Recursive Multi-model Complementary Deep Fusion for Robust Salient Object Detection via Parallel Sub Networks

Zhenyu Wu\textsuperscript{1} Shuai Li\textsuperscript{1} Chenglizhao Chen\textsuperscript{1,2}\textsuperscript{*} Aimin Hao\textsuperscript{1} Hong Qin\textsuperscript{3}
\textsuperscript{1}Beihang University \textsuperscript{2}Qingdao University \textsuperscript{3}Stonybrook University

Abstract

Fully convolutional networks have shown outstanding performance in the salient object detection (SOD) field. The state-of-the-art (SOTA) methods have a tendency to become deeper and more complex, which easily homogenize their learned deep features, resulting in a clear performance bottleneck. In sharp contrast to the conventional “deeper” schemes, this paper proposes a “wider” network architecture which consists of parallel sub networks with totally different network architectures. In this way, those deep features obtained via these two sub networks will exhibit large diversity, which will have large potential to be able to complement with each other. However, a large diversity may easily lead to the feature conflicts, thus we use the dense short-connections to enable a recursively interaction between the parallel sub networks, pursuing an optimal complementary status between multi-model deep features. Finally, all these complementary multi-model deep features will be selectively fused to make high-performance salient object detections. Extensive experiments on several famous benchmarks clearly demonstrate the superior performance, good generalization, and powerful learning ability of the proposed wider framework.

Keywords: Salient Object Detection, Deep Learning, Multi-model Fusion
1. Introduction

The objective of salient object detection is to identify the most visually distinctive object in the given image [1]. As a preprocessing tool, salient object detection (SOD) has a wide range of practical applications including visual tracking [2], localization [3], video saliency [4], image captioning [5][6], image retrieval [7], visual question answering [8] and object retargeting [9].

Previous works frequently treat the SOD as a multi-level perception task [10][11][12], in which its key rationale is to make full use of the saliency clues at different perception levels [13]. Recently, the fully convolutional networks (FCNs) has been adopted for the robust SOD, in which such success should be attributed to its ability to learn hierarchical saliency clues. Thus, the current state-of-the-art (SOTA) models [14][15][16] generally focus on how to utilize the hierarchical deep features in “single network” to produce high-quality SOD. Nevertheless, the hierarchical deep features revealed in an identical network have a tendency to be homogenization, resulting in a limited performance eventually.

In the view of the neuroscience, human visual system mainly comprises two largely independent subsystems that mediate different classes of visual behaviors [17][18]. The subcortical projection from the retina to the cerebral cortex is strongly dominated by the two pathways that are relayed by the magnocellular (M) and parvocellular (P) subdivisions of the lateral geniculate nucleus (LGN), in which the Parallel pathways generally exhibit two main characteristics: 1) the M cells contribute to the low-level transient processing (e.g., visual motion perception, eye movement, etc.) while the P cells contribute more to the high-level recognition tasks (e.g., object recognition, face recognition, etc.); 2) the M and P cells are separated in the LGN, but it is recombined in visual cortex latter.

Motivated by the above-mentioned theory, we propose to use two parallel networks (see Fig. 1) to mimic the binocular vision of human visual system. The key point of the
Figure 1: The major difference between our method and the conventional methods.

Our proposed parallel network architecture is its ability to conduct multi-level saliency estimation while avoiding the conventional single network architecture induced feature homogenization problem. To achieve it, we devise a novel multi-model deep fusion framework, which attempts to fully exploit the complementary deep features from two different parallel subnetworks; i.e., the coarse-level saliency localization network and the fine-scale detail polishing network. Meanwhile, inspired by the aforementioned attributes, we adopt the inter-model short-connections to recursively ensure a complementary status between each of our subnetworks. Moreover, we utilize an FCNs based saliency regressor to conduct selective deep fusion over those inter-model deep features, achieving a high-performance SOD eventually.

It should be noted that our “wide” scheme is solely implemented by using simple network architecture, yet it has achieved remarkable performance improvement comparing to the conventional complicated “deeper” schemes. And such performance improvements are mainly induced by the newly designed multi-model fusion scheme, in which the adopted simple network architecture is a hallmark of the proposed method. Moreover, as far as we knew, our paper is the first attempt to handle the SOD from the “wider” perspective.

To demonstrate the advantages of our method, we have conducted massive quantitative comparisons against 14 most representative SOTA methods over 5 widely used publicly available datasets. Also, we have conducted extensive ablation studies to comprehensively verify the effectiveness of each essential component in our method.
Specifically, the salient contributions of this paper can be summarized as follows:

- We provide a deeper insight into the SOD task by imitating the binocular vision of human perception process;

- To alleviate the obstinate feature homogenization problem in single network case, we utilize parallel sub-networks to automatically reveal saliency clues at different spatial levels;

- We propose an end-to-end salient object detection model that learns diversity saliency clues in an iterative manner, aiming to achieve an optimal complementary status between the deep features extracted by our parallel sub-networks;

- We also provide a novel selective fusion strategy to fuse multi-model saliency clues for high-performance salient object detection, archiving the new SOTA performance over the five adopted datasets.

- The source code is available at: https://github.com/Diamond101010/RMMDF, which may has large potential to benefit the image salient object detection community in the future.

2. Related Work

Early methods largely adopt various hand-crafted visual features [19, 20, 21] to model the human visual attention [22], which are limited in generalization and effectiveness. Moreover, these methods have low computational efficiency and damage the potential feature structure. See [23, 24] for more details about traditional. Here we mainly discuss deep learning based saliency detection models.

2.1. Single-stream Network

The single-stream network is one of standard architecture adopted by many state-of-the-art methods, consisting of a sequential cascade of convolution layers, pooling
layers and non-linear activation operations. Li et al. [25] first proposed a convolutional neural networks (CNNs) based computational model, which incorporates the multi-scale deep features via simply vector-wise feature concatenation. Inspired by the great success of Fully Convolutional Network (FCN) [26] in semantic segmentation, recent deep SOD models adapt popular classification models, e.g., VGGNet and ResNet to directly output whole saliency maps. Wang et al. [27] propose a recurrent fully convolutional networks which recurrently refines the saliency prediction based on the input image and the saliency priors from heuristic calculation or prediction of previous time step. Similarly, Liu et al. [28] proposed a hierarchical refinement model in which the coarse saliency map by gradually combining shallower features using recurrent layers. In [14], short connections are introduced from deeper side-outputs to shallower ones. In this way, higher-level features can help lower side-outputs to better locate the salient regions, while lower-level features can help enrich the higher-level side-outputs with finer details. Zhang et al. [29] also proposed a novel method to aggregate multi-level CNNs based deep features, in which the key rationale is to simultaneously integrate high-level semantical information with low-level details for the robust SOD. Wang et al. [30] present progressive feature polishing network, a simple yet effective framework to progressively polish the multi-level features to be more accurate and representative.

2.2. Multi-stream Networks

The recent development of network architecture has a tendency to become deeper and more complicated [31]. Zeiler et al. [32] have demonstrated that a deeper architecture can generally generate more discriminative features at the expense of more complex architecture, leading the network difficult to train. In sharp contrast to the “deeper” strategy, the “wider” architecture may become an intuitive choice, in this paper the term “wider” means to design network architecture with parallel sub-networks. For example, Lin et al. [33] proposed a bilinear architecture, utilizing two feature extractors to obtain multi-scale deep features for image recognition. Saito et al. [34] proposed a nov-
el model for visual question answering, which attempts to learn discriminative features by using two independent sub-networks to conduct feature extraction for multi-source data. Kim et al. [35] proposed to utilize a newly designed parallel feature pyramid network for object detection. Yang et al. [36] present a deep compact code learning solution for efficient cross-modal similarity search. Deng et al. [37] propose a novel strategy to exploit the semantic similarity of the training data and design an efficient generative adversarial framework. Deng et al. [38] propose a novel two-stream ConvNet architecture, which learns hash codes with class-specific representation centers.

Recently, Multi-stream network, which typically has multiple network streams for explicitly learning multi-scale saliency features with different structures, is adopted in the image saliency detection and achieve promising results. Zhao et al. [39] designed a multi-context deep learning framework, in which the parallel revealed global context and local context are combined in an unified deep learning framework to jointly locate the salient object. Wang et al. [40] utilized parallel sub-networks to respectively conduct pixel-level/object-level saliency computation, and then the revealed saliency clues will be fused as the SOD result. Li et al. [41] built a multi-task deep network to explore the common saliency consistency between the salient object detection and the semantic segmentation. Wang et al. [42] design a two-stream network, i.e., a classification network and a caption generation network, to highlight the most important regions for corresponding tasks. Wu et al. [43] propose to integrate features of deeper layers in attention stream to get an initial saliency map, which is used to refine the features of the detection stream to generate the final map.

Actually, the “wider” structure has its merit to balance the trade-off between the saliency performance and the network complexity. However, because the parallel structure adopted by the above-mentioned methods are trained independently, those parallel learned deep features may not be able to effectively complement each other, not to mention those feature conflicts may lead the overall performance even worse.
In contrast to the above-mentioned methods, the proposed model is completely different in 2 aspects: 1) We utilize a novel recursive learning strategy to train parallel sub-networks to obtain a complementary status between two subnetworks; 2) As for those already learned complementary deep features, we utilize a selective fusion module to ensure an optimal fusion status for high-quality SOD result.

3. Network Architecture

Motivation Existing state-of-the-art methods have a tendency to design deeper and more complicated network to improve the SOD performance along with expensive computation overhead. Recently, Zagoruyko et al. [44] suggest that wide residual network is far superior over their commonly used thin and very deep counterparts in terms of computational complexity and accuracy. Though the previous works have demonstrated that a wider network is effective, it has not been fully exploited in salient object detection framework. On the other hand, as shown in Figure 1-A, previous works [45, 46, 47] focused on how to effectively aggregate multi-level visual features within a single-stream network, ignoring the connection between different structure network. Saito et al. [34] show that the features extracted from different structure networks contain different information. As shown in Fig. 2 some information should be preserved (or lost) only by VGGNet, whereas some are preserved only by ResNet.
Inspired by above-mentioned, we propose to design a bi-stream network consisting of two different sub-networks, in which these sub-networks will potentially be able to provide complementary discriminative saliency clues generated by different models. Our goal is to fully take advantage of complementary information present in different kinds of features.

As shown in Fig. 3, we re-designed the basic convolutional blocks of feature extractor. Compared to the original residual block of ResNet, we designed another parallel branches to mining complementary deep features. In other words, these two parallel sub-networks will focus on different saliency clues by using independent loss function to obtain diversity features.

We utilize $\mathbf{X} = \{\mathbf{X}_i, i \in [1, 5]\}$ to denote the input maps for each convolutional block in the VGG-16 sub network, in which the $W_i$ and the $b_i$ respectively represent the predefined kernel and bias. Thus, the learning procedure of our method can be uniformly formulated as Eq. 1.

$$\mathbf{X}_{i+1} \leftarrow \text{Conv}(\mathbf{X}_i) : W_i^s \ast \mathbf{X}_i + b_i,$$  \hspace{1cm} (1)
Figure 4: The pipeline of our proposed method. Our network follows the encoder-decoder style, yet it different from previous methods, in which the encoder consists of two backbones with different structures, i.e., VGG16 and ResNet50. The input image is firstly passed through the Encoder to extract multi-scale convolutional deep features. Then, we use both the newly proposed Dense Aggregation Module (Sec. 4.2) and Selective Deep Fusion Module (Sec. 5) to make full use the multi-scale deep features which are extracted from VGG16 and ResNet50 respectively. The decoder network takes the multi-scale convolutional features as input to generate a finer saliency prediction $M^t$, which will latterly be refined by recursively using those low-level deep features in previous stage (Sec. 4.1). In each learning stage ($<N$), our method simultaneously uses the detail refinement module (to alleviate the spatial info loss) and the dense aggregation module (to avoid the learning ambiguity) to ensure the complementary status between the parallel sub-networks. When our recursive learning reaches the final stage ($=N$), we simultaneously feed the last feature layer of ResNet-50 and all side layers of VGG-16 into the selective deep fusion network to produce the final SOD results.

where $\text{Conv}(\cdot)$ denotes the convolutional operation and the superscript $s$ denotes the convolutional stride. Similarly, we represent the input maps for convolutional blocks in our ResNet-50 subnetwork as $F = \{F_i, i \in [1,5]\}$. De-convolutional layers are to progressively produce the fine-scale saliency score map $M^t$, where the superscript $t$ denotes the recursive learning stage.

Fig. 4 illustrates the overview of the proposed model, which mainly consists of three components: 1) detail refinement module; 2) dense aggregation module; and 3) selective deep fusion. All these components will cooperate our recursive multi-model deep learning, which will be respectively introduced in the following sections.
Figure 5: The illustration of the proposed modules. The sub-figure A is the detailed architecture of the Detail Refinement Module (Sec. 4.1) in the \( t \)-th stage. We resize the \( M^t \) to the same size of the \( X^t \) and concatenate them together by performing convolutional operation. Then, the combined features will be fed into the next stage, obtaining the \( M^{t+1} \) with better details. The sub-figure B shows how to convert the multi-level deep features \( X^t \) into the integrated feature maps \( X^t_i \), which will latter prepare a set of finer deep features for the next learning stage (Sec. 4.2).

4. Inter-model Deep Fusion

4.1. Detail Refinement Module

Following the widely used encoder-decoder network architecture, the proposed detail refinement module (DRM) utilizes the ResNet-50 sub-network to conduct an end-to-end saliency regression for the fine-scale saliency predictions, which will latter be applied to another parallel sub-network (VGG-16) to alleviate the spatial information loss problem, recursively.

Actually, the conventional networks usually adopt multiple convolution and pooling operations for their saliency regression, which easily degrade their performance due to the spatial information progressively vanishes in deep layers. To alleviate it, Hou et al. [14] proposed to resort short-connections to integrate inter-layer deep features to compensate the lost spatial details. However, deep features obtained by an identical single network have a tendency of homogenization, which heavily limits the
complementary status between inter-layer deep features.

To further improve it, we propose to construct dense connections between our parallel networks, see the pictorial demonstration in Fig. 5-A. Since the output of last layer of ResNet-50 can well represent the saliency details, we use it to recursively refine its parallel VGG-16 features ($X^t_i, i \in [1, 2, 3, 4, 5]$). Also, we resize the resolution of $M^t$ according to each target features $X^t_i$, and then fuse these linked deep features by using a $3 \times 3$ convolution. Here we formulate the recursively fusion procedure as Eq. 2.

$$X^t_i + 1 \leftarrow \begin{cases} \text{Conv}\{\text{Cat}(X^t_i, \uparrow (M^t))\}, & \text{if } \xi(M^t) < \xi(X^t_i) \\ \text{Conv}\{\text{Cat}(X^t_i, \downarrow (M^t))\}, & \text{if } \xi(M^t) > \xi(X^t_i) \end{cases}$$

where $\uparrow (\cdot)$ and $\downarrow (\cdot)$ denote the up-sampling and down-sampling operations respectively. $\text{Cat}(\cdot)$ denotes concatenate operation and the function $\xi(\cdot)$ returns the feature size of the given input.

So far, by using Eq. 2 we have utilized the fine-scale saliency predicted by the ResNet-50 sub-network to refine its parallel sub-network VGG-16. Meanwhile, in order to achieve an optimal inter-model complimentary status, those deep features generated by VGG-16 in turn are used to reduce the false positive regions detected by the ResNet branch. Therefore, we recursively update $M (M^t + 1 \leftarrow M^t)$ in the ResNet-50 sub-network.

4.2. Dense Aggregation Module

Previous works [29, 14, 16] have shown that a good saliency model should take full advantage of its intermediate multi-level deep features, in which high-level deep features usually concentrate on the high-level semantical information while low-level features frequently focus on the subtle details.

As we all know, the lower-level features contain many spatial details along with non-salient distractors, while, because the higher-level features focus more on those
discriminative regions, such non-salient distractors in deep features are gradually suppressed when the CNNs go deeper. Since the non-salient distractors are in lower-level features, the straightforward fusion strategies (e.g., the point-to-point style \cite{48}) will easily introduce inconsistencies/conflictions. To address this issue, we have devised a novel dense aggregation scheme, which refines each layer of the ResNet branch by integrating all-level features of the VGGNet branch, see the pictorial demonstration in Fig.5B. In this way, the distractors hidden in the low-level features will be suppressed effectively.

For each recursive learning stage (i.e., noted by superscript $t$), we first utilize $1 \times 1$ convolution to reduce the feature channel. Thus, we can easily aggregate each feature block $X^t_i$ to 32 channel feature map $\hat{X}^t_i$. Then, for each $\hat{X}^t_i$, we resize $\hat{X}^t_j$ ($j \neq i$) to be an identical size of $\hat{X}^t_i$ and aggregate all theses resized feature maps to an identical size of each ResNet-50' feature block $F^t_i$ by using $1 \times 1$ convolution, which can be formulated as Eq.3

$$X^t_i = \begin{cases} \text{Conv} \{\text{Cat}(\hat{X}^t_1, \uparrow (\hat{X}^t_2), ..., \uparrow (\hat{X}^t_5))\} & \text{if } i = 1 \\ \text{Conv} \{\text{Cat}(..., \downarrow (\hat{X}^t_{i-1}), \hat{X}^t_i, \uparrow (\hat{X}^t_{i+1}), ...)\} & \text{if } i = \{2, 3, 4\} \\ \text{Conv} \{\text{Cat}(\downarrow (\hat{X}^t_1), ..., \downarrow (\hat{X}^t_4), \hat{X}^t_5)\} & \text{if } i = 5 \end{cases} \tag{3}$$

where $\uparrow (\cdot)$ and $\downarrow (\cdot)$ respectively denote the up-sampling/down-sampling operation, $\text{Cat}(\cdot)$ denotes the concatenation operation.

In general, those computed deep feature $X^t_i$ ($i \in \{1, 2, 3, 4, 5\}$) can well represent the intermediate coarse-level saliency clues in the VGG-16 sub-network, and we recursively aggregate these features into the ResNet-50 sub-network as Eq.4

$$F^t_{i+1} \leftarrow \text{Conv}(F^t_i, X^t_i), \tag{4}$$

11
where $X^t_i$ denotes the processed $i$-th feature block in ResNet-50 at the $t$ learning stage. Once the ResNet-50’ deep features $F^t_i$ have been updated to $F^{t+1}_i$, we can achieve more finer saliency map $M^{t+1}$ accordingly, which will be used to initiate another round of recursively learning in our detail refinement module.

In summary, there are totally three major advantages regarding the proposed dense aggregation module:

1) Each coarse-level deep features generated from VGG-16 facilitate the computation of fine-scale saliency prediction of current ResNet-50 network, which ensures an effective complementary status between our parallel sub-networks;

2) The proposed dense aggregation scheme can correct the consistency of those intermediate multi-level deep features, which making the fine-scale saliency prediction of ResNet-50 network more accurate;

3) The coarse-level deep features produced by VGG-16 can be treated as attention maps to suppress the false positive regions detected by the ResNet branch.

5. Selective Deep Fusion

The conventional methods have well-investigated various hand-crafted fusion strategies (e.g., the multiplicative based ones, the additive based ones, and the maximum combination based ones) to combine saliency clues which are revealed at different spatial-levels. However, these methods are elaborately designed for certain types of image scenes, which may fail to generalize well in other image scenes. Therefore, we propose to utilize a newly designed selective deep fusion to handle the above-mentioned limitation.

Concretely, when CNNs extract multi-level features from an input image, the distractors in features are gradually suppressed as CNNs go deeper. On the one hand, from the perspective of CNNs based deep features, the non-salient distractors more tend to be existed in shallower layers, and these distractors are gradually compressed by the
Table 1: Details of our selective deep fusion module, in which the “DeC.” denotes the DeConv and the “ConvC.” denotes the Conv-Classifier. For simplicity, we have omitted the channel number of the “Output” because they have an identical channel number (i.e., 64), excepting for the last ConvC. which has 2 channels only.

<table>
<thead>
<tr>
<th>Layers</th>
<th>Conv1</th>
<th>Conv2</th>
<th>Conv3</th>
<th>Conv4</th>
<th>DeC.4</th>
<th>DeC.3</th>
<th>DeC.2</th>
<th>DeC.1</th>
<th>ConvC.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Kernel</td>
<td>33</td>
<td>3×3</td>
<td>3×3</td>
<td>3×3</td>
<td>3×3</td>
<td>3×3</td>
<td>3×3</td>
<td>3×3</td>
<td>1×1</td>
</tr>
<tr>
<td>Channel</td>
<td>64</td>
<td>64</td>
<td>64</td>
<td>64</td>
<td>64</td>
<td>64</td>
<td>64</td>
<td>64</td>
<td>2</td>
</tr>
<tr>
<td>Output</td>
<td>256×256</td>
<td>128×128</td>
<td>64×64</td>
<td>32×32</td>
<td>32×32</td>
<td>64×64</td>
<td>128×128</td>
<td>256×256</td>
<td>256×256</td>
</tr>
</tbody>
</table>

high-level saliency-related localization information. Thus, for each feature layer, we assign a selective deep fusion as a refinement module, which resorts the high-level features from other deeper branches to compress those distractors in low-level features. On the other hand, considering the previous stage $M_{t-1}$ usually contains both high-level semantic information (e.g., locations) and low-level class-agnostic saliency details (e.g., edges), we combine the fine-scale saliency clue ($M_{t-1}$) into the SDF module as well, which can be formulated as Eq. (5):

$$S^l_t = \sum_{i=l}^{5} \text{Conv} \left( TF(M_{t-1}) \otimes TF(X^l_i) \right), \quad (5)$$

where $TF(\cdot)$ is a scale transformation operation along with a 1 × 1 convolutional layer with 32 output channel number, which aims to ensure the spatial size consistent with the corresponding $F^l_t$; the operator $\otimes$ denotes the element-wise multiplication; $S^l_t$ denotes the fused feature, which intrinsically contains complementary saliency clues.

After obtaining $S^l_t$, we feed it into another branch to learn complementary saliency clues. For example, the lowest-level SDF takes all levels (i.e., 5 levels) of deep features $\{X^l_i, i = 1, 2, \ldots, 5\}$ as input, while the top-level SDF only takes a single $X^5_5$ as input. Thus, the selective fusion costs less computation overhead than point-to-point fusion strategy. Moreover, the performance increases because less distractors have been introduced into the feature integration.

We show the architecture details of the proposed selective deep fusion module in
Figure 6: Visual comparison of saliency maps. Note that GT stands for Ground truth. Apparently, it can be observed that the proposed model is able to handle diverse challenging scenes.

Tab. 1. Actually, this module mainly consists of two components: the encoder layers and the decoder layers. The encoder layers are composed of 4 convolutional layers. Each of these convolution layers is followed by a batch normalization and a ReLU activation function. Meanwhile, we assign each encoder layer with one corresponding decoder.

6. Experiments and Results

6.1. Implementation Details

The proposed method is developed on the public deep learning framework Caffe. We run our model in a quad-core PC with an i7-6700 CPU (3.4 GHz and 8 GB RAM) and an NVIDIA GeForce GTX 1080 GPU (with 8G memory). Our model is trained
<table>
<thead>
<tr>
<th>Method</th>
<th>DUT-OMRON maxFβ S-m MAE</th>
<th>DUTS-TE maxFβ S-m MAE</th>
<th>HKU-IS maxFβ S-m MAE</th>
<th>ECSSD maxFβ S-m MAE</th>
<th>PASCAL-S maxFβ S-m MAE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ours</td>
<td>0.824 0.843 0.053</td>
<td>0.887 0.895 0.041</td>
<td>0.941 0.925 0.028</td>
<td>0.945 0.928 0.031</td>
<td>0.895 0.861 0.078</td>
</tr>
<tr>
<td>RANet20 [52]</td>
<td>0.799 0.825 0.058</td>
<td>0.874 0.874 0.044</td>
<td>0.928 0.908 0.036</td>
<td>0.941 0.917 0.042</td>
<td>0.866 0.847 0.078</td>
</tr>
<tr>
<td>R’Net20 [53]</td>
<td>0.793 0.824 0.061</td>
<td>0.855 0.861 0.050</td>
<td>0.921 0.903 0.039</td>
<td>0.935 0.915 0.044</td>
<td>0.864 0.847 0.075</td>
</tr>
<tr>
<td>CPD19 [53]</td>
<td>0.797 0.825 0.056</td>
<td>0.865 0.868 0.044</td>
<td>0.925 0.906 0.034</td>
<td>0.939 0.918 0.037</td>
<td>0.884 0.828 0.089</td>
</tr>
<tr>
<td>PoolNet19 [54]</td>
<td>0.805 0.831 0.054</td>
<td>0.889 0.886 0.037</td>
<td>0.936 0.919 0.030</td>
<td>0.949 0.926 0.035</td>
<td>0.902 0.847 0.081</td>
</tr>
<tr>
<td>AFNet19 [55]</td>
<td>0.797 0.826 0.057</td>
<td>0.862 0.866 0.046</td>
<td>0.923 0.905 0.036</td>
<td>0.935 0.918 0.042</td>
<td>0.885 0.833 0.086</td>
</tr>
<tr>
<td>MWS19 [42]</td>
<td>0.718 0.756 0.109</td>
<td>0.769 0.757 0.092</td>
<td>0.856 0.818 0.084</td>
<td>0.878 0.828 0.096</td>
<td>0.814 0.753 0.151</td>
</tr>
<tr>
<td>PAGRN18 [56]</td>
<td>0.771 0.775 0.071</td>
<td>0.855 0.837 0.056</td>
<td>0.918 0.887 0.048</td>
<td>0.927 0.889 0.061</td>
<td>0.869 0.793 0.115</td>
</tr>
<tr>
<td>DGI18 [57]</td>
<td>0.774 0.806 0.062</td>
<td>0.829 0.841 0.050</td>
<td>0.911 0.895 0.036</td>
<td>0.922 0.903 0.041</td>
<td>0.881 0.828 0.082</td>
</tr>
<tr>
<td>RADF18 [16]</td>
<td>0.786 0.813 0.072</td>
<td>0.814 0.824 0.072</td>
<td>0.907 0.888 0.050</td>
<td>0.917 0.895 0.060</td>
<td>0.857 0.797 0.119</td>
</tr>
<tr>
<td>RAS18 [58]</td>
<td>0.787 0.814 0.062</td>
<td>0.831 0.838 0.060</td>
<td>0.913 0.887 0.045</td>
<td>0.921 0.893 0.056</td>
<td>0.852 0.774 0.125</td>
</tr>
<tr>
<td>SRM17 [14]</td>
<td>0.769 0.798 0.069</td>
<td>0.827 0.835 0.059</td>
<td>0.906 0.888 0.046</td>
<td>0.917 0.895 0.054</td>
<td>0.868 0.817 0.100</td>
</tr>
<tr>
<td>Amulet17 [29]</td>
<td>0.743 0.781 0.098</td>
<td>0.778 0.803 0.085</td>
<td>0.896 0.883 0.052</td>
<td>0.915 0.894 0.059</td>
<td>0.862 0.820 0.103</td>
</tr>
<tr>
<td>UCF17 [59]</td>
<td>0.735 0.758 0.132</td>
<td>0.771 0.778 0.118</td>
<td>0.886 0.866 0.074</td>
<td>0.911 0.883 0.078</td>
<td>0.851 0.808 0.128</td>
</tr>
<tr>
<td>DSS [14]</td>
<td>0.727 0.748 0.092</td>
<td>0.785 0.790 0.081</td>
<td>0.880 0.852 0.067</td>
<td>0.877 0.836 0.090</td>
<td>0.824 0.749 0.144</td>
</tr>
</tbody>
</table>

Table 2: Comparison of quantitative results including the max F-measure, S-measure and MAE on five well-known SOD benchmarks: DUT-OMRON [49], DUTS-TE [39], ECSSD [50], HKU-IS [39] and PASCAL-S [51]. The top three results are highlighted in red, green, and blue, respectively.

on the MSRA10K dataset. Then, we test our model on other datasets. Due to the limited GPU memory, we set the mini-batch size to 4. We use the stochastic gradient descent (SGD) method to train with a momentum 0.99, and the same weight decay 0.0005. Also, for our feature integration module, we use SGD with a momentum 0.9, and weight decay 0.0005. We set the learning rate as $10^{-8}$ and it reduces by a factor of 0.1 at 10k iterations. The training process of our model takes about 14 hours. During testing, the proposed model runs about 14 FPS with $256 \times 256$ resolution. We assign the number of recursive stage $N=3$ according to the qualitative results demonstrated in Fig. 9.

The performance improvements of our method are mainly brought by the newly-designed multi-model fusion scheme and we can implement the parallel sub-networks using “simple” networks. For each sub-network, the complexity/memory require-
Algorithm 1 Training Details.

Require: Training data $I = \{(I_i, y_i)\}_{i=1}^{N}$; Max epoch number $N=100$; Number of iterations: $T$;

1: for $t = 1, ..., N$ do
2:     for $i = 1, ..., T$ do
3:         Data-loading: image, gt = DataLoader(I);
4:         Predicting: pred = Model(image);
5:         Computing Loss: loss= BCELoss(pred, gt);
6:     Backpropagate loss and updating parameters: loss.backward().
7: end for
8: end for

Algorithm 1 to show the details of our training process. We set the max epoch number $N=100$ and the iteration number $T$ varies with the training data and batch size. As shown in Algorithm 1, we first first load the training set using the DataLoader(). Next, we begin to train the defined model using the Binary Cross Entropy Loss. Finally, at the end of each iteration, we will back-propagate the loss and update the network parameters. The above procedure will be repeated until reaching the max epoch number.

6.2. Datasets and Evaluation Metrics

We have evaluated our method on 5 widely used publicly available datasets, including DUT-OMRON [49], DUTS-TE [39], ECSSD [50], HKU-IS [39] and PASCAL-S [51]. DUT-OMRON contains 5,168 high-quality images. Images of this dataset have one or more salient objects with complex backgrounds. DUTS-TE has 5,019 images with high-quality pixel-wise annotations, which is selected from the currently largest SOD benchmark DUTS. ECSSD has 1,000 natural images, which contain many semantically meaningful and complex structures. As an extension of the complex scene saliency dataset, ECSSD is obtained by aggregating the images from BSD [60] and
Figure 7: Quantitative comparisons (PR curves and F-measure curves) between our method and 14 state-of-the-art methods over 5 adopted datasets, in which the left part is the PR curve and the right part is the F-measure curve. Due to the limitation of space, we only provide the quantitative results over 3 datasets here, and the remaining 3 results can be found in Fig. 8.

PASCAL VOC [61]. HKU-IS contains 4,447 images. Most of the images in this dataset have low contrast with more than one salient object. PASCAL-S contains 850 natural images with several objects, which are carefully selected from the PASCAL VOC dataset with 20 object categories and complex scenes.

We have adopted 4 commonly used standard metrics to evaluate our method, including Precision-Recall curve, F-measure, S-measure [62], and Mean Absolute Error.
6.3. Comparison with the state-of-the-art methods

We have compared our method with 14 most representative SOTA methods, including Amulet17 [29], DSS17 [14], UCF17 [59], SRM17 [15], RAS18 [58], RAD-F18 [10], PAGRN18 [56], DGRL18 [57], MWS19 [42], CPD19 [43], AFNet19 [55], PoolNet19 [54], RANet20 [52] and $R^2$Net20 [53]. For all of these methods, we use the original codes with recommended settings or the saliency maps provided by the authors. Moreover, our results are diametrically generated by model without relying on any post-processing and all the predicted saliency maps are evaluated with the same evaluation code.

Quantitative Comparisons. As a commonly used quantitative evaluation venue, we first investigate our model using the PR curves. As shown in the left of Fig. [7] and
<table>
<thead>
<tr>
<th>Method</th>
<th>Model(MB)</th>
<th>Encoder(MB)</th>
<th>Decoder(MB)</th>
<th>FLOPs(G)</th>
<th>Params(M)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ours</td>
<td>263.7</td>
<td>138.5</td>
<td>125.2</td>
<td>74.23</td>
<td>69.48</td>
</tr>
<tr>
<td>PoolNet19[54]</td>
<td>278.5</td>
<td>94.7</td>
<td>183.8</td>
<td>88.91</td>
<td>68.26</td>
</tr>
<tr>
<td>BASNet19[63]</td>
<td>348.5</td>
<td>87.3</td>
<td>261.2</td>
<td>127.32</td>
<td>87.06</td>
</tr>
<tr>
<td>DGRL18[57]</td>
<td>573</td>
<td>95.6</td>
<td>477.4</td>
<td>215.62</td>
<td>146.37</td>
</tr>
</tbody>
</table>

Table 3: The number of model size, FLOPs and parameters comparisons of our method with 3 state-of-the-art models.

Fig. 8, our model can consistently outperform the state-of-the-art models on all tested benchmark datasets. Specifically, the proposed model outperforms other models on DUT-OMRON datasets. Meanwhile, our model also is evaluated by F-measure curves as shown in the right of Fig. 7 and Fig. 8 which also demonstrates the superiority of our method. The detailed F-measure, MAE values are provided in Table 2 in which our method also performs favorably against other state-of-the-art approaches. As for the DUT-OMRON dataset, our model achieves 82.4% in max F-measure and 0.053 in MAE while the second best (PoolNet19) achieves 80.5% in max F-measure and 0.054% in MAE. Also, similar tendencies can be found in the HKU-IS dataset, which is one of the most challenge datasets. Compared to the recent published RANet20, our model increases 1.3% in max F-measure and decreases 8% in MAE.

**Qualitative Comparisons.** We demonstrate the qualitative comparisons in Fig. 6. The proposed method not only detects the salient objects accurately and completely, but preserves subtle details also. Specifically, the proposed model can adapt to various scenarios as well, including the small object case (row 3), the object occlusion case (raw 6), the complex background case (row 7), and the low contrast case (row 9). Moreover, our method can consistently highlight the foreground regions with sharp object boundaries.

**Running Time and Model Complexity Comparisons.** Table 5 shows the running time comparisons, this evaluation was conducted on the same machine with an i7-6700 CPU and a GTX 1080 GPU, in which our model achieves 14 FPS. Besides, Table 3 shows the model complexity comparisons, in which we may easily notice that most
Table 4: Ablation study of our model on DUT-OMRON [49], DUTS-TE [39], ECSSD [50], and HKU-IS [39]. We change one component at a time, to assess individual contributions. VGGNet and ResNet are used as the backbone. DRM is Details Refinement Module, DAM denotes Dense Aggregation Module and SDF stand for Selective Deep Fusion Module.

6.4. Component Evaluation

To validate the effectiveness of our method, we have evaluated several key components of the proposed model on the DUT-OMRON, DUTS-TE, ECSSD and HKU-IS dataset. We start with two single-stream networks and progressively extend it with our newly designed modules, including the parallel backbones, the detail refinement module, the dense aggregation module and the selective deep fusion module.

As shown in Table 4, our newly designed parallel architecture equipped with detail refinement module only (see the 3rd row) can achieve much better performance than the single sub-network (the 1st row and 2nd row). Meanwhile, the overall performance of the proposed parallel architecture with dense aggregation module can get the overall performance further improved, see the 4th row in Table 4. Specially, we notice that the proposed selective deep fusion module obtains a significant performance improvement,
Figure 9: Qualitative illustration of our recursive learning scheme, where $t$ denotes the saliency maps obtained at different learning stages.

see the 5th row. All these results have demonstrated the effectiveness of the proposed method.

<table>
<thead>
<tr>
<th>Method</th>
<th>Our</th>
<th>CPD19</th>
<th>AFNet19</th>
<th>DGRL18</th>
<th>RADF18</th>
<th>SRM17</th>
<th>Amulet17</th>
<th>UCF17</th>
<th>DSS17</th>
<th>RFCN18</th>
</tr>
</thead>
<tbody>
<tr>
<td>Time(s)</td>
<td>0.073</td>
<td>0.063</td>
<td>0.062</td>
<td>0.150</td>
<td>0.154</td>
<td>0.070</td>
<td>0.093</td>
<td>0.134</td>
<td>0.201</td>
<td>4.72</td>
</tr>
</tbody>
</table>

Table 5: Runtime comparison (GPU time) with previous deep learning based saliency models.

6.5. Recursive Learning Validation

As described in Sec. 4, our method is trained in a recursive manner. To validate the effectiveness of our stage-wise recursive learning scheme, we perform a detailed comparison of the proposed model at different recursive learning stages using max F-measure, S-measure and MAE scores. As shown in the last three rows of Table 4, the overall performance of our method becomes better as the stage-wise recursive learning continues, and the corresponding qualitative demonstrations can be found in Fig. 9.

6.6. Ablation Study for Different Scale Input

Considering that the M cells contribute to the low-level transient processing while the P cells contribute more to the high-level recognition tasks, we also investigate the effectiveness adopting different input scales for different networks. As shown in Table 6, we have newly conducted a series of quantitative experiments to validate it.
Table 6: Ablation study for different scale input. For example, \( S_{\text{ResNet}} = 1, S_{\text{VGG}} = 0 \)\( \text{.5} \) denotes that the ResNet branch takes the whole image as input while the VGG branch reduces the image size by half.

Instead of being beneficial to the SOD task, the experimental results show that different input scale for different network decreases the overall performances. The main reason could be that, as demonstrated by the previous work \([64]\), the low-resolution image usually shows unimpressive representation, which is mainly induced by its limited information towards the SOD task (e.g., blurry object boundaries). In the proposed network, the VGG branch was designed to play a role of coarse localization, and thus it can ensure good performance (about 0.5% degeneration in its performance) when assigning a small scale to its features. Meanwhile, compared with the VGG sub-network, the ResNet branch that is designed for tiny saliency details has a significant performance degeneration (about 1%) due to a low-resolution input.

6.7. Limitations

Compared with previous works, our method can capture more powerful saliency clues from different saliency perspective while avoiding the obstinate feature confictions by using the proposed multi-model fusion scheme. In the clutter background case, our method can well suppress those non-salient regions and preserves subtle salient details, which is proved by the increased precision rate and F-measure score in Fig. 7 and Fig. 8.
Nonetheless, we have noticed a slight decrease regarding the average recall rate, which is mainly induced by an unbalanced bias in our multi-model fusion when computing those complementary deep features. Another limitation of our model is the computational overhead for the stage-wise training. In the future, we plan to explore a more efficient fusion approach by using the off-the-shelf model compression techniques to alleviate the computational burden.

7. Conclusion

In this paper, we proposed a novel multi-model fusion scheme, in which two parallel sub-networks are coordinated to learn complementary deep features recursively. The key rationale of our proposed method is to take full advantage of the complementary features encoded in different sub-networks, revealing saliency clues from different perspectives. To achieve this goal, we newly design three components: 1) Detail Refinement Module, 2) Dense Aggregation Module, and 3) Selective Deep Fusion Module. Specifically, we propose a detail refinement module to recursively compensate for the lost spatial details, and the dense aggregation module is designed to make full use of multi-level deep features. Meanwhile, we propose a selective deep fusion module to effectively fuse complementary information encoded in different sub-branches. Experiments show that the proposed model outperforms existing state-of-the-art algorithms on five benchmark datasets.
References


[33] T.-Y. Lin, A. RoyChowdhury, and S. Maji, “Bilinear cnn models for fine-grained


[41] X. Li, L. Zhao, L. Wei, M. Yang, F. Wu, Y. Zhuang, H. Ling, and J. Wang, “Deepsaliency: Multi-task deep neural network model for salient object detec-


