Supplementary Materials for Revisiting Image Pyramid Structure for High Resolution Salient Object Detection

Taehun Kim[1][0000−0001−9322−9741], Kunhee Kim[1], Joonyeong Lee[1], Dongmin Cha[1], Jiho Lee[1], and Daijin Kim[1]

[1]Dept. of CSE, Pohang University of Science and Technology (POSTECH), Korea
{taehoon1018, kunkim, joonyeonglee, cardongmin, jiholee, dkim}@postech.ac.kr
https://github.com/plemeri/InSPyReNet.git

1 Method Details

1.1 Backbone Networks

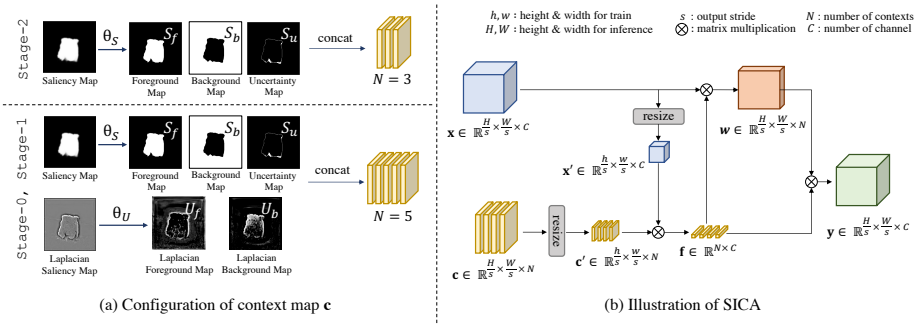
In this section, we describe how the feature maps from the backbone network is retrieved. Since we adopt two different types of backbone networks, we explain the minor details of each model.

For Res2Net [1] backbone network, we use 26w×8s setting. Because the main difference between ResNet [2] and Res2Net is a building block, the overall architecture is identical. So, to explain where we extract the feature maps from the Res2Net backbone, we refer to the ResNet paper. The feature map for the `Stage-1` is extracted from conv1, and for `Stage-j`, where $j > 1$, we extract feature maps from the last layer which has a name of conv$j$_x (*e.g.*, conv4_6 of Res2Net50 for `Stage-4`).

For Swin Transformer [3], it is slightly different from conventional CNN-based models in the fact that they divide an image into tokens. Unlike ResNet (or Res2Net), there is a layer which works similar to the stem layer (conv_1), a patch partition layer which generates a size of 4×4 patches and aggregates to its spatial dimension for each embedding. The feature map for the `Stage-1` is extracted from the patch partition layer, and for `Stage-j`, where $j > 1$, we extract feature maps from the last layer of stage j[1]. Since Vision Transformers interpret an image as a sequence of patches, we rearrange patches to a 2-dimensional feature map for each stage.

Overall, the hierarchical structure is identical to each other, so we can easily adopt Swin Transformer to other methods, so as we mention in the main paper, we implement 5 SotA models, Chen *et al.* [4], F^3Net [5], LDF [6], MINet [7] and PA-KRN [8], and conduct experiments with Res2Net and Swin Transformer backbones.

[1] Note that the term 'stage j' is from [3].



(a) Configuration of context map **c** (b) Illustration of SICA

Fig. 1. Context map configuration (a) and details of SICA (b).

1.2 Scale Invariant Context Attention

In this section, we provide more details of SICA. Since the overall computation of SICA is similar to the OCRNet, we have to obtain soft object regions (or, context maps), which is a coarse segmentation maps corresponds to each class. However, salient object detection is a class-agnostic segmentation task, so we only have a salient region, and its inverse area. UACANet [9] first proposed uncertainty area to provide additional context maps, which are often related to the object boundary region. We notice that the saliency map has an object contextual information, but the Laplacian saliency map also has a boundary context. Therefore, we design SICA with additional context clues from the Laplacian saliency map.

However, while UACA set the threshold value as 0.5, we argue that a fixed threshold does not guarantee the optimal saliency map. F-measure based loss (FLoss) [10] claims that the optimal threshold obtained with an exhaustive search on the test dataset is impractical for real-world application. While they still used exhaustive search after applying FLoss to find the optimal threshold, it is still clear that 0.5 is not the optimal threshold for most cases. We also claim that the threshold needs to be adaptive for extracting context information, and needs to be different for each stage because the saliency prediction from each stage may have different statistics due to the scale differences. So, we choose to learn threshold values for each stage and use them to extract meaningful context regions. We illustrate the context map configuration details in Fig. 1a.

First, we compute context maps as follows,

$$\begin{aligned} S_f = \max(S - \theta_S, 0),\ S_b = \max(\theta_S - S, 0), \\ S_u = \theta_S - \text{abs}(S - \theta_S), \end{aligned} \tag{1}$$

$$U_f = \max(U - \theta_U, 0),\ U_b = \max(\theta_U - U, 0), \tag{2}$$

where we denote trainable threshold θ_S and θ_U for input saliency map S and Laplacian saliency map U, respectively, and initialize them with 0.5. S_f, S_b, and S_u are foreground, background, and uncertainty context maps from S. Likewise,

U_f and U_b are foreground and background context map from U. Note that because there is no Laplacian saliency map in `Stage-3`, there are no U_f and U_b for SICA in `Stage-2`. Then, we aggregate context maps for simplicity as follows,

$$\mathbf{c} = \begin{cases} [S_f, S_b, S_u], & \text{if } \texttt{Stage-2}. \\ [S_f, S_b, S_u, U_f, U_b], & \text{otherwise}. \end{cases} \tag{3}$$

$\mathbf{c}$ consists of three or five context maps depending on its stage (see Fig. 1a).

Then, the input feature map from the encoder and the saliency map from the decoder are resized to the size from the training session. We denote the size of the input image as $H \times W$, the output stride of the current stage as s, and the number of channels and context maps as C and N respectively. So, the input feature map $\mathrm{x} \in \mathbb{R}^{\frac{H}{s} \times \frac{W}{s} \times C}$ and input context maps $\mathrm{c} \in \mathbb{R}^{\frac{H}{s} \times \frac{W}{s} \times N}$ are resized according to the shape from training time $h \times w$ with bi-linear interpolation. The resized feature map $\mathrm{x}' \in \mathbb{R}^{\frac{h}{s} \times \frac{w}{s} \times C}$ and context map $\mathrm{c}' \in \mathbb{R}^{\frac{h}{s} \times \frac{w}{s} \times N}$ is used to compute object region representation $\mathrm{f} \in \mathbb{R}^{N \times C}$.

$$\mathbf{f}_k = \sum_{(x,y) \in \mathcal{I}} \mathbf{c}_k(x, y)\mathbf{x}(x, y), \tag{4}$$

where $\mathcal{I}$ is a lattice domain of S and U. Since the matrix multiplication is done on the spatial dimension, f has a same shape with or without resize, yet has better representation ability.

Subsequently, we compute the attention score w by computing the similarity score between $\mathbf{f}$ and $\mathbf{x}(x, y)$,

$$w_k(x, y) = \frac{\exp(\mathcal{T}_{\mathbf{x}}(\mathbf{x}(x, y))^\top \mathcal{T}_{\mathbf{f}}(\mathbf{f}_k))}{\sum_{l=1}^{K} \exp(\mathcal{T}_{\mathbf{x}}(\mathbf{x}(x, y))^\top \mathcal{T}_{\mathbf{f}}(\mathbf{f}_l))}, \tag{5}$$

where $\mathcal{T}_{\mathbf{x}}(\cdot)$ and $\mathcal{T}_{\mathbf{f}}(\cdot)$ denotes transformation functions implemented by consecutive convolution layer, batch normalization, and ReLU activation. Lastly, with context representation vector $\mathbf{f}$ and attention map w as a weighting factor, we compute a context enhanced feature map $\mathbf{y}$ as follows,

$$\mathbf{y}(x, y) = \mathcal{T}_{\mathbf{y}}(\sum_{l=1}^{K} w_l(x, y)\mathcal{T}_{\mathbf{f}'}(\mathbf{f}_l)), \tag{6}$$

where $\mathcal{T}_{\mathbf{y}}(\cdot)$ and $\mathcal{T}_{\mathbf{f}'}(\cdot)$ are transformation functions and $K = 3$ if `Stage-2`, otherwise, $K = 5$ (see Eq. (3) and Fig. 1a). The input and output feature maps of SICA are concatenated and forwarded to a simple decoder with convolution blocks to predict the Laplacian saliency map (Fig. 1b).

1.3 Implementation of Image Pyramid Operations.

We implement EXPAND and REDUCE operations [11] in Pytorch [12]. We provide a source code of both operations in Algorithm 1. We obtain the 1D

Algorithm 1 PyTorch pseudocode of image pyramid operations.

```
# Module for image pyramid operations (EXPAND, REDUCE)
# ksize: kernel size for Gaussian filter
# sigma: standard deviation for Gaussian filter
# channels: number of channels for pyramid operation
#           (For saliency map, set 1. For RGB image, set 3.)
class ImagePyramid:
    def __init__(self, ksize=7, sigma=1, channels=1):
        self.ksize = ksize
        self.sigma = sigma
        self.channels = channels

        k = cv2.getGaussianKernel(ksize, sigma)
        k = np.outer(k, k)
        k = torch.tensor(k).float()
        self.kernel = k.repeat(channels, 1, 1, 1)

    # call to use GPU
    def cuda(self):
        self.kernel = self.kernel.cuda()
        return self

    # EXPAND operation
    def expand(self, x):
        z = torch.zeros_like(x)
        x = torch.cat([x, z, z, z], dim=1)
        x = F.pixel_shuffle(x, 2)
        x = F.pad(x, (self.ksize // 2, ) * 4, mode='reflect')
        x = F.conv2d(x, self.kernel * 4, groups=self.channels)
        return x

    # REDUCE operation
    def reduce(self, x):
        x = F.pad(x, (self.ksize // 2, ) * 4, mode='reflect')
        x = F.conv2d(x, self.kernel, groups=self.channels)
        return x[:, :, ::2, ::2]
```

Gaussian kernel with `cv2.getGaussianKernel` from OpenCV [13], and use outer operation to generate the 2D Gaussian kernel. For EXPAND operation, we use pixel-shuffle operation from PyTorch, and we used even indices for REDUCE operation.

For pyramid blending, we use `cv2.getStructuringElement` function with `cv2.MORPH_ELIPSE` argument for dilation and erosion. Also, we use `dilation` and `erosion` functions from `kornia.morphology` [14]. The kernel size for dilation and erosion is set to 5, 9, and 17 for `Stage-2`, `Stage-1`, and `Stage-0` respectively.

2 Experiments

2.1 Ablation Studies

Gaussian Filter in Image Pyramid. Even the kernel size k and the standard deviation σ of the Gaussian kernel g for image pyramid operation is usually set to 5 and 1 respectively, we compare our image pyramid operations with different kernel sizes and standard deviations. Results in Tab. 1 shows that when the σ gets larger, the overall performance decreases. Furthermore, when k is 7 and σ is set to 1, we obtain the best result among different settings.

Table 1. Left: Ablation study of kernel size k and the standard deviation σ of the Gaussian kernel $G(k, \sigma)$ for image pyramid operations of InSPyReNet (Res2Net50) on DUTS-TE and DUT-OMRON. Right: Ablation study of training strategies for InSPyReNet (SwinB) on DAVIS-S and HRSOD-TE. *pred* and *gt* denotes the image pyramid structure applied in prediction and ground truth respectively. S.G. denotes Stop-Gradient. $\mathcal{L}_{pc}$ denotes pyramidal consistency loss.

k	σ	DUTS-TE			DUT-OMRON		
		S_α	$F_{\max}$	MAE	S_α	$F_{\max}$	MAE
5	1	0.902	0.890	0.037	0.839	0.783	0.059
5	3	0.897	0.882	0.041	0.837	0.779	0.059
5	5	0.888	0.876	0.042	0.834	0.770	0.060
7	1	**0.904**	**0.892**	**0.035**	**0.845**	**0.791**	0.059
7	3	0.897	0.884	0.037	0.841	0.777	**0.058**
7	5	0.888	0.878	0.038	0.833	0.774	0.061

pred	*gt*	S.G.	$\mathcal{L}_{pc}$	DAVIS-S				HRSOD-TE			
				$S_\alpha \uparrow$	$F_{\max} \uparrow$	MAE↓	mBA↑	$S_\alpha \uparrow$	$F_{\max} \uparrow$	MAE↓	mBA↑
✓				0.935	0.937	0.016	0.693	0.931	0.933	0.023	0.682
	✓			0.937	0.939	0.014	0.714	0.934	0.937	0.019	0.712
✓	✓			0.938	0.935	0.016	0.699	0.932	0.931	0.022	0.695
✓	✓	✓		0.945	0.942	0.014	0.719	0.944	0.942	0.017	0.715
✓	✓	✓	✓	0.955	**0.959**	0.010	0.727	0.947	0.948	0.018	0.729
✓	✓	✓	✓	**0.962**	0.959	**0.009**	**0.743**	**0.952**	**0.949**	**0.016**	**0.738**

Training Strategies. To demonstrate the effect of our training strategies such as supervision under image pyramid structure (*pred* and *gt*), pyramidal consistency loss ($\mathcal{L}_{pc}$), stop-gradient(S.G.), we train InSPyReNet with different settings and evaluate on HR benchmarks. Results in Tab. 1 shows that without image pyramid structure on prediction branch (*pred*) shows unsatisfactory results in terms of mBA. This is because while other strategies (*gt*, S.G., $\mathcal{L}_{pc}$) are considered as supervision strategies, *pred* is embedded as a model architecture. With other strategies, we can notice that S.G. provides some improvements in terms of SOD metrics, which means it gives more stable results for pyramid blending. Also, $\mathcal{L}_{pc}$ shows extra improvements for mBA, which means it ensures the image pyramid structure in a sense of high-frequency residual information of Laplacian image.

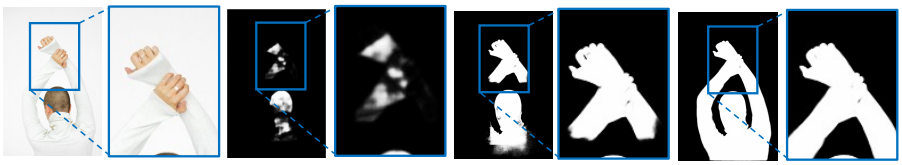


Fig. 2. Visual demonstration for ablation study of SICA. The sample is taken from AIM-500 [15].

SICA. As shown in Fig. 2, methods without SICA generates unpleasant artifacts on the boundary areas, which can be interpreted as the failure of Laplacian images of the saliency map from SICA. On the other hand, InSPyReNet with SICA shows detailed predictions, but without pyramid blending, it misses some major object parts. With SICA and pyramid blending shows best results, show-

ing less failure in terms of both capturing the whole salient object body parts and high-frequency object boundary details.

Table 2. Quantitative results of applying pyramid blending for previous pyramid-based SOD method (Chen *et al.* [4]) on three HR and two LR benchmarks. P.B. denotes pyramid blending. ↑ indicates larger the better, and ↓ indicates smaller the better.

Algorithms	DAVIS-S				HRSOD-TE				UHRSD-TE			
	S_α ↑	$F_{\max}$ ↑	MAE↓	mBA↑	S_α ↑	$F_{\max}$ ↑	MAE↓	mBA↑	S_α ↑	$F_{\max}$ ↑	MAE↓	mBA↑
w/o pyramid blending												
Chen *et al.* [4]	0.934	0.925	0.018	0.697	0.915	0.907	0.032	0.684	0.915	0.919	0.034	0.712
Ours	0.953	0.949	0.013	0.705	0.945	0.941	0.019	0.700	0.927	0.932	0.032	0.724
w/ pyramid blending												
Chen *et al.* [4]	0.916	0.893	0.023	0.583 (-16.4%)	0.909	0.891	0.033	0.590 (-13.7%)	0.903	0.906	0.042	0.590 (-17.1%)
Ours	0.962	0.958	0.009	0.732 (+3.8%)	0.952	0.955	0.015	0.732 (+4.6%)	0.932	0.938	0.029	0.741 (+2.3%)

Fig. 3. Visual comparison of applying pyramid blending for previous pyramid-based SOD method (Chen *et al.*) and InSPyReNet. Samples are taken from UHRSD [16]. *Best viewed by zooming in.*

2.2 Applying pyramid blending to previous Image Pyramid based Model

It is easy to apply pyramid blending if the base model outputs image pyramid of saliency map same as InSPyReNet. We choose Chen *et al.* [4] since it has a great reproducibility and has a pyramid structure. We trained Chen *et al.* with same backbone (SwinB) for fair comparison. As shown in Tab. 2, pyramid blending for Chen *et al.* worsen results especially for the mBA measure which is a major reason for pyramid blending. We also provide some qualitative results of Chen *et al.* with pyramid blending in Fig. 3, which shows some clear degradation. Thus, without a well-defined image pyramid based model designed for image blending, it cannot be used for HR prediction.

2.3 Training InSPyReNet with HR datasets

To demonstrate the potential of our method, we utilize HR datasets (HRSOD-TR, UHRSD-TR) alongside DUTS-TR for our training dataset. First, we use HR datasets for training with fixed LR scale (*i.e.*, 384×384), meaning that we do not use HR annotations and regard them as another LR datasets. Results show that our method well generalizes to the HR prediction with extra LR datasets (Tab. 3). Note that we do not include this experiments for our final results since even we resize HR datasets into LR scale, annotations are still remains high-quality thanks to the interpolation method, so it is not fair to claim that we are using only LR datasets.

Table 3. Quantitative results on three HR and two LR benchmarks for training with extra HR datasets. D: DUTS-TR, H: HRSOD-TR, U: UHRSD-TR. The best results for each metric are denoted as **bold**. ↑ indicates larger the better, and ↓ indicates smaller the better. * indicates that the dataset is resized into LR scale.

Train Datasets	HR benchmarks												LR benchmarks					
	DAVIS-S				HRSOD-TE				UHRSD-TE				DUTS-TE			DUT-OMRON		
	$S_\alpha\uparrow$	$F_{\max}\uparrow$	MAE↓	mBA↑	$S_\alpha\uparrow$	$F_{\max}\uparrow$	MAE↓	mBA↑	$S_\alpha\uparrow$	$F_{\max}\uparrow$	MAE↓	mBA↑	$S_\alpha\uparrow$	$F_{\max}\uparrow$	MAE↓	$S_\alpha\uparrow$	$F_{\max}\uparrow$	MAE↓
PGNet [16]																		
H,U	0.954	0.956	0.010	0.730	0.938	0.939	0.020	0.727	0.935	0.930	0.026	0.765	0.861	0.828	0.038	0.790	0.727	0.059
Ours Trained with LR scale (*i.e.*, 384×384)																		
D, H*	0.963	0.966	0.008	0.744	0.958	0.958	0.014	0.752	0.937	0.945	0.027	0.754	**0.936**	**0.934**	**0.022**	0.878	0.836	0.044
H*,U*	0.963	0.967	0.008	0.732	0.947	0.945	0.020	0.741	0.949	0.956	**0.020**	0.765	0.925	0.922	0.028	0.874	0.835	0.048
D,H*,U*	0.970	0.972	**0.007**	0.743	0.951	0.951	0.018	0.748	0.950	**0.957**	**0.020**	0.767	0.931	0.928	0.024	**0.880**	**0.837**	**0.042**
Ours Trained with HR scale (*i.e.*, 1024×1024)																		
D,H	0.972	0.976	**0.007**	**0.770**	**0.960**	**0.957**	**0.014**	0.766	0.936	0.938	0.028	0.785	0.934	0.927	0.023	0.859	0.799	0.049
H,U	**0.973**	**0.977**	**0.007**	**0.770**	0.956	0.956	0.018	**0.771**	**0.953**	**0.957**	**0.020**	**0.812**	**0.936**	0.932	0.024	0.872	0.823	0.046

Moreover, to understand the potential of InSPyReNet as is, we trained our method with HR datasets in HR scale. In this case, we do not deploy pyramid blending since we are not training in LR scale, and hence there is no LR pyramid to merge with. For HR training, we follow the training size from [16] (*i.e.*, 1024×1024). As shown in Tab. 3, our method shows great results with fully supervised manner for HR prediction, even we do not specifically design InSPyReNet for HR prediction without pyramid blending. This experiment shows that with simple image pyramid structure from our method can further be utilized for HR prediction with HR datasets. Overall, results trained with HR scale shows better performance especially for mBA, but slightly worse on LR benchmarks than results trained on LR scale. From this experiment, although the performance on LR and HR benchmarks tends to prefer the corresponding training scale, InSPyReNet well adopts to each other.

We also provide qualitative results in Fig. 4. Compared to the same setting of PGNet [16] trained with HRSOD-TR and UHRSD-TR, our method substantially outperforms the quality of high-frequency details of saliency maps. Moreover, we can notice that without UHRSD-TR whether we train with LR or HR scale, we cannot expect good results for complex scenes (first and third samples in Fig. 4). This is because while DUTS-TR and HRSOD-TR are in favor of "centered" objects (*e.g.*, our methods without UHRSD-TR in the first sample),

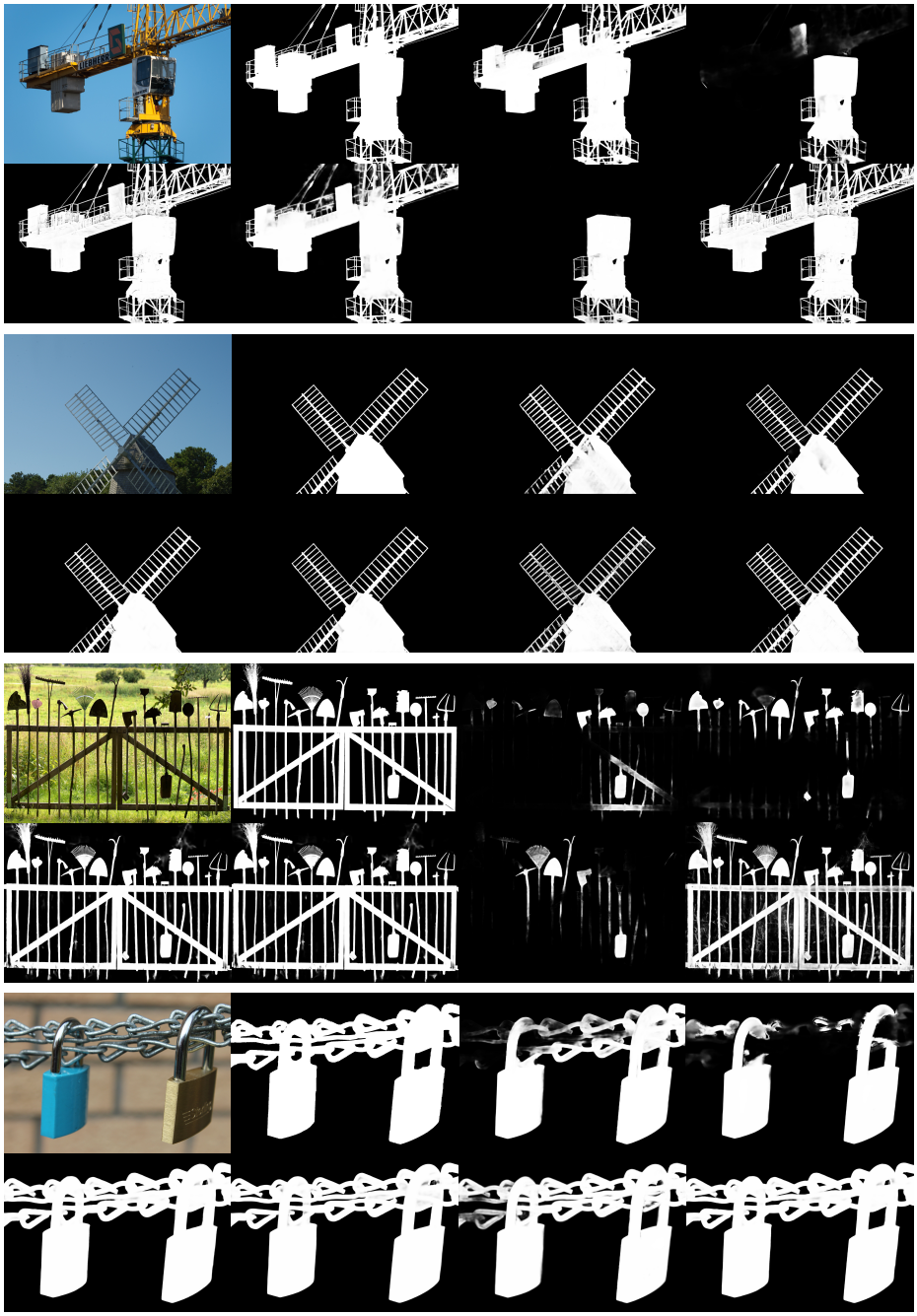


Fig. 4. Visual comparison of PGNet [16] (H, U) and our methods trained with HR datasets. Results are ordered as image, ground truth, PGNet, and *Ours*(D, H*) in the first row, and *Ours*(H*, U*), *Ours*(D, H*, U*), *Ours*(D, H), *Ours*(H, U) in the second row from left to right for each sample. *Best viewed by zooming in.*

while UHRSD-TR more focus on complex details which usually cover the whole image like thrid sample in Fig. 4.

Table 4. Quantitative results of IS-Net and our method trained with DIS5K. We trained our model with LR scale (*e.g.*, 384 × 384) and HR scale (*e.g.*, 1024 × 1024).

DIS-VD				DIS-TE1				DIS-TE2				DIS-TE3				DIS-TE4			
$S_\alpha\uparrow$	$F_{\max}\uparrow$	$HCE_\gamma\downarrow$	mBA↑	$S_\alpha\uparrow$	$F_{\max}\uparrow$	$HCE_\gamma\downarrow$	mBA↑	$S_\alpha\uparrow$	$F_{\max}\uparrow$	$HCE_\gamma\downarrow$	mBA↑	$S_\alpha\uparrow$	$F_{\max}\uparrow$	$HCE_\gamma\downarrow$	mBA↑	$S_\alpha\uparrow$	$F_{\max}\uparrow$	$HCE_\gamma\downarrow$	mBA↑
IS-Net [17]																			
0.813	0.791	1116	0.741	0.787	0.740	149	0.736	0.823	0.799	340	0.740	0.836	0.830	687	0.746	0.830	0.827	2888	0.743
Ours Trained with LR scale (*i.e.*, 384 × 384)																			
0.887	0.876	905	0.765	0.862	0.834	148	0.745	0.893	0.881	316	0.759	0.902	0.904	582	0.774	0.891	0.892	**2243**	0.779
Ours Trained with HR scale (*i.e.*, 1024 × 1024)																			
0.900	**0.889**	**904**	**0.800**	**0.873**	**0.845**	**110**	**0.797**	**0.905**	**0.894**	**255**	**0.803**	**0.918**	**0.919**	**522**	**0.808**	**0.905**	**0.905**	2336	**0.799**

2.4 Dichotomous Image Segmentation (DIS5K)

We trained our model and compare to the baseline model of DIS5K, IS-Net [17] which is currently shows SotA performance. To demonstrate the potential of our method which can be trained with LR scale (*e.g.*, 384×384) to boost up training time and reduce required resources, we trained our model with both LR and HR scales. In Tab. 4, our method shows regardless of training scale shows great performance compared to the baseline model, IS-Net [17]. Moreover, in Fig. 5, we provide qualitative results to demonstrate that our model can produce detailed output regardless of the training scale.

3 Discussion

Table 5. Comparison of boundary quality measures (BDE [18], mBA [19], BIoU [20]) with HR SOD methods. D: DUTS-TR, H: HRSOD-TR, U: UHRSD-TR. Three best results in order except our method are colored as red, blue, and green.

Algorithms	Train Datasets	DAVIS-S			HRSOD-TE		
		BDE	mBA	BIoU	BDE	mBA	BIoU
Zeng *et al.* [21]	D, H	44.359	0.618	0.662	88.017	0.623	0.659
Tang *et al.* [22]	D, H	14.266	0.716	0.785	46.495	0.693	0.744
PGNet [16]	D	34.957	0.707	0.769	46.923	0.693	0.749
PGNet [16]	D, H	14.463	0.716	0.790	45.292	0.714	0.772
PGNet [16]	H, U	12.725	0.730	0.814	57.147	0.727	0.781
Ours	D	-	0.743	0.850	-	0.738	0.826

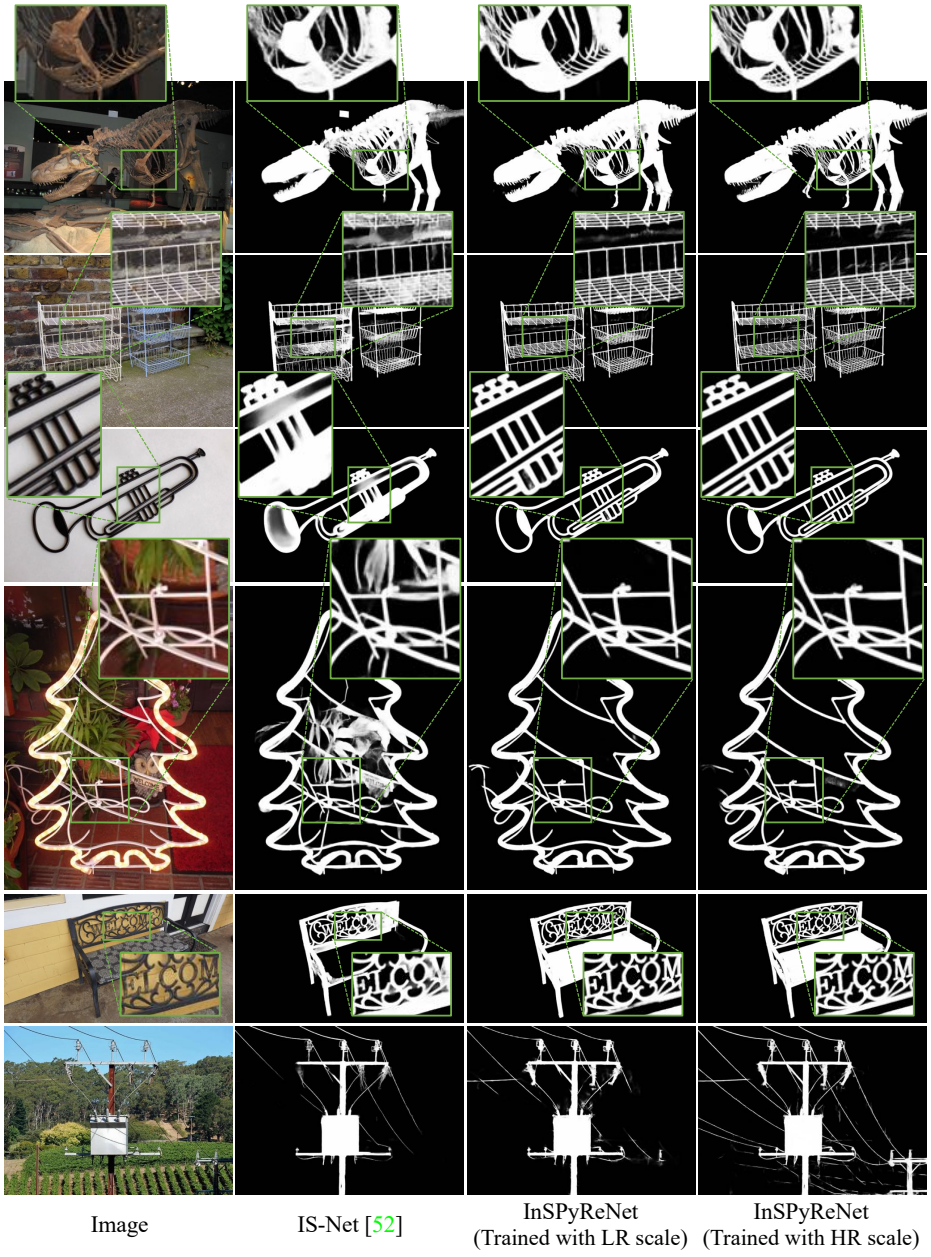


Fig. 5. Visual comparison of IS-Net [17] and our method trained with DIS5K. We trained our model with both LR and HR scales each. *Best viewed by zooming in.*

3.1 Selection of Boundary Metrics

Although many SOD methods dedicated to the HR benchmarks [16, 21, 22] use Boundary Displacement Error (BDE) [18], we suggest to use mean boundary accuracy (mBA) [19] instead for following reasons. First, it is substantially outdated metric for measuring boundary quality since BDE was proposed in 2002, when image segmentation methods highly depended on low-level signal analysis of the given image. Now we're in the era of deep learning. We can easily generate more accurate, high-quality segmentation results, and hence need to use metrics like mBA or Boundary IoU (BIoU) [20]. Second, we could not find any of official, non-official implementation related to the BDE, and the only source that we found is unable to access. While, mBA and BIoU have official implementations. We report mBA in the main paper because it does not require modification for SOD since [19] also works for a binary segmentation map like SOD, while BIoU requires major modification since the official code only provides BIoU embedded in the evaluation codes for Average Precision and Panoptic Quality. Third, the evaluation results with BDE shows inconsistent while mBA and BIoU shows consistent results as shown in Tab. 5. On the other hand, mBA and BIoU shows consistent results across different HR methods. Thus, we use mBA for boundary metric.

Fig. 6. Visual illustration for the failure case (first row: global context failure, second row: local detail failure) of InSPyReNet. *Best viewed by zooming in.*

3.2 Potential Vulnerability of InSPyReNet

While we show that our InSPyReNet can produce high-quality results in HR benchmarks, we can notice that there are some failure cases in terms of two different aspects. First, as shown in the first row from Fig. 6, if the LR branch

in pyramid blending fails to predict the saliency object, it suffers from global context failure. Second, even if the LR branch in pyramid blending successfully predict the saliency branch, we still have a chance to fail reconstructing local details when the HR branch fails to generate high-frequency details. In the second row from Fig. 6, the LR branch detected the body part of the bicycle, but from HR branch, it fails to predict spokes of the wheel and details of the front basket.

References

1. Gao, S.H., Cheng, M.M., Zhao, K., Zhang, X.Y., Yang, M.H., Torr, P.: Res2net: A new multi-scale backbone architecture. IEEE TPAMI **43** (2021) 652–662 1
2. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: CVPR. (2016) 770–778 1
3. Liu, Z., Lin, Y., Cao, Y., Hu, H., Wei, Y., Zhang, Z., Lin, S., Guo, B.: Swin transformer: Hierarchical vision transformer using shifted windows. arXiv preprint arXiv:2103.14030 (2021) 1
4. Chen, S., Tan, X., Wang, B., Hu, X.: Reverse attention for salient object detection. In: ECCV. (2018) 234–250 1, 6
5. Wei, J., Wang, S., Huang, Q.: F^3net: Fusion, feedback and focus for salient object detection. In: AAAI. Volume 34. (2020) 12321–12328 1
6. Wei, J., Wang, S., Wu, Z., Su, C., Huang, Q., Tian, Q.: Label decoupling framework for salient object detection. In: CVPR. (2020) 13025–13034 1
7. Pang, Y., Zhao, X., Zhang, L., Lu, H.: Multi-scale interactive network for salient object detection. In: CVPR. (2020) 9413–9422 1
8. Xu, B., Liang, H., Liang, R., Chen, P.: Locate globally, segment locally: A progressive architecture with knowledge review network for salient object detection. In: AAAI. (2021) 3004–3012 1
9. Kim, T., Lee, H., Kim, D.: Uacanet: Uncertainty augmented context attention for polyp segmentation. In: ACM MM. (2021) 2167–2175 2
10. Zhao, K., Gao, S., Wang, W., Cheng, M.M.: Optimizing the f-measure for threshold-free salient object detection. In: ICCV. (2019) 8849–8857 2
11. Burt, P., Adelson, E.: The laplacian pyramid as a compact image code. IEEE Transactions on Communications **31** (1983) 532–540 3
12. Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., et al.: Pytorch: An imperative style, high-performance deep learning library. Advances in neural information processing systems **32** (2019) 3
13. Bradski, G.: The OpenCV Library. Dr. Dobb's Journal of Software Tools (2000) 4
14. Riba, E., Mishkin, D., Ponsa, D., Rublee, E., Bradski, G.: Kornia: an open source differentiable computer vision library for pytorch. In: Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision. (2020) 3674–3683 4
15. Li, J., Zhang, J., Tao, D.: Deep automatic natural image matting. In: IJCAI. (2021) 800–806 5
16. Xie, C., Xia, C., Ma, M., Zhao, Z., Chen, X., Li, J.: Pyramid grafting network for one-stage high resolution saliency detection. arXiv preprint arXiv:2204.05041 (2022) 6, 7, 8, 9, 11

17. Qin, X., Dai, H., Hu, X., Fan, D.P., Shao, L., Van Gool, L.: Highly accurate dichotomous image segmentation. In: ECCV, Springer (2022) 38–56 9, 10
18. Freixenet, J., Munoz, X., Raba, D., Martí, J., Cufí, X.: Yet another survey on image segmentation: Region and boundary information integration. In: ECCV, Springer (2002) 408–422 9, 11
19. Cheng, H.K., Chung, J., Tai, Y.W., Tang, C.K.: Cascadepsp: Toward class-agnostic and very high-resolution segmentation via global and local refinement. In: CVPR. (2020) 8890–8899 9, 11
20. Cheng, B., Girshick, R., Dollár, P., Berg, A.C., Kirillov, A.: Boundary iou: Improving object-centric image segmentation evaluation. In: CVPR. (2021) 15334–15342 9, 11
21. Zeng, Y., Zhang, P., Zhang, J., Lin, Z., Lu, H.: Towards high-resolution salient object detection. In: ICCV. (2019) 7234–7243 9, 11
22. Tang, L., Li, B., Zhong, Y., Ding, S., Song, M.: Disentangled high quality salient object detection. In: ICCV. (2021) 3580–3590 9, 11