Pattern Recognition 106 (2020) 107404

Contents lists available at ScienceDirect

Pattern Recognition

journal homepage: www.elsevier.com/locate/patcog

U^2-Net: Going deeper with nested U-structure for salient object detection

Xuebin Qin[*], Zichen Zhang, Chenyang Huang, Masood Dehghan, Osmar R. Zaiane, Martin Jagersand

Department of Computing Science, University of Alberta, Edmonton, Alberta T6G 2E8, Canada

ARTICLE INFO

Article history:
Received 27 November 2019
Revised 6 April 2020
Accepted 28 April 2020
Available online 16 May 2020

Keywords:
Salient object detection
Convolutional neural network
Network architecture design
Nested U-structure
Multi-scale feature extraction

ABSTRACT

In this paper, we design a simple yet powerful deep network architecture, U^2-Net, for salient object detection (SOD). The architecture of our U^2-Net is a two-level nested U-structure. The design has the following advantages: (1) it is able to capture more contextual information from different scales thanks to the mixture of receptive fields of different sizes in our proposed ReSidual U-blocks (RSU), (2) it increases the depth of the whole architecture without significantly increasing the computational cost because of the pooling operations used in these RSU blocks. This architecture enables us to train a deep network from scratch without using backbones from image classification tasks. We instantiate two models of the proposed architecture, U^2-Net (176.3 MB, 30 FPS on GTX 1080Ti GPU) and U^2-Net† (4.7 MB, 40 FPS), to facilitate the usage in different environments. Both models achieve competitive performance on six SOD datasets. The code is available: https://github.com/NathanUA/U-2-Net.

1. Introduction

Salient Object Detection (SOD) aims at segmenting the most visually attractive objects in an image. It is widely used in many fields, such as visual tracking and image segmentation. Recently, with the development of deep convolutional neural networks (CNNs), especially the rise of Fully Convolutional Networks (FCN) [1] in image segmentation, the salient object detection has been improved significantly. It is natural to ask, what is still missing? Let's take a step back and look at the remaining challenges.

There is a common pattern in the design of most SOD networks [2–5], that is, they focus on making good use of deep features extracted by existing backbones, such as Alexnet [6], VGG [7], ResNet [8], ResNeXt [9], DenseNet [10], etc. However, these backbones are all originally designed for image classification. They extract features that are representative of semantic meaning rather than local details and global contrast information, which are essential to saliency detection. And they need to be pre-trained on ImageNet [11] data which is data-inefficient especially if the target data follows a different distribution than ImageNet.

This leads to our first question: can we design a new network for SOD, that allows training from scratch and achieves comparable or better performance than those based on existing pre-trained backbones?

There are a few more issues on the network architectures for SOD. First, they are often overly complicated. It is partially due to the additional feature aggregation modules that are added to the existing backbones to extract multi-level saliency features from these backbones. Secondly, the existing backbones usually achieve deeper architecture by sacrificing high resolution of feature maps. To run these deep models with affordable memory and computational cost, the feature maps are down scaled to lower resolution at early stages. For instance, at the early layers of both ResNet and DenseNet [10], a convolution with stride of two followed by a max-pooling with stride of two are utilized to reduce the size of the feature maps to one fourth of the input maps. However, high resolution also plays an important role in segmentation besides the deep architecture.

Hence, our follow-up question is: can we go deeper while maintaining high resolution feature maps, at a low memory and computation cost?

Our main contribution is a novel and simple network architecture, called **U^2-Net**, that addresses the two questions above. First, U^2-Net is a two-level nested U-structure that is designed for SOD without using any pre-trained backbones from image classification. It can be trained from scratch to achieve competitive performance.

[*] Corresponding author.
E-mail addresses: xuebin@ualberta.ca (X. Qin), zichen2@ualberta.ca (Z. Zhang), chuang8@ualberta.ca (C. Huang), masood1@ualberta.ca (M. Dehghan), zaiane@ualberta.ca (O.R. Zaiane), mj7@ualberta.ca (M. Jagersand).

https://doi.org/10.1016/j.patcog.2020.107404
0031-3203/© 2020 Elsevier Ltd. All rights reserved.

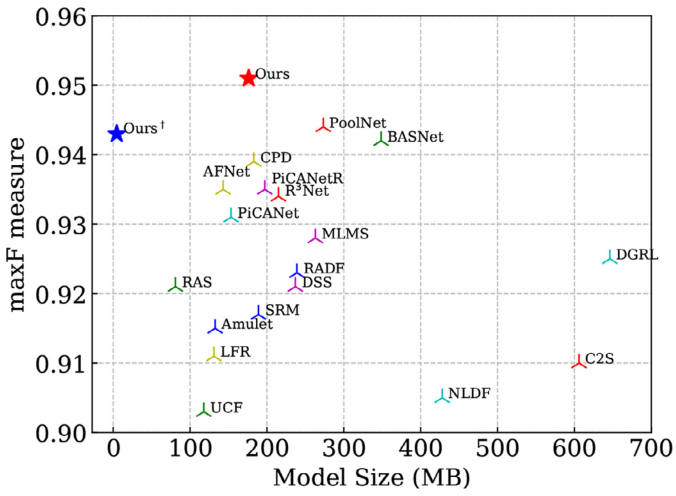

Fig. 1. Comparison of model size and performance of our U^2-Net with other state-of-the-art SOD models. The $maxF_\beta$ measure is computed on dataset ECSSD [12]. The red star denotes our U^2-Net (Ours) (176.3 MB) and the blue star denotes our small version U^2-Net† (Ours†) (4.7 MB). (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

Second, the novel architecture allows the network to go deeper, attain high resolution, without significantly increasing the memory and computation cost. This is achieved by a nested U-structure: on the bottom level, we design a novel ReSidual U-block (RSU), which is able to extract intra-stage multi-scale features without degrading the feature map resolution; on the top level, there is a U-Net like structure, in which each stage is filled by a RSU block. The two-level configuration results in a nested U-structure (see Fig. 5). Our U^2-Net (176.3 MB) achieves competitive performance against the state-of-the-art (SOTA) methods on six public datasets, and runs at real-time (30 FPS, with input size of 320 $\times$ 320 $\times$ 3) on a 1080Ti GPU. To facilitate the usage of our design in computation and memory constrained environments, we provide a small version of our U^2-Net, called **U^2-Net†** (4.7 MB). The U^2-Net† achieves competitive results against most of the SOTA models (see Fig. 1) at 40 FPS.

2. Related works

In recent years, many deep salient object detection networks [13,14] have been proposed. Compared with traditional methods based on hand-crafted features like foreground consistency [15], hyperspectral information [16], superpixels' similarity [17], histograms [18,19] and so on, deep salient object detection networks show more competitive performance.

2.1. Multi-level deep feature integration

Recent works [1,20] have shown that features from multiple deep layers are able to generate better results [21]. Then, many strategies and methods for integrating and aggregating multi-level deep features are developed for SOD. Li et al.(MDF) [2] propose to feed an image patch around a target pixel to a network and then obtain a feature vector for describing the saliency of this pixel. Zhang et al.(Amulet) [22] predict saliency maps by aggregating multi-level features into different resolutions. Zhang et al.(UCF) [23] propose to reduce the checkerboard artifacts of deconvolution operators by introducing a reformulated dropout and a hybrid upsampling module. Luo et al.[3] design a saliency detection network (NLDF+) with a 4 $\times$ 5 grid architecture, in which deeper features are progressively integrated with shallower features. Hou et al.(DSS+) [24] propose to integrate multi-level features by introducing short connections from deep layers to shallow layers. Chen et al.(RAS) [25] predict and refine saliency maps by iteratively using the side output saliency of a backbone network as the feature attention guidance. Zhang et al.(BMPM) [21] propose to integrate features from shallow and deep layers by a controlled bi-directional passing strategy. Deng et al.(R^{3}Net+) [5] alternately incorporate shallow and deep layers' features to refine the predicted saliency maps. Wu et al.(MLMS) [26] improve the saliency detection accuracy by developing a novel Mutual Learning Module for better leveraging the correlation of boundaries and regions. Wu e.g.[27] propose to use Cascaded Partial Decoder (CPD) framework for fast and accurate salient object detection. Deep methods in this category take advantage of the multi-level deep features extracted by backbone networks and greatly raise the bar of salient object detection against traditional methods.

2.2. Multi-scale feature extraction

As mentioned earlier, saliency detection requires both local and global information. A 3 $\times$ 3 filter is good for extracting local features at each layer. However, it is difficult to extract global information by simply enlarging the filter size because it will increase the number of parameters and computation costs dramatically. Many works pay more attention to extracting global context. Wang et al.(SRM) [28] adapt the pyramid pooling module [29] to capture global context and propose a multi-stage refinement mechanism for saliency maps refinement. Zhang et al.(PAGRN) [30] develop a spatial and a channel-wise attention module to obtain the global information of each layer and propose a progressive attention guidance mechanism to refine the saliency maps. Wang et al.(DGRL) [4] develop an inception-like contextual weighting module to localize salient objects globally and then use a boundary refinement module to refine the saliency map locally. Liu et al.(PiCANet) [31] recurrently capture the local and global pixel-wise contextual attention and predict the saliency map by incorporating it with a U-Net architecture. Zhang et al.(CapSal) [32] design a local and global perception module to extract both local and global information from features extracted by backbone network. Zeng et al.(MSWS) [33] design an attention module to predict the spatial distribution of foreground objects over image regions meanwhile aggregate their features. Feng et al.(AFNet) [34] develop a global perception module and attentive feedback modules to better explore the structure of salient objects. Qin et al.(BASNet) [14] propose a predict-refine model by stacking two differently configured U-Nets sequentially and a Hybrid loss for boundary-aware salient object detection. Liu et al.(PoolNet) [13] develop encoder-decoder architecture for salient object detection by introducing a global guidance module for extraction of global localization features and a multi-scale feature aggregation module adapted from pyramid pooling module for fusing global and fine-level features. In these methods, many inspiring modules are proposed to extract multi-scale features from multi-level deep features extracted from existing backbones. Diversified receptive fields and richer multi-scale contextual features introduced by these novel modules significantly improve the performance of salient object detection models.

In summary, multi-level deep feature integration methods mainly focus on developing better multi-level feature aggregation strategies. On the other hand, methods in the category of multi-scale feature extraction target at designing new modules for extracting both local and global information from features obtained by backbone networks. As we can see, almost all of the aforementioned methods try to make better use of feature maps generated by the existing image classification backbones. Instead of developing and adding more complicated modules and strategies to use

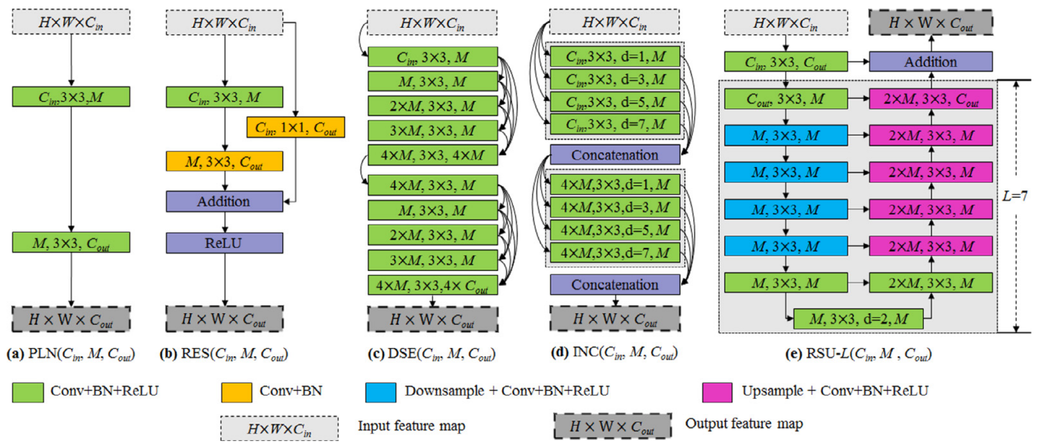

Fig. 2. Illustration of existing convolution blocks and our proposed residual U-block RSU: (a) Plain convolution block PLN, (b) Residual-like block RES, (c) Inception-like block INC, (d) Dense-like block DSE and (e) Our residual U-block RSU.

these backbones' features, we propose a novel and simple architecture, which directly extracts multi-scale features stage by stage, for salient object detection.

3. Proposed method

First, we introduce the design of our proposed residual U-block and then describe the details of the nested U-architecture built with this block. The network supervision strategy and the training loss are described at the end of this section.

3.1. Residual U-blocks

Both local and global contextual information are very important for salient object detection and other segmentation tasks. In modern CNN designs, such as VGG, ResNet, DenseNet and so on, small convolutional filters with size of 1×1 or 3×3 are the most frequently used components for feature extraction. They are in favor since they require less storage space and are computationally efficient. Fig. 2(a)–(c) illustrates typical existing convolution blocks with small receptive fields. The output feature maps of shallow layers only contain local features because the receptive field of 1×1 or 3×3 filters are too small to capture global information. To achieve more global information at high resolution feature maps from shallow layers, the most direct idea is to enlarge the receptive field. Fig. 2(d) shows an inception like block [21], which tries to extract both local and non-local features by enlarging the receptive fields using dilated convolutions. However, conducting multiple dilated convolutions on the input feature map (especially in the early stage) with original resolution requires too much computation and memory resources. To decrease the computation costs, PoolNet [13] adapt the parallel configuration from pyramid pooling modules (PPM) [29], which uses small kernel filters on the downsampled feature maps other than the dilated convolutions on the original size feature maps. But fusion of different scale features by direct upsampling and concatenation (or addition) may lead to degradation of high resolution features.

Inspired by U-Net [35], we propose a novel **Re**S**idual** **U**-block, **RSU**, to capture intra-stage multi-scale features. The structure of RSU-$L(C_{in}, M, C_{out})$ is shown in Fig. 2(e), where L is the number of layers in the encoder, C_{in}, C_{out} denote input and output channels, and M denotes the number of channels in the internal layers of RSU. Hence, our RSU mainly consists of three components:

(i) an input convolution layer, which transforms the input feature map $\mathbf{x}$ $(H \times W \times C_{in})$ to an intermediate map $\mathcal{F}_1(\mathbf{x})$ with channel of C_{out}. This is a plain convolutional layer for local feature extraction.

(ii) a U-Net like symmetric encoder-decoder structure with height of L which takes the intermediate feature map $\mathcal{F}_1(\mathbf{x})$ as input and learns to extract and encode the multi-scale contextual information $\mathcal{U}(\mathcal{F}_1(\mathbf{x}))$. $\mathcal{U}$ represents the U-Net like structure as shown in Fig. 2(e). Larger L leads to deeper residual U-block (RSU), more pooling operations, larger range of receptive fields and richer local and global features. Configuring this parameter enables extraction of multi-scale features from input feature maps with arbitrary spatial resolutions. The multi-scale features are extracted from gradually downsampled feature maps and encoded into high resolution feature maps by progressive upsampling, concatenation and convolution. This process mitigates the loss of fine details caused by direct upsampling with large scales.

(iii) a residual connection which fuses local features and the multi-scale features by the summation: $\mathcal{F}_1(\mathbf{x}) + \mathcal{U}(\mathcal{F}_1(\mathbf{x}))$.

To better illustrate the intuition behind our design, we compare our residual U-block (RSU) with the original residual block [8] in Fig. 3. The operation in the residual block can be summarized as $\mathcal{H}(\mathbf{x}) = \mathcal{F}_2(\mathcal{F}_1(\mathbf{x})) + \mathbf{x}$, where $\mathcal{H}(x)$ denotes the desired mapping of the input features $\mathbf{x}$; $\mathcal{F}_2, \mathcal{F}_1$ stand for the weight layers, which are convolution operations in this setting. The main design difference between RSU and residual block is that RSU replaces the plain, single-stream convolution with a U-Net like structure, and replace the original feature with the local feature transformed by a weight layer: $\mathcal{H}_{RSU}(\mathbf{x}) = \mathcal{U}(\mathcal{F}_1(\mathbf{x})) + \mathcal{F}_1(\mathbf{x})$, where $\mathcal{U}$ represents the multi-layer U-structure illustrated in Fig. 2(e). This design change empowers the network to extract features from multiple scales directly from each residual block. More notably, the computation overhead due to the U-structure is small, since most operations are applied on the downsampled feature maps. This is illustrated in Fig. 4, where we show the computation cost comparison between RSU and other feature extraction modules in Fig. 2(a)–(d). The FLOPs of dense block (DSE), inception block (INC) and RSU all grow quadratically with the number of internal channel M. But RSU has a much smaller coefficient on the quadratic term, leading to an improved efficiency. Its computational overhead compared with plain convolution (PLN) and residual block (RES) blocks, which are both linear w.r.t. M, is not significant.

3.2. Architecture of U^2-Net

Stacking multiple U-Net-like structures for different tasks has been explored for a while. These methods usually stack U-Net-like

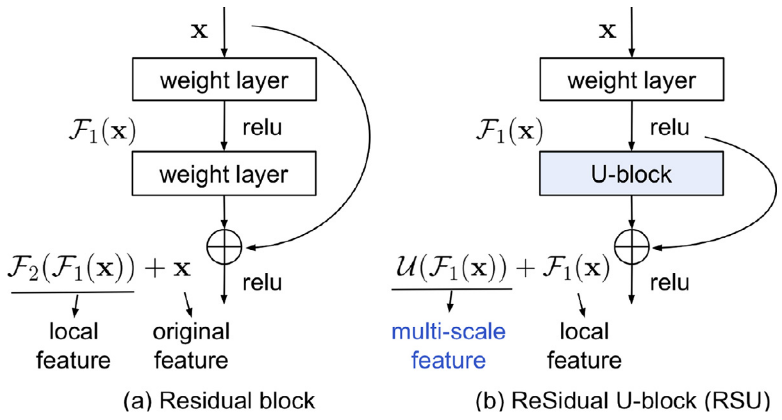
x
weight layer
$\mathcal{F}_1(\mathbf{x})$
relu
weight layer
$\mathcal{F}_2(\mathcal{F}_1(\mathbf{x})) + \mathbf{x}$
relu
local feature
original feature
(a) Residual block
x
weight layer
$\mathcal{F}_1(\mathbf{x})$
relu
U-block
$\mathcal{U}(\mathcal{F}_1(\mathbf{x})) + \mathcal{F}_1(\mathbf{x})$
relu
multi-scale feature
local feature
(b) ReSidual U-block (RSU)

Fig. 3. Comparison of the residual block and our RSU.

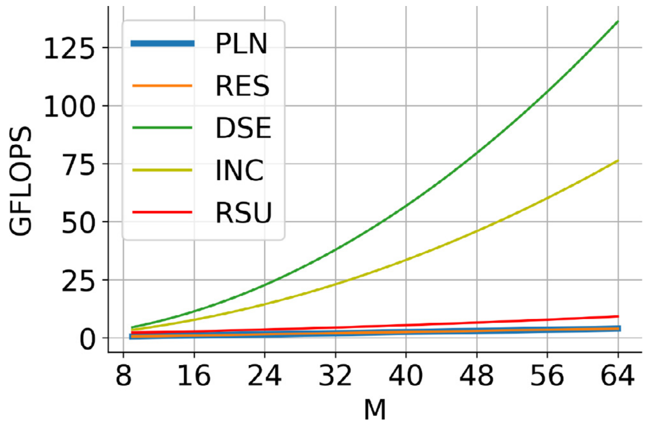

Fig. 4. Computation costs (GFLOPS Giga Floating Point Operations) of different blocks shown in Fig. 2: the computation costs are calculated based on transferring an input feature map with dimension 320 × 320 × 3 to a 320 × 320 × 64 output feature map. "PLN", "RES", "DSE", "INC" and "RSU" denote plain convolution block, residual block, dense block, inception block and our residual U-block respectively.

structures sequentially to build cascaded models and can be summarized as "(**U** × *n*-**Net**)", where n is the number of repeated U-Net modules. The issue is that the computation and the memory costs get magnified by n.

In this paper, we propose a different formulation, **U**^{n}**-Net**, of stacking U-structure for salient object detection. Our exponential notation refers to nested U-structure rather than cascaded stacking. Theoretically, the exponent n can be set as an arbitrary positive integer to achieve single-level or multi-level nested U-structure. But architectures with too many nested levels will be too complicated to be implemented and employed in real applications.

Here, we set n as 2 to build our U^2-Net. Our U^2-Net is a two-level nested U-structure shown in Fig. 5. Its top level is a big U-structure consists of 11 stages (cubes in Fig. 5). Each stage is filled by a well configured residual U-block (RSU) (bottom level U-structure). Hence, the nested U-structure enables the extraction of intra-stage multi-scale features and aggregation of inter-stage multi-level features more efficiently.

As illustrated in Fig. 5, the U^2-Net mainly consists of three parts: (1) a six stages encoder, (2) a five stages decoder and (3) a saliency map fusion module attached with the decoder stages and the last encoder stage:

(i) In encoder stages **En_1, En_2, En_3** and **En_4**, we use residual U-blocks RSU-7, RSU-6, RSU-5 and RSU-4, respectively. As mentioned before, "7", "6", "5" and "4" denote the heights (L) of RSU blocks. The L is usually configured according to the spatial resolution of the input feature maps. For feature maps with large height and width, we use greater L to capture more large scale information. The resolution of feature maps in **En_5** and **En_6** are relatively low, further down-sampling of these feature maps leads to loss of useful context. Hence, in both **En_5** and **En_6** stages, RSU-4F are used, where "F" means that the RSU is a dilated version, in which we replace the pooling and upsampling operations with dilated convolutions (see Fig. 5). That means all of intermediate feature maps of RSU-4F have the same resolution with its input feature maps.

(ii) The decoder stages have similar structures to their symmetrical encoder stages with respect to **En_6**. In **De_5**, we also use the dilated version residual U-block RSU-4F which is similar to that used in the encoder stages **En_5** and **En_6**. Each decoder stage takes the concatenation of the upsampled feature maps from its previous stage and those from its symmetrical encoder stage as the input, see Fig. 5.

(iii) The last part is the saliency map fusion module which is used to generate saliency probability maps. Similar to HED [20], our U^2-Net first generates six side output saliency probability maps $\mathcal{S}_{side}^{(6)}$, $\mathcal{S}_{side}^{(5)}$, $\mathcal{S}_{side}^{(4)}$, $\mathcal{S}_{side}^{(3)}$, $\mathcal{S}_{side}^{(2)}$, $\mathcal{S}_{side}^{(1)}$ from stages **En_6, De_5, De_4, De_3, De_2** and **De_1** by a 3 × 3 convolution layer and a sigmoid function. Then, it upsamples these saliency maps to the input image size and fuses them with a concatenation operation followed by a 1 × 1 convolution layer and a sigmoid function to generate the final saliency probability map $\mathcal{S}_{fuse}$ (see bottom right of Fig. 5).

In summary, the design of our U^2-Net allows having deep architecture with rich multi-scale features and relatively low computation and memory costs. In addition, since our U^2-Net architecture is only built upon our RSU blocks without using any pre-trained backbones adapted from image classification, it is flexible and easy to be adapted to different working environments with insignificant performance loss. In this paper, we provide two instances of our U^2-Net by using different configurations of filter numbers: a normal version **U^2-Net** (176.3 MB) and a relatively smaller version **U^2-Net†** (4.7 MB). Detailed configurations are presented in the last two rows of Table 1.

3.3. Supervision

In the training process, we use deep supervision similar to HED [20]. Its effectiveness has been proven in HED and DSS. Our training loss is defined as:

$$\mathcal{L} = \sum_{m=1}^{M} w_{side}^{(m)} \ell_{side}^{(m)} + w_{fuse} \ell_{fuse} \quad (1)$$

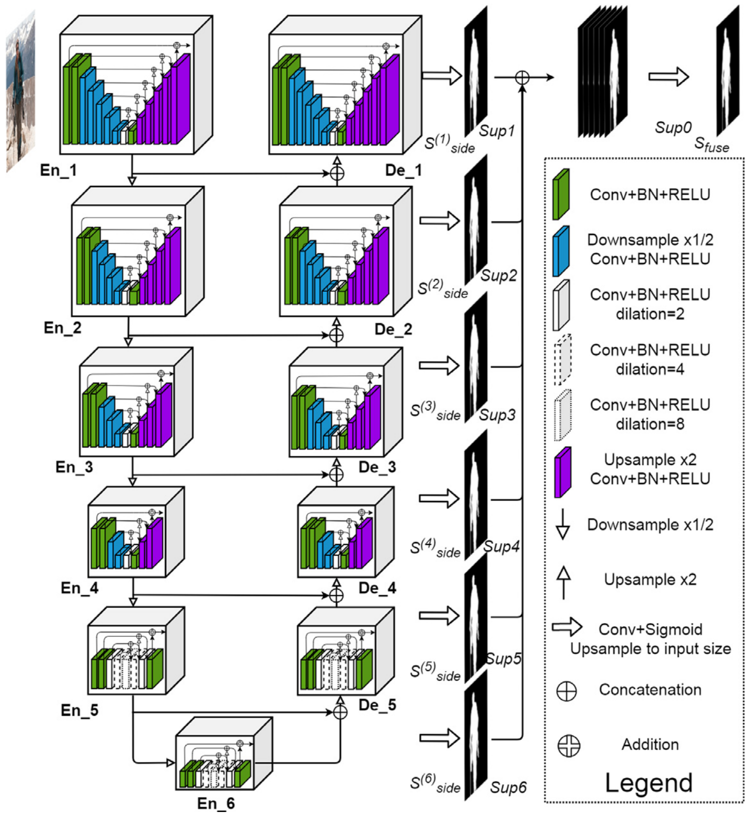

Fig. 5. Illustration of our proposed U^2-Net architecture. The main architecture is a U-Net like Encoder-Decoder, where each stage consists of our newly proposed residual U-block (RSU). For example, **En_1** is based on our RSU block shown in Fig. 2(e). Detailed configuration of RSU block of each stage is given in the last two rows of Table 1.

Table 1
Detailed configurations of different architectures used in ablation study. "PLN", "RES", "DSE", "INC", "PPM" and "RSU" denote plain convolution block, residual block, dense block, inception block, Pyramid Pooling Module and our residual U-block respectively. "NIV U^2-Net" denotes U-Net with its each stage replaced by a naive U-Net block. "I", "M" and "O" indicate the number of input channels (C_{in}), middle channels and output channels (C_{out}) of each block. "En_*i*" and "De_*j*" denote the encoder and decoder stages respectively. The number "*L*" in "NIV-*L*" and "RSU-*L*" denotes the height of the naive U-block and our residual U-block.

Architecture with different blocks	Stages										
	En_1	En_2	En_3	En_4	En_5	En_6	De_5	De_4	De_3	De_2	De_1
PLN U-Net	I:3	I:64	I:128	I:256	I:512	I:512	I:1024	I:1024	I:512	I:256	I:128
	M:64	M:128	M:256	M:512	M:512	M:512	M:512	M:256	M:128	M:64	M:64
	O:64	O:128	O:256	O:512	O:512	O:512	O:512	O:256	O:128	O:64	O:64
RES U-Net	I:3	I:64	I:128	I:256	I:512	I:512	I:1024	I:1024	I:512	I:256	I:128
	M:64	M:128	M:256	M:512	M:512	M:512	M:512	M:256	M:128	M:64	M:64
	O:64	O:128	O:256	O:512	O:512	O:512	O:512	O:256	O:128	O:64	O:64
DSE U-Net	I:3	I:64	I:128	I:256	I:512	I:512	I:1024	I:1024	I:512	I:256	I:128
	M:32	M:32	M:64	M:128	M:128	M:128	M:128	M:64	M:32	M:16	M:16
	O:64	O:128	O:256	O:512	O:512	O:512	O:512	O:256	O:128	O:64	O:64
INC U-Net	I:3	I:64	I:128	I:256	I:512	I:512	I:1024	I:1024	I:512	I:256	I:128
	M:32	M:32	M:64	M:128	M:128	M:128	M:128	M:64	M:32	M:16	M:16
	O:64	O:128	O:256	O:512	O:512	O:512	O:512	O:256	O:128	O:64	O:64
PPM U-Net	I:3	I:64	I:128	I:256	I:512	I:512	I:1024	I:1024	I:512	I:256	I:128
	M:32	M:32	M:64	M:128	M:128	M:128	M:128	M:64	M:32	M:16	M:16
	O:64	O:128	O:256	O:512	O:512	O:512	O:512	O:256	O:128	O:64	O:64
NIV U^2-Net	NIV-7	NIV-6	NIV-5	NIV-4	NIV-4F	NIV-4F	NIV-4F	NIV-4	NIV-5	NIV-6	NIV-7
	I:3	I:64	I:128	I:256	I:512	I:512	I:1024	I:1024	I:512	I:256	I:128
	M:32	M:32	M:64	M:128	M:256	M:256	M:256	M:128	M:64	M:32	M:16
	O:64	O:128	O:256	O:512	O:512	O:512	O:512	O:256	O:128	O:64	O:64
U^2-Net (**Ours**)	RSU-7	RSU-6	RSU-5	RSU-4	RSU-4F	RSU-4F	RSU-4F	RSU-4	RSU-5	RSU-6	RSU-7
	I:3	I:64	I:128	I:256	I:512	I:512	I:1024	I:1024	I:512	I:256	I:128
	M:32	M:32	M:64	M:128	M:256	M:256	M:256	M:128	M:64	M:32	M:16
	O:64	O:128	O:256	O:512	O:512	O:512	O:512	O:256	O:128	O:64	O:64
U^2-Net† (**Ours†**)	RSU-7	RSU-6	RSU-5	RSU-4	RSU-4F	RSU-4F	RSU-4F	RSU-4	RSU-5	RSU-6	RSU-7
	I:3	I:64	I:64	I:64	I:64	I:64	I:128	I:128	I:128	I:128	I:128
	M:16	M:16	M:16	M:16	M:16	M:16	M:16	M:16	M:16	M:16	M:16
	O:64	O:64	O:64	O:64	O:64	O:64	O:64	O:64	O:64	O:64	O:64

where $\ell_{side}^{(m)}$ ($M = 6$, as the Sup1, Sup2, ..., Sup6 in Fig. 5) is the loss of the side output saliency map $S_{side}^{(m)}$ and ℓ_{fuse} (Sup0 in Fig. 5) is the loss of the final fusion output saliency map S_{fuse}. $w_{side}^{(m)}$ and w_{fuse} are the weights of each loss term. For each term ℓ, we use the standard binary cross-entropy to calculate the loss:

$$\ell = -\sum_{(r,c)}^{(H,W)}[P_{G(r,c)}logP_{S(r,c)} + (1 - P_{G(r,c)})log(1 - P_{S(r,c)})] \qquad (2)$$

where (r, c) is the pixel coordinates and (H, W) is image size: height and width. $P_{G(r,c)}$ and $P_{S(r,c)}$ denote the pixel values of the ground truth and the predicted saliency probability map, respectively. The training process tries to minimize the overall loss $\mathcal{L}$ of Eq. (1). In the testing process, we choose the fusion output ℓ_{fuse} as our final saliency map.

4. Experimental results

4.1. Datasets

4.1.1. Training dataset

We train our network on **DUTS-TR**, which is a part of DUTS dataset [36]. **DUTS-TR** contains 10,553 images in total. Currently, it is the largest and most frequently used training dataset for salient object detection. We augment this dataset by horizontal flipping to obtain 21,106 training images offline.

4.1.2. Evaluation datasets

Six frequently used benchmark datasets are used to evaluate our method including: DUT-OMRON [37], DUTS-TE [36], HKU-IS [2], ECSSD [12], PASCAL-S [38], SOD [39]. **DUT-OMRON** includes 5168 images, most of which contain one or two structurally complex foreground objects. DUTS dataset consists of two parts: DUTS-TR and DUTS-TE. As mentioned above we use DUTS-TR for training. Hence, **DUTS-TE**, which contains 5019 images, is selected as one of our evaluation dataset. **HKU-IS** contains 4447 images with multiple foreground objects. **ECSSD** contains 1000 structurally complex images and many of them contain large foreground objects. **PASCAL-S** contains 850 images with complex foreground objects and cluttered background. **SOD** only contains 300 images. But it is very challenging. Because it was originally designed for image segmentation and many images are low contrast or contain complex foreground objects overlapping with the image boundary.

4.2. Evaluation metrics

The outputs of the deep salient object methods are usually probability maps that have the same spatial resolution with the input images. Each pixel of the predicted saliency maps has a value within the range of 0 and 1 (or [0, 255]). The ground truth are usually binary masks, in which each pixel is either 0 or 1 (or 0 and 255) where 0 indicates the background pixels and 1 indicates the foreground salient object pixels.

To comprehensively evaluate the quality of those probability maps against the ground truth, six measures including (1) Precision-Recall (PR) curves, (2) maximal F-measure (maxF_β) [40], (3) Mean Absolute Error (MAE) [13,14,31], (4) weighted F-measure (F_β^w) [41], (5) structure measure (S_m) [42] and (6) relaxed F-measure of boundary (relaxF_β^b) [14] are used:

(1) PR curve is plotted based on a set of precision-recall pairs. Given a predicted saliency probability map, its precision and recall scores are computed by comparing its thresholded binary mask against the ground truth mask. The precision and recall of a dataset are computed by averaging the precision and recall scores of those saliency maps. By varying the thresholds from 0 to 1, we can obtain a set of average precision-recall pairs of the dataset.

(2) F-measure F_β is used to comprehensively evaluate both precision and recall as:

$$F_\beta = \frac{(1+\beta^2)\times Precision \times Recall}{\beta^2 \times Precision + Recall}. \qquad (3)$$

We set the β^2 to 0.3 and report the maximum F_β (maxF_β) for each dataset similar to previous works [21,31,40].

(3) ***MAE*** is the Mean Absolute Error which denotes the average per-pixel difference between a predicted saliency map and its ground truth mask. It is defined as:

$$MAE = \frac{1}{H\times W}\sum_{r=1}^{H}\sum_{c=1}^{W}|P(r,c) - G(r,c)| \qquad (4)$$

where P and G are the probability map of the salient object detection and the corresponding ground truth respectively, (H, W) and (r, c) are the (height, width) and the pixel coordinates.

(4) weighted F-measure (F_β^w) is utilized as a complementary measure to maxF_β for overcoming the possible unfair comparison caused by "interpolation flaw, dependency flaw and equal-importance flaw" [31]. It is defined as:

$$F_\beta^w = (1+\beta^2)\frac{Precision^w \cdot Recall^w}{\beta^2 \cdot Precision^w + Recall^w}. \qquad (5)$$

(5) S-measure (S_m) is used to evaluate the structure similarity of the predicted non-binary saliency map and the ground truth. The S-measure is defined as the weighted sum of region-aware S_r and object-aware S_o structural similarity:

$$S = (1-\alpha)S_r + \alpha S_o. \qquad (6)$$

where α is usually set to 0.5.

(6) relax boundary F-measure relaxF_β^b is utilized to quantitatively evaluate boundaries' quality of the predicted saliency maps [14]. Given a saliency probability map $P \in [0, 1]$, its binary mask P_{bw} is obtained by a simple thresholding operation (threshold is set to 0.5). Then, the $XOR(P_{bw}, P_{erd})$ operation is conducted to obtain its one pixel wide boundary, where P_{erd} denotes the eroded binary mask of P_{bw}. The boundaries of ground truth mask are obtained in the same way. The computation of relaxed boundary F-measure relaxF_β^b is similar to Eq. (3). The difference is that relax$Precision^b$ and relax$Recall^b$ other than $Precision$ and $Recall$ are used in Eq. (3). The definition of relaxed boundary precision (relax$Precision^b$) is the fraction of predicted boundary pixels within a range of ρ pixels from ground truth boundary pixels. The relaxed boundary recall (relax$Recall^b$) is defined as the fraction of ground truth boundary pixels that are within ρ pixels of predicted boundary pixels. The slack parameter ρ is set to 3 as in the previous work [14]. Given a dataset, its average relaxF_β^b of all predicted saliency maps is reported in this paper.

4.3. Implementation details

In the training process, each image is first resized to 320 × 320 and randomly flipped vertically and cropped to 288 × 288. We are not using any existing backbones in our network. Hence, we train our network from scratch and all of our convolutional layers are initialized by Xavier. The loss weights $w_{side}^{(m)}$ and w_{fuse} are all set to 1. Adam optimizer is used to train our network and its hyper parameters are set to default (initial learning rate lr = 1e−3, betas = (0.9, 0.999), eps = 1e−8, weight_decay = 0). We train the network until the loss converges without using validation set which follows the previous methods [13,21,31]. After 600k iterations (with a batch size of 12), the training loss converges and the whole training process takes about 120 hours. During testing, the input images ($H \times W$) are resized to 320 × 320

Table 2
Results of ablation study on different blocks, architectures and backbones. "PLN", "RES", "DSE", "INC", "PPM" and "RSU" denote plain convolution block, residual block, dense block, inception block, pyramid pooling module and our residual U-block respectively. "NIV U^2-Net" denotes U-Net with its each stage replaced by a naive U-Net block. The "Time (ms)" (ms: milliseconds) costs are computed by averaging the inference time costs of images in ECSSD dataset. Values with bold fonts indicate the best two performance.

Configuration	DUT-OMRON		ECSSD		Time (ms)
	maxF_β	*MAE*	maxF_β	*MAE*	
Baseline U-Net	0.725	0.082	0.896	0.066	**14**
PLN U-Net	0.782	0.062	0.928	0.043	**16**
RES U-Net	0.781	0.065	0.933	0.042	19
DSE U-Net	0.790	0.067	0.927	0.046	70
INC U-Net	0.777	0.069	0.921	0.047	57
PPM U-Net	0.792	0.062	0.928	0.049	105
Stacked HourglassNet [43]	0.756	0.073	0.905	0.059	103
CU-NET [44]	0.767	0.072	0.913	0.061	50
NIV U^2-Net	0.803	0.061	0.938	0.085	30
U^2-Net w/ VGG-16 backbone	0.808	0.063	0.942	**0.038**	23
U^2-Net w/ ResNet-50 backbone	0.813	**0.058**	0.937	0.041	41
(Ours) RSU U^2-Net	**0.823**	**0.054**	**0.951**	**0.033**	33
(Ours†) RSU U^2-Net†	**0.813**	0.060	**0.943**	0.041	25

and fed into the network to obtain the saliency maps. The predicted saliency maps with size of 320 × 320 are resized back to the original size of the input image ($H \times W$). Bilinear interpolation is used in both resizing processes. Our network is implemented based on Pytorch 0.4.0. Both training and testing are conducted on an eight-core, 16 threads PC with an AMD Ryzen 1800 × 3.5 GHz CPU (32 GB RAM) and a GTX 1080ti GPU (11 GB memory). We will release our code later.

4.4. Ablation study

To verify the effectiveness of our U^2-Net, ablation studies are conducted on the following three aspects: (i) basic blocks, (ii) architectures and (iii) backbones. All the ablation studies follow the same implementation setup.

4.4.1. Ablation on blocks

In the blocks ablation, the goal is to validate the effectiveness of our newly designed residual U-blocks (RSUs). Specifically, we fix the outside Encoder-Decoder architecture of our U^2-Net and replace its stages with other popular blocks including plain convolution blocks (PLN), residual-like blocks (RSE), dense-like blocks (DSE), inception-like blocks (INC) and pyramid pooling module (PPM) other than RSU block, as shown in Fig. 2(a)–(d). Detailed configurations can be found in Table 1.

Table 2 shows the quantitative results of the ablation study. As we can see, the performance of baseline U-Net is the worst, while PLN U-Net, RES U-Net, DES U-Net, INC U-Net and PPM U-Net achieve better performance than the baseline U-Net. Because they are either deeper or have the capability of extracting multi-scale features. However, their performance is still inferior to both our full size U^2-Net and small version U^2-Net†. Particularly, our full size U^2-Net improves the maxF_β about 3.3% and 1.8%, and decreases the *MAE* over 12.9% and 21.4% against the second best model (in the blocks ablation study) on DUT-OMRON and ECSSD datasets, respectively. Furthermore, our U^2-Net and U^2-Net† increase the maxF_β by 9.8% and 8.8% and decrease the *MAE* by 34.1% and 27.0%, which are significant improvements, on DUT-OMRON dataset against the baseline U-Net. On ECSSD dataset, although the maxF_β improvements (5.5%, 4.7%) of our U^2-Net and U^2-Net† against the baseline U-Net is slightly less significant than that on DUT-OMRON, the improvements of *MAE* are much greater (50.0%, 38.0%). Therefore, we believe that our newly designed residual U-block RSU is better then others in this salient object detection task. Besides, there is no significant time costs increasing of our residual U-block (RSU) based U^2-Net architectures.

4.4.2. Ablation on architectures

As we mentioned above, previous methods usually use cascaded ways to stack multiple similar structures for building more expressive models. One of the intuitions behind this idea is that multiple similar structures are able to refine the results gradually while reducing overfitting. Stacked HourglassNet [43] and CU-Net [44] are two representative models in this category. Therefore, we adapted the stacked HourglassNet and CU-Net to compare the performance between the cascaded architectures and our nested architectures. As shown in Table 2, both our full size U^2-Net and small size model U^2-Net† outperform these two cascaded models. It is worth noting the both stacked HourglassNet and CU-Net utilizes improved U-Net-like modules as their stacking sub-models. To further demonstrate the effectiveness of our nested architecture, we also illustrate the performance of an U^2-Net based on naive U-blocks (NIV) other than our newly proposed residual U-blocks. We can see that the NIV U^2-Net still achieves better performance than these two cascaded models. In addition, the nested architectures are faster than the cascaded ones. In summary, our nested architecture is able to achieve better performance than the cascaded architecture both in terms of accuracy and speed.

4.4.3. Ablation on backbones

Different from the previous salient object detection models which use backbones (*e.g.* VGG, ResNet, etc.) as their encoders, our newly proposed U^2-Net architecture is backbone free. To validate the backbone free design, we conduct ablation studies on replacing the encoder part of our full size U^2-Net with different backbones: VGG16 and ResNet50. Practically, we adapt the backbones (VGG-16 and ResNet-50) by adding an extra stage after their last convolutional stages to achieve the same receptive fields with our original U^2-Net architecture design. As shown in Table 2, the models using backbones and our RSUs as decoders achieve better performance than the previous ablations and comparable performance against our small size U^2-Net. However, they are still inferior to our full size U^2-Net. Therefore, we believe that our backbones free design is more competitive than backbones-based design in this salient object detection task.

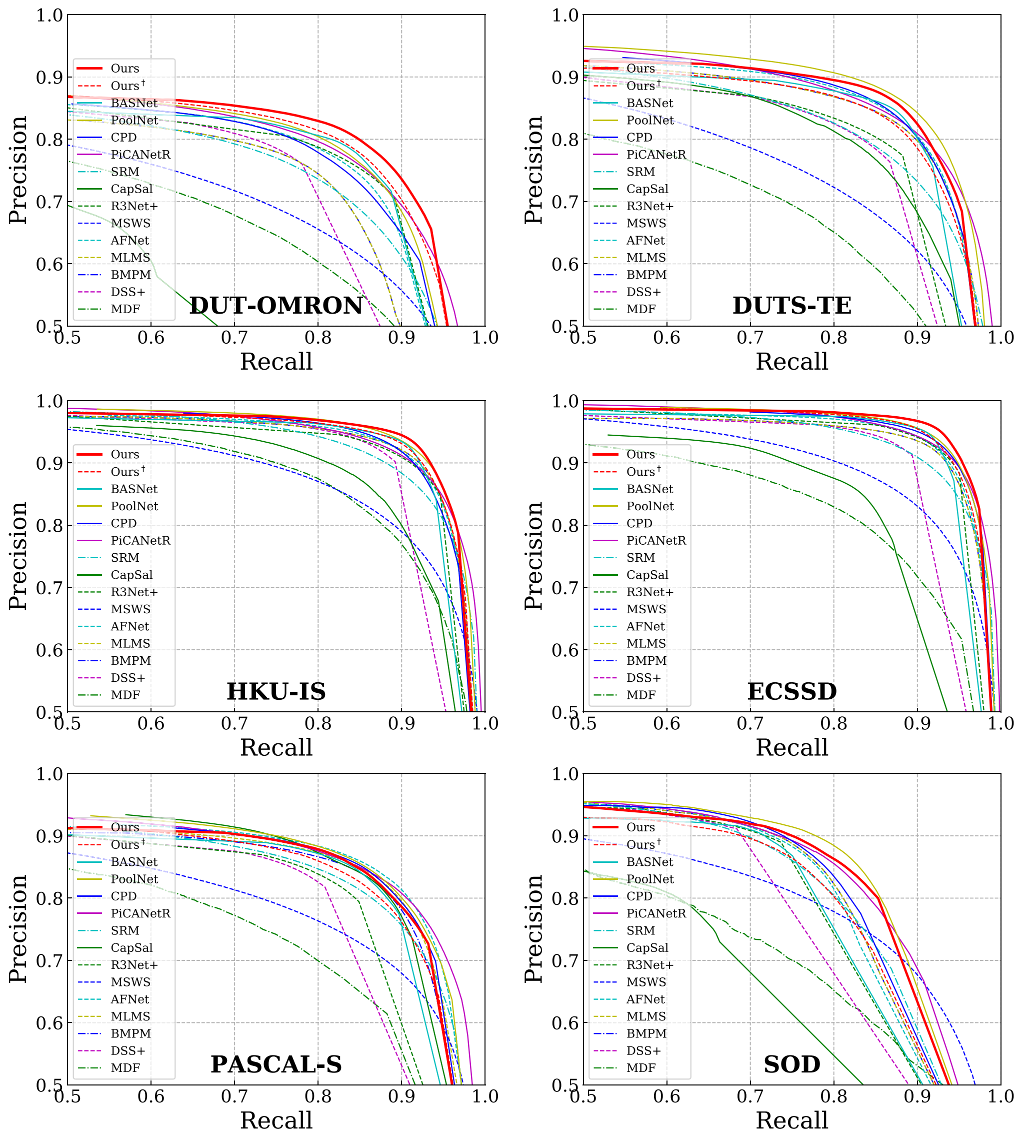

Fig. 6. Precision-Recall curves of our models and other typical state-of-the-art models on six SOD datasets.

4.5. Comparison with state-of-the-arts

We compare our models (full size U^2-Net, 176.3 MB and small size U^2-Net†, 4.7 MB) with 20 state-of-the-art methods including one **AlexNet** based model: **MDF**; 10 **VGG** based models: **UCF, Amulet, NLDF, DSS, RAS, PAGRN, BMPM, PiCANet, MLMS, AFNet**; one **DenseNet** based model **MSWS**; one **ResNeXt** based model: **R^{3}Net**; and seven **ResNet** based models: **CapSal, SRM, DGRL, PiCANetR, CPD, PoolNet, BASNet**. For fair comparison, we mainly use the salient object detection results provided by the authors. For the missing results on certain datasets of some methods, we run their released code with their trained models on their suggested environment settings.

4.5.1. Quantitative comparison

Fig. 6 illustrates the precision-recall curves of our models (U^2-Net, 176.3 MB and U^2-Net†, 4.7 MB) and typical state-of-the-art methods on the six datasets. The curves are consistent with the Tables 3 and 4 which demonstrate the state-of-the-art performance of our U^2-Net on DUT-OMRON, HKU-IS and ECSSD and competitive performance on other datasets. Tables 3 and 4 compares five (six include the model size) evaluation metrics and the model size of our proposed method with others. As we can see, our U^2-Net achieves the best performance on datasets DUT-OMRON, HKU-IS and ECSSD in terms of almost all of the five evaluation metrics. On DUTS-TE dataset our U^2-Net achieves the second best overall performance, which is slightly inferior to PoolNet. On PASCAL-S, the performance of our U^2-Net is slightly inferior to AFNet, CPD and PoolNet. It is worth noting that U^2-Net achieves the second best performance in terms of the boundary quality evaluation metric relaxF_{β}^{b}. On SOD dataset, PoolNet performs the best and our U^2-Net is the second best in terms of the overall performance.

Our U^2-Net† is only 4.7 MB, which is currently the smallest model in the field of salient object detection. With much fewer number of parameters against other models, it still achieves surprisingly competitive performance. Although its performance is not as good as our full size U^2-Net, its small size will facilitate its applications in many computation and memory constrained environments.

Table 3
Comparison of our method and 20 SOTA methods on DUT-OMRON, DUTS-TE, HKU-IS in terms of model size, *maxF*$_\beta$ (↑), *MAE* (↓), weighted F^w_β (↑), structure measure S_m (↑) and relax boundary F-measure *relax*F^b_β (↑). Red, Green, and Blue indicate the best, second best and third best performance.

Method	Backbone	Size(MB)	**DUT-OMRON** (5168)					**DUTS-TE** (5019)					**HKU-IS** (4447)				
			$maxF_\beta$	MAE	F^w_β	S_m	$relaxF^b_\beta$	$maxF_\beta$	MAE	F^w_β	S_m	$relaxF^b_\beta$	$maxF_\beta$	MAE	F^w_β	S_m	$relaxF^b_\beta$
MDF$_{\text{TIP16}}$	AlexNet	112.1	0.694	0.142	0.565	0.721	0.406	0.729	0.099	0.543	0.723	0.447	0.860	0.129	0.564	0.810	0.594
UCF$_{\text{ICCV17}}$	VGG-16	117.9	0.730	0.120	0.573	0.760	0.480	0.773	0.112	0.596	0.777	0.518	0.888	0.062	0.779	0.875	0.679
Amulet$_{\text{ICCV17}}$	VGG-16	132.6	0.743	0.098	0.626	0.781	0.528	0.778	0.084	0.658	0.796	0.568	0.897	0.051	0.817	0.886	0.716
NLDF+$_{\text{CVPR17}}$	VGG-16	428.0	0.753	0.080	0.634	0.770	0.514	0.813	0.065	0.710	0.805	0.591	0.902	0.048	0.838	0.879	0.694
DSS+$_{\text{CVPR17}}$	VGG-16	237.0	0.781	0.063	0.697	0.790	0.559	0.825	0.056	0.755	0.812	0.606	0.916	0.040	0.867	0.878	0.706
RAS$_{\text{ECCV18}}$	VGG-16	81.0	0.786	0.062	0.695	0.814	0.615	0.831	0.059	0.740	0.828	0.656	0.913	0.045	0.843	0.887	0.748
PAGRN$_{\text{CVPR18}}$	VGG-19	-	0.771	0.071	0.622	0.775	0.582	0.854	0.055	0.724	0.825	0.692	0.918	0.048	0.820	0.887	0.762
BMPM$_{\text{CVPR18}}$	VGG-16	-	0.774	0.064	0.681	0.809	0.612	0.852	0.048	0.761	0.851	0.699	0.921	0.039	0.859	0.907	0.773
PiCANet$_{\text{CVPR18}}$	VGG-16	153.3	0.794	0.068	0.691	0.826	0.643	0.851	0.054	0.747	0.851	0.704	0.921	0.042	0.847	0.906	0.784
MLMS$_{\text{CVPR19}}$	VGG-16	263.0	0.774	0.064	0.681	0.809	0.612	0.852	0.048	0.761	0.851	0.699	0.921	0.039	0.859	0.907	0.773
AFNet$_{\text{CVPR19}}$	VGG-16	143.0	0.797	0.057	0.717	0.826	0.635	0.862	0.046	0.785	0.855	0.714	0.923	0.036	0.869	0.905	0.772
MSWS$_{\text{CVPR19}}$	Dense-169	48.6	0.718	0.109	0.527	0.756	0.362	0.767	0.908	0.586	0.749	0.376	0.856	0.084	0.685	0.818	0.438
R^{3}Net+$_{\text{IJCAI18}}$	ResNeXt	215.0	0.795	0.063	0.728	0.817	0.599	0.828	0.058	0.763	0.817	0.601	0.915	0.036	0.877	0.895	0.740
CapSal$_{\text{CVPR19}}$	ResNet-101	-	0.699	0.101	0.482	0.674	0.396	0.823	0.072	0.691	0.808	0.605	0.882	0.062	0.782	0.850	0.654
SRM$_{\text{ICCV17}}$	ResNet-50	189.0	0.769	0.069	0.658	0.798	0.523	0.826	0.058	0.722	0.824	0.592	0.906	0.046	0.835	0.887	0.680
DGRL$_{\text{CVPR18}}$	ResNet-50	646.1	0.779	0.063	0.697	0.810	0.584	0.834	0.051	0.760	0.836	0.656	0.913	0.037	0.865	0.897	0.744
PiCANetR$_{\text{CVPR18}}$	ResNet-50	197.2	0.803	0.065	0.695	0.832	0.632	0.860	0.050	0.755	0.859	0.696	0.918	0.043	0.840	0.904	0.765
CPD$_{\text{CVPR19}}$	ResNet-50	183.0	0.797	0.056	0.719	0.825	0.655	0.865	0.043	0.795	0.858	0.741	0.925	0.034	0.875	0.905	0.795
PoolNet$_{\text{CVPR19}}$	ResNet-50	273.3	0.808	0.056	0.729	0.836	0.675	0.880	0.040	0.807	0.871	0.765	0.932	0.033	0.881	0.917	0.811
BASNet$_{\text{CVPR19}}$	ResNet-34	348.5	0.805	0.056	0.751	0.836	0.694	0.860	0.047	0.803	0.853	0.758	0.928	0.032	0.889	0.909	0.807
U^2-Net (Ours)	RSU	176.3	0.823	0.054	0.757	0.847	0.702	0.873	0.044	0.804	0.861	0.765	0.935	0.031	0.890	0.916	0.812
U^2-Net† (Ours)	RSU	4.7	0.813	0.060	0.731	0.837	0.676	0.852	0.054	0.763	0.847	0.723	0.928	0.037	0.867	0.908	0.794

Table 4
Comparison of our method and 20 SOTA methods on ECSSD, PASCAL-S, SOD in terms of model size, *maxF*$_\beta$ (↑), *MAE* (↓), weighted F^w_β (↑), structure measure S_m (↑) and relax boundary F-measure *relax*F^b_β (↑). Red, Green, and Blue indicate the best, second best and third best performance.

Method	Backbone	Size(MB)	**ECSSD** (1000)					**PASCAL-S** (850)					**SOD** (300)				
			$maxF_\beta$	MAE	F^w_β	S_m	$relaxF^b_\beta$	$maxF_\beta$	MAE	F^w_β	S_m	$relaxF^b_\beta$	$maxF_\beta$	MAE	F^w_β	S_m	$relaxF^b_\beta$
MDF$_{\text{TIP16}}$	AlexNet	112.1	0.832	0.105	0.705	0.776	0.472	0.759	0.142	0.589	0.696	0.343	0.746	0.192	0.508	0.643	0.311
UCF$_{\text{ICCV17}}$	VGG-16	117.9	0.903	0.069	0.806	0.884	0.669	0.814	0.115	0.694	0.805	0.493	0.808	0.148	0.675	0.762	0.471
Amulet$_{\text{ICCV17}}$	VGG-16	132.6	0.915	0.059	0.840	0.894	0.711	0.828	0.100	0.734	0.818	0.541	0.798	0.144	0.677	0.753	0.454
NLDF+$_{\text{CVPR17}}$	VGG-16	428.0	0.905	0.063	0.839	0.897	0.666	0.822	0.098	0.737	0.798	0.495	0.841	0.125	0.709	0.755	0.475
DSS+$_{\text{CVPR17}}$	VGG-16	237.0	0.921	0.052	0.872	0.882	0.696	0.831	0.093	0.759	0.798	0.499	0.846	0.124	0.710	0.743	0.444
RAS$_{\text{ECCV18}}$	VGG-16	81.0	0.921	0.056	0.857	0.893	0.741	0.829	0.101	0.736	0.799	0.560	0.851	0.124	0.720	0.764	0.544
PAGRN$_{\text{CVPR18}}$	VGG-19	-	0.927	0.061	0.834	0.889	0.747	0.847	0.090	0.738	0.822	0.594	-	-	-	-	-
BMPM$_{\text{CVPR18}}$	VGG-16	-	0.928	0.045	0.871	0.911	0.770	0.850	0.074	0.779	0.845	0.617	0.856	0.108	0.726	0.786	0.562
PiCANet$_{\text{CVPR18}}$	VGG-16	153.3	0.931	0.046	0.865	0.914	0.784	0.856	0.078	0.772	0.848	0.612	0.854	0.103	0.722	0.789	0.572
MLMS$_{\text{CVPR19}}$	VGG-16	263.0	0.928	0.045	0.871	0.911	0.770	0.855	0.074	0.779	0.844	0.620	0.856	0.108	0.726	0.786	0.562
AFNet$_{\text{CVPR19}}$	VGG-16	143.0	0.935	0.042	0.887	0.914	0.776	0.863	0.070	0.798	0.849	0.626	0.856	0.111	0.723	0.774	-
MSWS$_{\text{CVPR19}}$	Dense-169	48.6	0.878	0.096	0.716	0.828	0.411	0.786	0.133	0.614	0.768	0.289	0.800	0.167	0.573	0.700	0.231
R^{3}Net+$_{\text{IJCAI18}}$	ResNeXt	215.0	0.934	0.040	0.902	0.910	0.759	0.834	0.092	0.761	0.807	0.538	0.850	0.125	0.735	0.759	0.431
CapSal$_{\text{CVPR19}}$	ResNet-101	-	0.874	0.077	0.771	0.826	0.574	0.861	0.073	0.786	0.837	0.527	0.773	0.148	0.597	0.695	0.404
SRM$_{\text{ICCV17}}$	ResNet-50	189.0	0.917	0.054	0.853	0.895	0.672	0.838	0.084	0.758	0.834	0.509	0.843	0.128	0.670	0.741	0.392
DGRL$_{\text{CVPR18}}$	ResNet-50	646.1	0.925	0.042	0.883	0.906	0.753	0.848	0.074	0.787	0.839	0.569	0.848	0.106	0.731	0.773	0.502
PiCANetR$_{\text{CVPR18}}$	ResNet-50	197.2	0.935	0.046	0.867	0.917	0.775	0.857	0.076	0.777	0.854	0.598	0.856	0.104	0.724	0.790	0.528
CPD$_{\text{CVPR19}}$	ResNet-50	183.0	0.939	0.037	0.898	0.918	0.811	0.861	0.071	0.800	0.848	0.639	0.860	0.112	0.714	0.767	0.556
PoolNet$_{\text{CVPR19}}$	ResNet-50	273.3	0.944	0.039	0.896	0.921	0.813	0.865	0.075	0.798	0.832	0.644	0.871	0.102	0.759	0.797	0.606
BASNet$_{\text{CVPR19}}$	ResNet-34	348.5	0.942	0.037	0.904	0.916	0.826	0.856	0.076	0.798	0.838	0.660	0.851	0.113	0.730	0.769	0.603
U^2-Net (Ours)	RSU	176.3	0.951	0.033	0.910	0.928	0.836	0.859	0.074	0.797	0.844	0.657	0.861	0.108	0.748	0.786	0.613
U^2-Net† (Ours)	RSU	4.7	0.943	0.041	0.885	0.918	0.808	0.849	0.086	0.768	0.831	0.627	0.841	0.124	0.697	0.759	0.559

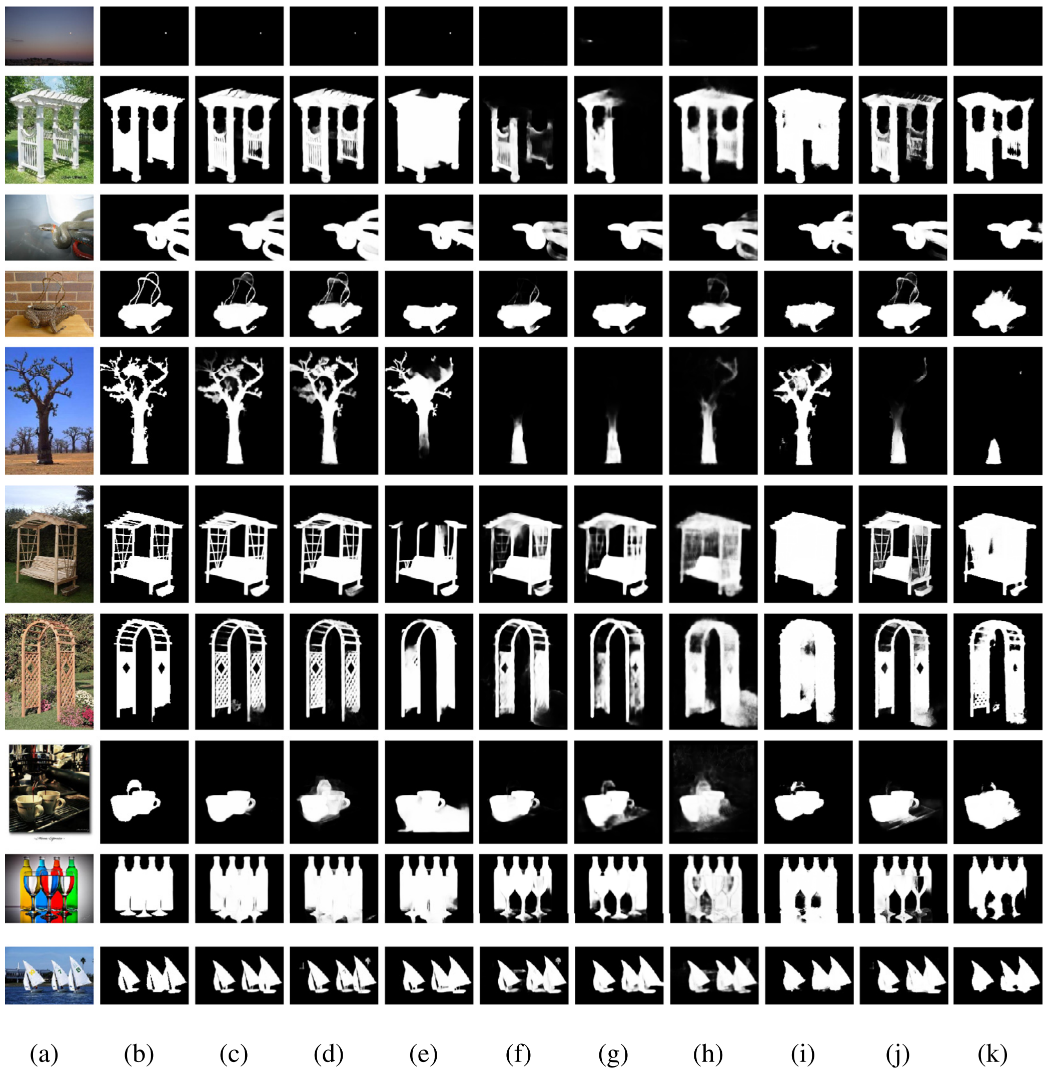

Fig. 7. Qualitative comparison of the proposed method with seven other SOTA methods: (a) image, (b) GT, (c) Ours, (d) Ours†, (e) BASNet, (f) PoolNet, (g) CPD, (h) PiCANetR, (i) R^{3}Net+, (j) AFNet, (k) DSS+, where '+' indicates the CRF post-processing.

4.5.2. *Qualitative comparison:*

To give an intuitive understanding of the promising performance of our models, we illustrate the sample results of our models and several other state-of-the-art methods in Fig. 7. As we can see, our U^2-Net and U^2-Net† are able to handle different types of targets and produce accurate salient object detection results.

The 1st and 2nd row of Fig. 7 show the results of small and large objects. As we can observe, our U^2-Net and U^2-Net† are able to produce accurate results on both small and large objects. Other models either prone to miss the small target or produce large object with poor accuracy. The 3rd row shows the results of target touching image borders. Our U^2-Net correctly segments all the regions. Although U^2-Net† erroneously segments the bottom right hole, it is still much better than other models. The 4th row demonstrates the performance of models in segmenting targets that consists of both large and thin structures. As we can see, most of other models extract large regions well while missing the cable-wise thin structure except for AFNet (col (j)). The 5th row shows a tree with relatively clean background of blue sky. It seems easy, but it is actually challenging to most of the models because of the complicated shape of the target. As we can see, our models segment both the trunk and branches well, while others fail in segmenting the complicated tree branch region. Compared with the 5th row, the bench shown in the 6th row is more complex thanks to the hollow structure. Our U^2-Net produces near perfect result. Although the bottom right of the prediction map of U^2-Net† is imperfect, its overall performance on this target is much better than other models. Besides, the results of our models are more homogenous with fewer gray areas than models like PoolNet (col (f)), CPD (col (g)), PiCANetR (col (h)) and AFNet (col (j)). The 7th row shows that our models can produce results even finer than the ground truth. Labeling these small holes in the 7th image is burdensome and time-consuming. Hence, these repeated fine structures are usually ignored in the annotation process. Inferring the correct results from these imperfect labeling is challenging. But our models show promising capability in segmenting these fine structures thanks to the well designed architectures for extracting and integrating high resolution local and low resolution global information. The 8th and 9th row are illustrated to show the strong ability of our models in detecting targets with cluttered backgrounds and complicated foreground appearance. The 10th row shows that our models are able to segment multiple targets while capturing the details of the detected targets (see the gap region of the two pieces of sail of each sailboat). In summary, both our full size and small size models are able to handle various scenarios and produce high accuracy salient object detection results.

5. Conclusions

In this paper, we proposed a novel deep network: U^2-Net, for salient object detection. The main architecture of our U^2-Net is a two-level nested U-structure. The nested U-structure with our newly designed RSU blocks enables the network to capture richer local and global information from both shallow and deep layers regardless of the resolutions. Compared with those SOD models built upon the existing backbones, our U^2-Net is purely built on the proposed RSU blocks which makes it possible to be trained from scratch and configured to have different model size according to the target environment constraints. We provide a full size U^2-Net (176.3 MB, 30 FPS) and a smaller size version U^2-Net† (4.7 MB, 40 FPS) in this paper. Experimental results on six public salient object detection datasets demonstrate that both models achieve very competitive performance against other 20 state-of-the-art methods in terms of both qualitative and quantitative measures.

Although our models achieve competitive results against other state-of-the-art methods, faster and smaller models are needed for computation and memory limited devices, such as mobile phones, robots, etc. In the near future, we will explore different techniques and architectures to further improve the speed and decrease the model size. In addition, larger diversified salient object datasets are needed to train more accurate and robust models.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

This work is supported by the Alberta Innovates Graduate Student Scholarship and Natural Sciences and Engineering Research Council of Canada (NSERC) Discovery Grants Program, NO.: 2016-06365.

References

[1] J. Long, E. Shelhamer, T. Darrell, Fully convolutional networks for semantic segmentation, in: Proceedings of the IEEE Conference on Computer Vision and pattern Recognition, 2015, pp. 3431–3440.
[2] G. Li, Y. Yu, Visual saliency detection based on multiscale deep CNN features, IEEE Trans. Image Process. 25 (11) (2016) 5012–5024.
[3] Z. Luo, A. Mishra, A. Achkar, J. Eichel, S. Li, P.-M. Jodoin, Non-local deep features for salient object detection, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2017, pp. 6593–6601.
[4] 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: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2018, pp. 3127–3135.
[5] Z. Deng, X. Hu, L. Zhu, X. Xu, J. Qin, G. Han, P.-A. Heng, R3net: recurrent residual refinement network for saliency detection, in: Proceedings of the 27th International Joint Conference on Artificial Intelligence, AAAI Press, 2018, pp. 684–690.
[6] A. Krizhevsky, I. Sutskever, G.E. Hinton, ImageNet classification with deep convolutional neural networks, in: Advances in Neural Information Processing Systems, 2012, pp. 1097–1105.
[7] K. Simonyan, A. Zisserman, Very deep convolutional networks for large-scale image recognition, arXiv preprint arXiv:1409.1556(2014).
[8] K. He, X. Zhang, S. Ren, J. Sun, Deep residual learning for image recognition, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2016, pp. 770–778.
[9] S. Xie, R. Girshick, P. Dollár, Z. Tu, K. He, Aggregated residual transformations for deep neural networks, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2017, pp. 5987–5995.
[10] G. Huang, Z. Liu, L. van der Maaten, K.Q. Weinberger, Densely connected convolutional networks, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2017, pp. 2261–2269.
[11] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, L. Fei-Fei, ImageNet: a large-scale hierarchical image database, in: 2009 IEEE Conference on Computer Vision and Pattern Recognition, IEEE, 2009, pp. 248–255.
[12] Q. Yan, L. Xu, J. Shi, J. Jia, Hierarchical saliency detection, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2013, pp. 1155–1162.
[13] J.-J. Liu, Q. Hou, M.-M. Cheng, J. Feng, J. Jiang, A simple pooling-based design for real-time salient object detection, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2019, pp. 3917–3926.
[14] X. Qin, Z. Zhang, C. Huang, C. Gao, M. Dehghan, M. Jagersand, BASNet: boundary-aware salient object detection, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2019, pp. 7479–7489.
[15] J. Zhang, K.A. Ehinger, H. Wei, K. Zhang, J. Yang, A novel graph-based optimization framework for salient object detection, Pattern Recognit. 64 (2017) 39–50.
[16] J. Liang, J. Zhou, L. Tong, X. Bai, B. Wang, Material based salient object detection from hyperspectral images, Pattern Recognit. 76 (2018) 476–490.
[17] Q. Zhang, Z. Huo, Y. Liu, Y. Pan, C. Shan, J. Han, Salient object detection employing a local tree-structured low-rank representation and foreground consistency, Pattern Recognit. 92 (2019) 119–134.
[18] S. Lu, C. Tan, J.-H. Lim, Robust and efficient saliency modeling from image co-occurrence histograms, IEEE Trans. Pattern Anal. Mach. Intell. 36 (1) (2013) 195–201.
[19] S. Lu, J.-H. Lim, Saliency modeling from image histograms, in: European Conference on Computer Vision, Springer, 2012, pp. 321–332.
[20] S. Xie, Z. Tu, Holistically-nested edge detection, in: Proceedings of the IEEE International Conference on Computer Vision, 2015, pp. 1395–1403.
[21] L. Zhang, J. Dai, H. Lu, Y. He, G. Wang, A bi-directional message passing model for salient object detection, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2018, pp. 1741–1750.
[22] P. Zhang, D. Wang, H. Lu, H. Wang, X. Ruan, Amulet: aggregating multi-level convolutional features for salient object detection, in: Proceedings of the IEEE International Conference on Computer Vision, 2017, pp. 202–211.
[23] P. Zhang, D. Wang, H. Lu, H. Wang, B. Yin, Learning uncertain convolutional features for accurate saliency detection, in: Proceedings of the IEEE International Conference on Computer Vision, 2017, pp. 212–221.
[24] Q. Hou, M.-M. Cheng, X. Hu, A. Borji, Z. Tu, P. Torr, Deeply supervised salient object detection with short connections, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2017, pp. 5300–5309.
[25] S. Chen, X. Tan, B. Wang, X. Hu, Reverse attention for salient object detection, in: Proceedings of the European Conference on Computer Vision (ECCV), 2018, pp. 234–250.
[26] R. Wu, M. Feng, W. Guan, D. Wang, H. Lu, E. 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, 2019, pp. 8150–8159.
[27] Z. Wu, L. Su, Q. Huang, Cascaded partial decoder for fast and accurate salient object detection, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2019, pp. 3907–3916.
[28] T. Wang, A. Borji, L. Zhang, P. Zhang, H. Lu, A stagewise refinement model for detecting salient objects in images, in: Proceedings of the IEEE International Conference on Computer Vision, 2017, pp. 4039–4048.
[29] H. Zhao, J. Shi, X. Qi, X. Wang, J. Jia, Pyramid scene parsing network, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2017, pp. 2881–2890.
[30] X. Zhang, T. Wang, J. Qi, H. Lu, G. Wang, Progressive attention guided recurrent network for salient object detection, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2018, pp. 714–722.
[31] N. Liu, J. Han, M.-H. Yang, PiCANet: learning pixel-wise contextual attention for saliency detection, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2018, pp. 3089–3098.
[32] L. Zhang, J. Zhang, Z. Lin, H. Lu, Y. He, CapSal: leveraging captioning to boost semantics for salient object detection, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2019, pp. 6024–6033.
[33] Y. Zeng, Y. Zhuge, H. Lu, L. Zhang, M. Qian, Y. Yu, Multi-source weak supervision for saliency detection, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2019, pp. 6074–6083.
[34] M. Feng, H. Lu, E. Ding, Attentive feedback network for boundary-aware salient object detection, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2019, pp. 1623–1632.
[35] O. Ronneberger, P. Fischer, T. Brox, U-Net: convolutional networks for biomedical image segmentation, in: International Conference on Medical Image Computing and Computer-Assisted Intervention, Springer, 2015, pp. 234–241.
[36] L. Wang, H. Lu, Y. Wang, M. Feng, D. Wang, B. Yin, X. Ruan, Learning to detect salient objects with image-level supervision, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2017, pp. 136–145.
[37] C. Yang, L. Zhang, H. Lu, X. Ruan, M.-H. Yang, Saliency detection via graph-based manifold ranking, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2013, pp. 3166–3173.
[38] Y. Li, X. Hou, C. Koch, J.M. Rehg, A.L. Yuille, The secrets of salient object segmentation, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2014, pp. 280–287.
[39] V. Movahedi, J.H. Elder, Design and perceptual validation of performance measures for salient object segmentation, in: 2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition-Workshops, IEEE, 2010, pp. 49–56.
[40] R. Achanta, S. Hemami, F. Estrada, S. Susstrunk, Frequency-tuned salient region detection, in: 2009 IEEE Conference on Computer Vision and Pattern Recognition, 2009, pp. 1597–1604.
[41] R. Margolin, L. Zelnik-Manor, A. Tal, How to evaluate foreground maps, in: 2014 IEEE Conference on Computer Vision and Pattern Recognition, 2014, pp. 248–255.

[42] D.-P. Fan, M.-M. Cheng, Y. Liu, T. Li, A. Borji, Structure-measure: a new way to evaluate foreground maps, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2017, pp. 4548–4557.
[43] A. Newell, K. Yang, J. Deng, Stacked hourglass networks for human pose estimation, in: European Conference on Computer Vision, Springer, 2016, pp. 483–499.
[44] Z. Tang, X. Peng, S. Geng, L. Wu, S. Zhang, D. Metaxas, Quantized densely connected U-Nets for efficient landmark localization, in: Proceedings of the European Conference on Computer Vision (ECCV), 2018, pp. 339–354.

Xuebin Qin obtained his M.Sc. degree from the Peking University, Beijing, China, in 2015. Since September, 2015, He is a Ph.D. student in the department of Computing Science, University of Alberta, Canada. His research interests include tracking, segmentation, detection and their applications in robotics. He has published several papers in vision and robotics conferences such as CVPR, BMVC, ICPR, WACV and IROS.

Zichen Zhang is a Ph.D. student in Computing Science with a Specialization in Statistical Machine Learning at the University of Alberta. He obtained his M.Sc degrees from Dalhousie University and the University of Alberta and B.E degree from Huazhong University of Science and Technology. He's interested in machine learning and its applications in robotics perception and control.

Chenyang Huang obtained his M.Sc. degree from the University of Alberta, Edmonton, Canada, in 2019. He is currently pursuing a Ph.D. degree in the Department of Computing Science of the same university. His research is mainly focusing on deep learning, natural language processing, and computer vision. He has publications on some prestigious conferences such as NAACL and CVPR.

Masood Dehghan obtained his Ph.D. degree from the National University of Singapore, in 2013. He is a senior research fellow in the department of Computing Science, University of Alberta. His research interests include vision-guided robotics, hybrid vision-force robot control and computer vision.

Osmar R. Zaïane is a Professor in Computing Science at the University of Alberta, Canada, and Director of the Alberta Machine Intelligence Institute (Amii). Dr. Zaiane obtained his Ph.D. from Simon Fraser University, Canada, in 1999. He has published more than 300 papers in refereed international conferences and journals. He is Associate Editor of many International Journals on data mining and data analytics and served as program chair and general chair for scores of international conferences in the field of knowledge discovery and data mining. Dr. Zaiane received numerous awards including the 2010 ACM SIGKDD Service Award from the ACM Special Interest Group on Data Mining, which runs the world's premier data science, big data, and data mining association and conference.

Martin Jagersand's research interests are in Robotics, Computer Vision, and Graphics, especially vision guided motion control and vision-based human-robot interfaces. He studied physics at Chalmers Sweden (MSc 1991). He was awarded a Fulbright fellowship for graduate studies in the USA. He studied Computer Science at the Univ. of Rochester, NY (M.Sc. 1994, Ph.D. 1997). He held an NSF CISE postdoc fellowship, at Yale University, and then was a research faculty in the Engineering Research Center for Surgical Systems and Technology at Johns Hopkins University. He is now a faculty member at the University of Alberta.