Supplementary Material: Zoom In and Out: A Mixed-scale Triplet Network for Camouflaged Object Detection

Youwei Pang[1†], Xiaoqi Zhao[1†], Tian-Zhu Xiang[3], Lihe Zhang[1*] and Huchuan Lu[1,2]
[1]Dalian University of Technology, China [2]Peng Cheng Laboratory, China
[3]Inception Institute of Artificial Intelligence, UAE
{lartpang, zxq}@mail.dlut.edu.cn, tianzhu.xiang19@gmail.com, {zhanglihe, lhchuan}@dlut.edu.cn

Abstract

This appendix will introduce more details that cannot be expanded in the main text, while showing the performance on SOD. The main contents are summarized as follows:

1. *Model Details, Sec. 1*
   (a) *E-Net, Sec. 1.1*
   (b) *C-Net, Sec. 1.2*
   (c) *Decoder Framework, Sec. 1.3*
   (d) *Baseline Model, Sec. 1.4*
   (e) *Model ⑤, Sec. 1.5*
2. *HMU: Perspective of Kernel Pyramid, Sec. 2*
3. *More Comparisons, Sec. 3*
   (a) *PR & F_β curves of COD Methods, Sec. 3.1*
   (b) *Comparisons of Param. & FLOPs, Sec. 3.2*
   (c) *Intermidiate Feature Maps of the Decoder, Sec. 3.3*
   (d) *Effectiveness of UAL, Sec. 3.4*
   (e) *Different Forms of λ, Sec. 3.5*
   (f) *Different Forms of UAL, Sec. 3.6*
   (g) *Performance in More Complex Scenes, Sec. 3.7*
4. *Experiments on SOD, Sec. 4*
   (a) *Datasets, Sec. 4.1*
   (b) *Implementation Details, Sec. 4.2*
   (c) *Comparisons with State-of-the-arts, Sec. 4.3*
5. *Limitations and Future Work, Sec. 5*

1. Model Details

1.1. E-Net

E-Net is based on the feature extraction part of ResNet-50 [5] and the layers after the "layer4" are removed. We collect the feature maps before passing the first max-pooling layer and the output feature maps of "layer1", "layer2", "layer3" and "layer4" as the output feature maps of the E-Net. The numbers of channels corresponding to them are 64, 256, 512, 1024, and 2048, respectively.

1.2. C-Net

Following the setting of the method [37], in C-Net, we use an ASPP [1] simplified according to our needs as the feature compression layer corresponding to the "layer4" of E-Net and other layers are simply composed of an independent "Conv3×3-BN-ReLU" (3×3 CBR) unit. The numbers of output channels of all levels are set to 64 in our models.

The ASPP layer is composed of five CBR branches. The kernel sizes and dilation rates of them are $1, 3, 3, 3, 1$ and $1, 2, 5, 7, 1$. All convolution operations use the padding to ensure that the input and output sizes are consistent. A global average pooling operation and an up-sampling operation are used before and after the second 1×1 CBR branch to capture the global context information and restore it to the original size. All results of the five branches are concatenated along the channel dimension and fused by a 3×3 CBR unit to obtain the output.

1.3. Decoder Framework

The decoder networks of our models in all experiments follow the same framework as shown in Fig. 2. Before being fed into the fusion unit (FU), the up-sampled deeper feature map is directly added to the shallow feature map.

In our all experiments, N_f and N_l are set to 1. The numbers of input & output channels of the last 3×3 CBR unit are 64 and 32, respectively. The number of output channels of the "Conv1 $\times$ 1" is 1 and a sigmoid layer is cascaded to convert the logits map to the prediction. In the decoder of the proposed ZoomNet, the FU is set to the HMU and the other layers remain the same.

†These authors contributed equally to this work.
*Corresponding author.

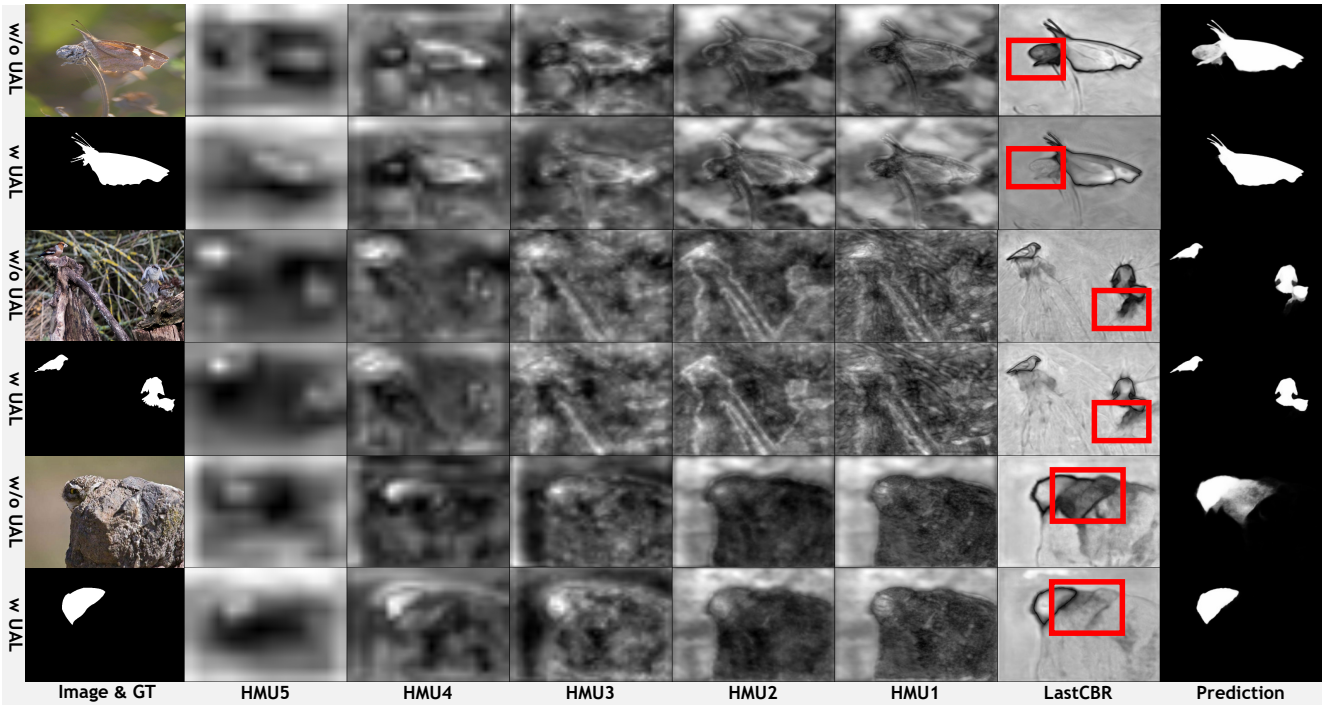


Figure 1. Visual comparison of intermediate feature maps from different stages of the decoder for showing the effects of the proposed UAL. Please zoom in for more details. HMU: Hierarchical mixed-scale unit; LastCBR: The last "Conv3 $\times$ 3-BN-ReLU" structure before the layer generating the logits map.

```python
class StackedCBRBlock(nn.Sequential):
    def __init__(self, in_c, out_c, num_blocks=1, kernel_size=3):
        super().__init__()
        self.kernel_setting = dict(kernel_size=kernel_size, stride=1, padding=kernel_size // 2)
        cs = [in_c] + [out_c] * num_blocks
        self.channel_pairs = tuple(self.slide_win_select(cs, win_size=2, win_stride=1, drop_last=True))
        for i, (i_c, o_c) in enumerate(self.channel_pairs):
            self.add_module(name=f"cbr_{i}", module=CBR(i_c, o_c, **self.kernel_setting))
    @staticmethod
    def slide_win_select(items, win_size=1, win_stride=1, drop_last=False):
        i = 0
        while i + win_size <= len(items):
            yield items[i: i + win_size]
            i += win_stride
        if not drop_last:
            yield items[i: i + win_size]
```

Listing 1. Code of stacked CBR units.

1.4. Baseline Model

In the ablation study, we introduce a simple encoder-decoder network as our baseline model to evaluate the performance of different proposed components. It contains a feature extraction network "E-Net", a simple multi-level feature compression convolutional network "C-Net", and a basic convolutional decoder where the FU is set to the 3×3 CBR unit. In the following text, "CBR1-5" are used to refer to these five units.

1.5. Model ⑤

In Tab. 2 of the main text, based on the baseline model ①, we construct the model ⑤ with the similar amount of parameters and FLOPs to ④ to reflect the effectiveness of the method and the rationality of the design. For increasing the number of parameters and FLOPs, we made the following modifications to the baseline model ①:

- The number of output channels of all levels of C-Net: $64 \rightarrow 128$.

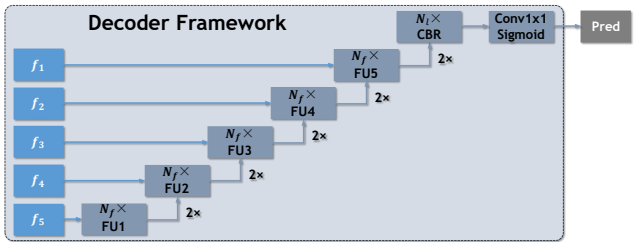


Figure 2. Illustration of the basic framework adopted by the decoder in our proposed method. FU: The fusion unit for fusing the up-sampled feature map from the previous FU and the shallower feature map f_i ($i = 1, 2, ..., 5$). 2×: The bi-linear interpolation operation with a factor of 2. CBR: The "Conv3 × 3-BN-ReLU" unit. Conv1 × 1: The convolution operation with a kernel size of 1 × 1. N_f and N_l: The numbers of FUs and the last CBR units.

- The number of input/output channels of CBR1-5 units of the basic convolutional decoder: $64 \rightarrow 128$.
- The number of input channels of the last CBR unit: $64 \rightarrow 128$.
- The number of CBR units (N_f and N_l) of all levels of the basic convolutional decoder: $1 \rightarrow 3$.
- The kernel size of the convolution operation in all levels of the basic convolutional decoder: $3 \rightarrow 5$.

To facilitate understanding, the corresponding code for the stacked CBR units used here is listed in List. 1.

Algorithm 1 The iteration struction in the HMU

Input: $\{g_j\}_{j=1}^{G}$: feature groups; $G \geq 2$: the number of groups; $C = 32$: the number of channels in a single feature group g_j; $\mathcal{S}$: splitting operation; $\mathcal{T}_{C_o \times C_i}$: stacked CBR units with initial input and final output channel numbers of C_i and C_o as listed in List. 1; $\mathcal{C}$: concatenation operation;

Output: $\{g'^2_j\}_{j=1}^{G}$: the feature set for generating the modulation vector α; $\{g'^3_j\}_{j=1}^{G}$: the feature set used to be modulated and generate the final output of the HMU;

for $i \leftarrow 1, G$ **do**
  if $i = 1$ **then** ▷ Group 1
    $g'^1_i, g'^2_i, g'^3_i \leftarrow \mathcal{S}(\mathcal{T}^i_{3C \times C}(g_i))$;
    $g'^1_{prev} \leftarrow g'^1_i$;
  else if $i = G$ **then** ▷ Group G
    $g'^2_i, g'^3_i \leftarrow \mathcal{S}(\mathcal{T}^i_{2C \times 2C}(\mathcal{C}(g_i, g'^1_{prev})))$;
  else ▷ Group i, $1 < i < G$
    $g'^1_i, g'^2_i, g'^3_i \leftarrow \mathcal{S}(\mathcal{T}^i_{3C \times 2C}(\mathcal{C}(g_i, g'^1_{prev})))$;
    $g'^1_{prev} \leftarrow g'^1_i$;
  end if
end for

Table 1. Comparisons of the number of parameters and FLOPs based on https://github.com/lartpang/MethodsCmp corresponding to recent COD methods. All evaluations follow the inference settings in the corresponding papers.

Method	Ours	UGTR [30]	C^2F-Net [19]	UJSC [6]	PFNet [14]	MGL-R [33]	SLSR [13]	SINet [4]
Params.	32.382M	48.868M	28.411M	217.982M	46.498M	63.595M	50.935M	48.947M
FLOPs	203.496G	1.007T	26.167G	112.341G	53.222G	553.939G	66.625G	38.757
FPS	24.030	16.640	65.759	34.178	62.590	13.373	58.782	56.509

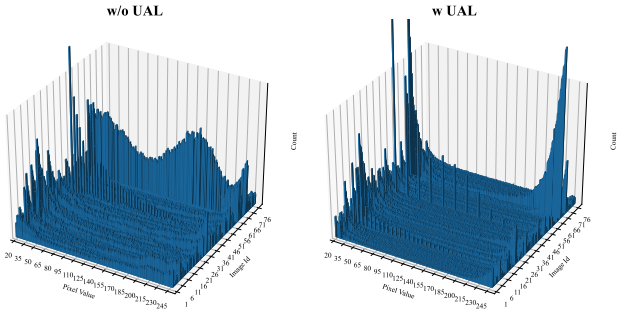


Figure 3. Visual comparison of histograms of all 76 prediction results on the CHAMELEON [18] dataset, which is a stack of the histogram of each prediction. A good result should embody a closely binarized histogram at both ends. For a more clear demonstration, only the interval with pixel values between 20 and 245 is counted here. It is best to zoom in for more details.

Table 2. Comparisons of different increasing strategies of λ. λ_{const}: A constant value and it is set to 1. t and T: The current and total number of iterations, respectively. λ_{min} and λ_{max}: The minimum and maximum values of λ, and they are set to 0 and 1 in our experiments. "Linear$_{t_{min} \rightarrow t_{max}}$": The linearly increasing interval in the iterations is $[t_{min}, t_{max}]$. clip: Values outside the interval are clipped to the interval edges.

Strategy	λ	$S_m \uparrow$	$F^w_\beta \uparrow$	MAE $\downarrow$	$F_\beta \uparrow$	$E_m \uparrow$
Cosine	$\lambda_{min} + \frac{1}{2}(1 - \cos(\frac{t}{T}\pi))(\lambda_{max} - \lambda_{min})$	0.838	0.729	0.029	0.766	0.911
Linear$_{0 \rightarrow T}$	clip$(\lambda_{min} + \frac{t - t_{min}}{t_{max} - t_{min}}(\lambda_{max} - \lambda_{min}), \lambda_{min}, \lambda_{max})$	0.834	0.723	0.029	0.760	0.908
Linear$_{0.3T \rightarrow 0.7T}$		0.832	0.719	0.030	0.758	0.904
Constant	λ_{const}	0.830	0.717	0.030	0.757	0.906

2. HMU: Perspective of Kernel Pyramid

The iteration structure of feature groups in HMU is actually equivalent to an integrated multi-path kernel pyramid structure with partial parameter sharing. In order to understand this intuitively, we highlight the feature information flow of different groups in the iterative structure in Fig. 5. Specifically, the 3 × 3 CBR unit corresponding to the feature group in the iteration structure can be split according to the output feature groups. As shown in the "Integrated Kernel Pyramid" on the left of Fig. 5, each original CBR unit with an output channel number of $3C$ is converted to three independent CBR units with a shared input. And the numbers of output channels of them are C. When we further decouple the integrated form on the left into the form on the right, we can clearly see that the information flow

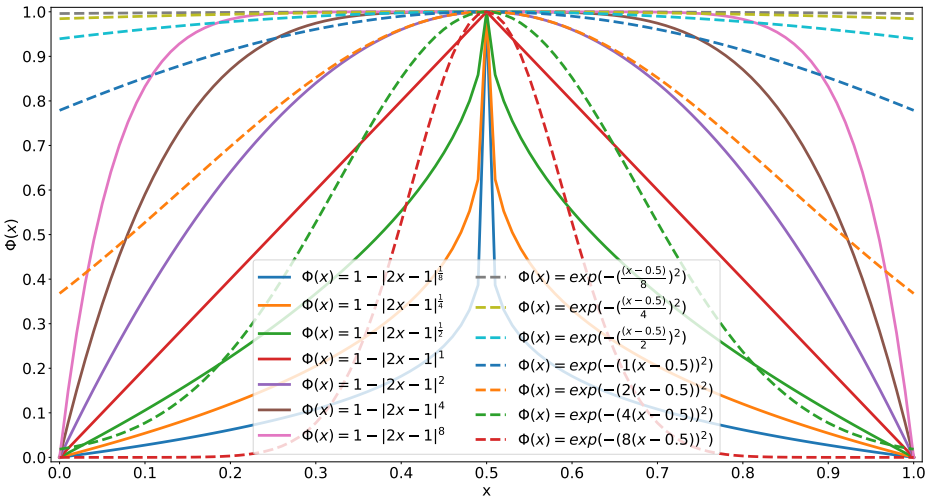


Figure 4. Curves of different forms of the proposed UAL.

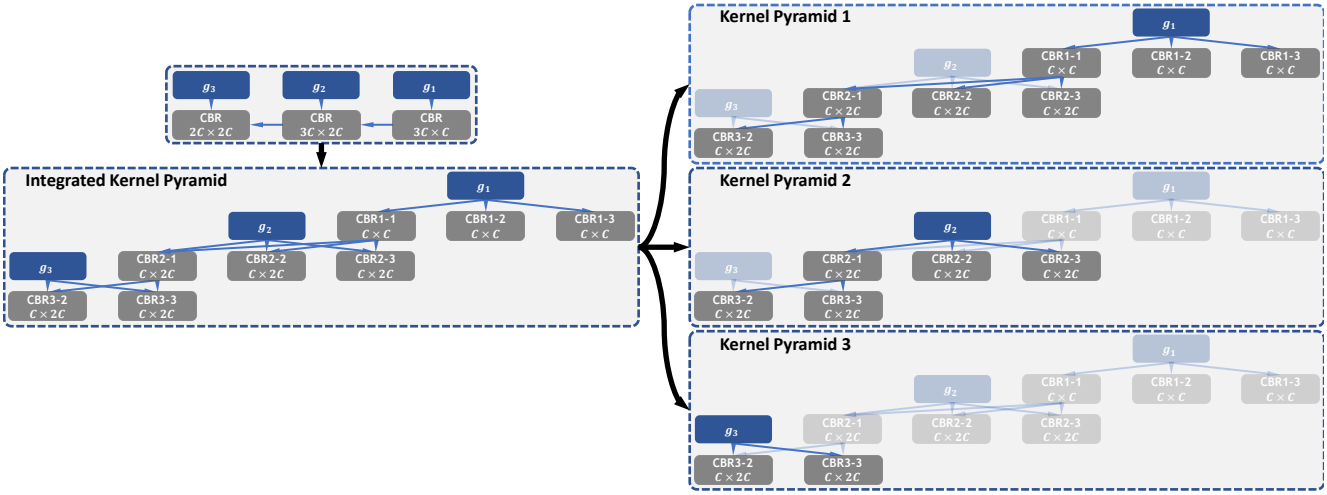


Figure 5. The iteration structure of feature groups in HMU can be regarded as an integrated kernel pyramid. Without loss of generality, we show the situation of the number of groups $G = 3$ in the figure. The actual final model is set to $G = 6$. The only difference lies in the number of repetitions of the kernel pyramid structure in the middle. "CBRl-j": The "Conv3 $\times$ 3-BN-ReLU" structure corresponding to the input feature group g_l and the jth output feature group. $C_o \times C_i$: The numbers of input and output channels of the CBR unit is C_i and C_o, respectively.

paths corresponding to different feature groups each form a multi-branch kernel pyramid structure and there are some shared parameters between these pyramids.

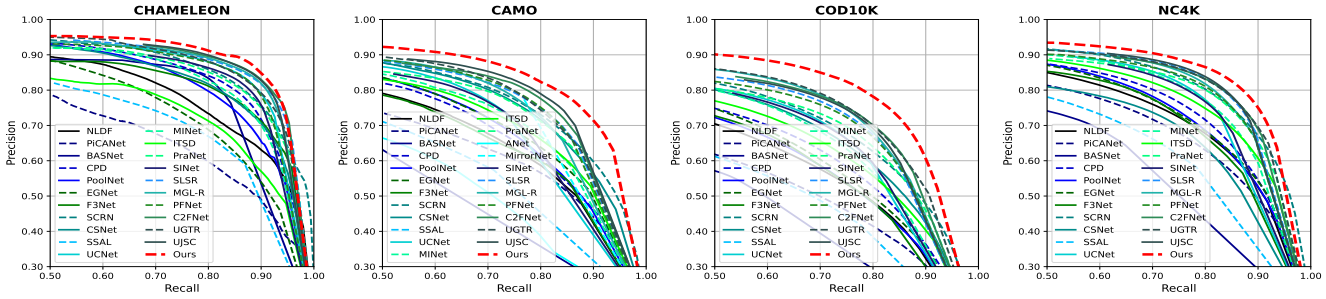
As mentioned in the main text of the paper, some of the channels in the output feature of each branch are used together to generate the modulation vector. It not only weights the channels inside each branch, but also weights different branches. If viewed from the aforementioned perspective of the kernel pyramid, such an operation can be seen as a relative modulation of the different kernel pyramids contained in the iterative structure of the HMU.

Besides, in our HMU, C is set to 32. The number of channels of the final output feature of the HMU is the same as the input feature, both are 64. We also list the algorithm of the iteration structure in Alg. 1 to present the process more clearly and to complement the related statement in the main text.

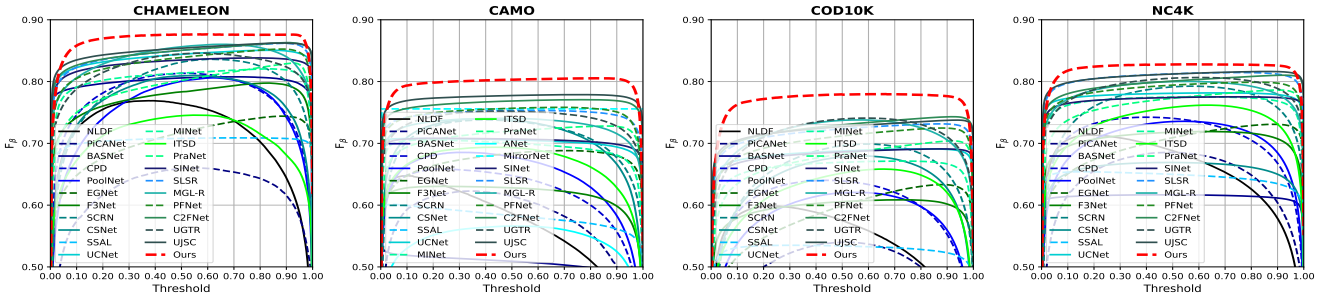
3. More Comparisons

3.1. PR & F_β curves of COD Methods

In Fig. 6, we show the PR & F_β curves of different methods on four COD datasets. The red curve represents our method.



(a) PR curves.



(b) F_β curves.

Figure 6. PR and F_β curves of the proposed model and recent SOTA algorithms over four COD datasets.

3.2. Comparisons of Param. & FLOPs

In Tab. 1, we list the number of parameters and FLOPs of existing COD methods and ours. Our method provides a performance-robust solution with the second-smallest amount of parameters for the COD task. But there may be still some redundancy in the design of the inference structure. The adopted explicit scale-independent design may bring additional inference cost. We will explore and improve this in future work.

3.3. Intermediate Feature Maps of the Decoder

We show the intermediate feature maps from different stages of the decoder in Fig. 1.

3.4. Effectiveness of UAL

In Fig. 3, we visualize the histogram maps of all results on CHAMELEON [18].

3.5. Different Forms of λ

The different adjustment functions of the coefficient λ and their results of UAL are list in Tab. 2.

3.6. Different Forms of UAL

The different forms of UAL are shown in Fig. 4.

3.7. Performance in More Complex Scenes

Actually, COD10K-TE is a very representative test dataset with rich and diverse scenarios and objects. Besides, there is also a very complex small-scale dataset CPD1K [39]. Tab. 3 shows the results of our method and some state-of-the-art competitors (all are trained without CPD1K-TR). The test results on CPD1K-TE can reflect the adaptability of the model to complex scenarios. The experiment shows the superior performance of our method in more complex scenarios.

Table 3. Comparison results of methods trained without CPD1K-TR on CPD1K-TE [39].

Model	$S_m \uparrow$	$F_\beta^\omega \uparrow$	MAE $\downarrow$	$F_\beta \uparrow$	$E_m \uparrow$
ZoomNet	**0.759**	**0.537**	**0.011**	**0.578**	**0.843**
C^2FNet	0.743	0.495	0.016	0.528	0.840
PFNet	0.722	0.460	0.017	0.494	0.819

4. Experiments on SOD

In order to show good generalization and further verify the rationality of the structural design, we evaluate the proposed model on the SOD task.

4.1. Datasets

Our experiment on SOD is based on the existing five SOD datasets, DUT-OMRON [29] (5168), DUTS [20] (10553 + 5017), ECSSD [28] (1000), HKU-IS [7] (4447) and Pascal-S [7] (850). We only use the training set of DUTS for training. During the test phase, we use the remaining data for inference.

Table 4. More detailed comparison results on the SOD task. The best results are highlighted in **red**, **green** and **blue**. These results are based on the VGG [17], ResNet [5] and T2T-ViT [31] version of the corresponding method.

Model	Backbone	Year	DUT-OMRON					DUTS-TE					ECSSD					HKU-IS					PASCAL-S				
			$S_m \uparrow$	$F^{\omega}_{\beta} \uparrow$	MAE $\downarrow$	$F_{\beta} \uparrow$	$E_m \uparrow$	$S_m \uparrow$	$F^{\omega}_{\beta} \uparrow$	MAE $\downarrow$	$F_{\beta} \uparrow$	$E_m \uparrow$	$S_m \uparrow$	$F^{\omega}_{\beta} \uparrow$	MAE $\downarrow$	$F_{\beta} \uparrow$	$E_m \uparrow$	$S_m \uparrow$	$F^{\omega}_{\beta} \uparrow$	MAE $\downarrow$	$F_{\beta} \uparrow$	$E_m \uparrow$	$S_m \uparrow$	$F^{\omega}_{\beta} \uparrow$	MAE $\downarrow$	$F_{\beta} \uparrow$	$E_m \uparrow$
RAS [2]	VGG16	2018	0.814	0.695	0.062	0.731	0.860	0.839	0.740	0.059	0.779	0.889	0.893	0.857	0.056	0.887	0.931	0.887	0.843	0.045	0.875	0.940	0.793	0.735	0.106	0.790	0.846
MLMSNet [24]	VGG16	2019	0.809	0.681	0.064	0.710	0.848	0.862	0.761	0.049	0.792	0.907	0.911	0.871	0.045	0.890	0.944	0.907	0.859	0.039	0.878	0.950	0.845	0.785	0.075	0.814	0.893
PAGENet [21]	VGG16	2019	0.824	0.722	0.062	0.743	0.858	0.854	0.769	0.052	0.793	0.896	0.912	0.886	0.042	0.904	0.947	0.903	0.865	0.037	0.884	0.948	0.838	0.789	0.079	0.819	0.885
PiCANet [10]	ResNet50	2018	0.832	0.695	0.065	0.729	0.876	0.869	0.755	0.051	0.791	0.920	0.917	0.867	0.046	0.890	0.952	0.904	0.840	0.043	0.866	0.950	0.852	0.779	0.078	0.812	0.899
BASNet [16]	ResNet34	2019	0.836	0.751	0.056	0.767	0.871	0.866	0.803	0.048	0.823	0.903	0.916	0.904	0.037	0.917	0.951	0.909	0.889	0.032	0.902	0.951	0.834	0.797	0.079	0.824	0.883
CPD [25]	ResNet50	2019	0.825	0.719	0.056	0.742	0.868	0.869	0.795	0.043	0.821	0.914	0.918	0.898	0.037	0.913	0.951	0.905	0.875	0.034	0.892	0.950	0.844	0.800	0.074	0.827	0.888
PoolNet [9]	ResNet50	2019	0.831	0.725	0.054	0.747	0.867	0.887	0.817	0.037	0.840	0.926	0.926	0.904	0.035	0.918	0.956	0.919	0.888	0.030	0.903	0.958	0.864	0.819	0.067	0.846	0.905
EGNet [35]	ResNet50	2019	0.841	0.738	0.053	0.760	0.878	0.887	0.816	0.039	0.839	0.927	0.925	0.903	0.037	0.918	0.955	0.918	0.887	0.031	0.902	0.958	0.850	0.804	0.076	0.833	0.892
HRS [32]	ResNet50	2019	0.772	0.645	0.066	0.690	0.841	0.829	0.746	0.051	0.791	0.899	0.883	0.859	0.054	0.894	0.934	0.882	0.851	0.042	0.883	0.941	0.799	0.744	0.091	0.792	0.866
SCRN [26]	ResNet50	2019	0.837	0.720	0.056	0.749	0.875	0.885	0.803	0.040	0.833	0.925	**0.927**	0.900	0.037	0.916	0.956	0.916	0.876	0.034	0.894	0.956	**0.865**	0.813	0.066	0.840	0.906
F3Net [22]	ResNet50	2020	0.838	0.747	0.053	0.766	0.872	0.888	0.835	**0.035**	0.852	0.927	0.924	0.912	**0.033**	**0.925**	0.955	0.917	0.900	**0.028**	0.910	0.958	0.857	0.823	0.064	0.843	0.901
GCPANet [3]	ResNet50	2020	0.839	0.734	0.056	0.756	0.869	0.891	0.821	0.038	0.841	0.929	**0.927**	0.903	0.035	0.916	0.955	0.920	0.889	0.031	0.901	0.958	0.864	0.819	**0.063**	0.840	0.906
LDF [23]	ResNet50	2020	0.839	0.752	**0.052**	0.770	0.869	0.892	**0.845**	**0.034**	0.861	0.930	0.924	**0.915**	0.034	**0.927**	0.954	0.919	**0.904**	**0.028**	0.913	0.958	0.859	**0.829**	**0.062**	**0.851**	0.905
DFI [8]	ResNet50	2020	0.840	0.738	0.055	0.762	0.877	0.887	0.817	0.039	0.840	0.928	**0.927**	0.906	0.035	0.920	0.957	0.919	0.890	0.031	0.903	**0.961**	0.864	0.824	0.066	0.849	0.907
GateNet [37]	ResNet50	2020	0.838	0.729	0.055	0.757	0.876	0.885	0.809	0.040	0.837	0.928	0.920	0.894	0.040	0.913	0.952	0.915	0.880	0.033	0.897	0.955	0.854	0.804	0.071	0.835	0.900
ITSD [40]	ResNet50	2020	0.840	0.750	0.061	0.768	0.880	0.885	0.824	0.041	0.840	0.929	0.925	0.910	0.034	0.921	**0.959**	0.917	0.894	0.031	0.904	0.960	0.859	0.823	0.066	0.843	**0.910**
MINet [15]	ResNet50	2020	0.833	0.738	0.056	0.757	0.869	0.884	0.825	0.037	0.844	0.927	0.925	0.911	**0.033**	0.923	0.957	0.919	0.897	0.029	0.909	0.960	0.854	0.818	0.066	0.841	0.901
VST [11]	T2T-ViT$_t$-14	2021	**0.850**	0.755	0.058	**0.774**	**0.888**	**0.896**	0.828	0.037	0.845	**0.939**	**0.932**	0.910	**0.033**	0.920	**0.964**	**0.928**	0.897	0.029	0.907	**0.968**	**0.871**	0.827	**0.062**	0.847	**0.918**
SAMNet [12]	Handcraft	2021	0.830	0.699	0.065	0.734	0.877	0.849	0.729	0.058	0.768	0.901	0.907	0.858	0.050	0.883	0.945	0.898	0.837	0.045	0.864	0.946	0.822	0.743	0.095	0.784	0.869
SGL-KRN [27]	ResNet50	2021	**0.846**	**0.765**	**0.049**	**0.783**	**0.885**	**0.893**	**0.847**	**0.034**	**0.865**	**0.939**	0.923	0.910	0.036	0.924	0.954	**0.921**	**0.904**	**0.028**	**0.915**	**0.961**	0.854	0.823	0.070	0.849	0.900
CTDNet [38]	ResNet50	2021	**0.844**	**0.762**	**0.052**	**0.779**	**0.881**	**0.893**	**0.847**	**0.034**	**0.862**	**0.935**	0.925	**0.915**	**0.032**	**0.927**	0.956	**0.921**	**0.909**	**0.027**	**0.918**	**0.961**	0.859	**0.829**	0.064	**0.851**	0.904
Auto-MSFNet [34]	ResNet50	2021	0.832	**0.757**	**0.050**	0.772	0.875	0.877	0.841	**0.034**	0.855	0.931	0.914	**0.916**	**0.033**	**0.927**	0.954	0.908	0.903	**0.027**	0.912	0.959	0.849	**0.830**	**0.063**	**0.852**	0.902
Ours	ResNet50	2021	0.841	0.755	0.053	0.771	0.872	**0.900**	**0.854**	**0.033**	**0.866**	**0.936**	**0.935**	**0.926**	**0.027**	**0.933**	**0.963**	**0.931**	**0.918**	**0.023**	**0.923**	**0.967**	**0.869**	**0.844**	**0.057**	**0.860**	**0.917**

4.2. Implementation Details

For a fair comparison on SOD, the proposed model is retrained on DUTS [20] following the training strategies and techniques of [15, 22, 23, 26, 36]. The learning rate is initialized to 0.05 and follows a linear warm-up and linear decay strategy. And the main scale is changed to 352×352 to achieve a trade-off between performance and speed. The model tends to converge after 50 epochs with a batch size of 22.

4.3. Comparisons with State-of-the-arts

We compare the proposed model with 22 existing methods. All the results are listed in Tab. 4 and shown in Fig. 7. Our model outperforms all these competitors, which shows that the proposed model can deal with the more general binary segmentation task.

5. Limitations and Future Work

Although our ZoomNet provides a powerful and effective solution for the COD task, some limitations still exist and are worth exploring further.

1. In the current work, the shared feature extraction structure explicitly collects complementary information from different scales on the image pyramid, which is designed to mimic the behavior of zooming in and out. But for human beings, the process of information extraction and integration should be implicit and internalized in the process of knowledge learning. Moreover, this explicit scale-independent design also brings the additional inference cost. Although our method has achieved good performance on COD and SOD tasks, the inference speed is still slightly slower than the current fastest method, C2FNet [19].

2. Besides, there is still room for improvement in the way of mining effective clues from small-scale features in SIU.

In future work, we will try to further simplify the inference structure to make it more in line with the actual human decision-making process and optimize the ability of our method to extract contextual cues from small-scale features.

References

[1] Liang-Chieh Chen, George Papandreou, Iasonas Kokkinos, Kevin Murphy, and Alan L Yuille. Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 40(4):834–848, 2017. 1

[2] Shuhan Chen, Xiuli Tan, Ben Wang, and Xuelong Hu. Reverse attention for salient object detection. In *Proceedings of European Conference on Computer Vision*, pages 234–250, 2018. 6

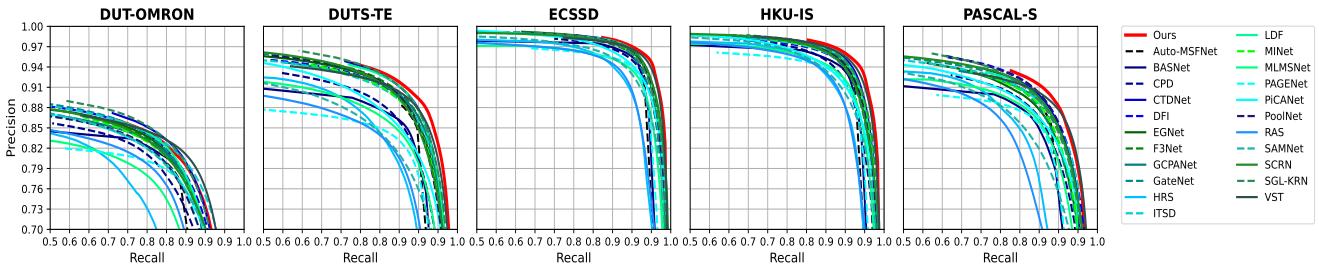
[3] Zuyao Chen, Qianqian Xu, Runmin Cong, and Qingming Huang. Global context-aware progressive aggregation network for salient object detection. In *AAAI Conference on Artificial Intelligence*, volume 34, pages 10599–10606, 2020. 6

[4] Deng-Ping Fan, Ge-Peng Ji, Guolei Sun, Ming-Ming Cheng, Jianbing Shen, and Ling Shao. Camouflaged object detection. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, pages 2777–2787, 2020. 3

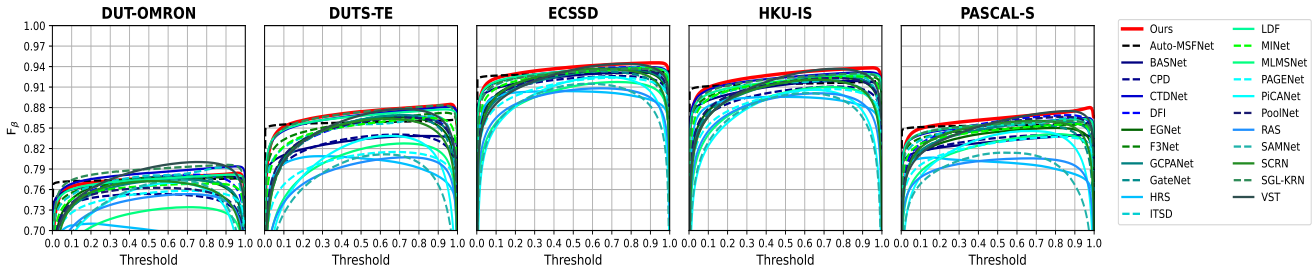
[5] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, pages 770–778, 2016. 1, 6

[6] Aixuan Li, Jing Zhang, Yunqiu Lyu, Bowen Liu, Tong Zhang, and Yuchao Dai. Uncertainty-aware joint salient object and camouflaged object detection. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, 2021. 3

[7] Guanbin Li and Yizhou Yu. Visual saliency based on multiscale deep features. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, pages 5455–5463, 2015. 5



(a) PR curves.



(b) F_β curves.

Figure 7. PR and F_β curves of the proposed model and recent SOTA algorithms over five SOD datasets.

[8] Jiang-Jiang Liu, Qibin Hou, and Ming-Ming Cheng. Dynamic feature integration for simultaneous detection of salient object, edge and skeleton. *IEEE Transactions on Image Processing*, pages 1–15, 2020. 6

[9] Jiang-Jiang Liu, Qibin Hou, Ming-Ming Cheng, Jiashi Feng, and Jianmin Jiang. A simple pooling-based design for real-time salient object detection. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, pages 3912–3921, 2019. 6

[10] Nian Liu, Junwei Han, and Ming-Hsuan Yang. Picanet: Learning pixel-wise contextual attention for saliency detection. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, pages 3089–3098, 2018. 6

[11] Nian Liu, Ni Zhang, Kaiyuan Wan, Ling Shao, and Junwei Han. Visual saliency transformer. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 4722–4732, October 2021. 6

[12] Yun Liu, Xin-Yu Zhang, Jia-Wang Bian, Le Zhang, and Ming-Ming Cheng. SAMNet: Stereoscopically attentive multi-scale network for lightweight salient object detection. *IEEE Transactions on Image Processing*, 30:3804–3814, 2021. 6

[13] Yunqiu Lyu, Jing Zhang, Yuchao Dai, Aixuan Li, Bowen Liu, Nick Barnes, and Deng-Ping Fan. Simultaneously localize, segment and rank the camouflaged objects. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, 2021. 3

[14] Haiyang Mei, Ge-Peng Ji, Ziqi Wei, Xin Yang, Xiaopeng Wei, and Deng-Ping Fan. Camouflaged object segmentation with distraction mining. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, June 2021. 3

[15] Youwei Pang, Xiaoqi Zhao, Lihe Zhang, and Huchuan Lu. Multi-scale interactive network for salient object detection. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, pages 9410–9419, June 2020. 6

[16] Xuebin Qin, Zichen Zhang, Chenyang Huang, Chao Gao, Masood Dehghan, and Martin Jagersand. Basnet: Boundary-aware salient object detection. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, pages 7479–7489, 2019. 6

[17] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014. 6

[18] Przemysław Skurowski, Hassan Abdulameer, Jakub Błaszczyk, Tomasz Depta, Adam Kornacki, and Przemysław Kozieł. Animal camouflage analysis: Chameleon database, 2017. http://kgwisc.aei.polsl.pl/index.php/pl/dataset/63-animal-camouflage-analysis. 3, 5

[19] Yujia Sun, Geng Chen, Tao Zhou, Yi Zhang, and Nian Liu. Context-aware cross-level fusion network for camouflaged object detection. In Zhi-Hua Zhou, editor, *International Joint Conference on Artificial Intelligence*, pages 1025–1031. ijcai.org, 2021. 3, 6

[20] Lijun Wang, Huchuan Lu, Yifan Wang, Mengyang Feng, Dong Wang, Baocai Yin, and Xiang Ruan. Learning to detect salient objects with image-level supervision. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, pages 136–145, 2017. 5, 6

[21] Wenguan Wang, Shuyang Zhao, Jianbing Shen, Steven CH Hoi, and Ali Borji. Salient object detection with pyramid attention and salient edges. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, pages 1448–1457, 2019. 6

[22] Jun Wei, Shuhui Wang, and Qingming Huang. F^3net: Fusion, feedback and focus for salient object detection. In

AAAI Conference on Artificial Intelligence, volume 34, pages 12321–12328, 2020. 6

[23] Jun Wei, Shuhui Wang, Zhe Wu, Chi Su, Qingming Huang, and Qi Tian. Label decoupling framework for salient object detection. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, pages 13022–13031, 2020. 6

[24] Runmin Wu, Mengyang Feng, Wenlong Guan, Dong Wang, Huchuan Lu, and Errui Ding. A mutual learning method for salient object detection with intertwined multi-supervision. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, pages 8150–8159, 2019. 6

[25] Zhe Wu, Li Su, and Qingming Huang. Cascaded partial decoder for fast and accurate salient object detection. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, pages 3907–3916, 2019. 6

[26] Zhe Wu, Li Su, and Qingming Huang. Stacked cross refinement network for edge-aware salient object detection. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 7264–7273, 2019. 6

[27] Binwei Xu, Haoran Liang, Ronghua Liang, and Peng Chen. Locate globally, segment locally: A progressive architecture with knowledge review network for salient object detection. *AAAI Conference on Artificial Intelligence*, 35(4):3004–3012, May 2021. 6

[28] Qiong Yan, Li Xu, Jianping Shi, and Jiaya Jia. Hierarchical saliency detection. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, pages 1155–1162, 2013. 5

[29] Chuan Yang, Lihe Zhang, Huchuan Lu, Xiang Ruan, and Ming-Hsuan Yang. Saliency detection via graph-based manifold ranking. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, pages 3166–3173, 2013. 5

[30] Fan Yang, Qiang Zhai, Xin Li, Rui Huang, Ao Luo, Hong Cheng, and Deng-Ping Fan. Uncertainty-guided transformer reasoning for camouflaged object detection. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 4146–4155, October 2021. 3

[31] Li Yuan, Yunpeng Chen, Tao Wang, Weihao Yu, Yujun Shi, Zi-Hang Jiang, Francis E.H. Tay, Jiashi Feng, and Shuicheng Yan. Tokens-to-token vit: Training vision transformers from scratch on imagenet. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 558–567, October 2021. 6

[32] Yi Zeng, Pingping Zhang, Jianming Zhang, Zhe Lin, and Huchuan Lu. Towards high-resolution salient object detection. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 7233–7242, 2019. 6

[33] Qiang Zhai, Xin Li, Fan Yang, Chenglizhao Chen, Hong Cheng, and Deng-Ping Fan. Mutual graph learning for camouflaged object detection. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, 2021. 3

[34] Miao Zhang, Tingwei Liu, Yongri Piao, Shunyu Yao, and Huchuan Lu. *Auto-MSFNet: Search Multi-Scale Fusion Network for Salient Object Detection*, page 667–676. Association for Computing Machinery, New York, NY, USA, 2021. 6

[35] Jia-Xing Zhao, Jiang-Jiang Liu, Deng-Ping Fan, Yang Cao, Jufeng Yang, and Ming-Ming Cheng. Egnet: Edge guidance network for salient object detection. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 8779–8788, 2019. 6

[36] Ting Zhao and Xiangqian Wu. Pyramid feature attention network for saliency detection. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, pages 3080–3089, 2019. 6

[37] Xiaoqi Zhao, Youwei Pang, Lihe Zhang, Huchuan Lu, and Lei Zhang. Suppress and balance: A simple gated network for salient object detection. In *Proceedings of European Conference on Computer Vision*, pages 35–51, 2020. 1, 6

[38] Zhirui Zhao, Changqun Xia, Chenxi Xie, and Jia Li. *Complementary Trilateral Decoder for Fast and Accurate Salient Object Detection*, page 4967–4975. Association for Computing Machinery, New York, NY, USA, 2021. 6

[39] Yunfei Zheng, Xiongwei Zhang, Feng Wang, Tieyong Cao, Meng Sun, and Xiaobing Wang. Detection of people with camouflage pattern via dense deconvolution network. *IEEE Signal Processing Letters*, 26(1):29–33, Jan 2019. 5

[40] Huajun Zhou, Xiaohua Xie, Jian-Huang Lai, Zixuan Chen, and Lingxiao Yang. Interactive two-stream decoder for accurate and fast saliency detection. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, pages 9138–9147, 2020. 6