Interactive Two-Stream Decoder for Accurate and Fast Saliency Detection

Huajun Zhou[1], Xiaohua Xie[1,2,3]*, Jianhuang Lai[1,2,3], Zixuan Chen[1], Lingxiao Yang[1]
[1]School of Data and Computer Science, Sun Yat-sen University, China
[2]Guangdong Province Key Laboratory of Information Security Technology, China
[3]Key Laboratory of Machine Intelligence and Advanced Computing, Ministry of Education, China

Abstract

Recently, contour information largely improves the performance of saliency detection. However, the discussion on the correlation between saliency and contour remains scarce. In this paper, we first analyze such correlation and then propose an interactive two-stream decoder to explore multiple cues, including saliency, contour and their correlation. Specifically, our decoder consists of two branches, a saliency branch and a contour branch. Each branch is assigned to learn distinctive features for predicting the corresponding map. Meanwhile, the intermediate connections are forced to learn the correlation by interactively transmitting the features from each branch to the other one. In addition, we develop an adaptive contour loss to automatically discriminate hard examples during learning process. Extensive experiments on six benchmarks well demonstrate that our network achieves competitive performance with a fast speed around 50 FPS. Moreover, our VGG-based model only contains 17.08 million parameters, which is significantly smaller than other VGG-based approaches. Code has been made available at: https://github.com/moothes/ITSD-pytorch.

1. Introduction

Saliency detection is a task to segment the most visually distinctive objects or regions in an image. The main challenge of this task is to distinguish the salient objects as well as their boundaries. Different from other segmentation techniques, such as semantic or instance segmentation, saliency detection always focuses on a few primary regions. Therefore, it often serves as the first step in many researches, like object tracking [20], object recognition [28], action categorization [1] and so on.

*Corresponding author. This project is supported by the Natural Science Foundation of China (61672544, 61876104), the Key Field R & D Program of Guangdong Province (2019B010155003), and the Tip-top Scientic and Technical Innovative Youth Talents of Guangdong Special Support Program (2016TQ03X263).

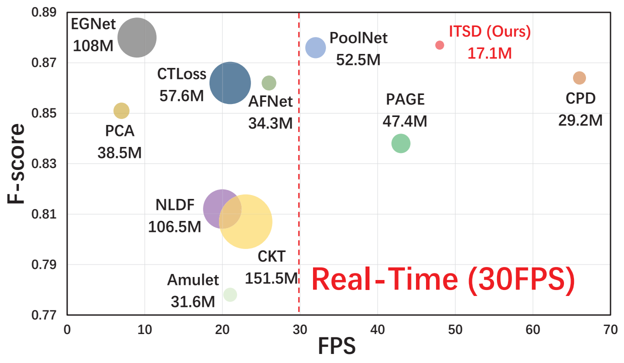

Figure 1. F-score and FPS comparisons of our ITSD with AFNet [8], PCA [22], Amulet [45], EGNet [48], CKT [18], CTLoss [4], NLDF [24], PoolNet [21], PAGE [35] and CPD [37] on DUTS-TE dataset [31]. All methods are based on VGG [29] network. The radius of circles are proportional to their model sizes.

Recently, Convolutional Neural Networks (CNNs) [16, 29, 10] have been introduced for this task and achieved very promising results on many benchmarks. One representative network – U-Net [27] effectively combines low-level and high-level cues and therefore can generate more accurate detection results. More recently, many researches [18, 24, 48, 26] further improve the U-shape structure by incorporating contour information.

However, existing saliency detection models still hold many problems that are not well addressed. First, the resulting saliency maps are still far away from being satisfactory, especially when encountering complex scenes. Second, to achieve good performance, most of existing works have led to a radical increase in the number of model parameters. Furthermore, the complexity of such models causes a slow detection speed. As an example, EGNet [48] provides state-of-the-art performance as shown in Figure 1. This approach contains around 108 million trainable parameters and runs around only 10 FPS for inference, which may hinder its usage in other applications like real-time video understanding. On the other hand, some smaller and faster models, such as CPD [37] and PAGE [35], cannot obtain comparable results. It is interesting to investigate whether we can design a lightweight model for accurate and fast detection.

In this work, we aim at discovering the contour information of saliency targets to improve the performance and to reduce the computational cost. Note that the contour maps discussed in this paper are calculated from the boundary of saliency regions, not the boundaries of all objects. Close to our method, some previous works [48, 24, 26, 21] also take advantage of contour for saliency detection. However, the correlation between the contour map and the saliency map is rarely explored, which can be further utilized as an important cue to improve the result segmentation maps as our experiment demonstrated.

To exploit both saliency and contour information, we propose a lightweight two-stream model that uses two branches to learn the representations of salient regions and their contours respectively. Furthermore, to promote the network to learn their correlation, we propose a new fusion module that can well exploit information from each branch. Additionally, we develop an adaptive contour loss to explore the hard examples located near the boundary of the salient regions. Extensive experiments validate that the proposed method achieves comparable results against some state-of-the-art approaches on six popular benchmarks. Moreover, our model just has 17.08 million parameters and runs at around 50 FPS, which provides a good balance between accuracy, model size and speed as shown in Figure 1. In summary, our main contributions are:

1) We discuss the correlation between the saliency maps and corresponding contour maps.
2) We propose a lightweight Interactive Two-Stream Decoder (ITSD) for saliency detection by exploring multiple cues of the saliency and contour maps. We further propose a fusion module to learn their correlation.
3) We develop an Adaptive ConTour (ACT) loss to improve the representation power of the learned network by taking advantage of hard examples.
4) We conduct a series of experiments to demonstrate the effectiveness and efficiency of the proposed model.

2. Related work

In this section, we mainly discuss some works based on deep neural networks. For other researches please refer the recent survey [2, 6, 34, 23, 33] for more details.

2.1. High accuracy saliency detection

To adapt the deep networks to the segmentation task, Ronneberger et al. introduced U-Net [27] to gradually expand the learned feature maps to input sizes by utilizing an encoder-decoder structure. In the U-shape network, features from the encoder are skip-connected to the features with the same spatial sizes in the decoder, where this simple connection has been demonstrated its power in various pixel-wise predictions [14, 39, 43].

Due to the remarkable performance of U-Net, many researches follow this structure for saliency detection. Zhang et al. [45] achieved good performance by improving the fusion module in U-Net. Besides, hierarchical supervision signals are attached to intermediate features. Luo et al. [24] predicted salient regions by integrating both local and global features from a U-shape network. Li et al. [18] developed a multi-task architecture to predict salient objects and their contours simultaneously. In [22], Liu et al. employed multiple LSTMs [11] to capture both global and local contexts by scanning over images in four directions. Furthermore, Zhang et al. [47] integrated spatial and channel-wise attention to assist the network to learn more distinctive features. Moreover, [44, 13, 42] introduced various structures to better integrate hierarchical representations.

2.2. High efficiency saliency detection

In the last few years, efficiency becomes an important criterion with increasing attention, multiple approaches have been proposed to improve the processing speed. For example, Chen et al. [3] predicted a global saliency map and recursively utilized it to adjust feature distribution by integrating the reversed saliency maps into learned features. By using limited number of parameters to learn side-output residual features, they achieved good performance with a speed of 35 FPS. Since many methods generate blurred boundaries of saliency objects, Feng et al. [8] developed an Attentive Feedback Module (AFM) to refine the coarse predictions. Furthermore, to underline the object boundary, they used L2 loss for boundary pixels, and employed binary cross-entropy (BCE) loss for the other pixels. Rather than concatenating hierarchical features, Wu et al. [36] only used low-level features to accelerate the process to a speed over 100 FPS. However, lacking of multi-level feature fusion results in inferior performance.

Recently, Liu et al. [21] claimed that features from the top of encoders are gradually diluted by fusing low-level features in existing U-shape networks. Thus, they introduced a global guidance module to alleviate this problem. Since the bottommost features require heavy calculation cost, Wu et al. [37] cut off the skip-connections of these features to accelerate the network to 60 FPS. Furthermore, they employed a cascaded structure and achieved remarkable detection performance.

Contrary to the abovementioned works, we keep our network lightweight by restricting the number of channels to 64 and employing the channel-pooling layers to reduce the computation cost in feature fusion.

2.3. Use of contour information

Since existing methods are struggling with object boundary, contour cue attracts increasing interest in recent years.

First, several works introduced contour into networks by

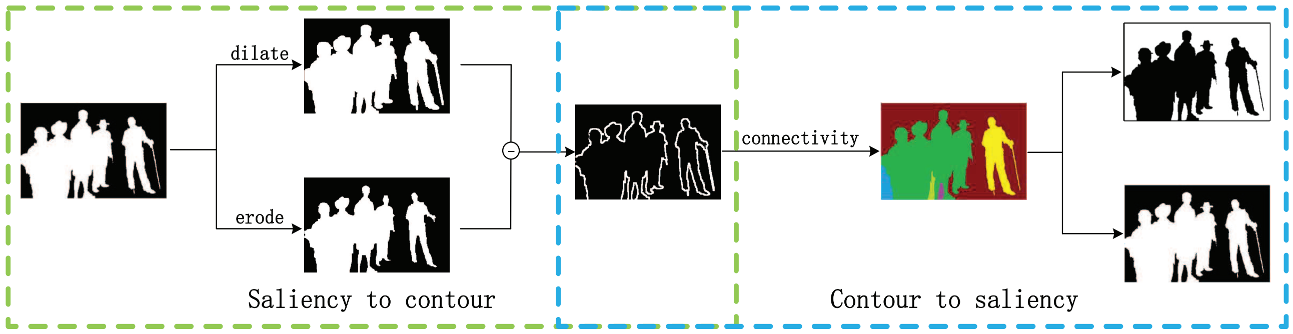

Figure 2. The transformation methods between the saliency maps and the contour maps are shown in green and blue dash boxes respectively. The white and black regions mean foreground and background respectively, while color image is employed to show different regions.

proposing a boundary-aware objective function. Qin et al. [26] proposed a contour-aware objective function for saliency detection, which is a mixture formulation of BCE, structural similarity index (SSIM) and Intersection over Union (IOU). Chen et al. [4] claimed that many boundary pixels are hard examples. To urge the network to pay more attention to these pixels in the training phase, they developed a weighted BCE loss according to the groundtruth contours. Additional hyperparameters are employed to determine the range of hard examples to learn more robust networks.

Second, constructing a boundary-aware network becomes an impressive method in the saliency detection task. Zhao et al. [48] used contour as another supervised signal to guide the learning process of bottom features. Afterward, these features are up-sampled to input size and concatenated with high-level representations. In [35], Salient Edge Detector (SED) was presented to predict salient objects and their contours simultaneously by a residual structure. Furthermore, Li et al. [18] proposed a contour-to-saliency transferring method to integrate the feature from different branches. However, since the intermediate blocks are trained only using the loss backpropagated from the top layers, no restriction is attached to their features to conclude more useful information.

In conclusion, contour-aware objective function ignores the correlation between the salient regions and the contour maps. Besides, constructing boundary-aware networks only utilizes the contour cue to improve the saliency maps. In contrast, we additionally employ the saliency supervision to improve the distinctiveness of contour representation, which in return helps our model learn more powerful saliency representation through our new fusion module.

3. Approach

3.1. Analysis on correlation

We observe that the contour map can be easily generated by calculating the difference between dilated and eroded saliency maps, as shown in green dash box in Figure 2. To obtain a saliency map from the generated contour, we adopt seed filling algorithm [15] to find the region within the closed contour. However, we do not know whether pixels in the closed contour are foreground or not. Therefore, the contour will generate two contrary saliency maps as shown in blue dash box in Figure 2. In summary, the saliency maps and the contour maps are highly associated with each other, except that the saliency maps exactly define background and foreground. These observations motivate us to use two branches for representing saliency and contour respectively and to take their correlation into consideration for improving predictions.

We are awared that some works also use contour cue in saliency detection task. They either regarded bottom features of the predicted contour map as a complementary cue [35, 48] or treated contour pixels different from the other ones [4, 26]. Although these methods can produce good saliency maps, they (1) cannot make sure the learned two branches are complementary to each other and (2) ignored the high correlation between contour and saliency maps.

Contrary to these works, we integrate saliency maps, contour cues, and their correlation into our framework. We not only add such cues into objective function but also fuse features from both branches to exploit the correlation between them. Detail structure of our method is illustrated in the following section.

3.2. The proposed FCF module

As our analysis in the above section, we need two individual branches to represent saliency and contour cues, as well as adding a module to fuse the correlation between these two types of signals. Figure 3 shows two variants of the proposed fusion modules. In both of our modules, two branches are used to represent saliency S_i and contour C_i and additional submodules to explore their correlation. The first proposed module is a straightforward method, referred as naive Feature Correlation Fusion (FCF-a), which uses a fusion block F_i to combine features from two branches. Then the fused feature is concatenated with each branch

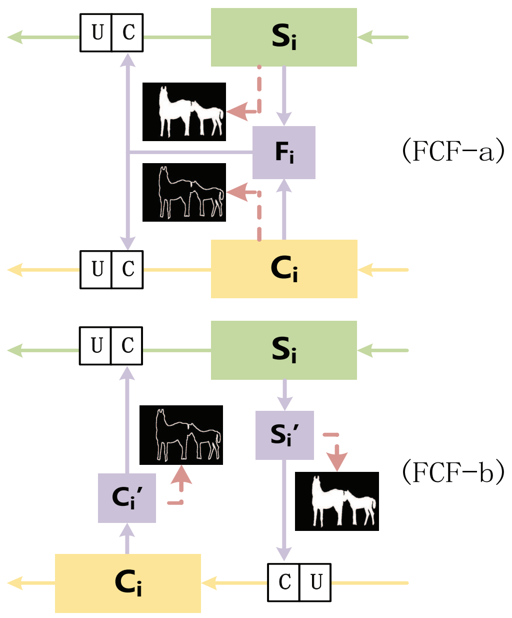

Figure 3. Two variants of our FCF modules for correlation fusion.

as inputs to the next layers. F_i is instantiated with two channel pooling layers and a convolutional layer. It is noteworthy that each branch is supervised by corresponding maps, while no supervision is attached to the common blocks. This simple module cannot guarantee that the fused features can be complementary to each branch. Therefore, it is necessary to design a new fusion method to improve representation power of both branches.

To this end, we develop an interactive Feature Correlation Fusion (FCF-b) module to integrate the correlation. Like the above module, two branches S_i and C_i are adopted. Furthermore, some intermediate connections denoted as $S_i^{'}$ and $C_i^{'}$ directly deliver the features from one branch to another as shown in Figure 3. $S_i^{'}$ and $C_i^{'}$ contain several convolution layers to generate two outputs. One output is used for predicting corresponding maps, while the other one is transmitted into another branch. Unlike FCF-a, supervisions are attached to these connections to ensure the transferred features are related to their original branch. In this way, the fused features are exactly from two cues.

3.3. Overall network architecture

The proposed network in line with the encoder-decoder pattern like most saliency detection models. We provide a detailed introduction in the following.

Feature encoder: Here, standard VGG [29] or ResNet [10], pre-trained on ImageNet [7], are adopted as our feature extractor for fairly compared with other popular saliency detection methods. For VGG network, fully-connected layers are truncated for our lightweight purpose. Both networks contain hierarchical feature maps with progressively decreasing size to well integrate diverse information of images. Those feature maps (outputs before sub-sampling layers) with different sizes are collected as the encoder features, denoted as $E_i, i = 1, 2, 3, 4, 5$. An example based on VGG network is shown in Figure 4.

To reduce feature channels and computation loads, we attach a channel pooling layer on the top of each selected feature map to enable information flow through different channels. Compared with traditional convolution operator, it has much fewer computational costs without any extra learnable parameters. We define channel pooling as:

$$A_i = cp(E_i), \tag{1}$$

$$cp(X) = collect_{j\in[0,m-1]}(max_{k\in[0,\frac{n}{m}-1]}X^{j\times\frac{n}{m}+k}), \tag{2}$$

where j, k are integers and $i \in [1, 5]$ is the index of each feature. In addition, cp denotes the channel pooling operator and $X^{j\times\frac{n}{m}+k}$ indicates the $(j\times\frac{n}{m}+k)$-th channel of the feature map X. Similar with Maxout [9], this layer collects the maximum values over every $\frac{n}{m}$ channels, where n and m are the input and output channels of features respectively and n is divisible by m.

Interactive two-stream decoder: As Figure 4 shown, five features are provided by the encoder, we apply five FCF-b modules to integrate these features correspondingly, forming our interactive two-stream decoder (ITSD). In our framework, all S_i and C_i are cascaded to compose saliency and contour stream. To transmit the learned features between two streams, we develop some intermediate connections that implement the following formulas:

$$S_i' = f_{s_i'}(S_i), P_i^s = cp(S_i'), \tag{3}$$

$$C_i' = f_{c_i'}(C_i), P_i^c = cp(C_i'), \tag{4}$$

where f denotes convolutional operations, and the subscript of f stands for its corresponding branch. Moreover, P_i^S and P_i^C represent task related predictions, where intermediate supervisions are attached.

Because the final goal is to predict the saliency map, we only integrate encoder features into the saliency stream. Our ITSD is formulated with:

$$S_i = f_{s_i}(upsample(concat(S_{i+1}, C_{i+1}', A_i))), \tag{5}$$

$$C_i = f_{c_i}(concat(S_i', upsample(C_{i+1}))), \tag{6}$$

where upsample and concat are the operators of upsampling and concatenation. For the final prediction, all features in the saliency stream are concatenated to balance hierarchical information, which can be formulated as:

$$S_0 = f_{s_0}(concat([upsample(S_i), i = 1, 2, 3, 4, 5])), \tag{7}$$

$$P_0^s = cp(S_0), \tag{8}$$

where all S_i are up-sampled to the input size before concatenation and the final prediction is aggregated by pooling over the concatenated feature.

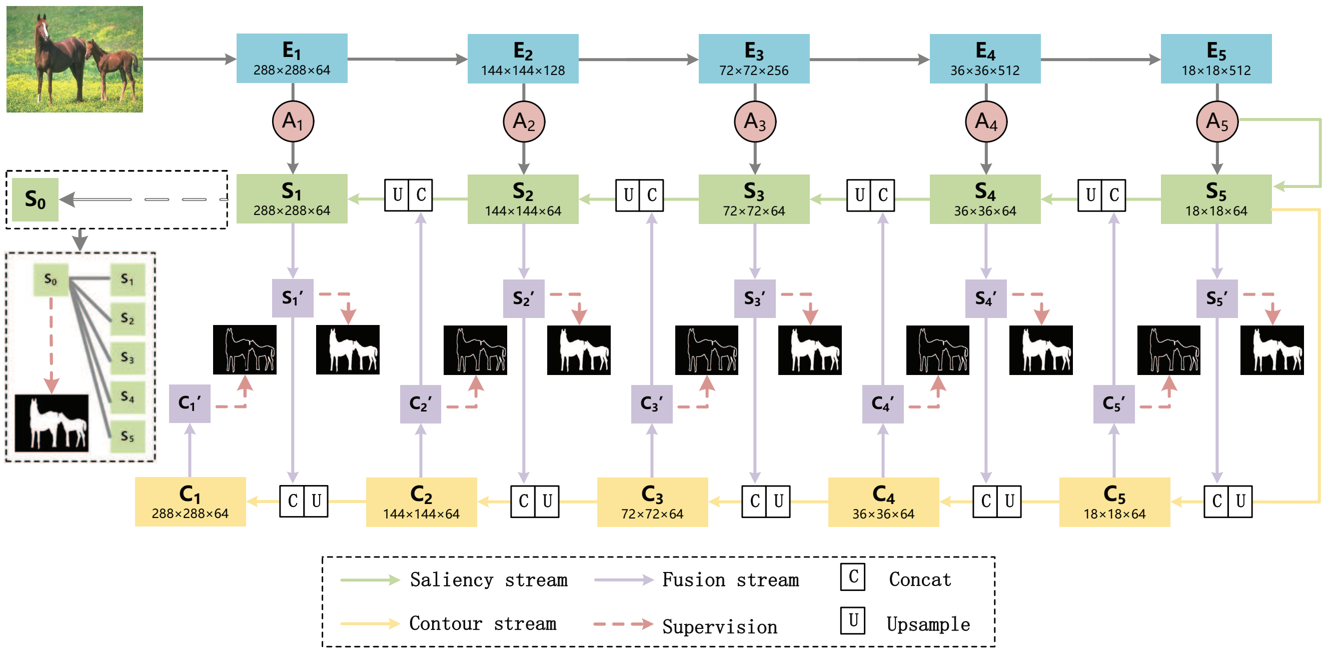

Figure 4. VGG-based network of the proposed model. E_i, A_i, S_i and C_i indicates encoder, embedding block, saliency branch and contour branch respectively. Interactive connections, including $S_i^{'}$ and $C_i^{'}$, are fusion blocks named by their source branch. S_0 upsamples all features in saliency stream to input size for generating the final prediction.

3.4. Objective function

Since our network is based on two-stream framework, we train our model using two losses. For the contour branch, conventional BCE loss is employed due to its robustness in segmentation task:

$$l_{bce}(x, y) = y \log(x) + (1 - y) \log(1 - x) \tag{9}$$

$$L^c(P^c, G^c) = -\frac{1}{n} \sum_{k=1}^{n} l_{bce}(p_k^c, g_k^c) \tag{10}$$

where p_k^c and g_k^c are pixels in the prediction (P^c) and ground truth (G^c) contour maps respectively. k indicates the index of each pixel while n is the number of pixels.

For the saliency branch, it is hard to correctly classify many pixels near to the object boundaries, where those pixels are called hard examples. To improve the resulting map, contour loss [4] increases their weights to promote the network to pay more attention to such pixels:

$$L^s(P^s, G^s, G^c) = -\frac{1}{n} \sum_{k=1}^{n} (g_k^c \times m + 1) l_{bce}(p_k^s, g_k^s) \tag{11}$$

where P^s and G^s are the predicted and ground truth saliency maps respectively. m is the factor for hard examples. All weights plus one to prevent too imbalance ratios between hard and easy examples. However, the definition of hard examples is ambiguous, as illustrated in Figure 5. In another word, we cannot easily determine some pixels near the boundary are hard or not. A simple way is to use some hyperparameters, which are empirical set from extensive validations, to control the degree of proximity.

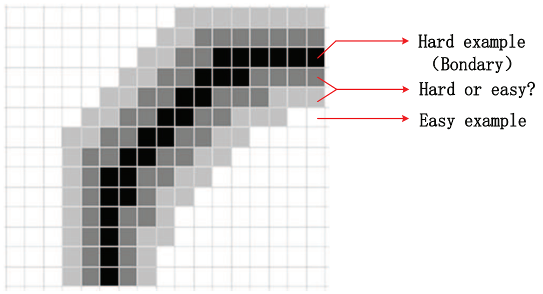

Figure 5. Illustration of hard examples. It is hard to determine some pixels near the boundary are hard examples or not.

In this paper, we provide an alternative method and develop an adaptive contour loss for the saliency branch. Specifically, we find two important properties in the predicted contour map: (1) for negative pixels, higher predicted values mean that the network is hard to distinguish these examples; (2) for positive pixels, higher values should be attended. Based on above observations, we can conclude that the predicted contour map inherently matches the desired weight and use it as our adaptive weight. Thus, the whole adaptive loss function can be obtained:

$$L^s(P^s, P^c, G^s, G^c) = -\frac{1}{n} \sum_{k=1}^{n} (max(p_k^c, g_k^c) \times m + 1) l_{bce}(p_k^s, g_k^s) \tag{12}$$

It is noteworthy that we compute the maximum value over p_l^c and g_l^c to weight the saliency loss. In conclusion, the total loss of our model is:

$$L(P^s, P^c, G^s, G^c) = \sum_{i=0}^{5} L^s(P_i^s, P_i^c, G_i^s, G_i^c) + \lambda \sum_{j=1}^{5} L^c(P_j^c, G_j^c) \tag{13}$$

where P_0^c is substituted by P_1^c and λ provides a balance between contour and saliency losses.

4. Experiments

To validate the proposed method, we conduct a set of experiments on 6 public benchmarks. The DUTS [31] dataset, which contains 10553 images for training (DUTS-TR) and 5019 images for testing (DUTS-TE), is exploited for training and testing respectively. Meanwhile, others datasets, like SOD [25], PASCAL-S [19], ECSSD [40], HKU-IS [17] and DUT-O [41], are only employed as test sets due to their relatively small scales, which contain 300, 850, 1000, 4447 and 5168 images respectively. VGG16 and ResNet50 are employed as the backbone of the proposed network.

For training, we apply random flipping, random cropping and multi-scale training as data augmentation strategy. A GTX1080 Ti GPU is required to train our network with batch size of 8. λ and m are sets to 1 and 4 respectively. Stochastic Gradient Descent (SGD) is used to train our model for 25k iterations in total. The learning rate is set as 0.01 for the first 20k iterations. After that, learning rate is decayed by a factor of 0.1 for the next 5k iterations.

To quantitatively evaluate the performance, F_β-measure [30] and mean absolute error (MAE) are adopted in our experiments. The F_β-measure is a weighted combination of precision and recall value for saliency maps, which can be calculated by:

$$F_\beta = \frac{(1+\beta^2) \times Precision \times Recall}{\beta^2 \times Precision + Recall} \tag{14}$$

where β^2 is set to 0.3 as recommended in [30]. Since different thresholds cause floating numbers of F_β-measure, we use the best score over all thresholds from 0 to 255, named maximum F_β-measure [24, 44]. In addition, MAE measures the pixel-wise average absolute difference between predictions and ground truths:

$$MAE = \frac{1}{n} \sum_{i=1}^{n} |x_i - y_i| \tag{15}$$

where x and y mean prediction and ground truth, n indicates the total number of pixels.

4.1. Main results

We compare our model with 16 existing saliency detection models, including 10 nearly real-time models, such as RFCN [32], Amulet [45], UCF [46], NLDF [24], CKT [18], BMP [44], PCA [22], PAGE [35], DSS [12], EGNet [48] and 6 real-time models, such as RA [3], AFNet [8], BASNet [26], CPD [37], PoolNet [21], SCRN [38]. Quantitative results are shown in Tables 1 & 2.

Among the compared nearly real-time models, EGNet achieves remarkable results on six test sets with a speed of 9 FPS. The proposed network achieves comparable scores with 6× smaller model size and 5× faster speed. Among the compared real-time models, PoolNet provides the best results with a speed of 32 FPS, while our approach outperforms PoolNet in 4 datasets with faster processing speed. Switching to ResNet, our ITSD obtains competitive results, while still running at real-time speed.

Additionally, some visual examples are shown in Figure 6. Unlike other networks usually lose some parts of objects, the integrity of salient objects is well preserved. Generally, our network shows its robustness and effectiveness in processing complicated images.

Furthermore, we compare FLOPs and the number of parameters with other popular methods in Table 2. Input sizes are set according to their publicity code that provides the performance in Table 1. With the least number of parameters and fewer computational cost, our network still provides comparable performance. It clearly demonstrates the efficiency of our ITSD network.

4.2. Correlation analysis

We conduct controlled experiments to prove that the correlation between saliency and contour cues can assist the learning process of each other. First, similar with P_0^s and S_0, additional P_0^c and C_0 are generated from the assemble of C_1 to C_5 in our network. Next, four conditions of intermediate supervision: (1) no supervision, (2) contour-only, (3) saliency-only and (4) both of them, are tested using the maximum F_β-measure scores as shown in Table 3.

Compared with the network trained with no intermediate supervision, adding saliency or contour cue can improve the performance. More importantly, using both two cues can provides further improvement in our experiments. These results clearly demonstrate that each prediction task can benefit from the other supervised signal. Moreover, the combination of these two cues provides the best results.

4.3. Internal comparisons on ITSD modules

Compared with basic U-shape module, the main differences of the proposed network are the contour supervision and the two-stream structure. Therefore, we train four networks to verify the effectiveness of these components by gradually eliminating them. First, we train our ITSD with the proposed FCF-a and FCF-b modules respectively. Second, we remove the intermediate supervision of contour branch from FCF-b, denoted as FCF-NoCS. Al last, a basic

Table 1. Quantitative comparisons of different saliency models on six benchmark datasets in terms of maximum F_β-measure and MAE, which are marked as F^*_β and mae. Red and blue text indicate the best and the second best performance respectively. FPS is tested using public code. PAGE is trained on THUS10K [5]. In ResNet-based comparisons, BasNet uses ResNet34 while others use ResNet50.

Method	FPS	SOD		PASCAL-S		ECSSD		HKU-IS		DUTS-TE		DUT-O	
		F^*_β	mae	F^*_β	mae	F^*_β	mae	F^*_β	mae	F^*_β	mae	F^*_β	mae
VGG-based													
RFCN [32]	9	.807	.166	.850	.132	.898	.095	.898	.080	.783	.090	.738	.095
Amulet [45]	16	.798	.145	.837	.099	.915	.059	.897	.051	.778	.085	.743	.098
UCF [46]	23	.803	.169	.846	.128	.911	.078	.886	.074	.771	.117	.735	.132
NLDF [24]	12	.842	.125	.829	.103	.905	.063	.902	.048	.812	.066	.753	.080
DSS [12]	25	.837	.127	.828	.107	.908	.062	.900	.050	.813	.064	.760	.074
CKT [18]	23	.829	.119	.850	.086	.910	.054	.896	.048	.807	.062	.757	.071
BMP [44]	22	.851	.106	.859	.081	.928	.044	.920	.038	.850	.049	.774	.064
PAGE [35]	25	.796	.110	.835	.078	.931	.042	.930	.037	.838	.051	.791	.066
PCA [22]	5.6	.855	.108	.858	.081	.931	.047	.921	.042	.851	.054	.794	.068
CTLoss [4]	26	.861	.109	.876	.079	.933	.043	.927	.035	.872	.042	.792	.073
EGNet [48]	9	.869	.110	.863	.076	.941	.044	.929	.034	.880	.043	.826	.056
RA [3]	35	.844	.124	.834	.104	.918	.059	.913	.045	.826	.055	.786	.062
AFNet [8]	45	.855	.110	.867	.078	.935	.042	.923	.036	.862	.046	.797	.057
CPD [37]	66	.850	.114	.866	.074	.936	.040	.924	.033	.864	.043	.794	.057
PoolNet [21]	32	.859	.115	.857	.078	.936	.047	.928	.035	.876	.043	.817	.058
ITSD (Ours)	48	.869	.100	.871	.074	.939	.040	.927	.035	.877	.042	.813	.063
ResNet-based													
BasNet [26]	70	.851	.114	.854	.076	.942	.037	.928	.032	.860	.047	.805	.056
CPD [37]	62	.852	.110	.864	.072	.939	.037	.925	.034	.865	.043	.797	.056
PoolNet [21]	18	.867	.100	.863	.075	.940	.042	.934	.032	.886	.040	.830	.055
EGNet [48]	7.8	.890	.097	.869	.074	.943	.041	.937	.031	.893	.039	.842	.052
SCRN [38]	32	.860	.111	.882	.064	.950	.038	.934	.034	.888	.040	.812	.056
ITSD (Ours)	43	.880	.095	.871	.071	.947	.035	.934	.031	.883	.041	.824	.061

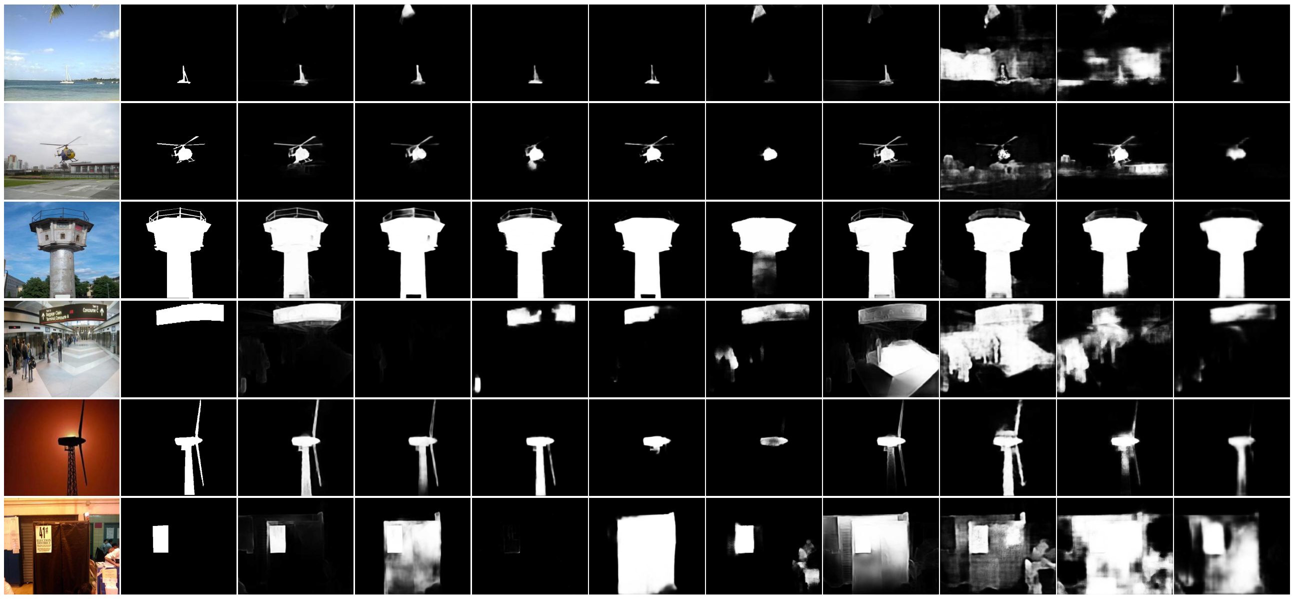
(a) Image (b) GT (c) Ours (d) EgNet (e) Page (f) CPD (g) NLDF (h) CTLoss (i) UCF (j) Amulet (k) SRM

Figure 6. Example images of segmentation results.

Table 2. The number of parameters and FLOPs comparisons of our method with some state-of-the-art networks. All methods adopt VGG as backbone network.

Method	Input size	FLOPs(G)	Params(M)
EGNet [48]	~380 × 320	291.90	108.07
PoolNet [21]	400 × 300	117.10	52.51
PAGE [35]	224 × 224	101.98	47.40
CPD [37]	352 × 352	59.46	29.23
ITSD (Ours)	288 × 288	**57.47**	**17.08**

Table 3. Sal and Con denote saliency and contour supervision respectively, while slc and ctr indicate network predictions.

Supervised		PASCAL-S		DUTS-TE		HKU-IS	
Sal	Con	slc	ctr	slc	ctr	slc	ctr
		.852	.541	.850	.650	.915	.714
	✓	.861	.551	.857	.677	.921	.730
✓		.869	.562	.868	.672	.923	.729
✓	✓	**.871**	**.564**	**.877**	**.683**	**.927**	**.734**

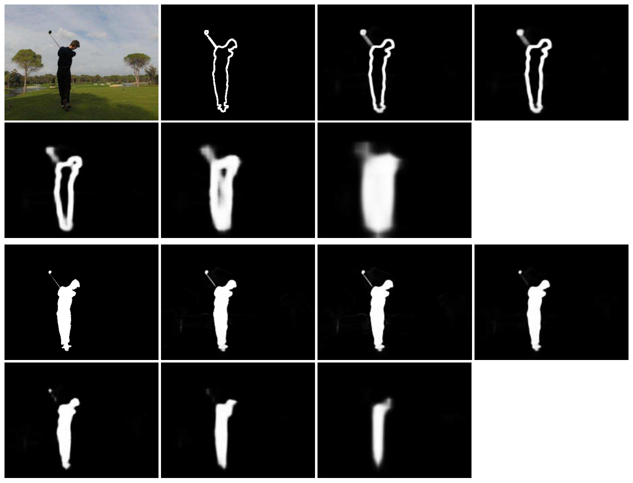

Figure 7. Predicted maps of intermediate layers. The first two rows illustrate the original image, ground truth of contour and five P_i^c (from 1 to 5) respectively. Moreover, the last two rows show ground truth of saliency and six P_i^s (from 0 to 5) respectively.

U-shape network is constructed as the baseline by truncating the whole contour branch in FCF-b module. All results are illustrated in Table 4.

Compared with the U-shape network, better performance of FCF-NoCS demonstrates the importance of contour branch. In addition, comparison between FCF-NoCS and FCF-b validates the importance of contour supervision for saliency network. For the proposed two modules, FCF-b achieves better performance on all tested datasets.

In Figure 7, we visualize one example of intermediate outputs from the proposed network. As one can see, the generated contours and saliency maps show coarse-to-fine predictions along with the decoding process. It clearly indicates that our two-stream decoder can gradually refine the contour and saliency predictions.

Table 4. Controlled experiments on the proposed network. U-shape and FCF-NoCS are two controlled networks, while FCF-a and FCF-b are two variants of the proposed module.

Module	ECSSD		DUTS-TE		HKU-IS	
	F_β	MAE	F_β	MAE	F_β	MAE
U-shape	.919	.052	.842	.062	0.913	.048
FCF-NoCS	.929	.043	.868	.045	0.921	.035
FCF-a	.930	.041	.865	.046	0.919	.036
FCF-b	.939	.040	.877	.042	0.927	.033

Table 5. Experiment results of the proposed network using different objective functions.

Loss	ECSSD		DUTS-TE		HKU-IS	
	F_β	MAE	F_β	MAE	F_β	MAE
F-score	.929	.043	.845	.050	0.915	.045
BCE	.931	.040	.861	.046	0.921	.040
CTLoss	.935	.040	.872	.045	0.925	.036
ACT	.939	.039	.877	.042	0.927	.035

4.4. Ablation studies on objective function

Except for conventional BCE loss, other losses are proposed to train the saliency models recently. To evaluate the effectiveness of these losses, four different losses are employed in the saliency branch to train our model, including BCE loss, F-score loss [49], contour loss (CTLoss) [4] and the proposed ACT loss. Results are shown in Table 5.

F-score loss is struggling with small gradients when accuracy is high, which causes the worst performance in our experiments. Among the compared methods, contour loss shows the best performance. Our proposed method, ACT loss, outperforms CTLoss since it uses the predicted contour map to weight pixels automatically. Furthermore, it is noteworthy that most hyperparameters in the contour loss are abandoned in our ACT loss, which largely reduces the efforts for searching optimal parameters.

5. Conclusion

In this paper, we first observed that saliency and contour maps were highly associated with each other, but had a minor difference in foreground and background definition. Based on our findings, we proposed an Interactive Two-Stream Decoder (ITSD), which consists of two individual branches for representing saliency and contour stream respectively, and a novel Feature Correlation Fusion (FCF) module for their correlation combination. Furthermore, we found the predicted contour maps can be utilized as a weight functon to automatically weight hard examples and thus proposed a new adaptive contour loss (ACT) for training. Extensive experiments well demonstrated the efficiency and effectiveness of the proposed network.

References

[1] Ashwan Abdulmunem, Yu-Kun Lai, and Xianfang Sun. Saliency guided local and global descriptors for effective action recognition. *Computational Visual Media*, 2(1):97–106, 2016.

[2] Ali Borji, Ming-Ming Cheng, Qibin Hou, Huaizu Jiang, and Jia Li. Salient object detection: A survey. *Computational Visual Media*, pages 1–34, 2014.

[3] Shuhan Chen, Xiuli Tan, Ben Wang, and Xuelong Hu. Reverse attention for salient object detection. In *Proceedings of the European Conference on Computer Vision*, pages 234–250, 2018.

[4] Zixuan Chen, Huajun Zhou, Xiaohua Xie, and Jianhuang Lai. Contour loss: Boundary-aware learning for salient object segmentation. *arXiv preprint arXiv:1908.01975*, 2019.

[5] Ming-Ming Cheng, Niloy J Mitra, Xiaolei Huang, Philip HS Torr, and Shi-Min Hu. Global contrast based salient region detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 37(3):569–582, 2014.

[6] Runmin Cong, Jianjun Lei, Huazhu Fu, Ming-Ming Cheng, Weisi Lin, and Qingming Huang. Review of visual saliency detection with comprehensive information. *IEEE Transactions on Circuits and Systems for Video Technology*, 29(10):2941–2959, 2018.

[7] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 248–255, 2009.

[8] Mengyang Feng, Huchuan Lu, and Errui Ding. Attentive feedback network for boundary-aware salient object detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1623–1632, 2019.

[9] Ian J Goodfellow, David Warde-Farley, Mehdi Mirza, Aaron Courville, and Yoshua Bengio. Maxout networks. *arXiv preprint arXiv:1302.4389*, 2013.

[10] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 770–778, 2016.

[11] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[12] Qibin Hou, Ming-Ming Cheng, Xiaowei Hu, Ali Borji, Zhuowen Tu, and Philip Torr. Deeply supervised salient object detection with short connections. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 41(4):815–828, 2019.

[13] Xiaowei Hu, Chi-Wing Fu, Lei Zhu, and Pheng-Ann Heng. Sac-net: Spatial attenuation context for salient object detection. *arXiv preprint arXiv:1903.10152*, 2019.

[14] Zilong Huang, Xinggang Wang, Lichao Huang, Chang Huang, Yunchao Wei, and Wenyu Liu. Ccnet: Criss-cross attention for semantic segmentation. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 603–612, 2019.

[15] Shyan-Bin Jou and Ming-Dar Tsai. A fast 3d seed-filling algorithm. *The Visual Computer*, 19(4):243–251, 2003.

[16] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems*, pages 1097–1105, 2012.

[17] Guanbin Li and Yizhou Yu. Visual saliency based on multiscale deep features. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015.

[18] Xin Li, Fan Yang, Hong Cheng, Wei Liu, and Dinggang Shen. Contour knowledge transfer for salient object detection. In *Proceedings of the European Conference on Computer Vision*, pages 355–370, 2018.

[19] Yin Li, Xiaodi Hou, Christof Koch, James M. Rehg, and Alan L. Yuille. The secrets of salient object segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2014.

[20] Pengpeng Liang, Yu Pang, Chunyuan Liao, Xue Mei, and Haibin Ling. Adaptive objectness for object tracking. *IEEE Signal Processing Letters*, 23(7):949–953, 2016.

[21] Jiang-Jiang Liu, Qibin Hou, Ming-Ming Cheng, Jiashi Feng, and Jianmin Jiang. A simple pooling-based design for real-time salient object detection. *arXiv preprint arXiv:1904.09569*, 2019.

[22] Nian Liu, Junwei Han, and Ming-Hsuan Yang. Picanet: Learning pixel-wise contextual attention for saliency detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3089–3098, 2018.

[23] Yun Liu, Ming-Ming Cheng, Xiaowei Hu, Jia-Wang Bian, Le Zhang, Xiang Bai, and Jinhui Tang. Richer convolutional features for edge detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 41(8):1939 – 1946, 2019.

[24] Zhiming Luo, Akshaya Mishra, Andrew Achkar, Justin Eichel, Shaozi Li, and Pierre-Marc Jodoin. Non-local deep features for salient object detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6609–6617, 2017.

[25] Vida Movahedi and James H Elder. Design and perceptual validation of performance measures for salient object segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, 2010.

[26] Xuebin Qin, Zichen Zhang, Chenyang Huang, Chao Gao, Masood Dehghan, and Martin Jagersand. Basnet: Boundary-aware salient object detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 7479–7489, 2019.

[27] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*, pages 234–241. Springer, 2015.

[28] Ueli Rutishauser, Dirk Walther, Christof Koch, and Pietro Perona. Is bottom-up attention useful for object recognition? In *Proceedings of the 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, volume 2, pages II–II, 2004.

[29] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[30] S. Susstrunk, R. Achanta, F. Estrada, and S. Hemami. Frequency-tuned salient region detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2009.

[31] Lijun Wang, Huchuan Lu, Yifan Wang, Mengyang Feng, Dong Wang, Baocai Yin, and Xiang Ruan. Learning to detect salient objects with image-level supervision. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017.

[32] Linzhao Wang, Lijun Wang, Huchuan Lu, Pingping Zhang, and Xiang Ruan. Saliency detection with recurrent fully convolutional networks. In *European Conference on Computer Vision*, 2016.

[33] Wenguan Wang, Qiuxia Lai, Huazhu Fu, Jianbing Shen, and Haibin Ling. Salient object detection in the deep learning era: An in-depth survey. *arXiv preprint arXiv:1904.09146*, 2019.

[34] Wenguan Wang, Jianbing Shen, Jianwen Xie, Ming-Ming Cheng, Haibin Ling, and Ali Borji. Revisiting video saliency prediction in the deep learning era. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2019.

[35] Wenguan Wang, Shuyang Zhao, Jianbing Shen, Steven CH Hoi, and Ali Borji. Salient object detection with pyramid attention and salient edges. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1448–1457, 2019.

[36] Runmin Wu, Mengyang Feng, Wenlong Guan, Dong Wang, Huchuan Lu, and Errui Ding. A mutual learning method for salient object detection with intertwined multi-supervision. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 8150–8159, 2019.

[37] Zhe Wu, Li Su, and Qingming Huang. Cascaded partial decoder for fast and accurate salient object detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3907–3916, 2019.

[38] Zhe Wu, Li Su, and Qingming Huang. Stacked cross refinement network for edge-aware salient object detection. In *The IEEE International Conference on Computer Vision*, October 2019.

[39] Wenqiang Xu, Haiyang Wang, Fubo Qi, and Cewu Lu. Explicit shape encoding for real-time instance segmentation. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 5168–5177, 2019.

[40] Qiong Yan, Li Xu, Jianping Shi, and Jiaya Jia. Hierarchical saliency detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2013.

[41] Chuan Yang, Lihe Zhang, Huchuan Lu, Xiang Ruan, and Ming-Hsuan Yang. Saliency detection via graph-based manifold ranking. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2013.

[42] Yu Zeng, Huchuan Lu, Lihe Zhang, Mengyang Feng, and Ali Borji. Learning to promote saliency detectors. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1644–1653, 2018.

[43] Fan Zhang, Yanqin Chen, Zhihang Li, Zhibin Hong, Jingtuo Liu, Feifei Ma, Junyu Han, and Errui Ding. Acfnet: Attentional class feature network for semantic segmentation. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 6798–6807, 2019.

[44] Lu Zhang, Ju Dai, Huchuan Lu, You He, and Gang Wang. A bi-directional message passing model for salient object detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1741–1750, 2018.

[45] Pingping Zhang, Dong Wang, Huchuan Lu, Hongyu Wang, and Xiang Ruan. Amulet: Aggregating multi-level convolutional features for salient object detection. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 202–211, 2017.

[46] Pingping Zhang, Dong Wang, Huchuan Lu, Hongyu Wang, and Baocai Yin. Learning uncertain convolutional features for accurate saliency detection. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 212–221, 2017.

[47] Xiaoning Zhang, Tiantian Wang, Jinqing Qi, Huchuan Lu, and Gang Wang. Progressive attention guided recurrent network for salient object detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 714–722, 2018.

[48] Jia-Xing Zhao, Jiangjiang Liu, Den-Ping Fan, Yang Cao, Jufeng Yang, and Ming-Ming Cheng. Egnet: Edge guidance network for salient object detection. *arXiv preprint arXiv:1908.08297*, 2019.

[49] Kai Zhao, Shanghua Gao, Qibin Hou, Dan-Dan Li, and Ming-Ming Cheng. Optimizing the f-measure for threshold-free salient object detection. *arXiv preprint arXiv:1805.07567*, 2018.