

Learning to Promote Saliency Detectors

Yu Zeng¹, Huchuan Lu¹, Lihe Zhang¹, Mengyang Feng¹, Ali Borji²

¹ Dalian University of Technology, China

² University of Central Florida, USA

zengyu@mail.dlut.edu.cn, lhchuan@dlut.edu.cn, zhanglihe@dlut.edu.cn,
mengyangfeng@gmail.com, aliborji@gmail.com

Abstract

The categories and appearance of salient objects vary from image to image, therefore, saliency detection is an image-specific task. Due to lack of large-scale saliency training data, using deep neural networks (DNNs) with pre-training is difficult to precisely capture the image-specific saliency cues. To solve this issue, we formulate a zero-shot learning problem to promote existing saliency detectors. Concretely, a DNN is trained as an embedding function to map pixels and the attributes of the salient/background regions of an image into the same metric space, in which an image-specific classifier is learned to classify the pixels. Since the image-specific task is performed by the classifier, the DNN embedding effectively plays the role of a general feature extractor. Compared with transferring the learning to a new recognition task using limited data, this formulation makes the DNN learn more effectively from small data. Extensive experiments on five data sets show that our method significantly improves accuracy of existing methods and compares favorably against state-of-the-art approaches.

1. Introduction

Detecting salient objects or regions of an image, i.e. saliency detection, is useful for many computer vision tasks. As a preprocessing step, saliency detection is appealing for many practical applications, such as content-aware video compression [37], image resizing [2], and image retrieval [10]. A plethora of saliency models have been proposed in the past two decades to locate conspicuous image regions [4, 6, 5]. Although much effort has been devoted and significant progress has been made, saliency detection remains a challenging open problem.

Conventional saliency detection methods usually utilize low-level features and heuristic priors which are not robust enough to discover salient objects in complex scenes, neither are capable of capturing semantic objects. Deep neural

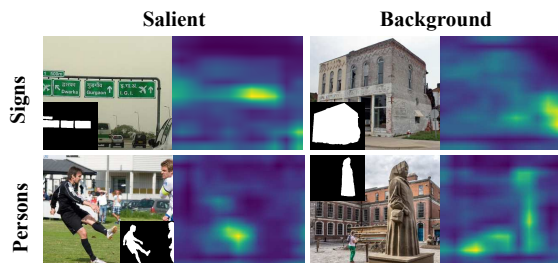


Figure 1. Images and the corresponding feature maps from the last convolution layer of VGG16 [25]. The small binary mask in each image indicates the salient object of this image.

networks (DNNs) have been used to remedy the drawbacks of conventional methods. They can learn high-level semantic features from training samples, thus are more effective in locating semantically salient regions, yielding more accurate results in complex scenes.

DNNs usually need to be trained on a large dataset, while training data for saliency detection is very limited. This issue is generally solved by pre-training on a large dataset for other tasks, such as image classification, which easily leads to several problems. First, saliency detection is an image-specific task, and labels should be assigned to pixels depending on the image content. However, features produced by pre-trained feature extractors are supposed to work for all images. For example, signs and persons are salient objects in the first column of Figure 1, while they belong to the background in the second column. However, the regions of signs and persons are indiscriminately highlighted in the feature maps in the two columns. With this kind of feature extractor, the prediction model might be enforced to learn to map similar features into opposite labels, which is difficult for small training dataset. Second, categories and appearance of salient objects vary from image to image, while small training data is not enough to capture the diversity. For example, the six salient objects shown in Figure 1 come from six different categories and differ wildly in their appearance. Consequently, it might be hard to learn a unified detector to handle all varieties of salient objects.

Considering the large diversity of salient objects, we avoid training a deep neural network (DNN) that directly maps images into labels. Instead, we train a DNN as an embedding function to map pixels and the attributes of the salient/background regions into a metric space. The attributes of the salient/background regions are mapped as anchors in the metric space. Then, a nearest neighbor (NN) classifier is constructed in this space, which assigns each pixel with the label of its nearest anchor. As a non-parametric model, the NN classifier can adapt well to new data and handle the diversity of salient objects. Additionally, since the classification task is performed by the NN classifier, the goal of the DNN is turned to learning a general mapping from the attributes of the salient/background regions to anchors in the embedding space. Compared with directly learning to detect diverse salient objects, this would be easier for the network to learn on limited data.

Concretely, we show the pipeline of our proposed method in Figure 2. During training, the DNN is provided with the true salient and background regions, of which the label of a few randomly selected pixels are flipped, to produce anchors. The output of the NN classifier constitutes a saliency map. The DNN can be trained end-to-end supervised by the loss between this saliency map and the ground truth. When testing on an image, the saliency map of each image is obtained as in training, but using approximate salient/background regions detected by an existing method. Although the approximate salient/background region is not completely correct, it is often with similar attributes to the true salient/background region. Thus, the corresponding embedding vectors (*i.e.* anchors) would be close to the ones of the true salient/background regions. Further, to produce better results, we propose an iterative testing scheme. The result of the NN classifier is utilized to revise anchors, yielding increasingly more accurate results.

Our method can be viewed as a zero-shot learning problem, in which the approximate salient/background regions detected by an existing method provide attributes for unseen salient objects, and the model learns from the training data to learn an image-specific classifier from the attributes to classify pixels of this image. Extensive experiments on five data sets show that our method can significantly improve accuracy of existing methods and compares favorably against state-of-the-art approaches.

2. Related works

Generally, saliency detection methods can be categorized into two streams: top-down and bottom-up saliency. Since our work addresses bottom-up saliency, here we mainly review recent works on bottom-up saliency, meanwhile shortly mention top-down saliency. We also explore the relation between our proposed method and top-down saliency.

Bottom-up (BU) saliency is stimuli-driven, where saliency is derived from contrast among visual stimuli. Conventional bottom-up saliency detection methods often utilize low-level features and heuristic priors. Jiang *et al.* [12] formulate saliency detection via an absorbing Markov chain on an image graph model, where saliency of each region is defined as its absorbed time from boundary nodes. Yang *et al.* [32] rank the similarity of the image regions with foreground cues or background cues via graph-based manifold ranking. Since the conventional methods are not robust in complex scenes neither capable of capturing semantic objects, deep neural networks (DNNs) are introduced to overcome these drawbacks. Li *et al.* [16] train CNNs with fully connected layers to predict saliency value of each superpixel, and to enhance the spatial coherence of their saliency results using a refinement method. Li *et al.* [18] propose a FCN trained under the multi-task learning framework for saliency detection. Zhang *et al.* [34] present a generic framework to aggregate multi-level convolutional features for saliency detection. Although the proposed method is also based on DNNs, the main difference between ours and these methods is that they learn a general model that directly maps images to labels, while our method learns a general embedding function as well as an image-specific NN classifier.

Top-down (TD) saliency aims at finding salient regions specified by a task, and is usually formulated as a supervised learning problem. Yang and Yang [33] propose a supervised top-down saliency model that jointly learns a Conditional Random Field (CRF) and a discriminative dictionary. Gao *et al.* [9] introduced a top-down saliency algorithm by selecting discriminant features from a pre-defined filter bank.

Integration of TD and BU saliency has been exploited by some methods. For instance, Borji [3] combines low-level features and saliency maps of previous bottom-up models with top-down cognitive visual features to predict fixations. Tong *et al.* [26] proposed a top-down learning approach where the algorithm is bootstrapped with training samples generated using a bottom-up model to exploit the strengths of both bottom-up contrast-based saliency models and top-down learning methods. Our method also can be viewed as an integration of TD and BU saliency. Although both our method and the method of Tong *et al.* [26] formulate the problem as top-down saliency detection specified by initial saliency maps, there are certain difference between the two. First, Tong’s method trains a strong model via bootstrap learning with training samples generated by a weak model. In contrast, our method maps pixels and the approximate salient/background regions into a learned metric space, which is related to zero-shot learning. Second, thanks to deep learning, our method is capable of capturing semantically salient regions and does well on complex

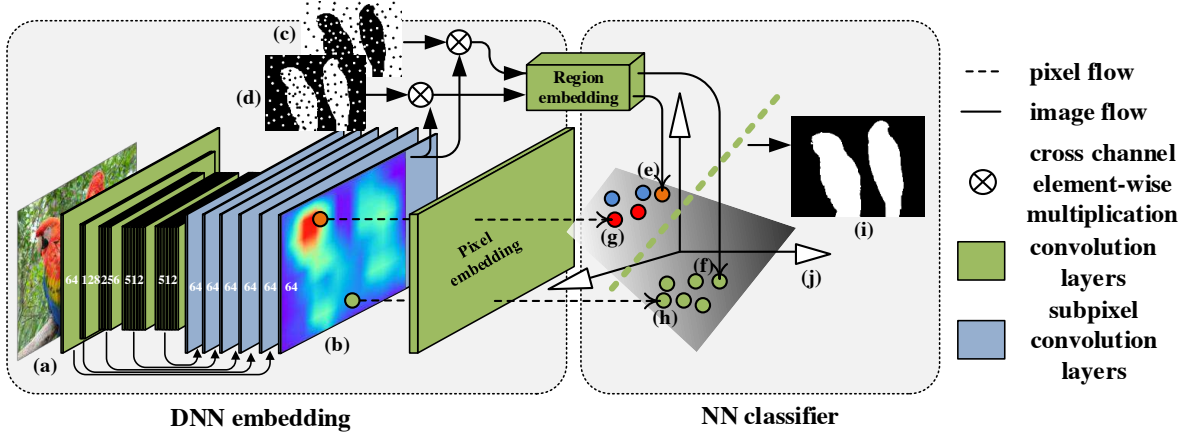


Figure 2. The pipeline of the proposed method. The input image (a) is first passed through our revised VGG network, resulting in an 512 channel feature map (b) of the same size as the input image. Each pixel is mapped to vectors *e.g.*, (g) and (h) in the learned metric space (j). Salient and background regions is also mapped to vectors *i.e.* anchors in the learned metric space. For instance, (e) and (f) are salient and background anchors of this image respectively. During training, the salient and background pixels for producing anchors are selected using a randomly flipped ground truth ((d) and (e) in the figure), see Sec.3.1. A nearest neighbor classifier is built that classifies each pixel based on its distance to the anchors (see Eqn.3). Classification results of all pixels constitute a saliency map (i), of which loss between the ground truth is used to supervise the network. During testing, the anchors are firstly produced according to an initial saliency map, here (e) is the initial saliency map. Given anchors, the nearest neighbor classifier can produce a new saliency map (i), which is utilized to revise the initial map as in Eqn.3. Then the revised map is used to produce new approximation to the anchors. Iterating the testing process would result in an increasingly more accurate result.

scenes, while Tong’s method uses hand-crafted features and heuristic priors, which are less robust, Third, our method produces pixel-level results, while Tong’s method computes saliency value of each image region to assemble a saliency map, which tends to be coarser.

3. The Proposed Method

Our method consists of three components: 1) a DNN as an embedding function *i.e.* the anchor network, that maps pixels and regions of the input image into a learned metric space, 2) a nearest neighbor (NN) classifier in the embedding space learned specifically for this image to classify its pixels, and 3) an iterative testing scheme that utilizes the result of the NN classifier to revise anchors, yielding increasingly more accurate results.

3.1. The anchor network

Let \mathbf{x}_{mn} denote a pixel of an image X_m . Each image consists a salient and a background region, *i.e.* $X_m = \mathcal{C}_{m1} \cup \mathcal{C}_{m2}$. Each pixel of an image either belongs to salient or background regions, denoted as $n \in \mathcal{C}_{mk}, k = 1, 2$, respectively. We use an embedding function modeled by a DNN ϕ with parameter θ , to map each pixel to a vector in a D-dimensional space:

$$\phi_{mn} = \phi(\mathbf{x}_{mn}; \theta), \quad (1)$$

where ϕ_{mn} is the embedding vector to the corresponding pixel \mathbf{x}_{mn} .

The salient or background region \mathcal{C}_{mk} is also mapped into vectors in D -dimensional metric space by a DNN ψ with parameter η :

$$\mu_{mk} = \psi(\mathcal{C}_{mk}; \eta), \quad (2)$$

in which μ_{mk} is the mapping of the salient or background region, *i.e.* anchors.

We assume that in the embedding space, all pixels of an image cluster around the corresponding anchors of this image. Then a nearest neighbor classifier can be built specifically for this image by classifying each pixel according to its nearest anchor. The probability of a pixel \mathbf{x}_{mn} of image X_m belonging to \mathcal{C}_{mk} can be given by the softmax over its distance to the anchors:

$$p(\mathcal{C}_{mk}|\mathbf{x}_{mn}) = \frac{\exp\{-d(\phi_{mn}, \mu_{mk})\}}{\sum_j \exp\{-d(\phi_{mn}, \mu_{mj})\}}, \quad (3)$$

where ϕ_{mn} and μ_{mk} are the vectors of pixel \mathbf{x}_{mn} and the salient / background anchor given by Eqn.1 and 2. $d(\cdot)$ denotes Euclidean distance.

The CNN embeddings can be trained using a gradient-based optimization algorithm through maximizing the log likelihood with respect to θ and η on the training set:

$$\mathcal{L} = \sum_{m,n} t_{mn} \log p(\mathcal{C}_{m1}|\mathbf{x}_{mn}) + (1-t_{mn}) \log p(\mathcal{C}_{m2}|\mathbf{x}_{mn}), \quad (4)$$

where t_{mn} is the label of pixel \mathbf{x}_{mn} . $t_{mn} = 1$ when $\mathbf{x}_{mn} \in \mathcal{C}_1$, *i.e.* salient and $t_{mn} = 0$ when $\mathbf{x}_{mn} \in \mathcal{C}_2$, *i.e.* background.

In practice, the ground-truth will not be available during testing, and the anchors are produced according to a prior saliency map, which is inaccurate. Therefore, to match training and testing conditions, during training we randomly flip the label of each pixel with probability p when producing the anchors using Eqn.2. In addition, this random flipping also increases diversity of training samples, thus helping reduce overfitting. We explain the training process of the anchor network in Alg.1. Here, \mathcal{L}_m denotes the log likelihood on the image X_m .

Algorithm 1: Training the anchor network.

Input : Training set $\{(X_m, \mathbf{t}_m)\}$,
in which $t_{mn} = 1$ indicates $\mathbf{x}_{mn} \in \mathcal{C}_{m1}$,
and $t_{mn} = 0$ otherwise.
Output: CNN embedding $\phi(\cdot; \theta)$ and $\psi(\cdot; \eta)$

- 1 **for** training iterations **do**
- 2 Sample a pair of training image and ground truth map (X_m, \mathbf{t}_m) from the training set.
- 3 Randomly flip the elements in \mathbf{t}_m with probability p .
- 4 Compute the embedding vector ϕ_{mn} of each pixel given by Eqn.1 and produce anchors μ_{mk} as in Eqn.2.
- 5 Compute gradient of log likelihood \mathcal{L}_m on this image with respect to θ and η .
- 6 Update θ and η according to $\nabla_{\theta, \eta} \mathcal{L}_m$ using a gradient based optimization method.
- 7 **end**

3.2. Iterative testing scheme

In the testing phase, since the ground-truth is unknown, it is not possible to obtain precise salient and background regions to produce anchors as in the training time. Therefore, we produce anchors using approximate salient/background regions \hat{C}_{mk} selected according to the saliency map $Y_m^{(0)}$ of an existing method. An iterative testing scheme is proposed to gradually revise the anchors using the result of the NN classifier.

In the t -th iteration ($t > 0$), the anchors are generated according to salient/background region \hat{C}_{mk} selected by the prior saliency map $Y_m^{(t)}$. Given the anchors, we use the nearest neighbor classifier as in Eqn.3 to compute the probability of each pixel belonging to salient regions, *i.e.* saliency value, constructing another saliency map $Z_m^{(t)}$. Then, the prior saliency map is updated with

$$Y_m^{(t+1)} = \frac{t}{t+1} Y_m^{(t)} + \frac{1}{t+1} Z_m^{(t)}, \quad (5)$$

where $Y_m^{(t+1)}$ is the prior saliency map which will be used for selecting salient and background regions in the next it-

eration. This means that the prior map is updated to a weighted sum of itself and the new result. After the first iteration, the prior map is completely replaced by the new result. The weight of the new result decreases with iterating, which insures stability of the iteration process. The testing algorithm of the proposed method is shown in Alg.2.

Figure 3 shows the process of the initial maps being promoted by the proposed method. Although the initial saliency map may not precisely separate the foreground and background, it can often partially separate them, and thus can provide information regarding categories and appearance of salient objects in the image. For instance, in the first image of Figure 3, though only a small part of the foreground is highlighted, the initial map can tell us that the foreground may be a gorilla, and the background contains a piece of green. Then, its selected foreground / background regions should be similar to the true foreground / background regions, leading to the corresponding anchors close to the true ones in the learned metric space. Thereby the nearest neighbor classification given by Eqn.3 can produce a good result. As the iterations progress, the approximate anchors gradually approach to the true ones, which would result in a better result. This, in turn could provide an increasingly accurate approximation to the anchors, and thus a more accurate result. As shown in Figure 3, the initial maps are not appealing, while the modified maps by our method look much better.

Algorithm 2: Testing algorithm of the proposed method.

Input : The input image X , the initial saliency map $Y^{(0)}$, the number of iterations T .
Output: The promoted saliency map $Y^{(T)}$.

- 1 Compute the embedding vector ϕ_n of each pixel \mathbf{x}_n of X . **for** $t \in \{1, \dots, T\}$ **do**
- 2 Select the approximate salient \hat{C}_1 and background region \hat{C}_2 according to $Y^{(t)}$.
- 3 Produce the approximate anchor $\hat{\mu}_k = \psi(\hat{C}_k; \eta)$, $k = 1, 2$.
- 4 Compute saliency value of each pixel according to Eqn.3 to constitute another saliency map $Z_m^{(t)}$.
- 5 Update the prior saliency map:
 $Y^{(t+1)} \leftarrow \frac{t}{t+1} Y^{(t)} + \frac{1}{t+1} Z^{(t)}$
- 6 **end**

It is known that DNNs, which typically consist of many parameters, have to be trained on large datasets to obtain good performance. For tasks where training data is scarce, such as saliency detection, revising a DNN that has been pre-trained on image classification datasets is the most viable option. Therefore, we also adopt a pre-trained DNN for our purpose rather than training a DNN from scratch. We modify the VGG16 [25] network, pre-trained on the Im-

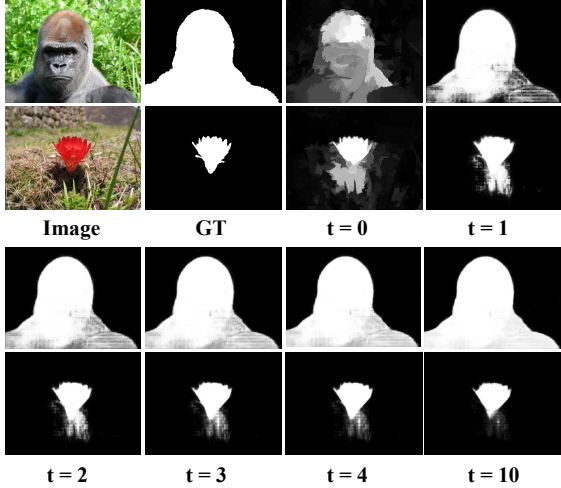


Figure 3. The process of the initial maps being promoted by the proposed method.

ageNet [7] dataset, into the pixel embedding $\phi(\cdot, \theta)$ and region embedding $\psi(\cdot, \eta)$. Since the DNN serves as an embedding instead of a classifier in the proposed method, we remove all the fully connected layers of VGG, and only retain its feature extractor component (VGG feature extractor). The VGG feature extractor consists of 5 convolution blocks, each of which contains several convolution and non-linear layers, as well as a pooling layer. We show the network architecture and the overall structure of the proposed method in Figure 2, and describe the details in the next two subsections. In the figures and the text of this section, nonlinearity layers and batch-normalization layers are omitted to avoid clutter. The combination of a convolution/fully connected layer, a batch-normalization layer and a ReLU nonlinear Convolution layers are referred to as a convolution/fully connected layers in this section.

3.3. Pixel embedding

Although effective in extracting hierarchical features, VGG feature extractor makes the feature maps smaller than the input image. This is not desirable for our method, because in order to map each pixel of the input image to a vector in the learned metric space, the embedding CNN should produce feature maps of the same resolution as the input image. We adopt two strategies to obtain larger feature maps: 1) remove the pooling layers of the last two convolution blocks and use dilated convolutions in these blocks to maintain receptive field of the convolution filters, and 2) append a subpixel convolution layer after each convolution block of the VGG feature extractor to upsample the feature maps of each convolution blocks to the input image size. Subpixel convolution is an upsampling strategy originally proposed in [24] for image super-resolution. To produce a C -channel tensor of N times the input size, the subpixel convolution firstly performs convolution on the feature map

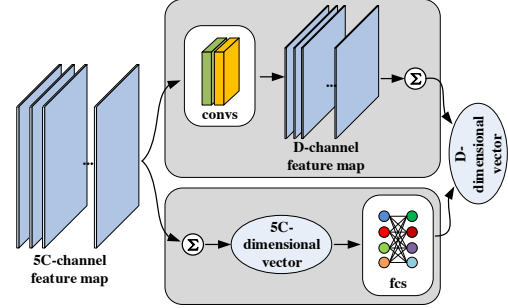


Figure 4. Two different structures of the region embedding. The Σ symbol denotes averaging over pixels of the region. Top and bottom streams indicate Conv-based and FC-based region embedding respectively.

to get a $N^2 \times C$ -channel tensor of the input size. Then, the elements of the $N^2 \times C$ -channel tensor are rearranged into a C -channel output tensor of N times the size of the input tensor.

Five C -channel feature maps can be produced through adding a subpixel convolution layer after each of the five convolution blocks. Then the five C -channel feature maps are cascaded into a $5C$ -channel feature map. Directly using the features of this $5C$ -channel feature map to represent each pixel is not the best option since features of different convolution blocks are in different ranges. To solve this, we add two extra convolution layers after the subpixel convolution layers, to convert the $5C$ -channel feature maps into a D -channel feature map, in which each pixel corresponds to a D -dimensional vector. In our implementation, we set C to 64 and D to 512.

3.4. Region embedding

For simplicity, we let the pixel embedding $\phi(\cdot, \theta)$ and the region embedding $\psi(\cdot, \eta)$ share the common feature extractor and subpixel convolution upsample layers. New layers are appended after the subpixel convolution layers to map the $5C$ -channel feature map of an image region to a D -dimensional vector.

As shown in Figure 4, we consider two different structures of the region embedding: Conv-based and FC-based region embedding. In the Conv-based region embedding, the $5C$ -channel feature map of an image region is passed into convolution layers, resulting in a D -channel feature map. Then the D -dimensional embedding vector is given by averaging the D -channel feature map over pixels. The FC-based region embedding uses fully connected layers to map the average over pixels of the $5C$ -channel feature map into a D -dimensional vector.

4. Experiments

4.1. Datasets

We apply our method to five benchmark datasets to evaluate its performance. Details of these datasets are as fol-

lows.

ECSSD [31] contains 1000 natural images with multiple objects of different sizes. Some of the images come from the challenging Berkeley-300 dataset.

PASCAL-S [19] stems from the validation set of PASCAL VOC2010 [8] segmentation challenge and contains 850 natural images.

HKU-IS [16] has 4447 images with high-quality pixel-wise annotations. Images in this dataset are chosen to include multiple disconnected objects or objects touching the image boundary.

SOD [31] has 300 images, and was originally designed for image segmentation. Pixel-wise annotations of salient objects were generated by [13]. This dataset is challenging since many images contain multiple objects either with low contrast or touching the image boundary.

DUTS [27] is a large scale dataset containing 10533 training images and 5019 test images. All the training images are collected from the ImageNet DET training/val sets [7], while test images are collected from the ImageNet DET test set and the SUN dataset [30]. Accurate pixel-level ground truths are provided.

4.2. Evaluation metrics

We employ Precision-Recall curve, F-measure curve, F-measure score and MAE score to quantitatively evaluate the performance of the proposed method and compare with other methods.

The precision of a binary map is defined as the ratio of the number of salient pixels it correctly labels, to all salient pixels in this binary map. The recall value is the ratio of the number of correctly labeled salient pixels to all salient pixels in the ground-truth map:

$$\text{precision} = \frac{|TS \cap DS|}{|DS|}, \quad \text{recall} = \frac{|TS \cap DS|}{|TS|}, \quad (6)$$

in which TS denotes true salient pixels, DS denotes detected salient pixels by the binary map, and $|\cdot|$ denotes cardinality of a set.

The F-measure, denoted as F_β , is an overall performance indicator computed by the weighted harmonic of precision and recall:

$$F_\beta = \frac{(1 + \beta^2) \cdot \text{precision} \cdot \text{recall}}{\beta^2 \cdot \text{precision} + \text{recall}}, \quad (7)$$

where β^2 is set to 0.3 as suggested in [1] to emphasize the precision.

Given a saliency map whose intensities are in the range of 0 and 1, a series of binary maps can be produced by thresholding the saliency map with different values in [0, 1]. Precision and recall values of these binary maps can be computed according to Eqn. 6. F-measure can be computed

according to Eqn. 7. Plotting the (precision, recall) pairs of all the binary maps results in the precision-recall curve, and plotting the (F-measure, threshold) pairs results in the F-measure curve.

Also as suggested in [1], we use twice the mean value of the saliency maps as the threshold to generate binary maps for computing the F-measure. Notice that some works have reported slightly different F-measures using different thresholds. But as far as we know, twice the mean value is the most commonly used threshold.

As complementary to PR curves, mean absolute error (MAE) is used to quantitatively measure the average difference between the saliency map S and the ground truth map G :

$$\text{MAE} = \frac{1}{H} \sum_{i=1}^H |S_i - G_i|.$$

MAE indicates how similar a saliency map is compared to the ground truth. It is widely used in different pixel-level prediction tasks such as semantic segmentation and image cropping [22].

4.3. Implementation details

Our method is implemented in Python with the PyTorch¹ toolbox. We train and test our model on a PC with a 3.6GHz CPU, 32GB RAM and a GTX 1080 GPU.

We train our model on the training set of DUTS dataset. As in [20], we augment the training data by horizontal flipping and cropping the images to reduce overfitting. The probability p of randomly flipping ground truth when producing anchors during training is set to 0.05. We compare two type of region embedding in Sec.4.4, and adopt the Conv-based one in other experiments. Adam [14] optimization method is used for training our model. Learning rate is set to 1e-3. We do not use a validation set, and train our model until its training loss converges. The training process takes almost 16 hours and converges after around 300k iterations with mini-batch of size 1.

When comparing performance with other methods, the number of iterations T in the iterative testing scheme (Alg. 2) is set to 1. We discuss the effect of larger T values in Sec.4.4. When testing, the proposed method runs at about 15 fps with 256 256 resolution on our computer with a 3.6GHz CPU and a GTX 1080 GPU. We release our code for future comparisons²³.

4.4. Ablation studies

Quantitative comparison between the two types of region embedding is shown in Table 1. From this comparison

¹<https://github.com/pytorch>

²<http://ice.dlut.edu.cn/lu/>

³<https://github.com/zengxianyu/lps>

Baseline	UCF		RFCN		ELD	
Methods	F_β	MAE	F_β	MAE	F_β	MAE
BS	0.8394	0.0776	0.8337	0.1069	0.8098	0.0789
FC	0.8902	0.0575	0.8941	0.0628	0.8820	0.0592
Conv	0.8805	0.0560	0.8885	0.0570	0.8689	0.0577

Table 1. Comparison in terms of F-measure (the larger the better) and MAE (the smaller the better) between two types of region embedding evaluated on ECSSD dataset. The best and the second best methods are in red and green respectively. BS: baseline; FC: baseline promoted by the proposed method with FC-based region embedding; Conv: baseline promoted by the proposed method with Conv-based region embedding.

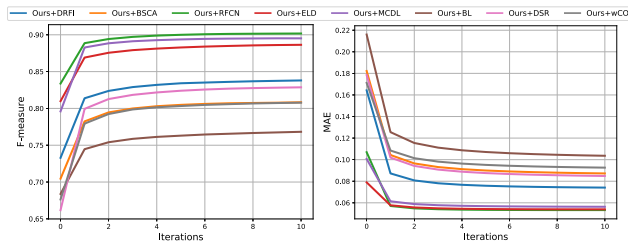


Figure 5. Quantitative effect evaluated on ECSSD dataset in terms of F-measure and MAE of the proposed iterative testing scheme. Different lines represents the effect of applying the proposed method on different algorithms.

we can see that the performance of FC-based and Conv-based region embedding is comparable. The FC-based region embedding yields relatively larger F-measure, while Conv-based region embedding is more superior in terms of MAE.

We show the effect of the proposed iterative approximation scheme in Figure 5. As shown in Figure 5, the first iteration improve the F-measure and decrease MAE most significantly. The improvement slows down with iterations, and saturates gradually.

4.5. Performance

We choose 13 state-of-the-art methods as baselines, including 8 deep learning based methods (Amulet [34], SRM [29], UCF [35], DHS [20], NLDF [21], ELD [15], RFCN [28], DSS [11]), and 5 conventional contenders (BSCA [23], DRFI [13], wCO [36], DSR [17], BL [26]). We apply our method to promote the performance of each baseline method, by using its predicted saliency maps to generate initial anchors in Eqn.3. Figure 6 shows the PR curves of the baseline methods and the one promoted by our method. Table 2 shows the F-measure and MAE scores of 8 deep learning based methods and the corresponding promoted results. The quantified improvements in F-measure and MAE of applying our method to conventional methods are shown in Table 3. As shown in Figure 6, Table 2, and Table 3, our method drastically promotes all the baseline methods.

Based on our results, we make several fundamental ob-

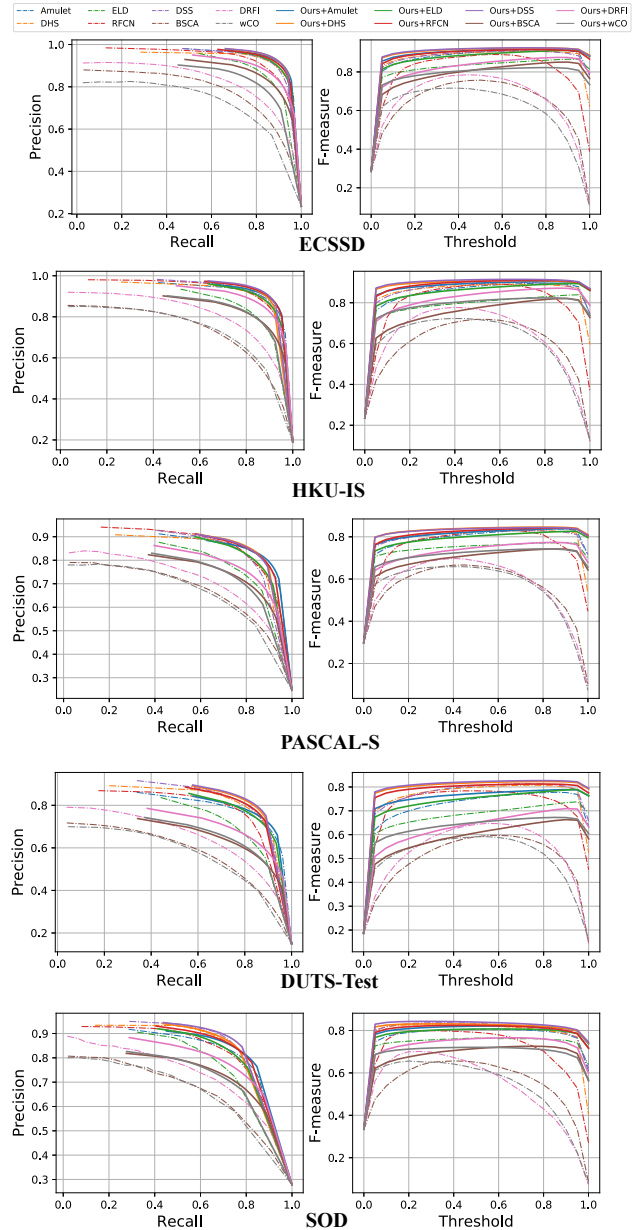


Figure 6. PR curves and F-measure curves of our method and the state-of-the-art methods.

servations:

1. Our proposed method decreases the MAE of SRM, the best-performing method to date, by 15.3% on HKU-IS dataset and 14.2% on ECSSD dataset.
2. Although our method is based on deep learning, it also performs well when applied to conventional methods. For instance, our method decreases the MAE of DRFI by around 50% on both ECSSD and HKU-IS datasets. Our method does not rely on any specific choice of the initial map, and generalizes well across different baseline methods.

		ECSSD		HKU-IS		PASCALS		DUTS-Test		SOD	
Methods		F_β	MAE	F_β	MAE	F_β	MAE	F_β	MAE	F_β	MAE
Amulet	BS	0.8691	0.0590	0.8388	0.0521	0.7677	0.0982	0.6755	0.0851	0.7546	0.1407
	Ours	0.8963	0.0509	0.8772	0.0446	0.7985	0.0920	0.7281	0.0828	0.7769	0.1336
SRM	BS	0.8921	0.0542	0.8739	0.0458	0.8007	0.0850	0.7570	0.0587	0.8004	0.1265
	Ours	0.9151	0.0465	0.9042	0.0388	0.8240	0.0810	0.8023	0.0558	0.8036	0.1170
UCF	BS	0.8394	0.0776	0.8076	0.0740	0.7056	0.1262	0.6288	0.1173	0.6989	0.1640
	Ours	0.8805	0.0560	0.8530	0.0546	0.7703	0.1044	0.6911	0.1051	0.7520	0.1470
DHS	BS	0.8716	0.0588	0.8550	0.0525	0.7787	0.0937	0.7242	0.0670	0.7736	0.1278
	Ours	0.9058	0.0482	0.8923	0.0421	0.8155	0.0859	0.7822	0.0610	0.7925	0.1216
NLDF	BS	0.8781	0.0626	0.8735	0.0477	0.7787	0.0990	0.7426	0.0650	0.7906	0.1242
	Ours	0.9046	0.0523	0.8986	0.0413	0.8121	0.0905	0.7867	0.0612	0.8058	0.1206
ELD	BS	0.8098	0.0789	0.7694	0.0736	0.7179	0.1227	0.6277	0.0923	0.7115	0.1545
	Ours	0.8689	0.0577	0.8443	0.0511	0.7694	0.1022	0.7043	0.0805	0.7606	0.1384
RFCN	BS	0.8337	0.1069	0.8349	0.0889	0.7511	0.1323	0.7135	0.0901	0.7425	0.1696
	Ours	0.8885	0.0570	0.8831	0.0437	0.7968	0.0946	0.7688	0.0666	0.7856	0.1323
DSS	BS	0.8728	0.0617	0.8557	0.0501	0.7733	0.1031	0.7202	0.0648	0.7867	0.1262
	Ours	0.9075	0.0492	0.8995	0.0394	0.8117	0.0906	0.7867	0.0588	0.8061	0.1187

Table 2. Comparison in terms of F-measure (the larger the better) and MAE (the smaller the better) score of our method against other deep learning based methods. The best and the second best methods are in red and green respectively. BS: the baseline; Ours: the promoted result of applying our method on the baseline.

		ECSSD		HKU-IS		PASCALS		DUTS-Test		SOD	
Methods		F_β	MAE	F_β	MAE	F_β	MAE	F_β	MAE	F_β	MAE
BSCA	BS	0.7046	0.1821	0.6544	0.1747	0.6005	0.2228	0.4995	0.1961	0.5835	0.2516
	Ours	0.7823	0.1043	0.7386	0.1075	0.6690	0.1654	0.5533	0.1711	0.6634	0.2001
DRFI	BS	0.7329	0.1642	0.7218	0.1444	0.6181	0.2065	0.5406	0.1746	0.6343	0.2240
	Ours	0.8136	0.0872	0.8061	0.0722	0.6943	0.1443	0.5895	0.1457	0.7069	0.1686
wCO	BS	0.6763	0.1711	0.6769	0.1423	0.5998	0.2018	0.5058	0.1531	0.5987	0.2293
	Ours	0.7792	0.1084	0.7765	0.0883	0.6844	0.1551	0.5932	0.1365	0.6732	0.1878
DSR	BS	0.6617	0.1783	0.6773	0.1421	0.5574	0.2148	0.5182	0.1454	0.5962	0.2344
	Ours	0.7993	0.1018	0.7992	0.0798	0.6806	0.1570	0.6353	0.1201	0.6916	0.1834
BL	BS	0.6838	0.2159	0.6597	0.2070	0.5742	0.2487	0.4896	0.2379	0.5797	0.2669
	Ours	0.7445	0.1255	0.7066	0.1255	0.6397	0.1788	0.5074	0.2007	0.6354	0.2053

Table 3. Comparison in terms of F-measure (the larger the better) and MAE (the smaller the better) score of our method against the conventional methods. The best and the second best methods are in red and green respectively. BS: the baseline; Ours: the promoted result of applying our method on the baseline.

- Notice that the results shown here are obtained by iterating Alg. 2 only once for fast testing speed. As shown in Sec.4.4, better results can be achieved through iterating Alg. 2 more times.

Figure 7 shows a visual comparison of saliency maps produced by some state-of-the-art methods and the promoted ones by our method. It can be seen that the saliency maps produced by our methods highlight salient regions that are missed by the baselines. Further, our method can suppress the background regions that are wrongly labeled as salient by the baseline methods.

5. Conclusion

In this paper, we propose a novel learning method to promote existing salient object detection methods. Extensive experiments on five benchmark datasets show that our method can significantly improve accuracy of existing methods and compares favorably against state-of-the-arts.

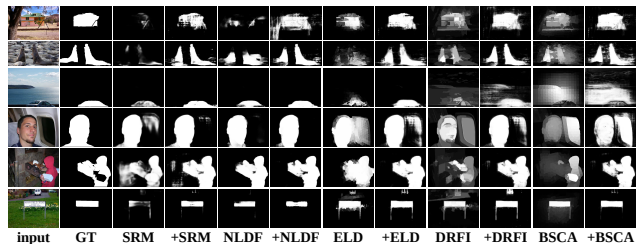


Figure 7. Visual comparison of the algorithms promoted by our method against the baseline algorithms. Input: input images; GT: ground truth maps; A plus sign denotes the algorithm promoted by our method.

Acknowledgment

This work was supported by the Natural Science Foundation of China under Grant 61725202, 61472060 and 61371157. In addition, the authors would like to thank Li-jun Wang, Hongshuang Zhang and Yunhua Zhang for their help.

References

- [1] R. Achanta, S. Hemami, F. Estrada, and S. Susstrunk. Frequency-tuned salient region detection. In *Computer vision and pattern recognition, 2009. cvpr 2009. IEEE conference on*, pages 1597–1604. IEEE, 2009.
- [2] R. Achanta and S. Süssstrunk. Saliency detection for content-aware image resizing. In *Image Processing (ICIP), 2009 16th IEEE International Conference on*, pages 1005–1008. IEEE, 2009.
- [3] A. Borji. Boosting bottom-up and top-down visual features for saliency estimation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 438–445. IEEE, 2012.
- [4] A. Borji, M.-M. Cheng, H. Jiang, and J. Li. Salient object detection: A benchmark. *IEEE Transactions on Image Processing*, 24(12):5706–5722, 2015.
- [5] A. Borji and L. Itti. State-of-the-art in visual attention modeling. *IEEE transactions on pattern analysis and machine intelligence*, 35(1):185–207, 2013.
- [6] A. Borji, D. N. Sihite, and L. Itti. Quantitative analysis of human-model agreement in visual saliency modeling: A comparative study. *IEEE Transactions on Image Processing*, 22(1):55–69, 2013.
- [7] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. Imagenet: A large-scale hierarchical image database. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 248–255. IEEE, 2009.
- [8] M. Everingham, L. Van Gool, C. K. Williams, J. Winn, and A. Zisserman. The pascal visual object classes (voc) challenge. *International journal of computer vision*, 88(2):303–338, 2010.
- [9] D. Gao, S. Han, and N. Vasconcelos. Discriminant saliency, the detection of suspicious coincidences, and applications to visual recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 31(6):989–1005, 2009.
- [10] Y. Gao, M. Wang, Z.-J. Zha, J. Shen, X. Li, and X. Wu. Visual-textual joint relevance learning for tag-based social image search. *IEEE Transactions on Image Processing*, 22(1):363–376, 2013.
- [11] Q. Hou, M.-M. Cheng, X. Hu, A. Borji, Z. Tu, and P. Torr. Deeply supervised salient object detection with short connections. In *Computer Vision and Pattern Recognition (CVPR), 2017 IEEE Conference on*, pages 5300–5309. IEEE, 2017.
- [12] B. Jiang, L. Zhang, H. Lu, C. Yang, and M.-H. Yang. Saliency detection via absorbing markov chain. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1665–1672, 2013.
- [13] H. Jiang, J. Wang, Z. Yuan, Y. Wu, N. Zheng, and S. Li. Salient object detection: A discriminative regional feature integration approach. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2083–2090, 2013.
- [14] D. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.
- [15] G. Lee, Y.-W. Tai, and J. Kim. Deep saliency with encoded low level distance map and high level features. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 660–668, 2016.
- [16] G. Li and Y. Yu. Visual saliency based on multiscale deep features. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5455–5463, 2015.
- [17] X. Li, H. Lu, L. Zhang, X. Ruan, and M.-H. Yang. Saliency detection via dense and sparse reconstruction. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2976–2983, 2013.
- [18] X. Li, L. Zhao, L. Wei, M.-H. Yang, F. Wu, Y. Zhuang, H. Ling, and J. Wang. Deepsaliency: Multi-task deep neural network model for salient object detection. *IEEE Transactions on Image Processing*, 25(8):3919–3930, 2016.
- [19] Y. Li, X. Hou, C. Koch, J. M. Rehg, and A. L. Yuille. The secrets of salient object segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 280–287, 2014.
- [20] N. Liu and J. Han. Dhsnet: Deep hierarchical saliency network for salient object detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 678–686, 2016.
- [21] Z. Luo, A. Mishra, A. Achkar, J. Eichel, S. Li, and P.-M. Jodoin. Non-local deep features for salient object detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017.
- [22] F. Perazzi, P. Krähenbühl, Y. Pritch, and A. Hornung. Saliency filters: Contrast based filtering for salient region detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 733–740. IEEE, 2012.
- [23] Y. Qin, H. Lu, Y. Xu, and H. Wang. Saliency detection via cellular automata. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 110–119, 2015.
- [24] W. Shi, J. Caballero, F. Huszár, J. Totz, A. P. Aitken, R. Bishop, D. Rueckert, and Z. Wang. Real-time single image and video super-resolution using an efficient sub-pixel convolutional neural network. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1874–1883, 2016.
- [25] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.
- [26] N. Tong, H. Lu, X. Ruan, and M.-H. Yang. Salient object detection via bootstrap learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1884–1892, 2015.
- [27] L. Wang, H. Lu, Y. Wang, M. Feng, D. Wang, B. Yin, and X. Ruan. Learning to detect salient objects with image-level supervision. In *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*, pages 136–145, 2017.
- [28] L. Wang, L. Wang, H. Lu, P. Zhang, and X. Ruan. Saliency detection with recurrent fully convolutional networks. In *European Conference on Computer Vision*, pages 825–841. Springer, 2016.
- [29] T. Wang, A. Borji, L. Zhang, P. Zhang, and H. Lu. A stage-wise refinement model for detecting salient objects in im-

- ages. In *Proceedings of the IEEE International Conference on Computer Vision*, 2017.
- [30] J. Xiao, J. Hays, K. A. Ehinger, A. Oliva, and A. Torralba. Sun database: Large-scale scene recognition from abbey to zoo. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3485–3492. IEEE, 2010.
- [31] Q. Yan, L. Xu, J. Shi, and J. Jia. Hierarchical saliency detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1155–1162, 2013.
- [32] C. Yang, L. Zhang, H. Lu, X. Ruan, and M.-H. Yang. Saliency detection via graph-based manifold ranking. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3166–3173, 2013.
- [33] J. Yang and M.-H. Yang. Top-down visual saliency via joint crf and dictionary learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2296–2303. IEEE, 2012.
- [34] P. Zhang, D. Wang, H. Lu, H. Wang, and X. Ruan. Amulet: Aggregating multi-level convolutional features for salient object detection. In *Proceedings of the IEEE International Conference on Computer Vision*, 2017.
- [35] P. Zhang, D. Wang, H. Lu, H. Wang, and B. Yin. Learning uncertain convolutional features for accurate saliency detection. In *Proceedings of the IEEE International Conference on Computer Vision*, 2017.
- [36] W. Zhu, S. Liang, Y. Wei, and J. Sun. Saliency optimization from robust background detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2814–2821, 2014.
- [37] F. Zund, Y. Pritch, A. Sorkine-Hornung, S. Mangold, and T. Gross. Content-aware compression using saliency-driven image retargeting. In *Image Processing (ICIP), 2013 20th IEEE International Conference on*, pages 1845–1849. IEEE, 2013.