PiCANet: Pixel-Wise Contextual Attention Learning for Accurate Saliency Detection

Nian Liu, *Member, IEEE*, Junwei Han, *Senior Member, IEEE*, and Ming-Hsuan Yang, *Fellow, IEEE*

***Abstract*— Existing saliency models typically incorporate contexts holistically. However, for each pixel, usually only part of its context region contributes to saliency prediction, while other parts are likely either noise or distractions. In this paper, we propose a novel pixel-wise contextual attention network (PiCANet) to selectively attend to informative context locations at each pixel. The proposed PiCANet generates an attention map over the contextual region of each pixel and construct attentive contextual features via selectively incorporating the features of useful context locations. We present three formulations of the PiCANet via embedding the pixel-wise contextual attention mechanism into the pooling and convolution operations with attending to global or local contexts. All the three models are fully differentiable and can be integrated with convolutional neural networks with joint training. In this work, we introduce the proposed PiCANets into a U-Net model for salient object detection. The generated global and local attention maps can learn to incorporate global contrast and regional smoothness, which help localize and highlight salient objects more accurately and uniformly. Experimental results show that the proposed PiCANets perform effectively for saliency detection against the state-of-the-art methods. Furthermore, we demonstrate the effectiveness and generalization ability of the PiCANets on semantic segmentation and object detection with improved performance.**

***Index Terms*— saliency detection, attention network, global context, local context, semantic segmentation, object detection.**

I. Introduction

CONTEXTUAL information plays a crucial role in saliency detection, which is typically reflected in the form of contrast. In [1], Itti *et al*. propose to compute the feature difference between each pixel and its surrounding regions in a Gaussian pyramid as contrast. Some other models [2]–[4] also employ the contrast mechanism to model visual saliency. In these methods, local context or global context is utilized as the reference to evaluate the contrast of each image pixel, which is referred as the local contrast or the global contrast, respectively. Generally, a feature representation is first extracted for each image pixel. The features of all the referred contextual locations are aggregated into an overall representation as the contextual feature to infer contrast.

(a) (b) (c)

Fig. 1. **Generated global and local pixel-wise contextual attention maps for two example pixels.** (a) shows the original image with two example pixels, in which the white one locates on the foreground dog while the blue one locates on the background. (b) and (c) show their generated global and local contextual attention maps, respectively. Hot color indicates large attention weights. For each case, the referred context region is given by the red box.

Recently, numerous saliency detection methods based on convolutional neural networks (CNNs) have been developed. While a few methods [5]–[7] operate on each pixel or superpixel where deep features are extracted from multiscale image regions and combined for saliency detection, other approaches [8]–[22] extract multilevel features from fully convolutional networks (FCNs) [23] and use various neural network modules to infer saliency by combining the feature maps. Most of the methods mentioned above utilize the entire context regions to construct the contextual features due to the fixed intrinsic structure of CNNs.

In most existing methods, context regions are holistically leveraged, i.e., every context location contributes to the contextual feature construction. Intuitively this is a sub-optimal choice since most pixels have both useful and useless context parts. In a given context region of a specific pixel, some of its context locations contain relevant information and contribute to its final prediction, while some others are irrelevant and may serve as distractions. We give an intuitive example in Figure 1. For the white dot on the foreground dog in the top row, we can infer its global contrast by comparing it with the background regions. While by referring to the other parts of

Manuscript received July 24, 2019; revised February 12, 2020; accepted April 6, 2020. Date of publication April 23, 2020; date of current version July 2, 2020. This work was supported in part by the National Science Foundation of China under Grant U1801265, in part by the Key Research and Development Program of Guangdong Province under Grant 2019B010110001, in part by the Research Funds for Interdisciplinary Subject, NWPU, and in part by NSF CAREER under Grant 1149783. The associate editor coordinating the review of this manuscript and approving it for publication was Prof. Jing-Ming Guo. *(Corresponding author: Junwei Han.)*

Nian Liu is with the School of Automation, Northwestern Polytechnical University, Xi'an 710072, China, and also with the Department of Engagement Services, Mohamed Bin Zayed University of Artificial Intelligence, AbuDhabi, United Arab Emirates (e-mail: liunian228@gmail.com).

Junwei Han is with the School of Automation, Northwestern Polytechnical University, Xi'an 710072, China (e-mail: junweihan2010@gmail.com).

Ming-Hsuan Yang is with the School of Engineering, University of California at Merced, Merced, CA 95343 USA (e-mail: mhyang@ucmerced.edu).

Digital Object Identifier 10.1109/TIP.2020.2988568

the dog, we can also conclude that this pixel is part of the foreground dog, and uniformly recognize the whole body of the dog. We can also infer the global contrast and belonging of the blue dot in the second row similarly. Clearly, it is of great importance to use relevant contextual regions, instead of using all areas holistically, for saliency detection.

In this work, we propose a Pixel-wise Contextual Attention Network (PiCANet) to extract useful context regions for each image pixel, which is different from the traditional soft attention model [24]. The proposed PiCANet simultaneously generates an attention map at each pixel over its context region. As such, the relevance of the context locations with respect to the referred pixel are encoded in the corresponding attention weights. For each pixel, we use the weights to selectively aggregate the features of its context locations and obtain an attentive contextual feature. Thus our model only incorporates useful contextual knowledge and depresses other noisy and distractive information for each pixel, which significantly facilitates saliency prediction. The examples in Figure 1 illustrate different attention maps generated by our model for different pixels.

We design three forms of the PiCANet with contexts of different scopes and usage of different attention mechanisms. The first two are based on weighted average to pool global and local contextual features for linearly aggregating the contexts. We refer them as *global attention pooling* (GAP) and *local attention pooling* (LAP). The third one adopts local attention weights in the convolution operation to control the information flow for the convolutional feature extraction at each pixel. We refer this form of the PiCANet as *attention convolution* (AC). All three PiCANets are fully differentiable and can be flexibly embedded into CNNs with end-to-end training.

Based on the proposed PiCANet models, we construct a network by hierarchically embedding them into a U-Net [25] architecture for salient object detection. In this work, we progressively adopt global and local PiCANets in different decoder modules with multiscale feature maps, thereby constructing attentive contextual features with varying context scopes and scales. As a result, saliency inference can be facilitated by these enhanced features. Aside from saliency detection, we also validate the effectiveness of PiCANets on semantic segmentation and object detection based on widely used baseline models. The results demonstrate that PiCANets can be used as general neural network modules for dense prediction tasks.

The contributions of this work are summarized as follows. First, we propose PiCANets to select informative context regions and construct attentive contextual features for each pixel to facilitate saliency prediction. Second, we design three differentiable formulations for PiCANets, where the pixel-wise contextual attention mechanism is introduced into pooling and convolution operations with attention over global or local contexts. Third, we embed the PiCANets into a U-Net architecture to hierarchically incorporate attentive global and multiscale local contexts for salient object detection. Experimental results demonstrate the effectiveness of the proposed PiCANets and our saliency model when compared with other state-of-the-art methods. We also analyze the function of the learned global and local attention maps. Fourth, we apply the proposed PiCANets to semantic segmentation and object detection, thus demonstrating their effectiveness and generalization ability to other vision tasks.

Preliminary results of this work were presented in [26] and the differences are summarized as follows. First, based on the two forms of the PiCANet proposed in [26], we propose the third formulation by introducing the pixel-wise contextual attention into the convolution operation to modulate the information flow. Experimental results show that it achieves more performance gains for saliency detection. Second, we add explicit supervision for the learning of global attention, which can help to learn global contrast better and improve model performance. Third, our new model obtains better results than [26] and performs favorably against other state-of-the-art methods. Fourth, we conduct experiments on semantic segmentation and object detection to validate the effectiveness and the generalization ability of the proposed PiCANets.

II. Related Work

A. Attention Networks

Attention models have been applied to various vision tasks. In [27], Xu *et al*. adopt a recurrent attention model to find relevant image regions for image captioning. Sermanet *et al*. [28] propose to select discriminative image regions for fine-grained classification via a recurrent attention model. Similarly, several visual question answering models [29], [30] use attention networks to extract features from question-related image regions. In object detection, Li *et al*. [31] adopt an attention model to incorporate target-related regions in global context for optimizing object classification. Hu *et al*. [32] propose to generate channel attention to select useful feature channels for image classification. In [33], three types of attention models are proposed to generate attention for each channel or spatial position or both of them with residual connections. Woo *et al*. [34] sequentially adopt channel attention and spatial attention to refine convolutional features in CNNs. While attention models are demonstrated to be helpful in learning more discriminative feature representations via finding informative image regions or feature channels, these models only generate one attention map over the whole image (or generate one attention map at each time in a recurrent model). Namely, the above existing models are only optimized to generate *image-wise attention*.

For dense prediction tasks (*e.g*, semantic segmentation and saliency detection), it is of interest to generate one attention map for each pixel since different pixels have different useful context regions. Nevertheless, previous work does not exploit this idea until recently. In [35], Wang *et al*. propose the Non-local (NL) model to generate a contextual attention map over the global feature map at each position, where the attention map is obtained by computing each attention weight separately using each query-key position pair. The feature at each position is augmented with the attentive weighted sum of the features for the entire image. However, this model is computationally inefficient since all positions of the whole feature map are involved in the attention model. To alleviate this problem, Huang *et al*. [36] propose to generate attention maps and

propagate contextual information on the criss-cross path, thus improving both effectiveness and efficiency. In [37], [38], the computational load of capturing long-range relation in attention models is reduced by finding a compact set of global attention bases with using the second-order pooling method and expectation-maximization algorithm, respectively. On the other hand, He *et al*. [39] aggregate contextual features of multiscale sub-regions with global-guided local affinity coefficients. In the PSANet [40], two attention maps at each position are generated to collect the contextual information of all other positions for the current position and distribute the current feature to all other positions, respectively. The technical differences between the proposed PiCANets and these methods are discussed in Section III-E.

B. Saliency Detection With Deep Learning

Recently, numerous saliency detection models have shown promising results with the adoption of deep neural networks. Most existing models use diverse networks to combine multilevel global or local contexts holistically without further distinguishing useful context regions. In [5], [7], [41], CNNs are used to extract multiscale contextual features from cropped image patches, where all pixels in the patches are involved in the contextual feature construction. Recent deep saliency models usually extract intermediate multilevel convolutional feature maps from a CNN encoder first, and then fuse multiscale contextual features with various modules. Liu and Han [10] use recurrent convolutional layers [42] in a U-Net architecture to hierarchically fuse previous saliency maps with local features. In [17], Wang *et al*. adopt the convLSTM [43] in a similar refinement network. Hou *et al*. [12] propose to use dense connections to fuse multiscale features. In [14], Zhang *et al*. adopt a resolution-based feature integration module to fuse multiscale features via multiple downsampling and upsampling branches. All the above-mentioned feature fusion schemes use all positions in the feature maps for fusing contextual features. In contrast, we propose the PiCANets to only incorporate informative context regions.

We note there are other saliency models using attention mechanisms. Specifically, Kuen *et al*. [8] employ a recurrent attention model to select local regions for refining the saliency map. Zhang *et al*. [20] generate spatial and channel attention for each feature map. In [19], [22], a spatial attention map is adopted in each decoding module to weight the feature map. In [18], Zhang *et al*. use the element-wise gating attention to control the message passing among different feature maps. As these models generate attention once for the whole feature map in each module, they all fall into the *image-wise attention* category. In contrast, PiCANets are designed to generate each pixel a contextual attention map simultaneously, and thus are more suitable for saliency detection.

III. PiCANet Model

In this section, we present three forms of the proposed PiCANet. Suppose we have a convolutional (Conv) feature map $\boldsymbol{F} \in \mathbb{R}^{W \times H \times C}$, with W, H, and C denoting its width, height and number of channels, respectively. For each location (w, h) in $\boldsymbol{F}$, the GAP module generates global attention over the entire feature map $\boldsymbol{F}$, while the LAP module and the AC module generate attention over a local neighbouring region $\bar{\boldsymbol{F}}^{w,h}$ centered at (w, h). For the attention mechanism, the GAP and LAP modules first use softmax to generate normalized attention weights and then adopt weighted average to pool the features in the context region, whereas the AC mechanism generates attention weights modulated by sigmoid and uses them as gates to control the information flow of each context location in convolution.

A. Global Attention Pooling

The GAP network architecture is shown in Figure 2(a). To enable all pixels to generate their own global attention, we first need them to be capable to "see" the whole feature map $\boldsymbol{F}$, where a network with the entire image as its receptive field is required. Although a fully connected layer is a straightforward choice, it entails learning a large number of parameters. Alternatively, we use the ReNet model [44] as a more efficient and effective mechanism to perceive the global context, as shown in the orange dashed box in Figure 2(a). Specifically, two LSTMs [45] along each row of $\boldsymbol{F}$ scan the pixels one-by-one from left to right and from right to left, respectively. The obtained two feature maps are then concatenated to combine both left and right contextual information of each pixel. Next, the ReNet uses another two LSTMs to scan each column of the obtained feature map in both bottom-up and top-down orders. Similarly, the two obtained feature maps are concatenated to combine both bottom and top contexts. By successively using horizontal and vertical bidirectional LSTMs to scan the feature maps, each pixel can encode its contextual information from all four directions, thereby effectively integrating the global context. As the ReNet can execute each bidirectional LSTM in parallel and share the parameters for each pixel, the process is computationally efficient.

Based on the above-described ReNet module, a 1×1 Conv layer is used to transform the output feature map to D channels, where typically $D = W \times H$. We use the softmax activation function to compute the normalized attention weights $\boldsymbol{\alpha} \in \mathbb{R}^{W \times H \times D}$. Specifically, at a pixel (w, h) with the transformed feature $\boldsymbol{x}^{w,h}$, the attention weights $\boldsymbol{\alpha}^{w,h}$ can be obtained by:

$$\alpha_i^{w,h} = \frac{\exp\left(x_i^{w,h}\right)}{\sum_{j=1}^{D} \exp\left(x_j^{w,h}\right)}, \tag{1}$$

where $i \in \{1, \ldots, D\}$, $\boldsymbol{x}^{w,h}$ and $\boldsymbol{\alpha}^{w,h} \in \mathbb{R}^D$. In (1), $\alpha_i^{w,h}$ represents the attention weight of the i^{th} location in $\boldsymbol{F}$ with respect to the pixel (w, h).

For each pixel, as shown in Figure 2(b), we use its attention weights to pool the features in $\boldsymbol{F}$ via weighted average. As a result, we obtain an attentive contextual feature map $\boldsymbol{F}_{GAP}$. At each location we have:

$$\boldsymbol{F}_{GAP}^{w,h} = \sum_{i=1}^{D} \alpha_i^{w,h} \boldsymbol{f}_i, \tag{2}$$

where $\boldsymbol{f}_i \in \mathbb{R}^C$ is the feature at the i^{th} location of $\boldsymbol{F}$.

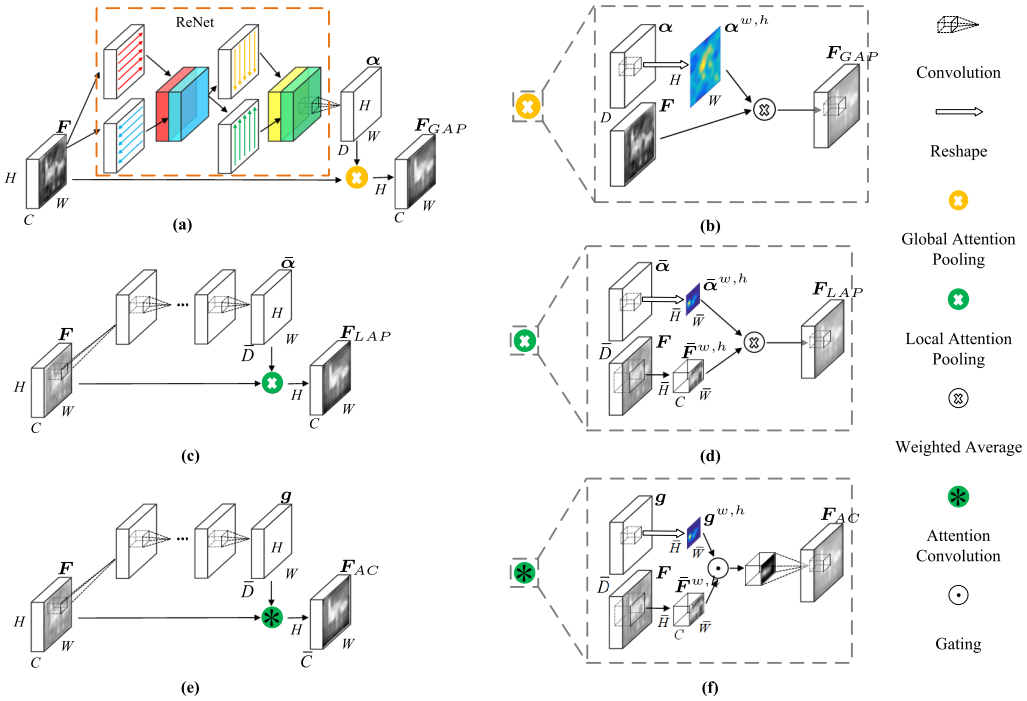


Fig. 2. **Illustration of the proposed PiCANets.** (a)(c)(e) illustrate the proposed global attention pooling, local attention pooling, and attention convolution network architectures, respectively. (b)(d)(f) show detailed operations of GAP, LAP, and AC, respectively.

This operation is similar to the commonly used pooling layer in CNNs, except that we use the generated attention weights to adaptively pool features for each specific context instead of using fixed pooling templates in each pooling window.

B. Local Attention Pooling

The design of the LAP module is similar to GAP except that it only operates over a local neighbouring region for each pixel, as shown in Figure 2(c). Given the width $\bar{W}$ and the height $\bar{H}$ of the local region, we use several Conv layers on top of $\boldsymbol{F}$ with their receptive field size achieving $\bar{W} \times \bar{H}$. Thus, we enable each pixel (w, h) to "see" the local neighbouring region $\bar{\boldsymbol{F}}^{w,h} \in \mathbb{R}^{\bar{W} \times \bar{H} \times C}$ centered at it. Then, similar to GAP, we use another Conv layer with $\bar{D} = \bar{W} \times \bar{H}$ channels and the softmax activation function to compute the local attention weights $\bar{\boldsymbol{\alpha}} \in \mathbb{R}^{W \times H \times \bar{D}}$. For each pixel (w, h), as shown in Figure 2(d), we use its attention weights $\bar{\boldsymbol{\alpha}}^{w,h}$ to compute the attentive contextual feature $\boldsymbol{F}_{LAP}^{w,h}$ as the weighted average of $\bar{\boldsymbol{F}}^{w,h}$:

$$\boldsymbol{F}_{LAP}^{w,h} = \sum_{i=1}^{\bar{D}} \bar{\alpha}_i^{w,h} \bar{\boldsymbol{f}}_i^{w,h}, \tag{3}$$

where $\bar{\boldsymbol{f}}_i^{w,h}$ is the feature at the i^{th} location of $\bar{\boldsymbol{F}}^{w,h}$, and $\boldsymbol{F}_{LAP} \in \mathbb{R}^{W \times H \times C}$.

C. Attention Convolution

Similar to the LAP model, the proposed AC module also generates and utilizes local attention for each pixel. The difference is that the AC module generates sigmoid attention weights and use them as gates to control whether each context location needs to be involved in the convolutional feature extraction for the center pixel. The detailed network architecture is shown in Figure 2(e). Given the Conv kernel size $\bar{W} \times \bar{H}$ and the number of output channels $\bar{C}$, similar Conv layers are first used as in the LAP module to generate local attention gates $\boldsymbol{g} \in \mathbb{R}^{W \times H \times \bar{D}}$ except that we use the sigmoid activation function for the last Conv layer. Similar to (1), we have:

$$g_i^{w,h} = \frac{1}{1 + \exp\left(-x_i^{w,h}\right)}, \tag{4}$$

where $i \in \{1, \ldots, \bar{D}\}$, and $g_i^{w,h}$ is the attention gate of the i^{th} location in $\bar{\boldsymbol{F}}^{w,h}$, determining whether its information should flow to the next layer for the feature extraction at (w, h).

We add $\boldsymbol{g}$ into a convolution layer on top of $\boldsymbol{F}$ where the detailed operations are shown in Figure 2(f). At pixel (w, h), we first use the attention gates $\boldsymbol{g}^{w,h}$ to modulate the features in $\bar{\boldsymbol{F}}^{w,h}$ via pixel-wise multiplication. We then multiply the result feature matrix with the convolution weight matrix $\boldsymbol{W} \in \mathbb{R}^{\bar{W} \times \bar{H} \times C \times \bar{C}}$ to compute the attentive contextual feature $\boldsymbol{F}_{AC}^{w,h}$. By decomposing convolution into per-location operation, we have:

$$\boldsymbol{F}_{AC}^{w,h} = \sum_{i=1}^{\bar{D}} g_i^{w,h} \bar{\boldsymbol{f}}_i^{w,h} \boldsymbol{W}_i + \boldsymbol{b}, \tag{5}$$

where $\boldsymbol{W}_i \in \mathbb{R}^{C \times \bar{C}}$ is the i^{th} spatial element of $\boldsymbol{W}$ and $\boldsymbol{b} \in \mathbb{R}^{\bar{C}}$ is the convolution bias. We note the computed attentive feature map $\boldsymbol{F}_{AC} \in \mathbb{R}^{W \times H \times \bar{C}}$.

Compared to LAP, the AC module further introduces non-linear transformation on top of the attended features, which

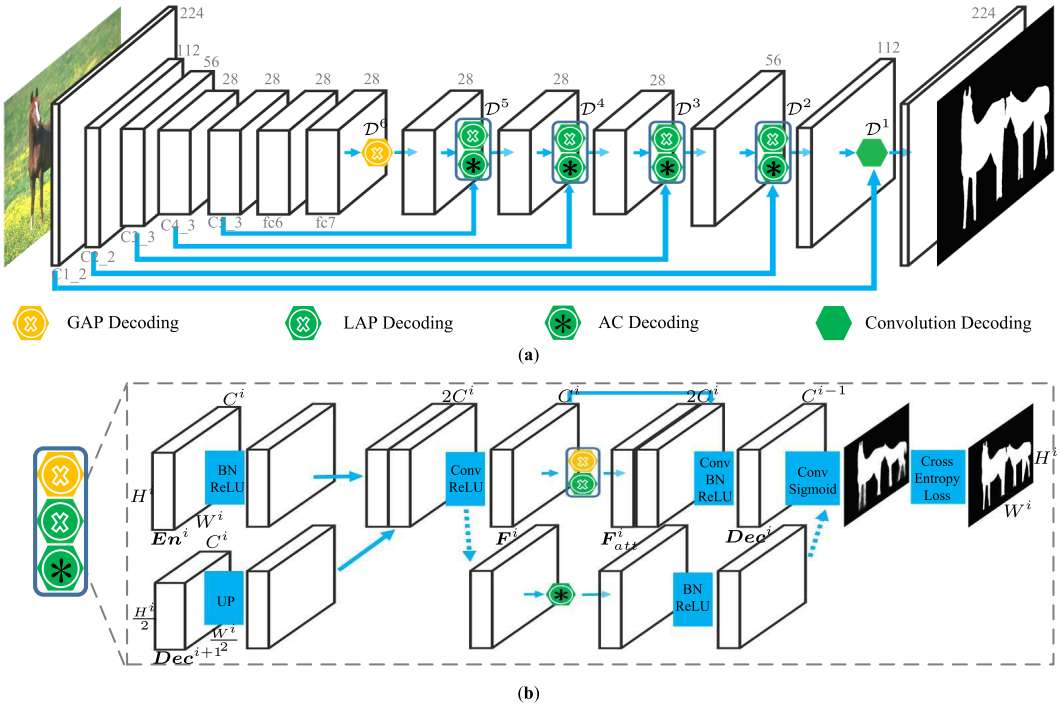


Fig. 3. **Architecture of the proposed saliency network with PiCANets.** (a) Overall architecture of our saliency network. For simplicity, we only show the last layer of each block in the VGG network, i.e., the C*_* layers and fc* layers. We use $\mathcal{D}^i$ to indicate the i^{th} decoding module. The spatial sizes are marked over the cuboids that represent the feature maps. (b) Illustration of an attentive decoding module, either using GAP, LAP, or AC. We use $\boldsymbol{En}^i$ and $\boldsymbol{Dec}^i$ to denote the i^{th} encoding feature map or decoding feature map, respectively. While $\boldsymbol{F}^i$ and $\boldsymbol{F}^i_{att}$ are used to denote the i^{th} fusion feature map and the attentive contextual feature map, respectively. "UP" denotes upsampling. Some crucial spatial sizes and channel numbers are also marked. Since using AC leads to a slightly different network structure compared with using GAP and LAP, we use dashed arrows to denote the different part of the network path.

may lead to more discriminative feature abstraction at the expense of more model parameters.

D. Effective and Efficient Implementation

The pixel-wise attending operation of the proposed PiCANets can be easily carried out in parallel for all pixels via GPU. The dilation convolution algorithm [46] can also be used to sample uniformly-spaced contexts at each pixel. As such, we can efficiently attend to large context regions with significantly reduced computational cost by using a small D or $\bar{D}$ with dilation. Meanwhile, all the three formulations (2)(3)(5) of the PiCANets are fully differentiable, thereby enabling end-to-end training with other Convnet modules via the back-propagation algorithm [47]. When using deep layers to generate the attention weights, batch normalization (BN) [48] can also be used to facilitate gradient propagations for training the models effectively.

E. Differences With Prior Work

The proposed PiCANets differ from the NL-style models [35]–[39] and the PSANet [40] in several aspects. First, the NL-style models generate one attention weight for each query position with respect to one key position by using their pairwise features while the PiCANets use the ReNet or Conv layers to consider all related key positions for each query position and generate their attention weights holistically. Although the PSANet also uses Conv layers to generate attention weights, the Conv kernel size is set to 1 which does not match the sizes of the attending contexts. In contrast, we use the ReNet or specifically designed Conv layers to strictly match those contexts. Second, all these models use attentive pooling or linear summation to aggregate contextual features while the proposed AC module further uses the attention convolution operation to modulate the Conv information flow. Third, as the NL model [35] and the PSANet consider all positions as the attending context, the entailed computational cost and memory requirement are very high. Hence, these models can only be used for small feature maps. To alleviate this problem, other models adopt the criss-cross attention path [36], learn compact global attention bases [37], [38], or attend to sub-regions [39]. In contrast, the PiCANets sample limited context positions by using dilation or adopting local PiCANets, and thus can be applied to feature maps at various scales.

IV. Salient Object Detection With PiCANets

In this section, we use PiCANets hierarchically to detect salient objects. As shown in Figure 3(a), the whole network is based on a U-Net [25] architecture which uses skip-connections in an encoder-decoder architecture to connect intermediate encoder feature maps to decoder modules. Different from [25], we improve the encoder network to construct large and powerful encoder features and also embed the proposed PiCANets in the decoder to select useful contextual regions.

A. Encoder Network

As the GAP module requires the input feature map to have a fixed size, we resize images to a fixed size of 224×224 as the network input. The VGG16 [49] network is used as our encoder to utilize its parameters pre-trained on ImageNet [50]. It originally contains 13 Conv layers, 5 max-pooling layers, and 2 fully connected layers. As shown in Figure 3(a), in order to preserve relative large spatial sizes in higher layers for accurate saliency detection, we reduce the pooling strides of the pool4 and pool5 layers to be 1 and introduce dilation of 2 for the Conv kernels in the Conv5 block. We also follow [46] to transform the last 2 fully connected layers to Conv layers. Specifically, we transform the fc6 layer to a 3×3 Conv layer with 1024 channels and transform the fc7 layer to a 1×1 Conv layer with the same channel number. As such, we can use the pre-trained parameters of these two layers for preserving rich high-level features and turn the whole VGG network into an FCN. The stride of the whole encoder network is reduced to 8, and the spatial size of the final feature map is 28×28.

B. Decoder Network

As shown in Figure 3(a), the decoder network has six decoding modules, named $\mathcal{D}^6, \mathcal{D}^5, \ldots, \mathcal{D}^1$ in sequential order. In $\mathcal{D}^i$, we usually generate a decoding feature map $\boldsymbol{Dec}^i$ by fusing the preceding decoding feature $\boldsymbol{Dec}^{i+1}$ with an intermediate encoder feature map $\boldsymbol{En}^i$. We select $\boldsymbol{En}^i$ as the last Conv feature map before the ReLU activation of the i^{th} Conv block in the encoder part, where its size is denoted as $W^i \times H^i \times C^i$ and all the six selected encoder feature maps are marked in Figure 3(a). An exception is that in $\mathcal{D}^6$, $\boldsymbol{Dec}^6$ is directly generated from $\boldsymbol{En}^6$ without the preceding decoding feature map and $\boldsymbol{En}^6$ comes from the fc7 layer.

The detailed decoding process is shown in Figure 3(b). We first pass $\boldsymbol{En}^i$ through a BN layer and the ReLU activation for normalization and non-linear transformation to get ready for the subsequent fusion. For $\boldsymbol{Dec}^{i+1}$, it usually has a half size of $W^i/2 \times H^i/2$, and thus we upsample it to $W^i \times H^i$ via bilinear interpolation. We concatenate $\boldsymbol{En}^i$ with the upsampled $\boldsymbol{Dec}^{i+1}$ and fuse them into a feature map $\boldsymbol{F}^i$ with C^i channels by using a Conv layer and the ReLU activation. Then we utilize either GAP, LAP, or AC on $\boldsymbol{F}^i$ to obtain the attentive contextual feature map $\boldsymbol{F}^i_{att}$, where we use $\boldsymbol{F}_{att}$ as the general denotation of $\boldsymbol{F}_{GAP}$, $\boldsymbol{F}_{LAP}$, and $\boldsymbol{F}_{AC}$. Since for GAP and LAP, at each pixel $\boldsymbol{F}^i_{att}$ is simply a linear combination of $\boldsymbol{F}^i$, we use it as complementary information for the original feature. Thus we concatenate and fuse $\boldsymbol{F}^i$ and $\boldsymbol{F}^i_{att}$ into $\boldsymbol{Dec}^i$ via a Conv layer with BN and the ReLU activation. We keep the spatial size of $\boldsymbol{Dec}^i$ as $W^i \times H^i$ but set its number of channels to be the same as that of $\boldsymbol{En}^{i-1}$, i.e., C^{i-1}. For AC, as it has already merged the attention and convolution operations, we directly set its number of output channels to be C^{i-1} and generate $\boldsymbol{Dec}^i$ after using BN and the ReLU activation, which is shown as the dashed path in Figure 3(b).

Since the GAP model carries out the attention operation over the entire feature map (which is computationally expensive), we only use it in early decoding modules that have small feature maps but with high-level semantics. We note that adopting GAP in $\mathcal{D}^6$ and using LAP or AC in latter modules leads to the best performance. For computational efficiency, we do not use any PiCANet in $\mathcal{D}^1$, in which case $\boldsymbol{En}^1$ and $\boldsymbol{Dec}^2$ are directly fused into $\boldsymbol{Dec}^1$ by vanilla Conv layers. We present the analysis of the network settings with different usage of PiCANets in Section V-D.

C. Loss Function

To facilitate the network training, we adopt deep supervision for each decoding module. In $\mathcal{D}^i$, we use a Conv layer with one output channel and the sigmoid activation on top of $\boldsymbol{Dec}^i$ to generate a saliency map $\boldsymbol{S}^i$ with size $W^i \times H^i$. The ground truth saliency map is resized to the same size, denoted as $\boldsymbol{G}^i$, for the network training based on the average cross-entropy saliency loss $\boldsymbol{L}^i_S$:

$$\begin{aligned}\boldsymbol{L}^i_S = -\frac{1}{W^i H^i}\sum_{w=1}^{W^i}\sum_{h=1}^{H^i} &\boldsymbol{G}^i(w,h)\log \boldsymbol{S}^i(w,h) \\ &+ (1-\boldsymbol{G}^i(w,h))\log(1-\boldsymbol{S}^i(w,h)),\end{aligned} \tag{6}$$

where $\boldsymbol{G}^i(w,h)$ and $\boldsymbol{S}^i(w,h)$ denote their saliency values at the location (w,h).

In [26], the global attention is found to be able to learn global contrast, i.e., the attention map of foreground pixels mainly highlights background regions and vice verse. However, the learned global attention maps may not be accurate or complete. Thus, we propose to explicitly learn the global attention in the GAP module. We simulate the global contrast mechanism to extract foreground and background regions from the ground truth saliency maps for supervising the learning of the global attention at background and foreground pixels, respectively. Taking $\mathcal{D}^6$ as the example, we first generate the normalized ground truth global attention map $\boldsymbol{A}^{w,h}$ for each pixel (w,h) in $\boldsymbol{F}^6$:

$$\boldsymbol{A}^{w,h} = \begin{cases} \dfrac{\boldsymbol{G}^6}{\sum \boldsymbol{G}^6}, & \text{if } \boldsymbol{G}^6(w,h) = 0, \\ \dfrac{1-\boldsymbol{G}^6}{\sum(1-\boldsymbol{G}^6)}, & \text{if } \boldsymbol{G}^6(w,h) = 1. \end{cases} \tag{7}$$

We then use the averaged KL divergence loss between $\boldsymbol{A}^{w,h}$ and $\boldsymbol{\alpha}^{w,h}$ at each pixel as the global attention loss $\boldsymbol{L}^6_{GA}$:

$$\boldsymbol{L}^6_{GA} = \frac{1}{W^6 H^6}\sum_{w,w'=1}^{W^6}\sum_{h,h'=1}^{H^6} \boldsymbol{A}^{w,h}(w',h')\log\frac{\boldsymbol{A}^{w,h}(w',h')}{\boldsymbol{\alpha}^{w,h}(w',h')}, \tag{8}$$

where $\boldsymbol{\alpha}^{w,h}(w',h') = \alpha^{w,h}_{(h'-1)W^6+w'}$.

The final loss is obtained by a weighted sum of the saliency losses in different decoding modules and the global attention loss:

$$\boldsymbol{L} = \sum_{i=1}^{6} \gamma^i \boldsymbol{L}^i_S + \gamma^{GA}\boldsymbol{L}^6_{GA}. \tag{9}$$

V. Experiments

A. Datasets

We use six widely used saliency benchmark datasets to evaluate our method. The SOD dataset [51] contains 300 images

with complex backgrounds and multiple foreground objects. The ECSSD dataset [52] has 1,000 semantically meaningful and complex images. The PASCAL-S dataset [53] consists of 850 images selected from the PASCAL VOC 2010 segmentation dataset. The DUT-O database [54] includes 5,168 challenging images, each of which usually has a complicated background and one or two foreground objects. The HKU-IS dataset [5] contains 4,447 images with low color contrast and multiple foreground objects in each image. The DUTS dataset [55] is currently the largest salient object detection benchmark dataset. It has 10,553 images in the DUTS-TR training set, and 5,019 images in the DUTS-TE test set.

B. Evaluation Metrics

We use four metrics for performance evaluation. The first one is the precision-recall (PR) curve. A predicted saliency map $\boldsymbol{S}$ is first binarized by a threshold and compared with the corresponding ground truth saliency map $\boldsymbol{G}$. By varying the threshold between 0 and 255, we can obtain a series of precision-recall value pairs to draw the PR curve.

The second metric is the F-measure score which considers both precision and recall:

$$F_\beta = \frac{(1+\beta^2) Precision \times Recall}{\beta^2\ Precision + Recall}, \tag{10}$$

where β^2 is set to 0.3 to emphasize more on precision as commonly used in prior work [10], [12], [17]–[19]. Finally, we report the max F-measure score under the optimal threshold.

The third metric is the Mean Absolute Error (MAE). It is the average absolute per-pixel difference between $\boldsymbol{S}$ and $\boldsymbol{G}$.

All of the three above metrics are based on pixel-wise errors and seldom take structural knowledge into account. Similar to [56], [57], we use the Structure-measure S_m [58] to evaluate both region-aware and object-aware structural similarities and take their average as the S_m score.

C. Implementation Details

1) Network Structure: In the decoding modules, all of the Conv kernels are set to 1×1. In the GAP module, we use 256 hidden neurons for the ReNet, and a 1×1 Conv layer to generate $D = 100$ dimensional attention weights, which can be reshaped to 10×10 attention maps. In the attending operation, we set $dilation$ to 3 for attending to the 28×28 global context. In each LAP or AC module, we first use a 7×7 Conv layer with $dilation$ of 2, zero padding, and the ReLU activation to generate an intermediate feature map with 128 channels. We then use a 1×1 Conv layer to generate $\bar{D} = 49$ dimensional attention weights, from which 7×7 attention maps can be obtained. Thus we can attend to 13×13 local context regions with $dilation$ of 2 and zero padding.

2) Training and Testing: We use the DUTS-TR set as our training set. For data augmentation, we resize each image to 256×256 pixels with random mirror-flipping and then randomly crop 224×224 image regions for training. The whole network is trained end-to-end using stochastic gradient descent (SGD) with momentum. For the weight of each loss term in (9), we empirically set $\gamma^6, \gamma^5, \ldots, \gamma^1$ as 0.5, 0.5, 0.5, 0.8, 0.8, and 1, respectively, without further tuning. In addition, γ^{GA} is set to 0.2 based on the performance validation. We train the decoder part with random initialization and the learning rate of 0.01 and finetune the encoder with a 0.1 times smaller learning rate. We set the batchsize to 9, maximum iteration step to 40,000, and use the "multistep" policy to decay the learning rates by a factor of 0.1 at the $20{,}000th$ and the $30{,}000th$ step. The momentum and weight decay are set to 0.9 and 0.0005, respectively.

We implement our model based on the Caffe [59] library. A GTX 1080 Ti GPU is used for acceleration. When testing, each image is directly resized to 224×224 pixels and fed into the network, then we can obtain its predicted saliency map from the network output without any post-processing. The prediction process takes 0.127 seconds for each image. The source code of this work is available at https://github.com/nianliu/PiCANet.

D. Component Analyses

1) Progressively Embedding PiCANets: To demonstrate the effectiveness of progressively embedding PiCANets in the decoder network, we show quantitative results of different model settings in Table I. We first take the basic U-Net [25] architecture as our baseline model and progressively embed global and local PiCANets into the decoding modules as described in Section IV-B. For the local PiCANets, which include both LAP and AC, we take the latter as the example here. In Table I, "+6GAP" means we only embed a GAP module in $\mathcal{D}^6$, while "+6GAP_5AC" means an AC module is further embedded in $\mathcal{D}^5$ (other settings can be inferred similarly). Quantitative results show that adding GAP in $\mathcal{D}^6$ can moderately improve the model performance, and progressively embedding AC in latter decoding modules helps achieve better results, finally leading to significant performance gain compared with the baseline model.

2) Different Embedding Settings: We present experimental results using different embedding settings of our global and local PiCANets, including only adopting local PiCANets ("+65432AC"), and embedding GAP in more decoding modules ("+65GAP_432AC" and "+654GAP_32AC"). Table I shows that the proposed saliency network with all these three settings generally performs slightly worse than the setting "+6GAP_5432AC". We do not use GAP in other decoding modules since it is time-consuming for large feature maps.

3) AC vs. LAP: As the AC and LAP modules are both local PiCANets, we evaluate the setting of using LAP in $\mathcal{D}^5$ to $\mathcal{D}^2$ ("+6GAP_5432LAP"). Compared with the setting "+6GAP_5432AC", Table I shows that using AC is a slightly better choice for saliency detection.

4) Attention Loss: The global attention loss $\boldsymbol{L}^6_{GA}$ is used to facilitate learning the global contrast in GAP in all previously discussed settings. We evaluate its effectiveness by setting $\gamma^{GA} = 0$ to ban this loss term in training, which is denoted as "+6GAP_5432AC_w/o_$\boldsymbol{L}^6_{GA}$" in Table I. This model performs slightly worse than the setting "+6GAP_5432AC", which indicates that using this loss term is slightly beneficial.

TABLE I

QUANTITATIVE COMPARISON OF DIFFERENT MODEL SETTINGS FOR SALIENCY DETECTION. "*GAP", "*AC", AND "*LAP" MEAN WE EMBED THESE PICANETS IN CORRESPONDING DECODING MODULES. "LC", "MAXP", AND "AVEP" MEAN LARGE-KERNEL CONVOLUTION, MAX-POOLING, AND AVERAGE POOLING, RESPECTIVELY. RED INDICATES THE BEST PERFORMANCE

Dataset	SOD [51]			ECSSD [52]			PASCAL-S [53]			HKU-IS [5]			DUT-O [54]			DUTS-TE [55]		
Metric	F_β	S_m	MAE	F_β	S_m	MAE	F_β	S_m	MAE	F_β	S_m	MAE	F_β	S_m	MAE	F_β	S_m	MAE
Baseline																		
U-Net [25]	0.836	0.753	0.122	0.906	0.886	0.052	0.852	0.809	0.097	0.894	0.877	0.045	0.762	0.794	0.072	0.823	0.834	0.057
Progressively embedding PiCANets																		
+6GAP	0.839	0.759	0.119	0.915	0.896	0.049	0.862	0.818	0.094	0.903	0.887	0.044	0.784	0.810	0.070	0.837	0.845	0.056
+6GAP_5AC	0.847	0.773	0.114	0.921	0.903	0.048	0.868	0.826	0.091	0.910	0.894	0.043	0.786	0.817	0.069	0.843	0.852	0.054
+6GAP_54AC	0.853	0.780	0.110	0.927	0.910	0.045	0.872	0.829	0.090	0.915	0.901	0.041	0.797	0.825	0.067	0.851	0.858	0.054
+6GAP_543AC	0.863	0.789	**0.105**	0.933	0.915	0.045	0.877	0.832	0.089	0.921	0.906	0.040	0.803	0.830	0.068	0.854	0.862	0.054
+6GAP_5432AC	0.858	0.786	0.107	0.935	**0.917**	**0.044**	**0.883**	**0.838**	**0.085**	**0.924**	**0.908**	**0.039**	**0.808**	**0.835**	0.065	**0.859**	**0.867**	0.051
Different embedding settings																		
+65432AC	0.858	0.784	0.110	0.932	0.914	0.045	0.880	0.831	0.087	0.922	0.904	0.040	0.805	0.831	**0.063**	**0.859**	0.865	0.051
+65GAP_432AC	**0.866**	**0.795**	**0.105**	**0.937**	**0.917**	**0.044**	0.876	0.835	0.088	**0.924**	**0.908**	**0.039**	0.805	0.832	0.067	0.858	0.866	0.052
+654GAP_32AC	0.859	0.785	0.107	0.935	0.916	**0.044**	0.877	0.835	0.087	0.922	0.905	0.040	0.802	0.828	0.066	0.855	0.864	0.052
AC vs. LAP																		
+6GAP_5432LAP	**0.866**	0.788	0.106	0.934	0.916	**0.044**	0.880	0.835	0.087	0.923	0.905	0.040	0.799	0.829	0.066	0.857	0.862	0.052
Attention loss																		
+6GAP_5432AC _w/o_$\boldsymbol{L}_{GA}^6$	0.857	0.785	0.109	0.930	0.913	0.045	0.879	0.835	0.086	0.921	0.905	0.040	0.801	0.829	0.066	0.856	0.863	0.053
Comparison with vanilla pooling and Conv layers																		
+6ReNet_5432LC	0.851	0.774	0.114	0.920	0.900	0.049	0.866	0.820	0.093	0.907	0.891	0.043	0.786	0.816	0.071	0.841	0.850	0.056
+6G_5432L_AveP	0.842	0.770	0.114	0.918	0.899	0.048	0.865	0.823	0.092	0.905	0.889	0.043	0.782	0.811	0.071	0.837	0.847	0.056
+6G_5432L_MaxP	0.845	0.771	0.116	0.918	0.899	0.048	0.866	0.819	0.093	0.905	0.889	0.042	0.776	0.808	0.070	0.838	0.848	0.055
Comparison with other attention models																		
+SENet [32]	0.853	0.773	0.111	0.923	0.904	0.050	0.872	0.827	0.091	0.912	0.895	0.044	0.780	0.812	0.068	0.839	0.851	0.054
+ResAtt [33]	0.848	0.769	0.115	0.933	0.911	0.047	0.879	0.831	0.088	0.921	0.902	0.040	0.798	0.825	**0.063**	0.857	0.862	**0.050**
+CBAM [34]	0.860	0.784	0.110	0.926	0.907	0.049	0.874	0.831	0.090	0.914	0.899	0.044	0.787	0.817	0.069	0.840	0.852	0.056
+NL [35]	0.846	0.784	0.117	0.919	0.900	0.056	0.866	0.824	0.097	0.910	0.895	0.047	0.778	0.812	0.076	0.835	0.846	0.062

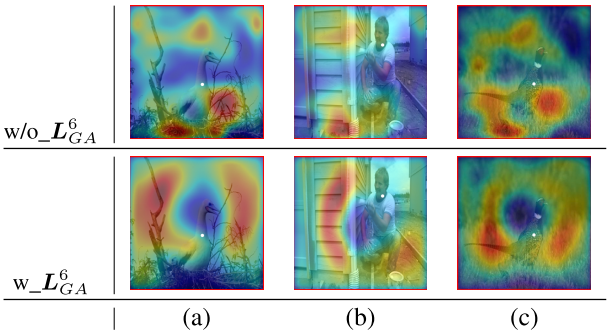


Fig. 4. **Visual comparison on the global attention maps learned with and without using the global attention loss $\boldsymbol{L}_{GA}^6$.** We show the global attention maps learned without (top row) and with (bottom row) using $\boldsymbol{L}_{GA}^6$ at three pixels in three images.

We show some examples of the global attention maps learned with ("w_$\boldsymbol{L}_{GA}^6$") and without ("w/o_$\boldsymbol{L}_{GA}^6$") using the global attention loss $\boldsymbol{L}_{GA}^6$ in Figure 4. The global attention maps learned without using $\boldsymbol{L}_{GA}^6$ (top row) mainly focus on some discrete and key local regions, whereas those learned with using $\boldsymbol{L}_{GA}^6$ (bottom row) can cover most background regions.

We also experiment with similar losses for training local attention. For each pixel, we use local regions that have the same saliency label with itself as the ground truth attention map and adopt the same KL loss. However, we find that this scheme does not perform well. This can be attributed to that regions with the same saliency label do not exactly have similar appearance, especially in cluttered scenes. Thus, the supervision signal is very noisy.

5) Comparison With Vanilla Pooling and Conv Layers: Since in PiCANets we introduce attention weights into pooling and Conv operations to selectively incorporate global and local contexts, we compare them with vanilla pooling and Conv layers which holistically integrate these contexts. We use the ReNet model [44] in $\mathcal{D}^6$ to capture the global context and use large Conv kernels of same size (i.e., 7×7 kernels with $dilation = 2$) in $\mathcal{D}^5$ to $\mathcal{D}^2$ to capture the large local contexts, which is denoted as "+6ReNet_5432LC" in Table I. We also adopt max-pooling (MaxP) and average-pooling (AveP) to incorporate the same-sized contexts, which are denoted as "+6G_5432L_AveP" and "+6G_5432L_MaxP", respectively. In $\mathcal{D}^6$ we first use global pooling and then upsample the pooled feature vector to the same size with $\boldsymbol{F}^6$, while in other decoding modules we employ the same-sized local pooling kernels. Table I shows that using the standard pooling schemes can bring moderate performance gain, but still gets worse results when compared with the "+6GAP_5432AC" and "+6GAP_5432LAP" settings.

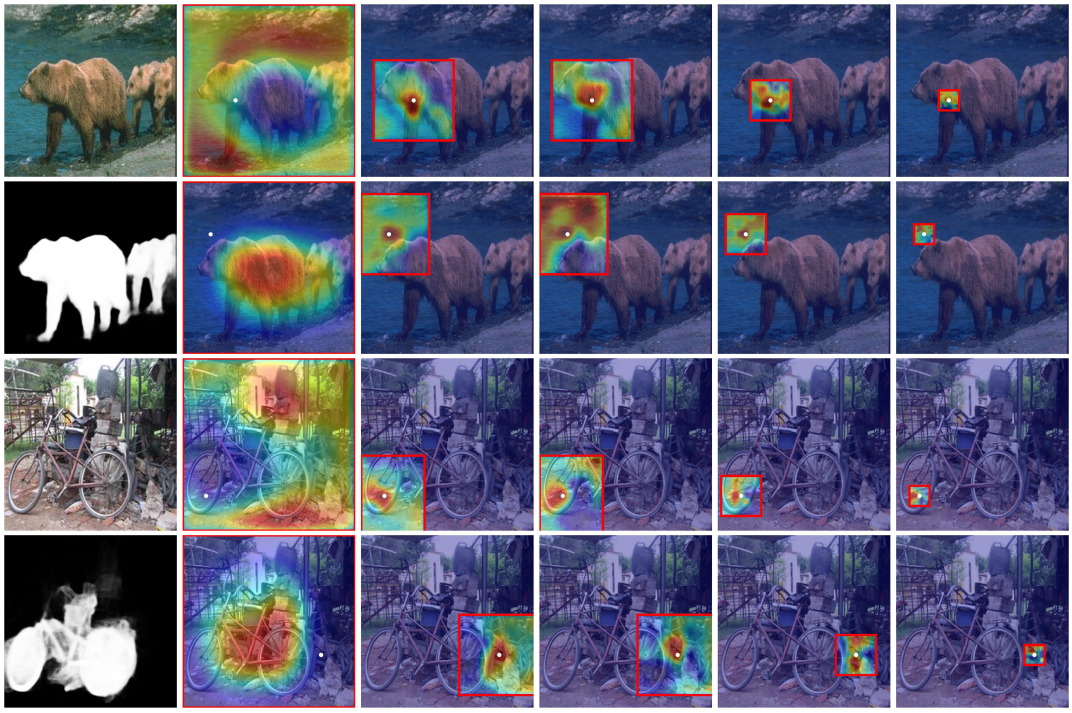


Fig. 5. **Illustration of the generated attention maps of the proposed PiCANets**. The first column shows two images and our saliency maps while the other columns show the attention maps in five attentive decoding modules. For each image, we give two example pixels (denoted as white dots), including a foreground pixel in the first row and a background pixel in the bottom row. The referred context regions are marked by red rectangles.

6) Comparison With Other Attention Models: We also introduce some other attention models in decoder modules for further comparisons, including the SENet model [32], the Residual attention (ResAtt) model [33], the CBAM model [34], and the NL model [35]. Specifically, we embed the first three models in $\mathcal{D}^6$ to $\mathcal{D}^2$, respectively, and use the NL model only in $\mathcal{D}^6$ to $\mathcal{D}^3$ since the spatial size of $\mathcal{D}^2$ is too large to use the NL model due to GPU memory limitation. The experimental results in Table I shows that our "+6GAP_5432AC" model performs favorably against all these four attention models, indicating the superiority of the proposed PiCANets. Among the four compared models, the NL model [35] performs the worst. This may be because it can not be used in $\mathcal{D}^2$ and when generating attention weights, its pair-wise relation can not perceive the whole context region as our PiCANets do.

7) Visual Analyses: We present some visual results to demonstrate the effectiveness of the proposed PiCANets. In Figure 6(a) we show two images and their ground truth saliency maps while (b) shows the predicted saliency maps of the baseline U-Net (the top row in each group) and our model (bottom rows). The proposed saliency model can locate salient objects more accurately and highlight their whole bodies more uniformly with the help of PiCANets. In Figure 6(c), we show comparison of the Conv feature maps $\boldsymbol{F}^6$ (top rows) against the attentive contextual feature maps $\boldsymbol{F}^6_{att}$ (bottom rows) in $\mathcal{D}^6$, and (d) shows $\boldsymbol{F}^2$ (top rows) and $\boldsymbol{F}^2_{att}$ (bottom rows) in $\mathcal{D}^2$. In $\mathcal{D}^6$ the global PiCANet helps better discriminate foreground objects from backgrounds, while the local PiCANet in $\mathcal{D}^2$ enhances the feature maps to be smoother, which helps uniformly segment the foreground objects.

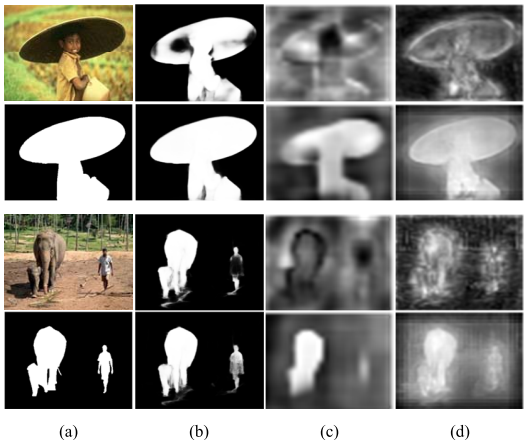


Fig. 6. **Visual comparison of our model against the baseline U-Net**. We show two groups of examples. (a) Two testing images and their ground truth saliency maps. (b) Saliency maps of the baseline U-Net (the top row in each group) and our model (bottom rows). (c) $\boldsymbol{F}^6$ (top rows) and $\boldsymbol{F}^6_{att}$ (bottom rows). (d) $\boldsymbol{F}^2$ (top rows) and $\boldsymbol{F}^2_{att}$ (bottom rows).

To better understand how do PiCANets work, we visualize the generated attention maps of background and foreground pixels in two images in Figure 5. The generated global attention maps are shown in the second column. They indicate that the GAP modules successfully learn global contrast to attend to foreground objects for background pixels and vice verse. Thus the GAP module can help to differentiate salient objects

TABLE II
QUANTITATIVE EVALUATION OF THE STATE-OF-THE-ART SALIENT OBJECT DETECTION MODELS. WE COMPARE THE PROPOSED PICANET++ SALIENCY MODEL WITH OTHER 13 STATE-OF-THE-ART METHODS AND OUR PRELIMINARY MODEL IN TERMS OF THE F-MEASURE SCORE, THE MEAN ABSOLUTE ERROR, AND THE STRUCTURE-MEASURE. RED AND BLUE INDICATE THE BEST AND THE SECOND BEST PERFORMANCE, RESPECTIVELY

Dataset	SOD [51]			ECSSD [52]			PASCAL-S [53]			HKU-IS [5]			DUT-O [54]			DUTS-TE [55]		
Metric	F_β	S_m	MAE	F_β	S_m	MAE	F_β	S_m	MAE	F_β	S_m	MAE	F_β	S_m	MAE	F_β	S_m	MAE
MDF [5]	0.760	0.633	0.192	0.832	0.776	0.105	0.781	0.672	0.165	-	-	-	0.694	0.721	0.092	0.711	0.727	0.114
DCL [9]	0.825	0.745	0.198	0.901	0.868	0.075	0.823	0.783	0.189	0.885	0.861	0.137	0.739	0.764	0.157	0.782	0.795	0.150
RFCN [11]	0.807	0.717	0.166	0.898	0.860	0.095	0.850	0.793	0.132	0.898	0.859	0.080	0.738	0.774	0.095	0.783	0.791	0.090
DHS [10]	0.827	0.747	0.133	0.907	0.884	0.059	0.841	0.788	0.111	0.902	0.881	0.054	-	-	-	0.829	0.836	0.065
Amulet [14]	0.808	0.755	0.145	0.915	0.894	0.059	0.857	0.821	0.103	0.896	0.883	0.052	0.743	0.781	0.098	0.778	0.803	0.085
NLDF [13]	0.842	0.753	0.130	0.905	0.875	0.063	0.845	0.790	0.112	0.902	0.879	0.048	0.753	0.770	0.080	0.812	0.815	0.066
DSS [12]	0.846	0.749	0.126	0.916	0.882	0.053	0.846	0.777	0.112	0.911	0.881	0.040	0.771	0.788	0.066	0.825	0.822	0.057
SRM [15]	0.845	0.739	0.132	0.917	0.895	0.054	0.862	0.816	0.098	0.906	0.887	0.046	0.769	0.798	0.069	0.827	0.835	0.059
RA [22]	0.852	0.761	0.129	0.921	0.893	0.056	0.842	0.772	0.122	0.913	0.887	0.045	0.786	0.814	**0.062**	0.831	0.838	0.060
PAGRN [20]	0.840	0.714	0.151	0.927	0.889	0.061	0.861	0.792	0.111	0.918	0.887	0.048	0.771	0.775	0.071	**0.855**	0.837	0.056
C2S-Net [21]	0.824	0.758	0.128	0.911	0.896	0.053	0.864	0.827	0.092	0.899	0.889	0.046	0.759	0.799	0.072	0.811	0.831	0.062
BMP [18]	**0.856**	0.784	0.112	0.928	0.911	0.045	0.877	0.831	**0.086**	**0.921**	**0.907**	**0.039**	0.774	0.809	0.064	0.851	**0.861**	**0.049**
DGRL [19]	0.849	0.770	0.110	0.925	0.906	**0.043**	0.874	0.826	**0.085**	0.913	0.897	**0.037**	0.779	0.810	**0.063**	0.834	0.845	**0.051**
PiCANet [26]	0.855	**0.787**	**0.108**	**0.931**	**0.914**	0.047	**0.880**	**0.837**	0.088	**0.921**	0.906	0.042	**0.794**	**0.826**	0.068	0.851	**0.861**	0.054
PiCANet++ (ours)	**0.858**	**0.786**	**0.107**	**0.935**	**0.917**	**0.044**	**0.883**	**0.838**	**0.085**	**0.924**	**0.908**	**0.039**	**0.808**	**0.835**	0.065	**0.859**	**0.867**	**0.051**

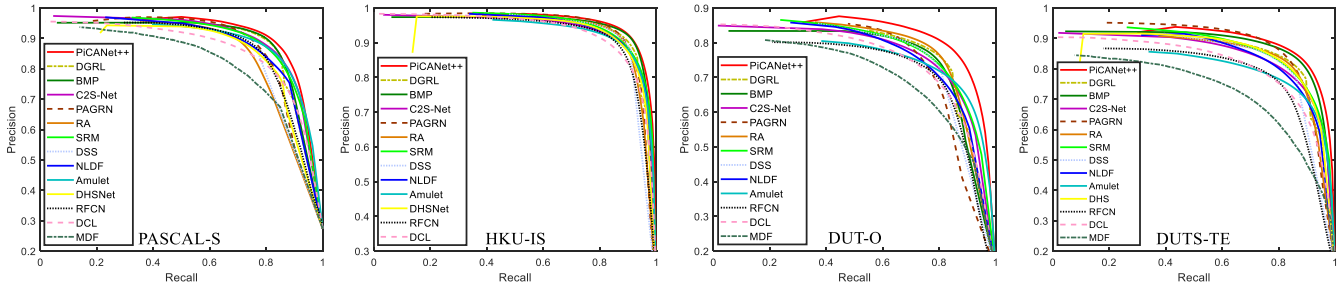


Fig. 7. **Evaluation on four large datasets in terms of the PR curve.** Generally, the proposed PiCANet++ saliency model generates higher PR curves than other state-of-the-art methods, especially on the DUT-O dataset, indicating that it achieves both better precision and recall.

from backgrounds. Regarding local attention, since we use fixed attention size (13×13) for different decoding modules, we can incorporate multiscale attention from large contexts to small ones, as shown by red rectangles in the last four columns of Figure 5. The attention maps show that the local attention mainly attends to the regions with similar appearance to the referred pixel, thereby enhancing the saliency maps to be uniform and smooth, as shown in the first column.

E. Evaluation Against the State-of-the-Art Methods

We use the "+6GAP_5432AC" setting as our saliency model for evaluation against 13 state-of-the-art methods including DGRL [19], BMP [18], C2S-Net [21], PAGRN [20], RA [22], SRM [15], DSS [12], NLDF [13], Amulet [14], DHS [10], RFCN [11], DCL [9], and MDF [5]. For fair comparisons, we either use their released saliency maps to conduct the evaluation or we use their source codes to generate the saliency maps. We also include the preliminary PiCANet saliency model [26] for performance comparison and name the improved saliency model proposed in this work as "PiCANet++".

Table II shows quantitative results in terms of three metrics.[1] The PR curves on four large datasets are also shown in Figure 7. The proposed PiCANet++ saliency model performs favorably against all other models, especially in terms of the F-measure and the Structure-measure metrics despite some other models adopt the conditional random field (CRF) as a post-processing technique or use deeper backbones.

Figure 8 shows qualitative comparisons. The proposed model can handle various challenging scenarios, including images with complex backgrounds and foregrounds (rows 2, 3, 5, and 7), varying object scales, object touching image boundaries (rows 1, 3, and 8), and object having similar appearance with the background (rows 4 and 7). With the proposed PiCANets, our saliency model can localize salient objects more accurately and uniformly than other models in complex visual scenes.

F. Limitations and Future Work

We show some failure cases of the proposed PiCANet++ saliency model in Figure 9. The proposed model mainly

[1]MDF [5] is partly trained on the HKU-IS dataset while DHS [10] is partly trained on the DUT-O dataset.

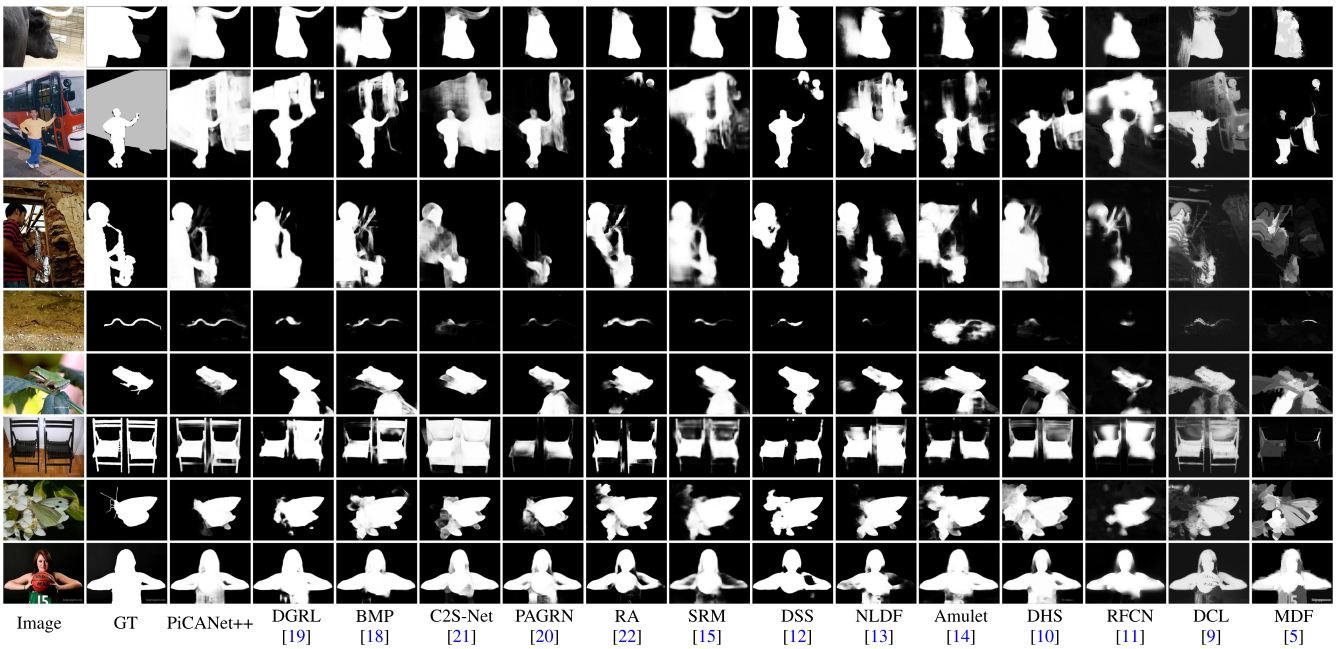


Fig. 8. **Qualitative evaluations.** We compare the saliency detection results of the PiCANet++ saliency model and other 13 methods. (GT: ground truth)

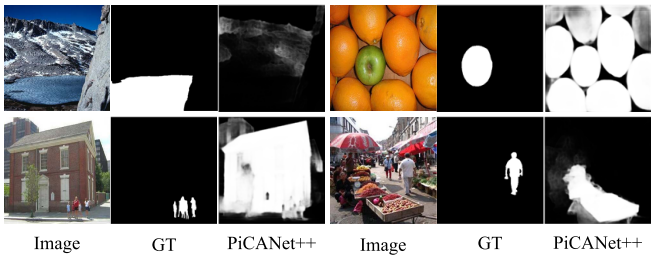


Fig. 9. **Failure cases.** We show four examples of failure cases to demonstrate the model limitations of the proposed PiCANet++ saliency network.

fails in three cases. First, the PiCANet++ model fails to find salient objects when images have no obvious foreground objects (as shown in the upper left image). These images are challenging for almost all existing saliency models. Second, the PiCANet++ model cannot find salient objects that have low-level contrast with the backgrounds (e.g., the top-right case with color contrast). This is because deep CNN networks are not effective for learning low-level features such as color, shape, and texture. Third, similar to the other state-of-the-art methods, our model can be distracted by complex image backgrounds (as shown in the bottom two images). One reason is that the used training set, i.e., the DUTS-TR dataset, mainly contains images with clean backgrounds. Using more challenging training images may alleviate this issue. Another possible solution is to adopt visual reasoning models to compute the degree of saliency of each object and determine the most salient ones.

We plan to improve the proposed PiCANets in several aspects. Since the LAP and AC modules operate in local contexts, we can use them recurrently to propagate attentive contextual information to long-range pixels. On the other hand, we can improve the current PiCANet designs. The ReNet module used in GAP for integrating global contexts can be replaced with other more effective models, such as the criss-cross path used in [36].

G. Application on Other Vision Tasks

We use two dense prediction tasks, i.e., semantic segmentation and object detection, to demonstrate the effectiveness and generalization ability of the proposed PiCANet models.

1) Semantic Segmentation: For semantic segmentation, we first use DeepLab [46] as the baseline model and embed PiCANets into the ASPP module. Since ASPP uses four 3×3 Conv branches with $dilation = \{6, 12, 18, 24\}$, we construct four local PiCANets (i.e., AC or LAP modules) with 7×7 kernels and $dilation = \{2, 4, 6, 8\}$ to incorporate the same sized receptive fields. In each branch, the corresponding local PiCANet is stacked on top of the Pool5 feature map to extract the attentive contextual feature map, which is subsequently concatenated with the Fc6 feature map as the input for the Fc7 layer. We train the model by following the training protocols in [46]. For simplicity, we do not use other strategies proposed in [46], *e.g*, MSC and CRF. For fair comparisons, we also compare PiCANets with vanilla Conv layers with the same large Conv kernels, as denoted by "+LC".

Table III shows the model performance on the PASCAL VOC 2012 val set in terms of mean IOU. Similar to the results on saliency detection, integrating AC and LAP both improve the model performance, while the latter is better for semantic segmentation. Using large Conv kernels here leads to no performance gain, which is probably because their function of holistically incorporating multiscale large receptive fields is heavily overlapped with the ASPP module. The results also demonstrate the effectiveness of the proposed PiCANets.

We also use the U-Net [25] architecture with both global and local PiCANets for semantic segmentation. The network

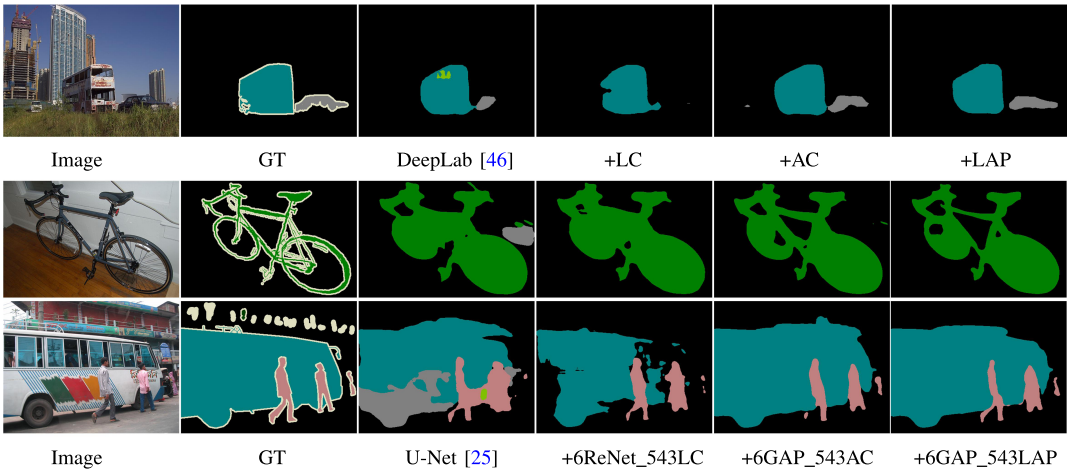


Fig. 10. **Visual comparison of different semantic segmentation model settings.** We show one image for the DeepLab [46] based models and two images for the U-Net [25] based models.

TABLE III
QUANTITATIVE COMPARISON OF DIFFERENT SEMANTIC SEGMENTATION MODEL SETTINGS ON THE PASCAL VOC 2012 VAL SET IN TERMS OF MIOU. RED INDICATES THE BEST PERFORMANCE IN EACH ROW

DeepLab [46]	+LC	+AC	+LAP
68.96	68.90	69.33	**70.12**
U-Net [25]	+6ReNet_543LC	+6GAP_543AC	+6GAP_543LAP
68.60	72.12	72.78	**73.12**

TABLE IV
QUANTITATIVE COMPARISON OF DIFFERENT OBJECT DETECTION MODEL SETTINGS ON THE PASCAL VOC 2007 TEST SET IN TERMS OF MAP. "+478LC_910RENET" MEANS WE USE VANILLA CONV LAYERS WITH LARGE KERNELS FOR THE CONV4_3, FC7, AND CONV8_2 LAYERS AND ADOPT RENET FOR THE CONV9_2 AND CONV10_2 LAYERS. OTHER MODEL SETTINGS CAN BE INFERRED ACCORDINGLY. RED INDICATES THE BEST PERFORMANCE

SSD [60]	+478LC_910ReNet	+478AC_910GAP	+478LAP_910GAP
77.2	77.5	77.9	**78.0**

architecture is similar to the saliency model except that we use 384×384 as the input image size and do not use dilation convolution in the encoder. Furthermore, we set the GAP module in $\mathcal{D}^6$ with the 12×12 kernel size and $dilation = 1$ and only use the first four decoding modules to save GPU memory. There are four settings in the experiments. Table III shows that although adopting ReNet and large Conv kernels can improve the model performance, using PiCANets to select useful context locations can achieve more performance gain.

We present sample segmentation results in Figure 10. Similar to the saliency results, using PiCANets helps obtain more accurate object localization and boundary alignment.

2) Object Detection: For object detection, we embed the PiCANets into the SSD [60] network for experiments. SSD uses the VGG [49] 16-layer network as the backbone and conducts bounding box regression and object classification from six Conv feature maps, i.e., Conv4_3, FC7, Conv8_2, Conv9_2, Conv10_2, and Conv11_2. We use local PiCANets with the 7×7 kernel size and $dilation = 2$ for the first three feature maps and adopt GAP for the latter two according to their gradually reduced spatial sizes. The network structure of Conv11_2 is kept unchanged since its spatial size is 1. Considering the network architecture and the spatial size of each layer, we make the following network designs:

- For Conv4_3 and FC7, we directly stack an AC module on each of them. Or we can also use the LAP modules, where we concatenate the obtained attentive contextual feature maps with themselves as the inputs for the multi-box head.
- For Conv8_2, when using LAP, we stack a LAP on top of Conv8_1 and concatenate the obtained attentive contextual feature with it as the input for the Conv8_2 layer. When using AC, we directly replace the vanilla Conv layer of Conv8_2 with an AC module.
- For Conv10_2 and Conv11_2, we deploy a GAP module on each of the Conv10_1 and Conv11_1 layers, where the kernel size is set to be equal to the feature map size. Then the obtained attentive contextual features are concatenated with them as the inputs for Conv10_2 and Conv11_2.

We also experiment with a model setting to use the ReNet and vanilla Conv layers with large kernels to substitute the GAP and AC modules for a fair comparison.

Similar to the SSD300 model, we use 300×300 as the input image size and evaluate on the PASCAL VOC 2007 test set with the mAP metric. The quantitative comparison results are reported in Table IV. It shows that using PiCANets with attention can bring more performance gain than the conventional way to holistically incorporate global and local contexts, which is consistent with the previous conclusions for saliency detection and semantic segmentation. Figure 11 shows sample detection results. The proposed PiCANets can either

SSD [60] +478LC_910ReNet +478AC_910GAP +478LAP_910GAP

Fig. 11. **Visual comparison of different object detection model settings.** Please notice the increased detection boxes and zoom in for noticing the improvement on the confidence scores brought by using PiCANets.

help generate more accurate bounding boxes, or improve the confidence scores, or detect missing objects.

VI. Conclusion

In this paper, we propose novel PiCANets to adaptively attend to useful contexts for each image pixel. We formulate the proposed PiCANets into three forms based on the pooling and convolution operations over global or local contexts. These modules are fully differentiable and can be plugged into Convnets. We apply PiCANets to a U-Net based architecture in a hierarchical fashion to detect salient objects. With the help of the attended contexts, our model performs favorably against other state-of-the-art methods. We also present in-depth analyses and show that the global PiCANet helps to learn global contrast while local PiCANets learn smoothness. Furthermore, we validate PiCANets on semantic segmentation and object detection to further show their effectiveness and generalization ability to other vision tasks.

References

[1] L. Itti, C. Koch, and E. Niebur, "A model of saliency-based visual attention for rapid scene analysis," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 20, no. 11, pp. 1254–1259, Nov. 1998.

[2] B. Han, H. Zhu, and Y. Ding, "Bottom-up saliency based on weighted sparse coding residual," in *Proc. 19th ACM Int. Conf. Multimedia MM*, 2011, pp. 1117–1120.

[3] M.-M. Cheng, N. J. Mitra, X. Huang, P. H. Torr, and S.-M. Hu, "Global contrast based salient region detection," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 37, no. 3, pp. 569–582, Aug. 2014.

[4] D. A. Klein and S. Frintrop, "Center-surround divergence of feature statistics for salient object detection," in *Proc. Int. Conf. Comput. Vis.*, Nov. 2011, pp. 2214–2219.

[5] G. Li and Y. Yu, "Visual saliency based on multiscale deep features," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2015, pp. 5455–5463.

[6] N. Liu, J. Han, T. Liu, and X. Li, "Learning to predict eye fixations via multiresolution convolutional neural networks," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 29, no. 2, pp. 392–404, Feb. 2018.

[7] R. Zhao, W. Ouyang, H. Li, and X. Wang, "Saliency detection by multi-context deep learning," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2015, pp. 1265–1274.

[8] J. Kuen, Z. Wang, and G. Wang, "Recurrent attentional networks for saliency detection," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 3668–3677.

[9] G. Li and Y. Yu, "Deep contrast learning for salient object detection," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 478–487.

[10] N. Liu and J. Han, "DHSNet: Deep hierarchical saliency network for salient object detection," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 678–686.

[11] L. Wang, L. Wang, H. Lu, P. Zhang, and X. Ruan, "Saliency detection with recurrent fully convolutional networks," in *Proc. Eur. Conf. Comput. Vis.*, 2016, pp. 825–841.

[12] Q. Hou, M.-M. Cheng, X. Hu, A. Borji, Z. Tu, and P. Torr, "Deeply supervised salient object detection with short connections," in *IEEE Conf. Comput. Vis. Pattern Recognit.*, 2017, pp. 5300–5309.

[13] Z. Luo, A. Mishra, A. Achkar, J. Eichel, S. Li, and P.-M. Jodoin, "Non-local deep features for salient object detection," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 6593–6601.

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

[15] T. Wang, A. Borji, L. Zhang, P. Zhang, and H. Lu, "A stagewise refinement model for detecting salient objects in images," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Oct. 2017, pp. 4019–4028.

[16] N. Liu and J. Han, "A deep spatial contextual long-term recurrent convolutional network for saliency detection," *IEEE Trans. Image Process.*, vol. 27, no. 7, pp. 3264–3274, Jul. 2018.

[17] W. Wang, J. Shen, X. Dong, and A. Borji, "Salient object detection driven by fixation prediction," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 1711–1720.

[18] L. Zhang, J. Dai, H. Lu, Y. He, and G. Wang, "A bi-directional message passing model for salient object detection," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 1741–1750.

[19] T. Wang *et al.*, "Detect globally, refine locally: A novel approach to saliency detection," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 3127–3135.

[20] X. Zhang, T. Wang, J. Qi, H. Lu, and G. Wang, "Progressive attention guided recurrent network for salient object detection," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 714–722.

[21] X. Li, F. Yang, H. Cheng, W. Liu, and D. Shen, "Contour knowledge transfer for salient object detection," in *Proc. Eur. Conf. Comput. Vis.*, 2018, pp. 355–370.

[22] S. Chen, X. Tan, B. Wang, and X. Hu, "Reverse attention for salient object detection," in *Proc. Eur. Conf. Comput. Vis.*, 2018, pp. 234–250.

[23] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2015, pp. 3431–3440.

[24] D. Bahdanau, K. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate," in *Proc. Int. Conf. Learn. Representations*, 2015, pp. 1–15.

[25] O. Ronneberger, P. Fischer, and T. Brox, "U-Net: Convolutional networks for biomedical image segmentation," in *Proc. Int. Conf. Med. Image Comput. Comput.-Assist. Intervent*, 2015, pp. 234–241.

[26] N. Liu, J. Han, and M.-H. Yang, "PiCANet: Learning pixel-wise contextual attention for saliency detection," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 3089–3098.

[27] K. Xu *et al.*, "Show, attend and tell: Neural image caption generation with visual attention," in *Proc. Int. Conf. Mach. Learn.*, Jun. 2015, pp. 2048–2057.

[28] P. Sermanet, A. Frome, and E. Real, "Attention for fine-grained categorization," 2014, *arXiv:1412.7054*. [Online]. Available: http://arxiv.org/abs/1412.7054

[29] H. Xu and K. Saenko, "Ask, attend and answer: Exploring question-guided spatial attention for visual question answering," in *Proc. Eur. Conf. Comput. Vis.*, 2016, pp. 451–466.

[30] Z. Yang, X. He, J. Gao, L. Deng, and A. Smola, "Stacked attention networks for image question answering," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 21–29.

[31] J. Li *et al.*, "Attentive contexts for object detection," *IEEE Trans. Multimedia*, vol. 19, no. 5, pp. 944–954, May 2017.

[32] J. Hu, L. Shen, and G. Sun, "Squeeze-and-excitation networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 7132–7141.

[33] F. Wang *et al.*, "Residual attention network for image classification," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 3156–3164.

[34] S. Woo, J. Park, J. Lee, and I. S. Kweon, "CBAM: Convolutional block attention module," in *Proc. Eur. Conf. Comput. Vis.*, Sep. 2018, pp. 3–19.

[35] X. Wang, R. Girshick, A. Gupta, and K. He, "Non-local neural networks," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 7794–7803.

[36] Z. Huang, X. Wang, L. Huang, C. Huang, Y. Wei, and W. Liu, "CCNet: Criss-cross attention for semantic segmentation," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Oct. 2019, pp. 603–612.

[37] Y. Chen, Y. Kalantidis, J. Li, S. Yan, and J. Feng, "Aˆ2-Nets: Double attention networks," in *Proc. Neural Inf. Process. Syst.*, 2018, pp. 352–361.

[38] X. Li, Z. Zhong, J. Wu, Y. Yang, Z. Lin, and H. Liu, "Expectation-maximization attention networks for semantic segmentation," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Oct. 2019, pp. 9167–9176.

[39] J. He, Z. Deng, L. Zhou, Y. Wang, and Y. Qiao, "Adaptive pyramid context network for semantic segmentation," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2019, pp. 7519–7528.

[40] H. Zhao *et al.*, "Psanet: Point-wise spatial attention network for scene parsing," in *Proc. Eur. Conf. Comput. Vis.*, 2018, pp. 267–283.

[41] N. Liu, J. Han, D. Zhang, S. Wen, and T. Liu, "Predicting eye fixations using convolutional neural networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2015, pp. 362–370.

[42] M. Liang and X. Hu, "Recurrent convolutional neural network for object recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2015, pp. 3367–3375.

[43] S. Xingjian, Z. Chen, H. Wang, D.-Y. Yeung, W.-K. Wong, and W.-C. Woo, "Convolutional lstm network: A machine learning approach for precipitation nowcasting," in *Proc. Neural Inf. Process. Syst.*, 2015, pp. 802–810.

[44] F. Visin, K. Kastner, K. Cho, M. Matteucci, A. Courville, and Y. Bengio, "ReNet: A recurrent neural network based alternative to convolutional networks," 2015, *arXiv:1505.00393*. [Online]. Available: http://arxiv.org/abs/1505.00393

[45] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Comput.*, vol. 9, no. 8, pp. 1735–1780, 1997.

[46] L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille, "DeepLab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected CRFs," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 40, no. 4, pp. 834–848, Apr. 2018.

[47] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors," *Nature*, vol. 323, no. 6088, pp. 533–536, Oct. 1986.

[48] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *Proc. Int. Conf. Mach. Learn.*, 2015, pp. 448–456.

[49] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," 2014, *arXiv:1409.1556*. [Online]. Available: http://arxiv.org/abs/1409.1556

[50] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "ImageNet: A large-scale hierarchical image database," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2009, pp. 248–255.

[51] V. Movahedi and J. H. Elder, "Design and perceptual validation of performance measures for salient object segmentation," in *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit. Workshops*, Jun. 2010, pp. 49–56.

[52] Q. Yan, L. Xu, J. Shi, and J. Jia, "Hierarchical saliency detection," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2013, pp. 1155–1162.

[53] Y. Li, X. Hou, C. Koch, J. M. Rehg, and A. L. Yuille, "The secrets of salient object segmentation," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2014, pp. 280–287.

[54] C. Yang, L. Zhang, H. Lu, X. Ruan, and M.-H. Yang, "Saliency detection via graph-based manifold ranking," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2013, pp. 3166–3173.

[55] L. Wang *et al.*, "Learning to detect salient objects with image-level supervision," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 136–145.

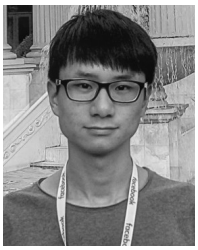
[56] D.-P. Fan, M.-M. Cheng, J.-J. Liu, S.-H. Gao, Q. Hou, and A. Borji, "Salient objects in clutter: Bringing salient object detection to the foreground," in *Proc. Eur. Conf. Comput. Vis. (ECCV)*, New York, NY, USA, 2018, pp. 196–212.

[57] K.-J. Hsu, C.-C. Tsai, Y.-Y. Lin, X. Qian, and Y.-Y. Chuang, "Unsupervised CNN-based co-saliency detection with graphical optimization," in *Proc. Eur. Conf. Comput. Vis.*, Sep. 2018, pp. 502–518.

[58] D.-P. Fan, M.-M. Cheng, Y. Liu, T. Li, and A. Borji, "Structure-measure: A new way to evaluate foreground maps," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Oct. 2017, pp. 4558–4567.

[59] Y. Jia *et al.*, "Caffe: Convolutional architecture for fast feature embedding," in *Proc. ACM Int. Conf. Multimedia - MM*, 2014, pp. 675–678.

[60] W. Liu *et al.*, "SSD: Single shot multibox detector," in *Proc. Eur. Conf. Comput. Vis.*, 2016, pp. 21–37.



Nian Liu (Member, IEEE) received the B.S., M.S., and Ph.D. degrees from the School of Automation, Northwestern Polytechnical University, in 2020, 2015, and 2012, respectively. He is currently a Postdoctoral Researcher with the Mohamed Bin Zayed University of Artificial Intelligence, United Arab Emirates. His research interests include computer vision and machine learning, especially on saliency detection and deep learning.

Junwei Han (Senior Member, IEEE) is currently a Full Professor with Northwestern Polytechnical University, Xi'an, China. His research interests include computer vision, multimedia processing, and brain imaging analysis. He is an Associate Editor of the IEEE TRANSACTIONS ON HUMAN-MACHINE SYSTEMS, *Neurocomputing*, *Multidimensional Systems and Signal Processing*, and *Machine Vision and Applications*.

Ming-Hsuan Yang (Fellow, IEEE) received the Ph.D. degree in computer science from the University of Illinois at Urbana–Champaign in 2000. In 2008, he was a Senior Research Scientist at the Honda Research Institute working on vision problems related to humanoid robots. He is a Professor in electrical engineering and computer science at the University of California at Merced, Merced, CA, USA. He received the NSF CAREER Award in 2012, the Senate Award for Distinguished Early Career Research at UC Merced in 2011, and the Google Faculty Award in 2009. He served as an Associate Editor for the IEEE TRANSACTIONS ON PATTERN ANALYSIS AND MACHINE INTELLIGENCE from 2007 to 2011. He is an Associate Editor of the *International Journal of Computer Vision*, *Image and Vision Computing*, and the *Journal of Artificial Intelligence Research*.