 Singapore
yunyikeyyk@gmail.com

Takahiro Tsubono†
Computer Vision & AI Technology Lab
Open8 Singapore
tsubonot@open8.com

Abstract

Contour information plays a vital role in salient object detection. However, excessive false positives remain in predictions from existing contour-based models due to insufficient contour-saliency fusion. In this work, we designed a network for better edge quality in salient object detection. We proposed a contour-saliency blending module to exchange information between contour and saliency. We adopted recursive CNN to increase contour-saliency fusion while keeping the total trainable parameters the same. Furthermore, we designed a stage-wise feature extraction module to help the model pick up the most helpful features from previous intermediate saliency predictions. Besides, we proposed two new loss functions, namely Dual Confinelement Loss and Confidence Loss, for our model to generate better boundary predictions. Evaluation results on five common benchmark datasets reveal that our model achieves competitive state-of-the-art performance.

1. Introduction

Salient object detection (SOD) aims to detect and segment the most attention-attractive region or object in a visual scene. Unlike eye fixation prediction (FP) [34], SOD requires obtaining the entire region with clear boundaries. Due to its essential role and wide applications in image understanding [53], image captioning [8][52], search engines [54], AR/VR [30] and video summarization [25], recently, various methods have been proposed in the field.

Since 2015, Convolutional Neural Networks (CNNs) [10][15] have been adopted for SOD tasks. Though algorithms like PiCANet [23], BMPM [55], and PAGRNN [47] achieved significantly better results, the predicted object usually has poor boundaries. To obtain more precise boundaries, proposed by Qin et al., BASNet [31] adopted a boundary refinement U-Net [52] at the end of the saliency detection network and trained their model using various losses. In [5], Chen et al. proposed Contour Loss (CTLoss), which was a weighted Binary Cross-Entropy (BCE) loss, to improve the boundary predictions. Alternatively, models like EGNet [59], PoolNet [22], and ITSD [52] fused contour predictions with saliency by explicitly supervising a contour branch. With contour cues, models yielded better boundary predictions.

However, above mentioned models still hold several problems that can be further improved. First, for better performance, many studies have introduced a huge number of trainable parameters. EGNet contains 108 million parameters, BASNet and its extended work, U2Net [30], have more than 87 and 44 million parameters, respectively (Fig. 2). The huge number of parameters not only leads to increased consumption of computational resource, but also makes the model difficult to train. Second, though

†Corresponding author.
Figure 2: Mean F-score, total parameter and MAE comparison between RCSB with 13 state-of-the-art models, including EGNet [49], PiCANet [23], C2SNet [18], PoolNet [22], PAGE [37], Amulet [46], ITSD [52], BASNet [31], U2Net [30], NLDF [24], AFNet [9], F3Net [39] and CPD [40] on DUTS-TE [35] dataset. Bubble size represents model size.

object boundary is greatly improved for contour-based networks [22][49][52], predictions still have excessive false positives, as shown in Fig. 1c and 1d. Finally, to our best knowledge, for all deep learning-based models in the current field, intermediate saliency or contour predictions are generated and supervised via side branches, which introduce redundant parameters and inefficiency.

To address the abovementioned issues, we proposed a recursive contour-saliency blending network, namely, RCSBNet, for high accuracy salient object detection. We adopted a recursive CNN to reduce total trainable parameters while we can make our model deep. Unlike previous studies [22][49][52], where the contour and saliency are explicitly trained via two branches, we introduced a Contour-Saliency Blending (CSB) module in our network so that contour and saliency are intertwined and fused every step in the recursion. Meanwhile, to further improve the efficiency, we proposed a Stage-wise Feature Extraction (SFE) module to directly supervise intermediate saliency and contour predictions in the primary network without using any side branch. Lastly, we divided the training task into accuracy and confidence, and proposed Dual Confinement Loss (DCLoss) and Confidence Loss (CLoss) respectively for better model performance. To sum up, our contributions are as follows:

1. We proposed an efficient and accurate network, RCSBNet. By using a recursive CNN and the proposed Stage-wise Feature Extraction (SFE) module, contour and saliency are fused more efficiently and effectively.
2. We developed two loss functions, DCLoss and CLoss, to further help the boundary prediction.
3. Our model has only 27.9 million parameters, which is significantly smaller and efficient than most of the networks in the field (Fig. 2).
4. We conducted comprehensive evaluations on 5 widely used benchmark datasets and compared with 13 state-of-the-art methods. Our method achieves competitive state-of-the-art results among all datasets.

2. Related Work

Early approaches based on hand-crafted priors [6][12][44] have limited effectiveness and generalization ability. The very first deep salient object detection (SOD) methods [17][50] used multi-layer perceptron to predict saliency score for each image. These methods suffered from low efficiency and damage of feature structures due to flattening. Later, some studies introduced a fully convolutional network (FCN) and achieved promising results.

Recurrent Networks. In [20], a recurrent convolutional neural network (RCNN) was proposed for object detection. The main idea was to unfold the same convolution layer several times while weights are shared. It had the advantage that model depth can now be deeper by unfolding, while the total number of trainable parameters remains the same. It also revealed that, by increasing the number of recursions, better results would be obtained.

In 2016, the recurrent CNN was introduced to salient object detection task, and proposed by Wang et al. RFCN [36] recursively refines the saliency prediction from previous time step. Later, proposed by Kuen et al. [16], a recurrent network was designed to refine selected image sub-regions iteratively. In [47], Zhang et al. designed a multi-path recurrent model for saliency detection by transferring global information from deep layers to shallower layers. Hu et al. [33] proposed their salient object detection model by concatenating multi-layer deep features recurrently. It was proved that saliency predictions will be refined by using recurrent mechanism.

Utilizing Contour Information. In recent years some studies explored and verified the effectiveness of involving contour information to improve the accuracy of saliency prediction. In [5], Chen et al. considered boundary pixels as hard samples and proposed a contour loss, which was a weighted BCE loss, to train their network. Qin et al. [31] combined Structural Similarity Index (SSIM), Intersection over Union (IOU), and BCE as their contour-aware loss function to achieve better boundary quality. In another seminal work, Salient Edge Detector (SED) [37] was introduced to simultaneously generate saliency and contour predictions by using a residual structure. Furthermore, PoolNet [22] applied multi-task training and fused the contour information with saliency predictions. Later, ITSD [52] proposed a two-stream network to convert saliency and contour interactively and yielded good boundary predictions. These studies further corroborated the importance of employing contour information.

<table>
<thead>
<tr>
<th>Model</th>
<th>MAE</th>
<th>Parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>RCSBNet</td>
<td>0.03</td>
<td>27.9M</td>
</tr>
<tr>
<td>EGNet [49]</td>
<td>0.04</td>
<td>106.5M</td>
</tr>
<tr>
<td>PiCANet [23]</td>
<td>0.05</td>
<td>47.2M</td>
</tr>
<tr>
<td>C2SNet [18]</td>
<td>0.05</td>
<td>34.5M</td>
</tr>
<tr>
<td>PoolNet [22]</td>
<td>0.06</td>
<td>108M</td>
</tr>
<tr>
<td>PAGE [37]</td>
<td>0.06</td>
<td>25.6M</td>
</tr>
<tr>
<td>Amulet [46]</td>
<td>0.07</td>
<td>57.6M</td>
</tr>
<tr>
<td>ITSD [52]</td>
<td>0.07</td>
<td>47.9M</td>
</tr>
<tr>
<td>BASNet [31]</td>
<td>0.08</td>
<td>27.9M</td>
</tr>
<tr>
<td>U2Net [30]</td>
<td>0.08</td>
<td>35.4M</td>
</tr>
<tr>
<td>NLDF [24]</td>
<td>0.09</td>
<td>26.5M</td>
</tr>
<tr>
<td>AFNet [9]</td>
<td>0.09</td>
<td>52.3M</td>
</tr>
<tr>
<td>F3Net [39]</td>
<td>0.10</td>
<td>57.6M</td>
</tr>
<tr>
<td>CPD [40]</td>
<td>0.10</td>
<td>52.3M</td>
</tr>
</tbody>
</table>
information to improve saliency predictions.

3. Proposed Method

3.1. Overall Architecture

In previous contour-related networks \cite{22,49,52}, either the contour was supervised in a separate branch to guide the saliency prediction, or it was fused with saliency stage by stage to achieve better boundary predictions. Both approaches gave promising results, but there are two major disadvantages: 1) \textit{late fusion}: contours are fused with saliency at the end of each stage. 2) \textit{limited fusion}: the number of fusion is limited by the number of U-Net stages. For \textit{late fusion}, we designed a Contour-Saliency Blending Unit (CSBU) so that contour and saliency information can be exchanged at a much earlier stage, while for \textit{limited fusion}, a recursive mechanism was adopted to circumvent this constraint.

As shown in Fig. 3, the proposed RCSBNet is essentially a U-Net, where we employ pre-trained ResNet-50 as our encoder and a customized decoder adopting recursive CNNs. Contour and saliency are blended in the recursion block by the Contour-Saliency Blending Unit (CSBU). Then saliency and contour features are split and fed into the Stage-wise Feature Extraction (SFE) module for supervised learning. At the last stage of the decoder, we concatenate the prediction of contour and saliency with the input image, followed by an additional recursive block, to generate the final predictions.

3.2. ResNet-50 as the Encoder

We employ the pre-trained ResNet-50 as our encoder. Since it contains a large number of feature maps in high-level blocks, same as \cite{52}, we apply channel pooling (CP) to reduce the number of channels to 64, and the operation can be expressed as:

$$CP = \text{collect}_{j \in [0, m-1]}(\text{max}_{k \in [0, n-1]} X^j \times \frac{n}{m} + k)$$  \hspace{1cm} (1)

where $X$ represents the raw feature map with total $n$ channels and $j, k$ are integers. We divide $X$ into $m$ groups then apply max-pooling and concatenate all $m$ maps. In our experiments, $m$ is set to 64. A convolution layer is attached after the channel pooling layer to prepare the features for further processing by the decoder, as shown in Fig. 3.

3.3. Contour-Saliency Blending Unit (CSBU)

Contour information is usually supervised and fused with saliency at the end of each U-Net stage. For us, we want to fuse the contour and saliency at earlier stages and as many times as possible. Inspired by Shuffle-Net \cite{48}, where convolution was divided into groups and information is exchanged by shuffling the weight channels, we designed our Contour-Saliency Blending Unit (CSBU). As illustrated in Fig. 4a, the CSBU has a contour branch and saliency branch. Features generated by each branch are concatenated and blended. Mathematically, let $F^{\text{sal}}_{k \times k}$ and $F^{\text{ctr}}_{k \times k}$ denote the convolution for saliency and contour with kernel size $k$ respectively, while $(C_{n-1}, S_{n-1})$ and $(C_n, S_n)$ represent contour and saliency features before and after CSBU. For simplicity, we omit ReLU \cite{2} and BatchNorm \cite{11} in our formula, and then the proposed CSBU can be modeled as:

$$CTR = F^{\text{ctr}}_{3 \times 3}(C_{n-1})$$  \hspace{1cm} (2)

$$SAL = F^{\text{sal}}_{3 \times 3}(S_{n-1})$$  \hspace{1cm} (3)

$$C_n = F^{\text{ctr}}_{3 \times 3}(F^{\text{ctr}}_{1 \times 1}([CTR, SAL]))$$  \hspace{1cm} (4)

$$S_n = F^{\text{sal}}_{3 \times 3}(F^{\text{sal}}_{1 \times 1}([CTR, SAL]))$$  \hspace{1cm} (5)

and thus:

$$S_n, C_n = \text{CSBU}(S_{n-1}, C_{n-1})$$  \hspace{1cm} (6)

By applying CSBU, contour and saliency information can be utilized by the network at a much earlier stage.

3.4. Recursive Block

We now introduce more details of the recursive block, as illustrated in Fig. 4b. Firstly let us consider a single recursive block then we extend to more general cases. We denote $R$ as the total number of recursions in a single block, and $f, r \in [1, R]$ denote the $r^{th}$ recursion of the CSBU. Based on Eq. 6, let $X_{n-1}$ denote the input tuple of $(S_{n-1}, C_{n-1})$, and $X_R$ represent the output of the $R^{th}$ recursion, then we have:

$$X_R = f^R(...(f^2(f^1(X_{n-1})+X_{n-1})+X_{n-1})+...+X_{n-1})$$  \hspace{1cm} (7)

note that weights for CSBU are shared in recursion, though they have different superscripts in the formula. Then we apply a single convolution layer, denoted as $F_{3 \times 3}$, at the end of recursion with skip connection. Thus the output of our recursive block, $X_n$, is:

$$X_n = F_{3 \times 3}(X_R + X_{n-1}) + X_{n-1} = h_b(X_{n-1})$$  \hspace{1cm} (8)

where $h_b$ represents the recursive block function.

Now let $G$ represent the total number of recursive blocks, as shown in Fig. 4c in our network we simply stack all the recursive blocks together. Thus given the input $R_{n-1}$, a tuple of $(S_{n-1}, C_{n-1})$, the output of $g$-th block, $R_n$, is:

$$R_n = h_b^g(h_b^{g-1}(...h_b^1(h_b(R_{n-1}))))$$  \hspace{1cm} (9)

where $h_b^g$ represents the $g$-th recursive block function.

By applying recursive blocks and stacking them together, contour and saliency can now be fused $G \times R$ times at each stage of the U-Net, which will improve the network performance significantly. We will show more results in Sec. 4.
Figure 3: Network architecture for RCSBNet. Pre-trained ResNet-50 is used as the backbone, channel number is reduced to 64 via Channel Pooling (CP) layer. Recursive Contour-Saliency Blocks (RCSB) with $G$ blocks and $R$ recursions are then attached and followed by Stage-wise Feature Extraction (SFE) module to generate contour and saliency predictions. At the end of the network, a refinement module is adopted to further refine the predictions.

3.5. Stage-wise Feature Extraction (SFE) Module

It is prevalent that saliency networks are densely supervised, where intermediate stage features are generated and supervised against ground truths to guide the model for better convergence. Common practices create a side branch with a few convolution layers to generate predictions from current U-Net stage features, following which the stage features are passed on to the next U-Net stage in the primary network (Fig. 4e). Different from others, we regard stage predictions as the best result the network can obtain so far...
and apply a new round of feature extraction based on current stage predictions. Therefore, stage predictions are now in the network’s main branch, acting as a single channel layer, and supervised against ground truths, as illustrated in Fig. 4d. By doing so, the next block is expected to extract valuable features from current best results and discard all useless features, which might otherwise be carried along by recursion and residual connections.

To generate stage predictions, unlike previous studies 3,22,52, we do not use max-pooling because it will bring false positives to the next stage, nor the element-wise multiplication between contour and saliency like SCRN 41 due to the introduction of false negatives. Instead, we employ a $1 \times 1$ convolution and a scaling factor to help sigmoid function classify saliency and background. To formulate the SFE module mathematically, let $C^i$ and $S^j$ represent incoming $i$ channels of feature maps, and $K$ represent the scaling factor learned by the network, then stage prediction $P^j$ with $j$ channels can be expressed as:

$$P^j_{sal} = F^j_{1 \times 1}(S^{64}_{n-1}) * K_{sal}$$

$$P^j_{ctr} = F^j_{1 \times 1}(C^{64}_{n-1}) * K_{ctr}$$

where $*$ represents element-wise multiplication. For extracted feature maps $M^n_j$ with $j$ channels:

$$M^{64}_n = F_{3 \times 3}(U^{p \times 2}[\sigma(P^{sal}_j), \sigma(P^{ctr}_j)])$$

where $[\ ]$ stands for concatenating, $U^{p \times 2}$ is up-sampling, and $\sigma$ stands for sigmoid function. SFE is a simple module but it is very effective, we will demonstrate more in ablation studies.

3.6. Loss Functions

Unsure prediction is very common in salient object detection in the form of shady areas (Fig. 5b). Many studies try to improve the performance by focusing on hard pixels near the boundary 5,39. Boundary pixels are indeed hard samples, but not all the hard samples are near the boundary. We notice that the network can correctly predict the saliency for some images, but with low pixel values; while for some other networks, the network is confident of generating false negatives (FN) or false positives (FP), as illustrated in Fig. 5c and 5f. It points out two kinds of difficulties encountered by the network: 1) unconfident but accurate predictions and 2) confident but inaccurate predictions. Hence in addition to accuracy, we factor in the confidence of predictions, and we introduce Confidence Loss (CLoss), $L_{con}$, as defined as:

$$L_{con}(\beta, \lambda) = -\frac{1}{n} \sum_{i=1}^{n} [W_c \cdot BCE(x_{i,j}, \hat{y}_{i,j}) + \lambda W_c]$$

where $\lambda$ is set to 0.3 by parameter search. By applying this loss, it will encourage the model to make more confident predictions into either foreground (close to 1) or background (close to 0).

**Dual Confinement Loss.** Most of the contour-based models use a separate contour branch to guide saliency predictions. In 5, and 52, a weight map generated from contour ground truth is applied to BCE loss to improve saliency boundary quality while the contour branch is supervised by BCE loss without any weights. It is reasonable because contour information is from a separate branch and is only used to guide saliency. Since in our network, saliency and contour streams are going to exchange information; thus we not only use contour to guide saliency predictions but also want to use saliency

![Figure 5: (a) ground truth saliency. (b) predicted saliency. (c) focal loss ($\alpha, \gamma = 2, 2$) weight map calculated using (a) and (b). (d) confidence loss ($\beta, \lambda = 2, 1$) weight map calculated using (a) and (b). (e) unconfident but correct area (red) in prediction. (f) confident false positives (orange) and false negatives (green) in prediction. Compare (d) with (c), focal loss is less sensitive to unsure predictions, while confidence loss generates high weights especially on object boundaries, which eventually help the model generate sharper edges.](image-url)
to guide contour predictions. Based on this, we designed our Dual Confinement Loss (DCLoss), and losses for each stream, $L_{sal}$ and $L_{ctr}$, are defined as:

$$L_{sal} = -\frac{1}{n} \sum_{i=1}^{n} [W_{sal} \times BCE(x_i^{sal}, y_i^{sal})]$$  \hspace{1cm} (15)$$

$$L_{ctr} = -\frac{1}{n} \sum_{i=1}^{n} [W_{ctr} \times BCE(x_i^{ctr}, y_i^{ctr})]$$  \hspace{1cm} (16)$$

where weight matrix $W_{sal}$ and $W_{ctr}$ are calculated by:

$$W_{sal} = \max(x_i^{ctr}, y_i^{ctr}) \times \theta + 1$$  \hspace{1cm} (17)$$

$$W_{ctr} = \max(x_i^{sal}, y_i^{sal}) \times \theta + 1$$  \hspace{1cm} (18)$$

where $x, y$ stand for prediction and ground truth. Max-pooling is used because higher weights need to be applied when $x$ is not close to $y$. Empirically, we set $\theta = 4$ in our experiments. Then the DCLoss, $L_{DC}$, is defined as:

$$L_{DC} = L_{sal} + L_{ctr}$$  \hspace{1cm} (19)$$

**Loss for Training.** Compared with low-level predictions, it should be relatively easy to emphasize accuracy on high-level predictions due to its small feature dimension. Meanwhile, a failure in high-level predictions will impact its following decoders and eventually cause false positives or negatives. Thus, we train stages 1 to 5 of our network against accuracy-related losses, i.e. DCLoss and weighted IOU loss mentioned in [39], and train the refinement module against $L_{DC}$ only. Our loss functions for saliency and contour are defined as:

$$L_{saliency} = L_{DC}^{1-5} + L_{wIOU}^{5} + L_{ref}^{ref}(\beta, \lambda = 2, 0.3)$$  \hspace{1cm} (20)$$

$$L_{contour} = L_{DC}^{1-5} + L_{ref}^{ref}(\beta, \lambda = 2, 0.3)$$  \hspace{1cm} (21)$$

where superscripts $1-5$ and $ref$ represent the five decoder stages and refinement module in Fig. 2.

4. Experiments

4.1. Datasets


4.2. Implementation Details

DUTS-TR [35] is used for training with input images resized to 256x256, then random horizontal flipping and 90° rotation are applied as the augmentation. We use pre-trained ResNet-50 [10] as the encoder. The number of recursive blocks $G$ is set to (1,2,2,3,3,5) with recursion $R = 3$ for the network (Fig. 3) by parameter search. Besides, we apply Leaky ReLU [13] and adopt the Adam optimizer [14] with default hyperparameters to train our network. Learning rates for encoder and decoder are set to $10^{-5}$ and $10^{-4}$ respectively, and they are halved every 20 epochs with a total of 100 epochs using a batch size of 4. During testing, images are resized to 256x256, and the predictions (256x256) are resized back to their original size by using bilinear interpolation. We use PyTorch [29] and a single RTX 3090 GPU for our model and experiments. Code is available at: https://github.com/BarCodeReader/RCSB-PyTorch

4.3. Evaluation Metrics

**Precision-Recall (PR) curve, $F_\beta$-measure [1], Mean Absolute Error (MAE), $F_\beta^2$-measure [26], and $E_\xi$-measure [7] are adopted in our experiments.**

**PR-Curve.** By applying different thresholds from 0 to 255, PR curve is obtained by comparing the ground truth masks against the binarized saliency predictions.

**F-measure.** The $F_\beta$-measure is calculated by precision and recall value of saliency maps: $F_\beta = \frac{(1+\beta^2) \times \text{Precision} \times \text{Recall}}{\beta^2 \times \text{Precision} + \text{Recall}}$, where $\beta^2$ is set to 0.3 [11]. We report the average score over all thresholds from 0 to 255 and denote as $F_\beta$ [24,45].

**MAE.** MAE is the mean value of the sum of pixel-wise absolute differences between predictions $x$ and ground truths $y$: $MAE = \frac{1}{n} \sum_{i=1}^{n} |x_i - y_i|$. **Weighted F-measure.** $F_\beta^{\omega}$ uses weighted precision and weighted recall to measure both exactness and completeness of the prediction against ground truth. It is designed to improve the existing $F_\beta$-measure.

**E-measure.** By using local pixel values and the image-wise mean, $E_\xi$ calculates the similarity between the prediction and the ground truth.

4.4. Comparisons with State-of-the-art Results

We compare our results with 13 state-of-the-art salient object detection networks, including EGNet [49], PoolNet [22], ITSD [52], AFNet [9], PAGE [37], CPD [40], BASNet [31], CAGNet [27], GateNet [51], U2Net [30], GCPA [4], MInet [28], and F3Net [39]. Saliency maps used are provided by authors.

**Quantitative Evaluation.** To compare our work with the state-of-the-art networks, detailed experimental results in terms of four metrics are listed in Table 1. Among all the models, RCSBNet achieves outstanding results across all four metrics on most datasets. Besides, PR and F-measure curves are demonstrated in Fig. 7. Our F-measure curves are flatter than all other models, which reveals that our results are closer to binary predictions and invariant to threshold changes.

**Qualitative Evaluation.** Visual comparisons are listed in Fig. 6. Compare with other contour-based network
Table 1: Quantitative comparisons between RCSBNet and other 13 methods on five benchmark datasets in terms of the average F-measure $\bar{F}_M$, MAE $M$, $E_k$, and $F^\uparrow_M$. ↑ $\downarrow$ means the larger/smaller the value, the better the results. Red, Green, and Blue indicate the best, second best and third best performance.

<table>
<thead>
<tr>
<th>Method</th>
<th>DUTS-TE</th>
<th>HKU-IS</th>
<th>PASCAL-S</th>
<th>ECSSD</th>
<th>DUT-OMRON</th>
</tr>
</thead>
<tbody>
<tr>
<td>EGNet[9]</td>
<td>0.815</td>
<td>0.931</td>
<td>0.900</td>
<td>0.887</td>
<td>0.920</td>
</tr>
<tr>
<td>PoolNet[10]</td>
<td>0.819</td>
<td>0.933</td>
<td>0.902</td>
<td>0.889</td>
<td>0.921</td>
</tr>
<tr>
<td>CAGNet[11]</td>
<td>0.804</td>
<td>0.931</td>
<td>0.902</td>
<td>0.894</td>
<td>0.915</td>
</tr>
<tr>
<td>BASNet[12]</td>
<td>0.805</td>
<td>0.943</td>
<td>0.907</td>
<td>0.909</td>
<td>0.920</td>
</tr>
<tr>
<td>Ours</td>
<td>0.855</td>
<td>0.943</td>
<td>0.907</td>
<td>0.909</td>
<td>0.920</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Method</th>
<th>DUTS-TE</th>
<th>HKU-IS</th>
<th>PASCAL-S</th>
<th>ECSSD</th>
<th>DUT-OMRON</th>
</tr>
</thead>
<tbody>
<tr>
<td>PAGE[13]</td>
<td>0.793</td>
<td>0.946</td>
<td>0.879</td>
<td>0.785</td>
<td>0.894</td>
</tr>
<tr>
<td>BASNet[14]</td>
<td>0.805</td>
<td>0.947</td>
<td>0.884</td>
<td>0.803</td>
<td>0.908</td>
</tr>
<tr>
<td>GCPA[15]</td>
<td>0.806</td>
<td>0.940</td>
<td>0.897</td>
<td>0.817</td>
<td>0.909</td>
</tr>
<tr>
<td>MI[16]</td>
<td>0.817</td>
<td>0.938</td>
<td>0.891</td>
<td>0.821</td>
<td>0.915</td>
</tr>
<tr>
<td>CASNet[17]</td>
<td>0.817</td>
<td>0.938</td>
<td>0.891</td>
<td>0.821</td>
<td>0.915</td>
</tr>
<tr>
<td>Ours</td>
<td>0.855</td>
<td>0.943</td>
<td>0.907</td>
<td>0.909</td>
<td>0.920</td>
</tr>
</tbody>
</table>

Figure 6: Visual comparisons between our method and 10 state-of-the-art networks. * stands for models utilizing contour information. More comparisons are provided in the supplementary material.

Figure 7: First row: Precision-Recall Curves comparison on five saliency benchmark datasets. Second row: F-measure Curves comparison on five saliency benchmark datasets.
results, our method yields better boundary predictions. As shown in the graph, our model can produce accurate and complete saliency maps with better edges.

4.5. Ablation Studies

Effectiveness of the early fusion (EF). To prove that early fusion of contour and saliency information will boost model performance, an experiment was conducted by removing the fusion branch illustrated in Fig. 4a and make it into two separate streams. We list both quantitative and qualitative measures between the two approaches on DUTS-TE and ECSSD datasets, as shown in Table 2 and Fig. 8.

(a) GT  (b) w/ EF  (c) w/o EF  (d) w/ EF  (e) w/o EF

Figure 8: Qualitative comparisons for early fusion. (a) Ground-truth. (b) & (d) Saliency and contour prediction with early fusion. (c) & (e) Saliency and contour prediction without early fusion. As can be seen with early fusion, results are more complete for both contour and saliency predictions.

Table 2: Quantitative comparisons for the effectiveness of early fusion.

<table>
<thead>
<tr>
<th></th>
<th>DUTS-TE</th>
<th>ECSSD</th>
</tr>
</thead>
<tbody>
<tr>
<td>w/ early fusion</td>
<td>.855 .034 .903 .840</td>
<td>.927 .033 .923 .916</td>
</tr>
<tr>
<td>w/o early fusion</td>
<td>.849 .037 .870 .837</td>
<td>.922 .035 .921 .909</td>
</tr>
</tbody>
</table>

Effectiveness of refinement module and supervised on confidence. To study the importance of the refinement module and prove the effectiveness of supervision on confidence, we conducted 4 experiments on DUTS-TE and ECSSD datasets covering all the cases for our comparison, as listed in Table 3. For conciseness, we denote reference module and supervision on confidence as Ref. and Conf., respectively.

It can be observed that each approach will boost the performance, and when they are combined, we obtained the best results.

Effectiveness of SFE module and different loss functions. To study the importance of each loss function and SFE module, we conduct a series of controlled experiments on the DUTS-TE dataset. We train the model by using BCE loss only, then include weighted IOU Loss \( L^w \), DCLoss, and CLoss step by step. Detailed results are listed in Table 4.

To prove the effectiveness of SFE module, qualitative comparisons are illustrated in Fig. 9. As can be seen, with SFE module, model can effectively suppress excessive wrong predictions. Though SFE is a simple feature extraction module, it improves overall performance.

5. Conclusions

In this paper, we have introduced an efficient and accurate model using a recursive CNN together with a Contour-Saliency Blending (CSB) module. To further improve the model’s efficiency, a Stage-wise Feature Extraction (SFE) module is adopted. It is a simple module but can suppress wrong predictions effectively. Furthermore, we divided the training objectives into accuracy and confidence and proposed two loss functions to guide model convergence. The predicted salient objects achieved comparable state-of-the-art results on five benchmark datasets.
References


