Region Refinement Network for Salient Object Detection

Zhuotao Tian[1] **Hengshuang Zhao**[1] **Michelle Shu**[2] **Jiaze Wang**[1]
Ruiyu Li[3] **Xiaoyong Shen**[3] **Jiaya Jia**[1,3]

[1]The Chinese University of Hong Kong   [2]John Hopkins University
[3]Youtu Lab, Tencent

{zttian,hszhao,jzwang,leojia}@cse.cuhk.edu.hk   shu.michelle97@gmail.com
{royryli,dylanshen}@tencent.com

Abstract

Albeit intensively studied, false prediction and unclear boundaries are still major issues of salient object detection. In this paper, we propose a Region Refinement Network (RRN), which recurrently filters redundant information and explicitly models boundary information for saliency detection. Different from existing refinement methods, we propose a Region Refinement Module (RRM) that optimizes salient region prediction by incorporating supervised attention masks in the intermediate refinement stages. The module only brings a minor increase in model size and yet significantly reduces false predictions from the background. To further refine boundary areas, we propose a Boundary Refinement Loss (BRL) that adds extra supervision for better distinguishing foreground from background. BRL is parameter free and easy to train. We further observe that BRL helps retain the integrity in prediction by refining the boundary. Extensive experiments on saliency detection datasets show that our refinement module and loss bring significant improvement to the baseline and can be easily applied to different frameworks. We also demonstrate that our proposed model generalizes well to portrait segmentation and shadow detection tasks.

1   Introduction

Salient objects are one primary information source in the human perception of natural scenes. Salient Object Detection (SOD), as an essential part of computer vision, aims at identifying the salient objects in wild scenes. Previous SOD methods, as fundamental tools, have benefited a wide range of applications. Topics that are closely or remotely related to visual saliency include semantic segmentation [81, 6, 7, 28], instance segmentation [17, 43, 5], object detection[50, 57, 27, 83], to name a few.

Early SOD methods [86, 58, 1] utilize low-level, hand-crafted features that are vulnerable to complex scenarios. Recently, deep learning based methods [62, 65, 4, 25] greatly improved performance on benchmark datasets. However, this task is still challenging in natural scenes due to the varying attributes such as tiny and thin structures, various background and multiple objects.

To alleviate these difficulties, the work of [13, 63] used recurrent structures to refine results. These methods do not control the information flow during the recurrent refinement process and allow redundant information pass, possibly resulting in performance reduction, which will be discussed more in Section 4.3.2. To selectively pass useful information for better results, we design a cascaded region refinement module with a gating mechanism that can be viewed as an attention map. It is guided by the ground truth to control information flow and helps the network focus on refining only salient regions. Our extensive experiments in Section 4.4.1 show that this design is general for applying to other frameworks and improves their respective performance.

Common boundary area errors on salient object predictions are also addressed. As shown later in Figure 2(d)&(f), absolute errors between ground truth and predicted saliency maps mainly stem from the boundary area. To obtain a better prediction, we propose Boundary Refinement Loss (BRL) to help refine boundary areas. Compared with previous boundary strategies [64, 71], BRL is more

efficient because it does not require additional parameters and computation. Although the method of [47] also considers boundary enhancement, our ablation study in Section 4.3.1 show that our strategy achieves better performance because both foreground and background near the boundary are supervised and optimized.

Our contribution in this paper is threefold.

- We propose a new effective region refinement module.
- We propose an efficient enhancement strategy to refine prediction on the boundary.
- Our model achieves new state-of-the-art results on five representative benchmark datasets. Also, our designs can be easily applied to other frameworks.

2   Related Work

In recent years, salient object detection has attracted much attention due its wide range of applications [62, 65, 15, 4, 25, 80, 64, 76, 61, 23, 34, 68, 40, 33, 31, 14, 9, 19, 75, 22, 79, 82, 49, 16, 10]. Methods designed for salient object detection can be divided into two categories of deep learning based methods and those without learning. Non-deep-learning methods are primarily based on low-level features like regional contrast [73, 12, 29, 11, 51] and priors [44, 74, 26, 66, 37]. More details can be found in [2]. In this section, we mainly discuss deep learning based solutions.

Early deep learning methods use features generated by deep neural networks (DNNs) to calculate saliency of image units, like super-pixels and proposals. For example, in [35], with the features extracted from DNN, the score of each super-pixel of an image can be easily calculated. In [60], two networks are utilized (one for local superpixel estimation and the other for global proposal search). The final saliency map is generated by a weighted sum of salient object regions.

Later, end-to-end frameworks [78, 36, 20, 21, 13, 32, 77, 42, 63, 41] became the mainstream and many effective structures were proposed. In [78], multi-level features are aggregated from a backbone network for salient object detection. [36] proposes a deep contrast network with a pixel-level fully convolutional stream and a segment-wise spatial pooling module to deal with blurry regions. [20] introduces short connections to the skip-layer structures within the holistically-nested edge detector (HED) [70] to boost performance. [77] designs a bi-directional structure that enables information flow between lower and higher layers, and [42] proposes an attention mechanism that helps capture contextual information for each pixel in both local and global forms. [41] proposes to hierarchically and progressively refine the saliency map with local context information. Our work is different in that we use recurrent structures and exploit boundary enhancement.

Recurrent structures have been proved useful [32, 13, 63]. In R3Net [13], Residue Refinement Blocks (RRBs) are recurrently applied to the extracted deep feature to refine the output. [32] combines spatial transformer and recurrent network units to iteratively perform saliency refinement on sub-regions of images. [63] leverages a stage-wise refinement network to refine saliency prediction in a coarse-to-fine manner. Our method is inherently different from the above methods. With a novel supervised mask, our proposed cascaded refinement module involves the attention-like mechanism. It significantly improves performance by keeping meaningful information.

On the other hand, boundary (or contour) is essential to consider [71, 64, 47, 54]. C2S-Net [71] has two branches: one predicts contour and the other generates saliency maps with a contour-to-saliency transferring method to utilize contour learning in saliency map prediction. [64] proposes a local Boundary Refinement Network to adaptively learn the local contextual information for each pixel, which helps capture relationships in neighbors. [54] uses a separate branch to generate boundary prediction and fuses it with a region saliency map to refine results. Both methods involve additional parameters to form a new branch for boundary prediction. Differently, [47] proposes an additional IoU Boundary Loss generated by gradients of ground truth and predicted salient region map. Likewise, our boundary refinement strategy only involves an additional loss, but it directly optimizes the boundary region rather than gradients, resulting in even better performance.

3   Our Method

In this section, we motivate our solution and detail the new Region Refinement Module (RRM) and Boundary Refinement Loss (BRL), vastly useful in saliency detection. Then we propose the Region Refinement Network (RRN).

3.1   Region Refinement Module

Recently proposed refinement modules refine input features without any selection process. We note input features always bring redundant information that may cause error or ambiguity. For example, for features extracted from multiple layers [20, 46, 13], pixels belonging to tiny structures, such as animal hair and tail, need additional spatial information, while pixels for large objects like elephant body and
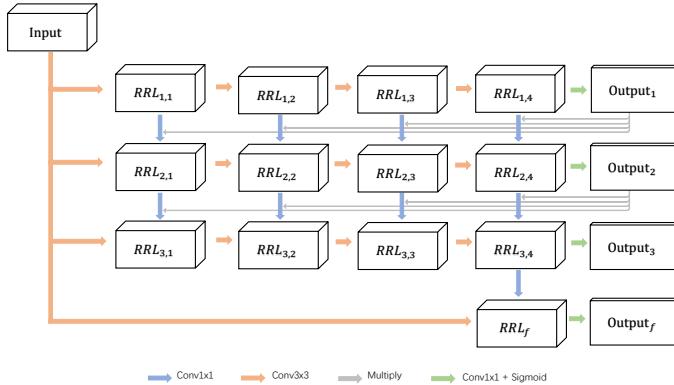
Figure 1: Example of Region Refinement Module ($N = 3, M = 4$).

airplane need more semantic clues. Without carefully handling this issue, the final prediction may contain hard samples for false positives with high probability and false negatives with low probability. To alleviate this difficulty, we propose Region Refinement Module (RRM) in the cascaded structure that helps accumulate useful information to the final feature. In addition, different from previous work in Section 2, we utilize the output salient map in each intermediate refinement stage as an attention map to guide the information flow. Our experiments with/without using the mask in Section 4.3.2 show that this gating mechanism can refine features in a particular way for better overall performance.

3.1.1 Module Structure

As shown in Figure 1, RRM is composed of several Region Refinement Layers (RRL). N is the number of RRL, and M is the number of 3×3 convolutions (equals to the number of 1×1 convolutions) in each RRL. Each RRL receives coarse features from the input and previous layers and passes the refined features to the next layer. At the end of each RRL, the output is processed by a Sigmoid function. It is then used as an attention mask to gate the refined feature to the next layer, making the model focus more on refining salient pixels.

$$RRL_{i,j} = \begin{cases} \mathcal{G}(X) & i = 1, j = 1 \\ \mathcal{G}(\mathcal{F}(RRL_{i-1,j} \odot Y_{i-1}) + X) & i > 1, j = 1 \\ \mathcal{G}(RRL_{i,j-1}) & i = 1, j > 1 \\ \mathcal{G}(\mathcal{F}(RRL_{i-1,j} \odot Y_{i-1}) + RRL_{i,j-1}) & \text{Otherwise} \end{cases} \quad (1)$$

$$Y_i = \mathcal{F}_\rho(RRL_{i,M}) \quad i = 1, 2, ..., N \quad (2)$$

The formulation of RRM is shown in Eqs. (1) and (2) where X is the input feature, Y_i is the output from RRL_i, N is the number of layers, M is the number of 3×3 convolutions, $\odot$ denotes pixel-wise multiplication, $\mathcal{F}$ represents 1×1 convolution with ReLU function, $\mathcal{G}$ represents 3×3 convolution with ReLU function and $\mathcal{F}_\rho$ represents 1×1 convolution with Sigmoid function.

As Eq. (1) shows, output Y_i from current layer RRL_i is used as a gate to control the information passed to next layer RRL_{i+1} by a pixel-wise multiplication, which helps filter out redundant feature with low probability and retain the information of interest with high probability in the previous stage.

$$RRL_f = \mathcal{F}(U(RRL_{N,M})) + \mathcal{G}(U(X)) \quad (3)$$

$$Y_f = \mathcal{F}_\rho(RRL_f) \quad (4)$$

At last layer RRL_N, as Eqs. (3) and (4) show (U is the upsampling function), RRM generates final output Y_f by upsampling $RRL_{N,M}$ and input feature twice larger then applying 1x1 convolution to both.

In RRM we calculate cross entropy loss for all outputs. Since RRM only takes Y_f as the final output, the final loss combines Y_f and Y_i as

$$L = L_f + \frac{\lambda}{N} \sum_{i=1}^{N} L_i = L_f + \sigma \sum_{i=1}^{N} L_i \quad (5)$$

where L_f is the cross entropy loss of Y_f, L_i is the cross entropy loss of Y_i. σ is used for balancing the loss of previous outputs. σ equals $\frac{\lambda}{N}$ where N is the number of RRL and λ is a balancing factor for Y_i. The effect of σ is discussed in Section 4.3.2.
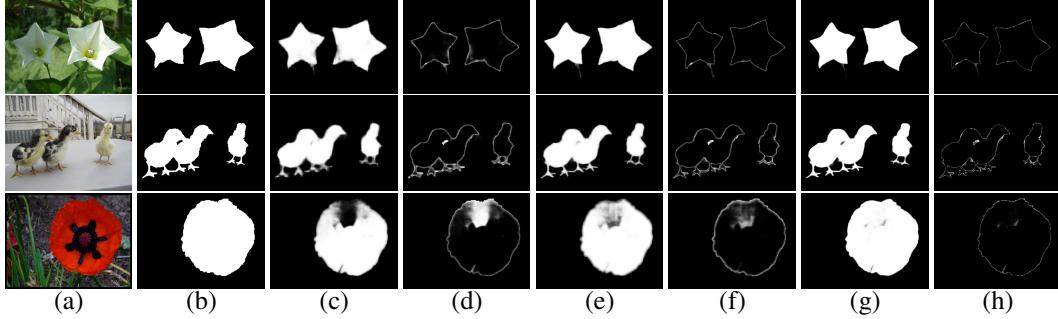
Figure 2: Samples of the predictions and error maps. (a) is input image. (b) is ground truth. (c) is saliency map of **baseline**. (d) is error map of (c). (e) is saliency map of **baseline+Grad** [47]. (f) is error map of (e). (g) is saliency map of **baseline+BRL**. (h) is error map of (g).

(a) Input Image   (b) Ground Truth   (c) Boundary Area
Figure 3: Example of boundary area generation.

3.2 Boundary Refinement Loss

3.2.1 Motivation

As boundary causes errors on saliency prediction as Figure 2 shows, recent work such as [71] and [54] added a boundary (or contour) detection branch to the framework to enhance foreground mask generator. However, these structures bring additional parameters and computation and make the system more complex. In addition, because of the fragility of boundary prediction, recent SE2Net[84] applied fully connected conditional random field (CRF) [30] on both salient region prediction and boundary prediction to refine results before post-processing. Performing CRF twice is quite time-consuming.

Differently, we propose a simple and yet effective refining strategy called Boundary Refinement Loss (BRL) on the existing segmentation branch. BRL only needs generating a loss on predictions, which involves negligible computation and no additional parameter. An efficient boundary refinement strategy is also proposed in [47] that utilizes gradients (denoted as 'Grad') to generate the IoU Loss. We note that the boundary mask generated by gradients is not stable since the boundary is composed of a lot of values between 0 and 1 while this method only covers a small portion of the boundary area. The loss generating area of BRL covers both foreground and background, which is more robust than that of [47]. Therefore, such a strategy of [47] performs less well than BRL, as shown in Table 10.

As shown in Figure 2(g)-(h), results refined by the proposed BRL have a much sharper boundary than baseline in (c)-(d) and 'Grad'[47] in (e)-(f). Even the paws can be clearly detected in (g). Additionally, as shown in the last row of Figure 2, the internal integrity of the salient region is better maintained if the boundary is refined well.

3.2.2 Loss Formulation

Before training, we generate a boundary area for each image as shown in Figure 3. Since the width of the gradient boundary generated by Sobel operator is very small, we extract the boundary mask as shown in Figure 3(c). Note that each pixel belonging to the original boundary expands on its surroundings to form the mask in (c). Therefore the new boundary area covers both background and foreground. We denote the expanding distance of each pixel as the *Width* of the boundary area. We find *Width*= 5 yields best results and set it as the default without specification.

During training, BRL simply adds an extra boundary loss L_b onto segmentation loss in the form of

$$L_b = \mathcal{T}(P \odot B, G \odot B) \tag{6}$$

where $\odot$ denotes pixel-wise multiplication, P is the foreground prediction, G is the ground truth, B is the boundary area generated before training as Figure 3(c) and $\mathcal{T}$ is the boundary loss function. In Eq. (6), G and P are masked by B, which filters out other areas and lets the loss function focus only on the foreground and background segmentation in the new boundary area. Similar to previous work [47], we choose the IoU Loss as $\mathcal{T}$.
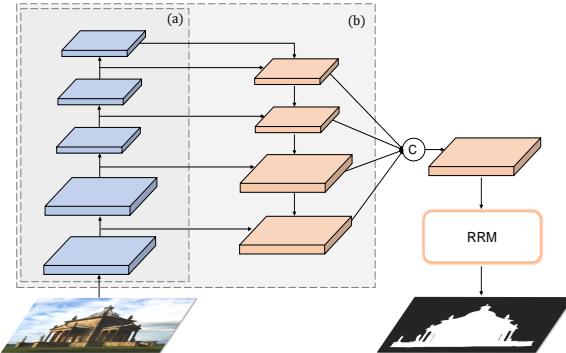
Figure 4: Network Structure. (a) is ResNet-50[18], (b) is FPN[39]. RRM is our proposed Region Refinement Module. "C" means feature concatenation followed by a pyramid pooling module.

$$L = L_f + \sigma \sum_{i=1}^{N} L_i + \eta(L_{bf} + \sigma \sum_{i=1}^{N} L_{bi}) \qquad (7)$$

If BRL is used, the total loss L is shown in Eq. (7) where L_{bf} and L_{bi} are boundary losses of Y_f and $\mathrm{Y}_i (i = 1, 2, ..., N)$, and η is used for weighing the sum of L_{bf} and L_{bi}. Following [47], we set η to 1.0 in all experiments. As shown later in experiments, BRL improves the results by a large margin by refining the boundary area on segmentation prediction.

3.3 Region Refinement Network

To prove the effectiveness of RRM and BRL, we proposed the Region Refinement Network (RRN) as shown in Figure 4. The network contains a base network (b) and a Region Refinement Module (RRM). The base network sends features extracted from input images to RRM for further refinement, and BRL is then applied to the outputs of RRM.

Our baseline model (Figure 4(b)) is built upon ResNet-50[18] (Figure 4(a)). In [43], it is observed that features in multiple levels together are helpful for accurate prediction. We accordingly modify the network in the FPN[39] style, as shown in Figure 4(b). By fusing features from four intermediate layers from FPN to form a multilevel feature before sending to the pyramid pooling module, the new base network outperforms the original PSPNet[81] by a large margin, as demonstrated in Table 10. The fusion of feature maps in different sizes is done by bilinear interpolation and concatenation.

For a fair comparison, we select VGG16 [53], ResNet-50 [18] and ResNet-101 [18] as our backbones. As for the VGG16 model, we extract features from Conv2_2, Conv3_3, Conv4_3 and Conv5_3 and send them to the FPN part to form a merged feature with $1/4$ of the resolution. For models based on ResNet-50 and 101, we use the same backbone configuration as PSPNet [81], and extract the last feature maps of the first MaxPool, Conv2, Conv3, Conv4 and Conv5 to the fusion stage. Features extracted from the backbone network are processed by 1×1 convolutions with 128 output channels.

4 Experiments

4.1 Settings

Our framework is based on PyTorch. The backbone network is initialized on VGG16, ResNet-50 and ResNet-101 trained on ImageNet. Other layers are initialized with PyTorch default setting. We adopt SGD with momentum as the optimizer where momentum and weight decay are set to 0.9 and 0.0001 respectively. Following [6], we use the 'poly' learning rate decay policy where current learning rate equals to the base learning rate multiplying $(1 - \frac{current_{iter}}{max_{iter}})^{power}$. We set the base rate to 0.01 and power to 0.9. As for hyper-parameters, we set N, M, $Width$, σ and η to 4, 3, 5, 2.0 and 1.0 respectively. Data augmentation is important for over-fitting prevention. In our experiments, we first perform mirroring and re-scaling from 0.75 and 1.25, and then add random rotation from -10 to 10 degrees on training images. Finally, we randomly crop 401×401 patches from the processed images as training samples. The batch size is 16, and we run 50 epochs for training in our setting. We output the saliency map from our network without additional refinement like the fully connected conditional random field (CRF) [30] used in [36], [20], [13] and [42]. Our experiments are conducted on an NVIDIA Tesla P40 GPU and Intel(R) Xeon(R) CPU E5-2680 v4 @ 2.40GHz.

Datasets We use the training data of DUTS [61] (10,553 images) for training as previous work [46],[77],[77],[42]. Then we evaluate our model on HKU-IS [35] (4,447 images), ECSSD [73] (1,000 images), DUTS [61] (5,018 images), DUT-OMRON [74] (5,168 images) and PASCAL-S [38] (850

Method	Training	ECSSD		HKU-IS		OMRON		DUTS		PASCAL-S	
		$F_\beta\uparrow$	$MAE\downarrow$	$F_\beta\uparrow$	$MAE\downarrow$	$F_\beta\uparrow$	$MAE\downarrow$	$F_\beta\uparrow$	$MAE\downarrow$	$F_\beta\uparrow$	$MAE\downarrow$
VGG-16 Backbone											
Amulet[2017] [78]	MS-10K	0.918	0.059	0.912	0.051	0.824	0.098	0.845	0.084	0.871	0.099
C2S-Net[2018] [71]	MS-10K	0.911	0.059	0.900	0.051	0.791	**0.079**	0.844	0.066	0.865	**0.086**
Ours	MS-10K	**0.921**	**0.052**	**0.922**	**0.041**	**0.827**	0.090	**0.853**	**0.064**	**0.873**	0.103
NLDF[2017] [47]	MS-B	0.910	0.063	0.909	0.048	0.798	0.079	0.842	0.065	0.857	0.098
Chen *et al.*[2018] [8]	MS-B	0.906	0.056	0.898	0.045	0.807	**0.062**	0.851	0.059	0.820	0.101
Ours	MS-B	**0.925**	**0.051**	**0.922**	**0.039**	**0.812**	0.073	**0.858**	**0.057**	**0.859**	**0.097**
PAGR[2018] [80]†	DUTS	0.911	0.061	0.904	0.047	0.768	0.071	0.849	0.055	0.851	0.089
BMPM[2018] [77]	DUTS	0.929	0.045	0.922	0.039	0.800	0.064	0.877	0.048	0.875	0.074
PiCANet[2018] [42]	DUTS	0.935	0.047	0.924	0.042	**0.833**	0.068	0.878	0.053	**0.886**	0.078
CPD[2019] [67]	DUTS	0.928	0.041	0.915	0.033	0.813	**0.057**	0.877	0.043	0.876	0.073
Ours	DUTS	**0.936**	**0.040**	**0.932**	**0.031**	0.809	0.060	**0.884**	**0.040**	**0.886**	**0.072**
ResNet-50 Backbone											
SRM[2017] [63]	DUTS	0.923	0.054	0.913	0.046	0.812	0.069	0.857	0.058	0.870	0.084
DGRL[2018] [64]	DUTS	0.931	0.046	0.918	0.041	0.821	0.066	0.868	0.053	0.873	0.082
PiCANet[2018] [42]	DUTS	0.938	0.047	0.923	0.043	**0.840**	0.065	0.886	0.050	0.889	0.075
CPD-R[2019] [67]	DUTS	0.937	0.037	0.916	0.034	0.826	0.056	0.878	0.043	0.881	0.071
Ours	DUTS	**0.946**	**0.034**	**0.941**	**0.027**	0.833	**0.054**	**0.893**	**0.038**	**0.900**	**0.067**
ResNet-101 Backbone											
R3Net-C[2018][13]‡	MS-10K	0.934	0.040	0.922	0.036	0.841	0.063	0.866	0.057	0.841	0.092
R3Net-C[2018][13]‡∗	DUTS	0.943	0.042	0.932	0.038	0.840	0.065	0.896	0.045	0.895	0.072
Ours	DUTS	**0.954**	**0.030**	**0.944**	**0.025**	**0.843**	**0.052**	**0.901**	**0.036**	**0.900**	**0.063**

Table 1: Results on five benchmark datasets. 'MS-10K': MSRA10K[12], 'MS-B': MSRA-B[45]. 'DUTS': the training data of DUTS[61]. 'C': results refined by CRF[30]. † : VGG-19 backbone. ‡: ResNeXt-101[69] backbone. ∗: Reproduced result. The best results are shown in **Red**.
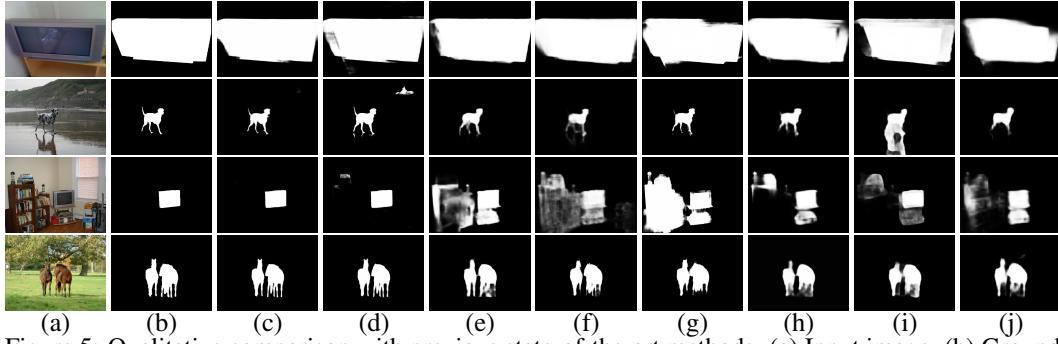

(a)　(b)　(c)　(d)　(e)　(f)　(g)　(h)　(i)　(j)

Figure 5: Qualitative comparison with previous state-of-the-art methods. (a) Input image. (b) Ground truth. (c) Baseline+BRL+RRM. (d) Baseline+BRL. (e) CPD-R [67]. (f) PiCANet-R [42]. (g) R3-Net [13]. (h) DGRL [64]. (i) C2S-Net [71]. (j) SRM [63]. Results of R3-Net are refined by CRF [30].

images). For a fair comparison, we also train our model (VGG16) on MSRA-10K[12](10000 images) and MSRA-B[45](2500 images).

Evaluation Metrics Our evaluation metrics are Mean Absolute Error (MAE) and maximum F-measure (F_β). MAE is the average pixel-wise error between prediction and ground truth as

$$MAE = \frac{1}{HW} \sum_{i=1}^{H} \sum_{j=1}^{W} |P(i,j) - G(i,j)| \tag{8}$$

where P is prediction, G is ground truth, H and W are height and width of the testing image. F_β is used for measuring the overall performance [8] with a balancing factor β as

$$F_\beta = \frac{(1 + \beta^2)(Recall + Precision)}{\beta^2 Precision + Recall} \tag{9}$$

where β^2 is set to 0.3 as suggested in [3]. A good result usually yields a large F_β and a small MAE.

4.2 Comparison with the State of the Art

In this section, we compare our proposed method with several state-of-the-art deep learning based algorithms shown in Table 1. For a fair comparison, we calculate the results on the salient maps; or the models provided by authors with the same evaluation code. It shows that our method achieves the new state-of-the-art on all backbone methods. The speeds of our final models with VGG16, ResNet50 and ResNet101 are 54.8, 52.1 and 32.8 FPS respectively on 352×352 input.

For visual comparison, we compare the salient maps with state-of-the-art methods (CPD-R [67], PiCANet-R [42], R3-Net [13], DGRL [64], C2S-Net [71], SRM [63]) shown in Figure 5. In Figure 5, our final model (c) is composed of both BRL and RRM. It is more robust to large objects,

Method	ECSSD $F_\beta\uparrow$	MAE$\downarrow$	HKU-IS $F_\beta\uparrow$	MAE$\downarrow$	OMRON $F_\beta\uparrow$	MAE$\downarrow$	DUTS $F_\beta\uparrow$	MAE$\downarrow$	PASCAL-S $F_\beta\uparrow$	MAE$\downarrow$
PSPNet[81]	0.930	0.048	0.928	0.039	0.820	0.066	0.886	0.046	0.877	0.081
Baseline	0.936	0.043	0.931	0.035	0.819	0.063	0.886	0.044	0.878	0.079
Baseline+Grad[47]	0.939	0.042	0.935	0.033	0.824	0.062	0.888	0.044	0.877	0.080
Baseline+BRL	0.943	0.038	0.934	0.030	**0.833**	0.061	0.889	0.042	0.882	0.076
Baseline+BRL+RRM	**0.946**	**0.034**	**0.941**	**0.027**	**0.833**	**0.054**	**0.893**	**0.038**	**0.900**	**0.067**

Table 2: Comparisons of different boundary refinement strategies. The baseline is shown in Figure 4(b). All models use ResNet50 as the backbone.

σ	ECSSD $F_\beta\uparrow$	MAE$\downarrow$	HKU-IS $F_\beta\uparrow$	MAE$\downarrow$	OMRON $F_\beta\uparrow$	MAE$\downarrow$	DUTS $F_\beta\uparrow$	MAE$\downarrow$	PASCAL-S $F_\beta\uparrow$	MAE$\downarrow$
0.00	0.940	0.037	0.936	0.030	0.830	0.059	0.892	0.042	0.878	0.074
0.25	**0.947**	**0.034**	0.940	**0.027**	0.832	0.057	**0.898**	0.039	0.891	**0.070**
0.50	0.946	**0.034**	**0.941**	**0.027**	0.833	**0.054**	0.893	**0.038**	**0.900**	**0.067**
0.75	0.946	0.035	0.939	0.028	**0.839**	0.057	0.895	0.040	0.889	0.072
1.00	0.944	0.036	0.940	0.028	**0.839**	0.057	0.891	0.040	0.880	0.073

Table 3: Comparison between results of different σ. The best results are shown in red.

tiny structures like limbs and ears, complex background, and multiple objects, which prove its effectiveness. Additionally, our method yields much better boundaries in salient maps than other methods.

4.3 Ablation Study and Analysis

4.3.1 Effectiveness of BRL

In order to demonstrate the effectiveness of our proposed Boundary Refinement Loss (BRL), we compare BRL with the enhancement strategy (denoted as 'Grad') proposed in [47] and the baseline structure in Table 10. 'Grad' yields slight improvements over the baseline model except for PASCAL-S in terms of F_β and MAE. Our proposed BRL makes the baseline model generate higher quality results on all benchmark datasets. The results are even comparable with previous state-of-the-art ones. The qualitative comparison is shown in Figure 2 where BRL in (g)-(h) refines boundary better than Grad [47] in (e)-(f). We put experiments on different boundary *Width* in the supplementary file.

Method	ECSSD $F_\beta\uparrow$	MAE$\downarrow$	HKU-IS $F_\beta\uparrow$	MAE$\downarrow$	OMRON $F_\beta\uparrow$	MAE$\downarrow$	DUTS $F_\beta\uparrow$	MAE$\downarrow$	PASCAL-S $F_\beta\uparrow$	MAE$\downarrow$
Mask	**0.946**	**0.034**	**0.941**	**0.027**	0.833	**0.054**	0.893	**0.038**	**0.900**	**0.067**
Non-Mask	0.943	0.036	0.937	0.030	**0.834**	0.058	**0.894**	0.042	0.890	0.074

Table 4: Comparisons of model with/without mask. $N = 4, M = 3$ and $\sigma = 0.5$ in two experiments.

4.3.2 Effectiveness of RRM

As shown in Figure 5(c), RRM further refines the results by reducing false prediction existing in Figure 5(d), and the model with RRM ($N = 4, M = 3$) achieves best performance in Table 10. In Eq. (7), we follow [47] to set η to 1.0. There are three parameters (N, M and σ) that are not decided.

In this section, we first conduct experiments on different values of σ to find the best balancing factor. Then, we show the effectiveness of the mask used as attention for controlling information flow. Examples illustrating the refinement process in each stage of RRM and experiments on different values of N (layer number) and M (convolution number) are shown in the supplementary material. We find RRM with ($N = 4, M = 3$) brings the best performance.

Balancing Factor (σ) The side outputs from intermediate layers of RRM generate auxiliary losses $\sigma \sum_{i=1}^{N} L_i$ that may influence the learning of final output. In Table 3, to find the best σ for balancing learning between side and final output, we select five values for σ as [0, 0.25, 0.5, 0.75, 1.0] and find that 0.5 yields the best performance. When $\sigma = 0$, there is no supervision on side outputs, which causes self-learned attention maps. However, as shown in our experiments, the supervision of ground truth on side outputs produces better attention maps to control the information flow in RRM. Also, when σ is close to one, the loss information back-propagated from the final output is dominated by previous layers' outputs; a large σ yields sub-optimal final results. Therefore σ should be close to 0.5 for optimal results shown in Table 3.

Mask in RRM In Figure 1, we show the output of the previous layer is used as an attention mask applied to the next layer to control the information flow. We compare models with and without masks in RRM to evaluate the usefulness of the mask and show results in Table 4. It is clear that when N and σ are set reasonably, without the mask to control the useful information flow in the module, redundant information becomes the bottleneck.

4.4 Applications

Our method (RRN) can be applied to other tasks to further prove its effectiveness. For Portrait segmentation based on the portrait dataset [52], we apply RRN and make a comparison with the results of [67]. As listed in Table 6, RRN (VGG-16 backbone) achieves the best mIoU performance.

	scGAN [48]	NLDF [47]	DSS [20]	BMPM [77]	JDR [59]	DSC [24]	CPD [67]	**Ours**
SBU	9.10	7.02	7.00	6.17	8.14	5.59	4.19	**4.12**
ISTD	8.98	7.50	10.48	7.10	7.35	8.24	6.76	**6.16**
UCF	11.50	7.69	10.56	8.09	11.23	8.10	7.21	**6.89**

Table 5: Comparisons of models on three shadow detection datasets: SBU, ISTD and UCF. The evaluation metrics is Balanced Error Rate (BER). A smaller BER means a better result.

Methods	PFCN+ [52]	NLDF [47]	DSS [20]	BMPM [77]	CPD [67]	**Ours**
mIoU	95.9%	95.6%	96.2%	96.2%	96.6%	**96.7%**

Table 6: Comparisons of models on Portrait Segmentation.

We also apply RRN (VGG-16 backbone) to shadow detection and evaluate it on the SBU [56, 72], ISTD [59] and UCF [85] datasets. We first train our model on SBU dataset, which is the largest publicly available annotated shadow dataset with 4,089 training images and test the model on SBU testing images, ISTD and UCF. We use the Balanced Error Rate (BER) as our evaluation metrics and the same evaluation code as [55, 56, 48]. As shown in Table 5, our method outperforms all previous state-of-the-art methods on the benchmark.

4.4.1 Other Frameworks

Region Refinement Module and Boundary Refinement Loss, as two independent refinement strategies, can be applied to other frameworks generally. Besides the baseline in Table 10, we also use it in Cascaded Partial Decoder (CPD) [67] (ResNet-50) and R3Net [13] (ResNeXt-101), two state-of-the-art SOD frameworks, to verify our designs. In our experiments, we use the default training setting and use fixed initial random seeds for fair comparisons. We simply insert RRM (without the final upsampling stage RRL_f) into the two models without any further change.

CPD We apply two 4-layers RRMs in two aggregation modules of CPD before the final convolution of each aggregation and use the pre-generated boundary masks for BRL during training. As listed in Table 7, though our reproduced baseline is less effective as claimed in the original paper, RRM and BRL both improve performance of the model on most of the benchmark datasets. They still help the final model (CPD+BRL+RRM) outperform the original one. It is worth noting that each RRM only brings about 3MB increase in terms of the model size. The speeds (on 352×352 input) of CPD and CPD with two RRMs are 47.3 and 36.2 FPS respectively.

Method	ECSSD		HKU-IS		OMRON		DUTS		PASCAL-S	
	$F_\beta\uparrow$	$MAE\downarrow$	$F_\beta\uparrow$	$MAE\downarrow$	$F_\beta\uparrow$	$MAE\downarrow$	$F_\beta\uparrow$	$MAE\downarrow$	$F_\beta\uparrow$	$MAE\downarrow$
CPD (original)	0.937	0.037	0.916	0.034	0.826	0.056	0.878	0.043	0.881	0.071
CPD (reproduced)	0.938	0.039	0.924	0.034	0.831	0.058	0.879	0.046	0.878	0.075
CPD + BRL	0.940	0.037	0.927	0.031	0.840	**0.056**	0.882	0.044	0.886	0.070
CPD + RRM	0.939	**0.036**	0.927	0.032	0.830	0.058	0.883	0.044	0.881	0.070
CPD + BRL + RRM	**0.941**	**0.036**	**0.932**	**0.030**	**0.846**	0.059	**0.890**	**0.043**	**0.900**	**0.066**

Table 7: Comparisons of CPD [67] with/without our proposed RRM and BRL.

R3Net On R3Net, we apply a single 4-layers RRM after the high-level integrated features and before the following Residual Refinement Blocks. R3Net in the original paper is trained on MSRA10K [12] with results refined by CRF[30]. We also train the improved models on DUTS-TR/MSRA10K and test them with/without CRF to verify the performance as shown in Table 8. The speeds (on 352×352 input) of R3Net and R3Net+RRM and R3Net+RRM+CRF are 21.8, 19.2 and 2.4 FPS.

Method	ECSSD		HKU-IS		OMRON		DUTS		PASCAL-S	
	$F_\beta\uparrow$	$MAE\downarrow$	$F_\beta\uparrow$	$MAE\downarrow$	$F_\beta\uparrow$	$MAE\downarrow$	$F_\beta\uparrow$	$MAE\downarrow$	$F_\beta\uparrow$	$MAE\downarrow$
DUTS + Non-CRF										
R3Net-D	0.943	0.042	0.932	0.038	0.840	0.065	0.896	0.045	0.895	0.072
R3Net-D + BRL	0.946	0.037	**0.936**	0.035	0.844	0.063	0.901	**0.040**	0.897	0.068
R3Net-D + RRM	0.943	0.040	0.933	0.034	0.843	0.063	0.900	0.042	**0.899**	0.069
R3Net-D + RRM + BRL	**0.947**	**0.035**	**0.936**	**0.030**	**0.850**	**0.062**	**0.903**	**0.040**	**0.899**	**0.066**
MSRA10K + CRF										
R3Net-M (original)	0.934	0.040	0.922	0.036	0.841	0.063	0.866	0.057	0.841	0.092
R3Net-M (reproduced)	0.936	0.040	0.927	0.034	0.845	0.069	0.866	0.062	0.843	0.092
R3Net-M + BRL	0.938	0.039	0.928	0.034	0.845	**0.065**	0.870	**0.060**	0.844	0.089
R3Net-M + RRM	0.936	0.038	0.929	0.032	0.846	0.067	0.869	0.060	0.853	**0.087**
R3Net-M + RRM + BRL	**0.940**	0.038	**0.931**	**0.031**	**0.851**	0.068	**0.875**	0.060	**0.857**	0.088

Table 8: Comparisons of R3Net with/without RRM and BRL. R3Net-D: Model trained on DUTS training data. R3Net-M: Model trained on MSRA10K. 'Original': the performance reported in paper.

5 Conclusion

We have presented a novel Region Refinement Module (RRM) to optimize saliency prediction with a gating mechanism. Our boundary refinement loss (BRL) refines the boundary directly on the prediction without introducing new parameters and extra computation. These two designs not only help our Region Refinement Network (RRN) achieve new state-of-the-art results on five popular benchmark datasets but also can be easily applied to other frameworks. Possible future work includes extending these designs to video saliency detection and further improving the performance.

6 Appendix

This section is the supplementary material for Region Refinement Network for Salient Object Detection. In Section 6.1, we illustrate the mean, variance and error bars of our model. More experiments on our proposed Region Refinement Module (RRM) and Boundary Refinement Loss (BRL) can be found in Section 6.2. In Section 6.3 and Section 6.4, we show more examples for visual comparison and links for publicly available datasets used in our experiments.

6.1 Mean, Variance and Error Bar

We train extra 10 models to show the mean, variance and error bars. Due to the limited computing resources, extra experiments are based on VGG16 backbone and trained 50 epochs on MSRA-B (2500 images) dataset to have a quick analysis. Finally, we evaluate these models on HKU-IS [35] (4447 images), ECSSD [73] (1000 images) DUTS [61] (5018 images), DUT-OMRON [74] (5168 images) and PASCAL-S [38] (850 images). The results are shown in Table 9 and Figure 6.

Model	ECSSD		HKU-IS		OMRON		DUTS		PASCAL-S	
	$F_\beta\uparrow$	$MAE\downarrow$	$F_\beta\uparrow$	$MAE\downarrow$	$F_\beta\uparrow$	$MAE\downarrow$	$F_\beta\uparrow$	$MAE\downarrow$	$F_\beta\uparrow$	$MAE\downarrow$
1	0.925	0.054	0.921	0.043	0.813	0.079	0.858	0.064	0.863	0.096
2	0.924	0.050	0.921	0.040	0.813	0.075	0.861	0.061	0.853	0.098
3	0.921	0.052	0.922	0.042	0.814	0.075	0.861	0.062	0.856	0.096
4	0.922	0.051	0.921	0.040	0.815	0.079	0.861	0.063	0.855	0.096
5	0.922	0.052	0.919	0.041	0.820	0.077	0.857	0.065	0.855	0.098
6	0.921	0.053	0.922	0.044	0.811	0.075	0.859	0.062	0.854	0.096
7	0.921	0.051	0.920	0.040	0.816	0.077	0.862	0.062	0.856	0.096
8	0.923	0.051	0.922	0.041	0.815	0.078	0.860	0.064	0.862	0.095
9	0.924	0.052	0.922	0.039	0.814	0.075	0.861	0.062	0.856	0.100
10	0.925	0.050	0.921	0.040	0.813	0.079	0.855	0.064	0.860	0.094
Mean	0.923	0.052	0.921	0.041	0.814	0.077	0.859	0.063	0.857	0.097
Std $(\times 10^{-3})$	1.536	1.200	0.943	1.483	2.289	1.700	2.110	1.221	3.256	1.628

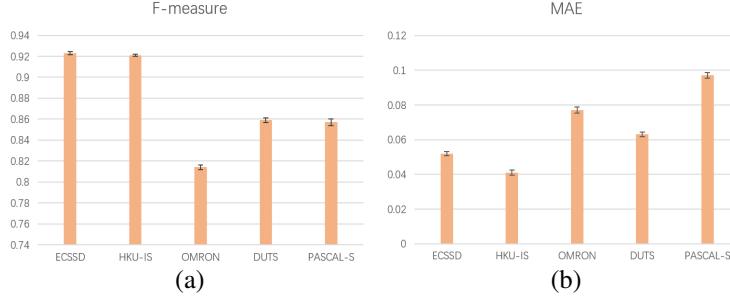Table 9: Extra 10 models to show the mean and standard deviation.

Figure 6: The error bar of F-measure (a) and MAE (b) on five datasets.

Method	ECSSD		HKU-IS		OMRON		DUTS		PASCAL-S	
	$F_\beta\uparrow$	$MAE\downarrow$	$F_\beta\uparrow$	$MAE\downarrow$	$F_\beta\uparrow$	$MAE\downarrow$	$F_\beta\uparrow$	$MAE\downarrow$	$F_\beta\uparrow$	$MAE\downarrow$
Baseline	0.936	0.043	0.931	0.035	0.819	0.063	0.886	0.044	0.878	0.079
Baseline+BRL	0.943	0.038	0.934	0.030	0.833	0.061	0.889	0.042	0.882	0.076
Baseline+BRL+RRM$_1$	0.945	0.035	0.938	0.028	**0.835**	0.059	0.892	0.040	0.883	0.074
Baseline+BRL+RRM$_2$	0.942	0.035	0.939	**0.027**	0.817	0.056	0.892	**0.038**	0.882	0.072
Baseline+BRL+RRM$_3$	0.946	0.035	**0.941**	**0.027**	**0.835**	0.055	0.893	**0.038**	0.883	0.072
Baseline+BRL+RRM$_4$	0.946	**0.034**	**0.941**	**0.027**	0.833	**0.054**	0.893	**0.038**	**0.900**	**0.067**
Baseline+BRL+RRM$_5$	0.946	0.035	0.940	**0.027**	0.833	0.056	0.884	0.039	0.894	0.070
Baseline+BRL+RRM$_6$	**0.947**	0.034	0.938	**0.027**	**0.835**	0.057	**0.895**	0.039	0.892	0.071
Baseline+BRL+RRM$_7$	0.945	0.035	0.839	0.028	0.832	0.057	0.888	0.040	0.888	0.070

Table 10: Comparisons of different layer numbers N. All models use ResNet50 as the backbone and are trained on DUTS training data.

6.2 More Details and Ablation Studies

6.2.1 Region Refinement Module (RRM)

In the proposed Regional Refinement Network (RRN) and applications on CPD[67] and R3Net[13], the input features have 64, 96 and 256 output channels respectively (we process the input feature in RRN by a 1×1 convolution with 64 output channels), therefore all intermediate 3×3 and 1×1 convolutions in RRM have same input and output channels, except for the last convolution in each RRL (Region Refinement Layer) outputs a salient map with 1 output channel.

Layer Number (N) The results of different N are shown in Table 10. Note that when N is set to zero, the model degrades to Baseline+BRL in Table 10. Our results from RRM$_0$ to RRM$_7$ are gradually refined, decent in terms of both F_β and MAE. We observe that the model with RRM$_4$ is

optimal in most cases even if it contains fewer parameters than $RRM_N(N > 4)$. We thus choose to apply RRM_4 in our final model.

Number of 3×3 and 1×1 Convolution (M)   Then, apart from the vertical analysis on layer number N, we further perform horizontal analysis on the number of convolutions. Since the numbers of 1×1 convolutions and 3×3 convolutions are equal in each Region Refinement Layer (RRL), we use M to denote the numbers of two convolutions in each RRL. The results of M on RRN are as Table 11 shows. $M = 3$ yields the best performance.

M	ECSSD $F_\beta\uparrow$	ECSSD $MAE\downarrow$	HKU-IS $F_\beta\uparrow$	HKU-IS $MAE\downarrow$	OMRON $F_\beta\uparrow$	OMRON $MAE\downarrow$	DUTS $F_\beta\uparrow$	DUTS $MAE\downarrow$	PASCAL-S $F_\beta\uparrow$	PASCAL-S $MAE\downarrow$
0	0.925	0.048	0.928	0.036	0.795	0.074	0.880	0.048	0.881	0.081
1	0.930	0.044	0.929	0.033	0.804	0.067	0.885	0.042	0.879	0.075
2	0.925	0.041	0.930	**0.031**	0.797	0.062	0.885	0.041	0.883	**0.072**
3	**0.936**	**0.040**	**0.932**	**0.031**	0.809	**0.060**	0.884	**0.040**	**0.886**	**0.072**
4	0.933	0.042	0.930	**0.031**	**0.811**	0.063	**0.890**	0.042	0.883	0.073

Table 11: Comparisons of different numbers of 3×3 and 1×1 convolutions in each RRL. All models ($N = 4$) are based on VGG16 and trained on DUTS training data[61]. $M = 0$ means no RRM.

RRM Refinement Process   We take outputs from RRL_1, RRL_2, RRL_3 and RRL_4 to better illustrate the refining process in RRM. As shown in Figure 7, false predictions are gradually eliminated from RRL_1 to RRL_4.
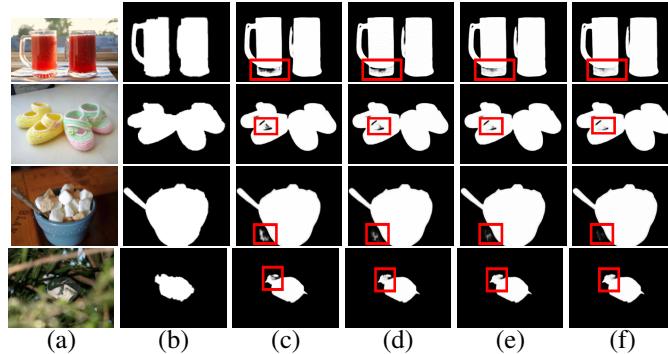

| (a) | (b) | (c) | (d) | (e) | (f) |

Figure 7: Visualization of the refinement process. (a) Input image. (b) Ground truth. (c)-(f) are outputs of RRL_1, RRL_2, RRL_3 and RRL_4.

6.2.2   Boundary Refinement Learning (BRL)

Boundary Mask Generation   In BRL, we use boundary mask for extra supervision on boundary area. In this section, we show the impacts of different boundary widths. As shown in Table 12, $Width = 5$ yields the best performance.

Method	ECSSD $F_\beta\uparrow$	ECSSD $MAE\downarrow$	HKU-IS $F_\beta\uparrow$	HKU-IS $MAE\downarrow$	OMRON $F_\beta\uparrow$	OMRON $MAE\downarrow$	DUTS $F_\beta\uparrow$	DUTS $MAE\downarrow$	PASCAL-S $F_\beta\uparrow$	PASCAL-S $MAE\downarrow$
No Bound	0.917	0.061	0.915	0.050	0.804	0.088	0.852	0.074	0.853	0.105
Width=0	0.917	0.056	0.918	0.048	0.806	0.079	0.853	0.073	0.849	0.099
Width=1	0.923	0.052	0.918	0.045	0.812	0.086	0.855	0.071	0.855	0.100
Width=3	0.923	0.051	**0.921**	0.042	0.808	0.085	0.855	0.068	**0.862**	0.099
Width=5	**0.925**	**0.050**	**0.921**	**0.039**	0.810	0.073	**0.861**	**0.058**	0.852	**0.096**
Width=7	0.924	**0.050**	0.920	0.041	0.807	0.077	0.857	0.062	0.855	0.098
Width=9	0.922	0.051	0.918	0.040	0.809	**0.072**	0.860	**0.058**	0.851	0.098
Width=11	0.921	0.054	0.920	0.045	**0.814**	0.090	0.853	0.074	0.854	0.102

Table 12: Comparisons of different boundary widths in BRL. All models ($N = 4$, $M = 3$) are based on VGG16 and trained on MSRA-B training data[61]. 'No Bound' means BRL is not implemented. $Width = 0$ means the 'Grad' enhancement strategy of [47] is implemented.

6.2.3   More Results of R3Net

We provide the detailed results of our application on R3Net [13] in Table 13 where we add the performance of DUTS+CRF and MSRA10K+Non-CRF to the original table.

Method	ECSSD		HKU-IS		OMRON		DUTS		PASCAL-S	
	$F_\beta\uparrow$	$MAE\downarrow$	$F_\beta\uparrow$	$MAE\downarrow$	$F_\beta\uparrow$	$MAE\downarrow$	$F_\beta\uparrow$	$MAE\downarrow$	$F_\beta\uparrow$	$MAE\downarrow$
DUTS+Non-CRF										
R3Net-D	0.943	0.042	0.932	0.038	0.840	0.065	0.896	0.045	0.895	0.072
R3Net-D + BRL	0.946	0.037	**0.936**	0.035	0.844	0.063	0.901	**0.040**	0.897	0.068
R3Net-D + RRM	0.943	0.040	0.933	0.034	0.843	0.063	0.900	0.042	**0.899**	0.069
R3Net-D + RRM + BRL	**0.947**	**0.035**	**0.936**	**0.030**	**0.850**	**0.062**	**0.903**	**0.040**	**0.899**	**0.066**
DUTS + CRF										
R3Net-D	0.945	0.032	0.936	0.028	0.840	0.056	0.900	0.038	0.893	0.065
R3Net-D + BRL	0.947	0.032	**0.938**	0.027	0.843	0.058	0.904	0.036	0.898	0.064
R3Net-D + RRM	0.946	0.032	0.936	0.027	0.845	**0.055**	0.901	**0.035**	**0.899**	0.062
R3Net-D + RRM + BRL	**0.949**	**0.030**	**0.938**	**0.026**	**0.849**	0.058	**0.906**	0.036	**0.899**	**0.061**
MSRA10K + Non-CRF										
R3Net-M	0.933	0.050	0.921	0.046	0.841	0.078	0.862	0.070	0.852	0.100
R3Net-M + BRL	0.936	0.044	0.922	0.040	0.842	**0.072**	0.866	0.067	0.857	0.097
R3Net-M + RRM	0.938	0.047	0.924	0.044	0.845	0.077	0.868	0.066	0.859	0.095
R3Net-M + RRM + BRL	**0.939**	**0.042**	**0.928**	**0.037**	**0.848**	0.074	**0.871**	**0.065**	**0.866**	**0.091**
MSRA10K + CRF										
R3Net-M (original)	0.934	0.040	0.922	0.036	0.841	0.063	0.866	0.057	0.841	0.092
R3Net-M (reproduced)	0.936	0.040	0.927	0.034	0.845	0.069	0.866	0.062	0.843	0.092
R3Net-M + BRL	0.938	0.039	0.928	0.034	0.845	**0.065**	0.870	**0.060**	0.844	0.089
R3Net-M + RRM	0.936	0.038	0.929	0.032	0.846	0.067	0.869	0.060	0.853	**0.087**
R3Net-M + RRM + BRL	**0.940**	**0.038**	**0.931**	**0.031**	**0.851**	0.068	**0.875**	**0.060**	**0.857**	0.088

Table 13: Comparisons of R3Net with/without RRM and BRL. R3Net-D: Model trained on DUTS training data. R3Net-M: Model trained on MSRA10K. 'Original': the performance reported in paper.

6.3 More Examples of Visual Comparisons

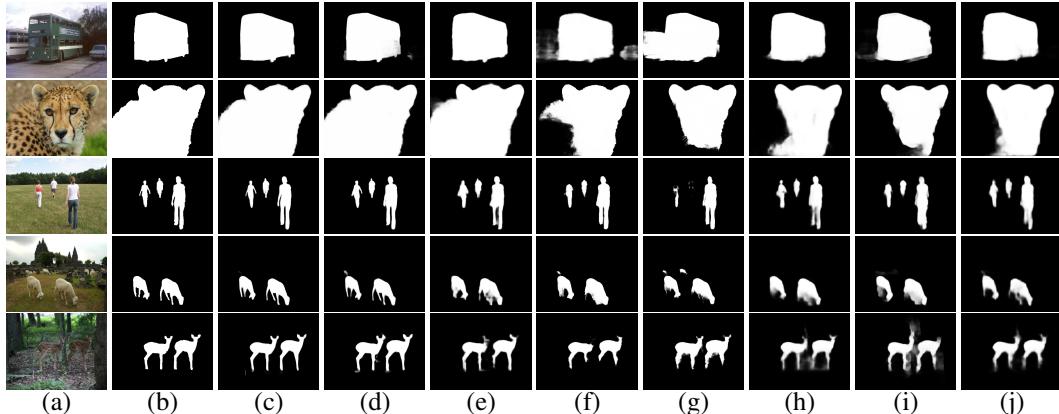More visual comparisons are shown in Figure 8.

Figure 8: Qualitative comparison with previous state-of-the-art methods. (a) Input image. (b) Ground truth. (c) Baseline+BRL+RRM. (d) Baseline+BRL. (e) CPD-R [67]. (f) PiCANet-R [42]. (g) R3-Net [13]. (h) DGRL [64]. (i) C2S-Net [71]. (j) SRM [63]. Results of R3-Net are refined by CRF [30].

6.4 Links for Datasets

6.4.1 Salient Object Detection

HKU-IS [35]: https://i.cs.hku.hk/~gbli/deep_saliency.html
ECSSD [73]: http://www.cse.cuhk.edu.hk/leojia/projects/hsaliency/dataset.html
DUTS [61]: http://saliencydetection.net/duts/
DUT-OMRON [74]: http://saliencydetection.net/dut-omron/
PASCAL-S [38]: http://www.cbi.gatech.edu/salobj/
MSRA-B [45] & **MSRA-10K** [12]: https://mmcheng.net/zh/msra10k/

6.4.2 Shadow Detection

UCF [85]: http://aqua.cs.uiuc.edu/site/projects/shadow.html
SBU [56, 72]: https://www3.cs.stonybrook.edu/~cvl/dataset.html
ISTD [59]: https://drive.google.com/file/d/1I0qw-65KBA6np8vIZzO6oeiOvcDBttAY/view

6.4.3 Portrait Segmentation

Flickr Portrait Dataset[52]: http://xiaoyongshen.me/webpage_portrait/index.html

References

[1] R. Achanta, S. S. Hemami, F. J. Estrada, and S. Süsstrunk. Frequency-tuned salient region detection. In *CVPR*, 2009.

[2] A. Borji, M. Cheng, Q. Hou, H. Jiang, and J. Li. Salient object detection: A survey. *Arxiv Preprint*, 2014.

[3] A. Borji, D. N. Sihite, and L. Itti. Salient object detection: A benchmark. In *ECCV*, 2012.

[4] H. Chen and Y. Li. Progressively complementarity-aware fusion network for RGB-D salient object detection. In *CVPR*, 2018.

[5] L. Chen, A. Hermans, G. Papandreou, F. Schroff, P. Wang, and H. Adam. Masklab: Instance segmentation by refining object detection with semantic and direction features. In *CVPR*, 2018.

[6] L. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille. Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. *IEEE Trans. Pattern Anal. Mach. Intell.*, 2018.

[7] L. Chen, G. Papandreou, F. Schroff, and H. Adam. Rethinking atrous convolution for semantic image segmentation. *Arxiv Preprint*, 2017.

[8] S. Chen, X. Tan, B. Wang, and X. Hu. Reverse attention for salient object detection. In *ECCV*, 2018.

[9] X. Chen, A. Zheng, J. Li, and F. Lu. Look, perceive and segment: Finding the salient objects in images via two-stream fixation-semantic cnns. In *ICCV*, 2017.

[10] M. Cheng, Q. Hou, S. Zhang, and P. L. Rosin. Intelligent visual media processing: When graphics meets vision. *J. Comput. Sci. Technol.*, 2017.

[11] M. Cheng, J. Warrell, W. Lin, S. Zheng, V. Vineet, and N. Crook. Efficient salient region detection with soft image abstraction. In *ICCV*, 2013.

[12] M. Cheng, G. Zhang, N. J. Mitra, X. Huang, and S. Hu. Global contrast based salient region detection. In *CVPR*, 2011.

[13] Z. Deng, X. Hu, L. Zhu, X. Xu, J. Qin, G. Han, and P. Heng. R^3net: Recurrent residual refinement network for saliency detection. In *IJCAI*, 2018.

[14] D. Fan, M. Cheng, J. Liu, S. Gao, Q. Hou, and A. Borji. Salient objects in clutter: Bringing salient object detection to the foreground. In *ECCV*, 2018.

[15] R. Fan, Q. Hou, M. Cheng, T. Mu, and S. Hu. S^4net: Single stage salient-instance segmentation. *Arxiv Preprint*, 2017.

[16] R. Fan, Q. Hou, M. Cheng, G. Yu, R. R. Martin, and S. Hu. Associating inter-image salient instances for weakly supervised semantic segmentation. In *ECCV*, 2018.

[17] K. He, G. Gkioxari, P. Dollár, and R. B. Girshick. Mask R-CNN. In *ICCV*, 2017.

[18] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *CVPR*, 2016.

[19] S. He, J. Jiao, X. Zhang, G. Han, and R. W. H. Lau. Delving into salient object subitizing and detection. In *ICCV*, 2017.

[20] Q. Hou, M. Cheng, X. Hu, A. Borji, Z. Tu, and P. H. S. Torr. Deeply supervised salient object detection with short connections. In *CVPR*, 2017.

[21] Q. Hou, M. Cheng, X. Hu, A. Borji, Z. Tu, and P. H. S. Torr. Deeply supervised salient object detection with short connections. *IEEE Trans. Pattern Anal. Mach. Intell.*, 2019.

[22] Q. Hou, D. Masiceti, P. K. Dokania, Y. Wei, M. Cheng, and P. H. S. Torr. Bottom-up top-down cues for weakly-supervised semantic segmentation. In *EMMCVPR*, 2017.

[23] P. Hu, B. Shuai, J. Liu, and G. Wang. Deep level sets for salient object detection. In *CVPR*, 2017.

[24] X. Hu, L. Zhu, C. Fu, J. Qin, and P. Heng. Direction-aware spatial context features for shadow detection. In *CVPR*, 2018.

[25] M. A. Islam, M. Kalash, and N. D. B. Bruce. Revisiting salient object detection: Simultaneous detection, ranking, and subitizing of multiple salient objects. In *CVPR*, 2018.

[26] B. Jiang, L. Zhang, H. Lu, C. Yang, and M. Yang. Saliency detection via absorbing markov chain. In *ICCV*, 2013.

[27] C. Jiang, H. Xu, X. Liang, and L. Lin. Hybrid knowledge routed modules for large-scale object detection. In *NeurIPS*, 2018.

[28] P. Jiang, F. Gu, Y. Wang, C. Tu, and B. Chen. Difnet: Semantic segmentation by diffusion networks. In *NeurIPS*, 2018.

[29] Z. Jiang and L. S. Davis. Submodular salient region detection. In *CVPR*, 2013.

[30] P. Krähenbühl and V. Koltun. Efficient inference in fully connected crfs with gaussian edge potentials. In *NeurIPS*, 2011.

[31] S. S. S. Kruthiventi, V. Gudisa, J. H. Dholakiya, and R. V. Babu. Saliency unified: A deep architecture for simultaneous eye fixation prediction and salient object segmentation. In *CVPR*, 2016.

[32] J. Kuen, Z. Wang, and G. Wang. Recurrent attentional networks for saliency detection. In *CVPR*, 2016.

[33] G. Lee, Y. Tai, and J. Kim. Deep saliency with encoded low level distance map and high level features. In *CVPR*, 2016.

[34] G. Li, Y. Xie, L. Lin, and Y. Yu. Instance-level salient object segmentation. In *CVPR*, 2017.

[35] G. Li and Y. Yu. Visual saliency based on multiscale deep features. In *CVPR*, 2015.

[36] G. Li and Y. Yu. Deep contrast learning for salient object detection. In *CVPR*, 2016.

[37] X. Li, H. Lu, L. Zhang, X. Ruan, and M. Yang. Saliency detection via dense and sparse reconstruction. In *ICCV*, 2013.

[38] Y. Li, X. Hou, C. Koch, J. M. Rehg, and A. L. Yuille. The secrets of salient object segmentation. In *CVPR*, 2014.

[39] T. Lin, P. Dollár, R. B. Girshick, K. He, B. Hariharan, and S. J. Belongie. Feature pyramid networks for object detection. In *CVPR*, 2017.

[40] J. Liu, Q. Hou, M. Cheng, J. Feng, and J. Jiang. A simple pooling-based design for real-time salient object detection. In *CVPR*, 2019.

[41] N. Liu and J. Han. Dhsnet: Deep hierarchical saliency network for salient object detection. In *CVPR*, 2016.

[42] N. Liu, J. Han, and M. Yang. Picanet: Learning pixel-wise contextual attention for saliency detection. In *CVPR*, 2018.

[43] S. Liu, L. Qi, H. Qin, J. Shi, and J. Jia. Path aggregation network for instance segmentation. In *CVPR*, 2018.

[44] T. Liu, J. Sun, N. Zheng, X. Tang, and H. Shum. Learning to detect A salient object. In *CVPR*, 2007.

[45] T. Liu, Z. Yuan, J. Sun, J. Wang, N. Zheng, X. Tang, and H. Shum. Learning to detect a salient object. *IEEE Trans. Pattern Anal. Mach. Intell.*, 2011.
[46] Y. Liu, D. Fan, G. Nie, X. Zhang, V. Petrosyan, and M. Cheng. DNA: deeply-supervised nonlinear aggregation for salient object detection. *Arxiv Preprint*, 2019.
[47] Z. Luo, A. K. Mishra, A. Achkar, J. A. Eichel, S. Li, and P. Jodoin. Non-local deep features for salient object detection. In *CVPR*, 2017.
[48] V. Nguyen, T. F. Y. Vicente, M. Zhao, M. Hoai, and D. Samaras. Shadow detection with conditional generative adversarial networks. In *ICCV*, 2017.
[49] Y. Qin, H. Lu, Y. Xu, and H. Wang. Saliency detection via cellular automata. In *CVPR*, 2015.
[50] S. Ren, K. He, R. B. Girshick, and J. Sun. Faster R-CNN: towards real-time object detection with region proposal networks. In *NeurIPS*, 2015.
[51] C. Scharfenberger, A. Wong, K. Fergani, J. S. Zelek, and D. A. Clausi. Statistical textural distinctiveness for salient region detection in natural images. In *CVPR*, 2013.
[52] X. Shen, A. Hertzmann, J. Jia, S. Paris, B. L. Price, E. Shechtman, and I. Sachs. Automatic portrait segmentation for image stylization. *Comput. Graph. Forum*, 2016.
[53] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. In *ICLR*, 2015.
[54] J. Su, J. Li, C. Xia, and Y. Tian. Selectivity or invariance: Boundary-aware salient object detection. *Arxiv Preprint*, 2018.
[55] T. F. Y. Vicente, M. Hoai, and D. Samaras. Noisy label recovery for shadow detection in unfamiliar domains. In *CVPR*, 2016.
[56] T. F. Y. Vicente, L. Hou, C.-P. Yu, M. Hoai, and D. Samaras. Large-scale training of shadow detectors with noisily-annotated shadow examples. In *ECCV*, 2016.
[57] J. Wang, T. A. Bohn, and C. X. Ling. Pelee: A real-time object detection system on mobile devices. In *NeurIPS*, 2018.
[58] J. Wang, H. Jiang, Z. Yuan, M. Cheng, X. Hu, and N. Zheng. Salient object detection: A discriminative regional feature integration approach. *IJCV*, 2017.
[59] J. Wang, X. Li, and J. Yang. Stacked conditional generative adversarial networks for jointly learning shadow detection and shadow removal. In *CVPR*, 2018.
[60] L. Wang, H. Lu, X. Ruan, and M. Yang. Deep networks for saliency detection via local estimation and global search. In *CVPR*, 2015.
[61] L. Wang, H. Lu, Y. Wang, M. Feng, D. Wang, B. Yin, and X. Ruan. Learning to detect salient objects with image-level supervision. In *CVPR*, 2017.
[62] Q. Wang, W. Zheng, and R. Piramuthu. Grab: Visual saliency via novel graph model and background priors. In *CVPR*, 2016.
[63] T. Wang, A. Borji, L. Zhang, P. Zhang, and H. Lu. A stagewise refinement model for detecting salient objects in images. In *ICCV*, 2017.
[64] T. Wang, L. Zhang, S. Wang, H. Lu, G. Yang, X. Ruan, and A. Borji. Detect globally, refine locally: A novel approach to saliency detection. In *CVPR*, 2018.
[65] W. Wang, J. Shen, X. Dong, and A. Borji. Salient object detection driven by fixation prediction. In *CVPR*, 2018.
[66] Y. Wei, F. Wen, W. Zhu, and J. Sun. Geodesic saliency using background priors. In *ECCV*, 2012.
[67] Z. Wu, L. Su, and Q. Huang. Cascaded partial decoder for fast and accurate salient object detection. In *CVPR*, 2019.
[68] C. Xia, J. Li, X. Chen, A. Zheng, and Y. Zhang. What is and what is not a salient object? learning salient object detector by ensembling linear exemplar regressors. In *CVPR*, 2017.
[69] S. Xie, R. B. Girshick, P. Dollár, Z. Tu, and K. He. Aggregated residual transformations for deep neural networks. In *CVPR*, 2017.
[70] S. Xie and Z. Tu. Holistically-nested edge detection. In *ICCV*, 2015.
[71] L. Xin, Y. Fan, C. Hong, L. Wei, and S. Dinggang. Contour knowledge transfer for salient object detection. In *ECCV*, 2018.
[72] T. F. Yago Vicente, M. Hoai, and D. Samaras. Noisy label recovery for shadow detection in unfamiliar domains. In *CVPR*, 2016.
[73] Q. Yan, L. Xu, J. Shi, and J. Jia. Hierarchical saliency detection. In *CVPR*, 2013.
[74] C. Yang, L. Zhang, H. Lu, X. Ruan, and M. Yang. Saliency detection via graph-based manifold ranking. In *CVPR*, 2013.
[75] D. Zhang, J. Han, and Y. Zhang. Supervision by fusion: Towards unsupervised learning of deep salient object detector. In *ICCV*, 2017.
[76] J. Zhang, T. Zhang, Y. Dai, M. Harandi, and R. I. Hartley. Deep unsupervised saliency detection: A multiple noisy labeling perspective. In *CVPR*, 2018.
[77] L. Zhang, J. Dai, H. Lu, Y. He, and G. Wang. A bi-directional message passing model for salient object detection. In *CVPR*, 2018.
[78] P. Zhang, D. Wang, H. Lu, H. Wang, and X. Ruan. Amulet: Aggregating multi-level convolutional features for salient object detection. In *ICCV*, 2017.
[79] P. Zhang, D. Wang, H. Lu, H. Wang, and B. Yin. Learning uncertain convolutional features for accurate saliency detection. In *ICCV*, 2017.
[80] X. Zhang, T. Wang, J. Qi, H. Lu, and G. Wang. Progressive attention guided recurrent network for salient object detection. In *CVPR*, 2018.
[81] H. Zhao, J. Shi, X. Qi, X. Wang, and J. Jia. Pyramid scene parsing network. In *CVPR*, 2017.
[82] R. Zhao, W. Ouyang, H. Li, and X. Wang. Saliency detection by multi-context deep learning. In *CVPR*, 2015.
[83] P. Zhou, B. Ni, C. Geng, J. Hu, and Y. Xu. Scale-transferrable object detection. In *CVPR*, 2018.
[84] S. Zhou, J. Wang, F. Wang, and D. Huang. Se2net: Siamese edge-enhancement network for salient object detection. *Arxiv Preprint*, 2019.
[85] J. Zhu, K. G. G. Samuel, S. Z. Masood, and M. F. Tappen. Learning to recognize shadows in monochromatic natural images. In *CVPR*, 2010.

[86] W. Zhu, S. Liang, Y. Wei, and J. Sun. Saliency optimization from robust background detection. In *CVPR*, 2014.