UNDER REVIEW

RESCIENCE C

Replication / ML Reproducibility Challenge 2020

[Re] Interactive Two-Stream Decoder for Accurate and Fast Saliency Detection

Chun-Wei TAN[1], Yi Ke YUN[1], Yuto ITAYA[1], Wei Yeng SEOW[1], Dick Shern EOW[1], and Takahiro TSUBONO[1]

[1]OPEN8 Singapore Pte. Ltd., 109 North Bridge Road, Singapore 179097

Edited by
(Editor)

Received
01 November 2018

Published
–

DOI
–

Abstract This article describes our attempts to reproduce the work presented in the paper - Interactive Two-Stream Decoder for Accurate and Fast Saliency Detection which published in CVPR2020, as part of the ML Reproducibility Challenge 2020. The work exploited the correlation between saliency and contour information by introducing interactive connections between the saliency and contour streams in the decoder. The interactive connections allow the saliency and contour information from the respective stream to be transmitted across the branch and therefore obliging the network to learn the correlation between saliency and contour information. Our work consists of three parts: (1) evaluation using the provided pre-trained models from the original paper; (2) evaluation using our retrained models; (3) parameters analysis. Despite differences in experimental results, the reproducibility of the studied paper is further ascertained by conducting the statistical test.

1 Introduction

Saliency detection is a mechanism to detecting visually attentive objects or regions in an image. To better discover the visually distinctive objects or regions, the saliency detection methods are expected to understand both the global and local features [1]. The global features are essential to provide high-level semantic information for identifying salient objects while the local features are exploited for features refinement. The investigated paper – Interactive Two-Stream Decoder for Accurate and Fast Saliency Detection (ITSD) [2] explores the correlation between saliency (global) and contour (local) information by introducing interactive connections between the saliency and contour streams in the decoder. An adaptive contour loss is also introduced in the saliency stream to weight more on near boundary pixels of the salient objects, thus improves the network capability to handling the salient object boundary. The method was evaluated on six publicly available saliency datasets and encouraging results were reported.

The remaining of the paper is organized as follows: Section 2 provides a brief review of the studied paper which describes the overall architecture of the ITSD. Section 3 presents our attempts to reproduce the reported results on six benchmark datasets. Statistical analysis is also presented to confirm the reproducibility of work from the investigated paper. Section 4 presents our further analysis on the effect of various training parameters. Lastly, the paper is concluded in Section 5.

2 Summary of the Article

The ITSD employs an encoder-decoder structure as its network architecture as shown in Fig. 1(a). The backbone of the encoder can be either of a pretrained VGG or ResNet. In order to keep the model lightweight, two measures are adopted: (1) the fully-connected

Copyright © 2020 C.-W. TAN et al., released under a Creative Commons Attribution 4.0 International license.
Correspondence should be addressed to Chun-Wei TAN (chunweit@open8.com)
The authors have declared that no competing interests exist.
Code is available at https://github.com/rescience-c/template..

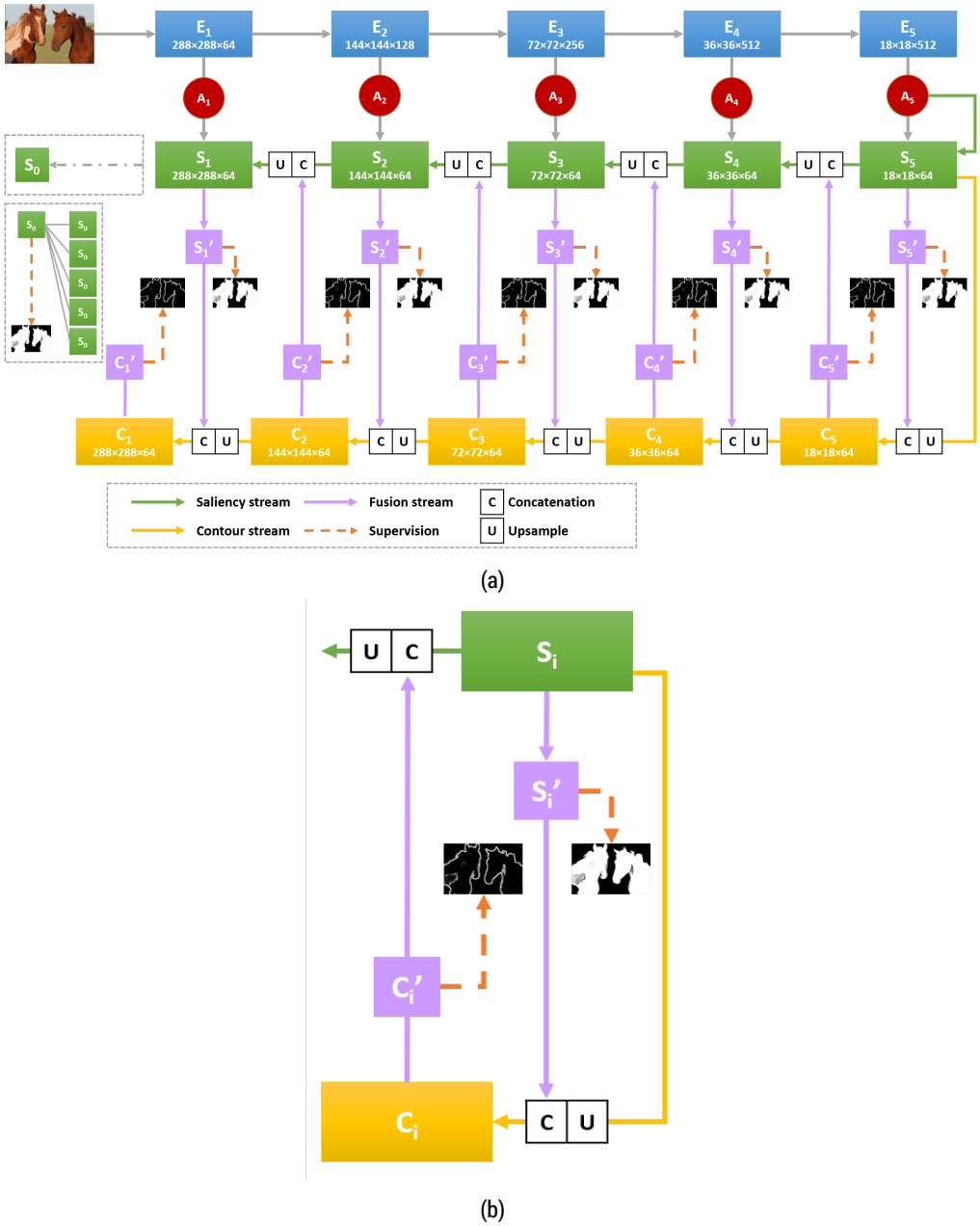


Figure 1. (a) VGG-based encoder-decoder architecture (figure reproduced from the original paper). (b) the FCF module provides interactive connection between saliency and contour streams.

(FC) layers in the standard VGG network is truncated with the aim to reduce the model size while there is no such need for the ResNet-based encoder since it has no FC layer; (2) channel pooling is applied to each of the encoded feature maps. To better recover the saliency map, the decoder of the network employs a two-stream structure, which comprises of five cascaded Feature Correlation Module (FCF) modules. The FCF, as shown in Fig. 1(b), provides interactive connection (fusion stream) which allows the respective saliency and contour information to be transmitted across the stream, and therefore encourages the network to learn the correlation. To generate the final saliency map, all of the intermediate features from the saliency stream are first up-sampled to the input image size followed by concatenation operation.

2.1 Objective Function

The ITSD employs two loss functions respectively for the saliency stream and the contour stream. The contour stream uses a conventional BCE loss function and is given as follows:

$$l_{bce}(x, y) = y \log(x, y) + (1 - y) \log(1 - x) \tag{1}$$

$$L^c(P^c, G^c) = -\frac{1}{n} \sum_{k=0}^{n-1} l_{bce}(p_k^c, g_k^c) \tag{2}$$

where p_k^c and g_k^c respectively denote pixels in the predicted (P^c) and ground truth (G^c) contour maps, while k indicates the index of the n pixels contour map. An adaptive contour loss (ACT) which leverages on the contour information is proposed to provide adaptive weight to the saliency stream, which allows the model to pay more attention to those pixels near to the salient object boundary. The ACT is defined as follows:

$$L^s(P^s, P^c, G^s, G^c) = -\frac{1}{n} \sum_{k=0}^{n-1} (max(p_k^c, g_k^c) * m + 1) l_{bce}(p_k^s, g_k^s) \tag{3}$$

where m is a factor to emphasize hard examples, while p_k^s and g_k^s are k-value in the predicted saliency map (P^s) and ground truth saliency map (G^s), respectively. Given with the Eq.(2) and (3), the total loss of the model can be computed as:

$$L(P^s, P^c, G^s, G^c) = \sum_{i=0}^{5} L^s(P_i^s, P_i^c, G_i^s, G_i^c) + \lambda \sum_{j=1}^{5} L^c(P_j^c, G_j^c) \tag{4}$$

where λ is a weighting factor to balance between contour and saliency losses.

3 Reproduction of Work

The authors have made the source code associated with the paper publicly available on GitHub as well as links to download their pretrained models [3]. Six salient object benchmark datasets as summarized in Table 1 were employed in our reproduction attempts for evaluation purpose. Two performance metrics, *i.e.*, F_β-measure and mean absolute error (MAE), which respectively given in Eq. (5) and Eq. (6), were employed to evaluate the performance of the models.

$$F_\beta = \frac{(1 + \beta^2) * Precision * Recall}{\beta^2 * Precision + Recall} \tag{5}$$

$$MAE = \frac{1}{n} \sum_{i=1}^{n} \mid x_i - y_i \mid \tag{6}$$

Our reproduction of work is mainly consisting of two parts: (1) we performed the evaluation by using the pretrained models as made available by the authors; (2) we performed the evaluation by using the retrained ITSD models. Both the evaluations were performed on the six benchmark datasets. Table 2 and 3 summarize our evaluation results as well as the comparison against the reported results in the article. The evaluation results in terms of F_β were close to the reported values, with the neglectable differences of up to 1%. However, the degree of differences in terms of MAE was quite noticeable, with the difference as high as 11.4% was observed. After discussion with the authors, there could be several factors that may lead to such discrepancy. One of the possible causes may be due to differences in environment settings, such as versions of some Python packages. On the other hand, the reported results represented the best scores respectively for each of the dataset on different metrics, as the evaluation was done along the training process. Therefore, the pretrained models/final models may not be the representation of the optimal models to perform well on all six benchmark datasets, which in turn led to such discrepancy in our evaluation. In order to confirm our assumption, we performed experiments to capture the best epoch results along the training, as given in the Table 4 and 5. The best epoch results (in parentheses) were observed to achieve either similar or better performance as compared to the reported results.

Datasets	Number of Images	Remarks
DUTS-TR	10553	Training
DUTS-TE	5019	Testing
SOD	300	
PASCAL-S	850	
ECSSD	1000	
HKU-IS	4447	
DUTS-OMRON	5168	

Table 1. Training and testing datasets

Datasets	F_β			MAE		
	Our results	Reported	Difference (%)	Our results	Reported	Difference (%)
ECSSD	0.946	0.947	0.11	0.034	0.035	2.86
DUTS-TE	0.885	0.883	0.23	0.039	0.041	4.88
DUT-OMRON	0.819	0.824	0.61	0.059	0.061	3.28
PASCAL-S	0.876	0.871	0.57	0.064	0.071	9.86
SOD	0.872	0.880	0.91	0.092	0.095	3.16
HKU-IS	0.936	0.934	0.21	0.029	0.031	6.45

Table 2. Evaluation results using the pretrained ResNet-based model

Datasets	F_β			MAE		
	Our results	Reported	Difference (%)	Our results	Reported	Difference (%)
ECSSD	0.939	0.939	0.00	0.039	0.040	2.50
DUTS-TE	0.880	0.877	0.34	0.041	0.042	2.38
DUT-OMRON	0.805	0.813	0.98	0.059	0.063	6.35
PASCAL-S	0.871	0.871	0.00	0.067	0.074	9.46
SOD	0.860	0.869	1.04	0.106	0.100	6.00
HKU-IS	0.931	0.927	0.43	0.031	0.035	11.43

Table 3. Evaluation results using the pretrained VGG-based model

To retrain the ITSD models, we followed the training protocol as described in the original paper. The VGG and ResNet, which pretrained on ImageNet dataset, were employed as the backbone of the model. Data augmentation such as random flipping, random cropping and scaling was applied on the DUTS-TR dataset with the aim to increase variability in the dataset. The models were trained on two GTX2080TI GPUs using SGD at the learning rate of 0.01 for the first 20k iterations and decayed by a factor of 0.1 for the

Datasets	F_β			MAE		
	Our results (best epoch result)	Reported	Difference (%)	Our results (best epoch result)	Reported	Difference (%)
ECSSD	0.945 (0.947)	0.947	0.21	0.035 (0.034)	0.035	0.00
DUTS-TE	0.883 (0.883)	0.883	0.00	0.040 (0.039)	0.041	2.44
DUT-OMRON	0.823 (0.824)	0.824	0.12	0.057 (0.055)	0.061	6.56
PASCAL-S	0.870 (0.872)	0.871	0.11	0.068 (0.066)	0.071	4.23
SOD	0.859 (0.875)	0.880	2.39	0.099 (0.091)	0.095	4.21
HKU-IS	0.935 (0.935)	0.934	0.11	0.029 (0.029)	0.031	6.45

Table 4. Evaluation results using the retrained ResNet-based model

Datasets	F_β			MAE		
	Our results (best epoch result)	Reported	Difference (%)	Our results (best epoch result)	Reported	Difference (%)
ECSSD	0.937 (0.941)	0.939	0.21	0.039 (0.037)	0.040	2.50
DUTS-TE	0.874 (0.879)	0.877	0.34	0.042 (0.041)	0.042	0.00
DUT-OMRON	0.801 (0.804)	0.813	1.48	0.062 (0.059)	0.063	1.59
PASCAL-S	0.869 (0.875)	0.871	0.23	0.068 (0.066)	0.074	8.11
SOD	0.863 (0.866)	0.869	0.69	0.103 (0.093)	0.100	3.00
HKU-IS	0.929 (0.932)	0.927	0.22	0.032 (0.030)	0.035	8.57

Table 5. Evaluation results using the retrained VGG-based model

remaining 5k iterations. The other training parameters, *i.e.*, $\lambda = 1$, $m = 4$, and batch size $= 8$ were strictly followed the suggested values in the paper. The evaluation results from the ResNet-based and VGG-based models are summarized in the Table 4 and 5, respectively. It can be observed that the degree of differences can be as high as 2.4% in F_β and 8.6% in MAE. In order to quantitatively justifying the reproducibility of the results, we carried out statistical analysis using paired t-test at significance level of 0.05 to verify our null hypothesis if there was no significant difference between our evaluation results and the reported results in the article. The paired t-test statistics value can be calculated as follows:

$$t = \frac{\bar{d}}{s/\sqrt{n_d}} \tag{7}$$

where $\bar{d}$ is the mean differences, s is the standard deviation and n_d is the number of observations, *i.e.*, number of the employed benchmark datasets. Table 6 summarizes the p-value obtained from the calculated t-score with all the p-value greater than the significance level, which implies failing to reject the null hypothesis. In other words, the effort presented in the article is reproducible, as evidenced by the statistical analysis.

	p-value	
	ResNet-based	VGG-based
F_β	0.2956	0.1050
MAE	0.4261	0.3276

Table 6. Statistical test at significance level of 0.05

4 Parameters Analysis

In this section, we provide further analysis on various parameters that had not been elaborated in the original paper to provide more insights which may be deemed benefit for those working in the field. All the experiments conducted in this section employed the ResNet-based ITSD model, with all the training parameters kept unchanged (see Section 3) unless specified otherwise.

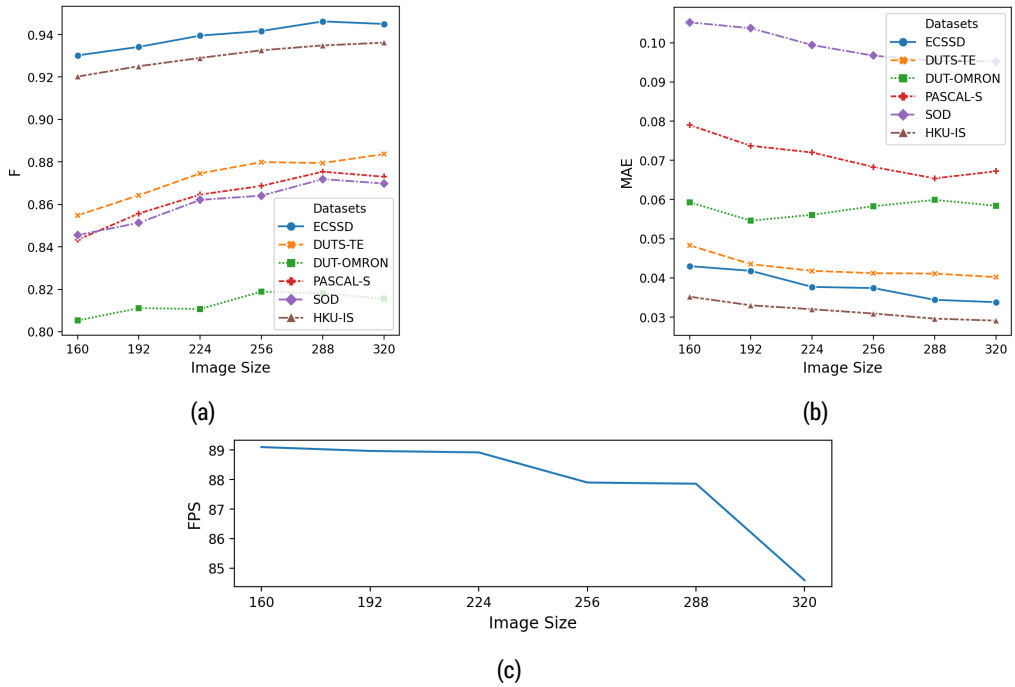


Figure 2. Performance of different input image sizes and inference speed. (a) F_β. (b) MAE (c) inference speed

4.1 Image sizes and inference speed

We have seen different input image sizes were employed for the saliency detection, such as [4, 5, 1]. Ref. [2] arbitrary chosen an input image size of 288×288 which motivates us to determine the impact of varying input image sizes on the performance of the ITSD model. We performed the evaluation using the models which trained with input image sizes of $160 \times 160, 192 \times 192, 224 \times 224, 256 \times 256, 288 \times 288, 320 \times 320$. Generally, the performance was better by using larger input image size, as can be observed from the evaluation results as shown in Fig. 2(a) and (b). In addition, we also provide the evaluation of inference speed with different input image sizes on a single GPU, as can be observed from Fig. 2(c). The default input image size 288×288 reported the inference speed ran at around 88 FPS while the inference speed for the input image size of 160×160 achieved 89 FPS, at the expense of the saliency performance. The input image size of 288×288 emerges to be a better choice which balances the saliency performance and the inference speed.

4.2 Hard examples factor m

The hard examples factor m in Eq. 3 serves to emphasize the boundary pixels so the model can better handling the boundary of the salient regions. However, the effect of such parameter had not been discussed in the original paper. Fig. 3 shows our evaluation results of using various models respectively trained with $m = \{1, 2, 3, 4, 5\}$. On average, some marginal gains were observed in both metrics with the hard examples factor $m = 2$.

4.3 Dilation and erosion

The contour map is obtained by computing the difference between the saliency maps produced by two morphological operations, *i.e.*, dilation and erosion. The resultant

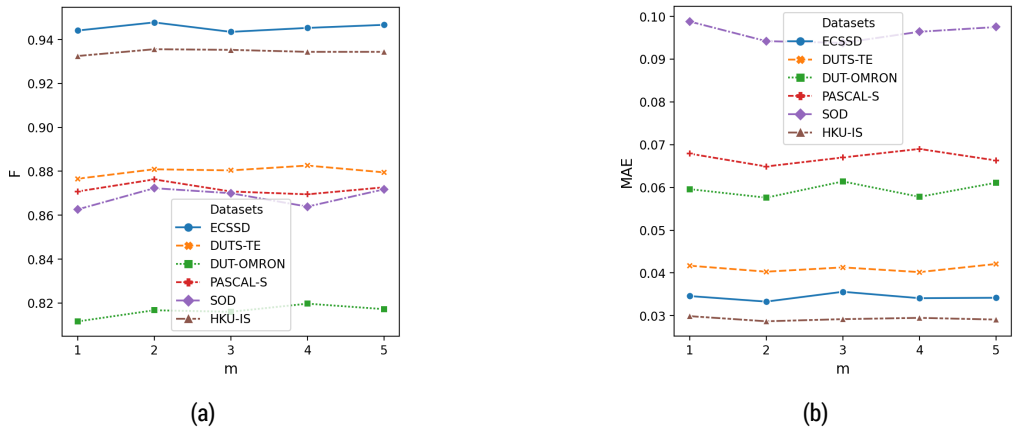


Figure 3. Effect of different hard example factor (a) F_β. (b) MAE

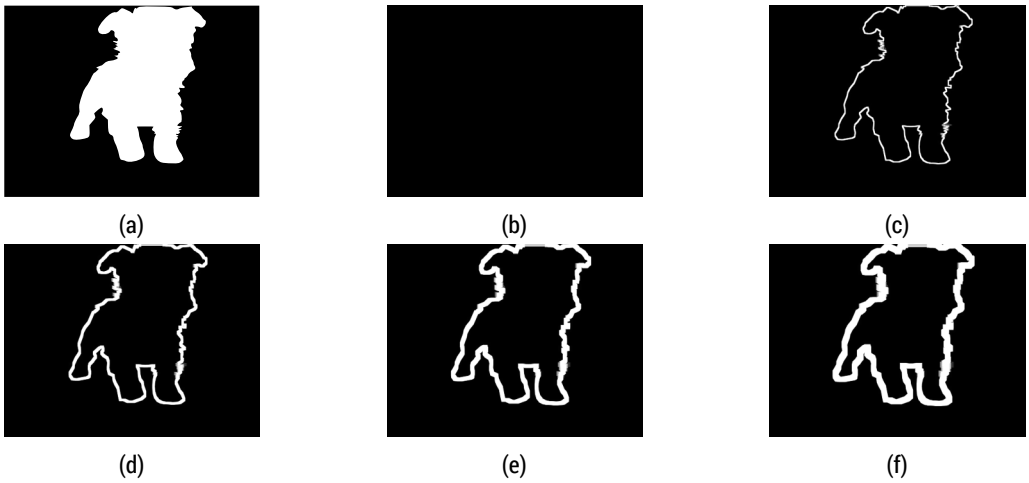


Figure 4. Contour maps generated using different morphological kernels (a) mask, (b) 1×1, (c) 3×3, (d) 5×5, (e) 7×7, (f) 9×9

from such operation is a contour map with the border width determined by the morphological kernel size, which in turn affecting how the model learn about the hard example pixels near the salient object boundary. In the original authors' implementation, morphological kernel size of 5×5 was used. In order to study the effect of different morphological kernel sizes, we considered $1 \times 1, 3 \times 3, 5 \times 5, 7 \times 7, 9 \times 9$ in our experiments. Fig 4 shows samples of contour maps generated using different morphological kernel sizes. Note that no contour information is produced by using the morphological kernel size of 1×1 and therefore the Eq. 3 will be solely depend on the P^c to weight the saliency loss function. As can be observed from Fig. 5, the morphological kernel size of 3×3 can generally provide better performance than the default kernel size, but such improvement are only marginal.

4.4 Lambda, λ

The λ in Eq. 4 serves to balance between the saliency and contour losses, which was set to $\lambda = 1$ in the original paper. In other words, we can determine how much emphasis to be placed on the contour loss by adjusting the λ. In our experiments, we performed evaluation for a range of values, *i.e.,* $\lambda = [0.0, 1.6]$, with a step size of 0.2. Based on the average results from our evaluation, the $\lambda = 0.6$ had shown to achieve better perfor-

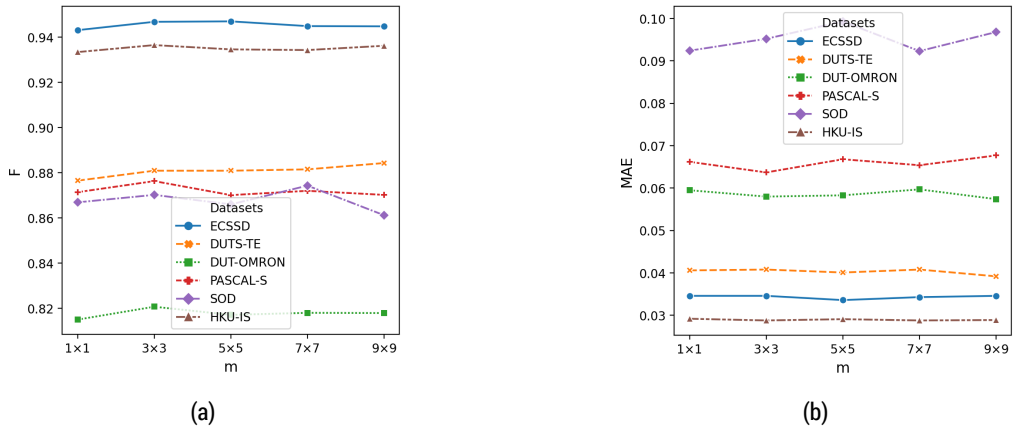


Figure 5. Effect of morphological kernel size (a) F_β. (b) MAE

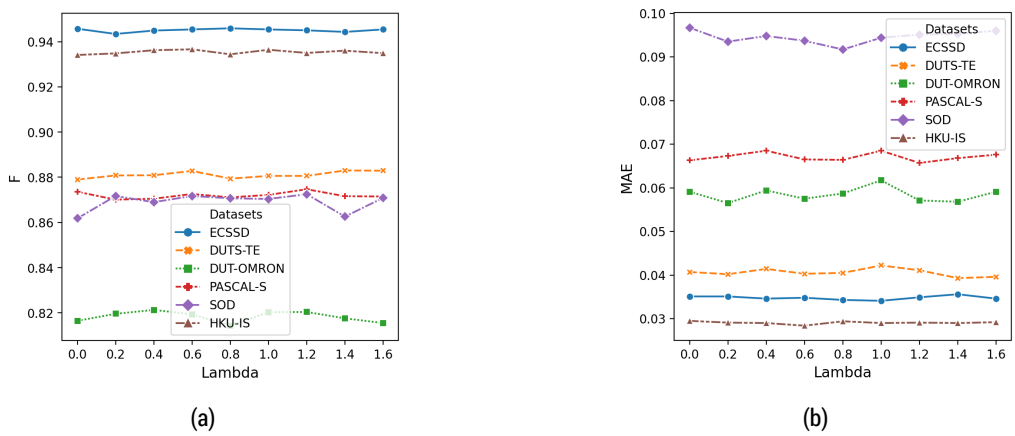


Figure 6. Effect of lambda (a) F_β. (b) MAE

mance, which suggested that the model is biased towards the saliency information as such information is more crucial, while complemented by the contour information for further performance gain. The $\lambda = 0$ totally suppresses the contour loss, which can be considered as a special case of Eq. 4, where only saliency information is considered.

4.5 Optimized vs. default parameters

	Optimized	Default
Input image size	288×288	288×288
Morphological kernel size	3×3	5×5
Hard examples factor, m	2	4
Lambda, λ	0.6	1

Table 7. Optimized vs. default parameters

Sections 4.1 - 4.4 show our evaluations on the various parameters of the ITSD model, and each individual parameter has been shown to induce certain degree of improvement on the average performance metrics. As such, we carried out experiment by using the optimized parameters, as summarized in Table 7. Fig. 7 presents the average results of the comparison between the models trained with optimized and the default parameters, which only marginal gain was observed. Our primary intention to carry out the

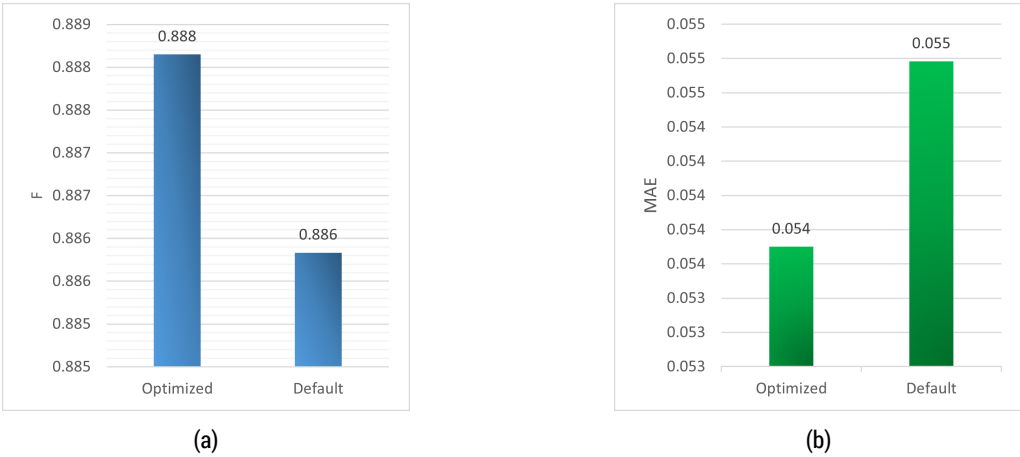


Figure 7. Optimized vs. default parameters. (a) F_β. (b) MAE

experiments in Sections 4.1 - 4.4 was to study the implication of each parameter towards the saliency detection performance. Therefore, each of the experiment conducted only evaluated the parameter of concerned while other parameters were kept unchanged in order to better facilitate the analysis. One should consider a proper parameter optimization approach such as grid search to exhaustively searches through the parameter spaces to better estimate the best parameters' combination.

4.6 Constrained lambda

Although the free parameter, *i.e.,* λ in Eq. 4 was analyzed in Section 4.4, it is still uncertain which of the losses contribute more towards the optimization of the network. In order to further analyze the importance of each loss term, we imposed constraint on the Eq. 4 by considering a weighted sum rule, as given as follows:

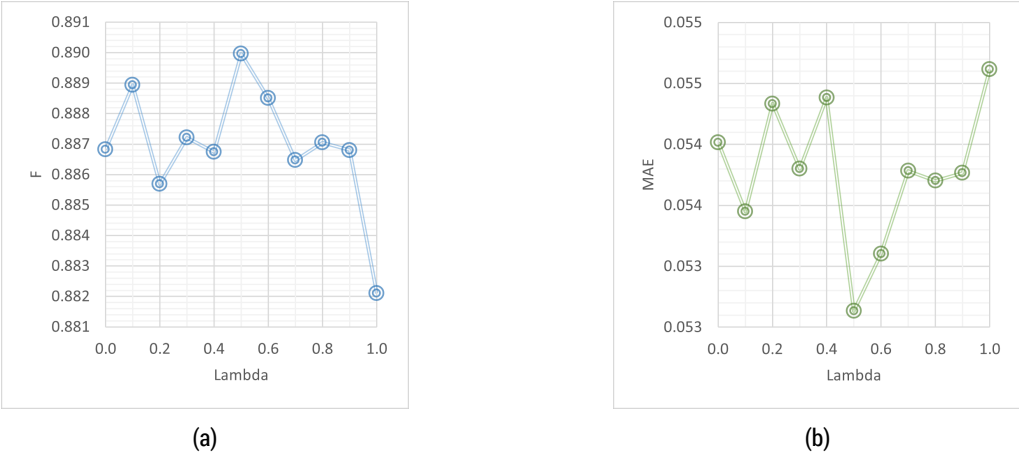
$$L(P^s, P^c, G^s, G^c) = \sum_{i=0}^{5}(1-\lambda)L^s(P_i^s, P_i^c, G_i^s, G_i^c) + \lambda \sum_{j=1}^{5} L^c(P_j^c, G_j^c) \quad \text{s.t. } 0 \leq \lambda \leq 1 \tag{8}$$

By constraining the lambda would enable us to analyze the correlation between the saliency and contour losses, as can be observed from average performance metrics in Fig. 8. The best saliency detection performance was achieved when the $\lambda = 0.5$, which suggested that both the saliency and contour losses contributed equally and complemented each other. While the worst performance was reported when the $\lambda = 1.0$, which implied that the contour loss alone appeared to be suboptimal for optimizing the model.

5 Discussion and conclusion

We had some issues to get the published code working initially but those issues were solved after some discussion with the authors. One of the major issues was the inconsistent use of the name of the parameter keys in the given PyTorch models, which resulted in failing to load the model weight. The issue can be solved by updating the name of the parameter keys to be compatible with the defined models.

Despite of all the challenges we faced during the work reproduction, we eventually managed to reproduce the work of the investigated paper, *i.e.,* ITSD. Our reproduction attempts were mainly comprised of two parts: (1) reproduction using the pretrained mod-



(a) (b)

Figure 8. Constrained lambda. (a) F_β. (b) MAE

els as shared by the authors; (2) reproduction by training the models. Our attempt using the pretrained models was of no success to reproduce the exact results as reported in the original paper, as can be observed in Table 2 and 3. After seeking clarification from the authors, one of the possible causes of such discrepancy may be due to the environment settings. Another possible reason may be due to the best epoch results were reported in the paper, while the shared models represented the final training artifacts. Our assumption was evidenced by carrying out the experiments to capture the best epoch results, which we successfully obtained either similar or better performance as compared to the reported results. Our second reproduction attempt was to perform the retraining of the ITSD models. Despite differences in the experimental results, we carried out statistical test, *i.e.,* paired t-test, to further confirmed the reproducibility of the presented work in the studied paper. Further to our reproduction attempts, we had also carried out experiments to perform evaluation on the various parameters which had not been discussed in the original paper, which we hope it can be beneficial to those working in the same direction.

6 Acknowledgements

We would like to take this opportunity to thank the authors for the insightful discussion and guidance to make this work possible.

References

1. X. Qin, Z. Zhang, C. Huang, C. Gao, M. Dehghan, and M. Jagersand. "BASNet: Boundary-Aware Salient Object Detection." In: **2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)**. 2019, pp. 7471–7481. DOI: 10.1109/CVPR.2019.00766.
2. H. Zhou, X. Xie, J. .-.-H. Lai, Z. Chen, and L. Yang. "Interactive Two-Stream Decoder for Accurate and Fast Saliency Detection." In: **2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)**. 2020, pp. 9138–9147. DOI: 10.1109/CVPR42600.2020.00916.
3. H. Zhou, X. Xie, J. .-.-H. Lai, Z. Chen, and L. Yang. **ITSD-pytorch**. 2020. URL: https://github.com/moothes/ITSD-pytorch.
4. J.-J. Liu, Q. Hou, M.-M. Cheng, J. Feng, and J. Jiang. "A Simple Pooling-Based Design for Real-Time Salient Object Detection." In: **IEEE Conference on Computer Vision and Pattern Recognition**. 2019.
5. J. Zhao, J. Liu, D. Fan, Y. Cao, J. Yang, and M. Cheng. "EGNet: Edge Guidance Network for Salient Object Detection." In: **2019 IEEE/CVF International Conference on Computer Vision (ICCV)**. 2019, pp. 8778–8787. DOI: 10.1109/ICCV.2019.00887.