ReconfigISP: Reconfigurable Camera Image Processing Pipeline

Ke Yu[1,2] Zexian Li[1,3] Yue Peng[1] Chen Change Loy[4] Jinwei Gu[1,5]
[1]SenseTime Research and Tetras.AI [2]CUHK-SenseTime Joint Lab, The Chinese University of Hong Kong
[3]Beihang University [4]S-Lab, Nanyang Technological University [5]Shanghai AI Laboratory
ericyu16@hotmail.com lizexian0427@gmail.com
{pengyue, gujinwei}@tetras.ai ccloy@ntu.edu.sg

Abstract

Image Signal Processor (ISP) is a crucial component in digital cameras that transforms sensor signals into images for us to perceive and understand. Existing ISP designs always adopt a fixed architecture, e.g.*, several sequential modules connected in a rigid order. Such a fixed ISP architecture may be suboptimal for real-world applications, where camera sensors, scenes and tasks are diverse. In this study, we propose a novel Reconfigurable ISP (ReconfigISP) whose architecture and parameters can be automatically tailored to specific data and tasks. In particular, we implement several ISP modules, and enable back-propagation for each module by training a differentiable proxy, hence allowing us to leverage the popular differentiable neural architecture search and effectively search for the optimal ISP architecture. A proxy tuning mechanism is adopted to maintain the accuracy of proxy networks in all cases. Extensive experiments conducted on image restoration and object detection, with different sensors, light conditions and efficiency constraints, validate the effectiveness of ReconfigISP. Only hundreds of parameters need tuning for every task*[1].

1. Introduction

A digital camera uses an ISP to transform original RAW images to high-quality RGB images that can be displayed on screen. Existing ISP systems typically adopt a human-designed pipeline [16] comprised of specialized modules, each of which addresses a subtask, such as denoising [12, 22], demosaicing [11, 17], white balancing [13], *etc*. Traditional ISP is highly modular and efficient. However, once the pipeline is fixed, it can hardly be adapted to different application scenarios. A costly and manual tuning of parameters is required.

In this study, our goal is to keep the modular design of the traditional ISP, but learn a reconfigurable ISP architecture that can quickly adapt to various tasks (*e.g*., restoration and object detection), scenes (*e.g*., daytime and nighttime) and runtime constraints. To this end, we present a novel and versatile *reconfigurable* ISP framework, named ReconfigISP. The proposed framework allows changes and rerouting in the ISP architecture, *i.e*., what basic modules should be chosen and how they are connected, optimized towards designated objective functions. Given the flexibility, specialized modules can be combined to generate numerous ISP pipelines to handle various application scenarios, but still preserving the modularity and parameters of each module. Experimental results show that our framework achieves a significant improvement over traditional camera pipelines (Sec. 4). Examples are shown in Fig. 1.

Searching for an optimal ISP pipeline is a new and interesting problem. In this work, we make several technical contributions to address some of the non-trivial challenges: *(1) Tunable ISP modules* - Most existing ISP modules are non-differentiable. As a result, the parameters of these modules cannot be efficiently optimized with end-to-end training, making the search for optimal pipeline intractable. To address this issue, we use a convolutional neural network to imitate each non-differentiable module. The proxy network observes input images and module parameters, together with global image statistics that the module may need. The network is trained to resemble the output of the original non-differentiable module. With all modules differentiable, the ISP architecture can be efficiently explored with the help of neural architecture search. *(2) Proxy tuning and online pruning* - With the aforementioned framework, we still need to resolve the gap in data distribution. In particular, the data we use for ISP architecture search depend on the target task, and they might be unseen to a pre-trained proxy network. To mitigate the domain gap, we devise an effective proxy tuning approach – fine-tuning proxy networks with the observed fresh data on the fly during ISP architecture search. Our approach updates the ISP architecture and proxy networks alternately. And it gradually prunes modules with low performance to accelerate the

[1]Codes and models are available at https://d ngmj rw.jollibeefood.restlab-nku.com/project/reconfigisp/

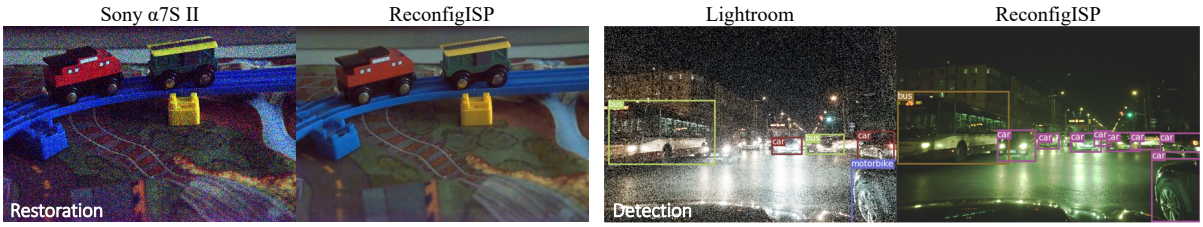


Figure 1: While inheriting the modularity of traditional ISP, ReconfigISP can adapt to different tasks, such as image restoration and object detection by learning to reconnect specialized ISP modules. For each task, the left image shows the results of a default camera processing pipeline while the right image presents our output (zoom in for best view). In the task of object detection, ReconfigISP optimizes the ISP for more accurate detections rather than perceptual quality.

training process.

We note that there is a surge of interest in deep ISP approaches [5, 7, 27], which converts an ISP pipeline to a single end-to-end deep network. ReconfigISP offers a compelling alternative to the deep ISP paradigm. First, ReconfigISP can adjust its pipeline purely driven by objective functions, while existing deep ISP solutions do not offer such flexibility. For instance, to fulfill a specific runtime constraint, a deep ISP solution will need to redesign the network architecture from scratch. Perhaps more notably, ReconfigISP preserves the modularity of the image formation process stages. It is interpretable as each module plays a clear role in the ISP pipeline. Hence, our method still allows human intervention if needed, and it offers insights to design principles of effective ISP pipelines beyond just a reconfigurable framework. Moreover, to cope with different processing functionalities, deep ISP methods typically require an extensive network with millions of parameters. Expensive retraining with a large quantity of data is needed for every new task to prevent overfitting. The data requirement is unrealistic, especially if one wishes to repurpose the ISP for complex tasks like object objection. In contrast, our framework only needs to tune hundreds of parameters with limited data. We will validate the above advantages of ReconfigISP in our experiments.

2. Related Work

ISP Modules. An ISP usually involves an elaborated pipeline to handle image noise, chrominance, luminance, sharpness, *etc*. In this study, we prepare a concise module pool, as shown in Table 1, with focus on common algorithms for denoising, demosaicing, tone mapping and white balancing. More details about the modules can be found in the supplementary material. It is noteworthy that many classic algorithms are non-differentiable, for instance, denoising algorithms like Bilateral [28], Median [15] and NLM [4], and BM3D [6]; and tone mapping algorithms like Reinhard [25] and Crysisengine. We convert these algorithms to proxy networks to facilitate ISP architecture search. Note that we also consider some lightweight deep models in our module pools, including Path-Restore [31] and DemosaicNet [11]. All proxy modules will be released.

ISP Pipelines. There are several studies in the literature that focus on either traditional camera pipelines or deep ISP approaches. Hu *et al*. [14] propose Exposure, a white-box image post-processing framework with eight differentiable filters. Karaimer *et al*. [16] devise a 12-stage software ISP that allows users to study the effects of each algorithm in the context of a full ISP. However, such traditional ISP framework does not support end-to-end optimization. Tseng *et al*. [29] propose a proxy optimization scheme to efficiently search for the best ISP hyperparameters. Mosleh *et al*. [20] further develop an evolutionary algorithm to optimize ISP parameters on the hardware. Different from existing proxy approach [29], we train a differentiable proxy for each non-differentiable algorithm instead of the whole ISP pipeline. Thus, we can explore different ISP architectures while the method of [29] only applies to a fixed pipeline.

Deep-learning-based ISP is drawing considerable research interest in recent years. Schwartz *et al*. [27] propose DeepISP that uses a deep neural network that contains a low-level stage and a high-level stage. Dai *et al*. [7] devise a network with attention and wavelet transform. Chen *et al*. [5] propose SID dataset containing paired data between low light and normal illuminance. A U-Net structure trained with SID dataset significantly outperforms the traditional pipeline embedded in the camera. In this work, we use this U-Net model as a strong competitor to verify the effectiveness of our method. The aforementioned studies adopt a fixed network to replace the ISP pipeline. The network is usually large, containing massive number of parameters that require tuning or retraining for every new task. As mentioned in the introduction, these approaches do not enjoy the same flexibility and modularity as in the proposed ReconfigISP, which is readily adaptable to different application scenarios at a low cost by adjusting both ISP architecture and parameters given the designated cost functions.

3. Methodology

The overview framework of ReconfigISP is shown in Fig. 2. In Sec. 3.1, we introduce differentiable proxy networks to replace non-differentiable modules. In Sec. 3.2,

Table 1: The module pool contains 22 algorithms. "Domain" represents the input and output patterns. "Category" describes the functionality of the algorithms. The number in the parentheses denotes the quantity of configurable parameters for each algorithm. The *italic algorithms* are non-differentiable on GPU in our implementation.

Domain	Category	Algorithm
RAW ->RAW	Denoising	*Bilateral-Bayer (3)*, *Median-Bayer (1)*, *NLM-Bayer (3)*, Path-Restore-Bayer (0)
RAW ->sRGB	Demosaicing	*Laplacian (0)* [30], Nearest (0), Bilinear (0) [10], DemosaicNet (0) [11]
sRGB ->sRGB	Denoising	*Bilateral (3)* [28], *Median (1)* [15], *NLM (3)* [4], *BM3D (5)* [6], Path-Restore (0) [31]
	Gamma Correction	Gamma (1)
	Global Tone Mapping	*Reinhard (2)* [25], *Crysisengine (1)*, *Filmic (2)* [9], Manual (3)
	White Balance	*Whitepatch (1)* [26], Grayworld (0), Linear (3), Quadratic (30) [27]

the algorithm for ISP architecture search is discussed. Finally, in Sec. 3.3, we explain how module parameters are fine-tuned after the ISP architecture is fixed.

3.1. Differentiable Proxy Networks

Given the module pool shown in Table 1, our goal is to search for the most effective ISP pipeline composed of these algorithms. Some of the algorithms are intrinsically differentiable and thus can be optimized in an end-to-end manner, while others are non-differentiable, causing difficulties in architecture and parameter optimization. To address this challenge, we make a differentiable proxy network for each non-differentiable module.

Formally, let $\hat{f}_j$ denote the j-th module and f_j denote its proxy network with weights w_j. Suppose $\hat{f}_j$ is differentiable, and we need not apply any change to this algorithm, *i.e.*, $f_j = \hat{f}_j$. As shown in Fig. 2 (a), to guarantee that the proxy network resembles the original algorithm for different input images $\mathbf{X}$ and parameters p_j, we randomly sample images and parameters while training the proxy network. Global image statistics are extracted for non-local modules. Let L_p denotes the fidelity loss for proxy training, *e.g.*, L1 or L2 loss. The objective function is then formulated as $L_p\left(f_j(\mathbf{X}, p_j; w_j), \hat{f}_j(\mathbf{X}, p_j)\right)$.

3.2. Reconfigurable ISP Architecture

Differentiable ISP Architecture Search. Our ISP architecture search is inspired by existing work on neural architecture search (NAS). Considering that NAS based on reinforcement learning [2, 32] and evolutionary methods [23] is expensive, we adopt the more efficient differentiable NAS. In particular, we choose Darts [19], with our newly proposed online pruning and proxy tuning to solve the specific challenges in ISP. Details are provided as follows.

Given all the differentiable proxies at hand, we incorporate these small ISP modules into a super network for architecture search, as shown in Fig. 2 (b). Let K denotes the maximum length of the desired ISP pipeline. At each step k, there are $N+1$ modules parallelly connected, *i.e.*, N differentiable algorithms and one skip connection module. Note that not all modules are valid at each step. For example, an algorithm designed for sRGB images cannot work in the RAW domain. Therefore, we need to specify the valid domain for each step, and then a set of valid index V can be derived. If $(k, j) \in V$, then the j-th module is valid at step k. For every valid module f_{kj}, it is assigned an architecture weight α_{kj}, representing how likely it is to select this module. All the architecture weights should be non-negative and sum to one within a single step, *i.e.*, $\sum_{j=1}^{N} \alpha_{kj} = 1, \text{s.t.}, \alpha_{kj} > 0, (k, j) \in V$.

Next, we briefly introduce the inference process of the constructed super network. Let $\mathbf{X}$ denote the input image. The intermediate input image at step k is denoted by $\mathbf{X}_{k-1}$. The network inference at step k can be formulated as:

$$\mathbf{X}_k = \sum_{j=1}^{N} \alpha_{kj} f_j(\mathbf{X}_{k-1}, p_{kj}; w_j), \text{ s.t. } (k, j) \in V, \quad (1)$$

where p_{kj} represents the input parameters of the algorithm f_j at step k, and w_j denotes the weight of proxy network if f_j is a differentiable proxy. For example, suppose f_j denotes the proxy network of a median filter, then p_{kj} is the filter size at k-th step, and w_j represents the network weights.

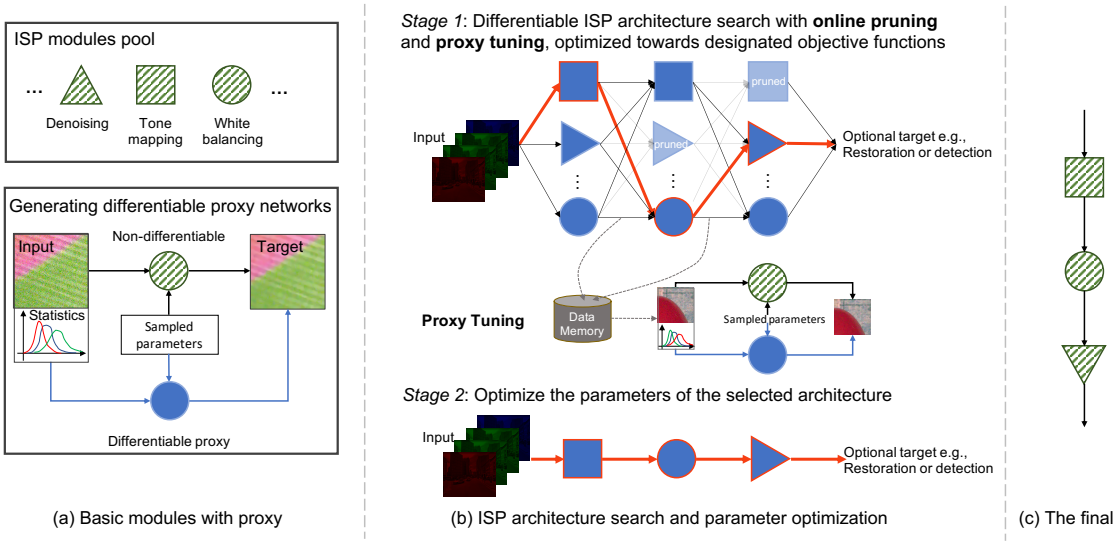
We alternately update the architecture α_{kj} and the parameters of each ISP module p_{kj}. Let $L(\hat{\mathbf{X}}, \mathbf{Y})$ denote the loss function, where $\hat{\mathbf{X}} = \mathbf{X}_K$ is the output of the ISP pipeline, and $\mathbf{Y}$ is the target image or label. The loss function depends on the specific task. To update the algorithm parameters, we derive the gradients as:

$$\delta_{p_{kj}} = \nabla_{p_{kj}} L(\hat{\mathbf{X}}, \mathbf{Y}; p, \alpha), \quad (2)$$

where p and α denote all the algorithm parameters $\{p_{kj}\}$ and architecture weights $\{\alpha_{kj}\}$, respectively.

As for the update of architecture, directly applying the gradients is suboptimal. As discussed in Darts [19], the optimal algorithm parameters may vary according to the specific architecture. Thus, we should also consider the algorithm parameters while optimizing the architecture. Specifically, we conduct one step of meta learning to get better architecture gradients:

$$\begin{aligned} \hat{p} &= p - \xi \nabla_p L(\hat{\mathbf{X}}_{train}, \mathbf{Y}_{train}; p, \alpha), \\ \delta_{\alpha_{kj}} &= \nabla_{\alpha_{kj}} L\left(\hat{\mathbf{X}}_{val}, \mathbf{Y}_{val}; \hat{p}, \alpha\right), \end{aligned} \quad (3)$$



(a) Basic modules with proxy

(b) ISP architecture search and parameter optimization

(c) The final ISP

Figure 2: The overview of ReconfigISP. (a) We implement a comprehensive set of ISP modules ($N = 22$), and construct a differentiable proxy for each non-differentiable algorithm. (b) In stage 1, we search for the optimal ISP architecture using Darts [19] with the newly proposed online pruning and proxy tuning. Finally, in stage 2, the algorithm parameters of the selected architecture are fine-tuned for the best performance. (c) The final ISP adopts the searched architecture with original ISP modules and optimized parameters.

where ξ represents the learning rate for meta training. The data are divided into two groups, for meta training and meta validation, respectively.

Online Pruning. The aforementioned searching algorithm is still time consuming in practice due to the expensive structure of the super network. Each algorithm is repeatedly conducted at every step, resulting in a huge quantity of computations. To alleviate this computational burden, we prune those candidate modules with low architecture weights.

An intuitive pruning strategy is to set a hard threshold of the architecture weight. The module whose architecture weight is smaller than the threshold should be discarded. However, we found such strategy sensitive to the optimization hyper-parameters. The pruning results can be different with multiple trials. Hence, we set a relative threshold according to the architecture weight with the highest potential. In particular, the pruning mechanism can be formulated as

$$\begin{cases} (k,j) \in V, & \alpha_{kj} > \eta \max_j \alpha_{kj}, \\ (k,j) \notin V, & \text{otherwise}, \end{cases} \tag{4}$$

where $\eta \in (0,1)$ is the relative threshold.

Proxy Tuning. As we may apply ReconfigISP to different tasks, the data distribution can be very diverse. Thus, the proxy training data may not cover the data distribution of the target task. Moreover, the intermediate result $\mathbf{X}_k$ is a combination of the outputs from multiple algorithms, which is unseen in the proxy training process. Therefore, the proxy network may fail to approach the original algorithm given this data distribution gap. To mitigate this issue, we propose proxy tuning, which fine-tunes the proxy networks using the observed new data during the architecture search.

The process of proxy tuning is illustrated in Fig. 2 (b). Specifically, we build a data memory M to store the intermediate outputs from the super network. The data memory is organized as a queue, with a maximum capacity of $|M|_{\max}$. At each training timestep t, the intermediate results $\mathbf{X}_1, \ldots, \mathbf{X}_K$ are added to the queue, and some past images are removed if the queue is full. At every t_p step, a batch of data $\mathbf{X}_m$ are randomly sampled from the data memory and backpropagation is applied to derive the gradients of proxy network weights:

$$\delta_{w_j} = \nabla_{w_j} L_p \left(f_j(\mathbf{X}_m, \tilde{p}_j; w_j), \hat{f}_j(\mathbf{X}_m, \tilde{p}_j) \right), \tag{5}$$

where $\hat{f}_j$ denotes the original algorithm corresponding to the proxy network f_j, and $\tilde{p}_j$ represents a set of randomly sampled parameters for this algorithm. L_p denotes the loss function for proxy training. We do not perform proxy tuning on the module that does not need a proxy network.

Summary. The overall procedure for ISP architecture search is illustrated in Fig. 2 (b). The searching algorithm is summarized in Algorithm 1. At each timestep t, we update architecture and algorithm parameters, refresh the data memory, and perform online pruning to cut off unnecessary modules. At every t_p steps, we conduct proxy tuning to ensure the effectiveness of each proxy network.

3.3. ISP Parameters Optimization

After the architecture search, we select the ISP pipeline with the highest architecture weights. Proxy networks are replaced back with the original modules (Fig. 2 (c)). In particular, at step k, the index of the selected module is $a_k = \text{argmax}_j \alpha_{kj}$, and the selected module is denoted by f_{a_k}.

Algorithm 1 ISP Architecture Search

Prepare for the module pool $\{f_j\}$ with proxy weights $\{w_j\}$
Initialize algorithm parameters $\{p_{kj}\}$ and architecture $\{\alpha_{kj}\}$
Initialize data memory $M = \emptyset$, set the memory size $|M|_{\max}$
Specify learning rate γ, total iterations T, tuning interval t_p
for $t = 1, T$ **do**
  Sample training data $\mathbf{X}_{train}$, $\mathbf{Y}_{train}$, $\mathbf{X}_{val}$, $\mathbf{Y}_{val}$
  $\alpha_{kj} \leftarrow \alpha_{kj} - \gamma\delta_{\alpha_{kj}}$ ▷ Update architecture, Eq. (3)
  $p_{kj} \leftarrow p_{kj} - \gamma\delta_{p_{kj}}$ ▷ Update parameters, Eq. (2)
  Update memory M using intermediate data $\mathbf{X}_k$
  Update valid module $V = \{(k, j)\}$ ▷ Pruning, Eq. (4)
  if $t \equiv 0 \mod t_p$ **then**
    Sample data $\mathbf{X}_m$ from memory M
    $w_j \leftarrow w_j - \gamma\delta_{w_j}$ ▷ Proxy tuning, Eq. (5)
  end if
end for

Given the searched ISP architecture, the algorithm parameters are not optimal, because there are still multiple algorithms not pruned at each step in the super network. Therefore, we further optimize the algorithm parameters specialized for the selected ISP pipeline. The inference process at the k-th step is formulated as $\mathbf{X}_k = f_{a_k}(\mathbf{X}_{k-1}, p_{a_k}; w_{a_k})$, where p_{a_k} is the algorithm parameters we aim to optimize, and w_{a_k} stands for the network weights after proxy tuning.

The loss function depends on the specific task. For image restoration, we adopt L2 loss, $L = ||\hat{\mathbf{X}} - \mathbf{Y}||_2^2$, where $\hat{\mathbf{X}} = \mathbf{X}_K$ and $\mathbf{Y}$ denote the output image and the ground truth, respectively. As for object detection, we follow the design of YOLOv3 [24]. The loss function is composed of a localization term and a classification term to penalize coordinate errors and class label mismatch, respectively. Besides the above loss functions, ReconfigISP can be optimized for any objective function as long as backpropagation is applicable.

4. Experiments

To facilitate quantitative analysis, we set the total length of ISP as $K = 5$[2], with one step in the RAW domain, one demosaicing step, and three steps in the sRGB domain. In this specific setting, the overall ReconfigISP framework contains 226 architecture and module parameters in total. We adopt L2 loss for proxy training. The data memory size is chosen as 1,000. The online tuning threshold η is set as 0.2. The proxy tuning interval t_p is 20.

The total number of iterations are 200,000 and 80,000 for architecture search and parameter optimization, respectively. The learning rate γ and meta learning rate ξ are the same. The initial value is 1×10^{-4}, and then it is decayed by a half every quarter of the training process. SRCNN [8] is used as the proxy network, and the differentiable proxies are pre-trained on SIDD [1] dataset. We use Adam [18] optimizer and implement our framework on PyTorch [21]. Our experiments are conducted on four NVIDIA GeForce GTX 1080 GPUs.

[2]Note that this is a simplified pipeline since in practice an ISP pipeline contains many modules. But our method still shows competitive results in comparison to the very deep ISP approach (see Sec. 4.5).

Table 2: Quantitative results on SID Dataset [5].

SID 0.1s to 10s	RGB		Gray	
	PSNR	SSIM	PSNR	SSIM
Sony α7 II	15.69	0.1550	20.34	0.3352
ReconfigISP	**25.65**	**0.7527**	**31.90**	**0.9008**

Table 3: Quantitative results on S7 ISP Dataset [27].

S7 ISP	RGB		Gray	
	PSNR	SSIM	PSNR	SSIM
Samsung S7	21.08	0.4518	23.99	0.6297
ReconfigISP	**23.31**	**0.7007**	**26.83**	**0.7697**

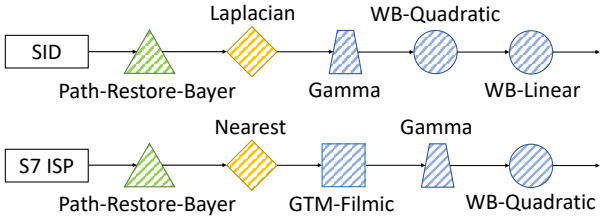


Figure 3: The searched ISP architecture on SID and S7 ISP datasets for image restoration. The green, yellow and blue colors represent the mapping domain of RAW→RAW, RAW→sRGB and sRGB→sRGB, respectively. GTM and WB stand for global tone mapping and white balance, respectively.

4.1. Evaluation on Image Restoration

To validate the effectiveness of the proposed method for different sensors, scenes and light conditions, we select two challenging benchmarks that the proxy networks have never observed – SID dataset [5] containing images captured by a DSLR camera, and S7 ISP dataset [27] that is composed of smartphone images.

SID Dataset. We verify the effectiveness of our method on SID dataset [5], which contains several challenging images captured under extreme low light. We conduct experiments on the Sony subset, where images are captured by Sony α7S II. To verify the superiority of our algorithm on small-scale data, we randomly select 5% of the training images. Without loss of generality, we consider a specific setting with 0.1s short exposure and 10s long exposure.

The searched ISP pipeline is shown in Fig. 3 (top). A deep denoising method Path-Restore-Bayer is chosen in the RAW domain. The Laplacian interpolation is used for demosaicing. In the sRGB domain, gamma correction is applied followed by two white balancing modules, indicating that it is challenging to adjust the white balance in extreme low-light conditions.

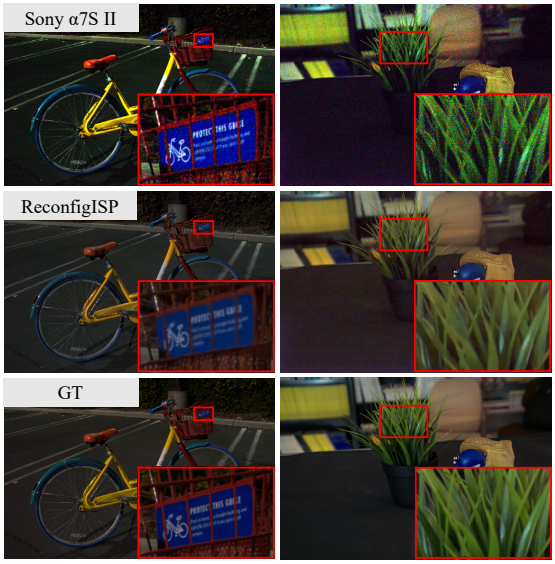


Figure 4: Qualitative results on SID dataset [5]. The input and ground-truth exposure time are 0.1s and 10s, respectively.

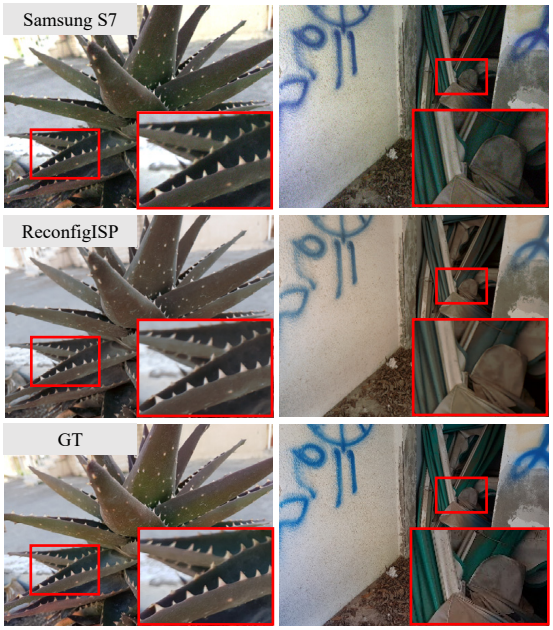


Figure 5: Qualitative results on S7 ISP dataset [27].

Quantitative comparisons are shown in Table 2. Sony α7 II represents the results produced by the default camera pipeline, multiplied by an exposure compensation coefficient of 100 (10s divided by 0.1s). We present the PSNR and SSIM results of color and gray images in Table 2. Our ReconfigISP surpasses the default camera ISP by a large margin, *i.e.*, more than 10 dB improvement. Qualitative results are presented in Fig. 4. Compared to the Sony pipeline, our method yields cleaner results with less noise, and the exposure looks more natural. It is worth noting that our method achieves such compelling performance with only hundreds of parameters being tuned.

S7 ISP Dataset. S7 ISP [27] is a dataset collected from Samsung S7 rear camera. Following the setting of SID, we only use 5% of the training data. The quantity of validation and test subsets is not changed.

The searched ISP architecture is shown in Fig. 3 (bottom). The pipeline is slightly different from that of SID. In particular, Nearest interpolation is used instead of Laplacian filter. Moreover, a global tone mapping method Filmic is applied prior to gamma correction. We observe that Filmic works well for S7 ISP dataset, but it may amplify artifacts when applied to SID dataset, where the noise is severe under extreme low light.

Quantitative results are presented in Table 3. It is observed that ReconfigISP is superior to the Samsung camera pipeline by more than 2 dB on both RBG and gray images. Visual results are shown in Fig. 5. The default Samsung ISP fails to remove the color noise in low-light condition. On the contrary, ReconfigISP yields clean outputs while preserving the details.

4.2. Adapting to Specific Efficiency Constraints

In real-world applications, we may have different efficiency constraints because of limited computational resources and the demand for real-time display. ReconfigISP adjusts the performance-complexity trade-off by introducing an efficiency term in the loss function. Specifically, during ISP architecture search, the loss L is multiplied by $(Lat)^{\beta}$, where Lat stands for the latency of the current ISP. The larger β is, the more efficient pipeline will ReconfigISP select. In particular, ReconfigISP, ReconfigISP-Fast and ReconfigISP-Faster correspond to $\beta = 0$, $\beta = 0.14$ and $\beta = 0.28$, respectively. Note that such a flexibility is not available in traditional ISP pipelines and deep ISP approaches.

The performance and CPU[3] runtime of different ISP frameworks are presented in Table 4. Camera ISP represents the camera processing pipeline of the corresponding dataset. ReconfigISP-Faster achieves up to 50 times speedup compared to ReconfigISP, at the expense of performance. We observe that the acceleration is attributed to fewer computations for denoising. Nevertheless, ReconfigISP-Faster is still better than or comparable to the default camera ISP. The selected ISP architectures are described in the supplementary material.

4.3. Evaluation on Object Detection

To verify the effectiveness of the proposed framework on high-level tasks such as object detection, we collect a dataset, named OnePlus[4], containing several driving scenes

[3]Intel(R) Core(TM) i7-8700 CPU @ 3.20GHz
[4]The dataset is available on the project page.

Figure 6: Qualitative results on OnePlus dataset for object detection (Zoom in for best view).

Table 4: Results of different efficiency constraints. We report CPU runtime to process one megapixel.

Dataset and Metric	SID [5]		S7 ISP [27]	
	PSNR	Time (s)	PSNR	Time (s)
Camera ISP	15.69	-	21.08	-
ReconfigISP	25.65	1.16	23.31	1.53
ReconfigISP-Fast	23.72	0.63	22.70	0.61
ReconfigISP-Faster	20.55	0.049	20.42	0.031

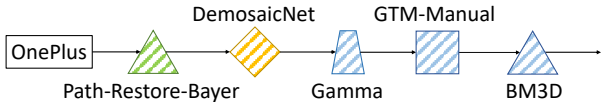


Figure 7: The searched ISP architecture for object detection.

in low-light condition. We use OnePlus 6T A6010 smartphone to take photos, and the sensor type is Sony Exmor IMX519. To achieve low-light constraint, we set ISO=6,400, exposure time=0.01s. With the help of Lightroom, we simultaneously capture RAW and JPEG images. The JPEG images are produced by the default ISP pipeline of Lightroom.

We focus on three classes of object in the street scenes – person, car and bus. We use LabelMe [3] to annotate 141 images, with 50 images for training and the remaining 91 images for evaluation. In the experiments, we adopt YOLOv3 [24] as the detector. For inference, YOLOv3 observes the images generated by ReconfigISP and predicts the objects. As for backpropagation, YOLOv3 passes the gradients back to ReconfigISP to facilitate architecture search. Note that YOLOv3 is not updated in the training process.

The selected ISP architecture is presented in Fig. 7. We observe that this pipeline is significantly different from those for image restoration (see Fig. 3). In particular, a deep model DemosaicNet [11] is used for demosaicing. Denoising is not only performed in the RAW domain, but also conducted in the sRGB domain using BM3D [6]. A global tone mapping method is chosen to enlarge the image contrast. Interestingly, white balance is not adjusted in the whole pipeline, indicating that the color may be less important than noise and contrast, for detection task.

Table 5: Object detection results on OnePlus Dataset.

OnePlus	Person	Car	Bus	mAP
Lightroom	0.318	0.469	0.167	0.318
Software ISP [16]	0.427	0.659	0.458	0.515
ReconfigISP	**0.515**	**0.697**	**0.592**	**0.601**

Quantitative comparisons are shown in Table 5. We report the Average Precision (AP) of each class together with the mean AP (mAP). ReconfigISP is compared with Lightroom and Software ISP [16]. ReconfigISP surpasses the default Lightroom pipeline by a large margin. Compared with Software ISP [16], ReconfigISP achieves better performance on all three classes. We present qualitative results in Fig. 6. ReconfigISP is significantly superior to the other two methods, giving accurate object classes and locations under low-light condition. Such optimized performance is achieved despite the discoloration, recapitulating that color is not as important in object detection task.

4.4. Further Analysis

The Effects of ISP Architecture. To verify the importance of ISP architecture search, we compare our method to a random search baseline. In particular, the ISP architecture is randomly sampled and tuned. The best ISP is chosen among more than 1,000 optimized pipelines. This gives the random baseline an unfair advantage. To have a complete analysis, we swap the ISP architectures optimized for different tasks and observe the performance change. This analysis is conducted on S7 ISP dataset for image restoration and OnePlus dataset for object detection. The corresponding ISP architectures are denoted by ReconfigISP-S7 (Fig. 3) and ReconfigISP-OnePlus (Fig. 7), respectively. After exchanging the ISP architecture, the algorithm parameters are further fine-tuned.

The results are presented in Table 6. On S7 ISP dataset, the optimal architecture surpasses ReconfigISP-OnePlus by nearly 3 dB. On OnePlus detection dataset, the specialized

Table 6: Ablation study on the ISP architecture.

Dataset	S7 ISP [27]		OnePlus
Metric	PSNR	SSIM	mAP
Random Search	23.13	0.6897	0.566
ReconfigISP-S7	**23.31**	**0.7007**	0.352
ReconfigISP-OnePlus	20.55	0.6846	**0.601**

Table 7: Results with different data quantity on SID [5] dataset.

Number of Training Patches	100	500	3,000
U-Net [5]	18.43	24.13	**26.62**
ReconfigISP	**22.73**	**24.98**	25.61

pipeline is superior to the S7 architecture by a large margin. Moreover, the specialized ReconfigISP consistently achieves better performance compared to random search. These results demonstrate that architecture optimization is crucial for ISP, and our ReconfigISP successfully identifies appropriate ISP architectures for diverse target tasks.

The Effects of Proxy Tuning. We investigate the importance of proxy tuning on S7 ISP dataset. Without proxy tuning, the last module "WB-Quadratic" is replaced with "WB-Grayworld", which is a simple white balancing algorithm without any tuned parameters. After optimizing the algorithm parameters for this ISP architecture, the PSNR and SSIM performance are 21.01 dB and 0.6811, respectively. Compared to ReconfigISP-S7 with PSNR 23.31 dB and SSIM 0.7007, the new architecture without proxy tuning suffers a significant performance drop. We attribute this phenomenon to the data distribution gap between different tasks and datasets. When a proxy network fails to imitate the original algorithm, the architecture search becomes less effective. Thus, proxy tuning mechanism is essential to our ReconfigISP framework.

4.5. Comparisons to Deep ISP

We compare our method with a deep ISP that adopts a widely used U-Net architecture [5]. U-Net can be treated as an upper bound as it is a pure deep network solution with around 7 million tunable parameters, whilst ours involves traditional algorithms and the number of tunable parameters is just 226, with 54 architecture parameters and 172 algorithm parameters. In Table 7, we present the quantitative results with different number of training patches[5]. Our method is more robust than U-Net when the data size is small. The searched ISP architecture is the same as that in Fig. 3, except for the case of 100 training patches, where "WB-Quadratic" is replaced with "WB-Whitepatch". When the data size becomes large, U-Net unsurprisingly outperforms the current ReconfigISP (with a pipeline length of five). But this renders an unfair comparison to our method. Qualitative results are shown in Fig. 8. With 100 training patches, U-Net outputs a noisy image with distorted color and exposure. ReconfigISP generates a better result, indicating that our method is more robust to small-scale data. Apart from parameter and data efficiency, ReconfigISP possesses unique advantages of modularity and interpretability, that are not available in U-Net or other deep approaches. Moreover, U-Net cannot flexibly adapt to specific efficiency constraints as we did in Sec. 4.2. A redesign of the architecture is inevitable.

[5]Each training patch has a resolution of 192×192. Data augmentation is used including cropping and flipping.

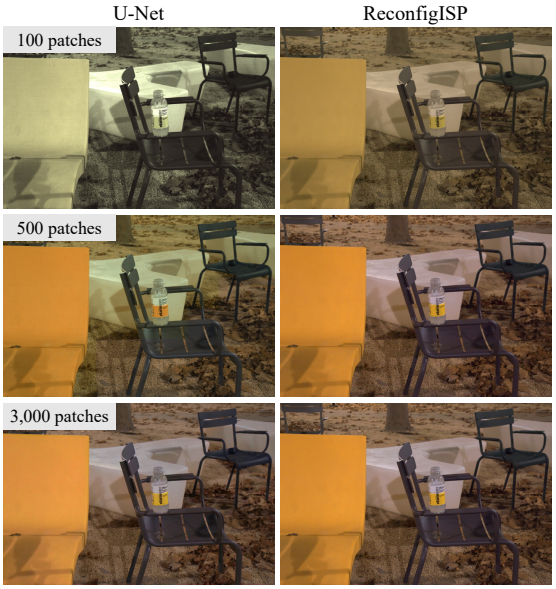


Figure 8: Qualitative comparisons to U-Net [5] with different quantity of training data.

5. Discussion and Conclusion

Tuning an ISP pipeline is notoriously labourious. In this paper, we devise a reconfigurable ISP (ReconfigISP) that can efficiently and flexibly adjust the ISP architecture given a specific task, only by tuning hundreds of architecture weights and algorithm parameters automatically driven by designated loss functions. Experimental results show that ReconfigISP outperforms traditional ISP pipelines in both performance and flexibility. Notably, ReconfigISP maintains the modularity and interpretability of traditional ISP pipelines. Thanks to this unique feature, the ISP architecture search offers additional insights for ISP design and tuning. For instance, deep denoising methods are more robust in RAW domain compared to sRGB domain. In addition, the complexity of denoising algorithms determines the ISP efficiency to a large extent. Lastly, white balancing contributes little to high-level tasks such as object detection.

Acknowledgement. This study is supported under the RIE2020 Industry Alignment Fund – Industry Collaboration Projects (IAF-ICP) Funding Initiative, as well as cash and in-kind contribution from the industry partner(s).

References

[1] Abdelrahman Abdelhamed, Stephen Lin, and Michael S. Brown. A high-quality denoising dataset for smartphone cameras. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018. 5

[2] Bowen Baker, Otkrist Gupta, Nikhil Naik, and Ramesh Raskar. Designing neural network architectures using reinforcement learning. In *Proceedings of the International Conference on Learning Representations*, 2017. 3

[3] Adela Barriuso and Antonio Torralba. Notes on image annotation. *arXiv preprint arXiv:1210.3448*, 2012. 7

[4] Antoni Buades, Bartomeu Coll, and Jean-Michel Morel. A non-local algorithm for image denoising. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2005. 2, 3

[5] Chen Chen, Qifeng Chen, Jia Xu, and Vladlen Koltun. Learning to see in the dark. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018. 2, 5, 6, 7, 8

[6] Kostadin Dabov, Alessandro Foi, Vladimir Katkovnik, and Karen O. Egiazarian. Image denoising with block-matching and 3d filtering. In *Electronic Imaging*, 2006. 2, 3, 7

[7] Linhui Dai, Xiaohong Liu, Chengqi Li, and Jun Chen. Awnet: Attentive wavelet network for image isp. In *Proceedings of the European Conference on Computer Vision Workshops*, 2020. 2

[8] Chao Dong, Chen Change Loy, Kaiming He, and Xiaoou Tang. Image super-resolution using deep convolutional networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2016. 5

[9] Gabriel Eilertsen, Rafal Mantiuk, and Jonas Unger. A comparative review of tone-mapping algorithms for high dynamic range video. *Computer Graphics Forum*, 2017. 3

[10] Pascal Getreuer. Linear methods for image interpolation. *Image Processing On Line*, 2011. 3

[11] Michaël Gharbi, Gaurav Chaurasia, Sylvain Paris, and Frédo Durand. Deep joint demosaicking and denoising. *ACM Transactions on Graphics*, 2016. 1, 2, 3, 7

[12] Shi Guo, Zifei Yan, Kai Zhang, Wangmeng Zuo, and Lei Zhang. Toward convolutional blind denoising of real photographs. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019. 1

[13] Eugene Hsu, Tom Mertens, Sylvain Paris, Shai Avidan, and Frédo Durand. Light mixture estimation for spatially varying white balance. *ACM Transactions on Graphics*, 2008. 1

[14] Yuanming Hu, Hao He, Chenxi Xu, Baoyuan Wang, and Stephen Lin. Exposure: A white-box photo post-processing framework. *ACM Transactions on Graphics*, 2018. 2

[15] Thomas S. Huang, George J. Yang, and Gregory Y. Tang. A fast two-dimensional median filtering algorithm. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 1979. 2, 3

[16] Hakki Can Karaimer and Michael S Brown. A software platform for manipulating the camera imaging pipeline. In *Proceedings of the European Conference on Computer Vision*, 2016. 1, 2, 7

[17] Daniel Khashabi, Sebastian Nowozin, Jeremy Jancsary, and Andrew W Fitzgibbon. Joint demosaicing and denoising via learned nonparametric random fields. *IEEE Transactions on Image Processing*, 2014. 1

[18] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *Proceedings of the International Conference on Learning Representations*, 2015. 5

[19] Hanxiao Liu, Karen Simonyan, and Yiming Yang. Darts: Differentiable architecture search. In *Proceedings of the International Conference on Learning Representations*, 2019. 3, 4

[20] Ali Mosleh, Avinash Sharma, Emmanuel Onzon, Fahim Mannan, Nicolas Robidoux, and Felix Heide. Hardware-in-the-loop end-to-end optimization of camera image processing pipelines. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2020. 2

[21] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in pytorch. In *Proceedings of the Advances in Neural Information Processing Systems Workshops*, 2017. 5

[22] Tobias Plötz and Stefan Roth. Neural nearest neighbors networks. In *Proceedings of the Advances in Neural Information Processing Systems*, 2018. 1

[23] Esteban Real, Alok Aggarwal, Yanping Huang, and Quoc V Le. Regularized evolution for image classifier architecture search. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2019. 3

[24] Joseph Redmon and Ali Farhadi. Yolov3: An incremental improvement. *arXiv preprint arXiv:1804.02767*, 2018. 5, 7

[25] Erik Reinhard, Michael Stark, Peter Shirley, and James Ferwerda. Photographic tone reproduction for digital images. In *Proceedings of the 29th Annual Conference on Computer Graphics and Interactive Techniques*, 2002. 2, 3

[26] Alessandro Rizzi, Carlo Gatta, and Daniele Marini. Color correction between gray world and white patch. In *IS&T/SPIE Electronic Imaging*, 2002. 3

[27] Eli Schwartz, Raja Giryes, and Alex M Bronstein. Deepisp: Toward learning an end-to-end image processing pipeline. *IEEE Transactions on Image Processing*, 2018. 2, 3, 5, 6, 7, 8

[28] Carlo Tomasi and Roberto Manduchi. Bilateral filtering for gray and color images. In *Proceedings of the IEEE International Conference on Computer Vision*, 1998. 2, 3

[29] Ethan Tseng, Felix Yu, Yuting Yang, Fahim Mannan, Karl ST Arnaud, Derek Nowrouzezahrai, Jean-François Lalonde, and Felix Heide. Hyperparameter optimization in black-box image processing using differentiable proxies. *ACM Transactions on Graphics*, 2019. 2

[30] Yi-Ming Wu, Chiou-Shann Fuh, and Jui-Pin Hsu. Color interpolation for single ccd color camera. *IEEE Southwest Symposium on Image Analysis and Interpretation*, 2004. 3

[31] Ke Yu, Xintao Wang, Chao Dong, Xiaoou Tang, and Chen Change Loy. Path-restore: Learning network path selection for image restoration. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2021. 2, 3

[32] Barret Zoph and Quoc V Le. Neural architecture search with reinforcement learning. In *Proceedings of the International Conference on Learning Representations*, 2017. 3