Trash or Treasure? An Interactive Dual-Stream Strategy for Single Image Reflection Separation

Anonymous Author(s)
Affiliation
Address
email

Abstract

Single image reflection separation (SIRS), as a representative blind source separation task, aims to recover two layers, *i.e.* transmission and reflection, from one mixed observation, which is challenging due to the highly ill-posed nature. Existing deep learning based solutions typically restore the target layers individually, or with some concerns at the end of the output, barely taking into account the interaction across the two streams/branches. In order to utilize information more efficiently, this work presents a general yet simple interactive strategy, namely *your trash is my treasure* (YTMT), for constructing dual-stream decomposition networks. To be specific, we explicitly enforce the two streams to communicate with each other block-wisely. Inspired by the additive property between the two components, the interactive path can be easily built via transferring, instead of discarding, deactivated information by the ReLU rectifier from one stream to the other. Both ablation studies and experimental results on widely-used SIRS datasets are conducted to demonstrate the efficacy of YTMT, and reveal its superiority over other state-of-the-art alternatives. Moreover, we show that such a simple strategy has much potential to be applied to other tasks like blind denoising. The implementation is quite simple and our code will be made publicly available.

1 Introduction

Blind source separation, a long-standing problem in signal processing, aims to recover multiple intrinsic components from their mixture, the difficulty of which comes from its ill-posedness, *i.e.*, without extra information, there is an infinite number of feasible decompositions. Particularly in computer vision, image reflection separation (IRS) is a representative scenario that often occurs when taking pictures through a transparent medium such as glass. In such cases, the captured images will contain both the scene transmitted through the medium (transmission) and reflection. On the one hand, reflections are annoying for high-quality imaging, and may interfere with the performance of most, if not all, of classic and contemporary vision oriented algorithms such as object detection and segmentation. On the other hand, one may also want to see what happens in the reflection. Hence, developing effective transmission-reflection decomposition techniques is desired.

Formally, the captured superimposed image I can be typically modeled as a linear combination of a transmission layer T and a reflection R, *i.e.* $I = T + R$.[1] Over last decades, a large number of schemes have been devised to solve the decomposition problem. Various statistical priors and regularizers have been proposed to mitigate the ill-posed dilemma, while diverse deep networks have

[1]Many problems follow the same additive model, such as denoising ($I = B + N$, where B and N denote clean image and noise, respectively), and intrinsic image decomposition ($\log I = \log A + \log S$, where A and S stand for albedo and shading, respectively.) The proposed strategy can be potentially applied to all these tasks, but due to page limit, we concentrate on the task of SIRS to verify primary claims in this paper.

Submitted to 35th Conference on Neural Information Processing Systems (NeurIPS 2021). Do not distribute.

been recently built for the sake of performance improvement, please see Sec. 2 for details. However, in the literature, the additive property, say $I = T + R$, has been hardly investigated, except for acting as a reconstruction constraint.

Let us consider that, for any estimation pair $\hat{T}$ and $\hat{R}$ satisfying the additive property, there always exists an error/residual Q subject to $I = \hat{T} + \hat{R} = (T + Q) + (R - Q)$. Once Q is somehow obtained, the only thing needed to do is subtracting it from $\hat{T}$, and adding it into $\hat{R}$ instead of simply discarding. Under the circumstances, no information is trash, only misplaced. To be symmetric, we rewrite Q as $Q := Q_T - Q_R$, yielding $T = \hat{T} - Q_T + Q_R$ and $R = \hat{R} - Q_R + Q_T$. In other words, the two targets including $\hat{T}$ and $\hat{R}$ can gain from each other by exchanging, rather than discarding, their respective "trash" factors Q_T and Q_R. Driven by the above fact, a question naturally arises: *Can such an interaction/exchange be applied to intermediate deep features of dual-stream networks?*

Contributions. This paper answers the above question by designing a general interactive dual-stream/branch strategy, namely *your trash is my treasure* (YTMT). An obstacle to realizing YTMT was how to determine exchanging information. Intuitively, activation functions are competent for the job, which are developed to select (activate) a part of features from inputs. In this work, we adopt the ReLU that is arguably the most representative and widely-used activation manner, while others could be also qualified like [4, 24, 12, 2]. Please notice that, instead of simply discarding the deactivated features (trash) of the one stream, we alternatively deliver them to the other stream as compensation (treasure). By doing so, there are two main merits: 1) the information losing and dead ReLU problems can be consequently mitigated, and 2) the decreasing speed in training error can be significantly accelerated. The implementation is quite simple and flexible. We provide two optional YTMT blocks as examples and apply them on both plain and UNet architectures to verify the primary claims. Both ablation studies and experimental results on widely-used SIRS datasets are conducted to demonstrate the efficacy of YTMT, and reveal its superiority over other state-of-the-art alternatives. Moreover, we also apply the YTMT strategy on the blind denoising task to reveal its applicability.

2 Related Work

Over the past decades, much attention to resolving the image reflection separation problem has been drawn from the community. From the perspective of the required input amount, existing methods can be divided into two classes, *i.e.*, multi-image based and single image based methods.

Multiple Image Reflection Separation (MIRS). Utilizing multiple images [7, 28, 33, 27, 1, 8, 31, 22, 9, 29, 39, 32, 40, 10] used to be a prevalent way to cope with the task, as more information complementary to single superimposed images, like varied conditions and relative motions, can be explored from a sequence of images to accomplish the task. For instance, Agrawal *et al.* [1] use flash and no-flash image pairs to remove reflections and highlights, while Szeliski *et al.* [33] employ focused and defocused pairs. A variety of approaches [7, 27, 8, 22, 33, 39, 40] have been proposed to seek relationships between different images, among which Farid and Adelson [7] employ independent component analysis to reduce reflections and lighting. Gai *et al.* [8] alternatively develop an algorithm to estimate layer motions and linear mixing coefficients for the recovery. Li and Brown [22] adopt the SIFT-flow to align the images, while Xue *et al.* [39] enforce a heavy tailed distribution on the obstruction and background components to penalize the overlapped gradients, making the two layers independent. Moreover, Yang *et al.* [40] introduce a generalized double-layer brightness consistency constraint to connect optical flow estimation and layer separation. *Despite the satisfactory performance of MIRS methods, the need for specified shooting conditions and/or professional tools heavily limits their applicability.*

Single Image Reflection Separation (SIRS). In practice, single image based schemes are more attractive. No doubt that, compared with those MIRS techniques, single image based ones are also more challenging due to less information available, demanding extra priors to regularize the solution. To mitigate the ill-posedness, Levin *et al.* [19, 18] favor decompositions that have fewer edges and corners by imposing sparse gradients on the layers, in the same spirit as [39]. Levin and Weiss [17] allow users to manually annotate some dominant edges for the layers as explicit constraints for the problem, which requires careful human efforts to obtain favorable results. Li and Brown [23] assume one layer is smoother than the other, and penalize differently on the two layers in the gradient domain to split the reflection and transmission. *Despite a progress made toward addressing the problem, the assumption and requirement could be frequently violated in real situations. Besides, these methods all rely on handcrafted features that may considerably restrict their performance.*

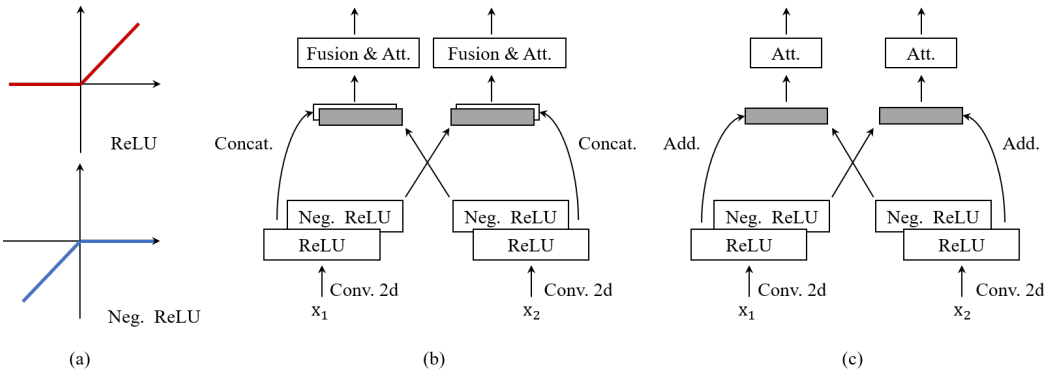


Figure 1: The functional behaviors of ReLU and negative ReLU are plotted in (a). Two YTMT block options are given in (b) and (c). To fuse features from the normal and YTMT connections, the first option uses channel concatenation and 1×1 convolutions, while the second simply uses feature addition. Pixel and channel attention mechanism (denoted by Att.) is utilized in both (b) and (c).

With the emergence of deep neural networks, learning based methods [6, 43, 41, 37, 20, 38] have shown their power and become dominant for the task. Concretely, CEILNet [6] comprises of two stages, similarly to [17, 36]. An edge map is estimated in the first stage, which performs as the guidance for the second stage to produce the final transmission layer. However, the network is hard to capture high-level information, thus having trouble guaranteeing the perceptual quality. Zhang *et al.* [43] further take the perception cue into consideration by coordinating hyper-column features [11], the perceptual and adversarial losses. The exclusivity loss (see also the gradient production penalization in [39]) is designed for assuring gradients to be exclusive between the two components. This manner is also adopted by [37]. The main drawback is the insufficient capacity in processing cases with large transmission-reflection overlapped regions. Recently, ERRNet [37] was proposed, which enlarges the receptive field by introducing channel attention and pyramid pooling modules. It does not explicitly predict the reflection layer, which directly discards possible benefit from the supervision of reflection. IBCLN [20] proposes an iterative boost convolutional LSTM network to progressively split the two layers. However, such an iterative scheme slows down the training and predicting procedure. The errors will also be accumulated due to the high dependency on the outputs of previous stages. Though the learning based strategies have further stepped forward in single image reflection separation compared with the traditional methods, they either restore the target layers individually, or together with some concerns dealing with outputs (*e.g.*, enforcing the linear combination constraint and/or concatenating them together as input). In other words, *they barely take into account the interaction across the two streams/branches, which is key to the target task and also other dual-stream decomposition tasks.* This study is mainly to demonstrate the effectiveness of such an interaction consideration.

3 Deep Interactive Dual-Stream Learning for SIRS

3.1 YTMT Strategy

In this part, the proposed deep interactive dual-stream strategy will be detailed by centering around the concept that your trash is my treasure, which would be beneficial to a variety of two-component decomposition tasks using dual-branch networks. Prior works [21, 35] have shown evidence on the effectiveness of passing information between two branches though, our YTMT strategy performs in a novel and principled way. We first give the definition of the **negative ReLU** function as follows:

$$\mathrm{ReLU}^{-}(\mathbf{x}) := \mathbf{x} - \mathrm{ReLU}(\mathbf{x}) = \min(\mathbf{x}, 0), \tag{1}$$

where $\mathrm{ReLU}(\mathbf{x}) := \max(\mathbf{x}, 0)$. By the negative ReLU, the deactivated features can be easily retained. Figure 1 (a) exhibits the behaviors of the ReLU and negative ReLU.

Here, let $\mathbf{x}^0$ be the input to the first layer of the network, and $\tilde{\mathbf{x}}_i^l$ ($i \in \{1, 2\}$ for two branches) denotes the feature obtained by the i-th branch after l stacked layers, *i.e.* $\tilde{\mathbf{x}}_i^l := \mathcal{H}_i^l(\mathbf{x}^0)$. The inputs to the $(l+1)$-th layer are as follows:

$$\mathbf{x}_1^l := \mathrm{ReLU}(\tilde{\mathbf{x}}_1^l) \oplus \mathrm{ReLU}^{-}(\tilde{\mathbf{x}}_2^l); \quad \mathbf{x}_2^l := \mathrm{ReLU}(\tilde{\mathbf{x}}_2^l) \oplus \mathrm{ReLU}^{-}(\tilde{\mathbf{x}}_1^l), \tag{2}$$

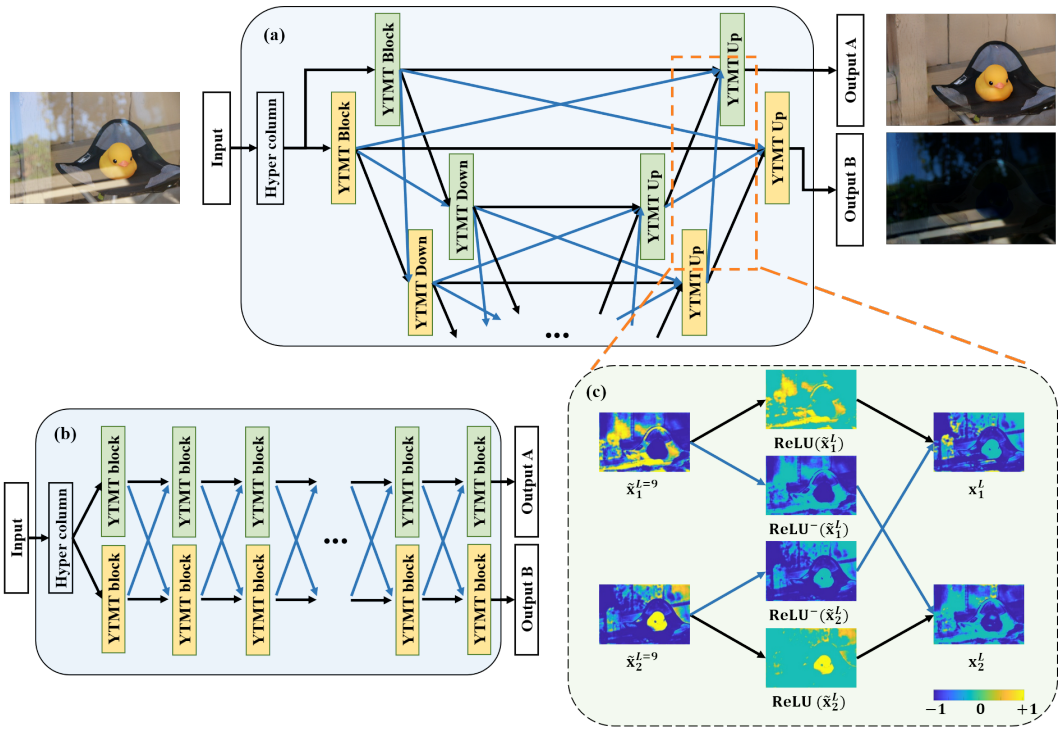


Figure 2: Illustration of the YTMT networks. (a) and (b) offer the YTMT versions modified on the U-shape and plain network architectures, respectively. The input is first augmented by hyper column [43] and then fed into the YTMT blocks. (c) shows a visualized example of producing $\mathbf{x}_1^L$ and $\mathbf{x}_2^L$ from $\tilde{\mathbf{x}}_1^L$ and $\tilde{\mathbf{x}}_2^L$. For better view, all the features are normalized by softmax.

where $\oplus$ can be either the concatenation operation or the addition between features activated by the ReLU function (called **normal connection**) and those by the ReLU^- (**YTMT connection**), as shown in Fig. 1 (b) and (c). As can be obtained from Eq. (2), the amount of information in $\mathbf{x}_1^l$ and $\mathbf{x}_2^l$ is equivalent to that in $\tilde{\mathbf{x}}_1^l$ and $\tilde{\mathbf{x}}_2^l$. This property guarantees no information flowing away from the interaction, which substantially avoids the problems of vanishing/exploding gradients and dead ReLU. Figure 2 (c) depicts a visual example of producing $\mathbf{x}_1^L$ and $\mathbf{x}_2^L$ from $\tilde{\mathbf{x}}_1^L$ and $\tilde{\mathbf{x}}_2^L$ (in this case, $L = 9$). In addition, this strategy can significantly speed up the decreasing of training error similarly to the ResNet design [13], which will be empirically validated in Sec. 4.

We again emphasize that the proposed YTMT strategy is general and flexible, which can be implemented in various forms according to different demands. Figure 1 provides two YTMT block options. The one in (b) fuses features from the normal and YTMT connections by channel concatenation and 1×1 convolutions, while the second in (c) simply employs feature addition. In addition, pixel and channel attention mechanism is introduced in both (b) and (c) to select and re-weight the merged features (see [25] for details). Moreover, the YTMT blocks can be applied to most, if not all, of dual-stream backbones by simple modification. In Fig. 2 (a) and (b), two commonly-used architectures, *i.e.* U-shaped [26] and plain [42] networks, are present. We will shortly show how to construct these YTMT based networks, demonstrate their improvement over the backbones and the superior performance on specific applications over other competitors.

3.2 YTMT based Design for SIRS

As for the task of SIRS, we adopt the U-shaped network as backbone, which can be readily implemented by replacing the blocks in the UNet architecture with proposed YTMT block options. In addition, there are YTMT skip connections along the normal ones. As for the forward process, the input images are first augmented by the hyper column [43], gaining 1475 channels in total, and then delivered into the YTMT dual streams. The head YTMT blocks in the network firstly map the number of channels to 64 via 1×1 convolutions and then employ the YTMT strategy to derive YTMT and normal connections. In the subsequent YTMT Down blocks, the streams are max-pooled with the

stride of 2, and the number of channels are then doubled. The information for the other stream are also fused in the block. Note that YTMT blocks can hold the number of channels invariant after the feature fusion of the two streams, the form of which can be one of the options in Fig. 1 (b) and (c), where the fusion operation in (b) is implemented by 1×1 convolutions to halve the number of concatenated channels and (c) the feature addition. Therefore, we can follow the UNet style to structure the network, where 1024 channels are at the most. The combination of bilinear upsampling and YTMT block, namely YTMT Up is then utilized to upsample the feature maps and reconstruct the layer separations block-by-block. With respect to the skip connections between the down paths and up paths, the YTMT blocks firstly fuse information from the other stream and then the concatenations of the UNet are used. The separations are applied Sigmoid rectifiers before the final outputs for SIRS. After training the first stage to converge, we can choose to then initialize a second stage with parameters from the first one, and with the first stage frozen, the outputs are exploited as the input to train the second stage until it also converges. The information of the transmission and reflection pairs passing through the second stage will further interact. With consideration of YTMT block options ("C" for channel concatenation and "A" for feature addition) and using two-stage or not ("S" for single-stage and "T" for two-stage), YTMT based design for SIRS has several variants, including YTMT-{UCS, UAS, UCT, UAT}, based on the choose of model architecture ("U" for UNet here). More detailed network configuration can be found in the supplementary material. The performance difference between these YTMT variants will be studied in Sec. 4.3.

The objective function considered for SIRS consists of reconstruction loss, perceptual loss, exclusion loss, and adversarial loss. In what follows, each term is explained.

Reconstruction loss. Preceding methods have revealed that edges are essential for a valid separation [17, 39, 6, 35]. To endow our model with awareness of the gradients, we follow [6, 37] to penalize the gradient difference between predictions and targets, along with the MSE via:

$$\mathcal{L}_{rec} := a\|\hat{T} - T\|_2^2 + b\|\hat{R} - R\|_2^2 + c\|\nabla\hat{T} - \nabla T\|_1 + d\|\hat{T} + \hat{R} - I\|_1, \tag{3}$$

where $\|\cdot\|_1$ stands for the ℓ_1 norm, and $\|\cdot\|_2$ the ℓ_2 norm. In addition, we empirically set $a = 0.3$, $b = 0.9$, $c = 0.6$, $d = 0.2$ in all of our experiments. Since reflections are usually weak, the penalization on $\nabla\hat{R} - \nabla R$ is omitted for stable training.

Perceptual Loss. The perceptual loss [14] assists models in achieving high perceptual quality. We minimize the ℓ_1 difference between the features of predicted components and those of ground-truths at layers 'conv2_2', 'conv3_2', 'conv4_2', and 'conv5_2' of a VGG-19 model [30] pretrained on the ImageNet dataset [3]. Denoting the features of the input at layer i as $\phi_i(\cdot)$, we have the following:

$$\mathcal{L}_{per} := \sum_j \omega_j \|\phi_j(T) - \phi_j(\hat{T})\|_1 + \sum_j \omega_j \|\phi_j(R) - \phi_j(\hat{R})\|_1, \tag{4}$$

where ω_js balance the weights of different layers, $\{0.38, 0.21, 0.27, 0.18, 6.67\}$ as default.

Exclusion Loss. Prior work [39, 43] shows that enforcing the exclusivity on the two components in the gradient domain is beneficial to separation tasks, which is defined as

$$\mathcal{L}_{exc} := \frac{1}{N}\sum_{n=0}^{N-1} \|\Psi(\hat{T}^{\downarrow n}, \hat{R}^{\downarrow n})\|_2^2 \quad \text{with} \quad \Psi(\hat{T}, \hat{R}) := \tanh\left(\lambda_T|\nabla\hat{T}|\right) \circ \tanh\left(\lambda_R|\nabla\hat{R}|\right), \tag{5}$$

where $\hat{T}^{\downarrow n}$ and $\hat{R}^{\downarrow n}$ are $\hat{T}$ and $\hat{R}$ down-sampled by 2^n times. In addition, λ_T and λ_R are normalization factors and $\circ$ is element-wise multiplication.

Adversarial Loss. The adversarial loss regulates the SIRS solution on the real-world image manifold. With the adversarial training of the main network $\mathcal{G}$ and a discriminator $\mathcal{D}$, they seek the equilibrium by optimizing the adversarial loss $\mathcal{L}_{adv}$ [15], as follows:

$$\mathcal{L}_{adv}^G := -\log(\mathcal{D}(T, \hat{T})) - \log(1 - \mathcal{D}(\hat{T}, T)); \quad \mathcal{L}_{adv}^D := -\log(1 - \mathcal{D}(T, \hat{T})) - \log(\mathcal{D}(\hat{T}, T)), \tag{6}$$

where $\mathcal{D}$ stays invariant with [37].

Our overall objective function turns out to be:

$$\mathcal{L}_{tot} := \mathcal{L}_{rec} + \lambda_1\mathcal{L}_{per} + \lambda_2\mathcal{L}_{exc} + \lambda_3\mathcal{L}_{adv}, \tag{7}$$

with $\lambda_1 = 0.1$, $\lambda_3 = 1$, and $\lambda_3 = 0.01$ empirically set.

Table 1: Quantitative results on four real-world benchmark datasets of different methods. The best results are indicated in red and the second best results in blue.

Datasets	Metrics	Input	CEILNet	Zhang *et al.*	BDN	ERRNet	IBCLN	Lei *et al.*	YTMT-UCT
Real20 (20)	PSNR	19.16	18.45	22.55	18.41	22.89	21.86	22.35	23.26
	SSIM	0.732	0.690	0.788	0.726	0.803	0.762	0.793	0.806
Objects (200)	PSNR	23.74	23.62	22.68	22.72	24.87	24.87	23.81	24.87
	SSIM	0.878	0.867	0.879	0.856	0.896	0.893	0.882	0.896
Postcard (199)	PSNR	21.31	21.24	16.81	20.71	22.04	23.39	21.48	22.91
	SSIM	0.877	0.834	0.797	0.859	0.876	0.875	0.873	0.884
Wild (55)	PSNR	26.06	22.36	21.52	22.36	24.25	24.71	23.84	25.48
	SSIM	0.890	0.821	0.832	0.830	0.853	0.886	0.866	0.890

Table 2: Quantitative comparison in reflection recovery between the top-3 methods in transmission recovery. The best results are highlighed in red and the second best results in blue.

Datasets	Metrics	ERRNet	IBCLN	YTMT-UCT
Real20 (20)	PSNR	23.55	22.36	24.30
	SSIM	0.446	0.469	0.542
Objects (200)	PSNR	26.02	19.97	26.45
	SSIM	0.446	0.226	0.499
Postcard (199)	PSNR	22.47	13.16	23.41
	SSIM	0.419	0.230	0.478
Wild (55)	PSNR	25.52	20.83	27.33
	SSIM	0.460	0.298	0.590

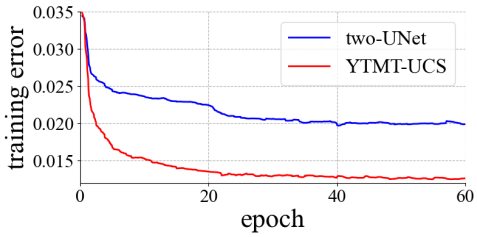


Figure 3: Training error on our training data of the two-UNet and YTMT-UCS. The proposed interaction strategy leads to a much faster decreasing speed.

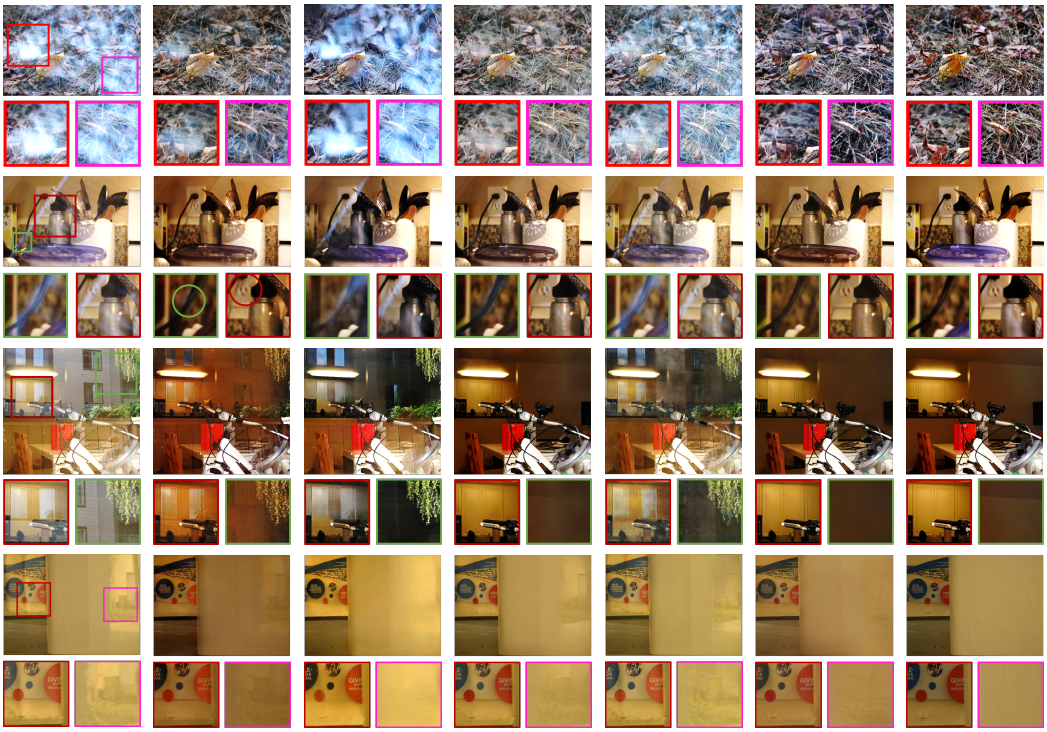
4 Experimental Validation on SIRS

4.1 Implementation Details

Our training data consists of both real-world and synthesis images, as used in [37]. Among these data, 90 pairs of input and transmission groundtruth are collected by Zhang *et al.* [43], and 7,643 image pairs chosen from the PASCAL VOC dataset [5] to synthesize superimposed images following CEILNet [6]. All of the models used in our experiments and ablation studies are trained with them. Moreover, our model is implemented in PyTorch and optimized with Adam optimizer, whose $\beta_1 = 0.9$, and $\beta_2 = 0.999$. The learning rate is initialized as 10^{-4} and then reduced by half at epoch 60, 80, and 100, respectively. The training is stopped at epoch 120. To compare with other SIRS state-of-the-art methods, we employ the YTMT-UCS, which consists of 10 YTMT block pairs per stage.

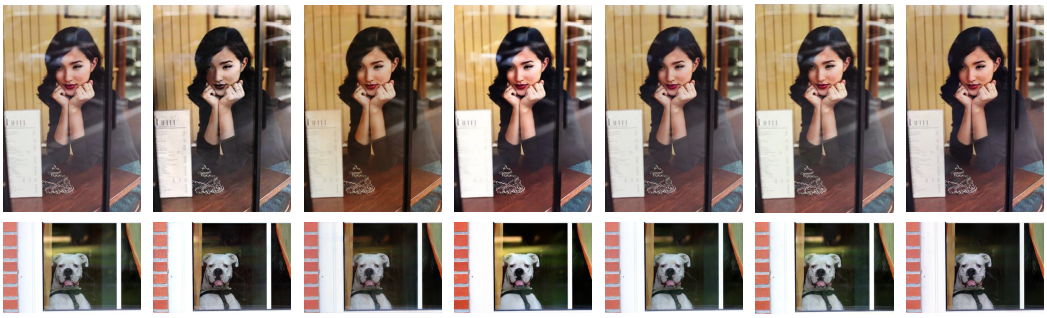
4.2 Comparison with State-of-the-art Methods

We select the proposed YTMT-UCT network to compare with prior state-of-the-art methods, including Zhang *et al.* [43], BDN [41], ERRNet [37], IBCLN [20] and Lei *et al.*[16], on four real-world dataset, involving Real20 [43] and three subsets of SIR^2 [34]. The PSNR and SSIM metrics are utilized to evaluate all the competing methods as shown in Table 1. Furthermore, we report the performance difference of the reflection recovery in Table 2, among the methods with the top 3 transmission recovery. Given ERRNet does not output reflection layers and the predicted reflections of IBCLN still mingle with large parts of transmission layers, we compare their reflection layers by $I - \hat{T}$ for fairness. The YTMT-UCT achieves the best results on almost all of the testing datasets, in terms of both transmission and the reflection recovery, which indicates that the YTMT strategy gives the most accurate separations with the help of frequent feature interaction between the transmission and reflection streams. The difficulty raising in Real20 comes from the largely overlapped regions and various reflection patterns, while the challenge of SIR^2 are mainly low-quality transmission layers and reflections blended with the monochromatic background. Therefore, we further conduct the qualitative comparison among these methods in Fig. 4 to show our capacity for this challenging samples. As can be observed, the method proposed by Zhang *et al.* produces results with severe color distortion and cannot handle the aforementioned globally overlapping problem. BDN and IBCLN



Input Zhang *et al.* [43] BDN [41] ERRNet [37] IBCLN [20] Ours GT

Figure 4: Visual comparison of state-of-the-art methods and ours on real-world testing dataset, including Real20 [43] and SIR^2 [6]. It shows that the results by our method are more visually striking compared with those by the competitors, with less artifacts and color distortion (please pay attention to the regions framed by red boxes).



Input CEILNet [6] Zhang *et al.* [43] BDN [41] ERRNet [37] IBCLN [20] Ours

Figure 5: Visual comparison of YTMT-UAT and state-of-the-art methods on real45 dataset[6]. It shows that our method produces results with fewer residual reflections.

generalize badly on the real20 dataset, failing to remove conspicuous reflections, and reflections are even enhanced by the BDN due to the inaccurate reflection estimation amplified by the cascade structure. Meanwhile, both the methods refine the separations stage-by-stage or iteration-by-iteration, where new outputs are concatenated to be fed into the next stage/iteration, instead of inner interaction or feature exchanging. As a consequence, the error would be rather accumulated. ERRNet introduces some useful network building blocks, but still leaves some obvious reflections. It merely estimates the transmission layer, discarding the interaction between two components. In conclusion, feature interaction is highly desired for the satisfactory separations. Moreover, the training error shown in Fig. 3 further accounts for the information efficacy of YTMT strategy, given the large margin of the convergence between the two-branch and YTMT-UCS solutions. More comparisons can be found in the supplementary material.

Table 3: Ablation study on different network architectures. All the models are trained for 70 epochs for fair comparison.

Datasets	Metrics	two-UNet	Split UNet	Concat. UNet	YTMT-UCS	YTMT-UCT
Real20	PSNR	20.57	22.05	22.79	23.05	23.09
	SSIM	0.752	0.793	0.802	0.805	0.802
Objects	PSNR	23.85	23.36	24.06	24.46	24.58
	SSIM	0.883	0.876	0.886	0.891	0.891
Postcard	PSNR	20.82	21.94	22.02	22.66	22.75
	SSIM	0.863	0.867	0.869	0.885	0.884
Wild	PSNR	24.20	25.22	24.71	25.24	25.46
	SSIM	0.881	0.890	0.859	0.887	0.892

Table 4: Ablation study on different YTMT variants and their two-stage versions, *i.e.* single-stage YTMT network with concat. variant as shown in Fig. 1 (b), with additive variant as in Fig. 1 (c) and their two-stage versions, respectively. All the models are trained for 70 epochs.

Datasets	Metrics	YTMT-UAS	YTMT-UCS	YTMT-UAT	YTMT-UCT
Real20	PSNR	23.26	23.05	23.39	23.09
	SSIM	0.801	0.805	0.809	0.802
Objects	PSNR	24.71	24.46	25.40	24.58
	SSIM	0.893	0.891	0.899	0.891
Postcard	PSNR	22.45	22.66	23.01	22.75
	SSIM	0.871	0.885	0.874	0.884
Wild	PSNR	24.49	25.24	25.19	25.46
	SSIM	0.870	0.887	0.880	0.892

4.3 Ablation Study

To further assess the capacity of the YTMT strategy for the task, we have developed several baselines to illustrate the effectiveness of our proposed YTMT strategy (see results in Table 3). The models are two-UNet without connections, splitting the channels in half and exchanging the features, directly exchanging the activation and concatenating them, single-stage U-shaped YTMT and the two-stage YTMT network, respectively. In the light of the superiority over other alternatives with proximal number of parameters, YTMT solutions do benefit from the strategy introducing the complementary interaction Q between the dual streams. We then move onto the concrete YTMT designs as shown in the Table 4. For this section, the choose of YTMT blocks as well as the usage of two stages are compared. It can be seen that the YTMT-UAT shows leading results on the first two datasets. Moreover, it shows that one more stage can indeed further improve the results. Note that the single-stage models are directly trained for 70 epochs and two-stage models are initialized with the best checkpoint of corresponding single-stage ones, and then trained to 70 epochs in total. The detailed architectures of all above networks and more visual comparisons can be found in the supplementary.

5 Application to Blind Denoising

Noisy images can be formulated as $I = B + N$, where B and N denote the clean background and noise, respectively, which also follows the additive model. Therefore, we further apply our YTMT strategy to the blind AWGN removal task to show its flexibility. It can be seen from Table 5 that the performance of DnCNN degrades rapidly while the noise level is going higher. On the contrary, ours performs more stably and surpasses both the DnCNN and two-branch DnCNN solutions with even fewer parameters. The visual comparison of results is shown in Fig. 6, where YTMT-PCS solution gives outputs with least artifacts and residual noise. The phenomenon illustrates the flexibility of YTMT strategy not only comes from the regularized subspace by two branches, but also the information efficiency of the YTMT interaction. Therefore, the YTMT strategy is quite hopeful to be generalized to other additive problems, like intrinsic image decomposition. We detail the YTMT network used in this section as follows.

Table 5: Quantitative comparison on the BSD68 grayscale dataset between DnCNN (1.01M), Two-branch DnCNN (1.16M) and a simple YTMT network (0.92M). The models are trained in the noise level range of [0,75] for 50 epochs, and tested on $[25, 100]$.

Models	Metrics	Noise Levels							
		25	50	75	80	85	90	95	100
DnCNN	PSNR	28.81	25.85	24.17	23.56	22.58	21.28	19.85	18.46
	SSIM	0.814	0.693	0.612	0.578	0.521	0.449	0.376	0.314
DnCNN-2	PSNR	28.48	25.51	23.77	23.34	22.83	22.22	21.54	20.80
	SSIM	0.798	0.669	0.580	0.557	0.530	0.497	0.461	0.415
YTMT-PCS	PSNR	28.78	25.71	24.09	23.83	23.54	23.07	22.32	21.08
	SSIM	0.812	0.688	0.605	0.592	0.573	0.539	0.486	0.418

Input DnCNN DnCNN-2 YTMT-PCS

Figure 6: Visual comparison of models for blind denoising at noise level 85. The columns 2-4 are 9-layer DnCNN, 11-layer two-branch DnCNN and 10-layer YTMT-PCS, respectively.

The plain YTMT network is developed for this task, namely YTMT-PCS. It can be easily structured by the stack of proposed YTMT blocks (see Fig. 2 (b)). Concretely, the input is augmented by the hyper column firstly, gaining 1475 channels in total, and then fed into the dual streams. The head YTMT blocks first convert the number of channels to 64 with 1×1 convolutions, then use the YTMT strategy to connect with both the streams. The number of channels is kept to be 64 until the final YTMT blocks, which maps the number to be 1 or 3, depending on the original input. In this task, YTMT-PCS consists of 5 YTMT block pairs and BatchNorm layers are omitted, having 0.92M parameters in total.

We enforce the linear combination with $\|\hat{B} + \hat{N} - I\|_2^2$, where $\hat{B}$ denotes the estimated background, $\hat{N}$ the predicted noise. The reconstruction loss is defined as:

$$\mathcal{L} = \|\hat{B} - B\|_2^2 + \gamma_1 \|\hat{N} - N\|_2^2 + \gamma_2 \|\hat{B} + \hat{N} - I\|_2^2 \tag{8}$$

where $\gamma_1 = 2.0$ and $\gamma_2 = 3.0$. Both the models are trained within the level range of $[0, 75]$, and tested in $[25, 100]$, and the other settings are controlled to be the same. All the models are trained for 50 epochs with initial learning rate of 0.001 and 0.0001 after 30 epochs.

6 Conclusion

In this paper, we proposed a general rule for deep interactive dual-stream/branch learning, namely your trash is my treasure (denoted as YTMT), which says that two branches should communicate with each other frequently by exchanging, rather than discarding, the information useless to themselves. Activation functions are deemed to be suitable for determining which part of information to exchange. As an example, the widely-used ReLU was chosen to validate the primary claims of this work. In addition, we offered several YTMT based designs based on both U-shaped and plain backbones to show the flexibility. Extensive experimental results on public SIRS and denoising datasets have been provided to verify the effectiveness of YTMT, and demonstrate the clear advantages of our method in comparison with other state-of-the-art alternatives. Another merit of YTMT comes from its acceleration in decreasing error during training. It is positive that our strategy can derive diverse designs and be beneficial to many two-component decomposition tasks.

References

[1] Amit K. Agrawal, Ramesh Raskar, Shree K. Nayar, and Yuanzhen Li. Removing photography artifacts using gradient projection and flash-exposure sampling. *ACM Trans. Graph.*, 24(3):828–835, 2005.

[2] Djork-Arné Clevert, Thomas Unterthiner, and Sepp Hochreiter. Fast and accurate deep network learning by exponential linear units (elus). In *ICLR*, 2016.

[3] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Fei-Fei Li. Imagenet: A large-scale hierarchical image database. In *CVPR*, pages 248–255, 2009.

[4] Charles Dugas, Yoshua Bengio, François Bélisle, Claude Nadeau, and René Garcia. Incorporating second-order functional knowledge for better option pricing. In *NeurIPS*, pages 472–478, 2000.

[5] Mark Everingham, Luc Van Gool, Christopher K. I. Williams, John M. Winn, and Andrew Zisserman. The pascal visual object classes (VOC) challenge. *IJCV*, 88(2):303–338, 2010.

[6] Qingnan Fan, Jiaolong Yang, Gang Hua, Baoquan Chen, and David P. Wipf. A generic deep architecture for single image reflection removal and image smoothing. In *ICCV*, pages 3258–3267, 2017.

[7] Hany Farid and Edward H. Adelson. Separating reflections and lighting using independent components analysis. In *CVPR*, pages 1262–1267, 1999.

[8] Kun Gai, Zhenwei Shi, and Changshui Zhang. Blind separation of superimposed moving images using image statistics. *TPAMI*, 34(1):19–32, 2012.

[9] Xiaojie Guo, Xiaochun Cao, and Yi Ma. Robust separation of reflection from multiple images. In *CVPR*, pages 2195–2202, 2014.

[10] Byeong-Ju Han and Jae-Young Sim. Reflection removal using low-rank matrix completion. In *CVPR*, pages 3872–3880, 2017.

[11] Bharath Hariharan, Pablo Andrés Arbeláez, Ross B. Girshick, and Jitendra Malik. Hypercolumns for object segmentation and fine-grained localization. In *CVPR*, pages 447–456, 2015.

[12] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *ICCV*, pages 1026–1034, 2015.

[13] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, pages 770–778, 2016.

[14] Justin Johnson, Alexandre Alahi, and Li Fei-Fei. Perceptual losses for real-time style transfer and super-resolution. In *ECCV*, pages 694–711, 2016.

[15] Alexia Jolicoeur-Martineau. The relativistic discriminator: a key element missing from standard GAN. In *ICLR*, 2019.

[16] Chenyang Lei, Xuhua Huang, Mengdi Zhang, Qiong Yan, Wenxiu Sun, and Qifeng Chen. Polarized reflection removal with perfect alignment in the wild. In *CVPR*, pages 1747–1755, 2020.

[17] Anat Levin and Yair Weiss. User assisted separation of reflections from a single image using a sparsity prior. *TPAMI*, 29(9):1647–1654, 2007.

[18] Anat Levin, Assaf Zomet, and Yair Weiss. Learning to perceive transparency from the statistics of natural scenes. In *NeurIPS*, pages 1247–1254, 2002.

[19] Anat Levin, Assaf Zomet, and Yair Weiss. Separating reflections from a single image using local features. In *CVPR*, pages 306–313, 2004.

[20] Chao Li, Yixiao Yang, Kun He, Stephen Lin, and John E. Hopcroft. Single image reflection removal through cascaded refinement. In *CVPR*, pages 3562–3571, 2020.

[21] Siyuan Li, Wenqi Ren, Jiawan Zhang, Jinke Yu, and Xiaojie Guo. Single image rain removal via a deep decomposition–composition network. *Computer Vision and Image Understanding*, 186:48–57, 2019.

[22] Yu Li and Michael S. Brown. Exploiting reflection change for automatic reflection removal. In *ICCV*, pages 2432–2439, 2013.

[23] Yu Li and Michael S. Brown. Single image layer separation using relative smoothness. In *CVPR*, pages 2752–2759, 2014.

[24] Andrew L Maas, Awni Y Hannun, and Andrew Y Ng. Rectifier nonlinearities improve neural network acoustic models. In *Proc. icml*, page 3, 2013.

[25] Xu Qin, Zhilin Wang, Yuanchao Bai, Xiaodong Xie, and Huizhu Jia. Ffa-net: Feature fusion attention network for single image dehazing. In *AAAI*, pages 11908–11915, 2020.

[26] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *MICCAI*, pages 234–241, 2015.

[27] Bernard Sarel and Michal Irani. Separating transparent layers through layer information exchange. In *ECCV*, pages 328–341, 2004.

[28] Yoav Y. Schechner, Nahum Kiryati, and Ronen Basri. Separation of transparent layers using focus. *IJCV*, 39(1):25–39, 2000.

[29] Christian Simon and In Kyu Park. Reflection removal for in-vehicle black box videos. In *CVPR*, pages 4231–4239, 2015.

[30] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. In *ICLR*, 2015.

[31] Sudipta N. Sinha, Johannes Kopf, Michael Goesele, Daniel Scharstein, and Richard Szeliski. Image-based rendering for scenes with reflections. *ACM Trans. Graph.*, 31(4):100:1–100:10, 2012.

[32] Chao Sun, Shuaicheng Liu, Taotao Yang, Bing Zeng, Zhengning Wang, and Guanghui Liu. Automatic reflection removal using gradient intensity and motion cues. In *ACM MM*, pages 466–470, 2016.

[33] Richard Szeliski, Shai Avidan, and P. Anandan. Layer extraction from multiple images containing reflections and transparency. In *CVPR*, page 1246, 2000.

[34] Renjie Wan, Boxin Shi, Ling-Yu Duan, Ah-Hwee Tan, and Alex C. Kot. Benchmarking single-image reflection removal algorithms. In *ICCV*, pages 3942–3950, 2017.

[35] Renjie Wan, Boxin Shi, Ling-Yu Duan, Ah-Hwee Tan, and Alex C. Kot. CRRN: multi-scale guided concurrent reflection removal network. In *CVPR*, pages 4777–4785, 2018.

[36] Renjie Wan, Boxin Shi, Ah-Hwee Tan, and Alex C. Kot. Depth of field guided reflection removal. In *ICIP*, pages 21–25, 2016.

[37] Kaixuan Wei, Jiaolong Yang, Ying Fu, David P. Wipf, and Hua Huang. Single image reflection removal exploiting misaligned training data and network enhancements. In *CVPR*, pages 8178–8187, 2019.

[38] Qiang Wen, Yinjie Tan, Jing Qin, Wenxi Liu, Guoqiang Han, and Shengfeng He. Single image reflection removal beyond linearity. In *CVPR*, pages 3771–3779, 2019.

[39] Tianfan Xue, Michael Rubinstein, Ce Liu, and William T. Freeman. A computational approach for obstruction-free photography. *ACM Trans. Graph.*, 34(4):79:1–79:11, 2015.

[40] Jiaolong Yang, Hongdong Li, Yuchao Dai, and Robby T. Tan. Robust optical flow estimation of double-layer images under transparency or reflection. In *CVPR*, pages 1410–1419, 2016.

[41] Jie Yang, Dong Gong, Lingqiao Liu, and Qinfeng Shi. Seeing deeply and bidirectionally: A deep learning approach for single image reflection removal. In *ECCV*, pages 675–691, 2018.

[42] Kai Zhang, Wangmeng Zuo, Yunjin Chen, Deyu Meng, and Lei Zhang. Beyond a gaussian denoiser: Residual learning of deep CNN for image denoising. *TIP*, 26(7):3142–3155, 2017.

[43] Xuaner Cecilia Zhang, Ren Ng, and Qifeng Chen. Single image reflection separation with perceptual losses. In *CVPR*, pages 4786–4794, 2018.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes]
   (c) Did you discuss any potential negative societal impacts of your work? [No] , since the SIRS task does not have potential to cause negative societal impacts.
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [N/A]
   (b) Did you include complete proofs of all theoretical results? [N/A]
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes]
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] , please see Sec. 4.
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [No]
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes] , please see Sec. 4.
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes]
   (b) Did you mention the license of the assets? [N/A]
   (c) Did you include any new assets either in the supplemental material or as a URL? [Yes]
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [Yes] , the datasets used are public.
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [No] , no identification labels are involved in our work.
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]