# A Simple Saliency Detection Approach via Automatic Top-Down Feature Fusion

Yu Qiu[a], Yun Liu[b], Hui Yang[b], Jing Xu[a,*]

[a]College of Artificial Intelligence, Nankai University, Tianjin, P.R.China
[b]College of Computer Science, Nankai University, Tianjin, P.R.China

**Abstract**

It is widely accepted that the top sides of convolutional neural networks (CNNs) convey high-level semantic features, and the bottom sides contain low-level details. Therefore, most of recent salient object detection methods aim at designing effective fusion strategies for side-output features. Although significant progress has been achieved in this direction, the network architectures become more and more complex, which will make the future improvement difficult and heavily engineered. Moreover, the manually designed fusion strategies would be sub-optimal due to the large search space of possible solutions. To address above problems, we propose an Automatic Top-Down Fusion (ATDF) method, in which the global information at the top sides are flowed into bottom sides to guide the learning of low layers. We design a novel *valve* module and add it at each side to control the coarse semantic information flowed into a specific bottom side. Through these *valve* modules, each bottom side at the top-down pathway is expected to receive necessary top information. We also design a *generator* to improve the prediction capability of fused deep features for saliency detection. We perform extensive experiments to demonstrate that ATDF is simple yet effective and thus opens a new path for saliency detection.

*Keywords:* Salient object detection, saliency detection, multi-level feature fusion.

## 1. Introduction

Salient object detection aims at detecting the most conspicuous and eye-attracting regions in an image [1, 2]. It can be used as a pre-processing step for many computer vision tasks, including visual tracking [3], image retrieval [4], and weakly supervised learning [5]. Although many methods have been proposed to push forward the state of the arts [6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17], it is still a challenging problem to accurately detect salient objects in natural images.

Conventional saliency detection methods [18, 2, 19, 20] usually design hand-crafted low-level features and heuristic priors, which are difficult to represent semantic objects and scenes. Recently, convolutional neural networks (CNNs), which can intelligently extract multi-scale and multi-level features directly from raw images, have significantly promote the development of saliency detection [7, 21, 22, 12, 14, 23, 24, 25]. Generally, the top layers of CNNs contain semantic information that can better locate the coarse places of salient objects, while the bottom layers contain fine details such as object boundaries [14, 26]. Both the semantic information and fine details are important to saliency detection, so most existing deep learning based methods mainly focus on how to integrate the multi-level side-output features using various carefully-designed connections [14, 13, 10, 17, 23, 15, 27, 28, 29]. For examples, Hou *et al.* [14] fused several predictions, each of which was obtained by fusing some specific side outputs with short connections. However, their integration was not robust due to their manually selected connection combinations from lots of possible solutions. Moreover, Zhang *et al.* [10] first integrated multi-level feature maps into multiple resolutions and then adaptively learned to combine these feature maps at each resolution. However, it is difficult for the manually designed fusion strategies to take full advantage of the complementary top semantic information and bottom fine details. Intuitively, not all multi-level features are helpful for saliency detection, and for different images, deep features from multiple sides may have different importance. Therefore, simply fusing multi-level features without selection for all input images is suboptimal for multi-scale learning. It is essential to learn a selection rate, with which each side in the fusing pathway can receive necessary information from the preceding side and filter out redundant information.

Based on above observations, we present an Automatic Top-Down Fusion (ATDF) model which is able to automatically flow the global information at the top sides of CNNs into bottom sides. We propose a *valve* module to perform automatic selection for the information flow. Each side adds such a *valve* module to receive the specifically useful and instructive global information to guide its learning. Hence the top semantic information can guide the learning of bottom layers, and the bottom side outputs can accurately predict both the location and details of salient objects. Moreover, we design a simple yet effective *generator* module that can further improve the prediction capability of aggregated multi-level features for saliency prediction. By this way, ATDF is able to take good advantage of the automatic fusion of multi-level features extracted from CNNs. Experiments on six challenging datasets demonstrate

*Corresponding author.
*Email address:* xujing@nankai.edu.cn (Jing Xu)

that our method favorably outperforms existing state-of-the-art methods in terms of popular evaluation metrics. Compared with the preliminary conference version [30], we provide a more clear presentation, conduct more experimental evaluations and analyses, and add sufficient ablation studies for the better understanding of the proposed method. The main contributions of this paper include:

- We propose a simple yet effective *valve* module to perform automatic information selection for each side in the top-down flow pathway of CNNs.
- We adopt the *valve* module and another *generator* module to build an elegant network[1] which is demonstrated to achieve state-of-the-art performance for saliency detection through extensive experiments.

## 2. Related Work

Traditional saliency detection methods usually extract various hand-crafted low-level features and then apply classifiers to classify these features [31, 32, 33, 34, 35]. Most of them utilize heuristic saliency priors, such as center prior [36, 18], color contrast[1, 2], and background prior [37, 38, 39]. For examples, Han *et al.* [40] integrated image features including color, luminance and edge orientation and then applied the Markov random field to segment salient objects. Nevertheless, it is difficult for these hand-craft features and priors to capture high-level semantic information which are important to accurately locate objects [41, 42, 43].

Recently, CNNs have demonstrated their powerful capability in many computer vision tasks including saliency detection [44, 45, 46, 47, 48, 25, 49, 50, 51]. For example, Li *et al.* [52] used the multi-level deep features extracted from a CNN to compute the saliency value for each super-pixel [53]. Wang *et al.* [6] adopted two CNNs to integrate both local estimation and global search for saliency detection. However, these models use image patches as inputs and thus miss the high-level spatial information of the whole image. More recently, *CNN-based image-to-image saliency detection* [17, 54, 55, 24, 23, 15, 56, 57, 10, 58, 13, 59, 60, 14, 9, 12, 21, 61] has dominated this field by viewing saliency detection as a pixel-wise regression task and performing image-to-image predictions. Hence we mainly review CNN-based image-to-image saliency detection in the following.

A hot topic in this direction is to use various manually-designed strategies for multi-level and multi-scale feature fusion. Wang *et al.* [8] fused deep features from upper layers to lower layers to construct a saliency detector. Liu *et al.* [21] proposed a two-stage network which first detects salient objects coarsely, and then hierarchically and progressively refines the details of saliency maps. Hou *et al.* [14] integrated multi-scale side-outputs using short connections to takes good advantage of multi-level and multi-scale features. Luo *et al.* [12] made use of the local and global information through a multi-resolution grid structure. Zhang *et al.* [10] directly concatenated feature maps from both the top sides and bottom sides of CNNs. Zhang *et al.* [23] introduced a bi-directional message passing module, where message passing rate can be controlled by a gate function.

Instead of fusing mutli-level and mutli-scale features in a single information flow, there are also some flexible designs. Wu *et al.* [62] proposed a cascaded partial decoder framework for saliency detection, which discards features of shallower layers and utilizes the generated saliency map to refine the features of backbone network. Wang *et al.* [56] presented a recurrent network to locate salient objects and then refine them with local contextual information. Liu *et al.* [25] explored the effect of pooling in CNNs. They designed a global guidance module which can provide layers at different feature levels the location information of potential salient objects. They also design a feature aggregation module to make the high-level semantic information well fused with the low-level features. The above methods mainly focus on how to integrate the multi-level and multi-scale side-output features using carefully designed connections. However, not all side-output features are equally important for saliency detection, and the manually designed selection strategies would be sub-optimal.

Some other studies focus on explore the effect of attention mechanisms for saliency detection. For example, Liu *et al.* [15] proposed a pixel-wise contextual attention network to learn to selectively attend to informative context locations for each pixel. Meanwhile, Zhang *et al.* [54] proposed an attention-guided U-shape network which generates attentive features using channel-wise and spatial attention mechanisms to improve the effectiveness of feature integration process. Zhao *et al.* [63] proposed a pyramid feature attention network to learn more detailed information and contextual features together. However, most existing methods integrate multi-level features without distinction, which would lead to information redundancy.

Instead of exploring new connection strategies, we investigate how to automatically flow the global information at the top sides of CNNs into bottom sides to guide the learning of low layers. The proposed *valve* module can make our model take good advantage of the automatic fusion of multi-level features extracted from CNNs.

## 3. Method

### 3.1. Overall Framework

In this paper, we propose an Automatic Top-Down Fusion (ATDF) model to address the problems of sub-optimal multi-level side-output feature fusion in manually designed fusion strategies. As shown in Fig. 1 (a), the main architecture of ATDF is beneficial from the encoder-decoder networks. Our model uses VGG16 net [64] as the backbone model by making two modifications: (i) We remove the final fully connected layers to serve for image-to-image translation; (ii) We remain the last pooling layer and meanwhile additionally introduce a max pooling layer (the $\mathcal{S}_6^e$ block in the right of the encoder path in Fig. 1 (a)) to enlarge the receptive field and obtain global information.

[1] Code is available at `https://github.com/yun-liu/ATDF`.

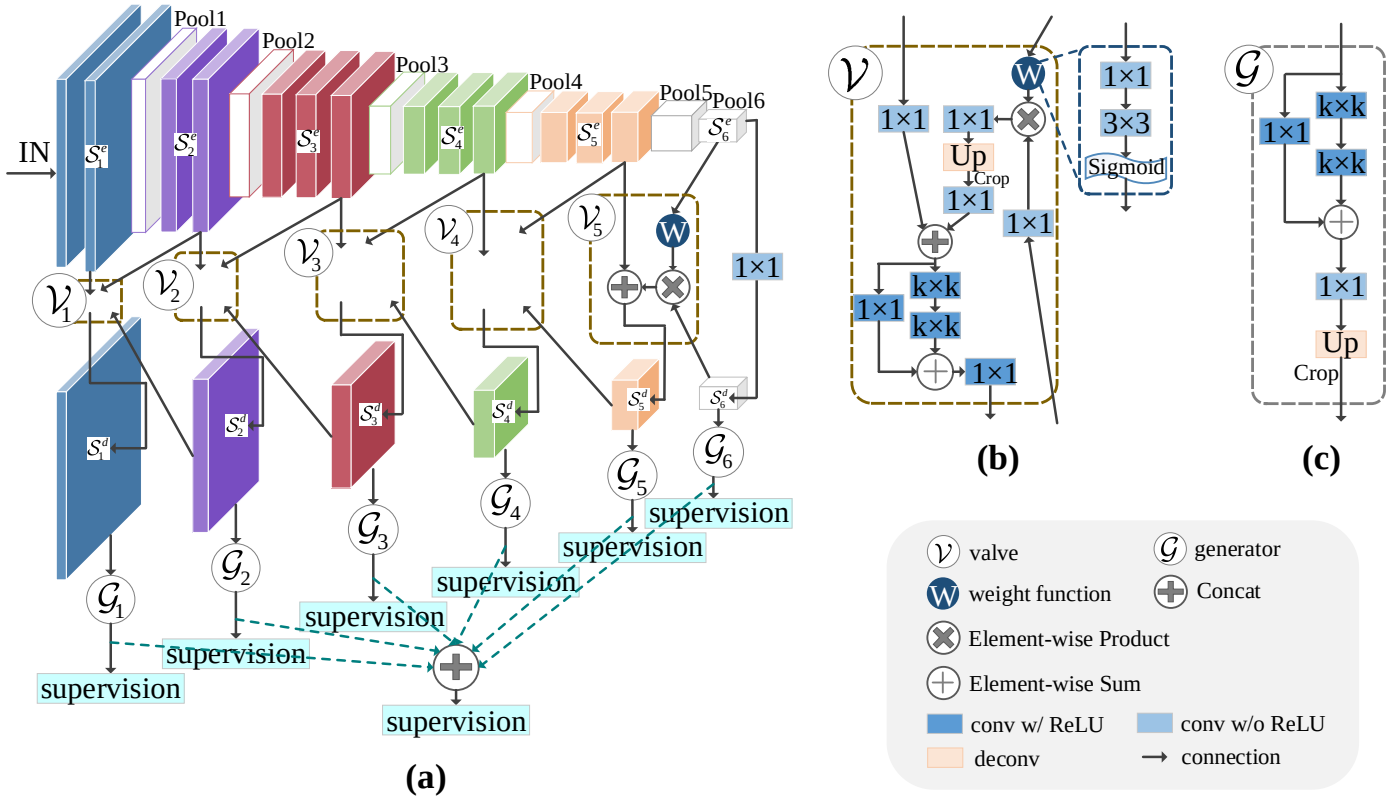


Figure 1: (a) Network architecture of the proposed saliency detector with the VGG16 backbone. (b) Illustration of the *valve* module. (c) Illustration of the *generator* module. We add a series of side losses (cross-entropy losses) after each *generator* to perform deep supervision. Note that the kernel sizes $k \times k$ for convolution layers in the *valve* and *generator* modules will be presented in Section 3.2 and Section 3.3, respectively.

The resulting VGG16 net (encoder) is composed of six blocks, which are denoted as $\{\mathcal{S}_1^e, \mathcal{S}_2^e, \mathcal{S}_3^e, \mathcal{S}_4^e, \mathcal{S}_5^e, \mathcal{S}_6^e\}$ from bottom to top, respectively. Note that $\mathcal{S}_6^e$ represents the added pooling layer. The receptive fields of $\mathcal{S}_1^e$ to $\mathcal{S}_6^e$ are gradually enlarged to capture more high-level information, so that $\mathcal{S}_6^e$ is considered to contain the most rich global semantic information. Symmetrically, the decoder is in a top-down view, which can be defined as the automatic top-down fusion path, containing six blocks, *i.e.*, $\{\mathcal{S}_6^d, \mathcal{S}_5^d, \mathcal{S}_4^d, \mathcal{S}_3^d, \mathcal{S}_2^d, \mathcal{S}_1^d\}$. In the contracting path (encoder), the resolution of the feature map in each block is the half of the preceding one, and contrarily in the expanding path (decoder). Following [14, 65], the side output of each block means the connection from the last layer of this block. In the expanding path, each feature map $\mathcal{S}_i^d$ ($i \in \{1, 2, 3, 4, 5\}$) originates from the preceding one $\mathcal{S}_{i+1}^d$ and the $\mathcal{S}_{i+1}^e$ and $\mathcal{S}_i^e$ in the contracting path, controlled by a *valve* module. Specially, $\mathcal{S}_6^d$ is the first side of expanding path and only originates from $\mathcal{S}_6^e$ in the contracting path using a $1 \times 1$ convolution operation.

The *valve* module can adaptively determine the flow of the useful high-level information from top sides to bottom sides, which will be further presented in Section 3.2. Specifically, we add a *generator* module to generate saliency maps at each side, which will be introduced in Section 3.3. We adopt deep supervision [14] at the intermediate sides to perform predictions. A $1 \times 1$ convolution is performed on the concatenation of the predicted saliency maps from the *generator* modules at all sides to obtain the final prediction which is also supervised by the ground truth in the training. The proposed ATDF is trained end-to-end using the cross-entropy loss between the predicted saliency map and the ground truth.

### 3.2. The Valve Module

Since the fusion effectiveness of multi-level features determines the performance of saliency detection, how to exactly integrate multi-scale side-output features is the main point in this paper. Different from existing methods [14, 10], our ATDF is able to automatically flow necessary semantic information from the top sides to bottom sides, guiding the learning of lower layers. To achieve this goal, we design a *valve* module, which make the high-level information flowing in the expanding path is non-redundant without manual intervention. By this way, the integrated features are complementary and only target the saliency detection in various scenes.

The architecture of the *valve* module is illustrated in Fig. 1 (b). The ATDF has five *valve* modules which are denoted as $\{\mathcal{V}_5, \mathcal{V}_4, \mathcal{V}_3, \mathcal{V}_2, \mathcal{V}_1\}$ from top to bottom, respectively. Each *valve* $\mathcal{V}_i$ ($i \in \{1, 2, 3, 4, 5\}$) has three inputs, *i.e.*, $\mathcal{S}_{i+1}^d$ in the expanding path, $\mathcal{S}_{i+1}^e$ and $\mathcal{S}_i^e$ in the contracting path. After a series of operations for the inputs, the next block $\mathcal{S}_i^d$ is obtained from the valve $\mathcal{V}_i$.

The *valve* module mainly contains three operations. For the *valve* $\mathcal{V}_i$ ($i \in \{1, 2, 3, 4, 5\}$), the input $\mathcal{S}_{i+1}^d$ is first connected to a $1 \times 1$ convolution layer, followed by a deconvolution layer with the fixed *bilinear* kernel for upsampling [65, 66]. After this

step, the $\mathcal{S}_{i+1}^d$ is upsampled by a factor of 2. Then, we adopt a standard crop operation in deep learning to cut off the offset of the feature map generated by the deconvolution, so that the size of $\mathcal{S}_{i+1}^d$ is aligned with that of $\mathcal{S}_i^e$. We denote the result of this step as $\mathcal{V}_1^i$ and formulate this step as

$$\mathcal{V}_i^1 = Crop(DeConv(Conv(\mathcal{S}_{i+1}^d))). \tag{1}$$

Secondly, we simply perform a linear transformation for the input $\mathcal{S}_i^e$ using $1 \times 1$ convolution without non-linearization. Then, we concatenate the result of above transformation and $\mathcal{V}_i^1$ to obtain $\mathcal{V}_i^2$. For a clear presentation, we can formulate this step as

$$\mathcal{V}_i^2 = Concat(Conv(\mathcal{S}_i^e), Conv(\mathcal{V}_i^1)). \tag{2}$$

Finally, a residual block with two sequential convolution layers is connected to the feature map $\mathcal{V}_i^2$. These two sequential convolution layers are with kernel size 5×5 for $\{\mathcal{V}_5, \mathcal{V}_4\}$ and kernel size $3 \times 3$ for $\{\mathcal{V}_3, \mathcal{V}_2, \mathcal{V}_1\}$. The numbers of output channels are 256, 256, 128, 128, and 64 from $\mathcal{V}_5$ to $\mathcal{V}_1$, respectively. Moreover, the element-wise addition is performed on the feature map obtained after these two convolution layers and $\mathcal{V}_i^2$ to form a residual structure. Then, we adopt another $1 \times 1$ convolution to fuse the feature map after the element-wise addition operation. We provide detailed ablation studies for these parameters in Section 5. We formulate this step as

$$\mathcal{V}_i = \sigma(Conv(\sigma(Conv(\sigma(Conv(\mathcal{V}_i^2)))) + \mathcal{V}_i^2)) \tag{3}$$

where $\sigma(\cdot)$ denotes non-linearization, *i.e.*, ReLU [67] here.

Moreover, we design a weight function to control the proportion of the top information ($\mathcal{S}_{i+1}^d$) flowing into the lower side, in which the information flowing rate is in the range of $[0, 1]$. The weight function is defined as a convolutional layer with sigmoid activation. The formula of the weight function is $\mathcal{W}_i = Sigmoid(Conv(Conv(\mathcal{S}_{i+1}^e)))$, where the kernel sizes of the two sequential convolution layers are $1 \times 1$ and $3 \times 3$, respectively. Next, we make the aforementioned first step (Equ. (1)) better by adding this weight function. After the $1 \times 1$ convolution that is applied on $\mathcal{S}_{i+1}^d$, we perform the element-wise multiplication on the result of above convolution and $\mathcal{W}_i$, which can be expressed as $\mathcal{W}_i \otimes Conv(\mathcal{S}_{i+1}^d)$. As before, we then connect a $1 \times 1$ convolution layer, followed by a deconvolution layer with the fixed *bilinear* kernel. We reformulate $\mathcal{V}_i^1$ and $\mathcal{V}_i^2$ with $\mathcal{V}_i^{1\prime}$ and $\mathcal{V}_i^{2\prime}$ as

$$\mathcal{V}_i^{1\prime} = Crop(DeConv(Conv(\mathcal{W}_i \otimes Conv(\mathcal{S}_{i+1}^d)))) \tag{4}$$

and

$$\mathcal{V}_i^{2\prime} = Catcat(Conv(\mathcal{S}_i^e), Conv(\mathcal{V}_i^{1\prime})). \tag{5}$$

The $\otimes$ is element-wise multiplication. The final output of valve module $\mathcal{V}_i{}'$, *i.e.*, the next block $\mathcal{S}_i^d$, can be expressed as

$$\mathcal{S}_i^d = \mathcal{V}_i{}' = \sigma(Conv(\sigma(Conv(\sigma(Conv(\mathcal{V}_i^{2\prime})))) + \mathcal{V}_i^{2\prime})). \tag{6}$$

The optimization of the valve is from top to bottom, so that the learning of weight function is also from top to bottom.

Now, we interpret the idea behind the design of such *valve* module. The $\mathcal{S}_6^d$ to $\mathcal{S}_1^d$ are gradually generated along the information flow of the expanding path (decoder), and the generation of lower layers relies on the higher layers. $\mathcal{S}_i^d$ ($i \in \{1, 2, 3, 4, 5\}$) has two inputs of $\mathcal{S}_i^e$ and $\mathcal{S}_{i+1}^d$. $\mathcal{S}_i^e$ is the corresponding layer in the encoder, which is the main contribution to $\mathcal{S}_i^d$. While the $\mathcal{S}_{i+1}^d$ contains higher-level information which should be flowed down to introduce more high-level semantic information for $\mathcal{S}_i^d$. However, lots of noises, *i.e.*, the inessential information for $\mathcal{S}_i^d$, exist in $\mathcal{S}_{i+1}^d$, which should be filtered. The obvious candidate filter which can constraint the information flowed from $\mathcal{S}_{i+1}^d$ is $\mathcal{S}_{i+1}^e$. As for the reason, $\mathcal{S}_{i+1}^e$ is the only block that is most related to both the $\mathcal{S}_i^e$ and $\mathcal{S}_{i+1}^d$ which can determine the fine details and coarse places of salient objects, respectively. On one hand, $\mathcal{S}_{i+1}^e$ is directly generated from $\mathcal{S}_i^e$ in the encoder, so it can learn what $\mathcal{S}_i^e$ "wants". On the other hand, $\mathcal{S}_{i+1}^e$ is the main contribution of $\mathcal{S}_{i+1}^d$, so it "knows" what $\mathcal{S}_{i+1}^d$ "has". Therefore, $\mathcal{S}_{i+1}^e$ has the ability to extract the complementary information from $\mathcal{S}_{i+1}^d$ for $\mathcal{S}_i^e$.

### *3.3. The Generator Module*

The semantic information is learned in the top sides and gradually flows to the bottom sides along the expanding path. The ATDF should be optimized from top to down if we want to achieve automatic top-down information fusion. Therefore, instead of imposing supervision at the final layer of the expanding path, we use the scheme of adding deep supervision for all side outputs. As mentioned above, the *valve* module can automatically control the flow of global information from top sides to bottom sides, so we can obtain hybrid hierarchical feature maps. To further improve the capability of aggregated hierarchical information for saliency prediction, we adopt six *generator* modules $\{\mathcal{G}_6, \mathcal{G}_5, \mathcal{G}_4, \mathcal{G}_3, \mathcal{G}_2, \mathcal{G}_1\}$ after six sides, *i.e.*, $\{\mathcal{S}_6^d, \mathcal{S}_5^d, \mathcal{S}_4^d, \mathcal{S}_3^d, \mathcal{S}_2^d, \mathcal{S}_1^d\}$, respectively. These *generator* modules can further process the feature maps to better fuse the high-level semantic information and low-level spatial details for accurate saliency prediction at specific scales.

The architecture of the *generator* is displayed in Fig. 1 (c). The output $\mathcal{S}_i^d$ of each side in the expanding path is first connected to a residual convolution block with two sequential convolution layers, which are with kernel size of $7 \times 7$ for $\mathcal{S}_6^d$, $5 \times 5$ for $\{\mathcal{S}_5^d, \mathcal{S}_4^d\}$, and kernel size $3 \times 3$ for $\{\mathcal{S}_3^d, \mathcal{S}_2^d, \mathcal{S}_1^d\}$. The numbers of channels for them are 512, 256, 256, 128, 128 and 64, respectively. The element-wise addition is then performed on the feature map after above convolution layers and $\mathcal{S}_i^d$ to build the residual path. We add a $1 \times 1$ convolution layer without non-linearization to change the channel number to 1 for each side. This 1-channel feature map is the saliency prediction at each side. We upsample the saliency map to the same size as original image using a deconvolution layer with the fixed *bilinear* kernel, followed by a crop layer to remove the offset. For better description, we summarize the operations of the *generator* module as the following formula:

$$\begin{aligned} \mathcal{F}_i &= \sigma(Conv(\sigma(Conv(\mathcal{S}_i^d)))) + Conv(\mathcal{S}_i^d), \\ \mathcal{G}_i &= Crop(DeConv(Conv(\mathcal{F}_i))), \end{aligned} \tag{7}$$

in which $Crop$ is the standard crop operation to cut off the offset of the deconvolution layer as in Equ. (1).

With the *generator* modules, we can obtain a saliency prediction map at each side, after which we add the deep supervision. A series of side losses (cross-entropy losses) are added after the *generator* to perform deep supervision, which leads to obvious performance gain for saliency detection [14]. Finally, we concatenate the predicted maps and fuse the concatenated map with a $1 \times 1$ convolution to obtain the final prediction. In the training, the final prediction is also supervised by the ground-truth saliency maps.

## 4. EXPERIMENTS

In this section, we first introduce the experiment setups, including the implementation details, the used datasets and the evaluation criteria. Then we compare our proposed salient object detector with 16 recent state-of-the-art models.

### *4.1. Experimental Setup*

**Implementation Details.** The proposed network is implemented using the Caffe [68] framework. The parameters of the layers contained in VGG16 [64] and ResNet [69] are initialized by the pretrained ImageNet model. The weights of other convolution layers are initialized from the zero-mean Gaussian distribution with standard deviation 0.01. Biases are initialized to 0. We use deconvolution layers with bilinear in-terpolation kernels to achieve the upsampling operations, which are frozen in the training process. Since the deconvolution layers do not need training, we exclude them when computing the number of parameters. We optimize the proposed network using SGD. The learning rate policy is *poly*, in which the current learning rate equals the base one multiplying $(1 - curr\_iter/max\_iter)^{power}$. Here, the hyper parameters *power* is set to 0.9, and we run 20000 SGD iterations (*max_iter*) in total. The initial learning rate is set to 2.5e-8. We set the momentum and weight decay to 0.9 and 0.0005, respectively. We use the unweighted *sigmoid* cross-entropy loss for salient object detection. All the experiments are performed on a TITAN Xp GPU.

**Datasets.** To fine-tune our model, we follow recent studies [15, 13] to utilize the DUTS [70] training dataset that consists of 10553 images with pixel-wise saliency annotation. We extensively evaluate our method on the DUTS test set and other five popular datasets including SOD [71], HKU-IS [52], ECSSD [72], DUT-OMRON [37], and THUR15K [73]. These six test datasets contain 5019, 300, 4447, 1000, 5168, and 6232 natural complex images with corresponding human labeling.

**Evaluation Criteria.** In this paper, we adopt four evaluation metrics to evaluate the performance of saliency detection models: the max $F$-measure score $F_\beta$, mean absolute error (MAE), weighted $F$-measure score $F_\beta^\omega$, and average $F$-measure score $F_\beta^m$. Other than numerical results, we also provide the precision *vs.* recall curves (PR curves), $F$-measure *vs.* threshold curves (FT curves), and histograms for precision, recall, and $F$-measure scores.

Specifically, the predicted saliency map of numerical continuity can be converted into a binary map using varying thresholds in the range of [0, 1]. When comparing the binarized saliency maps (corresponding to different thresholds) with the ground truth, we can get a series of precision-recall value pairs for each image. The performance of a dataset is the average over all images. The F-measure, *i.e.*, $F_\beta$, is an overall performance evaluation computed by the weighted harmonic of precision and recall. The formula of $F$-measure score is

$$F_\beta = \frac{(1+\beta^2) \times Precision \times Recall}{\beta^2 \times Precision + Recall}, \tag{8}$$

in which $\beta^2$ is typically set to 0.3 to emphasize more on precision as suggested by [37, 10, 14, 23, 56, 15, 54, 25, 63]. Here, we calculate the $F_\beta$ values under all thresholds and report the best one at the optimal threshold.

Given a saliency map $P$ and the corresponding ground truth $G$ that are normalized to [0, 1], MAE can be calculated as

$$MAE = \frac{1}{H \times W} \sum_{i=1}^{H} \sum_{j=1}^{W} |P(i,j) - G(i,j)| \tag{9}$$

where $H$ and $W$ are image height and width, respectively. $P(i, j)$ and $G(i, j)$ denote the saliency score at location $(i, j)$.

Moreover, Margolin *et al.* [74] proposed a new measure, called weighted $F$-measure score $F_\beta^\omega$. This new measure can amend some flaws, *e.g.*, interpolation flaw, dependency flaw and equal-importance flaw, caused by traditional saliency evaluation metrics.

Since the maximum $F$-measure ($F_\beta$) are computed based on a optimal threshold, we consider a series of $F$-measure scores using different thresholds and adopt the average $F$-measure across all thresholds for evaluation:

$$F_\beta^m = \frac{1}{|T|} \sum_{t \in T} F[t] \tag{10}$$

where $T$ denotes all possible thresholds and $F[t]$ represents the $F$-measure at the threshold $t$.

In addition, the PR curves are plotted using the precision and recall values at different thresholds to summarize the trade-off between precision and recall. FT curves illustrate the F-measure scores at various thresholds. The histograms for precision, recall, and $F$-measure show a visual comparison of the overall performance of various methods.

### *4.2. Performance Comparison*

We compare our proposed salient object detector with 16 recent state-of-the-art models, including MDF [52], LEGS [6], ELD [7], RFCN [8], DCL [9], DHS [21], Amulet [10], UCF [11], NLDF [12], PiCA [15], C2S [16], RAS [17], DSS [14], SRM [13], BRN [56], and CPD-R [62]. For fair comparisons, all these salient object detectors are tested using their released code and pretrained models provided by the authors with default settings. We do not report the results of MDF [52] on the HKU-IS dataset [52], because MDF includes HKU-IS data as

| | Methods | SOD | | HKU-IS | | ECSSD | | DUT-OMRON | | THUR15K | | DUTS-test | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | $F_\beta$ | MAE | $F_\beta$ | MAE | $F_\beta$ | MAE | $F_\beta$ | MAE | $F_\beta$ | MAE | $F_\beta$ | MAE |
| VGG16 | MDF [52] | 0.764 | 0.182 | - | - | 0.807 | 0.138 | 0.680 | 0.115 | 0.669 | 0.128 | 0.707 | 0.114 |
| | LEGS [6] | 0.733 | 0.194 | 0.766 | 0.119 | 0.830 | 0.118 | 0.668 | 0.134 | 0.663 | 0.126 | 0.652 | 0.137 |
| | ELD [7] | 0.758 | 0.154 | 0.837 | 0.074 | 0.866 | 0.081 | 0.700 | 0.092 | 0.726 | 0.095 | 0.727 | 0.092 |
| | RFCN [8] | 0.802 | 0.161 | 0.892 | 0.080 | 0.896 | 0.097 | 0.738 | 0.095 | 0.754 | 0.100 | 0.782 | 0.089 |
| | DCL [9] | 0.831 | 0.131 | 0.892 | 0.063 | 0.895 | 0.080 | 0.733 | 0.095 | 0.747 | 0.096 | 0.785 | 0.082 |
| | DHS [21] | 0.822 | 0.128 | 0.889 | 0.053 | 0.903 | 0.062 | - | - | 0.752 | 0.082 | 0.807 | 0.066 |
| | Amulet [10] | 0.795 | 0.144 | 0.897 | 0.051 | 0.913 | 0.061 | 0.743 | 0.098 | 0.755 | 0.094 | 0.778 | 0.085 |
| | UCF [11] | 0.805 | 0.148 | 0.888 | 0.062 | 0.901 | 0.071 | 0.730 | 0.120 | 0.758 | 0.112 | 0.772 | 0.112 |
| | NLDF [12] | 0.837 | 0.123 | 0.902 | 0.048 | 0.902 | 0.066 | 0.753 | 0.080 | 0.762 | 0.080 | 0.806 | 0.065 |
| | PiCA [15] | 0.836 | **0.102** | **0.916** | **0.042** | **0.923** | **0.049** | 0.766 | 0.068 | **0.783** | 0.083 | **0.837** | **0.054** |
| | C2S [16] | 0.819 | **0.122** | 0.898 | 0.046 | 0.907 | 0.057 | 0.759 | 0.072 | **0.775** | 0.083 | 0.811 | 0.062 |
| | RAS [17] | **0.847** | 0.123 | **0.913** | 0.045 | **0.916** | 0.058 | **0.785** | **0.063** | 0.772 | **0.075** | **0.831** | 0.059 |
| | DSS [14] | **0.842** | **0.122** | **0.913** | **0.041** | 0.915 | **0.056** | **0.774** | **0.066** | 0.770 | **0.074** | 0.827 | **0.056** |
| | **ATDF (ours)** | **0.859** | **0.114** | **0.927** | **0.032** | **0.931** | **0.044** | **0.795** | **0.055** | **0.796** | **0.066** | **0.863** | **0.042** |
| ResNet | SRM [13] | 0.840 | 0.126 | 0.906 | 0.046 | 0.914 | 0.056 | 0.769 | 0.069 | 0.778 | 0.077 | 0.826 | 0.059 |
| | BRN [56] | 0.843 | **0.103** | 0.910 | 0.036 | 0.919 | 0.043 | 0.774 | 0.062 | 0.769 | 0.076 | 0.827 | 0.050 |
| | PiCA [15] | 0.852 | **0.103** | 0.917 | 0.043 | 0.929 | 0.049 | 0.789 | 0.065 | 0.788 | 0.081 | 0.853 | 0.050 |
| | CPD-R [62] | 0.857 | 0.110 | 0.925 | 0.034 | 0.936 | 0.041 | 0.797 | 0.056 | 0.799 | 0.068 | 0.865 | 0.043 |
| | **ATDF (ours)** | **0.862** | 0.110 | **0.933** | **0.031** | **0.939** | **0.040** | **0.814** | **0.051** | **0.801** | **0.064** | **0.877** | **0.037** |

| | Methods | SOD | | HKU-IS | | ECSSD | | DUT-OMRON | | THUR15K | | DUTS-test | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | $F_\beta^w$ | $F_\beta^m$ | $F_\beta^w$ | $F_\beta^m$ | $F_\beta^w$ | $F_\beta^m$ | $F_\beta^w$ | $F_\beta^m$ | $F_\beta^w$ | $F_\beta^m$ | $F_\beta^w$ | $F_\beta^m$ |
| VGG16 | MDF [52] | 0.528 | 0.654 | - | - | 0.619 | 0.736 | 0.494 | 0.620 | 0.508 | 0.618 | 0.507 | 0.636 |
| | LEGS [6] | 0.550 | 0.684 | 0.616 | 0.722 | 0.692 | 0.787 | 0.523 | 0.628 | 0.538 | 0.630 | 0.510 | 0.616 |
| | ELD [7] | 0.634 | 0.744 | 0.743 | 0.797 | 0.783 | 0.839 | 0.593 | 0.659 | 0.621 | 0.683 | 0.607 | 0.678 |
| | RFCN [8] | 0.591 | 0.749 | 0.707 | 0.835 | 0.725 | 0.848 | 0.562 | 0.697 | 0.592 | 0.712 | 0.586 | 0.733 |
| | DCL [9] | 0.669 | 0.784 | 0.770 | 0.844 | 0.782 | 0.854 | 0.584 | 0.697 | 0.624 | 0.705 | 0.632 | 0.733 |
| | DHS [21] | 0.685 | 0.801 | 0.816 | 0.871 | 0.837 | 0.887 | - | - | 0.666 | 0.7266 | 0.705 | 0.780 |
| | Amulet [10] | 0.674 | 0.775 | 0.817 | 0.862 | 0.839 | 0.886 | 0.626 | 0.697 | 0.650 | 0.715 | 0.657 | 0.733 |
| | UCF [11] | 0.673 | 0.778 | 0.779 | 0.840 | 0.805 | 0.865 | 0.574 | 0.668 | 0.613 | 0.697 | 0.595 | 0.697 |
| | NLDF [12] | 0.708 | 0.824 | 0.838 | 0.891 | 0.835 | 0.894 | 0.634 | 0.740 | 0.676 | 0.740 | 0.710 | 0.788 |
| | PiCA [15] | **0.721** | 0.806 | 0.847 | 0.884 | **0.862** | 0.900 | **0.691** | 0.739 | 0.688 | **0.743** | **0.745** | 0.799 |
| | C2S [16] | 0.700 | 0.808 | 0.835 | 0.874 | 0.849 | 0.891 | 0.663 | 0.734 | 0.685 | 0.739 | 0.717 | 0.779 |
| | RAS [17] | **0.718** | **0.833** | **0.850** | **0.894** | 0.855 | **0.901** | **0.695** | **0.764** | **0.691** | **0.748** | 0.739 | **0.804** |
| | DSS [14] | **0.711** | **0.829** | **0.862** | **0.899** | **0.864** | **0.912** | 0.688 | **0.760** | **0.702** | 0.740 | **0.752** | **0.820** |
| | **ATDF (ours)** | **0.718** | **0.841** | **0.883** | **0.920** | **0.888** | **0.926** | **0.716** | **0.789** | **0.739** | **0.782** | **0.797** | **0.856** |
| ResNet | SRM [13] | 0.670 | 0.811 | 0.835 | 0.886 | 0.849 | 0.898 | 0.658 | 0.748 | 0.684 | 0.748 | 0.721 | 0.799 |
| | BRN [56] | 0.670 | 0.811 | 0.835 | 0.886 | 0.849 | 0.898 | 0.658 | 0.748 | 0.684 | 0.748 | 0.721 | 0.799 |
| | PiCA [15] | **0.722** | 0.806 | 0.841 | 0.884 | 0.863 | 0.900 | 0.695 | 0.739 | 0.690 | 0.743 | 0.754 | 0.799 |
| | CPD-R [62] | 0.713 | 0.839 | 0.875 | 0.910 | 0.893 | 0.923 | 0.719 | 0.782 | 0.730 | 0.773 | 0.794 | **0.847** |
| | **ATDF (ours)** | **0.722** | **0.844** | **0.892** | **0.926** | **0.901** | **0.932** | **0.742** | **0.808** | **0.748** | **0.789** | **0.870** | 0.818 |

Table 1: Comparison between our ATDF and 16 state-of-the-art methods in terms of $F_\beta$ (↑), MAE (↓), $F_\beta^w$ (↑), and $F_\beta^m$ (↑) on six datasets. For VGG backbone, we highlight the top three results of each column in **red**, **green** and **blue**, respectively. For ResNet backbone, we only highlight the top performance of each column in **red**.

its training data. Due to the same reason, we do not evaluate DHS [21] on the DUT-OMRON dataset [37].

**Quantitative Evaluation.** We summarize the numeric comparison with respect to $F_\beta$, MAE, weighted $F$-measure score $F_\beta^\omega$, and average $F$-measure score $F_\beta^m$ on six datasets in Table 1. We use both VGG16 [64] and ResNet [69] as our backbones and report results on both of them. With VGG16 [64] backbone, ATDF achieves $F_\beta$ values of 85.9%, 92.7%, 93.1%, 79.5%, 79.6%, and 86.3%, which is 1.2%, 1.1%, 0.8%, 1.0%, 1.3%, and 2.6% higher than the second best model on the SOD, HKU-IS, ECSSD, DUT-OMRON, THUR15K, and DUTS-test datasets, respectively. For MAE, ATDF also achieves the best results on all datasets except SOD, on which a little lower performance than PiCA [15] is achieved. Moreover, the $F_\beta^\omega$ values of ATDF are 2.1%, 2.4%, 2.1%, 3.7%, and 4.5% higher than the second best method on the HKU-IS, ECSSD, DUT-OMRON, THUR15K, and DUTS-TE datasets, respectively. For SOD dataset, ATDF performs slightly worse than PiCA [15]. In Table 1, we also evaluate ATDF and above-mentioned competitors using $F_\beta^m$. The VGG16 version of ATDF achieves 0.8%, 2.1%, 1.4%, 2.5%, 3.4%, and 3.6% higher $F_\beta^\omega$-measure than the second best performance on the SOD, HKU-IS, ECSSD, DUT-OMRON, THUR15K, and DUTS-TE datasets, respectively. Overall, ATDF significantly outperforms other competitors in almost all cases, while PiCA [15] seems to achieves the second place.

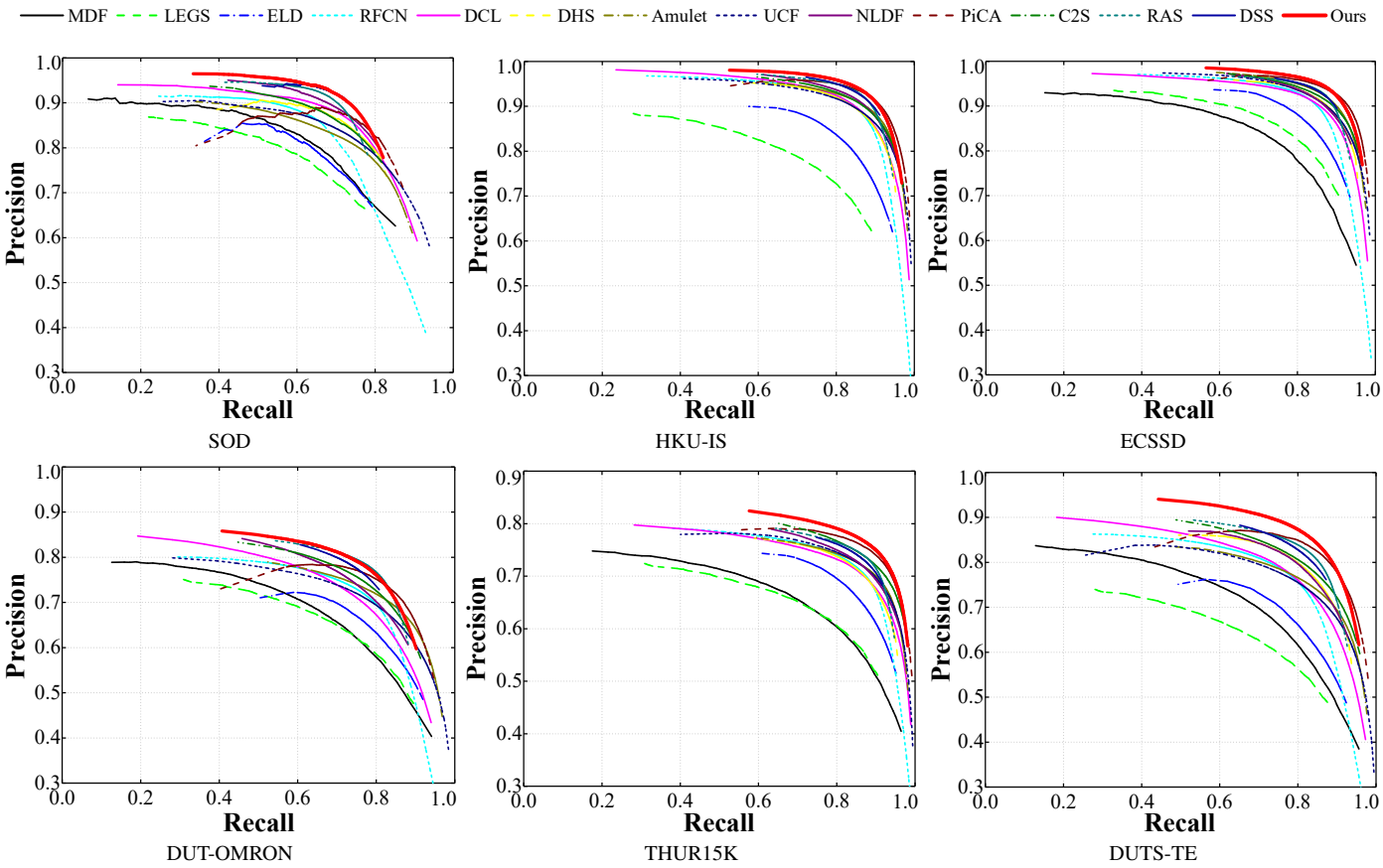


Figure 2: PR curves of the proposed ATDF and 13 state-of-the-art methods with the VGG16 backbone net. ATDF performs favorably against all competitors on all datasets.

We also use the ResNet as ATDF's backbone net and compare with ResNet based methods in Table 1. With the ResNet [69] backbone, ATDF also achieves significantly better performance than all baselines. It suggests that ATDF is robust to different backbone networks. Therefore, we can come to the conclusion that the proposed ATDF performs favorably against state-of-the-art approaches.

**PR curves, FT curves, and histograms.** We display the PR curves, FT curves and the histograms for precision, recall, and F-measure scores of ATDF and baselines on all datasets using VGG16 backbone net in the Fig. 2, Fig. 3, and Fig. 4, respectively. It is obvious that the proposed ATDF substantially performs favorably against other counterparts.

**Qualitative Evaluation.** To further prove the performance of our model over previous methods, we show the qualitative comparison between ATDF and previous state-of-the-art methods in Fig. 5. We select some representative images from various datasets to incorporate a variety of difficult circumstances, including complex scenes, salient objects with thin structures, low contrast between foreground and background, multiple objects with different sizes, and *etc*. Taking all circumstances into account, we can see that our model can successfully segment the objects with fine details, leading to better saliency predictions in various scenarios, including small objects, large objects, and low contrast between foreground and background.

## 5. Ablation Studies

In this section, we conduct a series of ablation experiments to further analyze the effectiveness of each model component.

**Loss Function.** Different from other models that use class-balanced cross-entropy *sigmoid* loss function [14, 10, 25] which can be formulated as

$$L_R = -\sum_{x,y}[\beta \cdot (1 - l_{x,y}) \cdot log(1 - P_{x,y}) + (1 - \beta) \cdot l_{x,y} \cdot log(P_{x,y})], \tag{11}$$

we use the unweighted *sigmoid* cross-entropy loss to train the proposed ATDF, which can be formulated as

$$L_U = -\sum_{x,y}[(1 - l_{x,y}) \cdot log(1 - P_{x,y}) + l_{x,y} \cdot log(P_{x,y})]. \tag{12}$$

Here, $\beta = |Y^+|/|Y|$ and $1 - \beta = |Y^-|/|Y|$. $Y^+$ and $Y^-$ represent salient pixels and non-salient pixels of the ground truth, respectively. $l_{x,y} \in \{0, 1\}$ is the label of pixel $(x, y)$, and $P_{x,y}$ is the predicted probability of pixel $(x, y)$ belonging to the foreground, which is computed by the standard *sigmoid* function.

We replace our loss function $L_U$ by the class-balanced cross-entropy loss function $L_R$. The results are shown in Table 2.

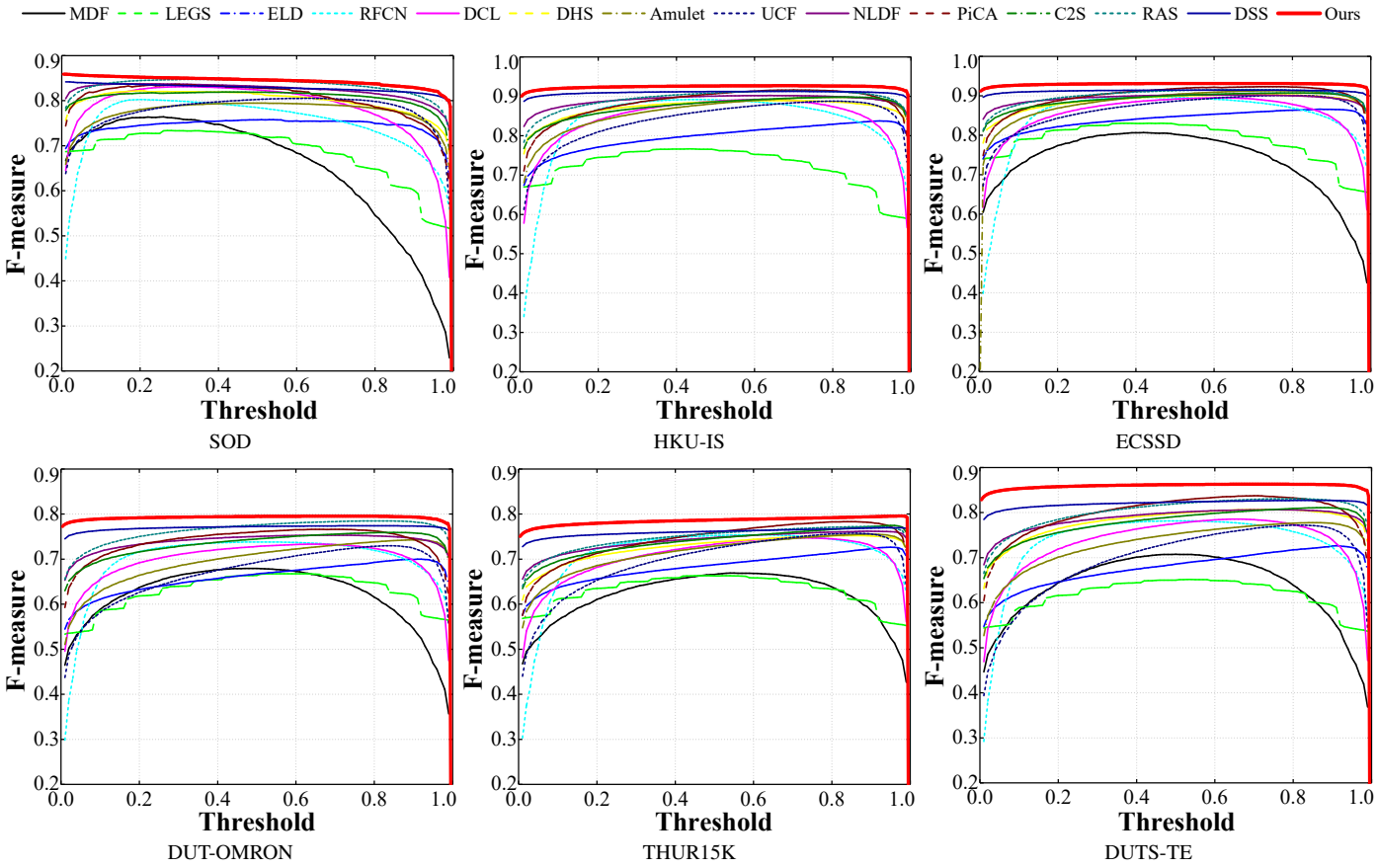


Figure 3: FT curves of the proposed ATDF and 13 state-of-the-art methods with the VGG16 backbone net. ATDF performs favorably against all competitors on all datasets.

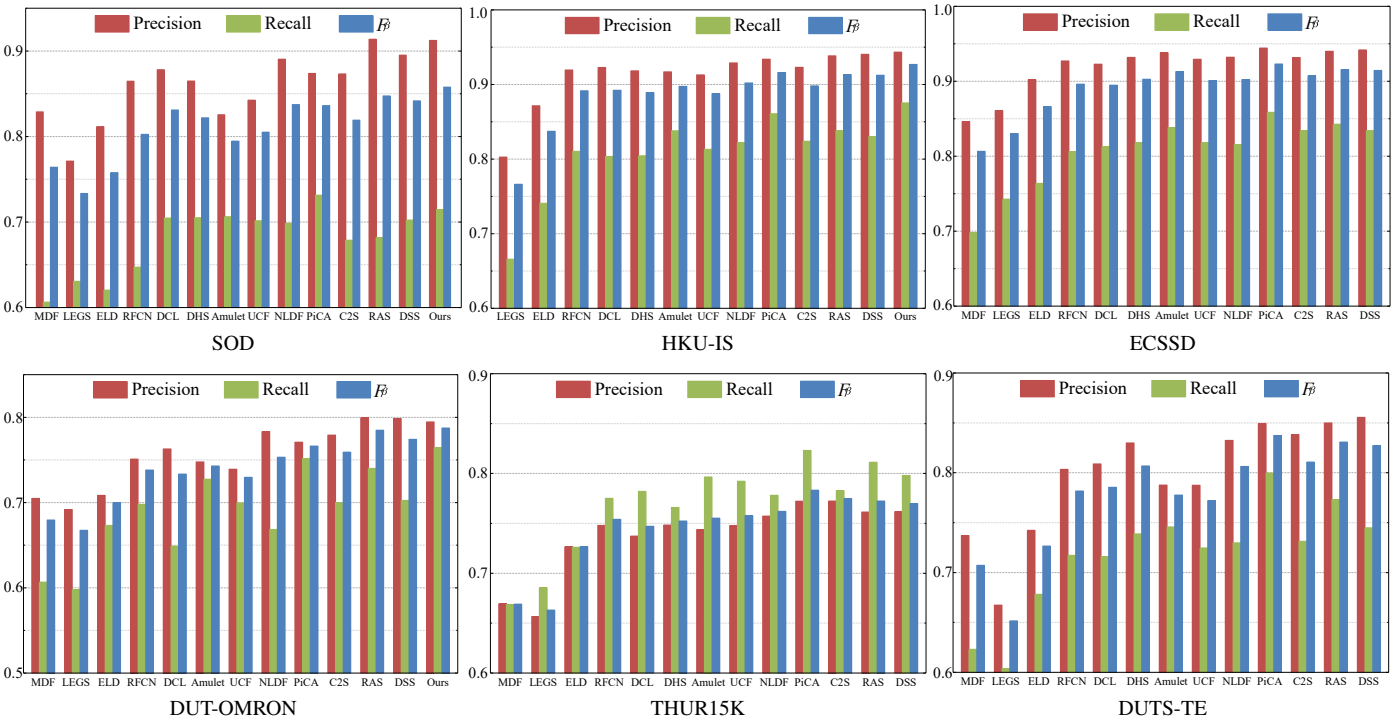


Figure 4: Histograms for precision, recall, and $F$-measure scores with the VGG16 backbone net. ATDF performs favorably against all competitors on all datasets.


Figure 5: Qualitative comparison between ATDF and recent state-of-the-art methods.

We can observe that the model with unweighted *sigmoid* cross-entropy loss performs better for salient object detection. ATDF with $L_R$ achieves $F_\beta$ values of 85.9%, 92.3%, 92.9%, 78.5%, 79.2%, and 85.7%, which is 0%, 0.4%, 0.2%, 1.0%, 0.4%, and 0.6% worse than ATDF with $L_U$ on the SOD, HKU-IS, ECSSD, DUT-OMRON, THUR15K, and DUTS-test datasets, respectively. For MAE, ATDF with $L_R$ achieves 10.7%, 3.3%, 4.3%, 6.0%, 7.1%, and 4.6%. Except for SOD dataset, on which ATDF with $L_U$ achieves a little lower performance than ATDF with $L_R$, the performance of ATDF with $L_U$ is 0.1%,

| No. | SOD | | HKU-IS | | ECSSD | | DUT-OMRON | | THUR15K | | DUTS-test | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | $F_\beta$ | MAE | $F_\beta$ | MAE | $F_\beta$ | MAE | $F_\beta$ | MAE | $F_\beta$ | MAE | $F_\beta$ | MAE |
| 1 | 0.859 | 0.107 | 0.923 | 0.033 | 0.929 | 0.043 | 0.785 | 0.060 | 0.792 | 0.071 | 0.857 | 0.046 |
| 2 | 0.847 | 0.115 | 0.922 | 0.033 | 0.927 | 0.044 | 0.781 | 0.061 | 0.789 | 0.072 | 0.851 | 0.047 |
| * | 0.859 | 0.114 | 0.927 | 0.032 | 0.931 | 0.044 | 0.795 | 0.055 | 0.796 | 0.066 | 0.863 | 0.042 |

Table 2: Evaluation results for the effectiveness of the loss function and weight function in Section 5. The first row uses class-balanced cross-entropy loss function $L_R$. The second row removes the weight function from the proposed network. * denotes the default settings.

| No. | Module | Side 1 | Side 2 | Side 3 | Side 4 | Side 5 | Side 6 |
|---|---|---|---|---|---|---|---|
| 1 | Valve | $(64, 3\times3)\times2$ | $(128, 3\times3)\times2$ | $(128, 3\times3)\times2$ | $(256, 3\times3)\times2$ | $(256, 3\times3)\times2$ | - |
| 2 | Valve | $(64, 3\times3)\times2$ | $(128, 3\times3)\times2$ | $(128, 3\times3)\times2$ | $(512, 5\times5)\times2$ | $(512, 5\times5)\times2$ | - |
| 3 | Valve | $(64, 3\times3)\times2$ | $(128, 3\times3)\times2$ | $(128, 3\times3)\times2$ | $(128, 5\times5)\times2$ | $(128, 5\times5)\times2$ | - |
| 4 | Valve | $(64, 3\times3)\times2$ | $(64, 3\times3)\times2$ | $(64, 3\times3)\times2$ | $(128, 5\times5)\times2$ | $(128, 5\times5)\times2$ | - |
| 5 | Generator | $(128, 3\times3)\times2$ | $(128, 3\times3)\times2$ | $(256, 5\times5)\times2$ | $(256, 5\times5)\times2$ | $(512, 5\times5)\times2$ | $(512, 5\times5)\times2$ |
| 6 | Generator | $(64, 3\times3)\times2$ | $(128, 3\times3)\times2$ | $(128, 3\times3)\times2$ | $(256, 5\times5)\times2$ | $(256, 5\times5)\times2$ | $(256, 7\times7)\times2$ |
| 7 | Generator | $(64, 3\times3)\times2$ | $(128, 3\times3)\times2$ | $(128, 3\times3)\times2$ | $(256, 5\times5)\times2$ | $(256, 5\times5)\times2$ | $(1024, 7\times7)\times2$ |
| * | Valve | $(64, 3\times3)\times2$ | $(128, 3\times3)\times2$ | $(128, 3\times3)\times2$ | $(256, 5\times5)\times2$ | $(256, 5\times5)\times2$ | - |
| | Generator | $(64, 3\times3)\times2$ | $(128, 3\times3)\times2$ | $(128, 3\times3)\times2$ | $(256, 5\times5)\times2$ | $(256, 5\times5)\times2$ | $(512, 7\times7)\times2$ |

Table 3: Various parameter settings for ablation studies. * means the default settings used in this paper. The column of *Module* indicates which module is changed, and another module remains the default settings in the meanwhile.

| No. | SOD | | HKU-IS | | ECSSD | | DUT-OMRON | | THUR15K | | DUTS-test | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | $F_\beta$ | MAE | $F_\beta$ | MAE | $F_\beta$ | MAE | $F_\beta$ | MAE | $F_\beta$ | MAE | $F_\beta$ | MAE |
| 1 | 0.857 | 0.116 | 0.924 | 0.033 | 0.929 | 0.046 | 0.785 | 0.058 | 0.793 | 0.066 | 0.859 | 0.043 |
| 2 | 0.855 | 0.116 | 0.926 | 0.033 | 0.931 | 0.045 | 0.789 | 0.056 | 0.795 | 0.066 | 0.864 | 0.042 |
| 3 | 0.859 | 0.115 | 0.927 | 0.032 | 0.932 | 0.044 | 0.792 | 0.056 | 0.797 | 0.065 | 0.863 | 0.042 |
| 4 | 0.857 | 0.116 | 0.927 | 0.032 | 0.932 | 0.045 | 0.793 | 0.055 | 0.796 | 0.065 | 0.864 | 0.042 |
| 5 | 0.860 | 0.116 | 0.927 | 0.033 | 0.930 | 0.044 | 0.794 | 0.055 | 0.796 | 0.066 | 0.863 | 0.042 |
| 6 | 0.856 | 0.122 | 0.927 | 0.032 | 0.931 | 0.044 | 0.790 | 0.056 | 0.795 | 0.066 | 0.862 | 0.043 |
| 7 | 0.856 | 0.119 | 0.927 | 0.033 | 0.932 | 0.045 | 0.788 | 0.057 | 0.796 | 0.066 | 0.863 | 0.043 |
| * | 0.859 | 0.114 | 0.927 | 0.032 | 0.931 | 0.044 | 0.795 | 0.055 | 0.796 | 0.066 | 0.863 | 0.042 |

Table 4: Evaluation results for various parameter settings. See Table 3 for parameter settings with corresponding numbers.

0.1%, 0.5%, 0.5%, and 0.4% better than ATDF with $L_R$ on the HKU-IS, ECSSD, DUT-OMRON, THUR15K, and DUTS-test datasets, respectively.

**Weight Function.** In the valve module, We design a weight function to control the information flowing rate. To prove the effectiveness of the weight function, we further conduct experiments without the weight function. As the results shown in Table 2, without the weight function in the *valve* module, we obtain ($F_\beta$, MAE) of (84.7%, 11.5%), (92.2%, 3.3%), (92.7%, 4.4%), (78.1%, 6.1%), (78.9%, 7.2%), and (85.1%, 4.7%) which are (1.2%, 0.1%), (0.5%, 0.1%), (0.4%, 0), (1.4%, 0.6%), (0.7%, 0.6%), and (1.2%, 0.5%) worse than ATDF on six datasets as the same order of Table 1, respectively. Hence the designed weight function can significantly benefit the proposed saliency detector.

**Parameter Settings.** To evaluate the influence of various parameter settings for the *valve* module and *generator* module, *e.g.*, the convolution parameters of the two residual blocks and the weight function, we additionally perform seven ablation studies with VGG16 backbone. Table 3 shows various parameters settings, while the corresponding evaluation results are shown in first seven rows of the Table 4. We can observe that our proposed model is not sensitive to different parameter settings, and the default setting is slightly better than other settings.

## 6. Conclusion

In this paper, we propose a novel Automatic Top-Down Fusion (ATDF) model for saliency detection, which can automatically flow the global information at the top sides into bottom sides to guide the learning of bottom sides. To do this, we design a *valve* module to filter out redundant and remain the necessary top global information for each side. ATDF is simple yet effective in integrating the multi-level convolutional features of deep networks. Therefore, ATDF overcomes the problem of the sub-optimal saliency detection in recent manually designed fusion strategies. Extensive experiments on six datasets demonstrate that ATDF outperforms 16 recent state-of-the-art methods in terms of various evaluation metrics.

## Acknowledgement

This work is supported by Science and Technology Planning Project of Tianjin (17JCZDJC30700 and 18ZXZNGX00310), Fundamental Research Funds for the Central Universities of Nankai University (63191402).

## References


[1] R. Achanta, S. Hemami, F. Estrada, S. Susstrunk, Frequency-tuned salient region detection, in: IEEE Conf. Comput. Vis. Pattern Recog. (CVPR), 2009, pp. 1597–1604.

[2] M.-M. Cheng, N. J. Mitra, X. Huang, P. H. Torr, S.-M. Hu, Global contrast based salient region detection, IEEE Trans. Pattern Anal. Mach. Intell. (TPAMI) 37 (3) (2015) 569–582.

[3] V. Mahadevan, N. Vasconcelos, Saliency-based discriminant tracking, in: IEEE Conf. Comput. Vis. Pattern Recog. (CVPR), 2009, pp. 1007–1013.

[4] Y. Gao, M. Wang, Z.-J. Zha, J. Shen, X. Li, X. Wu, Visual-textual joint relevance learning for tag-based social image search, IEEE Trans. Image Process. (TIP) 22 (1) (2013) 363–376.

[5] Y. Wei, X. Liang, Y. Chen, X. Shen, M.-M. Cheng, J. Feng, Y. Zhao, S. Yan, STC: A simple to complex framework for weakly-supervised semantic segmentation, IEEE Trans. Pattern Anal. Mach. Intell. (TPAMI) 39 (11) (2017) 2314–2320.

[6] L. Wang, H. Lu, X. Ruan, M.-H. Yang, Deep networks for saliency detection via local estimation and global search, in: IEEE Conf. Comput. Vis. Pattern Recog. (CVPR), 2015, pp. 3183–3192.

[7] G. Lee, Y.-W. Tai, J. Kim, Deep saliency with encoded low level distance map and high level features, in: IEEE Conf. Comput. Vis. Pattern Recog. (CVPR), 2016, pp. 660–668.

[8] L. Wang, L. Wang, H. Lu, P. Zhang, X. Ruan, Saliency detection with recurrent fully convolutional networks, in: Eur. Conf. Comput. Vis. (ECCV), 2016, pp. 825–841.

[9] G. Li, Y. Yu, Deep contrast learning for salient object detection, in: IEEE Conf. Comput. Vis. Pattern Recog. (CVPR), 2016, pp. 478–487.

[10] P. Zhang, D. Wang, H. Lu, H. Wang, X. Ruan, Amulet: Aggregating multi-level convolutional features for salient object detection, in: Int. Conf. Comput. Vis. (ICCV), 2017, pp. 202–211.

[11] P. Zhang, D. Wang, H. Lu, H. Wang, B. Yin, Learning uncertain convolutional features for accurate saliency detection, in: Int. Conf. Comput. Vis. (ICCV), 2017, pp. 212–221.

[12] Z. Luo, A. K. Mishra, A. Achkar, J. A. Eichel, S. Li, P.-M. Jodoin, Non-local deep features for salient object detection, in: IEEE Conf. Comput. Vis. Pattern Recog. (CVPR), 2017, pp. 6609–6617.

[13] T. Wang, A. Borji, L. Zhang, P. Zhang, H. Lu, A stagewise refinement model for detecting salient objects in images, in: Int. Conf. Comput. Vis. (ICCV), 2017, pp. 4019–4028.

[14] Q. Hou, M.-M. Cheng, X. Hu, A. Borji, Z. Tu, P. Torr, Deeply supervised salient object detection with short connections, IEEE Trans. Pattern Anal. Mach. Intell. (TPAMI) 41 (4) (2019) 815–828.

[15] N. Liu, J. Han, M.-H. Yang, PiCANet: Learning pixel-wise contextual attention for saliency detection, in: IEEE Conf. Comput. Vis. Pattern Recog. (CVPR), 2018, pp. 3089–3098.

[16] X. Li, F. Yang, H. Cheng, W. Liu, D. Shen, Contour knowledge transfer for salient object detection, in: Eur. Conf. Comput. Vis. (ECCV), 2018, pp. 355–370.

[17] S. Chen, X. Tan, B. Wang, X. Hu, Reverse attention for salient object detection, in: Eur. Conf. Comput. Vis. (ECCV), 2018, pp. 234–250.

[18] H. Jiang, J. Wang, Z. Yuan, Y. Wu, N. Zheng, S. Li, Salient object detection: A discriminative regional feature integration approach, in: IEEE Conf. Comput. Vis. Pattern Recog. (CVPR), 2013, pp. 2083–2090.

[19] N. Tong, H. Lu, X. Ruan, M.-H. Yang, Salient object detection via bootstrap learning, in: IEEE Conf. Comput. Vis. Pattern Recog. (CVPR), 2015, pp. 1884–1892.

[20] S. S. Naqvi, J. Mirza, T. Bashir, A unified framework for exploiting color coefficients for salient object detection, Neurocomputing 312 (2018) 187–200.

[21] N. Liu, J. Han, DHSNet: Deep hierarchical saliency network for salient object detection, in: IEEE Conf. Comput. Vis. Pattern Recog. (CVPR), 2016, pp. 678–686.

[22] Quan, Rong and Han, Junwei and Zhang, Dingwen and Nie, Feiping and Qian, Xueming and Li, Xuelong, Unsupervised salient object detection via inferring from imperfect saliency models, IEEE Trans. Multimedia (TMM) 20 (5) (2017) 1101–1112.

[23] L. Zhang, J. Dai, H. Lu, Y. He, G. Wang, A bi-directional message passing model for salient object detection, in: IEEE Conf. Comput. Vis. Pattern Recog. (CVPR), 2018, pp. 1741–1750.

[24] W. Wang, J. Shen, X. Dong, A. Borji, Salient object detection driven by fixation prediction, in: IEEE Conf. Comput. Vis. Pattern Recog. (CVPR), 2018, pp. 1711–1720.

[25] J.-J. Liu, Q. Hou, M.-M. Cheng, J. Feng, J. Jiang, A simple pooling-based design for real-time salient object detection, in: IEEE Conf. Comput. Vis. Pattern Recog. (CVPR), 2019, pp. 3917–3926.

[26] Y. Liu, M.-M. Cheng, X. Hu, J.-W. Bian, L. Zhang, X. Bai, J. Tang, Richer convolutional features for edge detection, IEEE Trans. Pattern Anal. Mach. Intell. (TPAMI) 41 (8) (2019) 1939–1946.

[27] Y. Liu, D.-P. Fan, G.-Y. Nie, X. Zhang, V. Petrosyan, M.-M. Cheng, DNA: Deeply-supervised nonlinear aggregation for salient object detection, arXiv preprint arXiv:1903.12476.

[28] L. Zhang, Y. Zhang, H. Yan, Y. Gao, W. Wei, Salient object detection in hyperspectral imagery using multi-scale spectral-spatial gradient, Neurocomputing 291 (2018) 215–225.

[29] Y. Ji, H. Zhang, K. Tseng, T. W. S. Chow, Q. M. J. Wu, Graph model-based salient object detection using objectness and multiple saliency cues, Neurocomputing 323 (2019) 188–202.

[30] Y. Qiu, Y. Liu, X. Ma, L. Liu, H. Gao, J. Xu, Revisiting multi-level feature fusion: A simple yet effective network for salient object detection, in: IEEE Int. Conf. Image Process. (ICIP), 2019, pp. 4010–4014.

[31] C. Gong, D. Tao, W. Liu, S. J. Maybank, M. Fang, K. Fu, J. Yang, Saliency propagation from simple to difficult, in: IEEE Conf. Comput. Vis. Pattern Recog. (CVPR), 2015, pp. 2531–2539.

[32] W.-C. Tu, S. He, Q. Yang, S.-Y. Chien, Real-time salient object detection with a minimum spanning tree, in: IEEE Conf. Comput. Vis. Pattern Recog. (CVPR), 2016, pp. 2334–2342.

[33] C. Xia, J. Li, X. Chen, A. Zheng, Y. Zhang, What is and what is not a salient object? learning salient object detector by ensembling linear exemplar regressors, in: IEEE Conf. Comput. Vis. Pattern Recog. (CVPR), 2017, pp. 4321–4329.

[34] B. Jiang, Z. He, C. Ding, B. Luo, Saliency detection via a multi-layer graph based diffusion model, Neurocomputing 314 (2018) 215–223.

[35] Y. Xiao, B. Jiang, A. Zheng, A. Zhou, A. Hussain, J. Tang, Saliency detection via multi-view graph based saliency optimization, Neurocomputing 351 (2019) 156–166.

[36] Z. Jiang, L. S. Davis, Submodular salient region detection, in: IEEE Conf. Comput. Vis. Pattern Recog. (CVPR), 2013, pp. 2043–2050.

[37] C. Yang, L. Zhang, H. Lu, X. Ruan, M.-H. Yang, Saliency detection via graph-based manifold ranking, in: IEEE Conf. Comput. Vis. Pattern Recog. (CVPR), 2013, pp. 3166–3173.

[38] W. Zhu, S. Liang, Y. Wei, J. Sun, Saliency optimization from robust background detection, in: IEEE Conf. Comput. Vis. Pattern Recog. (CVPR), 2014, pp. 2814–2821.

[39] Han, Junwei and Zhang, Dingwen and Hu, Xintao and Guo, Lei and Ren, Jinchang and Wu, Feng, Background prior-based salient object detection via deep reconstruction residual, IEEE Trans. Circ. Syst. Video Technol. (TCSVT) 25 (8) (2014) 1309–1321.

[40] Han, Junwei and Ngan, King Ngi and Li, Mingjing and Zhang, Hong-Jiang, Unsupervised extraction of visual attention objects in color images, IEEE Trans. Circ. Syst. Video Technol. (TCSVT) 16 (1) (2005) 141–145.

[41] Han, Junwei and Zhang, Dingwen and Cheng, Gong and Liu, Nian and Xu, Dong, Advanced deep-learning techniques for salient and category-specific object detection: A survey, IEEE Signal Process. Mag. (SPM) 35 (1) (2018) 84–100.

[42] M.-M. Cheng, Y. Liu, W.-Y. Lin, Z. Zhang, P. L. Rosin, P. H. Torr, BING: Binarized normed gradients for objectness estimation at 300fps, Computational Visual Media 5 (1) (2019) 3–20.

[43] Z. Zhang, Y. Liu, X. Chen, Y. Zhu, M.-M. Cheng, V. Saligrama, P. H. Torr, Sequential optimization for efficient high-quality object proposal generation, IEEE Trans. Pattern Anal. Mach. Intell. (TPAMI) 40 (5) (2017) 1209–1223.

[44] D. Zhang, J. Han, Y. Zhang, Supervision by fusion: Towards unsupervised learning of deep salient object detector, in: Int. Conf. Comput. Vis. (ICCV), 2017, pp. 4048–4056.

[45] J. Zhang, T. Zhang, Y. Dai, M. Harandi, R. Hartley, Deep unsupervised saliency detection: A multiple noisy labeling perspective, in: IEEE Conf. Comput. Vis. Pattern Recog. (CVPR), 2018, pp. 9029–9038.

[46] H. R. Tavakoli, F. Ahmed, A. Borji, J. Laaksonen, Saliency revisited: Analysis of mouse movements versus fixations, in: IEEE Conf. Comput. Vis. Pattern Recog. (CVPR), 2017, pp. 6354–6362.

[47] C. Lang, J. Feng, S. Feng, J. Wang, S. Yan, Dual low-rank pursuit: Learn-

ing salient features for saliency detection, IEEE Trans. Neur. Net. Learn. Syst. (TNNLS) 27 (6) (2016) 1190–1200.
[48] X. Li, L. Zhao, L. Wei, M.-H. Yang, F. Wu, Y. Zhuang, H. Ling, J. Wang, DeepSaliency: Multi-task deep neural network model for salient object detection, IEEE Trans. Image Process. (TIP) 25 (8) (2016) 3919–3930.
[49] X. Xi, Y. Luo, P. Wang, H. Qiao, Salient object detection based on an efficient end-to-end saliency regression network, Neurocomputing 323 (2019) 265–276.
[50] K. Fu, Q. Zhao, I. Y. Gu, J. Yang, Deepside: A general deep framework for salient object detection, Neurocomputing 356 (2019) 69–82.
[51] C. Li, Z. Chen, Q. M. J. Wu, C. Liu, Deep saliency detection via channel-wise hierarchical feature responses, Neurocomputing 322 (2018) 80–92.
[52] G. Li, Y. Yu, Visual saliency based on multiscale deep features, in: IEEE Conf. Comput. Vis. Pattern Recog. (CVPR), 2015, pp. 5455–5463.
[53] M.-M. Cheng, Y. Liu, Q. Hou, J. Bian, P. Torr, S.-M. Hu, Z. Tu, HFS: Hierarchical feature selection for efficient image segmentation, in: Eur. Conf. Comput. Vis. (ECCV), Springer, 2016, pp. 867–882.
[54] X. Zhang, T. Wang, J. Qi, H. Lu, G. Wang, Progressive attention guided recurrent network for salient object detection, in: IEEE Conf. Comput. Vis. Pattern Recog. (CVPR), 2018, pp. 714–722.
[55] Y. Zeng, H. Lu, L. Zhang, M. Feng, A. Borji, Learning to promote saliency detectors, in: IEEE Conf. Comput. Vis. Pattern Recog. (CVPR), 2018, pp. 1644–1653.
[56] T. Wang, L. Zhang, S. Wang, H. Lu, G. Yang, X. Ruan, A. Borji, Detect globally, refine locally: A novel approach to saliency detection, in: IEEE Conf. Comput. Vis. Pattern Recog. (CVPR), 2018, pp. 3127–3135.
[57] M. A. Islam, M. Kalash, N. D. Bruce, Revisiting salient object detection: Simultaneous detection, ranking, and subitizing of multiple salient objects, in: IEEE Conf. Comput. Vis. Pattern Recog. (CVPR), 2018, pp. 7142–7150.
[58] X. Chen, A. Zheng, J. Li, F. Lu, Look, perceive and segment: Finding the salient objects in images via two-stream fixation-semantic CNNs, in: Int. Conf. Comput. Vis. (ICCV), 2017, pp. 1050–1058.
[59] N. Liu, J. Han, A deep spatial contextual long-term recurrent convolutional network for saliency detection, IEEE Trans. Image Process. (TIP) 27 (7) (2018) 3264–3274.
[60] N. Liu, J. Han, T. Liu, X. Li, Learning to predict eye fixations via multiresolution convolutional neural networks, IEEE Trans. Neur. Net. Learn. Syst. (TNNLS) 29 (2) (2018) 392–404.
[61] J. Chi, C. Wu, X. Yu, H. Chu, P. Ji, Saliency detection via integrating deep learning architecture and low-level features, Neurocomputing 352 (2019) 75–92.
[62] Z. Wu, L. Su, Q. Huang, Cascaded partial decoder for fast and accurate salient object detection, in: IEEE Conf. Comput. Vis. Pattern Recog. (CVPR), 2019, pp. 3907–3916.
[63] T. Zhao, X. Wu, Pyramid feature attention network for saliency detection, in: IEEE Conf. Comput. Vis. Pattern Recog. (CVPR), 2019, pp. 3085–3094.
[64] K. Simonyan, A. Zisserman, Very deep convolutional networks for large-scale image recognition, in: Int. Conf. Learn. Represent. (ICLR), 2015.
[65] S. Xie, Z. Tu, Holistically-nested edge detection, in: Int. Conf. Comput. Vis. (ICCV), 2015, pp. 1395–1403.
[66] J. Long, E. Shelhamer, T. Darrell, Fully convolutional networks for semantic segmentation, in: IEEE Conf. Comput. Vis. Pattern Recog. (CVPR), 2015, pp. 3431–3440.
[67] Nair, Vinod and Hinton, Geoffrey E, Rectified linear units improve restricted boltzmann machines, in: Int. Conf. Mach. Learn. (ICML), 2010, pp. 807–814.
[68] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, T. Darrell, Caffe: Convolutional architecture for fast feature embedding, in: ACM Int. Conf. Multimedia (ACM MM), 2014, pp. 675–678.
[69] K. He, X. Zhang, S. Ren, J. Sun, Deep residual learning for image recognition, in: IEEE Conf. Comput. Vis. Pattern Recog. (CVPR), 2016, pp. 770–778.
[70] L. Wang, H. Lu, Y. Wang, M. Feng, D. Wang, B. Yin, X. Ruan, Learning to detect salient objects with image-level supervision, in: IEEE Conf. Comput. Vis. Pattern Recog. (CVPR), 2017, pp. 136–145.
[71] V. Movahedi, J. H. Elder, Design and perceptual validation of performance measures for salient object segmentation, in: IEEE Conf. Comput. Vis. Pattern Recog. (CVPR), 2010, pp. 49–56.
[72] Q. Yan, L. Xu, J. Shi, J. Jia, Hierarchical saliency detection, in: IEEE Conf. Comput. Vis. Pattern Recog. (CVPR), 2013, pp. 1155–1162.
[73] M.-M. Cheng, N. J. Mitra, X. Huang, S.-M. Hu, SalientShape: Group saliency in image collections, The Vis. Comput. (TVC) 30 (4) (2014) 443–453.
[74] R. Margolin, L. Zelnik-Manor, A. Tal, How to evaluate foreground maps?, in: IEEE Conf. Comput. Vis. Pattern Recog. (CVPR), 2014, pp. 248–255.